NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

THEORY AND COMPUTATION OF MOVING MESH METHODS FOR SOLVING TIME-DEPENDENT PARTIAL DIFFERENTIAL EQUATIONS

by

Yuhe Ren

B.Sc. Xi'an Jiaotong University 1982

M.Sc. Xi'an Jiaotong University 1987

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in the Department of Mathematics and Statistics

of

Simon Fraser University

SIMON FRASER UNIVERSITY

December, 1991

ISBN 0-315-78229-3

APPROVAL

Name: Yuhe Ren

Degree: Doctor of Philosophy

Title of thesis: **Theory And Computation Of Moving Mesh Methods For Solving Time-Dependent Partial Differential Equations**

Examining Committee: Dr. G. A. C. Graham, Chairman

Dr. Robert D. Russell, Senior Supervisor

Dr. Randall J. LeVeque
Department of Mathematics, University of Washington

Dr. Manfred R. Trummer

Dr. Tao Tang

Dr. Joseph E. Flaherty, External Examiner
Department of Computer Science
Rensselaer Polytechnic Institute

Date Approved: December 9, 1991

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

Theory and Computation of Moving Mesh Methods for Solving Time-Dependent Partial Differential Equations

Author: _____
(signature)

Yuhe Ren
(name)

Oct. 29, 1991
(date)

Abstract

In this thesis, we study the theory and computation of moving mesh methods for solving one dimensional time-dependent partial differential equations. Chapter 1 is a survey of moving mesh methods, wherein we briefly discuss the three kinds of moving mesh methods – the coordinate transformation method, moving finite element methods and moving finite difference methods. In Chapter 2, we investigate various aspects of the moving mesh problem for the solution of partial differential equations in one space dimension. In particular, we study methods based (explicitly or implicitly) upon an equidistribution principle. Equidistribution is shown to be equivalent to the problem of solving a particular PDE for this new computational coordinate system. Implementation of a discrete version of equidistribution to compute a moving mesh corresponds to solving a weak form of the PDE. The stability of equidistribution is discussed, and we argue that stability can be significantly affected by the way in which this solution process is carried out. A simple moving mesh method is constructed using this framework, and numerical examples are given to illustrate its robustness. In Chapter 3, we study the moving finite element (MFE) methods and one moving finite difference method. Of particular interest are the regularization term and gradient weighted function in MFE. Further, we compare the MFE with the moving finite difference method. The last Chapter contains the conclusions and a discussion of some moving mesh problems deserving further study.

Acknowledgements

I would like to deeply thank my senior supervisor, Dr. Robert D. Russell for his guidance, encouragement and patience during the preparation of this thesis.

Thanks also go to Mrs. Sylvia Holmes and Mr. Lixin Liu for their help.

Financial support from Dr. R. D. Russell and Simon Fraser University is greatly appreciated.

Dedication

To my mother and
the memory of my father.

Contents

List of Tables

List of Figures

Chapter 1

A Survey of Moving Mesh Methods

1.1 Method of lines for solving PDEs

The method of lines (MOL) is a semi-discrete method for solving partial differential equations. The method is quite reliable and convenient. The idea of the MOL is simple. We consider the PDE problem

$$u_t = f(u)$$

where f is a nonlinear differential spatial operator. Boundary and initial conditions are given. In the MOL, there are two ways to discretise the PDE. The first one consists of discretizing in temporal t and then solving a boundary value ODE problem by BVP codes like COLSYS, COLCON and AUTO [AMR88]. This approach is called the transverse method of lines. The second scheme involves discretizing in the space x first, and then solving an initial value problem for temporal t by IVP codes, for example LSODI [HI80] and DASSL [PE82]. This approach is called the longitudinal method of lines. At a computational levels, the longitudinal approach has been much more popular than the transverse approach.

The possible advantages of MOL are :

(1) By separating the problems of space and time discretization it is easy to establish stability and convergence for the method used in MOL.

(2) The powerful numerical techniques for solving ODEs, such as dynamically regridding the stepsize and still maintain stability and a desired time integration accuracy, can be directly applied to the PDE case.

(3) Programming effort can be substantially reduced by making use of reliable ODE codes.

(4) By solving the ODEs very accurately one can compare the accuracy and efficiency of different approximations of spatial derivatives.

The possible disadvantages in using MOL are that the reduced ODEs may become very stiff problems and one may lose overall optimization of the method by decoupling the analysis of the space and time discretization [HY76]. The spatial mesh does not change for problems with transient regions like a moving wave front or a shock layer, for which the fine mesh would be needed throughout the domain, and for problems in which the solution becomes very smooth, it may even be desirable to coarsen the mesh.

The moving mesh methods for solving PDEs, which we consider in this thesis are based on the longitudinal method of lines, where standard finite element methods or finite difference methods can be used to approximate the spatial differential terms. However, the spatial mesh is allowed to change with time. This gives the corresponding moving finite element methods or moving finite difference methods in a moving mesh frame, respectively. This allows automatic selection of meshes for both spatial x and temporal t according to the behaviour of the original PDE itself. It is natural to change the spatial mesh smoothly with time. Here we consider the mesh selection strategy with the number of mesh points fixed. Thus, local refinement mesh strategy and how it might interact with the moving mesh strategy, are not considered either.

1.2 Adaptive moving mesh methods

1.2.1 Introduction

The efficiency of a numerical algorithm for solving a class of problems can be critically affected by its computer implementation. Adaptive mesh methods are for instance much more efficient than uniform mesh methods for solving time-dependent partial differential equations with large gradients such as shock waves, propagatied boundary layers *etc.*. There are three main approaches for adaptive mesh methods: the h-refinement methods, which add or delete mesh points according to the profile of the solution and control the mesh points by the local errors of the solution; the p-refinement methods, which alter the order of the numerical method to fit the local solution characteristics; and the moving mesh methods, in which a fixed number of mesh points move automatically to minimize the errors of the solution. Further, we may use a combination of the moving mesh methods and the local refinement methods [AF86b1], [AF86b2].

Adaptive mesh methods for solving ODEs were surveyed in [RU79]. The purpose of this chapter is to outline a variety of the moving mesh methods for solving partial differential equations. Despite their efficiency for solving PDEs, the moving mesh methods are just in their beginning, especially as regards their theory. We discuss the current numerical methods in a general way and avoid giving firm conclusions. We consider moving finite element methods, moving finite difference methods and coordinate transformation methods borrowed from adaptive mesh methods for solving ODEs. Both parabolic and hyperbolic problems are studied.

1.2.2 Coordinate transformation method

Based on the corresponding method used in solving ODEs [WH79], White [WH82] applies the coordinate transformation method to solve first-order systems of partial differential

equations of the form

$$A(u,x,t)u_t + B(u,x,t)u_x = C(u,x,t)$$

where $(x,t) \in [a,b] \times [0,\infty)$, and appropriate boundary and initial conditions are given. For good accuracy a non-uniform mesh must be selected in the region of rapid solution change. The goal of the coordinate transformation is to transform the solution such that in the new coordinate a uniform mesh can be used.

White introduces the arclength transformation for initial/boundary-value problems. He changes from (x,t)-coordinates to new computational coordinates (s,T), where s is the arclength of the solution. A pair of computational coordinates is defined by

$$s = \int_a^x [1 + \|u_\xi(\xi,t)\|_2^2]^{\frac{1}{2}} d\xi/\theta \tag{1.2.1}$$

and

$$T = t, \tag{1.2.2}$$

where θ is the total arclength of the solution defined by

$$\theta = \int_a^b [1 + \|u_\xi(\xi,t)\|_2^2]^{\frac{1}{2}} d\xi. \tag{1.2.3}$$

For simplicity, we consider the scalar differential equation case. The transformed problem in arclength coordinates is now given by

$$x_s A u_T + [B - x_T A]u_s = x_s C, \tag{1.2.4}$$

$$x_s^2 + \|u_s\|_2^2 - \theta^2 = 0, \tag{1.2.5}$$

and

$$\theta_s = 0. \tag{1.2.6}$$

The solution of the transformed problems (1.2.4), (1.2.5) and (1.2.6) can be approximated on a uniform mesh because the arclength transformation automatically smooths out regions of the rapid change. The well-known Box scheme [KE71] is used by White in actual

computation, viz.

$$(u)_{j+\frac{1}{2}}^{k+\frac{1}{2}} \approx \frac{1}{4}[u_j^k + u_{j+1}^k + u_j^{k+1} + u_{j+1}^{k+1}],$$

$$(u_s)_{j+\frac{1}{2}}^{k+\frac{1}{2}} \approx \frac{1}{2}[(\frac{u_{j+1}^k - u_j^k}{\triangle s}) + (\frac{u_{j+1}^{k+1} - u_{j+1}^k}{\triangle s})],$$

$$(u_T)_{j+\frac{1}{2}}^{k+\frac{1}{2}} \approx \frac{1}{2}[(\frac{u_j^{k+1} - u_j^k}{\triangle T_k}) + (\frac{u_{j+1}^{k+1} - u_{j+1}^k}{\triangle T_k})].$$

These give

$$\begin{aligned} & \tfrac{1}{2\triangle s}[x_{j+1}^{k+1} - x_j^{k+1} + x_{j+1}^k - x_j^k] A \tfrac{1}{2\triangle T_k}[u_{j+1}^{k+1} - u_{j+1}^k + u_j^{k+1} - u_j^k] \\ + \; & (B - \tfrac{1}{2\triangle T_k}[x_{j+1}^{k+1} - x_{j+1}^k + x_j^{k+1} - x_j^k] A) \tfrac{1}{2\triangle s}[u_{j+1}^{k+1} - u_j^{k+1} + u_{j+1}^k - u_j^k] \\ = \; & \tfrac{1}{2\triangle s}[x_{j+1}^{k+1} - x_j^{k+1} + x_{j+1}^k - x_j^k] C, \end{aligned}$$

and

$$(\frac{1}{2\triangle s}[x_{j+1}^{k+1} - x_j^{k+1} + x_{j+1}^k - x_j^k])^2 + \|\frac{1}{2\triangle s}[u_{j+1}^{k+1} - u_j^{k+1} + u_{j+1}^k - u_j^k]\|_2^2 - (\frac{1}{2}(\theta^{k+1} + \theta^k))^2 = 0.$$

And then Newton's method, for example, can be used to solve these nonlinear equations.

The resulting systems generally turn out to be sensitive and difficult to solve numerically. Although the reasons are not entirely clear, note that the transformed equation is nonlinear even if the original one is linear. Furthermore, sometimes the method has physically meaningless solutions when the solution becomes multivalued. At such a point, shock conditions should be used in order to approximate the correct solution.

Dwyer, Kee and Sanders studied the coordinate transformation method for problems in fluid mechanics and heat transfer, in which meshes were controlled by the gradient and the curvature of the solution [DKS80].

1.2.3 Moving finite difference methods

Consider the time-dependent partial differential equation

$$u_t = f(u), \tag{1.2.7}$$

where f is a spatial differential operator. In the Lagrangian frame, meshes move continuously with time, and the original equation (1.2.7) can be rewritten in the form

$$\dot{u}_i - u_{x_i}\dot{x}_i = f(u_i) \tag{1.2.8}$$

for the i-th mesh, where $\dot{u}_i = u_t(x_i) + u_{x_i}\dot{x}_i$ is a total derivative at the i-th mesh point, and u_{x_i} is approximated by a standard difference central

$$u_{x_i} = \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}}.$$

This is a stable scheme used in moving finite difference methods.

We consider now the moving equation for the moving mesh $x_i(t)$. First of all, we introduce the moving mesh equations derived by Hyman [HY82] and Petzold [PE87].

These schemes have two computational stages: first, choosing a moving mesh based on minimizing the time rate of change of the solution in the Lagrangian coordinates, and second, calculating the mesh based on the deBoor's equidistribution algorithm [BO73], *i.e.*, *static regriding.*

Hyman studied the PDE case

$$u_t = f(u, u_x, u_{xx}).$$

The system case

$$F(u_t, u_x, u_{xx}) = 0,$$

was studied by Petzold.

For simplicity, here we consider the scalar case

$$u_t = f(u, u_x, u_{xx}). \tag{1.2.9}$$

In the Lagrangian frame, equation (1.2.9) becomes

$$\dot{u} - u_x\dot{x} = f(u, u_x, u_{xx}). \tag{1.2.10}$$

The mesh velocity $\dot{x}$ is chosen to minimize the time rate of change of u and x in this Lagrangian frame. Thus the moving mesh equation is

$$\min_{\dot{x}}[\|\dot{u}\|^2 + \alpha\|\dot{x}\|^2] = \min_{\dot{x}}[\sum \dot{u}^2 + \alpha\dot{x}^2] = \min_{\dot{x}}[\sum (f(u) + u_x\dot{x})^2 + \alpha\dot{x}^2] \tag{1.2.11}$$

where α is a positive scaling parameter. This results in the moving mesh equation

$$\dot{x} = -\frac{f(u, u_x, u_{xx})u_x}{\alpha + u_x u_x}. \tag{1.2.12}$$

For the hyperbolic problem, it is natural that a mesh point moves along the characteristic direction of the equation if the parameter α is zero, and in this case the meshes move along the direction of minimizing the rate of change of the solution. For example, if

$$f(u) = au_x$$

we have moving mesh equation

$$\dot{x} = a$$

which is the characteristic equation.

It is interesting that if we minimize the time rate of change of u and x in the Lagrangian frame with respect to the velocity $\dot{u}$ of the solution instead of the velocity $\dot{x}$ of the mesh in equation (1.2.11), we obtain the same moving mesh equation as (1.2.12).

A practical difficulty is that the mesh points derived by equation (1.2.12) may easily cross one another. An extra regularization term is needed to prevent such mesh point crossings. For this purpose, Petzold has used the penalty function

$$\lambda(\|\frac{\dot{x}_j - \dot{x}_{j-1}}{x_j - x_{j-1}}\|_2^2 + \|\frac{\dot{x}_{j+1} - \dot{x}_j}{x_{j+1} - x_j}\|_2^2) \tag{1.2.13}$$

and minimized the new objective function,

$$\min_{\dot{x}_j}[\|\dot{u}_j\|_2^2 + \alpha\|\dot{x}_j\|_2^2 + \lambda(\|\frac{\dot{x}_j - \dot{x}_{j-1}}{x_j - x_{j-1}}\|_2^2 + \|\frac{\dot{x}_{j+1} - \dot{x}_j}{x_{j+1} - x_j}\|_2^2)] \tag{1.2.14}$$

where λ is a positive parameter, which leads to

$$\alpha\dot{x}_j + \dot{u}_j u_{x_j} + \lambda[\frac{\dot{x}_j - \dot{x}_{j-1}}{(x_j - x_{j-1})^2} - \frac{\dot{x}_{j+1} - \dot{x}_j}{(x_{j+1} - x_j)^2}] = 0. \tag{1.2.15}$$

The parameters α, and λ are chosen as 1.0 and 0.2, respectively, in [PE87]. The penalty term acts to diffuse the mesh velocity since the penalty term in equation (1.2.15) approximates $\lambda(\dot{x})_{xx}$ for a uniform mesh spacing, $x_j - x_{j-1} = x_{j+1} - x_j$. Although mesh points can still cross, (1.2.15) is quite a robust moving mesh equation.

For reliable computation, deBoor's algorithm can be required to adjust the mesh position after a time step obtained by equation (1.2.15). Such a so-called dual reconnecting mesh strategy is described in [HY82], [HY84], and [PE87]. This strategy equidistributes the mesh based on the first and second derivatives of the solution, such that the mesh points satisfy

$$h_i\|u_x\| + h_i^2\|u_{xx}\| \leq TOL \tag{1.2.16}$$

where TOL is a user-defined error tolerance.

It is possible to interpolate the solution from the mesh resulting from (1.2.12) directly and use it to find a new the equidistributing mesh from atisfying (1.2.16). Thus, this requires interpolating between every mesh point at each step. To avoid the large computational expense for this approach, Hyman and Petzold use the dual reconnecting mesh approach, which is a compromise between choosing the best mesh, and avoiding needless interpolations. In particular, two reference meshes are computed at the beginning of the each time level. The first reference mesh and the solution are obtained by solving equations (1.2.10) and (1.2.15). The second reference mesh is redistributed to satisfy equation (1.2.16). Therefore, the second reference mesh divides the space into reference zones. Mesh points are added or deleted so that there is exactly one mesh point per reference zone, and mesh points at the edges of zones which are too close to other mesh points are moved apart. This scheme has worked well in practice, and on most time steps it requires few interpolations between mesh points.

To apply this approach to practical problems, they also derive the scaled moving mesh equation from minimizing a weighted l_2 norm in (1.2.11), where $\|\dot{u}_j\|_2^2 = \sum_{i=1}^{NP}(\frac{\dot{u}_{i,j}}{w_i})^2$ and $\|\dot{x}_j\|_2^2 = (\frac{\dot{x}_j}{w_{NP+1}})^2$, which NP is the number of PDEs in the original system. The scaled

moving mesh equation at x_j is

$$\sum_{i=1}^{NP} \frac{\dot{u}_{i,j} u_{x_i,j}}{w_i^2} + \alpha \frac{\dot{x}_j}{w_{NP+1}^2} + \lambda[\frac{\dot{x}_j - \dot{x}_{j-1}}{(x_j - x_{j-1})^2} - \frac{\dot{x}_{j+1} - \dot{x}_j}{(x_{j+1} - x_j)^2}] = 0$$

where w_i are weights to be chosen by

$$w_i = \max\{|u_{max_i} - u_{min_i}|, floor(i)\}.$$

Here u_{max_i} is the maximum of the i-th component of u_i over all mesh points, and u_{min_i} is the minimum of u_i over all mesh points.

Another moving finite difference method is proposed by Dorfi and Drury [DD87]. A method with smoothing procedure for both spatial and temporal variables is derived based on the equidistribution principle for solving one-dimensional initial value problems.

Equidistribution for the mesh points requires that

$$\int_{x_i}^{x_{i+1}} M(\xi, t) d\xi = \frac{1}{N}\theta(t) \qquad 0 \le i \le N, \tag{1.2.17}$$

where $\theta(t) = \int_a^b M(\xi, t) d\xi$, and $M \ge 0$ a so-called monitor function. Dorfi and Drury use the arclength monitor defined by

$$M(x, u) = \sqrt{1 + u_x^2}. \tag{1.2.18}$$

Using the forward difference scheme for u_x, the discrete arclength monitor at each subinterval $[x_i, x_{i+1}]$ is

$$M_i = \sqrt{1 + \frac{(u_{i+1} - u_i)^2}{(x_{i+1} - x_i)^2}}. \tag{1.2.19}$$

The fundamental form of a moving mesh equation is derived by considering the point concentration

$$n_i = \frac{1}{x_{i+1} - x_i} \tag{1.2.20}$$

and having

$$n_i \propto M_i.$$

If we consider two neighboring subintervals for equidistribution,

$$\int_{x_{i-1}}^{x_i} M(\xi,t)d\xi = \int_{x_i}^{x_{i+1}} M(\xi,t)d\xi, \quad \text{for} \quad i = 1,\cdots,N. \tag{1.2.21}$$

By applying the mid-point quadrature rule, we have

$$(x_i - x_{i-1})M_{i-1} = (x_{i+1} - x_i)M_i, \quad \text{for} \quad (1 \le i \le N).$$

We can rewrite this in the form

$$\frac{n_{i-1}}{M_{i-1}} = \frac{n_i}{M_i}.$$

Further, Dorfi and Drury introduce a penalty function for smoothing both spatial and temporal meshes based on the stability condition

$$\frac{k}{k+1} \le \frac{n_{i+1}}{n_i} \le \frac{k+1}{k} \tag{1.2.22}$$

where k is chosen to be 1 or 2. We also require

$$n_i \propto \sum_j M_j (\frac{k}{k+1})^{|i-j|}, \tag{1.2.23}$$

where the right-hand-side is a smoothing kernel which is a Green's function associated with the difference operator

$$1 - k(k+1)\delta^2,$$

where δ is a centered difference operator. So smoothing in space is

$$\tilde{n}_i = n_i - k(k+1)(n_{i+1} - 2n_i + n_{i-1}) \propto M_i. \tag{1.2.24}$$

To smooth the temporal variable, let

$$\hat{n}_i = \tilde{n}_i + \frac{\tau}{\Delta t}(\tilde{n}_i - \tilde{n}_i^{(old)}) \propto M_i$$

where Δt is the time step, and τ is a positive time scale which depends on the problem.

Finally, we obtain moving equidistribution by setting

$$(\tilde{n}_{i-1} + \tau\frac{d\tilde{n}_{i-1}}{dt})/M_{i-1} = (\tilde{n}_i + \tau\frac{d\tilde{n}_i}{dt})/M_i. \tag{1.2.25}$$

In equation (1.2.25), the monitor function M is implicitly unchanged from the last time to the updated time over the time step τ. The parameter τ acts as a delay factor.

Verwer *et al.* studied this method [VBFZ88]. The inequality (1.2.22) is necessary for stability of the scheme since it prevents meshes from crossing each other. Blom and Verwer [BV89] compared the arclength monitor with the curvature monitor for the moving mesh. The equations (1.2.25) are found to be much easier to solve with the arclength monitor than with the curvature monitor, which uses an approximation to the second derivative of the solution.

Another way of controlling meshes involves the concept of a mesh function, which was first introduced by Hyman to dynamically adjust the mesh locations [HL86]. There are many ways to define the mesh function. Hyman and Larrouturou use one that satisfies an ordinary differential equation

$$\dot{m}_{i+\frac{1}{2}} = \frac{\beta}{\tau}(\bar{m} - m_{i+\frac{1}{2}}), \tag{1.2.26}$$

where $m_{i+\frac{1}{2}}$ is the value of the mesh function at $x_{i+\frac{1}{2}} = \frac{1}{2}(x_i + x_{i+1})$, $\bar{m}$ is the average value of the mesh function m, and β determines the relaxation time with respect to the timescale τ. Adjerid and Flaherty use a similar approach [AF86b2].

Madsen adopts the idea of a mesh function when considering the system of partial differential equations

$$U_t = F(U), \quad \text{for} \quad x_l < x < x_r, \ t > t_0,$$

where $U = (u_1, \cdots, u_{NDPE})^T$ and $F = (f_1, \cdots, f_{NDPE})^T$. Using the method of lines, the PDE system is approximated with the semi-discrete ODE system

$$\frac{du_{k,i}}{dt} = f_{k,i} + (u_{k,i})_x \frac{dx_i}{dt}, \quad \text{for} \quad k = 1, \cdots, NPDE.$$

Mesh points satisfy the spatial ODE

$$\frac{dx_i}{dt} = \textit{mesh function}, \quad \text{for} \quad i = 1, \cdots, NPTS.$$

Madsen [MA84] defines a mesh function with the property:

$$\lim_{h_i \to \infty} m(i) = 0,$$

where $m(i)$ is the mesh function value associated with the i-th mesh interval and $h_i = x_{i+1} - x_i$. If x_l and x_r are assumed to be the left and right fixed boundaries, respectively, then

$$\sum_{i=1}^{NPTS-1} h_i = x_r - x_l,$$

both $\dot{x}_r$ and $\dot{x}_l$ are zero. Differentiating above equation with respect to t, $\sum_{i=1}^{NPTS-1} \frac{dh_i}{dt} = 0$. The moving mesh equation is defined by

$$\frac{dh_i}{dt} = \frac{dx_{i+1}}{dt} - \frac{dx_i}{dt} = M - m(i), \tag{1.2.27}$$

where

$$M = [\sum_{i=1}^{NPTS-1} m(i)]/(NPTS - 1)$$

is the average mesh function value over all of the mesh zones. Madsen uses the five different mesh functions

$$m_1(i) = \sum_{k=1}^{NPDE} W_k |(f_k)_h - (f_k)_{2h}|/T_1, \tag{1.2.28}$$

$$m_2(i) = \sum_{k=1}^{NPDE} W_k |u_{k,i+1} - u_{k,i}|/T_2, \tag{1.2.29}$$

$$m_3(i) = \sum_{k=1}^{NPDE} W_k |(u_x)_{k,i+1} - (u_x)_{k,i}|/T_3, \tag{1.2.30}$$

$$m_4(i) = \sum_{k=1}^{NPDE} W_k |K_{k,i+1} - K_{k,i}|/T_4, \tag{1.2.31}$$

$$m_5(i) = |x_{i+1} - x_i|/T_5, \tag{1.2.32}$$

where $(f_k)_h$ and $(f_k)_{2h}$ represent the right hand side of the k-th PDE evaluated for the current step size h and $2h$, respectively. W_k is a user-specified component weight, $K_{k,i}$ is the curvature of the k-th solution component at x_i and each T_l is chosen so that $\sum_{l=1}^{5} m_l(i) = 1$.

The first mesh function m_1 is designed to ensure that the right-hand sides of the PDEs are well approximated, $m_2(i), m_3(i)$ and $m_4(i)$ are designed to control the amount of change of the solution u according to first derivative and curvature of solution and $m_5(i)$ is designed to control excessive adjacent mesh distortion. In practice, the total mesh function may be useful as a linear combination of the individual mesh functions, *i.e.*,

$$m(i)_{total} = Am_1(i) + Bm_2(i) + Cm_3(i) + Dm_4(i) + Em_5(i) \tag{1.2.33}$$

where A, B, C, D, and E are user specified constants. The scheme contains many problem-dependent parameters, viz. W_k, T_i, A, B, C, D, E. Although extensive tests of this method seem not to have been carried out, preliminary tests appear encouraging. For example, the problem of two opposite travelling wave pulses is tested with $A = 1, B = 0, C = 0, D = 1, E = 1$ and mesh points $N = 100$, and reasonable results are obtained.

Verwer, Blom and Sanz-Serna use static regridding to obtain the reference mesh, then solve the PDE described by the Lagrangian frame [VBS88] and [BSV88]. Their technique is the so-called 'intermediate' technique between static regridding methods where meshes remain fixed for intervals of time, and dynamic moving mesh methods, where the mesh movement and the PDE integration are fully coupled.

To solve

$$u_t = L(u), \tag{1.2.34}$$

where L is a linear or nonlinear spatial differential operator, they use the Lagrangian time-stepping scheme

$$\begin{aligned}
&[\theta(x_{i+1}^{n+1} - x_{i-1}^{n+1}) + (1-\theta)(x_{i+1}^{n} - x_{i-1}^{n})](\tfrac{u_i^{n+1} - u_i^n}{\tau}) \\
&-[\theta(u_{i+1}^{n+1} - u_{i-1}^{n+1}) + (1-\theta)(u_{i+1}^{n} - u_{i-1}^{n})](\tfrac{x_i^{n+1} - x_i^n}{\tau}) \\
&= \theta(x_{i+1}^{n+1} - x_{i-1}^{n+1})L_{h,i}(u^{n+1}) + (1-\theta)(x_{i+1}^{n} - x_{i-1}^{n})L_{h,i}(u^n),
\end{aligned} \tag{1.2.35}$$

where τ is the time step. For $\theta = \frac{1}{2}$, (1.2.35) is the Lagrangian Crank-Nicolson scheme; for $\theta = 1$, it becomes the Lagrangian Implicit Euler scheme. The 'intermediate' approach

has two successive computational stages. The first is the mesh prediction stage which computes a mesh at the forward (n+1)-th level. Then we get the solution on a fixed mesh by the Implicit Euler step, and use the deBoor's regridding algorithm to generate the final mesh at the (n+1)-th level. The second stage is the integration stage which computes the solution u_i^{n+1} by the Lagrangian Crank-Nicolson scheme with $\theta = \frac{1}{2}$. The PDEs (1.2.35) are computed two times for $\theta = \frac{1}{2}$ and $\theta = 1$ in the approach.

Two monitors are introduced in the implementation. The time error monitor controls the time-step selection and the space error monitor adapts the number of moving mesh points.

Smooke and Koszykowski develop a fully adaptive method, which first discretizes in time t, then solves the boundary value problems [SK83]. Meshes are derived from the mesh monitor at discrete time levels. The scheme interpolates values of the solution from the old meshes to the new meshes. They do not couple the calculation of the meshes and the solution of the PDE. The details are as follows: Consider the partial differential equation

$$u_t = f(x, t, u, u_x, u_{xx}), \quad a < x < b, \ t > 0. \tag{1.2.36}$$

Here they use a Backward-Euler scheme to approximate the time derivative. The equation (1.2.36) becomes the boundary value problems,

$$\frac{u^{n+1}(x) - u^n(x)}{\Delta t^{n+1}} = f(x, t^{n+1}, u^{n+1}, u_x^{n+1}, u_{xx}^{n+1}) + e^{n+1}(x) \tag{1.2.37}$$

where the time step $\Delta t^{n+1} := t^{n+1} - t^n$, and $e^{n+1}(x)$ is the discretization error derived by the Backward-Euler scheme

$$e^{n+1}(x) = [\frac{\partial^2 u(x,\xi)}{\partial t^2}]\Delta t^{n+1}/2, \quad \xi \in [t^n, t^{n+1}]. \tag{1.2.38}$$

Ignoring the discretization error, we can rewrite (1.2.37) in the form

$$f(x, t^{n+1}, u^{n+1}, u_x^{n+1}, u_{xx}^{n+1}) - \frac{u^{n+1}(x)}{\Delta t^{n+1}} = -\frac{u^n(x)}{\Delta t^{n+1}}. \tag{1.2.39}$$

Many researchers have used the adaptive mesh method for solving boundary value problems. For example, White has chosen the arclength as a monitor to equidistribute mesh [WH79]. Pereyra and Sewell have obtained a mesh based on equidistributing the local truncation error [PS75]. BVP-ODE solvers such as COLSYS, COLNEW and COLPAR involve adaptive mesh strategies.

For solving BVP-ODE (1.2.39), we realize that an interpolation scheme is required to obtain the solution at time level n at the mesh point x_j^{n+1}. For this purpose, Smooke and Koszykowski use linear interpolation. Specifically consider the mesh points at two consecutive time levels in Fig. 1.1 and assume that the mesh point x_j^{n+1} lies in an interval $[x_k^n, x_{k+1}^n]$ from the n-th level. To obtain u_j^n, they use linear interpolation

$$u_j^n = u_k^n + (\frac{u_{k+1}^n - u_k^n}{x_{k+1}^n - x_k^n})(x_j^{n+1} - x_k^n) \tag{1.2.40}$$

or a corresponding expression if x_j^{n+1} is closer to x_{k+1}^n.

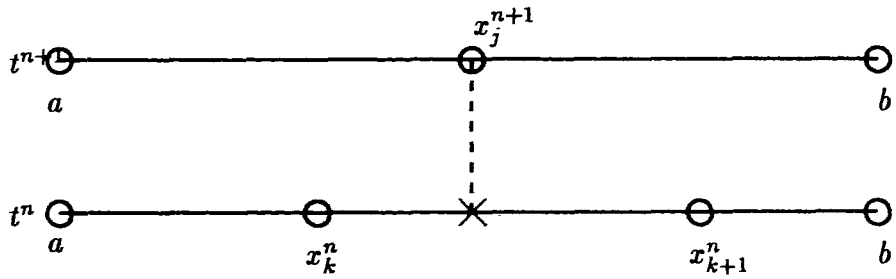

Figure 1.1: Interpolation points at two time levels

Finally, they derive the fully discrete adaptive scheme

$$\begin{aligned} & f(x_j^{n+1}, t^{n+1}, u_j^{n+1}, u_x^{n+1}(x_j), u_{xx}^{n+1}(x_j)) - \frac{u_j^{n+1}}{\Delta t^{n+1}} \\ & = -\frac{1}{\Delta t^{n+1}}(u_k^n + (\frac{u_{k+1}^n - u_k^n}{x_{k+1}^n - x_k^n}))(x_j^{n+1} - x_k^n), \end{aligned} \tag{1.2.41}$$

with the appropriate initial and boundary conditions. But one can not solve the above nonlinear equation, since the mesh point x_j^{n+1} at the (n+1)-st level is still unknown. They use an equidistribution strategy to obtain all meshes prior to the (n+1)-st level. To get the meshes at the (n+1)-st level, they guess mesh points $\{x_j^{n+1}\}$ moving along the simple ODE equation

$$\frac{dx_j}{dt} = \frac{x_j^n - x_j^{n-1}}{\triangle t^n} \tag{1.2.42}$$

with initial condition $x_j(0) = x_j^{n-1}$. Integrating from time 0 to t yields

$$x_j(t) = (\frac{x_j^n - x_j^{n-1}}{\triangle t^n})t + x_j^{n-1} \tag{1.2.43}$$

or so integrating from t^n to t^{n+1}, we have

$$x_j^{n+1} = (\frac{x_j^n - x_j^{n-1}}{\triangle t^n})\triangle t^{n+1} + x_j^n. \tag{1.2.44}$$

This is a linearly extrapolated equation using time levels n and $n-1$.

Though this scheme generally works quite well, there are two major difficulties which occur when an extrapolation scheme is used to move the mesh. The re-ordering of meshes if mesh points cross one another is the first. The second occurs when mesh points are extrapolated out of the spatial domain $[a, b]$.

Larrouturou considers a very simple moving mesh method for solving the flame propagation problem

$$T_t = T_{xx} + \Omega(T, Y), \tag{1.2.45}$$

$$Y_t = Y_{xx}/Le - \Omega(T, Y), \tag{1.2.46}$$

where $x \in [x_0, x_N]$ and $\Omega(T, Y) = \frac{\beta^2}{2Le} Y \exp(-\frac{\beta(1-T)}{1-\alpha(1-T)})$ is the normalized reaction rate. Here, the *Lewis* number Le of the reactant, the reduced activation energy β of the reaction and the nondimensional heat-release parameter α are positive constants. The initial and boundary conditions are given. Larrouturou supposes the meshes move continuously as a rigid body to catch the flame front [LA89], that is, meshes move at each time t with

the same time-dependent velocity $V(t)(= \dot{x}_j(t))$. In this simplified Lagrangian frame, the flame-propagation becomes

$$T_t = T_{xx} + \Omega(T,Y) + V(t)T_x, \tag{1.2.47}$$

$$Y_t = Y_{xx}/Le - \Omega(T,Y) + V(t)Y_x. \tag{1.2.48}$$

The meshes are chosen by an equidistribution strategy with a physical qualitative monitor, for example, the temperature T.

Suppose

$$\int_{x_0}^{x_N} T(x,t)dx = constant. \tag{1.2.49}$$

Integrating equation (1.2.47) from x_0 to x_N yields

$$\frac{d}{dt}\int_{x_0}^{x_N} Tdx = T_x(x_N) - T_x(x_0) + V(t)[T(x_N) - T(x_0)] + \int_{x_0}^{x_N} \Omega(T,Y)dx. \tag{1.2.50}$$

Using equation (1.2.49), we obtain the mesh velocity

$$V(t) = [-\int_{x_0}^{x_N} \Omega(T,Y)dx + T_x(x_0) - T_x(x_N)]/[T(x_N) - T(x_0)]. \tag{1.2.51}$$

A simple explicit finite difference scheme is used to approximate the PDE. In order to better adapt the mesh points to the solution profiles, the new meshes are computed by equidistributing the mesh monitor. Then the solutions at the new meshes are obtained by interpolation from the old meshes. Larrouturou proposes a conservative interpolation which is used instead of linear interpolation. This implementation is much simpler: Let $T^{(old)}$ be the piecewise constant function defined on the interval $[x_0, x_N]$ by

$$T^{(old)} = T_i^{(old)}, \qquad \text{if} \quad x \in [x_{i-\frac{1}{2}}^{(old)}, x_{i+\frac{1}{2}}^{(old)}], \tag{1.2.52}$$

and consider

$$E(x) = \int_{x_0}^{x} T^{(old)}(\xi)d\xi, \tag{1.2.53}$$

where $E(x)$ is a continuous piecewise linear function on $[x_0, x_N]$. Then the new solution for the temperature T in the interval $[x_{j-\frac{1}{2}}^{(new)}, x_{j+\frac{1}{2}}^{(new)}]$ is given by

$$T_j^{(new)} = \frac{E(x_{j+\frac{1}{2}}^{(new)}) - E(x_{j-\frac{1}{2}}^{(new)})}{x_{j+\frac{1}{2}}^{(new)} - x_{j-\frac{1}{2}}^{(new)}}. \tag{1.2.54}$$

It has been shown that conservative interpolation preserves the positivity and monotonicity of the solution and the properties of linear interpolation [LA89].

1.2.4 Moving finite element methods

Moving finite element methods are another class of methods which have been quite successful in solving time-dependent partial differential equations with shocks or large gradients. Miller and Miller first introduced the moving finite element method in 1981 [MM81]. Subsequently, many researchers have studied further variants of moving finite element methods. Baines *et al.* studied the "local" moving finite element method [BW88], [JWB88]; Herbst *et al.* generalized the moving finite element [MM81] to a moving Petrov-Galerkin methods for solving transport equations [HMS82]; Adjerid and Flaherty proposed the adaptive moving finite element method [AF86b1], [AF86b2]; and Mosher used a variable node finite element method [MO85]. We review these moving finite element methods in the following sections.

