Adaptive Mesh Refinement for 1-D Hyperbolic PDEs

Saumya Sinha[*], Kenneth J. Roche[†]

March 8, 2015

Abstract: We describe an adaptive mesh refinement algorithm that extends high resolution wave-progpagation techniques to hyperbolic systems in non-conservative form. The method for keeping numerical conservation at grid cell interfaces is described. The algorithm was tested for simple 1D problems. Results are compared to static mesh solutions for the same problems computed with Clawpack[1].

Contents

[*]Department of Applied Mathematics, University of Washington, Seattle, Washington U.S.A. Email:saumya@uw.edu

[†]High Performance Computing Group, Pacific Northwest National Laboratory, 1 Battelle Road, Richland, Washington U.S.A. Email:kenneth.roche@pnnl.gov

List of Figures

List of Tables

1 Introduction

We describe an adaptive mesh refinement (AMR) strategy for the solution of hyperbolic systems of conservation laws. The solutions of these equations are often smooth and easily approximated over large portions of their domain, but contain locally isolated regions with steep gradients, shocks or discontinuities. Our AMR strategy models and tracks such regions, by using rectangular patches over Cartesian grids to refine both the space and time coordinates.

2 The AMR algorithm

In this paper, we consider only one-dimensional systems. The coarse grid consists of a collection of intervals or cells that cover the entire domain, while refinements cover a subset of the domain and have finer cell-widths. If the cell-width in a fine grid is refined by an even integer R_L, then the time-step is also refined by the same factor, so that the ratio $\Delta t/\Delta x$ remains unchanged and for every single time-step on the coarse grid, R_L time steps must be taken on the fine grid. An error estimation procedure based on Richardson extrapolation, described in more detail in section 2.2, is used to flag the cells which need refinement. Flagged cells are clustered into patches, and a fine grid is generated for each patch (including a buffer zone). A finite volume method in wave-propagation form is used to advance the solution on the coarse grid over a single time-step, and then the solution on the fine grid(s) is advanced over R_L time steps to catch-up. Every R time steps, the fine grids are regenerated, and the error estimation procedure is re-applied to capture the movement of the discontinuities. The refinement and solution process is repeated. Each of the above steps is described in more detail below.

2.1 Advance solution over coarse grid

After initializing the discrete coarse grid, the first step is to advance the solution over a single time-step on the coarse grid. A standard high resolution wave propagation method (with limiter corrections) is used to solve the Riemann problem.

Consider a linear system $q_t + A(x)q_x = 0$, where $q \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times m}$ is diagonalizable with real eigenvalues. On the cell $[x_{i-1/2}, x_{i+1/2}]$, the matrix $A(x)$ is approximated by a matrix A_{i-1} for cell $i-1$ with eigenvalues $\lambda^1_{i-1}, \ldots, \lambda^m_{i-1}$. Then for the Riemann problem at $x_{i-1/2}$, we decompose $q_i - q_{i-1}$ into waves $\mathcal{W}^p_i, p = 1, \ldots, m$ in such a way that the left-going waves are eigenvectors of the matrix A_{i-1}, traveling with speeds $\lambda^p_{i-1} < 0$, while the right-going waves are eigenvectors of A_i traveling with speeds $\lambda^p_i > 0$. The update formula is then

$$q^{n+1}_i = q^n_i - \frac{k}{h}\sum_{p=1}^{m}\left[(\lambda^p_i)^+\mathcal{W}^p_i + (\lambda^p_{i+1})^-\mathcal{W}^p_{i+1}\right] - \frac{k}{h}\left(\tilde{F}_{i+1} - \tilde{F}_i\right) \tag{1}$$

where k is the length of the time-step and h is the cell-width in the spatial grid, and for any real number a, $a^+ = \max\{a, 0\}$ and $a^- = \max\{-a, 0\}$.

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

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

where $\tilde{W}^p_{i+1}$ is some limited version of the wave $\mathcal{W}^p_i$. When $\tilde{W}_i = W_i$, the method is reduced to upwind + second order correction, i.e. Lax-Wendroff method.

This method is extended to nonlinear systems using the Roe approximate Riemann solver, which linearizes the problem at each cell-interface in such a way that conservation is guaranteed. This method is symbolically written as

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

where $\mathcal{A}^+\Delta(q_i)$ represents the right-going fluctuation from the left interface of cell i, while $\mathcal{A}^-\Delta(q_i)$ is the left-going fluctuation from the right interface of cell i.

