Automated Analyses of the Tri-Lab Verification Test Suite on Uniform and Adaptive Grids for Code Project A

Francis X. Timmes T-DO
Galen Gisler X-2
George M. Hrbek X-8

Los Alamos National Laboratory
Los Alamos, NM 87545 USA

October 30, 2005

LA-UR-05-6865
and
LA-CC-05-101

List of Figures

List of Tables

1. Executive Summary

What's New:

- Automation of the verification analyses. Generating numerical solutions, comparing the numerical and analytical solutions, performing a verification analysis, and plotting the key results does not need to be run by hand anymore. Execution of a single Perl script will complete these tasks for all 1D versions of the test problems.
- Execution of the verification analyses on adaptive grids. Most complex simulations use adaptive mesh refinement (AMR). It is important to know if the fundamental assertion of AMR, that solutions on adaptive grids are as accurate as the solutions on corresponding uniform grids, is fulfilled for the Tri-Lab problems.
- Archiving of analytic solution codes and input decks. Building on previous efforts often required knowing who to ask for what. In some cases, new analytic solution codes had to be written from scratch because the `owner' left the Lab leaving no obvious traces of their codes. All analytic solution codes and input decks are now in a centralized repository.

Results:

- Improvement in computational efficiency by using adaptive grids on the Tri-Lab test problems is usually about a factor of ten for 4-6 levels of mesh refinement, but this productivity gain isn't generally accompanied by a corresponding reduction in the error between the numerical and analytical solutions when the default AMR settings are used. For several Tri-Lab test problems the errors on adaptive grids appear to be larger than the errors on corresponding uniform grids.
- Performance on realistic applications might be robust if multi-material or multi-temperature physics in complex geometries mitigates difficulties exposed by simpler test problems on adaptive grids in basic geometries.

Recommended Directions:

- Develop and implement better AMR packages for Code Project A. Tremendous resources have gone into developing the physics modules, but not the grid technology on which that physics is computed. There doesn't appear to be a single team directly associated with Code Project A examining present and next-generation AMR technology - this represents a growth opportunity.
- Augment the present suite with new problems that exercise multi-material and/or multi-temperature capabilities. In parallel, develop rigorous calculation verification procedures for complex physics problems that don't admit an exact solution. Finally, automated verification analysis should be completed for two and three dimensional versions of all the existing Tri-Lab test problems to assess how well AMR hydrocodes retain fidelity to the underlying physics when motions and gradients are not grid-aligned.

2. Automation

Generating numerical solutions, comparing the numerical and analytical solutions, performing a verification analysis, and plotting the key results does not need to be run by hand anymore. Execution of an automation tool will complete these tasks for all 1D versions of the test problems. Extending the automation to 2D and 3D versions of the Tri-Lab test suite should be a priority of next year's program.

The automation tool requires a control deck containing command lines with specific information about the particular analyses (i.e., test problem definition, code designation, number of simulation runs, and the type of convergence analysis to be performed) to be performed, and a list of directories in which to find the results of the numerical simulations (Hrbek et al. 2005).

RAGE is run on a set of input decks, one for for each grid resolution. This creates a set of binary dump files which contain the solution data on the native grid. John Grove's AMHCTOOLS (2005) extracts the solution data and outputs it to an ASCII text file. While exacting the native data isn't vital on uniform meshes, it is critical for verification analyses on adaptive grids because RAGE interpolates the native data onto a uniform mesh when returning data in either HDF 4 or SDSS formats. The analytic solution is then run live (as opposed to pre-computed) on the exact same grid as used in the RAGE calculations. This is useful when convergence analyses are done on adaptive grids. Various error norms are then computed for each grid by comparing the analytic and numerical solutions. Finally the convergence coefficients are calculated by comparing error norms on all grids.

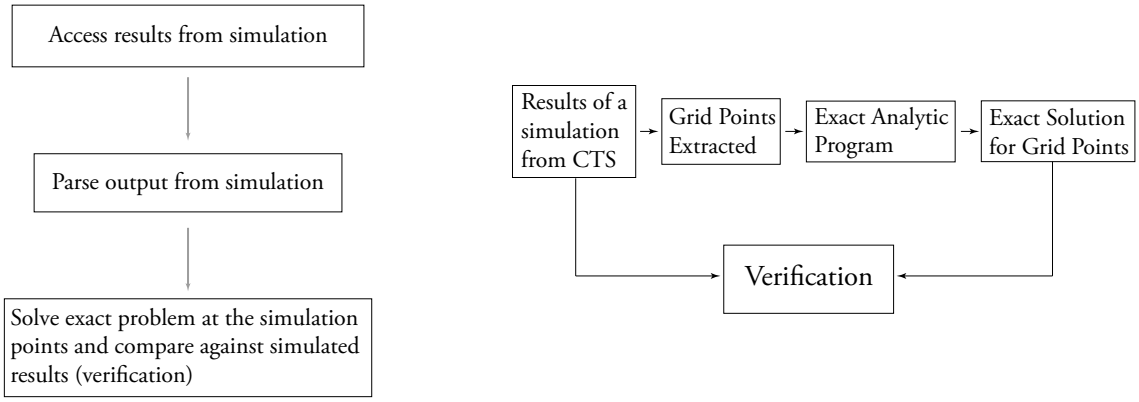

Figure 1. - Present (left) and potential future (right) flow of the automated verification analysis for Code Project A.

Figure 1 shows the present automation work-flow at a higher level. Each simulation is identified and parsed to extract the structured mesh for that time snapshot (Step 1). This mesh information is then used in the computer program to generate the equivalent analytic solution (Step 2) and perform the verification analysis and generate a report of the rates of convergence (Step 3). Note that this approach also allows for a modular implementation. This procedure varies slightly from that used for Code Project B, which treats Steps 2 and 3 as entirely separate entities. In the next phase of automation outlined in Figure 2, only the exact analytic solutions will be generated inside the exact analytic programs and the verification analyses will be handled in a Verification module.

An initial efforts has also been made to identify potential impediments to having the same verification analysis tools operate for both Code projects A and B (Hrbek et al. 2005). This approach is essential to abstracting automation efforts and allowing a common, modular script to be written. Issues such as the consistency in file directory naming conventions and common file structures were considered. Each of the ASC Projects has its own output file structures. When these are added to the binary link files for the Legacy codes, being able to easily access results from each of the computer simulation codes becomes daunting. The previously mentioned AMHCTOOLS Project of John Grove may be able to provide a tool for interpreting the various formats. Alternatively, an API option suggested by Mabel Grey-Vigil (X-3) and Chuck Wingate (X-3) for direct query of dump files could be another option.

3. Tri-Lab Verification Test Suite

3.1 The Su & Olson Problem

The Su & Olson problem is a one-dimensional, half-space, non-equilibrium Marshak wave problem. The radiative transfer model is a one-group diffusion approximation with a finite radiation source boundary condition, where the radiative and material fields are out of equilibrium. As the energy density of the radiation field increases, energy is transfered to the material. Su & Olson (1996) found a solution, to quadrature, for the distribution of radiative energy and material temperature as a function of spacetime. This problem is useful for verifying time-dependent radiation diffusion codes.

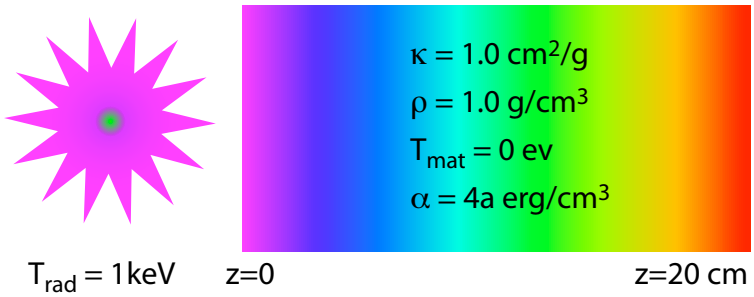

Figure 3. - Setup and parameters for the Su & Olson problem.

3.1.1 Analytical Solution

Given the one-dimensional domain ($0 \leq z \leq \infty$), the radiation temperature $T_{\mathrm{rad,boundary}}(z=0,t)$ at the boundary, the total opacity κ, and the desired solution point in time t, one begins by forming the dimensionless variables

$$x = \sqrt{3}\kappa z \qquad \tau = \left(\frac{4ac\kappa}{\alpha}\right)t \qquad \epsilon = \frac{4a}{\alpha} \ , \tag{1}$$

where a is the radiation constant, c is the speed of light, and α is determined from the specific heat of the material $c_v = \alpha T^3_{\mathrm{mat}}$. For the Tri-Lab Verification problem it is assumed the material equation of state obeys $E_{\mathrm{mat}} = aT^4_{\mathrm{mat}}$, so that $\alpha = 4a$. One now computes the dimensionless solution for the radiation energy density

$$u(x,\tau,\epsilon) = 1 - \frac{2\sqrt{3}}{\pi}\int_0^1 \left[\frac{\sin\left[x\gamma_1(\eta,\epsilon)+\theta_1(\eta)\right]}{\eta\sqrt{3+4\gamma_1^2(\eta,\epsilon)}}\right] \mathrm{d}\eta$$

$$- \frac{\sqrt{3}e^{-\tau}}{\pi}\int_0^1 \left[\frac{\sin\left[x\gamma_2(\eta,\epsilon)+\theta_2(\eta)\right]}{\eta(1+\epsilon\eta)\sqrt{3+4\gamma_2^2(\eta,\epsilon)}}\right] e^{\frac{-\tau}{\epsilon\eta}}\,\mathrm{d}\eta \tag{2}$$

followed by the dimensionless solution for the material energy density

$$v(x,\tau,\epsilon) = u(x,\tau,\epsilon) - \frac{2\sqrt{3}}{\pi}\int_0^1 \left[\frac{\sin\left[x\gamma_3(\eta,\epsilon)+\theta_3(\eta)\right]}{\sqrt{4-\eta^2+4\epsilon\eta^2(1-\eta^2)}}\right] e^{-\tau(1-\eta^2)}\,\mathrm{d}\eta$$

$$+\frac{\sqrt{3}e^{-\tau}}{\pi}\int_0^1 \left[\frac{\sin\left[x\gamma_2(\eta,\epsilon)+\theta_2(\eta)\right]}{\eta\sqrt{3+4\gamma_2^2(\eta,\epsilon)}}\right] e^{\frac{-\tau}{\epsilon\eta}}\,\mathrm{d}\eta\ , \tag{3}$$

where

$$\gamma_1(\eta,\epsilon) = \eta\sqrt{\epsilon+\frac{1}{(1-\eta^2)}} \qquad \gamma_2(\eta,\epsilon) = \sqrt{(1-\eta)\left(\epsilon+\frac{1}{\eta}\right)} \qquad \gamma_3(\eta,\epsilon) = \sqrt{(1-\eta^2)\left(\epsilon+\frac{1}{\eta^2}\right)} \tag{4}$$

and

$$\theta_n(\eta,\epsilon) = \arccos\sqrt{\frac{3}{3+4\gamma_n^2(\eta,\epsilon)}}\ , \qquad n=1,2,3. \tag{5}$$

The physical radiation energy density and material temperatures follow as

$$E_{\rm rad}(z,t) = u(x,\tau,\epsilon)\cdot E_{\rm rad}(z=0,t) = u(x,\tau,\epsilon)\cdot aT^4_{\rm rad,boundary}$$

$$aT^4_{\rm mat}(z,t) = v(x,\tau,\epsilon)\cdot E_{\rm rad}(z=0,t) = v(x,\tau,\epsilon)\cdot aT^4_{\rm rad,boundary} \tag{6}$$

Figure 4 shows the γ_i of equation (4), θ_i of equation (5), and some representations of the four integrands in equations. (2) and (3). While all the integrands are oscillatory for the parameter choices used in Figure 4 - they are not always oscillatory! - the first integral in equation (3) for the material energy density (green curves) is particularly challenging to evaluate.

A code to compute this mildly complicated analytic solution wasn't readily available when this project started in October 2004. Gordon Olson apparently had the only functional code and was no longer at LANL. So, one of us (Timmes) wrote a new fortran code to calculate the analytic solution (see Appendix B). This new code duplicates all four tables in the Su & Olson (1996) paper.

On March 28, 2005, Timmes received an email message from Gordon Olson containing his fortran code for the solution to the Su & Olson problem. A comparison of the two codes several similarities: both codes use a Romberg integration and both codes address cases when the integrands are not oscillatory. A comparison of the solutions generated by each code should be addressed in the future efforts.

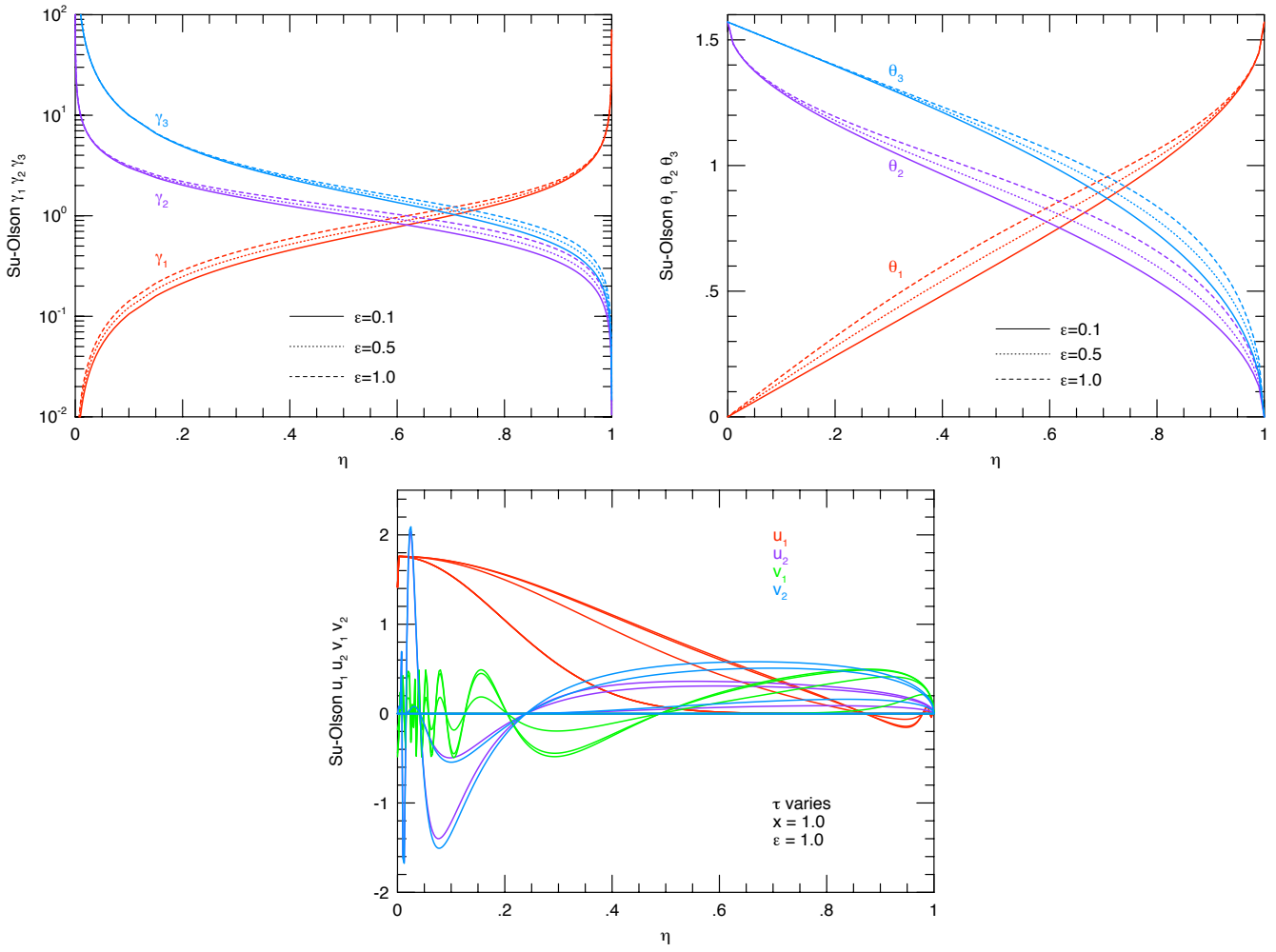

Figure 4. - Key components of the Su & Olson solution. The top left panel shows the γ_i of equation (4), the top right panel shows the θ_i of equation (5), and the bottom panel shows the four integrands of equations. (2) and (3) for a few select choices of their input parameters.

3.1.2 Verification Analysis on 1D Uniform and Adaptive Grids

RAGE 20041126.030, 20041126.032, 20050331.02, 20050331.31, 20050331.41, and 20050818.0000 was used to generate numerical solutions to the Su & Olson problem on uniform and adaptive grids. This problem was mainly run on the Linux cluster Lambda. The base RAGE input deck for this problem is the same one used in Kamm & Kirkpatrick (2004). The input deck was simplified and annotated (see Appendix A) to make it more amenable to automation, adaptive meshes, and future studies.

Solutions were generated on uniform grids with 50, 100, 200, 400, 800, 1600 and 3200 cells at 1×10^{-11} s, 1×10^{-10} s, and 1×10^{-9} s. Solutions were also computed on adaptive meshes with an effective resolution (if maximally refined everywhere) of 100, 200, 400, 800, 1600, and 3200 cells at the same time points. These six effective resolution grids correspond to six levels of adaptive mesh refinement. The 100 cell formed the Level 1 adaptive grid, and the

`sizemat` parameter was used to increase the potential maximal refinement level. Other mesh refinement variables such as `numfine`, `numlev` `mxcells` and `sizbnd` were left at their default values (not used).

John Grove's excellent AMHCTOOLS (2005) was used to obtain the solution data on the native grid. While exacting the native data isn't vital on uniform meshes, it is critical for verification analyses on adaptive grids because RAGE interpolates the native data onto a uniform mesh when returning data in either HDF 4 or SDSS formats.

Figure 5 compares the solutions on uniform and adaptive meshes. Solutions are shown for 0.001, 0.01 and 0.1 shakes. Initially, the radiation streams into the slab and the material temperature lags behind the radiation temperature. As the radiation energy density builds up, the material temperature catches up, and by t = 0.1 sh, the radiation and material temperatures are essentially identical.

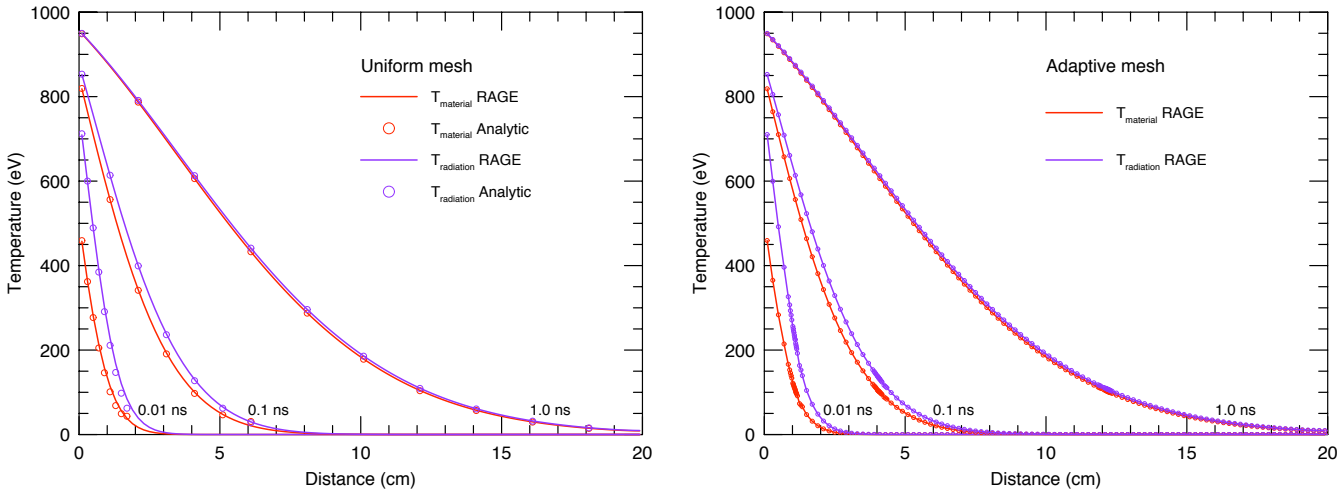

Figure 5. - The plot on the left compares the numerical (solid curves) and analytical (open circles) solutions for the radiation temperature (purple) and material temperature (red) on a uniform grid with 200 cells. The plot on the right shows the numerical solution (lines) on an adaptive mesh with 4 levels of refinement, and circles mark the location of the adaptive grid points.

The right hand side plot in Figure 5 shows there is a fair amount of adaptive mesh refinement occurring at early times (0.01 ns), but by 0.1 ns there is only a small clumping of mesh refinement around the 4 cm point. At 1.0 ns the small clump of refinement has moved out to 12 cm. This small clump of refinement is due to generation of an unphysical oscillation in the energy densities.

Following Chapelle (1999), we'll define the patch-extent weighted absolute L_1 norm, the relative L_1 norm, the patch-extent weighted L_2 norm, and the relative L_2 norm as

$$L_{1,\mathrm{abs}} = \frac{\sum(f_i^{\mathrm{exact}} - f_i^{\mathrm{rage}})\Delta x_i}{\sum \Delta x_i} \qquad L_{1,\mathrm{rel}} = \frac{\sum(f_i^{\mathrm{exact}} - f_i^{\mathrm{rage}})}{\sum f_i^{\mathrm{exact}}}$$

$$L_{2,\mathrm{abs}} = \left[\frac{\sum(f_i^{\mathrm{exact}} - f_i^{\mathrm{rage}})^2\Delta x_i}{\sum \Delta x_i}\right]^{1/2} \qquad L_{2,\mathrm{rel}} = \left[\frac{\sum(f_i^{\mathrm{exact}} - f_i^{\mathrm{rage}})^2}{\sum(f_i^{\mathrm{exact}})^2}\right]^{1/2} \tag{7}$$

Figure 6 shows the absolute value of the point-wise $L_{1,rel}$ error for the radiation and material temperatures on uniform and adaptive grids at t=0.1 sh. The cusps in the curves are due to sign changes in the error. The large, persistent errors at the x=20 cm right boundary are probably caused by the Marshak wave hitting the right boundary at 0.1 sh. But there may be two additional reasons why the errors get large for small values of the temperature or energy densities. First, there are limits to how accurate one can obtain the analytical solution near the leading edge of the Marshak wave where the deviations from the ambient background are small. Second, in RAGE, one runs this problem with a non-zero ambient material temperature, while the analytic calculation assumes an identically zero initial value. These issues should be investigated in a follow-up report.

Away from the right boundary, the convergence behavior on uniform grids saturates around 800 grid points. Running the Su & Olson problem with 3200 uniform grid points doesn't appear to improve the quality of the solution.

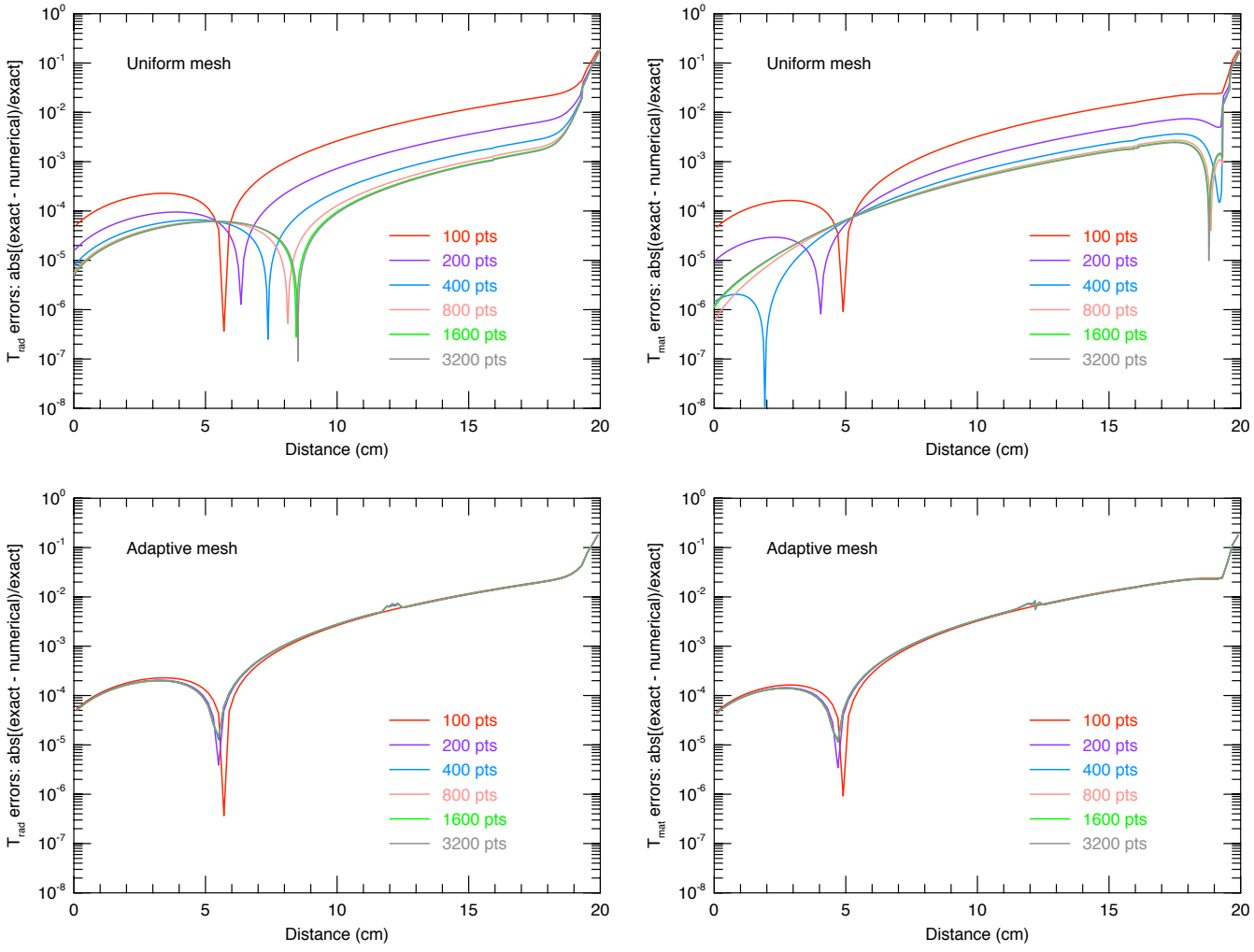

Figure 6. - Absolute value of the point-wise $L_{1,rel}$ norm for the radiation and material temperatures on uniform and adaptive grids at t=0.1 sh.

Figure 6 suggests the default values for the adaptive mesh refinement criteria don't produce significant amounts of refinement. As a result, the errors are basically independent of the refinement level. Decreasing the parameter `tevcut` from its default value of 100 eV to 10eV and 1eV, as suggested by Mike Gittings, didn't significantly increase the number of grid points. Adaptive mesh refinement, used mainly with the default parameter settings, seems to produce little improvement in reducing the errors in the Su & Olson problem. Note the deviations around 12.0 cm from a smooth solution in the adaptive mesh solutions. This feature is absent in the uniform mesh case, and appears to be seeded very early ($t \leq 10^{-11}$ s) in the evolution. The nature and removal of this feature should be investigated in a follow-up report.

Figure 7 shows various global error norms of radiation and material temperatures on uniform grids at t=0.1 sh. The global convergence rate at the finest resolutions is dominated by the large, stubborn residuals at the right boundary (see Figure 6).

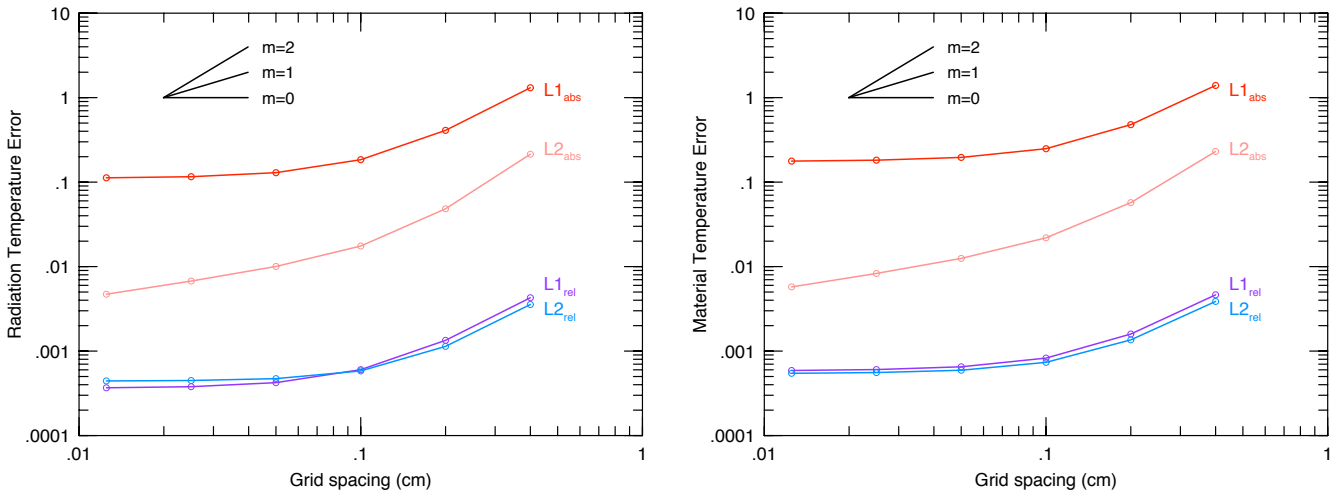

Figure 7. - Global error norms for the radiation and material temperature as a function of uniform grid spacing. The black lines show slopes of 0, 1, and 2. After an initial convergence rate of $q \sim 1.5$, all the norms display a slow convergence rate of $q \sim 0.1$.

Assuming the global error between the numerical and analytical solutions depends only on the cell size, Error = A $(\Delta x)^q$, the rate of convergence q was computed for all uniform grid pairs as

$$q = \log\left[\frac{E_{\text{fine grid}}}{E_{\text{course grid}}}\right] \Big/ \log\left[\frac{\Delta x_{\text{fine grid}}}{\Delta x_{\text{coarse grid}}}\right] \quad . \tag{8}$$

Table 1 shows the convergence rate on uniform grids for the $L_{1,\text{abs}}$ norm decreases from $q \sim 1.5$ for the largest cell sizes to $q \sim 0.1$ at the finest resolutions. Similar convergence rates hold for the other error norms. A slowing of the convergence rate at smaller cell sizes was also found by Kamm & Kirkpatrick (2004). A few runs with a smaller maximum allowed time-steps (RAGE parameter `de_tevpct`=0.003, 0.001, and 0.0005) did not significantly alter the convergence rates.

Table 1
Global Convergence Coefficients for the Su & Olson Problem[1]

		T_{rad}			T_{mat}	
# of cells	$L_{1,abs}$	q	A	$L_{1,abs}$	q	A
50	1.310					
100	0.401	1.674	916.2	0.479	1.541	579.4
200	0.184	1.154	83.20	0.249	0.946	37.41
400	0.129	0.510	2.750	0.196	0.342	1.525
800	0.115	0.159	0.336	0.182	0.109	0.375
1600	0.112	0.043	0.155	0.177	0.036	0.232

[1] For the $L_{1,abs}$ norm on uniform grids with RAGE 20050818.0000

The ratio of the CPU resources consumed by the uniform and adaptive grid calculations are shown in Figure 8 as a function of refinement level. While the improvement in computational efficiency is larger than a factor of 10 for 6 levels of mesh refinement, Figure 6 demonstrates this isn't necessarily accompanied by a corresponding reduction in the errors.

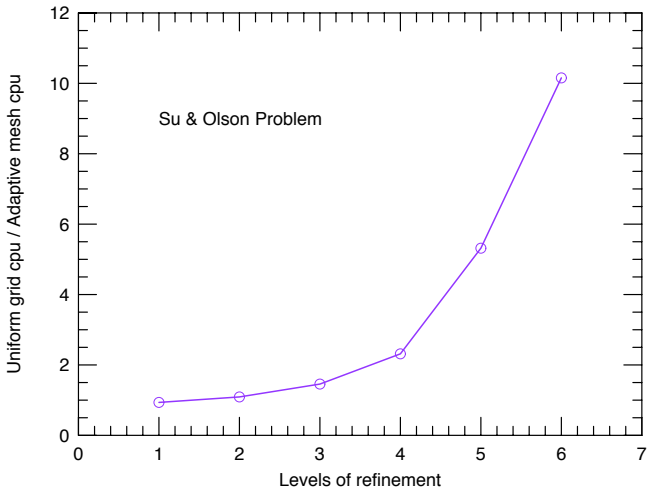

Figure 8. - CPU usage on uniform and adaptive grids for the one-dimensional Su & Olson problem.

3.2 The Coggeshall #8 Problem

Coggeshall (1991) developed a collection of analytic, similarity solutions for the one-dimensional hydrodynamics equations. Analytical solution #8 represents a fluid flow with heat conduction:

$$\rho(r,t) = \rho_0 \; r^{(k-1)/(\beta-\alpha+4)} \; t^{-(k+1)-(k-1)/(\beta-\alpha+4)}$$

$$T(r,t) = T_0 \; r^{(1-k)/(\beta-\alpha+4)} \; t^{(1-\gamma)(k+1)+(k-1)/(\beta-\alpha+4)}$$

$$v(r,t) = r/t \;\; . \tag{9}$$

Note the parameter labeled `$\rho_0$' in equation (10) doesn't have units of a mass density, nor does `T_0' have units of a temperature. Instead, each has units such that the when multiplied by the spacetime components of equation (9), they yield quantities with the expected units. Both RAGE and the analytic solution codes provided with this problem (Kamm & Kirkpatrick 2004) use α=-1, β=2, $k = 2$ for spherical symmetry (see Figure 9), and a perfect gas equation of state with γ=5/3. In this case, ρ_0 = 3 g cm$^{-3+1/7}$ s$^{-3-1/7}$, and T_0 = 2000 eV cm$^{-1/7}$ s$^{-2+1/7}$.

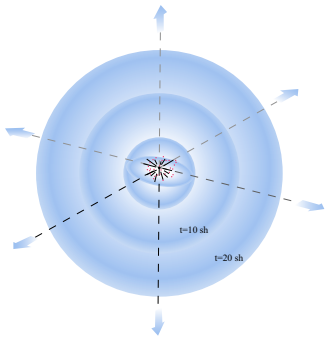

Figure 9. - Setup for the spherically symmetric Coggeshall #8 problem. The analytic solution at t=10 sh is used as the initial condition in RAGE. The numerical solution is then evolved to t=20 sh.

3.2.1 Analytical Solution

The similarity point solution of must be modified for an use in, or comparison with, Eulerian hydrocodes. Define

$$\texttt{seventh} = \frac{k-1}{\beta-\alpha+4} \qquad \texttt{rexp} = \texttt{seventh} \qquad \texttt{texp} = (k+1) + \texttt{seventh} \;\; . \tag{10}$$

For a cell with left `xl` and right `xr` boundaries, the cell volume, mass, and momentum are

$$\texttt{dv} = \frac{4\pi}{k+1}\left(\texttt{xr}^{k+1} - \texttt{xl}^{k+1}\right) \qquad \texttt{dm} = 4\pi\rho_0\left[\frac{\texttt{xr}^{k+1+\texttt{rexp}} - \texttt{xl}^{k+1+\texttt{rexp}}}{\texttt{texp1}\; t^{\texttt{texp}}}\right]$$

$$\texttt{dmv} = 4\pi\rho_0\left[\frac{\texttt{xr}^{k+2+\texttt{rexp}} - \texttt{xl}^{k+2+\texttt{rexp}}}{(\texttt{texp}+1)\; t^{\texttt{texp}+1}}\right] \;\; . \tag{11}$$

The mass density ρ, velocity v, energy density E_v, specific energy density e, temperature T, and pressure P are then

$$\rho = \frac{\texttt{dm}}{\texttt{dv}} \qquad v = \frac{\texttt{dmv}}{\texttt{dm}} \qquad E_v = \frac{\rho_0 T_0 c_v}{t^5}$$

$$e = \frac{E_v}{\rho} \qquad T = \frac{e}{c_v} \qquad P = (\gamma - 1)e \ . \tag{12}$$

Initial comparisons between the numerical and analytical results led to the finding that the supplied analytic code had an error in the time exponent for the momentum `dmv`. The analytic code used $(\texttt{texp} - 1)$ instead of $(\texttt{texp} + 1)$. A corrected code is listed in Appendix B.

