SAND2014-18363PE

Exceptional service in the national interest

Sandia National Laboratories

Resilient Iterative Linear Solvers via Skeptical Programming

James Elliott[1,2], Mark Hoemmen[1], Frank Mueller[2]

[1]Sandia National Laboratories

[2]North Carolina State University

NC STATE UNIVERSITY
Department of Computer Science

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2014-1285C

Overview

Sandia National Laboratories

- Questions?
- A Problem
- Myth of a Reliable Machine
- Reliability For Algorithms
  - Is the root cause of a fault important or is its manifestation?
- Skeptical Programming
  - All iterative ABFT schemes assume some form of detection… how?
- Questions?

Questions

Sandia National Laboratories

- Quantitative comparisons of ABFT schemes
  - Solve the same problem?
- Fault Models?
  - How critical are they? Granularity? (node, process, operation)
  - Fault rates? (we know so little)
- Reliable Programming
  - Data is often replicated (devices and hosts)
- Correct solution (research) vs performance
  - Bridging the gap… theory/math and systems

The Problem (well one of them)

Sandia National Laboratories

Function: Puppy = make_puppy(Dog, Dog)

The Problem (well one of them)

Sandia National Laboratories

Function: Puppy = make_puppy(Dog, Dog)

The Problem (well one of them)

Sandia National Laboratories

Function: Puppy = make_puppy(Dog, Dog)

+

The Problem (well one of them)

Sandia National Laboratories

Function: Puppy = make_puppy(Dog, Dog)

+

=

The Problem (Well one of them)

Sandia National Laboratories

The Problem (well one of them)

Two things happened:

1. You noticed a fault, therefore …
2. You can respond (correct the fault)

Detection and Correction

The Problem (well one of them)

Sandia National Laboratories

How to detect that a silent transient fault occured?

- Enforce fine-grained correctness (systems approach)
  - ✓ **Do calculations multiple times and vote**
  - × **Ignores numerical properties of algorithms**

The Problem (well one of them)

Sandia National Laboratories

How to detect?

- Enforce fine-grained correctness (systems approach)
  - ✓ **Do calculations multiple times and vote**
  - × **Ignores numerical properties of algorithms**
- Convergence (numerical error dampening)
  - ✓ **Iterative methods have proven properties to converge to a correct solution given preconditions/assumptions about the inputs**
  - × **Convergence promise invalidated if assumptions/preconditions are invalidated at any point.**

The Problem (well one of them)

Sandia National Laboratories

How to detect?

- Enforce fine-grained correctness (systems approach)
  - ✓ **Do calculations multiple times and vote**
  - × **Ignores numerical properties of algorithms**
- Convergence (numerical error dampening)
  - ✓ **Iterative methods have proven properties to converge to a correct solution given preconditions/assumptions about the inputs**
  - × **Convergence promise invalidated if assumptions/preconditions are invalidated at any point.**
- Can we do algorithm-based fault tolerance while still exploiting the numerical properties?

Myth of a Reliable Machine

Sandia National Laboratories

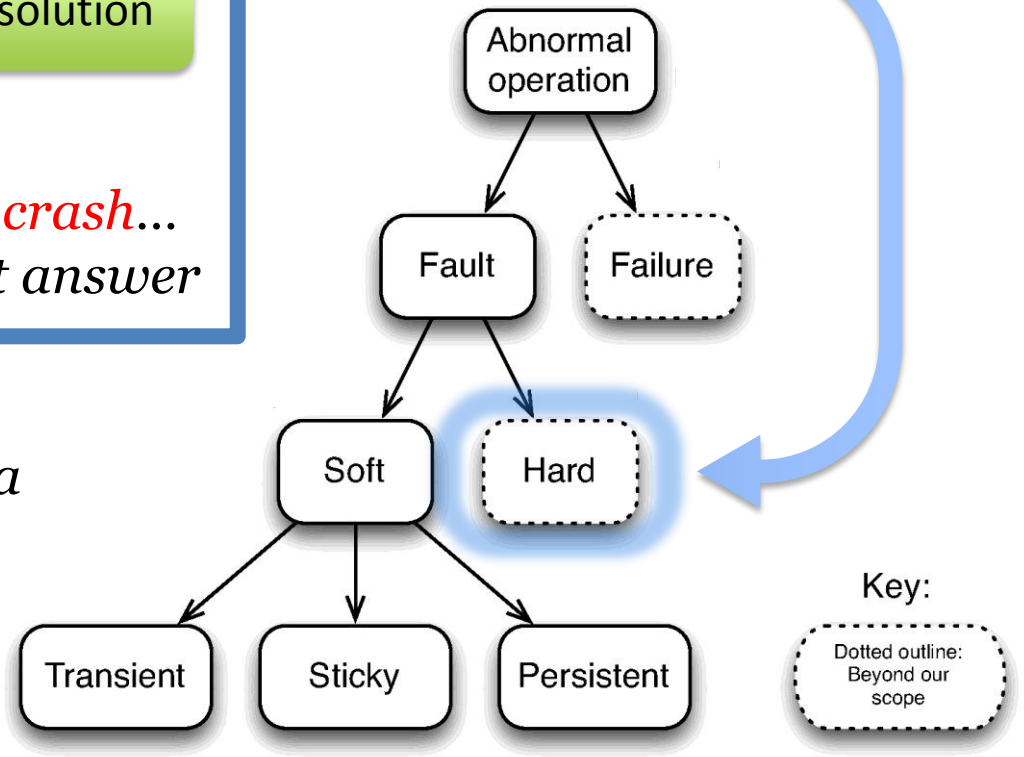
Current algorithms assume ***numerical reliability***

Reliable Machine

start → operations → solution

As long the application does not crash…
we will get the right answer

Hard Faults: Not in scope of this work



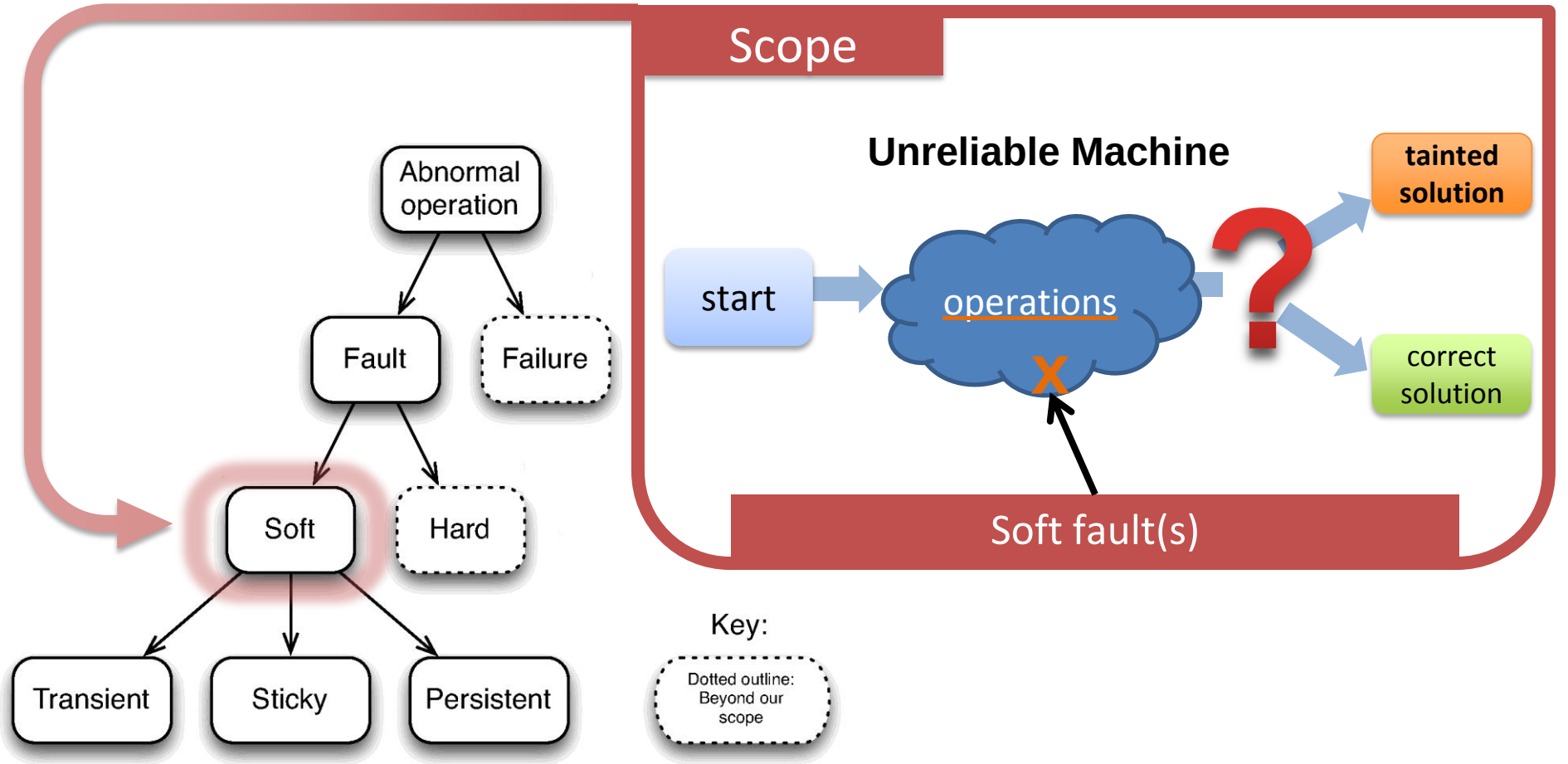
Machine reliability addressed via checkpoint/restart, replication, algorithm based approaches

Myth of a Reliable Machine

Sandia National Laboratories

Numerical reliability:

numerical operations may be unreliable



Reliability for Algorithms

Sandia National Laboratories

Numerical Method = Theory + Data

(algorithm) (with assumptions)

- **Two Categories:**
  - **Data:** data required theoretically by the algorithm (Preconditions/Assumptions)

    E.g., for GMRES: A, x0, orthonormal subspace of A, right-hand side
  - **Meta-data:** "stuff" required to implement the algorithm

    E.g., loop counters, sparse data structures (indices), code

Regardless of who implements GMRES or in what language, the data can always be assumed (otherwise we would not be considering GMRES!)
Language and Implementation details may result in vastly different meta-data.

- Most extensible research: **focus on faults in data.**

Reliability for Algorithms

Sandia National Laboratories

- Result of silent fault in **data**:
  - **Silent Data Corruption (SDC)**
- Result of silent transient fault in **meta-data**
  - Faults in code:
    - Weird behavior
    - Perform wrong operation (e.g., add instead of divide) **= SDC** (output is “tainted”)
  - Faults in loop counters, indices (if no segfault/crash)
    - Operate on incorrect data **= SDC** (output is “tainted”)
    - Iterate too much or too little **= SDC** (output is “tainted”)
- Perhaps more

Key: corrupted data model *many* types of faults!

Big Idea: Bounded Error

Sandia National Laboratories

- Focus on ensuring **bounded error,** *rather than* ***detecting and correcting <u>all errors</u>*.**

Big Idea: Bounded Error

Sandia National Laboratories

- Focus on ensuring **bounded error,** *rather than* ***detecting and correcting all errors*****.**
- Use preconditions/assumptions on inputs with algorithms' theoretical basis to derive invariants.

Big Idea: Bounded Error

Sandia National Laboratories

- Focus on ensuring **bounded error,** *rather than* ***detecting and correcting all errors*****.**
- Use preconditions/assumptions on inputs with algorithms' theoretical basis to derive invariants.
- Invariant checks are silly in a reliable environment
  - If unreliable – **invariants serve as cheap detectors** that the algorithm is in a **theoretically impossible state**.

