JOURNAL OF COMPUTATIONAL PHYSICS **131,** 280–298 (1997)
ARTICLE NO. CP965592

# Solution of Evolutionary Partial Differential Equations Using Adaptive Finite Differences with Pseudospectral Post-processing

L. S. Mulholland,* Y. Qiu,† and D. M. Sloan‡

*Department of Mathematics, University of Strathclyde, Glasgow G1 1XH, Scotland,*
E-mail: ‡d.sloan@strath.ac.uk

Received February 9, 1996; revised September 23, 1996

A coordinate transformation approach is described that enables pseudospectral methods to be applied efficiently to unsteady differential problems with steep solutions. The work is an extension of a method presented by Mulholland, Huang, and Sloan for the adaptive pseudospectral solution of steady problems. A coarse grid is generated by a moving mesh finite difference method that is based on equidistribution, and this grid is used to construct a time-dependent coordinate transformation. A sequence of spatial transformations may be generated at discrete points in time, or a single transformation may be generated as a continuous function of space and time. The differential problem is transformed by the coordinate transformation and then solved using a method that combines pseudospectral discretisation in space with a suitable integrator in time. Numerical results are presented for unsteady problems in one space dimension. 

## 1. INTRODUCTION

Adaptive grid methods have been used widely during the last few years for solving differential equations with steep, but continuous solutions. There is ample numerical evidence that significant improvements in accuracy and computational efficiency can be obtained by adapting mesh points so that they are concentrated in regions of large solution variation. A useful approach in adaptive schemes is the concept of equidistribution, which seeks to distribute some function uniformly over the domain of the problem. This function is usually some measure of the local computational error or the local solution variation. The text by Thompson *et al.* [17] and the paper by Huang and Sloan [11] give an interpretation of equidistribution in the context of adaptive grid generation for steady, one- and two-dimensional problems. An equidistribution principle is developed in [11] and it is used to formulate a finite difference grid generation algorithm in two space dimensions.

The aim of this paper is to describe a robust method for the adaptive solution of evolutionary partial differential equations (PDEs) in one space dimension. For unsteady problems, adaptive methods may be classified as *static* or *dynamic*. In the static approach the numerical solution is advanced in time on a fixed nonuniform grid, and after each step (or series of steps) a regridding is carried out to give new nodal locations that are adapted in some sense to the computed solution. This is followed by an interpolation process that yields approximations on the new grid to serve as initial values for the next time step. In the static method there is no essential coupling between the discretisation of the PDE and the grid generation. The reader is referred to papers by Sanz-Serna and Christie [16] and Bieterman and Babuška [2] for illustrations of the static regridding approach.

In the dynamic adaptive methods there is a coupling between the approximate solution of the PDE and the mesh generation. These methods—often referred to as moving mesh methods—are potentially very powerful. The mesh evolves with the solution in some near-optimal manner, and this enables sharp fronts to be computed with a high degree of accuracy. The key disadvantage of the dynamic approach is that a large set of unknowns has to be evaluated at each time step. Dynamic methods that have attracted a considerable interest are the moving finite element method of Miller and Miller [13, 14] and the moving finite difference method of Dorfi and Drury [6]. Recently, Huang, Ren, and Russell [8, 9] have presented several moving mesh PDEs that are based on the concept of equidistribution. Their moving mesh equations have solutions that are continuous in space and time, and the equations are designed so that the nodal locations satisfy an equidistribution principle (EP). In practice, the moving mesh equations in [8, 9] are conjoined with the given PDE in the discretisation process.

Our purpose here is to present a moving mesh method that incorporates a pseudospectral (PS) post-processing step. We give an extension of a method for the adaptive PS solution of steady problems that was presented recently by Mulholland, Huang, and Sloan [15]. This earlier work

* Supported by the Engineering and Physical Sciences Research Council (EPSRC).

† Supported by the University of Strathclyde and by the ORS Awards Scheme.



0021-9991/97 $25.00

by Mulholland *et al.* showed how to improve the effectiveness of PS methods for steep solutions in one and two space dimensions by adapting a coordinate transformation to a quickly computed finite difference solution. Application of the coordinate transformation smooths out regions of high gradient and a highly accurate PS solution is then obtained in the transformed spatial coordinate. Coordinate transformations have been used by several authors as a means of smoothing out steep fronts prior to PS discretisation. However, the transformation adopted has generally been based on functions of known structure, and the qualitative nature of the solution has been known a priori (for example, see [1] and references therein). The transformation presented in [15], on the other hand, uses many parameters, and these are determined by adapting the map to features of the solution that have been generated numerically: this approach gives the method wide applicability. Here, the numerically generated transformation idea is applied to time-dependent problems. A coarse grid is generated by a moving mesh finite difference method that is based on equidistribution, and this grid is used to construct a time-dependent coordinate transformation. It is shown that the transformation may be generated at discrete points in time—essentially a map between physical and computational coordinates in space. We shall refer to the solution process based on this mode of map generation as the *discrete time algorithm.* We also show that the moving mesh finite difference solution at many discrete time values may be used to generate a transformation in which the physical coordinate is expressed as a continuous function of both time and computational space coordinate. This second formulation enables the PDE to be transformed to coordinates in which the solution is extremely smooth in both time and space, and this provides ideal conditions for a highly accurate solution technique based on PS discretisation in space conjoined with an accurate integrator in time. This approach will be called the *continuous time algorithm.*

The coarse adapted mesh is readily generated using one of the moving mesh partial differential equations (MMPDEs) of Huang, Ren, and Russell [8]. However, numerical experiments persuaded us that there are gains in accuracy and efficiency if the MMPDE is replaced by a method that couples the numerical solution of the PDE with the algebraic condition arising from the equidistribution constraint. The mesh is thus generated by solving a system of differential algebraic equations (DAEs)—a process that has been described as stable by Coyle, Flaherty, and Ludwig [5]. The DAE approach has been adopted throughout our numerical computations.

The discrete time algorithm is described in Section 2. Given the coarse mesh solution and the PS post-processed solution at time $t$, the moving mesh method is used to obtain the coarse mesh solution at time $t + \Delta t$. Employing a process analogous to that adopted by Mulholland *et al.* [15] for steady problems, a Chebyshev polynomial expansion is fitted to the coarse mesh nodal values at time $t + \Delta t$ to yield a coordinate transformation. The post-processed solution of the transformed equations is then carried forward to time $t + \Delta t$ using a Crank–Nicolson or BDF (see, for example [12]) time step and PS discretisation in the computational space coordinate. Section 3 deals with some of the dynamics of the DAE moving mesh method. In particular, we show that there is no node crossing in our implementation and that a local quasi-uniformity condition is maintained. The continuous time algorithm is presented in Section 4. In this case, the coarse mesh solution is computed over many time steps and the array of nodal values at discrete values of computational coordinate and time is used to construct a continuous map in one space dimension and time. The transformed equation is then solved using a BDF method in time coupled with PS discretisation in space. Numerical results are given in Sections 2 and 4 to illustrate the performances of the algorithms, and Section 5 contains conclusions and comments on our PS algorithms.

## 2. DISCRETE TIME ALGORITHM

### 2.1. *Test Problems*

All numerical computations described in the paper were performed on problems associated with the one-dimensional Burgers equation, and reference is made to these problems at several points within the paper. It is appropriate, therefore, to describe the algorithms as they apply to Burgers' equation, noting, of course, that they will have much wider applicability. Furthermore, it is convenient to present the test problems at this preliminary stage. Accordingly, we consider the PDE

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \varepsilon\frac{\partial^2 u}{\partial x^2} = 0, \qquad x \in (x_L, x_R), t > 0, \quad (2.1)$$

subject to initial and boundary conditions

$$u(x, 0) = u_0(x), \qquad x_L \le x \le x_R, \quad (2.2)$$

$$u(x_L, t) = b_L(t), \qquad u(x_R, t) = b_R(t), t \ge 0. \quad (2.3)$$

Here $\varepsilon$ is a positive constant that is small ($\varepsilon \ll 1$) in the cases of computational interest. The first two test problems given below have been used in computational experiments by Blom, Sanz-Serna, and Verwer [3], and the third has been used in numerical tests by Canuto *et al.* [4]. Exact solutions for the second and third problems have been given by Whitham [18].

*Problem* I. This is the problem with exact solution

$$u(x, t) = c - d\tanh\left\{\frac{d}{2\varepsilon}(x - ct - x_0)\right\}, \quad (2.4)$$

where $c = \frac{1}{2}(u^- + u^+)$, $d = \frac{1}{2}(u^- - u^+)$, with $d > 0$. It describes a travelling front joining the upstream state $u^-$ and the downstream state $u^+$. The front moves with velocity $c$ and is initially at location $x = x_0$. The exact solution provides the initial and boundary values. The boundaries are placed at $x_L = 0$, $x_R = 1$, and the parameter values used are $u^- = 1$, $u^+ = 0$, $x_0 = \frac{1}{4}$.

*Problem* II. This problem has exact solution

$$u(x,t) = 1 - 0.9\frac{r_1}{r_1 + r_2 + r_3} - 0.5\frac{r_2}{r_1 + r_2 + r_3}, \quad (2.5)$$

where

$$r_1 = \exp\left(-\frac{x - 0.5}{20\varepsilon} - \frac{99t}{400\varepsilon}\right),$$

$$r_2 = \exp\left(-\frac{x - 0.5}{4\varepsilon} - \frac{3t}{16\varepsilon}\right), \qquad r_3 = \exp\left(-\frac{x - 3/8}{2\varepsilon}\right).$$

This solution represents two fronts that merge as time evolves. As in the first problem, the boundaries are placed at $x_L = 0$, $x_R = 1$.

*Problem* III. The third problem has a solution that is $2\pi$-periodic in space, and it represents a wave moving with velocity $c$ in the direction of increasing $x$. The exact solution is

$$u(x,t) = c - 2\varepsilon\frac{(\partial\phi/\partial x)(x - ct, t + 1)}{\phi(x - ct, t + 1)}, \quad (2.6)$$

where

$$\phi(x,t) = \sum_{n=-\infty}^{\infty} \exp(-(x - (2n + 1)\pi)^2/4\varepsilon t).$$

For this problem the boundaries are placed at $x_L$ and $x_L + 2\pi$, and $x_L$ will be specified as computational results are presented. The parameter $c$ is set to the value 4 in all computations. Note that the exact solution is given incorrectly in Canuto [4].

### 2.2. *Formulation of Algorithm*

Initially, we outline the moving mesh method that is used to generate the coarse mesh solution. To this end, Eq. (2.1) is recast in terms of independent variables $\eta$ and $t$, where $\eta$ is defined by a one-to-one coordinate transformation of the form

$$x = x(\eta, t). \quad (2.7)$$

$x$ and $\eta$ denote the spatial coordinates in physical and computational space, respectively, and the time-dependent map (2.7) from the computational domain, $D_c$, to the physical domain, $D_p$, is constructed using an EP. At time $t$, the map (2.7) defines a set of nodes on $D_p$ that corresponds to a given uniform mesh on $D_c$ [11]. Without loss of generality we choose $D_c$ to be $[-1, 1] \subset \mathbf{R}$ and let (2.7) relate the evenly spaced nodes

$$\eta_i = -1 + \frac{2i}{n}, \qquad i = 0, 1, ..., n, \quad (2.8)$$

to the nodes $\{x_i\}_{i=0}^n$ in $[x_L, x_R] = D_p$. Here

$$x_L = x_0(t) < x_1(t) < \cdots < x_n(t) = x_R, \qquad \forall t \geq 0,$$

and

$$x_i(t) = x(\eta_i, t), \qquad i = 0, 1, ..., n. \quad (2.9)$$

Under the transformation (2.7) the dependent variable in (2.1) becomes $u(x(\eta, t), t)$, and this is denoted by $w(\eta, t)$. The aim is to follow the time evolution of the approximate solution at moving nodes (2.9), these being related by the map (2.7) to the fixed nodes (2.8) in $D_c$. It is convenient, therefore, to express the time derivative in (2.1) as a derivative along lines of constant $\eta$, and this requires the relation

$$\frac{\partial w}{\partial t} = \frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\frac{\partial x}{\partial t}, \quad (2.10)$$

in the obvious notation. If $\partial u/\partial t$ is eliminated from the differential equation (2.1) by means of (2.10) we obtain

$$\dot{u} - \dot{x}\frac{\partial u}{\partial x} + u\frac{\partial u}{\partial x} - \varepsilon\frac{\partial^2 u}{\partial x^2} = 0, \quad (2.11)$$

where $\dot{u} \equiv \partial w/\partial t$ and $\dot{x} \equiv \partial x/\partial t$ denote derivatives in which $\eta$ is held constant.