3.2.2 Verification Analysis on 1D Uniform and Adaptive Grids

RAGE 20041126.030, 20041126.032, 20050331.02, 20050331.31, 20050331.41, and 20050818.0000 on the Linux cluster Lambda was used to generate numerical solutions to the Coggeshall #8 problem. The starting RAGE input deck for this problem was the same one used by Kamm & Kirkpatrick (2004). The input deck was simplified and annotated (see Appendix A) to make it more amenable to automation, adaptive meshes, and future studies.

Initial conditions at 10 sh were generated by the analytic solution on uniform grids with 100, 200, 400, 800, 1600, 3200, and 6400 cells. Appendix A details some of the subtleties associated with the initial conditions. Solutions were also computed on adaptive meshes with an effective resolution (if maximally refined everywhere) of 100, 200, 400, 800, 1600, 3200 and 6400 cells. These six effective resolution grids correspond to six levels of adaptive mesh refinement. The 100 cell formed the Level 1 adaptive grid, and the `sizemat` parameter was used to increase the potential maximal refinement level. Other mesh refinement variables such as `numfine`, `numlev` `mxcells` and `sizbnd` were left at their default values (not used).Grove's AMHCTOOLS (2005) was used to extract the solution data on the native grid from the binary dump files.

Figure 10 shows solutions at 20 sh on uniform and adaptive meshes. The lower panel suggests that allowing 4 levels of refinement adds some additional grid at the left and right boundaries, but otherwise has the same grid as the base 100 cell simulation. This might be expected given the usage of default regridding parameters and the similarity nature of the solution; once the spatial grids are refined at the initial conditions, no new grid is added. Note that the 1 level and 4 level simulations attain different solutions at the left boundary.

Figure 11 shows the point-wise $L_{1,rel}$ error for the density, pressure, and velocity on uniform and adaptive grids. In general, the errors get smaller with increasing uniform grid resolution. However, there are large, persistent errors at the boundaries. Errors at the left boundary may be due to the coordinate singularity at the origin. It is unclear as of this writing what is causing the errors at the right boundary. On adaptive grids, there is little reduction in the error as the spatial resolution is increased when the default AMR parameters are used. Once gradients are refined at the initial conditions, iit seems that no new grid is added because the solution is self-similar.

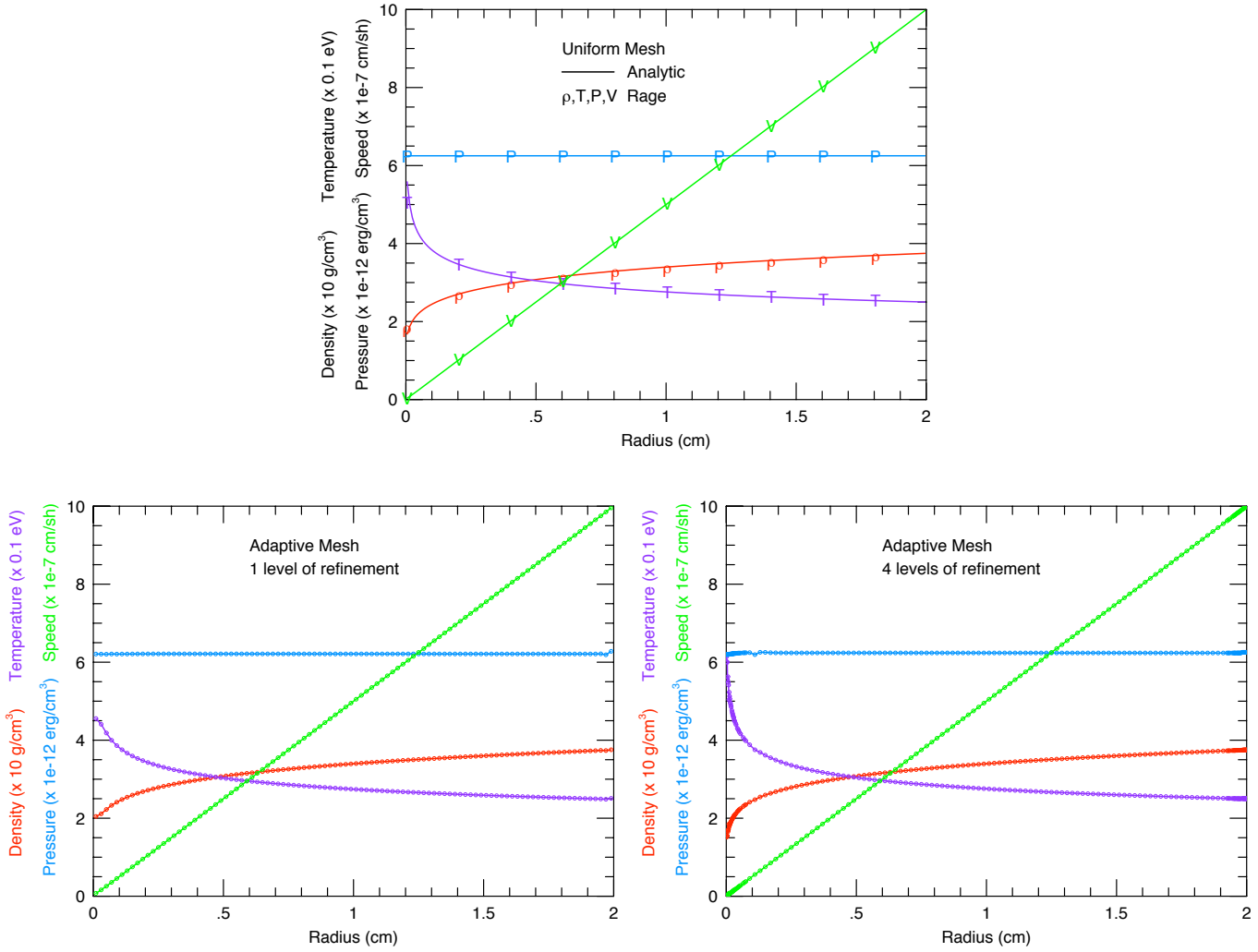

Figure 10. - The top plot shows the analytical (solid curves) and numerical (symbols) solutions at t=20 sh for the mass density (red), velocity (green), pressure (blue), and temperature (purple). The bottom plots show the solution with 1 level of refinement (left) and 4 levels of refinement (right), where circles mark the location of the adaptive grid points.

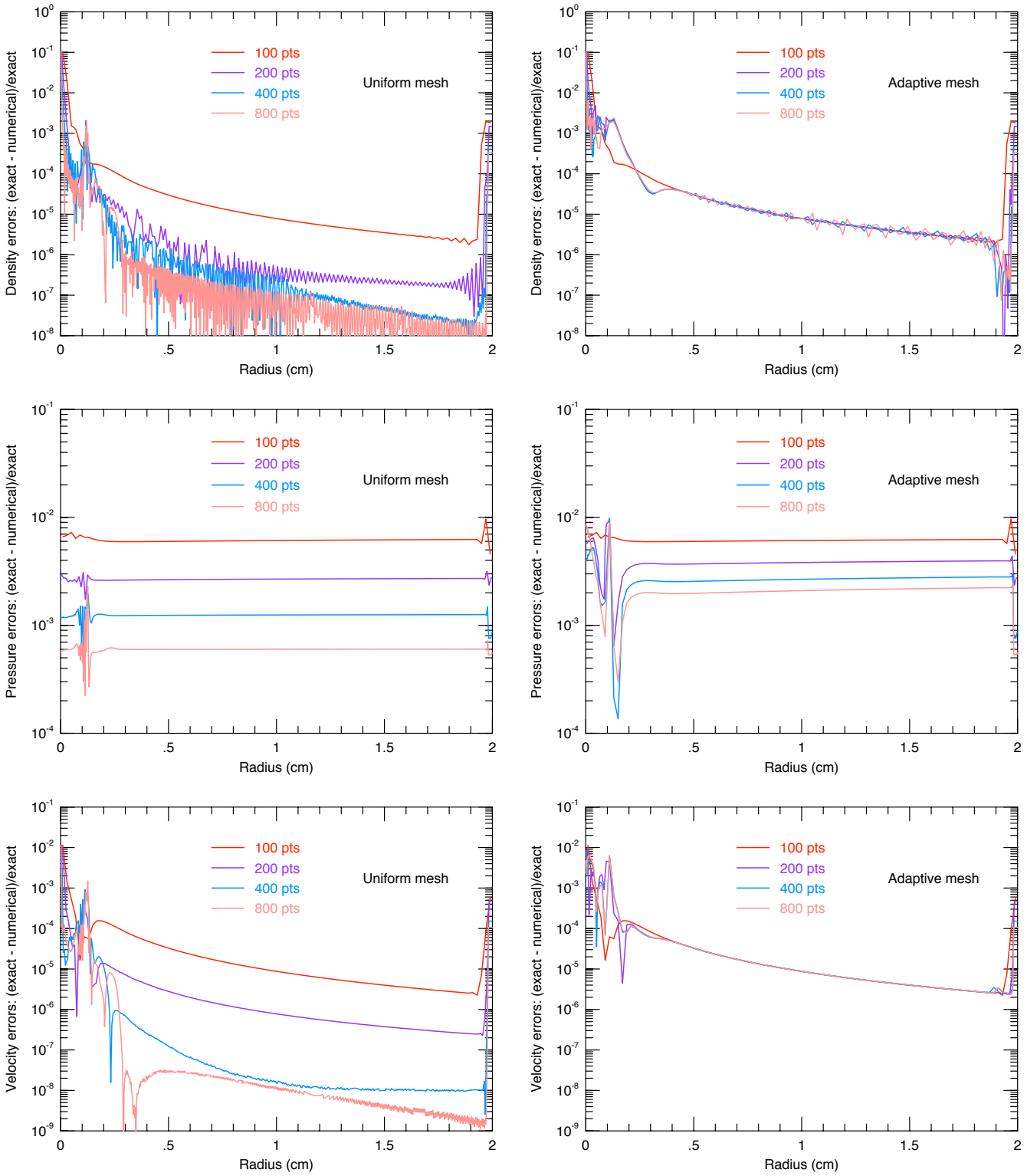

Figure 11. - Absolute value of the point-wise $L_{1,\mathrm{rel}}$ error for the density, pressure, and velocity on uniform and adaptive grids at 20 sh.

Figure 12 shows various global error norms on uniform grids computed from equation (7). The rate of convergence on coarse grids is usually quadratic ($q \sim 2$), but degrades at finer grids to sub-linear ($q \leq 0.5$) because of the large, persistent errors at the boundaries. Note the rate of convergence in the velocity becomes worse as the spatial resolution Table 2 details the global convergence rate properties of the $L_{1,\mathrm{abs}}$ norm, assuming the error ansatz of equation (8).

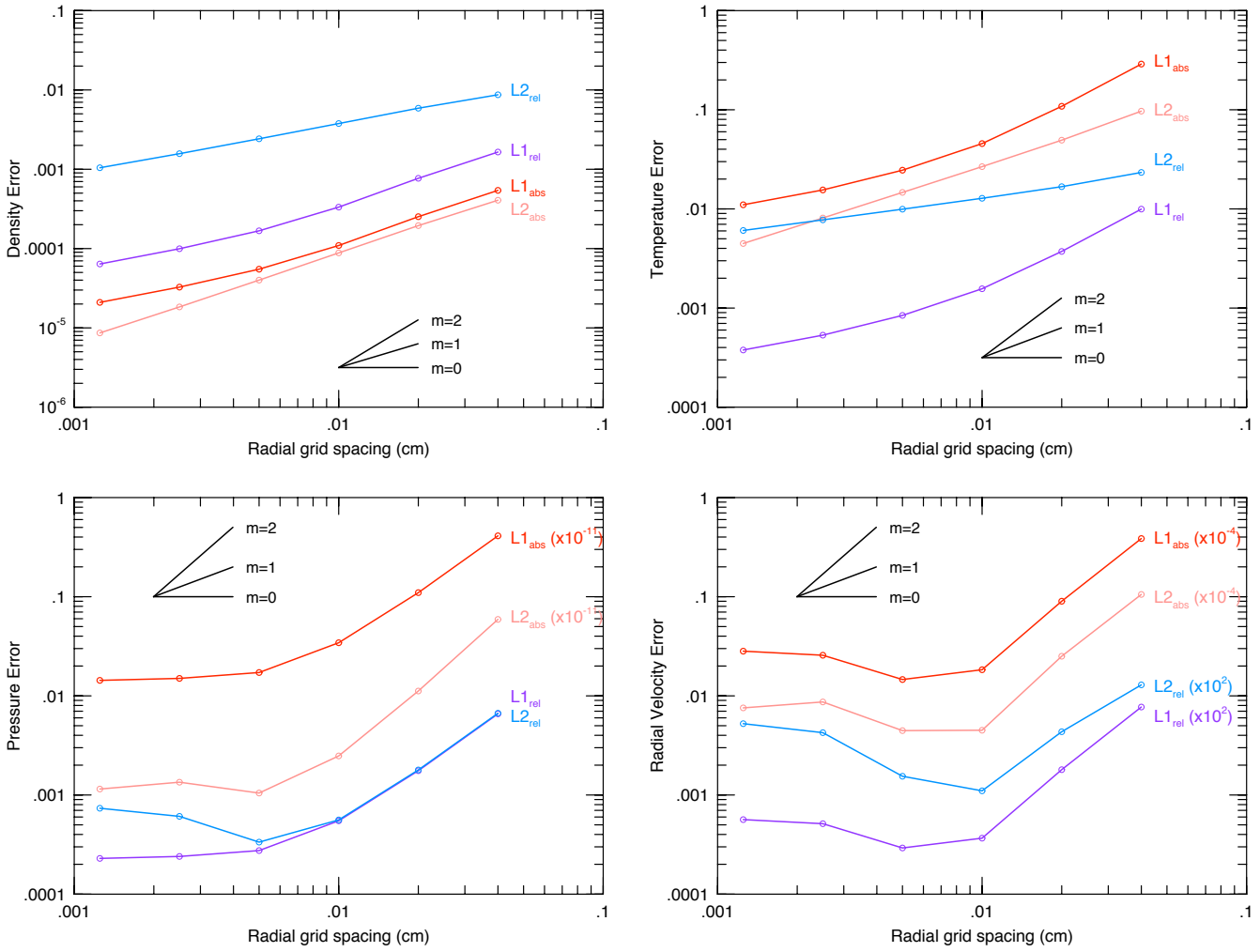

Figure 12. - Global error norms on uniform grids for the mass density, material temperature, pressure, and velocity at t=20 sh. The black lines show slopes of 0, 1, and 2.

Table 2
Global convergence coefficients for the Coggeshall #8 problem[1]

	ρ		v		P		T	
# of cells	q	A	q	A	q	A	q	A
100	1.10	0.040	2.10	1.4E+07	1.904	7.1E+13	1.415	70.33
200	1.21	0.065	2.30	3.5E+07	1.678	2.5E+13	1.249	34.15
400	0.992	0.021	0.327	1041.7	0.998	6.8E+11	0.892	5.143
800	0.751	0.005	-0.813	1.120	0.197	5.6E+09	0.661	1.291
1600	0.639	0.003	-0.137	103.0	0.065	2.3E+09	0.497	4.311

[1] For the $L_{1,abs}$ norm on uniform grids with RAGE 20050818.0000

The ratio of the CPU resources consumed by the uniform and adaptive grid calculations are shown in Figure 13 as a function of refinement level. While the improvement in computational efficiency is larger than a factor of 10 for 7 levels of mesh refinement, Figure 11 suggests the reduction in CPU resources by using an adaptive grid isn't necessarily attended by a similar reduction in L_1 error norms.

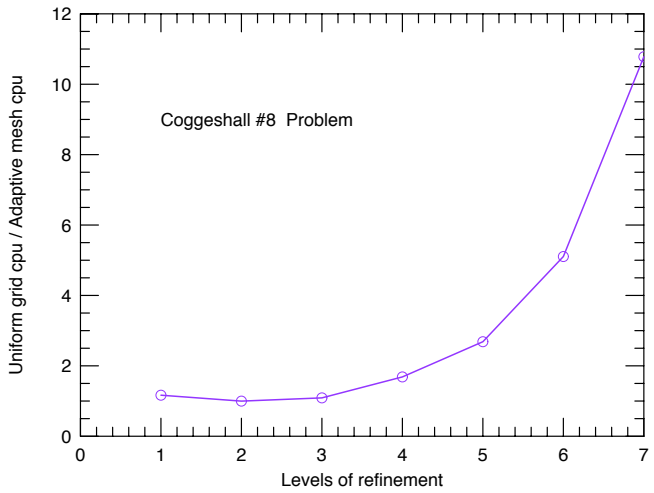

Figure 13. - CPU usage on uniform and adaptive grids for the one-dimensional Coggeshall #8 problem.

3.3 The Mader Problem

Ignite a slab of HE on one side and let a one-dimensional detonation propagate to the other side. A simple rarefaction or Taylor wave follows the detonation (Kirkpatrick et al. 2004, Kamm & Kirkpatrick 2004). Expansion of material in the rarefaction depends on the boundary condition where the detonation is initiated, which is usually modeled as a freely moving surface or a moving piston. For the Mader problem, a stationary piston is used. In this case, the head of the rarefaction remains at the detonation front since the flow is sonic there, and the tail of the rarefaction is halfway between the front and the piston.

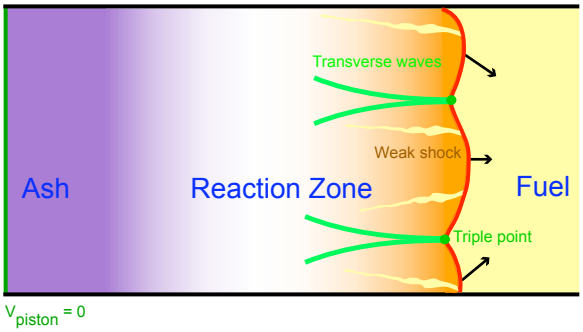

Figure 14. - Setup for the one-dimensional Mader problem. A 5 cm slab of HE is at rest with a density of 1.875 g/cm^3, temperature of 0.025 eV, and an equation of state characterized by $\gamma = 3$. The Chapman-Jouget detonation speed is set to 0.8 cm/μs, thus taking 6.25 μs for the detonation to reach the x=0 boundary. The rich structure of a multi-dimensional detonation is absent in one-dimensional simulations, and a simple rarefaction wave follows the detonation front.

3.3.1 Analytical Solution

The analytic result for the rarefaction wave behind the detonation is given very concisely on page 24 of Fickett & Davis (1979). For comparison with an Eulerian hydrocode, the point solution must be averaged. Given the time t, position in the fixed lab frame $x_{\rm lab}$, width of the cell Δx, the piston speed $v_{\rm piston}$ Chapman-Jouget pressure behind the detonation from P_j, Chapman-Jouget detonation speed D_j, and the equation of state characterized by γ, one forms

$$\rho_0 = \frac{P_j}{D_j^2}\,(\gamma+1) \qquad \rho_j = \rho_0\left(\frac{\gamma+1}{\gamma}\right)$$
$$c_j = D_j\left(\frac{\gamma}{\gamma+1}\right) \qquad v_j = \frac{D_j}{\gamma+1} \qquad x_{\rm det} = D_j t - x_{\rm lab}\ , \tag{13}$$

where the last expression for $x_{\rm det}$ gives the spatial coordinate relative to the (moving) detonation front. Auxiliary expressions of use are

$$a = \frac{1}{2c_j t} \qquad b = \frac{2-(\gamma-1)u_j/c_j}{(\gamma+1)} \qquad d = \frac{2}{\gamma}$$

$$x_l = x_{\rm det} - \frac{1}{2}\Delta x \qquad x_r = x_{\rm det} + \frac{1}{2}\Delta x$$

$$x_L = x_{\rm det} - \Delta x \qquad x_R = x_{\rm det} + \Delta x \ . \tag{14}$$

The transition between the rarefaction fan and the constant state is determined from the quantities

$$v_m = \left(v_j - \frac{2c_j}{\gamma - 1}\right)\left(\frac{\gamma - 1}{\gamma + 1}\right) \qquad x_p = \frac{1}{2}(\gamma + 1)t(v_{\rm piston} - v_m) \tag{15}$$

In the rarefaction fan, $x_{\rm det} < x_p$, the solution for the material speed, pressure, sound speed and mass density is

$$v(x_{\rm lab}, t) = \frac{2}{(\gamma + 1)}\frac{x_{\rm det}}{t} + \left[v_j - \frac{2c_j}{(\gamma - 1)}\right]\left(\frac{\gamma - 1}{\gamma + 1}\right)$$

$$P(x_{\rm lab}, t) = P_j\left[\frac{(ax_R + b)^{b+1} - (ax_L + b)^{b+1}}{a(b + 1)\Delta x}\right]$$

$$c(x_{\rm lab}, t) = c_j(ax_R + b)$$

$$\rho(x_{\rm lab}, t) = \rho_j\left[\frac{(ax_R + b)^{d+1} - (ax_L + b)^{d+1}}{a(d + 1)\Delta x}\right] \ . \tag{16}$$

Outside the rarefaction fan, $x_{\rm det} > x_p$, the solution for the material speed, pressure, sound speed and mass density is a constant state given by

$$v(x_{\rm lab}, t) = v_{\rm piston}$$

$$P(x_{\rm lab}, t) = P_j\left[1 + (\gamma - 1)\frac{v(x_{\rm lab}, t) - v_j}{2c_j}\right]^{d/(\gamma - 1)}$$

$$c(x_{\rm lab}, t) = c_j\left[1 + (\gamma - 1)\frac{v(x_{\rm lab}, t) - v_j}{2c_j}\right]$$

$$\rho(x_{\rm lab}, t) = \rho_j\left(\frac{P(x_{\rm lab}, t)}{P_j}\right)^{1/\gamma} \ . \tag{17}$$

A fortran code to calculate the analytic solution is given in appendix B, and based on the codes used by Kamm & Kirkpatrick (2004).

3.3.2 Verification Analysis on 1D Uniform and Adaptive Grids

NOBEL 20041126.002 and 20050331.021 was run on the Linux cluster Lambda to generate numerical solutions to the Mader problem using the initial conditions and parameters described in Figure 14. These initial conditions were evolved to 6.25×10^{-6} s and Grove's AMHCTOOLS (2005) was used to extract the solution data on the native grid from the binary dump files.

The NOBEL input deck was the same one used by Kamm & Kirkpatrick (2004), but modified to make it amenable to automation, and adaptive meshes (see Appendix A). The Mader problem's special `.teos` equation of state file had to be re-generated due to internal changes in the NOBEL code since Kamm & Kirkpatrick (2004).

Initial conditions were generated on uniform grids with 100, 200, 400, 800, 1600, 3200, and 6400 cells. Solutions were also computed on adaptive meshes with an effective resolution (if maximally refined everywhere) of 100, 200, 400, 800, 1600, 3200 and 6400 cells. These seven effective resolution grids correspond to seven levels of adaptive mesh refinement. The 100 cell formed the Level 1 adaptive grid, and the `sizemat` parameter was used to increase the potential maximal refinement level. All other mesh refinement variables were left at their default values.

Figure 15 shows solutions to the Mader problem on uniform and adaptive grids. The detonation wave is initiated at x = 5.0 cm and propagates to the left. The top panel compares the analytical and numerical solutions for sound speed, velocity, density, and pressure at 6.25 μs on a 200 cell uniform mesh. These quantities decrease smoothly from the head of the detonation at x=0.0 cm to the center of the slab at 2.5 cm. In this region, the profiles for sound speed, velocity and density are linear with position, while the pressure profile is a cubic. Since the piston is stationary, the constant state profile begins exactly halfway through the domain. Even at this visual level of comparison, one can see differences between the numerical and analytical solutions. In essence, this is because the numerical detonation front does not quite reach x=0.0 cm in 6.25 μs.

The middle and bottom panels of Figure 15 show four stills taken from a movie of the Mader problem with 4 levels of refinement. Circles mark the locations of the adaptive grid points. The total number of grid points and the time is given above each still in the title line. The first three plots are consecutive frames from the movie that show the `flashing' grid phenomena which occurs throughout most of the simulation time. When using the default AMR parameter settings, the adaptive grid appears to refine (218 grid points in the first still), de-refine (148 grid points in the second still), and refine again (220 grid points in the third still). A positive feature is the refinement of regions slightly ahead of the detonation. That is, the detonation propagates into a maximally refined region. Sharp eyes will also notice a clumping of the grid points behind the shock in the first and third stills. That is, there is a dense grouping of grid points followed by a less dense grouping followed by another dense grouping, It is not until quite late in the simulation that the grid behavior stabilizes (132 grid points in the fourth still) and displays no discernible evidence of flashing or clumping.

Figure 16 shows the absolute value of the pointwise $L_{1,\mathrm{rel}}$ error for the density, pressure, and velocity on uniform and adaptive grids at 6.25 μs. Large, persistent errors exist at the x=0 cm left boundary, and are due to the numerical detonation wave not reaching the left boundary by 6.25 μs. The head of the detonation front remains about 0.03 cm away from making contact with and burning the material at the boundary, independent of resolution. Failure to reach the prescribed spacetime point may be due to a slower detonation speed that was introduced by the need to re-create the `.teos` equation of state file, or that the forrest-fire model parameters implemented through the `.teos` file were

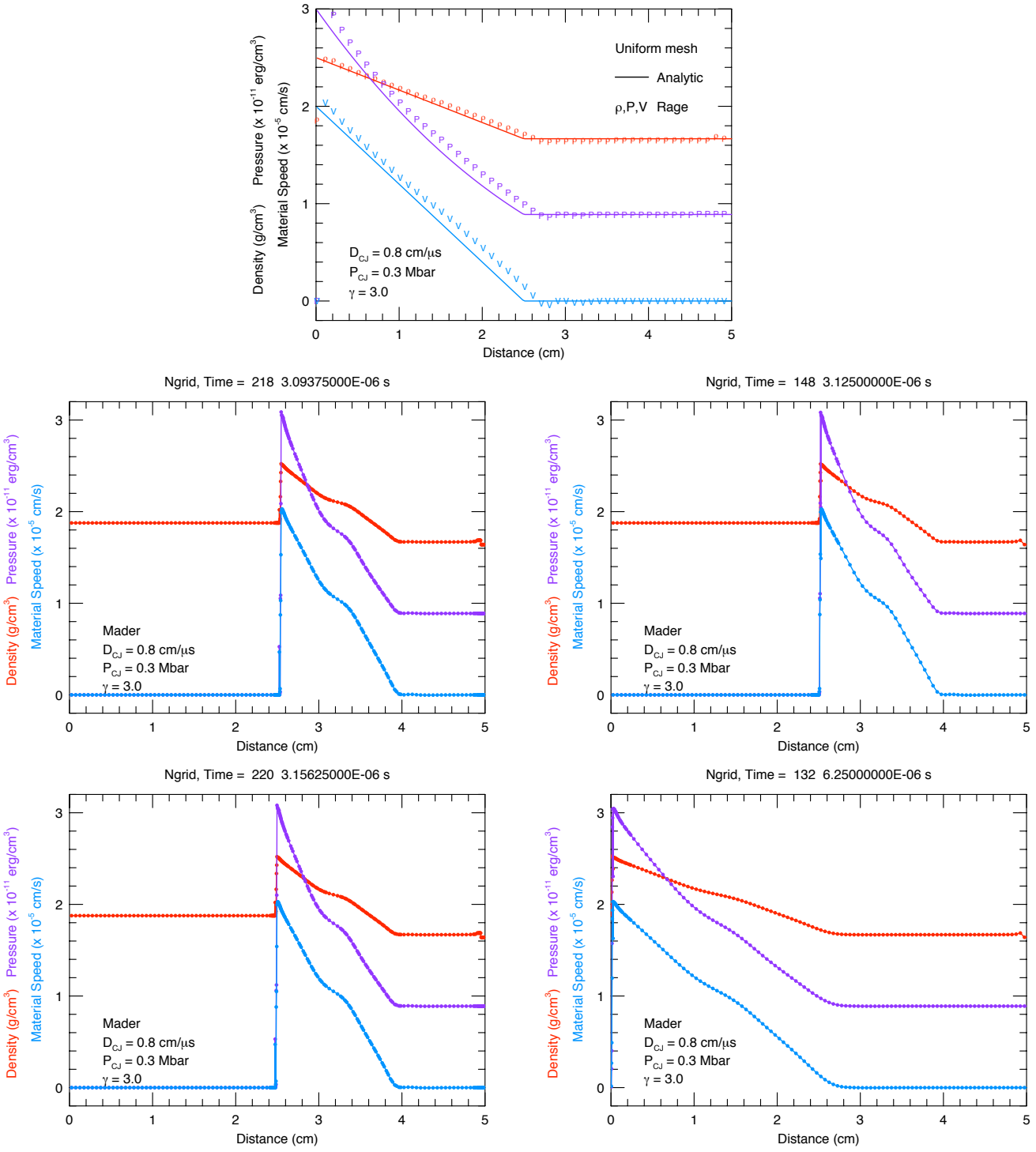

Figure 15. - The top plot shows analytical (curves) and numerical (symbols) solutions for the mass density (red), the velocity (blue), and pressure (purple) at 6.25 μs on a 200 cell uniform mesh. The middle and bottom panels show stills from a movie of the Mader problem with 4 levels of refinement, with circles marking the locations of the adaptive grid points.

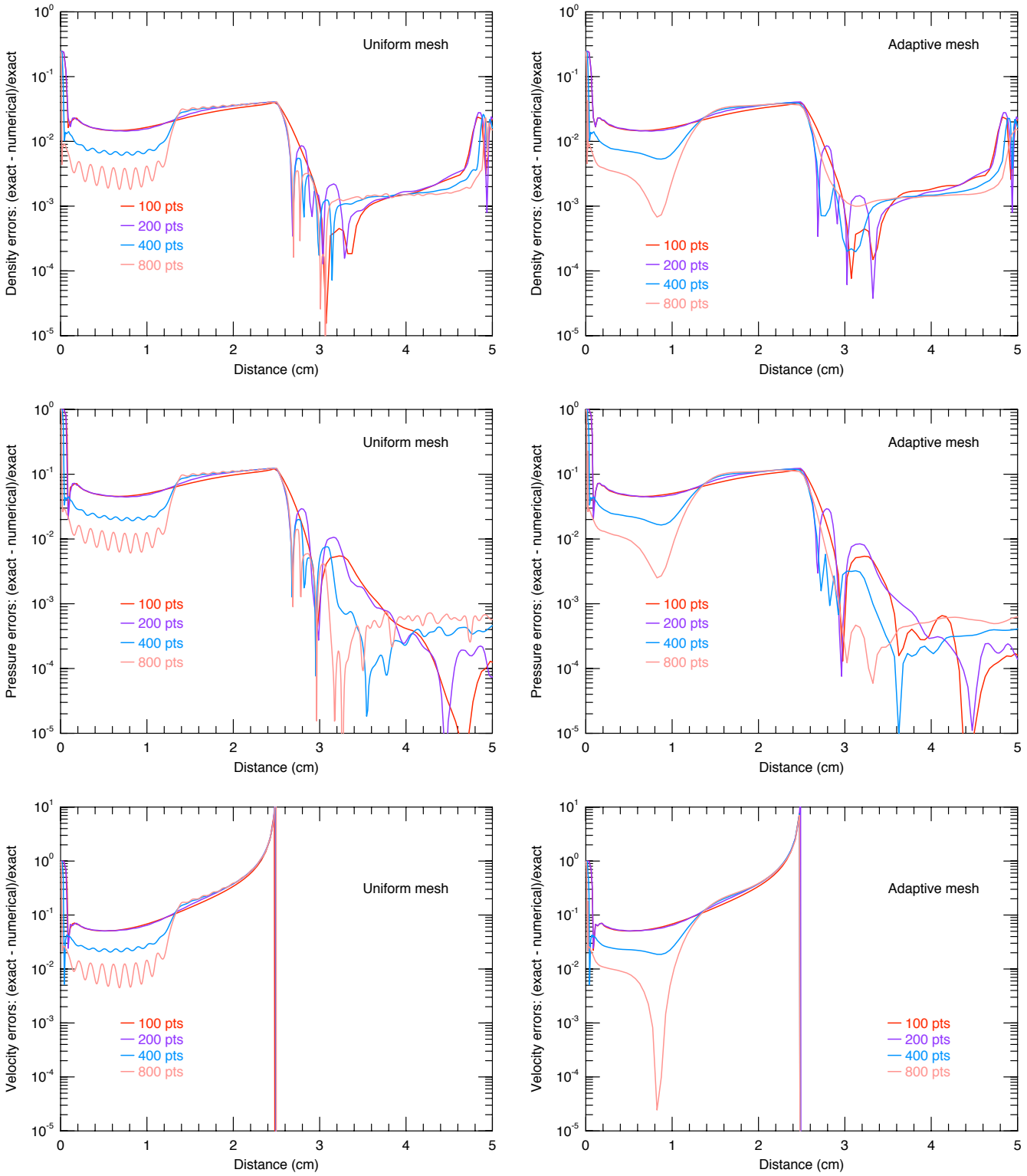

Figure 16. - Absolute value of the point-wise $L_{1,\mathrm{rel}}$ error for the density, pressure, and velocity on uniform and adaptive grids at 6.25 μs.

derived for a uniform grid spacing of 0.025 cm (K. New, private communication 2005, Kamm & Kirkpatrick 2004). This discrepancy or user misunderstanding should be investigated in a follow-up report.

Recall the constant state begins exactly halfway through the domain, at 2.5 cm. In this region, Figure 16 suggests that the errors on uniform and adaptive grid solutions remain constant as the resolution is increased when the default AMR parameter set is used. In the case of the pressure (middle panel) the error becomes larger with increasing resolution. In the rarefaction fan behind the detonation front, $0.0 \leq x \leq 2.5$ cm, the errors generally decrease as the cell spacing decreases on both uniform and adaptive grids.

Figure 17 shows the global error norms on uniform grids computed from equations (7). The rate of convergence on coarse grids is generally quadratic, $q \sim 2$, but degrades at finer grids to sub-linear, $q \leq 0.5$, because of the large persistent errors at the left boundary and transitions to the constant state outside the rarefaction fan. Table 3 details the convergence rate properties for the $L_{1,\mathrm{abs}}$ norm.

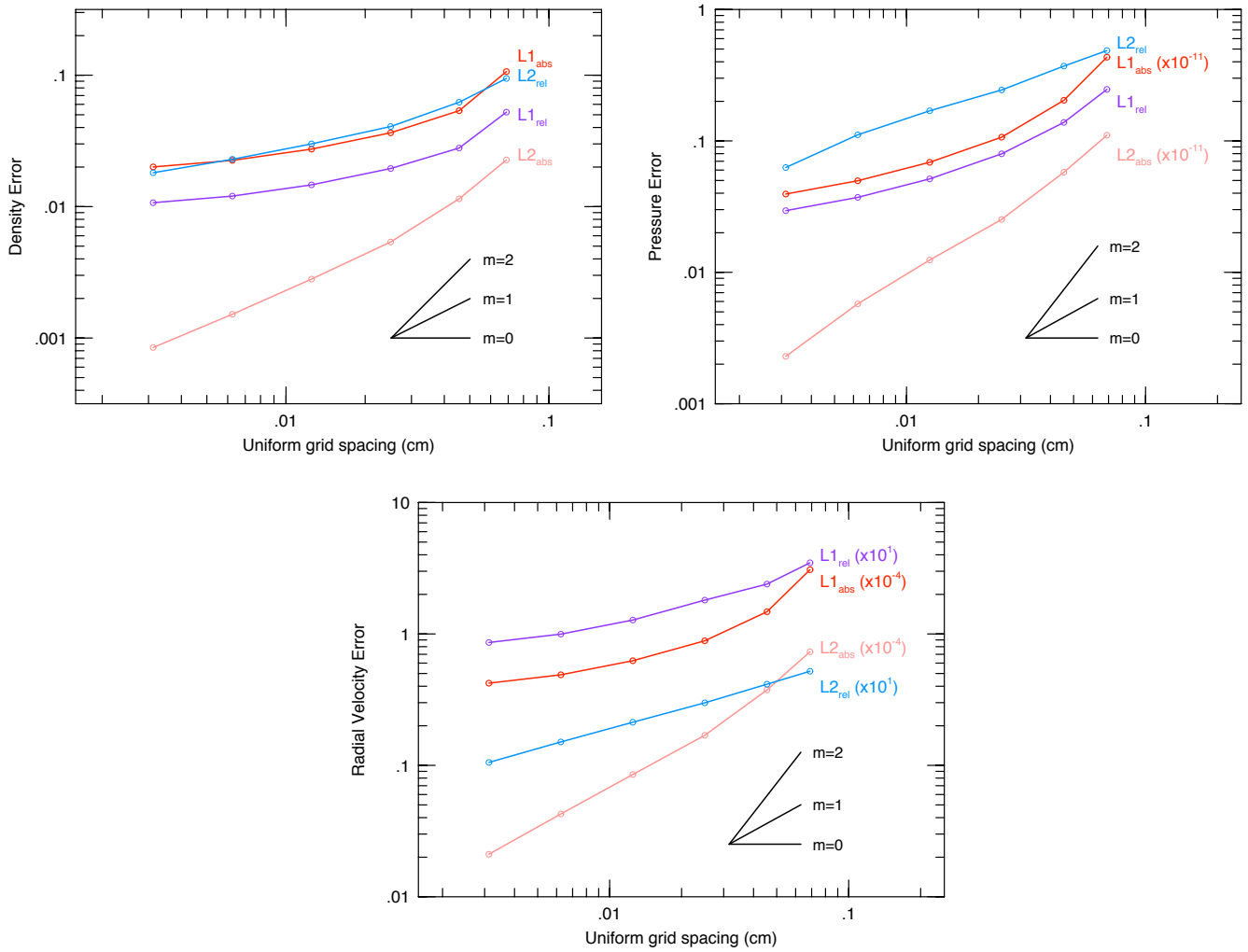

Figure 17. - Global rates of convergence for the mass density, pressure, and material velocity. The black lines show slopes of 0, 1, and 2.

