JOURNAL OF COMPUTATIONAL PHYSICS **82**, 454–486 (1989)

An Adaptive Moving Grid Method for One-Dimensional Systems of Partial Differential Equations

J. G. VERWER AND J. G. BLOM

Centre for Mathematics and Computer Science, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

AND

J. M. SANZ-SERNA

Departamento de Matematica Aplicada y Computacion, Facultad de Ciencias, Universidad de Valladolid, Spain

Received: April 28, 1988

We describe a fully adaptive, moving grid method for solving initial-boundary value problems for systems of one-space dimensional partial differential equations whose solutions exhibit rapid variations in space and time. The method, based on finite-differences, is of the Lagrangian type and has been derived through a co-ordinate transformation which leads to equidistribution in space of the second derivative. Our technique is "intermediate" between static regridding methods, where nodes remain fixed for intervals of time, and continuously moving grid methods, where the node movement and the PDE integration are fully coupled. In our approach, the computation of the moving grids and the solution on these grids are carried out separately, while the nodes are moved at each time-step. Two error monitors have been implemented, one to govern the time-step selection and the other to eventually adapt the number of moving nodes. The method allows the use of different moving grids for different components in the PDE system. Numerical experiments are presented for a set of five sample problems from the literature, including two problems from combustion. © 1989 Academic Press, Inc.

1. INTRODUCTION

In this paper we describe a general method for the numerical solution of initial-boundary value problems for systems of partial differential equations (PDEs) in one space dimension. The class of problems considered have the form

$$u_t = L(u), \qquad x_L < x < x_R, \qquad t > t_0, \tag{1.1a}$$

$$u(x, 0) = u^0(x), \qquad x_L < x < x_R, \tag{1.1b}$$

$$g_L(x, t, u(x), u_x(x)) = 0, \qquad x = x_L, \qquad t > t_0, \tag{1.1c}$$

$$g_R(x, t, u(x), u_x(x)) = 0, \qquad x = x_R, \qquad t > t_0, \tag{1.1d}$$

where L represents a linear or nonlinear spatial differential operator of the second order. Of course, many problems from physics and other areas of application necessitate the solution of such systems, a task that must be undertaken numerically, except for the rare cases where analytical techniques are available. In recent years, several sophisticated MOL (method of lines) packages have been developed for one-space dimensional PDE systems like (1.1) (see, e.g., [3, 15]). These packages, which exploit the success of the automatic stiff ordinary differential equations solvers, operate in a semi-automatic way, in the sense that they automatically adjust the time-step but employ, throughout the computation, a fixed spatial grid, chosen by the user before the integration starts. Such a semi-automatic approach is very efficient not only in cases where the solution does not exhibit much spatial activity and a uniform grid is adequate, but also in problems where the regions of rapid variation in space do not move and are known a priori, so that a graded mesh can be suitably positioned. However, for solutions possessing sharp moving spatial transitions, like travelling wavefronts or emerging layers, a grid held fixed throughout the calculation can be computationally inefficient, since, to afford a mildly accurate approximation, such a mesh would easily have to contain hundreds or even thousands of nodes. In such cases, adaptive and moving grid methods, which adjust automatically both the space and the time-stepsizes, are usually more efficient.

The finite-difference, moving grid Lagrangian method developed in this paper has been designed for the efficient computation of solutions containing very sharp spatial and temporal transitions, like those arising in many combustion problems. Of course, the method can also be applied to compute less challenging solutions, but then it is likely not to be competitive with fixed grid MOL algorithms. We would also like to emphasize that we are not concerned here with genuinely discontinuous shock solutions as those arising in hyperbolic problems, but rather with solutions with extremely large but finite derivatives.

The present paper follows our two earlier contributions [5, 6], where we have studied several finite-difference, Lagrangian moving grid schemes. These schemes are "intermediate" between the static regridding methods [14, 22–24], where nodes remain fixed for intervals of time, and continuously moving grid methods, where the node movement and the PDE integration are fully coupled [2, 10, 17, 18, 20, 26]. While the research in [5, 6] has enabled us to identify a promising scheme, the implementation considered in those papers used fixed time-steps and did not allow a dynamic variation of the number of spatial grid-points. Besides, the numerical experiments reported only referred to the Burgers' equation. Therefore, the material in [5, 6] only provided a first step toward our ultimate goal in this area: the development of a user-oriented, fully automatic code, applicable to a wide variety of problems. In the present work, we describe how to incorporate variable time-steps and how to vary the number of grid-points. Furthermore we discuss the successful application of our method to five sample problems from the moving grid literature, including two interesting and difficult models from combustion theory.

At each integration step of our algorithm, two simple error monitors are computed. One of them governs the time-step selection and the other the location and, eventually, the number of space nodes. Thus the method not only automatically adjusts the space grid to regions of high spatial activity, but also provides a facility to adapt the number of nodes in order to meet a user-specified tolerance. This adaptation is embedded in the generation of the new space nodes at the forward time-level, which is based on equidistribution of the second space derivative. It should be emphasized that the dynamic adaptation of the number of nodes is, to some extent, of secondary importance because, even if the quantity of spatial nodes is held fixed, the nodes move to cater for the spatial activity of the solution. Another facility of our algorithm is that different spatial grids can be chosen for different PDE components. This avoids the use of finely meshed zones in regions where, componentwise, they are not needed, but introduces overhead costs originating from the more complicated linear algebra and the extra interpolation tasks.

Above, we used the term error monitor instead of error estimator in order to emphasize that the quantities involved are not approximations to true local errors, but only heuristic, cheap means for efficiently computing rapidly varying solutions of a widely different nature by keeping at a fairly acceptable level both the number of space nodes and the number of time-steps.

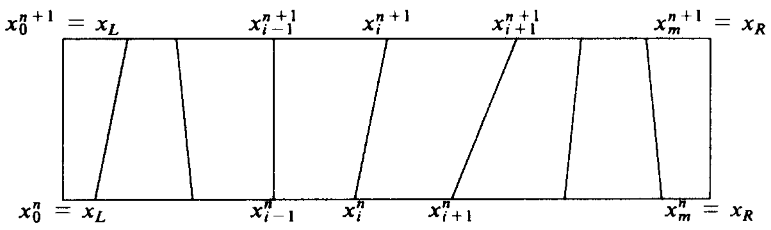
In the next section we outline the Lagrangian method underlying our fully adaptive moving grid procedure. In Section 3 we derive the error monitors which govern the time-step and number of space nodes selection. In Section 4 we briefly discuss the possibility of using different moving grids for different PDE components. Results of extensive numerical testing are presented in Section 5 and Section 6 is devoted to final comments and conclusions.

2. The Method of Solution

Until further notice it is assumed that the method is applied using the same spatial grid for all components of the solution u.

2.1. *The Time-Stepping Scheme*

We advance the solution in time over a trapezoidal space-time grid



Trapezoids covering the strip $x_L < x < x_R$, $t_n < t < t_{n+1}$. (2.1) by means of the Lagrangian time-stepping scheme

$$\begin{aligned}(\theta(x_{i+1}^{n+1}-x_{i-1}^{n+1})+(1-\theta)(x_{i+1}^{n}-x_{i-1}^{n}))&\left(\frac{u_i^{n+1}-u_i^n}{\tau}\right)\\ -(\theta(u_{i+1}^{n+1}-u_{i-1}^{n+1})+(1-\theta)(u_{i+1}^{n}-u_{i-1}^{n}))&\left(\frac{x_i^{n+1}-x_i^n}{\tau}\right)\\ =\theta(x_{i+1}^{n+1}-x_{i-1}^{n+1})L_{h,i}(u^{n+1})+(1-\theta)(x_{i+1}^{n}-x_{i-1}^{n})&L_{h,i}(u^n).\end{aligned} \tag{2.2}$$

The notation u_i^n represents the discrete approximation to the value $u(x_i^n, t_n)$ and $L_{h,i}$ stands for a suitable finite-difference replacement of the spatial differential operator L. In our current implementation $L_{h,i}$ is obtained by replacing in L the operators $\partial/\partial x$ and $\partial^2/\partial x^2$ by standard central differences, but other choices for $L_{h,i}$ are clearly possible. The index i varies between 1 and $m-1$, where m is the number of trapezoids covering the strip (2.1). As usual, t_n and $t_{n+1}=t_n+\tau$ are consecutive time-levels. The time-step τ may depend on n, although this dependence is often not reflected in the notation. Note that if the grid in (2.1) is rectangular, i.e., if there is no grid motion, then (2.2) reduces to the familiar θ-rule. The parameter values $\theta=1$ and $\theta=\frac{1}{2}$ yield the Lagrangian implicit Euler and Lagrangian Crank–Nicolson schemes, respectively. These are the only values for θ we consider. Obviously, the scheme must be supplemented with boundary conditions.

The scheme (2.2) may be derived as follows. Let (s, T) be new independent variables linked to the old independent variables (x, t) through a co-ordinate transformation

$$x=x(s, T), \qquad t=T, \qquad 0<s<1, \quad T>0. \tag{2.3}$$

The Lagrangian form of (1.1a) is obtained by expressing u_t in terms of u_T,

$$u_T-u_x x_T=L(u), \qquad 0<s<1, \quad T>0. \tag{2.4}$$

The scheme (2.2) is now derived by first multiplying (2.4) by $\partial x/\partial s$ to obtain

$$x_s u_T-u_s x_T=x_s L(u), \qquad 0<s<1, \quad T>0, \tag{2.5}$$

followed by standard central differencing on the uniform s-grid $\{s_i=ih,\ 0\leqslant i\leqslant m,\ h=1/m\}$.

In [6] we have shown the close relation between the scheme (2.2) and a finite-element scheme using piecewise linear approximations over trapezoidal space-time elements, due to Bonnerot and Jamet [7] (see also Davis and Flaherty [9]). However, the Bonnerot–Jamet scheme may suffer from a harmful form of instability [6] and the finite-difference scheme (2.2) is free from that drawback. Experimentation with the nonlinear Burgers' equation has indicated that the Crank–Nicolson form (2.2) performs somewhat better than the slightly different Crank–Nicolson scheme which would result from differencing (2.4), rather than the less natural form (2.5). For the backward Euler case, both forms lead to the same difference expression.

The basic idea of the Lagrangian approach is to choose the variables (s, T) so that, with them, the problem becomes easier to handle numerically than it was with the original pair (x, t). Note that although the new time T equals the old time t, the derivatives $\partial u/\partial t$ and $\partial u/\partial T$ are different. The former measures the changes of u as a function of t at a fixed x-value (Eulerian description), the latter at a fixed s-value (Lagrangian description). Thus the choice of an appropriate new spatial variable s may:

(i) Soften the spatial behaviour of the solution, via the concentration of $x(s, T)$ trajectories in those regions where u varies rapidly as a function of x.

(ii) Soften the temporal behaviour of the solution. This would happen if, through a right choice of transformation, the Lagrangian derivative u_T can be made significantly smaller than the original Eulerian derivative u_t.

Ideally, we would like to find s and T in such a way that the solution of (2.5) does not possess fast transitions in space *and* time, and therefore can be integrated with fairly large time-steps on a coarse uniform s-mesh. This mesh defines, via the transformation $x = x(s, T)$, a moving, nonuniform x-mesh, which should allow an efficient integration. However, the nature of the solution u being approximated dictates to what extent the aims (i) and (ii) above can be simultaneously achieved. We shall illustrate this point when discussing our test problems.