In the moving mesh method a mesh generating equation based on equidistribution is conjoined with (2.11) to give a system of equations that determines the time evolution of $x(\eta, t)$ and $u(x(\eta, t), t) = w(\eta, t)$ at nodes (2.8). We seek approximations to the time-dependent vectors $\{x_i\}_{i=0}^n$ and $\{u_i\}_{i=0}^n$, where

$$u_i(t) = u(x(\eta_i, t), t) = w_i(t), \quad (2.12)$$

and $x_i(t)$ is given by (2.9). The semi-discrete version of (2.11) that we use in the approximation process is

$$\dot{u}_i - \dot{x}_i \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}} + u_i \frac{u_{i+1} - u_{i-1}}{x_{i+1} - x_{i-1}} = \frac{2\varepsilon}{x_{i+1} - x_{i-1}} \left( \frac{u_{i+1} - u_i}{x_{i+1} - x_i} - \frac{u_i - u_{i-1}}{x_i - x_{i-1}} \right) \tag{2.13}$$

for $i = 1, 2, ..., n - 1$, with boundary conditions

$$u_0(t) = b_L(t), \qquad u_n(t) = b_R(t), t \geq 0. \tag{2.14}$$

Following Huang *et al.* [8] we use the EP

$$\int_{x_L}^{x(\eta,t)} M(s,t)\, ds = \eta\theta(t), \tag{2.15}$$

where

$$\theta(t) = \int_{x_L}^{x_R} M(s,t)\, ds. \tag{2.16}$$

$M$ denotes the monitor function that has to be equally distributed between nodes in $D_p$, and here we use a function based on scaled arc-length [11] defined by

$$M(x,t) = \sqrt{1 + \alpha^2 \left( \frac{\partial u}{\partial x} \right)^2}, \tag{2.17}$$

where $\alpha$ is a real parameter that determines the extent to which the solution gradient influences grid placement. Differentiation of (2.15) with respect to $\eta$ gives the differential form of the EP,

$$\frac{\partial}{\partial \eta} \left( M(x(\eta,t),t) \frac{\partial}{\partial \eta} x(\eta,t) \right) = 0. \tag{2.18}$$

Huang *et al.* [8] use (2.18) to derive a series of MMPDEs, and numerical experiments described in [9] suggest that the most accurate of these is

$$\frac{\partial^2 \dot{x}}{\partial \eta^2} = -\frac{1}{\tau} \frac{\partial}{\partial \eta} \left( M \frac{\partial x}{\partial \eta} \right), \tag{2.19}$$

in which $\tau$ is a small positive parameter (ideally, $0 < \tau \ll 1$). If the spatial derivatives in this equation are discretised by second-order central differences on the grid (2.8) we obtain the semi-discrete system of moving mesh equations defined by

$$\dot{x}_{i-1} - 2\dot{x}_i + \dot{x}_{i+1} = -\frac{1}{\tau} \left[ \tilde{M}_{i+1/2}(x_{i+1} - x_i) - \tilde{M}_{i-1/2}(x_i - x_{i-1}) \right], \tag{2.20}$$

for $i = 1, 2, ..., n - 1$, with

$$x_0(t) = x_L \qquad \text{and} \qquad x_n(t) = x_R. \tag{2.21}$$

In Eq. (2.20), $\tilde{M}_{i+1/2}$ is a smoothed monitor function defined as in [11, 15] by

$$\tilde{M}_{i+1/2} = \frac{\sum_{k=i-p}^{i+p} M_{k+1/2} (q/(q+1))^{|k-i|}}{\sum_{k=i-p}^{i+p} (q/(q+1))^{|k-i|}}, \tag{2.22}$$

where

$$M_{i+1/2} = \sqrt{1 + \alpha^2 \left( \frac{u_{i+1} - u_i}{x_{i+1} - x_i} \right)^2}, \tag{2.23}$$

$q$ is a positive real number, and $p$ is a non-negative integer. In all computations involving the smoothing process (2.22) we set the parameter $q$ to the value 2.

The data required for the time-dependent coordinate transformation may be generated by a numerical integration of the moving mesh equations (2.13) and (2.20). We performed numerical experiments using these equations, with various values selected for the parameter $\tau$ in (2.20). The computations indicated that accuracy improves as $\tau$ decreases, but the differential system becomes very stiff when $\tau \ll 1$, and there is a corresponding increase in CPU time. Computational efficiency is improved, with no diminution in accuracy, if (2.20) is replaced by the algebraic equation

$$\tilde{M}_{i+1/2}(x_{i+1} - x_i) - \tilde{M}_{i-1/2}(x_i - x_{i-1}) = 0. \tag{2.24}$$

Apart from preliminary experiments with (2.20), all coarse mesh generation presented in this paper is effected using the DAE consisting of (2.13) and (2.24), with boundary conditions (2.14) and (2.21). To provide initial conditions for the time integration we obtain $\{x_i\}_{i=1}^{n-1}$ by solving the steady equidistribution problem (2.24), with $u_i \equiv u_0(x_i)$ in the monitor function. The locations of the equidistributed nodes, together with $u_i(0) = u_0(x_i)$, for $i = 1, 2, ..., n - 1$, serve as initial conditions.

Equations (2.13) and (2.24) are integrated on a fairly coarse mesh using a first-order backward Euler method for (2.13) (although low in accuracy, it is stable for stiff systems). At each integration step the nonlinear system of $2(n - 1)$ algebraic equations is solved by a Newton iteration with exact Jacobian. Continuation in $\alpha$ and $\varepsilon$, as described for steady problems in Mulholland *et al.* [15] is used at time zero, but $\alpha$ and $\varepsilon$ are fixed at all subsequent time steps. When the finite difference solution has been computed at time $t$, a smooth map

$$x = x(\xi, t) \tag{2.25}$$

is constructed by fitting a polynomial of degree $m$ to the values $\{\bar{x}_i\}_{i=0}^m$ that approximate $x$ at the Chebyshev nodes

$$\bar{\xi}_i = -\cos\frac{\pi i}{m}, \qquad i = 0, 1, ..., m. \tag{2.26}$$

The values $\{\bar{x}_i\}_{i=0}^m$ are obtained by linear interpolation on the solution $\{x_j\}_{j=0}^n$ computed on the evenly spaced grid (2.8) at time $t$. Details concerning the construction of (2.25) are given in Mulholland *et al.* [15]. Here we need only present sufficient information to define parameters that have to be specified in the numerical computations. To this end, we note that the smooth coordinate map (2.25) is represented by

$$x = P_m(\xi, t) = \sum_{k=0}^{m} \sigma_k a_k(t) T_k(\xi), \tag{2.27}$$

where $a_k$ is a time-dependent coefficient, $T_k$ is the Chebyshev polynomial of the first kind of degree $k$, and $\sigma_k$ is a filter function given by

$$\sigma_k = \exp[-32(k/m)^\gamma]. \tag{2.28}$$

The smoothing parameter $\gamma$ has to be specified.

Assuming the existence of the time-dependent map (2.25) relating $x$ to a coordinate $\xi \in D_c = [-1, 1]$ we may recast the differential problem (2.1)–(2.3) as a problem with dependent variable $v(\xi, t) = u(x(\xi, t), t)$ and independent variables $\xi$ and $t$. The transformed system is

$$\begin{aligned} &\dot{v} - \frac{\dot{x}}{x_\xi}\frac{\partial v}{\partial \xi} + \frac{v}{x_\xi}\frac{\partial v}{\partial \xi} = \frac{\varepsilon}{(x_\xi)^3}\left[x_\xi \frac{\partial^2 v}{\partial \xi^2} - x_{\xi\xi}\frac{\partial v}{\partial \xi}\right], \\ &\xi \in [-1, 1], t > 0, \end{aligned} \tag{2.29}$$

where $\dot{v} \equiv \partial v/\partial t$ and $\dot{x} \equiv \partial x/\partial t$ denote derivatives in which $\xi$ is held constant, and $x_\xi \equiv \partial x/\partial \xi$. The initial and boundary conditions for (2.29) are

$$v(\xi, 0) = u_0(x(\xi, 0)), \qquad -1 \le \xi \le 1, \tag{2.30}$$

$$v(-1, t) = b_L(t), \qquad v(1, t) = b_R(t), t \ge 0. \tag{2.31}$$

The problem (2.29)–(2.31) is solved for $v$ using standard PS discretisation in space based on nodes

$$\xi_i = -\cos\frac{\pi i}{N}, \qquad i = 0, 1, ..., N, \tag{2.32}$$

and a stiffly stable integrator in time such as Crank–Nicolson or BDF. In this algorithm, the PS method provides an accurate solution at the time values $t_0, t_1, ...$ that are used in the moving mesh finite difference method, and the moving mesh solution is carried to a typical time level $t_{s+1}$ in preparation for the PS solution at this time. This means that the coefficients $a_k(t)$ in the map (2.27) are available at $t_s$ and $t_{s+1}$, which provides $\{x(\xi_i, t_\nu) = x_i^\nu\}_{i=0}^N$ for $\nu = s, s+1$. Furthermore, in solving (2.29)–(2.31) for $\{v(\xi_i, t_{s+1}) = v_i^{s+1}\}_{i=1}^{N-1}$ the approximation $\{v_i^s\}_{i=0}^N$ is known. Approximations to derivatives with respect to $\xi$ in (2.29) are represented at node $\xi_i$ as summations of the form

$$(g_\xi)_i = \sum_{j=0}^{N} D_{ij}^{(1)} g_j, \qquad (g_{\xi\xi})_i = \sum_{j=0}^{N} D_{ij}^{(2)} g_j, \tag{2.33}$$

where $D^{(1)}$ and $D^{(2)}$ are, respectively, the first-order and second-order PS differentiation matrices (see [4, 7]), and $g$ denotes $x^s$, $x^{s+1}$, $v^s$, or $v^{s+1}$. A discretisation of (2.29) by Crank–Nicolson is

$$\begin{aligned} &\frac{v_i^{s+1} - v_i^s}{\Delta t} - \frac{x_i^{s+1} - x_i^s}{\Delta t}\frac{1}{2}\left[\left(\frac{v_\xi}{x_\xi}\right)_i^{s+1} + \left(\frac{v_\xi}{x_\xi}\right)_i^{s}\right] \\ &\qquad + \frac{1}{4}\left\{(v_i^{s+1} + v_i^s)\left[\left(\frac{v_\xi}{x_\xi}\right)_i^{s+1} + \left(\frac{v_\xi}{x_\xi}\right)_i^{s}\right]\right\} \\ &\qquad = \frac{\varepsilon}{2}\left[\left(\frac{v_{\xi\xi}}{x_\xi^2}\right)_i^{s+1} + \left(\frac{v_{\xi\xi}}{x_\xi^2}\right)_i^{s} - \left(\frac{x_{\xi\xi}v_\xi}{x_\xi^3}\right)_i^{s+1} - \left(\frac{x_{\xi\xi}v_\xi}{x_\xi^3}\right)_i^{s}\right], \end{aligned} \tag{2.34}$$

for $i = 1, 2, ..., N-1$ and $s = 0, 1, ....$

$\{v_i^0\}_{i=0}^N$ is given by (2.30), and $v_0^{s+1}$ and $v_N^{s+1}$ are given by (2.31). The nonlinear system (2.34) is solved by a Newton iteration with initial estimate at step $s+1$ provided by the solution at step $s$. BDF time discretisation is performed in an analogous manner.

The key steps in evolving the solution from time $t_s$ to time $t_{s+1}$ by the discrete time algorithm are summarised below.

Discrete Time Algorithm. Select initial and maximum time steps, $\Delta t$ and $\Delta t_{\max}$; discretisation integers $n$ and $N$, and integers $p$ and $m$ for smoothing (2.22) and for the map (2.27); real parameters $\alpha$ for (2.17), $\gamma$ for (2.28), and tolerances FDTOL, PSTOL for the Newton iterations associated, respectively, with the finite difference system (2.13), (2.24) and with the PS system (2.34).

