Adaptive Mesh Refinement using Wave-Propagation Algorithms for Hyperbolic Systems

MARSHA J. BERGER[1] AND RANDALL J. LEVEQUE[2]

Dedicated to Ami Harten for his many contributions and warm sense of humor

Abstract. An adaptive mesh refinement algorithm developed for the Euler equations of gas dynamics has been extended to employ high-resolution wave-propagation algorithms in a more general framework. This allows its use on a variety of new problems, including hyperbolic equations not in conservation form, problems with source terms or capacity functions, and logically-rectangular curvilinear grids. This framework requires a modified approach to maintaining consistency and conservation at grid interfaces, which is described in detail. The algorithm is implemented in the AMRCLAW package, which is freely available.

1 Introduction

The multi-dimensional wave-propagation algorithm described in [14] is a "high-resolution" method that is second order accurate on smooth solutions while maintaining sharp discontinuities through the use of slope-limiters. While based on ideas developed for hyperbolic systems of conservation laws

$$q_t + f(q)_x + g(q)_y = 0$$

in the context of shock capturing, these methods apply in a more general framework that allows their application to other hyperbolic systems which are not in conservation form. Variable-coefficient hyperbolic systems of the form

$$q_t + A(x,y)q_x + B(x,y)q_y = 0$$

arise, for example, in studying acoustics or elasticity in heterogeneous materials with varying material properties (see Section 6).

For most practical problems, it is desirable to use mesh refinement to cluster grid points in regions where they are most needed, for example around shocks or other regions where the solution

[1]Courant Institute, New York University, 251 Mercer St., New York, NY 10012. berger@cims.nyu.edu
[2]Department of Applied Mathematics and Department of Mathematics, University of Washington, Box 352420, Seattle, WA 98195-2420. rjl@amath.washington.edu

January, 1997. To appear in a special issue of SIAM J. Numer. Anal. dedicated to Ami Harten.
Revised July, 1997.

has steep gradients or complicated structure. This should be done in an adaptive manner, based on the behavior of the solution, and for time-dependent problems the region of refinement must move adaptively with the interesting structure. An effective adaptive mesh refinement (AMR) strategy has been developed [1–6] that uses Cartesian grids, with refinement in both space and time, over rectangular patches. The refinement is by an arbitrary even integer ratio (typically 4) and further recursive refinement can be done within these patches to an arbitrary depth. This algorithm was originally developed for the Euler equations of gas dynamics using flux-differencing methods, but can be easily extended to other systems of conservation laws. A crucial ingredient is the manner in which fluxes at grid-refinement interfaces are coordinated to insure that the method is globally conservative. This is described in detail in [5] and reviewed in Section 2.1.

In this paper we show how the wave-propagation algorithm can be used in conjunction with this AMR strategy. This algorithm is written in a more general form that does not use flux differencing *per se*, though in the special case of a conservation law the method can be re-expressed in flux-differencing form and is fully conservative. In Section 4 we show that at grid-refinement interfaces it is possible to apply a correction procedure that maintains global conservation when applied on a conservation law (and reduces to the flux-based approach), while retaining the more general wave-propagation framework that allows application on more general hyperbolic systems.

We also discuss how further extensions presented in [14] can be incorporated into AMR. This includes source terms, capacity-form differencing, and applications on mapped curvilinear grids. These topics are discussed in Sections 5 and 6.

The domain is assumed to be rectangular, at least in computational space. Metric terms can be included so that we are actually solving a problem on a nonrectangular physical domain with a curvilinear grid, obtained by a smooth mapping of the Cartesian computational grid. The grid Jacobian function can be properly incorporated using the "capacity function" described in Section 5.2.

The wave-propagation algorithm has been implemented in a general software package (in FORTRAN) called CLAWPACK (Conservation LAWs PACKage, a holdover from earlier versions that applied only to conservation laws). This package is available from `netlib`[11]. This software has recently been combined with Berger's implementation of AMR, incorporating the extensions described in this paper. This is now freely available as the AMRCLAW package[6]. All of the numerical results presented in this paper were obtained with AMRCLAW, and several such examples are included with the package. Details on the use of this package can be found in the on-line documentation.

2 The AMR algorithm

The adaptive mesh refinement algorithm for conservation laws is fully described in [5] and only a brief summary will be given here. The AMR approach to adaptive mesh refinement uses a collection of logically rectangular meshes that make up the coarse grid; refinements cover a subset of the domain and use smaller rectangular grid patches. These fine patches can be recursively nested until a given level of accuracy is attained. Typically, if a patch at Level L is refined in x and y by an even integer R_L, then the time step is also refined by the same factor, so that R_L time steps must be taken on the refined grid at Level $L+1$ for each step on the grids at Level L. The mesh ratios $\Delta t/\Delta x$ and $\Delta t/\Delta y$ are then the same on all grids, ensuring stability with explicit difference schemes.

Every K time steps on a particular grid level, all finer level grids are regenerated in order to follow moving features of the flow. An error estimation procedure based on Richardson extrapolation determines the regions where resolution of the solution is insufficient. This procedure compares the solution obtained by taking 2 steps on the existing grid with one computed by taking 1 time step on a grid that is twice as coarse in each direction. Cells where the error is greater than some tolerance are flagged for refinement. Other criteria might be used in addition to, or instead of, this error estimate, e.g. identifying steep gradients in some variable. A buffer zone around the flagged cells is also flagged to insure that features of interest do not escape from the refinement region over the next K time steps. The buffer width and K are adjustable parameters which must be coordinated.

Flagged cells are then organized into rectangular grid patches, typically containing several hundred to several thousand grid points per patch. Note that some cells not tagged for refinement are also included in new fine grid patches. Typically, our grid generation algorithm produces grids with 70% of the cells within new grid patches tagged as needing refinement; the remaining 30% are untagged but still lie within the new patch boundaries. By taking very small patches one could avoid refining too many cells, but this must be balanced with the competing desire to create relatively few separate patches, and to minimize computational overhead on the boundaries of fine grids. See [1] for more details on the refinement and clustering algorithms. See Figure 6 for an example of refinement on logically rectangular patches.

An alternative would be to use a quadtree data structure (*e.g.*, [7], [15]) in which only the flagged cells are refined, but the storage overhead of these data structures, typically 30 to 50 words per cell, usually exceeds the storage overhead associated with the block structured approach, even with the 30% additionally refined grid.

A finite volume method is used to advance the solution on the resulting grid hierarchy. Cell averages of each variable are stored in each grid cell. When solving a conservation law, these cell averages are updated by a flux-differencing algorithm based on fluxes through the cell edges. (In section 3 we review wave-propagation methods and their extension to nonconservative hyperbolic systems.) The integration proceeds by grid level. All grids on level 1 are first integrated over a time step. Then grids at level 2 are integrated over R_1 time steps to catch up. This approach is applied recursively on each level.

Boundary conditions on all grids are imposed using "ghost cells". The computational domain is extended by G ghost cells in each direction, and values are assigned to the ghost cells at the start of each time step. (G=2 for the methods described here). At a physical boundary, the user must set the ghost cell values at each time step based on the problem specification (e.g. extrapolation at an outflow boundary or reflection at a solid wall). However, the boundary of a fine grid may be interior to the domain. In this case, if there is no neighboring fine grid to supply values for the ghost cells, they are interpolated from a coarser parent grid. Space-time interpolation must in general be used since more time steps are taken on the fine grid than on the coarse grid, and at intermediate times there are no coarse grid values available. However, since coarse grids are always advanced first, we have data available from both an earlier and later time on the coarse grid from which to interpolate.

The techniques developed in this paper for adaptive refinement are most easily described in one space dimension, where the refined grid patches become intervals. We concentrate on this case in the development below, but all of these techniques carry over directly to multiple space dimensions, and have been implemented in two dimensions in the AMRCLAW software. Two-dimensional results are presented in Section 6.