A. Moving finite element methods

Consider the time-dependent partial differential equation

$$u_t = L(u), \quad a < x < b, \ t > 0. \tag{1.2.55}$$

In the moving mesh frame or Lagrangian frame, meshes depend on the time and profile of the solution of the partial differential equation. Consider the partition

$$\Pi : a = x_0 < \cdots < x_i(t) < x_{i+1}(t) < \cdots < x_{N+1} = b. \tag{1.2.56}$$

Miller and Miller introduce the piecewise linear moving finite element to approximate the solution, so

$$U(x,t) = \sum_{i=1}^{N} U_i(t)\alpha_i(x), \tag{1.2.57}$$

where α_i are the standard piecewise linear basis functions

$$\alpha_i = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & \text{for } x_{i-1} \le x \le x_i, \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & \text{for } x_i \le x \le x_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.2.58}$$

and U_i are the amplitudes of solution at x_i. Differentiating equation (1.2.57) with respect to t gives

$$\dot{U}(x,t) = \sum_{i=1}^{N} \dot{U}_i(t)\alpha_i + \beta_i \dot{x}_i, \tag{1.2.59}$$

where

$$\beta_i = \begin{cases} -u_{x_i}\alpha_i & \text{for } x_{i-1} \le x \le x_i, \\ -u_{x_{i+1}}\alpha_i & \text{for } x_i \le x \le x_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{1.2.60}$$

Here β_i are discontinuous piecewise linear functions. The divided difference

$$m_i := \frac{u_i - u_{i-1}}{x_i - x_{i-1}}$$

approximates the first derivative of the solution on $[x_{i-1}, x_i]$, *i.e.* $m_i \approx u_x$, for $x \in [x_{i-1}, x_i]$.

We minimize

$$\| \dot{U} - L(U) \|_2^2 \tag{1.2.61}$$

with respect to the velocities of meshes x_i and the velocities of the solution U_i. This **Least Square** approach leads to a system of Ordinary Differential Equations

$$\sum_{i=1}^{N} < \alpha_i, \alpha_j > \dot{U}_i + < \beta_i, \alpha_j > \dot{x}_i = < L(U), \alpha_j >, \qquad j = 1, \ldots, N. \tag{1.2.62}$$

$$\sum_{i=1}^{N} < \alpha_i, \beta_j > \dot{U}_i + < \beta_i, \beta_j > \dot{x}_i = < L(U), \beta_j >, \qquad j = 1, \ldots, N. \tag{1.2.63}$$

where $< \cdot, \cdot >$ is the standard L_2 - inner product. The $U_i(0)$ are obtained from the initial conditions for the partial differential equation, and a uniform mesh are usually chosen for the initial values $x_i(0)$. For fixed meshes, this is the same as the standard finite element method with a piecewise linear basis.

Let $Y := (U_1, x_1, \ldots, U_i, x_i, \ldots, U_N, x_N)^T \in R^{2N}$. We write equations (1.2.62) and (1.2.63) in matrix form,

$$A(Y)\dot{Y} = G(Y), \qquad \text{for} \quad t > 0. \tag{1.2.64}$$

Here $A(Y)$ is the mass-matrix with block-tridiagonal form, where

$$G(Y) = (g_1, g_2, \cdots, g_{2N-1}, g_{2N})^T,$$

$$g_{2j-1} =< \alpha_j, L(u) >,$$

and

$$g_{2j} =< \beta_j, L(u) > .$$

Wathen and Baines introduce the concept of the "local" moving finite element and show that the block-tridiagonal matrix A can be represented as

$$A = M^T X M, \tag{1.2.65}$$

with

$$M = \begin{pmatrix} 1 & 1 \\ -m_i & -m_{i+1} \end{pmatrix}, \tag{1.2.66}$$

and

$$X = \begin{pmatrix} \frac{1}{3}\triangle x_i & 0 \\ 0 & \frac{1}{3}\triangle x_{i+1} \end{pmatrix}. \tag{1.2.67}$$

Hence it is easy to show that $A(Y)$ is a singular if two situations occurred. The first is so-called parallelism ($m_i = m_{i+1}$, for some $i \in \{1, \cdots, N\}$). Geometrically, this implies that three neighboring points (x_{i-1}, u_{i-1}), (x_i, u_i) and (x_{i+1}, u_{i+1}) are located on the same straight line, and the curvature u_{xx} at the point (x_i, u_i) is zero. The second is the degeneracy

of A whenever two mesh points come too close together or cross one another; in this case A becomes ill-conditioned and numerically singular. The mesh crossing can be eliminated by reducing the time step [HMW86], or by deleting mesh points that are too close [WB85].

In order to overcome above two problems, Miller adds regularization terms or penalty functions in the residual minimization to obtain new objective functions [MI81]. He minimizes

$$\| \dot{U} - L(U) \|_2^2 + \sum_{i=1}^{N+1} (\varepsilon_i \triangle \dot{x}_i - S_i)^2 \tag{1.2.68}$$

with respect to $\dot{U}_i$, and $\dot{x}_i$, where $\varepsilon_i^2 = \frac{c_1^2}{\triangle x_i - \delta}$, $\varepsilon_i S_i = \frac{c_2^2}{(\triangle x_i - \delta)^2}$, c_1, c_2, and δ are problem-dependent constants and δ is a user-defined minimum mesh distance and is also used for the activation energy. The regularization terms affect the moving mesh equation (1.2.63) by adding

$$-\varepsilon_i^2 \dot{x}_{i-1} + (\varepsilon_i^2 + \varepsilon_{i+1}^2)\dot{x}_i - \varepsilon_{i+1}^2 \dot{x}_{i+1},$$

and

$$\varepsilon_i S_i - \varepsilon_{i+1} S_{i+1},$$

to the left- and right-hand sides of equation (1.2.63), respectively. Then it can be shown that the mass-matrix A becomes positive definite [MM81]. Miller interprets the physical meaning of regularization. The ε-terms are called '*internodal viscosity*' terms which avoid parallelism by controlling the relative motion between the meshes. The S-terms are called '*internodal spring forces*' which prevent numerical drift by providing a force between too close mesh points.

For shock problems, most of the mesh points move into the steepest regions of the shocks, because the total derivative $\dot{U}$ is close to being a delta function and large errors $\dot{U} - L(u)$ arise for the least square method with the L_2-norm. To de-emphasize the steep portions of the solution, the gradient weighted MFE method with the gradient weighted function $w(u_x) = \sqrt{1 + u_x^2}$ is introduced by Miller [MI83]. Thus, the system of Ordinary Differential

Equation for gradient weighted MFE becomes

$$< \dot{U}, \alpha_j w > \;=\; < L(u), \alpha_j w >, \tag{1.2.69}$$

$$< \dot{U}, \alpha_j w > \;=\; < L(u), \alpha_j w > + \textit{reg. terms}, \tag{1.2.70}$$

instead of equations (1.2.62) and (1.2.63). However, this system shares many of the same difficulties as equations (1.2.62) and (1.2.63), as we shall see in Chapter 3.

Herbst *et al.* [HMS82] have derived a moving **Petrov-Galerkin method** based on the moving finite element approximation equation

$$U(x,t) = \sum_{i=1}^{N} U_i(t)\alpha_i(x). \tag{1.2.71}$$

The α_i are defined as in (1.2.58), but they use piecewise Hermite cubic polynomials as test functions, viz. S_j and T_j given by

$$S_j(x) = [\alpha_j(x)]^2[3 - 2\alpha_j(x)], \tag{1.2.72}$$

and

$$T_j(x) = [\alpha_j(x)]^2[\alpha_j(x) - 1](\frac{d\alpha_j(x)}{dx})^{-1}. \tag{1.2.73}$$

Differentiating equation (1.2.71) with respect to t gives

$$\dot{U}(x,t) = \sum_{i=1}^{N} \dot{U}_i(t)\alpha_i + \beta_i \dot{x}_i, \tag{1.2.74}$$

where the β_j are given in (1.2.60). Then, by **Petrov-Galerkin method**, the coupled system of Ordinary Differential Equations is

$$\sum_{i=1}^{N} < \alpha_i, S_j > \dot{U}_i + < \beta_i, S_j > \dot{x}_i =< L(U), S_j >, \qquad j = 1, \ldots, N. \tag{1.2.75}$$

$$\sum_{i=1}^{N} < \alpha_i, T_j > \dot{U}_i + < \beta_i, T_j > \dot{x}_i =< L(U), T_j > . \qquad j = 1, \ldots, N. \tag{1.2.76}$$

To clarify the form of the above coupled ODE system, we write down the mass-matrix of the left hand side in (1.2.75) and (1.2.76):

$$\frac{1}{20}[3\triangle x_j\dot{U}_{j-1} + 7(\triangle x_j + \triangle x_{j+1})\dot{U}_j + 3\triangle x_{j+1}\dot{U}_{j+1}]$$
$$+\frac{1}{20}[3m_j\triangle x_j\dot{x}_{j-1} + 7(m_j\triangle x_j + m_{j+1}\triangle x_{j+1})\dot{x}_j + (3m_{j+1}\triangle x_{j+1})\dot{x}_{j+1}$$
$$=< L(u), S_j >, \tag{1.2.77}$$

and

$$\frac{1}{60}[2(\triangle x_j)^2\dot{x}_{j-1} + 3(\triangle x_j^2 - \triangle x_{j+1}^2)\dot{U}_j - 2(\triangle x_{j+1})^2\dot{U}_{j+1}]$$
$$+\frac{1}{60}[2(\triangle x_j^2 m_j\dot{x}_{j-1} + 3(\triangle x_j^2 m_j - (\triangle x_{j+1}^2 m_{j+1})\dot{x}_j - 2(\triangle x_{j+1})^2 m_{j+1}\dot{x}_{j+1}$$
$$=< L(u), T_j >, \tag{1.2.78}$$

where $\triangle x_j = x_j - x_{j-1}$. It is clear that the mass-matrix (1.2.75) and (1.2.76) is sometimes singular. So regularization terms are again required to avoid a singular mass-matrix. Herbst *et al.* also use the regularization terms of Miller [MI81] (see also [MI83] and [FU85]).

Comparing the moving finite element method of Miller and Miller with the moving Petrov-Galerkin method, Miller showed that if the moving finite element [MM81] is used in the H^{-1}-norm rather than the L_2-norm in the residual minimization, the two methods are equivalent for scalar PDEs in 1-D to within the regularization terms [MI83].

MFE [MM81] can also be derived by a suitable coordinate transformation (see Mueller and Carey [MC85], Lynch [LY82] and Baines [BA88]). To see this, consider the time-dependent partial differential equation

$$u_t = L(u). \tag{1.2.79}$$

Suppose (ξ, T) are new independent variables, and (x, t) are the original variables, where a coordinate transformation is defined by

$$x = \hat{x}(\xi, T), \tag{1.2.80}$$

$$t = T, \tag{1.2.81}$$

$$u(x,t) = \hat{u}(\xi,t). \tag{1.2.82}$$

Using the derivative chain rule

$$u_t = \hat{u}_T + \hat{u}_\xi \xi_t = \hat{u}_T - u_x \hat{x}_T, \tag{1.2.83}$$

$$\hat{u}_\xi = x_\xi u_\xi. \tag{1.2.84}$$

For the new coordinates (ξ, T), equation (1.2.79) becomes

$$\hat{u}_T - u_x \hat{x}_T = L(u). \tag{1.2.85}$$

Using the notation

$$\dot{u} = \hat{u}_T, \ \ \dot{x} = \hat{x}_T, \ \ u_x = \frac{\partial u}{\partial x},$$

(1.2.85) becomes

$$\dot{u} - u_x \dot{x} - L(u) = 0. \tag{1.2.86}$$

For a numerical approximation u of the exact solution, define the residual R by

$$R = \dot{u} - u_x \dot{x} - L(u).$$

Using the **Least Square Variational Method** for $\|R\|_2^2$ with respect to $\dot{u}$ and $\dot{x}$, we have the weak forms

$$< R, \psi >= 0 \tag{1.2.87}$$

and

$$< R, u_x \varphi >= 0 \tag{1.2.88}$$

for all admissible test function $\psi(= \delta\dot{u})$ and $\varphi(= \delta\dot{x})$.

Consider finite element approximations

$$u = \sum_j U_j(T)\phi_j(\xi), \tag{1.2.89}$$

$$x = \sum_j x_j(T)\phi_j(\xi), \tag{1.2.90}$$

where the $\phi_j(\xi)$ are basis functions for the finite element space.

Since

$$\dot{u} = \sum_j \dot{U}_j \phi_j(\xi), \tag{1.2.91}$$

$$\dot{x} = \sum_j \dot{x}_j \phi_j(\xi), \tag{1.2.92}$$

we obtain that

$$\psi = \delta\dot{u} = \phi_i(\xi) \quad \text{and} \quad \varphi = \delta\dot{x} = \phi_i(\xi).$$

Equations (1.2.87) and (1.2.88) become

$$< R, \phi_i >= 0 \tag{1.2.93}$$

and

$$< R, u_x\phi_i >= 0 \tag{1.2.94}$$

for all i, or

$$< \sum_j \dot{U}_j(T)\phi_j(\xi) - u_x \sum_j \dot{x}_j(T)\phi_j, \phi_i >=< L(u), \phi_i >, \tag{1.2.95}$$

$$< \sum_j \dot{U}_j(T)\phi_j(\xi) - u_x \sum_j \dot{x}_j(T)\phi_j, u_x\phi_i >=< L(u), u_x\phi_i > . \tag{1.2.96}$$

This is Miller's MFE if the basis functions ϕ_i are piecewise linear functions [MC85], [LY82] and [BA88].

Instead of minimizing $\|\dot{U} - L(u)\|_2^2$ with respect to the mesh $\dot{x}_i$ to determine the mesh equation, Mosher [MO85] gets the moving mesh equation based upon the equidistribution principle

$$\dot{M}_i = 0, \qquad 1 \le i \le N, \tag{1.2.97}$$

where

$$M_i = \begin{cases} x_1 & \text{if } i = 1 \\ f_{i-1} - f_i & \text{if } 2 \le i \le N-1 \\ x_N - 1 & \text{if } i = N \end{cases} \tag{1.2.98}$$

with f_i a function of the gradient and the curvature defined by

$$f_i = \sqrt{m_i^2 + \varepsilon_1}(x_{i+1} - x_i) + B_1\sqrt{(m_i - m_{i-1})^2 + \varepsilon_2}(x_{i+1} - x_{i-1}) + B_2\sqrt{(m_{i+1} - m_i)^2 + \varepsilon_2}(x_{i+2} - x_i) \quad (1.2.99)$$

for $i = 1, \cdots, N-1$. For ends points, B_1 is zero if $i = 1$ and B_2 is zero if $i = N-1$. Here B_1, B_2, ε_1, ε_2 are user-chosen constants. In his computations, $B_1 = B_2 = 0.025, \varepsilon_1 = 10^{-2}$, $\varepsilon_2 = 10^{-6}$ are used [MO85].

B. Adaptive finite element method

In this section, we discuss the adaptive finite element method proposed by Adjerid and Flaherty [AF86b1], [AF86b2]. The **Galerkin method** is used to discretize the partial differential equation in space with piecewise linear basis functions. They introduce an error estimate to control the moving mesh based on an equidistribution principle.

Consider

$$u_t = L(u). \quad (1.2.100)$$

The finite element approximation U of the exact solution u is defined using the weak form

$$< U_t, \varphi > = < L(U), \varphi >, \quad (1.2.101)$$

where φ is a suitable test function. Introducing a partition

$$\Pi: \quad a = x_0 < x_1(t) < \cdots < x_N(t) < x_{N+1} = b. \quad (1.2.102)$$

If

$$U(x,t) = \sum_{i=0}^{N+1} U_i(t)\alpha_i(x), \quad (1.2.103)$$

where α_i are piecewise linear functions defined in (1.2.58), then

$$U_t = \sum_{i=0}^{N+1}\{\dot{U}_i(t)\alpha_i(x) + \sum_{j=i-1}^{i+1} U_i(t)\frac{\partial \alpha_i}{\partial x_j}\dot{x}_j(t)\}. \quad (1.2.104)$$

To estimate the error between the exact solution u and the finite element approximation U, we define another weak form

$$< U_t + e_t, v >=< L(U+e), v > \tag{1.2.105}$$

for an admissible test function v. Then, we approximate the error e by piecewise quadratic functions

$$E(x,t) = \sum_{i=0}^{N+1} E_i(t)\Psi_i(x), \tag{1.2.106}$$

where

$$\Psi_i = \begin{cases} \frac{4[x-x_{i-1}(t)][x_i-x]}{[x_i-x_{i-1}]^2} & \text{for } x \in (x_{i-1}(t), x_i(t)), \\ 0 & \text{otherwise.} \end{cases} \tag{1.2.107}$$

Then the weak form is used to estimate the error, viz.

$$< U_t + E_t, \Psi_j >=< L(U+E), \Psi_j > . \tag{1.2.108}$$

Once we know the error estimate, the meshes can be controlled by requiring

$$\dot{x}_i(t) - \dot{x}_{i-1}(t) = -\lambda(\|E_i\Psi_i\|_1 - \bar{E}) \quad \text{for } i = 1, 2, \cdots, N, \tag{1.2.109}$$

where λ is a positive constant and

$$\|E\|_1 := \{\int_0^1 [(E)^2 + (E_x)^2]dx\}^{\frac{1}{2}}. \tag{1.2.110}$$

Here $\|E_i\Psi_i\|_1$ is the local error in H^1 on $(x_{i-1}(t), x_i(t))$ and $\bar{E}$ is the average error in H^1.

The moving mesh equation (1.2.109) is similar to equation (1.2.26). This method is more efficient for solving parabolic problems than for solving hyperbolic problems. It can estimate the error for each time step, although the method is quite sensitive to the choice for λ [AF86b1], [AF86b2].

Chapter 2

Theory & Computation of Moving Mesh Methods

2.1 Introduction

In this chapter we investigate various aspects of the moving mesh problem for the solution of partial differential equations in one space dimension. In particular, we study methods based (explicitly or implicitly) upon an equidistribution principle. Equidistribution is shown to be equivalent to the problem of solving a particular PDE for this new computational coordinate system. Implementation of a discrete version of equidistribution to compute a moving mesh corresponds to solving a weak form of the PDE. The stability of the equidistribution is discussed, and we argue that stability can be significantly affected by the way in which this solution process is carried out. A simple moving mesh method is constructed using this framework, and numerical examples are given to illustrate its robustness.

One of the most important computational consideration when solving partial differential equations (PDEs) having nontrivial solutions is the decision of how to automatically and stably choose a nonuniform mesh which suitably adapts to the solution behaviour. For initial value PDEs, constructing a moving mesh with time can be essential if the problem

is to be solved efficiently, and often if it is to be solved at all. The resolution of this issue has proven to be surprisingly difficult, and theoretical results have been particularly slow in coming. Considerable controversy surrounds the questions of which overall strategy to use and how best to choose a moving mesh for a given strategy [FVZ90], even though few basic mesh selection principles are around. Here, we investigate one of the key mesh selection strategies, that where equidistribution is explicitly done with respect to some measure of the error, and we discuss how these results have implications with regard to some other strategies as well. We focus on PDEs in one space dimension.

First consider the case of solving an ordinary differential equation (ODE), e.g.

$$u_{xx} = f(x, u, u_x) \tag{2.1.1}$$

with boundary conditions $u(a) = \beta_1$, $u(b) = \beta_2$. The equidistribution idea, apparently first introduced by deBoor [BO73] and Dodson [DO72], is based upon the simple idea that if some measure of the error $M(x)$ is available, then a good choice for a mesh $\pi : a = x_0 < x_1 < \cdots < x_N = b$ would be one for which the contributions to the error over the subintervals are equalized (or "distributing equally"). In practice, most strategies find π by only approximately equidistributed with respect to the so-called monitor function $M(x)$, although White [WH79] provides a framework for doing this distribution exactly. He defines a change of variables $s = \frac{1}{\theta}\int_a^x M(\xi)d\xi$, where $\theta := \int_a^b M(\xi)d\xi$, and then forms a new system of ODEs consisting of the original ODE (say, (2.1.1)) rewritten in terms of this computational variable s, and the ODE

$$\frac{dx}{ds} = \frac{\theta}{M(x)}. \tag{2.1.2}$$

Equidistribution then corresponds to choosing π with $s(x_{i+1}) - s(x_i) = \frac{1}{N}, i = 0, 1, \cdots, N-1$. This continuous form has been found to be a useful theoretical tool for interpreting schemes, but it has generally been found to be not too reliable computationally because the new ODE system can be extremely sensitive to solve. While all of the reasons for numerical

difficulties are not well understood, a major one is that the *transformed* ODE (in s) can be extremely nonlinear and its solution badly behaved due to the introduction of interior layers [RU79], [SM82]. Still, equidistribution strategies are widely used in conjunction with the original ODE (like (2.1.1)), and in this way they have enjoyed general success.

Consider now an initial/boundary value PDE

$$u_t = f(u, u_x, u_{xx}) \tag{2.1.3}$$

with $u(x,0), a \leq x \leq b$ and $u(a,t), u(b,t), t > 0$ given. Our concern is to investigate properties of an equidistribution procedure, and in many respects this does not depend upon the form of the PDE itself. For example, the PDE could instead be a system of equations in $\mathbf{u} = (u_1, \ldots, u_n)^T$ like

$$\mathbf{F}(\mathbf{u_t}, \mathbf{u_x}, \mathbf{u_{xx}}) = \mathbf{0}. \tag{2.1.4}$$

Many variations of equidistribution strategies have been investigated in practice. The first ones generally did a *static regridding* [HY83], where equidistribution to determine a new mesh is done after the solution to the PDE is computed at the new time level - e.g. see [AF86a]. Later, the PDE and mesh solution processes were combined to do *dynamic regridding* [HY83]. Several moving mesh methods based upon equidistribution were investigated by Coyle, Flaherty and Ludwig [CFL86]. Hyman [HY83] studied a moving mesh strategy for PDEs of the form (2.1.3), and later Petzold [PE87] did so for the implicit PDE (2.1.4), for details see page 6.

Mathematically, the goal of finding mesh *functions* $\{x_i(t)\}_{i=1}^{N-1}$, or moving meshes

$$\Pi : \{a = x_0 < x_1(t) < \cdots < x_{N-1}(t) < x_N = b\} \tag{2.1.5}$$

which are equidistributing for all values of t means that we want

$$\int_{x_{i-1}(t)}^{x_i(t)} M(x,t)dx = \frac{1}{N}\int_a^b M(x,t)dx =: \frac{1}{N}\theta(t), \quad i = 1, \cdots, N. \tag{2.1.6}$$

This equidistribution equation can be written equivalently as

$$\int_a^{x_i(t)} M(x,t)dx = \frac{i}{N}\int_a^b M(x,t)dx = \frac{i}{N}\theta(t), \quad i = 0, 1, \cdots, N. \tag{2.1.7}$$

With static regridding, equation (2.1.6) or (2.1.7) is approximately satisfied at every new time level, where the monitor function $M(x,t)$ depends upon the just computed solution of the PDE at this time level. (For notational convenience, the explicit dependence of M upon u is not specified.) With dynamic regridding, the PDE is solved together with (2.1.6) or (2.1.7), and the rate at which the mesh moves is a function of how θ changes with time. In actual fact, the regridding strategies only solve (2.1.7) approximately, producing so-called *asymptotically equidistributing* meshes, but this distinction will not be a focus of our presentation.

White [WH82] also studies the PDE case, which we discussed in page 4. As for the ODE, he replaces the physical variables x, t with a new set of computational coordinates s, T defined via the equidistribution process, viz.

$$s = \frac{1}{\theta}\int_a^x M(\xi,t)d\xi, \quad T = t. \tag{2.1.8}$$

He obtains a new PDE system consisting of the original PDE for u rewritten in term of s and T, and

$$\frac{\partial x}{\partial s} = \frac{\theta}{M(x,T)}. \tag{2.1.9}$$

Several attempts to solve this transformed PDE for u (as a function of these new computational variables) have been made, e.g., see [WH82], [DKS80]. This involves forming a discretization of the transformed PDE and solving a coupled system for the numerical solution and the equidistributing mesh which (approximately) satisfies (2.1.6). The resulting system is, however, generally sensitive to solve numerically [SM82]. The transformed PDE is nonlinear even if the original one is linear, and sometimes physically meaningless solutions are obtained. Still, as we shall see, it provides a useful model with which to interpret particular numerical schemes.

In [CFL86] the stability of the equidistribution process is studied. In particular, differentiating (2.1.7) with respect to t, they study the equations

$$M(x_i,t)\dot{x}_i + \int_a^{x_i} M_t(x,t)dx = \frac{i}{N}\frac{d\theta}{dt}, \quad i = 1,\cdots,N-1. \tag{2.1.10}$$

Using linear perturbation techniques for the mesh points, they perturb x_i by δx_i and take

$$M(x_i+\delta x_i,t)(\dot{x}_i+\delta\dot{x}_i) + \int_a^{x_i+\delta x_i} M_t(x,t)dx = \frac{i}{N}\frac{d\theta}{dt} \tag{2.1.11}$$

and linearize (2.1.11) to get the first order terms

$$M(x_i(t),t)\delta\dot{x}_i(t) + \frac{\partial M}{\partial x}(x_i(t),t)\dot{x}_i(t)\delta x_i(t) + M_t(x_i(t),t)\delta x_i(t) = 0,$$

i.e.,

$$\frac{d}{dt}[M(x_i(t),t)\delta x_i(t)] = 0. \tag{2.1.12}$$

Integrating from $t = 0$ to t,

$$\delta x_i(t) = \frac{M(x_i(0),0)}{M(x_i(t),t)}\delta x_i(0). \tag{2.1.13}$$

Doing this analysis and an accompanying numerical study, they conclude that mesh equidistribution, while unquestionably a desirable property for the mesh points, requires extreme care for its implementation because of potential instabilities for dissipative PDEs, where the perturbation terms in (2.1.13) can grow rapidly. A number of attempts are made to eliminate this potential instabilities for dissipative PDEs for the moving meshes. Generally, these involved some form of regularization [HN84], [PE87].

In the next section, we discuss the equidistribution problem within another framework, showing why the stability analysis needs to be interpreted cautiously. This is not surprising since it is unclear exactly how one would reconcile instability results concerning equidistribution with what is known about moving finite element (MFE) methods [MM81] in page 18. These methods have proven extremely effective for parabolic PDEs [GDM81]. Herbst *et al.*

have shown by Taylor expansion that the moving finite element equation (1.2.62) for the transport equation

$$u_t + (V(u))_x = \varepsilon u_{xx}$$

leads to the equation

$$\varepsilon(m_{i+1} - m_i)\triangle x_{i+1} u_{xx}(x_i^+, t) = \varepsilon(m_{i+1} - m_i)\triangle x_i u_{xx}(x_i^-, t) + (m_{i+1} - m_i)O(\triangle x^2)$$

where $\triangle x = \max_i(\triangle x_i)$ and $\triangle x_i = x_i - x_{i-1}$. The MFE methods have been shown to be related to a weak form of equidistribution [FU85, HSM83, TS86]. Also, we can see that for parabolic PDEs the matrix system can become singular if an equidistribution relationship is violated, *i.e.*, $m_{i+1} = m_i$ [HSM83]. For hyperbolics, *i.e.*, $\varepsilon = 0$, they run into difficulty at the very point where the equidistribution property is lost [FVZ90]. Further, Thrasher and Sepehmoori have proven that the MFE [MM81] equations without regularization terms satisfy a weak equidistribution relationship [TS86]. In order to avoid difficulties, Miller [MI81] introduces regularization terms, which implies the new equidistribution relation

$$\begin{aligned} &3(m_{i+1} - m_i)\triangle x_{i+1}\varepsilon u_{xx}(x_i^+, t) + \epsilon_{i+1}^2(\dot{x}_{i+1} - \dot{x}_i) - \epsilon_{i+1}S_{i+1} \\ &= 3(m_{i+1} - m_i)\triangle x_i \varepsilon u_{xx}(x_i^-, t) + \epsilon_i^2(\dot{x}_i - \dot{x}_{i-1}) - \epsilon_i S_i + O(\triangle x^2). \end{aligned}$$

From the above equation, we see that the penalty function in this regularization plays the key role of *preserving* equidistribution [HSM83] when $m_{i+1} = m_i$ or $\varepsilon = 0$. Nevertheless, the practical implications of these often tenuous theoretical connections are difficult to interpret, leaving many stability issues still open to question.

2.2 Equidistribution PDE

In order to analyze further the stability of moving meshes satisfying an equidistribution principle, we derive a PDE which provides a new interpretation of equidistribution. From (2.1.8),

$$s\theta(t) = \int_a^x M(\xi, t)d\xi$$

so assuming M is a smooth function, along lines where $s(t)$ is *constant* with respect to time t,

$$s_t\theta + s\dot{\theta} = s\dot{\theta} = \int_a^x M_t(\xi,t)d\xi + M\dot{x} \tag{2.2.14}$$

implying

$$s_x\dot{\theta} + s\dot{\theta}_x = s_x\dot{\theta} = M_t(x,t) + \frac{\partial}{\partial x}(M\dot{x}). \tag{2.2.15}$$

Thus, we have the differential form

$$\frac{\partial}{\partial t}M(x,t) + \frac{\partial}{\partial x}(M(x,t)\dot{x}) = \frac{\dot{\theta}}{\theta}M(x,t). \tag{2.2.16}$$

or

$$\frac{\partial}{\partial t}M + div(M\dot{x}) = \frac{\dot{\theta}}{\theta}M. \tag{2.2.17}$$

Consequently, doing equidistribution implicitly corresponds to finding a solution to (2.2.16). Although this is technically an integro-differential equation, we shall refer to it as a **hyperbolic conservation-type PDE**. (In the next section the integral term is eliminated through a change of variables.) Differentiating (2.1.6) with respect to t, we obtain

$$\int_{x_{i-1}}^{x_i} M_t(x,t)dx + M(x_i,t)\dot{x}_i(t) - M(x_{i-1},t)\dot{x}_{i-1}(t) = \frac{1}{N}\dot{\theta}$$

or

$$\int_{x_{i-1}}^{x_i} M_t(x,t)dx + \int_{x_{i-1}}^{x_i} \frac{\partial}{\partial x}(M\dot{x})dx = \frac{1}{N}\dot{\theta},$$

Using (5), it is shown that the discrete equidistribution process for the mesh (2.1.5) corresponds to finding a solution to

$$\int_{x_{i-1}}^{x_i} [M_t(x,t) + \frac{\partial}{\partial x}(M(x,t)\dot{x})]dx = \frac{1}{N}\dot{\theta} = \int_{x_{i-1}}^{x_i} \frac{\dot{\theta}}{\theta}M(x,t)dx. \tag{2.2.18}$$

Thus, by letting the cell $[x_{i-1}, x_i]$ shrink to a point we obtain the moving mesh PDE (2.2.16).

This viewpoint is valuable in several respects. Practically, the effects of discretizing (2.1.6) and (2.2.16) are similar. However, considerable experience has been gained from solving PDEs in conservation-type form like (2.2.16), so it is natural to try to develop new

methods and interpret previous ones using this formulation. It also provides a physical interpretation of these moving mesh methods in terminology familiar in fluid dynamics. The mesh points serve a similar function as particles of flow. In particular, the equidistributing coordinates are chosen using a **quasi-Lagrangian approach**: the moving mesh is along lines of constant $s(t) = \int_a^{x(t)} \frac{M(\xi,t)}{\theta(t)} d\xi$. Here, (2.1.6) satisfies a finite version of the integral form, or *weak form.* The weak form (2.1.6) of the PDE shows that the "flux" of the error density function M is equivalent across the subintervals, or across each cell $[x_{i-1}, x_i]$. If the total measure of error in the interval $[a, b]$ is constant, then $\dot{\theta} = 0$, and the moving mesh equation (2.2.16) becomes the Euler equation for the "fluid" with density function $M(x,t)$. Finally, the case where meshes are calculated using static regridding can be viewed as corresponding to the steady flow case in fluid dynamics, where the error density function $M(x,t)$ is independent of time. A Lagrangian approach corresponds to the choice $M(x,t) = u_x$ and $\dot{\theta} = 0$. In this case integration of (2.2.16) gives the well-known conservation law [LA73]

$$u_t + u_x \dot{x} = 0 \tag{2.2.19}$$

and $x(t)$ is simply a characteristic. For the arclength monitor function $M(x,t) = \sqrt{1+u_x^2}$, if $u_x \gg 0$ then $M(x,t) = u_x$, and we see how the moving mesh equation reflects the shock behaviour where characteristics cross. For hyperbolic PDEs, the MFE method with no regularization has also been shown to produce moving meshes along characteristics [BA88, HMS82].

While the PDE (2.2.16) has to our knowledge not been used previously to interpret mesh selection schemes in a general setting, similar approaches have been investigated in special contexts. As mentioned in Chapter 1 Larrouturou [LA89] develops an inexpensive moving mesh method for which the mesh points move with a time-dependent velocity $\dot{x}(t)$, the monitor function is chosen as a physical quantity (temperature), and the total energy $\theta(t)$ is constant. This gives a PDE like (2.2.16) to solve for $\dot{x}(t)$, but $\dot{\theta}(t) = 0$. In Larrouturou's actual implementation, the new mesh is computed using static regridding.

Theoretically, White's approach [WH82], being based upon equidistribution, involves satisfying (2.2.16) exactly. He writes the PDE in terms of the computational ("Lagrangian") coordinates (s, T), and since the solution has no steep gradients in these coordinates, a uniform mesh with $s(x_{i+1}) - s(x_i) = \frac{1}{N}$ is used for the transformed PDE. He generally works with arclength as the monitor function, i.e., $M(x,t) = \sqrt{1+u_x^2}$.

From the relation between the moving mesh problem with equidistribution and (2.2.16), it is easy to see how difficulties can arise computationally. Approximation to the left-hand-side should generally be done with a conservative scheme, or one could expect difficulties to occur. While many excellent methods of such type are available, when $\dot{\theta} \neq 0$ this term can cause considerable numerical difficulty, and finding suitable numerical methods just to solve a PDE of this type is a far from well understood matter [LY88]. The situation here is of course further complicated because the moving mesh PDE is coupled to the *original* PDE.

To see how difficulties can arise in general for the moving mesh equation, suppose that we assume that $\dot{\theta} = 0$. Then (2.2.16) becomes

$$\frac{\partial M}{\partial t} + \frac{\partial}{\partial x}(M\dot{x}) = 0. \tag{2.2.20}$$

If we use the non-conservative form

$$\frac{\partial M}{\partial t} + M\frac{\partial}{\partial x}(\dot{x}) + \frac{\partial M}{\partial x}\dot{x}(t) = 0 \tag{2.2.21}$$

and discretize using a standard method of lines procedure, we obtain

$$M_t(x_i(t),t) + M(x_i(t),t)\frac{\dot{x}_{i+1}(t) - \dot{x}_i(t)}{x_{i+1}(t) - x_i(t)} + \frac{\partial M(x_i(t),t)}{\partial x}\dot{x}_i(t) = 0,$$

or

$$M_t(x_i,t)(x_{i+1} - x_i) + M(x_i,t)(\dot{x}_{i+1} - \dot{x}_i) + \frac{\partial M}{\partial x}\dot{x}_i(x_{i+1}(t) - x_i(t)) = 0. \tag{2.2.22}$$

Thus,

$$\frac{d}{dt}[M(x_i(t),t)(x_{i+1}(t) - x_i(t))] = 0, \tag{2.2.23}$$

and upon integrating, we get

$$x_{i+1}(t) - x_i(t) = \frac{M(x_i(0),0)}{M(x_i(t),t)}(x_{i+1}(0) - x_i(0)). \tag{2.2.24}$$

Unless $\frac{M(x_i(0),0)}{M(x_i(t),t)}$ remains small, which it generally would *not* do for dissipative PDEs, the moving mesh points can easily cross and/or leave the domain $[a,b]$.

These observations apply as well for to the differential equation (2.1.13) developed in [CFL86] using linear perturbation techniques. It is useful to investigate this linear perturbation analysis further. Expanding (2.1.12) and dividing by δx_i, we have

$$M(x_i(t),t)\frac{\delta \dot{x}_i(t)}{\delta x_i(t)} + \frac{\partial M}{\partial x}\dot{x}_i(t) + M_t(x_i(t),t) = 0. \tag{2.2.25}$$

Letting $\delta x_i \to 0$, we obtain

$$M(x_i(t),t)\frac{\partial}{\partial x}(\dot{x}_i) + \frac{\partial M}{\partial x}\dot{x}_i(t) + M_t(x_i(t),t) = 0. \tag{2.2.26}$$

or

$$[\frac{\partial M}{\partial t} + \frac{\partial}{\partial x}(M\dot{x})]_{x=x_i} = 0. \tag{2.2.27}$$

The steps from (2.2.25) to (2.2.27) can be retraced.

Thus, the perturbation equation (2.1.13) used in [CFL86] to study stability of the equidistribution process can be obtained from setting the source term in (2.2.16) to zero and writing the resulting equation at $x = x_i$ in nonconservative form. In retrospect, we see that (2.1.13) resembles a conservation of mass equation, where θ corresponds to total mass which is unchanging with time. Finally, note that (2.1.13) is obtained from perturbing only the left-hand-side of the equation (2.1.7) or (2.1.10), since there is no perturbation expansion for the term $\frac{1}{N}$. We conclude that, while (2.1.13) is extremely useful for interpreting the stability of many *implementations* of equidistribution procedures, the stability properties of the equidistribution principle itself are more complicated.

