Adaptive control in roll-forward recovery for extreme scale multigrid

Markus Huber¹, Ulrich Rüde²,³ and Barbara Wohlmuth¹

Abstract
With the increasing number of compute components, failures in future exa-scale computer systems are expected to become more frequent. This motivates the study of novel resilience techniques. Here, we extend a recently proposed algorithm-based recovery method for multigrid iterations by introducing an adaptive control. After a fault, the healthy part of the system continues the iterative solution process, while the solution in the faulty domain is reconstructed by an asynchronous online recovery. The computations in both the faulty and the healthy subdomains must be coordinated in a sensitive way, in particular, both under- and over-solving must be avoided. Both of these waste computational resources and will therefore increase the overall time-to-solution. To control the local recovery and guarantee an optimal recoupling, we introduce a stopping criterion based on a mathematical error estimator. It involves hierarchically weighted sums of residuals within the context of uniformly refined meshes and is well-suited in the context of parallel high-performance computing. The recoupling process is steered by local contributions of the error estimator before the fault. Failure scenarios when solving up to $6.9 \times 10^{11}$ unknowns on more than 245,766 parallel processes will be reported on a state-of-the-art peta-scale supercomputer demonstrating the robustness of the method.

Keywords
Algorithm-based fault tolerance, high-performance computing, multigrid methods, error estimator, adaptive recovery

1. Introduction
The driving force behind the research interest in achieving larger computing systems with more than $10^{18}$ floating point operations (flops) per second (exa-scale) originates in the potentials to expand science and engineering by simulation. For the upcoming exa-scale computing era, expected in the next decade (Cappello et al., 2009, 2014; Dongarra et al., 2011), the enormous increase in compute power also constitutes new grand challenges for these architectures. One of the key challenges will be building reliable systems with massively increasing components while keeping the energy consumption of the system low. This increase automatically reduces the mean time between failures (MTBF) of hard faults that cause a physical loss of compute cores, nodes, and cabinets. Therefore, using today’s checkpoint/restart techniques become difficult, since the time for checkpointing and restart may exceed the expected MTBF (see Cappello, 2009; Cappello et al., 2009, 2014; Oldfield et al., 2007; Ropars et al., 2013; Shahzad et al., 2013, and references therein). In addition, each checkpoint requires time for writing and automatically yields a higher power consumption. Moreover, the memory necessary for checkpointing must be provided (Kohl et al., 2019) and must be kept in a reliable state, which may further increase the energy consumption. Besides hard faults, an increase of faults that are not immediately noticeable is also expected. These faults occur in “bit-flips” corrupting, for example, floating point number representations, and are called soft faults. In the hardware, a significant amount of power is consumed by error correction code memory used to protect from these soft errors. Removing such power-hungry hardware may be necessary to operate exa-scale systems due to its high energy costs. On the other hand, it would additionally make each component increasingly unreliable (see Cappello, 2009; Lammers, 2010; Winstead and Rodrigues, 2012, and references therein).

To minimize the overhead of reliable computations and make exa-scale computation possible by reducing the energy costs, other abstraction layers in the software have

¹ Technische Universität München, München, Germany
² Friedrich-Alexander Universität Nürnberg-Erlangen, Erlangen, Germany
³ CERFACS, Parallel Algorithms Project, Toulouse, France

Corresponding author:
Markus Huber. Technische Universität München, Bolzmannstraße 3, 85748 Garching, Germany.
Email: huber@ma.tum.de
to provide suitable fault-tolerance techniques. Algorithm-based fault tolerance (ABFT) uses the inherent characteristics of the algorithm to obtain resilient results and saves energy and other resources. However, it is the user’s or programmer’s task to define regions in the code which have to be protected or to define data that is necessary for continuing computations in case of a failure and to implement recovery techniques. ABFT was introduced by Huang and Abraham (1984) by using checksums to control computations for basic matrix operations. Since then, many efforts have been made in various directions to provide and incorporate reliability at the algorithmic level (Anfinson and Luk, 1988; Boley et al., 1992; Bosilca et al., 2009, 2015).

One widely used parallelization model for state-of-the-art high-performance software uses the message passing interface (MPI). Here, the loss of a core or compute node results in an immediate termination of the MPI application. To provide a fault-free run-time system to the application by, for example, MPICH-V (Bouteiller et al., 2006), red-MPI (Fiala et al., 2012) might not be sustainable for extreme scale (see Heroux, 2013). More promising is the extension of the MPI standard in a fault-tolerant version such as the user-level failure mitigation (ULFM) (Brandt and Livne, 2011; Hackbusch, 1985) for solving the linear system on a hierarchy of meshes. These methods can have optimal complexity with respect to the number of degrees of freedom (DOFs) and can be implemented very efficiently for high-performance simulations (Baker et al., 2016; Gmeiner et al., 2016; Notay and Napov, 2015; Sundar et al., 2012). Therefore, ABFT techniques for soft and hard errors are of interest for these methods. The influence of soft faults on algebraic multigrid is studied in Casas et al. (2012). In Altenbernd and Göddeke (2018), soft errors are detected and corrected by checksums within multigrid schemes. The mathematical theory of the influence of soft faults on multigrid schemes is developed in Ainsworth and Glusa (2016b, 2016a). Hard faults in the context of multigrid are investigated in Göddeke et al. (2015), where the multigrid hierarchy is used to save compressed checkpoints and to restart from them. In Huber et al. (2016), we introduced an ABFT technique for resilient multigrid methods for hard faults. More specifically, we have focused on a local recovery method within multigrid V-cycles and combined it with domain decomposition ideas to develop a global recovery method. We have shown that the delay caused by faults in terms of run-time can be completely compensated when accelerating the local recovery by additional compute power.

1.1. Algorithm-based fault-tolerant techniques for linear solvers

In scientific and engineering simulations, partial differential equations (PDEs) play a central role. The methods for solving linear systems arising from PDEs by discretization are of special interest for employing ABFT techniques, because they are often memory- and time-consuming.

Direct solvers are considered in Luk and Park (1988), Davies and Chen (2013), and Du et al. (2012) by using checksums to protect from soft errors. These methods are not applicable to iterative methods such as successive overrelaxation (SOR) and Krylov-space methods and were extended in Bridges et al. (2012) and Langou et al. (2008). A resilient overlapping domain decomposition solver is considered in Sargsyan et al. (2015). The presented solver can inherently deal with hard faults by neglecting their contributions and with soft faults by a filtering technique. This can be further improved by using a priori error bounds to identify admissible local solutions in the solver (Mycek et al., 2017). Especially for hard faults, local recovery techniques enjoy increased interest (Agullo et al., 2013; Gamell et al., 2017; Göddeke et al., 2015; Huber et al., 2016; Langou et al., 2008). They identify the lost part in the simulation and perform a local online recovery. Typically, the internal state of a solver comprises both static and dynamic data. In this article, we will focus on the reconstruction of the dynamic numerical data. Stals (2019) introduces complex methods to reconstruct also the mesh data that is needed for adaptive multigrid solvers. For the reconstruction of the static data, we refer to the in-memory checkpointing approaches and their performance analysis presented (e.g. Kohl et al., 2019).

In the following, we will use multigrid methods (Brandt and Livne, 2011; Hackbusch, 1985) for solving the linear system on a hierarchy of meshes. These methods can have optimal complexity with respect to the number of degrees of freedom (DOFs) and can be implemented very efficiently for high-performance simulations (Baker et al., 2016; Gmeiner et al., 2016; Notay and Napov, 2015; Sundar et al., 2012). Therefore, ABFT techniques for soft and hard errors are of interest for these methods. The influence of soft faults on algebraic multigrid is studied in Casas et al. (2012). In Altenbernd and Göddeke (2018), soft errors are detected and corrected by checksums within multigrid schemes. The mathematical theory of the influence of soft faults on multigrid schemes is developed in Ainsworth and Glusa (2016b, 2016a). Hard faults in the context of multigrid are investigated in Göddeke et al. (2015), where the multigrid hierarchy is used to save compressed checkpoints and to restart from them. In Huber et al. (2016), we introduced an ABFT technique for resilient multigrid methods for hard faults. More specifically, we have focused on a local recovery method within multigrid V-cycles and combined it with domain decomposition ideas to develop a global recovery method. We have shown that the delay caused by faults in terms of run-time can be completely compensated when accelerating the local recovery by additional compute power.

1.2. Contribution

In this article, we extend the global recovery idea proposed in Huber et al. (2016), in which we fixed the number of iterations in the recovery a priori. The quality of the recovery depends sensitively on this number. Thus, there is a need for an automatic and a posteriori selection of the number of iterations in the recovery before recoupling that is based on a rigorous criterion. This extension is of interest, since remaining in the recovery mode for too long increases the run-time without improving the approximation toward the solution. On the other hand, interrupting the recovery process too early leads to a pollution of the approximation, since the local error components in the failed domain are still large due to the fault. Hence, additional multigrid iterations are necessary to converge to a sufficient approximation such that the total run-time increases. A sophisticated recoupling strategy is of
importance to allow a run-time minimal recovery. It consists of two components: an error indicator for the quality of the recovery in the faulty domain and a recoupling bound. The recovery is stopped, and the recoupling of faulty and healthy processors is enforced, when the approximation quality in the faulty domain is below this bound.

We apply a mathematically motivated error estimator for both components to guarantee an efficient recoupling. Many error estimator concepts exist in the mathematical literature that are mostly proposed for the discretization error or the quantity of interest (see Ainsworth and Oden, 2000). However, most of them are too costly for high-performance computing. We will consider the hierarchically weighted (HW) residual estimator (Rüde, 1993b), which employs multigrid components to estimate the algebraic error and can be implemented with minimal overhead within multigrid schemes. It can be extended to estimate the total error that is the combined error consisting of algebraic and discretization component. All features of the estimator make it useful for our recovery. Depending on the scenario described, it may be beneficial to use different recoupling strategies. For the definition of the recoupling bounds, we distinguish between the following two strategies. The global recoupling bound enforces to solve the local subproblem to the size of the algebraic error before the fault. The local recoupling bound requires that the approximation quality after the recovery fulfills a local condition that is motivated by a portion of the global error associated with a single process. Global and local bounds only differ by a weighting with the number of processors.

