Fault-Tolerant Multigrid Algorithms Based on Error Confinement and Full Approximation

Mirco Altenbernd and Dominik Göddeke

SIAM PP '16
MS49: Resilience Towards Exascale Computing
April 2016

Institute for Applied Analysis and Numerical Simulation
Chair Computational Mathematics for Complex Simulations

mathematik ians fakultät8
SimTech Cluster of Excellence
University of Stuttgart Germany

The resilience challenge

Some important observations

- More components at exascale $\Rightarrow$ higher probability of failure
- Active debates to sacrifice reliability for energy efficiency
- MTBF $< 1\,\mathrm{h}$ $\Rightarrow$ all simulation software must be prepared

Classical techniques

- Reliability in hardware (ECC protection etc.) too power-hungry
- Checkpoint-restart too memory-intensive (and too slow)
- Triple modular redundancy too power-hungry, but: can be more energy-efficient and faster for large fault rates

Focus of this talk: ABFT for multigrid

General concept: algorithm-based fault tolerance

- Exploit algorithmic properties to detect and correct faults
- Can be more efficient than middleware

Previous work

- Scenario: Node loss
- Self-stabilisation properties

In this talk: Silent Data Corruption

- Scenario: Bitflips
- Black-box smoother protection

Resilient multigrid

Silent data corruption

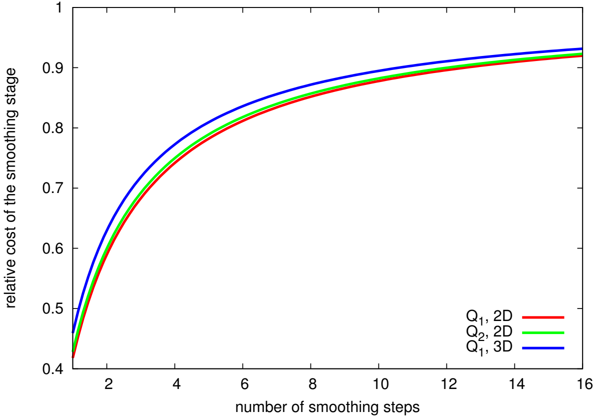
Silent data corruption

- Soft transient faults that lead to wrong solutions
- Sometimes noticeable a posteriori (divergence), mostly not
- Causes: radiation, leaking voltage, silicon ageing, ...

Core idea of our approach

- Use FAS multigrid to increase robustness
- Based on nonlinear MG: not just a correction on each level but a true approximation of the solution
- Linear case: numerically equivalent, less than one fine SPMV overhead per cycle
- One additional vector necessary

Focus on smoother stage



- Relative cost of the smoother stage as a function of the number of smoothing steps
- Detailed model: in the paper

Black-box smoother protection

Theoretical justification for the down-cycle

- Obvious: residual $\mathbf{r}$ converges to zero on finest grid
- Easy to prove: residual (monotonously) converges to zero on all grids

Theoretical justification for the up-cycle

- Slightly non-trivial proof: correction vector $\mathbf{c}$ converges (monotonously) to zero on all grids

Consequence: good fault indicators

- Both readily available without additional computation

Black-box smoother protection

Practical realisation after smoothing on level k

- Compute index set of possibly faulty components of $\mathbf{c}$ or $\mathbf{r}$ by comparing against level-specific threshold from earlier iteration
- Extend by one layer of indices coupled by $\mathbf{A}$
- Replace faulty components by unsmoothed values (down-cycle), or by recomputed correction from (non-faulty) coarser grid
- Adaptively update threshold

Black-box smoother protection

$\mathcal{L} = \texttt{detect\_and\_localise}(c, t_k^c)$ // detect faulty components

if $\mathcal{L} \neq \emptyset$ **then**

  $\tilde{v} = \mathbf{I}_k^{k-1} v^{(0)}$ // restrict (non-faulty) initial approximation