Table 3
Global Convergence coefficients for the Mader problem[1]

	ρ		P		V	
# of cells	q	A	q	A	q	A
100	1.65	131.9	1.82	1.1E+14	1.78	6.5E+07
200	0.65	1.126	1.08	3.3E+12	0.85	8.1E+05
400	0.41	0.328	0.64	3.1E+11	0.50	1.3E+05
800	0.28	0.151	0.47	1.1E+11	0.36	5.3E+04
1600	0.16	0.068	0.34	4.7E+10	0.21	2.0E+04

[1] For the $L_{1,abs}$ norm on uniform grids with NOBEL 20050331.021

The ratio of the CPU resources consumed by the uniform and adaptive grid calculations are shown in Figure 18 as a function of refinement level. The improvement in computational efficiency is about a factor 10 for 6 levels of mesh refinement. This reduction in CPU resources by using adaptive grids is accompanied by a reduction in error that is no worse than the errors incurred by using a uniform grid.

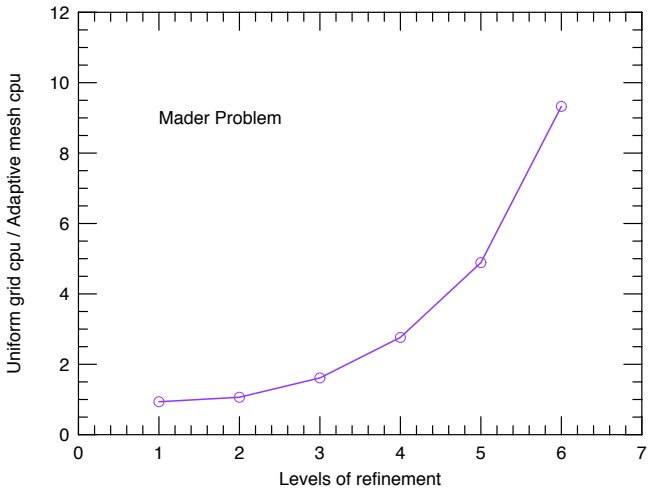

Figure 18. - CPU usage on uniform and adaptive grids for the one-dimensional Mader problem.

3.4 The Reinicke & Meyer-ter-Vehn Problem

The Reinicke Meyer-ter-Vehn (1991, henceforth RMTV) problem considered in the Tri-Lab Verification Test Suite has an initial concentrated energy source of sufficient magnitude so that heat conduction dominates the fluid flow, that is, a thermal front leads a hydrodynamic shock. RMTV examined the self-similar case and found that the fluid equations reduced to a set of four ordinary differential equations (ODEs). Due to evaluation of the initial conditions and multiple-region integration of the complicated ODEs, the RMTV problem has the distinction of possessing the most complicated `analytical' solution in the Tri-Lab Test Suite. Nevertheless, this problem is useful for verifying time-dependent thermal conduction codes in the presence of shocks.

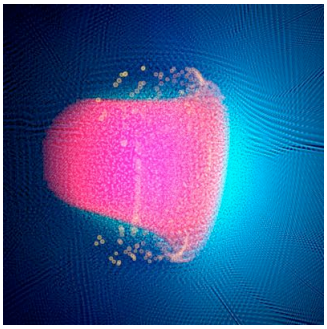

Figure 19. - Smooth Particle Hydrodynamics simulation of a heat front with a following planar shock wave.

3.4.1 Analytical Solution

Given the spherically symmetric domain ($0 \le r \le \infty$), one assumes a polytropic equation of state for the gas,

$$(\gamma - 1)\, E = \frac{P}{\rho} = \Gamma\, T \ , \tag{18}$$

a heat conductivity χ given by

$$\chi = \chi_0\, \rho^a\, T^b \ , \tag{19}$$

and a flow evolving into a cold (T = 0) gas with a power-law density profile

$$\rho_0(r) = g_0 r^\kappa G(\xi) \ . \tag{20}$$

Here γ is the adiabatic exponent, E is the specific internal energy (erg/gr), P is the pressure (erg/cm^3), ρ is the matter density (gr/cm^3), Γ is the Gruneisen coefficient, T is the matter temperature (eV), r is the distance from the origin g_0 is mass density at the origin, and $G(\xi)$ is a dimensionless function of the self-similar position variable ξ. The RMTV problem in the Tri-Lab Test Suite (see Kamm 2000a) considers the case n=3 (spherical), γ=5/4, Γ=1.0. χ_0 = 1.0, a=-2.0, b=6.5, g_0=1.0.

The analytic solution begins by forming

$$\alpha = \frac{2b - 2a + 1}{2b - (n+2)a + n}$$

$$\sigma = \frac{2b - 1}{\alpha(1-a)}$$

$$\kappa = \frac{(2b-1)n + 2}{2a - 2b - 1}$$

$$\mu = \frac{2}{(\gamma - 1)} \quad . \tag{21}$$

The scale factor ζ, of the self similar solution is then given by

$$\zeta = \left[\frac{1}{\alpha} \left(\frac{\beta_0 \, \Gamma^{b+1} \, g_0^{1-a}}{2\chi_0} \right)^{1/(2b-1)} \right]^{\alpha} \quad , \tag{22}$$

where β_0 is an eigenvalue such that the solution behaves properly near the origin. Kamm (2000a) found $\beta_0 = 7.197534 \times 10^7$, which we adopt for the purposes of automating a version of the RMTV problem.

This problem is also unique since the position of the thermal front is specified and the time required for the solution to reach this point is derived. Here the leading edge of the thermal conduction front is at $r_{\rm front}$ = 0.9 cm in physical space and $\xi_{\rm front}$ = 2.0 in self-similarity space. The physical time corresponding to these specified positions is

$$t = \left(\frac{r_{\rm front}}{\zeta \, \xi_{\rm front}} \right)^{1/\alpha} \quad . \tag{23}$$

The position of the shock front is also specified to be at $\xi_{\rm shock}$ = 1.0. The shock front position in physical space is then given by

$$r_{\rm shock} = \zeta \, \xi_{\rm shock} \, t^{\alpha} \quad . \tag{24}$$

Integration of the ODEs starts from before the heat front and proceeds through the heat front, through the shock front, and finally towards the origin. To obtain the initial conditions one seeks the value of a variable $U_{\rm start}$ where the function

$$f(U_{\rm start}) = \ln(1-\epsilon) + \beta_0 \, \xi_{\rm front}^{(2b-1)/\alpha} \int_0^{U_{\rm start}} y^{b-1} \, (1-y)^{b-a} \, \frac{1-2y}{\mu - (\mu+1)y} \, {\rm d}y \tag{25}$$

is zero. The variable $U_{\rm start}$ is related to the fluid velocity and is the initial condition for one of the four ODEs. In this root-find, ϵ is a small value, taken to be 1×10^{-12}. With $U_{\rm start}$ determined, one forms the abscissa from which the integration begins

$$\xi_{\rm start} = \xi_{\rm front} \, \exp \left[\beta_0 \, \xi_{\rm front}^{(2b-1)/\alpha} \int_0^{U_{\rm start}} y^{b-1} \, (1-y)^{b-a} \, \frac{1-2y}{\mu - (\mu+1)y} \, {\rm d}y \right]$$

$$r_{\rm start} = \zeta \, \xi_{\rm start} \, t^{\alpha} \quad , \tag{26}$$

along with the initial conditions for the remaining three ODEs

$$H_{\rm start} = \frac{1}{(1-U_{\rm start})\,\xi_{\rm start}^{\sigma}}$$

$$W_{\rm start} = \frac{\mu-(\mu+1)\,U_{\rm start}}{2}$$

$$\Theta_{\rm start} = U_{\rm start}(1-U_{\rm start}) \quad . \tag{27}$$

A solution is desired at position $r_{\rm want}$, which corresponds to a self-similarity position

$$\xi_{\rm want} = \frac{r_{\rm want}}{\zeta\, t^{\alpha}} \quad . \tag{28}$$

If the solution is desired well ahead of the thermal front, $r_{\rm want} \geq r_{\rm start}$, then the physical solution is just that of the cold material into which the disturbance propagates:

$$\rho = g_0\, r^{\kappa}$$

$$v = 0.0 \qquad E = 0.0 \qquad P = 0.0 \qquad T = 0.0 \quad . \tag{29}$$

If the solution is desired at a position between the leading edge of the thermal conduction front and the shock front, $r_{\rm shock} < r_{\rm want} < r_{\rm start}$, form the following five auxiliary functions

$$\Omega = \frac{W\, H^{1-a}}{\Theta^{b}\,\beta_0}$$

$$g_1 = \sigma - (n+\kappa+\sigma)U$$

$$g_2 = U(\frac{1}{\alpha} - U) + \Theta(2\Omega - \kappa - \sigma)$$

$$g_3 = \Omega(\mu(U-1)+2W) + \mu(\frac{1}{\alpha}-1) - nU - (n+\kappa+\sigma)W$$

$$g_4 = -2(1+\Omega) \quad , \tag{30}$$

and integrate the following four ordinary differential equations

$$U' = g_1 - \frac{(U-1)(g_2-(U-1)g_1))}{\Theta-(U-1)^2}$$

$$H' = \frac{W(g_2-(U-1)g_1)}{\Theta-(U-1)^2}$$

$$W' = g_3 - (U' + W\,\frac{H'}{H})$$

$$\Theta' = \Theta\, g_4 \quad , \tag{31}$$

from $\ln(\xi_{\rm start})$ to $\ln(\max(\xi_{\rm shock}, \xi_{\rm want}))$. At the end of this integration one has values of the four variables $U_{\rm want}$, $H_{\rm want}$, $W_{\rm want}$, and $\Theta_{\rm want}$.

If the solution is desired at a point between the shock front and near the origin, $r_{\rm small} < r_{\rm want} < r_{\rm shock}$, first integrate to the position of the shock using the equation (14). Then apply the standard shock jump conditions to form a new set of initial conditions:

$$
\begin{aligned}
U_{\rm start} &= 1 - \frac{\Theta_{\rm want}}{1 - U_{\rm want}} \\
H_{\rm start} &= H_{\rm want} \frac{(1 - U_{\rm want})^2}{\Theta_{\rm want}} \\
W_{\rm start} &= \frac{\Theta_{\rm want} W_{\rm want} - \frac{(1-U_{\rm want})^4 - \Theta_{\rm want}^2}{2(1-U_{\rm want})}}{(1 - U_{\rm want})^2} \\
\Theta_{\rm start} &= \Theta_{\rm want} \quad .
\end{aligned}
\tag{32}
$$

The ODEs in equation (32) are integrated again from $\ln(\xi_{\rm shock})$ to $\ln(\xi_{\rm small})$, where $\xi_{\rm small}$ was chosen to be 1×10^{-6} in order to avoid the singularity at the origin. At the end of this second integration one has the values of the four variables $U_{\rm want}$, $H_{\rm want}$, $W_{\rm want}$, and $\Theta_{\rm want}$.

Finally, for any point between the leading edge of the thermal front and near the origin, $r_{\rm small} < r_{\rm want} < r_{\rm start}$, the solution in physical space is given by:

$$
v = \alpha \, \frac{r}{t} \, U_{\rm want} \qquad\qquad \rho = g_0 \, r^{\kappa} \, \xi^{\sigma} \, H_{\rm want}
$$

$$
E = \frac{(\alpha r/t)^2}{\gamma - 1} \, \Theta_{\rm want} \qquad\qquad P = (\gamma - 1) \, \rho \, E \qquad\qquad T = \frac{(\alpha r/t)^2}{\Gamma} \, \Theta_{\rm want} \quad . \tag{33}
$$

A fortran code to calculate the analytic solution is given in appendix B, and is based on the codes used in Kamm (2000a).

3.4.2 Verification Analysis on 1D Uniform and Adaptive Grids

RAGE 20041126.030, 20041126.032, 20050331.02, 20050331.31, 20050331.41 and 20050818.0000 was used to generate numerical solutions to the RMTV problem on the Linux cluster Lambda. The starting RAGE input deck for this problem was the same one used by Kamm & Kirkpatrick (2004), and modified to make it more amenable to automation and analysis on adaptive meshes (see Appendix A). Initial conditions (see Appendix B) were evolved to $5.1251245293611 \times 10^{-10}$ s and Grove's AMHCTOOLS (2005) was used to extract the solution data on the native grid from the binary dump files. Extracting the solution data on the native mesh is important because if one requests the simulation data from RAGE in either HDF 4 or SDSS formats, RAGE interpolates the native data onto a uniform mesh. Avoiding an interpolation of the numerical solution right off the bat seems prudent for a proper verification analysis.

Solutions were generated on uniform grids with 100, 200, 400,and 800 cells. Solutions were also computed on adaptive meshes with an effective resolution (if maximally refined everywhere) of 100, 200, 400, and 800 cells. These four effective resolution grids correspond to four levels of adaptive mesh refinement. The 100 cell formed the Level 1 adaptive grid, and the `sizemat` parameter was used to increase the potential maximal refinement level. Other mesh refinement variables such as `numfine`, `numlev` `mxcells` and `sizbnd` were left at their default values.

RAGE core dumped when trying to run 1600 (or more) cells finer grid resolutions:

```
1600 1600 1 0 2.501650E-03 2.501650E-03 ...
initialize region
GMV cannot open file
Unable to open file
scloadx3d:  unable to open file ‘‘’’ for reading
scloadx3d:  No such file or directory
Segmentation fault (core dumped)
```

This is probably due to the crude manner in which the initial power-law density profile given by equation (20) was implemented. Every zone was made a separate region. RAGE must have some limit on the number of regions allowed, and we may have simply exceeded that limit. While this could be investigated further, it is far more preferable to find a better way to implement the initial density profile. In either case, note the initial power-law density profile must be cell averaged. For a zone with a left boundary `rl`, a right boundary `rr` the mass and volume of the zone is

$$M = \frac{4\pi g_0}{3+\kappa}\left[\mathtt{rr}^{3+\kappa} - \mathtt{rl}^{3+\kappa}\right] \qquad V = \frac{4\pi}{3}\left[\mathtt{rr}^{3} - \mathtt{rl}^{3}\right] \ , \tag{35}$$

so that the (constant) density within the zone is $\rho = M/V$.

Figure 20 shows solutions to the RMTV problem on uniform and adaptive grids. The wave structure is initiated at x = 0.0 cm and propagates to the left. The top panel compares the analytical and numerical solutions for the density, pressure, temperature and material speed at $5.1251245293611 \times 10^{-10}$ s on a 200 cell uniform mesh. The analytic and numerical solutions appear reasonable at this level of visual comparison, although there is a difference in the locations of the thermal front's leading edge (green curve).

The middle and bottom panels of Figure 20 show four stills taken from a movie of the RMTV problem with 4 levels of refinement. Circles mark the locations of the adaptive grid points. The total number of grid points and the time is given above each still in the title line. The first frame from the movie, left plot of the middle panel, shows RAGE's gallant attempt to refine the poor manner in which the initial power-law density profile was implemented. Note the general trend of a decreasing number of grid points as the simulation proceeds. As in the Mader problem (see Figure 15), there is a clumping of the grid points behind the shock which is particularly visible in the pressure (purple curves). There is a dense grouping of grid points followed by a less dense grouping followed by another dense grouping. Such clumping persists even until the ending time (final movie frame). It is possible in this case that the grid clumping is an artificial feature caused by the manner in which the initial power-law density profile was put into RAGE.

Figure 21 shows the absolute value of the pointwise $L_{1,\mathrm{rel}}$ error for the density, temperature, and material speed on uniform and adaptive grids. The large errors near the left boundary are caused by the singularity at the origin.

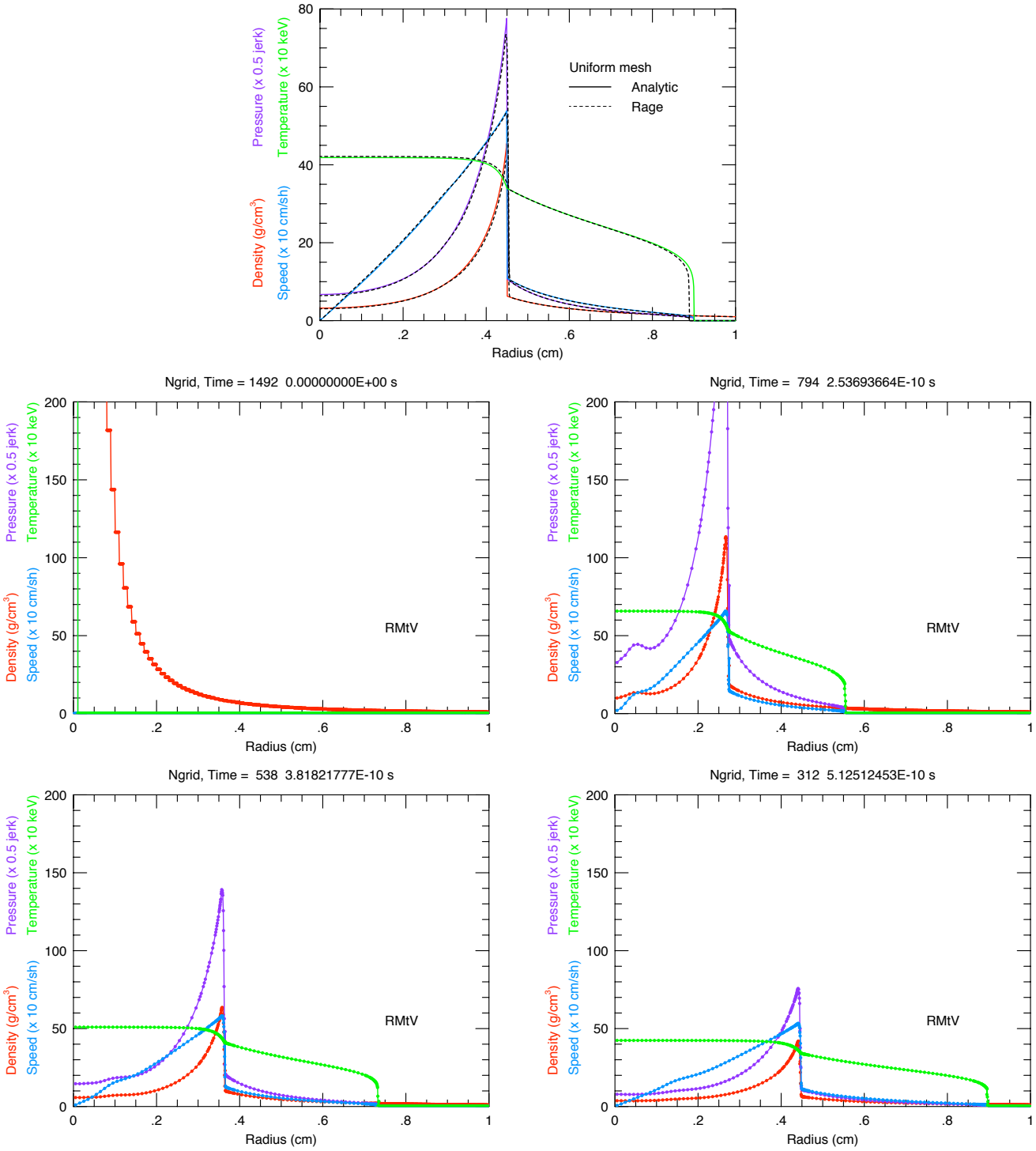

Figure 20. - The top plot shows analytical (solid) and numerical (dashed) solutions for the mass density, velocity, pressure, and temperature at $5.1251245293611 \times 10^{-10}$ s on a 200 cell uniform mesh. The middle and bottom panels show stills from a movie of the RMTV problem with 4 levels of refinement, with circles marking the locations of the adaptive grid points.

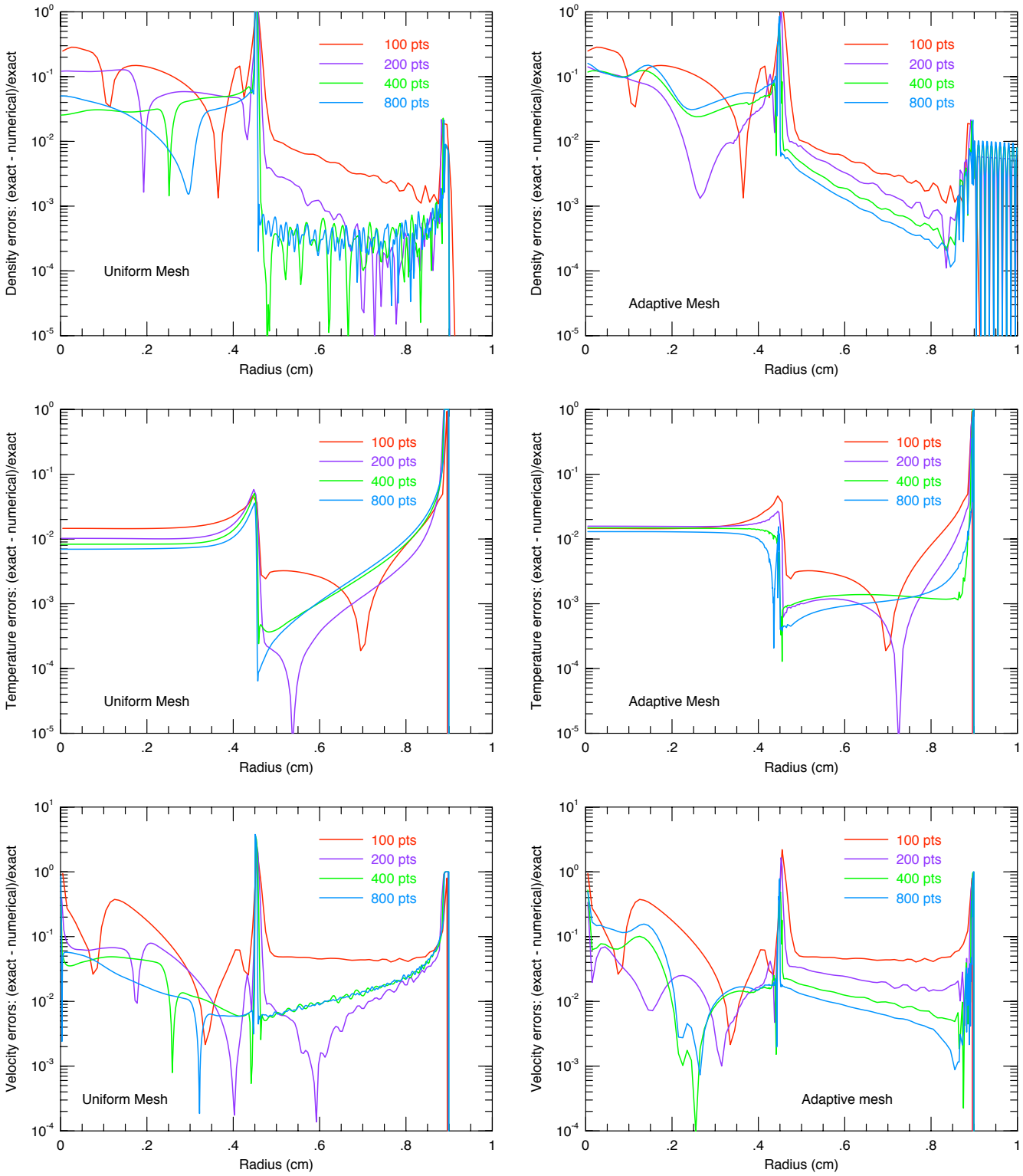

Figure 21. - Absolute value of the point-wise $L_{1,\mathrm{rel}}$ error for the density, pressure, and speed on uniform and adaptive grids at $5.1251245293611\times10^{-10}$ s.

There are also insistent, but expected, errors near the r=0.45 cm position of the shock front and the r=0.9 cm position of the heat front. Of more concern is the behavior of the numerical solution on both uniform and adaptive grids. First consider the region between the origin and shock at 0.45 cm. On uniform grids the convergence appears to be saturating by 800 cells. On adaptive grids with the default AMR seetings the solution gets worse with increasing resolution --- the 800 cell solution has larger errors than the 400 cell solution which in turn has larger errors than the 200 cell solution. Next consider the region between the shock front at 0.45 cm and the thermal front at 0.90 cm. On uniform grids the errors in the density solution saturate, but the temperature and velocity errors increase with increasing resolution. On adaptive grids, the errors continue to get smaller with increasing resolution.

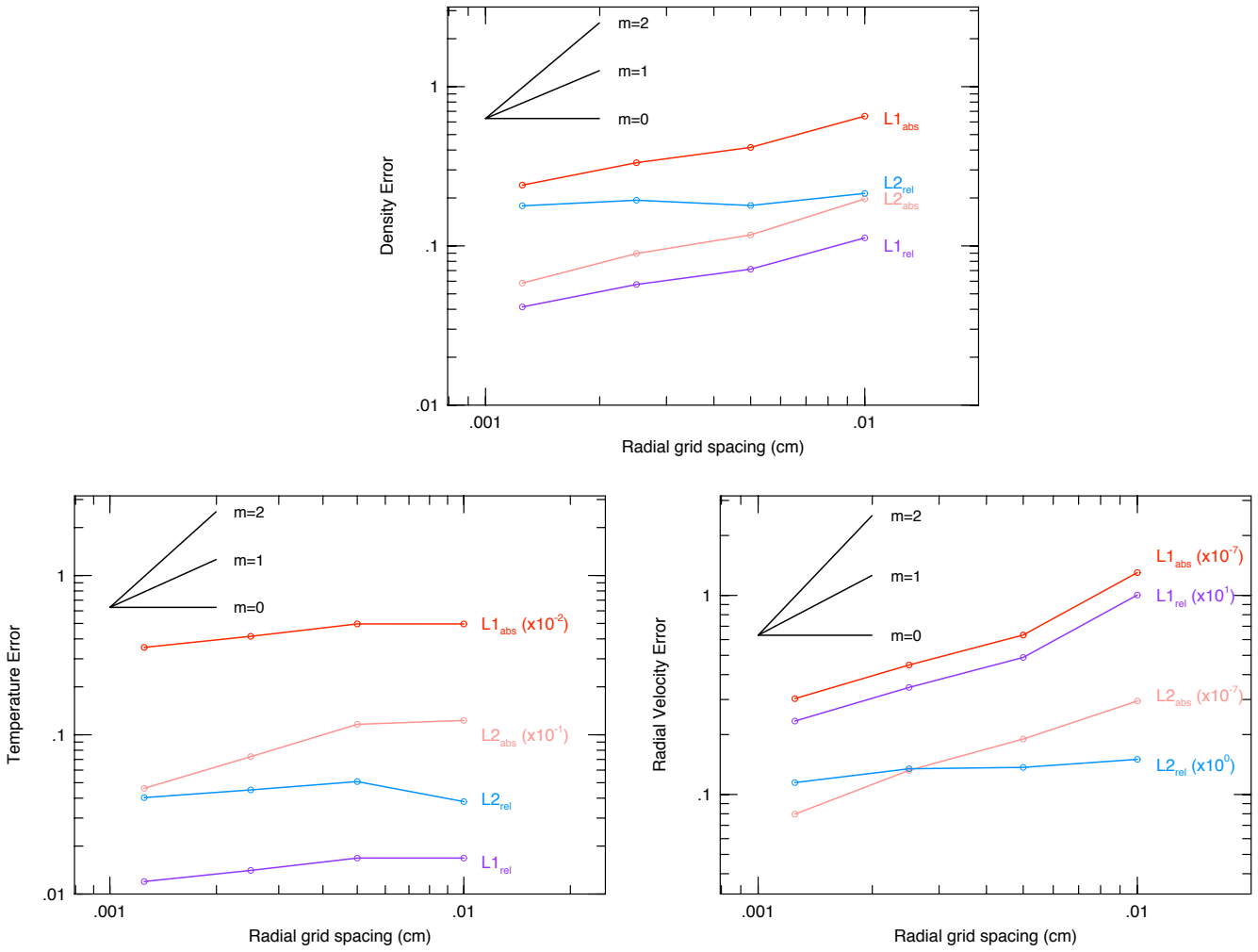

Figure 22. - Global rates of convergence for the mass density, pressure, and material velocity. The black lines show slopes of 0, 1, and 2.

Figure 22 shows the global error norms on uniform grids computed from equations (7). The rate of convergence on coarse grids is generally linear, $q \sim 1$, for the density and material velocity but degrades at finer grids to sub-linear,

$q \leq 0.4$. The convergence rate of the temperature is worse on coarse grids and gets marginally better, q $\sim$ 0.25 on finer grids. A few runs with a smaller maximum allowed time-step did not significantly change the convergence rates at the finest grid resolutions. The reason for this convergence behavior is the persistent errors at the origin, shock front, and thermal front, along with the undesirable conduct of the errors with increasing resolution between the shock front and the thermal front. Table 4 details the convergence rate properties of the $L_{1,abs}$ norm.

Table 4
Global Convergence coefficients for the Reinicke Meyer-ter-Vehn Problem[1]

	ρ		v		P		T	
# of cells	q	A	q	A	q	A	q	A
200	0.65	13.1	1.04	1.5E+07	0.76	8.5E+17	0.00	49.6
400	0.31	2.25	0.50	8.9E+07	0.42	1.4E+17	0.29	195
800	0.47	5.50	0.56	1.3E+08	0.57	3.4E+17	0.23	166

[1] For the $L_{1,abs}$ norm on uniform grids with RAGE 20050818.0000

The ratio of the CPU resources consumed by the uniform and adaptive grid calculations are shown in Figure 23 as a function of refinement level. There is almost no improvement in computational efficiency using adaptive grids because initially there are more cells than 'specified' by `sizmat` and the domain size. For a domain size of 1.0 cm and `sizmat` set to 0.00125 there should be, at maximum, 800 grid points. Yet Figure 20 shows a maximum of 1429 cells at t=0.0 s. Why this is occurring should be addressed in future efforts.

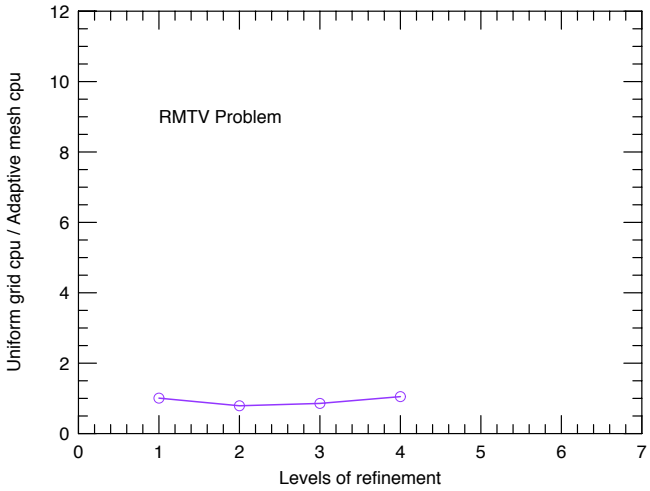

Figure 23. - CPU usage on uniform and adaptive grids for the one-dimensional RMTV problem.

3.5 The Noh Problem

The Noh problem (Noh 1987) is a standard verification problem for hydrocodes. A gamma-law gas is initialized with a uniform, radially inward speed. A shock forms at the origin and propagates outward as the gas stagnates. This problem tests a code's ability to transform kinetic energy into internal energy, and the ability to follow supersonic flows. The analytical solution is easy to calculate, and the convergence of the hydrocode solution can be directly determined.

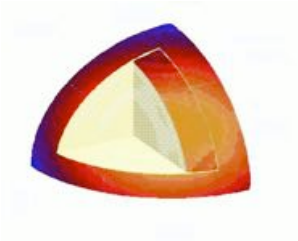

Figure 24. - Noh's problem in 3D - spherical shock

3.5.1 Analytical Solution

Given the domain characterized at t=0 by a constant density ρ_0, material inflow speed, v_0, and adiabatic index γ

$$\rho_0 = 1.0 \qquad v_0 = -1.0 \qquad \gamma_0 = 5/3 \ , \tag{36}$$

the solution at time t=0.3 is desired for any given value of $r_{\rm want}$. The shock front speed and location are

$$v_s = \frac{1}{2}(\gamma - 1)\,|v_0| \qquad r_s = v_s\,t \ . \tag{37}$$

If the solution is desired ahead of the blast wave, $r_{\rm want} \le r_2$, then

$$\rho = \rho_0 \left(1 - \frac{v_s\,t}{r_{\rm want}}\right)^{n-1} \qquad v = v_0 \qquad E = 0 \qquad P = 0 \ , \tag{38}$$

where $n = 3$ is for a spherical shock, $n = 2$ for a cylindrical shock, and $n = 1$ for a planar shock. If the solution is between the origin and the shock front, $r_{\rm want} > r_2$, then

$$\rho = \rho_0 \left(\frac{\gamma+1}{\gamma-1}\right)^n \qquad v = 0 \qquad E = \frac{1}{2} = v_0^2 \qquad P = (\gamma-1)\rho E \ . \tag{39}$$

For the canonical case of γ = 5/3 these equations yield the well-known result that a strong planar shock produces a maximum density contrast of 4 for a planar shock, 16 for a cylindrical, and 64 for a spherical shock.

3.5.2 Verification Analysis on 1D Uniform and Adaptive Grids

RAGE 20041126.030, 20041126.032, 20050331.02, 20050331.31, 20050331.41 and 20050818.0000 was used to generate numerical solutions to the 1D Noh problem on the Linux cluster Lambda. An inflow boundary condition

is imposed to preserve the inward flow indicated by equation (36). The starting RAGE input deck for this problem was the same one used by Kamm & Kirkpatrick (2004), and modified to make it more amenable to automation and analysis on adaptive meshes (see Appendix A). Grove's AMHCTOOLS (2005) was used to extract the solution data on the native grid from the binary dump files.

Solutions were generated on 1D uniform grids with 100, 200, 400, 800, 1600, 3200, and 6400 cells at t=0.3 s. Solutions were also computed on 1D adaptive meshes with an effective resolution (if maximally refined everywhere) of 100, 200, 400, 800, 1600, 3200, 6400 cells. These seven effective resolution grids correspond to seven levels of adaptive mesh refinement. The 100 cell formed the Level 1 adaptive grid, and the `sizemat` parameter was used to increase the potential maximal refinement level. Other mesh refinement variables such as `numfine`, `numlev` `mxcells` and `sizbnd` were left at their default values.

Figure 25 shows solutions to the 1D Noh problem on uniform and adaptive grids. The top panel compares the analytical and numerical solutions for the density, pressure, temperature and material speed on a 800 cell uniform grid. RAGE, like most other hydrodynamics codes, produces anomalous `wall-heating' near the origin. This heating causes premature stagnation, with densities lower than predicted in the centermost cells. In this RAGE calculation, the central zones has a stagnation density above 75 g/cc. Further out, matter stagnates at densities of 58-62 g/cc. The extent to which the anomalous heating occurs depends on the nature of the discontinuity at the origin, so the wall heating can be considerably less than this (sometimes completely negligible) in realistic problems. If the converging flow does not produce a 1-zone discontinuity in velocity at the origin (as occurs in the setup of the Noh test problem), RAGE's performance might be adequate.

The middle and bottom panels of Figure 25 show four stills taken from a movie of the 1D Noh problem with 4 levels of refinement when the default AMR parameters are used. Circles mark the locations of the adaptive grid points. The total number of grid points and the time is given above each still in the title line. Most disconcerting is the poor resolution of the shock front with spikes in front of the shock wave and just behind the shock wave. There is also evidence of grid clumping ahead of the shock front; a dense grouping of grid points followed by a gap followed by another dense grouping. Such clumping persists even until the ending time.

