LA-UR-13-20949
Approved for public release; distribution is unlimited.

Title: Simulating Turbulent Mixing from Richtmyer-Meshkov and Rayleigh-Taylor Instabilities in Converging Geometries using Moving Cartesian Grids

Author(s): Woodward, Paul R.
Jayayaraj, J.
Lin, p.-H.
Knox, Mike
Porter, David H.
Fryer, Christopher L.
Dimonte, Guy
Joggerst, Candace C.
Rockefeller, Gabriel M.
Dai, William W.
Kares, Robert J.
Thomas, Vincent A.

Intended for: DECDC2012, 2012-10-22/2012-10-26 (Livermore, California, United States)

Simulating Turbulent Mixing from Richtmyer-Meshkov and Rayleigh-Taylor Instabilities in Converging Geometries using Moving Cartesian Grids

[1,2] P. R. Woodward, [1] J. Jayaraj, [1] P.-H. Lin, [1] M. Knox, [1] D. H. Porter, [2]C. L. Fryer, [2] G. Dimonte, [2] C. Joggerst, [2] G.M. Rockefeller, [2]W. Dai, [2]R. J. Kares, [2]V. A. Thomas

[1] *University of Minnesota, Minneapolis, Minnesota 55455*

[2]*Los Alamos National Laboratory, Los Alamos, NM*

Abstract

The feasibility of accurate simulations of Richtmyer-Meshkov and Rayleigh-Taylor instabilities in idealized test problems motivated by inertial confinement fusion (ICF) applications has been studied using a hydrodynamics code based on the Piecewise-Parabolic Method (PPM) and the Piecewise-Parabolic Boltzmann (PPB) scheme for multifluid volume fraction advection. Test problems producing radial convergence factors of about 6, as measured with the 50% mixing contour, were adapted from those studied by Youngs. Gamma-law gases with gamma values of 5/3 and perfectly cylindrical and spherical outer boundary conditions were employed in both 2-D and 3-D studies. Convergence under grid refinement and symmetry preservation tests indicate that at sufficiently high grid resolutions accurate numerical treatments of these idealized, hydrodynamics-only problems are possible on moving Cartesian meshes. The simulations are made practical as well by code restructuring techniques targeting today's multi- and many-core computing devices. Results are presented for spherical cases with two- and three-mode initial perturbations using grids of 4160^3 and 10560^3 cells computed on the NSF's Blue Waters sustained petascale system at NCSA at the University of Illinois.

Background and Motivation

Accurately simulating the radial compression by an order of magnitude or more of an inertial confinement fusion (ICF) capsule and the lighter fuel it confines is challenging because of the multiple hydrodynamic instabilities that come into play at the capsule surfaces. Numerical schemes that exploit the simplicity of a Cartesian mesh can benefit from relative coding ease as well as a potential for very high execution speed. However, the natural topology of the mesh does not reflect the symmetries of the ICF problem. Numerical schemes designed to address this defect by using grids that capture the spherical symmetry of the initial state more closely have a different set of challenges to deal with in producing accurate simulations at affordable cost. In this work, we explore the potential for the Cartesian approach to overcome its natural limitations on this type of problem by the use of very fine grids. This approach is made practical by structuring the code implementation to exploit the grid topology to produce very high performance on today's multi- and many-core computing devices. The grid is also made to move radially inward in a homologous fashion, in order to capture the most basic aspect of the problem, if not its spherical topology. In this initial study, no adaptive mesh refinement (AMR) is introduced. Thus our study can be understood as providing an underestimate of the potential for Cartesian Eulerian approaches to accurately simulate ICF hydrodynamics in the simplified, idealized test problems described below. This study is a follow-on to a 2-D study presented in [1].

We find that the various grid imprint effects familiar to investigators attempting to apply Cartesian mesh codes to the study of problems with unstable multifluid interfaces in a spherical or cylindrical basic geometry can be almost completely eliminated. There are two key points that qualify this conclusion. First, one must have a very fine grid, and, second, one must introduce at unstable multifluid interfaces very small amplitude and very high frequency perturbations in the initial state. These two requirements work together, because of course it is impossible to accurately follow the necessary small and high frequency disturbances without a fine grid. The fundamental role played by the fine grid demands that the simulation code must achieve a high fraction of the computing system's peak performance. We discuss a set of code design and implementation practices that we believe can assure this. For our PPM code, we achieve 12% of the peak performance of the processors we use on NSF's Blue Waters petascale computing system when running at scale in 32-bit mode (which we find entirely adequate) and sustaining 1.5 Pflop/s overall performance. On Intel-based CPU processor cores, the performance this code achieves is between 40% and 80% higher as a fraction of the peak performance, depending upon the processor type. We set out in this article how we achieve this code performance, because we believe that it is critical in enabling accurate simulations of spherical problems with Cartesian meshes. We are convinced that our methods can be adapted to other codes embodying other numerical approaches, although we have not yet demonstrated that by performing such an adaptation.

In this article, we present comparisons of results from two different versions of an ICF test problem. One does and one does not include the very small and high frequency initial perturbation at the unstable multifluid interfaces. Our intent is to illustrate the dramatic impact of such additional perturbations upon the simulation results. These arise only because the multifluid interfaces are physically unstable. We introduce additional disturbances just below the limit of visual detectability in the initial state. Consequently, the resulting behavior is, we believe, behavior whose appearance one would be unlikely to rule out in a real laboratory experiment. We argue that adding such disturbances in the initial state is thus good practice. It gives us a means of more accurately determining the consequences of the interface instability in the real world. Although in a world of simulation we might rule out such disturbances, in the real world this would most likely be impossible.

We discuss our numerical methods briefly, since there is of course also some dependence of our results upon these details. Although we have tried to use very accurate numerical methods, we do not believe that it is the accuracy of our particular methods that delivers the greatest benefits in simulating our ICF flows. We have studied that issue in 2D, and our results are reported in [1]. The present study can be viewed as an extension to 3D of that earlier work using our PPM code. Due to computational cost in 3D, we have not performed our test problems here with more than just one code.

In this study, we have not included any of the small physical effects, such as viscosity, heat conduction, or surface tension (not present at our gas-gas interfaces), that might play a role in sufficiently small physical systems. Our focus is instead on the capability of the numerical methods to produce accurate results free from obvious imprints of the Cartesian grid that might arise from purely numerical effects in our ICF problems. Once that capability is established, it is clear that the regularizing terms of these small physical effects could be added to the governing equations and approximated by the numerical method. Once added, in problems that require them, these effects can only, in our experience, make the simulation easier to perform with high accuracy. Thus we believe that if we can eliminate the issue of grid imprint effects in the present study, we need not be concerned about it in problems where small physical effects tend to regularize the solution still further.

ICF Hydrodynamics Test Problems

We have adapted our ICF test problems from that introduced by Youngs in 2008 [2]. Work with a similar ICF test problem has also been reported by Thomas and Kares [42]. A spherical shell of gas with gamma 5/3 and density unity extends from radius 1.0 to 1.2. This shell and the uniform, gamma 5/3 gas of density 0.01 it encloses are initially at a constant pressure of 0.1. Outside the initial spherical shell is a uniform gas with gamma 5/3, density 0.1, and pressure 160. The outer gas extends initially to a radius of 1.4, beyond which we apply an outer boundary condition. Thus initially:

$r < 1.0$: $\rho = 0.01,\ p = 0.1,\ u = 0$ $\qquad$ $1.0 < r < 1.2$: $\rho = 1.0,\ p = 0.1,\ u = 0$

$1.2 < r < 1.4$: $\rho = 0.1,\ p = 160,\ u = 0$ $\qquad$ $1.4 < r$: $\rho = 0.1,\ p = 160,\ u_r = -9.75\, r$

In all these regions, the gas has a gamma-law equation of state with $\gamma = 5/3$. Our problem domain is $-1.5 < x, y, z < 1.5$, and u_r is the radial component of velocity.

At each radius where the initial state changes suddenly, we smooth this transition out over a smearing distance δ. If s_{in} and s_{out} are fluid state variables inside and outside of the transition zone in radius in the initial state, then within this zone the state variable value is given by: $s = (1-f)\, s_{in} + f\, s_{out}$, where the blending fraction f is given by: $f = [\, 1 + \sin(\pi(r-r_t)/\delta)] / 2$ when $|(r-r_t)| < \delta/2$. Here r_t is the transition radius. Our intent is to choose δ so that perturbations of the transition due to the numerical representation on a grid are reduced. Thus δ scales with the grid cell size, Δx. In the results reported here, we use $\delta = 12\, \Delta x$. This smearing at the computational microscale in the initial state is especially important at the inner and outer surfaces of the dense shell, where our PPB multifluid fractional volume advection technique, described below and in [3-6], is applied. PPB uses 10 moments of the subcell distribution of the fractional volume of the dense shell fluid to describe the transition from shell to inner or outer gas as a smooth transition. This numerical treatment of the unstable contact discontinuities in the ICF test problem complements the more standard smearing of shock transitions to reduce unphysical oscillations in the state behind the shock (see for example the discussion in [7] for this procedure in PPM). This sort of numerical smearing at the grid scale is essential if we aspire to produce results that are essentially independent of our choice of grid topology. We note that both surfaces of the dense shell will be struck by very strong shocks before anything else happens to them. The shocks will compress our initial structures, smeared over 12 grid cell widths, into much thinner structures, smeared only over 3 grid cell widths.

Each grid line, in x, y, or z, has a constant inward velocity given by its initial location, and this velocity is constant in time for that grid line. This grid line velocity is given by: $u_{gx} = -\,9.75\, x_{initial}$ with similar expressions for y and z.

Outside the radius on our moving grid that was initially at a radius of 1.4, i.e. outside $r_{bdry} = 1.4\,(1 - 9.75\, t)$, we impose an inward flow at the local grid velocity with a density of 0.1 and pressure $p = 160\,(0.1/160)^{10\,(t\,-\,0.05)}$, with $p = 160$ when $t < 0.05$. Because nearly the entire compression occurs before time 0.05, this ultimate reduction of the outer pressure is unimportant. The initial strong shock enters the enclosed light gas at around time 0.015, triggering the Richtmyer-Meshkov instability (see Fig. 8), and it hits the origin around time 0.0342. Strong deceleration of the dense shell begins a bit before time 0.04, triggering the Rayleigh-Taylor instability of the inner shell surface. With these initial base states and boundary conditions, we have done multiple problems distinguished by the mode numbers and amplitudes of the perturbations applied to the inner and outer surfaces of the initial dense shell.

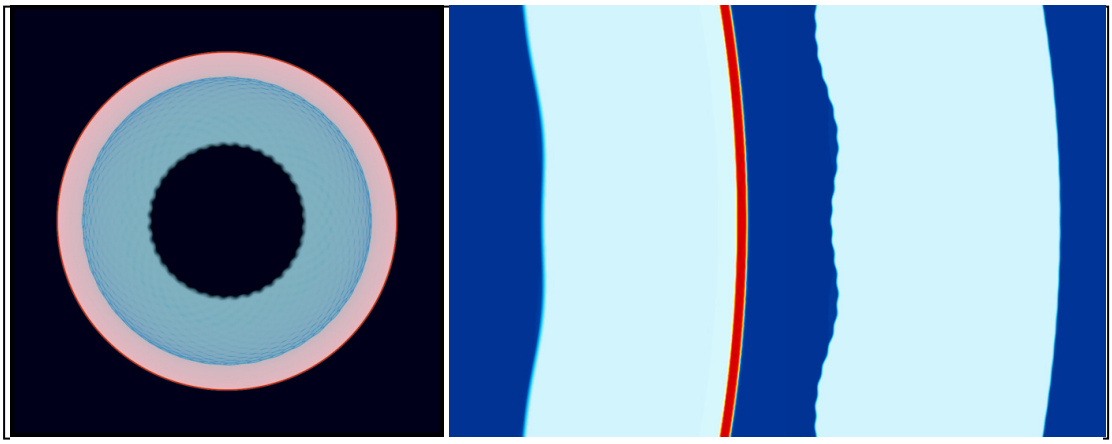

Figure 1a. The density distribution is shown at the left for the ICF test problem #1 at t = 0.0025. The initial disturbance of the inner shell surface, unchanged since t = 0, is evident. A slice through the equator and down through 7/26 of the problem domain is shown. The initial disturbance is thus sliced obliquely at the back, so it is far more easily seen. At the equator, it is clear that this disturbance amplitude is very small. A zoomed-in view of a section of the density distribution in a very thin equatorial slice is shown for Problem 1 at t = 0.0025 (center) and for Problem 2 at t = 0 (right). The very high-order mode is just perceptible on the inner surface for Problem 2. At the outer surface, where it is 5 times smaller, it is essentially imperceptible, as it involves displacements of only 0.55 grid cell widths from a circle 4224 grid cells in radius.

Problem 1: The inner surface of the dense shell is perturbed by displacing the surface according to the sum of 2 spherical harmonic modes, Y_{lm}, where the mode numbers (l,m) are (81,39) and (74,36). Mode (81,39) is given an amplitude at the equator equal to 2% of its azimuthal wavelength at the equator (y = 0), and mode (74,36) is given half this amplitude. This produces a modulation of mode (81,39) with 3 wavelengths at the equator, in each of which precisely 13 wavelengths of the higher mode are enclosed (see Fig. 1a). The behavior of the inner shell surface in each of these 3 modulation lobes should therefore be statistically identical at the equator. Since modes 3 and 39 are not compatible with any natural mode of the Cartesian mesh, numerical artifacts arising from mesh imprints should stand out clearly in the simulation results. In particular, growth of mode 4 at the equator is a clear signal of inadequate mesh resolution. Also, perturbations of the shell surfaces that arise at the points of the compass, where the unstable shell surface crosses grid planes at glancing angles, will also stand out clearly if present. We have therefore chosen our initial disturbance so that we can detect flaws in the numerical treatment at a glance.

Problem 2. The inner surface of the dense shell is perturbed as in problem 1, but now the amplitudes of modes (81,39) and (74,36) at the equator and radius unity are increased to 5% and 2.5% of the azimuthal wavelength of mode (81,39) there. Also, mode (65×11, 31×11) is added at 4.252% of its much smaller azimuthal wavelength at the equator and radius unity. This additional, high-mode perturbation is added to the inner shell surface displacement and also to the displacement of the outer shell surface, at 5 times smaller amplitude, where it is the only applied initial perturbation.

The high-mode perturbation in problem 2 serves two purposes. First, it applies a very high frequency perturbation at nearly a visually imperceptible initial amplitude, and thus serves to generate more realistic behavior. The displacement amplitude is 0.0001567 at the equator, which is so small that we presume that it would be very difficult to detect in a real laboratory experiment. The idea here is that any real experiment in a laboratory could not avoid having such tiny perturbations, although they would not be expected to consist of just a single mode. By putting in just one such high frequency mode, we will be able to judge whether or not the

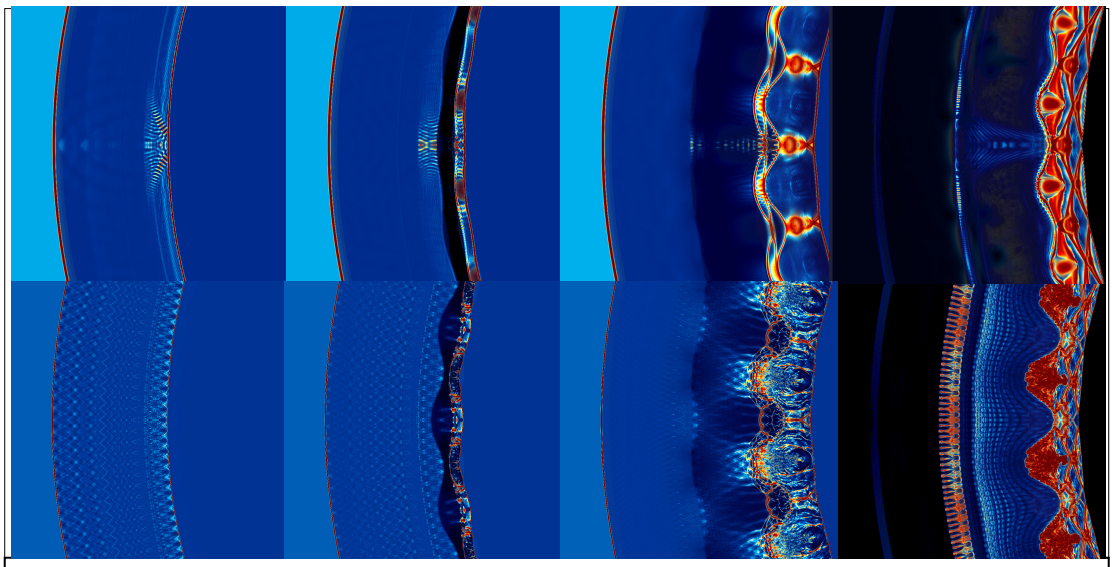

Figure 1b. The negative divergence of velocity is shown at times 0.0125, 0.015, and 0.020 (from left to right) for Problem 1 (top) and for Problem 2 (bottom) near the inner surface of the dense shell in a thin equatorial slice. At the far right, the vorticity magnitude is shown at time 0.020. For Problem 1 spurious sound waves have been introduced near the midplane of the images, because precisely here the thin numerical representation of the strong incident shock is tangent to the planes of the Cartesian grid. The very high-order mode in Problem 2 has introduced a nearly imperceptible signal in the initial state that overwhelms any spurious beat frequencies between the strong shock representation and the grid planes in this problem. Thus Problem 1 has a slight numerical glitch, clearly of very little importance yet still visible, while in Problem 2 such features are entirely absent. At the cost of introducing a high-order mode at an amplitude that would not be detectable in a laboratory experiment, we have essentially eliminated in Problem 2 the grid imprint phenomenon that we observe in Problem 1. The 4160^3 grid of the Problem 1 simulation shown here has 223.4 grid cell widths to cover the distance between the crests of the principal sinusoidal signal in these images, while the 10560^3 grid of the Problem 2 simulation that is shown here uses 567 grid cell widths to span this same distance.

numerical scheme follows the small-scale dynamics correctly, at least in regard to the obvious symmetries of this high-order mode.

The second function served by this high-order mode is to give the simulation something non-trivial to follow at the outer boundary of the dense shell, which is highly unstable during the compression of the shell. The real behavior of this unstable disturbance should overwhelm any behavior that is generated purely by numerical effects, as illustrated in Figure 1b. The character of the spherical harmonics Y_{lm} when m/l is about 1/2 guarantees that the disturbance will very nearly vanish in amplitude in a fairly broad region near each pole. In the polar regions, we will therefore find that our simulations reveal the behavior that would apply in the case where there were essentially no perturbations at all of the spherical inner and outer capsule surfaces. By comparing the simulated behavior in these regions with that at lower latitudes, we will easily see the impact of this tiny initial perturbation of the outer surface, as well as of the impact of the imprint of the inner surface perturbation that is caused by waves communicating that disturbance to the outer shell surface. Such waves can be seen in the right-hand images in Figure 1b just as they are striking the outer surface of the shell.

In Problem 2, we use a perturbation of the inner shell surface that is 2.5 times larger than in Problem 1. This causes more mixing of the dense shell gas with the enclosed lighter gas. We do this in order to obtain a larger turbulent mixing region, because we wish to use the results of the simulation of this problem as a direct numerical simulation (DNS) experiment that can help to validate statistical models of turbulent mixing in large eddy simulations

(LES). The larger initial perturbation ultimately gives us significantly more of the problem domain that is useful for this model validation purpose.

Finally, the high-order mode displacement of the inner shell surface, at 5 times the amplitude used on the much more unstable outer surface, is useful in very slightly breaking symmetries of the other two, lower-order modes. Again, we expect this to produce a more realistic simulation. We can judge the success of this approach by comparing the results with those for Problem 1, where no such high-order perturbation is applied to either the inner or the outer surface of the shell. To be accurately followed, the very small length and time scales that are introduced into this problem by our high-order mode demand a very fine grid. We can therefore gauge the accuracy of the calculation by observing the behavior of this very high frequency mode of very small initial amplitude. If it is treated faithfully, we can then hope that the much larger and longer-wavelength disturbances are treated even more accurately.

The scheme of this work is that Problem 1 exposes both successes and difficulties of the Cartesian approach to this sort of ICF problem, and Problem 2 illustrates a means of resolving these difficulties without giving up the advantages of the Cartesian approach. Our primary focus in this article is on the behavior of the numerical scheme on these demanding problems. Comparison of the simulation results with statistical models of turbulent, compressible mixing will be presented in a later article.

A primary conclusion of this work is to confirm our ability to produce very accurate simulations for these problems at acceptable cost, with few if any problems that need human attention during the course of even the very largest simulations now feasible. Because these simulations are made practical by the rather unusual code implementation techniques that we have devised, these are also described along with the numerical methods. The largest simulation presented here has a grid of over a trillion cells. It was run on the NSF's Blue Waters machine at 1.5 petaflop/s, sustained. These two test problems do not benefit appreciably from adaptive meshes, save for the grid motion that roughly follows the compression of the dense shell. A uniform, moving Cartesian mesh is excellent at capturing all necessary details. A huge advantage of such a mesh is that it lends itself easily to very efficient and accurate computation at extreme scale. It is for this reason that the highly scalable implementation of our PPM code is as important to describe as its embedded numerical algorithms. Using nearly every node of the Blue Waters machine at NCSA, and counting all disk I/O and messaging time against the code performance, our PPM code runs at 12% of the doubled peak performance of this portion of the machine in 32-bit mode. It is this fact, more than any other aspect of the computations presented here, that makes the Cartesian grid approach to ICF problems practical. This performance is achieved in large part through automated code transformations that boost the code performance on the multi-core CPUs of the machine. We therefore briefly outline these code transformations here as well.

Symmetry Preservation

We have designed our ICF test problems to expose to immediate view any symmetry breaking that arises purely from the numerical scheme. In simulating the ICF process, it is especially important that symmetry breaking be physical and not numerical in origin. In order to compress the fuel inside the spherical capsule enough so that it can burn, the near spherical symmetry of the initial state needs to be very accurately maintained, except of course for departures generated by the initial perturbations of the unstable multi-material interfaces.

In this work, we take the view that some very small level of initial disturbances is unavoidable in the real world, even though these disturbances can be avoided in principle in a world of simulation. In our simulations, and in any simulations for that matter, computation with

finite precision and on a grid, however fine and however oriented, will introduce perturbations of the flow with trivial amplitudes and with very high frequencies. These will be amplified by the physical instabilities of the flow. They will ultimately produce a chaotic flow at the microscale in those regions where the flow is unstable. We believe this behavior to be benign, because it simulates real behavior, at least after the chaotic flow is established. In a real experiment, such chaotic flow must also develop, and, we believe, in the same locations. However, the real chaos will develop out of different initial perturbations that are fundamentally unknowable, but whose *details* are equally fundamentally unimportant.

For this reason, we do not attempt to remove every sort of "bad behavior" of the numerical scheme by adding to the problem regularizing terms representing gross overestimates of those processes that regulate the microscale in real flows, but on length and time scales well beyond our ability to capture on our grid. One may envision two sequences of simulations representing each type of numerical approach – both, hopefully, convergent. One sequence would add in regularizing terms sufficient to damp out all questionable numerical behaviors on each grid. It would then progressively refine the grid while simultaneously reducing the size of these regularizing terms until the simulated flow converges in an appropriate statistical sense. The other sequence would use features of the numerical scheme to regularize the computation on each grid. These features of the scheme need not explicitly model any known physical processes, but they must act on progressively finer length and time scales as the grid is progressively refined. They must also allow the simulated flow to converge in the same appropriate statistical sense used in the first convergent sequence.

