MOX-Report No. 19/2021

FT-GCR: a fault-tolerant generalized conjugate residual elliptic solver

Gillard, M.; Benacchio, T.

FT-GCR: a fault-tolerant generalized conjugate residual elliptic solver

Mike Gillard[1], Tommaso Benacchio[2]

[5] Loughborough University, UK
`m.gillard@lboro.ac.uk`

[2] MOX – Modelling and Scientific Computing,
Dipartimento di Matematica, Politecnico di Milano, Milan, Italy
`tommaso.benacchio@polimi.it`

Abstract

With the steady advance of high performance computing systems featuring smaller and smaller hardware components, the systems and algorithms used for numerical simulations increasingly contend with disruptions caused by hardware failures and bit-levels misrepresentations of computing data. In numerical frameworks exploiting massive processing power, the solution of linear systems often represents the most computationally intensive component. Given the large amount of repeated operations involved, iterative solvers are particularly vulnerable to bit-flips.

A new method named FT-GCR is proposed here that supplies the preconditioned Generalized Conjugate Residual Krylov solver with detection of, and recovery from, soft faults. The algorithm tests on the monotonic decrease of the residual norm and, upon failure, restarts the iteration within the local Krylov space. Numerical experiments on the solution of an elliptic problem arising from a stationary flow over an isolated hill on the sphere show the skill of the method in addressing bit-flips on a range of grid sizes and data loss scenarios, with best returns and detection rates obtained for larger corruption events. The simplicity of the method makes it easily extendable to other solvers and an ideal candidate for algorithmic fault tolerance within integrated model resilience strategies.

1 Introduction

Numerical models used in the simulation of physical processes rely on dependable hardware as well as on software that is robust against bit-level faults. While both aspects have been taken for granted in the past decades, the emergence and exponential growth of available computing power on the same time frame has spurred a race towards more accurate and efficient simulations, with top high-performance computing facilities fast approaching 1 exaflop and millions of computing nodes [10]. A case in point is numerical weather prediction, where global atmospheric models currently run at resolutions only accessible to local area models a few years ago [4].

As cores multiply in number and shrink in size, distributed computing systems become more vulnerable to loss of resources caused by hardware failures, while algorithms are disrupted by increasingly frequent soft faults, spurious corruptions in their binary representation. Standard disk checkpointing and restart resilience strategies clearly cease to be viable at the point where then overhead associated with the backup procedure exceeds the mean time between failures [5].

At the end point of partially implicit time discretisation strategies for evolution problems, solvers for linear systems and associated preconditioners often represent an efficiency bottleneck in numerical models. The ability of linear solvers to detect, correct and recover from faults with minimal overhead therefore takes on paramount importance in efforts towards integrated model resilience [2]. In this context, Krylov solvers have been equipped with interpolation-restart mechanisms against hard faults without overhead in the case of no failures [1], and multigrid-based compressed checkpointing strategies [6] have also been proposed. A fault-tolerant version of the GMRES method (FT-GMRES) has been tested in the context of selected reliable and unreliable model sections [7, 9]. Ultimately, FT-GMRES both relies on an additional flexibility of the underlying adapted model (flexible GMRES) to adopt differing preconditioning and stopping criteria between successive Krylov subspace minimisation steps (outer iterations), and uses a bounded state function (Hessenberg matrix) to detect possible errors (soft-faults). Such options are not directly transferable to other Krylov subspace solvers, where the solution proposed here is generally compatible, but provides a much less robust detection mechanism than that of FT-GMRES.

This paper proposes a framework for algorithmic fault tolerance and local data recovery within iterative methods used for the solution of linear systems. A Fault-Tolerant Generalised Conjugate Residual (FT-GCR) elliptic solver is designed and implemented with the capability

3

to detect and recover from soft faults. A lightweight mechanism for the detection of soft faults in the form of bit flips within the GCR Krylov solver is introduced, based on a simple check exploiting the monotonic decrease of the solver residual norm at each iteration of the algorithm. In addition, the method is equipped with fault recovery that enables resuming operation following the fault.

Numerical experiments are carried out on the solution of an elliptic problem arising from the discretisation of steady potential flow past a hill on the sphere. A range of configurations are simulated, with spherical octahedral Gaussian grids up to O640 (corresponding to approximately 15 km horizontal resolution) running on up to 3240 MPI processes, and with various data loss values and injection probabilities of soft fault events, with between one in five to four in a million grid points affected by the soft fault. With the added fault-tolerance, the solver is able to recover effectively from soft faults, reaching convergence in fewer iterations compared with the unprotected solver version. In addition, higher detection rates and return on fault tolerance scores are found for larger corruption events. Finally, the paper contains an evaluation and discussion of the findings with an outlook to future development and applications.

