Mapping Research Requirements to Software Tools

Dr. Christopher S. Simmons
Director, Cyberinfrastructure Researcher Support
Office of Information Technology
Department of Computer Science
UT Dallas
simmons@utdallas.edu

Framing the problem

Computational Science Software ...

- ... development is not (well) funded
- ... complexity is constantly increasing
- ... applicability is ever increasing

> "For every complex problem there is an answer that is clear, simple, and wrong."
> H.L. Mencken (1949)

Developing your own software to implement every feature you need is perhaps "clear, simple and wrong".

Software Giants?

"Being able to quickly evaluate good (scientific) software is increasingly important." C. Simmons (Just Now!)

So ...

- "If I have seen further, it is by standing on the shoulders of giants." I. Newton (1676)
- How can we identify software giants?
  - By enumerating requirements for a healthy software ecosystem, I hope to help you eliminate some software immediately
  - By providing processes and tools to help increase confidence, I hope to help you identify "good" software

By the end of this talk … (Goals/Outline)

- By the end of this talk, you should have a strategy for identifying whether software is worth evaluating
- By the end of this talk, you should have a game plan for building confidence in that software
- By the end of this talk, you should have a strategy for identifying what needs to be done before writing your own software
- By the end of this talk, you should have a game plan for writing your own software
- And, by the end of this talk, you will have heard generally accepted recommendations on scientific software as well as my highly-opinionated ones.

Version Control

Minimum Guidelines – Actually using version control is the first step

Ideal Usage

- Put everything under version control
- Consider putting parts of your home directory under VC
- Use a consistent project structure and naming convention
- Commit often and in logical chunks
- Write meaningful commit messages
- Do all file operations in the VCS
- Set up change notifications if working with multiple people

Common Version Control Systems

- CVS
- Subversion (SVN)
- Git

Build Systems

Minimum Guidelines

- Any build system is a good start but it's more than an alias
- Bash script
- Makefile
- Host specific Makefiles

A proper build system

- Cross platform
- Supports dependencies
- Supports finding #include files and libraries
- Supports parallel builds

Common build tools are Make, autoconf/automake, cmake, SCons

Types of Documentation

Minimal acceptable documentation

- ReadMe file
- Commented Source
- Bugs / TODO lists
- Changelog

Full Blown Documentation

- Model, Discretization and Code Verification Documentation
- Requirements Document
- Autogenerated Technical Documentation and code structure from commented source files
- End-User Document if applicable
- Issue Tracking

Scientific vs. Other Computing

- Scientific Computing
  - has to be right (or at least quantifiably wrong)
  - has to be fast
  - doesn't have to be easy to use
  - doesn't have to be pretty
- Other Types of Computing
  - errors are tolerable
  - speed is negotiable
  - often has to be pretty or easy to use or both

Trust No One . . . not even yourself

Unit Testing

- Tests individual units of source code
- A Unit is the smallest testable part of a code
- Each Unit test should be independent from others
- Best to write the Unit and the Unit test at the same time

Regression Testing

- Examples applications, unit tests, benchmark problems
- Catches unintended consequences of revisions
- Design of a suite of regression tests is an art

- Consider Code and Feature Coverage
- Test early and Test often
- Consider automating tests

Why Verify?

Reinhart and Kenneth S. Rogoff: Growth in a Time of Debt, 2010

- "Our main result is that whereas the link between growth and debt seems relatively weak at normal debt levels, median growth rates for countries with public debt over roughly 90 percent of GDP are about one percent lower than otherwise"
- Dataset: "inflation and GDP growth across varying levels of debt for 20 advanced countries over the period 1946 through 2009"
- Rogoff testified in front of the Senate Budget Committee

Inform Decision Makers

- France: 30 Billion Euros in Austerity Measures
- England: 11.5 Billion Pounds in Austerity Measures
- "Spain's national budget cuts of almost 14 percent and regional budget cuts of up to 10 percent in health and social services"

Verification Failure

Excel Error!

- "Instead of AVERAGE(L30:L49), AVERAGE(L30:L44) was used."
- When corrected, GDP/DEBT ratios above 90% had growth of 2.2%

Surely those were isolated events, right?

Other examples

- JPM VaR model for synthetic credit portfolio muted volatility by factor of two ($6B in losses)
- Mars Surveyor (metric versus english units)
- Sleipner A - North Sea Oil Platform Collapse (“serious error in the finite element analysis”)
- Therac-25 (overdosed more than two-dozen patients with gamma radiation)
- Patriot missile mistiming (only off by one third of a second!)

Verification

Verification of Scientific Software

- Verification ensures that the outputs of a computation accurately reflect the solution of the mathematical models.

Code Verification

- Ensuring that the code used in the simulation correctly implements the intended numerical discretization of the model.
  - This concept is *not* unique to Scientific Software

Solution Verification

- Are the errors from the numerical discretization sufficiently small?
- Is the convergence rate consistent with the numerical scheme?

Method of Exact Solutions

- Numerically solve the governing equations for which the solution can be determined analytically.

MMS

- Often, analytical solutions either:
  - Do not exist (Navier-Stokes)
  - Do not fully exercise equations (e.g. a symmetric solution, nonlinearities)
- Alleviate this using Method of Manufactured Solutions (MMS)
  - Simply put, we “create” our own solutions

Manufactured solution to Laplace's Equation

Laplace's Equation:

$$\nabla^2 \phi = 0$$

In two dimensions:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

"Manufacture" a solution, with two constants:

$$\phi(x, y) = (Ly - y)^2 (Ly + y)^2 + (Lx - x)^2 (Lx + x)^2$$

Calculating the Source Term

We insert our manufactured solution back into the governing equations:

$$\frac{\partial^2((Lx-x)^2(Lx+x)^2)}{\partial x^2} + \frac{\partial^2((Ly-y)^2(Ly+y)^2)}{\partial y^2} = 0$$

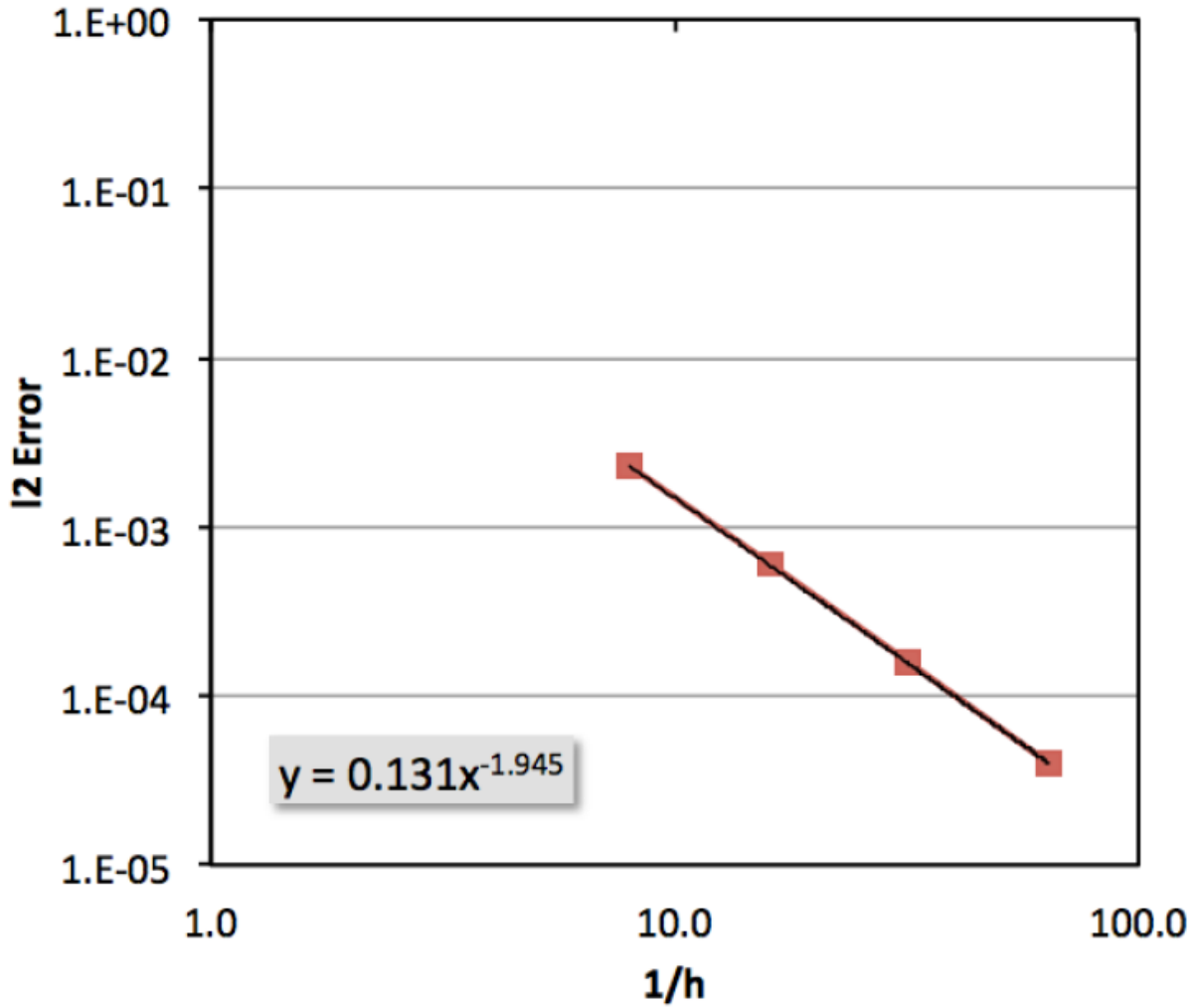
$$\begin{aligned}
&= 2(Lx-x)^2 - 8(Lx-x)(Lx+x) + 2(Lx+x)^2 \\
&+ 2(Ly-y)^2 - 8(Ly-y)(Ly+y) + 2(Ly+y)^2 \\
&\neq 0
\end{aligned}$$

This does not satisfy Laplace's Equation!

