Approved for Unlimited Release
LA-UR-15-26973

The LANL Code Verification Test Suite

Scott W. Doebling, Daniel M. Israel,
James R. Kamm, Robert L. Singleton, Jr.

Computational Physics Division (XCP)

Correspondence:
Scott W. Doebling
doebling@lanl.gov

MultiMat, Sept 7-11, 2015
Würzburg, Germany

The LANL Code Verification Test Suite

- Test accuracy and convergence of multi-physics codes
- Follow a staged delivery life-cycle
  - Requirements, Design, Implement, Test
- Support multi-physics code developers and end-users

Requirements

- Manage code verification test problems
- Consistency, repeatability, pedigree
- Compatible with multiple multi-physics codes
- Interoperability with existing LANL verification and validation systems
- Usable and maintainable by the LANL code end-user communities

Combinatorial effect of requirements

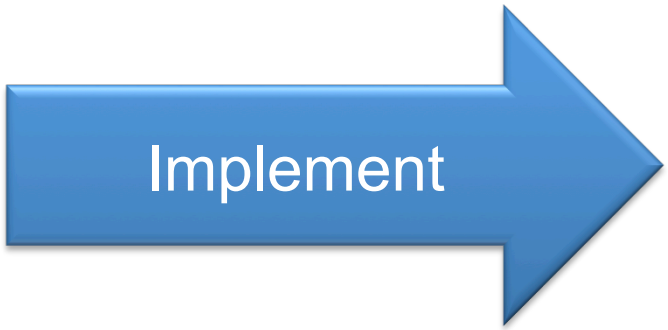
- Physics areas (~10)
- Test problems (10+ per physics area)
- Model and parameter options (~?)
- Physics codes (~5)
- Dimensionality (3-5)

Design

= A lot of things to do!

Design

- ExactPack
- Verification Test Suite
- Document-driven V&V framework



ExactPack: An open source Python package

- Test problem analytic, semi-analytic, and manufactured solutions
- Code verification analysis tools
- Designed to enable adding new solvers

Will be released as
Open Source Software at

`www.github.com/losalamos`

ExactPack contains solvers for many common verification problems

- Compressible hydro
  - Sedov, Noh, Guderley, Riemann shock tube (9 variants)
- Hydro + conduction/radiation
  - Coggeshall
  - Su-Olson
  - Reinicke Meyer-ter-Vehn
- Reactive flow
  - Escape of HE products
  - Mader
  - Steady Detonation Reaction Zone

Sedov

Riemann

Coggeshall

Your pic here?

ExactPack has several more solvers under development

- Hydro + conduction/radiation
  - Lowrie-Edwards non-equilibrium diffusion
  - Plasma diffusion
- Reactive flow
  - Several programmed HE burn tests
  - Generalized steady ZND wave tests
- Smooth hydro
  - Noh's uniform collapse problem
- Solid Mechanics
  - Blake problem

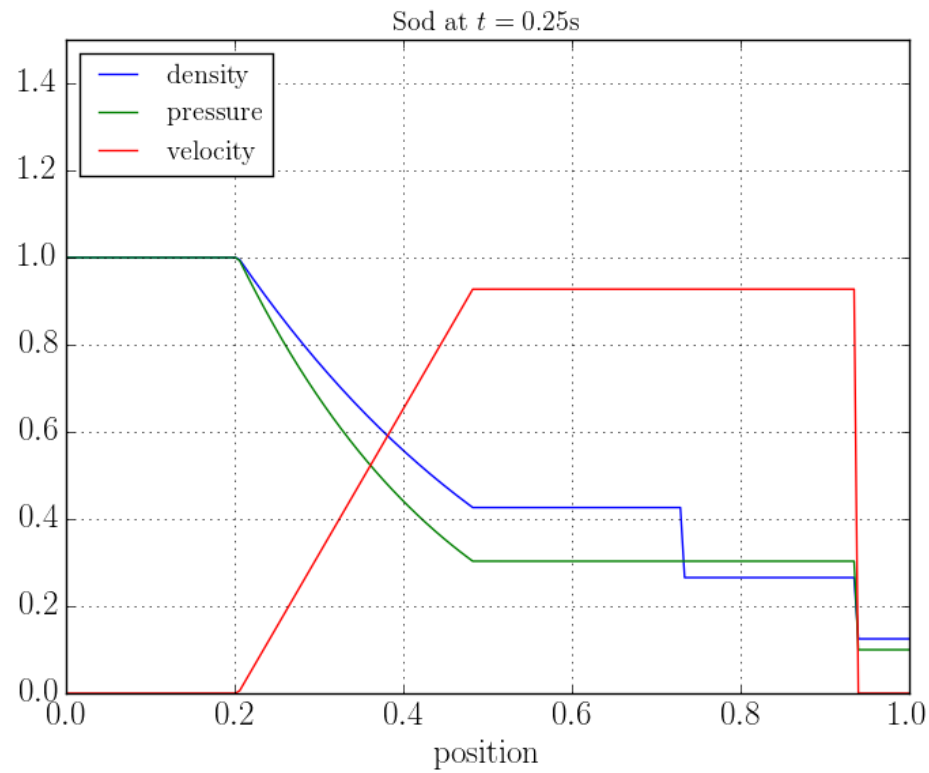
ExactPack Use case: Generate and plot exact solution

```
from exactpack.solvers.riemann import Sod
```
← import Sod object

```
r = numpy.linspace(0.0, 1.0, 1000)
t = 0.25
```
← spatial array and time

```
solver = Sod()
soln = solver(r, t)
```
← **solver** object
← **solution** object