2 The GCR and a Fault-Tolerant extension

We consider the following elliptic problem for the unknown ϕ:

$$\mathcal{L}(\phi) = \mathcal{R}, \tag{1}$$

where $\mathcal{L}$ denotes the elliptic operator and $\mathcal{R}$ the right-hand side [11, 12, 13].
The GCR elliptic solver belongs to a class of Krylov subspace methods, minimises the l_2 norm of the residual, r, of problem (1), and solves the kth-order damped oscillator equation with preconditioner operator $\mathcal{P}$ [15]:

$$\frac{\partial^k \mathcal{P}(\phi)}{\partial \tau^k} + \frac{1}{T_{k-1}(\tau)}\frac{\partial^{k-1}\mathcal{P}(\phi)}{\partial \tau^{k-1}} + ... + \frac{1}{T_1(\tau)}\frac{\partial^1 \mathcal{P}(\phi)}{\partial \tau^1} = \mathcal{L}(\phi) - \mathcal{R}. \tag{2}$$

The damped oscillator equation (2) is discretised in pseudo-time τ and optimal parameters $T_1, ..T_{k-1}$ are determined to assure the minimisation of $\langle r, r \rangle$ for the field ϕ, see [11, 13] for a detailed discussion. The solver employs bespoke left-preconditioning (see, e.g., [8]) to address the large condition number induced by the domain anisotropy for global atmospheric problems.

The iterative nature of the GCR method provides ample opportunity to recompute faulty data at minimal cost to the solver, given an ability to detect such faults. A basic detection method can be derived, given that for each iteration of GCR (n in Algorithm 1), the kth order dampening takes place on a Krylov subspace, $\mathcal{K}_n$, such that $\mathcal{K}_n \subseteq \mathcal{K}_{n+1}$, $\forall n \in 1, 2, \ldots$. As such, each iteration of GCR minimises the l_2 norm of the residual, r, on that Krylov subspace. Since the norm is non-increasing on each subspace, it is also non-increasing between subspaces. If during computation the discrete l_2 norm value increases, that most likely indicates a problem with the solver.

Algorithm 1 describes the fault-tolerant GCR (FT-GCR) method, including the backup and fault detection steps, following [11] for notation. FT-GCR detection and correction additions are shown in red. The 'last known good' solution, denoted $[\#]^{\star}$, is backed up at after a complete pass of GCR (n) where no faults were detected.

Resiliency testing of GCR(k) indicates that the method is quite resilient to soft fault events (e.g., bit flips), although such events often cause the solver to stagnate for an iteration. Most often the solver continues on, albeit often converging at a slow rate. This remains the case even for very large fault events (where many data points are corrupted). Fault events sometimes cause convergence issues at much later solver passes, and the response of the solver to fault events is difficult to predict.

Therefore, resilience is built into the GCR solver as follows. When a fault is detected, the GCR solver can be easily reverted to a state previously deemed good (i.e., no fault detected), at the cost of at most a full iteration of Algorithm 1 (if a fault occurs during the $\nu = k-1$ pass). The backup of the known good configuration occurs following the exit check and fault test, at the start of a new Krylov subspace loop (i.e., backing up the output of the full k passes on the last Krylov subspace). This ensures the backup solution underwent the full iterative process for a given subspace – backing up a known good configuration for $\nu < k$ would be akin to varying the order of the damped oscillator (2) between outer GCR iterations - which can cause some instability to the overall solution.

3 Test problem and numerical results

We consider the case of potential flow $\mathbf{v}$ over a Gaussian-shaped hill, with governing equations:

Algorithm 1 FT-GCR(k) (including red section) - GCR(k) (excluding red section)

For any initial guess, ϕ^0, set $r^0 = \mathcal{L}\left(\phi^0\right) - \mathcal{R}$, $p^0 = \mathcal{P}^{-1}\left(r^0\right)$; then iterate:
for $n = 1, 2, ...$ until convergence **do**
    for $\nu = 0, ..., k-1$ **do**
        $\beta = \dfrac{\langle r^\nu \mathcal{L}\left(p^\nu\right)\rangle}{\langle \mathcal{L}\left(p^\nu\right) \mathcal{L}\left(p^\nu\right)\rangle}$
        $\phi^{\nu+1} = \phi^\nu + \beta p^\nu$
        $r^{\nu+1} = r^\nu + \beta \mathcal{L}\left(p^\nu\right)$
        if $\|r^{\nu+1}\|_2 \le \epsilon$ **then**
            exit
        end if
        if $r^{\nu+1} \ge r^\nu$ **then**
            $n = n - 1$
            reset $[\phi, r, p, \mathcal{L}(p)]^0$ to $[\phi, r, p, \mathcal{L}(p)]^*$
        else if $\nu = 0$ **then**
            set $[\phi, r, p, \mathcal{L}(p)]^*$ to $[\phi, r, p, \mathcal{L}(p)]^0$
        end if
        $e = \mathcal{P}^{-1}\left(r^{\nu+1}\right)$
        Compute $\mathcal{L}(e)$
        for $l = 0, .. \nu$ **do**
            $\alpha_l = \dfrac{\langle \mathcal{L}(e) \mathcal{L}\left(p^l\right)\rangle}{\langle \mathcal{L}\left(p^l\right) \mathcal{L}\left(p^l\right)\rangle}$
            $p^{\nu+1} = e + \sum_l^\nu \alpha_l p^l$
            $\mathcal{L}\left(p^{\nu+1}\right) = \mathcal{L}(e) + \sum_l^\nu \alpha_l \mathcal{L}\left(p^l\right)$
        end for
    end for
    reset $[\phi, r, p, \mathcal{L}(p)]^k$ to $[\phi, r, p, \mathcal{L}(p)]^0$
end for