It is very difficult and costly to demonstrate that both these sequences converge, and that both converge to the same result. An attempt at such a demonstration in a very much simpler problem can be found in [8]. An entire book has been devoted to this issue, to which we refer the curious reader [9]. In this paper, we take the view that the second sequence discussed here is the more rapidly convergent one, although we have not demonstrated this. We also assume that it converges to the "right answer" appropriate to the limit in which all regularizing phenomena operate on scales much smaller than our ability to resolve in a practical simulation. We could of course have bugs in our code, although we have worked tirelessly to root these out. For this reason, we have stated our test problems clearly, so that other investigators can check our results using other codes, which, if they have bugs, must then surely have *other* bugs. If their converged results agree with ours, we may conclude that this common converged result is most likely correct.

The above philosophical ansatz motivates us to distinguish between acceptable and unacceptable perturbations of the flow that are introduced through our numerical treatment. Perturbations that are undeniably small in scale compared to important features of the flow will be judged acceptable if their *immediate* effects are also small and unimportant. Because of the physical phenomenon of turbulence, the ultimate effects of these perturbations will indeed be important; they will lead to the development of chaotic flow at the microscale in the simulation. For the reasons stated earlier, this we judge to be a benign effect, helping our simulation to correspond more closely to the reality of fluid behavior which we all see around us.

We do find some perturbations of numerical origin unacceptable. These have length and time scales that are comparable to those that are important in the developing flow. They have the potential to profoundly influence the outcome of the numerical experiment through the unstable growth of features that cannot represent the action of any conceivable physical process operating statistically on only a microscale. We have designed our test problems 1 and 2 to make such unacceptable flow features of numerical origin immediately recognizable. We will indeed see them, and they will be unmistakable. A major goal of this work is to explore means of either eliminating these numerical artifacts or of rendering their effects

upon the simulated flow unimportant.

Numerical Methods Used

Our PPM code incorporates a version of the original Piecewise-Parabolic Method (PPM) [10,7,11,3] that has been described in [12]. This PPM scheme differs from the original one by elaborately interpolating the Riemann invariant distributions inside each grid cell, then from these distributions it constructs the implied distributions of the physical variables – pressure, density, and velocity. Constraints are applied to the Riemann invariants, and then to the physical variables in order to arrive at distributions of each that are consistent. Our ICF problems are very violent, and therefore some fine points of interpolation procedure described in [12] are ineffective. However, we introduce a special, parabola-based interpolation and advection scheme, the Piecewise-Parabolic Boltzmann (PPB) method, to handle the very difficult task of tracking the mixing fraction of dense shell gas with the background lighter gas both outside and inside the ICF capsule. We thus rely upon the well-known properties of the PPM scheme to handle the gas dynamics of strong shocks interacting with complex flows, while we call upon a far more accurate advection scheme to handle the behavior of the unstable multifluid regions. Results of this combination of numerical techniques applied to less violent, slow-flow, Rayleigh-Taylor problems are reported in [13-15]. Results for such problems in the context of stellar evolution are reported in [5,16,17].

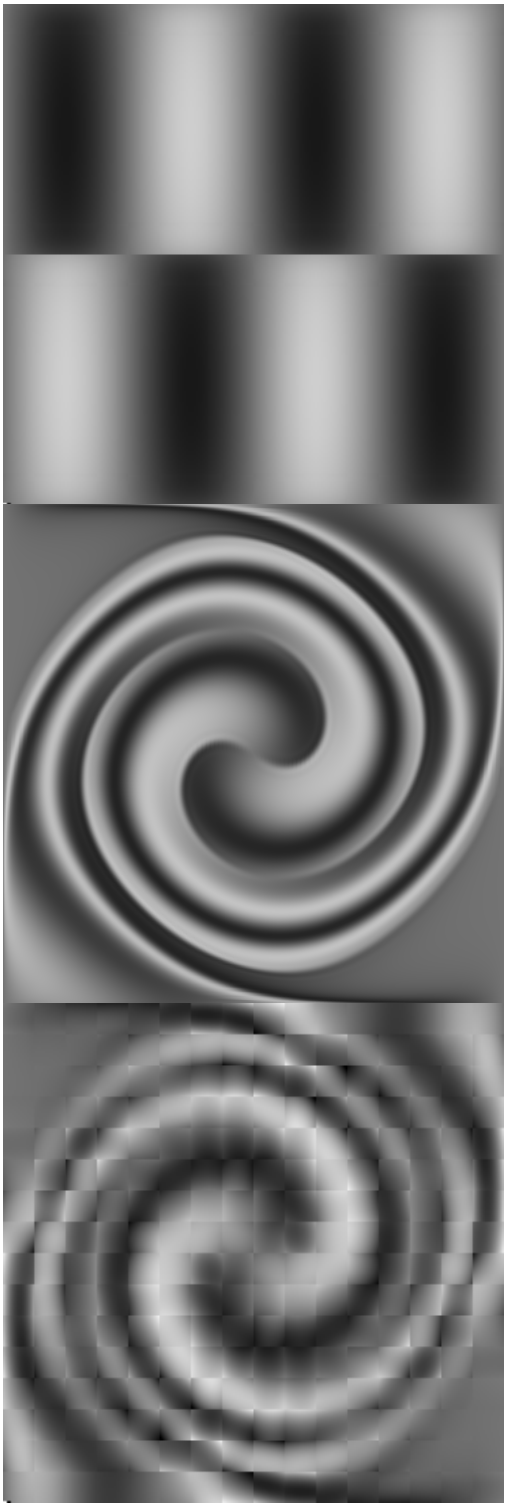

Figure 2. Illustration of the resolving power of PPB (see text).

The Piecewise-Parabolic Boltzmann method (PPB) is derived from van Leer's advection Scheme VI [18]. Van Leer did not develop a constraint technique for his Scheme VI, nor did he extend it to multiple dimensions. Both these additions were made in the early 1980s (see [3]). Our version of PPB also adds improved constraints upon the interpolation parabolae. These recognize that interpolated values of the mixing fraction (the fraction of the local gas volume occupied by the dense shell fluid) at cell interfaces should be 0 or 1 if one of the cells meeting at this interface has 0 or 1 for its cell-averaged mixing fraction. Forcing this value at the interface helps to maintain the thinness of multifluid boundaries. We also demand that the interpolated mixing fraction

remain within the physically allowable range of 0 to 1. No other constraints on the interpolated mixing fraction are applied.

Our version of PPB used in this study is identical to that contained in the XRAGE code release of 2005. It has been described in [13] and in [3-6]. Here we only list a few of the aspects of PPB multifluid advection that bear directly upon the results of this study. In Figure 2, we take from [13] an illustration of the remarkable resolving power of PPB advection in a (2-D) swirling flow of a type encountered in turbulent mixing regions. The top image in Figure 2 shows the initial, smooth state realized on a fine grid of 1024^2 cells, while the image below it shows the result of advecting this scalar field with a swirling velocity field using PPB on this same grid. At the bottom is the result of performing this same problem calculation on a coarse grid of just 16^2 cells. Inside each cell of this coarse grid, the internal structure determined by PPB's 6 moments in 2D is displayed.

The key advantage of PPB advection is its incorporation of sub-cell information that is independent of any information contained in other grid cells. This information is updated by the scheme on every time step. It consists of 10 lower-order moments of the fractional volume variable, *f*, the fraction of the local volume occupied by the dense shell fluid:

$$\langle f\tilde{x}^l\tilde{y}^m\tilde{z}^n\rangle = \iiint_{-1/2}^{1/2} f\tilde{x}^l\tilde{y}^m\tilde{z}^n \, d\tilde{x}\, d\tilde{y}\, d\tilde{z}$$

Here the centered and scaled Cartesian coordinates, $\tilde{x}$, $\tilde{y}$, $\tilde{z}$, are used in order to make the PPB advection scheme perform well using 32-bit rather than 64-bit arithmetic. They place the origin of each cell's private Cartesian coordinate system at the center of the cell, and they scale the coordinate to unit cell width in each dimension. The resulting logically cubical grid cells correspond nicely to the actual grid cells we use in this study, since those are also cubical and Cartesian, although our moving grid makes their widths and center locations dependent upon time. Use of these private cell-centered coordinates forces us to perform a little more arithmetic in each time step update, but this is well worth its cost due to the doubled speed we achieve with 32-bit precision.

Our PPB scheme updates the first 10 moments, with *lmn* = {000, 100, 010, 001, 200, 110, 101, 020, 011, 002}. The updates are performed in 1-D passes, along with the PPM hydrodynamic computation. The procedure is described in detail in the LCSE internal report available on the Web at [4]. In the *x*-pass, for example, moments *lmn* = {000, 100, 200} are used to construct, then constrain, then advect a function $f(\tilde{x})$ using a distribution of the advection velocity in the *x*-dimension that is constant on streamlines and that varies linearly across each cell at the time level half-way through the time step. At each cell interface half-way through the time step, this linear velocity distribution assumes the time-and-space-averaged values at that interface that is computed according to the PPM gas dynamics scheme. Values of $f(\tilde{x})$ that have been interpolated and constrained are then transported without change by this 1-D velocity field during the time step, and new values of the 3 moments involved are determined by integration over the resulting distribution within each cell. From the 2 moments *lmn* = {010, 110}, we construct, *but need not constrain*, and then advect a function $f_{\tilde{y}}(\tilde{x})$ in a similar fashion, with a similar procedure for a function $f_{\tilde{z}}(\tilde{x})$. Still simpler functions $f_{\tilde{x}\tilde{y}}(\tilde{x})$, $f_{\tilde{x}\tilde{z}}(\tilde{x})$, $f_{\tilde{y}\tilde{z}}(\tilde{x})$, $f_{\tilde{y}\tilde{y}}(\tilde{x})$, $f_{\tilde{z}\tilde{z}}(\tilde{x})$ are constructed and advected. Advection of the last two is more complicated than may immediately be apparent, because of the interdependence of the second- and zeroth-order moments, given our definitions of them. 1-D passes are very well-suited to this advection scheme, and are extremely efficient. Constraints need only be applied in each pass to the single function $f(\tilde{x})$, where *x* is the direction of the pass. This affords an enormous simplification.

Once we have computed the time-and-space averaged velocities at the grid cell interfaces, we perform the PPB advection computation to obtain the new values of the 10 moments of f without reference to the values of any other variables. This computation gives us advected volumes of the two fluids, but not advected masses. We obtain volumes of the two gases within the cells *at the beginning of the time step* that cross the cell interfaces into neighboring cells. To convert these advected volumes into advected masses, needed for strict mass conservation, we must introduce interpolations of the individual fluid densities as functions of cell volume coordinates at the beginning of the time step. We find these interpolation parabolae using our standard PPM procedure.

Here we invoke an important assumption. We assume strict pressure and temperature equilibrium inside each cell. This implies that the two fluids, capsule and fuel in this case, must have at each point in the cell a ratio of their densities that is given by that of their mean molecular weights – which we assume is constant over the entire duration of the problem. In both our test problems, we take this ratio to be 100. The pressure and temperature equilibrium assumption permits us to derive the individual densities of the two fluids given only the averaged density of the mixture plus the mixing fraction, f. This is a huge simplification, because we need not store densities and internal energies for each fluid. When we interpolate a parabola to represent the variation of the density of one of the fluids across a cell, this implies such a parabola for the other fluid. Together with our moments for the distribution of the mixing fraction, f, we can derive the implied distribution of density for the mixture. Because careful interpolation is very expensive, this represents a great saving in computational labor. There is still another advantage. Our equilibrium assumption means that even in a cell containing no gas of one type, a reasonable average density for that gas is implied. Consequently, we have no need to interpolate gas density across a discontinuity at a multifluid boundary: the density of each gas is well-behaved across such a boundary. It is instead the mixing fraction, f, that jumps suddenly across this boundary. However, we have for f our very much more accurate PPB description, with its 10 moments in each cell. This allows, as a practical matter, a smooth description of f across a multifluid boundary that is only about 2 grid cells thick. This representation is prevented from becoming too sharp, which would introduce numerical oscillations or glitches, because it is forced to consist of a parabola extending all the way across each grid cell. Our interpolated distribution of f is therefore very sharp – only a couple of grid cells in thickness – and at the same time very smooth, because it is defined by parabolae in these cells that are determined by subcell information that is operationally equivalent, as a rule of thumb, to a two- or three-fold grid refinement for PPM (cf. [4]) for just this single, all-important variable.

In slow-flow, Rayleigh-Taylor instability problems (cf. [13-15]), we find that the elaborate approach of the PPB scheme is sufficient to essentially eliminate the appearance of certain bad behaviors familiar to us from the PPM advection scheme when applied to multifluid problems (cf. for example, [19,20]). PPB advection, with its formal fifth-order accuracy, is capable of moving multifluid interfaces with very detailed structure great distances through the mesh with no noticeable diffusion. This behavior is possible, because PPB consistently treats the internal structure of the multifluid interface transition from 0 to 1 in the mixing fraction, f, as a smooth transition. Unlike PPM advection, it does not switch between fundamentally different interpolation strategies dynamically. Switching strategies in this way can cause PPM to introduce small glitches, which can later become amplified by a physical instability to form large glitches. We have been using multifluid PPM+PPB for Rayleigh-Taylor problems in the weakly compressible regime since 2004, and find it very much superior to PPM alone for these problems, as the results reported in [21] and later in [13] attest. In 2010, Almgren et al. reported similar experience with high-order advection and Rayleigh-Taylor problems using their CASTRO code [22]. This experience, we find, does not carry

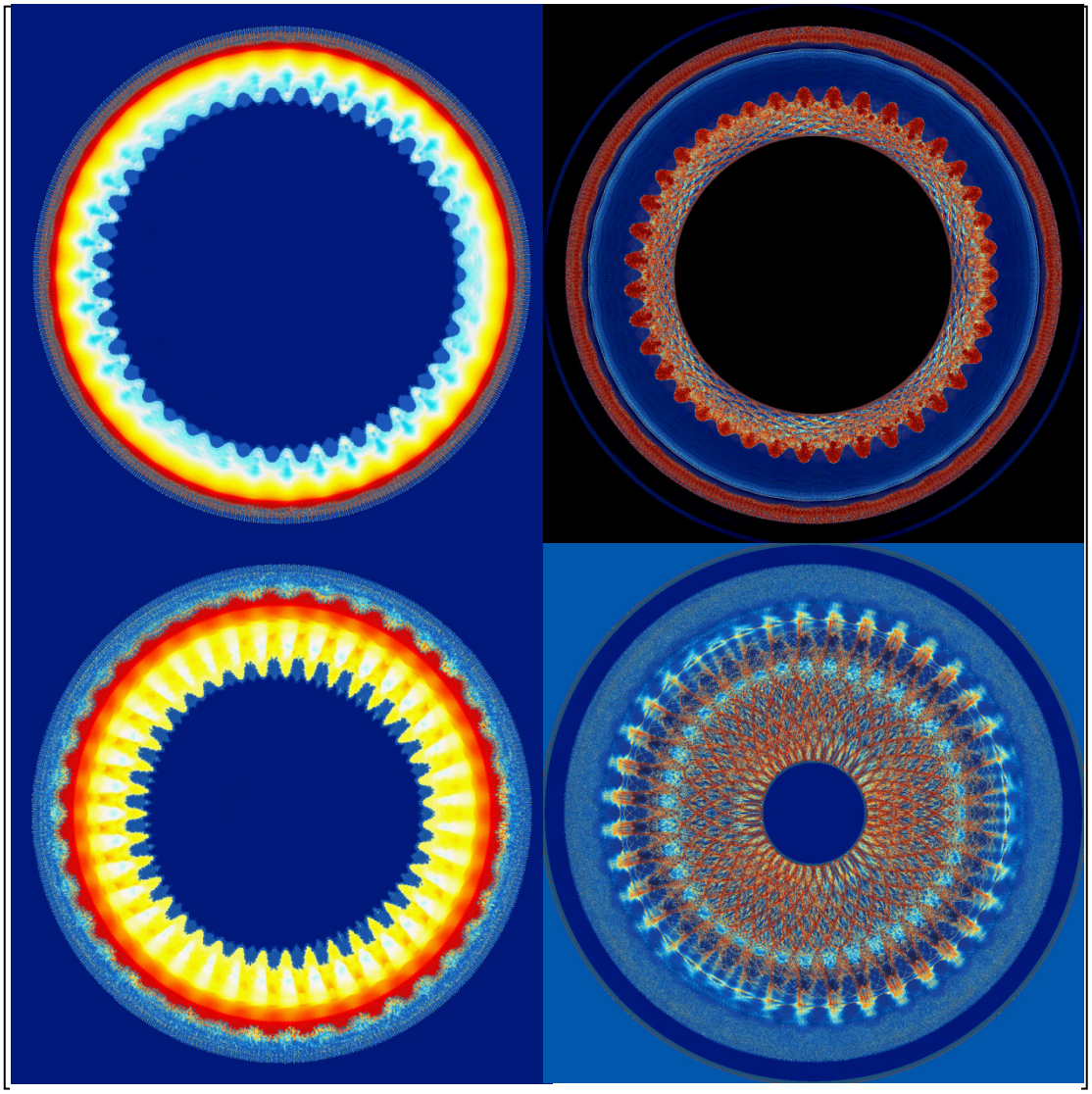

Figure 3. Density (left), magnitude of vorticity (top-right), and negative divergence of velocity (bottom-right) in a slice of thickness 3/44 through the equatorial plane for Problem 2 at t = 0.025833 and 0.0325 (10560^3 grid). See text for discussion.

over to the much more violent ICF problems considered here. This can be seen in 2-D for both our code and for CASTRO in our earlier code comparison work reported in [1]. We find that the interaction of strong shocks, as they are handled in PPM, with our very carefully treated multifluid interfaces produce familiar sorts of glitches, whose causes were explained decades ago in [7]. We will address these issues after first reviewing the observed behavior of our PPM code on our two test problems.

ICF Test Problem #2 Behavior

To give an idea of the fluid flow in our ICF test problems, defined above, we begin by showing our best approximation to this flow, beginning from the initial conditions of Problem 2. After gaining a general understanding of the flow in this way, we will discuss in detail specific challenges, and techniques for addressing them, that these problems involve. We have attacked our Problem 2 using a grid of 10560^3 cells, which move homologously inward at roughly the same rate as the outer shell boundary, as described in the definition of this test problem. Each of the views in Figs. 3-7 has a diameter of just about $2.8\ (1 - 9.75\ t)$, the

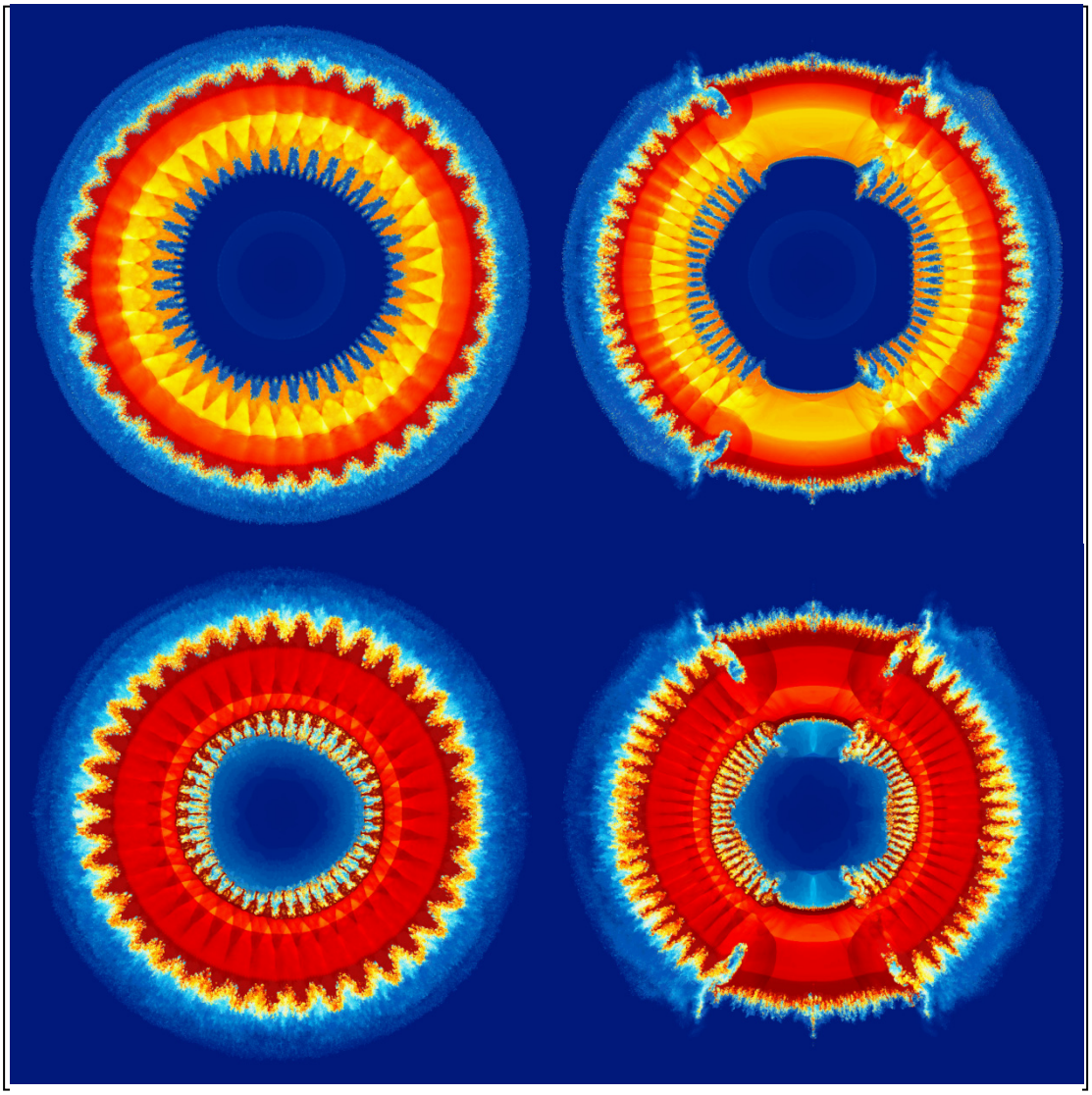

Figure 4. Density in a slice through the equatorial plane (left) and a longitudinal plane (right) for Problem 2 at t = 0.0375 and 0.0425 (10560^3 grid). See text for discussion. Images showing the positions of shock waves more clearly appear in Figures 8 and 9. The principal shock reflects from the origin at time 0.0342. In the images at the top, its reflection can be seen propagating outward a little more than half way to the inner surface of the dense shell. In the images at the bottom, this reflected shock is working its way into the dense shell gas.

width of the active simulated region inside our inflowing boundary gas. The entire calculation required less than 41 hours of computation on the portion of the NCSA Blue Waters machine that does not contain GPU accelerators. This computation time was accomplished in stages of roughly 6 hours each over the course of 2 weeks around Christmas, 2012. Our PPM code sustained 1.5 Pflop/s performance in 32-bit mode, which precision is entirely adequate for this very demanding problem, because of the careful design of the numerical methods involved.

This computation, with over one trillion cells, is intended to establish to the best of our present ability the detailed nature of the solution to our Problem 2. We therefore hope that its very fine grid is *more* than adequate. We can afford this luxury because the Blue Waters machine is so powerful that even this trillion cell grid does not cause the machine any particular difficulty.