Through a simple change of variables, (2.2.16) can be converted from a differential-integral equation in $M(x,t)$ to a differential equation for which the stability analysis of

[CFL86] is more generally applicable. Introducing the transformation

$$W(x,t) := \frac{M(x,t)}{\theta} = \frac{M(x,t)}{\int_a^b M(x,t)dx}, \tag{2.2.28}$$

it is easy to see that (2.2.16) takes the equivalent form

$$\frac{\partial W(x,t)}{\partial t} + \frac{\partial (W(x,t)\dot{x})}{\partial x} = 0. \tag{2.2.29}$$

Thus, the "average energy" function $W(x,t)$, for which $\int_a^b W(x,t)dx \equiv 1$, satisfies the conservation equation (2.2.29). The transformation (2.2.28) is of a similar type to the Cole-Hopf transformation [WH74], although the context and purpose are quite different. From (2.1.6), the weak form of the PDE (2.2.29) is

$$\int_{x_i}^{x_{i+1}} W(x,t)dx = \frac{1}{N}, \quad i = 1,\ldots,N.$$

i.e., the total "average energy" between any two mesh lines remains constant. There is in principle no reason why the moving mesh approaches constructed in terms of $M(x,t)$ can not use $W(x,t)$ instead. The derivation from (2.2.20) through (2.2.24) can be repeated with W replacing M, and under the appropriate corresponding conditions (except with no right-hand-side which needs to be ignored) we see that the potential for mesh crossings now occurs if

$$\frac{W(x_i(0),0)}{W(x_i(t),t)} = \frac{M(x_i(0),0)}{M(x_i(t),t)}\frac{\theta(t)}{\theta(0)},$$

a measure of the *average* change in $M(x_i(t),t)$, grows. Comparing with (2.2.24), one would hope that the moving mesh equations derived using this new variable would be more robust, if not necessarily more efficient. Finally, the analysis [CFL86] is directly applicable to (2.2.29), so stability of an non-conservative equidistribution process is given by (2.1.13) with W replacing M.

2.3 Analysis of equidistribution PDE

In this section, we study the pure equidistribution PDE without considering the physical PDE (the original PDE). In particular, consider the equidistribution partial differential

equation

$$\frac{\partial w}{\partial t} + \frac{\partial (w\dot{x})}{\partial x} = 0 \tag{2.3.30}$$

where $\int_a^b w(x,t)dx = 1$. Letting

$$a := \frac{d(w\dot{x})}{dw} = \dot{x} + w\frac{d\dot{x}}{dw}, \tag{2.3.31}$$

we can write (2.3.30) in the form

$$w_t + a(w)w_x = 0. \tag{2.3.32}$$

This asserts that w is a constant along trajectories $x = x(t)$ which propagates with speed

$$w\frac{d\dot{x}}{dw} + \dot{x} = a(w) \tag{2.3.33}$$

Suppose that the solution of equation (2.3.30) is smooth on each side of a smooth curve $x = y(t)$, across which w develops a shock and is discontinuous. Denote by w_l and w_r the values of w on the left and right sides, respectively, of $x = x_i$, and assume that the curve y intersects the interval $a \leq x \leq b$ at time t. Then

$$1 = \int_a^b w(x,t)dx = \int_a^{x_i} w(x,t)dx + \int_{x_i}^b w(x,t)dx, \tag{2.3.34}$$

and taking the derivative with respect to t we have

$$0 = \int_a^{x_i} w_t dx + w_l s + \int_{x_i}^b w_t dx - w_r s \tag{2.3.35}$$

where $s = \frac{dx_i}{dt}$ for the speed with which the discontinuity propagates. Since

$$w_t = -(w\dot{x})_x,$$

$$\begin{aligned} 0 &= \textstyle\int_a^{x_i}(-w\dot{x})_x dx + w_l s + \int_{x_i}^b (-w\dot{x})_x dx - w_r s \\ &= -(w\dot{x}_i)_l + (w\dot{x})_{x=a} + w_l s + (-w\dot{x})_{x=b} + (w\dot{x})_r - w_r s. \end{aligned} \tag{2.3.36}$$

The conservation law asserts that

$$(w\dot{x})_a - (w\dot{x})_b = 0.$$

Then the jump condition or Rankine - Hugoniot condition

$$s = \frac{(w\dot{x})_r - (w\dot{x})_l}{w_r - w_l} = \dot{x}_i \frac{w_r - w_l}{w_r - w_l} = \dot{x}_i \tag{2.3.37}$$

is automatically satisfied.

For the hyperbolic equation

$$\frac{\partial w}{\partial t} + a(w)w_x = 0,$$

there is a general solution $w = F(x - ta)$. If we define w to be a density per unit length and $w\dot{x}$ to be a flux per unit time, then the solution of this equation can be viewed as having kinematic wave behaviour, where $a(w)$ is the propagation velocity of the wave.

Breaking condition [WH74]

A continuous wave breaks and requires a shock if and only if the propagation velocity a decreases as x increases. Therefore when the shock is included we have

$$a_l > s > a_r \tag{2.3.38}$$

where all velocities are measured positive in the direction of increasing x. From the breaking condition (2.3.38), and the definition of the velocity a, we get

$$(\dot{x})_l + w_l(\frac{\partial \dot{x}}{\partial w})_l > s > (\dot{x})_r + w_r(\frac{\partial \dot{x}}{\partial w_r}). \tag{2.3.39}$$

If $(\dot{x})_l = (\dot{x})_r = \dot{x}$, then from the jump condition we have $s = \dot{x}$. Now (2.3.39) becomes

$$(\dot{x})_l + w_l(\frac{\partial \dot{x}}{\partial w})_l > \dot{x} > (\dot{x})_r + w_r(\frac{\partial \dot{x}}{\partial w_r}),$$

or

$$w_l(\frac{\partial \dot{x}}{\partial w})_l > 0 > w_r(\frac{\partial \dot{x}}{\partial w_r}).$$

Since w is the positive ***average*** monitor function in mesh selection, we obtain the breaking condition for the equidistribution PDE (2.3.30)

$$(\frac{\partial \dot{x}}{\partial w})_l > 0 > (\frac{\partial \dot{x}}{\partial w_r}). \tag{2.3.40}$$

When the breaking condition is satisfied, we get one weak solution of the PDE (2.3.30) that is a shock wave. Thus the breaking condition corresponds to mesh points crossing each other.

2.4 Implementations of equidistribution

In this section we consider ways in which the equidistribution process can be implemented. First, we consider how to choose the monitor or density function which controls the movement of the mesh points. This is more difficult than for ODEs due to the additional variable t. There are three basic choices of $M(x,t)$ which have been widely used in practice: (i) an arclength monitor function [WH82, DD87], (ii) a combination of gradient and curvature [MD88, DKS80, HL86, DD87, MO85], and (iii) truncation error or solution residual - used directly for ODEs [RU79], and either explicitly [AF86a, BB86] or implicitly [MM81, HSM83] for moving finite element methods for PDEs.

Stability properties of the moving mesh equations, while dependent upon the choice of monitor function, are to some extent arbitrary since they usually behave asymptotically much like some fractional power of a solution derivative (*e.g.*, see [RC78]). Here, we use the arclength monitor function

$$M(x,t) = \sqrt{1+u_x^2}. \tag{2.4.41}$$

Our first implementation of a moving mesh method involves using an approximation for (2.2.16) of the form

$$\frac{\partial}{\partial t}M(x_i(t),t) + \frac{M_{i+1}\dot{x}_{i+1}(t) - M_i\dot{x}_i(t)}{x_{i+1}(t)-x_i(t)} = \frac{\dot{\theta}}{\theta}M(x_i(t),t), \quad 1 \le i \le N-1. \tag{2.4.42}$$

For the numerical examples presented in the next section, $\dot{x} > 0$ so this simple upwind approximation to $(M\dot{x})_x$ is sufficient. On the interval $[x_i, x_{i+1}]$, we use the monitor function discretization

$$M_i := M(x_i,t) = \sqrt{1+(\frac{u_{i+1}-u_i}{x_{i+1}-x_i})^2}. \tag{2.4.43}$$

To maintain discrete conservative,

$$\theta(t) = \int_a^b M(x,t)dx = \sum_{i=0}^{N-1} \int_{x_i}^{x_{i+1}} M(x,t)dx$$

and

$$\dot{\theta}(t) = \int_a^b M_t(x,t)dx = \sum_{i=0}^{N-1} \int_{x_i}^{x_{i+1}} M_t(x,t)dx$$

are approximated using left rectangle rules

$$\int_{x_i}^{x_{i+1}} M(x,t)dx \approx (x_{i+1} - x_i)M(x_i(t),t), \tag{2.4.44}$$

$$\int_{x_i}^{x_{i+1}} M_t(x,t)dx \approx (x_{i+1} - x_i)M_t(x_i(t),t). \tag{2.4.45}$$

We test this moving mesh strategy, hereafter called **Method I**, both with and without the right-hand-side in (2.4.42). For both, only the fixed boundary case $\dot{x}_0(t) = \dot{x}_N(t) = 0$ is considered.

We also consider a moving mesh method developed using (2.2.29). Approximating this over $[x_i, x_{i+1}]$ at $t = t_{n+1} = (n+1)\Delta t$ by

$$\frac{W_i^{n+1} - W_i^n}{\Delta t} + \frac{W_i^{n+1}\dot{x}_{i+1} - W_{i-1}^{n+1}\dot{x}_i}{x_{i+1} - x_i} = 0, \tag{2.4.46}$$

upon rearrangement we obtain

$$W_i^{n+1}(x_{i+1} - x_i) + \Delta t(W_i^{n+1}\dot{x}_{i+1} - W_{i-1}^{n+1}\dot{x}_i) = (x_{i+1} - x_i)W_i^n. \tag{2.4.47}$$

A similar approximation on $[x_{i-1}, x_i]$ gives

$$W_{i-1}^{n+1}(x_i - x_{i-1}) + \Delta t(W_{i-1}^{n+1}\dot{x}_i - W_{i-2}^{n+1}\dot{x}_{i-1}) = (x_i - x_{i-1})W_{i-1}^n. \tag{2.4.48}$$

Equidistribution implies

$$\int_{x_i}^{x_{i+1}} W(x,t)dx = \int_{x_{i-1}}^{x_i} W(x,t)dx. \tag{2.4.49}$$

Equating the right-hand-sides of (2.4.47), (2.4.48), which are approximations to (2.4.49) at $t = t_n$, we obtain

$$\begin{aligned} & W_i^{n+1}(x_{i+1} - x_i) + \triangle t(W_i^{n+1}\dot{x}_{i+1} - W_{i-1}^{n+1}\dot{x}_i) \\ = \; & W_{i-1}^{n+1}(x_i - x_{i-1}) + \triangle t(W_{i-1}^{n+1}\dot{x}_i - W_{i-2}^{n+1}\dot{x}_{i-1}) \; . \end{aligned}$$

Since each term involves $W(x, t_{n+1}) = \frac{M(x,t_{n+1})}{\theta(t_{n+1})}$, $\theta(t)$ can be eliminated, leaving the discrete approximation

$$\tau(M_i\dot{x}_{i+1} - 2M_{i-1}\dot{x}_i + M_{i-2}\dot{x}_{i-1}) = M_{i-1}(x_i - x_{i-1}) - M_i(x_{i+1} - x_i) \tag{2.4.50}$$

at $t = t_{n+1}$, where to avoid confusion with the time integration steps later on we write $\tau := \triangle t$. This moving mesh strategy, which we refer to as **Method II** (also using (2.4.43) and with $\dot{x}_0(t) = \dot{x}_N(t) = 0$), is considered in the next section.

A great variety of moving mesh equations have been obtained by others, taking the various choices of monitor functions and approximation schemes. In the remainder of this section, we show how some of these are related to the equidistribution relationships derived in §2, either in the differential form (2.2.16) or the weak form (2.1.6).

For the moving finite element methods of Miller and Miller [MM81] and Herbst et al. [HSM83], the moving mesh equations are derived from the weak form of the PDEs written in Lagrangian form. In particular, a given PDE $u_t = L(u)$ is converted to its Lagrangian form $\dot{u} - u_x\dot{x} = L(u)$. Suitable weight functions $\phi_i(x)$ and $\psi_i(x)$ are chosen, and the residual

$$R(u) = \dot{u} - u_x\dot{x} - L(u)$$

is required to satisfy the orthogonality relations

$$\int_a^b \phi_i(x)R(u)dx = 0, \tag{2.4.51}$$

$$\int_a^b \psi_i(x)R(u)dx = 0. \tag{2.4.52}$$

The choice

$$\phi_i(x) = \alpha_i(x) \tag{2.4.53}$$

and

$$\psi_i(x) = \beta_i(x) = -u_x\alpha_i(x) \tag{2.4.54}$$

where $\alpha_i(x)$ is the hat function and $\beta_i(x)$ is a discontinuous piecewise linear functions on $[x_{i-1}, x_{i+1}]$ (see (1.2.58) and (1.2.60)) is made in [MM81], and the choice of the piecewise cubic Hermite polynomials

$$\phi_i(x) = [\alpha_i(x)]^2[3 - 2\alpha_i(x)], \tag{2.4.55}$$

$$\psi_i(x) = [\alpha_i(x)]^2[\alpha_i(x) - 1][\frac{d\alpha_i(x)}{dx}]^{-1} \tag{2.4.56}$$

is used in [HSM83]. Both of these can be shown to be implicitly based upon a weak form of the PDE (2.2.16). In particular, requiring that

$$\int_{x_{i-1}}^{x_i} M(x,t)dx = \int_{x_i}^{x_{i+1}} M(x,t)dx \tag{2.4.57}$$

is satisfied for the monitor function

$$M_i(x,u) = (x_i - x_{i-1})R(u_i),$$

we obtain the moving mesh equation corresponding to (2.4.55) and (2.4.56), and for

$$M(x,t) = \begin{cases} -u_xR(u) & \text{for } x \in [x_{i-1}, x_i] \\ u_xR(u) & \text{for } x \in [x_i, x_{i+1}] \end{cases}$$

we obtain the moving mesh equation corresponding to (2.4.53) and (2.4.54) [FU85].

Aside from stability, one of the most troublesome problems for a moving mesh method is the tendency for mesh points to cross. For equidistribution (2.1.6), this easily happens if M changes sign, so to avoid this the early papers on equidistribution define a monitor function to be non-negative. For MFE methods, the associated equidistribution property above holds for a monitor function which changes sign, and consistent with this the fact that regularization terms generally need be added to prevent mesh crossings. In contrast, we find that for discretizations formed directly from (2.2.16), for positive monitor functions the problem of mesh crossing itself can be minimal.

If instead of (2.2.16) we take

$$\frac{\partial}{\partial x}(M\dot{x}) = 0$$

and write it in the non-conservative form

$$\frac{\partial M}{\partial x}\dot{x} + M\frac{\partial \dot{x}}{\partial x} = 0,$$

then the discretization

$$\frac{M_i - M_{i-1}}{x_i - x_{i-1}}\dot{x}_i + M_i\frac{\dot{x}_i - \dot{x}_{i-1}}{x_i - x_{i-1}} = 0$$

gives

$$\dot{x}_i - \dot{x}_{i-1} = -\frac{\dot{x}_i}{M_i}(M_i - M_{i-1}).$$

This is similar to the moving mesh equation of [AF86b1], except they attempt to optimize a parameter value which is used in place of $\frac{\dot{x}_i}{M_i}$.

If the monitor function is simply the solution to the PDE, i.e. $M(x,u) = u$, then (2.2.16) becomes

$$\frac{\partial u}{\partial t} + \frac{\partial(u\dot{x})}{\partial x} = u\frac{\dot{\theta}}{\theta},$$

or in non-conservative form

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\dot{x} + u\frac{\partial \dot{x}}{\partial x} = u\frac{\dot{\theta}}{\theta}. \tag{2.4.58}$$

In developing a moving mesh strategy, Petzold [PE87] attempts to minimize, for a suitable parameter α, the objective function

$$\phi(\dot{x},\dot{u}) = (\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\dot{x})^2 + \alpha\dot{x}^2,$$

which is a measure of the change in the solution u and mesh x with respect to time t. Since meshpoints can easily cross with this choice, she introduces a penalty function $\lambda[(\frac{\dot{x}_i - \dot{x}_{i-1}}{x_i - x_{i-1}})^2 + (\frac{\dot{x}_{i+1} - \dot{x}_i}{x_{i+1} - x_i})^2]$. This can be viewed as a "replacement" to the missing term $u\frac{\partial \dot{x}}{\partial x}$ in (2.4.58), which gives a scheme that in some sense minimizes the "source error energy" $u\frac{\dot{\theta}}{\theta}$ for the PDE when moving mesh points in time. The usefulness of this interpretation of Petzold's scheme to develop other practical moving mesh strategies remains to be investigated.

One of the most reliable moving mesh discretizations is due to Dorfi and Drury [DD87] and analyzed in [VBFZ88]. It is similar to (2.4.50), where the general relationship between them involves using an artificial dissipation term in conjunction with (2.4.50). We will discuss them in Chapter 3.

2.5 Numerical results

Here we give some numerical examples to examine the moving mesh strategy from **Method I**- with and without the right-hand-side of (2.4.42) - and the strategy from **Method II**. We choose three examples, consisting of one hyperbolic and two parabolic problems. both using and not using the source term.

To discretize the PDE

$$\frac{\partial u}{\partial t} = f(u, u_x, u_{xx}), \tag{2.5.59}$$

we first write it in the Lagrangian form

$$\dot{u} - u_x \dot{x} = f(u, u_x, u_{xx}). \tag{2.5.60}$$

Next, using a central difference scheme for the spatial derivatives, we obtain

$$\dot{u}_i - \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}} \dot{x}_i = f_i, i = 2, \ldots, N. \tag{2.5.61}$$

Thus, we solve the coupled system of equations (2.5.61) and (2.4.42), with and without $\dot{\theta} = 0$. This ODE system is solved using the code LSODI of Hindmarsh [HI80]. An approximate Jacobian is computed by LSODI internally using difference quotients. For simplicity, an initial uniform mesh is used in each case. In the tables of numerical results reported, *nst* and *nje* are respectively the number of steps and number of Jacobian evaluations taken by LSODI up to the time given, and *nqn* and *tstep* are respectively the order of the last successful method and the last successful stepsize. All runs were made on Sparcstations in a distributed computing environment, and computer times are not given. **Method I** is more

expensive using the right-hand-side in (2.4.42) than not using it, but the difference is not very significant (always less than 20% for these problems).

Problem 1: This problem, a scalar reaction diffusion problem from combustion theory, has been used by several authors to test their moving mesh strategies [AF86a, PE87, FVZ90]. It is a model of a single step reaction with diffusion,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + D(1+a-u)\exp\frac{-\delta}{u}, \quad t>0, \quad 0<x<1,$$

$$u_x(0,t) = 0, \quad u(1,t) = 1, \quad t>0,$$

$$u(x,0) = 1, \quad 0 \le x \le 1,$$

where the constant heat release is a, reaction rate is R, activation energy is δ, and Damkohler number is $D = R\exp\delta/(a\delta)$. The solution represents the temperature of a reactant in a combustion. For small time the temperature gradually increases from unity with a "hot spot" forming at $x = 0$. At a finite time ignition occurs and temperature at $x = 0$ jumps rapidly from unity to $1 + a$. A flame front then forms and propagates towards $x = 1$ with speed proportional to $\exp(a\delta)/2(1+a)$. Here a is about unity and δ is large, so the flame front moves exponentially fast after ignition. The problem reaches a steady state once the flame propagates to $x = 1$. This problem serves as a good test of moving mesh methods because of the sensitivity of tracking the flame front [AF86b1].

The derivative boundary condition $\frac{\partial u}{\partial x}(0,t) = 0$ is approximated by $\frac{u_1-u_2}{x_1-x_2} = 0$, or $u_1 - u_2 = 0$. The problem is solved for $a = 1, \delta = 20$, and $R = 5$, using a moving mesh with $N = 20$ and with $N = 40$. The results are compared with a reference solution (solid lines in the Figures) obtained by LSODI, using the method of lines with standard central differences on (2.5.59) and $N = 500$ equal spaced mesh points, with absolute tolerance $atol = 10^{-8}$ and relative tolerance $rtol = 10^{-6}$. The problem is quite sensitive to the tolerances for LSODI. For example, for $atol = rtol = 10^{-3}$, the numerical solution (not given here) moves too fast and is very inaccurate.

Fig. 2.1 shows the numerical solution computed using **Method I** with $\dot{\theta} \neq 0$, for $N = 20$, $atol = 10^{-5}$ and $rtol = 10^{-4}$. The solution is fairly accurate except for an error caused by the solution moving too fast so that it gives a slight shift for $t = 0.27$ and 0.28. This error is largely caused by the time integration, as the results change qualitatively when smaller tolerances are used in LSODI (see below). The corresponding results for $\dot{\theta} = 0$ are shown in Fig. 2.2. Note that the solution is inaccurate at the left boundary when $t = 0.26$, and the solution is not very well equidistributed with respect to arclength, especially near the left boundary. The sensitivity of the problem with respect to integrator tolerances is severe, as performing the same runs with larger tolerances can easily give poorer results, but even using $atol = 10^{-6}$ and $rtol = 10^{-5}$ gives lower accuracy (c.f. Fig. 2.3 and Fig. 2.4).

For **Method II** with $atol = 10^{-5}, rtol = 10^{-5}, \tau = 10^{-5}$ and $N = 20$, the numerical solution moves slightly slower than the reference solution before reaching steady state (see Fig. 2.5). Reducing the spatial mesh to $N = 40$, the solution has fairly high accuracy throughout, as shown in Fig. 2.6. Reducing τ or the integrator tolerances does not qualitatively affect the numerical solution, although from our experience τ should be kept smaller than the time integration stepsize used in LSODI. Note that the arclength is considerably better distributed between mesh points than for the other moving mesh equation.

The time-stepping information for the runs summarized in the Figures are given in Table 2.1. In particular, the number of steps and Jacobian evaluations, order of the integration method, and final step size used by LSODI are listed.

Problem II: Burgers' equation

Our next example is Burgers' equation

$$\frac{\partial u}{\partial t} = -\frac{\partial f(u)}{\partial x} + \varepsilon \frac{\partial^2 u}{\partial x^2}, \quad t > 0, \quad 0 < x < 1,$$

$$u(0,t) = 0, \quad u(1,t) = 0, \quad t > 0$$

$$u(x,0) = u_0(x),$$

t	Fig. 2.1(Method I):60.88(cpu)				Fig. 2.2(Method I):56.60(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.26	33	14	3	0.000286	30	11	3	0.000728
0.27	80	21	3	0.000374	80	25	3	0.000390
0.28	108	25	3	0.000369	114	33	2	0.000368
0.29	138	28	3	0.000395	145	40	3	0.000466
t	Fig. 2.3(Method I):52.64				Fig. 2.4(Method I):36.74			
0.26	53	13	3	0.000280	54	12	3	0.000514
0.27	128	23	4	0.000175	121	24	3	0.000216
0.28	187	31	3	0.000161	168	31	3	0.000274
0.29	246	39	3	0.000187	212	38	3	0.000263
t	Fig. 2.5(Method II):45.41				Fig. 2.6(Method II):272.12			
0.26	46	12	3	0.000302	57	16	3	0.000121
0.27	146	34	3	0.000183	143	34	1	0.000109
0.28	193	43	3	0.000259	197	44	2	0.000265
0.29	233	52	2	0.000348	236	56	2	0.000393

Table 2.1: Problem I

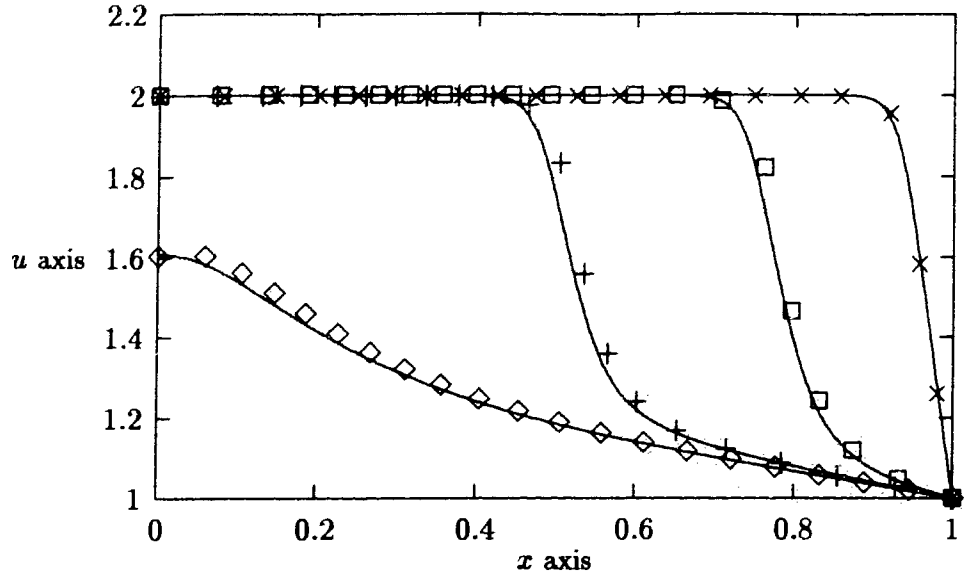

Figure 2.1: Problem I, using method I with $\dot{\theta} \neq 0$, t= $0.26, 0.27, 0.28, 0.29$; *atol* = 10^{-5}, *rtol* = 10^{-4}, mesh points N = 20

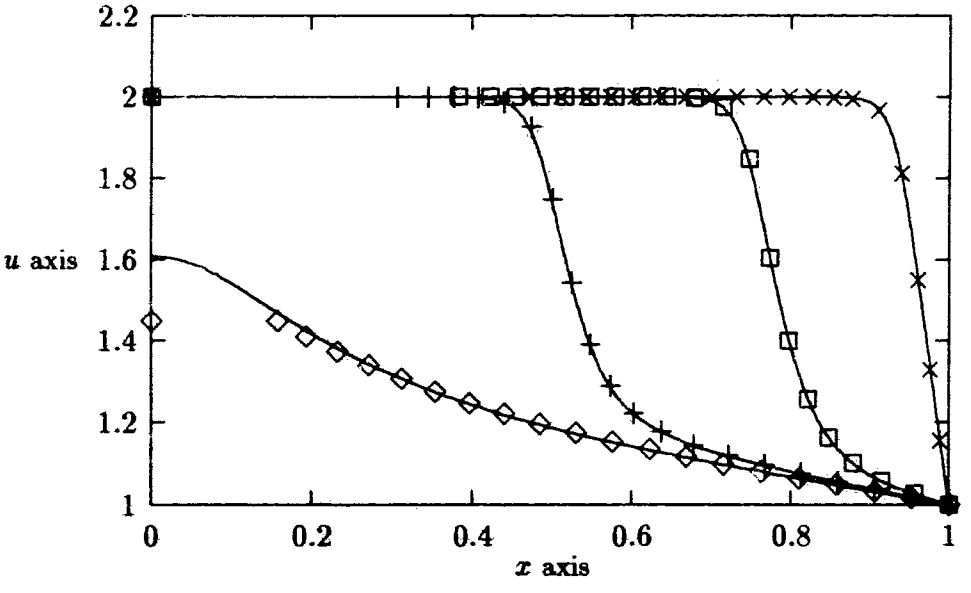

Figure 2.2: Problem I, using method I with $\dot{\theta} = 0$, t= $0.26, 0.27, 0.28, 0.29$; *atol* = 10^{-5}, *rtol* = 10^{-4}, mesh points N = 20

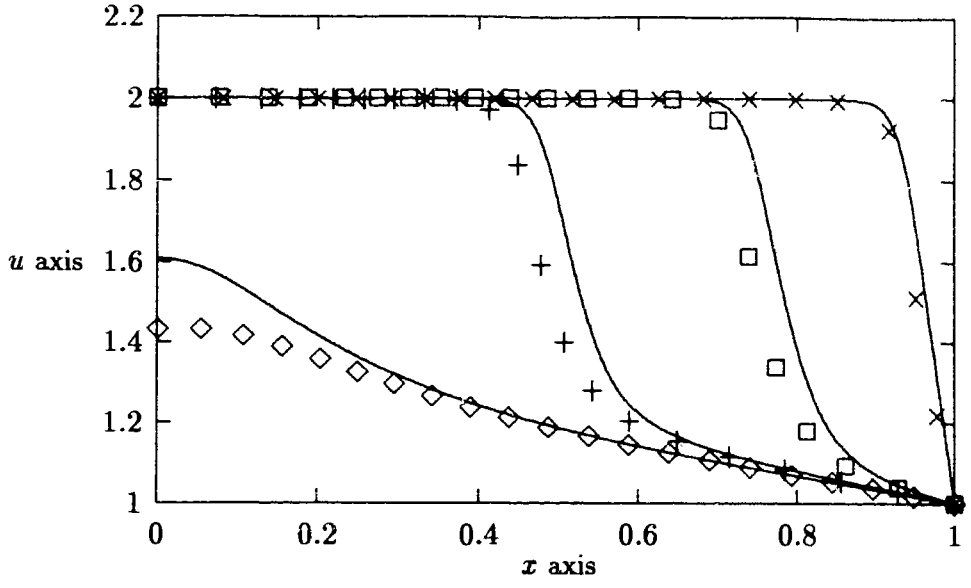

Figure 2.3: Problem I, using method I with $\dot{\theta} \neq 0$, t= $0.26, 0.27, 0.28, 0.29$; *atol* = 10^{-6}, *rtol* = 10^{-5}, mesh points N = 20

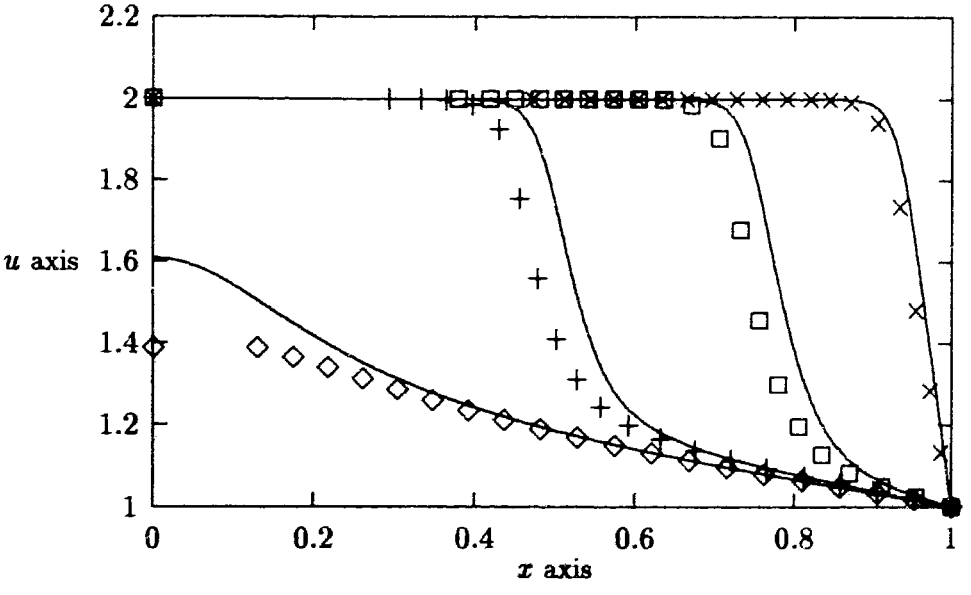

Figure 2.4: Problem I, using method I with $\dot{\theta} = 0$, t= $0.26, 0.27, 0.28, 0.29$; *atol* = 10^{-6}, *rtol* = 10^{-5}, mesh points N = 20

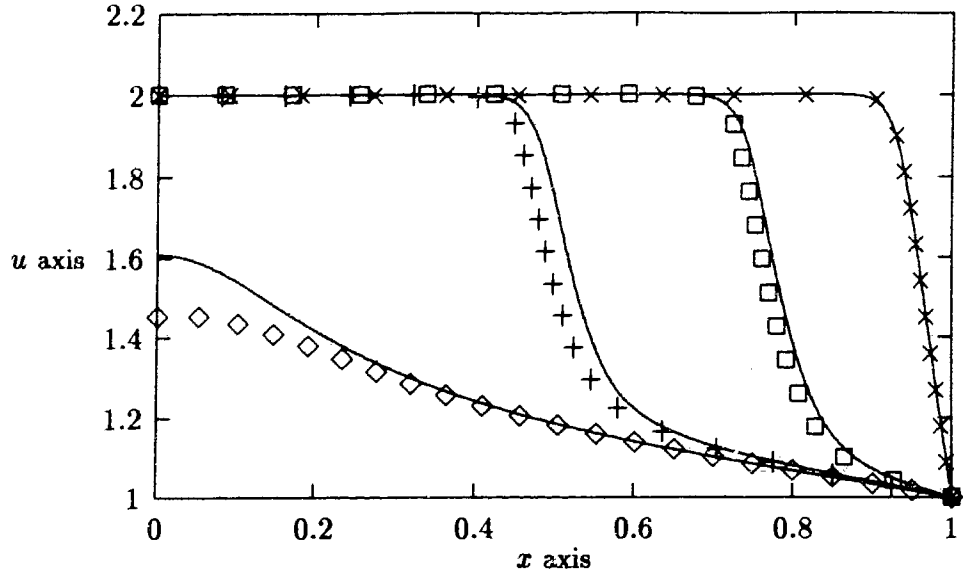

Figure 2.5: Problem I, using method II, t= $0.26, 0.27, 0.28, 0.29$; $atol = 10^{-5}, rtol = 10^{-5}$, mesh points N =20

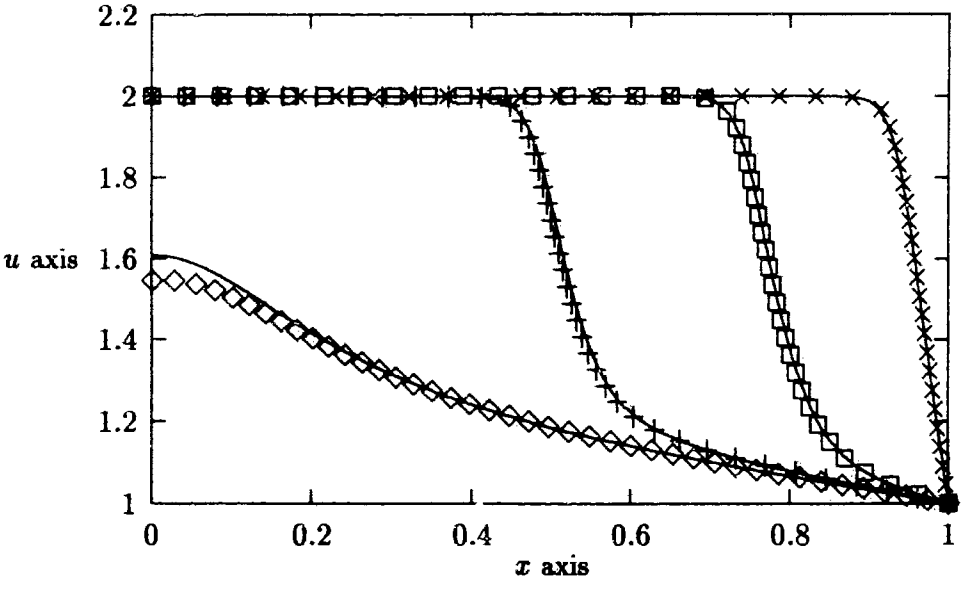

Figure 2.6: Problem I, using method II, t= $0.26, 0.27, 0.28, 0.29$; $atol = 10^{-5}, rtol = 10^{-5}$, mesh points N =40

where $f(u) = u^2/2$. This problem is also often used as a test (occasionally the only test) of mesh selection strategies.

We use $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-4}$ and the smooth initial solution $u_0(x) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x)$. For small times and ε, the exact solution is a pulse that moves in the positive x direction while steepening. The reference solution (solid lines) is computed as in Problem I except with $N = 1000$, *rtol* $= 10^{-6}$ and *atol* $= 10^{-8}$. The solution is shown for time=0.2, 0.4, 0.8, 1.0, 1.4 and 2.0. For **Method II**, $\tau = 10^{-5}$.

Using **Method I** with $\dot{\theta} = 0$, the method easily breaks down due to mesh crossing for $\varepsilon = 10^{-2}$. For example, for $N = 20$, *atol* $= 10^{-3}$ and *rtol* $= 10^{-3}$, breakdown occurs because the second mesh point crosses the left boundary and becomes negative at $t = 0.35$. For $\varepsilon = 10^{-4}$, *atol* $= 10^{-4}$ and *rtol* $= 10^{-5}$ several mesh points cross each other on the interval $[0.519, 0.597]$ at $t = .2$. This is consistent with the theoretical and numerical findings of [CFL86] regarding potential instability of (2.2.20). For $\varepsilon = 10^{-2}$, the presence of the right-hand-side term $\dot{\theta} \neq 0$ now stabilizes the results. Fig. 2.7 and Fig. 2.8 show the solutions and mesh points for $N = 20$ with *atol* $= 10^{-3}$, *rtol* $= 10^{-3}$ and *atol* $= 10^{-4}$, *rtol* $= 10^{-5}$, respectively. The corresponding time-stepping information is given in Table 2.2. The solutions are quite accurate except at the points of zero gradient ($u_x = 0$), where the graph is somewhat higher than that for the reference solution. This same problem occurs using **Method II**; results for the same parameter values are given in Fig. 2.9 and 2.2. Note, too, that the degree of equidistribution is rather poor in this region. Using $N = 40$, these inaccuracies are remedied, and the problem resolution is generally quite satisfactorily. These results are given in 2.10.

