

An Adaptive Pole-Matching Method for Interpolating Reduced-Order Models

Yao Yue, Lihong Feng, and Peter Benner,

Abstract—An adaptive parametric reduced-order modeling method based on interpolating poles of reduced-order models is proposed in this paper. To guarantee correct interpolation, a pole-matching process is conducted to determine which poles of two reduced-order models correspond to the same parametric pole. First, the pole-matching in the scenario of parameter perturbation is discussed. It is formulated as a combinatorial optimization problem and solved by a branch and bound algorithm. Then, an adaptive framework is proposed to build repository ROMs at adaptively chosen parameter values, which well represent the parameter domain of interest. To achieve this, we propose techniques including a predictor-corrector strategy and an adaptive refinement strategy, which enable us to use larger steps to explore the parameter domain of interest with good accuracy. The framework also consists of regression as an optional post-processing phase to further reduce the data storage. The advantages over other parametric reduced-order modeling approaches are, e.g., compatibility with any model order reduction method, constant size of the parametric reduced-order model with respect to the number of parameters, and capability to deal with complicated parameter dependency.

Index Terms—numerical acceleration techniques, reduced order modeling, model-order reduction, frequency domain techniques, spectrum analysis

I. INTRODUCTION

PARAMETRIC model order reduction (PMOR) [1] has become a popular tool in the last decades to drastically reduce the computational cost of model-based parametric studies in many fields in engineering, e.g., circuit simulations [2], structural mechanics [3], [4] and chromatography [5].

Among the many methods proposed, projection-based PMOR methods have received the most research attention [6], [7], [1]. Most of these methods first build a global basis by collecting data/derivative information at different parameter values, e.g., snapshots for time-domain systems [6] and (cross-)moments in frequency-domain systems [7]. Then, these methods project the parametric full-order model (FOM) onto the subspace spanned by the basis to compute a parametric reduced-order model (pROM). However, these methods have several disadvantages. First, they assume that a parametric

FOM is available in the state-space form, which is not always possible in industrial applications. For example, many engineering programs can only compute a FOM from the original partial differential equations at a given parameter value and they may use different meshes for discretization at different parameter values. In addition, they tend to lose efficiency when the number of parameters increases because the dimension of the subspace often increases fast with the number of parameters.

Recently, there are some research efforts that build pROMs by interpolating local matrices or bases. In [8], it was proposed to interpolate the basis matrices on the geodesic of the Stiefel manifold. However, the costs of both data storage and computation are high and according to our experience in [9], it is difficult to achieve high accuracy with this approach. Another line of research is to conduct PMOR via the interpolation of local matrices. As is shown in [10], directly interpolating the local reduced-order state-space matrices does not work in general since the ROMs are usually in different state-space coordinate systems. Therefore, it was proposed in [10] to compute a global basis from the local bases and to transform all ROMs to a consistent state-space coordinate system defined by the global basis. Another approach was proposed in [11] to apply a congruence transformation to obtain “consistent bases” in a heuristic sense, which is done by solving a Procrustes optimization problem. It has also been proposed to interpolate the local frequency response functions (FRFs) [12], [13]. However, as we showed in [9], these methods do not work in general. For a comprehensive review of these methods, we refer to [9].

In [14], [9], we proposed a pole-matching based parametric reduced-order modeling method, which builds a pROM by interpolating given nonparametric reduced-order models (ROMs). It interpolates the positions and residues of all poles of the local ROMs, which has a clear physical meaning and removes all additional degrees of freedom introduced by realization. This method only requires these ROMs to be accurate enough, no matter by what methods they are built. We showed that this method can even interpolate a ROM built by a projection-based method and a ROM built by a data-driven method. To match the poles between different ROMs, we proposed to solve a combinatorial optimization problem. However, we did not provide an efficient algorithm to solve this optimization problem: using a brutal-force method quickly becomes too expensive computationally. Furthermore, misleading matching results may occur if the pole distributions of the two ROMs are very different.

In this paper, we improve the efficiency and robustness of

Y. Yue, L. Feng and P. Benner are with Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

Manuscript submitted on July 30, 2019.

This paper is an expanded version from the IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization, Boston, MA, USA, May 29-31, 2019.

©2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

the pole-matching based parametric reduced-order modeling method using the following techniques:

- 1) The branch and bound method for pole-matching. To guarantee correct pole-matching, we propose to use the optimization approach only in the case of pole perturbation, i.e., the corresponding poles of the two ROMs only differ slightly. The proposed branch and bound method is very efficient for this scenario because most enumeration cases are normally branched out thanks to the good initial guess.
- 2) An adaptive framework for parametric reduced-order modeling. This framework serves as an efficient and accurate way to select the parameter values adaptively, at which we build repository ROMs in order to represent the parameter range of interest well. A predictor-corrector technique and an adaptive refinement strategy are proposed to explore the parameter domain of interest using larger-steps with good accuracy. A regression technique is also proposed as a post-processing phase to further reduce the data storage.

This paper is extended from our conference paper [15]. Compared to [15], we propose a new adaptive parametric reduced-order modeling framework, which consists of new techniques like adaptive refinement and regression. The new framework is more accurate and more robust. Our numerical results show that the new method not only improves the relative error from a magnitude of 10^{-2} to a magnitude of 10^{-4} , but also saves the storage of the pROM. Last but not least, we discuss the branch and bound method in more detail.

This paper is constructed as follows. In Section II, we briefly review the results in [14]. Then in Section III, we discuss matching of perturbed poles and propose the branch and bound method. Based on this, a framework to adaptively build a pROM is proposed in IV. We end the paper with the numerical results in Section V and conclusions in Section VII.

II. BACKGROUND: THE POLE-MATCHING PMOR METHOD

Assume that we have a series of ROMs and denote the ROM built for the parameter value p_i ($i = 1, 2, \dots, n_p$) by $(A^{(i)}, B^{(i)}, C^{(i)})$:

$$\begin{aligned} (sI - A^{(i)})x^{(i)}(s) &= B^{(i)}u(s), \\ y(s) &= C^{(i)}x^{(i)}(s), \end{aligned} \quad (1)$$

where $A^{(i)} \in \mathbb{R}^{k \times k}$, $B^{(i)} \in \mathbb{R}^{k \times 1}$, and $C^{(i)} \in \mathbb{R}^{1 \times k}$. Assume further that for each i , all eigenvalues of $A^{(i)}$ are simple.

