Yue *et al. Adv. Model. and Simul. in Eng. Sci.(2019)6:10*
https://doi.org/10.1186/s40323-019-0134-y

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

Reduced-order modelling of parametric systems via interpolation of heterogeneous surrogates

Yao Yue*, Lihong Feng and Peter Benner

*Correspondence:
yue@mpi-magdeburg.mpg.de
Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstrasse 1, 39108 Magdeburg, Germany

Abstract

This paper studies parametric reduced-order modeling via the interpolation of linear multiple-input multiple-output reduced-order, or, more general, surrogate models in the frequency domain. It shows that realization plays a central role and two methods based on different realizations are proposed. Interpolation of reduced-order models in the Loewner representation is equivalent to interpolating the corresponding frequency response functions. Interpolation of reduced-order models in the (real) pole-residue representation is equivalent to interpolating the positions and residues of the poles. The latter pole-matching approach proves to be more natural in approximating system dynamics. Numerical results demonstrate the high efficiency and wide applicability of the pole-matching method. It is shown to be efficient in interpolating surrogate models built by several different methods, including the balanced truncation method, the Krylov method, the Loewner framework, and a system identification method. It is even capable of interpolating a reduced-order model built by a projection-based method and a reduced-order model built by a data-driven method. Its other merits include low computational cost, small size of the parametric reduced-order model, relative insensitivity to the dimension of the parameter space, and capability of dealing with complicated parameter dependence.

Keywords: Parametric model order reduction, Interpolation methods, Data-driven methods, Pole analysis

Introduction

Model order reduction (MOR), as a flourishing computational technique to accelerate simulation-based system analysis in the last decades, has been applied successfully to high-dimensional models arising from various fields such as circuit simulations [1–3], (vibro) acoustics [4], design of microelectromechanical systems [5], and chromatography in chemical engineering [6]. The objective of MOR is to compute a reduced-order model (ROM) of small size k that captures some important characteristics of the original high-dimensional model of size n (normally $k \ll n$), such as dominant moments and leading Hankel singular values. When parametric studies are performed, it is desired to build a parametric ROM (pROM), which not only incorporates the modeling parameters as free parameters, but also approximates the input–output behavior of the full-order model

© The Author(s) 2019. This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Springer Open

(FOM) well for any parameter value within the domain of interest. Therefore, many parametric MOR (PMOR) methods have been proposed; see, e.g., the recent survey [7].

In general, PMOR methods can be categorized into two general types according to PMOR review paper [7]:

1. Building a single pROM by using a global basis over the parameter space. These methods [8] have received intensive research attention and proved to be efficient in many applications in the past few years, most notably the reduced basis method [9]. Despite its success in many application fields, these methods normally meet difficulty in dealing with many parameters since both the computational cost and the size of the basis matrices may grow exponentially with the dimension of the parameter space because of the curse of dimensionality.
2. Building an interpolatory pROM by interpolating local matrices or bases at parameter samples, e.g., each of which is obtained by applying nonparametric MOR at the corresponding parameters. These methods are less studied and their performance is not so satisfactory. According to [7], this approach can be again categorized into three types:

    a. Interpolation among local basis matrices [10]. Since the basis matrices $V_i \in \mathbb{R}^{n\times k}$ are elements on the Stiefel manifold, this method interpolates V_i along the geodesic on the Stiefel manifold with the help of the tangent space. However, assuming that the basis matrix evolves on the geodesic is only heuristic: it is only one of the infinite possible evolution paths on the manifold. According to our experience in numerical tests, the resulting ROM often diverges. Another disadvantage is that, this method requires the storage of all the basis matrices, which may not only be expensive, but also infeasible in many cases. First, when the ROMs are built by a non-projection-based MOR method, e.g., data-driven MOR methods or MOR methods based on physical reasoning, no bases are computed. In addition, in many practical applications, the parametric FOM is not available and what we can obtain are nonparametric FOMs built for some parameter samples, which may cause difficulty for these methods. For example, these FOMs may be obtained from discretization of partial differential equations with different meshes, may also be of different dimensions, and their realizations may be not consistent.
    b. Interpolation among local reduced-order state-space matrices [11,12]. As is shown in [11], directly interpolating the local reduced-order state-space matrices does not work in general. A major difficulty is that, a dynamical system has infinitely many realizations and interpolating between different realizations can result in completely wrong results. For example, a system with the same input–output behavior can be obtained by rearranging the rows of the matrices, but interpolating two such system matrices normally makes no physical sense, e.g., interpolating between $\begin{bmatrix} K(p_1) & C \\ 0 & I \end{bmatrix}$ and $\begin{bmatrix} 0 & I \\ K(p_2) & C \end{bmatrix}$, where $K(p)$, C are square matrices with the same size and I and 0 are corresponding identity and zero matrices, respectively. Therefore, a natural idea is to apply a congruence transformation to obtain consistent bases, i.e., by solving an optimization problem to get the transformation matrices, and then interpolate these consistent ROMs

on some manifold [12]. Another choice is to conduct a singular value decomposition (SVD) on the union of all basis matrices to calculate the dominant "global subspace", onto which we re-project all ROMs and conduct interpolation [11]. However, like the methods discussed in 2(a), the ROMs must be of the same dimension and all bases have to be stored.

c. Interpolation among the local frequency response functions (FRFs) [13,14]. We will show later in the paper that interpolating Loewner ROMs [15] built from the local FRFs is equivalent to interpolating the local FRFs. Therefore, interpolation among the local FRFs can be seen as a special case of interpolation among the local reduced-order state-space matrices. Furthermore, we propose a further technique to compress the Loewner ROMs to save the storage space. Although this method is intuitive and easy to implement, it suffers from the problem of fixed poles: the positions of the poles do not change with the parameter, but are rather determined by the parameter values used for interpolation.

The goal of the present paper is twofold:

- This paper will propose a pole-matching reduced-order modeling method that interpolates linear multiple-input multiple-output (MIMO) ROMs in the frequency domain. Inspired by modal analysis in mechanical engineering [16], the pole-matching method relies completely on analyzing the positions and residues of the poles, rather than trying to recover the state vector of the FOM. We propose to first convert all ROMs to a unified realization, namely the pole-residue realization that stores the positions and residues of poles explicitly. Then, we match the poles according to their positions and residues in order to capture the evolution of the poles in the parametric system. Finally, we interpolate the positions and residues of all matched poles to obtain the parametric ROMs. This method does not require the storage of basis matrices, is capable of interpolating ROMs of different sizes, and works even when a parametric FOM does not exist, e.g., when ROMs are built by a data-driven MOR method or when FOMs at different parameter values result from different discretization methods or different grids. It can also interpolate ROMs of different nature, e.g., interpolating a ROM built by a mathematical MOR method and another ROM obtained from physical reasoning. It is relatively insensitive to the number of parameters and does not assume specific properties of the FOM, e.g., the affinity property required by the reduced basis method [9]. Numerical results show that the pole-matching method is more accurate than the previously proposed methods for our test cases.
- The other goal of this paper is to show the importance of the realization of a dynamical system w.r.t. interpolation. For comparison purposes, we will develop another MIMO PMOR method in the frequency domain, namely the interpolation method for Loewner ROMs. We will show that interpolating the ROMs built by the Loewner framework in the original form is equivalent to interpolating the FRFs. Furthermore, we will also discuss the interpolation of Loewner ROMs in the compressed form, which is more efficient w.r.t. computation and storage. Unlike the pole-matching method, which captures the change of positions and residues of the poles, the interpolation of Loewner ROMs builds parametric ROMs with fixed pole positions. Therefore, using different realizations, the parametric ROMs follow different evolu-

tion paths. Although both methods have clear physical meanings, the interpolation method for Loewner ROMs follows the path that the real-world system is unlikely to take, while the pole-matching method follows a more natural path and provides accurate parametric ROMs for all our test cases.

Let us emphasize here that the proposed method is more a reduced-order parametric modeling approach than a new PMOR method. Our pole-matching approach assumes the availability of locally valid surrogate models of the FOM. These are assumed to be obtained at feasible sampling points in parameter space, but they can result from various surrogate modeling methods, like

- projection-based (or any other computational) MOR methods that compute a non-parametric ROM at the fixed parameter value,
- data-driven approaches like the Loewner framework or dynamic mode decomposition [17],
- system identification methods,
- etc.

We do not even assume that the local surrogates that we interpolate are obtained from the same approach—we can employ a mixture of surrogate models obtained by any of the methods listed above. A particular suitable area of application for our method would be the situation when only an oracle is available, e.g. a running code producing either a state-space model given a fixed parameter or an input–output sequence of time series or frequency-response data. Our approach is fully non-intrusive as it does not require any further knowledge on the system!

Notation

Throughout the paper, $\imath$ is the imaginary unit, M^{T} represents the transpose of the matrix M, and $M_{\beta}^{\alpha,\mathrm{T}}$ denotes $(M_{\beta}^{\alpha})^{\mathrm{T}}$.

Background

In this section, we will briefly review two different types of (P)MOR methods: projection-based methods and the Loewner framework, which is a data-driven (P)MOR method. Though we do not explicitly use any of these methods in our approach, we will frequently refer to them and will also use them for comparison purposes in the numerical examples section. Therefore, we include this brief review for better readability.

Projection-based (P)MOR

This paper focuses on PMOR of state-space systems in the frequency-domain:

$$\begin{aligned}\big(s\mathcal{E}(p)-\mathcal{A}(p)\big)X(s,p)&=\mathcal{B}(p)u(s),\\ Y(s,p)&=\mathcal{C}(p)X(s,p),\end{aligned}\tag{1}$$

where $\mathcal{E}(p), \mathcal{A}(p) \in \mathbb{R}^{n\times n}$, $\mathcal{B}(p) \in \mathbb{R}^{n\times m_I}$, $\mathcal{C}(p) \in \mathbb{R}^{m_O\times n}$, $u(s) \in \mathbb{R}$ and $p \in \mathcal{D}$. A projection-based PMOR method [7,18] first builds two bases $Q, U \in \mathbb{R}^{n\times k}$ (normally, $k \ll n$), and then approximates $X(s,p) \approx Ux(s,p)$ ($x(s,p) \in \mathbb{R}^k$) in the range of U, and finally forces the residual to be orthogonal to the range of Q to obtain the pROM:

$$\begin{aligned}\big(sE(p) - A(p)\big)x(s,p) &= B(p)u(s),\\ y(s,p) &= C(s,p)x(s,p),\end{aligned} \tag{2}$$

where $[E(p), A(p)] = Q^T[\mathcal{E}(p), \mathcal{A}(p)]U$, $B(p) = Q^T\mathcal{B}(p)$ and $C(p) = \mathcal{C}(p)U$.

Now we discuss some subcategories under the general framework presented above:

- *(P)MOR based on projection using local bases* These methods build the bases Q and U only using the data computed at a given parameter value, say p_0. The resulting ROM is valid for p_0, but if derivative information with respect to p is also included in Q and U, we can obtain a local pROM [8,19,20]. This type of (p)ROMs can be used as the building blocks in our proposed pole-matching PMOR method. When derivative information is included in the bases, we can achieve Hermitian interpolation in our proposed method, which has the potential of reducing the number of needed parameter samples in order to cover a specific region in the parameter space.
- *PMOR based on projection using global bases*. Many PMOR methods build pROMs by projecting the nonparametric ROM onto a global subspace that contains the data obtained at different parameter values of p, namely p_1, p_2, ..., p_j [7,8,11]. Denoting the bases built by a nonparametric MOR method at p_i by U_i, the global bases U can be obtained by computing the SVD of $[U_1, U_2, \ldots, U_j]$.
- *PMOR based on interpolating local bases* These methods interpolate the bases precomputed at different p values, say U_i for p_i, to compute the bases for the requested parameter value p_* [10,12]. A straightforward interpolation normally does not work due to two reasons:

  1. The bases U_1, U_2, ..., U_j may be inconsistent. Suppose that the parametric dynamics is well captured by the family of subspaces $\mathcal{U}(p) := \text{colspan}\{U(p)\}$. Further, let $\mathcal{U}_1$, $\mathcal{U}_2$, ..., $\mathcal{U}_j$ well represent $\mathcal{U}(p)$ at p_1, p_2, ..., p_j. Then, intuitively, the interpolation of these subspaces is meaningful. Nevertheless, the basis matrices U_i for these subspaces computed by a MOR method using sampling at p_i will generally yield reduced-order models with states living in different coordinate systems as for any orthogonal $K \in \mathbb{R}^{k\times k}$, we have $\text{colspan}\{U_iK\} = \text{colspan}\{U_i\} = \mathcal{U}_i$. Hence, if we just interpolate the computed bases U_{i-1} and U_i, we consequently interpolate states in different coordinate systems which leads to inconsistencies and poor numerical approximation in state-space. Therefore, in the general case, one should try to compute consistent bases before we conduct interpolation. In [12], it was proposed to compute the bases U_iK for p_i "as consistent as possible" with the basis U_{i-1} for p_{i-1}, by aligning the subspaces via solving the optimization (Procrustes) problem

$$\min_{K\in\mathbb{R}^{k\times k}:K^TK=I} \|U_iK - U_{i-1}\|. \tag{3}$$

  2. Directly interpolating orthonormal matrices U_1, ..., U_j normally does not result in an orthonormal matrix. For example, if $U_2 = -U_1$, a direct linear interpolation at $\frac{p_1+p_2}{2}$ gives 0, which is apparently not a basis. Therefore, the success of these methods require interpolating on the correct manifold, e.g., on the Grassmann manifold or the Stiefel manifold.