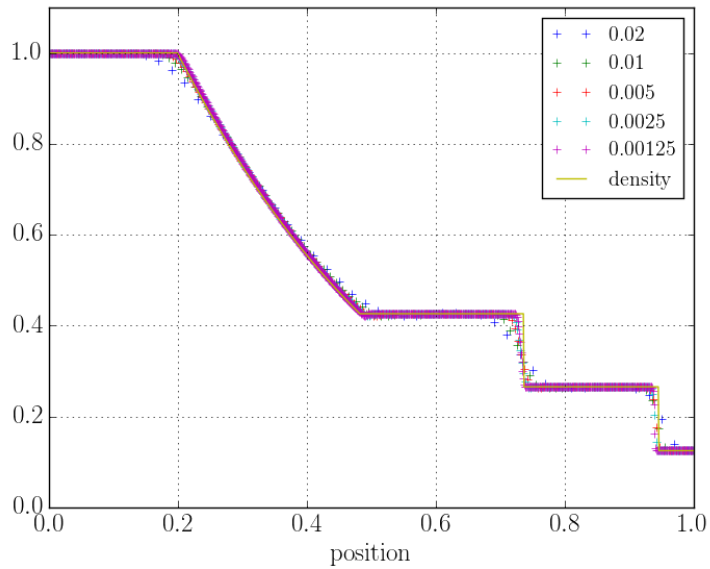
```
soln.plot('density')
soln.plot('pressure')
soln.plot('velocity')
```
solution object **plot method**



ExactPack use case: Code verification study

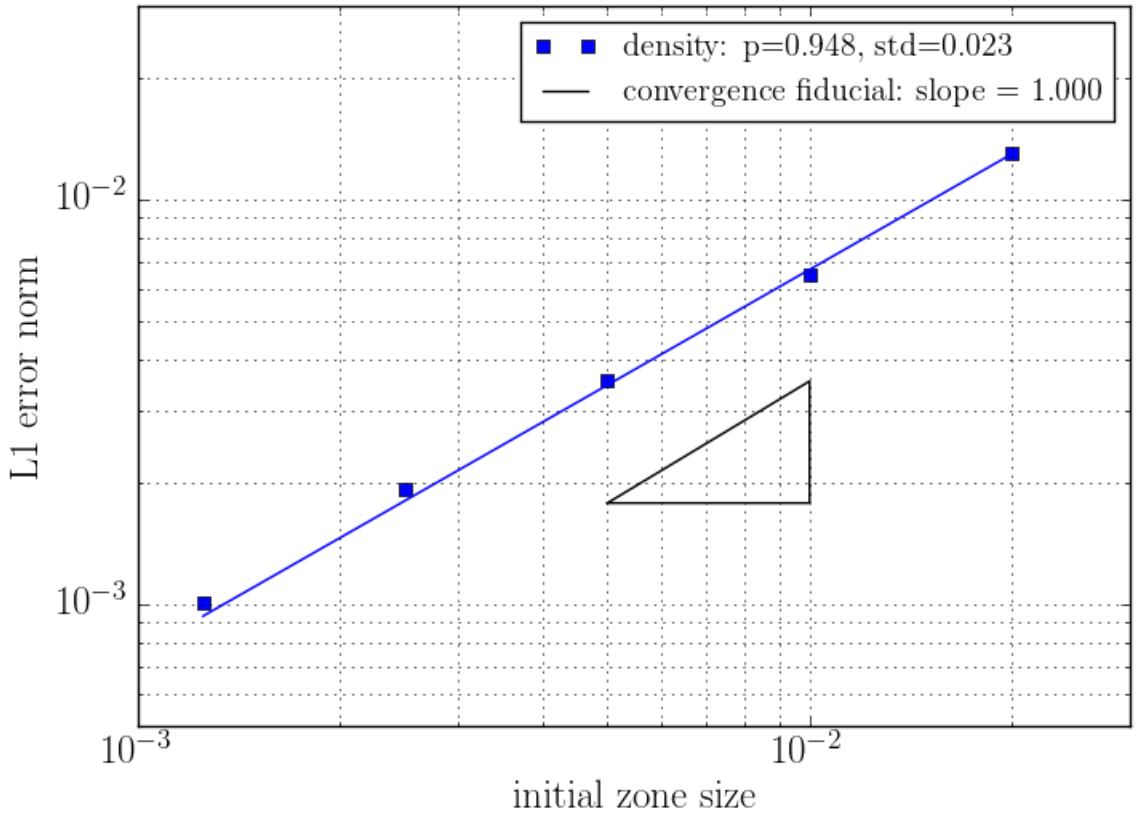
```
from exactpack.sedov.riemann import Sod
  from exactpack.analysis import CodeVerificationStudy

  study = CodeVerificationStudy(code-output,
                                Sod( ),
                                dx=[0.02, 0.01, 0.005, 0.0025, 0.00125]
                                domain=(0, 1.0),
                                reader=code-reader)
```

```
study.plot('density')
```

