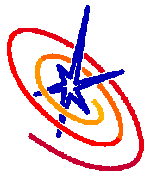


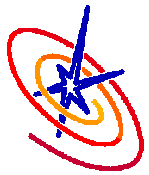
STONY BROOK UNIVERSITY

Explosive Astrophysics with Flash

Alan Calder (alan.calder@stonybrook.edu)
Sean Couch (smc@flash.uchicago.edu)

Many, many others!

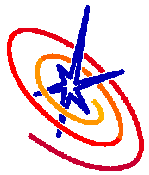
HIPACC Summer School
July 20, 2011



Outline

STONY BROOK UNIVERSITY

- Introducing Flash
  - Components of Flash
  - Running a simulation with Flash
- Simulating with Flash
  - Block Structured AMR
  - System Requirements
  - Verification and Validation: definitions and methods
  - Case studies with Flash
  - Issues relevant to performing quality simulations.
- Sample nuclear astrophysics problems.
  - 2-d cellular detonation
  - 1-d flame
  - 2-d white dwarf detonation



Introducing Flash

STONY BROOK UNIVERSITY

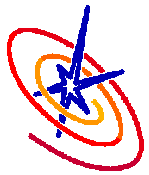
- Flash has been under development since 1997 at the Flash Center for Computational Science at the University of Chicago.

- Many, many developers.
  - Current developers: Anshu Dubey (group leader), John Bachan, Sean Couch, Chris Daley, Milad Fatenejad , Norbert Flocke , Carlo Graziani, Shravan Gopal , Cal Jordan, Dongwook Lee, Dean Townsley, Klaus Weide.

  - Past major developers: Katie Antypas, Alan Calder, Jonathan Dursi, Robert Fisher, Kevin Olson, Timur Linde, Tomek Plewa, Paul Ricker, Katherine Riley, Andrew Siegel, Dan Sheeler, Frank Timmes, Natasha Vladimirova, Greg Weirs, Mike Zingale.

  - Documentation

    http://flash.uchicago.edu/site/flashcode/user_support

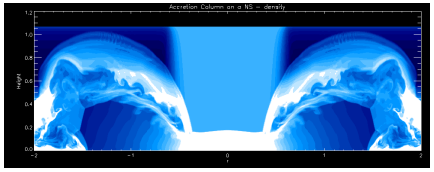


What is Flash?

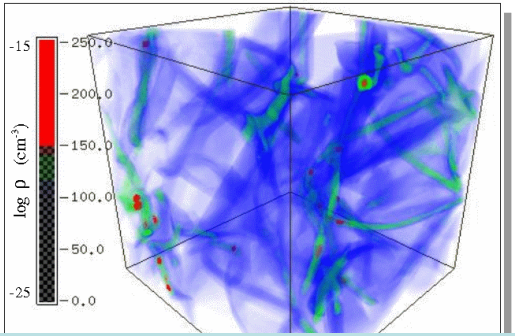
STONY BROOK UNIVERSITY

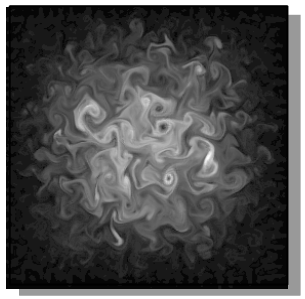
- Flash is composed of units (code modules) that are combined to construct a code for a particular application.
- Flash is written principally in Fortran, with some C and Python. The physics units are in Fortran.
- A setup script performs the steps to construct the code for an application. It processes information in configuration scripts for a particular problem and for the units. It also takes arguments.
- Flash applications include
  - Nuclear astrophysics
  - Cosmology
  - Fluid dynamics
  - High energy density physics

The FLASH Code

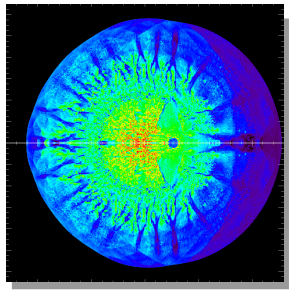


Shortly: Relativistic accretion onto NS

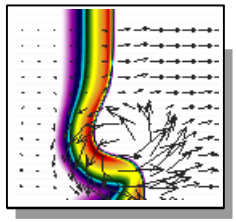




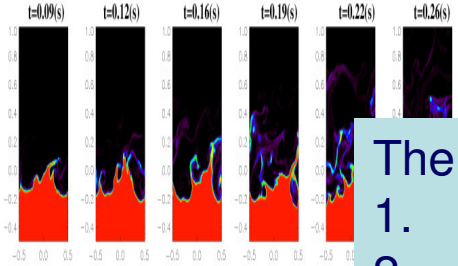
Compressed turbulence



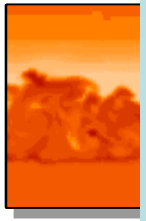
Type Ia Supernova



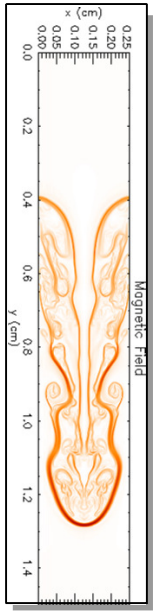
Flame-vortex interactions



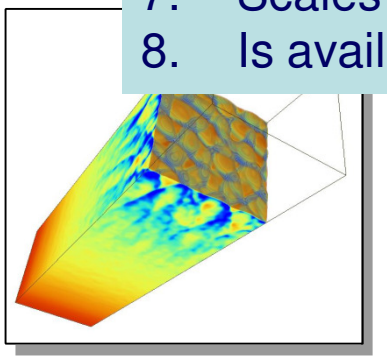
Wave breaking on white dwa



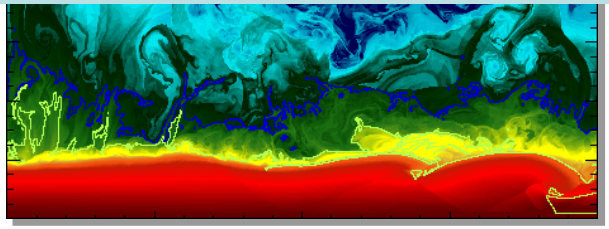
Nova outbu



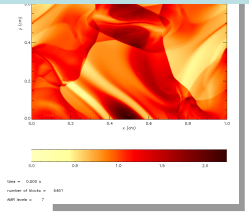
Magnetic Rayleigh-Taylor



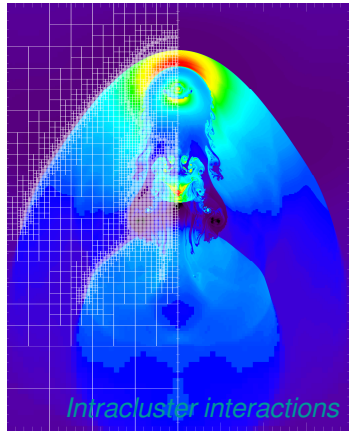
Cellular detonation



Helium burning on neutron stars



Orszag/Tang MHD vortex

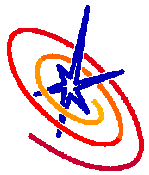


Intracluster interactions

Richtmyer-Meshkov instability

The FLASH code

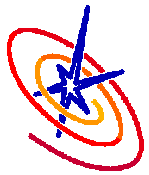
1. Parallel, adaptive-mesh simulation code
2. Designed for compressible reactive flows
3. Solves reactive Euler equations using the PPM
4. Included self-gravity
5. Newly-implemented HED capabilities
6. Newly-implemented MHD capabilities.
7. Scales and performs well.
8. Is available on the web: http://flash.uchicago.edu



Flash Infrastructure Capabilities

STONY BROOK UNIVERSITY

- Configuration (setup)
- Mesh Management
  - PARAMESH- block structured AMR
  - Chombo- patch based AMR
  - Uniform grid
- Parallel I/O
  - HDF 5
  - PnetCDF
  - Fortran
- Monitoring
- Performance monitoring
- Verification testing
  - Unit
  - Regression



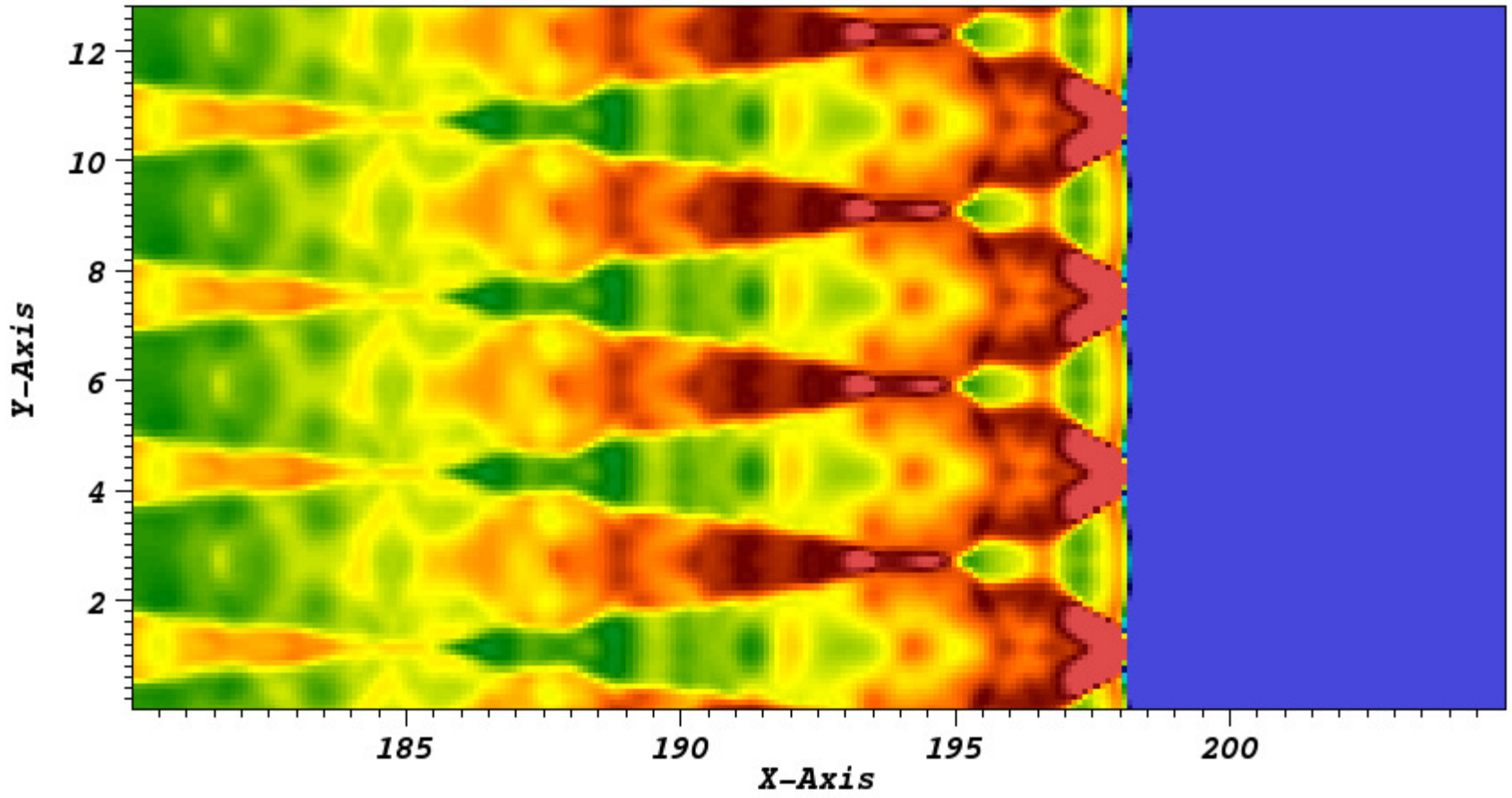
Flash Physics

STONY BROOK UNIVERSITY

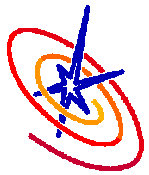
- Flash Units:
  - Hydrodynamics, MHD, RHD
  - Material equations of state
  - Nuclear physics and other source terms
  - Gravity- applied and self-gravitating
  - Material properties
  - Cosmology
  - High energy density
- Particles
  - Lagrangian tracers
  - Active (massive)

Example: Cellular Detonation Problem

- Nuclear astrophysics example : a cellular detonation.



Problem described in Timmes et al. ApJ 543 938 (2000)



Running Flash I

STONY BROOK UNIVERSITY

Move to the project directory

```
cd /project/projectdirs/training/HIPACC_2011/calder/
```

Unpack the tar file

```
tar xzvf FLASH4-alpha
```

Move to the directory in which the code resides

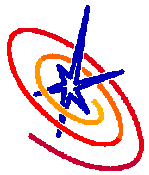
```
cd FLASH4-alpha
```

Run the setup script to set up the cellular detonation problem

```
./setup Cellular -auto -site=hopper.nersc.gov
```

Move into the newly-created object directory and compile the code

```
cd object
make
```



Running Flash I

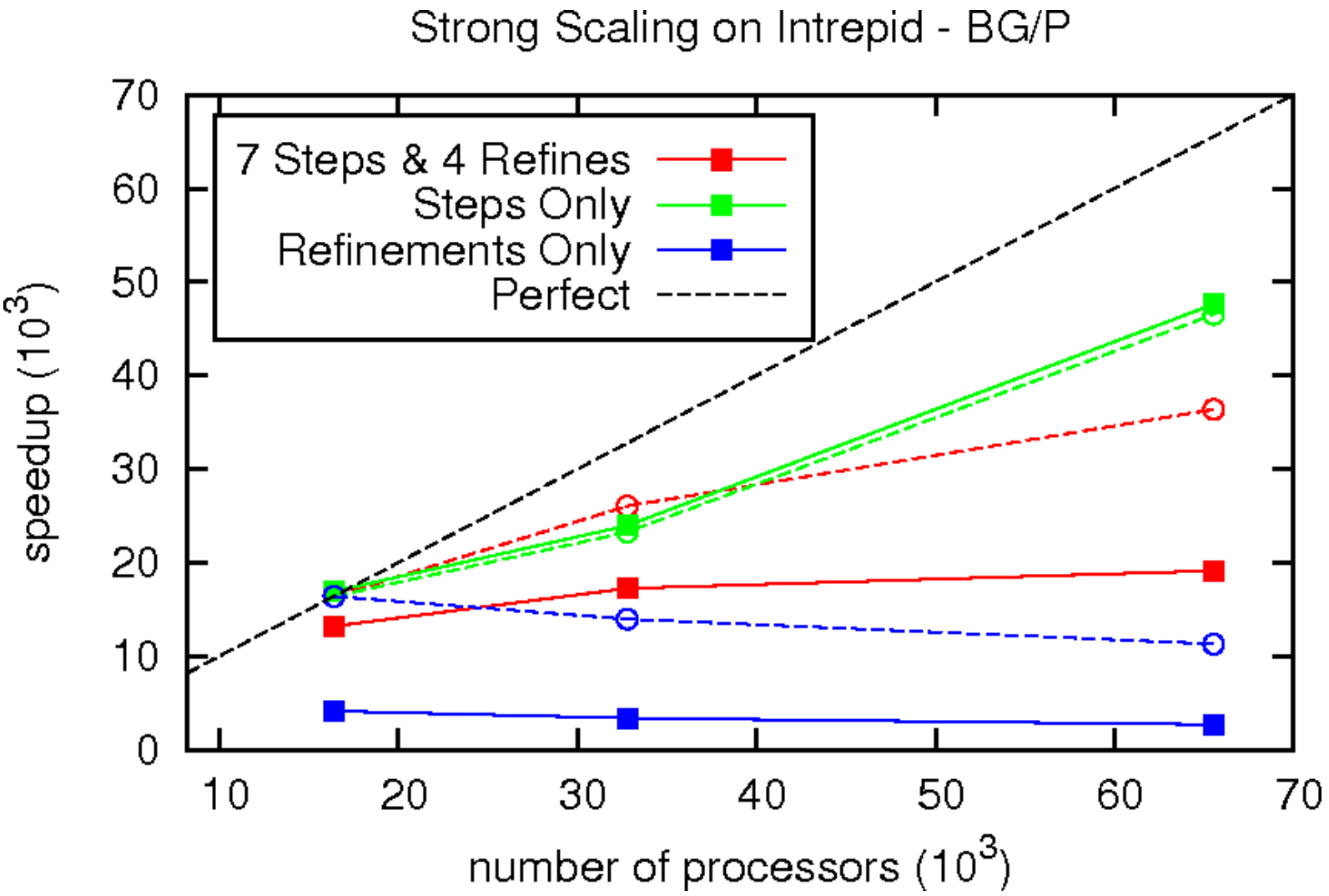
STONY BROOK UNIVERSITY

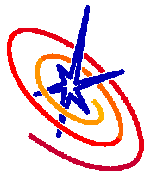
- Important note: One can set many parameters at setup time to tailor the simulation for the architecture, etc.
- An example is maxblocks, which sets the maximum number of blocks per processor element. Recall Katie Antypas's point about the memory per core decreasing from Franklin to Hopper. Must account for thiis!

```
./setup Cellular -auto -site=hopper.nersc.gov -
  objdir=obj_cellular2 -maxblocks=200
```

- More on estimating the memory utilization later.

