Adaptive Interpolatory MOR by Learning the Error Estimator in the Parameter Domain

Sridhar Chellappa, Lihong Feng, Valentín de la Rubia and Peter Benner

Abstract Interpolatory methods offer a powerful framework for generating reduced-order models (ROMs) for non-parametric or parametric systems with time-varying inputs. Choosing the interpolation points adaptively remains an area of active interest. A greedy framework has been introduced in [11, 13] to choose interpolation points automatically using *a posteriori* error estimators. Nevertheless, when the parameter range is large or if the parameter space dimension is larger than two, the greedy algorithm may take considerable time, since the training set needs to include a considerable number of parameters. As a remedy, we introduce an adaptive training technique by learning an efficient *a posteriori* error estimator over the parameter domain. A fast learning process is created by interpolating the error estimator using radial basis functions (RBF) over a fine parameter training set, representing the whole parameter domain. The error estimator is evaluated only on a coarse training set including a few parameter samples. The algorithm is an extension of the work in [9] to interpolatory model order reduction (MOR) in frequency domain. Beyond the work in [9], we use a newly proposed inf-sup-constant-free error estimator in the frequency domain [13], which is often much tighter than the error estimator using the inf-sup constant. Three numerical examples demonstrate the efficiency and validity of the proposed approach.

Sridhar Chellappa
Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, Magdeburg, 39106, Germany e-mail: `chellappa@mpi-magdeburg.mpg.de`

Lihong Feng
Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, Magdeburg, 39106, Germany e-mail: `feng@mpi-magdeburg.mpg.de`

Valentín de la Rubia
Departamento de Matemática Aplicada a las TIC, ETSI de Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain e-mail: `valentin.delarubia@upm.es`

Peter Benner
Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, Magdeburg, 39106, Germany e-mail: `benner@mpi-magdeburg.mpg.de`

1 Introduction

MOR based on system theory and interpolation [3, 11, 12, 15, 16] has been developed as a class of efficient MOR methods among others. Detailed summary of those methods and comparison of them with other classes of MOR methods can be found in some survey papers and books [1, 2, 4, 7, 8, 17, 29]. A major advantage of the interpolatory methods is their flexibility in reducing systems with time or parameter varying inputs, since they are based on the transfer function or the input-output relation of the systems, which is independent of the input signal. On the contrary, the snapshot MOR methods, such as proper orthogonal decomposition (POD) and the reduced basis methods (RBM) are input-dependent, and are often less efficient in reducing systems with varying inputs as compared with the interpolatory MOR methods [7].

A major topic of interest in interpolatory MOR methods is how to determine the interpolation points, so as to adaptively construct the ROM. Many methods have appeared in the last ten years, some are heuristic [5, 14, 20, 22], some entail high computational complexity [11, 32], and some are inefficient for systems with more than one parameter [16, 21]. Random interpolation points are used in [6].

Recently, a new error estimator for the reduced transfer function error and an algorithm for iteratively choosing the interpolation points are proposed in [13], which overcomes many difficulties being faced by the above mentioned interpolatory methods. It is neither heuristic nor needs a high computational cost. Moreover, it is a parametric MOR method and applicable to systems with more than two parameters. One shortcoming of the method is that the interpolation points are selected from a given training set, which must be decided a priori and becomes larger and larger with the increase of the parameter range or the parameter space dimension. Such a technique is standard also for the RBM, where a training set must be given before a greedy algorithm starts. This makes the greedy algorithm slow down when there is a large number of samples in the training set due to the large dimension or large range of the parameter domain. This is due to the fact that at each iteration of the greedy algorithm, an error estimator needs to be repeatedly computed for all the samples in the training set. Many adaptive training techniques have been proposed recently for RBM [9, 18, 19, 23]. In contrast, no efficient training techniques are proposed for the interpolatory MOR methods, though similar greedy algorithms using fixed training sets are proposed in [11, 13]. In this work, we extend the adaptive training technique in [9] for RBM to an adaptive training technique for the interpolatory MOR methods in [11, 13].

The main contribution of this work is an efficient algorithm to adaptively choose interpolation points for parametric, linear time-invariant (LTI) systems having a wide range of parameter values or with a large parameter space dimension. Compared with the greedy algorithms proposed in [11, 13], we have added two new ingredients to the greedy algorithms: (i) a sharp *a posteriori* output (or state) error estimator with low computational costs and (ii) a surrogate with even lower computational costs for learning the error estimator over the whole parameter domain. The aim is instead of computing the error estimator over the whole parameter domain the for ROM construction, a surrogate estimator is computed. In this way, the error estimator is computed only on a coarse training set at each iteration of the greedy algorithm, and for parameters outside of the coarse training set, the surrogate estimator is computed. Finally, the training set needs to be initialized by including only a few parameter samples, and can be iteratively updated using the surrogate estimator instead of the error estimator itself. As a consequence, a significant amount of computational cost can be saved for such systems.

The idea is similar to the one in [9] for the RBM. However, in [9], an error estimator in time domain is used, where the inf-sup constant needs to be computed for each parameter in the training set, which is computationally inefficient for large-scale systems. In this work, we use an inf-sup-constant-free error estimator newly proposed in [13]. It is suitable for interpolatory MOR methods, since it estimates the transfer function error in the frequency domain.

The paper is organized as follows. In Section 2 we briefly review interpolatory MOR methods based on projection. The greedy interpolatory methods [11, 13] for parametric systems are reviewed in Section 3. In Section 4, we introduce the basic idea of RBF interpolation and elaborate on the process of learning the error estimator using a surrogate estimator constructed by RBF interpolation. Based on this surrogate estimator, we propose the greedy algorithm IPSUE with adaptive training technique for adaptively choosing the interpolation points in a more efficient and fully adaptive way. We present numerical results on three real-word examples in Section 4 to show the robustness of IPSUE and conclude the work in the end.

2 Interpolatory MOR

In this work, we are interested in MOR of parametric LTI systems in the *state-space representation* given by

$$\Sigma(\boldsymbol{\mu}) : \begin{cases} \mathbf{E}\dot{\mathbf{x}}(t,\boldsymbol{\mu}) = \mathbf{A}(\boldsymbol{\mu})\mathbf{x}(t,\boldsymbol{\mu}) + \mathbf{B}(\boldsymbol{\mu})\mathbf{u}(t), \\ \ \ \mathbf{y}(t,\boldsymbol{\mu}) = \mathbf{C}(\boldsymbol{\mu})\mathbf{x}(t,\boldsymbol{\mu}),\ \ \mathbf{x}(0,\boldsymbol{\mu}) = 0. \end{cases} \tag{1}$$

Here, $\boldsymbol{\mu} := \left[\mu^1, \mu^2, \dots, \mu^d\right]^\mathsf{T} \in \mathbb{R}^d$ is the vector of parameters (geometric or physical). $\mathbf{x}(t,\boldsymbol{\mu}) \in \mathbb{R}^n$ is the state vector and n is typically very large. $\mathbf{u}(t) \in \mathbb{R}^m$ is the input vector and $\mathbf{y}(t,\boldsymbol{\mu}) \in \mathbb{R}^p$ is the output vector. $\mathbf{A}(\boldsymbol{\mu}) \in \mathbb{R}^{n\times n}$ is the state matrix, $\mathbf{B}(\boldsymbol{\mu}) \in \mathbb{R}^{n\times m}$ is the input matrix, $\mathbf{C}(\boldsymbol{\mu}) \in \mathbb{R}^{p\times n}$ is the output matrix. For the case when $m = p = 1$, Eq. (1) is referred to as a single-input, single-output (SISO) system. Otherwise, it is known as multi-input, multi-output (MIMO) system.