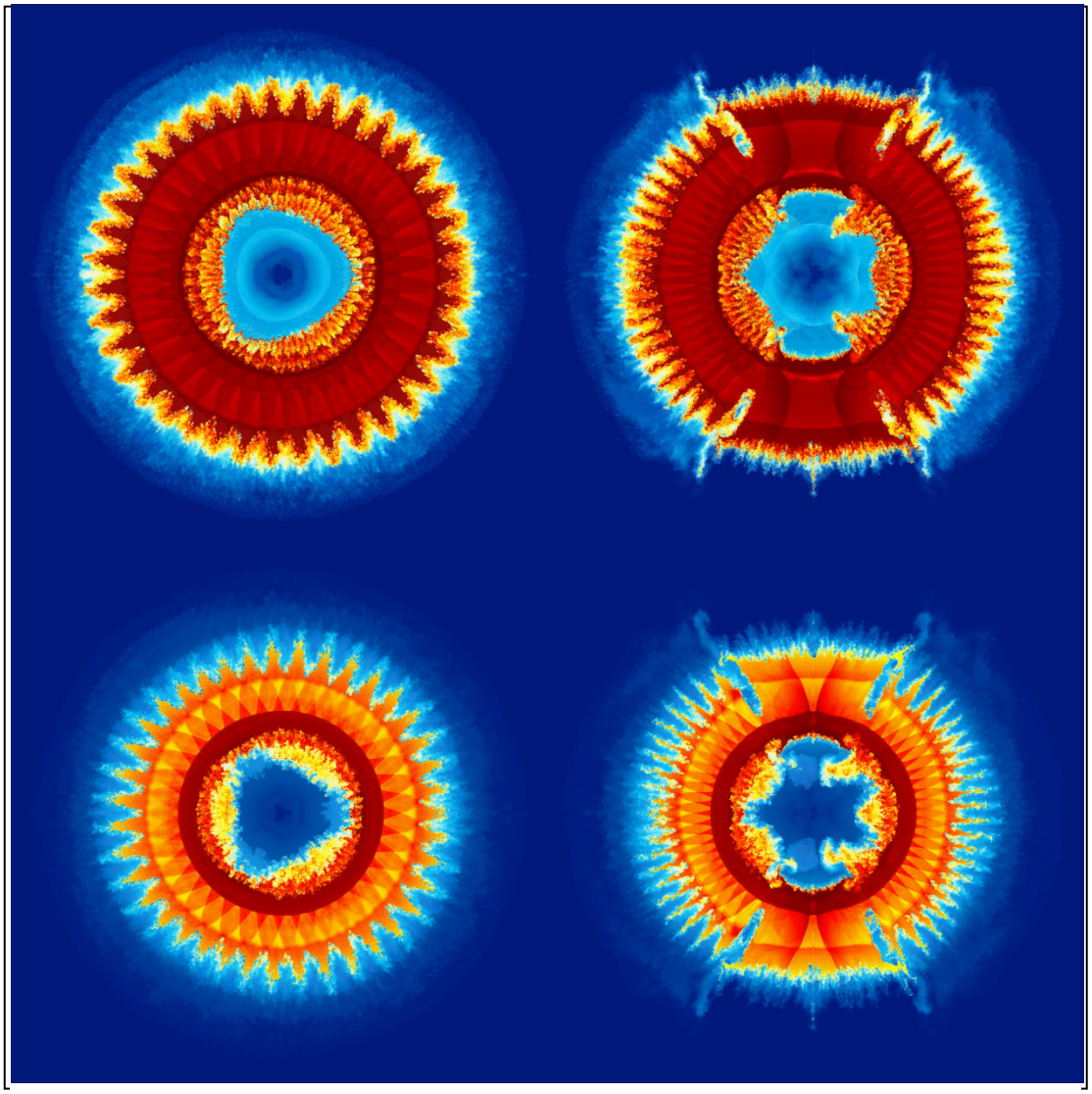

Figure 5. Density in a slice through the equatorial plane (left) and a longitudinal plane (right) for Problem 2 at t = 0.045 (top) and 0.048333 (bottom) on a 10560^3 grid. The mixing region at the inner dense shell surface is becoming turbulent due to the action of secondary shear instabilities during this time interval. A second reflected shock from the origin begins to impinge on the mixing region at time 0.045 and has passed through it by t = 0.048333. In the images at the bottom, a different color map is used to bring out detail in the dense shell.

We will not waste space by showing more than a portion of the imploding shell at early times. A far more complete set of images from this simulation, rendered at full resolution, is available for viewing on the Web at www.lcse.umn.edu/ICF2012. Because the visualization data for each snapshot of this flow consists of 7 subdirectories of 1331 files each, totaling in aggregate 16.5 TB, we restrict ourselves in this early report to volume-rendered views through thin slabs of the problem domain oriented in planes perpendicular to the Y-axis, the axis joining the two poles of the capsule, and the Z-axis, which lies in the equatorial plane. Views of the whole of such slabs have the plane sliced through the origin in the foreground and are 1/44 of the domain thick, a distance of 3/44 times the grid compression factor at the time. This is the thickness of one of the 1331×64 cubical subdomains that are updated by individual CPU dies, each containing 8 CPU cores attached directly to a local shared memory. Thus this particular slab thickness is preferred, given the layout of this computation.

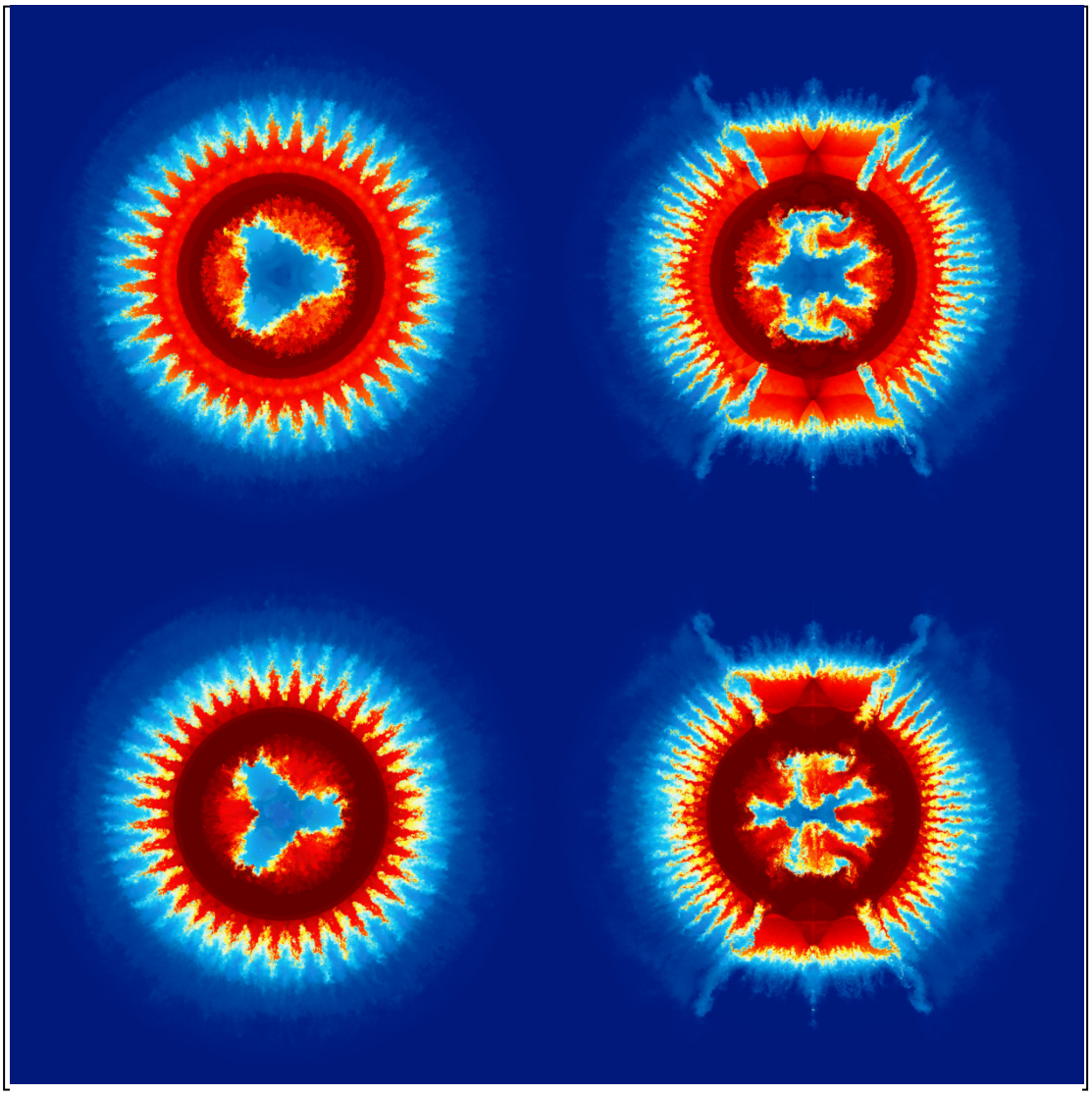

Figure 6. Density in a slice through the equatorial plane (left) and a longitudinal plane (right) for Problem 2 at t = 0.0508333 and 0.05333 (10560^3 grid). See text for discussion.

We use perspective volume rendering, with a choice of opacity that emphasizes the near surface of the slab while still giving an impression of the deeper layers. As in traditional mechanical drawing, we attempt to give a sense of the full 3-D aspect by using our two orthogonal slices. Better representations, for which we do not have sufficient space here, can be found on the Web site listed earlier. We use color maps that are nonlinear. These are not intended to be quantitative; instead they are meant to give a clear impression of the flow features as distinct from any numerical values of specific flow variables.

Generally, a progression of colors with increasing variable value is used, starting with dark blue and proceeding to aqua, white, yellow, red, and darker red. A quantitative analysis of the simulation data will be given in a later article. The details of flow features, nevertheless, are very revealing of both numerical successes and problems, as our discussion below should make clear. We can also use images of this sort to visually confirm convergence or lack of convergence upon grid refinement. Such visual tests are not at all casual; they are generally much harder to pass than quantitative comparison of various combinations of variables averaged over surfaces or other subdomains of the problem.

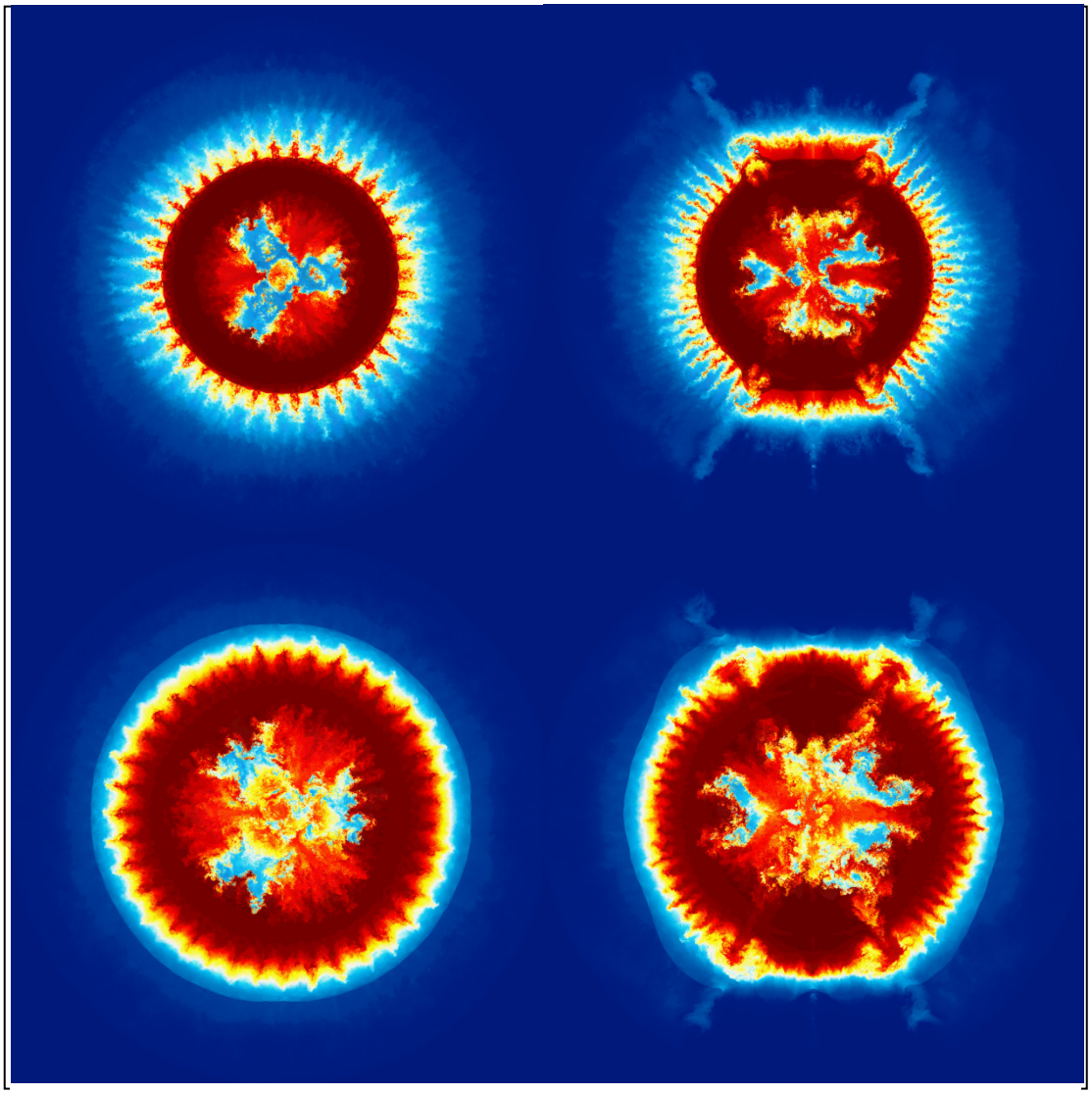

Figure 7. Density in a slice through the equatorial plane (left) and a longitudinal plane (right) for Problem 2 at t = 0.055833 and 0.060833 (10560^3 grid). Between times 0.055 and 0.055833, the capsule has reached its greatest compression. See text for discussion.

It is customary in discussing problems like those studied here to present a line plot tracing the paths in radius as a function of time of the primary shocks and multifluid interfaces in this problem. This type of plot, however, assumes that the fluid behavior is nearly spherically symmetric. In the cases studied here, such spherical symmetry does not obtain, and as a result such a plot would be misleading. Our code automatically generates profiles at its grid resolution of many quantities averaged over spherical surfaces. Extracting shock and material interface locations from these profiles, however, is quite difficult, with the exception of the single, initial strong shock that can be seen in Figures 8 and 9 followed by a wealth of weaker shocks that travel on transverse and intersecting paths. The images of the density in equatorial and longitudinal slices in Figs. 3-7 clearly show how misleading average density values on spheres are in giving an impression of the degree of material interface spreading due to the developing mixing layer in these problems.

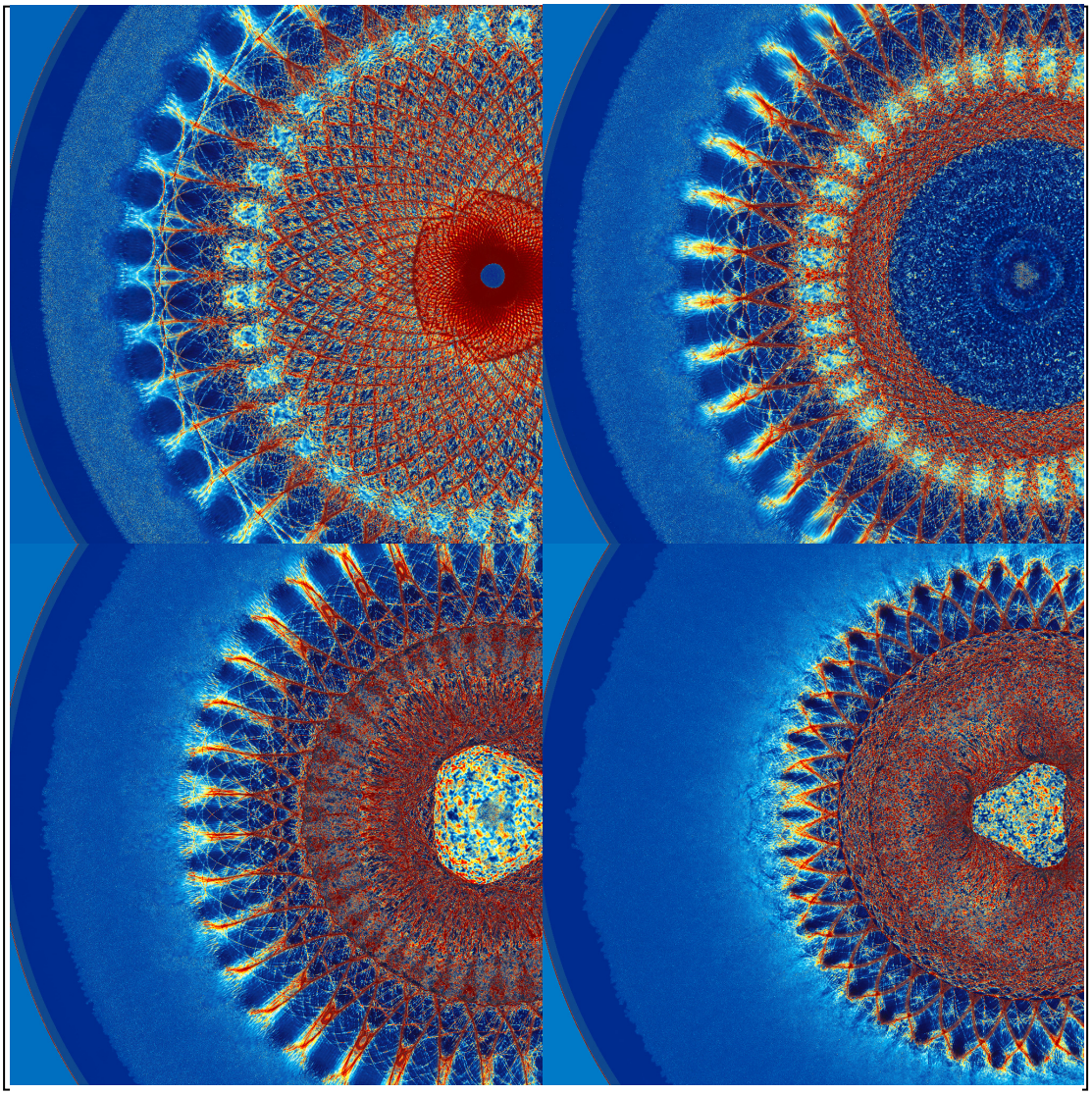

Figure 8. Views in equatorial slices for Problem 2 at t = 0.034167 (top-left), 0.038333 (top-right), 0.0425 (bottom-left), and 0.049167 (bottom-right). The negative divergence of velocity is shown in close-up views. The grid resolution is 10560^3. The initial strong shock is shown just about to reflect at the origin (top-left) and after this reflection (top-right), when the 3 shock caustics are just striking the Richtmyer-Meshkov fingers from the inner surface of the dense shell. At the bottom-left, we see the first strong shock reflection from the inner surface of the dense shell approaching the origin. At the bottom right, a second such reverberation is approaching the origin once more. A second outward propagating shock wave front can be seen about to overtake the one from the initial shock that was reflected at the origin. The initial passage of the strong shock over the inner surface of the dense shell is shown in extreme close-up views in Figure 1b. See text for discussion.

To give a better impression of the timing and spatial paths of the primary shock waves in Problem 2, we show in Figures 8 and 9 images of the negative divergence of velocity in equatorial and longitudinal slices at different times. In Figure 1b, we show extreme close-up views that reveal the complexity of the primary shock interaction with the initial perturbation of the inner surface of the dense shell and the subsequent focusing of the shock on the origin, at just about time 0.0342 (just after the time of the image at the top-left in Figure 8). Strong shock waves appear as red lines in these images. In Figure 9, we show longitudinal slices.

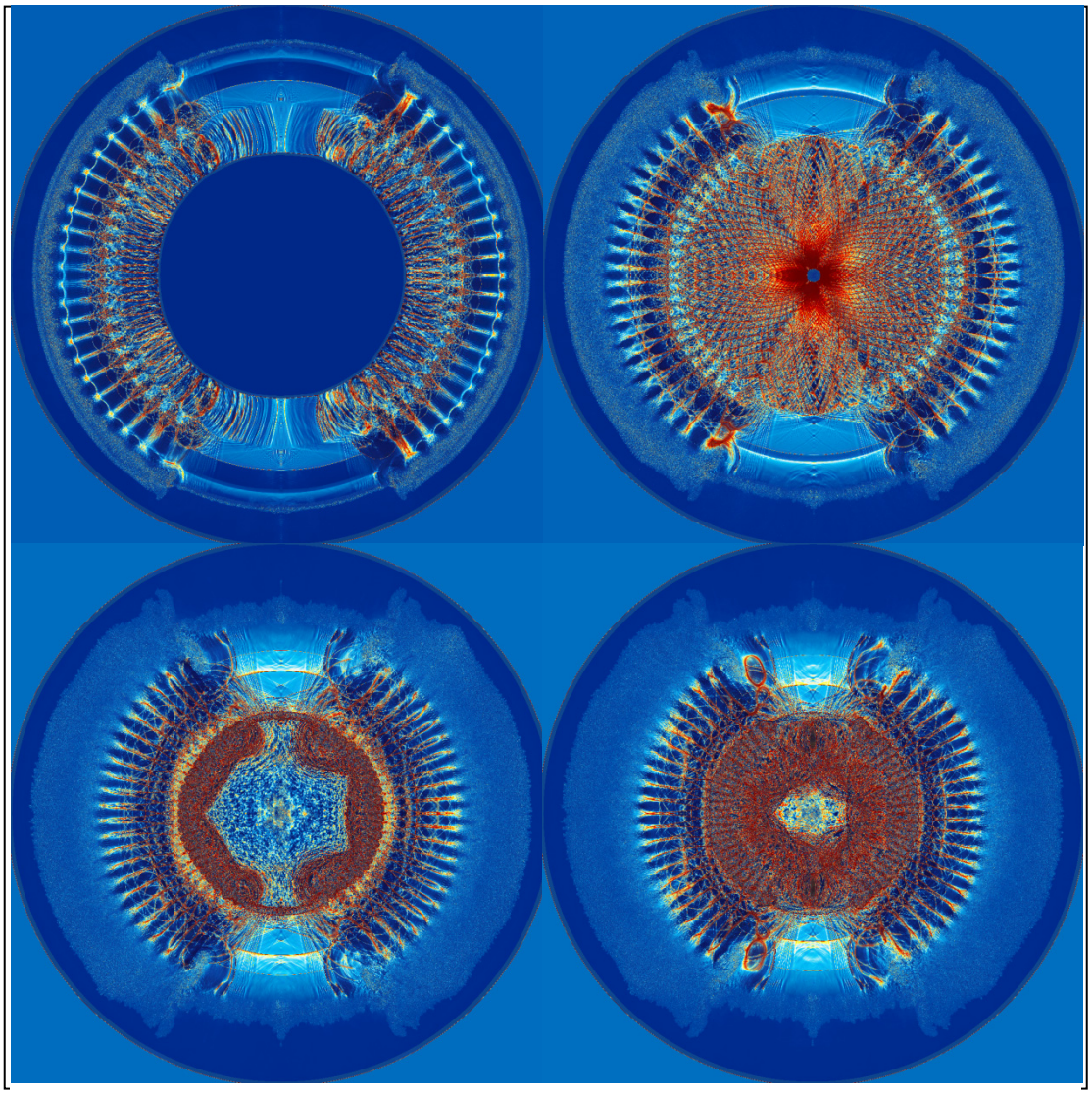

Figure 9. Views in longitudinal slices, along the "prime meridian," for Problem 2 at t = 0.0275 (top-left), 0.034167 (top-right), 0.040833 (bottom-left), and 0.0425 (bottom-right). The negative divergence of velocity is shown. The grid resolution is 10560^3. The images at the top show the initial strong shock after emerging from the dense shell being focused as it begins to converge on the origin (top-left) and just before it reaches the origin (top-right). At the bottom-left, we see the reflection of this shock impinging upon the fingers of the dense shell that have formed from the Richtmyer-Meshkov instability. At the bottom-right, the reflected shock has reached the main body of the dense shell, and a powerful and complex reverberation of shock waves is again focusing upon the origin. The dense shell is decelerated by multiple shock reverberations in quick succession, each producing further Richtmyer-Meshkov instabilities amidst the Rayleigh-Taylor instability of the mixing region.

