FAULT RESILIENT DOMAIN DECOMPOSITION PRECONDITIONER FOR PDES

KHACHIK SARGSYAN*, FRANCESCO RIZZI†, PAUL MYCEK‡, COSMIN SAFTA§, KARLA MORRIS¶, HABIB NAJM‖, OLIVIER LE MAÎTRE**, OMAR KNIO††, AND BERT DEBUSSCHERE‡‡

Abstract.
The move towards extreme-scale computing platforms challenges scientific simulations in many ways. Given the recent tendencies in computer architecture development, one needs to reformulate legacy codes in order to cope with large amounts of communication, system faults and requirements of low-memory usage per core.

In this work, we develop a novel framework for solving partial differential equations (PDEs) via domain decomposition that reformulates the solution as a state-of-knowledge with a probabilistic interpretation. Such reformulation allows resiliency with respect to potential faults without having to apply fault detection, avoids unnecessary communication and is generally well-suited for rigorous uncertainty quantification studies that target improvements of predictive fidelity of scientific models. We demonstrate our algorithm for one-dimensional PDE examples where artificial faults have been implemented as bit-flips in the binary representation of subdomain solutions.

Key words. Domain decomposition, fault resilience, ℓ_1 minimization

AMS subject classifications. 35J25, 65N55, 62J05

*Sandia National Laboratories, 7011 East Ave, MS 9051, Livermore, CA 94550 (ksargsy@sandia.gov).

†Sandia National Laboratories, Livermore, CA (fnrizzi@sandia.gov).

‡Duke University, Durham, NC (paul.mycek@duke.edu).

§Sandia National Laboratories, Livermore, CA (csafta@sandia.gov).

¶Sandia National Laboratories, Livermore, CA (knmorri@sandia.gov).

‖Sandia National Laboratories, Livermore, CA (hnnajm@sandia.gov).

**Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur, Orsay, France (olm@limsi.fr).

††Duke University, Durham, NC (omar.knio@duke.edu).

‡‡Sandia National Laboratories, Livermore, CA (bjdebus@sandia.gov).

1. Introduction. Given the trends in computer architecture development, scientific simulations on future extreme- scale platforms will face many challenges such as the need to operate with relatively low memory per core, cope with costly data movement, make use of heterogeneous hardware, scale to very large numbers of cores, and deal with hardware and software faults, among many other challenges [38]. Overcoming these challenges is essential for many science applications in terms of enabling simulations on future extreme-scale architectures. The current paper addresses the issue of resilience against system faults.

The term *system faults* covers a wide range of hardware and software faults, which can be categorized as *soft* or *hard faults* [5]. Hard faults result in termination of the program that is executed, *e.g.* due to a node crash. Soft faults, such as bit-flips in memory, do not cause immediate program termination, but can lead to faulty results or a program crash further down. Soft faults are often hard to detect, in which case they are labeled *silent* faults.

Fault tolerance in high performance computing has been the subject of research for almost three decades, with increasing intensity as capability simulations are moving into the realm of tens of thousands of processors where the system mean time to interrupt (MTTI) drops to a mere couple of hours [8]. With increasing numbers of transistors per chip, error rates are expected to grow significantly [32,37,39]. Various approaches have been developed to study system reliability, ranging from graph-based analyses [36] to empirical fault injection [31]. The impact of system faults ranges from negligible to major, depending on the software and the type of fault [31].

By far the most commonly used fault-tolerance approach is checkpointing, either onto physical storage or diskless through checksums. While being conceptually straightforward and robust, checkpointing does introduce a sizeable overhead and does not scale well with the system and simulation size [4]. This scaling problem arises from the need for wide-range (often global) aggregation of information to construct the checkpoint data and is, therefore, an imminent limitation even if the current MTTI can be maintained for future computing systems. With current technologies, the time to handle resilience is expected to exceed the MTTI of top supercomputers in near future [9, 31]. More recent work has focused on local, rather than global, checkpointing and recovery [43]. Alternative, emerging approaches are various forms of algorithm-based fault tolerance (ABFT) [4, 15, 18, 20], effective use of state machine replication [21] or process-level redundancy [39], and algorithmic error correction code [33]. Many other approaches for resilience in High Performance Computing have been developed. For a more comprehensive overview of the many sources of system faults and the fast-growing body of research to achieve resilience, we refer the reader to the review papers by Cappello *et al.* [8, 10],

While the many developments in the community towards resilience against system faults are promising, almost all of the approaches rely on being able to detect the faults in order to mitigate them. This is particularly a problem for handling silent faults. Also, most approaches are geared towards handling just one type of fault, *e.g.* resilience against soft faults, but not hard faults. In this paper, we present a novel algorithmic approach towards achieving resilience against both soft and hard faults, without the need to explicitly detect them. Our algorithm is based on an overlapping domain decomposition framework. It represents the solution as a *state-of-knowledge*, and updates this state in a fault-resilient manner, based on samples of the solution on the subdomains.

Domain decomposition methods [28, 41], besides being attractive from a parallel

computing viewpoint, are very well positioned from a resiliency standpoint, as they localize faults, perhaps with the exception of faults produced during communication. With the increased emphasis on parallel computing, domain decomposition methods provide a convenient avenue to *divide-and-conquer*. However, while the scalability and convergence of such methods have been extensively studied in both PDE [7,19,34,42, 44] and linear solver [2,24–26] contexts, there are not – to the best of our knowledge – many studies considering fault resilience specific to domain decomposition algorithms. Chen *et al.* [15] proposed an algorithm-based recovery method for iterative system solvers to enable resilience to fail-stop failures based on data partitioning tailored to the characteristics of the iterative scheme, while Larson *et al.* [30] achieved fault tolerance by combining solutions on sparse grids. Both approaches can in principle be reformulated in a domain decomposition paradigm, but with a distinct flavor of fault-detection or redundancy present.

The approach presented in this paper iteratively advances the solution state by constructing maps that relate the solutions at the subdomain boundaries to each other, and then solving a fixed point problem to get the updated solution at the subdomain boundaries. Resilience is obtained by relying on a novel – in this context – robust regression approach to learn the mappings between the solution at the subdomain boundaries. In this algorithm, failed runs due to node or software failures show up as missing data, and corrupted data due to bit-flips or other silent errors shows up as data noise in the regression. The local maps are parametrized as polynomial expansions, similar to the local Polynomial Chaos method developed in [14]. Our approach, however, allows for nonlinear problems, and tackles resilience against soft and hard faults as the primary objective.

The current paper focuses primarily on the formulation of the algorithm and demonstrates its strong resilience to soft faults in the application to 1D linear and nonlinear differential equations. In linear problems, convergence is reached in just one iteration, for almost all cases. For non-linear problems, the number of iterations required to reach convergence depends only very mildly on the error rate. The application of the algorithm to 2D problems along with a demonstration of resilience against both hard and soft faults will be treated in [35].

The paper is organized as follows. In Section 2 we develop the algorithm, focusing on the one-dimensional case, while detailing and testing the fault-resilient boundary-to-boundary map learning procedure. Then, Section 3 demonstrates the algorithm on both linear and non- linear PDE test cases. We relegate conclusions and a discussion of potential generalizations to Section 4.

2. Algorithm formulation.

2.1. General approach. In this work, we reformulate PDE-based simulations in terms of a description of the current state-of-knowledge about the *true* solution. In this context, the true solution is the solution one would obtain with the chosen numerical method in the absence of any faults in the system. Starting from a range around a reasonable mean value that captures the initial knowledge about the solution, targeted simulations are used to refine the knowledge about the solution until the state-of-knowledge converges to the true answer with sufficient confidence. As such, there is no need to characterize all types of system faults that can occur in a simulation; one focuses solely on the information that a simulation provides, and on using it to reduce the uncertainty in the knowledge about the true solution. Our iterative solution refinement approach relies on domain decomposition and combines information from subdomain PDE solutions that can be obtained with regular PDE solvers. As such,

this approach can be seen as a domain-decomposition *preconditioner* for the resilient solution of partial differential equations (PDEs).

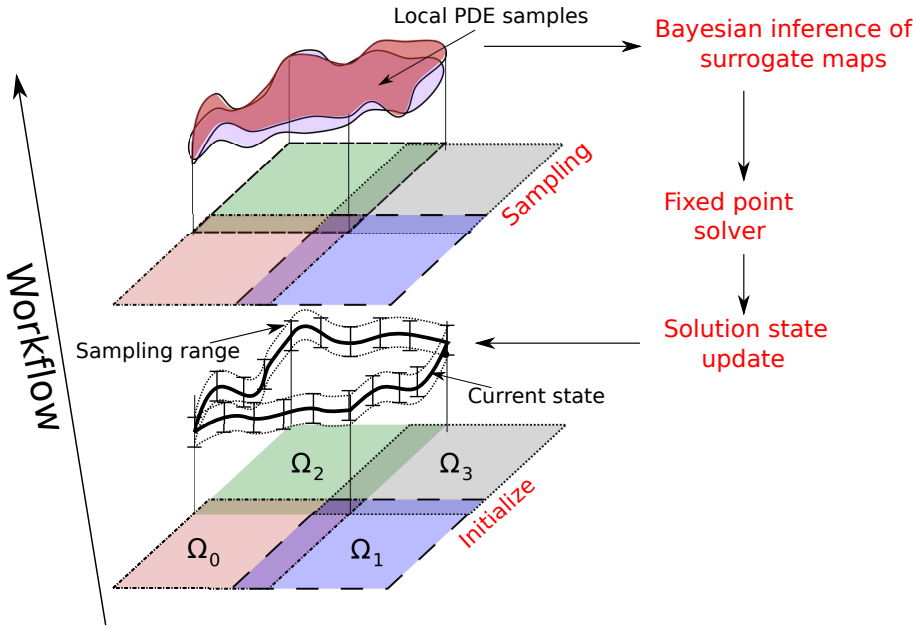


FIG. 1. *Workflow schematic of the approach for a 2-dimensional, 4 overlapping subdomain case.*

As an example of a generic elliptic PDE, consider the form

$$\mathcal{L}y(\boldsymbol{x}) = h(\boldsymbol{x}) \tag{2.1}$$

with a differential operator $\mathcal{L}$, and a given forcing function $h(\boldsymbol{x})$, for $\boldsymbol{x} \in \Omega$, where Ω is an open and bounded subset of $\mathbb{R}^n$. Consider the solution $y(\boldsymbol{x})$ for a Dirichlet boundary condition $y(\boldsymbol{x})|_{\boldsymbol{x}\in\partial\Omega} = y_{\partial\Omega}$ on the boundary $\partial\Omega$ of the problem domain Ω. Generally, an overlapping domain decomposition consists of N *subdomains* $\{\Omega_i\}_{i=1}^N$ with corresponding boundaries $\{\partial\Omega_i\}_{i=1}^N$ such that $\cup_{i=1}^N \bar{\Omega}_i = \bar{\Omega}$, and for each Ω_i there is at least one overlapping neighbor Ω_j such that $\Omega_i \cap \Omega_j \neq \emptyset$. The algorithm relies on a representation of subdomain boundary conditions that captures a mean state-of-knowledge and an uncertainty range around it. This representation is updated iteratively, starting from an initial guess. In other words, the object of interest is the set of solution fields $y_i \equiv y(\boldsymbol{x})|_{\boldsymbol{x}\in\partial\Omega_i}$ at each subdomain boundary, denoted by $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$. With the complete knowledge of the true value of $\boldsymbol{y}$, denoted by $\tilde{\boldsymbol{y}}$, a single solve per subdomain would recover the full solution over the original domain Ω. In order to keep the focus on the subdomain solution faults, we consider this last step fault-free and skip its consideration. In principle, this last step can be made fault-free with simple redundancy, essentially adding a fixed number to the total required subdomain solves, or by implementing within *e.g.* a selective reliability framework [5,10] with targeted, highly-reliable solves at the last step. Our algorithm obtains iteratively improving, *i.e.* sequentially range-reducing, representations of $\tilde{\boldsymbol{y}}$. Each subdomain Ω_i will include, due to overlaps, a set of k_i boundaries $\{\partial\Omega_{j|i}\}_{j=j_1,\ldots,j_{k_i}}$ of other subdomains, see Figure 1. Consider the k_i maps from the boundary conditions on $\partial\Omega_i$ to boundaries of other subdomains that are internal to

Ω_i, *i.e.* maps of form $\partial\Omega_i \mapsto \partial\Omega_{j|i}$ for $j = 1, \ldots, k_i$. These maps are induced by the solution of the elliptic PDE on Ω_i with Dirichlet boundary conditions:

$$f_{i\to j}(y_i) = y_{j|i} \text{ for } j = 1, 2, \ldots, k_i, \tag{2.2}$$

where $y_{j|i}$ denotes the restriction of y_j to $\partial\Omega_{j|i}$, *i.e.* the part of the boundary $\partial\Omega_j$ that is inside the subdomain Ω_i. The system of such *boundary maps* for all subdomains $i = 1, 2 \ldots, N$, given the global boundary conditions $y(\boldsymbol{x})|_{\boldsymbol{x}\in\partial\Omega}$ can be written in a fixed-point form $\boldsymbol{y} = \mathcal{F}\boldsymbol{y}$ and is satisfied by the true solution $\tilde{\boldsymbol{y}}$. Note that these boundary maps $f_{i\to j}$ are purely on y-values and are simply restrictions of the subdomain solutions at the corresponding boundaries. While general non-linear solvers can in principle solve the system, it would require global communication and repeated online subdomain solutions in order to evaluate $f_{i\to j}$ for each iteration in the solution of this system. Instead, we construct *surrogate maps*, *i.e.* approximations $g_{i\to j}$ for each outer-to-inner boundary map $f_{i\to j} \approx g_{i\to j}$, employing Bayesian techniques and a set of *training* solutions pertaining to each subdomain. For linear PDEs, the boundary maps are linear as well, as shown in the Appendix, and linear surrogate maps are exact. Thus one can view the surrogate construction simply as a learning of the linear coefficients of the boundary maps. In general, for non-linear problems, the introduction of linear surrogate maps will carry an additional source of discrepancy, due to the linear approximation of a generally non-linear map. However, we will introduce an iterative approach with systematic reduction of the range over which the surrogate map is constructed, thus rendering linear surrogate increasingly more accurate.