Both bounds can be evaluated in case of a failure by a single global communication and without additional computations such that the overhead is kept to a minimum. In the recovery, modified forms of the HW estimator are applied to obtain a quantitative estimated error in the faulty domain. It has been verified through numerical experiments that the recoupling strategy employing these versions of the HW estimator guarantees an optimal recovery method without any run-time delay in comparison to a fault-free execution. We test the adaptive a posteriori recovery strategy on a state-of-the-art supercomputer.

2. Model problem, multigrid solver, and data structure

In this section, we introduce the model problem, the discretization by a standard low-order finite element method and the multigrid solver. The implementation of the multigrid method is carried out within the high-performance hierarchical hybrid grids (HHG) framework (see Bergen, 2006; Gmeiner et al., 2016). The underlying data structure of the framework is similar to many parallel multigrid implementations (Falguette and Jones, 2000; Hülsemann et al., 2005) and constitutes no restriction for the proposed ABFT method. Our techniques can be realized within other frameworks such as described in Sundar et al. (2012) and Baker et al. (2016).

2.1. Model problem

We consider a bounded polyhedral domain \( \Omega \subset \mathbb{R}^3 \) and denote by \( u \in V = H^1_0(\Omega) \), the solution of the Poisson equation

\[
-\Delta u = f \quad \text{in } \Omega, \\
\frac{\partial u}{\partial \mathbf{n}} = 0 \quad \text{on } \partial \Omega, \tag{1}
\]

with \( f \in L^2(\Omega) \), where \( L^2(\Omega) \) is the space of all square integrable functions on \( \Omega \), and \( H^1_0(\Omega) \) is the standard Sobolev space with zero trace on the boundary. The homogeneous boundary condition in (1) will simplify the notation; a generalization to inhomogeneous and more general boundary conditions is easily possible (see Section 5). The weak formulation of (1) then reads: find \( u \in V \) such that it satisfies

\[
a(u, v) = f(v) \quad \forall v \in V, \tag{2}
\]

where \( a(\cdot, \cdot) \) is the bilinear form

\[
a : V \times V \to \mathbb{R}, \\
a(u, v) := \int_\Omega \nabla u \cdot \nabla v \, dx, \tag{3}
\]

and \( f(\cdot) \) a linear functional.
The intergrid transfer operators are realized by linear intergrid operators associated with the finest mesh, we apply multigrid methods. Note that the equivalence constant only depends on the shape regularity of the triangulation $\mathcal{T}_L$. We call (6) therefore also discrete weighted $L^2$-norm. In what follows, we will focus on an open subdomain $\omega \subset \Omega$, that is, we will be working with a subset $\mathcal{I}_{\ell,\omega}$ of DOFs contained in $\omega$ of the index set of all DOFs $\{1, \ldots, n\}$.

The mesh hierarchy $\mathcal{T}$ is used to construct a geometric multigrid solver with its finest level for $\ell = 0$ and finest level for $\ell = L$. To solve the algebraic equation $A_\ell u_\ell = f_\ell$ associated with the finest mesh, we apply multigrid V-cycles (see, e.g., Hackbusch, 1985, algorithm (2.5.4)). The intergrid transfer operators are realized by linear interpolation $I_{\ell+1}^\ell : = (I_{\ell+1}^\ell)^T$ for $\ell = 0, \ldots, L - 1$. We choose the operators in (4) for level $\ell = 0, \ldots, L - 1$ defined by direct assembly on each level $\ell$ as coarse grid operators within the V-cycle approximation scheme. For smoothing, we use standard point-wise relaxation routines, for example, (damped) Jacobi or (colored) Gauss–Seidel smoothers. The parallel implementation is based on a hybrid realization of the smoothers; for details, see Gmeiner et al. (2016).

In the following, we will study the convergence process of an approximation $u^k_\ell$ to the exact solution $u_\ell$ of (4) on level $L$ within a sequence of V-cycle iterations $k = 0, 1, 2, \ldots$. We study here the algebraic error, that is

$$f : V \rightarrow \mathbb{R}, \quad f(v) := \int_\Omega f v \, dx.$$
domain decomposition. Note all our results will be carried out in the HHG framework, which constitutes such a software package employing this hybrid data structure.

An efficient communication model is to introduce a layer of halos (ghost layers) that hold copies of master data on other memory units. Data dependencies can therefore be organized across process boundaries and make parallel computations possible. The data on these ghost layers can only be read such that it needs to be updated when the master data is modified to hold consistent values. Here, a systematic construction of a ghost layer data structure will be highlighted which is induced by the geometry of the mesh.

A hierarchically organized data structure and communication model can be defined on top of the uniformly refined meshes of the family $\mathcal{T}$. For each of these meshes, the nodal values of a discretization level lie on the vertices, edges, faces, and in the interior of an element of the input mesh $\mathcal{T}_0$. For two tetrahedra in 3-D, this is illustrated in Figure 1 (top). We classify these nodal values according to a system of container data structures. Interior nodal values of an element $T_i \in \mathcal{T}_0$ are stored in a 3-D container, the nodal values lying on the interior of an input mesh face $\partial T_{ij} = \partial T_i \cap \partial T_j$ for $T_i, T_j \in \mathcal{T}_0, i \neq j$, in a 2-D container, the values on the interior of an input mesh edge in a 1-D container and the nodal values of the input mesh vertices in a 0-D container.

In a subsequent step, we introduce the ghost layers. Through the above geometric classification, each nodal value is stored in a unique container and is referred as master copy. These containers are now enriched with copies of neighboring nodal values stored as master data in another container, which are called ghost values. Thus, for an element $T_i \in \mathcal{T}_0$ all nodal values on $\partial T_i$ become ghost values. For the face data structure, the values which lie on $\partial F_{ij}$ and additional layers holding the values of the adjacent tetrahedra define the ghost layer nodal values. Edges and vertices are enriched similarly. In Figure 1 (bottom), the ghost layer enrichment for two input tetrahedra and the face lying in between is illustrated. These extra ghost layers are essential for the efficient implementation of local operations such as relaxation methods acting on the master nodes.

The parallelization on a distributed memory system is based on the distribution of the containers uniquely to processors. If the master data of an interface data structure (face, edge, and vertex) has changed on a processor, it needs to be updated in the ghost layers. Depending on the memory location, the update is a local copy routine if the container belongs to the same memory unit, or MPI communication, if the containers are located on different memory units in the network. Here, we eventually introduce additional copies of the face, edge, and vertex containers so that the communication can be efficiently implemented on the processors of a distributed memory system.

This construction obviously leads to redundancy in face, edge, and vertex data. However, the extra stored data are of lower complexity and induce a negligible memory overhead in the typical cases. It makes it possible to implement a very efficient parallel multigrid solver and permits the numerical recovery of the data in these structures. In case of a failure, the loss of master face, edge, and vertex data can be recovered by the redundant information on other processes. Therefore, we focus in the following on the algorithmic recovery of the unknown data of a refined input element.

3. Resilience for the multigrid solver

In this section, we introduce the model for dealing with faults within large-scale distributed memory systems from Huber et al. (2016). The impact of a fault on the iteration process of multigrid methods is studied by considering an exemplary fault situation within the V-cycle schemes. This showcase will also serve in the following to study the adaptively controlled recovery algorithm.

3.1. Faulty computing environment

In the following, we consider hard failures, that is, the loss of a computing entity such as one or several cores or a complete compute node. The simulation of faults in a computing environment is a wide research topic (see e.g. Benso and Prinetto, 2010; Hsueh et al., 1997) and is beyond our consideration. Moreover, the immediate detection and instant replacement of faulty processors is not part of the MPI standard and not yet routinely supported by hardware and software.

However, efforts are under way. Currently, only extensions such as ULFM are available (Bland et al., 2012, 2013). Therefore, we artificially inject a fault into our simulation and state that the data of specific processes are

![Figure 1. Top: two refined input elements; bottom: ghost layer structure of two input elements.](image)
lost and do not explicitly kill the processes. The data of these faulty entities are lost and have to be recovered.

We distinguish between static and dynamic data recovery. Static data such as internal data structure, stiffness matrix, or right-hand side has to be recovered by checkpointing methods or by a new setup. We study the recovery of the dynamic data (unknown vector) numerically and set the nodal values of the lost entities to zero, when injecting the fault. Since we only consider settings in which the nodal data values of one process crashes, and the nodal data values of one process are notified and returning immediately back to the top level of the fault is injected after post-smoothing and before pre-smoothing on the finest grid level. This can be easily generalized by stopping the calculation when a failure is notified and returning immediately back to the top level of the fault-free execution and after injecting a failure is six to seven iterations.

4. Adaptive recoupling fault-recovery strategy

In this section, we extend the Dirichlet–Dirichlet (DD) algorithm introduced in Huber et al. (2016) by an adaptive recoupling strategy. In case of a failure, the algorithm decouples faulty and healthy processes and solves independent Dirichlet problems on both associated subdomains by fixing the interface data. Therefore, communication between healthy and faulty subdomains is avoided and the locally enlarged error does not pollute the unknown values of the healthy domain. To compensate for the resulting error difference between faulty and healthy domain and to obtain a nearly equally distributed error after the recovery, more computations are necessary in the faulty domain. Therefore, we introduce more compute power in the faulty domain in the form of a superman speedup factor $\eta_{F} \in [1, \infty)$ to account for this imbalance. This can be realized by further domain partitions of the faulty domain and assigning more compute processes which were handled by the node at each time step.
a single process before the fault. Hence, the superman \( h_s \) is
the ratio between the compute power in the faulty domain after and before the fault. Due to the wrong interface data, we have to recouple both subproblems to solve the global problem. The recovery is efficient when we can guarantee almost equal execution times in healthy and faulty domain, that is, \( t_F \approx t_I \), and a sufficient recovery of the data in the faulty domain. Here, the performance sensitively depends on the choice of iterations \( n_F \) and \( n_I \), that is, the number of iterations executed in the faulty and healthy domain. It is possible to identify an optimal paring \((n_F, n_I)\) depending on the superman factor \( h_s \) such that the delay caused by the fault can be completely compensated. However, a suboptimal choice of these parameters leads on the one hand an insufficient recovery of the data in the faulty domain. This is called under-solving. On the other hand, by remaining for too long in the recovery, the global approximation obtained from the subproblems suffers from the wrong interface data. This is called over-solving. Both waste computational resources and time and therefore delay the convergence to the global solution. For instance, in Figure 4, the DD algorithm is applied with suboptimal \((n_F, n_I)\) pairing with superman speedup \( h_s = 4 \). One example shows the results of under-solving with \((n_F, n_I) = (4, 1)\) and the other over-solving with \((n_F, n_I) = (24, 6)\). Both yield a delay in convergence of three to four iterations.

