Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

# Bi-directional coupling between a PDE-domain and an adjacent Data-domain equipped with multi-fidelity sensors

Dongkun Zhang, Liu Yang, George Em Karniadakis [*]

*Division of Applied Mathematics, Brown University, Providence, RI 02912, USA*

A R T I C L E   I N F O

*Article history:*
Received 1 December 2017
Received in revised form 17 July 2018
Accepted 20 July 2018
Available online 24 July 2018

*Keywords:*
Gaussian process regression
Domain decomposition
Multi-fidelity
Machine learning

A B S T R A C T

We consider a new prototype problem in domain decomposition with the solution in one domain governed by a known partial differential equation (PDE) whereas the solution in an adjacent domain is reconstructed by information gathered from distributed sensors (data) of variable fidelity. The PDE-domain and the Data-domain are tightly coupled, as the PDE solution is driven by the collected data, while the information gathered from its associated sensors is influenced by the PDE solution. Our overall methodology is based on the Schwarz alternating method and on recent advances in Gaussian process regression (GPR) using multi-fidelity data. The effectiveness of the proposed domain decomposition algorithm is demonstrated by examples of Helmholtz equations in both one-dimensional (1D) and two-dimensional (2D) domains.

## 1. Introduction

There has been substantial progress in the last ten years on data inference models, but in many practical situations the data can only be collected in a restricted area from a number of scattered and heterogeneous sensors of variable fidelity. In subsurface applications, for example, we may have such measurements in one region, where we may or may not know the physics of the process, and this data domain may be adjacent to a more homogeneous domain in which we know the physical process and hence the mathematical model in the form of a PDE. We assume here that we have bi-directional coupling, such as the dispersion in porous media, between the data domain and the PDE domain. Currently, there is no scientific method to model this bi-directional propagation of information and the existing data inference techniques, e.g., for streaming data, are relying on very accurate high-fidelity input. Here we address, for the first time, such hybrid data-PDE problems and develop a method that propagates the information through the data and PDE domains. This new concept is schematically shown in Fig. 1, where we allow a bi-directional coupling between the data domain (right, yellow) and the PDE domain (left, blue). In addition, we assume that we have available on the data domain diverse sensors depicted by green and red to represent different fidelities of the data. In particular, we have only a few green data points that we fully trust whereas the majority of the data is represented by red crosses to denote less accurate or potentially even misleading information. This algorithm would naturally fall into the scope of domain decomposition methods, which have already been studied extensively within a deterministic problem setup [1–6], based on the classical Schwarz alternating algorithm [7].

[*] Corresponding author.
*E-mail address:* george_karniadakis@brown.edu (G.E. Karniadakis).

https://doi.org/10.1016/j.jcp.2018.07.039
0021-9991/© 2018 Elsevier Inc. All rights reserved.

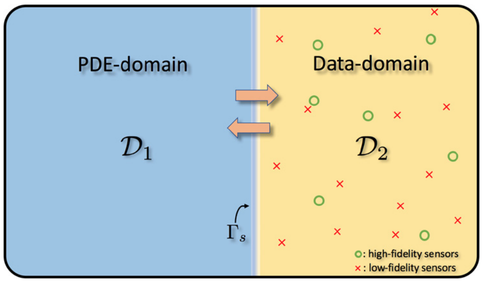


**Fig. 1.** The physical domain $\mathcal{D}$ is decomposed into non-overlapping subdomains $\mathcal{D}_1$ and $\mathcal{D}_2$, where $\mathcal{D}_1$ is the PDE-domain and $\mathcal{D}_2$ is the Data-domain equipped with high-fidelity sensors (green circles) and low-fidelity sensors (red crosses). Information from the PDE-domain and the Data-domain propagates in both directions across the interface $\Gamma_s$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

This domain coupling is also a natural extension of the previous work on propagation of stochastic solutions across domains [8–11].

Various methods [12–14] for solving a PDE based on machine learning tools have been developed recently, among which GPR [15] serves as an effective means for designing such data-driven algorithms. Raissi et al. [16] have proposed a GPR based algorithm for inferring solutions of differential equations from noisy multi-fidelity data, where the cheap, frequent but low-fidelity observations could help with the expensive, scarce and high-fidelity observations to obtain solutions with better accuracy. Inspired by such previous work in GPR and domain decomposition methods, we develop a new algorithm that reconstructs the solution from scattered sensors in the data subdomain coupled with an adjacent PDE subdomain, and we arrive at a converged global solution using an iterative method, similar to the Schwarz alternating method [7].

The organization of this paper is as follows. In section 2, we set up the domain decomposition problem and specify the two types of subdomains. In section 3, we introduce the Schwarz iterative method and the numerical GPR method for solving PDEs, using data of single- and multi-fidelity; the core algorithm of this paper is also described in this section. In section 4, we demonstrate the performance of the proposed algorithm in both 1D and 2D examples, and we conclude with a brief summary in section 5.

## 2. Problem setup

Suppose $\mathcal{D}$ is the entire physical domain and is divided into two non-overlapping subdomains $\mathcal{D}_1$ and $\mathcal{D}_2$ as depicted in Fig. 1, where $\Gamma_s$ represents the interface shared by those two subdomains. We refer to $\mathcal{D}_1$ as the *PDE-domain*, because the quantity of interest (QoI) $u(x)$ is governed by a known PDE in $\mathcal{D}_1$,

$$\begin{cases} \mathcal{L}_x u(x) = f(x), & x \in \mathcal{D}_1, \\ \mathcal{B}_x u(x) = g(x), & x \in \partial\mathcal{D}_1 \backslash \Gamma_s, \end{cases} \tag{1}$$

where $\mathcal{L}_x$ is the partial differential operator, $f(x)$ is the forcing term, $\mathcal{B}_x$ is a proper boundary condition operator acting on all boundaries of $\mathcal{D}_1$, except for $\Gamma_s$, and $g(x)$ is the prescribed boundary condition. We refer to $\mathcal{D}_2$ as the *Data-domain*, where we place sensors of the solution $u(x)$ at sparse locations, and collect data from the sensors. In another scenario, we may have some sparse measurements of the forcing (right-hand-side) term $f(x)$ of a known PDE, and hence we will subsequently solve the PDE using numerical GPR[16] instead of the classical discretization method as in domain $\mathcal{D}_1$. Moreover, these sensors provide information of variable fidelity.

Since a physically admissible solution $u(x)$ should satisfy some continuity conditions at the interface $\Gamma_s$, such as the continuity of $u(x)$ and $\nabla u(x)$, the value of $u(x)$ on $\Gamma_s$, which may be undetermined at first, depends on $u(x)$ in both $\mathcal{D}_1$ and $\mathcal{D}_2$. Our goal is to design a domain decomposition algorithm that imposes the aforementioned continuity constraints and will make use of both the PDE in $\mathcal{D}_1$ and the sparse multi-fidelity sensors data in $\mathcal{D}_2$ to solve for $u(x)$ in the entire domain $\mathcal{D}$.

## 3. Methodology

### 3.1. Alternating Schwarz algorithm

As an example, consider the elliptic Helmholtz equation,

$$\begin{aligned} -\nabla^2 u + \lambda^2 u - f(x) = 0, \quad & x \in \mathcal{D}, \\ u = 0, \quad & x \in \partial\mathcal{D}. \end{aligned} \tag{2}$$

For a 1D domain, consider the decomposition in Fig. 2, where we refer to $\mathcal{D}_m$ as the *dominus* domain for which we impose a Dirichlet type interface condition at $x_b$, and $\mathcal{D}_s$ as the *servus* domain with a Neumann type interface condition.

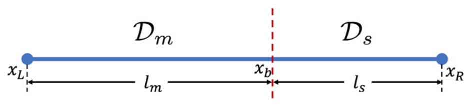


**Fig. 2.** 1D domain decomposition diagram, where $\mathcal{D}_m$ is the *dominus* domain with length $l_m$, and $\mathcal{D}_s$ is the *servus* domain with length $l_s$. The two subdomains share the boundary point $x_b$.

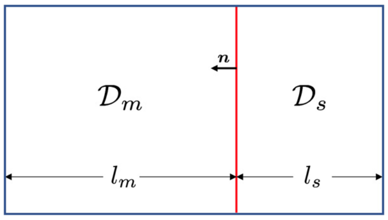


**Fig. 3.** 2D domain decomposition diagram, where $\mathcal{D}_m$ is the *dominus* domain, $\mathcal{D}_s$ is the *servus* domain, and $\boldsymbol{n}$ is the outer norm of $\mathcal{D}_s$ at the interface.