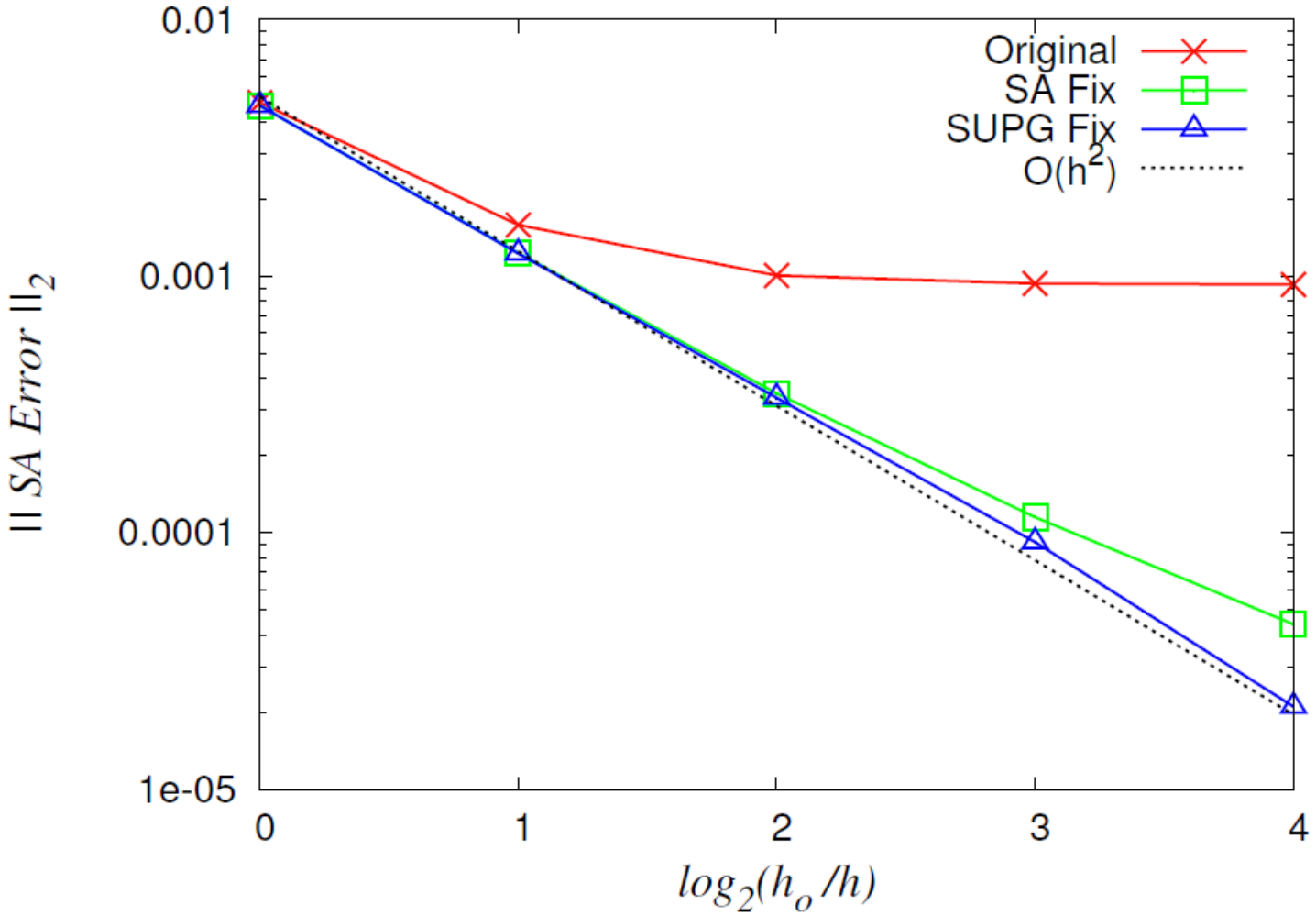
To balance the equation, add the residual to the RHS as a source term.

Example Results: What we're hoping for

2nd Order Central Finite-difference Scheme



This Process Finds Bugs



Useful for Detecting Subtle Bugs

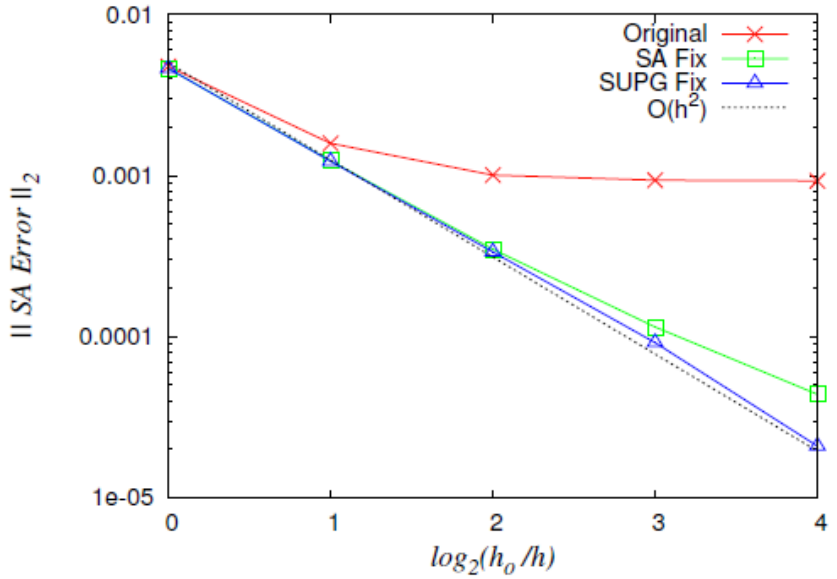
Verification of FIN-S

- FANS, Spalart-Allmaras
- Derivative:

$$\frac{d(sa)}{dx} = \frac{1}{\rho} * \left(\frac{d(\rho * sa)}{dx} - sa \frac{d\rho}{dx} \right)$$

- In code:

$$\frac{d(sa)}{dx} = \frac{1}{\rho} * \frac{d(\rho * sa)}{dx} - sa \frac{d\rho}{dx}$$



Summary

What we have learned:

- We can generate MS even for systems that do not have solutions
- The MMS is a powerful method to verify rates of convergence

Why is this not more commonly done?

- Solution Generation is complex and time intensive
- Providing high accuracy code is not as easy as you would think (hope?)

A Real Example

MMS Creation Process

- Start by "manufacturing" a suitable closed-form exact solution
- For example, the 10 parameter trigonometric solution of the form: (Roy, 2002)

$$\hat{u}(x,y,z,t) = \hat{u}_0 + \hat{u}_x\, f_s\left(\frac{a_{\hat{u}x}\pi x}{L}\right) + \hat{u}_y\, f_s\left(\frac{a_{\hat{u}y}\pi y}{L}\right) + \\ + \hat{u}_z\, f_s\left(\frac{a_{\hat{u}z}\pi z}{L}\right) + \hat{u}_t\, f_s\left(\frac{a_{\hat{u}_t}\pi t}{L}\right)$$

- Apply this solution to equations of interest, solve for source terms (residual)

Accomplished using packages such as Maple, Mathematica, SymPy, Macsyma, etc.

Maple MMS: 3D Navier-Stokes Energy Term

$$
\begin{aligned}
Qe = &-\frac{a_{px}\pi p_x}{L}\frac{\gamma}{\gamma-1}\sin\left(\frac{a_{px}\pi x}{L}\right)\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]+\\
&+\frac{a_{py}\pi p_y}{L}\frac{\gamma}{\gamma-1}\cos\left(\frac{a_{py}\pi y}{L}\right)\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]+\\
&-\frac{a_{pz}\pi p_z}{L}\frac{\gamma}{\gamma-1}\sin\left(\frac{a_{pz}\pi z}{L}\right)\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]+\\
&+\frac{a_{\rho x}\pi\rho_x}{2L}\cos\left(\frac{a_{\rho x}\pi x}{L}\right)\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]\left(\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2+\right.\\
&\left.+\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2+\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2\right)+\\
&-\frac{a_{\rho y}\pi\rho_y}{2L}\sin\left(\frac{a_{\rho y}\pi y}{L}\right)\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]\left(\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2+\right.\\
&\left.+\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2+\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2\right)+\\
&+\frac{a_{\rho z}\pi\rho_z}{2L}\cos\left(\frac{a_{\rho z}\pi z}{L}\right)\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]\left(\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2+\right.\\
&\left.+\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2+\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2\right)+\\
&+\frac{a_{ux}\pi u_x}{2L}\cos\left(\frac{a_{ux}\pi x}{L}\right)\left\{\left(\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]^2+\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]^2+\right.\\
&\left.+3\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]^2\right)\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]+\\
&+\left[p_0+p_x\cos\left(\frac{a_{px}\pi x}{L}\right)+p_y\sin\left(\frac{a_{py}\pi y}{L}\right)+p_z\cos\left(\frac{a_{pz}\pi z}{L}\right)\right]\frac{2\gamma}{(\gamma-1)}\Bigg\}+\\
&-\frac{a_{uy}\pi u_y}{L}\sin\left(\frac{a_{uy}\pi y}{L}\right)\left[v_0+v_x\cos\left(\frac{a_{vx}\pi x}{L}\right)+v_y\sin\left(\frac{a_{vy}\pi y}{L}\right)+v_z\sin\left(\frac{a_{vz}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\cdot\\
&\cdot\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]+\\
&-\frac{a_{uz}\pi u_z}{L}\sin\left(\frac{a_{uz}\pi z}{L}\right)\left[w_0+w_x\sin\left(\frac{a_{wx}\pi x}{L}\right)+w_y\sin\left(\frac{a_{wy}\pi y}{L}\right)+w_z\cos\left(\frac{a_{wz}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{a_{\rho x}\pi x}{L}\right)+\rho_y\cos\left(\frac{a_{\rho y}\pi y}{L}\right)+\rho_z\sin\left(\frac{a_{\rho z}\pi z}{L}\right)\right]\cdot\\
&\cdot\left[u_0+u_x\sin\left(\frac{a_{ux}\pi x}{L}\right)+u_y\cos\left(\frac{a_{uy}\pi y}{L}\right)+u_z\cos\left(\frac{a_{uz}\pi z}{L}\right)\right]+
\end{aligned}
$$