For $\varepsilon = 10^{-4}$, the problem causes considerable difficulty. At about $t = 0.2$, a shock layer forms near $x = 0.6$. Setting $N = 20$, using **Method I** with $\dot{\theta} \neq 0$ LSODI stops at the very steep layer at $t = 0.218224$ due to a small step size (*tstep* $= 10^{-6}$ and *nst* $= 955$ for *atol* $= 10^{-4}$, *rtol* $= 10^{-5}$). With (2.4.50) and corresponding parameter values. LSODI also stops, now at $t = 0.341905$ with *tstep* $= 0.0$ and *nst* $= 949$. Using $N = 40$, LSODI is able

t	Fig. 2.7(Method I):78.19(cpu)				Fig. 2.8(Method I):106.76(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.2	18	7	2	0.020208	39	10	3	0.005603
0.4	43	22	2	0.018662	83	21	3	0.009411
0.8	71	32	2	0.020863	165	43	3	0.012868
1.0	78	33	3	0.030399	186	49	4	0.014660
1.4	98	43	1	0.043301	218	57	3	0.022651
2.0	106	44	3	0.115478	234	59	3	0.050789
t	Fig. 2.9(Method II):50.87				Fig. 2.10(Method II):205.36			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.2	88	24	3	0.004330	110	25	2	0.022014
0.4	103	29	2	0.002011	116	26	3	0.045140
0.8	171	57	2	0.008990	122	28	4	0.124988
1.0	197	64	3	0.012833	123	28	4	0.124988
1.4	215	69	3	0.033861	127	29	4	0.184520
2.0	242	79	2	0.055271	130	29	4	0.184520

Table 2.2: Problem II

to progress further but still soon fails. The same difficulty of breakdown when the shock develops can occur for this problem with a high order MFE method using Hermite cubic test functions [HMS82], although other methods are successful [FVZ90, M181].

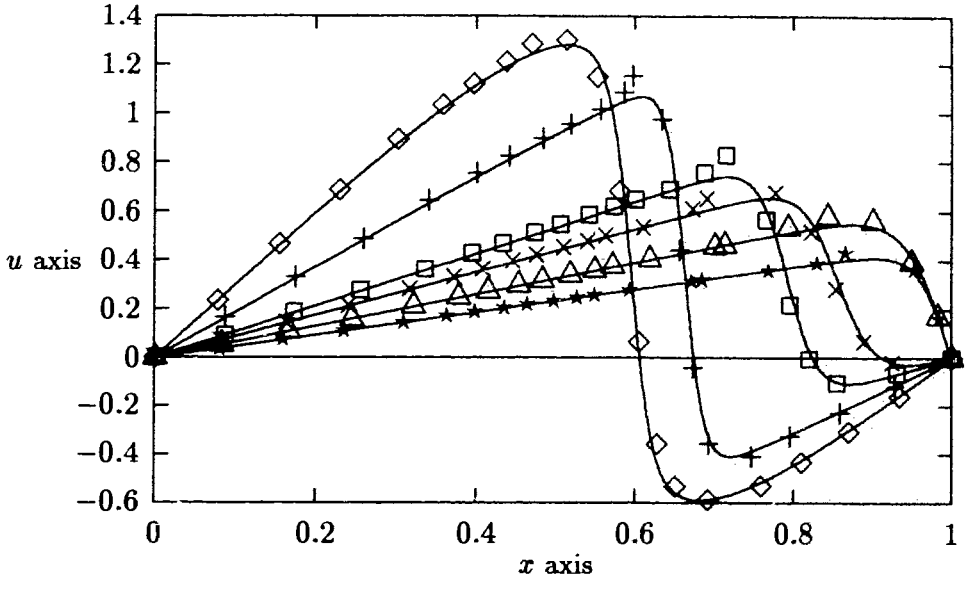

Figure 2.7: Burgers' problem, using method I with $\dot{\theta} \neq 0, \varepsilon = 10^{-2}$, $t = 0.2, 0.4, 0.8, 1.0, 1.4, 2.0$; $atol = 10^{-3}$, $rtol = 10^{-3}$, mesh points $N = 20$.

Problem III: Buckley-Leverett equation.

The third example is the hyperbolic conservative Buckley-Leverett equation

$$u_t + f(u)_x = 0$$

with the non-convex flux function

$$f(u) = \frac{u^2}{u^2 + \frac{1}{2}(1-u)^2}.$$

as in, *e.g.*, [CP79]. The moving mesh methods of [AF86b1] and [GDM81] test the problem with an artificial viscosity term εu_{xx} added (see also [JWB88]).

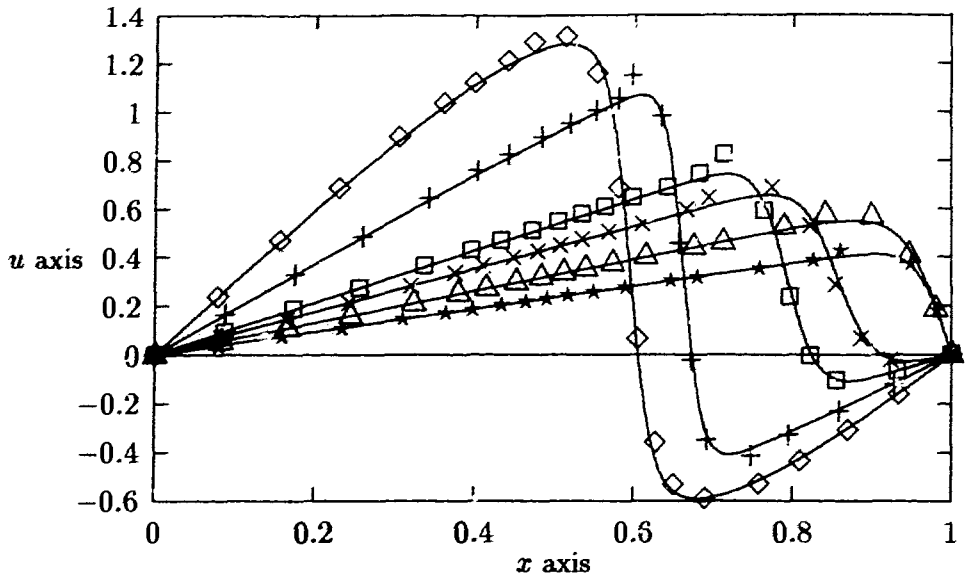

Figure 2.8: Burgers' problem, using method I with $\dot{\theta} \neq 0, \varepsilon = 10^{-2}$, $t = 0.2, 0.4, 0.8, 1.0, 1.4, 2.0$; $atol = 10^{-4}$, $rtol = 10^{-5}$, mesh points $N = 20$.

We consider the continuous initial data condition

$$u(x,0) = \frac{0.1}{0.1+x}, \quad 0 \leq x \leq 1$$

and boundary conditions

$$u(0,t) = 1, \quad u(1,t) = \frac{1}{11},$$

where we express the right boundary condition for LSODI in the form $\dot{u}_N(t) = 0$. The reference solution is determined as in the other two problems, with $N = 500$, $atol = 10^{-8}$ and $rtol = 10^{-6}$, and the solution profile shown for $t = 0.1, 0.2, 0.3, 0.4$. With $N = 20$, results with and without the right-hand-side term in **Method I** are given in Fig. 2.11 and Fig. 2.12, respectively. These numerical solutions are virtually identical and move faster than the reference solution. For **Method II** with $atol = 10^{-4}$, $rtol = 10^{-5}$ and $\tau = 10^{-5}$, LSODI stops due to the steep layer for $t = 0.303228$ with $tstep = 0.0$ and $nst = 575$. (Again, mesh crossing is not a difficulty.) Adding the artificial viscosity term mentioned

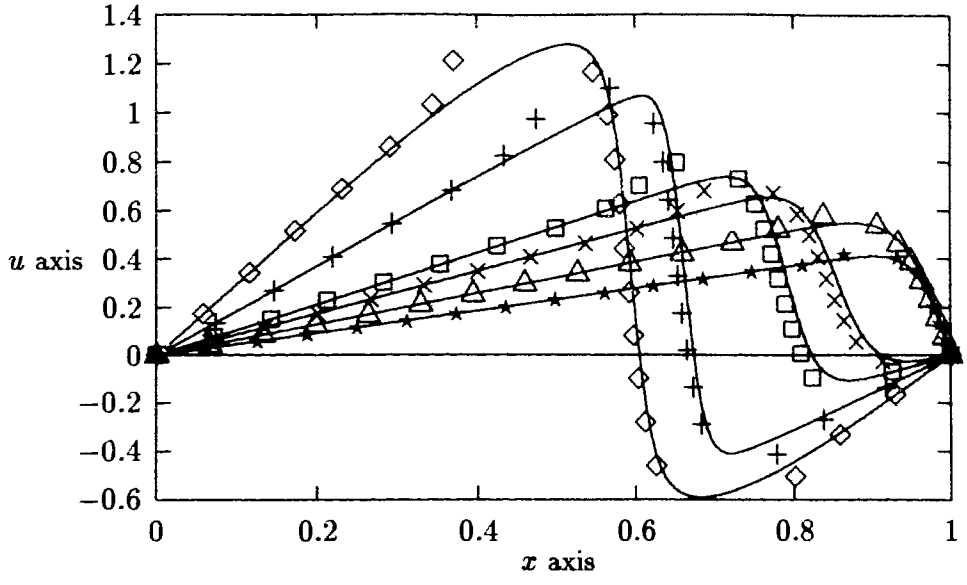

Figure 2.9: Burgers' problem, using method II, $\varepsilon = 10^{-2}$, $t = 0.2, 0.4, 0.8, 1.0, 1.4, 2.0$; *atol* $= 10^{-4}$, *rtol* $= 10^{-5}$, mesh points $N = 20$.

above, here with $\varepsilon = 10^{-4}$, the problem is solved more satisfactorily than before, using *atol* $= 10^{-4}$, *rtol* $= 10^{-5}$. The results, given in Fig. 2.13, are qualitatively unchanged for smaller tolerances, like for example *atol* $= 10^{-5}$, *rtol* $= 10^{-6}$ (see Table 2.3). The scheme developed in [AF86b1] has no difficulty for this problem when solved as a parabolic PDE using real viscosity with $\varepsilon = 10^{-3}$. However, it is interesting to wonder when a difficulty arises when solving hyperbolic PDEs because the scheme is nonconservative when viewed as a scheme for solving the moving mesh PDE.

Problem IV: Heat conduction problem

As a fourth example, we consider a heat conduction problem,

$$u_t + f(x,t) = \mu u_{xx}, \quad -3 < x < 3, \ t > 0,$$

where the initial conditions, Dirichlet boundary conditions, constant diffusion μ, and source

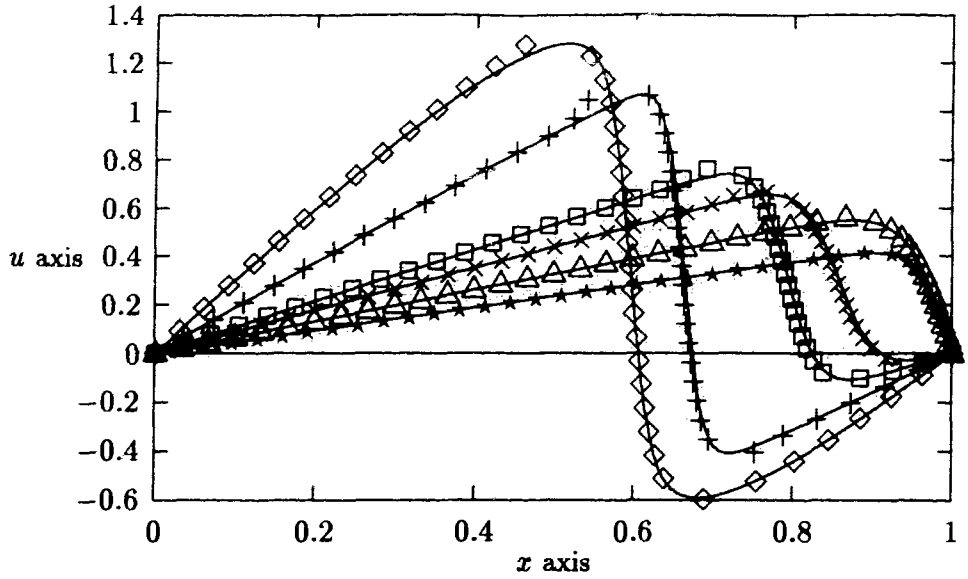

Figure 2.10: Burgers' problem, using method II, $\varepsilon = 10^{-2}$, $t = 0.2, 0.4, 0.8, 1.0, 1.4, 2.0$; *atol* $= 10^{-4}$, *rtol* $= 10^{-5}$, mesh points $N = 40$.

t	Fig. 2.11(Method I):436.29(cpu)				Fig. 2.12(Method I):315.05(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.1	21	4	4	0.009126	22	4	4	0.008524
0.2	32	6	4	0.009126	33	5	4	0.008524
0.3	43	8	4	0.009126	48	7	4	0.006048
0.4	57	13	4	0.005544	74	13	3	0.003204
t	Fig. 2.13(a)(Method II):407.85				Fig. 2.13(b)(Method II):713.21			
0.1	60	14	3	0.014108	75	17	3	0.012381
0.2	67	15	3	0.014108	85	18	3	0.009835
0.3	75	19	3	0.013226	95	24	3	0.007773
0.4	606	307	3	0.000147	883	372	2	0.000415

Table 2.3: Problem III

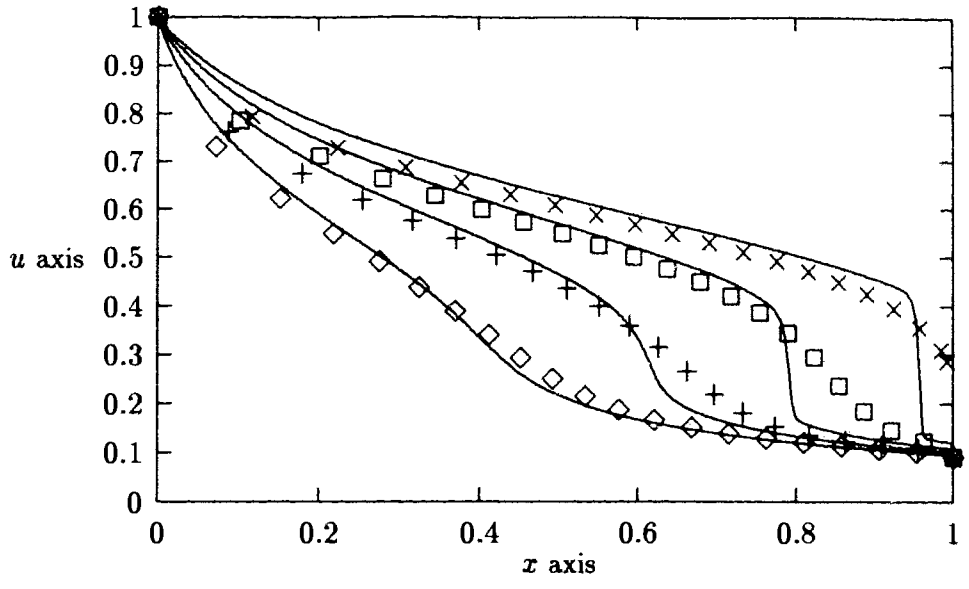

Figure 2.11: Buckley-Leverett problem, using method I with $\dot{\theta} \neq 0$, $t = 0.1, 0.2, 0.3, 0.4$; *atol* $= 10^{-5}$, *rtol* $= 10^{-6}$, mesh points $N = 20$.

term f are chosen such that the exact solution is

$$u(x,t) = tanh[r_1(x-1) + r_2 t], \quad -3 \leq x \leq 3, \ t \geq 0.$$

This problem differs from the previous three examples, since the solution travels in the negative x direction when r_1 and r_2 are positive. We solve this problem for $r_1 = r_2 = 5$ and $\mu = 10^{-4}$, and show the results at $t = 0.05, 0.50, 1.00, 1.50, 2.00$. Coyle, Flaherty and Ludwig studied this problem [CFL86].

We test $\tau = 10^{-5}$ and *atol* $=$ *rtol* $= 10^{-4}$. Fig 2.14 shows that computed solutions are oscillating, and cpu time is 209.85. For $\tau = 10^{-3}$, the results have little improvement as shown in Fig. 2.15, and cpu time is 79.35. When increasing τ to 10^{-1}, Fig. 2.16 shows that the results are better than for $\tau = 10^{-3}$, but the computed solutions are above the exact solutions, and cpu time is 31.14. For $\tau = 1.0$, the accuracy of the computed solutions is quite good as shown in Fig. 2.17, and cpu time is 24.28.

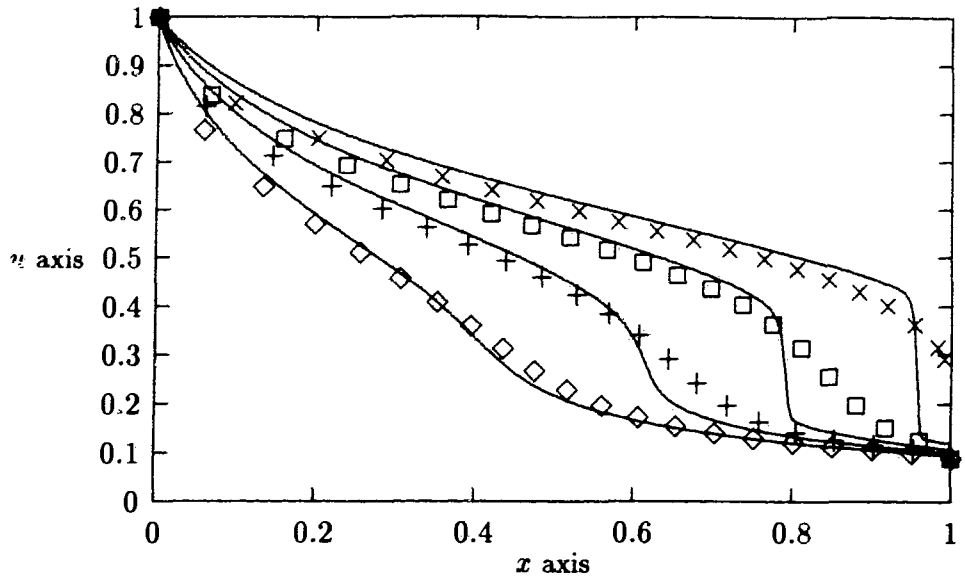

Figure 2.12: Buckley-Leverett problem, using method I with $\dot{\theta} = 0$, t=0.1, 0.2, 0.3, 0.4; *atol* $= 10^{-5}$, *rtol* $= 10^{-6}$, mesh points $N = 20$.

t	Fig. 2.14(Method II):209.85(cpu)				Fig. 2.15(Method II):79.35(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.05	55	16	2	0.018596	48	13	2	0.007598
0.50	88	30	2	0.023122	81	23	2	0.018953
1.00	213	87	1	0.001667	120	32	3	0.031767
1.50	286	114	2	0.014560	149	41	3	0.017471
2.00	340	137	2	0.011856	200	58	3	0.004397
t	Fig. 2.16(Method II):31.14				Fig. 2.17(Method II):24.28			
0.05	11	4	3	0.008237	6	3	3	0.030933
0.50	35	8	3	0.031682	23	6	3	0.029440
1.00	49	11	3	0.030759	41	8	4	0.025956
1.50	67	17	3	0.040294	57	11	3	0.042617
2.00	86	21	4	0.038142	72	15	4	0.026683

Table 2.4: Problem IV

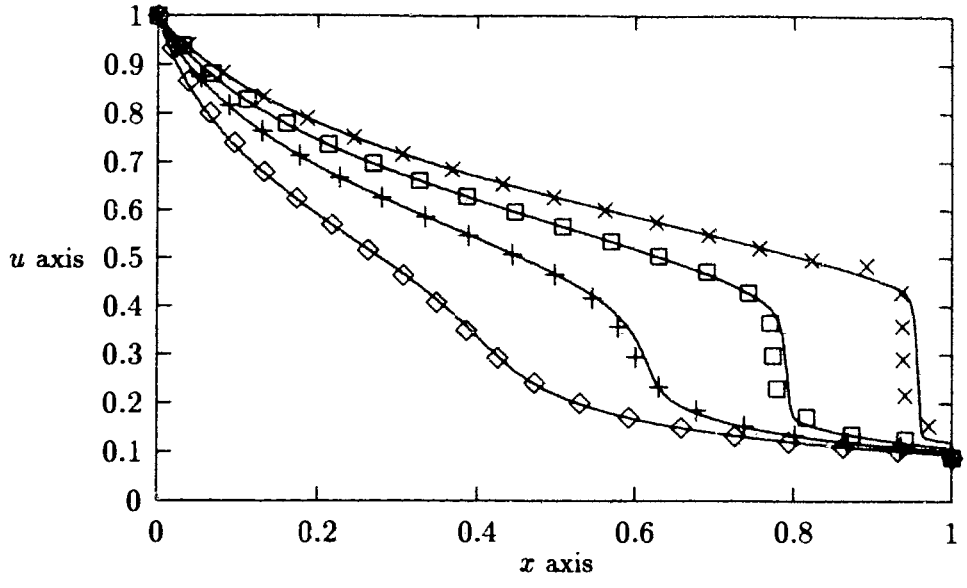

Figure 2.13: Buckley-Leverett problem, using method II, t=0.1, 0.2, 0.3, 0.4; mesh points $N = 20$, (a) $atol = 10^{-4}$, $rtol = 10^{-5}$ (Results for (b) $atol = 10^{-5}$, $rtol = 10^{-6}$ indistinguishable.)

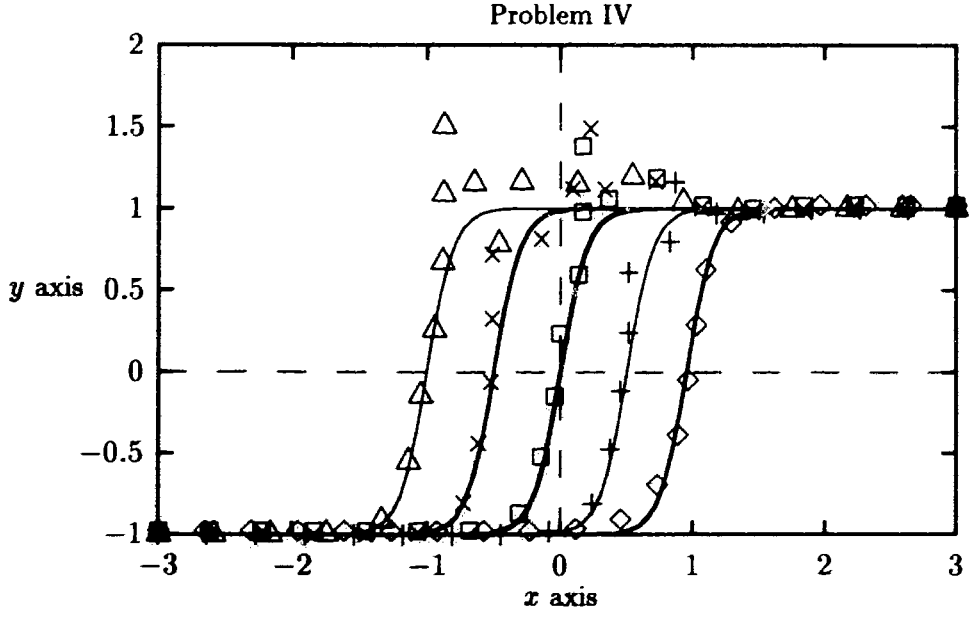

Figure 2.14: using method II, $\tau = 10^{-5}$ and $atol = rtol = 10^{-4}$, $cpu = 209.85$.

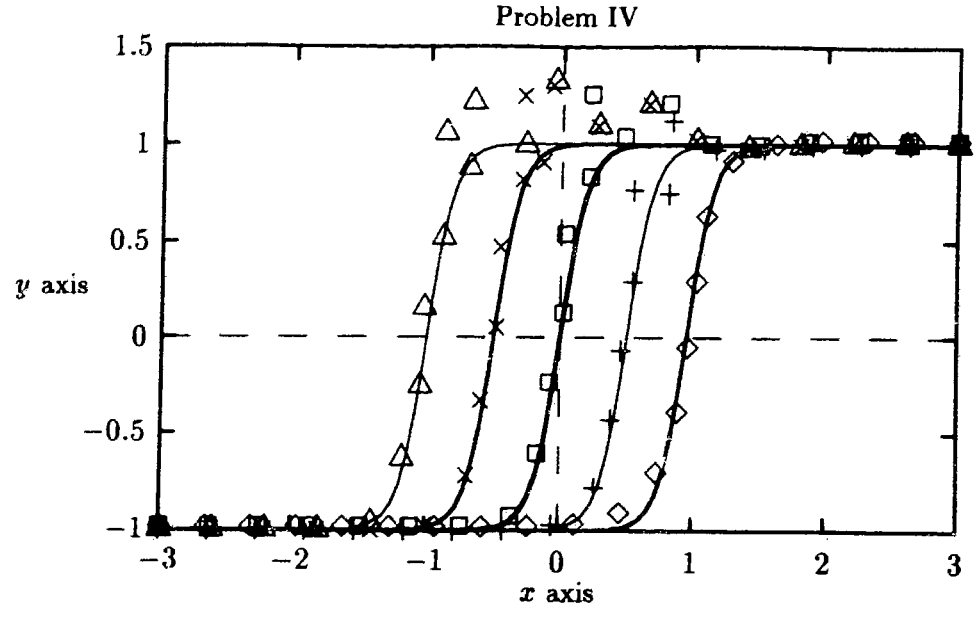

Figure 2.15: using method II, $\tau = 10^{-3}$ and $atol = rtol = 10^{-4}$, $cpu = 79.35$.

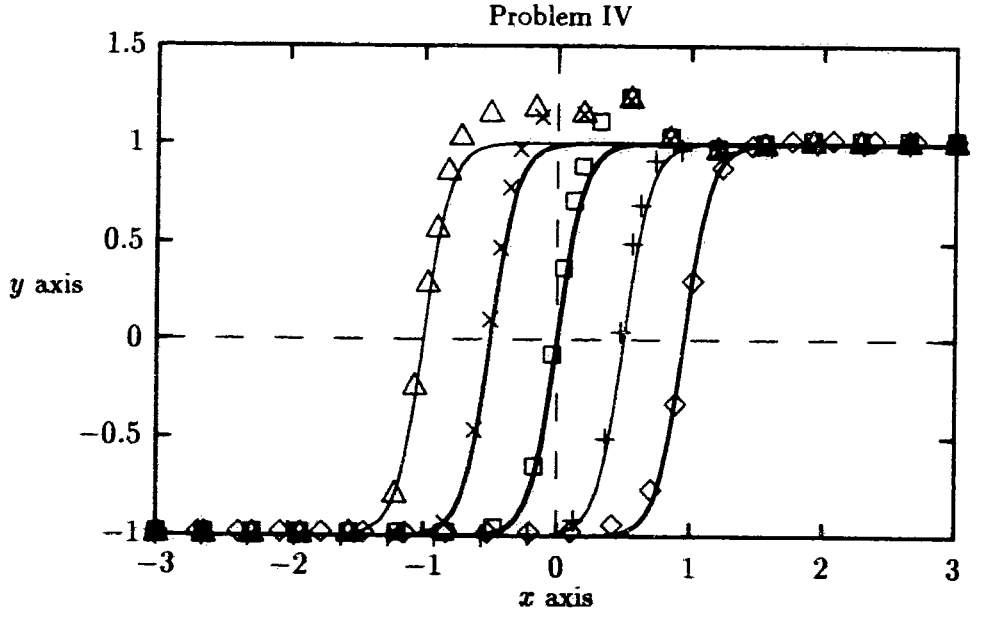

Figure 2.16: using method II, $\tau = 10^{-1}$ and $atol = rtol = 10^{-4}$, $cpu = 31.14$.

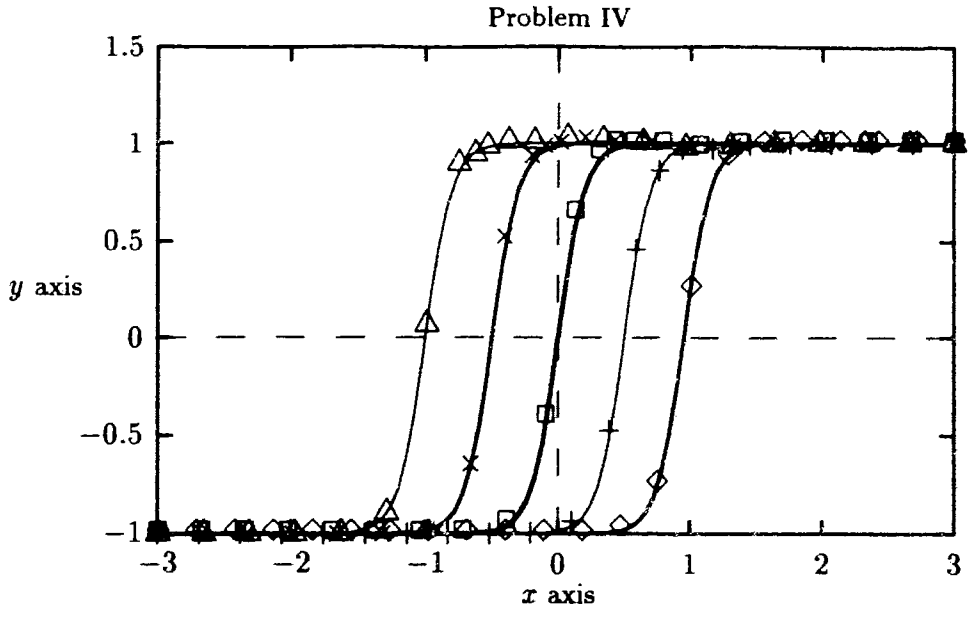

Figure 2.17: using method II, $\tau = 1.0$ and $atol = rtol = 10^{-4}$, $cpu = 24.28$.

2.6 Conclusions

We have presented a new formulation of the equidistribution strategy in terms of a PDE. Previously, authors who have explicitly used equidistribution have generally developed moving mesh procedures which make use of (2.1.6), the integrated or weak form of the conservative integral. We intend to develop further robust moving mesh strategies based directly upon the differential form (2.2.16) or (2.2.29). Here, our intention has been to present some simple ones. The purpose has not been to give extensive numerical results or a detailed comparison with other methods, which will be done in Chapter 3. Nevertheless, the results indicate that the schemes given here, with simple improvements such as smoothing of the mesh (for Problem II) when necessary, should prove competitive with those which have been recommended by others [FVZ90]. Use of conservative-type schemes to approximate the PDEs is natural and probably essential in many contexts. The importance of the right-hand-side term of (2.2.16) is unclear, and we have included numerical results for $\dot{\theta} = 0$ partly to determine the effect and partly because this corresponds to what many previous implementations have

used.

The numerical methods used here are quite simple and presented mainly for illustrative purposes. Constructing more robust moving mesh methods could well require the incorporation of regularization terms as in [DD87, FVZ90, MA84, MI83, VBFZ88], and possibly a more complicated monitor function, an obvious choice being some combination of arclength and curvature. However, while using the arclength monitor function can limit the number of mesh points which are placed in the transition region, strong nonlinearities which arise using a curvature monitor function can also cause computational difficulties [BV89].

Efficient ways to produce the moving mesh equations using this approach, particularly for higher-order systems (2.1.4) and for the higher-dimensional form of (2.2.17) or (2.2.29), remain to be investigated. Still, it is important to realize that the scheme is not plagued with mesh crossings the way most other simple moving mesh schemes are. When the PDE (2.2.16) (including the right-hand side) is approximated, we have found very little difficulty of this type. In one case (Burgers' equation with different initial conditions than given here) (2.4.42) gave mesh crossing with a large tolerance, but this was fixed when the tolerance was reduced. While there is no need to add penalty functions for this reason, it may still be necessary to perform a mesh smoothing to prevent problem stiffness when steep solution layers occur (as was the difficulty in Problem II in section § 2.4). Obviously, a desirable ultimate goal is the development of a robust scheme with minimal requirements for a user to select contentious problem-dependent parameters.

This moving mesh PDE interpretation can be used to understand stability properties for moving mesh strategies and extends the understanding of the stability properties as given in [CFL86]. While the stability issue for methods based upon equidistribution is a very complicated one, and there is no doubt that a complicated interaction takes place between the PDE (2.1.3) and the mesh PDE (2.2.16) or (2.2.29), we expect that this viewpoint will be used to develop a deeper understanding of stability properties for currently used methods which have proven reliable. It is important to realize how many moving mesh methods

are based upon equidistribution, making it possibly the single most important concept in the development of moving mesh methods. Many of these methods use equidistribution explicitly, like [AF86a, HL86, PE87] and those in [CFL86], and many of these often have stability difficulties [CFL86]. There are also the ones like the moving finite element methods [MM81, HSM83] and the elliptic grid generation methods [MD88] which have been developed from another viewpoint but for which equidistribution has turned out to play a role - just *how* fundamental is unclear at this stage. The considerable success of some of these may be due in part to the fact that the moving mesh PDE (2.2.16) is solved implicitly, so that inadequate approximations from using nonconservative schemes or from ignoring the important right-hand-side term have been circumvented. Of course another underlying issue of critical importance is that of deciding what monitor function to use, and it is unrealistic to expect that a single choice for M would serve as a panacea for most problems.

Chapter 3

A Study of Moving Mesh Methods

3.1 Introduction

In this Chapter, we will further study some of the moving mesh methods discussed in Chapter 1. While the analysis of the error and the convergence for moving mesh methods are very difficult problems, we will study them in the future. Here we consider the performance in practice of the most promising methods. Recall that methods based upon equidistribution principles have been studied by many people, *e.g.* [BO73, CFL86, DO72, WH79] and that the powerful moving finite element methods of Miller *et al.* also implicitly satisfy equidistribution principles in certain cases. Related methods will be central in our study here.

Although general comparison of moving mesh methods has been seldom done in the past, one useful study was made by Furzeland *et al.* [FVZ88]. They compared three moving meshes methods for one-dimensional pdes, *viz.*, one moving finite element method of Miller & Miller, and two moving finite difference methods, one by Petzold and by the other by Dorfi & Drury. Zegeling and Blom [ZB90] have more recently evaluated the gradient-weighted moving finite element methods of Miller [MI83]. The gradient-weighting is introduced into the MFE method by Miller [MI83] along with a new penalty function to prevent mesh points

becoming too close to one another in the steepest part of the solution layers. Obviously, the penalty function and gradient-weighting are two key factors in moving finite element methods. A new penalty function can be important to balance the gradient-weighting of the modified moving finite element methods, but it is unclear precisely what the relationship between the penalty function and the gradient-weighting is. In this chapter we study the penalty function and the gradient-weighted strategy both in MFE [MI81] and in GWMFE[MI83]. We also study the moving finite difference method introduced earlier (see equation (1.2.25)), for which mesh points are chosen by a discrete equidistribution.

3.2 Moving mesh methods

3.2.1 Moving finite element methods

Two versions of moving finite element methods have been introduced by Miller *et al.*, one the original MFE [MM81] with a piecewise linear basis function, and the other the gradient-weighted MFE. For the latter, the gradient weighted function used to prevent too many mesh points from entering steep layers [MI83] gives an ODE system which can be very difficult to solve.

MFE (1)

In this section we briefly review Miller's unmodified moving finite element method (for details, see page 18).

Consider the partial differential equation

$$\dot{u} = L(u), \qquad t > 0, \quad x \in [a, b] \tag{3.2.1}$$

where $L(u)$ is an nonlinear spatial differential operator. The solution of equation (3.2.1) is solved by a finite element methods. The solution is expressed as

$$U(x,t) = \sum_{i=1}^{N} U_i \alpha_i(x,t), \tag{3.2.2}$$

where the non-uniform mesh points $\{x_i\}$ are allowed to move according to the profile of the solution.

The residual of the partial differential equation is defined for the L_2-norm

$$\| \dot{U} - L(U) \|_2^2 . \tag{3.2.3}$$

The solutions U_i at mesh points x_i are determined by minimizing the residual with respect to the velocity of the amplitude $\dot{U}_i$. We have the standard Galerkin finite element equation (1.2.62) for $\dot{U}_i$. For the determination of the mesh points x_i, Miller & Miller [MM81] minimize the residual of the partial differential equation once again, but with respect to the velocity of mesh points $\dot{x}_i$. This gives equation (1.2.63).

The mass matrix $A(Y)$ in the moving finite element system (1.2.64) is block tridiagonal. Wathen [WA86] shows that $A(Y)$ is singular only in two cases: (i) $m_i = m_{i+1}$ which is called parallelism. Geometrically, it means that the three neighboring points (x_{i-1}, U_{i-1}), (x_i, U_i) and (x_{i+1}, U_{i+1}) are collinear, lying on a straight line. In this case, the basis functions $\{\alpha_i\}$ and $\{\beta_i\}$ are not linearly independent. (ii) Mesh points are too close to one other, which makes the mass matrix A become ill-conditioned.

To overcome singular or ill-conditioned A, Miller introduces a penalty function for the mesh equations. That is, he minimizes equation (1.2.68) with respect to $\dot{x}_i$, which makes the mass matrix A positive definite [MI81]. We refer to this moving finite element method as **MFE (1)**.

Gradient-weighted MFE method (GWMFE)

Meshes driven by the L_2 norm in the minimization of the residual $\dot{U} - L(U)$ move most of the meshes into the steepest region of the solution to minimize the errors since large errors arise in the steepest region [MI83]. Miller uses the gradient-weighted L_2 minimization in the normal direction of the solution to try to '*de-emphasize*' the steep regions of the solution.

The gradient-weighted function $w(u_x) = \frac{1}{\sqrt{1+u_x^2}}$ is constant on each subinterval $[x_{i-1}, x_i]$

since $w(m_i) = \frac{1}{\sqrt{1+m_i^2}}$ for $x_{i-1} \leq x \leq x_i$.

