Imogen User's Manual

Erik Keever

January 10, 2017

1 Setup

This section will guide you through the Imogen acquisition and setup process.

The setup process consists in making sure that you have the proper libraries available and informing Imogen of where they are through the Make.config file in the top level directory. The template Make.config.default should be copied to Make.config and then setup per instructions below.

1.1 Download

Imogen lives online at `https://github.com/imogenproject/gpuImogen`

To download it into directory *./gpuimogen* from the public repo,

git clone https://github.com/imogenproject/gpuImogen.git ./gpuimogen

1.2 Matlab

Imogen runs in Matlab and so obviously Matlab is a prerequisite to have. A standard Matlab installation will have everything you need - the parallel computing toolkit is not needed.

The *MATLAB_DIR* variable in Make.config must be set to the root of the Matlab installation (e.g. /opt/Matlab/R2013b). Running Imogen in a different Matlab version than used here to compile the binary modules is less likely than might be imagined to cause problems.

1.3 MPI

MPI, even if parallel processing is not used, must be available (Imogen still uses MPI to check that it is running serially), including development headers to compile the Matlab mex files.

In Make.config, *MPI_DIR* must refer to the directory containing the include/ folder that holds the mpi.h file, i.e. the –prefix specified when MPI was installed.

1.4 Parallel Gateway

Imogen uses the Parallel Gateway (PGW) library as a go-between with MPI.

Make.config's *PGW_DIR* variable must be set to the install directory containing PGW.

1.5 CUDA

Imogen's GPU routines are written in CUDA and access to cuda 4.0 or later libraries is required. The runtime alone isn't sufficient, the SDK is needed to provide the headers.

Once it is installed set *CUDA_DIR* to the install location.

Other variables to be set are

- *CUDA_LDIR* - either 'lib' on 32bit or 'lib64' on 64bit machines
- *NVARCH* - *compute_20* for Fermi, *compute_30* for Kepler
- *NVCODE* - *sm_20* for Fermi, *sm_30* for Kepler

NVARCH and NVCODE are further explained by the nvcc manual page.

1.6 Compiling

Once all the above variables are set, run *make* in both the mpi/ and gpuclass/ directories. Both build in parallel so feel free to use -j.

1.7 Filesystem

Imogen will open/create the directory /Results. It is not required that this be a shared filesystem.

1.8 Cluster access

Imogen invokes itself in parallel with the './imogen cluster ...' format.

This takes place on lines 95 to 124 of the run/imogen file. Before running in parallel on a cluster it would be a good idea to examine this section since it will almost certainly need to be amended, if only to set the package names to their local values.

1.9 Troubleshooting

It is very important that PGW, MPI and the CUDA modules all be compiled using the same compiler. PGW in particular is insistent about the version of libfortran it accesses.

Unfortunately this is growing progressively more difficult as CUDA requires newer GCCs and Mathworks' MEX compiler-wrapper lags.

If Imogen fails because Matlab complains about being unable to resolve an MPI symbol, the problem is that Matlab defaults to lazy symbol resolution when loading dynamic libraries and apparently something about the MPI lib trips up on this. The solution:

- LD_PRELOAD="$MPIDIR/lib/libmpi.so"

Putting this in your .bashrc (or as sysadmin, in the system's /etc/profile.d) will make it go away permanently.

2 Invocation

Imogen is run through the script run/imogen

There are three separate general types of run, all of which end up doing "matlab -r":

- serial - One process uses ones GPU; 'imogen' calls Matlab.
- parallel - N processes select N GPUs; 'imogen' calls mpirun calls Matlab
- cluster - N processes on M nodes; 'imogen' writes a script for qsub that calls Matlab

Serial and parallel are for small size simulations that work on a single SMP system, while the cluster option uses qsub to fire jobs onto large systems.

The invocation syntaxes (also available through ./imogen -help) is as follows:

- ./imogen serial runfile.m [stream# [GPU#]];
- ./imogen runfile.m; A shortcut that assumes serial, stream 0 and GPU 0.
- ./imogen parallel runfile.m [stream#] [NP]

3 Selftest

This is not yet complete – Erik

Imogen includes a comprehensive full-simulation test, the simulation *run_FullTestSuite.m*. This "simulation" will run dozens of test cases for which an exact solution (or reasonable facimile thereof) is available and determine whether Imogen is going them right and, where analytic answers are available, provide convergence orders.

Note that there are a lot of tests to run, and this test falls under the "go have lunch then check back" class.

4 Physics solved by the Imogen simulation engine

4.1 Full Equation Set

The broadest writing of the equations Imogen is capable of solving:

$$
\begin{aligned}
\frac{\partial \rho}{\partial t} + \nabla(\rho v) &= 0 \\
\frac{\partial \rho \vec{v}}{\partial t} + \nabla_i\left(\rho \vec{v} v_i + \mathbb{I} P\right) &= -\rho \nabla \phi_g & -\sum_k n_{d,k} F_{drag,k} \\
\frac{\partial E_{tot}}{\partial t} + \nabla v(E_{tot} + P) &= \rho v \cdot \nabla \phi_g - \Gamma_0 \rho^2 T^\theta & -\sum_k n_{d,k} v_{d,k} \cdot F_{drag,k} \\
\frac{\partial \rho_{d,k}}{\partial t} + \nabla(\rho_{d,k} v_{d,k}) &= 0 \\
\frac{\partial \rho_{d,k} v_{d,k}}{\partial t} + \nabla(\rho_{d,k} v_{d,k} v_{d,k}) &= n_{d,k} F_{drag,k} \\
\frac{\partial T_{d,k}}{\partial_t} + \nabla(v_{d,k} T_{d,k}) &= & heat + radi
\end{aligned}
$$

Where here k denotes an arbitrary number of dust fluids.

