A MOVING MESH METHOD FOR ONE-DIMENSIONAL HYPERBOLIC CONSERVATION LAWS*

JOHN M. STOCKIE†, JOHN A. MACKENZIE‡, AND ROBERT D. RUSSELL†

Abstract. We develop an adaptive method for solving one-dimensional systems of hyperbolic conservation laws, that employs a high resolution Godunov-type scheme for the physical equations, in conjunction with a moving mesh PDE governing the motion of the spatial grid points. Many other moving mesh methods developed to solve hyperbolic problems use a fully implicit discretization for the coupled solution-mesh equations, and so suffer from a significant degree of numerical stiffness. We employ a semi-implicit approach that couples the moving mesh equation to an efficient, explicit solver for the physical PDE, with the resulting scheme behaving in practice as a two-step predictor-corrector method. In comparison with computations on a fixed, uniform mesh, our method exhibits more accurate resolution of discontinuities for a similar level of computational work.

Key words. moving mesh, adaptivity, equidistribution, shock capturing, hyperbolic conservation laws, finite volume methods.

AMS subject classifications. 65M06, 65M50, 76L05, 35L65, 35L67.

1. Introduction.

In this paper, we present an adaptive algorithm for computing solutions to one-dimensional systems of hyperbolic conservation laws. We consider problems of the form

$$\boldsymbol{q}_t + \boldsymbol{f}(\boldsymbol{q})_x = 0, \tag{1.1a}$$

$$\boldsymbol{q}(x,0) = \boldsymbol{q}_0(x), \tag{1.1b}$$

where $\boldsymbol{q}(x,t) \in I\!R^m$ is an m-vector, $a \le x \le b$, and $t \ge 0$. The system is "hyperbolic" in the sense that the Jacobian matrix $\partial \boldsymbol{f}/\partial \boldsymbol{q}$ has real eigenvalues and is diagonalizable with m linearly independent eigenvectors. Such problems are characterized by moving discontinuities (*i.e.*, fronts or shocks) that separate regions of flow where the solution is smooth. The major challenge in solving hyperbolic systems numerically is to capture the discontinuous solutions with sufficient accuracy while also keeping the computational cost within acceptable limits.

The vast majority of numerical methods for solving hyperbolic problems have been developed for fixed, uniform grids. An important class of such methods is based on the Godunov scheme [16], which is a first order, finite volume method that employs the exact solution to local Riemann problems at cell interfaces to enhance the resolution of discontinuities. Accuracy can be further improved by using higher order variants, such as the flux– and slope–limiter methods reviewed in [28, 31], and computational cost is reduced by linearizing the equations and solving approximate Riemann problems in each cell instead.

The discontinuities that are characteristic of hyperbolic problems typically move in time, and so the solution at a particular point in space can change very rapidly (see Fig. 1.1). As a result, computations on a fixed, uniform spatial mesh can require that

*This work was supported by fellowships from the Pacific Institute for the Mathematical Sciences and the MITACS National Centre of Excellence, and a research grant from NSERC.

†Department of Mathematics and Statistics, Simon Fraser University, Burnaby, B.C., Canada, V5A 1S6 (jms@cs.sfu.ca, rdr@cs.sfu.ca).

‡Department of Mathematics, University of Strathclyde, Glasgow, U.K., G1 1XH (j.a.mackenzie@strath.ac.uk).

Submitted to *SIAM Journal on Scientific Computing*, 1999.

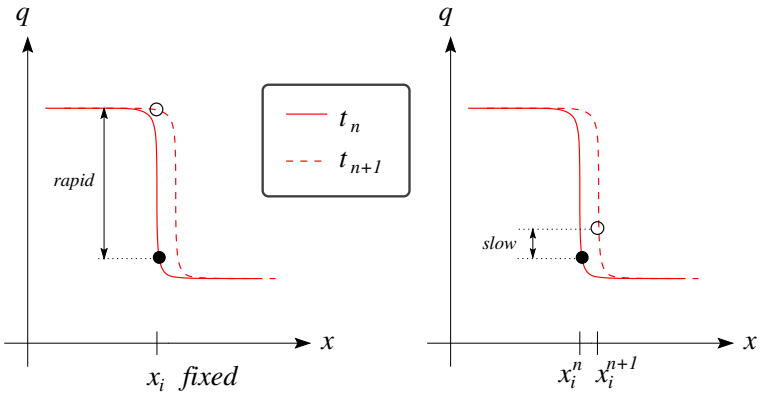


FIG. 1.1. *A comparison of the variation in solution from one time step to the next on fixed and moving meshes.*

the time step be extremely small in order to reduce the error arising from the time variation in the solution at points near the front. On the other hand, it is possible to take a much larger time step when the solution is smooth and does not exhibit such large variation from one mesh point to the next. In principle therefore, it is desirable to employ a non-uniform mesh that is sparse in regions where the solution is smooth and more concentrated near discontinuities. Since the steep fronts are not stationary, it is attractive to allow the mesh points to move in time so that fine grid resolution can be maintained near discontinuities, thereby attaining a balance between accuracy and efficiency.

There has been some success in solving hyperbolic problems on adaptive spatial meshes, starting with Harten and Hyman [17] who demonstrated how to extend a Godunov scheme to handle moving grids in one dimension. They employed a static regridding technique, in which the solution and mesh are evolved separately, with the mesh points updated in each time step to explicitly track discontinuities. Another static refinement strategy that has proven very successful, especially for higher dimensional problems, is the adaptive mesh refinement method [3] in which the mesh is refined locally based on some measure of the solution error using Cartesian sub-grids. Biswas *et al.* [4] combined both mesh movement and local mesh refinement in a finite element framework.

In contrast with static refinement techniques is the second major class of adaptive methods based on dynamic refinement, in which one explicitly derives an equation governing the spatial mesh, that moves mesh points naturally to where they are most needed. The mesh equation is often derived from an *equidistribution principle*, which attempts to equally distribute some measure of solution error over the spatial domain. This approach was used to solve one-dimensional hyperbolic problems in [30] and [11], employing a flux splitting method for discretizing the physical PDE. The major disadvantage to this technique is that the coupling between the solution and mesh equations is non-linear, often requiring a Newton iteration in each time step, which can be very costly. This problem is further exacerbated by the dense clustering of mesh points near discontinuities, which degrades the convergence of the iteration.

In many moving mesh methods it has been found necessary to introduce an additional artificial viscosity term of the form $\mu \boldsymbol{q}_{xx}$ into (1.1), and solve the resulting parabolic system instead. This approach has been taken in connection with spatial

discretisations based on finite differences [23, 29, 12], finite elements [15, 22], and collocation [19]. However, even with the viscous regularization, there are still convergence problems associated with taking the hyperbolic limit $\mu \to 0$ that can seriously degrade performance [15].

In this work, we propose an adaptive method that combines the flexibility and accuracy afforded by a dynamically moving mesh with the increased shock resolution capability of a Godunov-type scheme. The resulting method has the potential to reduce the cost of the moving mesh component of the algorithm significantly by eliminating the need for such a high concentration of grid points near discontinuities. We employ the moving mesh PDE or MMPDE approach of Huang *et al.* [18] in which the number of grid points is constant and the points move throughout the domain, subject to a time-dependent PDE. The main difference from many of the other moving grid methods mentioned earlier is that the MMPDE incorporates a temporal smoothing term which improves the performance of the solution–mesh iteration.

For the physical PDE, we use the wave-propagation method introduced by LeVeque [21] and implemented in the software package CLAWPACK [20]. The resulting method is similar to that of Chen [9], wherein a Godunov scheme was coupled with an MMPDE in a fully implicit time discretization based on the method of lines. However, we construct a more efficient semi-implicit scheme that behaves in practice as an explicit, predictor–corrector step. Furthermore, our method satisfies a discrete conservation principle which may not be the case for implicit discretizations based on a straightforward method-of-lines approach (see [14], [9] and [12]). Our strategy is based on the principle that there is no reason to solve the physical PDE and mesh equation with the same spatial discretization or to the same level of accuracy, since they are equations of different type (one hyperbolic, one parabolic) and have fundamentally different interpretations (one physical, and the other an artificial construct).

We begin in Section 2 with a description of the numerical method, including details of the wave propagation scheme for the hyperbolic conservation law, the discretization of the moving mesh equation, and the iteration that couples the two together. Section 3 presents several possible monitor functions that are designed specifically to resolve discontinuous solutions and shows how a monitor function can be generalized for systems of conservation laws where multiple fronts arise. In Section 4, numerical tests are performed on several problems encompassing both scalar conservation laws and systems, to demonstrate the accuracy, efficiency and robustness of the moving mesh method. Throughout two major issues of concern are choosing a monitor function that is tailored to capturing discontinuous flow features, and temporal smoothing for controlling mesh motion.