Performance Example



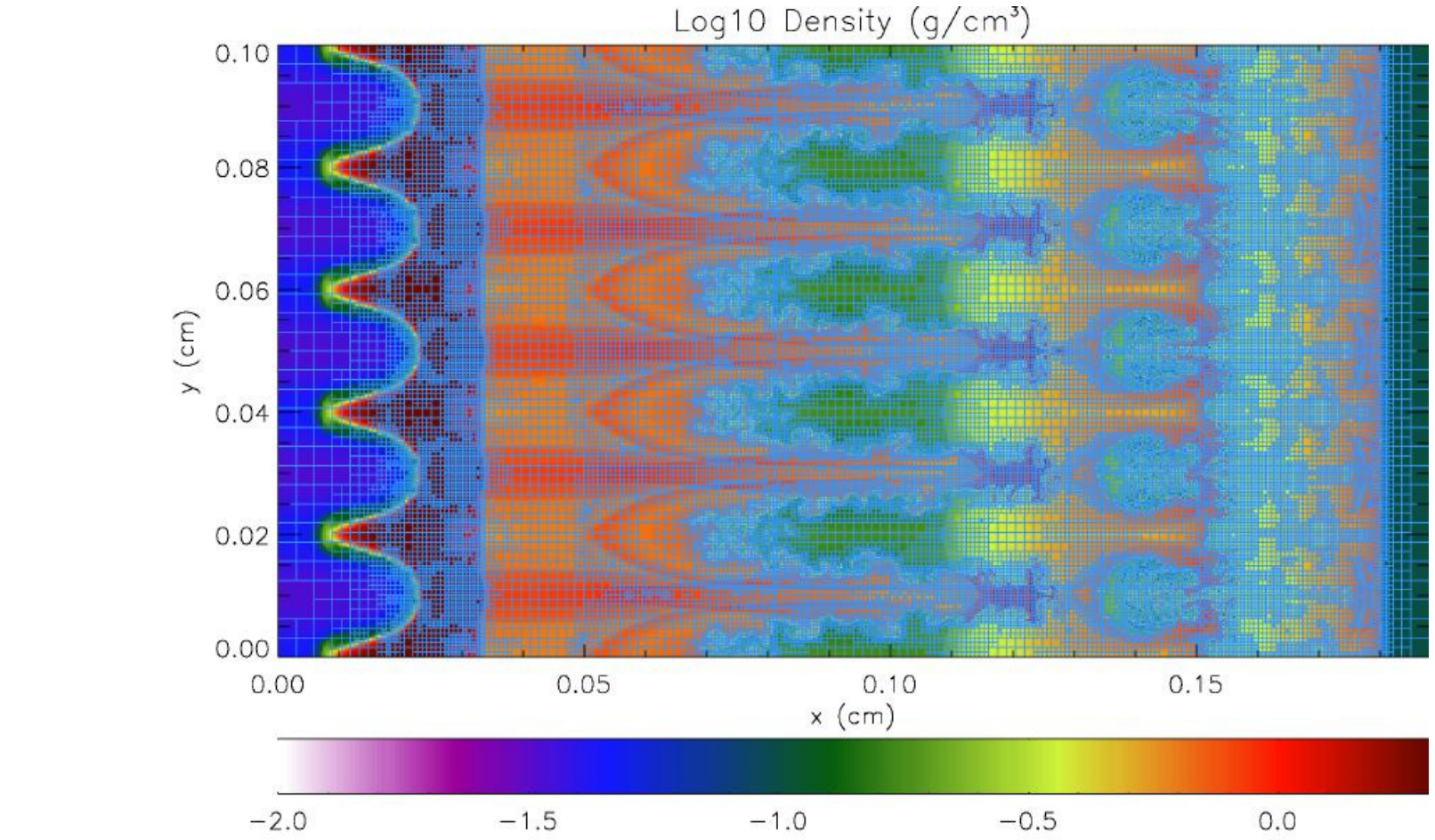


Adaptive Mesh Refinement

STONY BROOK UNIVERSITY

- AMR seeks to minimize computational expense by adding resolution elements only as needed to resolve features in the flow.
- Flash uses a block-structured approach to AMR.
- The simulation domain is divided into a series of logically-Cartesian blocks with the resolution set by the number of blocks in a region of physical space.
- The number of blocks changes (adapts) via a parent-child relationship.
- The mesh package (e.g. PARAMESH)
  - Manages the creation of grid blocks
  - Builds and maintains the tree structure that tracks the spatial relationship between blocks
  - Distributes blocks among available processors
  - Handles inter-block and inter-processor communication.
  - Tracks physical boundaries and enforces boundary conditions

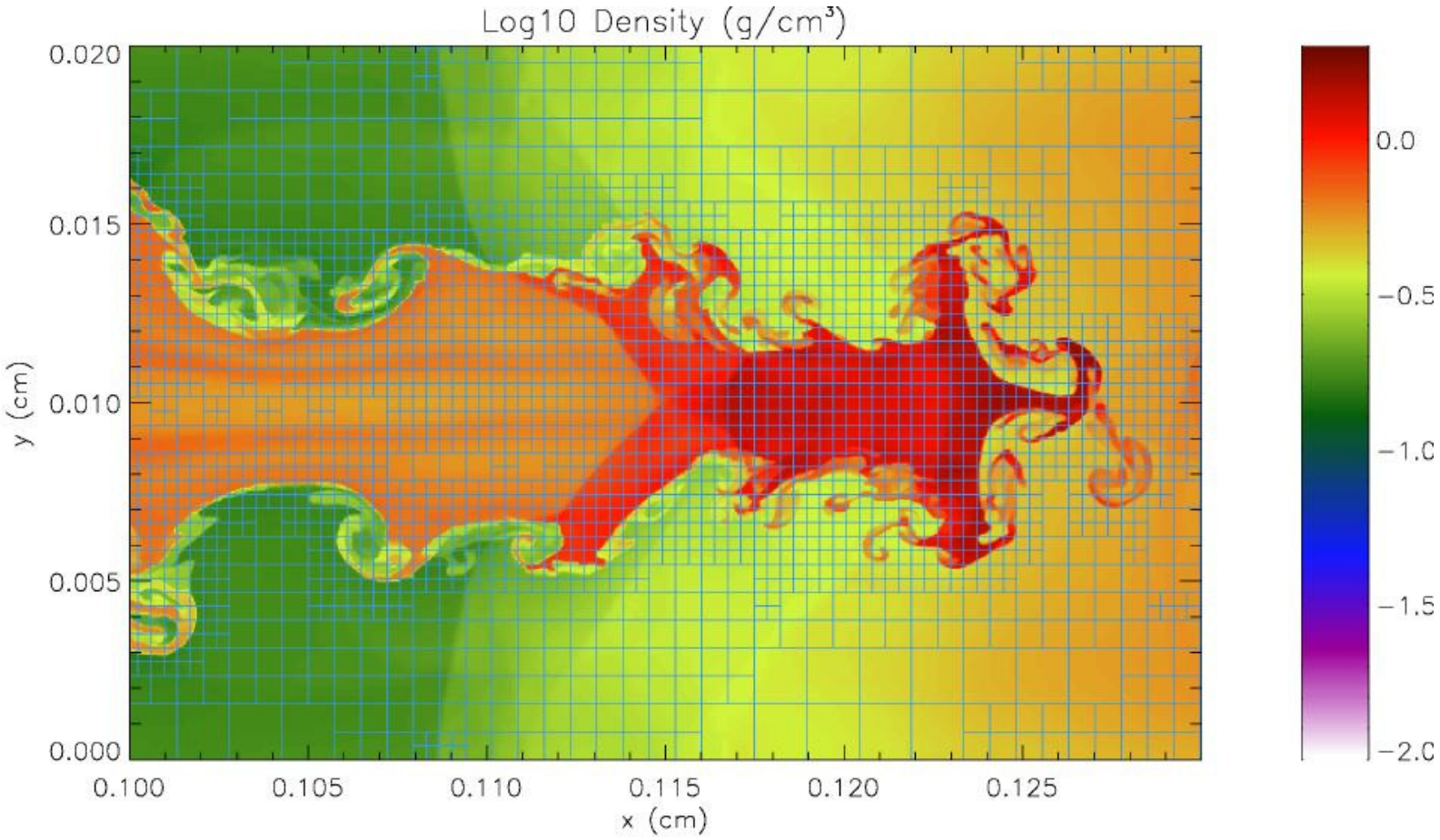
Adaptive Mesh Refinement



time = 64.001 ns
number of blocks = 90166, AMR levels = 9

/scratch4/calder/new_val_res_study/work_2048/3lay_hdf_plt_cnt_0064

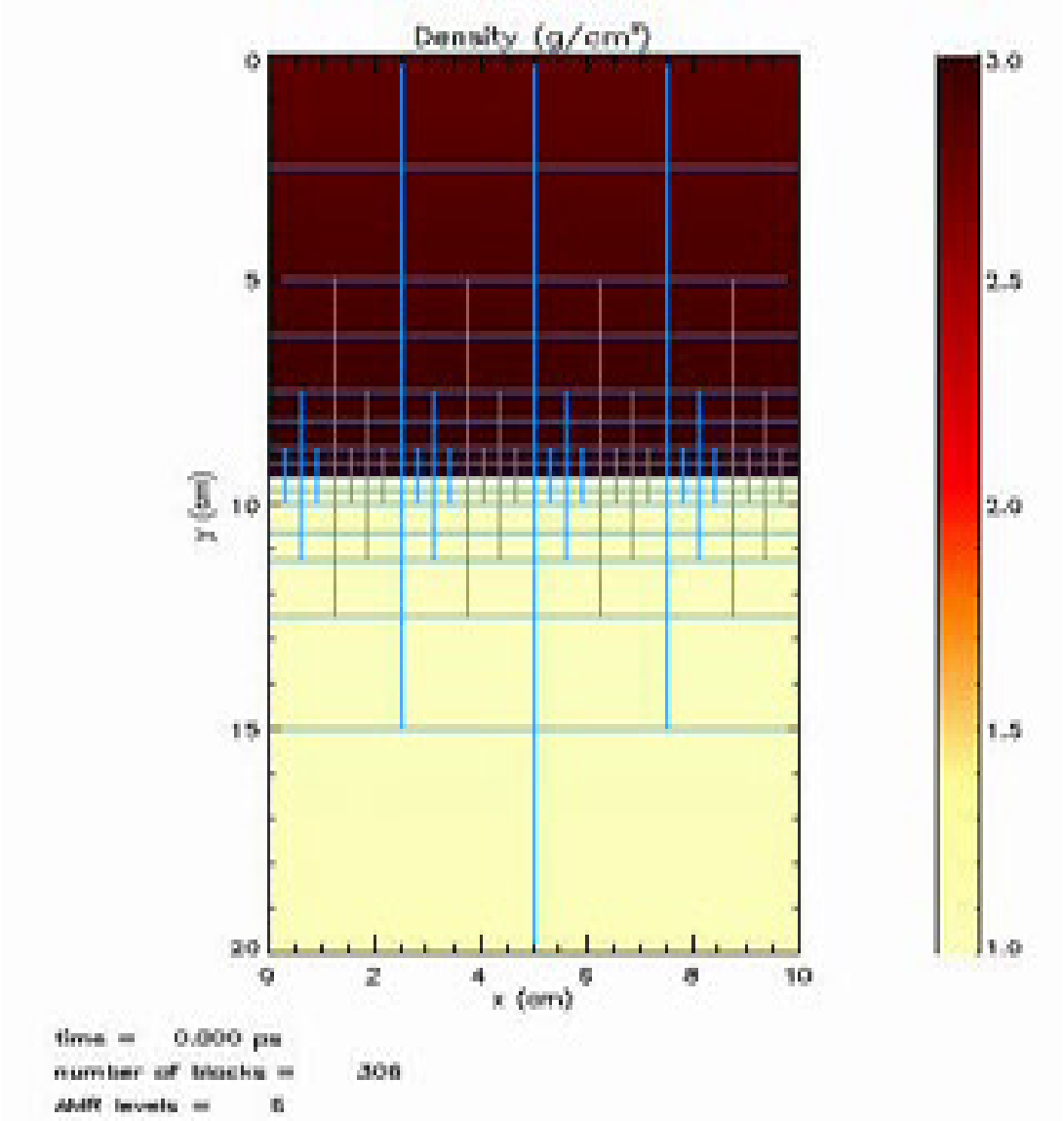
Adaptive Mesh Refinement



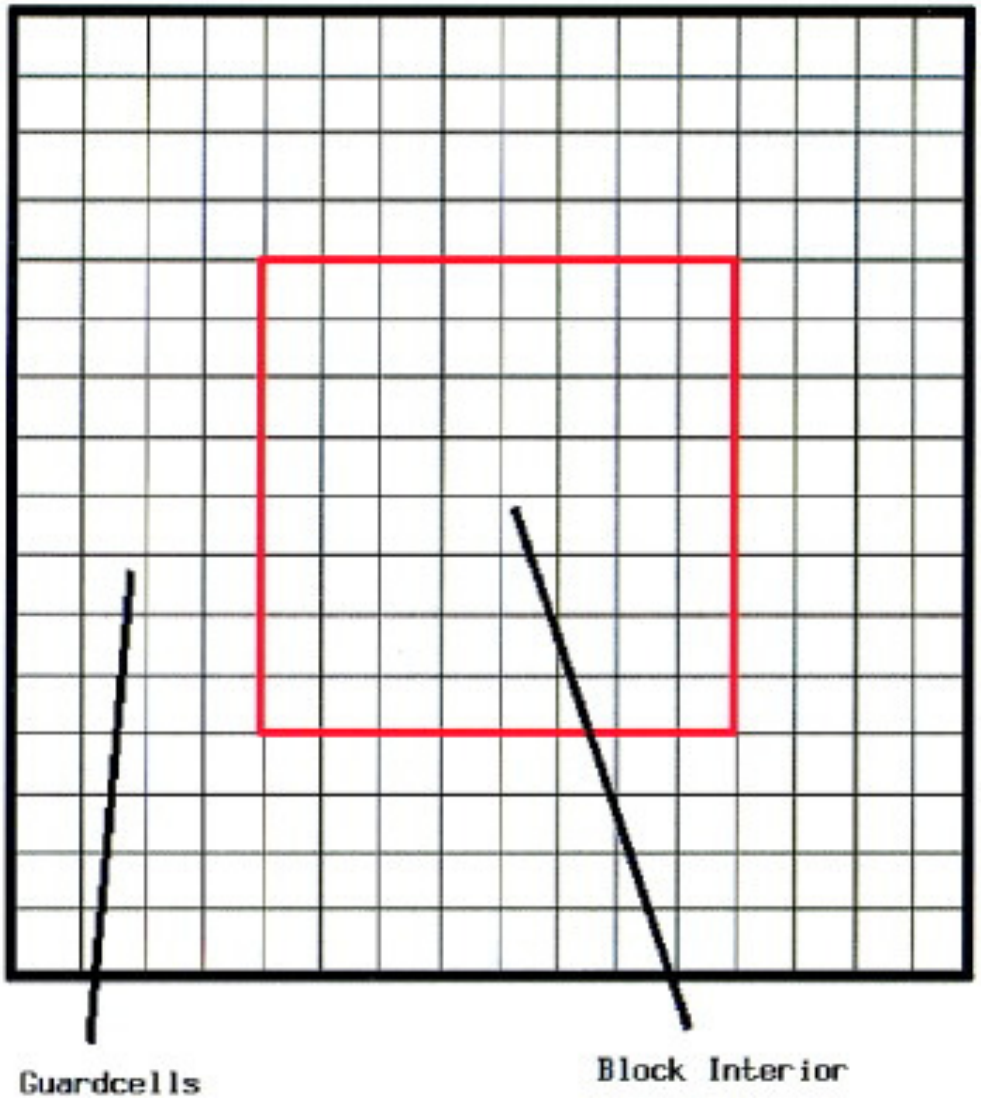
time = 64.001 ns
number of blocks = 90166, AMR levels = 9

/scratch4/calder/new_vol_res_study/work_2048/3lay_hdf_plt_cnt_0064

Adaptive Mesh Refinement

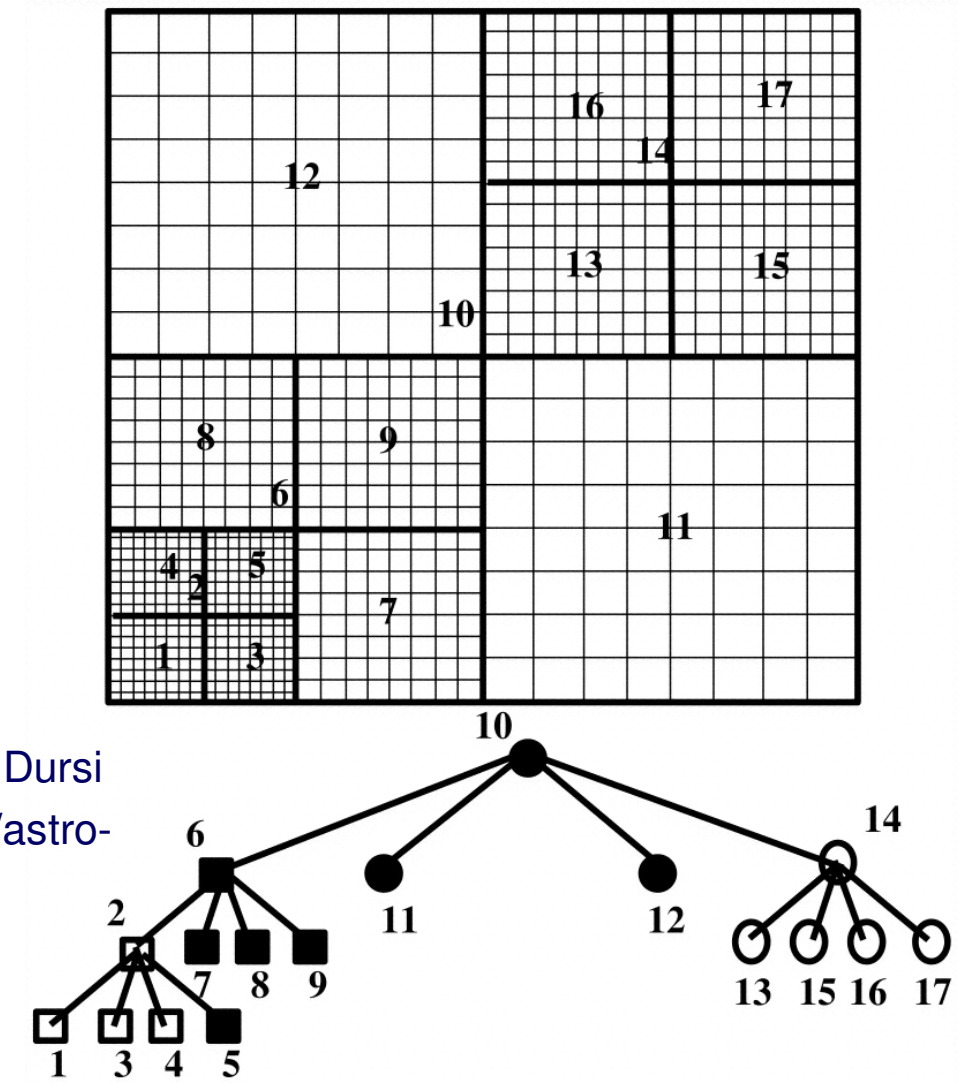


Example Block



From Fryxell et al. ApJS 131 273 (2000)

