Verification and Validation

Simulation and Modeling (CSCI 3010U)

Faisal Z. Qureshi

http://vclab.science.ontariotechu.ca

Verification and validation

How do we guarantee that the simulations is correct? That the results acquired through simulation are valid and that these are useful for the purposes of the task at hand?

This is an important consideration. Still little attention has been paid to this issue.

Example: flight simulators

- How do we certify a flight simulator?
- FAA specifies the criteria needed to certify a flight simulator.
- These criteria takes into account the fidelity of simluator.

Definitions

Verification: correctness of simulations, ie, given a mathematical model, the implementation is correct.

Validation: did we pick the right "mathematical model" for the task at hand?

Verification

- Software verification
- A CS problem
- Program testing is not verification

Code vs. solution verification

- Code verification
  - Program correctly implements an algorithms
- Solution verification
  - Program is correctly applied to a simulation problem

Approaches for code verification

- Use simple cases (leverage analytical solutions for comparison)
- Use multiple programs
  - If all of these agree then that is good (they might still have the same error)
  - If these do not agree then that is bad
  - Consistent results give us some confidence, but these are not proof of correctness
- Work backwards, start with a solution and identify the inputs required to reach that solution

Approaches for code verification

- Test for a range of parameter values
  - Time step
  - Grid size
  - Other parameters
- Test for the length of simulation
- Test for multiple runs for stochastic simulations
- Exploit knowledge about the problem (e.g., *energy conservation*)

Approaches for code verification

- None of these techniques guarantee that the simulation produces the correct value
- These techniques serve to increase our confidence in the results produced by the simulation
- These techniques also reduces the chance of *undetected errors* in the simulation

Validation

- Much more involved then verification
- Starts with a verified simulation

The key idea is to compare the simulation to the “real thing”

Validation challenges

- How do we measure the real system?
- Can we even perform experiments on the real system?
- How accurate are experimental results?
- Can we determine input (values) to our simulation?

Validation database

The idea is to perform a series of experiments and determine that these agree with the simulation

Validation database contains:

- a detailed description of the experiment;
- the results of the experiment;
- the results from the simulation; and
- the degree to which the two sets of results agree.

Validation experiments

- Plan your experiments
  - Large and complex experiments can be **very** costly in terms of time and money
- Cover as many cases as possible
  - Need an understanding of the “real” conditions under which the simulation will be used

Validation experiments

- Complete control over the system
  - E.g., Carrying out an experiment in a lab setting
- No control over the system
  - E.g., Weather simulation
  - It may not even be possible to observe the input parameters, or to measure these accurately
  - May have the one chance to carry out the experiment

Key takeaway: keep a detailed record of the experiment

Component-based approach

- Divide and conquer
- Identify sub-components and validate these individually
- Work your way up to entire simulation

Key assumption: if all components are valid then there is a greater likelihood that the complete simulation is accurate

Identifying relevant situations

- Identify important (say, *safety critical*) situations, and focus on these situations first
- This will allow us to formulate a test plan
  - Focus on simpler cases first

Real world vs. simulation

Say we are interested in some quantity u. Our simulation produces this quantity u_{discrete} and this quantity can also be measured in the "real" world: u_{nature}. We are then interested in measure $\Delta = u_{\mathrm{nature}} - u_{\mathrm{discrete}}$.

Now in order to measure the quantity u, we need to carry out some experiments. So what we are really getting is $u_{\mathrm{experiment}}$, which may not be the same as u_{nature}.

Real world vs. simulation (contd.)

In this setup, we often end up measuring $\Delta = (u_{\text{nature}} - u_{\text{experiment}}) + (u_{\text{experiment}} - u_{\text{exact}}) + (u_{\text{exact}} - u_{\text{simulation}})$

u_{exact} is the value that will be produced if our simulation was running on an "ideal" computer. Here the first term is called the *experimental error* and the last term is the *result of verification*.

Real world vs. simulation (contd.)

In this expression the main source of uncertainty comes from the middle term.

The key is to minimize both the first term and the last term.

Metrics

- How do we even compute Δ?
- What metrics should we use?
- Should we take the square of differences?
- Should take the absolute value?
- Should we take the maximum difference?
- What if output values evolve over time? Should we compute the distance between two curves?
- The standard Root Mean Squared (RMS) often doesn't work
  - In case of weather simulations, for example, we are more interested in the extremes
- Visualization is extremely important

Validation approaches

- Senstivity analysis
  - Doesn't always work for stochastic simulations
- Probabilistic techniques
  - Monte Carlo simulations
  - Helps to know the distributions for input parameteres
  - Often needs very high computational resources

Conclusions

- The problem of validating simulations is far from solved
- We have many good ideas about how to approach this problem, still the area is a fertile ground for new, breakthrough research