2.1 Conservation at grid interfaces

At an interface between coarse and fine grids, we must also insure that the formulas used to update the solution on each grid are consistent with one another. In particular, when a conservation law is being solved we must preserve global conservation.

To illustrate the conservative flux correction required at grid interfaces, and the manner in which this must be modifed for the wave-propagation algorithms, we will consider the one-dimensional case with only two levels of grid and a factor of two refinement. Denote the coarse grid spacing by h and the time step by k. The grid spacing on the fine grid is $\hat{h} = h/2$ and $\hat{k} = k/2$. Figure 1(a) shows the coarse grid over a single time step in x-t space. Figure 1(b) shows also the finer grid that is overlaid. We will denote the values on this fine grid by $\hat{q}_i$, and assume that there are m cells on this grid, which ends just before cell j on the coarse grid.

On the coarse grid we compute fluxes F_i^0 as indicated in Figure 1(a), and update the coarse grid values by the flux-differencing algorithm

$$q_i^1 = q_i^0 - \frac{k}{h}(F_{i+1}^0 - F_i^0). \tag{1}$$

Note that F_i represents the flux at the left edge of cell i, which is the interface between grid cells $i-1$ and i, instead of the commonly-used but more cluttered notation $F_{i-1/2}$.

On the fine grid we use the fluxes indicated in Figure 1(c). In each of the two time steps on this grid, we will use flux-differencing of the form

$$\hat{q}_i^{n+1} = \hat{q}_i^n - \frac{\hat{k}}{\hat{h}}(\hat{F}_{i+1}^n - \hat{F}_i^n) \tag{2}$$

for $i \le m$ and $n = 1, 2$. To compute fluxes near the right edge of this grid, we will need to use values in ghost cells which are also indicated in Figure 1(c). These values are determined using space-time interpolation from the coarse grid values q_j^0 and q_j^1, as described above.

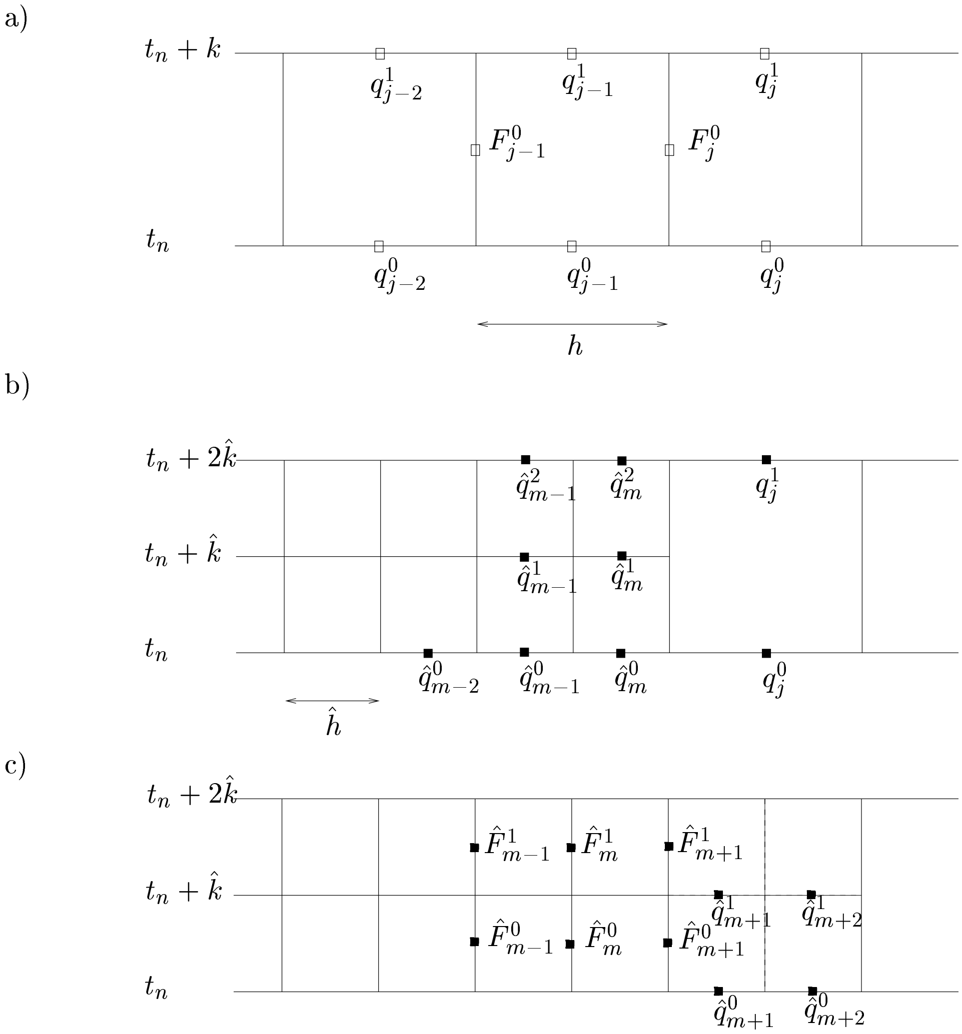


Figure 1: *a) Coarse grid for the one-dimensional example, shown in space-time. Values of* q *and fluxes* F *are indicated. b) Also shows the fine grid that is overlaid for and the grid interface. Values of* $\hat{q}$ *on the fine grid are indicated. c) The flux values* $\hat{F}$ *needed on the fine grid. The ghost cells* $\hat{q}_{m+1}$ *and* $\hat{q}_{m+2}$ *are also indicated. These values are needed to compute the fine grid fluxes at the grid interface.*

We now need to coordinate the values obtained on the two different grids at the final time. First, in the coarse grid cells overlapped by fine grid cells ($i \leq j-1$), we replace the value q_i^1 by the average of the fine-grid values:

$$q_{j-1}^1 \;:=\; \frac{1}{2}(\hat{q}_{m-1}^2 + \hat{q}_m^2). \tag{3}$$

This is sensible since the fine-grid values are presumably more accurate than the coarse-grid value, and is also crucial in maintaining global conservation when regridding, if the fine grid is eliminated, and important in maintaining stability since information must be allowed to pass from the fine to coarse grid as well as in the other direction.

To be fully conservative, however, we also must modify the coarse-grid value q_j^1. In initially computing q_j^1 we used the coarse-grid flux F_j^0 at the left edge of this cell. To be conservative, we must instead use a left-edge flux that agrees with the flux used in determining the fine-grid values to the left of this cell. Instead of using F_j^0, we should use $\frac{1}{2}(\hat{F}_{m+1}^0 + \hat{F}_{m+1}^1)$, and hence q_j^1 is corrected by the difference between the two:

$$q_j^1 \;:=\; q_j^1 + \frac{k}{h}\left[\frac{1}{2}(\hat{F}_{m+1}^0 + \hat{F}_{m+1}^1) - F_j^0\right]. \tag{4}$$

We then have global conservation of the total mass at multiples of the coarse grid time step in the sense that the total mass

$$\hat{h}\sum_{i\leq m}\hat{q}_i + h\sum_{i\geq j} q_j$$

is conserved up to boundary effects at the farfield.

3 The wave-propagation algorithms

In one space dimension, the wave-propagation algorithm described in [14] is based on solving a Riemann problem at each interface between grid cells, and using the resulting wave structure to update the solution in the grid cell to each side. This is, of course, the basis for a host of methods for conservation laws, dating back to Godunov's method[8]. The Riemann problem consists of the original conservation law together with piecewise constant initial data defined by the two neighboring cell values. For a wide class of conservation laws, the Riemann problem can be solved (either exactly or approximately) and the solution is a similarity solution consisting of a set of waves propagating at constant speeds. For a system of conservation laws, this solution can be used to define a flux at the cell interface, yielding Godunov's method. Second order accuracy can be achieved in various ways, (e.g. by introducing slope information), and "slope-limiters" or "flux-limiters" are then used to give good resolution of discontinuities without spurious oscillations. See, *e.g.*, [12] for a general discussion of such methods.

