Front Tracking Algorithm Using Adaptively Refined Meshes

Zhiliang Xu[1], James Glimm[12], and Xiaolin Li[1]

[1] Department of Applied Mathematics and Statistics, University at Stony Brook
xuzhi@ams.sunysb.edu, linli@ams.sunysb.edu
[2] Center for Data Intensive Computing, Brookhaven National Laboratory
glimm@ams.sunysb.edu

Summary. We propose a new algorithm which combines the front tracking with an adaptively refined Cartesian grid for solving systems of nonlinear conservation laws.

1 Introduction

The numerical simulation of time dependent complex flows which contain linear and nonlinear discontinuities is a challenge. To maintain the quality of the computed solution and to capture the solution structures of different scales, it is desirable to use adaptive grid refinement (AMR).

The AMR strategy is to expend the most computation effort in the region where it is most necessary by using more grid points here. Block-structured Cartesian grids for the refinement of complicated geometric configurations were proposed [BO84, BJ85]. The study of block-structured AMR for hydrodynamics has been developed systematically by M. Berger and P. Collela [Ber87, BC89].

The front tracking method is also an adaptive method. A front represents either a contact discontinuity, a shock wave or a rarefaction wave edge of the flow. The front represented by a lower dimensional manifold is embedded in the spatial grid. The geometry of a front is represented by a curve composed of piecewise linear segments in 2D, and a surface composed of a set of connected triangles in 3D. Driven by the flow dynamics, the front follows the flow motion and moves freely across the underlying spatial grids to track the desired flow features. Interfacial numerical diffusion across the fronts, which is a common phenomenon for the capturing method, is prevented by the tracking method. The tracking method also reduces post-shock oscillations. Two sub-processes: the front propagation and the updating of the solution on the spatial grids, complete a single computational time step. The front tracking method has been used to simulate complex fluid mixing geometries in 2D [GGMM88, CGSZ96], and in 3D [LG96, GGL98].

This paper describes a combination of these two methods. We have merged the Front Tracking code FronTier with the AMR code Overture which is based on the

Berger-Collela cell based refinement algorithm, [Ber87, BC89] and developed and maintained at LLNL.

2 The 2D AMR Front Tracking Algorithm

We describe the modules of the AMR front tracking algorithm according to the order of their execution in this section. We use block-structured uniform Cartesian grids. We use Overture to generate the AMR grid hierarchy. FronTier is responsible for updating the numerical solutions on each patch.

2.1 Initialization of the AMR grids

For an initial regular, coarse mesh filled with data by the FronTier initializer, the cells in this mesh where the estimated error is larger than some threshold are determined to be tagged. The Berger-Collela cell based refinement algorithm then generates refinement grids from these tagged cells by organizing them into a set of rectangular patches. The patches are aligned with the original mesh. The patches are properly nested, so that successive refinement levels are allowed, with the restriction that adjacent refinement levels will differ by one level only, usually representing refinement by a factor of 2 or 4 in each dimension. The fine patches are recursively added to the next coarse level patches until the desired solution accuracy is obtained. The patches with the same grid spacing belong to the same grid level. The refinement grids are organized in a hierarchy. The coarsest grid which discretizes the whole domain belongs to level 0, the next added refinement grids belong to level 1 and so on.

The error estimation provides the basis for the AMR strategy. The Overture implementation of the error estimate is based on the evaluation of a weighted sum of the first and second derivatives of one or several state variables. The estimator generally achieves a maximum value at discontinuities. In this way, the grid adaptation thus tends to refine this region, which is exactly what we desire.

We address the tracking of the contact discontinuity. We assure that the tracked front is completely embedded inside the finest grid. To ensure there are a sufficient number of the fine grid cells to cover the contact discontinuity, we use a controlling parameter which specifies the minimum distance from the front to the fine grid boundary. Cells with a distance to the front smaller than this parameter are always tagged during the refinement. Thus the finest level patches always cover a minimum neighborhood of the tracked front.

To implement the above described module, we have built a data translation interface to translate the FronTier initial state data to the Overture error estimation class and to translate the refinement grid hierarchy from Overture to FronTier.

