COLLEGE OF ENGINEERING
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES
UNIVERSITY OF MICHIGAN

Lecture 22
Testing, Verification, Validation

Prof. Brendan Kochunas

11/20/2019

NERS 590-004

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Outline

- Motivation
- Review of Testing
- Definitions
- Code Verification
- Solution Verification
- Validation & Prediction

Why should I care about V&V?

- How do you know someone else's code is correct?
- How does someone else know your code is correct?
- Correct in what sense?
  - Does it do what its supposed to?
  - Does it represent reality?
- How can we develop tests that are meaningfully providing evidence of verification and validation?

Today’s Learning Objectives

- Understand difference of Verification and Validation
- Understand testing strategies for verification and validation testing

Further Reading

- Oberkampf, W., & Roy, C. (2010). *Verification and Validation in Scientific Computing*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511760396
    - This book is 665 pages!

- https://doi.org/10.1017/CBO9780511760396
- https://search.lib.umich.edu/catalog/record/015554090

- Specifically, chapters: 2, 5, 6, 7, 12

Ross's Taxonomy of Testing

A Taxonomy of Testing

- Testing is the backbone of software quality assurance (SQA).
- Types of testing
    - *Unit Testing* – Test individual units of program *in isolation*
        - Should run very fast: < 1 second (a couple seconds is ok)
    - *Integral Testing* – Testing program components together
        - Should run fast: < 1 minute (a couple minutes is ok)
    - *Regression Testing* – Test whole program for changes in program output
        - Should run fast: < 1 minute (a couple minutes is ok)
    - *Verification Testing* – Test that you are “doing things right”
        - Can happen at unit or integral or regression level. Comparison analytic solutions or manufactured solutions.
    - *Validation Testing* – Whole program testing “doing the right thing”; simulating reality, comparison to experiment.
        - May be long running: minutes to hours
    - *Memory Testing* – Expensive testing that does detailed memory simulations to detect errors (valgrind)
    - *Coverage Testing* – Figure out how much of your source code is actually covered by testing
    - *Portability Testing* – test on different platforms and with different compilers
- Other types of testing exist

Testing Layers

Correctness Testing

Nightly Testing
Secondary Tested (ST)
CATEGORIES [BASIC CONTINUOUS NIGHTLY]
(includes all testing*)

Post-Push CI Testing
Secondary Tested (ST)
CATEGORIES [BASIC CONTINUOUS]
(includes more regression testing)

Pre-Push CI Testing
Primary Tested (PT)
CATEGORIES [BASIC]
(unit tests & some regression tests)

*Additional Categories:
Heavy or Weekly

Coverage Testing

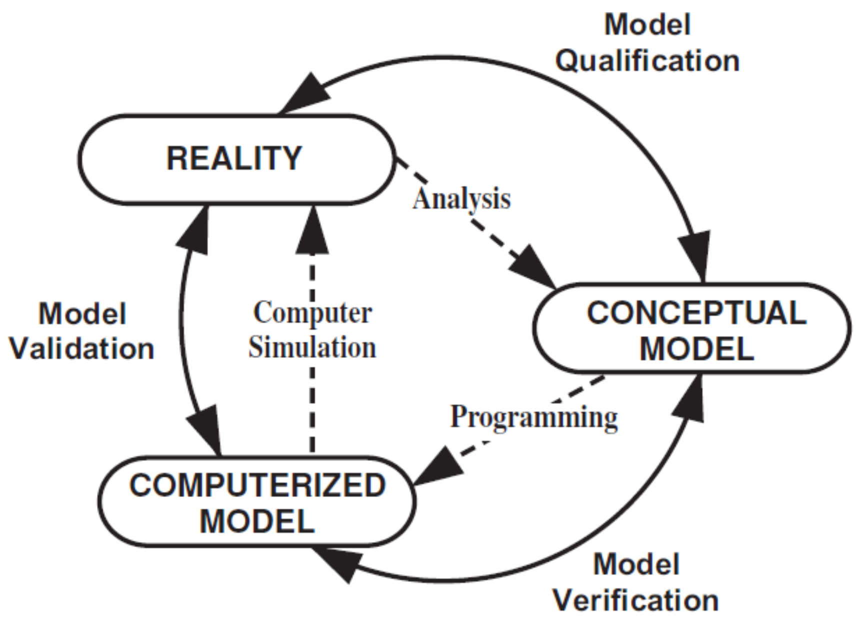
Memory (Valgrind) Testing

Terminology

What is V&V?

Early Definitions of Verification and Validation

- *Model verification*: substantiation that a computerized model represents a conceptual model **within specified limits of accuracy**.
- *Model validation*: substantiation that a computerized model within its domain of applicability possesses a **satisfactory range of accuracy** consistent with the **intended application** of the model.



