DISCRETE A PRIORI BOUNDS FOR THE DETECTION OF CORRUPTED PDE SOLUTIONS IN EXASCALE COMPUTATIONS

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

Abstract. *A priori* bounds are derived for the discrete solution of second-order elliptic partial differential equations (PDEs). The bounds have two contributions. First, the influence of boundary conditions is taken into account through a discrete maximum principle. Second, the contribution of the source field is evaluated in a fashion similar to that used in the treatment of the continuous *a priori* operators. Closed form expressions are in particular obtained for the case of a conservative, second-order finite difference approximation of the diffusion equation with variable scalar diffusivity. The bounds are then incorporated into a resilient domain decomposition framework, in order to verify the admissibility of local PDE solutions. The computations demonstrate that the bounds are able to detect most of system faults, and thus considerably enhance the resilience and the overall performance of the solver.

Key words. elliptic PDE, maximum principle, discrete bounds, resilience, exascale computing, domain decomposition

1. Introduction.

Future exascale systems are expected to suffer from much higher fault rates than today's petascale platforms [40]. This raises many new scientific challenges to achieve reliable computations, one of which being the ability to overcome the occurrence of system faults.

Arguably the most popular fault-tolerance technique for petascale is checkpoint-restart. It relies on periodic saves of the system state, which allows one to restore the system to a previous state whenever an error is detected. The Local Failure Local Recovery (LFLR) strategy, focusing on local checkpointing and recovery was proposed as an improvement of the original global checkpoint-restart [41]. Alternative approaches include algorithm-based fault tolerance (ABFT) [3, 15, 16, 10], effective use of state machine replication [19] or process-level redundancy [39], and algorithmic error correction code [26]. Many other approaches for fault-tolerance in extreme-scale computing have been developed (see, *e.g.*, [8]). In exascale systems, however, the time needed for checkpointing may be close to the mean time between failures [23, 40], thus causing the system to spend an excessive amount of time checkpointing and restarting, rather than advancing towards the solution [8].

To address this issue, we have recently developed a resilient solver for elliptic partial differential equations (PDEs), see [33, 32]. The solver is based on an overlapping domain decomposition method (see, *e.g.*, [30, 43]) and the resilient update involves the solution of local problems (independent from one subdomain to another), which is well suited for massive parallelism. To deal with soft faults, the solver represents the solution as a state-of-knowledge, and updates this state in a resilient manner. In this framework, hard faults, such as a node crashing or a communication failing, are seamlessly treated as missing data and may thus be disregarded.

In the approaches of [33, 32], the resilient update is achieved through robust regression. In this work, we explore the possibility of further improving the resilience capabilities of such techniques by

*Duke University, 144 Hudson Hall, Box 90300, Durham, NC 27708 (paul.mycek@duke.edu).
†Sandia National Laboratories, Livermore, CA (fnrizzi@sandia.gov).
‡Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur, Orsay, France (olm@limsi.fr).
§Sandia National Laboratories, Livermore, CA (ksargsy@sandia.gov).
¶Sandia National Laboratories, Livermore, CA (knmorri@sandia.gov).
‖Sandia National Laboratories, Livermore, CA (csafta@sandia.gov).
**Sandia National Laboratories, Livermore, CA (bjdebus@sandia.gov).
††Duke University, Durham, NC (omar.knio@duke.edu).

checking the admissibility of the local solutions before they are fed to the regression. To this end, we derive *a priori* bounds for the discrete solution of linear elliptic PDEs. Their derivation builds on an existing discrete maximum principle [11], and accounts for the source field in a fashion similar to that used in the derivation of continuous *a priori* bounds [20]. This results in the expression of lower and upper bounds for the discrete solution on a subdomain, for given local boundary conditions.

This idea is motivated by the fact that robust regression techniques can be expensive and may fail if too many samples are faulty. Performing regression may then take more time for some problems with faulty samples than for other uncorrupted problems, thus causing parallel imbalance, which could be detrimental to the scalability of the solver. Moreover, our current machinery for dealing with failed resilient state-of-knowledge updates consists in repeating the update step. A complete new set of boundary conditions is sampled and the corresponding local PDE solves and regressions need to be performed again, resulting approximately in the doubling of the original computational time. With the introduction of the bounds to check the admissibility of the local PDE solutions, we expect to limit the occurrence of such scenarios.

It is worth mentioning that alternative options were pursued before opting for the *a priori* bounds that are developed in this paper. Specifically, we initially considered the use of more general matrix norms to derive local bounds on the PDE solution, which only required conditions that are usually fulfilled by the matrices involved in classical numerical methods. More precisely, solving a discretized PDE problem usually amounts to solving a matrix system of the form $\mathbf{A}\mathbf{u} = \mathbf{b}$, where $\mathbf{u}$ represents the unknowns. Using well-known bounds for the L_∞ norm of the matrix inverse [38, 25, 21], this readily yields $\|\mathbf{u}\|_\infty \leqslant \|\mathbf{A}^{-1}\|_\infty \|\mathbf{b}\|_\infty$, as $\|\mathbf{b}\|_\infty$ is cheap and easy to determine. However, these inverse matrix bounds, which are also easy to compute by simple inspection of the elements of $\mathbf{A}$, scale poorly with the matrix size. For instance, for a simple 1D Laplace problem discretized using a second-order finite difference scheme, the upper bound estimate for $\|\mathbf{A}^{-1}\|_\infty$ reaches 10^{29} for only 100 unknowns.

Another alternative we pursued consisted in deriving continuous bounds for the solution of PDEs on a given inner interface, corresponding to the boundary of a neighboring overlapping subdomain. Using functional analysis, we derived bounds for $\| \int_\Gamma u(s)\mathbf{n}(s)\, ds\|_2$, where u denotes the solution and where $\Gamma \doteq \partial\Omega_j \cap \Omega_i$ denotes the part of the boundary $\partial\Omega_j$ that lies in Ω_i. For the diffusion equation, such bounds involve the measure of the interface, the measure of the overlap $\Omega_i \cap \Omega_j$, the norm of the source term on Ω_i, the minimum diffusivity as well as the Poincaré constant on Ω_i. While appealing, the approach however provides a continuous bound that does not take discretization errors into account.

The approach presented in this paper is based on discrete bounds that are at the same time reasonably sharp and general enough to apply to a wide variety of numerical methods such as finite elements or finite differences. They involve constants that are problem-specific and that depend on the numerical scheme used for the discretization as well. Their implementation is illustrated for the case of conservative, second-order finite difference approximation of the steady diffusion equation, for which suitable expressions for the appropriate constants are obtained.

The paper is organized as follows. Section 2 is dedicated to the derivation of the discrete bounds. Relevant results for continuous and discrete operators are first briefly summarized, and general discrete *a priori* bounds are then derived for discretized linear elliptic PDEs. The optimal constants are determined for the second-order finite difference approximation of the diffusion equation, and 1D numerical examples are shown. Section 3 then illustrates the usefulness of the bounds for our algorithm in the context of resilience to faults. After presenting detection rates for 2D examples, we demonstrate how the bounds enhance the performance of the algorithm, namely by improving its

success rates for limited sampling rates. Finally, general conclusions as well as a discussion of the proposed strategy, including ongoing work and potential improvements, are presented in Section 4.

2. Discrete bounds. In this Section, we derive *a priori* bounds for the discrete solution of second-order, elliptic PDEs. We first recall two well known results for the continuous case. Let $\Omega \subset \mathbb{R}^D$ be an open bounded domain, with boundary $\partial\Omega$ and closure $\bar{\Omega} \doteq \Omega \cup \partial\Omega$, and let $\mathcal{L}$ be an elliptic, second-order operator defined as:

$$\mathcal{L}u = a^{ij}_{\mathcal{L}}(\mathbf{x})D_{ij}u + b^{i}_{\mathcal{L}}(\mathbf{x})D_i u + c_{\mathcal{L}}(\mathbf{x})u, \tag{2.1}$$

where $u \in C^0(\bar{\Omega}) \cap C^2(\Omega)$, $D_i \doteq \partial/\partial x^i$, $D_{ij} \doteq \partial^2/\partial x^i \partial x^j$ and the matrix $[a^{ij}_{\mathcal{L}}(\mathbf{x})]$ is symmetric positive-definite. In Eq. (2.1), $[b^{i}_{\mathcal{L}}(\mathbf{x})]$ is a vector field, $c_{\mathcal{L}}(\mathbf{x})$ is a scalar field, and the summation convention is understood, *i.e.* repeated indices imply a summation over those indices. Then, the following maximum principle holds [20]:

THEOREM 2.1. *Let $\mathcal{L}$ be an elliptic operator of the form (2.1) with $c_{\mathcal{L}} \leqslant 0$, and let $u \in C^0(\bar{\Omega}) \cap C^2(\Omega)$ such that $\mathcal{L}u \geqslant 0$ in Ω. Then $\sup_\Omega u \leqslant \sup_{\partial\Omega} u^+$, where $u^+ \doteq \max(0, u)$. If $c_{\mathcal{L}} = 0$, then $\sup_\Omega u \leqslant \sup_{\partial\Omega} u$.*

Furthermore, we have the following *a priori* bounds for the solution [20]:

THEOREM 2.2. *Let $\mathcal{L}$ be an elliptic operator of the form (2.1) with $c_{\mathcal{L}} \leqslant 0$, and let $u \in C^0(\bar{\Omega}) \cap C^2(\Omega)$ such that $\mathcal{L}u = f$ in a bounded domain Ω. Then*

$$\sup_\Omega |u| \leqslant \sup_{\partial\Omega} |u| + C \sup_\Omega \left(|f|/\lambda\right), \tag{2.2}$$

where C is a constant depending only on $\operatorname{diam}\Omega$ *and on $\beta \doteq \sup_\Omega(\|\mathbf{b}_{\mathcal{L}}\|/\lambda)$, and where $\lambda(\mathbf{x})$ denotes the minimum eigenvalue of the matrix $[a^{ij}(\mathbf{x})]$. In particular, if Ω lies between two parallel planes a distance d apart, then* (2.2) *is satisfied with $C = e^{(\beta+1)d} - 1$.*

In this paper, we are interested in the solution of Dirichlet problems defined as follows: given a function f defined over Ω and $g \in C^0(\partial\Omega)$, find $u \in C^0(\bar{\Omega}) \cap C^2(\Omega)$ such that

$$\begin{cases} \mathcal{L}u(\mathbf{x}) = f(\mathbf{x}) & \forall \mathbf{x} \in \Omega \\ u(\mathbf{x}) = g(\mathbf{x}) & \forall \mathbf{x} \in \partial\Omega. \end{cases} \tag{2.3}$$

The function f is usually referred to as the source term.

2.1. Discretized elliptic problem. We now describe the discretized Dirichlet problem. The matrix formulation is general enough so that it can apply to a wide range of discretization schemes.

First, we define the grid as a family $\bar{\Omega}_h \doteq \{\mathbf{x}_i\}_{i=1}^{n_\mathrm{t}} \subset \bar{\Omega}$ of $n_\mathrm{t} \doteq n + m$ points of $\bar{\Omega}$, where $\Omega_h \doteq \{\mathbf{x}_i\}_{i=1}^{n} \subset \Omega$ is the set of n interior points, and $\partial\Omega_h \doteq \{\mathbf{x}_{n+i}\}_{i=1}^{m} \subset \partial\Omega$ is the set of m boundary points. In addition, let u_i, for $i \in \{1, \ldots, n_\mathrm{t}\}$ denote the discrete approximation of the solution to the Dirichlet problem (2.3) at point $\mathbf{x}_i$. The values u_i corresponding to the interior points, *i.e.* for $i \in \{1, \ldots, n\}$, represent unknown values of the discrete approximation, while the values u_{n+i} corresponding to boundary points, *i.e.* for $i \in \{1, \ldots, m\}$, are prescribed by discrete boundary conditions g_i. Generally, $g_i = g(\mathbf{x}_{n+i})$, for $i \in \{1, \ldots, m\}$.

In the following, we assume that the discrete Dirichlet problem can be written in matrix form according to:

$$\mathbf{A}\mathbf{u} = \mathbf{b} - \tilde{\mathbf{A}}\mathbf{g}. \tag{2.4}$$

We need to determine the unknown vector $\mathbf{u} = (u_1, \dots, u_n) \in \mathbb{R}^n$, given the discrete source term $\mathbf{b} = (b_1, \dots, b_n) \in \mathbb{R}^n$ and discrete boundary conditions $\mathbf{g} = (g_1, \dots, g_m) \in \mathbb{R}^m$. The matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ (resp. $\tilde{\mathbf{A}} \in \mathbb{R}^{n \times m}$) will be referred to as the *interior matrix* (resp. the *boundary matrix*). This classical form of the discrete problem can be obtained in a wide range of numerical methods, including finite differences and finite elements.