But wait, there's more!

$$
\begin{aligned}
&-\frac{\alpha_{\mu\mu}^2\pi^4 k\rho_x\left[2\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)^2\rho_x+\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\mu\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]\left[p_0+p_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+p_y\sin\left(\frac{\alpha_{\mu\eta}\pi y}{L}\right)+p_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]}{RL^2\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\mu\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^3}+\\
&-\frac{\alpha_{\eta\eta}^2\pi^2 k\rho_y\left[2\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)^2\rho_y+\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]\left[p_0+p_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+p_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+p_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]}{RL^2\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^3}+\\
&-\frac{4}{3}\frac{\alpha_{\mu\mu}^2\pi^2\mu v_x}{L^2}\left[\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)^2u_x-\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\left[u_0+u_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+u_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+u_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]+\\
&-\frac{\alpha_{\eta\eta}^2\pi^2\mu v_y}{L^2}\left[\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)^2u_y-\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)\left[u_0+u_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+u_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+u_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]+\\
&-\frac{\alpha_{\mu\mu}^2\pi^2\mu v_z}{L^2}\left[\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)^2u_z-\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\left[u_0+u_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+u_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+u_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]+\\
&-\frac{\alpha_{\mu\mu}^2\pi^2\mu v_x}{L^2}\left[\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)^2v_x-\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\left[v_0+v_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+v_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+v_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]+\\
&-\frac{4}{3}\frac{\alpha_{\eta\eta}^2\pi^2\mu v_y}{L^2}\left[\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)^2v_y-\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)\left[v_0+v_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+v_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+v_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]+\\
&-\frac{\alpha_{\mu\mu}^2\pi^2\mu v_z}{L^2}\left[\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)^2v_z-\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\left[v_0+v_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+v_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+v_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]+\\
&-\frac{\alpha_{\mu\mu}^2\pi^2\mu w_x}{L^2}\left[\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)^2w_x-\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\left[w_0+w_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+w_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+w_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]+\\
&-\frac{\alpha_{\eta\eta}^2\pi^2\mu w_y}{L^2}\left[\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)^2w_y-\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)\left[w_0+w_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+w_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+w_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]+\\
&-\frac{4}{3}\frac{\alpha_{\mu\mu}^2\pi^2\mu w_z}{L^2}\left[\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)^2w_z-\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\left[w_0+w_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+w_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+w_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\right]+
\end{aligned}
$$

$$
\begin{aligned}
&-\frac{2\alpha_{\mu\mu}\alpha_{\mu\mu}\pi^2 k\rho_x\rho_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^2}+\\
&-\frac{2\alpha_{\eta\eta}\alpha_{\eta\eta}\pi^2 k\rho_y\rho_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^2}+\\
&-\frac{2\alpha_{\mu\mu}\alpha_{\mu\mu}\pi^2 k\rho_z\rho_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^2}+\\
&+\frac{4}{3}\frac{\alpha_{\mu\mu}\alpha_{\eta\eta}\pi^2\mu v_x v_y}{L^2}\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\\
&-\frac{4}{3}\frac{\alpha_{\mu\mu}\alpha_{\mu\mu}\pi^2\mu v_x w_z}{L^2}\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)+\\
&-\frac{2\alpha_{\eta\eta}\alpha_{\eta\eta}\pi^2\mu v_y w_z}{L^2}\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)+\\
&+\frac{2\alpha_{\mu\mu}\alpha_{\mu\mu}\pi^2\mu v_z w_x}{L^2}\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)+\\
&-\frac{4}{3}\frac{\alpha_{\eta\eta}\alpha_{\eta\eta}\pi^2\mu v_y w_z}{L^2}\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)+\\
&-\frac{2\alpha_{\eta\eta}\alpha_{\eta\eta}\pi^2\mu v_z w_y}{L^2}\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right).
\end{aligned}
$$

$$
\begin{aligned}
&-\frac{\alpha_{\mu\mu}\pi v_x}{L}\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\left[u_0+u_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+u_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+u_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\cdot\\
&\cdot\left[v_0+v_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+v_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+v_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]+\\
&+\frac{\alpha_{\eta\eta}\pi v_y}{2L}\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)\left\{\left(\left[u_0+u_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+u_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+u_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^2+\left[w_0+w_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+w_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+w_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^2+\right.\right.\\
&\left.\left.+3\left[v_0+v_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+v_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+v_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^2\right)\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]+\right.\\
&\left.+\left[p_0+p_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+p_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+p_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\frac{2\gamma}{\gamma-1}\right\}+\\
&+\frac{\alpha_{\mu\mu}\pi v_z}{L}\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\left[w_0+w_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+w_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+w_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\cdot\\
&\cdot\left[v_0+v_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+v_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+v_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]+\\
&+\frac{\alpha_{\mu\mu}\pi w_z}{L}\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)\left[u_0+u_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+u_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+u_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\cdot\\
&\cdot\left[w_0+w_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+w_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+w_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]+\\
&+\frac{\alpha_{\eta\eta}\pi v_y}{L}\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)\left[v_0+v_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+v_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+v_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\cdot\\
&\cdot\left[u_0+u_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+u_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+u_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]+\\
&-\frac{\alpha_{\mu\mu}\pi v_z}{2L}\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\left\{\left(3\left[u_0+u_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+u_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+u_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^2+\left[v_0+v_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+v_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+v_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^2+\right.\right.\\
&\left.\left.+\left[w_0+w_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+w_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+w_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]^2\right)\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]+\right.\\
&\left.+\left[p_0+p_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+p_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+p_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]\frac{2\gamma}{(\gamma-1)}\right\}+\\
&+\frac{\alpha_{\mu\mu}^2\pi^2 k p_x\cos\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]}+\\
&+\frac{\alpha_{\eta\eta}^2\pi^2 k p_y\sin\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]}+\\
&+\frac{\alpha_{\mu\mu}^2\pi^2 k p_z\cos\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)}{L^2R\left[\rho_0+\rho_x\sin\left(\frac{\alpha_{\mu\mu}\pi x}{L}\right)+\rho_y\cos\left(\frac{\alpha_{\eta\eta}\pi y}{L}\right)+\rho_z\sin\left(\frac{\alpha_{\mu\mu}\pi z}{L}\right)\right]}+
\end{aligned}
$$

