UCRL-JC-132741
PREPRINT

Odyssey

D. Braddy, S. Brown, G. Cook, C. Kueny, M. Lambert, D. Peters

This paper was prepared for submittal to the
1998 Nuclear Explosives Development Conference
Las Vegas, NV
October 25-30, 1998

October 1, 1998

ODYSSEY

Dennis Braddy, Stewart Brown, Grant Cook,
Chris Kueny, Mike Lambert, Doug Peters
Lawrence Livermore National Laboratory

We present results obtained with the Odyssey simulation code. Odyssey is a 1, 2, and 3 dimensional AMR code using cartesian, cylindrical, and spherical coordinates. The results provide an interesting snapshot of Odyssey at this point in its development. Results include parallel performance and scaling, Eulerian hydrodynamics algorithm comparisons, ADI based diffusion solvers on hierarchical meshes, ECB treatment of material interfaces in diffusion solves.

Keywords: AMR, hydrodynamics, Godunov, CE/SE, ADI, diffusion, hierarchical mesh, parallel

Introduction

Odyssey is an AMR code whose object is to study issues relating to AMR and to Eulerian hydrodynamics. Our mission is to evaluate these technologies in comparison to others such as ALE or free Lagrange. The immediate goal is to develop a base set of capabilities which are robust and well understood. These will next be applied to carefully selected simulations which will help us to understand the long term benefits of these technologies relative to competing ones which are at a more mature state of development and use.

The principle physics to include is hydrodynamics and radiation diffusion. These two together let us explore explicit methods which require only local interactions between cells and implicit methods in which all cells of a simulation interact through a global matrix solve.

In order to assess the hydrodynamics as distinct from the adaption strategy we have two hydrodynamics packages. One uses a directionally split, piece-wise linear Godunov MUSCL solver and the other uses the conservation element/solution element (CE/SE) method of Chang and is not directionally split.

For the diffusion solver we use the ADI method adapted to a hierarchy of grids. The ADI method is computationally robust and some recent work in the literature suggests good parallel scaling properties for large numbers of processors. To improve the treatment of diffusion coefficients in cells with more than one material we use the embedded curved boundary (ECB) method and exploit the material interface which was reconstructed in the hydrodynamics phase.

Eulerian Hydrodynamics

In the course of evaluating Eulerian hydrodynamics we have identified issues which are important to the quality of solution obtained. Issues which we discuss here are: approximations in Riemann solvers; pure characteristic solutions; and directional splitting schemes. Our Godunov solver is directionally split so that we only have to solve “1d hydrodynamics” problems. The CE/SE method is unsplit. In comparing the two we noted many differences in addition to the similarities. Since both packages run in the same code we can be sure that the initial value problems are the same. Also ancillary information such as the equation of state information is the same for both. So when we look at differences we are really peering directly into the algorithms of the two packages.

In our study of the two packages we became aware of the factors which contribute to the differences in the answers that we were seeing. We discuss first the problem of the approximate Riemann solver. Next we look at the directional splitting. As this is work in progress we have no definitive conclusion as yet.

Godunov Package

Many Godunov based hydrodynamics schemes use approximate Riemann solvers to improve the performance of their packages. The approximate solvers fall into various categories, for example those based on Roe’s method. Odyssey had an approximate solver as described by Colella, Glaz, van Leer, and others [1-3]. The approximate elements of the Riemann solver are:

1. No iterations (or few) to solve for the contact pressure and velocity.

2. Linear interpolation through the rarefaction fan

3. Low order Taylor expansion to find the perturbed state density

For hydrodynamic flows in which there are jumps in field quantities exceeding an order of magnitude these approximations result in observable deterioration of the solution. Most of the Riemann problems solved in the coarse of a simulation involve small jumps where these approximations are valid. However, a certain number of cells will encompass jumps for which they are bad. The result is incorrect propagation speeds of shocks and contacts and a lack of fidelity in various flow features.

We replaced the original Riemann solver with a solver which can be run with the above approximations in arbitrary combinations. We verified the exact solver against published shock tube solutions, then proceeded to run a very strong shock tube problem with each of these approximations in all combinations. The idea is to understand the consequences of the approximations individually and collectively.

We then performed a set of eight simulations on a Mach 80 shock tube using all the combinations of the above listed approximations. We also ran a series of eight on a Mach 1.7 shock tube. In the weak shock tube, the contact discontinuity and the shock front separate by more than one cell in two cycles. By contrast, for the strong shock tube the contact and shock remain less than a cell apart for more than thirty cycles. This is why such problems are difficult for Eulerian hydrodynamics schemes.

In Fig. 1, 2, and 3 we plot the log of density, log of pressure, and velocity for each of the Mach 80 simulations along with the analytic solution. The curves corresponding to approximation which don't get the shock speed right are shown in blue. The remaining approximations give results which are more difficult to distinguish as right or wrong. They are shown in red. The same set of simulations with a Mach 1.7 shock tube showed no significant differences among the various approximations.