The test functions $\{\alpha_i\}$ and $\{\beta_i\}$ are replaced by $\{\alpha_i w\}$ and $\{\beta_i w\}$ in equation (1.2.62) and (1.2.63), and Miller minimizes the same residual of the PDEs as for MFE (1). A system of $2N$ ODEs for the $2N$ unknowns $\{U_i\}$ and $\{x_i\}$ arises, *viz.*,

$$\sum_{i=1}^{N} < \alpha_i, \alpha_j w > \dot{U}_i + < \beta_i, \alpha_j w > \dot{x}_i =< L(U), \alpha_j w >, \tag{3.2.4}$$

$$\sum_{i=1}^{N} < \alpha_i, \beta_j w > \dot{U}_i + < \beta_i, \beta_j w > \dot{x}_i =< L(U), \beta_j w > \tag{3.2.5}$$

for $j = 1, \cdots, N$. The matrix form is $\tilde{A}(Y)\dot{Y} = \tilde{F}(Y)$. The mass-matrix $\tilde{A}$ is singular too when parallelism $m_i = m_{i+1}$ occurs and when $\triangle x_i = 0$ or $\triangle U_i = 0$.

In order to prevent the matrix $\tilde{A}$ from becoming singular, Miller suggested using '*arclength viscosities*' and '*arclength spring forces*' at each subinterval to balance the gradient-weighting in the moving finite element method. He then minimizes the residual of the PDEs with this gradient-weighting and new penalty function,

$$||| \dot{U} - L(U) |||_2^2 + \sum_{i=1}^{N+1} (\varepsilon_i \dot{l}_i - S_i)^2 \tag{3.2.6}$$

with respect to the velocity of the amplitude $\dot{U}_i$ and the velocity of the meshes $\dot{x}_i$. Here the gradient-weighted L_2-norm is defined by $||| \dot{U} - L(U) |||_2^2 := \int_a^b (\dot{U} - L(U))^2 w dx$, $\varepsilon_i^2 := \frac{A^2}{l_i}$, $\varepsilon_i S_i := \frac{B^2}{l_i^2}$, where A and B are problem-dependent constants, (B is usually chosen as 0), l_i is an arclength for the subinterval $[x_{i-1}, x_i]$, and since U_x is constant on each subinterval for a piecewise linear basis,

$$l_i = \sqrt{(\triangle x_i)^2 + (\triangle U_i)^2} = \int_{x_{i-1}}^{x_i} \sqrt{1 + m_i^2} dx = \int_{x_{i-1}}^{x_i} \sqrt{1 + U_x^2} dx. \tag{3.2.7}$$

MFE (2)

GWMFE uses a gradient-weighted L_2 norm and an arclength-type penalty function to prevent the meshes from becoming too close in the steepest region of the solution and to prevent parallelism. The gradient-weighted function has little effect in the non-steep regions of the

solutions, and at non-steep parts GWMFE acts as MFE(1). In the steepest regions, the second equation (3.2.5) of GWMFE approaches the first equation (1.2.62) of MFE, *i.e.*,

$$< \alpha_i, \beta_j w > \rightarrow < \alpha_i, \alpha_j >,$$

$$< \beta_i, \beta_j w > \rightarrow < \beta_i, \alpha_j >,$$

and

$$< L(U), \beta_j w > \rightarrow < L(U), \alpha_j >,$$

as $u_x \rightarrow \infty$, where $w = \frac{1}{\sqrt{1+u_x^2}}$ and

$$\lim_{u_x \rightarrow \infty} \frac{u_x}{\sqrt{1+u_x^2}} = 1.$$

We shall test a moving finite element method using the standard L_2 norm of the residual, but using the *arclength-type* penalty function of GWMFE as used in (3.2.6). We minimize the function

$$\| \dot{U} - L(U) \|_2^2 + \sum_{i=1}^{N+1} (\varepsilon_i \dot{l}_i - S_i)^2 \tag{3.2.8}$$

with respect to $\dot{U}_i$ and $\dot{x}_i$. The penalty function has an important effect both in GWMFE and in MFE for controlling the mesh points and preventing the mass-matrix of the systems of ODEs from becoming singular. We refer to this moving mesh strategy as **MFE (2)**.

Notes for implementation of moving finite element methods

The approximation of the second derivative u_{xx} has to be considered carefully in moving finite elements methods, since u_{xx} is a δ-function at each mesh point when using piecewise linear bases. Miller *et al.* use a mollification scheme for second order terms, and this results in

$$< u_{xx}, \alpha_i > = (m_{i+1} - m_i) \tag{3.2.9}$$

$$< u_{xx}, \beta_i > = -(m_{i+1} - m_i)(\frac{m_{i+1} + m_i}{2}) \tag{3.2.10}$$

$$< u_{xx}, \alpha_i w > = \ln(\sqrt{1 + m_{i+1}^2} + m_{i+1}) - \ln(\sqrt{1 + m_i^2} + m_i) \tag{3.2.11}$$

$$< u_{xx}, \beta_i w > = \sqrt{1 + m_i^2} - \sqrt{1 + m_{i+1}^2}. \tag{3.2.12}$$

For the implementation of (3.2.11) and (3.2.12), roundoff problems have to be considered in the implementation, since formulas (3.2.11) and (3.2.12) are susceptible to a great loss of relative accuracy if m_i and m_{i+1} are small, and (3.2.11) also gives problems if either m_i or m_{i+1} is large and negative. Miller *et al.* deal with $< u_{xx}, \beta_i w >$ as

$$\sqrt{m_i^2+1} - \sqrt{m_{i+1}^2+1} = \frac{m_i^2}{1+\sqrt{m_i^2+1}} - \frac{m_{i+1}^2}{1+\sqrt{m_{i+1}^2+1}}$$

to prevent loss of the relative accuracy when m_i or m_{i+1} is small. For the problems of large and negative m_i or m_{i+1}, $< u_{xx}, \alpha_i w >$ is approximated using $sign(m_i)\ln(|m_i| + \sqrt{m_i^2+1})$ instead of $\ln(m_i + \sqrt{m_i^2+1})$.

Blom *et al.* evaluate $\ln(m_i + \sqrt{m_i^2+1})$ by a Taylor expansion if $\eta = \frac{m_i}{\sqrt{m_i^2+1}}$ is small, taking

$$\ln(m_i + \sqrt{m_i^2+1}) = \frac{1}{2}\ln(\frac{1+\eta}{1-\eta}) \approx \eta + \frac{1}{3}\eta^3 + \frac{1}{5}\eta^5 + \frac{1}{7}\eta^7. \tag{3.2.13}$$

In fact, since w is a constant on each subinterval when using piecewise linear bases in moving finite element methods,

$$\begin{aligned} < u_{xx}, \alpha_i w > &= \int_{x_{i-1}}^{x_{i+1}} u_{xx}\alpha_i w dx \\ &= \int_{x_{i-1}}^{x_i} u_{xx}\alpha_i w dx + \int_{x_i}^{x_{i+1}} u_{xx}\alpha_i w dx \\ &= w_i \int_{x_{i-1}}^{x_i} u_{xx}\alpha_i dx + w_{i+1}\int_{x_i}^{x_{i+1}} u_{xx}\alpha_i dx \\ &= -w_i m_i + w_{i+1} m_{i+1} \end{aligned} \tag{3.2.14}$$

which is the same as only using the first term of a Taylor series (3.2.13).

3.2.2 Method III

In this section, we briefly discuss method III, the moving mesh equation that Dorfi and Drury [DD87] derived based upon the equidistribution principle (for details, see page 9). The main idea of method III is that the mesh equidistribution is smoothed both in spatial and temporal variables. Verwer *et al.* [VBFZ88] studied this method. Blom and Verwer

[BV89] compared the arclength monitor with the curvature monitor used in the moving mesh equation. They found that using an arclength monitor makes the system of moving mesh equations easier to solve than using a curvature monitor.

As discussed in equation (1.2.25) of Chapter 1, the mesh equation for **Method III** is

$$\frac{(\tilde{n}_{i-1} + \tau\frac{d\tilde{n}_{i-1}}{dt})}{M_{i-1}} = \frac{(\tilde{n}_i + \tau\frac{d\tilde{n}_i}{dt})}{M_i} \qquad \tau \geq 0 \qquad (2 \leq i \leq N-1) \qquad (3.2.15)$$

where $\tilde{n}_i$ is smoothing the point concentrations as defined in (1.2.24), n_i is the point concentrations as defined in (1.2.20) and k is the parameter in (1.2.22). The monitor M is implicitly assumed to be unchanged from the last time to the current time over the time interval τ. Dorfi and Drury approximate an arc-length monitor $M = \sqrt{1+u_x^2}$ on each subinterval $[x_i, x_{i+1}]$ by $M_i = \sqrt{1+(\frac{u_{i+1}-u_i}{x_{i+1}-x_i})^2}$. The parameter τ acts as a delay factor, which we have seen serves like a time stepsize. The temporal smoothing term

$$\tau\frac{d\tilde{n}_i}{dt}$$

in (3.2.15) can be interpreted as an **artificial viscosity** term

$$\nu\frac{d}{dx^2}(\dot{n})$$

that smooths out discontinuities in the mesh flow.

3.3 Time integration for ODEs

In our experience, the numerical time integration component is a very important factor in determining reliability of the numerical solutions and parameter values for moving mesh methods. In this section, we discuss implement aspects to this time integration of the system of ordinary differential equations that arises. To better understand moving mesh methods and to fairly compare the methods, here for all methods we use the same stiff ODE solver, LSODI, without any modification, although in certain cases some special strategies such as a preconditioning scheme in Newton's method and mesh crossing test might be required.

It is interesting to see how others have dealt with the numerical integration problem. Hrymak *et al.* have tested a moving finite element method using LSODI [HMW86]. They added another error norm for the corrector in the predictor-corrector algorithm of LSODI in order to check whether or not it was necessary to reduce the time-step because of possible mesh crossings. Carlson and Miller developed the 1-D code GWMFE1DS for their moving finite element methods, including a gradient weighted MFE. A second-order accurate Diagonally-Implicit Runge-Kutta method (DIRK2) which is an "A-stable" method, has been used as the time integrator for the system of ODEs. A block diagonally preconditioning scheme has been used for Newton's method. In GWMFE1DS, Carlson and Miller test a relative error tolerance (usually 10%) on mesh spacing and check for negative Δx_i in predicted values. An evaluation of the GWMFE was reported in [ZB90]. Zegeling and Blom tested the GWMFE1DS code with their own criterion for both the time error and the convergence of the Newton process. They required the following conditions:

$$\|\nu/tol\| < 1,$$

and

$$\max_{x_i} \frac{|\nu(x_{i+1}) - \nu(x_i)|}{(x_{i+1} - x_i)\rho} < 1$$

where ν is a vector either containing an estimate of the time error or the last correction in the Newton process, and ρ is a user-defined parameter. Furzeland *et al.* tested the three moving mesh methods (see [FVZ90]) using the stiff solver SPGEAR within the SPRINT package. SPGEAR is based on the LSODI code of Hindmarsh [HI80] for solving differential algebraic systems (DAEs), and it contains both the family of Admas methods up to order 12 and the family of GEAR/backward difference formulas (BDF) methods of up to order 5. Interestingly, Furzeland *et al.* [FVZ90] were unable to successfully run GWMFE with LSODI.

3.4 Numerical results

The reduced ODE system from MOL is a linearly implicit system of first order ODE. We use LSODI to solve the initial value problem

$$a(t,y)\frac{dy}{dt} = b(t,y).$$

If a is singular, this is a differential-algebraic system. The user needs to provide the subroutines for computing the residual function and $a(t,y)$. The initial values of $\frac{dy}{dt}$ and the Jacobian are internally generated by LSODI. The parameter *rtol* is the relative tolerance parameter, and *atol(i)* is the absolute tolerance parameter. The estimated local error in $y(i)$ will be controlled so as to be roughly less than

$$ewt(i) = rtol * abs(y(i)) + atol(i).$$

Thus the local error test passes if, in each component, either the absolute error is less than *atol(i)*, or the relative error is less than *rtol*. Actual global errors may exceed these local tolerances. t_0 The initial value of the independent variable is t_0, and *tout* is the first point where output is desired. The difference $tout - t_0$ is the outside stepsize of LSODI. Recall that *nst* and *nje* are respectively the number of steps and number of Jacobian evaluations taken by LSODI up to the time given, and *nqn* and *tstep* are respectively the order of the last successful method and the last successful stepsize.

Problem I (A scalar reaction-diffusion problem)

For details see page 47. For this problem we use $tout - t_0 = 10^{-3}$ for LSODI, since LSODI fails to solve the problem for $tout - t_0 = 10^{-2}$. The initial value is $t_0 = 0.25$.

Results for MFE(1)

As we tested in Chapter 2, this problem is quite sensitive to the tolerance (*atol* and *rtol*). We have tried many values for c_1, c_2 and δ in equation (1.2.68), where the parameter c_1 is a coefficient of the '*internodal viscosity*' terms, the parameter c_2 is a coefficient of the '*internodal spring forces*' terms, and the parameter δ is a user-defined minimum mesh

distance and is also used for the activation energy [MI81]. Although we test some values used in [FVZ90] and [HMW86], it is hard to find good ones. MFE(1) is very sensitive to these parameters.

We first test the method with $c_1 = 10^{-3}$ (and $c_1 = 10^{-2}$), $c_2 = 10^{-4}$, $\delta = 5 * 10^{-5}$ used in [HMW86] and $atol = rtol = 10^{-4}$(for $atol = rtol = 10^{-3}$, the computed solution moves too fast - see page 47). The results are very poor.

Then we use the values $c_1 = 0.1$, $c_2 = 5 \times 10^{-4}$, $\delta = 10^{-4}$, which succeed in solving Burgers' problem (as we shall see later) and $atol = rtol = 10^{-5}$. At the time $t = 0.26$, the numerical solution is quite good, but for later times ($t = 0.27, 0.28, 0.29$), the numerical solutions move slower than the reference solutions. The results are given in Fig. 3.1, where the cpu time is 77.76. If the tolerances (*atol* and *rtol*) are increased to 10^{-4}, the numerical solution moves too fast initially and jumps to 2.0 at $t = 0.26$. But for $t = 0.27, 0.28, 0.29$, the solutions (given in Fig. 3.2) are better than before; the cpu time is 68.23. If c_2 is decreased from 5×10^{-4} to 10^{-4}, the results in Fig. 3.3 are the same as those in Fig. 3.1, and the cpu time is 75.70.

Finally, we use the values $c_1 = 0.025$, $c_2 = 0$, $\delta = 0$ tested in [FVZ90], and for $atol = rtol = 10^{-3}$, as we known before, the computing solution moves too fast. For $atol = rtol = 10^{-5}$, the computing solution moves slower than the reference solution. The behaviour of the results is similar to that in Fig. 3.1. For $atol = rtol = 10^{-4}$, at the time $t = 0.26$, the numerical solution is incorrect, but for later times ($t = 0.27, 0.28, 0.29$), the numerical solution becomes better gradually (see Fig 3.4).

Results for GWMFE

Problem sensitivity and trouble in determining the method parameters was also a difficulty with GWMFE. The parameters A and B are the coefficients of the '*arclength viscosities*' terms and '*arclength spring forces*' terms in (3.2.6), respectively.

(1) We have tried the values suggested by Miller [MI88]; unfortunately LSODI, fails to solve this problem. For $A = (10^{-6})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-4}$, GWMFE

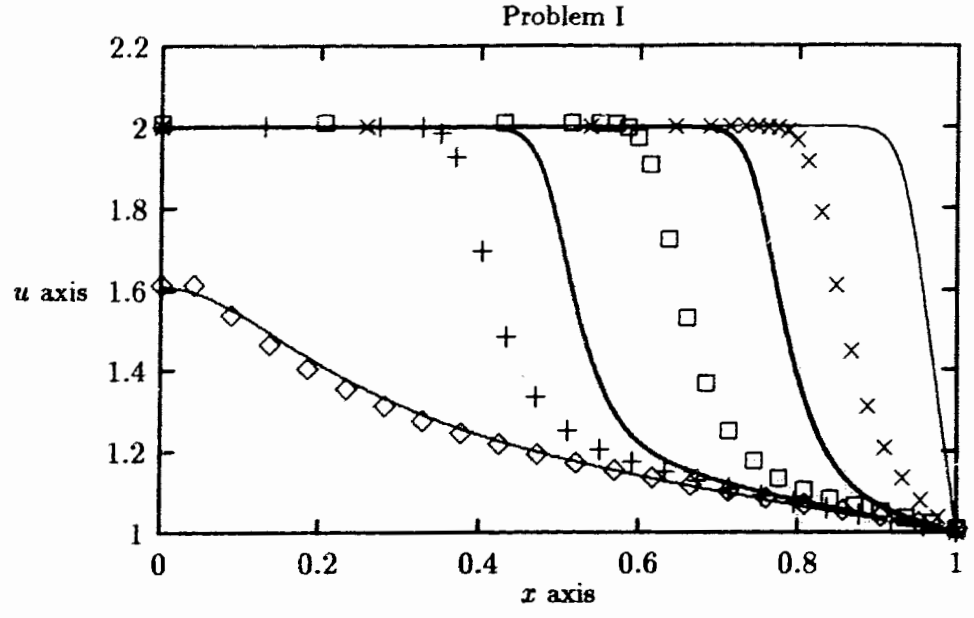

Figure 3.1: $c_1 = 0.1$, $c_2 = 5 \times 10^{-4}$, $\delta = 10^{-4}$, and $atol = rtol = 10^{-5}$, $cpu = 77.76$.

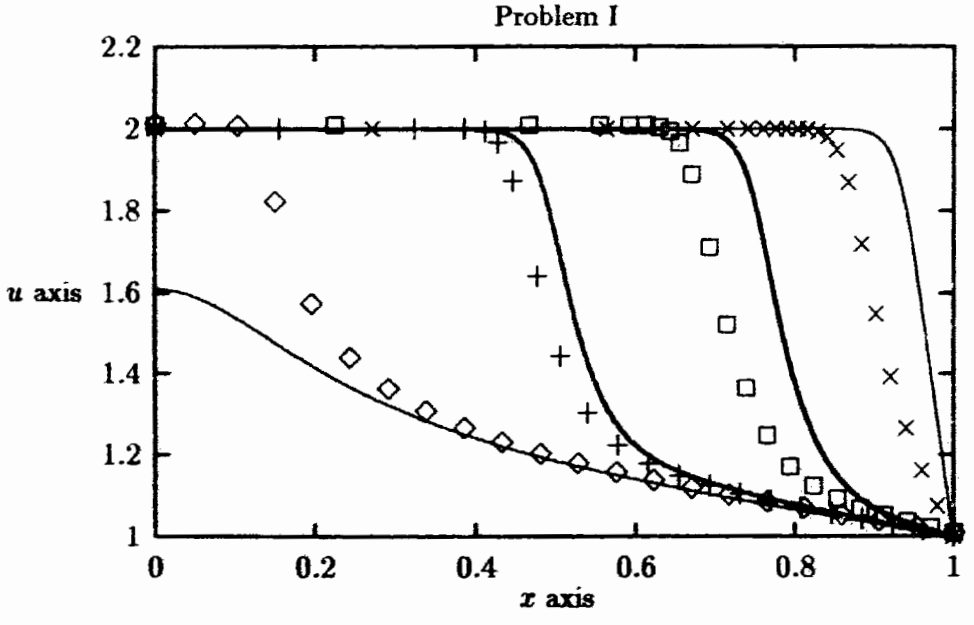

Figure 3.2: $c_1 = 0.1$, $c_2 = 5 \times 10^{-4}$, $\delta = 10^{-4}$, and $atol = rtol = 10^{-4}$, $cpu = 68.23$.

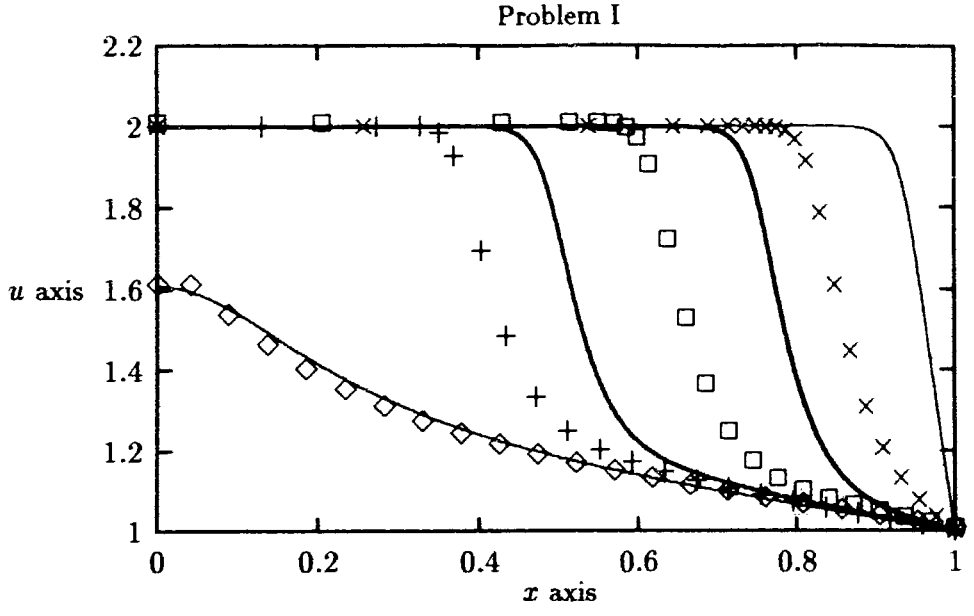

Figure 3.3: $c_1 = 0.1$, $c_2 = 10^{-4}$, $\delta = 10^{-4}$, and $atol = rtol = 10^{-5}$, $cpu = 75.70$.

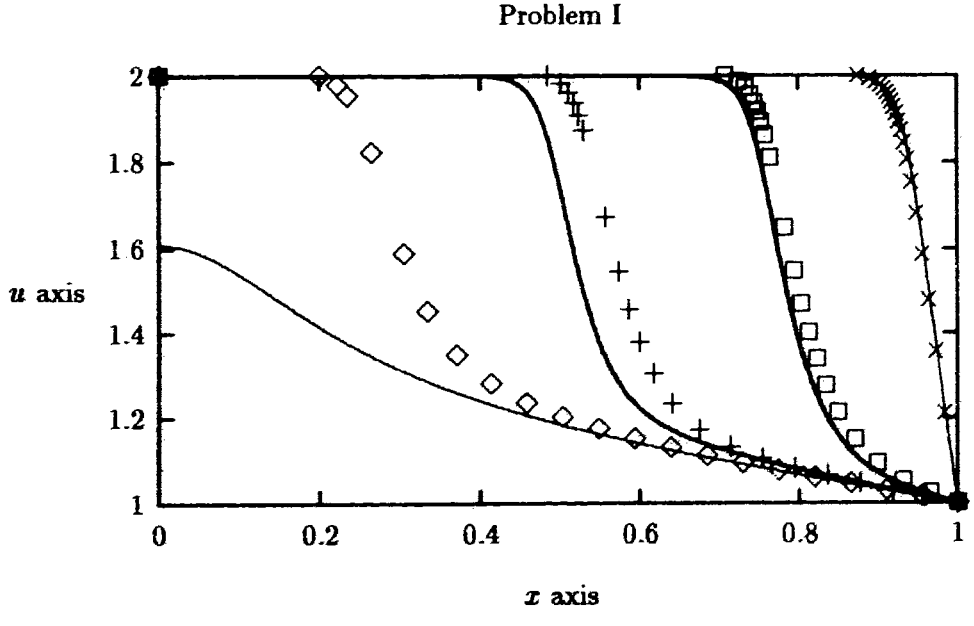

Figure 3.4: $c_1 = 0.025$, $c_2 = 0$, $\delta = 0$, and $atol = rtol = 10^{-4}$, $cpu = 131.58$.

fails because the time stepsize is zero at $t = 0.25$ (initial time for LSODI solver).

(2) For $A = 0.01$, $B = 0$ and $atol = rtol = 10^{-5}$, GWMFE fails at $t = 0.261$. The time stepsize is zero there, and three mesh points are crossing the left boundary ($x = 0$).

(3) For $A = 0.01$, $B = 0$ and $atol = rtol = 10^{-4}$, GWMFE spends much more time than MFE(1), *viz.*, the cpu time is 826.46. The accuracy of the solutions have not improved much using GWMFE, except that most of mesh points have moved to the layers. The results are given in Fig. 3.5.

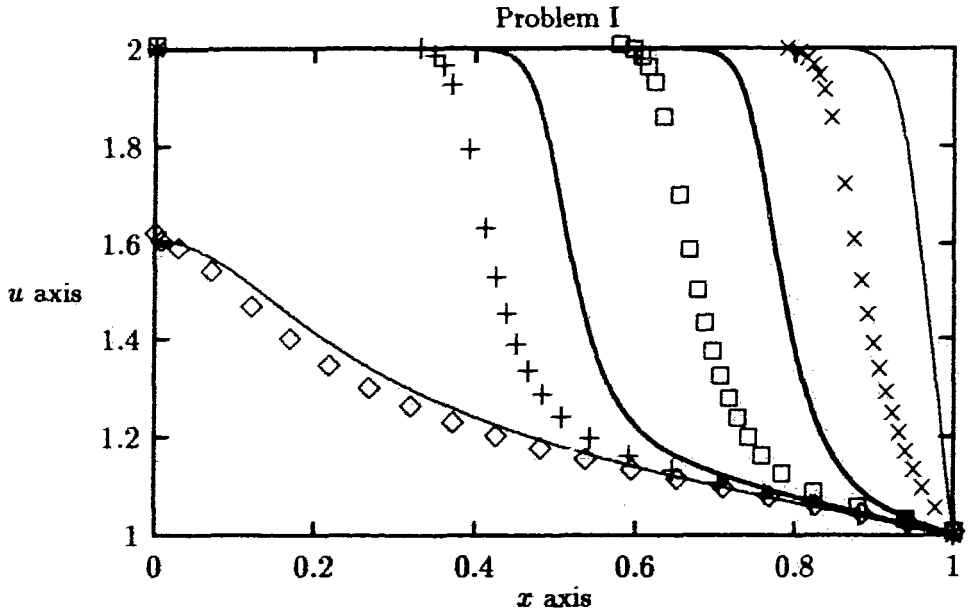

Figure 3.5: $A = 0.01$, $B = 0$ and $atol = rtol = 10^{-4}$, $cpu = 826.46$.

Results for MFE(2)

For $A = 0.01$, $B = 0$ and $atol = rtol = 10^{-4}$, the numerical solution near $x = 0$ for time $t = 0.26$ is incorrect. For later times ($t = 0.27, 0.28, 0.29$), the accuracy of the solutions are quite satisfactory and most of mesh points have moved into the layers; the cpu time is 147.35. The results are given in Fig. 3.6. If the tolerances (atol and rtol) are decreased to 10^{-5}, the accuracy of the solution becomes worse, and the cpu time is 182.92. The results are given in Fig. 3.7.

Results for Method III

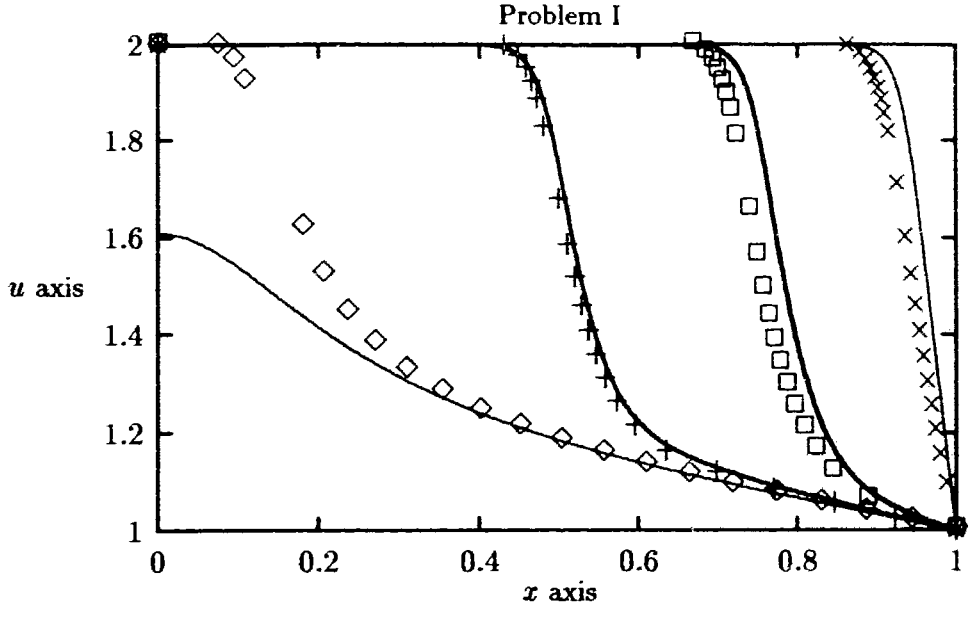

Figure 3.6: $A = 0.01$, $B = 0$ and $atol = rtol = 10^{-4}$, $cpu = 147.35$.

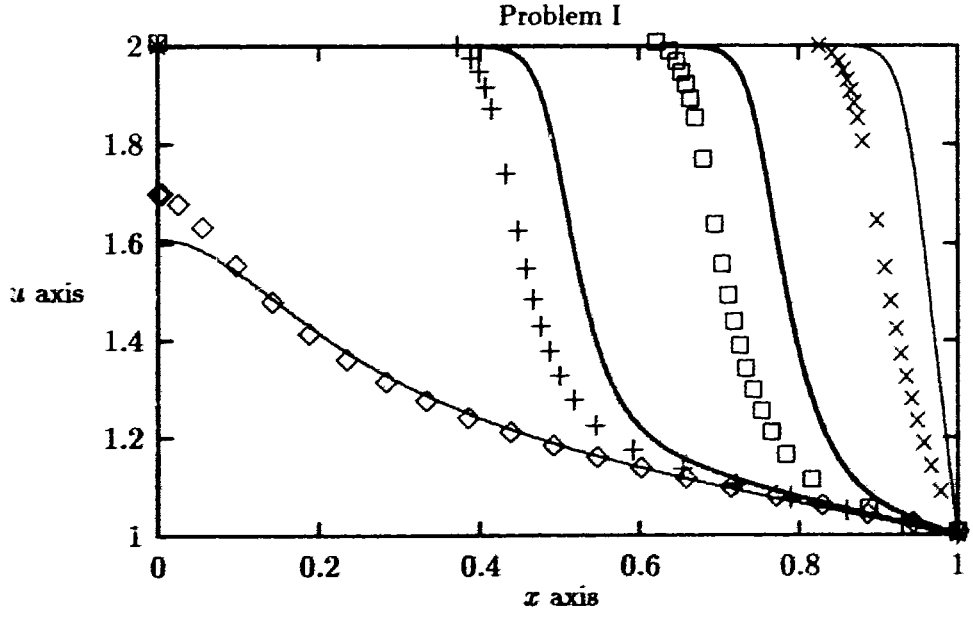

Figure 3.7: $A = 0.01$, $B = 0$ and $atol = rtol = 10^{-5}$, $cpu = 182.92$.

For $k = 2, \tau = 10^{-3}$ and $atol = rtol = 10^{-4}$, the numerical solution near $x = 0$ for time $t = 0.26$ is incorrect. The numerical solutions at $t = 0.27$ and 0.28 move slower than the reference solution, but at time $t = 0.29$, the numerical solution is quite accurate. The total cpu time is 23.73, which is much less than that for MFE(1) and MFE(2). The results are given in Fig. 3.8. Decreasing the tolerances (atol and rtol) to 10^{-5}, the results are much worse (see Fig. 3.9). For $k = 1$, the results have not improved, as shown in Fig. 3.10. For $k = 2$, $atol = rtol = 10^{-5}$ and $\tau = 10^{-4}$, the results shown in Fig. 3.11 are basically the same as those for $\tau = 10^{-3}$ given in Fig. 3.9, and the cpu time is 24.83. For $\tau = 10^{-6}$ and $\tau = 10^{-8}$, the results are unchanged.

The results for Problem I are summarized in Table 3.1 and Table 3.2. Note that the time stepsize for solving the MFE ODE system for Problem I gradually becomes larger when mesh points move into the steep regions of the solutions. For Method III, it is interesting that the resultant ODE system alternates between large and small.

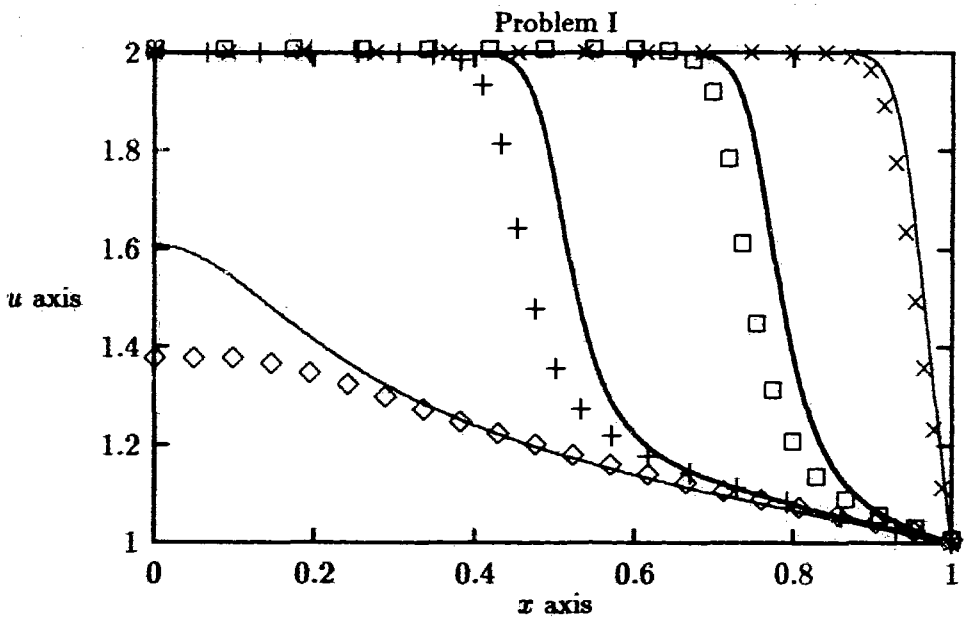

Figure 3.8: $k = 2, \tau = 10^{-3}$ and $atol = rtol = 10^{-4}$, $cpu = 23.73$.

Problem II (Burgers' equation)

We solve Burgers' equation with $\varepsilon = 10^{-4}$ and use $N = 20$ mesh points. For details of

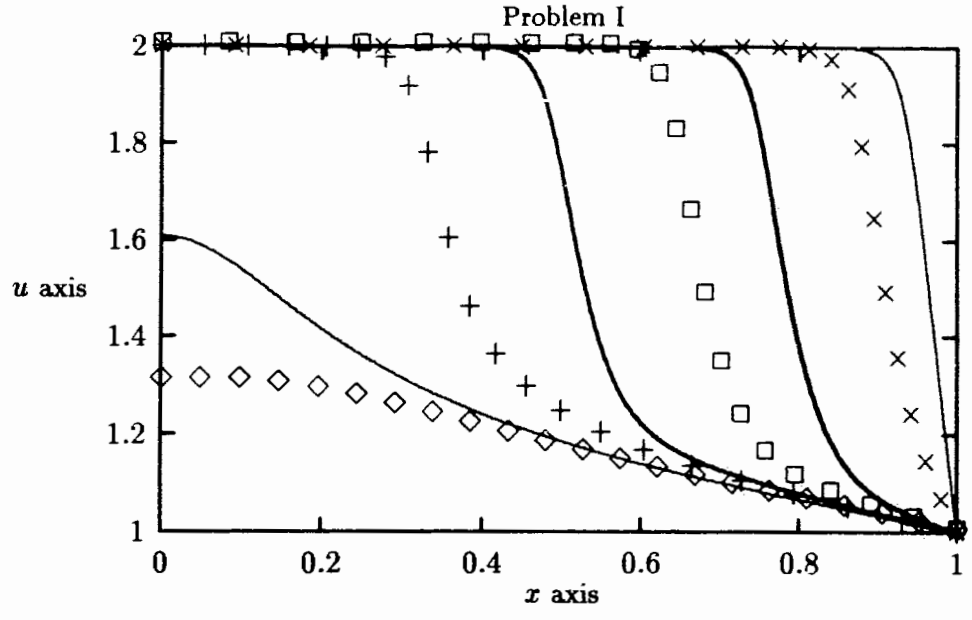

Figure 3.9: $k = 2, \tau = 10^{-3}$ and $atol = rtol = 10^{-5}$, $cpu = 25.25$.

