JOURNAL OF COMPUTATIONAL PHYSICS 89, 349–388 (1990)

A Numerical Study of Three Moving-Grid Methods for One-Dimensional Partial Differential Equations Which Are Based on the Method of Lines*

R. M. FURZELAND

Koninklijke/Shell-Laboratorium, Amsterdam (Shell Research B.V.), P.O. Box 3003, 1003 AA Amsterdam, The Netherlands

AND

J. G. VERWER AND P. A. ZEGELING†

Centre for Mathematics and Computer Science (CWI), P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Received July 7, 1988; revised September 6, 1989

In recent years, several sophisticated packages based on the method of lines (MOL) have been developed for the automatic numerical integration of time-dependent problems in partial differential equations (PDEs), notably for problems in one space dimension. These packages greatly benefit from the very successful developments of automatic stiff ordinary differential equation solvers. However, from the PDE point of view, they integrate only in a semi-automatic way in the sense that they automatically adjust the time step sizes, but use just a fixed space grid, chosen a priori, for the entire calculation. For solutions possessing sharp spatial transitions that move, e.g., travelling wave fronts or emerging boundary and interior layers, a grid held fixed for the entire calculation is computationally inefficient, since for a good solution this grid often must contain a very large number of nodes. In such cases methods which attempt automatically to adjust the sizes of both the space and the time steps are likely to be more successful in efficiently resolving critical regions of high spatial and temporal activity. Methods and codes that operate this way belong to the realm of adaptive or moving-grid methods. Following the MOL approach, this paper is devoted to an evaluation and comparison, mainly based on extensive numerical tests, of three moving-grid methods for 1D problems, viz., the finite-element method of Miller and co-workers, the method published by Petzold, and a method based on ideas adopted from Dorfi and Drury. Our examination of these three methods is aimed at assessing which is the most suitable from the point of view of retaining the acknowledged features of reliability, robustness, and efficiency of the conventional MOL approach. Therefore, considerable attention is paid to the temporal performance of the methods.

* This work has been carried out in connection with a joint CWI/Shell project on "Adaptive Grids." For this project P. A. Zegeling has received support from the "Netherlands Foundation for the Technical Sciences" (STW), future Technical Science Branch of the Netherlands Organization for the Advancement of Research (NWO) Contract CWI55.092.

† To whom reprint requests should be addressed.

0021-9991/90 $3.00

1. INTRODUCTION

It is well known that many discretizations of time-dependent problems in partial differential equations (PDEs) can be derived by means of the following two-stage procedure. First, the space variables are discretized on a selected space mesh, mainly using finite-difference or finite-element approximations, so as to convert the PDE problem into a system of, usually stiff, ordinary differential equations (ODEs) with time as an independent variable. The discretization in time of this ODE system then yields the required fully discretized scheme. In the literature this two-stage approach is often referred to as the method of lines (MOL). With this approach in mind, several sophisticated PDE packages have been developed in recent years, notably for one-space-dimensional problems [3, 4, 11, 15, 25, 26]. These MOL packages greatly benefit from the very successful developments of automatic stiff ODE solvers. Needless to say, the development of implicit BDF codes, initiated by Gear and further improved by, among others, Hindmarsh and Petzold, is a key factor here (see [11, 24] and the references therein). Indeed, certainly for intelligent users who know their problem, Gear-type solvers have proved to be highly efficient, robust, and reliable, in that they work for a broad class of problems and usually solve the stiff ODE system under consideration in an accurate and efficient way. The experiences with MOL packages have revealed clearly that this is also true of semi-discrete PDE problems.

However, from the PDE point of view, conventional MOL packages integrate in a semi-automatic way in the sense that they adjust the time step sizes automatically, but use a fixed space grid, chosen a priori, for the entire calculation. Depending on the degree of spatial activity, such a space grid is usually equispaced or mildly nonuniform. In many cases this semi-automatic approach works very satisfactorily, notably for problems in which the solution does not exhibit a high degree of spatial activity, but also for problems where regions of rapid variation in space do not move when time evolves (stationary layers). However, for solutions possessing sharp moving spatial transitions, like travelling wave fronts or emerging boundary and interior layers, a grid held fixed for the entire calculation can be computationally inefficient, since this grid will almost certainly have to contain a very large number of nodes. In such cases, methods which attempt to adjust automatically both the space and the time step sizes are likely to be more successful in efficiently resolving critical regions of high spatial and temporal activity. Methods and codes which operate this way belong to the realm of adaptive or moving-grid methods.

Over the past several years the interest in moving-grid methods has rapidly increased. Unfortunately, very few, if any, moving-grid software packages, generally applicable up to nearly the same level of efficiency, robustness, and reliability as conventional packages, are available yet, even for the relatively simple 1D case. Admittedly, for an interesting variety of difficult example problems, various adaptive techniques have been shown to be potentially very efficient, a prominent example being the moving-finite-element method invented by Miller and his co-workers [6, 10, 16, 17, 18, 19]. However, most, of the techniques, including the

moving-finite-element method, require some form of tuning to ensure that the automatic choice of the changing space nodes is safely governed. This additional tuning is to the detriment of reliability. Experience so far has made it clear that, in general, the automatic space node selection is intrinsically difficult, in the sense that the tuning, being rather problem-dependent, does not lend itself to automation. Hence, algorithms employing moving-grid techniques usually require considerably more expertise of the user than most of the common fixed-grid algorithms in order for the best possible results in terms of efficiency, robustness and reliability to be obtained. Noteworthy, in this connection, is that the moving-grid construction, with the accompanying tuning, is often a determining factor for the computational effort spent in the time integration. Traditionally, this point has been neglected in most of the work on time-dependent problems, probably because the greater part of the development effort is spent in doing a good job in the spatial direction.

Following the philosophy of the MOL approach, this paper is devoted to an evaluation and comparison, mainly based on extensive numerical tests, of three moving-grid methods for 1D problems, viz., the finite-element method of Miller *et al.*, the method published by Petzold [23], and a method based on ideas adopted from Dorfi and Drury [8]. The two latter ones are finite-difference methods. Concerning the time integration, all these moving-grid methods can be straightforwardly combined with a stiff solver, just as in the conventional MOL approach. In the referenced papers, interesting results have been shown already, using such a type of time integrator. Our examination of the three methods, presented in this paper, is principally aimed at assessing which of the three methods is most suitable from the point of view of retaining the acknowledged features of reliability, robustness, and efficiency of the conventional MOL approach. As already indicated by the remark made above, in such an examination the moving-grid determination should be considered not only in relation to spatial solvability properties, but also in relation to the time-stepping process. Hence we shall pay considerable attention to the question of efficiency of the time-stepping process.

Briefly, the paper is arranged as follows. In Section 2 we present an outline of the three methods under consideration, proceded by some general observations on the Lagrangian approach. This approach underlies the two finite-difference methods, while also the finite-element method can be interpreted this way. This section concentrates on the semi-discretization. Section 3 is very short and deals with the numerical time integration by means of stiff BDF solvers. In Section 4 we discuss results of extensive numerical testing on a set of three test models from existing moving-grid literature. This test set includes 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. It is worth emphasizing at the outset that these problems show different solution behaviour. This is of importance with respect to our aim, which is to assess which of the three methods under consideration best enables the acknowledged features of reliability, robustness, and efficiency of the conventional MOL approach to be

realized. We are aware, of course, that experience based on a test set containing three example problems is necessarily limited. By choosing problems differing in solution behaviour, however, we are confident that our conclusions and recommendations have a much wider scope. This holds particularly true for the time integration aspect. Our conclusions and recommendations are summarized in Section 5.

To conclude this introduction we wish to emphasize that in the present paper we do not consider the extension of the methods to higher space dimensional problems. It should be acknowledged, however, that work reported by Miller, Baines, Wathen, and others contains interesting results in this direction for the moving-finite-element method. (see, e.g., [6] and the references therein). We do not know of higher space dimensional applications of the two finite-difference methods examined here.

2. Outline of the Moving-Grid Techniques

In order for this article to be read independently, we present in this section a brief outline of the main principles on which the three moving-grid methods are based. In view of the need for brevity, as well as for simplicity of presentation, this outline concentrates on the scalar form. This restriction is not essential. Concerning the automatic grid generation, the principles behind the three methods are the same as for systems and none of the three methods really distinguishes between scalar problems and systems (the necessary changes for systems are always at the implementation level, see [8, 10, 23], where applications to systems are discussed). For clarity, Section 2 deals only with the semi-discretization. We begin our outline with some general observations on the Lagrangian approach.

2.1. *The Lagrangian Approach*

Virtually all of the space mesh adapting techniques for time-dependent problems attempt to move the nodes in such a way that, in regions of high spatial activity, there is enough spatial resolution. In other words, the construction of these methods is aimed at minimizing the number of space nodes relative to a certain level of spatial accuracy. On the other hand, in most time-dependent applications large spatial gradients are accompanied by large temporal gradients, the standard example being provided by the simple running wave form $u(x, t) = w(x - ct)$. It thus is natural not only to minimize the computational effort put into the spatial discretization, but also attempt to minimize the computational effort put into the time integration. Lest we miss the obvious, on a non-moving mesh a steep wave form such as $u(x, t) = w(x - ct)$ will require standard time-stepping techniques, including the sophisticated Gear methods, to use small time steps. This is inevitable, because when on a non-moving mesh the moving front passes a grid point, the

solution at this grid point will change very rapidly. Small time steps are then necessary to retain accuracy.

The above observation naturally leads one to consider the Lagrangian approach, which is best introduced via a co-ordinate transformation. Consider the PDE problem

$$\partial u/\partial t = f(u), \qquad x_L < x < x_R, \quad t > 0, \tag{2.1}$$

where f represents a differential operator involving only spatial derivatives, e.g.,

$$\partial u/\partial t = f(u) := -\partial c(u)/\partial x + \varepsilon\, \partial^2 u/\partial x^2 + g(u), \qquad x_L < x < x_R, \quad t > 0, \quad \varepsilon > 0. \tag{2.1'}$$

The space interval is supposed to be fixed for all times $t > 0$ under consideration. Let (s, t) be new independent variables linked with the old independent variables (x, t) through a co-ordinate transformation $x = x(s, t)$. Denote $v(s, t) = u(x, t)$. Then the total derivative of u is $\partial v/\partial t = \partial u/\partial x\ \partial x/\partial t + \partial u/\partial t$ and the Lagrangian form of (2.1) reads

$$\partial v/\partial t = \partial u/\partial x\ \partial x/\partial t + f(u), \qquad s_L < s < s_R, \quad t > 0, \tag{2.2}$$

and that of (2.1'),

$$\partial v/\partial t = \partial u/\partial x\ \partial x/\partial t - \partial c(u)/\partial x + \varepsilon\, \partial^2 u/\partial x^2 + g(u), \qquad s_L < s < s_R, \quad t > 0. \tag{2.2'}$$

Note that $\partial u/\partial t$ measures the changes of u as a function of t at a fixed x value (Eulerian description) and $\partial v/\partial t$ at a fixed s value (Lagrangian description). Thus the basic idea of the Lagrangian approach is that in the variables (s, t) the problem should be easier to handle numerically than in the original pair (x, t). Ideally, in the new variables any rapid transition should be absent; we can then take acceptable step sizes in the time direction while using a coarse uniform s-grid in space. A suitable nonuniform x-grid then exists according to the change of variables $x = x(s, t)$.

In classical Lagrangian methods, as are being applied successfully to some types of fluid-flow problems, the movement of the nodes is attached, in an a priori manner, to a physically motivated, specific flow quantity. For example, for a problem like (2.1') it makes sense to attach the movement of the nodes to the convection term $\partial c(u)/\partial x$, i.e., to choose $\partial x/\partial t = dc(u)/du$ so as to obtain the parabolic equation $\partial v/\partial t = \varepsilon\, \partial^2 u/\partial x^2 + g(u)$ (in a moving reference frame). The rationale behind this choice is that parabolic problems without large first-order terms usually possess smoother solutions and thus are less difficult to solve numerically. Of course, the numerical realization of the prescription $\partial x/\partial t = dc(u)/du$ involves its own difficulties, but these are usually surmountable.

Because we aim at application to a wide variety of problems we require that the transformation be based on a general "systematic rule," e.g., spatial equidistribution. In fact, the choice of this "systematic rule" determines to a great extent the

moving-grid method under consideration. This will be illustrated quite clearly in the remaining sections. Here we wish to point out that it is not always possible to smooth the solution, through the co-ordinate transformation, in space and in time simultaneously. This obviously depends on the nature of the solution sought, which can be nicely illustrated by examining Problem I of Section 4 (cf. [27, Section 5.3]). Let us consider its solution near the left boundary, while the steep front is forming (the ignition phase). Assuming a uniform grid at the initial line (a choice suggested by the constant initial solution $u(x, 0) = 1$), the derivatives $\partial x/\partial t$ of many of the trajectories should be negative in order for the required refinement in the region of the steep front to be obtained, which is in accordance with the objective of smoothing the problem in space. However, during the formation of the front, $\partial u/\partial x < 0$ and $\partial u/\partial t > 0$. It then follows immediately that $\partial v/\partial t > \partial u/\partial t$, violating the objective of getting a smoother problem in time. Most Lagrangian-type methods do underly the first objective through a co-ordinate transformation based on spatial equidistribution properties. Spatial equidistribution forces nodes to migrate to regions of high spatial activity. So, during the formation of the front, for the present combustion problem these methods offer not benefit as far as the time stepping is concerned. Once the front is formed and starts to propagate, both smoothing objectives are fulfilled if the transformation underlies spatial equidistribution, because then $\partial x/\partial t > 0$ and still $\partial u/\partial x < 0$ and $\partial u/\partial t > 0$. Any simple travelling wave form $u(x, t) = f(x - ct)$ is a trivial solution, in this respect, provided the grid trajectories satisfy $\partial x/\partial t = -c$. Interestingly, the Lagrangian approach followed by Petzold [23] underlies the second objective. This approach, originally due to Hyman [13], is basically aimed at finding those trajectories along which the time rate of solution change is minimized, that is, $\partial v/\partial t < \partial u/\partial t$. However, during the formation of the front in the present combustion example , this must imply that, in the front region, $\partial x/\partial t > 0$, which means that points are moved away from the front, as thus the first objective is violated. This is contrary to the desired aim; however, Petzold's algorithm has a built-in regridding step which corrects this deficiency (see the next section). For this method it also holds that, once the front is formed and starts to propagate, both smoothing objectives are fulfilled.

