Verification

Presented at
Better Scientific Software tutorial

SC17, Denver, Colorado

Anshu Dubey
Argonne National Laboratory

EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

IDEAS productivity

License, citation and acknowledgements

License and Citation

- Requested citation: Anshu Dubey, Verification, tutorial, in SC '17: International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, Colorado, 2017. DOI: 10.6084/m9.figshare.5593345.

Acknowledgements

- 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.

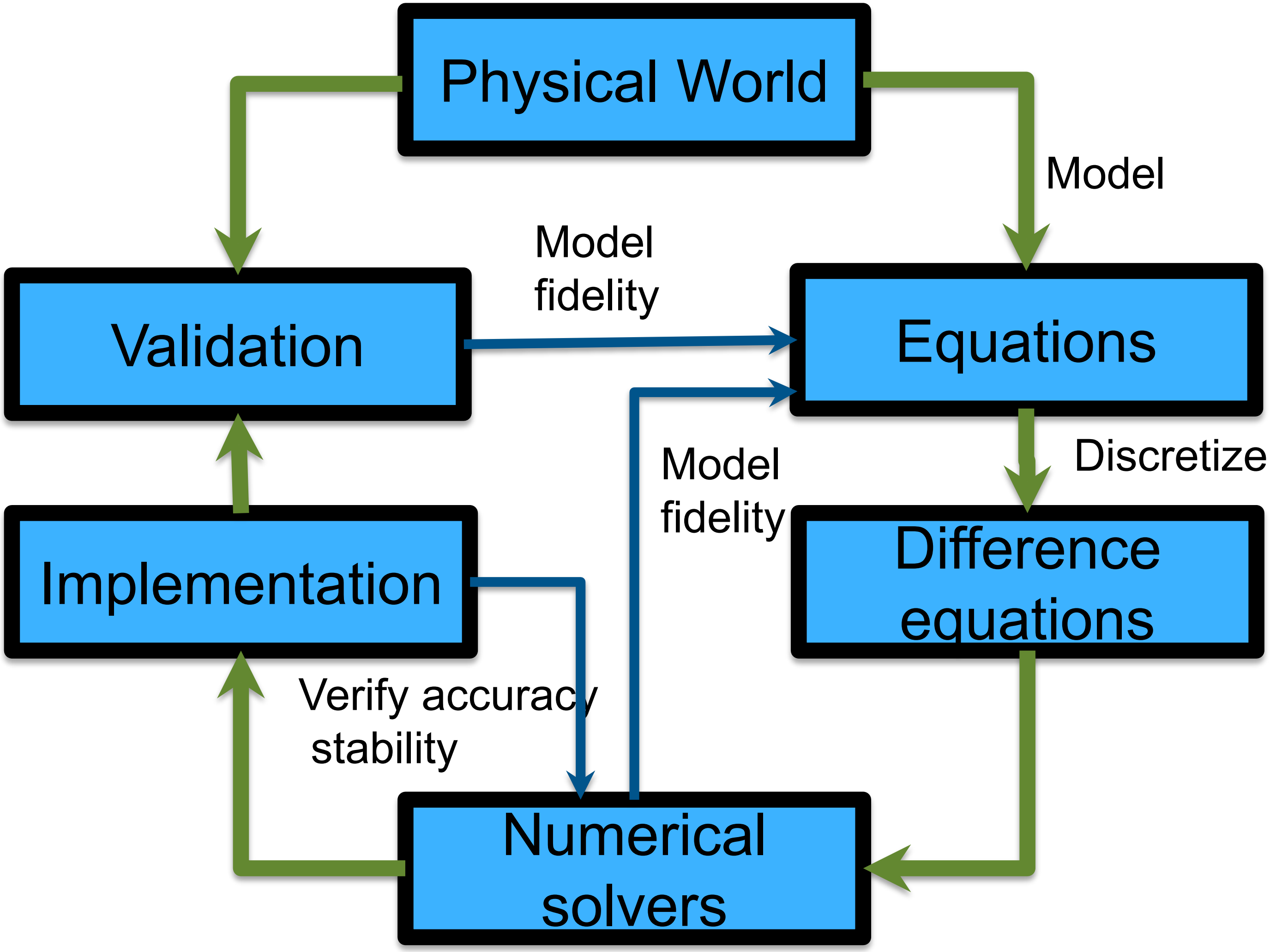
Scientific Software Verification

Challenges specific to scientific software

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

CSE verification challenges

- Floating point issues
  - Different results
    - On different platforms and runs
    - Ill-conditioning can magnify these small differences
      - Final solution may be different
      - Number of iterations may be different