2.2 determining refinement - error estimation

The idea behind the refinement step comes from first determining where a refinement may be useful. If the solution $q(x,t)$ is smooth enough, we assume that method D (application of eqn(1)) has accuracy $O(s)$ in both space and time. The local truncation error is $q(x,t+k) - Dq(x,t) = \tau(x,t) + kO(k^{s+1} + h^{s+1}))$ where τ is the leading error. Taking two time steps suggests $q(x,t+2k) - D^2q(x,t) = 2\tau(x,t) + kO(k^{s+1} + h^{s+1}))$. Now, coarsen the grid to have space and time widths $2h$, $2k$ and let the differencing scheme be called D_{2h} in this case. The local truncation error in this case reads $q(x,t+2k) - D_{2h}q(x,t) = 2^{s+1}\tau(x,t) + O(h^{s+2})$. Thus, an error growth estimate at time t can be formed by taking the difference in the error taking two steps with the regular integration scheme, and a single large step using every other grid point:

$$\frac{D^2q(x,t) - D_{2h}q(x,t)}{2^{s+1} - 2} = \tau(x,t) + O(h^{s+2})\,. \tag{3}$$

The difference between the solutions obtained on the two grids at each point is proportional to the local truncation error at that point. At coarse cells where the difference between the two sets of values exceed some tolerance, all four cells contained in the real grid are flagged as requiring refinement. See [3] for more details. *Clustering* is simply to fill gaps in neighboring cells flagged for refinement. Should there be a cell not flagged for refinement between cells that require refinement, the cell in between will be refined as well because in the next step it is likely refinement will be required. A buffer zone around the flagged cells is added to track features of interest in the refined region. This buffer zone must be affiliated with the number of time steps evaluated at a particular level. If a patch at level L is refined by even integer R_L, then the time-step is equally refined implying that R_L steps must be taken on the refined grid at level $L+1$ for each step on grid at level L.

2.3 assigning values to new fine grid points

Once regions where a fine grid are determined, values must be assigned to the fine grid points. Current time fine grid values are assigned by evaluating a linear interpolation akin to flux limiting between coarse cell values in neighboring cells. Intermediate time values on the fine grid are assigned by evaluating a bilinear space-time interpolation based on coarse grid values between neighboring cells at the current and subsequent time. This approach is needed because more time steps are taken on the fine grid than the coarse grid and there are no coarse grid values available at intermediate times.

2.4 flux updates on refined and coarse meshes

After recursively creating the fine grid, how to evolve the newly refined grid needs to be considered. To evolve, conservation of fluxes at grid interfaces is required. We refer to Fig 1 for this discussion. There are essentially three cases to consider: coarse cell to coarse cell, fine cell to fine cell, and overlapping coarse cell and fine cell regions. Let F_i be the flux at the left edge of coarse cell i. This is different notation than what we typically wrote in class, $F_{i-\frac{1}{2}}$.

