Received: 25 January 2023 | Revised: 30 June 2023 | Accepted: 16 August 2023

DOI: 10.1002/nme.7348

RESEARCH ARTICLE

WILEY

Multi-fidelity error estimation accelerates greedy model reduction of complex dynamical systems

Lihong Feng[1] | **Luigi Lombardi**[2] | **Giulio Antonini**[3] | **Peter Benner**[1,4]

[1]Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany

[2]Non Volatile Memories Design, Micron Semiconductor, Avezzano, Italy

[3]UAq EMC Laboratory, Department of Industrial and Information Engineering and Economics, University of L'Aquila, L'Aquila, Italy

[4]Fakultät für Mathematik, Otto-von-Guericke-Universität, Magdeburg, Germany

Correspondence
Lihong Feng, Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstrasse 1, 39106 Magdeburg, Germany.
Email: feng@mpi-magdeburg.mpg.de

Abstract

Model order reduction usually consists of two stages: the offline stage and the online stage. The offline stage is the expensive part that sometimes takes hours till the final reduced-order model is derived, especially when the original model is very large or complex. Once the reduced-order model is obtained, the online stage of querying the reduced-order model for simulation is very fast and often real-time capable. This work concerns a strategy to speed up the offline stage of model order reduction for large and complex systems. In particular, it is successful in accelerating the greedy algorithm that is often used in the offline stage for reduced-order model construction. We propose to replace the high-fidelity error estimator in the greedy algorithm with multi-fidelity error estimation. Consequently, the computational complexity of the greedy algorithm is reduced and the algorithm converges more than two times faster without incurring noticeable accuracy loss.

KEYWORDS

adaptivity, dynamical systems, electromagnetics, error estimation

1 | INTRODUCTION

Model order reduction (MOR) has achieved much success in many areas of computational science with its capability of realizing real-time simulation and providing accurate results. Different MOR methods, their applications, and the promising results they produce can be found in the survey papers[1-3] and books.[4-9]

MOR usually needs an offline stage for constructing the reduced-order model (ROM). For many intrusive MOR methods that are based on projection, the offline stage is realized via a greedy algorithm.[10-15] The greedy algorithm is used to properly select *important* parameter samples that contribute most to the solution space. The offline computational time is basically the runtime of the greedy algorithm. For large-scale systems, the offline computation is expensive and the runtime could be longer than several hours.[16] Sometimes, the system is not very large, for example, the number of degrees of freedom is only $O(10^5)$, but the system structure is complicated, so that the greedy algorithm still takes long time to converge, see, for example, the third example in Section 4 of this work.

It is known that an efficient error estimator makes the greedy algorithm successful in producing an accurate ROM without running many iterations.[10,12,17,18] Therefore, many efforts have been made in this direction to develop computable error estimators for different problems.[12,14,17,19-29] However, more attention has been paid to improve the effectivity or accuracy of the error estimator than to develop more efficient strategies to accelerate the greedy process.[26,28-31] Recently, some techniques are proposed to improve the adaptivity of the greedy algorithm.[30,32-34]

In References 30 and 35, we proposed a surrogate model for error estimation, and proposed an adaptive greedy algorithm by alternatively using this surrogate error estimator and the original error estimator during the greedy algorithm. The focus in References 30 and 35 was to make the greedy process adaptive by starting from a coarse training set of a small number of parameter samples, and adaptively update the coarse training set with the aid of a surrogate error estimator. The original error estimator is computed only over the coarse training set, while the surrogate error estimator helps to pick out candidates of *important* parameter samples from a fine training set, which are then collected in the coarse training set.

In this work, we emphasize the role of the surrogate error estimator and propose the concept of bi-fidelity error estimation and multi-fidelity error estimation. In fact, a bi-fidelity error estimation has been used in the adaptive greedy algorithm proposed in References 30 and 35 without being formally defined, that is, the original (high-fidelity) error estimator, and the surrogate (low-fidelity) error estimator. To further improve the convergence speed of the greedy algorithm, we propose multi-fidelity error estimation built upon the bi-fidelity error estimation. Here, we use a more efficient high-fidelity error estimator than the two different high-fidelity error estimators used in References 30 and 35. Although the proposed multi-fidelity error estimation is dependent on the original high-fidelity error estimator, the idea of using multi-fidelity error estimation is general and can be extended to develop multi-fidelity error estimation associated with other high-fidelity error estimators.

Unlike the problems considered in References 30 and 35, whose ROMs can be constructed by standard greedy algorithms within seconds to minutes, we consider in this work much more complicated problems. The standard greedy algorithm takes up to several hours to converge for such problems (see the third example in Section 4). By using the proposed multi-fidelity error estimation, the greedy algorithm achieves more than 2.5× speed-up and produces ROMs with little loss of accuracy. The speed-up is also higher than using the bi-fidelity error estimation.

In the next section, we present the greedy algorithm in the standard form. Then we analyze some ingredients of the algorithm, which contribute most to the computational cost. Starting from those computationally expensive parts, we develop possible strategies to reduce the computational complexity in Section 3. As a consequence, it becomes clear that the resulting strategy develops a greedy algorithm with multi-fidelity error estimation. The proposed algorithm is then applied to large time-delay systems with many delays. The numerical tests on three large time-delay systems with more than 100 delays are demonstrated in Section 4. Conclusions are given in the end.

2 | STANDARD GREEDY ALGORITHM

The standard greedy algorithm was first proposed for steady systems without time evolution.[10,14] Then it was extended to proper orthogonal decomposition (POD)-greedy for dynamical systems,[12] which is used to construct the ROM using snapshots in the time domain. Later the greedy algorithm found its capability in adaptively choosing interpolation points for frequency-domain MOR methods.[19,20] The greedy algorithm for steady systems and frequency-domain MOR has the same formulation, whereas POD-greedy for time-domain MOR of time-dependent systems needs an singular value decomposition step at each greedy iteration. In this work, we focus on the greedy algorithm, though the proposed scheme can be easily extended to POD-greedy (Algorithm 1).

Algorithm 1. Standard greedy algorithm

Input: the FOM, a training set Ξ composed of parameter samples taken from the parameter domain $\mathcal{P}$, error tolerance tol<1, $\Delta(\mu)$ to estimate the error.

Output: Projection matrix V.

1: Choose initial parameter $\mu^* \in \Xi$.
2: $V \leftarrow \emptyset$, $\varepsilon = 1$.
3: **while** $\varepsilon >$ tol **do**
4: Compute the snapshot(s) $x(\mu^*)$ by solving the FOM at $\mu = \mu^*$.
5: Update V by $V = \text{orth}\{V, x(\mu^*)\}$, (e.g., using the modified Gram–Schmidt process with deflation.)
6: Compute μ^* such that $\mu^* = \arg\max_{\mu \in \Xi} \Delta(\mu)$.
7: $\varepsilon = \Delta(\mu^*)$.
8: **end while**

We consider constructing a ROM for the following full-order model (FOM) using the greedy algorithm,

$$F(x(\mu), \mu) = B(\mu). \tag{1}$$

Here, $F(x(\mu), \mu) \in \mathbb{C}^{n\times n_I}$, $x(\mu) \in \mathbb{C}^{n\times n_I}$, and $B(\mu) \in \mathbb{C}^{n\times n_I}$. $\mu \in \mathcal{P}$ is a parameter in the parameter domain $\mathcal{P}$. The variable n is the order of the FOM, which can be the number of degrees of freedom after numerical discretization of PDEs describing a physical phenomenon. The proposed algorithms are also applicable to problems with $n_I > 1$.

The ROM can be obtained via Galerkin projection using a projection matrix $V \in \mathbb{R}^{n\times r}$, $r \ll n$, as below,

$$\hat{F}(Vz(\mu), \mu) = \hat{B}(\mu), \tag{2}$$

where $\hat{F}(Vz(\mu), \mu) = V^T f(Vz(\mu), \mu) \in \mathbb{C}^{r\times n_I}$, $z(\mu) \in \mathbb{C}^{r\times n_I}$, and $\hat{B}(\mu) = V^T B(\mu) \in \mathbb{C}^{r\times n_I}$.

The standard greedy process used to compute the projection matrix V is described in Algorithm 1. Step 4 in Algorithm 1 solves the FOM at μ^*, and Step 6 computes an error estimator $\Delta(\mu)$ at all μ in Ξ. These two steps constitute the most computational expensive part of the greedy algorithm. However, Step 4 is unavoidable, since $x(\mu^*)$ is needed for the reduced basis construction.The computational complexity of Step 6 could be reduced, if the cardinality of Ξ, that is, $|\Xi|$ is kept small, so that $\Delta(\mu)$ needs not be evaluated at many parameter samples. This is the motivation of the surrogate error estimator proposed in References 30 and 35. The surrogate error estimator can be seen as a low-fidelity error estimator as compared to the original error estimator $\Delta(\mu)$, since it is only an approximation to $\Delta(\mu)$, but is much cheaper to compute.

In the next section, we present a greedy algorithm using bi-fidelity error estimation, where a low-fidelity error estimator is computed following the method in References 30 and 35. Based on this, a greedy algorithm using multi-fidelity error estimation associated with a particular high-fidelity error estimator for MOR of linear parametric systems, is proposed.

Remark 1. In a strict sense, Algorithm 1 should be called standard **weak** greedy algorithm in contrast to the **strong** greedy algorithm that employs the true error over the whole parameter domain.[15] When a finite training set including samples of μ and/or an error estimator are/is employed to select the parameter μ^*, the greedy process is then called the weak greedy algorithm.[11] In fact, "greedy process," "greedy algorithm," or "greedy sampling" are also frequently used for the weak greedy setting in the literature.[10,13,14] The aim of this paper is to propose multi-fidelity error estimation to accelerate the weak greedy process. For simplicity, we just omit "weak" in the following discussions. Furthermore, when bi-fidelity or multi-fidelity error estimation is used in the greedy process, the weak greedy algorithm is not in its standard form any more. However, the process of selecting the parameter μ^* is similar, that is, the one corresponding to the largest value of an error indicator is selected at each iteration. From this aspect, it can still be seen as a greedy process. Therefore, the greedy algorithms discussed in the following sections are in the general and loose senses.

3 | GREEDY ALGORITHM WITH BI-FIDELITY AND MULTI-FIDELITY ERROR ESTIMATION

This section presents greedy algorithms with bi-fidelity and multi-fidelity error estimation, respectively.

3.1 | Greedy algorithm with bi-fidelity error estimation

Algorithm 2 is the greedy algorithm with bi-fidelity error estimation. Its original version using a specific high-fidelity error estimator was firstly proposed in Reference 30.

The key step of Algorithm 2 is Step 8, where the RBF surrogate $\Delta_{\mathtt{rbf}}(\mu)$ is computed using values of $\Delta(\mu)$ at the samples of μ in the small parameter set Ξ_c. Basically, $\Delta_{\mathtt{rbf}}(\mu)$ is represented by a weighted sum of RBFs, that is,

$$\Delta_{\mathtt{rbf}}(\mu) = \sum_{i=1}^{m} w_i \Phi(\mu - \mu_i), \tag{3}$$

where $\Phi(\mu)$ are RBFs, μ_i are the samples in Ξ_c, and m is the cardinality of Ξ_c, which is small. Once w_i are known, the surrogate error estimator $\Delta_{\mathtt{rbf}}(\mu)$ is known. The weights w_i are computed via enforcing $\Delta_{\mathtt{rbf}}(\mu)$ to interpolate $\Delta(\mu)$ at $\mu_j, \forall \mu_j \in \Xi_c$, that is, $\Delta_{\mathtt{rbf}}(\mu_j) = \Delta(\mu_j)$. Inserting $\mu = \mu_j \in \Xi_c$ into (3), the weights w_i can be computed by solving

Algorithm 2. Greedy algorithm with bi-fidelity error estimation

Input: the FOM, a training set Ξ_c composed of a small number of parameter samples of μ taken from the parameter domain $\mathcal{P}$, a set Ξ_f composed a large number of parameter samples of μ from $\mathcal{P}$, error tolerance tol<1, $\Delta(\mu)$ to estimate the error.

Output: Projection matrix V.