Other Definitions

IEEE

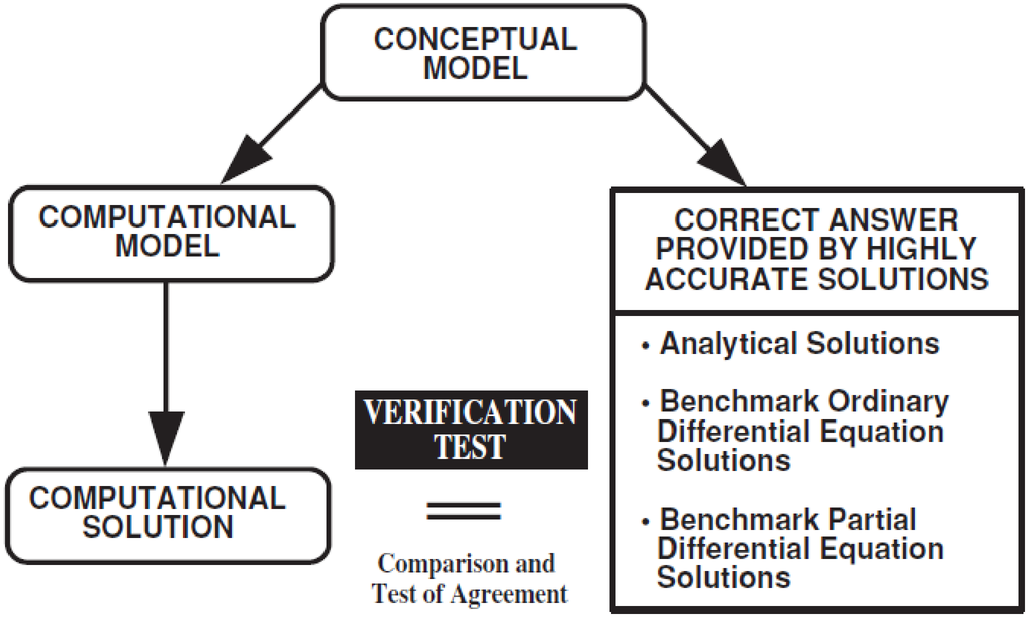
- *Verification:* the process of evaluating the products of a software development phase to provide assurance that they meet the requirements defined for them by the previous phase.
- *Validation*: the process of testing a computer program and evaluating the results to ensure compliance with specific requirements.

Department of Defense

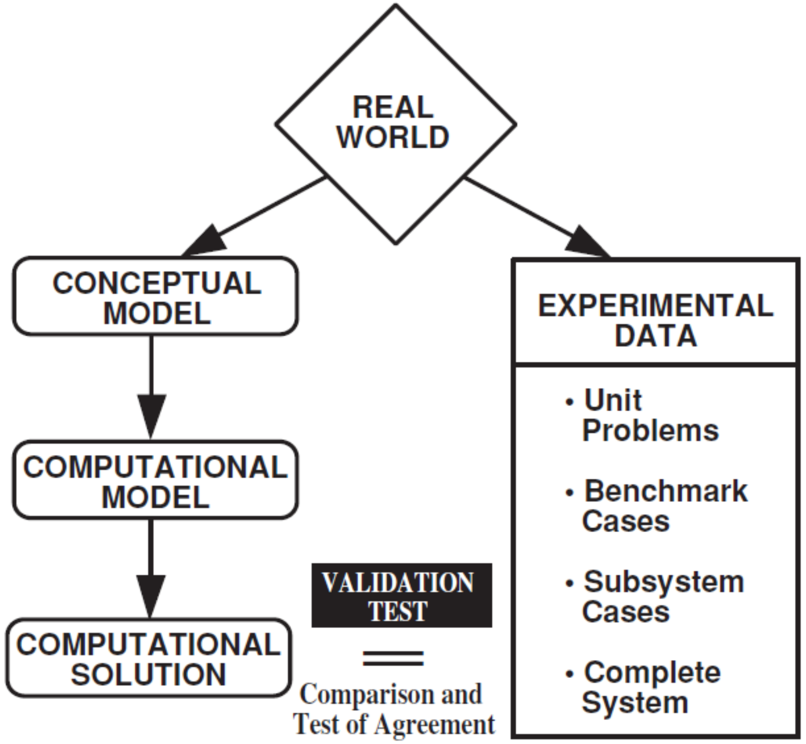
- *Verification*: the process of determining that a model implementation **accurately represents the developer's concep**tual description of the model.
- *Validation*: the process of determining the degree to which a model is an **accurate representation of the real world** from the **perspective of the intended use**s of the model.