Example Domain

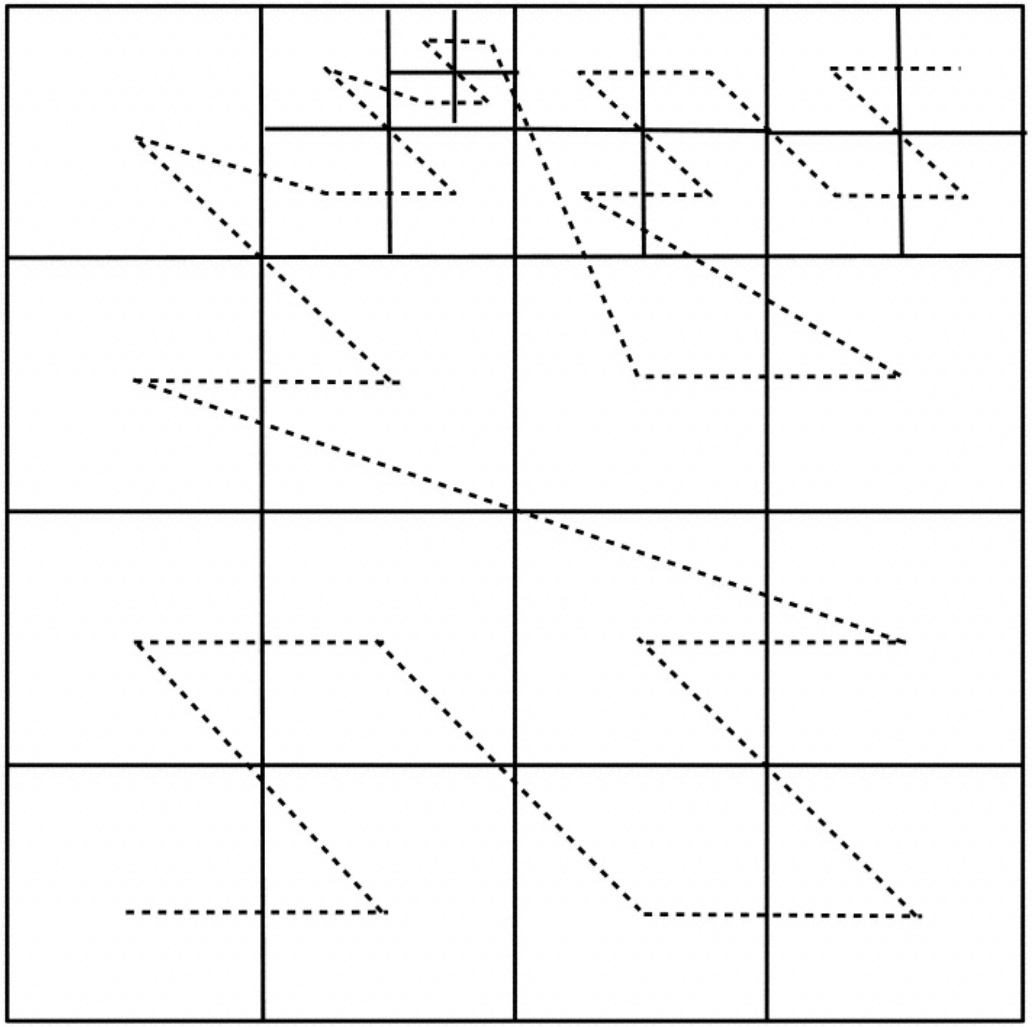


Subcycling- Zingale & Dursi

http://xxx.lanl.gov/abs/astro-ph/0310891

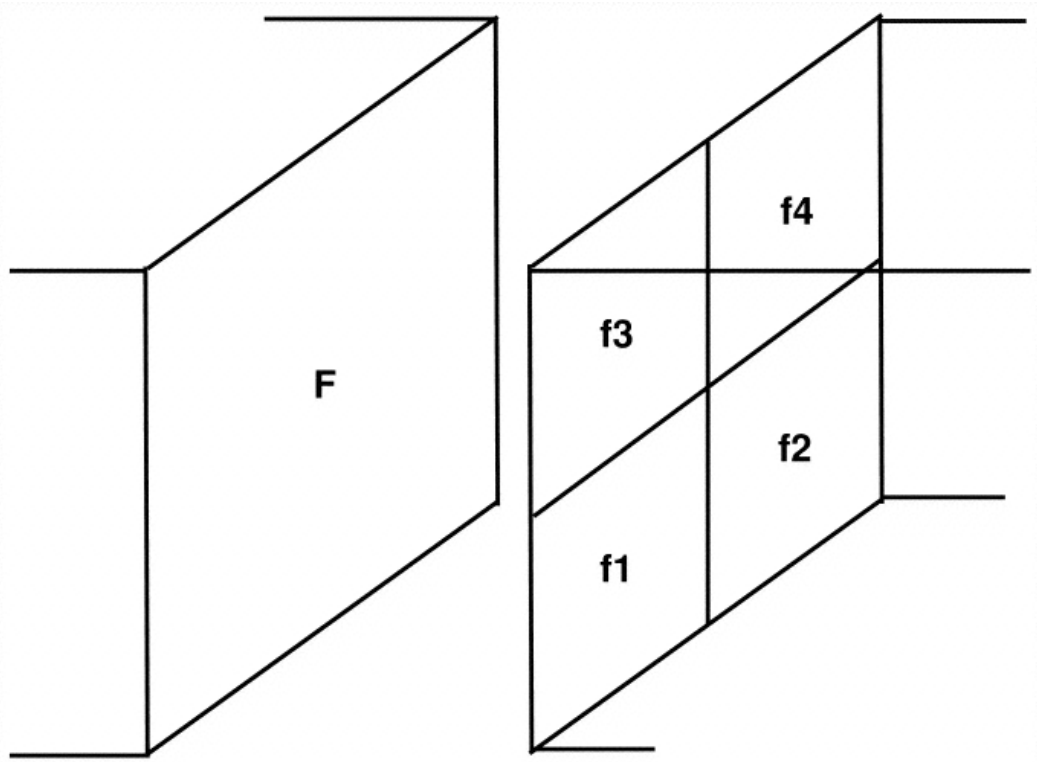
From Fryxell et al. ApJS 131 273 (2000)

Morton Space-filling Curve



From Fryxell et al. ApJS 131 273 (2000)

Flux Conservation at Block Boundaries

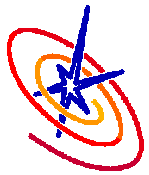


$$F = [\, f1 + f2 + f3 + f4 \,]$$

From Fryxell et al. ApJS 131 273 (2000)

Refinement Criteria

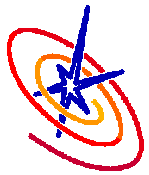
- Default is to look at the magnitude of 2nd derivative for specified quantities (e.g. density and pressure).
- As Ann mentioned, there can be sharp features (edge of star) that are not critical. One can also use locations, values of certain variables, etc.
- Best description I've heard: "black art."
- Key is to test!



Initial Conditions

STONY BROOK UNIVERSITY

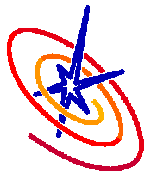
- Very important- When one constructs initial conditions, one writes a routine to initialize the blocks.
- Typical uniform mesh codes perform a loop stuffing arrays over the whole domain. Initializing a block requires querying its location and initialize the variables accordingly. This process can be difficult to conceptualize.
- For this Summer School, we will look at three setups relevant to simulating Type Ia supernovae
  - cellular detonation
  - 1-d deflagration
  - 2-d detonation in a white dwarf.
- Homework assignment: Modify the cellular detonation setup to instead simulate a 1-d deflagration. (Details below).



System Requirements for Flash

STONY BROOK UNIVERSITY

- Compiler for Fortran (F90) and C.
- Installed copy of the Message Passing Interface (MPI)
- Installed packages for I/O, Hierarchical Data Format (HDF) or Parallel NetCDF (PnetCDF). There is a Fortran I/O option, but your mileage will vary.
- Flash is distributed with the PARAMESH AMR library included. For Chombo AMR, one must have the Chombo library included.
- For the implicit Diffuse and Hypre solvers, one must have the Hypre library installed.
- GNU make utility.
- Python 2.2 or later for the setup script.



Running Flash II

STONY BROOK UNIVERSITY

Move into or create the run directory (usually in the scratch space)

```
mkdir 20110712_cellular
cd 20110712_cellular
```

Copy the required files into the run directory

```
cp …/FLASH4-alpha_release/object/SpeciesList.txt .
cp …/FLASH4-alpha_release/object/helm_table.dat .
cp …/FLASH4-alpha_release/object/flash.par .
cp …/FLASH4-alpha_release/object/flash4 .
```

Copy or create a run script in the run directory

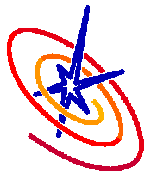
`vi (or emacs) hopper.run` (example next slide)

Submit the job to the queue:

```
qsub hopper.run
```

Monitor the job as you please:

```
qstat -u username
```



Example run script for Hopper

STONY BROOK UNIVERSITY

Example run script:

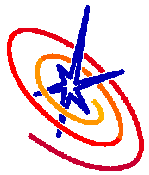
```
#PBS -N cellular01
#PBS -q debug
#PBS -l mppwidth=12
#PBS -l walltime=0:29:00
#PBS -e output.$PBS_JOBID.err
#PBS -o output.$PBS_JOBID.out

cd $PBS_O_WORKDIR

echo Starting `date`

aprun -n 12 ./flash4

echo Ending `date`
```



Operator Splitting

STONY BROOK UNIVERSITY

The different modules operating on the state are strung together via Strang spliitting, a fractional step method, in which the operators act on parts of one time step.

Consider advection + reaction:

$$\text{Problem A:} \quad q_t + u q_x = 0$$
$$\text{Problem B:} \quad q_t = -\lambda q$$

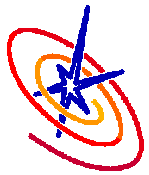
Want to solve combined problem:

$$q_t = (A + B)\, q \qquad \text{where} \quad A = -u\partial_x \ \text{ and } \ B = \lambda(x)$$

Solve via

$$q_t = Aq \qquad \Delta t/2$$
$$q_t = Bq \qquad \Delta t$$
$$q_t = Aq \qquad \Delta t/2$$

From R. LeVeque in “Computational Methods or Astrophysical Fluid Flow”



Driver Routine Illustrating Strang Splitting

STONY BROOK UNIVERSITY

```
do dr_nstep = dr_nbegin, dr_nend

   call Hydro( blockCount, blockList, &
         dr_simTime, dr_dt, dr_dtOld, dr_fSweepDir)
   call RadTrans(blockCount, blockList, dr_dt, pass=1)
   call Diffuse(blockCount, blockList, dr_dt, pass=1)
   call Driver_sourceTerms(blockCount, blockList, dr_dt, pass=1)
   call ravity_potentialListOfBlocks(blockCount,blockList)

   call Hydro( blockCount, blockList, &
               dr_simTime, dr_dt, dr_dtOld, dr_rSweepDir)
   call RadTrans(blockCount, blockList, dr_dt, pass=2)
   call Diffuse(blockCount, blockList, dr_dt, pass=2)
   call Driver_sourceTerms(blockCount, blockList, dr_dt)
   call Gravity_potentialListOfBlocks(blockCount,blockList)

end do
```

Finite Volume Methods

Euler equations : $$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \mathbf{S}(\mathbf{U}) \ ,$$

Conserved quantities: $\mathbf{U} = (\rho, \rho v, \rho E)^T$

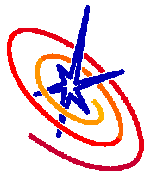
Volume average $$\frac{1}{V}\int \left\{ \frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) \right\} dV = \frac{1}{V}\int \mathbf{S}(\mathbf{U}) dV$$

V Volume

take $\langle \mathbf{U} \rangle$ as volume average of $\mathbf{U}$

Integral form of Euler equations : $$\frac{\partial \langle \mathbf{U} \rangle}{\partial t} + \frac{1}{V}\oint \mathbf{F}(\mathbf{U})\mathbf{n} \cdot \mathbf{dS} = \langle \mathbf{S}(\mathbf{U}) \rangle \ .$$

From Zingale et al. 2002



Finite Volume Methods

STONY BROOK UNIVERSITY

In finite volume methods, the average value of the unknown is given by

$$\langle f \rangle_i = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} f(x) dx$$

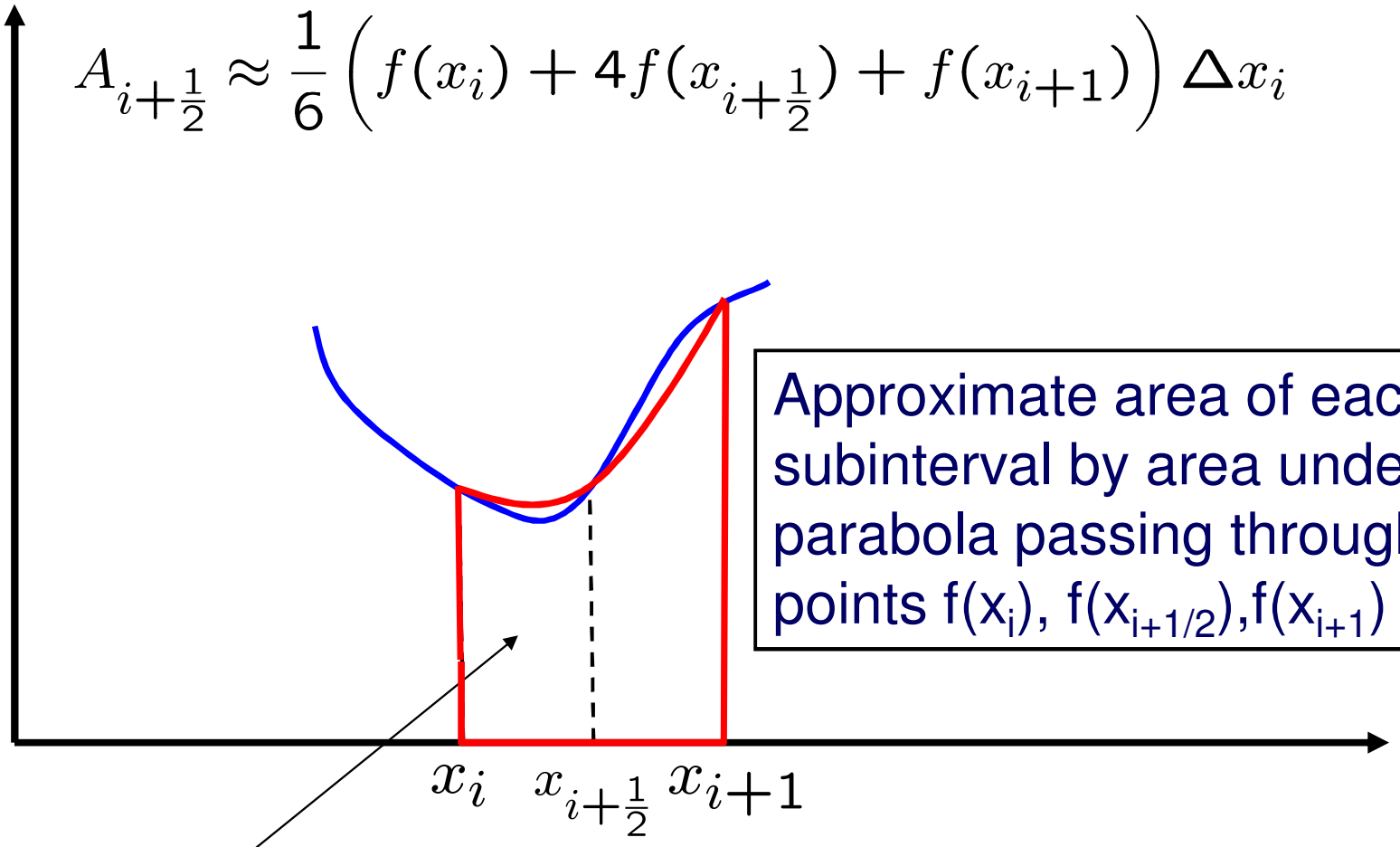
Where $x_{i+1/2}$, $x_{i-1/2}$ are the positions of the left, right edges of the zone.

One performs a reconstruction (piecewise constant, linear, parabolic, …) to get f(x) and then one integrates that.

Parabolic example is Simpson's rule (seen for integration)

Simpson's Rule

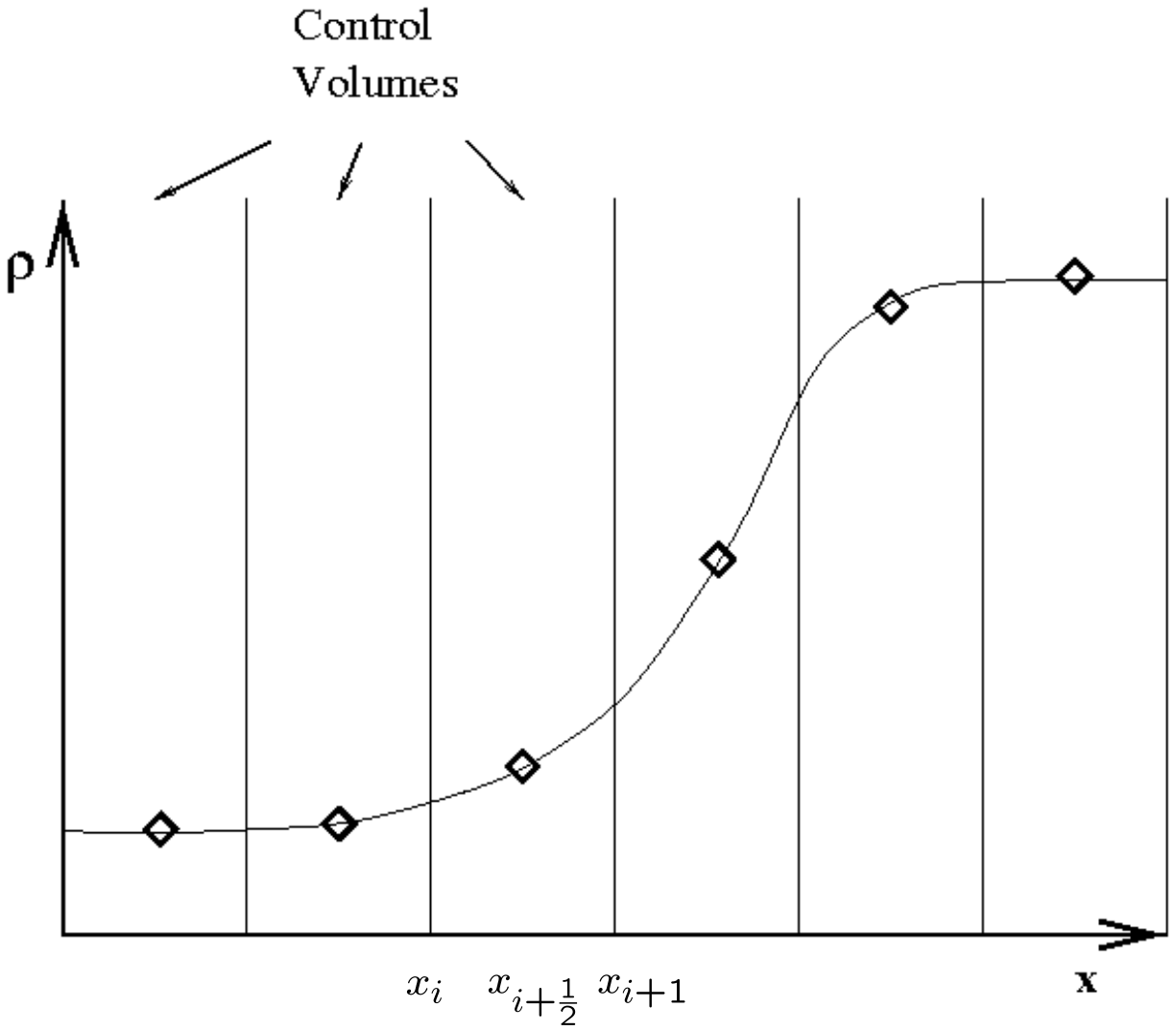
$$A_{i+\frac{1}{2}} \approx \frac{1}{6}\left(f(x_i) + 4f(x_{i+\frac{1}{2}}) + f(x_{i+1})\right)\Delta x_i$$