1: Choose initial parameter $\mu^* \in \Xi_c$.
2: $V \leftarrow \emptyset$, $\varepsilon = 1$.
3: **while** ε>tol **do**
4: Compute the snapshot(s) $x(\mu^*)$ by solving the FOM at $\mu = \mu^*$.
5: Update V by $V = \text{orth}\{V, x(\mu^*)\}$, (e.g., using the modified Gram-Schmidt process with deflation.)
6: Compute μ^* such that $\mu^* = \arg\max_{\mu\in\Xi_c} \Delta(\mu)$.
7: Compute μ^o such that $\mu^o = \arg\min_{\mu\in\Xi_c} \Delta(\mu)$.
8: Compute the radial basis function (RBF) surrogate $\Delta_{\texttt{rbf}}(\mu)$ using values of $\Delta(\mu)$ corresponding to the samples of μ in Ξ_c via∼(3) and (4).
9: Evaluate $\Delta_{\texttt{rbf}}(\mu)$ over Ξ_f and pick out a parameter μ_c from the large parameter set Ξ_f corresponding to the largest value of $\Delta_{\texttt{rbf}}(\mu)$, that is, $\mu_c = \arg\max_{\mu\in\Xi_f} \Delta_{\texttt{rbf}}(\mu)$ from Ξ_f.
10: Update the small parameter set Ξ_c: if $\Delta_{\texttt{rbf}}(\mu_c)$>tol, enrich Ξ_c with μ_c, that is, $\Xi_c = \{\Xi_c, \mu_c\}$, if $\Delta(\mu^o)$<tol, remove μ^o from Ξ_c: $\Xi_c = \Xi_c \backslash \mu^o$.
11: $\varepsilon = \Delta(\mu^*)$.
12: **end while**

the linear system as below,

$$\begin{bmatrix} \Phi(\mu_1-\mu_1) & \dots & \Phi(\mu_1-\mu_m) \\ \vdots & \vdots & \vdots \\ \Phi(\mu_m-\mu_1) & \dots & \Phi(\mu_m-\mu_m) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix} = \begin{bmatrix} \Delta(\mu_1) \\ \vdots \\ \Delta(\mu_m) \end{bmatrix}. \tag{4}$$

Since values of $\Delta(\mu)$ at $\mu \in \Xi_c$ are available, the weights can be easily computed by solving the above small linear system with $m \times m$ being the dimension of the coefficient matrix. As this is a rather small system, we usually do not observe ill conditioning. Otherwise, we can use a regularized version of (4).[30,35]

At each iteration of the bi-fidelity greedy algorithm, the linear system is solved for once (Step 8), then the surrogate estimator $\Delta_{\texttt{rbf}}(\mu)$ is evaluated over a larger parameter set Ξ_f using the weighted sum in (3) (Step 9). This process of computing the weights and evaluating $\Delta_{\texttt{rbf}}(\mu)$ is much faster than evaluating the high-fidelity error estimator over a training set Ξ whose cardinality $|\Xi|$ is much larger than $|\Xi_c|$. This is usually the case for the standard greedy algorithm, where $|\Xi| > |\Xi_c|$. Finally, at each iteration of the bi-fidelity algorithm, the total computational cost of Steps 6–8: computing the high-fidelity error estimator $\Delta(\mu)$ over Ξ_c, solving the linear system (4) and evaluating the surrogate estimator $\Delta_{\texttt{rbf}}(\mu)$ over Ξ_f is still much cheaper than computing the high-fidelity error estimator $\Delta(\mu)$ over a training set Ξ, whose cardinality is, for example, twice that of $|\Xi_c|$, as shown in the numerical tests.

Besides computing μ^* corresponding to the maximal value of the error estimator $\Delta(\mu)$ over Ξ_c, the minimal value of $\Delta(\mu)$ is also computed in Step 7. The corresponding parameter μ^o could be deleted from Ξ_c if $\Delta(\mu^o)$ is already below the tolerance tol, see Step 10. In this way, the cardinality of the training set Ξ_c remains almost constant, and can further save computations as compared with enriching Ξ_c only. We will show in the numerical results that adding and removing samples to and from Ξ_c gets ROMs with similar accuracy (even smaller) as only adding samples to Ξ_c, but leads to even faster convergence of the greedy algorithm.

Remark 2. In Step 9, it is also possible to choose more than one parameter from Ξ_f by modifying Step 9 as: choose n_{add} samples from Ξ_f corresponding to n_{add} largest values of $\Delta_{\texttt{rbf}}(\mu)$. Similarly, In Step 7, one can also choose $n_{\text{del}} > 1$ parameter samples corresponding to n_{del} smallest values of $\Delta(\mu)$ from Ξ_c. However, this will more or less increase the computational time at each iteration, since more computations are needed to choose those samples. Furthermore, to make sure that only samples at which $\Delta_{\texttt{rbf}}(\mu)$ is larger than the tolerance

tol are added to Ξ_c, and only samples at which $\Delta(\mu)$ is smaller than tol are removed, additional calculations are necessary to check if all the selected samples meet the above criteria and should be finally selected or removed (see Step 10). Therefore, adding/removing at most one parameter sample each time should be more efficient. In the numerical tests, we also show results when $n_{\text{add}} = n_{\text{del}} = 2$ and $n_{\text{add}} = n_{\text{del}} = 5$ at each iteration of Algorithm 2.

The bi-fidelity error estimation is general and can be applied to any high-fidelity error estimators. For example, the high-fidelity error estimator in Reference 30 estimates the error of the ROM for nonlinear time-dependent parametric systems in the time domain, while the high-fidelity error estimator in Reference 35 estimates the error of the ROM in the frequency-domain for linear parametric systems.

3.2 | Greedy algorithm with multi-fidelity error estimation

The multi-fidelity error estimation we are going to introduce depends on the formulation of the high-fidelity error estimator $\Delta(\mu)$. To illustrate the basic concept, we use an error estimator proposed in Reference 20 as the high-fidelity error estimator and discuss how to further reduce the computational load by using multi-fidelity error estimation.

3.2.1 | An error estimator for linear parametric systems

The error estimator is applicable to estimating the output error of the ROM for FOMs in the following linear parametric form,

$$\begin{aligned} M(\mu)x(\mu) &= B(\mu), \\ y(\mu) &= C(\mu)x(\mu). \end{aligned} \tag{5}$$

Here, $M(\mu) \in \mathbb{R}^{n\times n}$, $B(\mu) \in \mathbb{R}^{n\times n_I}$, $C(\mu) \in \mathbb{R}^{n_O\times n}$, $x(\mu) \in \mathbb{R}^{n\times n_I}$, $y(\mu) \in \mathbb{R}^{n_O\times n_I}$. We consider the general case that both $B(\mu)$ and $C(\mu)$ are matrices, that is, systems with multiple inputs and multiple outputs. The ROM of the above linear parametric system can be derived via Galerkin projection using a projection matrix V composed of the reduced basis. That is,

$$\begin{aligned} \hat{M}(\mu)z(\mu) &= \hat{B}(\mu), \\ \hat{y}(\mu) &= \hat{C}(\mu)z(\mu), \end{aligned} \tag{6}$$

where $\hat{M}(\mu) = V^T M(\mu)V$, $\hat{B}(\mu) = V^T B(\mu)$, $\hat{C}(\mu) = C(\mu)V$.

For the general situation when both $B(\mu)$ and $C(\mu)$ are matrices, the error of the i,jth entry of the output matrix $\hat{y}(\mu)$ is

$$\begin{aligned} &|y_{ij}(\mu) - \hat{y}_{ij}(\mu)| \\ &= |C_i(\mu)(M^{-1}(\mu)B(\mu) - V\hat{M}^{-1}(\mu)\hat{B}_j(\mu))| \\ &= |C_i(\mu)M^{-1}(\mu)(B_j(\mu) - M(\mu)\underbrace{V\hat{M}^{-1}(\mu)\hat{B}_j(\mu)}_{\hat{x}_j(\mu):=Vz_j(\mu)})| \\ &= |C_i(\mu)M^{-1}(\mu)r_j(\mu)|, \end{aligned} \tag{7}$$

where $C_i(\mu)$ is the ith row of $C(\mu)$ and $B_j(\mu)$ is the jth column of $B(\mu)$. Here, we have defined: $z_j(\mu) = \hat{M}^{-1}(\mu)\hat{B}_j(\mu)$, that is, $\hat{M}(\mu)z_j(\mu) = \hat{B}_j(\mu)$, $\hat{x}_j(\mu) := Vz_j(\mu)$ and $r_j(\mu) := B_j(\mu) - M(\mu)\hat{x}_j(\mu)$. It is seen that

$$\hat{M}(\mu)z_j(\mu) = \hat{B}_j(\mu),$$

is a reduced-order model of

$$M(\mu)x_j(\mu) = B_j(\mu), \tag{8}$$

and $\hat{x}_j(\mu) \approx x_j(\mu)$, the jth column of $x(\mu)$. Finally, $r_j(\mu)$ is the residual induced by $\hat{x}_j(\mu)$.

From the last equation in (7), we see that to compute the absolute error of $\hat{y}_{ij}$, the following residual system needs to be solved:

$$M(\mu)x_{r_j}(\mu) = r_j(\mu). \tag{9}$$

Instead, we construct a ROM of it:

$$V_r^T M(\mu)V_r z_{r_j}(\mu) = V_r^T r_j(\mu), \tag{10}$$

so that $x_{r_j}(\mu) \approx \hat{x}_{r_j}(\mu) = V_r z_{r_j}(\mu)$. Finally,

$$|y_{ij}(\mu) - \hat{y}_{ij}(\mu)| \approx |C_i(\mu)\hat{x}_{r_j}(\mu)|.$$

Note that $\hat{x}_{r_j}(\mu)$ depends on $B_j(\mu)$, since $r_j(\mu)$ depends on $B_j(\mu)$. Each column $B_j(\mu)$ is associated with a $\hat{x}_{r_j}(\mu)$. The overall error of $\hat{y}(\mu)$ as a matrix can be estimated as:

$$\|y(\mu) - \hat{y}(\mu)\|_{\max} := \max_{i,j} |y_{ij}(\mu) - \hat{y}_{ij}(\mu)| \approx \max_{i,j} |C_i(\mu)\hat{x}_{r_j}(\mu)| =: \tilde{\Delta}(\mu). \tag{11}$$

$\tilde{\Delta}(\mu)$ defined in (11) is one of the error estimators proposed in Reference 20, where the proposed error estimators were shown to outperform other existing error estimators in the literature[19,29] in terms of both accuracy and computational efficiency. Note that $\tilde{\Delta}(\mu)$ in (11) is not guaranteed to be an error bound, therefore we use $\approx$ instead of $\leq$. Lemma 1 in Section 3.2.3 shows the accuracy of $\tilde{\Delta}(\mu)$. Furthermore, it has been discussed in Reference 20 that $\tilde{\Delta}(\mu)$ is even more accurate but has less computational complexity than other proposed estimators, including the one used in Reference 35. Even with this error estimator, the greedy algorithm could take several hours to converge for some complex systems, for example, a large-scale time-delay system presented in Section 4.3. For such systems, although the standard greedy algorithm can already be accelerated by the bi-fidelity greedy algorithm, we suggest a possibility to further improve the bi-fidelity greedy algorithm by introducing multi-fidelity error estimation.

We notice that in order to compute $\tilde{\Delta}(\mu)$, an extra projection matrix V_r has to be constructed for $\hat{x}_{r_j}(\mu)$. Although $\hat{x}_{r_j}(\mu)$ is dependent on the individual column of $B(\mu)$, the matrix V_r can be uniformly constructed based on the whole matrix $B(\mu)$. Then V_r is valid for any column of $B(\mu)$. It is proved in Reference 20 that taking $V_r = V$ leads to $\tilde{\Delta}(\mu)$ identically zero for all μ. Therefore, V_r should be additionally computed.

3.2.2 | Standard greedy algorithm using $\tilde{\Delta}(\mu)$

For easy understanding of the multi-fidelity error estimation, we first present Algorithm 3, the standard greedy algorithm using $\tilde{\Delta}(\mu)$ in (11) as the error estimator. There, some additional steps are added to compute V_r, see Steps 5, 7, and 8. In Step 7 of Algorithm 3, V_r is not only updated by $x(\mu^r)$, but also by V. This is due to the fact that the solution $x_{r_j}(\mu)$ to the residual system in (9) can be written as

$$\begin{aligned} x_{r_j}(\mu) &= M(\mu)^{-1} r_j(\mu) \\ &= M(\mu)^{-1}(B_j(\mu) - M(\mu)\hat{x}_j(\mu)) \\ &= M(\mu)^{-1} B_j(\mu) - V z_j(\mu) \\ &\approx \tilde{V}_r z_{r_j} - V z_j(\mu). \end{aligned} \tag{12}$$

It is clear that $x_{r_j}(\mu)$ is a linear combination of $(M(\mu))^{-1}B_j(\mu)$ and the columns of V. Therefore, V contributes to the subspace approximating the solution space of $x_{r_j}(\mu)$ and cannot be neglected. It is also noticed that $(M(\mu))^{-1}B_j(\mu)$ is in fact the solution $x_j(\mu)$ in (8), while $Vz_j(\mu)$ is $\hat{x}_j(\mu)$ that approximates $x_j(\mu)$. This means $x_{r_j}(\mu)$ is the difference between $x_j(\mu)$ and $\hat{x}_j(\mu)$, which is a nonzero vector. Therefore, we should compute another matrix $\tilde{V}_r$, so that $x_j(\mu) \approx \tilde{V}_r z_{r_j}(\mu)$, but $\tilde{V}_r z_{r_j}(\mu) \neq \hat{x}_j(\mu) = Vz_j(\mu)$. Finally, x_{r_j} is approximated by the difference between $\tilde{V}_r z_{r_j}(\mu)$ and $Vz_j(\mu)$. In other words, it is

Algorithm 3. Standard greedy algorithm using $\tilde{\Delta}(\mu)$ for linear parametric systems

Input: the FOM, a training set Ξ composed of parameter samples taken from the parameter domain $\mu \in \mathcal{P}$, error tolerance tol<1.