The two finite-difference methods we examine are based on the standard, central semi-discretization of the above Lagrangian form (2.2). More precisely, completely in line with the common MOL approach, consider numerical, continuous-time trajectories

$$x_L = X_0 < \cdots < X_i(t) < X_{i+1}(t) < \cdots < X_{N+1} = x_R \qquad \text{for} \quad 0 \leqslant t \leqslant t_{\text{end}}, \tag{2.3}$$

with the associated grid functions

$$U = [U_1, ..., U_N]^T, \qquad X = [X_1, ..., X_N]^T.$$

Thus, U represents the semi-discrete approximation to the PDE solution u restricted to the moving space grid X and is the solution of the ODE system

$$U_i' = X_i'[(U_{i+1} - U_{i-1})/(X_{i+1} - X_{i-1})] + F_i(U, X), \qquad 1 \leqslant i \leqslant N, \tag{2.4}$$

where the symbol $'$ denotes differentiation to time (U_i' denotes the semi-discrete total derivative) and the operator F stands for the difference operator replacing the differential operator f on the grid. For example, the right-hand side function of (2.1′) is approximated at grid point i, $1 \leqslant i \leqslant N$, by

$$F_i(U, X) = -\{[c[U_{i+1}) - c(U_{i-1})]/[X_{i+1} - X_{i-1}]\} + \varepsilon\{((U_{i+1} - U_i)/(X_{i+1} - X_i) - (U_i - U_{i-1})/(X_i - X_{i-1}))/(0.5(X_{i+1} - X_{i-1}))\} + g_i(U).$$

In the discussion to follow, we neglect the treatment of boundary conditions, since these are dealt with in the usual way. We recall that, for convection–diffusion problems with steep gradient or near-shock behaviour, the use of central differencing of first-order terms is not ideal and one would probably consider stable upwind or flux-corrected approximations. In this paper, the central approximation is used, since it facilitates comparisons between the three methods (the finite-element method uses standard piecewise linear basis functions) and because it represents a severe test for a good moving grid $X(t)$. Any deviation from an ideal Lagrangian grid movement, assuming this exists, will soon result in unphysical, oscillatory solutions. As already indicated above, the definition of $X(t)$ is highly important and determines to a great extent the complete moving-grid method.

2.2. *Method I*

Method I is the finite-difference moving-grid method proposed by Petzold [23] (version A). Each time step consists of two computational stages: a moving Lagrangian step, involving the application of a stiff ODE solver to an augmented semi-discrete system, followed by a second (regridding) stage in which a redistribution of points at the forward time level is carried out through a De Boor-type equidistribution algorithm. Both are equally important for the application of the method. However, in contrast to most methods, grid points are not necessarily moved in the desired direction of high spatial activity. Loosely speaking, one of the purposes of the regridding stage is to correct this deficiency.

The Semi-discrete System

We begin our outline with the derivation of the (augmented) semi-discrete system, which consists of the equations of system (2.4) together with grid equations for the implicit determination of the unknown grid X. Consider the Lagrangian form (2.2) where, for convenience of notation, u' and x' now denote the derivatives $\partial v/\partial t$ and $\partial x/\partial t$, respectively. The underlying transformation, which is originally due to Hyman [13], is chosen to mimimize in a certain sense the total derivative u'. This is done by selecting x' such that

$$(u')^2 + \alpha(x')^2 = (\partial u/\partial t + x'\,\partial u/\partial x)^2 + \alpha(x')^2$$

is minimized, where $\alpha \geqslant 0$ is a real number. Differentiation to x' and equating to

zero yields the differential expression $u'\,\partial u/\partial x + \alpha x' = 0$, which can be written as the single ODE,

$$x' = (-\partial u/\partial t\ \partial u/\partial x)/(\alpha + (\partial u/\partial x)^2),$$

with x as a dependent and t as an independent variable, provided u, $\partial u/\partial t$, and $\partial u/\partial x$ are known functions of x. For given $x(0)$ on the initial space interval, the solution of this ODE defines the trajectory $x(t)$ $(t \geqslant 0)$ along which the rate of change of u; that is,

$$u' = (\alpha\ \partial u/\partial t)/(\alpha + (\partial u/\partial x)^2)$$

is minimized in the above sense. It is hereby tacitly assumed that the above ODE is uniquely solvable. The parameter α serves to regularize the transformation. For $\alpha = 0$ we have $u' = 0$, which in general cannot be a solution. Observe that, in regions where $(\partial u/\partial x)^2$ is negligible relative to α, the transformation has no effect. The travelling wave form $u(x, t) = w(x - ct)$ nicely shows the idea behind this transformation. For this solution we have

$$x' = (x(\partial w/\partial x)^2)/(\alpha + (\partial w/\partial x)^2), \qquad u' = (-\alpha c(\partial w/\partial x)^2)/(\alpha + (\partial w/\partial x)^2)$$

and for $\alpha = 0$ the grid point $x(t)$ moves with the wave speed c. Recall, however, that grid points are not always moved in the desired direction.

Hence, the transformation employed leads to the grid equation

$$u'\,\partial u/\partial x + \alpha x' = 0, \qquad s_L < s < s_R, \quad t > 0.$$

When combined with (2.2), it can be solved for the unknowns u and x. The grid equation is written in this form to avoid ill-conditioning problems in the numerical solution process [23]. At this stage it is pointed out that in actual application the new variable s is not used explicitly; that is, computations will always be performed in terms of the original variables (x, t). Note that explicit use of s would require that its bounds be properly defined, which we have not done. Like (2.2), this grid equation is spatially discretized on the grid (2.3) so that we obtain

$$U_i'[(U_{i+1} - U_{i-1})/(X_{i+1} - X_{i-1})] + \alpha X_i' = 0, \qquad 1 \leqslant i \leqslant N. \tag{2.5}$$

Equations (2.4), (2.5) form the augmented, semi-discrete system and define the unknown variables U and X.

In addition to the regularization term $\alpha x'$, the grid computation needs an extra regularization to prevent neighbouring grid points from crossing. Note that, even when the single ODE for the exact grid trajectory is uniquely solvable, the grid trajectories for a set of given initial points may approach each other arbitrarily closely. Petzold [23] has suggested that, instead of (2.5),

$$U_i'[(U_{i+1} - U_{i-1})/(X_{i+1} - X_{i-1})] + \alpha X_i' + \lambda((X_i' - X_{i-1}')/(X_i - X_{i-1})^2 - (X_{i+1}' - X_i')/(X_{i+1} - X_i)^2) = 0, \qquad 1 \leqslant i \leqslant N, \tag{2.6}$$

should be used, where $\lambda > 0$ is the second regularization parameter. This form results when

$$(u_i')^2 + \alpha(x_i')^2 + \lambda((x_i' - x_{i-1}')^2/(x_i - x_{i-1})^2 - (x_{i+1}' - x_i')^2/(x_{i+1} - x_i)^2) \quad (2.7)$$

is minimized. This regularization term is related to the "internodal viscosity" term of the moving-finite-element method (see Section 2.4). The use of this type of regularization is based on heuristic considerations. If neighbouring points tend to approach each other very closely, the denominators in (2.6) will eventually decrease beyond the level needed to let the regularization term dominate the entire expression. If this happens, the minimization procedure will result in nearly equal neighbouring grid velocities, with the effect that, when time evolves, neighbouring points are prevented from approaching further.

Necessarily, the regularization is problem-dependent and in actual application there is no guarantee that points will not cross. On the other hand, at sufficiently large λ the grid becomes non-moving. Hence, if λ is chosen too large, it may happen that points are forced to stay apart too much so that locally the grid is not fine enough to resolve anticipated small-scale structures. Following [23], we have used throughout the values $\alpha = 1$, $\lambda = 0.2$. Needless to say other choices of regularization terms are conceivable. It should be emphasized, though, that it is not easy, if possible at all, to find an optimal regularization. It is noteworthy that regularization always has some smoothing effect on the grid trajectories, which is desirable for the time integration. Hence, regularization not only influences the spatial solvability performance of the moving-grid method, but also the performance of the stiff solver.

In order to bring the augmented semi-discrete system (2.4), (2.6) into a more compact form, we introduce the vector $Y = [U_1, X_1, ..., U_i, X_i, ..., U_N, X_N]^T$. The semi-discrete system then takes the linearly implicit form

$$A(Y)\,Y' = G(Y) \qquad \text{for} \quad t > 0 \qquad \text{and} \qquad Y(0) \text{ given}, \quad (2.8)$$

where $A(Y)$ is block tridiagonal and the $(2i-1)$th and the $(2i)$th element of the vector-valued function G are given by

$$G_{2i-1}(Y) = F_i(U, X), \quad G_{2i}(Y) = 0 \qquad (1 \leqslant i \leqslant N). \quad (2.9)$$

Inspection of the matrix $M = -A$ reveals that for any vector Y its symmetric part $(M + M^T)/2$ is negative definite, so that, according to the known property that the real part of any eigenvalue is smaller than or equal to the maximum eigenvalue of $(M + M^T)/2$, the matrix A is non-singular. This means that system (2.8) is a genuine, stiff ODE system. Even when grid points cross, the matrix remains non-singular. This is handy because it means that crossing need not be fatal. More precisely, after each (modified) Newton iteration within an implicit moving integration step with the stiff solver, a check on crossing is made. If crossing is detected, the current step is interrupted and redone with a smaller step size.

The Regridding Step

The above transformation is interesting in itself, because it provides a smoother problem in time. This will be beneficial for the numerical integration process. A disadvantage is that this transformation may not necessarily move the grid points in the direction of high spatial activity. To overcome this deficiency, an intermediate regridding is carried out, in principle after every successful moving integration step.

Suppose the moving step with the stiff solver has delivered the numerical (vector) values U^n, X^n at the nth forward time level. Then, by application of a De Boor-type regridding algorithm, which uses U^n, X^n for input, a second level n grid is determined. This new grid, Z^n, say, satisfies (approximately)

$$\Delta z\,|\partial u/\partial x| + (\Delta z)^2\,|\partial^2 u/\partial x^2| = \text{constant}. \tag{2.10}$$

It would lead us too far to discuss the implemented De Boor algorithm in detail. Here we only remark that we keep the number of moving points fixed, whereas Petzold [23] adapts the number of moving points so that (approximately)

$$\Delta z\,|\partial u/\partial x| + (\Delta z)^2\,|\partial^2 u/\partial x^2| \leqslant \text{specified tolerance},$$

while the number of points is the smallest number needed to satisfy this inequality. We have decided to work with a fixed, given number of moving points for comparison with the other two methods. In conclusion, Z^n equidistributes (2.10), which has the effect that points are concentrated in regions of high spatial activity. This alleviates the deficiency mentioned above. Because the grid Z^n will normally differ from X^n, it is necessary to interpolate from X^n onto Z^n prior to the next moving integration step. This is done via the "dual reconnecting grid" approach, which is a compromise between choosing the best grid and avoiding needless interpolations. Briefly, the idea is as follows. Z^n divides the space interval into zones. Each zone is allowed to contain one point from X^n. If a zone contains just one point, no interpolation takes place. If a zone is empty, a point is added and a (monotone) interpolation is carried out. If there are more points from X^n in a zone, points are deleted. Grid points at the edge of zones which are too close to other points are moved apart. In this way the final grid to be used for the next step is created. Hence, on most time steps only a few interpolations are carried out (and eventually none). This is of importance, since interpolation usually damages the accuracy a little. Another attractive feature of the dual reconnecting grid approach is that points can be added and deleted locally. This is advantageous when locally the solution undergoes sudden rapid changes (birth of new layers).

When considered on its own, the idea of intermediate regridding is interesting because, as a sort of added bonus, it provides the possibility of more direct control on the placement of nodes through equidistribution (and connected herewith heuristic spatial error monitoring based on (2.16)). One could say that the intermediate regridding step makes the regularization less critical, though regularization

should not be omitted. A considerable disadvantage of regridding is that it necessitates interpolation and that it interrupts the time-stepping process. Frequent interpolation may damage the accuracy considerably, while the interruption of the time-stepping process causes a restart situation for the stiff solver (in our case a BDF solver). In other words, after a regridding the Jacobian matrix is updated and the integration is continued with the implicit Euler method on the newly chosen grid (with that step size that would have been used on the next step had there been no restart). The inevitable consequence is that, when there are many genuine regriddings, the solver does not get the chance of switching to a higher order formula, which no doubt is detrimental when high accuracy in time is needed. It is clear that this situation is somewhat in contradiction with the MOL approach and that there is room for some improvement here [23].

2.3. *Method II*

Method II is also a finite-difference method based on the semi-discrete Lagrangian from (2.4). The main ideas of moving the grid are derived from Dorfi and Drury [8]. An implicit equation for $X(t)$ is used which underlies a spatial equidistribution transformation based on an arc-length monitor function. An importance feature of Method II is that the grid movement is regularized by employing a smoothing technique in both space and time. The spatial grid smoothing ensures that the ratio of adjacent grid intervals is restricted, thus controlling clustering and grid expansion. The temporal grid smoothing ensures a smooth progression of $X(t)$ by preventing the points from responding too quickly to current solution gradients. This is highly desirable for efficient numerical time stepping.