Approximate area of each subinterval by area under a parabola passing through points $f(x_i)$, $f(x_{i+1/2})$,$f(x_{i+1})$

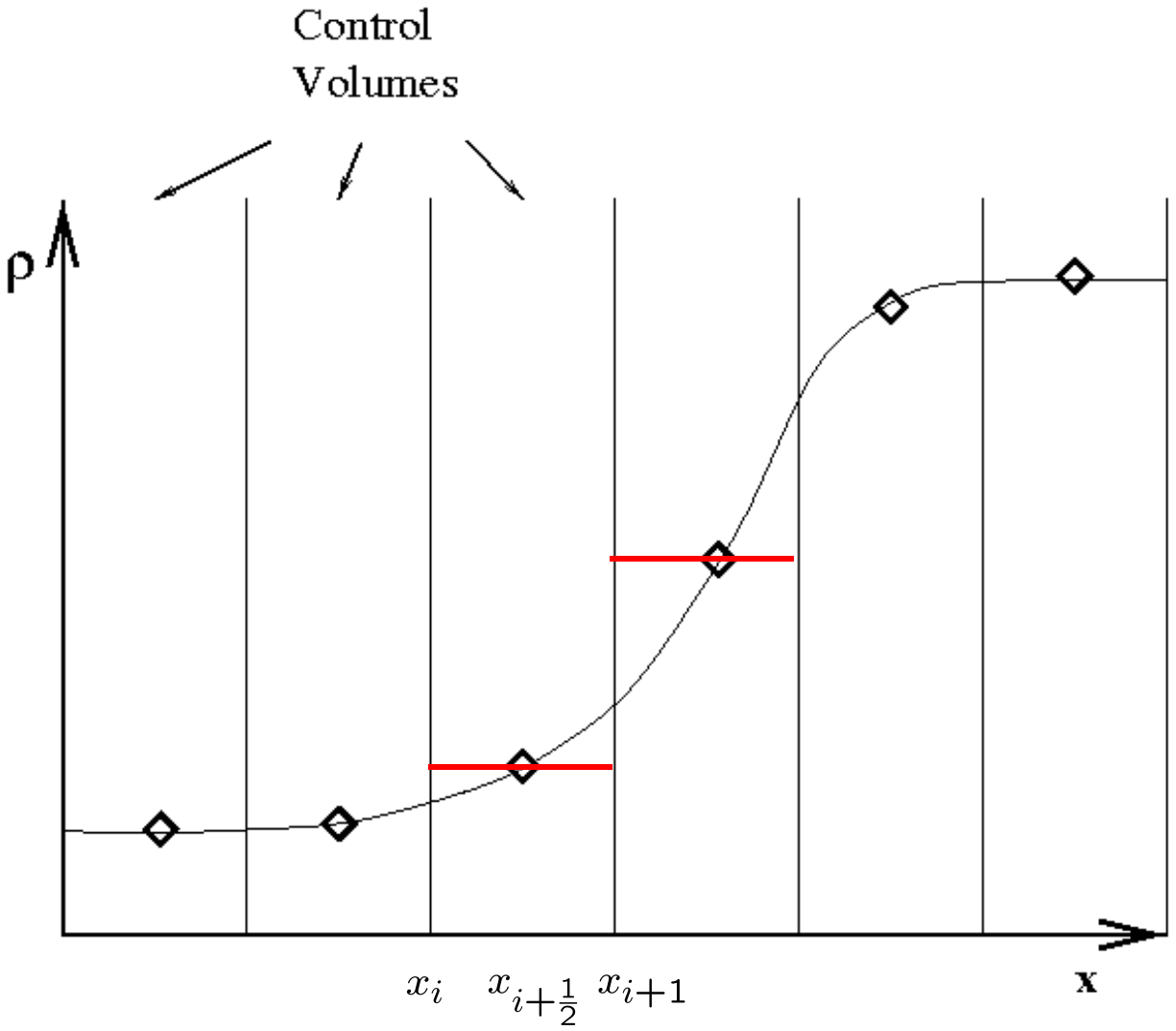
$A_{i+\frac{1}{2}}$ = Area of i+1/2th subinterval

Finite Volume Hydrodynamics Methods



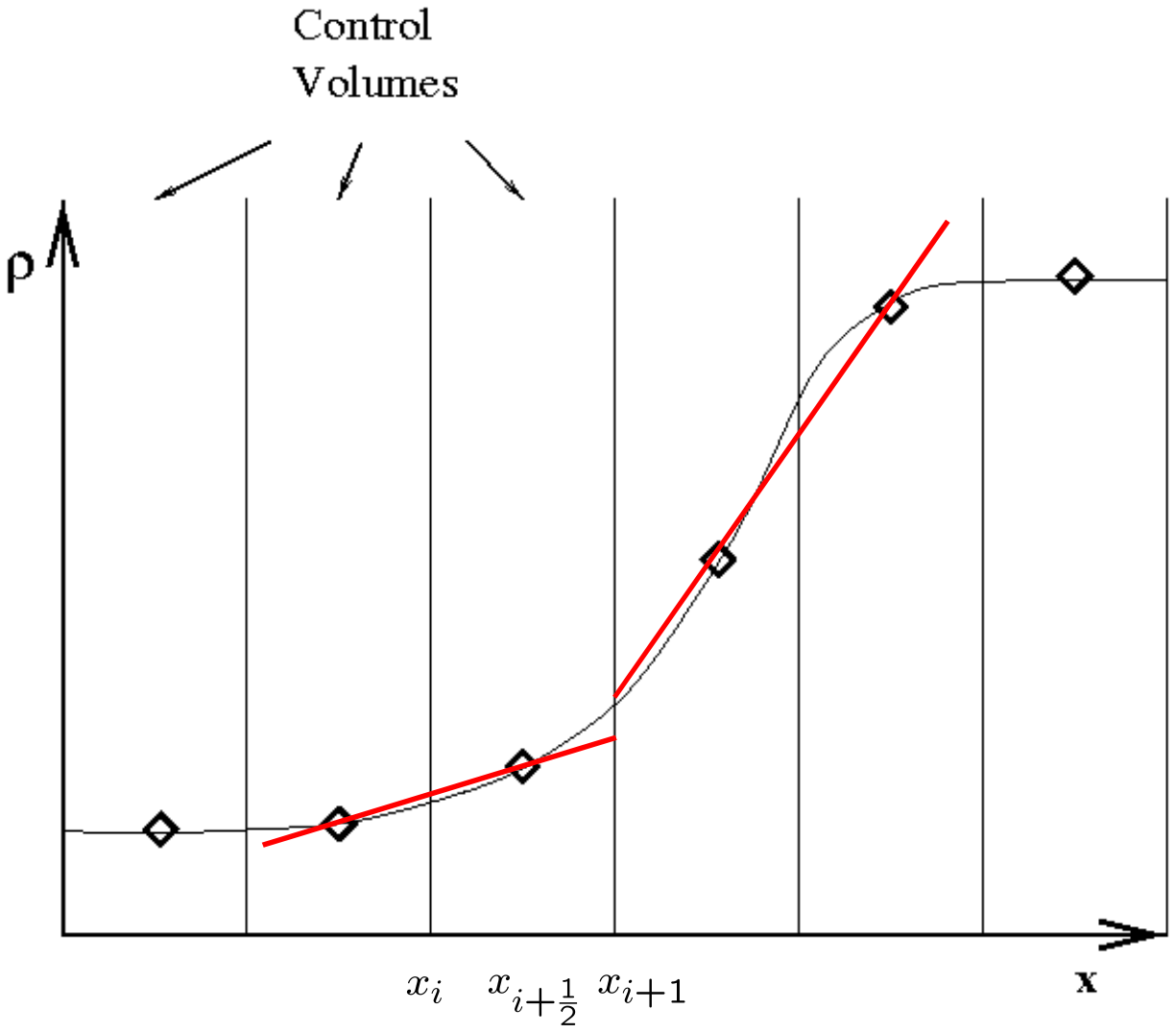
Divide the domain into zones the interact with fluxes

Godunov Method



The original Godunov method solved the Riemann problem at the boundaries between zones using the average values

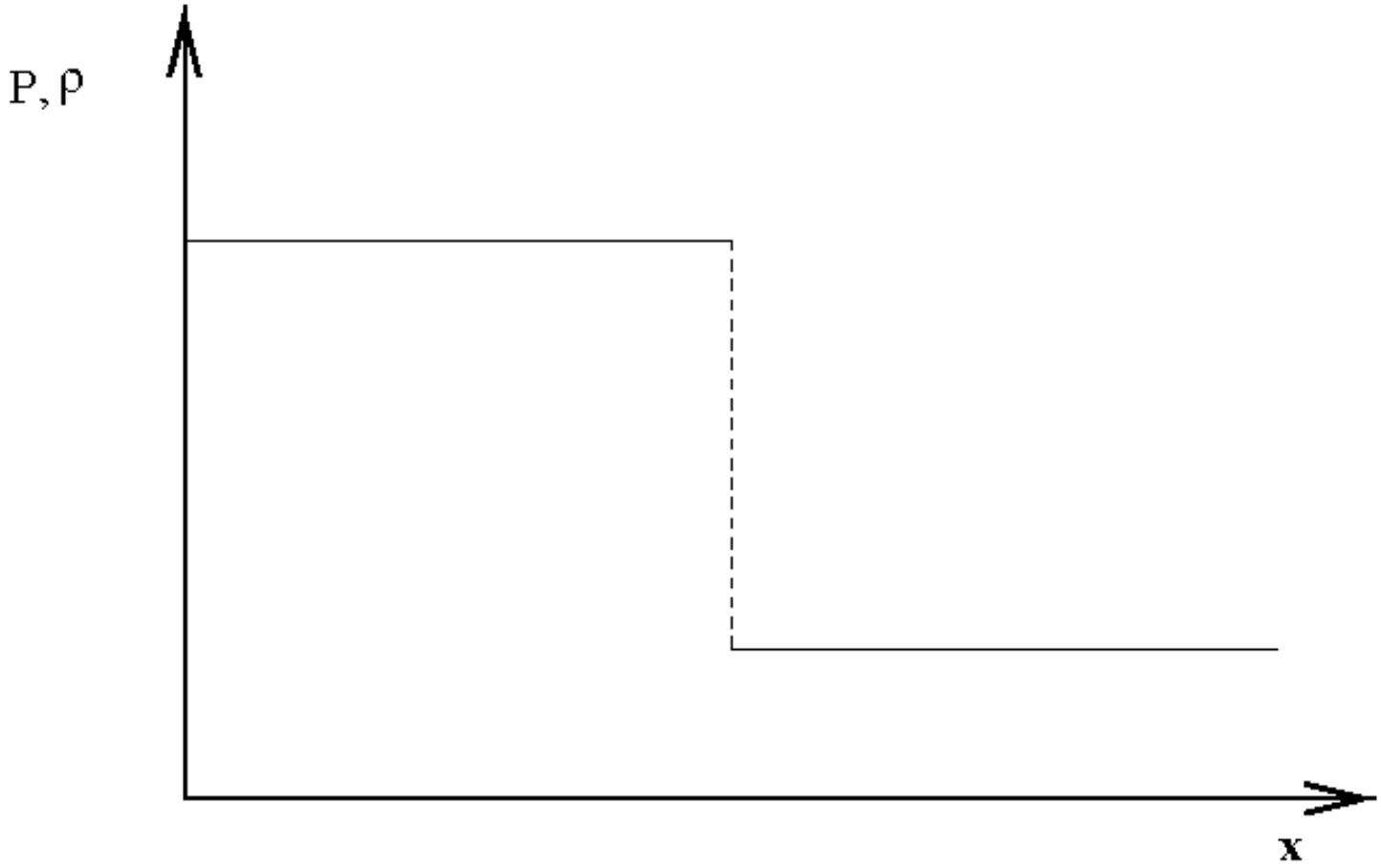
Piecewise Linear Godunov Methods



Piecewise linear methods perform a linear interpolation.

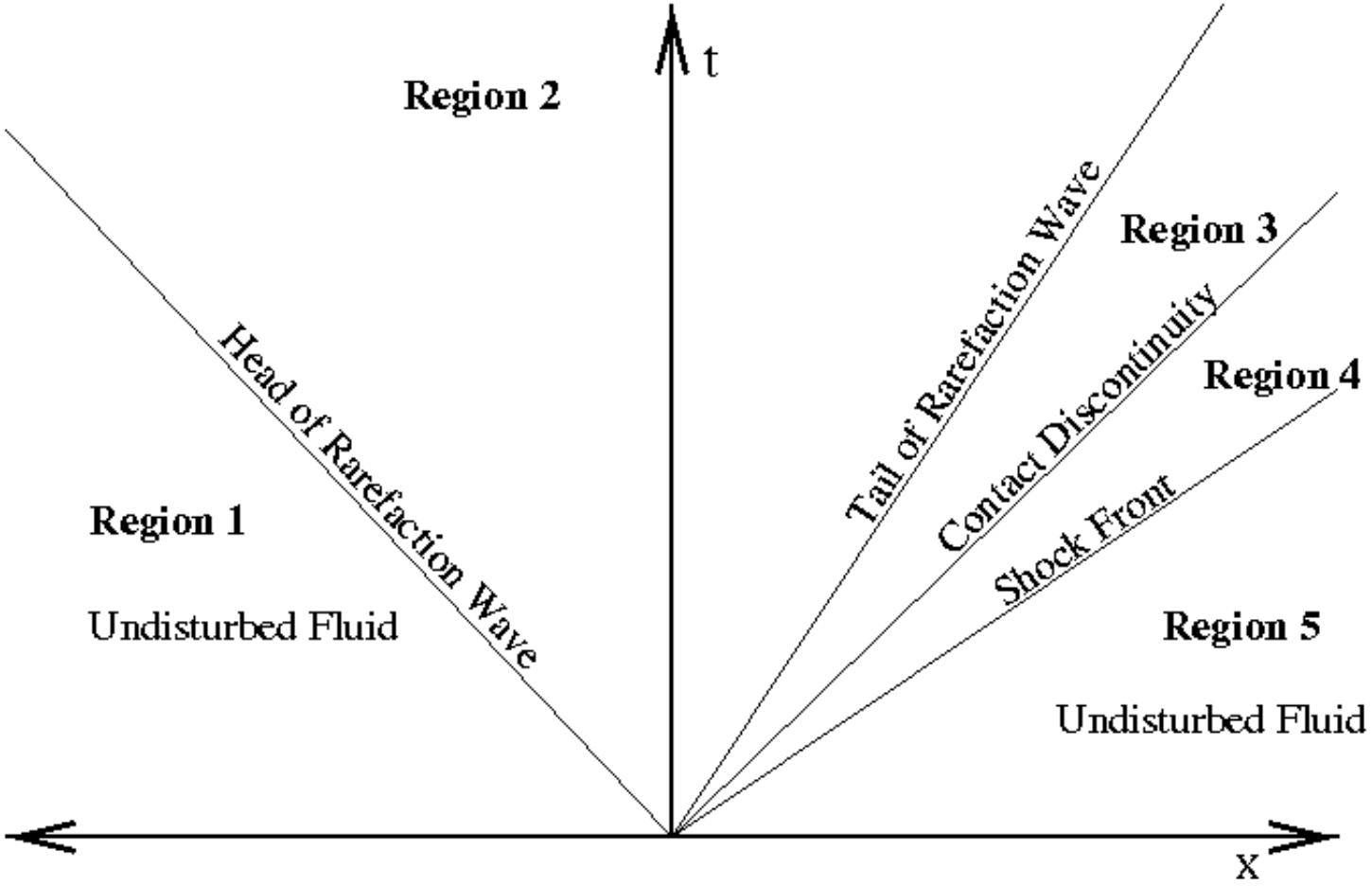
Higher-order methods exist. PPM is Piecewise-Parabolic Method

Riemann Problem: Shock Tube



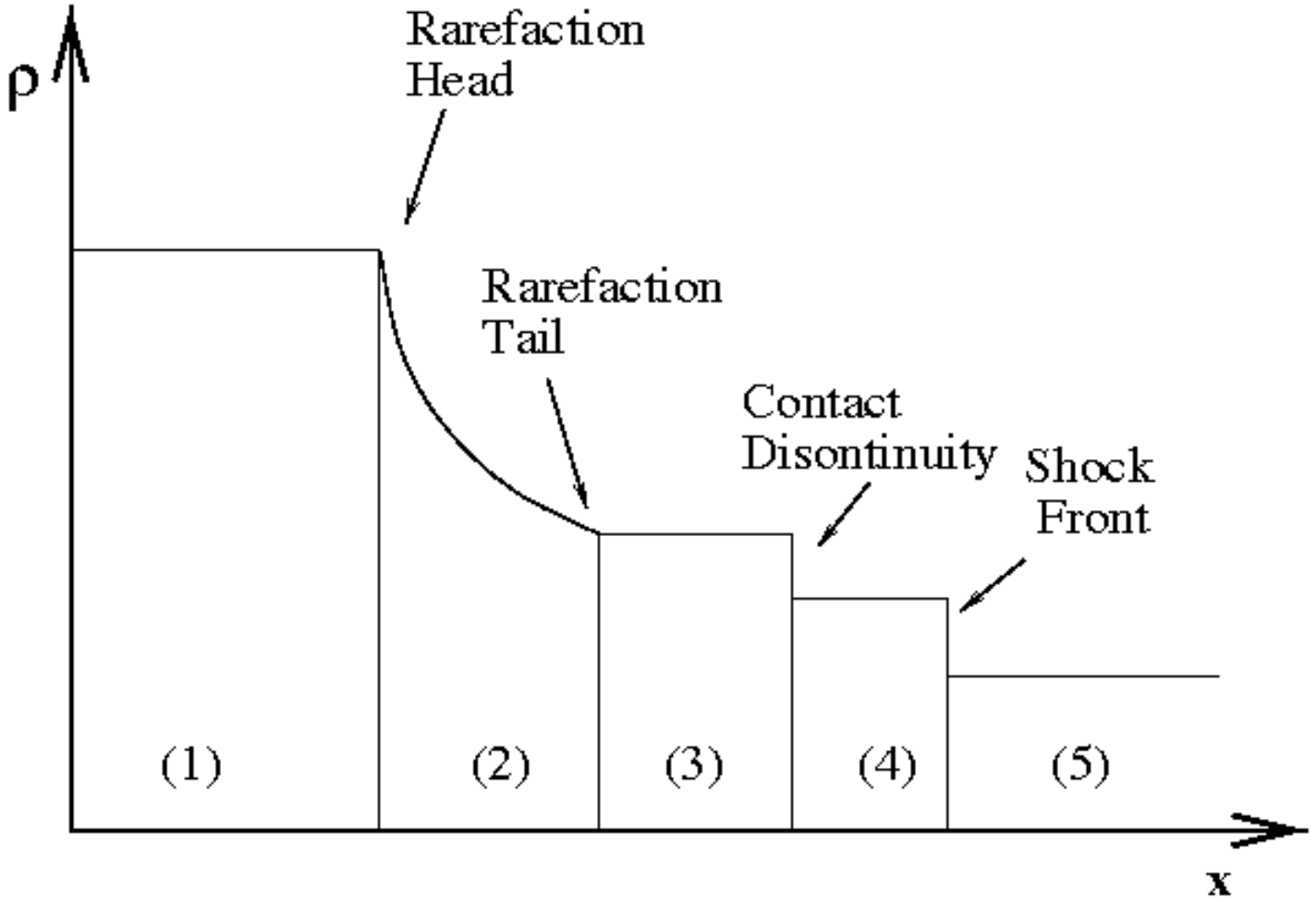
Initial conditions: a discontinuity in density and pressure