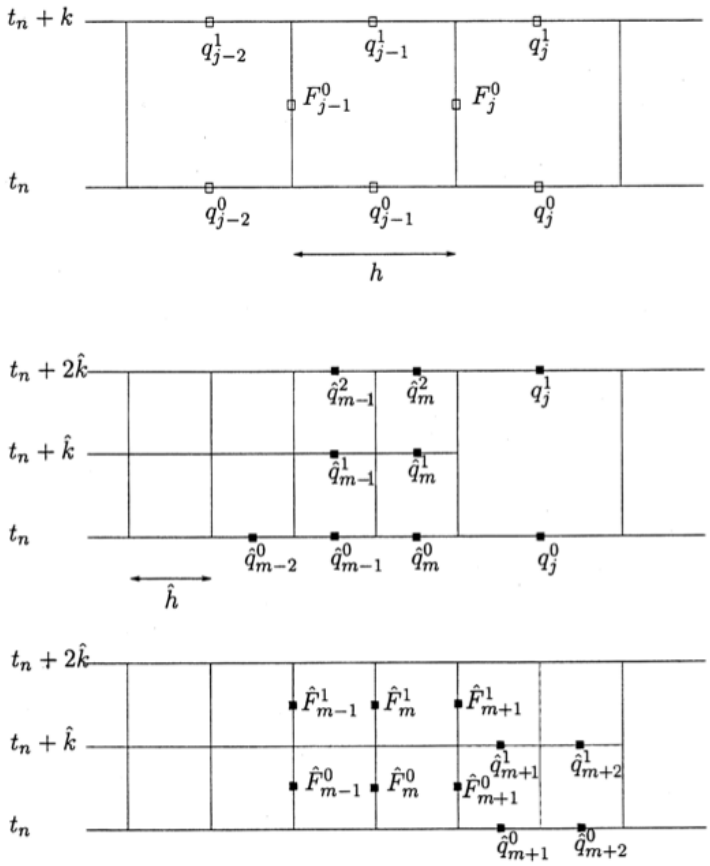

Figure 1: Essential figure for flux adapting Cartesian grids. The top grid is the normal coarse grid we are used to using with fluxes indicated. The middle diagram shows the case where a fine grid overlays a coarse grid and the interface region of fine and coarse grids. Recall that $2\hat{h} = h$ and $2\hat{k} = k$. The bottom diagram depicts the ghost cell region imposed on the coarse cell interface as well as the fine grid fluxes needed to bridge the interface.

Updating between coarse grid cells is, to lowest order, simply:

$$q_i^1 = q_i^0 - \frac{k}{h}(F_{i+1}^0 - F_i^0) \tag{4}$$

where h, k are the space and time lattice spacing respectively.

In regions that contain m adjacent single level refined cells let the lattice spacing be denoted

by $\hat{h}, \hat{k}$. We have, in the stated order, $\forall i = 1, m$

$$\begin{aligned} \hat{q}_i^1 &= \hat{q}_i^0 - \frac{\hat{k}}{\hat{h}}(\hat{F}_{i+1}^0 - \hat{F}_i^0) \\ \hat{q}_i^2 &= \hat{q}_i^1 - \frac{\hat{k}}{\hat{h}}(\hat{F}_{i+1}^1 - \hat{F}_i^1) \end{aligned} \tag{5}$$

in the refined cells where we assume each refinement is $2X$ as in the figure.

In regions where the coarse and fine grids overlap, care must be taken. Two ghost cells are introduced, $\hat{q}_{m+1}, \hat{q}_{m+2}$. The values in these cells are decided by evaluating a polynomial at these fine grid locations determined by interpolating coarse grid values (as stated earlier). To complete the time step, q_i^1 is replaced by the average of the fine grid values in the overlapped region:

$$q_{j-1}^1 = \frac{1}{2}(\hat{q}_{m+1}^2 + \hat{q}_{m+2}^2) \tag{6}$$

and the corresponding left edge flux is similarly modified so that q_i^1 is finally modified as:

$$q_i^1 \to q_i^1 + \frac{k}{h}(\frac{1}{2}(\hat{F}_{m+1}^0 + \hat{F}_{m+1}^1) - F_i^0)\,. \tag{7}$$

2.5 updating grid interfaces with wave propagation -correction step

With flux-differencing to ensure conservation, a new coarse grid value is updated by an average of fine-grid values in any cell covered by a fine grid. The key to conservation is that flux into coarse cells is accounted for by flux out of adjacent fine cells, and vice versa, as described in the previous section. Wave propagation algorithms are written in a manner that they can be extended to hyperbolic problems not in conservation form retaining wave features but with no flux function. For wave-propagation algorithms, a similar correction is needed for the waves to yield conservation. The wave-propagation formis used to update the values at the interface between fine and coarse grid cells on each grid independently using ghost-cell values as needed near grid interfaces.

Recall from class that $A^+\Delta q_{i-\frac{1}{2}}$ measures the net effect of all right going waves present from $x_{i-\frac{1}{2}}$, and $A^-\Delta q_{i-\frac{1}{2}}$ measures the net effect of all left going waves from the same interface. We call these *fluctuations*. Whether conservative or non-conservative systems are treated, second-order corrections are still written as flux-differences. However, the first-order upwind terms written in terms of the fluctuations must be handled differently. To maintain conservation we must *first* solve the Riemann problem between the ghost cell value $\hat{q}_{m+1}^0$ on the fine grid and the coarse grid value q_j^0, and add to the coarse cell value q_j^1 the resulting total fluctuation (dropped indices) $A^-\Delta q + A^+\Delta q$ weighted by $\hat{k}/h = 1/2$ since the time step is $\hat{k}$ while the cell size is h. *Second* we must also solve a Riemann problem on the fine grid between $\hat{q}_{m+1}^1$ and q_j^0 and add these fluctuations into q_j^1.