Output: Projection matrix V.

1: Choose initial parameter $\mu^* \in \Xi$ for V, and initial parameter $\mu^r \neq \mu^* \in \Xi$ for V_r.
2: $V \leftarrow \emptyset$, $V_r \leftarrow \emptyset$, $\varepsilon = 1$.
3: **while** ε>tol **do**
4: Compute the snapshot(s) $x(\mu^*)$ by solving the FOM, that is, $x(\mu^*) = (M(\mu^*))^{-1}B(\mu^*)$.
5: Compute the snapshot(s) $x(\mu^r)$ by solving the FOM, that is, $x(\mu^r) = (M(\mu^r))^{-1}B(\mu^r)$.
6: Update V by $V = \text{orth}\{V, x(\mu^*)\}$, (e.g., using the modified Gram-Schmidt process with deflation.)
7: Update V_r by $V_r = \text{orth}\{V, V_r, x(\mu^r)\}$.
8: Compute μ^r such that $\mu^r = \arg\max_{\mu\in\Xi} \max_{j=1,\ldots,n_I} \|r_j(\mu) - M(\mu)\hat{x}_{r_j}(\mu)\|$, ($n_I$ is the total number of columns of $B(\mu)$).
9: Compute μ^* such that $\mu^* = \arg\max_{\mu\in\Xi} \tilde{\Delta}(\mu)$.
10: $\varepsilon = \tilde{\Delta}(\mu^*)$.
11: **end while**

approximately represented as the linear combination of the columns of both V_r and V. This approximation also explains Steps 5 and 7 of Algorithm 3: Step 5 computes the reduced basis vectors contributing to $\tilde{V}_r$, Step 7 computes the complete reduced basis vectors contributing to V_r. New reduced basis vectors for both V and V_r are computed at each iteration of the greedy algorithm. Steps 8 and 9 compute the new *important* parameter samples for V_r and V, respectively. In general, μ^r should be different from μ^*, since $\tilde{\Delta}(\mu) \neq \max_{j=1,\ldots,n_I} \|r_j(\mu) - M(\mu)\hat{x}_{r_j}(\mu)\|$. Here, $r_j(\mu) - M(\mu)\hat{x}_{r_j}(\mu)$ is nothing but the residual induced by the approximate solution $(\hat{x}_{r_j}(\mu))$ to the residual system (9).

3.2.3 | Greedy algorithm with multi-fidelity error estimation

The computational complexity of Algorithm 3 using the error estimator $\tilde{\Delta}(\mu)$ comes from Steps 4–9. Efficiency of Step 9 can be improved by using the bi-fidelity error estimation as shown in Algorithm 2. The computations in Steps 4 and 6 are unavoidable, since V is used to compute the ROM of the original FOM and should be updated till acceptable error tolerance is satisfied. In contrast, V_r in Step 7 needs not be updated at every iteration. This implies that the ROM of the residual system does not have to be very accurate, since it is not the ROM that we seek, but an auxiliary ROM aiding the computation of $\tilde{\Delta}(\mu)$.

An immediate consequence of theorem 4.2 in Reference 20 for single-input and single-output systems is the following lemma for systems with multiple inputs and multiple outputs:

Lemma 1. *The error of the output $\hat{y}(\mu)$ of the ROM (6) can be bounded as*

$$\tilde{\Delta}(\mu) - \delta(\mu) \leq \|y(\mu) - \hat{y}(\mu)\|_{\max} \leq \tilde{\Delta}(\mu) + \delta(\mu), \tag{13}$$

where $\delta(\mu) := \max_{i,j} |C_i(\mu)(x_{r_j}(\mu) - \hat{x}_{r_j}(\mu))| \geq 0$.

Proof. From (7), we know

$$|y_{ij}(\mu) - \hat{y}_{ij}(\mu)| = |C_i(\mu)x_{r_j}(\mu)| \approx |C_i(\mu)\hat{x}_{r_j}(\mu)|.$$

Then

$$\begin{aligned} |y_{ij}(\mu) - \hat{y}_{ij}(\mu)| &= |C_i(\mu)x_{r_j}(\mu)| + |C_i(\mu)\hat{x}_{r_j}(\mu)| - |C_i(\mu)\hat{x}_{r_j}(\mu)| \\ &\leq |C_i(\mu)\hat{x}_{r_j}(\mu)| + \underbrace{|C_i(\mu)x_{r_j}(\mu) - C_i(\mu)\hat{x}_{r_j}(\mu)|}_{\delta_{ij}(\mu)}. \end{aligned} \tag{14}$$

On the other hand,

$$\begin{aligned}|C_i(\mu)\hat{x}_{r_j}(\mu)| &= |C_i(\mu)\hat{x}_{r_j}(\mu)| + |C_i(\mu)x_{r_j}(\mu)| - |C_i(\mu)x_{r_j}(\mu)| \\ &\le |C_i(\mu)x_{r_j}(\mu)| + \delta_{ij}(\mu). \end{aligned} \tag{15}$$

From (11), (15) and the definition of $\delta(\mu)$, we have

$$\tilde{\Delta}(\mu) = \max_{i,j} |C_i(\mu)\hat{x}_{r_j}(\mu)| \le \|y(\mu) - \hat{y}(\mu)\|_{\max} + \delta(\mu).$$

Similarly, from (14), we get

$$\|y(\mu) - \hat{y}(\mu)\|_{\max} \le \tilde{\Delta}(\mu) + \delta(\mu).$$

This completes the proof. ∎

From the definition of $\delta(\mu)$, it is seen that the more accurate the ROM of the residual system, the smaller $\delta(\mu)$ is. As a result, $\tilde{\Delta}(\mu)$ should measure the true error more accurately so that the *important* parameters it selects are closer to those selected by the true error, given the same training set Ξ. On the contrary, if the ROM of the residual system is less accurate, $\tilde{\Delta}(\mu)$ will be less accurate, too. However, at a certain stage, when $\tilde{\Delta}(\mu)$ is already small, the right-hand side of the residual system $r_j(\mu)$ will also be small, so that it can be expected that both $x_{r_j}(\mu)$ and $\hat{x}_{r_j}(\tilde{\mu})$ are close to zero. This leads to a small $\delta(\mu)$. Variation of a small $\delta(\mu)$ will not cause big variation of the difference between $\tilde{\Delta}(\mu)$ and the true error $\|y(\mu) - \hat{y}(\mu)\|_{\max}$. The trend, though not the exact route, of error decay could still be anticipated so that *important* parameters corresponding to the error peaks can also be detected. The above analyses are also justified by the numerical results in the next section, see, for example, Figures 4 and 11.

This motivates the multi-fidelity error estimation. We set a second tolerance $\epsilon >$tol, and when $\tilde{\Delta}(\mu) \le \epsilon < 1$, we stop updating the ROM of the residual system, that is, stop implementing Steps 5, 7, and 8 of Algorithm 3. The error estimator $\tilde{\Delta}(\mu)$ after this stage may not be as accurate as it would be when keep updating the ROM of the residual system. However, the difference should be small as $\tilde{\Delta}(\mu)$ is already below a small value ϵ. Without implementing Step 5, we have saved computations of simulating the FOM. For large and complex systems, solving the FOM even once is not fast. The computation in Step 7 is relatively cheap if the system is not very large. The computational cost in Step 8 is not low for certain complex problems, though some μ-independent parts of $r_j(\mu)$ and $\hat{x}_{r_j}(\mu)$ can be precomputed. For example, this is the case for the time-delay systems in the next section.

Stop updating the ROM of the residual system leads to V_r remaining unchanged. As V_r is a part of the high-fidelity error estimator $\tilde{\Delta}(\mu)$, this means that part of $\tilde{\Delta}(\mu)$ is not updated at later iteration steps of the greedy algorithm (see (10), (11), and (21)). Once V_r is not updated, $\tilde{\Delta}(\mu)$ automatically becomes a low-fidelity estimator $\tilde{\Delta}_l(\mu)$. When $\tilde{\Delta}_l(\mu)$ is combined with its RBF surrogate $\tilde{\Delta}^l_{\texttt{rbf}}(\mu)$, we obtain multi-fidelity error estimation in the greedy process. The multi-fidelity error estimation means that during the greedy process we use error estimators with different levels of fidelity to estimate the ROM error ($\tilde{\Delta}(\mu)$, $\tilde{\Delta}_l(\mu)$) and enrich Ξ_c ($\tilde{\Delta}_{\texttt{rbf}}(\mu)$, $\tilde{\Delta}^l_{\texttt{rbf}}(\mu)$). This is detailed in Algorithm 4 and illustrated in Figure 1. Compared with the standard greedy algorithm, the overall saving in computational costs is noticeable, which can be seen from the numerical results in the next section.

The concept of multi-fidelity error estimation could also be applied to other high-fidelity error estimators. For example, Step 15 could be modified as "Stop updating partial information of $\Delta(\mu)$," if some parts of the high-fidelity error estimator $\Delta(\mu)$ are not "essential" for computing $\Delta(\mu)$.

3.3 | Application to MOR for time-delay systems

In this section, we consider applying Algorithm 2, the greedy algorithm with bi-fidelity error estimation, Algorithm 3, the standard greedy algorithm and Algorithm 4, the greedy algorithm with multi-fidelity error estimation to large-scale

Algorithm 4. Greedy algorithm with multi-fidelity error estimation

Input: the FOM, a training set Ξ_c composed of a small number of parameter samples taken from the parameter domain $\mu \in \mathcal{P}$, a set Ξ_f composed of a large number of parameter samples of μ from $\mathcal{P}$, error tolerance tol<1, a second larger tolerance 1>ϵ>tol.

Output: Projection matrix V.

1: Choose initial parameter $\mu^* \in \Xi_c$ for V, and initial parameter $\mu^r \neq \mu^* \in \Xi_c$ for V_r.
2: $V \leftarrow \emptyset$, $V_r \leftarrow \emptyset$, $\varepsilon = 1$.
3: **while** ε>tol **do**
4: If $\varepsilon \leq \epsilon$ stop performing Steps 6, 8, and 10, such that $\tilde{\Delta}(\mu)$ is automatically degraded to a low-fidelity error estimator $\tilde{\Delta}_l(\mu)$ and $\tilde{\Delta}(\mu)$ in the later iterations is replaced by $\tilde{\Delta}_l(\mu)$.
5: Compute the snapshot(s) $x(\mu^*)$ by solving the FOM, that is, $x(\mu^*) = (M(\mu^*))^{-1}B(\mu^*)$.
6: Compute the snapshot(s) $x(\mu^r)$ by solving the FOM, that is, $x(\mu^r) = (M(\mu^r))^{-1}B(\mu^r)$.
7: Update V by $V = \text{orth}\{V, x(\mu^*)\}$ (e.g., using the modified Gram-Schmidt process with deflation).
8: Update V_r by $V_r = \text{orth}\{V, V_r, x(\mu^r)\}$.
9: Compute $\mu^* = \arg\max_{\mu\in\Xi_c} \tilde{\Delta}(\mu)$, compute $\mu^o = \arg\min_{\mu\in\Xi_c} \tilde{\Delta}(\mu)$.
10: Compute μ^r such that $\mu^r = \arg\max_{\mu\in\Xi_c} \max_{j=1,\ldots,n_I} \|r_j(\mu) - M(\mu)\hat{x}_{r_j}(\mu)\|$, % n_I is the total number of columns of $B(\mu)$.
11: Compute RBF surrogate $\tilde{\Delta}_{\text{rbf}}(\mu)$ of $\tilde{\Delta}(\mu)$ using values of $\tilde{\Delta}(\mu)$ corresponding to the samples of μ in Ξ_c via (3) and (4). Note that when $\tilde{\Delta}(\mu)$ is degraded to $\tilde{\Delta}_l(\mu)$, we actually compute the RBF surrogate $\tilde{\Delta}^l_{\text{rbf}}(\mu)$ of $\tilde{\Delta}_l(\mu)$.
12: Evaluate $\tilde{\Delta}_{\text{rbf}}(\mu)$ over Ξ_f and pick out a parameter μ_c from the large parameter set Ξ_f corresponding to the largest value of $\tilde{\Delta}_{\text{rbf}}(\mu)$, that is, $\mu_c = \arg\max_{\mu\in\Xi_f} \tilde{\Delta}_{\text{rbf}}(\mu)$.
13: Update the small parameter set Ξ_c: if $\tilde{\Delta}_{\text{rbf}}(\mu_c)$>tol, enrich Ξ_c with μ_c, that is, $\Xi_c = \{\Xi_c, \mu_c\}$, if $\tilde{\Delta}(\mu^o) \leq$tol, remove μ^o from Ξ_c: $\Xi_c = \Xi_c \backslash \mu^o$.
14: $\varepsilon = \tilde{\Delta}(\mu^*)$.
15: **end while**

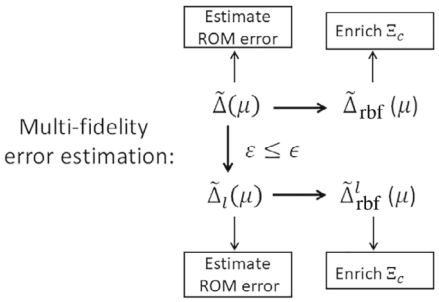

FIGURE 1 The multi-fidelity error estimation in Algorithm 4.

time-delay systems with many delays. The time-delay systems are defined as:

$$\begin{aligned} \sum_{j=0}^{d} E_j \dot{x}(t-\tau_j) &= \sum_{j=0}^{d} A_j x(t-\tau_j) + Bu(t), \qquad \forall\ t \geq 0 \\ y(t) &= Cx(t), \end{aligned} \tag{16}$$

with an initial condition $x(t) = \Phi(t) \in \mathbb{C}^n, \forall\ t \in [-\tau_d, 0]$. Here, $E_0, \ldots, E_d, A_0, \ldots, A_d \in \mathbb{C}^{n\times n}, B \in \mathbb{C}^{n\times n_I}, C \in \mathbb{C}^{n_o\times n}$, $0 = \tau_0 < \tau_1 < \cdots < \tau_d$ and n is the order of the delay system. The transfer function of the delay system is defined as:

$$H(s) = C\mathcal{K}^{-1}(s)B, \tag{17}$$

where $\mathcal{K}(s) = s\sum_{j=0}^{d} E_j e^{-s\tau_j} - \sum_{j=0}^{d} A_j e^{-s\tau_j}$, $s = 2\pi\jmath$ is the variable in the frequency domain, f is the ordinary frequency with unit Hz and $\jmath$ is the imaginary unit. If the system has multiple input ports and output ports, then

$n_I > 1$, $n_O > 1$, $H(s)$ is actually a matrix. The i,jth entry of $H(s)$ is the transfer function corresponding to input port j and output port i.