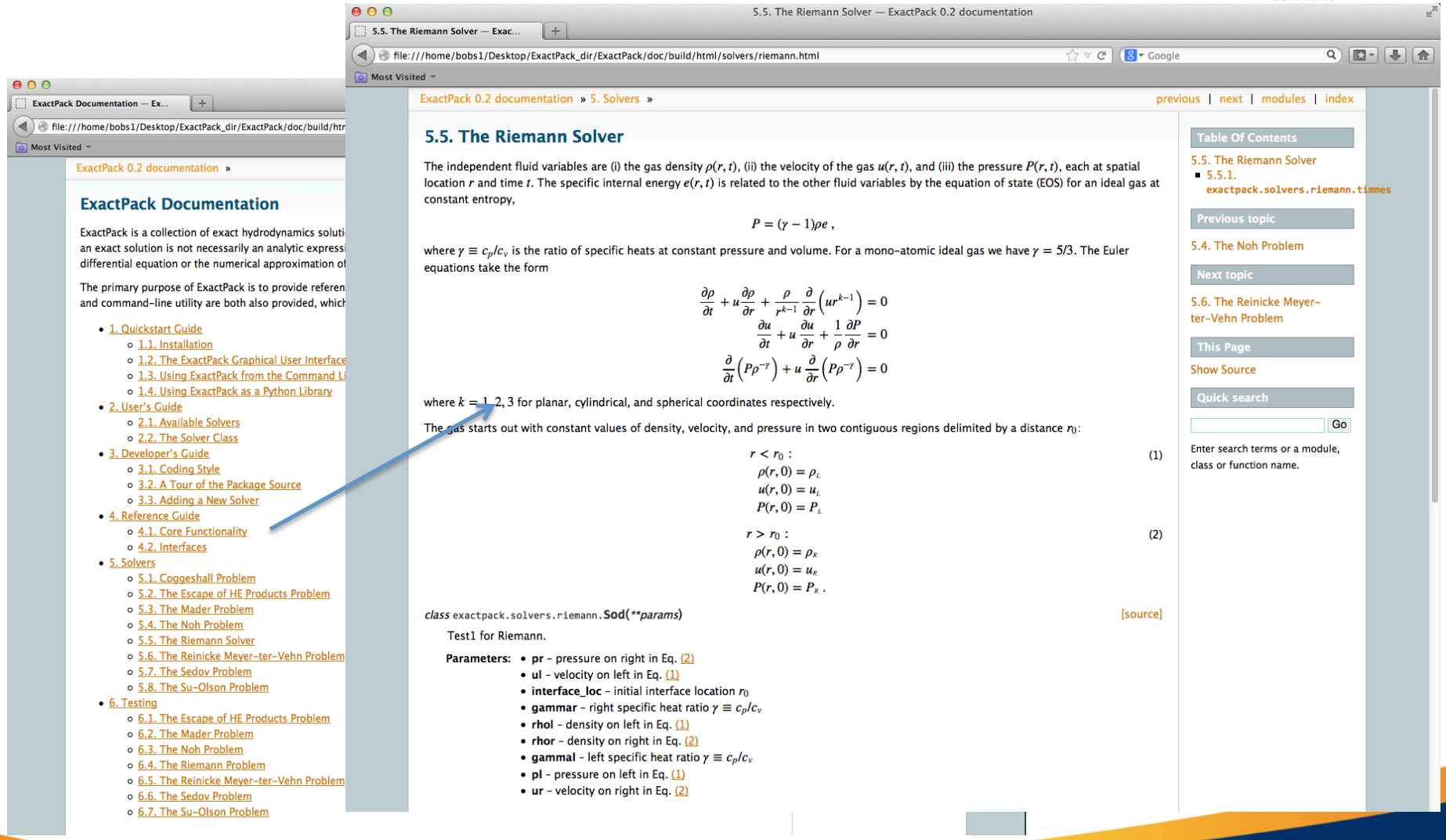


ExactPack use case: Code verification study

```
study.convergence('density').plot(fiducial=1.0)
```



ExactPack solvers are fully documented



Verification Test Suite

- Test problems definitions
- Test problem setups
- Simulation management

Example of standardized definition: The Sedov Problem

Sedov Problem

Leonid I. Sedov (1907–1999)

Description: The Sedov Problem is a mathematical idealization of a shock generated via an explosion. It consists of spherically symmetric flow of an inviscid, non-heat conducting, compressible, polytropic gas, driven by a single zone with non-trivial initial energy. This problem tests a code's ability to convert internal energy into kinetic energy and has a quasi-analytic, self-similar solution that requires one numerical quadrature.

