Computers and Structures 92–93 (2012) 206–215

Contents lists available at SciVerse ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

Bayesian assimilation of multi-fidelity finite element models

F.A. DiazDelaO *[,1], S. Adhikari [2]

Civil and Computational Engineering Centre, College of Engineering, Swansea University, Singleton Park, Swansea, SA2 8PP, United Kingdom

ARTICLE INFO

Article history:
Received 10 September 2010
Accepted 7 November 2011
Available online 9 December 2011

Keywords:
Metamodel
Gaussian process emulator
Multi-fidelity
Domain decomposition methods
Finite element method

ABSTRACT

A complex system can be modeled using various fidelities with the finite element method. A high-fidelity model is expected to be more computationally expensive compared to a low-fidelity model and in general may contain more degrees of freedom and more elements. This paper proposes a novel multi-fidelity approach to solve boundary value problems using the finite element method. A Bayesian approach based on Gaussian process emulators in conjunction with the domain decomposition method is developed. Using this approach one can seamlessly assimilate a low-fidelity model with a more expensive high-fidelity model. The idea is illustrated using elliptic boundary value problems.

1. Introduction

The size of the finite element models has increased significantly over the past decade. For example, in the automotive industry, 2 to 5 million degrees-of-freedom models are quite common nowadays. Such high-resolution models, combined with detailed physics can give good fidelity to experimental results. However, a potential disadvantage is that such large models may be computationally expensive. One alternative to address this problem is to use a low-resolution model. Although such low-resolution models are often used during the design iteration, they are likely to be low-fidelity and may miss some crucial physics. The motivation of this paper is therefore to investigate the possibility of improving the fidelity of a low-fidelity model without completely solving a detailed high-fidelity problem.

A common tool when solving expensive finite element models is the domain decomposition method [1–7], a divide-and-conquer algorithm aimed at solving partial differential equations (PDEs). Its main feature is that a linear system of discretised PDEs is recast as a set of smaller linear systems to be solved separately. A finite element model can thus be parallelized by partitioning the domain Ω in a number of subdomains. This allows an increase in the resolution of the model, along with a reduction in CPU requirements. There is, however, a potential disadvantage with this approach, since in order to obtain the finite element solution for each subdomain, the governing PDEs must be solved in the interface of each pair of subdomains. This paper presents a metamodeling approach, known as *Gaussian process emulation*, in order to approximate the solution to the interface problem. Broadly speaking, a Gaussian process emulator works by treating a set of *training runs* as data, which is in turn used to update some prior beliefs about the output of an expensive *simulator* such as the interface problem. These beliefs are represented by a Gaussian stochastic process prior distribution. As noted by some authors [8], the choice of a Gaussian process is made for much the same reasons that the Gaussian distribution repeatedly appears in statistics: it is analytically tractable, flexible, and quite often realistic. After conditioning on the training runs and updating the prior distribution, the mean of the resulting posterior distribution approximates the output of the simulator at any untried node in the input domain of the interface problem, whereas it reproduces the known output at each input belonging to the training runs. The idea of using a stochastic process to solve domain decomposition problems has been employed in the past [9]. However, the implementation of Gaussian process emulators is relatively simple and flexible. In addition to this, Gaussian process emulators have already been implemented in various scientific fields in order to alleviate the computational burden of expensive simulators with encouraging results. These fields include climate prediction [10,11], environmental science [12], medicine [13–15], structural dynamics [16,17], reservoir forecasting [18], hydrogeology [19], quality control [20], heat transfer [21], and reliability analysis [22], among others.

The paper is organized as follows. Section 2 introduces the concepts of mutli-fidelity modeling in the context of finite element method. A brief overview of domain decomposition and

* Corresponding author. Tel.: +44 01792 602088; fax: +44 01792 295676.
E-mail addresses: f.a.diazdelao@swansea.ac.uk (F.A. DiazDelaO), s.adhikari@swansea.ac.uk (S. Adhikari).
URLs: http://engweb.swan.ac.uk/~diazdelao/ (F.A. DiazDelaO), http://engweb.swan.ac.uk/~adhikaris/ (S. Adhikari).
[1] Project officer at ASTUTE (http://www.astutewales.com/).
[2] Professor of Aerospace Engineering.

0045-7949/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compstruc.2011.11.002

metamodeling is given in Section 3. Section 4 discusses the theory behind the Gaussian process emulation. Section 5 explains the proposed coupling between domain decomposition and Gaussian process emulators. Section 6 presents some numerical results of the proposed method applied to the deformation of a membrane on a domain with different geometries. Section 7 offers some conclusions based on the results obtained in the paper.

2. Multi-fidelity finite element modeling

Let Ω be a bounded domain in $\mathbb{R}^2$ with Lipschitz continuous boundary $\partial\Omega$, that is, there exist a finite number of covering open sets $\mathcal{O}_\ell$ such that, for every $\ell, \partial\Omega \cap \mathcal{O}_\ell$ is the graph of a Lipschitz continuous function and $\Omega \cap \mathcal{O}_\ell$ lies on one side of this graph. Let $\mathcal{T}^h$ be a family of conforming meshes (triangles) which are shape-regular as the mesh size $h \to 0$. Consider the elliptic PDE with the following Dirichlet boundary condition

$$\begin{aligned} &-\nabla[\alpha(\mathbf{x})\nabla u(\mathbf{x})] + \beta(\mathbf{x})u(\mathbf{x}) = \phi(\mathbf{x}); \quad \mathbf{x} \in \Omega \\ &u(\mathbf{x}) = 0; \quad \mathbf{x} \in \partial\Omega \end{aligned} \tag{1}$$

The Hilbert space $L^2(\Omega)$ and Sobolev space $H^k(\Omega)$ are respectively endowed with inner products $(u,v) = \int_\Omega u(\mathbf{x})v(\mathbf{x})d\mathbf{x}$ and $(u,v)_k = \int_\Omega u(\mathbf{x})v(\mathbf{x})d\mathbf{x} + \int_\Omega \left(\frac{du}{d\mathbf{x}}\right)\left(\frac{dv}{d\mathbf{x}}\right)d\mathbf{x} + \cdots + \int_\Omega \left(\frac{d^k u}{d\mathbf{x}^k}\right)\left(\frac{d^k v}{d\mathbf{x}^k}\right)d\mathbf{x}$. The aim is to obtain the function $u : \Omega \to \mathbb{R}$ which satisfies the conditions of problem (1) for a given $\phi : \Omega \to \mathbb{R}$.

Applying the standard finite element method [23,24], the PDE in problem (1) can be expressed in the matrix form as

$$\mathcal{K}u = f \tag{2}$$

where $\mathcal{K} \in \mathbb{R}^{N\times N}$ is the stiffness matrix, $u \in \mathbb{R}^N$ is the displacement vector and $f \in \mathbb{R}^N$ is the forcing vector. Note that N is the number of degrees of freedom in the underlying finite element mesh.

Suppose L_f and H_f are two finite element models to solve problem (1). Let $n_e^{(L_f)}$ and $n_e^{(H_f)}$ denote the number of elements in the finite element meshes of L_f and H_f induced by Eq. (2) and let $N^{(L_f)}$ and $N^{(H_f)}$ be the number of degrees of freedom. Let $h^{(L_f)}$ and $h^{(H_f)}$ be the respective element size. Also, let $u^*(\mathbf{x})$ and $U^*(\mathbf{x})$ be a solution to L_f and H_f respectively, and let $u^{(r)}$ be the exact solution to (1). That way, $\|u^{(r)}(\mathbf{x}) - u^*(\mathbf{x})\|$ and $\|u^{(r)}(\mathbf{x}) - U^*(\mathbf{x})\|$ are the differences between the exact solution and the solution to the models L_f and H_f, with $\|\cdot\|$ being the Euclidean norm. We call L_f a low-fidelity model and H_f a high-fidelity model if the following inequalities hold:

1. $\|u^{(r)}(\mathbf{x}) - u^*(\mathbf{x})\| > \|u^{(r)}(\mathbf{x}) - U^*(\mathbf{x})\|$ (Accuracy)
2. $h^{(L_f)} > h^{(H_f)}$ (Resolution)
3. $N^{(L_f)} < N^{(H_f)}$ (number of degrees of freedom)
4. $n_e^{(L_f)} < n_e^{(H_f)}$ (number of elements)

It is important to note that the concepts of low and high fidelity based on the above definition are relative. A single refinement of a given low-fidelity mesh would imply a different increase in the fidelity of the model depending on the particular characteristics of the problem at hand. A more accurate description of L_f and H_f would therefore be "lower" and "higher" fidelity models respectively. Keeping this note in mind, the current low and high fidelity terminology is kept in the remainder of the paper. Also, note that in the above definition it is implicitly assumed that both models L_f and H_f have same polynomial order p. A general hp finite element model is beyond the scope of this paper (see for example [25]). Fig. 1 shows two finite element models on a D-shaped domain, each with a different fidelity level.