We can set the recoupling parameters heuristically a priori before applying the recovery strategy. However, the choice is influenced by several problem and solver-dependent factors such as the coarse-level solver. Therefore, an a posteriori choice of these parameters is necessary to find optimal parameter pairings without extensive numerical studies to determine them.

### 4.1. Adaptive recoupling algorithm

In the following, we extend the DD algorithm of Huber et al. (2016, algorithm 2) by a recoupling strategy for the faulty and healthy subproblem when the faulty problem is sufficiently well approximated. Therefore, we briefly recall some components of the algorithm. We rewrite the linear system \( A_L u_L = f_L \), see (4) in terms of Dirichlet problems on the subdomains and add the ghost layer data structures (see Section 2.3) at the interfaces to represent the necessary coupling conditions. The extended stiffness matrix is given by

\[
A_L = \begin{pmatrix}
    A_{H} & A_{I} & 0 & 0 & 0 \\
    0 & \mathbf{Id} & -\mathbf{Id} & 0 & 0 \\
    A_{I} & 0 & A_{G} & 0 & A_{F} \\
    0 & 0 & -\mathbf{Id} & \mathbf{Id} & 0 \\
    0 & 0 & 0 & A_{F} & A_{F}
\end{pmatrix}
\]

and the unknown and the right-hand side vectors by

\[
u_L = \begin{pmatrix}
u_{X_0} \\
    v_{X_0} \\
    v_{X_0} \\
    v_{X_0} \\
    v_{X_0}
\end{pmatrix}, \quad f_L = \begin{pmatrix}
    f_{X_0} \\
    0 \\
    0 \\
    0 \\
    f_{X_0} 
\end{pmatrix},
\]

The submatrices are associated with the block unknowns. In more general settings, they also depend on the basis functions and the PDE. The submatrices \( A_{XX} \) with \( X \in \{F, G, I\} \) are the blocks of the stiffness matrix \( A_L \) corresponding to the parts of the vector \( u_L \) with dependencies just in \( \Omega_X \). The block matrices \( A_{XY} \) with \( X \in \{F, G, I\} \) and \( Y \in \{I, F, G, G\} \), \( X \neq Y \), represent the couplings in the stiffness matrix between the subdomains \( \Omega_F, \Omega_G, \) and \( \Omega_I \). Because of the symmetry of the bilinear form (3), we can identify \( A_{H} \) with \( A_{H}^T \), and \( A_{F} \) with \( A_{F}^T \). The consistency of the redundant data across process boundaries is guaranteed by row 2 and row 4 in the extended system. The coupling of the interface data in healthy and faulty domain is presented by row 3. In case of a fault, the data corresponding to the subdomain \( \Omega_F \) are lost. However, interface data are duplicated and still available on another processor, that is \( u_{X_0} \) and \( u_{X_0}^I \) are lost but \( u_{X_0} \) and \( u_{X_0}^I \) are still available. Thus, it can be easily recovered by inherently using the distributed data structures. For the volume container, the inner node values, that is, \( u_{X_{I^0}} \), are not available. By neglecting the interface couplings (row 3) and fixing the interface values \( u_{X_{I^0}} \), we numerically reconstruct the lost values in \( \Omega_F \) by approximating a Dirichlet problem corresponding to row 5. Asynchronously, we approximate also a Dirichlet problem in the healthy domain corresponding to row 1.

To coordinate the recoupling between faulty and healthy domain, we propose a stopping criterion for the approximation accuracy of the faulty subproblem. Therefore, we define and fix a bound \( \sigma \), before starting the asynchronous computation in the healthy and faulty subdomain. The corresponding stopping criterion must be fulfilled before

1: Solve (4) by multigrid cycles until (11) is satisfied.
2: if fault has occurred then
3: STOP solving (4).
4: Recover boundary data $u_{I_{0}}^{j}$ using line 4 in (12).
5: Initialize $u_{I_{0}}^{j}$ with zero.
6: Compute the bound $\sigma$ and fix $\kappa > 0$ for the re-coupling criterion (14).
7: In parallel do:
8: a) In faulty domain: Approximate $A_{I_{0}}u_{I_{0}}^{j} = f_{I_{0}}^{j} - A_{I_{0}}u_{I_{0}}^{j}$ by multigrid cycles with superman factor $\eta_{I_{0}}^{j}$.
9: After each cycle: Compute the approximation indicator $\eta_{I_{0}}^{j}$.
10: If the stopping criterion (14) is fulfilled: Send signal to healthy domain and GOTO line 18.
11: b) In healthy domain:
12: In parallel do:
13: i) Approximate $A_{I_{0}}u_{I_{0}}^{j} = f_{I_{0}}^{j} - A_{I_{0}}u_{I_{0}}^{j}$ by multigrid cycles.
14: ii) If signal was sent by faulty domain, then interrupt multigrid cycle and GOTO line 16.
15: end do
16: Prolongate correction and GOTO line 18.
17: end do
18: RETURN to line 1 with new values $u_{I_{0}}^{j}$ in $\Omega_{1}$ and $u_{I_{0}}^{j}$ in $\Omega_{2}$.
19: end if

recoupling. During the recovery by V-cycles in the faulty domain, we check after each iteration $j = 0, 1, \ldots$ if the local subproblem is sufficiently well approximated, and the error has fallen below this bound. If the error indicator $\eta_{I_{0}}^{j}$ satisfies,

$$\eta_{I_{0}}^{j} < \kappa \sigma, \quad (14)$$

then, a signal is sent to the healthy domain to stop its computations, where $\kappa > 0$ has to be chosen properly. The smaller $\kappa$ is, the more difficult condition (14) is to satisfy. Asynchronously, that is, independent of the V-cycle iteration and error estimation in the faulty domain, V-cycles are also executed in the healthy domain. For the recoupling process of the subproblems, the natural parallel synchronization of the multigrid algorithm must be respected. The recovery processes operating in the faulty subdomain can signal their termination to the processes of the healthy domain. If such a signal is received, the healthy processes will proceed until the next canonical synchronization point of a parallel multigrid algorithm and then terminate the iteration in the respective V-cycle. More precisely, the current correction in the healthy domain is prolonged to the top level as quickly as possible; then, the recoupling is performed, and the regular iteration is resumed. The explicit choices of the error estimator $\eta_{I_{0}}^{j}$ and the recoupling bound $\sigma$ will be the focus of the following sections. The algorithm is presented as Algorithm 1. The adaptively controlled algorithm requires in line 2 a collective communication to inform all processes about the fault, in line 4 and 18 point-to-point communication to recover the interface data and to synchronize the interface after the recovery. Additionally, we require two collective communications to determine the recoupling criterion threshold and one for informing all processes about the recoupling.

The criteria (11) for stopping the global solution approximation (see line 1 in Algorithm 1) and (14) for recoupling the subproblems in the recovery (see line 12 in Algorithm 1) can be realized by different approaches. Our explicit choices are described in Section 4.2.3.

4.2. Selection of the recoupling criterion

In the following subsections, we summarize the idea of the hierarchical weighted (HW) error estimator and demonstrate its efficiency for controlling the algebraic error of the iteration process by a sequence of numerical experiments. The error estimator will in the following serve to define indicators for the error in the faulty and healthy domain and will then be applied in the recoupling criterion (14).

4.2.1. Hierarchical weighted error estimator. Multigrid methods are often such efficient solvers that their computational cost (see Brandt and Livne, 2011; Stüben and Trottenberg, 1982) is lower than conventional error estimators. More precisely, a multigrid method can compute a sufficiently accurate numerical approximation more cheaply than some of the error estimators that have been proposed. Such expensive error estimators that additionally require an exact discrete solution before they can be applied are of theoretical interest, cf. the textbook of Ainsworth and Oden (2000), for a discussion on different types of estimators, but they are often unsuitable for resource-aware large-scale computing.

As introduced in Section 2.2.2, the $L^2$-norm of the residual of the linear system (4) on level $L$ is often taken as indicator of how accurate the approximation within an iterative scheme is, for example, in the stopping criterion (11). However, evaluating the residual on just one level yields insufficient information on the quality of an iterate, since
with minimal eigenvalue \( \lambda_{\text{min}}(A_L) = \mathcal{O}(h_L^2) \) and maximal eigenvalue \( \lambda_{\text{max}}(A_L) = \mathcal{O}(h_L). \) The ratio of the constants deteriorates with decreasing mesh size \( h_L. \)

In the context of multigrid methods, one obvious way is to estimate the error by exploiting the existing natural mesh hierarchy. In particular, HW error estimators are available to measure both the algebraic error in the \( L^2 \)- and in the \( H^1 \)-norm (cf. Rüde, 1993b). These estimators are efficient in terms of both lower and upper error bounds and in terms of run-time. In addition, the total error consisting of the algebraic and the discretization error component can be estimated, opening up the possibility to algorithmically balance the error contributions within adaptive control strategies (see Rüde, 1993a). We focus in the following on the HW error estimation for the algebraic error in the \( L^2 \)-norm. Similar approaches can be found for the discretization error in the \( H^1 \)-norm in the hierarchical error estimator in Deuflhard et al. (1989) or for the discretization error of adaptive boundary elements in Färmann (1998).

