JOURNAL OF COMPUTATIONAL PHYSICS 113, 279–290 (1994)

Moving Mesh Methods Based on Moving Mesh Partial Differential Equations

WEIZHANG HUANG, YUHE REN, AND ROBERT D. RUSSELL

Department of Mathematics and Statistics, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6

Received November 6, 1992; revised December 27, 1993

Several versions of a moving mesh method are developed based on a mesh spatial smoothing technique and on the moving mesh PDEs derived in a previous paper. These versions are quite simple and easy to program. They are applied to three bench-mark one-dimensional problems which show different solution behaviour. The numerical results clearly demonstrate that the present methods are capable of accurately tracking rapid spatial and temporal transitions. © 1994 Academic Press, Inc.

1. INTRODUCTION

Adaptive mesh methods have been widely used in the last decade for solving differential equations which involve large solution variations, such as shock waves, boundary layers, and contact surfaces (e.g., see [HGH91]). It has been amply demonstrated that significant improvements in accuracy and efficiency can be gained by adapting mesh points so that they are concentrated about areas of large solution variation.

For the numerical solution of time-dependent differential equations, adaptive mesh methods can be roughly divided into two categories, static and dynamic. For static methods the redistribution of the nodes, the possible addition of new nodes, and the interpolation of dependent variables from the old mesh to the new mesh are all done at a fixed time. For dynamic methods, or *moving mesh methods*, a mesh equation which involves node speeds is employed to move a mesh having a fixed number of nodes in such a way that the nodes remain concentrated in regions of rapid variation of the solution. The mesh equation and the original differential equation are generally solved simultaneously for the physical solution and the mesh. Interpolation of dependent variables from the old mesh to the new mesh is unnecessary.

Among moving mesh methods, the moving finite element method (MFE) of K. Miller [MM81, Mil81] and the moving finite difference method of Dorfi and Drury [DD87] have aroused considerable interest. The MFE uses a very natural and elegant formulation to control mesh movement. The solution and mesh are both obtained by a process closely associated with equidistribution of one error measure: the residual of the original equation written in finite element form. While the MFE has been subject to some criticism because of its complexity and sensitivity with respect to certain user defined input parameters [FVZ90], proper choice of these parameters unquestionably leads to an efficient method. The method in [DD87] is based upon a moving mesh equation obtained directly from an equidistribution principle. It is recommended in [FVZ90] for actual applications because of its simplicity and relative insensitivity with respect to selected parameters.

The key in developing moving mesh methods lies in formulating a satisfactory mesh equation. It has proven to be surprisingly difficult to derive consistently reliable moving mesh equations. In addition to the capability of concentrating a sufficient number of points in regions of rapid variation of the solution, a satisfactory mesh equation should be simple, easy to program, and reasonably insensitive to the choice of its adjustable parameters. As compared with the problem of discretizing the underlying physical equation, this task is somewhat artificial. That is, the construction of a moving mesh equation cannot be guided completely by physical arguments and must rely on some numerical principles.

In [HRR92], several moving mesh partial differential equations (MMPDEs) based on the equidistribution principle are derived and studied both theoretically and numerically. Some of these MMPDEs are new, and several are related to methods developed in [And83a, And83b, FCLD83, HiSp83, Mad84, Gre85, AF86a, AF86b, CFL86, HL86, DD87, RR92, Ren92]. The first and second of these MMPDEs have some computational difficulty due to the basic term $\partial M/\partial t$, where M denotes the underlying monitor function. The other five basic MMPDEs are both simple and easy to implement. Moreover, it is found in [HRR92] that under very general conditions for these (MMPDEs 3–7) not only are mesh crossings guaranteed

0021-9991/94 $6.00

not to occur, but the meshes retain equidistribution of the monitor function.

The objective of this paper is to develop and test moving mesh methods based on MMPDEs 3–7 and on a spatial mesh smoothing technique. Here, we emphasize that, although some of these methods are related to several existing moving mesh methods, there are key differences. In particular, the discrete approximations to the MMPDEs do differ from related discrete moving mesh equations used previously, and the mesh spatial smoothing technique is different.

An outline of the paper is as follows: In Section 2 moving mesh methods based on the MMPDEs are developed. In Section 3, these methods are applied to three bench-mark problems, a reaction–diffusion equation, the well-known convection–diffusion equation of Burgers and a system of two quasi-nonlinear hyperbolic equations. Section 4 contains conclusions and further discussion.

2. MOVING MESH METHODS

In this section, we develop moving mesh methods based on MMPDEs 3–7, derived in [HRR92]. We start with a review of the equidistribution principle and then describe the discrete approximations to the MMPDEs in Subsection 2.1. In Subsection 2.2, a spatial mesh smoothing technique is presented. A summary description of the moving mesh methods is given in Subsection 2.3.

2.1. *MMPDEs*

Let x and ξ denote the physical and computational coordinates, respectively, assumed without loss of generality to be over the unit interval $[0, 1]$. A one-to-one coordinate transformation between these domains is denoted by

$$x = x(\xi, t), \quad \xi \in [0, 1], \quad x(0, t) = 0, \quad x(1, t) = 1, \quad (1)$$

where t denotes time. We employ the notation

$$f_x \equiv \frac{\partial f}{\partial x} \equiv \left.\frac{\partial f}{\partial x}\right|_{t\text{fixed}}$$

$$f_t \equiv \frac{\partial f}{\partial t} \equiv \left.\frac{\partial f}{\partial t}\right|_{x\text{fixed}}$$

$$\frac{\partial f}{\partial \xi} \equiv \left.\frac{\partial f}{\partial \xi}\right|_{t\text{fixed}} = \left.\frac{\partial f}{\partial x}\right|_{t\text{fixed}} \left.\frac{\partial x}{\partial \xi}\right|_{t\text{fixed}}$$

$$\dot{f} \equiv \frac{df}{dt} \equiv \left.\frac{\partial f}{\partial t}\right|_{\xi\text{fixed}} = \left.\frac{\partial f}{\partial x}\right|_{t\text{fixed}} \left.\frac{\partial x}{\partial t}\right|_{\xi\text{fixed}} + \left.\frac{\partial f}{\partial t}\right|_{x\text{fixed}} \quad (2)$$

for an arbitrary function $f = f(x, t) = f(x(\xi, t), t)$. For a given uniform mesh on the computational domain

$$\xi_i = \frac{i}{n}, \qquad i = 0, 1, \ldots, n, \quad (3)$$

the corresponding mesh in x is

$$\{x_0, x_1, \ldots, x_n\}. \quad (4)$$

For an arbitrary function f on this computational mesh, denote $f_i = f(\xi_i, t)$.

For a monitor function $M(x, t)$ (>0) which provides some measure of the computational error in the solution of the underlying physical PDE, the one-dimensional equidistribution principle (EP) can be expressed in its integral form [Whi79] as

$$\int_0^{x(\xi,t)} M(\tilde{x}, t)\, d\tilde{x} = \xi\theta(t), \quad (5)$$

where

$$\theta(t) = \int_0^1 M(\tilde{x}, t)\, d\tilde{x}. \quad (6)$$

Differentiating (5) with respect to ξ once and twice, we obtain two differential forms of the EP,

$$M(x(\xi, t), t)\frac{\partial}{\partial \xi} x(\xi, t) = \theta(t) \quad (7)$$

and

$$\frac{\partial}{\partial \xi}\left\{M(x(\xi, t), t)\frac{\partial}{\partial \xi} x(\xi, t)\right\} = 0. \quad (8)$$

These EPs, (5), (7), and (8), which do not contain the node speed $\dot{x}(\xi, t)$, are called quasi-static EPs (QSEPs) in [HRR92].