$$\mathbf{v} = \mathbf{v_a} - \nabla\phi$$
$$\nabla \cdot (\rho\mathbf{v}) = 0 \tag{3}$$

where $\mathbf{v_a}$ is the ambient velocity and ρ is the prescribed density. The problem is cast in non-orthogonal, terrain-following coordinates, whereby the vertical coordinate is adjusted to the hill profile, depending on the vertical extension of the domain H and bottom topography h. The terrain-following reference frame involves metric terms, see [14] for details.

Model (3) is then discretized in space using a finite volume method in the horizontal and a finite difference method in the vertical. The underlying Octahedral global meshes are generated using the Atlas

library [3]. Combining the transformed equations brings to an elliptic problem of the form (1), where $\mathcal{L}$ is a discrete Poisson operator, which is then solved using the fault-tolerant preconditioned GCR(k) (Algorithm 1). For further details about the system of equations, the GCR solver, and the finite volume discretisation used here, see, e.g., [8, 11, 12, 13, 14].

The behaviour of FT-GCR in solving the elliptic problem (1) is investigated to determine the response of the fault-tolerant solver at scale. MPI-enabled runs are carried out for a range of grid resolutions with suitable core counts and tolerances for the residual l_2 norm-based solver exit criterion, and for 0.0004%, 0.04%, 0.2%, 1%, 5%, and 20% total data loss (Table 1). For the simulations in this paper, a purely horizontal ambient velocity with $\mathbf{v_{a,x}} = 20\,\text{m/s}$ is used, and the values $H = 40800\,\text{m}$, $h = 4000\,\text{m}$ and $h_r = 3E05\,\text{m}$ for domain height and maximum hill height and hill radius are considered.

Table 1: Approximate horizontal grid resolutions at the Equator Δx_{EQ}, tolerance `tol` on the l_2 norm of the residual, number of total grid points, number of MPI processes and grid points per process for the numerical experiments with Octahedral grids ON considered in this paper. A fixed number of 51 vertical levels is used throughout the simulations.

ON	O40	O80	O160	O320	O640
Δx_{EQ} [km]	227	119	61	31	15.5
`tol`	E-14	E-12	E-10	E-8	E-6
Points	399840	1452480	5516160	21477120	84733440
MPI procs	36	108	216	864	3240
Points/proc	1110	13448	25538	24858	26152

Since the simulations are distributed over many computational processes, faults are set to occur on a single MPI process per fault injection event. The faulty process is determined randomly, and the procedure is re-randomised following each injection event. Injection occurs after the preconditioning stage with implausibly high probabilities of either 2% or 5%. A maximum of 10 individual fault events are allowed. The fault causes a given number of entries of $e = P^{-1}\left(r^{\nu+1}\right)$, the preconditioner output, to suffer a bit flip. The amount of data entries corrupted during each fault event is varied between runs, with the number of data values corrupted by each event given as a percentage of the total number of array entries.

Figure 1 demonstrates results for the runs with O80 grid, 2% fault injection probability, see also Table 2 in the Appendix A for results with the range of grids O40 to O640. Each output dataset features results from around 100 protected runs and 100 unprotected runs, both batches using the same parameters, but with fully randomised faults

(the seeds of both fault events and the data values corrupted are unique for each run[1]). Baseline convergence of an un-corrupted simulation is also established by running without fault injection.

In evaluating the results, it should first be noted that if all faults are detected by FT-GCR, convergence will match the baseline. This does not mean that there is no computational cost to the detection and correction of such faults – any calculations performed between the backup of a known good configuration and the detection of a fault are wasted. These calculations amount to, at most, one full GCR iteration (denoted by n in Algorithm 1) per fault. In addition, undetected faults may not always have an appreciable effect on the convergence. Upon examination of individual runs, early faults – those injected in the first few iterations – often go undetected, the reason being the sharp reduction in solution error in the first iterations of the elliptic solver. Therefore, errors caused by early faults are usually hidden by the overall reduction in the residual norm, yet those same faults often appear to have more significant implications to stability than faults injected towards the end of the simulation, where the error is much smoother. Late faults also tend to be detected more often, and the solver seems to fare better when the general solution error is already both small and smooth.