(1) Form initial conditions $\{x_i^0, u_i^0\}_{i=0}^n$ for the coarse grid solution via (2.24). Form the map (2.25) at $t = t_0 = 0$, hence $\{x_i^0\}_{i=0}^N$, and initial conditions $\{x_i^0, v_i^0\}_{i=0}^N$ via (2.30). $s := 0$.

(2) Obtain $\{x_i^{s+1}, u_i^{s+1}\}_{i=0}^n$ from DAE (2.13) and (2.24).

(3) Form the map $x = x(\xi, t_{s+1})$ and hence $\{x_i^{s+1}\}_{i=0}^N$.

(4) Obtain the PS solution $\{v_i^{s+1}\}_{i=0}^N$ by Crank–Nicolson or BDF solution of (2.34).

**TABLE I**

$L_\infty$ Errors for MMPDE and DAE Solutions of Problem II

| $\tau$ | $L_\infty$ error | Normalised CPU |
|---|---|---|
| 1.0 | $1.89 \times 10^{-1}$ | 1.07 |
| $10^{-5}$ | $1.75 \times 10^{-2}$ | 1.00 |
| $10^{-10}$ | $1.54 \times 10^{-2}$ | 2.88 |
| DAE ($\tau = 0$) | $1.47 \times 10^{-2}$ | 0.86 |

*Note.* $\varepsilon = 10^{-3}$, using $n = 30$, $\alpha = 2$, $p = 3$, and $0 < t \leq 0.5$. First-order BDF employed with initial $\Delta t = \Delta t_{\max} = 0.05$.

(5) $s := s + 1$. Go to 2.

The moving mesh solution of (2.13) and (2.24) uses a very simple time-step control. In terms of the maximum norm, if consecutive iterates fail to agree to within FDTOL after three Newton cycles the time-step is halved and the calculation repeated. Rapid Newton convergence at any step leads to an increase in $\Delta t$ prior to the next step, provided the revised $\Delta t$ does not exceed $\Delta t_{\max}$. PSTOL is used in an analogous manner: if a PS iteration has failed to converge after three Newton cycles the time-step is halved and the FD calculation is repeated. All computations performed using the discrete time algorithm employed the values FDTOL = $10^{-5}$ and PSTOL = $10^{-8}$.

2.3. *Numerical Results for the Discrete Time Algorithm* Initial numerical experiments compared the MMPDE method of Huang, Ren, and Russell [8] with the DAE method in terms of accuracy and computational efficiency. Results are presented for the integration of Problem II over the time interval $0 < t \leq 0.5$ using (2.13) and (2.20) with several values of $\tau$, and also using the DAE (2.13) and (2.24). In the computations we solved the moving mesh finite difference equations for the parameter set $\{\varepsilon = 10^{-3}$, $n = 30$, $\alpha = 2$, $p = 3\}$, with initial $\Delta t$ and $\Delta t_{\max}$ set at 0.05.

Table I shows the maximum pointwise error throughout the specified time interval for several values of $\tau$ and the CPU time normalised by the value at $\tau = 10^{-5}$. For the MMPDE we found high accuracy and relatively low computational cost for $\tau = O(10^{-q})$, with $3 \leq q \leq 7$. In this region the $L_\infty$ error is fairly insensitive to the value of $\tau$. For $\tau$ values of order unity the mesh movement is slow relative to the movement of solution features, and this can give rise to inaccuracies. For values of $\tau$ smaller than $10^{-7}$ the problem becomes very stiff and the CPU time increases substantially. The DAE approach was found to be as accurate as the MMPDE method and the cost was marginally less. Furthermore, this method does not require a choice to be made for $\tau$ and it is very robust. The comparative numerical experiments led us to use the DAE approach for all subsequent calculations.

**TABLE II**

$L_\infty$ Errors for PS Solutions of Problem II

| $N$ | $\Delta t$ | $L_\infty$ error | Normalised CPU |
|---|---|---|---|
| 64 | 0.05 | $4.61 \times 10^{-3}$ | 1.00 |
| 96 | 0.05 | $4.07 \times 10^{-3}$ | 1.92 |
| 64 | 0.01 | $3.76 \times 10^{-4}$ | 2.58 |

*Note.* $\varepsilon = 10^{-3}$, using $n = 30$, $\alpha = 2$, $p = 3$ in (2.13) and (2.24), and $m = 64$, $\gamma = 2$ in (2.34). Integration of (2.34) over $0 < t \leq 0.5$ by Crank–Nicolson.

Table II shows results for Problem II with PS post-processing combined with Crank–Nicolson time integration. Here we used a fixed value of $\Delta t$ to integrate over the time interval $0 < t \leq 0.5$, and the aim was to gain some insight into the key source of the computational error. Results for $N = 64$ and $N = 96$ with a fixed $\Delta t$ show no change in $L_\infty$ error, and this suggests that at $N = 64$ the error is dominated by time-step error. To support this view we have presented the $L_\infty$ error for $N = 64$ and a reduced $\Delta t$: the reduction in computational error indicates that much might be gained by employing a more accurate time integrator. CPU times are normalised relative to the time for $N = 64$ and $\Delta t = 0.05$.

In Table III we have displayed the $L_\infty$ errors obtained by PS post-processing carried out using BDF methods of orders up to 5. Here we see that high accuracy can be achieved with methods of order 4 or 5: furthermore, the CPU time is virtually independent of the order of the BDF method employed. This set of experiments led us to adopt a suite of BDF methods with orders 1 to 5 for all subsequent integrations of Eq. (2.34) using the discrete time algorithm. This is linked, of course, with the time-step control outlined in subsection 2.2.

In Table IV we have compared our moving mesh finite difference solution to Problem I with results obtained by Blom *et al.* [3] using their first-order implicit-Euler Lagrangian (IEL) scheme. As in [3], a fixed value of $\Delta t$ was used throughout the integration. Note that the DAE results are

**TABLE III**

$L_\infty$ Errors for PS Solutions of Problem II

| $K$ | $L_\infty$ error |
|---|---|
| 2 | $2.91 \times 10^{-4}$ |
| 3 | $2.41 \times 10^{-4}$ |
| 4 | $8.00 \times 10^{-5}$ |
| 5 | $3.56 \times 10^{-5}$ |

*Note.* $\varepsilon = 10^{-3}$, using $n = 30$, $\alpha = 2$, $p = 3$ in (2.13) and (2.24), and $m = 64$, $\gamma = 2$, and $N = 64$ in (2.34). Integration of (2.34) by Kth-order BDF over time interval $0 < t \leq 0.5$ with initial $\Delta t = 0.001$ and $\Delta t_{\max} = 0.01$.

**TABLE IV**

$L_\infty$ Errors for Problem I

| | $t = 1.0$ | | | $t = 1.5$ | | |
|---|---|---|---|---|---|---|
| | $n = 20$ | $n = 40$ | $n = 80$ | $n = 20$ | $n = 40$ | $n = 80$ |
| Method in [3] | $1.60 \times 10^{-1}$ | $9.99 \times 10^{-2}$ | $3.53 \times 10^{-2}$ | $9.85 \times 10^{-2}$ | $1.63 \times 10^{-1}$ | $1.50 \times 10^{-1}$ |
| DAE | $1.15 \times 10^{-2}$ | $3.23 \times 10^{-3}$ | $9.26 \times 10^{-4}$ | $2.65 \times 10^{-2}$ | $2.54 \times 10^{-2}$ | $2.37 \times 10^{-2}$ |

*Note.* $\varepsilon = 10^{-3}$, using $\alpha = 2$, $p = 3$ in (2.13) and (2.24). First-order BDF employed with fixed $\Delta t = 1/40$.

better by an order of magnitude at $t = 1.0$ and marginally better at $t = 1.5$. The computed solution to this problem is less accurate around $t = 1.5$ where the moving front is close to the boundary at $x = 1$: we shall return to this difficulty later in subsection 4.2.

Table V presents a comparison between the PS solution of Problem I and the solution obtained in [3] using a grid with $n = 80$ and a fixed time-step $\Delta t = 1/80$. Our coarse grid solution was obtained using the same fixed time-step with $n = 30$, and the PS solution used $N = 64$. The table shows that our PS method gives an improvement of 3 orders of magnitude at $t = 1.0$, but only an improvement of between 1 and 2 orders of magnitude at the more difficult time of $t = 1.5$. In the interval $0 < t \leq 1.0$ the maximum $L_\infty$ error for the PS method is $1.63 \times 10^{-5}$ at $t = 0.2875$. The CPU times are not presented in [3]; however, it is of interest to note that for integration up to $t = 1.0$, the CPU time for our moving mesh FD method with $n = 80$ exceeds the CPU time for the PS method (with data as in Table V) by a factor of 4.

Figure 1 displays the computed solution to Problem I using the discrete time algorithm with initial $\Delta t = 0.001$, $\Delta t_{\max} = 1/80$, and all other input parameters as presented in Table V. Throughout the paper, the orientation of the axes is selected to give the best view. In Fig. 1(a) we see the solution plotted as a function of time, $t$, and the computational coordinate, $\xi$, in the interval $0 < t \leq 1.0$. The display clearly illustrates the smoothness of the solution in the transformed coordinate system. The steep front is evident in Fig. 1(b) where the solution is shown in terms of $x$ and $t$. The computational error is shown in terms of $t$ and $\xi$ in Fig. 1(c) and the map $x = x(\xi, t)$ is given in Fig. 1(d). The smooth evolution of the coarse grid nodes $\{x_i(t)\}_{i=0}^n$ is shown in Fig. 1(e) and in Fig. 1(f) we show the computed PS solution at $t = 1.5$. This display shows clearly that the error becomes very large toward the end of the time interval [0, 1.5] as the front reaches the boundary at $x = 1$. In this period of time there is rapid mesh movement and the time-step control does not reduce $\Delta t$ sufficiently fast. This problem is overcome in Subsection 4.2 using a more sophisticated time-step control.

**TABLE V**

$L_\infty$ Errors for Problem I

| | $t = 1.0$ | $t = 1.5$ |
|---|---|---|
| Method in [3] | $4.06 \times 10^{-2}$ | $8.04 \times 10^{-2}$ |
| adaptive PS | $1.19 \times 10^{-5}$ | $2.57 \times 10^{-3}$ |

*Note.* $\varepsilon = 10^{-3}$, using $n = 30$, $\alpha = 2$, $p = 3$ in (2.13) and (2.24), and $m = 64$, $\gamma = 2$ and $N = 64$ in (2.34). Integration of (2.34) by fifth-order BDF. First-order BDF employed with $n = 80$ and $\Delta t = 1/80$ in [3].

Our moving mesh finite difference solution to Problem II is compared in Table VI with the IEL results for this problem given in [3]. Input data are similar to those given in Table IV with results presented at $t = 0.25$ and $t = 1.0$. For this problem the DAE results are better by an order of magnitude at both values of time. A comparison between IEL results and our PS solution is given in Table VII. As in Table V, in Table VII the IEL results were obtained with $n = 80$ and a fixed time-step $\Delta t = 1/80$. Our coarse grid solution used the same time-step, with $n = 30$, and the PS solution used $N = 64$. In this case, the PS method gives an improvement of 3 orders of magnitude at both output times. In the interval $0 < t \leq 1.0$ the maximum $L_\infty$ error for the PS method is $4.38 \times 10^{-5}$ at $t = 0.65$.

Figure 2 displays the computed solution to Problem II using the discrete time algorithm with initial $\Delta t = 0.001$, $\Delta t_{\max} = 1/80$, and all other input parameters as presented in Table VII. Figures 2(a) and 2(b) show that small time steps are used by the adaptive algorithm in the initial phase and also around $t = 0.5$, when the two fronts are merging. Note also in Fig. 2(c) that despite the time-step reduction near $t = 0.5$ the computational error is greatest in this region. The smoothness of the particle paths is evident in Fig. 2(e), and the merging of the fronts is clearly shown in this display.