Related to these QSEPs, various MMPDEs are derived in [HRR92]. The ones of concern here are

$$\frac{\partial^2}{\partial \xi^2}(M\dot{x}) = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right), \quad \text{(MMPDE3)}$$

$$\frac{\partial}{\partial \xi}\left(M\frac{\partial \dot{x}}{\partial \xi}\right) = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right), \quad \text{(MMPDE4)}$$

$$-\dot{x} = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right), \quad \text{(MMPDE5)}$$

$$\frac{\partial^2 \dot{x}}{\partial \xi^2} = -\frac{1}{\tau}\frac{\partial}{\partial \xi}\left(M\frac{\partial x}{\partial \xi}\right), \quad \text{(MMPDE6)}$$

and

$$\frac{\partial}{\partial\xi}\left(M\frac{\partial\dot{x}}{\partial\xi}\right)-2\frac{\partial}{\partial\xi}\left(M\frac{\partial x}{\partial\xi}\right)\frac{\partial\dot{x}}{\partial\xi}\Big/\frac{\partial x}{\partial\xi}=-\frac{1}{\tau}\frac{\partial}{\partial\xi}\left(M\frac{\partial x}{\partial\xi}\right). \tag{MMPDE7}$$

These MMPDEs not only force the mesh $(x(\xi, t))$ toward equidistribution but also prevent the mesh from crossing. More specifically, the term $-(1/\tau)(\partial/\partial\xi)(M(\partial M/\partial\xi))$ plays the fundamental role of a correction term to make the mesh equidistribute the monitor function and a source for the mesh movement as a stabilizing term for the mesh trajectories. The parameter τ represents a timescale for forcing the mesh toward equidistribution (see [HRR92]).

Here, we discretize MMPDEs 3–7 in space with centred finite differences on the uniform mesh (3) and use the method of lines. Their discrete approximations are given, respectively, by

$$\frac{1}{(1/n)^2}\left[M_{i+1}\dot{x}_{i+1}-2M_i\dot{x}_i+M_{i-1}\dot{x}_{i-1}\right]=-\frac{E_i}{\tau}, \tag{9}$$

$$\frac{M_{i+1}+M_i}{2(1/n)^2}(\dot{x}_{i+1}-\dot{x}_i)-\frac{M_i+M_{i-1}}{2(1/n)^2}(\dot{x}_i-\dot{x}_{i-1})=-\frac{E_i}{\tau}, \tag{10}$$

$$-\frac{\dot{x}_i}{(1/n)^2}=-\frac{E_i}{\tau(1/n)^2}, \tag{11}$$

$$\frac{1}{(1/n)^2}\left[\dot{x}_{i+1}-2\dot{x}_i+\dot{x}_{i-1}\right]=-\frac{E_i}{\tau}, \tag{12}$$

$$\frac{M_{i+1}+M_i}{2(1/n)^2}(\dot{x}_{i+1}-\dot{x}_i)-\frac{M_i+M_{i-1}}{2(1/n)^2}(\dot{x}_i-\dot{x}_{i-1})-2E_i\frac{\dot{x}_{i+1}-\dot{x}_{i-1}}{x_{i+1}-x_{i-1}}=-\frac{E_i}{\tau}. \tag{13}$$

Here, E_i is the discrete approximation of $(\partial/\partial\xi)(M\cdot\partial x/\partial\xi)$ at $\xi=\xi_i$ given by

$$E_i=\frac{M_{i+1}+M_i}{2(1/n)^2}(x_{i+1}-x_i)-\frac{M_i+M_{i-1}}{2(1/n)^2}(x_i-x_{i-1}). \tag{14}$$

Note that using a value for τ in (11) (MMPDE5) has roughly the same effect as using that value divided by n^2 in the other MMPDEs. This can be useful in the selection of τ for MMPDE5 (see Section 3).

Approximations related to these MMPDEs have been considered by various authors (see Table I). Scheme (9) can be derived from a familiar MMPDE (MMPDE1 in [HRR92]) with a special discretization technique, and it has been studied in [RR92, Ren92] for the case where M is not smoothed in the way described in the next subsection. Blom and Verwer [BV89] use scheme (10) and suggest a certain spatial smoothing of the node distances, but they unfortunately find that it is difficult to get the scheme to converge with a Newton process. It is important to reiterate, more generally, that the discrete approximations of these MMPDEs may be qualitatively very different from related but distinct discrete moving mesh equations. For example, the scheme (13) is quite different from the (discrete) moving mesh equation in the method of Dorfi and Drury [DD87], although MMPDE7 can be derived from it. Equally importantly, a different spatial mesh smoothing technique than has been used previously will be combined with the approximations of the MMPDEs, as we see next.

2.2. *Spatial Smoothing*

It is well known that for moving finite difference methods, some sort of smoothing of the mesh is often useful in order to obtain reasonable accuracy in the computed solution (e.g., see [DD87, FVZ90]). In [DD87], Dorfi and Drury use a technique which smooths the node concentration defined by $1/(x_{i+1}-x_i)$. In [VBFZ89], Verwer *et al.* prove that smoothing the node concentration is basically equivalent to smoothing the monitor function over all points (that is, Eq. (15) below with $p=n$). Since smoothing the monitor function is more straightforward to apply than smoothing the mesh concentration in higher dimensions, that technique is employed here. Specifically, the values of the smoothed monitor function $\tilde{M}$ at nodes are defined by

$$\tilde{M}_i=\sqrt{\sum_{k=i-p}^{i+p}(M_k)^2\left(\frac{\gamma}{1+\gamma}\right)^{|k-i|}\Big/\sum_{k=i-p}^{i+p}\left(\frac{\gamma}{1+\gamma}\right)^{|k-i|}}, \tag{15}$$

where γ is a positive constant called the *smoothing parameter* and p is a nonnegative integer which we refer to as the *smoothing index*. The summations in (15) are understood to contain only elements with indices in the range

TABLE I

Summary Information for MMPDEs 3–7

MMPDE	Discrete approx.	Related references
3	(9)	[RR92, Ren92]
4	(10)	[HL86, BV89]
5	(11)	[And83b]
6	(12)	[AF86a, AF86b, Mad84, Gre85]
7	(13)	[DD87]

between zero and n. The final discrete moving mesh equations are obtained by replacing M_i by $\tilde{M}_i$ in (9)–(13). For simplicity, these moving mesh equations with $\tilde{M}$ will be called the *smoothed moving mesh equations*. Note that the replacement of M_i by $\tilde{M}_i$ is basically equivalent to using a smoother monitor function.

The smoothing parameter γ has been used by many authors and has a natural physical meaning (e.g., see [DD87]). However, the smoothing index p seems to be new in this context and warrants some remarks. Note that p determines the range of smoothing (averaging). The non-smooth and three-point average cases correspond to $p=0$ and $p=1$, respectively. The three-point average, which is commonly used in adaptive methods, results in a five-block-diagonal algebraic system (where the dimensions of the blocks depend on the number of underlying physical PDEs). For general p, the algebraic system is $(3+2p)$-block-diagonal. Therefore, more cost to solve the nonlinear system is generally associated with higher values of p. On the other hand, the higher the value of p, the smoother the resulting mesh. Determining an optimal value for p is not normally an easy task. In our experience, the moving mesh methods with $p=1$ or 2 or 3 usually give good results.

2.3. *Moving Mesh Methods*

We now give a more complete description of the moving mesh methods. Consider a time-dependent problem of the form

$$\frac{\partial u}{\partial t}=f(u), \qquad 0<x<1, \quad t>0, \tag{16}$$

subject to appropriate boundary and initial conditions, where f represents a differential operator involving only spatial derivatives. For example, Burgers' equation has the form (16) with

$$f(u)=\varepsilon\frac{\partial^2 u}{\partial x^2}-\frac{\partial}{\partial x}\left(\frac{u^2}{2}\right), \qquad \varepsilon>0. \tag{17}$$

Using the coordinate transformation (1), (16) can be rewritten in the (quasi-)Lagrangian form

$$\dot{u}-\frac{\partial u}{\partial x}\dot{x}=f(u), \qquad 0<x<1, \quad t>0. \tag{18}$$

Discretizing by centred finite differences gives

$$\dot{u}_i-\frac{(u_{i+1}-u_{i-1})}{(x_{i+1}-x_{i-1})}\dot{x}_i=f_i, \qquad i=1, 2, \ldots, n-1, \tag{19}$$

where f_i denotes the discrete approximation to the differential operator f at $\xi=\xi_i$ using a conservative centred finite difference scheme.

For a given monitor function $M(x, t)$, the system to solve numerically consists of (19), one of the smoothed discrete moving mesh systems (9)–(13), and the corresponding boundary and initial conditions for the mesh x and solution u. Hereafter, we refer to the methods associated with MMPDEs 3–7 as simply Methods 3–7.