Figure 26 plots the $L_{1,\mathrm{rel}}$ error for the density, pressure, and material speed for various grid resolutions. Note the density plots in the top panel have a different x-axis scale.The large errors from the anomalous heating at the origin is evident in both the uniform and adaptive grid simulations. The large errors near the right boundary is present in the uniform and adaptive grid model and is probably due to how the inflow boundary condition is implemented in RAGE. It is encouraging, however, that between the origin and the chock from that both uniform and adaptive grid calculations show a steady decrease in the size of the errors as the spatial resolution is increased.

Figure 27 shows the global error norms on uniform grids computed from equations (7). The rate of convergence of the density on coarse grids is generally sub-linear, $q \leq 0.5$, and doesn't significantly improve or degrade as resolution increases. The pressure and material speed have convergence rates that are nearly linear at all grid resolutions. A few runs with a smaller maximum allowed time-step did not significantly change the convergence rates at the finest grid resolutions. The reason for this convergence behavior is due to wall heating and errors in the density far upstream from the shock front. Table 5 details the convergence rate properties of the $L_{1,\mathrm{abs}}$ norm.

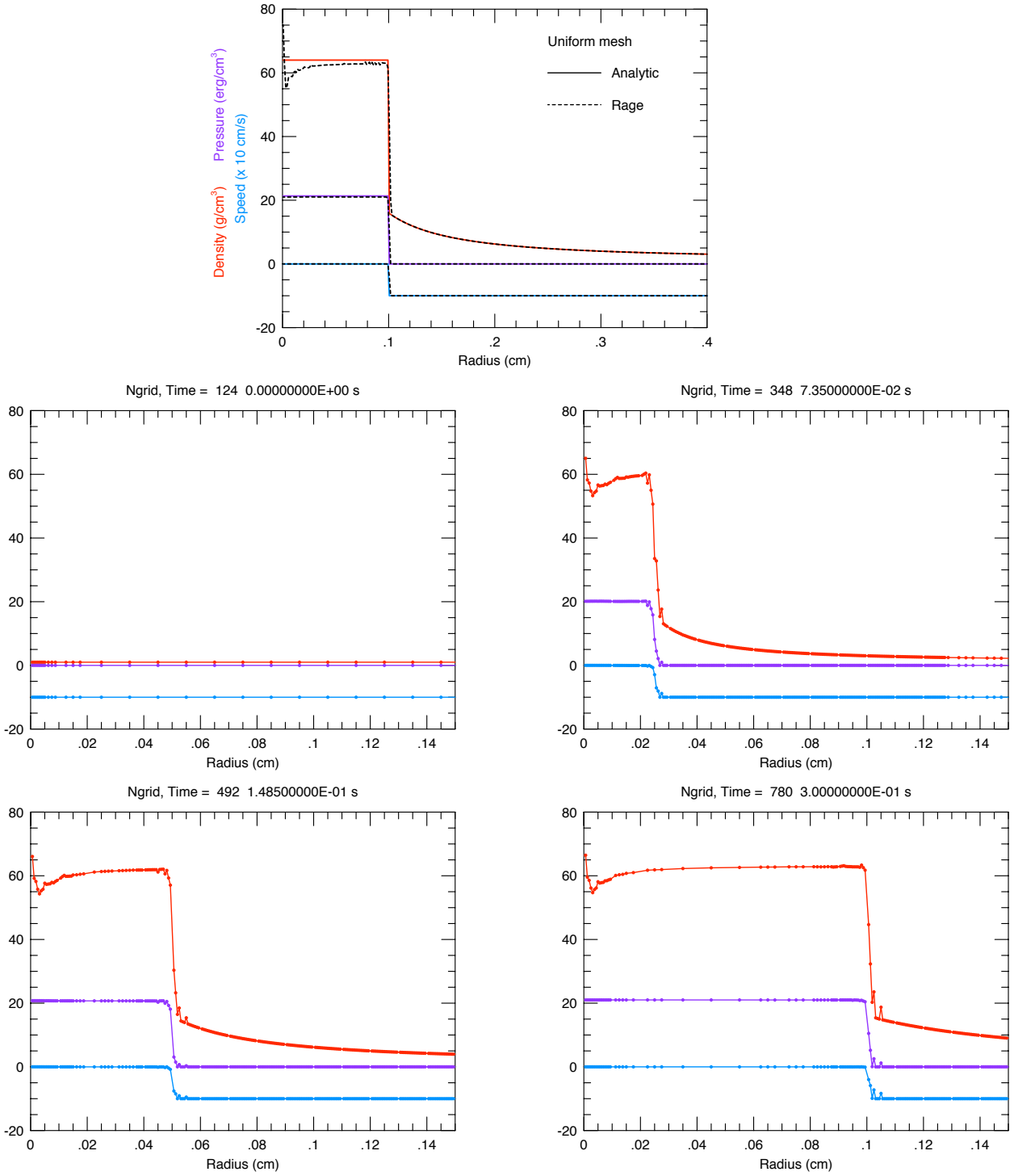

Figure 25. - The top plot shows analytical (solid) and numerical (dashed) solutions for the mass density, pressure and material speed at 0.3 s on a 800 cell uniform grid. The middle and bottom panels show stills from a movie of the 1D Noh problem with 4 levels of refinement, with circles marking the locations of the adaptive grid points.

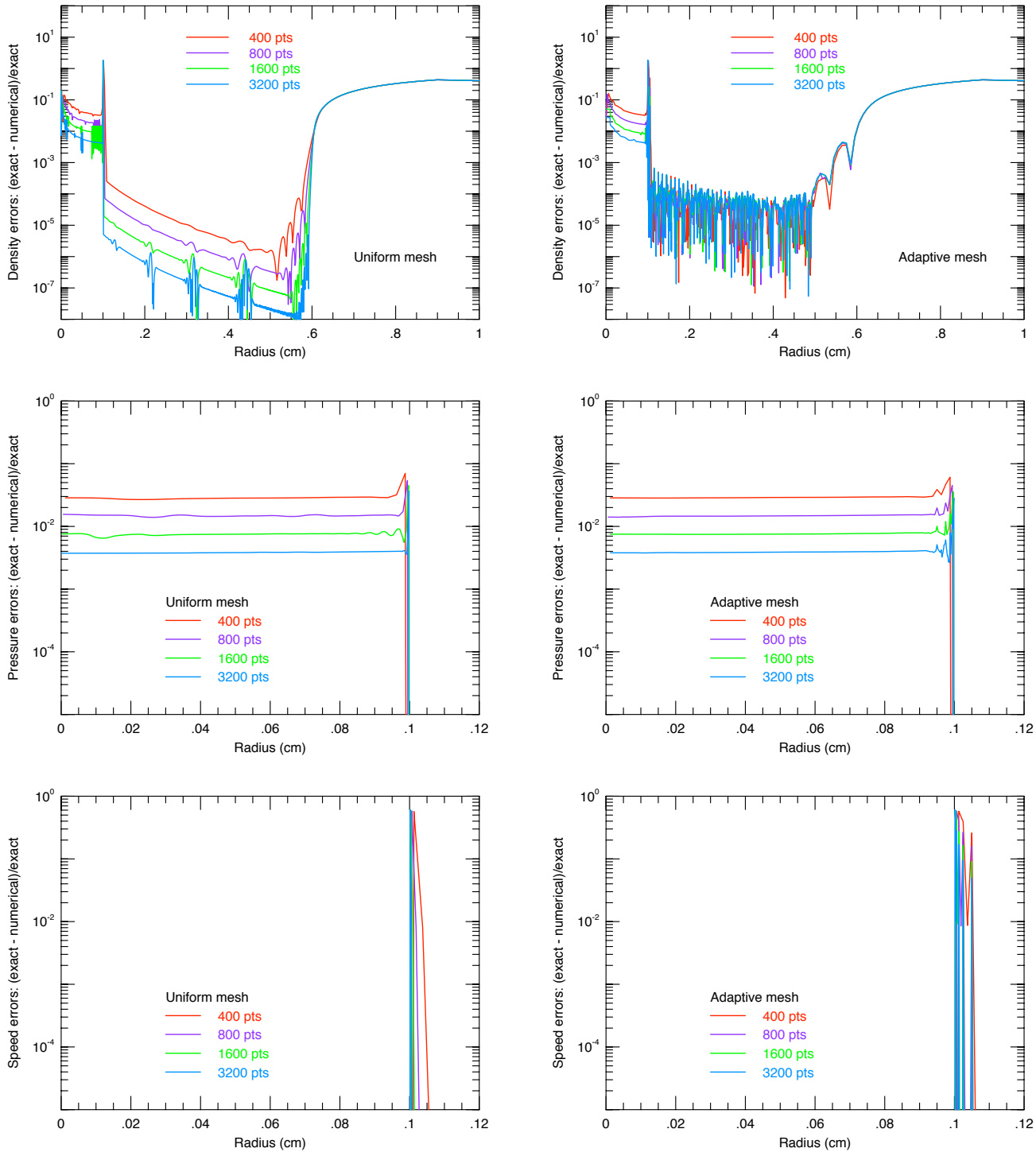

Figure 26. - Absolute value of the point-wise $L_{1,\mathrm{rel}}$ error for the density, pressure, and material speed on uniform and adaptive grids at 0.3 s.

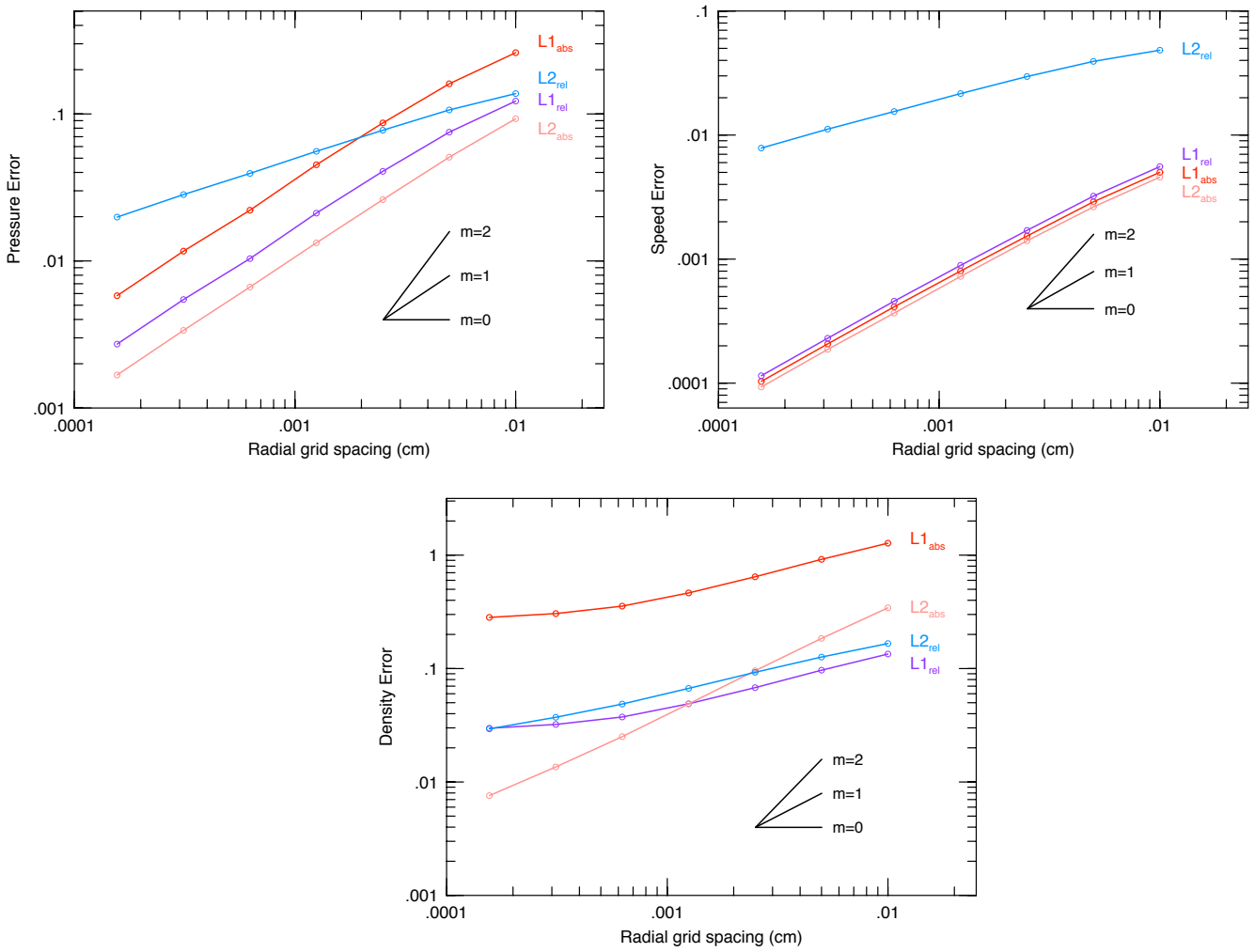

Figure 27. - Global rates of convergence for the mass density, pressure, and material velocity. The black lines show slopes of 0, 1, and 2.

Table 5

Global Convergence coefficients for the Noh Problem[1]

	ρ		v		P	
# of cells	q	A	q	A	q	A
200	0.47	11.2	0.79	0.19	0.70	6.64
400	0.51	13.7	0.92	0.37	0.88	17.4
800	0.47	10.9	0.94	0.41	0.95	25.0
1600	0.38	6.17	0.96	0.48	1.03	43.3
3200	0.21	1.77	0.99	0.62	0.92	20.2
6400	0.11	0.75	1.00	0.68	1.00	39.6

[1] For the $L_{1,abs}$ norm on uniform grids with RAGE 20050818.0000

The ratio of the CPU resources consumed by the uniform and adaptive grid calculations are shown in Figure 28 as a function of refinement level. There is almost no improvement in computational efficiency using adaptive grids because the entire domain is refined within the first few times steps. This is also seen in the 2D Noh problem discussed in the next section. It appears the radially inwards velocity field causes the refinement.

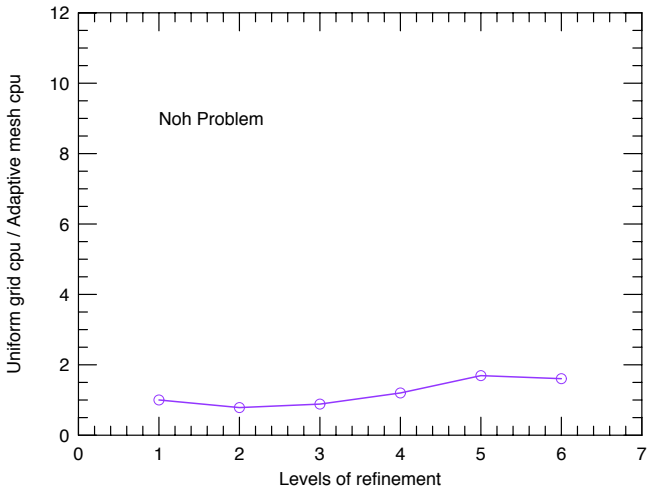

Figure 28. - CPU usage on uniform and adaptive grids for the one-dimensional Noh problem.

3.5.3 Verification Analysis on 2D Uniform and Adaptive Grids

Most modern, complex simulations are done in two and three dimensions. It is important to consider how well hydrocodes duplicate the physics in situations where motions and gradients are not grid-aligned. The 2D spherically symmetric Noh problem is also good for testing the isotropy of successful code solutions. The ideal geometry for simulating the spherically symmetric Noh problem in two dimensions is the (r,z) cylindrical coordinate system, with the axis being one boundary of the problem domain, and the equatorial plane a reflective boundary.

All the 2D simulations were performed with either RAGE 20041126.002, 20041126.032 or 20041126.042. They were mainly run on the QSC HP/Compaq ES245 Alpha cluster, on varying number of processors. Some of the smaller problems were executed on desktop Apple Macintosh G4's under the Absoft Compiler, and a few sample runs on the Linux cluster Flash. No significant differences were found for these problems among these versions and computers. For these 2D simulations, the HDF-4 output routines were used to access the data. In the course of ensuring that the files generated were true 64-bit, one of us (Gisler) consulted with Dale Ranta of SAIC, who discovered a bug in the production of 64-bit scientific data sets by RAGE due to a bad index. This was corrected and incorporated into the current versions of the code. To produce 64-bit scientific data sets one must set `hdf_64bit_sds = .true.`

All the uniform grid simulations were done of a domain [0,1.5] $\times$ [0,1.5] in r-z coordinates. The baseline simulation, henceforth denoted c_1, was computed on a uniform grid of cell size $\Delta z = \Delta r = 0.01$, so the grid is 150$\times$150 cells. Initial conditions and inflow boundary conditions were prescribed as in the 1D Noh problem. The solutions are evaluated at time t=0.6, at which time the shock has reached r=0.2. The uniform grid convergence study

involved simulations c_2, c_4, and c_8, with cell sizes $\Delta z = \Delta r = 0.005$, 0.0025, and 0.00125 respectively, decreasing by a factor two each time. The finest-grid simulation therefore had 1200×1200 cells for a total of 1,440,000 cells.

RAGE produced output files in 64-bit scientific data format on a 600×600 pixel frame. Analytical results were produced on the same sized frame. This frame size allows for easy visual inspection, and having a single output frame size for all resolutions eases the verification analysis. This output grid size is the same as the c_4 computational grid, and differs from the other computational grids by factors of 2 or 4 in one dimension, so potential problems from aliasing should be a moot issue.

Figure 29 shows the numerical solution at the simulation end time (t=0.6) for the four cases in the uniform-grid convergence study, c_1, c_2, c_4, and c_8 arranged clockwise starting from upper left. The images are arranged so that they share a common center for easy comparison of features, but the cylindrical axis is always vertical and the radial axis horizontal. Starting at the top left in Figure 29, notice the two dark lines adjacent to and parallel to the horizontal axis within the shock region and a single dark line similarly adjacent to the vertical axis. These grid aligned features are anomalous artifacts of RAGE's hydrodynamics. There is also a broad valley (lighter region) at 45°. Moving to the top right, there is a distinct sharpening of the shock interface, a narrowing of the 45° valley feature and of the grid-aligned stripes. At bottom right, the grid-aligned stripes narrow further with increased resolution, but are still noticeable, as is the valley. In the finest resolution calculation at bottom left, the pattern is distinctly that of a herringbone, with a rather broad 45 degree spine (the valley seen at lower resolution has become a hump) and narrow grid-aligned ribs. These density plots give impression that the asymmetry due to the numerics is improving with increased resolution, but this is misleading, as will be seen below.

Figure 30 shows the $L_{1,abs}$ norm computed from a generalization of equation 7. If RAGE's solution were a perfect match to the analytical solution, the color would be uniformly dark blue. The same features pointed out in the RAGE density plots are apparent here, but it is also clear that the accuracy of the simulation dramatically improves as the resolution is increased. Not only does the shock thickness decrease, but the overall amplitude of the variations within the shocked region decrease as well. It is also possible to discern in these difference frames the influence of the constant-inflow boundary condition as a faintly visible front marching inward from the outer r and z boundaries, now having reached about 1/3 of the way across the frame. A proper detailed analysis will exclude that region from consideration.

Because conditions in the Noh problem differ so drastically in the regions inside and outside the shock, and also in order to quantify the asymmetry so apparent in Figures 29 and 30, we have chosen to perform convergence and symmetry analyses separately in annular bands centered on the origin. These bands are illustrated in Figure 31, in which we also display a magnified (and flipped) version of the high-resolution plot from the lower-left-hand corner of Figure 29. The bands, numbered 0 through 9 from the innermost radial band outward, are chosen in such a way as to bracket the shock (in band 4) and to have roughly similar numbers of scientific data set cells (i.e., not computational cells) per band, except for the outermost band 9, which we discard from the analysis because of the influence of the boundary. Also shown in Figure 31 are the band-averaged L1 norms as a function of cell size, illustrating that convergence outside the shock is second-order, while convergence within and at the shock is first-order.

The band $L_{1,abs}$ norms are tabulated in Table 6 for the four runs considered here, as well as the convergence properties in each band. For judging the $L_{1,abs}$ norms, recall that the analytical value of the density within the shock

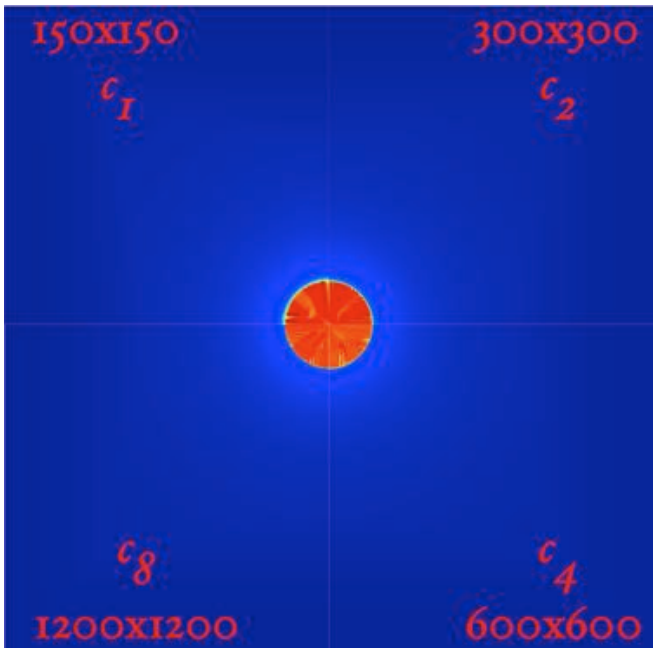

Figure 29. - Density plots of the spherical-compression Noh problem in 2D at t=0.6 s at four successively finer refinement levels, going clockwise from top left. The plots are linear in density and autoscaled. In the analytical solution the density has a value of 64 units inside the shocked region, declining sharply to 1 unit outside the shock.

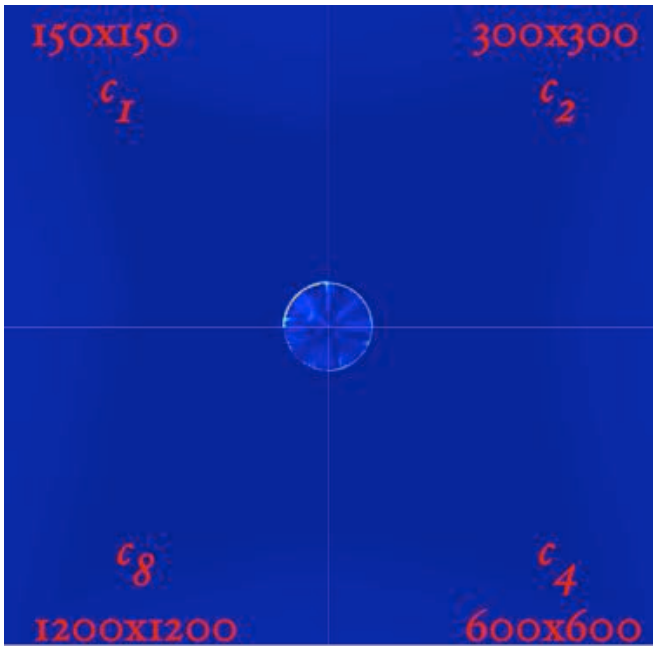

Figure 30. - The $L_{1,\mathrm{abs}}$ norm for the density from the same 4 solutions of the Noh problem shown in Figure 29.

(bands 0 through 4) is 64, while the value outside is very nearly 1. So the solution is highly accurate (and converging rapidly) outside the shock, while within the shock the accuracy is a few percent and converging slowly.

Table 6
Accuracy and convergence for the Noh problem on 2D uniform grids

	Band averaged L1 norms				Pointwise convergence indices		
band	c_1	c_2	c_4	c_8	q_1	q_2	q_3
dx	0.01	0.005	0.0025	0.00125			
0	3.77954	2.52065	1.90834	1.47468	0.58	0.40	0.37
1	4.04392	2.70827	1.9326	2.54857	0.58	0.49	-0.40
2	4.08581	2.62225	2.68611	3.00168	0.64	-0.03	-0.16
3	4.33881	2.76868	3.58724	3.64777	0.65	-0.37	-0.02
4	15.2744	8.54892	6.26586	3.56781	0.84	0.44	0.81
5	0.54414	1.65e-02	6.31e-03	9.06e-04	5.04	1.39	2.80
6	0.03284	1.20e-02	4.65e-03	5.86e-04	1.46	1.36	2.99
7	2.74e-02	1.04e-02	4.18e-03	4.21e-04	1.40	1.32	3.31
8	2.35e-02	9.21e-03	3.82e-03	2.97e-04	1.35	1.27	3.68

Table 7
Asymmetry in the 2D Noh problem on uniform grids

	rms deviation of band means			
band	c_1	c_2	c_4	c_8
dx	0.01	0.005	0.0025	0.00125
0	1.74058	1.78574	1.80998	1.8383
1	1.7499	1.80254	1.8445	1.8555
2	1.76745	1.81975	1.84371	1.86319
3	1.76173	1.81585	1.82221	1.8412
4	1.2892	1.13534	1.06157	0.986527
5	0.409548	0.394162	0.394108	0.394155
6	0.350507	0.350276	0.350265	0.350304
7	0.318307	0.318171	0.31818	0.318229
8	0.289846	0.289774	0.289796	0.289849

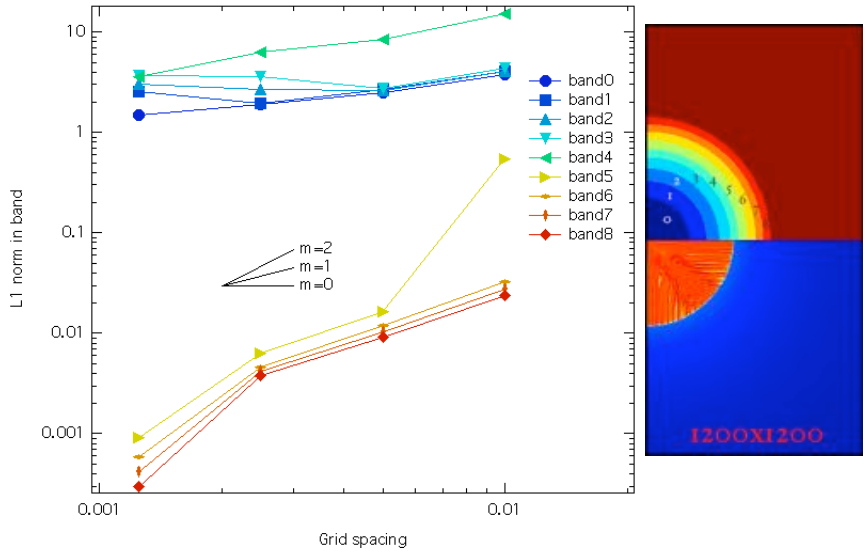

Figure 31. - The right image shows analysis bands 0 through 8 and a magnified density plot from the highest resolution run c_8. The left plot shows the band-averaged $L_{1,\mathrm{abs}}$ norms with the lines color-keyed to the bands plotted at right. Also shown are solid black lines indicating first and second-order convergence.

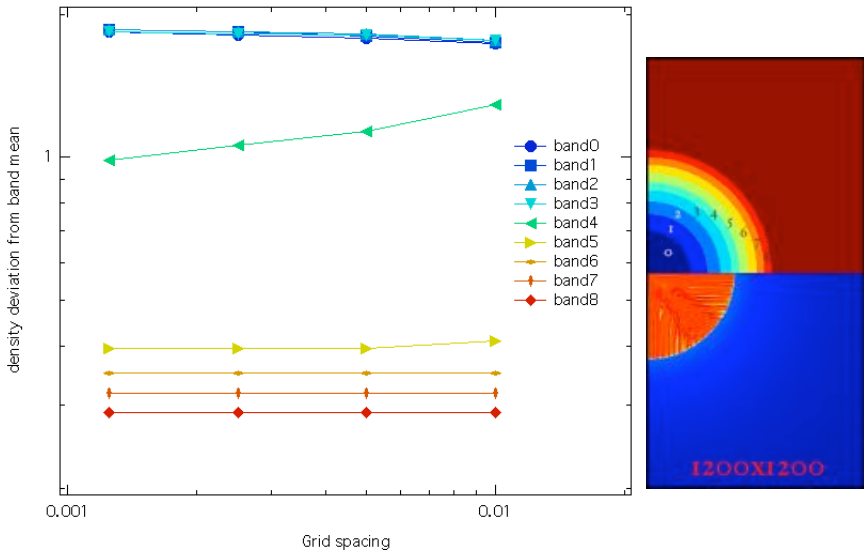

Figure 32. - The asymmetry within each band in the Noh problem, defined as the root- mean-square deviation from the band mean, is plotted as a function of cell size. Only in the shock-straddling band 4 is this asymmetry found to decrease with increasing resolution, and this is solely due to the thinning of the steep shock transition within that band. The band definitions and color coding are the same as for Figure 31.

We define the asymmetry within a band as the root-mean-square deviation from the band mean. Since the bands are narrow, the deviations are mainly due to the herringbone pattern, except where a band straddles the shock (band 4), where the deviations are dominated by the steep shock transition. The asymmetry within the shock (the herringbone pattern, at high-resolution) is found not to diminish with increasing resolution, as shown in Figure 32 and Table 7.

While the character of the asymmetry changes from the broad valley and thin stripes seen in the lowest resolution run in Figure 32 to the fine herringbone seen in the highest resolution run, there is no substantial improvement in symmetry as resolution increases. Within the shock, the asymmetry becomes a little worse with increasing resolution, as shown by the numbers in Table 7. Again recall that the solution within the shock is ρ=64, so the rms deviation from symmetry is 2-3%. The asymmetry numbers are less than or comparable to the band-averaged L1 norms shown in Table 6. Assuming these errors add in quadrature, we have an overall accuracy of order 5% within the shock in the Noh problem, not substantially improving with resolution.

The convergence analysis that Kamm and Kirkpatrick (2004) undertook did not include a study of the adaptive mesh refinement. The analytical procedure for deriving convergence assumes a uniform grid, and so is not (strictly speaking) valid for an adaptive grid. The fundamental assertion of AMR, however, is that if refinement occurs only where it is needed, then convergence on an adaptive grid should be as good as convergence on a uniform grid. Implementations of adaptive grids of course depend on the criteria for deciding where to refine and de-refine (perhaps more important for considerations of efficiency), and the procedures for assigning quantities to the new cells created in refinement or averaging quantities from cells are de-refined.

A series of adaptive grid simulations a_2, a_4, and a_8 were run that had minimum cell sizes the same as the corresponding uniform cell simulation. All the adaptive grid simulations started with the baseline c_1 150×150 cell grid. Table 8 details the properties of all the runs.

Table 8
Adaptive and Uniform mesh simulations in 2D for the Noh problem

Type	Run	dr=dz	sum_cell	sum_cpuhr
Baseline	c_1	0.01	22,500	0.102
Uniform grid	c_2	0.005	90,000	2.48
	c_4	0.0025	360,000	6.71
	c_8	0.00125	1,440,000	68.7
AMR		grid (min)	(at end)	
	a_2	0.005	112,208	17.3
	a_4	0.0025	470,012	53.9
	a_8	0.00125	1,887,408	196

Table 8 shows the high cost of running an adaptive grid on the Noh problem in 2D. The cell counts are considerably

larger than the cell counts for the corresponding uniform grid simulations, indicating that the AMR machinery is refining the grid essentially everywhere in the problem domain. The reported sum_cell count includes the active top-level cells and all the inactive (larger) parent cells. In fact this refinement occurs at the first cycle (cycle zero) of the problem. This is puzzling because there are no gradients in any physical property at the first cycle. Density, pressure, and temperature are all uniform throughout the grid. Speed is also uniform, directed radially inward, but the r and z velocity components separately vary from cell to cell to keep the radial velocity uniform. Mike Clover of SAIC informed one of us (Gisler) that the refinement that occurs at cycle zero does indeed depend on the velocity components separately. We shall see below, however, that the use of AMR in the Noh problem leads to very deleterious consequences regardless of (or in spite of) the initial refinement.

Figure 33 shows the difference plots (L1 norm of density, analogous to Figure 30) for the baseline run c_1 and the three adaptive mesh runs a_2, a_4, and a_8. To the eye, there is comparable improvement going from c_1 to a_2 as there is in going from c_1 to c_2 in Figure 30. But going on to a_4 the grid-aligned features become distinctly more prominent, and they also appear farther from the axes than the comparable features in c_4 of Figure 30. The finest resolution AMR run, a_8, has a surprising and catastrophic anomaly on the axis, which turns out to be a low density bubble straddling the shock. This is probably related to the carbuncle phenomenon that is known to affect some Godunov schemes (Quirk 1994). As pointed out in the discussion accompanying Table 8, in these adaptive grid runs, the RAGE code is adapting everywhere down to the finest allowed cell size on the initial time step, which accounts for the high cell counts and long run times of adaptive mesh runs in the Noh problem. However, if the code kept the finest allowed cell size everywhere, then the results should be identical to the corresponding uniform grid runs. Since they are not, the problem must arise during de-refinement, and indeed we find that the on-axis bubble is seeded at one of the de-refined regions.

Both the herringbone asymmetry problem (in either uniform or adaptive solutions) and the carbuncle anomaly are now being examined by the Code Project A team to identify and eliminate the bugs that are causing these.

Figure 34 shows the convergence plot for the adaptive mesh runs, analogous to Figure 31 for the uniform mesh runs. Clearly (as is already evident to the eye from Figure 33) the adaptive mesh runs are not converging.

We also conducted a study of the effects of changing the input variable `numrho` on the questions of accuracy, asymmetry and convergence. This variable determines the method of interpolation used in the hydrodynamics. The interest in this arose because complex multi-material problems that are run using this code exhibit strikingly different behaviors for different values of `numrho`, and it is therefore of interest to perform verification and validation exercises to determine which value retains the most fidelity to the physics. We find that changing `numrho` over all its possible values has essentially no effect either on accuracy or on asymmetry in the Noh problem, and the convergence behavior is identical for `numrho=1` and `numrho=6`. The default value of `numrho` in the code is 1, which invokes a standard minmod interpolation scheme. The deprecated `numrho=0` is a simple donor-cell method; `numrho=2` and `numrho=3` are the extended and iterated minmod schemes respectively; `numrho=4` is the standard van Leer (van Leer, 1974) interpolation scheme, and `numrho=6` is a modified van Leer method with a different weight parameter. There is no `numrho=5` option. In these runs we used `hydro_option=1`, which makes the interpolation consistent between the derivative finders and the Riemann solver.

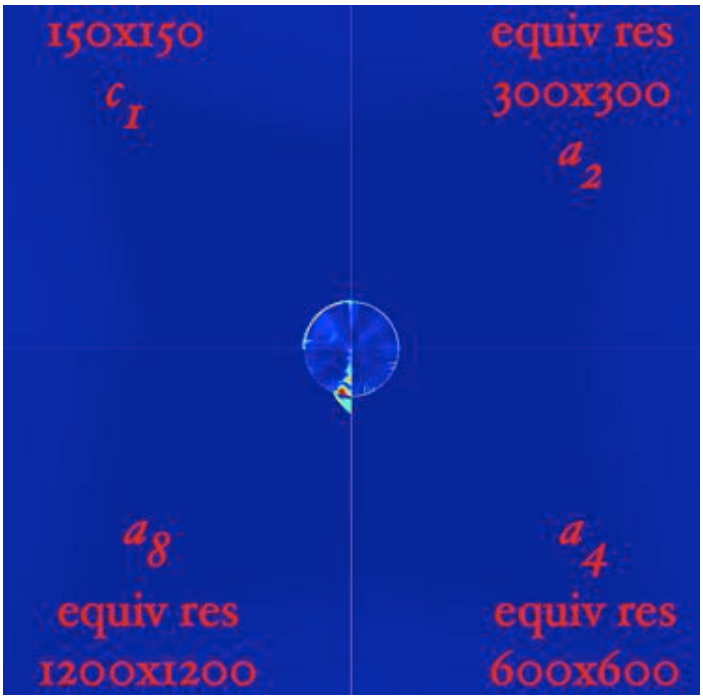

Figure 33. - Plots of the L1 norm for density in the Noh problem from the baseline run c_1 and the three adaptive mesh runs a_2, a_4, and a_8 Each a_i run has the same smallest cell size as the uniform cell size in the c_i run with the same subscript, giving the same equivalent resolution.