The ROM we seek should preserve the same structure of the FOM but have a much smaller dimension. We assume that the state vector lies (approximately) in the span of a low-dimensional linear subspace $\mathcal{V} \subset \mathbb{R}^{n\times r}$, $r \ll n$, such that, $\mathbf{x}(t,\boldsymbol{\mu}) \approx \mathbf{V}\hat{\mathbf{x}}(t,\boldsymbol{\mu})$. Column vectors in the matrix $\mathbf{V} \in \mathbb{R}^{n\times r}$ constitute an orthogonal basis of $\mathcal{V}$. Replacing $\mathbf{x}(t,\boldsymbol{\mu})$ in Eq. (1) with its approximation $\mathbf{V}\hat{\mathbf{x}}(t,\boldsymbol{\mu})$ and further imposing Petrov-Galerkin projection on the residual introduced by the approximation in a test subspace $\mathcal{W} \subset \mathbb{R}^{n\times r}$ leads to

$$\mathbf{W}^{\mathsf{T}}\Big(\mathbf{V}\dot{\hat{\mathbf{x}}}(t,\boldsymbol{\mu}) - \mathbf{A}(\boldsymbol{\mu})\mathbf{V}\hat{\mathbf{x}}(t,\boldsymbol{\mu}) - \mathbf{B}(\boldsymbol{\mu})\mathbf{u}(t)\Big) \equiv \mathbf{0},$$

where, the column vectors in the matrix $\mathbf{W} \in \mathbb{R}^{n\times r}$ correspond to an orthogonal basis of $\mathcal{W}$. The resulting ROM is given as

$$\hat{\Sigma}(\boldsymbol{\mu}) : \begin{cases} \hat{\mathbf{E}}\dot{\hat{\mathbf{x}}}(t,\boldsymbol{\mu}) = \hat{\mathbf{A}}(\boldsymbol{\mu})\hat{\mathbf{x}}(t,\boldsymbol{\mu}) + \hat{\mathbf{B}}(\boldsymbol{\mu})\mathbf{u}(t), \\ \quad \hat{\mathbf{y}}(t,\boldsymbol{\mu}) = \hat{\mathbf{C}}(\boldsymbol{\mu})\hat{\mathbf{x}}(t,\boldsymbol{\mu}), \;\; \hat{\mathbf{x}}(0,\boldsymbol{\mu}) = 0. \end{cases} \tag{2}$$

Here, $\hat{\mathbf{x}}(t,\boldsymbol{\mu}) \in \mathbb{R}^r$ is the reduced state vector, $\hat{\mathbf{E}}(\boldsymbol{\mu}) = \mathbf{W}^{\mathsf{T}}\mathbf{E}(\boldsymbol{\mu})\mathbf{V} \in \mathbb{R}^{r\times r}$, $\hat{\mathbf{A}}(\boldsymbol{\mu}) = \mathbf{W}^{\mathsf{T}}\mathbf{A}(\boldsymbol{\mu})\mathbf{V} \in \mathbb{R}^{r\times r}$, $\hat{\mathbf{B}}(\boldsymbol{\mu}) = \mathbf{W}^{\mathsf{T}}\mathbf{B}(\boldsymbol{\mu}) \in \mathbb{R}^{r\times m}$, $\hat{\mathbf{C}}(\boldsymbol{\mu}) = \mathbf{C}(\boldsymbol{\mu})\mathbf{V} \in \mathbb{R}^{p\times r}$ are the reduced system matrices, and $\hat{\mathbf{y}}(t,\boldsymbol{\mu})$ is the reduced output vector. The goal of MOR is to find the two subspaces $\mathcal{V},\mathcal{W} \in \mathbb{R}^{n\times r}$. Different MOR methods vary in how they generate the matrices $\mathbf{W},\mathbf{V}$.

Interpolatory MOR methods construct $\mathbf{V},\mathbf{W} \in \mathbb{R}^{n\times r}$ based on the transfer function of the system, which is independent of the input signal. The transfer function of the system described in Eq. (1) is given by

$$\mathbf{H}(\tilde{\boldsymbol{\mu}}) := \mathbf{C}(\boldsymbol{\mu})\big(\overbrace{s\mathbf{E} - \mathbf{A}(\boldsymbol{\mu})}^{=:\mathscr{A}(\tilde{\boldsymbol{\mu}})}\big)^{-1}\mathbf{B}(\boldsymbol{\mu}). \tag{3}$$

Here, $\tilde{\boldsymbol{\mu}} := \left[s, \mu^1, \mu^2, \ldots, \mu^d\right]^{\mathsf{T}} \in \mathbb{R}^{d+1}$ is the vector of parameters with the additional Laplace variable $s \in \jmath\mathbb{R}$, where $\jmath$ is the imaginary unit. The corresponding ROM of Eq. (3) is of the form,

$$\hat{\mathbf{H}}(\tilde{\boldsymbol{\mu}}) := \hat{\mathbf{C}}(\tilde{\boldsymbol{\mu}})\hat{\mathscr{A}}(\tilde{\boldsymbol{\mu}})^{-1}\hat{\mathbf{B}}(\boldsymbol{\mu}), \tag{4}$$

with $\hat{\mathscr{A}}(\tilde{\boldsymbol{\mu}}) := s\hat{\mathbf{E}} - \hat{\mathbf{A}}(\boldsymbol{\mu})$.

Many interpolatory methods have been proposed for linear systems, especially for linear non-parametric systems. The most representative methods are the moment-matching methods [15, 16], where the $\mathcal{H}_2$-optimal method IRKA [16] constructs a ROM satisfying the necessary conditions of local optimality. All these methods are known to be applicable to non-parametric systems. Later, IRKA is extended to MOR for parametric systems [3], where some pairs of projection matrices are constructed for given samples of parameters, then they are combined together to get the final pair of projection matrices. No rule is used for selecting the samples. In [21], a method for parametric systems is proposed based on $\mathcal{H}_2 \otimes \mathcal{L}_2$-optimality, but is only applicable to systems with one parameter and is facing high computational complexity for systems with $n \geq 1000$.

Choosing interpolation points using a greedy algorithm guided by an *a posteriori* error bound is proposed in [11]. However, computing the error estimator needs to compute the smallest singular values of the large matrix $\mathscr{A}(\tilde{\boldsymbol{\mu}})$, the inf-sup constant. An inf-sup-constant-free error estimator is newly proposed in [13], which can be efficiently computed, and is also much tighter than the error bound [11] for many systems with small inf-sup constants. A similar greedy algorithm is proposed in [13] for choosing the interpolation points using the new error estimator. The adaptive training approach proposed in this work is based on the greedy algorithm and the new error estimator in [13]. In the next section, we briefly review the error estimator and the corresponding greedy algorithm.

3 Greedy Method for Choosing Interpolation Points

The transfer function can be seen as a mapping from the space of inputs $\mathbb{R}^m$ to the space of outputs $\mathbb{R}^p$ passing through a high dimensional intermediate state in $\mathbb{R}^n$. If we look at the matrix product $\mathscr{A}^{-1}(\tilde{\boldsymbol{\mu}})\mathbf{B}(\boldsymbol{\mu})$ in $\mathbf{H}(\boldsymbol{\mu})$, we may consider the primal system,

$$\mathscr{A}(\tilde{\boldsymbol{\mu}})\mathbf{X}_{\text{pr}}(\tilde{\boldsymbol{\mu}}) = \mathbf{B}(\boldsymbol{\mu}). \tag{5}$$

Here, $\mathbf{X}_{\text{pr}}(\tilde{\boldsymbol{\mu}}) \in \mathbb{R}^n$ is the primal state vector. The reduced primal system is defined as,

$$\hat{\mathscr{A}}(\tilde{\boldsymbol{\mu}})\hat{\mathbf{X}}_{\text{pr}}(\tilde{\boldsymbol{\mu}}) = \hat{\mathbf{B}}(\boldsymbol{\mu}). \tag{6}$$

The approximate primal solution is given by $\tilde{\mathbf{X}}_{\text{pr}}(\tilde{\boldsymbol{\mu}}) := \mathbf{V}\hat{\mathbf{X}}_{\text{pr}}(\tilde{\boldsymbol{\mu}})$ and the corresponding residual is

$$\mathbf{r}_{\text{pr}}(\boldsymbol{\mu}) = \mathbf{B}(\boldsymbol{\mu}) - \mathscr{A}(\tilde{\boldsymbol{\mu}})\tilde{\mathbf{X}}_{\text{pr}}(\tilde{\boldsymbol{\mu}}). \tag{7}$$

Additionally, by considering the matrix product $\mathbf{C}(\boldsymbol{\mu})\mathscr{A}^{-1}(\tilde{\boldsymbol{\mu}})$ in $\mathbf{H}(\boldsymbol{\mu})$, we have the following dual system,

$$\mathscr{A}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\mathbf{X}_{\text{du}}(\tilde{\boldsymbol{\mu}}) = \mathbf{C}^{\mathsf{T}}(\boldsymbol{\mu}). \tag{8}$$

$\mathbf{X}_{\text{du}}(\tilde{\boldsymbol{\mu}}) \in \mathbb{R}^n$ is the dual state vector. The reduced dual system is given as,

$$\hat{\mathscr{A}}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\hat{\mathbf{X}}_{\text{du}}(\tilde{\boldsymbol{\mu}}) = \hat{\mathbf{C}}^{\mathsf{T}}(\boldsymbol{\mu}). \tag{9}$$

The approximate dual solution is given by $\tilde{\mathbf{X}}_{\text{du}}(\tilde{\boldsymbol{\mu}}) := \mathbf{V}_{\text{du}}\hat{\mathbf{X}}_{\text{du}}(\tilde{\boldsymbol{\mu}})$ and the corresponding residual is,

$$\mathbf{r}_{\text{du}}(\boldsymbol{\mu}) = \mathbf{C}^{\mathsf{T}}(\boldsymbol{\mu}) - \mathscr{A}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\tilde{\mathbf{X}}_{\text{du}}(\tilde{\boldsymbol{\mu}}). \tag{10}$$