- Unit testing
  - Sometimes producing meaningful testable behavior too dependent upon other parts of the code
- Definitions don’t always fit

CSE verification challenges

- Integration testing may have hierarchy too
- Particularly true of codes that allow composability in their configuration
- Codes may incorporate some legacy components
  - Its own set of challenges
    - No existing tests of any granularities
- Examples – multiphysics application codes that support multiple domains

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

Test Development

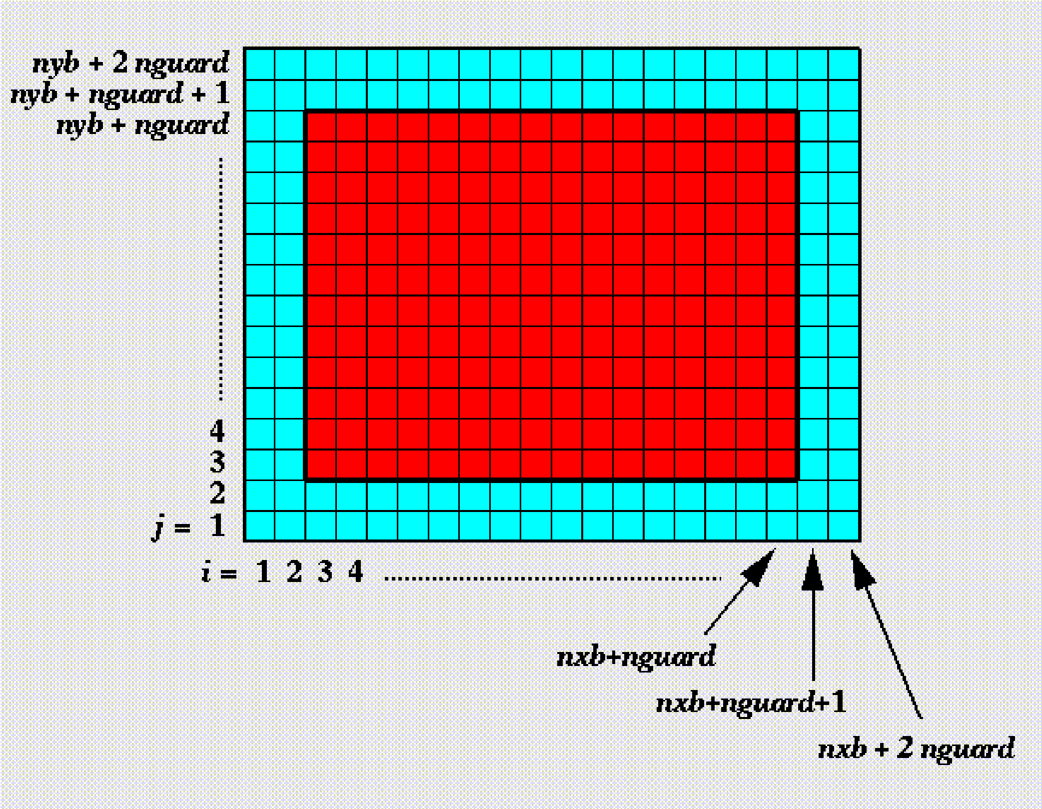
- Development of tests and diagnostics goes hand-in-hand with code development
  - Non-trivial to devise good tests, but extremely important
  - Compare against simpler analytical or semi-analytical solutions
    - They can also form a basis for unit testing
- In addition to testing for “correct” behavior,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