The wave-propagation algorithms are based on using the waves directly to update cell values, including second order corrections with "wave limiters". For conservation laws these methods can be rewritten in conservation form by defining flux functions in terms of the waves, but they are implemented in a way that allows their application to hyperbolic problems not in conservation form, for which there is still a well-defined wave structure but no flux function. To illustrate this, consider the advection equation with variable velocity,

$$q_t + u(x)q_x = 0. \tag{5}$$

This equation, sometimes called the "color equation", is not in conservation form. The value of q is constant along characteristics but the integral of q is not conserved. Assume $u(x) > 0$ everywhere.

At the interface $x_{i-1/2}$ between cells $i-1$ and i we can define the Riemann problem as

$$\bar{q}_t + \bar{u}(x)\bar{q}_x = 0$$

where the function $\bar{u}(x)$ and initial data $\bar{q}$ are given by

$$\bar{q}(x,0) = \begin{cases} q_{i-1} & \text{if } x < x_{i-1/2} \\ q_i & \text{if } x > x_{i-1/2} \end{cases} \qquad \bar{u}(x) = \begin{cases} u_{i-1} & \text{if } x < x_{i-1/2} \\ u_i & \text{if } x > x_{i-1/2} \end{cases} \tag{6}$$

where u_{i-1} and u_i are cell-centered values of $u(x)$ in cells $i-1$ and i. In the solution to this Riemann problem, the wave $\mathcal{W}_i \equiv \Delta q = q_i - q_{i-1}$ simply propagates with speed u_i (to the right since $u > 0$). Over time step k this wave moves distance ku_i into cell i, and modifies the cell average q_i by $\frac{k}{h}u_i(q_i - q_{i-1})$. The first order upwind method, in wave-propagation form, is thus

$$q_i^{n+1} = q_i - \frac{k}{h}u_i(q_i - q_{i-1}).$$

More generally, if $u(x)$ has arbitrary sign, the first order wave-propagation algorithm is

$$q_i^{n+1} = q_i - \frac{k}{h}(u_i^+(q_i - q_{i-1}) + u_i^-(q_{i+1} - q_i)). \tag{7}$$

where $u^+ = \max(u, 0)$ and $u^- = \min(u, 0)$.

Alternatively, the Riemann problem might be defined by using an edge value $u_{i-1/2}$ at the interface between cells $i-1$ and i. The update formula (7) would then become

$$q_i^{n+1} = q_i - \frac{k}{h}(u_{i-1/2}^+(q_i - q_{i-1}) + u_{i+1/2}^-(q_{i+1} - q_i)). \tag{8}$$

This has advantages in two-dimensional incompressible flow (see [13]), but for illustration here we will use the formulation (7), which is based on the cell-centered velocites (6) rather than edge values.

High-resolution second-order corrections are easily introduced, and in fact this can be done in a "flux-differencing" form even for nonconservative equations such as the color equation. This simplifies the procedure for insuring conservation when applied to a conservation law. At the cell interface $x_{i-1/2}$, we define

$$\tilde{F}_i = \frac{1}{2}|u_i|\left(1 - \frac{k}{h}|u_i|\right)\tilde{\mathcal{W}}_i$$

where $\tilde{\mathcal{W}}_i$ is a limited version of the wave $\mathcal{W}_i = q_i - q_{i-1}$:

$$\tilde{\mathcal{W}}_i = \begin{cases} \texttt{limiter}(\mathcal{W}_i, \mathcal{W}_{i-1}) & \text{if } u_i > 0 \\ \texttt{limiter}(\mathcal{W}_i, \mathcal{W}_{i+1}) & \text{if } u_i < 0 \end{cases}$$

where $\texttt{limiter}(a, b)$ represents some standard limiter such as minmod or superbee. After including these corrections, the method (7) becomes

$$q_i^{n+1} = q_i - \frac{k}{h}(u_i^+(q_i - q_{i-1}) + u_i^-(q_{i+1} - q_i)) - \frac{k}{h}(\tilde{F}_{i+1} - \tilde{F}_i). \tag{9}$$

Note that if u is constant then the advection equation (5) is a conservation law, and in this case (9) reduces to (assuming $u > 0$, for example),

$$\begin{aligned} q_i^{n+1} &= q_i - \frac{k}{h}u(q_i - q_{i-1}) - \frac{k}{h}(\tilde{F}_{i+1} - \tilde{F}_i) \\ &= q_i - \frac{k}{h}(F_{i+1} - F_i) \end{aligned}$$

where

$$F_i = uq_{i-1} + \tilde{F}_i.$$

This is the numerical flux for a standard flux-limiter method on the advection equation (see [19], for example). In particular, if no limiter is used and $\tilde{\mathcal{W}}_i \equiv \mathcal{W}_i$, this reduces to the Lax-Wendroff method.

The more general form (9) is not in flux-differencing form, but works just as effectively on the color equation as standard flux-differencing does on the constant-coefficient advection equation. The form (9) is easily extended to general hyperbolic systems. Consider the variable-coefficient linear system

$$q_t + A(x)q_x = 0$$

where now $q \in \mathbb{R}^m$ and $A(x) \in \mathbb{R}^{m\times m}$ is diagonalizable with real eigenvalues. Then for the Riemann problem at $x_{i-1/2}$ we decompose $q_i - q_{i-1}$ into waves $\mathcal{W}_i^p$ (for $p = 1,\ 2,\ \dots,\ m$) in such a way that left-going waves are eigenvectors of the matrix A_{i-1} defined on cell $i-1$, traveling with speeds $\lambda_{i-1}^p < 0$ (eigenvalues of A_{i-1}), while the right-going waves are eigenvectors of A_i traveling with speeds $\lambda_i^p > 0$. The update formula is then

$$q_i^{n+1} = q_i^n - \frac{k}{h}\sum_{p=1}^{m}\left[(\lambda_i^p)^+\mathcal{W}_i^p + \left(\lambda_{i+1}^p\right)^-\mathcal{W}_{i+1}^p\right] - \frac{k}{h}(\tilde{F}_{i+1} - \tilde{F}_i).$$

The summation term gives the first-order upwind method, while the $\tilde{F}_i$ fluxes are again the second-order corrections defined now by

$$\tilde{F}_i = \frac{1}{2}\sum_{p=1}^{m} |\lambda_i^p| \left(1 - \frac{k}{h}|\lambda_i^p|\right) \tilde{\mathcal{W}}_{i+1} \tag{10}$$

where $\tilde{\mathcal{W}}_{i+1}$ is a limited version of the wave $\mathcal{W}_i^p \in \mathbb{R}^m$ obtained by comparing it with $\mathcal{W}_{i+1}^p$ (if $\lambda_i^p < 0$) or with $\mathcal{W}_{i-1}^p$ (if $\lambda_i^p > 0$). More details, including a worked example for acoustics in a heterogeneous medium, can be found in [14] and the documentation with [11].