With auxiliary equations

$$
\begin{aligned}
\nabla^2 \phi_{self} &= 4\pi G \rho \\
\phi_g &= \phi_{self} + \phi_{static} + \sum_i \frac{-GM_i}{|x - x_i|} \\
F_{drag} &= \frac{s_0^2 F_{epstein} + s^2 F_{stokes}}{s_0^2 + s^2} \\
F_{epstein} &\approx \frac{-4\pi}{3} \rho_{gas} s^2 \sqrt{\frac{8}{\gamma\pi}} \sqrt{c_s^2 + 9\pi |dv|^2/128} \vec{dv} \\
F_{stokes} &= -.5 C_d \pi s^2 \rho_{gas} |dv| \vec{dv}
\end{aligned}
$$

In the above, G is the gravitational coupling constant and ϕ_{static} represents a stationary potential field defined at simulation start time. F_{drag} generates a smooth interpolation between the rarefied gasdynamics regime experienced by a particle much smaller than the mean free path, $F_{epstein}$, and the viscous drag on a particle in the continuum limit given by F_{stokes}. $F_{epstein}$ as given represents the approximation given by Kwok (1975) for the exact Epstein drag, due to the high cost of evaluating multiple transcendental functions occurring in it.

The Euler equations, unadorned, are self-similar and lack any characteristic scale. Activating any of the non-ideal physics in Imogen introduces scales and requires choosing appropriate scaling constants.

Conservative transport terms are in black. Gravitation related terms are in red. Optically thin radiation terms are in green. Dust terms are in blue.

Self gravity, stationary potential, and compact objects are enableable separately. Activating any of them requires choosing a gravitational coupling constant G.

Activating radiative emission requires choosing a radiation scaling constant Γ_0. The code computes as written, using the square of *mass density*. Because we assume an ideal gas law, we are actually considering

$$\Gamma' n^2 (P/\rho)^\theta$$

with $\Gamma' = \Gamma_0 \mu^{\theta-2} K_b^{-\theta}$.

Imogen contains the following unitful objects which must be defined if the relevant physics are in use.

- G - Gravitational coupling constant
- K_b - Boltzmann constant
- μ - Gas mean molecular weight
- m_k - Mass of dust particles of type k
- s_k - Size of dust particles of type k

There are four fundamental unitful quantities in Imogen: mass, length, time, and temperature.

As an example of how they might be chosen: Kojima disks are defined with G=1 and M* = 1. The orbital speed at a radius of 1 is defined as 1. By using $p = \rho^\gamma$ as the equation of state, implicitly k_b/μ is chosen.

4.2 CFD

At its core Imogen solves the Euler equations, written here in conservative form:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i} \rho v_i &= 0 \\ \frac{\partial (\rho v_i)}{\partial t} + \frac{\partial}{\partial x_j} (T_{ij} + \mathbb{I} P) &= 0 \\ \frac{\partial E}{\partial t} + \frac{\partial}{\partial x_i} v_i (E + P) &= 0 \end{aligned}$$

with mass density ρ, fluid bulk velocity v_i, momentum stress tensor $T_{ij} = \rho v_i v_j$, identity tensor $\mathbb{I}$, scalar thermal pressure P, and total energy density $E = \frac{1}{2}\rho v^2 + \epsilon$.

These are closed by the equation of state $P(\epsilon)$. Imogen implements the adiabatic ideal gas equation of state, $P = (\gamma - 1)\epsilon$ for $\gamma > 1$ (physical gas values are $1 < \gamma \leq 5/3$).

With the assumption of a gamma law, the energy flux is written into the code as $v_i \times (\frac{1}{2}\rho v^2 + \frac{\gamma}{\gamma-1} P)$ with P calculated before fluxing.

4.2.1 Operator splitting

Imogen uses a dimensionally split flux solving scheme. Assuming a splittable Hamiltonian exists in the form

$$\mathbf{H} = \mathbf{E} + \mathbf{F} + \mathbf{G}$$

where in our case, E, F, and G represent the fluid fluxes in 3 non-degenerate directions (normally taken as 3 orthogonal directions in a curvilinear coordinate system: Here, fluxes in the x, y, and z directions), an approximate scheme of the form

$$e^{\epsilon\mathbf{H}} = e^{\epsilon\mathbf{E}} e^{\epsilon\mathbf{F}} e^{\epsilon\mathbf{G}} + \mathbb{O}(\epsilon^2)$$

always exists, and the reader may easily verify that the local error associated with doing this with two operators equals their commutator times ϵ^2.

Strang [reference!] showed that by performing such a first order step, then applying the same operators in exactly reversed order, will cancel all the first-order errors. This Strang splitting provides an easy way of constructing second spatial order solvers.

Denoting the operators that solve the inviscid flux equations in the three directions as $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$, there are 6 ways to order them which are the elements of the permutation group of $\{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$. In a two-dimensional simulation, there are only two orderings ($XYYX$ and $YXXY$).