Since the numerical computation step carried by FronTier is organized into front propagation and interior state update sub-processes, for each AMR patch, we construct a front structure and a wave structure. FronTier then calls the front propagation solver and interior finite difference solver to update the numerical solutions.

2.2 The Single Time Step Tracking Algorithm

To support dynamic tracking, a front propagation driver associated with the underlying physics and a set of geometric manipulation routines are provided [GIM81, GGL98, GGL99a]. Basic operations such as the creation and the re-meshing of the front and the untangling of self-intersecting fronts are managed by geometric manipulation routines.

The front is oriented. Each front point has two states defined, one on each side. The front propagation is accomplished by propagating each point on the front. To propagate the front point, we define the normal and tangential direction on the front. We then solve

$$U_t + n \cdot ((n \cdot \nabla)F(U)) + \tau \cdot ((\tau \cdot \nabla)F(U)) = 0, \tag{1}$$

where $n = (n_1, n_2)$ is the normal, $\tau = (-n_2, n_1)$ is the corresponding tangent. Eq. (1) is solved by dimensional splitting. Given the two side states U^n at the time t_n, we first solve

$$U_t + n \cdot ((n \cdot \nabla)F(U)) = 0, \text{ with } U(t = t_n) = U^n. \tag{2}$$

The time evolution defines a generalized Riemann problem as introduced in [CGMP86]. From the solution of this problem, we obtain the new location of the front point and the two new states on each side of the front. We denote these two states as U^{cn}. Then we solve

$$U_t + \tau \cdot ((\tau \cdot \nabla)F(U)) = 0, \text{ with } U(t = t_n) = U^{cn}, \tag{3}$$

with any finite difference solver on each side. The point propagation is complete. Since the front contained in the finest grid level is most accurate, we only maintain the front in this level. The geometric operation routines are called after point propagation to re-mesh or untangle the front if there is the need. The front in a grid communicates with neighboring grid fronts to maintain its integrity.

The last step of a single tracking step is to update the states on the spatial grids. See [CGMP86, GGL98, GGL99, GGL99a, GGL00] for a detailed discussion of the front propagation and the interior sweep.

Except for the physical boundaries, the boundary conditions on the refinement grids are imposed by use of ghost cells, which extend the patch cells by a number related to the stencil size of the finite difference scheme in each direction. Using these ghost cell state values in the usual way, the equations can be integrated in the given patch with sufficient boundary data to fill all finite difference stencils. After the updating of spatial grid solutions on the patches, solutions defined on the coarse grid cells which are covered by a finer grid and the solutions defined on the cells which are on a coarse-fine grid boundary but are not covered by any fine grid are subject to modification. At the coarse-fine grid boundaries, the refluxing procedure must be applied to ensure conservation and to reduce errors. The cell states of a fine level patch are averaged to the next coarse level patch in which this fine level patch is nested. With the front present, averaging of states from two sides of the front is not allowed.

The adaptation in the temporal dimension is not implemented at the present time. At the end of the single time step, all patches are synchronized with the same discretized time step. The discrete time increment Δt is the one calculated from the finest level grids.

2.3 Regridding the AMR grids

The dynamic evolution of the solutions requires regeneration of the refinement grid hierarchy after a specified number of time steps to capture the most significant features of the solution. As with the initialization step, FronTier gathers the states on the spatial grids, sends them to Overture, and Overture performs the regridding step according to the information from the states obtained.

The solution states defined on the old refinement grid hierarchy are interpolated to the new refinement grid hierarchy. The fronts in the old fine grid sets are assembled. Then for each of the new refinement grids, a proper portion of the assembled front is cut if it is present inside the grid. The solution is then advanced on the regridded grid hierarchy.

2.4 Parallel computation

FronTier is a fully parallelized software package running on distributed-memory systems while the current Overture implementation is not. Based on the parallelization of FronTier, we also implemented a parallel AMR tracking algorithm. The current strategy is the following: A global domain is divided into rectangular subdomains in FronTier. Each subdomain is assigned to a single processor. In each subdomain, we perform the adaptive refinement. Because the Overture error estimator does not communicate through subdomain boundaries, there might not be enough refinement across the subdomain boundaries. To ensure sufficient refinement, for two adjacent subdomains, assume one has a finer refinement (say level l) patch than the other. The subdomain with less refinement also creates a level l patch with specified depth (usually 4 to 6 cells) and length to match the refined level l patch of its neighboring subdomain. In this way, the refinement is not influenced by the subdomain decomposition.