Big Idea: Bounded Error

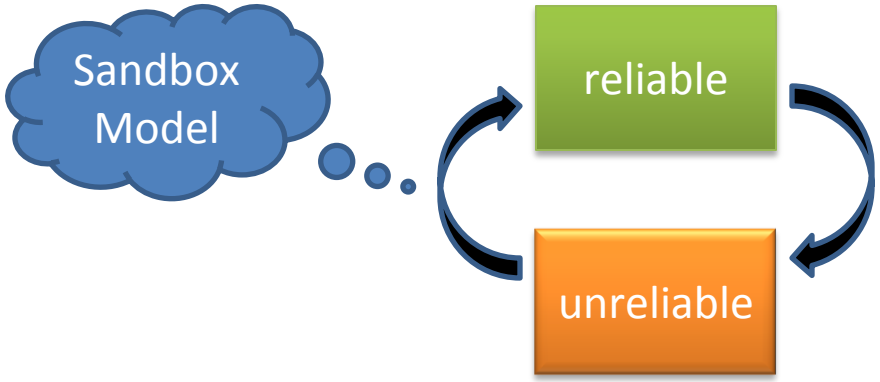
Sandia National Laboratories

- Focus on ensuring **bounded error,** *rather than* ***detecting and correcting all errors*****.**
- Use preconditions/assumptions on inputs with algorithms' theoretical basis to derive invariants.
- Invariant checks are silly in a reliable environment
  - If unreliable – **invariants serve as cheap detectors** that the algorithm is in a **theoretically impossible state**.
- No promise to detect/correct all errors
  - One piece of the resilient algorithm pie
- We call this Skeptical Programming.
  - **Be skeptical of computations performed on an unreliable machine**

Reliability

Looking from the Reliable Set:

- **Sandbox Model**



Operate unreliably, but do not trust the results.

- **Selective Reliability:**

  What operations need to be reliable?

Universe of **operations** inherently ***unreliable***

Unreliable ops:
Preconditioner call (and all its operations)

Reliable ops:
Flexible GMRES, (currently all collectives)

Selective Reliability

Algorithm 1 Flexible GMRES (FGMRES)

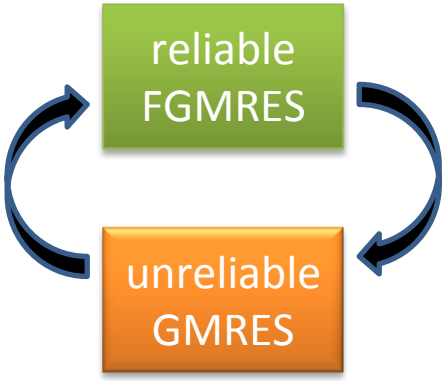
Input: Linear system $Ax = b$ and initial guess x_0
Output: Approximate solution x_m for some $m \geq 0$

1: $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ ▷ Unpreconditioned initial residual
2: $\beta := \|\mathbf{r}_0\|_2$, $\mathbf{q}_1 := \mathbf{r}_0/\beta$
3: **for** $j = 1, 2, \ldots$ until convergence **do**
4: Solve $\mathbf{q}_j = \mathbf{M}_j\mathbf{z}_j$ ▷ Apply current preconditioner
5: $\mathbf{v}_{j+1} := \mathbf{A}\mathbf{z}_j$ ▷ Apply the matrix $\mathbf{A}$
6: **for** $i = 1, 2, \ldots, k$ **do** ▷ Orthogonalize
7: $h_{i,j} := \mathbf{q}_i \cdot \mathbf{v}_{j+1}$
8: $\mathbf{v}_{j+1} := \mathbf{v}_{j+1} - h_{i,j}\mathbf{q}_i$
9: **end for**
10: $h_{j+1,j} := \|\mathbf{v}_{j+1}\|_2$
11: Update rank-revealing decomposition of $\mathbf{H}(1{:}j, 1{:}j)$
12: **if** $\mathbf{H}(j+1, j)$ is less than some tolerance **then**
13: **if** $\mathbf{H}(1{:}j, 1{:}j)$ not full rank **then**
14: Did not converge; report error
15: **else**
16: Solution is $\mathbf{x}_{j-1}$ ▷ Happy breakdown
17: **end if**
18: **else**
19: $\mathbf{q}_{j+1} := \mathbf{v}_{j+1}/h_{j+1,j}$
20: **end if**
21: $\mathbf{y}_j := \arg\min_y \|\mathbf{H}(1{:}j+1, 1{:}j)\mathbf{y} - \beta\mathbf{e}_1\|_2$
22: $\mathbf{x}_j := \mathbf{x}_0 + [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_j]\mathbf{y}_j$ ▷ Compute solution update
23: **end for**

M_j are the preconditioners:

$$\mathbf{z}_j = \text{gmres}(\mathbf{A}, \mathbf{q}_j)$$

M_j represents using GMRES as a preconditioner… inside FGMRES.



Skeptical Programming

Sandia National Laboratories

Algorithm 1: **GMRES**

for $l = 1$ to do
  $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(j-1)}$
  $\mathbf{q}_1 := \mathbf{r} / ||r||_2$
  for $j = 1$ to *restart* do
    $\mathbf{w}_0 := \mathbf{A}\mathbf{q}_j$
    for $i = 1$ to j do
      $h_{i,j} := \mathbf{q}_i \cdot \mathbf{w}_{i-1}$
      $\mathbf{w}_i := \mathbf{w}_{i-1} - h_{i,j}\mathbf{q}_i$
    end
    $h_{j+1,j} := ||\mathbf{w}||_2$
    $\mathbf{q}_{j+1} := \mathbf{w}/h_{j+1,j}$
    Find $\mathbf{y} = \min ||\mathbf{H}_j\mathbf{y} - ||\mathbf{b}||\,\mathbf{e}_1||_2$
    Evaluate convergence criteria
    Optionally, compute $\mathbf{x}_j = \mathbf{Q}_j\mathbf{y}$
  end
end

Skeptical Programming

Sandia National Laboratories

Algorithm 1: **GMRES**

for $l = 1$ to do
  $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(j-1)}$
  $\mathbf{q}_1 := \mathbf{r} / ||r||_2$
  for $j = 1$ to *restart* do
    $\mathbf{w}_0 := \mathbf{A}\mathbf{q}_j$
    for $i = 1$ to j do
      $h_{i,j} := \mathbf{q}_i \cdot \mathbf{w}_{i-1}$
      $\mathbf{w}_i := \mathbf{w}_{i-1} - h_{i,j}\mathbf{q}_i$
    end
    $h_{j+1,j} := ||\mathbf{w}||_2$
    $\mathbf{q}_{j+1} := \mathbf{w} / h_{j+1,j}$
    Find $\mathbf{y} = \min ||\mathbf{H}_j\mathbf{y} - ||\mathbf{b}|| \, \mathbf{e}_1||_2$
    Evaluate convergence criteria
    Optionally, compute $\mathbf{x}_j = \mathbf{Q}_j\mathbf{y}$
  end
end

Orthogonalization
Builds upper Hessenburg matrix (H)

Skeptical Programming

Sandia National Laboratories

Algorithm 1: GMRES

for $l = 1$ to do
  $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(j-1)}$
  $\mathbf{q}_1 := \mathbf{r} / ||r||_2$
  for $j = 1$ to *restart* do
    $\mathbf{w}_0 := \mathbf{A}\mathbf{q}_j$
    for $i = 1$ to j do
      $h_{i,j} := \mathbf{q}_i \cdot \mathbf{w}_{i-1}$
      $\mathbf{w}_i := \mathbf{w}_{i-1} - h_{i,j}\mathbf{q}_i$
    end
    $h_{j+1,j} := ||\mathbf{w}||_2$
    $\mathbf{q}_{j+1} := \mathbf{w} / h_{j+1,j}$
    Find $\mathbf{y} = \min ||\mathbf{H}_j\mathbf{y} - ||\mathbf{b}|| \, \mathbf{e}_1||_2$
    Evaluate convergence criteria
    Optionally, compute $\mathbf{x}_j = \mathbf{Q}_j\mathbf{y}$
  end
end

Theoretical Bounds on the Arnoldi Process

$$\|\mathbf{w}_0\| = \|\mathbf{A}\mathbf{q}_j\| \leq \|\mathbf{A}\|_2 \|\mathbf{q}_j\|_2$$
$$\|\mathbf{w}_0\| \leq \|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F$$

From isometry of orthogonal projections,

$$|h_{i,j}| \leq \|\mathbf{A}\|_F$$

- $h_{i,j}$ form Hessenberg Matrix
- Bound only computed once, valid for entire solve

Skeptical Programming

Sandia National Laboratories

Algorithm 1: GMRES

for $l = 1$ to do
  $\mathbf{r} := \mathbf{b} - \mathbf{A}\mathbf{x}^{(j-1)}$
  $\mathbf{q}_1 := \mathbf{r} / ||r||_2$
  for $j = 1$ to *restart* do
    $\mathbf{w}_0 := \mathbf{A}\mathbf{q}_j$
    for $i = 1$ to j do
      $h_{i,j} := \mathbf{q}_i \cdot \mathbf{w}_{i-1}$
      $\mathbf{w}_i := \mathbf{w}_{i-1} - h_{i,j}\mathbf{q}_i$
    end
    $h_{j+1,j} := ||\mathbf{w}||_2$
    $\mathbf{q}_{j+1} := \mathbf{w} / h_{j+1,j}$
    Find $\mathbf{y} = \min ||\mathbf{H}_j\mathbf{y} - ||\mathbf{b}|| \, \mathbf{e}_1||_2$
    Evaluate convergence criteria
    Optionally, compute $\mathbf{x}_j = \mathbf{Q}_j\mathbf{y}$
  end
end

Theoretical Bounds on the Arnoldi Process

$$\|\mathbf{w}_0\| = \|\mathbf{A}\mathbf{q}_j\| \leq \|\mathbf{A}\|_2 \|\mathbf{q}_j\|_2$$
$$\|\mathbf{w}_0\| \leq \|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F$$

From isometry of orthogonal projections,

$$|h_{i,j}| \leq \|\mathbf{A}\|_F$$

- $h_{i,j}$ form Hessenberg Matrix
- Bound only computed once, valid for entire solve
- **Identifies that something isn't right**

Sandia National Laboratories

Preconditioners

$$\tilde{\mathbf{z}}_i = \mathbf{M}^{-1}\mathbf{q}_i$$

Preconditioners

$$\tilde{\mathbf{z}}_i = \mathbf{M}^{-1}\mathbf{q}_i$$

Fault:
subdomain permutes preconditioner's output
(preserves length)

X-axis:
subdomain also scales permuted output

Preconditioners

$$\tilde{\mathbf{z}}_i = \mathbf{M}^{-1}\mathbf{q}_i$$

Fault:
subdomain permutes preconditioner's output
(preserves length)

X-axis:
subdomain also scales permuted output

120 *inner* preconditioner applies using Flexible Gmres + Gmres + Additive Schwarz (with zero overlap) ILU(0) as subdomain solver (right preconditioning).

Preconditioners

$$\tilde{\mathbf{z}}_i = \mathbf{M}^{-1}\mathbf{q}_i$$

Fault:
subdomain permutes preconditioner's output
(preserves length)

X-axis:
subdomain also scales permuted output

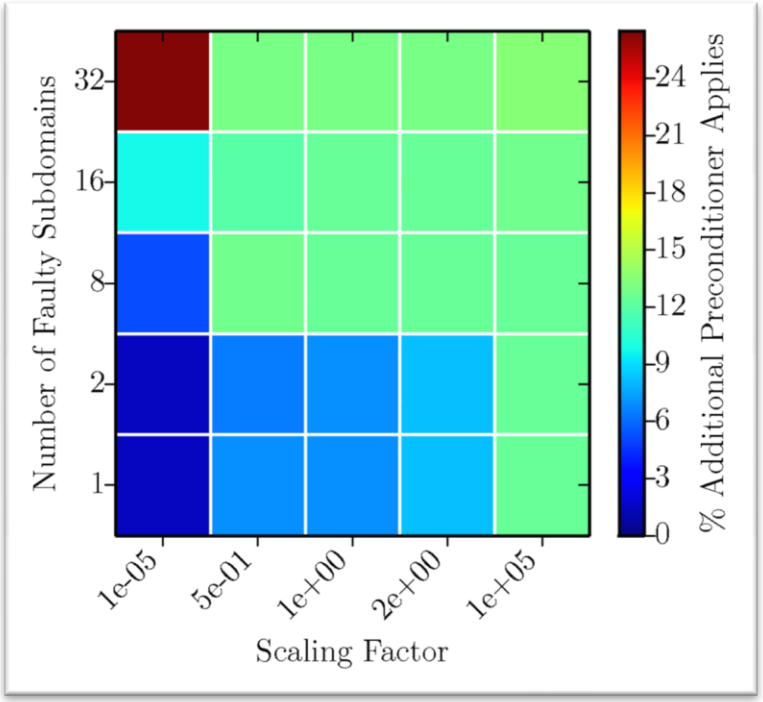
reliable

faulty

120 *inner* preconditioner applies using Flexible Gmres + Gmres + Additive Schwarz (with zero overlap) ILU(0) as subdomain solver (right preconditioning).