For parametric LTI systems, adopting the spirit of the RBM, [11] introduced a method to automatically generate a ROM through a greedy algorithm. The authors introduce a primal-dual residual-based *a posteriori* error estimator for the transfer function approximation error $\|\mathbf{H}(\tilde{\boldsymbol{\mu}}) - \hat{\mathbf{H}}(\tilde{\boldsymbol{\mu}})\|$, for both SISO and MIMO systems. For SISO systems it reads,

$$|\mathbf{H}(\tilde{\boldsymbol{\mu}}) - \hat{\mathbf{H}}(\tilde{\boldsymbol{\mu}})| \leq \frac{\|\mathbf{r}_{\text{pr}}(\tilde{\boldsymbol{\mu}})\|_2 \|\mathbf{r}_{\text{du}}(\tilde{\boldsymbol{\mu}})\|_2}{\sigma_{\min}(\boldsymbol{\mu})}. \tag{11}$$

Here, $\sigma_{\min}(\boldsymbol{\mu})$, called the inf-sup constant, is the smallest singular value of the matrix $\mathscr{A}(\boldsymbol{\mu})$ as defined in Eq. (3). The primal and dual residuals are given in Eq. (7) and Eq. (10), respectively. The work [13] improves the method in [11] by avoiding the calculation of the inf-sup constant required for the error estimator. This is achieved by introducing a dual-residual system,

$$\mathscr{A}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\mathbf{e}_{\text{du}}(\tilde{\boldsymbol{\mu}}) = \mathbf{r}_{\text{du}}(\tilde{\boldsymbol{\mu}}). \tag{12}$$

The following theorem from [13] gives the *a posteriori* error bound,

Proposition 1 *The transfer function approximation error can be bounded as,*

$$|\mathbf{H}(\tilde{\boldsymbol{\mu}}) - \hat{\mathbf{H}}(\tilde{\boldsymbol{\mu}})| \leq \left|\tilde{\mathbf{X}}_{du}^{T}(\tilde{\boldsymbol{\mu}})\mathbf{r}_{pr}(\tilde{\boldsymbol{\mu}})\right| + \left|\mathbf{e}_{du}^{T}(\tilde{\boldsymbol{\mu}})\mathbf{r}_{pr}(\tilde{\boldsymbol{\mu}})\right|.$$

For a proof of Proposition 1, we refer to [13]. In this form, the error bound is not computationally efficient since one needs to solve the full order dual-residual system (12) to obtain $\mathbf{e}_{\text{du}}(\tilde{\boldsymbol{\mu}})$. Instead, system (12) is reduced by an orthogonal matrix $\mathbf{V}_e \in \mathbb{R}^{n\times\ell}$ as below,

$$\hat{\mathscr{A}}_e^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\hat{\mathbf{e}}_{\text{du}}(\tilde{\boldsymbol{\mu}}) = \hat{\mathbf{r}}_{\text{du,e}}(\tilde{\boldsymbol{\mu}}), \tag{13}$$

where $\hat{\mathscr{A}}_e := \mathbf{V}_e^{\mathsf{T}}\,\mathscr{A}(\tilde{\boldsymbol{\mu}})\,\mathbf{V}_e$ and $\hat{\mathbf{r}}_{\text{du,e}} := \mathbf{V}_e^{\mathsf{T}}\,\mathbf{r}_{\text{du}}(\tilde{\boldsymbol{\mu}})$. The projection matrices $\mathbf{V}, \mathbf{V}_{\text{du}}$ and $\mathbf{V}_e$ corresponding to the primal, dual and the dual-residual system are generated offline.

By using the approximate solution to the dual-residual system, an efficiently computable error estimator is obtained.

$$|\mathbf{H}(\tilde{\boldsymbol{\mu}}) - \hat{\mathbf{H}}(\tilde{\boldsymbol{\mu}})| \lesssim \left|\tilde{\mathbf{X}}_{\text{du}}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\mathbf{r}_{\text{pr}}(\tilde{\boldsymbol{\mu}})\right| + \left|\tilde{\mathbf{e}}_{\text{du}}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\mathbf{r}_{\text{pr}}(\tilde{\boldsymbol{\mu}})\right| =: \Delta(\tilde{\boldsymbol{\mu}}), \tag{14}$$

where $\tilde{\mathbf{e}}_{\text{du}}(\tilde{\boldsymbol{\mu}}) := \mathbf{V}_e\,\hat{\mathbf{e}}_{\text{du}}(\tilde{\boldsymbol{\mu}})$. For ease of comparison, we first present the greedy algorithm for parametric systems introduced in [13] as Algorithm 1.

It is automatic apart from the need for determining, *a priori*, a suitable training set Ξ. The method proceeds by picking points from Ξ that maximize the error estimator at every iteration and updating the three projection matrices $\mathbf{V}, \mathbf{V}_{\text{du}}, \mathbf{V}_e$. However, there is no principled way to select the training set *a priori*. If not adequately sampled, the training set may result in a ROM whose error is not uniformly below the tolerance. When the parameters involved can take on a wide range of values, or if many parameters are involved, then the number of parameter samples in Ξ becomes large and the offline computation costs rise. We propose to solve this issue by constructing a surrogate model for $\Delta(\tilde{\boldsymbol{\mu}})$ in Eq. (14) and assure that computing the surrogate is much cheaper than computing the error estimator itself. The next section discusses this idea.

Remark 1 The algorithm is also applicable to MIMO systems. In this case the transfer function is matrix-valued. The key is how to compute the error estimator $\Delta(\tilde{\boldsymbol{\mu}})$. We first estimate the error of the reduced transfer function entry-wise, i.e.

$$|\mathbf{H}_{ij}(\tilde{\boldsymbol{\mu}}) - \hat{\mathbf{H}}_{ij}(\tilde{\boldsymbol{\mu}})| \lesssim \left|\tilde{\mathbf{X}}_{\text{du}}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\mathbf{r}_{\text{pr}}(\tilde{\boldsymbol{\mu}})\right| + \left|\tilde{\mathbf{e}}_{\text{du}}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\mathbf{r}_{\text{pr}}(\tilde{\boldsymbol{\mu}})\right| =: \Delta_{ij}(\tilde{\boldsymbol{\mu}}). \tag{15}$$

Note that the ij-th entry of the transfer function corresponds to the input signal at the j-th input port and the signal at the i-th output port. Then, $\tilde{\mathbf{X}}_{\text{du}}(\tilde{\boldsymbol{\mu}})$ is the

Algorithm 1 Greedy ROM Construction for Parametric Systems [13]

```
Input: System matrices $\mathbf{A}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$, $\mathbf{C}(\boldsymbol{\mu})$ , Training set $\Xi$ of cardinality $N_\mu$ covering the interesting parameter ranges, Tolerance $\epsilon_{tol}$.
Output: Projection matrix $\mathbf{V}$.
 1: Initialize $\mathbf{V} = [\,]$, $\mathbf{V}_{\text{du}} = [\,]$, $\mathbf{V}_e = [\,]$, $\epsilon = 1 + \epsilon_{tol}$, fix $\eta$, the number of moments to be matched.
 2: Initial interpolation point $\tilde{\boldsymbol{\mu}}^1$: the first sample in $\Xi$. $\tilde{\boldsymbol{\mu}}^1_\alpha$: the last sample in $\Xi$. Set $i = 1$.
 3: while $\epsilon > \epsilon_{tol}$ do
 4:     Solve Eq. (5) at interpolation point $\tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{\mu}}^i$ and update projection matrix
            $\mathbf{V} = \mathsf{orth}\big([\mathbf{V}\ \mathtt{mmm}(\mathscr{A}(\tilde{\boldsymbol{\mu}}^i), \mathbf{B}(\tilde{\boldsymbol{\mu}}^i), \eta, \tilde{\boldsymbol{\mu}}^i)]\big).$
 5:     Solve Eq. (8) at interpolation point $\tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{\mu}}^i$ and update projection matrix
            $\mathbf{V}_{\text{du}} = \mathsf{orth}\big([\mathbf{V}_{\text{du}}\ \mathtt{mmm}(\mathscr{A}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}}^i), \mathbf{C}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}}^i), \eta, \tilde{\boldsymbol{\mu}}^i)]\big).$
 6:     Solve Eq. (12) at interpolation point $\tilde{\boldsymbol{\mu}} = \tilde{\boldsymbol{\mu}}^i_\alpha$ and update projection matrix,
            $\mathbf{V}_e = \mathsf{orth}\big([\mathbf{V}_e\ \mathbf{V}_{\text{du}}\ \mathtt{mmm}(\mathscr{A}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}}^i_\alpha), \mathbf{C}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}}^i_\alpha), \eta, \tilde{\boldsymbol{\mu}}^i_\alpha)]\big).$
 7:     $i = i + 1$.
 8:     $\tilde{\boldsymbol{\mu}}^i = \arg\max_{\tilde{\boldsymbol{\mu}} \in \Xi} \Delta(\tilde{\boldsymbol{\mu}})$.
 9:     $\tilde{\boldsymbol{\mu}}^i_\alpha = \arg\max_{\tilde{\boldsymbol{\mu}} \in \Xi} \left|\tilde{\mathbf{e}}^{\mathsf{T}}_{\text{du}}(\tilde{\boldsymbol{\mu}})\mathbf{r}_{\text{pr}}(\tilde{\boldsymbol{\mu}})\right|$.
10:     $\epsilon = \Delta(\tilde{\boldsymbol{\mu}}^i)$.
11: end while
```

solution to the dual system by considering the right hand side as the i-th row vector of $\mathbf{C}(\boldsymbol{\mu})$, namely, $\mathbf{C}^{\mathsf{T}}(:, i)$ in Eq. (9). Correspondingly, the residual $\mathbf{r}_{\text{pr}}(\tilde{\boldsymbol{\mu}})$ is obtained by solving Eq. (6) with the right hand side being $\mathbf{B}(:, j)$, the j-th column of $\mathbf{B}(\boldsymbol{\mu})$. Then, $\Delta(\tilde{\boldsymbol{\mu}}) = \arg\max_{i,j} \Delta_{ij}(\tilde{\boldsymbol{\mu}})$.