Problem III is included to enable us to identify a source of difficulty for the adaptive algorithm. As the front enters the domain at $x = 0$ and leaves at $x = 2\pi$ we noticed that errors can be large immediately before the action ceases at $x = 2\pi$. With $\varepsilon = 10^{-3}$ there is a strong front at $x = 0$ and a weak front at $x = 2\pi$ at time $t = 0.002$, and the algorithm yields low-accuracy results at this time. To gain some insight into the source of the problem we have used

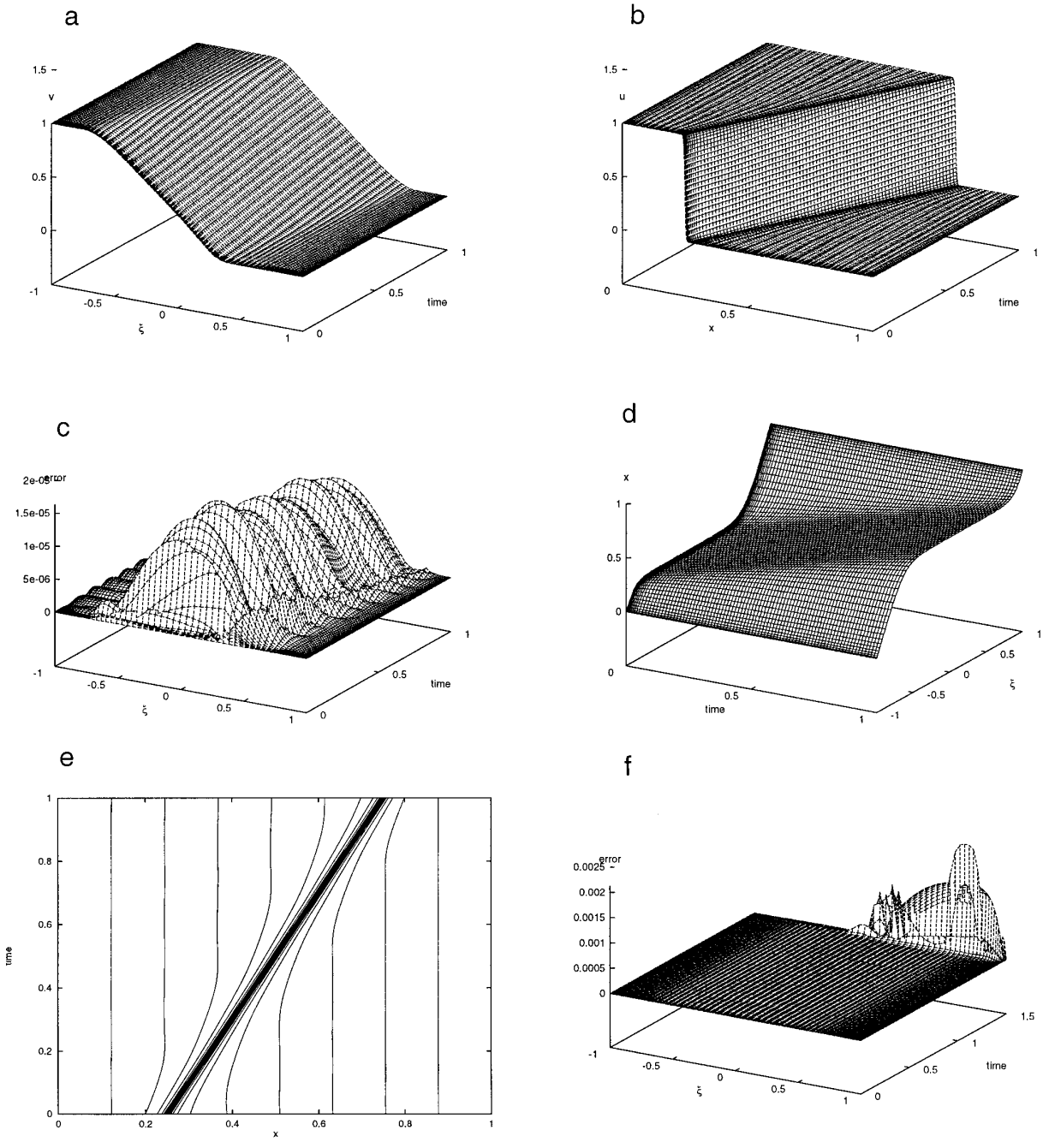


**FIG. 1.** Computed solution for Problem I over the interval $0 < t \leq 1.0$ with input data as in Table V. Parts (a) and (b) show $v(\xi, t)$ and $u(x, t)$. Computational error is in (c), map $x = x(\xi, t)$ is in (d), and coarse grid nodal paths are in (e). The large computational error as $t$ approaches 1.5 is shown in (f).

equidistribution to fit nodes to the exact solution at $t = 0.002$. Figure 3(a) shows the locations of the "equidistributed" nodes for $n = 30$, $\alpha = 2$, and $p = 3$. The unbroken line is simply a piecewise linear connection of adjacent nodes. The equidistribution fails completely to place nodes in the region of weak action near $x = 2\pi$, whereas the front near $x = 0$ appears to present no problems. The failure is related to the cut-off imposed on the smoothing process at the boundaries. By this we mean that the summations in (2.22) only include values of $k$ in the range 0, 1, ..., $n - 1$: if $k$ falls outside this range the corresponding terms in numerator and denominator are omitted.

**TABLE VI**

$L_\infty$ Errors for Problem II

| | $t = 0.25$ | | | $t = 1.0$ | | |
|---|---|---|---|---|---|---|
| | $n = 20$ | $n = 40$ | $n = 80$ | $n = 20$ | $n = 40$ | $n = 80$ |
| Method in [3] | $8.99 \times 10^{-2}$ | $1.18 \times 10^{-1}$ | $5.76 \times 10^{-2}$ | $1.21 \times 10^{-1}$ | $6.01 \times 10^{-1}$ | $1.48 \times 10^{-1}$ |
| DAE | $6.26 \times 10^{-2}$ | $5.91 \times 10^{-3}$ | $1.30 \times 10^{-2}$ | $3.99 \times 10^{-2}$ | $7.39 \times 10^{-2}$ | $3.07 \times 10^{-2}$ |

*Note.* $\varepsilon = 10^{-3}$, using $\alpha = 2$, $p = 3$ in (2.13) and (2.24). First-order BDF employed with fixed $\Delta t = 0.05$.

This one-sided smoothing is adopted in [11, 15]: in general, it presents no difficulties, but the solution of Problem III at $t = 0.002$ illustrates a weakness in the technique.

To show that proper smoothing near $x = 2\pi$ will increase the accuracy we used the periodicity of the solution to give values of $M_{k+1/2}$ in (2.22) for $k < 0$ and $k > n - 1$. This provided us with symmetric smoothing in each interior interval. The locations of the equidistributed nodes given by this modified smoothing technique are shown in Fig. 3(b). In this case there are nodes in the region of action near the right boundary. Figures 3(c) and 3(d) show the evolution of the coarse grid nodes during the time interval $0 < t \leq 0.002$ obtained by fitting nodes to the exact solution by equidistribution at a sample of points in the time interval. In Fig. 3(c) we see nodes moving away from $x = 2\pi$, whereas Fig. 3(d) indicates that the symmetric smoothing holds nodes close to the boundary until the front has moved completely into the region $x > 2\pi$. The improvement brought about by symmetric smoothing at the boundaries identifies a possible source of difficulty: unfortunately, for non-periodic problems it does not provide us with a solution. Further work is needed in improving the smoothing process at the boundary for general problems.

## 3. INVESTIGATION OF COARSE GRID SMOOTHNESS

The aim of this section is to examine the quality of the mesh generated by the finite difference solution of the DAE (2.13) and (2.24). It is important to generate a mesh that is free from abrupt variations, and this is the purpose of the smoothing process described by (2.22). The spatially smoothed mesh possesses the essential property of no node-crossing, and we demonstrate that it is likely to have a good measure of smoothness by virtue of its local quasi-uniformity property [10].

Define unsmoothed and smoothed monitor function vectors by

$$\left.\begin{aligned} \mathbf{M} &= (M_{0+1/2}, M_{1+1/2}, ..., M_{n-1+1/2})^T \\ \tilde{\mathbf{M}} &= (\tilde{M}_{0+1/2}, \tilde{M}_{1+1/2}, ..., \tilde{M}_{n-1+1/2})^T \end{aligned}\right\} \tag{3.1}$$

and a smoothing matrix, $D^{-1}G$, by

$$D\tilde{\mathbf{M}} = G\mathbf{M}, \tag{3.2}$$

where $D \in \mathbf{R}^{n\times n}$ is a diagonal matrix, and, for $i, j = 0, 1, ..., n - 1$,

$$[G]_{i,j} = g_{i,j} = \begin{cases} \nu^{|i-j|} & \text{if } |i-j| \leq p, \\ 0 & \text{otherwise.} \end{cases}$$

Here $\nu = q/(q + 1)$, with $q$ defined in (2.22), and $D$ has diagonal elements $D_{0+1/2}, D_{1+1/2}, ..., D_{n-1+1/2}$, with

$$D_{i+1/2} = \sum_{j=0}^{n-1} g_{i,j}. \tag{3.3}$$

It is clear from the definitions of $G$ and $D$ that $\|D^{-1}G\|_\infty = 1$, and it follows from (3.2) that

$$\|\tilde{\mathbf{M}}\|_\infty \leq \|\mathbf{M}\|_\infty. \tag{3.4}$$

A quasi-uniformity result is readily obtained for the special case of smoothing with $p = n - 1$. For this choice of $p$ the matrix $G$ is full, and the smoothing process (2.22) makes use of the monitor function values in all grid spaces. For $i = 0, 1, ..., n - 1$ we have

$$D_{i+1/2}\tilde{M}_{i+1/2} = \nu^i M_{1/2} + \nu^{i-1} M_{3/2} + ... + \nu M_{i-1/2} + M_{i+1/2}$$

**TABLE VII**

$L_\infty$ Errors for Problem II

| | $t = 0.25$ | $t = 1.0$ |
|---|---|---|
| Method in [3] | $8.30 \times 10^{-3}$ | $4.00 \times 10^{-3}$ |
| adaptive PS | $9.02 \times 10^{-6}$ | $9.08 \times 10^{-6}$ |

*Note.* $\varepsilon, n, \alpha$ and $p$ as in Table VI, and $m = 64, \gamma = 2$, and $N = 64$ in (2.34). Integration of (2.34) by fifth-order BDF. First-order BDF employed with $n = 80$ and $\Delta t = 1/80$ in [3].

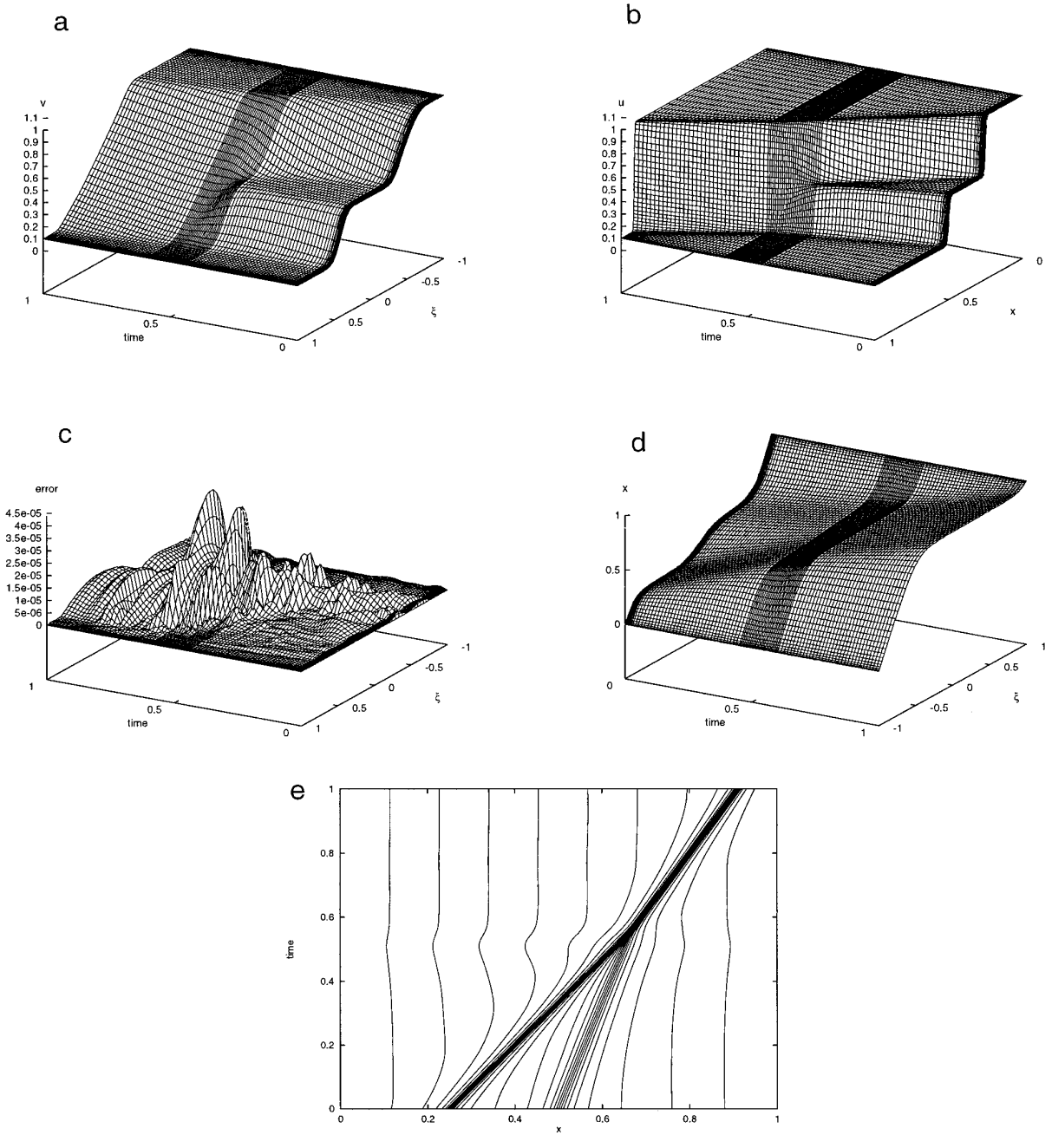


**FIG. 2.** Computed solution for Problem II over the interval $0 < t \leq 1.0$ with input data as in Table VII. Parts (a) and (b) show $v(\xi, t)$ and $u(x, t)$. Computational error is in (c), map $x = x(\xi, t)$ is in (d), and coarse grid nodal paths are in (e).