3. A brief overview of domain decomposition and metamodeling

3.1. The domain decomposition method

Let Ω be partitioned in S subdomains $\{\Omega_j : 1 \leqslant j \leqslant S\}$, such that $\overline{\Omega} = \bigcup_j \overline{\Omega}_j$. Suppose these domains are non-overlapping, that is $\Omega_j \bigcap \Omega_k = \emptyset$, $\forall j \neq k$. The interface is denoted by Γ, where

$$\Gamma = \bigcup_{i,j} (\partial\Omega_i \cap \partial\Omega_j) \setminus \partial\Omega \tag{3}$$

In Fig. 2, the D-shaped domain Ω from Fig. 1 is partitioned into subdomains Ω_1 and Ω_2. The interface Γ separates both subdomains. Fig. 3 shows the finite element mesh of the partitioned low-fidelity model L_f and the partitioned high-fidelity model H_f. In order to obtain the solution to these finite element models, it can be shown [1] that if Ω is the disjoint union of the subdomains $\Omega_1, \ldots, \Omega_S$, then the discretised PDEs governing the system's response can be recast as the following partitioned linear system

$$\begin{pmatrix} \mathcal{K}_1 & 0 & \cdots & 0 & \mathcal{B}_1^{\mathrm{T}} \\ 0 & \mathcal{K}_2 & \cdots & 0 & \mathcal{B}_2^{\mathrm{T}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \mathcal{K}_n & \mathcal{B}_S^{\mathrm{T}} \\ \mathcal{B}_1 & \mathcal{B}_2 & \cdots & \mathcal{B}_S & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \\ u_c \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_S \\ f_c \end{pmatrix} \tag{4}$$

The solution of the partitioned linear system (4) is obtained by solving the interface problem

$$\begin{aligned} &\left(C - \mathcal{B}_1\mathcal{K}_1^{-1}\mathcal{B}_1^{\mathrm{T}} - \cdots - \mathcal{B}_S\mathcal{K}_S^{-1}\mathcal{B}_S^{\mathrm{T}}\right)u_c \\ &\quad = f_c - \mathcal{B}_1\mathcal{K}_1^{-1}f_1 - \cdots - \mathcal{B}_S\mathcal{K}_S^{-1}f_S \end{aligned} \tag{5}$$

and then solving in parallel

$$\begin{aligned} u_1 &= \mathcal{K}_1^{-1}\left(f_1 - \mathcal{B}_1^{\mathrm{T}}u_c\right) \\ &\vdots \\ u_S &= \mathcal{K}_S^{-1}\left(f_S - \mathcal{B}_S^{\mathrm{T}}u_c\right) \end{aligned} \tag{6}$$

Fig. 4 shows the domain decomposition solution of problem (1) for H_f and for the values $\alpha(\mathbf{x}) = 1$, $\beta(\mathbf{x}) = 0$, and $\phi = 1$. The model represents the deformation of a membrane in the domain Ω.

The main problem with solving a finite element model using domain decomposition is that the Schur complement matrix

$$\Sigma = C - \mathcal{B}_1\mathcal{K}_1^{-1}\mathcal{B}_1^{\mathrm{T}} - \cdots - \mathcal{B}_S\mathcal{K}_S^{-1}\mathcal{B}_S^{\mathrm{T}} \tag{7}$$

is numerically expensive to obtain. Hence, solving the linear system (5) is likely to become a bottleneck of the domain decomposition strategy for a high-fidelity model H_f. A metamodeling approach is therefore proposed, whereby the solution to the interface problem (5) is approximated using only a few evaluations of a lower-fidelity model L_f.

3.2. Metamodeling approach

Let $u^*(\mathbf{x}) : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ be a finite element solution to problem (1). If $\mathbf{x} = (x,y)$ and $u^*(\mathbf{x}) = z$, then u^* is a function that maps $(x,y) \mapsto z$. Adopting this notation, a level set of the domain Ω with respect to the x-axis is defined as

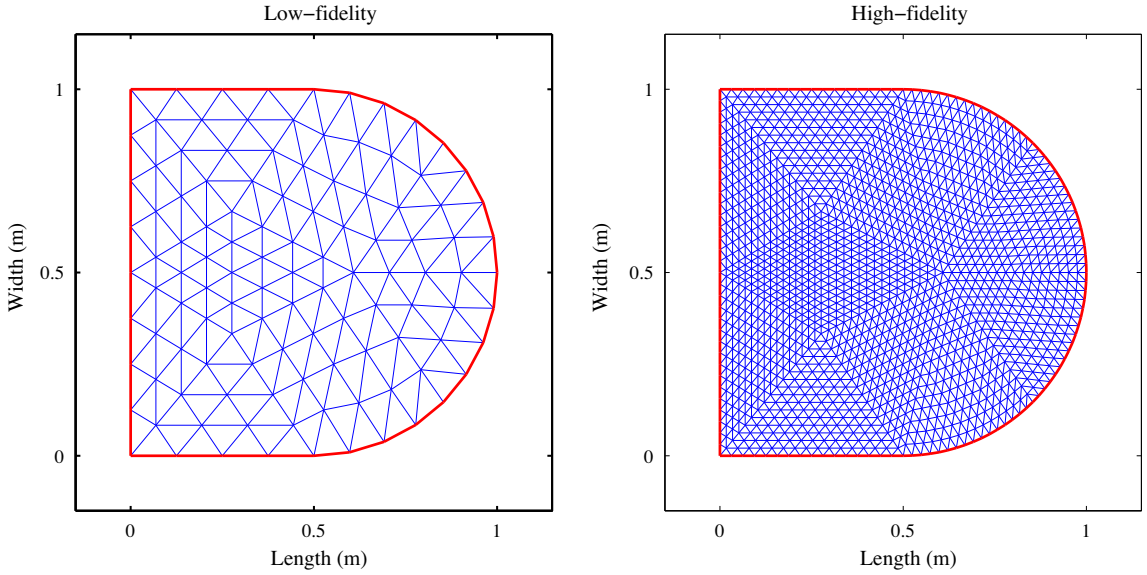
$$\mathcal{L}^x(c) = \{z | u^*(c,y) = z; c \in \mathbb{R}\} \tag{8}$$

and analogously for a level set with respect to the y-axis

$$\mathcal{L}^y(d) = \{z | u^*(x,d) = z; d \in \mathbb{R}\} \tag{9}$$

For $n_x, n_y \in \mathbb{Z}^+$, let Λ denote

$$\Lambda = \{\mathcal{L}^x(c_i), \mathcal{L}^y(d_j) | 1 \leqslant i \leqslant n_x, 1 \leqslant j \leqslant n_y\} \tag{10}$$



(a) Low-fidelity model L_f (b) High-fidelity model H_f

Fig. 1. Low and high-fidelity finite element models on the domain Ω.

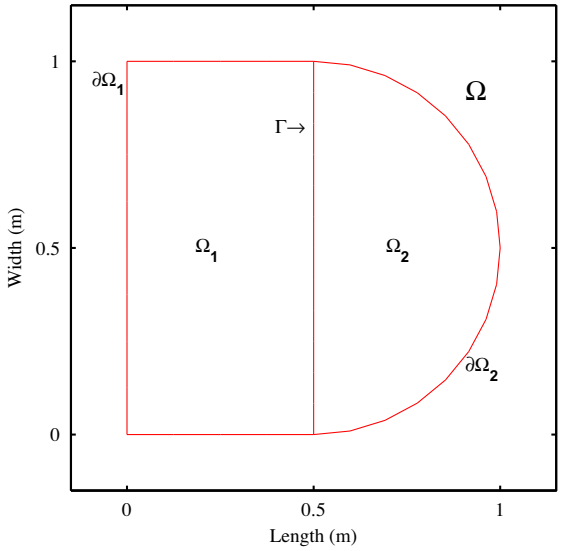


Fig. 2. Domain decomposition of Ω into the subdomains Ω_1 and Ω_2. The borders of each subdomain are $\partial\Omega_1$ and $\partial\Omega_2$ respectively. The interface Γ separates the subdomains.

that is, the family of level sets with respect to x and with respect to y on Ω, determined by every number in $\mathcal{C} = \{c_i, d_j \in \mathbb{R} | 1 \leqslant i \leqslant n_x, 1 \leqslant j \leqslant n_y\}$. Let $\Gamma^x(c_i)$ and $\Gamma^y(d_j)$ be the preimages of every $\mathcal{L}^x(c_i), \mathcal{L}^y(d_j) \in \Lambda$, that is