No communication with other subdomains

Mean percent additional preconditioner applies

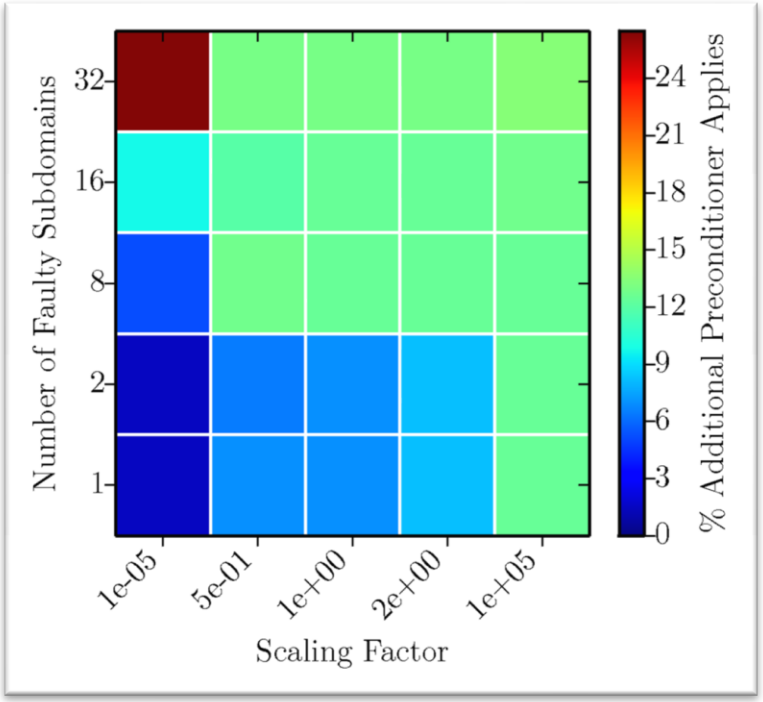


Fault:
subdomain permutes preconditioner's output
(preserves length)

X-axis:
subdomain also scales permuted output

120 *inner* preconditioner applies using Flexible Gmres + Gmres + Additive Schwarz (with zero overlap) ILU(0) as subdomain solver (right preconditioning).

Mean percent additional preconditioner applies



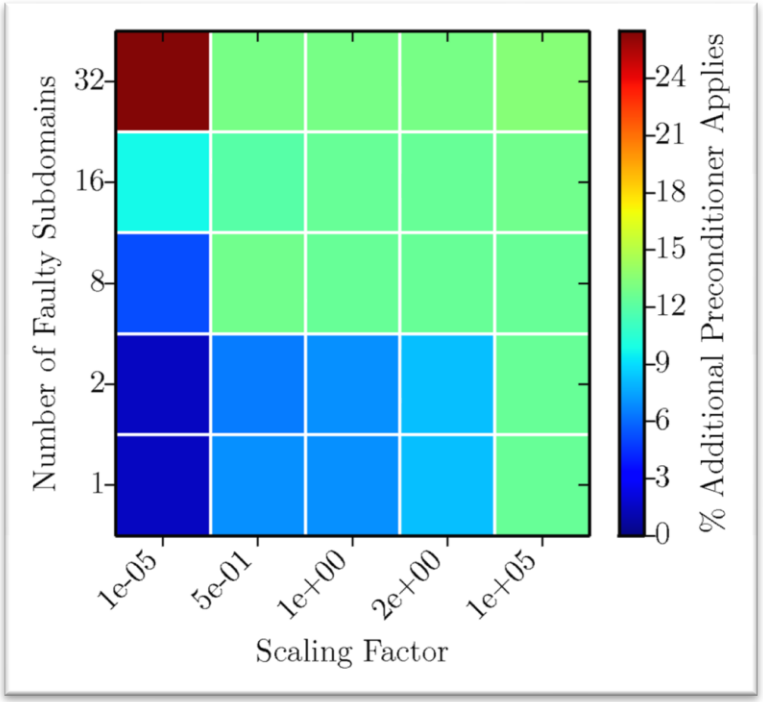
Fault:
subdomain permutes preconditioner's output
(preserves length)

X-axis:
subdomain also scales permuted output

120 *inner* preconditioner applies using Flexible Gmres + Gmres + Additive Schwarz (with zero overlap) ILU(0) as subdomain solver (right preconditioning).

Statistics from introducing a single fault at all possible (120) preconditioner applications.

Mean percent additional preconditioner applies



all subdomains faulty

1 faulty subdomain

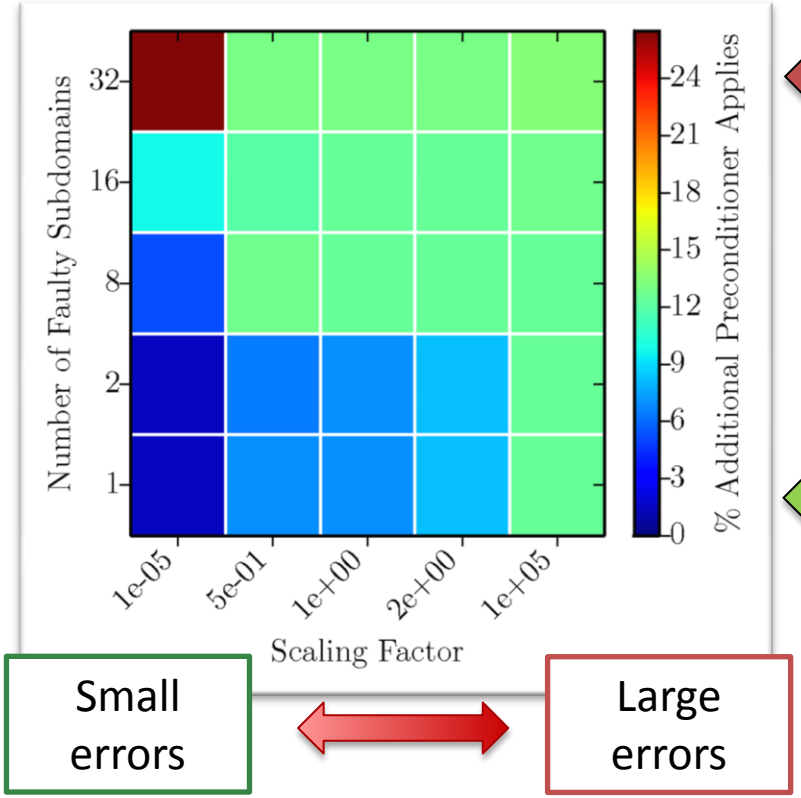
Fault:
subdomain permutes preconditioner's output
(preserves length)

X-axis:
subdomain also scales permuted output

120 *inner* preconditioner applies using Flexible Gmres + Gmres + Additive Schwarz (with zero overlap) ILU(0) as subdomain solver (right preconditioning).

Statistics from introducing a single fault at all possible (120) preconditioner applications.

Mean percent additional preconditioner applies



all subdomains faulty

1 faulty subdomain

Small errors

Large errors

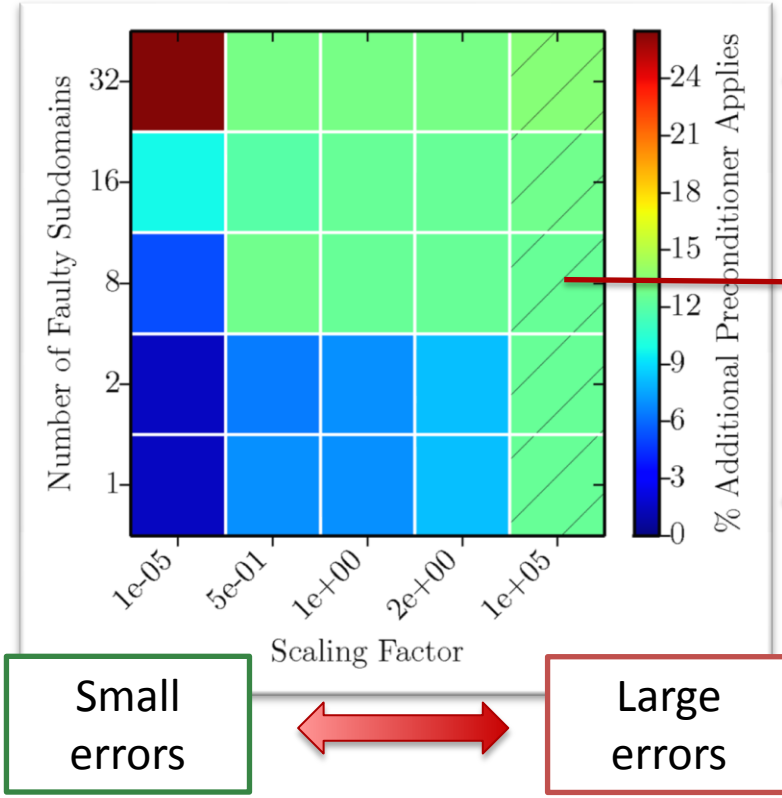
Fault:
subdomain permutes preconditioner's output
(preserves length)

X-axis:
subdomain also scales permuted output

120 *inner* preconditioner applies using Flexible Gmres + Gmres + Additive Schwarz (with zero overlap) ILU(0) as subdomain solver (right preconditioning).

Statistics from introducing a single fault at all possible (120) preconditioner applications.

Mean percent additional preconditioner applies



all subdomains faulty

Detected via norm bound

1 faulty subdomain

Small errors

Large errors

Fault:
subdomain permutes preconditioner's output
(preserves length)

X-axis:
subdomain also scales permuted output

120 *inner* preconditioner applies using Flexible Gmres + Gmres + Additive Schwarz (with zero overlap) ILU(0) as subdomain solver (right preconditioning).

Statistics from introducing a single fault at all possible (120) preconditioner applications.

Multiple Faults and Reactive FT

Sandia National Laboratories

- What to do when a fault is detected – options based on FT strategy

Solver Type	Restart	Abort Solve
GMRES		
Restarted GMRES	X	
Nested Solvers	X	X

Restarting:

theoretically bad for GMRES (you lose valuable state)

Aborting: inner solve exits abnormally

Both cases: (for right preconditioned solver)– must apply preconditioner to get solution. (This can be expensive!)