A. The Pole-Residue Realization

Since a ROM has infinitely many equivalent realizations, the simple idea of interpolating $A^{(i)}$, $B^{(i)}$ and $C^{(i)}$ to obtain a pROM normally does not work [10]. In [14], we proposed to convert all ROMs to a unified pole-residue realization

$$\begin{aligned} (sI - \Lambda^{(i)})x^{(i)}(s) &= \overline{B}^{(i)}u(s), \\ y(s) &= \overline{C}^{(i)}x^{(i)}(s), \end{aligned} \quad (2)$$

where using the notation $M_{\beta}^{\alpha, T}$ for $(M_{\beta}^{\alpha})^T$, we define

$$\Lambda^{(i)} = \text{diag} \left[\Lambda_1^{(i)}, \Lambda_2^{(i)}, \dots, \Lambda_m^{(i)} \right], \quad (3)$$

$$\overline{B}^{(i)} = \left[\overline{B}_1^{(i), T}, \overline{B}_2^{(i), T}, \dots, \overline{B}_m^{(i), T} \right]^T, \quad (4)$$

$$\overline{C}^{(i)} = \left[\overline{C}_1^{(i)}, \overline{C}_2^{(i)}, \dots, \overline{C}_m^{(i)} \right]. \quad (5)$$

The pole-residue realization is closely related to the rational form of the transfer function $H^{(i)}(s) = \frac{y(s)}{u(s)}$ of (1):

$$H^{(i)}(s) = \sum_{j=1}^{n_s} \frac{\overline{C}_j^{(i)}}{s - \lambda_j^{(i)}} + \sum_{j=1}^{n_d} \frac{\overline{C}_{j,1}^{(i)}(s - a_j^{(i)}) - \overline{C}_{j,2}^{(i)}b_j^{(i)}}{(s - a_j^{(i)})^2 + (b_j^{(i)})^2}, \quad (6)$$

where for a real eigenvalue $\lambda_j^{(i)}$, we have $\Lambda_j^{(i)} = \lambda_j^{(i)}$, $\overline{B}_j^{(i)} = 1$ and $\overline{C}_j^{(i)} \in \mathbb{R}$, which corresponds to the term

$$\frac{\overline{C}_j^{(i)}}{s - \lambda_j^{(i)}} = \overline{C}_j^{(i)}(s - \lambda_j^{(i)})^{-1}\overline{B}_j^{(i)}, \quad (7)$$

and for a pair of conjugate complex eigenvalues $a_j \pm ib_j$, we have $\Lambda_j^{(i)} \in \mathbb{R}^{2 \times 2}$, $\overline{B}_j^{(i)} \in \mathbb{R}^{2 \times 1}$, $\overline{C}_j^{(i)} \in \mathbb{R}^{1 \times 2}$, which corresponds to the term

$$\begin{aligned} & \frac{\overline{C}_{j,1}^{(i)}(s - a_j^{(i)}) - \overline{C}_{j,2}^{(i)}b_j^{(i)}}{(s - a_j^{(i)})^2 + (b_j^{(i)})^2} \\ &= \left[\overline{C}_{j,1}^{(i)}, \overline{C}_{j,2}^{(i)} \right] \left(sI - \begin{bmatrix} a_j^{(i)} & b_j^{(i)} \\ -b_j^{(i)} & a_j^{(i)} \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &\triangleq \overline{C}_j^{(i)}(s - \Lambda_j^{(i)})^{-1}\overline{B}_j^{(i)}. \end{aligned}$$

Therefore, the positions of the poles are stored in $\Lambda^{(i)}$ and the residues of poles are stored in $\overline{C}^{(i)}$. The matrix $\overline{B}^{(i)}$ contains no useful information: $\overline{B}_j^{(i)} \equiv 1$ for a real pole and $\overline{B}_j^{(i)} \equiv [1, 0]^T$ for a pair of conjugate complex poles.

For more details, we refer to [14], [9].

B. Efficient Storage

Assuming that $A^{(i)}$ has n_s real eigenvalues and n_d pairs of conjugate complex eigenvalues, we can store the ROM in the pole-residue realization with two matrices [14]:

$$D_i \in \mathbb{C}^{n_d \times 4} \quad \text{and} \quad S_i \in \mathbb{C}^{n_s \times 2}. \quad (8)$$

The j -th row of S_i is $(\lambda_j^{(i)}, \overline{C}_j^{(i)})$, which corresponds to the real pole $\frac{\overline{C}_j^{(i)}}{s - \lambda_j^{(i)}}$, while the j -th row of D_i is $(a_j^{(i)}, b_j^{(i)}, \overline{C}_{j,1}^{(i)}, \overline{C}_{j,2}^{(i)})$, which corresponds to a pair of conjugate complex poles $\frac{\overline{C}_{j,1}^{(i)}(s - a_j^{(i)}) - \overline{C}_{j,2}^{(i)}b_j^{(i)}}{(s - a_j^{(i)})^2 + (b_j^{(i)})^2}$. The order of the rows in D_i and S_i depends on their order in $\Lambda^{(i)}$: the order of rows in S_i (D_i) is the order of real (complex conjugate) poles in $\Lambda^{(i)}$.

III. MATCHING OF PERTURBED POLES: A BRANCH AND BOUND METHOD

To interpolate two sufficiently accurate ROMs, say (D_1, S_1) and (D_2, S_2) , to obtain a pROM, we first determine which poles in (D_1, S_1) and (D_2, S_2) correspond to the same parametric pole. This is what we call pole-matching. The order of the poles are influenced by: 1) the MOR algorithm; 2) the eigenvalue decomposition algorithm [16] that we use to compute the pole-residue realization (2). In this section, we discuss matching of perturbed poles under Assumption III.1. The matching of distant poles will be the topic of Section IV.

Assumption III.1. *The assumptions for the branch and bound method.*

- 1) *The sizes of D_1 and S_1 equal those of D_2 and S_2 , respectively;*
- 2) *All poles of the ROM (D_2, S_2) are perturbations of poles of ROM (D_1, S_1) , but for a pair of perturbed poles, their order in the D/S vectors may be different. More precisely, $r(\text{ROM}_1, \text{ROM}_2)$ is small, with r defined later in (26).*

Under Assumption III.1, pole-matching can be done by solving an optimization problem, i.e., find an optimal way to rearrange the rows of (D_2, S_2) so that it is the closest to (D_1, S_1) in some sense.

A. Matching of Perturbed Poles via Optimization

We first define two mapping vectors: $v_d = [v_{d,1}, v_{d,2}, \dots, v_{d,n_d}]$ for conjugate complex poles and $v_s = [v_{s,1}, v_{s,2}, \dots, v_{s,n_s}]$ for real poles, which satisfy

$$\{v_{d,1}\} \cup \{v_{d,2}\} \cup \dots \cup \{v_{d,n_d}\} = \{1, 2, \dots, n_d\}, \quad (9)$$

and

$$\{v_{s,1}\} \cup \{v_{s,2}\} \cup \dots \cup \{v_{s,n_s}\} = \{1, 2, \dots, n_s\}. \quad (10)$$

Then we define the mapping \mathcal{M} , which rearranges the rows of D (or S) according to the index vector v_d (or v_s):

$$\mathcal{M}(v_d; D) = [d_{v_{d,1}}^T, d_{v_{d,2}}^T, \dots, d_{v_{d,n_d}}^T]^T, \quad (11)$$

$$\mathcal{M}(v_s; S) = [s_{v_{s,1}}^T, s_{v_{s,2}}^T, \dots, s_{v_{s,n_s}}^T]^T, \quad (12)$$

where $D = [d_1^T, d_2^T, \dots, d_{n_d}^T]^T$ and $S = [s_1^T, s_2^T, \dots, s_{n_s}^T]^T$.