For small fault events (e.g., at 0.0004% data loss), faults are both difficult to detect and have a low impact on convergence, indicating that at least for this test case GCR is highly resilient to soft faults. In fact, even for very implausibly large fault events (e.g. 5% or 20%) convergence is only delayed by at most $\sim 20\%$. These large corruption events are, in effect, altering the values of every data entry in the entire array on a given process. That this type of event does not break the overall solver – even occasionally – is quite remarkable. The results also indicate that detection of data corruption becomes increasingly more difficult as the number of grid nodes increases (even when corruption events are scaled by this number).

---

[1]Note that actual data presented will be less due to a $\sim 20\%$ share of non-faulty runs, which are disregarded
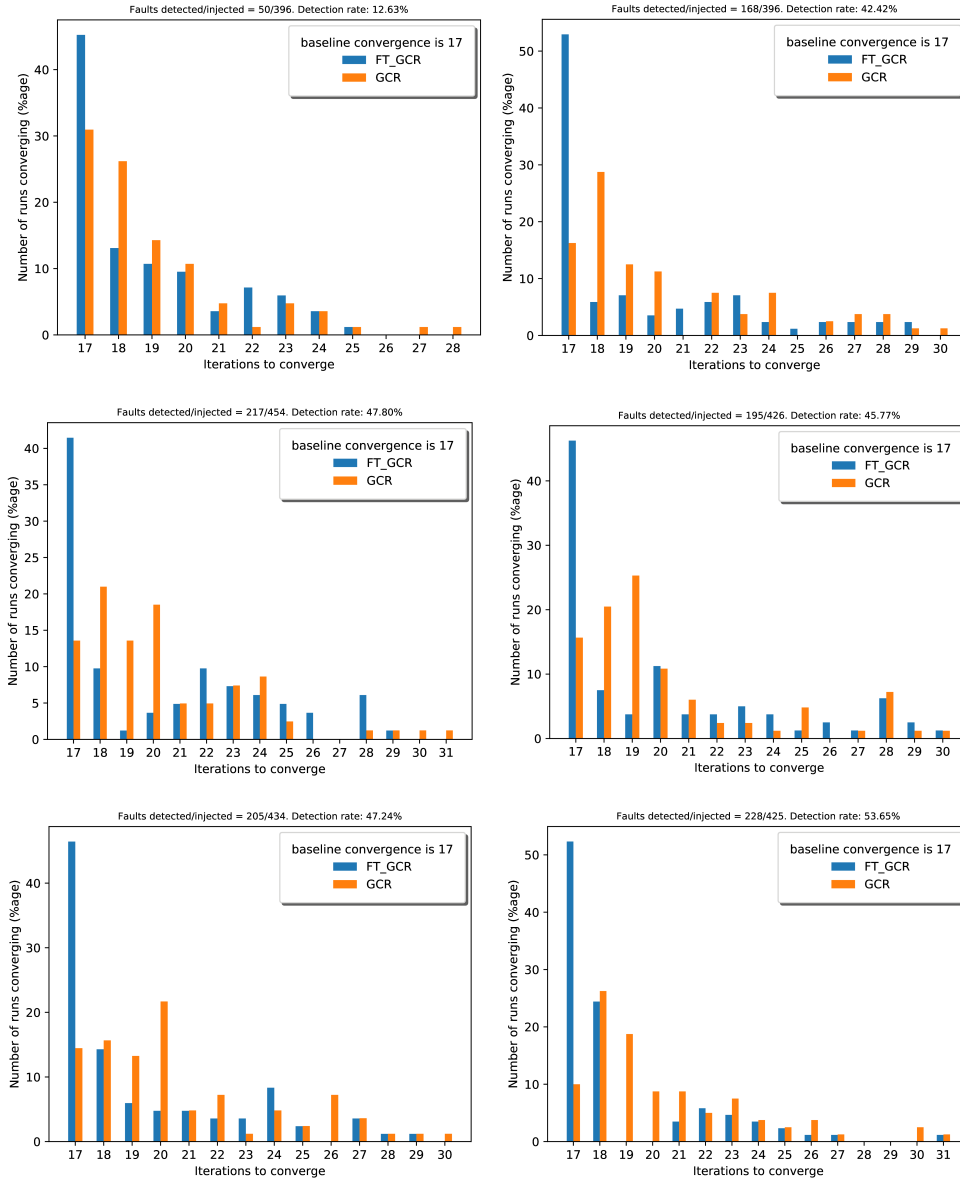
Figure 1: Numerical behaviour of GCR and FT-GCR when faults are introduced, O80 grid, 72 MPI tasks, size 5 Krylov space, 2% fault injection probability. Each histogram displays the percentage of runs that converged at a given iteration. Top left: 0.0004% data loss, top right: 0.04% data loss, middle left: 0.2% data loss, middle right: 1% data loss, bottom left: 5% data loss, bottom right: 20% data loss.
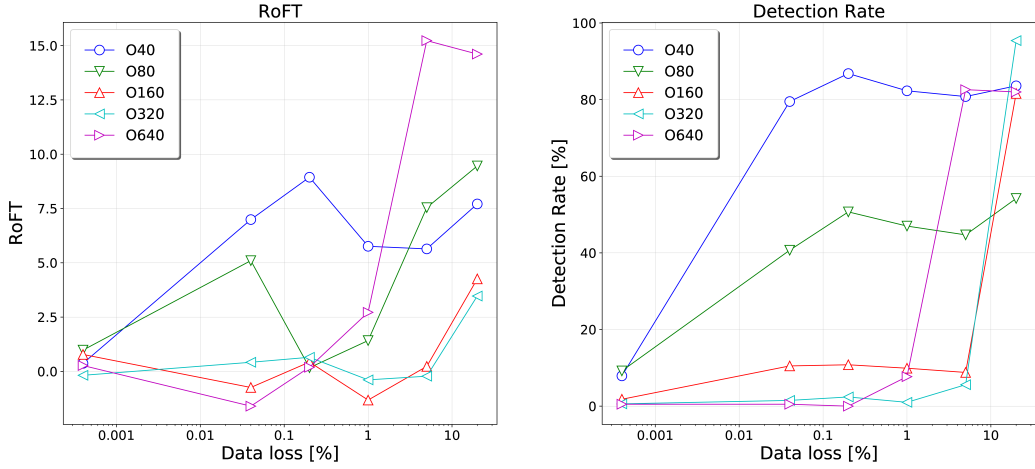
Figure 2: Return on Fault Tolerance score (RoFT, left) and FT-GCR Detection rate (right) as a function of data loss for the tests in Table 2.