The Semi-discrete System

We shall derive the semi-discrete grid equations for the implicit determination of the moving grid $X(t)$. Let us first recall the idea of the spatial equidistribution transformation which is used in Method II. Hence, the theoretical co-ordinate transformation supposed in Eq. (2.2) is now of the form

$$s(x, t) = \int_{x_L}^{x} M(\xi, t)\, d\xi/\eta(t), \qquad \eta(t) = \int_{x_L}^{x_R} M(\xi, t)\, d\xi,$$

where $M(x, t)$ is a chosen monitor function which should reflect space dependence of the PDE solution. The spatial equidistribution of this monitor function is enforced by dividing the interval $0 \leqslant s \leqslant 1$ into $N+1$ equal parts. Through the inverse transformation, the N theoretical grid trajectories $x_i(t) = x(i/N, t)$, $t \geqslant 0$ $(1 \leqslant i \leqslant N)$, where x_0, x_{N+1} are the given boundaries x_L and x_R, respectively, then satisfy the equidistribution relation

$$\int_{x_i}^{x_{i+1}} M(\xi, t)\, d\xi = \eta(t)/N \qquad (0 \leqslant i \leqslant N).$$

Consequently, in regions where M will be large, the grid trajectories will become close and vice versa. By applying the mid-point quadrature rule and inserting semi-discrete variables, at the semi-discrete level this equidistribution relation is taken to be

$$(X_{i+1} - X_i)\, M_i = \text{constant} \qquad (0 \leqslant i \leqslant N), \tag{2.11}$$

where M_i now represents the semi-discrete monitor value at the mid-point of the ith sub-interval $[X_i, X_{i+1}]$. Following Dorfi and Drury, we use the arc-length monitor

$$M_i = \{1 + (U_{i+1} - U_i)^2/(X_{i+1} - X_i)^2\}^{1/2},$$

which has the property of placing grid points along uniform arc-length intervals and gives good point placement at the "lip" of a shock. Of course, other choices of M are conceivable. Because M is positive, solution values X_i of (2.11) cannot cross. For a discussion of monitor functions and equidistribution, see, for example, Pereyra and Sewell [22], Furzeland [9], Carey and Dinh [5].

By elimination of the constant in (2.11), a set of N semi-discrete grid equations for the implicit determination of the moving grid X is obtained

$$(X_i - X_{i-1})\, M_{i-1} = (X_{i+1} - X_i)\, M_i \qquad (1 \leqslant i \leqslant N). \tag{2.12}$$

Combining these with the semi-discrete PDE equations (2.4) yields the (augmented) semi-discrete problem for the unknown grid functions U and X. However, as mentioned previously, Dorfi and Drury regularize the grid movement by performing a smoothing technique both in space and time. This amounts to modifying the grid equation system (2.12). We shall first describe their modification for the spatial grid smoothing.

For this purpose we introduce the point concentrations

$$n_i = 1/(X_{i+1} - X_i) \qquad (0 \leqslant i \leqslant N). \tag{2.13}$$

Using these variables, the grid equation system (2.12) is written in the form

$$n_{i-1}/M_{i-1} = n_i/M_i \qquad (1 \leqslant i \leqslant N) \tag{2.14}$$

and the spatial grid smoothing is then carried out by replacing the point concentrations in this system by their smoothed (numerically diffused) counterparts

$$\tilde{n}_i = n_i - \kappa(\kappa + 1)(n_{i+1} - 2n_i + n_{i-1}), \qquad \kappa > 0, \tag{2.15}$$

to obtain the new grid equation system

$$\tilde{n}_{i-1}/M_{i-1} = \tilde{n}_i/M_i \qquad (2 \leqslant i \leqslant N-1). \tag{2.16}$$

Neglecting the influence of the boundaries, it can be shown that this filtering procedure is equivalent to a certain smoothing procedure for the monitor function [8], thus ensuring that the adjacent point concentrations are restricted such that

$$\kappa/(\kappa+1) \leqslant n_{i-1}/n_i \leqslant (\kappa+1)/\kappa. \tag{2.17}$$

This spatial smoothing can also be achieved by "padding" the monitor function [14, 9], but this approach is not recommended here, since within the MOL framework the implicit coupling X and U then varies at each time step. For a given N and a given monitor function distribution, the choice of κ determines the minimum and maximum interval lengths. The monitor function determines the relative shape of the X_i distribution, and κ and N determine the absolute level of clustering [8]. In actual application, a value of κ of about 1 or 2 is recommended. This yields modestly graded space grids. In our experiments we have used $\kappa = 2$. The value of κ plays an important role in controlling space discretization errors on non-uniform grids (see, for example, [9]).

System (2.16) must be completed with boundary conditions. Following [8], at the boundaries the "concentration gradients" are set to zero,

$$n_0 = n_1 \qquad \text{and} \qquad n_{N-1} = n_N. \tag{2.18}$$

Note that the use of the grid equations (2.16), (2.18) introduces a five-point coupling in X. Needless to say, this slightly increases the computational costs of the method (per step).

The temporal grid smoothing described next replaces the set of algebraic equations (2.16) by the following set of differential equations

$$(\tilde{n}_{i-1} + \tau\, d\tilde{n}_{i-1}/dt)/M_{i-1} = (\tilde{n}_i + \tau\, d\tilde{n}_i/dt)/M_i, \qquad \tau \geqslant 0 \quad (2 \leqslant i \leqslant N-1), \tag{2.19}$$

again with boundary conditions (2.18). This system is constructed as follows. Consider the monitor function values $M(t)$ occurring in Eq. (2.16) (for convenience of notation we suppress the lower index i). The temporal grid smoothing hinges on the replacement of $M(t)$ by

$$R(t) = \int_0^\infty M(t - \sigma\tau)\, e^{-\sigma}\, d\sigma, \qquad \tau \geqslant 0,$$

where $M(t)$ is now thought of as being defined on the semi-infinite interval $[-\infty, t]$. In actual application, the extension to the interval $[-\infty, 0]$ is neglected. This is allowed due to the presence of the exponential damping factor and the fact that the parameter τ is supposed to be rather small (the choice $\tau = 0$ yields $R(t) = M(t)$). By partial integration the differential form

$$M(t) = R(t) + \tau\, dR(t)/dt$$

can be recovered, which is used to construct (2.19). More precisely, the numerically diffused point concentration values $\tilde{n}(t)$ of Eq. (2.16) are now taken proportional to $R(t)$, rather than to $M(t)$. Let $c(t)$ be the proportionality constant, that is, $R(t) = c(t)\,\tilde{n}(t)$. Substitution into this differential form gives

$$M(t) = c(t)(\tilde{n}(t) + \tau\, d\tilde{n}(t)/dt) + \tau\tilde{n}(t)\, dc(t)/dt.$$

If we then neglect the time dependence of the proportionality constant $c(t)$, and subsequently eliminate it, the grid equation system (2.19) is recovered.

The motivation behind the use of the monitor function $R(t)$, which is "averaged in time," is that, when the grid movement is attached to $R(t)$ rather than to $M(t)$, it is prevented from adjusting immediately to the new monitor values. Instead, the use of $R(t)$ forces the grid to adjust over a time interval of length τ from old to new monitor values, i.e., the parameter τ acts as a delay factor. The aim of this approach is to avoid temporal oscillations in the original grid trajectories defined by (2.16). These oscillations are typical for grids generated via numerical spatial equidistribution techniques. When applied to solutions with very large gradients, relatively large errors occur with these techniques. Needless to say, for the numerical time integration a smooth grid $X(t)$ is highly desirable; otherwise too many Jacobian evaluations are needed when an implicit solver is applied.

Albeit heuristic in nature, there is no doubt that the temporal grid smoothing procedure is of importance. The choice of the delay factor τ requires some expertise but, in our experience, this is not too critical. Increasing τ too much results in a grid that lags too far behind any propagating wave or shock. Note that, for sufficiently large values of τ, a non-moving grid results. Trivially, too small values for τ render no effect. In practice it makes sense to choose τ close to the anticipated temporal step size value such that, over one or a few time levels, the influence of past monitor values is felt. The stabilizing effect of τ is similar to that of the damping factor λ introduced in Coyle, Flaherty, and Ludwig [7].

To sum up, the semi-discrete grid equations (2.19) with the boundary conditions (2.18) determine the continuous-time moving grid $X(t)$. Of importance to note is that we work with the $2N$ unknowns U_i, X_i $(1 \leqslant i \leqslant N)$ and in our implementation the point concentration derivatives that occur in (2.15), (2.19) are replaced by

$$dn_i/dt = -(X'_{i+1} - X'_i)/(X_{i+1} - X_i)^2.$$

More specifically, in the numerical integration $U_i(t)$ and $X_i(t)$ are computed with the same integration formulas, which is different from the implementation in [8] (see formula (10)). Consequently, in our case the ith equation of system (2.19) couples the nodal points

$$X_{i+2}, \quad X_{i+1}, \quad X_i, \quad X_{i-1}, \quad X_{i-2},$$

with the nodal point velocities

$$X'_{i+2}, \quad X'_{i+1}, \quad X'_i \quad X'_{i-1}, \quad X'_{i-2},$$

and the solution values

$$U_{i-1}, \qquad U_i, \qquad U_{i+1}.$$

A little inspection reveals that the vector version of the final augmented semi-discretized system of ODEs can be brought to a linearly implicit ODE form with a known bandwidth in a similar way to that used for method I (cf. (2.8)),

$$A(Y)\,Y' = G(Y) \qquad \text{for} \quad t>0 \quad \text{and} \quad Y(0) \quad \text{given.} \tag{2.20}$$

This system will be integrated in time as described in Section 3. To conclude this section we remark that in none of our experiments with Method II have we used the initial grid generation algorithm proposed by Dorfi and Drury. We shall specify our initial grids with the numerical examples in Section 4.

2.4. *Method III*

Method III is the moving-finite-element method introduced by Miller *et al.* [16, 17]. This method also generates a system of continuous-time ODEs for mesh points and numerical approximations in these moving points. The grid movement is regularized by using penalty functions.

The Semi-discrete System

Consider the continuous-time grid X introduced in (2.3) with unknown components. On such a grid, the moving-finite-element method approximates the solution $u(x, t)$ of problem (2.1) by an expansion (summation from 1 to N)

$$U(x, t) = \sum_i U_i(t)\, l_i(x, X(t)), \tag{2.21}$$

where l_i are the standard piecewise linear basis functions that depend on the nodal positions X_i and U_i are the amplitudes of the approximate solution $U(x, t)$ at the corresponding nodal positions. Differentiating this expression with respect to t gives, after some elementary calculations

$$U_t = \sum_i U_i' l_i + X_i' b_i, \tag{2.22}$$

where the b_i are piecewise linear discontinuous basis functions with the same support as l_i. We have

$$b_i(x) = \begin{cases} -m_i l_i & \text{for} \quad X_{i-1} \leqslant x \leqslant X_i, \\ -m_{i+1} l_i & \text{for} \quad X_i \leqslant x \leqslant X_{i+1}, \\ 0 & \text{elsewhere,} \end{cases}$$

where $m_i = (U_i - U_{i-1})/(X_i - X_{i-1})$ is the slope of the semi-discrete approximation $U(x, t)$ on $[X_{i-1}, X_i]$. It is of interest to note that (2.22) is akin to the Lagrangian form (2.2); U_i' plays the role of the Lagrangian derivative $\partial v/\partial t$ and the nodal velocity X_i' that of $\partial x/\partial t$ (see [2, 9, 21] for a discussion of the Lagrangian nature of Method III).

The equations determining the semi-discrete unknowns U_i and X_i are now obtained in the standard Galerkin way by minimizing the square of the L_2 norm of the residual $R(U) = U_t - f(U)$ with respect to U_i' and X_i'. This gives a system of $2N$ equations in the $2N$ unknowns U_i, X_i (boundary conditions are incorporated in the standard way):

$$\sum_j \langle l_i, l_j \rangle U_j' + \langle l_i, b_j \rangle X_j' = \langle l_i, f(U) \rangle, \qquad 1 \leqslant i \leqslant N, \tag{2.23a}$$

$$\sum_j \langle b_i, l_j \rangle U_j' + \langle b_i, b_j \rangle X_j' = \langle b_i, f(U) \rangle, \qquad 1 \leqslant i \leqslant N, \tag{2.23b}$$

where $\langle \cdot, \cdot \rangle$ denotes the usual inner product. Assuming zero velocities, the first equation is readily recognized as the standard, semi-discrete Galerkin equation. The second equation originates from the additional minimization with respect to the nodal velocities. Using the linear forms for l_i and b_i, the inner products on the left-hand side may be evaluated to give, respectively,

$$\begin{aligned} &\tfrac{1}{6}[\Delta X_i U_{i-1}' + 2(\Delta X_i + \Delta X_{i+1}) U_i' + \Delta X_{i+1} U_{i+1}'] \\ &\quad - \tfrac{1}{6}[\Delta U_i X_{i-1}' + 2(\Delta U_i + \Delta U_{i+1}) X_i' + \Delta U_{i+1} X_{i+1}'] \\ &\quad = \langle l_i, f(U) \rangle, \qquad 1 \leqslant i \leqslant N, \end{aligned} \tag{2.24a}$$

$$\begin{aligned} &- \tfrac{1}{6}[\Delta U_i U_{i-1}' + 2(\Delta U_i + \Delta U_{i+1}) U_i' + \Delta U_{i+1} U_{i+1}'] \\ &+ \tfrac{1}{6}[m_i \Delta U_i X_{i-1}' + 2(m_i \Delta U_i + m_{i+1} \Delta U_{i+1}) X_i' + m_{i+1} \Delta U_{i+1} X_{i+1}'] \\ &= \langle b_i, f(U) \rangle, \qquad 1 \leqslant i \leqslant N, \end{aligned} \tag{2.24b}$$

where $\Delta X_i = X_i - X_{i-1}$, etc. Using the vector notation $Y = [U_1, X_1, ..., U_i, X_i, ..., U_N, X_N]^T$, we thus arrive at the continuous-time, semi-discrete moving-finite-element system

$$A(Y) Y' = G(Y) \qquad \text{for} \quad t > 0 \qquad \text{and} \qquad Y(0) \text{ given}, \tag{2.25}$$

where $A(Y)$ is a block tridiagonal matrix and $G(Y)$ is given by

$$G(Y) = (\langle l_1, f(U) \rangle, \langle b_1, f(U) \rangle, ..., \langle l_N, f(U) \rangle, \langle b_N, f(U) \rangle)^T.$$

The matrix $A(Y)$ contains only quantities from the left-hand sides of (2.24), which are related to the discretization of $\partial u/\partial t$ on the moving mesh (cf. (2.8), (2.20)). What remains now is to integrate this ODE system numerically to obtain the required fully discretized solution.