  $\tilde{c} = u_{k-1} - \tilde{v}$ // calculate new correction vector

  $\tilde{c} = \mathbf{P}_{k-1}^k \tilde{c}$ // prolongate new correction vector

  for $i \in \mathcal{L}$ **do**

    $c(i) = \tilde{c}(i)$ // replace faulty components

  end

  $t = \texttt{calc\_threshold}(c)$ // calculate new threshold value

  $q = t / t_k^c$ // relative reduction of threshold value

  if $q > 1$ // rescale other threshold values

  then

    for $i \in \{0, \dots, L\} \setminus \{k\}$ **do**

      $t_i^c = t_i^c \cdot 10 \cdot q$

    end

  end

  $t_k^c = t$ // store threshold value

else

  $t_k^c = \texttt{calc\_threshold}(c)$ // calculate and store threshold value

end

Checksum protection for transfer stage

Checksums

- Use identity $\mathbf{1}^\mathsf{T}(\mathbf{A}\mathbf{x}+\mathbf{y}) = (\mathbf{1}^\mathsf{T}\mathbf{A})\mathbf{x} + \mathbf{1}^\mathsf{T}\mathbf{y}$
- Precompute $\mathbf{1}^\mathsf{T}\mathbf{A}$ (column sums)
- Fault detection in $\mathbf{A}\mathbf{x}+\mathbf{y}$ by three dot products
- More elaborate schemes: detect and correct errors

Combined approach

- Black-box smoother protection, checksums for the rest
- Runtime comparison, fault-free case

	unprotected	**transfer stage** (checksums)	**smoothing stage** (new algorithm)	**FTMG** (both)
mean time	26.1562	27.4128	26.7905	27.9090
factor	1	1.0480	1.0243	1.0670

Numerical experiments

Test problems

- Different flavours of anisotropic diffusion-convection-reaction
- 2D, Q_1 FEM, eight levels, 263169 DOF

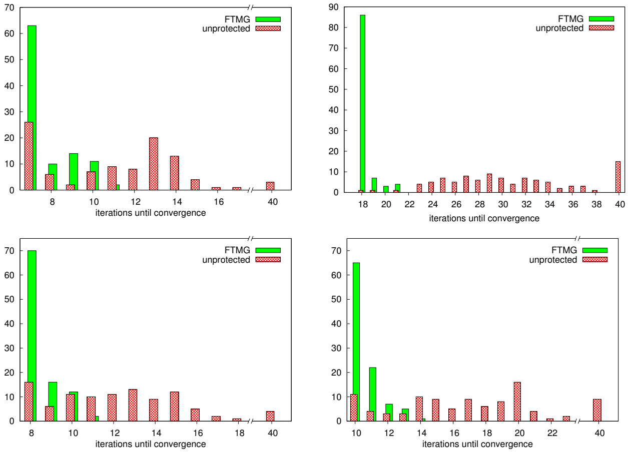
Solver configuration

- FAS-MG, V-cycle, 4+4 Jacobi $\omega = 0.7$ smoothing steps
- Bilinear interpolation for restriction and prolongation
- Natural injection for approximations

Fault injection

- 1 % bitflip probability in current approximation after each smoothing step on each level (deliberately unrealistically high)
- Actual component and bit fully random
- 100 repetitions per experiment

Numerical experiments



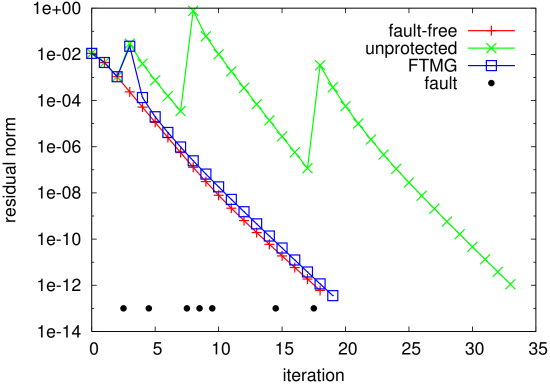
Histograms of iterations for poisson, anisotropic diffusion (andi), diffusion-convection (dico) and anisotropic diffusion-convection-reaction (andicore)

