Performance of Embedded Boundary Methods for CFD with Complex Geometry

David Trebotich[†], Brian Van Straalen, Dan Graves and Phil Colella

Applied Numerical Algorithms Group, Lawrence Berkeley National Laboratory
[†]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory

E-mail: `trebotich1@llnl.gov`

Abstract. In this paper, we discuss some of the issues in obtaining high performance for block-structured adaptive mesh refinement software for partial differential equations in complex geometry using embedded boundary / volume-of-fluid methods. We present the design of an adaptive embedded boundary multigrid algorithm for elliptic problems. We show examples in which this new elliptic solver scales to 1000 processors. We also apply this technology to more complex mathematical and physical algorithms for incompressible fluid dynamics and demonstrate similar scaling.

1. Introduction

We use a Cartesian grid embedded boundary (EB) method to discretize partial differential equations in the presence of complex geometry in 3D. In this approach, the irregular domain is discretized as a collection of control volumes formed by the intersection of the problem domain with the cubic Cartesian grid cells. The various operators are approximated using finite volume differences on the irregular control volumes based on the divergence theorem. This approach has been used as the basis for second-order accurate methods for elliptic, parabolic and hyperbolic PDEs in two and three dimensions [3, 9], and have been combined using the predictor-corrector approach in [1] to solve the equations of incompressible fluid dynamics in irregular geometries [10]. One of the principal advantages of an embedded boundary volume-of-fluid method is that it is compatible with a robust and accurate level-set technique for grid generation of geometries obtained from image data [4, 8]. Furthermore, the EB approach is amenable to performance-boosting strategies such as adaptive mesh refinement (AMR) and parallelism. Here, we present the algorithm design for high performance computations of elliptic problems using adaptive embedded boundary methods and extend this computational approach to the multiple elliptic operators needed to solve the PDEs which describe incompressible fluid dynamics.

2. AMR Multigrid Algorithm

First, we present the design of a non-EB adaptive multigrid algorithm for elliptic problems. `AMRMultigrid` is a natural extension of a multigrid iteration algorithm, with this particular version being based on [6]. A pseudo-code description of the algorithm is given in Figure 1. The operator L^{nf} is a two-level discretization of the Laplacian:

$$L^{nf}(\psi^l, \psi^{l-1,valid}) = D(\vec{G}^f(\psi^l, \psi^{l-1,valid})) \tag{1}$$

procedure Solve(l, l^{base})
$R := \rho - L(\varphi)$
while ($||R|| > \epsilon||\rho||$)
  AMRVCycleMG($l-1, l^{base}$)
  $R := \rho - L(\varphi)$
end while

procedure AMRVCycleMG(l, l^{base})
  if ($l = l^{max}$)
    $R^l := \rho^l - L^{nf}(\varphi^l, \varphi^{l-1})$
  end if
  if ($l > l^{base}$)
    $\varphi^{l,save} := \varphi^l$ on Ω^l
    $e^l := 0$ on Ω^l
    RelaxLevel($L^{nf}, e^l, R^l, n_{ref}^{l-1}, false$)
    $\varphi^l := \varphi^l + e^l$
    $e^{l-1} := 0$ on Ω^{l-1}
    $$R^{l-1} := \begin{cases} Average(R^l - L^{nf}(e^l, e^{l-1})) & \text{on } \mathcal{C}_{n_{ref}^{l-1}}(\Omega^l) \\ \rho^{l-1} - L^{comp,l-1}(\varphi) & \text{elsewhere in } \Omega^{l-1} \end{cases}$$
    AMRVCycleMG($l-1, l^{base}$)
    $e^l := e^l + I_{pwc}(e^{l-1})$
    $R^l := R^l - L^{nf}(e^l, e^{l-1})$
    $\delta e^l := 0$ on Ω^l
    RelaxLevel($L^{nf}, \delta e^l, R^l, n_{ref}^{l-1}, false$)
    $e^l := e^l + \delta e^l$
    $\varphi^l := \varphi^{l,save} + e^l$
  else
    SolveLevel(L^{nf}, e^0, R^0) (solve $L^{nf}(e^0) = R^0$ on Ω^0)
    $\varphi^0 := \varphi^0 + e^0$
  end if

Figure 1. Pseudo-code description of the AMR multigrid algorithm.