- *PMOR based only on (nonparametric) ROMs* These methods build the pROM totally based on ROMs, which may be not generated by projection-based (P)MOR methods. Even if they are generated by projection-based (P)MOR methods, it is assumed that the bases U and Q are unknown. For the ROMs built by data-driven methods, e.g., the Loewner framework, we are in this situation.

MOR using the Loewner framework

The objective of MOR using the Loewner framework [15] is to construct a ROM in the frequency domain in the form of

$$\begin{cases} (sE - A)x = Bu, \\ y = Cx, \end{cases} \tag{4}$$

using measured samples of the FRF:

$$H(s) = C(sE - A)^{-1}B. \tag{5}$$

The Loewner framework assumes that the FRF is only known as a black box: the matrices E, A, B and C are assumed to be unknown but the FRF can be measured at frequency samples. For MIMO systems, each parameter sample is paired with either a left or right tangential direction for the measurement:

1. A right triplet sample: (λ_i, r_i, w_i). Given a frequency sample λ_i and the right tangential direction $r_i \in \mathbb{C}^{m_I \times 1}$, we measure $w_i = H(\lambda_i)r_i$.
2. A left triplet sample: (μ_j, ℓ_j, v_j). Given a frequency sample μ_j and the left tangential direction $\ell_j \in \mathbb{C}^{1\times m_O}$, we measure $v_j = \ell_j H(\mu_j)$.

Given n_R right samples and n_L left samples, the Loewner framework computes the *Loewner Matrix* $\mathbb{L} \in \mathbb{R}^{n_L \times n_R}$ with

$$[\mathbb{L}]_{ij} = \frac{v_i r_j - \ell_i w_j}{\mu_i - \lambda_j}, \tag{6}$$

and the *Shifted Loewner Matrix* $\mathbb{L}_\sigma \in \mathbb{R}^{n_L \times n_R}$ with

$$[\mathbb{L}_\sigma]_{ij} = \frac{\mu_i v_i r_j - \ell_i w_j \lambda_j}{\mu_i - \lambda_j}. \tag{7}$$

Using the Loewner matrix and the shifted Loewner matrix, a ROM can be constructed as follows [15].

- *The Loewner ROM in the original form* can be constructed as:

$$\begin{aligned} E &= -\mathbb{L}, \qquad B = V = [v_1, v_2, \ldots, v_{n_L}], \\ A &= -\mathbb{L}_\sigma, \qquad C = W = [w_1, w_2, \ldots, w_{n_R}]. \end{aligned} \tag{8}$$

  It is highly likely that the matrix pencil $(\mathbb{L}_\sigma, \mathbb{L})$ is (numerically) singular, but even in that case, the original Loewner ROM defined in (8) serves as a singular representation of an approximated FRF.

- *The Loewner ROM in the compressed form* is a concise and regular ROM computed from the Loewner ROM in the original form. First, we compute a rank-revealing SVD in a compressed form:

$$s\mathbb{L} - \mathbb{L}_\sigma = Y\Sigma X^* \approx Y_k \Sigma_k X_k^*, \tag{9}$$

where s is a frequency sample freely chosen from the set $\{\lambda_i\} \cup \{\mu_j\}$, and k is the number of dominant singular values chosen for the truncated SVD. Then, a Loewner ROM in the compressed form is constructed as [15]:

$$E = -Y_k^* \mathbb{L} X_k, \quad A = -Y_k^* \mathbb{L}_\sigma X_k, \quad B = Y_k^* V, \quad C = W X_k. \tag{10}$$

The Loewner framework has been extended to accommodate parametric systems as a data-driven PMOR method [21]. However, in "Interpolatory PMOR in the Loewner realization" section, we will study another possibility, namely the interpolation of Loewner ROMs.

The pole-matching PMOR method based on the pole-residue realization

In this section, we will first introduce the pole-residue realization for ROMs and develop a pole-matching PMOR method for single-input single-out (SISO) systems, part of which was covered in our conference paper [22]. Then, we will generalize the pole-matching PMOR method to interpolate MIMO ROMs in "The pole-matching method for MIMO systems" section.

The pole-matching method relies exclusively on ROMs at samples

$$\begin{aligned} \left(sI^{(i)} - A^{(i)}\right)x(s) &= B^{(i)}u(s), \\ y(s) &= C^{(i)}x(s), \end{aligned} \tag{11}$$

where $A^{(i)} \in \mathbb{R}^{k\times k}$, $B^{(i)} \in \mathbb{R}^{k\times r_I}$, $C^{(i)} \in \mathbb{R}^{r_O\times k}$, $I^{(i)} \in \mathbb{R}^{k\times k}$ is the identity matrix, and the ROM $\left(A^{(i)}, B^{(i)}, C^{(i)}\right)$ is built for the parameter value p_i $(i = 1, 2, \ldots, n_p)$. It is not required that the same MOR method is used to build ROMs for all p_i values.

However, when we do apply a projection-based MOR method to the FOM (1) to obtain ROMs in the form of (2) at each parameter sample p_i, it is easy to obtain the ROMs that we need here in the form of (11) as long as $E(p_i)$ is nonsingular by assigning $A^{(i)} \leftarrow E(p_i)^{-1}A(p_i)$, $B^{(i)} \leftarrow E(p_i)^{-1}B(p_i)$, and $C^{(i)} \leftarrow C(p_i)$.

Remark 1 The assumption of nonsingular $E(p_i)$ is satisfied in many important cases. For example, it always holds when we apply a projection-based MOR method to reduce a system of parametric ordinary differential equations (ODEs) (1) at p_i because $E(p_i) = Q^T\mathcal{E}(p_i)U$, Q and U are both of rank k, and $\mathcal{E}(p_i)$ is nonsingular in a system of ODEs. In many situations, e.g., when the model stems from a parametric finite-element model, E will be constant and nonsingular ((projected) mass matrix), and this is the typical class of models we are considering here. For such models, a parameter-dependent $E(p)$ may occur if the geometry of the domain on which the finite-element model is constructed is parameterized. Still, then $E(p)$ will be a mass matrix and will in general be nonsingular, also after (Petrov-)Galerkin projection. Even for differential-algebraic equations with differentiability index one, classical model reduction techniques like balanced truncation [18] or

IRKA [23] usually yield reduced-order models with nonsingular E by allowing a nonzero feedthrough-term Du in the output equation; see [24] for details.

The pole-residue realization for SISO systems

Before presenting the MIMO case in "The pole-matching method for MIMO systems" section, we focus on the pole-matching method for SISO systems, i.e., systems in the form of (11) with $r_I = r_O = 1$, and for simplicity of notation, we denote $B_j = B_{j,1}$ and $C_j = C_{1,j}$. In this section, we focus on a single ROM built at the parameter sample p_i and omit the index $\cdot^{(i)}$ in the system (11) for simpler notation:

$$\begin{aligned} (sI - A)x &= B, \\ y &= Cx. \end{aligned} \tag{12}$$

For now, we assume that the matrix A is nonsingular and all its eigenvalues are simple: the more complicated cases will be discussed in "Practical considerations" section. For a real eigenvalue λ_j and its corresponding eigenvector v_j, we have

$$Av_j = \lambda_j v_j,$$

while for the conjugate complex eigenpairs $(a_j \pm \imath b_j, r_j \pm \imath q_j)$, the definition $A(r_j \pm \imath q_j) = (a_j \pm \imath b_j)(r_j \pm \imath q_j)$ leads to:

$$A \begin{bmatrix} r_j & q_j \end{bmatrix} = \begin{bmatrix} r_j & q_j \end{bmatrix} \begin{bmatrix} a_j & b_j \\ -b_j, & a_j \end{bmatrix}.$$

Define

$$\Lambda = \begin{bmatrix} \Lambda_1 & & & \\ & \Lambda_2 & & \\ & & \ddots & \\ & & & \Lambda_m \end{bmatrix}, \qquad P = [P_1, P_2, \ldots, P_m], \tag{13}$$

where for the single real eigenpair (λ_j, v_j),

$$\Lambda_j = \begin{bmatrix} \lambda_j \end{bmatrix}, \qquad P_j = \begin{bmatrix} v_j \end{bmatrix} \qquad \text{and} \qquad m_j = 1, \tag{14}$$

while for the conjugate complex eigenpairs $(a_j \pm \imath b_j, r_j \pm \imath q_j)$,

$$\Lambda_j = \begin{bmatrix} a_j & b_j \\ -b_j, & a_j \end{bmatrix}, \qquad P_j = \begin{bmatrix} r_j & q_j \end{bmatrix} \qquad \text{and} \qquad m_j = 2. \tag{15}$$

To facilitate the later more generic discussion for semisimple and defective eigenvalues, we assume $\Lambda_j \in \mathbb{R}^{m_j \times m_j}$ and $P_j \in \mathbb{R}^{k \times m_j}$ in general with m_j a possibly larger integer.

Then, the complex eigenvalue decomposition is described by the following real matrix Eq. [25]:

$$AP = P\Lambda, \tag{16}$$

and it follows the associated similarity transformation

$$A = P\Lambda P^{-1}. \tag{17}$$

Therefore

$$y = C(sI - A)^{-1}B = C(sI - P\Lambda P^{-1})^{-1}B = CP(sI - \Lambda)^{-1}P^{-1}B. \tag{18}$$

Define

$$C^{\mathrm{I}} = CP = [C_1^{\mathrm{I}}, C_2^{\mathrm{I}}, \ldots, C_m^{\mathrm{I}}], \qquad B^{\mathrm{I}} = P^{-1}B = [B_1^{\mathrm{I,T}}, B_2^{\mathrm{I,T}}, \ldots, B_m^{\mathrm{I,T}}]^{\mathrm{T}}, \tag{19}$$

where $C^{\mathrm{I}} \in \mathbb{R}^{1\times k}$, $B^{\mathrm{I}} \in \mathbb{R}^{k\times 1}$, $C_j^{\mathrm{I}} \in \mathbb{R}^{1\times m_j}$, and $B_j^{\mathrm{I}} \in \mathbb{R}^{m_j\times 1}$. Then, we derive

$$y = \sum_{i=1}^{m} C_j^{\mathrm{I}}(sI - \Lambda_j)^{-1}B_j^{\mathrm{I}}. \tag{20}$$

For the real eigenpair (λ_j, v_j), C_j^{I} and B_j^{I} are scalars, with which we define

$$C_j^{\mathrm{II}} = C_j^{\mathrm{I}}B_j^{\mathrm{I}} \quad \text{and} \quad B_j^{\mathrm{II}} = 1 \tag{21}$$

and derive

$$C_j^{\mathrm{I}}(sI - \Lambda_j)^{-1}B_j^{\mathrm{I}} = \frac{C_j^{\mathrm{I}}B_j^{\mathrm{I}}}{s - \lambda_j} = C_j^{\mathrm{II}}(sI - \Lambda_j)^{-1}B_j^{\mathrm{II}}, \tag{22}$$

while for the conjugate complex eigenpairs $(a_j \pm \imath b_j, r_j \pm \imath q_j)$, we first define $C_j^{\mathrm{I}} = [C_{j,1}^{\mathrm{I}}, C_{j,2}^{\mathrm{I}}] \in \mathbb{R}^{1\times 2}$ and $B_j^{\mathrm{I}} = [B_{j,1}^{\mathrm{I}}, B_{j,2}^{\mathrm{I}}]^{\mathrm{T}} \in \mathbb{R}^{2\times 1}$, and then derive

$$\begin{aligned}
&C_j^{\mathrm{I}}(sI - \Lambda_j)^{-1}B_j^{\mathrm{I}} \\
&\quad = \frac{C_j^{\mathrm{I}}\begin{bmatrix} s - a_j & b_j \\ -b_j & s - a_j \end{bmatrix}B_j^{\mathrm{I}}}{(s - a_j)^2 + b_j^2} \\
&\quad = \frac{\left(C_{j,1}^{\mathrm{I}}B_{j,1}^{\mathrm{I}} + C_{j,2}^{\mathrm{I}}B_{j,2}^{\mathrm{I}}, C_{j,2}^{\mathrm{I}}B_{j,1}^{\mathrm{I}} - C_{j,1}^{\mathrm{I}}B_{j,2}^{\mathrm{I}}\right)\begin{bmatrix} s - a_j & b_j \\ -b_j & s - a_j \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}}{(s - a_j)^2 + b_j^2} \\
&\quad = C_j^{\mathrm{II}}(sI - \Lambda_j)^{-1}B_j^{\mathrm{II}},
\end{aligned} \tag{23}$$

where we define

$$C_j^{\mathrm{II}} = (C_{j,1}^{\mathrm{I}}B_{j,1}^{\mathrm{I}} + C_{j,2}^{\mathrm{I}}B_{j,2}^{\mathrm{I}}, C_{j,2}^{\mathrm{I}}B_{j,1}^{\mathrm{I}} - C_{j,1}^{\mathrm{I}}B_{j,2}^{\mathrm{I}}) \quad \text{and} \quad B_j^{\mathrm{II}} = [1,\ 0]^{\mathrm{T}}. \tag{24}$$