In the case of pole perturbation, we achieve pole-matching by solving the following optimization problem

$$\min_{v_d, v_s} f(v_d, v_s) = \|D_1 W_d - \mathcal{M}(v_d, D_2) W_d\|_F^2 + \|S_1 W_s - \mathcal{M}(v_s, S_2) W_s\|_F^2, \quad (13)$$

where $W_d = \text{diag}\{w_p, w_p, w_r, w_r\}$ and $W_s = \text{diag}\{w_p, w_r\}$ are weighting matrices. The scalars w_p and w_r are the weights for the positions and residues of the poles, respectively. Now we give the motivation of the optimization problem. Assume that all poles are simple and the perturbation is sufficiently small. If all poles are correctly matched, the objective function is very small; otherwise, the objective function must be much larger.

B. A Branch and Bound Algorithm

To solve the combinatorial optimization problem (13), the exhaustive search method requires $n_d! \times n_s!$ enumerations, which is computationally feasible only for very small ROMs. In this section, we discuss efficient solution of (13).

First, we observe that the optimization problem (13) can be decoupled into two independent optimization problems:

$$\min_{v_d} f_d(v_d) = \|D_1 W_d - \mathcal{M}(v_d, D_2) W_d\|_F^2 \quad (14)$$

and

$$\min_{v_s} f_s(v_s) = \|S_1 W_s - \mathcal{M}(v_s, S_2) W_s\|_F^2. \quad (15)$$

Their solutions v_d^* and v_s^* solve the original optimization problem (13):

$$\min_{v_d, v_s} f(v_d, v_s) = \min_{v_d} f_d(v_d) + \min_{v_s} f_s(v_s), \quad (16)$$

$$\arg \min_{v_d, v_s} f(v_d, v_s) = \left(\arg \min_{v_d} f_d(v_d), \arg \min_{v_s} f_s(v_s) \right). \quad (17)$$

Using this method alone, we reduce the computational cost of (13) from $n_d! \times n_s!$ enumerations to $n_d! + n_s!$ enumerations. In the following discussions, we focus on the solution of (14). The optimization problem (15) can be solved similarly.

The second technique that we use to further reduce the computational cost of the combinatorial optimization problem (13) is the branch and bound method [17]. It can drastically reduce this computational cost in the scenarios that are the most useful in pole-matching, i.e., Scenario 1 and Scenario 2 on page 5. The efficiency of the branch and bound method for pole matching relies on the following theorem.

Theorem III.1. *Assume that $1 \leq i, j \leq n_d$ are two given indices ($i \neq j$). If there exist two mapping vectors $v_d^{(1)}$ and $v_d^{(2)}$ satisfying (9) and*

$$v_{d,i}^{(1)} = v_{d,j}^{(2)}, \quad v_{d,j}^{(1)} = v_{d,i}^{(2)}, \quad (18)$$

$$v_{d,k}^{(1)} = v_{d,k}^{(2)}, \quad (\forall 1 \leq k \leq n_d, k \neq i \text{ and } k \neq j) \quad (19)$$

$$f_d(v_d^{(2)}) - f_d(v_d^{(1)}) = c > 0, \quad (20)$$

then any mapping vector v_d satisfying (9) and

$$v_{d,i} = v_{d,i}^{(2)} \quad \text{and} \quad v_{d,j} = v_{d,j}^{(2)} \quad (21)$$

cannot be the optimal solution of (14).

Proof. Assume v_d^+ is the optimal solution of (14) satisfying (21). We construct v_d^- as

$$\begin{aligned} v_{d,i}^- &= v_{d,j}^+ = v_{d,i}^{(1)}, & v_{d,j}^- &= v_{d,i}^+ = v_{d,j}^{(1)}, \\ v_{d,k}^- &= v_{d,k}^+, & (\forall 1 \leq k \leq n_d, k \neq i \text{ and } k \neq j). \end{aligned}$$

Then,

$$\begin{aligned} f_d(v_d^+) - f_d(v_d^-) &= \|D_1 W_d - \mathcal{M}(v_d^+, D_2) W_d\|_F^2 - \|D_1 W_d - \mathcal{M}(v_d^-, D_2) W_d\|_F^2 \\ &= \left\| \begin{bmatrix} D_{1,i} \\ D_{1,j} \end{bmatrix} W_d - \begin{bmatrix} D_{2,v_{d,i}^+} \\ D_{2,v_{d,j}^+} \end{bmatrix} W_d \right\|_F^2 - \end{aligned}$$

$$\begin{aligned}
& \left\| \begin{bmatrix} D_{1,i} \\ D_{1,j} \end{bmatrix} W_d - \begin{bmatrix} D_{2,v_d^-,i} \\ D_{2,v_d^-,j} \end{bmatrix} W_d \right\|_F \\
&= \left\| \begin{bmatrix} D_{1,i} \\ D_{1,j} \end{bmatrix} W_d - \begin{bmatrix} D_{2,v_d^{(2)},i} \\ D_{2,v_d^{(2)},j} \end{bmatrix} W_d \right\|_F - \\
& \left\| \begin{bmatrix} D_{1,i} \\ D_{1,j} \end{bmatrix} W_d - \begin{bmatrix} D_{2,v_d^{(1)},i} \\ D_{2,v_d^{(1)},j} \end{bmatrix} W_d \right\|_F \\
&= \left\| D_1 W_d - \mathcal{M}(v_d^{(2)}, D_2) W_d \right\|_F^2 - \\
& \left\| D_1 W_d - \mathcal{M}(v_d^{(1)}, D_2) W_d \right\|_F^2 \\
&= f_d(v_d^{(2)}) - f_d(v_d^{(1)}) = c > 0.
\end{aligned}$$

Therefore,

$$f_d(v_d^-) = f_d(v_d^+) - c < f_d(v_d^+),$$

which contradicts the assumption that v_d^+ is the optimal solution of $\min_{v_d} f_d(v_d)$ defined in (14). \square

Based on Theorem III.1, a branch and bound algorithm is proposed with two motivations to save the computational cost:

- 1) Start at a hopefully good initial guess $v_d^{(0)} = (1, 2, \dots, n_d)$, which is computationally free.
- 2) Try to decrease f_d by swapping any two entries of the current v_d and use the result from Theorem III.1 to branch out the swaps that we are certain to be unable to further decrease f_d .

Remark III.1. *Remarks on the two motivations.*

On Motivation 1. *Although $v_d^{(0)} = (1, 2, \dots, n_d)$ is simple, it is actually often a quite good initial guess under the following assumption:*

Assumption III.2. *Conditions in favor of a good initial guess $v_d^{(0)} = (1, 2, \dots, n_d)$.*

- a) We use the same MOR method to compute the ROMs;
- b) The MOR algorithm normally preserves the order of parametric eigenvalues when $\|p_1 - p_2\|$ is small;
- c) Both ROMs are accurate enough and $\|p_1 - p_2\|$ is small enough.

Under this assumption, it is reasonable to presume that when the parameter is perturbed, 1) The order of the parametric eigenvalues is normally preserved. 2) Even when the order changes, normally only a small number of indices change at the same time.

