Testing of HPC Scientific Software
Part 1

Presented at
Better Scientific Software tutorial

ISC18, Frankfurt, Germany

Anshu Dubey
Computer Scientist, Argonne National Laboratory
University of Chicago

License, citation and acknowledgements

License and Citation

- Requested citation: Anshu Dubey, Testing of HPC Scientific Software: Part 1, tutorial, in ISC High Performance 2018: DOI: 10.6084/m9.figshare.6453017.

Acknowledgements

- **Alicia Klinvex contributed many slides**
- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Argonne National Laboratory, which is managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357

Verification

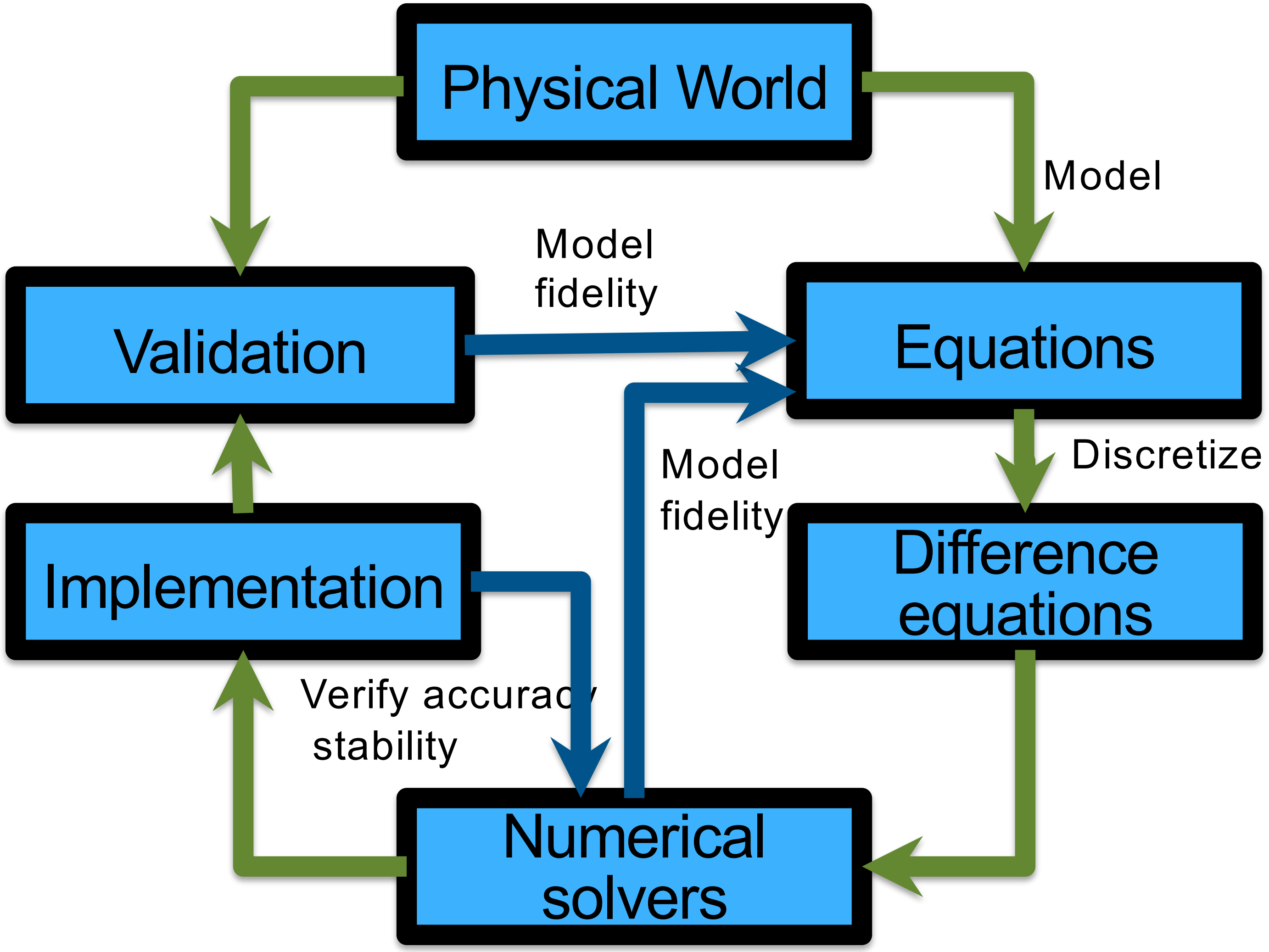
Definitions

Why is testing important?

Verification

- Code verification uses tests
  - It is much more than a collection of tests
- It is the holistic process through which you ensure that
  - Your implementation shows expected behavior,
  - Your implementation is consistent with your model,
  - Science you are trying to do with the code can be done.

Simplified schematic of science through computation



This is for simulations, but the philosophy applies to other computations too.

