LA-UR-15-22751 v2

ExactPack: A Master Code of Exact Solutions for Code Verification

Robert L. Singleton Jr.
Los Alamos National Laboratory
bobs1@lanl.gov

Collaborators:
Scott Doebling
Daniel Israel
James Kamm

ASME V&V Symposium 2015
Las Vegas, NV
13-15 May 2015

I. ExactPack: Exact Solutions and Verification

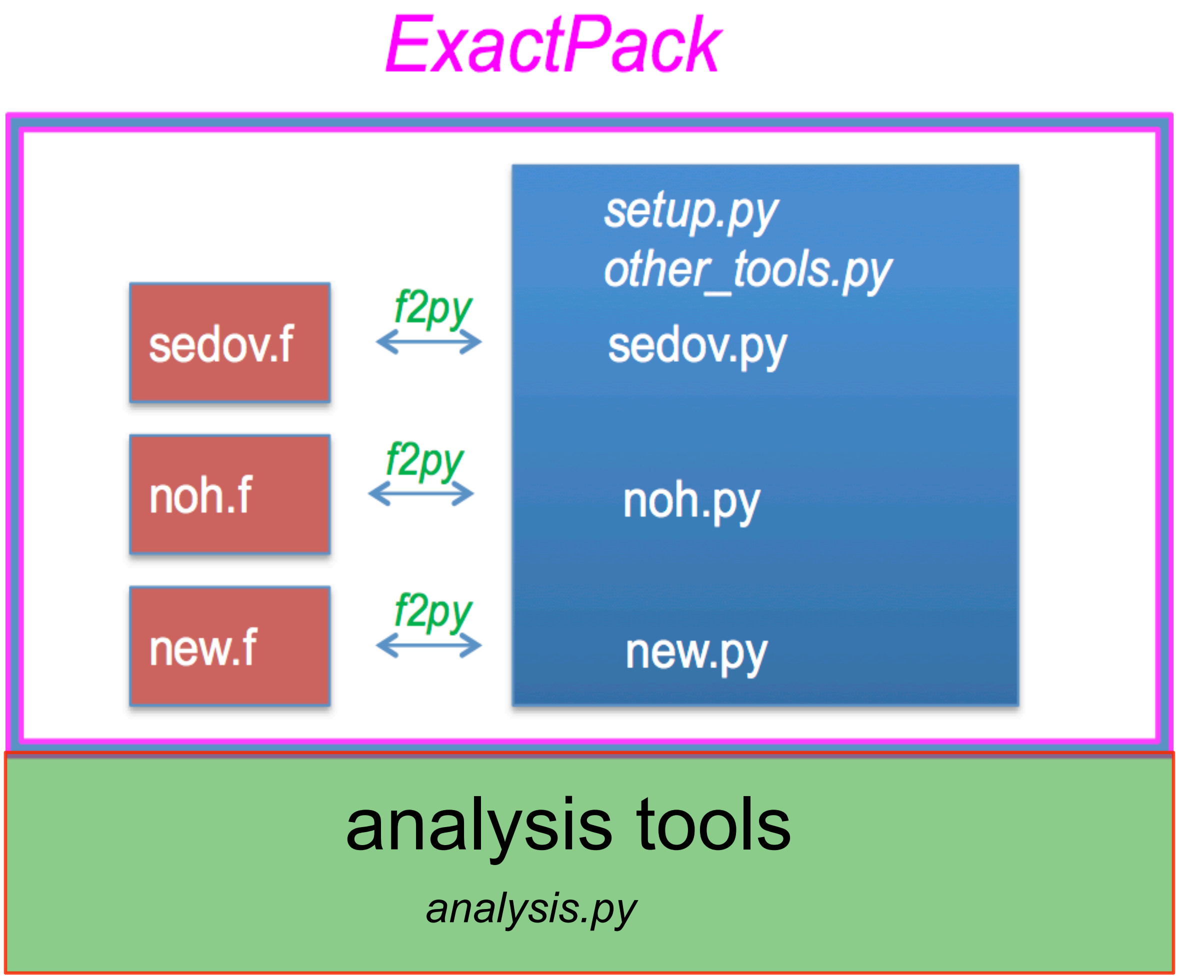
- ***ExactPack*** (LA-CC-14-047) is a Python package of exact solutions to physics problems and their corresponding verification analysis packages.
- Use as stand-alone code or a Python package in other codes
- Python driver scripts access a library of exact solutions written in Fortran, C, or python (and can be easily expanded)
- Self-documentation and unit tests
- **ExactPack** also contains a library of python analysis scripts for code verification (convergence analysis)
- Collaborative model, *i.e.* add your own solver on GitHub
  www.github.com/losalamos/ExactPack

We first select a verification problem: in this talk we choose the *Sod* shock tube for simplicity.