The surrogate map is constructed with Bayesian machinery, using a number of training runs. The surrogate range, *i.e.* the range of sampling of training runs, is chosen according to a current state-of-knowledge representation that reflects the magnitude of the uncertainty around the current state of the solution. The PDE is then solved on each subdomain for sampled values of its current boundary condition distributions. The resulting subdomain solution samples feed into a Bayesian inference of surrogates that relate the subdomain boundary conditions to each other. Intersection of these surrogate maps, *i.e.* the solution of the system $\{g_{i\to j}(y_i) = y_{j|i}\}_{i=1,\ldots,N}^{j=1,\ldots,k_i}$ provides a new sample of the subdomain boundary conditions that serves as an approximation of the true solution $\tilde{\boldsymbol{y}} = \{\tilde{y}_i\}_{i=1,\ldots,N}$. When subdomain solves fail (*e.g.* due to nodes crashing), the inference proceeds with fewer samples, and the associated loss of information merely results in a locally higher uncertainty. Erroneous subdomain solves (*e.g.* due to random bit-flips) are either rejected by a properly chosen prior distribution on the solution that strips away outrageously large errors, or accounted for by the Bayesian noise model. The schematics shown in Figures 1 and 2 demonstrate the workflow of the algorithm.

To describe the algorithm from a different perspective, note that the goal is to solve the fixed-point problem $\boldsymbol{y} = \mathcal{F}\boldsymbol{y}$ constructed from the intersection of the maps that relate the solutions at the subdomain boundaries to each other. While the general non-linear operator $\mathcal{F}$ is hard to compute, we first 'learn' an approximation $\mathcal{G}^{(T)} \approx \mathcal{F}$ at the T-th iteration that stems from surrogate maps, requiring M training, subdomain solves. Then, we find the fixed-point solution using the simple surrogate $\boldsymbol{y} = \mathcal{G}^{(T)}\boldsymbol{y}$, in order to arrive at the solution estimate $\boldsymbol{y}^{(T+1)}$ at the next iteration $T + 1$. Next, we shrink the surrogate construction range and obtain a more accurate surrogate map, $\mathcal{G}^{(T+1)}$, before repeating iteratively. With this formulation, one could draw clear parallels to classical Schwarz algorithms [12, 19, 41], where the solution iterations $\boldsymbol{y}^{(T+1)} = \mathcal{F}\boldsymbol{y}^{(T)}$ lead to the fixed point solution $\boldsymbol{y}$ without explicitly computing $\mathcal{F}$, or to Aitken-like acceleration techniques for linear operators [22, 23], where

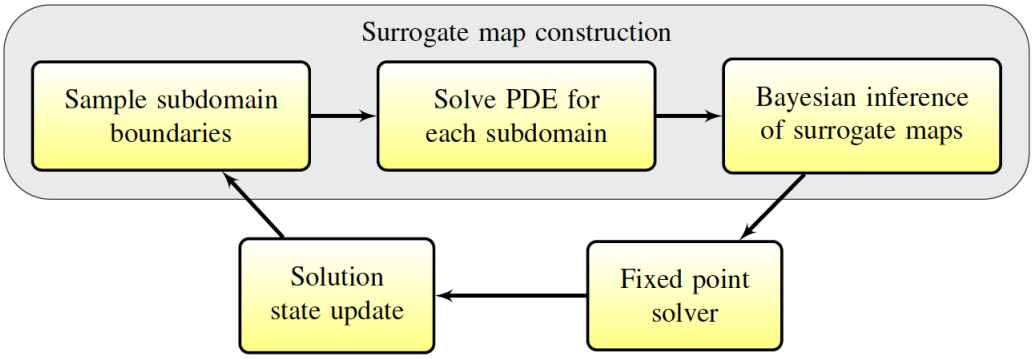


FIG. 2. *Schematic of one complete iteration of the algorithm.*

$\mathcal{F}$ is linear and easily found by two training subdomain solves in the 1D case. Our formulation, however, provides a general framework for non-linear operators, and a resilient way to construct the surrogate maps; it also embeds a state-of-knowledge, information-based interpretation in the iteration scheme. Note that the classical additive Schwarz algorithm converges if the operator $\mathcal{F}$ is a *contraction*, *i.e.* its spectral radius is less than 1. This means that if the surrogate is exact, *i.e.* $G^{(T)} = \mathcal{F}$, then the convergence of the proposed algorithm is equivalent to that of the classical additive Schwarz algorithm for the same PDE. For linear problems, $G^{(T)} = \mathcal{F}$ holds for any T, therefore our algorithm will converge after the very first iteration, unless a large enough number of faulty solutions renders the surrogate map $G^{(T)}$ inexact. Across several successive iterations, this is extremely unlikely as can be seen further in this paper. For non-linear problems convergence is generally not guaranteed. However, our approach constructs successively more accurate linear surrogates to the boundary-to-boundary maps at each iteration, thereby greatly enhancing the convergence properties.

In this paper, for purposes of clarity, we focus on the one-dimensional implementation of the algorithm as a proof-of-concept, and primarily consider the effects of soft faults. The extension to 2D problems with both hard and soft faults will be covered in [35].

2.2. One-dimensional preconditioner formulation. In this section, we provide a detailed formulation for a one-dimensional system. Consider N equal size subdomains with an overlap size h between two neighbors. The equal-size requirement for both subdomains and overlaps is merely out of convenience and by no means is a requirement in the algorithm. Denoting the full domain $\Omega = (a, b)$, we can write

$$[a, b] = \cup_{i=1}^{N} [a_i, b_i] \tag{2.3}$$

where the i-th subdomain is $\Omega_i = (a_i, b_i)$ with $a_i = a + (i-1)(b-a)/N - h/2$, $b_i = a + i(b-a)/N + h/2$, except $a_1 = a$ and $b_N = b$. The boundaries for each subdomain consist of two points only $\partial\Omega_i = \{a_i, b_i\}$. For convenience, we order the y-values at all internal boundaries according to an increasing order of their corresponding x-locations, *i.e.*

$$\boldsymbol{y} = (y_2^L, y_1^R, y_3^L, y_2^R, \ldots, y_N^L, y_{N-1}^R), \tag{2.4}$$

where $y_i^L = y(a_i)$ and $y_i^R = y(b_i)$ denote the solution at the left and right boundaries of the i-th subdomain, correspondingly. Each internal subdomain, *i.e.* for $1 < i < N$, involves two boundary conditions, y_i^L and y_i^R, and contains two boundaries of other subdomains (namely, y_{i-1}^R and y_{i+1}^L), while the first and the last subdomain have one varying boundary condition and contain only one relevant boundary of another subdomain. Therefore, we need to construct $2N-2$ maps, each one of them from the boundary conditions on a subdomain to the internal boundaries of interest in that subdomain

$$\begin{cases} y_2^L = f_1^R(y_1^R) \\ \begin{cases} y_{i-1}^R = f_i^L(y_i^L, y_i^R) \\ y_{i+1}^L = f_i^R(y_i^L, y_i^R) \end{cases} & \text{for } 1 < i < N \\ y_{N-1}^R = f_N^L(y_N^L) \end{cases} \tag{2.5}$$

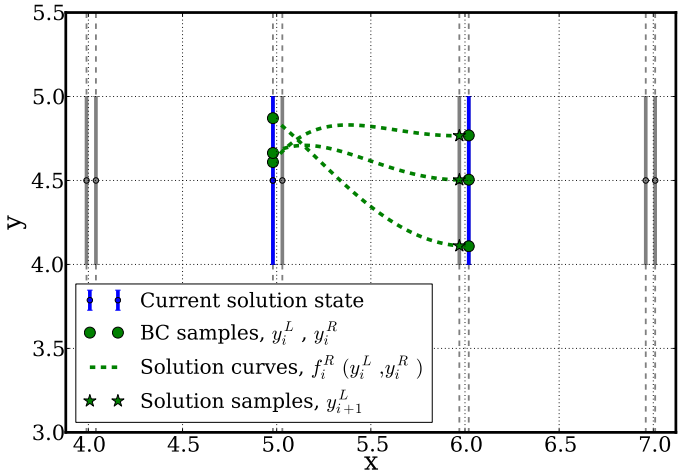


FIG. 3. *Demonstration of the surrogate boundary map construction, on the example of* $y_{i+1}^L = f_i^R(y_i^L, y_i^R)$.

In the system of equations (2.5), $f_i^{L(R)}$ denotes the map induced by the solution in the i-th subdomain, evaluated at the left (right) internal boundary of interest. Figure 3 demonstrates three sampled solutions and the output detected on the right-internal boundary of interest, within a single subdomain. With appropriate reordering, the system (2.5) can now simply be written as

$$\boldsymbol{y} = \mathcal{F}\boldsymbol{y}, \tag{2.6}$$

i.e. a fixed point problem, the solution $\tilde{\boldsymbol{y}}$ of which is the target we seek. Generally, the operator $\mathcal{F}$ is nonlinear and not known explicitly, making the solution of the fixed point problem problematic, since it would require PDE solves on each subdomain to evaluate $\mathcal{F}$ at each iteration of the fixed point solver. Therefore, we construct surrogate approximations $g_i^{L(R)} \approx f_i^{L(R)}$ leading to an approximate operator $\mathcal{G} \approx \mathcal{F}$. As shown in the Appendix, there is no approximation, *i.e.* $\mathcal{G} = \mathcal{F}$ for linear PDEs.

For nonlinear PDEs, we introduce iterative solution update strategy that renders the approximation increasingly more accurate as iterations progress. The solution of the fixed point problem $\boldsymbol{y} = \mathcal{G}\boldsymbol{y}$ will be considerably less expensive and intrusive, as it will only require evaluations of a priori found *surrogate* functions $g_i^{L(R)}$. Note that for simplicity we dropped the iteration counter T – we set the fixed point solution $\tilde{\boldsymbol{y}}$ to be the best-knowledge solution at the next iteration $\boldsymbol{y}^{(T+1)} = \tilde{\boldsymbol{y}}$, and shrink the range of sampling so that the surrogate map $\mathcal{G} = \mathcal{G}^{(T)}$ is updated to a more accurate one $\mathcal{G} = \mathcal{G}^{(T+1)}$ for the next iteration (see Section 2.4).

An important measure of the accuracy of the current solution $\boldsymbol{y}^{(T)}$ is the *residual* vector, defined as

$$\boldsymbol{z}^{(T)} = \mathcal{F}\boldsymbol{y}^{(T)} - \boldsymbol{y}^{(T)}, \tag{2.7}$$

which can be computed by extra subdomain solves using boundary conditions defined by the current solution $\boldsymbol{y}^{(T)}$, and subsequent subtraction the corresponding current solutions $\boldsymbol{y}^{(T)}$ from the resulting values at all boundaries. It follows from the definition that the residual (2.7) vanishes if the current solution $\boldsymbol{y}^{(T)}$ is the exact solution we seek.

We build surrogates that have polynomial form, as detailed further in Section 2.3. These surrogate maps are built over ranges centered at the current state of the solution $\boldsymbol{y}$. The vector of such ranges is denoted by $\boldsymbol{r}$ assuming the same ordering as in $\boldsymbol{y}$, *i.e.*

$$\boldsymbol{r} = (r_2^L, r_1^R, r_3^L, r_2^R, \ldots, r_N^L, r_{N-1}^R). \tag{2.8}$$

Note that the pairs $(\boldsymbol{y}, \boldsymbol{r})$ essentially encapsulate the state-of-knowledge of the solution. Indeed, $\boldsymbol{y}$ is the current solution state, while the surrogate range $\boldsymbol{r}$ is intended to be chosen so that the true solution is within the range $(\boldsymbol{y} - \boldsymbol{r}, \boldsymbol{y} + \boldsymbol{r})$ or not too far outside of it. In the latter case, the accuracy of the fixed point solver can be somewhat compromised due to extrapolation issues, but this effect is not crucial particularly for linear surrogates, and as the iteration counter marches on, the solution state still converges to the true solution.