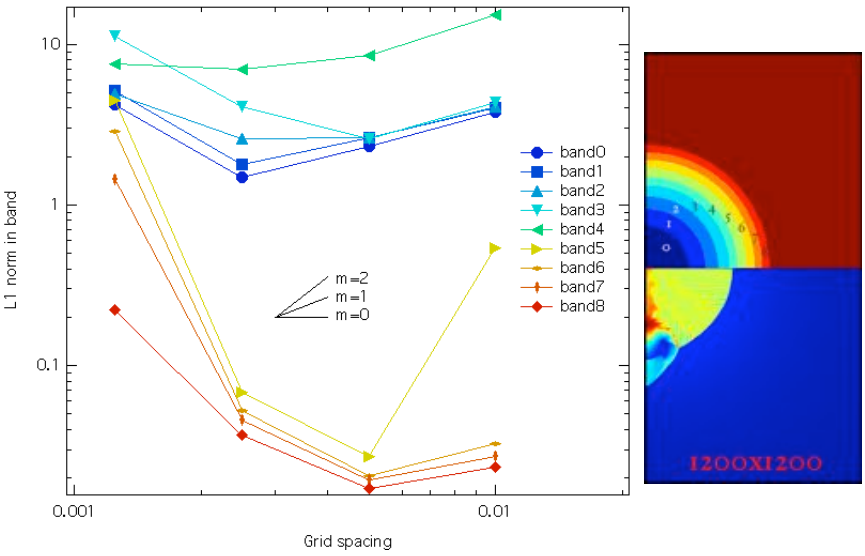

Figure 34. - Left panel is a convergence plot for the adaptive mesh runs, showing $L_{1,\mathrm{abs}}$ norms averaged in the bands at top right. Right panel shows the density the high-resolution run a_8, where the carbuncle on axis is clearly seen. Band definitions and color coding are the same as for Figure 31.

Following the circulation of a draft of this report, the code team worked on addressing the adaptive grid refinement issues encountered in the Noh problem.

We are pleased that the most recent versions of RAGE provide dramatic improvement in both convergence and symmetry for the Noh problem, and somewhat better run times. Figure 35 shows the L1 norm for density with alpha version 20050818.010, the latest official release as of this report. The carbuncle so obvious in a_8 of Figure 33 is clearly absent, although a magnifying glass would show a much smaller version of it in the same position. A convergence plot for these new runs is shown in Figure 36, which is to be compared with Figure 34. The behavior at small cell sizes is much improved, and linear convergence is now obtained for the innermost bands within the shocked region. There is still some divergence outside the shocked region, but the absence of the carbuncle has a noticeable effect on improving the overall divergence.

The improvements to the code in going from 20041126.042 to 20050818.010 involve improvement of the hydrodynamics throughout, and 'closer' examination of the conditions under which refinement and derefinement occur. Mike Gittings made a further improvement to the refinement criteria, introducing a new input variable `rho_eps` which can be used to relax the refinement, specifically addressing the Noh problem. The convergence plot for his release, version 20050809.400, is shown in Figure 36, showing a considerable improvement in the convergence at all cell sizes. The main culprit in the objectionable behavior we found with earlier versions of the code has to do with when and how refinement and derefinement occur. The cell counts in Gittings version are 91%, 88%, and 87% the numbers quoted in Table 8, for runs a_2, a_4, and a_8 respectively, and the run times are correspondingly smaller as well. Thus there is still considerable room for further improvement in comparing the adaptive grid performance numbers to the uniform grid performance.

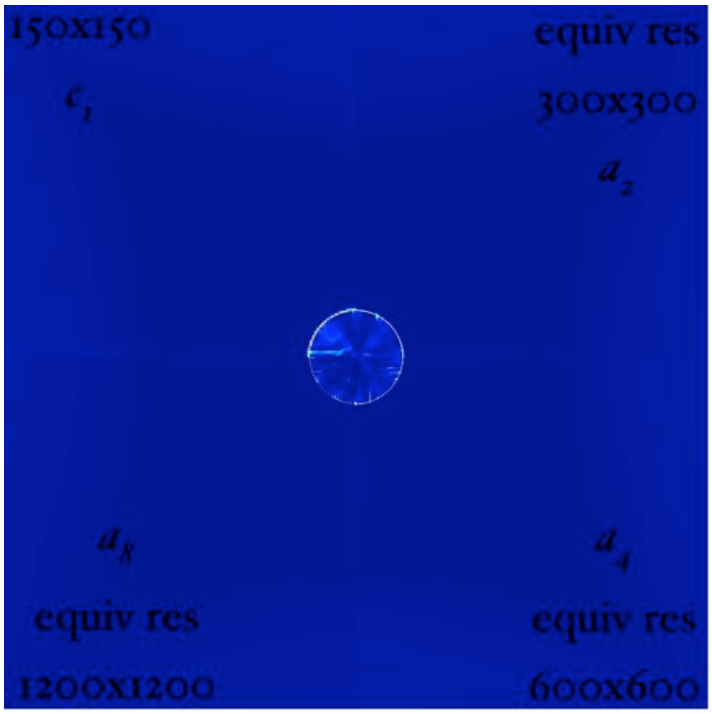

Figure 35. - Plots of the L1 norm for density in the Noh problem using RAGE 20050818.010 from the baseline run c_1 and the three adaptive mesh runs a_2, a_4, and a_8 Each a_i run has the same smallest cell size as the uniform cell size in the c_i run with the same subscript, giving the same equivalent resolution.

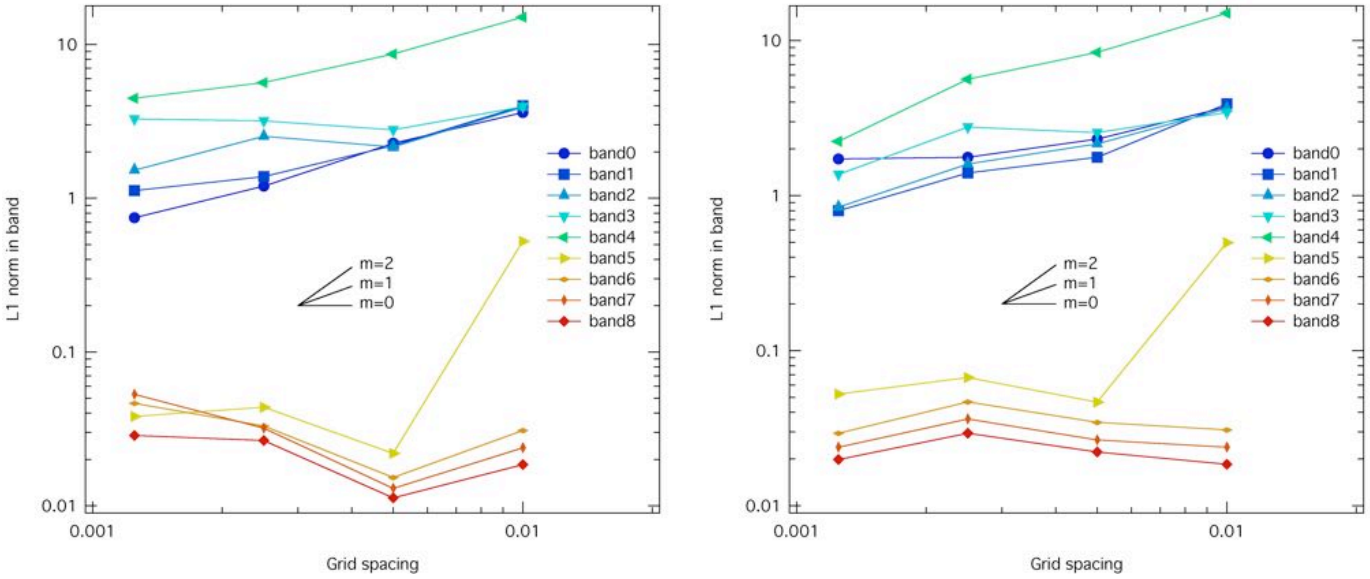

Figure 36. - Left plot shows the L1 norms using RAGE 20050818.010, while the right plot shows the same for the unofficial Gittings RAGE 20050809.400. The bands are defined as in Figure 34.

3.6 The Sedov Problem

A finite amount of energy is released at the origin at an initial time. The problem of finding self-similar, one-dimensional solutions for compressible hydrodynamics was considered by Sedov (1959), Taylor (1950), and von Neumann (1947). Sedov provided the most general closed-form solution, which we employ in the forms considered by Kamm (2000b).

There was perhaps something new to add to this venerable problem. When a known amount of energy is deposited, two energy integrals must be done. One or both of these integrals usually have a singularity at the lower limit of integration. Without specifically addressing the singularity, quadrature routines complain bitterly about inaccurate answers or too many iterations (e.g., Kamm 2000b) The research code provided for the Tri-Lab test suite was no exception. The trick is to note that that these singularities are integrable power-law singularities; hence removable. Doing so makes quadrature evaluations accurate, repeatable, and efficient rapid. John Bolstad at LLNL apparently implemented this improvement in his/their Sedov code (Kamm, private communication), but it doesn't seem to have been implemented in any LANL code. If a goal is to have a `blessed' exact solution code, then the code in Appendix B may be starting point for including this more robust numerical treatment.

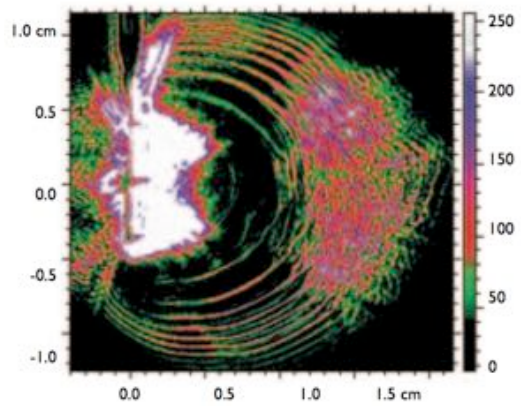

Figure 37. - The stability of Taylor-Sedov blast waves in low-density gases was investigated at the Trident laser facility. Stable and unstable propagation (with respect to the adiabatic index) was observed in experiments. The image shows a multiple-frame shadowgraph of a blast wave in 0.5 Torr of xenon with 66 J at 30-ns intervals.

3.6.1 Analytical Solution

Given the spherically symmetric ($n = 3$) domain characterized by a constant density ρ_0, speed, v_0, pressure P_0, specific internal energy E_0, and adiabatic index γ_0, one deposits an energy $E_{\rm blast}$ at time t=0 s. For the Tri-Lab test problem considered here these values are

$$\rho_0 = 1.0 \qquad v_0 = 0.0 \qquad P_0 = 0.0$$

$$E_0 = 0.0 \qquad \gamma_0 = 1.4 \qquad E_{\rm blast} = 0.851072 \tag{40}$$

The physical solution at time t=1 s is desired.

One begins the solution for the standard (a nonzero solution extends from the shock to the origin, at which the pressure is finite), constant density ($\omega = 0$) case by forming the exponents

$$a_0 = \frac{2}{n-2} \qquad a_2 = \frac{1-\gamma}{2(\gamma-1)+n}$$

$$a_1 = \frac{(n+2)\gamma}{2+n(\gamma-1)}\left[\frac{2n(2-\gamma))}{\gamma(n+2)^2} - a_2\right] \qquad a_3 = \frac{n}{2\gamma-1+n}$$

$$a_4 = \frac{a_1(n+2)}{2-\gamma} \qquad a_5 = \frac{2}{\gamma-2} \ , \tag{41}$$

and the frequent combinations

$$a = \frac{(n+2)(\gamma+1)}{4} \qquad b = \frac{\gamma+1}{\gamma-1}$$

$$c = \frac{(n+2)\gamma}{2} \qquad d = \frac{(n+2)(\gamma+1)}{(n+2)(\gamma+1) - 2(2+n(\gamma-1))}$$

$$e = \frac{2+n(\gamma-1)}{2} \ . \tag{42}$$

The dimensionless shock speed v_0 and the dimensionless post-shock state v_s form the limits of integration

$$v_0 = \frac{2}{(n+2)\gamma} \qquad v_s = \frac{4}{(n+2)(\gamma+1)} \ , \tag{43}$$

for the energy integrals

$$J_1 = \int_{v_0}^{v_s} \mathrm{d}v \ \frac{\gamma+1}{1-\gamma} \ v^2 \ \left[\frac{a_0}{v} + \frac{a_2 c}{cv-1} - \frac{a_1 e}{1-ev}\right]$$

$$\times \left[(av)^{a_0} \ (b\,(cv-1))^{a_2} \ (d\,(1-ev))^{a_1}\right]^{-(n+2)}$$

$$\times (b\,(cv-1))^{a_3} \ (d\,(1-ev))^{a_4} \ \left(b\left(1-\frac{cv}{\gamma}\right)\right)^{a_5} \ . \tag{44}$$

$$J_2 = \int_{v_0}^{v_s} \mathrm{d}v \ -\frac{\gamma+1}{2\gamma} \ v^2 \ \left(\frac{cv-\gamma}{1-cv}\right) \ \left[\frac{a_0}{v} + \frac{a_2 c}{cv-1} - \frac{a_1 e}{1-ev}\right]$$

$$\times \left[(av)^{a_0} \ (b\,(cv-1))^{a_2} \ (d\,(1-ev))^{a_1}\right]^{-(n+2)}$$

$$\times (b\,(cv-1))^{a_3} \ (d\,(1-ev))^{a_4} \ \left(b\left(1-\frac{cv}{\gamma}\right)\right)^{a_5} \ . \tag{45}$$

The dimensionless energy of the shock is then given by

$$\alpha = (n-1)\pi\left(J_1 + \frac{2J_2}{\gamma-1}\right) \qquad E_{\rm dim} = \frac{E_{\rm blast}}{\alpha} \tag{46}$$

The shock position r_2, shock speed u_s, and post-shock values of the material speed u_2, density ρ_2, pressure P_2, specific energy E_2, and the non-relativistic sound speed c_2 follow as

$$r_2 = \left(\frac{E_{\rm dim}}{\rho_0}\right)^{1/(n+2)} t^{2/(n+2)} \qquad u_s = \frac{2}{n+2}\,\frac{r_2}{t} \qquad u_2 = \frac{2u_s}{\gamma+1}$$

$$\rho_2 = \frac{\gamma+1}{\gamma-1}\,\rho_0 \qquad P_2 = \frac{2\rho_0 u_s^2}{\gamma+1} \qquad E_2 = \frac{P_2}{\rho_2(\gamma-1)} \qquad c_2 = \sqrt{\frac{\gamma P_2}{\rho_2}}\ . \tag{47}$$

Next, we locate the similarity value V^* that correlates to $r_{\rm want}$ by seeking the value of zero of the function $f(V^*)$

$$f(V^*) = r_2\,\left[(aV^*)^{-a_0}\ (b\,(cV^*-1))^{-a_2}\ (d\,(1-eV^*))^{-a_1}\right] - r_{\rm want} \tag{48}$$

With this value of V^* compute the four Sedov functions:

$$\lambda = (aV^*)^{-a_0}\ (b\,(cV^*-1))^{-a_2}\ (d\,(1-eV^*))^{-a_1}$$

$$f = a\ v\ \lambda$$

$$g = (b\,(cv-1))^{a_3}\,(d\,(1-ev))^{a_4}\left(b\left(1-\frac{(n+2)v}{2}\right)\right)^{a_5}$$

$$h = (av)^{na_0}\,(d\,(1-ev))^{a_4-2a_1}\left(b\left(1-\frac{(n+2)v}{2}\right)\right)^{1+a_5}\ . \tag{49}$$

If the solution is desired ahead of the blast wave, $r_{\rm want} \geq r_2$, then the physical solution is just that of the ambient material into which the disturbance propagates:

$$\rho = \rho_0 \qquad v = v_0 \qquad E = E_0 \qquad P = P_0 \qquad c = c_0\ , \tag{50}$$

otherwise if the solution is between the origin and the shock front, $0 < r_{\rm want} < r_2$, the physical solution is

$$\rho = \rho_2 g \qquad v = u_2 f \qquad P = P_2 h \qquad E = \frac{P}{\rho(\gamma-1)} \qquad c = \sqrt{\frac{P}{\rho\gamma}} \tag{51}$$

3.6.2 Verification Analysis on 1D Uniform and Adaptive Grids

RAGE 20041126.030, 20041126.032, 20050331.02, 20050331.31, 20050331.41 and 20050818.0000 was used to generate numerical solutions to the 1D Sedov problem on the Linux cluster Lambda. The starting RAGE input deck for this problem was the same one used by Kamm & Kirkpatrick (2004), and modified to make it more amenable to automation and analysis on adaptive meshes (see Appendix A). Grove's AMHCTOOLS (2005) was used to extract the solution data on the native grid from the binary dump files.

Solutions were generated on 1D uniform grids with 120, 240, 480, 960, 1920, and 3840 cells at t=1.0 s. Solutions were also computed on 1D adaptive meshes with an effective resolution (if maximally refined everywhere) of 120, 240, 480, 960, 1920, and 3840. These six effective resolutions correspond to six levels of grid refinement. The 120 cell grid formed the Level 1 adaptive grid, and the `sizemat` parameter was used to increase the potential maximal refinement level. Other mesh refinement variables such as `numfine`, `numlev` `mxcells` and `sizbnd` were left at their default values.

Figure 38 shows solutions to the 1D Sedov problem on uniform and adaptive grids. The top panel compares the analytical and numerical solutions for the density, pressure, temperature and material speed on a 480 cell uniform grid. The only obvious visual disagreement appears to be that the peak pressure generated by the shock is over-estimated by the numerical solution. The middle and bottom panels of Figure 38 show four stills taken from a movie of the Sedov problem with 4 levels of refinement. Circles mark the locations of the adaptive grid points. The total number of grid points and the time is given above each still in the title line. As in several other of the test problems which use the default AMR settings, there is a clumping of the grid points behind the shock which is particularly visible in the material speed (red curves). There is a dense grouping of grid points followed by a less dense grouping followed by another dense grouping. Why grid clumping occurs should be investigated in a future report.

Figure 39 plots the $\mathrm{L1}_{rel}$ error for the density, pressure, and material speed for various grid resolutions. Note the large, persistent errors near the left boundary. These are probably caused by deposition of the initial energy over a finite number of zones in a convergent geometry. There are also oscillatory errors as one approaches the r=1.0 cm position of the shock front. Between the origin and the shock front, the errors generally get smaller as the resolution is increased.

Figure 40 shows the global error norms on uniform grids computed from equations (7). The rate of convergence for all quantities on coarse grids is generally linear, $q \sim 1.0$, and doesn't significantly improve or degrade as resolution increases. Runs with a smaller maximum allowed time-step did not significantly change the convergence rates at the finest grid resolutions. Reasons for not having better than linear convergence behavior is probably due to the persistent errors near the origin and the first-order behavior of hydrocodes like rage near shock fronts. Table 9 details the convergence rate properties of the $L1_{\mathrm{abs}}$ norm on uniform grids.

The ratio of the CPU resources consumed by the uniform and adaptive grid calculations are shown in Figure 41 as a function of refinement level. The improvement in computational efficiency is about a factor ten at four levels of mesh refinement, and increasing. This reduction in CPU resources by using adaptive grids is accompanied by a reduction in error that is no worse than the errors incurred by using a uniform grid. For the Sedov problem, convergence on an adaptive grid is as good (but much less expensive) as convergence on a uniform grid, fulfilling the fundamental assertion of AMR.

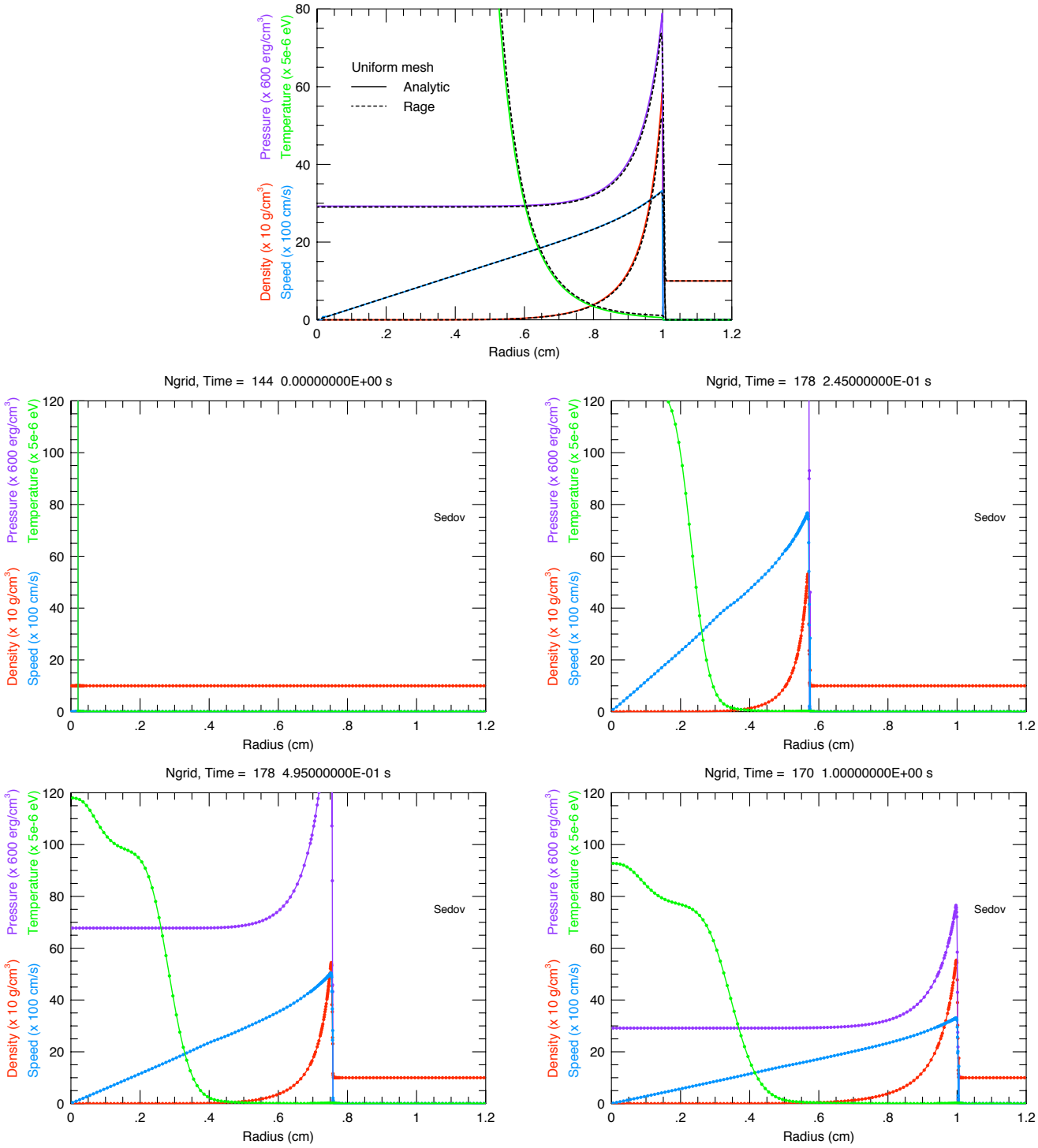

Figure 38. - The top plot shows analytical (solid) and numerical (dashed) solutions for the mass density, material speed, pressure, and temperature at 1.0 s on a 200 cell uniform mesh. The middle and bottom panels show stills from a movie of the Sedov problem with 4 levels of refinement, with circles marking the locations of the adaptive grid points.

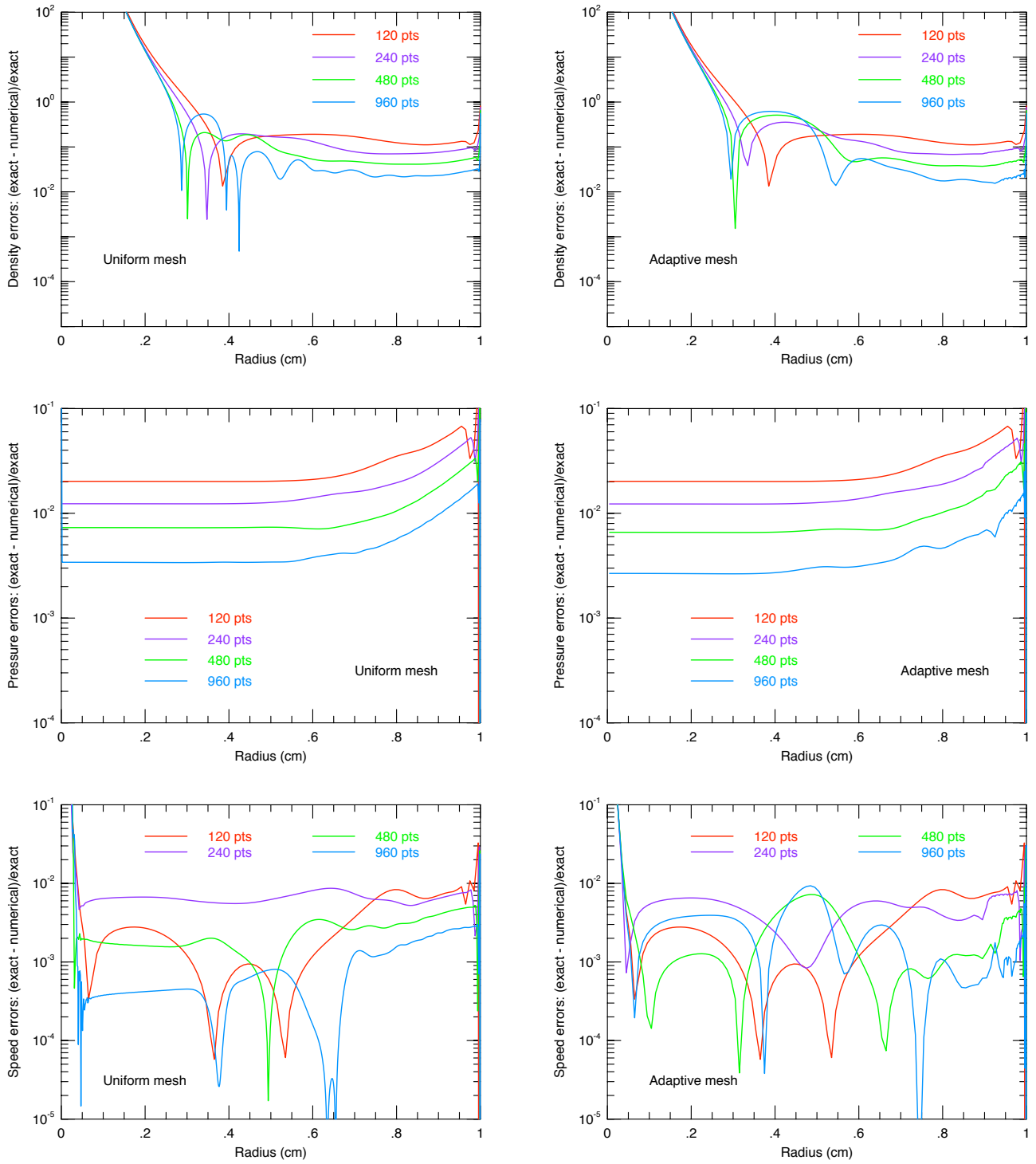

Figure 39. - Absolute value of the point-wise $L_{1,\mathrm{rel}}$ error for the density, pressure, and speed on uniform and adaptive grids at 1.0 s.

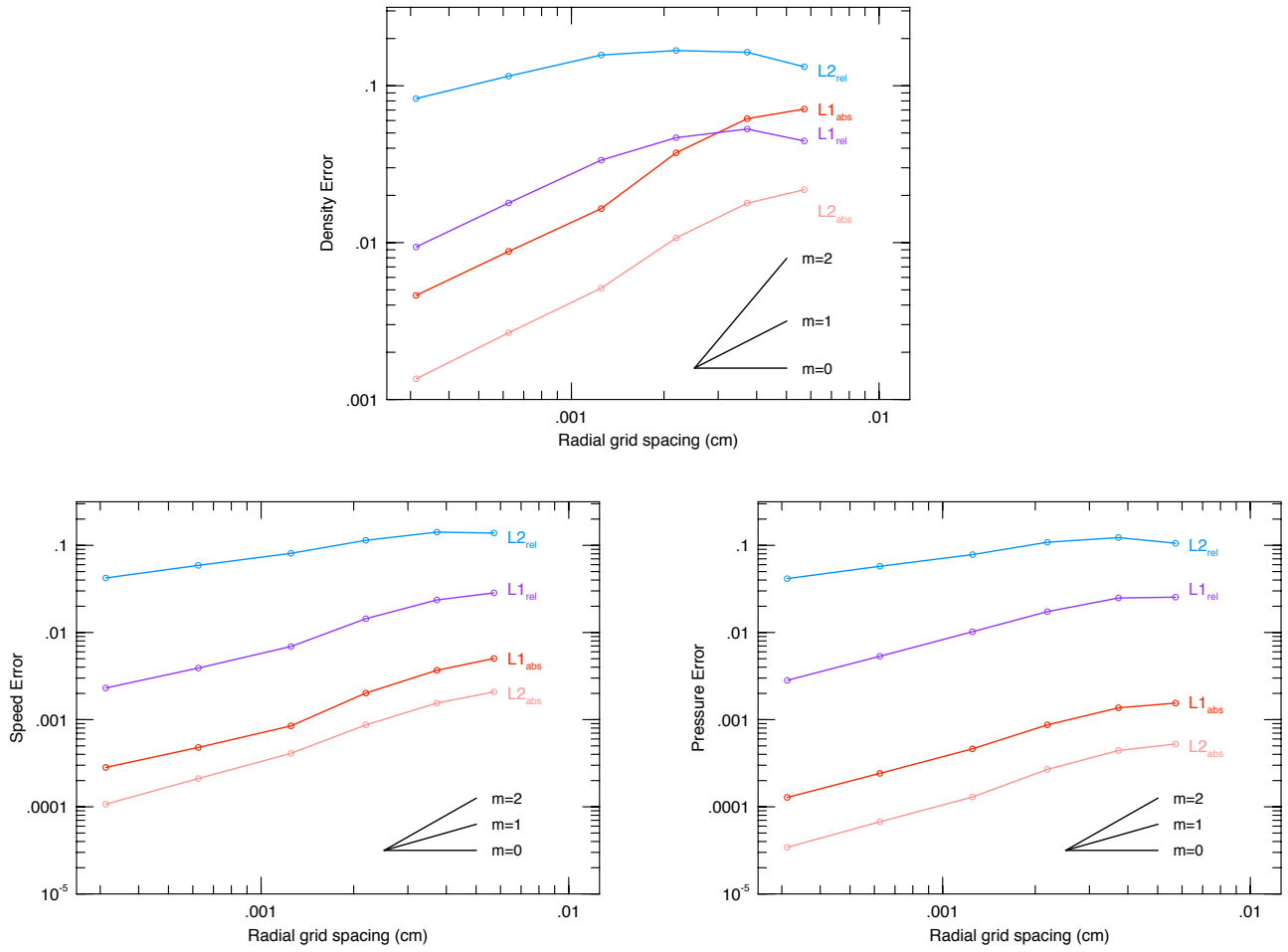

Figure 40. - Global rates of convergence for the mass density, pressure, and material velocity. The black lines show slopes of 0, 1, and 2.

Table 9

Global Convergence coefficients for the Sedov Problem[1]

	ρ		v		P	
# of cells	q	A	q	A	q	A
240	0.33	0.41	0.74	0.25	0.29	7.4E-3
480	0.94	14.3	1.13	2.50	0.85	0.18
960	1.46	372.3	1.55	34.9	1.13	1.08
1920	0.91	8.35	0.82	0.25	0.93	0.28
3840	0.93	10.1	0.77	0.15	0.92	0.25

[1] For the $L1_{abs}$ norm on uniform grids with RAGE 20050818.0000

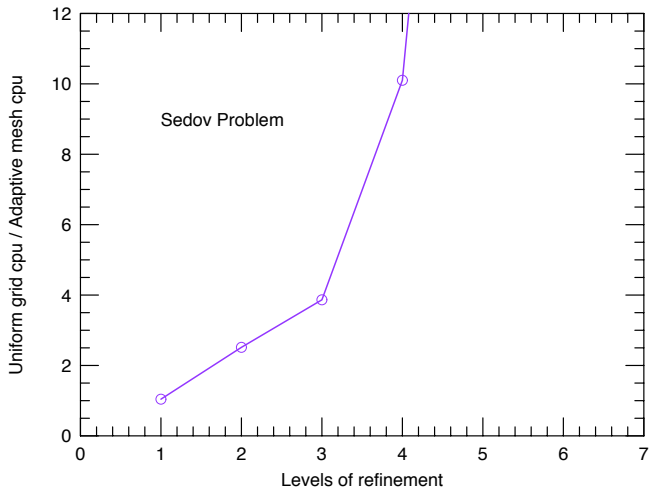

Figure 41. - CPU usage on uniform and adaptive grids for the one-dimensional Sedov problem.

3.6.3 Verification Analysis on 2D Uniform and Adaptive Grids

Identical notation and resolutions is used for the 2D Sedov runs as for the 2D Noh runs described §3.5.3. The Sedov problem domain is [0,1.5]×[0,1.5] in r-z coordinates. The baseline simulation, denoted c_1, is computed on a uniform grid of cell size $\Delta z = \Delta r = 0.01$, so the full grid is 150× 150 cells. A cylinder at the origin of radius and half-height 0.02 (four cells at the lowest resolution) is initialized with a specific internal energy of 16931.53947 erg/gm, giving a total initial energy of 0.851072 erg, per equation (40).

Figure 42 shows density contours of runs c_1, c_2, c_4, and c_8, with the axis of symmetry always vertical. The most noticeable feature of this plot is the narrowing of the shock as the resolution increases. In contrast to the Noh problem, there are no striking asymmetries in the Sedov problem. The $L_{1,\mathrm{abd}}$ norms are shown in Figure 43. It easy to spot an axial problem that arises for the two highest resolution cases (which may again be a manifestation of the carbuncle phenomenon), and a careful eye can spot asymmetric features at the lower resolutions as well. Quantitative analysis shows roughly equal asymmetries at all resolutions, though (as in the 2D Noh problem) they differ in character. Nevertheless, it is clear that asymmetries in the 2D Sedov problem are considerably less concerning than in the 2D Noh problem. This is a natural consequence of the kinematics: expansions smooth out asymmetries while compressions exacerbate them.

Figure 44 divides the problem domain into annular bands that have similar numbers of cells and straddle the shock (see Figure 42). Because not much of interest happens outside the shock, we place band 7 to straddle the shock, and we allow the innermost band, band 0 to be much larger than the other bands. Once again, we discard the outermost band 9 due to boundary effects. The convergence in the Sedov problem is at best first order (except for band 8 which overlaps the leading edge of the shock at low resolution). The shock band itself, band 7, is very closely first order in convergence over all pairs of resolutions, but for inner bands the convergence seems to flatten out at high resolutions, as if no more can be achieved inside with this hydrodynamics. The accuracy is appears quite good, however; the peak value of density at the shock is 5.6.

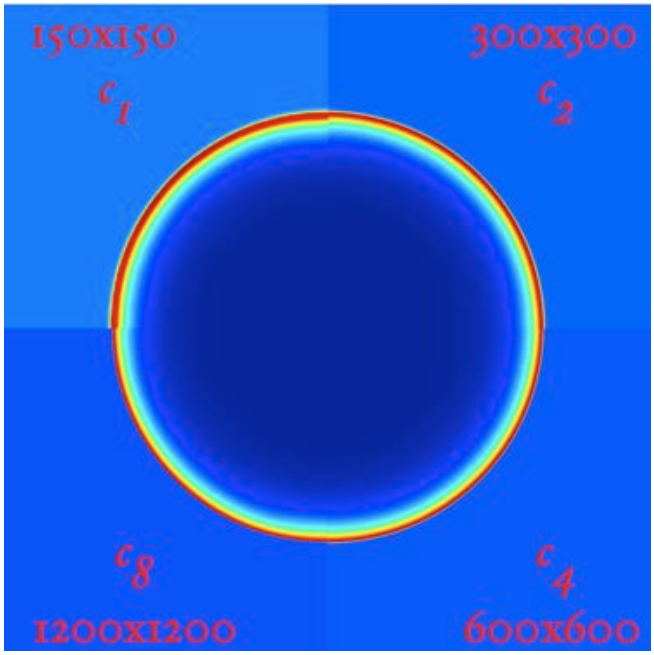

Figure 42. - Density contours from four numerical solutions on uniform grids to the standard Sedov problem at t=1.0 at successively finer refinement levels, going clockwise from top left.