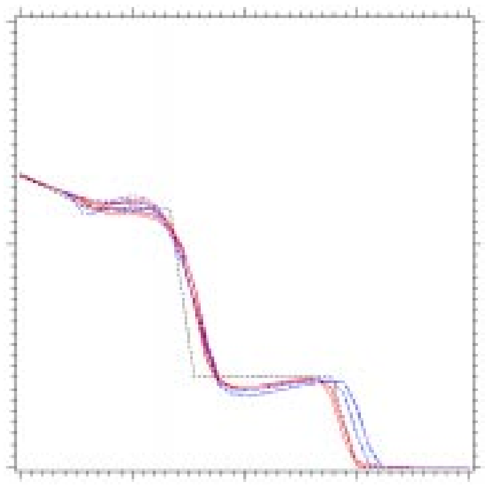
Figure 1. Log of density vs. position

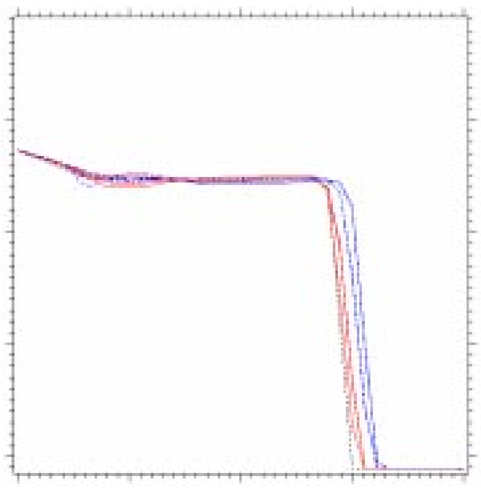
Figure 2. Log of pressure vs. position

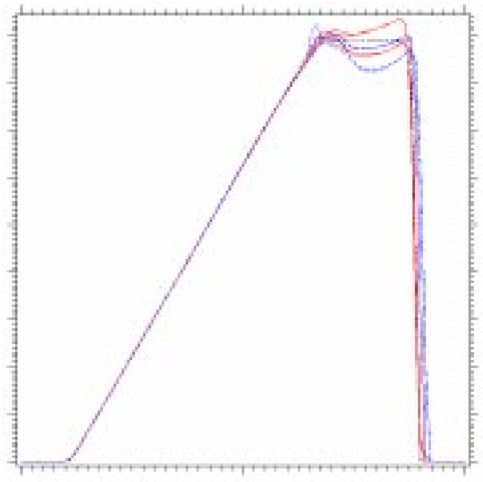
Figure 3. Velocity vs. position

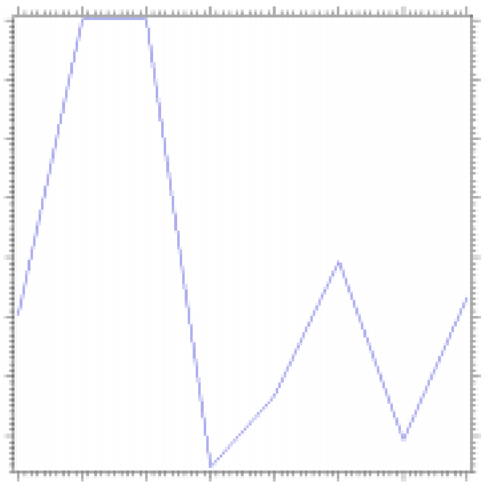
Figure 4. Rate vs. approximation

We also plot the normalized run time against the sequence of approximations to illustrate the relatively low cost to be paid to use an exact Riemann solver. Fig. 4 shows the results. The fastest run was only 7% faster than the slowest. In our implementation the Riemann solves take about half of the hydro time. The other half split between the characteristic tracing and the state updates. Also for the Mach 1.7 shock tube there were an average of 1.01 iterations per solve with a maximum of 6 and in the Mach 80 shock tube there were an average of 1.15 iterations per solve with a maximum of 12. These facts account for the relatively low expense of doing exact Riemann solves.

To help visualize the differences in the approximations we had Odyssey edit the contact pressure (Pc) and velocity (Uc), as well as the density, pressure, and velocity at the cell boundary at the half time step for each Riemann problem solves. In Fig. 5 we show a scatter plot of the cell edge density versus the 2d domain Pc x Uc. The points fall on definite tracks. We plotted data from three of the runs above. What we see is that all approximations agree at the high and low end of the scale for Pc and at the low end of the scale for Uc. So restricting the domain to largely exclude these regions we see the big differences in the remaining data. Each data set is plotted with a different palette consisting of shades of red, green, or blue. The darker color is farther away and the lighter color is closer along the view line of sight. Density (labelled Z) is the height above the Pc-Uc plane (labelled as X and Y respectively).