The wave-propagation algorithm is extended to nonlinear systems of conservation laws using a Roe approximate Riemann solver[16], which linearizes the problem at each cell interface in such a way that the wave-propagation approach is guaranteed to be conservative. The general wave-propagation algorithm is written symbolically as

$$q_i^{n+1} = q_i^n - \frac{k}{h}[\mathcal{A}^+\Delta q_i + \mathcal{A}^-\Delta q_{i+1}] - \frac{k}{h}[\tilde{F}_{i+1} - \tilde{F}_i] \tag{11}$$

where $\mathcal{A}^+\Delta q_i$ represents the right-going "fluctuation" from the i'th Riemann problem, at the left edge of Cell i, while $\mathcal{A}^-\Delta q_{i+1}$ is the left-going fluctuation from the Riemann problem at the right edge of this cell. Each fluctuation is just the sum over all waves moving in the appropriate direction of the wave speed multiplied by the wave strength. The notation is motivated by the fact that, for the constant-coefficient linear system $q_t + Aq_x = 0$, we have

$$\mathcal{A}^+\Delta q_i = A^+(q_i - q_{i-1}) = \sum_{p=1}^{m}(\lambda^p)^+\mathcal{W}_i^p$$

and

$$\mathcal{A}^-\Delta q_i = A^-(q_i - q_{i-1}) = \sum_{p=1}^{m}(\lambda^p)^-\mathcal{W}_i^p$$

where $A^\pm = R\Lambda^\pm R^{-1}$ with $A = R\Lambda R^{-1}$ being the eigendecomposition of A, so that $R = [r^1|r^2|\cdots|r^m]$ is the matrix of right eigenvectors. The waves $\mathcal{W}^p$ in this case are given by $\mathcal{W}^p = \alpha^p r^p$ where the vector of wave strengths is $\alpha = R^{-1}\Delta q$.

Note that in this linear case $A^+ + A^- = A$. For a general conservation law, the method (11) is conservative provided that

$$\mathcal{A}^+\Delta q_i + \mathcal{A}^-\Delta q_i = f(q_i) - f(q_{i-1}), \tag{12}$$

i.e., the fluctuations are defined by a flux-difference splitting. This is easy to see since we can then define

$$\bar{F}_i \equiv f(q_i) - \mathcal{A}^+\Delta q_i = f(q_{i-1}) + \mathcal{A}^-\Delta q_i \tag{13}$$

so that

$$\begin{aligned}\bar{F}_{i+1} - \bar{F}_i &= (f(q_i) + \mathcal{A}^- \Delta q_{i+1}) - (f(q_i) - \mathcal{A}^+ \Delta q_i) \\ &= \mathcal{A}^- \Delta q_{i+1} + \mathcal{A}^+ \Delta q_i\end{aligned}$$

Using this in (11) shows that the method can be rewritten in conservation form, with flux $F_i = \bar{F}_i + \tilde{F}_i$.

4 Wave-propagation at grid interfaces

At the interface between a fine and coarse grid, the wave-propagation form can still be used to update the values on each grid independently, using "ghost cell" values as needed near grid interfaces. We can still replace the new coarse grid value by an average of fine grid values in any cell covered by a fine grid. The only tricky part in extending the AMR algorithm to wave-propagation algorithms is the conservative correction of the coarse cells adjacent to finer grids, *e.g.*, the value q_j^1 in Figure 1(b). Recall that with the flux-differencing algorithm this value must be modified by the correction (4) to insure conservation, since then the flux "into" the coarse cell agrees with the total flux "out of" the adjacent fine cells. With the wave-propagation algorithm, we must apply a similar fix-up to insure that the waves match up in an appropriate manner to yield conservation when conservation is expected. This turns out to be only slightly more difficult when no numerical flux is available than for the flux-differencing form discussed in Section 2.

Note that the second order correction terms are written in flux-differencing form $\tilde{F}_{i+1} - \tilde{F}_i$ even for nonconservative systems, and so these terms can again be corrected using (4). It is only the first-order upwind terms written in terms of the fluctuations $\mathcal{A}^-\Delta q$ and $\mathcal{A}^+\Delta q$ that must be handled differently, and so we concentrate on the first-order alogrithm below.

Both the difficulty and the solution can be most easily seen by examining the constant coefficient advection equationn $q_t + uq_x = 0$ on the one-dimensional grid of Figure 1, in the case $u < 0$, so that waves are moving from the coarse grid to the fine grid. Suppose we are solving the Cauchy problem and the data has compact support while the fine and coarse grids extend off to $-\infty$ and $+\infty$ respectively. Then we hope to have conservation in the sense that

$$\hat{h} \sum_{i \le m} \hat{q}_i^2 + h \sum_{i \ge j} q_i^1 = \hat{h} \sum_{i \le m} \hat{q}_i^0 + h \sum_{i \ge j} q_i^0.$$

On the coarse grid we have

$$q_i^1 = q_i^0 - \frac{k}{h} u(q_{i+1}^0 - q_i^0)$$

and when we sum this over the coarse grid cells we get a telescoping of the q-differences everywhere except in the first cell, so that

$$h \sum_{i \ge j} q_i^1 = h \sum_{i \ge j} q_i^0 + kuq_j^0.$$

Similarly, after 2 steps on the fine grid, we find that

$$\hat{h}\sum_{i\leq m}\hat{q}_i^2 = \hat{h}\sum_{i\leq m}\hat{q}_i^0 - \hat{k}u(\hat{q}_{m+1}^0 + \hat{q}_{m+1}^1).$$

Summing these two results we see that the method is globally conservative only if we add in a correction to this global sum of magnitude

$$-kuq_j^0 + \hat{k}u(\hat{q}_{m+1}^0 + \hat{q}_{m+1}^1).$$

This can be accomplished by modifying the value q_j^1 by

$$\frac{k}{h}u\left[\frac{1}{2}(\hat{q}_{m+1}^0 + \hat{q}_{m+1}^1) - q_j^0\right]. \tag{14}$$

Note that this is exactly the modification (4) obtained when the flux-differencing form is used, since for the advection equation $f(q) = uq$.

Instead of relying on a flux function, we will interpret this correction in another way using wave propagation. Figure 2(a) shows all of the waves which affect the relevant cell values at the end of the time increment, if we only apply the wave-propagation algorithm on each grid separately. Clearly something is wrong with this picture. If we view the initial data as defining a piecewise constant function, then we need to solve the Riemann problem at each discontinuity and add the total fluctuation from the Riemann problem $(\mathcal{A}^-\Delta q + \mathcal{A}^+\Delta q)$ to some grid value(s) in order to maintain conservation. In general $\mathcal{A}^-\Delta q$ is added to the cell on the left, and $\mathcal{A}^+\Delta q$ to the cell on the right. As Figure 2(a) shows, we have failed to solve the Riemann problem between states $\hat{q}_{m+1}^0$ (the ghost-cell value on the fine grid) and q_j^0 at the initial time. To restore conservation we must solve the Riemann problem between these states and add in the resulting total fluctuation $\mathcal{A}^-\Delta q + \mathcal{A}^+\Delta q$, weighted by $\hat{k}/h = 1/2$ since the time step is $\hat{k}$ while the cell size is h, to the cell value q_j^1. In terms of maintaining conservation, this sort of correction could equally well be added to some other cell value(s) instead of to q_j^1, but this choice is clearly most reasonable from the figure and agrees with how modifications are applied in the flux-differencing framework.

Similarly, in the second step on the fine grid we must also solve a Riemann problem between $\hat{q}_{m+1}^1$ and q_j^0 and add these fluctuations into q_j^1. For the advection example considered above, these two corrections will sum to exactly the required correction (14), since this can be rewritten as

$$\frac{\hat{k}}{h}[u(\hat{q}_{m+1}^0 - q_j^0) + u(\hat{q}_{m+1}^1 - q_j^0)].$$

This correction is easily extended to an arbitrary hyperbolic system, since we presumably have a Riemann solver that produces $\mathcal{A}^-\Delta q$ and $\mathcal{A}^+\Delta q$ from the two states $\hat{q}_{m+1}^0$ and q_j^0. We modify q_j^1 by