Although the introduction of the variables s and T is helpful in the derivation and understanding of (2.2), it should be emphasized that (2.2) can also be regarded as a consistent discretization of the original (Eulerian) equation (1.1a) on the mesh (2.1) in the (x, t)-space, *regardless of the choice of the x-grid points.* This remark is relevant because in practice the grids must, of course, be determined along with the computation of the numerical solution and therefore the x-grid points actually used are subjected to errors and do not quite stem from a smooth transformation of a uniform s-grid. This lack of smoothness of the computed grid-points is one of the biggests problems in the development of general moving grid procedures, particularly so as far as error estimation is concerned. Note also that a rough $u(x, t)$ solution can only become smooth in the (s, T) variables if the roughness is transferred to the transformation $x = x(s, T)$ and this implies that, in the cases we are interested in, finding the "exact" grid positions is likely to be an ill-conditioned task.

2.2. *The "Intermediate" Approach*

Concerning the grid determination, our algorithm can be classified as belonging to the class of methods which are "intermediate" between the static regridding methods, where nodes remain fixed for intervals of time [14, 22–24], and continuously moving grid methods, where the node movement and the PDE integration are fully coupled [2, 10, 17, 18, 20, 26]. We have successfully applied this "intermediate" approach in [5, 6].

Given an x-grid at the nth time-level and the corresponding numerical solution, stepping to the $(n+1)$th time-level involves two successive computational stages:

The grid prediction stage which computes the grid at the forward $(n+1)$th level. First an implicit Euler step is performed on a fixed spatial grid (i.e., (2.2) is applied with $\theta = 1$ and $x_i^{n+1} = x_i^n$). The implicit Euler solution then acts as input for a de Boor [8] regridding algorithm which generates the grid-points at the advanced time-level by equidistributing a chosen monitor function. This equidistribution defines implicitly the co-ordinate transformation $x = x(s, T)$.

The integration stage which computes the approximations u_i^{n+1} according to (2.2) with $\theta = \frac{1}{2}$. Other values of θ also result in good performances, but our numerical experience shows that $\theta = \frac{1}{2}$ is in general more efficient, as it produces smaller errors in time.

The "intermediate" approach has some clear advantages. The node movement is easier to deal with than in a continuously moving grid method, where mesh tangling and ill-conditioning of the arising systems of algebraic equations are well-known threats. With de Boor's technique points cannot cross or leave the domain. In a sense, due to the explicitness of that technique, one has a more direct control over the grid movement than that achieved with penalty functions in the continuous approach. On the other hand, the intermediate approach very often allows time-steps significantly larger than those used by static regridding methods, which must operate with (larger) Eulerian derivatives. In this connection it is fair to mention that the intermediate approach precludes, to some extent, the full exploitation of the advantages of small Lagrangian derivatives u_T, because the grid prediction stage is carried out anyway on a nonmoving grid and is likely not to allow very large time-steps. However, the output of the static grid prediction stage is only used for finding the new grid and plays no direct role in the computation of the new solutions, and it turns out that, in practice, inaccuracies in the grid prediction do not greatly impair the performance of the overall procedure. In fact, the intermediate approach, as implemented in our current algorithm, is remarkably robust and allows the use of sufficiently large time-steps.

Turning now to computational costs, observe that the intermediate approach results in systems of algebraic equations of a smaller dimension than those necessary in continuously moving methods, where unknown x and u values are coupled. A drawback of the intermediate approach is that, per entire step, two systems of algebraic equations must be solved, due to the use of implicit formulas at both stages. (The use of an explicit prediction formula is not advisable [5, 6] because it may damage the robustness of the algorithm.)

2.3. *The Regridding*

As mentioned above, the actual regridding is carried out at the grid prediction stage and effects the co-ordinate transformation (2.3) which underlies the Lagrangian approach. We employ a transformation based on equidistribution of a second derivative monitor function, but, of course, other choices are conceivable. The smoothing capabilities of the used $x = x(s, T)$ transformation have been illustrated in [5] in the case of Burgers' equation. There we observed that, for moving front solutions, our "intermediate" algorithm moves the nodes in a Lagrangian fashion with the true speed of the fronts. We emphasize that this front tracking is achieved automatically by the algorithm and not via a user-supplied co-ordinate transformation. This capability is shared by other moving grid methods (e.g., the moving finite-element method [17, 18] and Petzold's finite-difference method [20]).

While the new pair of variables (s, T) is central in the theoretical derivation of the moving grid scheme, it should be stressed that the actual computation of the grids is completely achieved in terms of the physical variable pair (x, t), by using the cheap (inverse interpolation) procedure of de Boor [8]. As already mentioned, an attractive feature of this procedure is that, due to the explicit construction, the node ordering is always maintained so that nodes cannot cross each other or leave the space interval. Our current version of the de Boor algorithm is similar to that used in [5, 6], except for a slight change in the monitor function. For brevity, our description of the algorithm will be very sketchy and further details can be seen in the papers [5, 6].

The transformation is defined by

$$s(x, t) = \int_{x_L}^{x} M(\xi, t)\, d\xi/\eta(t), \qquad \eta(t) = \int_{x_L}^{x_R} M(\xi, t)\, d\xi, \tag{2.6}$$

where M is the second derivative monitor function

$$M(\xi, t) = \alpha + \sqrt{|u_{xx}(\xi, t)|}. \tag{2.7}$$

Note that, if u is vector valued, the symbol $|\cdot|$ in (2.7) is to be interpreted as a norm, for example, a weighted Euclidean norm.

If the x-grids arise from an equidistant s-grid, via the inverse transformation $x = x(s, t)$, then

$$\int_{x_i}^{x_{i+1}} M(\xi, t)\, d\xi = \eta(t)[s(x_{i+1}, t) - s(x_i, t)] = \eta(t)/m, \tag{2.8}$$

for $0 \leqslant i \leqslant m-1$. Hence the x-grid has the property that on each of its subintervals the average of the monitor function has the same value (equidistribution of M). As time evolves, this transformation causes grid trajectories to migrate to regions of high spatial activity, as governed by the choice of M. The parameter α serves to regularize the transformation in regions where the solution u is very flat, i.e., where $\partial^2 u/\partial x^2$ is nearly or truly zero. Hence its magnitude is related to the number of points to be placed in regions where, in space, the solution varies relatively slowly. Of further interest is that $\eta(t)$ may provide the basis for a heuristic space error monitor, which would suggest when to increase or decrease m. We shall discuss this in greater detail in the next section.

3. The Variable Time-Step and Variable Number of Nodes Procedures

In [5, 6] we reported some promising numerical results corresponding to Lagrangian methods implemented with a constant time-step τ and a fixed number of nodes m. However, it is clear that the efficient treatment of many practical problems requires that τ be varied in the course of the integration in accordance to the local (in time) behaviour of the solution. An important example is given by combustion problems where sudden ignitions are interspersed with periods of time in which less action takes place. Likewise, it may be desirable to change m as time evolves, although the need for this option is less because, even with m fixed, the method automatically makes the spatial grid finer in regions of high activity. In this section we describe cheap monitors for the dynamic selection of appropriate values for τ and m, and our strategies for the implementation of changes in τ and m.

3.1. *The Variable Time-Step Procedure*

3.1.1. *The Time Error Monitor*

The time-step selection is based on a local error expression. As mentioned before, the scheme (2.2) can be regarded either as a discretization of (2.5) on a uniform s-grid or as a discretization of (1.1a) on the nonuniform, time-dependent x-grid (2.1). Correspondingly, for a local error analysis of (2.2) two approaches can be followed:

(i) In the first, Taylor expansions are carried out in the Lagrangian coordinates (s, T) and the numerical approximations for u are interpreted as approximations to the true PDE solution values $u(s, T) = u(x(s, T), T)$ on the uniform s-grid. By following this approach, an expression for the local error in time of (2.2) is obtained which contains not only Lagrangian derivatives of the solution u, but also partial derivatives of the co-ordinate transformation. This approach would certainly be meaningful if the x-grid positions were actually derived through an analytically defined transformation $x = x(s, t)$. However, we have already mentioned that in practice the grid positions are subjected to (sometimes large) errors, so that the assumption $x_i = x(i/m, t)$ which underlies the approach is far from being satisfied.

(ii) The second approach is to expand in the physical coordinates (x, t) and thus to interpret the approximations to u as approximations in the numerical node values actually used. Following this interpretation there is no need to cater in the analysis for errors in the x-grid positions.

Numerical experimentation has shown that the second approach should be favoured and the following analysis is based on it.

We first consider the scheme (2.2) with $\theta = 1$ on the grid (2.1) (recall, however, that the implemented algorithm actually uses $\theta = \frac{1}{2}$). The notation u^n represents, for the time being, the exact solution restricted to the level n grid. We introduce the *space* local truncation error [25]

$$\alpha_i(t_n) = u_t(x_i^n, t_n) - L_{h,i}(u^n), \tag{3.1}$$

which originates from replacing the true differential operator L by the finite-difference operator on the numerical grid. Next we introduce the *full* local truncation error [25] (boundary conditions are ignored)

$$\beta_i(t_{n+1}) = u_i^{n+1} - u_i^n - \tau L_{h,i}(u^{n+1}) - \frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{x_{i+1}^{n+1} - x_{i-1}^{n+1}} (x_i^{n+1} - x_i^n). \tag{3.2}$$

The error (3.2) is the defect which arises by substituting the true PDE solution into the numerical scheme. In this section we are only interested in the time-discretization contribution to (3.2), i.e., in the part of (3.2) which cannot be made smaller by suitably refining the spatial grid.

We work under the very reasonable assumption that, as the space-time grid is refined, a constant C exists such that for all grids and for all i, n,

$$|r_i^n| \leqslant C\tau, \tag{3.3}$$

where

$$r_i^n = x_i^{n+1} - x_i^n; \tag{3.4}$$

thus the node velocity r_i^n/τ is assumed to be bounded independently of the number of points in the grid and of the time-step.

We first introduce the auxiliary quantity

$$\gamma_i(t_{n+1}) = u_i^{n+1} - u_i^n - \tau u_t(x_i^{n+1}, t_{n+1}), \tag{3.5}$$

and Taylor expand at the point (x_i^{n+1}, t_{n+1}), taking (3.3) into account, to get

$$\gamma_i(t_{n+1}) = -1/2\tau^2 u_{tt} - r\tau u_{xt} - \tfrac{1}{2} r^2 u_{xx} + r u_x + O(\tau^3), \tag{3.6}$$

where r denotes r_i^n. Next we write

$$\frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{x_{i+1}^{n+1} - x_{i-1}^{n+1}} = u_x + FDE_i^{n+1}, \tag{3.7}$$

where FDE_i^{n+1} is the (space) error involved. Hence, putting $p = x_{i+1} - x_i$ and $q = x_i - x_{i-1}$,

$$FDE_i^{n+1} = (1/2)(p-q) u_{xx} + \tfrac{1}{6}(p^2 - pq + q^2) u_{xxx} + \cdots. \tag{3.8}$$

On taking (3.1), (3.5), and (3.7) into (3.2) and considering the expansions (3.6)–(3.7), we conclude

$$\beta_i(t_{n+1}) = EST_i^{n+1} + \eta_i(t_{n+1}) + O(\tau^3), \tag{3.9}$$

where $\eta_i(t_{n+1})$ is the total spatial contribution

$$\eta_i(t_{n+1}) = \tau\alpha_i(t_{n+1}) - rFDE_i^{n+1}, \tag{3.10}$$

and EST_i^{n+1} represents the leading part of the time contribution to $\beta_i(t_{n+1})$ at (x_i^{n+1}, t_{n+1}):

$$EST_i^{n+1} = -1/2\tau^2 u_{tt} - r\tau u_{xt} - \tfrac{1}{2} r^2 u_{xx}. \tag{3.11}$$

Note that even though the expression (3.11) was derived in the (x, t)-space, it reflects nicely the Lagrangian nature of the scheme. For instance, if u is a travelling wave solution $u(x, t) = w(x - ct)$, EST vanishes at those grid-points which have the correct speed c. On the other hand, (3.11) involves Eulerian derivatives, which, in the applications we are interested in, are likely to be extremely large. Therefore the terms in the right-hand side of (3.11) may be individually very large and partially cancel to yield a small EST. Under those circumstances, it is clear that EST is numerically ill-defined and cannot be expected to be computed too accurately. As a further difficulty, note that the estimation of u_{tt} requires that the numerical solution at time-level $n-1$ should be kept in storage and that its estimation requires interpolation, since the u values available at different time-levels correspond to different x locations. This interpolation provides an extra source of inaccuracies in the computation of EST.