For convenience (see, *e.g.*, [11, 22]), the system (2.4) will be written in an augmented form as

$$\bar{\mathbf{A}}\bar{\mathbf{u}} = \bar{\mathbf{b}}, \quad \text{with } \bar{\mathbf{A}} \doteq \begin{bmatrix} \mathbf{A} & \tilde{\mathbf{A}} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad \text{and } \bar{\mathbf{b}} \doteq \begin{bmatrix} \mathbf{b} \\ \mathbf{g} \end{bmatrix}, \tag{2.5}$$

with $\bar{\mathbf{u}} = (u_1, \dots, u_{n_t}) \in \mathbb{R}^{n_t}$, $\mathbf{0} \in \mathbb{R}^{m \times n}$ and $\mathbf{I} \in \mathbb{R}^{m \times m}$. Then the matrix $\bar{\mathbf{L}} \doteq [\mathbf{A}, \tilde{\mathbf{A}}] \in \mathbb{R}^{n \times n_t}$ can be viewed as a discrete version of the operator $\mathcal{L}$ in (2.1), and will thus be referred to as the discrete elliptic operator. We will assume in what follows that $\bar{\mathbf{L}}$ can be decomposed as $\bar{\mathbf{L}}(a_\mathcal{L}, b_\mathcal{L}, c_\mathcal{L}) = \bar{\mathbf{L}}^0(a_\mathcal{L}, b_\mathcal{L}) + \bar{\mathbf{R}}(c_\mathcal{L})$, where $\bar{\mathbf{L}}^0 = [\bar{l}^0_{ij}] \in \mathbb{R}^{n \times n_t}$ is such that

$$\sum_{j=1}^{n_t} \bar{l}^0_{ij} = 0, \quad \forall i \in \{1, \dots, n\}. \tag{2.6}$$

This last condition implies that for any uniform vector $\bar{\mathbf{v}} = \lambda \bar{\mathbf{1}} \in \mathbb{R}^{n_t}$ with $\lambda \in \mathbb{R}$ and $\bar{\mathbf{1}} \doteq (1, \dots, 1) \in \mathbb{R}^{n_t}$, $\bar{\mathbf{L}}^0 \bar{\mathbf{v}} = \mathbf{0} \in \mathbb{R}^n$. In such a decomposition, $\bar{\mathbf{R}}\bar{\mathbf{u}}$ generally represents the discrete counterpart of the reactive term $c_\mathcal{L}(\mathbf{x})u$ in (2.1).

2.2. Discrete maximum principle. In what follows, the inequalities between matrices or vectors should be understood elementwise. In addition, for any scalar $a \in \mathbb{R}$, we denote $a^- \doteq \min\{0, a\}$ and $a^+ \doteq \max\{0, a\}$.

From the matrix form described in the previous paragraphs, a discrete maximum principle can be introduced with the following definition [11]:

DEFINITION 2.3. *A discrete elliptic operator $\bar{\mathbf{L}}$ is said to satisfy the discrete maximum principle if and only if any $\bar{\mathbf{u}} \in \mathbb{R}^{n_t}$ satisfying $\bar{\mathbf{L}}\bar{\mathbf{u}} \leqslant \mathbf{0}$ is such that:*

$$\max_{1 \leqslant i \leqslant n} u_i \leqslant \max_{1 \leqslant i \leqslant m} u^+_{n+i}. \tag{2.7}$$

Then, the following three theorems (from [11]) give conditions under which the discrete operator satisfies the discrete maximum principle.

THEOREM 2.4. *A discrete elliptic operator $\bar{\mathbf{L}}$ satisfies the discrete maximum principle if and only if the extended matrix $\bar{\mathbf{A}}$ in* (2.5) *is monotone (*i.e. *for any $\bar{\mathbf{r}} \in \mathbb{R}^{n_t}$, $\bar{\mathbf{A}}\bar{\mathbf{r}} \geqslant \mathbf{0}$ implies $\bar{\mathbf{r}} \geqslant \mathbf{0}$, see* e.g. *[13, 4, 44]), and $-\mathbf{A}^{-1}\tilde{\mathbf{A}}\tilde{\mathbf{1}} \leqslant \mathbf{1}$, where $\tilde{\mathbf{1}} \doteq (1, \dots, 1) \in \mathbb{R}^m$ and $\mathbf{1} \doteq (1, \dots, 1) \in \mathbb{R}^n$.*

THEOREM 2.5. *A discrete elliptic operator $\bar{\mathbf{L}} = [\bar{l}_{ij}]$ satisfies the discrete maximum principle if and only if the extended matrix $\bar{\mathbf{A}}$ in* (2.5) *is monotone and $\sum_{i=1}^{n_t} \bar{l}_{ij} \geqslant 0$, for any $i \in \{1, \dots, n\}$.*

THEOREM 2.6. *A discrete elliptic operator $\bar{\mathbf{L}}$ satisfies the discrete maximum principle if the extended matrix $\bar{\mathbf{A}}$ in* (2.5) *has the following properties:*

(i) The diagonal elements of $\mathbf{A}$ are all positive, while the off-diagonal elements of $\mathbf{A}$, as well as the elements of $\tilde{\mathbf{A}}$, are all nonpositive.

(ii) The matrix $\mathbf{A}$ is irreducible, i.e. *it cannot be transformed into a block upper triangular matrix by permutations of rows and columns.*

(iii) $\forall i \in \{1, \dots, n\}$, $\sum_{j=1}^{n_t} \bar{l}_{ij} \geqslant 0$ and $\exists i \in \{1, \dots, n\}; \sum_{j=1}^{n} a_{ij} > 0$.

The proofs of these theorems are given in [11]. As can be seen in [11], Theorem 2.6 can generally be used for low order schemes (such as second-order finite differences, as we shall see in Section 2.4), while more general theorems, such as Theorem 2.5, are needed for higher order schemes.

2.3. Discrete *a priori* bounds. In this subsection, we derive discrete *a priori* bounds for the solution of the discrete Dirichlet problem (2.5). These bounds represent the discrete counterpart of those given by Theorem 2.2 in the continuous case. The proof is actually similar to the continuous one given in [20], except that we explicitly distinguish the lower and the upper bound, and refine these bounds in the case where $\bar{\mathbf{R}} = \mathbf{0}$, for the purposes of the application to resilience presented in Section 3 below.

THEOREM 2.7. *Let Ω_h be such that there exists $d \in \{1, \dots, D\}$ for which $O \leqslant x_i^d \leqslant O + H$ for any $\mathbf{x}_i = (x_i^1, \dots, x_i^d) \in \Omega_h$, where O and H denote real numbers. Let $\bar{\mathbf{L}} = \bar{\mathbf{L}}^0 + \bar{\mathbf{R}}$ be a discrete elliptic operator with $\bar{\mathbf{R}} \leqslant \mathbf{0}$ such that $-\bar{\mathbf{L}}$ satisfies the discrete maximum principle, and let $\bar{\mathbf{u}} \in \mathbb{R}^{n_t}$ such that $\bar{\mathbf{L}}\bar{\mathbf{u}} = \mathbf{b}$, with $\mathbf{b} \in \mathbb{R}^n$. Define $\bar{\mathbf{w}} = (w_1, \dots, w_{n_t}) \in \mathbb{R}^{n_t}$ by $w_i = \exp(\alpha x_i^d)$, with $\alpha \in \mathbb{R}$, for $i \in \{1, \dots, n_t\}$. If there exists $\alpha \geqslant 0$ such that $\bar{\mathbf{L}}^0\bar{\mathbf{w}} \geqslant \boldsymbol{\lambda}$ for some $\boldsymbol{\lambda} \in \mathbb{R}^n$ with $\boldsymbol{\lambda} > \mathbf{0}$, then*

$$\begin{cases} \min\limits_{1 \leqslant i \leqslant n} u_i \geqslant \min\limits_{1 \leqslant i \leqslant m} u_{n+i}^- - C \max\limits_{1 \leqslant i \leqslant n} (b_i^+/\lambda_i), \\ \max\limits_{1 \leqslant i \leqslant n} u_i \leqslant \max\limits_{1 \leqslant i \leqslant m} u_{n+i}^+ - C \min\limits_{1 \leqslant i \leqslant n} (b_i^-/\lambda_i), \end{cases} \tag{2.8}$$

with $C \doteq e^{\alpha H} - 1$. If $\bar{\mathbf{R}} = \mathbf{0}$, then

$$\begin{cases} \min\limits_{1 \leqslant i \leqslant n} u_i \geqslant \min\limits_{1 \leqslant i \leqslant m} u_{n+i} - C \max\limits_{1 \leqslant i \leqslant n} (b_i^+/\lambda_i), \\ \max\limits_{1 \leqslant i \leqslant n} u_i \leqslant \max\limits_{1 \leqslant i \leqslant m} u_{n+i} - C \min\limits_{1 \leqslant i \leqslant n} (b_i^-/\lambda_i), \end{cases} \tag{2.9}$$

Proof. We restrict ourselves to the case where $0 \leqslant x_i^d \leqslant H$ for any $\mathbf{x}_i = (x_i^1, \dots, x_i^d) \in \Omega_h$. The translation to $O \leqslant x_i^d \leqslant O + H$ is straightforward. Let $\boldsymbol{\lambda} \in \mathbb{R}^{n_t}$ with $\boldsymbol{\lambda} > \mathbf{0}$ and let $\bar{\mathbf{v}} = (v_1, \dots, v_{n_t}) \in \mathbb{R}^{n_t}$ be defined elementwise as:

$$v_i \doteq \max_{1 \leqslant j \leqslant m} u_{n+j}^+ + \left(e^{\alpha H} - e^{\alpha x_i^d}\right) \max_{1 \leqslant j \leqslant n} \left(|b_j^-|/\lambda_j\right) \geqslant 0, \quad \forall i \in \{1, \dots, n_t\}. \tag{2.10}$$

Because $\bar{\mathbf{R}} \leqslant \mathbf{0}$, it follows that $\bar{\mathbf{R}}\bar{\mathbf{v}} \leqslant \mathbf{0}$ and thus $\bar{\mathbf{L}}\bar{\mathbf{v}} \leqslant \bar{\mathbf{L}}^0\bar{\mathbf{v}}$. Let $\alpha \geqslant 0$ such that $\bar{\mathbf{L}}^0\bar{\mathbf{w}} \geqslant \boldsymbol{\lambda}$, then

$$\bar{\mathbf{L}}^0\bar{\mathbf{v}} = -\max_{1 \leqslant j \leqslant n} \left(|b_j^-|/\lambda_j\right) \bar{\mathbf{L}}^0\bar{\mathbf{w}} \leqslant -\max_{1 \leqslant j \leqslant n} \left(|b_j^-|/\lambda_j\right) \boldsymbol{\lambda}. \tag{2.11}$$

Let us take $\bar{\mathbf{u}}$ such that $\bar{\mathbf{L}}\bar{\mathbf{u}} \geqslant \mathbf{b}$, then

$$\left[\bar{\mathbf{L}}(\bar{\mathbf{v}} - \bar{\mathbf{u}})\right]_i = (\bar{\mathbf{L}}\bar{\mathbf{v}})_i - (\bar{\mathbf{L}}\bar{\mathbf{u}})_i \leqslant -\lambda_i \left[\max_{1 \leqslant j \leqslant n} \left(|b_j^-|/\lambda_j\right) + b_i/\lambda_i\right] \leqslant 0, \quad \forall i \in \{1, \dots, n\}. \tag{2.12}$$

Because $-\bar{\mathbf{L}}$ satisfies the discrete maximum principle and because $-\bar{\mathbf{L}}(\bar{\mathbf{u}} - \bar{\mathbf{v}}) \leqslant \mathbf{0}$, it follows by definition that $\bar{\mathbf{u}} - \bar{\mathbf{v}}$ is such that

$$\max_{1 \leqslant i \leqslant m} (u_i - v_i) \leqslant \max_{1 \leqslant i \leqslant m} (u_{n+i} - v_{n+i})^+, \tag{2.13}$$

where

$$u_{n+i} - v_{n+i} = u_{n+i} - \max_{1 \leqslant j \leqslant m} u_{n+j}^+ - \left(e^{\alpha H} - e^{\alpha x_{n+i}^d}\right) \max_{1 \leqslant j \leqslant n} \left(|b_j^-|/\lambda_j\right) \leqslant 0, \quad \forall i \in \{1, \dots, m\}. \tag{2.14}$$

As a consequence, $\max_{1 \leqslant i \leqslant n}(u_i - v_i) \leqslant 0$ and then

$$\max_{1 \leqslant i \leqslant n} u_i \leqslant \max_{1 \leqslant i \leqslant n} v_i \leqslant \max_{1 \leqslant i \leqslant m} u_{n+i} + C \max_{1 \leqslant i \leqslant n} \left(|b_i^-|/\lambda_i\right), \quad C \doteq e^{\alpha H} - 1. \tag{2.15}$$

The bounds in (2.8) are obtained from (2.15) by taking $\bar{\mathbf{u}}$ such that $\bar{\mathbf{L}}\bar{\mathbf{u}} = \mathbf{b}$ and noticing that $\bar{\mathbf{L}}\bar{\mathbf{u}} \geqslant \mathbf{b}$ *and* $\bar{\mathbf{L}}(-\bar{\mathbf{u}}) \geqslant -\mathbf{b}$.