HW error estimators are theoretically based on a stable splitting of the underlying \( H^1(\Omega) \) function space of the continuous problem. Stable splittings can be presented in an abstract way using infinite dimensional Hilbert spaces (see Oswald, 1994, section 4.1, for 2-D and 3-D problems; Oswald, 2001, for 2-D and 3-D Rüde 1993b, chapter 2, for two dimensions). Since here it will be used for error estimation within the context of the hierarchy of meshes \( T_0, \ldots, T_L \) with the associated finite dimensional approximation subspaces \( V_0, \ldots, V_L \), we can work with the matrix and vector notation using the nodal value vectors.

As a better alternative than using the residual (9) for error estimation, one can thus use a sum of scaled residuals on a sequence of refined levels (see Rüde, 1993b). The residual of the \( \ell \)th linear system from (4) obtained by restriction of the \( \ell \)th residual reads

\[
\frac{1}{\lambda_{\text{max}}(A_L)} \| \mathbf{r}_L \|_{L^2} \leq \| \mathbf{r}_L \|_{L^2} \leq \frac{1}{\lambda_{\text{min}}(A_L)} \| \mathbf{r}_L \|_{L^2}.
\]

with minimal eigenvalue \( \lambda_{\text{min}}(A_L) = \mathcal{O}(h_L^2) \) and maximal eigenvalue \( \lambda_{\text{max}}(A_L) = \mathcal{O}(h_L). \) The ratio of the constants deteriorates with decreasing mesh size \( h_L. \)

The proofs for 2-D problems from Rüde (1993b, section 3.3) can be extended to 3-D by stable splitting arguments reported in Oswald (1994, section 4.2). We additionally remark that the hierarchical sum in (17) coincides with the BPX preconditioner of Bramble et al. (1990), which is derived without dimensionality restrictions.

4.2.2. Numerical experiments. In the following numerical experiments, we study the HW error estimator (17) for the problem and solver setup described in Section 3.2. We discretize the unit cube in a structured way by 3,072 initial tetrahedrons and consider two test cases, one with three levels of refinement and a larger one having six levels of refinement. The corresponding fine grid problems consist then of \( 2.5 \times 10^7 \) DOFs and \( 1.3 \times 10^8 \) DOFs, respectively. To solve the discretized problem, we apply \( V(3,3) \)-cycles. Further, on the coarsest grid level, we use a Jacobi-preconditioned conjugate gradient (PCG) method with 30 iteration steps. The number of coarse-level iterations is sufficient to guarantee mesh-independent convergence of the multigrid V-cycles. For the evaluation of the estimator (17), we compute after each V-cycle iteration the residual, scale it on each level and calculate the norm. We realize the scaling in the estimator by using one application of a parallel Jacobi-smoothing routine on levels \( \ell = 1, \ldots, L \).

On the coarsest level \( \ell = 0 \), for which the inversion of \( A_0 \) is necessary, the same PCG-method as for the V-cycle is used. Note, that although the problem is small and an exact inversion would be possible, we only apply the same number of coarse-level iterations as for the V-cycle. We observed in our experiments almost no influence on the algebraic error estimate, when solving the coarse grid problem to a higher accuracy. We note that the algebraic estimate can also be recorded in a similar form within the V-cycle iteration which is then computationally almost “for free”.

We compare the convergence of the algebraic error, the estimated algebraic error and the residual for refinement level 3 (left) and 6 (right) in Figure 5. To compute the algebraic error, we determine in a preprocessing step the exact finite element solution \( y_L. \) From the estimates for the residual, see (15), we scale the \( L^2 \)-norm of the residual by the reciprocal of the minimal and maximal eigenvalue of \( A_L \), which gives a lower and upper bound for the algebraic error. Therefore, we determine in another preprocessing step the extreme eigenvalues for \( A_L \) for \( L = 3 \). The derivation of these eigenvalues is computationally expensive, when standard algorithms are used, and is here only feasible for moderate system sizes. These eigenvalues are then extrapolated for the large system \( A_L, L = 6 \), by \( \lambda_{\text{min},6} = 2^{-3} \lambda_{\text{min},3} \) and \( \lambda_{\text{max},6} = 8^{-3} \lambda_{\text{max},3} \), where we use that the mesh is uniformly refined in each step and the knowledge about the \( h_L \)-scaling of the eigenvalues.
For both refinement levels, the residual shows in the first three to four iterations a pre-asymptotic speed of convergence and then reaches its asymptotic rate. While the algebraic error converges almost from the beginning with its asymptotic rate. The scaled residual clearly bounds the algebraic error from above and below, as stated in (15). However, with decreasing mesh size, the ratio between lower and upper bound deteriorates and does not provide sufficiently sharp bounds for the algebraic error. When we consider the HW error estimator, we observe a very good agreement for both $L = 3$ and $L = 6$ with the actual algebraic error in the whole convergence history.

In addition, we also compare the local distribution of the residual and the estimator with the algebraic error. For the recovery strategy, we are interested in the local error for which the estimator is used to define a local stopping criterion. In the two series of plots in Figures 6 and 7 for refinement level 3 (top) and 6 (bottom), we depict the point-wise evaluation of the algebraic error in the middle after five V-cycle iterations. On the right, the corresponding residual is presented. As local indicator, we visualize on the left the vector

![Figure 5](image.png)

**Figure 5.** Convergence of the algebraic error, estimated algebraic error and scaled residual by applying V(3,3)-cycles on a box with 3,072 initial tetrahedrons after 3 (left) and 6 (right) uniform refinement levels.

![Figure 6](image.png)

**Figure 6.** Visualization of the point-wise error components on a box with 3,072 initial tetrahedrons and 3 uniform refinement levels after 5 V(3,3)-cycles. Left: point-wise estimated algebraic error; middle: point-wise algebraic error; right: point-wise residual.

![Figure 7](image.png)

**Figure 7.** Visualization of the point-wise error components on a box with 3,072 initial tetrahedrons and 6 uniform refinement levels after 5 V(3,3)-cycles. Left: point-wise estimated algebraic error; middle: point-wise algebraic error; right: point-wise residual.
which is the point-wise contribution to the HW error estimator. The visualization of the algebraic error and the HW error estimator show a good visual agreement in terms of error scale and localization. Again, the residual is a poor indicator of the algebraic error in the size and for the error location. The quality of its estimate deteriorates by three orders of magnitude, when reducing the mesh size from level $L = 3$ to $L = 6$.

4.2.3. Explicit choices. In the following, we describe the stopping criterion choices in Algorithm 1 by applying the HW estimator (17). In the recovery algorithm, error estimation is required at two different parts. One is necessary for the evaluation of $\eta^k_{\Omega}$ after each V-cycle in the global stopping criterion (11) in line 1. The other one is applied for the recoupling criterion (14) in line 10. For this, an efficient recoupling bound $\sigma$ needs to be defined and the computation of $\eta^k_{\Omega}$ is necessary after each V-cycle iteration in $\Omega_F$. A summary and description of used symbols in the following can be found in Table 1.

In view of the previous Section 4.2.1, the HW estimator provides better estimates than the residual for the algebraic error. Therefore, we use the HW estimator in the global stopping criterion (14) and set

$$\eta^k_{\Omega} := \eta^k_{\text{HW},L}.$$  \hfill (20)

After each V-cycle, we check if (11) has been fulfilled to stop the iteration.

For defining suitable choices for the recoupling bound $\sigma$ and the error indicator $\eta^k_{\Omega}$, in the faulty subdomain, we use again the HW estimator in a slightly modified form to make the computations feasible for our recovery algorithm. We recall that each node is associated as master node to exactly one processor $p$. Let $\mathcal{I}_{wp}$ be the index set of the master nodes of processor $p$ and $\mathcal{P}$ the set of processors. Then, we define the error estimator contribution $\eta^k_{\text{HW},\mathcal{I}_{wp}}$ of the index set $\mathcal{I}_{wp}$ to the algebraic error by

$$\eta^k_{\text{HW},\mathcal{I}_{wp}} := \| \sum_{l=0}^{L} I_l^TD_l^{-1}L_l^\tau \|_{0,\mathcal{I}_{wp}}.$$  \hfill (21)

We find a direct relation of this contribution associated with each process to the HW estimator by the additive decomposition

$$\eta^k_{\text{HW},\mathcal{I}_{wp}}^2 = \sum_{p \in \mathcal{P}} n_{L,\mathcal{I}_{wp}} n_{L,\mathcal{I}_{wp}}^{\text{HW},\mathcal{I}_{wp}}.$$  \hfill (22)

We note that $\eta^k_{\text{HW},\mathcal{I}_{wp}}$ can be regarded as a mean quantity per node in $\mathcal{I}_{wp}$, whereas $\eta^k_{\text{HW},L}$ is the mean quantity per node for all nodes. By using (21), we can define different bounds $\sigma$ for the recoupling criterion (14). To do so, this contribution is automatically documented and updated on each process, when estimating the algebraic error after each V-cycle application. It is additionally stored for each process and is merely a scalar quantity creating almost no overhead. If a failure has occurred after $k = k_F$ iterations, a natural choice for the recoupling bound $\sigma$ would be to take its contribution to the total algebraic estimate before the fault, that is

$$\eta^k_{\text{HW},\mathcal{I}_{wp}}.$$  \hfill (23)

Since all the data on the faulty domain is lost, this value is not available, either. It could be stored by a check-pointing strategy for the estimator contribution. Here, we propose several other strategies without any necessity to introduce checkpoints. We define the following two recoupling bounds: The first recoupling bound is based on a global mean value:

$$[\sigma^{\text{GRB}}]^2 := \sum_{p \in \mathcal{P}} n_{L,\mathcal{I}_{wp}} [\eta^k_{\text{HW},\mathcal{I}_{wp}}]^2.$$  \hfill (24)

We call it global mean recoupling bound (GRB). The second one is defined by the maximum contribution:

$$[\sigma^{\text{LRB}}]^2 := \max_{p \in \mathcal{P}} n_{L,\mathcal{I}_{wp}} \eta^k_{\text{HW},\mathcal{I}_{wp}}.$$  \hfill (25)