The corresponding error expression for the Crank–Nicolson scheme actually used involves Eulerian derivatives of higher order and therefore would lead to practical problems even more severe than those just cited for the backward Euler case. We have decided to also use (3.11) for the Crank–Nicolson scheme, so that our estimations should be expected to be conservative. After all these considerations, (3.11) should be regarded as a sort of monitor, rather than as a true estimator. The numerical experiments reported later show that EST is indeed a successful monitor.

3.1.2. *The Implemented Strategy*

After the step up to t_{n+1} has been completed, the estimates EST_i^{n+1} at the individual grid-points are computed and then normed over the space mesh to get

$$NEST = \left(\sum_{i=0}^{m-1} \left(\frac{x_{i+1}^{n+1} - x_i^{n+1}}{2} \right) \left(|EST_i^{n+1}|^2 + |EST_{i+1}^{n+1}|^2 \right) \right)^{1/2} \tag{3.12}$$

where $EST_0^{n+1} = EST_m^{n+1} = 0$.

The variable time-step strategy is similar to that in most current ODE codes. $NEST$ is subjected to the test $NEST \leqslant TOLT$, where $TOLT$ is a user-specified tolerance parameter. If the test is passed, both the grid and the numerical solution at the $(n+1)$th level are accepted and the new stepsize is computed from

$$\tau_{n+1} = Q((TOLT/NEST)^{1/2})\tau_n, \qquad \tau_n = t_{n+1} - t_n. \tag{3.13}$$

$Q(z)$ is a threshold function designed to prevent the stepsize from zigzagging and to prevent future step rejections. $Q(z)$ varies, for $z \geqslant 1$, from 0.8 to 2.0 in a piecewise linear way. Hence, at most, the stepsize may be doubled and when the test is barely passed the stepsize decreases by 20%. Although the control is based on an absolute error test, it can, of course, be changed into a mixed absolute-relative test.

If $NEST > TOLT$, we have a step failure. Then we not only discard the values u^{n+1} just computed, but also the past values u^n. This backstep is justified by the fact that the expression for the error monitor (3.11) is only approximately known, which might imply that upon a step failure we have been a little too optimistic in one or more previous successful steps. The backstep also provides an extra safety margin in cases where, suddenly, the solution starts changing very rapidly, as is the case in most combustion problems. Finally, there is no explicit check on the quality of the grids computed in the grid prediction stage; when a step failure occurs, we must reckon with the possibility that this may be partly due to a not very favourable location of the nodes. By backstepping we enhance the chance of timely locating the nodes in good positions. When re-attempting the step $t_{n-1} \to t_n$, after a rejection, we use

$$\tau_{n-1} := 0.5\tau_{n-1} = (t_n - t_{n-1})/2 \qquad \text{and} \qquad x_i^n := (x_i^n + x_i^{n-1})/2; \tag{3.14}$$

in this way we save one grid prediction stage. After a rejection the threshold function $Q(z)$ is adapted in order to avoid a too rapid increase in stepsize, which might lead to a new step failure. Finally, backstepping is of course not possible at the start of the process. If the initial stepsize or the first stepsize after a backstep turns out to be too large, the step is simply redone while using (3.13) as stepsize estimate until step acceptance.

Note that, if u is vector-valued, the stepsize procedure can be applied componentwise. After a successful step, the stepsize is set equal to the minimum of (3.13) over all the solution components.

3.2. *The Variable m Procedure*

3.2.1. *The Space Error Monitor*

Our space error monitor is derived from the quantity η in (2.6). Define

$$m_{\text{var}} = \frac{\int_{x_L}^{x_R} \sqrt{|u_{xx}|}\, d\xi}{\sqrt{TOLS}}, \qquad \alpha = \frac{m_{\text{flat}} \sqrt{TOLS}}{x_R - x_L}, \tag{3.15}$$

where $TOLS$ stands for a tolerance parameter and m_{flat} for a nonnegative integer parameter to be specified later. Then

$$\eta(t) = \int_{x_L}^{x_R} \alpha + \sqrt{|u_{xx}(\xi, t)|}\, d\xi = m_{\text{flat}} \sqrt{TOLS} + m_{\text{var}} \sqrt{TOLS}. \tag{3.16}$$

The equidistribution property (2.8) implies

$$\int_{x_i}^{x_{i+1}} \sqrt{|u_{xx}(\xi, t)|}\, d\xi = \eta(t)/m - (x_{i+1} - x_i)\alpha \leqslant \eta(t)/m \leqslant \sqrt{TOLS}, \tag{3.17}$$

if m is chosen as (square brackets denote integer part)

$$m = [m_{\text{var}}] + m_{\text{flat}} + 1. \tag{3.18}$$

Neglecting the quadrature error involved, (3.17) then yields for all subintervals

$$(x_{i+1} - x_i)^2\, |u_{xx}(x_{i+1/2}, t)| \leqslant TOLS. \tag{3.19}$$

Thus, defining m by (3.18) implies that the quantity $(\Delta x)^2\, |u_{xx}|$ is always kept below the prescribed tolerance *TOLS*. The dimensionless integer m_{flat} is a user-defined parameter, to be interpreted as the approximate number of nodes which one would use if the solution were completely flat ($u_{xx} = 0$). We emphasize that criterion (3.19) is merely heuristic. In all applications the genuine spatial error, both local and global, will be essentially more complicated (cf. (3.10)). In a sense, the present monitoring is based on the degree of spatial difficulty of the problem thus assuming that the size of the second derivative is an appropriate measure in this respect.

3.2.2. *The Implemented Strategy*

The space grid control is carried out at each successful time-step just after the time-step control. In case of a step rejection ($NEST > TOLT$) no grid control is necessary because then the next grid to be used is defined by (3.14). When the formula (3.15) dictates a decrease or increase in m_{var} and hence in m, a de Boor loop is made to define the new nodal positions and the solution is transferred to the new grid by interpolation. In the implemented algorithm, simple linear interpolation is used, as it is feared that higher-order interpolants are likely to be unsuitable to interpolate rough functions on course grids [22].

A minimal value for m is prescribed and, further, measurements are taken in order to avoid minor changes of m. No change is performed if the newly computed m_{var} is between 0.5 and 1.2 times the current value, and, when an increase is made, the new m_{var} is at least 50% larger than the old. While changing stepsize in time is trivial due to the one-step nature of (2.2), changing the number of nodes may be troublesome. First, we cut off all the existing grid trajectories and continue the time-stepping on an entirely new grid. This may lead, until x-values and u-values adjust to each other, to some transient oscillations in the grid trajectories, due to the sensitivity of the estimation of (2.7) in solutions with steep fronts. Second, the interpolation associated with changing m is clearly a source of errors. (The fact that Lagrangian schemes with fixed m do not require interpolation is one of their advantages when compared to static regridding methods.)

The present straightforward heuristic strategy works satisfactorily and is almost free of extra costs. Due care should be exercised when choosing the parameters, since the dangers of operating with coarse tolerances are well known. In particular, although tempting, one should not choose too small values for the allowed minimum value of m in problems where at the initial time the solution starts out flat and later develops, very rapidly, large spatial gradients. In such a situation the grid may be too coarse to see timely the onset of the variations (the choice of $TOLT$ also plays a role in this connection). Even if this mismatch between the value of m and the behaviour of the solution u happens only for a few time-steps, inaccuracies may be introduced which at best are maintained over the remainder of the time interval, and at worst are greatly amplified by the time evolution. It should be mentioned that the previous considerations are not peculiar to the current heuristic estimators. Any estimation procedure, whether based on genuine error expressions or not, requires sufficient grid-points for estimating the involved quantities up to sufficient accuracy.

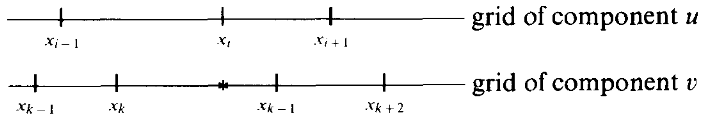
4. The Use of Different Moving Grids for Systems

In the "intermediate" approach the grid is determined at each time-level by means of an explicit de Boor procedure. It is therefore straightforward to compute different grids for the different components of the solution of a system of PDEs, in order to avoid the frequent sampling of a solution component in regions where that component varies slowly. This idea is not without difficulties. First, the block tridiagonal coupling which exists between solution values when a single grid is used is disturbed. In fact, the structure of the coupling is likely to change in time, due to the moving nature of the grids. Therefore, for solving the systems of linear algebraic equations which arise in the application of the Newton process, an efficient sparse matrix routine must be used instead of a less sophisticated band solver. This obviously leads to overhead costs which partly annihilate the anticipated savings. We have used the NAG routine F04AXF due to Duff [1, 11].

A second problem, of a more serious nature, concerns the spatial finite-difference operator. Let us consider the case of two components, denoted by u and v, and suppose that the first component of the differential operator L in (1.1a) is in the generic form

$$u_t = f(x, t, u, v, u_x, v_x, u_{xx}, v_{xx}). \tag{4.1}$$

The figure below displays a typical section of the grids



To approximate the right-hand side of (4.1) at the point $x = x_i$, the terms u, $\partial u/\partial x$

and $\partial^2 u/\partial x^2$ pose no difficulty; $\partial u/\partial x$ and $\partial^2 u/\partial x^2$ are differenced by means of the standard replacement based on x_{i-1}, x_i, x_{i+1}. However, v, $\partial v/\partial x$, and $\partial^2 v/\partial x^2$ are evaluated at x_i by linearly interpolating their respective approximations at the neighbouring points x_k and x_{k+1}, obtained by differencing in the v-grid. If x_i lies in one of the two v-grid subintervals adjacent to the boundary, linear interpolation of 3-point differences is not possible and we have resorted to piecewise constant interpolation, as any other alternative would lead to an increase in the coupling.

If composite expressions of variables living on different grids have to be approximated, such as the conservation law form $\partial u/\partial t = \partial(uv)/\partial x$, then it is not possible to first difference the variables on their own grid and then interpolate. For approximating $\partial(uv)/\partial x$ at the u-grid, one first has to interpolate the neighbouring v-values and then difference, a procedure which may lead to large spatial errors. The situation is even worse for $\partial^2(uv)/\partial x^2$, where 3-point differencing of linear interpolates results in an inconsistent replacement, and one should differentiate to get $u_{xx}v + 2u_x v_x + uv_{xx}$ and then approximate the individual terms.

These difficulties make us somewhat reluctant to advocate the multiple grid option. It is to be feared that in many cases the anticipated savings in grid-points will not make up for the drawbacks of this option. However, the underlying idea deserves some attention and in Section 5 we report a successful numerical illustration.

5. Numerical Results

We shall present results of extensive numerical experiments on a set of five sample problems from the literature, including two problems from combustion theory. Some implementation details are given first.