$$\frac{\hat{k}}{h}(\mathcal{A}^-\Delta q + \mathcal{A}^+\Delta q). \tag{15}$$

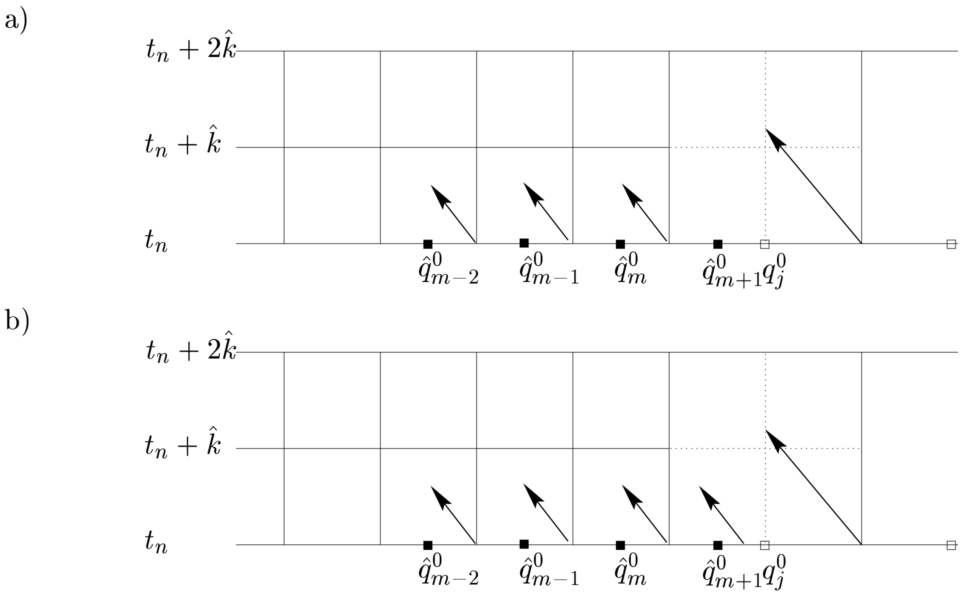


Figure 2: *a) Waves arising from solving all Riemann problems on the fine and coarse grids separately. b) The wave arising from solving the Riemann problem between the ghost cell value* $\hat{q}^0_{m+1}$ *and the coarse grid value* q^0_j *must also be included in order to maintain conservation.*

A similar modification must be made in each of the R time steps on the refined grid within the single coarse grid step, where R is again the refinement ratio. The fix-up algorithm thus takes the form

for $N = 0,\ 1,\ \ldots,\ R-1$ do
  solve the Riemann problem with data $\hat{q}_{m+1}^N$ and q_j^0 to compute $\mathcal{A}^-\Delta q$ and $\mathcal{A}^+\Delta q$,
  update $q_j^1 \ := \ q_j^1 + \frac{\hat{k}}{h}(\mathcal{A}^-\Delta q + \mathcal{A}^+\Delta q)$.

For the case of a conservation law, this will restore conservation and in fact agrees with the flux function modification (4) if numerical fluxes are defined by (13). Note that in view of (12), in the conservation law case the coarse grid value q_j^1 is simply updated by the entire flux difference $f(\hat{q}_{m+1}^N) - f(q_j^0)$ and we wouldn't need to actually solve the Riemann problem. However, implementing it as presented above gives a uniform and general formulation.

The above idea extends directly to two space dimensions (and also to three dimensions). The multi-dimensional wave-propagation algorithm consists of solving one-dimensional Riemann problems normal to each cell interface. This defines waves and fluctuations exactly as in one dimension. These are used as in the one-dimensional algorithm, and are also used to define "transverse corrections" by essentially solving a Riemann problem in the transverse direction using the fluctuations as data. These corrections are needed to give second order accuracy in multi-dimensions as well as to increase the stability limit to allow Courant numbers close to 1. The corrections are fully described in [14]. For our present purposes we need only note that these corrections modify the correction fluxes $\tilde{F}$ (and corresponding y-fluxes $\tilde{G}$) and are in flux-differencing form, so that they are automatically corrected at grid interfaces in the step (4) where corrections are made due to the fluxes. Numerical experiments demonstrate full second order accuracy for both conservation laws and nonconservative equations, when the above approach is used (along with the second-order correction terms discussed in Section 3).

5 Further extensions

The AMRCLAW software contains some further extensions of the wave-propagation algorithms discussed in [14] and implemented in CLAWPACK. These will be only briefly described here, with emphasis on new issues that arise in connection with mesh refinement.

5.1 Source terms

Consider the hyperbolic equation

$$q_t + Aq_x = \psi(q, x, t).$$

The source terms ψ can be handled using a standard fractional step method, also called a "splitting method". In this approach, we alternate between solving the homogeneous hyperbolic equation, ignoring the source term, and solving the ODE $q_t = \psi$. In the context of AMR, we need to apply this sequence within each time step on each grid. We must implement this carefully in order to avoid generating excessive noise at the interface which can contaminate the solution. Whenever we solve a Riemann problem we must ensure that the data on each side contains the same total contribution from source terms in order to avoid the generation of spurious waves. This must be carefully observed in computing ghost cell values for the fine grid at intermediate times, and also in solving the additional Riemann problem need for the conservative fix described in Section 4. Recall that this Riemann problem is between the ghost cell values at each intermediate time and the coarse grid value at the original time, but when source terms are included this coarse grid value must be modified to incorporate the correct source contributions.

These interface details will be presented below. First we discuss the basic fractional step method that we use on each grid. In each time step we first solve the homogeneous hyperbolic equation over time Δt and then use the resulting solution as initial data for the source-term equation over time Δt. This is a so-called "first-order" splitting (or "Godunov splitting"), as opposed to the "Strang splitting" in which one advances first by a half time step on one equation, then by a full time step on the other equation, and ends with a half time step on the first equation again[17], [18]. Formally the Strang splitting can give second order accuracy in situations where only first order accuracy is achieved with our choice, but in practical problems where methods of this nature are useful, the differences in resolution actually seen are generally negligible. This is because applying the Strang splitting over N time steps is equivalent to starting with a half time step with one operator, then alternating with Δt steps with each until the N'th step, where we finish with a half step of the first operator again. The change made by this minor modification of shifting one half a time step from the beginning to the end of the computation can be formally $O(\Delta t)$, and hence reduce the global accuracy to first order, but clearly will not degrade the overall resolution of the solution to any degree. The solution may simply be shifted by $O(\Delta t)$ relative to its correct location, for example. This is very different from the sort of errors introduced by a genuine first order method, *e.g.*, by using a first order upwind method in place of the high-resolution flux-limiter method. See [14] for more discussion of this point.

Using the simpler splitting has a number of advantages in the context of coupling source terms with the hyperbolic solver, particularly with AMR. For one thing it is less expensive, since the source terms are advanced only once per time step instead of twice. (Of course one could combine the half time steps together in the Strang splitting as alluded to above, but this is impractical when variable time steps are used, particularly with adaptive refinement.) The specification of boundary conditions is also simplified. The user-supplied routine that extends values from the computational grid to the ghost cells is called at the beginning of the time step, producing values in the ghost cells that guarantee the physical boundary conditions will be satisfied. For example, the normal momentum must be negated at a solid wall boundary for the Euler equations. Since the hyperbolic equation is advanced first, these conditions are used immediately. Then the source terms are advanced, typically a local ODE solve in each grid cell which does not require using ghost cell values. We do not need to worry about solving the ODE's in ghost cells since these values are replaced immediately at the start of the next time step.

Applying the hyperbolic solver first also simplifies the modifications needed at the grid interfaces to minimize noise generation, and the algorithm is a fairly simple extension of what has been presented already. Let $\mathcal{H}(k)$ represent the solution operator for the hyperbolic equation and $\mathcal{S}(k)$ the solution operator for the source terms. Then the fractional step method over one time step on a single grid takes the form

$$\begin{aligned} q^* &= \mathcal{H}(k)q^n \\ q^{n+1} &= \mathcal{S}(k)q^*. \end{aligned}$$

Now consider a refined grid, with refinement ratio R. As before, let $\hat{q}$ represent the solution on the fine grid, where the time step is $\hat{k}$, and let q be the coarse grid solution with time step $k = R\hat{k}$. The algorithm is then:

```
# Coarse grid update:
q* = H(k)q^0
q^1 = S(k)q*

# Initialize coarse grid value needed for conservation fix-up:
q_j^{0,0} = q_j^0

# Advance fine grids:
for N = 0, 1, ..., R-1 do
    Space-time interpolate the ghost cell values q̂^N_{m+1}, q̂^N_{m+2} using q^0 and q^1
    q̂* = H(k̂)q̂^N
    q̂^{N+1} = S(k̂)q̂*

    # Conservation fix-up:
    Solve the Riemann problem with data q̂^N_{m+1} and q_j^{0,N} to compute A^-Δq and A^+Δq
    q_j^1 := q_j^1 + (k̂/h)(A^-Δq + A^+Δq)
    # Apply source terms to coarse grid value:
    q_j^{0,N+1} = S(k̂)q_j^{0,N}
    end
```

This is basically a direct extension of the algorithm described previously except for the final step, which ensures that the Riemann problem solved in the correction phase has consistent data at each fine time step.