Remark 2 In order to build the projection matrices ($\mathbf{V}, \mathbf{V}_{\text{du}}, \mathbf{V}_e$), [13] makes use of the multi-moment matching (MMM) algorithm from [12]. The algorithm provides an orthogonal basis for the solution at a given interpolation point, obtained through multivariate power series expansion of the state vector. To be focused on our main contribution, we refer to [12, 13] for detailed computations. For use in the proposed algorithm, we give below the call to the algorithm in MATLAB®notation,

$$\mathbf{V}_{\mathsf{mmm}} = \mathtt{mmm}(\mathcal{A}(\tilde{\mu}_0), \mathcal{B}(\tilde{\mu}_0), \eta, \tilde{\mu}_0).$$

Here, $\mathcal{A}(\tilde{\mu}_0)$ denotes an arbitrary matrix evaluated at a given interpolation point $\tilde{\mu}_0$, $\mathcal{B}(\tilde{\mu}_0)$ corresponds to the right hand side matrix in Eqs. (5), (8) and (12), respectively. η is the number of moments to be matched in the power series. When $\eta = 0$, the MMM algorithm is equivalent to RBM, see [13] for more explanations.

4 Adaptive Training by Learning the Error Estimator in the Parameter Domain

In this section, we propose an adaptive training technique, so that the greedy algorithm starts with a training set with small cardinality, which is then updated iteratively by using a surrogate error estimator. Different works have considered surrogate models of error estimators/indicators [9, 10, 24]. All of these consider a surrogate in the context of the RBM. In this work, we deal with the frequency domain interpolatory MOR methods and focus on a surrogate model of an error estimator for the transfer function approximation error. The method we propose here is essentially an extension of the RBF-based error surrogate in [9] to the frequency domain. Beyond the work in [9], we introduce a learning process in Section 4.2 to show in detail how a surrogate estimator is constructed for any parameter in the whole parameter domain. We begin by introducing the method of RBF interpolation.

4.1 Radial Basis Functions

Radial Basis Functions belong to the family of *kernel methods* and are a popular technique to generate surrogate models of multivariate functions $f : \mathbb{R}^d \mapsto \mathbb{R}$, defined in a domain $\Omega \subset \mathbb{R}^d$. It may be the case that the function f itself is unknown and one only knows a set of inputs $M = \{\tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_2, \dots, \tilde{\boldsymbol{\mu}}_\ell\} \in \Omega$ and the corresponding function evaluations $F = \{f_1, f_2, \dots, f_\ell\} \subset \mathbb{R}$. Or, it may be the case that f is known, but very expensive to evaluate repeatedly. For either case, RBF serves to generate an interpolant $g : \mathbb{R}^d \mapsto \mathbb{R}$ of f given by

$$g(\tilde{\boldsymbol{\mu}}) = \sum_{i=1}^{\ell} c_i \Phi(\|\tilde{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}_i\|), \ \forall \tilde{\boldsymbol{\mu}} \in \Omega, \tag{16}$$

such that it interpolates the original function at the set of input points (or centers) in M, i.e., $f(\tilde{\boldsymbol{\mu}}_i) = g(\tilde{\boldsymbol{\mu}}_i)$, $i = 1, \dots, \ell$. Moreover, $|f(\tilde{\boldsymbol{\mu}}) - g(\tilde{\boldsymbol{\mu}})| \ll \texttt{tol}$, $\forall \tilde{\boldsymbol{\mu}} \in \Omega$. The functions $\Phi(\cdot)$ are the kernels defined as $\Phi(\tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_2) := \Phi(\|\tilde{\boldsymbol{\mu}}_1 - \tilde{\boldsymbol{\mu}}_2\|), \forall \tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_2 \in \Omega$. They are called radial basis functions owing to their radial dependence on $\tilde{\boldsymbol{\mu}}$. The coefficients $\{c_i\}_{i=1}^{\ell}$ are determined by solving the linear system of equations,

$$\underbrace{\begin{bmatrix} \Phi(\tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_1) & \Phi(\tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_2) & \cdots & \Phi(\tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_\ell) \\ \Phi(\tilde{\boldsymbol{\mu}}_2, \tilde{\boldsymbol{\mu}}_1) & \Phi(\tilde{\boldsymbol{\mu}}_2, \tilde{\boldsymbol{\mu}}_2) & \cdots & \Phi(\tilde{\boldsymbol{\mu}}_2, \tilde{\boldsymbol{\mu}}_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi(\tilde{\boldsymbol{\mu}}_\ell, \tilde{\boldsymbol{\mu}}_1) & \Phi(\tilde{\boldsymbol{\mu}}_\ell, \tilde{\boldsymbol{\mu}}_2) & \cdots & \Phi(\tilde{\boldsymbol{\mu}}_\ell, \tilde{\boldsymbol{\mu}}_\ell) \end{bmatrix}}_{\mathbf{R}} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_\ell \end{bmatrix}}_{c} = \underbrace{\begin{bmatrix} f(\tilde{\boldsymbol{\mu}}_1) \\ f(\tilde{\boldsymbol{\mu}}_2) \\ \vdots \\ f(\tilde{\boldsymbol{\mu}}_\ell) \end{bmatrix}}_{\mathfrak{b}}. \tag{17}$$

We need $\mathbf{R}$ to be invertible. Assuming that the centers $\tilde{\boldsymbol{\mu}}_i$ are pairwise distinct, it can be shown that $\mathbf{R}$ is positive definite for a suitable choice of the RBF $\Phi(\cdot)$ and thus

Eq. (17) has a unique solution. The class of RBF giving rise to positive definite $\mathbf{R}$ is limited. As a workaround, some additional constraints are imposed in practice, i.e.

$$\sum_{j=1}^{D} c_i p_j(\tilde{\boldsymbol{\mu}}) = 0, \quad i = 1, 2, \ldots, \ell,$$

so that a larger class of $\Phi(\cdot)$ can be admitted. The functions $p_1, p_2, \ldots, p_D$ are a basis of the polynomial space with suitable degree. In practice, we choose D to be equal to the number of scalar parameters d. With the new conditions imposed, the radial basis interpolant now becomes,

$$g(\tilde{\boldsymbol{\mu}}) := \sum_{i=1}^{\ell} c_i \Phi(\|\tilde{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}}_i\|) + \sum_{j=1}^{D} \lambda_j p_j(\tilde{\boldsymbol{\mu}}). \tag{18}$$

This results in a saddle-point system of dimension $N_{\text{RBF}} := (D + \ell) \times (D + \ell)$,

$$\begin{bmatrix} \mathbf{R} & \mathbf{P} \\ \mathbf{P}^{\mathsf{T}} & 0 \end{bmatrix} \begin{bmatrix} c \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathfrak{d} \\ 0 \end{bmatrix}. \tag{19}$$

With a proper choice of $p_1, p_2, \ldots, p_D$, the augmented coefficient matrix is positive definite for a wider choice of kernel functions $\Phi(\cdot)$. We refer to [31] for an exhaustive theoretical analysis of RBFs and the recent review paper [28] that analyses RBFs in the larger context of kernel based surrogate models.

4.2 Learning the Error Estimator over the Parameter Domain

As highlighted in the Introduction, one of the main bottlenecks of the standard greedy algorithm is that the error estimator ($\Delta(\tilde{\boldsymbol{\mu}})$) needs to be determined at every parameter in the training set.