2. The Numerical Method.

2.1. The Physical PDE: Godunov's Method on a Moving Mesh. We first describe the finite volume discretization of the hyperbolic conservation law (1.1a) on a non-uniform, moving mesh derived by Fazio and LeVeque in [13][1]. Consider a sequence of times t^n for $n = 0, 1, \cdots$, where the time steps $\Delta t^n = t^{n+1} - t^n$ need not be equal. Suppose that the spatial domain $[a, b]$ is subdivided using a set of points which move in time and are parameterized by $x_i(t) = x(\xi_i, t)$ where $\xi_i = \frac{i}{N}$ are the

[1] Fazio & LeVeque used a simple moving mesh strategy for the Euler equations in which the contact line is tracked explicitly, and a piecewise uniform mesh is fit to either side of the discontinuity. Our moving mesh approach is designed to capture discontinuities naturally as part of the solution process, and is therefore much more flexible.

equally-spaced computational coordinates, and

$$a \equiv x_0 < x_1(t) < x_2(t) < \cdots < x_{N-1}(t) < x_N \equiv b.$$

Fig. 2.1 depicts a typical computational cell with grid points evolving from time level t^n to $t^{n+1} = t^n + \Delta t^n$, and with $x_i^n = x_i(t^n)$. In the following discussion, we develop

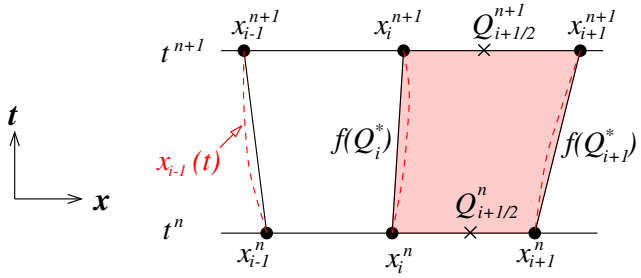


FIG. 2.1. *A computational cell on an adaptive spatial mesh. The dashed lines show the actual, curved, grid trajectories, which are approximated by straight lines in the xt–plane.*

the method in terms of the scalar quantity q to simplify notation, although the results extend easily to vectors.

In the *Godunov scheme* [16], the solution is assumed to be piecewise constant on each sub-interval $[x_i, x_{i+1}]$, and the discrete solution is taken to represent the average value of the actual solution along the lower cell boundary,

$$Q_{i+1/2}^n = \frac{1}{\Delta x_{i+1/2}^n} \int_{x_i^n}^{x_{i+1}^n} q(s, t^n)\, ds, \tag{2.1}$$

where $\Delta x_{i+1/2}^n = x_{i+1}^n - x_i^n$ is the local mesh spacing. We then integrate the conservation law (1.1a) across over the computational cell to obtain the formula

$$\kappa_{i+1/2}^{n+1} Q_{i+1/2}^{n+1} = \kappa_{i+1/2}^n Q_{i+1/2}^n - \frac{\Delta t^n}{\Delta \xi}\Big[(f(Q_{i+1}^*) - \dot{x}_{i+1}^n Q_{i+1}^*) - (f(Q_i^*) - \dot{x}_i^n Q_i^*)\Big], \tag{2.2}$$

where $\Delta\xi = \frac{1}{N}$ and $\kappa_{i+1/2}^n \doteq \Delta x_{i+1/2}^n / \Delta\xi$. The numerical flux $f(Q_i^*)$ is an approximation of the actual flux along the left slanted boundary of the cell and is computed using the solution to the local Riemann problem arising at the interface between the constant states $Q_{i-1/2}^n$ and $Q_{i+1/2}^n$. The intermediate state, Q_i^*, is actually the solution to the Riemann problem that lies along the straight-line characteristic $(x - x_i^n)/(t - t^n) = \dot{x}_i^n$. Here, we have assumed that the time derivative $\dot{x}_i = \partial x(\xi_i, t)/\partial t$ is constant over the time interval $[t^n, t^{n+1}]$, so that the edges of the cell are straight lines. The main difference between this and Godunov's scheme on a fixed mesh is the appearance of additional $\dot{x}$ terms in (2.2) that arise due to the movement of the mesh points.

The method in the form (2.2) is non-conservative but can be modified to get a conservative discretization. Again assuming $\dot{x}_i$ is constant in a cell, we can use $x_i^{n+1} = x_i^n + \Delta t^n\, \dot{x}_i^n$ to obtain

$$\kappa_{i+1/2}^n Q_{i+1/2}^n = \kappa_{i+1/2}^{n+1} Q_{i+1/2}^n - \frac{\Delta t^n}{\Delta \xi}(\dot{x}_{i+1}^n - \dot{x}_i^n) Q_{i+1/2}^n.$$

Substituting this expression into (2.2) yields the discrete system

$$\kappa^{n+1}_{i+1/2}Q^{n+1}_{i+1/2} = \kappa^{n+1}_{i+1/2}Q^{n}_{i+1/2} - \frac{\Delta t^n}{\Delta\xi}\Big[(f(Q^*_{i+1}) - \dot{x}^n_{i+1}(Q^*_{i+1} - Q^n_{i+1/2})) - (f(Q^*_i) - \dot{x}^n_i(Q^n_{i+1/2} - Q^*_i))\Big]. \tag{2.3}$$

The advantage to this form is that the quantity $\sum_i \kappa^n_{i+1/2}Q^n_{i+1/2}$ is conserved with n and constant states Q are preserved.

For reasons which will become clear shortly, we write the Godunov scheme in *wave propagation form*:

$$Q^{n+1}_{i+1/2} = Q^n_{i+1/2} - \frac{\Delta t^n}{\kappa^{n+1}_{i+1/2}\Delta\xi}\left(\mathcal{A}^+\Delta Q^n_i + \mathcal{A}^-\Delta Q^n_{i+1}\right), \tag{2.4}$$

with

$$\mathcal{A}^+\Delta Q^n_i = f(Q^n_{i+1/2}) - f(Q^*_i) - \dot{x}^n_i(Q^n_{i+1/2} - Q^*_i), \qquad \text{and}$$
$$\mathcal{A}^-\Delta Q^n_i = f(Q^*_i) - f(Q^n_{i-1/2}) - \dot{x}^n_i(Q^*_i - Q^n_{i-1/2}).$$

Here, $\mathcal{A}^+\Delta Q^n_i$ is the right-going flux difference from solving the Riemann problem between $Q^n_{i-1/2}$ and $Q^n_{i+1/2}$ and models the combined effect on the cell average $Q^n_{i+1/2}$ of waves entering from the left edge. Similarly, $\mathcal{A}^-\Delta Q^n_{i+1}$ is the left-going flux difference from the Riemann problem between $Q^n_{i+1/2}$ and $Q^n_{i+3/2}$ and models the combined effect of all waves entering the cell from the right.

The Godunov scheme can be very expensive, particularly for systems of conservation laws, where the Riemann problem at each cell interface requires the solution of a nonlinear system of equations. In practice, it is possible to solve the Riemann problem approximately based on the linearized system

$$\boldsymbol{q}_t + A\cdot\boldsymbol{q}_x = 0, \tag{2.5}$$

with A an $m\times m$ matrix. The well-known approximate Riemann solver of Roe [25] is the solution to such a linearization and yields a set of m wave speeds λ^p_i and jumps $\mathcal{W}^p_i$ across each wave, for $p = 1,\ldots,m$. Using the wave decomposition from the Roe solver, we can write

$$\mathcal{A}^+\Delta Q_i = \sum_{p=1}^{m}(\widetilde{\lambda}^p_i)^+\mathcal{W}^p_i, \tag{2.6a}$$

$$\mathcal{A}^-\Delta Q_i = \sum_{p=1}^{m}(\widetilde{\lambda}^p_i)^-\mathcal{W}^p_i, \tag{2.6b}$$

where $(\widetilde{\lambda}^p_i)^+ = \max(0,\widetilde{\lambda}^p_i)$, $(\widetilde{\lambda}^p_i)^- = \min(0,\widetilde{\lambda}^p_i)$, and $\widetilde{\lambda}^p_i = \lambda^p_i - \dot{x}^n_i$ are "shifted" wave speeds, incorporating the influence of the moving mesh. One of the major advantages of using this type of finite volume formulation is the natural way in which mesh motion is incorporated into the discrete equations, appearing only in the wave speeds. This should be contrasted with other methods based on finite differences that explicitly introduce terms of the form $q_x\dot{x}$, which can lead to problems with stability unless discretized carefully [23].

Godunov's method is only first order accurate in space, and so it suffers from a high degree of artificial dissipation which leads to significant smearing of the sharp fronts as the solution is evolved. To increase the accuracy of the discretization, one can use a high resolution flux correction (see [13, 21] for details).

Because the scheme is explicit in time, stability requires the time step to satisfy the *CFL condition* $\nu \leq 1$, where the CFL number is

$$\nu = \Delta t \max_{i,p} \left\{ \left| \frac{(\widetilde{\lambda}_i^p)^+}{\Delta x_{i+1/2}} \right| , \left| \frac{(\widetilde{\lambda}_{i+1}^p)^-}{\Delta x_{i+1/2}} \right| \right\} . \tag{2.7}$$

This inequality requires that the time step be small enough that waves from neighbouring cells do not intersect.

It is important to distinguish the difference between the CFL condition for a moving mesh method and that arising from a *fixed* non-uniform grid, which is the same as (2.7) except that $\widetilde{\lambda}_i^p$ is replaced by the unshifted wave speeds λ_i^p. On a fixed mesh, $\Delta x_{i+1/2}$ can be very small, leading to a very strict global requirement on the time step. In contrast, the moving mesh method allows for a much less restrictive CFL condition when the shifted wave speeds are close to zero; *i.e.*, provided that mesh points move at approximately the same speed as solution discontinuities: $\dot{x}_i \approx \lambda_i^p$. The possibility of relaxing the global CFL restriction for explicit calculations on non-uniform meshes is one of the major advantages of using a moving mesh for hyperbolic problems.