Riemann Problem: Shock Tube



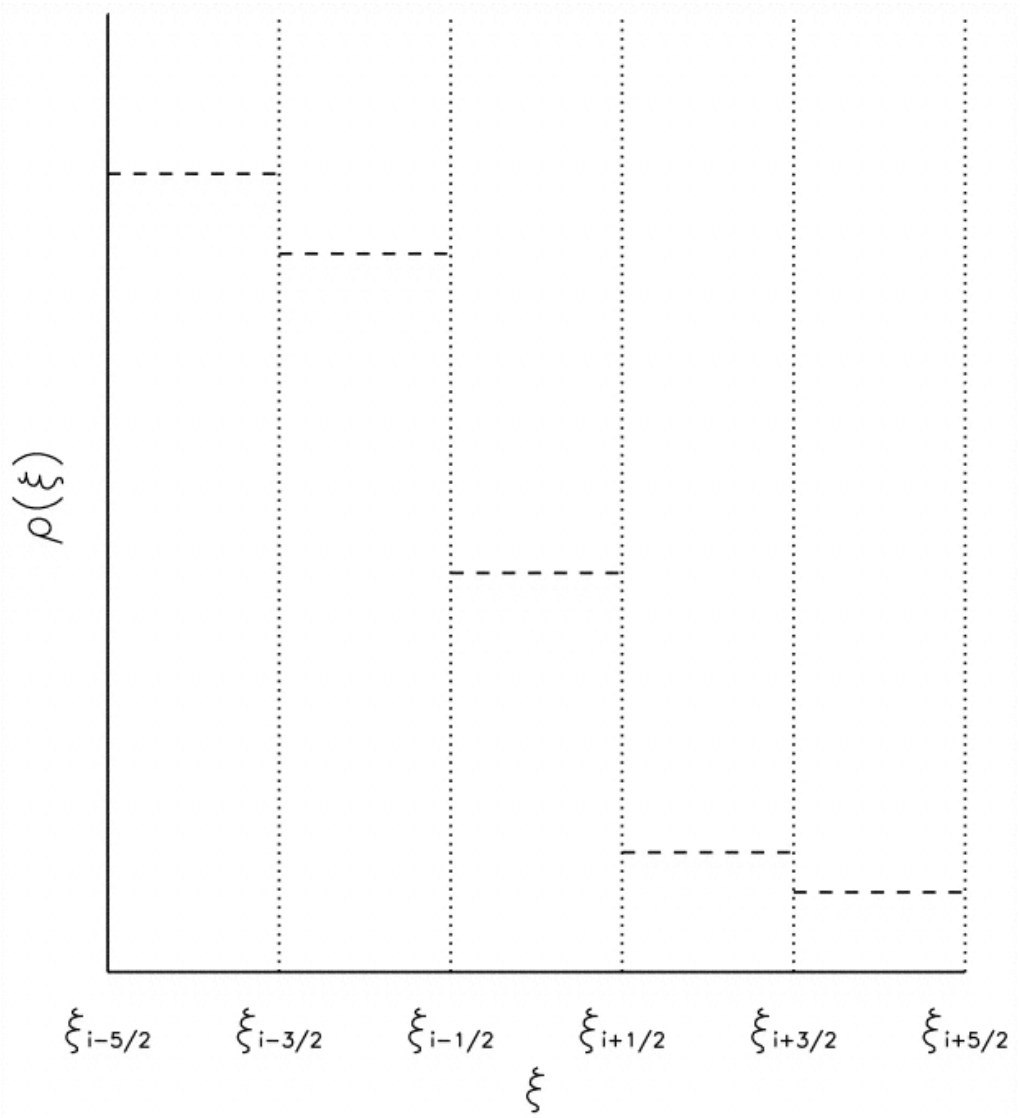
World diagram for Riemann problem

Riemann Problem: Shock Tube



PPM has special algorithms for these features

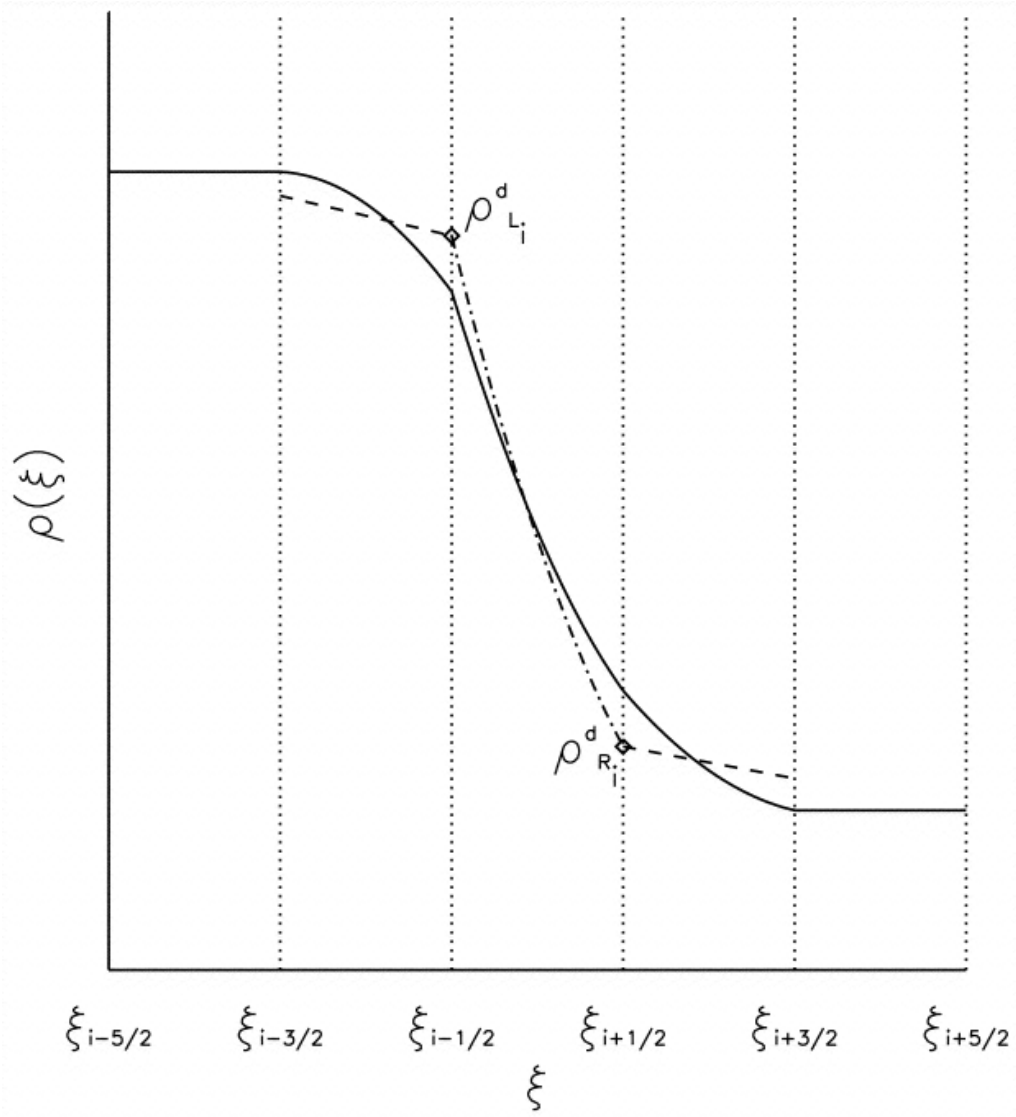
Contact Discontinuity Detection



Density profile that will trigger special algorithm

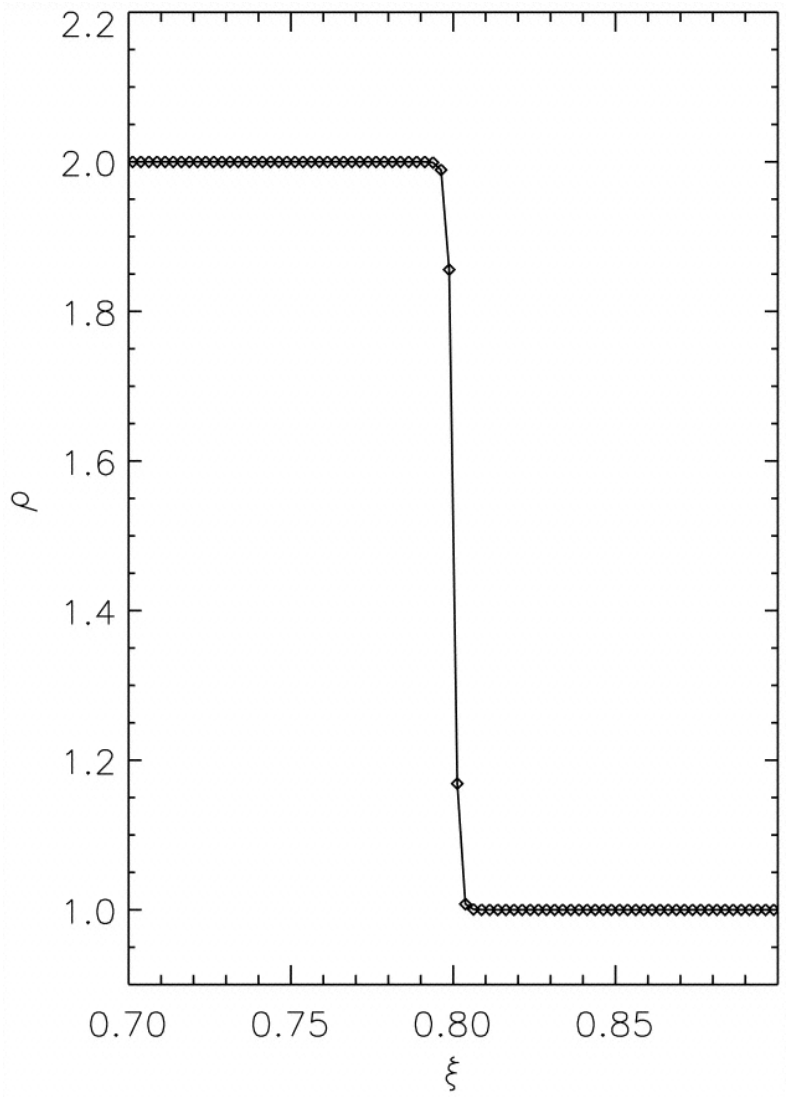
From Fryxell, et al. 2000

Contact Steepening Process



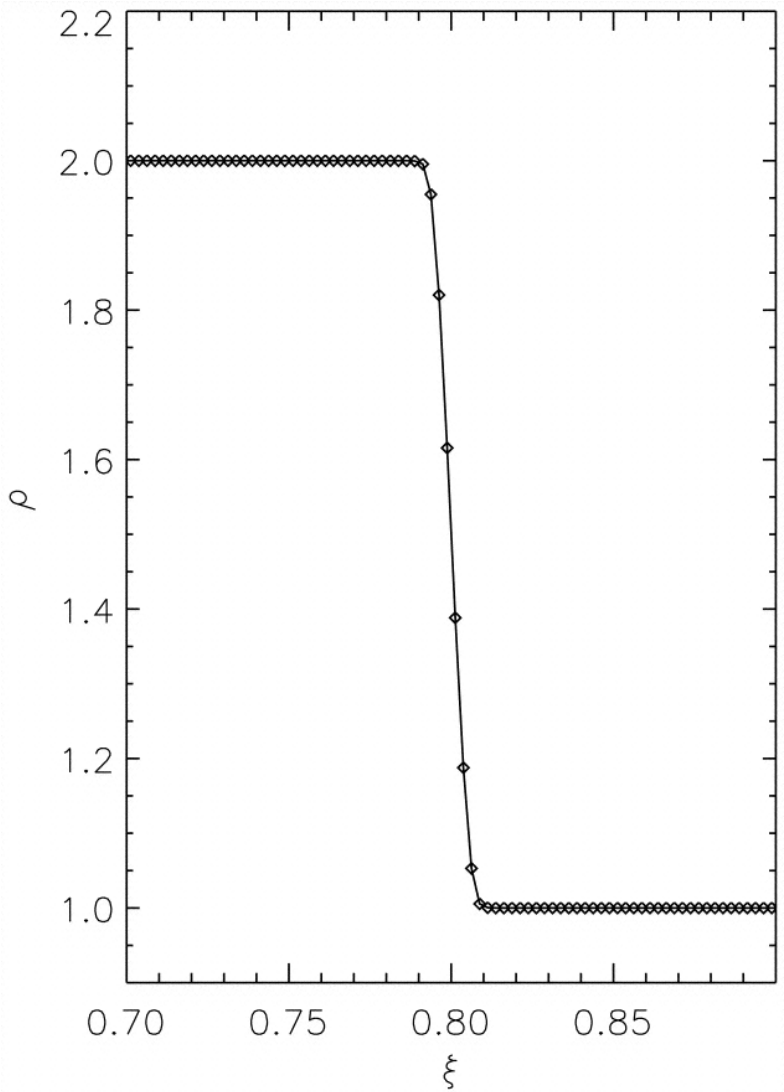
From Fryxell, et al. 2000

Contact Discontinuity w/ steepening



From Fryxell, et al. 2000

Contact Discontinuity w/o steepening



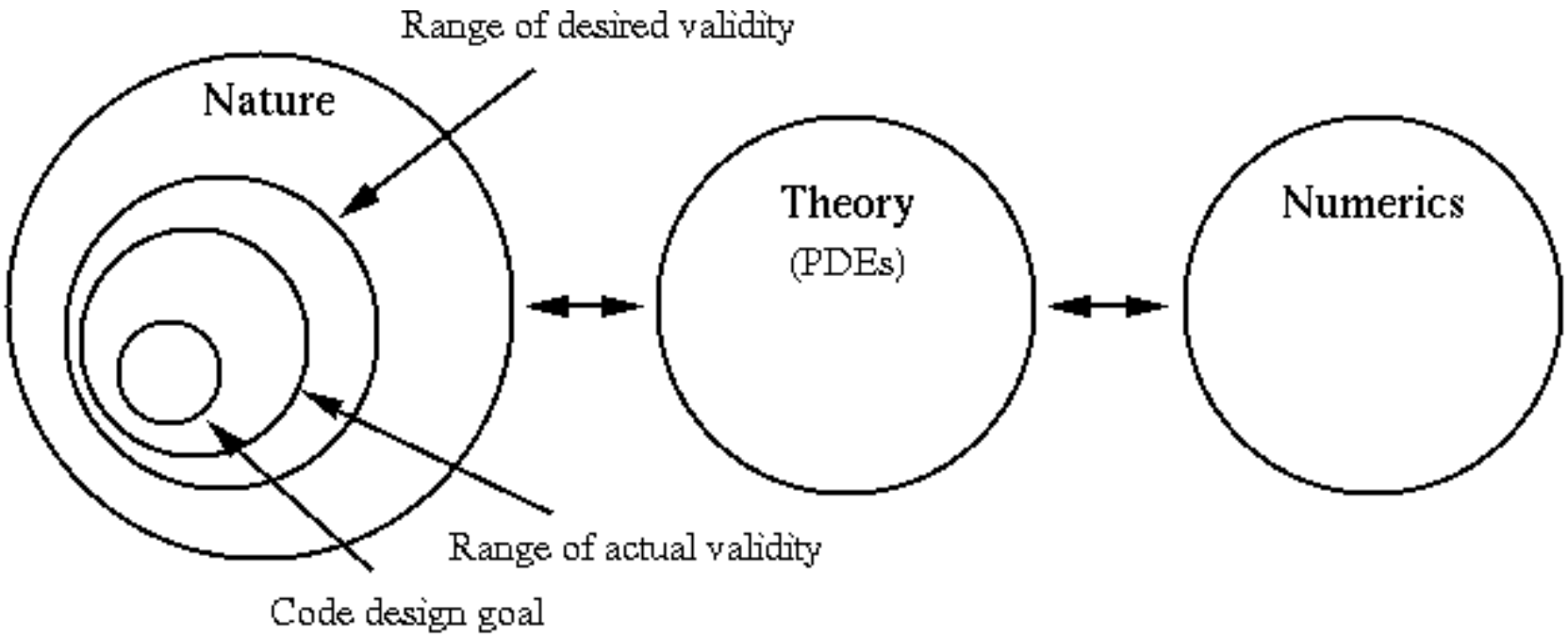
From Fryxell, et al. 2000

Intermission

V&V Vocabulary

- Verification- Determining that the implementation accurately represents the conceptual description of the model.
  - Quantify error.
  - Approach is a systematic refinement study (space and time).
- Validation- Determining the degree to which a model is an accurate representation of the real world.
  - Test key elements (modules) and integrated code.
  - Compare to actual experiments (quantify error).
- Calibration- process performed to improve the agreement between simulation and experiment. Not validation!
- Prediction- use of the code or model for an application for which it has not been validated.

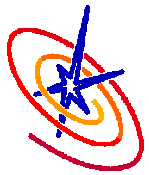
Verification and Validation



Verification: “solving the equations right”

Validation: “solving the right equations”

From Calder, et al. 2002



Our V&V Methodology

STONY BROOK UNIVERSITY

- Choose V&V tests/problems for particular code modules e.g. hydrodynamics. Test integrated code also if possible.