The strong curvature of the shock fronts as they are just emerging from the inner surface of the dense shell in the center-left views in Figure 1b gives rise to the laterally propagating waves that can be seen forming complex patterns behind the advancing main shock front at the top in Figures 8 and 9. The curvature of the very strong rarefaction wave rushing back into the dense shell after the shock emerges from the inner surface, seen at the center-right in Figure 1b, imprints on the outer surface of the shell a pattern matching the imposed spherical harmonic modes at the inner surface. In Problem 2, this imprint from these rarefactions on the outer surface causes the prominent, dense spikes that develop on the outer surface from

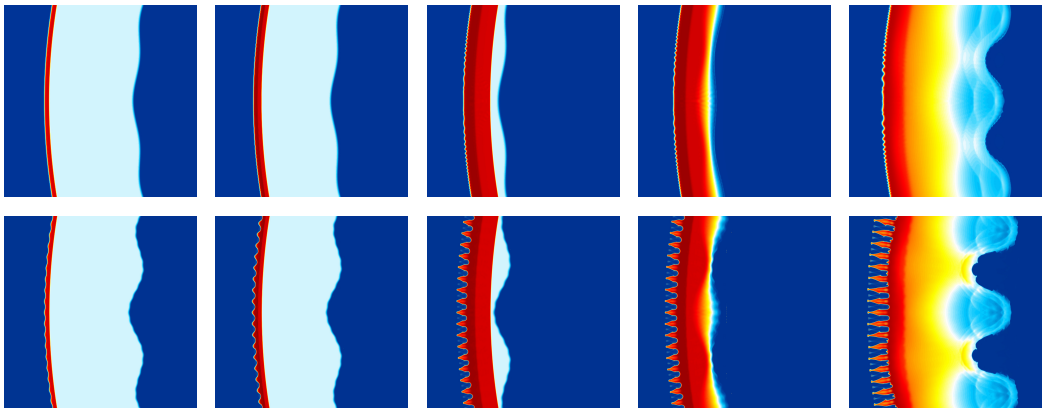

Figure 10. In both rows of images, we show a zoomed-in view of the density distribution in a small region in a very thin slice of the problem domain at the equator at problem times 0.0025, 0.0050, 0.0125, 0.0150, and 0.0225. A simulation of Problem 1 is shown in the top row, carried out on a grid of 4160^3 cells. A simulation of Problem 2 is shown in the bottom row, on a grid of 4800^3 cells. Despite the almost imperceptibly small initial amplitude of the high frequency mode in Problem 2 (it involves a displacement of the interface radius by just 1.6 parts in 12,000), by time 0.0025 it has been amplified considerably by the Richtmyer-Meshkov and Rayleigh-Taylor instability of the outer surface of the dense shell. The denser, shocked fluid of the shell shows up as red in the first 4 images. The shock emerges into the 100-times lighter gas inside the shell in the fourth image, amplifying the initial disturbance of the otherwise stable inner surface of the shell visibly in the fifth image. The high-frequency disturbance in Problem 2 has rather little effect upon the behavior of the inner shell surface, but it has a dramatic impact upon the behavior of the outer surface, which is far more unstable at these early problem times.

its continued Rayleigh-Taylor instability. In Figures 8 and 9, the non-spherical nature of the strong reverberating waves in the light gas inside the dense shell is evident. At late times, there are so many waves propagating in all directions in this gas that it is difficult to identify individual reverberations.

The initial phase of the ICF capsule implosion is shown in Figure 10. Here we contrast results for Problems 1 and 2 in calculations with nearly equal grid resolutions of 4160^3 and 4800^3 cells, respectively. In Problem 1, there is no initial perturbation of the outer shell surface, yet a characteristic pattern of ripples emerges there by time 0.0225, at the far right in the figure. This perturbation has a characteristic wavelength equal to the distance from one grid plane crossing of the shell surface to the next. As the very thin numerical representation of the multifluid interface changes from being centered upon a grid line to being centered on a grid cell, its detailed internal representation changes subtly and inevitably. The physical instability of the interface amplifies this subtle disturbance where its fundamental wavelength, measured in grid cell widths, is long enough to be followed without significant damping by the numerical scheme. These ripples of numerical origin appear wherever the thin surface of the capsule is tangent to the grid planes. Such ripples appear also, but less strongly, where it is tangent to diagonal planes of grid cells. These ripples are akin to the false, very weak sound waves that can be emitted by strong shocks whose representation is too thin for a given mesh. The shock phenomenon was identified early in the development of the PPM scheme (cf. [7]), and it has been effectively eliminated by the "smart" strong shock dissipation that is used in the version of PPM in our code (cf. [23], the more complete, Web version of [12]). These numerical artifacts mar the outer surface of the capsule in Problem 1. They are immediately recognizable and are easily identified as numerical rather than physical effects. However, the comparison in Figure 10 makes it quite clear that these effects, unsightly as they may seem, are of little dynamical importance. We will discuss this important issue in more detail later.

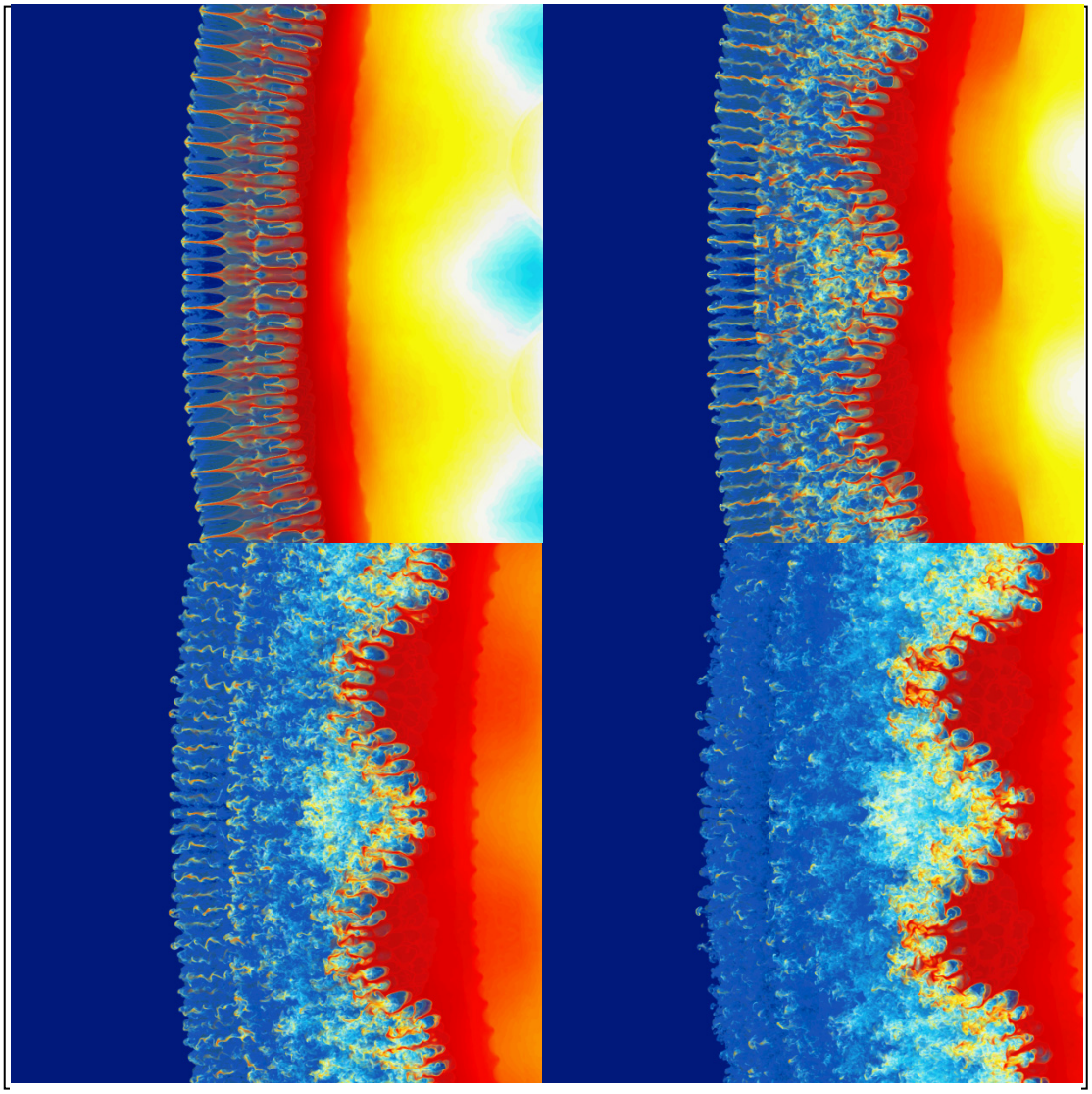

Figure 11. Close-up views of density in a slice through the equatorial plane for Problem 2 at t = 0.025833, 0.03, 0.0325, and 0.035 (10560^3 grid). See text for discussion.

In Figure 10, we can see that the initial development consists of a strong shock propagating into the dense shell material, compressing it by nearly the maximum factor allowed by the gamma-5/3 equation of state, namely a factor of 4. The shock compression of the dense shell excites the Richtmyer-Meshkov instability on both the outer surface and on the inner one. Because mixing of capsule and fuel gases at the inner surface is most important for inertial confinement fusion, we have placed our more significant perturbation on the inner surface. The perturbation of the outer surface in Problem 2 is trivial; we smear the initial representation of the multifluid interface over 12 grid cell widths, and we displace the interface, on our 10560^3 grid, by only 0.55 grid cell widths. We resolve the wavelength of this trivial disturbance with 77.8 cells, which is sufficient to give quite a good treatment of the development of this instability, despite its trivial initial amplitude. This development gives us a measure of the thickness of the mixing region at the outer surface that is essentially unavoidable in a 3-D world. This unstable mode is amplified continually by the Rayleigh-Taylor instability, since the acceleration of the multifluid interface is from the lighter to the denser fluid. The behavior of our code in treating Rayleigh-Taylor problems in slab geometry has been studied in depth, and has been reported in [13-15]. 78 cells per wavelength should give quite a good

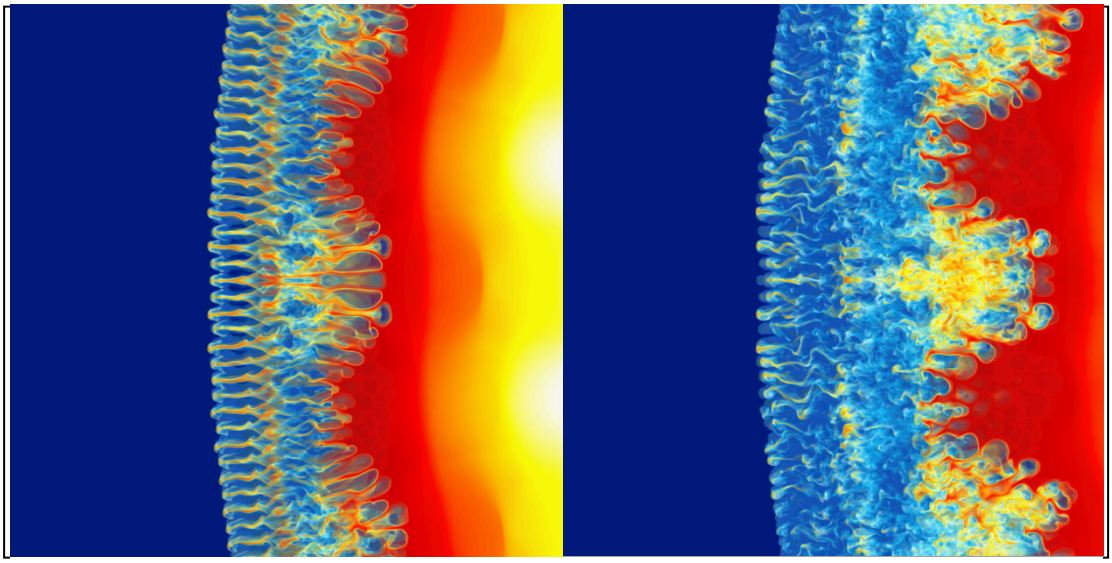

Figure 12. Close-up views of density in a slice through the equatorial plane for Problem 2 at t = 0.03 and 0.035 (4800^3 grid). See text for discussion.

representation, although convergence for the single-mode Rayleigh-Taylor problem at late times was seen only at 128 cells per wavelength, and then in a statistical sense of ensemble behavior, as reported in [14].

The development of the instability of the outer capsule surface in Problem 2 is shown in Figure 11. We focus on the section of an equatorial slice where the amplitude of the perturbation of the inner surface is largest, so that its imprint upon the outer surface will be greatest. This is the region at the left in the left-hand images in Figures 3-7. In the 4 images in Figure 11, we see the progression of the instability from a highly ordered state at the top-left, through development of disorder near the surface through interaction with waves arriving from the more strongly perturbed inner surface, to complete disorder.

This disorder involves of course nonlinear interaction with myriad tiny disturbances of numerical origin at the microscale defined by the computational grid. As stated earlier, we regard these interactions as benign. They provide a path to the chaos that would certainly apply in any attempt to realize this situation in a real laboratory experiment. The particular modes introduced at the microscale, and the details of their amplification through the physical instability, we take to be unimportant. This view is bolstered by the lack of any obvious falsifications of the chaotic flow clearly arising from our grid. Only the two, small, symmetrically shaped and placed modes at the center-right jump out to the practiced eye. These are falsifications resulting from *too much* symmetry imposed on the flow along the special, bisecting grid line of each image. The reader may judge for him- or herself how much this aspect of the result in this tiny region mars the overall simulation.

The flow shown in Figure 11 involves instability at a large density ratio, in this case a factor of ten. (On the inner surface of the capsule, this ratio is 100 initially). This is fairly demanding for numerical simulation. We have described the care with which we treat the multifluid interface, and we believe that this helps to make accurate handling of this high density ratio feasible. In support of this belief, we offer the comparison between simulations of Problem 2 with our PPM code running on grids of 10560^3, shown in Figure 11, and 4800^3 cells, in Figure 12. On the coarser grid, we have only 39 cells per disturbance wavelength, which is not sufficient for convergence according to our earlier Rayleigh-Taylor studies [14]. From comparing Figs. 11 and 12, it is clear that this detailed aspect of the simulation has not converged at the 4800^3 grid resolution. Nevertheless, the changes upon grid refinement are not dramatic,

which helps to build confidence in the solution on the finer grid.

The behavior of the inner surface of the ICF capsule is both more important and more interesting. The very high-order mode introduced at the inner surface in problem 2 has so short a time scale for amplification by the Richtmyer-Meshkov instability that this amplification is missed entirely by the time interval of 0.0025 that we used between output dumps near the beginning of this calculation. The last 2 images at the bottom right in Figure 10 give the best view of the growth of this mode following passage of the initial strong shock through the inner capsule surface. This growth has caused an array of closely spaced little jets of capsule material to develop, but it is already clear in Figure 10 that this process has had little impact upon the development of the longer-wavelength perturbations. These little jets of denser gas do not assume nearly the prominence of those growing from 5-times-smaller initial perturbations on the continuously unstable outer surface. This is no doubt due to the Rayleigh-Taylor *stability* of the inner surface during the initial implosion of the capsule.

The Richtmyer-Meshkov instability has the nature of a one-time-only deposition of vorticity to the multifluid interface. The refraction of the shock wave as it passes over the perturbed interface sets up velocity differences that lead to a continued, but essentially coasting growth of the initial disturbance. Secondary shear instabilities do develop, but the Rayleigh-Taylor stability of this inner shell surface during the implosion keeps the flow relatively orderly here. This is clearly seen in Figure 3, where the modulated mode 39 disturbance is seen to grow steadily without the development of chaotic behavior on length scales commensurate with it.

The light fuel gas enclosed by the capsule exhibits during this stage of the compression an elaborate and highly ordered pattern of shock waves and shear layers. These are seen at the right in Figure 3, well resolved on our trillion-cell grid. Their mode-3 modulation, arising from the interaction of the imposed modes 39 and 36, is clear. There is no breaking of this pattern along the vertical and horizontal grid axes, as we would expect to find at low grid resolution. There is also a marked absence of glitches in the near perfectly round outer edge of the hair-like Rayleigh-Taylor tendrils or in the associated features in the vorticity and velocity divergence images at the right in the figure. We presume that this lack of glitches is the result of the physical behavior we simulate for the imposed high-order mode overwhelming the numerical scheme's tendency to introduce false modes at the grid-plane-crossing frequency of the dense shell surface. The disparity in the amplitudes of these two phenomena, one real and one false, is evident in comparing the ripple sizes in the top and bottom rows of images in Figure 10.

The elaborate patterns of shocks and shear layers revealed in Figure 3 would surely trigger the automated grid refinement strategies of most AMR schemes. For this reason, our simulations would be very difficult to make more efficient by the introduction of AMR. The updating of undisturbed cells in the shell interior at the beginning of the problem and in the boundary region throughout the problem are a price we pay for our uniform Cartesian grid. However, AMR also has a price in the overhead involved with grid refinement and derefinement as well as in dealing with the load imbalances that develop and must be handled dynamically. We suspect that in this particular problem either approach – our uniform grid or the AMR grid – has about the same overall cost. However, our uniform grid approach has an immensely lower programming cost.

In Figure 4, we see the interaction of the perturbed inner shell surface with the very strong shock that is reflected from the origin. This shock can be seen propagating outward in the light gas at the top in Figure 4. It is clearly almost perfectly round, despite the mode-3 modulation of the complex pattern of shocks and shear layers through which it is moving. The

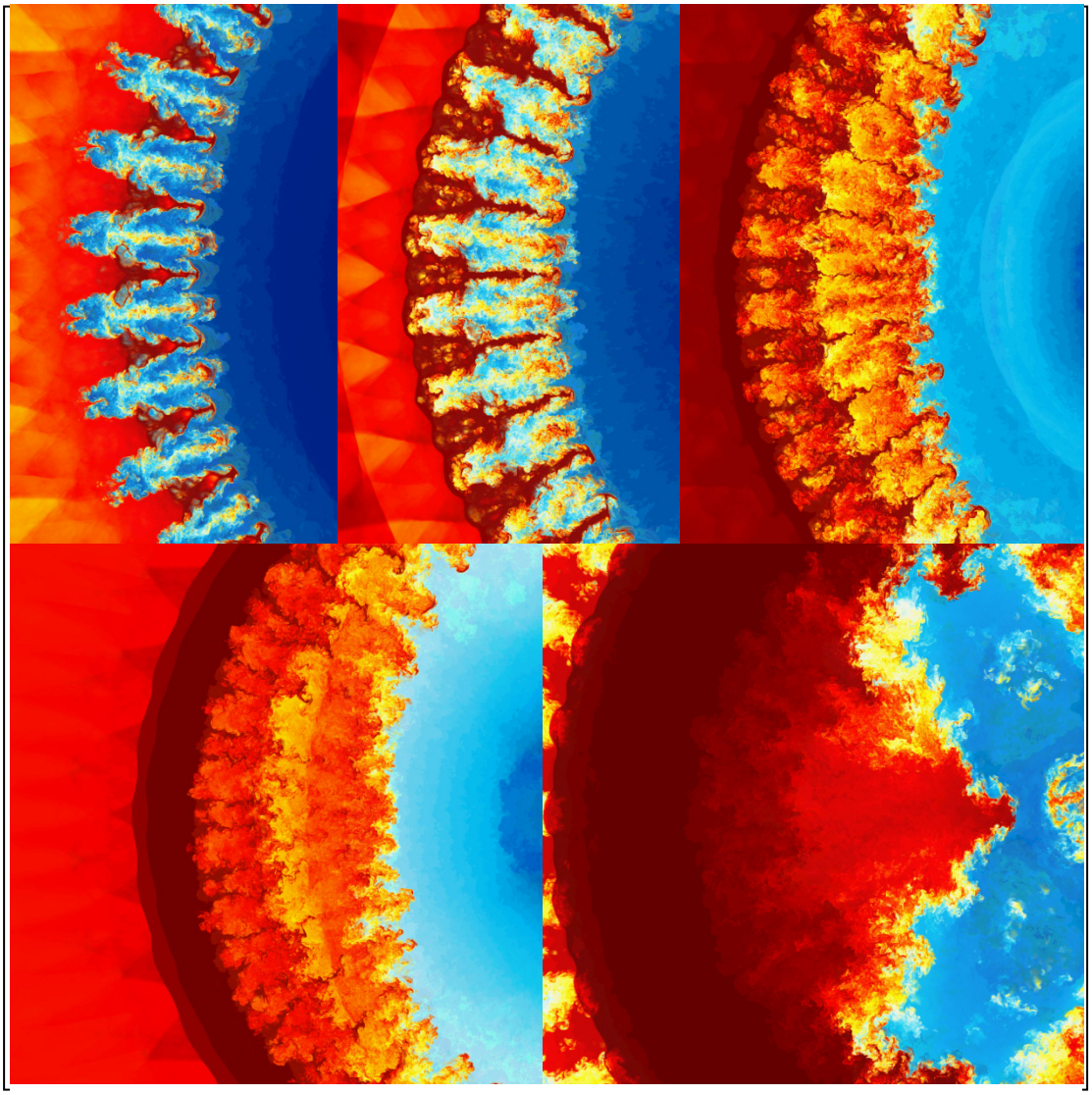

Figure 13. Close-up views of density in a slice through the equatorial plane for Problem 2 at t = 0.040833, 0.0425, 0.045, 0.046667, and 0.055 (10560^3 grid). See text for discussion.

inner capsule surface is anything but round at this point. Its mode-3 modulation is distinct in the equatorial plane (at the top left in Fig. 4), and the associated modulation in latitude is also distinct in the image at the top right. The interaction of this shock with the inner surface of the dense shell in the equatorial plane is shown in a series of close-up views in Figure 13. We will discuss these in more detail a bit later.

The image at the top right in Figure 4, supplemented by the close-up views in Figure 14, gives us a nice direct comparison between the simulated behavior of the perturbed shell surface and of the essentially unperturbed polar regions. Near the poles, the initial perturbation amplitudes were orders of magnitude smaller, because of the character of the spherical harmonics. At the inner shell surface, we see almost no disturbances in these polar regions, save tiny glitches exactly at the poles that arise from the confluence of both physical and grid symmetry there. These glitches and small disturbances are insignificant in comparison with the results of the growth of our imposed perturbations at lower latitudes. We presume that this near absence of numerically generated disturbances at the inner surface results from the physical stability of the acceleration there, once the strong shock has passed through it.