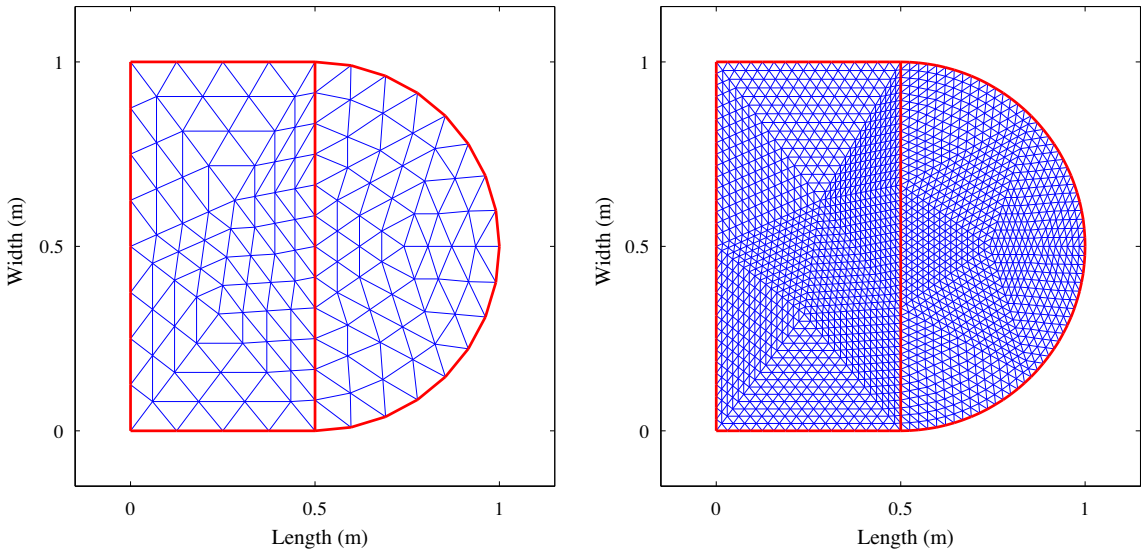
$$\Gamma^x(c_i) = \{y \in \mathbb{R} | u^*(c_i, y) = z\} \tag{11}$$

$$\Gamma^y(d_j) = \{x \in \mathbb{R} | u^*(x, d_j) = z\} \tag{12}$$

Notice that $u^*(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}$, whilst $\Gamma^x(c_i) \mapsto \mathcal{L}^x(c_i)$ and $\Gamma^y(d_j) \mapsto \mathcal{L}^y(d_j)$ are both mappings of the form $\eta_i^x : \mathbb{R} \rightarrow \mathbb{R}$ and $\eta_j^y : \mathbb{R} \rightarrow \mathbb{R}$. Moreover, for every $c_i, d_j \in \mathcal{C}$, the set of points defined by $\bigcup_{ij}\{\mathcal{L}^x(c_i) \bigcup \mathcal{L}^y(d_j)\}$ is the solution of the two-dimensional interface problem (5) whenever the interface between any two subdomains is parallel to any of the axis x and/or y. As mentioned before, this solution is the image of the mappings in $\mathcal{H} = \left\{\eta_i^x(\cdot), \eta_j^y(\cdot) \middle| 1 \leqslant i \leqslant n_x, 1 \leqslant j \leqslant n_y\right\}$. Note that the requirement of the interfaces being parallel to the coordinate axes is not a necessary condition: they could all be defined as rotations of elements in $\mathcal{L}^x(c_i)$. The definitions followed here are given without loss of generality and help illustrate the methodology proposed. Additionally, the requirement of the interfaces being straight lines might seem restrictive, as it is not always the case that a domain can be decomposed in this way. However, there exist cases for which prescribed straight line interfaces can be meaningful for modelling physical phenomena. In particular, they can be useful for solving contact problems [35,37,36], whereby two bodies Ω_1 and Ω_2 are bonded along their interface by a thin adhesive layer. Additionally, there are several examples in the literature in which theoretical work has been done assuming straight line interfaces [40,39] or segments of straight line interfaces [38]. There also exist domain decomposition methods for parabolic PDEs in which the interfaces are parallel lines. To overcome the disadvantage of data communication, some perturbations to the interfaces can be introduced [41]. However, they still contain segments of straight lines in which the proposed method could be applicable.

In order to reduce the computational cost of the interface problem, this paper proposes a statistical approximation to the mappings in $\mathcal{H}$ using Gaussian process emulators. The main idea is that the level sets defined by every interface problem in a low-fidelity finite element model can be emulated in order to approximate the interface problem in a corresponding high-fidelity finite element model. Without loss of generality, suppose that n *design points* $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are chosen in the input domain of any mapping $\eta(\cdot) \in \mathcal{H}$ (referred to as a simulator in the remainder of the paper). The set $\{\eta(\mathbf{x}_1), \ldots, \eta(\mathbf{x}_n)\}$, resulting from the evaluation of $\eta(\cdot)$ in each of the design points, is called *training set*. Following [26], a Gaussian process emulator should satisfy some minimal criteria:

1. Since by definition the output at each design point is known, the emulator should reproduce this output with no uncertainty.
2. At any $\mathbf{x}$ that is not a design point, the probability distribution provided by the emulator should produce a mean value that constitutes a plausible interpolation/extrapolation of the training data. The probability distribution around this predictive mean should also express the uncertainty about how the emulator might interpolate/extrapolate.



(a) Low-fidelity model L_f (b) High-fidelity model H_f

Fig. 3. The finite element mesh of the low and high-fidelity finite element models on the domain Ω, which is decomposed into the subdomains Ω_1 and Ω_2.

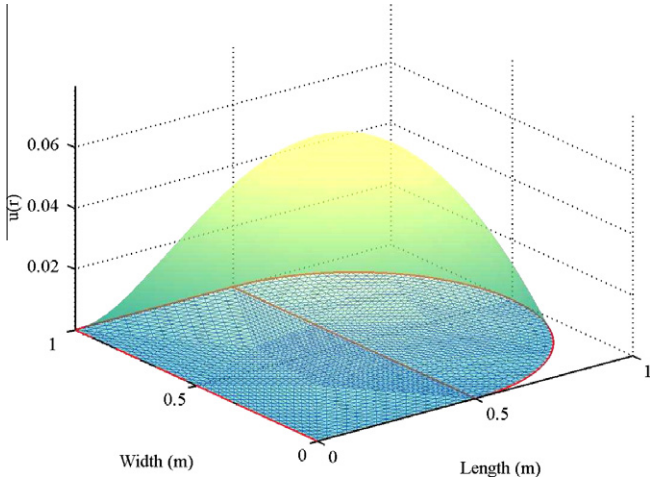


Fig. 4. Domain decomposition solution of the high-fidelity finite element model H_f.

In addition to these criteria, the key feature of a Gaussian process emulator is that it quantifies the uncertainty that arises from having a training set with limited size [27].

To illustrate the above criteria, take a one-dimensional simulator and suppose it is computationally intensive. Fig. 5(a) depicts the case when eight training runs (the circles) are used. The mean of the distribution provided by the emulator (the dots) approximates the output of the simulator (the solid line) at several untried inputs across the input domain. As required, it returns the known value of the simulator at each training run. Fig. 5(b) shows upper and lower bounds of two standard deviations for the predictive mean of the emulator. Note how uncertainty is equal to zero in each of the training runs, as it would be expected, since the emulator reproduces output of the simulator at these points.

4. Brief overview of Gaussian process emulators

Let $\eta(\cdot)$ be an expensive simulator, such that the cost of running $\eta(\cdot)$ makes it is practical to evaluate it only at a limited number of points in its domain. The uncertainty about the output of $\eta(\cdot)$ can be described through a probability distribution [28]. A Bayesian treatment is followed here, whereby prior beliefs about the relationship between the input and the unknown output are conditioned on a set of evaluations of $\eta(\cdot)$, thus combining subjective and objective information. Begin by assuming that $\eta(\cdot)$ admits the following stochastic representation

$$\eta(\cdot) = \mathbf{h}(\cdot)^{\mathrm{T}}\boldsymbol{\beta} + Z(\cdot) \tag{13}$$

where $\mathbf{h}(\cdot)$ is a vector of known functions and $\boldsymbol{\beta}$ is a vector of unknown coefficients. Some authors [14,29] note that $\mathbf{h}(\cdot)$ should be chosen to reflect the available information about the functional form of $\eta(\cdot)$. This is currently an area of active research. The function $Z(\cdot)$ is a stochastic process with mean zero and covariance function $Cov(\cdot,\cdot)$. Let $\mathbf{y} = [\eta(\mathbf{x}_1),\ldots,\eta(\mathbf{x}_n)]^{\mathrm{T}}$ be a vector of observations corresponding to the design points $\mathbf{x}_1,\ldots,\mathbf{x}_n$. These observations are used to update the prior distribution of the simulator. An analytically convenient choice for such prior is the following Gaussian process distribution [13]

$$\eta(\cdot)|\boldsymbol{\beta},\sigma^2 \sim \mathcal{N}(\mathbf{h}(\cdot)^{\mathrm{T}}\boldsymbol{\beta},\sigma^2 C(\cdot,\cdot)) \tag{14}$$

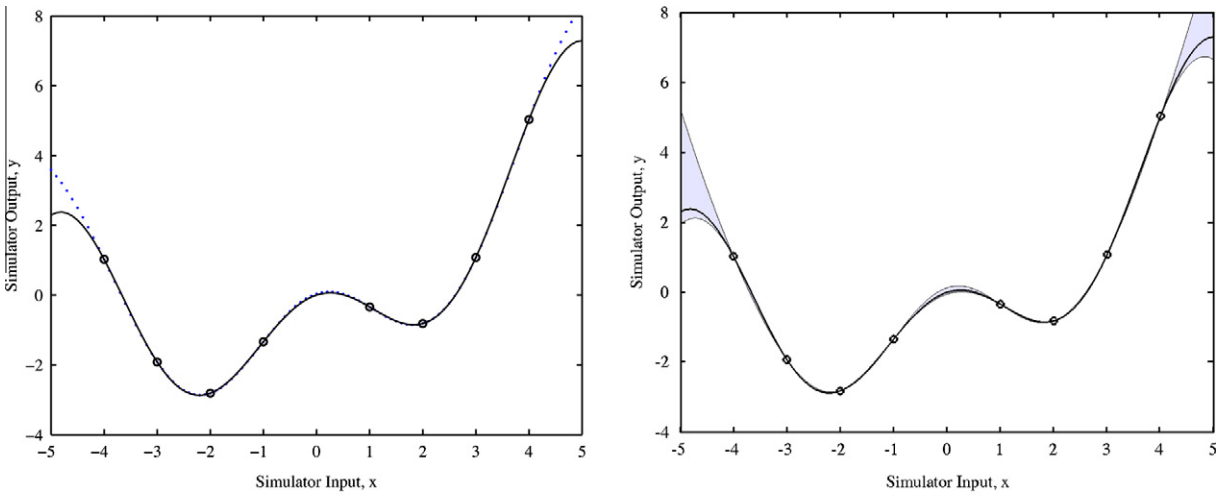
where $\mathbf{h}(\cdot)$ and $\boldsymbol{\beta}$ are defined as above.

