

Verification of PDE Discretization Library and Flow Application Codes

Roy H. Stogner

The University of Texas at Austin

September 30, 2009

3D Hypersonic Flow Approximations

Math, Software Components

- Discretized Formulation
- Spatial Discretization
- Time Discretization
- System Assembly
- Nonlinear Solver
- Linear Solver
- I/O
- Postprocessing

Approximate LoC

| | |
|-----------|---------|
| DPLR | 46,000 |
| Coupling | 6,000 |
| Ablation | 4,000 |
| Radiation | 1,000 |
| FIN-S | 6,000 |
| libMesh | 54,000 |
| PETSc | 170,000 |
| Other | 56,000 |

3D Hypersonic Flow Approximations

Components To Verify

- Discretized Formulation
- Spatial Discretization
- Time Discretization
- System Assembly
- Nonlinear Solver
- Linear Solver
- I/O
- Postprocessing

Potentially Buggy LoC

| | |
|-----------|---------|
| DPLR | 46,000 |
| Coupling | 6,000 |
| Ablation | 4,000 |
| Radiation | 1,000 |
| FIN-S | 6,000 |
| libMesh | 54,000 |
| PETSc | 170,000 |
| Other | 56,000 |

Regression Tests

Revise Software, Rerun Tests

- Example applications, unit tests, benchmark tests
- Catches changes' *unintended consequences*
- Continuous Build System automation
 - ▶ Tests not just run “by hand” by developers
 - ▶ libMesh, application tests in Trac/Bitten at INL
 - ▶ libMesh, FIN-S basic tests now in BuildBot at UT
 - ▶ Ablation, radiation, QUESO tests in BuildBot

Regression Tests

Revise Software, Rerun Tests

- Example applications, unit tests, benchmark tests
- Catches changes' *unintended consequences*
- Continuous Build System automation
 - ▶ Tests not just run “by hand” by developers
 - ▶ libMesh, application tests in Trac/Bitten at INL
 - ▶ libMesh, FIN-S basic tests now in BuildBot at UT
 - ▶ Ablation, radiation, QUESO tests in BuildBot

Unit Tests

- Written to test one object at a time
 - ▶ Reusable modules interact with all other code through a limited API
 - ▶ That API can be tested directly outside of application code
 - ▶ Test one method at a time, isolate problems locally
 - ▶ 108 Unit Tests in libMesh

High-level Assertions

Manual `assert()`, PETSc debug mode

- Active only in “debug” runs
- Verifies function preconditions, postconditions
- Approx. 4000 assertions in `libMesh`, FIN-S
- Can provide per-processor stack traces, etc.

Low-level Assertions

- Defining `_GLIBCXX_DEBUG`
 - ▶ Runtime bounds-checking of standard `vector`, `iterator` use
- `ifort` “-check bounds”
- Valgrind memory testing
- Out Of Bounds errors can lead to corrupt data, not just segfaults!

Assertions Have Uncovered **MANY** Library Bugs

- Uninitialized data
- Unpartitioned elements
- “Tearing” in neighbor map reconstruction
- Parallel vector operation miscommunication
- Parallel mesh adaptivity synchronization failures
- Out-of-order API calls
- $N_{elem} < N_{proc}$ bugs
- Bad I/O node numbering
- Unsupported I/O format features
- Failure to “deep copy” a Mesh

Assertions Have Uncovered **MANY** Library Bugs

- Uninitialized data
- Unpartitioned elements
- “Tearing” in neighbor map reconstruction
- Parallel vector operation miscommunication
- Parallel mesh adaptivity synchronization failures
- Out-of-order API calls
- $N_{elem} < N_{proc}$ bugs
- Bad I/O node numbering
- Unsupported I/O format features
- Failure to “deep copy” a Mesh

And **COUNTLESS** Application Bugs

- API mistakes
 - ▶ Array misallocation
 - ▶ Misordered function args
 - ▶ Sharing non-shareable objects
 - ▶ Querying data before calculating it
 - ▶ Off-by-one, transposed indexing errors
 - ▶ Finite elements on incompatible geometric elements
- Runtime problems
 - ▶ Invalid input files
 - ▶ Unsupported p-refinement levels
 - ▶ Attempting incompatible mesh modification

Parametric Testing

One Test Code, Many Tests

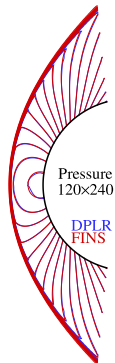
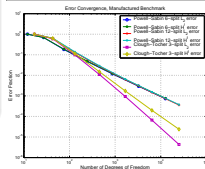
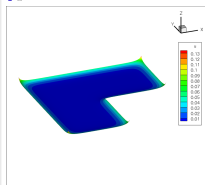
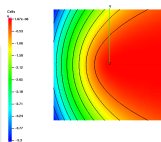
- Keep test codes generic
- Execute with many different parameter choices
- Compile time examples:
 - ▶ Algebraic solver interface
 - ▶ Real/complex arithmetic
 - ▶ Mesh data structure
- Run time examples:
 - ▶ Mesh geometry, element type
 - ▶ Discretization type, order
 - ▶ Partition type, processor count
 - ▶ Error indicator type
 - ▶ Adaptive refinement strategy
 - ▶ I/O format
 - ▶ Coupled physics model version