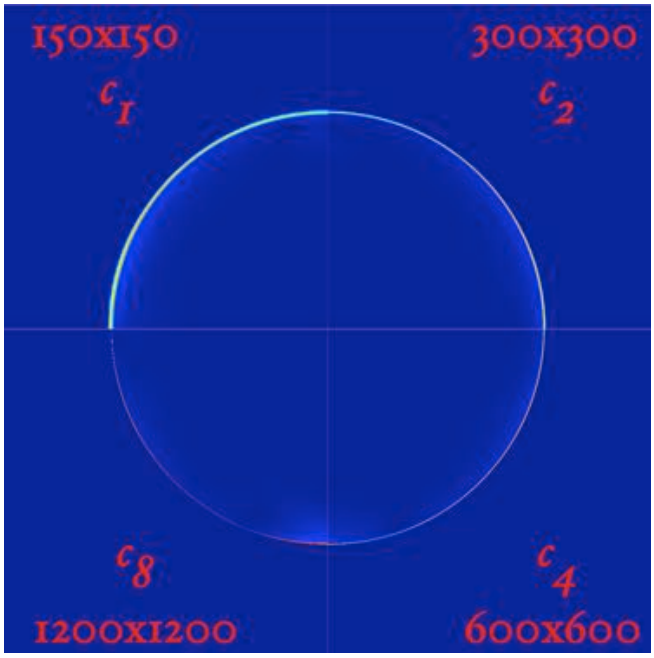

Figure 43. - Plots of the L1 norm of the density from the same four solutions of the Sedov problem shown in Figure 42.

The band-averaged L1 norms themselves are tabulated in Table 10, along with the pairwise convergence indices computed according to equation (7). In Figure 45 we plot the band-wise asymmetry as a function of cell-size, and the corresponding numbers are listed in Table 11. In both accuracy and symmetry, the 2D Sedov problem fares much better than the 2D Noh. But clearly, in neither case does symmetry improve with increasing resolution.

Table 10
Accuracy and convergence for the Sedov problem on 2D uniform grids

	Band averaged L1 norms				Pointwise convergence indices		
band	c_1	c_2	c_4	c_8	q_1	q_2	q_3
dx	0.01	0.005	0.0025	0.00125			
0	0.00717	0.00444	0.00374	0.00440	0.69	0.25	-0.24
1	0.03234	0.02080	0.01809	0.02422	0.64	0.20	-0.42
2	0.05003	0.03103	0.02782	0.03713	0.69	0.16	-0.42
3	0.07597	0.04559	0.04159	0.05244	0.74	0.13	-0.33
4	0.11953	0.07117	0.06328	0.07194	0.75	0.17	-0.19
5	0.20184	0.12967	0.10222	0.10258	0.64	0.34	-0.0
6	0.45072	0.27077	0.18569	0.18656	0.74	0.54	-0.01
7	1.31312	0.85681	0.50227	0.32812	0.62	0.77	0.61
8	0.00387	0.00000	0.00000	0.00000	14.96		

Table 11
Asymmetry in the Sedov problem on 2D uniform grids
rms deviation of band means

band	c_1	c_2	c_4	c_8
dx	0.01	0.005	0.0025	0.00125
0	0.00023	0.00024	0.00024	0.00025
1	0.00459	0.00473	0.00484	0.00494
2	0.00689	0.00706	0.00724	0.00741
3	0.01051	0.01074	0.01103	0.01128
4	0.01607	0.01644	0.01695	0.01734
5	0.02586	0.02649	0.02742	0.02805
6	0.04238	0.04421	0.04594	0.04676
7	0.02852	0.02471	0.02098	0.01842
8	0.01132	0.01128	0.01128	0.01128

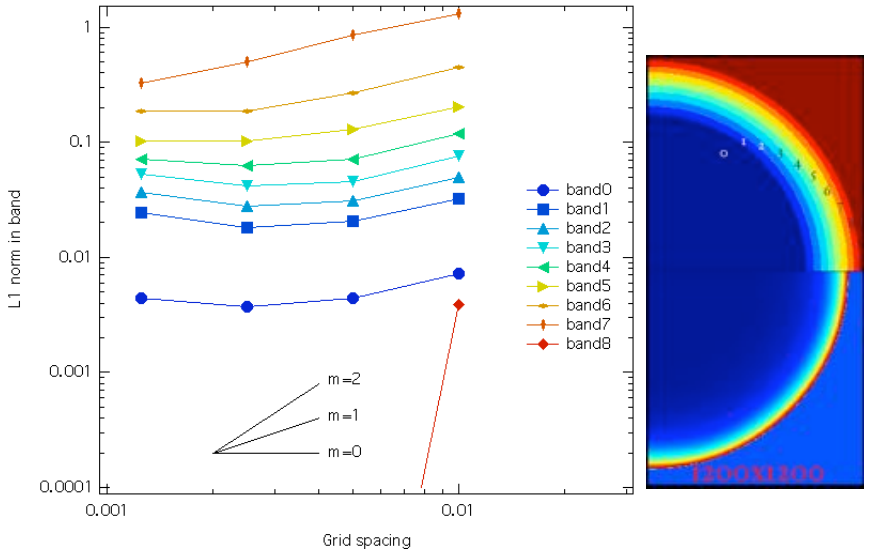

Figure 44. - At right are illustrated the analysis bands 0 through 8 and a magnified density plot from the highest resolution Sedov problem run of Figure 42. At left are the band-averaged L1 norms plotted as a function of cell size dx with the lines color-keyed to the bands plotted at right. Also shown are solid black lines indicating first and second-order convergence.

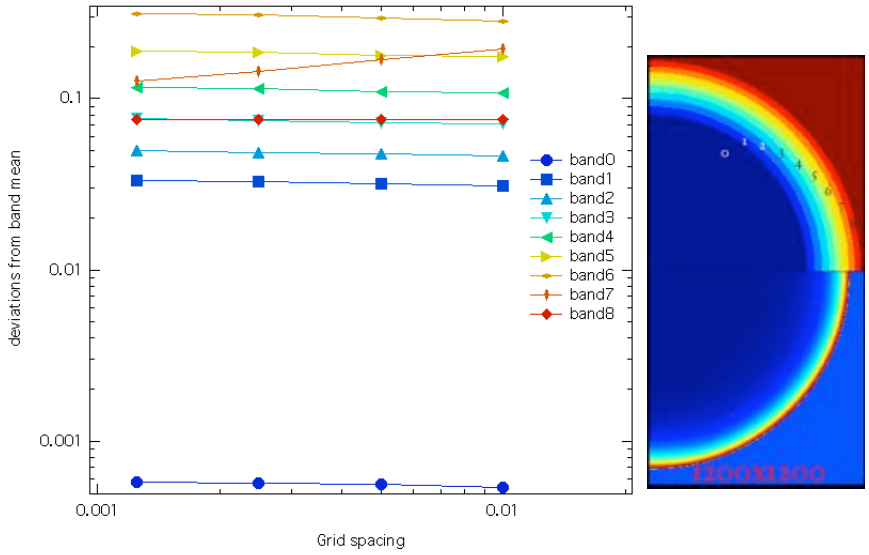

Figure 45. - Asymmetry within each band in the Sedov problem, defined as the root-mean-square deviation from the band mean. Only in the shock-straddling band 7 is this asymmetry found to decrease with increasing resolution, and this is solely due to the thinning of the steep shock transition within that band. The band definitions and color coding are the same as for Figure 44.

In the 2D Sedov problem, the adaptive grid refinement works extremely well, as suggested in Table 14, in concentrating the computing burden where it is most needed, that is to say precisely at the shock, and in saving considerable computational expense. In Figure 46 we show that this is also done without compromising the accuracy and convergence properties. Figures 43 and 46 are virtually identical; the AMR runs for the 2D Sedov problem show no hint of the disastrous anomaly that bedevils the AMR runs for the 2D Noh problem. There is again that axial feature at high resolution, that is present also in the uniform grid runs for this problem. The graphs for the AMR runs that are equivalent to Figures 44 and 45 are so closely identical to them that we'll not bother showing them. We do, however, display the numbers for accuracy and symmetry in the AMR runs in Tables 12 and 13, which may be compared with Tables 10 and 11 for the uniform grid runs.

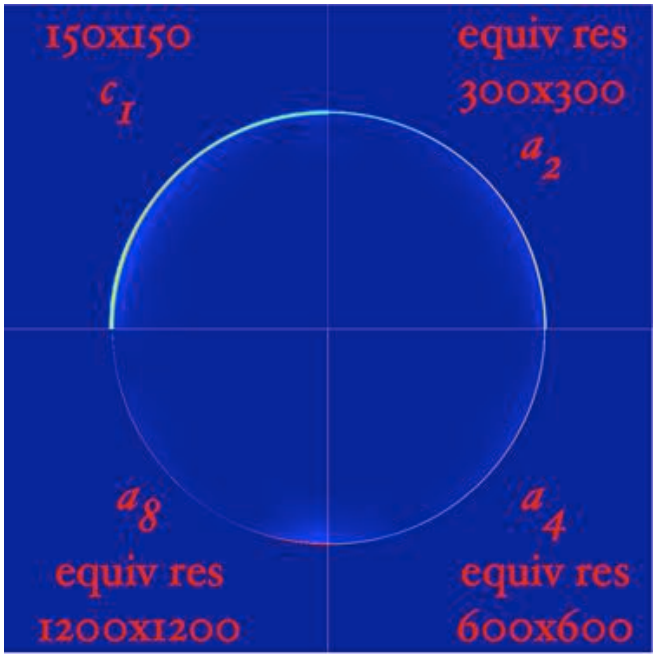

Figure 46. - Plots of the L1 norm for density in the Sedov problem from the baseline run c_1 and the three adaptive mesh runs a_2, a_4, and a_8 Each a run has the same smallest cell size as the uniform cell size in the c run with the same subscript, giving the same equivalent resolution.

Run times and cell counts for the 2D Sedov problem are listed in Table 14. It is immediately evident that AMR does a much better job on the 2D Sedov problem than it does on the 2D Noh problem. Cell counts in the AMR runs increase by a factor $\sqrt{\alpha}$ for a factor α decrease in the cell size, so the advantage over uniform grid runs goes roughly as $\alpha^{3/2}$. Accuracy is not compromised in the AMR runs on the 2D Sedov problem, in contrast to the difficulties encountered in the 2D Noh problem.

Table 12
Accuracy and convergence for the Sedov problem on 2D adaptive grids

	Band averaged L1 norms				Pointwise convergence indices		
band	c_1	a_2	a_4	a_8	q_1	q_2	q_3
dx	0.01	0.005	0.0025	0.00125			
0	0.00717	0.00411	0.00354	0.00429	0.80	0.22	-0.28
1	0.03234	0.02016	0.01754	0.02385	0.68	0.20	-0.44
2	0.05003	0.03030	0.02697	0.03591	0.72	0.17	-0.41
3	0.07597	0.04497	0.03903	0.04919	0.76	0.20	-0.33
4	0.11953	0.07289	0.05762	0.06596	0.71	0.34	-0.19
5	0.20184	0.12489	0.09857	0.10084	0.69	0.34	-0.03
6	0.45072	0.26237	0.17233	0.18914	0.78	0.61	-0.13
7	1.31312	0.83914	0.45389	0.26666	0.65	0.89	0.77
8	0.00387	0.0000	0.0000	0.0000	15.47		

Table 13
Asymmetry in the 2D Sedov problem on adaptive grids
rms deviation of band means

band	c_1	a_2	a_4	a_8
dx	0.01	0.005	0.0025	0.00125
0	0.00054	0.00056	0.00058	0.00059
1	0.03101	0.03212	0.03293	0.03375
2	0.04616	0.04743	0.04882	0.05007
3	0.07080	0.07246	0.07494	0.07689
4	0.10779	0.11016	0.11466	0.11775
5	0.17577	0.18042	0.18754	0.19262
6	0.28401	0.29694	0.31034	0.31616
7	0.19436	0.16737	0.13754	0.11797
8	0.07582	0.07553	0.07553	0.07553

Table 14
Adaptive and Uniform mesh simulations in 2D for the Sedov problem

Type	Run	dr=dz	sum_cell	sum_cpuhr
Baseline	c_1	0.01	22,500	0.685
Uniform grid	c_2	0.005	90,000	99.8
	c_4	0.0025	360,000	251
	c_8	0.00125	1,440,000	880
AMR		grid (min)	(at end)	
	a_2	0.005	30,984	10.6
	a_4	0.0025	39,444	13.4
	a_8	0.00125	60,104	25.8

Figure 47 shows the Sedov problem on an adaptive using an unofficial code release from Mike Gittings, version 20050809.400. This plot is indistinguishable by eye from the plot shown in Figure 42, and also indistinguishable from the same plot for the latest official release 20050818.010. The numbers in the band-averaged L1 norms differ very slightly between these two versions and as against the earlier version, at a fractional level of ~0.0001 or better, and the cell counts in both the Gittings and official Crestone versions are reduced by about 10% from the counts in the earlier version.

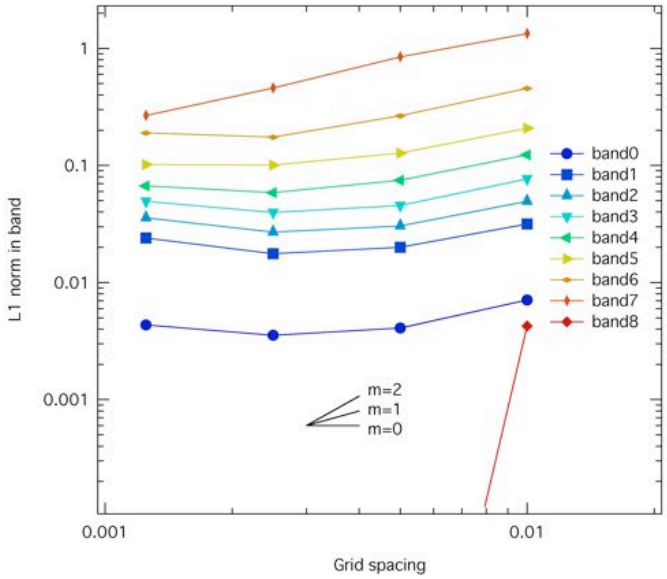

Figure 47. - Convergence plot for the Sedov problem using the unofficial Gittings RAGE version 20050809.400 showing the L1 norms in the bands defined as before.

4. Future Directions

Development and implementation of next-generation AMR packages would appear to represent a tremendous growth opportunity. It is clear that prodigious resources has gone into developing physics modules, but not the grid technology on which that physics is computed. This is reflected in the appearance of there not being a single group or team within the Lab that is examining existing AMR technology (Chombo, PARAMESH, SAMARI, HAMR, DAGH, SARA, SUUMA3D to name a few) with an eye towards the best parts of these technologies into practice at LANL.

New test problems that exercise multi-material and/or multi-temperature capabilities should be developed. In parallel, calculation verification procedures for complex physics problems that admit no exact solution must be encouraged to advance. The standard approach to conducting verification analysis where no exact solution exists presents two significant limitations. Computational solutions that converge by oscillation are not calculable and the technique is limited to a simple error ansatz. An improvement to the current method is needed. Perhaps the approach being developed by Smitherman and Kamm (2005) would suffice as a first step. In their method, the absolute value of the pointwise error is calculated, allowing for local oscillatory convergence. The equations are then solved using Newton's method for the convergence constants, discretization errors, and an estimated exact solution simultaneously. This procedure allows for a more complex ansatz if desired.

Automated verification analysis should be completed for two and three dimensional versions of all the existing Tri-Lab test problems to assess how well RAGE retains fidelity to the underlying physics when motions and gradients are not grid-aligned.

On September 6, 2005 the adaptive grid results of this report were presented to a well attended V&V meeting called by Kim New. Two high-level topics surfaced; verification methodology on adaptive grids, relevance of the Noh problem, and relevance of the Tri-Lab test suite problems. Some of these topics should set directions for future verification efforts.

The present verification analysis starts with a refinement level 1 base grid and increases number of levels of refinement. At each level of refinement, at each decrease in cell size, the accuracy of the solution and convergence coefficients are determined. Mike Gittings vocalized the opinion that this accuracy and convergence methodology was incorrect. He advocated a 'moving window' approach, where the number of refinement levels is held constant. That is, one starts with a level 1 grid and drills down (say) four levels of refinement. Then go the level 2 grid and drill down four levels of refinement, and so on. Several people (Bill Rider, Frank Timmes) vocalized the opinion that this alternative methodology isn't how AMR is traditionally thought about, advertised, or used. Objections were voiced that the Noh problem is highly idealized and pathological, presenting a hydrocode with a situation highly unlikely to present itself in real-life simulations. If tinkering with the refinement criteria in order to improve the performance of the code on the Noh problem results in deleterious performance on other problems, then the desirability of this tinkering becomes dubious at best. On the other hand, unlike most AMR packages, there is very little user control over the refinement criteria in RAGE. It needn't be an 'either or' situtation, the code should be flexible enough to handle a wide range of problems.

Performance on realistic applications might be robust should multi-material or multi-temperature physics in complex geometries unambiguously mitigate any difficulties exposed by simple test problems on adaptive grids in basic geometries. Questions of test problem relevance could be addressed by scaling simpler problems upwards, and

by having complicated problems stress test adaptive grid technologies and/or distilling the essential computational physics into simpler forms.

5. Acknowledgments

This work was undertaken at the suggestion of Jerry Brock and Kim New, whose interest and support is gratefully acknowledged. The careful and reasoned technical input of Jim Kamm, Chris Fryer, Alan Calder (University of Chicago ASC center), Ron Kirkpatrick, Kunnegunda Belle, Mike Gittings, Mike CLover, Bruce Fryxell (University of Georgia), and Bill Rider was essential in working through the details of this investigation. This work was performed by Los Alamos National Laboratory under contract W-7405-ENG-48 by the University of California for the U.S. Department of Energy.

6. References

Bethe, H.A., Fuchs, K., Hirschfelder, J.D., Magee, J.L., Peierls, R.E., & von Neumann, J., ``Blast Wave", LA-2000, August 1947.

Chapelle, D., www-rocq.inria.fr/modulef/Doc/GB/Guide6-14/node21.html, 1999

Coggeshall, S.V., Phys. Fluids A, 3, 5, 1991

Fickett, W., and Davis, W.C., Detonation, (UC Berkeley, 1979)

Gisler, G., ``Two-dimensional convergence study of the Noh and Sedov problems with RAGE: Uniform and Adaptive Grids", LA-UR-05-ZZZZ, 2005

Grove, J., AMHCTOOLS, LA-CC-05-052, 2005

Hrbek, G.M., Timmes, F.X., Brock, J.S., Kamm, J.R., ``The Pinocchio Project: An Automated Scripting Tool For Performing Verification Analysis On Physical Simulation Codes Phase 1 Report", LA-UR-05-3471, 2005

Kamm, J.R., ``Investigation of the Reinicke & Meyer-ter-Vehn Equations: I. The Strong Conduction Case", LA-UR-00-4304, 2000a

Kamm, J.R., ``Evaluation of the Sedov-von Neumann-Taylor Blast Wave Solution", LA-UR-00-6055, 2000b

Kamm, J.R., and Kirkpatrick, R., Verification analyses of code project A, LA-CP 04-04361, 2004

Kirkpatrick, R., Wingate, C., Kamm, J., HE Burn Test Problem, X-3-19U, 2004

Landau, L.D., Lifshitz, E.M., Fluid Mechanics, Pergamon Press, Oxford, 1959.

Noh, W.F., ``Errors for calculations of strong shocks using an artificial viscosity and an artificial heat flux", J. Comp. Phys. 72, 78-120, 1987.

Quirk, J.J., ``A contribution to the great Riemann solver debate", Int. J. Num. Meth. Fluids, 18, 555-574, 1994.

Sedov, L.I., ``Similarity and Dimensional Methods in Mechanics", Academic Press, New York, 1959.

Su, B., and Olson, G.L., J. Quant. Spectro. Radiat. Transfer, 56, 3, 1996

Taylor, G.I., ``The formation of a blast wave by a very intense explosion", Proc. Roy. Soc. London A 201, pp. 159-174 (1950).

Timmes, F.X., ``Progress Report On Three Tri-Lab Verification Test Suite Problems", LA-UR-05-2689, 2005

Timmes, F.X., ``Progress Report On Three (More) Tri-Lab Verification Test Suite Problems", LA-UR-05-5036, 2005

Van Leer, B., ``Towards the ultimate conservative difference scheme II. Monotonicity and conservation combined in a second order scheme", J. Comput. Phys., 14, 361-370 (1974).

7. Appendix A - Input Decks

7.1 For the Su & Olson Problem

```
! Su-Olson Verification Test Problem
! Changed BC's on 10/06/2003:
! Due to discussions with Tom Betlach a temp BC with 1) aTr**4=4Finc/c,
! 2) very opt thin 1st zone, 3) Milne = .true.  are now being used.

pname = "suo_100pt" !  problem name

! MESH SETUP
! 20 cm thick 1-D slab
! RECOMMENDATION: Do not use the mesh variables numrho, numfine, numlev,
! smallke, and mxcells.  Rather use sizemat.

imxset = 100
dxset = 0.2

! CALCULATION CONTROL

tmax = 1.0e-9 !  Ending simulation time in sec
dtedt(1) = 1.0e-9
dodmpxdt = .true.  !  turns on dt adjustment to get dumps at exact dedt time
tedit = 1.0e-9 !  time frequency of binary dump files

kread = -1 !  kread < 0 is new problem; kread>=0 = restart cycle
uselast = .true.  !  if .true.  and kread<0, "pname-lastdump" = restart file
ncmax = 400000 !  Max # of cycles
dtnext = 1.0e-16 !  initial time step
dtmax = 1.0e-10 !  maximum dt allowed
dtpct = 0.1 !  adjusts init.  time step only
de_tevpct =.001 !  max change in tev due to radiation
de_tevmin = 12.5 !  tev floor, below which de_tevpct is ignored

ncedit = 0 !  disable cycle frequency for binary dumps
modcyc = 10000 !  frequency of status edits
shortmodcyc = 200 !  frequency of short edits
ndtedt = 0 !  number of simulation times for std edits (default=0)

! PHYSICS MODULES

dohydro = .false.  !  turn off hydro for this Marshak problem (default=.true.)
hydro_version = 2 !
numrho = 1 !
doheat = .false.  !  turn off heat conduction
dorad = .true.  !  turn on radiation

! RADIATION
```

```
onetemp = .false. ! 2-T non-equilibrium diffusion
fluxlim = .false. ! switch for radiation fluxlimiter

! MATERIALS

nummat = 1 ! number of materials

! Use sizemat to control mesh refinement. Note that this has to be
! coordinated with the dxset parameter above.

sizemat(1) = 0.2 ! 1 level of refinement for material 1
!sizemat(1) = 0.1 ! 2 levels, effective 200 points
!sizemat(1) = 0.05 ! 3 levels, effective 400 points
!sizemat(1) = 0.025 ! 4 levels, effective 800 points
!sizemat(1) = 0.0125 ! 5 levels, effective 1600 points
!sizemat(1) = 0.00625 ! 6 levels, effective 3200 points
!sizemat(1) = 0.003125 ! 7 levels, effective 6400 points

! EOS and OPACITY

keos = -3 ! 0 = ideal gas; 1 = sesame; 2 = N/A; 3 = new TEOS files;
! keos < 0 for special analytic EOS
! for the su-olson problem, use keos = -3 for e = aT**4

matdef(1,1) = 0.0 !
matdef(61,1) = 0 ! power law opacity: kappa = coef*(tev/tevz)**power
matdef(62,1) = 1. ! krmax
matdef(63,1) = 0.0 ! power
matdef(64,1) = 1.0 ! coef
matdef(65,1) = 1.0 ! tevz
matdef(66,1) = 0.000001 ! krscat -- for verification test problem
! = 0.201 for Thompson scattering
! krtot = krscat + kra

! REGIONS
numreg = 1
matreg(1) = 1
rhoreg(1) = 1.0 ! density gm/cm**3
tevreg(1) = 1.0e-1 ! T eV

! CONSTANT FLUX BC -- implemented via Tr BC + Milne condition. =====
! Tom Betlach (SAIC) recommended using the aTr**4 = Finc w/ Milne
! & thin 1st zone to provide the desired BC needed
! for the Su-Olson problem.

nomilne = .false. ! default = .false.
milne_option = 1 ! sets milne BC to true milne (0=Dirchlett, 2=Spillman)
tevbcl = 1000. ! Tr = 1 keV, (not = 4th root of c/4a )
```

7.2 For the Coggeshall #8 Problem

```
! 1-D Coggeshall-8 problem
pname = "cog_100pt"
test_pname = "r008_3v" ! super important to set this name for proper initialization

! CONTROL
time = 1.0e-8 ! starting time; super important for proper initialization
tmax = 2.0e-8 ! Ending simulation time in sec
dtedt(1) = 2.0e-8
dodmpxdt = .true. ! turns on dt adjustment to get dumps at exact dedt time
tedit = 2.0e-8 ! time frequency of binary dump files

dtnext = 1.0e-16 ! initial time step
dtmax = 1.0e-10 ! maximum dt allowed
!dtforce = 1.0e-12

kread = -1
uselast = .true.
ncmax = 400000
ncedit = 0
modcyc = 0
shortmodcyc = 20

dohydro = .true.
dorad = .false.
doheat = .false.

! GRID
! RECOMMENDATION: Do not use the mesh variables numrho, numfine, numlev,
! smallke, and mxcells. Rather use sizemat.
imxset = 100
dxset = 0.02

! SPHERE
cylin = .false.
sphere = .true.

! MATERIALS
keos = 0
nummat = 1

! Use sizemat to control mesh refinement. Note that this has to be
! coordinated with the dxset parameter above.

sizemat(1) = 0.02 ! 1 level of refinement for material 1
!sizemat(1) = 0.01 ! 2 levels, effective 200 points
!sizemat(1) = 0.005 ! 3 levels, effective 400 points
!sizemat(1) = 0.0025 ! 4 levels, effective 800 points
!sizemat(1) = 0.00125 ! 5 levels, effective 1600 points
!sizemat(1) = 0.000625 ! 6 levels, effective 3200 points
```

```
!sizemat(1) = 0.0003125 ! 7 levels, effective 6400 points

matdef(1,1) = 0.0
matdef(16,1) = 0.6666666666
matdef(30,1) = 1.0e12 ! specific heat 1.e12 erg/g/ev = 0.1 jk/g/kev

! Opacity
matdef(61,1) = 0 ! Analytic Opacity:
matdef(62,1) = 1.0e20 ! krmax
matdef(63,1) = +2.0 ! Power: (tevz/te)**powt
matdef(64,1) = 5.485 ! Coef: 4*arad*tevz**3/3/coef = thermcoeff
matdef(65,1) = 1000.0 ! tevz
matdef(66,1) = 0.0 ! krscat
matdef(67,1) = 1.000 ! kpscale
matdef(68,1) = 0.000 ! rho**powd

! Thermal Conductivity
matdef(81,1) = 0
matdef(82,1) = 1.e21 ! Thermcoef 1.e21 erg/ev/cm/s = 1.0 jk/kev/cm/sh
matdef(83,1) = 5.0 ! powt: (te/tevz)**powt
matdef(84,1) = 1000. ! tevz
matdef(85,1) = -1.0 ! powd: rho**powd

! REGIONS
numreg = 2

matreg(1) = 1
rhoreg(1) = 1.0
siereg(1) = 1.0e12

matreg(2) = 1
rhoreg(2) = 3.0
tevreg(2) = 100.0
xdreg(2) = 1.0
xlreg(2) = 0.
xrreg(2) = 2.0

hydro_version = 2
numrho = 2

freeze_num = 1
freeze_x_lo(1) = 1.98
freeze_x_hi(1) = 2.00

numfbc = 12
timefbc(1) =1.0e-8,1.1e-8,1.2e-8,1.3e-8,1.4e-8,1.5e-8,1.6e-8,1.7e-8,1.8e-8,1.9e-8,2.0e-8,2.1e-8
flxtbcl(1)=-1.4286e17,-.66644e17,-.33224e17,-.175128e17,-.96801e16,-.55741e16,
-.33262e16,-.20479e16,-.12963e16,-.8411e15,-.558e15,-.377e15
```

7.3 For the Mader Problem

```
! Mader tri-lab verification problem

pname = "mad_100pt"

! grid
! do not use the mesh variables numrho, numfine, numlev,
! smallke, and mxcells.  rather use sizemat.
imxset = 102
dxset = 0.05

! ----- CONTROL
tmax = 6.25e-6 ! Ending simulation time in sec
dtedt(1) = 6.25e-6
dodmpxdt = .true. ! turns on dt adjustment to get dumps at exact dedt time
tedit = 6.25e-6 ! time frequency of binary dump files

dtnext = 1.0e-16 ! initial time step
dtmax = 1.0e-5 ! maximum dt allowed
!dtforce = 1.0e-12

kread = -1 ! kread < 0 is new problem; kread>=0 = restart cycle
uselast = .true. ! if .true. and kread<0, "pname-lastdump" = restart file
ncmax = 500000 ! Max # of cycles
ncedit = 0 ! disable cycle frequency for binary dumps
modcyc = 0 ! frequency of status edits
shortmodcyc = 20 ! frequency of short edits

hydro_version = 2
!numrho = 4

! ----- MATERIALS
eosfile = 'val.teos'

keos = 3
nummat = 5

matdef(1,1) = 5030 ! sesame air
matdef(1,2) = 3719 ! sesame Al
matdef(1,3) = 152777 ! Hom solid
matdef(1,4) = 162777 ! GAMMA LAW Validation HE
matdef(1,5) = 3719 ! sesame Al Spall


! Use sizemat to control mesh refinement.  Note that this has to be
! coordinated with the dxset parameter above.

sizemat(1) = 0.05 ! 1 levels, effective 100 points
sizemat(2) = 0.05 !
sizemat(3) = 0.05 !
```

```
sizemat(4) = 0.05 !
sizemat(5) = 0.05 !

!sizemat(1) = 0.025 !  2 levels, effective 200 points
!sizemat(2) = 0.025 !
!sizemat(3) = 0.025 !
!sizemat(4) = 0.025 !
!sizemat(5) = 0.025 !

!sizemat(1) = 0.0125 !  3 levels, effective 400 points
!sizemat(2) = 0.0125 !
!sizemat(3) = 0.0125 !
!sizemat(4) = 0.0125 !
!sizemat(5) = 0.0125 !

!sizemat(1) = 0.00625 !  4 levels, effective 800 points
!sizemat(2) = 0.00625 !
!sizemat(3) = 0.00625 !
!sizemat(4) = 0.00625 !
!sizemat(5) = 0.00625 !

!sizemat(1) = 0.003125 !  5 levels, effective 1600 points
!sizemat(2) = 0.003125 !
!sizemat(3) = 0.003125 !
!sizemat(4) = 0.003125 !
!sizemat(5) = 0.003125 !

!sizemat(1) = 0.0015625 !  6 levels, effective 3200 points
!sizemat(2) = 0.0015625 !
!sizemat(3) = 0.0015625 !
!sizemat(4) = 0.0015625 !
!sizemat(5) = 0.0015625 !

!  ----- REGIONS
numreg = 3

matreg(1) = 1 !  air
prsreg(1) = 1.0e6
tevreg(1) = 0.025

matreg(2) = 3 !  solid VHE
xlreg(2) = 0.0000
xrreg(2) = 5.00000
prsreg(2) = 1.0e6
tevreg(2) = 0.025

matreg(3) = 4 !  Gamma Law VHEB
xlreg(3) = 5.00000
xrreg(3) = 5.10000
rhoreg(3) = 2.500
tevreg(3) = 0.2
```

```
! ----- HE setup
he_size = 0.025
he_dtpct = 0.4
he_number = 1

! ----- VHE
he_unreacted(1) = 3 ! VHE
he_reacted(1) = 4 ! VHE
he_rate_size(1) = 0.40
he_model(1) = 2 ! Forest Fire
he_energy(1) = 0.0
he_pcrush(1) = 10.0e9 ! Multiple Shock Forest Fire
he_detvel(1) = 8.000e5
he_rhoz(1) = 1.875
he_pmin(1) = 10.0e9 ! Minimum Forest Fire Pressure
he_pcj(1) = 300.0e9 !

! GAMMA LAW VHE Forest Fire
he_num_coef(1) = 7
he_constants(1,1) = -7.4335806250e05,
7.5179600000E+05,-3.0147946875E+05,6.15952500000E+04,
-6.89248339844E+03,4.5775354003E+02,-1.80467948914E+01
he_rate_size(1) = 0.40
```

