LA-UR-14-23068
Approved for public release; distribution is unlimited.

Title: Anatomy of a Multi-Code Verification Test System

Author(s): Doebling, Scott W.

Intended for: ASME 2014 Verification & Validation Symposium, 2014-05-07/2014-05-09 (Las Vegas, Nevada, United States)

Issued: 2014-05-21 (rev.1)

Anatomy of a Multi-Code Verification Test System

Scott W. Doebling, Ph.D.

Group Leader, Verification & Analysis Group (XCP-8)
Computational Physics Division

Collaborators:
Dan Israel, Jim Kamm,
Bob Singleton

May 2014

A multi-code verification test system

- Perform code verification in a consistent manner
- Across multiple computational physics codes
- Overview of components and brief discussion
- Future talks (one today!) will describe the components in depth

Why a multi-code verification test system?

- Enable standardized code verification testing
- Objective evaluation of accuracy & convergence
- Objective basis for comparison across codes
- Ensure consistency and repeatability
- Multi-code verification is not “code-to-code comparison”

Anatomy of a multi-code verification test system

- Definitions for verification test problems
- Test problem exact solutions
- Input deck generation & simulation execution
- Code verification analysis
- Documentation & archiving

Definitions for verification test problems

- Specification of problem parameters
- Specification of computational setup
  - Code independent, e.g. Lagrangian vs Eulerian, to extent possible
- Specification of code outputs

Definitions for verification test problems

- Proposed standard definitions for Sedov, Noh, and Riemann published
  - *"Standardized Definitions for Code Verification Test Problems", LA-UR-14-20418.*
  - *Attached to the end of this document!*
- Revised versions will be published as we add more problems.

Example of standardized definition for verification test 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.

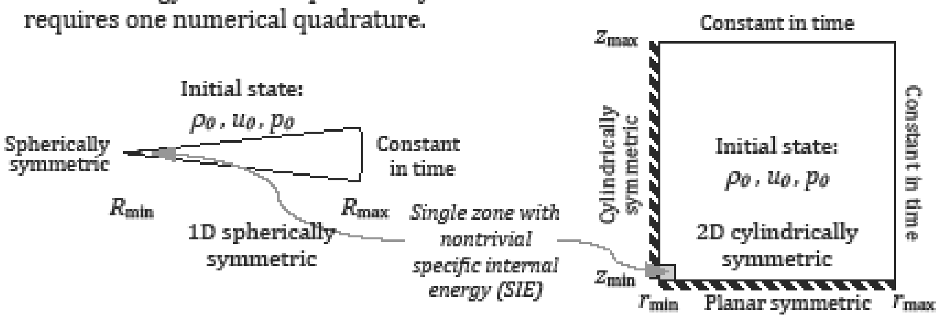


Figure 1: Initial configuration of the Sedov problem.

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

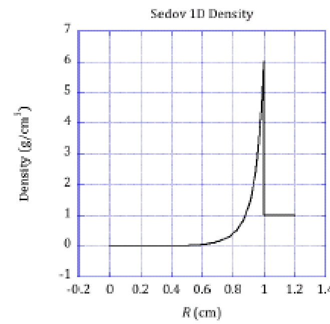


Figure 1: Sedov Problem density.

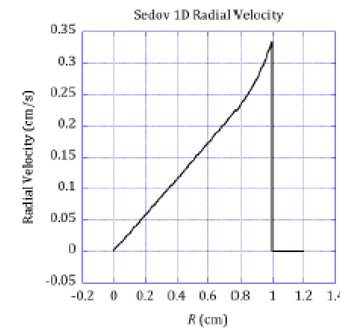


Figure 2: Sedov Problem radial velocity.

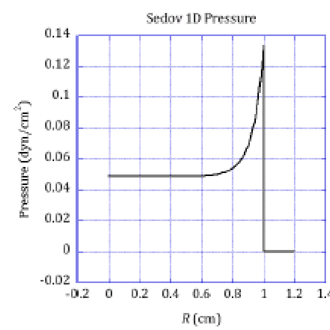


Figure 3. Sedov Problem pressure.

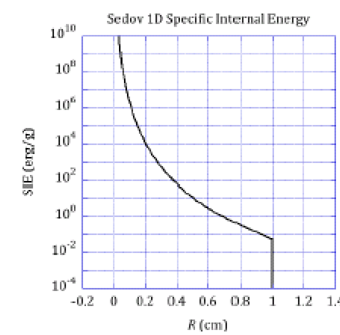


Figure 4. Sedov Problem SIE.