C-code output

```
RHO = rho_0 + rho_x * sin(a_rhox * PI * x / L) + rho_y * cos(a_rhoy * PI * y / L) + rho_z * sin(a_rhoz * PI * z / L) + rho_t * sin(a_rhot * PI * t / L);
 U = u_0 + u_x * sin(a_ux * PI * x / L) + u_y * cos(a_uy * PI * y / L) + u_z * cos(a_uz * PI * z / L) + u_t * cos(a_ut * PI * t / L);
 V = v_0 + v_x * cos(a_vx * PI * x / L) + v_y * sin(a_vy * PI * y / L) + v_z * sin(a_vz * PI * z / L) + v_t * sin(a_vt * PI * t / L);
 W = w_0 + w_x * sin(a_wx * PI * x / L) + w_y * sin(a_wy * PI * y / L) + w_z * cos(a_wz * PI * z / L) + w_t * cos(a_wt * PI * t / L);
 P = p_0 + p_x * cos(a_px * PI * x / L) + p_y * sin(a_py * PI * y / L) + p_z * cos(a_pz * PI * z / L) + p_t * cos(a_pt * PI * t / L);
 M1 = A_mu * pow(P / R / RHO, 0.3e1 / 0.2e1);
 M2 = P / R / RHO + B_mu;
 MU = M1 / M2;
 Q_e = (0.3e1 * a_ux * u_x * cos(a_ux * PI * x / L) + a_vy * v_y * cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * RHO * U * U / L / 0.2e1 + (a_ux * u_x * cos(a_ux * PI
* x / L) + 0.3e1 * a_vy * v_y * cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * RHO * V * V / L / 0.2e1 + (a_ux * u_x * cos(a_ux * PI * x / L) + a_vy * v_y * cos(a_vy *
PI * y / L) - 0.3e1 * a_wz * w_z * sin(a_wz * PI * z / L)) * PI * RHO * W * W / L / 0.2e1 + (0.4e1 * a_ux * a_ux * u_x * sin(a_ux * PI * x / L) + 0.3e1 * a_uy * a_uy * u_y * cos(a_uy * PI * y
/ L) + 0.3e1 * a_uz * a_uz * u_z * cos(a_uz * PI * z / L)) * MU * PI * PI * U * pow(L, -0.2e1) / 0.3e1 + (0.3e1 * a_vx * a_vx * v_x * cos(a_vx * PI * x / L) + 0.4e1 * a_vy * a_vy * v_y *
sin(a_vy * PI * y / L) + 0.3e1 * a_vz * a_vz * v_z * sin(a_vz * PI * z / L)) * MU * PI * PI * V * pow(L, -0.2e1) / 0.3e1 + (0.3e1 * a_wx * a_wx * w_x * sin(a_wx * PI * x / L) + 0.3e1 * a_wy *
a_wy * w_y * sin(a_wy * PI * y / L) + 0.4e1 * a_wz * a_wz * w_z * cos(a_wz * PI * z / L)) * MU * PI * PI * W * pow(L, -0.2e1) / 0.3e1 + (a_ux * u_x * cos(a_ux * PI * x / L) + a_vy * v_y *
cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * P / (Gamma - 0.1e1) / L - (0.2e1 * a_rhox * a_px * rho_x * p_x * cos(a_rhox * PI * x / L) * sin(a_px * PI * x / L) + 0.2e1
* a_rhoy * a_py * rho_y * p_y * sin(a_rhoy * PI * y / L) * cos(a_py * PI * y / L) + 0.2e1 * a_rhoz * a_pz * rho_z * p_z * cos(a_rhoz * PI * z / L) * sin(a_pz * PI * z / L)) * PI * PI * k *
pow(L, -0.2e1) / R * pow(RHO, -0.2e1) + (U * U + V * V + W * W) * a_rhot * PI * rho_t * cos(a_rhot * PI * t / L) / L / 0.2e1 - a_pt * PI * p_t * sin(a_pt * PI * t / L) / (Gamma - 0.1e1) / L -
(a_uy * u_y * sin(a_uy * PI * y / L) + a_vx * v_x * sin(a_vx * PI * x / L)) * PI * RHO * U * V / L - (a_uz * u_z * sin(a_uz * PI * z / L) - a_wx * w_x * cos(a_wx * PI * x / L)) * PI * RHO * U
* W / L + (a_vz * v_z * cos(a_vz * PI * z / L) + a_wy * w_y * cos(a_wy * PI * y / L)) * PI * RHO * V * W / L + (a_px * a_px * p_x * cos(a_px * PI * x / L) + a_py * a_py * p_y * sin(a_py * PI
* y / L) + a_pz * a_pz * p_z * cos(a_pz * PI * z / L)) * PI * PI * k * pow(L, -0.2e1) / R / RHO - (0.4e1 * a_ux * a_ux * u_x * u_x * pow(cos(a_ux * PI * x / L), 0.2e1) - 0.4e1 * a_ux * a_vy *
u_x * v_y * cos(a_ux * PI * x / L) * cos(a_vy * PI * y / L) + 0.4e1 * a_ux * a_wz * u_x * w_z * cos(a_ux * PI * x / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_uy * a_uy * u_y * u_y *
pow(sin(a_uy * PI * y / L), 0.2e1) + 0.6e1 * a_uy * a_vx * u_y * v_x * sin(a_uy * PI * y / L) * sin(a_vx * PI * x / L) + 0.3e1 * a_uz * a_uz * u_z * u_z * pow(sin(a_uz * PI * z / L), 0.2e1) -
0.6e1 * a_uz * a_wx * u_z * w_x * sin(a_uz * PI * z / L) * cos(a_wx * PI * x / L) + 0.3e1 * a_vx * a_vx * v_x * v_x * pow(sin(a_vx * PI * x / L), 0.2e1) + 0.4e1 * a_vy * a_vy * v_y * v_y *
pow(cos(a_vy * PI * y / L), 0.2e1) + 0.4e1 * a_vy * a_wz * v_y * w_z * cos(a_vy * PI * y / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_vz * a_vz * v_z * v_z * pow(cos(a_vz * PI * z / L), 0.2e1) +
0.6e1 * a_vz * a_wy * v_z * w_y * cos(a_vz * PI * z / L) * cos(a_wy * PI * y / L) + 0.3e1 * a_wx * a_wx * w_x * w_x * pow(cos(a_wx * PI * x / L), 0.2e1) + 0.3e1 * a_wy * a_wy * w_y * w_y *
pow(cos(a_wy * PI * y / L), 0.2e1) + 0.4e1 * a_wz * a_wz * w_z * w_z * pow(sin(a_wz * PI * z / L), 0.2e1)) * MU * PI * PI * pow(L, -0.2e1) / 0.3e1 + (a_ux * u_x * cos(a_ux * PI * x / L) +
a_vy * v_y * cos(a_vy * PI * y / L) - a_wz * w_z * sin(a_wz * PI * z / L)) * PI * P / L - a_ut * PI * u_t * RHO * U * sin(a_ut * PI * t / L) / L + a_vt * PI * v_t * RHO * V * cos(a_vt * PI *
t / L) / L - a_wt * PI * w_t * RHO * W * sin(a_wt * PI * t / L) / L + (U * U + V * V + W * W) * a_rhox * PI * rho_x * U * cos(a_rhox * PI * x / L) / L / 0.2e1 - (U * U + V * V + W * W) *
a_rhoy * PI * rho_y * V * sin(a_rhoy * PI * y / L) / L / 0.2e1 + (U * U + V * V + W * W) * a_rhoz * PI * rho_z * W * cos(a_rhoz * PI * z / L) / L / 0.2e1 - (a_rhox * a_rhox * rho_x *
sin(a_rhox * PI * x / L) + a_rhoy * a_rhoy * rho_y * cos(a_rhoy * PI * y / L) + a_rhoz * a_rhoz * rho_z * sin(a_rhoz * PI * z / L)) * PI * PI * k * P * pow(L, -0.2e1) / R * pow(RHO, -0.2e1) -
(0.2e1 * a_rhox * a_rhox * rho_x * rho_x * pow(cos(a_rhox * PI * x / L), 0.2e1) + 0.2e1 * a_rhoy * a_rhoy * rho_y * rho_y * pow(sin(a_rhoy * PI * y / L), 0.2e1) + 0.2e1 * a_rhoz * a_rhoz *
rho_z * rho_z * pow(cos(a_rhoz * PI * z / L), 0.2e1)) * PI * PI * k * P * pow(L, -0.2e1) / R * pow(RHO, -0.3e1) - (0.3e1 * a_rhox * a_ux * rho_x * u_x * cos(a_rhox * PI * x / L) * sin(a_ux *
PI * z / L) - 0.3e1 * a_rhox * a_wx * rho_x * w_x * cos(a_rhox * PI * x / L) * cos(a_wx * PI * x / L) + 0.3e1 * a_rhoy * a_vz * rho_y * v_z * sin(a_rhoy * PI * y / L) * cos(a_vz * PI * z / L)
+ 0.3e1 * a_rhoy * a_wy * rho_y * w_y * sin(a_rhoy * PI * y / L) * cos(a_wy * PI * y / L) + 0.2e1 * a_rhoz * a_ux * rho_z * u_x * cos(a_rhoz * PI * z / L) * cos(a_ux * PI * x / L) + 0.2e1 *
a_rhoz * a_vy * rho_z * v_y * cos(a_rhoz * PI * z / L) * cos(a_vy * PI * y / L) + 0.4e1 * a_rhoz * a_wz * rho_z * w_z * cos(a_rhoz * PI * z / L) * sin(a_wz * PI * z / L)) * MU * (0.3e1 * B_mu
* R * RHO + P) * PI * PI * W / (B_mu * R * RHO + P) * pow(L, -0.2e1) / RHO / 0.6e1 - a_pz * PI * p_z * Gamma * W * sin(a_pz * PI * z / L) / (Gamma - 0.1e1) / L - (0.3e1 * a_px * a_uy * p_x *
u_y * sin(a_px * PI * x / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_px * a_vx * p_x * v_x * sin(a_px * PI * x / L) * sin(a_vx * PI * x / L) - 0.2e1 * a_py * a_ux * p_y * u_x * cos(a_py * PI * y
/ L) * cos(a_ux * PI * x / L) + 0.4e1 * a_py * a_vy * p_y * v_y * cos(a_py * PI * y / L) * cos(a_vy * PI * y / L) + 0.2e1 * a_py * a_wz * p_y * w_z * cos(a_py * PI * y / L) * sin(a_wz * PI *
z / L) - 0.3e1 * a_pz * a_vz * p_z * v_z * sin(a_pz * PI * z / L) * cos(a_vz * PI * z / L) - 0.3e1 * a_pz * a_wy * p_z * w_y * sin(a_pz * PI * z / L) * cos(a_wy * PI * y / L)) * (0.3e1 * B_mu
* R * RHO + P) * MU * PI * PI * V / (B_mu * R * RHO + P) * pow(L, -0.2e1) / P / 0.6e1 + a_py * PI * p_y * Gamma * V * cos(a_py * PI * y / L) / (Gamma - 0.1e1) / L - (0.3e1 * a_rhox * a_uy *
rho_x * u_y * cos(a_rhox * PI * x / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_rhox * a_vx * rho_x * v_x * cos(a_rhox * PI * x / L) * sin(a_vx * PI * x / L) - 0.2e1 * a_rhoy * a_ux * rho_y * u_x
* sin(a_rhoy * PI * y / L) * cos(a_ux * PI * x / L) + 0.4e1 * a_rhoy * a_vy * rho_y * v_y * sin(a_rhoy * PI * y / L) * cos(a_vy * PI * y / L) + 0.2e1 * a_rhoy * a_wz * rho_y * w_z *
sin(a_rhoy * PI * y / L) * sin(a_wz * PI * z / L) - 0.3e1 * a_rhoz * a_vz * rho_z * v_z * cos(a_rhoz * PI * z / L) * cos(a_vz * PI * z / L) - 0.3e1 * a_rhoz * a_wy * rho_z * w_y * cos(a_rhoz
* PI * z / L) * cos(a_wy * PI * y / L)) * MU * (0.3e1 * B_mu * R * RHO + P) * PI * PI * V / (B_mu * R * RHO + P) * pow(L, -0.2e1) / RHO / 0.6e1 + (0.4e1 * a_rhox * a_ux * rho_x * u_x *
cos(a_rhox * PI * x / L) * cos(a_ux * PI * x / L) - 0.2e1 * a_rhox * a_vy * rho_x * v_y * cos(a_rhox * PI * x / L) * cos(a_vy * PI * y / L) + 0.2e1 * a_rhox * a_wz * rho_x * w_z * cos(a_rhox
* PI * x / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_rhoy * a_uy * rho_y * u_y * sin(a_rhoy * PI * y / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_rhoy * a_vx * rho_y * v_x * sin(a_rhoy * PI * y /
L) * sin(a_vx * PI * x / L) - 0.3e1 * a_rhoz * a_uz * rho_z * u_z * cos(a_rhoz * PI * z / L) * sin(a_uz * PI * z / L) + 0.3e1 * a_rhoz * a_wx * rho_z * w_x * cos(a_rhoz * PI * z / L) *
cos(a_wx * PI * x / L)) * MU * (0.3e1 * B_mu * R * RHO + P) * PI * PI * U / (B_mu * R * RHO + P) * pow(L, -0.2e1) / RHO / 0.6e1 - a_px * PI * p_x * Gamma * U * sin(a_px * PI * x / L) / (Gamma
- 0.1e1) / L + (0.4e1 * a_px * a_ux * p_x * u_x * sin(a_px * PI * x / L) * cos(a_ux * PI * x / L) - 0.2e1 * a_px * a_vy * p_x * v_y * sin(a_px * PI * x / L) * cos(a_vy * PI * y / L) + 0.2e1 *
a_px * a_wz * p_x * w_z * sin(a_px * PI * x / L) * sin(a_wz * PI * z / L) + 0.3e1 * a_py * a_uy * p_y * u_y * cos(a_py * PI * y / L) * sin(a_uy * PI * y / L) + 0.3e1 * a_py * a_vx * p_y * v_x
* cos(a_py * PI * y / L) * sin(a_vx * PI * x / L) - 0.3e1 * a_pz * a_uz * p_z * u_z * sin(a_pz * PI * z / L) * sin(a_uz * PI * z / L) + 0.3e1 * a_pz * a_wx * p_z * w_x * sin(a_pz * PI * z /
L) * cos(a_wx * PI * x / L)) * (0.3e1 * B_mu * R * RHO + P) * MU * PI * PI * U / (B_mu * R * RHO + P) * pow(L, -0.2e1) / P / 0.6e1 - (0.3e1 * a_px * a_uz * p_x * u_z * sin(a_px * PI * x / L)
* sin(a_uz * PI * z / L) - 0.3e1 * a_px * a_wx * p_x * w_x * sin(a_px * PI * x / L) * cos(a_wx * PI * x / L) + 0.3e1 * a_py * a_vz * p_y * v_z * cos(a_py * PI * y / L) * cos(a_vz * PI * z /
L) + 0.3e1 * a_py * a_wy * p_y * w_y * cos(a_py * PI * y / L) * cos(a_wy * PI * y / L) + 0.2e1 * a_pz * a_ux * p_z * u_x * sin(a_pz * PI * z / L) * cos(a_ux * PI * x / L) + 0.2e1 * a_pz *
a_vy * p_z * v_y * sin(a_pz * PI * z / L) * cos(a_vy * PI * y / L) + 0.4e1 * a_pz * a_wz * p_z * w_z * sin(a_pz * PI * z / L) * sin(a_wz * PI * z / L)) * (0.3e1 * B_mu * R * RHO + P) * MU *
PI * PI * W / (B_mu * R * RHO + P) * pow(L, -0.2e1) / P / 0.6e1;
```