Definition (*Gaussian stochastic process*). Let $\mathbf{x} \in \mathbb{R}^N$. Then $Z(\cdot)$ is a Gaussian stochastic process if for any $n \geqslant 1$ and any choice $\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$, the vector $[Z(\mathbf{x}_1),\ldots,Z(\mathbf{x}_n)]^{\mathrm{T}}$ has a multivariate normal distribution.

In order to choose a valid correlation function $C(\cdot,\cdot)$, some authors [30] consider correlation functions that are products of one dimensional correlations, specifically functions of the form

$$C(\mathbf{x},\mathbf{x}') = \prod_{i=1}^{n} \exp\left\{-b_i|\mathbf{x}_i - \mathbf{x}_i'|^{\gamma}\right\} \tag{15}$$

where $b_i > 0$ for all $i, \gamma \in \mathbb{R}_+$. One advantage of this type of correlation functions is that multidimensional integrals can be simplified as products of one-dimensional integrals. Moreover, if $\gamma = 2$, the correlation function is infinitely differentiable. This feature is convenient when Gaussian processes are used to model not only the simulator $\eta(\cdot)$, but also its derivatives [31]. For these reasons, a popular choice of correlation function is



(a) Predictive mean using 8 design points (b) Uncertainty about the predictive mean

Fig. 5. Approximation to the simulator $\mathbf{y} = 0.5\mathbf{x} - \mathbf{x}\sin(\mathbf{x})$ with the mean of a Gaussian process emulator and uncertainty (2 standard deviations) about this mean; (-): output of the simulator, (o): training runs, $(\cdots)$: emulator's predictive mean.

$$C(\mathbf{x},\mathbf{x}') = \exp\{-(\mathbf{x}-\mathbf{x}')^{\mathrm{T}}\mathbf{B}(\mathbf{x}-\mathbf{x}')\} \tag{16}$$

where $\mathbf{B}$ is a diagonal positive definite matrix. The diagonal of $\mathbf{B}$ is a vector $\mathbf{b} \in \mathbb{R}^n_+$ of smoothness parameters. These parameters quantify the rate at which the output varies as any $\mathbf{x}$ varies. One available technique to estimate the smoothness parameters is to derive the density function $f(\mathbf{B}|\mathbf{y})$ and obtain a maximum likelihood estimator of $\mathbf{b}$. It can be shown [32] that this density function is of the form

$$f(\mathbf{B}|\mathbf{y}) \propto (\hat{\sigma}^2)^{-\frac{(n-q)}{2}}|\mathbf{A}|^{-\frac{1}{2}}|\mathbf{H}^{\mathrm{T}}\mathbf{A}\mathbf{H}|^{-\frac{1}{2}} \tag{17}$$

where the terms σ^2, q, $\mathbf{A}$, and $\mathbf{H}$ are all defined below.

After conditioning on the training runs and updating the prior distribution (14), the mean of the resulting posterior distribution approximates the output of $\eta(\cdot)$ at any untried $\mathbf{x}$, whereas it reproduces the known output at each design point. The variance of the posterior distribution quantifies the uncertainty that arises from having only a limited number of evaluations of $\eta(\cdot)$ [11]. Very conveniently, the posterior is also a Gaussian process distribution, obtained as follows: Let $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1),\ldots,\mathbf{h}(\mathbf{x}_n)]^{\mathrm{T}}$, and $\mathbf{A} \in \mathbb{R}^{n\times n}$ such that $\mathbf{A}_{i\,j} = C(\mathbf{x}_i,\mathbf{x}_j)\forall i,j \in \{1,\ldots,n\}$. Then

$$\mathbf{y}|\boldsymbol{\beta},\sigma^2 \sim \mathcal{N}(\mathbf{H}\boldsymbol{\beta},\sigma^2\mathbf{A}) \tag{18}$$

To incorporate the information $\mathbf{y}$ and obtain the distribution of $\eta(\cdot)|\mathbf{y}$, use the following result [33].

Theorem 1. *Let $\mathbf{z} \in \mathbb{R}^N$ be a random vector such that $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu},\Sigma)$. Partition $\mathbf{z}$ as $(\mathbf{z}_1,\mathbf{z}_2)^T$, where $\mathbf{z}_1 \in \mathbb{R}^{N-n}$ and $\mathbf{z}_2 \in \mathbb{R}^n$. Consequently, partition $\mu = (\mu_1,\mu_2)^T$ and $\Sigma = \begin{pmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{pmatrix}$, so that $\mathbf{E}[\mathbf{z}_j] = \mu_j$ and $Cov(\mathbf{z}_j,\mathbf{z}_k) = \Sigma_{jk}$. Then, $\mathbf{z}_1|\mathbf{z}_2 \sim \mathcal{N}(\tilde{\boldsymbol{\mu}},\tilde{\Sigma})$, where $\tilde{\boldsymbol{\mu}} = \boldsymbol{\mu}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{z}_2 - \boldsymbol{\mu}_2)$ and $\tilde{\Sigma} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$.*

It follows that

$$\eta(\cdot)|\mathbf{y},\boldsymbol{\beta},\sigma^2 \sim \mathcal{N}(m^*(\cdot),\sigma^2C^*(\cdot,\cdot)) \tag{19}$$

where

$$m^*(x) = \mathbf{h}(x)^{\mathrm{T}}\boldsymbol{\beta} + \mathbf{t}(x)\mathbf{A}^{-1}(\mathbf{y}-\mathbf{H}\boldsymbol{\beta}) \tag{20}$$

$$C^*(\mathbf{x},\mathbf{x}') = C(\mathbf{x},\mathbf{x}') - \mathbf{t}(\mathbf{x})^{\mathrm{T}}A^{-1}\mathbf{t}(\mathbf{x}') \tag{21}$$

$$\mathbf{t}(\mathbf{x}) = [C(\mathbf{x},\mathbf{x}_1),\ldots,C(\mathbf{x},\mathbf{x}_n)]^{\mathrm{T}} \tag{22}$$

Removing the conditioning on $\boldsymbol{\beta}$ using standard integration techniques [13], obtain the posterior distribution

$$\eta(\cdot)|\mathbf{y},\sigma^2 \sim \mathcal{N}(m^{**}(\cdot),\sigma^2C^{**}(\cdot,\cdot)) \tag{23}$$

where

$$m^{**}(\mathbf{x}) = \mathbf{h}(\mathbf{x})^{\mathrm{T}}\hat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x})\mathbf{A}^{-1}(\mathbf{y}-\mathbf{H}\hat{\boldsymbol{\beta}}) \tag{24}$$

$$C^{**}(\mathbf{x},\mathbf{x}') = C^*(\mathbf{x},\mathbf{x}') + (\mathbf{h}(\mathbf{x})^{\mathrm{T}} - \mathbf{t}(\mathbf{x})^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{H})(\mathbf{H}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{H})^{-1}(\mathbf{h}(\mathbf{x}')^{\mathrm{T}} - \mathbf{t}(\mathbf{x}')^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{H})^{\mathrm{T}} \tag{25}$$

$$\hat{\boldsymbol{\beta}} = (\mathbf{H}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{y} \tag{26}$$

To estimate σ in Eq. (23), let q be the rank of $\mathbf{H}$. Then

$$\hat{\sigma}^2 = \frac{\mathbf{y}^{\mathrm{T}}(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{H})\mathbf{H}^{\mathrm{T}}\mathbf{A}^{-1})\mathbf{y}}{n-q-2} \tag{27}$$

As it can be seen, Gaussian process emulation consists in updating the prior distribution (14), which contains subjective information, by adding the objective information $\mathbf{y}$ in order to obtain the posterior distribution (23). This enables the calculation of the predictive mean $m^{**}(\cdot)$ given the data $\mathbf{y}$. Since $m^{**}(\cdot)$ does not include any term involving the expensive simulator $\eta(\cdot)$, it is a fast approximation of $\eta(\mathbf{x})$ for any $\mathbf{x}$ in its domain. The conditioning on σ^2 can also be eliminated, such that

$$\frac{\eta(\mathbf{x}) - m^{**}(\mathbf{x})}{\hat{\sigma}\sqrt{\frac{(n-q-2)C^{**}(\mathbf{x})}{n-q}}} \sim t_{n-q} \tag{28}$$

which is a Student's t-distribution with $n-q$ degrees of freedom (not to be confused with the degrees of freedom in a finite element method context). The complete process is summarized in Alg. 1.