The moving-finite-element method has aroused considerable interest yet at the same time has been subject to criticism because of its complexity and the inherent problems of parallelism and points drifting extremely close together. Parallelism occurs when the gradients of U on adjacent cells, say m_i and m_{i+1}, become equal. The $(i+1)$th column of A is then equal to m_i times the ith column, so that the mass matrix A becomes singular. When nodes drift extremely close together, the mesh may become tangled or nodes may even cross in the numerical integration process. Miller [18] suggests that these two problems can be overcome by introducing regularization terms (penalty functions) in the residual minimization. Instead, using $R(U)$ alone, the minimization is thus carried out for

$$\langle R(U), R(U)\rangle + \sum_j (e_j \Delta X_j' - S_j)^2,$$

where

$$e_i^2 = C_1^2/(\Delta X_i - d), \qquad e_i S_i = C_2^2/(\Delta X_i - d)^2, \tag{2.26}$$

with C_1, C_2, and d small, user-chosen constants. In particular, d serves as a user-defined minimal node distance. The modifications involved are only made to the mesh point equations (2.23b) and the combined effect is to add

$$e_i^2 \Delta X_i' - e_{i+1}^2 \Delta X_{i+1}' \qquad \text{and} \qquad e_i S_i - e_{i+1} S_{i+1}$$

to the left- and right-hand sides, respectively. The e-terms serve to avoid parallelism. It can be shown that the addition of these terms renders the mass matrix A diagonally dominant [18], and thus regular. They represent a form of "internodal" viscosity, since they penalize relative motion between nodes and, provided the penalty is sufficiently large to take over before the mass matrix becomes numerically singular, result in the degenerate nodes being carried along with the rest of the solution. The e-terms do prevent node overtaking in a dynamical way, since the internodal viscosities become infinite as Δx tends to zero; however, over longer time intervals, degenerate nodes (those caught in straight line segments where they are unneeded) may still slowly drift together. The S-terms, sometimes called internodal spring forces, serve to prevent this long-term numerical drift.

As for any other method, the regularization is somewhat heuristic and necessarily problem-dependent. For example, if C_1 is chosen too large, the grid movement is restricted ($C_1 \approx \infty$ gives a non-moving grid) with the result that there may not be sufficient refinement in regions of large spatial activity (a typical phenomenon is then that the grid moves slower than a front region). On the other hand, if C_1 is too small, the mass matrix A may become numerically singular. Also of great importance is that the minimal node distance d be small enough in relation to the anticipated small-scale structure. However, too small values of d and C_2 may allow numerical errors to lead to near node overtaking (or even worse), which is a source

of severe numerical difficulties in the time integration, even for the most robust stiff solver. When nodes drift extremely close together, the sets of nonlinear algebraic equations to be solved at each time step are likely to become badly conditioned. This hampers the Newton iterative process and results in a higher number of iterations and Jacobian updates than in the conventional MOL application. It is our experience that Method III is rather sensitive in this respect. We shall illustrate this extensively in the discussion of the numerical experiments.

The gradient-weighted method of Miller [18, 19] attempts to automate part of the regularization needed. In this method, the common L_2 minimization is replaced by a geometrically weighted L_2 minimization (based upon the motion of the graph in the normal direction) which serves to "de-emphasize" the steep portions of the solution. The aim is thus to avoid extreme clustering of points in regions with large gradients, something that is easily invoked by the ordinary L_2 minimization. The anticipated side effect of this weighting is that the regularization then becomes less critical. The code GWMFE1DS developed by Carlson and Miller is based on this gradient-weighted method. To gain some experience with this method, we have undertaken additional experiments with Problems I and II of Section 4 using GWMFE1DS. No improvement over the results obtained with our own MFE implementation was perceived, either in the grid positioning, or in the time stepping efficiency. Therefore we have decided to presents results only for our own MFE implementation. It is fair to point out, however, that we have limited experience with GWMFE1DS and that GWMFE1DS and our own MFE implementation use two entirely different integrators, which obviously prevents us from drawing definite conclusions concerning the merits of the gradient weighting, at least when discussing temporal efficiency aspects.

Miller and Carlson (Miller [20]) report successful GWMFE1DS trials on all three test problems I–III of Section 4 with their standard settings of a single regularization coefficient C_1^2. Their solutions seem to be extremely accurate with 40 nodes and still quite accurate with only 20 nodes. However, the running times seem to be somewhat excessive due to large numbers of time steps. It is conjectured that part of this problem may be due to their DIRK2 (diagonally implicit Runge-Kutta of order 2) stiff ODE solver and that if DIRK2 is replaced by a sophisticated, higher order BDF code, like DASSL or the code within the SPRINT package (see Section 3), the temporal efficiency may increase substantially. We plan to examine this in the near future. We have not attempted to incorporate such an examination in the research reported here, since the gradient weighted MFE leads to an ODE system (of the form (2.25)) that cannot be integrated with an existing stiff solver without breaking into the code. A few modifications dealing with the local error estimation and the nonlinear equation handling in the Newton process are desirable; in particular, an important improvement for both robustness and efficiency is to precondition the residuals of the ODE system [20]. Details and results will be reported later.

At this place we should also mention that the explicit time approach advocated by Baines, Wathen, and their co-workers (see [2] and the references contained in

[6]) is aimed at avoiding the necessity of regularization with the accompanying difficulties. However, while these explicit techniques work very successfully on purely first-order hyperbolic problems, they obviously suffer from the explicit time step restriction when applied to parabolic problems, including those of the diffusion–convection type, even with little diffusion. Therefore we consider explicit techniques as less feasible for use in a general-purpose MOL algorithm.

Finally, we list some of the inner products that are needed to handle our test problems (g represents a nonlinear source function; cf. [10]):

$$\langle l_i, U_{xx}\rangle = m_{i+1} - m_i,\ \langle b_i, U_{xx}\rangle = -(m_{i+1} - m_i)(m_{i+1} + m_i)/2,$$

$$\langle l_i, -UU_x\rangle = -\Delta U_i(U_i/9 + U_{i-1}/18) - \Delta U_{i+1}(U_i/9 + U_{i+1}/18),$$

$$\langle b_i, -UU_x\rangle = m_i\,\Delta U_i(U_i/3 + U_{i-1}/6) + m_{i+1}\,\Delta U_{i+1}(U_i/3 + U_{i+1}/6),$$

$$\langle l_i, g(U)\rangle = [g((U_{i-1} + U_i)/2)\,\Delta X_i + g(U_{i+1} + U_i)/2)\,\Delta X_{i+1}]/2,$$

$$\langle b_i, g(U)\rangle = -[m_i\,g((U_{i-1} + U_i)/2)\,\Delta X_i + m_{i+1}\,g((U_{i+1} + U_i)/2)\,\Delta X_{i+1}]/2.$$

3. The Numerical Time Integration

For the numerical time integration of the three derived semi-discrete systems (2.8), (2.20), and (2.25), we have used two existing stiff Gear solvers. All results for Method I have been obtained with the original (version A) source code of Petzold's own BDF code DASSL. The software implementing Methods II and III has been prepared by ourselves. Both these methods use the LSODI-based BDF code of the SPRINT package [3, 4] for the time integration. Because the Gear codes of SPRINT and DASSL are very much alike, the choice between the two should be of minor importance for the performances observed. For all three methods, use of the banded form of the equations is exploited in the Jacobian formation and numerical linear algebra computations.

From the user's point of view it is of interest to note that the stiff solvers can be used in the same way as in the conventional approach. Apart from providing a subroutine for the ODE system (numerical differencing for Jacobians was used) and specifying the initial vector $Y(0)$ and required output times, one must define the familiar local error tolerances *atol* and *rtol*, the desired local error norm, and optionally, an initial time-step value. Throughout, we have used $atol = rtol = TOL$ (to be specified) and the common L^2 norm. For the automatic grid determination one must specify N, the number of moving space nodes, and the various regularization parameters. Recall that for Method I their values have been specified already in Section 2.2. For Method II we still must specify τ (see Section 2.3) and for Method III, the parameters C_1, C_2, and d (see Section 2.4).

We emphasize that the choice of the regularization parameters is of importance, not only to obtain a good positioning of grid points, but also to obtain an efficient time-stepping process. This will be illustrated quite clearly in the next section, which

deals with the numerical experiments. In other words, we wish to pay considerable attention to the efficiency (number of time steps, Jacobian updates, and back solves) of the time-stepping process, a point which has been neglected in most of the moving-grid work on time-dependent problems.

4. Numerical Comparisons

We shall present results from extensive numerical testing with three example problems, viz., (I) a scalar reaction–diffusion equation that models a "hot spot" problem from combustion theory, (II) Burgers' equation, a scalar prototype for modelling nonlinear convection-diffusion phenomena, and (III) a system of two quasi-linear hyperbolic equations modelling the interaction of two waves travelling in opposite directions. It is worth emphasizing at the outset that these three problems have different solution behaviours. We recall that our main aim is to assess which of the three moving-grid methods is most suitable for retaining the acknowledged features of reliability, robustness, and efficiency of the conventional MOL approach. For this reason, our first problem was chosen such that a comparison with results obtained a non-moving grid is still feasible. This will enable us to compare the mutual efficiency of time-stepping on moving and non-moving grids, a point which has received insufficient attention in the moving-grid literature.

4.1. *Problem I: A Scalar Reaction-Diffusion Problem from Combustion Theory*

This problem is described in Adjerid and Flaherty [1] as a model of a single-step reaction with diffusion and reads

$$\begin{aligned} \partial u/\partial t = \partial^2 u/\partial x^2 + D(1 + a - u)\exp(-\delta/u), \qquad & 0 < x < 1,\ t > 0, \\ \partial u/\partial x(0, t) = 0, \qquad & u(1, t) = 1,\ t > 0, \\ u(x, 0) = 1, \qquad & 0 \leqslant x \leqslant 1, \end{aligned}$$

where $D = Re^{\delta}/(a\delta)$ and R, δ, a are constant numbers. The solution represents a 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 very high speed. The degree of difficulty of the problem is very much determined by the value of δ. Following [1], we have selected the problem parameters $a = 1$, $\delta = 20$, $R = 5$. Petzold [23] also used this problem as a test example, but with the more difficult parameter choice $a = 1$, $\delta = 30$, $R = 5$. The problem reaches a steady state once the flame propogates to $x = 1$. For the current choice of parameters, the steady state is reached slightly before time $t = 0.29$, which we take as the end point. The problem has also been used as a test example in [27], whence we have copied the plotted

reference solution (solid lines in the plots). We use times $t = 0.26$, 0.27, 0.28, 0.29 for output.

For the numerical process two solution phases should be distinguished, viz., the formation of the "hot spot" with the flame front (the ignition phase) and the propagation of this front to the right end point $x = 1$ (the propagation phase). Accurate handling of the formation of the "hot spot" and the ignition is of importance. The ignition proceeds very rapidly, causing a widely different time scale, so that variable steps in time are a necessity. A difficulty hereby is that the start of the ignition must be detected accurately and without overshoot by the local error control mechanism of the stiff solver, so that the step size can be rapidly reduced to a level small enough to simulate the ignition accurately. Small errors at this time point result in significantly larger global errors later on. Some trial and error tests have revealed that the BDF codes need a time tolerance TOL of 10^{-5}, using an initial step of size of 10^{-5}. For methods which are able to step in time with higher order formulas, such a small tolerance should cause no problems. It is certainly detrimental to a method which is forced to use a low order time-stepping formula, like Method I. For clarity we emphasize that due to the sensitivity of estimating the ignition point, the errors resulting from the time integration are more important than the errors resulting from the spatial discretization.

Because the flame is not very thin, this problem can also be satisfactorily solved in the conventional way on a uniform, non-moving mesh consisting of, say, about 40 to 100 nodes, at least for the current choice of $\delta = 20$. The problem is of interest for moving-grid methods of the Lagrangian type, since these methods should be able to reduce significantly the number of time steps needed to complete the propagation phase. Finally, in all the experiments described below, including those done on a uniform non-moving mesh, we have used 40 moving points and in all cases the start grid was taken to be uniform.

In the plots the solid or dashed lines represent accurate reference solutions while the marks represent the PDE approximations generated in the experiment discussed. Integration information, which serves to compare the mutual time-stepping efficiency of the three methods. is represented in terms of $STEPS$ = total number of successful time steps, $JACS$ = total number of Jacobian evaluations, and BS = total number of back solves. The two latter quantities determine, to a great extent, the CPU time needed to complete the integration over the specified time interval.

Results for Method I

For the present problem Method I (version (A)) is indeed not competitive because the low order time-stepping method turns out to be too expensive. Only during the formation of the "hot spot" can the advantage of using higher order in time formulas be really employed. At the start of the ignition and during the whole of the propagation phase, the method keeps regridding, which means that very many restarts are made with the first-order implicit Euler rule, for which the local accuracy demand of $TOL = 10^{-5}$ is simply too high. Figure 4.1 shows the PDE solution generated at the four output times (specified above) and gives the values

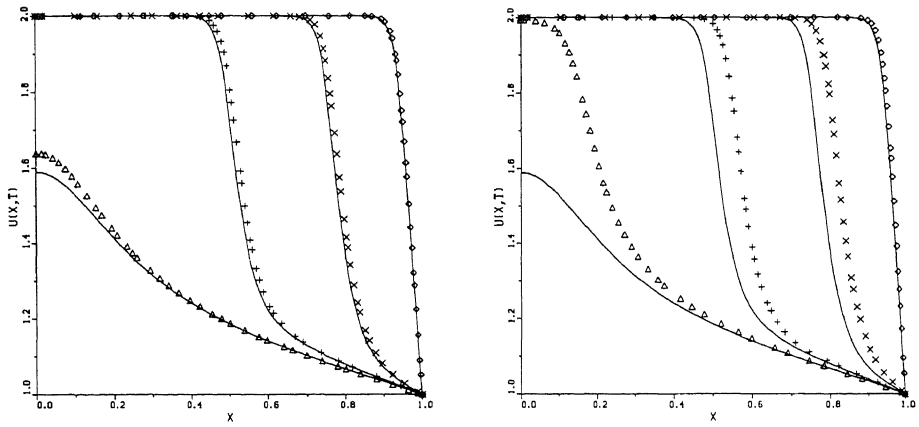


FIG. 4.1. Results for Problem I obtained with Method I. We have used $t = 0.26$, 0.27, 0.28, 0.29 for output. The left-hand plot corresponds to the version (A) run with $STEPS = 663$, $JACS = 709$, $BS = 1845$, and the right-hand plot to the implicit Euler run with $STEPS = 960$, $JACS = 663$, $BS = 1974$. Note that the numerical errors arise mainly from the time integration.

for *STEPS*, *JACS*, and *BS*. Observe that the numerical front is ahead of the true one. Concerning the quality of the reference solution we note that in [27] it is claimed that the reference solution is "exact up to plotting accuracy," except perhaps in the neighbourhood of $x = 0$ at the first output time $t = 0.26$. All experiments with the present flame problem, including those with Methods II and III, show a deviation here.