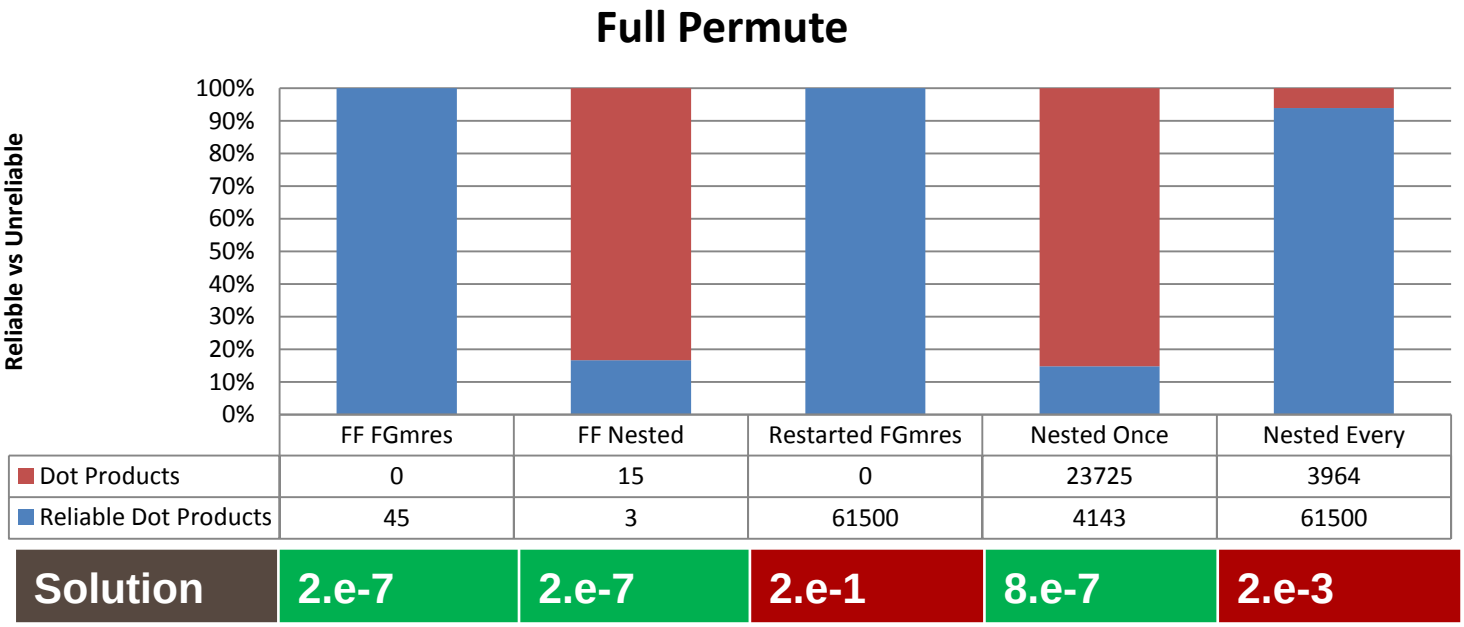
Residual checks (Ax-b), require preconditioner application

Multiple Faults and Reactive FT

Sandia National Laboratories

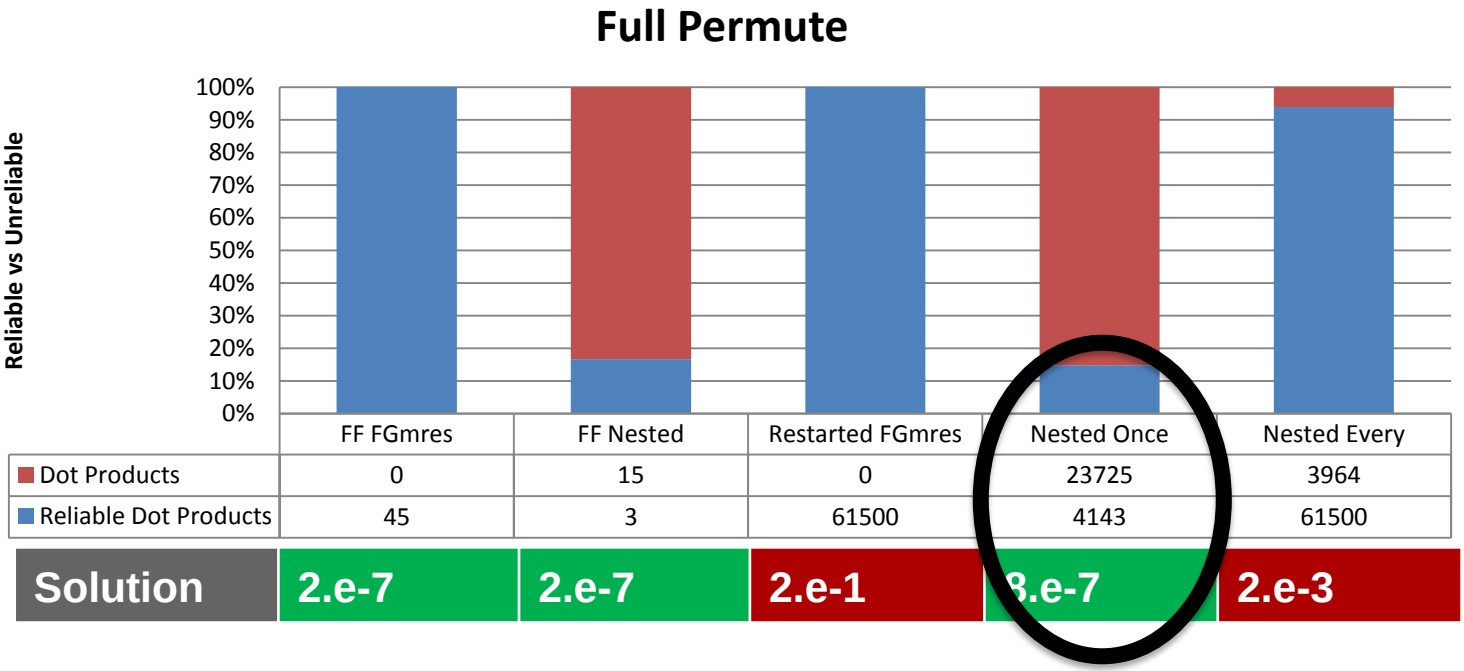
- Experiment, 10 MPI processes
  - at every preconditioner apply, single MPI rank is faulty
  - Fault: MPI process permutes (shuffles) all values in the output from it's preconditioner or swamps the min and max values in its output
  - Requirement: residual (||Ax-b||) less than 1x10^-6
  - Solver:
    - Reliable Restarted FGmres: Flexible GMRES, with reliable (triple modular redundancy) dot products and Sparse Mat-Vec (SpMV) – 3x cost in flops
    - Reliable Restarted FGmres + Gmres: Nested solver, e.g., FT-GMRES
  - Preconditioner :
    - 10 layer Multigrid – one of (if not) the best Poisson solvers
  - Solver + Preconditioner are widely used because of robustness and speed
    - Compares against a "good" solver

Multiple Faults and Reactive FT



	FF FGmres	FF Nested	Restarted FGmres	Nested Once	Nested Every
Dot Products	0	15	0	23725	3964
Reliable Dot Products	45	3	61500	4143	61500
Solution	**2.e-7**	**2.e-7**	**2.e-1**	**8.e-7**	**2.e-3**

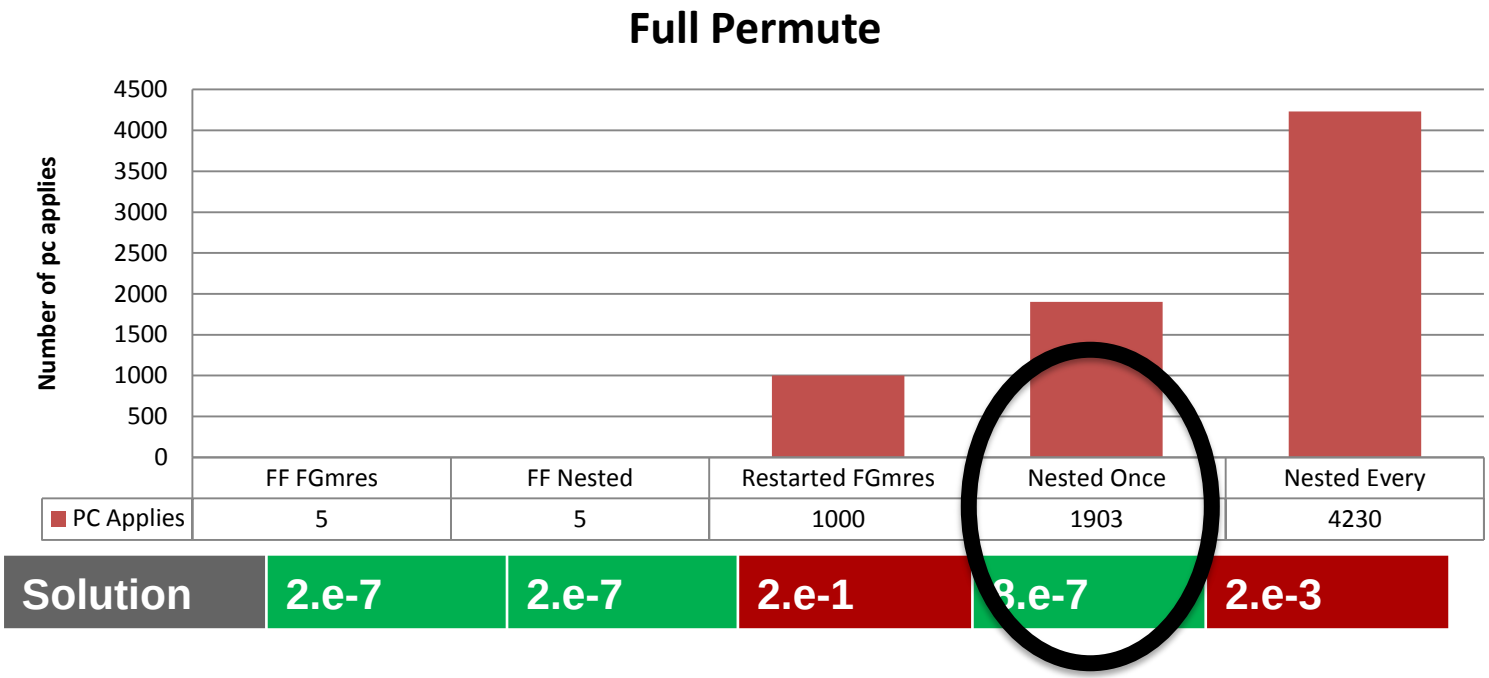
Multiple Faults and Reactive FT



Checks residual once, all work is “bad’, single checks ensures you do not go backwards

Total work equal to work done in failure free

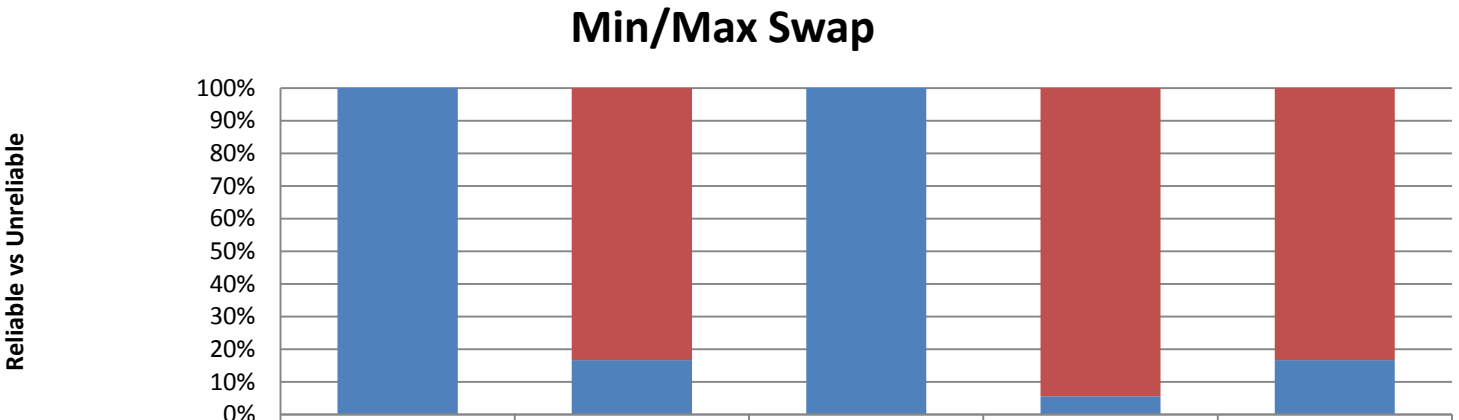
Multiple Faults and Reactive FT



Checks residual once, all work is “bad’, single checks ensures you do not go backwards

