Accuracy, Adaptive Methods and Complex Geometry*

Marsha Berger
Courant Institute
251 Mercer Street
New York, NY 10012

Michael Aftosmis
USAF/NASA Ames
Moffett Field, CA 94035

John Melton
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

We describe our approach to computing accurate solutions for time dependent fluid flows in complex geometry. We use regular, embedded, Cartesian grids wherever possible. We describe our adaptive mesh refinement algorithm and our strategy for treating geometric complexity using non-body-fitted Cartesian grids. The trade-offs of block-structured vs. unstructured data structures are presented. Several open problems are discussed that need to be resolved both to automate and improve the accuracy of computations.

1 Introduction

Time dependent fluid flows can be quite complex. Some of the complexity is due to the nonlinear behavior of the fluid, and some is due to complexity of the geometry. Either source makes it difficult to simulate the behavior of the fluid in an automatic way.

In this paper we describe our approach towards alleviating these computational difficulties. Our point of view is that there are a number of advantages to working on regular grids. We describe two components of our work to illustrate this. The first component is the adaptive mesh refinement algorithm (AMR), which is based on the use of locally uniform grid patches superimposed on an underlying coarse grid. AMR is extremely beneficial for resolving the many scales in time dependent complex flows. The error estimator used to trigger mesh refinement is an important piece of this algorithm and will be discussed in detail.

The second element of our work is our approach to complicated geometries. We use regular, embedded, Cartesian grids to discretize the space surrounding solid objects. These objects are not generally aligned with the grid, and their boundaries may cut arbitrarily through the mesh. This type of representation greatly automates the volume grid generation, which would otherwise be difficult for extremely complicated domains. The use of adaptivity is critical in this approach. The issue of data structures also comes up in this context. We summarize with a wish list of remaining issues which will lead to robust and accurate solvers.

*Proc. First AFOSR Conference on Dynamic Motion CFD

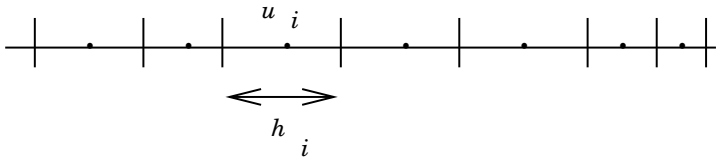
2 Accuracy: A Simple Example

As much as possible we prefer to work on uniform block structured grids because we believe they are more accurate, more efficient, and more flexible than alternatives. Numerical accuracy on irregular grids remains poorly understood, and we provide a simple example to illustrate this point. Efficiency and flexibility issues will be addressed after the description of the AMR algorithm.

Accuracy on regular and irregular grids can be compared in one dimension using a simple finite volume scheme. Consider the scalar advection equation $u_t + u_x = 0$, $0 \le x \le 1$, with periodic boundary conditions. The first order upwind scheme (Godunov's method) on a uniform grid is

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{h}(u_i^n - u_{i-1}^n). \tag{1}$$

Suppose we generate an irregular grid by taking random mesh widths $h_i \in [.9h, 1.1h]$, where h is the spacing used on the regular grid for comparison. The notation for the irregular grid is shown below.



Since the mesh width is no longer constant, eq. (1) is generalized to insure conservation with

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{h_i}(u_i^n - u_{i-1}^n) = Q\, u_i^n. \tag{2}$$

The local truncation error (LTE) is obtained by plugging in the exact solution, here denoted $u(x,t)$, into the difference scheme. We find that $u(x_i, t^{n+1}) - Qu(x_i, t^n) = \Delta t \cdot LTE$, where

$$LTE = u_x(\frac{h_i + h_{i-1}}{2h_i} - 1) + O(\Delta t + h). \tag{3}$$

If the mesh is smooth (i.e. $h_i = h_{i-1} + O(h^2)$), then the first term in parentheses is $O(h)$, and the method is consistent. However, for non-smooth meshes, the leading term is only $O(1)$. It would seem that the usual convergence analysis, where you take n steps making an error Δt each step, $n\Delta t =$ constant, would lead to $O(1)$ errors. Nevertheless, as has been shown by [24], the scheme does converge and manages to remain first order accurate. It is however less accurate than the corresponding scheme on a regular grid with the same number of points. Table 1 shows the results of a simple computational experiment with a Gaussian pulse initial condition integrated until time $t = 0.8$ with $\Delta t/h = .8$.