A ROM of the delay system, which has the same delays as the original system, can be obtained via Galerkin projection using a projection matrix $V \in \mathbb{R}^{n\times r}$, $r \ll n$, that is,

$$\begin{aligned} \sum_{j=0}^{d} \hat{E}_j \dot{z}(t-\tau_j) &= \sum_{j=0}^{d} \hat{A}_j z(t-\tau_j) + \hat{B}u(t), \qquad \forall\, t \geq 0, \\ \hat{y}(t) &= \hat{C}z(t), \end{aligned} \tag{18}$$

where $\hat{E}_j = V^T E_j V \in \mathbb{R}^{r\times r}, \hat{A}_j = V^T A_j V \in \mathbb{R}^{r\times r}, \hat{B} = V^T B \in \mathbb{R}^{r\times n_I}, \hat{C} = CV \in \mathbb{R}^{n_O\times r}$, with $r \ll n$ being the order of the ROM. The original state $x(t)$ in (16) can be recovered by the approximation: $x(t) \approx Vz(t)$. The transfer function of the ROM is

$$\hat{H}(s) = \hat{C}\hat{\mathcal{K}}^{-1}(s)\hat{B}, \tag{19}$$

where $\hat{\mathcal{K}}(s) = s\sum_{j=0}^{d} \hat{E}_j e^{-s\tau_j} - \sum_{j=0}^{d} \hat{A}_j e^{-s\tau_j}$. The projection matrix V can be constructed via approximating $H(s)$[36] as follows. Note that $H(s)$ is nothing but the output $y(\mu)$ of the linear parametric system in (5), with $M(\mu) = \mathcal{K}(s)$, $B(\mu) = B$ and $\mu = s$, that is,

$$\begin{aligned} \mathcal{K}(s)x(s) &= B, \\ H(s) &= C(s)x(s). \end{aligned} \tag{20}$$

The reduced transfer function $\hat{H}(s)$ is the output $\hat{y}(\mu)$ of the ROM in (6) with $\hat{M}(\mu) = \hat{K}(s)$ and $\hat{B}(\mu) = \hat{B}$.

It is easy to see that the projection matrix V used to construct the ROM (18) in the time domain is exactly the same matrix to obtain the reduced transfer function $\hat{H}(s)$. Therefore, V can be obtained by constructing a ROM of system (20) in the frequency domain, that is, by approximating the transfer function $H(s)$. This can be done by the standard greedy Algorithm 3 with the error estimator $\tilde{\Delta}(s)$, where V is iteratively computed by choosing proper samples of s.[16,36] In fact, the reduced transfer function $\hat{H}(s)$ interpolates the original transfer function $H(s)$ at the selected samples of s.[16] The matrix $M(\mu)$ in Steps 4 and 5 of Algorithm 3 is now replaced by $\mathcal{K}(s)$. The difference of the coefficient matrix $\mathcal{K}(s)$ from a single matrix $M(\mu)$ in the usual case is its high complexity. To solve the system in (20) is much more expensive than solving the system in (5) where $M(\mu)$ is a single matrix. On the one hand, the matrices constituting $\mathcal{K}(s)$ must be assembled to get $\mathcal{K}(s)$. On the other hand, the finally assembled matrix has some dense blocks, though each single matrix contributing to $\mathcal{K}(s)$ is sparse.

To further improve the efficiency of the standard greedy algorithm, we propose to apply Algorithms 2 and 4 to time-delay systems. The application is straightforward by simply replacing the FOM in (5) in both algorithms with the system in (20), that is, the matrix $M(\mu)$ is replaced by $\mathcal{K}(s)$, the input matrix $B(\mu)$ and the output matrix $C(\mu)$ are replaced by B and C in (20), respectively.

3.3.1 | Analysis on computational complexity

In this section, we present some analyses on computational complexity of a FOM solve, a ROM solve and that of estimating the error at a single parameter sample for the time-delay systems we considered. The matrix $\mathcal{K}(s) \in \mathbb{R}^{n\times n}$ in (17) consists of $2d$ matrices E_j and A_j, $j = 1, \ldots, d$. For the corresponding reduced-order model, the reduced matrix $\hat{\mathcal{K}}(s) \in \mathbb{R}^{r\times r}, r \ll n$ in (18) consists of the corresponding reduced matrices $\hat{E}_j, \hat{A}_j, j = 1, \ldots, d$. Here, $r \ll n$ is the size of the ROM. To solve the FOM in (20) and get $H(s) = C\mathcal{K}^{-1}(s)B$ at each single sample of s, $\mathcal{K}(s)$ is first assembled by E_j, A_j. This costs $O(n^2)$ operations. Then $\mathcal{K}^{-1}(s)B$ is computed with complexity $O(n^3)$, as $\mathcal{K}(s)$ is not sparse, though each E_j, A_j is sparse. Finally, it is multiplied by C with $O(n)$ complexity. The total computational complexity is: $O(n^2 + n^3)$. The complexity of solving the corresponding ROM $\hat{\mathcal{K}}(s)z(s) = \hat{B}$ at each sample of s and getting the reduced transfer function $\hat{H}(s) = \hat{C}\hat{\mathcal{K}}^{-1}(s)\hat{B}$ is straightforward to know, that is, $O(r^2 + r^3)$, including $O(r^2)$ operations of assembling the reduced matrix $\hat{\mathcal{K}}(s)$ from the reduced matrices $\hat{E}_j, \hat{A}_j$.

To assess the complexity of computing the error estimator at each single sample of s, we first see that the error estimator is defined in (11). It can be expanded in the form as below:

$$\tilde{\Delta}(s) = \|CV_r(V_r\mathcal{K}(s)V_r)^{-1}V_r^T\left[B - \mathcal{K}(s)V(V^T\mathcal{K}(s)V)^{-1}V^TB\right]\|_{\max}.$$

Here, $\|\cdot\|_{\max}$ is the matrix max-norm defined as the maximal magnitude of matrix entries. Note that $\mathcal{K}(s)$ is affinely dependent on the s-independent terms E_i, A_i. The above expansion can be further written as

$$\tilde{\Delta}(s) = \|CV_r(V_r\mathcal{K}(s)V_r)^{-1}\left[V_r^TB - V_r^T\mathcal{K}(s)V(V^T\mathcal{K}(s)V)^{-1}V^TB\right]\|_{\max}, \tag{21}$$

where the s-independent terms $VE_iV, V_rE_iV, V_rE_iV_r, VA_iV, V_rA_iV, V_rA_iV_r, i = 1, \ldots, d$ in $V\mathcal{K}(s)V, V_r\mathcal{K}(s)V, V_r\mathcal{K}(s)V_r$ can be precomputed, so that later, $V\mathcal{K}(s)V, V_r\mathcal{K}(s)V, V_r\mathcal{K}(s)V_r$ need only be assembled by the precomputed s-independent terms at each s sample. CV, CV_r, V_r^TB, V^TB are also s-independent and can be precomputed. The cost of computing the error estimator $\tilde{\Delta}(s)$ at a single sample of s is then counted given the above *already precomputed* reduced matrices. Finally, evaluating the error estimator at each sample of s has $O(r^3)$ complexity. Note that the error estimator is computed only during the offline greedy process of constructing the reduced basis V for the ROM. After V is constructed, the final ROM can be computed from the projected reduced matrices $\hat{E}_j = V^TE_jV, \hat{A}_j = V^TA_jA$. During each iteration of the greedy algorithm, the error estimator $\tilde{\Delta}(s)$ or its low-fidelity version $\tilde{\Delta}_l(s)$ is evaluated at all the samples in the training set Ξ (for standard greedy) or Ξ_c (for multi-fidelity greedy), depending on which algorithm is used. In summary,

- complexity of one FOM solve: $O(n^3)$.
- complexity of one ROM solve: $O(r^3)$.
- complexity of one estimation of the error: $O(r^3)$.

Numerical results in Figure 9 in Section 4.1 show the corresponding computational cost (runtime) of one ROM solve and one $\tilde{\Delta}(\mu)$ or $\tilde{\Delta}_l(\mu)$ evaluation w.r.t the ROM size. The cost of one FOM solve is also given.

4 | NUMERICAL TESTS

We consider three time-delay systems obtained from partial element equivalent circuit modeling and simulation, which transfer problems from the electromagnetic domain to the circuit domain.[37-40] When the propagation delays are explicitly kept for both partial inductances and coefficients of potential, time-delay systems can be derived.[41] Numerical tests are done with MATLAB R2021a on a computer server with two AMD EPYC 7763 64-core processors (each core has 32kB L1-cache, 512kB L2-cache, eight core share 32MB L3-cache) with hyper-threading. 1TB main memory, split into two 512 GB. Each CPU socket controls one part (NUMA architecture), SSD RAID 1 with 1TB. The code and data are available at: https://zenodo.org/record/7892188.

We test the standard greedy Algorithm 3, the bi-fidelity greedy Algorithm 2 and the multi-fidelity Algorithm 4 on three time-delay systems. The high-fidelity error estimator for all algorithms is $\tilde{\Delta}(\mu)$ in (11). To run the algorithms, we need to initialize the algorithms by doing the following:

- The samples in the training set Ξ, the small set Ξ_c and the large set Ξ_f are taken from the prescribed frequency domain and are generated using the MATLAB function `linspace`: `linspace`(f_0, f_1, cardi). Here, f_0 is the lowest frequency, f_1 is the highest frequency used in `linspace`, cardi is the corresponding cardinality of each set. The samples of s are then computed using the relation: $s = 2\pi jf$.
- For the multi-fidelity error estimation, we set $\epsilon = 0.1$ in Step 15 of Algorithm 4. For all algorithms, the error tolerance $\varepsilon = 0.001$.
- To compute the RBF surrogate, we choose the inverse multiquadratic RBF (IMQ) $\Phi = \frac{1}{1+(a\|\mu-\mu_i\|)^2}$ with the shape parameter $a = 30$.

We also need to define some variables uniformly used in all the tables and figures:

- The error $\max_{s\in\Xi_{\texttt{test}}}\|H(s) - \hat{H}(s)\|_{\max}$ of the transfer function $\hat{H}(s)$ of the ROM is finally computed over samples of $s \in \Xi_{\texttt{test}}$ drawn independently of the training sets, resulting in the validated error: Valid.err in Tables 1–7.

TABLE 1 Three-port divider: $n = 10{,}626$, $d = 93$ delays, tol = 0.001, adding/removing a single sample at each iteration.

Method	Iterations	Runtime (min)	r	Valid.err
Algorithm 3 (standard, $\lvert\Xi\rvert = 40$)	14	14.7	84	9.2×10^{-4}
Algorithm 2 (bi-fidelity, add only, $\lvert\Xi_c\rvert = 20$)	14	11.5	84	6×10^{-4}
Algorithm 2 (bi-fidelity, add-remove, $\lvert\Xi_c\rvert = 20$)	14	10.5	84	0.0022
Algorithm 4 (multi-fidelity, add only, $\lvert\Xi_c\rvert = 15$)	14	7.3	84	0.0013
Algorithm 4 (multi-fidelity, add-remove, $\lvert\Xi_c\rvert = 15$)	15	6.7	90	3.4×10^{-4}

TABLE 2 Three-port divider: $n = 10{,}626$, $d = 93$ delays, tol = 0.001, smaller $\lvert\Xi\rvert$ and $\lvert\Xi_c\rvert$, adding/removing a single sample at each iteration.