For linear problems, the boundary maps are linear, and constructing a linear surrogate is sufficient, producing no approximation error. Moreover, even for non-linear problems, most often a linear surrogate will suffice, particularly because the ranges of inputs over which the surrogate approximation is built, *i.e.* components of $\boldsymbol{r}$, will be chosen to shrink – by an appropriate solution state updating mechanism detailed in Section 2.4 – as one gets closer to the true solution, thereby rendering linear approximations increasingly more accurate. For these reasons, unless specifically mentioned otherwise, we will focus on linear surrogate maps. The fixed point system (2.5) will thus take the form

$$\begin{cases} y_2^L = \alpha_1^R + \gamma_1^R y_1^R \\ \begin{cases} y_{i-1}^R = \alpha_i^L + \beta_i^L y_i^L + \gamma_i^L y_i^R \\ y_{i+1}^L = \alpha_i^R + \beta_i^R y_i^L + \gamma_i^R y_i^R \end{cases} & \text{for } 1 < i < N \\ y_{N-1}^R = \alpha_N^L + \beta_N^L y_N^L. \end{cases} \tag{2.9}$$

The coefficients $\alpha_i^{L(R)}, \beta_i^{L(R)}, \gamma_i^{L(R)}$ are found a priori by fitting the linear map (2.9) to a set of training data obtained by solving the PDE on each subdomain for a sampled

set of boundary conditions. Casting $\tilde{\boldsymbol{y}}$ as a column vector, the linear map $\mathcal{G}$ is written in a matrix-vector form

$$\tilde{\boldsymbol{y}} = \boldsymbol{G}\tilde{\boldsymbol{y}} + \boldsymbol{h}, \quad \text{where} \tag{2.10}$$

$$\boldsymbol{G} = \begin{pmatrix} 0 & \gamma_1^R & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \beta_2^L & 0 & 0 & \gamma_2^L & 0 & 0 & 0 & \dots & 0 \\ \beta_2^R & 0 & 0 & \gamma_2^R & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \beta_3^L & 0 & 0 & \gamma_3^L & 0 & \dots & 0 \\ 0 & 0 & \beta_3^R & 0 & 0 & \gamma_3^R & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \dots & \dots & \beta_{N-1}^L & 0 & 0 & \gamma_{N-1}^L \\ 0 & 0 & \dots & \dots & \dots & \beta_{N-1}^R & 0 & 0 & \gamma_{N-1}^R \\ 0 & 0 & \dots & \dots & \dots & 0 & 0 & \beta_N^L & 0 \end{pmatrix}, \qquad \boldsymbol{h} = \begin{pmatrix} \alpha_1^R \\ \alpha_2^L \\ \alpha_2^R \\ \alpha_3^L \\ \alpha_3^R \\ \vdots \\ \alpha_{N-1}^L \\ \alpha_{N-1}^R \\ \alpha_N^L \end{pmatrix}. \tag{2.11}$$

The linear system can therefore be solved by simple matrix algebra. Denoting by I the identity matrix of size $2N-2$, one arrives at the solution

$$\tilde{\boldsymbol{y}} = (\boldsymbol{I} - \boldsymbol{G})^{-1}\boldsymbol{h} \tag{2.12}$$

Garbey *et. al.* [23] presented the Aitken acceleration method for Schwarz iterations of linear equations that makes use of Eq. (2.12). They also noted that the matrix $\boldsymbol{I} - \boldsymbol{G}$ is the same appearing in the classical additive Schwarz algorithm for linear problems. Therefore, if the naïve, deterministic iteration scheme converges for a given problem, then the operator $\mathcal{F}$ is a *contraction* and $\boldsymbol{y} = \mathcal{F}\boldsymbol{y}$ has a solution. For linear problems, $\mathcal{G} = \mathcal{F}$, therefore the matrix $\boldsymbol{I} - \boldsymbol{G}$ is invertible and our method converges *if* the classical additive Schwarz method converges for the same PDE. The convergence properties are not as straightforward for nonlinear problems, or in presence of more than one fault per set of subdomain sample solutions, since the linear surrogate map $\mathcal{G}$ is generally not exact in those cases. However, increasingly more accurate linear surrogates enhance the convergence properties as will be seen later in the paper. Note that the non-convergence can be detected by looking at the difference between subsequent iterations, followed by a redundant repeat of the same iteration which will clarify whether either a large number of bit-flips or the PDE itself is the reason. In [23], the entries of $\boldsymbol{G}$ are learnt by two solves per subdomain with unit boundary conditions. The latter is only possible because of the linearity of the boundary-to-boundary maps. Furthermore, the authors in [23] estimate $\boldsymbol{h}$ by simply marching one iteration forward with a single solution per subdomain $\boldsymbol{h} = \boldsymbol{y}^{(1)} - \boldsymbol{G}\boldsymbol{y}^{(0)}$. The algorithm described in the current paper is able to handle non-linear problems via surrogate maps, and also offers fault resilience through the proper choice of the regression approach to obtain the surrogate maps (as discussed in Section 2.3). The

workflow in the 1D setting is described in Algorithm 1.

Algorithm 1: A typical workflow of the method

Input:

- N overlapping subdomains $[a_i, b_i]$ for $i = 1, \ldots, N$, where $a_1 = a, b_N = b, b_j - a_{j+1} = h$ for $j = 1, \ldots, N-1$.
- Initial, uninformative knowledge of solution a vector of length $2N-2$, $\boldsymbol{y} = (y_2^L, y_1^R, y_3^L, y_2^R, \ldots, y_N^L, y_{N-1}^R)$, and the uncertainty range $\boldsymbol{r}$ around that solution.
- Parameters M (number of samples per subdomain for surrogate map construction), q (polynomial order of the surrogate map)

while *Stopping criterion not met* **do**

  foreach *Subdomain* **do**

    Boundary sampling: generate M samples distributed uniformly over the current ranges $\boldsymbol{r}$ for both ends of the subdomain

    Subdomain solver: solve the PDE using the sampled boundary conditions to obtain M values at the boundaries of other subdomains, internal to the current subdomain.

    Surrogate map construction: using the M values found above, employ Bayesian inference to obtain surrogate relationship for the boundary-to-boundary map.

  end

  Fixed point solver: solve the system of all surrogate maps to obtain a solution at all subdomain boundaries

  foreach *Subdomain* **do**

    Solution state update: Update the representation of the solution state at the boundaries of this subdomain, *i.e.* generate new values for $\boldsymbol{y}$ and $\boldsymbol{r}$.

  end

end

Result: Final knowledge about the solution $\boldsymbol{y}$.

The next two subsections detail key parts of the algorithm, *i.e.* the construction of surrogate maps and the adaptive strategies for choosing the uncertainty ranges and solution state update mechanisms, including appropriate stopping criteria.

2.3. Surrogate boundary map construction. A key feature that we seek in the surrogate map construction is resilience with respect to faulty behavior, *e.g.* perturbed or missing values. To enable such resiliency, Bayesian techniques will be invoked. The Bayesian approach is well-aligned with our paradigm. Indeed, it operates with any given information content available, incorporating uncertainties associated with missing or faulty data in a natural way [3, 11, 40].

To this end, polynomial regression is employed for the surrogate map construction in the 1D problem. Within a single subdomain, the boundary-to-boundary maps $f_{i\to j}$ have a form $y_{in} = f(y^L, y^R)$, where y^L, y^R are the boundary conditions on the left and right boundaries, respectively, while y_{in} is the solution at an inner point of interest, as described in Eq. (2.5)[1]. The goal is to approximate the function $f(y^L, y^R)$ over ranges

[1] Note that the first and the last subdomains have one input. However, for clarity of presentation, and without loss of generality, we describe the generic two-input case. In principle, even for those subdomains, one can think of the global boundary values as the second input.

$y^L \in [y^L_{(i)} - r^L_{(i)}, y^L_{(i)} + r^L_{(i)}]$ and $y^R \in [y^R_{(i)} - r^R_{(i)}, y^R_{(i)} + r^R_{(i)}]$ using the state-of-knowledge $(\boldsymbol{y}_{(i)}, \boldsymbol{r}_{(i)})$ at the current, i-th iteration. We postulate a polynomial representation

$$y_{in} \approx \sum_k c_k \Psi_k \left(\frac{y^L - y^L_{(i)}}{r^L_{(i)}}, \frac{y^R - y^R_{(i)}}{r^R_{(i)}} \right) \tag{2.13}$$

with respect to scaled values of the boundary conditions $(y^{L(R)} - y^{L(R)}_{(i)})/r^{L(R)}_{(i)} \in [-1, 1]$, in terms of an expansion of orthogonal Legendre polynomials up to a given bivariate order q. Note that while orthogonality is not employed directly, it helps to keep the coefficients stable as one increases the order. Besides, most of the work shown here employs linear surrogates, *i.e.* the basis set $\Psi_k(z_1, z_2)_{\{k=0,1,2\}} = \{1, z_1, z_2\}$. To 'learn' the relationship $f(\cdot)$, we sample the inputs M times and extract the corresponding output y_{in} for each sample by a single subdomain solve per set of inputs. Given these training samples, we invoke Bayes rule to infer polynomial coefficients c_k,

$$p(\boldsymbol{c}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{c}) p(\boldsymbol{c}) \tag{2.14}$$

where the data set $\mathcal{D}$ refers to the training samples, and the likelihood function $p(\mathcal{D}|\boldsymbol{c})$ essentially measures the goodness-of-fit of the polynomial representation to the data. Thus, the prior probability distribution $p(\boldsymbol{c})$ of the polynomial coefficient vector $\boldsymbol{c}$ is updated, yielding to a posterior distribution $p(\boldsymbol{c}|\mathcal{D})$,

The likelihood function usually pertains to the discrepancy between the simplified polynomial model and the actual solver results and is the key component of the Bayesian formulation. In the following, we will describe the classical, Gaussian likelihood construction, and we will argue that in presence of faults, it is far more resilient to utilize a Laplace likelihood form instead of a Gaussian one.

2.3.1. Gaussian likelihood. For notational convenience, the output of the function $f(\cdot)$ will be denoted by u, and the inputs will be denoted by $\boldsymbol{v}$. In practice, $\boldsymbol{v}$ is the vector of boundary conditions corresponding to a given subdomain, and u is the PDE solution evaluated at a given x-location of interest, *e.g.* at locations that serve as boundaries of other subdomains. The goal is, given a data set of *training* samples $\mathcal{D} = \{(\boldsymbol{v}_i, u_i)\}_{i=1}^M$, find the best fitting polynomial coefficients $\boldsymbol{c} = (c_0, \ldots, c_{K-1})$, so that

$$u_i = f(\boldsymbol{v}_i) \approx \sum_{k=0}^{K-1} c_k \Psi_k(\boldsymbol{v}_i) \equiv g_{\boldsymbol{c}}(\boldsymbol{v}_i) \tag{2.15}$$

Consider the classical, Gaussian log-likelihood form assuming an i.i.d. Gaussian noise model with a constant variance σ^2 for the *discrepancy* $\epsilon_i = u_i - g_{\boldsymbol{c}}(\boldsymbol{v}_i)$:

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{c}, \sigma^2) = \log p(\mathcal{D}|\boldsymbol{c}, \sigma^2) = -\frac{M}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{M} (u_i - g_{\boldsymbol{c}}(\boldsymbol{v}_i))^2. \tag{2.16}$$

Let us define the projection matrix $\boldsymbol{P}$ of size $M \times K$ by $\boldsymbol{P}_{ik} = \Psi_k(\boldsymbol{v}_i)$ for $i = 1, \ldots, M$ and $k = 0, \ldots, K-1$. Also, denote the vector of training outputs by $\boldsymbol{u} = (u_1, \ldots, u_M)$. Now the log-likelihood can be written in a more compact form

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{c}, \sigma^2) = -\frac{M}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c})^T (\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c}). \tag{2.17}$$

We assume independent uniform priors for components of $\boldsymbol{c}$ that are sufficiently wide, while uninformative, Jeffrey's prior [27] is used on σ, *i.e.* $p(\sigma) \sim \frac{1}{\sigma}$, which we note is equivalent to the commonly used $p(\sigma^2) \sim \frac{1}{\sigma^2}$ or $p(\log \sigma) \sim 1$. The latter is useful, since for practical purposes, and for enforcing positivity, one often works with $\log \sigma$, assuming a uniform prior for it.

According to Bayes rule, the posterior is proportional to the product of likelihood and prior,

$$p(\boldsymbol{c}, \sigma^2|\mathcal{D}) \propto \frac{1}{\sigma^M} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c})^T(\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c})\right) \frac{1}{\sigma^2}, \tag{2.18}$$

and, after standard mathematical rearranging, one can derive

$$\begin{aligned} p(\boldsymbol{c}|\sigma^2, \mathcal{D}) &\propto \frac{1}{\sigma^K} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{c} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{c} - \boldsymbol{\mu})\right), \\ \boldsymbol{c}|\sigma^2, \mathcal{D} &\sim \mathcal{MVN}(\underbrace{(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{u}}_{\boldsymbol{\mu}}, \sigma^2 \underbrace{(\boldsymbol{P}^T\boldsymbol{P})^{-1}}_{\boldsymbol{\Sigma}}), \end{aligned} \tag{2.19}$$

which, after marginalization, results in an Inverse Gamma distribution for σ^2 and a Multivariate Student-t distribution for $\boldsymbol{c}$ [17, 29]:

$$\sigma^2|\mathcal{D} \sim \mathcal{IG}\left(\underbrace{\frac{M-K}{2}}_{\alpha}, \underbrace{\frac{\boldsymbol{u}^T(\boldsymbol{I} - \boldsymbol{P}(\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T)\boldsymbol{u}}{2}}_{\beta}\right), \tag{2.20}$$

$$\boldsymbol{c}|\mathcal{D} \sim \mathcal{MST}(\boldsymbol{\mu}, \frac{\beta}{\alpha}\boldsymbol{\Sigma}, 2\alpha). \tag{2.21}$$

Note that the Multivariate Student's t-distribution has covariance $\frac{\alpha}{\alpha-1}\boldsymbol{\Sigma}$, and it behaves very similarly to a Gaussian distribution for large α. The mean estimate, as well as the maximum a posteriori (MAP) estimate $\boldsymbol{\mu}$ coincides with the well-known least-squares fit formula. The likelihood (2.16) assumes an i.i.d. Gaussian error model for the discrepancy between surrogate evaluations $g_{\boldsymbol{c}}(\boldsymbol{v}_i)$ and true solutions u_i. The magnitude of inferred σ is associated with such model discrepancy, *i.e.* the error produced due to polynomial approximation of the boundary map, and provides insight about the committed error size due to polynomial approximation. The marginal posterior mean for σ^2 is $m_{\sigma^2|\mathcal{D}} = \beta/(\alpha - 1)$, while the mode of the marginal distribution is $\underset{\sigma^2}{\operatorname{argmax}}\, p(\sigma^2|\mathcal{D}) = \beta/(\alpha+1)$. Furthermore, the MAP estimate of the joint posterior distribution of $(\boldsymbol{c}, \sigma^2)$ is $\underset{\sigma^2}{\operatorname{argmax}}\, p(\boldsymbol{c}, \sigma^2|\mathcal{D}) = \beta/(\alpha + 3/2)$. Generally, when the number of samples exceeds the number of unknown coefficients by an order of magnitude, both β and α are large leading to approximate relation $\sigma^2 \approx \beta/\alpha$.