In order to evaluate it cheaply, we first construct a surrogate model of the error estimator by learning the error estimator in the whole parameter domain using RBF interpolation. We have the multivariate function $f(\cdot) := \Delta(\tilde{\boldsymbol{\mu}})$ and the learning step involves determining the coefficients c in Eq. (19). First, we evaluate the error estimator at a small number of parameters in a coarse training set ($\Xi_c : [\tilde{\boldsymbol{\mu}}_1, \ldots, \tilde{\boldsymbol{\mu}}_{N_c}]$). These points shall serve as the centers $\tilde{\boldsymbol{\mu}}$ of the RBF interpolation with N_c as the number of centers. Note that, with regards to the discussion in Section 4.1, we have $\ell = N_c$.

Next, we define the kernel function ($\Phi(\cdot)$) and setup the linear system defined in Eq. (19). Many choices of the kernel function exist, and in the numerical experiments we have used the inverse multiquadric and the thin-plate spline kernel functions. For a deeper discussion, we refer to [31]. We note here that the assembling of the kernel matrix $\mathbf{R}$ can be done efficiently and software implementations exist to achieve this [28]. The right hand side is defined by $\mathfrak{d} := [\Delta(\tilde{\boldsymbol{\mu}}_1), \ldots, \Delta(\tilde{\boldsymbol{\mu}}_{N_c})]$.

Eq. (19) constitutes a small, dense system of linear equations. The computational cost of its evaluation scales as $O((N_c + D)^3)$. However, since N_c, D are small, the cost remains under control. Once knowing c after solving Eq. (19), the interpolant $g(\tilde{\boldsymbol{\mu}})$ of the error estimator is obtained over the whole parameter domain employing only function evaluations in (16). Thus the learned surrogate of the error estimator is $g(\tilde{\boldsymbol{\mu}})$. It is not difficult to see that computing the error estimator over the parameter domain is more expensive than using the surrogate $g(\tilde{\boldsymbol{\mu}})$.

- The cost of computing the surrogate $g(\tilde{\boldsymbol{\mu}})$ for *all the* parameter samples $\tilde{\boldsymbol{\mu}}$ in a certain parameter set with cardinality N_f is:
  - Solving small, dense ROM : $O(r^3) \times N_c$.
  - Matrix-vector product to evaluate residual : $O(nr) \times N_c$.
  - Vector-vector inner product to evaluate Eq. (14) : $O(n) \times N_c$.
  - Identify the coefficients c by solving Eq. (19): $O((N_c + D)^3)$.
  - Evaluate the interpolant through function evaluation Eq. (16) over a parameter set with cardinality N_f: $O((N_c + D)) \times N_f$.
- The cost of evaluating the error estimator $\Delta(\tilde{\boldsymbol{\mu}})$ for *all the* parameters samples $\tilde{\boldsymbol{\mu}}$ in a certain parameter set with cardinality N_f are,
  - Solving small, dense ROM : $O(r^3) \times N_f$.
  - Matrix-vector product to evaluate residual : $O(nr) \times N_f$.
  - Vector-vector inner product to evaluate Eq. (14) : $O(n) \times N_f$.

Here, n is the full order dimension of the system; the reduced size r is as small as $N_c + D$, i.e. $r \approx N_c + D$. For $N_f \gg N_c$, it is clear that computing the error estimator is more expensive than computing the surrogate.

4.3 Adaptive Choice of Interpolation Points with Surrogate Error Estimator

In Algorithm 2, we present the proposed adaptive method to choose interpolation points using a surrogate error estimator. We call the algorithm IPSUE - Interpolation Points using SUrrogate error Estimator. To follow the learning process in Subsection 4.2 in practice, we do not consider the entire domain $\mathbb{R}^d$, but a fine representation of it given by $\Xi_f := [\tilde{\boldsymbol{\mu}}_1, \ldots, \tilde{\boldsymbol{\mu}}_{N_f}]^\mathsf{T}$, containing $N_f \gg N_c$ parameters. Therefore, we consider two training sets: a coarse training set Ξ_c and a fine training set Ξ_f. In Step 4 of Algorithm 2, we perform Steps 4 - 6 from Algorithm 1. In Step 5, the error estimator is evaluated only over the coarse training set, an important distinction from Algorithm 1. In Step 6, the argument of the maximum is chosen as the next interpolation point for $\mathbf{V}$. Step 7 selects the parameter that maximizes the second summand of the error estimator and uses it as the interpolation point ($\tilde{\boldsymbol{\mu}}^i_\alpha$) for enriching $\mathbf{V}_e$ in the next iteration. As noted in [13], it is important that the interpolation points $\tilde{\boldsymbol{\mu}}^i$ and $\tilde{\boldsymbol{\mu}}^i_\alpha$ are distinct, in order to ensure that $\mathbf{V}_{\text{du}} \neq \mathbf{V}_e$. Then, in Step 8, using $\Delta(\tilde{\boldsymbol{\mu}}) \forall\, \tilde{\boldsymbol{\mu}} \in \Xi_c$ we learn the error estimator over the parameter domain (represented

Algorithm 2 Interpolation Points using SUrrogate error Estimator (IPSUE) algorithm

```
Input: System matrices $\mathbf{A}(\boldsymbol{\mu})$, $\mathbf{B}(\boldsymbol{\mu})$, $\mathbf{C}(\boldsymbol{\mu})$, coarse training set $\Xi_c$ of cardinality $N_c$, fine training
    set $\Xi_f$ of cardinality $N_f$ covering the interesting parameter ranges, tolerance $\epsilon_{tol}$.
Output: Projection matrix $\mathbf{V}$.
 1: Initialize $\mathbf{V} = [\,]$, $\mathbf{V}_{\text{du}} = [\,]$, $\mathbf{V}_e = [\,]$, $\epsilon = 1 + \epsilon_{tol}$, fix $\eta$, the number of moments to be
    matched. Set $i = 1$.
 2: Initial interpolation point $\tilde{\boldsymbol{\mu}}^1$: a random sample from $\Xi_c$ selected using rand(), $\tilde{\boldsymbol{\mu}}_\alpha^1$: another ran-
    dom sample from $\Xi_c$ selected using rand(). Here, rand() is the intrinsic MATLAB®function.
 3: while $\epsilon > \epsilon_{tol}$ do
 4:     Perform Steps 4 - 6 from Algorithm 1.
 5:     Use Eq. (14) and obtain $\Delta(\tilde{\boldsymbol{\mu}})\ \forall, \tilde{\boldsymbol{\mu}} \in \Xi_c$.
 6:     $\tilde{\boldsymbol{\mu}}^{i+1} = \arg\max_{\tilde{\boldsymbol{\mu}} \in \Xi_c} \Delta(\tilde{\boldsymbol{\mu}})$.
 7:     $\tilde{\boldsymbol{\mu}}_\alpha^{i+1} = \arg\max_{\tilde{\boldsymbol{\mu}} \in \Xi_c} \left|\tilde{\mathbf{e}}_{\text{du}}^{\mathsf{T}}(\tilde{\boldsymbol{\mu}})\mathbf{r}_{\text{pr}}(\tilde{\boldsymbol{\mu}})\right|$.
 8:     Form the RBF interpolant $g(\tilde{\boldsymbol{\mu}})$ of the error estimator $\Delta(\tilde{\boldsymbol{\mu}})$ over $\Xi_f$.
 9:     Select $n_a^{(1)}$. Identify $\left(\tilde{\boldsymbol{\mu}}_1^{(1)}, \ldots, \tilde{\boldsymbol{\mu}}_{n_a}^{(1)}\right)$ from $\Xi_f$ with the largest errors for $g(\tilde{\boldsymbol{\mu}})$. Usually,
        $n_a^{(1)} = 1$.
10:     Update the coarse training set with the newly identified parameters,
        $\Xi_c := \left[\Xi_c \cup \left(\tilde{\boldsymbol{\mu}}_1^{(1)}, \ldots, \tilde{\boldsymbol{\mu}}_{n_a}^{(1)}\right)\right]$.
11:     $i = i + 1$.
12:     $\epsilon = \Delta(\tilde{\boldsymbol{\mu}}^i)$.
13: end while
```

by Ξ_f) by determining $g(\tilde{\boldsymbol{\mu}}), \forall\, \tilde{\boldsymbol{\mu}} \in \Xi_f$. In Step 9, $n_a^{(1)}$ new parameters are identified from Ξ_f such that they have the largest errors measured by $g(\tilde{\boldsymbol{\mu}})$. The coarse training set is then updated with the newly identified points.

5 Numerical examples

In this section, we provide numerical results to show the efficiency of the proposed IPSUE algorithm. The first example is from circuit simulation used in [13]. It is characterized by its large parameter range. The second is a benchmark example of a microthruster device, from the MORwiki collection [30]. This model has 4 parameters. The final example is a finite element model of a waveguide filter, from [26]. All numerical tests were performed in MATLAB®2015a, on a laptop with Intel®Core™i5-7200U @ 2.5 GHZ, with 8 GB of RAM. In the numerical results, N_μ refers to the cardinality of the fixed training set Ξ, used in Algorithm 1, N_c, N_f are, respectively, the cardinality of the coarse and fine training sets used in Algorithm 2 and finally N_t denotes the cardinality of the parameter test set Ξ_t used for validating the accuracy of the final ROMs constructed by Algorithms 1 and 2. Also, we use the same test sets for comparing the performances of Algorithms 1 and 2.

