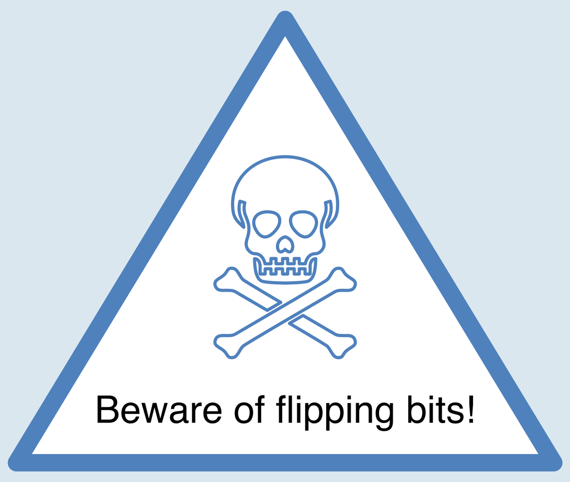
SPONSORED BY THE Federal Ministry of Education and Research

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, and Switzerland. This project also received funding from the German Federal Ministry of Education and Research (BMBF) grant 16HPC048.

Resilience in Spectral Deferred Corrections

Thomas Baumann[†‡], Sebastian Götschel[‡], Thibaut Lunet[‡], Daniel Ruprecht[‡], Ruth Schöbel[†], Robert Speck[†]

Resilience



Radiation can trigger current in transistor:

Main cause of faults: Bits can be flipped by solar or cosmic radiation!
Many other sources of faults: ageing hardware, voltage flashovers, …

Murphy's Law:
$P_{\text{failiure}} \propto N_{\text{potential failiure}}$
→ More parallelism ⟹ less reliability!

Error correction codes protect main memory, but caches remain exposed
In 2007 on largest machine at the time: **Uncorrectable faults in L1 cache about every 8h** [1]

Conventional generic resilience (checkpointing, replication) will not work on exascale machines!
Solution: Non-generic but more efficient algorithm-based fault-tolerance

Spectral Deferred Corrections (SDC)

Initial value problem in Picard form:
$$u(t) = u(t_0) + \int_{t_0}^{t} f(u(\tau))\, d\tau$$

Discretize with spectral quadrature:
$$\mathbf{u} = \mathbf{u}_0 + \Delta t Q F(\mathbf{u})$$

Preconditioned iteration with simpler (lower triangular) quadrature rule:
$$\left(I - \Delta t Q_\Delta F\right)\left(\mathbf{u}^{k+1}\right) = \mathbf{u}_0 + \Delta t\left(Q - Q_\Delta\right) F\left(\mathbf{u}^k\right)$$

Popular preconditioner LU-decomposition for stiff problems [2]:
$Q_\Delta = U^{\mathrm{T}}$ with $LU = Q^{\mathrm{T}}$

- Order can be equal to iteration count, depending on preconditioner
- Parallel-in-time extensions easy due to iterative nature
- Very malleable by choice of preconditioner(s)
- Iterating until a residual tolerance is reached is already a valid resilience strategy, but we can do even better!

Adaptivity

Estimate local error and adapt time step size to match preset tolerance as closely as possible

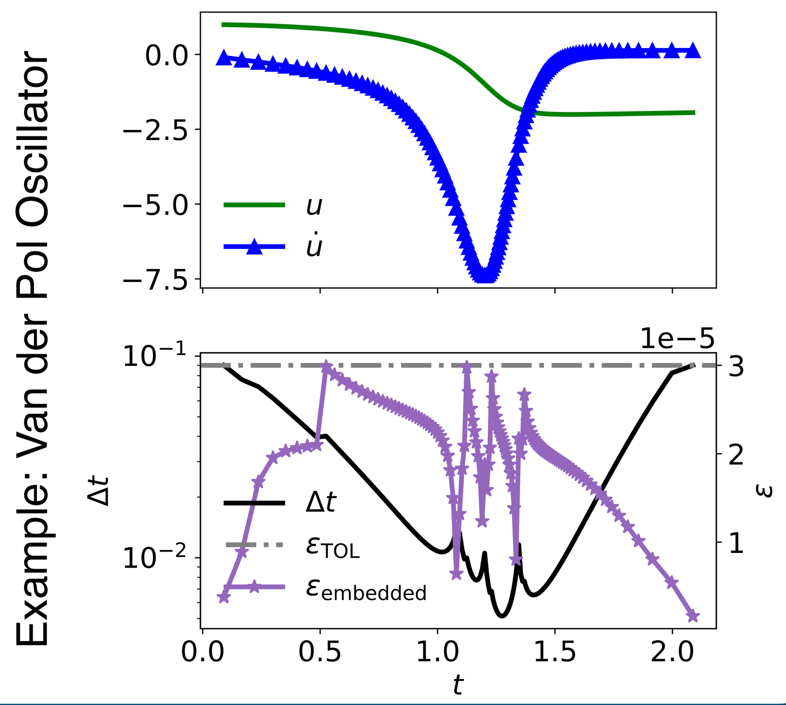
Scheme of order k at step n:
$$\frac{e_n'}{e_n} = \left(\frac{\Delta t_n'}{\Delta t_n}\right)^{k+1}$$