On Motivation 2. *Since we start from a “hopefully good” initial guess, it is quite likely that we fail to decrease f_d by swapping entries of v_d . This case is actually ideal for us because the failed swap can be branched out and we never need to try it again according to Theorem III.1. If the initial guess happens to be the optimal solution, we actually just need to branch out all possibilities, which only needs $n_d(n_d - 1)$ enumerations instead of $n_d!$ enumerations.*

To keep track of the branched out swaps, we use the matrix F defined as follows:

$$F(i, j) = \begin{cases} 0, & \text{when any } v_d \text{ with } v_{d,i} = v_{d,j}^{(0)} \text{ and} \\ & v_{d,j} = v_{d,i}^{(0)} \text{ has been branched out;} \\ 1, & \text{otherwise.} \end{cases} \quad (22)$$

More specifically, we initially set

$$F = J_{n_d} - I_{n_d}, \quad (23)$$

where $J_{n_d}, I_{n_d} \in \mathbb{R}^{n_d \times n_d}$ are the matrix of ones and the identity matrix, respectively. This means that initially, no swap has been branched out: only swapping an index with itself is forbidden. Note that when we swap the i -th and j -th entries of v_d , we actually swap the $v_{d,i}$ -th and $v_{d,j}$ -th rows of D_2 , or equivalently, swap the $v_{d,i}$ -th and $v_{d,j}$ -th entries of $v_d^{(0)}$.

Algorithm III.1. A Branch and Bound Method to Solve the Pole-Matching Optimization Problem (14)

- 1: Initialize $v_d = (1, 2, \dots, n_d)$, the current lowest objective value $b_l = f_d(v_d)$, and $F = J_{n_d} - I_{n_d}$.
- 2: **while** $\|F\|_F \neq 0$ **do**
- 3: **for** $i = 1$ **to** n_d **do**
- 4: **for** $j = 1$ **to** n_d **do**
- 5: **if** $F(v_{d,i}, v_{d,j}) = 0$ **then**
- 6: **continue** (go to line 4)
- 7: **else**
- 8: Set $v_d^* = v_d$.
- 9: Swap the i -th and the j -th entry of v_d^* .
- 10: **if** $f(v_d^*, v_d^{(0)}) \geq b_l$ **then**
- 11: Update $F(v_{d,i}, v_{d,j}) = 0$.
- 12: **else**
- 13: Set $v_d = v_d^*$ and $b_l = f_d(v_d^*)$, and break both for loops (go to line 2).
- 14: **end if**
- 15: **end if**
- 16: **end for**
- 17: **end for**
- 18: **end while**

Now we discuss the definition (22). Assume that we are at v_d^C and we have failed to increase f_d by swapping its i -th entry with its j -th entry. Denote the swapped vector by v_d^S . Then v_d^C and v_d^S satisfy the conditions (18), (19) and (20). Therefore, according to Theorem III.1, any v_d with $v_{d,i} = v_{d,i}^S = v_{d,j}^C$ and $v_{d,j} = v_{d,j}^S = v_{d,i}^C$ cannot be the solution of the optimization problem (14). Therefore, we set $F(v_{d,i}^C, v_{d,j}^C) = 0$ to forbid this swap, i.e., if the k -th and the l -th entries of any v_d satisfy

$$v_{d,k} = v_{d,i}^C, \quad v_{d,l} = v_{d,j}^C, \quad (24)$$

we do not need to try the swap of the k -th and l -th entries of v_d since it cannot decrease the objective f_d . Note that the matrix F is normally non-symmetric because if $F(v_{d,j}, v_{d,i}) = 0$, we forbid swapping the k -th and the l -th entries if $v_{d,k} = v_{d,j}^C$ and $v_{d,l} = v_{d,i}^C$, which is actually a different swap.

As for implementation, we can also use the more compact “potentially feasible tables” $\mathcal{F}_1, \dots, \mathcal{F}_{v_d}$ as we did in [14] instead of using the matrix F , especially when n_d is large. The potentially feasible table is defined as

$$\mathcal{F}_i = \{j | F(i, j) = 1\}. \quad (25)$$

So when a swap is branched out, it is simply removed from the potentially feasible table.

Now we discuss several scenarios in pole matching.

Scenario 1. All poles naturally match: the most optimistic scenario. This case occurs quite often under Assumption III.2 as we have discussed there. Using the branch and bound algorithm, we just need to verify that the initial $v_d^{(0)}$ is optimal and we only need to enumerate the v_d vector $n_d(n_d - 1)$ times instead of $n_d!$ times.

Scenario 2. All poles do not naturally match, but only a few swapping operations, say, m times, are required to obtain the optimal solution. In general, this scenario is unavoidable. e.g., when the dominances² of two poles cross. The computational cost in this scenario is bounded by $mn_d(n_d - 1)$ enumerations. The actual computational cost is normally much lower because many swaps have been branched out at an early stage.

Scenario 3. Almost all poles between (D_1, S_1) and (D_2, S_2) do not naturally match. In this scenario, the computational cost of optimization is high. Actually, if both ROMs are generated by the same MOR method with some “algorithmic continuity”, it is not so meaningful to conduct optimization, whose solution can be misleading. In such cases, it is highly possible that the “perturbation” is too large, except for some rare cases, e.g., the dominances of many pair of poles do cross at the same parameter value. Therefore, we avoid solving the optimization problem in this case, which is not only computationally expensive, but also useless. This is a motivation of the predictor-corrector method that will be presented in Section IV-A.

IV. THE ADAPTIVE FRAMEWORK TO BUILD THE PARAMETRIC ROM

The goal of this section is to adaptively build a series of repository ROMs, which will be used as interpolates for constructing the pROM. In this section, we confine our discussion to parametric systems with a single parameter $p \in \mathbb{R}$. We denote these repository ROMs, whose poles have already been matched to those of all other repository ROMs, by $\overline{\text{ROM}}_1, \overline{\text{ROM}}_2, \dots, \overline{\text{ROM}}_N$ built for the parameter values $p_1 < p_2 < \dots < p_N$, respectively. In this paper, the overline symbol $\overline{\cdot}$ indicates the name or variables of a repository ROM, e.g., $\overline{\text{ROM}}_i$ with its pole-residue realization $(\overline{D}_i, \overline{S}_i)$.

¹We use the term “potentially feasible” because the swaps recorded in this table might not be feasible, but we have not validated their feasibility yet.

²In this paper, we use the term “pole dominance” to denote the importance of a pole on the system output. We use it as a general term for all kinds of MOR methods. This term is borrowed from a specific case: the dominant pole algorithm [18], [4], which defines the dominance of a pole of the form

$\frac{\overline{C}_j^{(i)}}{s - \lambda_j^{(i)}}$ by $\frac{|\overline{C}_j^{(i)}|}{|\Re\{\lambda_j^{(i)}\}|}$ for this specific case.

For ease of notation, we introduce a mapping for pole-matching $\mathcal{P}_{D_1}(D_2)$ ($\mathcal{P}_{S_1}(S_2)$). When we match D_2 to D_1 , we denote the matching result of Algorithm III.1 by $\mathcal{P}_{D_1}(D_2)$. Using this notation, we define the distance between $\text{ROM}_1(D_1, S_1)$ and $\text{ROM}_2(D_2, S_2)$ as

$$r(\text{ROM}_1, \text{ROM}_2) = \left\| D_1 W_d - \mathcal{P}_{D_1}(D_2) W_d \right\|_F + \left\| S_1 W_s - \mathcal{P}_{S_1}(S_2) W_s \right\|_F \quad (26)$$

and the relative error by

$$e(\text{ROM}_1, \text{ROM}_2) = \frac{\left\| D_1 W_d - \mathcal{P}_{D_1}(D_2) W_d \right\|_F}{\left\| D_1 \right\|_F} + \frac{\left\| S_1 W_s - \mathcal{P}_{S_1}(S_2) W_s \right\|_F}{\left\| S_1 \right\|_F}. \quad (27)$$

For convenience, we denote the pole-matched ROM ($\mathcal{P}_{D_1}(D_2), \mathcal{P}_{S_1}(S_2)$) by $\mathcal{P}_{\text{ROM}_1}(\text{ROM}_2)$.