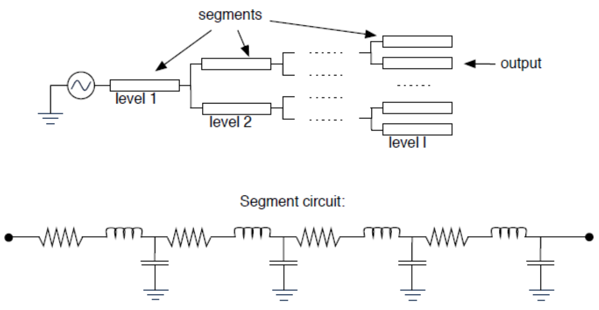


Fig. 1: RLC interconnect circuit.

Setting	Value
n	6134
ϵ_{tol}	10^{-3}
N_μ	90 parameters
N_c	$\{21, 27\}$ parameters
N_f	200 parameters
N_t	900 paramters
η	3

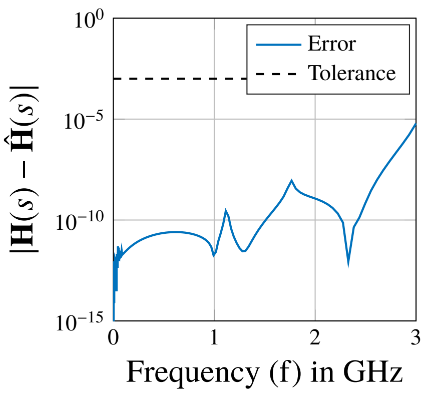
Table 1: Simulation settings for the RLC interconnect circuit.

5.1 RLC Interconnect Circuit

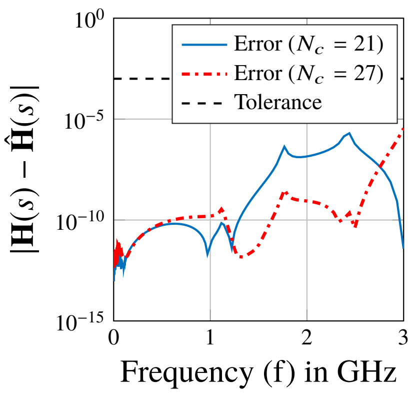
This example models the large-scale interconnects in integrated circuit (IC) design. It is represented in Fig. 1. The discretized model has dimension $n = 6134$. It is a non-parametric system in time domain, but in the frequency domain, the frequency f is considered as the parameter and the interpolation points are selected from a wide frequency range: $f \in [0, 3]$GHz. Table 1 gives the simulation settings used for implementing Algorithms 1 and 2 to generate the reduced order models for this example.

Test 1: Algorithm 1 applied to RLC model

To enable comparison, we use the same training set Ξ used in [13]. It consists of 90 samples covering the range of interest. The sampled frequencies are given by $f_i = 3 \times 10^{i/10}$, $s_i = 2\pi \jmath f_i$ with $i = 1, 2, \ldots, 90$. Algorithm 1 converges to the set tolerance in just 3 iterations. The obtained ROM is of dimension $r = 20$. On average, it takes 3.3 seconds for the algorithm to converge. For the sake of robustness, we test



(a) Algorithm 1: Error $|\mathbf{H}(s) - \hat{\mathbf{H}}(s)|$ over the test set.



(b) Algorithm 2: Error $|\mathbf{H}(s) - \hat{\mathbf{H}}(s)|$ over the test set.

Fig. 2: Results for the RLC model.

the ROM on a different set of test parameters Ξ_t with $N_t = 900$ parameters. Fig. 2a shows the error of $\hat{\mathbf{H}}(s)$ for the parameters in Ξ_t.

Test 2: Algorithm 2 applied to RLC model

Next, we test Algorithm 2 on the RLC interconnect model. For this, we consider two different coarse training sets Ξ_c of cardinality $21, 27$ sampled as, $\Xi_c^j = 3 \times 10^{j/10}$, $j = 1, 2, \ldots, 21$ and $j = 1, 2, \ldots, 27$. We consider different samplings for the fine training set in order to numerically illustrate that the proposed algorithm is independent of the kind of sampling used. The fine training set Ξ_f consists of 200 logarithmically distributed parameters in the first case and in the second case contains 200 parameters distributed as $\Xi_f^j = 3 \times 10^{j/10}$, $j = 1, 2, \ldots, 200$. For the RBF interpolation, we use thin-plate splines as the kernel function. It is given by $\Phi(\tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_2) := (\|\tilde{\boldsymbol{\mu}}_1 - \tilde{\boldsymbol{\mu}}_2\|_2)^2 \log_e(\|\tilde{\boldsymbol{\mu}}_1 - \tilde{\boldsymbol{\mu}}_2\|_2)$. Algorithm 2 converges to the specified tolerance in just 3 iterations for both choices of Ξ_c, with $n_a^{(1)} = 1$. In the first case, the obtained ROM is of dimension $r = 21$ and takes 1.6 seconds to converge in average. The second case results in a ROM of dimension 21 and takes 1.7 seconds on average to converge to the defined tolerance. Fig. 2b shows the error of $\hat{\mathbf{H}}(s)$ at parameters in the test set Ξ_t produced by the ROM obtained using Algorithm 2, with two different coarse training sets. Clearly, Algorithm 2 takes less time than Algorithm 1, while still producing a ROM that is uniformly below the tolerance, on an independent test set.

Setting	Value
n	4257
ϵ_{tol}	10^{-4}
N_μ	625 parameters, log-sampled
N_c	256 parameters, log-sampled
N_f	2401 parameters, log-sampled
N_t	1000 parameters, randomly sampled
η	1

Table 2: Simulation settings for the thermal model.

5.2 Thermal Model

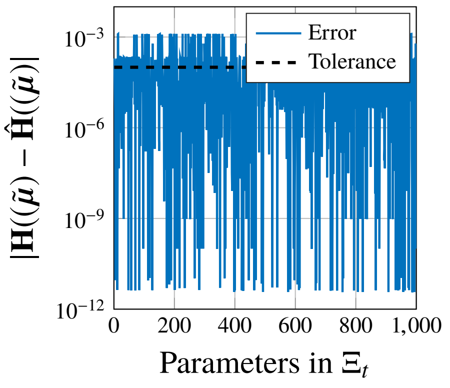
The second example is the model of the heat transfer inside a microthruster unit [30]. It is obtained after spatial discretization using the finite element method and has dimension $n = 4257$. The governing equation is given as,

$$\begin{aligned}\mathbf{E}\dot{\mathbf{x}}(t) &= (\mathbf{A}_0 - \sum_{i=1}^{3} h_i \mathbf{A}_i)\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t),\\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}.\end{aligned}$$

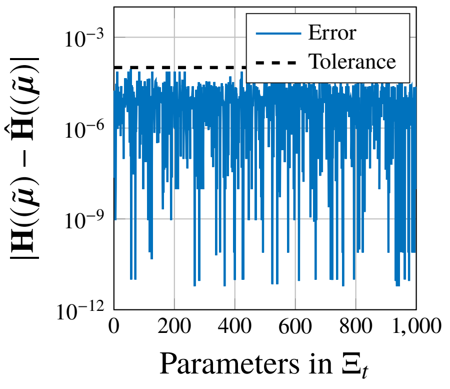
Here, $\mathbf{E}, \mathbf{A}_0$ are symmetric sparse matrices representing the heat capacity and heat conductivity, respectively. $\mathbf{A}_i$, $i \in \{1, 2, 3\}$ are diagonal matrices governing the boundary condition. The parameters $h_1, h_2, h_3 \in [1,\ 10^4]$ represent, respectively, the film coefficients of the top, bottom and side of the microthruster unit. We transform the above system to the frequency domain and apply Algorithm 1 and Algorithm 2. In the frequency domain, the system has four parameters $\tilde{\boldsymbol{\mu}} := (s, h_1, h_2, h_3)$ with $s = \jmath 2\pi f$. The frequency range of interest is $f \in [10^{-2},\ 10^2]$ Hz. The tolerance for the ROM is set as 10^{-4}.

Test 3: Algorithm 1 applied to the thermal model

Owing to the wide range of parameters, we consider a large fixed training set (Ξ). To construct it, we consider 5 logarithmically-spaced samples for each of the 4 parameters and form a grid consisting of 5^4 samples. For the test set Ξ_t, we form a grid of 8^4 logarithmically spaced parameters and randomly select 1000 parameters from it. The greedy algorithm takes 10 iterations to converge and results in a ROM of size $r = 86$. On an average over 5 runs, the greedy algorithm takes 254 seconds to converge. In Fig. 3a, we see the performance of the resulting ROM over Ξ_t. For several parameters, the ROM fails to meet the desired tolerance. This indicates that the training set was not fine enough to capture all the variations in the solutions over the parameter domain.