7.4 For the Reinicke & Meyer-ter-Vehn Problem

```
! Reinicke Meyer-ter-Vehn Strong Heat Wave, 1D Spherical

pname = "rmtv-100pt"

! CONTROL
tmax = 0.051251245293611e-8
dtedt(1) = 0.051251245293611e-8
dodmpxdt = .true. ! turns on dt adjustment to get dumps at exact dedt time
tedit = 0.051251245293611e-8
dtnext = 1.0e-16 ! initial time step
dtmax = 1.0e-10 ! maximum dt allowed
!dtforce = 1.0e-12
!dtname = 0.051251245293611e-8

kread = -1
uselast = .true.
ncmax = 500000
ncedit = 0 ! disable cycle frequency for binary dumps
modcyc = 0 ! frequency of status edits
shortmodcyc = 100 ! frequency of short edits
dohydro = .true.
doheat = .true.
dorad = .false.

onetemp = .true.
fluxlim = .false.


! GRID
! RECOMMENDATION: Do not use the mesh variables numrho, numfine, numlev,
! smallke, and mxcells. Rather use sizemat.
imxset = 100
dxset = 0.01

! SPHERE
cylin = .false.
sphere = .true. ! Sphere only in 1-D calculations:

! MATERIALS
nummat = 1

! Use sizemat to control mesh refinement. Note that this has to be
! coordinated with the dxset parameter above.

sizemat(1) = 0.01 ! 1 level of refinement for material 1
!sizemat(1) = 0.005 ! 2 levels, effective 200 points
!sizemat(1) = 0.0025 ! 3 levels, effective 400 points
!sizemat(1) = 0.00125 ! 4 levels, effective 800 points
!sizemat(1) = 0.000625 ! 5 levels, effective 1600 points
```

```
!sizemat(1) = 0.0003125 !  6 levels, effective 3200 points


! EOS
keos = 0
matdef(16,1) = 0.25 !  gamma-1 ==> gamma = 5/4
!  matdef(30,1) = 4.6418e+4 !  Cv 11604.5 / ( gamma - 1 )
matdef(30,1) = 4.0e13 !  Cv 1.e13 ==> 1 jk/kev/g

! Opacity
! The analytic models are structured such that Rosseland mean
! absorption coefficient (units:  cm**2/g) for this material,
! integrated over all frequencies, is equal to
!
! coef * (tevz/tev_use)**powt * (frac_mass/frac_vol)**powd
! with
! kramax = matdef(62,m) !  recommended default = 1.0e30
! powt = matdef(63,m) !  recommended default = 3
! coef = matdef(64,m) !  recommended default = 0.0
! tevz = matdef(65,m) !  recommended default = 1000.0
! krscat = matdef(66,m) !  recommended default = 0.2
! kpscale = matdef(67,m)
! powd = matdef(68,m) !  recommended default = 0
! model = matdef(69,m) !  recommended default = 1
! hnu_edge = matdef(70,m) !  recommended default = 0.0
! pct_jump = matdef(71,m) !  recommended default = 0.0

matdef(61,1) = 0 !  use analytic opacity
matdef(62,1) = 1.0e10 !  krmax:
matdef(63,1) = +3.5 !  power-in-temp-dependence:  "powt" in (tevz/te)**powt
matdef(64,1) = 5.48806 !  overall coefficient "coef"
matdef(65,1) = 1000.0 !  reference temperature "tevz"
matdef(66,1) = 0.001 !  ?scattering factor?  "krscat"
matdef(67,1) = 1.000 !  kpscale:  opacity multiplier
matdef(68,1) = 1.000 !  power-in-density-dependence:  "powd" in (frac_rho)**powd

! Thermal Conductivity
! The analytic thermal conductivity model uses the following
! formula for the thermal conductivity:
!
! thermcoef * rho**powd * (te/tevz)**powt
! with
! thermcoef = matdef(82,nm)
! powt = matdef(83,nm)
! tevz = matdef(84,nm)
! powd = matdef(85,nm)
!
matdef(81,1) = 0 !  use thermal conductivity
matdef(82,1) = 1.e21 !  thermcoef 1.e21 erg/ev/cm/s = 1.0 jk/kev/cm/sh
matdef(83,1) = 6.5 !  power-in-temp-dependence:  "powt" in (te/tevz)**powt
matdef(84,1) = 1000.0 !  reference temperature:  "tevz"
```

```
matdef(85,1) = -2.0 !  power-in-density-dependence:  "powd" in rho**powd

de_tevmin = 1.e-8

! REGIONS
! this a hack.  one should create a new "test_pname =" problem
! from within the code.  not only is this blky, but there is a limit
! to the number of allowed lines in an input deck

numreg = 100

! jrk e0 = 333.626 j = 333.626e+16 erg

matreg( 1) = 1
siereg( 1) = 9.931594312173E+18
xlreg( 1) = 0.000000000000E+00
rhoreg( 1) = 5.629839313048E+04
xrreg( 1) = 1.000000000000E-02
radtreg( 1) = 1.e-2

matreg( 2) = 99 * 1
siereg( 2) = 99 * 1.e-2
radtreg( 2) = 99 * 1.e-2
xrreg( 2) = 99 * 1.0000000000000

xlreg( 2) = 1.000000000000E-02
rhoreg( 2) = 6.850303421911E+03
xlreg( 3) = 2.000000000000E-02
rhoreg( 3) = 2.380873954607E+03
xlreg( 4) = 3.000000000000E-02
rhoreg( 4) = 1.177262508115E+03
xlreg( 5) = 4.000000000000E-02
rhoreg( 5) = 6.942969091718E+02
xlreg( 6) = 5.000000000000E-02
rhoreg( 6) = 4.551070997103E+02
xlreg( 7) = 6.000000000000E-02
rhoreg( 7) = 3.200877294838E+02
xlreg( 8) = 7.000000000000E-02
rhoreg( 8) = 2.367377498147E+02
xlreg( 9) = 8.000000000000E-02
rhoreg( 9) = 1.818217897620E+02
xlreg( 10) = 9.000000000000E-02
rhoreg( 10) = 1.438013640083E+02
xlreg( 11) = 1.000000000000E-01
rhoreg( 11) = 1.164314439232E+02
xlreg( 12) = 1.100000000000E-01
rhoreg( 12) = 9.609804602566E+01
xlreg( 13) = 1.200000000000E-01
rhoreg( 13) = 8.059461944066E+01
xlreg( 14) = 1.300000000000E-01
rhoreg( 14) = 6.851343990990E+01
```

xlreg(15) = 1.400000000000E-01
rhoreg(15) = 5.892286538130E+01
xlreg(16) = 1.500000000000E-01
rhoreg(16) = 5.118685380229E+01
xlreg(17) = 1.600000000000E-01
rhoreg(17) = 4.485941799852E+01
xlreg(18) = 1.700000000000E-01
rhoreg(18) = 3.962049740614E+01
xlreg(19) = 1.800000000000E-01
rhoreg(19) = 3.523568376602E+01
xlreg(20) = 1.900000000000E-01
rhoreg(20) = 3.153018579344E+01
xlreg(21) = 2.000000000000E-01
rhoreg(21) = 2.837156407216E+01
xlreg(22) = 2.100000000000E-01
rhoreg(22) = 2.565801955952E+01
xlreg(23) = 2.200000000000E-01
rhoreg(23) = 2.331028667754E+01
xlreg(24) = 2.300000000000E-01
rhoreg(24) = 2.126591828897E+01
xlreg(25) = 2.400000000000E-01
rhoreg(25) = 1.947518981816E+01
xlreg(26) = 2.500000000000E-01
rhoreg(26) = 1.789811941499E+01
xlreg(27) = 2.600000000000E-01
rhoreg(27) = 1.650227015196E+01
xlreg(28) = 2.700000000000E-01
rhoreg(28) = 1.526110852000E+01
xlreg(29) = 2.800000000000E-01
rhoreg(29) = 1.415276415527E+01
xlreg(30) = 2.900000000000E-01
rhoreg(30) = 1.315908266414E+01
xlreg(31) = 3.000000000000E-01
rhoreg(31) = 1.226489509217E+01
xlreg(32) = 3.100000000000E-01
rhoreg(32) = 1.145744928392E+01
xlreg(33) = 3.200000000000E-01
rhoreg(33) = 1.072596345336E+01
xlreg(34) = 3.300000000000E-01
rhoreg(34) = 1.006127288792E+01
xlreg(35) = 3.400000000000E-01
rhoreg(35) = 9.455548258065E+00
xlreg(36) = 3.500000000000E-01
rhoreg(36) = 8.902069438903E+00
xlreg(37) = 3.600000000000E-01
rhoreg(37) = 8.395042703200E+00
xlreg(38) = 3.700000000000E-01
rhoreg(38) = 7.929452049362E+00
xlreg(39) = 3.800000000000E-01
rhoreg(39) = 7.500937580674E+00
xlreg(40) = 3.900000000000E-01

rhoreg(40) = 7.105695462047E+00
xlreg(41) = 4.000000000000E-01
rhoreg(41) = 6.740395194319E+00
xlreg(42) = 4.100000000000E-01
rhoreg(42) = 6.402110868352E+00
xlreg(43) = 4.200000000000E-01
rhoreg(43) = 6.088263766976E+00
xlreg(44) = 4.300000000000E-01
rhoreg(44) = 5.796574226781E+00
xlreg(45) = 4.400000000000E-01
rhoreg(45) = 5.525021093713E+00
xlreg(46) = 4.500000000000E-01
rhoreg(46) = 5.271807435817E+00
xlreg(47) = 4.600000000000E-01
rhoreg(47) = 5.035331435137E+00
xlreg(48) = 4.700000000000E-01
rhoreg(48) = 4.814161585038E+00
xlreg(49) = 4.800000000000E-01
rhoreg(49) = 4.607015481401E+00
xlreg(50) = 4.900000000000E-01
rhoreg(50) = 4.412741625625E+00
xlreg(51) = 5.000000000000E-01
rhoreg(51) = 4.230303761139E+00
xlreg(52) = 5.100000000000E-01
rhoreg(52) = 4.058767348852E+00
xlreg(53) = 5.200000000000E-01
rhoreg(53) = 3.897287854658E+00
xlreg(54) = 5.300000000000E-01
rhoreg(54) = 3.745100577216E+00
xlreg(55) = 5.400000000000E-01
rhoreg(55) = 3.601511789154E+00
xlreg(56) = 5.500000000000E-01
rhoreg(56) = 3.465891001727E+00
xlreg(57) = 5.600000000000E-01
rhoreg(57) = 3.337664193271E+00
xlreg(58) = 5.700000000000E-01
rhoreg(58) = 3.216307866841E+00
xlreg(59) = 5.800000000000E-01
rhoreg(59) = 3.101343823186E+00
xlreg(60) = 5.900000000000E-01
rhoreg(60) = 2.992334552478E+00
xlreg(61) = 6.000000000000E-01
rhoreg(61) = 2.888879162618E+00
xlreg(62) = 6.100000000000E-01
rhoreg(62) = 2.790609774013E+00
xlreg(63) = 6.200000000000E-01
rhoreg(63) = 2.697188320821E+00
xlreg(64) = 6.300000000000E-01
rhoreg(64) = 2.608303707228E+00
xlreg(65) = 6.400000000000E-01
rhoreg(65) = 2.523669274486E+00

xlreg(66) = 6.500000000000E-01
rhoreg(66) = 2.443020540578E+00
xlreg(67) = 6.600000000000E-01
rhoreg(67) = 2.366113179529E+00
xlreg(68) = 6.700000000000E-01
rhoreg(68) = 2.292721211814E+00
xlreg(69) = 6.800000000000E-01
rhoreg(69) = 2.222635381074E+00
xlreg(70) = 6.900000000000E-01
rhoreg(70) = 2.155661695574E+00
xlreg(71) = 7.000000000000E-01
rhoreg(71) = 2.091620115594E+00
xlreg(72) = 7.100000000000E-01
rhoreg(72) = 2.030343370344E+00
xlreg(73) = 7.200000000000E-01
rhoreg(73) = 1.971675890015E+00
xlreg(74) = 7.300000000000E-01
rhoreg(74) = 1.915472840352E+00
xlreg(75) = 7.400000000000E-01
rhoreg(75) = 1.861599248696E+00
xlreg(76) = 7.500000000000E-01
rhoreg(76) = 1.809929211707E+00
xlreg(77) = 7.600000000000E-01
rhoreg(77) = 1.760345176208E+00
xlreg(78) = 7.700000000000E-01
rhoreg(78) = 1.712737285524E+00
xlreg(79) = 7.800000000000E-01
rhoreg(79) = 1.667002784614E+00
xlreg(80) = 7.900000000000E-01
rhoreg(80) = 1.623045478033E+00
xlreg(81) = 8.000000000000E-01
rhoreg(81) = 1.580775235447E+00
xlreg(82) = 8.100000000000E-01
rhoreg(82) = 1.540107539982E+00
xlreg(83) = 8.200000000000E-01
rhoreg(83) = 1.500963075251E+00
xlreg(84) = 8.300000000000E-01
rhoreg(84) = 1.463267347305E+00
xlreg(85) = 8.400000000000E-01
rhoreg(85) = 1.426950338191E+00
xlreg(86) = 8.500000000000E-01
rhoreg(86) = 1.391946188130E+00
xlreg(87) = 8.600000000000E-01
rhoreg(87) = 1.358192903658E+00
xlreg(88) = 8.700000000000E-01
rhoreg(88) = 1.325632089330E+00
xlreg(89) = 8.800000000000E-01
rhoreg(89) = 1.294208700845E+00
xlreg(90) = 8.900000000000E-01
rhoreg(90) = 1.263870817664E+00
xlreg(91) = 9.000000000000E-01

rhoreg(91) = 1.234569433383E+00
xlreg(92) = 9.100000000000E-01
rhoreg(92) = 1.206258262300E+00
xlreg(93) = 9.200000000000E-01
rhoreg(93) = 1.178893560755E+00
xlreg(94) = 9.300000000000E-01
rhoreg(94) = 1.152433961985E+00
xlreg(95) = 9.400000000000E-01
rhoreg(95) = 1.126840323327E+00
xlreg(96) = 9.500000000000E-01
rhoreg(96) = 1.102075584730E+00
xlreg(97) = 9.600000000000E-01
rhoreg(97) = 1.078104637640E+00
xlreg(98) = 9.700000000000E-01
rhoreg(98) = 1.054894203383E+00
xlreg(99) = 9.800000000000E-01
rhoreg(99) = 1.032412720285E+00
xlreg(100) = 9.900000000000E-01
rhoreg(100) = 1.010630238810E+00

7.5 For the Noh Problem

```
! 1-D spherical NOH problem
! UNITS are in cgs, except for temperature which is in ev

pname = "noh-100pt" !Problem name

! CONTROL

tmax = 0.3 !Maximum simulation time (real time in sec.)
dtedt(1) = 0.3
dodmpxdt = .true. ! turns on dt adjustment to get dumps at exact dedt time
tedit = 0.3 !Frequency of edit dumps in real simulation time

dtnext = 1.0e-16 ! initial time step
!dtmax = 1.0 ! maximum dt allowed
!dtforce = 1.0e-12

kread = -1 !kread<0 is new problem; kread>=0 then restart cycle = kread
uselast = .true. !If true and kread<0 then use "pname-lastdump" as restart file
ncmax = 500000 !Maximim # of cycles
ncedit = 0 !Frequency of edit dumps in # of cycles
modcyc = 0 !Frequency of status edits in # of cycles
shortmodcyc = 100 !Frequency of short edit dumps

! GRID
! RECOMMENDATION: Do not use the mesh variables numrho, numfine, numlev,
! smallke, and mxcells. Rather use sizemat.

imxset = 100 !Number of level 1 cells in the x-direction
dxset = 0.01 !Size of level 1 cells in x-direction

! MATERIALS

nummat = 1 !Number of materials

! Use sizemat to control mesh refinement. Note that this has to be
! coordinated with the dxset parameter above.

sizemat(1) = 0.01 ! 1 level of refinement for material 1
!sizemat(1) = 0.005 ! 2 levels, effective 200 points
!sizemat(1) = 0.0025 ! 3 levels, effective 400 points
!sizemat(1) = 0.00125 ! 4 levels, effective 800 points
!sizemat(1) = 0.000625 ! 5 levels, effective 1600 points
!sizemat(1) = 0.0003125 ! 6 levels, effective 3200 points

! SPHERE

cylin = .false. !true=r-z cylindrical geometry; false=not cylindrical
sphere = .true. !true=spherical geometry; false=not spherical
```

```
! "freeze regions" are inflow/outflow boundary conditions for an Eulerian mesh
! freeze_x_lo and freeze_x_hi determine the extent of the freeze region in
! the x-direction [cm]. The region goes from freeze_x_lo to freeze_x_hi.

freeze_num = 1 !Number of freeze regions
freeze_x_lo = 0.90 !Begin freeze region in x-direction [cm]
freeze_x_hi = 1.00 !End of freeze region in x-direction [cm]

! EOS

keos = 0 !0 = ideal gas EOS; 1 = SESAME EOS; 2 = N/A; 3= new TEOS files
matdef(16,1) = 0.6666667 !Specifying (gamma-1) Gamma = 5/3
matdef(30,1) = 1.0e12 !Specifying Cv - specific heat [erg/gm/ev]

! REGIONS

numreg = 1 !Number of regions

matreg(1) = 1 !Region 1 is of material 1
rhoreg(1) = 1.0 !Density of region 1 [gm/cc]
siereg(1) = 1.0e-10 !Specific Internal energy of region 1 [erg/gm]
xdreg(1) = -1.0 !Initial radial velocity of region 1 [cm/s]

! Hydro options

dohydro = .true. !turn on/off hydro, default is on (.true.)
hydro_version = 2 !hydro_version=0 is default
!If = 0, use standard hydro method(Lagrange and re-map). If = 1,
!use direct Eulerian hydro method. If = 2, use experimental hydro method.
```

7.6 For the Sedov Problem

```
! Sedov Blast Wave (1-d)

pname = "sed-120pt" !Problem name

! CONTROL
tmax = 1.0 !Maximum simulation time (real time in seconds)
dtedt(1) = 1.0
dodmpxdt = .true. ! turns on dt adjustment to get dumps at exact dedt time
tedit = 1.0 !Frequency of edit dumps in real simulation time
dtnext = 1.0e-16 ! initial time step
!dtmax = 1.0 ! maximum dt allowed
!dtforce = 1.0e-12

kread = -1 !kread<0 is a new problem; kread>=0 then restart cycle=kread
uselast = .true. !If true and kread<0 then use "pname-lastdum" as restart file
ncmax = 500000 !Maximum # of cycles
ncedit = 0 ! disable cycle frequency for binary dumps
modcyc = 0 ! frequency of status edits
shortmodcyc = 200 ! frequency of short edits

dohydro = .true. !true=hydro routines turned on (default); false=hydro not on
doheat = .false. !true=do heat conduction; false=do not do heat cond.(default)
onetemp = .true. !true=equil. diffusion (one temperature)
!false=nonequilibrium diffusion (two temperatures) (default)

! GRID
! RECOMMENDATION: Do not use the mesh variables numrho, numfine, numlev,
! smallke, and mxcells. Rather use sizemat.
imxset = 120 !Number of Level 1 zones in the x-direction:
dxset = 0.01 !Size of Level-1 zones in the x-direction:

! SPHERE
cylin = .false. !true=r-z cylindrical geometry; false=not cylindrical
sphere = .true. !true=spherical geometry; false=not spherical

! MATERIALS
nummat = 1 !Number of materials

! Use sizemat to control mesh refinement. Note that this has to be
! coordinated with the dxset parameter above.

sizemat(1) = 0.01 ! 1 level of refinement for material 1
!sizemat(1) = 0.005 ! 2 levels, effective 240 points
!sizemat(1) = 0.0025 ! 3 levels, effective 480 points
!sizemat(1) = 0.00125 ! 4 levels, effective 960 points
!sizemat(1) = 0.000625 ! 5 levels, effective 1920 points
!sizemat(1) = 0.0003125 ! 6 levels, effective 3840 points

! EOS
```

```
keos = 0 !Use ideal gas equation of state
matdef(16,1) = 0.4 !Specifying (gamma-1); gamma = 1.4
matdef(30,1) = 1.0e-4 !Specifying Cv - specific heat [erg/gm/ev]

! REGIONS
numreg = 2 !Number of regions

matreg(1) = 1 !Region 1 is of material 1
rhoreg(1) = 1.0 !Density of region 1 [gm/cc]
siereg(1) = 2.539731e-8 !Specific internal energy of region 1 [erg/cc]

matreg(2) = 1 !Region 2 is of material 1
rhoreg(2) = 1.0 !Density of region 2 [gm/cc]
siereg(2) = 2.539731e+4 !Specific internal energy of region 2 [erg/cc]
xlreg(2) = 0.0 !Left x-boundary of region 2
xrreg(2) = 0.02 !Right x-boundary of region 2
```

8. Appendix B – Analytic Codes

8.1 To the Su & Olson Problem

```
      program suo
      implicit none

c..solves the su-olson problem
c..cgs units throughout, except some duly noted temperatures that are in ev

c..declare
      character*80 outfile,string
      integer i,nstep,iargc
      double precision time,zpos,trad_bc_ev,opac,alpha,
     1                  erad,trad,trad_ev,tmat,tmat_ev,
     2                  zlo,zhi,zstep,value

c..some physics
      double precision clight,ssol,asol
      parameter (clight = 2.99792458d10,
     1           ssol = 5.67051d-5,
     2           asol = 4.0d0 * ssol / clight)

c..popular formats
01    format(1x,t4,a,t10,a,t22,a,t34,a,t46,a,t58,a,t70,a)
02    format(1x,i4,1p8e12.4)

c..input parameters
      trad_bc_ev = 1.0d3
      opac = 1.0d0
      alpha = 4.0d0*asol
      time = 1.0d-9

c..number of grid points, spatial domain, spatial step size
      nstep = 100
      zlo = 0.0d0
      zhi = 20.0d0
      zstep = (zhi - zlo)/float(nstep)

c..output file
      outfile = '100pt_1em10.dat'
      open(unit=2,file=outfile,status='unknown')
      write(2,02) nstep,time
      write(2,01) 'i','x','trad','tev'

c..use the mid-cell points to match various eularian hydrocodes
      do i=1,nstep
        zpos = zlo + 0.5d0*zstep + float(i-1)*zstep

        call so_wave(time,zpos,trad_bc_ev,opac,alpha,
     1               erad,trad,trad_ev,tmat,tmat_ev)

        write(6,40) i,zpos,trad_ev,tmat_ev
        write(2,40) i,zpos,trad_ev,tmat_ev
40      format(1x,i4,1p8e14.6)

      enddo
```

```
c..close up stop
      close(unit=2)
      end

      subroutine so_wave(time,zpos,trad_bc_ev,opac,alpha,
     1                   erad,trad,trad_ev,tmat,tmat_ev)
      implicit none
      save

c..provides solution to the su-olson problem

c..input:
c..time = time point where solution is desired
c..zpos = spaatial point where solution is desired
c..trad_bc_ev = boundary condition temperature in electron volts
c..opac = the opacity in cm**2/g
c..alpha = coefficient of the material equation of state c_v = alpha T_mat*

c..output:
c..erad = energy desnity of radiation field erg/cm**3
c..trad = temperature of radiation field kelvin
c..trad_ev = temperature of radiation field electron volts
c..trad = temperature of material field kelvin
c..trad_ev = temperature of material field electron volts

c..declare the input
      double precision time,zpos,trad_bc_ev,opac,alpha,
     1                 erad,trad,trad_ev,tmat,tmat_ev

c..local variables
      double precision trad_bc,ener_in,xpos,epsilon,tau,ialpha,
     1                 uans,vans,usolution,vsolution

c..some physics
      double precision clight,ssol,asol,kev,rt3,a4,a4c
      parameter (clight = 2.99792458d10,
     1           ssol = 5.67051d-5,
     2           asol = 4.0d0 * ssol / clight,
     3           kev = 8.617385d-5,
     4           rt3 = 1.7320508075688772d0,
     5           a4 = 4.0d0*asol,
     6           a4c = a4 * clight)

c..derived parameters and conversion factors
      trad_bc = trad_bc_ev/kev
      ener_in = asol * trad_bc**4
      xpos = rt3 * opac * zpos
      ialpha = 1.0d0/alpha
      tau = a4c * opac * ialpha * time
      epsilon = a4 * ialpha

c..get the dimensionless solutions
      uans = usolution(xpos,tau,epsilon)
      vans = vsolution(xpos,tau,epsilon,uans)

c..compute the physical solution
      erad = uans * ener_in
      trad = (erad/asol)**(0.25d0)
      trad_ev = trad * kev

      tmat = (vans*ener_in/asol)**(0.25d0)
      tmat_ev = tmat * kev
      return
      end

      double precision function usolution(posx_in,tau_in,epsilon_in)
      implicit none
```

```
      save

c..computes the u solution for the su-olson problem

c..declare the pass
      double precision posx_in,tau_in,epsilon_in

c..common block communication
      double precision posx,tau,epsilon
      common /bdoor/ posx,tau,epsilon
      integer jwant
      common /rots/ jwant

c..local variables
      external midpnt,upart1,upart2,
     1                        gamma_one_root,gamma_two_root
      logical bracket
      integer i,niter
      double precision xi1,xi2,midpnt,upart1,upart2,sum1,sum2,
     1                        gamma_one_root,gamma_two_root,
     2                        zbrent,eta_hi,eta_lo,eta_int

      double precision tol,eps,eps2,pi,rt3,rt3opi
      parameter (tol = 1.0d-6,
     &                         eps = 1.0d-10,
     &                         eps2 = 1.0d-8,
     &                         pi = 3.14159265358979323384d0,
     &                         rt3 = 1.7320508075688772d0,
     &                         rt3opi = rt3/pi)

c..transfer input to common block
      posx = posx_in
      tau = tau_in
      epsilon = epsilon_in

c..integrand may not oscillate for small values of posx
      eta_lo = 0.0d0
      eta_hi = 1.0d0
      sum1 = 0.0d0
      jwant = 1
      bracket = (gamma_one_root(eta_lo)*gamma_one_root(eta_hi).le.0.0)
      if (.not.bracket) then
        call qromo(upart1,eta_lo,eta_hi,eps,sum1,midpnt)

c..integrate over each oscillitory piece
      else
        do i=1,100
          jwant = i
          eta_int = zbrent(gamma_one_root,eta_lo,eta_hi,tol,niter)
          call qromo(upart1,eta_lo,eta_int,eps,xi1,midpnt)
          sum1 = sum1 + xi1
          eta_lo = eta_int
          if (abs(xi1) .le. eps2) goto 10
        enddo
10      continue
      end if

c..integrand may not oscillate for small values of posx
      eta_lo = 0.0d0
      eta_hi = 1.0d0
      sum2 = 0.0d0
      jwant = 1
      bracket = (gamma_two_root(eta_lo)*gamma_two_root(eta_hi).le.0.0)
      if (.not.bracket) then
        call qromo(upart2,eta_lo,eta_hi,eps,sum2,midpnt)

c..integrate from hi to lo on this piece
```

```
      else
       do i=1,100
        jwant = i
        eta_int = zbrent(gamma_two_root,eta_hi,eta_lo,tol,niter)
        call qromo(upart2,eta_hi,eta_int,eps,xi2,midpnt)
        sum2 = sum2 + xi2
        eta_hi = eta_int
        if (abs(xi2) .le. eps2) goto 20
       enddo
20     continue
       sum2 = -sum2
      endif

c..done
      usolution = 1.0d0 - 2.0d0*rt3opi*sum1 - rt3opi*exp(-tau)*sum2
      return
      end

      double precision function vsolution(posx_in,tau_in,epsilon_in,
     1                                    uans)
      implicit none
      save

c..computes the v solution for the su-olson problem

c..declare the pass
      double precision posx_in,tau_in,epsilon_in,uans

c..common block communication
      double precision posx,tau,epsilon
      common /bdoor/ posx,tau,epsilon
      integer jwant
      common /rots/ jwant

c..local variables
      external midpnt,vpart1,vpart2,
     1                 gamma_two_root,gamma_three_root
      logical bracket
      integer i,niter
      double precision xi1,xi2,midpnt,vpart1,vpart2,sum1,sum2,
     1                 gamma_two_root,gamma_three_root,
     2                 zbrent,eta_hi,eta_lo,eta_int

      double precision tol,eps,eps2,pi,rt3,rt3opi
      parameter (tol = 1.0d-6,
     &           eps = 1.0d-10,
     &           eps2 = 1.0d-8,
     &           pi = 3.14159265358979323 84d0,
     &           rt3 = 1.732050807568 8772d0,
     &           rt3opi = rt3/pi)

c..transfer input to common block
      posx = posx_in
      tau = tau_in
      epsilon = epsilon_in

c..integrand may not oscillate for small values of posx
      eta_lo = 0.0d0
      eta_hi = 1.0d0
      sum1 = 0.0d0
      jwant = 1
      bracket = (gamma_three_root(eta_lo)*gamma_three_root(eta_hi)
     1           .le. 0.0)
      if (.not.bracket) then
       call qromo(vpart1,eta_lo,eta_hi,eps,sum1,midpnt)

c..integrate over each oscillitory piece
```

```
c..from 1 to 0 on this part; this one really oscillates
      else
       do i=1,100
        jwant = i
        eta_int = zbrent(gamma_three_root,eta_hi,eta_lo,tol,niter)
        call qromo(vpart1,eta_hi,eta_int,eps,xi1,midpnt)
        sum1 = sum1 + xi1
        eta_hi = eta_int
        if (abs(xi1) .le. eps2) goto 10
       enddo
10     continue
       sum1 = -sum1
      end if

c..integrand may not oscillate for small values of posx
      eta_lo = 0.0d0
      eta_hi = 1.0d0
      sum2 = 0.0d0
      jwant = 1
      bracket = (gamma_two_root(eta_lo)*gamma_two_root(eta_hi).le.0.0)
      if (.not.bracket) then
       call qromo(vpart2,eta_lo,eta_hi,eps,sum2,midpnt)

c..integrate over each oscillitory piece
c..from 1 to 0 on this part; this one really oscillates
      else
       do i=1,100
        jwant = i
        eta_int = zbrent(gamma_two_root,eta_hi,eta_lo,tol,niter)
        call qromo(vpart2,eta_hi,eta_int,eps,xi2,midpnt)
        sum2 = sum2 + xi2
        eta_hi = eta_int
        if (abs(xi2) .le. eps2) goto 20
       enddo
20     continue
       sum2 = -sum2
      endif

c..done
      vsolution = uans - 2.0d0*rt3opi*sum1 + rt3opi*exp(-tau)*sum2
      return
      end


      double precision function upart1(eta)
      implicit none
      save

c..equation 36 of su & olson jqsrt 1996, first integrand

c..declare the pass
      double precision eta

c..common block communication
      double precision posx,tau,epsilon
      common /bdoor/ posx,tau,epsilon

c..local variables
      double precision numer,denom,gamma_one,theta_one,tiny
      parameter (tiny = 1.0d-14)

      numer = sin(posx*gamma_one(eta,epsilon) + theta_one(eta,epsilon))

      denom = eta * sqrt(3.0d0 + 4.0d0*gamma_one(eta,epsilon)**2)
      denom = max(tiny,denom)

      upart1= exp(-tau*eta*eta) * numer/denom
```

```
      return
      end

      double precision function upart2(eta)
      implicit none
      save

c..equation 36 of su & olson jqsrt 1996, second integrand

c..declare the pass
      double precision eta

c..common block communication
      double precision posx,tau,epsilon
      common /bdoor/ posx,tau,epsilon

c..local variables
      double precision numer,denom,gamma_two,theta_two,tiny
      parameter (tiny = 1.0d-14)

      numer = sin(posx*gamma_two(eta,epsilon) + theta_two(eta,epsilon))

      denom = eta * (1.0d0 + epsilon*eta) *
     &        sqrt(3.0d0 + 4.0d0*gamma_two(eta,epsilon)**2)
      denom = max(tiny,denom)

      upart2= exp(-tau/(max(tiny,eta*epsilon))) * numer/denom
      return
      end

      double precision function vpart1(eta)
      implicit none
      save

c..equation 42 of su & olson jqsrt 1996, first integrand

c..declare the pass
      double precision eta

c..common block communication
      double precision posx,tau,epsilon
      common /bdoor/ posx,tau,epsilon


c..local variables
      double precision numer,denom,gamma_two,theta_two,
     &                 gamma_three,theta_three,eta2,tiny
      parameter (tiny = 1.0d-14)

      eta2 = eta * eta

      numer = sin(posx*gamma_three(eta,epsilon) +
     &            theta_three(eta,epsilon))

      denom = sqrt(4.0d0 - eta2 + 4.0d0*epsilon*eta2*(1.0d0 - eta2))
      denom = max(tiny,denom)

      vpart1 = exp(-tau*(1.0d0 - eta2)) * numer/denom
      return
      end

      double precision function vpart2(eta)
      implicit none
      save

c..equation 42 of su & olson jqsrt 1996, second integrand
```

```
c..declare the pass
      double precision eta

c..common block communication
      double precision posx,tau,epsilon
      common /bdoor/ posx,tau,epsilon


c..local variables
      double precision numer,denom,gamma_two,theta_two,tiny
      parameter (tiny = 1.0d-14)

      numer = sin(posx*gamma_two(eta,epsilon) + theta_two(eta,epsilon))

      denom = eta * sqrt(3.0d0 + 4.0d0*gamma_two(eta,epsilon)**2)
      denom = max(tiny,denom)

      vpart2= exp(-tau/(max(tiny,eta*epsilon))) * numer/denom
      return
      end

      double precision function gamma_one_root(eta_in)
      implicit none
      save

c..used by a root finder to determine the integration inveral

c..declare the pass
      double precision eta_in

c..common block communication
      double precision posx,tau,epsilon
      common /bdoor/ posx,tau,epsilon
      integer jwant
      common /rots/ jwant

c..local variables
      double precision gamma_one,theta_one,pi,twopi
      parameter (pi = 3.14159265358979323384d0,
     1                        twopi = 2.0d0*pi)

c..go
      gamma_one_root = gamma_one(eta_in,epsilon)*posx
     &                       + theta_one(eta_in,epsilon)
     &                       - jwant * twopi
      return
      end

      double precision function gamma_two_root(eta_in)
      implicit none
      save

c..used by a root finder to determine the integration inveral

c..declare the pass
      double precision eta_in

c..common block communication
      double precision posx,tau,epsilon
      common /bdoor/ posx,tau,epsilon

      integer jwant
      common /rots/ jwant

c..local variables
      double precision gamma_two,theta_two,pi,twopi
      parameter (pi = 3.14159265358979323384d0,
```

```
     1                        twopi = 2.0d0*pi)

c..go
      gamma_two_root = gamma_two(eta_in,epsilon)*posx
     &                       + theta_two(eta_in,epsilon)
     &                       - jwant * twopi
      return
      end


      double precision function gamma_three_root(eta_in)
      implicit none
      save

c..used by a root finder to determine the integration inveral

c..declare the pass
      double precision eta_in

c..common block communication
      double precision posx,tau,epsilon
      common /bdoor/ posx,tau,epsilon

      integer jwant
      common /rots/ jwant

c..local variables
      double precision gamma_three,theta_three,pi,twopi
      parameter (pi = 3.14159265358979323384d0,
     1                        twopi = 2.0d0*pi)

c..go
      gamma_three_root = gamma_three(eta_in,epsilon)*posx
     &                       + theta_three(eta_in,epsilon)
     &                       - jwant * twopi
      return
      end

      double precision function theta_one(eta,epsilon)
      implicit none
      save

c..equation 38 of su & olson jqsrt 1996
c..declare the pass
      double precision eta,epsilon

c..local variables
      double precision gamma_one

      theta_one = acos(sqrt(3.0d0/(3.0d0 + 4.0d0*gamma_one(eta,epsilon)**2)))
      return
      end

      double precision function theta_two(eta,epsilon)
      implicit none
      save

c..equation 38 of su & olson jqsrt 1996
c..declare the pass
      double precision eta,epsilon

c..local variables
      double precision gamma_two

      theta_two = acos(sqrt(3.0d0 /(3.0d0 + 4.0d0*gamma_two(eta,epsilon)**2)))
      return
      end
```

```
      double precision function theta_three(eta,epsilon)
      implicit none
      save

c..equation 43 of su & olson jqsrt 1996
c..declare the pass
      double precision eta,epsilon

c..local variables
      double precision gamma_three

      theta_three = acos(sqrt(3.0d0 /(3.0d0 + 4.0d0*gamma_three(eta,epsilon)**2)))
      return
      end

      double precision function gamma_one(eta,epsilon)
      implicit none
      save

c..equation 37 of su & olson jqsrt 1996
c..declare the pass
      double precision eta,epsilon

c..local variables
      double precision ein,tiny
      parameter (tiny = 1.0d-14)

      ein = max(tiny,min(eta,1.0d0-tiny))
      gamma_one = ein * sqrt(epsilon + 1.0d0/(1.0d0 - ein*ein))
      return
      end

      double precision function gamma_two(eta,epsilon)
      implicit none
      save

c..equation 37 of su & olson jqsrt 1996
c..declare the pass
      double precision eta,epsilon

c..local variables
      double precision ein,tiny
      parameter (tiny = 1.0d-14)

      ein = max(tiny,min(eta,1.0d0-tiny))
      gamma_two = sqrt((1.0d0 - ein) * (epsilon + 1.0d0/ein))
      return
      end

      double precision function gamma_three(eta,epsilon)
      implicit none
      save

c..equation 43 of su & olson jqsrt 1996

c..declare the pass
      double precision eta,epsilon

c..local variables
      double precision ein,tiny
      parameter (tiny = 1.0d-14)

      ein = max(tiny,min(eta,1.0d0-tiny))
      gamma_three = sqrt((1.0d0 - ein*ein)*(epsilon + 1.0d0/(ein*ein)))
      return
      end
```

8.2 To the Coggeshall #8 Problem

```
      program cog_test
      implicit none

c..tests the cog8 solver
      character*80 outfile,string
      integer i,nstep,iargc
      double precision time,zpos,zleft,zright,rho0,temp0,
     1                  alpha,beta,gamma,cv,den,tev,ener,pres,vel,
     2                  zlo,zhi,zstep,value

c..popular formats
01    format(1x,t4,a,t10,a,t22,a,t34,a,t46,a,t58,a,t70,a)
02    format(1x,i4,1p8e12.4)

c..input parameters, time in shakes temp0 in eV, cv in erg/g/eV
      time = 2.0d0
      rho0 = 3.0d0
      temp0 = 100.0d0
      alpha = -1.0d0
      beta = 2.0d0
      gamma = 5.0d0/3.0d0
      cv = 1.0d12

c..number of grid points, spatial domain, spatial step size
      nstep = 200
      zlo = 0.0d0
      zhi = 2.0d0
      zstep = (zhi - zlo)/float(nstep)

c..output file
      outfile = '200pt_2p0sh.dat'
      open(unit=2,file=outfile,status='unknown')
      write(2,02) nstep,time
      write(2,01) 'i','x','den','vel','pres','tev'

c..to match hydrocode output, use the mid-cell points
      do i=1,nstep
        zpos = zlo + 0.5d0*zstep + float(i-1)*zstep
        zleft = zpos - 0.5d0 * zstep
        zright = zpos + 0.5d0 * zstep

        call cog8_1d_sph(time,zleft,zright,
     1                   rho0,temp0,alpha,beta,gamma,cv,
     2                   den,tev,ener,pres,vel)

        write(6,40) i,zpos,den,tev,ener,pres,vel
        write(2,40) i,zpos,den,tev,ener,pres,vel
40      format(1x,i4,1p8e14.6)

      enddo

c..close up stop
      close(unit=2)
      end

      subroutine cog8_1d_sph(time,xl,xr,
     1                   rho0,temp0,alpha,beta,gamma,cv,
     2                   den,tev,ener,pres,vel)
      implicit none
      save

c..solves the coggeshall problem #8 in one-dimension, spherical coordinates
c..s.v. coggeshall, phys fluids a 3, 757, 191

c..input:
```

```
c..time = time in shakes units of 1e-8 s,
c..xl = left boundary of cell in cm
c..xr = right boundary of cell in cm
c..rho0 = density constant g/cm**3
c..temp0 = temperature constant in ev
c..alpha = dimensionless constant
c..beta = dimensionless consnat
c..gamma = perfect gas
c..cv = specific heat @ constant volume, erg/g/eV

c..output:
c..den = density g/cm**3
c..tev = temperature ev
c..ener = specific internal energy erg/g
c..pres = presssure erg/cm**3
c..vel = velocity cm/sh

c..declare the pass
      double precision time,xl,xr,
     1                 rho0,temp0,alpha,beta,gamma,cv,
     2                 den,tev,ener,pres,vel

c..local variables
      integer ik,ikp1
      parameter (ik = 2, ikp1 = ik + 1)
      double precision seventh,rexp1,texp1,
     1                 dv,dm,dmv,de,aied,pi,volfac,velfac
      parameter (pi = 3.14159265358979323 84d0,
     1                  volfac = 4.0d0 * pi,
     2                  velfac = 1.0d8)

c..the various exponents
      seventh = ( ik - 1 ) / ( beta - alpha + 4.0d0)
      rexp1 = seventh
      texp1 = ( ik + 1 ) + seventh

c..this is the cell averaged solution, slightly different than
c..the exact point solution

      dv = volfac * ( xr**ikp1 - xl**ikp1 ) / ikp1
      dm = volfac * rho0 * ( xr**(ik+1+rexp1) - xl**(ik+1+rexp1) )
     &                   / (texp1 * time**texp1)
      dmv = volfac * rho0 * ( xr**(ik+2+rexp1) - xl**(ik+2+rexp1) )
     &                    / ((texp1 + 1) * time**(texp1 + 1))
      de = volfac * rho0 * ( xr**(ik+3+rexp1) - xl**(ik+3+rexp1) )
     &                   / ((texp1 + 2) * time**(texp1 + 2)) * 0.50d0

c..and the quantities of interest
      den = dm / dv
      vel = dmv / dm * velfac
      aied = rho0 * temp0 * cv / time**5
      ener = aied / den
      tev = ener / cv
      pres = (gamma - 1.0d0) * aied

      return
      end
```