Test problem exact solutions

- Code verification requires highly accurate – preferably “exact” - problem solutions
- For shock hydrodynamics, many individual exact solution codes have been written, but:
  - They exist in disparate locations
  - No standardized input or output format
  - Users must add own plotting package, etc.
- Need a single, standardized toolbox for these codes

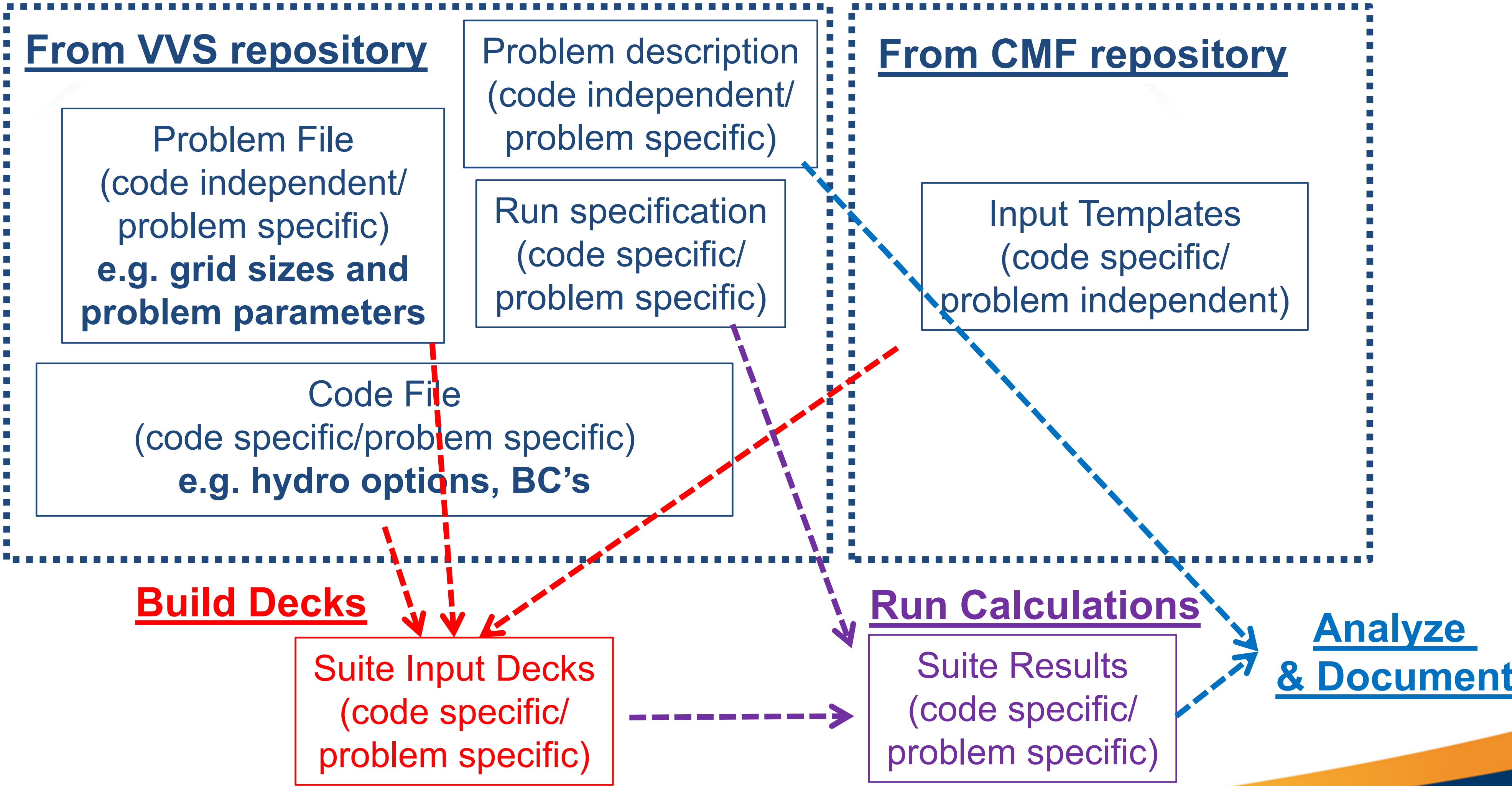
ExactPack: An Exact Solution Analysis Package

- New software package for exact solution codes
- Enables plotting of exact solutions
- Easily extended to new problems
- Use as stand-alone code or as a python package
- Code is robust, reliable, and maintainable
- Currently contains several compressible hydro problems: Noh, Sedov, Riemann, Guderley, …
- Will be available soon for public usage. Presentation by Bob Singleton immediately following!

Input deck generation & simulation management

- Need a reliable, repeatable, robust way to generate code input decks
- Consistency across codes with arbitrarily different input formats
- Use LANL “common model framework” (CMF) templates
- Problem-specific information stored in LANL V&V System (VVS) repository
- Build documentation from this repository

Using CMF templates and VVS problem definitions for a code verification study



Verification Pedigree and Reproducibility

- Pedigree of verification test suite results
  - Which problem
  - With what code settings
  - Done by who and when
- Ability to reproduce code test results:
  - Same problem in same code with specified version
  - Same problem in different code