Total work equal to work done in failure free

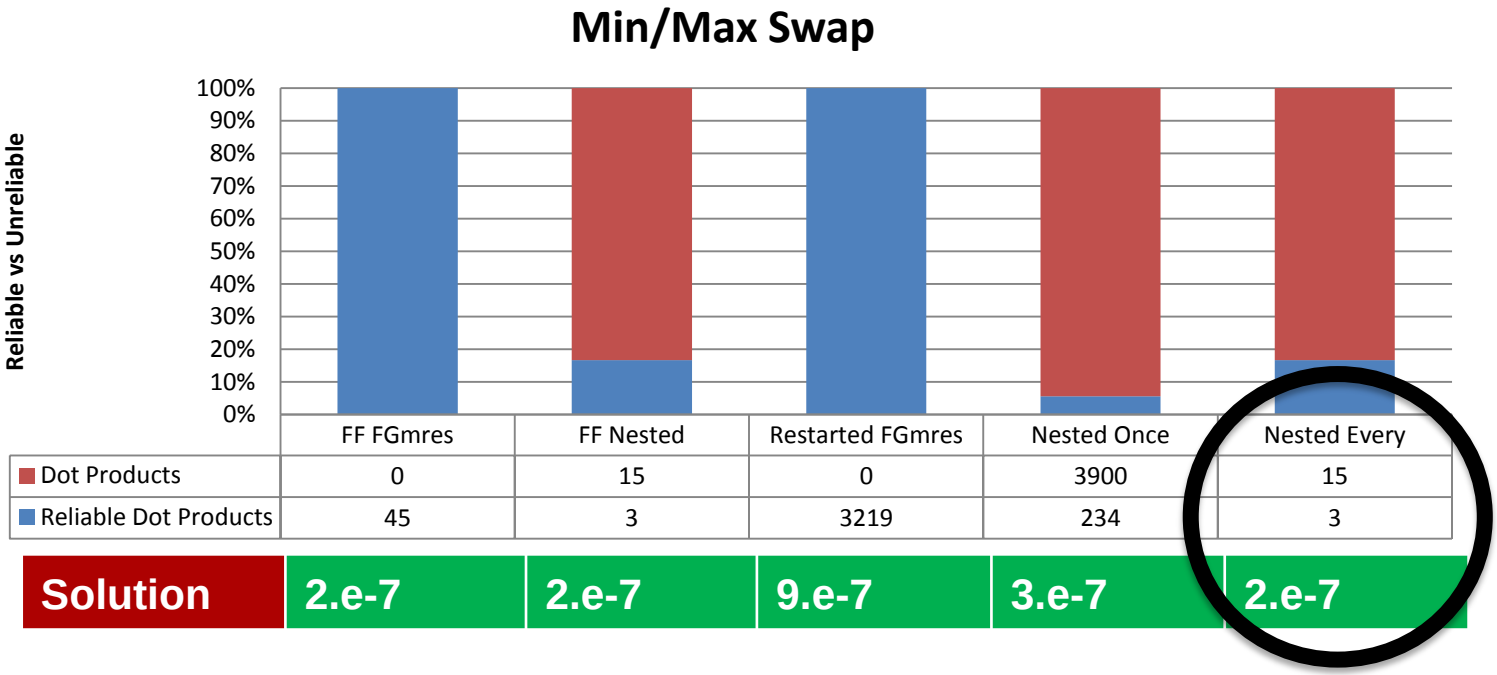
Multiple Faults and Reactive FT



	FF FGmres	FF Nested	Restarted FGmres	Nested Once	Nested Every
Dot Products	0	15	0	3900	15
Reliable Dot Products	45	3	3219	234	3

Solution	2.e-7	2.e-7	9.e-7	3.e-7	2.e-7

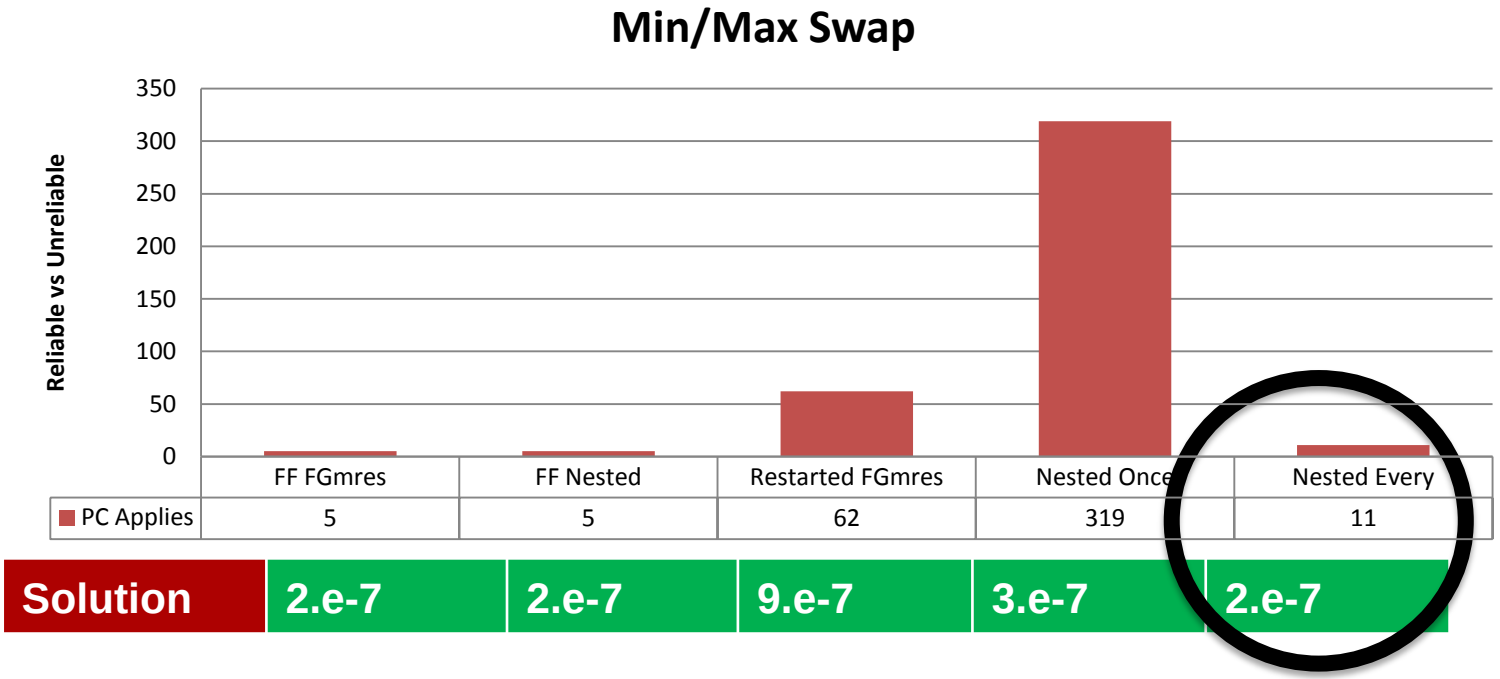
Multiple Faults and Reactive FT



	FF FGmres	FF Nested	Restarted FGmres	Nested Once	Nested Every
Dot Products	0	15	0	3900	15
Reliable Dot Products	45	3	3219	234	3
Solution	**2.e-7**	**2.e-7**	**9.e-7**	**3.e-7**	**2.e-7**

Checks residual every time, catches “bad” errors early
Total work equal to work done in failure free

Multiple Faults and Reactive FT

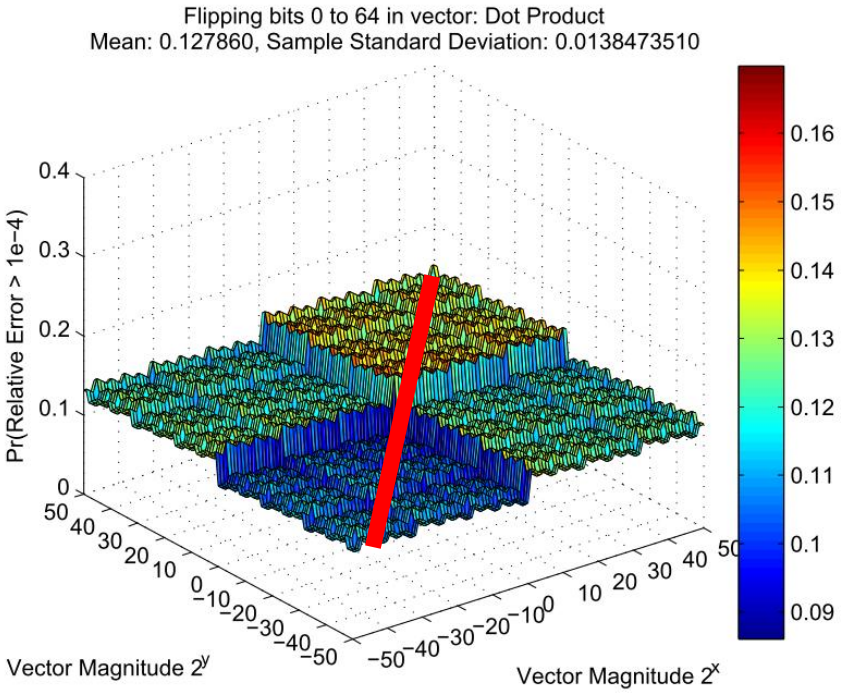


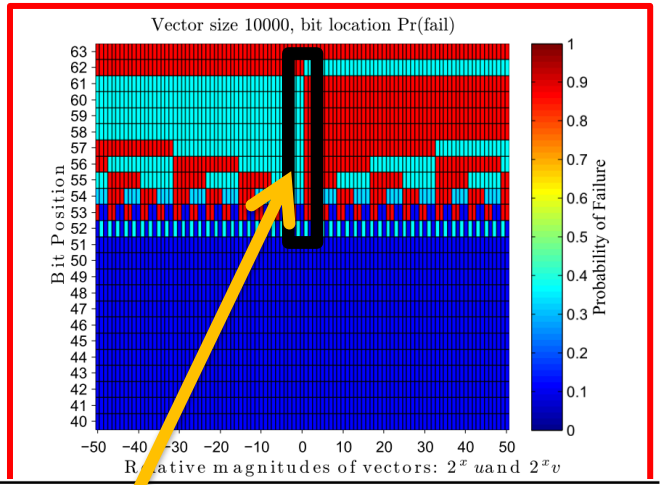
Checks residual every time, catches “bad” errors early
Total work equal to work done in failure free

James Elliott
Computer Science, North Carolina State Univ.

- Example: Dot products:

$$v = (-1)^{sign}\left(1 + \sum_{i=0}^{51} b_i 2^{i-52} \times 2^{e-1023}\right)$$



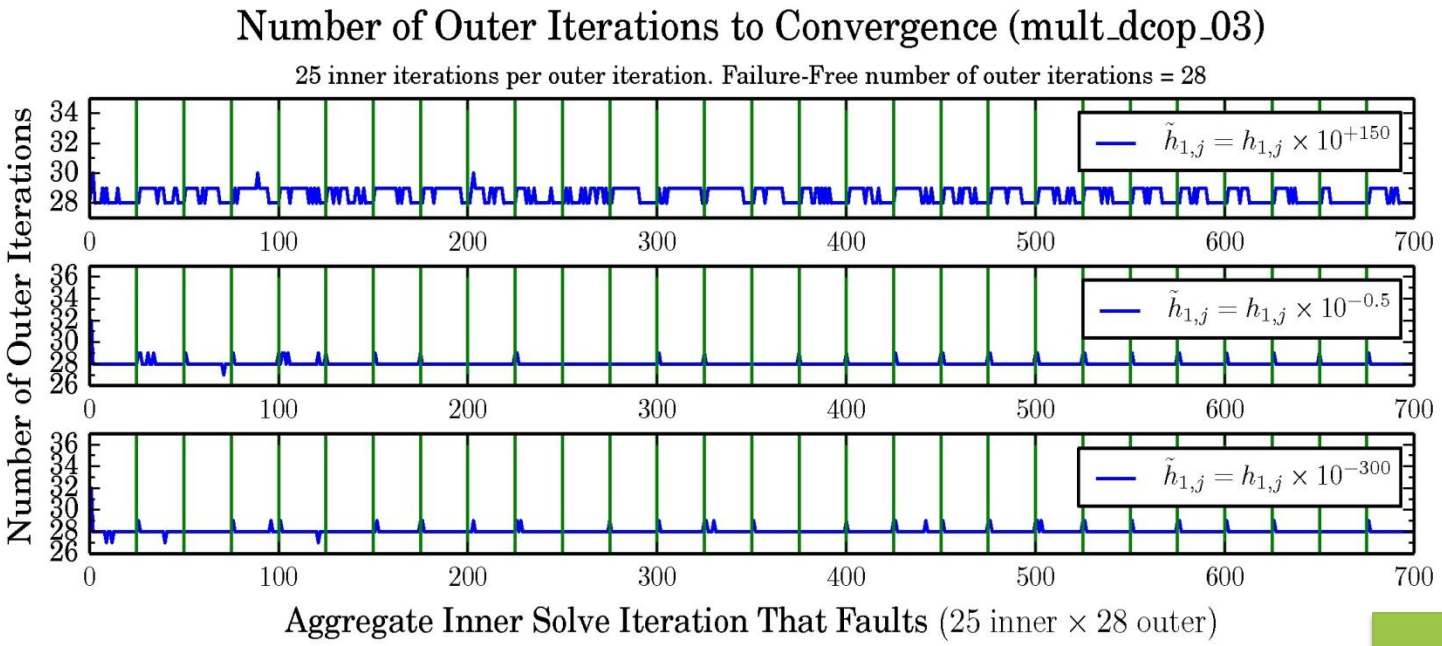


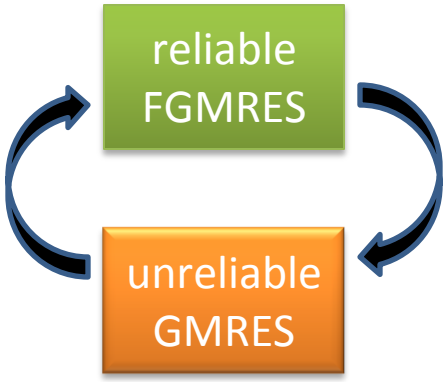
	Exponent Bits				
Base10	b_e	...	b_{52}	Bias Relation	Effective Exp.
2	10000000000			$2^{1024-1023}$	2^1
1	01111111111			$2^{1023-1023}$	2^0
.5	01111111110			$2^{1022-1023}$	2^{-1}

Figure shows probability of experiencing large error.