We employ a relaxed version of the alternating Schwarz algorithm for non-overlapping subdomains, first proposed by Funaro et al. [2] and modified in [5]:
*dominus domain*:

$$\begin{aligned} &-(u_{xx})_m^n + \lambda^2 u_m^n = f(x), \quad \text{on } \mathcal{D}_m, \\ &u(x_L) = 0, \\ &u_m^n(x_b) = \eta u_s^{n-1}(x_b) + (1-\eta) u_m^{n-1}(x_b), \quad \eta \in (0,1], \end{aligned} \tag{3}$$

*servus domain*:

$$\begin{aligned} &-(u_{xx})_s^n + \lambda^2 u_s^n = f(x), \quad \text{on } \mathcal{D}_s, \\ &u(x_R) = 0, \\ &(u_x)_s^n(x_b) = (u_x)_m^n(x_b). \end{aligned} \tag{4}$$

In Equations (3) and (4), $u_m^n$ and $u_s^n$ denote the solutions for their respective subdomains during the $n$th iteration. A proper relaxation parameter $\eta$ is imposed when updating the Dirichlet boundary condition to guarantee and speed up the convergence [6,17]. The iterative process stops when the change of solution at the interface between successive iterations is smaller than a designated threshold $\epsilon$, i.e.,

$$|u_m^n(x_b) - u_m^{n-1}(x_b)| + |u_s^n(x_b) - u_s^{n-1}(x_b)| < \epsilon. \tag{5}$$

Similarly, for a 2D domain decomposition, e.g. Fig. 3, we apply the following iterating boundary conditions:

$$u_m^n = \eta u_s^{n-1} + (1-\eta) u_m^{n-1}, \quad \boldsymbol{n} \cdot \nabla u_s^n = -\boldsymbol{n} \cdot \nabla u_m^n, \tag{6}$$

where $\boldsymbol{n}$ is the outward unit normal vector of the *servus* domain at the interface.

### 3.2. Numerical Gaussian process regression

The general form of a linear PDE is,

$$\begin{cases} \mathcal{L}_x u(x) = f(x), & x \in \Omega, \\ u(x) = p(x), & x \in \Gamma_D, \\ \frac{\partial}{\partial \boldsymbol{n}} u(x) = q(x), & x \in \Gamma_N, \end{cases} \tag{7}$$

where $\mathcal{L}_{(\cdot)}$ is a linear operator (here $\mathcal{L}_x$ means that the operator acts on the variable $x$), $f(x)$ is the external forcing term, $p(x)$ and $q(x)$ are the Dirichlet and Neumann boundary conditions on their respective boundaries $\Gamma_D$ and $\Gamma_N$, and $\boldsymbol{n}$ is the unit outer normal. Assuming that the PDE solution $u(x)$ belongs to $C^1(\Omega)$, the solution space can be approximated using Gaussian random processes with a covariance kernel characterized by tunable hyper-parameters $\theta$, while the actual solution is one of the possible Gaussian process trajectories. We shall infer the actual solution using a machine learning strategy, given the sensors' data of $f(x)$ and $u(x)$ at sparse locations and possibly other boundary information. The hyper-parameters $\theta$ in the covariance kernel can be estimated using a maximum likelihood estimation, and after that the solution $u(x)$ for any provided $x$ shall be predicted via the Bayesian estimation. In practice, the sensors' data might be polluted by random measurement error, which could be modeled by independent zero-mean Gaussian random variables with standard deviation $\sigma_u$ and $\sigma_f$ that are either known constants, or could be learned together with $\theta$ from the data.

To be more specific, let $\boldsymbol{x} = (x_1, x_2, ..., x_{N_0})$ be the locations where we place our $u(x)$ sensors. Assuming a Gaussian process prior of the solution $u(x)$, i.e.,

$$u(x) \sim \mathcal{GP}(0, g(x, x'; \theta)), \tag{8}$$

where $g(x, x'; \theta)$ is the covariance kernel that takes the form of, for example, the squared exponential kernel

$$g(x, x'; \theta) = \sigma_l^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D} \frac{(x^{(d)} - x'^{(d)})^2}{l_d^2}\right). \tag{9}$$

Non-stationary kernels, such as the neural network covariance function [15,18,19], can also be applied, and the main algorithm to be proposed does not restrict itself to stationary kernels either. For demonstrative purpose we will use the squared exponential kernel described here for our numerical tests. In Equation (9), $x^{(d)}$ is the $d$th dimensional coordinate of $x$ and $D$ is the dimension of the Data-domain, and the hyper-parameters to be optimized are $\theta = (\sigma_l, l_1, l_2, ..., l_D)$. The Dirichlet boundary conditions can be viewed as noiseless sensors (sensors that return the exact value) on its respective boundary and they will be handled in the same way as the other $u(x)$ sensors inside the domain. Thus we only need to consider the situation when we have Neumann boundary conditions. Let $\boldsymbol{z} = (z_1, z_2, ..., z_{N_1})$ be the locations where we place $f(x)$ sensors and $\boldsymbol{y} = (y_1, y_2, ..., y_M)$ be the locations of sampling points on the Neumann boundary. Since the derivatives and linear combinations of Gaussian processes are still Gaussian processes, we have

$$\begin{aligned} u'(\boldsymbol{y}) &\sim \mathcal{GP}(0, k(\boldsymbol{y}, \boldsymbol{y}'; \theta)),\\ f(\boldsymbol{z}) &\sim \mathcal{GP}(0, h(\boldsymbol{z}, \boldsymbol{z}'; \theta)), \end{aligned} \tag{10}$$

where

$$\begin{aligned} k(\boldsymbol{y}, \boldsymbol{y}'; \theta) &= \frac{\partial}{\partial \boldsymbol{n_1}} \frac{\partial}{\partial \boldsymbol{n_2}} g(\boldsymbol{y}, \boldsymbol{y}'; \theta),\\ h(\boldsymbol{z}, \boldsymbol{z}'; \theta) &= \mathcal{L}_{\boldsymbol{z}} \mathcal{L}_{\boldsymbol{z}'} g(\boldsymbol{z}, \boldsymbol{z}'; \theta). \end{aligned} \tag{11}$$

In Equation (11), $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ are the unit outer normal of the first and second entries of the covariance kernel function. Moreover, the joint distribution of $u(\boldsymbol{x})$, $u'(\boldsymbol{y})$ and $f(\boldsymbol{z})$ is

$$U = \begin{bmatrix} u(\boldsymbol{x}) \\ u'(\boldsymbol{y}) \\ f(\boldsymbol{z}) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_{00} & K_{01} & K_{02} \\ K_{10} & K_{11} & K_{12} \\ K_{20} & K_{21} & K_{22} \end{bmatrix}\right), \tag{12}$$

where

$$\begin{aligned} &K_{00} = g(\boldsymbol{x}, \boldsymbol{x}; \theta) + \sigma_u^2 \boldsymbol{I}, && K_{01} = \frac{\partial}{\partial \boldsymbol{n_2}} g(\boldsymbol{x}, \boldsymbol{y}; \theta), && K_{02} = \mathcal{L}_{\boldsymbol{z}} g(\boldsymbol{x}, \boldsymbol{z}; \theta),\\ &K_{10} = K_{01}^T, && K_{11} = k(\boldsymbol{y}, \boldsymbol{y}; \theta), && K_{12} = \frac{\partial}{\partial \boldsymbol{n_1}} \mathcal{L}_{\boldsymbol{z}} g(\boldsymbol{y}, \boldsymbol{z}; \theta),\\ &K_{20} = K_{02}^T, && K_{21} = K_{12}^T, && K_{22} = h(\boldsymbol{z}, \boldsymbol{z}; \theta) + \sigma_f^2 \boldsymbol{I}. \end{aligned} \tag{13}$$

In Equation (13), the additional variances due to the measurement error are included in $K_{00}$ and $K_{22}$ as $\sigma_u^2 I$ and $\sigma_f^2 I$.

The sensors' data and sample points on domain boundaries serve all together as the training data set. The covariance kernel hyper-parameter $\theta$ (including $\sigma_u$ and $\sigma_f$, if they are not given) will be estimated by minimizing the negative log marginal likelihood defined by

$$\mathcal{NLML} := -\log p(U | \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}; \theta, \sigma_u, \sigma_f), \tag{14}$$

which can be explicitly written as

$$\mathcal{NLML} = \frac{1}{2} \boldsymbol{Y}^T K^{-1} \boldsymbol{Y} + \frac{1}{2} \log |K| + \frac{n}{2} \log(2\pi), \tag{15}$$

where $\boldsymbol{Y} := [u(\boldsymbol{x}), u'(\boldsymbol{y}), f(\boldsymbol{z})]^T$, and $n$ is the total number of training points [15].