Manufactured Analytical Solutions Abstractions Library

Goal: Provide a repository and standardized interface for MMS usage

High Priority:

- Extreme fidelity to generated MMS
- Portability
- Traceability
- Extensible

Low Priority:

- Speed/Performance

Verifying the "Verifier"

Precision is not negotiable.

MASA Testing

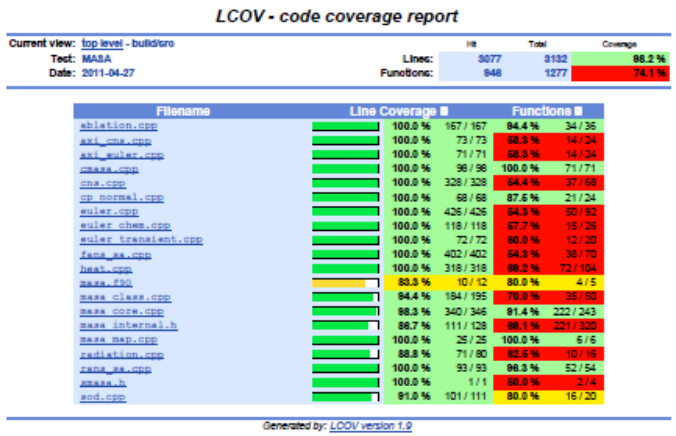
- Error target $<$ 1e-15
  - Absolute error on local machines
  - Relative error (other)
  - On all supported compiler sets
- -O0 not sufficient
  - -fp-model precise (Intel)
  - -fno-unsafe-math-optimizations (GNU)
  - -Kieee -Mnofpapprox (PGI)
- "make check"
  - Run by Buildbot every two hours

```
[nick@magus trunk]$ make check
----------------------------------------
Initializing MASA Tests
----------------------------------------
PASS: init.sh
PASS: misc
PASS: fail_cond
PASS: catch_exception
PASS: register
PASS: poly
PASS: uninit
PASS: vec
PASS: purge
PASS: heat_const_steady
PASS: euler1d



.           .
.           .
.           .


----------------------------------------
Finalizing MASA Tests
----------------------------------------
==================
All 65 tests passed
==================
```

Portability

Testing

- GIT: version control
- Buildbot: automated testing
- GCOV: line coverage
  - 15,826 lines of code
  - 13,195 lines of testing
  - 98%+ line coverage



LCOV - code coverage report

Current view: top level - build/src
Test: MASA
Date: 2011-04-27

	Hit	Total	Coverage
Lines:	3077	3132	98.2 %
Functions:	948	1277	74.1 %

Filename	Line Coverage		Functions	
ablation.cpp	100.0 %	167 / 167	94.4 %	34 / 36
axi_cns.cpp	100.0 %	73 / 73	58.3 %	14 / 24
axi_euler.cpp	100.0 %	71 / 71	58.3 %	14 / 24
cmasa.cpp	100.0 %	98 / 98	100.0 %	71 / 71
cns.cpp	100.0 %	328 / 328	54.4 %	37 / 68
cp_normal.cpp	100.0 %	68 / 68	87.5 %	21 / 24
euler.cpp	100.0 %	426 / 426	54.3 %	50 / 92
euler_chem.cpp	100.0 %	118 / 118	57.7 %	15 / 26
euler_transient.cpp	100.0 %	72 / 72	60.0 %	12 / 20
fana_ss.cpp	100.0 %	402 / 402	54.3 %	38 / 70
heat.cpp	100.0 %	318 / 318	68.2 %	72 / 104
masa.f90	83.3 %	10 / 12	80.0 %	4 / 5
masa_class.cpp	84.4 %	164 / 195	78.0 %	39 / 50
masa_core.cpp	98.3 %	340 / 346	91.4 %	222 / 243
masa_internal.h	86.7 %	111 / 128	68.1 %	221 / 320
masa_map.cpp	100.0 %	25 / 25	100.0 %	6 / 6
radiation.cpp	88.8 %	71 / 80	65.6 %	10 / 15
rana_ss.cpp	100.0 %	93 / 93	96.3 %	52 / 54
smasa.h	100.0 %	1 / 1	50.0 %	2 / 4
sod.cpp	91.0 %	101 / 111	80.0 %	16 / 20

Generated by: LCOV version 1.9

Software Environment

- Built with: Autotools, C++
- Supports Intel, GNU, Portland Group compilers
- C/C++ interfaces
- Fortran interfaces provided through iso_c_bindings
  - Fortran 2003 Standard
- Python interfaces generated with SWIG

Traceability

Doxygen provides code and model documentation

where $\phi = \rho, u, v, w$ or p, and $f_x(\cdot)$ functions denote either sine or cosine function. Note that in this case, ϕ_x, ϕ_y and ϕ_z are constants and the subscripts do not denote differentiation.

Although ? provide the constants used in the manufactured solutions for the 2D supersonic and subsonic cases for Euler and Navier-Stokes equations, only the source term for the 2D mass conservation equation (3.20) is presented.

Source terms for mass conservation (Q_ρ), momentum (Q_u, Q_v and Q_w) and total energy (Q_{e_t}) equations are obtained by symbolic manipulations of compressible steady Euler equations above using Maple 13 (?) and are presented in the following sections for the one, two and three-dimensional cases.

3.2.2.1 1D Steady Euler

The manufactured analytical solutions (3.52) for each one of the variables in one-dimensional case of Euler equations are:

$$\begin{aligned} \rho(x) &= \rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) \\ u(x) &= u_0 + u_x \sin\left(\frac{a_{ux}\pi x}{L}\right) \\ p(x) &= p_0 + p_x \cos\left(\frac{a_{px}\pi x}{L}\right) \end{aligned} \tag{3.26}$$

The MMS applied to Euler equations consists in modifying the 1D Euler equations (3.20) – (3.22) by adding a source term to the right-hand side of each equation:

$$\begin{aligned} \frac{\partial(\rho u)}{\partial x} &= Q_\rho \\ \frac{\partial(\rho u^2)}{\partial x} + \frac{\partial(p)}{\partial x} &= Q_u \\ \frac{\partial(\rho u e_t)}{\partial x} + \frac{\partial(pu)}{\partial x} &= Q_{e_t} \end{aligned} \tag{3.27}$$

so the modified set of equations (3.27) conveniently has the analytical solution given in Equation (3.53).

Source terms Q_ρ, Q_u and Q_{e_t} are obtained by symbolic manipulations of equations above using Maple and are presented in the following sections. The following auxiliary variables have been included in order to improve readability and computational efficiency:

$$\begin{aligned} \mathrm{Rho}_1 &= \rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) \\ U_1 &= u_0 + u_x \sin\left(\frac{a_{ux}\pi x}{L}\right) \\ \mathrm{P}_1 &= p_0 + p_x \cos\left(\frac{a_{px}\pi x}{L}\right) \end{aligned}$$

where the subscripts refer to the 1D case.