Verification Benchmark Problems

Choosing Test Problems

Capitalize on anything you know a priori:

- Known/manufactured solutions
 - ▶ Exact solution to discretized problem
 - ▶ Limit solution of continuous problem
 - ▶ Solution from alternative code, reduced model
 - ▶ Known quantities of interest, symmetries
- Known asymptotic convergence rates
- Known residuals



Jacobian Verification

$$\mathbf{J} \equiv \frac{\partial \mathbf{R}}{\partial \mathbf{u}}$$

Residual, Jacobian Construction

- Inherently physics-dependent
- But can be validated *against each other*

Finite Differencing

$$\mathbf{J}_{ij} \approx \frac{\mathbf{R}_i(\mathbf{u} + \varepsilon \mathbf{e}_j) - \mathbf{R}_i(\mathbf{u} - \varepsilon \mathbf{e}_j)}{2\varepsilon}$$

Greedy or element-wise algorithms handle sparsity

Complex-Step Perturbations

$$\mathbf{J}_{ij} \approx \frac{\text{Im}[\mathbf{R}_i(\mathbf{u} + \varepsilon \mathbf{e}_j \sqrt{-1})]}{\varepsilon}$$

Avoids floating point subtractive cancellation error

Automatic Differentiation

- Variable constructors seed derivatives
- Variable operations evaluate derivatives

Manufactured Solution Example

Convection-Diffusion Problem with Adjoints

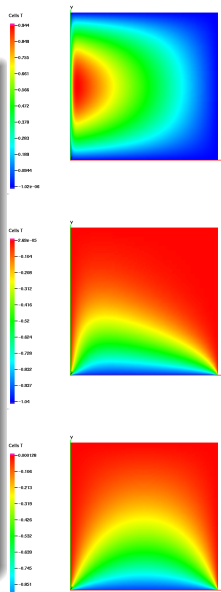
Residual equation:

$$\mathbf{R}(\mathbf{u}) = \nabla \cdot \alpha \nabla \mathbf{u} + \beta \vec{e}_x \cdot \nabla \mathbf{u} + \mathbf{f} = 0$$

Manufactured solution:

$$\mathbf{u} \equiv 4(1 - e^{-\alpha x} - (1 - e^{-\alpha})x)y(1 - y)$$

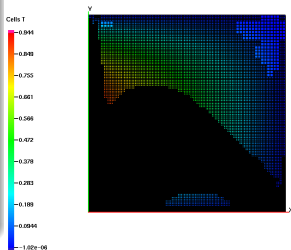
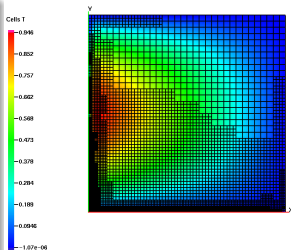
- Homogeneous Dirichlet boundary
- α controls flux strength, layer
- Choose any convection strength β , solve for \mathbf{f}
- $\beta = 0$ gives simple series adjoint solutions



Verification Examples

Goal-Oriented Refinement

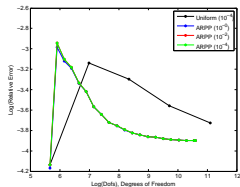
- Adjoint-based error indicator weighting
 - ▶ New `libMesh` feature
 - ▶ Test against manufactured solution problems first
- Convergence “plateaus” were found in multiple refinement strategies
- `UniformRefinementEstimator` required new code to solve for adjoint solution errors
- `PatchRecoveryErrorEstimator` required new seminorm integration (H^1 vs. $W^{1,\text{inf}}$) to give compatible error subestimates



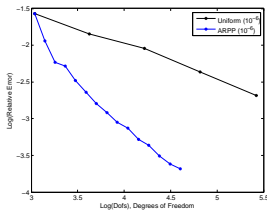
Verification Examples

Parameter Sensitivity

- Adjoint-based physics-independent discrete sensitivity calculations
 - ▶ New `libMesh` feature
 - ▶ Tested with manufactured solutions
- Convergence to analytic sensitivity plateaus at 2% relative error in **every** refinement strategy
- Finite differenced partial derivatives not responsible
- Manufactured solution allowed sensitivity subcomponent comparison to analytic solutions
- Sign errors in `libMesh` parameter sensitivity method



Verification Examples



Parameter Sensitivity Verification

- “Off by 100%” error remaining in one small solution term
- Switch to $u'' = f$, 1D quadratic solutions, manufactured residual test
- Identified bug in repeated `adjoint_solve rhs assembly`
- Returning to manufactured solution benchmark: now converges to true solution

Verification Continues

Thank you