If $\bar{\mathbf{R}} = \mathbf{0}$, then the condition $\mathbf{v} \geqslant \mathbf{0}$ is no longer needed as we can work directly on $\bar{\mathbf{L}}$. Then we can take $\mathbf{v}$ defined elementwise as

$$v_i \doteq \max_{1 \leqslant j \leqslant m} u_{n+j} + \left(e^{\alpha H} - e^{\alpha x_i^d}\right) \max_{1 \leqslant j \leqslant n} \left(|b_j^-|/\lambda_j\right), \quad \forall i \in \{1, \dots, n_\mathrm{t}\}, \tag{2.16}$$

and the rest of the proof remains valid, leading to the bounds in (2.9). □

The difference between the continuous and the discrete case is that in the former, the ellipticity of the operator suffices to fully define the constant C in Theorem 2.2. In the discrete case, however, the parameter α appearing in C depends on the discrete operator. In the following subsection, we derive a condition on α in the case of a conservative, second-order finite difference (FD) scheme for the scalar diffusion equation. We shall see that under such conditions, the *a priori* bounds in Theorem 2.7 apply to the corresponding discrete solution.

2.4. Second order finite differences. We now focus on the conservative, second-order FD approximation of the scalar diffusion equation. This particular problem will subsequently be used in Section 3 to illustrate our fault detection approach based on the bounds derived above. We show that Theorem 2.7 applies to this discrete operator. In particular, we show that we can find $\alpha \geqslant 0$ and $\boldsymbol{\lambda} > \mathbf{0}$ such that the conditions of Theorem 2.7 are satisfied.

2.4.1. Grid definition. We restrict our attention to a regular grid, that is a grid aligned with the coordinates and with uniform spacing h^d in each dimension. Specifically, we assume that Ω_h is contained in a box such that

$$\Omega_h \subset \bigtimes_{d=1}^{D} (O^d; O^d + H^d), \quad \text{with } O^d \in \mathbb{R}, \quad H^d = n_\mathrm{c}^d h^d \in \mathbb{R}, \quad \forall d \in \{1, \dots, D\}, \tag{2.17}$$

where $n_\mathrm{c}^d \in \mathbb{N}$ denotes the number of 1d cells (or elements) in the dth direction. Then the grid is defined as follows:

$$\bar{\Omega}_h = \left\{\mathbf{x} \in \bar{\Omega} \mid \forall d \in \{1, \dots, D\},\, x^d = O^d + kh^d,\, k \in \{0, \dots, n_\mathrm{c}^d\}\right\}, \tag{2.18}$$

$$\Omega_h = \left\{\mathbf{x} \in \bar{\Omega} \mid \forall d \in \{1, \dots, D\},\, x^d = O^d + kh^d,\, k \in \{1, \dots, n_\mathrm{c}^d - 1\}\right\}, \tag{2.19}$$

$$\partial\Omega_h = \bar{\Omega}_h \backslash \Omega_h, \tag{2.20}$$

with a given ordering of the $n_\mathrm{t} = n + m$ points $\mathbf{x}_i \in \bar{\Omega}_h$ and with

$$n_\mathrm{t} = \operatorname{card} \bar{\Omega}_h = \prod_{d=1}^{D} (n_\mathrm{c}^d + 1), \qquad n = \operatorname{card} \Omega_h = \prod_{d=1}^{D} (n_\mathrm{c}^d - 1), \qquad m = \operatorname{card} \partial\Omega_h = n_\mathrm{t} - n. \tag{2.21}$$

2.4.2. Discrete diffusion equation. We consider the (continuous) scalar diffusion operator in D dimensions, in its conservative (divergence) form:

$$\mathcal{L}u \doteq \nabla \cdot [\kappa \nabla u] = \sum_{d=1}^{D} \mathcal{L}^d u, \quad \text{with } \mathcal{L}^d u \doteq \frac{\partial}{\partial x^d}\left[\kappa \frac{\partial u}{\partial x^d}\right], \tag{2.22}$$

where $\kappa(\mathbf{x}) > 0$ denotes the diffusivity field. Provided that κ is differentiable in Ω, $\mathcal{L}$ can be written in a non-conservative form as in Eq. (2.1). We restrict our attention to the following conservative, second-order FD approximation $\bar{\mathbf{L}}$ of the continuous operator $\mathcal{L}$ (see, *e.g.*, [37]):

$$(\bar{\mathbf{L}}\bar{\mathbf{u}})_i \doteq \sum_{d=1}^{D} \frac{\kappa_i^{d-}\, u(\mathbf{x}_i - h^d\mathbf{e}^d) - \left[\kappa_i^{d-} + \kappa_i^{d+}\right] u(\mathbf{x}_i) + \kappa_i^{d+}\, u(\mathbf{x}_i + h^d\mathbf{e}^d)}{\left[h^d\right]^2}, \quad \forall i \in \{1,\dots,n\}, \tag{2.23}$$

where $\kappa_i^{d\pm} \doteq \kappa\left(\mathbf{x}_i \pm (h^d/2)\mathbf{e}^d\right)$. Note that the P_1 (piecewise linear) finite element (FE) discretization of the Dirichlet problem, with piecewise constant diffusivity on the elements, leads to the same definition of $\bar{\mathbf{L}}$. This operator can be recast as

$$(\bar{\mathbf{L}}\bar{\mathbf{u}})_i = \sum_{d=1}^{D} \tilde{\kappa}_i^d\, \frac{u(\mathbf{x}_i - h^d\mathbf{e}^d) - 2u(\mathbf{x}_i) + u(\mathbf{x}_i + h^d\mathbf{e}^d)}{\left[h^d\right]^2} + \tilde{\nabla}_i^d\kappa\, \frac{u(\mathbf{x}_i + h^d\mathbf{e}^d) - u(\mathbf{x}_i - h^d\mathbf{e}^d)}{2h^d}, \tag{2.24}$$

with $\tilde{\kappa}_i^d \doteq (\kappa_i^{d-} + \kappa_i^{d+})/2$ and $\tilde{\nabla}_i^d\kappa \doteq (\kappa_i^{d+} - \kappa_i^{d-})/h^d$.

2.4.3. Discrete *a priori* bounds. Let us now reformulate Theorem 2.7 for this particular operator. We first introduce three lemmas that eventually lead to Theorem 2.11, which corresponds to the application of Theorem 2.7 to the conservative, second-order FD operator described above.

LEMMA 2.8. *Let* $\beta^d \doteq \max_{1\leqslant i\leqslant n}\left(\left|\tilde{\nabla}_i^d\kappa\right| \Big/ \tilde{\kappa}_i^d\right)$ *with* $j \in \{1,\dots,D\}$, *and let* h^d *denote the mesh size in the* d*th direction. Then* $\beta^d h^d < 2$.

Proof. By definition, we have

$$\frac{|\tilde{\nabla}_i^d\kappa|}{\tilde{\kappa}_i^d} = \frac{2}{h^d}\frac{|\kappa_i^{d+} - \kappa_i^{d-}|}{\kappa_i^{d+} + \kappa_i^{d-}}. \tag{2.25}$$

The result follows directly from the triangle inequality and the fact that $\kappa > 0$ in Ω. ☐

LEMMA 2.9. *Let* Ω_h *such that there exists* $d \in \{1,\dots,D\}$ *for which* $x_i^d > 0$ *for any* $\mathbf{x}_i \in \Omega_h$. *Define* $\bar{\mathbf{w}} = (w_1,\dots,w_{n_t}) \in \mathbb{R}^{n_t}$ *by* $w_i = \exp(\alpha x_i^d)$, *with* $\alpha \in \mathbb{R}$. *If*

$$\alpha \geqslant \frac{1}{h^d}\log\left[\frac{h^d\left(\left[h^d\right]^2 + \left[\beta^d\right]^2 + 4\right)^{1/2} + \left[h^d\right]^2 + 2}{2 - \beta^d h^d}\right] \geqslant 0, \tag{2.26}$$

then $(\bar{\mathbf{L}}\bar{\mathbf{w}})_i \geqslant \tilde{\kappa}_i^d$ *for* $i \in \{1,\dots,n\}$.

Proof. Hereafter we drop the d superscript on h and β for clarity. Let us study the function $G(\alpha)$ defined as

$$G(\alpha) \doteq \frac{e^{-\alpha h} - 2 + e^{\alpha h}}{h^2} - \beta\,\frac{e^{\alpha h} - e^{-\alpha h}}{2h} - 1. \tag{2.27}$$

After a few manipulations, it follows that $G(\alpha) \geqslant 0$ is equivalent to

$$g(y) \doteq (\beta h - 2)y^2 + 2(h^2+2)y - (\beta h + 2) \leqslant 0, \quad y \doteq e^{\alpha h}. \tag{2.28}$$

The roots of the second-degree polynomial, g, are:

$$y_1 = \frac{h\sqrt{h^2+\beta^2+4} - h^2 - 2}{\beta h - 2}, \quad y_2 = \frac{-h\sqrt{h^2+\beta^2+4} - h^2 - 2}{\beta h - 2}. \tag{2.29}$$

Because $\kappa > 0$, it follows from Lemma 2.8 that $\beta h < 2$. Then, it is easy to see that $y_1 \in (0,1)$ and $y_2 \in (1,+\infty)$, so the only positive solution of $G(\alpha) = 0$ is $\alpha = \log(y_2)/h$. A simple study of the variations of g shows that if $\alpha \geqslant \log(y_2)/h$, that is if condition (2.26) is satisfied, then $G(\alpha) \geqslant 0$. As a consequence, assuming that condition (2.26) is satisfied, it follows that:

$$(\bar{\mathbf{L}}\bar{\mathbf{w}})_i = e^{\alpha x_i^d}\tilde{\kappa}_i^d \left[\frac{e^{-\alpha h} - 2 + e^{\alpha h}}{h^2} + \frac{\tilde{\nabla}_i^d \kappa}{\tilde{\kappa}_i^d} \frac{e^{\alpha h} - e^{-\alpha h}}{2h} \right] \geqslant \tilde{\kappa}_i^d \left[G(\alpha) + 1 \right], \quad \forall i \in \{1, \dots, n\}, \tag{2.30}$$

hence the result follows. □

LEMMA 2.10. *Let $\bar{\mathbf{L}}$ be the conservative, second-order finite difference operator defined by* (2.23)*, then $-\bar{\mathbf{L}}$ satisfies the maximum principle.*

Proof. Let us show that $-\bar{\mathbf{L}}$ satisfies all the conditions of Theorem 2.6. First, $-a_{ii} = \sum_{d=1}^{D}(\kappa_i^{d-} + \kappa_i^{d+}) > 0$ for $i \in \{1, \dots, n\}$. Furthermore, for any $i \in \{1, \dots, n\}$ and any $j \neq i \in \{1, \dots, n_{\mathrm{t}}\}$,

$$-\bar{l}_{ij} = \begin{cases} -\kappa_i^{d\pm} \leqslant 0 & \text{if } \mathbf{x}_j = \mathbf{x}_i \pm h^d \mathbf{e}^d,\ d \in \{1, \dots, D\}, \\ 0 & \text{otherwise,} \end{cases} \tag{2.31}$$

which shows that $-\bar{\mathbf{L}}$ satisfies condition (i). Second, it is well-known that well-posed elliptic problems should lead to irreducible finite difference matrices [17, 42]. Here it is easy to see that the directed graph associated to $\mathbf{A}$ is strongly connected, which is equivalent to $\mathbf{A}$ being irreducible [44], and thus conditions (ii) holds. Finally, we observe that $\sum_{j=1}^{n_{\mathrm{t}}} \bar{l}_{ij} = 0$ for any $i \in \{1, \dots, n\}$, and that $\sum_{j=1}^{n} a_{ij} > 0$ whenever $i \in \{1, \dots, n\}$ is such that $\mathbf{x}_i = \mathbf{x}_j \pm h^d \mathbf{e}^d$ where $\mathbf{x}_j$ is a boundary point. Therefore condition (iii) is satisfied as well. □

THEOREM 2.11. *Let $\Omega_h \subset \times_{d=1}^{D}(O^d; O^d + H^d)$. Let $\bar{\mathbf{L}}$ be the conservative, second-order finite difference operator defined by* (2.23)*, and let $\bar{\mathbf{u}} \in \mathbb{R}^{n_{\mathrm{t}}}$ such that $\bar{\mathbf{L}}\bar{\mathbf{u}} = \mathbf{b}$, with $\mathbf{b} \in \mathbb{R}^n$. Then*

$$\begin{cases} \min\limits_{1 \leqslant i \leqslant n} u_i \geqslant \min\limits_{1 \leqslant i \leqslant m} u_{n+i} - C^d \max\limits_{1 \leqslant i \leqslant n} (b_i^+ / \tilde{\kappa}_i^d) \\ \max\limits_{1 \leqslant i \leqslant n} u_i \leqslant \max\limits_{1 \leqslant i \leqslant m} u_{n+i} - C^d \min\limits_{1 \leqslant i \leqslant n} (b_i^- / \tilde{\kappa}_i^d) \end{cases}, \quad \forall d \in \{1, \dots, D\}, \tag{2.32}$$

where $C^d \doteq e^{\alpha H^d} - 1$ with

$$\alpha \geqslant \frac{1}{h^d} \log \left[\frac{h^d \left(\left[h^d\right]^2 + \left[\beta^d\right]^2 + 4 \right)^{1/2} + \left[h^d\right]^2 + 2}{2 - \beta^d h^d} \right]. \tag{2.33}$$

Proof. Let α as in (2.33), then it follows from Lemmas 2.9 and 2.10 that the conditions of Theorem 2.7 are satisfied with $\lambda_i = \tilde{\kappa}_i^d$, for any $i \in \{1, \dots, n\}$ and $d \in \{1, \dots, D\}$. Because $\bar{\mathbf{L}} = \bar{\mathbf{L}}^0$ with $\bar{\mathbf{L}}^0$ satisfying condition (2.6), the inequalities in (2.9) hold. □

2.4.4. Discussion. The expressions in Eq. (2.32) reveal that the bounds depend on four parameters, namely:

- the minimum and maximum of the boundary conditions;
- the diffusivity field (and its gradient);
- the source field;

- the size of the domain, more precisely H^d.