The mass conservation equation written as an operator is:

$$\mathcal{L} = \frac{\partial(\rho u)}{\partial x}$$

k	u_0	u_x	u_y	u_z	v_0	L
v_0	v_x	v_y	v_z	w_0	w_x	w_y
rho_0	rho_x	rho_y	rho_z	p_0	p_y	p_z
a_px	a_py	a_pz	a_rhox	a_rhoy	a_rhoz	a_ux
a_uy	a_uz	a_vx	a_vy	a_vz	a_wx	a_wy
a_wz	mu	Gamma				

Table 3.6: Parameters used by the 3D Steady Euler

- masa_eval_2d_exact_u()
- masa_eval_2d_exact_v()
- masa_eval_2d_exact_p()
- masa_eval_2d_exact_rho()
- masa_eval_2d_grad_u()
- masa_eval_2d_grad_v()
- masa_eval_2d_grad_p()
- masa_eval_2d_grad_rho()

3.2.3.3 3D Steady Euler

Initialization:

- euler_3d

Functions:

- masa_init()
- masa_eval_3d_source_rho_u()
- masa_eval_3d_source_rho_v()
- masa_eval_3d_source_rho_w()
- masa_eval_3d_source_rho_e()
- masa_eval_3d_source_rho()
- masa_eval_3d_exact_u()
- masa_eval_3d_exact_v()
- masa_eval_3d_exact_w()
- masa_eval_3d_exact_p()

Available Solutions in MASA 0.50.0

Equations	Dimensions	Time
Euler	1,2,3, axi	Transient, Steady
Non linear heat conduction	1,2,3	Transient, Steady
Navier-Stokes	1,2,3, axi	Transient, Steady
N-S + Sutherland	3	Transient, Steady
N-S + ablation	1	Transient, Steady
Burgers	2	Transient, Steady
Sod Shock Tube	1	Transient
Euler + chemistry	1	Steady
RANS: Spalart-Allmaras	1	Steady
FANS: SA	2	Steady
FANS: SA + wall	2	Steady
Radiation	1	Steady
SMASA: Gaussian	1	Steady

Importing New Solutions

Requirements

- Latex documents can be loaded directly into MASA documentation
  - Model document detailing analytical solution and source terms
  - Interface documentation detailing parameters and functions
- Source and analytical terms in C/C++/Fortran90
  - Can be integrated into your local MASA copy automagically (perl!)
  - Submit a patch
    - (unit tests would be nice)
- Willingness to share
- Publish these solutions!

- Success of MASA depends on use as a community tool

Shortcomings

Symbolic Shortcomings

- Even with factorization, source terms still massive
- Generating manufactured solutions was a **full time job** at PECOS
- Everything we have discussed has been generated outside of MASA
  - Not thrilled with Maple, Mathematica

Enter Automatic Differentiation

- AD numerically evaluates the derivative of a function
  - applies chain rule repeatedly
- Superior error characteristics (round-off)
- Slow (but we barely care)
- Several libraries: NAG, Sacado, MetaPhysicL, etc.

Snapshot

Release

- MASA 0.50.0 current release
- `https://github.com/manufactured-solutions/MASA`
- Open source, LGPL V2.1, free

Publications

- "MASA: a library for verification using manufactured analytical solutions"
  - DOI: 10.1007/s00366-012-0267-9
- A TRANSIENT MANUFACTURED SOLUTION FOR THE COMPRESSIBLE NAVIER-STOKES EQUATIONS WITH A POWER LAW VISCOSITY
- Manufactured Solutions for the Favre-Averaged Navier-Stokes Equations with Eddy-Viscosity Turbulence Models
- "A Linear Regression Framework for the Verification of Bayesian Model Calibration Algorithms" (submitted to ASME VVUQ)

Researching

- Define what you want from all packages
- Define what a package aims to provide (scope)

Reach out...

- Talk to other users in your field
- Talk to the developers of the package
  - Stable versus unstable code
  - How to contribute?
  - How to cite?
  - IRC / chat room
  - Mailing list

Things to pay attention to...

- Variable names
- Portability
- Source formatting
- Default build system has warnings enabled
- Build system has a test flag

Other things to note

- Finding a bug doesn't rule it out
- Does the project accept patches?
- Licensing
- If TACC has a module for, it is worth evaluating
- Commercial software doesn't mean high quality
- Be polite to the developers

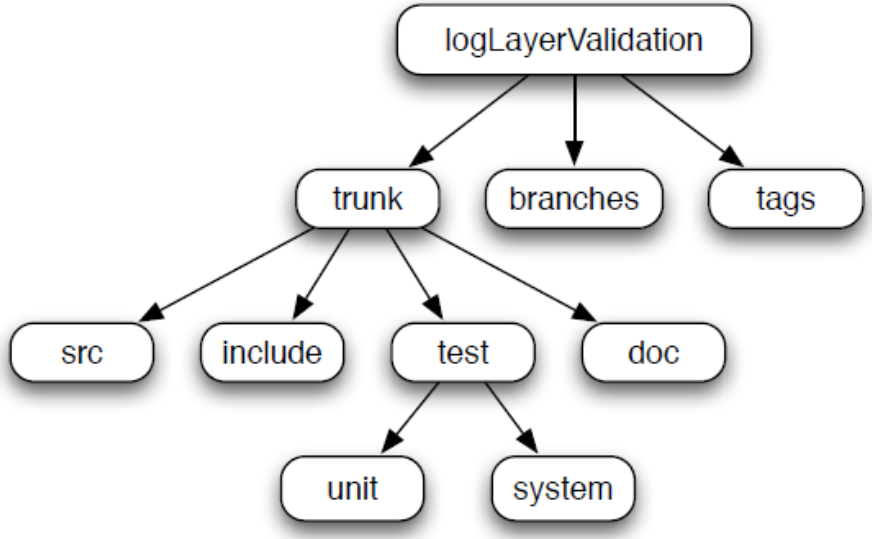
Sometimes you have to bite the bullet …

- Choose the right solution for your problem
  - Language
  - Hardware and portability
  - Open source / free vs performant (commercial) compilers / math libraries
- We ready to right code? Almost
  - Documentation
  - Requirements
  - Testing/Verification
  - Automation

We need a place to work…

- To do any work, our project needs a place to live
- Should not just be in your home directory => use version control and work on a local copy
- E.g. : svn co https://collab.tacc.utexas.edu/…
- Start working…
  - Create directories: svn mkdir foo
  - Add files: svn add foo/bar.C
  - Commit to repo: svn ci –m “enlightening commit”

Structure is important



- Trunk: primary development location
- Tags: “tagged” versions (e.g. releases)
- Braches: additional development locations for major refactoring

Model Documentation

- Before you implement anything, you should be able to write down *precisely* what you plan to implement
- Easy to keep track of what you have done and are doing
- Helps to produce papers and thesis after code is done

Model Documentation Cont.

- Ideally, everything necessary to implement your application, including:
  - All model equations, e.g., governing pdes or odes
  - Boundary conditions
  - Detailed discretization approach
  - Solution technique
  - Verification plan
- Implementation details are not required
  - That is, model document is not code documentation, not a developer guide, nor a user guide

LogLayer Model Document

1 Introduction **1**

2 Overlap Layer Mean Velocity Models **1**
- 2.1 Introduction to Overlap Layer Velocity Profile Modeling . . . 2
- 2.2 Logarithmic Forms . . . 2
- 2.3 Power Forms . . . 3
- 2.4 Model Summary . . . 4

3 The Experimental Data **4**
- 3.1 Measurements and Instrumentation . . . 4
- 3.2 Uncertainties . . . 4

4 The Calibration Problem **4**
- 4.1 Bayesian Formulation Overview . . . 7
- 4.2 Likelihood Models . . . 7
  - 4.2.1 Uncertain Velocity Measurements Only . . . 7
  - 4.2.2 Uncertain Velocity and Skin Friction Measurements . . . 8
  - 4.2.3 Uncertain Velocity, Skin Friction, and Wall Location . . . 9
- 4.3 Prior PDFs . . . 9

5 Verification Plan **9**

Select External Software

- Know what capability we desire (documented in model document)
- Now we need to decide how to implement that capability
- First question: What do I need to do myself, and what can I outsource?

Select External Software

- First question: What do I need to do myself, and what can I outsource?
- Input parsing: GRVY, Boost program options
- Linear & Non-linear solvers: Petsc, Trilinos
- Finite Elements: Libmesh, Deal II
- General Scientific Comptuing: GSL
- UQ/Statistical Algorithms: QUESO, DAKOTA
- MANY, MANY, MANY MORE!

Spring Mass Damper Example

- Interested in effects of uncertainty and validation philosophy
  - Input parser: GRVY
  - Numerical algorithms: GSL
  - Statistical algorithms: QUESO
- So, I only need to provide
  - Model routines
  - Data handling and uncertainty descriptions

Choose build system

- Have selected some libraries to lighten the workload
- About ready to start coding
- Just need a way to compile & link reproducibly
- Make, Autotools, SCons: you make the call

Unit Testing

- Ready to start writing code, which means we're ready to start writing tests!
- Main idea: Test a "unit" (e.g., a function) of code in isolation from everything else
- Motivation
  - Easier to write tests from small pieces of code rather than entire applications
  - Helps promote modular code design and frequent refactoring
  - Helps identify problem areas when system-level tests fail