Values of the three parameters τ, γ (smoothing) and p (smoothing index) need be selected for these moving mesh methods. In our experience, the choice of γ is fairly insensitive, and generally γ can be fixed. In this paper, we choose

$$\gamma=2. \tag{20}$$

The value for p is taken as 1, 2, 3, or 4. The selection of the value for τ is discussed in Section 3.

3. NUMERICAL EXPERIMENTS

In this section, numerical results are presented for the moving mesh methods applied to three problems, a reaction–diffusion equation which models a problem from combustion theory, the well-known convection–diffusion equation of Burgers, and a system of two quasi-nonlinear hyperbolic equations which may be considered as a prototype of an opposite travelling waves problem. We choose these problems as our test examples because they show qualitatively different solution behaviour and because they have been used extensively in the moving mesh literature. It is worth emphasizing that these problems are also the ones used in [FVZ90] to compare the reliability, robustness, and efficiency of three representative moving mesh methods.

Throughout, we shall use the arclength monitor function

$$M(x, t)=\sqrt{1+(\partial u/\partial x)^2} \tag{21}$$

and discretize with centred three-point finite differences at interior nodes and with one-sided two-point differences at boundary nodes. The ODE systems for the moving mesh methods are solved using the double precision version of the stiff ODE solver DASSL [Pet82]. The time integration method is chosen as the backward differentiation formulas (BDF), wherein an approximate Jacobian is computed by DASSL internally using finite differences. Other required input data are the initial solution, the initial mesh, and relative and absolute local time stepping error tolerances *rtol* and *atol* (in a root-mean-square norm). In all cases, a uniform initial mesh is used. The term

$$E(t)=\max_{i=1,\ldots,n-1}\frac{1}{M_i(t)}|E_i(t)| \tag{22}$$

is used to measure the level of mesh equidistribution and

$$H(t) = \min_{i=0,\ldots,n-1} (x_{i+1}(t) - x_i(t)) \tag{23}$$

is used to denote the minimal spacing at time t. We also use NTS, JAC, ETF, and CFN to denote the total number of time steps taken, Jacobian evaluations, error test failures, and convergence test failures in Newton iteration. All computations are performed on a SPARC 1+ in double precision.

3.1. *Problem 1: A Scalar Reaction–Diffusion Problem from Combustion Theory*

This problem is described in [AF86a] as a model of a single-step reaction diffusion and reads

$$\begin{aligned} &u_t = u_{xx} + \frac{Re^{\delta}}{a\delta}(1+a-u)\,e^{-\delta/u}, && 0<x<1, \quad t>0 \\ &u_x(0,t) = 0, \qquad u(1,t) = 1, && t>0 \\ &u(x,0) = 1, && 0 \leqslant x \leqslant 1, \end{aligned} \tag{24}$$

where R, δ, and a are constants. The solution represents the temperature of a reactant in a chemical system. For small times the temperature gradually increases from unity with a "hot spot" forming at $x = 0$. At a finite time, ignition occurs, causing the temperature at $x = 0$ to increase rapidly to $1 + a$. A flame front then forms and propagates towards $x = 1$ at a high speed. The degree of difficulty of the problem is determined by the value of δ. Following [AF86a, VBS89, FVZ90], we first choose the problem parameters $a = 1$, $R = 5$, and $\delta = 20$. For the current choice of parameters, the steady state is reached slightly before time $t = 0.29$, which we take as the end point of the time integration. We use times $t = 0.26, 0.27, 0.28, 0.29$ for output.

As pointed out in [FVZ90], a numerical difficulty is that the start of the ignition must be detected accurately without overshooting by the local error control mechanism of the stiff ODE solver. Small errors at this time can result in significantly larger global errors later on. Thus, small tolerances for the time integration are needed. Indicative of this sensitivity is the fact that all experiments in [FVZ90] for this flame problem with methods in [Pet87, DD87, MM81] show a deviation from the reference solution copied from [VBS89] in a neighbourhood of $x = 0$ at time $t = 0.26$. However, while it is claimed in [VBS89] that their reference solution with 151 moving mesh nodes may not be accurate at $t = 0.26$ in the neighbourhood of $x = 0$, errors do not seem to be as large as those in [FVZ90]. In fact, this reference solution listed in Verwer *et al.* [VBS89] has the value of $u(0, 0.26)$: ≈ 1.59, while the solution obtained using DASSL (with time tolerances $atol = rtol = 10^{-8}$) with *2001* uniform nodes has $u(0, 0.26)$: ≈ 1.61.

Deviations may be caused by inaccuracy in both the time integration and the discrete approximation to the problem, especially the approximation to the Neumann boundary condition at $x = 0$. To show this, we test two discrete approximations to the Neumann boundary condition,

$$\frac{u_1 - u_0}{x_1 - x_0} = 0 \tag{25}$$

and

$$\dot{u}_0 = \frac{2(u_1 - u_0)}{(x_1 - x_0)^2} + \frac{Re^{\delta}}{a\delta}(1 + a - u_0)\,e^{-\delta/u_0}. \tag{26}$$

The approximation (26) is obtained as follows: We introduce a fictitious node $x_{-1} \equiv -x_1$ which lies outside the physical domain, and we use the reflection boundary condition

$$\frac{u_1 - u_{-1}}{x_1 - x_{-1}} = 0 \tag{27}$$

and the difference equation at $x = x_0$,

$$\dot{u}_0 = \frac{u_1 - 2u_0 + u_{-1}}{(x_1 - x_0)^2} + \frac{Re^{\delta}}{a\delta}(1 + a - u_0)\,e^{-\delta/u_0}. \tag{28}$$

Then, eliminating u_{-1} from (27) and (28), we obtain (26), which is known to be more accurate than (25) (e.g., see [Fle88]).

In Table II, we list results for $u(0, 0.26)$ obtained with Method 4 and with $\tau = 10^{-3}$, $p = 2$. It can be seen from this table that when large tolerances (such as $atol = rtol = 10^{-3}$) are used, both approximations give large deviations. For small tolerances, the deviations can be reduced, but only

TABLE II

Problem 1

n	$atol = rtol$	(25)	(26)
20	10^{-3}	2.00000	2.00000
	10^{-5}	1.42419	1.64520
	10^{-6}	1.41794	1.61577
	10^{-8}	1.41677	1.61124
40	10^{-3}	2.00000	2.00000
	10^{-5}	1.50119	1.65020
	10^{-6}	1.48994	1.61996
	10^{-8}	1.48786	1.61527
200	10^{-8}	1.58786	1.61659

Note. Values of $u(0, 0.26)$ obtained with Method 4 (MMPDE4) and with $\tau = 10^{-3}$, $p = 2$. In [VBS89], the value obtained with 2001 non-moving, uniform nodes is about 1.61.

TABLE III

Problem 1: Results for Methods 3–7 with $n = 20$

MMPDE	p	τ	NTS	JAC	ETF	CFN	$u(0, 0.26)$
3	2	1.0	492	43	20	0	1.61561
		10^{-3}	267	40	16	0	1.61578
		10^{-5}	328	35	2	0	1.62038
		10^{-7}	351	38	3	0	1.62035
4	2	1.0	463	43	28	0	1.61561
	0	10^{-3}	291	42	18	0	1.61589
	1		249	42	16	0	1.61584
	2		243	40	17	0	1.61577
	4		250	37	18	0	1.61571
	2	10^{-5}	320	37	2	0	1.62038
		10^{-7}	357	37	1	0	1.62035
5	2	$4 \times 10^{+2}$	497	30	20	0	1.61560
		4×10^{-1}	299	32	20	0	1.61583
		4×10^{-3}	299	38	18	0	1.61566
		4×10^{-5}	356	37	2	0	1.62035
6	2	1.0	513	31	25	0	1.61561
		10^{-3}	241	34	20	0	1.61574
		10^{-5}	334	35	3	0	1.62037
		10^{-7}	363	37	1	0	1.62035
7	2	1.0	502	43	22	0	1.61561
		10^{-3}	225	39	19	0	1.61577
		10^{-5}	306	38	2	0	1.62038
		10^{-7}	354	35	1	0	1.62035