Method	Iterations	Runtime (min)	r	Valid.err
Algorithm 3 (standard, $\lvert\Xi\rvert = 30$)	14	12.9	84	0.0017
Algorithm 2 (bi-fidelity, add only, $\lvert\Xi_c\rvert = 15$)	13	10.1	78	0.0026
Algorithm 2 (bi-fidelity, add-remove, $\lvert\Xi_c\rvert = 15$)	13	9.0	78	0.0088

TABLE 3 Three-port divider: $n = 10{,}626$, $d = 93$ delays, tol = 0.001, adding/removing $n_{\text{add}} = n_{\text{del}} > 1$ samples at each iteration.

Method	Iterations	Runtime (min)	r	Valid.err
Algorithm 2 (bi-fidelity, add-remove, $\lvert\Xi_c\rvert = 15$, $n_{\text{add}} = 2$)	14	9.6	84	0.0039
Algorithm 2 (bi-fidelity, add-remove, $\lvert\Xi_c\rvert = 20$, $n_{\text{add}} = 2$)	14	10.6	84	0.0022
Algorithm 2 (bi-fidelity, add-remove, $\lvert\Xi_c\rvert = 20$, $n_{\text{add}} = 5$)	14	10.6	84	0.0022
Algorithm 4 (multi-fidelity, add-remove, $\lvert\Xi_c\rvert = 15$, $n_{\text{add}} = 2$)	12	6.2	84	0.0092
Algorithm 4 (multi-fidelity, add-remove, $\lvert\Xi_c\rvert = 15$, $n_{\text{add}} = 5$)	12	6.2	84	0.0092

TABLE 4 Three coplanar microstrips: $n = 16{,}644$, $d = 168$ delays, tol = 0.001, adding/removing a single sample at each iteration.

Method	Iterations	Runtime (min)	r	Valid.err
Algorithm 3 (standard, $\lvert\Xi\rvert = 30$)	11	50.7	132	8.5×10^{-4}
Algorithm 2 (bi-fidelity, add only, $\lvert\Xi_c\rvert = 10$)	11	27.6	132	0.0033
Algorithm 2 (bi-fidelity, add-remove, $\lvert\Xi_c\rvert = 10$)	11	26.5	132	8.2×10^{-4}
Algorithm 4 (multi-fidelity, add only, $\lvert\Xi_c\rvert = 10$)	11	21.6	132	0.0033
Algorithm 4 (multi-fidelity, add-remove, $\lvert\Xi_c\rvert = 10$)	11	20.1	132	8.2×10^{-4}

TABLE 5 Three coplanar microstrips: $n = 16{,}644$, $d = 168$ delays, tol = 0.001, larger $\lvert\Xi_c\rvert$, adding/removing a single sample at each iteration.

Method	Iterations	Runtime (min)	r	Valid.err
Algorithm 2 (bi-fidelity, add only, $\lvert\Xi_c\rvert = 15$)	11	33.7	132	0.0011
Algorithm 4 (multi-fidelity, add only, $\lvert\Xi_c\rvert = 15$)	12	24.9	144	0.0027

TABLE 6 Three coplanar microstrips: $n = 16{,}644$, $d = 168$ delays, tol = 0.001, adding/removing $n_{\text{add}} = n_{\text{del}} > 1$ samples at each iteration.

Method	Iterations	Runtime (min)	r	Valid.err
Algorithm 2 (bi-fidelity, add-remove, $\lvert\Xi_c\rvert = 10$, $n_{\text{add}} = 2$)	10	22.0	120	0.019
Algorithm 2 (bi-fidelity, add-remove, $\lvert\Xi_c\rvert = 10$, $n_{\text{add}} = 5$)	10	22.1	120	0.019
Algorithm 4 (multi-fidelity, add-remove, $\lvert\Xi_c\rvert = 10$, $n_{\text{add}} = 2$)	10	19.6	120	0.019
Algorithm 4 (multi-fidelity, add-remove, $\lvert\Xi_c\rvert = 10$, $n_{\text{add}} = 5$)	10	19.0	120	0.019
Algorithm 4 (multi-fidelity, add-remove, $\lvert\Xi_c\rvert = 15$, $n_{\text{add}} = 2$)	11	21.3	156	0.010

TABLE 7 Microstrip filter: $n = 50{,}904$, $d = 192$ delays, tol = 0.001, adding/removing a single sample at each iteration.

Method	Iterations	Runtime (h)	r	Valid.err
Algorithm 3 (standard, $\lvert\Xi\rvert = 30$)	12	4.9	48	3.2×10^{-4}
Algorithm 4 (multi-fidelity, add-remove, $\lvert\Xi_c\rvert = 15$)	13	3.0	52	5.7×10^{-4}
Algorithm 4 (multi-fidelity, add-remove, $\lvert\Xi_c\rvert = 10$)	10	2.2	40	0.0024

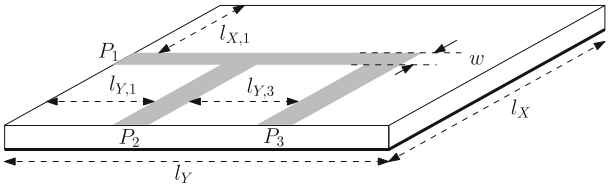

FIGURE 2 The three-port microstrip power-divider circuit.

- Runtime, the walltime of each algorithm till convergence.
- Iter., the total number of iterations of each algorithm.
- r, the order of the ROM.
- The true error at each iteration of Algorithm 3 is defined over Ξ, that is, $\max_{s\in\Xi} \|H(s) - \hat{H}(s)\|_{\max}$.
- The true error at each iteration of Algorithm 2 or Algorithm 4 is defined over Ξ_c, that is, $\max_{s\in\Xi_c} \|H(s) - \hat{H}(s)\|_{\max}$.

Note that Ξ_c could be enriched only by adding samples from Ξ_f to Ξ_c. As the high-fidelity error estimator $\tilde{\Delta}(\mu)$ needs to be computed at every sample in Ξ_c at each iteration, samples in Ξ_c whose corresponding error is already smaller than tol can also be removed from Ξ_c to keep the cardinality of Ξ_c constant, so that more computations can be saved. We consider both cases separately and compare their efficiency with respect to both runtime and accuracy.

4.1 | Test 1: results for a model of three-port divider

The model structure of a three-port microstrip power-divider circuit is shown in Figure 2 (P_1, P_2 and P_3 denote the ports). The dimensions of the circuit are [20, 20, 0.5] mm in the $[x, y, z]$ directions and the width of the microstrips is set as 0.8 mm. Furthermore, the dimensions l_{X1}, l_{Y1}, and l_{Y3} are 9, 7.2, and 7.2 mm, respectively. The relative dielectric constant is $\varepsilon_r = 2.2$. All the ports are terminated on 50 Ω resistances. The system has three input ports and three output ports with order $n = 10{,}626$, and it has $d = 93$ delays. The interesting frequency band is [0, 20] GHz.

For this model, $|\Xi| = 30$ or $|\Xi| = 40$ for the standard greedy Algorithm 3. For Algorithm 2 and Algorithm 4, $|\Xi_c| = 15$ or $|\Xi_c| = 20$ and $|\Xi_f| = 100$. The set Ξ_c is then updated during the iteration of the greedy algorithm. The samples in Ξ, Ξ_c or Ξ_f are generated using the MATLAB function `linspace`, with $f_0 = 1 \times 10^6$ Hz, $f_1 = 2 \times 10^{10}$ Hz for Algorithms 2 and 3 and $f_0 = 1 \times 10^8$ Hz or 1×10^5 Hz, $f_1 = 2 \times 10^{10}$ Hz for Algorithm 4. The 1000 samples for validating the ROM accuracy are created using the MATLAB function `logspace`, that is, `logspace`$(\log_{10}(f_l), \log_{10}(f_h), 1000)$. $f_l = 1 \times 10^4$ Hz and $f_h = 2 \times 10^{10}$ Hz.

In Table 1, we list the results of the three algorithms. The standard greedy algorithm is the slowest. The other algorithms are all faster. The bi-fidelity greedy algorithm by enriching Ξ_c only is slower than other bi-(multi-)fidelity algorithms, this is in agreement with our theoretical analysis in Section 3. The multi-fidelity algorithm by adding and removing a single sample to and from Ξ_c performs the best in terms of runtime and accuracy. Compared to the standard algorithm, it has reduced the offline runtime from 14.7 to 6.7 min. Finally, a speed-up factor 2.2 is achieved. Except for the bi-fidelity algorithm by adding and removing samples, the other algorithms have produced ROMs with validated errors below the tolerance or very close to the tolerance.

It is worth pointing out that if using fewer samples in Ξ for the standard greedy algorithm, the ROM has a validated error that is slightly larger than the tolerance, as shown in Table 2, where $|\Xi| = 30$. Also, the bi-fidelity greedy algorithms are less accurate if using fewer samples in Ξ_c, as shown in Table 2. There, the same Ξ_c for the multi-fidelity greedy algorithms is used, but less accurate ROMs are obtained.

In Table 3, we show the results of the bi-fidelity greedy algorithm and the multi-fidelity greedy algorithm when $n_{\text{add}} = n_{\text{del}} > 1$ samples are added or removed from the small training set Ξ_c at each iteration of the algorithm. The bi-fidelity algorithms produce similar results as those in Tables 1 and 2 given the same Ξ_c. For $|\Xi_c| = 15$, the bi-fidelity greedy algorithm with $n_{\text{add}} = n_{\text{del}} = 2$ converges in 14 iterations, running one more iteration than with $n_{\text{add}} = n_{\text{del}} = 1$ as shown in Table 2, and generates a ROM with slightly higher accuracy. On the contrary, given $|\Xi_c| = 15$, the multi-fidelity greedy algorithm with either $n_{\text{add}} = n_{\text{del}} = 2$ or $n_{\text{add}} = n_{\text{del}} = 5$ runs three iterations less than in the case of adding/removing a single sample as shown in Table 1, and constructs ROMs with lower accuracy. Furthermore, it is seen that increasing $n_{\text{add}} = n_{\text{del}}$ from 2 to 5 did not change the results for both algorithms. In general, adding/removing a single sample keeps the algorithms simple but efficient.

In Tables 1–3, except for the multi-fidelity greedy process with "add-remove" in Table 1, all other multi-fidelity greedy processes use $f_0 = 1 \times 10^8$ Hz and $f_1 = 2 \times 10^{10}$ Hz to generate the course training set Ξ_c. We find that using $f_0 = 1 \times 10^5$ Hz instead of $f_0 = 1 \times 10^8$ Hz, the multi-fidelity greedy process by adding/removing a single sample gives even more accurate results, as shown in Table 1. If $f_0 = 1 \times 10^8$ Hz is used instead, it can still achieve the accuracy with maximal error 0.0013 over the 1000 validation samples, which is already very close to the tolerance.

To illustrate the behavior of the error estimators further, we plot the decay of error estimators and their corresponding true errors during the greedy iterations. Since different μ^* are chosen according to different error estimators, the projection matrix V is updated with different snapshots, leading to ROMs with different accuracy. Consequently, the true errors of the ROMs are expected to be different.

Figures 3 and 4 are the results of the algorithms in Table 1. The left part of Figure 3 shows the error of the high-fidelity error estimator at each iteration of Algorithm 3 and the decay of the corresponding true error. The error estimator almost

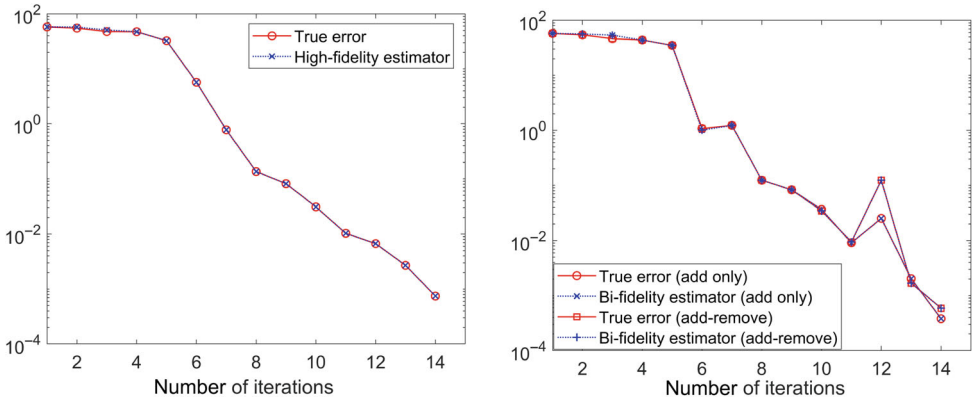

FIGURE 3 Error decay for divider model. Left: true error versus high-fidelity error estimation. Right: true error versus bi-fidelity error estimation.