2.2. The Moving Mesh PDE. We consider now the mesh computation and for this formulate a moving mesh PDE that is based on an equidistribution principle. Following [18], we require that the transformation $x(\xi, t)$ from physical to computational coordinates satisfy

$$\int_0^{x(\xi,t)} M(s,t)\, ds = \xi \cdot \int_0^1 M(s,t)\, ds. \tag{2.8}$$

The function $M(x,t) > 0$ is called the *monitor function* and can be the most important component of the adaptive mesh algorithm. In principle, M can be any appropriately-chosen measure of the numerical error in the solution of the physical PDE. However, an advantage of the moving mesh approach is that M can be chosen to capitalize on some underlying property of the solution (for example, self-similarity or scaling invariance [5]). This is a feature that can be exploited in order to place mesh points precisely where they are needed in order to achieve optimal accuracy.

It is possible to discretize the integral form (2.8) directly, leading to a set of algebraic equations for the mesh locations which can be extremely difficult to solve efficiently when coupled with the physical PDE. Instead we take the moving mesh PDE approach used by Huang *et al.* [18], wherein a partial differential equation describing the moving mesh points is obtained by taking the time derivative of (2.8) and introducing temporal smoothing. There are a large number of possible moving mesh PDEs, one being MMPDE4 of [18]:[2]

$$\frac{\partial}{\partial \xi}\left(M \frac{\partial \dot{x}}{\partial \xi} \right) = -\frac{1}{\tau} \frac{\partial}{\partial \xi}\left(M \frac{\partial x}{\partial \xi} \right) . \tag{2.9}$$

[2] Another moving mesh PDE, MMPDE6, performs better for problems in which there are viscous shocks [22] or blow-up [5]. Its success is attributable to a certain scaling invariance property. However, we have found in numerical experiments that MMPDE6 is unsuitable for solving hyperbolic problems, which may be related to some alternate form of scaling invariance.

The parameter τ can be thought of as the time scale over which the mesh relaxes towards equidistribution.

A commonly used form of M is the arclength monitor function

$$M = \sqrt{1 + |q_x|^2}. \tag{2.10}$$

The corresponding centered finite difference approximation at cell midpoints is

$$M_{i+1/2} = \sqrt{1 + \left| \frac{\overline{Q}_{i+1} - \overline{Q}_i}{x_{i+1} - x_i} \right|^2}, \tag{2.11}$$

where $\overline{Q}_i = (Q_{i+1/2}\Delta x_{i-1/2} + Q_{i-1/2}\Delta x_{i+1/2})/(\Delta x_{i+1/2} + \Delta x_{i-1/2})$ is a weighted average of cell-centered solution values, located at cell edges. Notice that M is largest where the solution changes most rapidly, and as a result the mesh equation (2.9) serves to concentrate grid points in regions with large solution gradients.

It is well known that some sort of smoothing of the mesh is required in order to maintain reasonable accuracy in the computation of a solution on an adaptive mesh. Rather than smoothing the mesh itself, a commonly applied technique in the moving mesh framework is to replace the monitor function in the equations above by a regularized version $\widetilde{M}$ given by

$$\widetilde{M}_{i+1/2} = \sqrt{\sum_{k=i-i_p}^{i+i_p} M_{k+1/2}^2 \left(\frac{\gamma}{1+\gamma}\right)^{|k-i|} \Bigg/ \sum_{k=i-i_p}^{i+i_p} \left(\frac{\gamma}{1+\gamma}\right)^{|k-i|}}. \tag{2.12}$$

$\widetilde{M}_{i+1/2}$ can be thought of as a weighted average of the neighbouring $2i_p + 1$ values of M (with weighting factors determined by the parameter $\gamma > 0$) that serves to eliminate local oscillatory or non-smooth behaviour that may arise from solving the MMPDE.

We discretize the MMPDE using centered finite differences in space, yielding

$$M_{i+1/2}\left(\dot{x}_{i+1} - \dot{x}_i\right) - M_{i-1/2}\left(\dot{x}_i - \dot{x}_{i-1}\right) = -\frac{E_i}{\tau}, \tag{2.13a}$$

where E_i is a centered approximation to the term on the right hand side of (2.9) given by

$$E_i = M_{i+1/2}\left(x_{i+1} - x_i\right) - M_{i-1/2}\left(x_i - x_{i-1}\right). \tag{2.13b}$$

Since the monitor function depends on the solution values $Q_{i+1/2}$, the discrete physical equation (2.4) and (2.13a) form a coupled, nonlinear system of equations to be solved in each time step. In one-dimensional problems, it is common to employ a fully implicit time discretization and solve the resulting system of stiff ODEs using a package such as DASSL [24]. In two dimensions this procedure becomes very costly, and other approaches are needed (see [6], for example). We wish to construct a time-stepping procedure that preserves the conservation properties of the physical PDE, an issue addressed in the following section.

2.3. Time Integration. Here we describe an algorithm for solving the discrete equations (2.4) and (2.13). By itself, the wave propagation scheme is explicit, but

when coupled with the mesh equation, the system becomes implicit due to the dependence of the mesh equation on Q through the monitor function.

In our experience, a straightforward explicit discretization in time will lead to instabilities and therefore some form of implicit time differencing is required (see also [23, 15]). Intuitively, it is easy to see the need for some form of iteration: if we were to employ a fully explicit algorithm, basing the mesh locations on the solution for the previous time step, then grid points can lag behind the solution, leading to serious violations of the CFL condition. However, a fully implicit scheme, wherein the coupled nonlinear system is solved in each time step, is far too expensive for practical calculations. Our method is similar to the fully explicit, predictor–corrector step constructed in [29], except that the step is iterated. Our aim in constructing a scheme is to leave the solution step for the physical PDE unaltered, so that we can employ the conservative, high resolution algorithms in CLAWPACK, and build around it a fixed point iteration on the mesh.

We propose the following:

Moving Mesh Algorithm

1. *Let* $n = 0$.
2. *Given an initial solution* Q^0 *at time* $t = t^0$, *equidistribute the mesh exactly using a discretization of the exact equidistribution principle* $(Mx_\xi)_\xi = 0$ *(corresponding to* $\tau = 0$*).*
3. *Step the solution to time level* $n+1$ *using the following iteration:*
   (a) *Let* $m = 0$. *Take a guess at the new mesh positions using* $x_i^{n+1,0} = x_i^n + \Delta t^n \dot{x}_i^n$, *unless this leads to mesh crossings, in which case we take the conservative guess* $x_i^{n+1,0} = x_i^n$.
   (b) *Use* CLAWPACK *to obtain an initial guess* $Q^{n+1,0}$ *for the solution at time level* $n+1$ *using the mesh* $x_i^{n+1,0}$. CLAWPACK *returns an adaptively-chosen time step* Δt^n *which is used for the remainder of the iteration.*
   (c) *Iterate on* m*:*
      i. *Compute the raw and smoothed monitor function values,* $M_{i+1/2}$ *and* $\widetilde{M}_{i+1/2}$, *and the quantity* E_i *from (2.13b), all based on the current solution iterate* $Q_{i+1/2}^{n+1,m}$.
      ii. *Employ a Crank-Nicholson discretization of (2.13a) for updating the mesh:*

$$\begin{aligned}&\widetilde{M}_{i+1/2}^{n+1,m}(x_{i+1}^{n+1,m+1} - x_i^{n+1,m+1}) - \widetilde{M}_{i-1/2}^{n+1,m}(x_i^{n+1,m+1} - x_{i-1}^{n+1,m+1})\\ &= \widetilde{M}_{i+1/2}^{n+1,m}(x_{i+1}^n - x_i^n) - \widetilde{M}_{i-1/2}^{n+1,m}(x_i^n - x_{i-1}^n) - \frac{\Delta t^n}{2\tau}\left[E_i^{n+1,m} + E_i^n\right],\end{aligned}$$

      for which a tridiagonal system is solved for the unknowns $x_i^{n+1,m+1}$.
      iii. *Use* CLAWPACK *to obtain the new solution approximation* $Q_i^{n+1,m+1}$, *using mesh positions* $x_i^{n+1,m+1}$, *mesh velocities* $\dot{x}_i^{n+1} = (x_i^{n+1,m+1} - x_i^n)/\Delta t^n$, *and a CFL restriction of* $\nu \le 0.9$.
      iv. *If* $\|x^{n+1,m+1} - x^{n+1,m}\|_1 > TOL$, *then increment* m, *and go to Step 3ci.*
4. *Once the iteration has converged, increment* n *and begin a new time step in Step 3.*

Some of the key aspects of the algorithm worth noting are that:

- the physical PDE is integrated explicitly in time using a conservative method;
- the convergence tolerance is applied on the mesh only;
- there is only a single parameter (the temporal smoothing, τ) needed to control the mesh motion;
- because the physical PDE is solved using the general-purpose solver CLAWPACK, the method is easy to generalize to other problems.