- Find an exact solution code to the problem, *e.g.* sod.f
  - internet
  - write one yourself
  - ask a friend (Jim Kamm)
- Create user interface
- Run code and plot data
- Compare against a hydro code that is to be *verified*.

This can be a long and error prone process itself! ExactPack is designed to provide these capabilities.

ExactPack integrates verification analysis and an exact solution library into a single stand-alone Python package



- **Exact solutions library**
- **Python API** integrates the solution library into a common framework:
  (a) import into scripts
  (b) GUI
  (c) command line control
- **Analysis tools**
  (a) plot solutions with run data
  (b) convergence study

NNSA

Exact solutions currently implemented in ExactPack

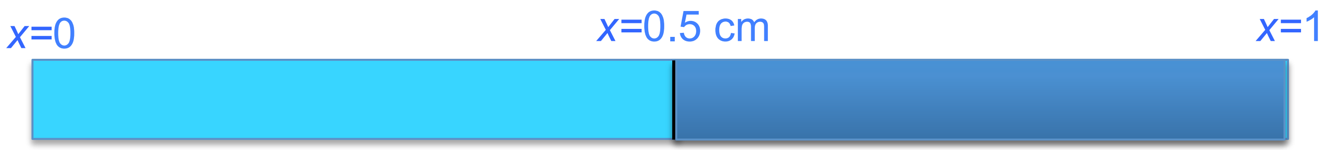
- **Riemann Shock Tubes (6 problem variants)**
  Sod, Einfeldt, Stationary-Contact, Slow-Shock, Shock-Contact-Shock, LeBlanc
- **Noh**
- **Sedov**
- **Riemann with the JWL EOS (2 variants)**
- **Guderley**
- **Coggeshall problems (20 of the 22 problems)**
- **RMTV**
- **Su-Olson**
- **Mader**
- **Escape of High Explosives Problem**
- **Steady Domain Reaction Zone Problem**