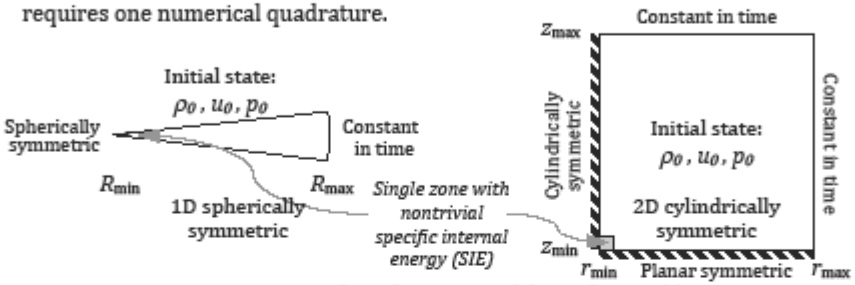


Figure 1: Initial configuration of the Sedov problem.

Table 1: Parameters for the Sedov problem.

t_{fin} [s]	γ [–]	ρ_0 [g/cm³]	u_0 [cm/s]	p_0 [dyn/cm²]	*Internal Energy* [erg]
1.0	7/5	1.0	0.0	$(2/5)\times10^{-12}$	1D: 0.851072 2D: 0.425536

Mesh*: $R_{min} = r_{min} = 0.0$, $R_{max} = r_{max} = 1.2$ cm; in 2D, $z_{min} = 0.0$, $z_{max} = 1.2$ cm.
1D spherical: $N_R = 60, 120, 240, 480$ 2D cylindrical: $N_r = N_z = 60, 120, 240, 480$

Table 2: SIE [erg/g] in first zone that gives internal energy in Table 1.

	60	120	240	480
1D	2.5397311×10^4	2.0317849×10^5	1.6254279×10^6	1.3003423×10^7
2D	1.6931541×10^4	1.3545233×10^5	1.0836186×10^6	8.6689490×10^6

Sedov Problem Results at t_{fin} = 1 s

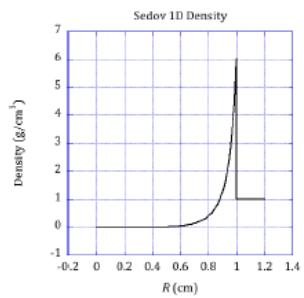


Figure 1: Sedov Problem density.

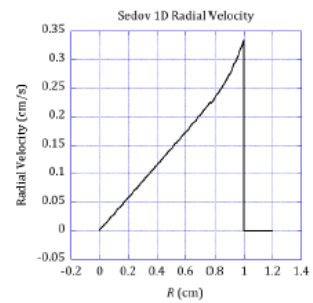


Figure 2: Sedov Problem radial velocity.

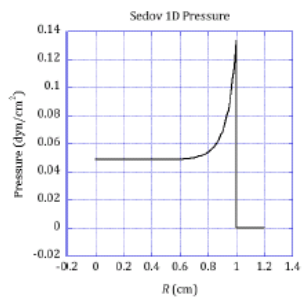


Figure 3. Sedov Problem pressure.

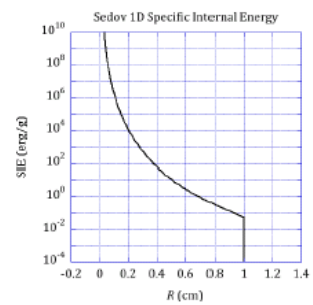


Figure 4. Sedov Problem SIE.

Definitions available in LA-UR-14-20418 (Rev. 2)

The VTS currently contains almost 400 test problem calculations

	1D	2D	3D
Noh	xRage, FLAG	xRage, FLAG	FLAG
Sedov	xRage, FLAG	xRage, FLAG	FLAG
Riemann (6 variants)	xRage, FLAG	xRage, FLAG	
Escape of HE Products	FLAG	FLAG	

- All problems use 4 or 5 grids of varying spatial resolution
- Explore code options such as AMR, CCH, and artificial viscosity settings

Simulation management is a key aspect of the VTS

- Maintain consistency of problem definitions across codes with arbitrarily different input formats
- Use parameter substitution directly into input decks
- Construct input decks using a code-independent problem abstraction plus code-specific templates
- Manage launching and monitoring of 100's of simulations

Designing a "document-driven" V&V framework

- Define desired verification study in a skeletal document
- "Make" the document, which launches the necessary calculations
- Uses Sphinx (standard Python toolbox) to generate a human-readable report in HTML or PDF