(a) Algorithm 1: Error $|\mathbf{H}(\tilde{\boldsymbol{\mu}}) - \hat{\mathbf{H}}((\tilde{\boldsymbol{\mu}})|$ over the test set.



(b) Algorithm 2: Error $|\mathbf{H}((\tilde{\boldsymbol{\mu}}) - \hat{\mathbf{H}}((\tilde{\boldsymbol{\mu}})|$ over the test set.

Fig. 3: Results for the thermal model.

Test 4: Algorithm 2 applied to the thermal model

We now consider Algorithm 2 applied to the thermal model. The coarse training set Ξ_c has 4^4 parameters, with logarithmic sampling. The fine training set Ξ_f has 7^4 parameters. For the RBF interpolation, we make use of thin-plate splines as the kernel function. Further, we set $n_a^{(1)} = 1$ in Step 9 of Algorithm 2 so that the coarse training set is updated with one new parameter per iteration. The same test set as in Test 3 is used. The resulting ROM has order $r = 85$ and its error over the test set is below the tolerance, as shown in Fig. 3b. The algorithm took 162 seconds to converge. The runtime was measured as an average over 5 independent runs of the algorithm. Compared with Algorithm 1, Algorithm 2 is able to meet the required tolerance with a much smaller training set and also in shorter time.

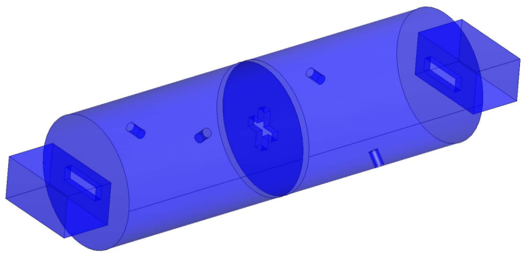


Fig. 4: Dual-mode waveguide filter model from [26].

5.3 Dual-Mode Circular Waveguide Filter

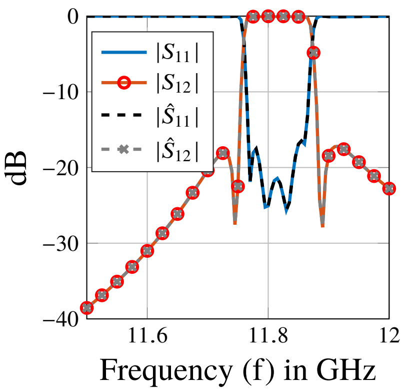
The next example is a MIMO system based on the model of a dual-mode circular waveguide filter from [26], see Fig. 4. It is a type of narrow bandpass filter widely used in satellite communication due to its power handling capabilities. Its operation is governed by the time-harmonic Maxwell's Equations. After discretization in space, the governing equations of the filter can be represented in the form of Eq. (5). The system consists of just the frequency parameter $s := \jmath 2\pi f$, where $f \in [11.5, 12]$ GHz is the operating frequency band of the filter. The affine form of the system matrix is, $\mathscr{A}(s) := \mathcal{S} + s^2\mathcal{T}$ and $\mathbf{B}(s) := s\mathcal{Q}$. We have $\mathcal{S}, \mathcal{T} \in \mathbb{R}^{n\times n}$ and $\mathcal{Q} \in \mathbb{R}^{n\times 2}$, with $n = 36426$. The system has two inputs and two outputs. Table 3 summarizes the simulation settings. The quantity of interest are the scattering parameters, obtained via post-processing [27] from the system output $\mathbf{y}(s) := \mathcal{Q}^\mathsf{T}\mathbf{X}(s)$. It is easy to see that $\mathbf{y}(s)$ has the same expression as $\mathbf{H}(\tilde{\boldsymbol{\mu}})$ in Eq. (3) for $\tilde{\boldsymbol{\mu}} = s$. The error estimator $\Delta(\tilde{\boldsymbol{\mu}})$ in Section 3 can be directly applied to estimate the error of $\hat{\mathbf{y}}(s)$ computed by the ROM. See [13] for detailed analysis. Since the system is MIMO, the scattering parameters at a given s are in the form of a complex-valued matrix given by

$$\mathbf{S} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}.$$

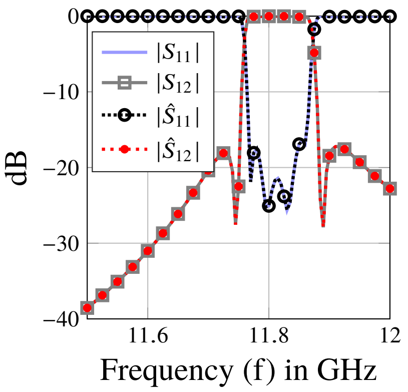
Scattering parameters are important in characterizing the performance of filters [25].

Test 5: Algorithm 1 applied to the dual-mode filter

Applying Algorithm 1 with the fixed training set Ξ to the model results in a ROM of size $r = 10$ with the greedy algorithm taking 5 iterations to converge. Since this example is a MIMO system, we make use of Eq. (15). The average runtime over 5 independent runs of Algorithm 1 was found to be 46 seconds. Fig. 5a plots the scattering parameters computed from FOM simulations and those obtained from



(a) Algorithm 1: Scattering parameters for the test set.



(b) Algorithm 2: Scattering parameters for the test set.

Fig. 5: Scattering parameters for the dual-mode filter.

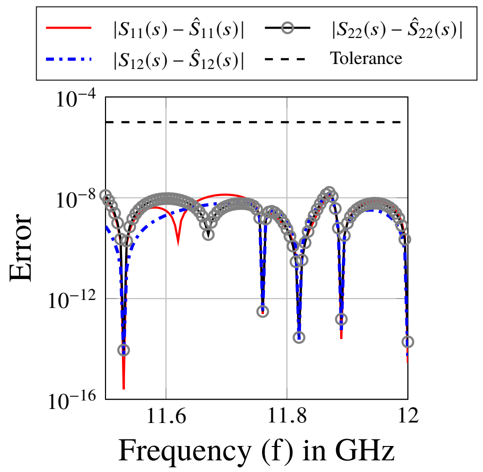
Setting	Value
n	36426
ϵ_{tol}	10^{-5}
N_μ	51 parameters, uniformly spaced
N_c	17 parameters, uniformly spaced
N_f	500 parameters, sampled randomly
N_t	101 parameters, uniformly spaced
η	1

Table 3: Simulation settings for the dual-mode filter.

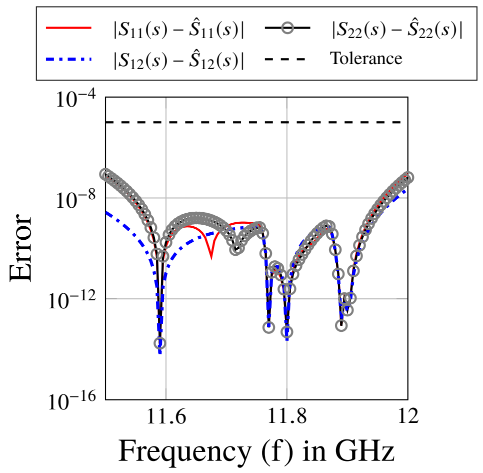
the ROM at the parameters in the test set. We plot the absolute values of full order scattering parameters S_{11}, S_{12} and the corresponding reduced ones $\hat{S}_{11}, \hat{S}_{12}$ on a Decibel scale. In Fig. 6a we plot the error of the scattering parameters $\hat{S}_{11}, \hat{S}_{12}, \hat{S}_{22}$ computed from the ROM, over the test set Ξ_t. Note that since $S_{12} = S_{21}$, we only show the error $|S_{12} - \hat{S}_{12}|$.

Test 6: Algorithm 2 applied to the Dual-mode filter

In Step 8 of Algorithm 2, we construct an RBF surrogate for each of Δ_{ij}, $i, j \in \{1, 2\}$ in Eq. (15) for this MIMO system. $n_a^{(1)}$ is set to be 1 and inverse multiquadric is used as the kernel function. It is given by $\Phi(\tilde{\boldsymbol{\mu}}_1, \tilde{\boldsymbol{\mu}}_2) := \frac{1}{1+\left(\gamma\|\tilde{\boldsymbol{\mu}}_1-\tilde{\boldsymbol{\mu}}_2\|_2\right)^2}$. γ is a user-defined parameter and we set $\gamma = 16$ in our experiments. We pick the maximum among the four surrogates and add the corresponding parameter to the coarse training set, i.e.



(a) Algorithm 1: Error of the scattering parameters computed from the ROM over the test set.



(b) Algorithm 2: Error of the scattering parameters computed from the ROM over the test set.