Now (falsely) assume that also
$$\frac{e_{n+1}}{e_n} = \left(\frac{\Delta t_{n+1}}{\Delta t_n}\right)^{k+1}$$

Replace: $e_{n+1} = \epsilon_{\mathrm{TOL}}$ and
estimate e_n with embedded method:
$\epsilon = \|u^{(k-1)} - u^{(k)}\|$
(superscript is iteration number)

Adaptivity step size update:
$$\Delta t_{n+1} = \beta \Delta t_n \left(\frac{\epsilon_{\mathrm{TOL}}}{\epsilon}\right)^{1/k}, \beta = 0.9$$

Fix faulty assumptions by only moving on to next step if $\epsilon \leq \epsilon_{\mathrm{TOL}}$.
These restarts give increased resilience!

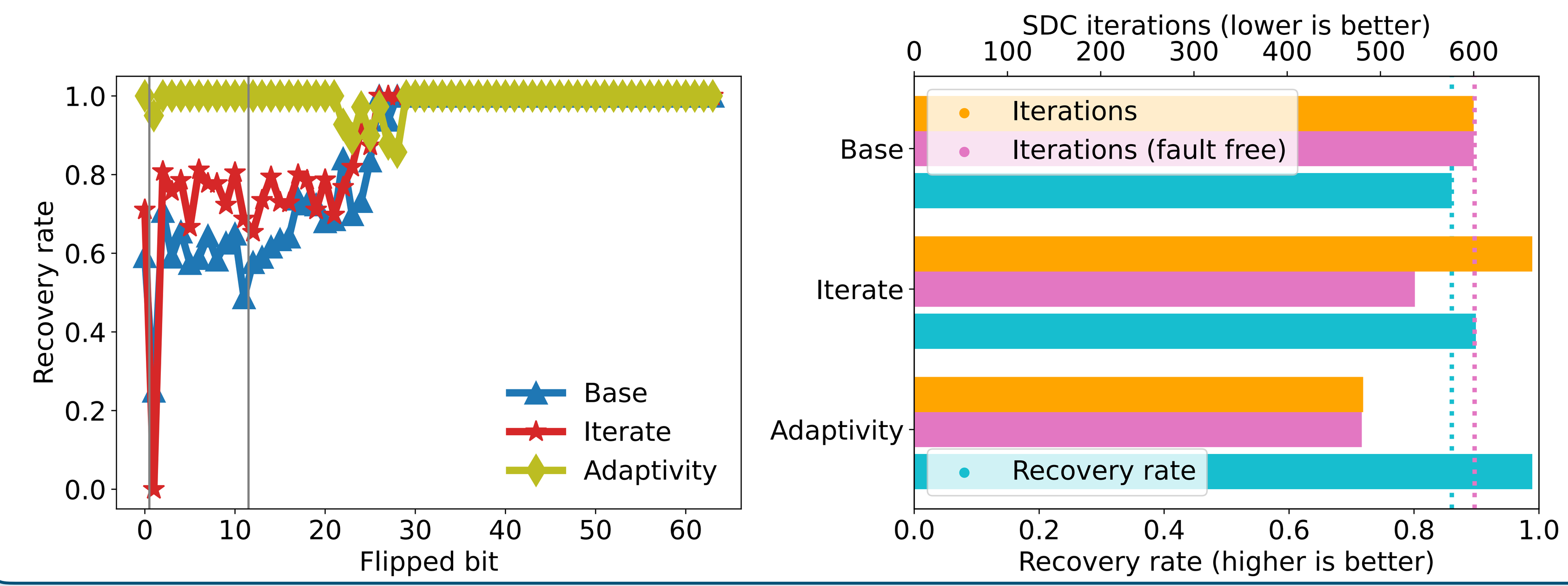


Resilience Statistics

Experiment:
- Flip random bit in the solution while solving Van der Pol problem
- Accept as recovered when final error is close to that of a fault free run

Schemes:
- Base: Fixed iteration count, fixed step size
- Iterate: Iterate until reaching residual tolerance, fixed step size
- Adaptivity: Fixed iteration count, continually adjust step size

Winner: Adaptivity: provides high resilience, while also boosting efficiency!