A sample verification template document

```
The Sod Problem
===============

The Sod problem is a shock tube with an ideal gas at two different
densities on either side of an interface.

.. job:: verif.sod.Sod

   The density profiles at time :math:`t=0.6` for the different meshes
   are shown in the following figure:

   .. jobreport:: profiles
      :plot:

   The :math:`L_2` error norm as a function of mesh size is shown in
   the following figure, along with the best fit curve.  The overall
   convergence rate is :jobreport:`convergence`.

   .. jobreport:: convergence_plot
      :plot:
```

section heading

regular text

suite to document

insert plot

generated text

A sample Python script defining a suite

```
class Sod(Job):
  def run(self):
    subprocess.call(['msub', 'msub.in'])

  def profile(self):
    …

  def convergence(self):
    p = …
    return "{:4.3g}".format(p)
  …
```

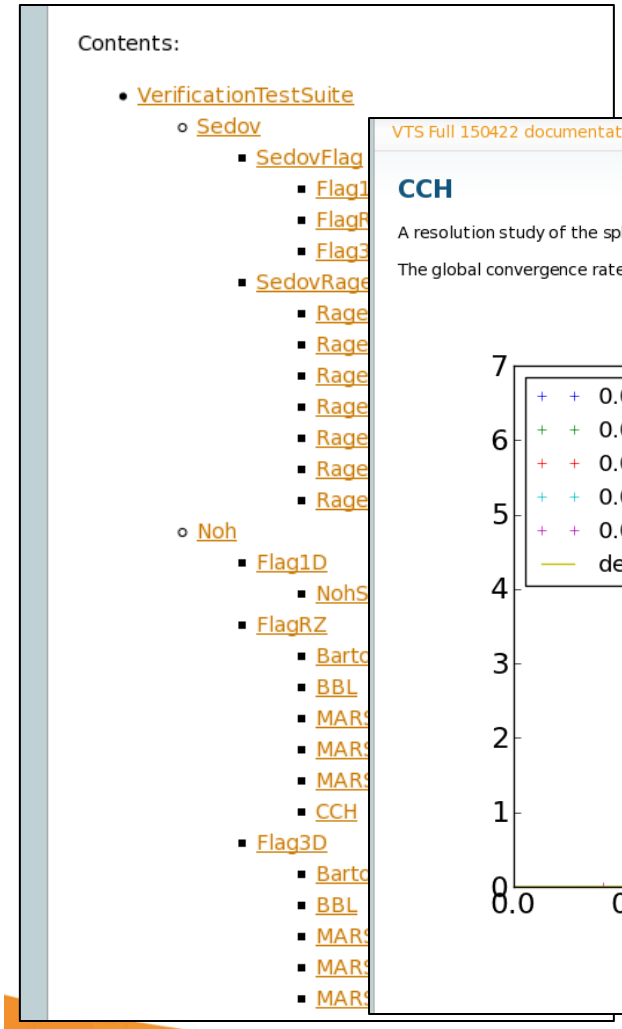
VTS provided base class defines basic functionality

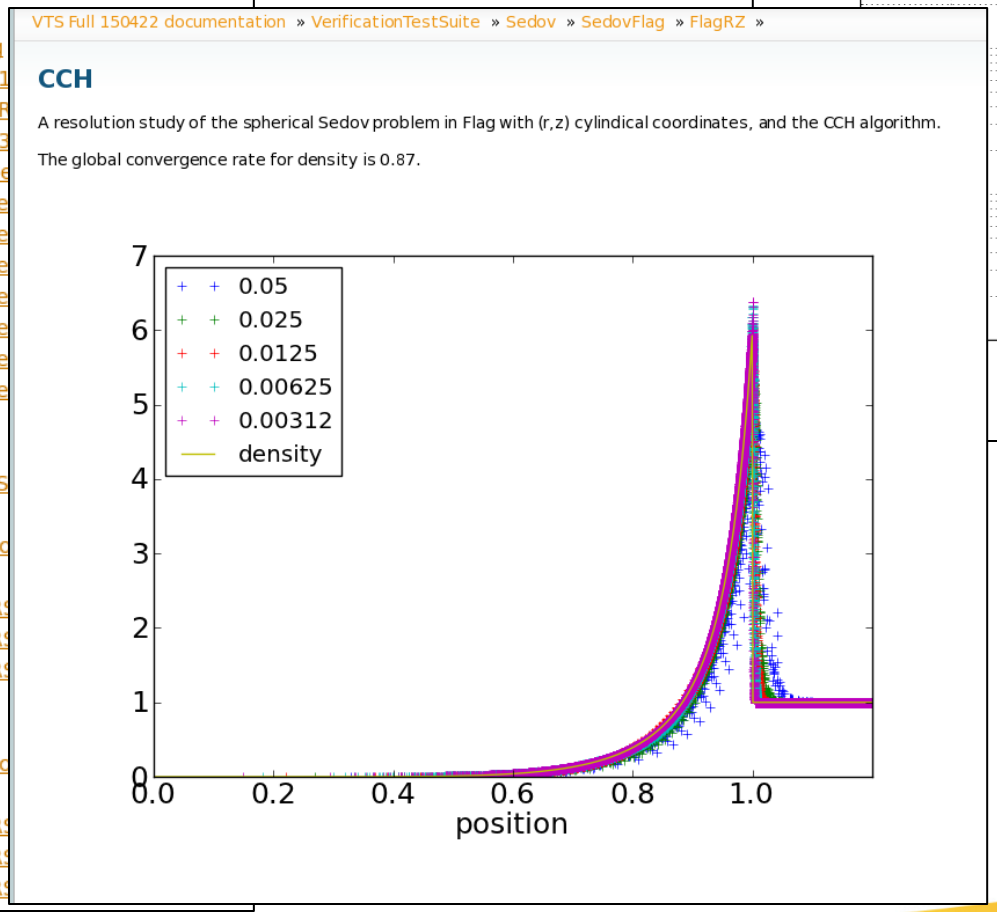
`run` method submits job to batch queuing system