Therefore,

$$\begin{aligned} y &= \sum_{j=1}^{m} C_j^{\mathrm{I}}(sI-\Lambda_j)^{-1}B_j^{\mathrm{I}} \\ &= \sum_{j=1}^{m} C_j^{\mathrm{II}}(sI-\Lambda_j)^{-1}B_j^{\mathrm{II}} \\ &= C^{\mathrm{II}}(sI-\Lambda)^{-1}B^{\mathrm{II}}, \end{aligned} \tag{25}$$

where $C^{\mathrm{II}} = [C_1^{\mathrm{II}}, C_2^{\mathrm{II}}, \ldots, C_m^{\mathrm{II}}]$ and $B^{\mathrm{II}} = [B_1^{\mathrm{II,T}}, B_2^{\mathrm{II,T}}, \ldots, B_m^{\mathrm{II,T}}]^{\mathrm{T}}$.

Definition 1 For the linear system (A, B, C) in (11), its *pole-residue realization* is defined as $(\Lambda, B^{\mathrm{II}}, C^{\mathrm{II}})$ in (25), namely

$$\begin{aligned} (sI-\Lambda)x &= B^{\mathrm{II}}, \\ y &= C^{\mathrm{II}}x. \end{aligned} \tag{26}$$

The pole-matching method

The pole-residue realization is a natural choice for the interpolation of ROMs because of the following theorem.

Theorem 1 *Assume that* $(\Lambda^{(i)}, B^{II,(i)}, C^{II,(i)})$ $(i = 1, 2, \ldots, n_p)$ *are ROMs in the pole-residue realization built for* p_i*, respectively, all of which are of the same dimension* k*. Assume further that the block structures of* $\Lambda^{(i)}$ *are the same, i.e., for any* $1 \le j \le m$*, the size of* $\Lambda^{(i)}$ *are the same for all i's. Then, interpolating the matrices* $(\Lambda^{(i)}, B^{II,(i)}, C^{II,(i)})$ *(for the parameter p) is equivalent to interpolating the positions and residues of each pole, respectively.*

Proof This is an apparent result from construction since in the pole-residue realization, the positions and residues of the poles are directly stored in the matrix Λ and C^{II}, respectively, for both real eigenvalues and conjugate complex eigenvalues, as is shown in Eqs. (14), (15), (21), (22), (23) and (24). □

Remark 2 This theorem shows that when we interpolate the pole-residue realizations $(\Lambda^{(i)}, B^{\mathrm{II},(i)}, C^{\mathrm{II},(i)})$, we interpolate the positions and residues of the poles, which are values intrinsic to the FRFs themselves. By contrast, interpolating systems of the realization $(\Lambda^{(i)}, B^{\mathrm{I},(i)}, C^{\mathrm{I},(i)})$, where we only assume that $\Lambda^{(i)}$ takes the form of (13) and impose no requirement on $B^{\mathrm{I},(i)}$ and $C^{\mathrm{I},(i)}$, may introduce additional degrees of freedom introduced by realization. For example, assume that $(\Lambda^{(i)}, B^{\mathrm{I},(i)}, C^{\mathrm{I},(i)})$ $(i = 1, 2)$ are two ROMs of dimension 3 describing the same system at the same parameter value with different realizations: $\Lambda^{(1)} = \Lambda^{(2)} = \mathrm{diag}\{\lambda_1, \lambda_2, \lambda_3\}$, ($\lambda_1$, λ_2 and λ_3 are three different real numbers), $B^{\mathrm{I},(1)} = [16, 2, 1]^{\mathrm{T}}$, $B^{\mathrm{I},(2)} = [4, 4, 4]^{\mathrm{T}}$, $C^{\mathrm{I},(1)} = [1, 8, 16]$, $C^{\mathrm{I},(1)} = [4, 4, 4]$. These two ROMs have the same FRF due to (22) and the residues for all poles are 16. A critical property of a sound interpolation-based framework is that, when we interpolate two equivalent systems at the same parameter value, the interpolated system should be equivalent to these two systems no matter what interpolation method we use. However, if we interpolate them linearly with the equal weight 0.5, we get $\Lambda^{(3)} = \mathrm{diag}\{\lambda_1, \lambda_2, \lambda_3\}$, $B^{\mathrm{I},(3)} = [10, 3, 2.5]^{\mathrm{T}}$, $C^{\mathrm{I},(3)} = [2.5, 6, 10]$, and the residues of the three poles become 25, 18 and 25, respectively, which are all wrong. Therefore, the motivation to define the pole-residue realization is

to remove the additional degrees of freedom, which may give rise to spurious results, by "modifying" C^{I} and B^{I} to C^{II} and B^{II}.

Remark 3 The computational cost of the eigendecomposition used in computing the pole-amplitude realization is low since we apply it to a ROM, which is typically of order $\mathcal{O}(10)$, in most cases less than 100. Therefore, considering the computing cost alone, any eigendecomposition method can be used. For numerical issues, we refer to "On defective eigenvalues" section because we will discuss general cases there without making the assumption that all eigenvalues of A are simple.

Remark 4 Although we have employed the eigendecomposition to compute the pole-amplitude realization like in modal analysis, the eigenmodes of the system may not be well approximated. Nevertheless, a ROM will not accurately capture the dynamics of the FOM unless it captures the dominant eigenmodes well enough, so usually, the dominant eigenmodes are captured quite well in a good ROM. Since our starting point is the ROM rather than the FOM, our full focus is on the input–output behavior of the system. Therefore, the computation of the modes of the system, which depends on all state variables of the FOM, is beyond our consideration. If eigenmodes are to be preserved, then it would be suggested to compute the ROMs at p_i using modal truncation [26] so that at p_i, the local ROMs contain the exact dominant eigenmodes. By our interpolation approach, the eigenmodes at non-sampling parameters are then approximated by the interpolated poles so that we expect good approximation up to the unavoidable interpolation error.

On the storage and interpolation of ROMs

Besides the state-space representation, the pole-residue realization can be stored and interpolated in a more efficient scheme.

- For a real eigenpair (λ_j, v_j), we only need to store two real numbers: λ_j and $C_j^{\mathrm{II}} = C_j^{\mathrm{I}} B_j^{\mathrm{I}}$ because $B_j^{\mathrm{II}} \equiv 1$ does not need to be stored.
- For a complex eigenpair $(a_j \pm \imath b_j, r_j \pm \imath q_j)$, we only need to store four real numbers: $a_j, b_j, C_{j,1}^{\mathrm{II}} = C_{j,1}^{\mathrm{I}} B_{j,1}^{\mathrm{I}} + C_{j,2}^{\mathrm{I}} B_{j,2}^{\mathrm{I}}$ and $C_{j,2}^{\mathrm{II}} = C_{j,2}^{\mathrm{I}} B_{j,1}^{\mathrm{I}} - C_{j,1}^{\mathrm{I}} B_{j,2}^{\mathrm{I}}$ because $B_j^{\mathrm{II}} \equiv [1,\ 0]^{\mathrm{T}}$ does not need to be stored.

Therefore, for an order-k ROM in the pole-residue realization, the storage is only $2k$: the vector $(\lambda_j, C_j^{\mathrm{II}})$ for a real eigenvalue and $(a_j, b_j, C_{j,1}^{\mathrm{II}}, C_{j,2}^{\mathrm{II}})$ for two conjugate complex eigenvalues.

Assuming that we have n_s real eigenvalues and n_d pairs of conjugate complex eigenvalues, we store the ROM with two matrices:

$$D \in \mathbb{C}^{n_d \times 4} \quad \text{and} \quad S \in \mathbb{C}^{n_s \times 2},$$

where each row of D stores a vector of the form $(a_j, b_j, C_{j,1}^{\mathrm{II}}, C_{j,2}^{\mathrm{II}})$ and each row of S stores a vector of the form $(\lambda_j, C_j^{\mathrm{II}})$.

Besides the storage efficiency, this storage scheme also has the following advantages:

- The order of eigenvalues can be easily rearranged, which provides us great flexibility in the pole-matching process.
- An unimportant pole can be easily removed, e.g., using the concept of pole dominance [27]. This can be useful when we want to interpolate ROMs of different orders:

for example, when a pole in one ROM cannot be matched to any pole of the other ROM and it is of low dominance, it can be removed in the interpolation process.
- It can be easily written back into the state-space representation.

Under the assumption that the matrix A is nonsingular and all its eigenvalues are simple, the pole-matching process to evaluate the ROM at $p_* \notin \{p_1, p_2, \ldots, p_{n_p}\}$ works as follows:

1. Given n_p ROMs built at $p_1, p_2, \ldots, p_{n_p}$, we first convert all these ROMs into the pole-residue representation $\{D^{(i)}, S^{(i)}\}$.
2. To get the ROM for p_*, we first choose an interpolation algorithm and accordingly, the pre-computed ROMs built at p's near p_*. For simplicity of presentation, we assume that p_1 and p_2 are chosen for the interpolation.
3. Match the positions and residues of the poles by matching the rows of $D^{(1)}$ and $D^{(2)}$, and the rows of $S^{(1)}$ and $S^{(2)}$, respectively. Denote the models after pole matching by $D_M^{(1)}, D_M^{(2)}, S_M^{(1)}$ and $S_M^{(2)}$.
4 Interpolate $D_M^{(1)}$ at p_1 and $D_M^{(2)}$ at p_2 to get D_* at p_*. Similarly, interpolate $S_M^{(1)}$ and $S_M^{(2)}$ to get S_*. The interpolated model at p_* is $\{D_*, S_*\}$.

The procedure above is only a general description. For example, we can use different criteria to match the poles. Here we give some examples.

1. The simplest method is to sort the rows of D and S according to their real parts or imaginary parts.
2. Another choice is to match the closest poles when all poles only move slightly between the two models.
3. We can also compute a local "merit function" for each pairing of two individual poles, one of $\{D^{(1)}, S^{(1)}\}$ and the other of $\{D^{(2)}, S^{(2)}\}$, i.e., a weighted sum of the distance between poles and the difference in the residue, and match the poles according to it. More specifically, to find the matched pole in the second ROM for a given pole in the first ROM, for a real pole $(\lambda_j^{(1)}, C_j^{\mathrm{II},(1)})$, we solve the optimization problem

$$\min_{i\in\{i|m_i=1\}} \left|\lambda_j^{(1)} - \lambda_i^{(2)}\right| + w\left|C_j^{\mathrm{II},(1)} - C_i^{\mathrm{II},(2)}\right|, \tag{27}$$

while for a conjugate complex eigenpairs $(a_j^{(1)}, b_j^{(1)}, C_{j,1}^{\mathrm{II},(1)}, C_{j,2}^{\mathrm{II},(1)})$, we solve

$$\min_{i\in\{i|m_i=2\}} \left|\left|a_j^{(1)}\right| - \left|a_i^{(2)}\right|\right| + \left|\left|b_j^{(1)}\right| - \left|b_i^{(2)}\right|\right| + w\left\|C_j^{\mathrm{II},(1)} - C_i^{\mathrm{II},(2)}\right\| \tag{28}$$

where w is a positive real number for weighting.
4. A global "merit function" can also be used, in which case the sum of the local "merit functions" is minimized. Suppose that the two ROMs have the same numbers of real poles and complex poles, respectively. We fix the order of the first ROM and represent the order of the poles in the second ROM after pole-matching by the vector ν, which is a permutation of $(1, 2, \ldots, n_d + n_s)$. Then we solve the following optimization problem to find ν:

$$\min_{\nu} \sum_{\substack{j\in\{j|m_j=1\}\\ i\in\{i|m_{\nu_i}=1\}}} \left|\lambda_j^{(1)} - \lambda_{\nu_i}^{(2)}\right| + w\left|C_j^{\mathrm{II},(1)} - C_{\nu_i}^{\mathrm{II},(2)}\right| + \sum_{\substack{j\in\{j|m_j=2\}\\ i\in\{i|m_{\nu_i}=2\}}} \left|\,|a_j^{(1)}| - |a_{\nu_i}^{(2)}|\right| + \left|\,|b_j^{(1)}| - |b_{\nu_i}^{(2)}|\right| + w\left\|C_j^{\mathrm{II},(1)} - C_{\nu_i}^{\mathrm{II},(2)}\right\|. \quad (29)$$

Note that all these methods have limitations. The first three methods may result in conflicts in pole-matching, a pole of one ROM is matched to multiple poles of the other ROM. Since the poles move when the parameters change, it can happen that $\lambda_1(p_1) \approx \lambda_2(p_2)$, $\lambda_1(p_2) \approx \lambda_2(p_1)$, and $\lambda_1(p_1)$ is far from $\lambda_1(p_2)$, which we call pole-crossing. If we simply match $\lambda_1(p_1)$ with $\lambda_2(p_2)$, and $\lambda_1(p_2)$ with $\lambda_2(p_1)$, we lose the true parametric dynamics of the problem. Pole crossing cannot be captured by the first and second methods, and the third and fourth method can also fail. But a trial and error method in pole-matching can always be tried if the engineer is able to tell whether the interpolated ROM is physically sound. When we are capable to build ROM ourselves, e.g., we have access to the FOM, we can build ROMs at more samples to better exploit the parameter space. An offline-online method can be developed to overcome these difficulties, but that is future work.