Figure 2 displays the relationship between the size of a fault event (labelled as data loss), the improvement in average convergence (left), and the detection rate of faults (right). The Return on Fault Tolerance (RoFT) score for a grid and percentage of data lost is computed as:

$$\text{RoFT} = \frac{\text{GCR} - \text{FT-GCR}}{\text{baseline}} \qquad (4)$$

where 'GCR' and 'FT-GCR' denote the average number of iterations to convergence for unprotected runs and protected runs, respectively, normalised by the baseline convergence. The RoFT value intends to represent the savings in convergence delay when the model is protected using the FT-GCR algorithm compared to the basic GCR algorithm. As noted above, the benefits of FT-GCR tend to strongly depend on the overall size of individual fault events. Larger faults are better detected, and detection of smaller faults is improved on smaller grids.

Figure 3 demonstrates the relationship between detection rates and improved convergence. This relationship is not directly linear, but the overall trend is that better detection has a positive effect on convergence. The data towards the bottom left of the plot is less significant, since at low detection rates the unknown effect of a fault on convergence is difficult to quantify. The expectation is that more impactful faults will be more often detected, while the less impactful faults are more likely to be missed. Limited testing indicates this is not a reliable assumption, for example faults early in the run are detected with lower data loss values.

In addition, increasing the number of faults per run using a 5% fault injection probability both increases the convergence delay of
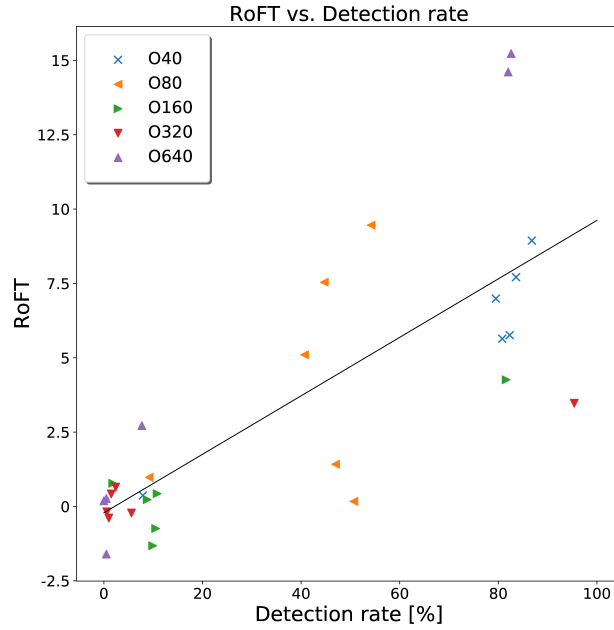
10

Figure 3: Scatter plot of Return on Fault Tolerance (RoFT) and FT-GCR Detection rate values and linear fit ($R^2 = 0.608$) for the tests in Table 2.

unprotected runs and increases the detection rate of faults (Table 3 in Appendix A). Figure 4 displays the corresponding RoFT and detection rates vs. the relative size of the fault event for the simulations with the O160 and O320 grids, contrasting the runs at 2% and 5% fault injection probability. In both cases, FT-GCR provides an improvement in performance as the number of fault events increases, with a more pronounced effect seen for very large fault events (5% and 20% data loss values).