$$+ \nu M_{i+3/2} + \ldots + \nu^{n-i-1} M_{n-1/2}$$

and

$$D_{i-1/2}\tilde{M}_{i-1/2} = \nu^{i-1} M_{1/2} + \nu^{i-2} M_{3/2} + \ldots + M_{i-1/2} + \nu M_{i+1/2} + \nu^2 M_{i+3/2} + \ldots + \nu^{n-i} M_{n-1/2}.$$

Hence $D_{i+1/2}\tilde{M}_{i+1/2} > \nu[\nu^{i-1} M_{1/2} + \ldots + \nu M_{i-3/2} + M_{i-1/2} + \nu M_{i+1/2} + \ldots + \nu^{n-i} M_{n-1/2}]$, since $1 > \nu^2$. This inequality may be written as

$$D_{i+1/2}\tilde{M}_{i+1/2} > \nu D_{i-1/2}\tilde{M}_{i-1/2}.$$

Similarly, it may be shown that

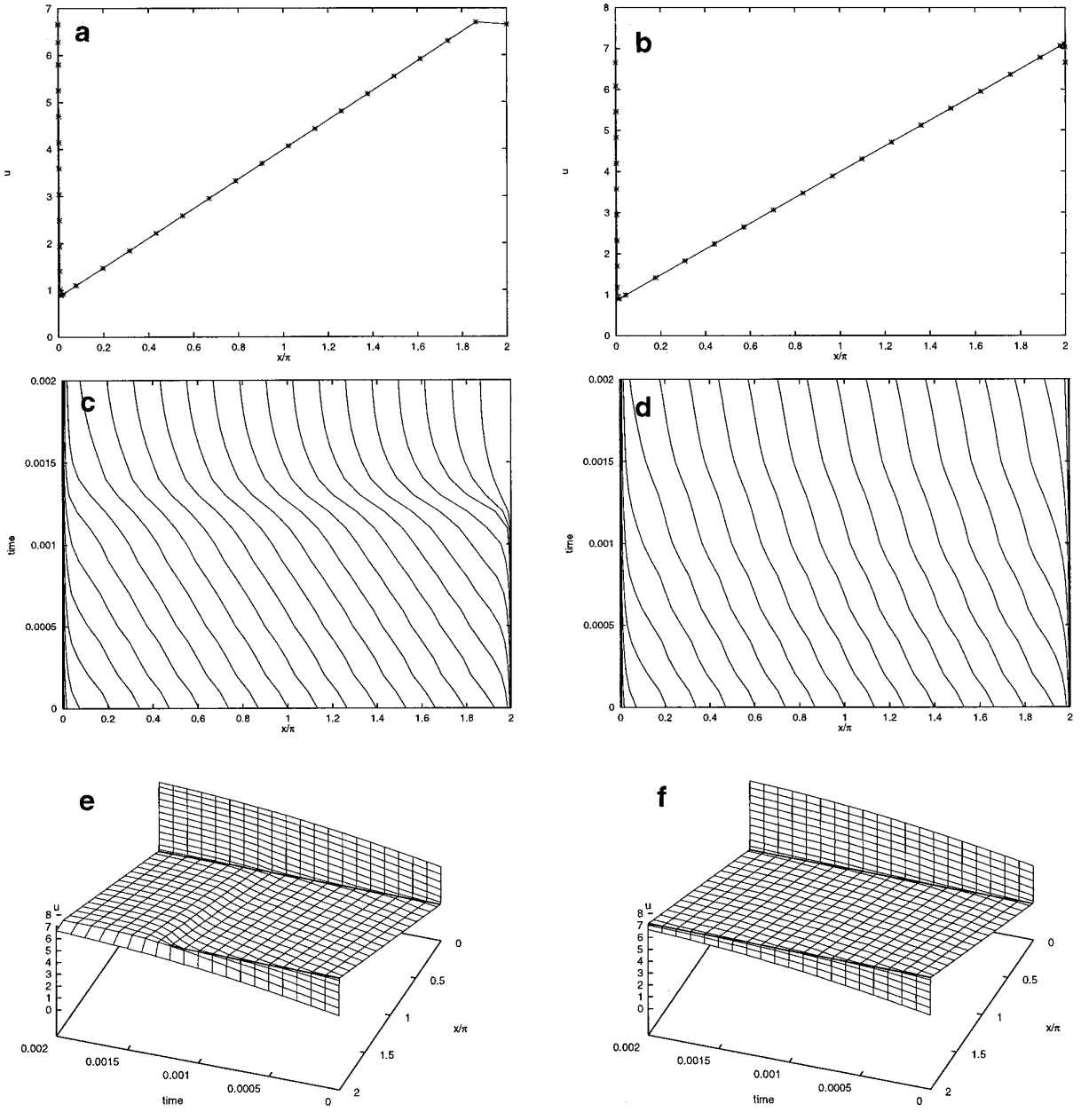


**FIG. 3.** Equidistributed nodes using the exact solution of Problem III at $t = 0.002$. Parameters are $\varepsilon = 10^{-3}$, $n = 30$, $\alpha = 2$, and $p = 3$. Parts (a) and (b) show, respectively, monitor function cut-off at the boundaries and periodicity used to remove cut-off. Parts (c) and (d) display the evolution of coarse grid nodes given by the cut-off and periodic monitor functions.

$$D_{i-1/2}\tilde{M}_{i-1/2} > \nu D_{i+1/2}\tilde{M}_{i+1/2},$$

and these conditions, together with the equidistribution equation (2.24), indicate that

$$\frac{D_{i+1/2}}{D_{i-1/2}}\,\nu < \frac{\tilde{M}_{i-1/2}}{\tilde{M}_{i+1/2}} < \frac{D_{i+1/2}}{D_{i-1/2}}\frac{1}{\nu}$$

and

$$\frac{D_{i+1/2}}{D_{i-1/2}}\,\nu < \frac{x_{i+1}-x_i}{x_i-x_{i-1}} < \frac{D_{i+1/2}}{D_{i-1/2}}\frac{1}{\nu}.$$

By summing the series (3.3) we obtain the quasi-uniformity condition

$$\frac{\nu[1-\nu^{n-i}+\nu-\nu^{i+1}]}{[1-\nu^{n-i+1}+\nu-\nu^{i}]} < \frac{x_{i+1}-x_i}{x_i-x_{i-1}} < \frac{[1-\nu^{n-i}+\nu-\nu^{i+1}]}{\nu[1-\nu^{n-i+1}+\nu-\nu^{i}]}, \tag{3.5}$$

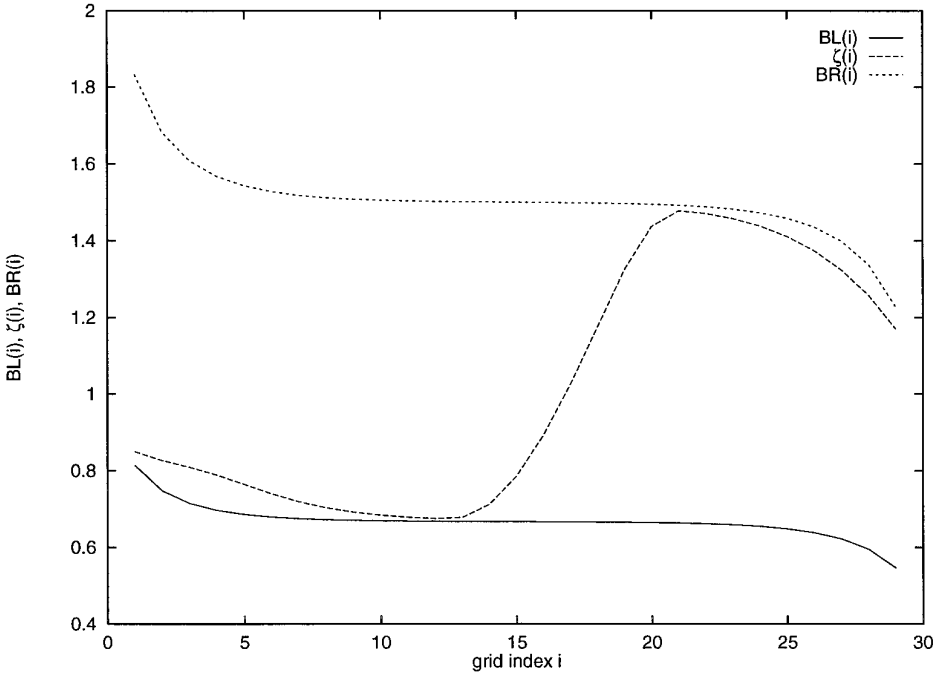


**FIG. 4.** Grid ratio and bounds in (3.5) for Problem I at $t = 1.0$ and data set $\{\varepsilon = 10^{-3}, n = 30, \alpha = 2, p = 15\}$.

for $i = 1, 2, ..., n - 1$. Since equidistribution is imposed at each time-step, we see that (3.5) will hold throughout the time evolution, provided $p = n - 1$. In practice, of course, $p$ will be much smaller than $n - 1$, and (3.5) can be taken as an approximation to the real situation.

To check the bounds in (3.5), Problem I was integrated to $t = 1.0$ using the parameter set $\{\varepsilon = 10^{-3}, n = 30, \alpha = 2\}$ and a range of values for $p$. Figure 4 shows $\zeta(i) = (x_{i+1} - x_i)/(x_i - x_{i-1})$ for $i = 1, 2, ..., 29$, together with the upper and lower bounds given by (3.5), with $p$ set equal to 15. For lower values of $p$ we find that $\zeta(i)$ is not contained within the bounds.

A result related to (3.5) has been given by Dorfi and Drury [6]. Their result,

$$\nu < \frac{x_{i+1} - x_i}{x_i - x_{i-1}} < \frac{1}{\nu}, \tag{3.6}$$

is obtained as above by moving the boundaries to $\pm\infty$ and by smoothing over the doubly infinite set of intervals. Table VIII shows minimum and maximum values of $\zeta(i)$ for $1 \le i \le n - 1$ at $t = 1.0$ in the solution of Problem I. Values of $\varepsilon$, $\alpha$, and $n$ are those used for Fig. 4 and the table gives results for a range of values of $p$. Note that when $p$ has been increased to 15 the minimum and maximum values have become contained in the interval $(2/3, 3/2) \equiv (\nu, 1/\nu)$.