However, the current paper focuses on the cases where the ROMs are given and the following requirements are fulfilled:

1. The given ROMs are accurate enough. Otherwise, we may not be able to match the poles even when the change of parameters is very small.
2. Sufficiently many ROMs are provided to represent the parametric dynamics of the system. To compute the ROM at p_*, the poles of the ROMs chosen for interpolation should not change too much. Otherwise, pole matching is difficult due to the lack of data.

Practical considerations

For simplicity of presentation, the discussion above assumed that the matrix A in (11) is real with simple eigenvalues. Now we extend the method to more general cases.

The case of complex A

Assume that A in (12) is a complex matrix with simple eigenvalues. Now in general, the complex eigenvalues occur no longer in conjugate pairs. Therefore, we simply conduct the complex eigenvalue decomposition to diagonalize A. The following computational procedure and storage scheme are the same as in the case of real A with all eigenvalues real. Note that when A is complex, the "residues" stored in C_j^{II} are also complex numbers in general. Therefore, we need to interpolate both the positions and the "complex residues" of the poles.

On semisimple eigenvalues

Assume that the dynamical system (11) has a semisimple eigenvalue λ_j with multiplicity m_j. Then Λ_j in (13) is an $m_j \times m_j$ diagonal matrix with all diagonal elements λ_j, and C^{I} and B^{I} are row vector and column vector of length m_j, respectively. Its corresponding contribution in the sum (20) is

$$\begin{aligned}
&\left[C^{\mathrm{I}}_{j,1}, C^{\mathrm{I}}_{j,2}, \ldots, C^{\mathrm{I}}_{j,m_j}\right]
\begin{bmatrix} \lambda_j & & & \\ & \lambda_j & & \\ & & \ddots & \\ & & & \lambda_j \end{bmatrix}
\begin{bmatrix} B^{\mathrm{I}}_{j,1} \\ B^{\mathrm{I}}_{j,2} \\ \vdots \\ B^{\mathrm{I}}_{j,m_j} \end{bmatrix} \\
&= \left[\sum_i^{m_j} C^{\mathrm{I}}_{j,i} B^{\mathrm{I}}_{j,i}\right][\lambda_j][1] \\
&\triangleq C^{\mathrm{II}}_j \Lambda^{\mathrm{II}}_j B^{\mathrm{II}}_j. \qquad (30)
\end{aligned}$$

As this derivation shows, a simple solution is to define $C^{\mathrm{II}}_j = \sum_i^{m_j} C^{\mathrm{I}}_{j,i} B^{\mathrm{I}}_{j,i}$, $\Lambda^{\mathrm{II}}_j = [\lambda_j]$, $B^{\mathrm{II}}_j = 1$ and treat it as if it were a simple pole, i.e., we need only to store $(\lambda_j, C^{\mathrm{II}}_j)$ since $B^{\mathrm{II}}_j \equiv 1$.

However, in reality, the simple strategy does not always work. Consider a parametric system with two poles, one pole with multiplicity 2 for any parameter value and the other pole normally simple. If the two parametric poles coincide at the parameter value p_*, the simple procedure above just gives one pole with multiplicity 3, which introduces difficulty in interpolation with ROMs built at other p values, which have two poles. Therefore, we must "separate" the two poles even they happen to be at the same position. This problem will be discussed in more detail in our future work.

On defective eigenvalues

When A has defective eigenvalue(s), the dynamical system (11) cannot be written into the proposed pole-residue realization because the eigenvalue decomposition (16) no longer exists. In this case, A is similar to a Jordan matrix $J = P^{-1}AP$, the numerical computation of which is highly unstable [25]. Although in practical computations, defective eigenvalues rarely occur, especially for a ROM, due to numerical noise, a nearly defective eigenvalue can also lead to P having a very large condition number, which may cause numerical instability, e.g., in the computation of $B^{\mathrm{I}} = P^{-1}B$ in (19). Therefore, in practical computations, we always check the condition number of P, which can be computed by, e.g., the MATLAB function "`condeig`". When it is very large, the algorithm breaks and fails. The solution of this problem is future work.

The pole-matching method for MIMO systems

Now we generalize the pole-matching PMOR method to MIMO systems. When $r_I > 1$ and/or $r_O > 1$ and all eigenvalues of $A \in \mathbb{R}^{k\times k}$ are simple, our derivation in "The pole-residue realization for SISO systems" section holds until (19) with $C^{\mathrm{I}} \in \mathbb{R}^{r_O\times k}$, $B^{\mathrm{I}} \in \mathbb{R}^{k\times r_I}$, $C^{\mathrm{I}}_j \in \mathbb{R}^{r_O\times m_j}$, and $B^{\mathrm{I}}_j \in \mathbb{R}^{m_j\times r_I}$.

Eigenpair (λ_j, v_j) with $m_j = 1$

To study an individual term $C^{\mathrm{I}}_j(sI - \Lambda_j)^{-1}B^{\mathrm{I}}_j$ in (20), let f denote the index of the first non-zero entry of B^{I}_j (which must exist, since otherwise the pole can be removed), and define

$$
\begin{aligned}
&C^{\mathrm{II}}_{j,i} = C^{\mathrm{I}}_{j,i} B^{\mathrm{I}}_{j,f}, && C^{\mathrm{II}}_{j} = [C^{\mathrm{II}}_{j,1}, C^{\mathrm{II}}_{j,2}, \ldots, C^{\mathrm{II}}_{j,r_O}]^{\mathrm{T}},\\
&C^{\mathrm{II}} = [C^{\mathrm{II}}_{1}, C^{\mathrm{II}}_{2}, \ldots, C^{\mathrm{II}}_{k}], && B^{\mathrm{II}}_{j,f} = 1, \quad B^{\mathrm{II}}_{j,i} = \frac{B^{\mathrm{I}}_{j,i}}{B^{\mathrm{I}}_{j,f}} \ (\forall j \neq f),\\
&B^{\mathrm{II}}_{j} = [B^{\mathrm{II}}_{j,1}, B^{\mathrm{II}}_{j,2}, \ldots, B^{\mathrm{II}}_{j,r_I}], && B^{\mathrm{II}} = [B^{\mathrm{II}}_{1}, B^{\mathrm{II}}_{2}, \ldots, B^{\mathrm{II}}_{k}]^{\mathrm{T}}.
\end{aligned} \tag{31}
$$

The contribution of (λ_j, v_j) in the weighted sum (20) is

$$
\begin{aligned}
C^{\mathrm{I}}_{j}(sI - \Lambda_j)^{-1} B^{\mathrm{I}}_{j} &= \frac{C^{\mathrm{I}}_{j} B^{\mathrm{I}}_{j}}{s - \lambda_j}\\
&= \frac{[C^{\mathrm{I}}_{j,1}, C^{\mathrm{I}}_{j,2}, \ldots, C^{\mathrm{I}}_{j,r_O}]^{\mathrm{T}} \cdot [B^{\mathrm{I}}_{j,1}, B^{\mathrm{I}}_{j,2}, \ldots, B^{\mathrm{I}}_{j,r_I}]}{s - \lambda_j}\\
&= \frac{[C^{\mathrm{II}}_{j,1}, C^{\mathrm{II}}_{j,2}, \ldots, C^{\mathrm{II}}_{j,r_O}]^{\mathrm{T}} \cdot [B^{\mathrm{II}}_{j,1}, B^{\mathrm{II}}_{j,2}, \ldots, B^{\mathrm{II}}_{j,r_I}]}{s - \lambda_j}\\
&= C^{\mathrm{II}}_{j}(sI - \Lambda_j)^{-1} B^{\mathrm{II}}_{j}.
\end{aligned} \tag{32}
$$

Remark 5 As we have discussed in Remark 2, C^{I}_{j} and B^{I}_{j} are not uniquely defined for a given FRF because of the realization freedom. However, the positions and residues of the poles depend only on the FRF. In the case above, the position of the pole is λ_i and the residues of the MIMO system are the entries of the matrix

$$
\begin{bmatrix}
C^{\mathrm{I}}_{j,1} B^{\mathrm{I}}_{j,1} & \cdots & C^{\mathrm{I}}_{j,1} B^{\mathrm{I}}_{j,r_I}\\
\vdots & \ddots & \vdots\\
C^{\mathrm{I}}_{j,r_O} B^{\mathrm{I}}_{j,1} & \cdots & C^{\mathrm{I}}_{j,r_O} B^{\mathrm{I}}_{j,r_I}
\end{bmatrix}. \tag{33}
$$

Therefore, all entries of C^{II}_{j} are determined by the residues of the poles. Actually, all entries of B^{II}_{j} are also determined by the residues of the poles because

$$
B^{\mathrm{II}}_{j,i} = \frac{B^{\mathrm{I}}_{j,i}}{B^{\mathrm{I}}_{j,f}} = \frac{C^{\mathrm{I}}_{j,g} B^{\mathrm{I}}_{j,i}}{C^{\mathrm{I}}_{j,g} B^{\mathrm{I}}_{j,f}}, \tag{34}
$$

where $C^{\mathrm{I}}_{j,g}$ is any nonzero entry for C^{I}_{j}. However, $B^{\mathrm{II}}_{j,i}$ cannot be used for interpolation purposes. For example, if we interpolate a ROM built for p_1 and another ROM built for p_2 with their weights $\omega(p)$ and $(1 - \omega(p))$, respectively, we should compute $B^{\mathrm{II}}_{j,i}$ by

$$
B^{\mathrm{II}}_{j,i}(p) = \frac{w(p) C^{\mathrm{I}}_{j,g}(p_1) B^{\mathrm{I}}_{j,i}(p_1) + (1 - w(p)) C^{\mathrm{I}}_{j,g}(p_2) B^{\mathrm{I}}_{j,i}(p_2)}{w(p) C^{\mathrm{I}}_{j,g}(p_1) B^{\mathrm{I}}_{j,f}(p_1) + (1 - w(p)) C^{\mathrm{I}}_{j,g}(p_2) B^{\mathrm{I}}_{j,f}(p_2)},
$$

which first estimates the residues of the poles by interpolation and then compute $B^{\mathrm{II}}_{j,i}$ with these residues, rather than by

$$
B^{\mathrm{II}}_{j,i}(p) = w(p) \frac{B^{\mathrm{I}}_{j,i}(p_1)}{B^{\mathrm{I}}_{j,f}(p_1)} + (1 - w(p)) \frac{B^{\mathrm{I}}_{j,i}(p_2)}{B^{\mathrm{I}}_{j,f}(p_2)},
$$

which loses the connection with the residues of poles at p. Therefore, it is insufficient to only store B_j^{II} for interpolation purposes. We need to store

$$B_j^{\mathrm{II,U}} = \left[C_{j,g}^{\mathrm{I}} B_{j,1}^{\mathrm{I}}, C_{j,g}^{\mathrm{I}} B_{j,2}^{\mathrm{I}}, \ldots, C_{j,g}^{\mathrm{I}} B_{j,r_I}^{\mathrm{I}}\right], \quad b_j^{\mathrm{II,L}} = C_{j,g}^{\mathrm{I}} B_{j,f}^{\mathrm{I}} \tag{35}$$

and compute B_j^{II} as

$$B_j^{\mathrm{II}} = \frac{1}{b_j^{\mathrm{II,L}}} B_j^{\mathrm{II,U}}. \tag{36}$$

The pole-residue realization for MIMO ROMs

For the complex eigenpairs $(a_j \pm \imath b_j, r_j \pm \imath q_j)$ with $m_j = 2$, it is difficult to derive

$$C_j^{\mathrm{I}}(sI - \Lambda_j)^{-1} B_j^{\mathrm{I}} = C_j^{\mathrm{II}}(sI - \Lambda_j)^{-1} B_j^{\mathrm{II}}, \tag{37}$$

where $C_j^{\mathrm{II}} \in \mathbb{R}^{r_O \times 2}$ and $B_j^{\mathrm{II}} \in \mathbb{R}^{2 \times r_I}$ with all their entries either constants or the residues of the poles. This requires solving a system of nonlinear equations under constraints, which is difficult. Actually, we do not even know whether the solution exists in general. The research of this topic is future work.