5.1. *Implementation Details*

The Newton Solver

Each time-step involves the solution of two different sets of nonlinear algebraic equations. As is the case with stiff ODE codes, the efficiency in the solution of the nonlinear equations is partly determined by the time-step selection process; a larger stepsize leads to larger Jacobian and solution variations over the step and thus works against the easy solvability of the equations. While in stiff ODE packages it is usually possible not to update the Jacobian for a number of consecutive steps, in our situation we update the Jacobian each time a new system is to be solved, i.e., twice per time-step. The reasons for this strategy are as follows. We have already noted that the transformation $x = x(s, T)$ possesses large derivatives. As a result, the nodal positions can be expected to change significantly over a single step, which in turn leads to large variations in the Jacobian. Furthermore, the numerical approximation corresponding to a given grid-point will change substantially over a time-step if the grid-point enters or leaves a steep layer. This difficulty is made all the worse by the fact that Lagrangian methods are constructed to operate with relatively large time-steps.

The Jacobian matrices are computed by numerical differentiation. We readily admit that the cost per step of our algorithm is high and, undoubtedly, improvements in the nonlinear equation solution will be beneficial for the final performance of the method.

A Newton iteration is terminated when a Newton correction is found which, in the maximum norm, is less than a user-prescribed tolerance *TOLN*. Note that the maximum norm provides a rather stringent choice. If convergence does not take place when five Newton steps have been taken, we have a Newton failure. If the failure takes place at the grid prediction stage of the step from t_n to $t_{n+1} = t_n + \tau_n$, we set $\tau_{n\text{new}} := \tau_n/2$ provided that $\tau_{n\text{new}}$ is larger than half the previous successful stepsize τ_{n-1}. If this last condition does not hold, then, to prevent rapid decreases in stepsizes, we go back to t_{n-1} and apply (3.14), just as we do if the time-accuracy test fails. If a Newton failure takes place at the moving grid stage of the step, we go back to t_{n-1}, since then the last computed grid is likely to be wrong.

There remains to mention the choice of the initial estimates. The converged implicit Euler solution of the grid prediction stage is ordinarily, after linear interpolation to the new grid, a satisfactorily initial estimate for the solution of the Lagrangian scheme (2.2). For the fixed grid Euler solution, a natural candidate for initial estimate is the converged Lagrangian solution u^n of the nth level. However, the stepsize is determined by the variable stepsize procedure for (2.2). Since, for a given accuracy, the moving grid scheme (2.2) has the potential of taking a larger stepsize than the static implicit Euler, the current stepsize can be too large for the next static step, which in turn can lead too quickly to a Newton failure. Clearly, it is undesirable to accept smaller stepsizes merely to accommodate the iterative Newton process of the static step. To alleviate this difficulty, we use, as initial estimate, the values, obtained by linear interpolation, of u^n at the grid-points of the $(n-1)$th level, a procedure, which, in the case of running wave solutions, gives more accurate predictions, provided that the grid-speeds are correct. For $n = 0$ we use the given initial vector.

The Choice of the Initial Grid

When the solution profile at the initial time-level contains large gradients, the choice of the initial grid deserves attention (see [5, 6]). In the sample problems below this is not the case, except perhaps for Problem V. For Problems II, III, and IV, an equispaced initial grid has been chosen. The initial grids for Problems I and V are nonuniform and have been computed with the de Boor algorithm.

Method Parameters

To assist in reading the following material, we recall the parameters in the method. First of all we have the Newton tolerance *TOLN*, the time-step parameter *TOLT*, and the initial time-step. For the spatial monitor there are two options. In the first option, the user must specify the total, fixed number of grid-points m and the regularization parameter α. In the second option m is varied dynamically and the user has to prescribe m_{flat} and *TOLS*, along with the minimum allowable value for m. In all the experiments we have kept $\alpha = 1/(x_R - x_L)$ and $m_{\text{flat}} = 4$. No doubt a fine tuning of these parameters would have enhanced the performance of the method, but we have chosen to only use either m or *TOLS* to control the grid.

Table Information

The tables to be presented display the following information:

NODES (max, min, aver) = the maximum, minimum, and average number of nodes over all time-steps.

STEPS = the number of successful, variable time-steps needed to complete the integration over the given time interval.

BS = the accumulated number of all backsolves.

ETF = the number of times the stepsize was reduced due to a failure of the time error test.

NTF = the number of times the stepsize was reduced due to a Newton failure.

JACS = the total number of Jacobian evaluations.

5.2. *Problem I: Burgers' Equation*

We consider the well-known Burgers' equation

$$\partial u/\partial t = -\partial f(u)/\partial x + \varepsilon \partial^2 u/\partial x^2, \qquad 0 < x < 1, \quad t > 0, \quad f(u) = u^2/2, \quad \varepsilon = 0.001,$$

and prescribe the smooth initial function $u(x, 0) = \sin(\pi x)$ and homogeneous Dirichlet conditions at $x = 0$ and $x = 1$. The solution is a wave that steepens and moves to the right until a layer is formed at the end point $x = 1$. This takes place for $t \approx 0.6$. Then, the solution slowly decays to zero, while the layer remains near $x = 1$. This problem is not as difficult as the other four. The analytical solution is available, but takes the form of a series not amenable to easy numerical computation. For each of the five problems we have computed by brute force an "exact" reference solution, which, in the plots, appears as a solid line.

TABLE I

Problem I. Integration Information

	NODES							
Cases	Max	Min	Aver	STEPS	BS	JACS	ETF	NTF
(i)			10	22	188	47	0	3
(ii)			20	27	190	56	2	0
(iii)	19	11	15	18	158	38	0	3
(iv)	34	18	25	28	199	59	1	2

Using an initial time-step of $\frac{1}{50}$ and $TOLN = 10^{-4}$, we have integrated this problem with the Crank–Nicolson scheme four times over the time interval $0 \leqslant t \leqslant 2$: (i) $m = 10$ and $TOLT = 0.1$; (ii) $m = 20$ and $TOLT = 0.025$; (iii) $TOLS = TOLT = 0.1$; (iv) $TOLS = TOLT = 0.025$. The allowed minimum for m in cases (iii) and (iv) is 10. In all four cases the algorithm has performed very well. Figure 5.1 depicts the grid trajectories with the computed solutions for the cases (ii) and (iii) at times $t = 0.6$ and $t = 2.0$. The solution of case (ii) is very accurate (graphically identical to the "exact" solid line solution). The crude choice for $TOLT$ and $TOLS$ made for case (iii), also yields a quite satisfactory approximation, although less accurate than that from run (ii). With respect to accuracy, cases (i) and (iv) are comparable to (iii) and (ii), respectively. Table I contains integration

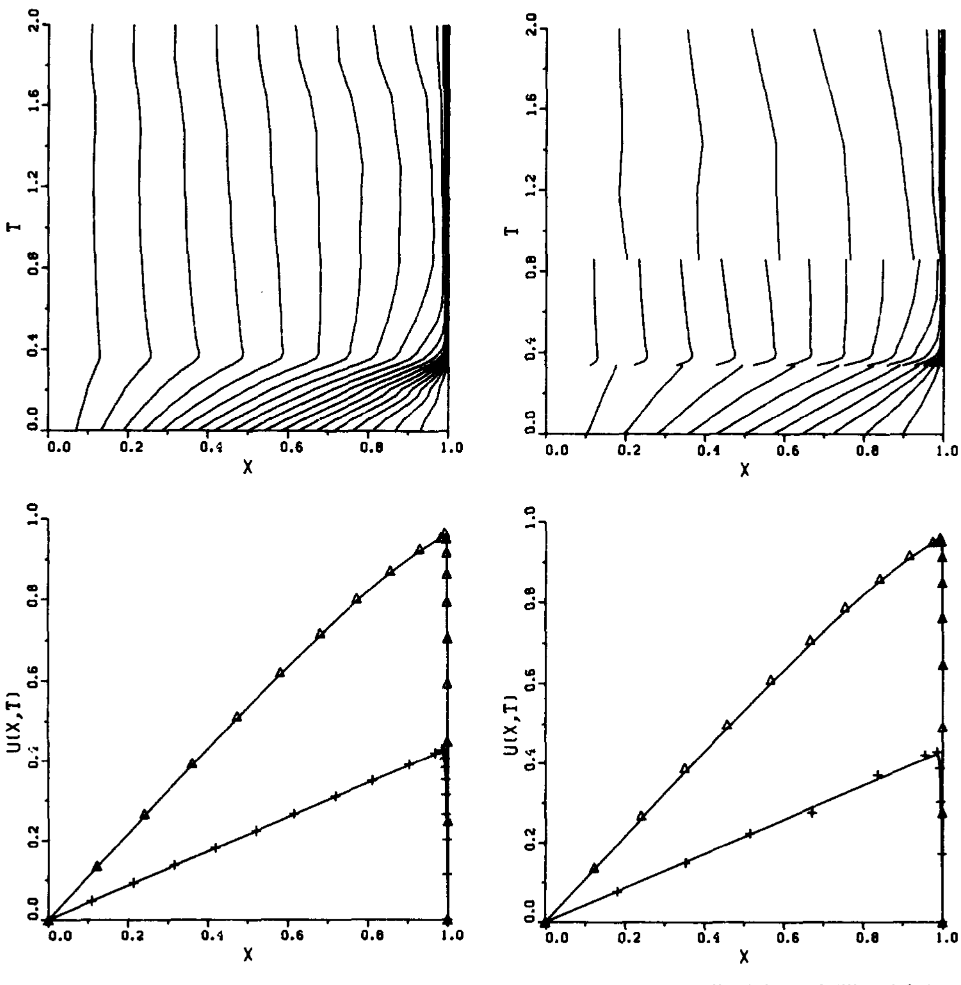


FIG. 5.1. Problem I. Grid trajectory and solution at $t = 0.6$, 2.0. Cases (ii) (left) and (iii) (right).

information in terms of NODES, STEPS, etc. Observe that in all four runs the number of time-steps required to reach $t = 2$ is small and that there are almost no step failures. The average number of Newton iterations required for these stepsize sequences amounts to approximately three, which is somewhat high. This is due to the rather large stepsizes and the value of 10^{-4} for *TOLN*, which, since we use the maximum norm, is certainly too stringent for the present application. Needless to say, a lower *TOLN* value yields less iterations.

5.3. *Problem II: Scalar Combustion Model*

This is a more challenging problem to solve numerically. It is described in Adjerid and Flaherty [2] as a model of a single step reaction with diffusion and reads

$$\partial u/\partial t = \partial^2 u/\partial x^2 + D(1 + a - u)\exp(-d/u), \qquad 0 < x < 1, \qquad 0 < t,$$
$$\partial u/\partial x(0, t) = 0, \qquad u(1, t) = 1, \qquad 0 < t,$$
$$u(x, 0) = 1, \qquad 0 \leqslant x \leqslant 1,$$

where $D = \mathrm{R}e^{d}/(ad)$ and R, d, a are constant numbers. The solution represents a temperature (of a reacting 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 rapidly increase to $1 + a$. A front then forms and propagates towards $x = 1$ with a very large speed (proportional to d). In real problems, a is close to 1 and d is large. The degree of difficulty of the problem is very much determined by the value of d. Following Adjerid and Flaherty [2], we have selected the problem parameters $a = 1$, $d = 20$, $R = 5$. The problem reaches a steady state once the flame gets to $x = 1$. For the current choice of problem parameters, this happens at time $t \approx 0.29$, which we take as the end point of our integrations.