The correction step extends to an arbitrary hyperbolic system (i.e. see 3.2 for linear hyperbolic system of PDEs) where we use a Riemann solver to obtain $A^-\Delta q$ and $A^+\Delta q$ from the two states $\hat{q}_{m+1}^0$ and q_j^0. This step is made in each of the R time steps on the refined grid within the single coarse-grid step, where R is the refinement ratio.

In summary, the correction algorithm reads:

for $N = 0, R-1$

solve the Riemann problem with data $\hat{q}^N_{m+1}$ and q^0_j to compute $A^-\Delta q$ and $A^+\Delta q$

update $q^1_j \to q^1_j + \frac{\hat{k}}{h}\ (A^-\Delta q + A^+\Delta q)$

2.6 Note on boundary conditions

Steps for the AMR algorithm:

(a) initialize coarse grid

(b) evolve time on coarse grid

(c) *recurse*: estimate error, create fine grid

- cell flagging
- clustering
- fine grid generation

(d) solve Riemann on refined grid levels

(e) correction step to ensure conservation

(f) go to step b

3 Examples of 1D AMR

3.1 variable coefficient

Here we state a 1D scalar problem based on a variable coefficient velocity term.The *color* equation is the problem formed by solving the following Riemann problem at the interface $x_{i-\frac{1}{2}}$ between cells $i-1$ and i:

$$q(x,t)_t + u(x)q(x,t)_x = 0 \tag{8}$$

The method reads, i.e. for the *color* eqn (see 3.1):

$$q_i^{n+1} = q_i^n - \frac{k}{h}(u_i^+(q_i^n - q_{i-1}^n) + u_i^-(q_{i+1}^n - q_i^n)) - \frac{k}{h}(\tilde{F_{i+1}} - \tilde{F_i}) \tag{9}$$

where

$$\tilde{F_i} = \frac{1}{2}|u_i|(1 - \frac{k}{h}|u_i|)\tilde{W_i} \tag{10}$$

and the limiter is applied to wave $W_i = q_i - q_{i-1}$ i.e. traveling at speed $u(x)$, such that

$$\tilde{W_i} = \begin{cases} limiter(W_i, W_{i-1}), & u_i > 0 \\ limiter(W_i, W_{i+1}), & u_i < 0 \end{cases} \tag{11}$$

$$q(x,0) := \phi(x) = \begin{cases} q_{i-1}, & x < x_{i-\frac{1}{2}} \\ q_i, & x > x_{i-\frac{1}{2}} \end{cases} \quad , \quad u(x) = \begin{cases} u_{i-1}, & x < x_{i-\frac{1}{2}} \\ u_i, & x > x_{i-\frac{1}{2}} \end{cases} \tag{12}$$

and the sign of $u(x)$ arbitrary for now.

3.2 linear system (acoustics)

We study here the 1D systems of the form

$$q(x,t)_t + Aq(x,t)_x = 0 \tag{13}$$

and A satisfies hyperbolicity conditions.

4 Results

4.1 Software implementation

Clawpack was used to generate numerical Riemann solutions to our test problems for the fixed mesh case. We utilized (Python, Matlab, C) for our AMR implementations. Please find our software here: (need software + examples ...) and use the `Readme.txt` file there to build and test the software. The test cases presented in this report are reproduced there.

4.2 Results of the Numerical experiments for the 1D cases

5 Summary

In this paper we described the adaptive mesh refinement algorithm presented in [4], and tested the algorithm for several simplistic 1D cases. Our results were compared against those from Clawpack. Our AMR implementations are available to anyone who wants them.

References

[1] http://www.clawpack.org

[2] R. Leveque Finite Volume Methods for Hyperbolic Problems Cambridge University Press (2002)

[3] M. Berger and P. Collela, Local Adaptive Mesh Refinement for Shock Hydrodynamics Journal of Computational Physics 82, 64-84 (1989)

[4] M. Berger and R. Leveque, Adaptive Mesh Refinement Using Wave-Propagation Algorithms for Hyperbolic Systems SIAM Journal of Numerical Analysis, 35(6):2298–2316, (1998)

[5] M. Berger, Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, *PhD Thesis*, Department of Computer Science, Stanford University, Stanford, CA 94305 (1982)