Imogen cycles through them iteration to iteration, the idea being that this will average away (or at least partly so) any systematic errors associated with a given ordering.

There are additional operators associated with MHD (magnetic fluxing operators), non-inertial frames, radiation, and scalar potental fields (fixed potentials, point objects or self-gravitation).

The MHD algorithm has major problems with low-beta plasma and seems to be subtly broken at present.

4.3 MHD

Imogen can evolve magnetic fields in the limit of Ideal MHD where the Euler equations become

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial x_i}\rho v_i \\
\frac{\partial (\rho v_i)}{\partial t} &= -\frac{\partial}{\partial x_j}(T_{ij} + \mathbb{I}(P + B^2/2) \\
\frac{\partial E}{\partial t} &= -\frac{\partial}{\partial x_i} v_i(E + P) - B_i(v_j B_j) \\
\frac{\partial B_i}{\partial t} &= \frac{\partial}{\partial x_i}(v_i B_j - B_i v_j)
\end{aligned}$$

now with the stress tensor $T_{ij} = \rho v_i v_j - B_i B_j$ and under the requirement $\nabla \mathbf{B} = 0$.

There is no computational requirement that the $\mathbf{B}$ field be initialized this way, however Imogen solves no advection equation for $\mathbf{B}$-charge and any divergence introduced will simply stay there.

4.4 Frame rotation

In many cases a simulation is desired of an object that exhibits coherent rotation at high speeds.

Imogen has the Eulerian explicit CFL constraint is set by

$$dt < dx(|v_i| + c_s)$$

and so if v_i can be significantly reduced by setting the grid to co-rotate with a flow, the timestep can be significantly increased for advection-dominated flows.

The non-inertial sourcing operator introduces its own errors and it is up to the user to determine the point at which the change is worth it.

Imogen checks for a rotating frame at simulation initialization and performs the transformation to the noninertial frame for you - simulations should be initialized in the lab inertial frame.

4.5 Radiation

Imogen can solve optically thin radiation with power law cooling,

(power law lambda)

Programming in different cooling laws is extremely simple as a test by making additional entries to the experiment/Radiation.m file, templatted after the thin power law radiation one. Of course, don't expect them to be as fast as the GPU accelerated routines.

While the power law cooling is reliable enough in and of itself, it has exhibited a remarkable ability to derange Imogen's fluid solver routines. None of them has exhibited the ability to handle an extremely slowly-moving radiating shock front (i.e. the motion associated with the finite accuracy of the simulation) without some measure of oscillatory density error at the adiabatic jump.

(Post pics of evil radiative shocks here)

4.6 Gravity

4.6.1 Point gravity

Imogen allows the addition of massive points to the fluid simulation which obey Newton's laws. Both point-point and point-fluid gravitational force are computed.

These points, "compact objects" as Imogen calls them, are used as standins for stars or planets which cannot be resolved at reasonably available levels of resolution.

Far away from the object, they behave as point masses. Within a distance defined as its 'radius', matter is considered as having been accreted. Each step, the mass and linear and angular momentum in cells whose center is within r of a compact mass are added to the compact object and replaced with grid's vaccuum values instead.

NOTE: This is not the desired behavior in some cases - e.g. a planet would require a surface pressure, not a surface vaccuum, in order to be embedded in the disk.

Point does not feel fluid gravity presently.

4.6.2 Self gravity

This is basically dead for now, especially in parallel.

The serial solver could easily enough be resurrected but there's no way I'm going to have time to setup a parallel Poisson solver, let alone shovel it onto the GPU.

5 Physics solvers in the initializers

Many fluid and physics subproblems come up when writing initial condition generators for Imogen. Some of these have convenient self-contained solvers available within Imogen.

The available units include:

- `J = HDJumpSolver(Mach, γ, θ)`: Returns a structure J describing the only physical shock solution to a hydrodynamic shock of strength $Mach$ in a fluid of adiabatic index γ and preshock shock-normal flow inclination θ (in degrees).
- `J = MHDJumpSolver(Mach, Alfven Mach, γ, θ)`: Solves the MHD jump equations for the slowest shock permitted.
- `F = fim(X, @f(X))`: Short for Fourier Integral Method, calculates the integral of anonymous function f on the presumed regularly spaced points X by calculating a polynomial that renders f 3 times continuous at the endpoints, integrating the polynomial directly and the residual by Fourier transform.
- `The ICM`: `icm.m` is currently very much a prototype and not ready for normal use. It is short for Integral Constraints Method.
- `t = pade(c_n, a, b, x)`: Given a polynomial $\sum_n c_n x^n$, calculates the (a,b) Pade approximant P(a)/Q(b) of the polynomial and evaluates it at x, for any $a + b \leq 6$.
- RadiatingFlowSolver class: Solves the equations of a power-law-radiating hydrodynamic or magnetohydrodynamic flow. It initializes itself using the 4th (MHD) or 6th (HD) order power series solution of the flow and then integrates using a 6th order implicit LMM, restarting with reduced step size when it encounters sharp flow features (knees).
- Linear wave eigenvectors: The function eigenvectorEntropy, eigenvectorAlfven, eigenvectorSonic, eigenevectorMA(rho, csq, v, b, k, wavetype) return the linear wave eigenvectors associated with the flows with the given uniform density, thermal acoustic soundspeed, velocity, magnetic field and wavevector. Alfven and sonic waves have wavetype +1 for a wave which advances towards **k** and -1 for against **k**. The MA wavetype can have either sign, with magnitude 2 for a fast MS wave and magnitude 1 for a slow MS wave. If a nonzero B is passed to a sonic wave, it will invoke the fast MS solver (as the fast wave is the one connected with the nonmagnetized sonic wave).
- The bow shock problem, in two-dimensional testing of a source with outflow, has acquired a routine analyticalOutflow() which solves the exact adiabatic expansion of a

$\gamma = 5/3$ gas away from a cylinder with the inner boundary condition of $v_{radial} = v_0$ at $r = r_0$.