Hence two phases can be distinguished in the solution, the formation of the front (the ignition phase) and its propagation to the right end point $x = 1$ (the propagation phase). For our numerical method, and presumably for most other moving grid methods, the ignition phase is the most challenging of the two. A first difficulty is that the formation of the front takes place very rapidly, originating widely different time and space scales. Consequently, variable time-steps are a necessity and very early in the ignition many of the points must move quickly to the left (the starting grid is uniform) to be in time to accurately resolve the front. A second difficulty relates to the smoothing capabilities of the implied change of variables $x = x(s, T)$. It was pointed out in Section 2.1 that, in the best possible case, the transformation should soften the problem both in space (by concentrating the nodes where appropriate) and in time (by achieving Lagrangian time derivatives smaller than the corresponding Eulerian time derivatives). However, for the problem at hand, these two objectives cannot be simultaneously reached. Assuming a uniform grid at the initial line (a choice suggested by the constant initial solution $u(x, 0) = 1$), and if the first objective is to be attained, the derivatives $\partial x/\partial T$ of a great deal of the trajectories should be negative in order to obtain the required refinement in the region of the front. However, during the formation of the front, $\partial u/\partial x < 0$ and $\partial u/\partial t > 0$. It follows immediately that then $\partial u/\partial T > \partial u/\partial t$, violating the second objective. Most Lagrangian type methods known to us, including ours, try to attain 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 in the present combustion problem, these methods offer no 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 attains 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 $u(x, t) = w(x - ct)$ becomes, in this respect, a trivial solution, provided that the grid trajectories satisfy $\partial x/\partial T = c$.

Interestingly, the Lagrangian approach followed by Petzold [20] meets the second objective. Her transformation is basically aimed at finding those trajectories along which the time rate of solution change is minimized, that is, $\partial u/\partial T \ll \partial u/\partial t$. However, during the formation of the front in the present combustion example, this then must imply that in the front region, $\partial x/\partial T > 0$, which means that points are moved away from the front, and thus the first objective is then violated. Petzold's algorithm has a built-in regridding step which corrects this deficiency. In her method, like in ours, both smoothing objectives are attained once the front is formed and starts to propagate.

At four selected times, adopted from Fig. 9 in Adjerid and Flaherty [2], Fig. 5.2 depicts two typical numerical Crank–Nicolson approximations. These correspond to the following fixed m cases: (i) $m = 20$, $TOLT = 0.005$; and (ii) $m = 40$, $TOLT = 0.001$. For both cases the initial stepsize is 10^{-2} and $TOLN = 10^{-4}$. Using 40 space nodes and 181 time-steps (see Table II), the Crank–Nicolson scheme yields a very good approximation, nearly up to plotting accuracy. The second derivative equidistribution is nicely shown in the location of the space nodes. Also note the rapid variation in the solution from $t = 0.26$ to $t = 0.27$, illustrating the need for variable time-steps. The widely different scales in the problem are also clearly

TABLE II

Problem II. Integration Information

	NODES							
Cases	Max	Min	Aver	STEPS	BS	JACS	ETF	NTF
(i)			20	116	835	236	1	9
(ii)			40	181	993	365	3	0
(iii)	16	10	14	112	846	232	1	11

shown in the grid trajectories. The solution obtained with 20 nodes and 116 time-steps is too crude, although its qualitative behaviour is in good accordance with the exact one. Note that 20 points suffice to represent the steady state solution, but are not enough to accurately follow the flame front during the ignition and propagation phase (a reduction of the stepsize in time yields no real improvement with $m = 20$).

Adjerid and Flaherty [2] show three approximations, two of them in their Fig. 9 and the third one in their Fig. 11. Our case (i) solution resembles their octagon solution of Fig. 9, which was also computed with 20 space nodes. None of their solutions is as accurate as our case (ii) solution. The differences are significant, especially during the ignition phase. Recall that we claim that our solid line

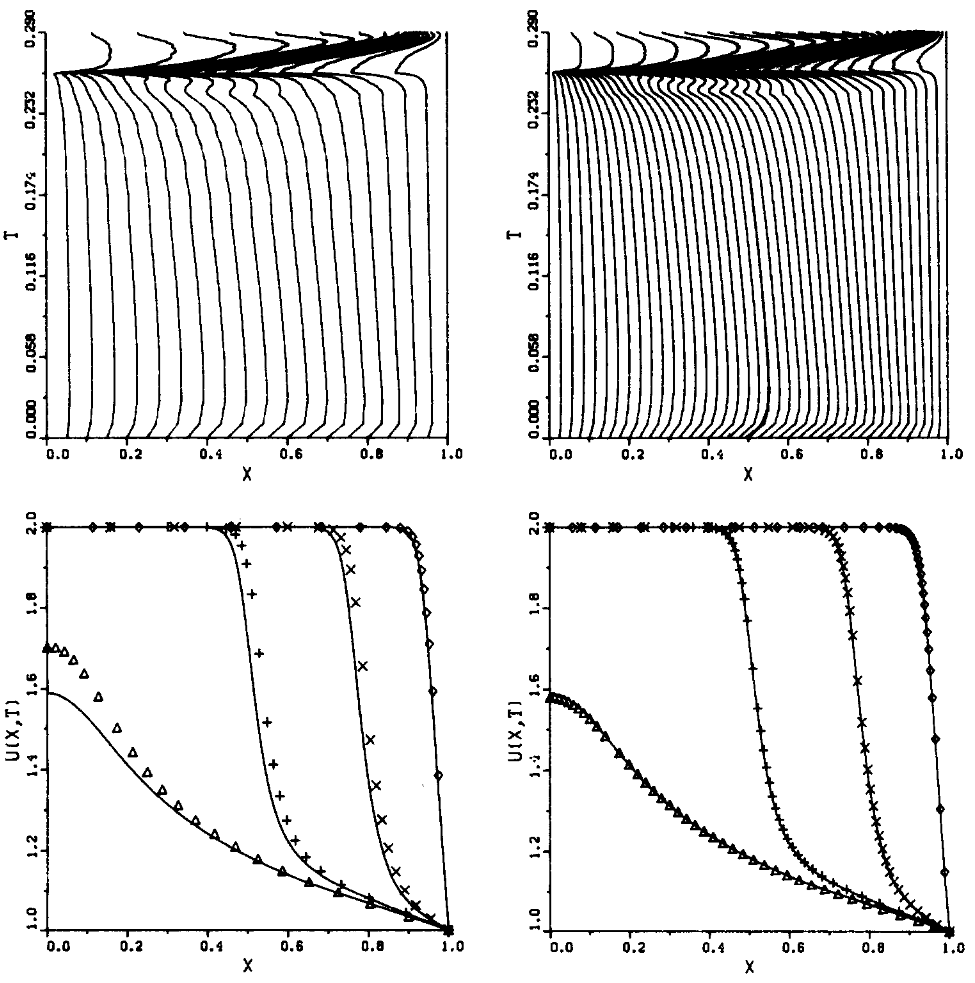


FIG. 5.2. Problem II. Grid trajectory and solutions at $t = 0.26$, 0.27, 0.28, 0.29. Cases (i) (left) and (ii) (right).

represents the exact solution, up to plotting accuracy. This "exact" solution was computed with our moving grid Crank–Nicolson scheme using (constant) $m = 150$, $TOLT = 10^{-4}$, $TOLN = 10^{-6}$. As an extra check, we have also solved the problem on a conventional, uniform, nonmoving mesh containing 2000 points, using the BDF code DASSL of Petzold [20] (with time tolerance equal to 10^{-8}) to integrate in time. The plots of this conventional method of lines solution are in full agreement with our reference solution, except for a slight difference at $t = 0.26$ in the neighbourhood of $x = 0$. Our ($m = 150$) moving solution yields $u(0, 0.26) \approx 1.59$, whereas the uniform ($m = 2000$) solution yields $u(0, 0.26) \approx 1.61$.

For the present combustion problem we are not yet satisfied with the performance of the currently implemented variable m option. When the ignition takes place the method still integrates with the number of nodes used initially. The consequence is that for a short time early in the ignition phase, too few points are used, which damages the accuracy (cf. the remark at the end of Section 3.2.2). Decreasing $TOLS$ would remedy this, but then the possibility exists that the method uses an unnecessarily large amount of nodes later in the computation. An alternative is to prescribe a suitable minimum for m. We found that satisfactory minima come very close to reasonable values in the fixed m mode. As an illustration of the previous remarks, we have included in Table II the results of one typical test, viz., case (iii): $TOLT = 0.005$, $TOLS = 0.05$ with a minimum of 10 points. Concerning accuracy, the computed solution very much resembles that of case (i) dealing with a fixed number of 20 points. We recall that decreasing $TOLS$ does not help much to improve the accuracy in the initial phase, unless the allowed minimum for m is also increased. Incidentally, it is of interest to note that in the cases (i) and (iii) for which $TOLT$ is equal, the integration performance in terms of STEPS etc., is virtually identical. This indicates that the variable m strategy does not interfere significantly with the time integration.

Of course, it remains possible that a more subtle strategy improves the performance on this problem of the variable m algorithm. However, it is also possible that a better recipe for coping with this specific sort of difficulty is to let the monitor function depend on the time rate of change at the boundaries. This makes sense because very often the birth of layers takes place at the boundary (see also Problems III and IV). The need for such a modified monitor exists mainly in the case of Neumann boundary conditions, and much less when Dirichlet conditions are prescribed. In the latter case, the rapid temporal and spatial variation is enforced in an explicit way by the data of the problem, which, for control procedures, is much easier to deal with. This is clearly illustrated in the next subsection, which also considers a combustion problem.

5.4. *Problem III: The Dwyer–Sanders Flame Propagation Model*

This model, first proposed as a test example in [12], simulates several basic features of flame propagation. It has two components, a mass density u and a temperature v. The PDE system is given by

$$\partial u/\partial t = \partial^2 u/\partial x^2 - uf(v), \qquad 0 < x < 1, \quad 0 < t \leqslant 0.006,$$

$$\partial v/\partial t = \partial^2 v/\partial x^2 + uf(v), \qquad 0 < x < 1, \quad 0 < t \leqslant 0.006,$$

where $f(v) = 3.52 * 10^6 \exp(-4/v)$. The initial functions are $u(x, 0) = 1$, $v(x, 0) = 0.2$ $(0 \leqslant x \leqslant 1)$. The boundary conditions at the left boundary read $\partial u/\partial x(0, t) = \partial v/\partial x(0, t) = 0$ and at the right $\partial u/\partial x(1, t) = 0$ and $v(1, t) = 0.2 + t/0.0002$ $(t \leqslant 0.0002)$ and $v(1, t) = 1.2$ $(t \geqslant 0.0002)$.

The given function for v at the boundary $x = 1$ represents a heat source. When v reaches its maximum, a steep flame front is generated which propagates from right to left at a relatively high speed. For $t > 0.003$, the speed of propagation of the front is almost constant. At the final time $t = 0.006$, the front has come close to the left boundary. As in the previous scalar combustion model, the present problem is made difficult both by the different scales involved and by the fact that the co-ordinate transformation does not help with the time integration at the formation stage.

In Table III, we present results of four runs, all of them with a value of 10^{-5} for the first stepsize and $TOLN = 10^{-3}$: (i) $TOLT = 0.05$, $m = 20$; (ii) $TOLT = 0.01$, $m = 30$; (iii) $TOLT = TOLS = 0.05$, minimal $m = 10$; (iv) $TOLT = TOLS = 0.01$, minimal $m = 10$. Figure 5.3 depicts the evolution of the exact solution and of the numerical solutions (i) and (iv). Both numerical approximations agree very well with the exact solution, in particular that for (iv) where no visible difference exists. In case (i) a little overshoot can be seen at the rear end of the front and the numerical front is a little too slow. Also observe that in this case, as in (iii), the NTF value is rather large, indicating that the choice of $TOLT = 0.05$ is on the optimistic side. The approximations of cases (iii) and (ii) are very similar to those of (i) and (iv), respectively. In the two variable number nodes runs, the average value "aver" is close to "max," because very early in the ignition phase the algorithm increases the number of nodes, without any further adaptation. This is what should happen, because when the ignition phase ends, the solution has become a travelling front and no further spatial change takes place.