Although the Gaussian likelihood offers simple analytical expressions for the posterior distributions as described above, it is not resilient with respect to faulty training samples, since the associated i.i.d. Gaussian error model does not capture the nature of the errors produced by faulty samples. The next section will detail the deficiencies of the Gaussian likelihood when dealing with potentially corrupt training samples.

2.3.2. Faulty data. In the situation where no accidental faults/bit-flips are present, the surrogate map can be written in a matrix form

$$\boldsymbol{u} = \boldsymbol{P}\boldsymbol{c}_0 + \boldsymbol{\epsilon}_1, \tag{2.22}$$

where the vector $\boldsymbol{c}_0$ is the best fitting parameter vector. The vector of residuals $\boldsymbol{\epsilon}_1$ here represents the misfit due to the non-polynomial character of the map $f(\cdot)$, *i.e.* it represents the model error. This error is non-zero only for non-linear problems. In Section 2.4, we discuss how we achieve negligible model discrepancy errors by shrinking the range over which the surrogate map is constructed, therefore making the forward function f more and more linear. In the following computations, this error term will be tracked, but we should note that it vanishes for linear problems and can be made arbitrarily small for non-linear problems.

Now let us assume the data is 'faulty', *i.e.* $\boldsymbol{u}$ is corrupted by an error vector $\boldsymbol{\delta}$:

$$\boldsymbol{u} + \boldsymbol{\delta} = \boldsymbol{P}\boldsymbol{c}_f + \boldsymbol{\epsilon}_2 \tag{2.23}$$

The representation induces a different model error $\boldsymbol{\epsilon}_2$. The error made in the fitted parameters $\boldsymbol{c}_e = \boldsymbol{c}_f - \boldsymbol{c}_0$ is then equal to

$$\boldsymbol{c}_e = (\boldsymbol{P}^T\boldsymbol{P})^{-1}\boldsymbol{P}^T\boldsymbol{\delta}. \tag{2.24}$$

and it obeys

$$\boldsymbol{\delta} = \boldsymbol{P}\boldsymbol{c}_e + \underbrace{(\boldsymbol{\epsilon}_2 - \boldsymbol{\epsilon}_1)}_{\equiv\boldsymbol{\epsilon}}. \tag{2.25}$$

Note that typically $\boldsymbol{\delta}$ is a vector of size M, typically 10 to 100, with at most one or perhaps two non-zero elements, as this is the error made in data due to sporadic, relatively rare bit-flips or faults. Due to basis orthonormality, we have $\boldsymbol{P}^T\boldsymbol{P} \approx M\boldsymbol{I}$ Therefore, we can estimate

$$||\boldsymbol{c}_e||_2 = \mathcal{O}\left(\frac{\delta}{M}\right) \tag{2.26}$$

where the scalar $\delta = ||\boldsymbol{\delta}||_2$ is the fault magnitude. Moreover, using (2.20) and $\sigma^2_{\text{MAP}} = \beta/(\alpha + 3/2)$, one can estimate the best value of σ^2 to be

$$\sigma^2 \sim \mathcal{O}\left(\frac{\delta^2}{M}\right) \tag{2.27}$$

For completeness, note also that the posterior covariance width is $\mathcal{O}(\sigma^2/M) = \mathcal{O}\left(\delta^2/M^2\right)$, *i.e.* the 'width' of the multivariate posterior is $\mathcal{O}\left(\delta/M\right)$ and is of the order of the induced coefficient error $||\boldsymbol{c}_e||_2$ according to (2.26). This means that a single perturbation of size δ induces an error in the coefficients of magnitude δ/M, trying to balance the single large misfit with many smaller ones. This is typical in least-squares based likelihoods, but is not acceptable if one targets fault resilience. For that reason, we will employ Laplace likelihoods for the surrogate map constructions. As discussed below, the Laplace likelihood avoids such 'balancing' issues and is much more robust with respect to the presence of a few number of outliers.

2.3.3. Laplace likelihood. One of the key algorithmic advances in this work is the use of Laplace likelihoods for the surrogate map construction. That is, we apply Bayesian inference of the parameters of interest employing i.i.d. Laplace distributions for the discrepancies $\epsilon_i = u_i - g_{\boldsymbol{c}}(\boldsymbol{v}_i)$, to get

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{c}, \lambda) = \log p(\mathcal{D}|\boldsymbol{c}, \lambda) = -M\log(2\lambda) - \frac{1}{\lambda}\sum_{i=1}^{M}|u_i - g_{\boldsymbol{c}}(\boldsymbol{v}_i)|, \tag{2.28}$$

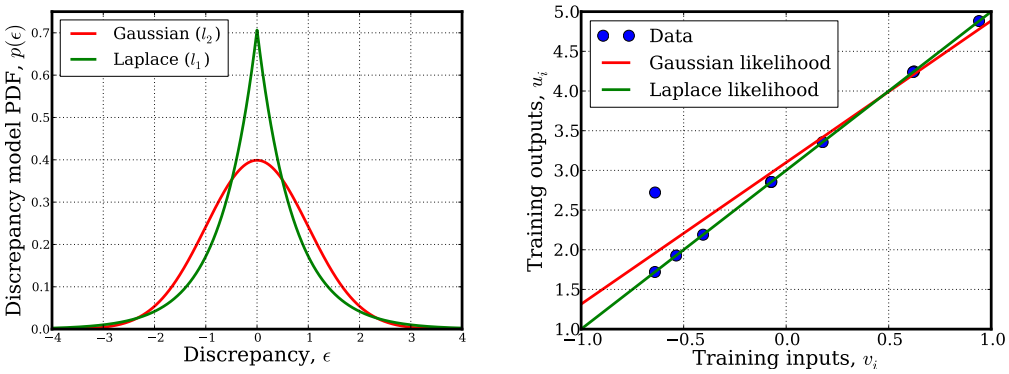


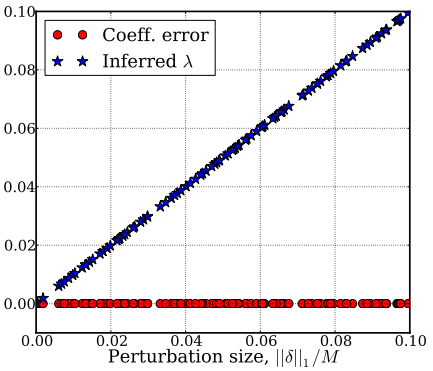
FIG. 4. *(left) Illustration of Gaussian and Laplace distributions of the same variance, (right) linear regressions with a single perturbed point, using both Gaussian and Laplace likelihood.*

which is closely related to the ℓ_1-norm of the discrepancy vector $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_M)$, as opposed to the ℓ_2-norm associated with the Gaussian likelihood.

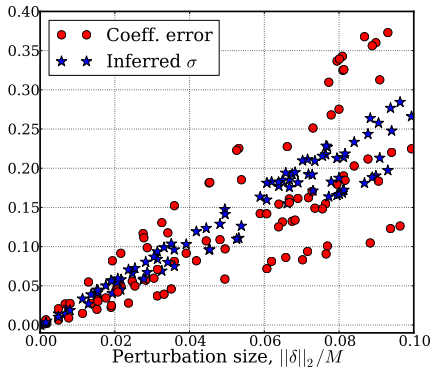
This likelihood reflects the fact that most of the errors are very small with only a few exceptions, which is consistent with the outputs of a solver that is most of the time exact, except for the occasional fault. The left plot of Figure 4 illustrates both Gaussian and Laplace probability distributions of unit variance, demonstrating the sharp peak near the 0-error value and longer 'tail' for the Laplace distribution. The right plot of Figure 4 demonstrates the fault-resilient property of the Laplace likelihood – the regression proceeds as if there is no perturbed point, while the Gaussian likelihood induces a linear function that tries to balance all the residual errors between $M-1$ 'good' and a single 'bad' point.

Unlike the Gaussian likelihood demonstrated in Section 2.3.1, the Laplace likelihood does not allow for an analytical solution and subsequent estimates similar to the ones derived in Section 2.3.2. Hence we use simple numerical studies to investigate the effect of errors in data. Figure 5 demonstrates the result of such studies. We used a simple linear, bivariate function to generate $M = 10$ samples, one (top row) or two (bottom row) of which are randomly selected and perturbed by a random amount, δ or (δ_1, δ_2), respectively. Then the Bayesian regression algorithm is employed with either the Laplace (left column) or the Gaussian (right column) likelihood. This procedure is done for an ensemble of 100 random perturbations of the same case, and the norm of the error made in the polynomial coefficients, as well as the inferred λ or σ are reported. Clearly, the Laplace likelihood almost always produces the true linear function without any error in coefficients, while the inferred value for λ is proportional to the ℓ_1-norm of the error. In fact, $\lambda = \delta/M$ for the one-error case and $\lambda = (|\delta_1| + |\delta_2|)/M$ for the two-error case[2]. Only for one of the 100 cases, in the two-error scenario, did the regression with the Laplace likelihood fail to recover the true coefficients. The Gaussian likelihood, shown in the right column, clearly produces errors in the coefficients that are comparable with the perturbation size. In fact, the linear trend is consistent with the estimates derived in Section 2.3.2, *i.e.* the coefficient error size is $||\boldsymbol{c}_e||_2 = \mathcal{O}(\delta/M)$ and the inferred Gaussian width is

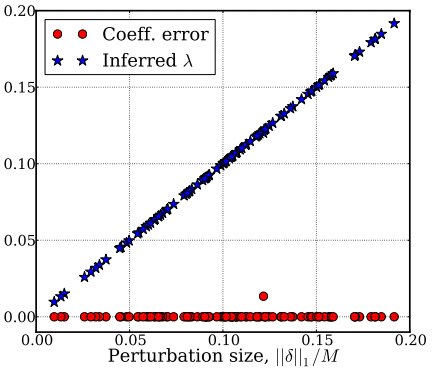
[2] In fact it can be proven, by taking the derivative of the likelihood with respect to λ, that when the true coefficients are recovered, the best value for λ is indeed $\sum_f |\delta_f|/M$.



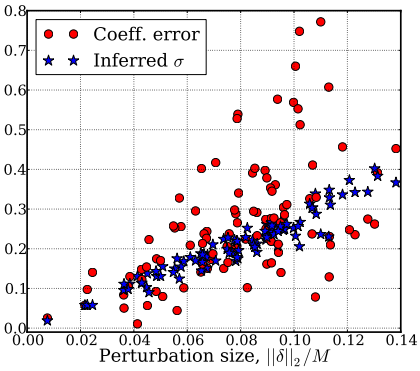
(a) Laplace likelihood, 1/10 faulty sample



(b) Gaussian likelihood, 1/10 faulty sample



(c) Laplace likelihood, 2/10 faulty samples



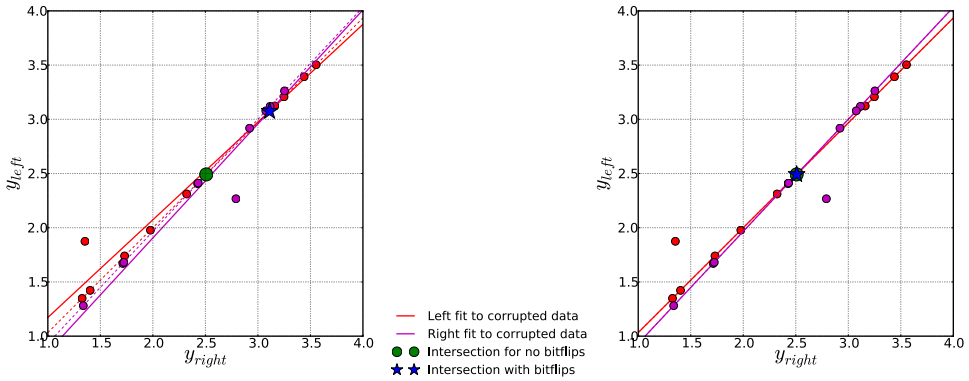
(d) Gaussian likelihood, 2/10 faulty samples

FIG. 5. *Demonstration of* 100 *regression tests with Laplace (left column) and Gaussian (right column) likelihoods, in the presence of random errors in one (top row) or two (bottom row) randomly selected data values, out of* $M = 10$ *samples. Plotted are inferred errors induced in the resulting coefficients and* λ *(or* σ*) versus the size of random perturbations.*

$\sigma = \mathcal{O}\left(\delta/\sqrt{M}\right)$.

Within the formulation of our 1D algorithm, the surrogate map construction proceeds twice for each internal subdomain and once for the boundary subdomains. Figure 6 demonstrates the advantage of the Laplace likelihood versus the Gaussian likelihood, for a linear problem, in the simplest, two-subdomain case, where each subdomain involves a surrogate construction for the map from the y-value at its boundary to the y-value at the internal boundary of the other subdomain. The fixed point, *i.e.* the intersection, of the maps $y_{left} \to y_{right}$ and $y_{right} \to y_{left}$ is the solution we seek. Clearly, with perturbed points, the surrogate map stays exact in the Laplace likelihood case, producing no error in the final fixed-point answer.