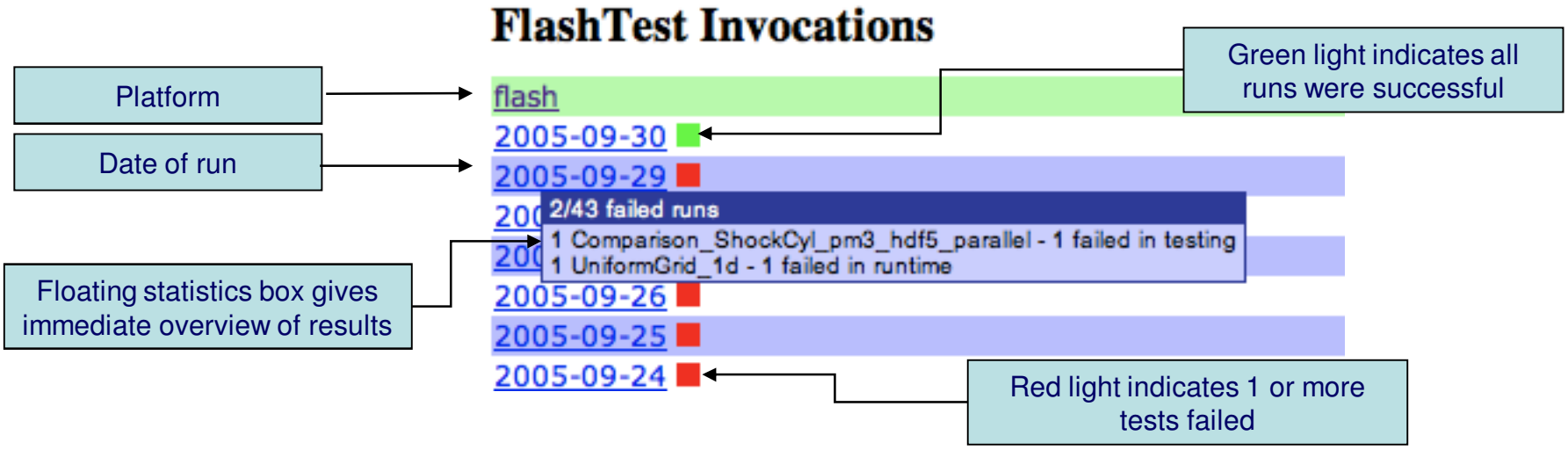
- Verification test problems
  - Unit tests
  - Investigate convergence of error with resolution
  - Investigate error in secondary modules e.g. EOS
  - Regularly re-verify with nightly/weekly automated tests

- Validation problems
  - Quantify measurements in experiment and simulation
  - Quantify error and uncertainty in experiment and simulation
  - Resolution study (verification-type tests)

- Note: astrophysics is largely prediction.

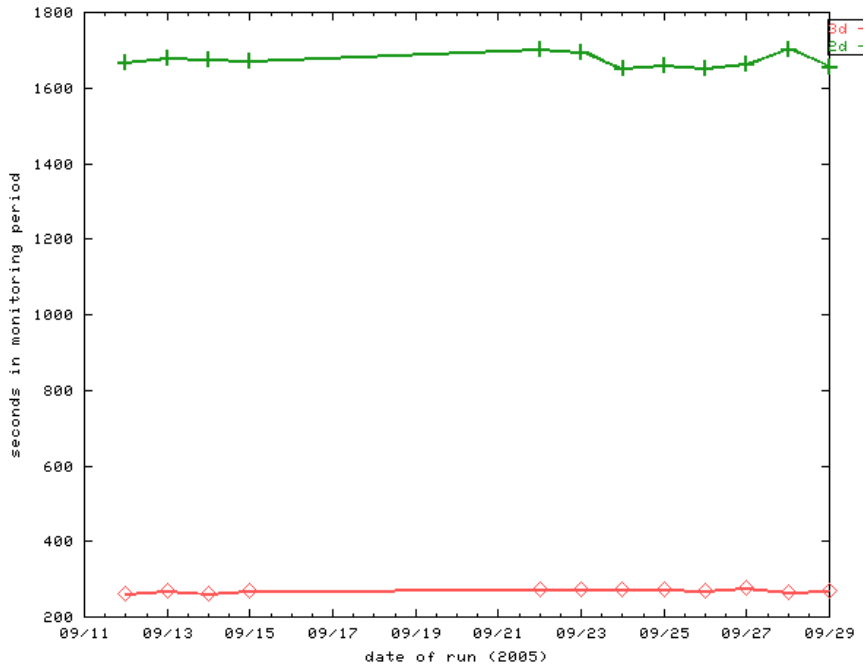
Verifiability: The New Test Suite

- Flash test suite automatically runs set of verification tests
- Essential for identifying bugs unintentionally introduced
- New test suite released with FLASH3 so users can monitor their own research problems and performance
  - test parameters easily modified through GUI
  - handles unit tests, compilation tests, comparison tests
  - automatically uploads test suite data to benchmarks database

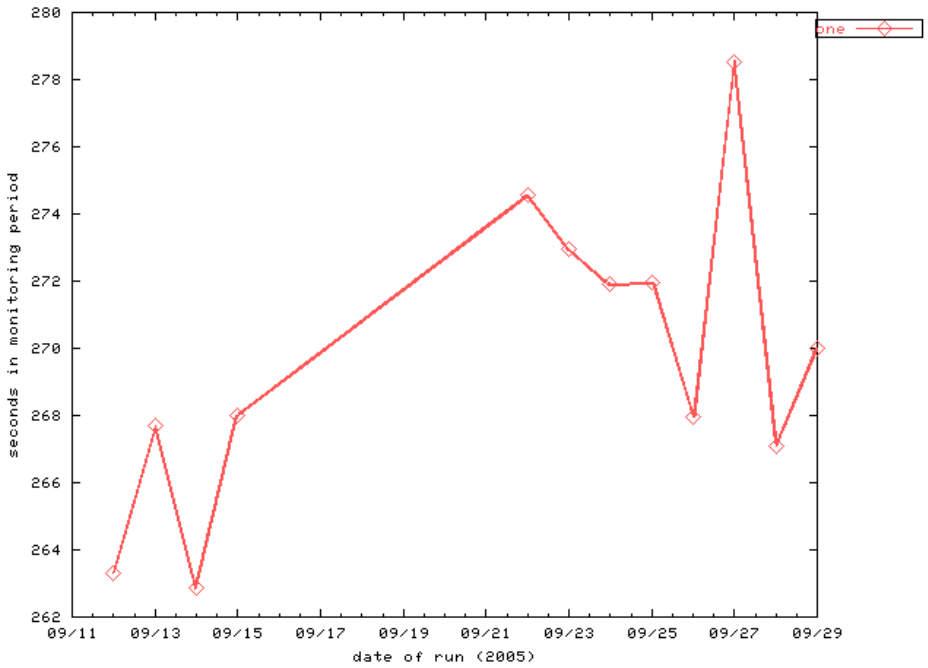


Test Suite Performance History

History of 2D and 3D Sedov Problems



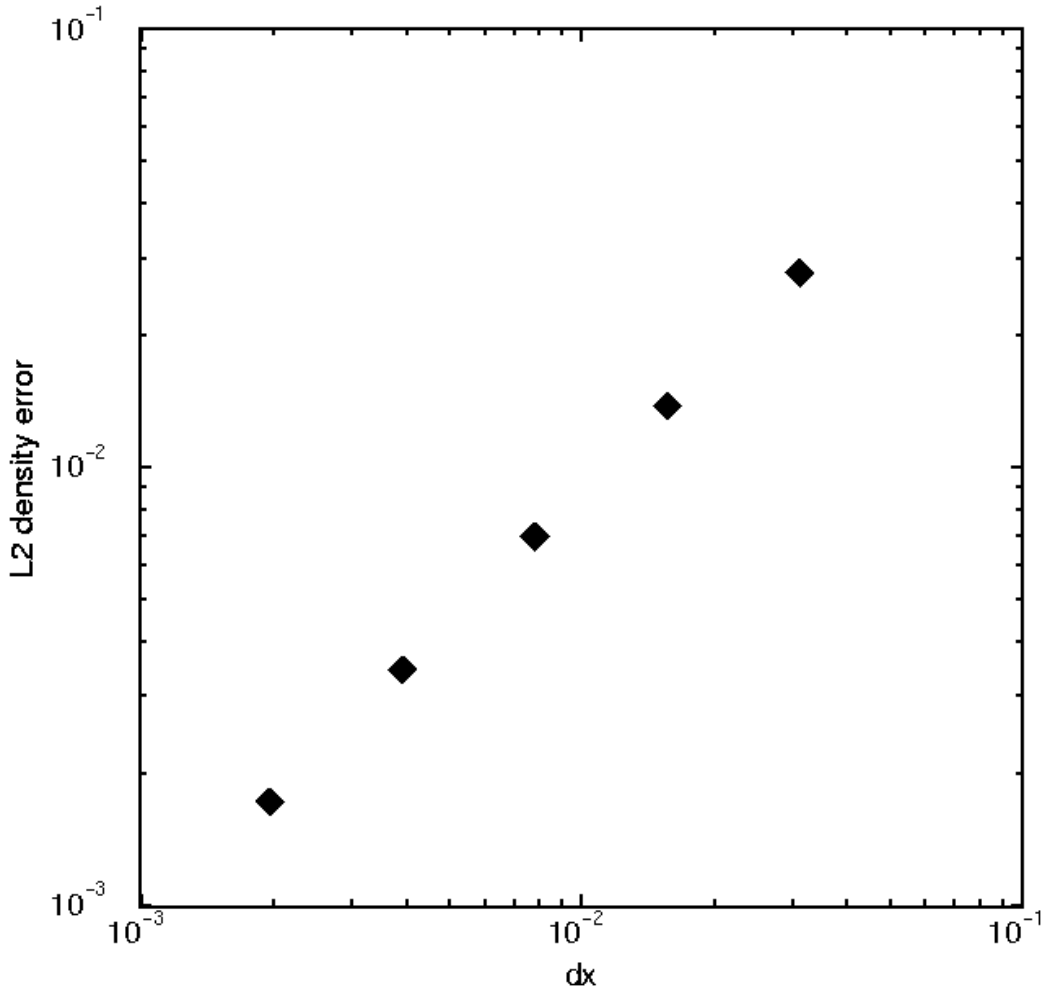
Details of 2D History



The test suite adds the ability to upload selected test log files to the database to track daily code performance.

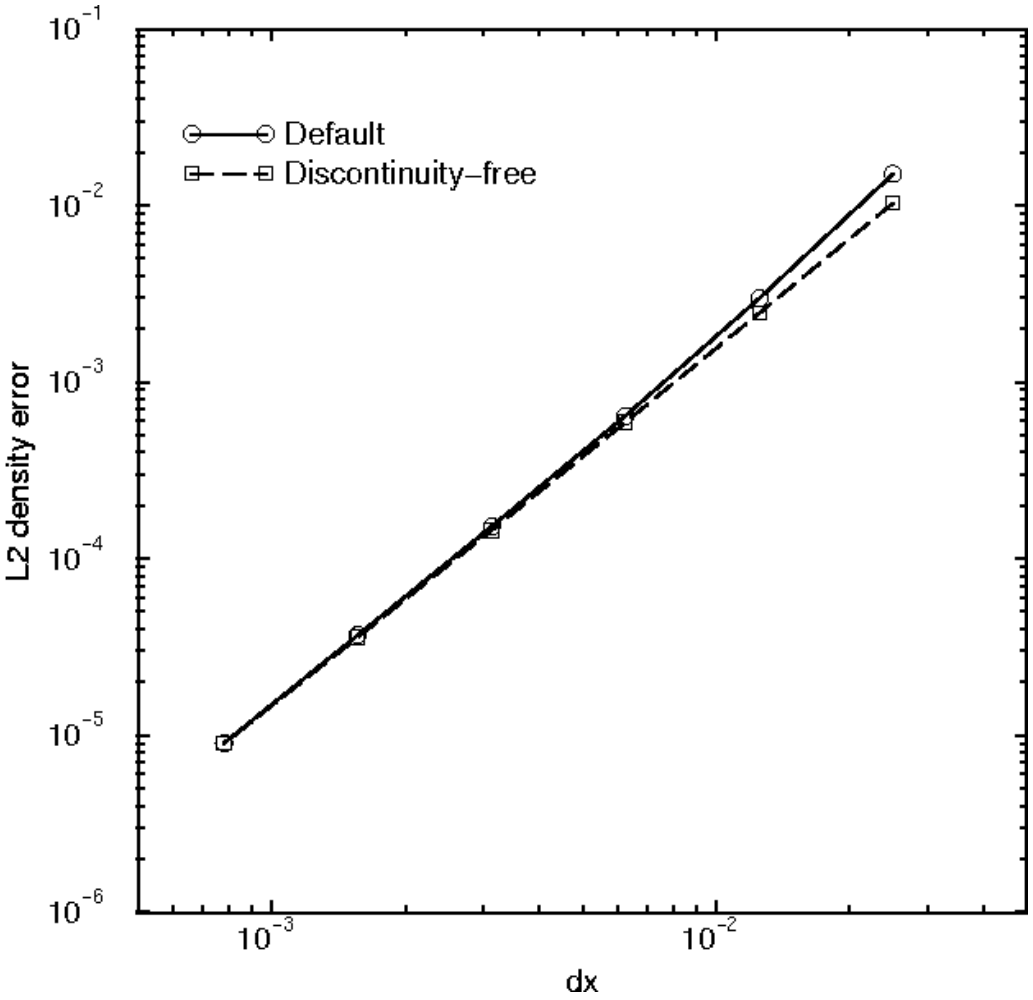
Verification Test: Sod Shock Tube

$$L_2 = \left[\frac{\sum \left(|\phi - \phi_a| \right)^2}{\sum \left(\phi_a \right)^2} \right]^{\frac{1}{2}}$$



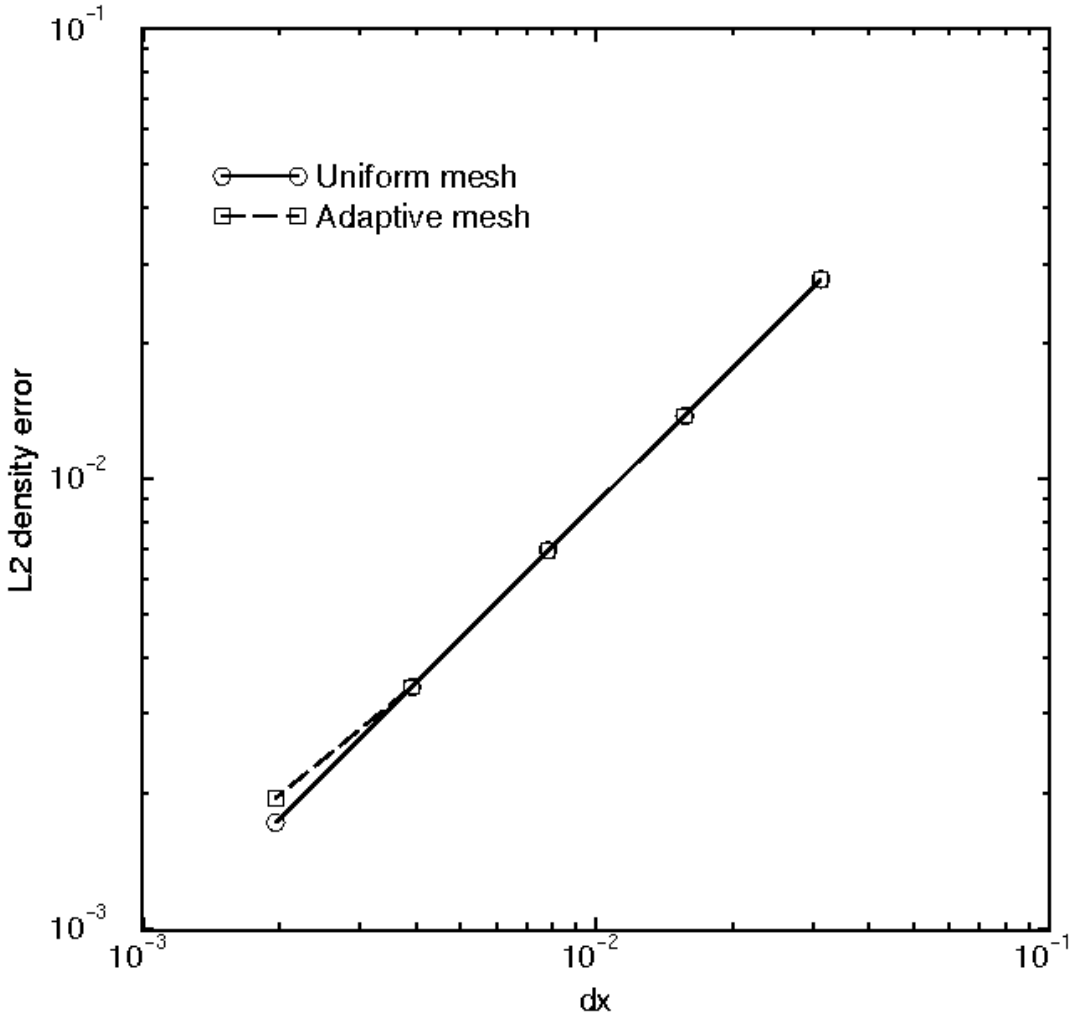
Demonstrates expected 1st order convergence of error

Verification Test: Isentropic Vortex



Demonstrates expected 2nd order convergence of error, but….