6 Programming Experiments

6.1 Common parameters of all experiments

All experiment initializers should inherit from the `Initializer` class and share its not inconsiderable breadth of parameters. Among the more important names are

- `cfl` coefficient by which the CFL-limit timestep is scaled. **WARNING IMOGEN DOES NOT SANITY CHECK THIS**
- `gamma` - The adiabatic index of the fluid
- `grid` - The [nx ny nz] grid resolution of the *global* domain. This should normally be set when creating the initializer and not messed with again.
- `iterMax` - the maximum number of timesteps to take
- `useInSituAnalysis` - If nonzero, Imogen uses the given in-situ analysis tool
- `inSituHandle` - @AnalyzerClass which must meet the requirements of an in-situ analyzer (see REF)
- `stepsPerInSitu` - Number of timesteps between calls to the analyzer
- `frameRotateOmega` - If nonzero, Imogen places the simulation into a rotating frame and applies appropriate source terms. **Simulation is still initialized in lab frame**
- `frameRotateCenter` - If frame is rotating, this is the center (**in cells**) about which the frame rotates on the (x, y) plane.

6.2 How to setup frame rotation

To turn on the rotating frame during start, simply set a nonzero value for the `frameRotateOmega` parameter, and assign a 2-element array $[x_0, y_0]$ to `frameRotateCenter` and Imogen will automatically transform the fluid momentum field into the noninertial frame at start.

Caution: x_0 and y_0 need to be in cell coordinates... yes it's dumb.

The rotation rate can be altered during simulation time using `alterFrameRotation(run, mass, ener, mom, newOmega)` in the user's in-situ analysis function. This has no physical effect (outside truncation error) and will not e.g. generate the Euler term, it simply alters the rate at which the grid traverses the fluid in the lab frame. Note also that the alterFrameRotation function is not GPU accelerated and should thus be called rarely.

It is not acceptable to simply alter `run.frameRotateOmega` during the run. This will alter the behavior of the sourcing function, however it will NOT update the *frame-dependent* kinetic energy density. Among other deleterious effects, increasing the rotation rate in this manner is therefore likely to cause negative pressure and crash the simulation.

6.3 How to setup radiation

6.4 How to setup a potential field

6.5 How to set up a compact object

6.6 How to setup in-situ analysis

Recognizing that it is not practical to save the entire simulation output for post-hoc analysis in most cases, Imogen allows the construction of in-situ analyzers. These are classes that must have two public members:

- `.FrameAnalyzer(mass, ener, mom, mag, run)`
- `.finish(run)`

They must be set at initialization time with these three parameters in the initializer:

- `useInSituAnalysis` - Set to nonzero to enable
- `inSituHandle` - @Class which must meet the requirements of an in-situ analyzer (see above)
- `stepsPerInSitu` - Number of timesteps between calls to the .FrameAnalyzer() function

Imogen calls the `.finish` method after exiting the CFD iteration loop. At this point the ISA needs to write to disk any data not already written because `imogen()` is about to return.

7 Experiments

7.1 Advection Test

The advection test places a single characteristic wave (sonic or entropic) onto a grid.

The fluid is time-evolved, and for saved data for which the critical time t_* has not been exceeded the solution is compared with an analytic solution from a characteristic analysis.

7.1.1 Initial Conditions

7.1.2 Analysis

Refs:

7.2 Bow Shock Test

7.2.1 Initial Conditions

7.2.2 Analysis

...

7.3 Centrifuge Test

The centrifuge test checks the ability of the code to evolve an analytically time-independent hydrodynamic flow and maintain its original time-independent values, and also provides a valuable test of the rotating frame's implementation.

7.3.1 Initial Conditions

The physical input parameters to the centifuge test are:

- `rho_0` specifies the density at either the center (Isochoric) or edge (isothermal or adiabatic) of the simulation region
- `P_0` specifies the pressure at the same points. The isothermal c_s and adiabatic k are defined through the pressure.
- `omegaCurve` is an anonymous function of an array r which gives $\omega(r)$. This is assumed to obey the constraints given. r is normalized between 0 (center) and 1 (edge).
- `eqnOfState` must be symbolically set to one of the `CentrifugeInitializer class' symbolic values EOS_ISOTHERMAL, EOS_ISOCHORIC or EOS_ADIABATIC.`

Two additional numeric parameters are

- `edgeFraction` - Sets how far the simulation domain extends past the edge of the rotating region, as a multiple of the radius of the centrifuge: $2(1 + e) = Nx$
- `frameParameters.omega` - Sets the rate at which the simulation frame is rotated. This can improve the timestep considerably if rotation is in one direction and advection speed is high. In fluid velocities, positive ω about $\hat{z}$ is such that particles on $+\hat{x}$ have momentum $+\hat{y}$. The fluid frame is opposite, such that positive `frameParameters.omega` will create the appearance of points on $\hat{x}$ swinging towards $-\hat{y}$.