Numerical tests have shown that this approach performs quite well. Attempts with other styles of splitting, *e.g.*, the Strang splitting, or the Godunov splitting with the order of $\mathcal{H}$ and $\mathcal{S}$ reversed, were much less successful. To gain a better appreciation of how mismatches at the interface can generate noise, the reader is encouraged to explore various strategies on the simple scalar equation

$$q_t + uq_x = q \tag{16}$$

with constant advection speed u and data q that is initially constant in space. If a numerical ODE method is used to solve $q_t = q$ (rather than using the trivial exact solution operator), then any approach will lead to the generation of noise at the interface simply because this ODE will be solved more accurately on the fine grid than on the coarse grid, leading to jumps in q at the interface and hence to waves propagating at speed u. The approach outlined above will only generate $O(k^2)$ noise, as long as the ODE solver is at least first order accurate, whereas any other approach considered would generate $O(k)$ noise.

5.2 Capacity-form differencing

The wave-propagation algorithm in [14] is also described in a more general form that applies to a quasilinear equation of the form

$$\kappa(x)q_t + Aq_x = 0$$

in one dimension, with obvious generalization to more dimensions. Here the function $\kappa(x)$ is called the "capacity function" since it represents in some way the capacity of the medium to hold the quantity q. Working in this form is particularly useful for equations in the conservation form

$$\kappa(x)q_t + f(q)_x = 0$$

where it is $\kappa(x)q(x,t)$ that is the conserved quantity rather than q alone, while the flux is defined in terms of q. As a one-dimensional example consider flow through a variable-area duct where $\kappa(x)$ is the cross-sectional area and q represents concentration per unit volume. In this case $\kappa_i h$ is the volume of the i'th grid cell. Flow in porous media is another example, where κ represents the porosity in one or more dimensions. This form also arises in using mapped curvilinear grids, in which case κ is the Jacobian of the grid transformation. More details are presented in [14].

The idea of capacity-form differencing is to replace the update formula (11) by

$$q_i^{n+1} = q_i^n - \frac{k}{\kappa_i h}(\mathcal{A}^+\Delta q_i + \mathcal{A}^-\Delta q_{i-1}) - \frac{k}{\kappa_i h}(\tilde{F}_{i+1} - \tilde{F}_i).$$

In the case of a conservation law, assuming (12) holds, this guarantees conservation of $h\sum_i \kappa_i q_i$. In the definition of $\tilde{F}_i$, we must also incorporate κ_i into the second order corrections, replacing (10) by

$$\tilde{F}_i = \frac{1}{2}\sum_{p=1}^{m} |\lambda_i^p| \left(1 - \frac{k}{h\kappa_i}|\lambda_i^p|\right) \tilde{\mathcal{W}}_i^p \tag{17}$$

Using capacity-form differencing with AMR is direct, provided we insure that conservation is maintained in transferring information between grids. Returning to Figure 1(b), suppose we have capacities κ_i defined on the coarse grid and $\hat{\kappa}_i$ defined on the fine grid. In a region where the two grids overlap, we assume that these values are consistent. For example, in Figure 1(b) we should have

$$\kappa_{j-1} = \frac{1}{2}(\hat{\kappa}_{m-1} + \hat{\kappa}_m) \tag{18}$$

so that the capacity of this coarse grid cell agrees with the total capacity of the two fine cells: $h\kappa_{j-1} = \hat{h}(\hat{\kappa}_{m-1} + \hat{\kappa}_m)$.

When updating the coarse grid value q^1_{j-1} by the average of the fine grid values $\hat{q}^2_{m-1}$ and $\hat{q}^2_m$, we must weight by the capacity functions and replace (3) by

$$q^1_{j-1} := \frac{\hat{\kappa}_{m-1}\hat{q}^2_{m-1} + \hat{\kappa}_m \hat{q}^2_m}{\hat{\kappa}_{m-1} + \hat{\kappa}_m}.$$

Finally, in the grid-interface correction (15), we replace h in the denominator by $\kappa_j h$.

6 Numerical Results

We present AMR computations on some examples from [14] to illustrate that source terms, nonconservative hyperbolic systems, capacity form differencing, and curvilinear grids can all be successfully handled. Further examples can be found on the AMRCLAW webpage[6], including some animations.

Example 6.1. We repeat Example 3.8.3 from [14], which consists of the linear equations of acoustics (a hyperbolic system of 3 equations) with a discontinuity in the sound speed across a line oblique to the grid. A plane wave strikes the interface at some angle, leading to transmitted and reflected waves. The time-evolution is best seen in Figure 4.

The acoustics equations for the pressure perturbation p and velocities u and v can be written

$$q_t + Aq_x + Bq_y = 0 \tag{19}$$

where

$$q = \begin{bmatrix} p \\ u \\ v \end{bmatrix}, \qquad A = \begin{bmatrix} 0 & K & 0 \\ 1/\rho & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 & K \\ 0 & 0 & 0 \\ 1/\rho & 0 & 0 \end{bmatrix}.$$

The coefficients are the density $\rho(x, y)$ and bulk modulus of elasticity $K(x, y)$. In the example ρ has a discontinuity across the interface while K is constant. The Riemann solvers for this system in the wave-propagation form are given in [14].

Figure 4(a) shows a contour plot of the initial pressure, a cosine hump as in [14] moving towards the upper right. The dashed line shows the location of the discontinuity in sound speed. The heterogeneous material is described by a density and bulk modulus of elasticity, and here the bulk

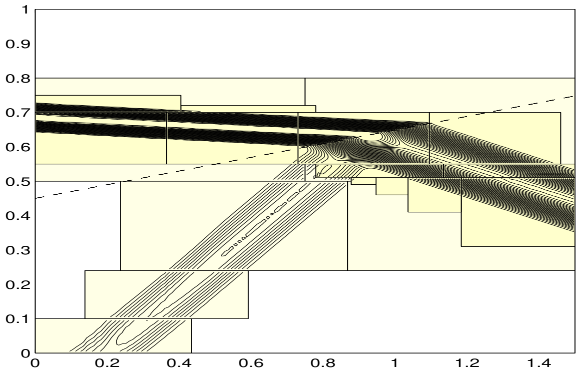


Figure 3: *Pressure contours for a plane wave hitting a discontinuity in sound speed in the acoustics equations, at time $t = 0.6$. The finer grids are shaded, with darker shading indicating finer levels. Three levels are used with a 76×50 grid at the coarsest level and refinement by a factor of 2 in each level.*

modulus is taken to be constant while the density is discontinuous, leading to the discontinuity in sound speed. In each grid cell the density is defined as the cell average of the true density over that cell. In [14] it is shown that the wave-propagation algorithm handles this problem well even when the discontinuity in density is not aligned with the grid. Figure 3 shows an AMRCLAW calculation where the coarsest grid is 76×50 and two levels of refinement are used with $R = 2$ in each, so that the finest grid compares with the resolution seen in the lower plots of Figure 8 of [14]. The tolerance used here was chosen in such a way that the weaker reflected wave has only been refined to Level 2. A smaller tolerance would cause Level 3 refinement of this wave as well. Here we have used a ratio of 2 for refinement in each level for demonstration purposes.