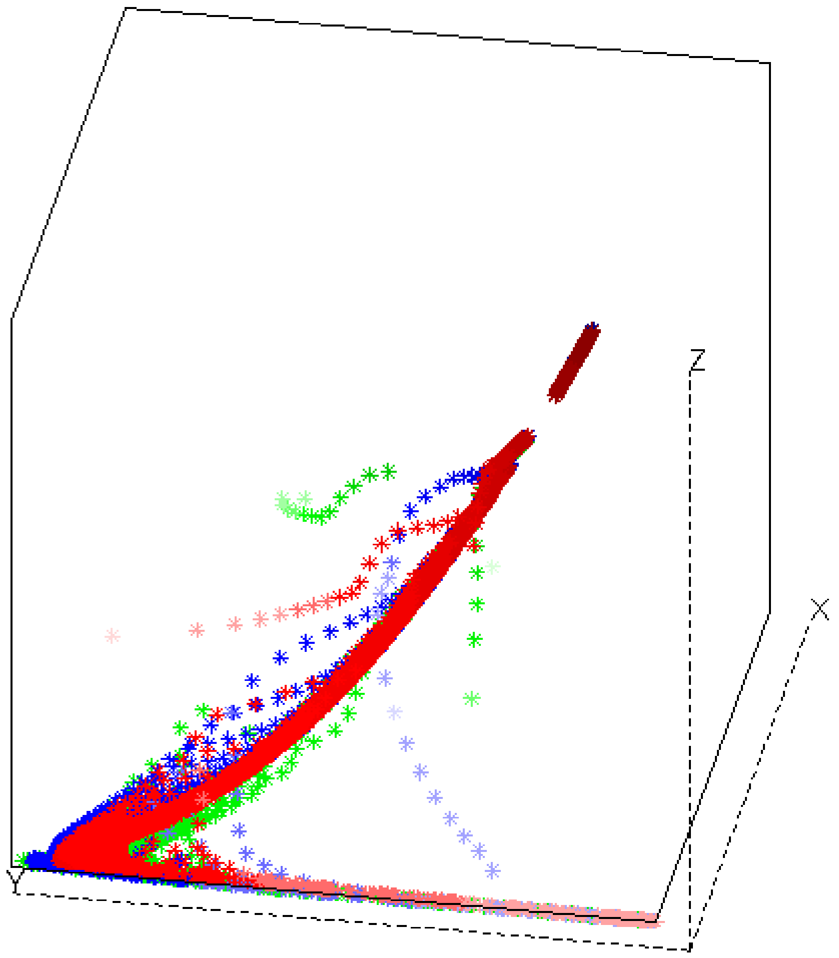

Figure 5. Density vs. Pc-Uc for three approximations

Characteristic Solutions

A shock-capturing technique such as Godunov hydrodynamics based upon a full solution to the Riemann problem at each cell edge at each time step is necessary for flows which contain strong discontinuities. In that standard technique, upwind characteristic tracing is done to each edge, j, of each cell to obtain field quantities at time n+1/2 from the cell values at time n. These values form the initial conditions for a Riemann problem to be solved, usually approximately, at each edge j. The edge values of the field quantities thus obtained determine the fluxes used during the advection step. For regions of sufficiently smooth flow, however, it is not necessary to solve the full Riemann problem.

We outline a method here which takes a more direct route to obtaining the half-time values of the advected edge quantities. Although this method is applicable in any number of dimensions, we discuss the two dimensional formulation for concreteness. A direct solution to the two-dimensional quasi-linear Euler equations

$$\frac{\partial Q}{\partial t} = -A\frac{\partial Q}{\partial x} \tag{1}$$

where

$$Q = \begin{bmatrix} \rho \\ u \\ v \\ P \\ E \end{bmatrix} \qquad A = \begin{bmatrix} u & \rho & 0 & 0 & 0 \\ 0 & u & 0 & \frac{1}{\rho} & 0 \\ 0 & 0 & u & 0 & 0 \\ 0 & c^2\rho & 0 & u & 0 \\ 0 & H & 0 & 0 & u \end{bmatrix}$$

is possible. Here, ρ is the density, u and v are the x and y velocities, respectively, P is the pressure, E is the internal energy, c is the speed of sound and H is the enthalpy. The values at edge j and time n+1/2 are obtained by expanding about the values at time n:

$$\begin{aligned} Q_j^{n+\frac{1}{2}} &\sim Q_j^n + \frac{\Delta t}{2}\frac{\partial Q}{\partial t} \\ &= Q_j^n - \frac{\Delta t}{2}A\frac{dQ}{dx} \end{aligned} \tag{2}$$

This allows us to use characteristic solutions to directly calculate the values of the advected field quantities.

We tested this algorithm on the Mach 80 shock, with the results shown in Figs. 6-8. The analytic solution is the black dotted line, and the code result is in red. In this 90 cell problem, the Riemann solution has been used for a region of four cells at the contact discontinuity and a region of five cells at the shock. The characteristic solution has been used for the remainder of the cells. These results are comparable with those obtained by using the full Riemann solution over the entire domain (see Figs 1-3), and show that the algorithm is stable.

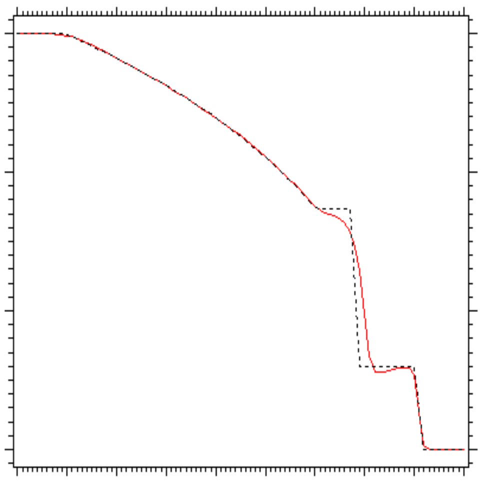
Figure 6. Log of density vs. position

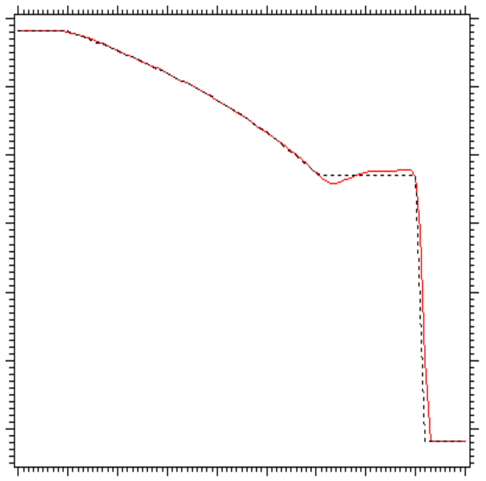
Figure 7. Log of pressure vs. position

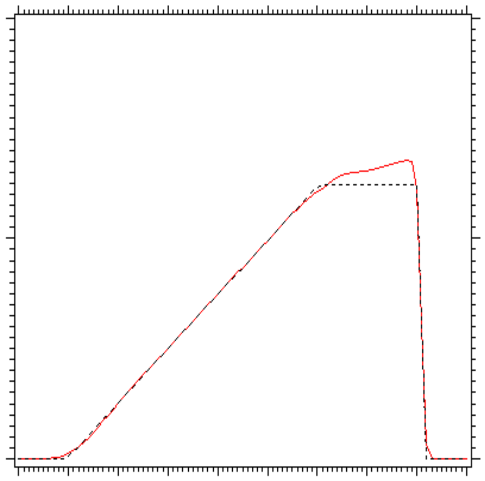
Figure 8. Velocity vs. position

CE/SE Package

There are two approaches to obtaining genuinely unsplit shock-capturing hydrodynamics algorithms which hold sufficient promise. The first is the Conservation Element and Solution Element (CE/SE) Method (Chang, 1995). The second approach is the Riemann Invariant Manifold (RIM) technique (Papalexandris, et. al, 1997). We will not be concerned any further here with the RIM approach.