The influence of the boundary conditions directly relates to the homogeneous PDE, for which the maximum principle applies, where only the boundary conditions are involved.

The diffusivity field plays a critical role and appears twice in the bounds. First, it is involved in the constant C^d through β^d. It is clear that, for a given discretization h^d, the function $\alpha \mapsto e^{\alpha H^d} - 1$ is increasing, so the tightest bounds in Theorem 2.11 are obtained for the smallest admissible α, and thus for the constant $\hat{C}^d$ defined as:

$$\hat{C}^d = \left[\frac{h^d \left(\left[h^d\right]^2 + \left[\beta^d\right]^2 + 4 \right)^{1/2} + \left[h^d\right]^2 + 2}{2 - \beta^d h^d} \right]^{H^d/h^d} - 1. \tag{2.34}$$

For a given h^d, α only depends on the diffusivity field (through β), and the smallest admissible value $\hat{\alpha}$ is an increasing function of β. As a consequence, high gradients of κ are detrimental to the sharpness of the bounds. On the contrary, high values of κ may help to decrease β^d and are beneficial to the sharpness of the bounds. On the other hand, it can be shown (using L'Hôpital's rule, for example), that if β^d vanishes regardless of the discretization (*i.e.* the diffusivity is constant), then $\hat{\alpha}$ converges to 1 as h^d tends to 0, and thus $\hat{C}^d$ converges to $e^{H^d} - 1$. In that case, the diffusivity field has no influence on C^d, but still has an influence on the last factor of the bounds. High values of κ are also beneficial to this part of the bounds.

The influence of the source field is twofold. First, if the source field has a constant sign over the domain, then the second term vanishes in either the lower or the upper bound, which then only depends on the boundary conditions. Second, high magnitudes of the source field deteriorate the sharpness of the bounds.

Lastly, the size of the domain appears in the constant C^d (or $\hat{C}^d$). Smaller domains then lead to smaller values of C^d and thus to sharper bounds.

We now illustrate the influence on the bounds of the four parameters mentioned above in light of finite difference solutions of the following 1D diffusion problem:

$$\begin{cases} \dfrac{\partial}{\partial x}\left[\kappa(x)\dfrac{\partial u}{\partial x}(x)\right] = f(x), \quad \forall x \in \Omega \doteq (X^-, X^+), \\ u(X^-) = U^-, \quad u(X^+) = U^+. \end{cases} \tag{2.35}$$

We will denote by $\underline{\mathrm{U}}$ (resp. $\overline{\mathrm{U}}$) the lower (resp. upper) bound obtained from Eq. (2.32) in Theorem 2.11 using the tightest constant $\hat{C} \doteq \hat{C}^1$.

Influence of the PDE parameters. We first investigate the influence of the PDE parameters (other than the boundary conditions) on the bounds. We consider a fixed domain $\Omega = (0, 1)$ and fixed boundary conditions $U^- = 0$, $U^+ = 1$. The diffusivity field, κ, and the source field, f, are defined as follows:

$$\begin{cases} \kappa(x) = \tanh\left[a_\kappa(x - 1/2)\right] + b_\kappa, \\ f(x) = c_f \tanh\left[5(x - 1/2)\right], \end{cases} \tag{2.36}$$

where a_κ, b_κ and c_f are real constants allowing us to tune the properties of κ and f.

In Table 1 we report the constants β and $\hat{C}$, as well as the lower and upper bounds $\underline{\mathrm{U}}$ and $\overline{\mathrm{U}}$ and actual minimum and maximum of the discrete solution, for different values of a_κ, b_κ and c_f. The

a_κ	b_κ	c_f	β	$\hat{C}$	$\underline{\mathrm{U}}$	$\overline{\mathrm{U}}$	$\min_{1\leqslant i\leqslant n} u_i$	$\max_{1\leqslant i\leqslant n} u_i$
0	1	0	0	1.72	0	1	0.001	0.999
0	1	0.1	0	1.72	-0.17	1.17	0.001	0.999
0	1	1	0	1.72	-1.70	2.70	0.001	0.999
0	1	10	0	1.72	-16.95	17.95	0.003	0.997
1	1	1	1.46	6.16	-4.34	12.29	0.002	0.999
2	1	1	3.52	43.06	-25.37	178.57	0.004	0.999
5	1	1	9.93	2.27e4	-1.13e4	1.66e6	0.017	1.049
10	1	1	20.0	5.10e8	-2.51e8	5.43e12	0.547	10.48
10	5	1	2.02	10.38	-1.71	3.56	0.001	0.999
10	10	1	1.0	4.05	-0.36	1.44	0.001	0.999
10	50	1	0.2	2.02	-0.04	1.04	0.001	0.999
10	100	1	0.1	1.86	-0.02	1.02	0.001	0.999

Table 1: Constants β and $\hat{C}$ for different diffusivity and source fields (parameters a_κ, b_κ and c_f), as well as lower and upper bounds ($\underline{\mathrm{U}}$ and $\overline{\mathrm{U}}$) and actual minimum and maximum of the discrete solution. The results were obtained using $U^- = 0$ and $U^+ = 1$ as left and right boundary conditions and $h = 0.001$.

first 4 rows of the table correspond to a constant diffusivity ($\kappa \equiv 1$) and an increasing magnitude of the source term (controlled by c_f). The first row corresponds to $f \equiv 0$ (homogeneous problem), for which we can see that the bounds directly result from the maximum principle and thus correspond to the minimum and maximum boundary conditions. For the next three rows of the block, the bounds linearly grow (in magnitude) as the source term increases. This trend is explained by the fact that $\hat{C}$ is not affected by changes in the source field. Consequently, because $\tilde{\kappa}_i^d$ also remains constant and uniform, the bounds linearly depend on the extrema of the source field.

The next block of 4 rows corresponds to a case where the source field lies between -1 and $+1$, and where the diffusivity field is between 0 and 2 with an increasing gradient, controlled by a_κ. Notice that the exact continuous gradient of κ reaches its maximum at $x = 1/2$, where it equals a_κ. As the gradient becomes steeper, the value of β increases, and so does the value of $\hat{C}$. Consistent with the facts that α is an increasing function of β for a given discretization, and that $\hat{C}$ is increasing exponentially with α for a given domain, we observe that the bounds also increase exponentially in magnitude as a_κ increases. For instance, for a maximum diffusivity gradient of 2 (second row in the block), the upper bound almost reaches 200, while the true solution remains below 1. This makes the diffusivity gradient a critical parameter as regards the sharpness of the bounds.

The last 4 rows correspond to a case where the maximum diffusivity gradient is high, with an increasing minimum diffusivity value (controlled by b_κ). As stated previously, the magnitude of the diffusivity appears twice in the bounds. Increasing b_κ dramatically decreases the value of β (and thus $\hat{C}$), but also decreases the absolute values of $\max_{1\leqslant i\leqslant n}(b_i^+/\tilde{\kappa}_i^d)$ and $\min_{1\leqslant i\leqslant n}(b_i^-/\tilde{\kappa}_i^d)$ in Eq. (2.32), which tightens the bounds. As a matter of fact, for a minimum diffusivity value greater than 4 ($b_k = 5$), the constants β and $\hat{C}$, and thus the bounds, have reasonable values, while with

$b_\kappa = 1$ they reached extremely large values. This confirms that the diffusivity field plays indeed a critical role, with either beneficial or detrimental effects, in the sharpness of the bounds.

Influence of the domain length. We now fix the PDE parameters. Specifically, we consider the fields defined by Eq. (2.36), with $a_\kappa = b_\kappa = c_f = 1$. In order to investigate the influence of the domain size, we vary the length of Ω around $x = 1/2$:

$$\Omega \doteq (1/2 - \delta H, 1/2 + \delta H), \tag{2.37}$$

keeping a constant discretization of $h = 0.001$ regardless of the domain size.

δH	β	$\hat{C}$	$\underline{\mathrm{U}}$	$\overline{\mathrm{U}}$	$\min_{1\leqslant i\leqslant n} u_i$	$\max_{1\leqslant i\leqslant n} u_i$
0.5	1.46	6.16	-4.34	12.29	0.002	0.999
0.1	1.10	0.40	-0.17	1.20	0.006	0.995
0.05	1.05	0.18	-0.04	1.05	0.011	0.990
0.01	1.01	0.03	-0.001	1.002	0.050	0.950
0.005	1.004	0.016	-0	1	0.1	0.9

Table 2: Constants β and $\hat{C}$ for different domain $\Omega \doteq (1/2 - \delta H; 1/2 + \delta H)$, as well as lower and upper bounds ($\underline{\mathrm{U}}$ and $\overline{\mathrm{U}}$) and actual minimum and maximum of the discrete solution. The results were obtained using $U^- = 0$ and $U^+ = 1$ as left and right boundary conditions and $h = 0.001$.

Table 2 reports the constants β and $\hat{C}$, as well as the lower and upper bounds $\underline{\mathrm{U}}$ and $\overline{\mathrm{U}}$ and actual minimum and maximum of the discrete solution, for different values of δH. The first row ($\delta H = 0.5$) corresponds to the first row of the second block in Table 1 for $\Omega = (0, 1)$. When the size of the domain is divided by 5 (second row, $\delta H = 0.1$) the value of $\hat{C}$ is reduced 15 folds, from 6.16 to 0.4. This significant drop is due to the exponential dependence of $\hat{C}$ on the size of the domain. To a smaller extent, the constant $\hat{C}$ is also influenced by the slight decrease of β, which is due to the minimum diffusivity increase when the domain is shrunk. As the domain size decreases, $\hat{C}$ keeps decreasing and thus the bounds get sharper. However, because we keep $h = 0.001$ constant, the minimum and maximum values of the solution on the grid tend to move away from the boundary conditions $U^- = 0$ and $U^+ = 1$.

3. Application to the resilient domain decomposition solver. In the context of exascale computing, massively parallel jobs will be subject to system faults with significantly higher rates than those in today's petascale systems [40]. As a consequence, PDE solves, among other operations, may return erroneous results. The bounds derived in the previous section can then naturally be used to detect such faulty solves. Depending on the magnitude of the error caused by the faults and on the sharpness of the bounds, checking that a PDE solution lies between these bounds may not be sufficient to overcome the occurrence of faults. Building on an existing domain decomposition approach for solving elliptic PDEs in a resilient fashion [33], we shall see how complementing it by checking bounds improves its resilience capabilities.

3.1. Resilient domain decomposition solver. Domain decomposition methods consist in dividing the domain over which the problem needs to be solved into subdomains on which one solves smaller, and thus cheaper, subproblems. In the resilient approach developed in [33, 32]

and summarized in Fig. 1, the domain Ω is split into N *overlapping* subdomains $\{\Omega_i\}_{i=1}^N$ with corresponding boundaries $\{\partial\Omega_i\}_{i=1}^N$ such that $\cup_{i=1}^N \bar{\Omega}_i = \bar{\Omega}$, where $\bar{\cdot}$ denotes the closure of a set. In such a decomposition, we assume that for each Ω_i there is at least one overlapping neighbor Ω_j (with $j \neq i$), *i.e.* such that $\Omega_i \cap \Omega_j \neq \varnothing$. The algorithm amounts to finding, for each subdomain Ω_i, the Dirichlet-to-Dirichlet maps from the boundary conditions on $\partial\Omega_i$ to the overlapping boundary values of its neighboring subdomains.

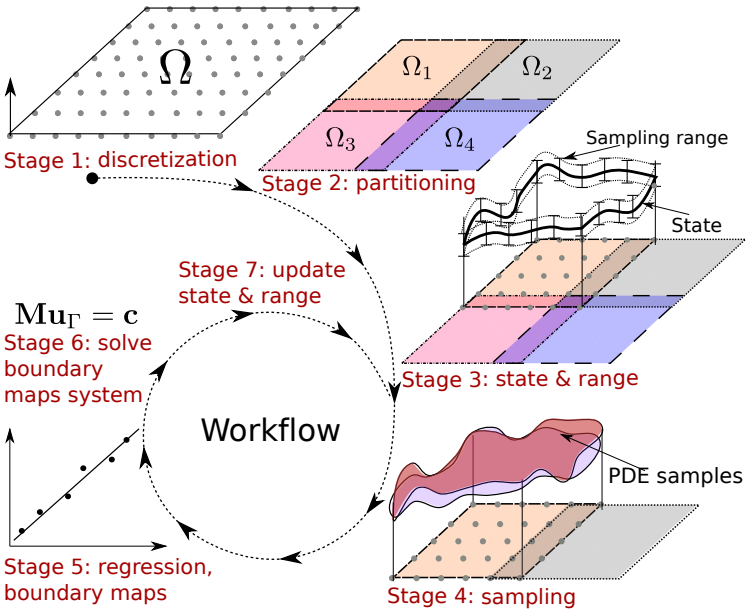


Figure 1: Schematic of the workflow of the resilient domain decomposition solver described in [33, 32], reproduced from [31].