It is clear that when most faults are detected, FT-GCR provides a noticeable improvement in convergence characteristics in comparison to unprotected GCR simulations. Whenever this is not the case, the impact of fault tolerance is more limited.
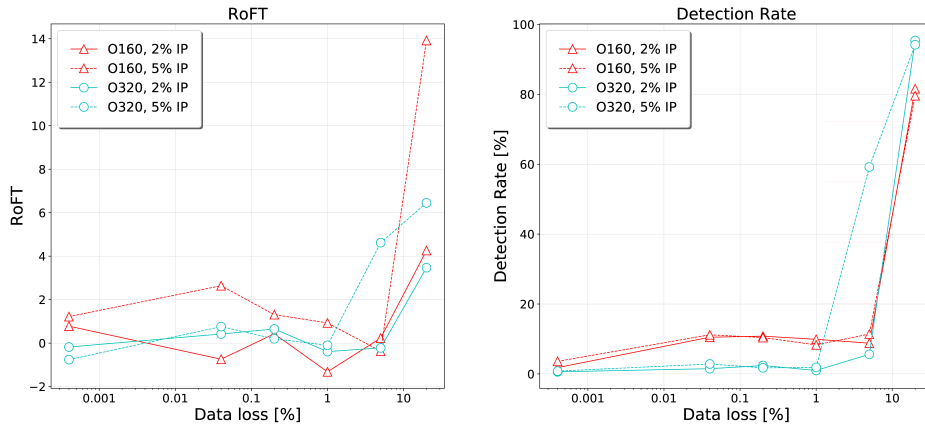
Figure 4: Return on Fault Tolerance score (RoFT, left) and FT-GCR Detection rate (right) as a function of data loss, O160 and O320 grids, 2% vs. 5% fault injection probability sensitivity tests in Table 3.

4   Discussion and Conclusion

We have introduced FT-GCR, a fault-tolerant version of the precon-ditioned Generalized Conjugate Residual elliptic solver. The fault detection mechanism of FT-GCR is based on a simple check on the residual norm decrease at each iteration. Once a fault is detected, recovery from the fault is based on a restart of the computation within the current Krylov subspace. Although the detection process uses a quantity whose evaluation requires global communications, it is also a quantity that is already computed. The backup of a known-good configuration increases storage requirements of the GCR implemen-tation by a small amount – 4 out of 14 full mesh arrays are backed up, corresponding to $\approx 30\%$ memory overhead for the GCR routine. Implementation within an existing GCR solver is therefore very simple, requiring only the addition of a backup loop, a fault detection subrou-tine, and a restart loop. The main workings of the algorithm remain unchanged.

The modified solver has been tested against an unmodified version on the solution of steady potential flow over a hill on the planet using the Octahedral Gaussian grid at horizontal resolutions ranging from 278 km to 15.5 km - therefore up to about a half of the current operational resolution of state-of-the-art medium-range weather forecast models - and considering a variety of choices for fault parameters such as percentage of data corrupted and chance of fault injection.

Parallel simulations with up to 3240 MPI processes have validated the capabilities of the method in protecting the solver's performance

when faults in the form of bit flips are introduced. The rate of detection of faults by FT-GCR has been shown to grow with the percentage of data corrupted, and when the fault-tolerant version is used, runs converge in fewer iterations than with standard GCR in the vast majority of cases. The return on fault tolerance is larger for higher detection rates, and both parameters increase with increasing resolution, an effect best seen with the finest grids and largest proportions of data corrupted by the faults.

Besides the overall picture of a positive effect of the modifications introduced for algorithmic resilience, the experiments have also highlighted a substantial degree of inherent robustness in the standard version of the GCR method. Even with massive fault injections, most GCR runs displayed iteration counts very close both to the faultless baseline run and to the FT-GCR run. Indeed no situations have been found where faults caused permanent stagnation of runs. It should also be noted that the variation of FT-GCR and GCR plots, as well as the negative values for the return parameter, at low detection rates are a result of the variability of runs. It would likely require thousands of additional data points to properly display the small improvement brought by low detection rates.

The tests presented here have been performed across a range of resolutions for demonstration purposes. The convergence acceleration and detection rate associated with FT-GCR were most pronounced at very large values of data loss, while the skill scores of the method for smaller, more realistic values of data loss seem less conclusive in the present version. Therefore, as currently implemented the method would be of limited use in modelling scenarios where it is important to achieve large detection rates even with very small corruption events. At the same time, the low computational cost and ease of implementation within a standard Krylov method make FT-GCR a useful tool in numerical frameworks where such solvers can take up a significant portion of wallclock time. In the context of parallel runs using FT-GCR as linear solver, for example within semi-implicit time discretizations in computational models of fluid flow, the fault tolerance against bit flips provided by a method such as the one presented here would ideally be complemented with recovery from node failures and systems resilience in an overall strategy as discussed in [2, 5].