Due to floating point implementation, not all bit flips create large error – can model the probability of experiencing specific magnitudes of error.

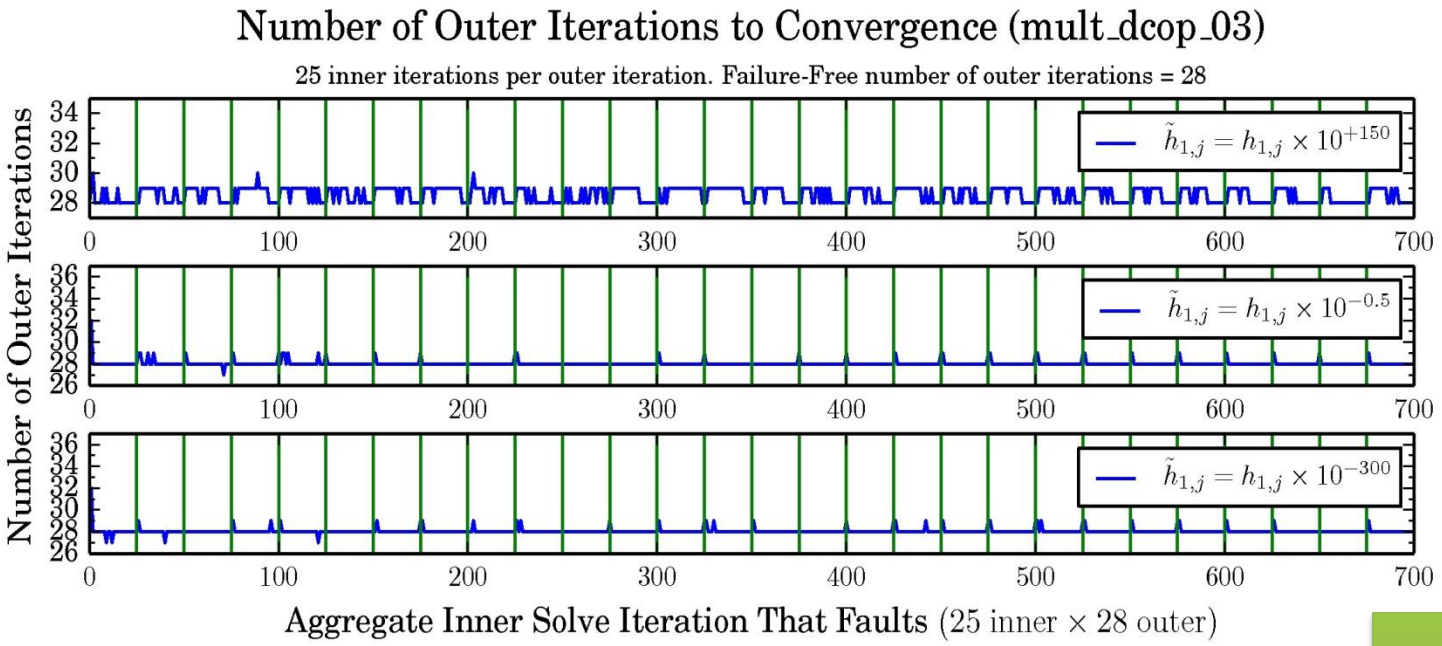
Experiments



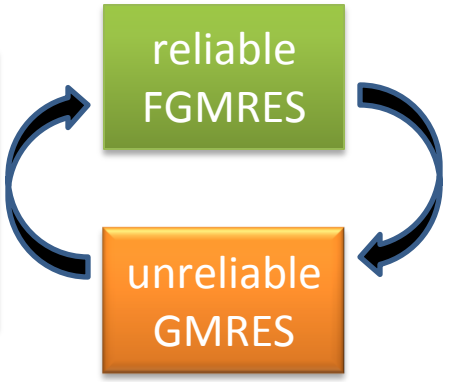


Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Experiments



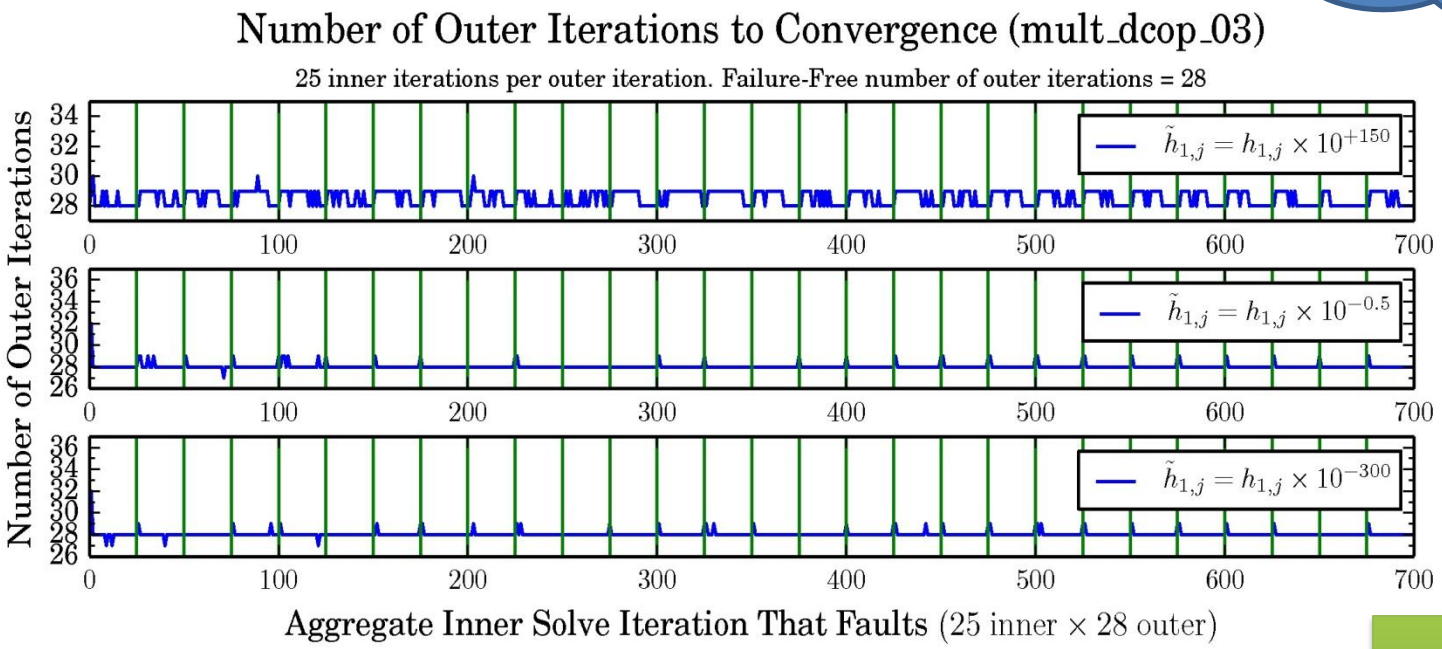
Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer
x-axis:
introducing single fault in x-th orthogonalization



Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Experiments

700 experiments per fault type



Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer
x-axis:
introducing single fault in x-th orthogonalization

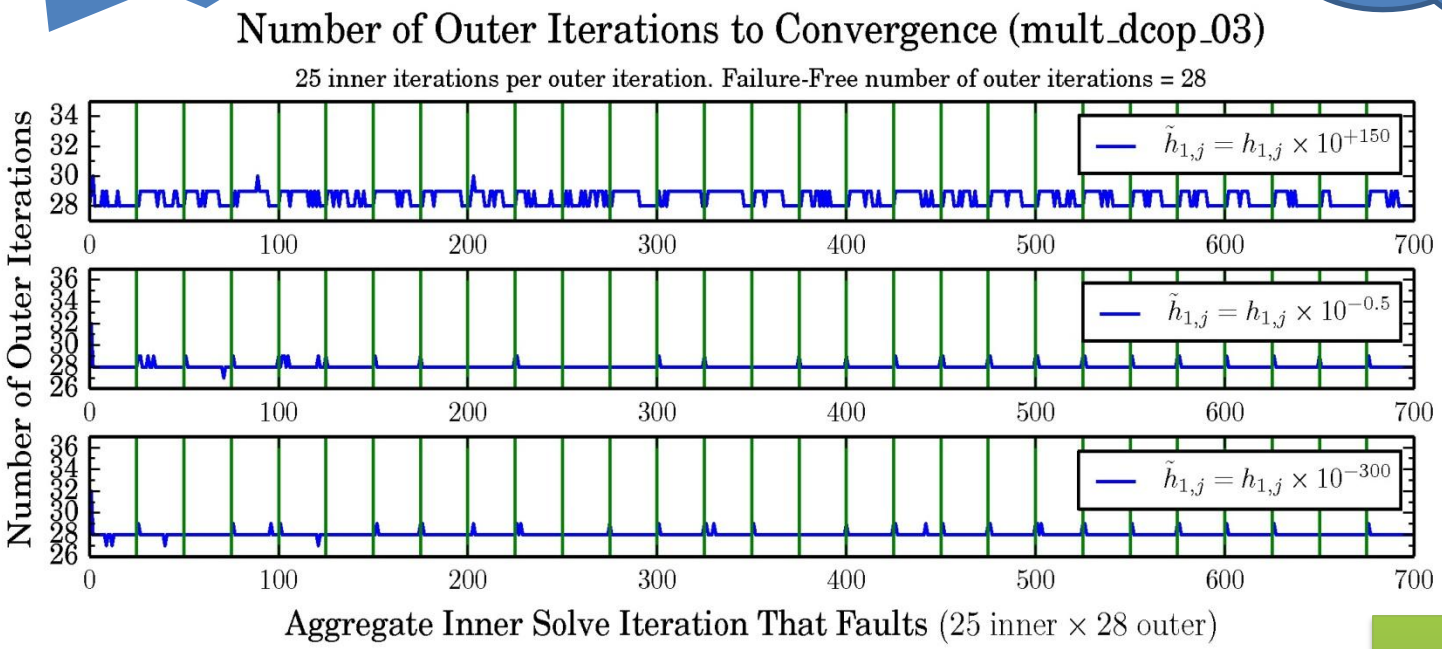
reliable FGMRES

unreliable GMRES

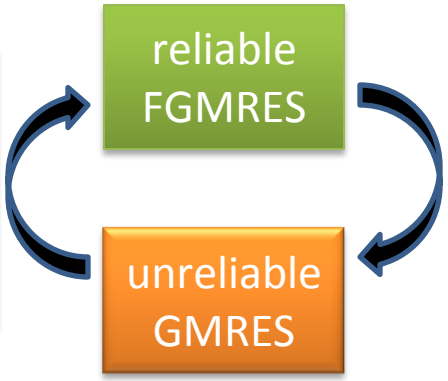
Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Inject ***single*** fault
Study impact