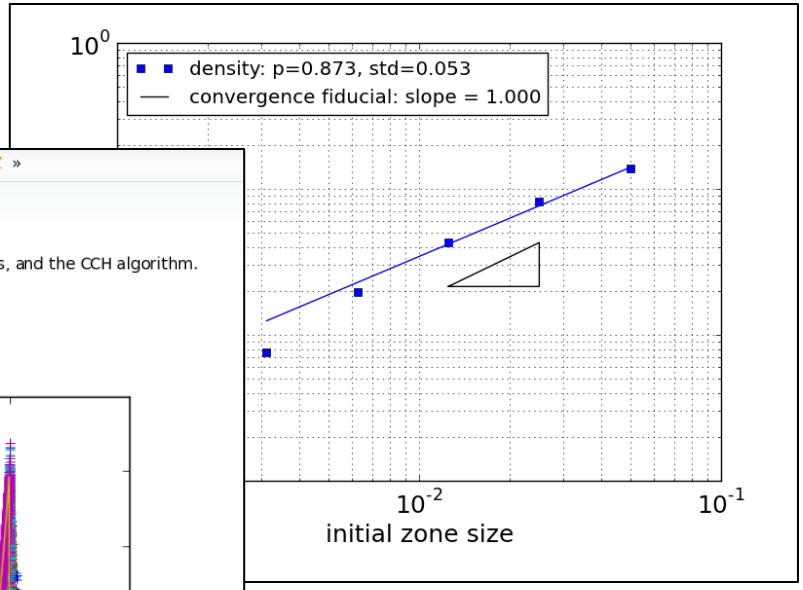
this method creates a plot

this method returns some text

Example of automated documentation (HTML format)