In contrast with the experience of the previous scalar problem, here the variable m option fully met our expectations, in spite of the uniform start grid and the rapid

TABLE III

Problem III. Integration Information

	NODES							
Cases	Max	Min	Aver	STEPS	BS	JACS	ETF	NTF
(i)			20	164	1453	351	0	23
(ii)			30	272	1418	550	2	4
(iii)	17	10	17	162	1301	338	0	14
(iv)	44	10	42	282	1480	573	2	5

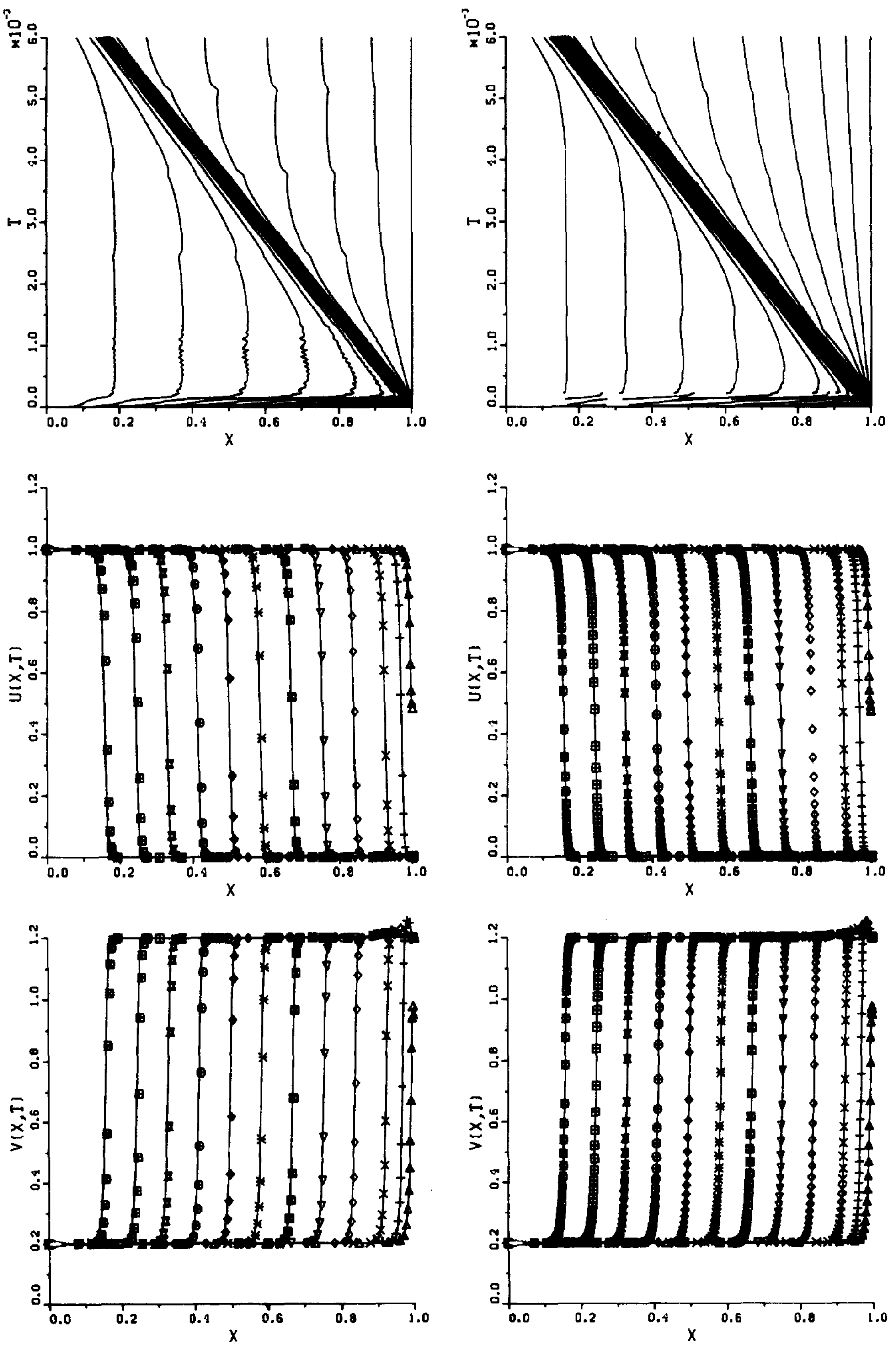


FIG. 5.3. Problem III. Grid trajectory and solutions at $t = 0.15E-3$, $0.3E-3$, $0.6E-3$ $(0.6E-3)0.6E-2$. Cases (i) (left) and (iv) (right).

development of the flame-front near the right boundary. We attribute this to the forcing temperature boundary function, which timely signals to the control mechanisms to move points sufficiently fast to the right boundary. It is evident that, in this connection, the value of the initial time-stepsize also plays a role. However, in the present context, the difference between Neumann and Dirichlet conditions may turn out to be essential. This will be illustrated again with Problem IV, where Neumann conditions are imposed and boundary layers are born, not only at the initial time, but also later in the evolution.

5.5. *Problem IV: A Problem from Mathematical Biology*

We consider the FitzHugh–Nagumo type equations

$$\partial u/\partial t = \partial^2 u/\partial x^2 + f(u) - v, \qquad f(u) = u(u-a)(1-u),$$
$$\partial v/\partial t = b(u - cv), \qquad 0 < x < 120, \quad 0 < t.$$

This system provides a conceptual model of ionic current flow across a semi-infinite nerve membrane; u is an electro-chemical potential and v a "recovery" variable. If $a = 0$ and $v = 0$, the equation for u becomes Fisher's equation. (Note that the equation for the variable v is an ODE.) The initial values for u, v are $u = v = 0$, while for u the following boundary conditions are imposed for $t > 0$: $\partial u/\partial x(0, t) = -I/2$, $\partial u/\partial x(120, t) = 0$. Note that the Neumann condition at the left is not consistent with the prescribed initial function. The parameters possess the values: $a = 0.139$, $b = 0.008$, $c = 2.54$, $I = 0.45$. I represents a constant current applied at the left end of the nerve and b is the reciprocal of the time scale associated with the recovery of the nerve.

We have taken this example from Bieterman and Babuška [4]. Numerical studies in Mitchell and Manoranjan [19] and elsewhere indicate the sensitivity of the solution behaviour to changes in a, b, c, and I. Bieterman and Babuška report that with the present choice of parameters, travelling waves develop at the left boundary. More precisely, repetitive pulses in u and v are generated with a firing frequency of about 0.0077 and develop into travelling waves which move with an approximate speed of 0.4. The time interval [0,200] is sufficiently large to recover the travelling wave forms.

TABLE IV

Problem IV. Integration Information

	NODES							
Cases	Max	Min	Aver	STEPS	BS	JACS	ETF	NTF
(i)	35	35	35	187	1256	378	1	3
(ii)	89	35	60	404	2233	817	5	0

The present problem is certainly not an easy one. Because incoming waves keep arising, the computation naturally asks for variable m. As t becomes large, more points are required to accurately represent the solution. Figure 5.4 and Table IV show the results of two runs: (i) $TOLT = TOLS = 0.05$; (ii) $TOLT = TOLS = 0.01$; in both $TOLN = 10^{-4}$ and the initial time-step is 10^{-6}. To prevent stepsize control problems due to the inconsistency between initial and boundary condition, the control was switched off for the first two steps. The allowed minimal value for m was set to 35. We need this value (for the crude tolerances) to accurately detect the first incoming wave. It was determined by trial and error. In case (i) this turned out to be also the maximum value required by the control procedure, so that case (i) is in fact a constant m run. Inspection of Fig. 5.4 reveals that, while initially 35 points are certainly enough, this is not the case later. Specifically, between times $t = 120$ and $t = 160$, when the second wave starts to develop, the control procedure fails to add more points near the boundary, with the result that, from that time on, the accuracy slowly degrades. This is due to the fact that there are not enough points near $x = 0$ (see the grid trajectories plot) to measure the size of the new arising gradients with sufficient accuracy. This deficiency of the control should, of course, diminish with $TOLS$. Indeed, for $TOLT = TOLS = 0.01$, with the same minimum of 35 points, the number of points is increased (twice) in the time evolution and the computed solution is sufficiently accurate over the entire time interval, although at the end phase errors are noticeable. This, presumably, is inevitable for this type of large-time wave calculations. Clearly, a further reduction of $TOLT$ and $TOLS$ will improve again, but of course at the expense of more work. Finally, as already indicated in the two previous subsections, we believe that the Neumann condition partly prevents the control procedures from detecting timely and accurately the rapid birth of the new incoming waves. It is very likely that an appropriate boundary adapted monitor would lead to a significant improvement, but no attempts in that direction have been undertaken so far.

5.6. *Problem V: An Opposite Travelling Waves Problem*

Our final example problem serves to illustrate the possibility of using different moving grids for different PDE components. We consider the two-component system (Madsen [16])

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

for $t > 0$ and $-0.5 < x < 0.5$, subjected to homogeneous Dirichlet boundary conditions and with initial conditions

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

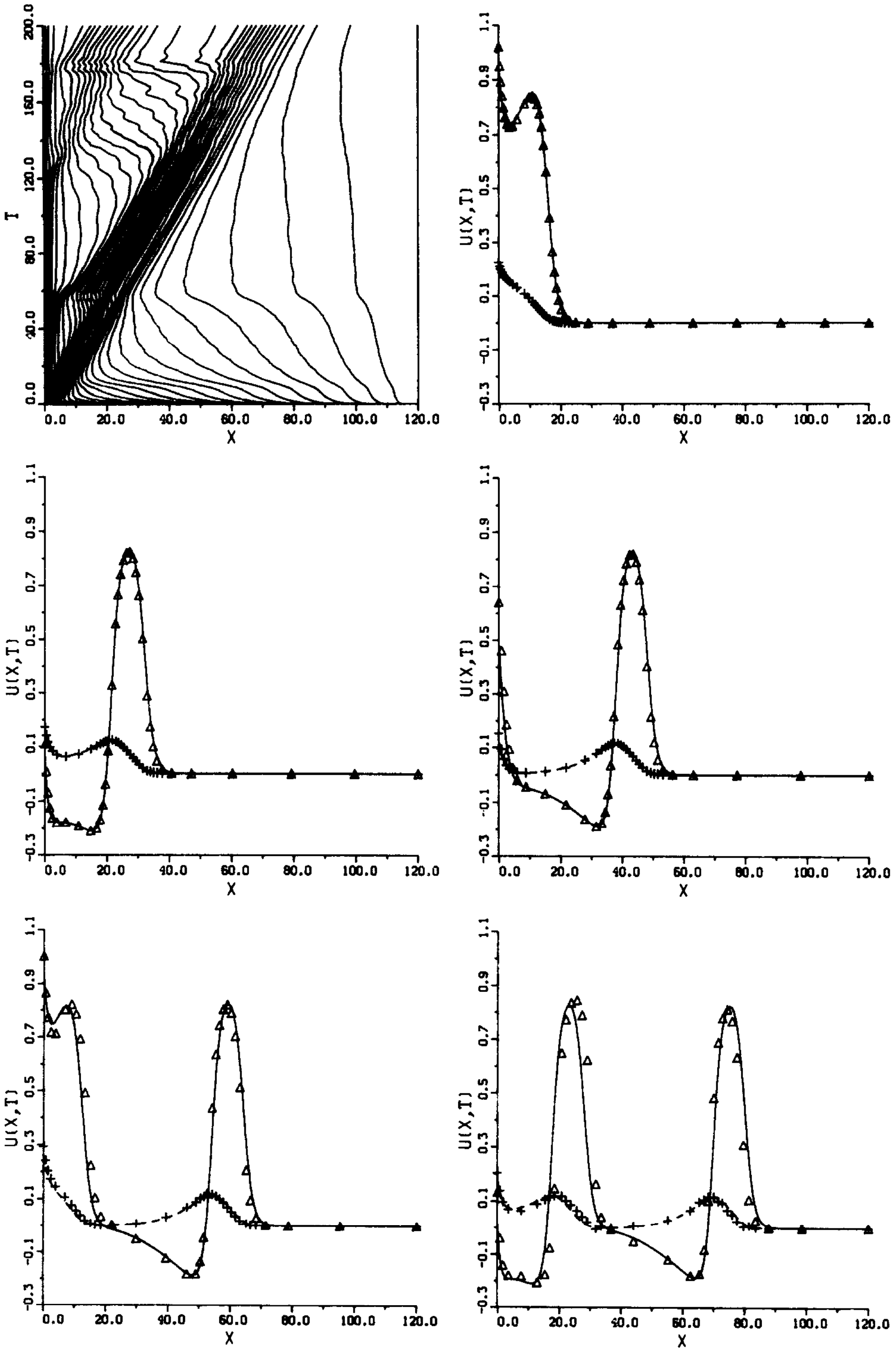


FIG. 5.4. Problem IV. Grid trajectory and solutions at $t = 40$, 80, 120, 160, 200. Case (i).