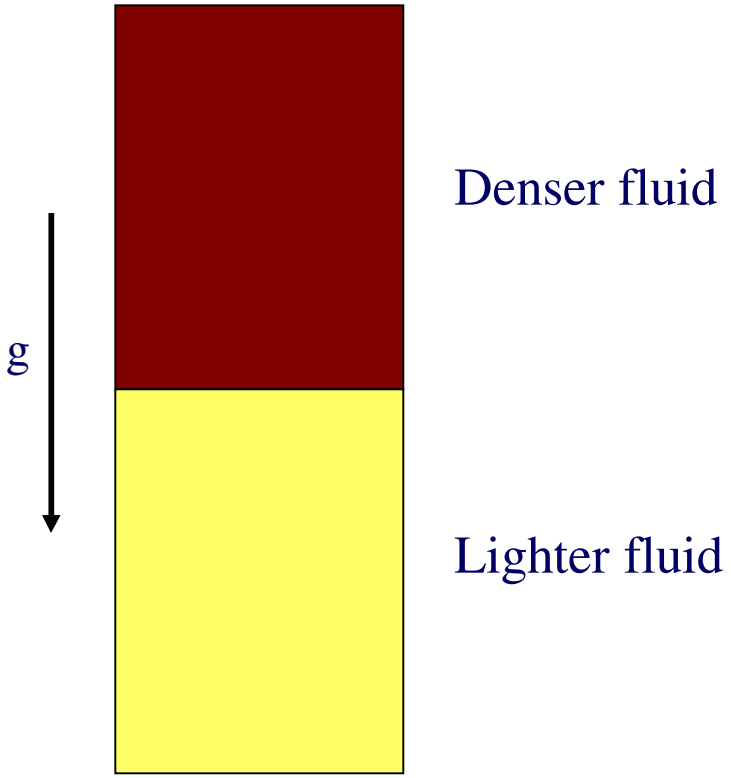
Sod Tube W/ AMR



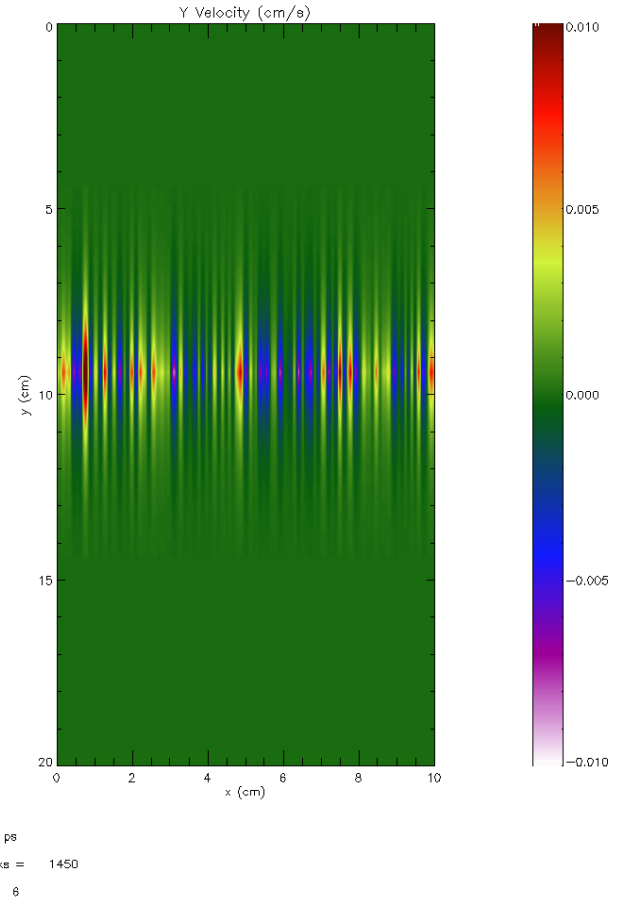
Demonstrates expected 1st order convergence of error, but note that the imperfect mesh refinement criteria degrades the solution!

Rayleigh-Taylor Instabilities

Density schematic:



Multi-mode velocity perturbation:



2.5-5 % sound speed with highest magnitude near the interface

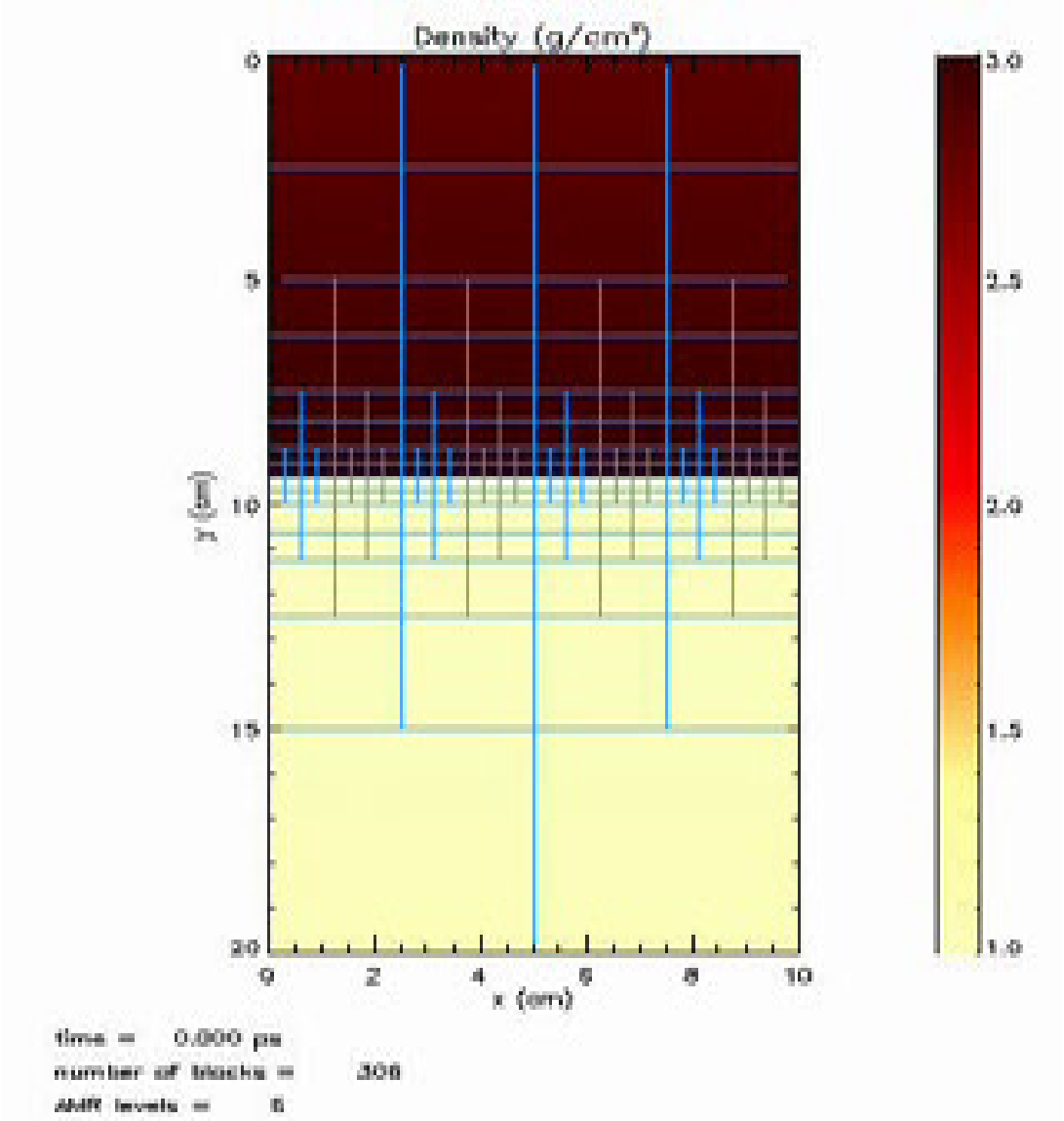
STONY BROOK UNIVERSITY

"α-Group" Consortium

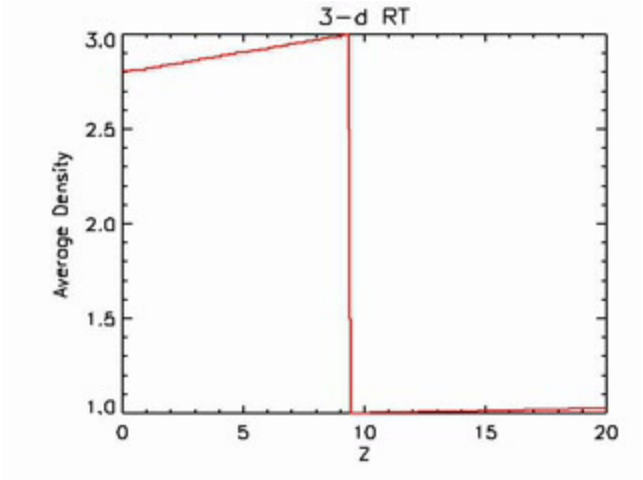
- Organized by G. Dimonte (Oct. 1998)
- Purpose – to determine if the t^2 scaling law holds for the growth of the RT mixing layer, and if so, to determine the value of α
  - simulation - experiment comparisons
  - inter-simulation comparisons

    $h_{b,s} = \alpha_{b,s}\, gAt^2$, where $A = (\rho_2 - \rho_1)/(\rho_2 + \rho_1)$
- Definition of standard problem set (D. Youngs)
- Dimonte et al. Phys. Fluids **16** 1668 (2004)

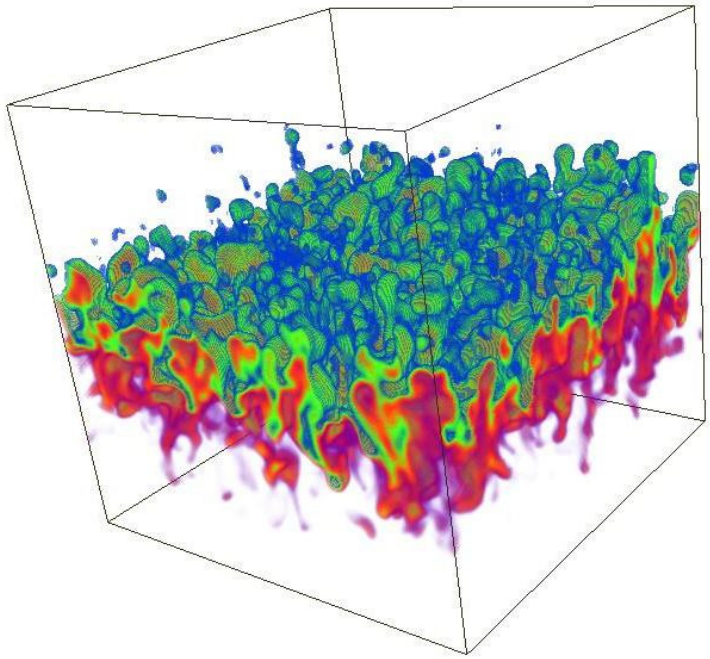
Multi-mode Rayleigh-Taylor: 2-d Simulation