The results on the nonuniform grid show first order convergence, with the magnitude of the error approximately 50% higher than the uniform grid case. We also note that restricting cell size perturbations to within 10% of each other represents fairly mild variation within the grid. (This is especially so in higher dimensions where economy frequently demands increased stretching). The effect is more pronounced as the perturbation size increases. Table 2 shows a comparison with cell size variations up to 25% of the average size. To show that the loss in

N	Uniform Grid Errors			Non-uniform Grid ($Perturbation < .1h$)		
	L1	L2	L∞	L1	L2	L∞
20	13%	12%	13%	18%	15%	17%
40	7.2%	6.4%	7.2%	9.7%	8.6%	9.7%
80	3.7%	3.3%	3.8%	5.1%	4.5%	5.2%
160	1.9%	1.7%	1.9%	2.7%	2.4%	2.7%

Table 1: Percent relative errors using uniform grid and perturbed grid with perturbation $\leq 10\%$.

N	Uniform $1.25h$			N	Non-uniform Grid ($Perturbation < .25h$)		
	L1	L2	L∞		L1	L2	L∞
16	16%	14%	16%	20	24%	21%	23%
32	8.8%	7.8%	8.7%	40	13%	12 %	13%
64	4.6%	4.1%	4.6%	80	7.4%	6.5%	7.4%
128	2.4%	2.1%	2.4%	160	3.9%	3.5%	4.0%

Table 2: Comparison using uniform grid with larger mesh width, versus non-uniform grid with larger perturbations.

accuracy is not just due to the larger cell sizes allowed by the perturbation, table 2 also shows results computed on a regular grid but using the largest cell size allowed in the perturbed grid. Even with the larger cell size, the error is still smaller on the uniform grid, and with fewer points it is less expensive to compute as well.

For problems with discontinuous solutions, the question of order of accuracy is even harder to answer. Here too however we prefer uniform grids. There is evidence that a discontinuity has smaller phase errors on uniform grids than on irregular grids, since it can relax into a discrete traveling wave. See references [19],[25] for some examples of this.

3 Adaptive Mesh Refinement

AMR was originally developed for inviscid, compressible flow [8], [7]. It has been extended to solve Navier-Stokes equations, reacting flow computations, incompressible and low Mach number equations, phase-field models and more [3],[14],[4],[18]. It has been applied on mapped grids in both 2 and 3 dimensions [6],[21] where the refinement is still regular when viewed in the computational domain. In the cases it has been applied, it offers orders of magnitude savings in computational and storage costs over an equivalent uniformly refined grid.

The key element in our adaptive mesh refinement algorithm is the use of block-structured locally refined grid patches to increase the resolution of an underlying coarser grid only where needed. In this approach, certain grid cells at one refinement level are tagged as needing to be in a finer level grid. Cells are then organized into rectangular grid patches, typically containing several hundred to several thousand grid points per patch. Figure 1, taken from [23], illustrates this procedure on an example where tagged cells around a circle are organized into 8 refined grid patches.

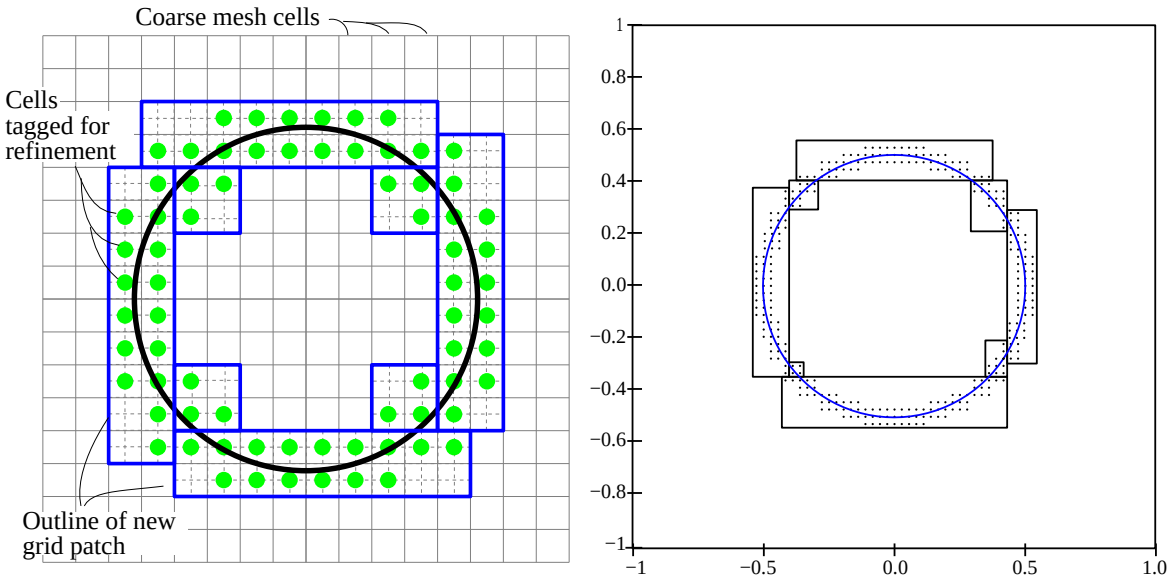


Figure 1: The schematic on the left shows grid cells tagged for refinement, with the locations of the new grid patches indicated. On the right is a sample calculation taken from [23].