These features combine together to yield a method that to our knowledge is distinct from any other adaptive approach that has appeared in the literature.

3. Monitor Functions for Resolving Discontinuities. The monitor function should in principle be chosen so that it represents some measure of the error in the computed solution in an appropriate norm. At points where the error is large, M should also be large so that mesh points will tend to concentrate in those areas where higher resolution is needed. While solution arclength (2.10) concentrates mesh points where the solution has large gradients and gives excellent results for many parabolic problems, have found that it is inappropriate for hyperbolic equations. Particularly when high resolution Godunov schemes are used, computed discontinuous flow features can be very steep, and so the moving mesh PDE may drive more points than necessary into the neighborhood of these discontinuities. Several investigations have already been made into how the arclength monitor function can be modified to avoid excessive clustering of points and so control the conditioning of the mesh equation [10, 1, 22]. Our objective in this section is to obtain regularized variants of the arclength monitor, which are still large where discontinuities arise, but not so large that they over-resolve steep layers.

One way to modify the arclength monitor function so as to balance the number of points inside and outside a steep internal layer is to introduce a "regularizing factor" α in the following manner:[3]

$$M = \sqrt{1 + \frac{1}{\alpha} |q_x|^2}. \tag{3.1}$$

The factor α allows one to reduce the magnitude of the monitor function in situations where $|q_x|$ is very large, thereby avoiding over-resolution of steep layers, while also ensuring that M still retains a significant peak near these discontinuities. Beckett & Mackenzie [1] applied this type of regularization in the context of singularly perturbed boundary value problems, and using a monitor function based on curvature: $M = 1 + \frac{1}{\alpha}|q_{xx}|^{1/m}$. Taking α to be any constant greater than one in (3.1) will tend to move points out of a steep front into regions where the solution is smooth. However, if we are to construct a monitor function that will give reasonably consistent results for shocks of arbitrary steepness, then α will have to depend on the solution.

A scaling of the arclength function using the maximum solution value (with $\alpha \propto \max_x |q|^2$) was applied in [11] to eliminate large variations in the solution components for systems of conservation laws. This monitor will still over-resolve discontinuous

[3] Note that the monitor functions $M = \sqrt{\alpha + |q_x|^2}$ and $M = \sqrt{1 + |q_x|^2/\alpha}$ give identical results because of the invariance of MMPDE4 in (2.9) under the scaling $M \mapsto cM$, where c is a constant. MMPDE6, incidentally, is not invariant under this scaling.

fronts, and so we consider instead a scaling based on the maximum derivative value,

$$\alpha = \alpha^{max} \doteq \max_x |q_x|^2/\beta, \tag{3.2}$$

where the parameter $\beta > 1$ controls the concentration of mesh points. A related regularization, used in [1], scaled the derivative term in (3.1) by its average value over the domain using

$$\alpha = \alpha^{avg} \doteq \frac{1}{|b-a|} \int_a^b |q_x|^2 \, dx. \tag{3.3}$$

The advantage to this type of monitor function is that there is no free parameter needed for mesh control.

3.1. Generalization to Systems of Equations. The discussion of monitor functions thus far has been restricted to the case where the unknown function is a scalar. The definition extends naturally to systems of equations in which $\boldsymbol{q}$ is an m-vector. Perhaps the most obvious approach is to replace the absolute value $|q_x|$ with the vector 2–norm $\|\boldsymbol{q}\|$ in (3.1). However, in problems such as the Euler equations of gas dynamics, the solution components can take on values that differ widely in magnitude, in which case it makes no sense to weight the individual components equally. Furthermore, shocks are characterized by jumps in all three components, whereas contact lines appear as discontinuities in the density only, as pictured in Fig. 3.1. As a result shocks are weighted more heavily in the calculation of the monitor function and contact lines are essentially ignored. This is undoubtedly what limits the accuracy of contact line resolution in other moving mesh computations such as those reported in [8, 22].

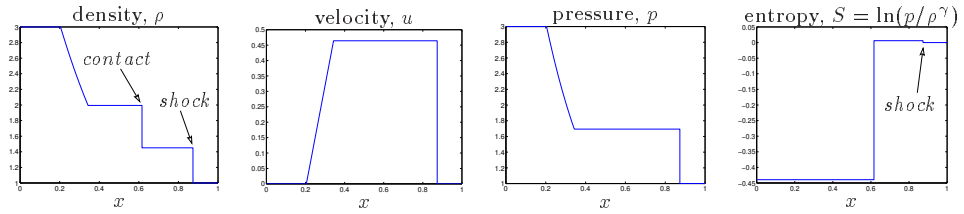


FIG. 3.1. *Plots of density, velocity, pressure and entropy for a solution to the Euler equations. In each plot, the rarefaction wave is on the left, the shock on the right, and the contact line lies in between.*

In order to obtain a monitor function that is better able to capture all discontinuous flow features, some form of rescaling must be performed on the solution components. Dorfi & Drury [11] used a weighted 2-norm of the solution in which each component $q^{(j)}$ is scaled by a factor $\alpha^{(j)} \approx \max_x |q^{(j)}|^2$, and as a result the contribution of each component to the monitor function is the same.

Rather than using a simple scaled vector norm, we construct two distinct monitor functions, one tailored to recognizing shocks and one to contact lines, and scale each separately. Since the velocity, u, is discontinuous only across shocks, a natural choice for constructing a "shock monitor" is the following:

$$M^s = \sqrt{1 + \beta \left(\frac{|u_x|}{\max_x |u_x|} \right)^2}, \tag{3.4}$$

although pressure could equally well be used in place of velocity. We choose to regularize the monitor function using α^{max}, since then the maximum magnitude of M is independent of the shock steepness or the magnitude of the velocity component. It is then possible to normalize the monitor functions so that different discontinuities are given equal weight, something which is not possible with the averaged regularization parameter α^{avg}.

In a similar manner, we construct a "contact monitor" function defined in terms of the entropy $S = \ln(p/\rho^\gamma)$ as

$$M^c = \sqrt{1 + \beta \left(\frac{|S_x|}{\max_x |S_x|} \right)^2}. \tag{3.5}$$

While the entropy is discontinuous across both shocks and contact lines (see Fig. 3.1) and so is not an ideal choice for a contact monitor, entropy typically has a much smaller jump across a shock than a contact line. Therefore, (3.5) should still be a good measure of the location of contact discontinuities.

The moving mesh algorithm requires a single monitor function, which we define using the simple convex combination

$$M^{cs} = \theta M^c + (1-\theta) M^s. \tag{3.6}$$

In all numerical simulations performed herein, we weight the two classes of discontinuity equally and take $\theta = \frac{1}{2}$. The resolution of the shock and contact line is fairly insensitive to the choice β in (3.4) and (3.5) provided it is taken large enough, and in practice a value of $\beta = 100$ has proven to give satisfactory results.

4. Numerical Results. In this section, we will focus on three test problems: the inviscid Burgers' equation; the Buckley-Leverett equation (which is distinguished by a non-convex flux function); and the Euler equations of gas dynamics (a system of 3 hyperbolic conservation laws). We evaluate the various monitor functions introduced in Section 3 by comparing accuracy and computational cost. We also investigate the importance of the temporal smoothing parameter τ appearing in the moving mesh equation, and whether there are any particular issues associated with solving problems that have discontinuous solutions. The solution quality and algorithm performance are much less sensitive to the level of spatial smoothing, and so we fix the parameters $i_p = 4$ and $\gamma = 2$ in (2.12) for the remainder.

Computational costs are reported as CPU times measured on a Sun SPARCstation–20, and in all of our simulations the overhead cost associated directly with computing the mesh amounted to at most 25% of the total CPU time. As a result, any differences in total cost between the fixed and moving mesh results can be attributed primarily to differences in the time step in conjunction with any additional moving mesh iterations required in each time step.

Errors are reported relative to an "exact" solution which is actually a finely-resolved numerical approximation obtained from a fixed mesh computation with $N = 2500$. An appropriate norm for measuring errors in discontinuous solutions is the L^1–norm, which we approximate using the formula

$$\|Q - q\|_1 = \sum_{i=1}^{N} \left| Q_{i+1/2} - q(x_{i+1/2}) \right| \cdot \Delta x_{i+1/2},$$

where Q is the computed solution and q the exact solution.

4.1. Inviscid Burgers' equation. Our first test of the moving mesh method is the inviscid Burgers' equation

$$q_t + \left(\frac{1}{2}q^2\right)_x = 0, \tag{4.1}$$

for $0 \leq x \leq 1$, and $t \geq 0$, with periodic boundary conditions and an initial solution profile given by $q_0(x) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x)$. The solution propagates to the right, steepening until the *singularity time* $t_s = 64/(129\pi) \approx 0.15792$, at which point a shock forms.