7.3.2 Analysis

The centrifuge test balances centripetal acceleration against a pressure gradient,

$$\rho r\omega(r)^2 = dP/dr$$

Under the ideal gas law,

$$P = K\rho T$$

and assuming that we are dealing with a thermodynamically simple gas or mixture whose internal degrees of freedom are pleasantly unchanging,

$$\rho r\omega(r)^2 = KTd\rho/dr + K\rho dT/dr$$

Two of ω, ρ and T have to be specified, then our differential equation plus boundary conditions defines the third.

We chose to define an arbitrary ω, except for the requirement that velocity behave itself at the origin (ω diverging slower than $1/r$) and with the awareness that ω will only be evaluated on a compact region normalized to $[0, 1]$, outside of which it is set to zero (in other words, fluid at rest at infinity).

A pressure-supported structure can't be isobaric but isothermal, isochoric and adiabatic equilibria can all be defined for any sane initial conditions. The ODE is solved by separation of variables; Because it occurs often, the quantity

$$\int_{r_0}^{r} r\omega(r)^2 dr \equiv \Phi(r, r0)$$

to save space.

Isothermal With a fixed temperature (represented in the code as a fixed isothermal soundspeed), the ODE

$$\Phi(r, r_0) = KT \int_{\rho_0}^{\rho} d\rho/\rho$$

is separated and has solution

$$\rho(r) = \rho_0 e^{\Phi(r,r_0)/KT}$$

With ρ_0 specified at the outer edge radius r_0 and an isothermal soundspeed KT, a physical solution exists for sane inputs.

Isochoric At a fixed volume (here taken as fixed density), the ODE

$$\Phi(r, r_0) = K \int_{T_0}^{T} dT$$

is separated and has solution

$$T(r) = (a^2 + \Phi(r, r0))/K$$

which gives pressure

$$P = \rho KT = \rho_0(a^2 + \Phi(r, r0))$$

With the initial temperature set by proxy by isothermal soundspeed a at the center and density fixed, the temperature ramps up as required and a solution exists for sane inputs.

Adiabatic Under adiabatic conditions we use the relation $P = K\rho^\gamma$ and so

$$\frac{\Phi(r, r_0)}{K\gamma} = \int_{\rho_0}^{\rho} \rho^{\gamma-2} d\rho$$

with solution

$$\rho(r) = \left[\rho_0^{\gamma-1} + \frac{(\gamma - 1)\Phi(r, r_0)}{K\gamma}\right]^{1/(\gamma-1)}$$

Defining ρ_0 at r_0 and given K, a solution exists for all sane inputs; Imogen defines it at the outer edge.

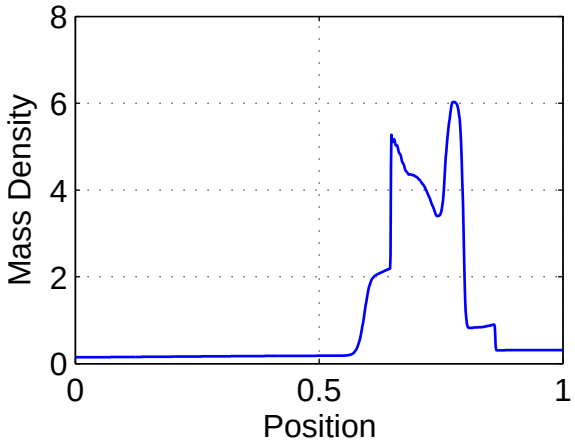

Figure 1: Double Blast at t = 0.038

7.4 Double Blast Test

The double blast test was introduced by Woodward & Colella in 1984. It is a one-dimensional test of two strongly interacting shock fronts. Except for very early times, it does not have an analytic solution & convergence is usually measured by comparison to a very highly refined grid.

7.4.1 Initial conditions

The domain consists of a grid $x \in [0, 1]$. The initial conditions for the test are:

- $\rho(x) = 1$
- $\vec{v}(x) = \vec{0}$
- $P(x < 0.1) = 100$
- $P(0.1 \leq x \leq 0.9) = .01$
- $P(x > 0.9) = 1000$
- Gamma-law gas, $\gamma = 7/5$

Two extremely strong shocks are launched towards the center. Repeated collisions between shocks and contact discontinuities generate an extremely complex flow with very fine details which is extremely difficult to reproduce on an Eulerian grid without immense levels of refinement.

7.4.2 Analysis

7.4.3 Initial Conditions

Boundary conditions on both ends of the domain are mirror.

Imogen's runfile parameters are

- `pLeft - Sets the initial pressure in the left tenth of the grid`
- `pMid - Sets the ambient pressure`
- `pRight - Similarly sets the pressure in the right tenth of the grid`

The default values are those of Woodward/Colella, `pLeft = 100`, `pMid = .01` and `pRight = 1000`.

Further reading: Woodward/Colella 1984

7.5 Einfeldt Strong Rarefaction Test

This test measures the solver's vulnerability to very strong rarefactions that can, in some cases, produce negative mass densities or pressures. The input parameters can be tweaked slowly to determine exactly how strong the rarefaction can become before producing values that are NAN.