Acknowledgments

A   Complete results for the runs

Tables 2 and 3 contain the complete results of the tests performed with the FT-GCR fault tolerant solver to produce Figures 2, 3 and 4 in the main text.

Table 2: Average number over all runs of faults per run and faults detected, detection rate for FT-GCR, average iterations for FT-GCR, average iterations for unprotected GCR, and Return on Fault Tolerance for each run with the grids and data loss values of Table 1, 2% fault injection probability. In the baseline configuration, convergence is reached in $\{19, 17, 17, 21, 21\}$ iterations for the $\{O40, O80, O160, O320, O640\}$ grids.

Grid	% data loss	Faults per run	Faults detected	Detection Rate [%]	Convergence FT-GCR	Convergence GCR	RoFT [%]
O40	0.0004	2.56	0.17	7.9	19.1	19.16	0.37
	0.04	2.80	2.20	79.5	19.15	20.48	6.99
	0.2	2.94	2.43	86.8	19.17	20.87	8.94
	1.0	3.02	2.49	82.3	19.28	20.37	5.76
	5.0	2.58	2.11	80.8	19.23	20.31	5.64
	20.0	2.89	2.45	83.6	19.15	20.62	7.71
O80	0.0004	2.36	0.3	9.2	18.83	19	0.98
	0.04	2.4	1.02	40.7	19.46	20.33	5.1
	0.2	2.79	1.33	50.7	20.32	20.35	0.17
	1.0	2.61	1.2	47	19.99	20.23	1.42
	5.0	2.6	1.23	44.7	19.39	20.67	7.54
	20.0	2.56	1.37	54.2	18.77	20.38	9.46
O160	0.0004	2.15	0.06	1.8	17.80	17.94	0.78
	0.04	2.27	0.18	10.5	17.93	17.8	-0.74
	0.2	2.04	0.21	10.8	17.55	17.62	0.43
	1.0	2.23	0.2	9.9	17.59	17.36	-1.32
	5.0	2.15	0.19	8.8	17.74	17.78	0.23
	20.0	2.19	1.72	81.6	17.4	18.13	4.26
O320	0.0004	2.23	0.01	0.6	21.08	21.04	-0.18
	0.04	2.28	0.04	1.5	21	21.09	0.42
	0.2	2.48	0.05	2.4	21.1	21.23	0.65
	1.0	2.37	0.02	1	21.09	21.01	-0.39
	5.0	2.36	0.1	5.6	21.05	21.01	-0.22
	20.0	2.58	2.5	95.4	20.99	21.72	3.47
O640	0.0004	2.33	0.02	0.5	21.13	21.18	0.27
	0.04	2.4	0.01	0.5	21.44	21.10	-1.6
	0.2	2.26	0.02	0	21.37	21.41	0.2
	1.0	2.81	0.28	7.7	23.86	24.43	2.72
	5.0	2.70	2.26	82.6	22.1	25.29	15.23
	20.0	2.42	2.1	82	22.39	25.46	14.61

Table 3: As in Table 2 but for the O160, O320 grids only and with 5% fault injection probability. In the baseline configuration, convergence is reached in $\{17, 21\}$ iterations for the $\{O160, O320\}$ grids.

Grid	% data loss	Faults per run	Faults detected	Detection Rate [%]	Convergence FT-GCR	Convergence GCR	RoFT [%]
O160	0.0004	4.73	0.14	3.6	17.97	18.18	1.22
	0.04	4.61	0.51	11.2	18.15	18.6	2.64
	0.2	4.71	0.47	10.4	18.79	19.01	1.31
	1.0	4.71	0.52	8.3	18.25	18.41	0.93
	5.0	4.6	0.57	11.4	18.41	18.35	-0.37
	20.0	5.36	4.14	79.6	17.91	20.27	13.92
O320	0.0004	4.86	0.04	0.8	21.23	21.07	-0.76
	0.04	4.73	0.1	2.84	21.13	21.29	0.76
	0.2	5.35	0.09	1.74	21.12	21.16	0.2
	1.0	5.13	0.09	1.86	21.2	21.18	-0.1
	5.0	5.61	3.16	59.21	21.42	22.39	4.62
	20.0	5.74	5.49	94.18	21.02	22.37	6.45

Note

References

[1] E. Agullo, L. Giraud, A. Guermouche, J. Roman, and M. Zounon. Numerical recovery strategies for parallel resilient Krylov linear solvers. *Numerical Linear Algebra with Applications*, 23(5):888–905, 2016.

[2] T. Benacchio, L. Bonaventura, M. Altenbernd, C. D. Cantwell, P. D. Düben, M. Gillard, L. Giraud, Göddeke D., E. Raffin, K. Teranishi, and N. P. Wedi. Resilience and fault tolerance in high-performance computing for numerical weather and climate prediction. *International Journal of High Performance Computing Applications*, in press, 2021. doi:https://doi.org/10.1177/1094342021990433.

