SOLVING HYPERBOLIC PDES USING CONSERVATIVE SUBDIVISION SCHEMES

MATS HOLMSTRÖM

Swedish Institute of Space Physics, Box 812, SE-98128 Kiruna, Sweden
E-mail: matsh@irf.se

Explicit methods for solving hyperbolic PDEs on conservative form are presented. The methods are finite volume versions of the finite difference methods presented in [1]. Wavelet based conservative subdivision schemes provide an adaptive representation of the solution by thresholding the coefficients of a wavelet representation. The subdivision scheme leads to simple relations between the cell averages and the wavelet coefficients. This provides a fast way to reconstruct any cell average, simplifying the implementation of high order methods. The automatic adaptivity leads to savings in computational time and memory requirement. The magnitude of the wavelet coefficients also provide information about the local regularity of the solution making it possible to use different numerical flux functions in different parts of the domain. As an example, the one-dimensional magnetohydrodynamic (MHD) equations are solved.

1 Introduction

One reason for using wavelet based methods when solving PDEs is that certain functions are well compressed in a wavelet basis. Assume that a time dependent hyperbolic PDEs on conservative form is discretized by N cell averages on a uniform grid. By compression is meant that the solution can be represented by N_s wavelet coefficients with an error proportional to ε, where $N_s \ll N$. Using a wavelet based method for solving PDEs we can speed up the computation of a solution by means of the sparse representation. The sparsity also saves memory.

If the PDE is on conservative form, and the numerical method on a uniform grid is conservative, it is natural that also the wavelet transform used should be conservative. The transform chosen in this paper is conservative subdivision. A property of this transform is that each wavelet coefficient is directly related to a cell average. This property makes the transformation between cell averages and wavelet coefficients very fast.

The sparse representation proposed in this paper automatically adapts to changes over time in the solution to the PDE. The accuracy of the solution is determined by the threshold parameter ε.

Another advantage of the sparse representation is that it allows for the construction of locally uniform grids, allowing for the use of centered methods and simplifying the implementation of higher order methods.

In a previous paper [1] by the author a sparse point representation for finite difference schemes was introduced this work presents similar ideas in a finite volume setting.

2 Conservative Subdivision

A quick introduction to conservative subdivision, following Donoho [2]. This is sometimes called average-interpolating subdivision but the former name seems more informative. We define cell averages of a one-dimensional function $u(x)$ as

$$u_{j,k} = \frac{1}{\Delta x} \int_{x_{j,k-1/2}}^{x_{j,k+1/2}} u(x)\, dx.$$

Here j denotes scale and k position, i.e. $x_{j,k} = 2^{-j}k$. The cell size is $\Delta x = x_{j,k+1/2} - x_{j,k-1/2}$.

If we, given cell averages on a coarser scale $u_{j,k}$, want to find cell averages on a finer scale $u_{j+1,k}$ (half the cell size) we find the (unique) quadratic polynomial that has integrals $u_{j,k-1}$, $u_{j,k}$ and $u_{j,k+1}$; and then find $u_{j+1,2k}$ and $u_{j+1,2k+1}$ by integrating the quadratic.

By repeating this recursively we can get a representation on an arbitrarily fine scale. We can also use higher order, even, polynomials. Increasing the order of the polynomial will increase the work, but improve compression of smooth solutions. Boundaries are easily handled by the transform, we simply use the closest availible quadratic. The subdivision is very fast (a few arithmetic operation for splitting a cell in two).

If we want to do the opposite, go from a fine $u_{j+1,k}$ to a coarse grid $u_{j,k}$. We have no choise due to conservation,

$$u_{j,k} = \frac{1}{2}\left(u_{j+1,2k} + u_{j+1,2k+1}\right).$$

Now the difference between original cell averages $u_{j+1,k}$ and averages computed by subdivision from the coarser scale $u_{j,k}$ can be viewed as wavelet coefficients (or details)

$$d_{j,k} = \tilde{u}_{j+1,2k} - u_{j+1,2k}$$

where $\tilde{u}_{j+1,2k}$ is computed by subdivision.