Given the assumption that the solution is a Gaussian process, the value $u(x_0)$ at $x_0 \in \Omega$ and all the training data follow a multi-variant Gaussian distribution,

$$\begin{bmatrix} u(x_0) \\ \boldsymbol{Y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} g(x_0, x_0; \sigma_u, \sigma_f) & \boldsymbol{a} \\ \boldsymbol{a^T} & K \end{bmatrix}\right), \tag{16}$$

where $\boldsymbol{a} = [g(x_0, \boldsymbol{x}; \theta), \frac{\partial}{\partial \boldsymbol{n_2}} g(x_0, \boldsymbol{y}; \theta), \mathcal{L}_z g(x_0, \boldsymbol{z}; \theta)]$. The posterior distribution of $u(x_0)$ given all the training data is then calculated from the Bayesian formula,

$$p(u(x_0)|\boldsymbol{Y}) = \mathcal{N}(\boldsymbol{a}K^{-1}\boldsymbol{Y}, g(x_0, x_0; \theta) - \boldsymbol{a}K^{-1}\boldsymbol{a}^{\boldsymbol{T}}). \tag{17}$$

Therefore, the maximum a posteriori estimation of $u(x_0)$ is $\boldsymbol{a}K^{-1}\boldsymbol{Y}$, with the prediction variance $g(x_0, x_0; \theta) - \boldsymbol{a}K^{-1}\boldsymbol{a}^{\boldsymbol{T}}$.

### 3.3. Inferring solutions of PDEs from multi-fidelity data

Here we consider a realistic situation where the information gathered by the sensors is of variable fidelity due to primarily different sensor qualities, for example, different resolutions. Typically, the availability of high-fidelity data is quite limited, while the low-fidelity data is much easier to collect. In general, low-fidelity data could come from inexpensive sensors or uncalibrated measurements, e.g., satellite data, or even from computations, e.g., using inexpensive reduced-order models. We denote the high-fidelity model of $u(x)$ by $u^h(x)$, and the low-fidelity model of $u(x)$ by $u^l(x)$. The auto-regressive model in [20,16,21] reads

$$u^h(x) = \rho u^l(x) + \delta(x), \tag{18}$$

where $u^l(x)$ and $\delta(x)$ are two independent Gaussian processes with

$$u^l(x) \sim \mathcal{GP}(0, g_1(x, x'; \theta_1)), \quad \delta(x) \sim \mathcal{GP}(0, g_2(x, x'; \theta_2)). \tag{19}$$

Here, $g_1(x, x'; \theta_1)$ and $g_2(x, x'; \theta_2)$ are covariance functions, $\theta_1, \theta_2$ denote their hyper-parameters and $\rho$ is the cross-correlation parameter to be learned from the data. Therefore, from Equation (18) we can get

$$u^h(x) \sim \mathcal{GP}(0, g(x, x'; \theta_1, \theta_2)), \tag{20}$$

where

$$g(x, x'; \theta_1, \theta_2) = \rho^2 g_1(x, x'; \theta_1) + g_2(x, x'; \theta_2). \tag{21}$$

Given that the operator $\mathcal{L}_x$ is linear, we arrive at the auto-regressive structure $f^h(x) = \rho f^l(x) + \gamma(x)$ on the forcing, where $\gamma(x) = \mathcal{L}_x \delta(x)$ and $f^l(x) = \mathcal{L}_x u^l(x)$ are two independent Gaussian processes. We note that the low-fidelity model $u^l(x)$ and $f^l(x)$ satisfies the same equation as the high-fidelity model.

Suppose we have two types of $f(x)$ sensors, where the highly accurate but expensive sensors are placed at location $\boldsymbol{z}^h$, while the sensors low accuracy are placed at $\boldsymbol{z}^l$. Thus, the joint distribution of all training data is

$$\begin{bmatrix} u^h(\boldsymbol{x}) \\ u'(\boldsymbol{y}) \\ f^h(\boldsymbol{z}^h) \\ f^l(\boldsymbol{z}^l) \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_{00} & K_{01} & K_{02} & K_{03} \\ K_{10} & K_{11} & K_{12} & K_{13} \\ K_{20} & K_{21} & K_{22} & K_{23} \\ K_{30} & K_{31} & K_{32} & K_{33} \end{bmatrix} \right). \tag{22}$$

In Equation (22), term $K_{00}$, $K_{01}$, $K_{02}$, $K_{10}$, $K_{11}$, $K_{12}$, $K_{20}$, $K_{21}$, $K_{22}$ are defined in the same way as in Equation (13), with $g(x, x'; \theta)$ replaced by Equation (21). Due to the independence assumption of $u^l(x)$ and $\delta(x)$, we can write down the rest of the matrix explicitly,

$$\begin{aligned} K_{03} &= \rho \mathcal{L}_{\boldsymbol{z}^l} g_1(\boldsymbol{x}, \boldsymbol{z}^l; \theta_1), & K_{13} &= \rho \frac{\partial}{\partial \boldsymbol{n_1}} \mathcal{L}_{\boldsymbol{z}^l} g_1(\boldsymbol{y}, \boldsymbol{z}^l; \theta_1), \\ K_{23} &= \rho \mathcal{L}_{\boldsymbol{z}^h} \mathcal{L}_{\boldsymbol{z}^l} g_1(\boldsymbol{z}^h, \boldsymbol{z}^l; \theta_1), & K_{33} &= \mathcal{L}_{\boldsymbol{z}^l} \mathcal{L}_{\boldsymbol{z}^l} g_1(\boldsymbol{z}^h, \boldsymbol{z}^l; \theta_1) + \sigma_{f^l}^2 \boldsymbol{I}, \end{aligned} \tag{23}$$

and $K_{30} = K_{03}^T$, $K_{31} = K_{13}^T$, $K_{32} = K_{23}^T$. The training and predicting procedures resemble those in the single-fidelity situation. We note that if possible, it is desirable to use nested sensors so that the high-fidelity sensors are at the same location as the low-fidelity sensors to reduce the computational cost [22], however, it is appreciated that may not possible in a real situation.

### 3.4. Domain decomposition algorithm with Gaussian process regression

**Algorithm: GPDD**

```
1: Initializing $u_m^0(x_b)$ with 0.
2: Set $k = 0$;
3: while $\|u_m^{k-1}(x_b) - u_m^k(x_b)\| \le \epsilon_{tol}$ do
4:     Solve for $u_m^k(x)$ with the given solver in $\mathcal{D}_m$;
5:     Calculate ${u'_m}^k(x_b)$ as the Neumann boundary condition in $\mathcal{D}_s$;
6:     Predict $u_s^k(x)$ with GPR in $\mathcal{D}_s$;
7:     Set $u_m^{k+1}(x_b) = u_s^k(x_b)$;
8:     $k = k + 1$;
9: end while
```

As the core of this work, we introduce the hybrid domain decomposition algorithm with Gaussian process regression (GPDD). Suppose we are provided with a PDE solver in the *dominus* domain $\mathcal{D}_m$, which we call the PDE-domain, while in the *servus* domain $\mathcal{D}_s$, also referred to as the Data-domain, we have access to sensors data. The GPDD algorithm serves to couple the data together with the PDE solver via the Dirichlet–Neumann type Schwarz iterative method. We note that a reversed version of interface conditions could also be applied, i.e., imposing a Dirichlet boundary condition to the Data-domain and a Neumann boundary condition to the PDE-domain. A comparison between both versions in terms of the solution accuracy is displayed in section 4. For simplicity, we set the PDE-domain to be the *dominus* domain and the Data-domain to be the *servus* domain unless it is specified otherwise. Depending on the type of sensors we have in $\mathcal{D}_s$, this GPDD algorithm can be applied in the following two cases:

*Case 1:* We have a PDE solver in the PDE-domain, while in the Data-domain we have access to the sensors' data of $u$. We assume that we do *not* know the PDE in the Data-domain.

*Case 2:* We have a PDE solver in the PDE-domain, while in the Data-domain we have access to the sensors' data of the forcing term $f$, which could be of variable fidelity. In this case, we know the exact form of the PDE left-hand-side operator $\mathcal{L}_x$ in the Data-domain. No other information about $u$, except for the boundary conditions, is needed in the Data-domain.

## 4. Numerical results

### 4.1. 1D Helmholtz equation

We solve the 1D Helmholtz equation

$$-u_{xx} + \lambda^2 u = f(x), \quad x \in [0,1], \quad u(0) = u(1) = 0, \tag{24}$$

where $\lambda = 1$ is a constant, $f(x)$ is the forcing term, and $u(x)$ is the QoI. The entire domain $\mathcal{D} : [0,1]$ is divided into two non-overlapping subdomains: the PDE-domain $\mathcal{D}_1 : [0, 0.6]$, and the Data-domain $\mathcal{D}_2 : [0.6, 1.0]$. The numerical PDE solver in $\mathcal{D}_1$ is implemented using a spectral/*hp* element method [6,23] with 6 spectral elements and the 10th order polynomial but will only be accessed as a black box. We manufactured the reference solution so that it displays two different length scales in different subdomains:

$$u(x) = \begin{cases} 3\sin\left(\frac{10\pi x}{3}\right), & x \in \mathcal{D}_1, \\ \sin\left(\frac{45\pi}{2}(x - 0.6)\right) - \frac{5}{2}\sin(5\pi(x - 0.6)), & x \in \mathcal{D}_2. \end{cases} \tag{25}$$

The resulting forcing term $f(x)$ is

$$f(x) = \begin{cases} \left(-\frac{100\pi^2}{3} + 3\right)\sin\left(\frac{10\pi x}{3}\right), & x \in \mathcal{D}_1, \\ \left(-\frac{2025\pi^2}{4} + 1\right)\sin\left(\frac{45\pi}{2}(x - 0.6)\right) & \\ \qquad + \frac{5}{2}(25\pi^2 - 1)\sin(5\pi(x - 0.6)), & x \in \mathcal{D}_2. \end{cases} \tag{26}$$

The GPDD solution is denoted by $\tilde{u}(x)$. Both cases in Section 3.4 will be considered, and we set the Schwarz iteration terminating threshold $\epsilon$ in Equation (5) to be $10^{-7}$. We test our GPDD algorithm performance using both noiseless and noisy sensor data.