Beside a completely unsplit approach to solving conservation laws, the CE/SE method also features a superior dissipation treatment that adds dissipation in a conservative fashion. Moreover, this method allows the amount of dissipation to be reduced significantly, while still avoiding ringing in the solution at the shock front.

We summarize here some of the results of a comparison between the directionally-split Godunov solver and the unsplit CE/SE solver. The example shown here employs a large amount of dissipation. For lower amounts of dissipation, and a more complete discussion of these issues, see Cook (1998) in these proceedings.

A problem that we examined was a 2d cartesian concentric, inward flow followed by a bounce and subsequent outward expansion. We would expect the CE/SE method to do a better job of maintaining symmetry because it transports the spatial derivatives of the state variables, hence its representation of the flow has knowledge of the curvature of the flow field which the directionally split solve lacks.

In the following two figures we see contour plots of the density for the two simulations. Figs. 9 and 10 each contain a set of contours from the inward flow phase and one from the outward flow phase.

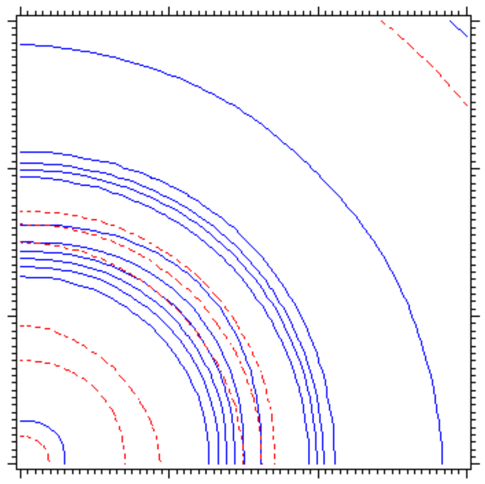
Figure 9. CE/SE result dotted red in/blue out

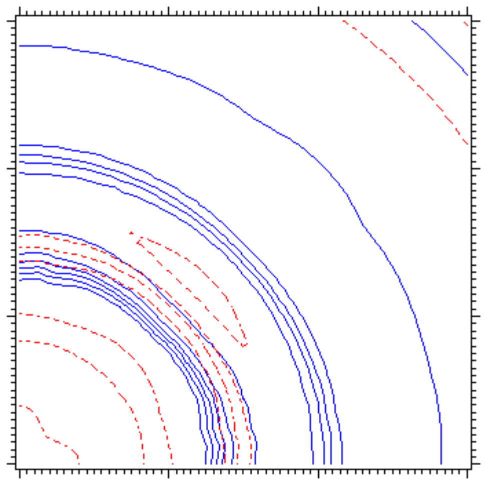
Figure 10. Godunov result dotted red in/blue out

Note the symmetry of the CE/SE result and the corresponding lack of symmetry in the Godunov result near the origin.

Diffusion

At present, an ADI discretization is the primary approach to flux-limited diffusion calculations in Odyssey. ADI is a second-order accurate discretization in time. In space, the Odyssey discretization is second-order accurate away from coarse-fine boundaries and first-order accurate adjacent to coarse-fine boundaries. Below are shown single-level multipatch calculations (Figs. 11 - 13) and one two-level calculation (Fig. 14). Boundary conditions for Figs. 11 and 12: Neumann top and bottom, Dirichlet left and right. Boundary conditions for Figs. 13 and 14: Neumann on all four sides.