It should be remarked that we counted an unusual number of 259 error test failures. The greater part of these occur directly after a genuine regridding, indicating that the step-size selection of the restart mechanism is not well tuned. To test this we have repeated the integration using a maximal order of one, so that then at all integration steps the implicit Euler method is used. We now counted 960 successful steps and only 28 error test failures, which is normal. The results of this experiment are also shown in Fig. 4.1. One sees that the results of the backward Euler run are less accurate, in spite of the fact that more time steps are used. This shows nicely that, during the formation of the "hot spot," version (A) benefits from the use of the higher order formulas. It should also be realized that a large number of step rejections will considerably increase *BS* and, most likely, also *JACS* (compare the given quantities of the two experiments). No attempt has been made to repair this failure because, even without these many rejections, the method would not be competitive with Methods II and III.

We recall that we have applied the method with a fixed number of nodes, whereas in [23] the number of nodes is variable. This, however, is of minor importance. Even with a variable number of nodes, many regriddings are performed, which is the main shortcoming. Admittedly, when a fixed number of nodes is used,

the dual reconnection strategy will probably lead to somewhat more interpolations, as it is then not possible to truly delete points.

A natural question is how Method I would perform if the regridding were carried out, not every time step, but every kth time step (k to be prescribed) or at prescribed times. If the chosen time intervals are large enough, so that DASSL can enlarge the order and use the same Jacobian, the drawback of the intermediate regriddings should then be alleviated considerably. In addition, the co-ordinate transformation governing the grid movement softens the solution behaviour in time, which in itself is beneficial for the time-stepping process. A word of warning is in order, of course. During the moving-integration process, the nodes may be sent away from the evolving front (cf. Section 2.1), which makes this alternative mode of operation a bit risky. Yet we do believe that this approach of intermediate regridding is much more promising and that it deserves further attention. By way of illustration, we have again solved the current flame problem with regridding at step points nearest to the prescribed times $t = 100k/0.29$ for $k = 1(1)100$. This gives a solution of comparable accuracy to that observed in the version (A) run, but with a significant reduction in computational costs. The data are $STEPS = 331$, $JACS = 98$, $BS = 739$, and we counted 35 error test failures. As anticipated, DASSL now also uses higher order formulas (mostly order 3) over the entire time interval.

Results for Method II

An important parameter of Method II is the grid parameter τ, which has been introduced to govern the temporal grid smoothing. Figure 4.2 shows typical results for four decreasing values of τ, of which the largest value has been chosen such that a non-moving grid results. This enables us to compare the mutual efficiency of time-stepping on a moving and a non-moving grid. We see that, as the values of τ decrease, the grid follows the flame better and better and *STEPS* is steadily reduced, which nicely reflects the Lagrangian nature of the method in the propagation phase. Further, and this is most important for efficiency reasons, the method keeps *JACS* and *BS* at the same low level, which is the desired MOL behaviour. Needless to say, compared to the first method, Method II performs much more efficiently. This is largely due to the fact that in all runs BDF orders up to three (occasionally four and five) were used over the entire time interval. The accuracy is also much better, though it should be observed that the numerical flame front is a little too fast over the entire solution interval because the scheme is taking too large time steps. The accuracy improves notably by reducing TOL, but at the cost of more computational work. The experiments indicate that it suffices to work with a fairly small value for τ. In fact, for the present problem, temporal grid smoothing turns out to have little effect. The choice $\tau = 10^{-8}$ yields $STEPS = 162$, $JACS = 41$, $BS = 511$, without a noticeable change in accuracy.

Finally, we wish to point out that the "non-moving, uniform grid computation" of Fig. 4.2 (the case $\tau = 1.0$) should not be interpreted as the conventional uniform grid computation, because the semi-discrete systems differ. Although this should

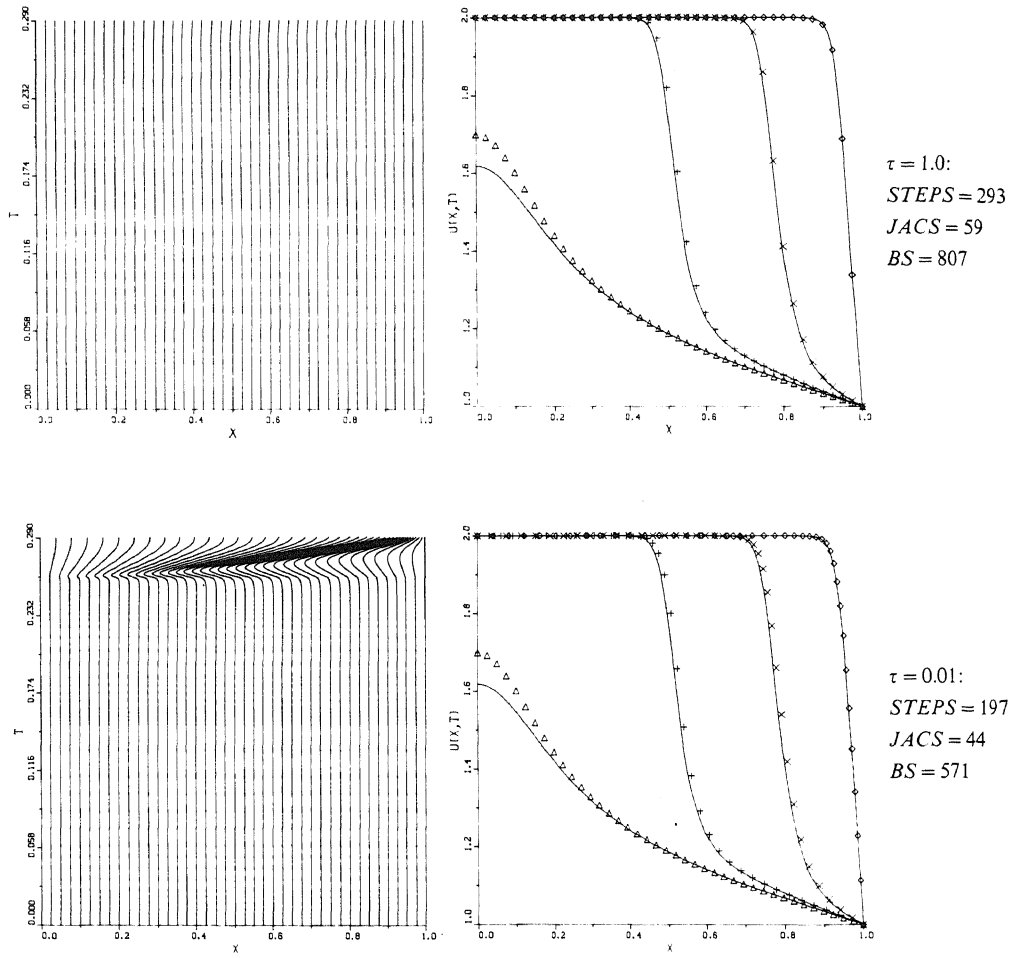
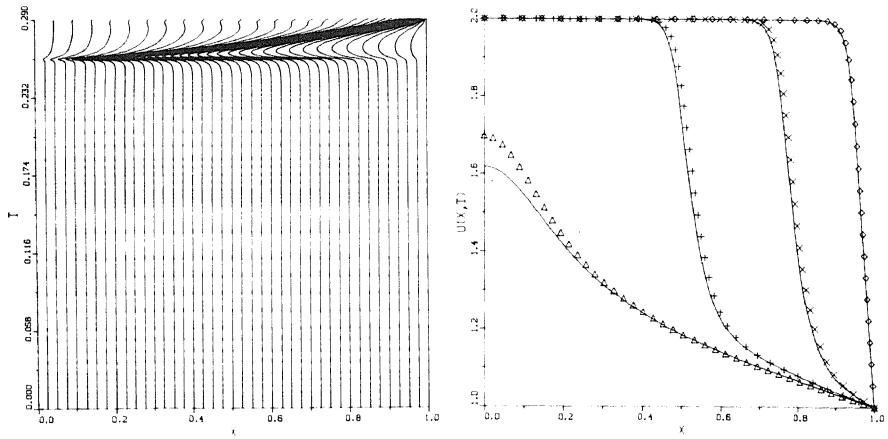


FIG. 4.2. Solutions and trajectories for Problem I generated by Method II. The output times are as in Fig. 4.1. Note that the numerical errors arise mainly from the time integration.

have no influence on the PDE solutions generated, it obviously may influence the solution process through the Newton iteration.

Results for Method III

Let us inspect Fig. 4.3, which shows plots of grids and PDE approximations for four decreasing values of the regularization parameter C_1, beginning with $C_1 = 10$ (in all four cases, $C_2 = d = 0$). This largest value for C_1 yields a virtually non-moving grid. The aim of this experiment, as mentioned with respect to Method II, is to illustrate the dependence of the time-stepping process on the grid movement.



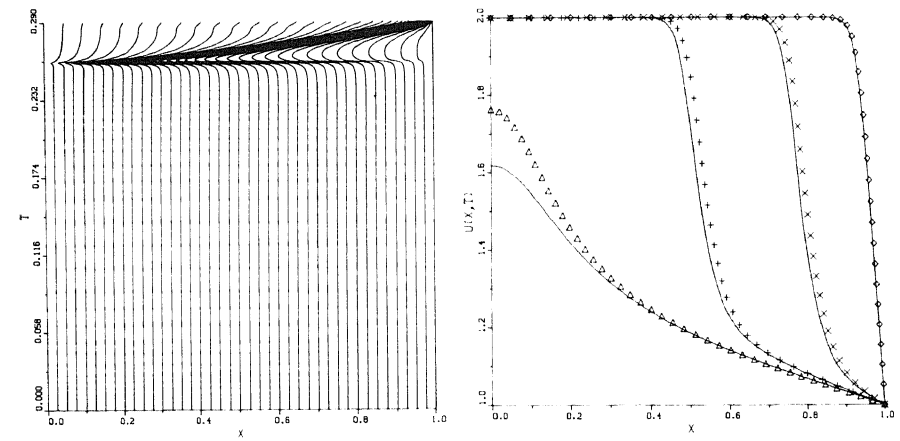


FIG. 4.2—*Continued.*

It is our experience that in this respect the finite-element method behaves less satisfactorily than Method II. The approximations on the uniform non-moving grid are very accurate, except perhaps within the vicinity of $x = 0$ during ignition. The attractive, conventional MOL behaviour is nicely visible. *JACS* is only a small fraction of *STEPS* and, also, *BS* is rather low. This is just why the conventional MOL approach is often so efficient. To avoid a possible misunderstanding, it should again be realized here that the "uniform grid computation" (the case $C_1 = 10$) differs from the conventional one, in the sense that the semi-discrete systems, and thus the Jacobian matrices encountered, are different. This may have some influence on the solution process but the PDE solutions generated should be identical.

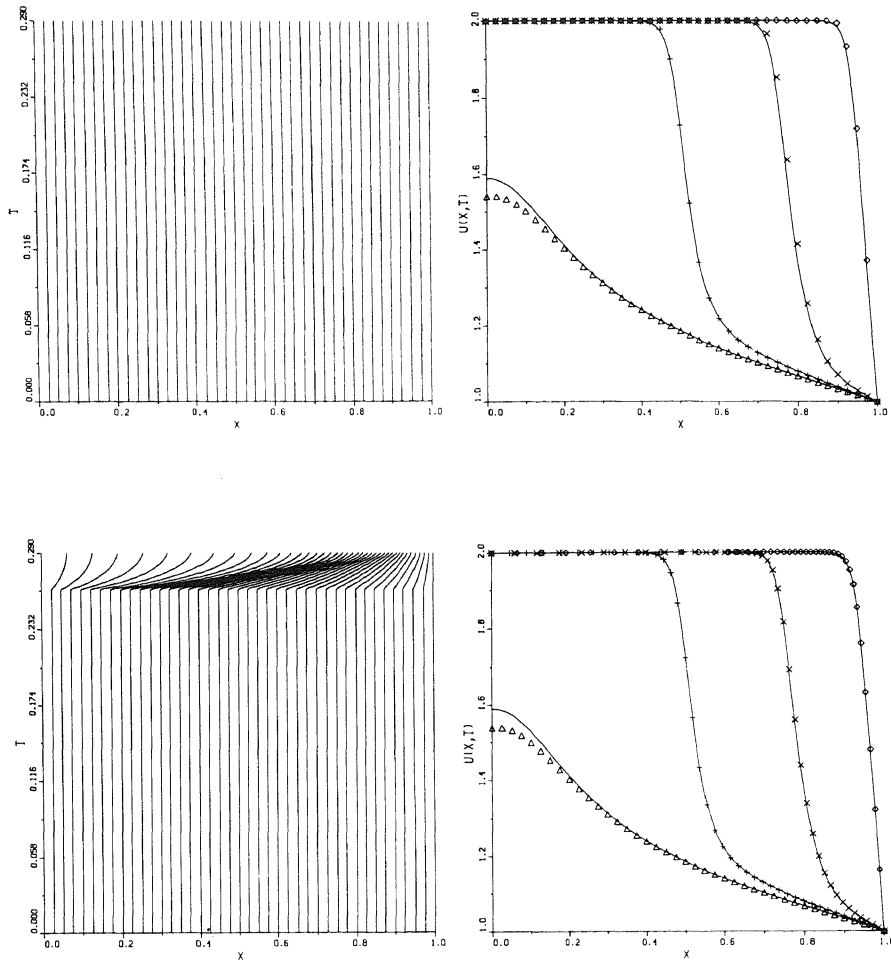


FIG. 4.3. Solutions and trajectories for Problem I generated by Method III. The output times are as in Fig. 4.1. Note that the numerical errors arise mainly from the time integration.