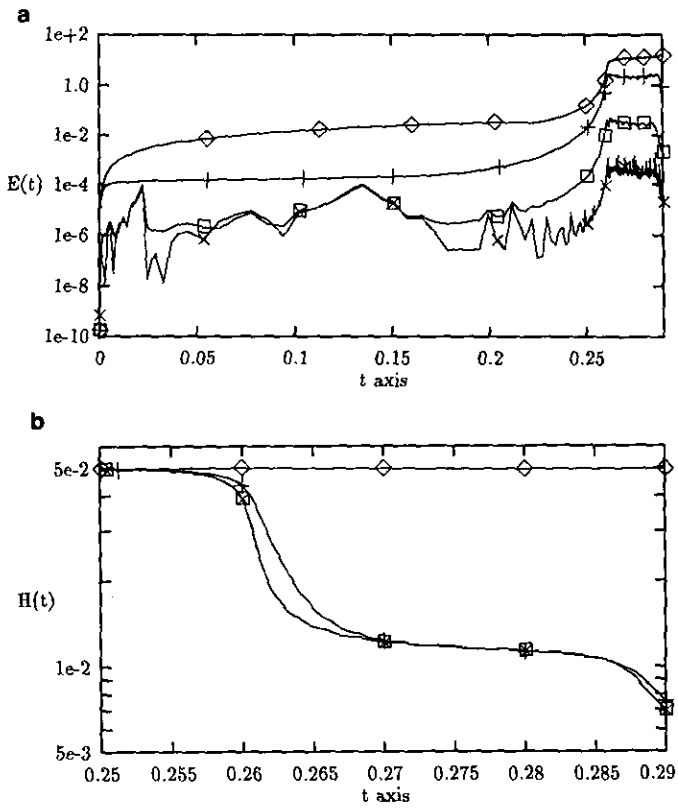

FIG. 1. *Problem* 1 *with* $\delta = 20$. Functions $E(t)$ and $H(t)$ for Method 4 (MMPDE4) with $n = 20$ and $p = 2$: ◇—◇, $\tau = 1.0$; +—+, $\tau = 10^{-3}$; □—□, $\tau = 10^{-5}$; ×—×, $\tau = 10^{-7}$.

with the approximation (26) can accurate results be obtained for small n.

In the plots for this case, the reference solution (solid lines) is the one obained with Method 4 and with $n = 200$, $\tau = 10^{-3}$, $p = 2$, and $atol = rtol = 10^{-6}$. The other computations are performed using the approximation (26) and $atol = rtol = 10^{-6}$.

An important parameter in the present methods is τ. Typical results are obtained for Method 4 (MMPDE4) with four decreasing values of τ, fixed $n = 20$, and $p = 2$. For $\tau = 1$, the start of the ignition is detected quite accurately, but a nearly nonmoving mesh results and the numerical flame front is too slow in the propagation phase. As τ decreases, the mesh follows the flame much better and the number of time steps (NTS) is reduced significantly (see Table III). Functions $E(t)$ and $H(t)$ for the four values of τ are shown in Fig. 1. It can be seen in this figure that, during the formation of the flame front, the nodes concentrate around the sharp gradient region and the minimal mesh spacing decreases quickly. Meanwhile, $E(t)$ rapidly increases by a factor of about 100. However, smaller values of $E(t)$ are obtained using smaller values of τ.

The results obtained with Methods 3 and 5–7 are similar to those with Method 4 and are summarized in Table III. One comment is in order regarding the choice of τ in Method 5. Recall from Section 2.1 that τ in Method 5 is roughly comparable to τ/n^2 for the other methods. This suggests that the four values: n^2, $10^{-3}n^2$, $10^{-5}n^2$, and $10^{-7}n^2$ may be chosen for τ in Method 5 corresponding to the selected four values 1.0, 10^{-3}, 10^{-5}, and 10^{-7} for τ in other methods. This choice for Method 5 indeed leads to results similar to those for other methods.

Figure 2 shows typical results for mesh trajectories and solutions for $\delta = 20$ obtained by Method 4 with $n = 20$, $p = 2$, and $\tau = 10^{-5}$. In this case the flame layer is not very thin, and nearly the same accuracy can be obtained in the conventional way with a uniform, nonmoving mesh consisting of about 40 nodes. In this sense it is a limited test of moving mesh methods.

A more interesting choice of problem parameters is $a = 1$, $R = 5$, and $\delta = 30$, used in [Pet87]. The flame layer in this case is much thinner, and higher mesh adaption is required. The observations made from case $\delta = 20$ are still true, except that smaller values of τ are needed in order to obtain accurate approximations. For example, when Method 4 with $n = 20$ and $p = 2$ is used, $\tau = 10^{-3}$ leads to a slowly propagating flame front, but $\tau = 10^{-5}$ results in the proper one. The latter result is shown in Fig. 3. We find that to

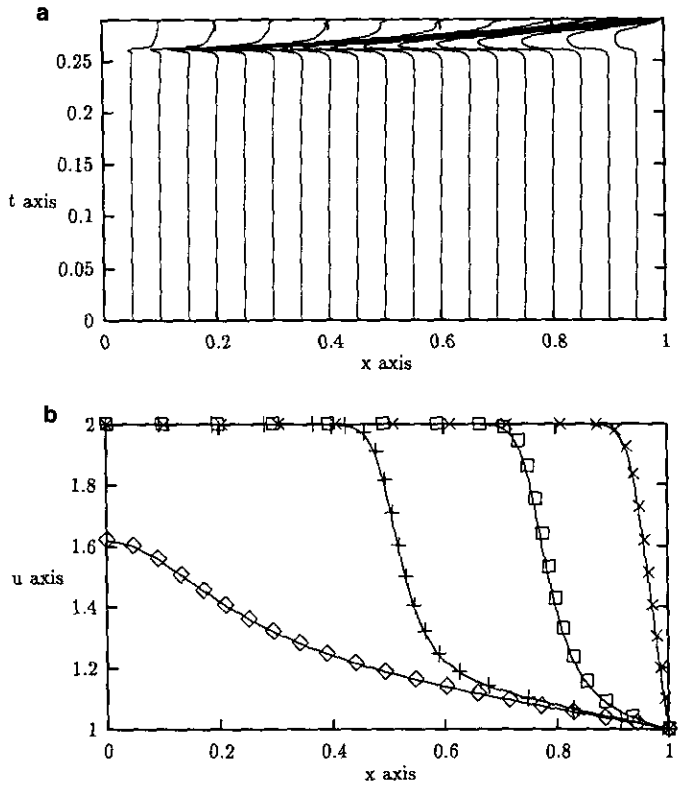

FIG. 2. *Problem* 1 *with* $\delta = 20$. Mesh trajectories and solutions (at $t = 0.26, 0.27, 0.28, 0.29$) for Method 4 (MMPDE4) with $n = 20$, $p = 2$, and $\tau = 10^{-5}$.

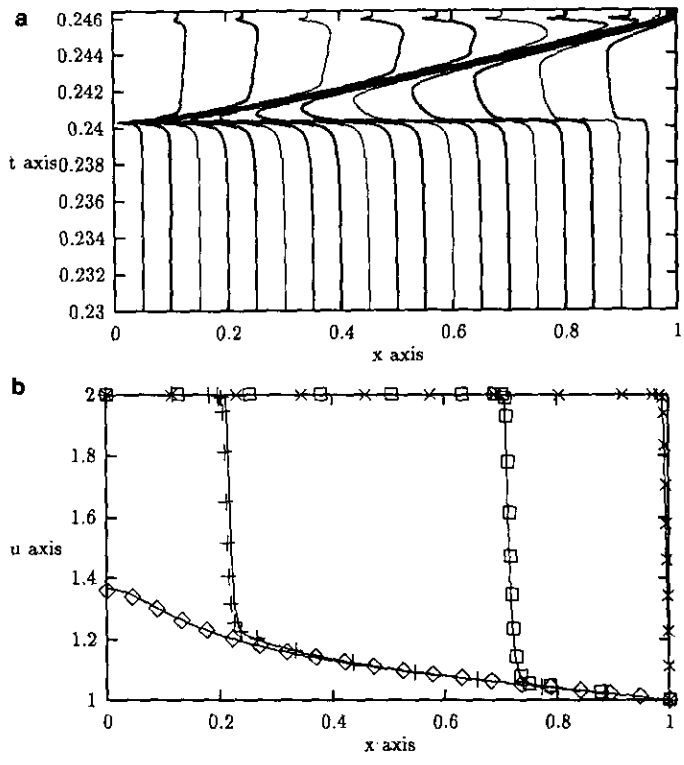

FIG. 3. *Problem* 1 *with* $\delta = 30$. Mesh trajectories and solutions (at $t = 0.240, 0241, 0.244, 0.247$) for Method 4 (MMPDE4) with $n = 20$, $p = 2$, and $\tau = 10^{-5}$. The performance yields NTS = 1253, JAC = 93, ETF = 18, and CFN = 0.

obtain the same accuracy more than 200 uniform nodes and orders of magnitude more CPU time are needed. The reference solution (in solid lines) in Fig. 3 is obtained by using DASSL with $atol = rtol = 10^{-8}$ and 2001 uniform nodes.