The solution up to time $t = 1.2$ is displayed in Fig. 4.1 for a moving mesh calculation with $N = 50$, $\tau = 0.05$ and regularization parameter α^{avg}, while the fixed grid solution with the same number of mesh points is given in Fig. 4.2 for comparison. It is evident that the moving mesh computation yields considerably

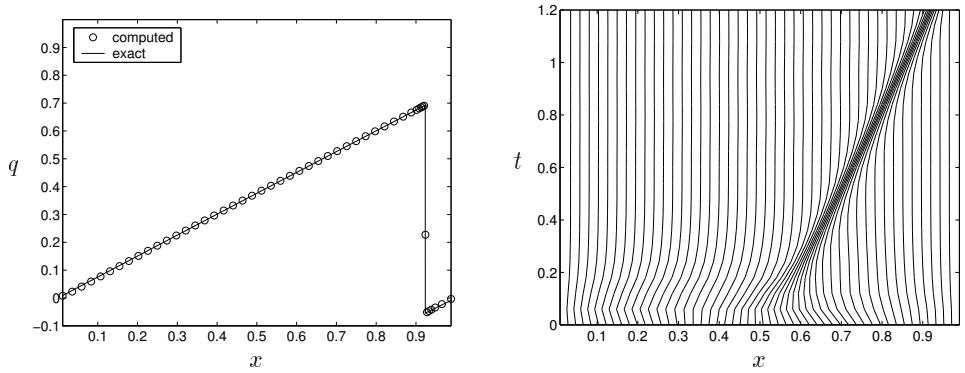


FIG. 4.1. *Moving mesh solution to Burgers' equation at time $t = 1.2$ sec. On the left is the solution and on the right are the mesh contours (α^{avg}, $\tau = 0.05$, $N = 50$).*

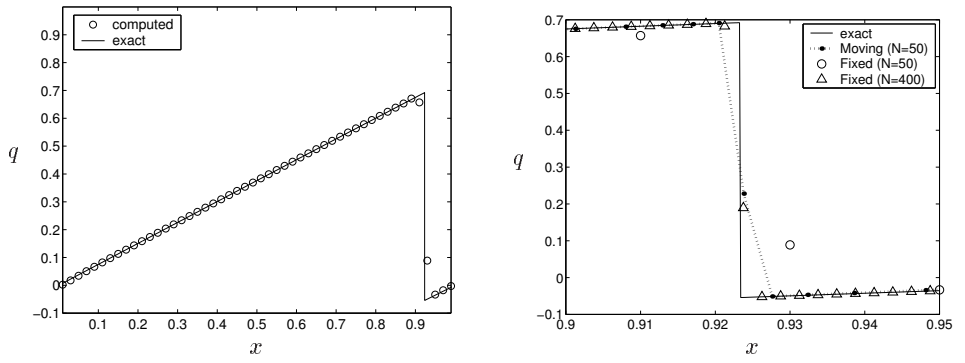


FIG. 4.2. *Fixed mesh solution with $N = 50$ points (left) and a blow-up of various fixed and moving mesh calculations in the neighborhood of the shock (right).*

better shock resolution than the fixed mesh method because of the clustering of mesh points near the shock. The ability of the mesh to capture and follow the moving shock is demonstrated by the mesh contour plot in Fig. 4.1.

TABLE 4.1
Comparison of various fixed and moving grid solutions for Burgers' equation. The arclength computation flagged with a "" experienced difficulties with stability of the mesh equation which accounts for the exceptionally large CPU time.*

Description	*CPU time*	*NT*	L^1 *error*	$\mathcal{R}$
Fixed mesh:				
$N = 50$	0.44	78	0.0042	
$N = 100$	0.92	149	0.0028	
$N = 200$	2.43	289	0.0015	
$N = 400$	7.36	562	0.0007	
Moving mesh ($\tau = 0.1$)*:*				
α^{max}, $N = 50$	1.66	185	0.0017	0.104
α^{avg}, $N = 50$	1.58	172	0.0013	0.101
α^{avg}, $N = 100$	4.84	331	0.0005	0.056
Arclength monitor ($N = 50$)*:*				
$\tau = 0.2$	2.67	323	0.0015	0.022
$*$ $\tau = 0.1$	31.20	4256	0.0016	0.001

A more quantitative comparison of solution errors is afforded by Table 4.1, which lists the L^1 error in the solution for various fixed and moving mesh calculations. The moving mesh calculations with 50 grid points are as accurate as for a fixed mesh with 200 points, and require less CPU time.

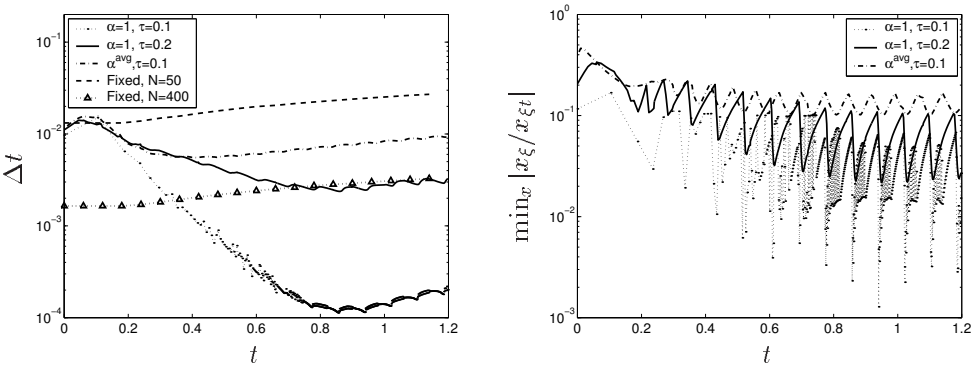
For the computations using regularization parameters α^{avg} and α^{max}, the number of iterations in each time step is at most 2. Therefore, the major factor determining the efficiency of the method is the CFL condition (2.7) arising from the physical PDE. The time steps in the moving mesh calculations are considerably larger than that which would be allowed on a fixed mesh with a similar level of refinement near the shock. This is illustrated by the time step plots in Fig. 4.3(a), which correspond to a 400-point fixed mesh and a moving grid with α^{avg}. However, the improvement is not as dramatic as we might have expected from (2.7) because the shifted wave speeds are never exactly zero. It is not possible to constrain mesh points to lie always along the front since they must periodically enter the shock layer from the right and leave from the left, as seen in Fig. 4.1. This phenomenon, known as *"mesh racing,"*[4] is unavoidable in r-adaptive methods for which mesh points are not explicitly added or removed throughout the adaptation procedure.

The connection between mesh racing and the CFL number (2.7) can be made more apparent by considering two mesh trajectories, x_{i-1} and x_i, that lie along a shock as pictured in Fig. 4.4. The mesh point x_i continues moving along the shock and so the wave speed is $\lambda \approx \dot{x}_i$. The local CFL number corresponding to the mesh point x_{i-1} is then given by

$$\nu_{i-1/2} \;=\; \frac{|\lambda - \dot{x}_{i-1}|\Delta t}{x_i - x_{i-1}} \;\approx\; \frac{|\dot{x}_i - \dot{x}_{i-1}|}{\Delta\xi} \cdot \frac{\Delta\xi}{x_i - x_{i-1}} \cdot \Delta t \;\approx\; \Delta t \Big/ \left|\frac{x_\xi}{x_{\xi t}}\right|. \tag{4.2}$$

Consequently, when mesh racing occurs, there is an accompanying stability constraint on the time step that is determined by the *"mesh racing factor,"* $\mathcal{R} \doteq \min_x |x_\xi / x_{\xi t}|$.

[4] While not standardized in the literature, the term "mesh racing" has been used in the past by K. Miller.



(a) Time step. *(b) Mesh racing factor.*

FIG. 4.3. *Plots of the time step (left) and mesh racing factor* $\mathcal{R} = \min_x |x_\xi / x_{\xi t}|$ *(right) for several fixed and moving mesh calculations. The moving mesh computations with arclength and* α^{avg} *monitor functions (with* $N = 50$*) demonstrate that smaller time steps correspond to smaller values of* $\mathcal{R}$*.*

Table 4.1 and Fig. 4.3(b) both show that smaller values of $\mathcal{R}$ (associated with clus-

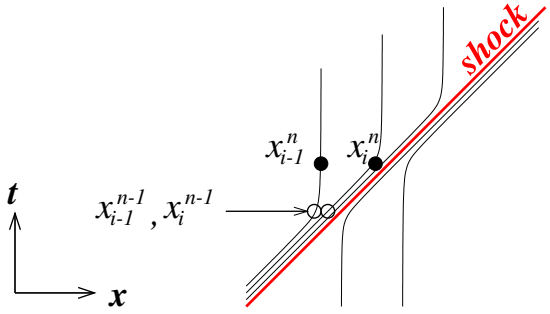


FIG. 4.4. *A plot of moving mesh trajectories in the* $x - t$ *plane, depicted in the neighborhood of a discontinuity, denoted by the thick grey line. Mesh racing is illustrated, in which two mesh trajectories lying along the shock,* x_{i-1} *and* x_i*, diverge from each other between one time level and the next.*

tering of grid points and large mesh curvature) correspond to reductions in the time step. More careful investigation shows that this correlation between time step and mesh racing factor persists for all moving mesh calculations.

Temporal smoothing plays a central role in the performance of the algorithm and the resolution of discontinuous solution features. The primary influence of the parameter τ is to smooth the mesh trajectories, or in other words to lessen the mesh "curvature", $x_{\xi t}$. Hence, temporal smoothing is integrally connected to mesh racing. Too much smoothing prevents the moving mesh algorithm from recognizing a discontinuity at all, resulting in a mesh that is nearly uniform. On the other hand, too little smoothing can lead to mesh racing, excessively small time steps, and difficulties with convergence of the solution-mesh iteration. Table 4.2 displays the CPU time and errors for several values of τ, from which it is evident that the algorithm can be quite sensitive to the level of smoothing.