We note that while the general surrogate construction has been introduced within a Bayesian framework, here and in the subsequent tests we are mainly interested in the MAP value, with an understanding that the posterior width becomes narrow enough for a reasonably large number of sampled points after a few iterations. Both



(a) Gaussian likelihood (b) Laplace likelihood

FIG. 6. *Illustration of the surrogate map intersection for a 2-subdomain test case. The red points and lines correspond to $y_{left} = f_1(y_{right})$, while purple ones correspond to $y_{right} = f_2(y_{left})$. Dashed lines correspond to true maps with no perturbed points. Intersection of dashed lines is shown by green circles, while the intersection of solid lines corresponds to blue stars.*

analytical calculations for the Gaussian case (see Section 2.3.2 and the note after Eq. (2.27)) and numerical studies for the Laplace-likelihood case (results not shown) indicate that the posterior width is inversely proportional to the sample size M and proportional to the extent of nonlinearity of the boundary-to-boundary maps. The latter vanishes for the linear case and reduces proportionally to the surrogate range for the non-linear case. The MAP value is found by employing optimization via Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm [6]. In practice, the optimization problem is challenged by the Laplace likelihoods due to discontinuous derivatives. To alleviate this issue, we use a hyperbolic-tangent approximation, in which $\tanh(z/\epsilon)$ replaces the derivative of $|z|$ instead of the sign function $\mathrm{sign}(z)$. Correspondingly, the antiderivative of $\tanh(z/\epsilon)$, $|z| + \epsilon \ln\left(1 + e^{-2|z|/\epsilon}\right)$ is employed to replace $|z|$. The 'smoothing' constant ϵ is set to 10^{-14}.

2.4. Surrogate range choice and solution state updating for non-linear problems. For linear differential equations, the boundary-to-boundary maps are linear, therefore a linear surrogate will be exact, and the fixed point problem will recover the true solution in a single iteration. Note that in an extremely unlikely case of two faults per single subdomain sampling set there is a considerable chance of an inaccurate linear surrogate that may a) render the final fixed point solution inexistent, *i.e.* the matrix $\boldsymbol{I} - \boldsymbol{G}$ not invertible, or b) fall short of the exact final solution. In the former case, however unlikely, one can implement a detection and subsequent repeat of the same step, while in the latter case another iteration will find the true solution, assuming negligible likelihood of multiple faults for two iterations in a row. For non-linear problems, linear surrogate approximation will impact the result of the fixed point solver, thus leading to an approximate answer within a single iteration/update. The accuracy of the surrogate approximation is directly affected by the size of the range over which the surrogate is constructed: the smaller the range, the more accurate the linear map is. However, the range needs to be large enough to avoid errors due to extrapolation. Ideally, the surrogate range $r_{(i)}$ of the given subdomain boundary at the i-th iteration should be chosen adaptively according to some measure of

distance from the true solution. Below we detail three different approaches,

- *MAP*: In this case, the range is driven by the MAP value λ^{MAP} from the previous iteration. Indeed, in the surrogate construction, we assumed a Laplace noise model and the MAP value of the noise magnitude parameter λ is a good indicator of the size of error in the data. Inspired by the linear case best estimate $\lambda^{\text{MAP}} = \sum_f |\delta_f|/M$, we set $r_{(i)} = fM\lambda^{\text{MAP}}_{(i)}$, with a 'safety' factor f. We found $f = 5$ to work well generally in all tests. The MAP value λ^{MAP} also can be interpreted as a lower bound on the posterior predictive standard deviation. In fact, if one has sufficiently many samples, the posterior itself is narrow enough making λ the sole contributor to the posterior predictive variance. However, this approach requires surrogate construction with the inference of both the surrogate coefficients and λ, which is generally expensive computationally as it requires many iterations to converge to the best values of both surrogate coefficients and λ.
- *Surrogate error*: One can instead employ a fixed λ for the surrogate construction in order to gain computational efficiency. The fixed value of λ is not relevant if one only is interested in the best, MAP values of the surrogate coefficients. In this case, the ℓ_1 error between the surrogate and the true map evaluated at M sample points is a good indicator of the overall committed error, and can be used to inform the range choice $r_{(i)} = f||\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c}_{\text{MAP}}||_1$, again within a 'safety' factor f. Here, too, a value of $f = 5$ was found to work well in general.
- *Solution difference*: While constructing the surrogate with fixed λ, the range may also be chosen according to the difference in the solution of the two current and previous iterations: $r_{(i)} = f|y_{(i)} - y_{(i-1)}|$. This approach, while more empirical than the previous two, is more resilient to faults since the surrogate range now is not getting corrupted by errors that are due to bit-flips. A somewhat heuristic justification is that if the convergence is expected to be exponential with respect to the number of iterations, then the committed error should be proportional to the difference of the errors in two previous iterations. If these latter errors have the same sign, then their difference is exactly the difference in solutions $|y_{(i)} - y_{(i-1)}|$. While recognizing that f will generally be problem dependent, we have found $f = 0.1$ to work the best in a variety of test cases. This approach of surrogate range choice has shown at least as good a convergence as the previous two, and better computational efficiency. In all of the tests further in this work, we therefore employ this rule for selecting the surrogate range.

On a conceptual level, our approach starts out with a (mean,range)=($\boldsymbol{y}$,$\boldsymbol{r}$) representation of the state-of-knowledge about the problem solution, and then updates this representation, the *solution state*, with information gathered from computations on the subdomains. At the i-th iteration, the solution state consists of the solution value itself $\boldsymbol{y}_{(i)}$ and the range $\boldsymbol{r}_{(i)}$ over which we construct the surrogate map, for each subdomain boundary. The initial representation $(\boldsymbol{y}_{(0)}, \boldsymbol{r}_{(0)})$ can be obtained in a variety of ways, such as expert opinion (*e.g.* the solution of an elliptic problem without forcing terms should be between the boundary condition values), or informed by a set of coarse grained solutions. Again, we emphasize that the surrogate range choice becomes relevant only for non-linear problems.

The next section details experiments with a few test problems, both linear and non-linear, to help demonstrate the algorithm developed so far in this paper.

3. Numerical tests.

3.1. Fault model. In this work we adopt a synthetic algorithm for fault generation. To mimic the occurrence of soft faults during the subdomain solves, we implemented a binary bit-flip simulator in our software framework while still working with decimal representations. This algorithm employs random bit-flips of binary representations of solution values selected at random, with prescribed probability. Specifically, the entries that are selected are then converted to a 64-bit binary format according to the IEEE 754 Standard [1]. Once in binary format, one of the 64 bits is picked at random and its state is flipped between 0 and 1 with a given probability p. The binary representations are then converted back to base 10 and the simulation is allowed to proceed forward. Depending on the bit-flip location, in the mantissa, exponent, or the sign bit, the faulty value may sometimes be only slightly different from the original value or it can be modified by orders of magnitude or have an opposite sign.

In the tests below, bit-flips are applied to the results that return from the subdomain solves. As such, they can be interpreted as a crude model for all faults that might happen during a subdomain solve. Another interpretation is that the subdomain solves rely on a different resilience approach to mitigate the effects of faults, and the fault model applied here then represents the faults that potentially occur during transmission of data from the subdomain solver back to the entity that builds the surrogate maps. For purposes of illustrating the algorithmic approach, the actual interpretation of the fault model is not crucial. Rather, it should be seen as a way to perturb the data. Given the wide range of values that can result from single bit-flips in a given number, the current fault model is quite powerful for testing the algorithms. In this work, we mainly tested bit-flip probabilities ranging from 0.1% to 1%, which are considered quite high taking into account expected failure rates reported in the literature [37].

Further note that in the implementation here, faults are assumed to only happen during the subdomain solves. As these solves are likely to be the most time-consuming parts of a production simulation on a large-scale platform, this assumption is reasonable, albeit not completely robust. For complete robustness, the current algorithm would need to be implemented in a framework with selective reliability, *e.g.* some nodes can be considered safe while others are prone to errors [5]. The use of more refined fault models will be illustrated in [35].

3.2. Linear problem with analytical solution on subdomains. As a test case with an analytically available solution, consider the linear problem

$$y'' + 2y' + y = x\cos x \tag{3.1}$$

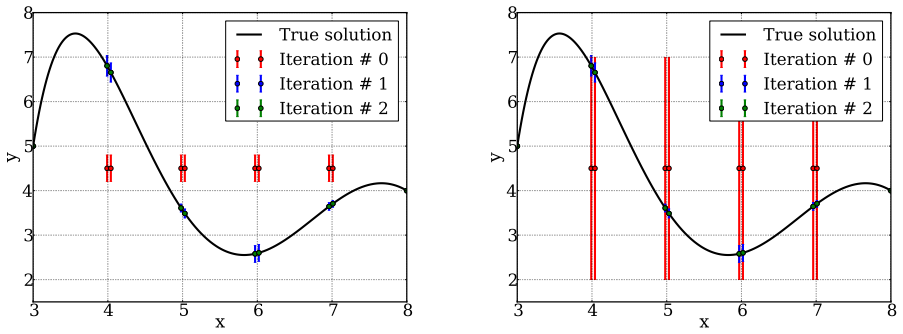
with boundary conditions $y(a) = y_a$ and $y(b) = y_b$. This differential equation has a solution

$$y(x) = Ce^{-x} + Dxe^{-x} + \frac{x}{2}\sin x - \frac{1}{2}(\sin x - \cos x)\,. \tag{3.2}$$

We set $a = 3$, $b = 8$, $y_a = 5$, $y_b = 4$, which uniquely define the values for constants C and D. This solution is depicted in Figure 7 with a solid black line.

In this first set of tests, for the purposes of illustrating the algorithm, we make use of the analytical solution to determine y on each of the subdomains. This means that the solution samples used to construct the boundary map surrogates are exact. Since

for linear problems, the boundary maps are linear, the linear surrogates constructed are also exact. Therefore, no matter what the initial non-zero surrogate range is, the solution state is expected to converge to the true answer in a single iteration, with a potential exception in case of bad conditioning of the regression problem. Figure 7 demonstrates the convergence for two cases where the initial solution ranges have been set to $r_{(0)} = 0.3$ and $r_{(0)} = 2.5$ for the internal boundaries. Clearly, after a single iteration, the mean solution state coincides with the true solution, while the range takes an extra iteration to reach nearly vanishing values by construction. The ranges at the global endpoints are chosen to be equal to zero by definition. The overlap size is set to $h = 0.05$, the number of subdomains is $N = 10$, while the number of samples per subdomain boundary is set to $M = 15$.



(a) $r_{(0)} = 0.3$ (b) $r_{(0)} = 2.5$

FIG. 7. *Convergence for the linear problem (3.2), in two cases of initial range choice for the internal boundaries. The initial solution states and ranges are illustrated with red dots and errorbars. The precomputed true solution is shown with black solid curve.*

To test the resilience with respect to system faults, we introduce the bit-flip scenario detailed in the beginning of Section 2.3.2. Figure 8 illustrates the convergence performance with or without bit-flips for two options of the surrogate map likelihood, Gaussian and Laplace, detailed in Section 2.3. Clearly, the presence of bit-flips deteriorates the convergence if the surrogates are built with Gaussian likelihoods, while the Laplace likelihood leads to a fault-resilient formulation. Note that even though the true solution was reached in one iteration, the algorithm was run for 10 iterations to better illustrate the resilience of the algorithm under the continual occurrence of bit-flips. In fact, the expected number of bit flips for N subdomains, M samples, T iterations and bit-flip probability p is $N_f = (2N-2)MpT$, where the factor $2N-2$ corresponds to the fact that there are 2 surrogates per each internal subdomain, and 1 surrogate for the first and last subdomains. For the case illustrated in Figure 8, the expected number of faults for each convergence curve is $N_f = 3.6$.