We conclude this problem with a comment about the choice for p. For the easier case $\delta = 20$ the spatial smoothing is not too critical since the flame layer is not very thin and $p = 0$ gives satisfactory results. However, some spatial smoothing is important for the case $\delta = 30$. Still, the computations with $p = 1,2,3$, and 4 all give reasonably accurate results.

3.2. *Problem 2: Burgers' Equation*

The second test problem is the well-known Burgers' equation, first with a smooth initial solution,

$$u_t = \varepsilon u_{xx} - (u^2/2)_x, \qquad 0 < x < 1, \quad t > 0, \quad \varepsilon = 10^{-4},$$
$$u(0, t) = u(1, t) = 0, \qquad t > 0, \tag{29}$$
$$u(x, 0) = \sin(2\pi x) + \tfrac{1}{2}\sin(\pi x), \qquad 0 \leqslant x \leqslant 1.$$

This problem frequently serves as a test example for moving mesh methods, e.g., in [GDM81, HMW86, FVZ90,

TABLE IV

Problem 2: Results for Methods 3–7

MMPDE	n	p	τ	NTS	JAC	ETF	CFN
3	20	2	10^{-3}	527	119	39	2
			10^{-5}	341	93	29	0
			10^{-7}	348	100	26	0
4	20	2	10^{-2}	822	176	65	0
		0	10^{-3}	1110	227	81	2
		1		432	105	31	0
		2		352	83	36	0
	40			443	108	36	0
	20		10^{-5}	347	82	25	0
			10^{-7}	349	99	26	0
5	20	2	10.0	384	83	30	1
			0.1	350	79	35	0
			10^{-3}	377	89	29	0
			10^{-5}	379	109	29	0
6	20	2	1.0	373	67	24	0
			10^{-3}	329	81	31	0
			10^{-5}	358	83	21	0
			10^{-7}	354	92	24	0
7	20	2	10^{-3}	551	122	43	3
			10^{-5}	364	82	27	0
			10^{-7}	389	102	22	0

Ren92]. The solution is a wave that develops a very steep gradient and subsequently moves towards $x = 1$. Because of the zero boundary values, the wave amplitude diminishes with increasing time. Following [Mil81] and [FVZ90], we consider the time interval [0, 2] and use $t = 0.2, 0.6, 1.0, 1.4$, and 2.0 for solution output points. This is quite a challenging problem for methods which employ centred difference approximations. The location of the fine mesh region is very critical, and these moving mesh methods tend to generate spurious oscillations as soon as the mesh becomes slightly too coarse in the layer region, just as with standard centred differences with a non-moving mesh.

We use $atol = 10^{-4}$ and $rtol = 10^{-5}$ in all computations for this problem. We first compute the solution using Method 4 (MMPDE4) with $n = 200$, $\tau = 10^{-3}$, and $p = 2$. The result is used as the reference solution which appears as a solid line in the plots. (Computing an accurate reference solution with a uniform mesh is prohibitively expensive.)

Table IV summarizes some computations done using Methods 3–7. Five values of τ are used with Method 4 but the numerical integration fails for $\tau = 10^{-1}$. The ODE solver indicates repeated error test failures and small time step-sizes. Upon examination, we find that the failure results from oscillations in the solution due to the mesh being too coarse in the steep gradient region. For $\tau = 10^{-2}$, although Method 4 works, the mesh points are still somewhat slow in moving into the steep region, and slight oscillations appear in the solution when this steepening occurs (about $t = 0.26$). This slow response of the mesh can be seen clearly from the graph of $H(t)$ in Fig. 4b. For smaller values of τ, Method 4 works well. The mesh follows the wave properly and, not only do the oscillations disappear, but the solutions are fairly accurate. In Table IV, we also see that NTS, JAC, and ETF are significantly reduced. Typical results are shown in Fig. 5. It is interesting to note that the mesh adjusts rapidly when the wave reaches the boundary $x = 1$. Figure 4a shows that the mesh continues to equidistribute the monitor function. The oscillations in $E(t)$ for small values of τ illustrate that the mesh may deviate from the equidistribution mesh but can still recover quickly. The efficiency of Method 4 in adapting the mesh can be seen from Fig. 4b. It shows that after the steep gradient forms, the minimal mesh spacing is reduced below 2×10^{-4}. Comparable results with a non-moving, uniform mesh would be prohibitively expensive.

Results obtained with Methods 3 and 7 are found to be similar to those with Method 4, except that the performance with $\tau = 10^{-3}$ is slightly inferior (see Table IV).

Method 5 with $\tau = 100$ also fails due to oscillations in the solution. It appears to adapt the mesh faster and to give a slightly more accurate solution than Method 4 with comparative τ values of 10^{-2}, 10^{-3}, and 10^{-5}.

While the computation with Method 6 also fails for

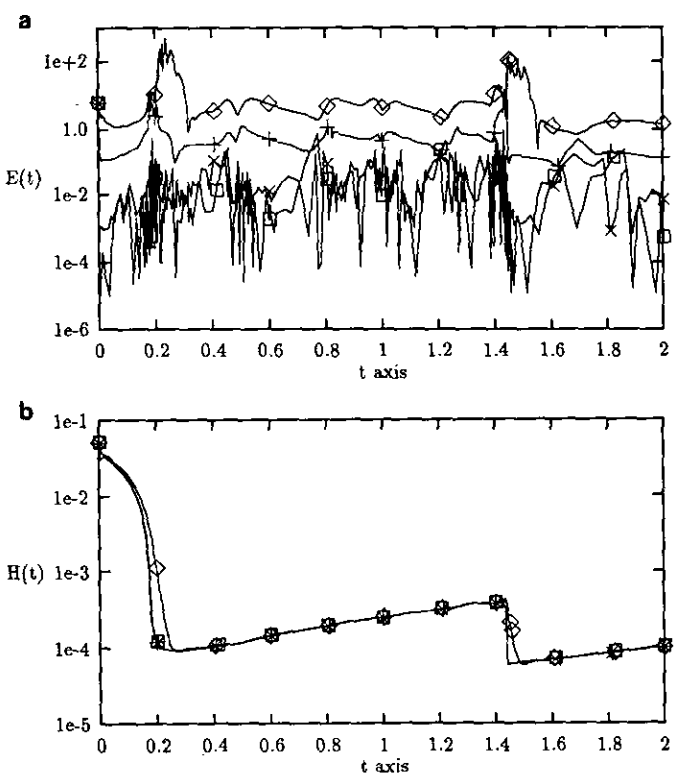

FIG. 4. *Problem* 2. Functions $E(t)$ and $H(t)$ for Method 4 (MMPDE4) with $n = 20$ and $p = 2$: ◇—◇, $\tau = 10^{-2}$; +—+, $\tau = 10^{-3}$; □—□, $\tau = 10^{-5}$; ×—×, $\tau = 10^{-7}$.