We call it local maximum recoupling bound (LRB). In the setup step of the recovery, both thresholds only require a single MPI-Reduce call. Also other local thresholds are possible, which can be based on the minimal process-wise error or the local mean value of the process-wise error. We further point out that in the definition of GRB and LRB, the residual $r^k_{L,\mathcal{I}_{wp}}$ before the fault is used. As already mentioned above, this is reasonable, since we store the local contribution of each process every time the global estimate $\eta^k_{\text{HW},L}$ is computed. Now it remains to define $\eta^k_{\Omega}$. A natural choice would be $\eta^{1/2}_{L,\mathcal{P}} \eta_{\text{HW},\mathcal{I}_{wp}}, p = p_F$, where now the residual of the actual recovery step $j$ is used. However, this definition is not suitable due to two reasons. Firstly, it does not necessarily tend to zero, and thus, we cannot guarantee that the stopping criterion (14) is ever reached. Secondly, it requires communication between faulty and healthy processes, and thus, it is not well suited for a low-cost indicator in large-scale computations. To overcome these shortcomings, we propose to use

$$[\eta^k_{\Omega}]^2 := n_{L,\mathcal{I}_{wp}} \left[ \sum_{l=0}^{L} I_l^TD_l^{-1}L_l^\tau \right]_{0,\mathcal{I}_{wp}}^2,$$  \hfill (26)

where $n_{L,\mathcal{I}_{wp}}$ is the number of inner nodes in $\Omega_F$, $D_{l,F} := (\text{diag}(A))_{FF}$ for $\ell = 1, \ldots, L$, $D_{0,F} := A_{0,FF}$ and $I_{l,F}^\tau$ is the prolongation operator from $V_{l,\Omega_F}$ to $V_{L,\Omega_F}$ with

$$V_{l,\Omega_F} := \{ v \in H^l_0(\Omega) : v = w|_{\Omega_F} \text{ for } w \in V_l \},$$

for $\ell = 0, \ldots, L$. Finally, the residual $r^k_{L,\Omega_F}$ is given by $(I^\tau_{l,F} - T_{l,F})^\tau_{l,F}$, where $r^k_{L,\Omega_F}$ is the fine scale residual restricted to all inner nodes of $\Omega_F$. It is obvious that as the number of
iterations \( j \) in the faulty domain tends to infinity, \( \eta_{\text{f}} \) tends to zero, and thus also \( \eta_{\text{Gb}} \). Moreover, no information exchange across the interface is required and thus no communication between faulty and healthy processes. From a mathematical point of view, \( \eta_{\text{Gb}} \) measures the algebraic error of a local Dirichlet problem on \( \Omega_f \), where the Dirichlet data are given by the values on \( \Omega_f \) before the fault.

By these choices of recoupling bounds and error indicator in the stopping criterion (14), we can conclude the following for the error in the faulty domain after the recovery. Assuming an equally load-balanced problem, then, \( n_{L_1}x \) is roughly the same for all processes and since \( L \) is large, \( n_{L_1}x \approx n_{L_1}x_{\Omega_f} \). Hence, the scaling in (24), (25), and (26) are approximately the same for all \( p \). Therefore, the choice \( \sigma_{\text{LBB}} \) guarantees that after the recovery the algebraic mean error per node in the faulty domain of the surrogate Dirichlet problem is bounded by the maximum mean error per process before the fault. While the bound \( \sigma_{\text{GRB}} \) guarantees that the solution of the faulty subproblem is approximated such that the mean error per node in the faulty subproblem is of the same order as the mean error per node of all nodes. To further quantify the relation between the approximation after the recovery in the faulty domain, the following bounds for GRB and LBB are useful

\[
\sigma_{\text{LBB}} \leq \sigma_{\text{GRB}} \leq (|\mathcal{P}| - 1)^{1/2} \sigma_{\text{LBB}}.
\]

Hence, the difference in the approximation quality obtained by the bounds in the recovery scales by \( \mathcal{O}(|\mathcal{P}|^{1/2}) \).

### 5. Recovery simulations in a faulty environment

In the following section, we apply the adaptively controlled recovery strategy of Algorithm 1 with recoupling/stopping criterion and error indicators of Section 4.2 to study the experimental fault scenario introduced in Section 3.2.

Our experiments are performed on the JUQUEEN (www.fz-juelich.de) supercomputer (cf. Jülich, 2015), an IBM BlueGene/Q system with a peak performance of 5.9 Petaflops, listed on position 22 of the TOP500 (www.top500.org.) list (49th edition, November 2017). Each of the 28,672 nodes is equipped with 16 cores that can execute up to four hardware supported threads to help hiding latencies. The HHG software is compiled by the IBM XL C/ C++ compiler V12.1 using -O3 -qstrict -arch=gp -attune=gp flags and is linked to MPICH2 version 1.5.

We specify in the following the simulation parameters. We set the tolerance of the stopping criterion (11) to \( TOL = 10^{-13} \).

The fault corrupts one MPI process such that the data corresponding to a refined macro-tetrahedron (nearly \( 2.7 \times 10^6 \) DOFs) is lost, as illustrated in Figure 2. Note, a macro-tetrahedron is assumed to be a collection of input tetrahedrons that conform themselves again a tetrahedron. For the algorithmic fault recovery, we apply Algorithm 1, in which the multigrid parameters remain the same as in the global solution process, that is \( V(3,3) \)-cycles. Since just one macro-tetrahedron is corrupted, the coarse grid consists of just a moderate number of DOFs (nearly 35), and thus, this coarse grid problem can be solved efficiently with few PCG iterations. In the healthy domain, we apply the PCG method as for the overall problem.

To realize the \textit{superman} strategy, we partition the faulty domain further onto two or four processes to achieve a superman speedup of approximately \( \eta_s = 2.4 \). Thus, we assign two or four processes to perform the computations within the faulty domain, which were handled by a single process before the fault had occurred. The success of any recovery strategy must be measured by how much the overall solution process is delayed as compared with the fault-free case.

We recall that in the experiments here we do not explicitly kill MPI processes but simulate their failures. In general, to balance the computations for each process, we equally distribute the load associated with a subdomain. Here, we adjust this load distribution such that the faulty domain is handled throughout our simulation by more processes.

We point out that the creation of new processes and their rearrangement in message passing context could have a significant impact on the performance, especially since this functionality is rarely used in current HPC practice. A systematic study will depend on the availability of stable and efficient support through system software and hardware.

#### 5.1. Single fault

The input mesh in our first experiment is the unit cube discretized by 24,576 tetrahedrons, which results after uniform refinement in a linear system with \( 1.3 \times 10^8 \) DOFs. To load balance the problem, we run these simulations with

---

**Table 1. Summary and description of symbols.**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>Number of multigrid iterations</td>
</tr>
<tr>
<td>( n )</td>
<td>Number of DOFs</td>
</tr>
<tr>
<td>( \mathcal{P} )</td>
<td>Set of processes</td>
</tr>
<tr>
<td>( e_k^L )</td>
<td>Algebraic error on level ( L ) after ( k ) iterations</td>
</tr>
<tr>
<td>( r_k^L )</td>
<td>Residual on level ( L ) after ( k ) iterations</td>
</tr>
<tr>
<td>( \eta_{\text{f}} )</td>
<td>Control factor for the recoupling threshold before the fault</td>
</tr>
<tr>
<td>( n_kX )</td>
<td>Number of iterations in faulty/healthy domain</td>
</tr>
<tr>
<td>( n_{L_1}x )</td>
<td>Number of DOFs in subdomain ( X )</td>
</tr>
<tr>
<td>( r_k^Lx )</td>
<td>Residual restricted on subdomain ( X )</td>
</tr>
<tr>
<td>( \eta_{\text{Gb}} )</td>
<td>Superman speedup</td>
</tr>
<tr>
<td>( \sigma_{\text{GRB}} )</td>
<td>Global recoupling threshold, see (24)</td>
</tr>
<tr>
<td>( \sigma_{\text{LBB}} )</td>
<td>Local recoupling threshold, see (25)</td>
</tr>
<tr>
<td>( \kappa )</td>
<td>Control factor for the recoupling threshold</td>
</tr>
<tr>
<td>( \eta_{\text{HWW}} )</td>
<td>Algebraic error estimator for the global problem, see (17)</td>
</tr>
<tr>
<td>( \eta_{\text{Gb}} )</td>
<td>Algebraic error estimator for the fault problem, see (26)</td>
</tr>
<tr>
<td>TOL</td>
<td>Tolerance of stopping criterion (11)</td>
</tr>
</tbody>
</table>
and recoupling strategy reduces significantly the costs in computing the fixed control parameter for decreasing relative faulty domain sizes with respect to meters $k$ recoupling bounds GRB and LRB including different parameters $\kappa$. In this section, we study how the algorithm behaves for the domain size $5.2$. Influence of the local criterion and the faulty domain size

48 MPI processes and assign to each process 512 ($= 8^3$) input mesh tetrahedrons. We use in the recoupling criterion the bound $\sigma^{LRB}$ and $\kappa = 1.0$ for the two superman acceleration factors $\eta_k = 2$ and $\eta_k = 4$. In Figure 8, we present the estimated algebraic error within a faulty solution process. The plots left and right refer to $\eta_k = 2$ and $\eta_k = 4$, respectively. To satisfy the recoupling criterion, six local multigrid iterations are necessary in the faulty subdomain which can be performed simultaneously, while 3.7 and 2.0 iterations are executed in the healthy domain, respectively, depending on the superman acceleration $\eta_k$. In the faulty domain, the error estimator is applied in addition to the V-cycles. Asynchronously, the computations in the healthy domain are executed. The recovery process is highlighted in light green in the figures. Note, that fractional numbers imply that the healthy domain receives the stopping signal for recoupling when the multigrid iteration in the healthy domain has not completed the most recent cycle. So, for example, when the healthy domain has executed the downward branch of a symmetric V-cycle, we will have executed one half of a V-cycle and denote this in the following tables and graphics by 0.5 iterations. Summarizing our experiments, we find that the recovery with $\eta_k = 2$ results in a delay of only up to three iterations and with $\eta_k = 4$ in up a delay of two iterations. Thus, the adaptive recoupling strategy reduces significantly the costs in comparison to applying no recovery strategy.