A. The Predictor-Corrector Strategy

The two motivations of the predictor-corrector strategy are:

Objective 1. We want to design a framework, in which Scenario 1 occurs most often, Scenario 2 occurs sometimes, and Scenario 3 occurs rarely.

Objective 2. However, we do not want to achieve Objective 1 simply by using small steps in p , which is inefficient.

In order to explore the parameter space adaptively, we make the following assumptions on the ROMs.

Assumption IV.1. *At any feasible parameter value p , a ROM of high accuracy can be built at request.*

Assumption IV.2. *The changes in positions and residues of all poles are small when the change of p is sufficiently small.*

Suppose that we have already computed the repository ROMs: $\overline{\text{ROM}}_1, \overline{\text{ROM}}_2, \dots, \overline{\text{ROM}}_i$ and we want to expand the series by adding new repository ROMs. At a new parameter value p_{i+1} ($p_{i+1} > p_i$), we first compute a candidate $\text{ROM}_{i+1}^{\text{cand}}$ and try to match its poles to the repository ROMs. Instead of applying Algorithm III.1 directly to ROM_i and $\text{ROM}_{i+1}^{\text{cand}}$, we propose to use a predictor-corrector strategy.

We first use the positions and residues of poles of $\overline{\text{ROM}}_1, \overline{\text{ROM}}_2, \dots, \overline{\text{ROM}}_i$ to calculate a “predicted ROM” $\text{ROM}_{i+1}^{\text{pred}}$ for p_{i+1} . For prediction, we can use extrapolation, e.g., linear, polynomial, spline, etc., on the positions and residues of the poles. If the calculated poles exhibit some noise, we can also use regression. Normally, the poles of $\text{ROM}_{i+1}^{\text{pred}}$ are closer to the poles of $\text{ROM}_{i+1}^{\text{cand}}$ than those of $\overline{\text{ROM}}_i$. To be safe, we check whether $r(\overline{\text{ROM}}_i, \text{ROM}_{i+1}^{\text{cand}}) > r(\text{ROM}_{i+1}^{\text{pred}}, \text{ROM}_{i+1}^{\text{cand}})$ holds. If it holds, we set $\overline{\text{ROM}}_{i+1} = \mathcal{P}_{\overline{\text{ROM}}_i}(\text{ROM}_{i+1}^{\text{cand}})$; otherwise, we set $\overline{\text{ROM}}_{i+1} = \mathcal{P}_{\text{ROM}_{i+1}^{\text{pred}}}(\text{ROM}_{i+1}^{\text{cand}})$.

Note that although we have “accepted” the current $\overline{\text{ROM}}_{i+1}$ as a repository ROM, it is not guaranteed yet that the interpolation will give accurate pMOR on the interval (p_i, p_{i+1}) . Actually, the pole-matching is not necessarily correct because Assumption III.1 may not be satisfied. However, even if the poles are wrongly matched, we store it for further use because

we do not want to waste the computational effort with which we built $\overline{\text{ROM}}_{i+1}$. Therefore, we introduce an index i_h to denote the last index that we are sure to have “high fidelity”, i.e., it is correctly matched and close enough to $\overline{\text{ROM}}_{i_h-1}$ to produce good interpolation results. Here we assign $i_h = i$, which means that the fidelity of $\overline{\text{ROM}}_{i+1}$ still needs to be checked. More details will be discussed in the next section.

B. An Adaptive Refinement Strategy

In the previous section, we have discussed the efficient generation and pole-matching of ROMs. However, this alone does not guarantee good accuracy of the pROM: it is possible that the poles are matched correctly, but the interpolation result does not capture the parametric evolution of the poles well. For example, for a system that has only one pole, although the pole is naturally “matched”, we still need a fine grid of p to capture the dynamics of the system when the dynamics is complex.

After we have computed $\overline{\text{ROM}}_{i+1}$ using the predictor-corrector strategy, we need to assess whether good accuracy can be achieved in the interval (p_i, p_{i+1}) using interpolation. To achieve this, we choose the test point $p^t = \frac{p_i + p_{i+1}}{2}$ and build two ROMs at p^t :

- 1) $\text{ROM}^{I,t}$ computed by interpolating the repository ROMs.
- 2) $\text{ROM}^{T,t}$ computed by a MOR algorithm.

Since under Assumption IV.1, $\text{ROM}^{T,t}$ is highly accurate, we regards $\text{ROM}^{I,t}$ as accurate only when

$$e(\text{ROM}^{I,t}, \text{ROM}^{T,t}) < \tau_e, \quad (28)$$

where τ_e is a small real number, e.g., 0.001. The value of τ_e should take the error of the MOR algorithm into consideration: τ_e should be sufficiently larger than an estimated error of the MOR algorithm. Otherwise, the relationship (28) may never be satisfied despite of refinement of the grid. In practical computations, we can actually change τ_e adaptively according to an error bound of $\text{ROM}^{T,t}$ rather than using the same τ_e for all iterations.

If (28) is met, we trust the fidelity of $\overline{\text{ROM}}_{i+1}$ and use the predictor-corrector for the next step. Otherwise, we need more ROMs to represent the interval (p_i, p_{i+1}) . A natural choice is $\text{ROM}^{T,t}$ since it has already been built. Therefore, we assign

$$\overline{\text{ROM}}_{i+2} \leftarrow \overline{\text{ROM}}_{i+1}, \quad \overline{\text{ROM}}_{i+1} \leftarrow \mathcal{P}_{\overline{\text{ROM}}_i}(\text{ROM}^{T,t}). \quad (29)$$

Then, we check the fidelity of the new $\overline{\text{ROM}}_{i+1}$ using the procedure described above:

- If (28) is violated, we do a further refinement for the interval (p_i, p_{i+1}) .
- Otherwise, we accept that $\overline{\text{ROM}}_{i+1}$ has high fidelity, update $i_h = i + 1$ and

$$\overline{\text{ROM}}_{i+2} \leftarrow \mathcal{P}_{\overline{\text{ROM}}_{i+1}}(\overline{\text{ROM}}_{i+2}) \quad (30)$$

because p_{i+1} (the former p^t) is closer to p_{i+2} (the former p_{i+1}) and the pole-matching result is more trustworthy. Next, we check whether $\overline{\text{ROM}}_{i+2}$ has high fidelity:

- If it has high fidelity, we set $i_h = i + 2$ and end this local refinement.

- Otherwise, we do a further refinement for the interval (p_{i+1}, p_{i+2}) using the procedure above.

For the whole procedure, we refer to Algorithm IV.1.