Therefore, we propose two methods to obtain the pole-residue realization for MIMO ROMs.

The pole-residue realization for MIMO ROMs in the complex form When it is not required to preserve real realization for real systems, a pole-residue realization can be easily computed if all eigenvalues are simple. We substitute the similarity transformation (17) with a true eigendecomposition with Λ diagonal rather than block diagonal. Therefore, we can apply the method developed in "Eigenpair (λ_j, v_j) with $m_j = 1$" section to all the eigenvalues to compute the pole-residue realization.

An advantage of this method is that the MIMO structure is strictly preserved: the sizes of Λ, B^{II} and C^{II} equal those of A, B and C, respectively. However, for a real system with conjugate complex eigenpairs, complex numbers are introduced in the pole-residue realization.

The pole-residue realization for MIMO ROMs in the real form To derive the pole-residue realization for MIMO ROMs in the real form, we first consider the SIMO (single-input multiple-output) case.

The SIMO case For a SIMO system (12)

with $C \in \mathbb{R}^{r_O \times k}$, $B \in \mathbb{R}^{k \times 1}$, we denote

$$C = [C[1]^{\mathrm{T}}, C[2]^{\mathrm{T}}, \ldots, C[r_O]^{\mathrm{T}}]^{\mathrm{T}}, \tag{38}$$

where $C[i]$ represents the i-th row of C. Then, for each row of C, say $C[i]$, we compute the pole-residue realization for the SISO system $(A, B, C(i,:))$, which we denote by $(\Lambda[i], B^{\mathrm{II}}[i], C^{\mathrm{II}}[i])$.

Note that

$$\begin{aligned} &\Lambda[1] = \Lambda[2] = \ldots = \Lambda[r_O] = \Lambda \\ \text{and} \quad &B^{\mathrm{II}}[1] = B^{\mathrm{II}}[2] = \ldots = B^{\mathrm{II}}[r_O] = B^{\mathrm{II}} \end{aligned} \tag{39}$$

hold because

- Despite the different output vector $C(i,:)$, the similarity transformation (17) is the same because for all these SISO systems, the positions of all poles are the same, respectively. Therefore, $\Lambda[i] = \Lambda$.
- According to (21) and (24), $B^{\mathrm{II}}[i]$ depends completely on the structure of $\Lambda[i]$. Since $\Lambda[i] = \Lambda$, $B^{\mathrm{II}}[i] = B^{\mathrm{II}}$ also holds.

Due to the property (39), the pole-residue realization for the SIMO system is $(\Lambda, B^{\mathrm{II}}, C^{\mathrm{II}})$ with

$$C^{\mathrm{II}} = \left[C^{\mathrm{II,T}}[1], C^{\mathrm{II,T}}[2], \ldots, C^{\mathrm{II,T}}[r_O]\right]^{\mathrm{T}}. \tag{40}$$

The MIMO case For a MIMO system (12) with $C \in \mathbb{R}^{r_O \times k}$, $B \in \mathbb{R}^{k \times r_I}$, we denote

$$B = \left[B[1], B[2], \ldots, B[r_I]\right], \tag{41}$$

where $B[i]$ denotes the i-th column of B.

We first compute the pole-residue realization for each SIMO system $(A, B[i], C)$, which we denote by $(\Lambda, B^{\mathrm{II}}, C^{\mathrm{II}}\{i\})$. Due to a similar argument as that for (39), we deduce that Λ and B^{II} does not change with i. However, $C^{\mathrm{II}}\{i\}$ does change with i because under different input vectors and output vectors, the residues of each pole are different in general.

Using the method proposed in [28], which reformulates the MIMO systems as parallel connection of split systems, the pole-residue realization of the MIMO system (12) in the real form is

$$\begin{aligned} \mathbf{\Lambda} &= \mathrm{diag}\left[\Lambda, \Lambda, \ldots, \Lambda\right], \\ \mathbf{B}^{\mathrm{II}} &= \mathrm{diag}\left[B^{\mathrm{II}}, B^{\mathrm{II}}, \ldots, B^{\mathrm{II}}\right], \\ \mathbf{C}^{\mathrm{II}} &= \left[C^{\mathrm{II}}\{1\}, C^{\mathrm{II}}\{2\}, \ldots, C^{\mathrm{II}}\{r_I\}\right]. \end{aligned} \tag{42}$$

Although this realization preserves the MIMO structure with real algorithms, the dimension of the ROM is multiplied by the number of inputs. In practical computations of FRFs, however, we do not need to formulate (42) explicitly. We compute the FRF column-wise, i.e., to compute the i-th column of the FRF, we only need to formulate $(\Lambda, B^{\mathrm{II}}, C^{\mathrm{II}}\{i\})$ for $(A, B[i], C)$.

On the storage and interpolation of MIMO ROMs In the MIMO case, the FRF $H(s)$ is a matrix function with $r_O \times r_I$ entries. We denote the function for the (j,k)-th entry by $H_{j,k}(s)$. All these functions have the same poles, which need to be stored only once. Normally, they have different residues for the poles, which needs to be stored individually. To conduct ROM interpolation, we conduct pole matching and pole interpolation using similar methods as discussed for the SISO case. A major difference is that now we have more residue information, which we can use to construct the merit function. With the

positions and residues of all poles for all FRF entries, an interpolated ROM can be constructed. The previous two sections actually give two procedures that construct a ROM from the positions and residues of all poles, one for a complex realization and the other for a real realization.

For the complex realization discussed in "The pole-residue realization for MIMO ROMs in the complex form" section, we only need to store the diagonal elements of Λ and the matrices C^{II} in (31), along with $B_j^{\mathrm{II,U}}$ and $b_j^{\mathrm{II,L}}$ ($j = 1, 2, \ldots, k$) in (35), a total of a total of $k \times (r_O + r_I + 2)$ complex numbers. When we conduct interpolation using this realization after pole-matching, we also need to check that the indices g and f in (35) are the same for each j, respectively.

For the real realization form discussed in "The pole-residue realization for MIMO ROMs in the real form" section, we need to store the positions of all poles and $\mathbf{C}^{\mathrm{II}}$ defined in (42), a total of $k \times (r_O \times r_I + 1)$ real numbers.

Interpolatory PMOR in the Loewner realization

In this section, we propose a method that interpolates ROMs built by the Loewner framework to approximate the dynamical system described by the FRF $H(s, p)$. We will deal with the interpolation of Loewner ROMs in the original form in "Interpolating Loewner ROMs in the original form" section, and the interpolation of Loewner ROMs in the compressed form in "Interpolating Loewner ROMs in the compressed form" section. Both methods rely on Assumption 1.

Assumption 1 We assume that for all sampled values for the parameter p_l ($l = 1, 2, \ldots, n_p$), the same frequency samples and right/left tangential directions are used. Therefore, given the frequency shifts μ_i and the corresponding left tangential direction ℓ_i ($i = 1, 2, \ldots N_\omega$) a left sample of $H(s, p_l)$ is defined by

$$(\mu_i, \ell_i, v_i(p_l)), \qquad \text{where } v_i(p_l) = \ell_i H(\mu_i, p_l). \tag{43}$$

Similarly, given the frequency shifts λ_j and the corresponding right tangential direction r_j ($j = 1, 2, \ldots N_\omega$), a right sample of $H(s, p_l)$ is defined by

$$(\lambda_j, r_j, w_j(p_l)), \qquad \text{where } w_j(p_l) = H(\lambda_j, p_l) r_j. \tag{44}$$

Under Assumption 1, the Loewner matrix and the shifted Loewner matrix at p_l, which we denote by $\mathbb{L}(p_l)$ and $\mathbb{L}_\sigma(p_l)$, respectively, are defined by

$$\left[\mathbb{L}(p_l)\right]_{ij} = \frac{v_i(p_l) r_j - \ell_i w_j(p_l)}{\mu_i - \lambda_j}, \qquad \left[\mathbb{L}_\sigma(p_l)\right]_{ij} = \frac{\mu_i v_i(p_l) r_j - \ell_i w_j(p_l) \lambda_j}{\mu_i - \lambda_j}. \tag{45}$$

Interpolating Loewner ROMs in the original form

The following theorem shows the physical meaning of interpolating Loewner ROMs in the original form.

Theorem 2 *Assume that (E_l, A_l, B_l, C_l) ($l = 1, 2, \ldots n_p$) are ROMs built at p_l by the Loewner framework in the original form with the left/right triplet samples $(\mu_i, \ell_i, v_i(p_l))$ and $(\lambda_j, r_j, w_j(p_l))$, which satisfy Assumption 1. Then, the following two pROMs built for the parameter value p are equal:*

a. The pROM $(E_a(p), A_a(p), B_a(p), C_a(p))$ *obtained by applying an arbitrary interpolation operator of the form*

$$M(p) = \sum_{l=1}^{n_p} M_l \phi_l(p), \qquad (\phi_i(p_j) = \delta_{ij}) \tag{46}$$

to each of E, A, B and C.

b. The pROM $(E_b(p), A_b(p), B_b(p), C_b(p))$ *built by the Loewner framework in the original form using the "interpolated left/right data" at p, which is obtained by using the interpolation operator* (46) *on the left/right samples of the FRF:*

$$\left(\mu_i, \ell_i, \sum_{i=1}^{n_p} v_i(p_l)\phi_l(p)\right) \quad \textit{and} \quad \left(\lambda_j, r_j, \sum_{l=1}^{n_p} w_j(p_l)\phi_l(p)\right). \tag{47}$$

A proof for Theorem 2 is given in Appendix A.

Theorem 2 shows that interpolating Loewner ROMs in the original form results in a pROM that is "optimal" from the perspective of the left/right triplet samples. However, this method is practically unsatisfactory as it needs more memory storage than the original left/right triplet samples. Therefore, our ultimate goal is to interpolate Loewner ROMs in the compressed form.

Interpolating Loewner ROMs in the compressed form

To study the interpolation of Loewner ROMs in the compressed form, we first parameterize Eq. (9) by denoting the (truncated-)SVD at the parameter value p_l by

$$\begin{aligned} s_l\mathbb{L}_l - \mathbb{L}_{\sigma l} = Y_l \Sigma_l X_l^* \approx Y_{l,k} \Sigma_{l,k} X_{l,k}^*, \qquad & s_l \in \{\lambda_{l,i}\} \cup \{\mu_{l,j}\}, \\ V_l = [v_{l,1}, v_{l,2}, \ldots, v_{l,n_L}], \qquad & W_l = [w_{l,1}, w_{l,2}, \ldots, w_{l,n_R}]. \end{aligned} \tag{48}$$

Proposition 1 *The matrices* X_l *and* Y_l *defined in* (48) *are generalized controllability and observability matrices of the system* (4) *at* p_l*, respectively.*

For a proof of Proposition 1, we refer to Theorem 5.2 in [15]. Therefore, we can compress the ROM in the original representation by ignoring the hardly controllable and observable vectors from X_l and Y_l, i.e., taking the first k dominant columns of X_l and Y_l, respectively, and then projecting the state vector to the range of $X_{l,k}$ and the dual state vector to the range of $Y_{l,k}$. This procedure leads to a Loewner ROM in the compressed form as in (10). Now we propose Algorithm 1 for generating Loewner ROMs in the compressed form that can directly be used for interpolation.

Theorem 3 *For any index l, the controllability matrix satisfies*

$$\textit{rowspan}\,\{X_l\} \subseteq \textit{rowspan}\,\{X\}$$

and the observability matrix satisfies

$$\textit{colspan}\,\{Y_l\} \subseteq \textit{colspan}\,\{Y\},$$

where X, Y are defined in (50) *and* (49)*, respectively.*

Algorithm 1 Generation of interpolation-oriented Loewner ROMs in the compressed form for the data set under Assumption 1.