TABLE 4.2
Comparison of various fixed and moving grid solutions for Burgers' equation (α^{avg}, $N = 50$). The arclength computation marked with a "" experienced difficulty with stability of the mesh equation.*

τ	*CPU time*	*NT*	L^1 *error*	$\mathcal{R}$
$*$ 0.01	63.38	324	0.0072	0.005
0.025	3.68	261	0.0016	0.050
0.05	1.96	236	0.0022	0.068
0.10	1.58	171	0.0013	0.101
0.20	1.25	117	0.0016	0.200
0.50	0.98	89	0.0035	0.769

The temporal smoothing parameter can also be interpreted as a time scale over which the grid relaxes towards equidistribution. Consequently, a secondary effect of temporal smoothing is a slight time delay in the mesh motion. This is manifested in Fig. 4.1 as a higher concentration of mesh points at the trailing edge of the shock.

A tangible connection can be made between mesh racing and the level of temporal smoothing, based on the analysis of moving mesh methods performed by Smith & Stuart [26] for a general class of time-dependent scalar PDEs. In this work, the authors derive the following bounds on the mesh derivatives in terms of τ and $\bar{M} \doteq \max_x(M)$:

$$|x_{\xi t}| \leq \frac{\bar{M}^2}{\tau} \qquad \text{and} \qquad x_\xi \geq \frac{1}{\bar{M}},$$

which can be combined to obtain a bound on the mesh racing factor,

$$\mathcal{R} \geq \frac{\tau}{\bar{M}^3}. \tag{4.3}$$

While this is only an lower bound on $\mathcal{R}$, it does suggest that our strategy of combining temporal smoothing with a regularized monitor function serves to control mesh racing. This is particularly evident in the case of the α^{max} regularization, for which M is bounded by a constant and in fact, $\mathcal{R} \geq \tau/\beta^{3/2}$. Referring to the contour plots in Fig. 4.5 and the entries in Table 4.2, τ clearly does have a mollifying influence on the mesh curvature and hence also on mesh racing.

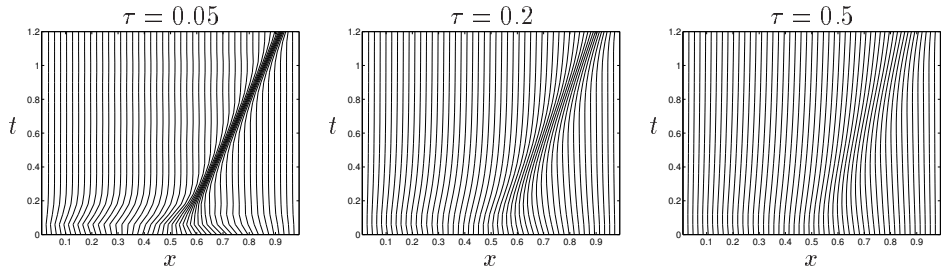


FIG. 4.5. *Mesh trajectories for various temporal smoothing parameters (α^{avg}, $N = 50$).*

We now move on to a discussion of the choice of monitor function. Using solution arclength for problems with steep fronts, the mesh equation is known to become ill-conditioned as the viscosity and hence also the front thickness go to zero [15], causing

mesh points to cluster in the layer. The results in Table 4.1 demonstrate that the arclength monitor requires a higher value of τ to control the mesh behaviour, and the iteration diverges even when τ is as large as 0.1. As long as the temporal smoothing is taken large enough the arclength results are comparable to that for other monitor functions, although the accuracy suffers somewhat because of the time lag in the mesh motion. However, as we will see later in computations with the Euler equations, arclength can introduce severe ill-conditioning in the mesh equation, so we do not advocate its use for hyperbolic problems.

We close with a brief comparison to other more common moving mesh approaches, and report on simulations for which the physical and mesh PDEs are discretized implicitly in time using a method of lines approach. The resulting nonlinear system is solved using the BDF solver DASSL [24] with error tolerances $atol = rtol = 10^{-6}$. We employ two different spatial discretizations: the first being a centered finite difference approximation of the viscous Burgers' equation with the viscosity $\mu \to 0$, and the second a Godunov scheme for the inviscid problem. The CPU times and errors are listed in Table 4.3, from which it is easy to see the significant increase in cost required for a fully implicit method. The close-up plots of the solution near the shock in Fig. 4.6 indicates the clustering together of points as the layer steepens, with CPU times demonstrating the corresponding increase in difficulty of solving the mesh equation.

TABLE 4.3

Comparison of moving grid calculations for Burgers' equation based on the method of lines ($N = 50$, $\tau = 0.05$). The columns NT and NJAC give the number of time steps and number of Jacobian evaluations respectively.

Description	*CPU time*	*NT*	*NJAC*	L^1 *error*
Inviscid Burgers, α^{max}	15.70	1264	221	0.0063
Viscous Burgers, $\alpha \equiv 1$, $\mu = 10^{-3}$	9.08	580	44	0.0041
$\mu = 10^{-4}$	14.70	830	79	0.0039
$\mu = 10^{-5}$	32.90	1523	214	0.0036

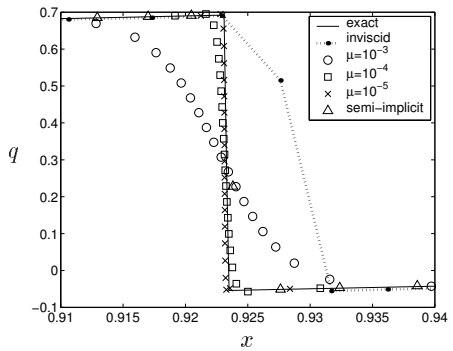


FIG. 4.6. *Close-up of the solution to the viscous and inviscid Burgers' equation at time $t = 1.2$ using a fully implicit method of lines approach. Our semi-implicit moving mesh solution is shown for comparison purposes.*

These difficulties can only be expected to worsen for hyperbolic problems which have no natural dissipative mechanism to control the mesh. In fact, the inviscid

Burgers' computation fails to converge at all using the arclength monitor function, and so an alternate monitor had to be employed. Using the α^{max} regularization we were able to compute the solution pictured in Fig. 4.6. While the L^1-error is only slightly larger in comparison to the other computations, there is clearly a significant error in the shock speed. This points to a serious drawback in using a straightforward method of lines approach for hyperbolic problems: namely, that the resulting discretization may not be conservative. This is one of our main motivations for constructing a semi-implicit algorithm using an explicit, conservative step in the physical solution.

4.2. Buckley-Leverett equation. We next consider the scalar Buckley-Leverett problem which has been used to describe the one-dimensional flow of two immiscible fluids in a porous medium, neglecting capillary pressure and gravity. This may be used to model gas and oil in a reservoir, for which the equations take the form of a scalar conservation law (1.1a), where q is the fluid saturation (water volume / pore volume) and the flux function is

$$f(q) = \frac{q^2}{q^2 + 0.5(1-q)^2}. \tag{4.4}$$

The solution is obtained on the interval $x \in [0, 1]$, with initial and boundary conditions

$$q(x,0) = \frac{1}{1+10x}, \qquad q(0,t) = 1, \qquad q(1,t) = \frac{1}{11},$$

corresponding to an "oil recovery" scenario where water is pumped into the reservoir at $x = 0$ and oil is forced out at $x = 1$. In contrast with Burgers' equation, the Buckley-Leverett flux function $f(q)$ is non-convex, which leads to difficulties with some numerical schemes and so serves as an excellent test of a numerical method.

The solution to time $t = 1.0$ *sec* is shown for a moving mesh in Fig. 4.7 and the corresponding fixed mesh simulation in Fig. 4.8. Again, the adaptive mesh does an excellent job of capturing and following the discontinuity, and resolves the shock layer much more accurately than the fixed mesh method with the same number of points.

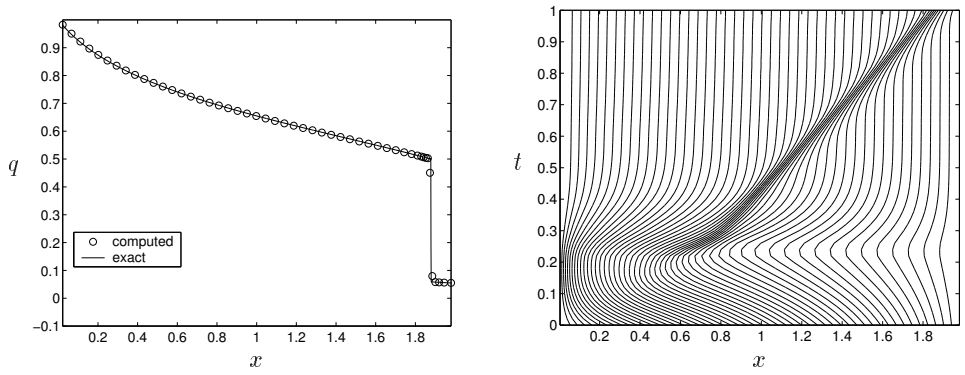


FIG. 4.7. *Moving mesh solution to the Buckley-Leverett equation* (α^{max}, $\tau = 0.05$, $N = 50$).