Specifically, on each subdomain Ω_i, boundary conditions are randomly sampled and the local subproblem is solved for each sample (stage 4 of Fig. 1). Regression is then used to infer the relationship between the boundary conditions on Ω_i and the solution at points on the boundaries of its neighboring subdomains (stage 5 of Fig. 1). Using these maps, we enforce compatibility conditions between the subdomains, which, in the case of a linear PDE, take the form of a system of linear equations. The boundary values obtained from the solution of this system correspond to the solution of the global Dirichlet problem on Ω at the subdomains boundaries (stage 6 of Fig. 1). More details on this approach can be found in Appendix A, as well as in [33, 32].

In this framework, resilience is achieved through sampling and robust regression techniques. Because *least squares* (LS) models are very sensitive to outliers (see [33] for examples in the context of bit-flips), we instead resort to *least absolute deviation* (LAD) models, which are known to be much more robust [29]. Solving LAD regression problems amounts to minimizing the L^1 norm of the residual vector. In a Bayesian context, this amounts to using a uniform prior together with a Laplace likelihood [33]. These types of techniques are widely used in compressed sensing problems [7], as it is known that the L^1 norm is a good approximation of the L^0 "norm" [6], which returns the number of non-zero entries in a vector. In practice, we solve the LAD regression problem

using the *iteratively reweighted least squares* algorithm [28, 35, 12, 34, 9, 14].

In the case of a linear PDE, the boundary-to-boundary maps could theoretically be determined by solving exactly as many independent local PDEs as the number, $N_i^\partial + 1$, of unknown affine coefficients (*i.e.* the number of unknown boundary points plus one) on each subdomain Ω_i. However, because of the potential occurrence of faults, we allow ourselves to slightly oversample the boundary conditions by a factor $\rho^* > 1$. In other words, on a subdomain Ω_i with N_i^∂ unknown boundary points, we define a target number of samples $s_i^* = \rho^*(N_i^\partial + 1)$ to be used for the regression problems on subdomain Ω_i. We will refer to ρ^* as the nominal sampling rate. In fact, the nominal sampling rate fixes the amount of extra work required to ensure the resilience of the approach. Indeed ρ^* should be increased from 1 as the expected fault rate increases, and we shall see that the verification of the solutions with the bounds allows to reduce the value of ρ^* (and so the overhead) compared to the original approach in [33].

The resilient solver was validated on 1D [33] and 2D [32, 31] elliptic problems. In the latest campaign of runs, tests up to about 110,000 cores showed excellent scalability. Specifically a parallel efficiency higher than 90% was reported for both weak and strong scaling experiments [31].

3.2. Implementation details and fault model. The implementation of the approach is based on a server-client framework, relying on the *message passing interface* (MPI). In such a framework, the MPI processes are grouped into servers and clients. The servers are assumed to be fault-free units holding the data, whereas the clients are designed solely to accept and perform work without any assumption on their reliability. The work to be performed on the clients takes the form of tasks that are created on the server, sent to the clients where the computational work is performed, and sent back to the server where the result of the task is handled. The server-client framework is described in detail in [32] and outlined in Appendix B. Typically, given the size of the problems considered in this paper, the MPI processes are split into one or two servers and a few tens of clients. For exascale computations, the framework may for instance involve more than 5,000 servers, managing 1,000 clients each. Our algorithm involves two main types of tasks: sampling and regression. The sampling tasks are given a sample of the boundary conditions and their computational work consists in solving the PDE for these boundary conditions. Each sampling task corresponds to one single sample of the boundary conditions on one subdomain. The regression tasks use the samples returned by the sampling task (collected on the servers) to infer the boundary-to-boundary maps. The final solve of the compatibility system is performed on the servers and is thus considered fault-free. The fault-free model assumed for the servers can be supported by either hardware or software, as detailed in Appendix B.

By assumption, soft errors are introduced on the clients. They can corrupt either the transmission of the incoming task (from the server to the client), or the computational work performed on the client, or the transmission of the returning task (from the client back to the server). In the context of the present study, in an effort to better investigate the effect on resilience of detecting corrupted PDE solutions using the previously derived bounds, we shall only corrupt returning sampling tasks. Data corruption is simulated by bit flips in the 64-bit binary representation (see the IEEE 754 Standard [1]) of double precision values. Flipping a bit simply consists in altering its value, *i.e.* changing it to 1 if it was originally a 0, and *vice versa*. A memoryless Poisson process was chosen as a means to introduce such errors. A Poisson process is uniquely defined by a single parameter, namely the rate of failure r, leading to a failure time distribution

$$F(t) = \int_0^t r \exp(-r\tau) d\tau = 1 - \exp(-rt). \tag{3.1}$$

From an implementation standpoint, to simulate the occurrence of a fault for a target operation we proceed as follows. For a given failure rate, we draw a sample from a standard uniform random number, and extract from the corresponding failure distribution $F(t)$ the amount of time until the next fault occurs. We then measure the execution time for the target operation to complete, and if that time exceeds the next failure time, then a fault is triggered. The reader may refer to [32] for more details on the fault model.

Once the fault is triggered, we proceed to simulate the effect of the fault using bit flips. We investigate two different effects of a fault occurring during communication. In the first scenario, we consider that one single variable is affected within the data packet that is being transmitted. In the second scenario, we consider that the whole data packet gets corrupted, resulting in the corruption of all the variables being transmitted. The data packet corresponding to the returning sampling task, performed on a 2D subdomain, basically contains two types of information. First, it contains the boundary conditions, at each boundary point of the subdomain, for which the task was performed, *i.e.* for which the local PDE was solved on this subdomain. These *input* variables will subsequently be used to feed the so-called design matrix for the regression. Second, the data packet contains the corresponding local solution, evaluated only at the inner points of interest, *i.e.* at points that belong to the boundaries of neighboring subdomains. These *output* variables will be used to feed the so-called response vector for the regression.

In principle, any (in the first corruption scenario) or all (in the second corruption scenario) of the variables in the data packet should be affected whenever a fault is triggered. However, in order to ease the analysis of the results, the corruption of the returning boundary conditions will not be considered in this study. In addition, in both scenarios our model assumes that any given double precision value may only be corrupted by one single bit flip. In other words, at most one of the 64 bits representing the same variable may be flipped.

When a (possibly corrupted) sampling task is returned from the client to the server, the latter checks the admissibility of the PDE solution in terms of the bounds. If the solution does not lie between the bounds, it is detected as corrupted. The detected samples are simply discarded, and the rest of the algorithm, in particular the regression stage, proceeds with a degraded number of samples $s_i < s_i^*$.

For a given PDE (with given diffusivity and source fields) and a fixed discretization, most of the terms in the expression of the bounds (see Eq. (2.32)) can be computed once and for all by the servers at the beginning of the computations. In fact, only the part corresponding to the homogeneous part (*i.e.* resulting from the maximum principle) depends on the boundary conditions. Therefore, the invariant terms are computed by the servers for each subdomain in a pre-processing stage, so that the servers simply need to add the contribution of the boundary conditions whenever they generate samples of the boundary conditions. When a sampling task returns, the servers can then use the bounds to determine whether the minimum and maximum values of solution that is being returned are admissible. Note that because the bounds were computed on the server from the uncorrupted boundary conditions, potential corruptions of the boundary conditions in the returning task, not considered here, would not significantly affect the resulting admissibility of that task. The sampling stage (stage 4 of Fig. 1), including the computation of the bounds and the admissibility check of the PDE solutions, is summarized in Algorithm 1.

It should be highlighted that there are as many (lower and upper) bounds as dimensions of space. As a result, during the pre-processing stage, one can choose the dimension that yields the tighter bounds. It is also worth mentioning that the terms appearing in the bounds are relatively cheap to compute on each subdomain. Specifically, their evaluation basically amounts to $\mathcal{O}(n)$

Algorithm 1: Simplified algorithm of the sampling stage (stage 4 of Fig. 1), including the computation of the bounds and the admissibility check of the PDE samples.

foreach *subdomain* Ω_i **do**
  // [SERVER] Pre-processing stage
  Compute the invariant parts of the bounds for Ω_i ;
  Choose dimension $d \in \{1, \dots, D\}$ that leads to the tightest bounds ;
  // [SERVER] Sample boundary conditions
  Sample s_i^* boundary conditions for Ω_i ;
  $s_i \leftarrow s_i^*$;
  foreach *sample* **do**
    // [SERVER]
    Add contribution of the boundary conditions to the bounds ;
    Send task to a client ;
    // [CLIENT]
    Receive task from server ;
    Solve the local PDE in Ω_i using the received sample of boundary conditions ;
    Send task (with the solution) back to server ; /* Corruption may occur here */
    // [SERVER]
    Receive returning task from client ; /* Task is potentially corrupted */
    if *received solution does not lie between the bounds* **then**
      Discard current sample ;
      $s_i \leftarrow s_i - 1$;
    end if
  end foreach
end foreach

operations, where n denotes the number of unknown points in the subdomain. The cost then naturally decreases with the size of the subdomains, which is in accordance with the reduction of complexity expected from a domain-decomposition approach.

3.3. Fault detection. We now illustrate the effect on resilience of checking the bounds with 2D numerical experiments. The following 2D diffusion equation will be solved using a conservative second-order FD scheme:

$$\begin{cases} \boldsymbol{\nabla} \cdot [\kappa(\mathbf{x}) \boldsymbol{\nabla} u(\mathbf{x})] = f(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega = (0,1)^2, \\ u|_{\partial\Omega} = 1. \end{cases} \tag{3.2}$$

The boundary conditions on the subdomains will be sampled in a range of length one around the global boundary conditions. Specifically, they will be sampled uniformly on $[0.5, 1.5]$.

We investigate three different problems, corresponding to three different definitions of the diffusivity and source fields, defined as follows.

Case A. This case will be our reference case. It involves a uniform diffusivity field and a source field that changes sign with a steep, localized gradient:

$$\begin{cases} \kappa(\mathbf{x}) = 1, \\ f(\mathbf{x}) = \tanh[d(\mathbf{x})/0.05], \end{cases} \tag{3.3}$$

with $d(\mathbf{x}) = 0.25 - \|\mathbf{x} - \mathbf{x}_0\|$, where $\mathbf{x}_0 = (0.5, 0.5)$ denotes the center of the full domain.

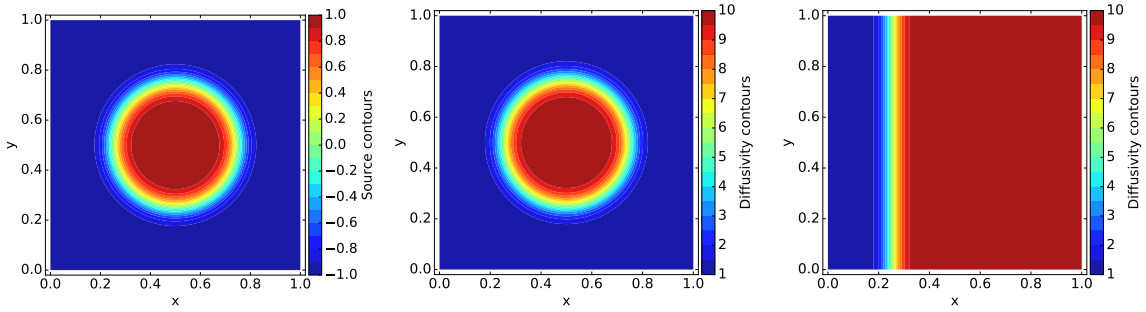
Case B. This case involves a uniform source field and a diffusivity field with a steep, localized gradient:

$$\begin{cases} \kappa(\mathbf{x}) = 4.5 \tanh[d(\mathbf{x})/0.05] + 5.5, \\ f(\mathbf{x}) = 1. \end{cases} \tag{3.4}$$

Case C. This case involves a uniform source field and a diffusivity field with a steep, localized gradient. The difference with case B is that here the gradient is unidirectional:

$$\begin{cases} \kappa(\mathbf{x}) = \kappa(x, y) = 4.5 \tanh[(x - 0.25)/0.05] + 5.5, \\ f(\mathbf{x}) = 1. \end{cases} \tag{3.5}$$

The contours of the (non-uniform) fields involved are plotted in Fig. 2.