Alg. 1: Gaussian process emulation

Input: Design points $\mathbf{x}_1,\ldots,\mathbf{x}_n$
Output: Predictive mean $m^{**}(\cdot)$
begin
1. Select n design points $\mathbf{x}_1,\ldots,\mathbf{x}_n$
2. Obtain the vector of observations $\mathbf{y} = [\eta(\mathbf{x}_1),\ldots,\eta(\mathbf{x}_n)]^{\mathrm{T}}$
3. Update the prior distribution (14) using $\mathbf{y}$ and obtain the posterior distribution (23)
4. Compute the predictive mean $m^{**}(\mathbf{x}) = \mathbf{E}[\eta(\mathbf{x})|\mathbf{y}]$ for any untried $\mathbf{x}$

end

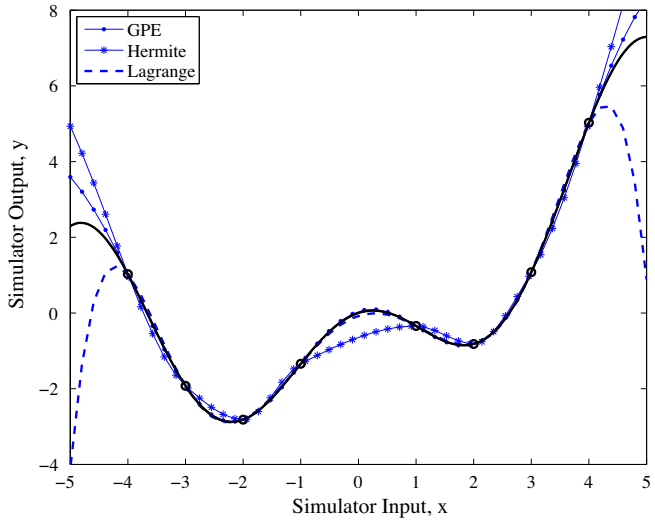


Fig. 6. Comparison of Gaussian process emulation against Hermite and Lagrange interpolation for 8 training runs of the simulator $\mathbf{y} = 0.5\mathbf{x} - \mathbf{x}\sin(\mathbf{x})$. The solid line is the output of the simulator.

A natural question that may arise is the comparison of Gaussian process emulation against other interpolation methods. Fig. 6 compares the GPE approximation to the simulator $\mathbf{y} = 0.5\mathbf{x} - \mathbf{x}\sin(\mathbf{x})$ with the Lagrange and Hermite interpolation. Regarding their cost, note that the computational load in the GPE algorithm is in the inversion of the correlation matrix $\mathbf{A}_{ij} = C(\mathbf{x}_i, \mathbf{x}_j)$ defined in Eq. (18). In practice, this matrix is not inverted, but decomposed in two triangular matrices by Cholesky decomposition. This operation has a cost of $\mathcal{O}(n^3)$. In comparison, Lagrange interpolation requires the solution of a linear system with a Vandermonde matrix, which also bears a cost of $\mathcal{O}(n^3)$. Alternative approaches exist, whereby the solution requires $\mathcal{O}(n^2)$ operations [43]. On the other hand, Hermite interpolation requires the solution of a tridiagonal system, with a cost of $\mathcal{O}(n)$. However, this method requires the derivatives of the interpolating polynomial and the function being interpolated to coincide in the endpoints. In order for this condition to hold, it is necessary to have either the values of the derivative at the endpoints or an accurate approximation to those values [44].

From a statistical point of view, an advantage of using Gaussian processes over classical methods is the following: Suppose it is not possible to produce simulator runs $\mathbf{y} = \eta(\mathbf{x})$ but only noisy versions thereof $\mathbf{y} = \eta(\mathbf{x}) + \epsilon$, with ϵ being random noise with variance σ_n^2. Then, the covariance function $C(\cdot,\cdot)$ becomes $C(\cdot,\cdot) + \delta_{ij}\sigma_n^2$, where δ_{ij} is the Kronecker delta. In that case, the Gaussian process emulator becomes a regression rather than an interpolation (by fitting a curve through the noisy data)[42]. This is convenient if statistics of the simulator's output (e.g. mean, variance, percentiles) are to be estimated.

5. The unification of Gaussian process emulation and domain decomposition

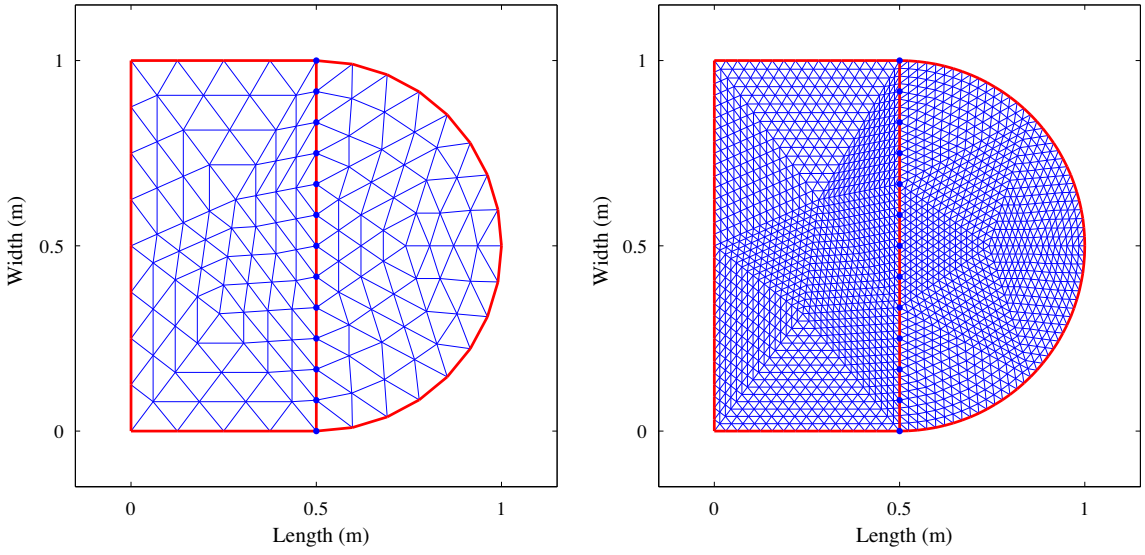
Let $u^*(\mathbf{x}) : \Omega \to \mathbb{R}$ be a finite element solution to the problem (1) for a low-fidelity finite element model L_f. Suppose that the solution $U^*(\mathbf{x}) : \Omega \to \mathbb{R}$ of a high-fidelity model H_f is to be obtained via domain decomposition. Assume the finite element mesh of H_f to be a refinement of the mesh of L_f. If a Delaunay triangulation is assumed for these meshes, it can be proved [34] that $\mathcal{N}(L_f) \subset \mathcal{N}(H_f)$, where $\mathcal{N}(L_f)$ and $\mathcal{N}(H_f)$ are the low and high-fidelity sets of nodes respectively. After being partitioned in subdomains, suppose the domain Ω contains n_x level sets with respect to the x-axis and n_y level sets with respect to the y-axis. As discussed in Section 3, a total of $n_x + n_y$ simulators of the form $\eta_i^x(\cdot) : \Gamma^x(c_i) \to \mathcal{L}^x(c_i)$ and $\eta_j^y(\cdot) : \Gamma^y(d_j) \to \mathcal{L}^y(d_j)$ can be specified for $u^*(\mathbf{x})$.

Notice that, since $\mathcal{N}(L_f) \subset \mathcal{N}(H_f)$, the sets $\mathcal{N}(H_f) \cap \Gamma^x(c_i)$ and $\mathcal{N}(H_f) \cap \Gamma^y(d_j)$ can be regarded as design points upon which Gaussian process emulators can be built in order to infer the output of each simulator $\eta_i^x(\cdot)$ and $\eta_j^y(\cdot)$. That way, all the values of $U^*(\mathbf{x})$ that solve the interface problem for H_f would be approximated by the emulator. The examples presented in the following section will help clarify this point.

6. Numerical examples

6.1. A case of 2 subdomains

The model of the deforming membrane presented in Section 3 was considered. The low-fidelity mesh consisted of 145 nodes,



(a) Low-fidelity model L_f (b) High-fidelity model H_f

Fig. 7. The finite element mesh of the low and high-fidelity finite element models on the domain Ω, which is decomposed into the subdomains Ω_1 and Ω_2. The design points on H_f are defined by the triangulation of L_f and therefore lie on the interface Γ.

whereas a refinement yielded a high-fidelity mesh with 2113 nodes. The level set $\mathcal{L}^x(0.5)$ for L_f was determined by solving the interface problem, by means of Eq. (5). The set $\mathcal{N}(H_f) \cap \Gamma^x(0.5)$ determined the design points upon which to build the Gaussian process emulator to approximate the solution to the interface problem for H_f. Fig. 7 illustrates how these design points are assimilated from L_f to H_f. Fig. 8 shows the simulator defined by $\eta^x : \Gamma^x(0.5) \to \mathcal{L}^x(0.5)$. It also shows the approximation to $\mathcal{L}^x(0.5)$ provided by the predictive mean of the Gaussian process emulator, as well as probability bounds of 2 standard deviations about this mean.

Once the solution of the interface problem for H_f was emulated, the approximate solution for the subdomains Ω_1 and Ω_2 was obtained in parallel by solving the system of equations (6). Fig. 9 shows the approximate solution to H_f for all the nodes in the high-fidelity mesh. The training runs obtained from the level set $\mathcal{L}^x(0.5)$, which were calculated by solving the low-fidelity model L_f, are also shown.