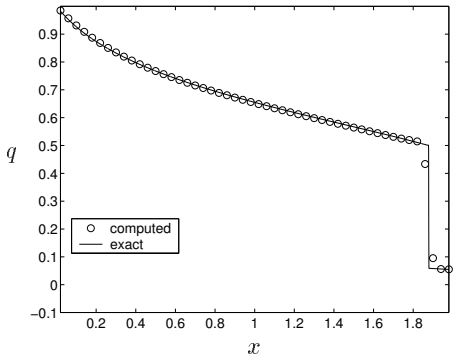


FIG. 4.8. *Fixed mesh solution to the Buckley-Leverett equation ($N = 50$).*

4.3. Euler equations. The Euler equations describing the evolution of an inviscid, compressible, polytropic gas in one dimension are

$$\frac{\partial}{\partial t}\begin{bmatrix} \rho \\ \rho u \\ e \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(e+p) \end{bmatrix} = 0 \tag{4.5}$$

where ρ is the density, u the velocity, p the pressure, $e = \frac{p}{\gamma - 1} + \frac{1}{2}\rho u^2$ the internal energy, and γ the ratio of specific heats ($\gamma = 1.4$ for air). The system (4.5) is in the form of a conservation law (1.1a) with $q = (\rho, \rho u, e)^T$ and $f(q) = qu + (0, p, pu)^T$.

We consider Sod's classical shock tube problem [27] which has the following initial conditions

$$q(x,0) = \begin{cases} (1.0, 0.0, 2.5), & \text{if } 0 \le x < 0.5 \\ (0.125, 0.0, 0.25), & \text{if } 0.5 \le x \le 1 \end{cases}$$

and reflecting boundary conditions at $x = 0$ and $x = 1$. In contrast with the Burgers' or Buckley–Leverett problems of the previous sections (in which the initial conditions are smooth), the selection of an appropriate initial mesh is of particular importance here because the initial conditions are discontinuous. In order for mesh points to be placed on or near the initial discontinuities, the data must be smoothed over some finite width. We therefore replace jumps in the initial data using hyperbolic tangent profiles with width $\epsilon \approx 0.005$, and then equidistribute the initial mesh exactly (*i.e.*, $\tau = 0$) with the smoothed data. The resulting mesh and solution data are used as the initial conditions for the numerical scheme.

The performance of the algorithm with the arclength monitor function was much worse than for Burgers' equation, requiring an excessively large value of τ for stability. We therefore consider only regularized monitor functions, and begin with $M = \sqrt{1 + \|\boldsymbol{q}_x\|_2^2/\alpha}$, based on a straightforward 2-norm of the solution components, with $\alpha = \int \|\boldsymbol{q}_x\|_2^2 \, dx$. The computed density and mesh contour plot are displayed in Fig. 4.9, from which we see that the moving mesh solution provides much better resolution of the shock and rarefaction wave than the fixed mesh algorithm (shown in Fig. 4.10). However, the contact discontinuity is not detected at all, and consequently the resolution of the contact line is no better than for a fixed mesh. These qualitative observations are supported by the L^1 error values reported in Table 4.4.

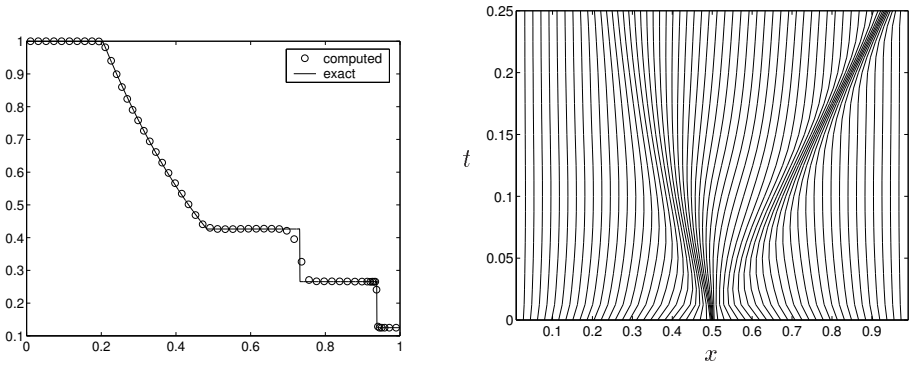


FIG. 4.9. *Moving mesh solution (left: density component; right: mesh contours) for the Euler equations, with monitor function based on the 2-norm (α^{avg}, $\tau = 0.005$, $N = 60$).*

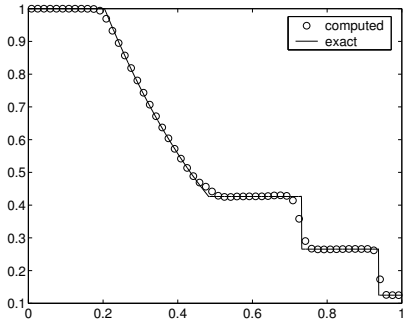


FIG. 4.10. *Fixed mesh solution for the Euler equations with $N = 60$.*

We next demonstrate the effectiveness of the monitor functions tailored specifically to resolving individual elementary waves. The mesh computed with the shock and contact monitor functions, M^s and M^c from (3.4) and (3.5), respectively, are pictured in Fig. 4.11. Clearly, both do an excellent job of capturing their corresponding

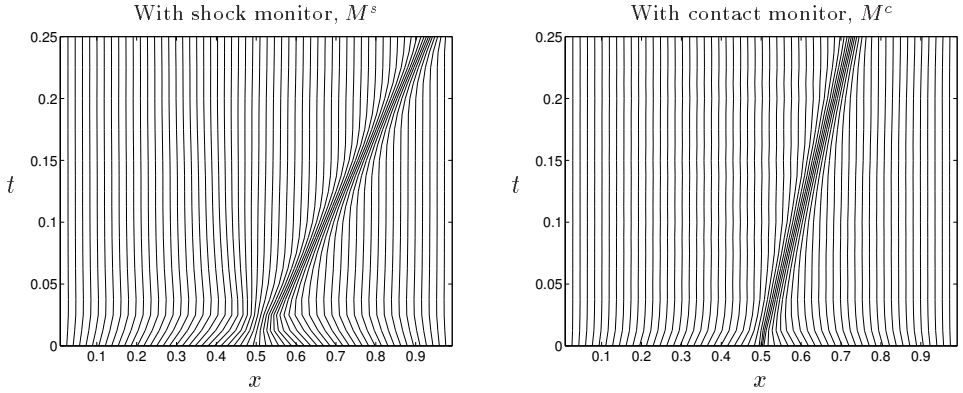


FIG. 4.11. *Mesh contours obtained using the shock and contact monitor functions ($\tau = 0.005$, $N = 60$).*

TABLE 4.4
Comparison of various fixed and moving grid solutions for the Euler equations.

Description	*CPU time*	*NT*	L^1 *error*	$\mathcal{R}$
Fixed grid:				
$N = 60$	0.82	56	0.0047	
$N = 120$	2.03	98	0.0026	
$N = 240$	6.60	179	0.0013	
$N = 480$	21.19	339	0.0006	
Moving grid ($N = 60$, $\tau = 0.005$)*:*				
α^{avg} (M based on $\|q_x\|_2^2$)	5.46	152	0.0032	0.0123
Moving grid ($N = 60$, $\tau = 0.005$, α^{max})*:*				
$M = M^c$	5.03	103	0.0041	0.0145
$M = M^s$	4.50	117	0.0043	0.0136
$M = M^{cs}$	3.32	80	0.0026	0.0150

discontinuity, although the errors in Table 4.4 demonstrate the loss in accuracy suffered when only one of the two discontinuous flow features is resolved. Furthermore, the computational time is increased, due primarily to reductions in time step from local CFL violations near the discontinuity that is not being adequately resolved.

The results computed with the monitor function M^{cs} are displayed in Fig. 4.12. While the solution is not as sharp at the individual fronts as the solution obtained

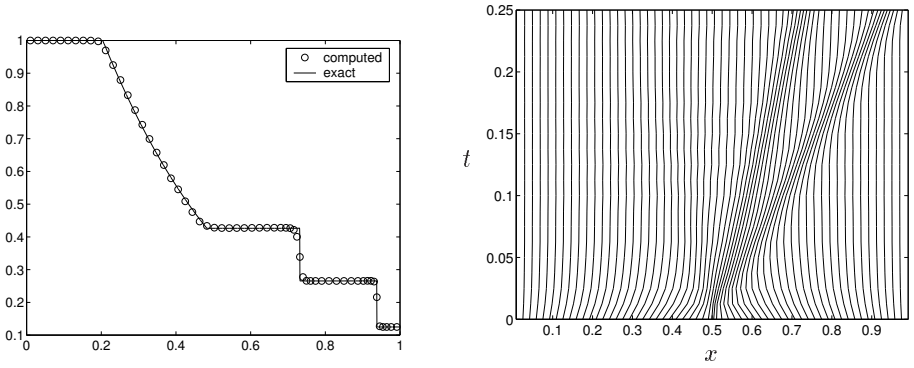


FIG. 4.12. *Moving mesh solution computed with the discontinuity-adaptive monitor function* M^{cs} ($\tau = 0.005$, $N = 60$).

with each monitor function individually, the averaged monitor function does capture both discontinuities and improves the overall accuracy in the solution relative to the fixed mesh method. The resolution of the contact line is especially good considering the difficulties in resolving this class of wave that are reported for other moving mesh methods (e.g., [12, 22]). It is also important to mention here that we have made no effort to optimize the efficiency of the moving mesh component of the code, and it is certain that considerable gains in CPU time can still be made.