3.3. Non-linear problem with analytical solution on subdomains. Next, we study a non-linear 1D problem with an analytically available solution to test our algorithm. Consider the non-linear differential equation

$$(y^3)'' = \beta \sin(\alpha x), \tag{3.3}$$

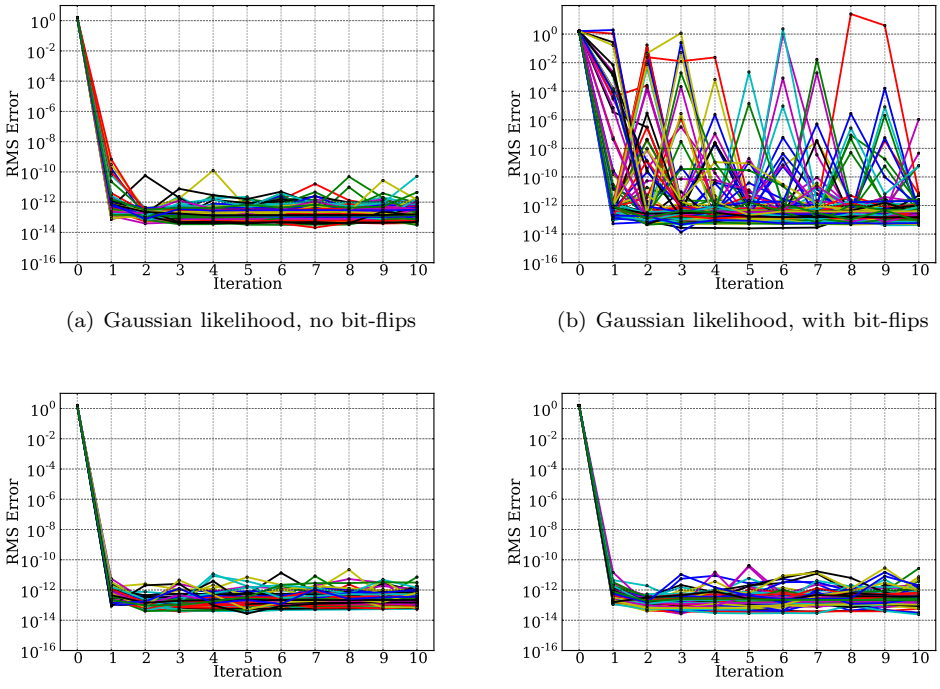
for some constant parameters α, β, on the domain $\Omega = (a, b)$. It clearly has the general form $\mathcal{L}y(x) = h(x)$ with a non-linear operator $\mathcal{L}$. The solution of (3.3) has the form

$$y(x) = \left(-\frac{\beta}{\alpha^2} \sin(\alpha x) + Cx + D \right)^{1/3}, \tag{3.4}$$

for appropriate values of C and D found by matching the boundary conditions $y(a) = y_a$ and $y(b) = y_b$. The default values for the tests illustrated in this work are:

$$\alpha = 5, \beta = 30 \qquad a = 1, b = 4 \qquad y_a = 2, y_b = 3. \tag{3.5}$$

The global solution for the selected boundary conditions is depicted in Figure 9(a) with a solid black line. To begin with, we test the boundary map intersection idea without invoking the surrogate approximation. This means, the fixed point problem $\boldsymbol{y} = \mathcal{F}\boldsymbol{y}$ is solved directly and the components of the operator $\mathcal{F}$, the boundary-to-boundary maps, are computed in each iteration of the fixed point solver. This entails subdomain solves at each inner iteration of the fixed point solver. Clearly, a single application of the fixed point solver finds the true solution, as illustrated in Figure 9. We note that the fixed point solver required 25 iterations in a 5-subdomain case, and 50 iterations in a 10 subdomain case. While this exercise shows that conceptually the fixed-point methodology works, the fixed point solver iterations involve large



(a) Gaussian likelihood, no bit-flips

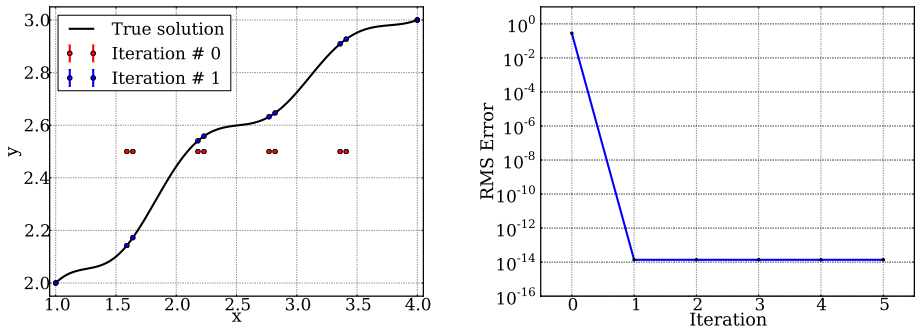
(b) Gaussian likelihood, with bit-flips

(c) Laplace likelihood, no bit-flips

(d) Laplace likelihood, with bit-flips

FIG. 8. *RMS error versus the iteration number for* 100 *different simulations of the linear problem. The bit-flip probability is set to* $p = 0.003$*, while the other parameters are at their default values, i.e.* $M = 15$ *samples,* $N = 5$ *subdomains with overlap size* $h = 0.05$ *and the factor in the surrogate range choice is set to* $f = 0.1$.

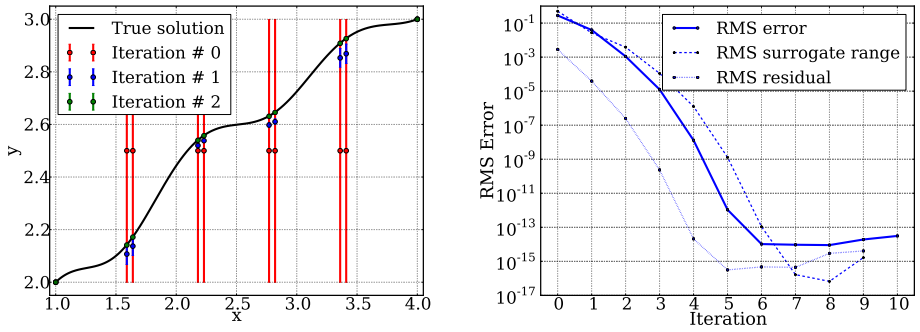
amounts of communication between subdomains and certainly do not meet our goals of scalability and fault resilience. Surrogate maps are therefore constructed *a priori* and used in place of the exact boundary-to-boundary maps as described in Section 2.3.



(a) One iteration of solution with exact maps (b) RMS error versus iteration number

FIG. 9. *Solution of the fixed point problem* $\boldsymbol{y} = \mathcal{F}\boldsymbol{y}$ *using the exact boundary-to-boundary maps without surrogates.*

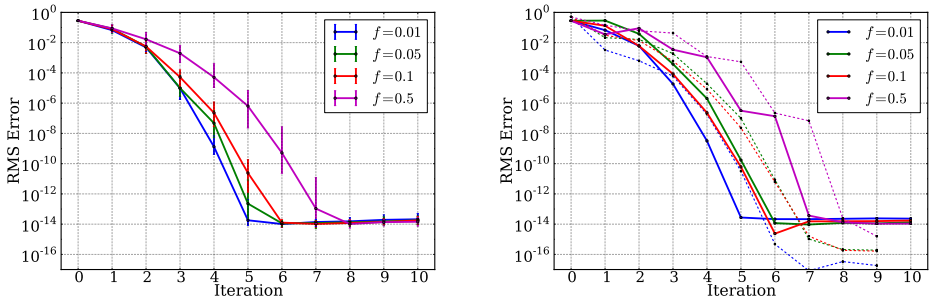
Figure 10 demonstrates the results of the algorithm for the default scenario, *i.e.* linear surrogate maps constructed with $M = 15$ samples (*i.e.* as many PDE solver invocations per subdomain per iteration) and $N = 5$ subdomains. The dashed line on the right plot indicates the ranges averaged in a root-mean-square (RMS) sense over the internal subdomain boundaries, chosen according to the solution difference approach, *i.e.* $r_{(i)} = f|y_{(i)} - y_{(i-1)}|$ for the i-th iteration, with $f = 0.1$.



(a) Three iterations for the non-linear problem (b) RMS error, surrogate range and residual

FIG. 10. *The results of the application of our approach to the non-linear problem. On the left plot, the errorbars indicate the surrogate range for the corresponding iteration. On the right, dashed line indicates the RMS average range chosen at each iteration, i.e. the RMS average of the vector* $\boldsymbol{r}_{(i)}$*, while the dotted line shows the RMS average of the computed residual vector according to Eq. (2.7).*

Furthermore, we analyze the dependence of the convergence on the factor f that helps define the surrogate range by $r_{(i)} = f|y_{(i)} - y_{(i-1)}|$. Figure 11 illustrates the fact that large values of f lead to much slower convergence. This is due to the fact



(a) Median and 25, 75% quantiles over 100 runs (b) Single run (solid) and RMS surrogate range (dashed)

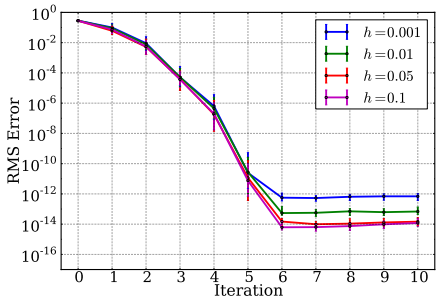
FIG. 11. *The convergence results as the factor f in the surrogate range choice is varied. The parameters are set to $M = 15$, $p = 0.0$, using linear surrogate maps and $N = 5$ subdomains. The left plot shows the RMS error versus iteration number across 100 identical simulations. On the right plot, a single simulation is illustrated for each value of f, together with the corresponding dashed curve that indicates how the RMS average of the surrogate range is evolving.*

that larger surrogate ranges lead to larger errors in the linear approximation of the boundary maps over the surrogate range. As a default value we choose $f = 0.1$, which seems to ensure that the chosen range at each step encompasses the true RMS error between the mean solution and the exact one. While smaller f leads to slightly faster convergence for this particular problem, we chose a more conservative default value $f = 0.1$ to reduce the potential need of extrapolating for general problems less-than-ideal initial solution states, as well as to reduce the impact of nonlinearities and faults in case they are activated. Also the RMS-average of the residual vector, computed from (2.7), is shown in Figure 10(b). The residual is at least a couple of orders of magnitude smaller than the actual error committed.

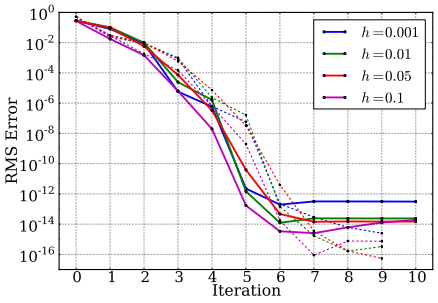
The dependence of the convergence on the size of the overlap between subdomains is demonstrated in Figure 12. Note that a smaller overlap size makes the convergence somewhat slower and raises the final error level. The final error is higher due to worse conditioning of the fixed-point solver due to weaker dependence built in the boundary-to-boundary maps. On the other hand, a larger overlap size would enhance the non-linearity of the boundary-to-boundary maps, resulting in less accurate linear surrogates. Besides, in these tests, larger overlaps result in larger subdomains, and would therefore become more computationally expensive if a discrete solver were used to compute the solutions on each subdomain. We have selected $h = 0.05$ as a reasonable default choice based on these tests.

Next, we illustrate the dependence on the number of subdomains. As Figure 13 clearly indicates, the larger the number of subdomains, the slower the convergence. The intuitive justification for this is that as the number of subdomains increases, it takes more iterations for the information to travel across the computational domain. Note that this behavior is very similar to the slow-down of the convergence of iterative PDE solvers as the number of grid points increases for a given computational domain.

Figure 14 demonstrates the convergence with respect to the number of samples used in the surrogate map construction. Clearly, the results remain the same for all the tested values of M, indicating that $M \geq 10$ is sufficient to accurately construct a two-variate linear surrogate that has three parameters. The convergence curves now

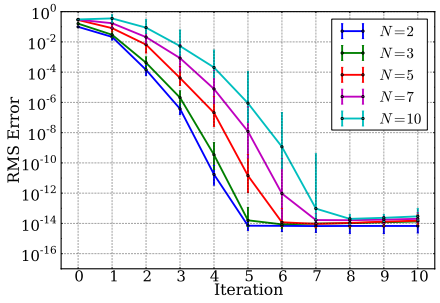


(a) Median and 25, 75% quantiles over 100 runs

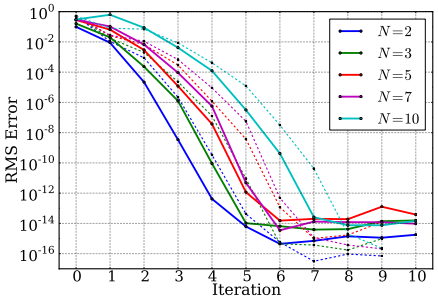


(b) Single run (solid) and RMS surrogate range (dashed)

FIG. 12. *Convergence with the number of iterations for various sizes of domain overlaps with the number of subdomains fixed at $N = 5$. The rest of the parameters are set to $M = 15$, $f = 0.1$, $p = 0.0$, using linear surrogate maps. The left plot shows the RMS error versus iteration number across 100 identical simulations. On the right plot, a single simulation is illustrated for each value of h, together with the corresponding dashed curve that indicates how the RMS average of the surrogate range is evolving.*



(a) Median and 25, 75% quantiles over 100 runs

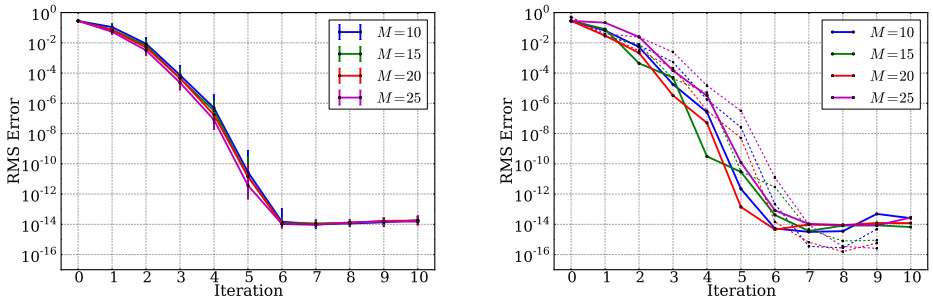


(b) Single run (solid) and RMS surrogate range (dashed)

FIG. 13. *Convergence with the number of iterations for various number of subdomains. The parameters are set to $M = 15$, $h = 0.5$, $f = 0.1$, $p = 0.0$, using linear surrogate maps. The left plot shows the RMS error versus iteration number across 100 identical simulations. On the right plot, a single simulation is illustrated for each value of N, together with the corresponding dashed curve that indicates how the RMS average of the surrogate range is evolving.*

nearly coincide with each other for different values of M. This effect is fairly intuitive, as linear surrogates have 3 coefficients that need to be found, and $M \geq 10$ is more than enough to find the best linear surrogate. In fact, for linear problems, $M = 4$ samples will be sufficient to exactly recover two-variate linear surrogates even with a single bit-flip injected. Triply-redundancy with $M = 12$ samples with default, Gaussian (least-squares) fit may still not be sufficient to recover the true solution, for a linear problem, if, say, one bit-flip is present per redundant solve. For non-linear problem, it is not clear yet how many samples are needed for accurate recovery of the best linear surrogate. We used $M \geq 10$ and no bit-flips to demonstrate the convergence behavior of the algorithm, while our preliminary tests (not shown) indicate that typically even

$M = 8$ is sufficient for having nearly all (among 100) replica simulations converge monotonically for the default, $p = 0.001$ case for both linear and non-linear problems. A more detailed investigation of this trade-off and associated cost compared to naïvely redundant algorithms is a matter of current research.