Since each subdomain has different refinement grids, the computation load is unbalanced. To balance the load on different processors, we need to distribute AMR patches across processors, namely, the processors with excessive load will send some of the patches to other processors with deficient load.

To distribute the unbalanced workload, we need to send the entire states of a patch across the processors. If two nested fine and coarse patches of two successive levels are assigned to different processors, we need to send the entire solution states of the fine patch to the coarse patch at the averaging step of the single time step. Besides these parallel communications, there are other communications associated with front tracking at a fixed grid level. The ghost cell states are filled at the end of the single time step. Except for the ghost cells on the reflected or periodic boundaries, the ghost cell portion of a given patch covers the interior of adjacent patches

at the same level or the next coarse level in which this patch is nested. If the ghost cell of a patch overlaps a cell from the interior of another same level patch, the state of this ghost cell is obtained by copying states from this overlapped cell. Otherwise, the ghost cell state is interpolated from the next coarse level patch in which the ghost cell is located. If these patches are scattered to different processors, we send these copied or interpolated states through the parallel communication. We also communicate fine-coarse boundary fluxes if the corresponding fine and coarse patches are computed in different processors. In the case of Front Tracking, it is also necessary to communicate ghost cell portions of the front. At the end of the normal and tangential front propagation and the re-meshing or the untangling of the front, the ghost cell portion of the front of a given patch is replaced by the front cut from the interior of the neighboring patches. The cut pieces of the front from the neighboring patches must be sent to this given patch either by copying, if these patches are in the same processor or by the parallel communication if these patches are in different processors.

For parallel communication, we construct a table. Each patch corresponds to an item in the table. Each item of this table records a patch id, the processor id in which this patch is computed, ids of all patches neighboring this patch at the same level, the id of the next coarse level patch embedding this given patch and the ids of the next fine level patches embedded in this patch. Each processor maintains a copy of this table. The parallel communication is determined from this table.

We are experimenting with the optimization of load balancing, which must be reconsidered in the context of front tracking, as the computational load is nonuniform, depending on the number of front points as well as the number of cells in a given patch. We use the algorithm developed by W. Crutchfield *et al* [CLB99] to determine the load distribution. This algorithm assumes the communication cost is ignorable. In our case, however, we have large and irregular communications which can not be ignored. To develop a better load balancing strategy is still a research issue.

3 Numerical Examples

We present two numerical examples illustrating our algorithm.

3.1 A spherical Richtmyer-Meshkov instability

We calculate an axisymmetric spherical RM instability. A Mach 10 imploding spherical shock strikes a spherical contact interface perturbed by 6 modes. We used 3 level refinements, with the refinement ratio of 2 and the level 0 grid of 100×100 cells on the 1×1 domain. We compare the result with the uniform grid (which is comparable to the finest level of the AMR grid) simulation. The AMR solution matches the uniform grid solution as shown by Fig. 1.

Fig. 1. Density and grid plots for the spherical RM simulation at time = 0.3. The left frame shows the density plot from AMR front tracking, and the middle frame shows the density plot from uniform grid front tracking. The right frame shows the AMR grid plots.

3.2 A simulation of spray formation in a diesel jet

The wide disparity of length scales in the diesel jet simulation problem (injection nozzle width 0.178mm, mean droplet size 10 micron, jet length during simulation 5cm) calls for AMR. In Fig. 2 we illustrate this capability. We used 5 processors, 3 level refinements with the refinement ratio of 2 and a level 0 grid of 62×250 cells on the $0.248cm \times 1cm$ domain. The left part of the frame shows the density for the simulation, and the right part of the frame shows the grid in a color plot at time 0.06, which is 60 percent of the pressure rising time. Each color in the grid plots represents a processor. The patches in the same subdomain are computed in different processors determined by the load balancing. The comparison with experiment data from Argonne National Laboratory and uniform grid simulations is conducted at Brookhaven National Laboratory.