1: Build the global basis Y by computing the economy-size SVD of

$$\begin{aligned}&\left[\begin{array}{c|c|c|c} s_1\mathbb{L}_1-\mathbb{L}_{\sigma 1} & s_2\mathbb{L}_2-\mathbb{L}_{\sigma 2} & \cdots & s_{n_p}\mathbb{L}_{n_p}-\mathbb{L}_{\sigma n_p}\end{array}\right] \\ &= Y\Sigma_H X_H \\ &\approx Y_K\Sigma_{H,K}X_{H,K},\end{aligned} \tag{49}$$

where $Y, \Sigma_H \in \mathbb{R}^{n\times n}$, $X_H \in \mathbb{R}^{n\times n_p n}$, and Y_K, $\Sigma_{H,K}$ and $X_{H,K}$ are obtained by truncated SVD using the first K singular values.

2: Build the global basis X by computing the economy-size SVD of

$$\begin{bmatrix} s_1\mathbb{L}_1-\mathbb{L}_{\sigma 1} \\ s_2\mathbb{L}_2-\mathbb{L}_{\sigma 2} \\ \cdots \\ s_{n_p}\mathbb{L}_{n_p}-\mathbb{L}_{\sigma n_p}\end{bmatrix} = Y_V\Sigma_V X \approx Y_{V,K}\Sigma_{V,K}X_K, \tag{50}$$

where $X, \Sigma_V \in \mathbb{R}^{n\times n}$, $Y \in \mathbb{R}^{n_p n\times n}$ and $Y_{V,K}$, $\Sigma_{V,K}$ and X_K are obtained by truncated SVD using the first K singular values.

3: Build the "Compressed" Representations using the global bases:

$$E_l = -Y_K^*\mathbb{L}_l X_K, \quad A_l = -Y_K^*\mathbb{L}_{\sigma,l}X_K, \quad B_l = Y_K^*V_l, \quad C_l = W_l X_K. \tag{51}$$

4: Given an interpolation operator of the form (46), the interpolatory pROM is given by

$$M(p) = \sum_{l=1}^{n_p} M_l\phi_l(p), \qquad M \in \{E, A, B, C\}. \tag{52}$$

A proof for Theorem 3 is given in Appendix A. According to Theorem 3, if we truncate X an Y to eliminate the hardly controllable and observable subspaces, respectively, all ROMs for $l = 1, 2, \ldots, n_p$ are accurately approximated by (51).

Remark 6 The Loewner pROM in the compressed form (51) is a good approximation of the Loewner pROM in the original form (46), as long as K is large enough so that X_K and Y_K capture all dominant components of X and Y. This is because

- The pROM (51) is actually obtained by applying the projection method with global bases to the pROM (46).
- At each interpolation point p_l, the controllability and the observability is captured well by (51) according to Theorem 3, i.e., the Loewner pROM (51) in the compressed form interpolates the Loewner ROMs in the original form well.

Results

In this section, we apply the developed methods to three applications: a microthruster model [19], a "FOM" model [21], and a footbridge model [29]. We compare three methods: the pole-matching method, interpolatory PMOR in the Loewner realization, and the interpolation of ROMs on the nonsingular matrix manifold [12].

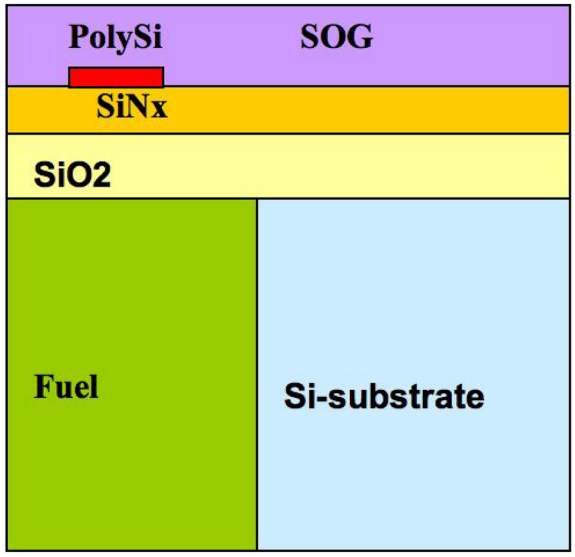

Fig. 1 A schematic 2D illustration of the microthruster

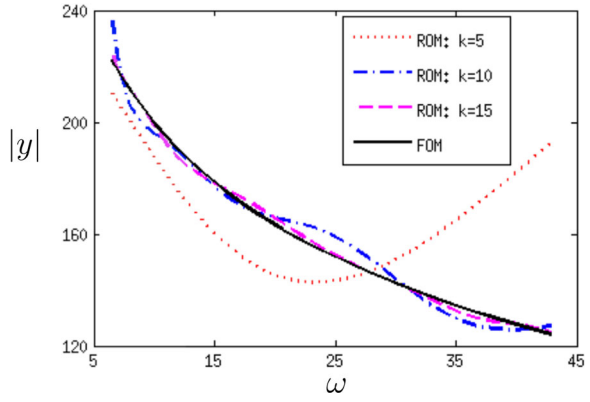

Fig. 2 The FRFs of Loewner ROMs and the FRF of the FOM at $p = 268.3$

PMOR on the microthruster model

In this section, we study the performance of the proposed methods on data-driven ROMs in the frequency domain. The FRFs used to compute the data-driven ROMs are generated by the microthruster model [19]. A schematic diagram of the microthruster is shown in Fig. 1. The microthruster model is of the form (4) with order $n = 4257$ and has a single parameter: the film coefficient.

First in Fig. 2, we show the convergence of nonparametric Loewner ROM in the compressed realization, which is generated with 100 samples for λ and 100 samples for μ. With the increase of the dimension, the FRF of the Loewner ROM becomes closer to that of the FOM. Therefore, the Loewner ROMs are suitable for our study of the interpolation of ROMs.

Then, we apply the three PMOR methods to the microthruster model.

- *The manifold method* This method interpolates ROMs on the nonsingular matrix manifold [12]. In the numerical tests, we first build ROMs at the parameter samples

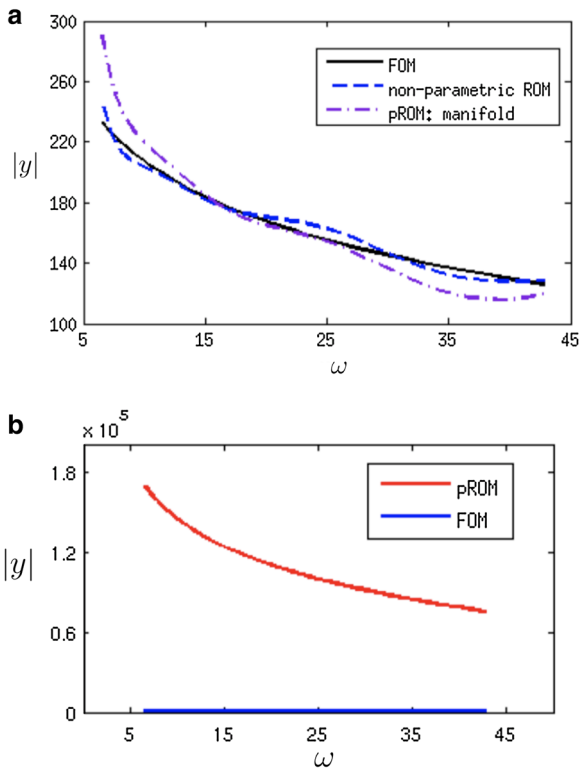

Fig. 3 The behavior of ROMs built by interpolation on the nonsingular matrix manifold. **a** The ROM with $k = 10$ and $p_* = 65.51$. **b** The ROM with $k = 11$ and $p_* = 65.51$

$p_1 = 10$, $p_2 = 268.3$, $p_3 = 7197$, based on which we use the manifold method to build the pROM. The FRF of the interpolated pROM at $p_* = 65.51$ is shown in Fig. 3. The approximation quality improves as the order of the ROM k increases up to 10. However, when $k > 10$, the approximation quality becomes unacceptable. Because of its unsatisfactory performance, we will not test this method any further in the next two numerical examples.

- *Interpolation of ROMs in the Loewner realization* (Algorithm 1) First, we interpolate the Loewner ROMs in the original form. Figure 4 shows that this method is much more accurate than the manifold method.

Furthermore, the pROM is much more stable: we have never observed divergence as we increase the order of the pROM. Then, we interpolate the Loewner ROMs in the compressed form built by Algorithm 1. In Fig. 5, we show the numerical results when different shifts $s \in \{\lambda_{i,l}\} \cup \{\mu_{j,l}\}$ (defined in (48)) are used. Within each subfigure, the nonparametric ROMs used for interpolation are built for all p_l's using the same shift s specified in the subtitle. It is shown that no matter what shift we use, the resulting ROM is accurate. As a reference, we show in Fig. 6 the numerical results for interpolating the Loewner ROMs in the compressed form using local bases (10) rather than using the global bases in Algorithm 1. The numerical results show that using individual bases rather than global bases in compressing the ROMs, we cannot obtain a pROM with high fidelity by interpolation. In Fig. 7a, we plot the response

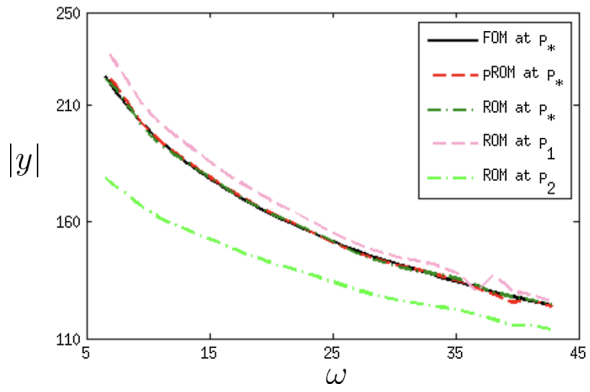

Fig. 4 Interpolation of Loewner ROMs in the original form. The pROM with order $k = 21$ is obtained by interpolating nonparametric Loewner ROMs in the original form at $p_1 = 10$ and $p_2 = 268.3$. The FRFs at $p_* = 65.51$ are shown for comparison

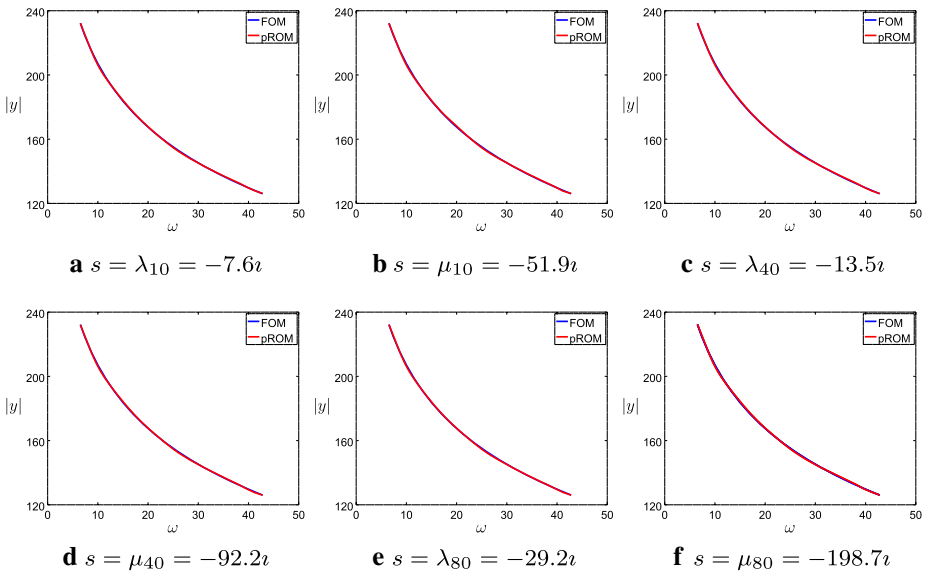

Fig. 5 Interpolation of Loewner ROMs built with global basis using Algorithm 1. The nonparametric Loewner ROMs are built in the compressed form with different frequency shift s. The FRFs at $p_* = 65.51$ are shown in the figure

surface of the FOM along with the absolute error of the interpolated ROMs generated by Algorithm 1. The figure plots FRFs for 29 samples for the parameter p: p_1, p_2, ..., p_{29}. The samples p_1, p_8, p_{15}, p_{22} and p_{29} are used to build the global bases, and the FRFs at all other p's are obtained by an interpolated ROM generated by Algorithm 1. In Fig. 7b, we show that using a more advanced spline interpolation, higher accuracy can be achieved.