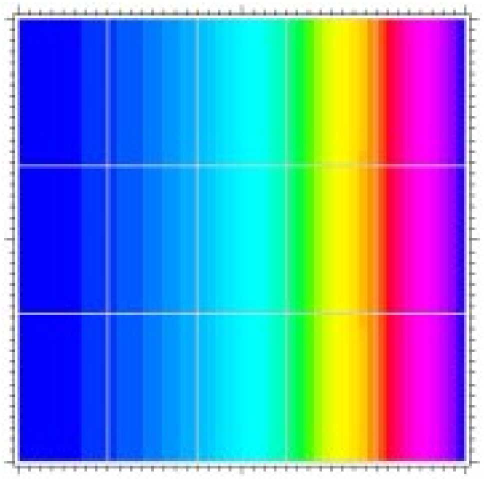

Figure 11. Single level multi-patch early time

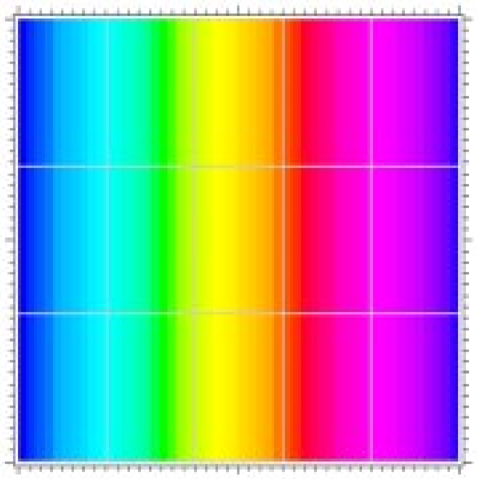

Figure 12. Near steady state

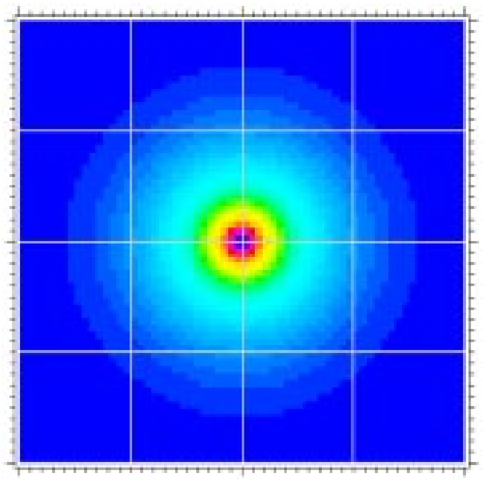

Figure 13. Diffusion with point source

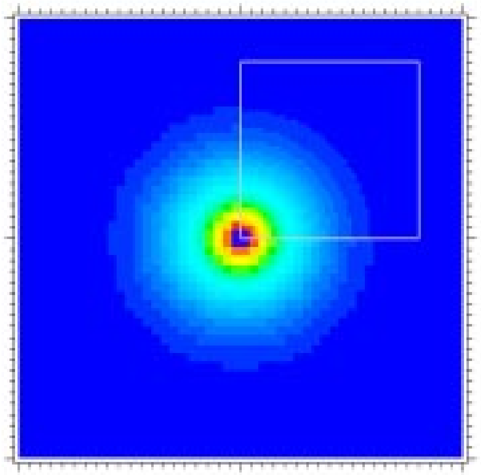

Figure 14. Point source with refinement

Patch boundaries are delineated by gray lines. The diffused unknown is processed in each patch independently of other patches, to the extent possible. This facilitates parallel implementation of the algorithm, for which near-optimal *NlogN* scalability has been empirically demonstrated for the steady state solution (Lambert et al 1997). In the two-level case, the upper right gray box encloses a region refined by a factor of two in each direction. Isotropy of point source diffusion is observed, and in all cases the calculation is relatively seamless across patch boundaries. Such behavior of an ADI discretization on a locally refined grid was just recently brought to our attention (Lambert 1998). ADI has been implemented in 1, 2, and 3 dimensional cartesian, cylindrical, and spherical coordinates. While the point source calculation with refinement (Fig. 14) is based on an inefficient proof-of-principle implementation, fully parallel multilevel ADI-based diffusion calculations are close at hand.

In Fig. 15 we show a simulation in two-dimensional spherical coordinates. The region beyond a fixed distance from the polar axis is given a low conductivity. Inside that low conductivity region is another of much higher conductivity, forming a conductive pipe. A cylindrical disk of higher temperature is then placed inside the pipe, and the simulation proceeds. Notice the planar nature of the diffusion front, not aligned with any coordinate surfaces, even after passing through the origin. Neumann boundary conditions are enforced at maximum radius.

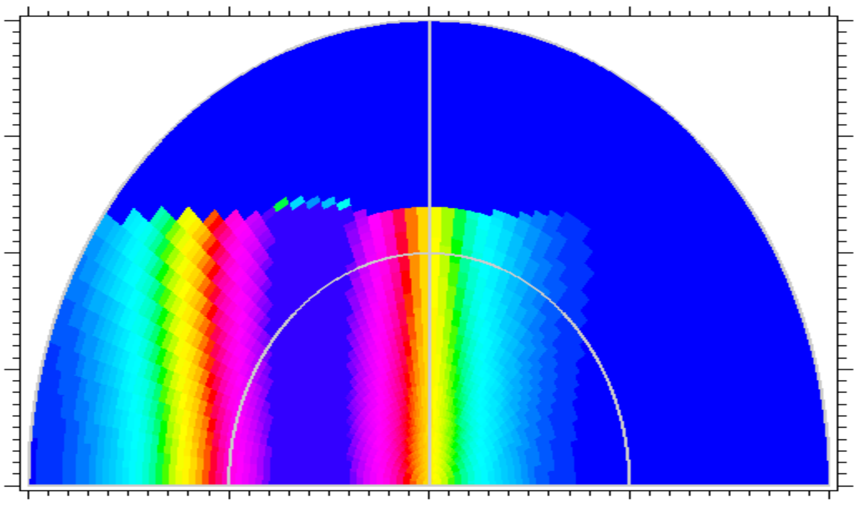

Figure 15. Diffusion from cylindrical slab into a pipe. Simulation done in spherical coordinates.

ECB Treatment of Material Interfaces in Diffusion

We are currently incorporating a new model for material interfaces into Odyssey's diffusion solver. This method, known as Embedded Curved Boundaries (ECB), has been used very successfully for the solution of elliptic systems (Hewett, 1997) using Dynamic ADI, and translates easily to the parabolic systems of diffusion. The method consists of translating arbitrarily shaped boundaries into a set of Piecewise Linear Boundaries (PLBs). Boundary values on these PLBs are then easily incorporated into the finite difference equations for the Laplace or diffusion operator. Fig. 16 shows a circular boundary (in black) along with its PLB representation (in red), and the points that enter into the finite differences. A traditional stairstep model for a Dirichlet boundary condition would simply use the boundary value φ_B in place of the quantity φ_{i-1} in the finite difference operator at the φ_i point. ECB instead uses the value φ_B along with the known distance $f\Delta x$ to write a new second-order-accurate finite difference operator. This is a simple idea; the work of course is in building machinery to efficiently and accurately translate the interfaces into ECB-style PLBs. This has reached a state of considerable maturity for the solution of elliptic equations. Dirichlet and Neumann boundaries have been treated, as well as diffusivity boundary conditions between materials (Hewett and Kueny, 1998) and extension to locally refined meshes (Kueny and Hewett, 1998).