To see the effectiveness of (3.6) we might consider a singular perturbation problem defined on [0, 1], with a solution having a boundary layer of thickness $O(\varepsilon)$ near $x = 1$. If we set $\nu$ to the realistic value 2/3 and assume that $x_{i+1} - x_i$ diminishes as $i$ increases, then the greatest rate of diminution with $i$ is given by the limiting case

$$x_{i+1} - x_i = \tfrac{2}{3}(x_i - x_{i-1}), \qquad i = 1, 2, ..., n - 1.$$

Using this model it is readily shown that for $\varepsilon = 10^{-3}$ the node $x_{n-1}$ is within the boundary layer provided $n$ is sufficiently large to ensure that

**TABLE VIII**

Minimum and Maximum Values of $\zeta(i)$ for Problem I at $t = 1.0$

| $p$ | $\min_{1 \le i \le n-1} \zeta(i)$ | $\max_{1 \le i \le n-1} \zeta(i)$ |
|---|---|---|
| 3 | 0.33 | 3.02 |
| 6 | 0.49 | 2.13 |
| 9 | 0.61 | 1.61 |
| 15 | 0.67 | 1.48 |

*Note.* Data as for Fig. 4.

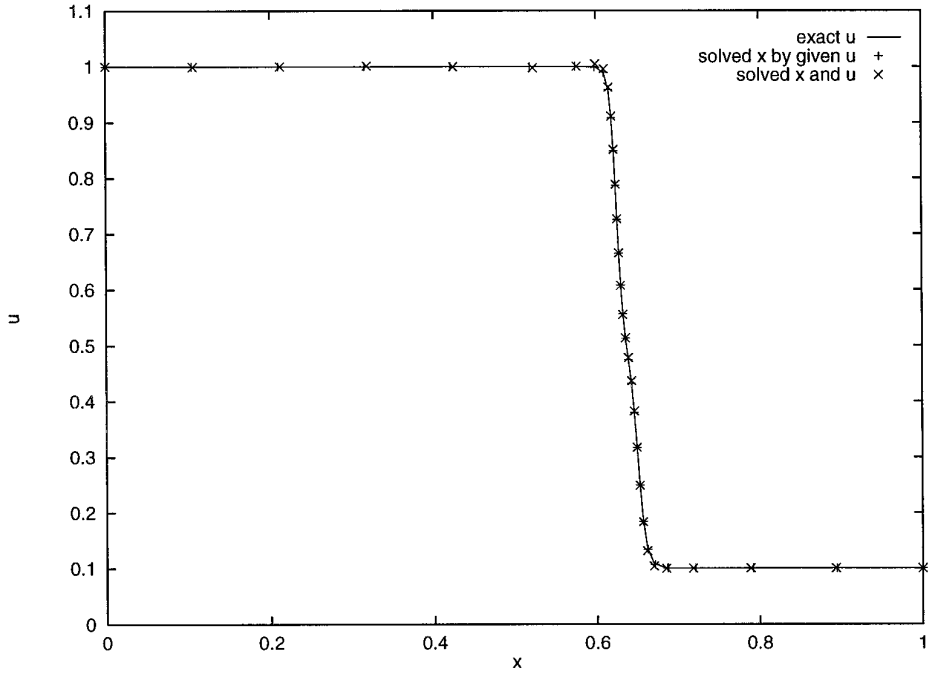


**FIG. 5.** Coarse grid for Problem II at $t = 0.5$ for the data set $\{\varepsilon = 10^{-3}, n = 30, \alpha = 2, p = 3\}$. First-order BDF used with initial $\Delta t = \Delta t_{\max} = 0.05$.

$$\left(\frac{3}{2}\right)^n > 501.$$

Since this is satisfied for $n$ greater than 15, we may assume that a grid with 30 intervals and satisfying (3.6) is capable of resolving a boundary layer of thickness $10^{-3}$. However, the grids displayed in Section 2 corresponding to adaptive solutions of problems with this degree of singularity appear to have values of $(x_{i+1} - x_i)/(x_i - x_{i-1})$ greatly in excess of $1/\nu$. The values of the ratio that are attained in computations are likely to be influenced by the solution being computed as well as by the type of smoothing adopted.

Alternative bounds on $(x_{i+1} - x_i)/(x_i - x_{i-1})$ may be obtained by taking the features of the exact solution into account when approximating $\tilde{M}_{i\pm1/2}$ in (2.24). Problem II was solved for $0 < t \leq 1.0$ using the moving mesh finite difference method, and it was found that for any value of $t$ in this interval the node locations were almost coincident with those found by applying equidistribution to the exact solution. Fitting nodes to the exact solution was based on the same parameter values. Figure 5 shows the exact solution at $t = 0.5$, with the nodes computed by the discrete time algorithm and those fitted to the exact solution superimposed. For $0 < t \leq 1.0$ the discrepancies between corresponding nodes computed by the two methods are greatest at $t = 0.5$ when the fronts merge. Computations of this type suggest that investigations into the evolution of nodes in the discrete time algorithm may be based on the assumption that the monitor function is evaluated in terms of the exact differential solution. This being the case, some features of the monitor function may be obtained from known behaviour of the exact differential solution.

The monitor function, $M_{i+1/2}$, used in (2.24) may be replaced by

$$M_{i+1/2} := M(x_{i+1/2}, t) = \sqrt{1 + \alpha^2 \left(\frac{\partial u}{\partial x}(x_{i+1/2}, t)\right)^2}, \quad \text{for } i = 0, 1, ..., n-1. \tag{3.7}$$

It is readily seen that

$$1 \leq M_{i+1/2} \leq 1 + \alpha \left|\frac{\partial u}{\partial x}(x_{i+1/2}, t)\right|.$$

Hence

$$1 \leq \|\mathbf{M}\|_\infty \leq 1 + \alpha R, \tag{3.8}$$

where $R = \sup_{0\leq x\leq 1} |\partial u/\partial x|$. For the problems considered in Section 2 the quantity $R$ is typically $O(1/\varepsilon)$ as $\varepsilon \to 0$. From (3.8) and (3.4) we have

$$1 \leq \|\tilde{\mathbf{M}}\|_\infty \leq 1 + \alpha R, \tag{3.9}$$

which yields

$$\frac{1}{K} \le \frac{\tilde{M}_{i-1/2}}{\tilde{M}_{i+1/2}} \le K, \tag{3.10}$$

where $K = 1 + \alpha R$. If (3.10) is combined with (2.24), a revised quasi-uniformity result is obtained as

$$\frac{1}{K} \le \frac{x_{i+1} - x_i}{x_i - x_{i-1}} \le K. \tag{3.11}$$

Finally, assumptions like those leading to (3.7) may be used to obtain a lower bound on $x_{i+1} - x_i$, for $i = 0, 1, ..., n - 1$. From (2.15) and (2.16) it is readily seen that

$$\int_{x(\eta_i,t)}^{x(\eta_{i+1},t)} M(s,t)\, ds = \frac{\theta(t)}{n}, \tag{3.12}$$

where

$$\theta(t) = \int_0^1 M(s,t)\, ds, \tag{3.13}$$

and $M$ is given by (2.17). A trapezoidal approximation for the left-hand-side of (3.12) gives

$$M_{i+1/2}(x_{i+1} - x_i) = \frac{\theta(t)}{n}. \tag{3.14}$$

If $\alpha \ge 1$ we note that

$$\theta(t) \le \alpha \int_0^1 \sqrt{1 + \left(\frac{\partial u}{\partial x}(x,t)\right)^2}\, dx = \alpha \tilde{\theta}(t), \tag{3.15}$$

where $\tilde{\theta}(t)$ is the total arc-length between $x = 0$ and $x = 1$ at time $t$. We may assume that $\tilde{\theta}(t)$ is bounded for $0 \le t \le T$, where $T$ is any finite value of time, and write

$$S = \sup_{0 \le t \le T} \tilde{\theta}(t).$$

Making use of (3.8) we have

$$1 \le M_{i+1/2} \le 1 + \alpha R = K,$$

and this, together with $\theta(t) \ge 1$, enables us to write

$$\frac{1}{nK} \le x_{i+1} - x_i \le \frac{\alpha S}{n}, \tag{3.16}$$

provided $\alpha \ge 1$.

The validity of the bounds in (3.16) is illustrated in Fig. 6. For Problem I we have displayed the bounds $1/nK$ and $\alpha S/n$, and the maximum and minimum values of $\eta(i) = x_{i+1} - x_i$ over the time interval $0 \le t \le 1.0$, with parameter values $\{\varepsilon = 10^{-3}, n = 30, \alpha = 2, p = 3\}$. In computing the quantity $R$ that is defined at (3.8) we used the exact solution. Note that the upper bound is sharp while the lower bound is low relative to the computed minimum.

## 4. CONTINUOUS TIME ALGORITHM

### 4.1. *Formulation of Algorithm*

The key distinction between this algorithm and the discrete time algorithm lies in the construction and subsequent use of the smooth map (2.25). The moving mesh finite difference components of the two algorithms are essentially similar. Thus, we generate the coarse mesh solution $\{x_i^s, u_i^s\}_{i=0}^n$ for $s = 0, 1, ...$ as before, by solving the DAE comprised of (2.13) and (2.24). Two aspects of the implementation details for the DAE solution differ marginally from those described in Section 2. Firstly, Eq. (2.13) is solved using a second-order BDF method, with the first-order backward Euler method applied only at the first step. Secondly, the simple time-step control is modified to incorporate a limit on mesh movement per time-step: this control makes use of an additional tolerance, VELTOL. The time-step controls that are used for the moving mesh equations are listed below.

*Time-Step Controls.* Specify a maximum time step, $\Delta t_{\max}$. Any increase in $\Delta t$ dictated by the controls must not violate $\Delta t \le \Delta t_{\max}$, where $\Delta t_{\max}$ has the same value for both FD and PS solutions.

*Coarse Grid Solution.* (i) The Newton iteration for $\{x_i^{s+1}, u_i^{s+1}\}_{i=0}^n$ is deemed not to have converged if the discrete $L_2$-norm of the updates exceeds FDTOL after 10 cycles, or if it increases in any cycle. The current $\Delta t$ is updated by means of

$$\Delta t := \Delta t \sqrt{\frac{3.5}{I}}, \tag{4.1}$$

where $I$ denotes the number of Newton iterations required for convergence at level FDTOL. In a case of failure to converge, $I$ is set to the value 12 and the step is repeated with the revised $\Delta t$.

(ii) At each time-step the condition

$$\max_i |\dot{x}(\eta_i, t_{s+1}) - \dot{x}(\eta_i, t_s)| \le \text{VELTOL} \tag{4.2}$$

is also imposed. If this condition fails then $\Delta t$ is halved and the step is repeated.

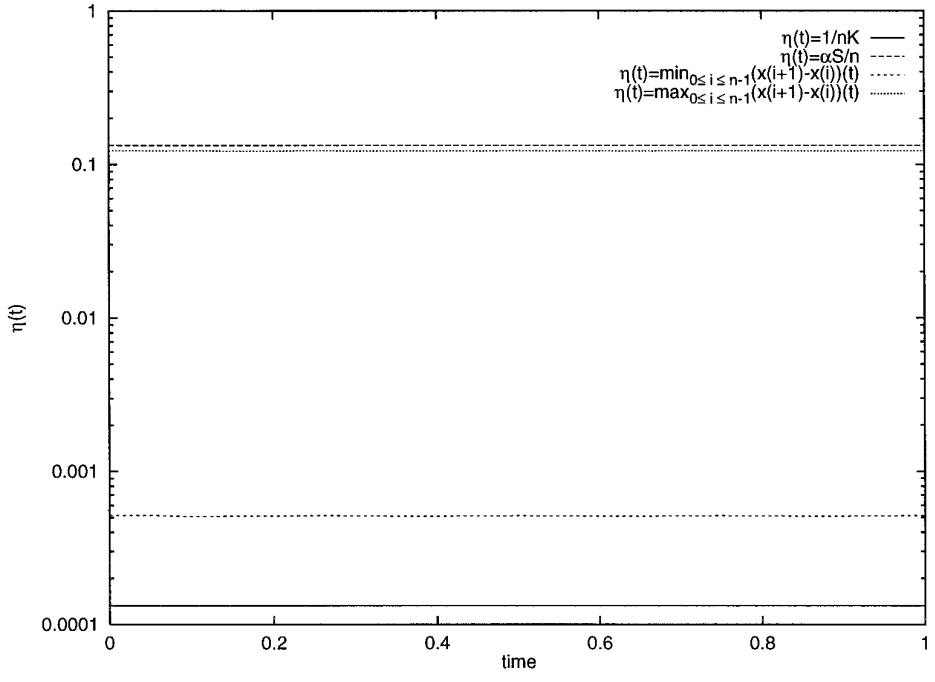


**FIG. 6.** Maximum and minimum grid spacing and bounds in (3.16) for Problem I over $0 \le t \le 1.0$ and data set $\{\varepsilon = 10^{-3}, n = 30, \alpha = 2, p = 3\}$.