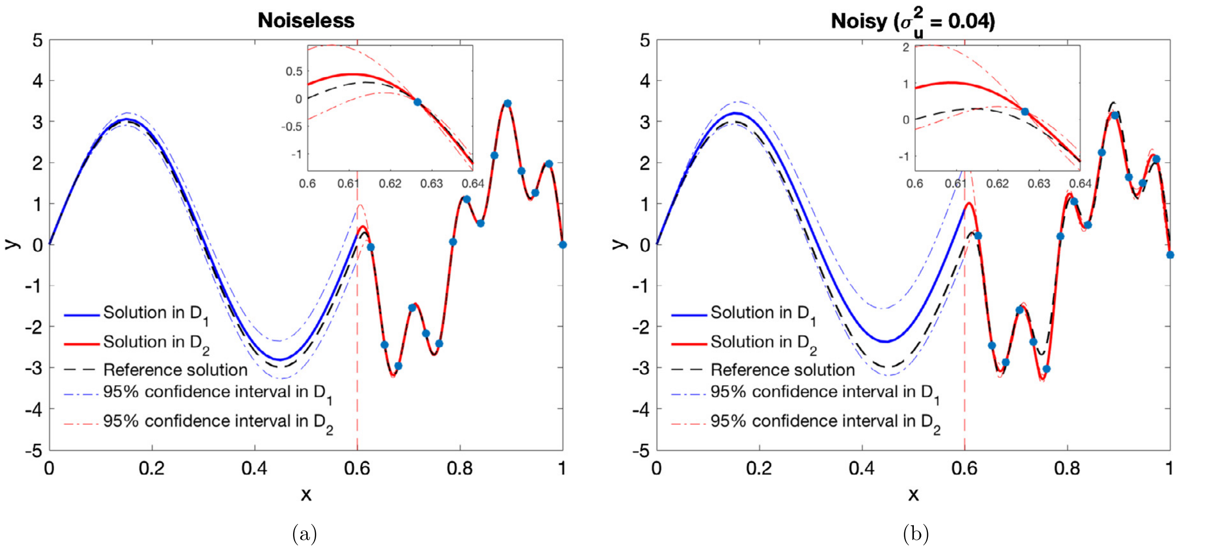


**Fig. 4.** Case 1: The GPDD solutions of the 1D Helmholtz equation using (a) noiseless sensors data, and (b) noisy sensors data polluted by the Gaussian noise of standard deviation $\sigma_u = 0.2$. The left-hand side is the PDE-domain and the right-hand side is the Data-domain. The interface between them is denoted by the vertical dashed red line. The Data-domain contains 15 $u(x)$ sensors, marked by the blue dots. The dashed lines mark the 95% confidence interval of the prediction.

**Table 1**
Case 1: Number of iterations needed for the Schwarz iterations to converge, using $\epsilon = 10^{-7}$ threshold.

| Number of sensors | 20 | 25 | 30 | 35 | 40 |
|---|---|---|---|---|---|
| Noiseless ($\sigma = 0$) | 17 | 15 | 15 | 14 | 13 |
| Noisy ($\sigma = 0.2$) | 18 | 17 | 17 | 17 | 16 |

#### 4.1.1. *Case 1*

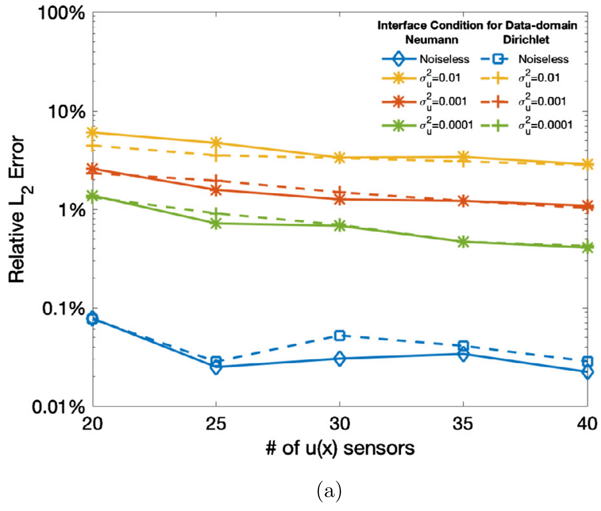
For demonstration purpose, we distribute the $u(x)$ sensors uniformly in the Data-domain $\mathcal{D}_2$. To generate noisy sensors data, we deliberately add independent Gaussian noise of a fixed standard deviation $\sigma_u$ to each sensor. In Figs. 4a and 4b, we compare the GPDD solution $\tilde{u}(x)$, calculated with both noiseless and noisy sensors, and the reference solution $u(x)$. As we can see, even if the exact solution displays a multi-scale property, for both cases, the GPDD solution $\tilde{u}(x)$ can approximate $u(x)$ very well. Although the problem to be solved is deterministic, predicting the solution in $\mathcal{D}_2$ is indeed the progress of maximum likelihood estimation, and therefore, for every $x \in \mathcal{D}_2$ there exists a Gaussian distribution associated with the predicted solution $\tilde{u}(x)$. Therefore, we can feed the PDE-domain $\mathcal{D}_1$ with a Dirichlet interface condition of a Gaussian distribution, instead of a single value. Due to the linearity of the PDE, solving the equation in $\mathcal{D}_1$ generates a Gaussian distributed Neumann boundary condition at the interface, which shall be passed to the numerical GPR solver in $\mathcal{D}_2$ naturally as a "noisy" boundary condition measurement. By repeatedly solving for $\tilde{u}(x)$ in both subdomains with Gaussian distributed boundary conditions, we manage to propagate the uncertainty back and forth between these two subdomains. We can observe that in Fig. 4a the exact solution falls into the 95% confidence interval around the predicted solution generated by noiseless sensors.

Table 1 shows the number of iterations needed for the Schwarz iterative scheme to converge when an "optimal" relaxation parameter $\eta = 0.58$ (calculated from the formula derived in [6]) is employed. Ideally, if both subdomains are equipped with traditional PDE solvers, the serial iterations should converge after two steps, but when we have a GPR solver in the Data-domain, we observe that more iterations are needed to converge, due to the fact that the estimated solution cannot be simply characterized by a fixed PDE during iterations, especially when the sensor data are polluted by the random measurement error (noise). Nevertheless, we can observe the trend that given more sensors, less iterations are needed for the GPDD algorithm to converge at least for noiseless data.

The accuracy of the numerical solution $\tilde{u}$ is measured by the relative $L_2$ error $\epsilon_r$, defined by

$$\epsilon_r = \frac{\|\tilde{u} - u\|}{\|u\|}, \tag{27}$$

where $\|\cdot\|$ denotes the $L_2$ norm taken in the entire domain. Fig. 5a compares the relative error of the solutions obtained from the proposed GPDD algorithm and the algorithm of reversed boundary conditions, i.e., feeding the PDE-domain with a Neumann boundary condition and the Data-domain with a Dirichlet boundary condition. Both algorithms produce solutions of similar accuracy, and obviously, we get the most accurate solutions with the noiseless sensors. If we put more $u(x)$



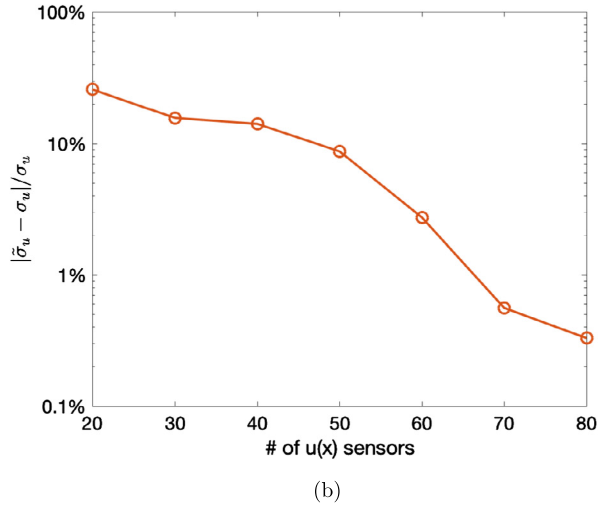
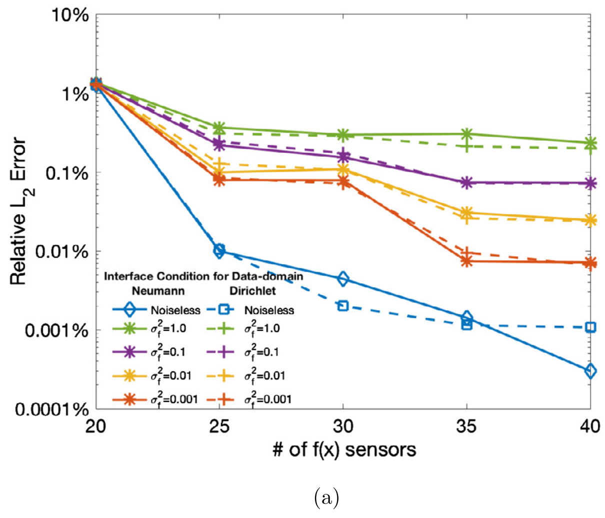