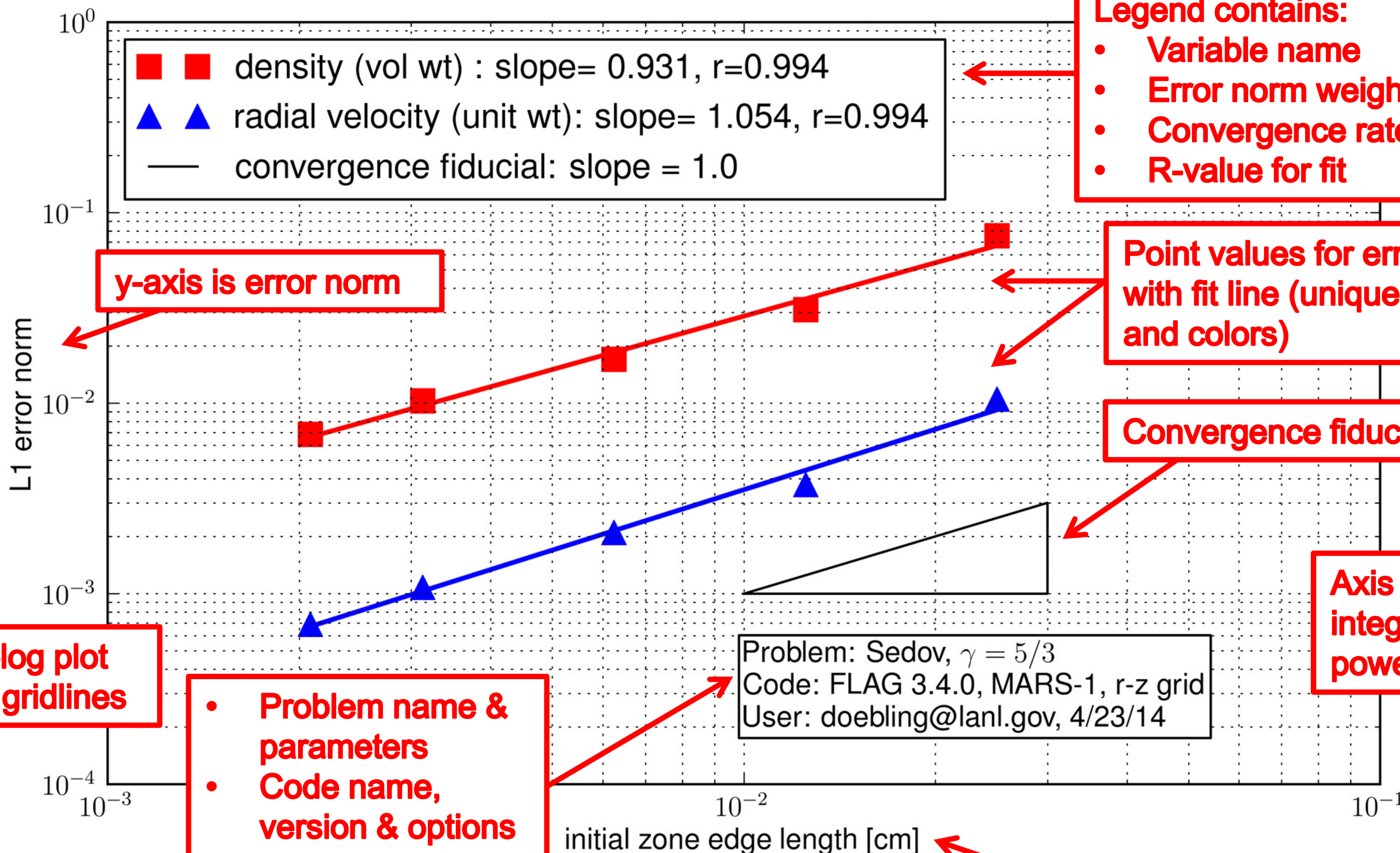
We are developing a standardized method for code verification analysis within the ASC program at LANL

Choices to be made (et al.)

workflow

- Perform calculations on series of grids
  - Standardize:
    - Problem definitions
    - Input decks
    - Grids
- Generate exact solution
  - Standardize: *ExactPack*
- Compare physical variables
- Evaluate global error metrics
  - Standardize:
    - Metrics
    - Weighting
    - Mapping between grids
- Perform convergence analysis
  - Standardize:
    - Error model
    - Method for calculating convergence rate
- Present verification results
  - Standardize:
    - Plot format
    - Report format

Working on a standardized format for presenting convergence results



Legend contains:
- Variable name
- Error norm weight
- Convergence rate
- R-value for fit

y-axis is error norm

Point values for error norm, with fit line (unique markers and colors)

Convergence fiducial

Axis limits are integer powers of 10

Log-log plot with gridlines

- Problem name & parameters
- Code name, version & options
- User name & date

x-axis is length scale (not number of zones)

Automated documentation

- Sphinx tools with RST markup language
- Store problem definition and skeleton of analysis report in repository
- Generate analysis results using standardized script
- Human-readable report in HTML, PDF, LaTex, etc.
- Generate “stoplight” metrics as required
- Distribute results to stakeholders
- Archive results at some interval (e.g. time-based or release-based)

A multi-code verification test system

- Perform code verification in a consistent manner
- Across multiple computational physics codes
- Overview of components and brief discussion
- Future talks (one today!) will describe the components in depth

Thank you, you've been a great audience!