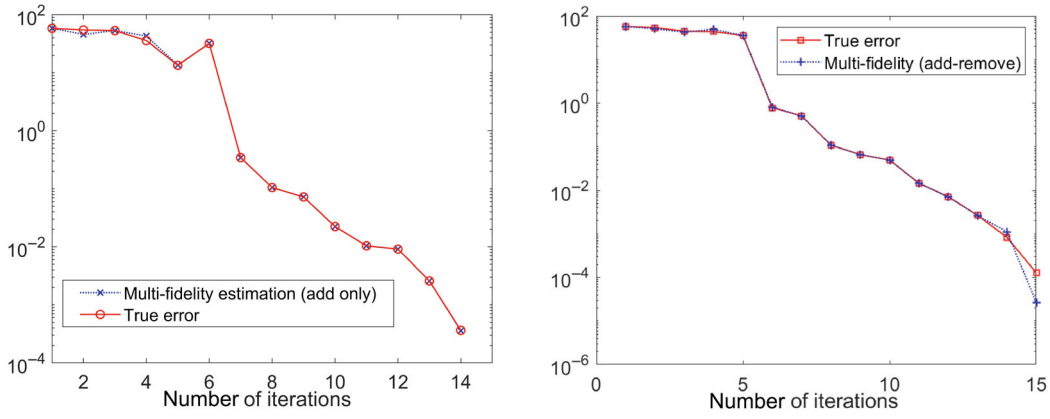

FIGURE 4 Error decay for divider model. Left: True error versus multi-fidelity error estimation (add only). Right: True error versus multi-fidelity error estimation (add/remove a single sample each iteration).

exactly matches the true error at all the iterations. The right part of Figure 3 plots the decay of the bi-fidelity error estimator with respect to the true error. The bi-fidelity error estimator in both of the two cases: only adding (add-only) samples to Ξ_c, adding and removing (add-remove) samples to and from Ξ_c, can accurately catch the true error. Both cases converge in 14 iterations, but the case "add-only" is more accurate as can be seen from Table 1.

Figure 4 plots the decay of the multi-fidelity error estimator and the corresponding true error decay. For clarity, the two cases "add-only" and "add-remove" are plotted in two separate figures. The multi-fidelity error estimation with "add-remove" is not as accurate as multi-fidelity error estimation with "add-only" at the last two iterations.

In Figure 5, we compare the prediction error of the standard greedy algorithm (Algorithm 3) with the multi-fidelity greedy procedure (Algorithm 4) as well as with a random sampling (with uniform distribution) procedure. The prediction error of each method is the error of the ROM generated at each iteration of the corresponding process. The error is computed over a test set $\Xi_{\texttt{test}}$ including the same 1000 frequency samples used for error validation in Tables 1–3, that is, $\text{Error}_{\texttt{pred}} = \max_{s\in\Xi_{\texttt{test}}} \|H(s) - \hat{H}(s)\|_{\max}$. For the random sampling procedure, we add samples of frequency one after another to enrich the reduced basis. Each time of adding one new random sample, we generate a new ROM and check the error of the ROM as compared to the original system. At the 15-iteration, both greedy procedures converge. But the random sampling process more or less stagnates. A most important advantage of greedy process over random sampling is that there is a stopping criterion given by a reliable error estimator. For random sampling, we never know when to stop or in other words, we never know how many samples should we use.

We do not expect that RBF surrogate $\tilde{\Delta}_{\texttt{rbf}}(\mu)$ can accurately generalize the error estimator $\tilde{\Delta}(\mu)$, rather it is sufficient that it can detect some samples in Ξ_f corresponding to peaks of the high-fidelity error estimator $\tilde{\Delta}(\mu)$. Those samples corresponding to the $\tilde{\Delta}(\mu)$-peaks are exactly what we want to include into the training set Ξ_c and to enrich it. We plot the prediction error of RBF over Ξ_f at some individual iteration steps of the multi-fidelity greedy algorithm (Algorithm 4) in Figures 6–8. We can see that the frequency locations corresponding to the peaks of the RBF interpolation are often close to those locations corresponding to the peaks of the error indicator. That means RBF could choose meaningful frequency samples corresponding to peaks of $\tilde{\Delta}(\mu)$. In particular, when we add only a single frequency sample at each iteration, only the one corresponding to the highest peak is chosen to enrich Ξ_c. It also happened that at a certain iteration step, the RBF interpolation could not correctly detect the highest error peak, then it may not contribute to the efficiency at this iteration. However, the overall contribution of RBF during the whole greedy iterations can still be seen from those figures.

The wall time of solving the FOM at a single parameter sample is around 11 s, including 6–7.2 s for assembling the matrix $\mathcal{K}(s)$ from the s-independent matrices $E_i, A_i, i = 1, \ldots, d$ and 5.5–7.8 s for solving the multiple right-hand-side linear system $\mathcal{K}(s)x(s) = B$ and getting $H(s) = C\mathcal{K}(s)^{-1}B$. This is somehow different from the cost analysis in Section 3.3.1. Theoretically, assembly $\mathcal{K}(s)$ would be less expensive than solving the linear system $\mathcal{K}(s)x(s) = B$. However, the runtime gives a slightly different conclusion. In Figure 9, we present the wall time of one ROM solve, one high-fidelity error estimator ($\tilde{\Delta}(s)$) evaluation in the standard greedy process, as well as one $\tilde{\Delta}(s)$ or $\tilde{\Delta}_l(s)$ evaluation in the multi-fidelity greedy process w.r.t. the ROM size r. The runtime of estimating the error is a bit longer, as can be seen from the above

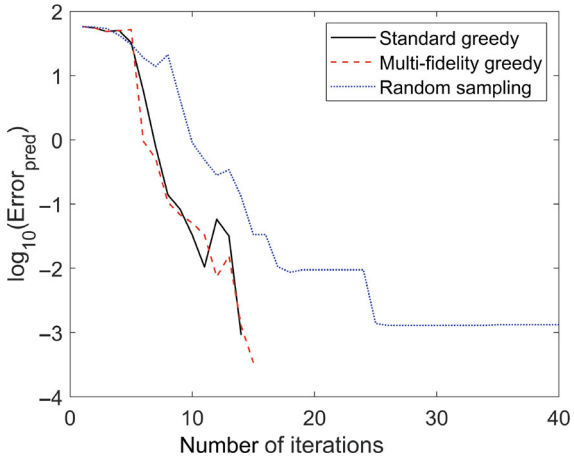

FIGURE 5 Divider model: prediction error over a testing frequency set of 1000 samples: prediction error of the standard greedy algorithm with high-fidelity error estimation versus prediction error of the greedy algorithm with multi-fidelity error estimation and prediction error of random sampling.

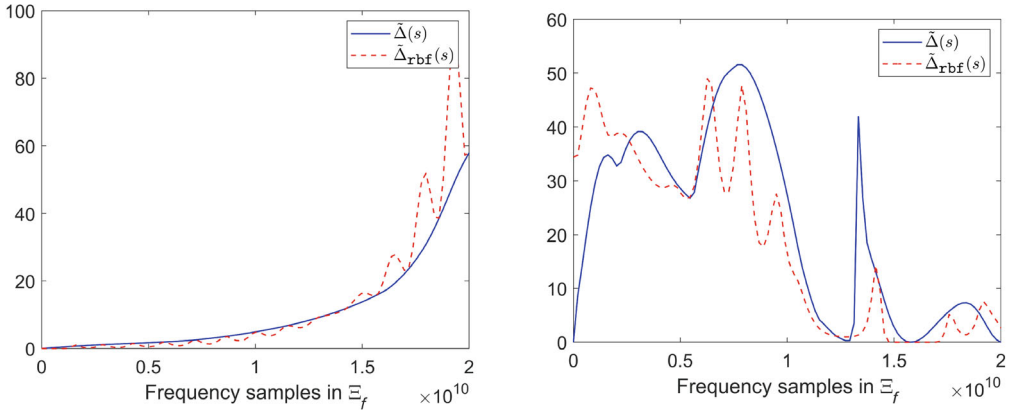

FIGURE 6 Divider model: radial basis function (RBF) surrogate estimator $\tilde{\Delta}_{\mathtt{rbf}}(s)$ versus the high-fidelity estimator $\tilde{\Delta}(s)$ over the fine parameter set Ξ_f at the first (left) and fourth (right) iteration step of the multi-fidelity greedy algorithm by add-removing a single sample at each iteration.

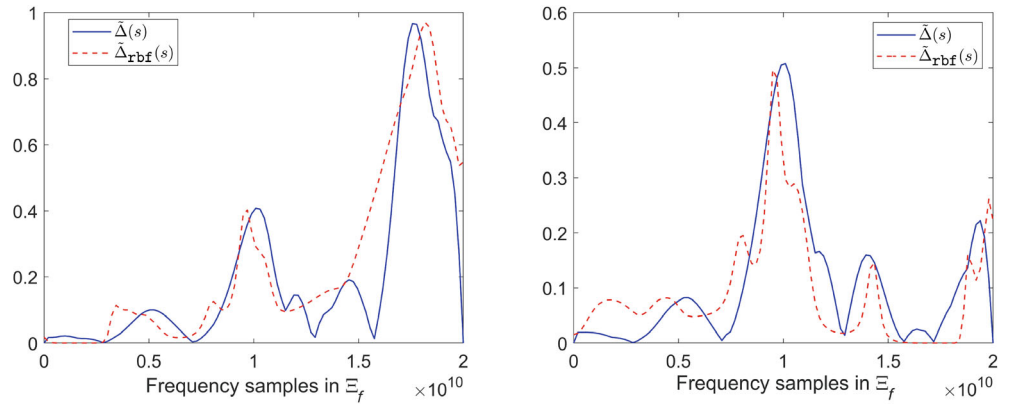

FIGURE 7 Divider model: radial basis function (RBF) surrogate estimator $\tilde{\Delta}_{\mathtt{rbf}}(s)$ versus the high-fidelity estimator $\tilde{\Delta}(s)$ over the fine parameter set Ξ_f at the sixth (left) and seventh (right) iteration step of the multi-fidelity greedy algorithm by add-removing a single sample at each iteration.

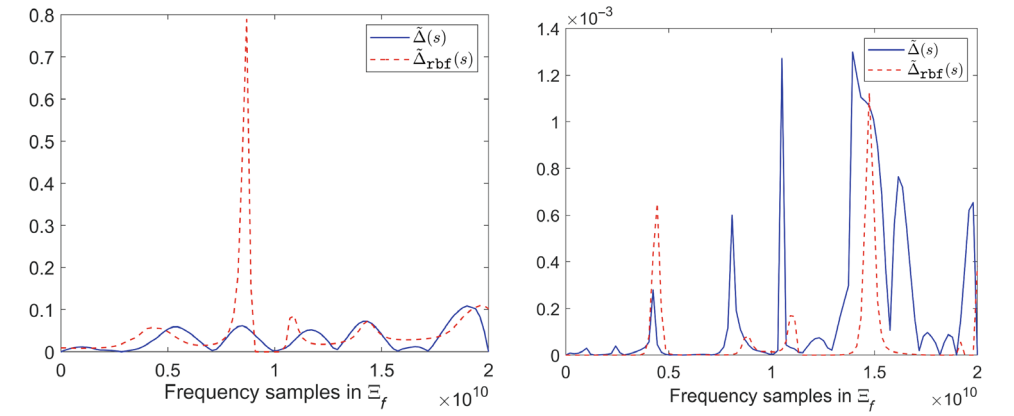

FIGURE 8 Divider model: radial basis function (RBF) surrogate estimator $\tilde{\Delta}_{\mathtt{rbf}}(s)$ vs the high-fidelity estimator $\tilde{\Delta}(s)$ over the fine parameter set Ξ_f at the eighth (left) and 12th (right) iteration step of the multi-fidelity greedy algorithm by add-removing a single sample at each iteration.

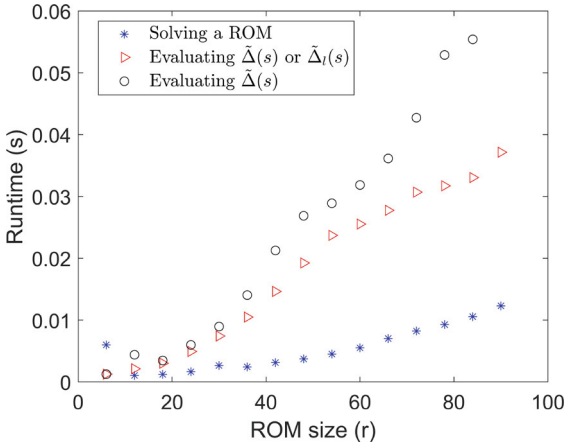

FIGURE 9 Divider model: runtime of solving the ROM, evaluating the high-fidelity error estimator, and runtime of evaluating the multi-fidelity error estimator during each iteration of the corresponding greedy algorithm. The ROM is the one generated by the multi-fidelity greedy algorithm: Algorithm 4.