5.2. Influence of the local criterion and the faulty domain size

In this section, we study how the algorithm behaves for the recoupling bounds GRB and LRB including different parameters $\kappa$. To study the robustness of the recoupling bounds for decreasing relative faulty domain sizes with respect to a fixed control parameter $\kappa$, we use a sequence of four input meshes, consisting of $8^3 \times 6 \times (2^m + 1)^3$, $m \in \{1, 2, 3, 4\}$ tetrahedrons discretizing the unit cube. The setup for the fault and recovery is the same as in the previous experiment, but now the relative size of the lost data decreases asymptotically from $6 \times 10^{-3}$ to $3.4 \times 10^{-5}$ with each input mesh.

In the following weak scaling experiments (see Tables 2 to 4), we perform experiments in excess of 29,480 parallel MPI processes. In the scaling experiment, the number of unknowns increases asymptotically by a factor of 8 per level so that we reach $8.2 \times 10^{10}$ unknowns in the largest calculation shown. Within the multigrid scheme, we fix the mesh hierarchy size to 5 such that the coarse grid problem grows proportionally in the weak scaling. The expected growth of the coarse grid condition number is a factor of 4 for each additional scaling level, so that we increase the number of PCG iterations by a factor of 2 to approximate the coarse grid problem sufficiently (cf. Gmeiner et al., 2014). For the largest simulations, we perform 240 coarse grid iterations per V-cycle. This does not yet affect the good scalability of the multigrid method. For even larger problems, a coarse grid solver with better scalability (such as an algebraic multigrid method) would have to be used.

5.2.1. Performance of the estimator and the recovery algorithm

In Table 2, we present the run-times of the HW estimator (column “global est”) and of the asynchronously applied V-cycle in the faulty and healthy subdomain of the recovery algorithm with respect to the superman $\eta_k = 2$ and $\eta_k = 4$. Additionally, we also include the run-times for error estimation (column “faulty est”) necessary for evaluating (14). In the estimation for the global stopping criterion and in the V-cycle iteration in the healthy domain, we solve the coarse-level problem by an increasing number of iterations. Therefore, in either case, the run-time increases: for the error estimation from 0.97 s to 1.57 s with a parallel
Table 2. Timings (in second) of HW estimator computation (Est) for the global and faulty (sub)domain problem. V-cycle run-times (in second) in the adaptive recovery for the healthy and faulty subproblem for speedup factors $\eta_k = 2$ and $\eta_k = 4$.

<table>
<thead>
<tr>
<th>Proc.</th>
<th>DOFs</th>
<th>Global</th>
<th>Healthy</th>
<th>Faulty $\eta_k = 2$</th>
<th>Faulty $\eta_k = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Est</td>
<td>V-cycle</td>
<td>Est/V-cycle</td>
<td>Est/V-cycle</td>
</tr>
<tr>
<td>162</td>
<td>$4.5 \times 10^5$</td>
<td>0.97</td>
<td>2.31</td>
<td>1.01</td>
<td>0.53</td>
</tr>
<tr>
<td>750</td>
<td>$2.1 \times 10^5$</td>
<td>1.07</td>
<td>2.44</td>
<td>1.00</td>
<td>0.51</td>
</tr>
<tr>
<td>4,372</td>
<td>$1.2 \times 10^6$</td>
<td>1.26</td>
<td>2.71</td>
<td>1.01</td>
<td>0.53</td>
</tr>
<tr>
<td>29,480</td>
<td>$8.2 \times 10^6$</td>
<td>1.57</td>
<td>3.09</td>
<td>1.01</td>
<td>0.54</td>
</tr>
</tbody>
</table>

DOF: degree of freedom; HW: hierarchically weighted.

Table 3. Additional time spans (in second) of the adaptive DD recovery strategy for the recoupling criterion with $\sigma^{GRB}$, $k \in \{10^i, i = -2, -1, \ldots, 2\}$ and superman speedup $\eta_k = 2$ (top table), $\eta_k = 4$ (bottom table) for a faulty solution process $k_F = 7$; time-to-solution for the fault-free execution and additional time span for no-recovery; number of iterations for fault-free and no-recovery execution in parenthesis; number of faulty cycles $n_F$ necessary to satisfy the recoupling criterion and corresponding healthy cycles $n_H$ in parenthesis for recovery simulations.

<table>
<thead>
<tr>
<th>Proc.</th>
<th>DOFs</th>
<th>Fault-free</th>
<th>No recovery</th>
<th>$\kappa = 10^2$</th>
<th>$\kappa = 10^4$</th>
<th>$\kappa = 10^6$</th>
<th>$\kappa = 10^8$</th>
<th>$\kappa = 10^{10}$</th>
<th>$\kappa = 10^{12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\sigma^{GRB}$</td>
<td>Fault-free</td>
<td>No recovery</td>
<td>Healthy</td>
<td>Speedup $\eta_k = 2$</td>
<td>Speedup $\eta_k = 4$</td>
</tr>
<tr>
<td>162</td>
<td>$4.5 \times 10^5$</td>
<td>31.90 (14)</td>
<td>18.30 (22)</td>
<td>10.20 (4/2.4)</td>
<td>9.25 (5/3.0)</td>
<td>6.29 (6/3.7)</td>
<td>7.50 (7/4.2)</td>
<td>6.56 (8/4.8)</td>
<td></td>
</tr>
<tr>
<td>750</td>
<td>$2.1 \times 10^5$</td>
<td>33.50 (14)</td>
<td>16.80 (21)</td>
<td>10.40 (4/2.3)</td>
<td>9.20 (5/2.8)</td>
<td>8.24 (6/3.4)</td>
<td>7.06 (7/3.9)</td>
<td>4.31 (8/4.7)</td>
<td></td>
</tr>
<tr>
<td>4,372</td>
<td>$1.2 \times 10^6$</td>
<td>36.50 (14)</td>
<td>15.40 (20)</td>
<td>7.99 (4/2.1)</td>
<td>7.04 (5/2.7)</td>
<td>3.06 (6/3.1)</td>
<td>2.15 (7/3.7)</td>
<td>0.57 (8/4.1)</td>
<td></td>
</tr>
<tr>
<td>29,480</td>
<td>$8.2 \times 10^6$</td>
<td>41.70 (14)</td>
<td>14.70 (19)</td>
<td>6.83 (3/1.4)</td>
<td>5.06 (4/1.8)</td>
<td>3.73 (5/2.3)</td>
<td>2.38 (6/2.8)</td>
<td>0.73 (7/3.1)</td>
<td></td>
</tr>
</tbody>
</table>

$\sigma^{GRB}$: DOF: degree of freedom; DD: Dirichlet–Dirichlet.

Table 4. Additional time spans-time (in second) of the adaptive DD recovery strategy for the recoupling criterion with $\sigma^{LBR}$, $k \in \{10^i, i = -2, -1, \ldots, 2\}$ and superman speedup $\eta_k = 2$ (top table), $\eta_k = 4$ (bottom table) for a faulty solution process $k_F = 7$; time-to-solution for the fault-free execution and additional time span for no-recovery; number of iterations for fault-free and no-recovery execution in parenthesis; number of faulty cycles $n_F$ necessary to satisfy the recoupling criterion and corresponding healthy cycles $n_H$ in parenthesis for recovery simulations.

<table>
<thead>
<tr>
<th>Proc.</th>
<th>DOFs</th>
<th>Fault-free</th>
<th>No recovery</th>
<th>$\kappa = 10^2$</th>
<th>$\kappa = 10^4$</th>
<th>$\kappa = 10^6$</th>
<th>$\kappa = 10^8$</th>
<th>$\kappa = 10^{10}$</th>
<th>$\kappa = 10^{12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\sigma^{LBR}$</td>
<td>Fault-free</td>
<td>No recovery</td>
<td>Healthy</td>
<td>Speedup $\eta_k = 2$</td>
<td>Speedup $\eta_k = 4$</td>
</tr>
<tr>
<td>162</td>
<td>$4.5 \times 10^5$</td>
<td>31.90 (14)</td>
<td>18.30 (22)</td>
<td>10.20 (4/2.4)</td>
<td>9.25 (5/3.0)</td>
<td>6.28 (6/3.7)</td>
<td>7.51 (7/4.2)</td>
<td>7.78 (9/5.3)</td>
<td></td>
</tr>
<tr>
<td>750</td>
<td>$2.1 \times 10^5$</td>
<td>33.50 (14)</td>
<td>16.80 (21)</td>
<td>10.40 (4/2.3)</td>
<td>9.20 (5/2.8)</td>
<td>8.24 (6/3.4)</td>
<td>7.06 (7/3.9)</td>
<td>4.31 (8/4.7)</td>
<td></td>
</tr>
<tr>
<td>4,372</td>
<td>$1.2 \times 10^6$</td>
<td>36.50 (14)</td>
<td>15.40 (20)</td>
<td>7.99 (4/2.1)</td>
<td>7.04 (5/2.7)</td>
<td>3.06 (6/3.1)</td>
<td>2.15 (7/3.7)</td>
<td>0.57 (8/4.1)</td>
<td></td>
</tr>
<tr>
<td>29,480</td>
<td>$8.2 \times 10^6$</td>
<td>41.70 (14)</td>
<td>14.70 (19)</td>
<td>6.83 (3/1.4)</td>
<td>5.06 (4/1.8)</td>
<td>3.73 (5/2.3)</td>
<td>2.38 (6/2.8)</td>
<td>0.73 (7/3.1)</td>
<td></td>
</tr>
</tbody>
</table>

$\sigma^{LBR}$: DOF: degree of freedom; DD: Dirichlet–Dirichlet.
efficiency of 62% and for the V-cycle iteration in the healthy domain from 2.31 s to 3.09 s with a parallel efficiency of 75%. Note that in the estimator the time for the coarse grid solver has a much larger influence than for the run-time of the V-cycle. On the finest level a residual computation and a scaling are necessary and on each other level, but the coarsest only one scaling is applied in the estimator. While for the V-cycle, six smoothing steps and one residual evaluation are necessary on each level. The run-times of the V-cycle in the healthy domain are similar to the times of one global V-cycle, since the problems have almost the same size; compare, for example, with Table 3, column “fault-free” and the times divided by the number of iterations. We observe perfect scalability of the V-cycle and error estimation in $\Omega_F$ for both superman factors. This is obvious, since the size of the faulty domain remains constant in the weak scaling and has shorter communication paths, that is, improved latency for collective and nearest-neighboring communication. When comparing the timings for superman speedup $\eta_h = 2$ with $\eta_h = 4$, the additional compute power (by a factor of 2) is perfectly observed in the timings. Also when comparing the run-times of the V-cycles in the healthy and faulty domain, they are reduced by a factor of 2 and 4 with respect to the superman speedup. For increasing problem size, this speedup factor even grows due to the increasing run-time of the V-cycle in the healthy domain. Summarizing, the scaling experiment shows that the HW estimator of the global problem and of the faulty subproblem adds less than 53% overhead to one V-cycle application, respectively, and therefore, is suitable for massively parallel computations within the recovery algorithm.