Let us now consider the remaining cases. As to be expected, we see that the grid follows the flame better and better for decreasing C_1, with the result that an even better resolution is obtained during the propagation phase. We also see that, in spite of the fact that *STEPS* slightly decreases with C_1, *JACS* and *BS* steadily grow. For example, the increase in *JACS* and *BS* when C_1 changes from 10 to 0.1 is significant, while the grid movement is still rather modest and also the nodes are well separated (hence, it here suffices to put $C_2 = d = 0$). Disappointingly, for the smaller C_1 values quite a lot of computational effort must be spent in order to solve the nonlinear systems which arise. It will be clear that, in such a situation,

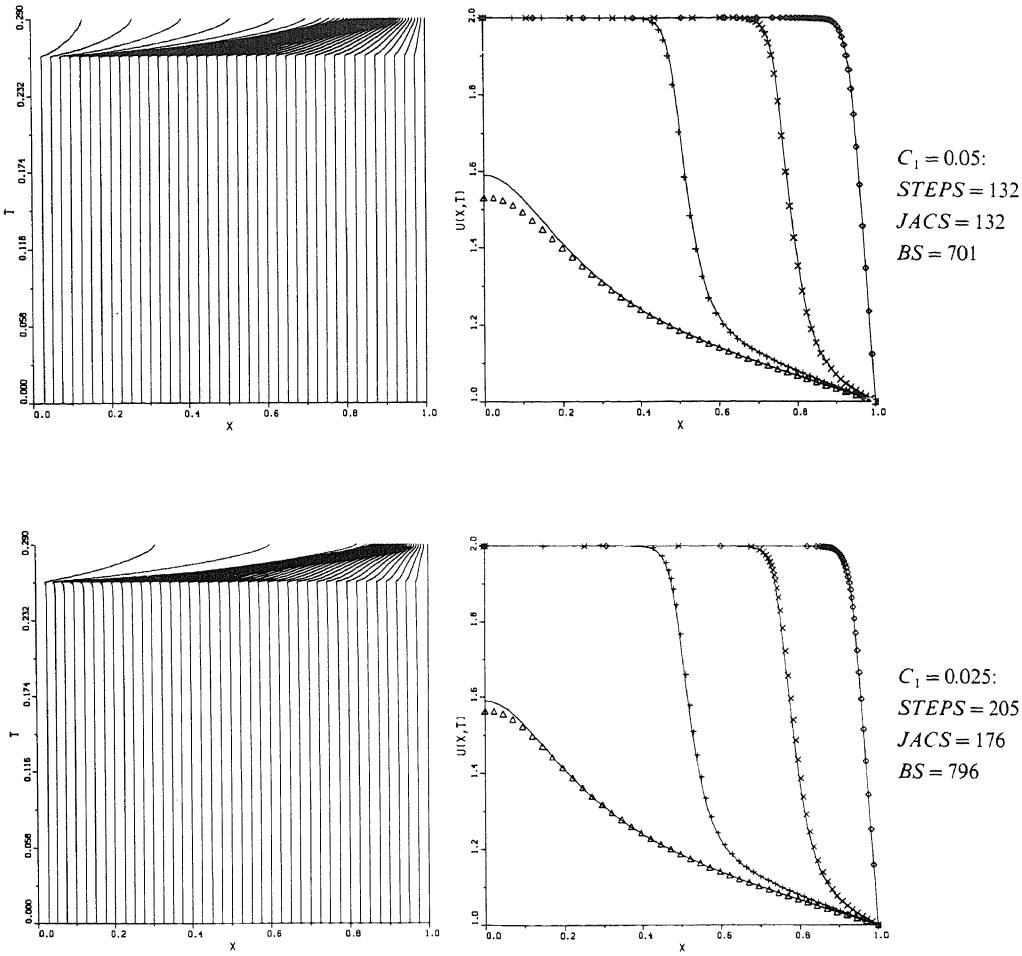


FIG. 4.3—*Continued.*

anticipated savings in total computational effort, due to a reduction of the number of space nodes, may well be largely annihilated due to the much larger costs for the time-stepping. It should further be observed that, in contrast to Method II, the Lagrangian nature of the moving-finite-element method during the propagation phase does not lead to a considerable decrease of time steps.

In a sense, the application of the moving-finite-element method places us in a dilemma. A near-"optimal" value for the regularization parameters would yield a near-"optimal" grid movement and an excellent approximation. On the other hand, the current experiment indicates that the grid may move at the expense of considerably higher computational costs. One might argue here that for C_1 small

the points come too close, since a further decrease of C_1 would yield node overtaking. However, in all four cases illustrated, the grid points are still sufficiently separated. We conjecture that the problems associated with the iterative Newton solution of the nonlinear algebraic moving-finite-element system are probably due to some kind of ill-conditionedness which is inherent to the moving-finite-element construction. This conjecture is supported by the observation that, in all four runs, BDF orders of three, and occasionally four, have been used over the entire time interval, which indicates that the semi-discrete solutions cannot be very unsmooth. Moreover, the number of time steps is not markedly large. In other words, the observed difficulty of the high frequency of Jacobian evaluations is probably hidden somewhere in the nonlinear equation system itself.

Another point of concern is the choice of the regularization parameters C_1, C_2, and d. In spite of the fact that the meaning of these parameters is sufficiently clear, it is not clear at the outset how to select them. Loosely speaking, the control offered by them is in a sense not direct enough. By way of illustration, consider the two choices $C_1 = 0.025$, 0.05. For $C_1 = 0.025$ the grid is positioned rather well, which can be seen by taking a closer look at the steady state solution. Most of the points are concentrated where the curvature is largest and also the distribution within the layer is good. On the other hand, one might still argue that the ratio of adjacent points left of the front is rather large, which is well known, may be detrimental to spatial accuracy. Doubling C_1 yields better ratios, but then the grid is somewhat too slow, with the result that now too many points are wasted in the flat part. We admit that these observations are rather subtle and that similar observations can made for Method II concerning the choice of the parameter τ. Still, it is our experience that fine-tuning Method III can be rather troublesome, which brings us in direct conflict with the important issues of robustness and reliability. For example, decreasing C_1 further to 0.0125 results in a totally wrong steady state solution, whereas the generated transient solution is perfectly all right (with $C_2 = d = 0$; this failure can be overcome by adjusting C_2 and d).

4.2. *Problem II: Burgers Equation*

Our second example is the well-known Burgers equation

$$\partial u/\partial t = -\partial f(u)/\partial x + \varepsilon \partial^2 u/\partial x^2, \qquad 0 < x < 1,\ t > 0,\ f(u) = u^2/2,\ \varepsilon = 10^{-4},$$

supplemented with the smooth initial function $u(x, 0) = \sin(2\pi x) + 0.5 \sin(\pi x)$ and homogeneous Dirichlet boundary conditions. This problem also served as a test example in [10, 12]. The solution is a wave that first develops a very steep gradient and subsequently moves towards $x = 1$. Because of the zero boundary values, the wave amplitude diminishes with increasing time. We consider the time interval $[0, 2]$ and use times $t = 0.2$, 0.6, 1.0, 1.4, 2.0 for output.

In contrast with the previous problem, the location of the fine grid region is very critical, since all three methods are known to generate spurious oscillations readily

if the grid in the layer region is too coarse, just as with standard central differences on a non-moving grid. Concomitant with this form of space instability is the danger of having non-smooth continuous-time, semi-discrete solutions. In other words, despite the fact that we move the grid, these solutions still have a tendency to oscillate, even for small grid deviations. There is no doubt that this non-smoothness is detrimental to any ODE solver and therefore the present problem provides a difficult test for any moving-grid method. In all experiments we have worked with 40 moving nodal points and a uniform start grid.

Results for Method I

For the above Burgers equation problem, Method I, at least the (A) version, falls dramatically behind when compared with Methods II and III. Using an initial step size of 10^{-5}, we have run the method for three values of *TOL*, viz. 10^{-2}, 10^{-3}, and 10^{-4}. In all three cases the method generates the correct spatial profile; however, the numerical wave runs much too fast, in particular, for the two lower tolerances. This must be attributed to the inaccuracy of the implicit Euler scheme, which is used in almost all steps, and to the very frequent interpolations. Taking into account the computational effort needed for $TOL = 10^{-4}$, a further reduction of *TOL* was not considered worthwhile. Again we must conclude that the disappointing performance is due to the regridding at virtually all steps, forcing the method to use the first-order Euler formula. In passing we note that for this problem the number of step failures, which in an experiment with Problem I turned out to be uncommonly large, is here virtually negligible.

Again the question arises as to what extent the less frequent regridding approach mentioned in the discussion of results for Problem I would be more promising. By way of illustration we have rerun the method for $TOL = 10^{-3}$, 10^{-4} while regridding only at step points nearest to the prescribed times $t = k/50$ for $k = 1(1)100$. As for Problem I, this gives a considerable improvement, both in accuracy and with respect to computational costs. The results are shown in Fig. 4.4. These results, while not yet competitive with those of Methods II and III, do, however, indicate clearly that the approach of occasional regridding in time is to be preferred to the approach of regridding at (nearly) every step, which underlies version (A). It is likely that there is room for considerable improvement. For example, the number of time steps for $TOL = 10^{-4}$ is about $(10)^{1/2}$ times larger than for $TOL = 10^{-3}$, which indicates that the (locally second order) implicit Euler method is still used very frequently. No doubt, had higher order formulas been used, better performance would have been observed.

Results for Method II

Figure 4.5 depicts the grids and solutions for Method II for $\tau = 10^{-1}$ and 10^{-3} ($TOL = 10^{-3}$ and the initial time step is 10^{-5}). The corresponding integration data are listed below, together with the results obtained for $\tau = 10^{-2}$ and 10^{-4}. The (plotting) accuracy for the three smaller τ values is the same and without doubt can be called excellent. Recall that the solid lines represent a highly accurate reference

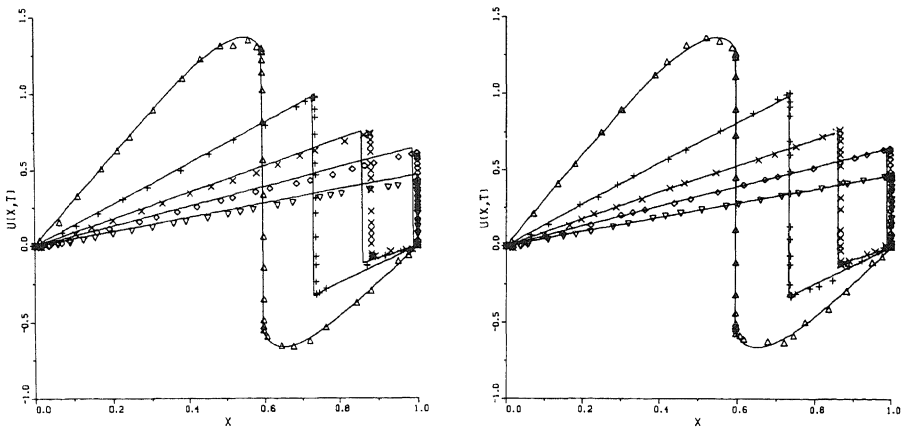


FIG. 4.4. Results for Problem II obtained with Method I using the intermediate regridding approach. The output times are $t = 0.2$, 0.6, 1.0, 1.4, 2.0. The left-hand plot corresponds to $TOL = 10^{-3}$ ($STEPS = 561$, $JACS = 482$, $BS = 1486$) and the right-hand one to $TOL = 10^{-4}$ ($STEPS = 1650$, $JACS = 919$, $BS = 3986$).

solution and that the marks correspond to the numerical solutions generated in the present experiment.

As already observed in the problem description, due to the small amount of diffusion, the semi-discrete solutions have a tendency to oscillate as soon as the grid becomes a little bit too coarse in the layer region. This makes the problem difficult to solve and, in fact, is the main cause for the relatively large number of Jacobian updates. It also explains the much larger effort and wiggles for $\tau = 0.1$, for which value the grid is a little bit too slow. It is obvious that for a problem like this, the choice of τ, which dictates the grid movement, is more critical than for Problem I. On the other hand, as for Problem I, a rather small value for τ (of the order of the averaged time step used) turns out to be most appropriate. Most of the time steps used were for the shock formation and collision with $x = 1$; very few steps were needed to propagate the shock from $x = 0.6$ to $x = 0.95$. Finally, the cusps in the ($\tau = 10^{-3}$) grid near $t = 1.4$ are due to the change of shape in the solution when the shock reaches the right-hand boundary. The fact that these are virtually absent in the $\tau = 10^{-1}$) grid nicely illustrates that here the temporal grid smoothing is too large.

$\tau = 10^{-1}$	$\tau = 10^{-2}$	$\tau = 10^{-3}$	$\tau = 10^{-4}$
$STEPS = 747$	$STEPS = 293$	$STEPS = 212$	$STEPS = 224$
$JACS = 428$	$JACS = 164$	$JACS = 120$	$JACS = 134$
$BS = 2595$	$BS = 910$	$BS = 708$	$BS = 749$

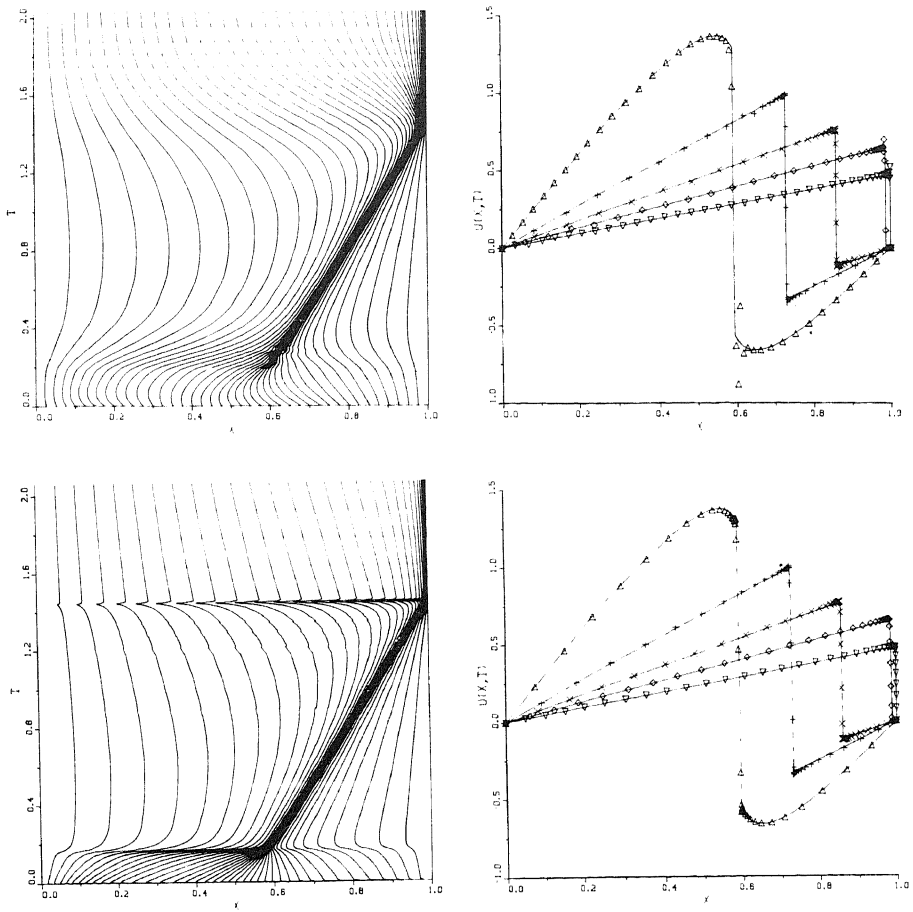


FIG. 4.5. Results for Problem II obtained with Method II. The output times are the same as in Fig. 4.4. The two upper plots correspond to $\tau = 0.1$ and the two lower ones to $\tau = 0.001$.