Example 6.2. The previous example does not fully test the new interface conditions between the fine and coarse grids in the nonconservative case. These acoustics equations fail to be in conservation form only along the interface where the density is discontinuous, and the wave stays embedded in Level 3 grids as it moves along this interface. As a more severe test we repeated this computation with a simple change in the error estimation procedure so that points are flagged for refinement only if $x < 0.6$. For $x > 0.6$ there is only the coarsest grid, so the wave moves from the initial fine grids onto the coarse grid as time advances. Figure 4 shows a sequence of times ending with the time shown in Figure 3(b). Some smearing of the wave is seen on the coarser grid, which is inevitable, but no difficulties are observed along the discontinuity in density.

Example 6.3. We repeat Example 3.10.1 from [14]. This is the advection equation

$$\rho q_t + (\rho u q)_x + (\rho v q)_y = 0. \tag{20}$$

for a tracer $q(x, t)$ in a density-stratified flow over a hump. We take $u(x, y),\ v(x, y))$ to be a fixed

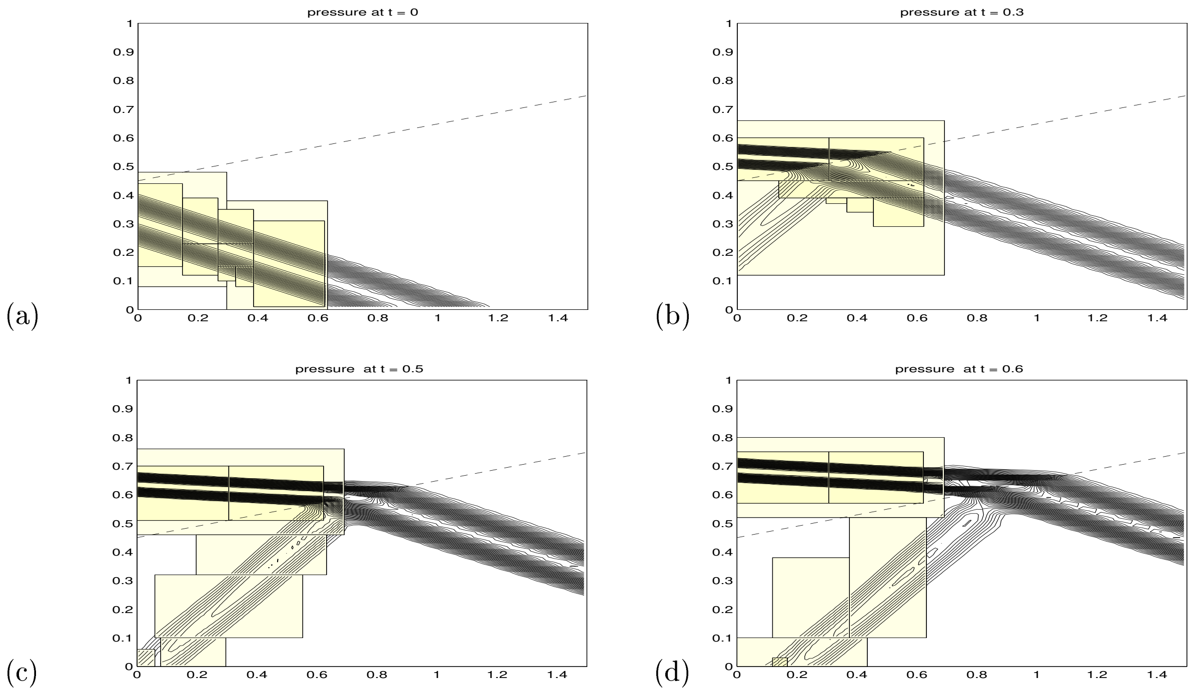


Figure 4: *Pressure contours for a plane wave hitting a discontinuity in sound speed in the acoustics equations, at four different times. In this test refinement was allowed only for* $x < 0.6$ *so that the wave moves out of the refined region.*

velocity field, chosen so that

$$(\rho u)_x + (\rho v)_y = 0 \tag{21}$$

while ρq is the conserved quantity.

In the test problem we consider flow over a hump with the bottom topography given by

$$B(x) = \frac{\alpha}{1+\beta x^2} \tag{22}$$

in the domain $-1 \le x \le 1$, $B(x) \le y \le 1$ (with $\alpha < 1$). The velocity field is chosen by using the "stream function"

$$\psi(x,y) = \frac{y - B(x)}{1 - B(x)}$$

to define

$$\begin{aligned} \rho u &= \psi_y = \frac{1}{1-B(x)}, \\ \rho v &= -\psi_x = \frac{B'(x)(1-y)}{(1-B(x))^2}, \end{aligned}$$

so that condition (21) is satisfied. Dividing by ρ gives the velocity field. Note that (u,v) is not divergence free and ψ is not a stream function for this velocity, though it is true that contours of constant ψ give streamlines of the flow. We use the density profile

$$\rho(x,y) = \rho(y) = e^{-\gamma y} \qquad \text{for some } \gamma, \tag{23}$$

as in [14], so that the velocities increase exponentially with y.

In the test below we use $\alpha = 0.6$, $\beta = 10$, and $\gamma = 2.5$. Figure 5 shows the initial data and exact solution at time $t = 0.18$ for data consisting of a circular blob of tracer:

$$q(x,y,0) = \begin{cases} 1 & \text{if } (x+0.75)^2 + (y-0.5)^2 < (0.2)^2 \\ 0 & \text{otherwise} \end{cases}$$

The problem is solved on a curvilinear grid. The irregular region of the x-y plane that can be mapped smoothly to a rectangle. Then equation (20) can be transformed to an advection equation on the rectangle and solved on a uniform Cartesian grid in this computational $\xi - \eta$ space. Here we use "Grid 2" of [14], with the mapping

$$X(\xi,\eta) = \xi, \qquad Y(\xi,\eta) = B(\xi) + \eta(1 - B(\xi)).$$

Capacity-form differencing is used where $\kappa = \rho J$ with J being the Jacobian of the grid mapping, as explained in [14]. Figure 6 shows computed results with 3 levels of refinement and refinement ratio 2 in each case. The finest grid has the same resolution as the calculation shown in Figure 10 in [14].

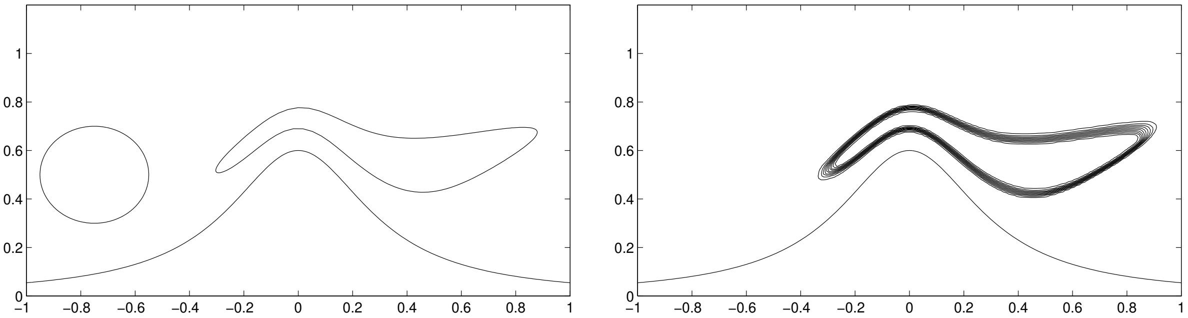


Figure 5: *Density-stratified flow over a hump. Left: initial data is 1 inside the circular region and 0 elsewhere. The region where the solution is 1 at time* $t = 0.18$ *is also shown. Right: Computed results on a* 200×100 *Cartesian grid. (From [14])*