The smoothing operator `RelaxLevel`($\varphi^f, R^f, r, final$) performs a multigrid V-cycle iteration on φ^f for the operator L^{nf}. In this V-cycle, level f is only coarsened until it is a factor of 2 finer than the next coarser AMR level. The coarse-grid values required for the boundary conditions are assumed to be identically zero.

Code re-use is facilitated by using a *Template Method* design pattern [5]. The purpose of the *Template Method* design pattern is to define an algorithm as a fixed sequence of steps where one or more of the steps is over-ridden. In our case we have a hierarchy of algorithms that we wish to re-use across a family of applications, such as the various elliptic operators involved in solving the equations of incompressible fluid dynamics. The hierarchy of algorithms is defined by the specific *variable steps* that an application must provide to complete the algorithm.

3. EB AMR Elliptic Operator

The `EBAMRElliptic` operator implements the `AMRMultigrid` template to solve elliptic problems with complex geometry. There are several issues that must be addressed to obtain solutions to elliptic problems in complex geometry where EB is employed: flexible data structures for fast computations where special discretization stencils are employed; elimination of temporary data; minimization of communication; and operator-dependent load-balancing.

3.1. Serial Performance

First, we optimized for performance in serial. Flux calculations in irregular cells involve specialized stencils. Aggregate stencil operations, where only pointers to memory and integer offsets are stored and recalled for calculations in irregular cells, resulted in the most significant EB-specific improvement – a factor of 100 over our previous initial implementation. A caching technique was also developed, based on pointers and integer offsets, in order to eliminate storage of temporary data structures, when, for example, updating data in place occurs. This turned previously high-order indirection access into single-order indirection and much shorter instruction depth, which is more suitable for current micro-processors.

The original relaxation strategy involved 2^D-color relaxation, requiring a data striding of 2^D. This was robust and stable but incurred a very high serial performance cost because of its poor re-use of cache-line data on cache-based processors. Careful analysis allowed us to replace this scheme with a 2-color red-black relaxation which both reduced the intra-level communication costs, and allowed greater re-use of data in cache lines on relaxation sweeps. The elimination of redundant application of boundary conditions also helped regularize the data access pattern.

3.2. Parallel Optimizations

In addition to serial optimization, distributed memory parallel performance requires algorithm modifications. Unlike the non-EB AMR elliptic solver, it is not adequate to perform a static load balance of boxes to processors based on the volume of a box. Prior to executing the `AMRMultigrid` algorithm, each AMR level is re-balanced based on our `EBEllipticLoadBalance` routine. `EBEllipticLoadBalance` computes the Laplacian operator on each box and measures the actual time taken. This data is then passed to the load balance routine that subdivides the space-filling curve. Space-filling curve algorithms are a means of ordering boxes within a level to minimize intra-level communication. We use a Morton ordering approach. If D is the spatial dimension of the problem, Morton ordering [7] is a 1-1 mapping of Z^D onto Z with good locality: the fraction of nearest neighbors in Z^D of the inverse image of an interval $\mathcal{I} \subset Z$ of length M whose Morton indices are not in $\mathcal{I}$ is $O(M^{-1/D})$. See [2] for details.

Some additional optimizations in parallel result from minimizing communication in the relaxation scheme. A "lazy" relaxation scheme needs only to exchange data with other boxes once per iteration, not the number of colors per iteration, because of the checker board operations. While this optimization is not quite the optimal strategy in serial, it is a significant improvement in parallel. Similarly, we can eliminate multiple applications of boundary conditions in this "lazy" routine.

Box edge and corner communication can also be minimized by trimming the number of ghost cells needed to communicate boundary conditions (e.g., Neumann requires one ghost cell per cell, Dirichlet, three). This optimization does not have a significant impact on the amount of data needing to be communicated, but it does have a significant impact on the number of processors a given processor must communicate with, thus reducing the number of messages. This improves latency, and intra-level relaxation communication is latency bound, not bandwidth limited.

Inconsistent stencils, that is, stencils where fluxes on embedded boundaries are simply determined by the cell-centered value of the irregular cell and not by proper second-order extrapolations, led to faster multigrid iterations; however, multigrid convergence was a factor of 5 worse, and, thus, not an overall win.