**Fig. 5.** Case 1: (a) The relative $L_2$ error in the entire domain of the predicted solution versus the number of $u(x)$ sensors in the Data-domain $\mathcal{D}_2$ for different levels of sensor's noise. Both the proposed GPDD algorithm and the algorithm with reversed interface conditions are implemented here. (b) The relative error of the predicted magnitude of noise $\tilde{\sigma}_u$ (from the GPDD algorithm), with respect to the input (nominal) noise $\sigma_u$.



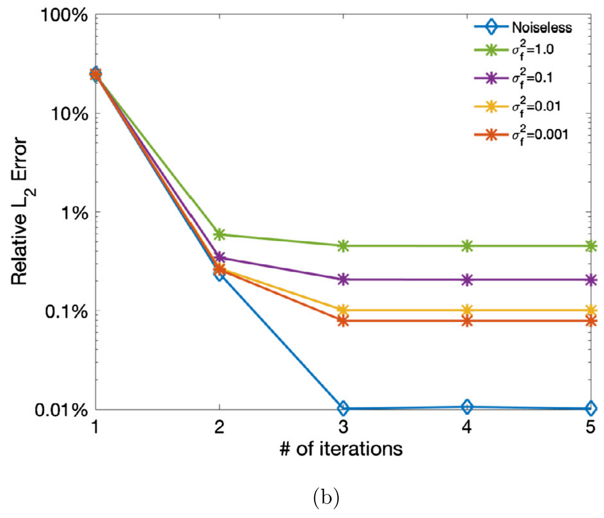


**Fig. 6.** Case 2: (a) The relative $L_2$ error in the entire domain of the predicted solution versus the number of $f(x)$ sensors in the Data-domain $\mathcal{D}_2$ for different levels of sensor's noise. Both the proposed GPDD algorithm and the algorithm with reversed interface conditions are implemented here. (b) For different levels of sensor noise, the iterative process of the GPDD algorithm converges within 5 iterations. In this example, 25 sensors are used in the Data-domain.

sensors in $\mathcal{D}_2$, we would generally obtain solutions with slightly better accuracy. This makes sense because the more data we collect, the better knowledge we have about the pattern of the solution. Also, as demonstrated in Fig. 5b, the standard deviation of sensors' noise, $\sigma_u$, is learned with the GPDD algorithm, and by increasing the number of sensors we learn the sensors' noise better.

### 4.1.2. *Case 2*

In this case, the sensors for $f(x)$ are placed in the Data-domain $\mathcal{D}_2$, and the linear operator $\mathcal{L}_x$ is known a priori:

$$\mathcal{L}_x := -\frac{d^2}{dx^2} + I. \tag{28}$$

We test the GPDD algorithm in both situations using either noiseless or noisy sensors. Similarly, the noisy $f$ sensor are manufactured by adding independent Gaussian random noise of standard deviation $\sigma_f$ to each sensor.

In Fig. 6a we compare the accuracy of the GPDD algorithm and the algorithm of reversed boundary conditions for different numbers of $f(x)$ sensors and different levels of sensors' noise. Again, both algorithms display similar accuracy. It is evident that the relative $L_2$ error decreases when we place a greater larger number of less noisy sensors in the Data-domain.

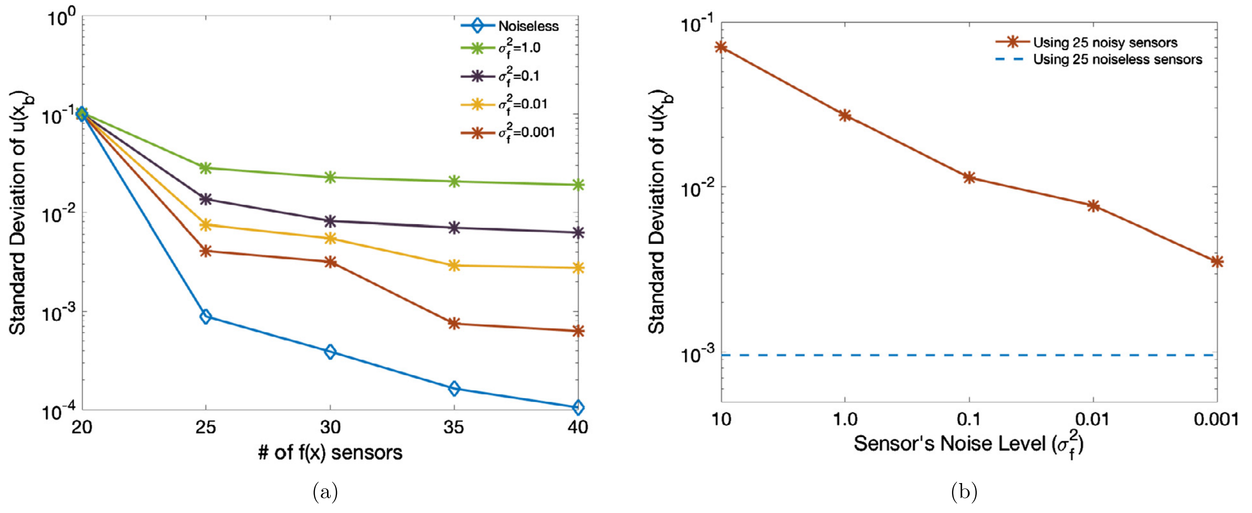


**Fig. 7.** Case 2: (a) The standard deviation of $u(x_b)$ is reduced when more $f(x)$ sensors are used. (b) The standard deviation of $u(x_b)$ is reduced when less noisy $f(x)$ sensors are being used.

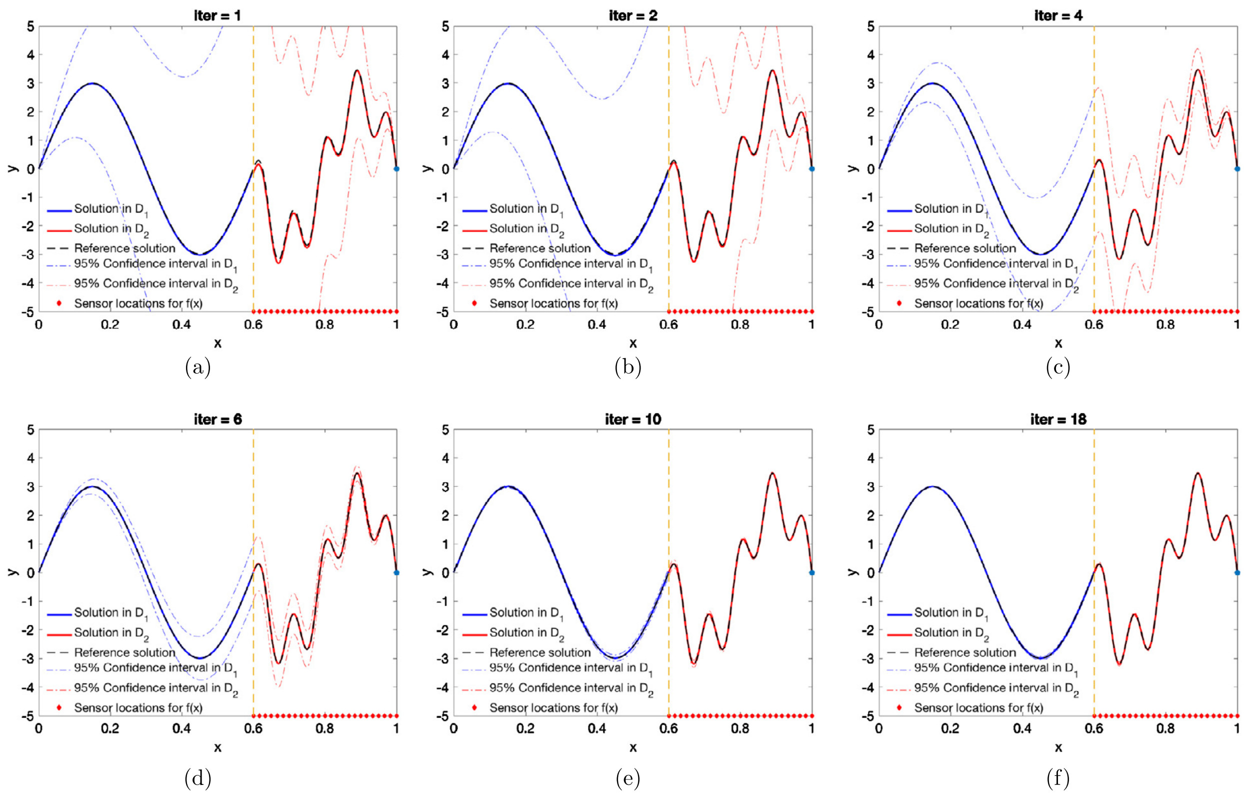


**Fig. 8.** Case 2: The solution profile of the 1D Helmholtz equation, using 25 noisy ($\sigma_f = 1.0$) sensors in $\mathcal{D}_2$. The 95% confidence interval of predicted solution is reduced quickly in less than 10 Schwarz iterations.