The computational advantage of the proposed approach can be appreciated by noting that 13 nodes lie in the interface of L_f. Thus, the cost of solving the interface problem for L_f was $\mathcal{O}(13^3)$, corresponding to the cost of solving the linear system (5). The refinement of the Delaunay triangulation led to an increase of 4 times the number of nodes on the interface, which implied a cost of $\mathcal{O}(4^3 \times 13^3)$. In order to compare the approximate solution of H_f, denoted by $\widetilde{U}^*(\mathbf{x})$, with the domain decomposition solution $U^*(\mathbf{x})$ the absolute difference $|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})|$ was computed for every node. By definition, the maximum difference over all the nodes is the infinity norm $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty$, which for this case had a magnitude of 8.12×10^{-4}. The contour defined by this comparison is shown in Fig. 10. By construction, the difference in every design point is equal to 0. The same is true for the boundary, given the boundary conditions of problem (1).

6.2. A case of two non-convex subdomains

An interesting case arises when the domain Ω or any of its subdomains is not convex. As a reminder, a set Ω is convex if for any pair of points x_1 and $x_2 \in \Omega$, the point $\lambda x_1 + (1-\lambda)x_2 \in \Omega$ for all $\lambda \in [0,1]$. Consider the domain shown in Fig. 11. Ω is clearly non-convex in $\mathbb{R}^2$, as it is trivial to construct a segment that connects two points belonging to the domain such that part of the segment does not belong to it. Due to this, the level set $\mathcal{L}^x(0.5)$ has a preimage $\Gamma^x(0.5)$ which is non-convex in $\mathbb{R}$, namely

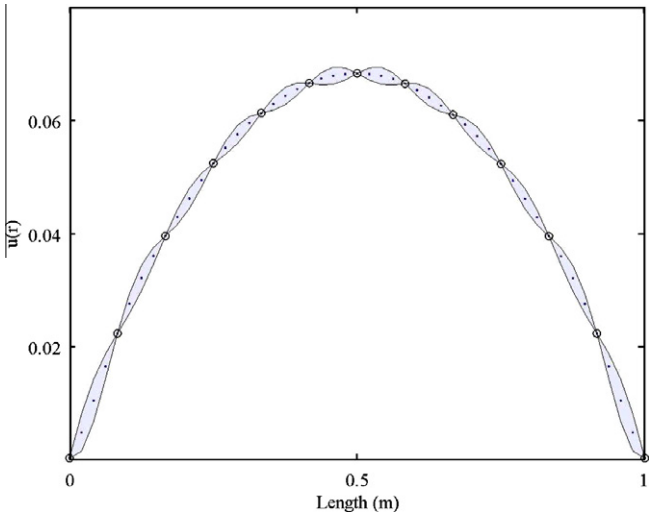


Fig. 8. Approximation to the level set $\mathcal{L}^x(0.5)$ of H_f with the mean of a Gaussian process emulator and uncertainty (2 standard deviations) about this mean; (o): training runs, (···): emulator's predictive mean.

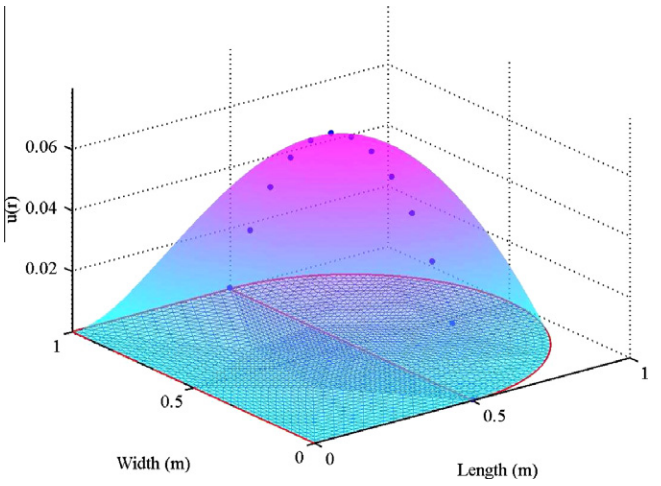


Fig. 9. Solution of the high-fidelity finite element model H_f. The blue dots represent the training runs obtained from the level set $\mathcal{L}^x(0.5)$ from the low-fidelity model L_f. The interface problem for H_f was approximated by a Gaussian process emulator. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

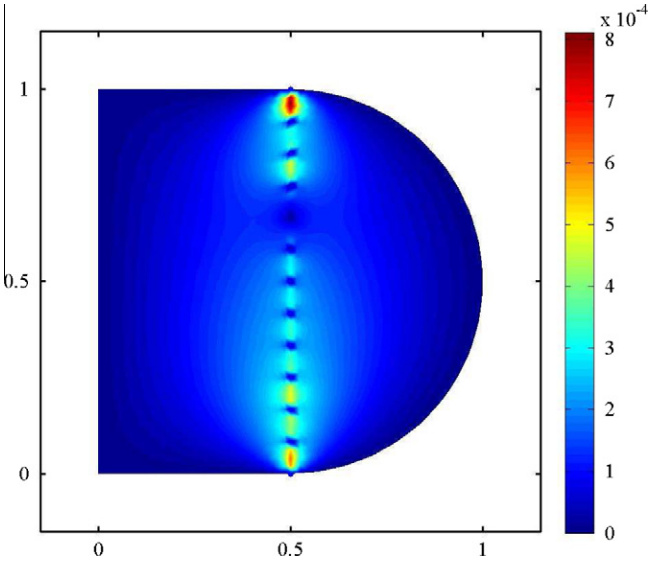
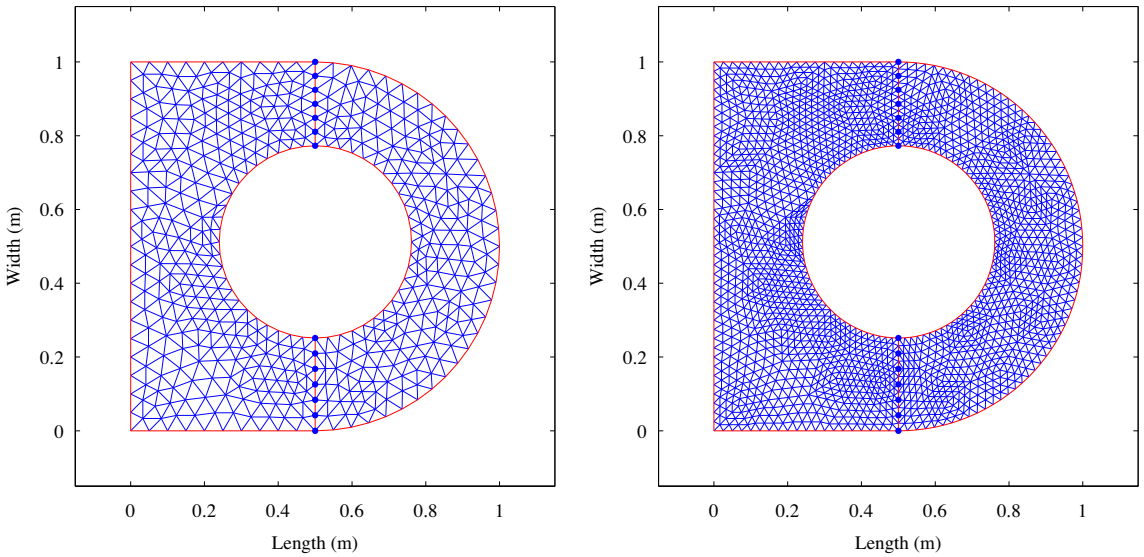


Fig. 10. Absolute value of the difference between the domain decomposition solution of H_f, denoted by $U^*(\mathbf{x})$, and the solution with the proposed method, denoted by $\widetilde{U}^*(\mathbf{x})$. By definition, the maximum of this difference for all nodes is $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty \simeq 8.12 \times 10^{-4}$.

$$\Gamma^x(0.5) = \{y \in \mathbb{R} | u^*(0.5, y) = z\} = \{y \in \mathbb{R} | u^*(0.5, y) = z; 0 \leqslant y \leqslant 0.25\} \cup \{y \in \mathbb{R} | u^*(0.5, y) = z; 0.75 \leqslant y \leqslant 1\} = \Gamma_1^x(0.5) \cup \Gamma_2^x(0.5) \tag{29}$$

Hence, in order to approximate $\mathcal{L}^x(0.5)$, two Gaussian process emulators were built, taking the respective design points from $\mathcal{N}(H_f) \cap \Gamma_1^x(0.5)$ and $\mathcal{N}(H_f) \cap \Gamma_2^x(0.5)$. The proposed method could then be applied as previously. The result of doing this can be seen in Fig. 12. Note that for the sake of brevity, figures with the predictive means of the emulators, as well as the uncertainty bounds are omitted. However, they resemble the parabolic curve in Fig. 8.

For this case, the low-fidelity mesh consisted of 532 nodes, whereas a refinement yielded a high-fidelity mesh with 2008 nodes. The refinement of L_f into H_f yielded two nodes per design point per preimage, such that the computational cost of solving each interface problem for L_f and obtaining the training runs was



(a) Low-fidelity model L_f (b) High-fidelity model H_f

Fig. 11. The finite element mesh of the low and high-fidelity finite element models on the non-convex domain Ω, which is decomposed into the subdomains Ω_1 and Ω_2. The set of design points lies on a non-convex interface.