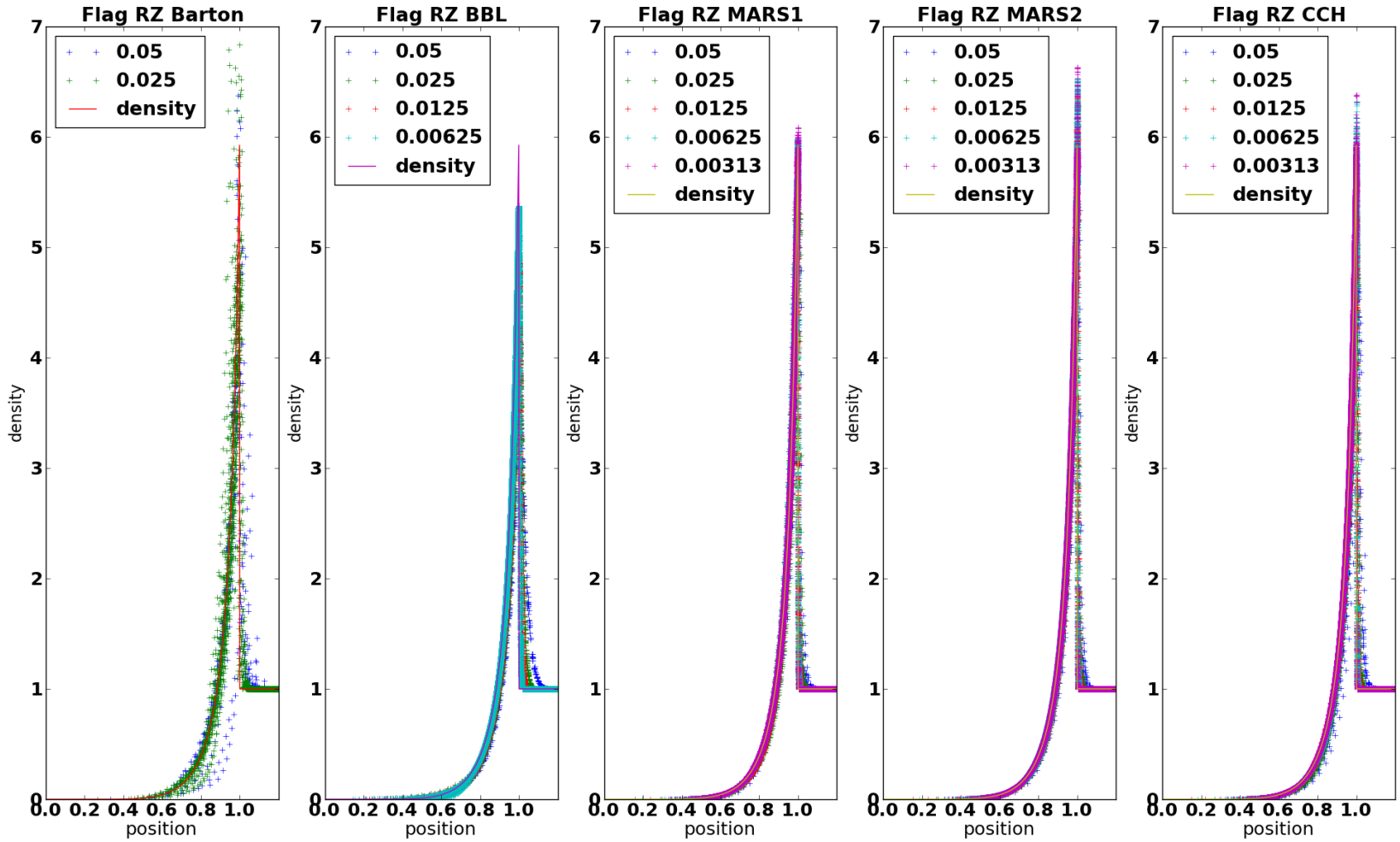




Test

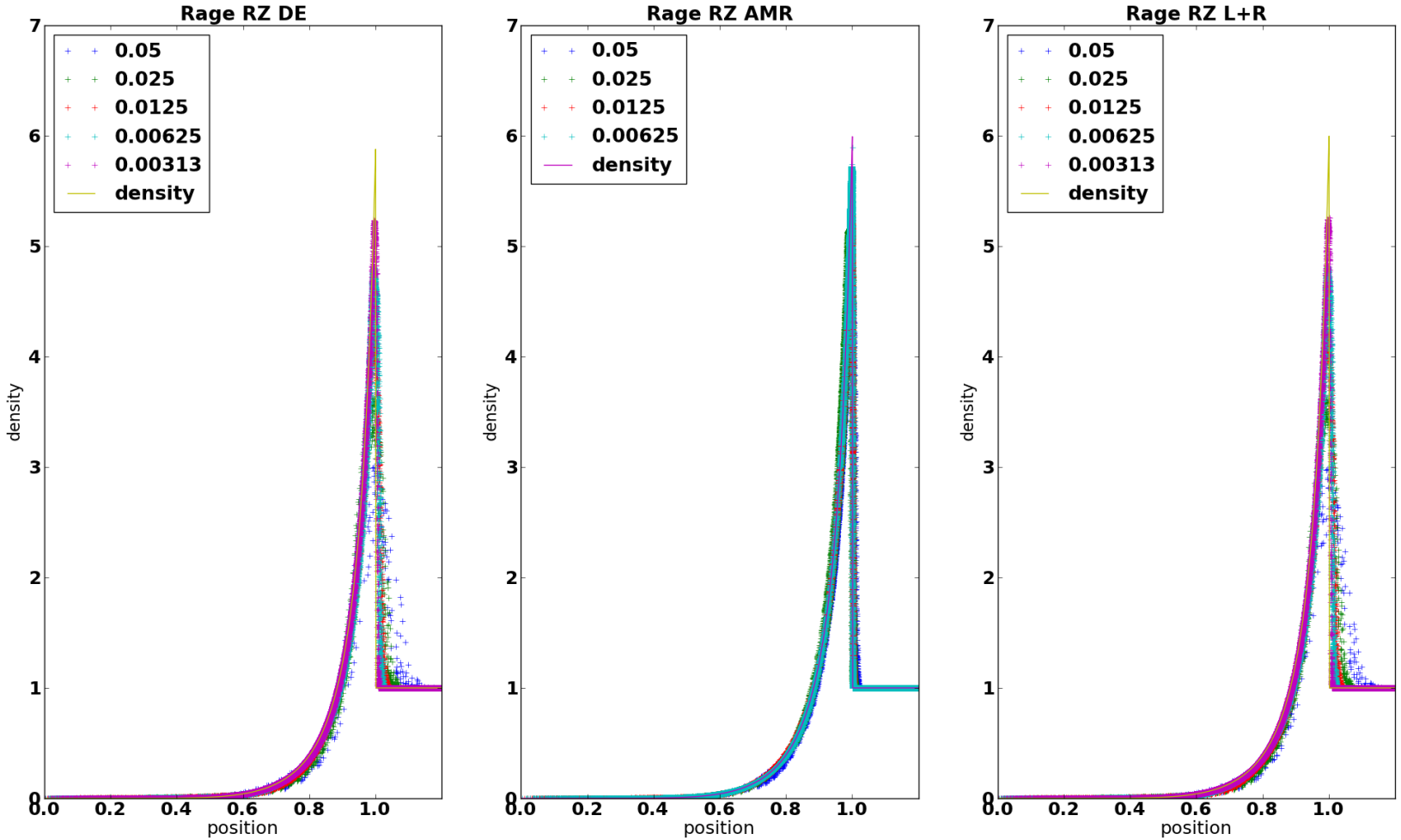
Use case for VTS: Comparing multiple hydro capabilities