While it is difficult to perform an in-depth comparison of accuracy and CPU time with other moving mesh results reported in the literature, we can still make some general comments. When compared to other methods (e.g., [14], [2] and [22]), even

those that employ fully implicit time-stepping, our moving mesh scheme requires at most the same number of time steps for a comparable level of accuracy. If we then take into account the added cost of evaluating and inverting the Jacobian matrix in implicit methods, our scheme is much cheaper. Furthermore, in the case of the Euler equations, we observe a significant improvement in the resolution of contact lines.

5. Conclusions. We have developed an adaptive method for solving one-dimensional hyperbolic conservation laws that is accurate and efficient. This is despite the fact that difficulties identified with solving parabolic problems on moving meshes in the literature (namely, mesh racing and ill-conditioning of the solution-mesh iteration) are exacerbated for hyperbolic problems due in large part to the absence of a natural dissipative mechanism in the physical equations.

The method is based on the existing fast, wave propagation algorithms implemented in the software package CLAWPACK. The accuracy of our results derives from our combining the high resolution, Godunov-type scheme for the physical equations with a moving mesh PDE that uses a monitor function specifically designed to capture solution discontinuities. Numerical results demonstrate the efficacy of the method in comparison with fixed mesh computations and other moving mesh results reported in the literature, particularly as regards resolution of contact discontinuities.

The efficiency of our solution algorithm stems primarily from our construction of a semi-implicit iteration for the solution and mesh, that behaves in practice as a two- or three-step predictor–corrector method. This should be compared with other moving mesh algorithms, which typically employ a fully implicit time discretization for the coupled solution–mesh system that iterates both solution and mesh to convergence. The rapid convergence of the iteration is aided by a combination of a discontinuity-adaptive monitor function that avoids over-resolving sharp fronts with temporal smoothing that mollifies the effects of mesh racing. The temporal smoothing parameter τ is the only free parameter required in the moving mesh algorithm. However, the level of smoothing required varies between problems, and so this is one area where further work is required. It is likely that performance can be enhanced by allowing τ to depend on the solution throughout a given computation.

We have come to two fundamental conclusions regarding adaptive solution of hyperbolic problems. First, the arclength monitor function, which is the one most commonly used in moving mesh computations, is not appropriate for hyperbolic problems, where high concentrations of mesh points near discontinuous solution features can cause the mesh equations to become ill-conditioned. Second, temporal smoothing is essential for controlling mesh racing, which can otherwise cause serious CFL violations and further degrade the solution–mesh iteration.

We are not advocating r-adaptive methods as a panacea for all hyperbolic problems, particularly because of the proven success of static refinement methods such as AMR [3]. However, we have shown that there is considerable advantage to be gained by some degree of dynamic mesh movement, and so an optimal "overall" solution strategy is likely to derive from a combination of grid movement (r–refinement) and grid subdivision (h–refinement).

The most obvious extension of this work is to higher dimensional problems where multiple shock interactions pose particular challenges for mesh adaptation. The CLAWPACK code has already proven very effective for problems on non-uniform stationary grids in two dimensions, and so our next step will be to generalize the wave propagation scheme to a moving grid. When used in combination with the variational approach for 2D moving meshes developed in Cao *et al.* [7], our discontinuity-adaptive

monitor function should exhibit similar advantages in higher dimensions.

There is also a need for further theoretical work on the behaviour of the nonlinear system of equations for the solution and mesh. It is our hope that an analysis for hyperbolic problems (perhaps in the context of the scalar advection equation) will provide insight into the importance of both the monitor function and temporal smoothing, and their relationship to the phenomenon of mesh racing.

REFERENCES

[1] G. Beckett and J. A. Mackenzie. On a uniformly accurate finite difference approximation of a singularly perturbed reaction-diffusion problem using grid equidistribution. Report 97/30, Department of Mathematics, University of Strathclyde, Glasgow, Scotland, 1997.

[2] J. B. Bell and G. R. Shubin. An adaptive grid finite difference method for conservation laws. *J. Comput. Phys.*, 52:569–591, 1983.

[3] M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Appl. Math.*, 35(6):2298, 1998.

[4] R. Biswas, J. E. Flaherty, and D. C. Arney. An adaptive mesh-moving and refinement procedure for one-dimensional conservation laws. *Appl. Numer. Math.*, 11:259–282, 1993.

[5] C. J. Budd, W. Huang, and R. D. Russell. Moving mesh methods for problems with blow-up. *SIAM J. Sci. Comp.*, 17(2):305–327, Mar. 1996.

[6] W. Cao, W. Huang, and R. D. Russell. An r-adaptive finite element method based upon moving mesh PDEs. *J. Comput. Phys.*, 149(2):221–244, 1999.

[7] W. Cao, W. Huang, and R. D. Russell. A study of monitor functions for two-dimensional adaptive mesh generation. *SIAM J. Sci. Comp.*, 1999. To appear.

[8] N. N. Carlson and K. Miller. Design and application of a gradient-weighted moving finite element code I: in one dimension. *SIAM J. Sci. Comp.*, 19(3):728–765, 1998.

[9] J. Chen. Numerical study of blowup problems and conservation laws with moving mesh methods. Master's thesis, Simon Fraser University, Burnaby, British Columbia, Canada, June 1996. (unpublished).

[10] K. Chen. Error equidistribution and mesh adaptation. *SIAM J. Sci. Comp.*, 15(4):798–818, 1994.

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

[12] K. Farrell and L. O. Drury. An explicit adaptive grid algorithm for one-dimensional initial value problems. *Appl. Numer. Math.*, 26(1/2):3–12, 1998.

[13] R. Fazio and R. J. LeVeque. Moving-mesh methods for one-dimensional hyperbolic problems using Clawpack. (unpublished, available at `ftp://amath.washington.edu/pub/rjl/papers/rf-rjl:movingmesh.ps.gz`), 1998.

[14] R. M. Furzeland, J. G. Verwer, and P. A. Zegeling. A numerical study of three moving grid methods for one-dimensional partial differential equations which are based on the method of lines. *J. Comput. Phys.*, 89(2):349–388, 1990.

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

[16] S. K. Godunov. Difference method of numerical computation of discontinuous solutions of hydrodynamic equations. *Mat. Sb.*, 47:271–306, 1959. (In Russian).

[17] A. Harten and J. M. Hyman. Self-adjusting grid methods for one-dimensional hyperbolic conservation laws. *J. Comput. Phys.*, 50:235–269, 1983.

[18] W. Huang, Y. Ren, and R. D. Russell. Moving mesh methods based on moving mesh partial differential equations. *J. Comput. Phys.*, 113:279–290, 1994.

[19] W. Huang and R. D. Russell. A moving collocation method for solving time dependent partial differential equations. *Appl. Numer. Math.*, 20:101–116, 1996.

[20] R. J. LeVeque. Clawpack – A software package for solving multidimensional conservation laws. In *Hyperbolic problems: theory, numerics, applications*, pages 188–197. World Scientific Publishing, River Edge, NJ, 1996.

[21] R. J. LeVeque. Wave propagation algorithms for multi-dimensional hyperbolic systems. *J. Comput. Phys.*, 131(2):327–353, 1997.

[22] S. Li. *Adaptive Methods and Software for Time-Dependent Partial Differential Equations.* PhD thesis, University of Minnesota, 1998.

[23] S. Li, L. Petzold, and Y. Ren. Stability of moving mesh systems of partial differential equations. *SIAM J. Sci. Comp.*, 20(2):719–738, 1999.

[24] L. R. Petzold. A description of DASSL: A differential/algebraic system solver. In R. S. Stepleman, editor, *Transactions on Scientific Computation*. IMACS, New Brunswick, NJ, 1982.

[25] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys.*, 43:357–372, 1981.

[26] J. H. Smith and A. M. Stuart. Analysis of continuous moving mesh equations. Technical Report SCCM-96-12, Scientific Computing and Computational Mathematics Program, Stanford University, 1996.

[27] G. A. Sod. A survey of finite difference methods for systems of nonlinear hyperbolic conservation laws. *J. Comput. Phys.*, 27:1–31, 1978.

[28] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM J. Numer. Anal.*, 21(5):995–1011, 1984.

[29] J. G. Verwer, J. G. Blom, and J. M. Sanz-Serna. An adaptive moving grid method for one-dimensional systems of partial differential equations. *J. Comput. Phys.*, 82(2):454–486, 1989.

[30] K.-H. A. Winkler, M. L. Norman, and M. J. Newman. Adaptive mesh techniques for fronts in star formation. *Physica D*, 12:408–425, 1984.

[31] P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *J. Comput. Phys.*, 54:115–173, 1984.