pure hydro

hydro + rad/conduction

high explosives

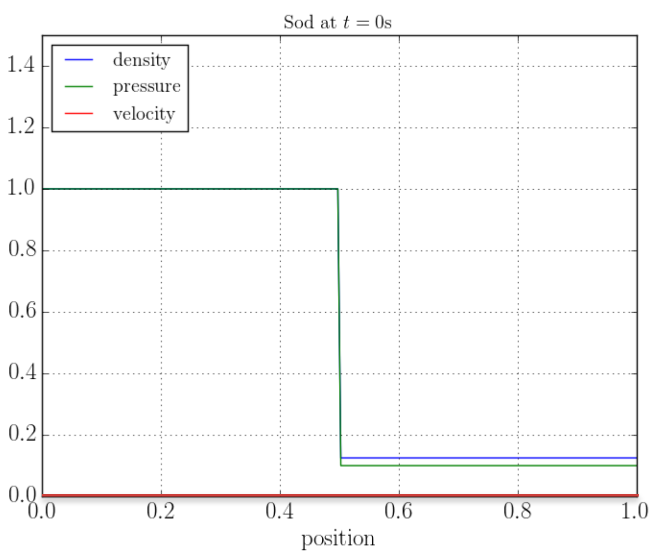
II. The Sod Shock Tube in ExactPack



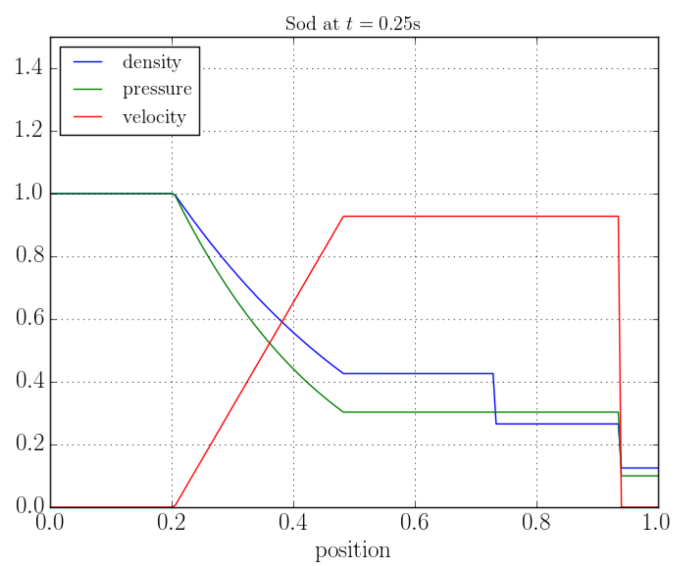
γ = 1.4
ρ= 1 g/cm^3
u = 0 cm/s
p = 1 dyne/cm^2

γ = 1.4
ρ= 0.125 g/cm^3
u = 0 cm/s
p = 0.1 dyne/cm^2

t=0 s



t=0.25 s



Adding a Fortran Solver to ExactPack

ExactPack/src/kamm/riemann/shktub.f90

```
Subroutine riemann_kamm(time, npts, x, xd0, gammal,
gammr, ... , rho, p, u, sound, sie)

!f2py  intent(out)   :: rho, p, u, sound, sie
!f2py  intent(hide) :: npts
!f2py  integer :: npts
!f2py  real      :: rho(nstep), p(nstep), u(nstep),
!f2py  real      :: sound(nstep), sie(nstep)
!f2py  real      :: time, x(npts)
!f2py  real      :: gammal, gammar, rhol, pl, ul,  …

Integer npts
real*8 xpos(npts), rho(npts), p(npts), ...
real*8 time, xd0, gammal, gammar, ...
...
Do it=1, npts
   xi=x(it)
   Call SHKTUB(xi,rhol,rhor,pl,pr,ul,ur,...,
    rhoi, pi, ui, souni, siei)
   rho(it)  = rhoi
   p(it)    = pi
   u(it)    = ui
   sound(it)= souni
   sie(it)  = siei
EndDo
End Subroutine riemann_kamm
```

original source code

step 1

Add a *wrapper* **riemann_kamm** to the original Fortran source code, and insert *f2py* directives.

Call the original subroutine

Return the python output quantities: rho, p, u, sound, sie

Adding a Fortran Solver to ExactPack

step 1: wrapper for original source

ExactPack/src/kamm/riemann/shktub.f90

```
Subroutine riemann_kamm(time, npts, x, xd0, gammal,
gammr, … , rho, p, u, sound, sie)

!f2py  intent(out)   :: rho, p, u, sound, sie
!f2py  intent(hide) :: npts
!f2py  integer :: npts
!f2py  real        :: rho(nstep), p(nstep), u(nstep),
!f2py  real        :: sound(nstep), sie(nstep)
!f2py  real        :: time, x(npts)
!f2py  real        :: gammal, gammar, rhol, pl, ul,  …

Integer npts
real*8 xpos(npts), rho(npts), p(npts), ...
real*8 time, xd0, gammal, gammar, ...
...
Do it=1, npts
   xi=x(it)
   Call SHKTUB(xi,rhol,rhor,pl,pr,ul,ur,...,
    rhoi, pi, ui, souni, siei)
   rho(it)  = rhoi
   p(it)    = pi
   u(it)    = ui
   sound(it)= souni
   sie(it)  = siei
EndDo
End Subroutine riemann_kamm
```

original source code

step 2: python-f90 interface

ExactPack/exactpack/riemann/kamm.py

```
class Riemann(ExactSolver)

def _run(self, r, t)
      rho, p, u, sound, sie = riemann_kamm(time=t, x=r,
      xd0=self.interface_loc, gammal=self.gammal, … )
```

step 3: short alias

```
class Sod(Riemann)

gammal = 1.4
gammar = 1.4
interface_loc = 0.5
rhol = 1.0
pl    = 1.0
ul    = 0.0
rhor = 0.125
pr     = 0.1
ur     = 0
```