Numerical experiments (Summary)

		#detects		#iterations			
problem	**#faults**	**direct**	**delayed**	**fault-free**	**FTMG**	**unprotected**	
`poisson`	3.82	1.89	0.61	7	7.79	10.82	(3)
`andi`	10.24	4.94	0.49	18	18.25	29.09	(15)
`dico`	4.92	2.01	0.39	8	8.46	11.89	(4)
`andicore`	5.77	2.40	0.25	10	10.55	16.14	(9)

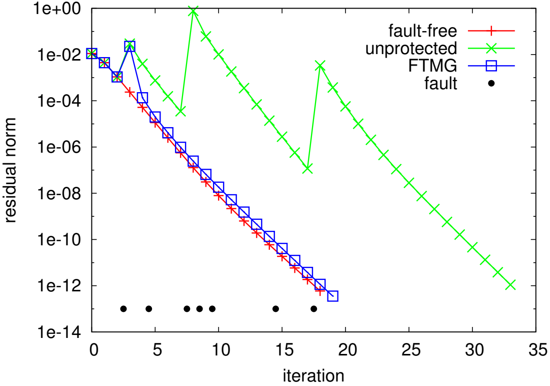
- Divergent runs in brackets
- Some faults not directly detected: delayed repair with ‘faultier’ correction

Detailed analysis of one andi test



- Blue bump: delayed repair with already faulty correction
- Two other significant faults: well repaired

Detailed analysis of one andi test



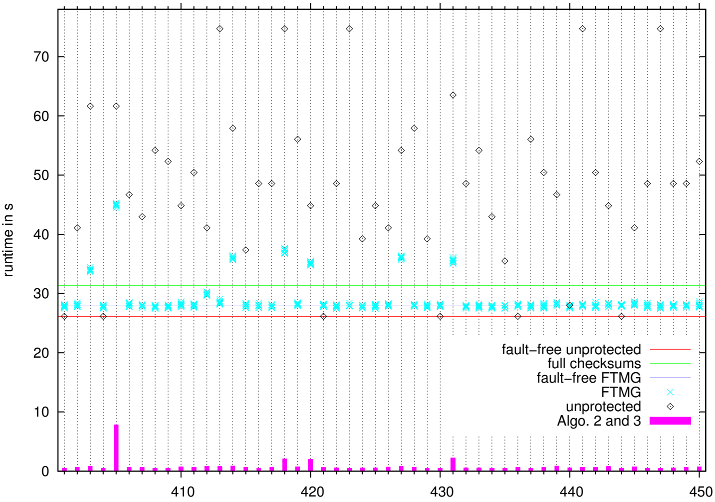
iter	cycle	level	step	change (%)	#det	#corr
3	up	6	1	9.99e+1	9	31
8	up	7	3	9.38e+1	25	49
18	down	7	4	1e+2	9	25

Detailed analysis of one andi test

iter	cycle	level	step	change (%)	#det	#corr
3	down	5	3	2.14e-4		
3	down	5	4	2.72e-11		
3	up	6	1	9.99e+1	9	31
5	up	5	3	5.55e-10		
8	down	7	3	1.81e-9		
8	up	7	3	9.38e+1	25	49
9	down	4	2	6.75e-3	49	81
9	up	1	2	4.47e-4	17	45
10	up	5	2	0		
15	up	1	1	1.25e-4	28	54
18	down	7	4	1e+2	9	25

- Irrelevant faults (small changes) are tolerated
- Small additional corrections (red)

Performance results



Summary

Summary and acknowledgements

- Fault tolerance becomes increasingly important
- Black-box ‘computer science’ techniques exist and work well, but: substantial overhead
- ABFT techniques may do better
- In this talk: some (early) ideas for multigrid

- Papers and more information
  - Silent data corruption: almost submitted
  - Minimised checkpointing: Parallel Comp. 49:117–135, 2015

DFG Priority Program 1648 *‘Software for Exascale Computing’*, grants GO 1758-2/1 and -2/2