Results for Method III

In all the experiments we have used the time tolerance value $TOL = 10^{-3}$ with initial step size 10^{-5}. As a first experiment we tried the method using regularization parameter values copied from Hrymak, McRae, and Westerberg [12]. Their values are: $C_1 = 0.01$, $C_2 = 10^{-4}$, and $d = 5.0 \times 10^{-5}$. Hrymak *et al.* integrate Problem II only until $t = 1$ and on this time interval the integration is successful. However, upon continuing the integration to the end point $t = 2$, we experienced node crossing near approximately $t = 1.4$. Increasing C_1, for example, overcomes the crossing. For $C_1 = 0.025$ the integration is successful over the entire time interval $0 \leqslant t \leqslant 2$ and leads to a very accurate solution, but at rather large costs, viz., $STEPS = 364$, $JACS = 270$, and $BS = 941$. This, in turn, can be improved by enlarging the mini-

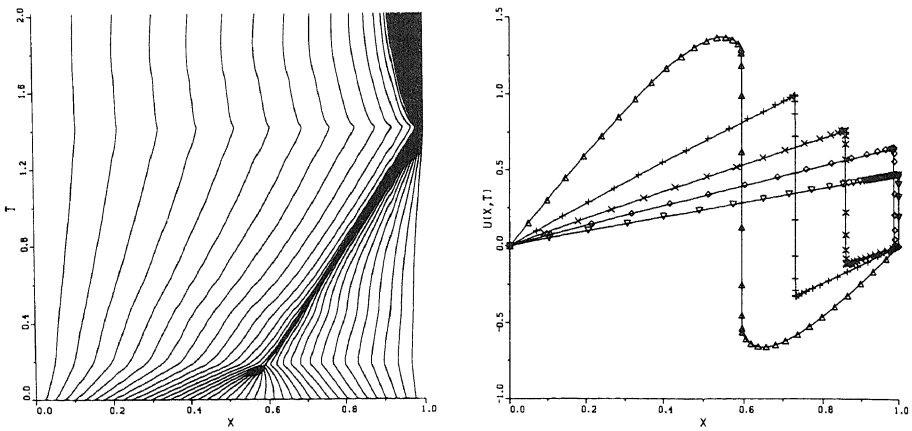


FIG. 4.6. Solutions for Problem II computed with Method III. The output times are the same as in Figs. 4.4 and 4.5. The parameter values are $C_1 = 0.025$, $C_2 = 10^{-4}$, $d = 10^{-4}$.

mal node distance parameter d, for example, $d = 10^{-4}$ yields an equally accurate solution. Figure 4.6 shows this solution, obtained using $C_1 = 0.025$, $C_2 = d = 10^{-4}$, for which the costs are $STEPS = 271$, $JACS = 190$, $BS = 719$.

A further increase of d, to about 5.0×10^{-4}, is not possible since then the grid in the layer becomes too coarse and thus the familiar oscillations arise. In the present experiment these also end in node crossing. In view of the oscillations, we recall that there should be an upper limit on d and that this upper limit is related to the size of the viscosity parameter ε in Burgers equation, since it is this parameter which determines the width of the layer region. Some further trial and error runs, with the earlier values $C_1 = 0.025$ and $C_2 = 10^{-4}$, revealed that the admissible range for d is not very large. We observed node crossings for $d = 10^{-5}$ and $d = 5.0 \times 10^{-4}$.

In conclusion, Method III is able to solve the present difficult Burgers equation problem with high accuracy, but not without considerable tuning. It should also be noted that the costs of the successful, accurate computation of Fig. 4.6 are larger than those of the successful runs with Method II for the smaller τ values. We attribute this to the fact that here SPRINT starts to integrate with the first-order implicit Euler scheme as soon as the wave develops the steep gradient and hence does not exploit the higher order BDF formulas. This, in turn, indicates that, for the convection dominated problem, the continuous-time, semi-discrete solution generated by the moving-finite-element method will be rather non-smooth in time, a situation we already anticipated in the problem description.

4.3. *Problem III: Waves Travelling in Opposite Directions*

Our third example problem is a two-component, semi-linear hyperbolic system, the solution of which is constituted by two waves travelling in opposite directions (copied from Madsen [15], see also [27]). The system is given by

$$\partial u/\partial t = -\partial u/\partial x - 100uv,$$

$$\partial v/\partial t = \partial v/\partial x - 100uv,$$

for $t > 0$ and $-0.5 < x < 0.5$, and the solution is subjected to homogeneous Dirichlet boundary conditions and to the initial condition

$u(x, 0) = 0.5(1 + \cos(10\pi x))$ for $x \in [-0.3, -0.1]$ and $u(x, 0) = 0$ otherwise,

$v(x, 0) = 0.5(1 + \cos(10\pi x))$ for $x \in [0.1, 0.3]$ and $v(x, 0) = 0$ otherwise.

Note that these are functions with a mere C^1 continuity, which represent wave pulses located at $x = -0.2$ and $x = 0.2$, respectively. Initially, the nonlinear term $100uv$ vanishes, so that for $t > 0$ these waves start to move without change of shape and with speed 1, u to the right and v to the left. At $t = 0.1$ they collide at $x = 0$ and the nonlinear term becomes positive, resulting in a nonlinear interaction leading to changes in the shapes and speeds of the waves. Specifically, the crests of the waves collide a little beyond $t = 0.25$ and they have separated again at approximately $t = 0.3$, so that from this time on the solution behaviour is again dictated by the linear terms. At the nonlinear interaction, the pulses lose their symmetry and experience a decrease in amplitude.

To save space, in this section we restrict ourselves to presenting results for Methods II and III (Method I was applied, but with rather inaccurate results). As output times we have selected the values $t = 0.1$, 0.2, 0.25, 0.3, 0.5 and in all experiments the integration has been started at $t = 0$ on a non-uniform, solution-adapted grid consisting of 41 points. For both methods we have used the time step tolerance value $TOL = 10^{-3}$ and an initial step size of 10^{-5}.

Results for Method II

Figure 4.7 shows the grid and the numerical approximations at the specified output times, obtained with a value of 10^{-3} for the grid delay parameter τ. We see that the solutions are fairly accurate and point out that the visible inaccuracies are only due to a somewhat optimistic choice for TOL and the number of points. These inaccuracies will vanish if more points and a smaller tolerance are used. Also the grid positioning is good over the entire time interval; i.e., there is sufficient refinement near the travelling waves before and after the interaction. In the present experiment we have replaced the (regularization) constant 1 of the arc-length monitor

$$(1 + (\partial u/\partial x)^2 + (\partial v/\partial x)^2)^{1/2}$$

by 0.1. The reason is that when the waves have separated they are no longer very steep, with the result that the value 1.0 is somewhat too large for obtaining sufficient refinement in the vicinity of the two waves, at least when only 41 points are used. With this number of points, it is also necessary that, after the separation, the grid refines properly in the vicinity of the waves, since otherwise spurious

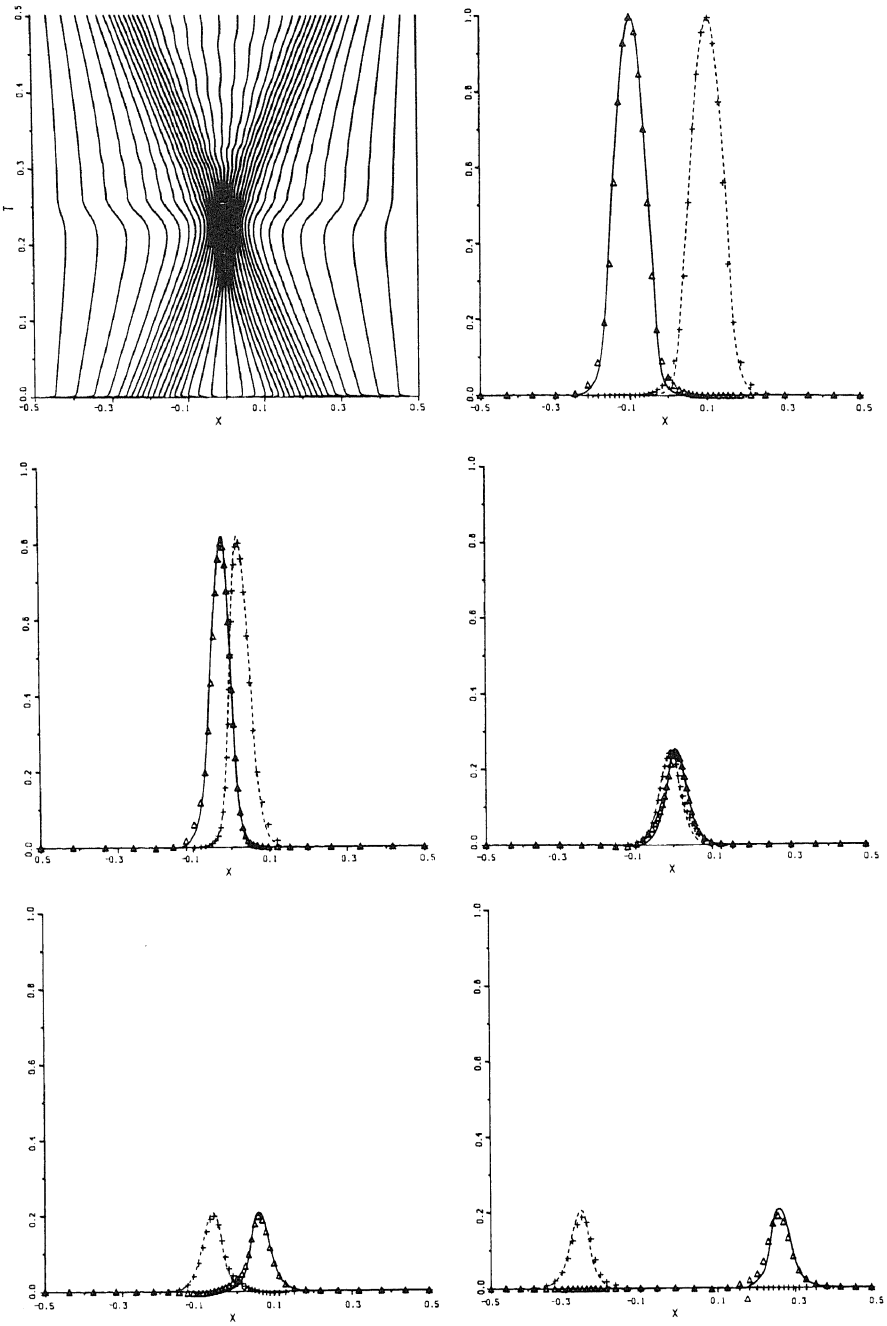


FIG. 4.7. Grid trajectories and solutions for Problem III computed with Method II. The output times are $t = 0.1, 0.2, 0.25, 0.3, 0.5$.