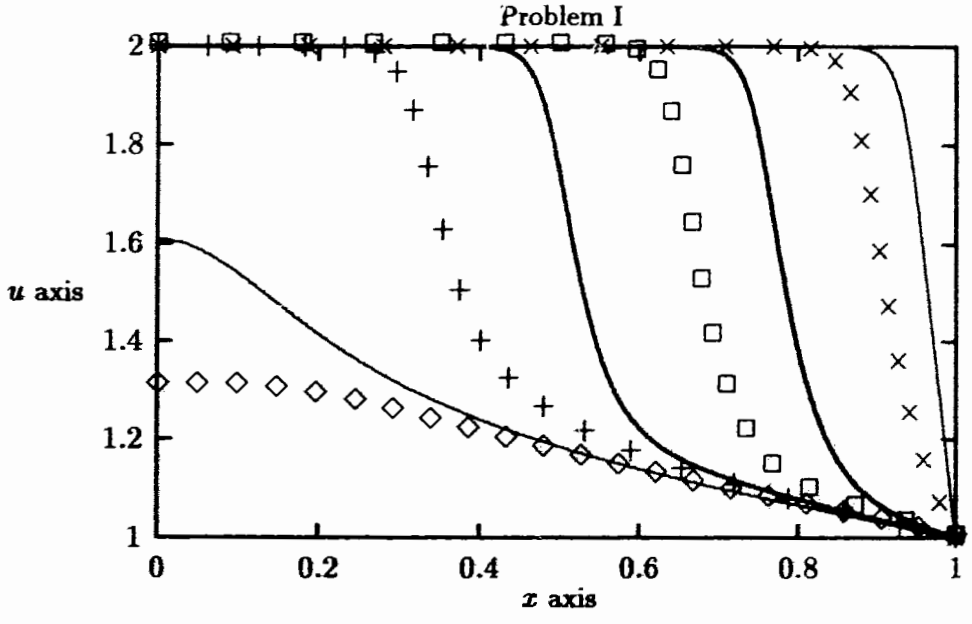

Figure 3.10: $k = 1, \tau = 10^{-3}$ and $atol = rtol = 10^{-5}$, $cpu = 26.44$.

t	Fig. 3.1(MFE(1)):77.76(cpu)				Fig. 3.2(MFE(1)):68.23(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.26	50	14	3	0.000236	45	22	3	0.000133
0.27	129	39	2	0.000189	76	41	3	0.000143
0.28	162	56	4	0.000381	98	57	2	0.000565
0.29	201	78	3	0.000323	115	71	2	0.000504
t	Fig. 3.3(MFE(1)):75.70				Fig. 3.5(GWMFE):826.46			
0.26	50	14	3	0.000236	174	121	2	0.000006
0.27	129	39	2	0.000189	759	532	2	0.000537
0.28	162	56	4	0.000381	777	545	2	0.001002
0.29	196	77	3	0.000349	791	556	3	0.001276
t	Fig. 3.6(MFE(2)):147.35				Fig. 3.7(MFE(2)):182.92			
0.26	68	51	2	0.000173	67	58	3	0.000146
0.27	111	75	3	0.000437	154	102	3	0.000226
0.28	132	91	2	0.000706	187	119	3	0.000401
0.29	148	105	3	0.000674	210	129	4	0.000545
t	Fig. 3.8(Method III):23.73				Fig. 3.9(Method III):25.25			
0.26	22	10	3	0.001152	39	10	3	0.001025
0.27	58	20	3	0.000645	91	19	3	0.000175
0.28	70	22	3	0.001114	115	23	3	0.000787
0.29	84	27	3	0.000572	131	26	3	0.000435
t	Fig. 3.10(Method III):26.44				Fig. 3.11(Method III):24.83			
0.26	39	10	3	0.001025	38	10	3	0.000931
0.27	92	18	4	0.000181	88	19	4	0.000206
0.28	118	23	3	0.000535	113	24	3	0.000683
0.29	133	27	3	0.000458	128	27	3	0.000516

Table 3.1: Problem I

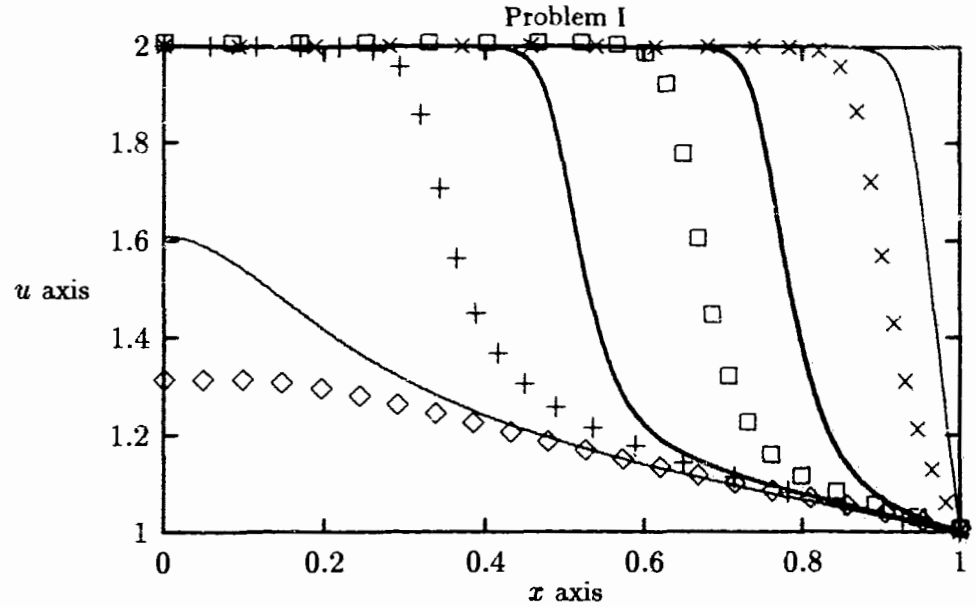

Figure 3.11: $k = 2, \tau = 10^{-4}$ and $atol = rtol = 10^{-5}$, $cpu = 24.83$.

the problem see page 48.

Results for MFE(1)

We use the tolerances $atol = rtol = 10^{-3}$ for MFE(1).

(1) First, we choose the parameter values $c_1 = 0.1$, $c_2 = 10^{-6}$(or 10^{-5}) and $\delta = 10^{-5}$. The reduced ODE system becomes stiff and MFE(1) stops at $t = 0.45$ because the time stepsize is zero. However there is no mesh point crossing.

(2) For $c_1 = 0.1$, $c_2 = 0$ and $\delta = 10^{-5}$, MFE(1) stops at $t = 1.0$ because the chosen

t	Fig. 3.4(MFE(1)):131.58(cpu)			
	nst	nje	nqn	tstep
0.26	84	58	2	0.000095
0.27	149	83	2	0.000361
0.28	186	105	2	0.000361
0.29	217	127	2	0.000383

Table 3.2: Problem I

stepsize is small (4×10^{-6}) for solving ODE system. Again there is no mesh point crossing.

(3) For $c_1 = 0.1$, $c_2 = 10^{-4}$ and $\delta = 10^{-4}$, MFE(1) fails at $t = 0.25$. The time stepsize is zero at this time, and some mesh points are crossing in the interval $[0.596838, 0.628945]$.

(4) For $c_1 = 0.025$, $c_2 = 10^{-4}$ and $\delta = 10^{-4}$, MFE(1) fails at $t = 0.2$. The time stepsize is 3×10^{-6}. At this time, some mesh points are crossing in $[0.590899, 0.590971]$ which is where oscillating solutions occur.

(5) For $c_1 = 0.2$, $c_2 = 10^{-5}$ and $\delta = 10^{-5}$, MFE(1) fails at $t = 0.2$. Although the time stepsize is not too small (0.012256), some mesh points are crossing in $[0.570000, 0.657410]$.

(6) For $c_1 = 0.01$, $c_2 = 10^{-4}$ and $\delta = 5 \times 10^{-5}$(or $\delta = 10^{-4}$), MFE(1) fails at $t = 0.2$. The time stepsize is 5×10^{-6} (or 7×10^{-6}), but there is no mesh crossing.

(7) For $c_1 = (4 \times 10^{-6})^{1/2}$, $c_2 = (10^{-9})^{1/2}$ and $\delta = 10^{-4}$, MFE(1) fails at $t = 0.2$. The time stepsize is zero, three mesh points are the same (0.586943), and the approximate solutions are much worse than the other cases above due to unacceptable oscillations.

(8) For $c_1 = 0.025$, $c_2 = 0$ and $\delta = 10^{-5}$, MFE(1) fails at $t = 0.2$. The time stepsize is zero, and some mesh points are crossing having successive values {0.590385, 0.590628, 0.590623, 0.590704}.

(9) For $c_1 = 0.025$, $c_2 = 10^{-3}$ and $\delta = 10^{-4}$, MFE(1) fails at $t = 1.4$. The time stepsize is 6.3×10^{-5}, where one mesh point (1.001088) is crossing the right boundary ($x = 1$), and the solution at 1.001088 jumps to 0.066214.

(10) For $c_1 = 0.1$, $c_2 = 0$ and $\delta = 10^{-4}$, MFE(1) fails at $t = 0.2$. The time stepsize is zero, but there is no mesh point crossing.

(11) For $c_1 = 10^{-3}$, $c_2 = 10^{-4}$ and $\delta = 10^{-4}$, MFE(1) fails at $t = 0.2$. The time stepsize is zero, and some mesh points are crossing having successive values {0.587316, 0.636860, 0.587304, 0.587398}. The solution at 0.636860 jumps to -147.259502.

(12) For $c_1 = 0.15$, $c_2 = 10^{-2}$ and $\delta = 10^{-4}$, MFE(1) fails at $t = 1.4$. The time stepsize is zero, and some mesh points are crossing, having successive values {0.991774, 0.993613, 0.991672, 0.993822}.

(13) For $c_1 = 0.05$, $c_2 = 10^{-3}$ and $\delta = 10^{-4}$, MFE(1) fails at $t = 0.2$. The time stepsize is zero, and the mesh points are crossing, having successive values {0.583234, 0.583353, 0.583454, 0.583259}. The solution at point 0.583454 jumps to -17.231277.

(14) For $c_1 = 0.01$, $c_2 = 10^{-2}$, $\delta = 10^{-4}$ and tolerances $atol = rtol = 10^{-3}$, oscillations in the solutions occur after $t = 0.2$, few mesh points are in the layers, and the cpu time is 146.79. The results are given in Fig. 3.12.

The choice $c_1 \geq 0.025$ was used by Furzeland *et al.*[FVZ90], and smaller c_2 was recommended by Gelinas *et al.*[GDM81], and Hrymak *et al.*[HMW86]. To increase c_1 and to decrease c_2, we test $c_1 = 0.1, c_2 = 10^{-3}, \delta = 10^{-4}$ and $atol = rtol = 10^{-3}$. The results are quite satisfactory except for small oscillations, as shown in Fig. 3.13. The cpu time is 159.98. Further, we test a smaller c_2, *viz.*, $c_2 = 5 \times 10^{-4}$. The results shown in Fig. 3.14 are now better, but the method is much more expensive, with a cpu time of 255.94.

To decrease c_1, we use $c_1 = 0.05, c_2 = 5 \times 10^{-4}, \delta = 10^{-4}$ and $atol = rtol = 10^{-3}$. The results shown in Fig. 3.15 are basically the same as those given in Fig. 3.14.

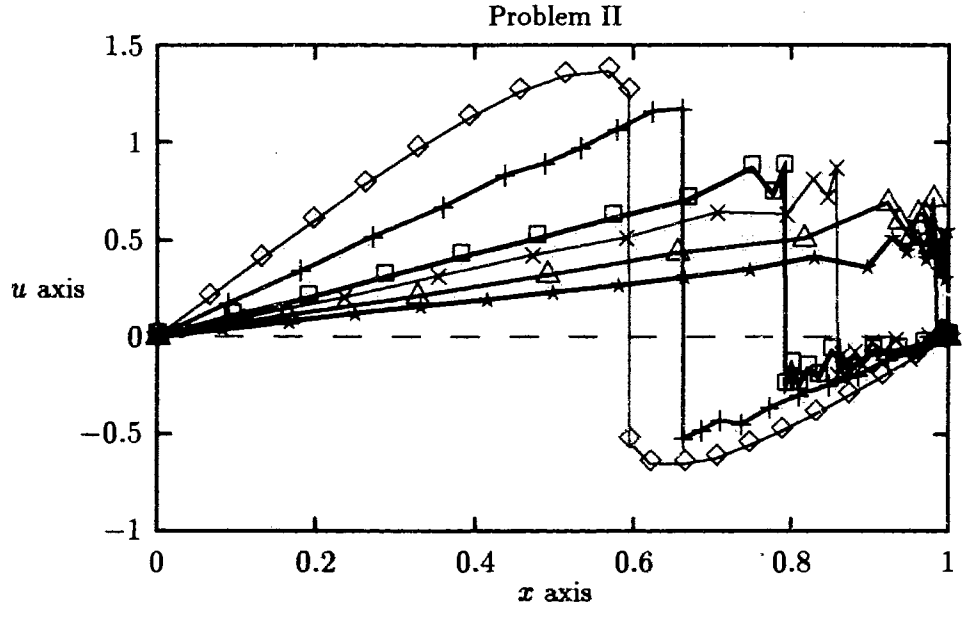

Figure 3.12: $c_1 = 0.01$, $c_2 = 10^{-2}$ and $\delta = 10^{-4}$, $atol = 10^{-3}$, $rtol = 10^{-3}$, cpu =146.79.

Results for GWMFE

t	Fig. 3.12(MFE(1)):146.79(cpu)				Fig. 3.13(MFE(1)):159.98(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.2	45	36	2	0.006515	68	47	1	0.001115
0.4	78	55	2	0.006846	199	148	1	0.003564
0.8	99	65	3	0.025080	228	165	1	0.017565
1.0	109	68	3	0.016578	233	166	1	0.045048
1.4	183	112	2	0.000726	244	175	2	0.005145
2.0	357	185	2	0.015561	316	213	2	0.139613
t	Fig. 3.14(MFE(1)):255.94				Fig. 3.15(MFE(1)):308.49			
0.2	68	52	1	0.000503	107	82	1	0.000479
0.4	390	249	1	0.001915	472	304	1	0.003273
0.8	455	294	1	0.056705	537	346	1	0.030485
1.0	458	296	1	0.056705	541	348	2	0.061189
1.4	468	305	1	0.000648	555	357	1	0.005229
2.0	525	338	2	0.095474	625	406	2	0.122648
t	Fig. 3.16(GWMFE):524.68				Fig. 3.17(GWMFE):570.59			
0.2	78	45	2	0.001179	119	57	2	0.000693
0.4	155	86	3	0.011499	235	111	4	0.006692
0.8	174	104	3	0.036544	260	136	4	0.026438
1.0	178	108	3	0.052949	266	142	4	0.042221
1.4	310	188	1	0.004809	445	208	1	0.003406
2.0	380	233	3	0.070649	565	261	3	0.050214

Table 3.3: Problem II

t	Fig. 3.18(GWMFE):553.30(cpu)				Fig. 3.19(MFE(2)):919.64(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.2	116	62	3	0.000568	83	65	2	0.001446
0.4	208	114	2	0.012319	425	315	1	0.000417
0.8	230	137	3	0.043453	821	628	1	0.002072
1.0	234	141	3	0.068609	900	693	1	0.004044
1.4	346	210	1	0.009262	967	746	2	0.014998
2.0	416	256	3	0.069342	1067	815	2	0.017230
t	Fig. 3.20(MFE(2)):945.90				Fig. 3.21(MFE(2)):868.76			
0.2	89	70	3	0.001382	95	72	2	0.001262
0.4	430	319	1	0.000252	391	278	1	0.001202
0.8	847	654	1	0.002083	820	620	1	0.001597
1.0	925	719	1	0.004059	890	677	1	0.003400
1.4	993	772	1	0.013676	938	709	2	0.004874
2.0	1092	839	1	0.024404	1024	768	1	0.022465

Table 3.4: Problem II

t	Fig. 3.22(MFE(2)):836.26(cpu)			
	nst	nje	nqn	tstep
0.2	83	57	2	0.001402
0.4	369	251	1	0.001120
0.8	713	515	1	0.001386
1.0	799	583	1	0.002010
1.4	902	662	1	0.002380
2.0	1032	735	1	0.019370

Table 3.5: Problem II

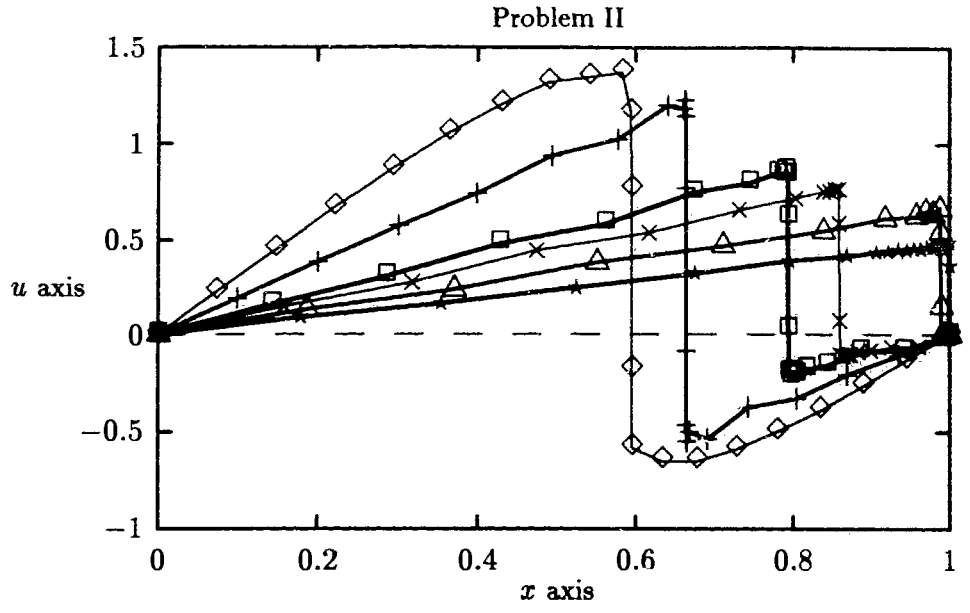

Figure 3.13: $c_1 = 0.1$, $c_2 = 10^{-3}$ and $\delta = 10^{-4}$, $atol = 10^{-3}$, $rtol = 10^{-3}$, cpu =159.98.

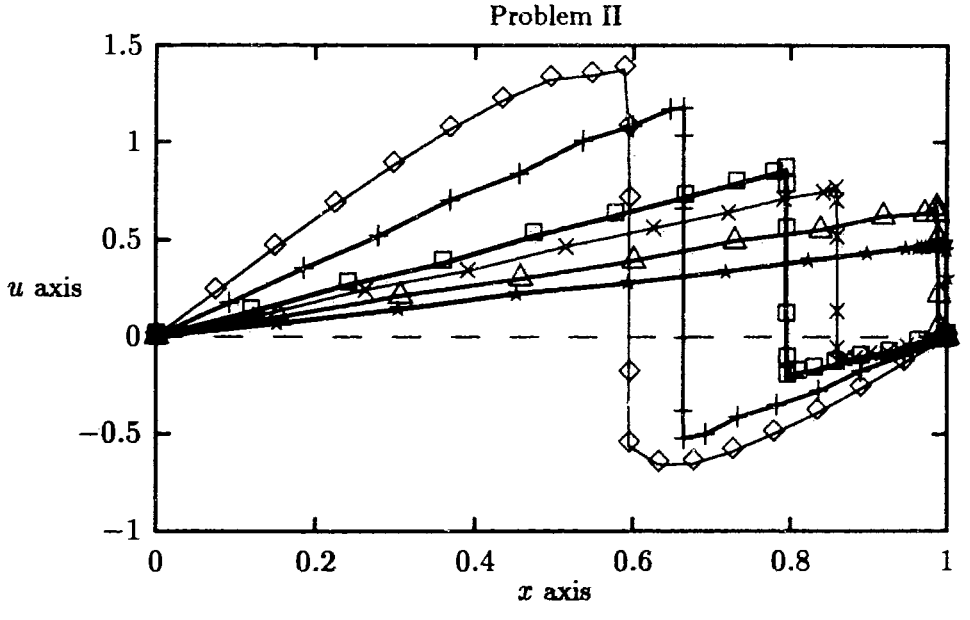

Figure 3.14: $c_1 = 0.1$, $c_2 = 5 \times 10^{-4}$ and $\delta = 10^{-4}$, $atol = 10^{-3}$, $rtol = 10^{-3}$, cpu =255.94.

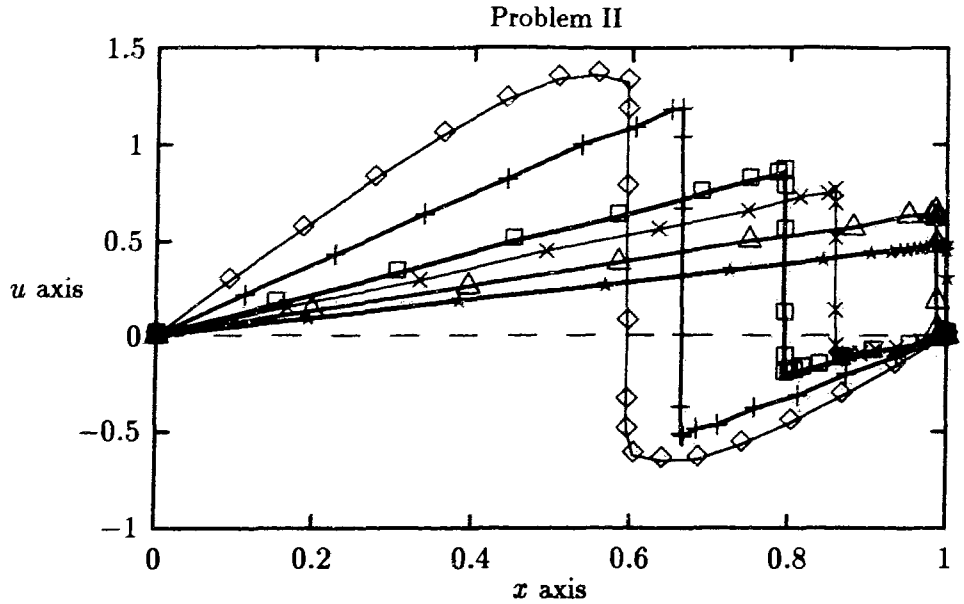

Figure 3.15: $c_1 = 0.05$, $c_2 = 5 \times 10^{-4}$ and $\delta = 10^{-4}$, $atol = 10^{-3}$, $rtol = 10^{-3}$, cpu =308.49.

We first choose the parameter A values as recommended by Miller [MI88], and the parameter B value that is ten times the suggested one (also from [MI88]).

(1) For $A = (10^{-6})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-3}$, GWMFE fails at $t = 1.25$. The time stepsize is 0.006148, and one mesh point is crossing with $\{0.934948, 0.943122, 0.939446\}$.

(2) For $A = (10^{-7})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-3}$, GWMFE fails at $t = 1.55$. The time stepsize is zero, and many mesh point crossings occur in the interval $[0, 1]$.

(3) For $A = (10^{-6})^{1/2}$, $B = (10^{-8})^{1/2}$ and $atol = rtol = 10^{-4}$, GWMFE fails at $t = 0.25$, the time stepsize is 0.003370, and the mesh points are crossing.

(4) For $A = (10^{-8})^{1/2}$, $B = (10^{-8})^{1/2}$ and $atol = rtol = 10^{-4}$, GWMFE fails at $t = 1.45$, the time stepsize is zero, and four mesh points are crossing the right bourdary ($x = 1$).

(5) For $A = (10^{-6})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-4}$, the results are quite good, as shown in Fig. 3.16. More mesh points move to the steep regions using GWMFE than when using MFE(1), but it runs much longer than MFE(1), with a cpu time of 524.68. Decreasing A to $(10^{-7})^{1/2}$, the results are unchanged, except the cpu time is 553.30. The results are

given in Fig. 3.18. Decreasing the tolerances to $atol = rtol = 10^{-5}$, and again using $A = B = (10^{-6})^{1/2}$, the results, shown in Fig. 3.17, are the same as for $atol = rtol = 10^{-4}$ in Fig. 3.16, except the cpu time is 570.59.

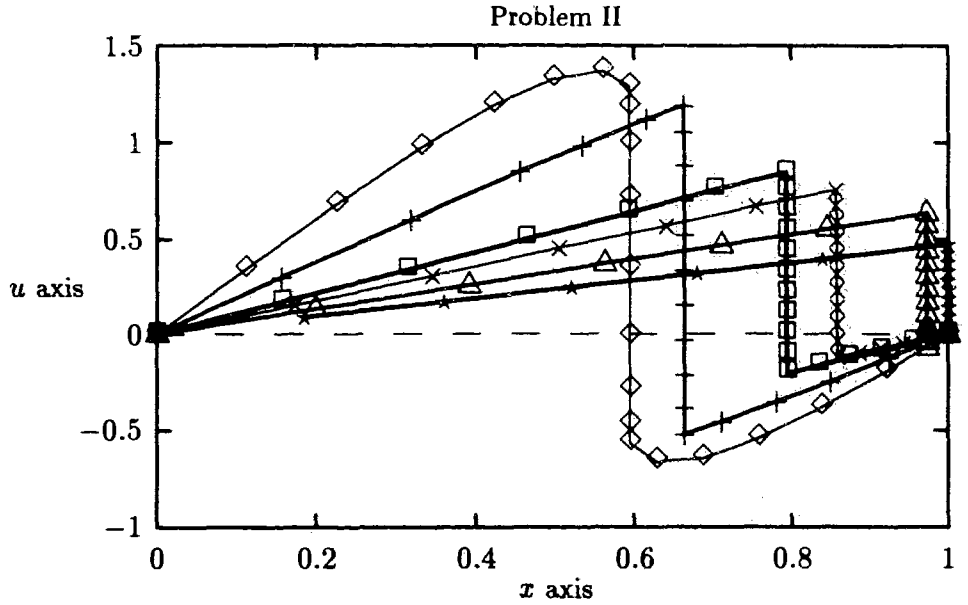

Figure 3.16: $A = (10^{-6})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-4}$, cpu= 524.68.

Results for MFE(2)

Similar values are tested as for GWMFE.

(1) For $A = (4 \times 10^{-6})^{1/2}$, $B = (10^{-9})^{1/2}$ and $atol = rtol = 10^{-3}$, MFE(2) fails at $t = 0.25$. The time stepsize is 2×10^{-6}, however there is no mesh point crossing.

(2) For $A = (10^{-6})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-3}$, MFE(2) fails at $t = 0.25$. The time stepsize is 3.6×10^{-5}, but there is no mesh point crossing.

(3) For $A = (10^{-6})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-4}$, MFE(2) fails at $t = 1.5$. The time stepsize is zero, and 9 mesh points are crossing the right boundary ($x = 1$).

(4) For $A = (10^{-4})^{1/2}, B = 0$ and $atol = rtol = 10^{-4}$, the results are satisfactory. When the solution becomes steeper, almost all mesh points move to the steep regions after $t = 0.4$ as shown in Fig. 3.19, but the cpu time is very high, 919.64. Increasing B to $(10^{-8})^{1/2}$ and

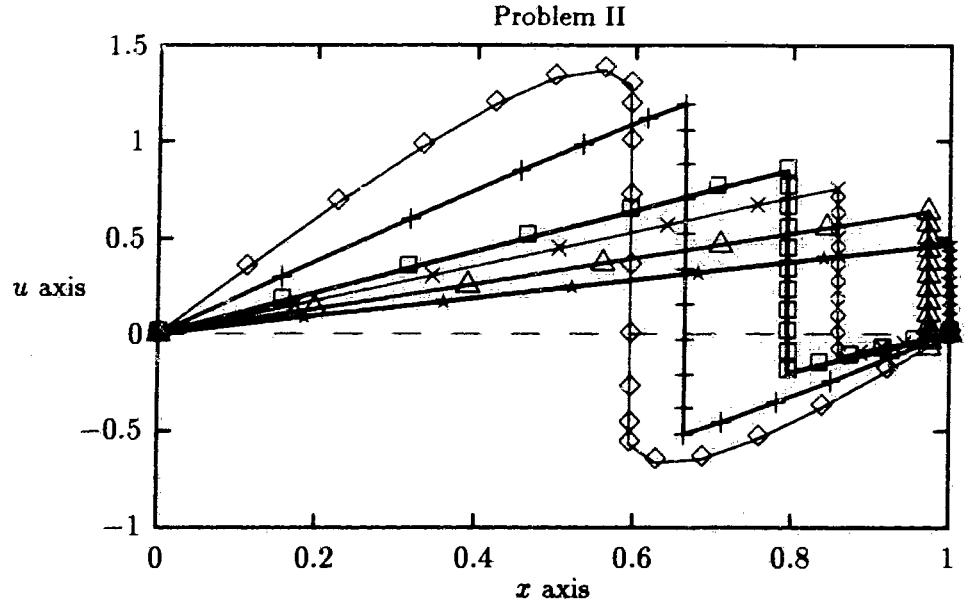

Figure 3.17: $A = (10^{-6})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-5}$, $cpu = 570.59$.

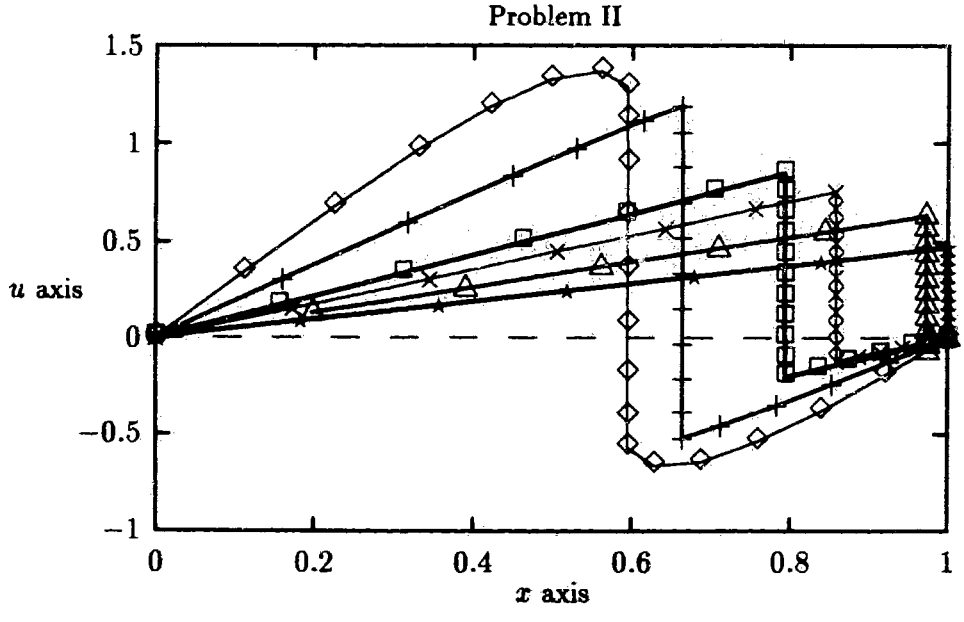

Figure 3.18: $A = (10^{-7})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-4}$, $cpu = 553.30$.

$(10^{-6})^{1/2}$, the results are given in Fig. 3.20 and in Fig. 3.21, respectively. The qualities of the solutions are unchanged. Further increasing B to $B = (10^{-5})^{1/2}$, the results are quite satisfactory. For late times a few more mesh points move away from the steep regions, as shown in Fig. 3.22, and the cpu time is 836.26.

It is important to note how points locate in the layer. Miller's GWMFE forces points into layer, and while the regularization term eventually does it more, it does less initially.

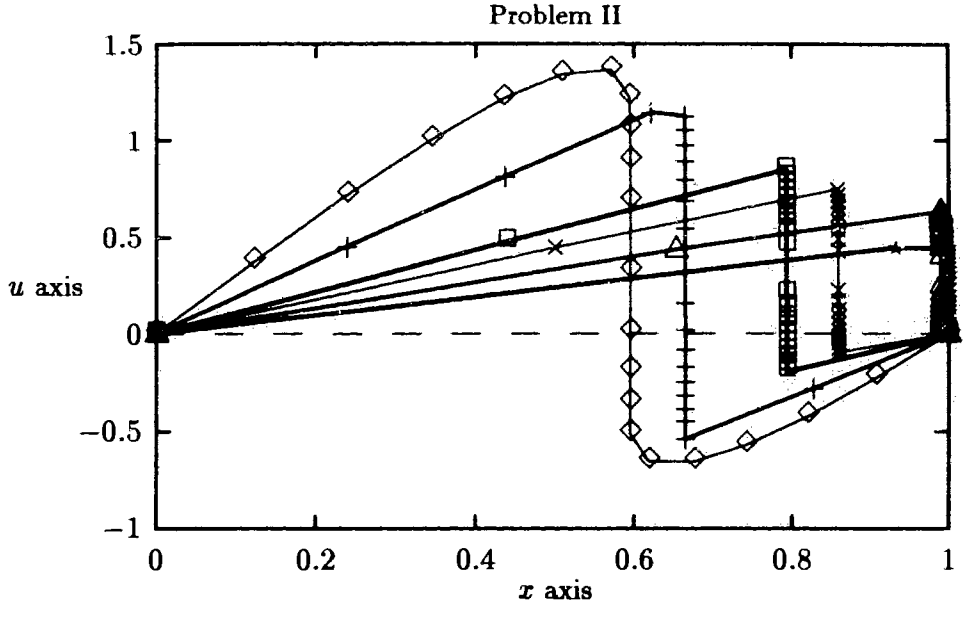

Figure 3.19: $A = (10^{-4})^{1/2}$, $B = 0$ and $atol = rtol = 10^{-4}$, $cpu = 919.64$.

Results for **Method III**

When using $N = 20$ mesh points, Method III fails to solve this problem, and as before failure of LSODI is because the numerical solutions are unacceptable due to oscillations, which may result from too few mesh points being in the steep regions. We use mesh points $N = 40$ to solve this problem.

(1) For $k = 2, \tau = 10^{-4}$ and $atol = rtol = 10^{-4}$, Method III fails at $t = 1.5$, the time stepsize is 8.2×10^{-5}, and there is no mesh points crossing.

(2) For $k = 2, \tau = 10^{-5}$ and $atol = rtol = 10^{-4}$, Method III fails at $t = 0.65$; although

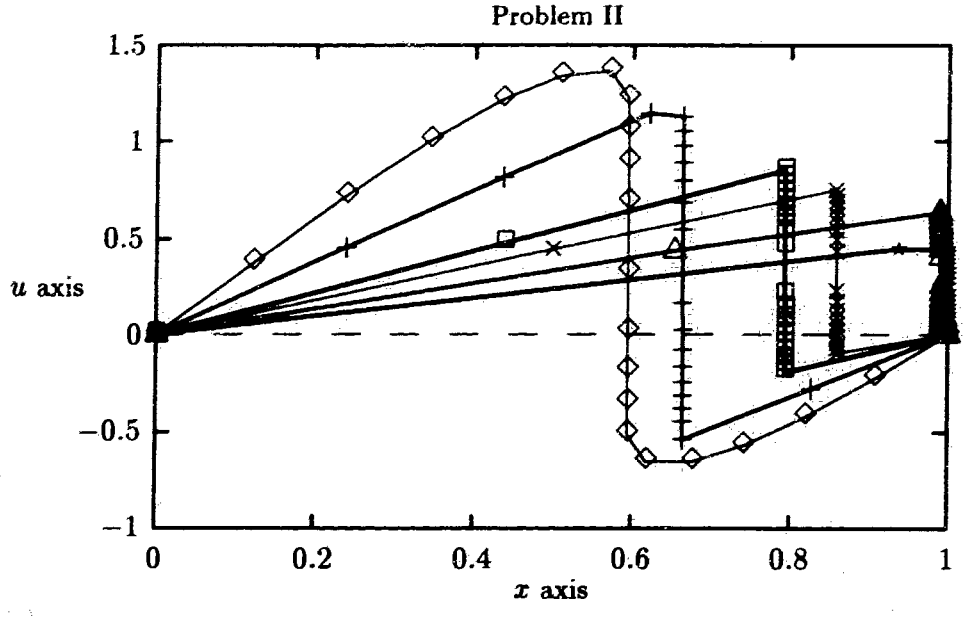

Figure 3.20: $A = (10^{-4})^{1/2}$, $B = (10^{-8})^{1/2}$ and $atol = rtol = 10^{-4}$, $cpu = 945.90$.

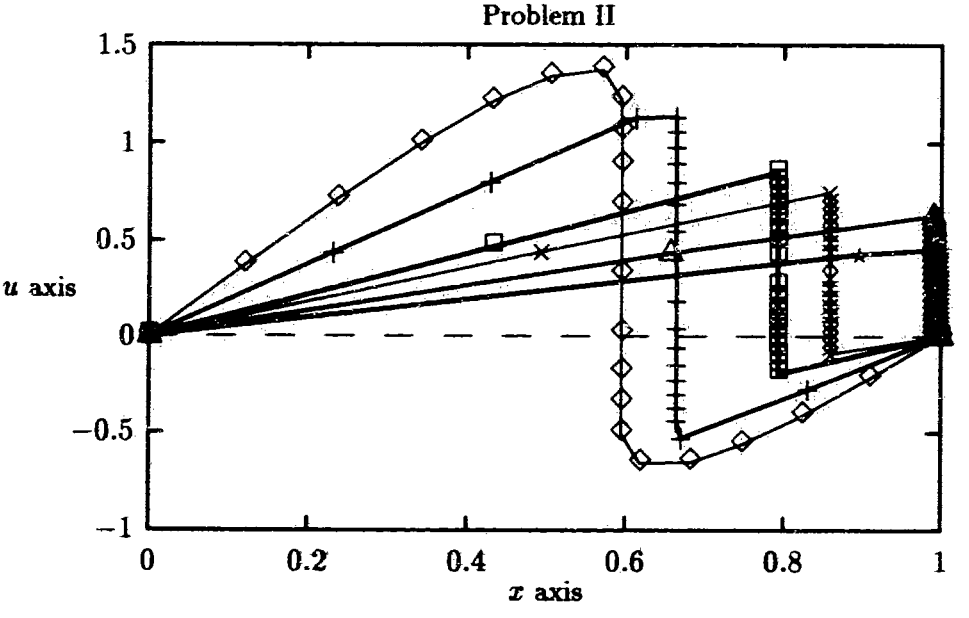

Figure 3.21: $A = (10^{-4})^{1/2}$, $B = (10^{-6})^{1/2}$ and $atol = rtol = 10^{-4}$, $cpu = 868.76$.