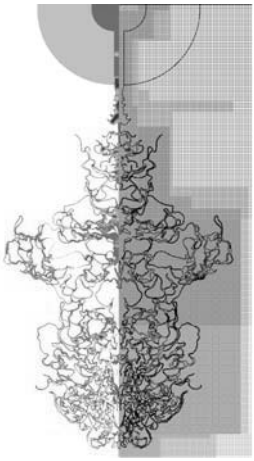

Fig. 2. Density plot of the jet simulation at t = 0.06

4 Conclusions and discussion

We have presented an AMR front tracking algorithm for calculation of time-depend flows. With this AMR front tracking, we have resolved the geometry features of various sizes and scales and maintained a sharp interface. This 2D methodology is also applicable to 3 dimensions, which is now under the development. The Berger-Collela cell based refinement algorithm gives unequal-sized grids. Dynamic load balancing is an important issue in the parallelization. The optimization of load balancing is still being investigated.

Acknowledgments. We would like to thank William B. Henshaw and Petri Fast at LLNL for helpful comments and suggestions during the code development.

References

BO84. M. Berger, J. Oliger: Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations. J. Comp. Phys., **53**, 484–512 (1984)

BJ85. M. Berger, T. Jameson: Automatic Adaptive Grid Refinement for the Euler Equations. AIAA, **23**, 561–568 (1985)

Ber87. M. Berger: Adaptive Finite Difference Methods in Fluid Dynamics. In: von Karman Lecture Notes on CFD NYU/DOE report 03077-277 (1987)

BC89. M. Berger and P. Colella: Local Adaptive Mesh Refinement for Shock Hydrodynamics. J. Comp. Phys., **82**, 64–84 (1989)

GGMM88. C. L. Gardner, J. Glimm, O. McBryan, R. Menikoff, D. Sharp, and Q. Zhang: The Dynamics of Bubble Growth for Rayleigh-Taylor Unstable Interfaces. Phys. Fluids, **31**, 447–465 (1988)

CGSZ96. Y. Chen, J. Glimm, D. H. Sharp, and Q. Zhang: A Two-Phase Flow Model of the Rayleigh-Taylor Mixing Zone Phys. Fluids, **8**, 816–825 (1996)

LG96. X.-L. Li, and J. Glimm: A Numerical Study of Richtmyer-Meshkov Instability in Three Dimensions In: Proceedings of the Second Asia CFD Conference, Tokyo (1996)

GGL98. J. Glimm, J. W. Grove, X.-L. Li, K.-M. Shyue, Q. Zhang, Y. Zeng: Three Dimensional Front Tracking SIAM J. Sci. Comp., **19**, 703–727 (1998)

GIM81. J. Glimm, E. Isaacson, D. Marchesin, and O. McBryan: Front Tracking for Hyperbolic Systems. Adv. Appl. Math., **2**, 91–119 (1981)

GGL99a. J. Glimm, J. W. Grove, X.-L. Li, and D. C. Tan: Robust Computational Algorithms for Dynamic Interface Tracking in Three Dimensions. SIAM J. Sci. Comp., **21**, 2240–2256 (2000)

CGMP86. I-L. Chern, J. Glimm, O. McBryan, and B. Plohr: Front Tracking for Gas Dynamics J. Comp. Phys., **62**, 83–110 (1986)

GGL99. J. Glimm, J. W. Grove, X.-L. Li, and N. Zhao: Simple Front Tracking. In: Contemporary Mathematics, Amer. Math. Soc (1999)

GGL00. J. Glimm, J. W. Grove, X.-L. Li, W. Oh, and H. Sharp: A Critical Analysis of Rayleigh-Taylor Growth Rates. J. Comp. Phys., **169**, 652–677 (2001)

CLB99. William Crutchfield, Vincent E. Beckner, Mike Lijewski, and Charles A. Rendleman, : Parallelization of Structured, Hierarchical Adaptive Mesh Refinement Algorithms. (1999)