American Institute of Aeronautics & Astronautrics

Verification

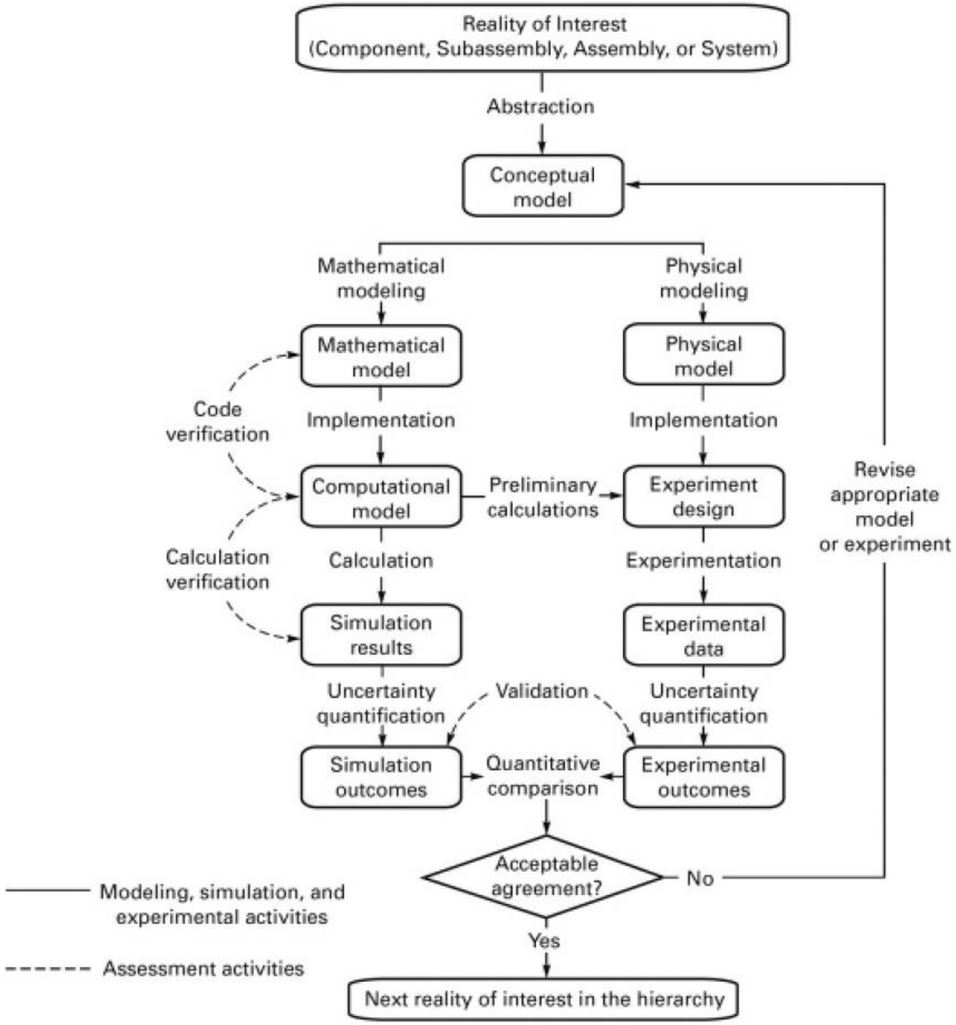


Validation



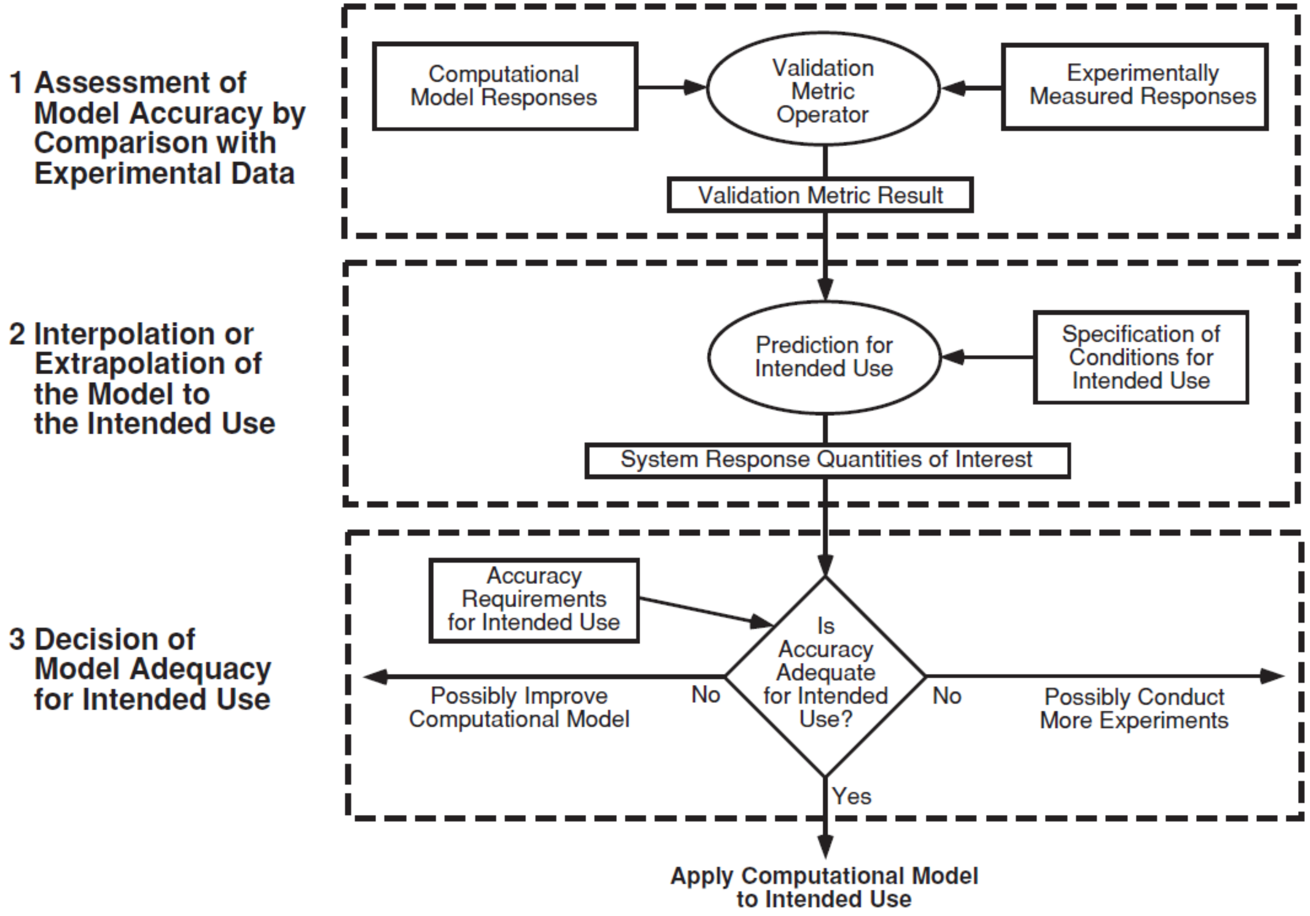
ASME

- *Code verification*: the process of determining that the numerical algorithms are correctly implemented in the computer code and of identifying errors in the software.
- *Solution verification*: the process of determining the **correctness of the input data**, the numerical accuracy of the solution obtained, and the **correctness of the output data** for a particular simulation.



Aspects of Validation

- Quantification of the accuracy of the computational model results by comparing the computed system response quantities (SRQs) of interest with experimentally measured SRQs
- Use of the computational model to make predictions, in the sense of interpolation or extrapolation of the model, for conditions corresponding to the model's domain of intended use.
- Determination of whether the estimated accuracy of the computational model results satisfies the accuracy requirements specified for the SRQs of interest.