By using the noiseless sensors, we achieve a very accurate solution with the relative $L_2$ error less than 0.01%. Even when we use noisy sensors of standard deviation $\sigma_f = 1.0$, we still obtain solutions with relative error less than 1%. Fig. 6b shows the convergence of the GPDD algorithm after a few iterations when 25 $f(x)$ sensors are uniformly placed in the Data-domain and an optimal $\eta = 0.58$ is adopted. The relative error of the GPDD solution decays to reach a stable level within 5 iterations, indicating very fast convergence of the GPDD algorithm for this case.

We are also interested in the uncertainty associated with our GPDD solution. Fig. 7a shows that when more sensors are placed in $\mathcal{D}_2$, the standard deviation of the predicted $\tilde{u}(x_b)$ decays, indicating that we are more confident with our solution. Fig. 7b also confirms that less noisy sensors would result in more confident prediction of solution. In Fig. 8 we

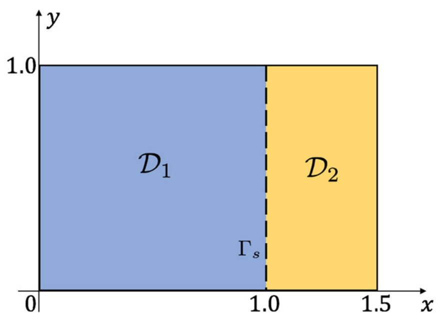


**Fig. 9.** The 2D physical domain and its decomposition for case 1: *dominus* domain: $\mathcal{D}_1 := [0,1] \times [0,1]$, and *servus* domain: $\mathcal{D}_2 := [1,1.5] \times [0,1]$.

plot the 95% confidence interval of the predicted solution $\tilde{u}(x)$, obtained with 25 noisy sensors ($\sigma_f = 1.0$) in $\mathcal{D}_2$. Here we intentionally choose the boundary relaxation parameter $\eta = 0.3$ to suppress the convergence rate. The confidence interval shrinks rapidly within the first few iterations, indicating that we are gaining confidence of our predictions through this Schwarz type iteration.

### 4.2. 2D Helmholtz equation

We extend our demonstration example into the 2D physical space, and solve the 2D Helmholtz equation:

$$-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u + \lambda^2 u = f(x,y), \quad (x,y) \in [0,1.5] \times [0,1]. \tag{29}$$

In this section, we also demonstrate the useful information that can be extracted from a network of *multi-fidelity* sensors.

#### 4.2.1. Case 1

We enforce a homogeneous Dirichlet boundary condition for the global domain $\mathcal{D}$:

$$u(1.5,y) = u(0,y) = u(x,0) = u(x,1) = 0. \tag{30}$$

The global domain $\mathcal{D}$ is divided into two non-overlapping subdomains as depicted in Fig. 9, while the PDE-domain $\mathcal{D}_1$ serves as the *dominus* domain, equipped with a Dirichlet interface condition at the interface $\Gamma_s$, and the Data-domain $\mathcal{D}_2$ is the *servus* domain with a Neumann interface condition. The manufactured reference solution is:

$$u(x,y) = \sin\left(\frac{4}{3}\pi x\right) \sin(\pi y), \tag{31}$$

and by choosing $\lambda = 1$,

$$f(x,y) = \left(\lambda^2 + \frac{25}{9}\pi^2\right) \sin\left(\frac{4}{3}\pi x\right) \sin(\pi y). \tag{32}$$

As shown in Fig. 10a, the $u$ sensors (green dots) are uniformly distributed in the Data-domain $\mathcal{D}_2$ to form a $10 \times 5$ lattice grid and the Neumann interface condition is sampled at the red diamonds. We use a relaxation parameter $\eta = 0.3$ and a stopping threshold $\epsilon = 10^{-5}$ for the Schwarz iterations. A visualization of the GPDD solution generated by noiseless sensors is displayed in Fig. 10a. Fig. 10b shows that the relative $L_2$ error decays after a few iterations, where less noisy sensors lead to more accurate predicted solutions.

#### 4.2.2. Case 2 with multi-fidelity data

We test our GPDD algorithm under the situation where we have two sources of $f$ sensors data of different fidelity, using a new 2D domain decomposition paradigm displayed in Fig. 11. For this test we enforce a homogeneous Neumann boundary condition on the global domain $\mathcal{D} := [-1,1] \times [-1,1]$, i.e.

$$u_x(1.5,y) = u_x(0,y) = u_y(x,0) = u_y(x,1) = 0. \tag{33}$$

The manufactured reference solution is:

$$u(x,y) = \sin\left(\frac{3\pi x}{2}\right)(2 + \cos(\pi y)). \tag{34}$$

As for the forcing term, the high-fidelity function is

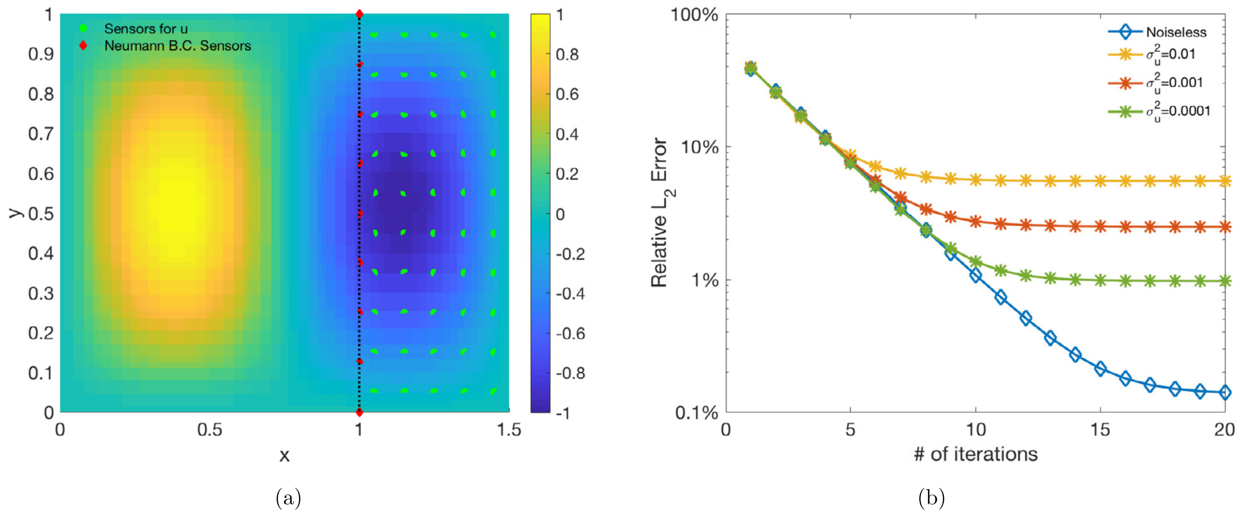


**Fig. 10.** Case 1: (a) The GPDD solution of the 2D Helmholtz equation using 50 $u$ sensors (marked in green). The sample points for the Neumann boundary conditions are marked in red. The black dotted line indicates the subdomain interface. (b) The relative $L_2$ error of solution in the entire domain versus the number of iterations.