The test was devised by Einfeldt to demonstrate the failure modes of many approximate Riemann solvers used in Godunov methods.

7.5.1 Initial Conditions

Einfeldt's paper considers states lying in a 4-parameter region $\{\rho, m, n, e\}$, with the one-dimensional domain separated into $x > 0$ and $x < 0$. Imogen sets the box size to 1.

Initial conditions in Einfeldt's notation are:

- $\rho(x) = 1$
- $\vec{v} = \{sign(x)m, n, 0\}$
- $E_{\text{tot}} = e$

Einfeldt's original test is conducted for an adiabatic equation of state with index $\gamma = 7/5$.

Boundary conditions are constant for X.

Einfeldt's notation permits the entry of manifestly unphysical conditions (total energy ¡ kinetic energy) and care is therefore required.

Imogen's physical input parameters to the Einfeldt Strong Rarefaction test are:

- `rho - Defines the mass density of the whole region`
- `m - Defines the X-momentum of the region (parallel to the grid)`
- `n - Defines the Y-momentum of the region (perpendicular to the grid)`
- `e - Defines the energy density of the region`

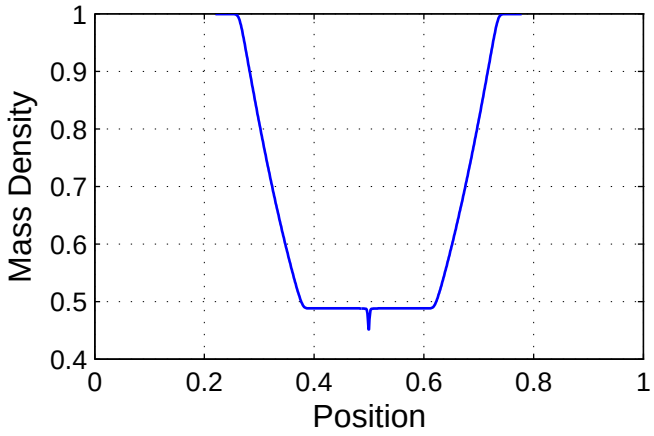

Figure 2: Einfeldt test at t = 0.1

7.5.2 Analysis

The evolution results in two rarefaction characteristics being emitted from the center of the grid, and potentially a stationary region in the center.

For weak rarefactions, the solution contains six clearly defined regions:

- $x < -c_s t$ - farther left than leftgoing rarefaction fan
- left rarefaction
- left stationary region
- right stationary region (separated by material contact)
- right rarefaction fan
- $x > c_s t$ - farther right than rightgoing rarefaction fan

As the Mach increases, the tails of the rarefaction fans move more and more slowly. At a critical Mach number M_* the middle stationary region vanishes. Above it, the velocity profile becomes simple ($v = x/t, |x| < c_s t$) but the density profile has proven difficult to solve (despite apparent simplicity).

7.6 Hachisu (HSCF) Disk Simulation

The Hachisu disk simulation generates a self-gravitating thick torus using the method of Hachisu (1982) of solving the Poisson gravity equation simultaneously with the q-parameterized rotation on cylinders model of a cylindrically symmetric gas disk.

This simulation is... not happy.

7.6.1 Initial Conditions

See Hachisu (1982) for derivation of equilibrium equations.

7.6.2 Analysis

...

7.7 Implosion Symmetry Test

The implosion symmetry test begins with initial conditions which have mirror symmetry about the $x = y$ axis. Fluid mechanics dictates that then the exact solution will hold this symmetry forever.

Dimension-split codes are virtually never able to reproduce this condition forever because their error commutators inevitably contain asymmetry-generating terms.

Because Imogen is dimensionally split, its error in this test is tracked by comparing the ratio of the antisymmetric to symmetric components of the density.

7.7.1 Initial Conditions

Nominally any condition of mirror symmetry will do. The condition used by Athena consists of a square domain of size 1x1, with

- $\rho(x + y < 1) = \rho_a$
- $\rho(x + y \geq 1) = 1$
- $P(x + y < 1) = P_a$
- $P(x + y \geq 1) = 1$
- $\vec{v} = 0$

By default the gas gamma is set to 7/5 with $\rho_a = .125$ and $P_a = .14$. All boundary conditions are mirrored.

The Imogen parameters are `Mcorner` and `Pcorner`.

This results in a standard Sod-like behavior in the middle of the corner/bulk region, and a 'jet' being launched along both edges, where the mirror boundary collides the two jets. These jets promptly develop shear-flow instabilities. When they meet in the corner, they expel a two-lobed vortex down the $x = y$ line.

This is typically the point where the asymmetry component becomes obvious as this vortex is intensely unstable to $x = y$-asymmetric disturbances.

7.7.2 Analysis

7.8 Jet Test

The jet test simulation initializes a uniform background, then sets up a fixed 'injector' box which fires fast-moving fluid into this background.

7.8.1 Initial Conditions

...

7.8.2 Analysis

Depending on the mach and the density of the injector fluid, several interesting behaviors are observable:

- Mach diamonds
- KH instability: The jet always creates intense shearing motion, which results in KH turbulence at the jet/background boundary
- RT instability: If the injector fluid is of low density, light fluid is accelerated into dense fluid and the "bubble" it blows out quickly deforms
- Sound generation: A large simulation with long runtimes will be seen emitting sound waves into the far-field, away from the actual jet action

7.9 Kelvin-Helmholtz Shearing Box Test

The Kelvin-Helmholtz effect refers to the instability of a shearing surface to transverse structure formation. It was first analyzed by Lord Kelvin (1871) and Hermann von Helmholtz (1868).