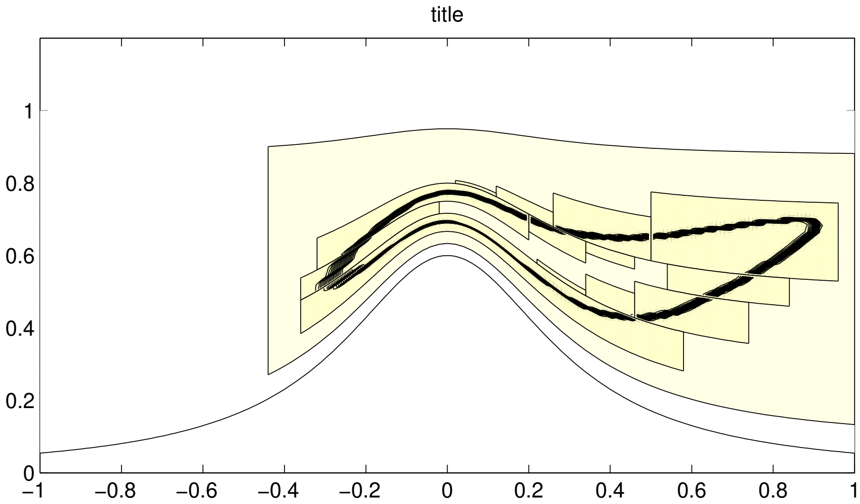
7 Conclusions

An adaptive mesh refinement algorithm developed for the Euler equations of gas dynamics has been extended to employ high-resolution wave-propagation algorithms in a more general framework. In particular, we have discussed the modifications needed to allow the application of this method to hyperbolic problems which are not in conservation form, following the approach of [14]. This has been done in a way that still maintains conservation when applied to a conservation law, in spite of the fact that a wave-propagation approach is used to update cell averages rather than standard flux-differencing. A generalization of this has also been presented for fractional step methods on hyperbolic equations with source terms.

We have also discussed extensions to capacity-form differencing. This formulation is useful in applying the methods on curvilinear grids, where refinement is done on logically rectangular patches. This is illustrated in Section 6 for one example with the advection equations. In the future we intend to explore the use of curvilinear grids more fully for nonlinear conservation laws such as the Euler equations, since body-fitted curvilinear grids are crucial for some applications.

Adaptive mesh refinement is even more crucial in solving three-dimensional problems than in two space dimensions. Recently CLAWPACK has been extended to three dimensions [9], [10] and work is underway to extend the AMRCLAW version. The underlying AMR code for the Euler equations has previously been successfully extended to three dimensions (*e.g.*, [1], [20]) and the ideas presented in this paper carry over easily as well.

(a)



(b)

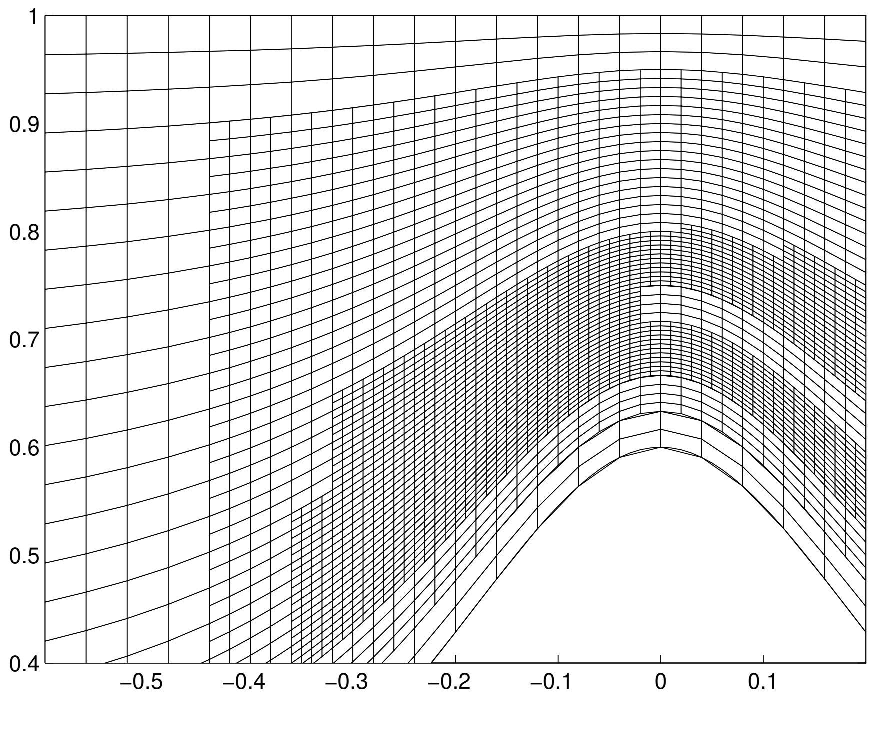


Figure 6: *(a) Computed density at time* $t = 0.18$ *for stratified flow over a hump. Contour levels are at* $0.05, 0.1, \ \ldots, \ 0.95$. *Compare to Figure 10 in [14]. (b) A blow-up of the grids near the top of the hump.*

Acknowledgments.

We are indebted to Smadar Karni for pointing out the difficulty of maintaining conservation at grid interfaces with the wave-propagation framework. This led to the development of some of the ideas presented here.

This work was supported in part by NSF Grants DMS-9505021, DMS-96226645, and DOE Grants DE-FG03-96ER25292, DE-FG02-88ER25053, DE-FG02-92ER25139, and AFOSR grant F49620-94-0132.

References

[1] J. Bell, M. J. Berger, J. Saltzman, and M. Welcome, *Three dimensional adaptive mesh refinement for hyperbolic conservation laws*, SIAM J. Sci. Stat., 15 (1994), pp. 127–138.

[2] M. Berger, *Adaptive mesh refinement for hyperbolic partial differential equations*, PhD thesis, Computer Science Deptartment, Stanford University, 1982.

[3] M. Berger and A. Jameson, *Automatic adaptive grid refinement for the Euler equations*, AIAA J., 23 (1985), pp. 561–568.

[4] M. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[5] M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84.

[6] M. J. Berger and R. J. LeVeque, AMRCLAW *software.* A test version is available at `http://www.amath.washington.edu/~rjl/amrclaw/`.

[7] D. De Zeeuw and K. Powell, *An adaptively-refined Cartesian mesh solver for the Euler equations.* AIAA Paper 91-1542, 1991.

[8] S. K. Godunov, Mat. Sb., 47 (1959), p. 271.

[9] J. O. Langseth and R. J. LeVeque, *Three-dimensional Euler computations using* CLAWPACK, in Conf. on Numer. Meth. for Euler and Navier-Stokes Eq., Montreal, P. Arminjon, ed., 1995. to appear.

[10] J. O. Langseth and R. J. LeVeque, *A wave-propagation method for three-dimensional hyperbolic conservation laws.* preprint, `ftp://amath.washington.edu/pub/rjl/papers/jol-rjl:claw3d.ps.Z`, 1997.

[11] R. J. LeVeque, CLAWPACK *software.* available from `netlib.att.com` in `netlib/pdes/claw` or on the Web at the URL
`http://www.amath.washington.edu/~rjl/clawpack.html`.

[12] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser-Verlag, 1990.

[13] R. J. LeVeque, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM J. Numer. Anal., 33 (1996), pp. 627–665.

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

[15] K. Powell, *Solution of the Euler and Magnetohydrodynamic Equations on Solution-Adaptive Cartesian Grids.* Von Karman Institute for Fluid Dynamics Lecture Series, 1996.

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

[17] G. Strang, *On the construction and comparison of difference schemes*, SIAM J. Num. Anal., 5 (1968), pp. 506–517.

[18] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth & Brooks/Cole, 1989.

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

[20] R. Walder, *Some Aspects of the Computational Dynamics of Colliding Flows in Astrophysical Nebulae*, PhD thesis, Astronomy Institute, ETH-Zürich, No. 10302, 1993.