5.2.2. Performance of the recoupling criterion. In the following weak scaling experiments in Tables 3 and 4, we study the robustness of the recoupling bounds for variations in $\kappa$ within the recovery. We present here run-times spent in the V-cycle for the global problem and for the subproblems in the recovery. We exclude the time measurements necessary for setting up the recovery in the faulty domain and the error estimation of the global problem. The run-times for the estimation are shown in Table 2. The distinction is made, since we differentiate between computation and controlling of the V-cycle iterations.

5.2.2.2. Performance of the fault-free and no-recovery simulation. In the fault-free execution, the iterations necessary to reach the stopping criterion stay at 14 within the weak scaling (in parenthesis in the column “fault-free”). The time increases slightly from 31.9 s to 41.7 s, which represents an excellent parallel efficiency of more than 76%. In the case, when no-recovery of the data is performed after the fault and we continue with global V-cycles after reinitialization, the number of iterations necessary to satisfy the global stopping criterion increases up to 22 iterations (in parenthesis in the column “no-recovery”). Asymptotically (for large problems), the additionally required number of iterations reduces due to the shrinking relative size of the lost data such that the delay in convergence decreases from 18.3 s to 14.7 s.

5.2.2.2. Setup of the adaptively controlled recovery simulation. When we apply the adaptively controlled recovery in the event of a fault, we report on the additional time spans which are necessary to satisfy the global criterion. Negative numbers indicate that the overall run-time actually improves in the case of a fault. The reason for this is that in an execution without fault the number of iterations necessary to satisfy the global stopping criterion is an integer value. In the recovery, we allow a fractional iteration number. In parenthesis, we show the number of cycles $(n_{F}/n_I)$ performed in the recovery in faulty and healthy domain. We apply in the faulty subdomain in addition to the V-cycles also the error estimator. Therefore, we observe a reduced iteration number, as expected, that is performed in the faulty domain in comparison to the cycles in the healthy domain. Asymptotically, the factor between $n_{F}$ and $n_{I}$ increases, since the run-time of one V-cycle in the healthy domain increases, while the computational work in the faulty domain remains constant.

In the recoupling criterion for the recovery, it is essential to identify a good control parameter $\kappa$. In our extensive experiments, we have found good factors in a relatively wide numerical range of $\kappa \in \{10^i, i = -2, -1, \ldots, 2\}$ depending on the problem size and $\eta_h$. By these choices, $n_F \in [4, 9]$ that is reasonable for the smallest problem size. The recoupling thresholds for GRB and LRB are presented in the first column in the Tables 3 and 4 (bottom tables), respectively. While the thresholds of LRB are asymptotically almost constant, GRB increases by a factor of 2 to 3. By the bounds in (27), the scaling factor between $\sigma^{GRB}$ and $\sigma^{RB}$ increases from a factor of $\sqrt{162} \approx 13$ to $\sqrt{29,480} \approx 172$ within the weak scaling. This is a rather moderate difference such that the number of iterations necessary to satisfy GRB decreases by one iteration for the largest problem size, while the number of iterations for LRB remains constant when considering a fixed $\kappa$.

5.2.2.3. Performance with superman speedup $\eta_h = 2$ and $\eta_h = 4$. For both superman speedup factors, optimal run-times cannot be obtained for the smallest two problem sizes and a small overhead of around 2 s and 6 s need to be accepted, respectively. The recovery is either insufficient and suggests an early recoupling or the recovery time takes too long. Here, we observe again effects of under- and over-solving. By fixing the interface data in the recovery, the error in the healthy domain can only be reduced slightly in comparison to the error in the faulty domain. Hence, the error in the healthy domain already saturates, while the error in the faulty domain is still large. A recoupling due to this saturation is not incorporated in the stopping criterion (14), since the remaining large error components in the faulty domain can still pollute the whole domain and delay the convergence. A higher speedup factor would be
necessary to reduce this delay further. For larger problems, this effect is reduced, and we can solve both subproblems to a higher accuracy without delaying the convergence. We can then slightly over-solve both problems, since the remaining large algebraic error components at the interface can be treated efficiently by the global V-cycles after the fault. Therefore, $\kappa$ can be chosen in a larger range for both recoupling bounds and superman speedup factors. For instance, for the largest problem size and the recoupling criterion with LRB and $\kappa = 10^{-2}$, four V-cycles can be executed in the healthy domain without delaying the convergence.

5.2.2.4. Performance comparison of the superman speedup factors. The control parameter $\kappa$ can be chosen larger for $\eta_h = 4$ than for $\eta_h = 2$, i.e. we can allow asymptotically a larger error tolerance in case of recoupling for the faster superman. The reason for this is that for $\eta_h = 4$ the recoupling is performed earlier than for $\eta_h = 2$. Therefore, to achieve the same accuracy as the fault-free execution in the same number of iterations, more global V-cycles can be applied after the recovery and can reduce a larger error sufficiently without delaying the convergence. For instance, consider the largest problem size and the criterion with GRB, then, for $\eta_h = 4$, $\kappa$ can be chosen smaller or equal to $10^0$, while for $\eta_h = 2$, $\kappa \leq 10^{-2}$ is necessary without delaying the convergence. This holds similarly true for LRB. Note, a too small $\kappa$ delays the convergence due to over-solving.

5.2.2.5. Performance comparison of the recoupling bounds. When increasing the problem size and the relative ratio between faulty and healthy domain decreases, a lower approximation quality of the faulty subproblem is required. This reduced accuracy requirement is adjusted by GRB in the recoupling criterion. For instance, for $\eta_h = 4$, an approximation of the faulty subproblem by six V-cycles is the minimal number of iterations to obtain the best possible run-times for the smallest problem size. For the largest problem size, these are five V-cycles. Hence, for $\kappa = 10^0$, GRB yields the minimal required $n_F$ to obtain optimal run-times. By using the tighter LRB in the recoupling criterion, the faulty subproblem is slightly over-solved.

Summarizing, both recoupling bounds perform similarly well with respect to the speedup factors and can yield optimal run-times. Asymptotically, $\kappa$ factors that are good for moderate sizes yield robust and efficient recoupling bounds and obtain optimal run-times. Minimal run-times are observed for both recoupling bounds with $\kappa \leq 10^{-1}$ for a superman speedup $\eta_h = 4$ and with $\kappa = 10^{-2}$ for $\eta_h = 2$. Asymptotically, optimal run-times without delay in convergence are also obtained for GRB and LRB with $\kappa \leq 10^0$ for a superman speedup $\eta_h = 4$. For $\eta_h = 2$, this effect can only be observed for LRB with $\kappa = 10^{-1}$.

5.2.3. Local error distribution. In this section, we study the local error distribution in the event of a fault and after applying a recovery strategy in detail. We reconsider the experiments of the previous Section 5.2.2. We recall that we can relate the error contribution of each process by (21) with a portion of the global estimated algebraic error. We study this process-wise estimated algebraic error for the problem sizes with $4.5 \times 10^8$ DOFs (left) and with $8.2 \times 10^{10}$ DOFs (right) in Figure 9. For the larger problem, we only depict the error contribution of each 100th processor.

5.2.3.1. Error distribution before and directly after the fault. In the top row, the process-wise error is presented before the fault (black) and after the fault (red). While the algebraic error for the smaller problem is equally distributed, we observe large oscillations by more than two orders of magnitude for the larger problem. The reason for this is that for a zero initial guess, the difference to the exact solution is small for some DOFs, and hence, the algebraic error is small. For a large process count, it is possible that a subdomain with exclusively small or large algebraic error components are assigned to a single process. Then, such oscillations as seen in the illustration can occur. By the crash, a huge local error is introduced in the faulty domain. This error is one to two orders of magnitude smaller for the larger than for the smaller problem. The DOFs per process stay constant in the scaling, but the size of the assigned subdomain is by a factor of $1/2^3$ smaller. Thus, the portion of the $L^2$-error per process is also reduced by a factor of $(2^3)^{1/2}$. As observed in the previous Section 5.2.2, this smaller error reduces the delay in the global termination in case of a failure. We also include in these illustrations the recoupling bound LRB for $\kappa = 10^1$. Note, the chosen $\kappa$ is not optimal, as seen by the study in the previous section for the smaller problem. However, it does not delay the convergence in case of the larger problem size. It shows the positive effect of the LRB strategy on the recovery by the tighter bound.