Note that some cells not tagged for refinement are also included in new fine grid patches. Typically, 70% of the cells contained within grid patches are tagged; the remaining 30% are untagged but still lie within the new patch boundaries. Even with this additional 30% overhead, the total amount of storage used in this grid-based approach is less than that required by a cell-based (unstructured) data structure, where the grid connectivity and node locations frequently take on the order of 30 to 50 words per cell. With the block-structured approach, this information is stored on a per grid basis, using less than 20 words per grid. For a three dimensional calculation with 1000 grids, this overhead is still negligible. Thus, at its most basic level, AMR can be viewed as an efficient memory manager.

Two factors unique to transient calculations make the AMR approach especially robust. First, with time dependent calculations some sort of refinement history is generally needed so that when the phenomenon needing refinement moves on, the mesh can be easily de-refined without skewness problems. Unstructured methods using point insertion and deletion with re-triangulation have to be careful or the interpolations after each refinement pass can be expensive and diffusive. Both these problems are trivial with the block-structured approach. A second benefit of AMR's organization is that it easily allows subcycling in time, permitting different time steps to be taken on different grids. Finer grids can take a time step appropriate both for the local flow conditions and local cell size without imposing an unduly small timestep on the rest of the calculation.

A complete AMR algorithm contains the following components:

```
1. time step controller
2. inter-grid communication to
   a. provide boundary conditions for interior grids
   b. insure conservation at grid interfaces
   c. initialize the solution on new grid patches
3. error estimator - cell tagging strategy
```

4. grid generation/patching algorithm

There are still open questions remaining with this approach. For incompressible flow, how does one define a divergence free flow field when there is no global mesh defining the solution at intermediate times? The use of anisotropic refinement of the grid blocks is an interesting area that has not been explored. Deciding what criteria should be used to trigger the mesh refinement is also important, and is discussed in the following section.

3.1 Error Estimation

One of the least understood parts of any adaptive approach is the method used to decide when to refine the grid. While many *ad hoc* methods often work quite well, problems frequently require manual tuning. Unfortunately, a sound theoretical basis for mesh adaptation for flows with shocks does not yet exist.

We favor the use of local truncation error estimates as a basis for mesh refinement. There is some theoretical foundation for this approach, since convergence results using smooth model problems have shown rates that depend on the local truncation error. The recent work of [20] contains a convergence result with mesh refinement and re-gridding that demonstrates the dependence of global error on the local truncation error τ times a power of the mesh scale h. Thus, reducing h where τ is large helps control the global error.

Our implementation of a local truncation error estimator is quite inexpensive and typically accounts for less than a few percent of the total CPU time. The approach relies on the regularity of structured grids, and is explained below assuming the difference scheme has the same order of accuracy in space and time. (If this doesn't hold, the estimator can be generalized to measure each separately). In the example below, we illustrate the estimation of the local truncation error for solving the scalar advection equation $u_t + u_x = 0, 0 \le x \le 1$ using Lax-Wendroff,

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{2h}\left(u_{i+1}^n - u_{i-1}^n\right) + \frac{\Delta t^2}{2h^2}(u_{i+1}^n - 2 \cdot u_i^n + u_{i-1}^n) = Q_h u_i^n.$$

Here u_i^n represents the approximate solution at time $t = n \cdot \Delta t$ at the point $x_i = i \cdot h$. The exact solution will be denoted $u(x_i, t^n)$.

The local truncation error at time t_n is

$$u(x_i, t^{n+1}) - Q_h u(x_i, t^n) = \frac{\Delta t^3}{6} u_{ttt} + \frac{\Delta t h^2}{6} u_{xxx} + h.o.t. \tag{4}$$

Note that if $\Delta t = Const \cdot h$, then the errors in time and space are the same order. (The same holds true for second order upwind MUSCL-type schemes, but the form of the error is more complicated). If two time steps are taken then to leading order the error is doubled,

$$u(x_i, t^{n+2}) - Q_h Q_h u(x_i, t^n) = 2 \cdot (\frac{\Delta t^3}{6} u_{ttt} + \frac{\Delta t h^2}{6} u_{xxx}). \tag{5}$$

The truncation error can be estimated without directly computing approximations to the high order derivatives in (4), and in fact, without even knowing the exact form of the error. First, a temporary grid coarsened by two in each space dimension is created and initialized using cell-centered variables where $x_{i+1/2} = \frac{x_i + x_{i+1}}{2}$ and $\bar{u}_{i+1/2} = \frac{u_i + u_{i+1}}{2}$. Let Q_{2h} denote the