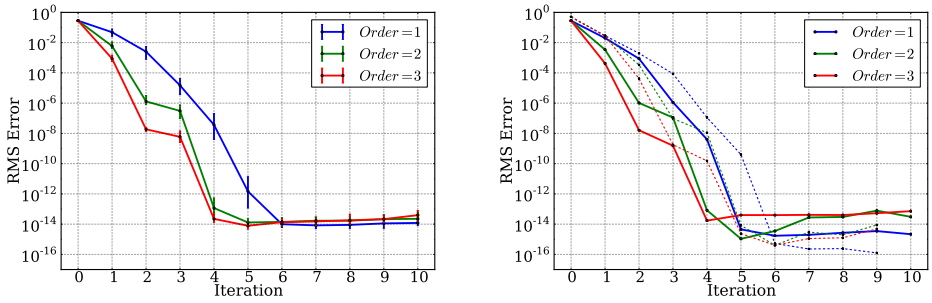


(a) Median and 25, 75% quantiles over 100 runs (b) Single run (solid) and RMS surrogate range (dashed)

FIG. 14. *The convergence results as the number of samples M for linear surrogate construction is changed. The parameters are set to $f = 0.1$, $p = 0.0$ and $N = 5$ subdomains. The left plot shows the RMS error versus iteration number across 100 identical simulations. On the right plot, a single simulation is illustrated for each value of M, together with the corresponding dashed curve that indicates how the RMS average of the surrogate range is evolving.*

Dependence of the results on the surrogate order is illustrated in Figure 15. We have chosen $M = 35$ for all cases to ensure that cubic polynomial surrogate construction is well-defined and is not affected by the sample-to-sample variability. The third-order polynomial surrogate includes 10 terms, and $M = 15$ was found to be insufficient to accurately find the third order fit with small enough ensemble-driven variability. It is clear that higher-order surrogates allow more accurate capture of the true boundary maps and therefore accelerate the convergence. Higher order cases, however, naturally require more samples in order to ensure little variability in the polynomial coefficients. While this demonstration serves as a proof-of-concept, we use linear surrogates primarily.

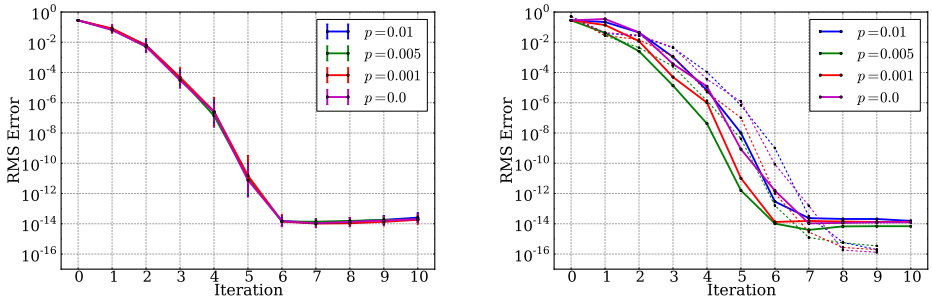
Finally, we demonstrate results with respect to errors in the subdomain solution results. For preliminary tests, we have implemented artificial fault generation by flipping a single, randomly selected bit in binary representations of a small fraction p of the outcomes of subdomain solves. An additional outlier detection is currently invoked on returned solutions that discards samples outside a range of $[-5, 15]$, which is a wide range of plausible output samples given that the boundary conditions are $y_a = 2$ and $y_b = 3$. Such outlier detection causes an effect similar to missing samples and is well-handled by the Bayesian surrogate construction. Figure 16 illustrates the convergence of the root-mean-square error for cases with and without artificial bit-flip injection. As discussed in Section 3.2, the expected number of of bit-flips for $T = 10$ iterations is $N_f = (2N - 2)MpT = 1200p$, *i.e.* 12 expected faults for a single run for $p = 0.01$ case (blue curve on Figure 16(b)) with $N = 5$ subdomains and $M = 15$ boundary-pair samples. However, as the error bars suggest, the bit-flips do not deteriorate the convergence. Even with the largest value of the bit-flip probability, *i.e.* $p = 0.01$, when one expects at least one bit-flip per iteration, the convergence curve indicates error reduction and is fully resilient showing nearly no dependence on



(a) Median and 25, 75% quantiles over 100 runs (b) Single run (solid) and RMS surrogate range (dashed)

FIG. 15. *Convergence with the surrogate order change. The parameters are set to $M = 35$, $N = 5$, $p = 0.0$ and $f = 0.1$. Note that we have used larger number of training samples in order to achieve a well defined third-order surrogate construction. The left plot shows the RMS error versus iteration number across 100 identical simulations. On the right plot, a single simulation is illustrated for each value of the surrogate order, together with the corresponding dashed curve that indicates how the RMS average of the surrogate range is evolving.*

bit-flip probability.



(a) Median and 25, 75% quantiles over 100 runs (b) Single run (solid) and RMS surrogate range (dashed)

FIG. 16. *Convergence with the number of iterations with and without bit-flips. The parameters are set to $M = 15$, $N = 5$, $f = 0.1$, using linear surrogate maps. The left plot shows the RMS error versus iteration number across 100 identical simulations. On the right plot, a single simulation is illustrated for each value of the bit-flip probability, together with the corresponding dashed curve that indicates how the RMS average of the surrogate range is evolving.*

3.4. Error propagation in the fixed-point system. In this section we study the impact of the subdomain surrogate accuracy on the fixed point solution for the non-linear problem. Within each iteration, the goal is to bring the current solution state $\boldsymbol{y}$ closer to the true solution $\tilde{\boldsymbol{y}}$ of the fixed point system (2.6). The magnitude of the error reduction in each iteration is limited, however, by the approximation error of the linear surrogate maps $\mathcal{G}$. Consequently, we study the effect of the current error and the surrogate error on the final error, within one iteration. The current, or initial, error is the RMS average of the vector $\boldsymbol{y}_{(0)} - \tilde{\boldsymbol{y}}$, and the final error is defined as the

RMS average of $\boldsymbol{y}_{(1)} - \tilde{\boldsymbol{y}}$, while the surrogate error is the RMS average of the error between the true boundary-to-boundary map and its linear surrogate. We present the results using the nonlinear example from Section 3.3 merely as an illustration of the error propagation through the fixed-point machinery $\mathcal{F}\boldsymbol{y} \approx \boldsymbol{y}$. Note that for linear problems, the analogous tests lead to trivial results as the final error is always expected to be near machine precision, and the surrogates are exact, no matter how far away the initial guess is from the true solution.

We have generated a random selection of initial conditions $\boldsymbol{y}_{(0)}$, and have chosen four different values for the surrogate range, resulting in four different amounts of surrogate error. The results, shown in Figure 17, suggest that both the initial error and the surrogate error impact the limiting behavior of the final error. In other words, if one starts far away from the true solution, *i.e.* with large initial errors, then the surrogate accuracy has no impact, while if the initial error is small, then the surrogate error directly impacts the magnitude of the final error. The same logic applies with respect to the surrogate error itself, that is, the final error converges with respect to the reduction of the initial error, but the convergence is limited by the error committed in the surrogate construction. For example, if the surrogate error is restricted to approximately 10^{-9} due to the selected surrogate range (blue dots in Figure 17), then, no matter how accurate the initial solution is, the final error is limited to 10^{-7}, simply because approximate surrogates are 'intersected' instead of the true boundary-to-boundary maps. In nearly all cases explored and reported in this paper, however, such 'saturation' is not reached and the error in the solution is reduced by several orders of magnitude in one iteration.

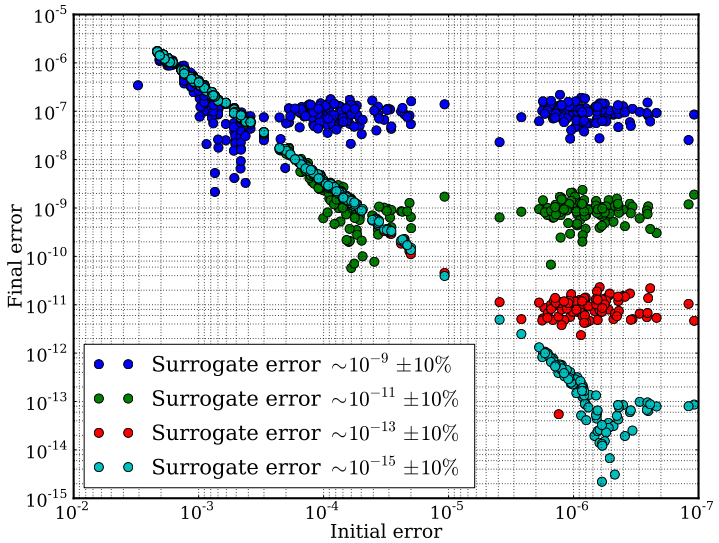


FIG. 17. *The error propagation through one iteration of the surrogate map construction and subsequent fixed point system solution. The plotted is the final RMS error after a single iteration versus the initial RMS error. Four surrogate ranges are chosen inducing four groupings of the associated surrogate map accuracies.*

3.5. Linear PDE with numerical solution on subdomains. Our final test case is a linear PDE

$$-\frac{\partial}{\partial x}\left(k(x)\frac{\partial y}{\partial x}\right) = f(x) \quad \text{in } \Omega = (0,1) \tag{3.6}$$

where the diffusivity and source terms are defined as

$$k(x) = \exp\left[\tanh\left(10\left(x-\frac{1}{2}\right)\right)\right], \qquad f(x) = \sin(5\pi x), \tag{3.7}$$

respectively, with BCs set to $y(0) = 0$ and $y(1) = 1$. In this case, a numerical solver is used to solve the PDE on each subdomain. Figure 18 shows the solution over the full domain, precomputed numerically using a dense grid.

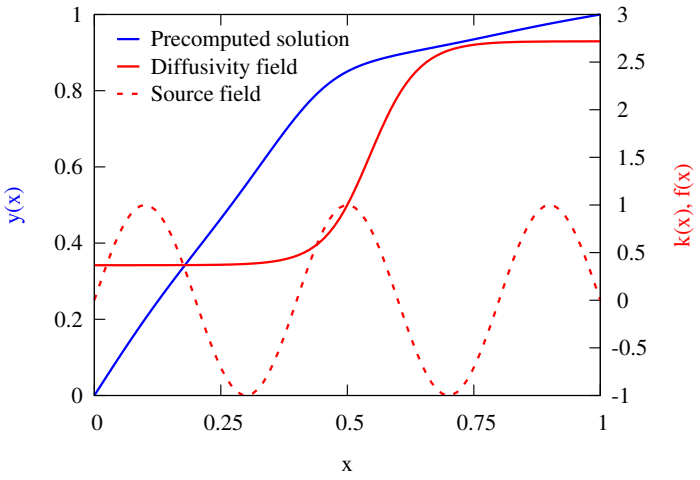
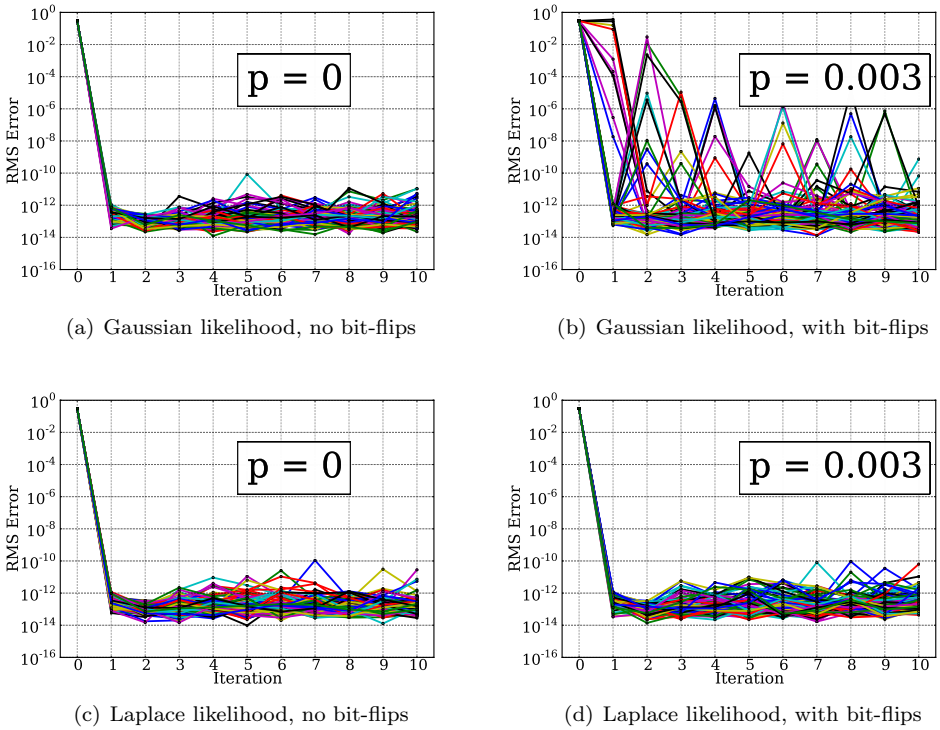


FIG. 18. *The precomputed solution of the diffusion equation (3.6), as well as the diffusivity and source fields.*

As this is a linear PDE, linear surrogates are exact and a single iteration reaches the true solution. Figure 19 illustrates the convergence with or without bit-flips for 100 identical simulations. Both the subdomain solutions and the predefined, true solution are computed on the same grid of size $\delta = 0.001$ explaining the convergence to nearly machine precision even on a finite grid. As discussed in Section 3.2, the expected number of faults per full set of iterations is $N_f = (2N-2)MpT$ and is about 2 for this case. The regression with Laplace likelihoods is computed using the Iteratively Reweighted Least Squares (IRLS) algorithm that provides an accurate and efficient approximation of the Laplace-likelihood objective function [13, 16]. This study clearly illustrates the advantage of using the Laplace likelihood as opposed to the Gaussian likelihood in the regression in order to be fault-resilient.