8.3 To the Mader Problem

```
      program trare
      implicit none
      save

c..test the rarefaction wave
      character*40 outfile
      integer i,nsteps,lout,lenstr
      double precision time,xmin,xmax,u_piston,p_cj,d_cj,gamma,xstep,
     1                  xlab,vel,pres,cs,den,xdet

c..popular format statements
01    format(1x,t10,a,t22,a,t34,a,t46,a,t58,a,t70,a)
02    format(1x,i4,1p6e12.4)

c..set the input parameters
      nsteps = 200
      time = 6.25d-6
      outfile = 'mad_200pt_t6p25.dat'

c..set domain, piston speed, cj pressure, detonaton speed, and eos gamma
c..for a mader test problem
      xmin = 0.0d0
      xmax = 5.0d0
      u_piston = 0.0d0
      p_cj = 3.0d11
      d_cj = 8.0d5
      gamma = 3.0d0
      xstep = (xmax - xmin)/float(nsteps)

c..open the output file
      lout = lenstr(outfile,40)
      open(unit=11, file=outfile(1:lout),status='unknown')
      write (11,01) 'xdet','xlab','vel','den','pres','csound'


c..to match typical eulerian hydrocode output use the cell centers
      do i = 1,nsteps
        xlab = xmin + 0.5d0*xstep + float(i-1)*xstep

        call rare(time,xlab,xstep,p_cj,d_cj,gamma,u_piston,
     1            vel,pres,cs,den,xdet)

c..burns from xmax to xminin rage, so reverse the sign of u on output
        write(11,02) i, xdet, xlab, -vel, den, pres, cs
      enddo

c..close up shop
      close(unit=2)
      end

      subroutine rare(time,xlab,dx,p_cj,d_cj,gam,u_piston,
     1                u,p,c,rho,xdet)
      implicit none
      save

c..returns the rarefaction wave solution given on page 24
c..of fickett and davis

c..input:
c..time = time for desired solutioon (s)
c..xlab = position in fixed lab frame, eularian frame (cm)
c..dx = width of grid cell (cm)
c..p_cj = chapman-jouget pressure (erg/cm**3)
c..d_cj = chapman-jouget density (g/cm**3)
c..gam = ratio of specific heats (dimensionless)
```

```
c..u_piston = speed of piston (cm/s)

c..output:
c..u = material speed (cm/s)
c..p = pressure (erg/cm**3)
c..c = sound speed (cm/s)
c..rho = mass density (g/cm**3)
c..xdet = position relative to detonation front, lagrangian frame (cm)

c..declare the pass
      double precision time,xlab,dx,p_cj,d_cj,gam,u_piston,
     1                  u,p,c,rho,xdet

c..local variables
      double precision gamp1,rho_0,rho_cj,c_cj,u_cj,
     1                  gamm1,aa,bb,b,d,dd,ee,bp1,dp1,
     2                  um,xp,half,x,x1,h,x2,dxp,ur,pr,cr,rhor,dist,tol

c..some constants and factors
      gamp1 = gam + 1.0d0
      rho_0 = gamp1 * p_cj /d_cj**2
      rho_cj = rho_0 * gamp1/gam
      c_cj = gam*d_cj/gamp1
      u_cj = d_cj/gamp1

      gamm1 = gam - 1.0d0
      aa = 1.0d0/(2.0d0 * c_cj * time)
      bb = (2.0d0 - gamm1 * u_cj /c_cj)/gamp1
      b = 2.0d0 * gam/gamm1
      d = 2.0d0/gamm1
      dd = 2.0d0/(time*gamp1)
      ee = gamm1 * (u_cj - 2.0d0*c_cj/gamm1)/gamp1
      bp1 = b + 1.0d0
      dp1 = d + 1.0d0

      um = gamm1 * (u_cj - 2.0d0*c_cj/gamm1)/gamp1
      xp = 0.5d0 * gamp1 * time * (u_piston - um)
      xdet = d_cj*time - xlab
      dist = abs(xdet - xp)
      tol = 0.1d0 * dx

c..solution in the frame relative to detonation front, lagrangian frame
      x = xdet
      half = 0.5d0*dx
      x1 = x - half

c..solution in the rarefaction fan
      if (dist.gt.tol .and. xdet.gt.xp) then
        u = dd*(x1+half) + ee
        p = p_cj*((aa*(x1+dx)+bb)**bp1 - (aa*x1 + bb)**bp1)/(dx*aa*bp1)
        c = c_cj*(aa*(x1+half) + bb)
        rho = rho_cj*((aa*(x1+dx)+bb)**dp1 - (aa*x1+bb)**dp1)/(dx*aa*dp1)

c..solution if right at the transition point
      else if (dist .le. tol) then

c.. partial q's
        x2 = x1+dx
        dxp = (x2-xp)
        h = dxp/2
        u = dd*(x1+h) + ee
        p = p_cj*((aa*(x1+dxp)+bb)**bp1 - (aa*x1+bb)**bp1)/(dxp*aa*bp1)
        c = c_cj*(aa*(x1+h) + bb)
        rho = rho_cj*((aa*(x1+dxp)+bb)**dp1-(aa*x1+bb)**dp1)/(dxp*aa*dp1)

c..residual q's
        ur = u_piston
```

```
          pr = p_cj*(1+gamm1*(u-u_cj)/(2.0d0*c_cj))**(2.0d0*gam/gamm1)
          cr = c_cj*(1+gamm1*(u-u_cj)/(2.0d0*c_cj))
          rhor = rho_cj*(p/p_cj)**(1.0d0/gam)

c..avg q's
          u = ur + (u-ur)*2.0d0*h/dx
          p = pr +(p-pr)*2.0d0*h/dx
          c = cr + (c-cr)*2.0d0*h/dx
          rho = rho + (rho - rhor)*2.0d0*h/dx

c..solution in the constant state
        else
          u = u_piston
          p = p_cj*(1+gamm1*(u-u_cj)/(2.0d0*c_cj))**(2.0d0*gam/gamm1)
          c = c_cj*(1+gamm1*(u-u_cj)/(2.0d0*c_cj))
          rho = rho_cj*(p/p_cj)**(1.0d0/gam)
        endif
        return
        end
```

8.4 To the Reinicke & Meyer-ter-Vehn Problem

```
      program rmtv_test
      implicit none

c..tests the rmtv solver
c..declare
      character*80 outfile,string
      integer i,nstep,iargc
      double precision zpos,
     1                 aval,bval,chi0,gamma,bigamma,rf,xif,xis,beta0,
     2                 g0,den,tev,ener,pres,vel,
     3                 zlo,zhi,zstep,value

c..popular formats
01    format(1x,t4,a,t8,a,t20,a,t32,a,t44,a,t56,a,t68,a,t80,a,
     1       t92,a,t104,a,t116,a,t128,a,t140,a,t152,a,t164,a)
02    format(1x,i4,1p8e12.4)

c..input parameters
      aval = -2.d0
      bval = 6.5d0
      chi0 = 1.0d0
      gamma = 1.25d0
      bigamma = 1.0d0
      rf = 0.9d0
      xif = 2.0d0
      xis = 1.0d0
      beta0 = 7.197534d+7
      g0 = 1.0d0

c..number of grid points, spatial domain, spatial step size
      zlo = 0.0d0
      zhi = 1.0d0
      nstep = 100
      zstep = (zhi - zlo)/float(nstep)

c..output file
      outfile = 'rmtv_100pt.dat'
      open(unit=2,file=outfile,status='unknown')
      write(2,02) nstep
      write(6,01) 'i','x','den','tev','ener','pres','velx'
      write(2,01) 'i','x','den','tev','ener','pres','velx'

c..to match hydrocode output, use the mid-cell points
      do i=1,nstep
        zpos = zlo + 0.5d0*zstep + float(i-1)*zstep

        call rmtv_1d(zpos,
     1               aval,bval,chi0,gamma,bigamma,
     2               rf,xif,xis,beta0,g0,
     3               den,tev,ener,pres,vel)

        write(6,02) i,zpos,den,tev,ener,pres,vel
        write(2,02) i,zpos,den,tev,ener,pres,vel
      enddo

c..close up stop
      close(unit=2)
      end

      subroutine rmtv_1d(rpos,
     1                   aval_in,bval_in,chi0,gamma,bigamma,
     2                   rf,xif_in,xis,beta0_in,g0,
     3                   den,tev,ener,pres,vel)
      implicit none
      save
```

```
c..solves the rmtv in one-dimension, spherical coordinates
c..this a highly simplified version of kamm's code that solves for
c..a given point value for a specific tri-lab verification test problem.

c..input:
c..rpos = desired radial position for the solution in cm
c..aval_in = power in thermal conductivity chi0 * rho**a * T**b
c..bval_in = power in thermal conductivity chi0 * rho**a * T**b
c..chi0 = coefficient in thermal conductivity chi0 * rho**a * T**b
c..gamma = ratio of specific heats
c..bigamma = Gruneisen coefficient (gamma-1)*ener = pres/den = G*temp
c..rf = position of the heat front in cm
c..xif = dimensionless position of the heat front
c..xis = dimensionless position of the shock front
c..beta0 = eigenvalue of the problem
c..g0 = heat front scaling parameter

c..output:
c..den = density g/cm**3
c..tev = temperature ev
c..ener = specific internal energy erg/g
c..pres = presssure erg/cm**3
c..vel = velocity cm/sh

c..declare the pass
      double precision rpos,
     1                 aval_in,bval_in,chi0,gamma,bigamma,
     2                 rf,xif_in,xis,beta0_in,g0,
     3                 den,tev,ener,pres,vel

c..local variables
      external rmtvfun,fun,derivs
      integer i,it
      real flag
      double precision twoa,twob,u_0,u_l,u_r,u_c,atemp,btemp,rmtvfun,
     1                 ustar,zeroin,fun,ans,errest,
     2                 xistar,rstar,gstar,hstar,wstar,tstar,
     3                 zeta,time,rs,xiwant,usub2,hsub2,wsub2,tsub2,
     4                 tol,zero,abserr,relerr
      parameter (tol = 1.0d-16, zero = 0.0d0,
     2           abserr = 1.0d-14, relerr = 1.0d-12)

c..for the ode integration
      integer nvar,iwork(1:5),jwork
      parameter (nvar = 4, jwork = 100 + 21*nvar)
      double precision ystart(nvar),ytemp(nvar),eta1,eta2,epsa,epsr,
     1                 xi_end,xi_small,work(jwork)
      parameter (epsr = 4.0d-10, epsa = 4.0d-10, xi_small=1.0d-4)


c..common block communication
      double precision aval,bval,amu,alpha,xif,beta0,kappa,sigma,xgeom
      common /rmtv1/ aval,bval,amu,alpha,xif,beta0,kappa,sigma,xgeom

c..transfer passed values to common
c..xgeom is for spherical coordinates
      aval = aval_in
      bval = bval_in
      xif = xif_in
      beta0 = beta0_in
      xgeom = 3.0d0

c..initialize work arrays
      do i=1,5
        iwork(i) = 0
      end do
```

```
      do i=1,jwork
        work(i) = 0.0d0
      end do

c..frequent factors
      twoa = 2.0d0 * aval
      twob = 2.0d0 * bval
      alpha = (twob - twoa + 1.0d0)/(twob - (xgeom + 2.0d0)*aval +xgeom)
      amu = 2.0d0 / (gamma - 1.0d0)
      kappa = -((twob - 1.0d0)*xgeom + 2.0d0)/(twob - twoa + 1.0d0)
      sigma = (twob - 1.0d0)/(alpha*(1.0d0 - aval))

c..equations 28, 30, 33, 29 of kamm 2000
c..for the scale factor, the phyiscal time, and the shock front position

      zeta = (((0.5d0 * beta0*bigamma**(bval+1.0d0) * g0**(1.0d0-aval) /
     1                chi0)**(1.0d0/(twob - 1.0d0)))/alpha)**alpha
      time = (rf/zeta/xif)**(1.0d0/alpha)
      rs = zeta * 1.0d0 * abs(time)**alpha

c..this section does a root find to obtain the initial conditions
c..bracket the initial zero-value of u
      it = 0
      u_0 = (amu*bval*xif**(-((2.0d0*bval) - 1.0d0 )/alpha)
     1          / beta0)**(1.0d0/bval)
      u_l = 0.0d0
      u_r = 0.5d0
      u_c = 0.5d0 * ( u_l + u_r )

10    continue
      it = it + 1
      atemp = rmtvfun(u_l)
      btemp = rmtvfun(u_r)
      if (atemp * btemp .lt. 0.0d0 ) go to 20
      if (it .ge. 100) stop 'cannot bracket zero in 100 tries'
      u_l = u_l + 0.1d0 * (u_c - u_l)
      u_r = u_r - 0.1d0 * (u_c - u_r)
      go to 10
20    continue

c..root bracketed, solve for the zero-value ustar
      ustar = zeroin(u_l,u_r,rmtvfun,tol)

c..form the converged value of the integral
      call quanc8(fun, zero, ustar, abserr, relerr,
     1                ans, errest, it, flag)

c..equation 11 of kamm 2000 for the position to start the integration from
      xistar = xif * exp(-(beta0 * (xif**((twob - 1.0d0)/alpha))*ans))
      rstar = xistar * zeta * time**alpha

c..equation 11, 13 of kamm 2000 for the initial values of the other functions
      gstar = 1.0d0/(1.0d0 - ustar )
      hstar = xistar**(-sigma) * gstar
      wstar = 0.5d0 * (amu - (( amu + 1.0d0)*ustar))
      tstar = ustar * (1.0d0 - ustar)

c..now integrate
c..beyond the heat front
      if (rpos .gt. rstar) then
        den = g0 * rpos**kappa
        vel = 0.0d0
        ener = 0.0d0
        pres = 0.0d0
        tev = 0.0d0

c..integrate from the heat front to perhaps the shock front
```

```
          else
            ystart(1) = ustar
            ystart(2) = hstar
            ystart(3) = wstar
            ystart(4) = tstar

            xiwant = rpos/zeta/time**alpha
            xi_end = max(xis,xiwant)
            eta1 = log(xistar)
            eta2 = log(xi_end)
            it = 1
            call ode(derivs,nvar,ystart,eta1,eta2,epsr,epsa,it,work,iwork)

c..apply equation 15 of kamm 2000 for the post-shock values if we must integrate
            if (rpos .le. rs) then
              usub2 = ystart(1)
              hsub2 = ystart(2)
              wsub2 = ystart(3)
              tsub2 = ystart(4)

              ystart(1) = 1.0d0 - (tsub2/(1.0d0 - usub2))
              ystart(2) = (1.0d0 - usub2 )**2 / tsub2 * hsub2
              ystart(3) = (tsub2*wsub2 - 0.5d0*((1.0d0-usub2)**4 - tsub2**2)
         &                    / (1.0d0 - usub2)) / (1.0d0 - usub2)**2
              ystart(4) = tsub2

c..and integrate to near the origin if nned be
              eta1 = eta2
              xi_end = max(xi_small,xiwant)
              eta2 = log(xi_end)
              it = 1
              call ode(derivs,nvar,ystart,eta1,eta2,epsr,epsa,it,work,iwork )
            end if

c..convert the integration variables to physical quantities
c..equations 5, 2 of kamm 2000

            vel = alpha * rpos * ystart(1) / time
            den = g0 * rpos**kappa * xi_end**sigma * ystart(2)
            ener = (alpha*rpos/time)**2 * ystart(4) / (gamma - 1.0d0)
            pres = (gamma - 1.0d0) * den * ener
            tev = (alpha*rpos/time)**2 * ystart(4) / bigamma


c..convert from jerk = 1e16 erg, kev = 1e3 ev, sh = 10e-8 s to cgs units
            vel = vel * 1.0d8
            ener = ener * 1.0d16
            pres = pres * 1.0d16
            tev = tev * 1.0d3
          end if

          return
          end

          double precision function rmtvfun(u)
          implicit none
          save

c..evaluates the expression for the initial integral for a root find

c..declare the pass
          double precision u

c..common block communication
          double precision aval,bval,amu,alpha,xif,beta0,kappa,sigma,xgeom
          common /rmtv1/ aval,bval,amu,alpha,xif,beta0,kappa,sigma,xgeom
```

```
c..local variables
      external fun
      integer numfun
      real flag
      double precision fun,zero,abserr,relerr,ans,errest,smallval
      parameter (zero = 0.0d0,
     1                    abserr = 1.0d-14,
     2                    relerr = 1.0d-12,
     3                    smallval = 1.0d-12)

c..standard quadrature routine
      call quanc8 (fun, zero, u, abserr, relerr,
     1                  ans, errest, numfun, flag)

      rmtvfun = log(1.0d0 - smallval) + (beta0 *
     1           (xif**(((2.0d0*bval) - 1.0d0)/alpha))*ans)

      return
      end

      double precision function fun(y)
      implicit none
      save

c.. evaluates the integrand of the initial integral

c..declare the pass
      double precision y

c..common block communication
      double precision aval,bval,amu,alpha,xif,beta0,kappa,sigma,xgeom
      common /rmtv1/ aval,bval,amu,alpha,xif,beta0,kappa,sigma,xgeom

c..equation 12 of kamm 2000

      fun = ((1.0d0 - (2.0d0 * y)) / (amu - ((amu + 1.0d0)*y)))
     1        * (y**(bval - 1.0d0))
     2        * ((1.0d0 - y)**(bval - aval))

      return
      end

      subroutine derivs ( t, y, yp )
      implicit none
      save

c.. evaluates the rhs of the system of odes

c..declare the pass
      double precision t,y(1:4), yp(1:4)

c..common block communication
      double precision aval,bval,amu,alpha,xif,beta0,kappa,sigma,xgeom
      common /rmtv1/ aval,bval,amu,alpha,xif,beta0,kappa,sigma,xgeom

c..local variables
      double precision g1,g2,g3,g4,temp,denom,y1m1,alphainv,omega,
     1                 eps16,eps12
      parameter (eps16 = 1.0d-16, eps12 = 1.0d-12)

c..some factors
      y1m1 = y(1) - 1.0d0

      if (alpha .eq. 0) stop 'alpha = 0 in routine derivs'
      alphainv = 1.0d0 / alpha

      if (abs(y(2)) .le. eps16 .or. abs(y(4)) .le. eps16) then
```

```
          write(6,*) 'derivs: y(2) or y(4) < eps16'
          omega = 1.0d0/eps12 * sign(1.0d0, y(3))* sign(1.0d0, beta0)
     &                * sign(1.0d0,y(2)) * sign(1.0d0, y(4))
        else
          if (abs(aval-1.0d0) .le. eps16) stop 'aval=1 in routine derivs'
          omega = y(3) * y(2)**(1.0d0 - aval) * y(4)**(-bval) / beta0
        end if


c..rhs of original (coupled) ode system
c..equation 26 of kamm 2000

        g1 = sigma - (xgeom + kappa + sigma)*y(1)
        g2 = y(1)*(alphainv - y(1)) + y(4)*(2.0d0*omega-kappa-sigma)
        g3 = omega*(amu*y1m1 + 2.0d0*y(3)) + amu*(alphainv - 1.0d0)
     &          - xgeom*y(1) - (xgeom+kappa+sigma)*y(3)
        g4 = -2.0d0 * (1.0d0 + omega )


c..rhs of uncoupled ode system
c..equations 24, 25 of kamm 2000

        denom = y(4) - y1m1**2
        if (abs(denom) .le. eps16) stop 'denom=0 in routine derivs'
        temp = g2 - (y1m1 * g1)
        yp(1) = g1 - ( y1m1 * temp ) / denom
        yp(2) = y(2) * temp / denom
        yp(3) = g3 - ( yp(1) + y(3) * yp(2) / y(2) )
        yp(4) = y(4) * g4

        return
        end
```

8.5 To the Noh Problem

```
      program noh_test
      implicit none

c..tests the noh solver

c..declare
      character*80 outfile,string
      integer i,nstep,iargc
      double precision time,zpos,
     1                 rho0,vel0,gamma,xgeom,
     2                 den,ener,pres,vel,
     3                 zlo,zhi,zstep,value

c..popular formats
01    format(1x,t4,a,t8,a,t22,a,t36,a,t50,a,t64,a,t78,a,t92,a)
02    format(1x,i4,1p8e12.4)


c..input parameters in cgs
      time = 0.3d0
      rho0 = 1.0d0
      vel0 = -1.0d0
      gamma = 5.0d0/3.0d0
      xgeom = 3.0d0

c..number of grid points, spatial domain, spatial step size
      nstep = 100
      zlo = 0.0d0
      zhi = 1.0d0
      zstep = (zhi - zlo)/float(nstep)

c..output file
      outfile = 'noh_100pt.dat'
      open(unit=2,file=outfile,status='unknown')
      write(2,02) nstep,time
      write(2,01) 'i','x','den','ener','pres','vel'


c..to match hydrocode output, use the mid-cell points
      do i=1,nstep
        zpos = zlo + 0.5d0*zstep + float(i-1)*zstep

        call noh_1d(time,zpos,
     1              rho0,vel0,gamma,xgeom,
     2              den,ener,pres,vel)

        write(2,40) i,zpos,den,ener,pres,vel
40      format(1x,i4,1p8e14.6)

      enddo

c..close up stop
      close(unit=2)
      end

      subroutine noh_1d(time,xpos,
     1                  rho1,u1,gamma,xgeom,
     2                  den,ener,pres,vel)
      implicit none
      save

c..solves the standard case, (as opposed to the singular or vacuum case),
c..constant density (omega = 0) sedov problem in one-dimension.

c..input:
```

```
c..time = temporal point where solution is desired seconds
c..xpos = spatial point where solution is desired cm

c..output:
c..den = density g/cm**3
c..ener = specific internal energy erg/g
c..pres = presssure erg/cm**3
c..vel = velocity cm/sh

c..declare the pass
      double precision time,xpos,
     1                 rho1,u1,gamma,xgeom,
     3                 den,ener,pres,vel

c..local variables
      double precision gamm1,gamp1,gpogm,xgm1,us,r2,rhop,rho2,u2,e2,p2

c..some parameters
      gamm1 = gamma - 1.0d0
      gamp1 = gamma + 1.0d0
      gpogm = gamp1 / gamm1
      xgm1 = xgeom - 1.0d0

c..immediate post-chock values using strong shock relations
c..shock velocity, position, pre- and post-shock density,
c..flow velocity, internal energy, and pressure

      us = 0.5d0 * gamm1 * abs(u1)
      r2 = us * time
      rhop = rho1 * (1.0d0 - (u1*time/r2))**xgm1
      rho2 = rho1 * gpogm**xgeom
      u2 = 0.0d0
      e2 = 0.5d0 * u1**2
      p2 = gamm1 * rho2 * e2

c..if we are farther out than the shock front
      if (xpos .gt. r2) then
        den = rho1 * (1.0d0 - (u1*time/xpos))**xgm1
        vel = u1
        ener = 0.0d0
        pres = 0.0d0

c..if we are between the origin and the shock front
      else
        den = rho2
        vel = u2
        ener = e2
        pres = p2
      end if

      return
      end
```

8.6 To the Sedov Problem

```
      program sedov_test
      implicit none

c..tests the sedov solver
c..declare
      character*80 outfile,string
      integer i,nstep,iargc
      double precision time,zpos,
     1                 eblast,rho0,vel0,ener0,pres0,cs0,gamma,
     2                 xgeom,den,ener,pres,vel,cs,zlo,zhi,zstep,value

c..popular formats
01    format(1x,t4,a,t8,a,t22,a,t36,a,t50,a,t64,a,t78,a,t92,a)
02    format(1x,i4,1p8e12.4)

c..input parameters in cgs
      time = 1.0d0
      eblast = 0.851072d0
      rho0 = 1.0d0
      vel0 = 0.0d0
      ener0 = 0.0d0
      pres0 = 0.0d0
      cs0 = 0.0d0
      gamma = 1.4d0
      xgeom = 3.0d0

c..number of grid points, spatial domain, spatial step size
      nstep = 120
      zlo = 0.0d0
      zhi = 1.2d0
      zstep = (zhi - zlo)/float(nstep)

c..output file
      outfile = 'sed_120pt.dat'
      open(unit=2,file=outfile,status='unknown')
      write(2,02) nstep,time
      write(2,01) 'i','x','den','ener','pres','vel','cs'

c..to match hydrocode output, use the mid-cell points
      do i=1,nstep
        zpos = zlo + 0.5d0*zstep + float(i-1)*zstep

        call sed_1d(time,zpos,
     1              eblast,rho0,vel0,ener0,pres0,cs0,gamma,xgeom,
     2              den,ener,pres,vel,cs)

        write(2,40) i,zpos,den,ener,pres,vel,cs
40      format(1x,i4,1p8e14.6)

      enddo

c..close up stop
      close(unit=2)
      end

      subroutine sed_1d(time,xpos,
     1                  eblast,rho0,vel0,ener0,pres0,cs0,gam_in,xgeom_in,
     2                  den,ener,pres,vel,cs)
      implicit none
      save

c..solves the standard case, (as opposed to the singular or vacuum case),
c..constant density (omega = 0) sedov problem in one-dimension.

c..input:
```

```
c..time = temporal point where solution is desired seconds
c..xpos = spatial point where solution is desired cm
c..eblast = energy of blast erg
c..rho0 = ambient density g/cm**3
c..vel0 = ambient material speed cm/s
c..pres0 = ambient pressure erg/cm**3
c..cs0 = ambient sound speed cm/s
c..gam_in = gamma law equation of state
c..xgeom_in = geometry factor, =3 in spegerical, 2 in cylindircal, 1 in planar

c..output:
c..den = density g/cm**3
c..tev = temperature ev
c..ener = specific internal energy erg/g
c..pres = presssure erg/cm**3
c..vel = velocity cm/sh

c..declare the pass
      double precision time,xpos,
     1                 eblast,rho0,vel0,ener0,pres0,cs0,gam_in,xgeom_in,
     3                 den,ener,pres,vel,cs

c..local variables
      external midpnt,midpowl,efun01,efun02,sedr0
      integer i
      double precision v0,vs,midpnt,midpowl,efun01,efun02,eval1,eval2,
     1                 alpha,endim,vv,vstep,us,u2,rho2,p2,e2,cs2,
     3                 zeroin,sedr0,vat,l_fun,f_fun,g_fun,h_fun,
     4                 xgeom_old,gamma_old,eps,eps2,pi
      parameter (eps = 1.0d-12, eps2=1.0d-16,
     1           pi = 3.141592653589793238 4d0)

c..common block communication
      double precision gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1                 a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val
      common /slap/ gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1              a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val

c..common block communication with the integration stepper
      double precision gam_int
      common /cmidp/ gam_int

c..initialize the old values
      data xgeom_old/-1.0d0/, gamma_old/-1.0d0/

c..set some parameters
      rwant = xpos
      gamma = gam_in
      gamm1 = gamma - 1.0d0
      gamp1 = gamma + 1.0d0
      gpogm = gamp1 / gamm1
      xgeom = xgeom_in
      xg2 = xgeom + 2.0d0

c..various exponents (kamm 2000, equations 42-47 with omega = 0)
      a0 = 2.0d0/xg2
      a2 = -gamm1/(2.0d0 * gamm1 + xgeom)
      a1 = xg2*gamma/(2.0d0 + xgeom*gamm1) *
     1          (((2.0d0 * xgeom * (2.0d0 - gamma))/(gamma * xg2**2)) - a2)
      a3 = xgeom / (2.0d0 * gamm1 + xgeom)
      a4 = xg2 * a1 / (2.0d0 - gamma)
      a5 = -2.0d0/(2.0d0 - gamma)

c..frequent combinations (kamm 2000, equations 33-37 with omega = 0)
      a_val = 0.25d0 * xg2 * gamp1
      b_val = gpogm
      c_val = 0.5d0 * xg2 * gamma
```

```
      d_val = (xg2 * gamp1)/(xg2*gamp1 - 2.0d0*(2.0d0 + xgeom*gamm1))
      e_val = 0.5d0 * (2.0d0 + xgeom * gamm1)

c..dimensionless origin and shock positions
      v0 = 2.0d0 / (xg2 * gamma)
      vs = 4.0d0 / (xg2 * gamp1)



c..two energy integrals and the dimensionless energy
c..lower limit of the second energy integral
c..has an integrable power-law singularity.
c..transfer that power-law exponent into gam_int
c..integration still expensive, only do it if things have changed

      if (xgeom .ne. xgeom_old .or. gamma .ne. gamma_old) then

        call qromo(efun01,v0,vs,eps,eval1,midpnt)

        gam_int = abs(a3 - a2*xg2 - 2.0d0)
        call qromo(efun02,v0,vs,eps,eval2,midpowl)

        xgeom_old = xgeom
        gamma_old = gamma
      end if

c..kamm equation 85 for and 66
      alpha = (xgeom - 1.0d0) * pi * (eval1 + 2.0d0 * eval2/gamm1)
      endim = eblast / alpha

c..immediate post-shock values
c..shock position, shock speed, material speed, density,
c..pressure, specific internal energy, and sound speed

      r2 = (endim/rho0)**(1.0d0/xg2) * time**(2.0d0/xg2)
      us = (2.0d0/xg2) * r2 / time
      u2 = 2.0d0 * us / gamp1
      rho2 = gpogm * rho0
      p2 = 2.0d0 * rho0 * us**2 / gamp1
      e2 = p2/gamm1/rho2
      cs2 = sqrt(gamma*p2/rho2)

c..if we are farther out than the shock front
      if (rwant .gt. r2) then
        den = rho0
        vel = vel0
        pres = pres0
        ener = ener0
        cs = cs0

c..if we are between the origin and the shock front
      else

c..locate the similarity value v that correlates to rwant
        vat = zeroin(0.9d0*v0,vs,sedr0,eps2)

c..with the converged value, get the sedov functions
        call get_fun0(vat,l_fun,f_fun,g_fun,h_fun)

c..density, speed, pressure, energy, and sound speed
        den = rho2 * g_fun
        vel = u2 * f_fun
        pres = p2 * h_fun
        ener = pres / gamm1 / den
        cs = sqrt (gamma * pres/den)
      end if
      return
```

```
      end

      double precision function efun01(v)
      implicit none
      save

c..evaluates the first energy integral for constant density
c..equation 67 and 73 of kamm 2000

c..declare the pass
      double precision v


c..common block communication
      double precision gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1                 a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val
      common /slap/ gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1                 a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val

c..go
      efun01 = -gpogm * v**2 *
     &         (a0/v + (a2*c_val/ (c_val*v - 1.0d0))
     &             - (a1*e_val/(1.0d0 - e_val*v))) *
     &         ((a_val*v)**a0 *
     &          (b_val * (c_val*v - 1.0d0))**a2 *
     &          (d_val * (1.0d0 - e_val*v))**a1)**(-xg2) *
     &         (b_val * (c_val*v - 1.0d0))**a3 *
     &         (d_val * (1.0d0 - e_val*v))**a4 *
     &         (b_val * (1.0d0 - c_val*v/gamma))**a5
      return
      end

      double precision function efun02(v)
      implicit none
      save

c..evaluates the second energy integrand for constant density (omega=0)
c..equation 67 and 74 of kamm 2000

c..integrand has an integrable power-law singularity at the lower limit
c..of integration of the form (c_val*v - 1.0d0)**(a3 - a2*xg2 - 2.0d0)
c..where the exponent is negative.

c..declare the pass
      double precision v

c..common block communication
      double precision gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1                 a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val
      common /slap/ gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1                 a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val

c..go
      efun02 = -0.5d0 * gamp1/gamma * v**2 *
     1         ((gamma - c_val*v)/(c_val*v - 1.0d0)) *
     &         (a0/v + (a2*c_val/ (c_val*v - 1.0d0))
     &             - (a1*e_val/(1.0d0 - e_val*v))) *
     &         ((a_val*v)**a0 *
     &          (b_val * (c_val*v - 1.0d0))**a2 *
     &          (d_val * (1.0d0 - e_val*v))**a1)**(-xg2) *
     &         (b_val * (c_val*v - 1.0d0))**a3 *
     &         (d_val * (1.0d0 - e_val*v))**a4 *
     &         (b_val * (1.0d0 - c_val*v/gamma))**a5
      return
      end

      double precision function sedr0(v)
```

```
      implicit none
      save

c..find the similarity parameter foir an exact radius value

c..declare the pass
      double precision v

c..common block communication
      double precision gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1                 a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val
      common /slap/ gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1                 a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val


c..local variables
      double precision l_fun, f_fun, g_fun,h_fun
      call get_fun0(v,l_fun,f_fun,g_fun,h_fun)
      sedr0 = r2*l_fun - rwant
      return
      end

      subroutine get_fun0(v,l_fun,f_fun,g_fun,h_fun)
      implicit none
      save

c..computes the functions lambda, f, g & h in the constant density case
c..kamm 200 equation 38-41

c..declare the pass
      double precision v,l_fun,f_fun,g_fun,h_fun

c..common block communication
      double precision gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1                 a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val
      common /slap/ gamma,gamm1,gamp1,gpogm,xgeom,xg2,rwant,r2,
     1                 a0,a1,a2,a3,a4,a5,a_val,b_val,c_val,d_val,e_val

c..local variables
      double precision eps
      parameter (eps = 1.0d-30)

c..go
      l_fun = (a_val*v)**(-a0)
      l_fun = l_fun * (b_val * max(eps, c_val*v - 1.0d0))**(-a2)
      l_fun = l_fun * (d_val * (1.0d0 - e_val*v))**(-a1)

      f_fun = a_val * v * l_fun

      g_fun = (b_val * max(eps, c_val*v - 1.0d0))**a3
      g_fun = g_fun * (b_val * (1.0d0 - 0.5d0*xg2*v))**a5
      g_fun = g_fun * (d_val * (1.0d0 - e_val*v))**a4

      h_fun = (a_val*v)**(a0*xgeom)
      h_fun = h_fun * (b_val * (1.0d0 - 0.5d0*xg2*v))**(1.0d0 + a5)
      h_fun = h_fun * (d_val * (1.0d0 - e_val*v))**(a4 - 2.0d0 * a1)
      return
      end

      subroutine midpowl(funk,aa,bb,s,n)
      implicit none
      save

c..this routine is an exact replacement for midpnt, except that it allows for
c..an integrable power-law singularity at the lower limit aa.

c..declare
```

```
      external funk
      integer n,it,j
      double precision func,funk,a,aa,b,bb,s,tnm,del,ddel,x,sum

c..common block communication
      double precision gam_int
      common /cmidp/ gam_int

c..a little conversion, recipe equation 4.4.3
      func(x) = 1.0d0/(1.0d0 - gam_int) * x**(gam_int/(1.0d0 - gam_int))
     &              * funk(x**(1.0d0/(1.0d0 - gam_int)) + aa)
      b = (bb - aa)**(1.0d0 - gam_int)
      a = 0.0d0

c..now exactly as midpnt
      if (n .eq. 1) then
        s = (b-a) * func(0.5d0*(a+b))
      else
        it = 3**(n-2)
        tnm = it
        del = (b-a)/(3.0d0*tnm)
        ddel = del + del
        x = a + (0.5d0 * del)
        sum = 0.0d0
        do j=1,it
          sum = sum + func(x)
          x = x + ddel
          sum = sum + func(x)
          x = x + del
        enddo
        s = (s + ((b-a) * sum/tnm)) / 3.0d0
      end if
      return
      end
```