Lax-Wendroff operator on this grid with twice the mesh spacing. If the mesh ratio $\lambda = \frac{\Delta t}{h}$ stays constant, then the truncation error of Q_{2h} is

$$u(x_{i+1/2}, t^{n+2}) - Q_{2h}u(x_{i+1/2}, t^n) = 8 \cdot (\frac{\Delta t^3}{6}u_{ttt} + \frac{\Delta t h^2}{6}u_{xxx}) + h.o.t. \tag{6}$$

The comparison

$$\frac{Q_h^2 u(x_i, t^n) + Q_h^2 u(x_{i+1}, t^n)}{2} - Q_{2h}u(x_{i+1/2}, t^n) = 6 \cdot (\frac{\Delta t^3}{6}u_{ttt} + \frac{\Delta t h^2}{6}u_{xxx}) \tag{7}$$

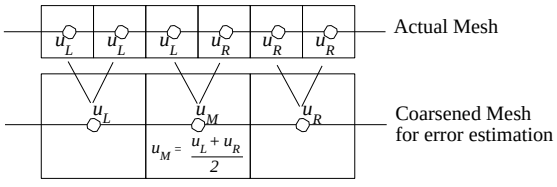
is proportional to the leading term of the local truncation error. In words, the solution on the fine grid is integrated twice and averaged, and then compared with one larger time step taken on the averaged (coarsened) grid.

Eqn. (6) is implemented by calling the same integrator to take one step on a coarser grid. Eqn. (5) is simply two steps of the usual scheme, applied to the usual grid. In our implementation of this error estimator, one of the steps is the usual one used in the integration of the pde; the second is thrown away. This makes the AMR program simpler, since for useful integration steps other work is also performed (for example, boundary values are saved for conservative matching with neighboring coarser grids, fine grids update coarser grids, etc.).

The reason this error estimator is inexpensive is that it is only applied on coarser level grids. The finest level grid is not allowed to have any further refinements, so we don't bother to estimate the error there. Since coarser grids are much less work than the finer grids, this adds little overhead. For example, for a typical refinement ratio of 4 in two dimensions, the coarser grid has 1/16 the number of grid points and takes 1/4 as many time steps. If half the domain were refined, the coarser level would take 1/32 the work of the finer level, or 3.2%. Suppose the error were estimated every other coarse time step. For every 2 integration steps on the coarse level, 1+1/4 wasted integration steps would be taken. The overhead would be $(5/8 \times 1/32) = 1.8\%$.

Even with the Richardson error estimates, more choices need to be made. Which variable should be examined in a system of equations? Should all variables use the same error threshold in deciding whether to refine the mesh? How should this threshold be chosen? Typically, a few time steps are taken to sample the magnitude of the error, and then the threshold can be more reliably set. If the boundary difference scheme is not the same order as the interior scheme, a separate procedure must be employed or the boundary cells will always be flagged.

This error estimation procedure assumes a smooth solution. To determine its behavior at discontinuities, we consider a piecewise constant solution that jumps from a left state u_L to u_R. Assume the discontinuity is aligned so that on the coarsened grid there is a middle state $u_M = (u_L + u_R)/2$.



The coarsened grid integration step yields $Q_{2h}u_M = u_M + \frac{\lambda}{2}(u_L - u_R)$. Two steps on the fine grid for the cells adjacent to the jump yield

$$Q_h^2\, u_L \;\; = \;\; u_L + (\lambda - 5/4\lambda^2 - \lambda^3/2 + 3/4\lambda^4)\,(u_L - u_R)$$

$$Q_h^2 u_R \;\; = \;\; u_R + (\lambda + 5/4\lambda^2 - \lambda^3/2 - 3/4\lambda^4)\,(u_L - u_R)$$

The error estimation procedure gives an estimate proportional to the discontinuity,

$$Q_{2h} - \frac{(Q_h^2 u_L + Q_h^2 u_R)}{2} = (\frac{-\lambda}{2} + \frac{\lambda^2}{2})\,(u_L - u_R).$$

For strong shocks this will always trigger the refinement. Since it is not always necessary to refine to the maximum level at shocks, a procedure to turn off the refinement must be included. For the nonlinear Euler equations however, if a planar stationary shock is aligned with the grid then (7) will predict no error, which is the case. If there is a reason to refine the shock it will have to be triggered by some additional criteria.