Multi-mode Rayleigh-Taylor: 3-d Simulation



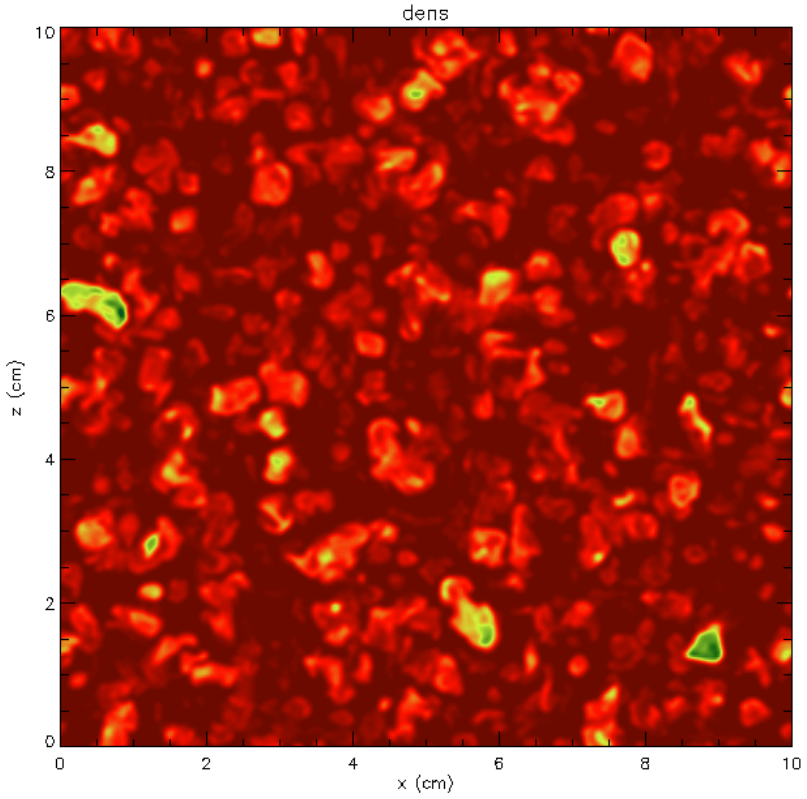
Horizontally Averaged Density



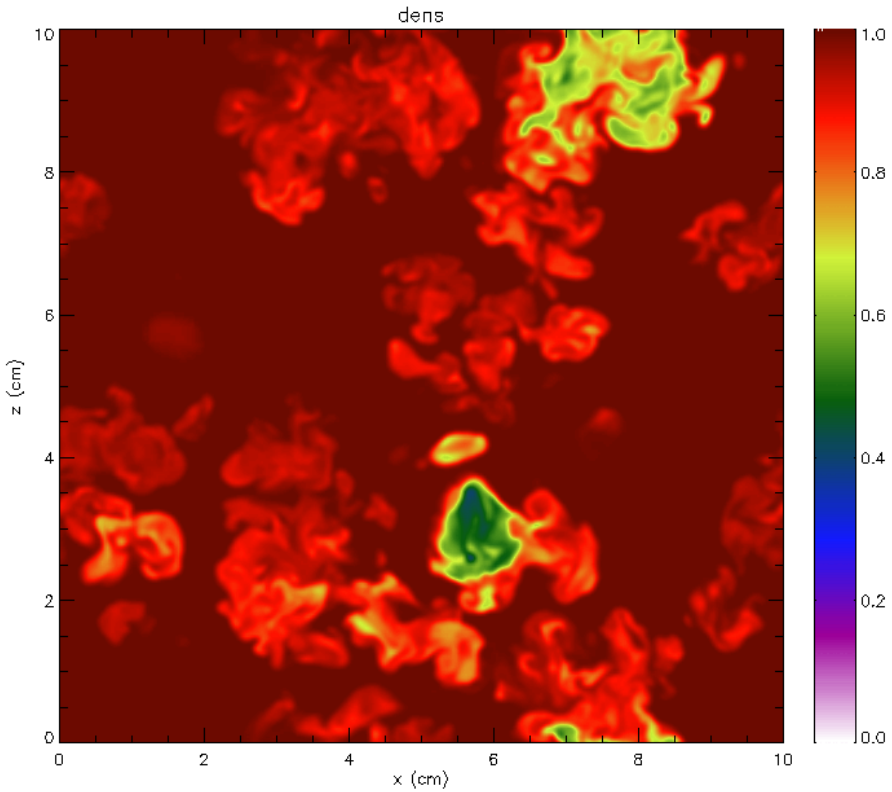
Modes 32-64 perturbed

Multi-mode Rayleigh-Taylor: Inverse Cascade

Bubbles of the lighter fluid in the denser fluid



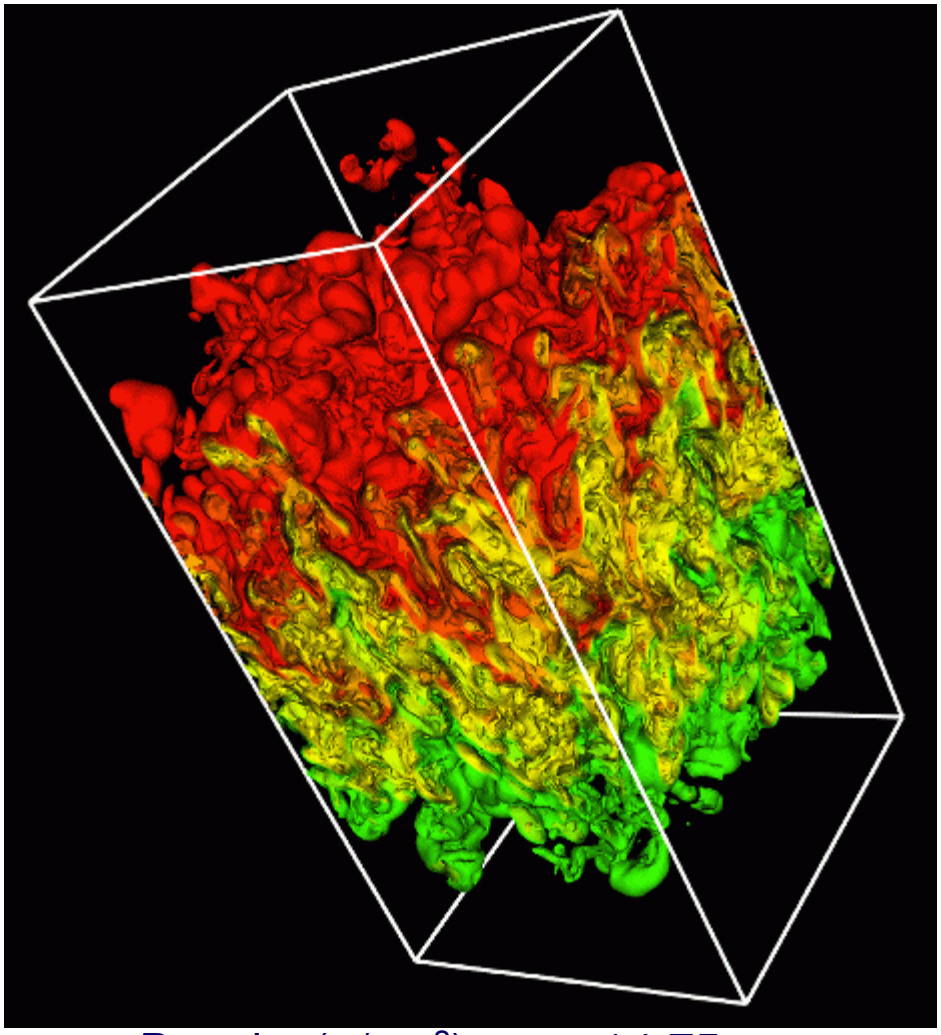
t = 7.00 sec



t = 14.75 sec

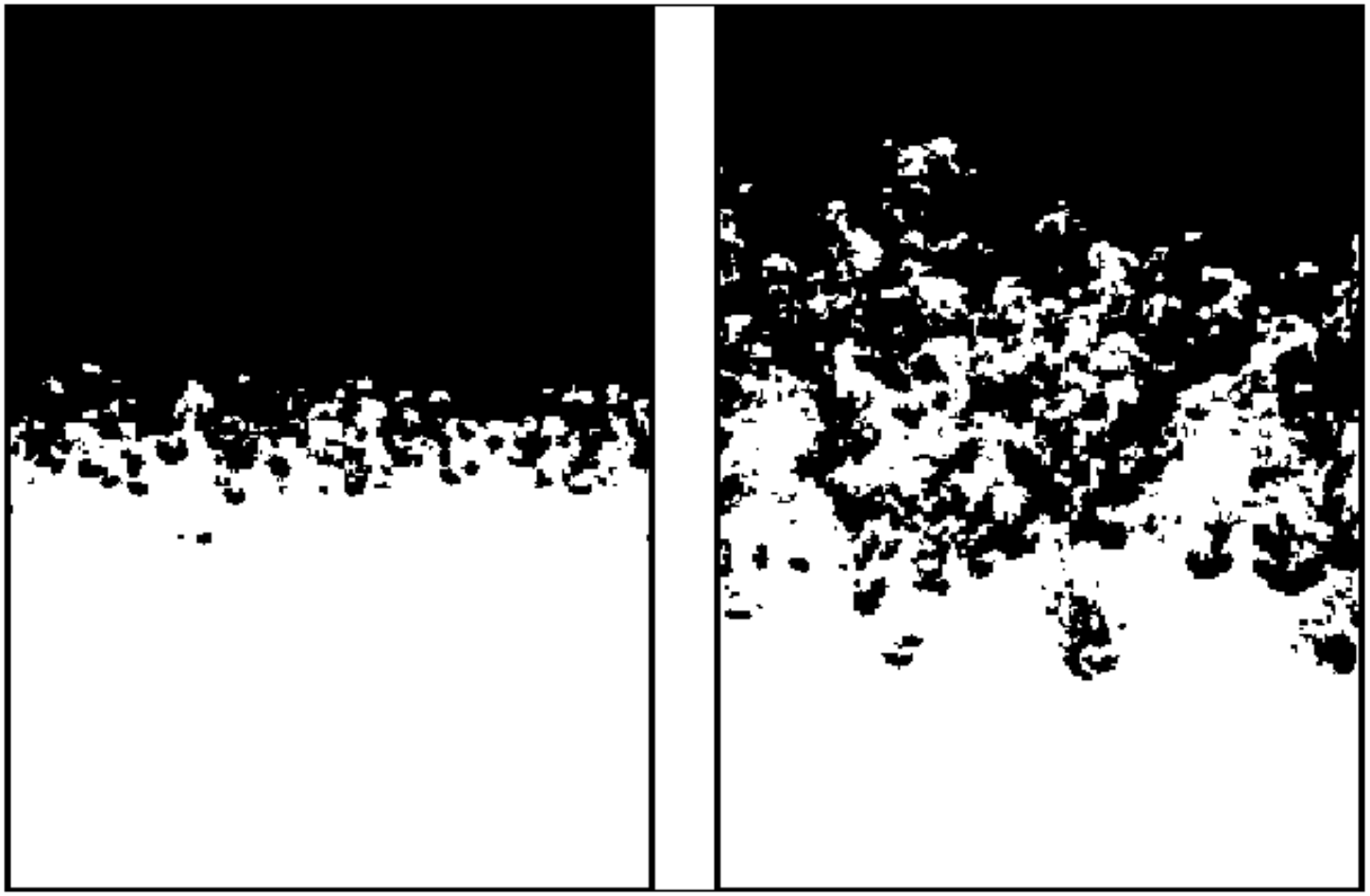
Multi-mode Rayleigh-Taylor

Rendering of Mixing Zone



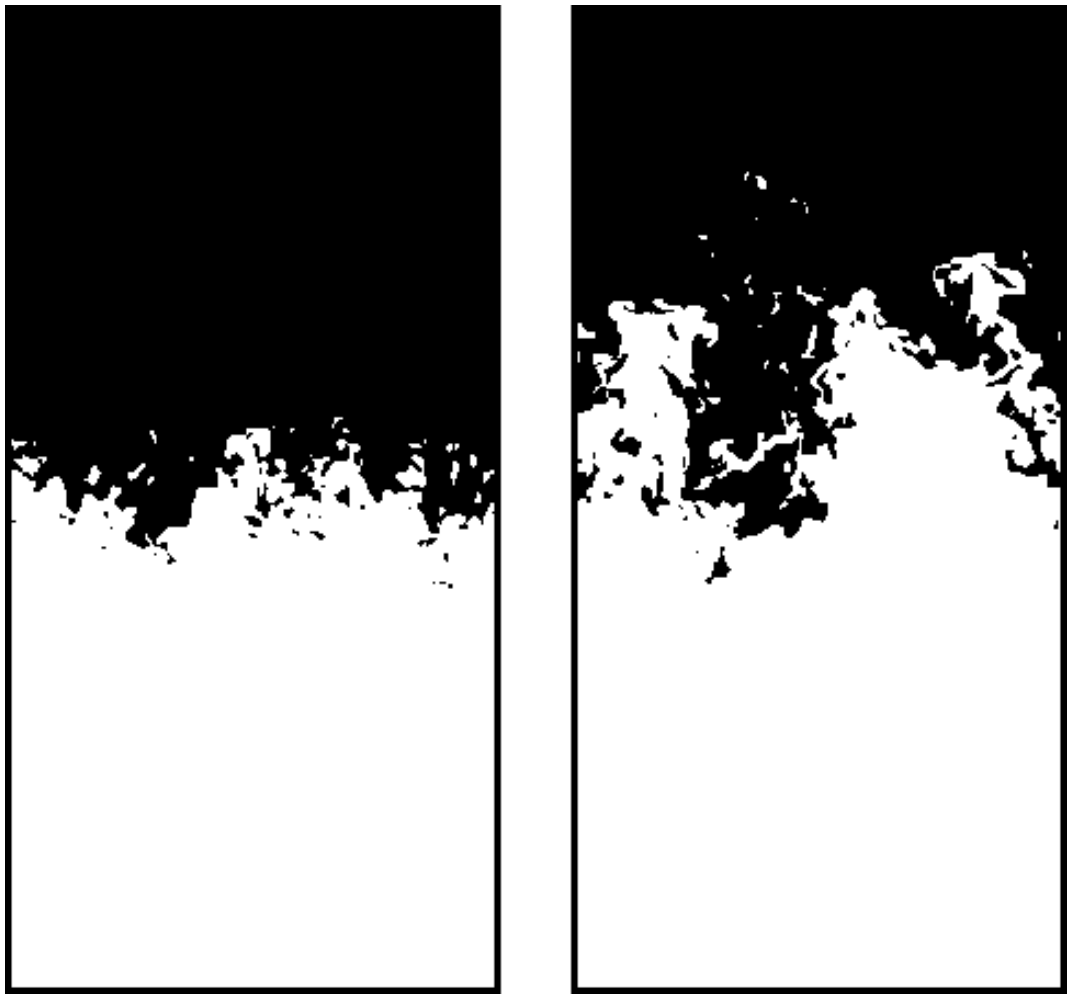
Density (g/cm^3) at t = 14.75 sec

Multi-mode R-T Experimental LIF Image



It looks similar to the simulation…..

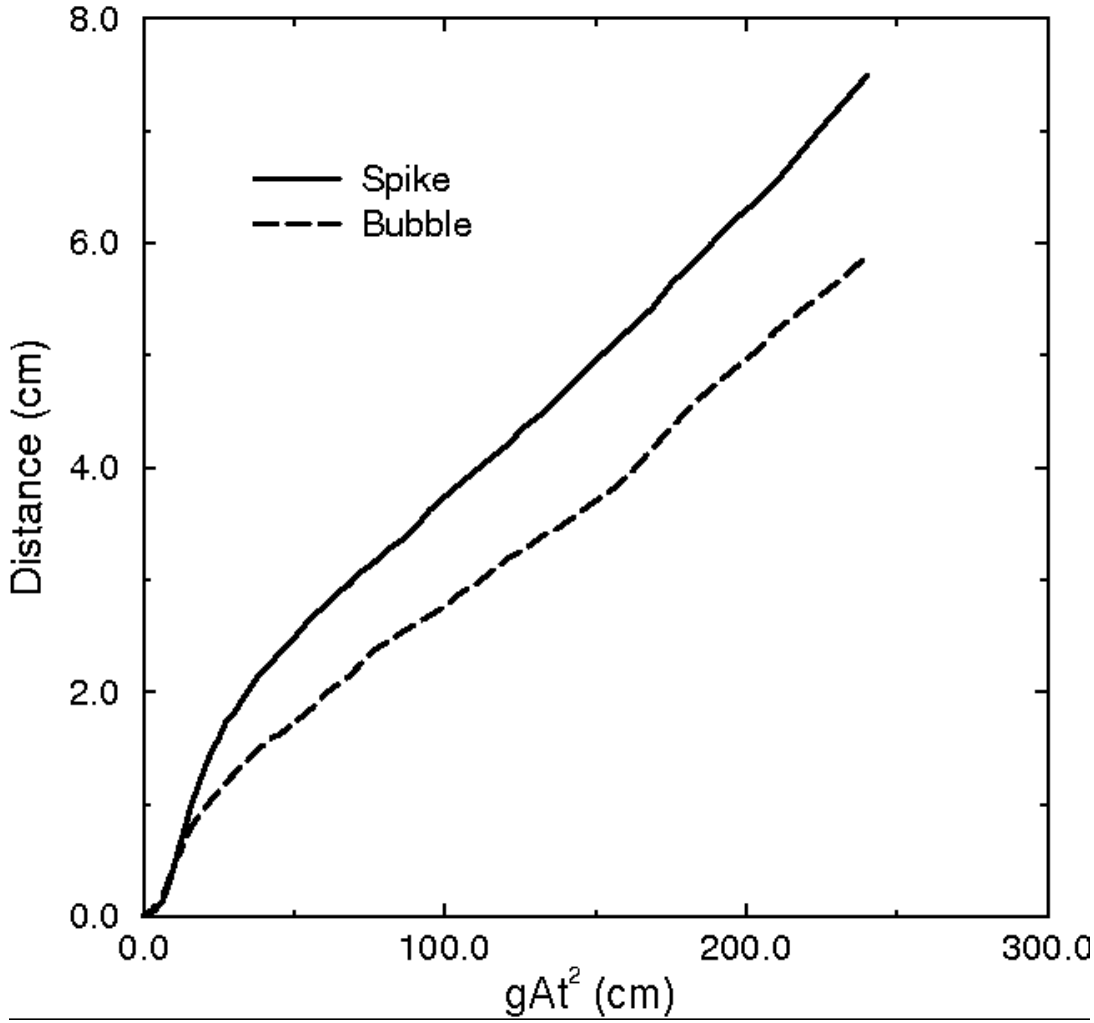
Multi-mode R-T Simulated LIF Image



It looks similar to the experiment…..

Multi-mode Rayleigh-Taylor

FLASH Simulation

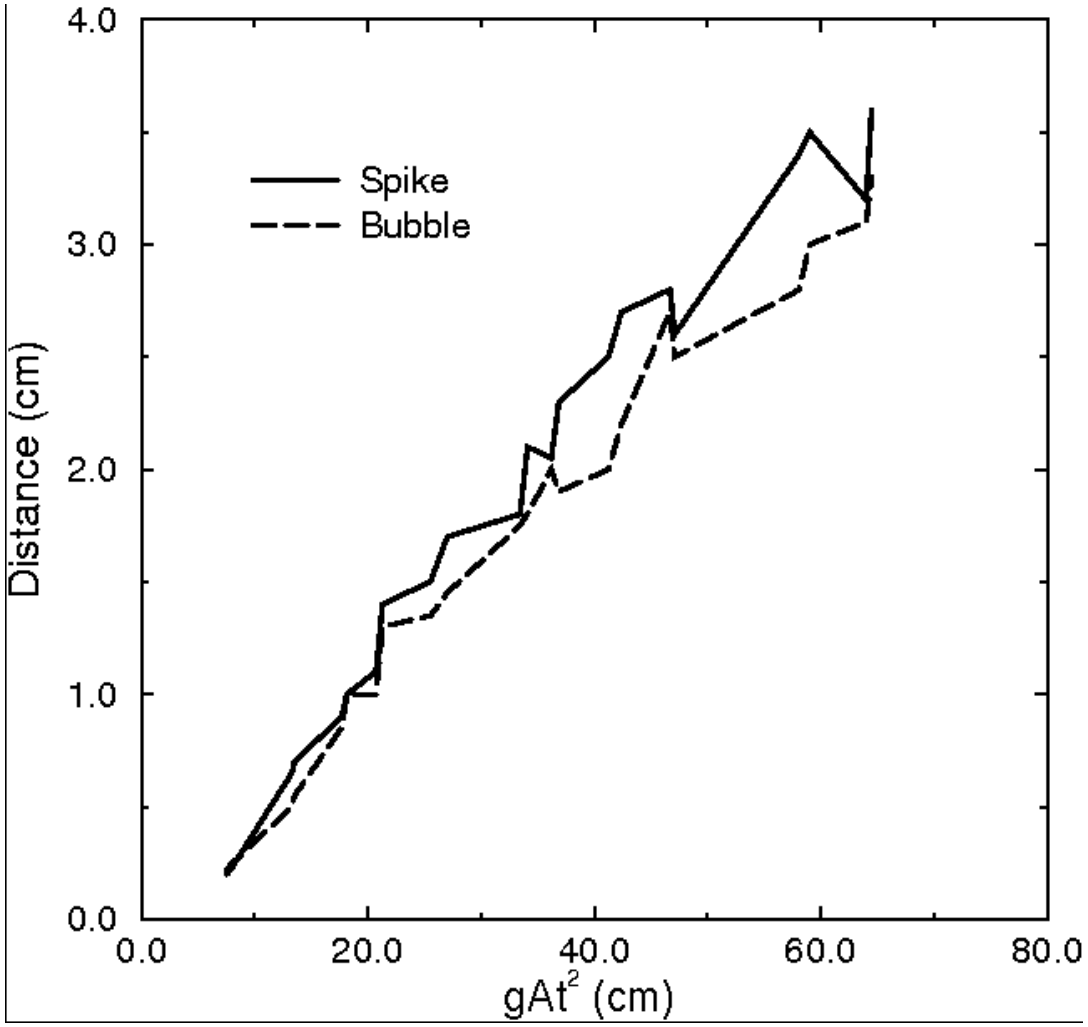


$\alpha_{spike} = 0.026$

$\alpha_{bubble} = 0.021$

Multi-mode Rayleigh-Taylor

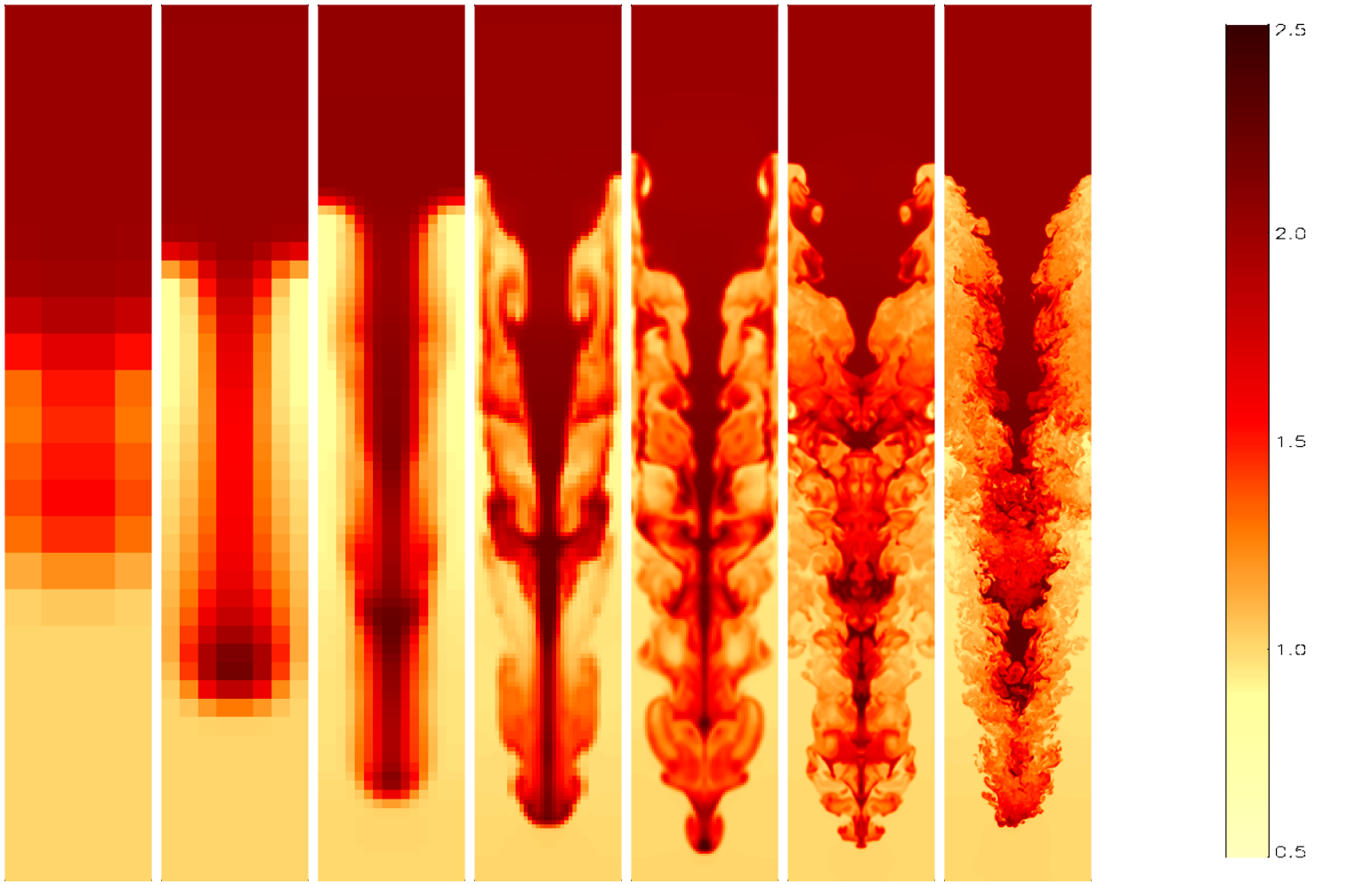
Experiment

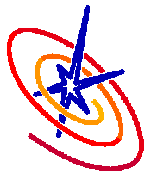


$\alpha_{spike} = 0.058$

$\alpha_{bubble} = 0.052$

Single-mode 3-D Rayleigh-Taylor

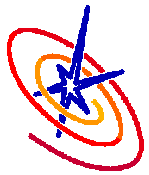




Flash Exercises

STONY BROOK UNIVERSITY

- Set up and run the cellular detonation problem. Visualize the results with VisIt.
- Investigate differences between hydrodynamics solvers for the case of the Kelvin-Helmholtz instability.
- Two SN Ia related problems- thermonuclear flame and a detonation in a white dwarf.



Thermonuclear Flame Setup

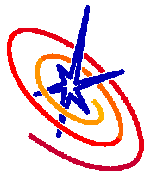
STONY BROOK UNIVERSITY

- Homework assignment: Starting with the Cellular detonation problem setup, modify it to simulate a deflagration.

- Hints:
  - What resolution?
  - How is a deflagration different from a detonation? More on this in the next lecture.
  - Only two files to modify:

    `Simulation_initBlock.F90`

    `flash.par`
  - Will need to use the (explicit) diffusion module. Do so with an option to the setup command.
  - `-with-unit=physics/Diffuse/DiffuseFluxBased`



White Dwarf Detonation Setup

STONY BROOK UNIVERSITY

- Example of setting up a white dwarf model with a detonation.
- Hydro + burning + self-gravity.
- Test: Turn off burning. How long will the code hold the model steady?
- Note that there may be issues!
- Name of setup is SnDet.

```
./setup SnDet -2d -auto -nxb=16 -nyb=16 +cylindrical
  -objdir=obj_SnDet
```

- Problem directories are in `source/Simulation/SimulationMain`

…and that leads us to

QUESTIONS AND DISCUSSION