5.2.3.2. Comparison of the error distribution after recovery with a fault-free and a no-recovery execution. In the bottom row, we study the error distribution when applying the adaptively recovery strategy in case of a fault to quantify its efficiency. We choose for the recovery the LRB strategy with $\kappa = 1.0$ and superman speedup $\eta_h = 4$. We depict the error distribution after the recovery and an additional V-cycle to account for a possible error propagation of the remaining large error components to the whole domain. We also include in the illustrations the error distribution in case of a fault-free and a no-recovery execution. To make these runs comparable, we proceed as follows. We identify the number of performed V-cycles in the healthy domain in the recovery in Table 4. These are 1.7 V-cycles for the smaller problem and 1.2 cycles for the larger problem. We assume that one global V-cycle can be executed as fast as one V-cycle in the healthy domain such that in case of a fault-free and no-recovery execution, 2 and 1 global V-cycles could be performed in the time of the recovery. We approximate these numbers with the closest integer
value. Thus, we compare the error distribution after the recovery and one V-cycle with the error distribution of the fault-free and no-recovery execution, when 3 and 2 additional V-cycles have been performed after the fault. In the fault-free execution, the error is equally reduced for each process by the additionally applied V-cycles. The large local error introduced by the fault pollutes the whole domain in case of the no-recovery execution such that the process-wise error is by around 5 orders of magnitude larger than in case of the fault-free execution for the smaller problem. In the larger problem, this difference is reduced to 3 orders of magnitude in average. The largest error components are still observed for neighboring processes of the faulty process.

The recovery strategy reduces the local large error of the faulty process to a moderate size. For the smaller problem, the process-wise error is larger than in the fault-free case. This difference yield a convergence delay (see Table 4). For the larger problem, there exist only small differences between the error obtained after the recovery strategy and in the fault-free case. The largest difference is observed for the faulty process shown by an error peak. In this case, the recovery strategy efficiently reduces all error components comparable to the error of the fault-free execution such that the global convergence is not delayed.

5.3. Multiple faults

In this section, we apply the adaptively controlled algorithm to a setting in which multiple faults occur to demonstrate that the algorithm is robust with respect to fault frequency variations. For a detailed survey on models on fault probabilities and MTBF, we refer the reader to Dongarra et al. (2015). Motivated by geophysical simulations (see, e.g. Bauer et al., 2016), we consider problems posed on a spherical shell, namely,

$$
\Omega = \{ x \in \mathbb{R}^3 : 0.55 < |x| < 1 \},
$$

where $| \cdot |$ denotes the Euclidean norm. The domain $\Omega$ is discretized by an initial mesh (see Figure 10) and is then uniformly refined to full resolution. Unlike in the previous experiments, we change the exact solution of the model problem (1) to

$$
u = \sin((x + \sqrt{2}y)\pi)\sinh(\sqrt{3}\pi).$$
For this example, we have a homogenous right-hand side and prescribe the Dirichlet boundary conditions according to the exact solution. Again, we consider the estimated algebraic error reduction by V-cycles, but now, we inject one fault after $k_1^F = 5$ iterations and another one after $k_2^F = 10$ iterations, affecting two macro-terahedrons, as illustrated in Figure 10. To guarantee a load-balanced problem, we assign again to each MPI process the same number of tetrahedrons of the input mesh. In Table 5, we study a sequence of successively refined input meshes to evaluate the performance of the recovery strategy. The coarsest level grows from $3.1 \times 10^4$ to $3.4 \times 10^8$ tetrahedrons in the weak scaling. We choose a superman acceleration of $\eta_s = 4$, since this has already shown favorable results in the context of our study of different recoupling bounds in Section 5.1 and 5.2. Moreover, the higher superman factor should also account for the late fault such that a recovery with a small delay is possible. We use the bound $\sigma^{LRB}$ in the recoupling criterion and vary $\kappa \in \{10^i, i = -2, -1, \ldots, 1\}$. Our largest simulation in this setup is executed on 245,766 processes. We again present the run-times for V-cycles and the additional time spans for the no-recovery and the recovery execution.

Note that the run-times in the fault-free case are characterized by a decrease of the number of iterations for a higher mesh resolution, that is, the global stopping criterion (11) with $TOL = 10^{-13}$ is reached with fewer iterations on finer meshes. This is caused by the discretization of the spherical shell geometry. A coarser initial mesh gives rise to less favorable element shapes that result in a worse multigrid convergence rate. A detailed study goes beyond the current contribution and has no direct effect on the efficiency of the fault recovery. For interested readers, we refer to Gmeiner et al. (2013) and Bauer et al. (2017).

Two failures lead to an increase in the total run-time of up to 59%, when no recovery is used. The automatically controlled recovery with parameters $\kappa = 10^{-1}$ and $\kappa = 10^{-2}$ achieves the best results with a small overhead. We want to emphasize that for a late failure at, for example here $k_2^F = 10$ the recovery must be very fast in order to achieve an optimal run-time result. In the case of 3,846 processors, for example, only the time of 5 cycles remains until the global stopping criterion should be satisfied. In this extreme case, a small overhead of a few seconds can be considered as excellent. The recovery still constitutes a significant acceleration in the case of this fault situation.

Figure 10. Input mesh for different spherical scheme discretizations. The location for the two faulty processors in the initial mesh is marked by a red and blue tetrahedron.

Table 5. Additional time span (in second) of the adaptive DD recovery strategy for recoupling criterion with $\sigma^{LRB}$, $\kappa \in \{10^i, i = -2, -1, \ldots, 1\}$, and superman speedup $\eta_s = 4$ for a faulty solution process $k_1^F = 5$ and $k_2^F = 10$; time-to-solution for the fault-free execution and additional time span for no-recovery execution; number of iterations for fault-free and no-recovery execution in parenthesis; number of faulty cycles $n_f$ for the first recovery after $k_1^F = 5$ and $n_f$ for the second fault after $k_2^F = 10$ necessary to satisfy the stopping criterion in parenthesis for the recovery simulations.

<table>
<thead>
<tr>
<th>Proc.</th>
<th>DOFs</th>
<th>Fault-free</th>
<th>No recovery</th>
<th>$\kappa=10^1$</th>
<th>$10^0$</th>
<th>$10^{-1}$</th>
<th>$10^{-2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>$1.7 \times 10^8$</td>
<td>59.0 (24)</td>
<td>20.5 (33)</td>
<td>5.9 (3.4)</td>
<td>1.5 (5.8)</td>
<td>2.2 (7.10)</td>
<td>2.3 (9.11)</td>
</tr>
<tr>
<td>486</td>
<td>$1.3 \times 10^9$</td>
<td>40.3 (17)</td>
<td>21.4 (27)</td>
<td>7.5 (2.7)</td>
<td>5.0 (3.9)</td>
<td>4.3 (4.10)</td>
<td>4.3 (5.12)</td>
</tr>
<tr>
<td>3,846</td>
<td>$1.1 \times 10^{10}$</td>
<td>38.3 (15)</td>
<td>22.9 (24)</td>
<td>8.1 (2.8)</td>
<td>3.4 (3.11)</td>
<td>1.7 (4.11)</td>
<td>1.9 (5.13)</td>
</tr>
<tr>
<td>30,726</td>
<td>$8.5 \times 10^{10}$</td>
<td>42.3 (15)</td>
<td>25.3 (24)</td>
<td>7.6 (2.7)</td>
<td>1.9 (3.10)</td>
<td>-0.2 (4.10)</td>
<td>2.4 (5.12)</td>
</tr>
<tr>
<td>245,766</td>
<td>$6.9 \times 10^{11}$</td>
<td>53.0 (15)</td>
<td>27.8 (23)</td>
<td>6.5 (3.8)</td>
<td>0.5 (4.10)</td>
<td>1.1 (5.10)</td>
<td>0.6 (6.12)</td>
</tr>
</tbody>
</table>

DOF: degree of freedom; DD: Dirichlet–Dirichlet.
Note, that a larger superman acceleration factor can help to reduce this overhead further.

6. Conclusion
This article presents a roll-forward technique that combines online global recovery methods within multigrid correction schemes and an adaptively steered synchronization. The online global recovery consists of asynchronous computations in the faulty and healthy domain. The local recovery of the faulty domain is additionally accelerated by the superman strategy. The recoupling of the faulty and healthy subproblem is controlled by a stopping criterion for computations in the faulty domain. This stopping criterion is motivated by an error estimator that uses the underlying hierarchical multigrid structure by weighted sums of residuals and is especially suited for large-scale computations. We studied the efficiency of the stopping criterion for two bounds that differ in their weighting by several test cases on the BlueGene/Q peta-scale system JUQUEEN with up to $6.9 \times 10^{11}$ DOFs on 245,766 parallel processes. It is shown that the algorithm presented can use the automatic recoupling by a choice of a suitable stopping criterion and recover from faults with no additional overhead in terms of run-time.

Declaration of Conflicting Interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the German Research foundation (DFG) [Priority Programme 1648 “Software for Exascale Computing” (SPPEXA); WO671/11-1]; and the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUQUEEN at Jülich Supercomputing Centre (JSC).

References


**Author biographies**

**Markus Huber** studied mathematics at the Technical University of Munich. He is a PhD student at the Chair of Numerical Mathematics by professor Barbara Wohlmuth. His research is mainly focused on fast, robust multigrid methods, fault-tolerant algorithms for high-performance applications and computationally efficient methods in geophysics.

**Ulrich Rüde** heads the Chair for Simulation at the University Erlangen-Nuremberg and is the leader of the Parallel Algorithms Team at the Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique (CERFACS) in Toulouse. He studied mathematics and computer science at the Technical University of Munich and the Florida State University. He has been a visiting professor at the University of Colorado, Boulder and the National University of Singapore. His research interests lie in computational science and engineering, with a focus on parallel numerical algorithms and high-end computing, in particular computational fluid dynamics, multilevel methods, and software engineering for high-performance computing. He is a fellow of the Society of Industrial and Applied Mathematics.

**Barbara Wohlmuth** is the head of the Chair for Numerical Mathematics at the Technical University of Munich. She studied mathematics in Munich and Grenoble. As a PostDoc, she stayed at the Courant Institute of Mathematical Sciences at New York University and as a visiting professor in France and Hong Kong. She works on the numerical simulation of partial differential equations, in particular, on advanced discretization techniques, multiscale iterative solvers such as large-scale hybrid multigrid methods, the mathematical modeling of coupled multi-field problems, and uncertainty quantification. As author of over 100 scientific publications, she serves on the Editorial Board of several international journals with special focus on numerical analysis and computational sciences and engineering. She is elected member of the DFG Review Board 312 for Mathematics and member of the Bavarian Academy of Sciences and the European Academy of Sciences. In 2012, she was awarded the prestigious Gottfried Wilhelm Leibniz Prize of the German Research Foundation (DFG).