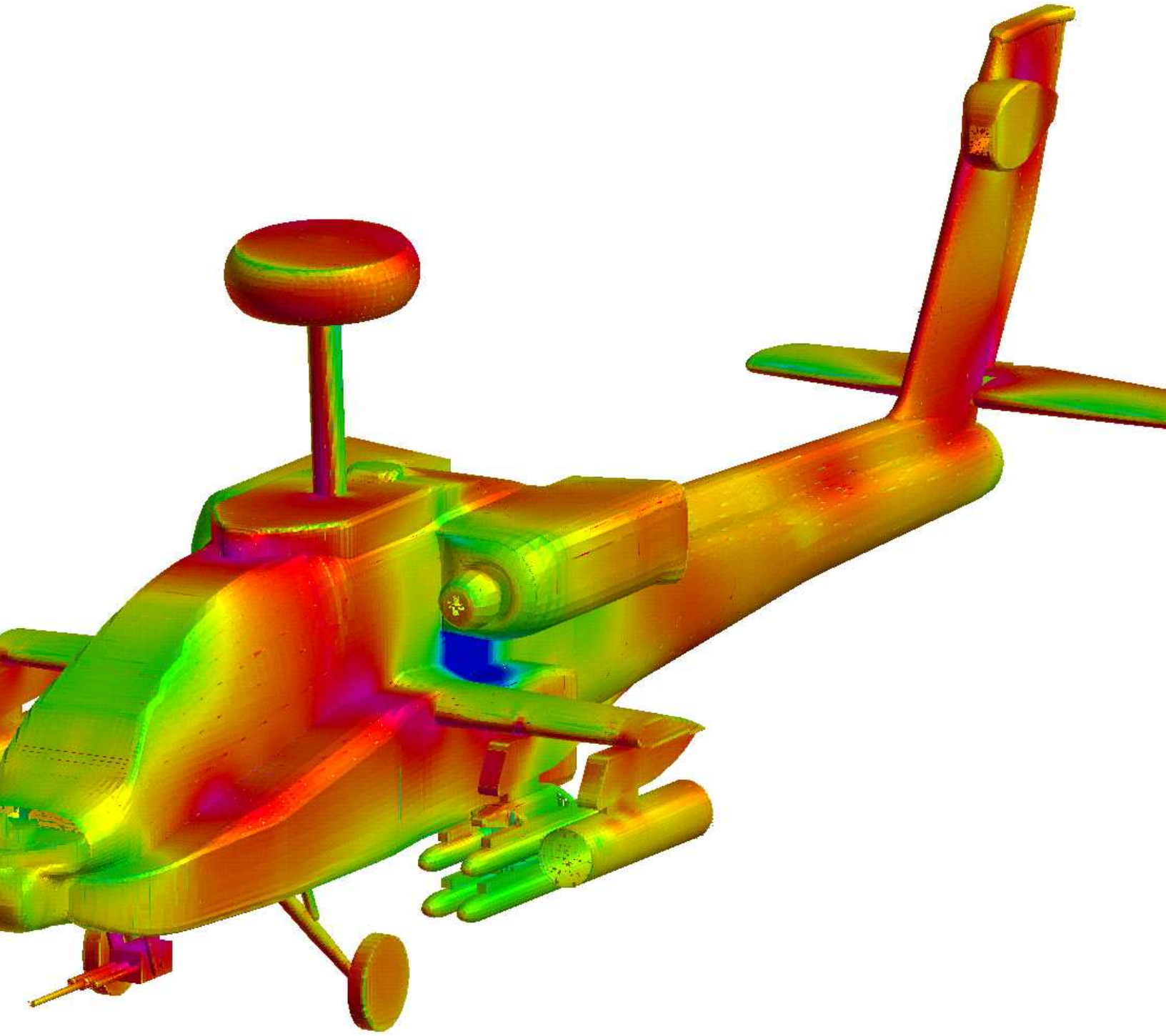
Other procedures for deciding where to refine the grid, such as measuring the gradient of the solution, are also easily seen to fail in certain cases. For example, in a simple test using a shock tube, gradient estimates overly refine the majority of the rarefaction wave, whereas it is the corners of the wave that need the extra resolution.

We conclude that the how-to of mesh adaptation is well-enough understood to be used routinely. (Unfortunately, the software infrastructure for this is not as far along, but is improving). A better understanding of where to refine the grid to control the error in the region of interest is still a pressing issue.

4 Cartesian Non-Body-Fitted Grids

As with the adaptation algorithm, our approach to treating complicated geometry is based on the use of regular grids as much as possible. We use a Cartesian mesh, rather than the body-fitted alternatives of either structured hexahedral or unstructured tetrahedral grids [17], [2]. In this approach, a physical object is simply 'cut out' from the underlying Cartesian grid, leaving a border of irregularly shaped cells at the intersection between the grid and the geometry. We call this a non-body-fitted grid because the faces of the intersected cells do not conform to the surface but instead intersect it in an arbitrary manner. By sticking to Cartesian grids and well-understood algorithms from computational geometry, very complex configurations can be handled without the labor-intensive case-by-case analysis usually required [16]. Since a Cartesian grid lacks the inherent resolution of a specially-constructed mapped coordinate system, the use of mesh refinement (both solution adaptive and to accurately represent details of the geometry) is crucial to the success of this approach. Figure 4 shows an example of the type of complicated geometry that is representable with this approach. This particular geometry was specified by 320,000 triangles in 82 separate components. The preprocessing step of intersecting the separately defined components, retriangulating them, and removing those in the interior of the geometry took under 4 minutes on an R4000 workstation (see [1] for details).