700
experiments per
fault type



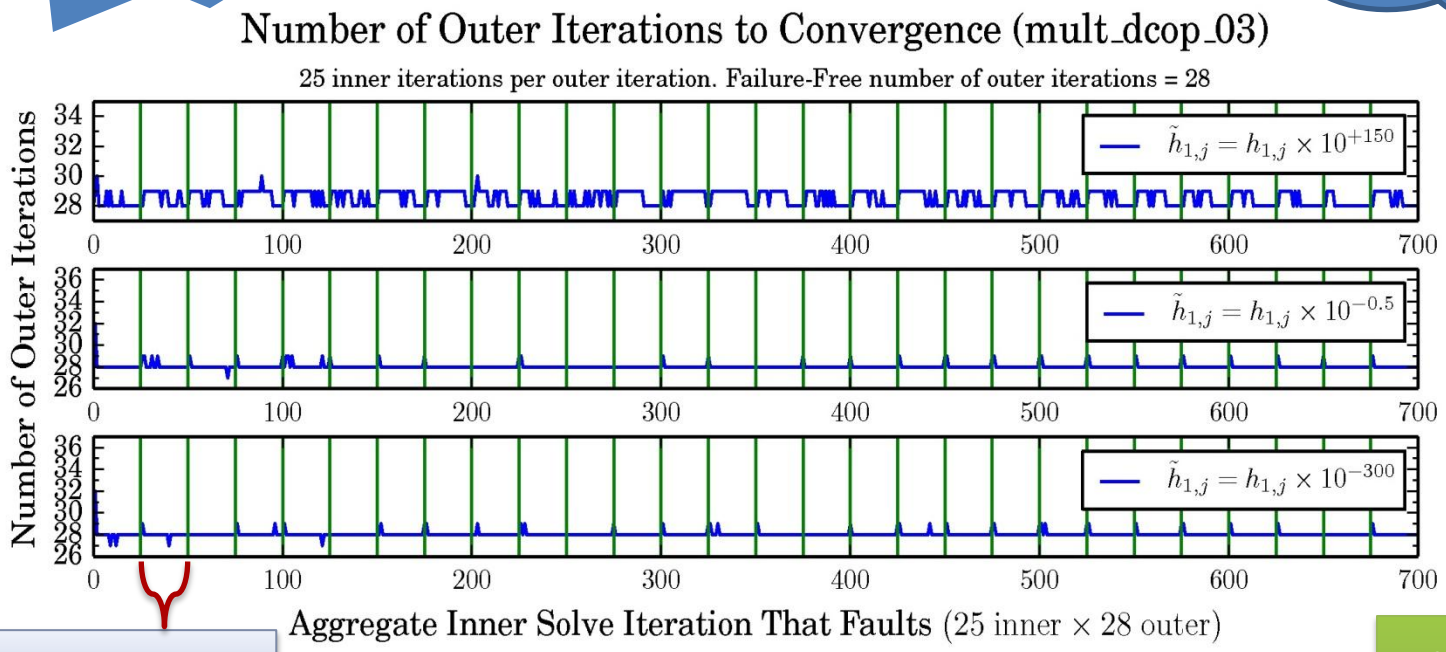
Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer
x-axis:
introducing single fault in x-th orthogonalization



Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Inject ***single*** fault
Study impact

700
experiments per
fault type



one inner solve

Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer
x-axis:
introducing single fault in x-th orthogonalization

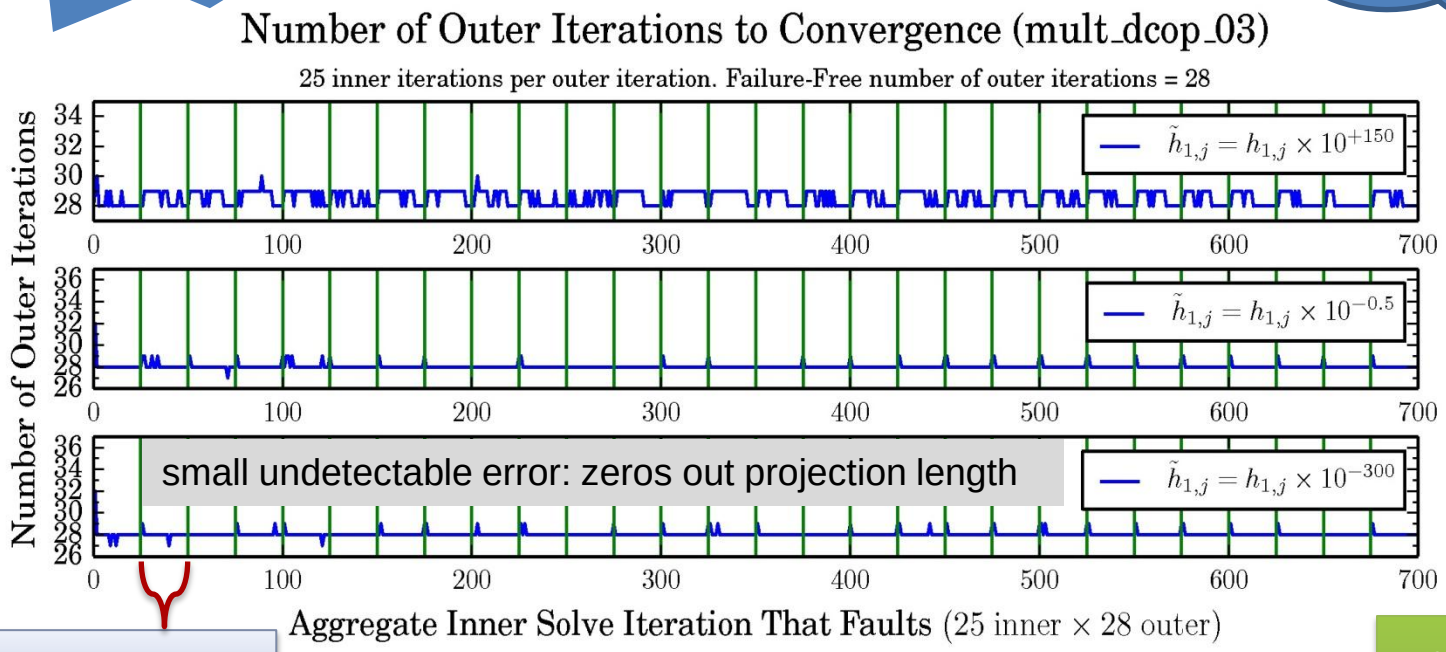
reliable
FGMRES

unreliable
GMRES

Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Inject ***single*** fault
Study impact

700
experiments per
fault type



one inner solve

Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer
x-axis:
introducing single fault in x-th orthogonalization

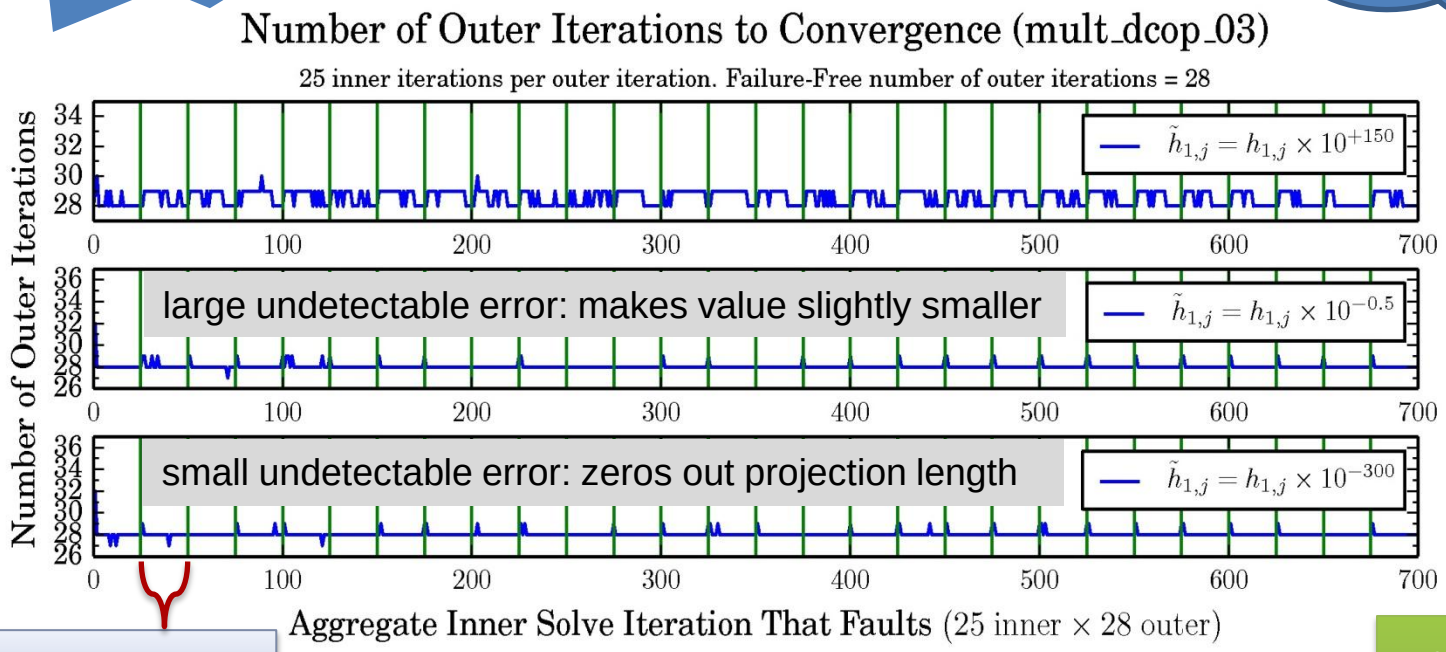
reliable
FGMRES

unreliable
GMRES

Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Inject ***single*** fault
Study impact

700
experiments per
fault type



one inner solve

Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer
x-axis:
introducing single fault in x-th orthogonalization

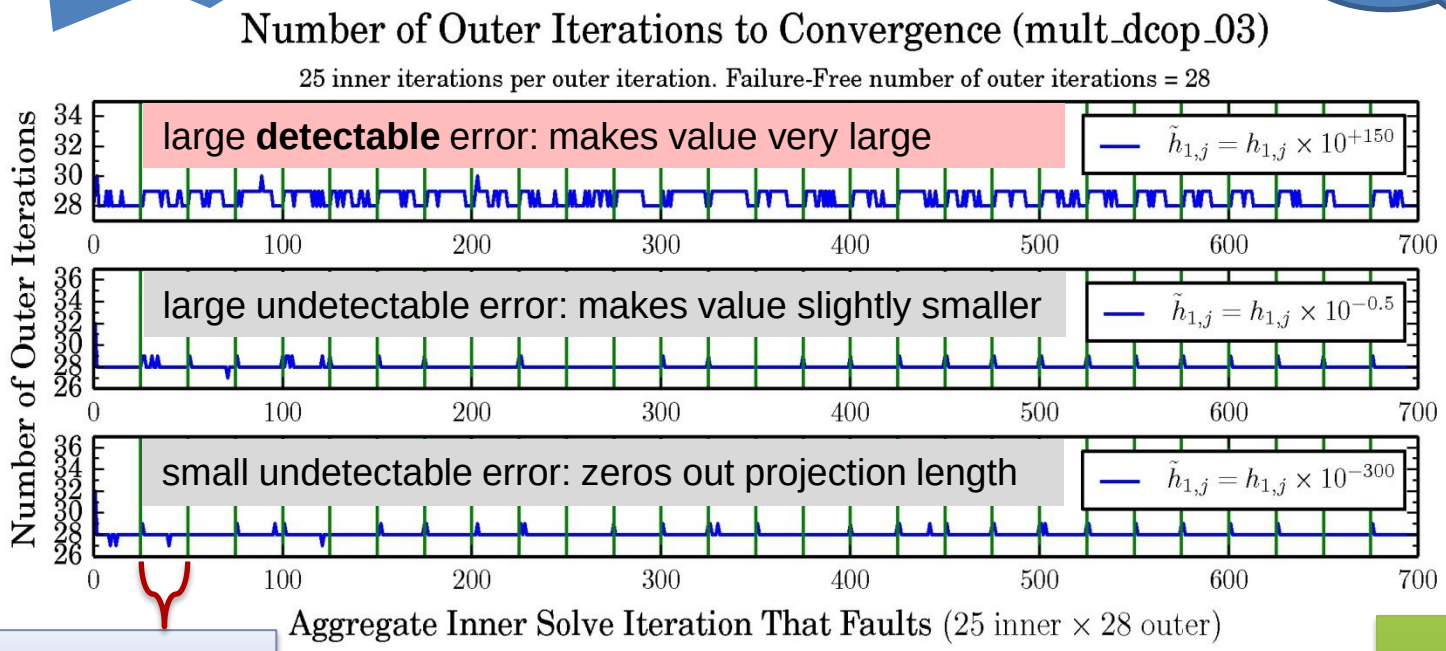
reliable
FGMRES

unreliable
GMRES

Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Inject ***single*** fault
Study impact