Sedov problem in FLAG, R-Z geometry, various hydro options



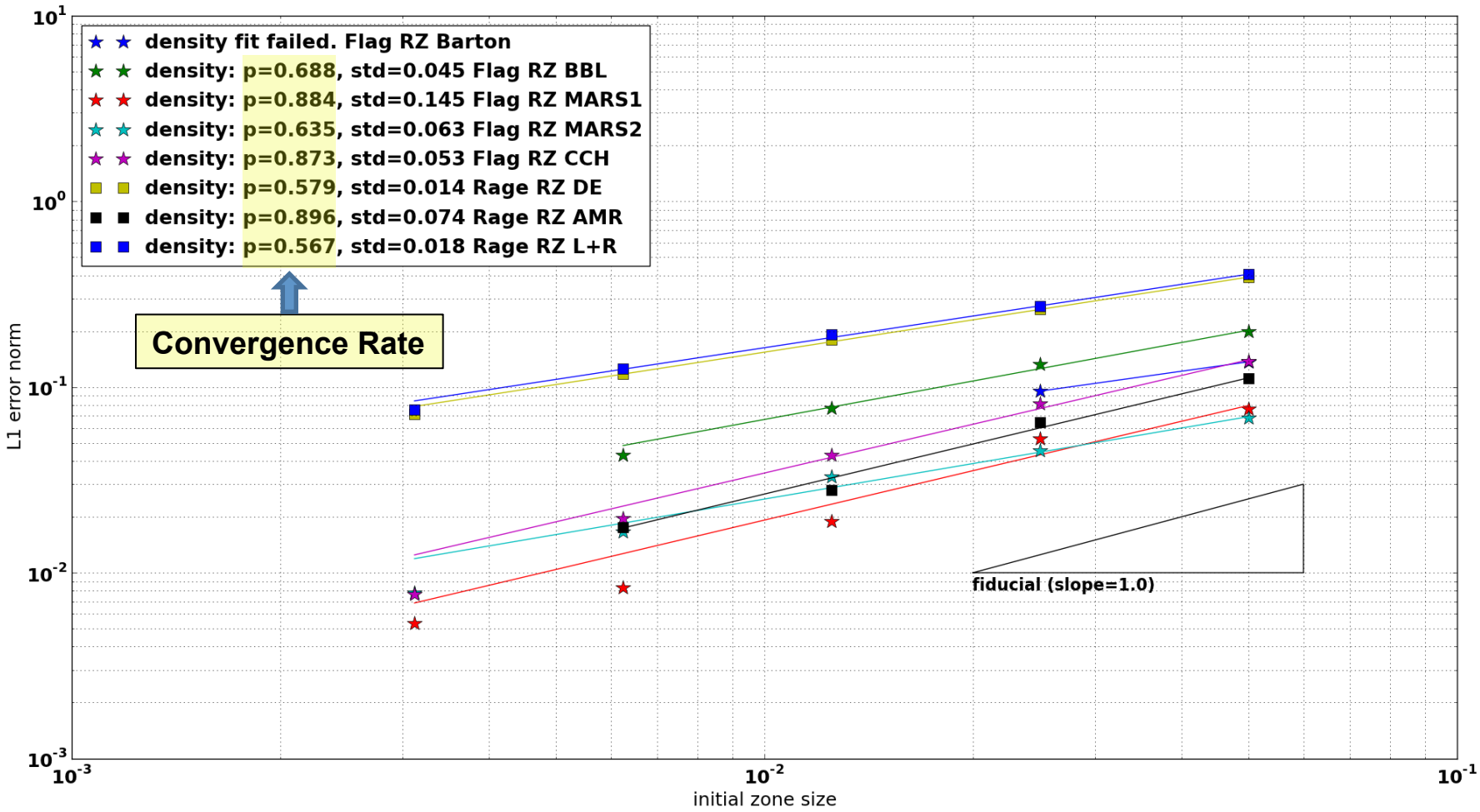
Use case for VTS: Comparing multiple hydro capabilities

Sedov problem in xRage, R-Z geometry, various hydro options



Use case for VTS: Comparing multiple hydro capabilities

Comparison of convergence behavior for various hydro options over 2 codes



Looking forward

- ExactPack release to GitHub
- More physics areas:
  - Reactive flow
  - Radiation-hydrodynamics
  - Solid mechanics
  - Smooth compressible flow
- Transition from development to usage
- Manufactured solution capabilities