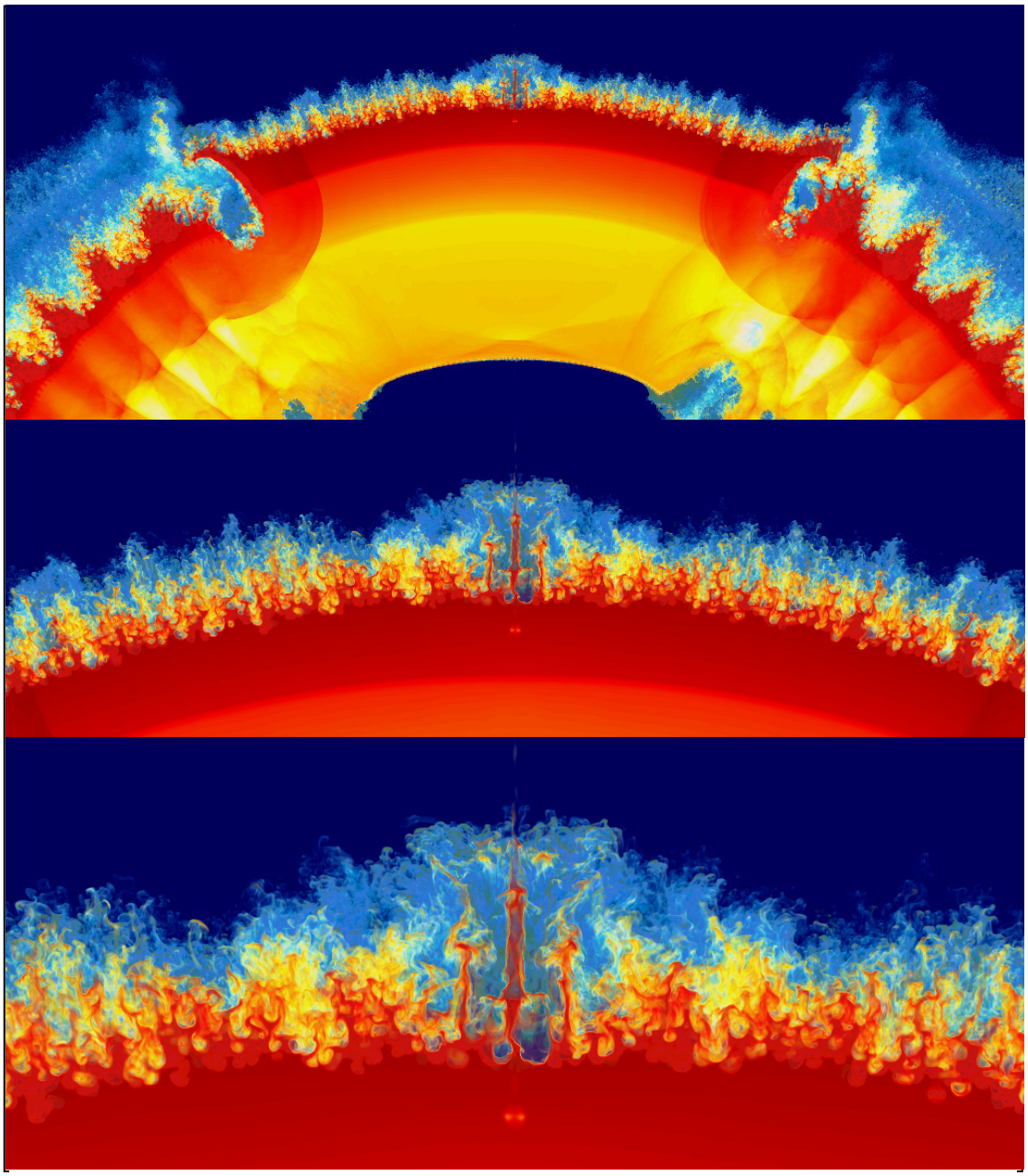

Figure 14. Close-up views of density in a longitudinal slice of thickness 3/44 for Problem 2 at t = 0.035 (10560^3 grid). See text for discussion.

The situation at the outer shell surface in the polar regions is similar, except that disturbances of numerical origin there have been amplified by the Rayleigh-Taylor instability. We have the usual polar jets and fountains, viewed close up in Figure 14. On this very fine grid they are really quite small. All along the outer shell surface in the polar regions we have development of many high frequency unstable modes. These have been generated by numerical effects, and they exhibit little organization beyond the microscale of the calculation. They give us a measure of the minimum level of interface spreading that we could reasonably expect to find in any realization of this implosion process in a laboratory. One could in fact argue, as we have earlier, that the much greater interface spreading found at lower latitudes is more reasonable to expect in the real world. In any case, we see that the interface spreading resulting from our imposed high-order mode on the outer surface is very much greater than

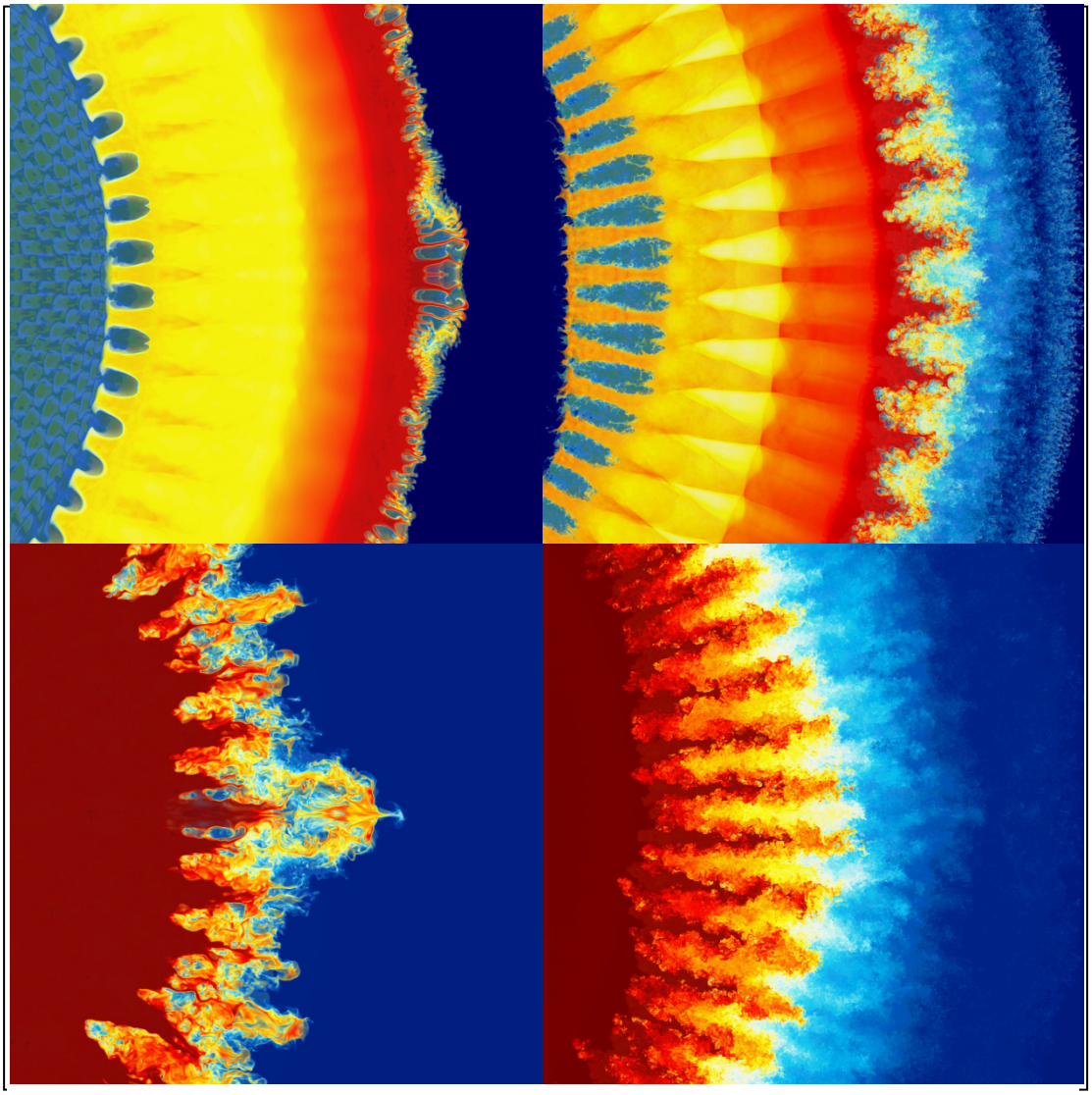

Figure 15. Close-up views of density in a longitudinal slice through the "prime meridian" with $z = 0$ for Problems 1 (left, 4160^3 grid) and 2 (right, 10560^3 grid) at $t = 0.035$ (top) and 0.050 (bottom). The addition of the high-order spherical harmonic disturbance has effectively eliminated the appearance of numerically-generated "fountain" features where the dense shell surface is tangent to the planes of the grid. These features, seen at the left, have been overwhelmed by the behavior of the high-order mode, which would be essentially undetectable in the initial state of a corresponding real laboratory experiment. However, neither behavior has an appreciable influence upon the behavior of the inner shell surface over the course of the simulation.

the minimum spreading that the numerical scheme can deliver on this grid, as measured by the spreading that it *has* delivered in the polar regions.

Our conclusions about the role of the high-order mode in this problem are reinforced by the results shown in Figure 15. Here we compare, at the same problem times, the density distributions in a slice through the "prime meridian" (with $z = 0$) near the equator. The 2.5 times larger initial amplitude of the longer-wavelength disturbances make the imprints of these modes on the outer surface much stronger in the images at the right, from Problem 2. However, asking the numerical method to follow the development of the high-order mode, introduced with truly trivial amplitude at the outer shell surface, has clearly eliminated the

tendency of the code to produce the "axis fountain" seen in Problem 1, where no such high-order perturbation was applied.

Here we note that in all our figures we slice the problem domain at the locations that reveal the worst numerical behaviors – the precise equatorial plane and longitudinal grid planes passing through the origin. Axis fountains and similar numerical artifacts seen more in Problem 1 than in Problem 2 are of very limited extent in the third dimension out of the plane of the image, especially on our finer grids. We should also note that in 3-D these numerical artifacts have far less influence on the dynamics, even on coarse grids, than they do in 2D. This is because their influence scales as the ratio of their area on the surface to the area of the entire spherical surface. Their lateral extent on the surface is always the same when measured in grid cell widths. Thus, with each grid refinement by a factor of 2 in 3D, the influence of these unwanted features declines by a factor of 4, while in 2D it declines by only a factor of 2.

The results from Problem 1 shown at the left in Figure 15 clearly display behaviors of numerical rather than physical origin that mar the simulation's representation of the outer surface of the ICF capsule. The disturbance of the inner surface of the dense shell is very cleanly simulated, as we see at the top-left in Figure 15. However, the imprint of this disturbance on the outer shell surface is confused by numerical artifacts to some extent at both of the times shown at the left in Figure 15. The images at the left and right in Figure 15 are not precisely comparable, because at the left, in Problem 1, we use perturbations of the inner capsule surface that are 2.5 times smaller than at the right, in Problem 2. Nevertheless, it is clear that the average location, the 50-50 mixing contour, in both simulations is pretty much at the same location, if we average over a distance comparable to the Rayleigh-Taylor spike spacing. The channels between these spikes of dense gas are longer in Problem 2, so that the light external fluid has bored deeper into the capsule material in this problem. The action of the Rayleigh-Taylor instability has allowed the outer gas to penetrate further in Problem 2 by pushing the dense gas aside rather than having to accelerate its mass inward.

In Problem 1 we have numerically generated features at the outer surface that are prominent only at the points of the compass. Even though our comparison with Problem 2 shows that these features do not spread the interface more than would be unavoidable in a real-world experiment, their location at only specific points can allow the outer gas to drill into the capsule material with increasing efficiency at these special locations. This does not happen appreciably in Problem 1 when it is run on a grid of 4160^3 resolution, or even at half this resolution. However, this is a potential failure mode for such simulations on Cartesian meshes at low grid resolutions.

Introducing the high-order mode at the outer surface of the capsule in Problem 2 replaces numerically generated features that are amplified by the physical instability with real features that are so amplified. These are evenly spread across the surface so that no special points form deeply penetrating channels into the capsule gas. Except of course those that form at the latitudes where all our spherical harmonic disturbances in these problems rapidly tend toward zero in approaching the poles. These channels are clearly evident in Figure 4 in the longitudinal slice views at the right. They are real features, not numerically generated ones. They arise from our exclusive use in these problems of spherical harmonic modes Y_{lm} for which m/l is roughly 1/2. We would not expect to see them in a laboratory experiment, because it would be impossible to produce initial disturbances that consisted exclusively of these special modes.

We thus see that the problem with the numerical treatment on our moving Cartesian grid of the outer shell surface in Problem 1 is that relatively isolated Rayleigh-Taylor modes are

introduced by the numerical treatment at special locations. These mess up the behavior at this surface that would happen in an idealized, but impossible world. The results of Problem 2 suggest that, overall, this messy behavior is less disturbing than the unstable behavior at the interface that would actually happen in any real experiment. Nevertheless, the real behavior would be highly unlikely to single out the special surface locations selected by the numerical scheme. By introducing a tiny disturbance at the microscale defined by the grid, in Problem 2, we do away with the special role played in the numerical treatment by those surface locations that happen to be tangent to planes of the grid. The resulting behavior is far more realistic. In particular, symmetry breaking by axis fountains and other numerical artifacts is almost completely eliminated. Overall, we have more spreading of the mixing layer at the outer surface, but we argue that this spreading would be unavoidable in any real experiment. Thus not only do our simulation results appear more realistic, we suggest that they actually *are* more realistic.

Our solution to the problem of Cartesian grid artifacts at unstable multifluid interfaces requires a scale separation between the imposed perturbations whose growth we wish to predict and the tiny perturbations, presumed undetectable in a physical experiment, that we introduce to regularize the code's numerical behavior. This scale separation is impossible without a fine grid, and that of course has been impossible in 3D until quite recently for these demanding problems. However, this demand for a scale separation between effects under study and unwanted effects that must necessarily come into play has a direct correspondence in laboratory experiments. As an example, we cite the single-mode Richtmyer-Meshkov experiments of Jacobs et al. (cf. for example [24]), where a diffused initial boundary between gases and a relatively large initial perturbation from rocking the apparatus were used to control unwanted features. Wall phenomena quite reminiscent of our numerical axis features nevertheless occurred. The initial spreading of the multifluid interface in these lab experiments was also helpful, in much the same way that our numerically spread initial interfaces are in the present study. In the lab, the size of the experiment had to be many times the smeared thickness of the initial interface, and the size of the initial perturbation had to be measurable despite this interface smearing. These considerations are familiar from the design of numerical experiments.

In figures 4-7 we see, in global slice views, the development of the mixing layer at the inner dense shell surface as it is struck by shocks reflected from the origin. The first such shock brings in its wake the Rayleigh-Taylor instability of this surface. The secondary shear instabilities, as well as the effects of the many reverberating local shock waves, ultimately lead to chaos. Close-up views of this process are shown in Figure 13, where we see equatorial slices in the region of maximum initial perturbation amplitude. The first reflected shock is seen roughly half-way through the mixing layer in the image at the top-left in Figure 13. It is laterally compressing the jets of dense fluid from the original Richtmyer-Meshkov instability of this surface. In the image at top-center, the shock has just reached the ends of the channels separating the dense fluid jets, and it has compressed these jets into thin tendrils, all pointing toward the origin. The directions of the jets are highly ordered, but small wiggles and bends along their lengths make each such jet individually distinct. These wiggles are amplified readily by shear instabilities, so that by the time of the top-right image in the figure, the identity of the jets is nearly lost. Later, in the image at the lower-left, only the bases of the jets can be discerned. In the final image at the lower-right we truly have a well-mixed fluid that is nevertheless confined to a single, well-defined plume.

The single plume at the bottom-right in Figure 13 arises from mode 3 at the equator. We did not introduce this mode directly; it has developed from nonlinear interaction of modes 39 and 36, helped along by a wealth of small-scale modes. This flow is chaotic, of course, but the

mode-3 plume that ultimately forms looks simple enough that its emergence should be completely predictable, even in this amazingly complex context. To make such a prediction at less cost than this trillion-cell computation (which cost is nevertheless affordable, but not every day), we would require a statistical model of the mixing process. Many investigators have come up with such models (see for example [25-28]), and one use for the output data of the simulations reported here is to compare that data with the predictions of such models.

Although one might not want to undertake a trillion-cell calculation every day, even on an exascale computing system, we assume that one would want to undertake 3-D rather than 2-D simulations of processes such as are studied here. In such calculations, one would be able to afford to capture the larger modes with reasonable accuracy; it would be simulating the effects of the many smaller-scale modes that would be challenging. This is the task of a closure model designed for use in a large-eddy simulation (LES). We have saved from our simulations averages over briquettes of 4^3 cells of the following variables and variable products: $f,\ f(1-f),\ \rho,\ V=1/\rho\,,\ p,\ \rho u_x^2,\ \rho u_y^2,\ \rho u_z^2,\ \rho u_x,\ \rho u_y,\ \rho u_z,\ \rho u_x u_y,\ \rho u_x u_z,\ \rho u_y u_z,\ \rho H u_x,\ \rho H u_y,\ \rho H u_z,\ -\vec{\nabla}\cdot\vec{\mathrm{u}},\ \left(\vec{\nabla}\times\vec{u}\right)_x,\ \left(\vec{\nabla}\times\vec{u}\right)_y,\ \left(\vec{\nabla}\times\vec{u}\right)_z,\ V\ \partial p/\partial x\,,\ V\ \partial p/\partial y\,,\ V\ \partial p/\partial z.$ Here, by H we denote the enthalpy per unit mass,

$$H=(\gamma/(\gamma-1))(p/\rho)+(1/2)(u_x^2+u_x^2+u_z^2)$$

Our trillion-cell simulation gives us a 2640^3 grid of such briquette-averages for each of these variables, and we have saved averages of f over briquettes of just 2^3 cells, because we compute it effectively at double resolution, due to PPB's 10 moments per grid cell.

We used data of this type in the past to construct and validate models of the statistical behavior of single-fluid, compressible turbulence (cf. [29-31]). In that work, we formed moving averages of such variable combinations over cubes of 32^3 cells. Within each such averaging bin, we constructed approximations to the 10 lowest-order moments of each variable combination, based upon the 8^3 participating briquette averages. Using these 10-moment-determined 3-D quadratic forms to represent the result of an idealized numerical method computing on a grid 32 times coarser than our original grid in each dimension, we evaluated the appropriateness of various closure assumptions that might be applied to such a numerical computation. The result was the subgrid-scale turbulence model reported in [29-31]. We believe that an extension of this procedure to our more complicated multi-fluid flows reported here should be useful in validating turbulence models that have been proposed for multi-fluid flows like these in an LES type of computation.

Convergence under Grid Refinement

We have referred in the above discussion at more than one point to simulations of these ICF test problems on coarser grids. Here we make a limited set of direct comparisons in order to establish the degree to which the simulations presented above have converged. In Figure 16, we show equatorial slices for both Problem 1 and Problem 2 from computations with different grid resolutions. These have been imaged in the same way, so that for each problem, the two computations can be directly compared. For both problems, convergence of the behavior inside the outer surface of the dense shell is very good. This is remarkable given the complexity of these flows, but from our earlier code comparison study in 2D [1] it is not unexpected at these grid resolutions. The computation at the top left, on its 2048^3 grid, took 9 hours on the Minnesota Supercomputing Institute's 1024-CPU machine. A calculation of this scale could be performed without comment at any leading university in the country, and at any national lab. The simulations at bottom left and top right in Figure 16, with their 4160^3 and 4800^3 grids, took 8 and 12 hours, respectively, on small fractions (< 1/6) of the NSF's Blue Waters machine. A year ago, these would have been very large calculations, but

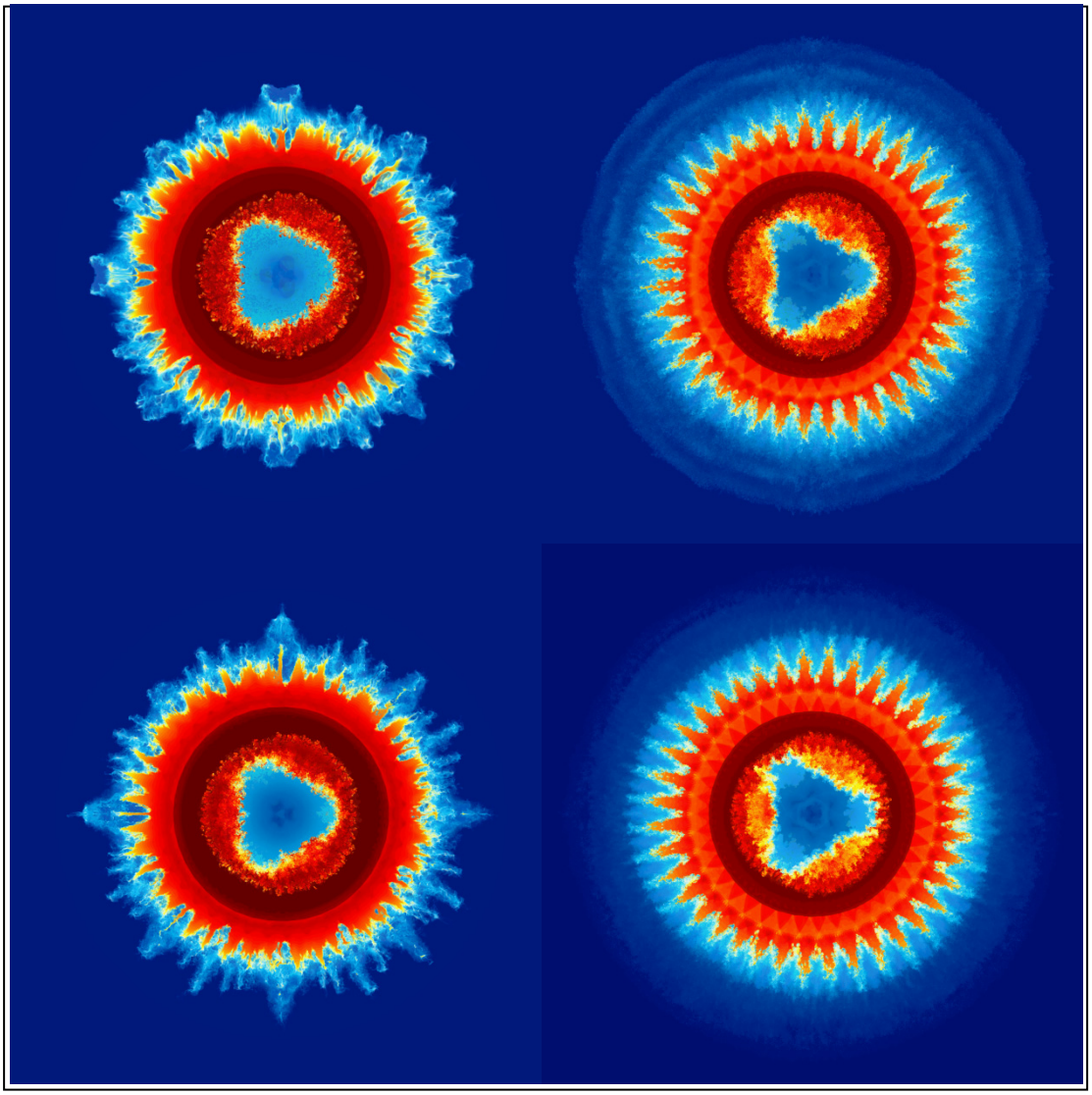

Figure 16. Views of density in equatorial slices for Problems 1 (left) & 2 (right) at t = 0.0525 (left) and 0.050 (right). The grid resolutions are 2048^3 (top left) and 4160^3 (bottom left) for Problem 1, and 4800^3 (top right) and 10560^3 (bottom right) for Problem 2. It has taken longer in Problem 1 for the mixing region at the inner surface of the dense shell to reach the size of that shown at the right for Problem 2. This is due to the smaller initial perturbation introduced in Problem 1. See text for discussion.

they are dwarfed by the simulation at the bottom right, with its 10560^3 grid. Today, this is a big calculation, but in two or three years it will no longer seem so.