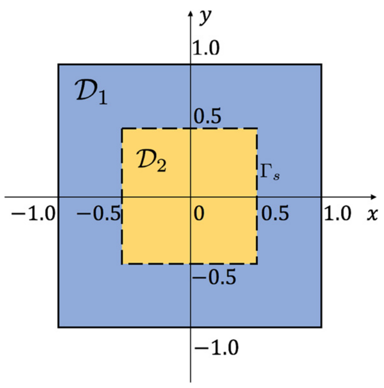


**Fig. 11.** The 2D physical domain and its decomposition for case 2: *servus* domain: $\mathcal{D}_2 := [-0.5, 0.5] \times [-0.5, 0.5]$, *dominus* domain: $\mathcal{D}_1 := [-1, 1] \times [-1, 1] \setminus \mathcal{D}_2$.

$$f^h(x, y) = \left(\left(1 + \frac{9\pi^2}{4}\right)(2 + \cos(\pi y)) + \pi^2 \cos(\pi y)\right) \sin\left(\frac{3\pi x}{2}\right), \tag{35}$$

while the low-fidelity function is chosen to be a scaling of the high-fidelity function combined with a non-trivial noise:

$$f^l(x, y) = \frac{4}{5} f(x, y) + \frac{2\pi x}{15} \sin(\pi y). \tag{36}$$

In order to evaluate the effectiveness of our method with multi-fidelity data, we compare the error of the GPDD solutions when different numbers of low-fidelity and high-fidelity $f$ sensors are placed in the Data-domain. For the first experiment, we fix the number and position of the high-fidelity sensors and investigate the effect of using an increase number of low-fidelity sensors. Positions of the sensors are illustrated in Fig. 12, where we intentionally choose a nested setup of the low-fidelity sensors so that the information from the previous tests is kept intact in the succeeding tests. Since the Data-domain is fed only with information of the derivatives of $u$, the predicted solution can vary by any constant. Therefore in addition to the sensors for $f$, we place 4 anchor points for $u$ in the Data-domain to pin the solution. The relaxation parameter $\eta$ is set to be 0.2. To avoid the influence of the initialization of hyper-parameters, in practice, we conduct 50 independent runs for each setup with randomly initialized hyper-parameters, and the errors are calculated using an average of the 15 runs with minimum $\mathcal{NLML}$s.

Fig. 13b indicates the decay of relative $L_2$ error in our predicted solution with different sensor setups. We compare in Fig. 13c the error of the numerical solutions after the iteration converges, as the number of low-fidelity sensors increases, the error shows a decreasing trend at first. However, when we continue adding more low-fidelity sensors, the error starts increasing; this is an indication that we reached the point of “diminishing return” from the low-fidelity sensors. Instead, for

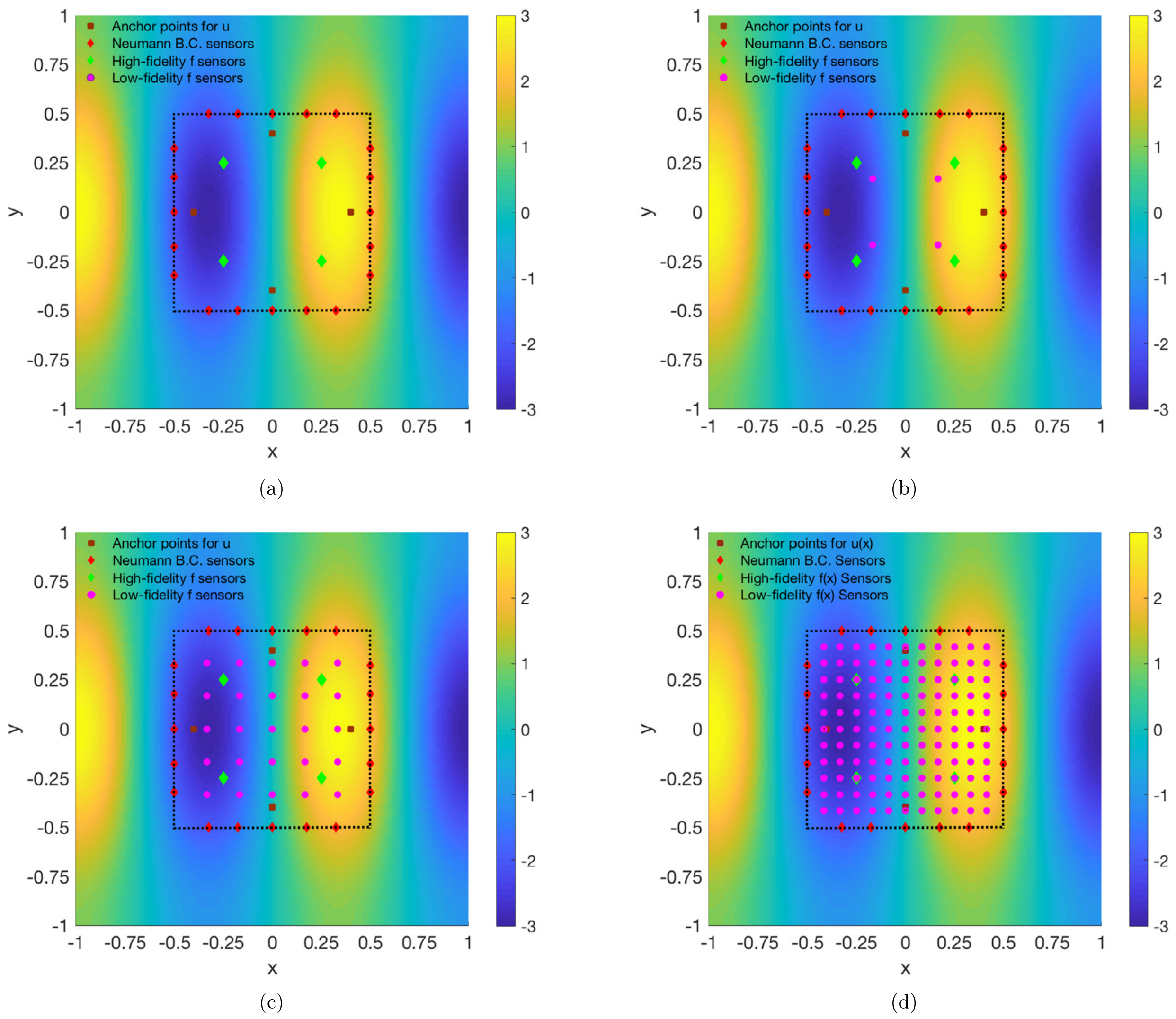


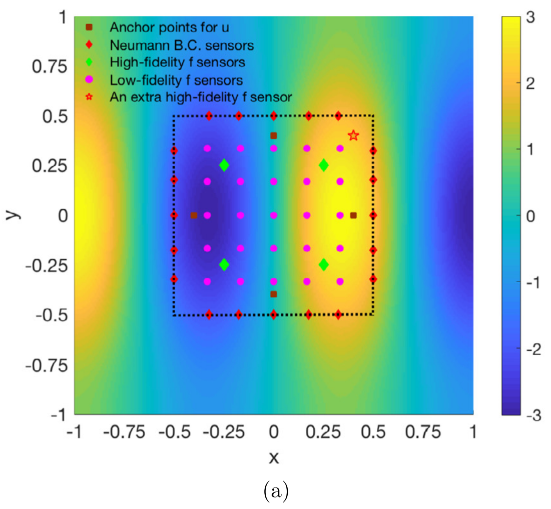
**Fig. 12.** Positions of the high-fidelity and low-fidelity sensors. The number of high-fidelity sensors is fixed at 4 while the number of low-fidelity sensors varies: 0, 4, 25, 121. The black dotted line indicates the subdomain interface.

the second experiment, we put just another high-fidelity sensor (Fig. 13a), the error could be further decreased (indicated by the yellow triangle in Fig. 13c. The general strategies of selecting a proper number and the positions of high-fidelity and low-fidelity sensors are important issues and are related to active learning so we plan to investigate it systematically in the future work.

### 4.3. Computational cost

Here we provide a rough estimation of the computational cost in the Data-domain. Note that in each iteration, $\mathcal{NLML}$ is minimized and a prediction is conducted in the Data-domain. During each minimization, $\mathcal{NLML}$ is repeatedly evaluated, and we denote the number of evaluations as $E$. Assume $N$ to be the total number of sensors and boundary condition points in the Data-domain, then the size of the covariance matrix used for computing $\mathcal{NLML}$ will be $N$ by $N$. As a consequence, during each evaluation of $\mathcal{NLML}$, the computational cost of generating the covariance matrix is $O(N^2)$, and the computational cost of calculating $\mathcal{NLML}$ with the covariance matrix is $O(N^3)$ if we use Cholesky decomposition to invert the covariance matrix. The covariance matrix and its inverse could be reused in the prediction stage. The total computational cost for each $\mathcal{NLML}$ evaluation is $O(N^3)$, and hence the total computational cost for each minimization is $O(EN^3)$.

