

Implications of System Errors in the Context of Numerical Accuracy

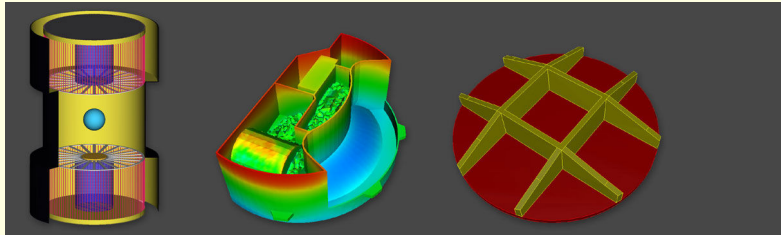
Mike Heroux, Patty Hough
Sandia National Laboratories

Vicki Howle
Texas Tech University

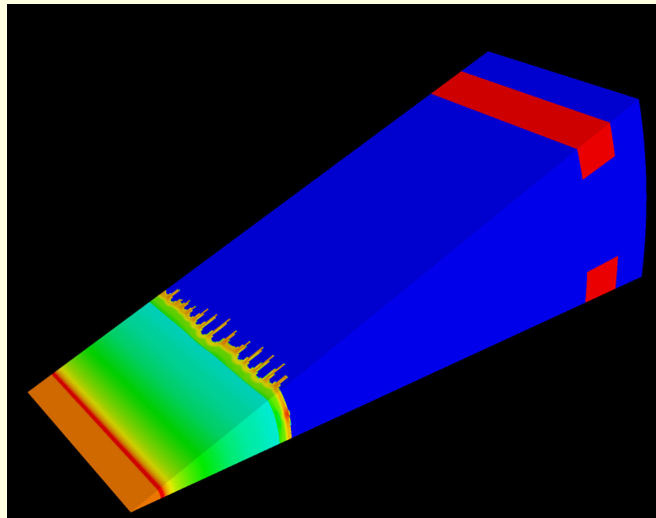
Questions for Applications

- What science/computing are you doing now, focused on computing more than science?
- What are you worried about, particularly thinking of future grand challenge science?
- What errors/faults do you want to be aware of/notified of?
- What do you want from tools/technologies?
- What have you done about it so far?
- What are you planning to do?
- What do you think is reasonable for apps people to do?

We develop numerical algorithms to support PDE-based simulations



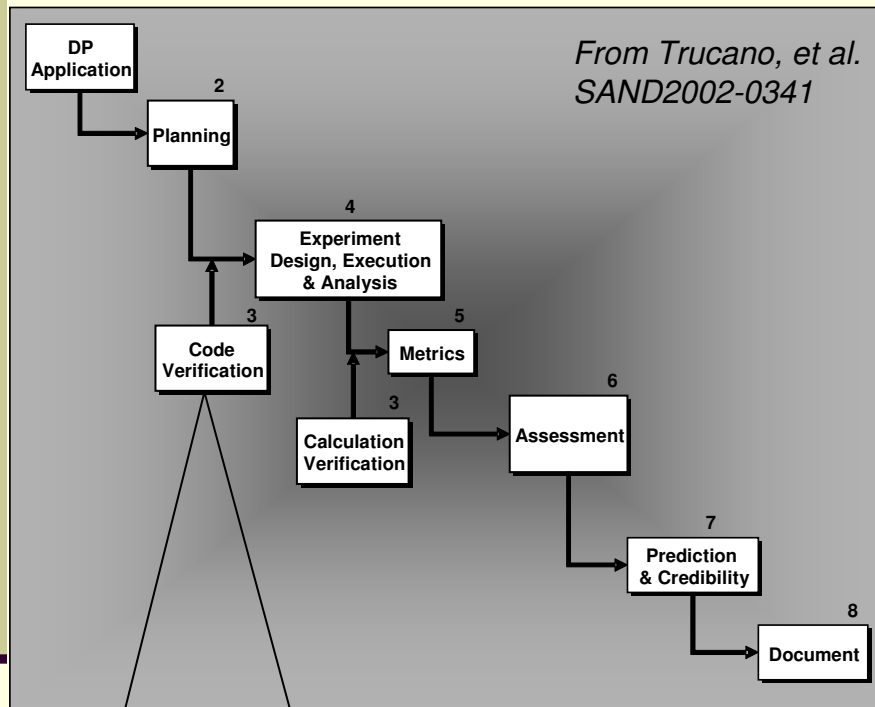
Examples of DAKOTA Applications: (from left) ICF Capsule Robust Design, Fireset Thermal Surety, Radar Support Structural Design



This visual depicts 20-degree periodic wedge simulation in 3-D of z-pinch liner implosion. Trilinos/ML's $H(\text{curl})$ multigrid magnetics solver is only viable solution method.