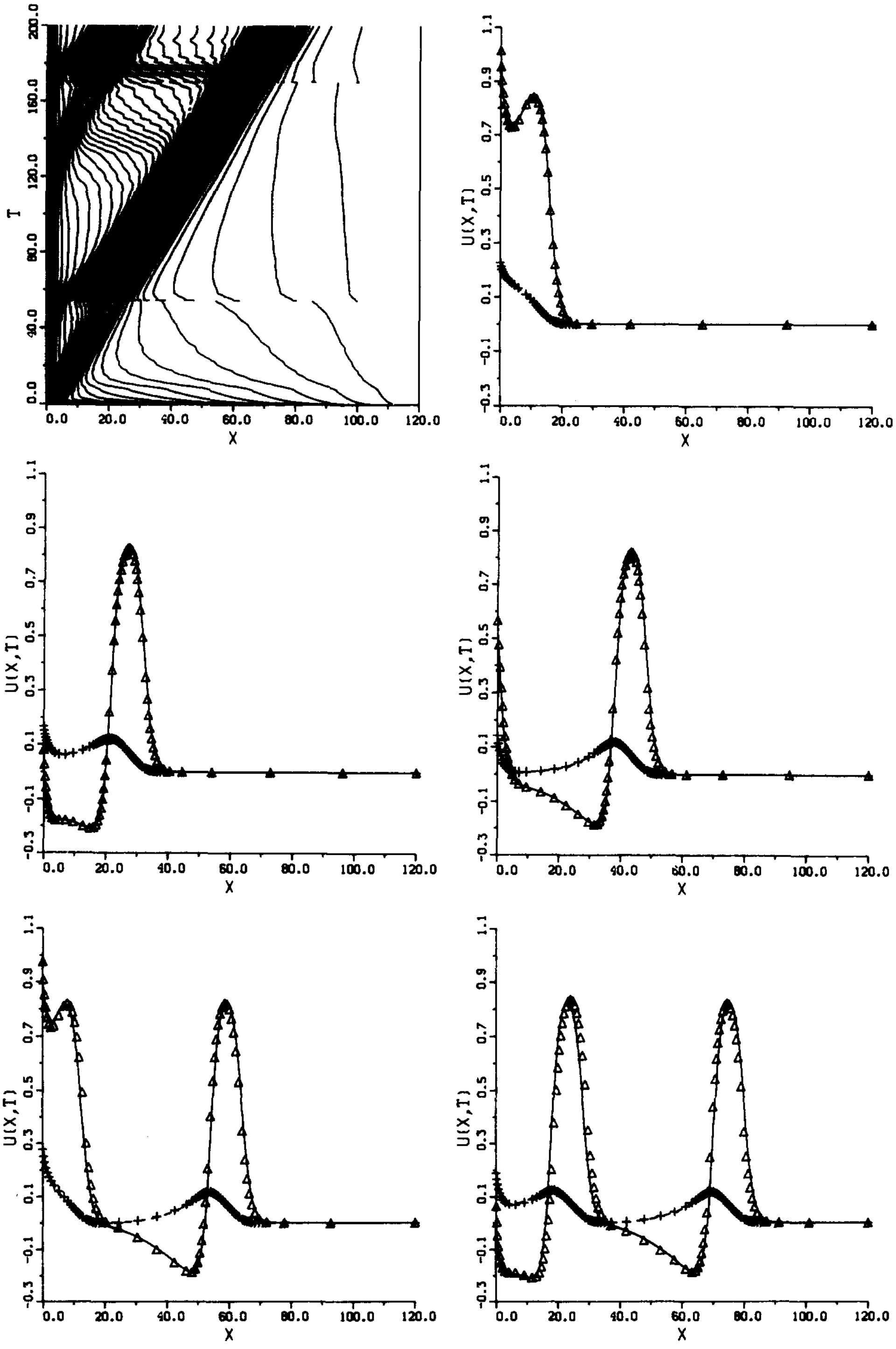


FIG. 5.4 (cont.) Problem IV. Grid trajectory and solutions at $t = 40$, 80, 120, 160, 200. Case (ii).

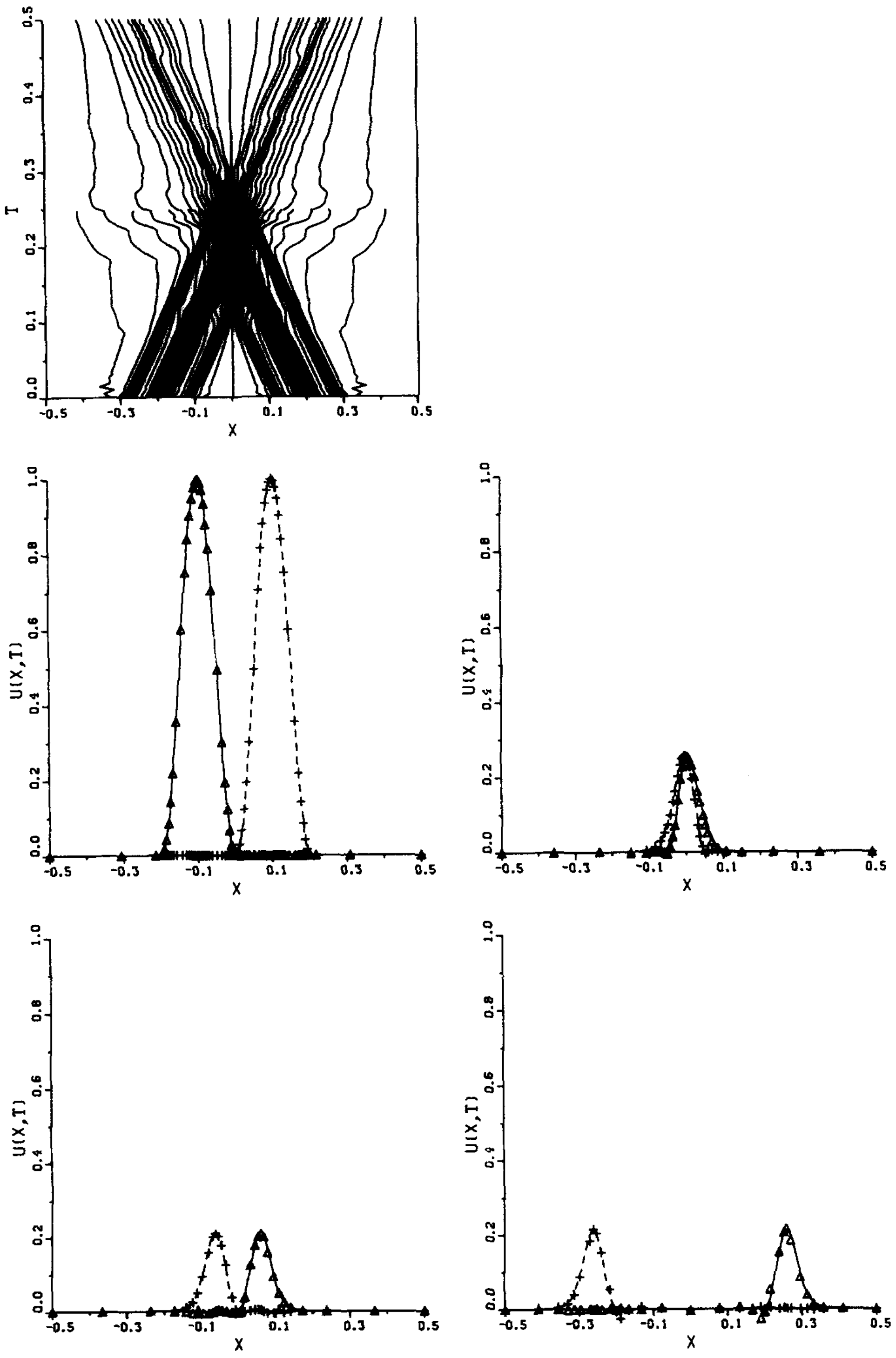


FIG. 5.5. Problem V. Grid trajectory and solutions at $t = 0.1$, 0.25, 0.3, 0.5. Case (iii).

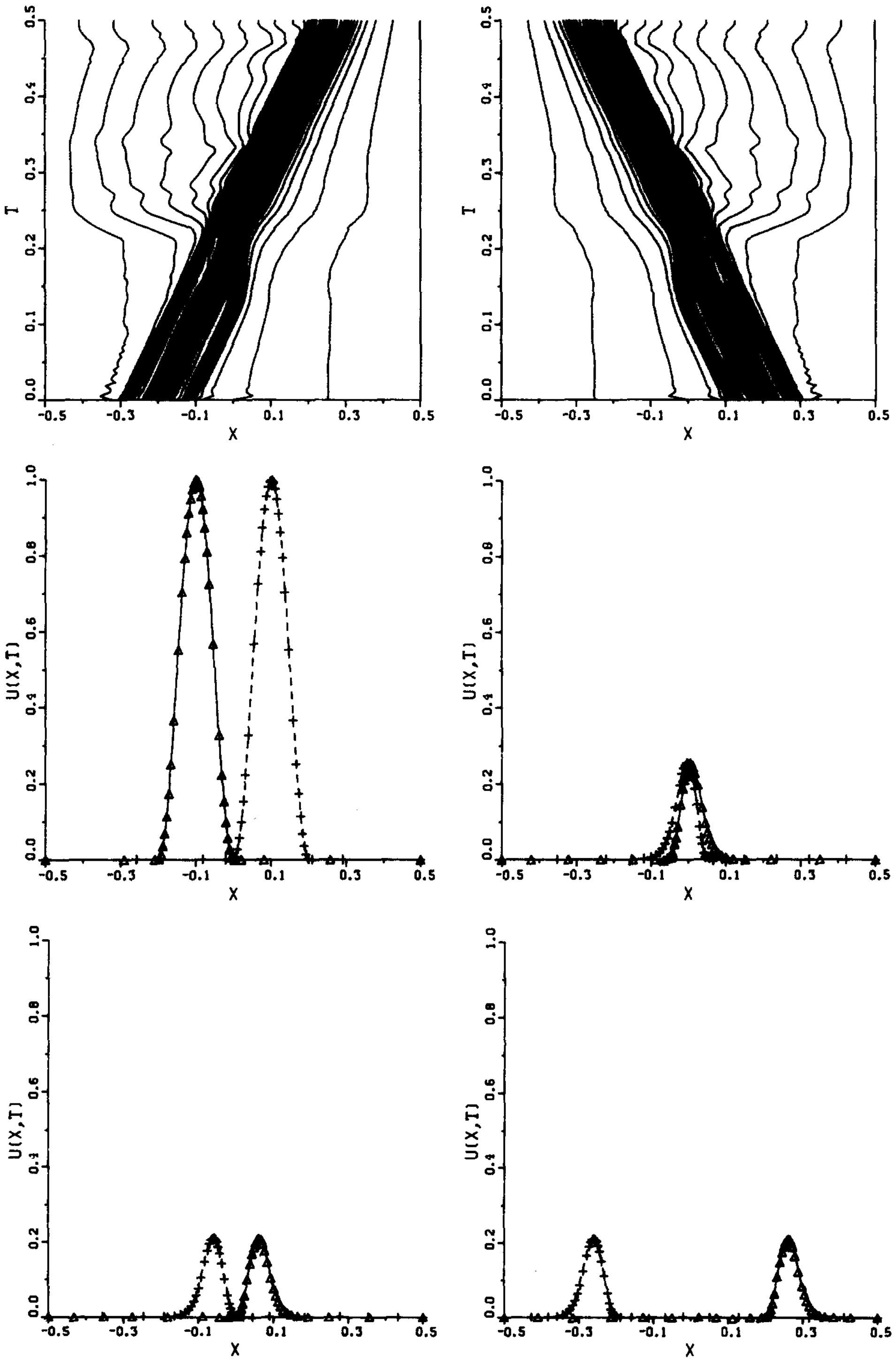


FIG. 5.5 (cont.) Problem V. Grid trajectories and solutions at $t = 0.1, 0.25, 0.3, 0.5$. Case (iv).