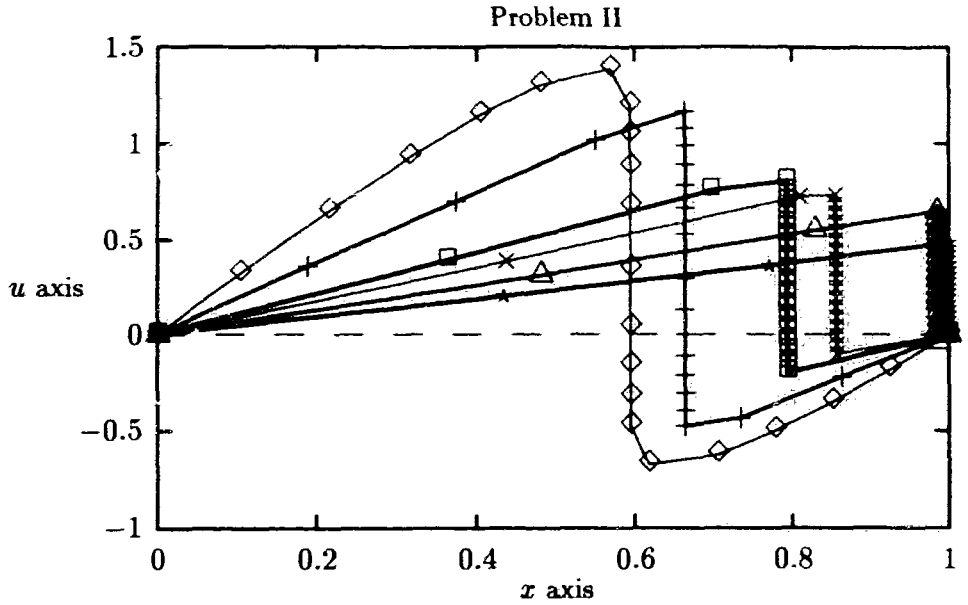

Figure 3.22: $A = (10^{-4})^{1/2}$, $B = (10^{-5})^{1/2}$ and $atol = rtol = 10^{-4}$, $cpu = 836.26$.

the time stepsize is not too small (1.7×10^{-2}), the next time stepsize is zero, and there is no mesh points crossing.

(3) For $k = 1, \tau = 10^{-2}$ and $atol = rtol = 10^{-3}$, Method III fails at $t = 0.25$, the time stepsize is 1.9×10^{-5}, and there is no mesh crossing.

(4) For $k = 2, \tau = 10^{-3}$ and $atol = rtol = 10^{-4}$, Method III succeeds in solving this problem (see in Fig. 3.23), but it spends a cpu time of 3153.53. Increasing τ to 10^{-2}, the qualities of the solutions do not improve, although it takes much less cpu time, *viz.*, 2175.36. The results are given in Fig. 3.24.

We change k from 2 to 1, and test $\tau = 10^{-3}$, $atol = rtol = 10^{-4}$. The results given in Fig. 3.26 are better than the corresponding results for $k = 2$ in Fig. 3.23, but the cpu time is still higher, 3060.02. Increasing τ to 10^{-2}, the qualities of the results are unchanged, but the cpu time reduces to 2088.24. These results are given in Fig. 3.25.

Problem III (Buckley-Leverett equation)

For this problem (see page 55), an artificial viscosity term εu_{xx} is needed. We choose

t	Fig. 3.23(Method III):3153.53(cpu)				Fig. 3.24(Method III):2175.36(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.2	133	48	2	0.001754	179	73	2	0.000383
0.4	213	87	1	0.004084	283	123	1	0.002914
0.8	260	110	1	0.012193	341	155	1	0.012938
1.0	280	120	2	0.009915	371	171	2	0.018942
1.4	391	184	2	0.001034	502	246	2	0.000993
2.0	1067	724	1	0.001520	639	307	1	0.007222
t	Fig. 3.25(Method III):2088.24				Fig. 3.26(Method III):3060.02			
0.2	163	90	1	0.000441	98	32	2	0.002425
0.4	258	140	1	0.003598	211	107	1	0.001019
0.8	343	191	2	0.008476	339	193	1	0.006910
1.0	374	211	1	0.009210	362	205	2	0.012618
1.4	442	249	2	0.002975	440	242	2	0.003939
2.0	569	294	2	0.054334	715	431	1	0.023336

Table 3.6: Problem II

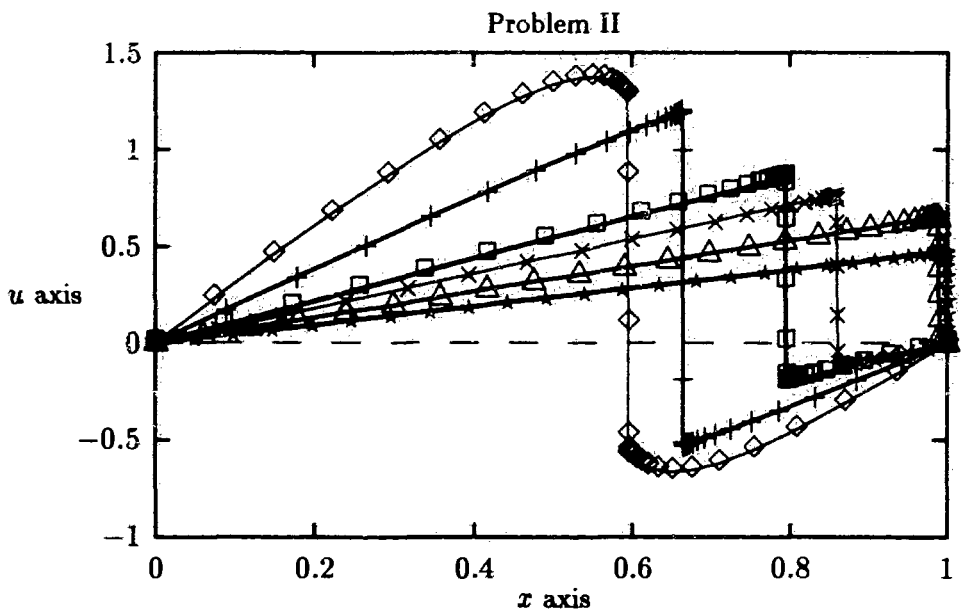

Figure 3.23: $k = 2, \tau = 10^{-3}$ and $atol = rtol = 10^{-4}$, $cpu = 3153.53$.

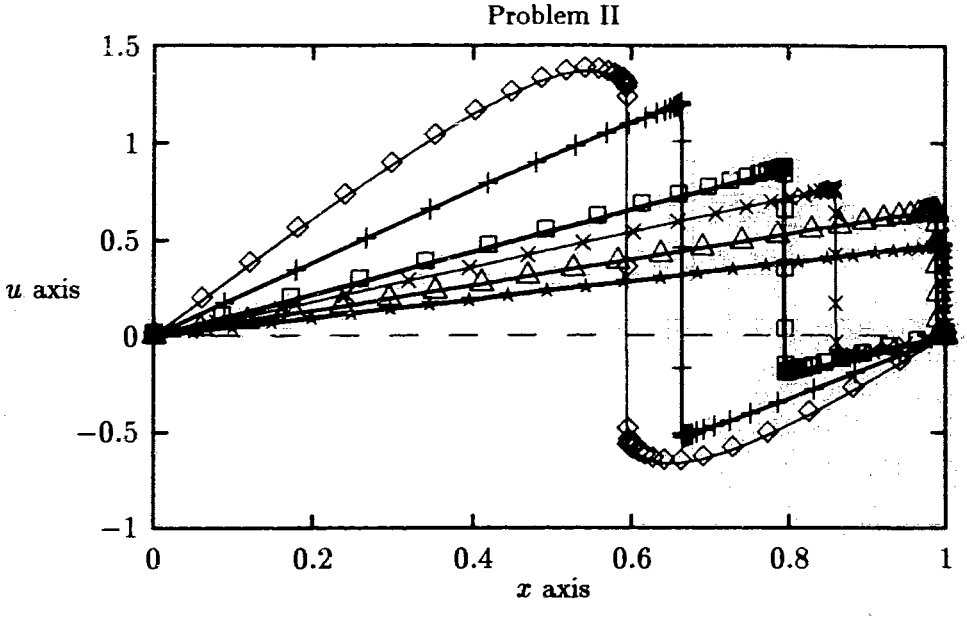

Figure 3.24: $k = 2, \tau = 10^{-2}$ and $atol = rtol = 10^{-4}$, $cpu = 2175.36$.

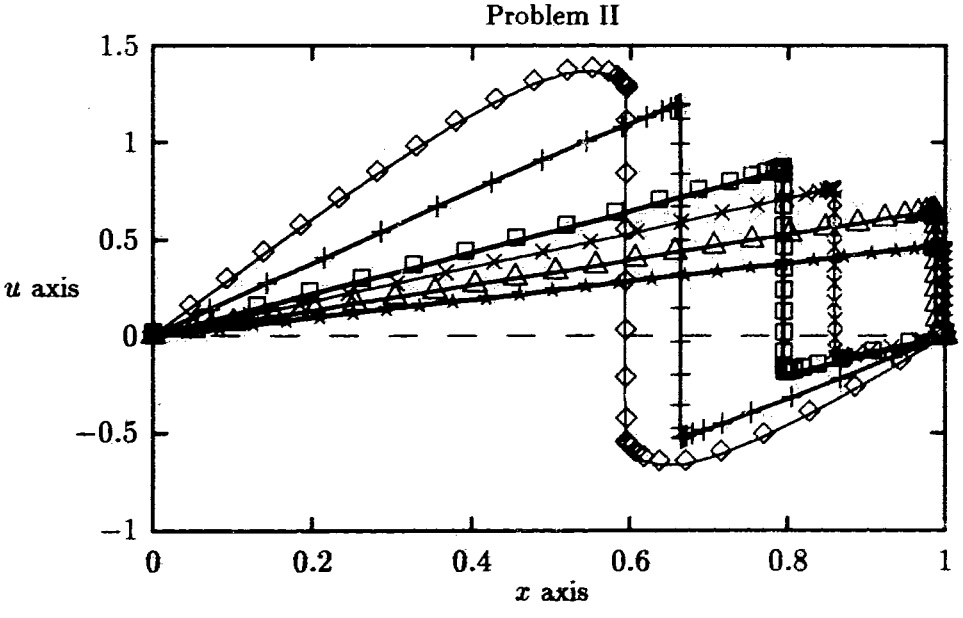

Figure 3.25: $k = 1, \tau = 10^{-2}$ and $atol = rtol = 10^{-4}$, $cpu = 2088.24$.

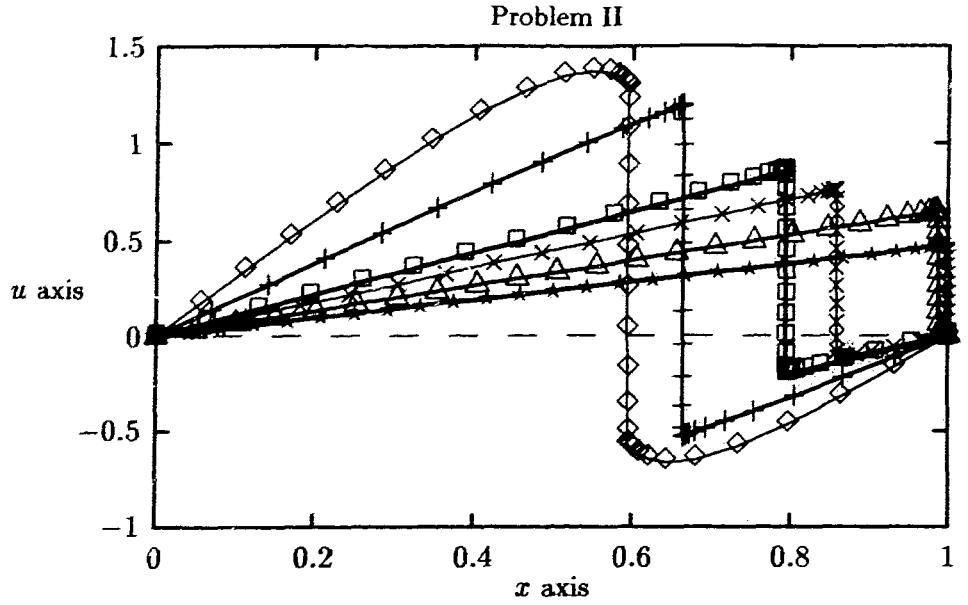

Figure 3.26: $k = 1, \tau = 10^{-3}$ and $atol = rtol = 10^{-4}$, $cpu = 3060.02$.

the parameter values as for Problem I and Problem II.

Results for MFE(1)

(1) For $c_1 = 0.1, c_2 = 10^{-4}, \delta = 10^{-4}$ and $atol = rtol = 10^{-5}$, the results for $t = 0.1, 0.2$ are correct. For later times, $t = 0.3$ and 0.4, a couple of solution values are incorrect as shown in Fig. 3.27, and few mesh points are in the steep regions. The cpu time is 102.04. Decreasing c_1 to 0.01, MFE(1) succeeds in solving the problem, but mesh points concentrate at the right boundary. (see Fig. 3.28). The cpu time increases to 260.02 for this case. Further decreasing c_2 to 0, the quality of the results does not improvement, as shown in Fig. 3.29, with cpu time now 536.59.

(2) For $c_1 = 0.025$, $c_2 = 10^{-4}$, $\delta = 10^{-4}$ and $atol = rtol = 10^{-5}$, the results are shown in Fig. 3.30. The numerical solutions for $t = 0.1, 0.2, 0.3$ are correct. For $t = 0.4$, only one solution point is incorrect, although few mesh points are near the layers, with points again moving to the right boundary ($x = 1$).

(3) For $c_1 = 0.01$, $c_2 = 10^{-4}$, $\delta = 5 \times 10^{-4}$ and $atol = rtol = 10^{-5}$, the results (given in Fig. 3.31) are similar to these given in Fig. 3.28.

t	Fig. 3.27(MFE(1)):102.04(cpu)				Fig. 3.28(MFE(1)):260.02(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.1	24	7	4	0.005902	24	12	4	0.004980
0.2	41	12	4	0.005902	45	25	3	0.001060
0.3	63	19	3	0.005738	83	40	3	0.007683
0.4	112	34	3	0.000477	127	58	4	0.000631
t	Fig. 3.29(MFE(1)):536.59				Fig. 3.30(MFE(1)):177.09			
0.1	24	12	4	0.004982	26	13	3	0.005360
0.2	54	26	2	0.000878	45	22	1	0.001199
0.3	92	40	3	0.001829	82	39	4	0.003704
0.4	191	87	2	0.001308	132	53	3	0.002080

Table 3.7: Problem III

The results are similar to those in [GDM81]. We conclude that success of the method depends heavily on parameter values and this dependence is very sensitive.

Results for GWMFE

For this problem, parameter values giving accurate results for GWMFE were much more difficult to find. The best values found were $A = (10^{-6})^{1/2}$, $B = 0$ and $atol = rtol = 10^{-5}$. Even then, while GWMFE solves the problem, the results are not satisfactory, as shown in Fig. 3.32. Most mesh points move to the right boundary, and it spends considerable time to solve the problem, with cpu time being 1207.86.

Results for MFE(2)

We first test the method with $A = (4 \times 10^{-6})^{1/2}$, $B = (10^{-9})^{1/2}$ as recommended by Miller and by Zegeling & Blom, and $atol = rtol = 10^{-5}$. As shown in Fig. 3.33, the accuracy of the numerical solution is poor, and the cpu time is 581.47.

For $A = (10^{-4})^{1/2}$, $B = 0$ and $atol = rtol = 10^{-5}$, MFE(2) is more successful at solving this problem. Mesh points now move into the steep region of the solution, and the cpu time is reduced to 455.61. The results are displayed in Fig. 3.34.

t	Fig. 3.31(MFE(1)):266.73(cpu)				Fig. 3.32(GWMFE):1207.86(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.1	24	12	4	0.004980	34	15	2	0.002413
0.2	42	24	3	0.001403	84	35	3	0.006260
0.3	88	44	2	0.004293	147	59	2	0.002816
0.4	136	63	3	0.002295	255	111	3	0.000361
t	Fig. 3.33(MFE(2)):581.47				Fig. 3.34(MFE(2)):455.61			
0.1	28	9	3	0.002936	25	13	4	0.007906
0.2	99	45	2	0.000499	56	31	2	0.004891
0.3	168	71	2	0.001005	120	70	2	0.000722
0.4	310	135	2	0.000951	249	149	2	0.001908

Table 3.8: Problem III

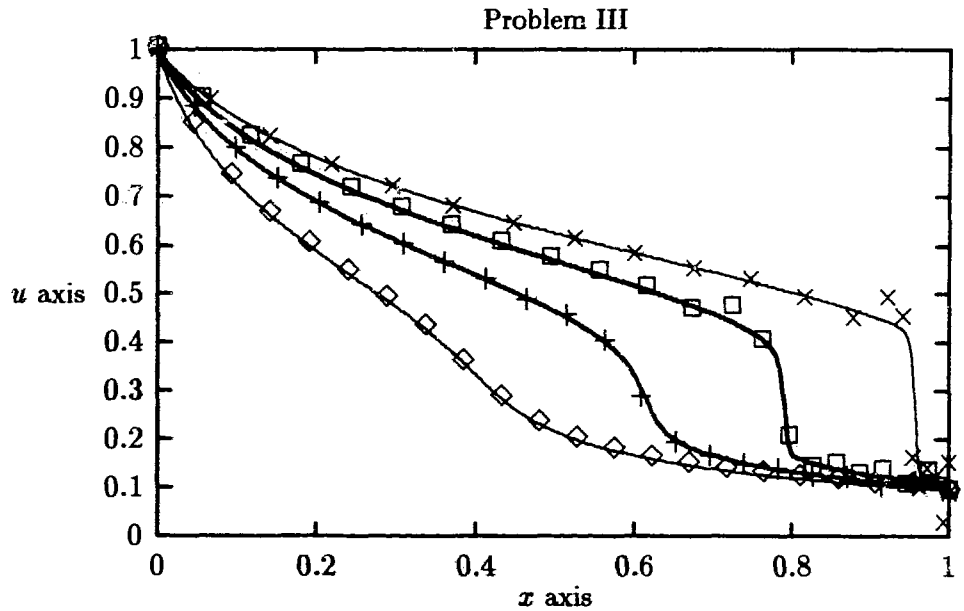

Figure 3.27: $c_1 = 0.1$, $c_2 = 10^{-4}$, $\delta = 10^{-4}$ and $atol = rtol = 10^{-5}$, $cpu = 102.04$.

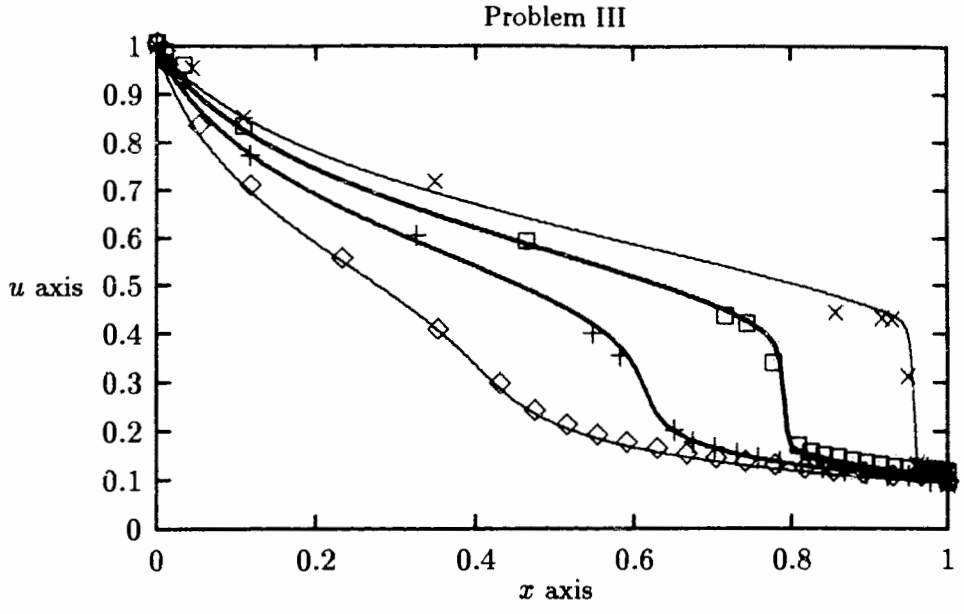

Figure 3.28: $c_1 = 0.01$, $c_2 = 10^{-4}$, $\delta = 10^{-4}$ and $atol = rtol = 10^{-5}$, $cpu = 260.02$.

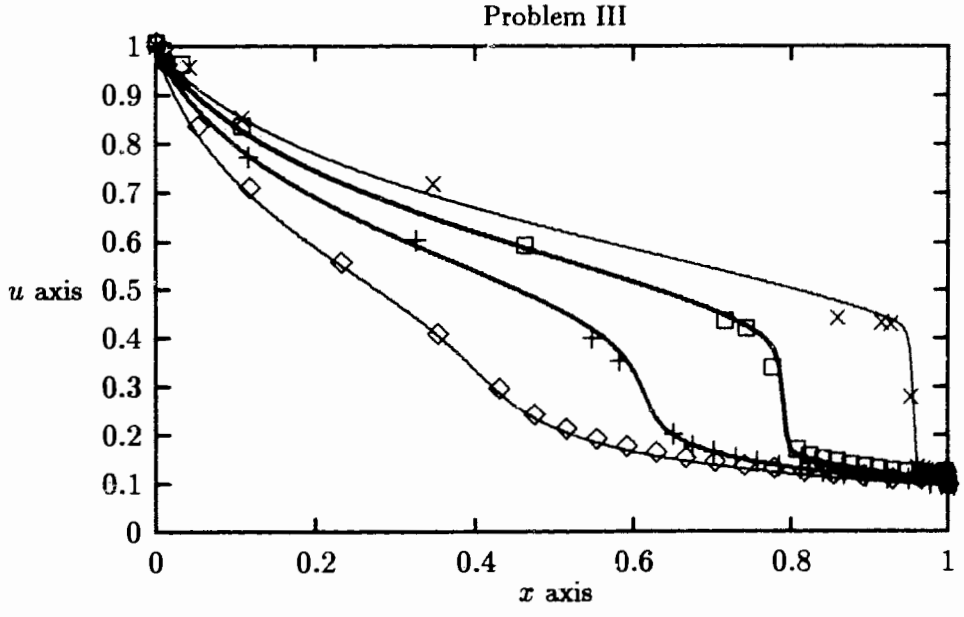

Figure 3.29: $c_1 = 0.01$, $c_2 = 0$, $\delta = 10^{-4}$ and $atol = rtol = 10^{-5}$, $cpu = 536.59$.

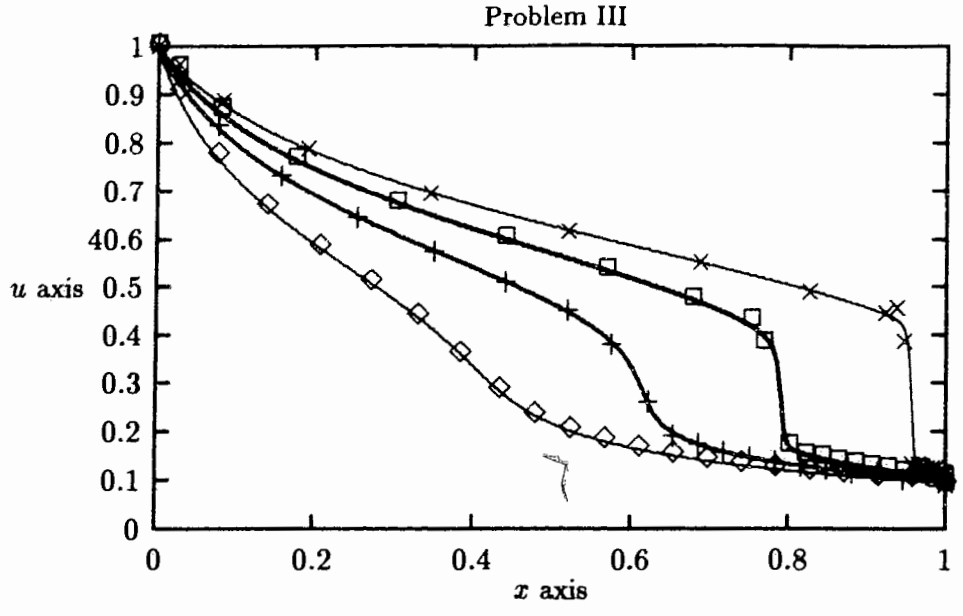

Figure 3.30: $c_1 = 0.025$, $c_2 = 10^{-4}$, $\delta = 10^{-4}$ and $atol = rtol = 10^{-5}$, $cpu = 177.09$.

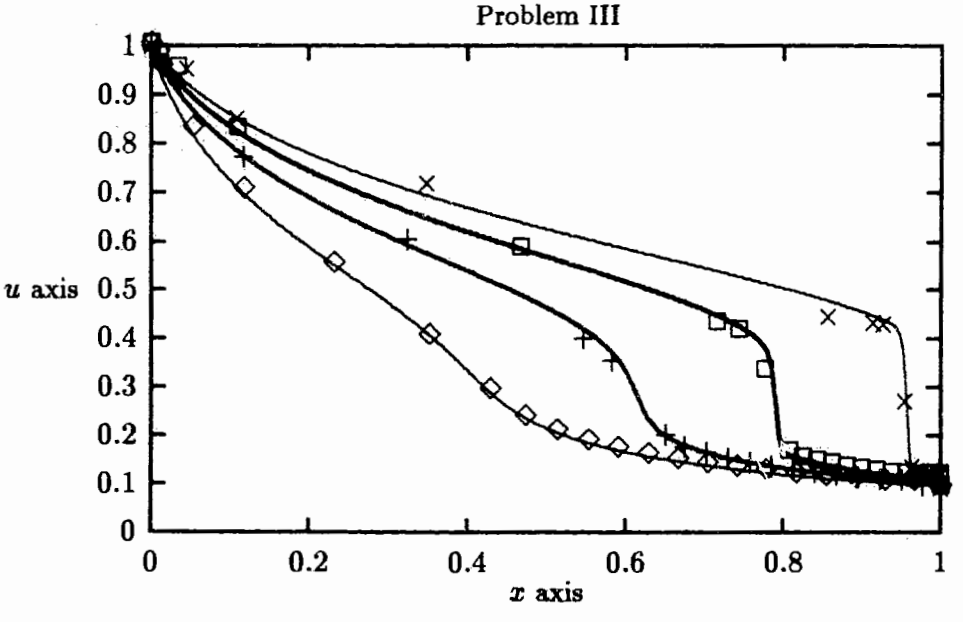

Figure 3.31: $c_1 = 0.01$, $c_2 = 10^{-4}$, $\delta = 5 \times 10^{-4}$ and $atol = rtol = 10^{-5}$, $cpu = 266.73$.

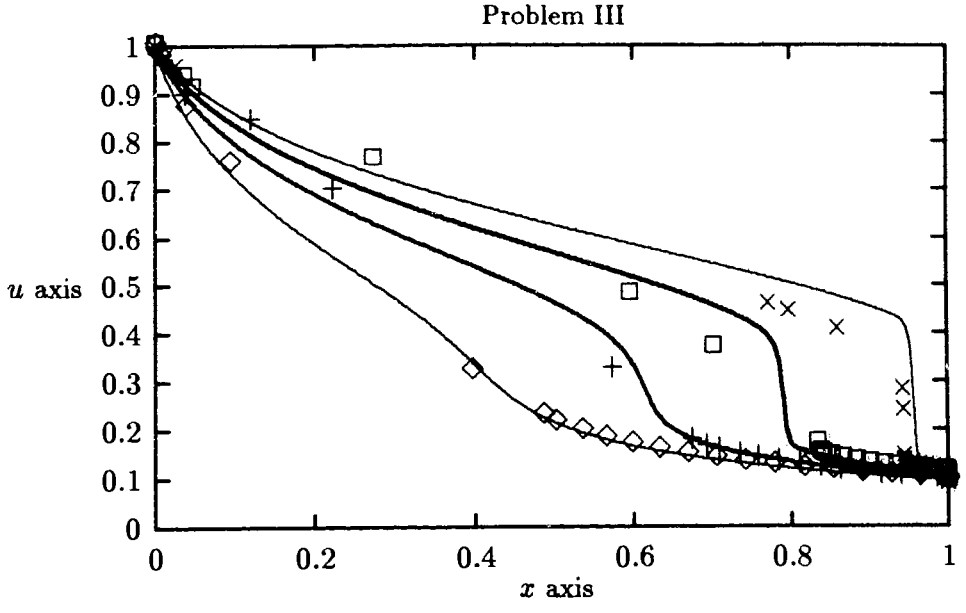

Figure 3.32: $A = (10^{-6})^{1/2}$, $B = 0.0$ and $atol = rtol = 10^{-5}$, $cpu = 1207.86$.

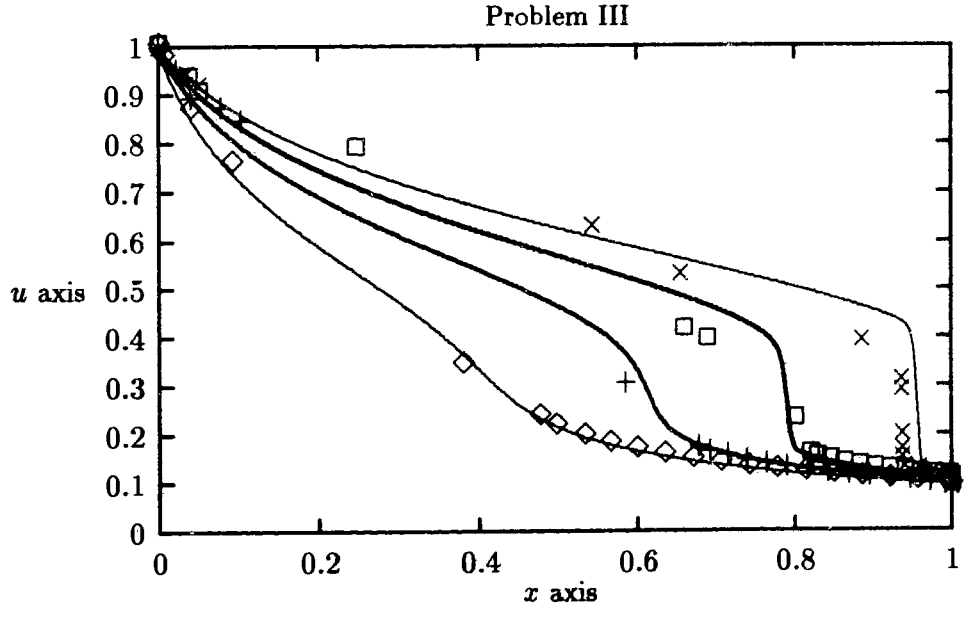

Figure 3.33: $A = (4 \times 10^{-6})^{1/2}$, $B = (10^{-9})^{1/2}$ and $atol = rtol = 10^{-5}$, $cpu = 581.47$.

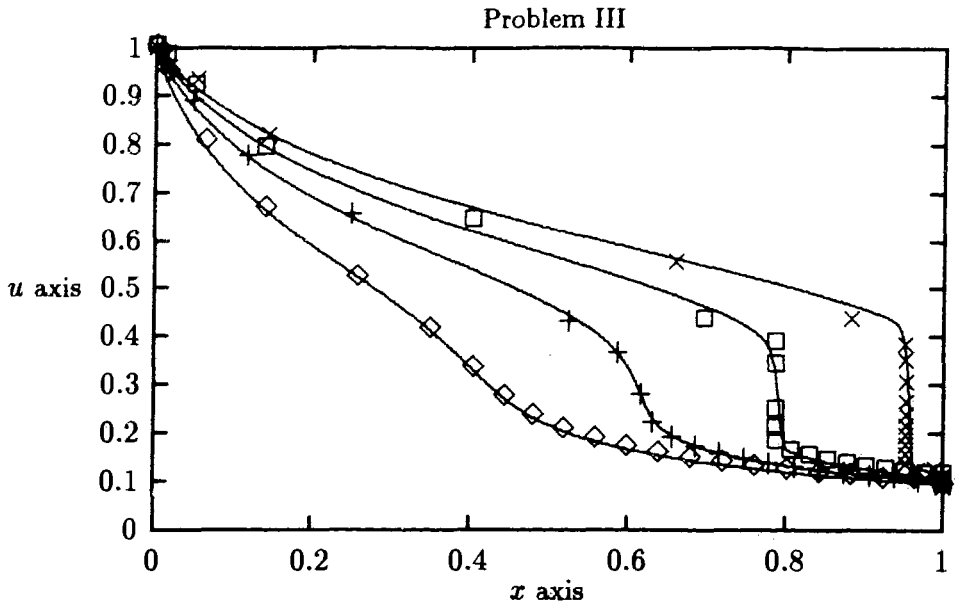

Figure 3.34: $A = (10^{-4})^{1/2}$, $B = 0$ and $atol = rtol = 10^{-5}$, $cpu = 455.61$.

Results for **Method III**

For this problem, Method III performs much better than the moving finite element methods. We first test the method with $k = 2$ and $atol = rtol = 10^{-5}$. The results are shown only for $\tau = 10^{-3}$ and 10^{-4} in Fig. 3.35 and Fig. 3.36, respectively. The qualities of the results for $\tau = 10^{-2}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}$, are similar. The results are satisfactory for $t = 0.1, 0.2, 0.3$. For $t = 0.4$, the numerical solutions at two mesh points move a little higher than the reference solution, as shown in Fig. 3.35 or Fig. 3.36. The cpu time is 69.14 for $\tau = 10^{-3}$ and 93.41 for $\tau = 10^{-4}$, respectively.

If we change k from 2 to 1, and use $\tau = 10^{-4}$, (see Fig. 3.37) the results are better than the results for $k = 2$, although it requires much more time to solve this problem, with cpu time now 273.05.

Problem IV (Heat conduction problem)

Recall that we have tested this problem in Chapter 2 (see page 57 for details). The solution front for this problem travels in the negative direction, which is different from the

t	Fig. 3.35(Method III):69.14(cpu)			
	nst	nje	nqn	tstep
0.1	61	12	4	0.009614
0.2	71	13	4	0.010785
0.3	81	16	4	0.008957
0.4	107	22	3	0.002802
t	Fig. 3.36(Method III):93.41			
	nst	nje	nqn	tstep
0.1	78	17	3	0.008243
0.2	91	18	3	0.006726
0.3	107	20	4	0.007252
0.4	140	29	3	0.003194
t	Fig. 3.37(Method III):273.05			
	nst	nje	nqn	tstep
0.1	86	19	3	0.010845
0.2	103	23	2	0.004965
0.3	118	25	3	0.006318
0.4	181	48	3	0.001443

Table 3.9: Problem III

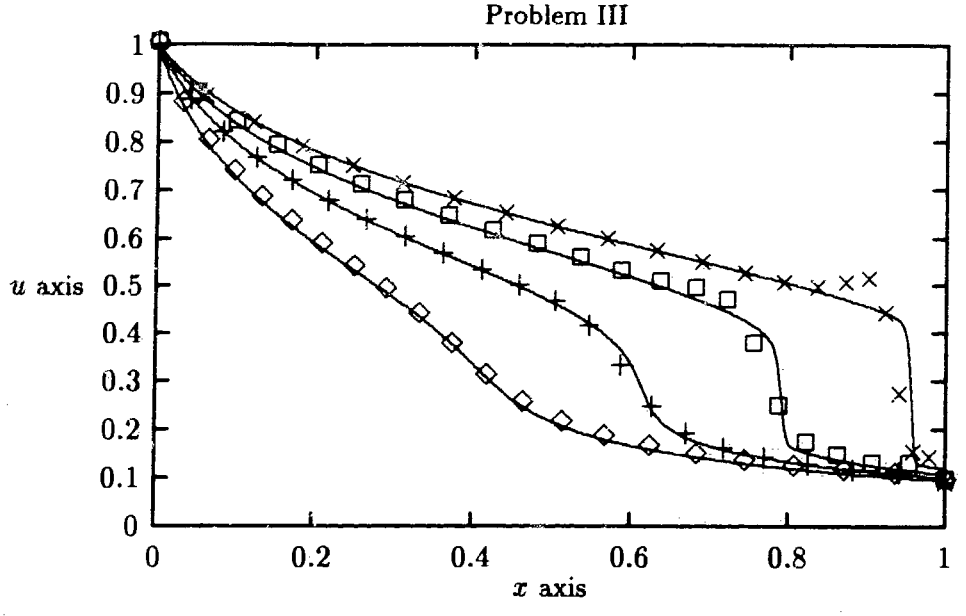

Figure 3.35: $k = 2, \tau = 10^{-3}$ and $atol = rtol = 10^{-5}$, $cpu = 69.14$.