Suppose that we need to make predictions at $M$ points in the Data-domain. During the iterations, we only need to make predictions on the Neumann boundary condition points for the sake of information fusion. The computational cost of



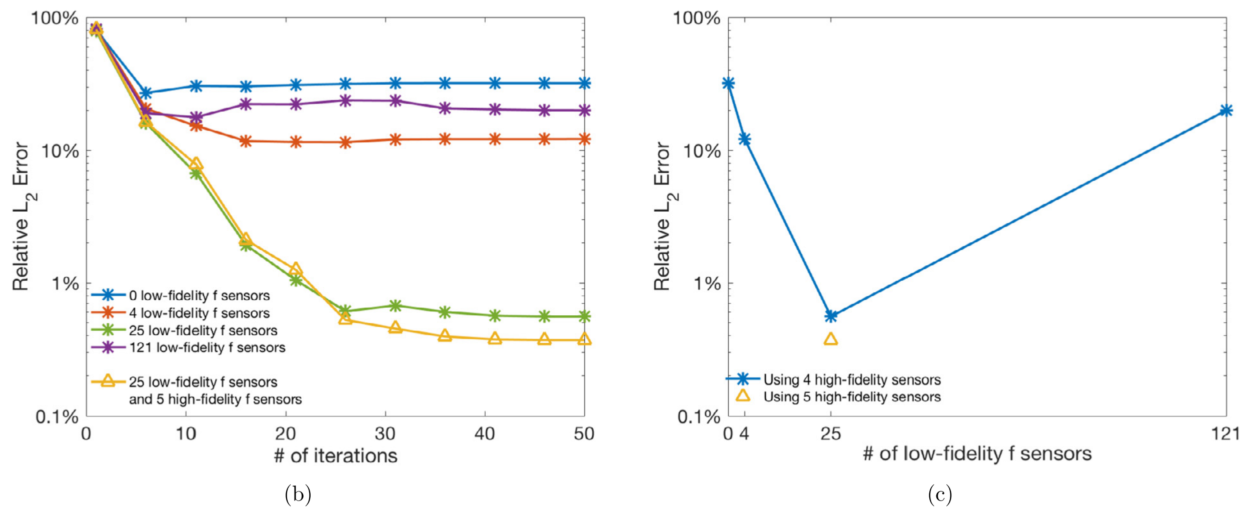


**Fig. 13.** (a) An illustration of adding an extra high-fidelity sensor (5 high-fidelity sensors in total) to the 25 low-fidelity sensors setup. (b) The relative $L_2$ error of solution decays after a few Schwarz iterations and remains at a stable level, showing convergence. (c) Error of the numerical solution at the end of the iteration versus the number of low-fidelity sensors. Using an extra high-fidelity sensor improves the accuracy.

generating covariance matrices, i.e., $g$ and $\boldsymbol{a}$ in Equation (16), is $O(MN) + O(M^2)$, and the computational cost of making predictions with the matrices is $O(MN^2) + O(M^2N)$. Therefore, the cost of making predictions is $O(MN^2) + O(M^2N)$.

We conclude that, roughly the computational cost in the Data-domain in each iteration is $O(EN^3) + O(MN^2) + O(M^2N)$.

## 5. Summary

In this paper we address the issue of coupling a solution in two types of domains, one with a traditional PDE solver, and the other one with sparse sensor data. We proposed a GPDD algorithm where the PDE-domain and the Data-domain are synchronized by the Schwarz type iterative method that can propagate information across the subdomain interface in both directions. The uncertainty in the GP prediction is spread and results in a distribution of the predicted global solution. The PDE-domain acts as the *dominus* domain where we impose a Dirichlet boundary condition at the interface, while the Data-domain acts as the *servus* domain where numerical GPR is used to infer the solution subject to the Neumann interface condition. The sensor data in the Data-domain can be either noiseless (exact) or noisy (with measurement error). Two specific situations were considered:

1. We have sensor data of the QoI in the Data-domain but we do not have a governing PDE. In this situation, GPR is performed in the Data-domain.

2. We have sensor data of the forcing term and we also know the governing PDE in the Data-domain. In this situation, we build a joint distribution of the solution and the forcing term. The solution in the Data-domain can be inferred using the numerical GPR.

The GPDD algorithm is proved to be reliable for solving linear equations in both 1D and 2D physical domains. The iterative process helps with the training of GP, as the error in solution and the variance of prediction decays fast after just a few iterations, which is a non-trivial result. We also observed that by using noiseless sensors and by using larger amount of sensors, we obtain more accurate and more trustworthy results. Moreover, multi-fidelity sensors could be incorporated with the GPDD framework. A combination of cheap low-fidelity sensors and expensive high-fidelity sensors can contribute to better solutions, which is of great significance in practice, especially because in most applications one has to operate at limited budget and resources.

There are several open questions and challenges related to the GPDD algorithm. For example, measurements in practice could be collected at a scale distant from that of the PDE model, and the small-scale behavior of the QoI would be extremely difficult to resolve and would be easily attributed to the sensor's noise without careful treatment. To deal with this situation, a sufficiently large number of sensors should be employed and we should also use the non-stationary GP kernels because they are more adapted to the locally small-scale changes. Moreover, this data-driven domain decomposition method could be integrated with the non-linear information fusion algorithm [24] and time-dependent non-linear GPR algorithm [18] to learn the complex space and time dependent cross-correlations in multi-fidelity data sets, and to safeguard our computations against erroneous data or the low-fidelity models that may provide wrong trends. Those are indeed very interesting research topics for future investigation.

## Acknowledgements

We gratefully acknowledge support from the ARO grant W911NF-18-1-0301, the NSF grant DMS-1736088 and the UTAH-ARL. We would also like to thank Dr. Raissi and Dr. Perdikaris for their sample code and guidance.

## References

[1] C. Canuto, D. Funaro, The Schwarz algorithm for spectral methods, SIAM J. Numer. Anal. 25 (1988) 24–40.
[2] D. Funaro, A. Quarteroni, P. Zanolli, An iterative procedure with interface relaxation for domain decomposition methods, SIAM J. Numer. Anal. 25 (1988) 1213–1236.
[3] P.-L. Lions, On the Schwarz alternating method. I, in: First International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Paris, France, 1988, pp. 1–42.
[4] B. Smith, P. Bjorstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, 2004.
[5] R. Henderson, G.E. Karniadakis, Hybrid spectral-element-low-order methods for incompressible flows, J. Sci. Comput. 6 (1991) 79–100.
[6] G.E. Karniadakis, S. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics, Oxford University Press, 2013.
[7] H.A. Schwarz, Uber einige Abbildungsaufgaben, J. Reine Angew. Math. 70 (1869) 105–120.
[8] Y. Chen, J. Jakeman, C. Gittelson, D. Xiu, Local polynomial chaos expansion for linear differential equations with high dimensional random inputs, SIAM J. Sci. Comput. 37 (2015) A79–A102.
[9] Q. Liao, K. Willcox, A domain decomposition approach for uncertainty analysis, SIAM J. Sci. Comput. 37 (2015) A103–A133.
[10] H. Cho, X. Yang, D. Venturi, G.E. Karniadakis, Algorithms for propagating uncertainty across heterogeneous domains, SIAM J. Sci. Comput. 37 (2015) A3030–A3054.
[11] D. Zhang, H. Babaee, G.E. Karniadakis, Stochastic domain decomposition via moment minimization, SIAM J. Sci. Comput. 40 (2018) A2152–A2173, https://doi.org/10.1137/17M1160756.
[12] T. Graepel, Solving noisy linear operator equations by Gaussian processes: application to ordinary and partial differential equations, in: ICML'03, AAAI Press, 2003, pp. 234–241.
[13] S. Särkkä, Linear operators and stochastic partial differential equations in Gaussian process regression, in: Artificial Neural Networks and Machine Learning, ICANN 2011, 2011, pp. 151–158.
[14] I. Bilionis, Probabilistic solvers for partial differential equations, arXiv preprint, arXiv:1607.03526, 2016.
[15] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, vol. 1, MIT Press, Cambridge, 2006.
[16] M. Raissi, P. Perdikaris, G.E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, J. Comput. Phys. 335 (2017) 736–746.
[17] M.J. Gander, Optimized Schwarz methods, SIAM J. Numer. Anal. 44 (2006) 699–731.
[18] M. Raissi, P. Perdikaris, G.E. Karniadakis, Numerical Gaussian processes for time-dependent and nonlinear partial differential equations, SIAM J. Sci. Comput. 40 (2018) A172–A198.
[19] G. Pang, L. Yang, G.E. Karniadakis, Approximation and PDE solution Neural-net-induced Gaussian process regression for function approximation and PDE solution, arXiv preprint, arXiv:1806.11187, 2018.
[20] M. Kennedy, A. O'Hagan, Predicting the output from a complex computer code when fast approximations are available, Biometrika 87 (2000) 1–13.
[21] P. Perdikaris, D. Venturi, G.E. Karniadakis, Multifidelity information fusion algorithms for high-dimensional systems and massive data sets, SIAM J. Sci. Comput. 38 (2016) B521–B538.
[22] L.L. Gratiet, J. Garnier, Recursive co-kriging model for design of computer experiments with multiple levels of fidelity, Int. J. Uncertain. Quantificat. 4 (2014) 365–386.
[23] D.A. Kopriva, Spectral Element Methods, Springer Netherlands, Dordrecht, 2009, pp. 293–354.
[24] P. Perdikaris, M. Raissi, A. Damianou, N.D. Lawrence, G.E. Karniadakis, Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling, Proc. R. Soc. Lond., Ser. A, Math. Phys. Eng. Sci. 473 (2017).