3.3. Results for Poisson's Equation

To demonstrate scaling and performance of our `EBAMRElliptic` operator we constructed a weak scaling test. In weak scaling, the size of the problem is increased with the number of processors, as opposed to holding the size of the problem fixed as in strong scaling (see [2] for details). For example, we solve Poisson's equation around a sphere inside of a box on 8 processors at

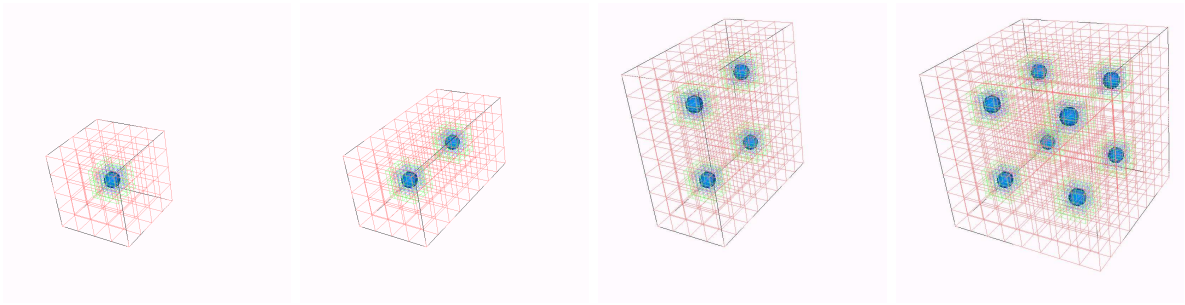

Figure 2. Example of replication of grids for scaling test. One replication unit (L) is equivalent to a base grid resolution of $32 \times 32 \times 32$, with 3 levels of adaptivity, factor of 2 refinement, and maximum grid resolution for an individual box in the AMR layout of $16 \times 16 \times 16$ on 8 processors. Grids are replicated by a factor of 2 from left to right, from 8-64 processors.

a base grid resolution of $32 \times 32 \times 32$ with 3 levels of adaptivity, factor of 2 grid refinement. This problem represents one replication unit. For 16 processors the number of cells and length is doubled in one dimension, adding another sphere such that the base grid resolution is $64 \times 32 \times 32$ and the domain is a rectangular prism. The replication is repeated for 32 processors in another dimension. Figure 2 shows a factor of 8 replication for the 64 processor test problem in the suite.

In Figure 3, we show plots of wall clock time for the elliptic solver part of the calculation on the Cray XT4 for 8 to 1024 processors. We normalize the times by the time to solve the smallest problem (one replication unit, 8 processors, or 0.074 seconds). We observed 54% efficient scaled speedup over a range of 8-256 processors, corresponding to a wall clock time of 0.074-0.135 seconds. For higher concurrencies, we observe 44% efficient scaled speedup over a range of 8-1024 processors.

3.4. Implementation and Results for Incompressible Navier-Stokes

We extended the EB elliptic solver performance optimizations to a predictor-corrector algorithm for incompressible, viscous flow (see [1], for example). The elliptic solver is re-used across multiple operators in one advance timestep: two Poisson equations for the pressure projections that enforce incompressibility (advection and update steps); three Helmholtz equations for the components of velocity; and an additional Poisson equation for the projection filter in the update. Performance is also optimized by making certain derived quantities and diagnostics optional, such as norm calculations for checking incompressibility of the velocity field, as well as I/O.

We performed a weak scaling test for the `EBAMRINS` solver that is based on replication similar to that for `EBAMRPoisson`, but with a coarser base grid by a factor of 2. In Figure 3 we demonstrate a scaling pattern for the `EBAMRINS` solver that is very similar to that of `EBAMRPoisson`. We note that this study was focused primarily on elliptic scaling, but the pattern indicates that the hyperbolics scale very well.