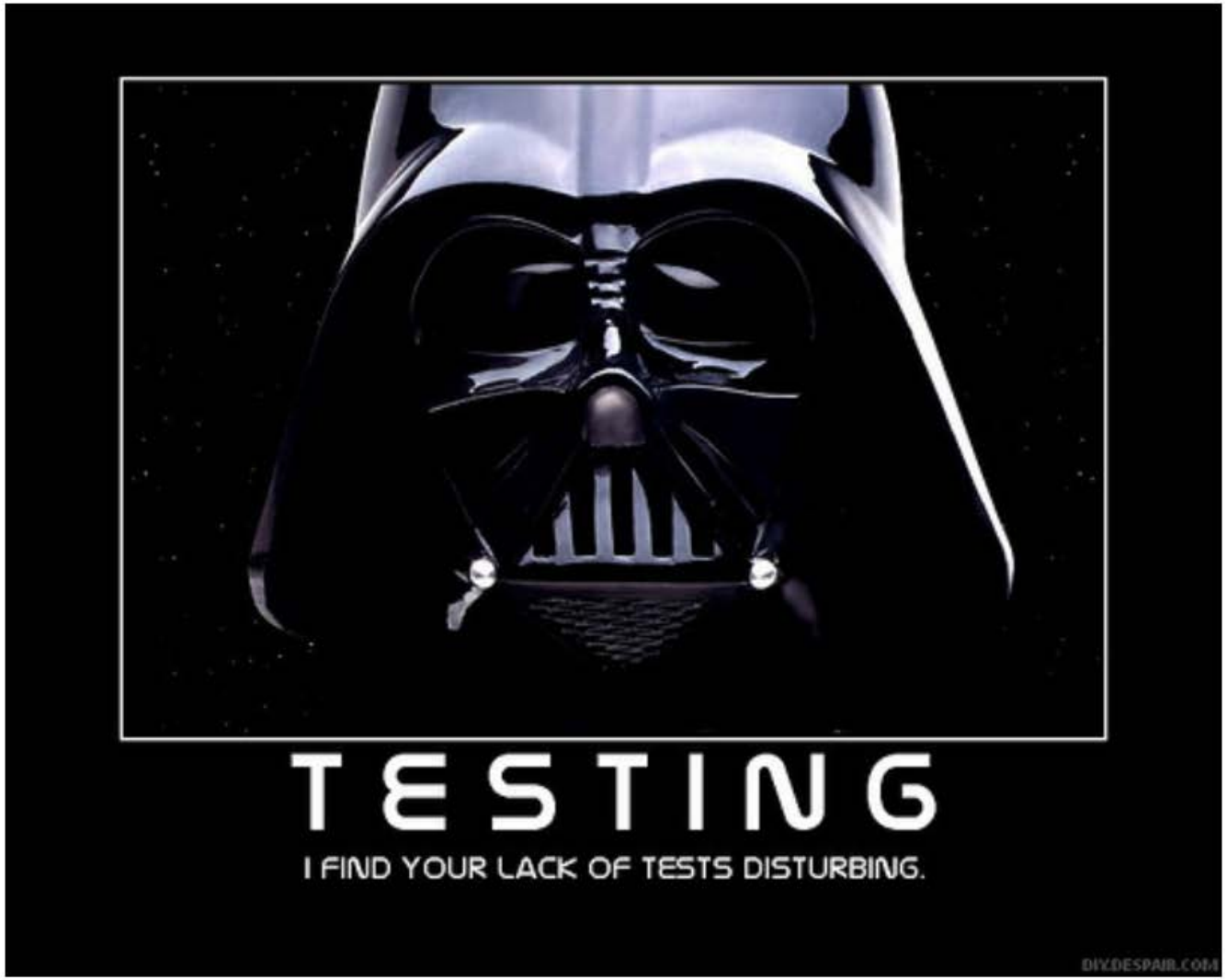
Unit Testing Tools

- Requirements
  - A driver: something to simulate usage of the unit being tested
  - Data stubs: test data the the unit requires
  - Asserts: check that the unit behaves as expected
  - Automation: enable “continuous” testing

Unit Testing Tools

- Many tools ("unit testing frameworks") available to help with these requirements:
- C/C++: CuTest, CppUnit, Boost Test, FCTX
- Fortran: fUnit, FRUIT
- MATLAB: MlUnit, MATLAB xUnit
- Python: nose
- http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

Testing...


TESTING
I FIND YOUR LACK OF TESTS DISTURBING.
DIY.DESPAIR.COM

Trust No One… not even yourself

- Unit Testing
  - Tests individual units of source code
  - A Unit is the smallest testable part of a code
  - Each Unit test should be independent from others
  - Write the unit and the test at the same time

Trust No One… not even yourself

- Regression Testing
  - Example applications, unit tests, benchmark problems
  - Catches unintended consequences of revisions
  - Design of a suite of regressions tests is an art
  - Consider Code and Feature Coverage
  - Test early and test often
  - Automate testing

Verification: Finding the Unknown Unknowns

- Solution Verification: Estimating Numerical Error
  - Discretization Error
  - Iterative Error
  - Round-off Error
- These errors cannot be avoided, but can (and must!) be quantified

Verification: Finding the Unknown Unknowns

- Code Verification
  - Software bugs
  - Numerical algorithm weaknesses
  - Model implementation mistakes
- Codes cannot practically be proven error-free, but can be proven to have errors. So we try our best to do the latter…

Example: Hypersonic Flow App

Math, Software Components

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

Approximate LoC

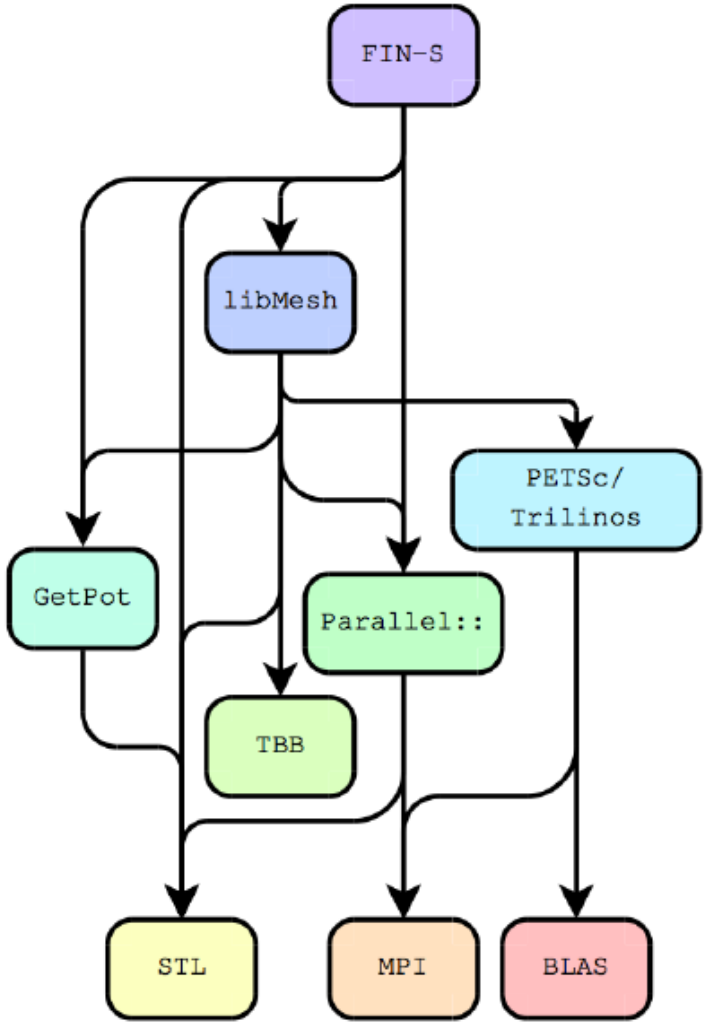
`DPLR`	46,000
Coupling	6,000
Ablation	6,000
Radiation	6,000
`FIN-S`	6,000
`libMesh`	54,000
`PETSc`	170,000
Contrib	56,000

Code Reuse

Examples

- MPI, BLAS
- Aztec, MUMPS, PETSc
- deal.II, FEniCS, libMesh

- Don't reinvent the wheel unnecessarily!
- Time spent rewriting something old is time that could have been spent writing something new.
- More eyes == fewer bugs
- Extend existing capabilities where possible.



Unit Tests

- Testing 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
- Revise Software => Rerun Tests

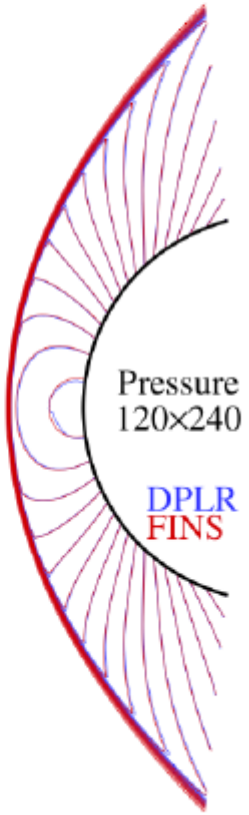
Verification Benchmark Problems

- Choosing Test Problems
- Known solutions
  - Exact solution to discretized problem
  - Limit solution of continuous problem
  - Known quantities of interest
- Known asymptotic convergence rates
- Known residuals

Code to Code Comparisons

"Lumped" Verification

- Differing results from:
  - Different models
  - Different formulations
  - Different discretizations



Hierarchic Models

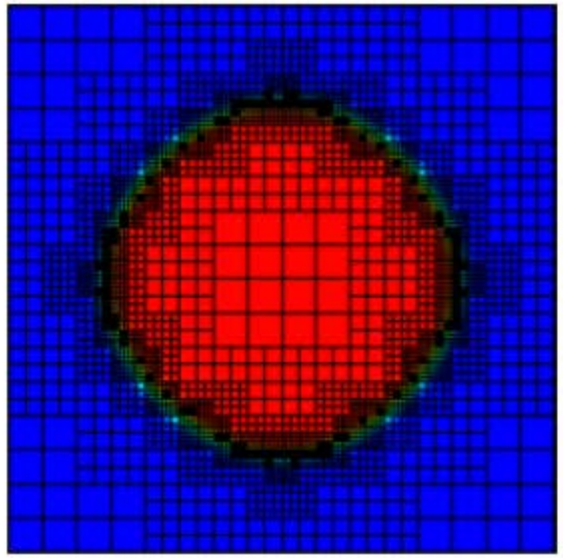
Per-Operator Tesing

- Coefficients selectively “turn off” parts of equations
- Allows easier construction of analytic solutions
- Assists with “narrowing down” other bugs

Model Simplification

- Model linearity can be tested in solver
- Reduce complex physics to simple physics case
- Code-to-code testing

Symmetry Tests



Symmetry In, Symmetry Out

- Mirror, radial symmetries
- Beware unstable solution modes!