Python script to plot Sod from ExactPack

from exactpack.solvers.riemann import Sod ⬅ import Sod object

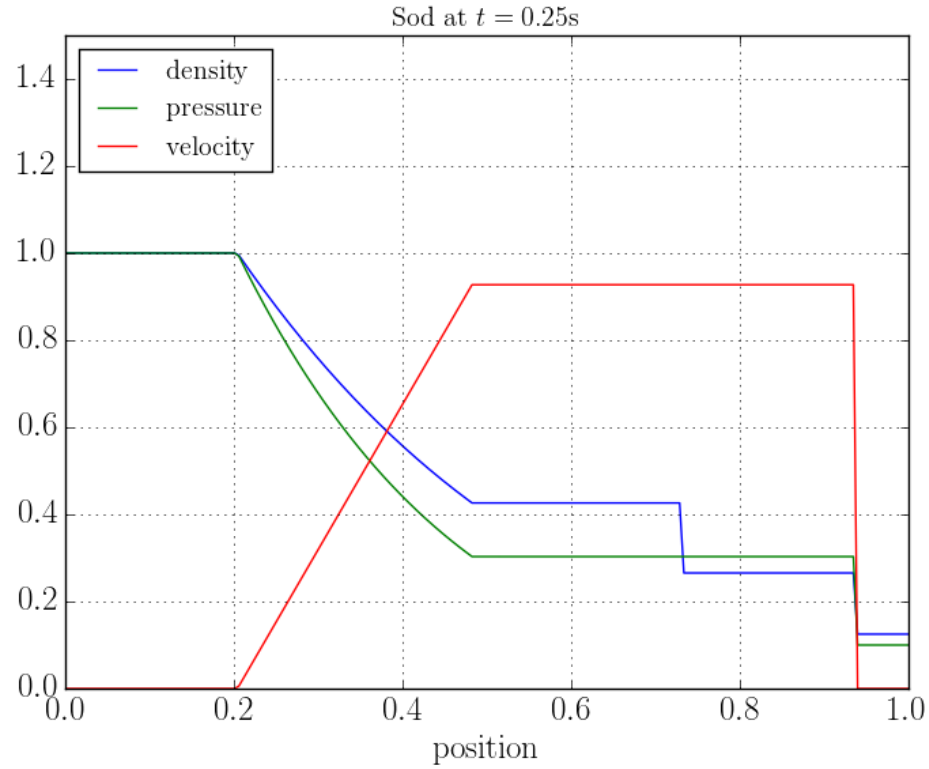
r = numpy.linspace(0.0, 1.0, 1000) ⬅ spatial array and time
t = 0.25

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

soln.plot('density')
soln.plot('pressure')
soln.plot('velocity')

solution object can plot itself, *i.e.* it has a **plot method**

…
Other plot directives
…
plt.savefig('sod.png')
soln.dump('sod.dat')
plt.show()