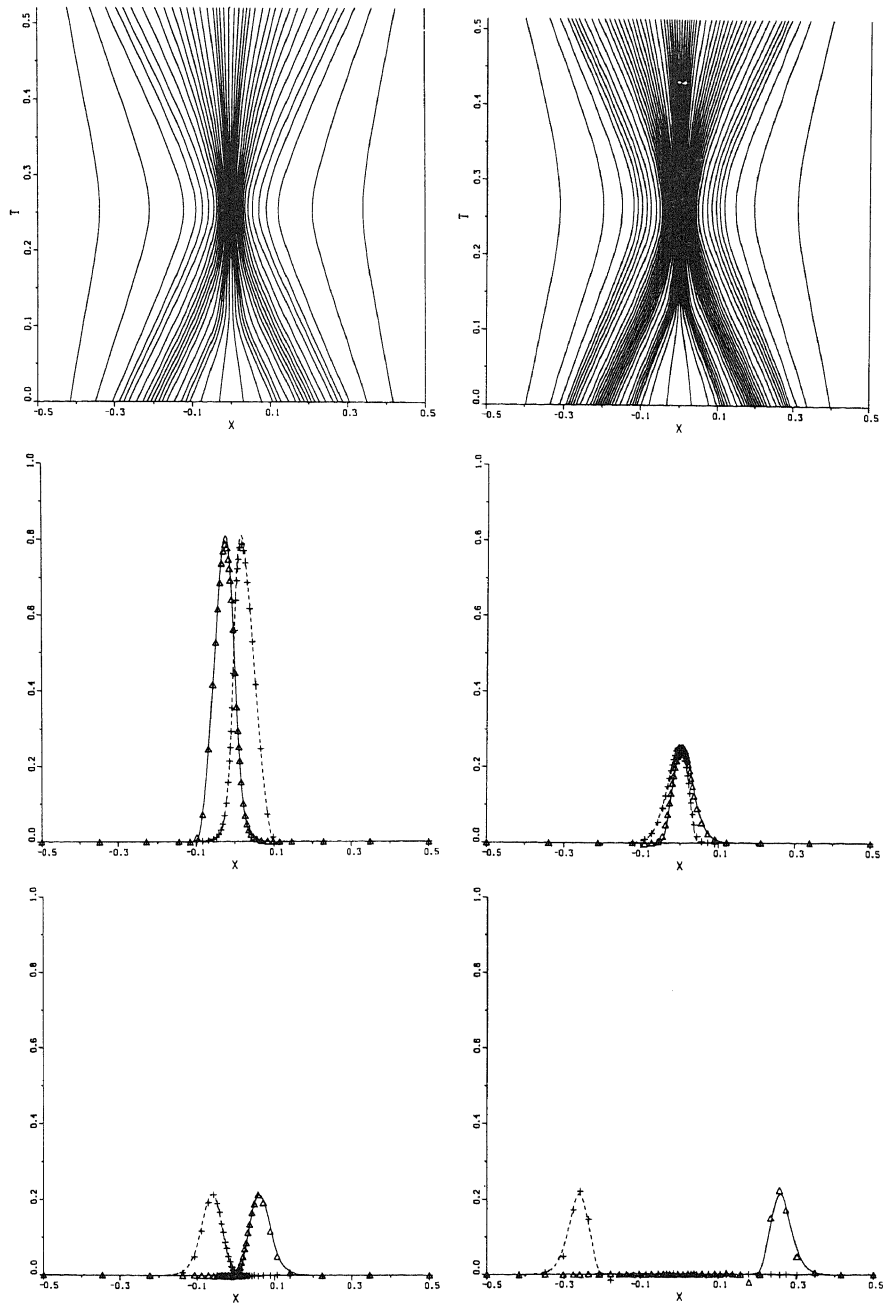


FIG. 4.8. Grid trajectories and solutions for Problem III computed with Method III. The left-hand grid plot and the four lower plots belong to the run with 40 moving points and $TOL = 10^{-3}$. Output times are the same as in Fig. 4.7, except that here $t = 0.1$ has been omitted. The right-hand grid corresponds to 60 points and $TOL = 10^{-4}$.

oscillations will become visible. Recall that after the separation we are just solving the first-order hyperbolic model problem using standard central differences. This experiment shows that it is desirable that the regularization constant of the monitor function be made solution-dependent, in some way or another. Finally, the costs of the run are $STEPS = 105$, $JACS = 58$, and $BS = 332$.

Results for Method III

A typical result obtained with the parameter values $C_1 = 0.05$, $C_2 = 10^{-4}$, and $d = 10^{-5}$ is shown in Fig. 4.8. The costs of the run are $STEPS = 71$, $JACS = 38$, and $BS = 177$. We see that up to approximately $t = 0.25$ the grid moves in the right way and the two numerical waves follow the exact ones quite accurately. As for Method II, the small, visible inaccuracies are due to a somewhat optimistic choice of TOL and the number of points. Unfortunately, the method fails to track accurately the separation of the waves, which can be seen by inspecting the grid. Although after the separation the solution is quite accurate, except for the wiggle at the tails (see $t = 0.5$), the grid positioning is not in accordance with the location of the two waves, in contrast with the positioning for $0 \leqslant t \leqslant 0.25$. For $t > 0.25$ the grid tends to become more or less uniform over the greater part of the space interval and does not refine in the vicinity of the travelling waves.

It is noted that this grid deficiency does not vanish upon increasing the number of points and the temporal accuracy level, at least for 60 moving points and $TOL = 10^{-4}$ (the right upper plot of Fig. 4.8 depicts the corresponding grid). Attempts to overcome it by changing the penalty parameters were not successful either; nor was the addition of a small amount of viscosity (10^{-4}) to suppress spurious oscillations. The addition of a small amount of artificial viscosity, which was suggested by K. Miller (personal communication), does reduce the oscillations in the solution, but does not have a visible impact on the grid. It is conjectured that the observed difficulty has to do with the property that, for the hyperbolic model problem we are actually solving after the separation, the moving-finite-element method moves the grid at speed one and returns the exact solution, but does not adjust the grid to the new separated pulse profile. Nodes are dragged out of the pulses as they separate because of the large, internodal viscosity coefficient C_1^2 which we found necessary to use to get the code to work. This value of C_1^2 is 100 to 600 times the standard choice of Miller [18].

5. Conclusions

We have examined three Lagrangian-based moving-grid methods for systems of 1D time-dependent partial differential equations. Our aim has been to assess which of these methods offers the best prospects for reliable, efficient and robust method-of-lines application, preferably with as little user intervention as possible. For this purpose we have carried out a numerical comparison with three different test examples. For the time integration we have used two existing, closely related stiff ODE

codes, both of which are based on the acknowledged BDF formulas. We formulate the following conclusions:

(i) We cannot recommend Method I for general use, although it is quite reliable and robust; for example, we never found it necessary to use values for the parameters α and λ different from the specified default values. The very frequent regriddings mean that the method has to integrate almost always with the first-order implicit Euler rule, thus preventing the Lagrangian procedure from exploiting the attractive, higher order BDF formulas. In many situations this will be detrimental to efficiency, apart from incurring the extra cost of a Jacobian update after regridding. A second drawback of regridding is the need to interpolate. In spite of the fact that accurate monotone interpolation is combined with the dual reconnection strategy, which implies that after a regridding the number of point interpolations is not very numerous, many successive interpolations can still cause a perceptible loss of spatial accuracy. In this connection it is worthwhile to note that one of the recognized advantages of Lagrangian schemes, when operating with a fixed number of moving points, is that they do not require interpolation.

Our experiments indicate that a significant improvement can be obtained when the number of regriddings is limited in some way or another (the intermediate regridding approach) because then the time-stepping can benefit more from the Lagrangian nature of the method. When considered on its own, the underlying Lagrangian transformation is of interest since the aim is to achieve smoothness in time, which is of course attractive, certainly when the higher order BDF formulas are available for the time integration.

(ii) We do not wish to conceal the fact that we have mixed feelings about the moving-finite-element approach underlying Method III, at least as far as our application is concerned. This is based on the following observations. In this approach the movement of the grid is basically governed by a minimization procedure, akin to the procedure for standard non-moving-grid Galerkin schemes. For practical application within an implicit method-of-lines procedure it is necessary, through the use of penalty terms, to regularize this minimization so as to avoid node overtaking and singular mass matrices. Inevitably, the choice of the parameters involved is problem-dependent and experience has revealed clearly that this often leads to troublesome application. Quite some tuning may be needed to make the grid move in a satisfactory way. In a sense, the effect of the regularization on the minimization does not seem to provide a sufficiently clear and unique set of rules for moving the grid. In this respect the spatial equidistribution approach which underlies Method II is more transparent.

The need for tuning is obviously in conflict with the aim of robustness. Another point of concern we should like to bring forward here is that the time-stepping behaviour of Method III is rather sensitive with respect to the grid movement. If the grid does not move in the right way, the time-stepping can easily become rather expensive. Furthermore, even if the grid does move satisfactorily, it may still happen that the time-stepping costs are rather large compared with the costs of

time-stepping in the conventional way on a non-moving grid (assuming of course that a non-moving grid is feasible); see, for example, the experiment carried out with Problem I. We admit that this comment will apply to any moving-grid procedure, including Method II. It is our experience, however, that in this respect the latter method behaves better. Finally we wish to recall (see Section 2.4) that we have only little experience with the gradient-weighted MFE method and that Miller [18–20] claims that gradient-weighted MFE requires considerably less tuning than ordinary MFE and, probably, also improves upon the time-stepping. As already mentioned, a separate investigation on gradient-weighted MFE will be undertaken and results will be reported elsewhere.

(iii) We believe that, for the application we have in mind, the approach of the finite-difference Method II is to be preferred above the moving-finite-element approach of Method III. We have found Method II easier to work with and implement than Method III and also more efficient. The grid movement of Method II is directly attached to equidistribution in space of a chosen monitor function whereas that of Method III has no underlying equidistribution principle and so there is no improvement mechanism for an incorrect initial node distribution. As already indicated under (ii), it is our experience that this approach provides a better and more unique way of automatically adjusting the grid to large spatial gradients.

However, Method II may easily encounter difficulties in tracking sharp corners of a solution where, nearby, the first derivative is not very large. A simple example of such a situation is provided by the model convection equation $u_t + u_x = 0$ with a triangular pulse as initial value. Computing the moving triangular pulse solution with Method II will result in a numerical solution showing the familiar spurious oscillations. This does not happen with genuine shocklike structures because these have an arclength associated with it. Very large spatial derivatives attract enough points to prevent the oscillations to arise but the triangular pulse form does not lead to sufficient refinement near the sharp moving corners. We have experienced numerically that this sort of difficulty will also arise when solving the Burgers equation with a trapezoidal pulse as an initial value instead of the sinusoidal one, a test example suggested by Keith Miller [20]. In this connection it should be emphasized that the MFE method does not suffer from this particular deficiency and can handle this sort of initial values in the Burgers equation with great accuracy using relatively few points [10, 20].

An important feature of the approach of Method II is the grid smoothing capability. Despite involving two method parameters, viz. κ and τ, the choice of these parameters has not proved to be troublesome. The meaning of κ is very clear and for general use κ can be taken equal to, say, 1 or 2. Admittedly, the actual choice to be made for τ is less clear. Our experience in the experiments is that it is best to keep τ small so that the grid movement is almost exclusively dictated by the spatial equidistribution at the forward time level, as long as this does not lead to oscillatory grids. However, for general use it is not recommended to set $\tau = 0$. The temporal grid smoothing property deserves some more study.

(iv) In conclusion, we consider the approach of Method II as most promising for a general method-of-lines application. In the near future we therefore plan to study this specific approach in more detail, with the aim of extending the current (ad hoc) implementation of Method II into a reliable, efficient and robust, user-oriented piece of software which can be easily linked to existing PDE packages like SPRINT.

ACKNOWLEDGMENTS

We are indebted to Linda Petzold for placing at our disposal her entire FORTRAN program for Method I, including the code DASSL. Harm Mekkering is acknowledged for carrying out the tests with this program. We are grateful to Keith Miller and Neil Carlson for sending us their gradient-weighted moving-finite-element code GWMFE1DS, and to Keith Miller for interesting discussions during his stay in Amsterdam. We should also like to thank Joke Blom for her programming assistance, valuable advice, and interest shown during the course of the investigations.

REFERENCES

1. S. ADJERID AND J. E. FLAHERTY, *SIAM J. Numer. Anal.* **23**, 778 (1986).
2. M. J. BAINES, in *Numerical Methods for Fluid Dynamics III*, edited by K. W. Morton and M. J. Baines (Oxford University Press, London, 1988).
3. M. BERZINS AND R. M. FURZELAND, Report TNER.85.058, Thornton Research Centre, Shell Research Limited, (1985 (unpublished).
4. M. BERZINS AND R. M. FURZELAND, Report No. 202, Department of Computer Studies, The University of Leeds, 1986 (unpublished).
5. G. F. CAREY AND H. T. DINH, *SIAM J. Numer. Anal.* **22**, 1028 (1985).
6. N. CARLSON AND K. MILLER, Report PAM-342, Center for Pure and Applied Mathematics, University of California at Berkeley, 1986 (unpublished).
7. J. M. COYLE, J. E. FLAHERTY, AND R. LUDWIG, *J. Comput. Phys.* **62**, 26 (1986).
8. E. A. DORFI AND L. O'. C. DRURY, *J. Comput. Phys.* **69**, 175 (1987).
9. R. M. FURZELAND, Report TNER.85.022, Thornton Research Centre, Shell Research Limited, 1985 (unpublished).
10. R. J. GELINAS, S. K. DOSS, AND K. MILLER, *J. Comput. Phys.* **40**, 202 (1981).
11. A. C. HINDMARSH, in *Advances in Computer Methods for Partial Differential Equations-IV*, edited by R. Vichnevetsky and R. S. Stepleman (IMACS, New York, 1981), p. 312.
12. A. N. HRYMAK, G. J. MCRAE, AND A. W. WESTERBERG, *J. Comput. Phys.* **63**, 168 (1986).
13. J. M. HYMAN, Report LA-UR-82-3690, Los Alamos National Laboratory, 1982 (unpublished).
14. J. KAUTSKY AND N. K. NICHOLS, *SIAM J. Sci. Stat. Comput.* **1**, 499 (1980).
15. N. K. MADSEN, in *PDE Software: Modules, Interfaces and Systems*, edited by B. Engquist and T. Smedsaas (North-Holland, Amsterdam, 1984), p. 207.
16. K. MILLER AND R. N. MILLER, *SIAM J. Numer. Anal.* **18**, 1019 (1981).
17. K. MILLER, *SIAM J. Numer. Anal.* **18**, 1033 (1981).
18. K. MILLER, in *Adaptive Computational Methods for Partial Differential Equations*, edited by I. Babuska, J. Chandra, and J. E. Flaherty, (SIAM, Philadelphia, 1983), p. 165.
19. K. MILLER, in *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, edited by I. Babuska, O. C. Zienkiewicz, Z. Gago, and E. R. de A. Oliveira (Wiley, New York, 1986), p. 325.
20. K. MILLER, University of California at Berkeley, Department of Mathematics, private communications, 1988, 1989.

21. A. C. MUELLER AND G. F. CAREY, *Int. J. Num. Meths. Eng.* **21**, 2099 (1985).
22. V. PEREYRA AND E. G. SEWELL, *Numer. Math.* **23**, 261 (1975).
23. L. R. PETZOLD, *Appl. Numer. Math.* **3**, 347 (1987).
24. L. R. PETZOLD, in *Scientific Computing*, edited by R. S. Stepleman (IMACS/North-Holland, Amsterdam, 1983), p. 65.
25. R. F. SINCOVEC AND N. K. MADSEN, *ACM Trans. Math. Software* **1**, 232 (1975).
26. R. F. SINCOVEC AND N. K. MADSEN, *ACM Trans. Math. Software* **1**, 261 (1975).
27. J. G. VERWER, J. G. BLOM, AND J. M. SANZ-SERNA, *J. Comput. Phys.* **82**, 454 (1989).