Jacobian Verification

Jacobian Verification

Test Analytic vs. Numeric Jacobians

- Relative error in matrix norm
- If match isn't within tolerance, either:
  - The discretization or floating point error has overwhelmed the finite differenced Jacobian
    - Unlikely for good choices of finite difference perturbations
    - Can be investigated
  - The residual is non-differentiable at that iterate
    - Can be checked analytically
  - The Jacobian calculation is wrong
  - The residual calculation is wrong

A Priori Asymptotic Convergence Rates

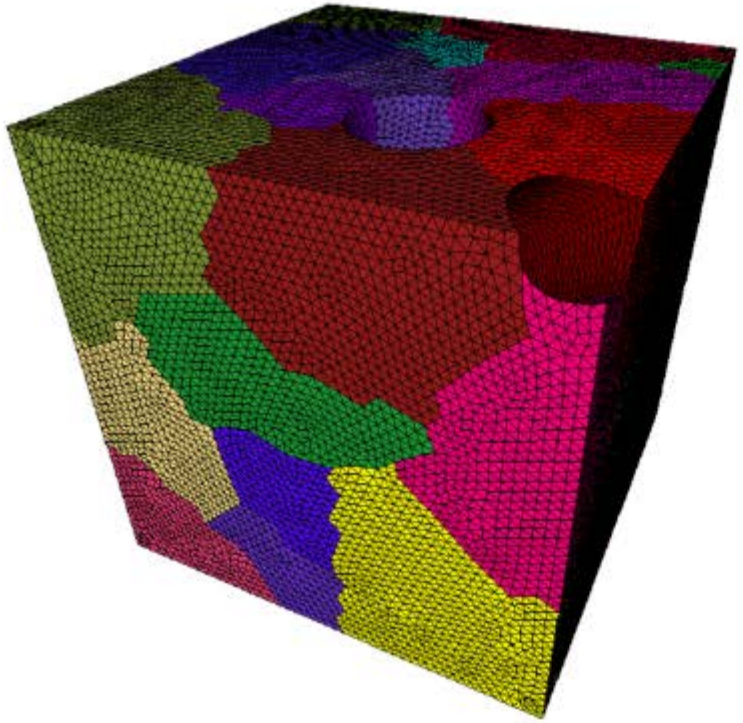
Pros

- Applicable to wide ranges of problems
- Does not require exact, or even manufactured, solution
- Part of solution verification, not just code verification

Cons

- Does not verify physics
- Requires asymptotic assumption
- Part of solution verification, not just code verification

Verification Never Ends



New Problems

- New Optimizations
- New Features
- New Combinations
- New Physics
- New Data
- New Requirements

A test suite that is never run is useless

- Automate testing
  - make check
  - svn hooks
  - crontab
  - Continuous Integration Server
- Standard procedure
  - Grab latest source code
  - Verify fresh configuration and compile
  - Verify solutions via regression tests
  - References solutions (and changelog) included in repository

Example output

```
./rtest.sh
[...]
------------------------------------------------------------
PECOS 1D Ablation Model: Version = 0.20
Build Date   = 2009-05-02 11:59
Build Host   = sqa.ices.utexas.edu
Build Arch   = x86_64
Build Rev    = r2650
Build Status = current
------------------------------------------------------------
[...]
Verifying solution differences using a tolerance of: 1e-08

CN           => Identical
O            => Identical
CO           => Identical
Recession    => Identical
Temperature  => Identical
NO           => Identical
C3           => Within tolerance (3.60731569E-19,3.62386249E-19)

Thank you, drive thru...

----------------------------------------------------------------------
(rtest): PASSED: Test 1 (graphite/10species)
----------------------------------------------------------------------
```

Verification is iterative and ongoing

At least two possibilities:

- The observed phenomenon is real and must be understood
- The observed phenomenon is caused by a bug

Debugging Philosophy

- Be skeptical of odd results—assume a bug and do everything to can to test for it
- View debugging as an opportunity to add new tests
- Once bug is found, add test that caught it

The Bug and Updated Test Suite

Broken Code

```
for (unsigned int ii=1; ii<N; ++ii)
  {
    const double yrat = p->expData->yPlus(ii)/p->expData->yPlus(ii-1);
    vp[ii] = modelRoutine<Mod>( *(p->physParams), yrat*zp, p->expData->deltaPlus() );
  }
```

Correct Code

```
for (unsigned int ii=1; ii<N; ++ii)
  {
    const double yrat = p->expData->yPlus(ii)/p->expData->yPlus(0);
    vp[ii] = modelRoutine<Mod>( *(p->physParams), yrat*zp, p->expData->deltaPlus() );
  }
```

New Test

- Created unit test that checks output of function against saved output
- Not perfect, but better than nothing

Generally Accepted Recommendations

General Recommendations

- Show other people your code early and often
- Reuse good software and evaluate multiple options
- Avoid platform dependent files (endianness)
- Always consider performance versus portability
- Prefer OSS solutions if possible
- Backup everything and anything

Don't reinvent the wheel

- Don't write your own input parser
- Don't write your own solver unless necessary
- Don't write anything until you evaluate options

Learn as much Unix as you can tolerate

Unix: A Culture in Itself

"Two of the most famous products of Berkeley are LSD and Unix. I don't think that this is a coincidence."
(Anonymous quote from The UNIX-HATERS Handbook.)

Highly Opinionated Recommendations

Consistent tools from desktop to supercomputer

- Learn a console-based text edition (vi(m) / emacs)
- Learn a unix shell
- Learn LaTeX
- Learn a scripting language (python / perl)
- man (-k) is your friend

Understand:

- compile / link cycle
- static versus dynamic libraries
- processor optimization
- storage hierarchy (local versus network disk)
- environment variables ($PATH and $LD_LIBRARY_PATH)

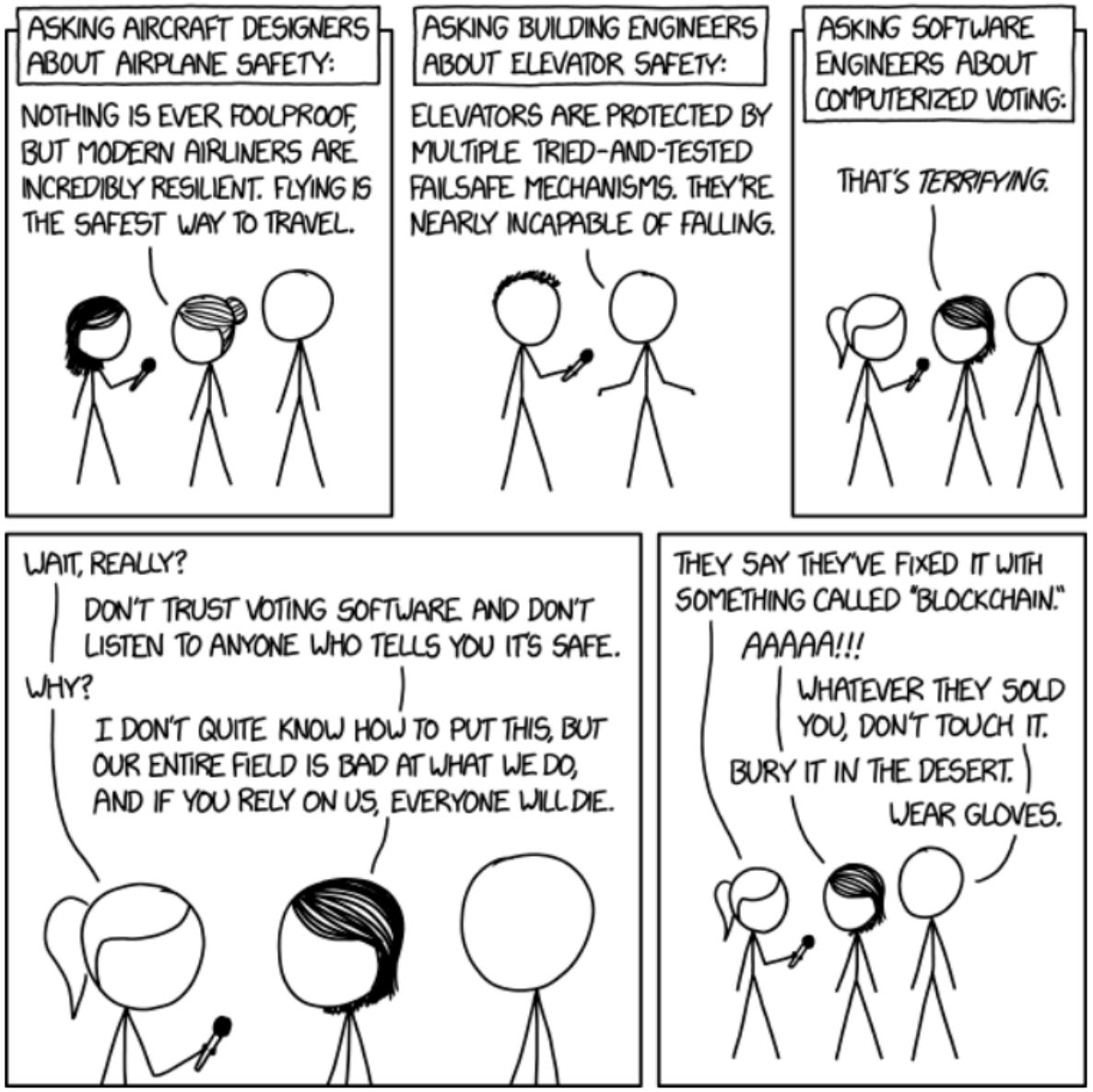
More Highly Opinionated Recommendations

More things to understand

- Actually learn how to use the compiler
- Learn about the hardware (memory, interconnect, disk)
- Learn about debugging and profiling
- Always have a restart capability

- Soapbox
  - Use text-based formats and version control whenever possible
  - Keep as much of your life as possible out of email
  - Check out OpenHPC!; Be NICE to SysAdmins

Relevant XKCD



VOTE EARLY, VOTE OFTEN —

Frozen machines, 243-percent turnout, and other woes in Georgia voting

A federal lawsuit alleges significant problems with voting in the Peach state.

JONATHAN M. GITLIN - 8/8/2018, 11:42 AM

John McAfee ✓
@officialmcafee

Follow

For all you naysayers who claim that "nothing is unhackable" & who don't believe that my Bitfi wallet is truly the world's first unhackable device, a $100,000 bounty goes to anyone who can hack it. Money talks, bullshit walks. Details on Bitfi.com

12:12 PM - 24 Jul 2018