- *Interpolation of ROMs in the pole-residue realization* To apply the interpolation method based on the pole-residue realization, we first study the pole-matching criterion. We use the criterion that the overall differences in pole positions and differences in pole residues are minimal, which agrees with the intuition on the optimal pole-matching solution. Figure 8 shows the poles of the ROMs for p_{22} and p_{29}, respectively.

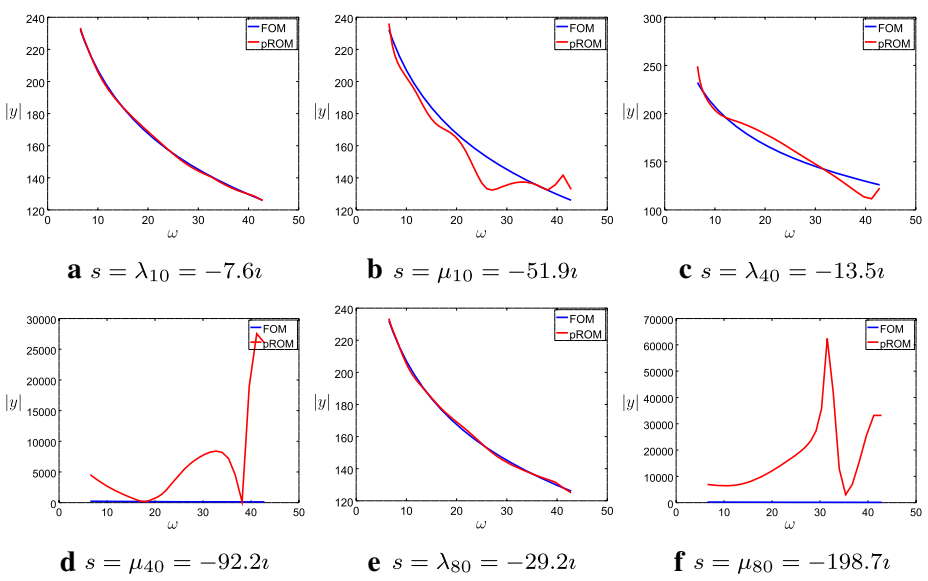

Fig. 6 Interpolation of Loewner ROMs built with individual bases. The nonparametric Loewner ROMs are built in the compressed form with different frequency shift s and they are interpolated matrix-wise directly using (46). The FRFs at $p_* = 65.51$ are shown in the figure

In this example, the ROMs are of the form (4) with complex E, A, B and C. Since E is nonsingular, we left multiply the system by E^{-1}, which is of the reduced dimension, to obtain the system of the form (11), for which the pole-residue realization is defined. In this example, the matrix A is complex. To conduct PMOR, we first convert the ROMs (in the form of (11)) at p_1, p_8, p_{15}, p_{22} and p_{29} to the pole-residue realization. Then, we conduct linear interpolation among the resulting ROMs. The result is shown in Fig. 7(c). Its accuracy is slightly better than the linear ROM interpolation of the Loewner representation.

Results on the parametric "FOM" model

Now we apply our method to the parametric "FOM" model presented in [21], which is adapted from the nonparametric "FOM" model in [30]:

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

where $\mathcal{C} = [10, 10, 10, 10, 10, 10, 1, \ldots, 1]$, $\mathcal{B} = \mathcal{C}^{\mathrm{T}}$, and $\mathcal{A}(p) = \mathrm{diag}(\mathcal{A}_1(p), \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4)$ with $\mathcal{A}_4 = -\mathrm{diag}(1, 2, \ldots, 1000)$,

$$\mathcal{A}_1(p) = \begin{bmatrix} -1 & p \\ -p & -1 \end{bmatrix}, \quad \mathcal{A}_2 = \begin{bmatrix} -1 & 200 \\ -200 & -1 \end{bmatrix}, \quad \mathcal{A}_3 = \begin{bmatrix} -1 & 400 \\ -400 & -1 \end{bmatrix}.$$

We first use interpolate ROMs in the Loewner realization (Algorithm 1) to obtain a pROM. This example, however, shows the limitation of Algorithm 1 in dealing with peak(s) that moves significantly with the change of parameter(s). As was discussed in [7],

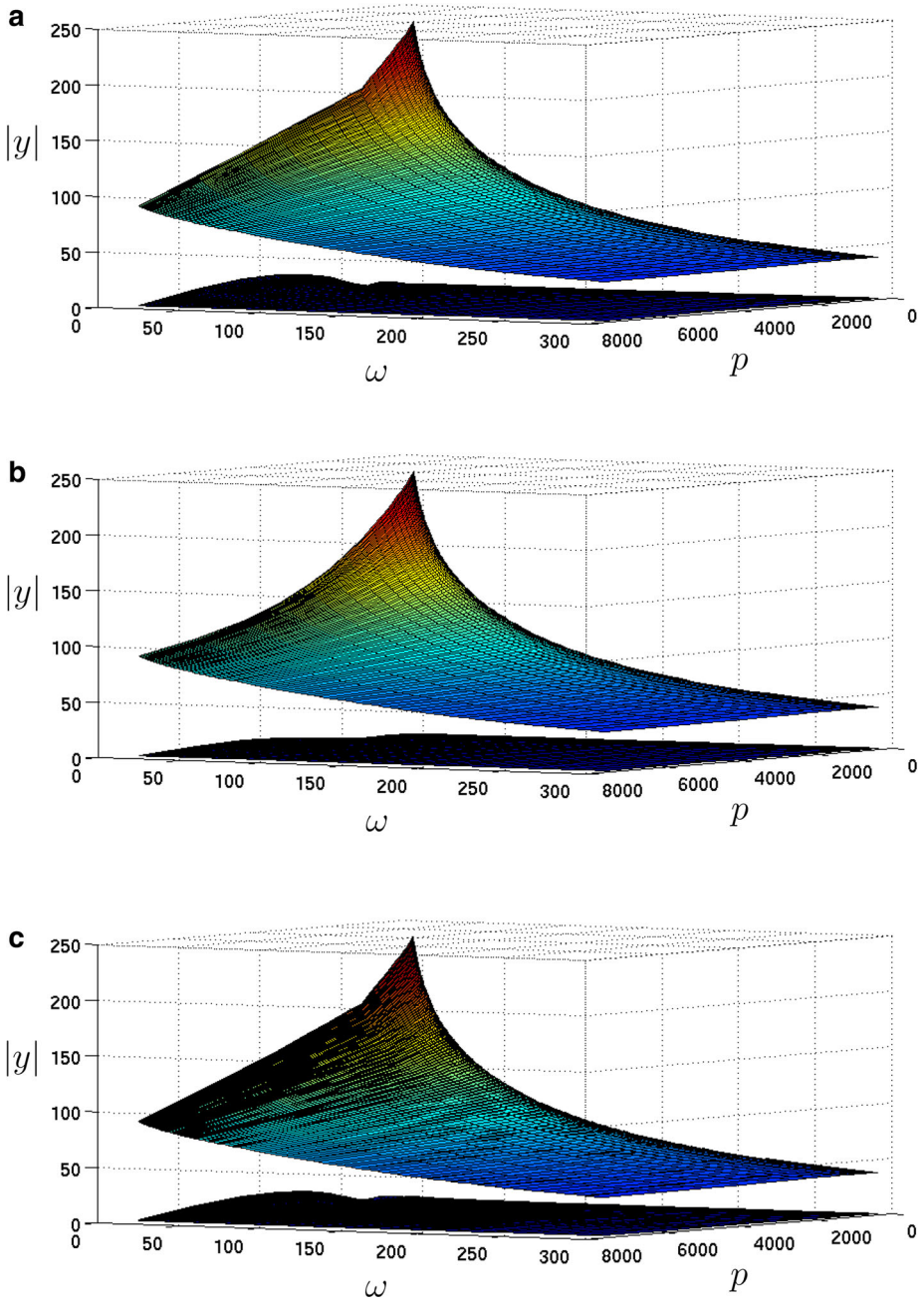

Fig. 7 Response surface and the absolute error. **a** Linear interpolation of the Loewner representation. The overall relative error is 2.4649×10^{-2}. **b** Spline interpolation of the Loewner representation. The overall relative error is 6.0671×10^{-3}. **c** Linear interpolation of the pole-residue representation. The overall relative error is 2.3485×10^{-2}

when we interpolate FRFs, the positions of poles do not change. This is because the interpolated FRF

$$\sum_{i=1}^{k} w_i(p) C_i (sE_i - A_i)^{-1} B_i \tag{53}$$

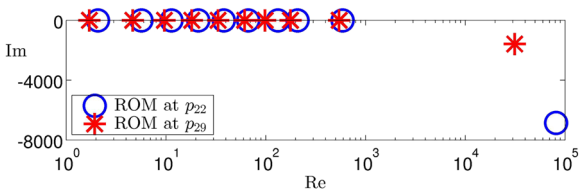

Fig. 8 The poles of the ROMs for p_{22} and p_{29}, respectively

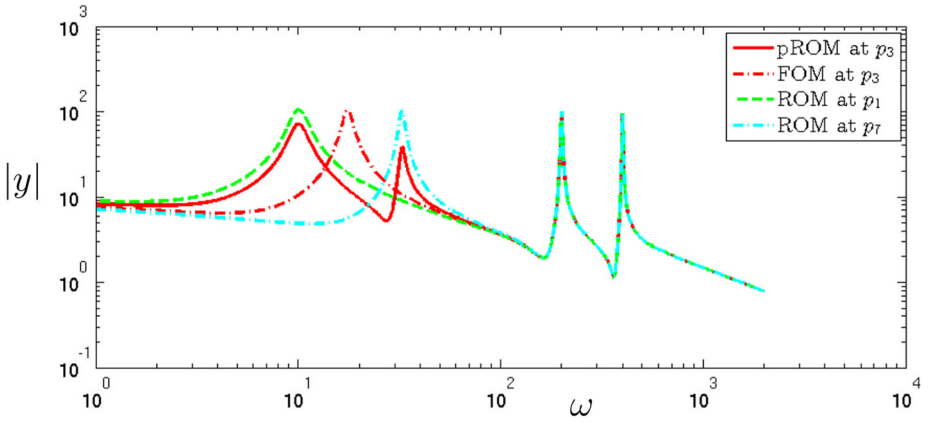

Fig. 9 Interpolating two ROMs built by the Loewner framework. The pROM is built by interpolating the ROMs built at p_1 and p_7

has a pole at any s, at which any of the individual FRFs

$$C_i(sE_i - A_i)^{-1}B_i \tag{54}$$

has a pole. Therefore, the poles of the interpolated FRF are the union of the poles of the interpolating FRFs. This phenomenon is clearly shown in Fig. 9: with the change of the parameter, the pROM generated by Algorithm 1 does not capture the "moving peak", which is the true dynamics, but rather evolves by waxing and waning of two fixed peaks, which also interpolates the two FRFs but seldom occurs in real applications.

Remark 7 We note the research efforts in avoiding the problem of "fixed peaks" when we interpolate the FRFs. For example, a pair of scaling parameters were introduced in [31]. The method works well when all peaks move in the same direction at a similar rate, which was proved by their numerical tests. In general, however, the method is insufficient to describe the movements of all poles because it actually only introduces one additional degree of freedom.

The ROM interpolation based on the pole-residue realization, on the contrary, is capable of capturing the moving peak because it interpolates the positions of the poles. In this example, we use two MOR methods to build the nonparametric ROMs at the parameter samples: a system identification method as a MOR method (the `ssest` function in MATLAB) [32], and a balanced truncation (BT) method (the `balred` function in MATLAB). In both cases, the two ROMs for interpolation are built at $p = 10$ and $p = 32.5$ with the

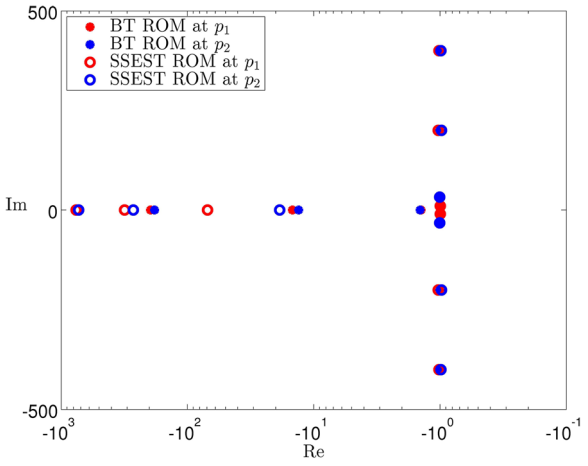

Fig. 10 The poles of the ROM in the pole-residue realization generated by different MOR methods: the balanced trancation (BT) method and the data-driven method "`ssest`" in MATLAB

dimension $k = 10$. The positions of the poles of the ROMs at these two parameters are shown in Fig. 10.