3 Sparse Representation

To get a sparse representation we threshold the wavelet coefficients, i.e., we remove a coefficient if $|d_{j,k}| < \varepsilon$. In practice we start at a coarsest scale

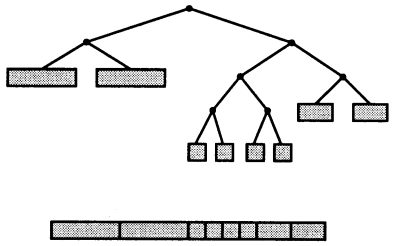

Figure 1. Sparse tree representation and corresponding grid.

and stop the transform when $|d_{j,k}| < \varepsilon$ for a cell. We then have a tree representation as shown in Fig. 1 (a binary tree with cells as leaves). When the solution evolves the representation can change by subdivision and coarsening of cells governed by the size of $|d_{j,k}|$. To allow for fine features to develop one can choose between adding an extra layer of cells (subdivide all cells) or refine a cell if $|d_{j,k}| > \varepsilon_2$. In this work the first approach was used.

The extension to higher-dimensional Cartesian grids is straight forward. In two-dimensions we subdivide along rows, then along columns and stop if the magnitude of all three wavelet coefficients $< \varepsilon$. This gives us a quad-tree. In three-dimensions we would get an oct-tree.

4 Time Evolution

We note that any cell average (on any level) can be found from the sparse representation by subdivision or coarsening. Methods for uniform grids can be used since we can always create a locally uniform grid, which simplifies using higher order methods.

We allow the solution to move, and features to develop by evolving the solution in time as follows:

1. Threshold the sparse representation.

2. Subdivide each cell one step.

3. Advance the solution in time by computing numerical fluxes at cell boundaries.

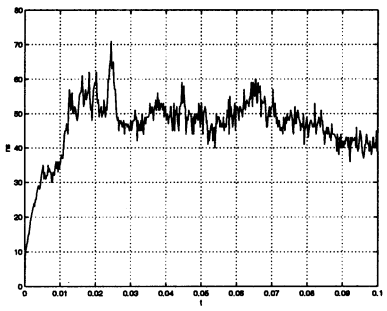

Figure 2. Number of upwind fluxes over time (out of 1024).

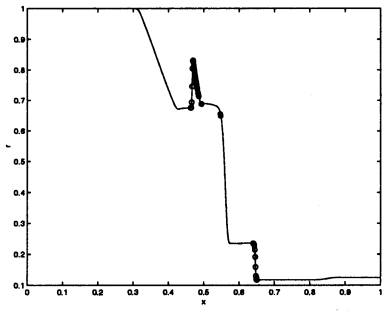

Figure 3. Plot of the density at $t = 0.1$. Positions of upwind fluxes marked.

Finally a comparison of the CPU time and error for the different methods is shown in Fig. 4. The error is at $t = 0.1$ in 1-norm for different thresholds on a grid with 1024 cells. We see that switch is fastest. This is the case since it is a small, one-dimensional problem, and what dominates the CPU time are the number of upwind fluxes due to the expensive eigen-decomposition. Tree will probably compare more favorably if we use more points on the finest grid (or go to higher dimensions).

7 Conclusions

We have presented a method to adaptively solve hyperbolic PDEs on conservative form. The sparse tree representation provides savings in computational time and memory. The transform used to construct the representation (Con-

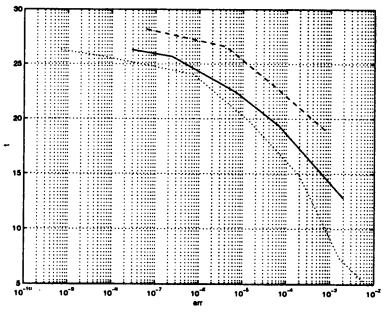

Figure 4. CPU time as a function of error at $t = 0.1$. Switch dotted, Tree & Switch solid, Tree dashed.

servative Subdivision) is fast. It is possible to use methods for uniform grids simplifying the use of higher order methods.

References

1. M. Holmström, Solving hyperbolic PDEs using interpolating wavelets. To appear in *SIAM J. Sci. Computing.*
2. D. L. Donoho, Smooth wavelet decompositions with blocky coefficient kernels. In L. L. Schumaker and G. Webb, eds., *Recent Advances in Wavelet Analysis*, pp. 259–308 (Academic Press, 1993).
3. M. Brio and C. C. Wu, An upwind differencing scheme for the equations of ideal magnetohydrodynamics. *J. Comput. Phys.* **75**, 400–422 (1988).
4. M. Gerritsen and P. Olsson, Designing an efficient solution strategy for fluid flows. II. Stable high-order central finite difference schemes on composite adaptive grids with sharp shock resolution. *J. Comput. Phys.* **147**, 293–317 (1998).
5. A. Harten, Adaptive multiresolution schemes for shock computations. *J. Comput. Phys.* **115**, 319–338 (1994).
6. B. Sjögreen, Numerical experiments with the multiresolution scheme for the compressible Euler equations. *J. Comput. Phys.* **117**, 251–261 (1995).