C. Compact Parametric ROM by Regression

After we have finished the construction of $\overline{\text{ROM}}_1, \overline{\text{ROM}}_2, \dots, \overline{\text{ROM}}_N$ for the whole interval of interest, we can further save data storage by using regression. In this paper, we do regression for each entry of $\overline{D}_i(j, k)$ (the (j, k) -th entry of \overline{D}_i built at p_i) and $\overline{S}_i(j, k)$ to obtain the parametric forms $\overline{D}(p; j, k)$ and $\overline{S}(p; j, k)$. Using a polynomial of degree q for each entry, we reduce the storage cost from $N(4n_d + 2n_s)$ to $(q+1)(4n_d + 2n_s)$ since we only need to store the coefficients of the polynomials. This reduction is usually drastic since a low-order polynomial is normally enough when the system dynamics is not too complex.

We summarize all the techniques discussed in Algorithm IV.1 with more details.

Algorithm IV.1. An Adaptive Pole-Matching PMOR method

- 1: **Input:** the parameter range of interest $[p^L, p^U]$.
- 2: **Initialization:** initialize step length u_0 , error tolerance τ_e , order of regression polynomials q , $i = 2$, $p_1 = p^L$, and compute $\overline{\text{ROM}}_1$.
- 3: **repeat**
- 4: **Phase 1: Predictor-Corrector**
- 5: Set $p_i = p_{i-1} + u_0$. If $p_i > p^U$, set $p_i = p^U$.
- 6: Build $\text{ROM}_i^{\text{cand}}$ with a MOR algorithm.
- 7: **if** $i=2$ **then**
- 8: $\overline{\text{ROM}}_i = \mathcal{P}_{\overline{\text{ROM}}_{i-1}}(\text{ROM}_i^{\text{cand}})$.
- 9: **else**
- 10: Compute $\text{ROM}_i^{\text{pred}}$ using extrapolation or regression of $\overline{\text{ROM}}_j$ ($j = 1, 2, \dots, i-1$).
- 11: **if** $r(\overline{\text{ROM}}_{i-1}, \text{ROM}_i^{\text{cand}}) > r(\text{ROM}_i^{\text{pred}}, \text{ROM}_i^{\text{cand}})$ **then**
- 12: $\overline{\text{ROM}}_i = \mathcal{P}_{\text{ROM}_i^{\text{pred}}}(\text{ROM}_i^{\text{cand}})$.
- 13: **else**
- 14: $\overline{\text{ROM}}_i = \mathcal{P}_{\overline{\text{ROM}}_{i-1}}(\text{ROM}_i^{\text{cand}})$.
- 15: **end if**
- 16: **end if**
- 17: **Phase 2: Adaptive Refinement**
- 18: Set $i_h = i - 1$.
- 19: **repeat**
- 20: At $p^t = \frac{p_{i_h} + p_{i_h+1}}{2}$, build an interpolated ROM, namely $\text{ROM}^{I,t}$, and a true ROM, namely $\text{ROM}^{T,t}$.
- 21: **if** $e(\text{ROM}^{I,t}, \text{ROM}^{T,t}) < \tau_e$ **then**
- 22: $i_h = i_h + 1$.
- 23: **else**
- 24: $i = i + 1$.
- 25: **for** $j = i - 1$ **to** $i_h + 1$ **do**
- 26: $\overline{\text{ROM}}_{j+1} \leftarrow \overline{\text{ROM}}_j, p_{j+1} \leftarrow p_j$.
- 27: **end for**
- 28: $\overline{\text{ROM}}_{i_h+1} = \mathcal{P}_{\overline{\text{ROM}}_{i_h}}(\text{ROM}^{T,t})$.
- 29: **for** $j = i - 1$ **to** $i_h + 1$ **do**
- 30: $\overline{\text{ROM}}_{j+1} \leftarrow \mathcal{P}_{\overline{\text{ROM}}_{i_h+1}}(\overline{\text{ROM}}_{j+1})$.
- 31: **end for**
- 32: **end if**

```

33: until  $i_h = i$ .
34: until  $p_i = p^U$ .
35: Optional Post-Processing: Regression
36: for  $u = 1$  to  $n_d$ ,  $v = 1$  to 4 do
37:   Apply the regression algorithm to  $\overline{D}_1(u, v)$ ,  $\overline{D}_2(u, v)$ ,
    $\dots$ ,  $\overline{D}_i(u, v)$  to compute the coefficients of the polynomial
    $d_0^r(u, v)$ ,  $d_1^r(u, v)$ ,  $\dots$ ,  $d_q^r(u, v)$ .
38: end for
39: for  $u = 1$  to  $n_s$ ,  $v = 1$  to 2 do
40:   Apply the regression algorithm to  $\overline{S}_1(u, v)$ ,  $\overline{S}_2(u, v)$ ,
    $\dots$ ,  $\overline{S}_i(u, v)$  to compute the coefficients of the polynomial
    $s_0^r(u, v)$ ,  $s_1^r(u, v)$ ,  $\dots$ ,  $s_q^r(u, v)$ .
41: end for

```

Note that when pole-matching becomes difficult, i.e., too many swaps take place, we can simply give it up, save the ROM (with pole unmatched) for the moment and insert another ROM for refinement. The pole-matching will be done automatically at a later stage. A further remark is that in line 29, we can also conduct fewer pole-matchings, e.g., only for $i_h + 1$.

D. The Offline-Online Strategy

The proposed PMOR method can be implemented in an offline-online manner.

- In the offline phase, we use Algorithm IV.1 to explore the parameter range of interest. It is relatively expensive: we need to adaptively choose the parameter values, at which we build ROMs, compute the ROMs, conduct pole matching, and optionally, perform regression.
- In the online phase, we simply interpolate the repository ROMs or evaluate the polynomials computed from regression. The online phase is computationally very cheap.

V. NUMERICAL RESULTS

In this section, we use two numerical examples to test the adaptive pole-matching PMOR method.

A. The Nonlinear Parametric “FOM” Model

This academic example is adapted from the parametric “FOM” model in [19] to introduce more complex system dynamics such as nonlinear parametric dependency and pole crossing, since in the original model, the parameter dependency is linear³. The model is of the form

$$\begin{aligned}
 (s\mathcal{I} - A(p))X(s, U) &= \mathcal{B}U, \\
 Y &= \mathcal{C}X(s, U),
 \end{aligned}$$

where $\mathcal{C} = [100, 100, 100, 100, 100, 100, 100, 100, 1, \dots, 1] \in \mathbb{R}^{1 \times 1008}$, $\mathcal{B} = \mathcal{C}^T$, and $\mathcal{A} = \text{diag}\{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, 1, 2, 3, \dots, 1000\} \in \mathbb{R}^{1008 \times 1008}$ with

$$\begin{aligned}
 \mathcal{A}_1 &= \begin{bmatrix} 4p - 42 & 8p + 200 \\ -8p - 200 & 4p - 42 \end{bmatrix}, \\
 \mathcal{A}_2 &= \begin{bmatrix} 2p - 50 & p^2 + 4p + 210 \\ -p^2 - 4p - 210 & 2p - 50 \end{bmatrix},
 \end{aligned}$$

³ The parametric “FOM” model in [19] is adapted from the nonparametric “FOM” model in [20]

$$\begin{aligned}
 \mathcal{A}_3 &= \begin{bmatrix} -25 + p & 100 + p^2 \\ -100 - p^2 & -25 + p \end{bmatrix}, \\
 \mathcal{A}_4 &= \begin{bmatrix} -25 + 2p & 150 - p^2 \\ -150 + p^2 & -25 + 2p \end{bmatrix}.
 \end{aligned}$$

In this model of order 1008, the parameter range of interest is $[-10, 10]$. In Algorithm IV.1, we set the initial conditions $u_0 = \frac{\pi}{3}$, $\tau_e = 0.001$ and $q = 5$. For all figures, the relative error is computed as $\frac{|\int_1^{1000} \mathcal{H}(\omega, p) - H(\omega, p) d\omega|}{|\int_1^{1000} \mathcal{H}(\omega, p) d\omega|}$.