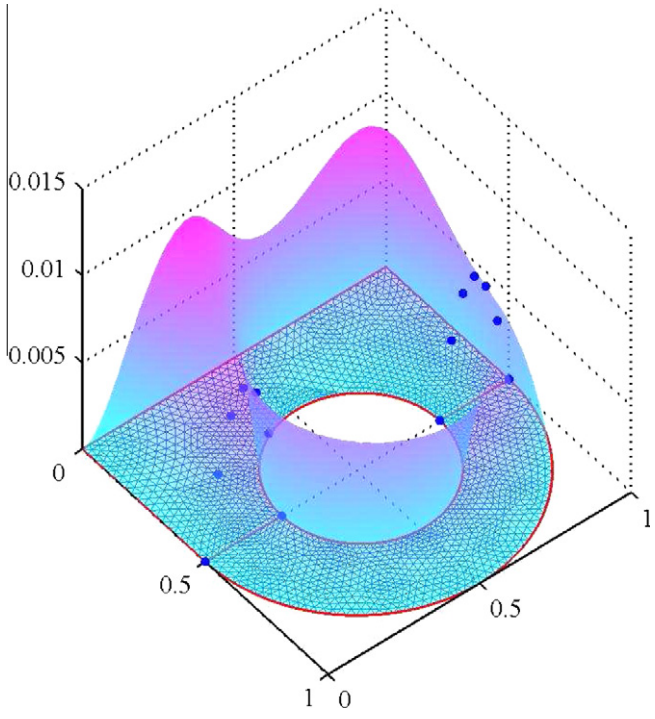


Fig. 12. Solution of the high-fidelity finite element model H_f. The blue dots represent the training runs obtained from the level set $\mathcal{L}^x(0.5)$ from the low-fidelity model L_f. The interface problem for H_f was approximated by two Gaussian process emulators, due to the non-convexity of the domain Ω. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$2 \times \mathcal{O}(7^3)$ as opposed to $2 \times \mathcal{O}(2^3 \times 7^3)$. The node-to-node comparison yielded a maximum difference of $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty = 3.80 \times 10^{-5}$. The corresponding contour is shown in Fig. 13.

6.3. A case of three subdomains

In this example, the proposed method is tested assuming a domain Ω that is partitioned into three subdomains. This partitioning

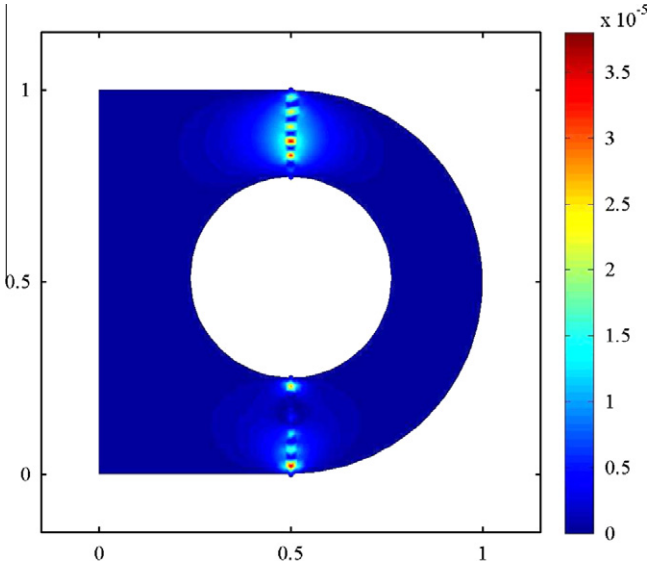
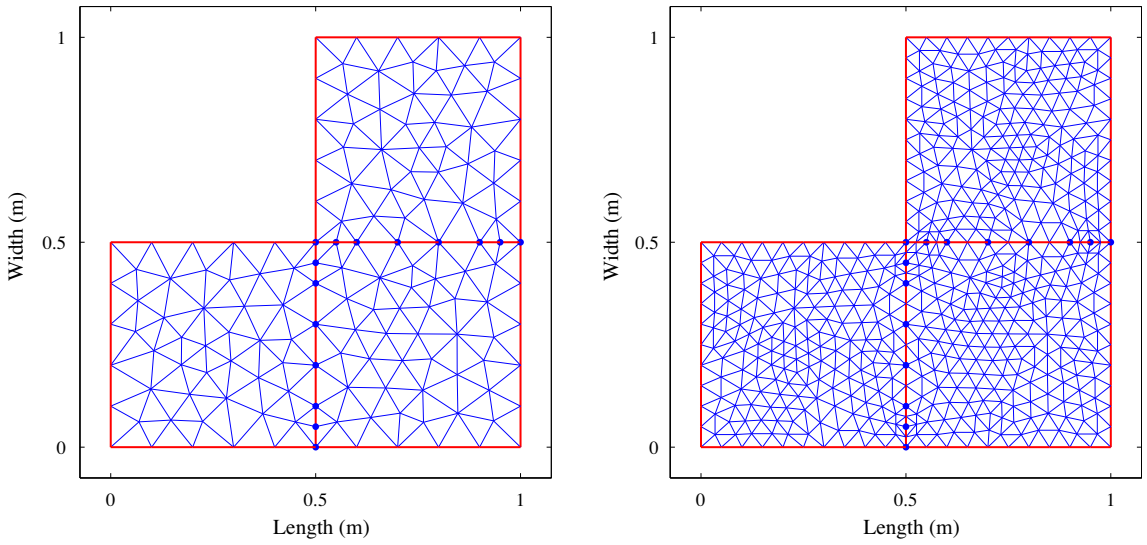


Fig. 13. Absolute value of the difference between the domain decomposition solution of H_f, denoted by $U^*(\mathbf{x})$, and the solution with the proposed method, denoted by $\widetilde{U}^*(\mathbf{x})$. By definition, the maximum of this difference for all nodes is $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty \simeq 3.8 \times 10^{-5}$.

determined two interfaces, whose intersection with the corresponding nodes produced two level sets, $\mathcal{L}^x(0.5)$ and $\mathcal{L}^y(0.5)$. The L-shaped domain Ω, as well as the low and high-fidelity meshes and the interfaces defining the design points are all depicted in Fig. 14. In this example, the low-fidelity mesh consisted of 150 nodes, whereas a refinement yielded a high-fidelity mesh with 557 nodes.

The simulators $\eta^x : I^x(0.5) \to \mathcal{L}^x(0.5)$ and $\eta^y : I^y(0.5) \to \mathcal{L}^y(0.5)$ were emulated running two Gaussian process emulators. The solution for the subdomains Ω_1, Ω_2, and Ω_3 was obtained in parallel following Eq. (6). The solution is shown in Fig. 15. Note that the refinement of L_f into H_f yielded again two nodes per design point on each interface, such that the computational cost of solving each interface problem for L_f and obtaining the training runs was $\mathcal{O}(8^3)$ as opposed to $\mathcal{O}(2^3 \times 8^3)$. Finally, The node-to-node comparison



(a) Low-fidelity model L_f (b) High-fidelity model H_f

Fig. 14. The finite element mesh of a low fidelity model on an L-shaped domain Ω, decomposed in three subdomains Ω_1, Ω_2, and Ω_3, thus inducing two interface problems.

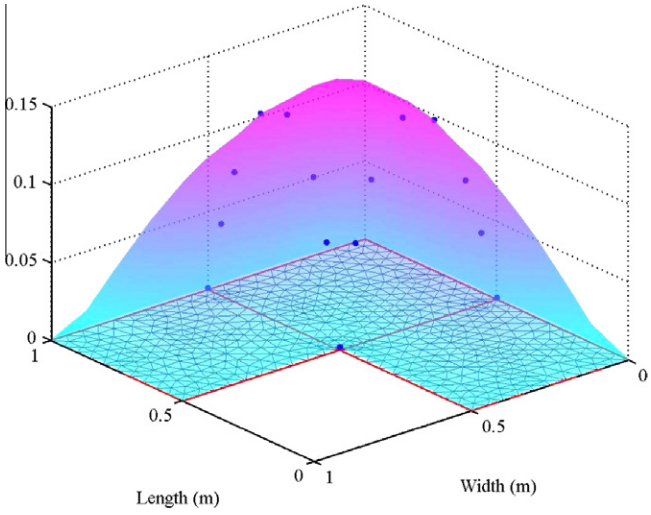


Fig. 15. Solution of the high-fidelity finite element model H_f. The blue dots represent the training runs obtained from the level sets $\mathcal{L}^x(0.5)$ and $\mathcal{L}^y(0.5)$ from the low-fidelity model L_f. The interface problems for H_f was approximated by Gaussian process emulators. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

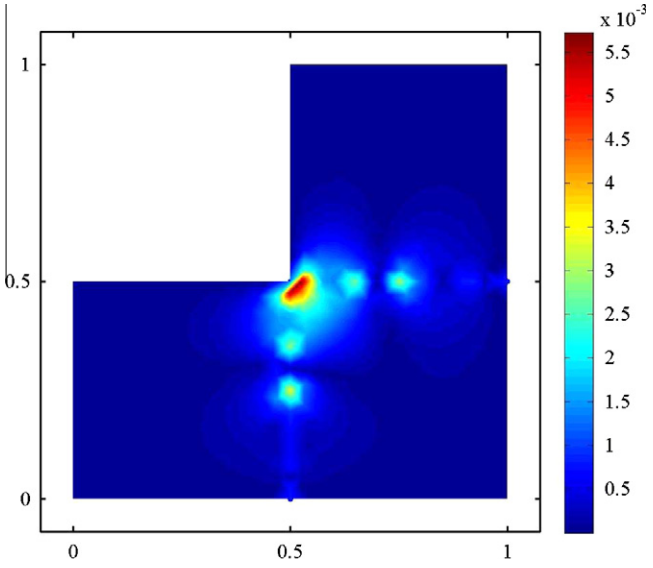


Fig. 16. Absolute value of the difference between the domain decomposition solution of H_f, denoted by $U^*(\mathbf{x})$, and the solution with the proposed method, denoted by $\widetilde{U}^*(\mathbf{x})$. By definition, the maximum of this difference for all nodes is $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty \simeq 5.71 \times 10^{-3}$.

yielded a maximum difference of $\|\widetilde{U}^*(\mathbf{x}) - U^*(\mathbf{x})\|_\infty = 5.71 \times 10^{-3}$. The resulting contour is shown in Fig. 16.