700
experiments per
fault type



one inner solve

Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer
x-axis:
introducing single fault in x-th orthogonalization

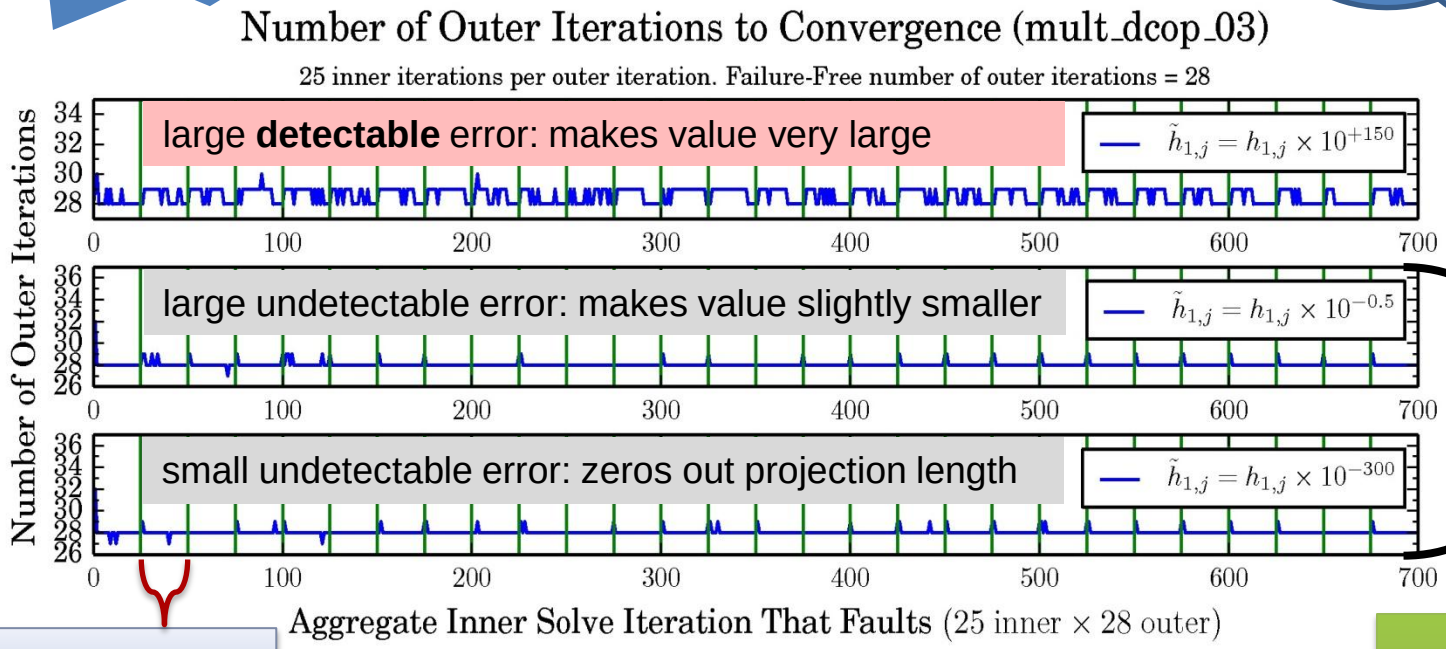
reliable
FGMRES

unreliable
GMRES

Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Inject ***single*** fault
Study impact

700
experiments per
fault type



Smaller than bound

one inner solve

Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer
x-axis:
introducing single fault in x-th orthogonalization

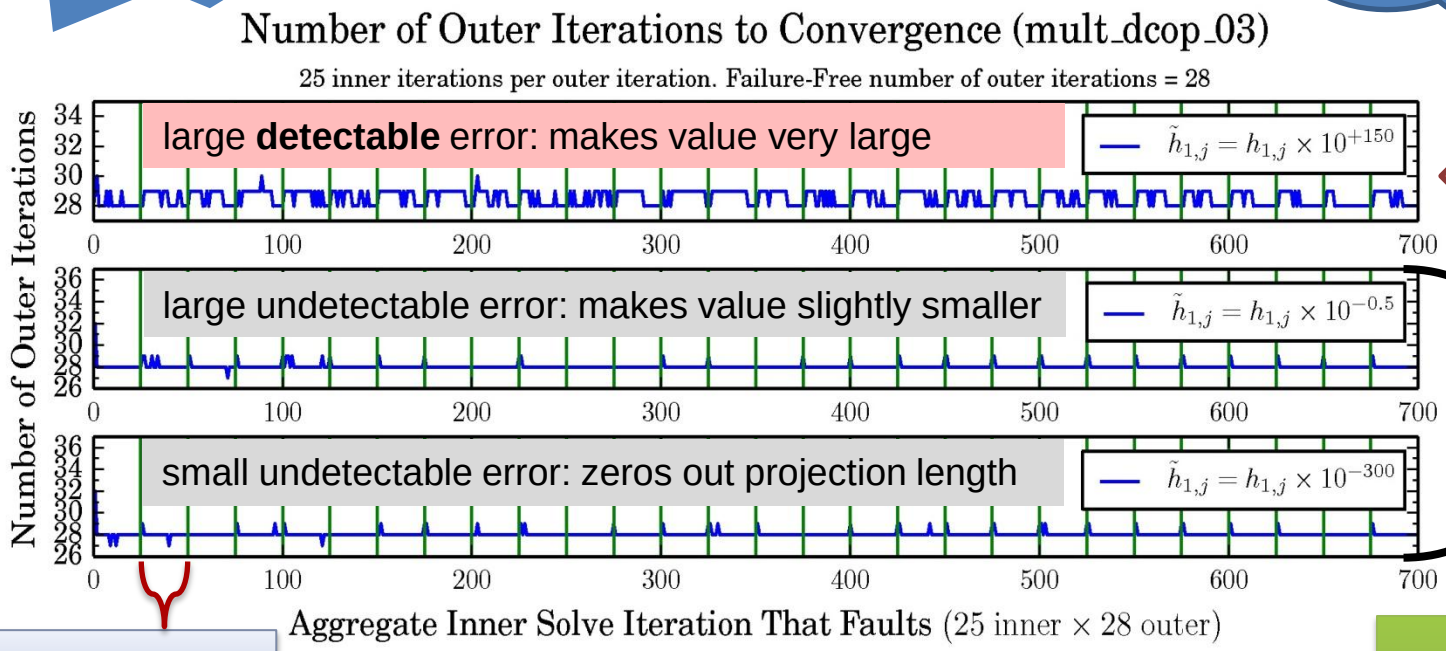
reliable FGMRES

unreliable GMRES

Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Inject ***single*** fault
Study impact

700
experiments per
fault type



Larger than bound

Smaller than bound

one inner solve

Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer
x-axis:
introducing single fault in x-th orthogonalization

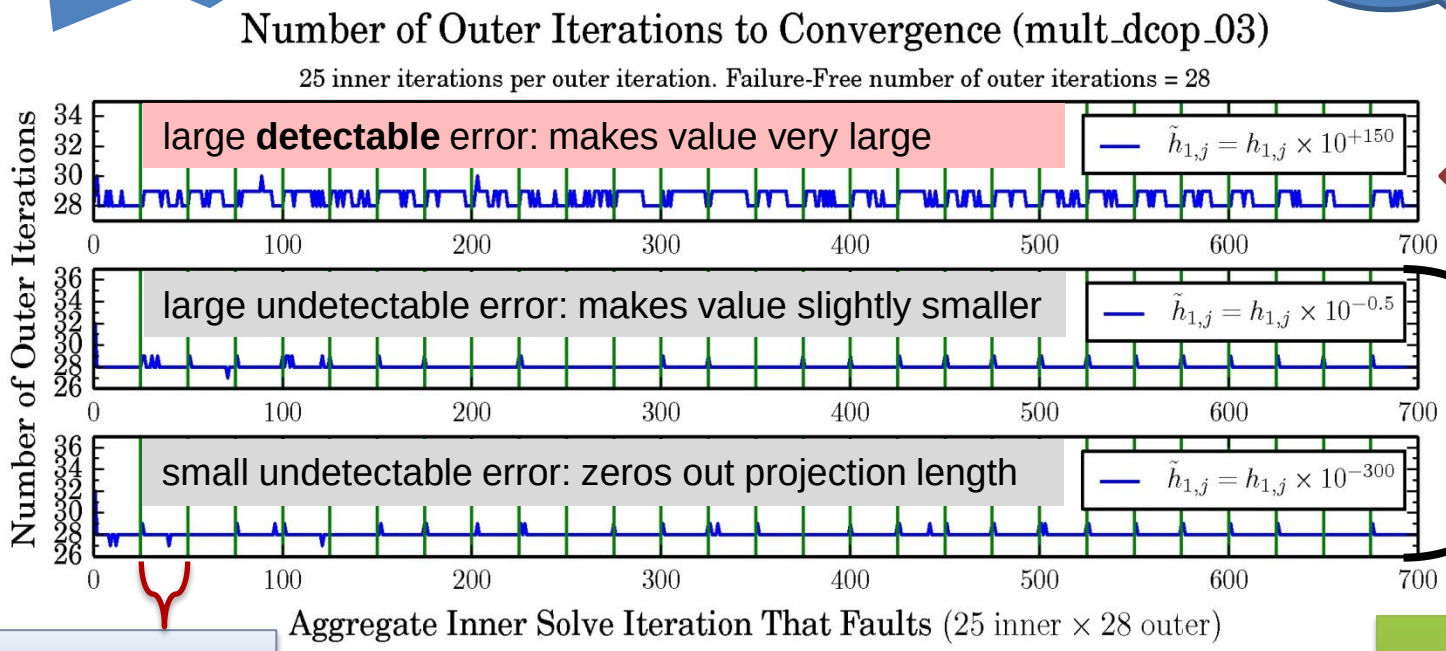
reliable FGMRES

unreliable GMRES

Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Inject ***single*** fault
Study impact

700
experiments per
fault type



Larger than bound

Smaller than bound

one inner solve

Failure Free: **700 orthgonalization calls**
28 outer iterations to converge
25 inner iterations per outer

reliable FGMRES

unbounded error = high variability in time to solution
bounded error = low variability, in time to solution

Prototype Design: Fault Tolerant GMRES iterative solver
Flexible GMRES (FGMRES) creates sandbox
Implemented using Trilinos

Conclusions

Sandia National Laboratories

- Enforcing bounds is cheap
- Enables numerical approaches to roll-through errors without “blowing up”

Future Work

- Link norm bound to Inexact Krylov convergence rate (push bounded faults back onto the matrix)
- Investigate link between reliable computations and redundancy (e.g., reorthogonalization vs redundant orthogonalization)

Questions/Comments: {jjellio,mhoemme}@sandia.gov mueller@cs.ncsu.edu

Papers: http://arxiv.org/find/cs/1/au:+Elliott_J

http://www4.ncsu.edu/~jjellio3/

Acknowledgements

This research was supported by the Consortium for Advanced Simulation of Light Water Reactors (http://www.casl.gov), an Energy Innovation Hub (http://www.energy.gov/hubs) for Modeling and Simulation of Nuclear Reactors under U.S. Department of Energy Contract No. DE-AC05-00OR22725.

This work was supported in part by grants from National Science Foundation (awards 1058779 and 0958311).

Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.