Example script using the CodeVerificationStudy object

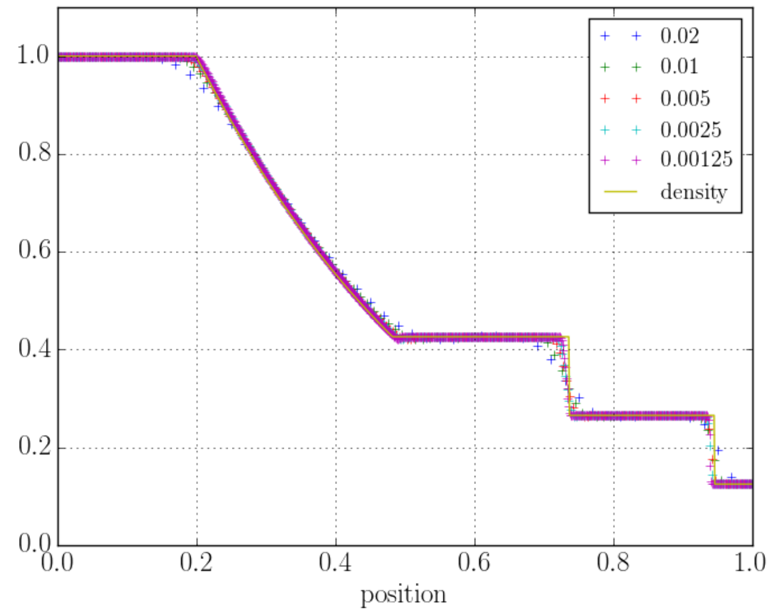
```
from exactpack.sedov.riemann import Sod
  from exactpack.analysis import CodeVerificationStudy


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

Plot study

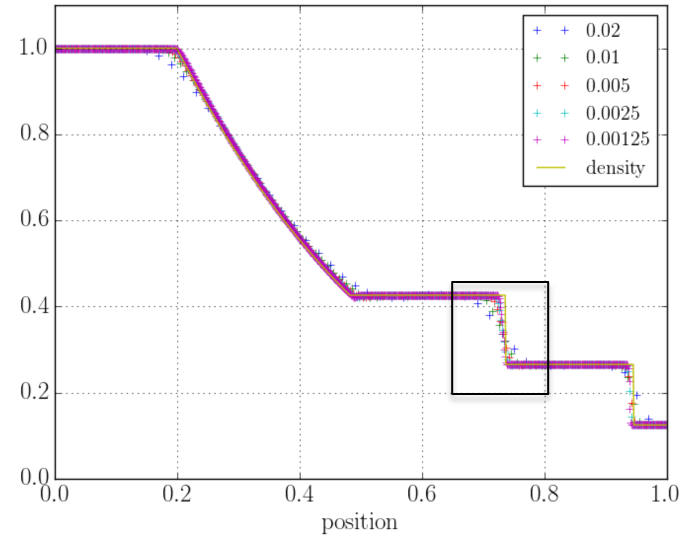
study.plot('density')

study object has a plot *method*



Use native Python within ExactPack

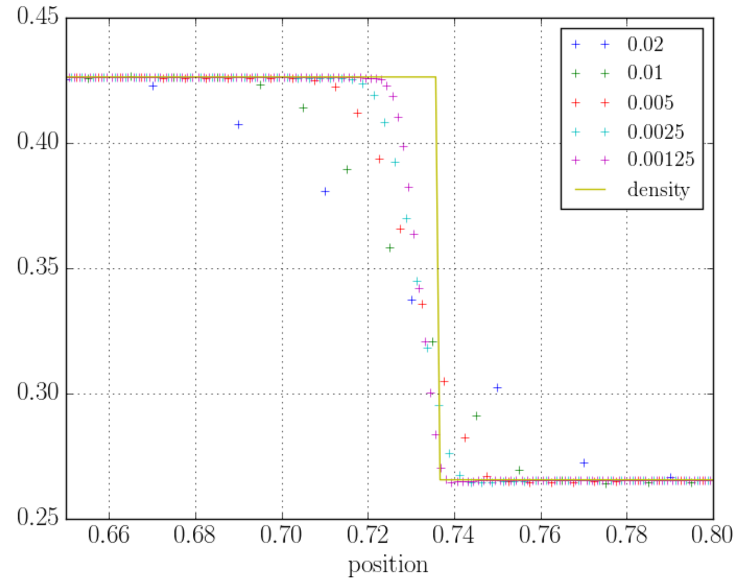
study.**plot**('density')



Zoom in on shock

import matplotlib.pyplot as plt

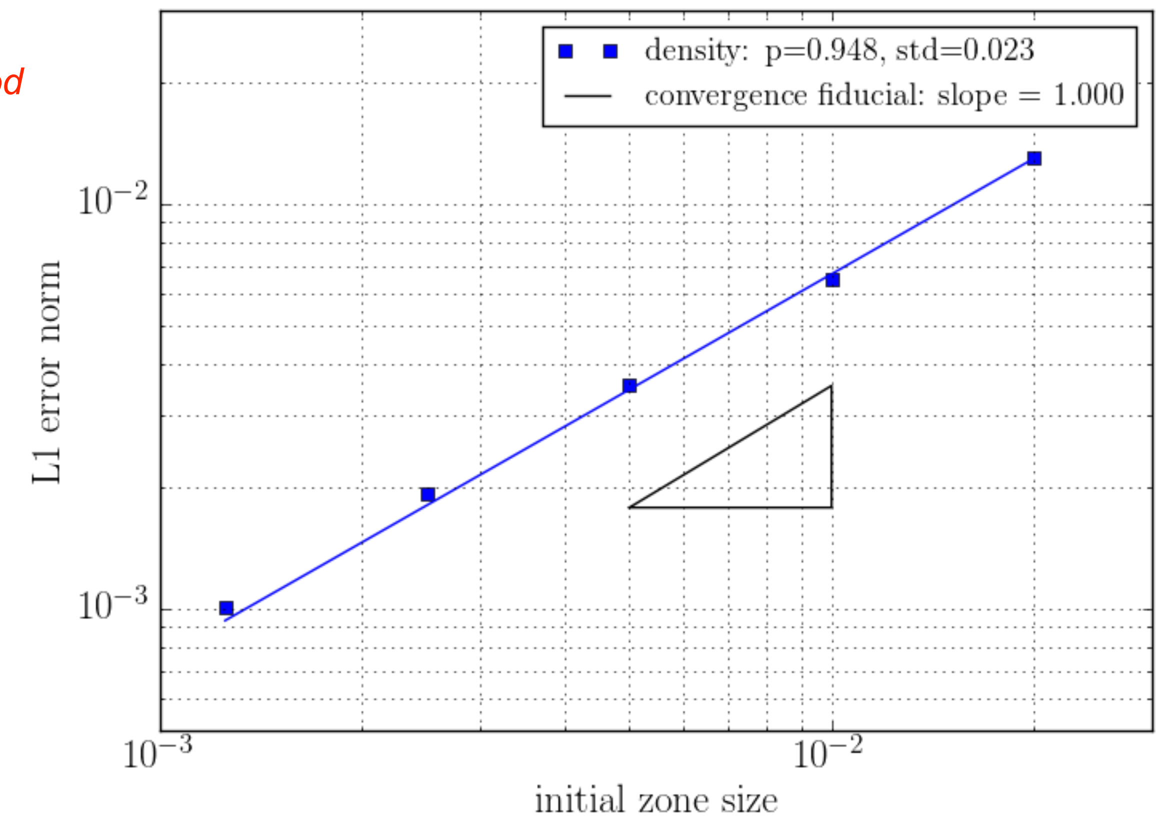
plt.xlim(0.65, 0.8)
plt.ylim(0.25, 0.45)



The *convergence* method for a study object

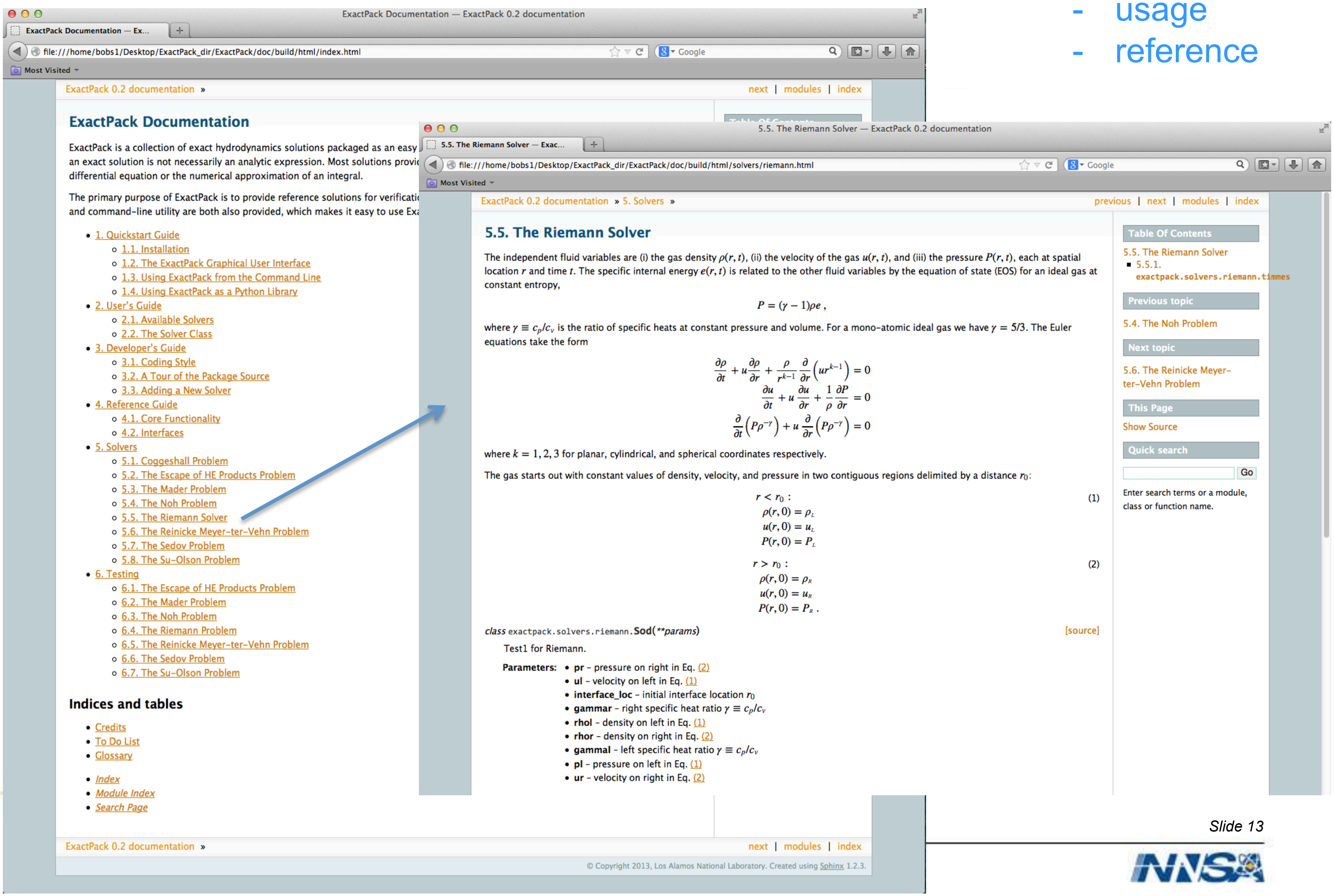
```
study.convergence('density').plot(fiducial=1.0)
```

the study object can use a convergence analysis *method*



III. ExactPack is fully documented

- theory
- usage
- reference


ExactPack Documentation — ExactPack 0.2 documentation

file:///home/bobs1/Desktop/ExactPack_dir/ExactPack/doc/build/html/index.html

ExactPack 0.2 documentation » | next | modules | index

ExactPack Documentation

ExactPack is a collection of exact hydrodynamics solutions packaged as an easy … an exact solution is not necessarily an analytic expression. Most solutions provi… differential equation or the numerical approximation of an integral.

The primary purpose of ExactPack is to provide reference solutions for verificati… and command-line utility are both also provided, which makes it easy to use Ex…

- 1. Quickstart Guide
  - 1.1. Installation
  - 1.2. The ExactPack Graphical User Interface
  - 1.3. Using ExactPack from the Command Line
  - 1.4. Using ExactPack as a Python Library