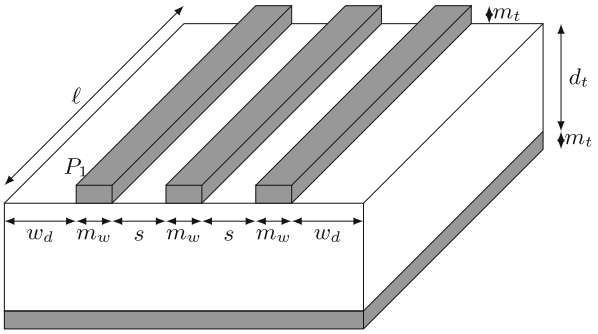

FIGURE 10 Three coplanar microstrips.

expansion of $\tilde{\Delta}(s)$, where two small matrix inverses need to be computed. To solve the ROM, only one small system with size r needs to be solved. Moreover, the column size of V_r is usually double of the column size of V, leading to a larger matrix $V_r\mathcal{K}(s)V_r$ than $V\mathcal{K}(s)V$ for the ROM solve. In the beginning stage of the multi-fidelity greedy algorithm, the high-fidelity error estimator $\tilde{\Delta}(s)$ is computed at each iteration. At the later stage, V_r in $\tilde{\Delta}(s)$ needs not be updated, so that $\tilde{\Delta}(s)$ is degraded to $\tilde{\Delta}_l(s)$. Evaluating the low-fidelity error estimator $\tilde{\Delta}_l(s)$ is cheaper, as the affine terms in $V_r\mathcal{K}(s)V_r$ need not be updated. Moreover, the size of $V_r\mathcal{K}(s)V_r$ does not increase anymore. This reduces the cost of evaluating the estimator during multi-fidelity greedy algorithm.

4.2 | Test 2: results for a model of coplanar microstrips

The second example is a model of a three coplanar microstrips structure shown in Figure 10. The width of the metal strips is $m_w = 0.178$ mm, the thickness of metal strips and ground plane is $m_t = 0.035$ mm while the left and right wing of the microstrips are $w_d = 3$ mm. Finally, the length of each strip is $\ell = 5$ cm, the thickness of the dielectric is $d_t = 0.8$ mm, and the spacing between two strips is $s = 0.3$ mm. The relative dielectric constant is set to be $\varepsilon_r = 4$ and the conductivity of the metal is assumed to be $\sigma = 5.8^7$ S/m. The six ports, located between the ends of each strip and the ground plane below, are terminated on load resistors $R_{\text{load}} = 50\ \Omega$, resulting in a system with six input ports and six output ports. The order of the FOM is $n = 16{,}644$, and there are $d = 168$ delays. The frequency band of interest is $[0, 10]$ GHz.

We set 30 samples for Ξ in the standard greedy Algorithm 3, that is, $|\Xi| = 30$. For Algorithms 2 and 4, $|\Xi_c| = 10$ or $|\Xi_c| = 15$, and $|\Xi_f| = 100$. The samples in Ξ, Ξ_c or Ξ_f are generated using the MATLAB function `linspace`, with $f_0 = 1 \times 10^6$ Hz and $f_1 = 1 \times 10^{10}$ Hz for all Algorithms. The 1000 samples used for validating the ROM accuracy are generated using the MATLAB function `logspace`, with $f_l = 100$ Hz and $f_h = 1 \times 10^{10}$ Hz.

The results of the three algorithms are listed in Table 4. The standard greedy Algorithm 3 takes 50.7 min, resulting in a ROM of order $r = 132$ with validated error below the tolerance tol. During the greedy iteration, if the small parameter set Ξ_c is enriched only (add only), the greedy algorithm with bi-fidelity error estimation and that with multi-fidelity error estimation converge within the same number of iterations, producing ROMs with the same sizes and validated errors. But the greedy algorithm with multi-fidelity error estimation is 6 minutes faster. Similar phenomenon happens to the case "add-remove." The greedy algorithm with bi-fidelity error estimation and that with multi-fidelity error estimation also converge within the same number of iterations and construct ROMs with the same sizes and accuracy. The runtimes of both algorithms are less compared to their "add only" versions. Finally, the greedy algorithm with multi-fidelity error estimation by adding and deleting samples to and from Ξ_c ("add-remove") is most efficient in terms of both runtime and accuracy. It is more than two times faster than the standard greedy algorithm resulting in a speed-up of 2.5×, and produces a ROM with even the smallest validated error.

We note that using $|\Xi_c| = 10$, the ROMs constructed by the bi-fidelity greedy algorithm and the multi-fidelity greedy algorithm with adding the samples only have validated errors larger than the tolerance. If we increase $|\Xi_c|$ from 10 to 15, both algorithms generate ROMs with improved accuracy. The results are presented in Table 5. However, the computational time also increases a lot. Again, the multi-fidelity greedy algorithm outperforms the bi-fidelity one w.r.t. both accuracy and runtime. In contrast to the results in Tables 1 and 2 for the divider model, the results for the coplanar microstrips model in both Tables 4 and 5 show that the bi-fidelity greedy algorithm ("add-remove") is more accurate than its "add-only" version.

Table 6 shows the results of the bi-fidelity greedy algorithm and the multi-fidelity greedy algorithm based on adding/removing multiple samples at each iteration. For both cases, that is, $n_{\text{add}} = n_{\text{del}} = 2$ and $n_{\text{add}} = n_{\text{del}} = 5$, the algorithms using $|\Xi_c| = 10$, converge in 10 iterations, one less iteration than they did with $n_{\text{add}} = n_{\text{del}} = 1$ in Table 4, resulting in ROMs with smaller order r but with larger validated errors. If we increase $|\Xi_c|$ to 15, then the multi-fidelity greedy algorithm generates a ROM with reduced error, but takes longer time to converge. The bi-fidelity greedy algorithm behaves similarly and its results for $|\Xi_c| = 15$ is not presented to avoid repetition. This example again shows that adding/removing a single parameter at each iteration outperforms the cases with $n_{\text{add}} = n_{\text{del}} > 1$, and produces ROMs with desired accuracy.

In Figure 11, we show the *important* frequency samples of f selected by the greedy algorithms in Table 4. For the case "add-remove," we find that the greedy algorithm with bi-fidelity error estimation and the one with multi-fidelity error estimation select the same *important* frequency samples. Therefore we only plot one group of samples for both algorithms, see the plot "bi-(multi-) add-remove" in the figure. For the case "add-only," both algorithms also select the same *important* frequency samples, see the plot "bi-(multi-) add-only" in the figure. This is in agreement with the results given in Table 4 where both algorithms for either case produce the same results. It is seen that the *important* frequency samples selected by the (bi-)multi-fidelity estimation could be different from those selected by the high-fidelity estimator. However, both can derive ROMs with good accuracy.

The left part of Figure 12 gives the error-peak frequencies detected by the multi-fidelity error estimation and the true error, respectively, at each iteration of the multi-fidelity greedy Algorithm 4: those frequencies correspond to the largest values of the error estimator/true error. The error-peak frequency detected by the error estimator at the ith iteration is then selected as the *important* frequency sample at the next iteration to update the reduced basis space. From

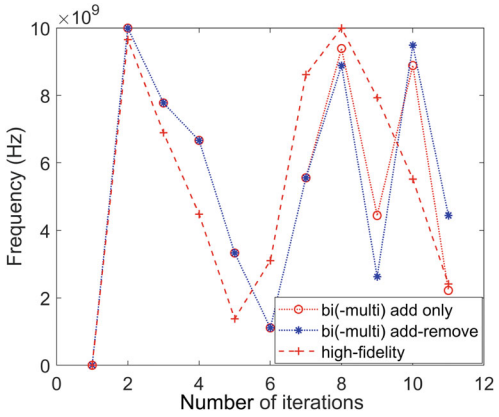

FIGURE 11 Important parameters selected by the greedy algorithms.

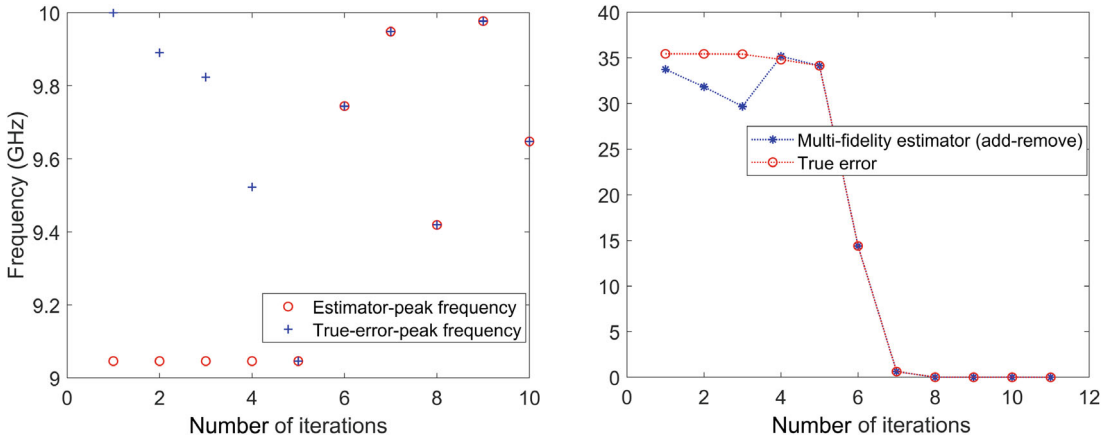

FIGURE 12 Left: Frequencies causing error/estimator peaks. Right: true error versus multi-fidelity error estimator.

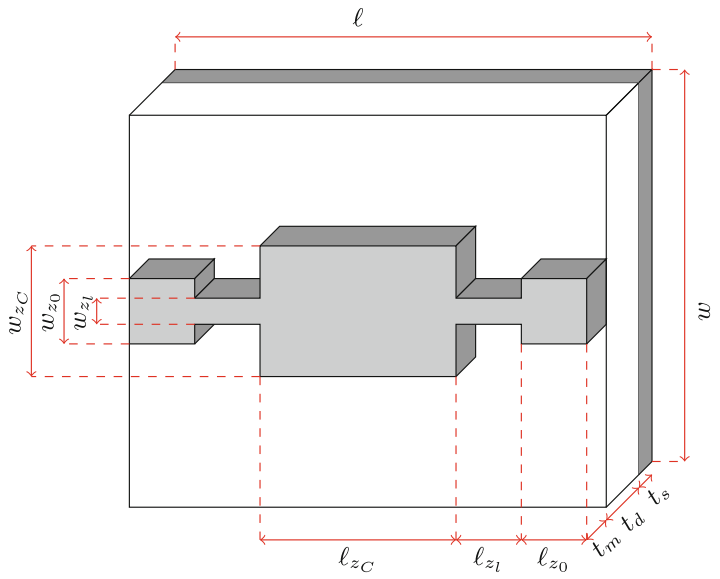

FIGURE 13 Microstrip filter.

iteration 5, the error-peak frequencies detected by the error estimator are exactly the same as those selected by the true error. This can be explained by the error decay in the right part of the figure. From the fifth iteration, the error estimator tightly catches the true error. Although it is less tight at the first 4 iterations, it still follows the overall trend of the error decay and therefore, can still detect reasonable error-peak frequencies. This example, once again, supports our theoretical analysis and demonstrates the efficacy of the proposed greedy algorithms with bi-(multi-) fidelity error estimation.

4.3 | Test 3: results for a model of microstrip filter

The third example is a model of a microstrip filter. The three-dimensional (3D) structure of a microstrip filter is depicted in Figure 13. The physical dimensions for the geometry of the 3D structure are: $w_{z_l} = 0.5$ mm, $w_{z_0} = 1.125$ mm, $w_{z_C} = 4$ mm, $\ell_{z_l} = 18.3$ mm, $\ell_{z_0} = 1$ mm, $\ell_{z_C} = 14.1$ mm, $w = 2.4$ cm, $\ell = 2\ell_{z_l} + 2\ell_{z_0} + \ell_{z_C}$, $t_m = 100$ μm, $t_s = 100$ μm, $t_d = 508$ μm. The two ends of the microstrip are terminated on 50 Ω resistors, leading to a system with two input ports and two output ports. The order of the FOM is $n = 50{,}904$, and there are $d = 192$ delays. The interesting frequency band is $[0, 10]$GHz.

Since results for first two examples have shown that the multi-fidelity error estimation by adding and removing a single sample outperforms the other proposed methods, we only compare the this multi-fidelity greedy process with the standard greedy algorithm. We use $|\Xi| = 30$ for the standard greedy Algorithm 3. For Algorithm 4, $|\Xi_c| = 15$, and $|\Xi_f| = 100$. The samples in Ξ, Ξ_c or Ξ_f are generated using the MATLAB the function `linspace`, with $f_0 = 1 \times 10^6$ Hz and $f_1 = 1 \times 10^{10}$ Hz for Algorithm 3 and $f_0 = 1 \times 10^8$ Hz and $f_1 = 1 \times 10^{10}$ Hz for Algorithm 4. The order of the FOM is much larger than the sizes of the first two examples leading to much slower FOM simulation at each frequency sample: around 8 min per FOM simulation. Therefore, we use 200 instead of 1000 samples to compute the validated error. The 200 samples are generated using the MATLAB function `logspace`, with $f_l = 100$ Hz and $f_h = 1 \times 10^{10}$ Hz. The corresponding results are listed in Table 7, which show that using multi-fidelity error estimation has saved at least 1.9 h of offline computation time. This indicates that for larger systems, the multi-fidelity error estimation could save more offline time. If we use a even smaller Ξ_c with $|\Xi_c| = 10$, a ROM with validated error 0.0024 is obtained. Over the 200 validated frequency samples, the error of the ROM is larger than the tolerance at 14 samples, with 7% failure. The runtime is reduced from 4.9 to 2.2 h as compared to the standard greedy algorithm with high-fidelity error estimation. In Figures 14 and 15, we plot the transfer functions (17) of the original system corresponding to different input–output ports and the transfer functions (19) of the ROM. Due to the symmetry of the device, the transfer function corresponding to the input port 2 and output port 2 is the same as the one corresponding to the input port 1 and output port 1. Similarly, the transfer function corresponding to the input port 1 and output port 2 is the same as the one corresponding to the input port 2 and output port 1. Consequently, we only plot two transfer functions. From those figures, it is evident that the transfer function $\hat{H}(s)$ reproduces the original $H(s)$. The ROMs of the other two examples also have accurate transfer functions. To avoid repetition, we only show these results for the third example.