Away from solid boundaries the Cartesian mesh is regular and locally uniform. Finite difference/volume schemes are easily implemented and retain their full order of accuracy (in smooth flow). Special difference formulas are used only at the solid boundaries and cut cell faces, along with easy to compute geometric quantities such as cell face and volume centroids. Since the grid is not smooth, it is very difficult to determine the order of accuracy of schemes near the wall, as discussed in section 2. In [11] a model problem with a known solution is used to numerically determine the order of accuracy. The results show that close to second



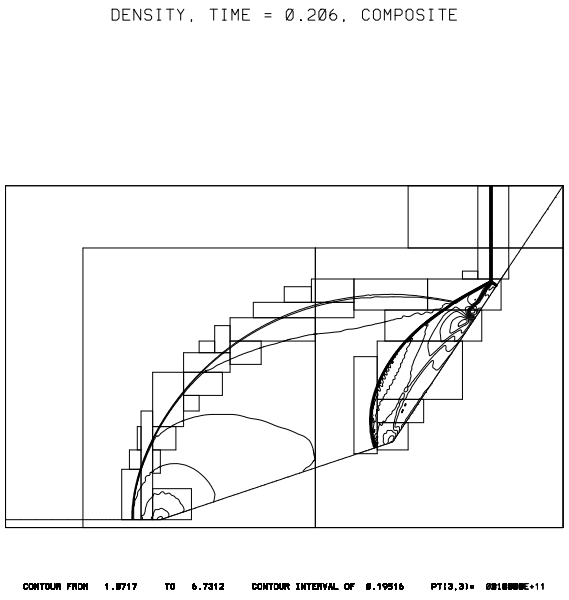


Figure 2: Time dependent shock reflection from a double ramp using h-box method with Cartesian grids. Density contours are shown.

order accuracy can be achieved. By improving the numerical accuracy at the cut cells near the surface and developing anisotropic refinement strategies, further improvement in the efficiency of this approach is possible.

Time dependent problems have an important additional requirement. Since cut cells may be several orders of magnitude smaller than a regular cell, schemes that are stable using a time step based on a full cell volume must be devised. Three approaches have been proposed to date, but none of them is completely satisfactory. These approaches are the h-box method [10], flux redistribution [12], and cell merging [22]. In [5] the Cartesian grid method was extended to handle moving objects using an alternate form of cell merging. The h-box method is the most accurate (somewhat less than second order) but is the most complicated, and has only been implemented in two dimensions. It involves constructing auxiliary cells normal and tangential to the boundary of the object in each cell, and intersecting these auxiliary cells with the Cartesian mesh. A promising development along these lines is an operator split version of the scheme, which appears to be much simpler to implement [15]. The second and third approaches to time dependent problems are only first order, but they are easier to program. Flux redistribution can also be viewed as a way to implement cell merging; however it is based on a very diffusive first order method of computing the fluxes. Development of a fully second order time dependent scheme that is stable at cut cells remains an open problem.

Figure 2 shows a computation of shock reflection off a double ramp using the h-box approach with AMR. Three levels of grid refinement were used; the grids are outlined. This test problem was part of the 1994 ICASE Workshop on Adaptive Methods.

At present, Cartesian non-body-fitted grids are used to compute only inviscid flow. A first step towards extending this approach to viscous flow was taken in [13]. The development of a viscous capability would greatly extend the applicability of the method.

5 Data Structures

Non-body-fitted Cartesian grids are a natural extension of our philosophy of using regular grids for purposes of adapting the mesh. An obvious question arises: are block-structured grid patches still a natural data structure for realistic three dimensional geometries? The requirement of representing an inherently two-dimensional surface, which may be quite convoluted, using three dimensional isotropic refinements blocked into patches may lead to storage inefficiencies. By comparison, one might initially think a code using cell-by-cell refinement along with an unstructured data structure should have less overhead. In this section, we review the algorithm used to generate the grid blocks and present results of numerical experiments comparing the memory overhead of the two approaches.