The behavior of the outer surface of the dense shell, as shown in Figure 16, has not converged for either calculation of Problem 1. If our interest is focused instead on the behavior of the inner regions, this is of no consequence. Failure to converge in Problem 1 on the correct outer surface behavior does not, apparently, affect the behavior of the inner regions. In Problem 2, we do achieve a high measure of convergence at the outer shell surface, in a statistical sense of course. The points of the compass are very slightly evident in the far outer fringes of the mixing region at the outer shell surface in the lower-resolution run at the top right in Figure 16. However, the falsification of this part of the flow is minor, and the densities (and hence the material mass) involved is trivial. The extent of this outer mixing region is nevertheless the same in both runs at the right in Figure 16, and hence the coarser grid still gives a

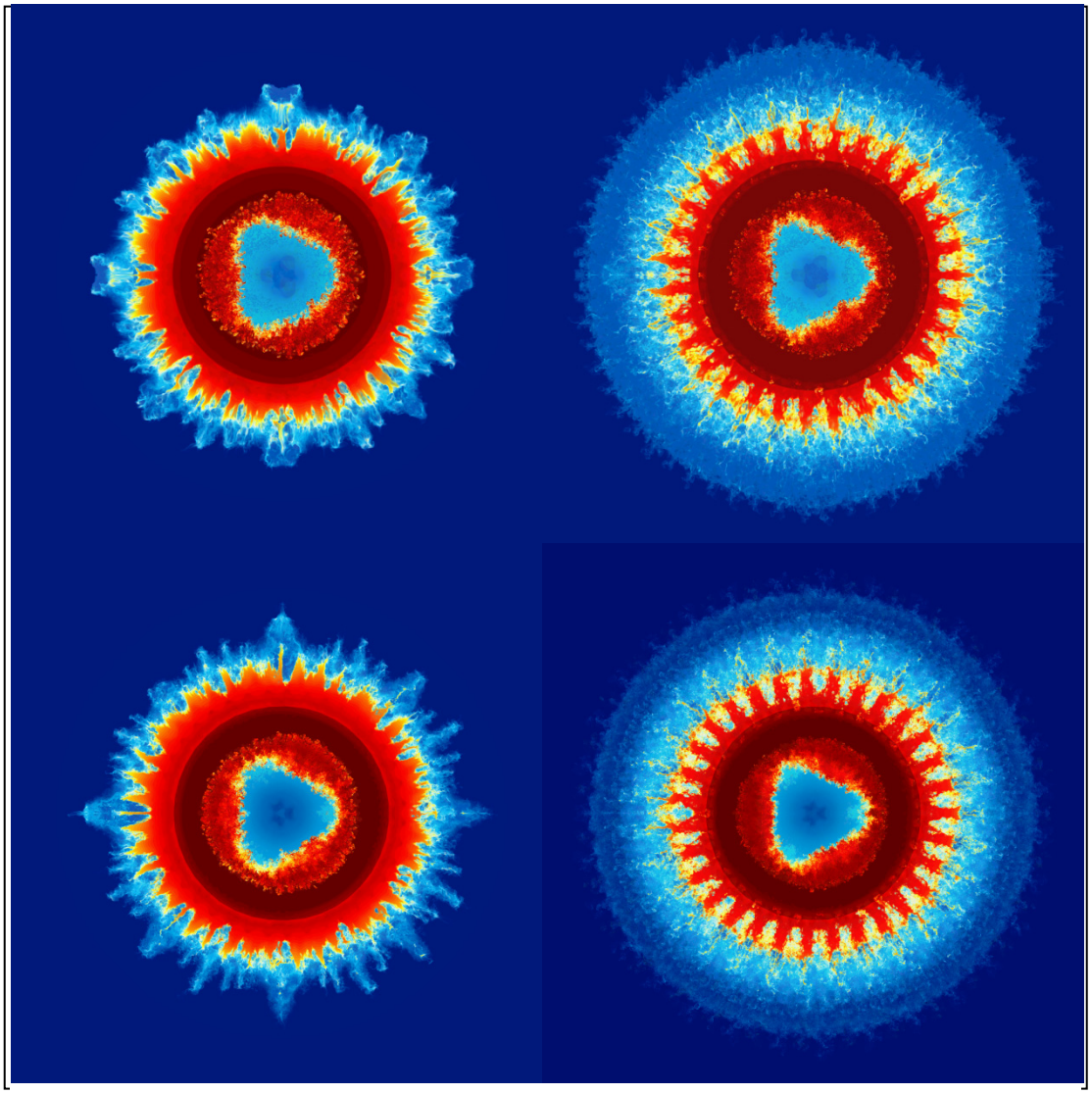

Figure 17. Views of density in equatorial slices for Problems 1 (left) & modified 2 (right) at t = 0.0525. The grid resolutions are 2048^3 (top left) and 4160^3 (bottom left) for Problem 1, and 2688^3 (top right) and 5376^3 (bottom right) for the modified Problem 2. In this comparison, the lower-mode perturbations of the inner shell surfaces are identical. See text for discussion.

good prediction of the extent of mixing here, although not quite so good a prediction of its detailed structure in the far outer region.

In comparing the runs at bottom left and top right in Figure 16, which were performed on grids of comparable resolution and at comparable computational cost, we see that the addition of the very high-order mode in Problem 2 delivers dramatically improved results. It is true that the initial perturbation amplitude on the inner shell surface was 2.5 times larger in Problem 2, but we do not believe that this has played a major role in producing such good results at the outer surface. As evidence we note the close similarity of the inner flow structures that is seen for Problems 1 and 2 in Figure 16, which we find by displaying results for the two problems at slightly different times.

To clarify this issue, we performed an additional mid-sized run, with a grid of 5376^3 cells, in the style of Problem 2. In this new run, we used the amplitudes of the mode (81,39) and

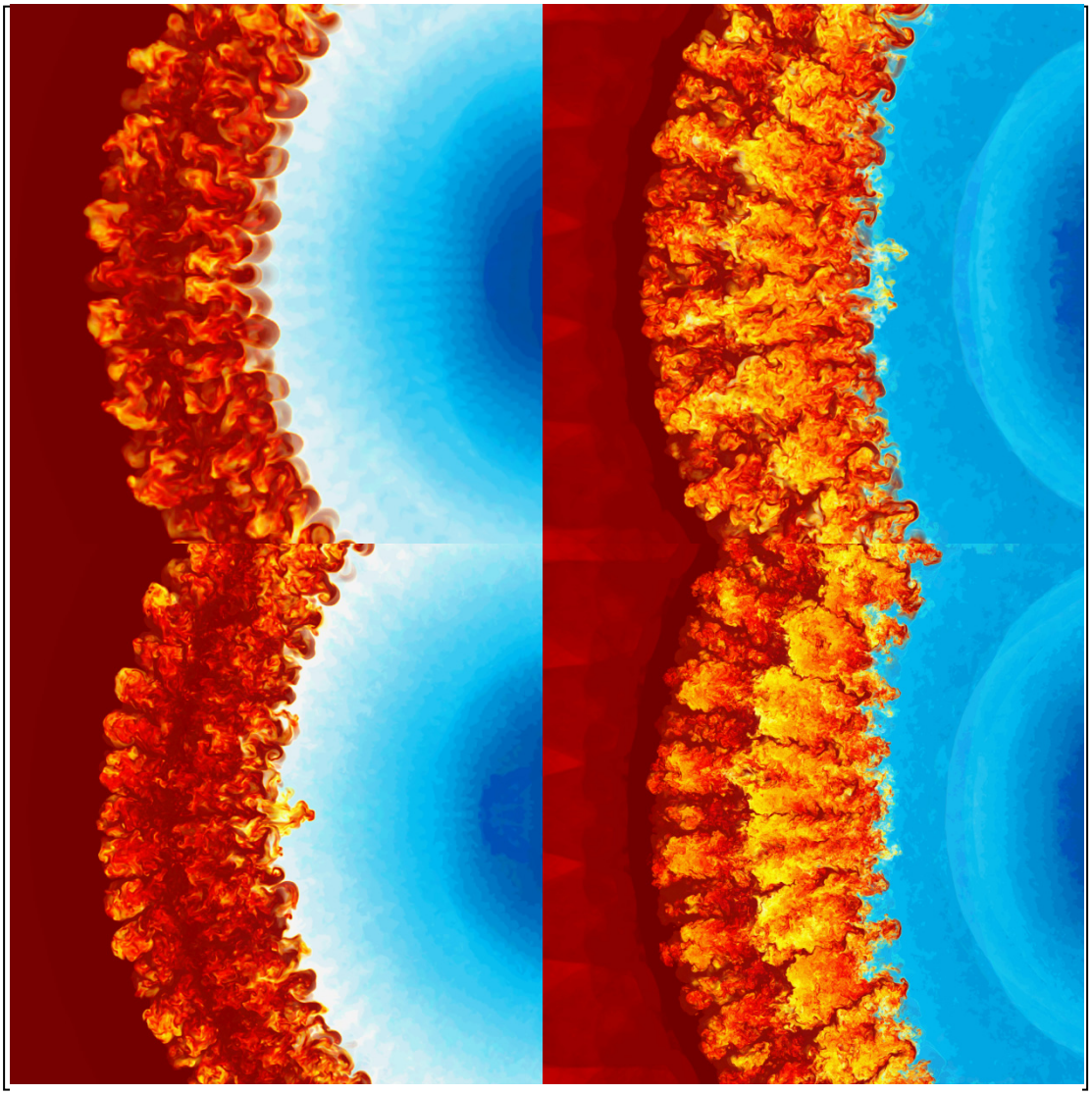

Figure 18. Views of density in equatorial slices for Problems 1 (left) & 2 (right) at t = 0.0475 (left) and 0.045 (right). The grid resolutions are 2048^3 (top left) and 4160^3 (bottom left) for Problem 1, and 4800^3 (top right) and 10560^3 (bottom right) for Problem 2. It has taken longer in Problem 1 for the mixing region at the inner surface of the dense shell to reach the size of that shown at the right for Problem 2. This is due to the smaller initial perturbation introduced in Problem 1. See text for discussion.

(74,36) perturbations of the inner surface of the dense shell that are used in Problem 1. We also experimented with adding a range of very high frequency and small amplitude modes to both the inner and outer shell surfaces, in the style of Problem 2. In addition to the mode (65×11, 31×11) used in Problem 2, we introduced modes (65×m, 31×m), where m = 5,…,10. The idea was to give the outer mixing region that develops several high-order modes that can interact in order to arrive more rapidly at a chaotic result. Nevertheless, each new mode is a harmonic of mode (65,31), so that that mode's periodicity will show up and allow us to verify that all these modes are being treated accurately by the numerical scheme. The overall amplitude of the sum of all these mode amplitudes was set for both the inner and the outer dense shell surface at 1% of the equatorial wavelength of mode (65×11, 31×11). This new disturbance therefore is essentially visually undetectable, like the high-order mode disturbance in Problem 2 on the outer shell surface. The individual mode amplitudes were

taken in the ratio of their equatorial wavelengths, and they were introduced with alternating signs.

A comparison of this new run, which we will call Problem 3, with the results for Problem 1 is shown in Figure 17. For Problem 3, on the right in the figure, we have two different grid resolutions, 2688^3 and 5376^3 cells. The larger of these runs took only 6 hours running on 456,192 cores of Blue Waters. The code ran 10% slower than expected, presumably because the portion of the machine incorporating GPUs was being upgraded, so that the toroidal communications network of the machine "had a big hole in it." Our high-order perturbations in Problem 3 reach down to half the wavelength of the one we add in Problem 2. In aggregate, they also emphasize periodicity at a 5-times-lower frequency still. Thus the high-order modes produce 31 fingers extending outward from the outer shell surface, rather than the 39 we would expect from the imprint on this surface of the low-mode perturbations of the inner surface. Nevertheless, the behavior of the inner-surface mixing region appears to be essentially the same in all four runs shown in Figure 15. Overall convergence for the two runs of Problem 3 at the right in this figure is also quite good.

Figures 16 and 17 show results near the end of each computation when the capsule is close to its smallest size. The general flow convergence these images establish does not imply that details of the mixing regions or of the gradual transition from well-separated dense spikes to well-mixed fluid that is chronicled in Figure 13 has converged. To address this issue, we show close-up views like those in Figure 13 for the 4 runs of Figure 16 at times near the middle of the transition to a well-mixed layer. These can be seen in Figure 18. Again, we show Problem 1 results at a slightly later time, to account for the smaller initial perturbation used in that case. Because these mixing regions are in the process of becoming chaotic, we cannot expect individual features to correspond precisely. Nevertheless, we can see that the correspondence is remarkably good for both pairs of runs. The finer grids do of course deliver more detail, but the positions, thicknesses and rough internal structures of these mixing regions are very similar. Our trillion-cell grid at the bottom right for Problem 2 represents a one-time experiment that establishes that the more affordable smaller grids may be used with confidence to make predictive statements about this type of flow. We need not reestablish this confidence level with each lower resolution experiment, but we must establish it at least once. The trillion-cell run additionally provides us with an exceptional data set against which we can compare the assumptions and predictions of subgrid-scale models of turbulent, compressible mixing, as we have mentioned earlier.

In our trillion-cell run, we used relatively large initial perturbations in order to produce a large mixing region in which we can gather statistical data for use in testing and developing subgrid-scale models of turbulent compressible mixing. For inertial confinement fusion (ICF) such large mixing regions are not desired outcomes. The question therefore arises whether using our code, our grids, and our technique of introducing initially undetectable perturbations to regularize the simulated flow behavior, we can accurately address problems in which the initial perturbations of interest are much smaller. To address this question, we ran our Problem 3, described above, with the perturbations in modes (81,39) and (74,36) at the inner shell surface reduced by a factor of 4 to 0.5%. We will call this Problem 4. These perturbations are detectable, but very small. We also reduced the amplitudes of all our high-order mode perturbations by a factor of 2, also to 0.5%. The result of this experiment is shown in Figure 19.

The results shown in Figure 19, for a run on a 5376^3 grid, clearly demonstrate that our PPM code successfully follows this much smaller perturbation's growth with a near total absence of numerical artifacts arising from its Cartesian grid. We note that the poles of the sphere are special due to the nature of the perturbations and the symmetries of this problem. They will

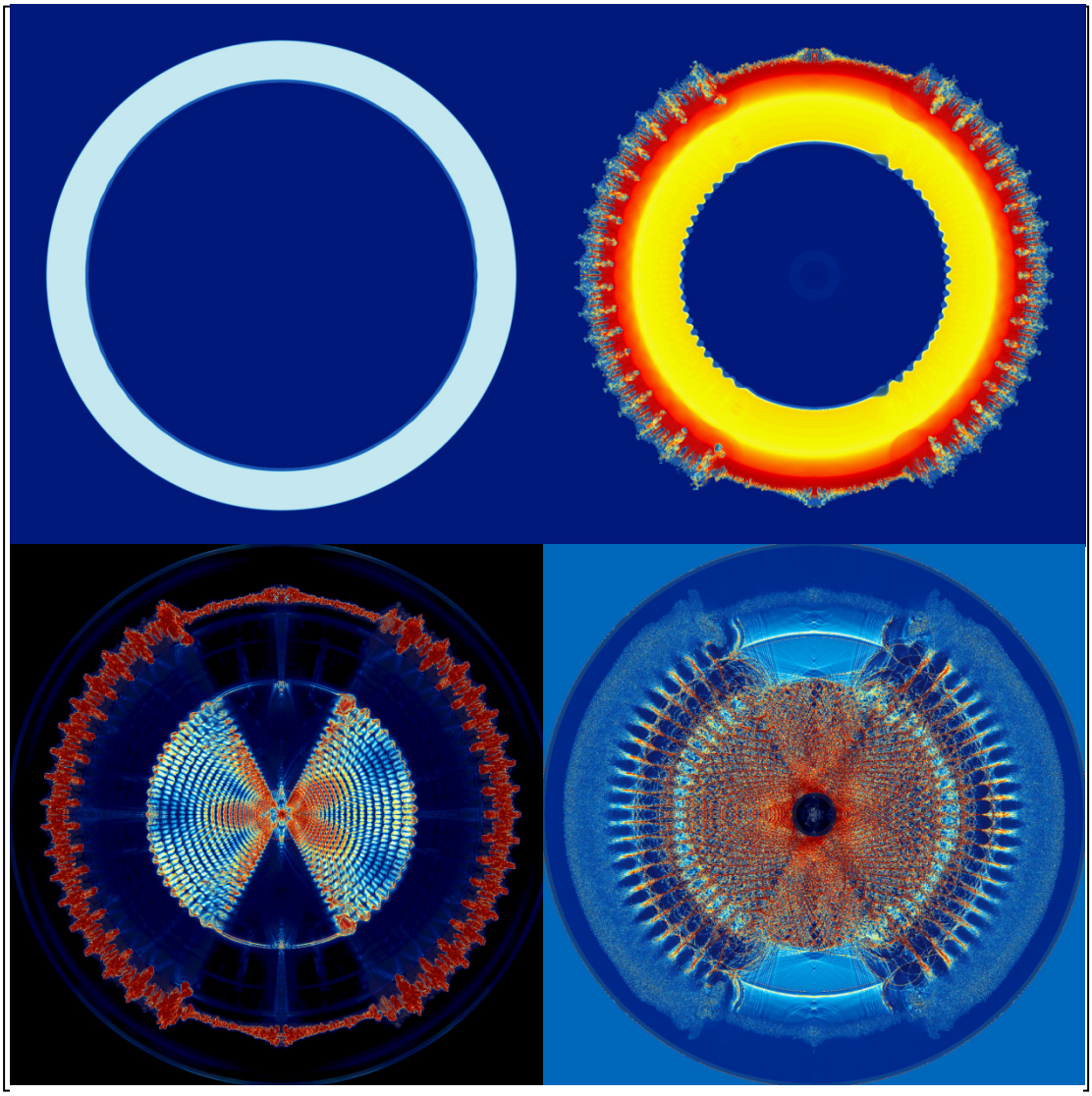

Figure 19. Views in longitudinal slices, along the "prime meridian," for Problem 4 at t = 0 (top-left) and 0.035. The density is shown in the top images, the vorticity magnitude at the bottom-left, and the negative divergence of velocity at the bottom-right. The grid resolution is 5376^3. Here we see that our PPM code can easily handle initial perturbations with amplitudes as low as 0.5% of their equatorial wavelengths across a range of wavenumbers. Numerical artifacts that one might expect in a Cartesian grid computation of this sort are essentially completely absent, save for the polar jets that would occur, we suggest, in any system of coordinates. Similar features at the equatorial crossings, where the unstable shell surface is also tangent to the grid planes, are notable for their complete absence. The image at the top-left, showing the initial state, should make clear that the initial perturbations we follow in this problem are extremely small. See text for discussion.

be special in any set of coordinates. Our Cartesian coordinate system most likely emphasizes this specialness of the poles far less than would, for example, spherical polar coordinates. Thus we suggest that the features at the poles are not numerical artifacts. We can test this assertion by rotating the initial state, so that its poles no longer line up with the coordinate axes. We plan to carry out this experiment in the near future, but our present opportunity in the Blue Waters friendly user access period has expired.

In Figure 20, we show the results for Problem 4 at later times. In this simulation, we have initial perturbations that are 10 times smaller than those we impose in Problem 2. Because

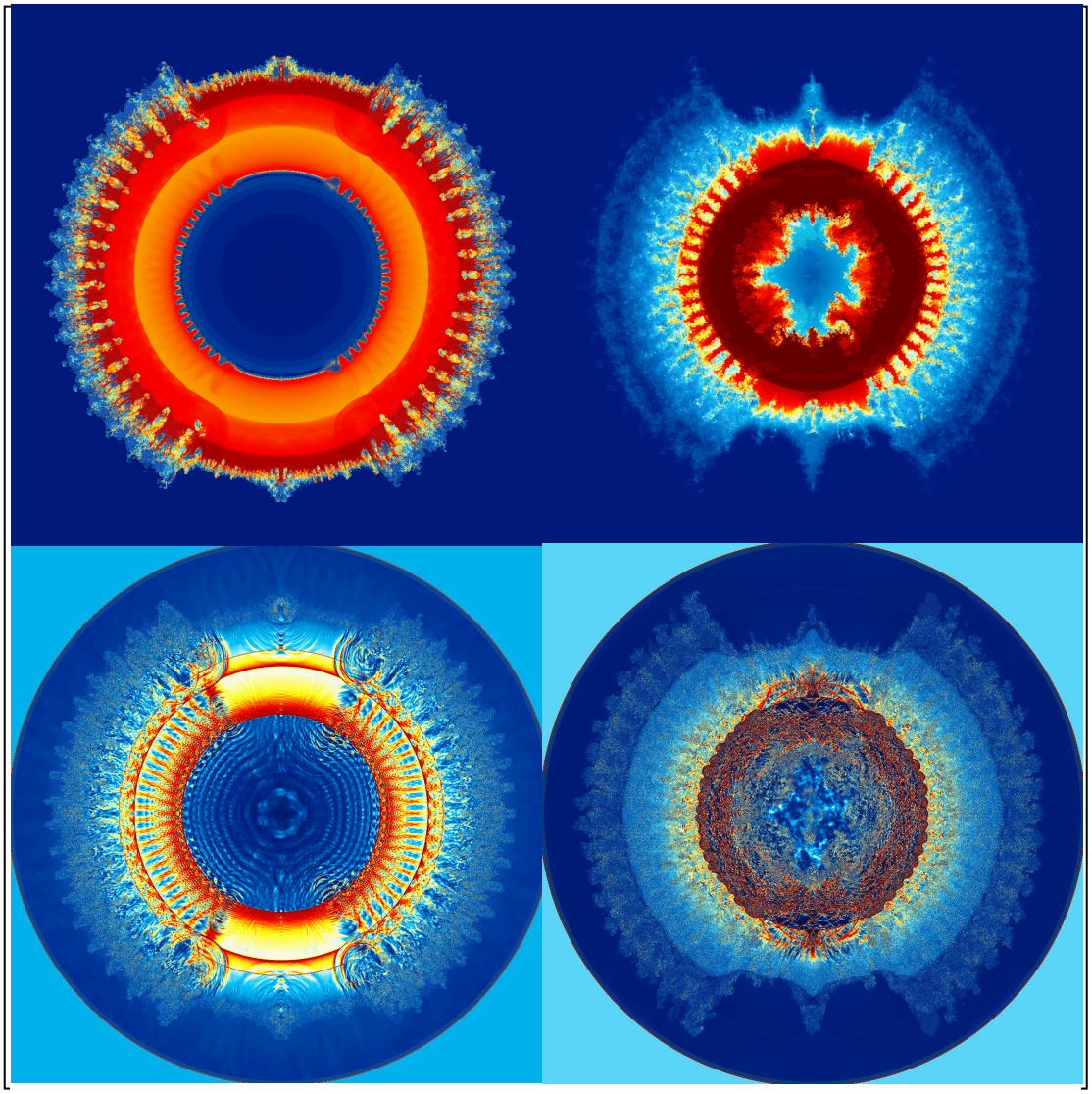

Figure 20. Views in longitudinal slices, along the "prime meridian," for Problem 4 at t = 0.040 (top-left) and 0.055. The density is shown in the top images and the negative divergence of velocity at the bottom. The grid resolution is 5376^3. Here we see that our PPM code can easily handle initial perturbations with amplitudes as low as 0.5% of their equatorial wavelengths across a range of wavenumbers. At the left, we see that the strong shock reflected from the origin is just beginning to interact with the unstable inner surface of the dense shell. At the right, this interaction has produced a sizable mixing region dominated by the mode-3 difference frequency between the two initial modes.

the flow is highly nonlinear, we cannot, however, expect our final mixing region to be ten times smaller. In Figure 20, we see the flow in longitudinal section at the time when the strong shock reflected from the origin first strikes the inner surface of the dense shell and again at the time when the shell reaches about its maximum compression. The Richtmyer-Meshkov fingers are much smaller than in Problem 2 when the reflected shock from the origin first strikes them. We must therefore wait a little longer for the mixing region to grow, but grow it does. By comparing the simulated behavior near the poles with that near the equator in Figure 20, we can get an idea of how much further we might be able to reduce the initial amplitude of our perturbation on this 5376^3 grid and still be able to compute the perturbation unpolluted by signals of numerical origin.