An analogous procedure is adopted for the PS solution, with FDTOL replaced by PSTOL in (i) above. The control (i) is closely related to that used for the computations described in Section 2, and (ii) imposes an additional constraint.

As stated earlier, the formation of the time-dependent coordinate map (2.25) is the defining feature of the continuous time algorithm. The construction of the map is similar to that presented in Mulholland *et al.* [15] for two-dimensional steady problems. The moving mesh equations are integrated over $r$ steps to time $t_r$, producing $\{x_i^s, u_i^s\}_{i=0}^n$ for $s = 0, 1, ..., r$, with $r$ chosen such that $t_r$ is the final time at which the solution is required. We then use bilinear interpolation to obtain values $\bar{x}_k^l$ for $k = 0, 1, ..., m$ and $l = 0, 1, ..., m$, where these values approximate $x$ at the Chebyshev nodes

$$\bar{\xi}_k = -\cos\frac{\pi k}{m}, \qquad t_l = \frac{t_r}{2}\left(1 - \cos\frac{\pi l}{m}\right). \tag{4.3}$$

The transformation for $x$ in (2.25) is approximated by

$$\bar{P}(\xi, t) = \sum_{i=0}^{m}\sum_{j=0}^{m} a_{i,j} T_i(\xi) T_j\left(\frac{2t - t_r}{t_r}\right) \tag{4.4}$$

and the interpolatory conditions

$$\left.\begin{aligned} &P(\bar{\xi}_k, t_l) = \bar{x}_k^l \\ &\text{for } k, l = 0, 1, ..., m \end{aligned}\right\} \tag{4.5}$$

yield the coefficients $a_{i,j}$. The map (4.4) is smoothed using a boundary-preserving filter as described in [15], and the filter employed here has the form

$$\sigma_{i,j} = \exp\left[-32\left(\left(\frac{i}{m}\right)^{\gamma_1} + \left(\frac{j}{m}\right)^{\gamma_2}\right)\right]. \tag{4.6}$$

The two parameters $\gamma_1$ and $\gamma_2$ permit different filter strengths in space and time. As shown in [15], the smooth map $x = x(\xi, t)$ is given in terms of (4.4) by

$$x = x(\xi, t) := P(\xi, t) = (1 - \xi^2) Q(\xi, t) + \xi, \tag{4.7}$$

where

$$Q(\xi, t) = \sum_{i=0}^{m-2}\sum_{j=0}^{m} \sigma_{i,j} b_{i,j} T_i(\xi) T_j\left(\frac{2t - t_r}{t_r}\right)$$

and coefficients $b_{i,j}$ are defined recursively in terms of $\{a_{k,l}\}_{k,l=0}^m$.

Given the map (4.7), it is now possible to integrate the transformed equation (2.29) over the time interval $0 \le t \le t_r$ using standard PS discretisation in space based on the nodes (2.32). Time integration is effected by BDF of order $s$ at time-step $s$ ($s \le 5$) and BDF of order 5 thereafter. Since the map (4.7) is given as a continuous function of $t$ it is not necessary to match the time-steps to those employed for the moving mesh equations. Indeed, it is hoped

that the smoothness of the map will enable us to use time-steps that are considerably larger than those used to generate the coarse mesh.

The key steps in evolving the solution up to time $t_r$ by the continuous time algorithm are summarised below.

CONTINUOUS TIME ALGORITHM. Select initial and maximum time-steps, $\Delta t$ and $\Delta t_{\max}$; discretisation integers $n$ and $N$, and integers $p$ and $m$ for smoothing (2.22) and for the map (4.7); real parameters $\alpha$ for (2.17), $\gamma_1$ and $\gamma_2$ for (4.6), and tolerances FDTOL, PSTOL, and VELTOL for Newton iterations and for (4.2).

(1) Obtain $\{x_i^s, u_i^s\}_{i=0}^n$ for $s = 0, 1, ..., r$, by solving DAE (2.13) and (2.24) by second-order BDF (first-order at Step 1) with time-step controls as outlined earlier in this section.

(2) Re-set initial $\Delta t$ and $\Delta t_{\max}$ for PS solution. Form the map (4.7) using coarse grid data. Solve (2.29) for $\{v_i^s\}_{i=0}^N$ and $s = 0, 1, ...$ in region $0 < t_s \le t_r$, by BDF of order up to 5, with time-step controls as used in Step 1.

### 4.2. *Numerical Results for the Continuous Time Algorithm*

Given a (relatively coarse) finite difference solution of a problem, there remain two distinct stages to the continuous time algorithm. The first stage is to construct the map (4.7) using, as input, the discrete set of adaptive grids produced by the adaptive FD method. This involves setting three parameter values:

$m$: the maximum degree of polynomial (in one variable) in the transformation (4.4);

$\gamma_1$: the first (spatial) filter parameter in (4.6);

$\gamma_2$: the second (temporal) filter parameter in (4.6).

To determine the usefulness of this process of constructing the map (4.7), we must discover how sensitive solutions are to variations in parameter values and whether optimal settings are problem dependent. To assist in this determination, we perform numerous calculations over ranges of parameter values for the three test problems (for Problem III we used the spatial domain $x \in [-1, 5]$ and set $\varepsilon = 0.01$). Table IX summarises the results on parameter value validity. For each of the three problems, $m$ is set, in turn, to 64, 128, and 156; for each of these is listed the minimum and optimal values for the filter parameters $\gamma_1$ and $\gamma_2$. The minimum value represents the smallest parameter setting for which convergent solutions were obtained, while the optimal value is that for which the maximum absolute error in the PS approximation was minimised. It should be noted that $\gamma_2$ has no maximum cut-off value (with results being relatively insensitive to the value chosen for this parameter), while $\gamma_1$ does have a maximum cut-off, though optimal values tend to be close to the minimum value. Additionally, Fig. 7 shows how sensitive the solution of Problem I is to variations in the two filter parameters. In Fig. 7(a), we set $m = 156$ and $\gamma_2 = 2.0$, and vary $\gamma_1$; from this we see that there is a fairly sharp optimal value close to the minimum cut-off value and that the solution deteriorates as $\gamma_1$ is increased beyond the optimal value. Similarly, Fig. 7(b) shows, on setting $m = 156$ and $\gamma_1 = 2.0$, how the solution varies with $\gamma_2$; here we note that, although there is an optimal value for $\gamma_2$ it is nowhere near as sharp as that for $\gamma_1$ and also that the solution does not continue to deteriorate as $\gamma_2$ moves away from this optimal value. Similar patterns emerge from solutions to Problems II and III.

**TABLE IX**

Minimum and Optimal Mapping Filter Parameter Values for the Three Test Problems and for Various Values of $m$

| Problem | $m$ | $\gamma_1$(min) | $\gamma_1$(opt) | $\gamma_2$(min) | $\gamma_2$(opt) |
|---|---|---|---|---|---|
| | 64 | 2.4 | 3.0 | 1.8 | 2.0 |
| I | 128 | 2.0 | 2.2 | 1.8 | 1.8 |
| | 156 | 2.0 | 2.1 | 1.8 | 1.8 |
| | 64 | 3.0 | 3.0 | 1.5 | 2.1 |
| II | 128 | 2.0 | 2.0 | 1.5 | 1.8 |
| | 156 | 1.9 | 1.9 | 1.5 | 1.9 |
| | 64 | — | — | — | — |
| III | 128 | 2.2 | 3.0 | 1.8 | 1.9 |
| | 156 | 2.0 | 2.4 | 1.8 | 1.8 |

The results summarised in Table IX and highlighted in Fig. 7 indicate that the best overall strategy is to choose $m$ large (e.g., $m = 156$) and choose $\gamma_1 \approx 2.0$ and $\gamma_2 \approx 2.0$—this provides close to optimal performance in all three test problems.

Once a suitable mapping has been constructed, the next (final) stage is the PS approximation of the transformed problem (2.29) where values for $x$, $x_\xi$, and $x_{\xi\xi}$ are computed from the map (4.7). This involves choosing values for the parameters $N$ (from (2.32)), $\Delta t_{\max}$, and VELTOL (from (4.2)). From numerous numerical experiments it was found that a value of 0.5 for the parameter VELTOL was adequate for the three test problems; this parameter only assumes importance when the mesh changes direction suddenly. If an extremely large value is chosen for VELTOL then no time-step control is effected by (4.2), whereas a very small value for VELTOL will result in inappropriately small time-steps being taken within a region of rapid mesh movement.

It is the values chosen for $\Delta t_{\max}$ and $N$ that provide the most interest since it is these that generally control the temporal and spatial errors, respectively. Efficiency dictates that there should be a balance between the two contributions to overall error; however, to better understand

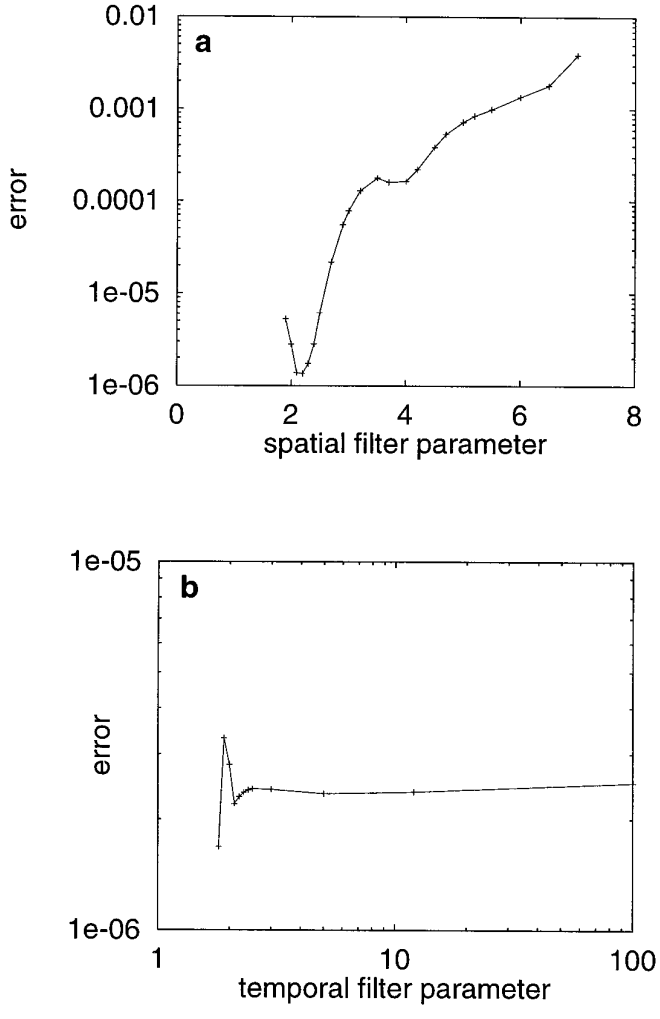


**FIG. 7.** Problem I solution sensitivity to variations in the filter parameters as measured by changes in maximum absolute errors. (a) $m = 156$, $\gamma_2 = 2.0$ and varying $\gamma_1$. (b) $m = 156$, $\gamma_1 = 2.0$ and varying $\gamma_2$.

how this balance might be achieved we must investigate separately the roles played by the two parameters by effectively eliminating (alternately) temporal and spatial contributions. Firstly, we demonstrate convergence (to zero) of the spatial error for the three test problems.

Table X shows the overall maximum absolute error in the PS approximation of Problem I with $t = 1.5$ and $\varepsilon = 0.001$. The approximation employed a map calculated using the non-optimal parameter values $m = 64$, $\gamma_1 = 3.0$, and $\gamma_2 = 2.0$. The parameter values used to effectively eliminate temporal errors were VELTOL = 0.1 and $\Delta t_{\max} = 0.0001$. The results clearly demonstrate spectral convergence of the PS approximation to the exact solution. The rapid mesh movement, evident at times close to $t = 1.5$, does not lead to a deterioration in accuracy of approximation—this is due to reductions in time-steps induced by the control (4.2) during both the FD and PS solutions of the problem. Figure 8, which plots the time-step history of the FD solution, shows that the time-step control based on the number of Newton iterations keeps the time-step at a level below the maximum for most of the evolution, but that control (4.2) considerably reduces the time-step as $t = 1.5$ is approached.