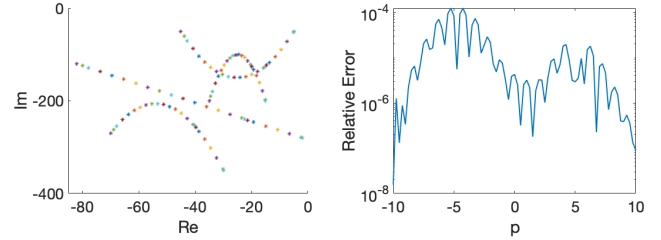


Fig. 1. Interpolation Results

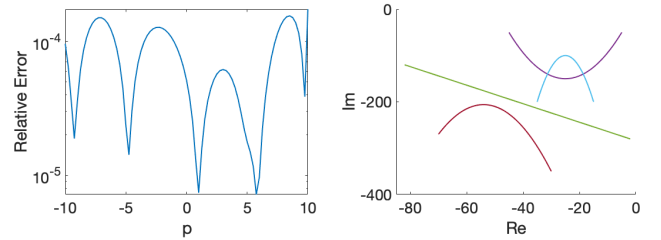


Fig. 2. Regression Results

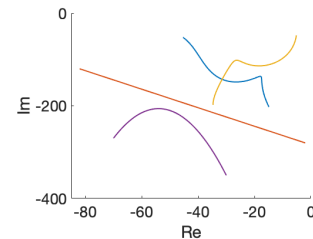
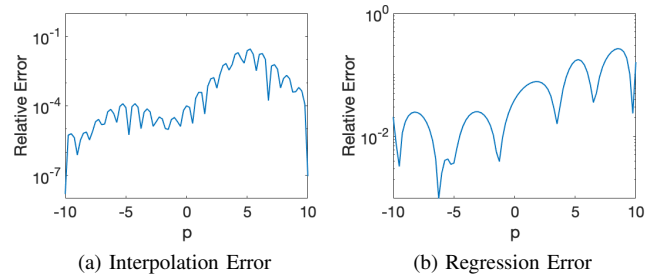


Fig. 3. Results without Adaptive Refinement

In Fig. 1, we show the results of interpolation, i.e., when we skip the post-processing for regression and use the repository

ROMs directly for interpolation. In total, 24 repository ROMs are built. In Fig. 2, we show the numerical results when the post-processing for regression is also performed. We observe that using order-5 polynomials for regression results in pretty accurate results. The data storage is reduced by a factor of $\frac{24}{5}$. In Fig. 3, we plot the results when we skip the adaptive refinement. In this case, 21 repository ROMs are built and we observe that some poles are wrongly matched and the regression follows wrong branches. This shows the importance of the adaptive refinement in guaranteeing accuracy, especially when the step length is large. The much better numerical result using adaptive refinement is only at the cost of 3 more repository ROMs.

B. The Branchline Coupler: a Microwave Device

In this section, we apply our method to a branchline coupler model, which is a discretization of a time-harmonic Maxwell's equation. The full-order model is of the form

$$\begin{aligned} (\mathcal{K}(\mu) - s\mathcal{M})X &= B, \\ Y &= CX, \end{aligned} \quad (31)$$

where $\mathcal{K}(\mu) = \frac{1}{\mu}\mathcal{K}_0$, $\mathcal{K}_0, \mathcal{M} \in \mathbb{R}^{33051 \times 33051}$, $B \in \mathbb{R}^{33051 \times 1}$, $C \in \mathbb{R}^{1 \times 33051}$ and $s = \omega^2$. For detailed description of the model, we refer to [21], [22]. The repository ROMs of order 10 are built by a Krylov method [23], in which we use the interpolating expansion points $s = 1, 25, 50, 75, 100$ and order 2 for each of these interpolating expansion points. Since the reduced matrix of $\mathcal{K}(\mu_i)$ is nonsingular, the repository ROMs can be easily written into the form (1). To build a pROM for the parameter range $\mu \in [0.5, 0.9]$, Algorithm IV.1 built 18 repository ROMs under the tolerance $\tau_e = 0.00001$. Figure 4 shows that both the interpolated pROM and the regressed pROM capture the dynamics of the system well.

VI. FURTHER DISCUSSIONS

Here are some further discussions.

- The regression is successful only when all matchings are correct for dominant poles. The influence of mismatched poles on the interpolated pROM is only local, which is normally not too bad because the wrongly matched poles normally have similar positions and residues. For regression, however, the influence will be global because wrong branches are chosen. In this case, we actually try to regress a non-smooth function with polynomials, which is difficult. This phenomenon is shown in Fig. 3.
- We would also like to emphasize that the interpolation approach is more general than the regression approach. The interpolation approach works well as long as the parameter dependence can be *locally* captured by the basis functions, e.g., using polynomials for exponential functions or trigonometric functions. The regression method, on the other hand, works well only for the cases that the parameter dependence can be *globally* captured by the basis functions. The interpolation approach even has potential to deal with eigenvalue bifurcation: even without any further considerations, the interpolation-based pROM

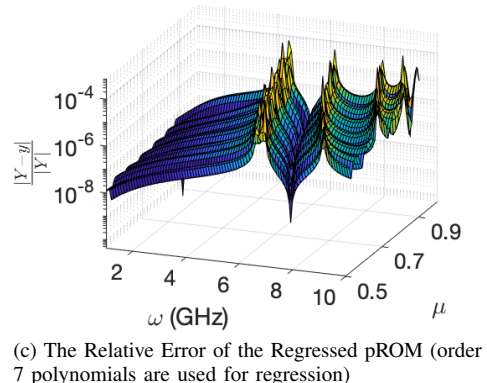
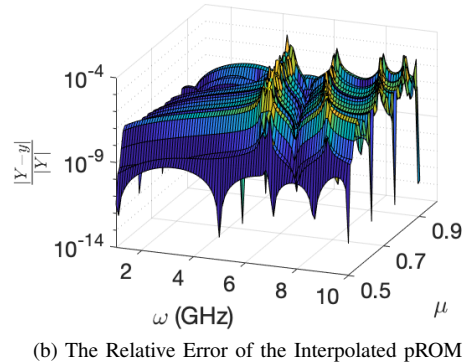
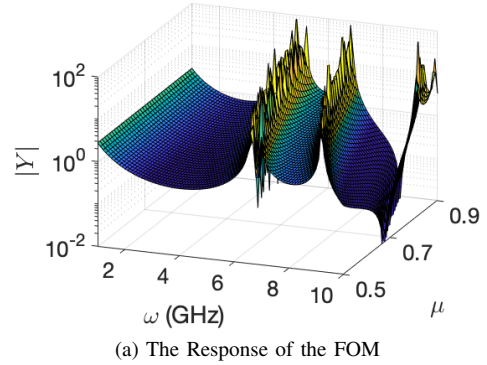


Fig. 4. Numerical Results for the Branchline Example. Here, Y and y stand for the outputs of the FOM and the pROM, respectively.

is accurate as long as the parameter value is not too close to the bifurcation point. More detailed treatments of eigenvalue bifurcation will be future work. Therefore, we use regression as an optional post-processing technique. When the error between the interpolation-based pROM and the regression-based pROM is not sufficiently small, we discard the regression-based pROM and just use the interpolation-based pROM.