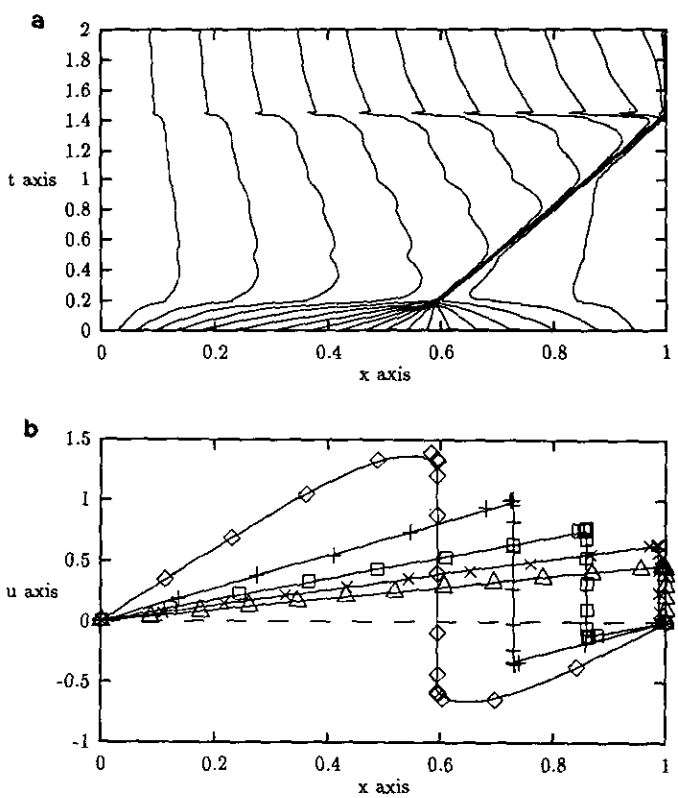

FIG. 5. *Problem* 2. Mesh trajectories and solutions (at $t = 0.2, 0.6, 1.0, 1.4$, and 2.0) for Method 4 (MMPDE4) with $n = 20$, $p = 2$, and $\tau = 10^{-3}$.

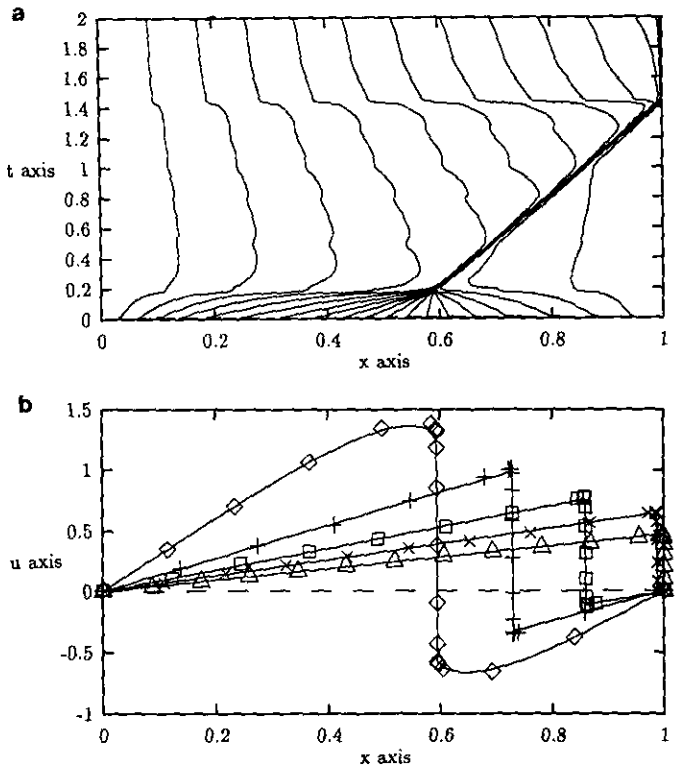

FIG. 6. *Problem* 2. Mesh trajectories and solutions (at $t = 0.2, 0.6, 1.0, 1.4$, and 2.0) for Method 6 (MMPDE 6) with $n = 20$, $p = 2$, and $\tau = 10^{-3}$.

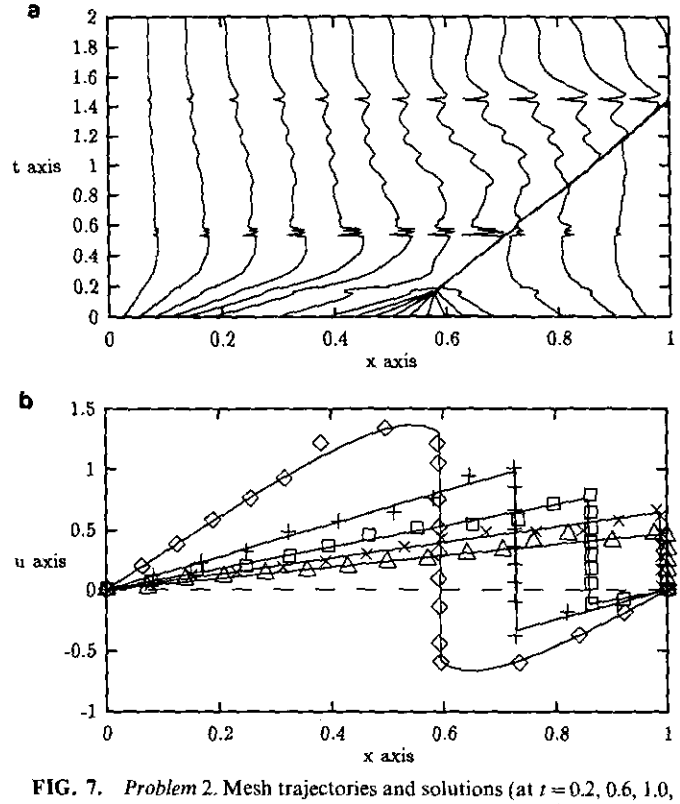

FIG. 7. *Problem* 2. Mesh trajectories and solutions (at $t = 0.2, 0.6, 1.0, 1.4$, and 2.0) for Method 4 (MMPDE4) with $n = 20$, $\tau = 10^{-3}$, and $p = 0$.

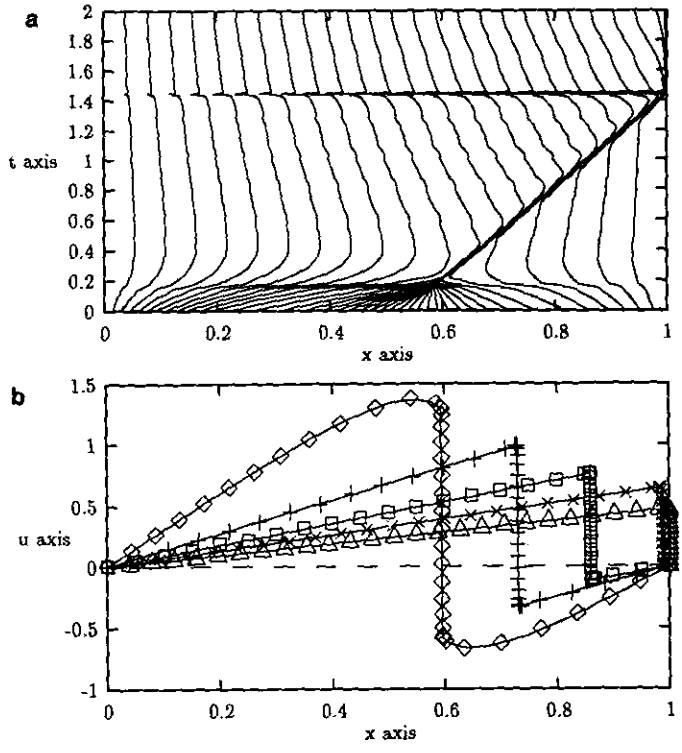

FIG. 8. *Problem* 2. Mesh trajectories and solutions (at $t = 0.2, 0.6, 1.0, 1.4$, and 2.0) for Method 4 (MMPDE4) with $n = 40$, $p = 2$, and $\tau = 10^{-3}$.