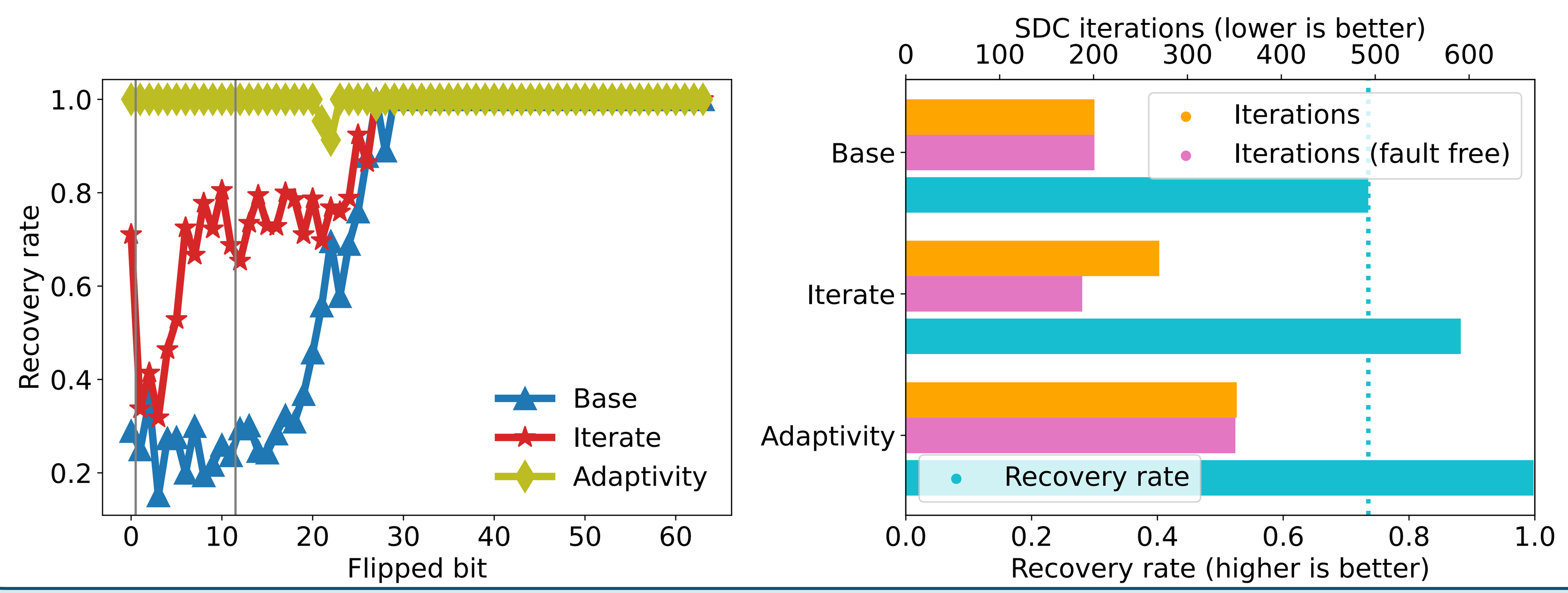


Diagonal Preconditioners

Diagonal preconditioners → Update quadrature nodes simultaneously in SDC iteration
⟹ **simple-to-implement small-scale time-parallelism**
Beware: Convergence properties with different preconditioners highly problem dependent! [3]

Experiment: van der Pol with diagonal preconditioner "MIN" from [3]:
Fewer iterations needed for convergence to same accuracy and parallelizable!

Same conclusions on resilience as with LU preconditioner:



Summary

Please keep faults in mind when developing code!
Do not despair: Iterative nature of SDC lends itself well to do unobtrusive resilience.
Non-generic and non-bulky resilience strategies can be implemented in your code as well!

Adaptivity is very promising!
+ Can boost resilience and efficiency at the same time, while being simple to implement
- Unclear how to extend this to parallel-in-time multi-step SDC, but various versions possible

Diagonal preconditioners are worth a try if you do SDC!
+ SDC iteration becomes embarrassingly parallel across the collocation nodes
- Convergence is highly problem dependent and not always properly mathematically founded

References

[1]: J. N. Glosli, D. F. Richards, K. J. Caspersen, R. E. Rudd, J. A. Gunnels and F. H. Streitz, "Extending stability beyond CPU millennium: a micron-scale atomistic simulation of Kelvin-Helmholtz instability," SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, 2007, pp. 1-11, doi: 10.1145/1362622.1362700.

[2]: Weiser, M. Faster SDC convergence on non-equidistant grids by DIRK sweeps. Bit Numer Math 55, 1219–1241 (2015). https://doi.org/10.1007/s10543-014-0540-y

[3]: Speck, R. Parallelizing spectral deferred corrections across the method. Comput. Visual Sci. 19, 75–83 (2018). https://doi.org/10.1007/s00791-018-0298-x

Contact: t.baumann@fz-juelich.de
[†]Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany. [‡]Hamburg University of Technology, Institute of Mathematics, Chair Computational Mathematics, Germany.

Member of the Helmholtz Association