4. Discussion and future work. In this work, we developed and demonstrated a novel algorithmic approach for solving PDEs that is resilient to multiple soft and hard faults. The approach relies on an overlapping domain decomposition similar to accelerated additive Schwarz methods. At any point in the iterative solution process, the current state of knowledge about the solution is represented as the best current estimate for the solution plus a range within which the fully converged solution is



(a) Gaussian likelihood, no bit-flips

(b) Gaussian likelihood, with bit-flips

(c) Laplace likelihood, no bit-flips

(d) Laplace likelihood, with bit-flips

FIG. 19. *Convergence with the number of iterations with and without bit-flips for the linear diffusion problem (3.6) for both Gaussian (top row) and Laplace (bottom row) likelihoods. The parameters are set to* $M = 8$, $N = 5$, $f = 0.1$, $h = 0.05$. *A total of 100 identical simulations are shown. The grid size for the discrete solver is kept at* $\delta = 0.001$ *for both the reference and subdomain solutions. The weight relaxation parameter for the IRLS-*ℓ_1 *optimization is set to* $\rho = 0.9$.

believed to reside. Using targeted solves of the PDE on the subdomains, for sampled values of their boundary conditions, maps are constructed that relate the solutions at the subdomain boundaries to each other. The intersection of these boundary maps produces an updated solution state, which is refined in successive iterations until a fully converged solution is obtained. The approach is applicable to both linear and non-linear PDEs.

The construction of the subdomain boundary solution maps relies on Bayesian inference, with a Laplace likelihood function that is very well suited to filter out outliers caused by data corruption in the results of subdomain solves. This makes the approach very resilient to faults in the subdomain solves without needing to explicitly detect the occurrence of such faults.

We have demonstrated the algorithm on one-dimensional synthetic PDEs with Dirichlet boundary conditions and with artificial fault injection in the form of bit-flips applied with a specified probability to the results of the subdomain solves. For linear problems, the approach converged to the true numerical solution of the PDE in one iteration, even in the presence of data corruption through bit-flips. For non-linear problems, the algorithm shows good convergence properties along with robustness against data corruption in the subdomain solves.

The approach is currently being extended to multi-dimensional PDEs [35], and

to PDEs with uncertain coefficients, which will be reported on elsewhere.

Acknowledgments. This work is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number 13-016717. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Appendix A. Proof of linearity of boundary maps for linear PDEs.

Consider a linear PDE $\mathcal{L}y(\boldsymbol{x}) = h(\boldsymbol{x})$ with Dirichlet boundary condition $y(\boldsymbol{x})|_{\boldsymbol{x}\in\Gamma} = y_\Gamma$ on the boundary $\Gamma = \partial\Omega$. Now perturb the boundary condition by Δy_Γ, *i.e.* set up a new problem $\mathcal{L}y(\boldsymbol{x}) = h(\boldsymbol{x})$ with boundary condition $y(\boldsymbol{x})|_{\boldsymbol{x}\in\Gamma} = y_\Gamma + \Delta y_\Gamma$. Denoting the differences between the solutions of the original PDE and the perturbed one by $\Delta y(x)$, we note that due to linearity of the operator $\mathcal{L}$, the function $\Delta y(\boldsymbol{x})$ satisfies the homogeneous PDE $\mathcal{L}\Delta y(\boldsymbol{x}) = 0$ with a Dirichlet boundary condition $\Delta y(\boldsymbol{x})|_{\boldsymbol{x}\in\Gamma} = \Delta y_\Gamma$. Due to homogeneity, for each fixed point $\boldsymbol{x}$, $\Delta y(\boldsymbol{x})$ is proportional to the perturbation magnitude Δy_Γ and does not depend on y_Γ itself. One can then write $\Delta y(\boldsymbol{x})/\Delta y_\Gamma = A(\boldsymbol{x})$ thus proving the linearity of the map $y_\Gamma \to y(\boldsymbol{x})$ for each fixed $\boldsymbol{x}$.

REFERENCES

[1] *IEEE Standard for Floating-Point Arithmetic*, tech. report, Microprocessor Standards Committee of the IEEE Computer Society, 2008.

[2] Michele Benzi, Andreas Frommer, Reinhard Nabben, and Daniel B Szyld, *Algebraic theory of multiplicative schwarz methods*, Numerische Mathematik, 89 (2001), pp. 605–639.

[3] J.M. Bernardo and A.F.M. Smith, *Bayesian Theory*, Wiley Series in Probability and Statistics, John Wiley & Sons Ld, Chichester, England, 2000.

[4] George Bosilca, Remi Delmas, Jack Dongarra, and Julien Langou, *Algorithm-based fault tolerance applied to high performance computing*, Journal of Parallel and Distributed Computing, 69 (2009), pp. 410–416.

[5] Patrick G Bridges, Kurt B Ferreira, Michael A Heroux, and Mark Hoemmen, *Fault-tolerant linear solvers via selective reliability*, arXiv.org, (2012).

[6] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM Journal on Scientific Computing, 16 (1995), pp. 1190–1208.

[7] X.-C. Cai, *A family of overlapping Schwarz algorithms for nonsymmetric and indefinite elliptic problems*, in Domain-based parallelism and problem decomposition methods in computational science and engineering, D. Keyes, Y. Saad, and D. Truhlar, eds., SIAM, Providence, RI, 1994.

[8] Franck Cappello, *Fault Tolerance in Petascale/ Exascale Systems: Current Knowledge, Challenges and Research Opportunities*, International Journal of High Performance Computing Applications, 23 (2009), pp. 212–226.

[9] F Cappello, A Geist, B Gropp, L Kale, B Kramer, and M Snir, *Toward Exascale Resilience*, International Journal of High Performance Computing Applications, 23 (2009), pp. 374–388.

[10] Franck Cappello, Al Geist, William Gropp, Sanjay Kale, Bill Kramer, and Marc Snir, *Toward exascale resilience: 2014 update*, Supercomputing frontiers and innovations, 1 (2014).

[11] B. P. Carlin and T. A. Louis, *Bayesian Methods for Data Analysis*, Chapman and Hall/CRC, 2011.

[12] X.-C. Cay, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numerische Matematik, 60 (1991), pp. 41–61.

[13] Rick Chartrand and Wotao Yin, *Iteratively reweighted algorithms for compressive sensing*, in Acoustics, speech and signal processing, 2008. ICASSP 2008. IEEE international conference on, IEEE, 2008, pp. 3869–3872.

[14] Yi Chen, John Jakeman, Claude Gittelson, and Dongbin Xiu, *Local Polynomial Chaos Expansion for Linear Differentical Equations with High Dimensional Random Inputs*, SIAM Journal on Scientific Computing, 37 (2015), pp. A79–A102.

[15] Zizhong Chen, *Algorithm-based recovery for iterative methods without checkpointing*, in Proceedings of the 20th international symposium on High performance distributed computing, HPDC '11, New York, NY, USA, 2011, ACM, pp. 73–84.

[16] Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C Sinan Güntürk, *Iteratively reweighted least squares minimization for sparse recovery*, Communications on Pure and Applied Mathematics, 63 (2010), pp. 1–38.

[17] David Denison, Christopher Holmes, Bani Mallick, and Adrian Smith, *Bayesian methods for nonlinear classification and regression*, vol. 386, John Wiley & Sons, 2002.

[18] Chong Ding, C. Karlsson, Hui Liu, T. Davies, and Zizhong Chen, *Matrix multiplication on gpus with on-line fault tolerance*, in Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on, 2011, pp. 311–317.

[19] M. Dryja and O. Widlund, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Report 339, Dep-t of Comp. Science, Courant Institute, 1987.

[20] Peng Du, Aurelien Bouteiller, George Bosilca, Thomas Herault, and Jack Dongarra, *Algorithm-based fault tolerance for dense matrix factorizations*, in Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming, PPoPP '12, New York, NY, USA, 2012, ACM, pp. 225–234.

[21] Kurt Ferreira, Jon Stearley, James H. Laros, III, Ron Oldfield, Kevin Pedretti, Ron Brightwell, Rolf Riesen, Patrick G. Bridges, and Dorian Arnold, *Evaluating the viability of process replication reliability for exascale systems*, in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis,

SC '11, New York, NY, USA, 2011, ACM, pp. 44:1–44:12.
[22] M. Garbey, *Acceleration of the Schwarz method for elliptic problems*, SIAM Journal on Scientific Computing, 26 (2005), pp. 1871–1893.
[23] M. Garbey and D. Tromeur-Dervout, *On some Aitken-like acceleration of the Schwarz method*, International Journal for Numerical Methods in Fluids, 40 (2002), pp. 1493–1513.
[24] Michael Griebel and Peter Oswald, *On the abstract theory of additive and multiplicative schwarz algorithms*, Numerische Mathematik, 70 (1995), pp. 163–180.
[25] ———, *Greedy and randomized versions of the multiplicative schwarz method*, Linear Algebra and its Applications, 437 (2012), pp. 1596–1610.
[26] M. Holst, *Algebraic Schwarz theory*, Technical Report CRPC-994-10, California Institute of Technology, 1994.
[27] E. T. Jaynes, *Prior Probabilities*, IEEE Transactions on Systems Science and Cybernetics, 4 (1968), pp. 227–241.
[28] D. Keyes, *How scalable is domain decomposition in practice?*, in Proceedings of the 11th International Conference on Domain Decomposition methods, Domain Decomposition Press, 1999, pp. 286–297.
[29] Samuel Kotz and Saralees Nadarajah, *Multivariate t-distributions and their applications*, Cambridge University Press, 2004.
[30] J W Larson, M Hegland, B Harding, S Roberts, L Stals, A P Rendell, P Strazdins, M M Ali, C Kowitz, R Nobes, J Southern, N Wilson, M Li, and Y Oishi, *Fault-Tolerant Grid-Based Solvers: Combining Concepts from Sparse Grids and MapReduce*, Procedia Computer Science, 18 (2013), pp. 130–139.
[31] Dong Li, Jeffrey S. Vetter, and Weikuan Yu, *Classifying soft error vulnerabilities in extreme-scale scientific applications using a binary instrumentation tool*, in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, Los Alamitos, CA, USA, 2012, IEEE Computer Society Press, pp. 57:1–57:11.
[32] Man-Lap Li, Pradeep Ramachandran, Swarup Kumar Sahoo, Sarita V. Adve, Vikram S. Adve, and Yuanyuan Zhou, *Understanding the propagation of hard errors to software and implications for resilient system design*, SIGOPS Oper. Syst. Rev., 42 (2008), pp. 265–276.
[33] K. Malkowski, P. Raghavan, and M. Kandemir, *Analyzing the soft error resilience of linear solvers on multicore multiprocessors*, in Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on, 2010, pp. 1–12.
[34] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Numerical mathematics and scientific computation, Clarendon Press, 1999.
[35] F. Rizzi, K. Morris, K. Sargsyan, P. Mycek, C. Safta, B.J. Debusschere, O. Le Maître, H.N. Najm, and O.M. Knio, *Partial differential equations solver resilient to soft and hard faults*, in preparation, (2015).
[36] A. Savino, S.D. Carlo, G. Politano, A. Benso, A. Bosio, and G. Di Natale, *Statistical reliability estimation of microprocessor-based systems*, Computers, IEEE Transactions on, 61 (2012), pp. 1521–1534.
[37] B. Schroeder and G.A. Gibson, *A large-scale study of failures in high-performance computing systems*, Dependable and Secure Computing, IEEE Transactions on, 7 (2010), pp. 337–350.
[38] John Shalf, Sudip Dosanjh, and John Morrison, *Exascale Computing Technology Challenges*, tech. report, 2011.
[39] A. Shye, T. Moseley, V.J. Reddi, J. Blomstedt, and D.A. Connors, *Using process-level redundancy to exploit multiple cores for transient fault tolerance*, in Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on, 2007, pp. 297–306.
[40] D.S. Sivia, *Data Analysis: A Bayesian Tutorial*, Oxford Science, 1996.
[41] Barry F. Smith, *Domain decomposition methods for partial differential equations*, in Proceedings of ICASE/LaRC Workshop on Parallel Numerical Algorithms, Univ. Press, 1995.
[42] Kian H Tan and Mart JA Borsboom, *On generalized Schwarz coupling applied to advection-dominated problems*, Contemporary Mathematics, 180 (1994), pp. 125–125.
[43] Keita Teranishi and Michael A Heroux, *Toward Local Failure Local Recovery Resilience Model using MPI-ULFM*, in the 21st European MPI Users' Group Meeting, New York, New York, USA, 2014, ACM Press, pp. 51–56.
[44] A. Toselli and O.B. Widlund, *Domain Decomposition Methods - Algorithms and Theory*, Springer Series in Computational Mathematics, Springer, 2005.