4. Conclusion

We have demonstrated the scalability of adaptive embedded boundary methods to 1000 processors for elliptic problems with complex geometry and have extended this technology to an incompressible Navier-Stokes solver. Aggregate stencil operations have all but eliminated the time needed to perform calculations for the special stencils in irregular cut cells resulting from the embedded boundary approach. Operator-dependent load balancing with Morton ordering helps minimize intra-level communication. An efficient relaxation scheme has further improved

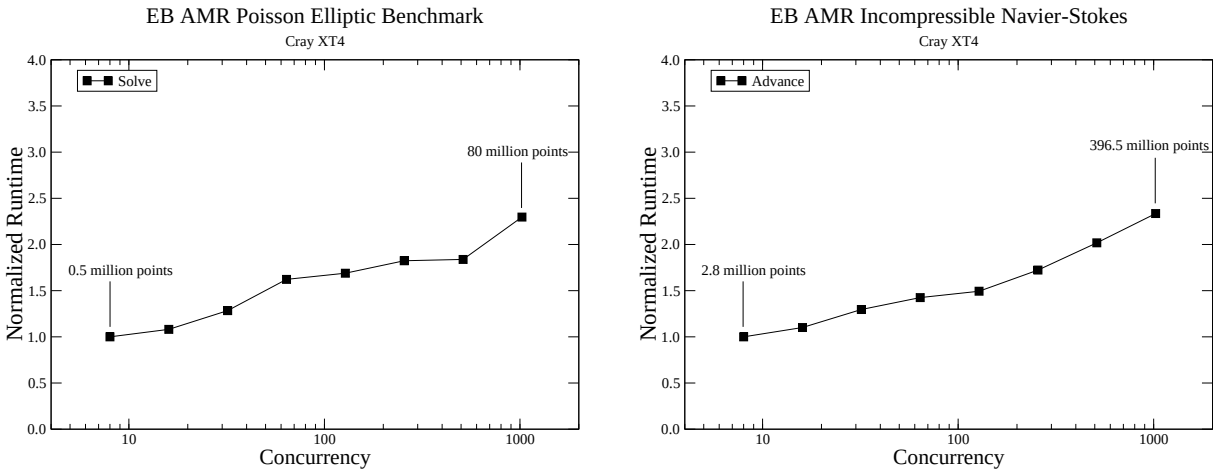

Figure 3. (L) Weak-scaling results for the `EBAMRPoisson` solver for 8 to 1024 processors. Replication of grids for the scaling test is shown in Figure 2. (R) Weak-scaling results for the `EBAMRINS` solver for 8 to 1024 processors, with a factor of 2 coarser base grids.

performance of elliptic solvers by better caching and cache re-use. In conclusion, embedded boundary methods are a fast, efficient and simple technology for handling grid generation for complex geometry and, when combined with adaptive mesh refinement, provide a powerful high-resolution tool for modeling multi-scale fluid dynamics in complex geometry.

References

[1] J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85:257–283, 1989.

[2] P. Colella, J. Bell, N. Keen, T. Ligocki, M. Lijewski, and B. Van Straalen. Performance and scaling of locally-structured grid methods for partial differential equations. *Journal of Physics: Conference Series*, 78(012013), 2007.

[3] P. Colella, D. T. Graves, B. Keen, and D. Modiano. A Cartesian grid embedded boundary method for hyperbolic conservation laws. *J. Comput. Phys.*, 211:347–366, 2006.

[4] T. Deschamps, P. Schwartz, D. Trebotich, P. Colella, D. Saloner, and R. Malladi. Vessel segmentation and blood flow simulation using level-sets and embedded boundary methods. In *International Congress Series*, volume 1268, pages 75–80, June 2004.

[5] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[6] D. Martin and K. Cartwright. Solving Poisson's equation using adaptive mesh refinement. Technical Report M96/66, University of California, Berkeley Electronic Research Laboratory, October 1996. Code and documentation are available at `http://barkley.me.berkeley.edu/~martin/public_html/AMRPoisson.html`.

[7] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd., Ottawa, Ontario, Mar 1966.

[8] P. Schwartz, D. Adalsteinsson, P. Colella, A. P. Arkin, and M. Onsum. Numerical computation of diffusion on a surface. *Proc. Natl. Acad. Sci. USA*, 102:11151–11156, 2005.

[9] P. O. Schwartz, M. Barad, P. Colella, and T. J. Ligocki. A Cartesian grid embedded boundary method for the heat equation and Poisson's equation in three dimensions. *J. Comput. Phys.*, 211:531–550, 2006.

[10] D. Trebotich, G. H. Miller, P. Colella, D. T. Graves, D. F. Martin, and P. O. Schwartz. A tightly couple particle-fluid model for DNA-laden flows in complex microscale geometries. *Computational Fluid and Solid Mechanics*, pages 1018–1022, 2005.