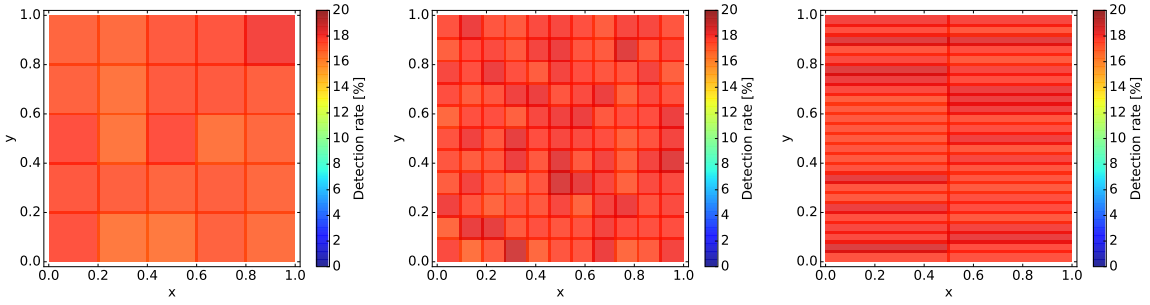
(a) Contours of f for case A. (b) Contours of κ for case B. (c) Contours of κ for case C.

Figure 2: Contours of the source and diffusivity fields for cases A, B and C.

We consider a fixed discretization of 200 finite difference cells in each direction, resulting in a mesh size $h = h_x = h_y = 0.005$. Five different partitions of the domain into subdomains will be used hereafter: 5×5, 11×11, 25×25, 2×25 and 25×2. The subdomains overlap by 2 cells in each direction for all partitions. The overlapping regions are visible on Fig. 3 to Fig. 7 as (narrow) darker stripes between the subdomains.

To better understand how the bounds can be useful on an actual problem, we first investigate their ability to locally detect corrupted samples on subdomains. To do so, we proceed to the sampling stage with a fixed target number of samples $s^* = s_i^* = 10,000$ on every subdomain Ω_i, and choose to corrupt every corresponding returning task. Each returned task, *i.e.* each corrupted PDE solution, is checked for admissibility in terms of the bounds. Samples that are thus detected

as corrupted are discarded, so that we eventually collect an actual number of samples $s_i \leqslant s^*$ on each subdomain. We then report the detection rate defined as $d_i \doteq 1 - s_i/s^*$ for each subdomain Ω_i. The other stages (regression and solve of the final system) are not considered. We investigate the two corruption strategies mentioned in Section 3.2. In the first scenario, only one variable is corrupted by a bit flip, while in the second scenario the whole data packet is corrupted.



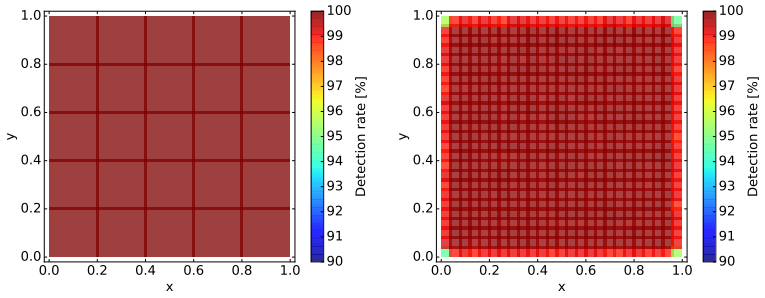
(a) 5×5 partition. (b) 11×11 partition. (c) 2×25 partition.

Figure 3: Detection rates for case A and different partitions. One single data point is corrupted during transmission.

Figure 3 depicts the detection rate on each subdomain for case A, with one single variable being corrupted. Regardless of the partitioning, the detection rate is relatively uniform over the subdomains, and lies between 16% and 20%. This means that the corruption of one (double precision) float can be detected by the bounds between 16% and 20% of the time. The average detection rate is slightly higher in Fig. 3b and 3c is slightly higher than it is in Fig. 3a. This can be explained by the fact that smaller subdomains lead to tighter bounds (see discussion in Section 2.4.4).

It is worth mentioning at this point that in the IEEE 754 standard [1], double precision variables are stored using 64 bits, one of which being the sign bit, 11 corresponding the exponent bits, and the remaining 52 corresponding to the mantissa bits. Flipping the sign bit naturally corresponds to changing the sign of the real number. Flipping mantissa bits result in the alteration of the decimal significant digits of the real number, *i.e.* its fractional part. Lastly, exponent bits are responsible for the integer part of the real number. Therefore, it is clear that the most noticeable bit flips are those affecting either the sign bit or the exponent bits. Notice that $12/64 \approx 19\%$, so it is fair to assume that the bit flips that are detected here affected the sign and exponent bits.

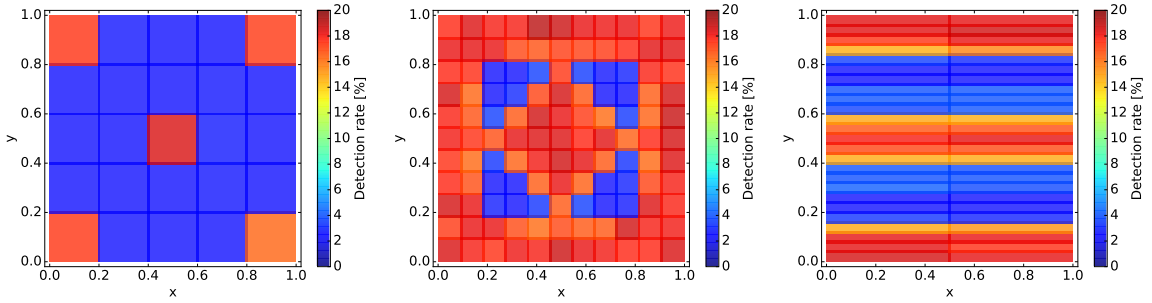
Figure 4 shows the detection rates for the same case, but in the second corruption scenario where the whole data packet is corrupted. In the case of a 5×5 partition depicted in Fig. 4, all the subdomains have a 100% detection rate. This is due to the fact that, on a given subdomain, the corrupted solution at each inner point of interest has about a 16% chance of being detected. Interior subdomains, *i.e.* subdomains that do not share a boundary with the global domain, have about 150 inner points of interest. Then the probability of detecting one of the corrupted values (typically, the minimum and/or the maximum value) is close to 100%. In the case of a 25×25 partition (Fig. 4), the interior subdomains still show detection rates close to 100%. Such subdomains have 24 inner points of interest. For "boundary subdomains" that share one boundary with the global



(a) 5×5 partition. (b) 25×25 partition.

Figure 4: Detection rates for case A and different partitions. The whole data packet is corrupted during transmission.

subdomain, the detection rate is slightly lower, around 98%. This can be explained by the fact that such subdomains have fewer inner points of interest, since they have only three overlapping neighbors. This drop in the detection rate is more pronounced in the corners, for subdomains sharing two boundaries with the global domain, and thus having only two overlapping neighbors. Such subdomain only have 13 inner points of interest, but still show a detection rate of about 95%.



(a) 5×5 partition. (b) 11×11 partition. (c) 2×25 partition.

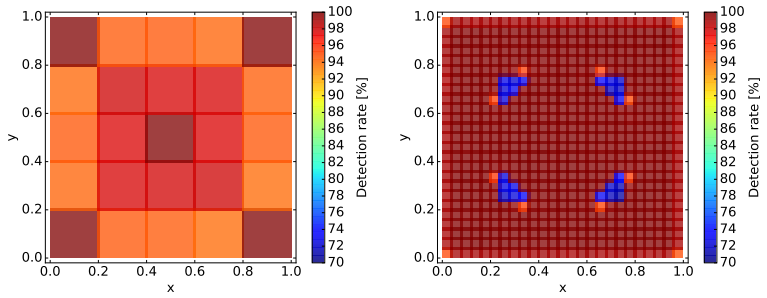
Figure 5: Detection rates for case B and different partitions. One single data point is corrupted during transmission.

Let us now focus on case B, where the diffusivity field has a sharp radial gradient. Figure 5 depicts the case where a single data point is corrupted. For the 5×5 partition (Fig. 5a), the corner subdomains and center subdomain show a detection rate close to 18%, similarly to case A, while the other subdomains show a poor detection rate of about 2%. This significant drop in the detection rate is explained by the diffusivity gradient, which deteriorates the sharpness of the bounds, as discussed previously in Section 2.4.4. Indeed, those subdomains with a poor detection rate correspond to subdomains where the diffusivity gradient does not vanish. Notice that even

the boundary subdomains are affected, although the diffusivity gradient is mild in these regions. This may be caused by the fact that the alteration of any exponent bit can often drive the variable very close to zero. As a consequence, because of the diffusivity gradient, the lower bound may become negative, and the admissibility check would then fail to detect the corruption caused by such exponent bit flips. It should be stressed that this is closely related to the choice of modeling fault by bit flips. We believe that other fault models would most likely lead to different detection rates on those particular subdomains.

The 11×11 partition (Fig. 5b) shows a consistent behavior. On the subdomains where the diffusivity is almost constant, the detection rates are between 18% and 20%, while most of the other subdomains show a low detection rate of about 2%. One exception is for the subdomains at the center of the domain in each dimension. For such subdomains, the detection rate is always greater than 18%, even in areas where the diffusivity gradient is high. In such cases, the gradient is actually mostly 1D, so that we can choose one direction d in which the derivative $\tilde{\nabla}_i^d \kappa$ (see Section 2.4.3) almost vanishes, leading to tighter bounds. Another observation is that in regions where the diffusivity gradient was mild but high enough to cause the detection rate to drop to 2% for the 5×5 partition, which correspond to the second "ring" of subdomains starting from the boundaries in the 11×11 partition, the detection rates remain above 18%. This can be explained by the fact that the subdomains are smaller, and consequently the bounds are tighter and the lower bound remains positive.

The 2×25 partition (Fig. 5c) confirms the observations made previously. It emphasizes the fact that even if the diffusivity gradient is localized, it still penalizes the whole subdomain. A good partitioning strategy therefore consists in designing a partition that best isolates the high gradient areas in a limited number of subdomains. The smaller and the more uniform the subdomains are, and the more isolated the high gradient regions will be, which is consistent with the domain decomposition paradigm.

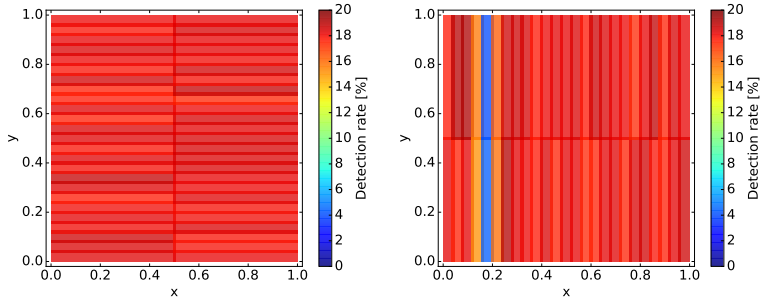


(a) 5×5 partition. (b) 25×25 partition.

Figure 6: Detection rates for case B and different partitions. The whole data packet is corrupted during transmission.

We now investigate the effect of corrupting the whole data packet on case B, illustrated in Fig. 6. For the 5×5 partition (Fig. 6a), the corner and the center subdomains show a 100% detection rate. This is consistent with the fact that they correspond to subdomains where the diffusivity is nearly constant. For the other subdomains, the detection rate is slightly lower. In

the boundary subdomains, the detection rate is about 90%, which is due to the fact that there is a small diffusivity gradient that makes the lower bound negative, as discussed earlier. The detection rate is lower than the other subdomains with high gradient diffusivity because they have fewer inner points. Concerning the 25×25 partition (Fig. 6b), the deterioration of the detection rate due to the diffusivity gradient is better isolated in specific subdomains, but the effect is also more pronounced, because the subdomains have fewer inner points. On such subdomains, the detection rates drop to approximately 70%. The fact that the poor detection areas can be localized in a limited number of subdomains allows one to adapt the resilient strategies from one subdomain to another. One may for instance increase the nominal sampling rate ρ^* in those subdomains where the bounds are expected to be less tight.



(a) 2×25 partition. (b) 25×2 partition.

Figure 7: Detection rates for case C and different partitions. One single data point is corrupted during transmission.

The detection rates for case C when a single data point is corrupted are reported on Fig. 7. For the 2×25 partition (Fig. 7a), the detection rate is uniform over the subdomains and is between 18% and 20%. Again, this is due to the fact that the diffusivity gradient is unidirectional. Specifically, the y-component of the diffusivity gradient vanishes everywhere. In addition, the subdomains are smaller in the y direction than in the x direction, which benefits the sharpness of the bounds. On the other hand, for the 25×2 partition (Fig. 7b), there is a vertical stripe of subdomains for which the detection rate drops to about 4%. These subdomain correspond to a region where the diffusivity and its gradient are such that the lower bound can become negative. For the reasons discussed previously, because of the inherent behavior of the bit flips, this leads to a drop in the detection rate. The case where the whole data packet is corrupted, not shown here, leads to a 100% detection rate on all the subdomains, for both partition (2×25 and 25×2).