The grid patch algorithm takes as input a set of tagged cells at a given level of refinement. These cells are next grouped into clusters. Each cluster will become a grid patch at the next level of refinement. The algorithm does not depend on how the cells were tagged: gradient estimates, error estimates, geometry curvature, or a combination of the above can be used. The patching algorithm uses a recursive bisection procedure. Initially, all tagged cells are enclosed in one patch. If the patch is inefficient, the grid is divided in two by a plane parallel to a coordinate direction. The efficiency of a patch is easily measured as

$$Efficiency = \frac{\#\, tagged\, cells}{total\, \#\, cells}.$$

The best subdivision is determined using a fast procedure based on signatures and edge detection algorithms from computer vision [9]. In general, the signature of a function f in a coordinate direction x is defined as

$$\Sigma_x = \int_y f(x,y)dy.$$

In our case, f can be viewed as a discrete binary function which is 0 or 1 depending on whether the cell is flagged or not. The signature is then the number of flagged cells in each coordinate direction. The rectangle is divided at an "edge" of the function f. Intuitively this is a transition from a flagged cell region to a unflagged region. In the computer vision literature, edges are detected as zeros of the second derivative of the signature. In a typical time dependent problem where new grids are generated every few time steps, the grid generation algorithm takes less than 1% of the total CPU time. The procedure is illustrated in figure 3.

We performed the following experiment to compare the overhead of the patched-based refinement with the one that refines the grid on a cell-by-cell basis. Block structured grids were generated by taking the grid produced by the cell-by-cell method as input, then processing it with the blocking algorithm. All cells in the original mesh were refined to the same level in the blocked case. Table 3 contains ratios of the total number of cells generated by the blocking algorithm to the number of cells in the input mesh. Three cell-by-cell meshes were used as input. Two of the input cases were from coarse and fine meshes generated for a simple ONERA M6 wing, while a third mesh came from a more complex and realistic transport aircraft configuration that included wings, flaps, and nacelles. The original input data set for the M6 wing contained 280,000 cells in the coarser case and 1,200,000 in the finer grid case. The complete aircraft has 2,000,000 cells. The input data had 8, 9 and 10 levels of refinement respectively.

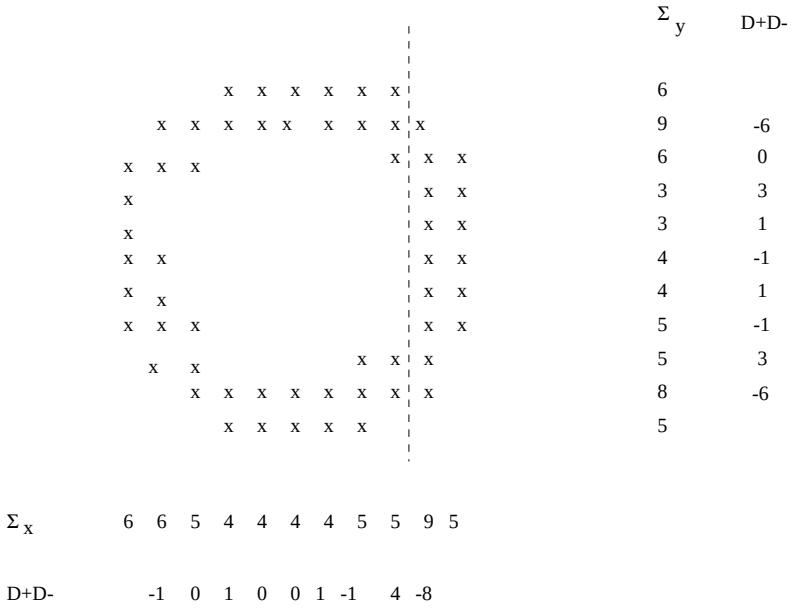


Figure 3: Illustration of algorithm that subdivides an inefficient grid patch. Tagged cells are marked with an "x". The best location and direction to subdivide the cluster is shown with a dotted line.

These results show that approximately twice as many cells are refined by the blocking algorithm as in the cell-by-cell case. Since the storage overhead for the blocked grids is negligible, and the overhead in the cell-by-cell case is at least 10 words per cell (and often on the order of 50), the total storage is still less. Of course other factors impact any comparison of overall efficiency. The blocked grids have twice as many cells to integrate, but the cell-by-cell mesh is hindered by indirect addressing, which is approximately twice as slow. Vector lengths will also play a role in the patch case. In future investigations we plan a more detailed investigation to quantify these trade-offs. However, these results make it clear that the block structured approach to AMR remains quite competitive for realistically complex problems.

6 Conclusions

In this paper we have surveyed our approach to adaptive mesh refinement and to Cartesian grid generation for complex configurations. Our goal in both these efforts is to perform high quality computations in as automatic a fashion as possible. While the last ten years have seen a lot of progress in this direction, we have tried to sketch the many difficulties remaining.