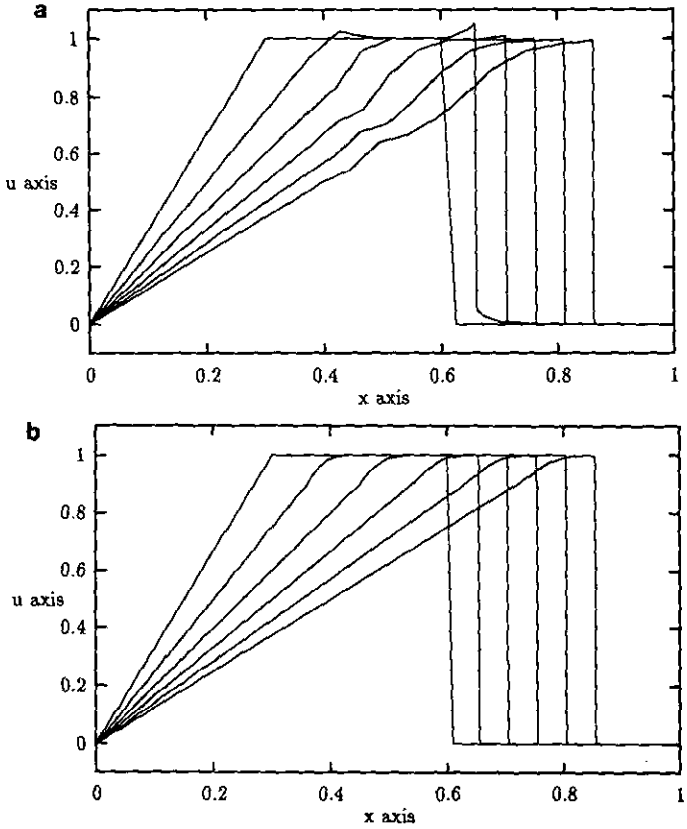

FIG. 9. *Problem* 2 *with corner structures.* Solutions (at $t = 0, 0.1, 0.2, 0.3, 0.4$, and 0.5) for Method 4 (MMPDE4) with an initial uniform mesh, $p = 4$, $\tau = 10^{-3}$, and $n = 40$ for (a), $n = 200$ for (b).

$\tau = 10$ due to oscillations in the solution, with $\tau = 1.0$, 0.1, and 10^{-2} it is successful, whereas Method 4 fails. Also, Method 6 produces smoother mesh trajectories and more accurate solutions (see Fig. 6).

To illustrate the importance of the spatial mesh smoothing for this problem, several computations are performed with Method 4. Figure 7 shows how the mesh trajectories and solutions obtained with $p = 0$ are very erratic, with visible oscillations occurring in the solution. Upon increasing p, the mesh trajectories become smoother and nonoscillatory solutions are obtained (see Figs. 7 and 5). Information about these runs is listed in Table IV.

The figures show clearly that the computed solution with these MMPDEs is generally quite accurate with $n = 21$ mesh points, except perhaps near the corners of the layers. These results compare very favourable with those elsewhere (e.g., see [FVZ90]). For reference purposes, we compute the solution with Method 4 and $n = 40$. Summary information is included in Table IV and Fig. 8.

It has been pointed out to us by Keith Miller that these methods with the arclength monitor function will perform poorly for problems with "sharp-but-not-steep" corner structures. To see this, consider Burgers' equation (29) with an initial solution $u(x, 0)$ given as the piecewise linear func-

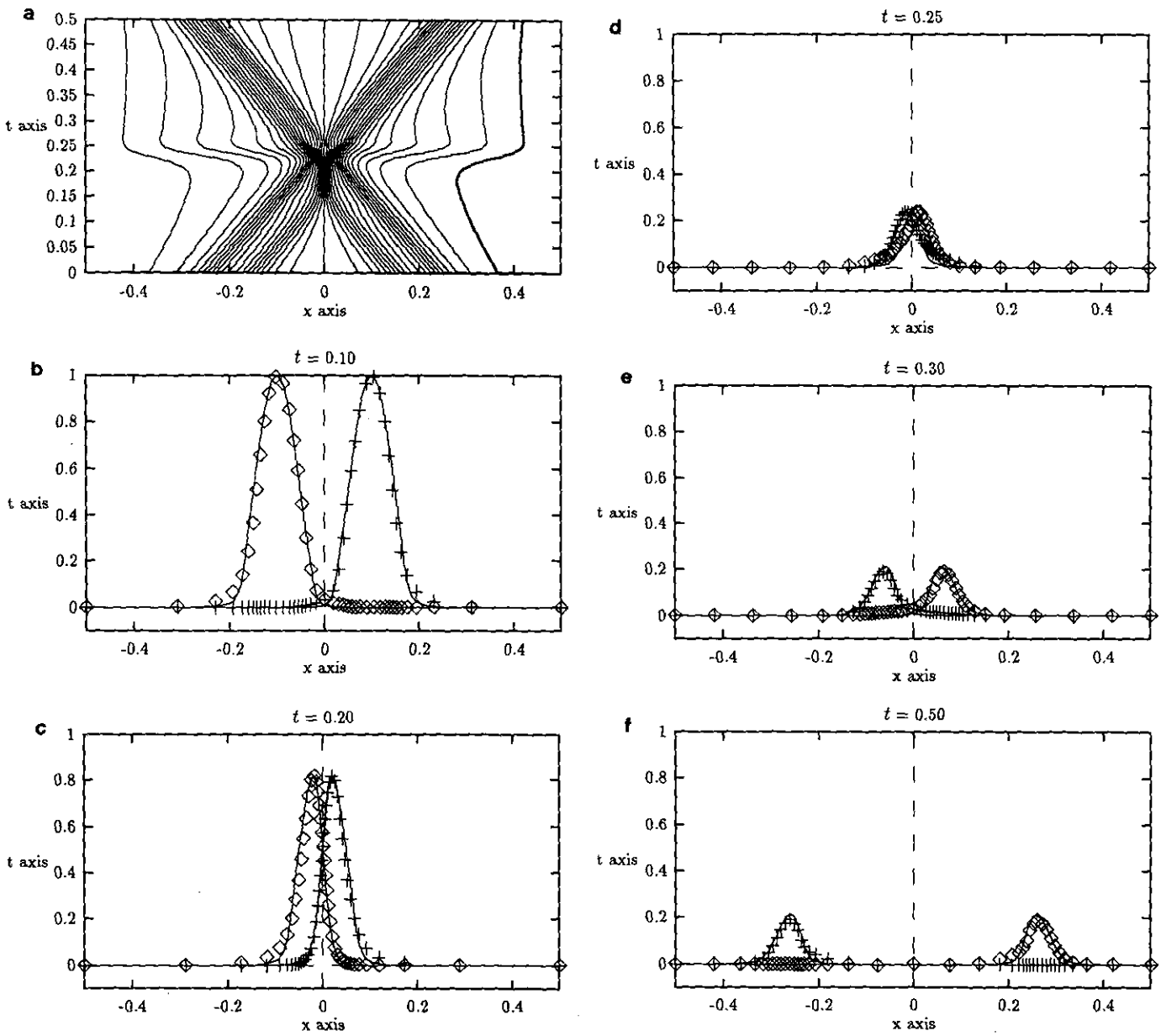

FIG. 10. *Problem* 3. Mesh trajectories and solutions (at $t = 0.1$, 0.2, 0.25, 0.3, and 0.5) obtained with Method 6 (MMPDE6) and with $n = 40$, $\tau = 10^{-3}$, and $p = 2$. The performance yields NTS = 207, JAC = 35, ETF = 1, and CFN = 0. In the plots, $\diamond$ represents $u(x, t)$ and + represents $v(x, t)$.

tion whose graph goes through the points (0, 0), (0.3, 1), (0.6, 1), (0.61, 0), and (1, 0). The initial corners at (0.6, 1) and (0.61, 0) will quickly become the corners of a steep shock moving with speed +0.5. These two "sharp-and-steep" corners will be adequately tracked and resolved with small local mesh spacing by the present methods with the arclength monitor function. The initial corner at (0.3, 1), however, will remain a "sharp-but-not-steep" corner moving ahead with speed +1. Thus there is nothing in the methods to produce small Δx's near this corner, nor is there anything in the methods to cause the nodes near this corner to track with the desired speed +1. Hence, oscillations occur near this moving corner. These are seen in Figs. 9a and 9b which show solutions obtained with MMPDE4 and $\tau = 10^{-3}$, $p = 4$. The figures also show that the oscillations can be significantly weakened by increasing the number of nodes. (We note here that the solution obtained with 2001 uniform nodes still involves very rapid oscillations.) For such problems the solution might be better resolved by using the curvature monitor function, as discussed in [BV89].