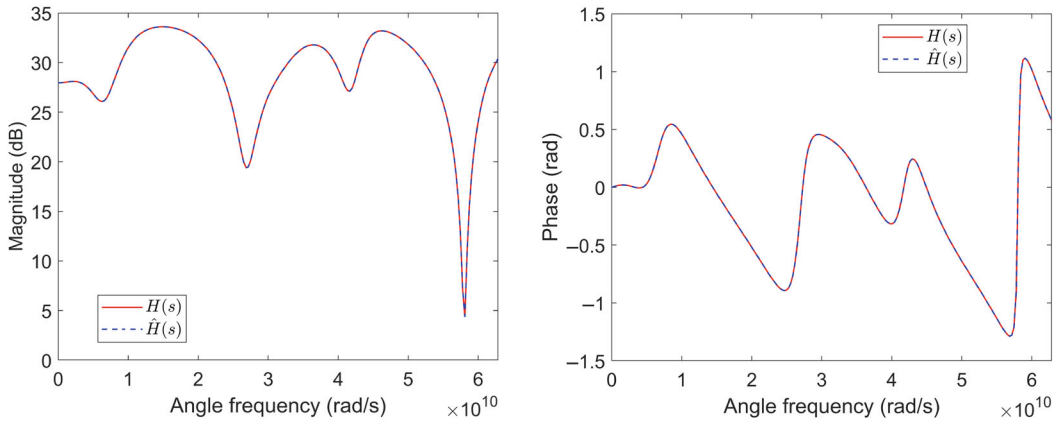

FIGURE 14 Comparison of the transfer functions from input port 1 to output port 1. Left: Magnitudes. Right: Phases.

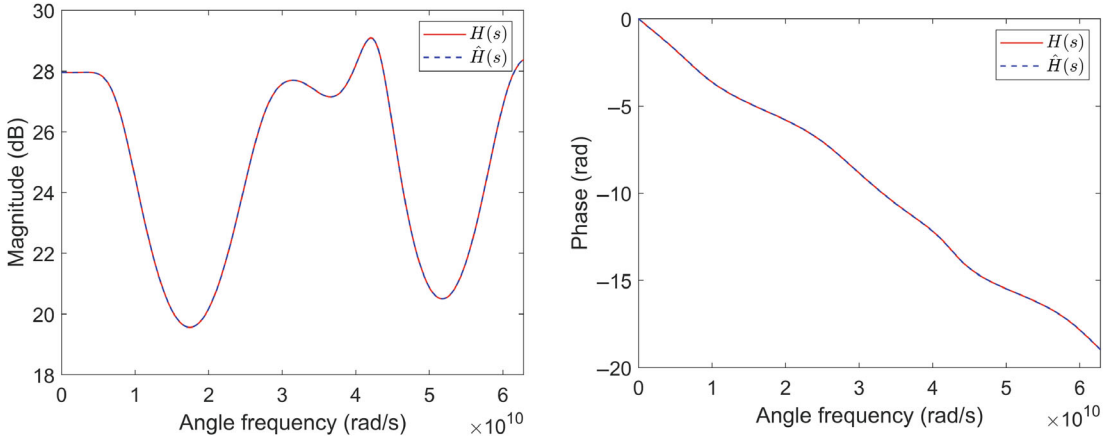

FIGURE 15 Comparison of the transfer functions from input port 1 to output port 2. Left: Magnitudes. Right: Phases.

5 | CONCLUSIONS

Concepts of bi-fidelity error estimation and multi-fidelity error estimation are proposed in this work. The concept of bi-fidelity error estimation is general and can be applied to any high-fidelity estimator. Although the multi-fidelity error estimation is dependent on the high-fidelity error estimation in consideration, the framework is general to a certain extend and could also be combined with other high-fidelity error estimators. The robustness of the proposed greedy algorithms with bi-fidelity and multi-fidelity error estimation is tested on three large time-delay systems with many delays. Although the standard greedy algorithm converges in a few iterations, the computational complexity at each iteration is high. As a consequence, the runtime is long for such systems. The proposed (bi-)multi-fidelity greedy processes have significantly accelerated the standard greedy algorithm with little loss of accuracy in many cases.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in zenodo at https://doi.org/10.5281/zenodo.7892188.

ACKNOWLEDGMENT

Open Access funding enabled and organized by Projekt DEAL.

ORCID

Lihong Feng https://orcid.org/0000-0002-1885-3269
Peter Benner https://orcid.org/0000-0003-3362-4103

REFERENCES

1. Antoulas AC. *Approximation of Large-Scale Dynamical Systems. Advances in Design and Control*. Vol 6. SIAM Publications; 2005.
2. Baur U, Benner P, Feng L. Model order reduction for linear and nonlinear systems: a system-theoretic perspective. *Arch Comput Methods Eng*. 2014;21(4):331-358.
3. Benner P, Gugercin S, Willcox K. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Rev*. 2015;57(4):483-531.
4. Antoulas AC, Beattie CA, Gugercin S. *Interpolatory Methods for Model Reduction. Computational Science & Engineering*. Society for Industrial and Applied Mathematics; 2020.
5. Benner P, Cohen A, Ohlberger M, Willcox K, eds. *Model Reduction and Approximation: Theory and Algorithms. Computational Science & Engineering*. SIAM Publications; 2017.
6. Benner P, Grivet-Talocia S, Quarteroni A, Rozza G, Schilders W, Silveira LM. In: Gruyter D, ed. *Model Order Reduction. System- and Data-Driven Methods and Algorithms*. Vol 1. De Gruyter; 2021.
7. Benner P, Grivet-Talocia S, Quarteroni A, Rozza G, Schilders W, Silveira LM. *Model Order Reduction*. Vol 2. *Snapshot-Based Methods and Algorithms*. De Gruyter; 2021.
8. Benner P, Grivet-Talocia S, Quarteroni A, Rozza G, Schilders W, Silveira LM. *Model Order Reduction. Applications*. Vol 3. De Gruyter; 2021.
9. Quarteroni A, Manzoni A, Negri F. *Reduced Basis Methods for Partial Differential Equations, Volume 92. La Matematica per Il 3+2*. Springer International Publishing; 2016.
10. Binev P, Cohen A, Dahmen W, DeVore R, Petrova G, Wojtaszczyk P. Convergence rates for greedy algorithms in reduced basis methods. *SIAM J Math Anal*. 2011;43(3):1457-1472.
11. Buffa A, Maday Y, Patera AT, Prudhomme C, Turinici G. A priori convergence of the greedy algorithm for the parametrized reduced basis. *ESAIM Math Model Num Anal*. 2012;46:595-603.
12. Haasdonk B, Ohlberger M. Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM Math Model Num Anal*. 2008;42(2):277-302.
13. Rozza G, Huynh DBP, Patera AT. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Arch Comput Method Eng*. 2008;15(3):229-275.
14. Veroy K, Prud'Homme C, Rovas DV, Patera AT. A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations. Paper presented at: 16th AIAA Computational Fluid Dynamics Conference. 2003; Orlando, FL.
15. Vore RD, Petrova G, Wojtaszczyk P. Greedy algorithms for reduced bases in banach spaces. *Constr Approx*. 2013;37:455-466.
16. Alfke D, Feng L, Lombardi L, Antonini G, Benner P. Model order reduction for delay systems by iterative interpolation. *Int J Numer Methods Eng*. 2021;122(3):684-706.
17. Grepl M. *Reduced-Basis Approximation a Posteriori Error Estimation for Parabolic Partial Differential* Equations. PhD Thesis. Massachussetts Institute of Technology (MIT). 2005.
18. Veroy K, Patera AT. Certified real-time solution of the parametrized steady incompressible Navier-stokes equations: rigorous reduced-basis a posteriori error bounds. *Int J Num Method Fluid*. 2005;47(8-9):773-788.

19. Feng L, Antoulas AC, Benner P. Some a posteriori error bounds for reduced order modelling of (non-)parametrized linear systems. *ESAIM Math Model Num Anal*. 2017;51(6):2127-2158.
20. Feng L, Benner P. On error estimation for reduced-order modeling of linear non-parametric and parametric systems. *ESAIM Math Model Num Anal*. 2021;55(2):561-594.
21. Grepl MA. Certiﬁed reduced basis methods for nonaﬃne linear time-varying and nonlinear parabolic partial differential equations. *Math Models Methods Appl Sci*. 2012;22(3):1150015.
22. Grepl MA, Maday Y, Nguyen NC, Patera AT. Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM Math Model Numer Anal*. 2007;41(3):575-605.
23. Grepl MA, Patera AT. A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. *Math Model Num Anal*. 2005;39:157-181.
24. Grunert D, Fehr J, Haasdonk B. Well-scaled, a-posteriori error estimation for model order reduction of large second-order mechanical systems. *Z Angew Math Mech*. 2019;100(8):1-43.
25. Haasdonk B, Ohlberger M. Efficient reduced models and a-posteriori error estimation for parametrized dynamical systems by offline/online decomposition. *Math Comput Model Dyn Syst*. 2011;17(2):145-161.
26. Hain S, Ohlberger M, Radic M, Urban K. A hierarchical a-posteriori error estimator for the reduced basis method. *Adv Comput Math*. 2019;45(2):2191-2221.
27. Rovas DV. *Reduced-Basis Output Bound Methods for Parametrized Partial Differential Equations*. PhD Thesis. Massachusetts Institute of Technology (MIT). 2003.
28. Schmidt A, Wittwar B, Haasdonk D. Rigorous and effective a-posteriori error bounds for nonlinear problems—application to RB methods. *Adv Comput Math*. 2020;46(32):30.
29. Smetana K, Zahm O, Patera AT. Randomized residual-based error estimators for parametrized equations. *SIAM J Sci Comput*. 2019;41(2):A900-A926.
30. Chellappa S, Feng L, Benner P. An adaptive sampling approach for the reduced basis method. *Realization and Model Reduction of Dynamical Systems - A Festschrift in Honor of the 70th Birthday of Thanos Antoulas*. Springer; 2022:137-155.
31. Zhang Y, Feng L, Li S, Benner P. An efficient output error estimation for model order reduction of parametrized evolution equations. *SIAM J Sci Comput*. 2015;37(6):B910-B936.
32. Benaceur A, Ehrlacher V, Ern A, Meunier S. Simultaneous empirical interpolation and reduced basis method for non-linear problems. *C R Acad Sci Paris*. 2015;353(12):1105-1109.
33. Benaceur A, Ehrlacher V, Ern A, Meunier S. A progressive reduced basis/empirical interpolation method for nonlinear parabolic problems. *SIAM J Sci Comput*. 2018;40(5):A2930-A2955.
34. Paul-Dubois-Taine A, Amsallem D. An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models. *Int J Numer Methods Eng*. 2015;102(12):1262-1292.
35. Chellappa S, Feng L, de la Rubia V, Benner P. Adaptive interpolatory MOR by learning the error estimator in the parameter domain. *Model Reduction of Complex Dynamical Systems. International Series of Numerical Mathematics*. Vol 171. Birkhäuser; 2021:97-117.
36. Beattie CA, Gugercin S. Interpolatory projection methods for structure-preserving model reduction. *Syst Control Lett*. 2009;58(3):225-232.
37. Ruehli AE. Inductance calculations in a complex integrated circuit environment. *IBM J Res Dev*. 1972;16(5):470-481.
38. Ruehli AE, Brennan PA. Efficient capacitance calculations for three-dimensional multiconductor systems. *IEEE Trans Microwave Theory Tech*. 1973;21(2):76-82.
39. Ruehli AE. Equivalent circuit models for three dimensional multiconductor systems. *IEEE Trans Microwave Theory Tech*. 1974;22(3):216-221.
40. Ruehli AE, Antonini G, Jiang L. *Circuit Oriented Electromagnetic Modeling Using the PEEC Techniques*. Wiley-IEEE Press; 2017.
41. Gianfagna C, Lombardi L, Antonini G. Marching-on-in-time solution of delayed PEEC models of conductive and dielectric objects. *IET Microwaves Antenna Propagat*. 2019;13(1):42-47.

How to cite this article: Feng L, Lombardi L, Antonini G, Benner P. Multi-fidelity error estimation accelerates greedy model reduction of complex dynamical systems. *Int J Numer Methods Eng*. 2023;124(23):5312-5333. doi: 10.1002/nme.7348