- For stockpile stewardship
 - Thermal
 - Mechanical
 - Structural dynamics
 - ...
- For science applications
 - Chemically reacting flow
 - Materials
 - ...
- Numerical algorithms we represent
 - Linear solvers
 - Optimization

Need evidence of credibility for simulations supporting high-consequence decisions



Are numerical algorithms implemented and *functioning* properly? Need to start considering effects of the computing environment in this assessment.

- Are we getting the right answer for the right reason?
- As architectures become more complex,
 - Implementation becomes more complicated
 - Simulation behavior becomes less predictable
 - **Failures come into play**
- Need confidence in the accuracy of our numerical algorithms

Hard errors have received the most attention to date

- Hard to miss
- Checkpoint/restart
 - Minimal response
 - Sufficient if failure rate not too high
- Still some open questions
 - How scalable are current approaches?
 - Can we predict failures?
 - Verification for checkpoint/restart?

Fault-tolerant algorithms are more efficient approaches to addressing hard errors

- Includes both “inherently fault-tolerant” and not
 - Recovery for iterative methods (Langou, *et al.*, 2007)
 - Meshless methods/chaotic relaxation, FFT (Geist & Engelmann, 2002)
 - Asynchronous parallel pattern search (Hough, *et al.*, 2001)
- But sometimes we need guaranteed correctness
 - Robust algorithms need correct computations
 - Examples: direct solvers, orthogonal subspaces
- And becoming more prevalent are...

Soft errors are becoming more prevalent due to small features operating at low voltages

- "At 8 nm process technology, it will be harder to tell a 1 from a 0." (Camp, 2008)
- 128k-node BlueGene/L: 1 soft error in L1 cache every 4-6 hours (Ziegler, *et al.*, 1996)
- ...
- Soft errors are scary to apps
 - Computation proceeds but is wrong
 - Careful verification required
 - What if verification has soft errors?

Consider GMRES as an example of how soft errors affect correctness

- Basic Steps

- 1) Compute Krylov subspace (sparse matrix-vector multiplies)
- 2) Compute orthonormal basis for Krylov subspace (matrix factorization)
- 3) Compute vector yielding minimum residual in subspace (linear least squares)
- 4) Map to next iterate in the full space
- 5) Repeat until residual is sufficiently small

- More examples in Bronevetsky & Supinski, 2008

Every calculation matters

- Small PDE Problem: Dim 21K, Nz 923K.
- ILUT/GMRES
- Correct computation 35 Iters: 343M FLOPS
- Two examples of a **single** bad floating point op

Description	Iterations	FLOPS	Recursive Residual Error	Solution Error
All Correct Calcs	35	343M	4.6e-15	1.0e-6
Iter=2, y[1] += 1.0 SpMV incorrect Ortho subspace	35	343M	6.7e-15	3.7e+3
Q[1][1] += 1.0 Non-ortho subspace	N/C	N/A	7.7e-02	5.9e+5

One possible approach is transactional computation

- Database transactions: atomic
- Transactional memory: atomic memory operation
- Transactional computation:
 - Designated sensitive computation region (orthogonalization step in *GMRES*)
 - Guarantee accurate computation or notify user

Needs to be coupled with guaranteed data regions

- User-designated reliable data region
- Extra protection to improve reliable data storage and transfer
- Examples
 - Original input data (needed for verification)
 - Linear solver: A, x, b
 - Orthogonal vectors for GMRES

More generally, what should application developers do?

- Abandon the assumption that the system can continue to guarantee reliability and correctness???
- Work with system, system software, middleware, etc. developers to learn what can be provided and to develop requirements
- Develop a more holistic view of application development - develop algorithms/applications suitable for running correctly through failure and handling multi-threading
- Reserve the right to use slower, more reliable systems

What I would like to see in future fault tolerance tools and technologies

- Integration
 - Vertical integration across hardware, system software, message-passing libraries, numerical algorithms, etc.
 - Integration across platform components (e.g., CPUs, storage, networks, etc.)
 - Community workshops
 - Multi-disciplinary development teams
- Standards
 - Influence the MPI 3 standard now
 - Interfaces for communicating with system components (CIFTS is a good start, but what if I don't want to use the FT backbone)
- Rigorous failure models
 - What failures occur and with what probability
 - Fault tolerance incurs overhead, so I want to manage my risks
- Flexible and scalable infrastructure
 - Scalable alternatives to failure prediction, detection, and recovery techniques for hard errors
 - Flexible platform management/scheduling
- Test beds
 - Need to do V&V in the presence of failures (in a controlled setting)
 - Need to distinguish between behavior resulting from fault and that resulting from other platform behavior