3.3. *Problem 3: Waves Travelling in Opposite Directions*

Our final example is a two-component, quasi-nonlinear hyperbolic system, the solution of which is composed of two waves travelling in opposite directions and located initially at $x = -0.2$ and $x = 0.2$. The system is given by

$$\begin{aligned} u_t &= -u_x - 100uv, && -0.5 < x < 0.5, \quad t > 0, \\ v_t &= v_x - 100uv, && -0.5 < x < 0.5, \quad t > 0, \\ u(-0.5, t) &= v(0.5, t) = 0, && t > 0, \end{aligned} \tag{30}$$

with the initial conditions

$$\begin{aligned} u(x, 0) &= \begin{cases} 0.5[1 + \cos(10\pi x)], & x \in [-0.3, -0.1], \\ 0, & \text{otherwise}; \end{cases} \\ v(x, 0) &= \begin{cases} 0.5[1 + \cos(10\pi x)], & x \in [0.1, 0.3], \\ 0, & \text{otherwise}. \end{cases} \end{aligned} \tag{31}$$

This problem is used in [VBS89, FVZ90] as a test example for moving mesh methods. The time integration interval is [0, 0.5] with output times $t = 0.1$, 0.2, 0.25, 0.3, and 0.5.

We use the arclength-like monitor function

$$M(x, t) = \sqrt{1 + 10(\partial u/\partial x)^2 + 10(\partial v/\partial x)^2}, \tag{32}$$

which is particularly adequate since the waves are not very steep after they separate, to obtain sufficient resolution in the vicinity of the two waves. In the computations $atol = 10^{-4}$ and $rtol = 10^{-5}$ are used.

For this problem, Methods 3–7 all give similar results. Figure 10 shows typical results obtained with Method 6. The reference solution, which is plotted in solid lines, is obtained with Method 6 and with $n = 200$, $\tau = 10^{-3}$, and $p = 4$.

It is interesting to note that, unlike in [FVZ90], Methods 3–7 use a uniform initial mesh. With the "forcing term" on the right-hand side for these MMPDEs, no initial mesh redistribution stage is necessary, and at the beginning of the computation the mesh quickly adjusts to equidistribute the arclength and follow the waves. From Fig. 10, the solutions are seen to be fairly accurate. Higher resolution is obtained with the use of more mesh points. Finally, note that the accurate refinement in the vicinity of the travelling waves is maintained during and after the wave interaction.

4. CONCLUSIONS AND COMMENTS

Several versions of a moving mesh method have been developed in previous sections based on the moving mesh PDEs derived in [HRR92] and on a spatial mesh smoothing technique. These versions are found to be quite simple and straightforward to implement. They are applied to three bench-mark one-dimensional problems displaying different types of solution behaviour. The numerical results are very encouraging and clearly show that the methods are capable of accurately tracking rapid spatial and temporal transitions.

The moving mesh methods involve three parameters. However, the parameter values are generally easy to select, and the performance of the methods are relatively insensitive to their choice. This is in contrast to experience with previous methods (e.g., see [FVZ90]). One important parameter is the mesh parameter τ. The numerical experiments demonstrate that good results can be obtained for a wide range of value of τ. For the most difficult problem of the three (Burgers' equation), the mesh smoothing parameter p must be chosen to be greater than one. Method 6 (MMPDE6) then works best, followed by Methods 4 (MMPDE4) and 5 (MMPDE5). (Note that MMPDE6 can be obtained by letting the monitor function M equal one on the left-hand side of MMPDE4.) In general, the results for MMPDE 3–7 appear to be comparable to or superior to those for previous moving mesh methods. Furthermore, unlike for many methods, extension of MMPDEs 3 and 4 to multidimensions is possible by using quasi-static multidimensional formulas, such as in [BS82, Dvi91, HS92]. Such an extension is currently under investigation.

ACKNOWLEDGMENT

The authors are grateful to Keith Miller, who suggested a number of valuable improvements in the original manuscript.

REFERENCES

[AF86a] S. Adjerid and J. E. Flaherty, *SIAM J. Numer. Anal.* **23**, 778 (1986).

[AF86b] S. Adjerid and J. E. Flaherty, *Comput. Methods Appl. Mech. Eng.* **55**, 3 (1986).

[And83a] D. A. Anderson, AIAA Paper, 83-1931, 1983, p. 311 (unpublished).

[And83b] D. A. Anderson, in *Adaptive Computational Methods for Partial Differential Equations*, edited by I. Babuška, J. Chandra, and J. E. Flaherty (SIAM, Philadelphia, 1983), p. 208–223.

[BS82] J. U. Brackbill and J. S. Saltzman, *J. Comput. Phys.* **46**, 342 (1982).

[BV89] J. G. Blom and J. G. Verwer, Report NM-N8902, CWI, Amsterdam, 1989 (unpublished).

[CFL86] J. M. Coyle, J. E. Flaherty, and R. Ludwig, *J. Comput. Phys.* **62**, 26 (1986).

[DD87] E. A. Dorfi and L. O'C. Drury, *J. Comput. Phys.* **69**, 175 (1987).

[Dvi91] A. S. Dvinsky, *J. Comput. Phys.* **95**, 450 (1991).

[FCLD83] J. E. Flaherty, J. M. Coyle, R. Ludwig, and S. F. Davis, in *Adaptive Computational Methods for Partial Differential Equations* edited by I. Babuška, J. Chandra, and J. E. Flaherty (SIAM, Philadelphia, 1983), p. 144.

[Fle88] C. A. J. Fletcher, *Computational Techniques for Fluid Dynamics*, Vol. I (Springer-Verlag, Berlin/Heidelberg, 1988).

[FVZ90] R. M. Furzeland, J. G. Verwer, and P. A. Zegeling, *J. Comput. Phys.* **89**, 349 (1990).

[GDM81] R. J. Gelinas, S. K. Doss, and K. Miller, *J. Comput. Phys.* **40**, 202 (1981).

[Gre85] J. B. Greenberg, *AIAA J.* **23**, 317 (1985).

[HGH91] D. F. Hawken, J. J. Gottlieb and J. S. Hansen, *J. Comput. Phys.* **95**, 254 (1991).

[HiSp83] R. G. Hindman and J. Spencer, AIAA Paper 83-0450, 1983, p. 1 (unpublished).

[HL86] J. M. Hyman and B. Larrouturou, Los Alamos, Report LA-UR-86-1678, 1986 (unpublished).

[HMW86] A. N. Hrymak, G. J. McRae, and A. W. Westerberg, *J. Comput. Phys.* **63**, 168 (1986).

[HRR92] W. Huang, Y. Ren, and R. D. Russell, *SIAM J. Numer. Anal.*, to appear.

[HS92] W. Huang and D. M. Sloan, *SIAM J. Sci. Comput.*, to appear.

[Mad84] N. K. Madsen, in *PDE Software: Modules, Interfaces and Systems*, edited by B. Engquist and T. Smedsaas (North-Holland, Amsterdam, 1984).

[Mi81] K. Miller, *SIAM J. Numer. Anal.* **18**, 1033 (1981).

[MM81] K. Miller and R. N. Miller, *SIAM J. Numer. Anal.* **18**, 1019 (1981).

[Pet82] L. R. Petzold, SAND82-8637, Sandia Labs, Livermore, CA, 1982 (unpublished).

[Pet87] L. R. Petzold, *Appl. Numer. Math.* **3**, 347 (1987).

[Ren92] Y. Ren, Ph.D. thesis, Department of Mathematics and Statistics, Simon Fraser University, Canada, 1992.

[RR92] Y. Ren and R. D. Russell, *SIAM J. Sci. Statist. Comput.* **13**, 1265 (1992).

[VBFZ89] J. G. Verwer, J. G. Blom, R. M. Furzeland, and P. A. Zegeling, in *Adaptive Methods for Partial Differential Equations*, edited by J. E. Flaherty, P. J. Paslow, M. S. Shephard, and J. D. Vasilakis (SIAM, Philadelphia, 1989), p. 160.

[VBS89] J. G. Verwer, J. G. Blom, and J. M. Sanz-Serna, *J. Comput. Phys.* **82**, 454 (1989).

[Whi79] A. B. White, Jr., *SIAM J. Numer. Anal.* **16**, 472 (1979).