7.9.1 Initial Conditions

Imogen sets up the shearing motion in the X direction.

The initial conditions describe two regions of fluid at equal pressures, with opposite momenta (such that the resulting structure is 'nominally stationary').

7.9.2 Analysis

7.10 Radiative Hydrodynamic Shock Simulation

Gasses subjected to a strong adiabatic shock will be heated very strongly. If the level of heating is sufficient, the shocked fluid may be expected to radiate.

This creates an extended 'shock' region consisting of the adiabatic jump, followed by a radiating region with increasing density. Eventually a cutoff is reached, either physically (optical opacity) or numerically (temperature floor in radiation function).

In the limit where the radiation time is taken to be extremely short compared to the flow time, the familiar adiabatic Rankine-Hugoniot condition that $E_2 = E_1$ is replaced by the isothermal condition $T_2 = T_1$, resulting in an isothermal shock.

7.10.1 Initial Conditions

The radiating shock simulation divides the X component of a 1-, 2- or 3-dimensional grid up into three sections, the preshock, cooling, and cold gas layers.

Imogen accepts as input the parameters `fractionPreshock` and `fractionCold`. The remainder forms the cooling region; Imogen chooses the grid spacing such that the distance

from the shock to the cold layer matches the length returned by the `RadiatingFlowSolver` class.

This distance is approximately equal to the cooling length $L_{\text{cool}} \approx v_x \frac{\dot{P}}{P}$ which in turn is defined by the shock's Mach number.

Imogen chooses frames such that the shock is initially stationary.

The `RadiatingFlowSolver` class will generate a cooling region solution that is effectively exact, far more accurate than the CFD code's truncation error.

Imogen requires an artificial parameter `Tcutoff` which is the temperature that the cooling flow is allowed to reach, in units of the preshock temperature. This value must be at LEAST one. In practice, it should be slightly larger $(1.02 - 1.05)$ to make both the shock jump discontinuity and the cold cutoff $\mathbb{C}^0$ points better behaved.

Without a temperature cutoff, a hydrodynamic radiating flow with a reasonable radiation law will cool to zero temperature, infinite density and zero velocity in finite time and distance.

7.10.2 Analysis

...

7.11 Rayleigh-Taylor Test

The Rayleigh-Taylor test simulates a compressible form of the RT instability: lighter fluid which is supporting (or being accelerated into) denser fluid is locally unstable.

In the inviscid case, any condition of pressure and density gradients pointing opposite ways will be unstable in this manner.

7.11.1 Initial Conditions

Imogen sets up its RT simulation in a rectangular box, with circular conditions on X and Z and a fixed potential field ('gravity') pulling in the -Y direction. The light/dense contact discontinuity is placed at $Y = .5$. The parameters are

- `rhoTop` - density of top fluid
- `rhoBottom` - density of lower fluid
- `P0` - the gas pressure at the BOTTOM of the column*
- `pertAmplitude` - the magitude of velocity perturbations
- `Kx, Ky, Kz` - wave vector for coherent perturbations
- `randomPert` - if true, uses random-per-cell velocities instead

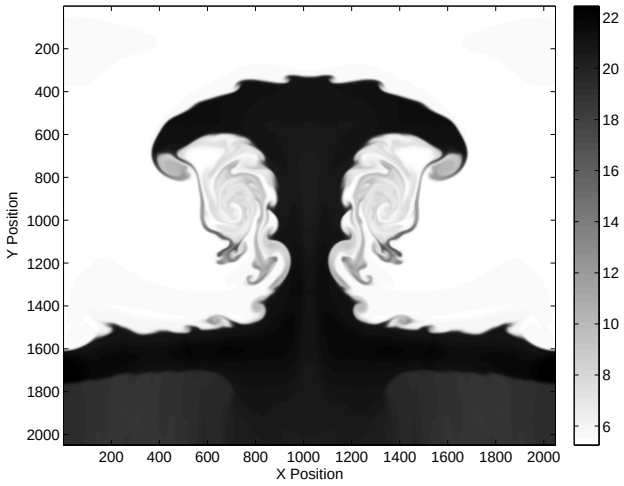

Figure 3: Richtmyer-Meshkov instability at t = 6504

7.12 Analysis

Lord Rayleigh's famous 1883 paper diagnosed the linear instability of the above described situation in the incompressible case.

If the fluid is inviscid and the change of density is instant (contact discontinuity), the growth rate of the linear instability in fact increases unboundedly with wavenumber and the system has *zero* lifetime.

Viscosity, finite density gradient, and surface tension all establish short-wavelength cutoffs and regularize the problem.

7.13 Richtmyer-Meshkov Instability Test

RMI occurs whenever a plane shockwave becomes incident upon a non-uniform density interface. The plane shock refracts through the interface, imparting differential vorticity that drives a jet from the denser fluid into the lighter fluid.

7.13.1 Analysis

For this run, we used a square grid 2048 zones across, for 20,000 iterations.

7.13.2 Initial Conditions

Boundary conditions are periodic around the X axis, constant around the Y axis, and mirrored around the Z axis.