- The proposed method only works when Assumption IV.2 holds. When $A^{(i)}$ in (1) is non-normal, its poles can be highly sensitive to perturbations of the parameters. We observed this phenomenon when we build order-22 Krylov-type ROMs for the branchline coupler model.
- The proposed method can be easily extended to multiple-input multiple-output systems using the results in [9].

- Stability can easily be preserved since our method deals directly with the poles. It is apparent that using linear interpolation, the stability will automatically be preserved since the linear interpolation of two poles in the left half-plane still lies in the left half-plane. Using other interpolation methods, although the stability is generally not automatically preserved, we can check whether the maximal real part of the interpolated (regressed) pole location is negative. If it is not, we can simply try another interpolation (regression) method and check it again. In the worst case, we can resort to linear interpolation for the parameter interval on which stability is difficult to be preserved with other interpolation/regression methods.

VII. CONCLUSIONS

An adaptive parametric reduced-order modeling method based on interpolation/regression of adaptively built ROMs is proposed. For correct interpolation/regression, we convert all ROMs to the pole-residue realization and propose to use a branch and bound method to conduct pole-matching efficiently. To explore the parameter range efficiently and accurately, we propose to use a predictor-corrector technique and an adaptive refinement strategy. A regression method is proposed to further reduce data storage. The numerical results verify the high accuracy and robustness of the proposed method.

REFERENCES

- [1] P. Benner, S. Gugercin, and K. Willcox, "A survey of model reduction methods for parametric systems," *SIAM Review*, vol. 57, no. 4, pp. 483–531, 2015.
- [2] L. Feng, Y. Yue, N. Banagaaya, P. Meuris, W. Schoenmaker, and P. Benner, "Parametric modeling and model order reduction for (electro-)thermal analysis of nanoelectronic structures," *J. Math. Ind.*, vol. 6, no. 1, pp. 1–10, 2016.
- [3] Y. Yue and K. Meerbergen, "Using Krylov-Padé model order reduction for accelerating design optimization of structures and vibrations in the frequency domain," *International Journal for Numerical Methods in Engineering*, vol. 90, no. 10, pp. 1207–1232, 2012.
- [4] M. Saadvandi, K. Meerbergen, and W. Desmet, "Parametric dominant pole algorithm for parametric model order reduction," *Journal of Computational and Applied Mathematics*, vol. 259, pp. 259–280, 2014.
- [5] Y. Zhang, L. Feng, S. Li, and P. Benner, "Accelerating PDE constrained optimization by the reduced basis method: application to batch chromatography," *Internat. J. Numer. Methods Engrg.*, vol. 104, no. 11, pp. 983–1007, 2015.
- [6] G. Rozza, D. B. P. Huynh, and A. T. Patera, "Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations," *Archives of Computational Methods in Engineering*, vol. 15, no. 3, pp. 229–275, 2008.
- [7] U. Baur, C. A. Beattie, P. Benner, and S. Gugercin, "Interpolatory projection methods for parameterized model reduction," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2489–2518, 2011.
- [8] D. Amsallem and C. Farhat, "Interpolation method for the adaptation of reduced-order models to parameter changes and its application to aeroelasticity," *AIAA J.*, vol. 46, pp. 1803–1813, 2008.
- [9] Y. Yue, L. Feng, and P. Benner, "Reduced-order modelling of parametric systems via interpolation of heterogeneous surrogates," *Advanced Modeling and Simulation in Engineering Sciences*, pp. 1–28, 2019, to appear.
- [10] H. Panzer, J. Mohring, R. Eid, and B. Lohmann, "Parametric model order reduction by matrix interpolation," *at-Automatisierungstechnik*, vol. 58, no. 8, pp. 475–484, 2010.
- [11] D. Amsallem and C. Farhat, "An online method for interpolating linear parametric reduced-order models," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2169–2198, 2011.
- [12] U. Baur and P. Benner, "Modellreduktion für parametrisierte Systeme durch balanciertes Abschneiden und Interpolation (Model Reduction for Parametric Systems Using Balanced Truncation and Interpolation)," *at-Automatisierungstechnik*, vol. 57, no. 8, pp. 411–420, 2009.
- [13] U. Baur, P. Benner, A. Greiner, J. G. Korvink, J. Lienemann, and C. Moosmann, "Parameter preserving model reduction for MEMS applications," *Math. Comput. Model. Dyn. Syst.*, vol. 17, no. 4, pp. 297–317, 2011.
- [14] Y. Yue, L. Feng, and P. Benner, "Interpolation of reduced-order models based on modal analysis," in *2018 IEEE MTT-S International Conference on Numerical Electromagnetic and Metaphysics Modeling and Optimization (NEMO)*, 2018.
- [15] —, "An adaptive method for interpolating reduced-order models based on matching and continuation of poles," in *2019 IEEE MTT-S International Conference on Numerical Electromagnetic and Metaphysics Modeling and Optimization (NEMO)*, 2019.
- [16] G. H. Golub and C. F. van Van Loan, *Matrix Computations*, 3rd ed. Baltimore and London: The Johns Hopkins University Press, 1996.
- [17] J. Clausen, "Branch and bound algorithms — principles and examples," University of Copenhagen, Tech. Rep., 1999.
- [18] N. Martins, L. T. G. Lima, and H. J. C. P. Pinto, "Computing dominant poles of power system transfer functions," *IEEE Trans. Power Syst.*, vol. 11, no. 1, pp. 162–170, 1996.
- [19] A. C. Ionita and A. C. Antoulas, "Data-driven parametrized model reduction in the Loewner framework," *SIAM J. Sci. Comput.*, vol. 36, no. 3, pp. A984–A1007, 2014.
- [20] Y. Chahlaoui and P. Van Dooren, "A collection of benchmark examples for model reduction of linear time invariant dynamical systems," SLICOT Working Note, Tech. Rep. 2002–2, 2002, available from www.slicot.org.
- [21] The MORwiki Community, "Branchline coupler," MORwiki – Model Order Reduction Wiki, 2013. [Online]. Available: http://modelreduction.org/index.php/Branchline_Coupler
- [22] M. W. Hess and P. Benner, "Fast evaluation of time-harmonic Maxwell's equations using the reduced basis method," *IEEE Trans. Microw. Theory Techn.*, vol. 61, no. 6, pp. 2265–2274, 2013.
- [23] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, ser. Advances in Design and Control. Philadelphia, PA: SIAM Publications, 2005, vol. 6.