[3] W. Deconinck, P. Bauer, M. Diamantakis, M. Hamrud, C. Kühnlein, P. Maciel, G. Mengaldo, T. Quintino, B. Raoult, P. K. Smolarkiewicz, and N. P. Wedi. Atlas : A library for numerical weather prediction and

climate modelling. *Computer Physics Communications*, 220:188–204, 2017.

[4] P. D. Düben, N. P. Wedi, C. Zeman, and S. Saarinen. Global simulations of the atmosphere at 1.45 km grid-spacing with the Integrated Forecasting System. *Journal of the Meteorological Society of Japan. Ser. II*, 2020. `https://doi.org/10.2151/jmsj.2020-016`.

[5] L. Giraud et al. Resiliency in numerical algorithm design for extreme scale simulations. *International Journal of High Performance Computing Applications*, under review, 2020.

[6] D. Göddeke, M. Altenbernd, and D. Ribbrock. Fault-tolerant finite-element multigrid algorithms with hierarchically compressed asynchronous checkpointing. *Parallel Computing*, 49:117–135, 2015.

[7] M. F. Hoemmen, M. A. Heroux, K. B. Ferreira, and P. G. Bridges. Fault-tolerant iterative methods via selective reliability. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2011.

[8] C. Kühnlein, W. Deconinck, R. Klein, S. Malardel, Z. P. Piotrowski, P. K. Smolarkiewicz, J. Szmelter, and N. P. Wedi. FVM 1.0: A nonhydrostatic finite-volume dynamical core formulation for IFS. *Geosci. Model Dev.*, 12:651–676, 2019.

[9] P. Sao and R. Vuduc. Self-stabilizing iterative solvers. In *Proceedings of the Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, page 4. ACM, 2013.

[10] T. C. Schulthess, P. Bauer, N. Wedi, O. Fuhrer, T. Hoefler, and C. Schär. Reflecting on the goal and baseline for Exascale Computing: A roadmap based on weather and climate simulations. *Computing in Science & Engineering*, 21(1):30–41, 2018.

[11] P. K. Smolarkiewicz and L. Margolin. Variational methods for elliptic problems in fluid models. *In:Proc. Workshop on Developments in Numerical Methods for Very High Resolution Global Models, 5-7 June 2000, ECMWF, Reading, UK, 137-159*, pages 137–159, 2000.

[12] P. K. Smolarkiewicz and J. Szmelter. MPDATA: An edge-based unstructured-grid formulation. *Journal of Computational Physics*, 206(2):624–649, 2005.

[13] P. K. Smolarkiewicz and J. Szmelter. A nonhydrostatic unstructured-mesh soundproof model for simulation of internal gravity waves. *Acta Geophysica*, 59:1109–1134, 2011.

[14] P.K. Smolarkiewicz and L.G. Margolin. Variational solver for elliptic problems in atmospheric flows. *Appl. Math. Comp. Sci*, 4(4):527–551, 1994.

[15] S. J. Thomas, J. P. Hacker, P. K. Smolarkiewicz, and R. B. Stull. Spectral preconditioners for nonhydrostatic atmospheric models. *Monthly Weather Review*, 131(10):2464–2478, 2003.

MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

16/2021   Salvador, M.; Dede', L.; Manzoni, A.
Non intrusive reduced order modeling of parametrized PDEs by kernel POD and neural networks

17/2021   Chew, R.; Benacchio, T.; Hastermann, G.; Klein, R.
Balanced data assimilation with a blended numerical model

18/2021   Gigante, G.; Vergara, C.
On the choice of interface parameters in Robin-Robin loosely coupled schemes for fluid-structure interaction

15/2021   Fumagalli, A.; Patacchini, F.S.
Model adaptation in a discrete fracture network: existence of solutions and numerical strategies

14/2021   Peli, R.; Menafoglio, A.; Cervino, M.; Dovera, L.; Secchi, P;
Physics-based Residual Kriging for dynamically evolving functional random fields

13/2021   Ferro, N.; Perotto, S.; Cangiani, A.
An anisotropic recovery-based error estimator for adaptive discontinuous Galerkin methods

10/2021   Di Michele, F.; May, J.; Pera, D.; Kastelic, V.; Carafa, M.; Smerzini, C.; Mazzieri, I.; Rubino,
Spectral elements numerical simulation of the 2009 L'Aquila earthquake on a detailed reconstructed domain

11/2021   Antonietti,P.F.; Manzini, G.; Mazzieri, I.; Scacchi, S.; Verani, M.
The conforming virtual element method for polyharmonic and elastodynamics problems: a review

12/2021   di Cristofaro, D.; Galimberti, C.; Bianchi, D.; Ferrante, R.; Ferro, N.; Mannisi, M.; Perotto, S
Adaptive topology optimization for innovative 3D printed metamaterials

09/2021   Riccobelli, D.; Noselli, G.; DeSimone, A.
Rods coiling about a rigid constraint: Helices and perversions