- 2. User's Guide
  - 2.1. Available Solvers
  - 2.2. The Solver Class
- 3. Developer's Guide
  - 3.1. Coding Style
  - 3.2. A Tour of the Package Source
  - 3.3. Adding a New Solver
- 4. Reference Guide
  - 4.1. Core Functionality
  - 4.2. Interfaces
- 5. Solvers
  - 5.1. Coggeshall Problem
  - 5.2. The Escape of HE Products Problem
  - 5.3. The Mader Problem
  - 5.4. The Noh Problem
  - 5.5. The Riemann Solver
  - 5.6. The Reinicke Meyer-ter-Vehn Problem
  - 5.7. The Sedov Problem
  - 5.8. The Su-Olson Problem
- 6. Testing
  - 6.1. The Escape of HE Products Problem
  - 6.2. The Mader Problem
  - 6.3. The Noh Problem
  - 6.4. The Riemann Problem
  - 6.5. The Reinicke Meyer-ter-Vehn Problem
  - 6.6. The Sedov Problem
  - 6.7. The Su-Olson Problem

Indices and tables

- Credits
- To Do List
- Glossary

- *Index*
- *Module Index*
- *Search Page*

ExactPack 0.2 documentation » | next | modules | index

© Copyright 2013, Los Alamos National Laboratory. Created using Sphinx 1.2.3.

5.5. The Riemann Solver — ExactPack 0.2 documentation

file:///home/bobs1/Desktop/ExactPack_dir/ExactPack/doc/build/html/solvers/riemann.html

ExactPack 0.2 documentation » 5. Solvers » | previous | next | modules | index

5.5. The Riemann Solver

The independent fluid variables are (i) the gas density $\rho(r,t)$, (ii) the velocity of the gas $u(r,t)$, and (iii) the pressure $P(r,t)$, each at spatial location r and time t. The specific internal energy $e(r,t)$ is related to the other fluid variables by the equation of state (EOS) for an ideal gas at constant entropy,

$$P = (\gamma - 1)\rho e\,,$$