PPM Code Implementation

It is often thought that the primary challenge of achieving high performance on petascale computing systems is that of identifying and exposing sufficient computational parallelism. However, in this work we have used a moving, uniform, Cartesian grid of immense scale. Exposing parallelism, although nontrivial, is not really challenging in this case. Instead, we find that the most difficult task is finding ways to coax modern multicore processors into delivering a reasonable fraction, between 10% and 30%, of their rated peak performance. Our PPM code achieves 12% of the peak 32-bit performance of the AMD Interlagos CPUs in the Blue Waters machine even in a full-system run. We include all sources of run time, not only CPU time, in the denominator of our petaflop/sec measurements, so that when the code reports 12% of peak rated performance, it must be running considerably faster on each CPU core. On Intel-Nehalem processor cores, roughly twice this fraction of rated peak performance is achieved by our code. We cannot definitively explain this observation, but suspect that it is related to the different numbers of words per CPU clock tick that can be transferred between the L1 data cache and the registers on the two different CPU cores. That this could be the rate-limiting feature of a computing platform for our code means that:

1) We must have completely overcome the limitation of memory bandwidth between each CPU core and its locally attached main memory.
2) We must have completely overlapped MPI messaging with computation.
3) We must have completely overlapped all disk I/O with computation.

We are quite convinced that, at a 95% level of confidence, all three of these assertions are true, at least on the Blue Waters machine running 87,846 MPI processes on 21,962 nodes, with 702,784 cores. Analysis of the code implementation and performance reveals that while running at this aggregate rate on the Blue Waters system, all that is required to make assertion #1 true is that the locally attached memory supply to each of the 8 cores it must feed a continuous bandwidth of 259 MB/sec, which is essentially nothing for a modern CPU. For assertion #2 to be valid, the half-duplex bandwidth supplied continuously to each network node (running four MPI processes) must be at least 99 MB/sec, which is also essentially nothing in today's world, but is nevertheless nontrivial on a toroidal network of the scale of that on the Blue Waters machine. We know from the logs produced by the code that all disk I/O is fully overlapped with computation. For this to be true requires that the machine provide to each of our code's 1331 I/O processes a continuous effective disk bandwidth of 13 MB/sec, which is again "nothing." Even a laptop computer could outpace that disk bandwidth by a factor of 5, but we require it from each of 1331 I/O servers simultaneously and continuously for hours or even days, writing and rewriting restart dump files.

How did we get the requirements to support our code's parallel execution down to these low levels? We did this through a hierarchical structure of the code and the computation, which we impose upon the machine, whether it is built in a hierarchical fashion or not. It is easiest to describe this organization of the computation beginning at the small end and building upwards.

The smallest – in fact, an *atomic*, or indivisible – unit of computation for our code is the grid briquette update. This is associated with an atomic unit of data, the grid briquette record. A grid briquette is a cube of m grid cells on a side. We choose $m = 4$, and we have forced other aspects of the computation to conform to this choice of m. Preferred data processing extents, or "vector lengths," are m, m^2, and m^3 for our briquettes. For our choice of m, these are then 4, 16, and 64. These work well with cache lines of 16 words (in 32-bit mode). They also work well with SIMD engines in CPUs and GPUs of all designs that process 4, 8,

or 16 words simultaneously. Even GPUs, which force the user to provide vectors of lengths 32 at absolute minimum, or 64, at a recommended minimum, are accommodated with our choice of $m = 4$.

The grid briquette record consists of all 16 variables that we update in our PPM+PPB algorithm, with each variable having 64 values prescribed for the 4^3 cells of the briquette. It is a nice coincidence then that a briquette record turns out, on Intel architecture systems like Blue Waters, to be precisely one memory page. The briquette record is the atomic unit of memory access and/or transfer. We never fetch just part of such a record, and no MPI message is made up of partial records. We can easily fit many such records of 4 KB into the on-chip cache of a CPU core. On the AMD Interlagos CPU, this memory, the private L2 cache, is 1 MB. On Intel CPUs it is 256 KB, and on Intel MIC chips, it is 512 KB. Even the smallest of these cache sizes is large enough to accommodate all the program instructions plus the private, cache-resident workspaces for 2 threads. We pack our fluid state data into the briquette records so that all we need to know about a briquette of the grid is in its record.

The following is the procedure that is followed by each individual thread of the computation, executing on a single core (with perhaps 2 or even 4 such threads executing on a core, if that proves useful). Note that the code for steps 1-9 for this procedure contains *no reference* to any thread library or to any parallel processing library such as MPI. This code, however, *does* contain information to enable vector processing using a SIMD engine. The procedure consists roughly of the following steps:

1) Prefetch the briquette that is the neighbor of the present briquette in the direction of the present 1-D pass (the next briquette in the processing sequence). Also prefetch any transverse neighbor briquettes of this one that may be needed by the algorithm.

2) Unpack the present briquette record into cache-resident, tiny, *aligned* 2-D arrays. The fast-running index is for the grid cells inside a single $m \times m$ plane perpendicular to the direction of the 1-D pass. The second index runs over such planes. The second index values are barrel-shifted upon each *grid-plane update*. Thus we unpack the contents of each briquette record into a set of tiny arrays that are circular buffers of grid planes. From any transverse neighbor briquettes, construct aligned grid planes of values that are offset one, two, or more cells in one or more transverse dimensions. These tiny arrays are cache resident because they fit into the cache, we constantly overwrite their contents, and we bring *nothing else* into the cache except for the prefetched briquette records, several of which also fit. Briquette records are expelled from the cache preferentially, because once we unpack them we never reference them again. It is critically important that all our tiny arrays be locally dimensioned and not in any common blocks or structures shared among threads in any way. This guarantees, operationally, that they are always in cache and private to our thread.

3) Do every conceivable operation that the algorithm requires using the newly unpacked grid plane.

4) Using the intermediate results of the previous step, which are placed into cache-resident circular buffer arrays of grid planes, do every possible operation that is newly enabled.

5) Using the intermediate results of the previous step, do every . . .

6) Using the intermediate results of the previous step, do every . . .

7) And so on through as many such operations as the algorithm requires. For our PPM+PPB implementation, there are 9 such steps, each with hundreds of lines of code.

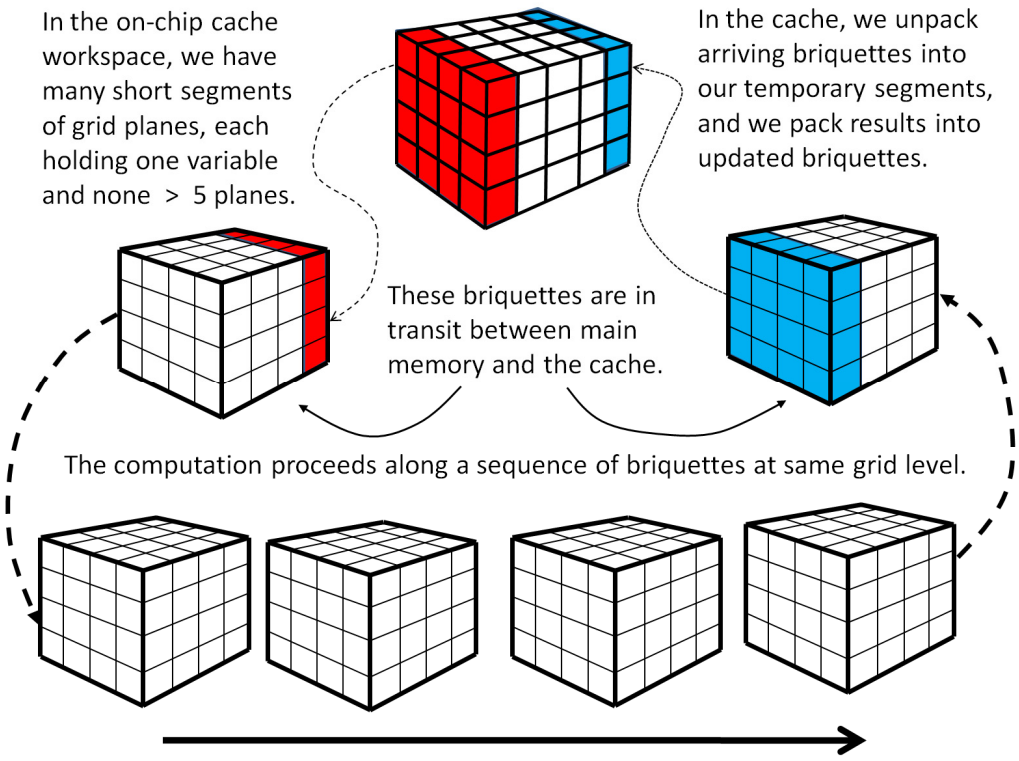

Figure 21. Diagrammatic illustration of the massively pipelined grid pencil update procedure that lies at the heart of the PPM code. A sequence of grid briquettes is shown that can be updated in a pipelined fashion, because they share common faces in a single dimension of a 1-D pass. They are shown separately at the bottom of the diagram, in order to emphasize that they need not reside next to each other in memory. Each briquette, however, has a corresponding data record that *is* contiguous in memory. See text.

8) At the end of this massively pipelined computation, pack the updated values of all the 16 variables into a new grid briquette record that resides permanently in the cache. It may be useful to transpose the data so that the packed grid planes lie perpendicular to the dimension of the next 1-D pass.

9) When the new briquette record is full, write it into a temporary *grid block* data structure. This consists of a rectangular solid of gird briquettes that is a temporary structure that is overwritten constantly and that is *shared* by all the cooperating threads of this MPI process. It is particularly felicitous if this grid block structure can reside permanently in an L3 data cache on the CPU chip.

10) If this is not the last briquette in this thread's *grid pencil* extending all the way across the grid block in the direction of the pass, then go to #1. Before going to #1 or going on, the master thread must call mpi_probe to give the messaging system software an opportunity to grab the CPU core to do its work. This produces a 30% boost in messaging performance on slow systems, and a boost of about 10% on Blue Waters. Of course, this should not be necessary.

11) If this is the last grid briquette, atomically select the next grid pencil to update in this pass.

12) If there are no more grid pencils to update in this pass, begin the next pass by selecting the first grid pencil and spin-waiting until all grid pencils of the previous pass that it

requires to be updated are so updated. Then begin updating that grid pencil by going again to #1.

13) If this is the last of the three 1-D passes in a time step, write back the updated grid briquettes (this is done as they become available, in step #9 above) to the new main-memory array of briquette records. Check the tiny, cache resident look-up table to see if this briquette should also be written into any MPI message buffers, and if so, write it into them.

At the end of each 3-D grid block update, performed cooperatively in a dynamic, self-scheduled way by the 8 threads of a CPU die in the Blue Waters machine, there is an unnecessary thread barrier. This barrier *is* unnecessary, but we find that machines nevertheless like it. Perhaps the threads like to stay close together, so that they can benefit from briquette records that have been brought into the cache by other threads. In any case they run faster with the barrier than without it, and they run correctly either way.

The cooperative 3-D grid block update is highly efficient. No data is ever read twice that only needs to be read once. A similar statement applies to data writing. *Essentially all* flops are SIMD flops with perfectly aligned vector operands of 16 words. The pipelining strategy can be extended to increase this vector length to 32 or 64, but we find that no device, except for a GPU, seems to prefer that. There are cache misses, because we must fetch the briquette records, and they are never in cache. But there are *almost no* cache misses. However, there can be a few L1 cache misses, and there can be L1 instruction cache misses, because we run through the *entire algorithm* on each transit of the loop outlined above, and this has many thousands of instructions. More importantly, there can be register misses. That is, we produce and save in cache for later use an enormous quantity of working intermediate data in our little circular buffers of grid planes. This data is never spilled to off-chip memory, but it is constantly spilled from the registers into the L1 cache. We therefore conclude that this data channel is a bottleneck for our computation, and is the critical rate-limiting feature of any computing platform, except for GPUs, for our code. On GPUs, the relative lack of on-chip data storage is our rate limiter (cf. [32,33]).

The above structure of the computational heart (*not* a kernel) of our computation is extremely complex. We generate it by first writing the relatively simple expression of a Fortran update for a single, isolated grid briquette with a ghost briquette on each side in the direction of the 1-D pass. This expression, which consists of many subroutines, is automatically transformed into the pipelined and memory compressed code outlined above. This transformation is done by our CFD-builder tool, and the resulting code is passed to a back-end code transformation tool that can produce from it C code with calls to vendor-provided libraries of intrinsic functions. These tools have been described in a number of recent reports [34-37]. This methodology removes any dependence of our code on advanced compiler features intended to support new hardware. Such compiler features can become available only fairly long after the corresponding computing platform is in place. We are able instead to take advantage of the intrinsic functions that vendors provide, apparently for the use of intensely motivated programmers, like those who maintain and update libraries of heavily used functions.

In our code, MPI messaging is completely overlapped with computation. We enable the use of non-thread-safe MPI libraries by having all MPI library calls made by the master thread of each MPI process. We have tried to reduce to an absolute minimum the code that this thread must execute while other threads might need to spin-wait idly. All reading from and writing to MPI message buffers is done as part of the grid block update code. This code is dynamically load balanced among the threads as described in the above list of steps followed by each thread in parallel. The master thread makes only a tiny number of MPI messaging calls and, of course, waits on message arrivals. However, the messaging is carefully planned so that

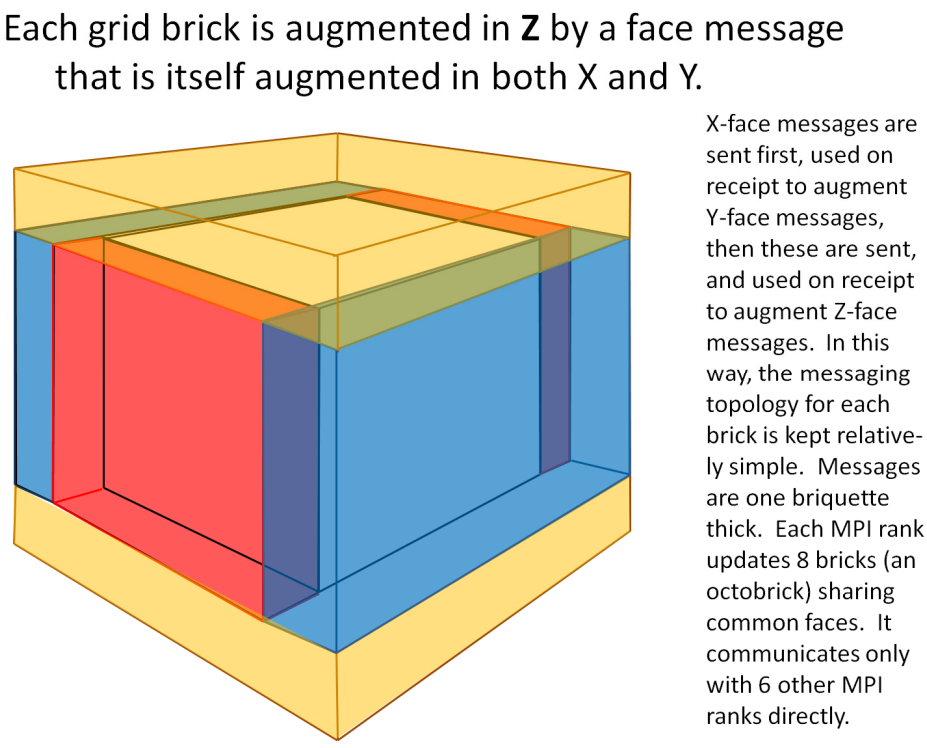

Figure 22. Diagrammatic illustration of the geometrical structure of layers of ghost briquettes that are communicated between grid bricks by MPI messaging. See text.

the master thread need never actually wait; the messages will always simply be there.

We assign to each MPI worker process a grid brick to update. For efficient messaging, we subdivide this grid brick into 8 equal sub-bricks. We call the original brick an *octobrick* and the 8 sub-bricks of it simply bricks. Each time step update consists of a special sequence of 8 brick updates. The update and messaging pattern repeats every 2 time steps. If we designate each brick by its (x,y,z) coordinates in the octobrick, then the repeating sequence of 16 grid brick updates is: (1,1,1), (2,1,1), (2,2,1), (1,2,1), (1,2,2), (2,2,2), (2,1,2), (1,1,2); followed by this list of 8 bricks in the reverse order. There is a general pattern of actions by the master thread after each of these grid brick updates is completed, but there are some exceptions to this pattern. Disregarding the exceptions for the moment, the standard pattern of actions for the master thread is as follows:

a) Wait on receipt of the Z-face message intended for the next brick to be updated in the sequence. We find that it is faster to use mpi_test in place of mpi_wait, and to spin-wait manually upon success. This should not be necessary, but it is undeniably faster code. Presumably each unsuccessful call to mpi_test encourages MPI to move the message along.

b) If the next grid brick update is not the one out of every sequence of 32 upon which we recompute the value of the time step interval, then set the value of a semaphore variable in a separate cache line of an array shared by all threads. This semaphore will indicate to all other threads that the message waited on in step #a above has indeed arrived. (In Fortran, all elements of common blocks are automatically shared, and they also begin on aligned locations in memory, which is useful in constructing semaphore arrays.)

c) Find the maximum of the individual maximum Courant numbers (which limit the time step value) determined by each of the other threads and stored in separate cache lines of a shared array. Then, if we have just completed the first brick update in our sequence of 16 before step #a above, perform a blocking mpi_recv on a message with time step limit information from the special MPI "time keeper" process for this team of MPI worker processes (teams are described below). This message comes in response to one sent on the previous trip through this list of actions just after the first grid brick update in the sequence of 16 was completed. Therefore, it will always simply be there, and although we are prepared to wait for it, we will never have to do that. This message contains the result of a global reduction for the global maximum value of the Courant number appropriate at the time of the previous grid brick update. If the next grid brick update is the one out of every 32 on which we recompute the time step value, then set the semaphore (which was not set in step #b above) to allow the other threads to proceed with the next grid brick update.

d) Wait on the receipt of the X-face message destined for the grid brick that was updated two brick updates previous to this present one, just completed before step #a above. When it arrives, copy briquette records along its edges into appropriate portions of Y- and Z-face message buffers. This small bit of data copying cannot be avoided. It does not hold up the other threads, which will have already begun updating the next grid brick. Dispatch the Y-face message that has been augmented. The augmented Z-face message will be sent off later.

e) Wait on the receipt of the Y-face message destined for the grid brick that was updated 6 brick updates previous to this present one. When it arrives, copy briquette records along its edges into appropriate portions of the Z-face message buffer and send that message off (the message topology is illustrated in Figure 22).

f) Last in this sequence of operations we send off messages from the grid brick we have just updated. Then if we remove the thread barrier at the end of the grid block or grid brick update, we will be able to perform all steps in this list up to this one while other threads continue to finish up the update for the present brick. If we do not have a thread barrier after the grid block or brick update, then here we must insert a spin-wait for the master thread until all other threads signify by setting separate semaphores that they have finished with the present grid brick. We compute the maximum of the Courant numbers found by those threads here in that case, before sending it off in the following step to our team's time keeper.

g) If this is the first brick update of our sequence of 16, then send Courant number information to our team's time keeper. We have not been able to make non-blocking sends work here on Blue Waters, although they should. We simply gave up and used a blocking send. The time keeper is very responsive, since it has almost nothing to do. Consequently, we found no significant performance penalty to use of the blocking send.

h) Send off the X-face message from the grid brick whose update was just completed. This message is ready to go, because the individual briquette records were written into its buffer by the threads as they updated the grid brick. All the master thread needs to do here is to dispatch the message.

This pattern of actions takes some trouble to describe, but it takes essentially no time for the master thread to do all this work. The design is to allow the other threads to do useful labor for as much of this very short time as possible.

There are exceptions to the above pattern of actions that arise from the need on every eighth

grid brick update to update the very same grid brick immediately afterwards in the sequence. This forces us to receive messages for that brick earlier than in the standard procedure listed above. This is a slight complicating factor upon the code expression. However, in no case does a message have less time than one complete grid brick update interval in which to reach its destination. Our large, trillion-cell run on Blue Waters used a grid brick size of 30^3 briquettes, or 120^3 cells. Thus each MPI process updated 240^3 cells, with 4 of these on each network node. Experimentation showed that we can reduce this grid brick size with essentially no loss in performance to 14×28×14 briquettes. This gives messages of equal size in each dimension for the node as a whole, but messages have only a quarter of the time in which to arrive, while they are only 2 times shorter in length. This computation has a half-duplex continuous messaging bandwidth requirement of 148 MB/sec to each node. Strong scaling of our application is thus possible until we hit the messaging bandwidth limitation somewhat below this grid brick size. At the grid brick sizes we use in this study, all MPI messaging is completely overlapped with computation.

We overlap disk I/O completely with computation as well. We organize the worker MPI processes into *teams*. Each team updates a rectangular solid sub-domain of the problem. We demand, for historical reasons, that each team have the same *even* number of members in both X and Z, and we allow it to have any number of members in Y. Correspondingly, we demand that the octobrick that is updated by each team member have the same number of briquettes in X and Z, and we allow it to have any number of briquettes in Y. The MPI ranks of team members are mapped inside the code to octobrick subdomains in a way that guarantees that each set of 4 team members update octobricks that share half their X- and Z-faces, and we demand that the next set of 4 team members update octobricks that all share Y-faces with those of the previous 4 team members. We assign team members in this fashion for however many teams it takes to provide a sufficient number of team leader and team time keeper processes to completely fill one node. These layout settings are prescribed via C-preprocessor parameters before recompiling the code for each run. Recompilation takes about 2 minutes.

Each team has a team leader process and a team time keeper process dedicated to its support. These lightly used processes allow the worker processes to carry on their work without essentially any interruption. They are consequently an overhead cost of our parallel code implementation. How high this cost is set depends upon the capability of the computing platform and the frequency and volume of desired disk output data. On Blue Waters, we experimented with different layouts. Our experience at modest scale (but our code proved almost perfectly scalable, once the overhead of these additional MPI processes is accepted) indicated that having just one team leader per 4 disk file system object storage targets (OSTs) was marginally efficient for ICF runs like those reported here. Experiments indicated that 2 OSTs per team leader might work well, but we decided to be conservative and to have one team leader per OST. The primary motivator for this choice was to get the time to generate a restart dump down to around 17 minutes, so that not much time is wasted upon each problem restart.