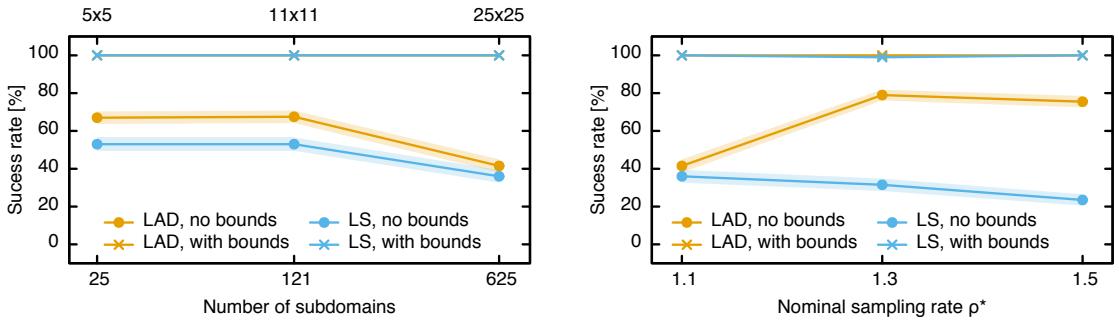
3.4. Improvement of resilience. Let us now investigate how detecting faulty samples benefits the resilience of our algorithm. As mentioned previously, our approach consists in using robust regression techniques (see [33]) to achieve resilience. However, such regression problems are more expensive to solve than classical regression problems such as ordinary least squares. To asses the gain obtained by using the bounds in terms of resilience, we now perform a complete solve of problem (3.2) using our resilient domain decomposition algorithm. At the end of the solve, we check the residual on the boundaries by computing the solution on each subdomain using the boundary conditions obtained from the final solve of the boundary-to-boundary compatibility equations (see

algorithm description in Section 3.1 and [33]). If the residual is below a certain threshold, set here to 10^{-6}, we consider that the solve is a success, otherwise, we consider it a failure. We repeat the experiment 200 times and we report the success rate, that is the proportion of experiments for which the solve is successful. In the following experiments, the fault rate is set to $r = 0.2$. Note that this rate was set to an exaggeratedly high value to be able to observe the effect of faults. It should be stressed that the fault rates are expected to be much lower on future exascale machines. In addition, in what follows, we only focus on case A, and consider the most realistic corruption scenario, that is the one where the whole data packet is corrupted during communication.

To compare the resilient capabilities of the regression, we consider two minimization problems, namely *least squares* (LS) and *least absolute deviations* (LAD). The former is known for not being robust to outliers and is therefore not expected to be resilient to faults. The latter is expected to disregard outliers and is therefore expected to perform better in the presence of faults. The LAD problem is solved using the *iteratively reweighted least squares* (IRLS) algorithm [9, 14], which consists in solving a sequence of weighted least squares problem. As such, it is clear that the IRLS is at least as expensive as a single LS solve. To examine the potential improvement of using the bounds, we compare the success rates over the 200 experiments using either LS or LAD, with or without the use of the bounds.

Figure 8a shows the success rate as a function of the number of subdomains, corresponding to partitions 5×5, 11×11 and 25×25, for the four different regression and bound scenarios described above, and using $\rho^* = 1.1$ as the nominal sampling rate. The shaded areas correspond to the standard deviation of the estimator of the success rate, assuming a binomial distribution of the number of successes. When the bounds are not used, the success rate is always below 70%, regardless of the regression technique and of the number of subdomains. For both regression techniques, the success rate remains nearly the same from 25 to 121 subdomains, and then drops for 625 subdomains. This is due to the fact that in the latter case, there are many more samples to be generated and transmitted, so that the average number of faults occurring during one complete solve (*i.e.* one realization of the experiment) is significantly higher. In fact the average numbers of faults per experiment are approximately 0.63 with 25 subdomains, 0.77 with 121 subdomains and 1.1 with 625 subdomains. As expected, because of the robust properties of LAD regression, the success rate is higher when using LAD than when using LS. When using the bounds, both regression techniques lead to a 100% success rate. This is in accordance with the nearly 100% detection rates that were obtained in the corresponding cases (see Section 3.3).

Figure 8b reports the success rate as a function of the nominal sampling rate ρ^* obtained with a 25×25 partition. When using the bounds, increasing the sampling rate does not deteriorate the 100% success rate obtained with the smallest sampling rate. However, the sampling rate has an effect on the success rate when the bounds are not used. Specifically, when using LS regression, the success rate decreases as the sampling rate increases. This can be explained by the fact that increasing the sampling rate increases the total number of faulty observations, from which LS regression is not able to recover. On the contrary, the success rate increases with the sampling rate when using LAD regression, which is due to the fact that even if the number of faulty samples increases, their proportion remains relatively constant. Because there are also more uncorrupted observations and owing to the robust properties of LAD regression, the resilience improves as the sampling rate increases. The plateau reached by the success rate around 80% for $\rho^* \geqslant 1.3$ is explained by occurrence of bit flips that lead to corrupted value close to the numerical limit of double precision numbers, around $\pm 10^{308}$. In such cases, the LS solve may fail, returning a NaN (not a number) or an infinite number. Then the first iteration of the IRLS algorithm fails as well,



(a) $\rho^* = 1.1$.

(b) 25×25 partition.

Figure 8: Success rates for case A and different regression problems, with and without the bounds. The whole data packet is corrupted during transmission. Fig. 8a: success rate as a function of the number of subdomains. Fig. 8b: success rate as a function of the nominal sampling rate. In both figures, the curves corresponding to the use of the bounds (LAD and LS with bounds) almost exactly overlap.

compromising the whole regression solve. These results confirms that checking the admissibility of local PDE solutions reinforces the resilience of the overall domain decomposition solver. They indicate that LS could be used instead of LAD in the cases considered here. However, solving the LAD problem with IRLS consists in solving a sequence of LS problems. As such, whenever the LS solve recovers the true solution, the IRLS would converge in one iteration, thus causing no overhead. Using IRLS remains safer, in case the bounds miss one faulty data or if faults are introduced in the regression task.

3.5. Improvement of computational efficiency. We have shown that checking the admissibility of the local PDE solutions using the bounds improves the resilience of the whole solver. We shall now examine the consequences on the computational efficiency of the approach. As stated previously, the solver may actually perform more than one iteration, each of which leading to an approximation of the solution at the subdomain interfaces. At the end of an iteration, the convergence is checked comparing the residual at the interfaces to a given tolerance value. If convergence is not achieved, in other words if the resilient solver is not successful, a new iteration is performed, consisting of a new resilient solve including the sampling, the regression, and the final solve of the compatibility equations (stages 4, 5 and 6 of Fig. 1, respectively). Clearly, in the absence of faults, the algorithm should converge in one iteration, as long as the regression problem remains overdetermined (*i.e.* it involves more independent linear equations than unknowns).

Figure 9 reports the number of iterations needed to achieve convergence as a function of the fault rate r (see Section 3.2), with and without the use of the bounds. The LAD regression problem is considered in both cases, and is solved using IRLS. For the range of fault rates considered here, the plots confirm that using the bounds allows to achieve convergence in one iteration in most cases, while not using the bounds can lead to several iterations. For the highest fault rate $r = 0.2$, the solver needs on average 2 iterations to converge when not using bounds, with a large variance. Considering that each iteration takes about the same time to complete, this means that the resilient

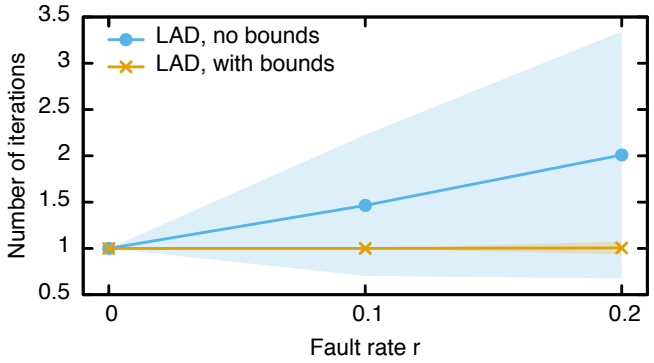


Figure 9: Number of iterations as a function of the fault rate r for case A, with and without the bounds. The LAD regression problem is solved, using a nominal sampling rate $\rho^* = 1.1$. The whole data packet is corrupted during transmission. Shaded areas correspond to one standard deviation below and above the mean value.

solve takes on average twice as much time as when using the bounds. The large variance indicates that there are cases where the solve can take even more time. To be fair, it should be mentioned that a more efficient strategy may be implemented whenever several iterations need to be performed. Specifically, subdomains on which all regression solves lead to small enough regression residuals, meaning that no faults occurred or that all the faults were filtered out by the bounds, can be labeled as "safe". The corresponding boundary-to-boundary maps are then deemed to be safe as well, so that they need not be computed during subsequent iterations. Nonetheless, using the bounds would still reduce the failure rate and thus improve the overall computational efficiency of the solver.

4. Conclusions. In this paper we derived *a priori* bounds for the discrete solution of second-order, elliptic partial differential equations. The bounds have two contributions. The first contribution comes from the boundary conditions, through the discrete maximum principle, previously derived in the literature [11]. The second contribution, involving the PDE coefficients for non-homogeneous problems, was obtained building on an existing derivation in the continuous case [20].

The discrete bounds are general enough to apply to discretized problems of the form $\mathbf{Au} = \mathbf{b} - \tilde{\mathbf{A}}\mathbf{g}$. They have the advantage of being cheap and easy to compute, only requiring $\mathcal{O}(n_i)$ operations on each subdomain Ω_i, where n_i denotes the number of unknowns in Ω_i. However, the conditions under which they apply depend on both the problem and the numerical scheme. We derived such conditions for the widely-used conservative, second-order finite difference approximation of the diffusion equation with variable scalar diffusivity. The general approach can be adapted for other elliptic problems, *e.g.* diffusion problems with tensor diffusion coefficient or reaction-diffusion problems, as well as for other numerical schemes such as fourth-order finite differences.

We applied the presently derived bounds to an existing resilient domain decomposition solver developed in [33]. Our test case was a 2D diffusion equation with variable scalar diffusivity, solved using a conservative, second-order finite difference scheme. We showed that the bounds are affected by the different parameters of the PDE, in particular by the diffusivity gradient. With careful partitioning, we provided an example showing that it is possible to contain regions where the

bounds are looser in a limited number of subdomains, for which the resilient strategy may therefore be strengthened. Our test showed that the bounds were able to detect most of the faulty solutions in the case where the whole data packet was corrupted during transmission. This led to a nearly perfect success rate in the global solution, even when using non-robust regression techniques, namely ordinary least squares. On the other hand, without using the bounds, the success rate remained below 100% even when solving robust regression problems, namely least absolute deviations, and even when increasing the number of samples. The gain in resilience translated into a gain in computational time in the context of our strategy to run new iterations of the overall algorithm until resilience is achieved.

Ongoing work concerns the application of the bounds to uncertain elliptic problems. Recently, we developed an approach to tackle elliptic problems with parametric uncertainty, in which stochastic quantities are approximated by truncated spectral expansions [27]. Because of the truncation error, the approximated boundary-to-boundary maps are not supposed to be exactly affine in the boundary conditions, which results in the need of much higher sampling rates for the robust regression to overcome faulty samples. The gain in resilience provided by the bounds in the deterministic case should be all the more prominent in uncertain problems. Another important improvement of the current resilient approach concerns the treatment of problems with large numbers of subdomains. Indeed, it is known that the condition number of the Dirichlet-to-Dirichlet operator increases with the number of subdomains, calling for appropriate treatments to ensure scalability with respect to the number of subdomains. Along these directions, hierarchical and multilevel extensions appear as natural strategies and, in this context, bounds verification is expected to play an important role in limiting computational overheads.

Finally, we plan to investigate different fault models, both in terms of the occurrence, which is currently simulated using a Poisson process, and of the way data is corrupted, which is currently mimicked by the introduction of bit flips. New occurrence models could involve more complex processes, relying for instance on gamma or Weibull distributions [36, 40].

Acknowledgments. This work was supported by the US Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Award Number DE-SC0010540. 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. The authors are also grateful to Dr. Habib Najm and Prof. Diogo Gomes for helpful discussions.

Appendix A. Details on the resilient domain decomposition solver.

We consider a Dirichlet problem of the form (2.3) in the global domain Ω, and we focus on the solution $u|_\Gamma$ to this problem at the boundaries of the inner boundaries (*i.e.* excluding the boundary $\partial\Omega$ of the global domain Ω) of the subdomains, defined by $\Gamma \doteq (\cup_{i=1}^N \partial\Omega_i) \cap \Omega = (\cup_{i=1}^N \partial\Omega_i) \backslash \partial\Omega$. Because of the overlapping decomposition, each subdomain Ω_i includes a set $\Gamma_i \doteq (\cup_{j=1}^N \partial\Omega_j) \cap \Omega_i$ of boundary parts belonging to neighboring subdomains.