Many stages in the lifecycle have components that may themselves be under research => need modifications

Stages and types of verification

- During initial code development
  – Accuracy and stability
  – Matching the algorithm to the model
  – Interoperability of algorithms
- In later stages
  – While adding new major capabilities or modifying existing capabilities
  – Ongoing maintenance
  – Preparing for production

Stages and types of verification

- If refactoring
  - Ensuring that behavior remains consistent and expected
- All stages have a mix of automation and human-intervention

Note that the stages apply to the whole code as well as its components

Benefits of testing

- Promotes high-quality software that delivers correct results and improves confidence
- Increases quality and speed of development, reducing development and maintenance costs
- Maintains portability to a variety of systems and compilers
- Helps in refactoring
  - Avoid introducing new errors when adding new features
  - Avoid reintroducing old errors

How common are bugs?

Programs do not acquire bugs as people acquire germs, by hanging around other buggy programs.
Programmers must insert them.
- Harlan Mills

- Bugs per 1000 lines of code (KLOC)
- Industry average for delivered software
  - 1-25 errors
- Microsoft Applications Division
  - 10-20 defects during in-house testing
  - 0.5 in released product

Code Complete (Steven McConnell)

Why testing is important:
the protein structures of Geoffrey Chang

- Some inherited code flipped two columns of data, inverting an electron-density map
- Resulted in an incorrect protein structure
- Retracted 5 publications
  - One was cited 364 times
- Many papers and grant applications conflicting with his results were rejected

He found and reported the error himself

Why testing is important: the 40 second flight of the Ariane 5

- Ariane 5: a European orbital launch vehicle meant to lift 20 tons into low Earth orbit
- Initial rocket went off course, started to disintegrate, then self-destructed less than a minute after launch
- Seven variables were at risk of leading to an Operand Error (due to conversion of floating point to integer)
  - Four were protected
- Investigation concluded insufficient test coverage as one of the causes for this accident
- Resulted in a loss of $370,000,000.

Why testing is important: the Therac-25 accidents

- Therac-25: a computer-controlled radiation therapy machine
- Minimal software testing
- Race condition in the code went undetected
- Unlucky patients were struck with approximately 100 times the intended dose of radiation, ~ 15,000 rads
- Error code indicated that no dose of radiation was given, so operator instructed machine to proceed
- Recalled after six accidents resulting in death and serious injuries

Test Definitions

- Unit tests
  - Test individual functions or classes
- Integration tests
  - Test interaction, build complex hierarchy
- System level tests
  - At the user interaction level
- Restart tests
  - Code starts transparently from a checkpoint
- Regression (no-change) tests
  - Compare current observable output to a gold standard
- Performance tests
  - Focus on the runtime and resource utilization

Test Development

- Development of tests and diagnostics goes hand-in-hand with code development
  - Non-trivial to devise good tests, but extremely important
  - Compare against simpler analytical or semi-analytical solutions
    - They can also form a basis for unit testing
- In addition to testing for “correct” behavior, also test for stability, convergence, or other such desirable characteristics
- Many of these tests go into the test-suite

Use of test harnesses

- Essential for large code
  - Set up and run tests
  - Evaluate test results
- Easy to execute a logical subset of tests
  - Pre-push
  - Nightly
- Automation of test harness is critical for
  - Long-running test suites
  - Projects that support many platforms

Jenkins
C-dash
Custom
(FlashTest)

Policies on testing practices

- Must have consistent policy on dealing with failed tests
  - Issue tracking
    - How quickly does it need to be fixed?
    - Who is responsible for fixing it?
- Someone needs to be in charge of watching the test suite

Policies on testing practices

- When refactoring or adding new features, run a regression suite before checkin
  - Be sure to add new regression tests for the new features
- Require a code review before releasing test suite
  - Another person may spot issues you didn't
  - Incredibly cost-effective

Maintenance of a test suite

- Testing regime is only useful if it is
  - Maintained
    - Tests and benchmarks periodically updated
  - Monitored regularly
    - Can be automated
  - Has rapid response to failure
    - Tests should pass most of the time

Agenda

Time	Topic	Speaker
2:00pm-2:30pm	Why Effective Software Practices are Essential for CSE Projects	Anshu Dubey, ANL
2:30pm-3:00pm	Introduction to Software Licensing	Michael A. Heroux, SNL
3:00am-3:30pm	Better (small) Scientific Software Teams	Michael A. Heroux, SNL
3:30am-4:00pm	Improving Reproducibility Through Better Software Practices	Michael A. Heroux, SNL
4:00pm-4:30pm	*Break*	
4:30pm-5:00pm	Testing HPC Scientific Software – Part 1	Anshu Dubey, ANL
5:00pm-5:30pm	Testing HPC Scientific Software – Part 2	Anshu Dubey, ANL
5:30pm-6:00pm	Code Coverage Hands-on and CI Demo	Anshu Dubey, ANL