Each team leader MPI process is exclusively responsible for all disk I/O for its team. This includes both archivable output data and restart dumps. For our large run on Blue Waters, each of our 1331 teams had only 64 workers to serve. The team output for visualization and quantitative analysis was 12.4 GB, and for a restart dump was 72 GB. The first set of data is highly compressed. Each worker, upon arriving at the time when this data is generated, produces the data in the course of its normal, pipelined computation described earlier. This data is held in its memory until the time of the next such output point, which for our large run was between 30 minutes to one hour later. It is "sent" to, or more properly offered to (i.e.

mpi_isend), the team leader as soon as it is ready. The team leader has all the time between this dump and the next one to accept from (mpi_recv) each team worker this data in a prescribed order. As soon as the team leader gets this data from a worker, it writes the data into a buffer, saving up such data until it has about 100 MB to write into each of the 7 files it is writing for the team as a whole for this time level. Restart dumps are created in this same way. Each worker process has about one extra GB of memory devoted, for our large Blue Waters run, to holding onto the restart data until the team leader gets around to receiving it. This is no problem, because Blue Waters nodes have lots of memory. Holding this data in memory allows the workers to go on with the computation while the team leader produces the report on the disks.

In our large Blue Waters run, we had 66 MPI processes for each team of 64 workers updating 960^3 cells. The overhead of this parallel implementation was therefore 1/32. or 3%. This was essentially the only parallel overhead, and we pay it even for a run with just a single team. Because we pay this overhead up front, the code is nearly perfectly scalable from one team to 1331 on Blue Waters. Had we chosen to have 128 workers per team, which our experiments indicated would not cause any workers to be held up by their leaders, then our overhead would have been only 1.5%. However, it would have taken about a half hour, rather than about 17 minutes, to write a restart dump with this layout. Then a minimum of one hour would have been wasted upon every problem restart. We decided that the extra cost of smaller teams was worth the associated saving in restart time. Had it been possible for us to place our team leader processes on the system's dedicated I/O nodes, we would have been able to save this overhead cost by hiding it from the center's charging algorithm. In any case, the overhead is small and affordable.

Looking Ahead to AMR

Our PPM code has been designed to support an AMR mode of operation in the style of the RAGE code at Los Alamos [38]. We do not compute the locations of our briquettes in our data structures, but instead we find them in look-up tables that are small enough to be cache resident. Our design plan is to demand that each briquette be at a single grid refinement level. If it is selected for refinement or derefinement, then these operations must be performed on the entire briquette. In this way, we will preserve the highly efficient, and massively pipelined core of our code. This is one reason why we have built the computation described in our list of operations around a temporary grid block data structure that can be resident in an L3 cache. At the level of this temporary grid block data structure, we can have our threads go through the briquette refinement and derefinement operations that are dynamically required to update the grid block in an AMR fashion. Each such operation can be SIMDized for very efficient execution, and all its needed data can be in cache. Our cooperating threads are already dynamically load balanced, so this need of an AMR execution will pose no additional problem.

The main additional feature we will need to add to support large AMR runs will be dynamical load balancing at the level of the entire machine. This type of computation is by now a fairly mature technology (see, for example, [39]). In the context of our code, we will make use of the hierarchical structure of teams that we already have. Our team leaders are relatively swamped with I/O tasks. They are dealing with very large messages already, and are fairly busy most of the time. Our team time keepers, on the other hand, have almost nothing to do at present. These processes can maintain the list of octobricks to be updated, the identities of team members to whom these updates are or have been assigned, the time intervals that these previous assigned updates have required, etc.

Without overtaxing these present time keeper processes, we can have them act as intermed-

iaries. Team members can receive octobrick update assignments from them, and if they do not own those octobricks, find from whom they must get the data without further involvement of the time keeper. The time keeper may also assign octobrick acquisition tasks, so that selected workers speculatively prefetch additional octobricks for the team. The time keepers are already constantly in touch with all other time keepers, through the time keeper of team 0. It would be no extra real trouble to receive updates of the octobrick layout, in a highly compressed form, over the entire machine. Using this data, the time keepers can coordinate speculative duplication of octobrick data and decide who ultimately will own that data after this time step pair is completed.

Programming the dynamic load balancing over the machine that is envisioned above is no small task. It is not clear that anyone can or has already written this program for us. However, such programs have been written, so we know it is possible. It will put additional stress on the machine's interconnect, but we see no reason that code execution cannot remain highly efficient. It may make sense to impose some constraints upon how unequal loads are allowed to grow. We already demand that each briquette be entirely on a single grid refinement level. It might make sense, at least for some problems, to demand that we fill in coarse briquettes with refined ones until we meet a requirement for having just two grid levels per grid brick. This should not introduce too much extra computation for grid bricks of, say, 6 grid briquettes on a side on the coarser of the 2 levels.

The ICF problems of this study can provide an example of how this might work. We could demand that the grid become one level finer where we first encounter the dense shell gas in coming inward from the problem boundary condition region. Then we could demand that the grid be refined one more level at the point where we first encounter the enclosed, inner fluid. This would give us our finest grid in the mixing region on the inner surface of the ICF capsule. During the entire course of this problem, up to and including time 0.055833, shown in Figure 7, we would not need more than 2 grid levels in any grid brick in the problem. If we were to need more than 2 such levels, we could of course, and at minimal cost, fill in the coarsest level of the 3 in this brick alone.

The largest possible load imbalance between individual grid bricks in the strategy just outlined would be a factor of about 8. One might then think that this would require us to assign 8 bricks to each participating MPI worker process. We note that we are already assigning 8 bricks to each MPI process even for our uniform grids. However, these are contiguous bricks, and hence if one has a grid refinement surface passing through it, the others are likely to have this also. We can get an idea of how extensive load imbalances would be by viewing the equatorial slices in Figure 17 and longitudinal slices in Figure 20 of our ICF problems. Using the grid refinement strategy outlined above, we would have just 3 grid levels, and the surfaces on which these occur would roughly be two concentric spheres. For Problem 1, inspection of the images at the left in Figure 17 shows that our outer grid refinement surface would depart significantly from a sphere, but the thrust of our argument in this article is that we would instead be solving problems more like Problems 2-4. For these problems, the grid refinement surfaces would be very nearly spherical, concentric, and well separated from each other for the vast bulk of the problem evolution.

The consequence of the topology of our ICF problems, coupled with the grid refinement strategy that they necessarily suggest, is that we would find that almost all of our octobricks are purely at a single grid level, and hence in excellent load balance. We would have a small set of octobricks for which the computational loads would be greater by less than 25% at most, or lesser by about a factor of 8 at most. This really is not a difficult load imbalance to handle operationally. Taking our 5376^3 grids used for the results in Figs. 17 and 20 as an example, we have an array of $12\times6\times12$ teams, each with 4^3 octobricks of $112\times224\times112$ cells

each. Our grid refinement surfaces at around time 0.0525 would have approximate radii of (0.42/1.5) and (1.15/1.5) as fractions of the problem domain radius. If we consider the average radial extent of an octobrick to be roughly (4/(3×24)) × sqrt(2) × 1.5 = 0.118, then our 2 grid refinement regions consist of fractions of the total problem volume equal to

$$\left(\frac{4\pi}{3}\right)\frac{((a+0.059)^3-(a-0.059)^3)}{27}$$

for $a = 0.42,\ 1.15$. These domain volume fractions are then roughly 1% and 7.3%. Thus we see that these regions, as a fraction of the whole, are small. We therefore have at most 8-fold load imbalances in less than an eighth of the volume. In these volumes, loads might be up to 25% greater or 87.5% lighter. An obvious strategy would be to simply do no load balancing at all, let up to 8.3% of the workers finish early and then be idle, and wait upon any who take 25% extra time. We do not recommend this strategy, but we note that its extra cost is not at all great. We would need to be redistributing our octobricks among workers in any strategy, but the cost differential of each octobrick update could, at the worst, simply be ignored. We also note that on our moving base Cartesian grid, these grid refinement surfaces are barely moving at all, as can immediately be seen from Figures 3-7. Thus we could take a relaxed approach to our redistribution of octobricks as the problem evolves. This is not to say that AMR implementation for this problem is easy. We merely point out that the resulting load imbalances from a very natural strategy of grid refinement are small and not very highly dynamic in nature.

The above arguments, of course, apply only to our ICF problems used in this study. However, it is likely that many problems have such natural simplifying features. One example where AMR is productively employed at present is in simulating the development of large-scale structure in early universe models (cf. for example [40,41]). In these problems, when the base expansion of the space as a whole is taken out, which is easily done, there is very little residual motion. AMR is indeed called for, but there are very strong simplifying features to the problem that can be exploited by an AMR strategy.

Conclusion

In this work, we have studied simplified test problems motivated by the inertial confinement fusion (ICF) process. We have not attempted to simulate in detail any specific laboratory experiment. Doing so would be immensely more difficult, as the ICF problem involves very complex physical processes that are not represented in our simulation code at all. Thus our results should not be over-interpreted as bearing directly on ICF problems. Instead, our goal has been to investigate a specific aspect of the computational challenge of ICF simulation: namely, the problem of accurately and conveniently representing unstable multifluid boundaries on a grid in the strongly converging spherical geometry essential to success in ICF.

We have shown through multiple numerical experiments that computations on Cartesian grids are definitely capable of preserving the important symmetries of these converging flows without introducing significant numerical artifacts. Two mechanisms are key in this regard: (1) we introduce at each unstable multifluid boundary a very small amplitude and very high frequency perturbation or perturbations that would be beneath detectability in the initial state of a corresponding laboratory experiment, and (2) we use a very fine computational grid that moves in a simple fashion along with the general flow convergence. Experiments that space does not permit us to report here have established that in addition to these features of our simulations, a smaller role is also played, in order of decreasing importance, by: (3) the high-order representation of the multifluid interface using PPB advection, with its explicit subcell resolution, (4) the smart dissipation added at slowly moving strong shocks by the PPM version we use, and (5) performing the computation in a frame of reference that

moves the grid alternately back and forth slightly in each dimension and in a fashion that does not significantly increase the Courant number ("grid jiggling").

The full ICF problem involves multiple additional physical processes. We suspect that all of these additional processes are easier to implement in a code using Cartesian grids. Our study to date says nothing about the accuracy of treating those aspects of the full problem in Cartesian coordinates, but we believe we have shown that there is no significant drawback to treating the hydrodynamics this way. This conclusion has been made possible only recently, by a combination of advanced algorithms such a PPB fractional volume advection for treating multifluid interfaces and advanced computing platforms like Blue Waters, which make the necessary fine grids eminently affordable. We believe that our special restructuring of the code for pipelined SIMD processing in parallel and at scale has played a major enabling role as well.

Acknowlegements

The construction and development of an interactive supercomputing system at the LCSE at the University of Minnesota has been supported by an NSF Computer Research Infrastructure grant, CNS-0708822. This system has proved vital in the development and testing of our PPM code. Our work has also been supported in part through the Minnesota Supercomputing Institute as well as by access to Los Alamos computing systems. We are grateful to acknowledge access to the NSF Blue Waters sustained petascale computing system, which has made the large simulations in this report possible. This has come through an NSF PRAC grant to Woodward, NSF/OCI-0832618. Development of our multifluid code with PPB fractional volume advection has been supported by the DoE Office of Science through MICS program grant DEFG02-03ER25569 to the University of Minnesota and also by a contract through the Los Alamos National Laboratory. Research work reported here as well as the development of our multifluid PPM gas dynamics code and translation tools for multi- and many-core processor systems has been supported through a contract from the Los Alamos National Laboratory to the University of Minnesota. Further support for work with our code translation tools has come through a contract from Sandia National Laboratory to the University of Minnesota. Support for special work in optimizing code performance for the Blue Waters system has come from a subcontract from NCSA's Blue Waters project to the University of Minnesota. This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

References

1. C. C. Joggerst, A. Nelson, P. R. Woodward, C. Lovekin, T. Masser, C. L. Fryer, P. Ramaprabhu, M. Francois, G. Rockefeller, 2012, "Cross-Code Comparisons of Mixing During the Implosion of Dense Cylindrical and Spherical Shells," submitted to *J. Comput. Phys*.
2. Youngs, D. L., and R. J. R. Williams, "Turbulent mixing in spherical implosions," *Int. J. Numer. Meth. Fluids* **56**, 1597-1603 (2008).
3. Woodward, P. R., "Numerical Methods for Astrophysicists," in *Astrophysical Radiation Hydrodynamics*, eds. K.-H. Winkler and M. L. Norman, Reidel, 1986, pp. 245-326. Available at www.lcse.umn.edu/PPMlogo.
4. P. R. Woodward, PPB, the Piecewise-Parabolic Boltzmann Scheme for Moment-Conserving Advection in 2 and 3 Dimensions. *LCSE Internal Report*, 2005. Available at:

www.lcse.umn.edu/PPBdocs.

5. Woodward, P. R., F. Herwig, D. H. Porter, T. Fuchs, A. Nowatzki, and M. Pignatari, "Nuclear Burning and Mixing in the First Stars: Entrainment at a Convective Boundary using the PPB Advection Scheme," in *First Stars III*, eds. B. W. O'Shea, A. Heger, and T. Abel, Am. Inst. of Phys. 2008; also available at www.lcse.umn.edu/FSIII.
6. Woodward, P. R., D. H. Porter, W. Dai, T. Fuchs, A. Nowatzki, M. Knox, G. Dimonte, F. Herwig, and C. Fryer, "The Piecewise-Parabolic Boltzmann Advection Scheme (PPB) Applied to Multifluid Hydrodynamics," preprint LA-UR 10-01823, available at www.lcse.umn.edu/PPMplusPPB.
7. P. R. Woodward and P. Colella, The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks, *J. Comput. Phys*. **54**, 115-173 (1984).
8. Sytine, I. V., D. H. Porter, P. R. Woodward, S. H. Hodson, and K.-H. Winkler, "Convergence Tests for Piecewise-Parabolic Method and Navier-Stokes Solutions for Homogeneous Compressible Turbulence," *J. Computational Physics* **158**, 225-238 (2000).
9. Grinstein, F., L. Margolin, and W. Rider, eds., *Implicit Large Eddy Simulation: Computing Turbulent Fluid Dynamics*, Cambridge Univ. Press (2006).
10. Woodward, P. R., and P. Colella, "High-Resolution Difference Schemes for Compressible Gas Dynamics," *Lecture Notes in Phys*. **141**, 434 (1981).
11. Colella, P., and P. R. Woodward, "The Piecewise-Parabolic Method (PPM) for Gas Dynamical Simulations," *J. Comput. Phys*. **54**, 174-201 (1984).
12. Woodward, P. R., "The PPM Compressible Gas Dynamics Scheme," in *Implicit Large Eddy Simulation: Computing Turbulent Fluid Dynamics*, eds. F. Grinstein, L. Margolin, and W. Rider (Cambridge Univ. Press, 2006).
13. Woodward, P. R., J. Jayaraj, P.-H. Lin, G. M. Rockefeller, C. L. Fryer, G. Dimonte, W. Dai, and R. J. Kares, "Simulating Rayleigh-Taylor (RT) instability using PPM hydrodynamics at scale on Roadrunner," LA-UR 11-00061, Proc. NECDC 2010 conference, Los Alamos, Oct., 2010, preprint available at www.lcse.umn.edu/NECDC2010.
14. Ramaprabhu, P., G. Dimonte, P. R. Woodward, C. L. Fryer, G. Rockefeller, K. Muthuraman, P.-H. Lin, and J. Jayaraj, 2012, "The late-time dynamics of the single-mode Rayleigh-Taylor instability," *Physics of Fluids* **24**, 074107 (2012).
15. Woodward, P. R., J. Jayaraj, P.-H. Lin, P.-C. Yew, M. Knox, J. Greensky, A. Nowatzki, and K. Stoffels, "Boosting the performance of computational fluid dynamics codes for interactive supercomputing," Proc. Intntl. Conf. on Comput. Sci., ICCS 2010, Amsterdam, Netherlands, May, 2010. Preprint available at www.lcse.umn.edu/ICCS2010.
16. Woodward, P. R., D. H. Porter, F. Herwig, M. Pignatari, J. Jayaraj, and P.-H. Lin. 2009. "The hydrodynamic environment of the s-process in the He-shell flash of AGB stars." in Proc. Nuclei in the Cosmos (NIC X), Mackinac Island, MI, 2008. arXiv:0901.1414 [astro-ph.SR]
17. Herwig, F., M. Pignatari, P. R. Woodward, D. H. Porter, G. Rockefeller, C. L. Freyer, M. Bennett, and R. Hirschi, 2011, "Convective-reactive proton-^{12}C combustion in Sakurai's object (V4334 Sagittarii) and implications for the evolution and yields from the first generations of stars," *Astrophysical Journal* **727**:89 (2011) , available at http://arxiv4.library.cornell.edu/pdf/1002.2241.

18. B. van Leer, Towards the Ultimate Conservative Difference Scheme. IV. A New Approach to Numerical Convection, *J. Comput. Phys.* **23**, 276-299 (1977).

19. G. M. Bassett and P. R. Woodward, Numerical Simulation of Nonlinear Kink Instabilities of Supersonic Shear Layers, *Journal of Fluid Mechanics* **284**, 323-340 (Feb. 10, 1995).

20. G. M. Bassett and P. R. Woodward, "Simulation of the Instability of Mach 2 and Mach 4 Gaseous Jets in 2 and 3 Dimensions," *Astrophysical Journal* **441**, (March 10, 1995).

21. "Bursts of Stellar Turbulence," article summarizing work by Porter, Woodward, and Knox with PPM+PPB on the Pittsburgh Supercomputing Center's Cray-XT3, in PSC's publication *Projects in Scientific Computing*, 2007 + cover.

22. Almgren, A. S., V. E. Beckner, J. B. Bell, M. S. Day, L. H. Howell, C. C. Joggerst, M. J. Lijewski, A. Nonaka, M. Singer, and M. Zingale, "CASTRO: a new compressible astrophysical solver. I. Hydrodynamics and self-gravity," *Astrophysical Journal*, **715**, 1221-1238 (2010).

23. Woodward, P. R., 2005. "The PPM Compressible Gas Dynamics Scheme," more complete, Web version of [12] available at www.lcse.umn.edu/ILES/PPM-for-ILES-2-19-05.pdf .

24. Chapman, P. R., and J. W. Jacobs, "Experiments on the three-dimensional incompressible Richtmyer-Meshkov instability," *Physics of Fluids* **18**, 074101 (2006).

25. Besnard, D. C., F. H. Harlow, R. M. Rauenzahn, and C. Zemach, "Turbulence Transport Equations for Variable-Density Turbulence and their Relationship to Two-Field Models," LA-12303-MS (1992).

26. Dimotakis, P. E., "Turbulent Mixing," *Ann. Rev. Fluid Mech.* **37**, 329-356 (2005).

27. Dimonte, G., and R. Tipton, "K-L Turbulence Model for the Self-Similar Growth of the Rayleigh-Taylor and Richtmyer-Meshkov Instabilities," *Phys. Fluids* **18**, 085101 (2006).

28. Banerjee, A., R. A. Gore, and M. J. Andrews, "Development and validation of a turbulent-mix model for variable-density and compressible flows," *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* 2010 Oct.; 82.046309.

29. Woodward. P. R., D. H. Porter, S. E. Anderson, T. Fuchs, and F. Herwig, Large-scale Simultaions of Turbulent Stellar Convection Flows and the Outlook for Petascale Computation, Proc. SciDAC 2006 conference, Denver, June, 2006; *Journal of Physics: Conference Series* **46** (2006), eds. W. M. Tang et al. Paper and PPT slides available from the conference Web site as well as at www.lcse.umn.edu/SciDAC2006.

30. Woodward, P. R., D. H. Porter, S. E. Anderson, and T. Fuchs, 2006. "Towards an Improved Numerical Treatment of Compressible Turbulence in Astrophysical Flows," in *Numerical Modeling of Space Plasma Flows*, ASP conference series, vol. 359, p. 97-110.

31. Woodward, P. R., and D. H. Porter, 2007. "A Model of Small-Scale Turbulence for use in the PPM Gas Dynamics Scheme," in *Applied Parallel Computing, State of the Art in Scientific Computing*, p. 1074-1083; Proc. PARA'06, Umea, Sweden, June, 2006.

32. Lin, P.-H., J. Jayaraj, and P. R. Woodward, "A Study of the Performance of Multifluid PPM Gas Dynamics on CPUs and GPUs," Proc. SAAHPC Conference, Knoxville, Tennessee, July, 2011. Preprint available at www.lcse.umn.edu/SAAHPC and presentation available at http://saahpc.ncsa.illinois.edu/presentations/lin.pdf.

33. Lin, P.-H., J. Jayaraj, P. Woodward, and P.-C. Yew, 2012, "A Study of Performance

Portability using Piecewise-Parabolic Method (PPM) Gas Dynamics Applications," Proc. International Conference on Computational Science, ICCS 2012, June, 2012, Omaha.

34. Woodward, P. R., J. Jayaraj, P.-H. Lin, and P.-C. Yew, 2008. "Moving Scientific Codes to the IBM Cell Processor and Other Multicore Microprocessor CPUs," *Computing in Science & Engineering*, Nov. 2008, 16-25; preprint available at www.lcse.umn.edu/CiSE.
35. Woodward, P. R., J. Jayaraj, P.-H. Lin, and W. Dai 2009, "First Experience of Compressible Gas Dynamics Simulation on the Los Alamos Roadrunner Machine," *Concurrency and Computation Practice and Experience*, **21,** 2160-2175 (2009), preprint available at www.lcse.umn.edu/RR-experience.
36. P.-H. Lin, J. Jayaraj, P. R. Woodward, and P.-C. Yew, 2011, "A Code Transformation Framework for Scientific Applications on Structured Grids," Technical Report 11-021, UMN Computer Science and Engineering Technical Report, Sept., 2011.
37. J. Jayaraj, P.-H. Lin, P. R. Woodward, and P.-C. Yew, 2012, "CFD Builder, A Library Builder for Computational Fluid Dynamics," submitted to IPDPS 2013.
38. Gittings, M., R. Weaver, M. Clover, T. Betlach, N. Byrne, R. Coker, E. Dendy, R. Hueckstaedt, K. New, W. R. Oakes, D. Ranta, and R. Stefan, "The RAGE radiation-hydrodynamic code," *Computational Science and Discovery* **1**, 015005 (2008), arXiv:0804.1394 [physics.comp-ph].
39. Luitjens, J., and M. Berzins, "Scalable parallel regridding algorithms for block-structured adaptive mesh refinement," *Concurrency and Computation: Practice and Experience* 2000; **00**:1-7.
40. Springel, V., "The cosmological simulation code GADGET-2," *Mon. Not. of the Royal Astron. Soc.* **364,** 1105 (2005).
41. Springel, V., S. D. M. White, A. Jenkins, C. S. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, J. A. Peacock, S. Cole, P. Thomas, H. Couchman, A. Evrard, J. Colberg, and F. Pearce, "Simulatons of the formation, evolution and clustering of galaxies and quasars," *Nature* **435**, 629 (2005).
42. Thomas, V. A., and R. J. Kares, "Drive Asymmetry and the Origin of Turbulence in an ICF Implosion," *Phys. Rev. Lett.* **109**, 075004 (2012). Preprint available at http://arXiv.org/abs/1210.3364 .