Example from Flash

- Grid ghost cell fill
    - Use some function to initialize domain
    - Two variables, in one only interior cells initialized, in the other ghost cells also initialized
    - Run ghost cell fill on the first variable – now both should be identical within known tolerance
  - Use redundant mechanisms

Against manufactured solution

- Verification of guard cell fill
- Use two variables A & B
- Initialize A including guard cells and B excluding them
- Apply guard cell fill to B



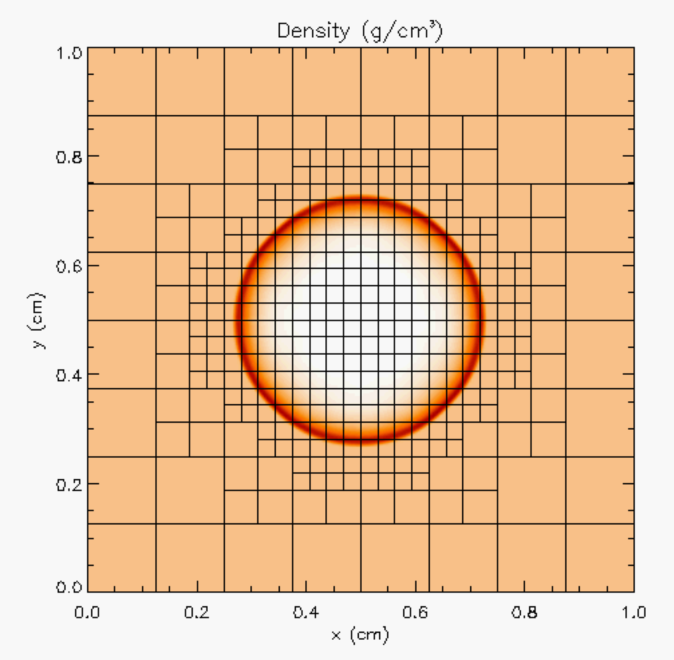
Example from Flash

- Eos
  - Use initial conditions from a known problem
  - Apply eos in two different modes – at the end all variables should be consistent within tolerance

- Hydrodynamics
  - Sedov blast problem has a known analytical solution
  - Runs with UG and AMR

Against analytical solution

- Sedov blast wave
- High pressure at the center
- Shock moves out spherically
- FLASH with AMR and hydro
- Known analytical solution



Though it exercises both mesh, hydro and eos, if mesh and eos are verified first, then this test verifies hydro

Building confidence

- First two unit tests are stand-alone
- The third test depends on Grid and Eos
  - Not all of Grid functionality it uses is unit tested
    - Flux correction in AMR
- If Grid and Eos tests passed and Hydro failed
  - If UG version failed then fault is in hydro
  - If UG passed and AMR failed the fault is likely in flux correction

Agenda

Tutorial evaluation form: http://bit.ly/sc17-eval

Time	Topic	Speaker
8:30am-8:45am	Why effective software practices are essential for CSE projects	David E. Bernholdt, ORNL
8:45am-9:15am	Introduction to software licensing	David E. Bernholdt, ORNL
9:15am-9:45am	Better (small) scientific software teams	Michael A. Heroux, SNL
9:45am-10:00am	Improving Reproducibility Through Better Software Practices	Michael A. Heroux, SNL
10:00am-10:30am	*Break*	
10:30am-10:45am	Testing of HPC Scientific Software: Introduction	Alicia M. Klinvex, SNL
10:45am-11:15am	Verification	Anshu Dubey, ANL
11:15am-11:45am	Evaluating project testing needs	Anshu Dubey, ANL
11:45am-12:00pm	Code coverage demo and CI demo	Alicia M. Klinvex, SNL