Code Verification

How do you prove that your program is a faithful representation of the original mathematical model?

Selection of Criteria/Metrics

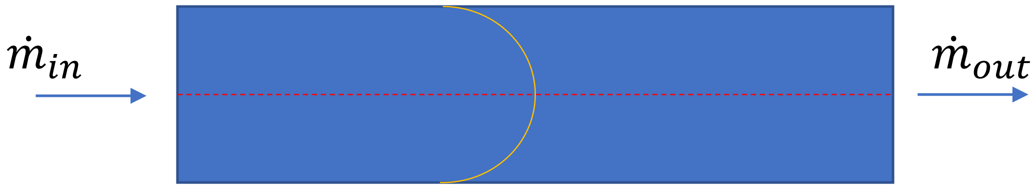
- Choosing the solution is a natural, but has practical issues.
  - Need a good reference solution or exact solution
  - Mapping of the numerical solution to the reference can be burdensome
  - Considering the full solution may be overwhelming w.r.t the amount data
- Solution: Norms!
  - L1 norms are good for problems where discontinuities or singularities might exist. Typically applied for discretization errors.
  - L2 norms also commonly used for discretization errors.
  - L-infinity will be the most sensitive
- Additionally consider, quantities of interest for the application

Types of Code Verification Tests

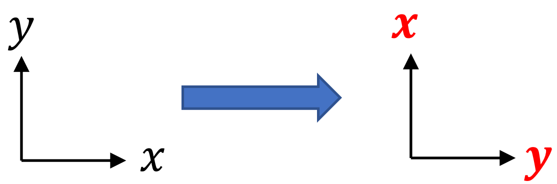
- Simple Tests
- Code-to-Code Verification
- Discretization Error Quantification
- Convergence Tests
- Order-of-Accuracy Tests