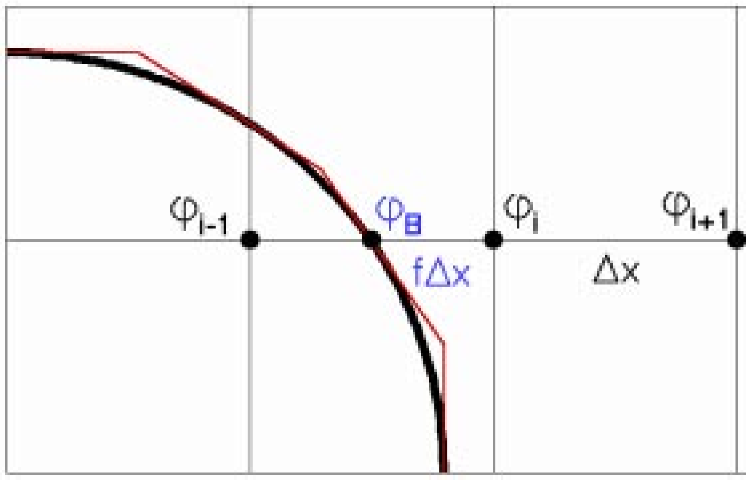

Figure 16. Piecewise linear representation of curved boundary

A comparison of stairstep and ECB boundaries for Laplace's equation in a cylindrical cavity is shown in Figs. 17 and 18. Both were solved on a 20 x 20 grid in cartesian coordinates; the error in the solution for the stairstep model was 34 times larger than for ECB. The difference in solutions is most obvious near the inner boundary surface. The stairstep model required a grid more than 32 times finer in each direction in order to match the accuracy of the ECB solution shown here. For elliptic equations there is very little runtime penalty for this. In fact, ECB in many cases seems to result in faster convergence, possibly due to less spurious fine structure to contend with as compared to the stairstep boundary model.

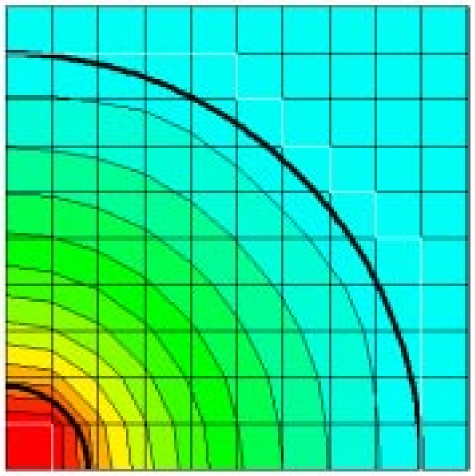

Figure 17. Stair step computation

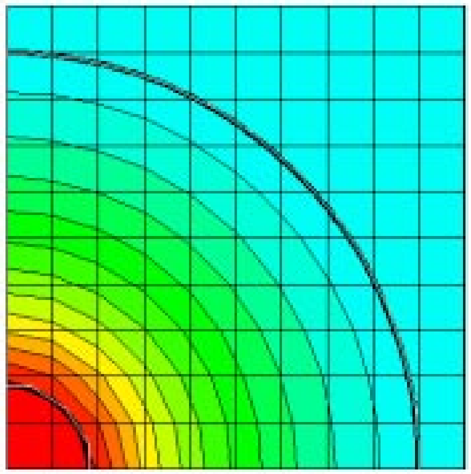

Figure 18. ECB computation

The first test case for ECB in Odyssey’s diffusion package is a configuration of constant temperature sources identical to that of Figs. 17 and 18, whose steady state temperature distribution will be the same as the solution seen there. The PLBs for this case are provided by the material interfaces reconstructed from the volume fractions after the hydrodynamics solve. Each cell with an interface through it contains a vertex as part of the reconstructed material interface; the PLBs are simply taken to be the lines connecting these vertices (Fig. 19). At points where these lines pass directly between cell centers, the finite differences for the diffusion equation are adjusted to take account of the interface.

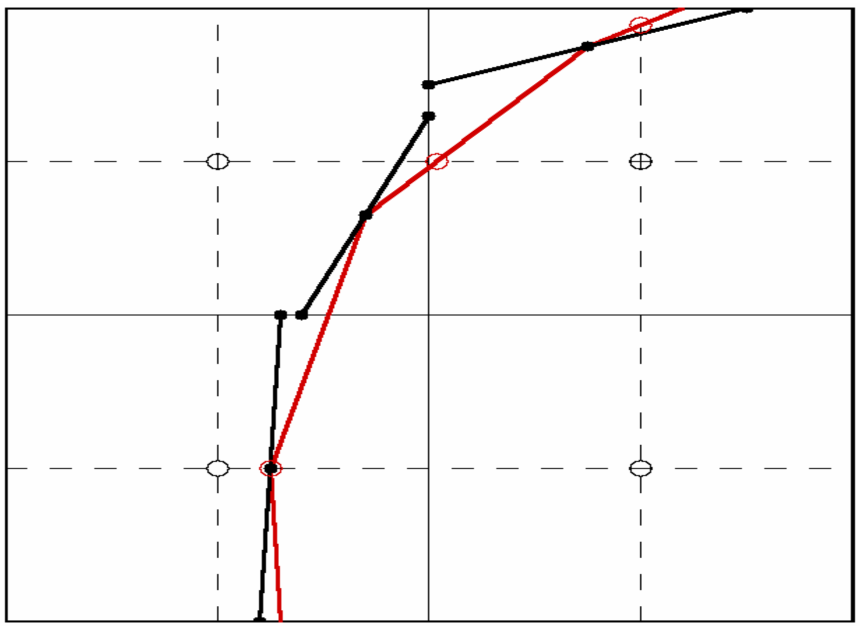

Figure19. Construction of PLB (red) from material interfaces (black)