**TABLE X**

Maximum Absolute Errors in the PS Approximation of Problem I with $t = 1.5$ and $\varepsilon = 10^{-3}$

| $N$ | Maximum absolute error |
|---|---|
| 32 | $4.40 \times 10^{-4}$ |
| 48 | $1.59 \times 10^{-5}$ |
| 64 | $9.46 \times 10^{-7}$ |
| 80 | $1.48 \times 10^{-7}$ |
| 96 | $3.66 \times 10^{-8}$ |
| 112 | $6.10 \times 10^{-9}$ |
| 128 | $1.10 \times 10^{-9}$ |

Similarly Table XI shows the maximum absolute errors associated with the PS approximation of Problems II and III with $\varepsilon$ set at 0.001 and 0.01, respectively. For both problems temporal errors were effectively eliminated by setting $\Delta t_{\max} = 0.001$ and VELTOL = 0.5. The results for Problem II again demonstrate spectral convergence while those for Problem III show rapid (order $> 7$) convergence.

By effectively eliminating spatial error from our calculations we can show how the temporal error diminishes on reducing $\Delta t_{\max}$. Table XII shows the overall error associated with the PS approximation of Problem II with $\varepsilon = 0.001$ and with the fixed settings $N = 128$ and VELTOL = 0.5. The results here seem to indicate a 4th order rate of convergence to the solution despite the fact that a 5th order time-integrator was used.

Overall the results for the continuous time algorithm are promising. The difference in results between the discrete and continuous time algorithms is due to the different treatments of the $\dot{x}$ term in (2.29): the discrete algorithm approximates this term using 5th order backward differ-

**TABLE XI**

Maximum Absolute Errors in the PS Approximation of Problems II (with $\varepsilon = 10^{-3}$) and III (with $\varepsilon = 10^{-2}$)

| $N$ | Problem II maximum absolute error | Problem III maximum absolute error |
|---|---|---|
| 48 | $1.8 \times 10^{-4}$ | $5.0 \times 10^{-2}$ |
| 64 | $1.0 \times 10^{-6}$ | $3.7 \times 10^{-4}$ |
| 96 | $2.3 \times 10^{-7}$ | $2.2 \times 10^{-5}$ |
| 128 | $8.2 \times 10^{-9}$ | $2.4 \times 10^{-6}$ |

*Note.* The settings $\Delta t_{\max} = 0.001$ and VELTOL = 0.5 were fixed throughout.

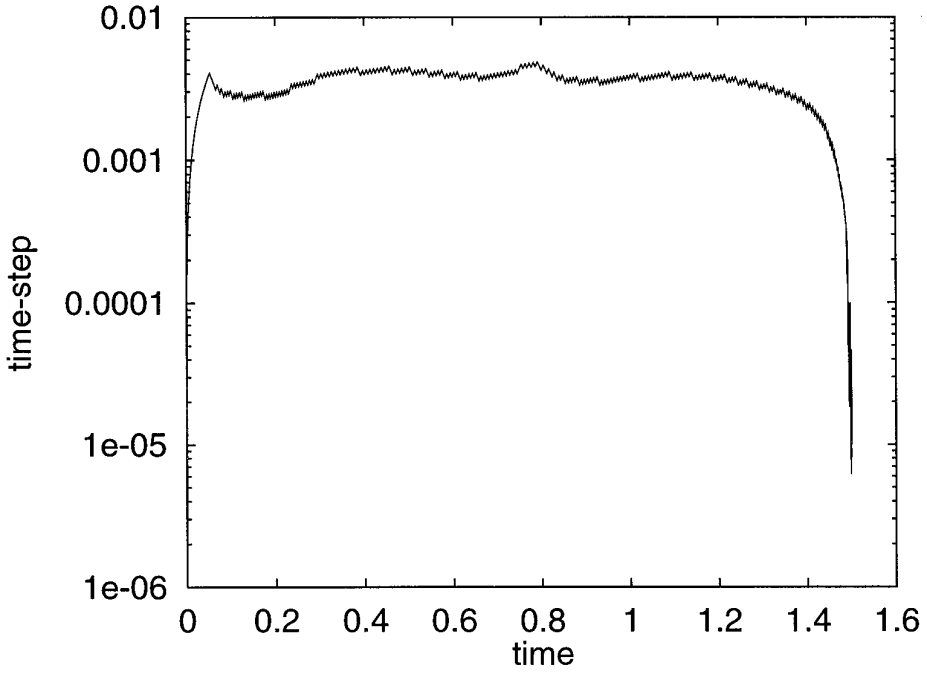


**FIG. 8.** Time-step history for the adaptive FD solution of Problem I with $\varepsilon = 10^{-3}$, $n = 32$, $\alpha = 8$, $p = 4$, $\Delta t_{\max} = 0.01$, and VELTOL = 0.5.

ences while the continuous algorithm derives an accurate value from a map computed *a priori*. Although $\dot{u}$ terms are also computed using 5th order backward differences, it is the error associated with the approximation of $\dot{x}$ that can dominate due to the amplification factor $1/x_\xi$, which can be very large at a steep front.

Figure 9 illustrates a situation in which an accurate evaluation of $\dot{x}$ is crucial. The rapid mesh movement is evident in the solution of Problem I as $t = 1.5$ is approached. It may be seen from the exact solution (2.4) that the value of $u(1, t)$ changes from approximately 0 to 0.5 as $t$ increases through an interval of width $O(\varepsilon)$ up to 1.5. The finite difference time-step control and the time smoothing in the continuous time algorithm create a less steep variation in time prior to the application of the PS post-processing. Numerical experiments indicate that the accuracy achieved by the discrete time algorithm may be improved beyond that shown in Table V if FDTOL is further reduced, but the cost becomes prohibitive at levels of accuracy well below those shown in Table X. In this type of situation the retention of a non-zero value of $\tau$ might be considered in (2.19) for the discrete time case, but the proper choice of $\tau$ remains a real problem.

**TABLE XII**

Maximum Absolute Errors in the PS Approximation of Problem II with $\varepsilon = 10^{-3}$, $N = 128$, and VELTOL = 0.5

| $\Delta t_{\max}$ | Maximum absolute error |
|---|---|
| 0.01 | $6.6 \times 10^{-5}$ |
| 0.005 | $4.0 \times 10^{-6}$ |
| 0.001 | $8.2 \times 10^{-9}$ |

## 5. CONCLUSIONS AND COMMENTS

Two algorithms are described for the solution of one-dimensional, time-dependent PDEs that have solutions involving moving fronts. The algorithms involve the coupling of a PS post-processing mechanism with a moving mesh finite difference method, and the key feature of our approach is that the FD solution can be improved by post-processing to give highly accurate results at little extra cost. Both algorithms are successful in following and resolving very sharp profiles. The high accuracy derives from the combination of PS discretisation in space with a high-order stiff time integrator and a suitable time-step control mechanism. This is particularly evident in the solution presented for Problem I at $t = 1.5$: the time-step control described in Subsection 4.1 reduces the steps appropriately when there is rapid mesh movement. Table X shows that even in the presence of this type of difficulty it is possible to gain spectral accuracy.

The implementation of the algorithms is straightforward, and they are both very robust. The results show that the approach is very promising and they demonstrate clearly that smoothing and time-step control are key issues in

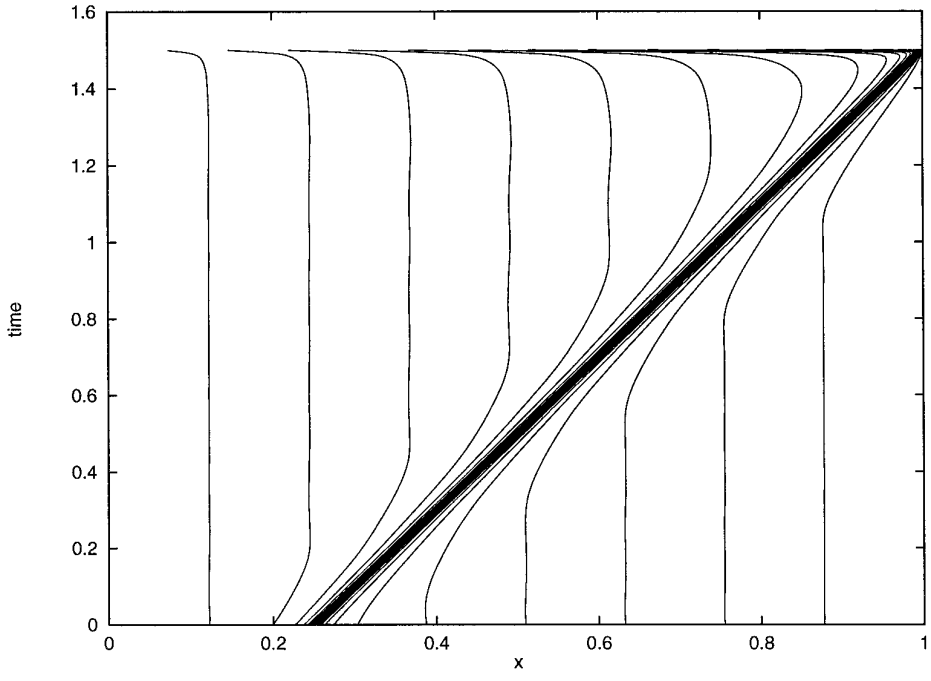


**FIG. 9.** Rapid mesh movement for Problem I as $t$ approaches 1.5.

achieving high accuracy. However, much remains to be done in identifying strengths and weaknesses. The method has to be applied to systems of equations. Currently work is in progress in extending the algorithms to deal with problems in two space dimensions.

## REFERENCES


1. A. Bayliss, A. Class, and B. J. Matkowsky, Adaptive approximation of solutions to problems with multiple layers by Chebyshev methods, *J. Comput. Phys.* **116,** 160 (1995).
2. M. Bieterman and I. Babuška, An adaptive method of lines with error control for parabolic equations of the reaction-diffusion type, *J. Comput. Phys.* **63,** 33 (1986).
3. 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 (1988).
4. C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics* (Springer-Verlag, Berlin/Heidelberg/New York, 1988).
5. 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 (1986).
6. E. A. Dorfi and L. O'C. Drury, Simple adaptive grids for 1-D initial value problems, *J. Comput. Phys.* **69,** 175 (1987).
7. B. Fornberg and D. M. Sloan, A review of PS methods for solving PDEs, *Acta Numerica,* 203 (1994).
8. W.-Z. Huang, Y. Ren, and R. D. Russell, Moving mesh partial differential equations (MMPDES) based on the equidistribution principle, *SIAM J. Numer. Anal.* **31,** 709 (1994).
9. W.-Z. Huang, Y. Ren, and R. D. Russell, Moving mesh methods based on moving mesh partial differential equations, *J. Comput. Phys.* **113,** 279 (1994).
10. W.-Z. Huang and R. D. Russell, Analysis of moving mesh PDEs with spatial smoothing, Simon Fraser Mathematics Research Report 93-17, 1993 (unpublished).
11. W.-Z. Huang and D. M. Sloan, A simple adaptive grid method in two dimensions, *SIAM J. Sci. Comput.* **15,** 776 (1994).
12. J. D. Lambert, *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem* (Wiley, Chichester, 1991), 2nd ed.
13. K. Miller and R. N. Miller, Moving finite elements, I, *SIAM J. Numer. Anal.* **18,** 1019 (1981).
14. K. Miller, Moving finite element methods, II, *SIAM J. Numer. Anal.* **18,** 1033 (1981).
15. L. S. Mulholland, W.-Z. Huang, and D. M. Sloan, Pseudospectral solution of near-singular problems using numerical coordinate transformations based on adaptivity, *SIAM J. Sci. Comput.* **18** (1997), in press.
16. J. M. Sanz-Serna and I. Christie, A simple adaptive technique for nonlinear wave problems, *J. Comput. Phys.* **67,** 348 (1986).
17. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, *Numerical Grid Generation* (North-Holland, New York, 1985).
18. G. B. Whitham, *Linear and Nonlinear Waves* (Wiley–Interscience, New York, 1974).