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 the speeds of the waves. Specifically, the crests of the waves collide a little beyond $t=0.25$ and they have separated again by $t=0.3$ approximately, 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.

We present results of four runs over the time interval $0 \leqslant t \leqslant 0.5$: (i) $TOLT=0.005$, $TOLS=0.05$, one grid; (ii) $TOLT=0.005$, $TOLS=0.05$, two grids; (iii) $TOLT=0.001$, $TOLS=0.01$, one grid; (iv) $TOLT=0.001$, $TOLS=0.01$, two grids. In all four experiments the initial time-step is 0.01, $TOLN=10^{-4}$, and the minimal value for m is 10. Figure 5.5 shows the computed approximations at $t=0.1$, 0.25, 0.3, and 0.5 for cases (iii) and (iv), of which (iv) is the most successful. The two numerical waves follow satisfactorily the exact solution. Some slight oscillations are observed at their tails. In the single grid case (iii), these oscillations are slightly more pronounced (and significantly more in the runs (i) and (ii) not depicted in the figure). However, in all four cases the numerical waves are in the right position. The oscillations disappear if more nodes and time-levels are used and originate from two error sources, viz., excessive numerical dispersion and lack of damping in the Crank–Nicolson scheme. The behaviour is similar to that encountered in standard fixed grid nondissipative Crank–Nicolson schemes. Also observe that if, locally, $\partial x/\partial T=1$ for the u-wave and -1 for the v-wave, then the Lagrangian form reduces to the stiff ODE system $du/dT=-100uv$, $dv/dT=-100uv$. It is well known that for such stiff problems the Crank–Nicolson scheme readily yields oscillations, if the time-step is too large and the solution not smooth. Further experiments have revealed that the backward Euler scheme obtained by setting $\theta=1$ in (2.2) produces significantly less oscillations. However, for most problems the choice $\theta=\frac{1}{2}$ is more accurate than any other.

TABLE V

Problem V. Integration Information

	NODES							
Cases	Max	Min	Aver	STEPS	BS	JACS	ETF	NTF
(i)	30	30	30	58	326	116	0	0
(ii)	20	20	20	56	309	112	0	0
	20	20	20					
(iii)	62	32	47	132	686	266	1	0
(iv)	39	39	39	131	685	264	1	0
	39	39	39					

In Table V we have listed the number of time-steps and grid-points used in the single and double grid runs. As expected, the number of time-steps in the two situations are almost equal, but, when using two grids, less points in space are needed. In this respect, the multiple grid option performs very satisfactorily for the present problem. It should be remarked, however, that the overhead involved in using more than one grid is large, so that the final gain in CPU time is less than that suggested by the reduction in grid-points. For the present example we observed only a small decrease in CPU time (about 10%), but recall that, as observed earlier, the two-grid runs are slightly more accurate.

6. Conclusions and Comments on Future Work

The numerical results presented in this paper are very encouraging. Obtained on a set of rather diverse sample problems, they clearly show that our fully adaptive moving grid algorithm, by using variable stepsizes in time and a variable number of moving space nodes, is capable of accurately tracking rapid spatial and temporal transitions.

The combination of a Lagrangian scheme like (2.2) with the "intermediate" approach, which generates the forward grid by fitting a predicted solution, has turned out to be successful. We generate the new grid through a conventional implicit Euler step followed by a de Boor type regridding. The choice of carrying out an extra implicit PDE calculation almost doubles the computational costs. Although this is clearly a drawback, the computation of the moving grid is an important task and cheap predictions should not be favoured if they deliver unsatisfactory grids. In our experience, the benefits to be gained by using the conventional implicit Euler prediction are robustness and reliability (see also Eiseman and Erlebacher [13]). Yet, for the near future, we will attempt to improve the efficiency of the algorithm by examining alternative predictions. A possibility consists of replacing the spatial equidistribution transformation by a transformation similar to that in Petzold [20] and the implicit Euler integration step by the solution of a tridiagonal system of linear algebraic equations. Details shall be reported elsewhere.

Our ultimate goal is the development of a user-oriented code, which can be applied almost as easily as existing, conventional MOL codes. In this connection note that while the literature on moving grid and adaptive methods has grown enormously in recent years, hardly any comparisons between different techniques have been published. No doubt, the availability of user-oriented codes would facilitate such comparisons. However, we wish to emphasize that the algorithm used to produce the results shown above is still very much in an experimental phase. In the near future, we plan to implement several possible improvements, mainly directed to the enhancement of the reliability of the procedure.

One such improvement would be to suggest a monitor function which signals effectively the rapid onset of steep layers near boundaries. As mentioned before, efficiently monitoring such solution phenomena is difficult, particularly in cases of Neumann boundary conditions.

A second area deserving attention is the de Boor regridding procedure. While this has worked satisfactorily in the test problems reported here (and note that no fine tuning of the grids was attempted), there is room for improvement. In fact, it is known [21, 22] that, when iterating the de Boor algorithm on a locally steep initial function to determine the starting grid, it is possible that the iterated grids do not converge, unless an unrealistic number of points is used. This fact not only is a clear drawback in itself, but also proves that it is possible to have very similar solution profiles that lead, via a de Boor loop, to clearly different meshes. For example, in the course of the experimental work, we once noticed that when approaching a steady state, the algorithm kept moving significantly the points in spite of the fact that the solution profile was changing very little. Furthermore, the gliding of the nodes from one time-level to the next implies corresponding changes in nodal values, which, on being detected by the step control mechanism, did prevent the time-steps from increasing. Such a gliding of the nodes may also appear in an application where excessively small stepsizes are being taken, because then the solution profile is almost unchanged from one time-level to the next. In passing, we note that in such a situation the assumption (3.4) is violated.

Finally, the application of a straightforward regridding algorithm like that used in this paper can result in grids with a fast change in grid spacing from one interval to the next. If the ratios become too large, both the stability and the accuracy of the approximation may deteriorate. The minimum and maximum ratios may be monitored by using so-called padded or filtered monitor functions (clearly explained by Furzeland [14]), or by smoothing the monitor function (see, e.g., formulas (4) and (5) in Dorfi and Drury [10]). Both approaches can be easily implemented to our regridding algorithm, since they essentially amount to an appropriate redefinition of the monitor function. These techniques enhance the quality of the generated grids and therefore the reliability of the complete algorithm. A disadvantage of this type of additional grid control is that a too strict bound on the grid ratios automatically requires more nodes than necessary for solving the small scale structures.

Acknowledgment

J. M. S. acknowledges the financial support from "Fondo Nacional para el Desarrollo de la Investigación Científico y Técnica," under Project PB-86-0313.

References

1. *NAG FORTRAN Library Manual, Mark 11* (N.A.G. Ltd., Oxford, 1983).
2. S. Adjerid and J. E. Flaherty, *SIAM J. Numer. Anal.* **23**, 778 (1986).
3. M. Berzins and R. M. Furzeland, Report No. 202, Department of Computer Studies, The University of Leeds, 1986 (unpublished).

4. M. Bieterman and I. Babuska, *J. Comput. Phys.* **63**, 33 (1986).
5. J. G. Blom, J. M. Sanz-Serna, and J. G. Verwer, Report NM-R8713, Centre for Mathematics and Computer Science, Amsterdam, 1987; *12th IMACS World Congress '88 on Scientific Computation, Paris, July 1988* (North-Holland, Amsterdam, in press).
6. J. G. Blom, J. M. Sanz-Serna and J. G. Verwer, *J. Comput. Phys.* **74**, 191 (1988).
7. R. Bonnerot and P. Jamet, *Int. J. Numer. Methods Eng.* **8**, 811 (1974).
8. C. de Boor, in *Conference on the Numerical Solution of Differential Equations, Dundee, Scotland, 1973*, edited by G. A. Watson, (Springer-Verlag, Berlin, 1974), p. 12.
9. S. F. Davis and J. E. Flaherty, *SIAM J. Sci. Stat. Comput.* **3**, 6 (1982).
10. E. A. Dorfi and L. O'C. Drury, *J. Comput. Phys.* **69**, 175 (1987).
11. I. S. Duff, AERE Report R. 8730, HMSO, London, 1977 (unpublished).
12. H. A. Dwyer and B. R. Sanders, Report SAND77-8275, Sandia National Laboratories, Livermore, 1978 (unpublished).
13. P. R. Eiseman and G. Erlebacher, ICASE Report 87-57, NASA Langley Research Center, Hampton, VA, 1987 (unpublished).
14. R. M. Furzeland, Report TNER.85.022, Thornton Research Centre, Shell Research Limited, 1985 (unpublished).
15. A. C. Hindmarsh, in *Advances in Computer Methods for Partial Differential Equations-IV*, edited by R. Vichnevetsky and R. S. Stepleman (IMACS, New Brunswick, NJ, 1981), p. 312.
16. N. K. Madsen, in *PDE Software: Modules, Interfaces and Systems*, edited by B. Engquist and T. Smedsaas (North-Holland, Amsterdam, 1984), p. 207.
17. K. Miller, *SIAM J. Numer. Anal.* **18**, 1033 (1981).
18. K. Miller and R. N. Miller, *SIAM J. Numer. Anal.* **18**, 1019 (1981).
19. A. R. Mitchell and V. S. Manoranjan, in *The Mathematics of Finite Elements and Applications IV*, edited by J. R. Whiteman (Academic Press, New York, 1982), p. 17.
20. L. R. Petzold, *Appl. Numer. Math.* **3**, 347 (1987).
21. J. D. Pryce, On the Convergence of Iterated Remeshing, Report, School of Mathematics, University of Bristol, 1986 (unpublished).
22. M. A. Revilla, *Int. J. Numer. Methods Eng.* **23**, 2263 (1986).
23. J. M. Sanz-Serna and I. Christie, *J. Comput. Phys.* **67**, 348 (1986).
24. M. D. Smooke and M. L. Koszykowski, *SIAM J. Sci. Stat. Comput.* **7**, 301 (1986).
25. J. G. Verwer and J. M. Sanz-Serna, *Computing* **33**, 297 (1984).
26. A. B. White, *SIAM J. Numer. Anal.* **19**, 683 (1982).