Parallel Implementation and Performance

In Odyssey we have two approaches to parallelism. These were dictated by ease of implementation in order to meet milestones for the code. The outcome is that we see clear benefits of one over the other. The first approach is to use threads (SMP parallelism) to process all grids at a single level in parallel. Amdahl’s Law instructs us that there will be a significant amount of serial work accompanying the parallel work in this approach. The second, complementary, approach is to divide the physical problem domain into local domains for each node used in the simulation. Each node then runs its own hierarchy and communicates boundary information to its neighbors. There is almost no serial work here and each node is free to use threads to process grids in its own hierarchy. This lets Odyssey combine both SMP and distributed parallelism (MPI).

The one and two dimensional graphics in Odyssey is also done in parallel. In the distributed mode each node communicates its domain size to a specified node which then computes the subregion of a pixel map. Each node then renders its data into a pixel map of its assigned size. The pixel maps are then sent to the specified node where they are assembled into a single pixel map then dumped out to the desired medium (screen window, PostScript file, JPEG file, CGM file, or MPEG file). Each plot request is assigned to a different node to spread out the work involved in assembling the pixel map and writing it out. This machinery was incorporated into PACT (Portable Application Toolkit) which is a set of tools used to facilitate portable simulation codes and is available via anonymous ftp at west.llnl.gov.

In the SMP mode, each grid’s data is used to compute the desired edit data for its part. A list of mappings is thus made and is then serially rendered then sent to the output device. Again this strategy allows both distributed and SMP parallelism to coexist and cooperate.

Fig. 20 is a plot of the speed-up for a given number of CPU’s versus the number of CPU’s. The speed-up for n CPU’s is the time required to run the problem with one cpu divided by the time required with n CPU’s. The maximum possible speed-up with n CPU’s is n. Curve A shows this theoretical maximum. Curve B shows the results measured using a Message Passing Interface (MPI) only (no pthreads) version of Odyssey to run a 3-D shock tube problem. The number of CPU’s varies logarithmically from 1 to 512, thus the ten data points are equally spaced on the horizontal log scale. The code was run on an IBM SP2 using all CPU’s on each node as separate processes.

Fig. 20 shows that Odyssey achieves excellent distributed parallel scaling. The measured speed-up is quite close to the theoretical maximum until near the maximum number of CPU’s, here communication costs are greatest. Imagine slicing a hotdog perpendicular to its long axis into 512 pieces. Each of those slices corresponds to a subdomain in the problem and each slice must communicate with its neighbors. The communication overhead therefore increases at least as fast as the number of slices. The zoning for the problem was chosen so that 512 is the maximum number of subdomains possible and therefore the communication costs are maximized there. By adjusting the zoning, it is possible to avoid the region of maximum overhead and show speed-ups within a few percent of the theoretical maximum.

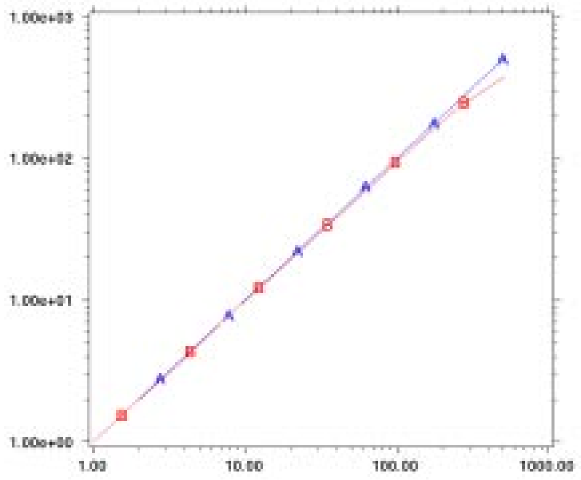

Figure 20. Speed-Up vs. # CPU's

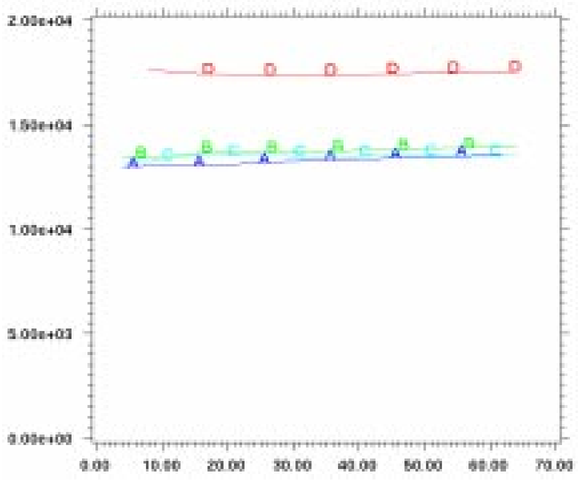

Figure 21. Wall Clock Time x # CPU's vs. #CPU's

Fig. 21 shows the time required to run another 3-D shock tube problem times the number of CPU's used (n) versus n for four different cases of code configuration and pattern of cpu use. Data was collected for n = 4, 8, 16, 32, and 64. Curve A data were collected using an MPI only (no pthreads) version of Odyssey. Data represented by curves B, C, and D were collected using an MPI and pthread configured version of the code. The problem was run on an IBM SP2 having four CPU's per node. For curve B all four CPU's on each node were used as separate processes. For curve C only one cpu per node was used. (Three were wasted.) In curve D one process with four threads was used on each node.