Our algorithm, which can be seen as an accelerated overlapping Schwarz algorithm, relies on the functions $\mathcal{F}_i$ that map, on each subdomain Ω_i, the local Dirichlet boundary conditions $v_i|_{\partial\Omega_i}$ to the solution $v_i|_{\Gamma_i}$, at the neighboring interfaces Γ_i, of the local Dirichlet problem defined by:

$$\begin{cases} \mathcal{L}v_i(\mathbf{x}) = f(\mathbf{x}) & \forall \mathbf{x} \in \Omega_i \\ v_i(\mathbf{x}) = g_i(\mathbf{x}) & \forall \mathbf{x} \in \partial\Omega_i. \end{cases} \tag{A.1}$$

We thus have:

$$v_i|_{\Gamma_i} = \mathcal{F}_i\left(v_i|_{\partial\Omega_i}\right), \quad \forall i \in \{1, \ldots, N\}. \tag{A.2}$$

Combining these maps for all subdomains, together with compatibility conditions at the subdomains, namely $v_i|_{\partial\Omega_j \cap \Omega_i} = v_j|_{\partial\Omega_j \cap \Omega_i}$, eventually yields a fixed-point problem of the form $v|_\Gamma = \boldsymbol{\mathcal{F}} v|_\Gamma$. Provided that the global problem is well-posed, the solution $v|_\Gamma$ of this reduced problem uniquely matches the solution $u|_\Gamma$ of the full problem (2.3) at the inner boundaries. The discretized version involves a reduced system of linear equations $\mathbf{M}\mathbf{u}_\Gamma = \mathbf{c}$, where $\mathbf{u}_\Gamma$ denotes the vector of solution values at discretization points along Γ.

To build the maps, we rely on a sampling strategy that involves solving the target PDE equation locally within each subdomain for sampled values of the boundary conditions on that subdomain (stage 4 of Fig. 1). We then resort to robust regression techniques to find resilient surrogates of the maps from potentially corrupted samples (stage 4 of Fig. 1). Finally, the reduced problem $\mathbf{M}\mathbf{u}_\Gamma = \mathbf{c}$ is assembled and solved (stage 5 of Fig. 1). One of the important features of the algorithm is that the construction of the boundary-to-boundary maps can be done for each subdomain *independently* from all the others. This allows us to satisfy data locality, which is one of the main factors contributing to scalability on extreme scale machines. In that sense, our approach differs considerably from the classical iterative overlapping Schwarz algorithm, which is not expected to scale well.

Appendix B. Detailed server-client framework.

Our server-client (SC) model relies on grouping MPI processes into servers and clients. Currently, the servers are assumed to be safe (or "sandboxed") units holding the data, whereas the clients are designed solely to accept and perform work without any assumption on their reliability. A client is simply defined as a set of MPI processes, which can take up part or all of a computing node. Conceptually, a client does not necessarily need to be limited to reside on a single node, but can spread over multiple nodes. Assuming the single-node scenario, however, would allow one to exploit in-node parallelism and faster memory access to share data within a client. Clearly, the SC framework involves substantial data exchange between servers and clients, but a key advantage of this structure relies in its inherent resiliency to hard faults, provided that the MPI framework is made fault-tolerant, *e.g.* using the *User Level Failure Mitigation* (ULFM) prototype for MPI [2]. Since the actual data is safely held by the servers, the SC is inherently resilient to clients crashing (partial or complete node failures), since this only translates into missing tasks. The asynchronous nature of the SC model is beneficial to reduce the communication wait times.

Figure 10 shows a schematic of our SC structure. We adopt a cluster-based model, namely the MPI ranks are grouped into separate clusters, with each cluster containing a server and, for resource balancing purposes, the same number of clients. These clusters are designed such that all servers can communicate between each other, while the clients within a cluster are only visible to the server within that cluster. Moreover, within any given cluster, clients are independent, *i.e.* they cannot communicate with each other. The data is distributed among the servers, and these are assumed to be highly resilient (safe or under a sandbox model implementation). The sandbox model assumed for the servers can be supported by either hardware or software. The former assumption is supported by hardware specifications on the variable levels of resilience that can be allowed within large computer systems. In the case of software support, a sandbox effect can be accomplished by a programming model relying on data redundancy and strategic synchronization [24, 5, 18]. Since the servers hold the data, they are responsible for generating work in the form of tasks, dispatching them to their pool of available clients, as well as receiving and processing tasks. To

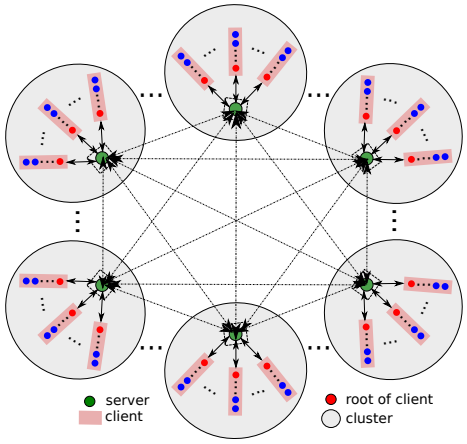


Figure 10: Schematic of the server-client implementation.

optimize communication, clients are designed such that it is their root process that receives a new task to perform. After receiving the task, the root process then broadcasts it to all the other ranks in the client, so that the client as a whole works in parallel to solve the task. This paradigm can be exploited in certain hardware configurations because leveraging local communication within a client is more efficient than having the server communicate a task to all the MPI ranks in a client. One example is the case where all ranks of a client live in the same node, so that they can exploit in-node parallelism and faster memory access. All communications between a server and its clients are done with non-blocking operations, allowing us to overlap them on the server side with the computational operations involved in the creation and processing of the tasks.

Due to separation of state from computation, and because of the nature of the algorithm, sampling and regression are executed by the clients through tasks. On the other hand, because the servers hold the state, they are responsible for performing the solve of the boundary maps system and the subdomains updating. The safety of the servers precludes any potential data corruption during these operations.

REFERENCES

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

[2] W. Bland, A. Bouteiller, T. Herault, G. Bosilca, and J. Dongarra. Post-failure recovery of mpi communication capability: Design and rationale. *International Journal of High Performance Computing Applications*, 27(3):244–254, 2013.

[3] G. Bosilca, R. Delmas, J. Dongarra, and J. Langou. Algorithm-based fault tolerance applied to high performance computing. *Journal of Parallel and Distributed Computing*, 69(4):410–416, Apr 2009.

[4] J. H. Bramble and B. E. Hubbard. New monotone type approximations for elliptic problems. *Mathematics of Computation*, 18(87):349–367, 1964.

[5] P. G. Bridges, K. B. Ferreira, M. A. Heroux, and M. Hoemmen. Fault-tolerant linear solvers via selective reliability. *ArXiv e-prints*, June 2012.

[6] E. J. Candès and Y. Plan. Near-ideal model selection by ℓ_1 minimization. *The Annals of Statistics*, 37(5A):2145–2177, 2009.

[7] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *Signal Processing Magazine, IEEE*,

25(2):21–30, 2008.

[8] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir. Toward exascale resilience: 2014 update. *Supercomputing frontiers and innovations*, 1(1), 2014.

[9] R. Chartrand and W. Yin. Iteratively reweighted algorithms for compressive sensing. In *Acoustics, speech and signal processing, 2008. ICASSP 2008. IEEE international conference on*, pages 3869–3872. IEEE, 2008.

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

[11] P. G. Ciarlet. Discrete maximum principle for finite-difference operators. *Aequationes mathematicae*, 4(3):338–352, 1970.

[12] D. Coleman, P. Holland, N. Kaden, V. Klema, and S. C. Peters. A system of subroutines for iteratively reweighted least squares computations. *ACM Trans. Math. Softw.*, 6(3):327–336, Sept. 1980.

[13] L. Collatz. *The numerical treatment of differential equations*, volume 60. Springer Science & Business Media, 2012.

[14] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2010.

[15] C. Ding, C. Karlsson, H. Liu, T. Davies, and Z. Chen. Matrix multiplication on gpus with on-line fault tolerance. In *Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on*, pages 311–317, 2011.

[16] P. Du, A. Bouteiller, G. Bosilca, T. Herault, and J. 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, pages 225–234, New York, NY, USA, 2012. ACM.

[17] P. DuChateau and D. Zachmann. *Applied Partial Differential Equations*. Dover Books on Mathematics. Dover Publications, 2012.

[18] C. Engelmann and T. Naughton. Toward a performance/resilience tool for hardware/software co-design of high-performance computing systems. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 960–969, Oct 2013.

[19] K. Ferreira, J. Stearley, J. H. Laros, III, R. Oldfield, K. Pedretti, R. Brightwell, R. Riesen, P. G. Bridges, and D. 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, pages 44:1–44:12, New York, NY, USA, 2011. ACM.

[20] D. Gilbarg and N. S. Trudinger. *Elliptic partial differential equations of second order*. springer, 2015.

[21] T.-Z. Huang and Y. Zhu. Estimation of $\|A^{-1}\|_\infty$ for weakly chained diagonally dominant m-matrices. *Linear Algebra and its applications*, 432(2):670–677, 2010.

[22] J. Karátson and S. Korotov. Discrete maximum principles for finite element solutions of nonlinear elliptic problems with mixed boundary conditions. *Numerische Mathematik*, 99(4):669–698, 2005.

[23] P. Laplante. *Dictionary of Computer Science, Engineering and Technology*. Taylor & Francis, 2000.

[24] M.-L. Li, P. Ramachandran, S. K. Sahoo, S. V. Adve, V. S. Adve, and Y. Zhou. Understanding the propagation of hard errors to software and implications for resilient system design. *SIGOPS Oper. Syst. Rev.*, 42(2):265–276, Mar. 2008.

[25] W. Li. The infinity norm bound for the inverse of nonsingular diagonal dominant matrices. *Applied Mathematics Letters*, 21(3):258–263, 2008.

[26] 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*, pages 1–12, 2010.

[27] P. Mycek, A. Contreras, O. Le Maître, K. Sargsyan, F. Rizzi, K. Morris, C. Safta, B. Debusschere, and O. Knio. A resilient domain decomposition polynomial chaos solver for uncertain elliptic PDEs. 2015. Submitted.

[28] M. R. Osborne. *Finite Algorithms in Optimization and Data Analysis*. John Wiley & Sons, Inc., New York, NY, USA, 1985.

[29] S. Portnoy, R. Koenker, et al. The Gaussian hare and the Laplacian tortoise: computability of squared-error versus absolute-error estimators. *Statistical Science*, 12(4):279–300, 1997.

[30] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Numerical mathematics and scientific computation. Clarendon Press, 1999.

[31] F. Rizzi, K. Morris, K. Sargsyan, P. Mycek, C. Safta, B. Debusschere, O. LeMaitre, and O. Knio. ULFM-MPI implementation of a resilient task-based partial differential equations preconditioner. In *Proceedings of the ACM Workshop on Fault-Tolerance for HPC at Extreme Scale*, FTXS '16, pages 19–26, New York, NY, USA, 2016. ACM.

[32] F. Rizzi, K. Morris, K. Sargsyan, P. Mycek, C. Safta, H. Najm, O. Le Maître, O. Knio, and B. Debusschere.

Partial differential equations preconditioner resilient to soft and hard faults. In *FTS – IEEE Cluster*, 2015.

[33] K. Sargsyan, F. Rizzi, P. Mycek, C. Safta, K. Morris, H. Najm, O. L. Maître, O. Knio, and B. Debusschere. Fault resilient domain decomposition preconditioner for PDEs. *SIAM Journal on Scientific Computing*, 37(5):A2317–A2345, 2015.

[34] J. A. Scales and A. Gersztenkorn. Robust methods in inverse theory. *Inverse problems*, 4(4):1071, 1988.

[35] E. J. Schlossmacher. An iterative technique for absolute deviations curve fitting. *Journal of the American Statistical Association*, 68(344):pp. 857–859, 1973.

[36] B. Schroeder, G. Gibson, et al. A large-scale study of failures in high-performance computing systems. *Dependable and Secure Computing, IEEE Transactions on*, 7(4):337–350, 2010.

[37] M. Shashkov. *Conservative finite-difference methods on general grids*, volume 6. CRC press, 1995.

[38] P. Shivakumar, J. J. Williams, Q. Ye, and C. A. Marinov. On two-sided bounds related to weakly diagonally dominant m-matrices with application to digital circuit dynamics. *SIAM Journal on Matrix Analysis and Applications*, 17(2):298–312, 1996.

[39] A. Shye, T. Moseley, V. Reddi, J. Blomstedt, and D. 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*, pages 297–306, 2007.

[40] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson, et al. Addressing failures in exascale computing. *International Journal of High Performance Computing Applications*, page 1094342014522573, 2014.

[41] K. Teranishi and M. A. Heroux. Toward local failure local recovery resilience model using mpi-ulfm. In *Proceedings of the 21st European MPI Users' Group Meeting*, EuroMPI/ASIA '14, pages 51:51–51:56, New York, NY, USA, 2014. ACM.

[42] J. Thomas. *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations*. Texts in Applied Mathematics. Springer New York, 2013.

[43] A. Toselli and O. Widlund. *Domain Decomposition Methods - Algorithms and Theory*. Springer Series in Computational Mathematics. Springer, 2005.

[44] R. S. Varga. *Matrix iterative analysis*, volume 27. Springer Science & Business Media, 2009.