where $\gamma \equiv c_p/c_v$ is the ratio of specific heats at constant pressure and volume. For a mono-atomic ideal gas we have $\gamma = 5/3$. The Euler equations take the form

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial r} + \frac{\rho}{r^{k-1}}\frac{\partial}{\partial r}\left(u r^{k-1}\right) = 0$$
$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial r} + \frac{1}{\rho}\frac{\partial P}{\partial r} = 0$$
$$\frac{\partial}{\partial t}\left(P\rho^{-\gamma}\right) + u\frac{\partial}{\partial r}\left(P\rho^{-\gamma}\right) = 0$$

where $k = 1, 2, 3$ for planar, cylindrical, and spherical coordinates respectively.

The gas starts out with constant values of density, velocity, and pressure in two contiguous regions delimited by a distance r_0:

$$r < r_0:\quad \rho(r,0) = \rho_L,\quad u(r,0) = u_L,\quad P(r,0) = P_L \tag{1}$$

$$r > r_0:\quad \rho(r,0) = \rho_R,\quad u(r,0) = u_R,\quad P(r,0) = P_R\,. \tag{2}$$

class exactpack.solvers.riemann.**Sod**(***params*) [source]

Test1 for Riemann.

Parameters:
- **pr** – pressure on right in Eq. (2)
- **ul** – velocity on left in Eq. (1)
- **interface_loc** – initial interface location r_0
- **gammar** – right specific heat ratio $\gamma \equiv c_p/c_v$
- **rhol** – density on left in Eq. (1)
- **rhor** – density on right in Eq. (2)
- **gammal** – left specific heat ratio $\gamma \equiv c_p/c_v$
- **pl** – pressure on left in Eq. (1)
- **ur** – velocity on right in Eq. (2)

Table Of Contents
- 5.5. The Riemann Solver
  - 5.5.1. exactpack.solvers.riemann.timmes

Previous topic: 5.4. The Noh Problem

Next topic: 5.6. The Reinicke Meyer-ter-Vehn Problem

This Page: Show Source

Quick search [Go]

Enter search terms or a module, class or function name.

IV. Conclusions

- ExactPack is a new utility for code verification
  - Library of exact solutions
  - Collection of analysis utilities
- Stand alone or integrate into other scripts
- Expandable: Python, Fortran (77/90/95), C
- Auto-documenting (Python doc-strings and rst)
- Unit testing (and more unit testing)
- Collaborative model: will be on www.github.com/losalamos/ExactPack

Abstract

For code verification one compares code output against known exact solutions. There are many exact solutions used in this capacity, such as the Noh and Sedov problems. Such exact solution codes are usually stand-alone programs that can be downloaded from the web, after which the user must roll out his or her own plotting package and analysis utilities. When comparing across multiple codes, this can be a time consuming and error prone process. ExactPack is a new utility that integrates many of these exact solution codes into a common API (application program interface), and can be used as a stand-alone code or a python package. ExactPack consists of python driver scripts that access a library of exact solutions written in Fortran or Python. We have documented the physics of each problem in the solution library, and provided complete documentation on how to extend the library to include additional exact solutions. ExactPack's code architecture makes it easy to extend the solution-code library to include additional exact solutions in a robust, reliable, and maintainable fashion. This talk will emphasize the ease with which a new solver package can be added to ExactPack, and will include a live demo of the plotting and analysis capabilities.