Simple Test Example

- Symmetry
    - Define a problem with geometry and boundary conditions symmetric about a plane
    - Can also look at periodic problems



- Rotational Invariance
    - Similar to the above except change coordinates

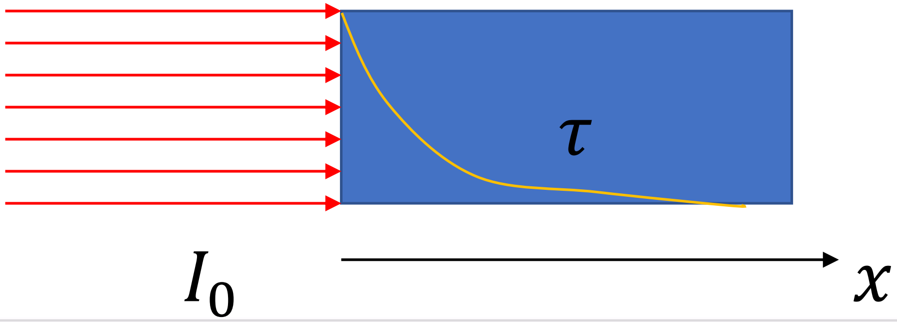


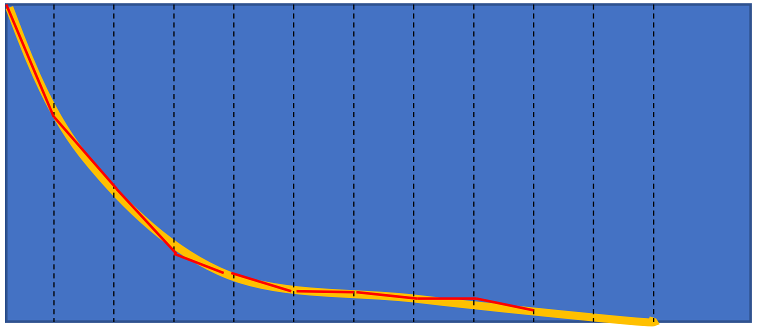
- Conservation (perform global integration)
    - of energy in heat transfer
    - of mass or momentum in fluid flow

$$\dot{m}_{in} = \dot{m}_{out}$$

Discretization Error Tests

- Compare the numerical solution to an exact solution
- Quantitative assessment of code output using a single mesh
- Example: beam attenuation





$$I(x) = I_0 \exp(-\tau x)$$

Convergence Tests

Mesh Refinement

- Essentially testing the fundamental theorem of calculus
- As you refine your discretization does the error reduce towards zero?

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Iterative Methods

- Fixed point method (lecture 7)

$$x^{(\ell+1)} = \mathbf{F}x^{(\ell)} + \mathbf{c}$$

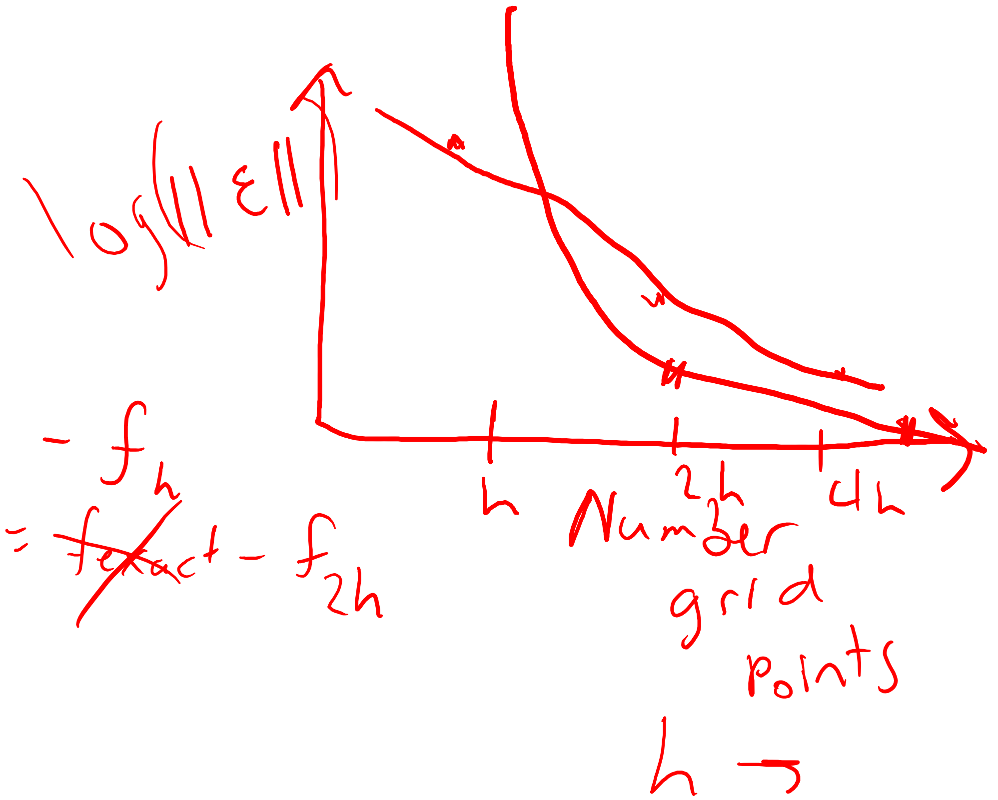
- Has a rate of convergence $\rho(\mathbf{F})$ the spectral radius.
- Does your implementation behave like:

$$\frac{\left\|\epsilon^{(\ell+1)}\right\|}{\left\|\epsilon^{(\ell)}\right\|} \approx \rho(\mathbf{F}) = \lim_{\ell \to \infty} \frac{\left\|\epsilon^{(\ell+1)}\right\|}{\left\|\epsilon^{(\ell)}\right\|}$$

Order-of-Accuracy Tests

- Run two calculations (or a series) with a uniform grid refinement factor, typically 2
  - Error in solution on grid h is ϵ_h $= f_{exact} - f_h$
  - Error in solution on refined grid $2h$ is ϵ_{2h} $= f_{exact} - f_{2h}$
- Computing Observed Order of Accuracy

$$\hat{p} = \frac{\ln\left(\frac{\epsilon_{2h}}{\epsilon_h}\right)}{\ln(2)}$$



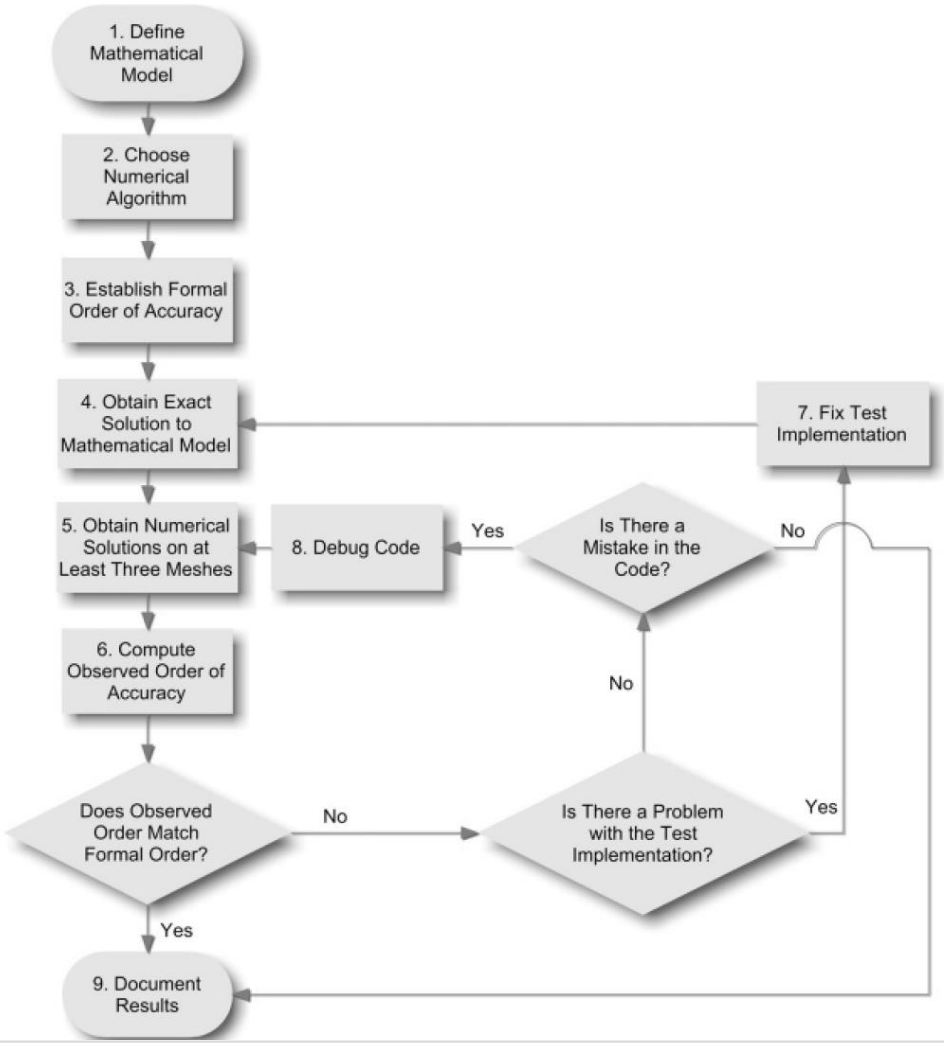
Formal Order of Accuracy

- Establish formal order of accuracy
  - Example: Finite Difference

$$f(x_i + \Delta x) = f(x_i) + \frac{df(x_i)}{dx}\frac{\Delta x}{1!} + \frac{df(x_i)}{dx}\frac{\Delta x^2}{2!} + \cdots$$

$$\frac{df(x_i)}{dx} \approx \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x} + \boldsymbol{O}\left(\boldsymbol{\Delta x^2}\right)$$

- Does your observed order of accuracy agree with theory?