Fig. 6: Results for the dual-mode filter.

in Step 9 of Algorithm 2, we replace $g(\tilde{\boldsymbol{\mu}})$ with $\max\limits_{i,j \in \{1,2\}} g_{i,j}(\tilde{\boldsymbol{\mu}})$. Algorithm 2 results in a ROM that is of the same size as the ROM from Test 5 ($r = 10$). However, on average, Algorithm 2 only needs 24 seconds to converge, almost half that of the time required in Test 5. In Fig. 6b we plot the errors of the scattering parameters computed from the ROM over the test set Ξ_t. Fig. 5b plots the scattering parameters from the FOM simulations and those computed by the ROM. Both algorithms result in ROMs meeting the specified tolerance, but Algorithm 2 requires much shorter time to generate the ROM.

6 Conclusion

In this work, we have proposed IPSUE, an adaptive algorithm for updating the training set and choosing the interpolation points for frequency domain MOR methods. Our target applications are cases where the problem parameters vary over a wide range of values, or the parameter space dimension is larger than two. In either of these cases, many interpolatory MOR algorithms may take a long time to generate the ROM. Moreover, a naive, heuristic sampling of the parameter training set may result in a ROM that is not robust. IPSUE offers a viable means to generate reliable ROMs that satisfy the user-defined tolerance and at the same time, without being offline expensive. The illustrated numerical examples show that it is a promising approach. As future work, we plan to apply the algorithm to more complex models.

Acknowledgements The first author is affiliated to the International Max Planck Research School for Advanced Methods in Process and Systems Engineering (IMPRS-ProEng).

References

1. Antoulas, A.C.: Approximation of Large-Scale Dynamical Systems, *Adv. Des. Control*, vol. 6. SIAM Publications, Philadelphia, PA (2005). DOI 10.1137/1.9780898718713
2. Antoulas, A.C., Sorensen, D.C., Gugercin, S.: A survey of model reduction methods for large-scale systems. Contemp. Math. **280**, 193–219 (2001)
3. Baur, U., Beattie, C.A., Benner, P., Gugercin, S.: Interpolatory projection methods for parameterized model reduction. SIAM J. Sci. Comput. **33**(5), 2489–2518 (2011). DOI 10.1137/090776925
4. Baur, U., Benner, P., Feng, L.: Model order reduction for linear and nonlinear systems: A system-theoretic perspective. Arch. Comput. Methods Eng. **21**(4), 331–358 (2014). DOI 10.1007/s11831-014-9111-2
5. Bechtold, T., Rudnyi, E., Korvink, J.: Error indicators for fully automatic extraction of heat-transfer macromodels for MEMS. Micromech. Microeng. **15**(3), 430–440 (2004). DOI 10.1088/0960-1317/15/3/002
6. Benner, P., Goyal, P., Pontes Duff, I.: Identification of dominant subspaces for linear structured parametric systems and model reduction. e-prints 1910.13945, arXiv (2019). URL `https://arxiv.org/abs/1910.13945`. Math.NA
7. Benner, P., Gugercin, S., Willcox, K.: A survey of model reduction methods for parametric systems. SIAM Review **57**(4), 483–531 (2015). DOI 10.1137/130932715
8. Benner, P., Mehrmann, V., Sorensen, D.C.: Dimension Reduction of Large-Scale Systems, *Lect. Notes Comput. Sci. Eng.*, vol. 45. Springer-Verlag, Berlin/Heidelberg, Germany (2005)
9. Chellappa, S., Feng, L., Benner, P.: An adaptive sampling approach for the reduced basis method. e-prints 1910.00298, arXiv (2019). URL `https://arxiv.org/abs/1910.00298`. Math.NA

10. Drohmann, M., Carlberg, K.: The ROMES method for statistical modeling of reduced-order-model error. SIAM/ASA Journal on Uncertainty Quantification **3**(1), 116–145 (2015). DOI 10.1137/140969841
11. Feng, L., Antoulas, A.C., Benner, P.: Some *a posteriori* error bounds for reduced-order modelling of (non-)parametrized linear systems. ESAIM: Math. Model. Numer. Anal. **51**(6), 2127–2158 (2017). DOI 10.1051/m2an/2017014
12. Feng, L., Benner, P.: Reduced Order Methods for modeling and computational reduction, MS&A Series, vol. 9, chap. 6: A robust algorithm for parametric model order reduction based on implicit moment matching, pp. 159–186. Springer-Verlag, Berlin, Heidelberg, New York (2014)
13. Feng, L., Benner, P.: A new error estimator for reduced-order modeling of linear parametric systems. IEEE Trans. Microw. Theory Techn. **67**(12), 4848–4859 (2019)
14. Feng, L., Korvink, J.G., Benner, P.: A fully adaptive scheme for model order reduction based on moment-matching. IEEE Transactions on Components, Packaging and Manufacturing Technology **5**(12), 1872–1884 (2015). DOI 10.1109/TCPMT.2015.2491341
15. Grimme, E.J.: Krylov projection methods for model reduction. Ph.D. thesis, University of Illinois at Urbana-Champaign, USA (1997)
16. Gugercin, S., Antoulas, A.C., Beattie, C.: $\mathcal{H}_2$ model reduction for large-scale linear dynamical systems. SIAM J. Matrix Anal. Appl. **30**(2), 609–638 (2008). DOI 10.1137/060666123
17. Gugercin, S., Stykel, T., Wyatt, S.: Model reduction of descriptor systems by interpolatory projection methods. SIAM J. Sci. Comput. **35**(5), B1010–B1033 (2013). DOI 10.1137/130906635
18. Haasdonk, B., Dihlmann, M., Ohlberger, M.: A training set and multiple bases generation approach for parameterized model reduction based on adaptive grids in parameter space. Math. Comput. Model. Dyn. Syst. **17**(4), 423–442 (2011)
19. Hesthaven, J.S., Stamm, B., Zhang, S.: Efficient greedy algorithms for high-dimensional parameter spaces with applications to empirical interpolation and reduced basis methods. ESAIM: Math. Model. Numer. Anal. **48**(1), 259–283 (2014)
20. Hetmaniuk, U., Tezaur, R., Farhat, C.: An adaptive scheme for a class of interpolatory model reduction methods for frequency response problems. Internat. J. Numer. Methods Engrg. **93**(10), 1109–1124 (2013). DOI 10.1002/nme.4436. URL `https://doi.org/10.1002/nme.4436`
21. Hund, M., Mlinarić, P., Saak, J.: An $\mathcal{H}_2 \otimes \mathcal{L}_2$-optimal model order reduction approach for parametric linear time-invariant systems. Proc. Appl. Math. Mech. **18**(1), e201800084 (2018). DOI 10.1002/pamm.201800084
22. Lee, H.J., Chu, C.C., Feng, W.S.: An adaptive-order rational Arnoldi method for model-order reductions of linear time-invariant systems. Linear Algebra Appl. **415**(2), 235–261 (2006). DOI https://doi.org/10.1016/j.laa.2004.10.011. Special Issue on Order Reduction of Large-Scale Systems
23. Maday, Y., Stamm, B.: Locally adaptive greedy approximations for anisotropic parameter reduced basis spaces. SIAM J. Sci. Comput. **35**(6), A2417–A2441 (2013)
24. Paul-Dubois-Taine, A., Amsallem, D.: An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models. Internat. J. Numer. Methods Engrg. **102**(5), 1262–1292 (2015)
25. Pozar, D.M.: Microwave Engineering. Wiley, New York, USA (1998)
26. de la Rubia, V., Mrozowski, M.: A compact basis for reliable fast frequency sweep via the reduced-basis method. IEEE Trans. Microw. Theory Techn. **66**(10), 4367–4382 (2018)
27. de la Rubia, V., Razafison, U., Maday, Y.: Reliable fast frequency sweep for microwave devices via the reduced-basis method. IEEE Trans. Microw. Theory Techn. **57**(12), 2923–2937 (2009)
28. Santin, G., Haasdonk, B.: Kernel methods for surrogate modeling. e-prints 1907.10556, arXiv (2019). URL `https://arxiv.org/abs/1907.10556`. Math.NA
29. Schilders, W.H.A., van der Vorst, H.A., Rommes, J.: Model Order Reduction: Theory, Research Aspects and Applications. Springer-Verlag, Berlin, Heidelberg (2008)
30. The MORwiki Community: MORwiki - Model Order Reduction Wiki. `http://modelreduction.org`

31. Wendland, H.: Scattered Data Approximation, *Cambridge Monographs on Applied and Computational Mathematics*, vol. 17. Cambridge University Press, Cambridge (2005)
32. Wolf, T., Panzer, H., Lohmann, B.: Gramian-based error bound in model reduction by Krylov subspace methods. IFAC Proceedings Volumes **44**(1), 3587–3592 (2011). DOI https://doi.org/10.3182/20110828-6-IT-1002.02809