Efficiency	ONERA M6 (coarse)	ONERA M6 (fine)	Full Aircraft
45%	2.3	2.2	2.1
50%	2.1	2.0	2.0
55%	2.0	1.7	1.8

Table 3: Ratio of total number of cells as a function of blocking efficiency parameter.

7 Acknowledgements

M. Berger was supported in part by AFOSR Grant 94-1-0132, and DOE Grants DE-FG02-88ER25053 and DE-FG02-92ER25139. Some of this work was performed at RIACS, whose support is gratefully acknowledged.

References

[1] M. Aftosmis, J. Melton, and M. Berger. Robust and efficient Cartesian mesh generation for component-based geometry. January 1997. Submitted to 35th AIAA Aerospace Sciences Meeting.

[2] M. Aftosmis, J. Melton, and M.J. Berger. Adaptation and surface modeling for Cartesian mesh methods. *AIAA-95-1725*, June 1995. 12th AIAA Computational Fluid Dynamics Conference.

[3] A. Almgren, J. Bell, P. Colella, and L. Howell. An adaptive projection method for the incompressible Navier-Stokes equations. In *Proc. IMACS 14th World Conference*, Atlanta, GA, July 1994.

[4] R. Almgren and A. S. Almgren. Phase field instabilities and adaptive mesh refinement. In *Modern Methods for Modeling Microstructure in Materials*. TMS-SIAM, Oct. 1995. Proc. TMS meeting, Cleveland, Ohio.

[5] S. Bayyuk. *Euler Flows with Arbitrary Geometries and Moving Boundaries*. PhD thesis, University of Michigan, 1996.

[6] M. Berger and A. Jameson. An adaptive multigrid method for the Euler equations. *Lecture Notes in Physics*, 218, 1984.

[7] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–84, 1989.

[8] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53:482–512, 1984.

[9] M. J. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE Trans. Sys. Man and Cyber.*, 21:1278–1286, Sept./Oct. 1991.

[10] M.J. Berger and R.J. LeVeque. Stable boundary conditions for Cartesian grid calculations. In *Proc. Symposium on Computational Technology for Flight Vehicles, Pergamon Press*, Nov. 1990. ICASE Report No. 90-37.

[11] M.J. Berger and J. Melton. An accuracy test of a Cartesian grid method for steady flow in complex geometries. In *Proc. Fifth Intl. Conf. Hyperbolic Problems*, 1994. Stonybrook, NY; Also, RIACS Report 95-02.

[12] I. Chern and P. Colella. A conservative front tracking method for hyperbolic conservation laws. Technical Report UCRL-97200, Lawrence Livermore National Laboratory, July 1987.

[13] W.J. Coirier. *An Adaptively Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*. PhD thesis, University of Michigan, 1994. See also NASA TM 106754.

[14] R. Pember et al. The modeling of a laboratory natural gas-fired furnace with a higher-order projection method for unsteady combustion. Technical Report UCRL-JC-123244, LLNL, February 1996.

[15] H. Forrer. Boundary treatment for a Cartesian grid method. Technical Report 96-04, ETH, April 1996.

[16] J. Melton. *Automated Three-Dimensional Cartesian Grid Generation and Euler Flow Solutions for Arbitrary Geometries*. PhD thesis, U.C. Davis, June 1996.

[17] J. Melton, M. Aftosmis, M. Berger, and M. Wong. 3D applications of a Cartesian grid Euler method. *AIAA-95-0853*, January 1995.

[18] M. Minion. *Two Methods for the Study of Vortex Patch Evolution on Locally Refined Grids*. PhD thesis, University of California, Berkeley, May 1994.

[19] W. Noh, M. Gee, and G. Kramer. Technical Report UCID-18515, Lawrence Livermore National Laboratory, 1979.

[20] J. Oliger and X. Zhu. Stability and error estimation for component adaptive grid methods. *J. Comp. Appl. Math, to appear*, 1996. Also, RIACS Report 94.14, August, 1994.

[21] R. Pember, J. Greenough, and P. Colella. An adaptive, higher-order Godunov method for gas dynamics in three-dimensional orthogonal curvilinear coordinates. Technical Report UCRL-JC-123351, LLNL, February 1996. Submitted to J. Comp. Phys.

[22] J.J. Quirk. An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two dimensional bodies. Technical Report 92-7, ICASE, 1992.

[23] A. Roma. *A Multilevel Self Adaptive Version of the Immersed Boundary Method*. PhD thesis, New York University, January 1996.

[24] B. Wendroff and A. White. Supraconvergent schemes for hyperbolic equations on irregular grids. *Notes on Numerical Fluid Mechanics*, 24, 1989.

[25] P. Woodward. Piecewise parabolic methods for astrophysical fluid dynamics. In K.-H. Winkler and M. Norman, editors, *Astrophysical Radiation Hydrodynamics*, 1986.