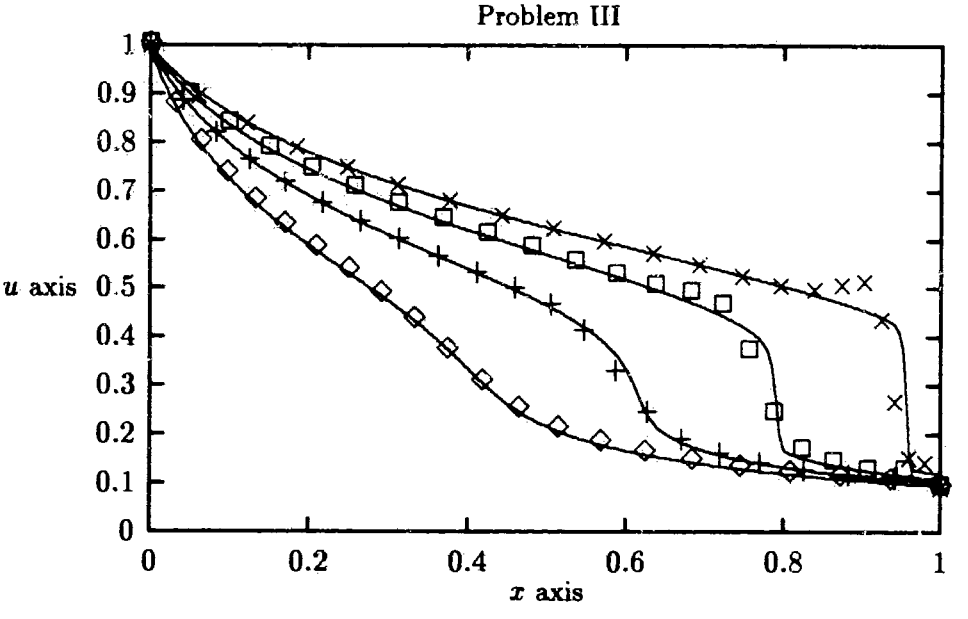

Figure 3.36: $k = 2, \tau = 10^{-4}$ and $atol = rtol = 10^{-5}$, $cpu = 93.41$.

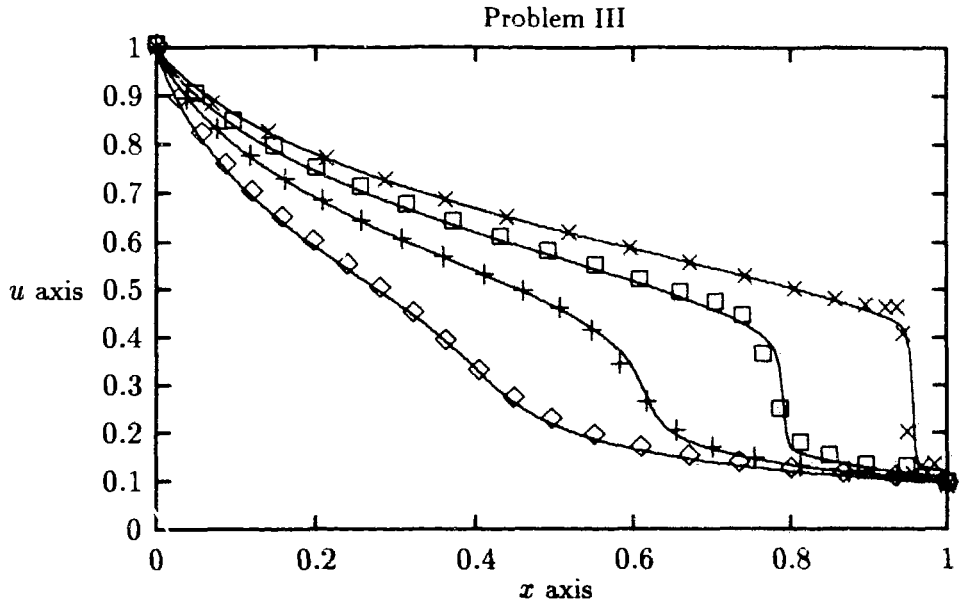

Figure 3.37: $k = 1, \tau = 10^{-4}$ and $atol = rtol = 10^{-5}$, $cpu = 273.05$.

previous three problems. We output results for $t = 0.05, 0.50, 1.00, 1.50, 2.0$.

Results for MFE(1)

We first test the method with the small parameter c_1 used in Problem I, II, and III, but MFE(1) fails to solve it because mesh points move in the positive direction while the solution front travels in the negative direction. Then we use a large parameter value for c_1, namely $c_1 = 3, 2$ which result in an almost non-moving mesh, with $c_2 = 0$, $\delta = 10^{-4}$ and $atol = rtol = 10^{-4}$. The results are shown in Fig. 3.38 and Fig. 3.39 respectively. The oscillating solution occurs at time $t = 1.0$, and almost all of the mesh points have moved to the right. The cpu times are 36.90 and 38.07, respectively.

Results for GWMFE

As for GWMFE, we first use the parameter values A and B recommended by Zegeling and Blom [ZB90]. GWMFE cannot solve it since mesh points move still in the positive direction. Then we increase the parameter A.

(1) For $A = (10^{-4})^{1/2}$, $B = 0$ and $atol = rtol = 10^{-4}$, LSODI is unable to find an initial time step (from $t_0 = 0.05$). The predicted time stepsize is very small (3×10^{-6}).

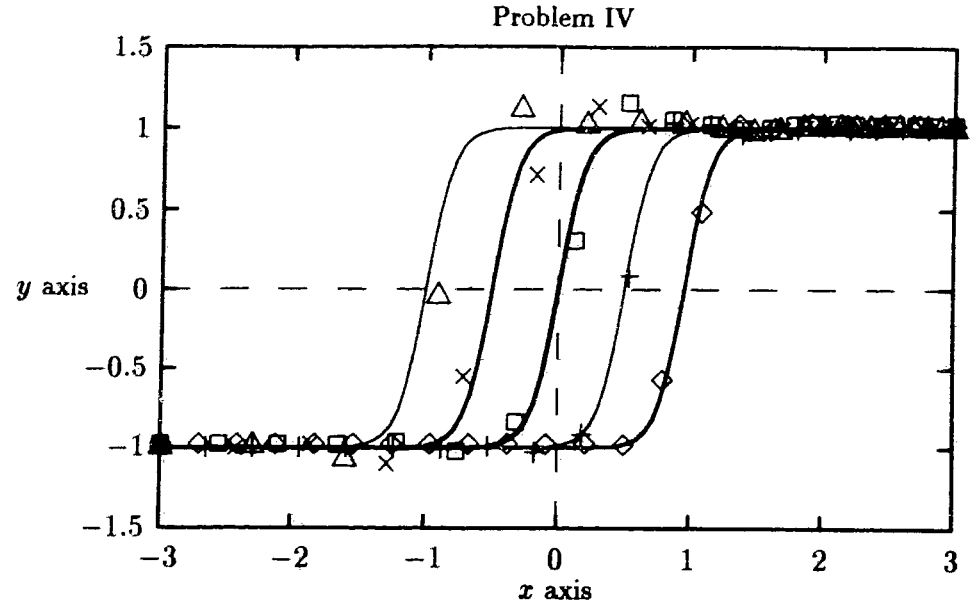

Figure 3.38: $c_1 = 3.0$, $c_2 = 0.0$, $\delta = 10^{-4}$ and $atol = rtol = 10^{-4}$, $cpu = 36.90$.

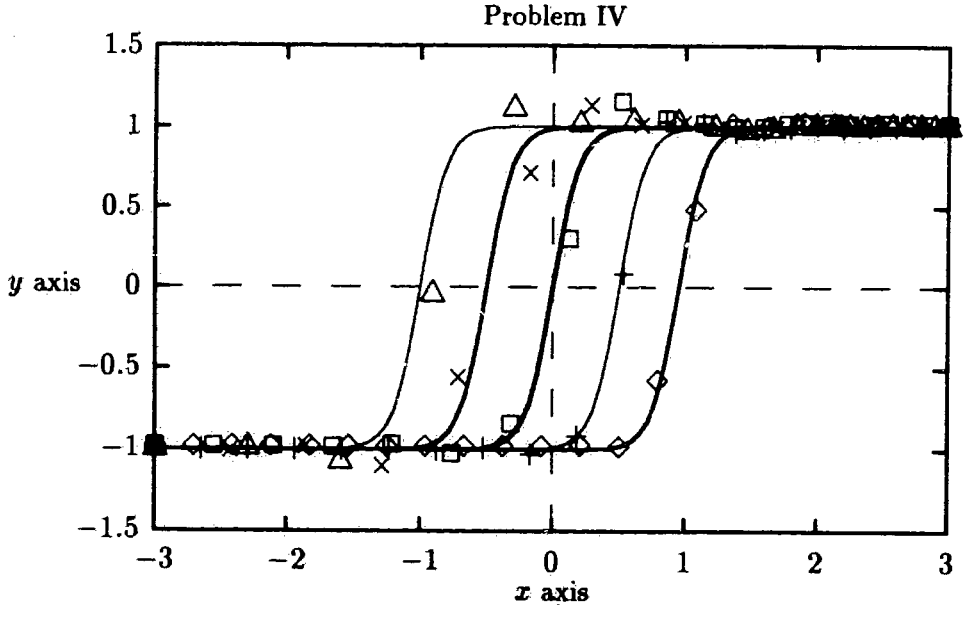

Figure 3.39: $c_1 = 2.0$, $c_2 = 0.0$, $\delta = 10^{-4}$ and $atol = rtol = 10^{-4}$, $cpu = 38.07$.

t	Fig. 3.40(Method III):46.16(cpu)				Fig. 3.41(Method III):39.74(cpu)			
	nst	nje	nqn	tstep	nst	nje	nqn	tstep
0.05	40	10	1	0.016568	43	11	2	0.018570
0.50	67	14	2	0.019736	66	16	1	0.003062
1.00	88	22	2	0.044406	82	19	3	0.037782
1.50	101	24	3	0.053726	98	22	3	0.039853
2.00	121	30	2	0.042174	111	25	3	0.041087

Table 3.10: Problem IV

(2) For $A = (10^{-2})^{1/2}$, $B = 0$ and $atol = rtol = 10^{-4}$, GWMFE stops at $t = 0.25$, again due to a small time stepsize (10^{-6}).

(3) For $A = 1.0$ (or 2.0), $B = 0$ and $atol = rtol = 10^{-4}$, GWMFE produces an unstable solution, so the integration terminates quickly because the computed solution becomes large.

(4) For $A = 1.0$, $B = 10^{-4}$ and $atol = rtol = 10^{-4}$, oscillatory solutions are produced near the front.

Results for **MFE(2)**

MFE(2) is also unsuccessful for this problem.

(1) For $A = (10^{-5})^{1/2}$, $B = (10^{-8})^{1/2}$ and $atol = rtol = 10^{-4}$, oscillatory solutions again occur.

(2) For $A = (10^{-6})^{1/2}$, $B = 0$ and $atol = rtol = 10^{-4}$, MFE(2) stops at the initial stop ($t_0 = 0.05$), with predicted time stepsize being zero.

Results for **Method III**

We test $k = 2, \tau = 10^{-3}$ and $atol = rtol = 10^{-4}$. As shown in Fig. 3.40, an oscillatory solution occurs near the front. There are some mesh points on the layers. The cpu time is 46.16. Decreasing τ to 10^{-4}, the qualities of the results are unchanged (see Fig. 3.41), and the cpu time is 39.74.

3.5 Conclusions

We have tested the moving finite element methods (*i.e.*, MFE(1), MFE(2), GWMFE) and one moving finite difference method, simply using the stiff solver LSODI for the MOL time integration. For the various moving finite element methods, we have had an emphasis on considering the regularization terms and the gradient weighted function. The main difficulty with the MFE methods is that the parameters are problem-dependent, and suitable choices can be very difficult to find, although the recommended choice of the parameters used by the code GWMFE1DS is sometimes helpful. The regularization terms of the GWMFE and MFE(2) force mesh points to move into the steep region of the solution, which causes the reduced ODE system to become extremely stiff. MFE(2) successfully catches the solutions with large gradients as shown in problems I, II, III. GWMFE and MFE(2) move more mesh points into the steep regions of the solutions than MFE(1). It is still not understood why the moving finite element methods can fail altogether to solve the problem where a front moves in the negative x direction (although see [GDM81] and [OD79]). It is conjectured that an upwind scheme for the discrete PDE is needed to match the direction of the mesh movement.

For Method III, there is not much difference if the parameter k is chosen to be 2 or 1. As we mentioned before, the temporal smoothing term serves as an artificial viscosity term, so the parameter τ should be kept small. Method III could not solve the Burgers' problem with 20 mesh points, since too few mesh points move into the steep region, which results in unstable solutions.

The choice of the optimal number of mesh points is a major problem for moving mesh methods. Method III is very expensive for Burgers' problem, although the cpu time for $\tau = 10^{-2}$ is 60% of the cpu time for $\tau = 10^{-3}$. To solve problems I and III, Method III is faster than the moving finite element methods, but it can have some difficulty getting the mesh near a corner in the solution, as shown in problem III. For the final problem, where

a front moves in the negative x direction, Method III has trouble due to oscillations in the computed solution.

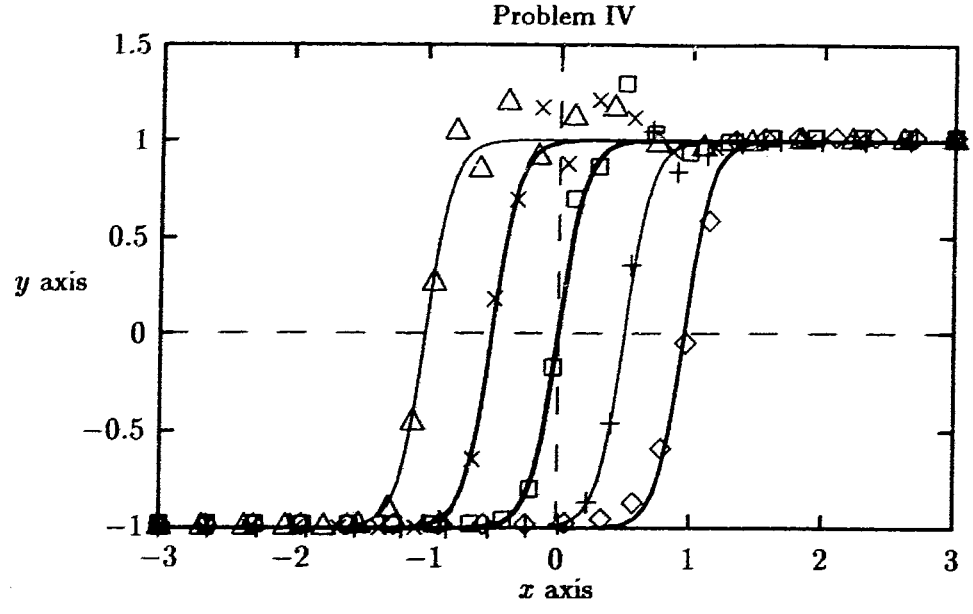

Figure 3.40: $k = 2, \tau = 10^{-3}$ and $atol = rtol = 10^{-4}$, $cpu = 46.16$.

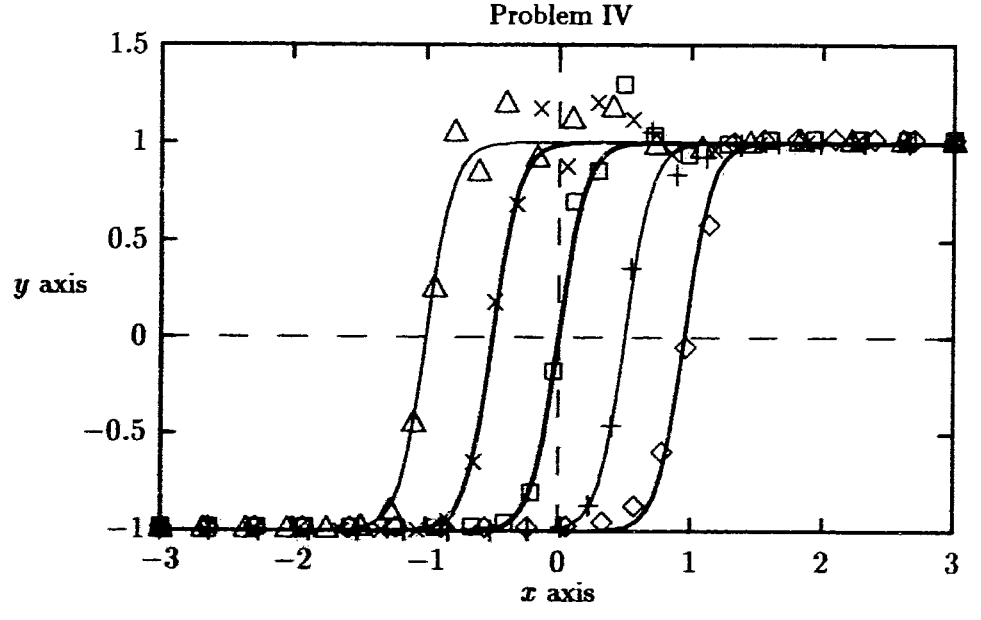

Figure 3.41: $k = 2, \tau = 10^{-4}$ and $atol = rtol = 10^{-4}$, $cpu = 39.74$.

Chapter 4

Summary and Further Problems

This Chapter contains the conclusions and a discussion of some moving mesh problems deserving further study. In this thesis, we study the theory and computation of moving mesh methods for solving one dimensional time dependent partial differential equations. We briefly discuss the three kinds of moving mesh methods – the coordinate transformation method, moving finite element methods and moving finite difference methods. The moving finite element method [MM81] with piecewise linear bases can also be derived by a suitable coordinate transformation [MC85], [LY82] and [BA88]. We discuss the advantages and disadvantages for various moving mesh methods. In Chapter 2, we study methods based (explicitly or implicitly) upon an equidistribution principle. We have presented a new formulation of the equidistribution strategy in terms of a PDE, which is shown to be equivalent to the problem of solving a particular PDE for this new computational coordinate system. We intend to develop further robust moving mesh strategies based directly upon the differential form (2.2.16) or (2.2.29). Here, our intention has been to present some simple ones. Nevertheless, the results indicate that the schemes given here, with simple improvements such as smoothing of the mesh (for Problem II) when necessary, should prove competitive with those which have been recommended by others [FVZ90].

This moving mesh PDE interpretation can be used to understand stability properties for

moving mesh strategies and extends the understanding of the stability properties as given in [CFL86]. While the stability issue for methods based upon equidistribution is a very complicated one, and there is no doubt that a complicated interaction takes place between the PDE (2.1.3) and the mesh PDE (2.2.16) or (2.2.29). We expect that this viewpoint will be used to develop a deeper understanding of stability properties for currently used methods which have proven reliable.

In Chapter 3, we study the moving finite element methods and one moving finite difference method. We have tested these methods using a simple method of lines approach and an existing stiff solver LSODI to solve the ODEs rather than more sophisticated codes. Of particular interest are the role of the regularization terms and the gradient weighted function in MFE. The regularization terms of the GWMFE and MFE(2) force mesh points to move into the stiff regions of the solutions, which cause the reduced ODE system to become extremely stiff. MFE(2) can easily catch the solutions with large gradients as shown in problems I, II, III. GWMFE and MFE(2) move more mesh points into the stiff regions of the solutions than does MFE(1). It is still not understood why the moving finite element method can fail to solve problems with moving fronts that move in the negative x-axis direction (although see [GDM81] and [OD79]). For Method III, the temporal smoothing term serves as an artificial viscosity term, and this interpretation helps show why the parameter τ should be kept small. Method III is very expensive when solving Burgers' problem, even though the cpu time can decrease fast when τ increases (*e.g.*, for $\tau = 10^{-2}$ it is only 60% that for $\tau = 10^{-3}$).

While it is extremely difficult to make general conclusions about the relative merits of the methods, some comments are appropriate. While Method I can be used to interpret previous moving mesh methods, it is not competive in general. We conjecture that an artificial viscosity term is needed to make the method more robust. For Burgers' problem, Method II is not competitive with MFE *if the appropriate parameter values are used.* Nevertheless, for the other problems, Method II proves competitive with other existing methods.

There are still many moving mesh questions deserving further study. The choice of the monitor function and number of mesh points are two such important issues for moving mesh methods (*e.g.*, for Methods II and III). We intend to extend moving mesh strategies based directly upon the differential form (2.2.16) or (2.2.29) to a system of partial differential equations. An equally important question is whether or not the moving mesh strategies (2.2.16) or (2.2.29) can be straight forwardly extended to 2-dimensional partial differential problems. We intend, for example, to investigate the addition of another moving mesh equation for controlling mesh movement in the y-direction,

$$\frac{\partial w_2}{\partial t} + \frac{\partial (w_2 \dot{y})}{\partial y} = 0,$$

where w_2 is the total "average energy" of the solution in the y-direction.

Another problems is how to solve PDEs which involve higher derivatives (*e.g.*, u_{xxx}, as for example, in the Korteweq-deVries equation) with MFE. The moving finite element methods with piecewise linear bases have some difficulties solving these higher order equations.

Finally, convergence and error analysis are further things to consider for moving mesh methods. Dupont studied the moving finite element method [MM81] without regularization terms for the case of smooth solutions of parabolic problems [DU82]. For scalar conservation laws, Lucier showed that the discontinuous solution may be approximated in $L^1(R)$ to within $O(N^{-2})$ by a piecewise linear function with $O(N)$ mesh points when mesh points are moved according to the method of characteristics [LU86]. Nevertheless, general analyses for moving mesh methods have yet to appear.

Bibliography

[AF86b1] S. Adjerid and J. E. Flaherty, A moving-mesh finite element method with local refinement for parabolic partial differential equations, *Comput. Meth. in Appl. Mech. Engng.* **55**, 3-26, 1986.

[AF86b2] S. Adjerid and 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**, 778-795, 1986.

[AF86a] D. C. Arney and J. E. Flaherty, A two-dimensional mesh moving technique for time-dependent partial differential equations, *J. Comput. Phys.* **67**, 124-144, 1986.

[AMR88] U. M. Ascher, R. M. M. Mattheij and R. D. Russell, *Numerical Solution Of Boundary Value Problems For Ordinary Differential Equations*, Prentice-Hall Inc., 1988.

[BA89] M. J. Baines, Moving finite elements and approximate Legendre transformations, Numerical Analysis Report 5, University of Reading, 1989.

[BA88] M. J. Baines, Moving finite elements and envelopes, manuscript, 1988.

[BB86] M. Bieterman and I. Babuska, An adaptive method of lines with error control for parabolic equations of the reaction-diffusion type, *J. Comput. Phys.* **63**, 33-66, 1986.

[BDF86] M. Berzins, P. M. Dew and R. M. Furzeland, Developing P.D.E. software using the method of lines and differential algebraic integrators, Univ. Leeds, Rept., 1986.

[BGMM89] A. Bayliss, D. Gottlieb, B. J. Matkowsky and M. Minkoff, An adaptive pseudo-spectral method for reaction diffusion problems, *J. Comput. Phys.* **81**, 421-443, 1989.

[BO73] C. deBoor, Good approximation by splines with variable knots. II, *in Springer Lecture Notes Series* **363**, Springer-Verlag, Berlin, 1973.

[BSV87] J. G. Blom, J. M. Sanz-Serna and J. G. Verwer, A Lagrangian moving grid scheme for one-dimensional evolutionary partial differential equations, Rept. NM-R8713, CWI, Amsterdam, 1987.

[BSV88] J. G. Blom, J. M. Sanz-Serna and J. G. Verwer, On simple moving grid methods for one-dimensional evolutionary partial differential equations, *J. Comput. Phys.* **74**, 191-213, 1988.

[BV89] J. G. Blom and J. G. Verwer, On the use of the arclength and curvature monitor in a moving-grid method which is based on the method of lines, Rept. NM-N8902, CWI, Amsterdam, 1989.

[BW88] M. J. Baines and A. J. Wathen, Moving finite element methods for evolutionary problems. I. Theory, *J. Comput. Phys.* **79**, 245-269, 1988.

[BZ89] J. G. Blom and P. A. Zegeling, A moving-grid interface for systems of one-dimensional time-dependent partial differential equations, Rept. NM-R89004, CWI, Amsterdam, 1989.

[CFL86] J. M. Coyle, J. E. Flaherty and R. Ludwig, On the stability of mesh equidistribution strategies for time-dependent partial differential equations, *J. Comput. Phys.* **62**, 26-39, 1986.

[CM86] N. Carlson and K. Miller, Gradient weighted moving finite elements in two dimensions, Rept. CPAM342, Univ. of California, Berkeley, 1986.

[CP79] P. Concus and W. Proskurowski, Numerical solution of a nonlinear hyperbolic equation by the Random Choice Method, *J. Comput. Phys.* **30**, 153-166, 1979.

[DD87] E. A. Dorfi and L. O'c. Drury, Simple adaptive grids for 1-D initial value problems, *J. Comput. Phys.* **69**, 175-195, 1987.

[DF82] S. F. Davis and J. E. Flaherty, An adaptive finite element method for initial-boundary value problems for partial differential equations, *SIAM J. Sci. Stat. Comput.* **3**, 6-27, 1982.

[DKS80] H. A. Dwyer, R. J. Kee and B. R. Sanders, Adaptive grid method for problems in fluid mechanics and heat transfer, *AIAA Journal*, **18**, 1205-1212, 1980.

[DM90] P. Degond and F. J. Mustieles, A deterministic approximation of diffusion equations using particles, *SIAM J. Sci. Stat. Comput.* **11**, 293-310, 1990.

[DO72] D. S. Dodson, Optimal order approximation by polynomial spline functions, Ph.D. thesis, Purdue Univ., 1972.

[DO86] J. D. P. Donnelly, A version of moving finite elements for one-dimensional parabolic equations, Rept. 86/19, Oxford Univ. 1986.

[DU82] T. Dupont, Mesh modification for evolution equations, *Math. Comput.* **39**, 85-107, 1982.

[FO77] J. E. Flaherty and R. E. O'Malley, Jr., The numerical solution of boundary value problems for stiff differential equations, *Math. Comput.* **31**, 66-93, 1977.

[FO84] J. E. Flaherty and R. E. O'Malley, Jr., Numerical methods for stiff systems of two-point boundary value problems, *SIAM J. Sci. Stat. Comput.* **5**, 865-885, 1984.

[FS86] P. A. Forsyth, Jr., and P. H. Sammon, Practical considerations for adaptive implicit methods in reservoir simulation, *J. Comput. Phys.* **62**, 265-281, 1986.

[FU85] R. M. Furzeland, The construction of adaptive space meshes for the discretisation of ordinary and partial differential equations, TNER. 85. 022, Thornton Research Center, 1985.

[FVZ90] R. M. Furzeland, J. G. Verwer and P. A. Zegeling, A numerical study of three moving grid methods for one-dimensional partial differential equations which are based on the method of lines, *J. Comput. Phys.* **89**, 349-388, 1990.

[GDM81] R. J. Gelinas, S. K. Doss and K. Miller, The moving finite element method: application to general partial differential equations with multiple large gradients, *J. Comput. Phys.* **40**, 202-249, 1981.

[GS87] V. Giovangigli and M. D. Smooke, Adaptive continuation algorithms with application to combustion problems, Rept. ME-101-87, Yale Univ., 1987.

[HA83] A. Harten, High resolution schemes for hyperbolic conservation laws, *J. Comput. Phys.* **49**, 357-393, 1983.

[HI80] A. C. Hindmarsh, LSODE and LSODI, two new initial value ordinary differential equation solvers, *ACM SIGNUM Newsletter* **15**, 10-11, 1980.

[HMW86] A. N. Hrymak, G. J. McRae and A. W. Westerberg, An implementation of a moving finite element method, *J. Comput. Phys.* **63**, 168-190, 1986.

[HN84] J. M. Hyman and M. J. Naughton, Static rezone methods for tensor-product grids, *in Proceedings of SIAM-AMS Conference on Large Scale Computations in Fluid Mechanics,* SIAM, Philadelphia, 1984.

[HMS82] B. M. Herbst, A. R. Mitchell and S. W. Schoombie , A moving Petrov-Galerkin method for transport equations, *Inter. J. Numer. Math. in Engng.* **18**, 1321-1336, 1982.

[HSM83] B. M. Herbst, S. W. Schoombie and A. R. Mitchell, Equidistributing principles in moving finite element methods, *J. Comput. Appl. Math.* **9**, 377-389, 1983.

[HSM84] B. M. Herbst, S. W. Schoombie and A. R. Mitchell, Generalized Petrov-Galerkin methods for the numerical solution of Burger's equation, *Inter. J. Numer. Math. in Engng.* **20**, 1273-1289, 1984.

[HL86] J. M. Hyman and B. Larrouturou, Dynamic rezone methods for partial differential equations in one space dimension, Los Alamo, Rept. LA-UR-86-1678, 1986.

[HY76] J. M. Hyman, The method of lines solution of partial differential equations, Coo-3077-139, Courant Institute, New York Univ., 1976.

[HY82] J. M. Hyman, Adaptive moving mesh methods for partial differential equations, Los Alamos, Rept. LA-UR-82-3690, 1982.

[HY83] J. M. Hyman, Adaptive moving mesh methods for partial differential equations, *in Advances in Reactor Computations,* American Nuclear Society Press, La Grange Park, Illinois, 24-43, 1983.

[HY84] J. M. Hyman, Moving mesh methods for initial boundary value problems, Rept. LA-UR-84-61, Los Alamos, 1984.

[JWB88] I. W. Johnson, A. J. Wathen and M. J. Baines, Moving finite element methods for evolutionary problems, II. Applications, *J. Comput. Phys.* **79**, 270-297, 1988.

[KE71] H. B. Keller, A new difference scheme for parabolic problems, *in Numerical Solution Of Partial Differential Equations II,* B. Hubbard, ed., Academic Press, New York, 327-350, 1971.

[LA89] B. Larrouturou, A conservative adaptive method for flame propagation, *SIAM J. Sci. Stat. Comput.* **10**, 742-755, 1989.

[LA73] P. D. Lax, *Hyperbolic Systems Of Conservation Laws And Mathematical Theory Of Shock Waves*, SIAM, Philadelphi, 1973.

[LE88] R. J. LeVeque, High resolution finite volume methods on arbitrary grids via wave propagation, *J. Comput. Phys.* **78**, 36-63, 1988.

[LU86] B. J. Lucier, A moving mesh numerical method for hyperbolic conservation laws, *Math. Comput.* **46**, 59-69, 1986.

[LY82] D. R. Lynch, Unified approach to simulation on deforming elements with application to phase change problems, *J. Comput. Phys.* **47**, 387-411, 1982.

[LY88] R. J. LeVeque and H. C. Yee, A study of numerical methods for hyperbolic conservation laws with stiff source terms, NASA Ames Reseach Center, Rept. 100075, 1988.

[MA84] N. K. Madsen, MOLAG: A method of lines adaptive grid interface for nonlinear partial differential equations, *in PDE Software: Modules, Interfaces and Systems*, B. Engquist & T. Smedsaas (eds.), North Holland, 1984.

[MC85] A. C. Mueller and G. F. Carey, Continuously deforming finite elements, *Inter. J. Numer. Meths. in Engng.* **21**, 2099-2126, 1985.

[MD88] K. Matsuno and H. A. Dwyer, Adaptive methods for elliptic grid generation, *J. Comput. Phys.* **77**, 40-52, 1988.

[MI81] K. Miller, Moving finite elements II, *SIAM J. Numer. Anal.* **18**, 1033-1057, 1981.

[MI83] K. Miller, Alternate modes to control the nodes in the moving finite element method, *in Adaptive Computational Methods For Partial Differential Equations*, I. Babuska, J. Chandra & J. E. Flaherty (eds.), SIAM, Philadelphia, 1983.

[MI88] K. Miller, Private Communication, 1988.

[MM81] K. Miller and R. N. Miller, Moving finite elements I , *SIAM J. Numer. Anal.* **18**, 1019-1032, 1981.

[MO85] M. C. Mosher, A variable node finite element method, *J. Comput. Phys.* **57**, 157-187, 1985.

[OD79] G. R. Otey and H. A. Dwyer, Numerical study of the interaction of fast chemistry and diffusion, *AIAA Journal*, **17**, 606-613, 1979.

[PE82] L. R. Petzold, A description of DASSL: A differential/algebraic system solver, SAND82-8637, Sandia Labs, Livermore, Cal., 1982.

[PE87] L. R. Petzold, Observations on an adaptive moving grid method for one-dimensional systems of partial differential equations, *Appl. Numer. Math.* **3**, 347-360, 1987.

[PR89] J. D. Pryce, On the convergence of iterated remeshing, *IMA J. Numer. Anal.* **9**, 315-335, 1989.

[PS75] V. Pereyra and G. Sewell, Mesh selection for discrete solution of boundary value problems in ordinary differential equations, *Numer. Math.* **23**, 261-268, 1975.

[RC78] R. D. Russell and J. Christiansen, Adaptive mesh selection strategies for solving boundary value problems, *SIAM J. Numer. Anal.* **15**, 59-80, 1978.

[RO88] D. J. Romero, The adaptive solution of evolutive and equilibrium distributed systems, Ph.D Thesis, Chem. Engng., Imperial College, London, 1988.

[RR91] Yuhe Ren and Robert D. Russell, Moving mesh techniques based upon equidistribution, and their stability, *to appear in SIAM J. Sci. Stat. Comput.* 1991.

[RS88] D. J. Romero and R. W. H. Sargent, Improving the robustness of the moving finite element method, *Comput. Chem. Engng.* **12**, 433-441, 1988.

[RSJ87] D. J. Romero, R. H. W. Sargent and W. P. Jones, The behaviour of the moving finite element in one dimension problems: error analysis and control, Imperial College, London, Rept. 3, 1987.

[RSR87] Y. Reuven, M. D. Smooke and H. Rabitz, Sensitivity analysis of one-dimensional mixed initial-boundary value problems: application to freely propagating premixed laminar flames, Yale Univ., Rept. ME-102-87, 1987.

[RU79] R. D. Russell, Mesh selection methods, *in Codes For Boundary Value Problems, Lecture Notes in Computer Science* **74** Childs *et al.*, eds., Berlin: Springer, 1979.

[SC77] N. L. Schryer, Numerical solution of time-varying partial differential equations in one space variable, Bell Lab. Comput. Sci. Tech. Rept. 53, 1977.

[SC80] N. L. Schryer, Numerical solution of coupled systems of partial differential equations in one spatial variable and time, Rept. July, 1980, Bell Laboratories.

[SK83] M. D. Smooke and M. L. Koszykowski, Fully adaptive solutions of one-dimensional mixed initial-boundary value problems with applications to unstable problems in combustion, SANDIA Rept. Sand83-8219, 1983.

[SK81a] R. D. Skeel, Improving routines for solving parabolic equations in one space dimension, Numer. Anal. Rept. No. 63, Dept. of Maths., Univ. of Manchester, U. K. 1981.

[SK81b] R. D. Skeel, Local error estimation and mesh adaptation for parabolic equations in one space dimension, Numer. Anal. Rept. No. 65, Dept. of Maths. Univ. of Manchester, U. K. 1981.

[SM82] M. D. Smooke, Solution of burner-stabilized pre-mixed laminar flames by boundary value methods, *J. Comput. Phys.* **48**, 72-105, 1982.

[TS86] R. Thrasher and K. Sepehmoori, On equidistributing principles in moving finite element methods, *J. Comp. Appl. Math.* **16**, 309-318, 1986.

[VBFZ88] J. G. Verwer, J. G. Blom, R. M. Furzeland and P. A. Zegeling, A moving grid method for one-dimensional PDEs based on the method of lines, Rept. NM-R8818, CWI, Amsterdam, 1988.

[VBS88] J. G. Verwer, J. G. Blom and J. M. Sanz-Serna, An adaptive moving grid method for one-dimensional systems of partial differential equations, Rept. NM-R8804, CWI, Amsterdam, 1988.

[VHS89] J. G. Verwer, W. H. Hundsdorfer and B. P. Sommeijer, Convergence properties of the Runge-Kutta-Chebyshev method, Rept. NM-R8907, CWI, Amsterdam, 1989.

[VR50] J. VonNeuman and R. D. Richtmyer, A method for the numerical calculation of hydrodynamic shocks, *J. Appl. Phys.* **21**, 232-237, 1950.

[VS86] F. Vadillo and J. M. Sanz-Serna, Studies in numerical nonlinear instability. II. A new look at Ut+UUx = 0, *J. Comput. Phys.* **66**, 225-238, 1986.

[WA86] A. J. Wathen, Mesh-independent spectra in the moving finite element equations, *SIAM J. Numer. Anal.* **23**, 797-813, 1986.

[WB85] A. J. Wathen and M. J. Baines, On the structure of the moving finite-element equations, *IMA J. Numer. Anal.* **5**, 161-182, 1985.

[WH74] G. B. Whitham, *Linear And Nonlinear Waves,* John Wiley & Sons, New York, 1974.

[WH79] A. B. White, Jr., On selection of equidistributing meshes for two-point boundary-value problems, *SIAM J. Numer. Anal.* **16**, 472-502, 1979.

[WH82] A. B. White, Jr., On the numerical solution of initial/boundary-value problems in one space dimension, *SIAM J. Numer. Anal.*, **19**, 683-697, 1982.

[WNN84] K. H. A. Winkler, M. L. Norman and M. J. Newman, Adaptive mesh techniques for fronts in star formation, *Physica* **12D**, 408-425, 1984.

[ZB90] P. A. Zegeling and J. G. Blom, An evaluation of the gradient weighted moving-finite-element method in one space dimension, Rept. NM-R 9006, CWI, Amsterdam, 1990.