7. Conclusions

A Bayesian method based on Gaussian process emulators to solve boundary value problems in the context of domain decomposition was proposed. The method can assimilate a low-fidelity finite element model with a computationally more expensive high-fidelity model, given that the solution of the governing elliptic PDE is sufficiently smooth. The solution of the interface problem of the low-fidelity model was shown to provide with the training runs upon which the emulators can be built in order to approximate the more expensive solution of the interface problem of a corresponding high-fidelity model. That way, the computational cost of the domain decomposition solution of the high-fidelity model was reduced. A good agreement between the proposed approximation and the direct domain decomposition solution was found for different geometries of the domain. It was shown that neither the domain, or even the partitioning subdomains, need to be convex. Future work will include the extension of the method to stochastic partial differential equations. A rigorous analytical error analysis should also be considered for future investigations.

Acknowledgments

FADO gratefully acknowledges the support of the Engineering and Physical Sciences Research Council (EPSRC) for the award of a studentship through an Ideas Factory grant and the Consejo Nacional de Ciencia y Tecnología (CONACYT) for the award of a scholarship from the Mexican Government. SA gratefully acknowledges

the support of the Royal Society through the award of Wolfson Research Merit award.

References

[1] Toselli A, Windlund O. Domain decomposition methods algorithms and theory. Berlin: Springer Series in Computational Mathematics; 2005.
[2] Rao ARM, Rao TVSRA, Dattaguru B. A new parallel overlapped domain decomposition method for nonlinear dynamic finite element analysis. Comput Struct 2003;81(26-27):2441–54.
[3] Gallimard L, Sassi T. A posteriori error analysis of a domain decomposition algorithm for unilateral contact problem. Comput Struct 2010;88(13-14):879–88.
[4] Guibert D, Tromeur-Dervout D. Parallel adaptive time domain decomposition for stiff systems of ODEs/DAEs. Comput Struct. 2007;85(9):553–62.
[5] Bendali A, Boubendir Y, Fares M. A FETI-like domain decomposition method for coupling finite elements and boundary elements in large-size problems of acoustic scattering. Comput Struct 2007;85(9):526–35.
[6] Sarkar A, Benabbou N, Ghanem R. Domain decomposition of stochastic PDEs: Theoretical formulations. Int J Numer Methods Eng 2009;77(5):689–701.
[7] Cliffe KA, Graham IG, Scheichl R, Stals L. Parallel computation of flow in heterogeneous media modelled by mixed finite elements. J Comput Phys 2000;164(2):258–82.
[8] Kennedy MC, O'Hagan A. Bayesian calibration of computer models. J Royal Statist Soc Ser B Statist Method 2001;63(3):425–50.
[9] Peirano E, Talay D. Domain decomposition in science and engineering. In: Fourteenth International Conference on Domain Decomposition Methods, National Autonomous University of Mexico (UNAM), Mexico City, Mexico, Chapter Domain decomposition by stochastic methods, 2003. pp. 131–147.
[10] Challenor P, Hankin R, Marsh R. Avoiding dangerous climate change. Cambridge, UK: Cambridge University Press; 2006. Chapter Achieving robust design from computer simulations, pp. 55–63.
[11] Rougier J. Probabilistic inference for future climate using an ensemble of climate model evaluations. Climatic Change 2007;81(3):247–64.
[12] Kennedy MC, Anderson CW, Conti S, O'Hagan A. Case studies in Gaussian process modelling of computer codes. Reliab Eng Syst Safety 2006;91(10-11):1301–9.
[13] Haylock R, O'Hagan A. Bayesian statistics, vol. 5. Oxford, UK: Oxford University Press; 1996. Chapter On inference for outputs of computationally expensive algorithms with uncertainty on the inputs.
[14] Oakley JE, O'Hagan A. Probabilistic sensitivity analysis of complex models: a Bayesian approach. J Roy Stat Soc B 2004;66(3):751–69.
[15] Kolachalama V, Bressloff N, Nair P. Mining data from hemodynamic simulations via Bayesian emulation. BioMed Eng 2007;6(47).
[16] DiazDelaO FA, Adhikari S. Structural dynamic analysis using Gaussian process emulators. Eng Comput 2010;27(5):580–605.
[17] DiazDelaO FA, Adhikari S. Gaussian process emulators for the stochastic finite element method. Int J Numer Methods Eng 2011;87(6):521–40.
[18] Busby D, Hierarchical adaptive experimental design for Gaussian process emulators. Reliab Eng Syst Safety 2009;94(7):1183–1193 [special issue on Sensitivity Analysis].
[19] Marrel A, Iooss B, Laurent B, Roustant O. Calculations of Sobol indices for the Gaussian process metamodel. Reliab Eng Syst Safety 2009;94(3):742–51.
[20] Bates R, Kennet R, Steinberg D, Wynn H. Achieving robust design from computer simulations. Q Tech Quant Manag 2006;3(2):161–77.
[21] McFarland J, Mahadevan S. Multivariate significance testing and model calibration under uncertainty. Comput Methods Appl Mech Eng 2008;197(29-32):2467–79.
[22] Daneshkhah A, Bedford T. Advances in mathematical modeling for reliability. The Netherlands: IOS Press; 2008. Chapter Sensitivity analysis of a reliability system using Gaussian processes, pp. 46–62.
[23] Bathe KJ. Finite element procedures. Englewood Cliffs, New Jersey, USA: Prentice Hall Inc.; 1995.
[24] Petyt M. Introduction to finite element vibration analysis. Cambridge, UK: Cambridge University Press; 1998.
[25] Babuška I, Suri M. The *p*- and *h*-*p* versions of the finite element method, an overview. Comput Methods Appl Mech Eng 1990;80(1-3):5–26.
[26] O'Hagan A. Bayesian analysis of computer code outputs: a tutorial. Reliab Eng Syst Safety 2006;91(10-11):1290–300.
[27] Rougier J, Sexton D, Murphy M, Stainforth D. Emulating the sensitivity of the HadSM3 climate model using ensembles from different but related experiments. Technical Report 07/04, MUCM, University of Sheffield, Sheffield, UK, 2007.
[28] Oakley J. Eliciting Gaussian process priors for complex computer codes. The Statist 2002;51(1):81–97.
[29] Oakley J. Estimating percentiles of computer code outputs. Appl Statist 2004;53:83–93.
[30] Sacks J, Welch W, Mitchell T, Wynn H. Design and analysis of computer experiments. Statist Sci 1989;4(4):409–35.
[31] O'Hagan A. Bayesian statistics, vol. 4. Cambridge, UK: Oxford University Press; 1992. Chapter Some Bayesian numerical analysis, pp. 345–363.
[32] Haylock R. Bayesian inference about outputs of computationally expensive algorithms with uncertainty on the inputs, Ph.D. thesis, University of Nottingham, Nottingham, UK 1996.
[33] Krzanowski W. Principles of multivariate analysis. Oxford, UK: Oxford University Press; 2000.
[34] George P. Automatic mesh generation: application to finite element methods. New York: Wiley; 1991.
[35] Geymonat G, Krasucki F, Marini D, Vidrascu M. A domain decomposition method for a bonded structure. Math Models Methods Appl Sci 1998;8(8):1387–402.
[36] Koko J. Convergence analysis of optimization-based domain decomposition methods for a bonded structure. Appl Numer Math 2008;58(1):69–87.
[37] Krasucki F, Mnch A, Ousset Y. Numerical simulation of debonding of adhesively bonded joint. Int J Solids Struct 2002;39(26):6355–83.
[38] Lui SH. A Lions non-overlapping domain decomposition method for domains with an arbitrary interface. IMA J Numer Anal 2009;29(2):332–49.
[39] Haase G, Kuhn M. Preprocessing for 2D FE-BE domain decomposition methods. Comput Visual Sci 1999;2:25–35.
[40] Funaro D, Quarteroni A, Zanolli P. An iterative procedure with interface relaxation for domain decomposition methods. SIAM J Numer Anal 1988;25(6):1213–36.
[41] Liao H, Shi H, Sun Z. Corrected explicit-implicit domain decomposition algorithms for two-dimensional semilinear parabolic equations. Sci China Ser A Math 2009;52(11):2362–88.
[42] Rasmussen CE, Williams C. Gaussian processes for machine learning. Massachusetts Institute of Technology: the MIT Press; 2006.
[43] Van Loan CE. Introduction to scientific computing: a matrix-vector approach using MATLAB. Prentice Hall PTR; 1999.
[44] Burden RL, Faires JD. Numerical analysis. Brooks/Cole Publishing Company; 2000.