The physical input parameters to the Richtmyer-Meshkov Instability test are:

- `mach - Defines the mach speed of the incoming shock wave`
- `rhotop - Defines the density of the lighter fluid`

- `rhobottom - Defines the density of the heavier fluid`

To create the non-uniform interface, we create a cosine wave with an amplitude 1/20 of the height of the grid and with a wavelength 2x the length of the grid. We then place the center of the wave 4 total wave amplitudes below the center of the grid, and set the mass density of the entire region below the wave equal to rhobottom. The rest of the grid is previously set to have density equal to rhotop, and is unchanged by the wave. We then create a shocked region from the top of the grid to a height just 20 zones above the peak of the wave. This shock is composed of a momentum and pressure equal to a blast wave with mach set by the input 'mach'.

Note that the entire region must then be given a momentum in the direction opposing the shock to maintain position on the grid. This opposing momentum is calculated by the same function as is used to calculate the initial shock.

7.14 Sedov-Taylor Explosion Test

The ST test measures the code's ability to capture high-Mach shocks and its ability to handle them in multiple dimensions.

The initial conditions (analytically) for a Sedov-Taylor explosion consist of a spherically symmetric fluid in N dimensions with

- $\rho(r) = \rho_0 r^- j$
- $v(r) = 0$
- $E_{\rm tot}(r) = E/(\frac{4}{3}\pi\epsilon^3)$

In the standard classical ST case, $j = 0$ and $N = 3$ (producing a spherical explosion that models the adiabatic phase of a nuclear explosion or supernova).

Details are in Sedov (1950), the best known of the original 3 ST papers. A modern derivation which includes a walk-through explanation of the analysis and model code can be found in Timmes 2000 & Kamm & Timmes (2002).

Under most conditions, the result is expected to maintain N-spherical symmetry.

KT also describe the existence of choices of physical ($j < N + 1$, resulting in finite mass within finite volume) solutions which have opposing pressure & density gradients; These would be unstable to Rayleigh-Taylor induced convection.

7.15 Sod Tube Test

This test was used by Gary Sod in his 1976 paper that compared numerous then-in-use schemes for 1D simulation of compressible supersonic dynamics.

While the original test parameters are no longer particularly challenging for any modern shock-capturing code, the Sod test is an enduring part of any test suite.

7.15.1 Initial Conditions

The Sod test refers to the solution of a particular Riemann problem:

- $\rho(x) = \{1, x < 0; .125, x \geq 0\}$
- $P(x) = \{1, x < 0; .1, x \geq 0\}$
- $\vec{v} = \vec{0}$

in an ideal gas with $\gamma = 7/5$.

Imogen uses a box of length 1, constant BCs, and runs to a time of 0.25.

7.15.2 Analysis

...

7.16 Shu-Osher Tube Test

Shu, C and Osher, S., "Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes, II", J. Computational Physics, 83, 32-78 (1989). The test is Example 8.

The grid is divided into two parts, with the left containing a uniform postshock flow and the right containing a sinusoidal entropy wave.

7.16.1 Initial Conditions

Boundary conditions are constant around the X axis and periodic everywhere else.

The physical input parameters to the Shu-Osher test are:

- `lambda` - Defines the entropy wavenumber in waves/box (Shu-Osher: 8)
- `mach` - Defines the strength of the shockwave (Shu-Osher: 3)
- `waveAmplitude` - Defines the amplitude of the entropy wave $d\rho/\rho$ (Shu-Osher: .2)

7.16.2 Analysis

Linearly, the interaction of the shock with the sinusoid modulates the shock strength and generates both entropy and sound characteristics propagating into the postshock region.

Imogen contains the linear (in infinitesmal entropy wave amplitude) solution and can generate output graphics to compare with.

The interaction of the shock with the entropy waves is a source of immediate nonlinearity. An attempt to solve this to second order has been made, but this analysis fails to account for other effects of equal order.

Downstream, the sonic characteristics nonlinearly interact with the entropy waves and inevitably steepen up into a train of weak shocks. In simulations of immensely high resolution (500000 cells), the interaction between the compressed entropy wave and the shock train seems to be a bottomless fount of 1-dimensional structure.

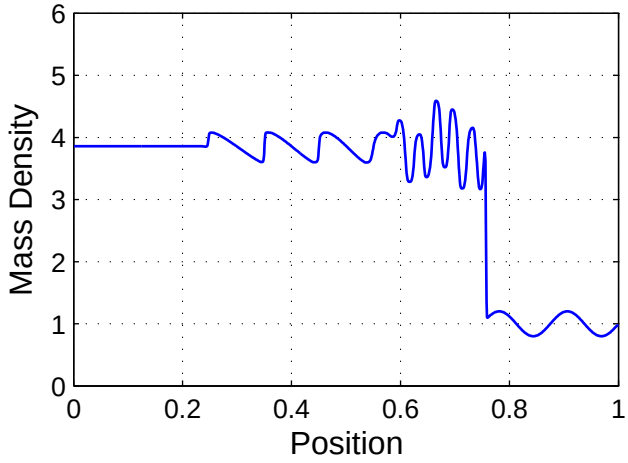

Figure 4: Shu-Osher Shocktube at t = 0.178

In the fourier analysis of the shock front's position, peaks corresponding to harmonic distortion, intermodulation and self-mixing with up to 6 terms is visible in the log-scale plot.