Solution Verification

Is the given numerical approximation of a mathematical model sufficiently accurate for its intended use?

Elements of Solution Verification

- Verification of Input Data
  - Examples: boundary conditions, coefficients, geometry approximations
- Verification of Post-Processing Tools
  - Examples: Excel Charts—how do you show pointwise data?
- Numerical error estimation
  - Examples: round-off, statistical sampling, iterative error, discretization error

How to estimate numerical errors

Condition of Matrix

$\kappa(A_h) \to 10^7 \to 10^{-10}$

$\kappa(A_{2h}) \to 10^8 \to 10^{-9}$

$\kappa(A_{128h}) \to 10^{16} \to$ 1 digit of accuracy

F.P. Round off

$\varepsilon \to$ smallest representable floating point number

double precision $\approx 10^{-17}$

Round off $\approx \kappa \cdot \varepsilon$

How to show iterative errors?

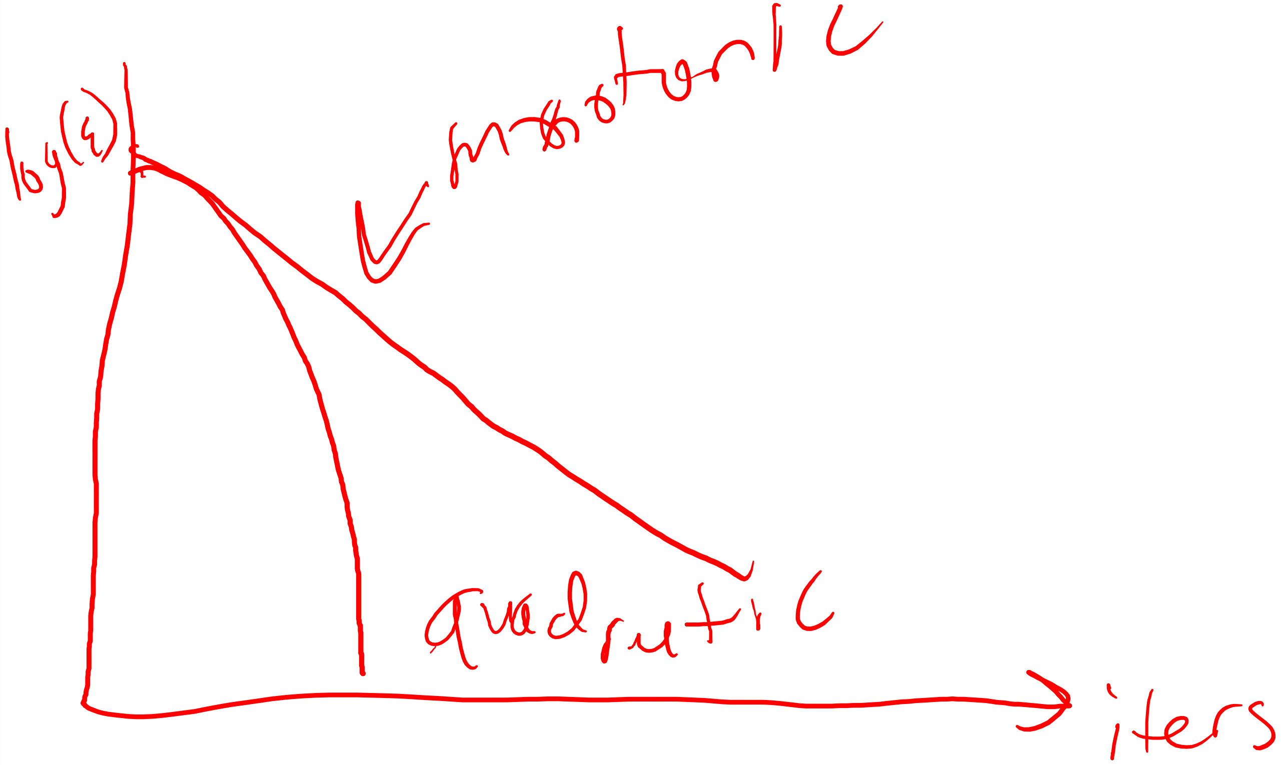
Oscillatory Errors

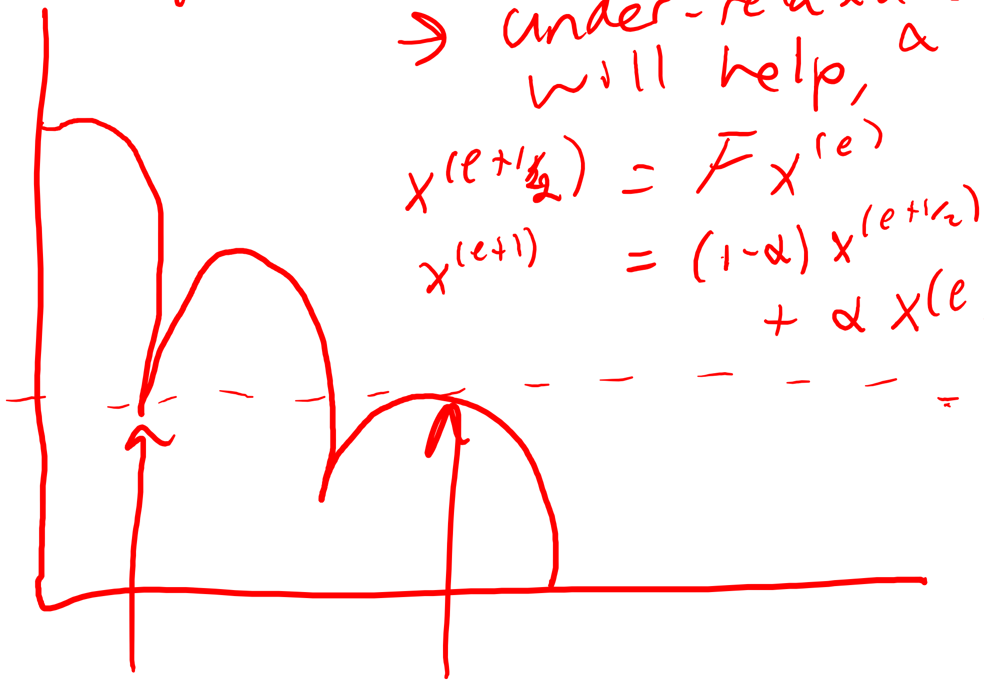
$\rho(A) \rightarrow$ complex

$\rightarrow$ under-relaxation will help, α

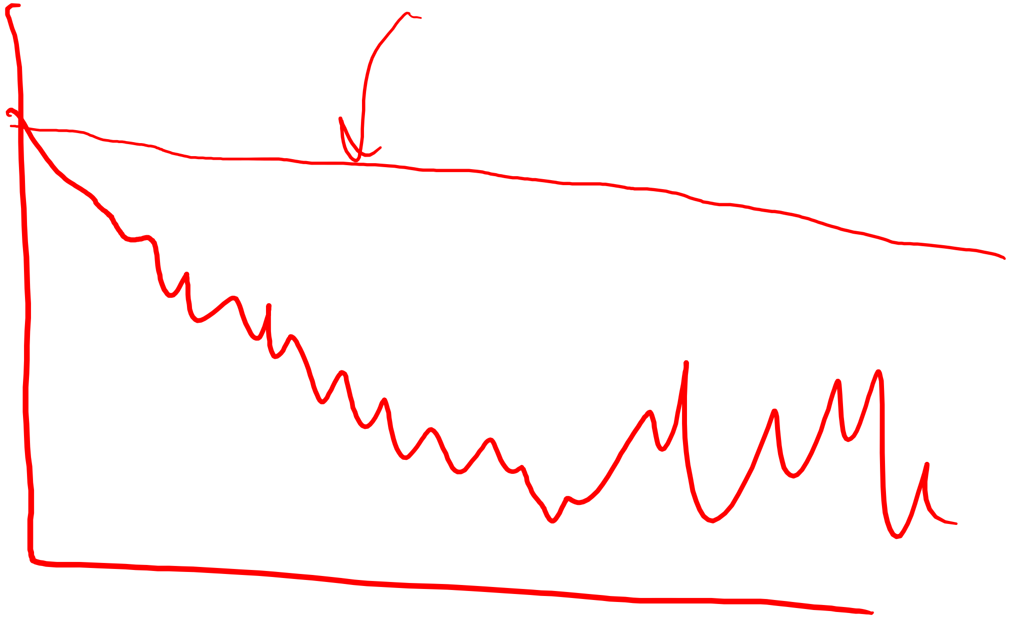
$$x^{(\ell+1/2)} = F x^{(\ell)}$$

$$x^{(\ell+1)} = (1-\alpha)\, x^{(\ell+1/2)} + \alpha\, x^{(\ell)}$$





How to show iterative errors?



$$\varepsilon_{crit} > \left| x^{(\ell+1)} - x^{(\ell)} \right|$$