Because the wall clock times were multiplied by n, horizontal lines would indicate speed-ups proportional to n. As you can see all four curves are very nearly horizontal, again indicating excellent scaling. Note that when threads were used (curve D) performance was significantly degraded relative to other cases -- run times increased by about one third. The other three cases produced results quite similar to one another. This means that a single version of Odyssey configured to support both MPI and pthreads can be used without significant lost of performance (as long as threads aren't actually used). It also means that there is no great incentive to waste CPU's (curve A). This is a most welcome result since we all want to be good citizens!

Fig. 22 shows speed-up versus number of threads where the number of CPU's is fixed at 8 (curve A), 4 (curve B), and 2 (curve C). For example, curve A has data points for 8 processes with 1 thread each, 4 processes with 2 threads each, 2 processes with 4 threads each, and 1 process with 8 threads. Once again the results are for a 3-D shock tube problem. The problem was run on a DEC Alpha host with 8 CPU's.

Basically, Fig. 22 demonstrates that for the current Odyssey it is always better, performance wise, to use more processes and fewer threads for a given number of CPU's. The results using threads are even worse on the DEC Alpha than on the IBM SP2. Although the speed-ups increase with the number of CPU's, increasing the number of threads beyond about 4 gives very little increase in speed-up. (Compare the speed-ups for the rightmost point on each curve.)

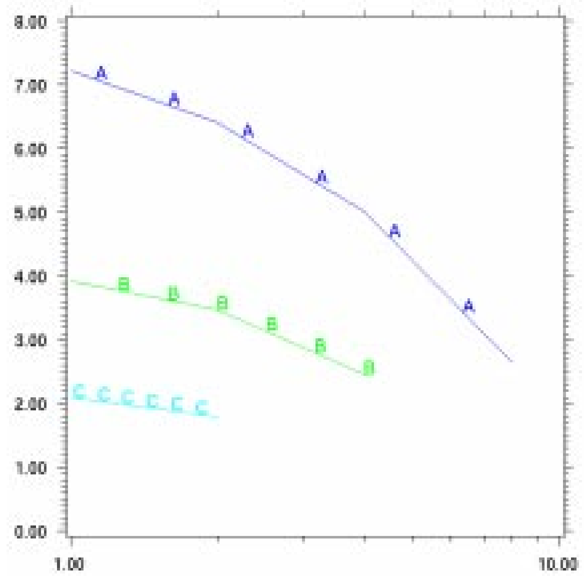

Figure 22. Speed-Up vs. # Threads

Odyssey’s distributed parallel performance and scaling are excellent. Where communication overhead noticeably affects performance, we have demonstrated that the costs are independent of message length and thus latency dominates communication costs in Odyssey on the IBM SP2. (This was accomplished by increasing the message buffer size and noting that performance was unaffected.) The latency is likely to decrease on future IBM systems due to both hardware and software improvements.

The fact that using pthreads (shared memory parallel) gives much worse performance than using all MPI tasks (distributed parallel) is not surprising. Dividing the problem domain into subdomains and running each subdomain on its own cpu means that almost 100% of the code is parallelized. But in the shared memory case only a fraction of the source code lies within threaded regions. Most of the cycles are expended in these threaded regions, but Amdahl’s Law nevertheless takes its toll. At some point we may extend the threaded regions and improve SMP performance. This would represent a considerable effort with no guarantee of performance significantly exceeding the current MPI only results.

Currently Odyssey allows for creating subdomains by specifying the axis to be sliced and how many slices. Future plans include generalizing this process to support slicing all or any combination of coordinate axes. This will allow a larger number of subdomains to be created and thus a greater number of CPU’s to work on a given problem.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.

References

van Leer, B. “Towards the Ultimate Conservative Difference Scheme. V. A Second-Order Sequel to Godunov’s Method”, JCP **32**, 101-136, (1979).

Colella, P., and Glaz, H., “Efficient Solution Algorithms for the Riemann Problem for Real Gases”, JCP **59**, 264-289 (1985).

Miller, G. and Puckett, E., “A High-Order Godunov Method for Multiple Condensed Phases”, JCP **128**, 134-164 (1996).

Chang, S. C., “The Method of Space-Time Conservation Element and Solution Element - A New Approach for Solving the Navier-Stokes and Euler Equations,” JCP **119**, 295, (1995).

Papalexandris, M.V., Leonard, A., and Dimotakis, P.E., “Unsplit Schemes for Hyperbolic Conservation-Laws with Source Terms in One Space Dimension,” JCP **134**, 31, (1997).

Cook, G.O., Jr., “The Effects of Operator Splitting in Computing Curved Shocks,” Proceedings of the 1998 NECDC.

Lambert, M., “ADI Methods for the Poisson Equation on Locally Refined Orthogonal Composite Grids”, Proc. Copper Mountain Conference on Iterative Methods, March 30 - April 3, 1998, Copper Mountain, CO.

Lambert, M., Rodrigue, G, and Hewett, D., “A Parallel DSDADI Method for Solution of the Steady State Diffusion Equation”, Parallel Computing **23**, 2041-2065, (1997).

Hewett, D., “The Embedded Curved Boundary Method for Orthogonal Simulation Meshes”, JCP 138, 585-616, (1997).

Hewett, D. and Kueny, C. “The Dielectric Boundary Condition for the Embedded Curved Boundary Method”., Proc. 16th Intl Conference on the Numerical Simulation of Plasmas, Feb. 10-12, 1998, Goleta, CA.

Kueny, C.and Hewett, D., “Emedded Curved Boundaries and Adaptive Mesh Refinement”, Proc. 16th Intl Conference on the Numerical Simulation of Plasmas, Feb. 10-12, 1998, Goleta, CA.