In Fig. 11a, b, we interpolate ROMs built by `balred` and `ssest`, respectively. The figures show that in this case, balanced truncation achieves better overall accuracy than "`ssest`". However, a more important observation is that, the errors at the interpolated parameter values are comparable to the errors at the interpolating parameter values. Therefore, the bigger error of the interpolated "`ssest`" pROM in the pole-residue realization results from the bigger error of the nonparametric "`ssest`" ROMs used for interpolation, rather than from interpolation itself. So in both cases, ROM interpolation based on the pole-residue realization gives satisfactory results.

Now we try to interpolate ROMs of different types. In Fig. 12a, we interpolate a BT ROM built at $p = 10$ and an `ssest` ROM built at $p = 32.5$. From Fig. 10, we can see that the non-dominant poles of the FOM are presented by significantly different poles of the ROM in the two different types of methods. However, their interpolation also gives accurate results as Fig. 12a shows.[1] Note that when we interpolate different types of ROMs, we should be particularly careful about pole-matching. If we skip the pole-matching procedure, we will get the result shown in Fig. 12b, which clearly presents the wrong evolution of peaks due to the interpolation of wrong poles.

Results for the footbridge model

In this section, we consider a large-scale footbridge model. The footbridge is located over the Dijle river in Mechelen (Belgium). It is about 31.354 m in length and a tuned mass damper is located in the center. The discretized footbridge model is

$$\begin{cases} \Big(\mathcal{K}_0 + i\omega\mathcal{C}_0 + (k_1 + i\omega c_1)\mathcal{K}_i - \omega^2\mathcal{M}_0\Big)X(\omega, k_1, c_1) = \mathcal{F}, \\ Y(\omega, k_1, c_1) = \mathcal{L}X(\omega, k_1, c_1), \end{cases} \tag{55}$$

[1] Numerical experiment shows that if we simply remove these "non-dominant poles" of the nonparametric ROMs before interpolation, the accuracy of the pROM becomes much worse.

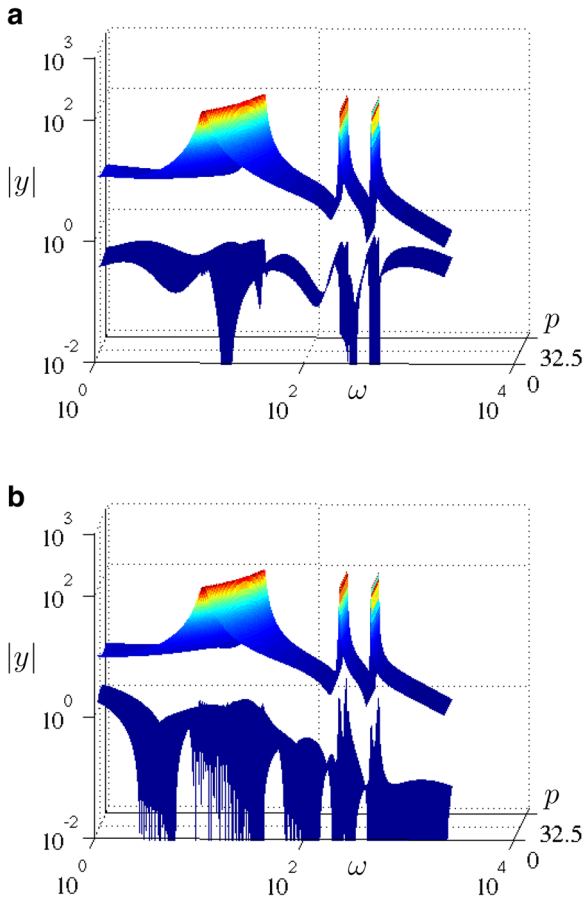

Fig. 11 Interpolation of ROMs of the same type: response surface and the absolute error. **a** Interpolation of BT ROMs. **b** Interpolation of `ssest` ROMs

where $\mathcal{K}_0$ and $\mathcal{M}_0$ are obtained from a finite element model with 25,962 degrees of freedom, $\mathcal{C}_0$ represents Rayleigh damping, $\mathcal{K}_1$ is a matrix with four non-zero entries that represents the interaction between the tuned mass damper and the footbridge, the input vector $\mathcal{F}$ represents a unit excitation at the center span, and the output vector $\mathcal{L}$ picks out the vibration at the center span. The model has two parameters, the stiffness of the damper k_1 and the damping coefficient of the damper c_1. To reduce the model, we apply the Krylov subspace method [33,34] on the first-order equivalent system

$$\begin{cases} \left(\begin{bmatrix} \mathcal{K}_0 + k_1\mathcal{K}_1 & 0 \\ 0 & \mathcal{I} \end{bmatrix} + i\omega \begin{bmatrix} \mathcal{C}_0 + c_1\mathcal{K}_1 & \mathcal{M}_0 \\ -\mathcal{I} & 0 \end{bmatrix} \right) \begin{bmatrix} X \\ i\omega X \end{bmatrix} = \begin{bmatrix} \mathcal{F} \\ 0 \end{bmatrix}, \\ y = [\mathcal{L}, 0] \begin{bmatrix} X \\ i\omega X \end{bmatrix}, \end{cases} \tag{56}$$

to obtain ROMs of the form (4) and then, we left multiply the system by E^{-1} to obtain the system of the form (11). The order of ROMs is set to 10.

In this example, we use four points (k_1, c_1) in the parameter space for interpolation: (10,000 N/m, 20 Ns/m), (20,000 N/m, 20 Ns/m), (10,000 N/m, 50 Ns/m) and (20,000 N/m, 50 Ns/m).

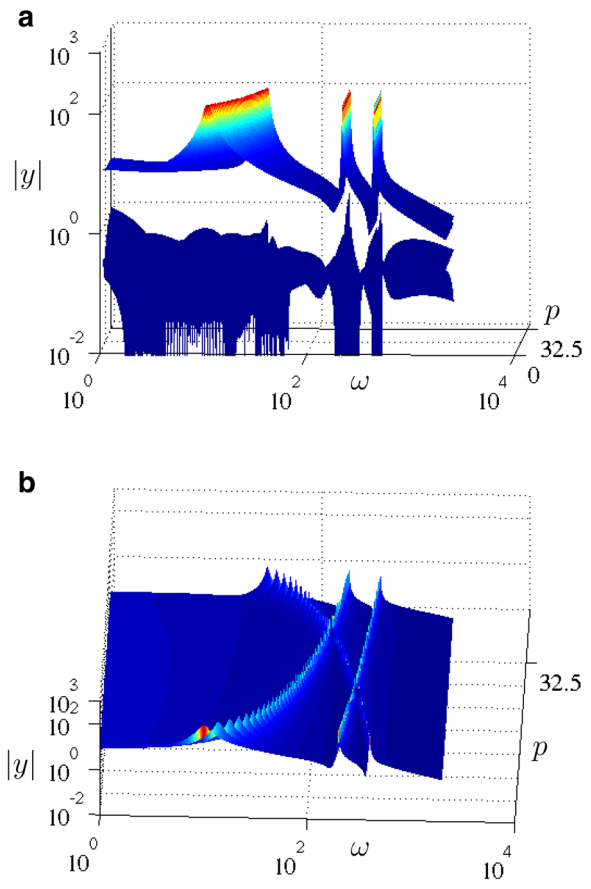

Fig. 12 Interpolation between ROMs of different types. **a** Interpolation between a BT ROM and an `ssest` ROM (with matched poles): response surfaces and the absolute errors. **b** Interpolation between a BT ROM and an `ssest` ROM (with mismatched poles): response surface

The FRFs corresponding to these four points are shown in Fig. 13. Using these four points, we conduct a 2-dimensional linear interpolation (function "`interp2`" in MATLAB) based on the pole-residue representation to get the ROM for $(k_1, c_1) = (15{,}000\,\mathrm{N/m}, 35\,\mathrm{Ns/m})$. Figure 14 shows that the interpolated ROM is accurate.

Discussion

Here are some further discussions:

- The advantages of the ROM interpolation method based on the pole-residue realization.
  - We do not need to know the explicit parametric expression of the system matrices because the parameters only vary in the interpolation of ROMs.
  - It does not assume the existence of the FOM and works well also with ROMs built by data-driven MOR methods.
  - It can even interpolate ROMs built by different MOR methods.
  - Its computational cost is relatively insensitive to the number of parameters.
  - It can deal with complex parameter dependency, e.g., nonlinear or nonaffine dependence. Since we employ an external interpolation method to handle the

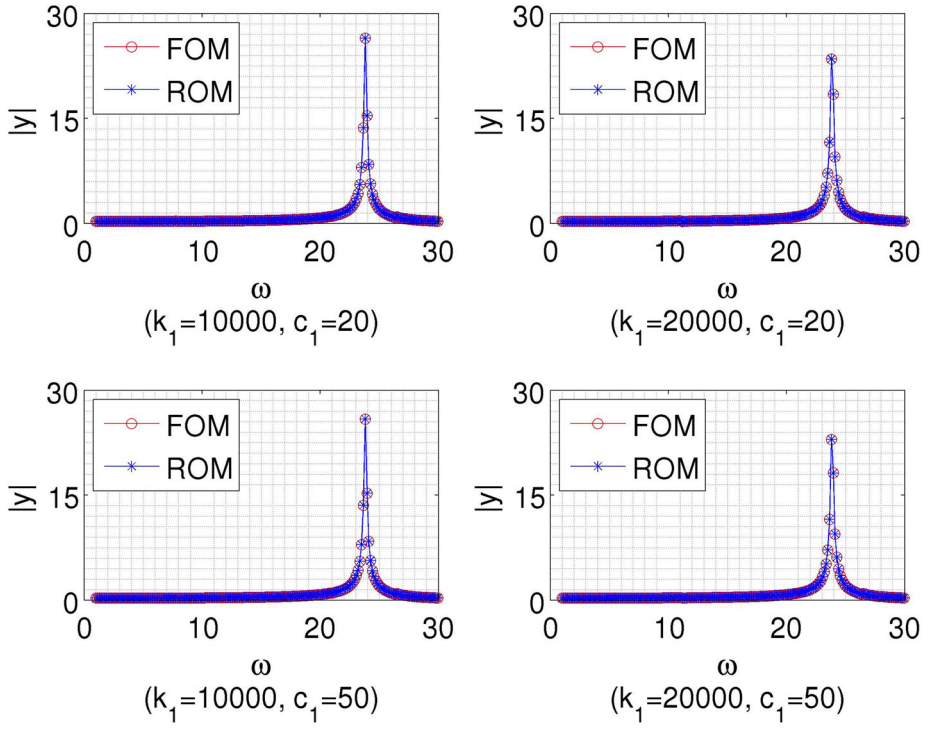

Fig. 13 The FRFs corresponding to the four interpolating points

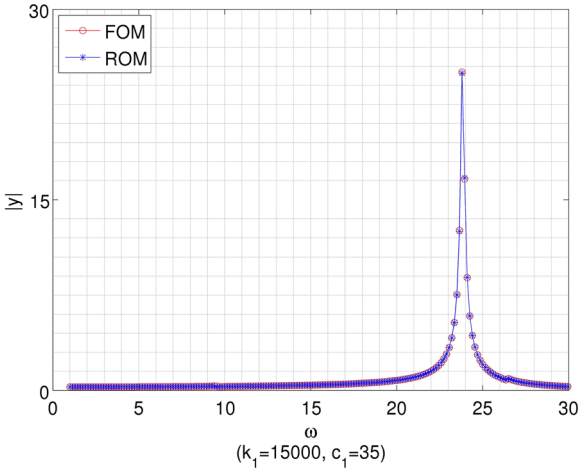

Fig. 14 The FRFs corresponding to the interpolated point

parameter dependency, the proposed method is effective as long as the parameter dependency can be locally captured well by the interpolation method.

- About stability. If we use linear interpolation for the pole-matching PMOR method, the stability is preserved since interpolating the poles in the left-half plane results in poles in the left-half plane. Using other interpolation methods, the stability can in general not be guaranteed. However, we can easily keep track of the interpolated poles in the pole-matching method. A straightforward solution is to use linear interpolation

instead when an interpolated pole lies outside the left-half plane. The Loewner PMOR interpolation method always preserves the stability because the FRF at any parameter is a weighted sum of FRFs corresponding to stable systems.

Conclusions

A pole-matching PMOR method that interpolates MIMO linear ROMs in the pole-residue realization was proposed. The method was tested for Loewner-type data-driven ROMs, balanced truncation ROMs, ROMs built by the system identification method `ssest`, and Krylov-type ROMs. For all these numerical tests, the method gives accurate results. Together with the PMOR method that interpolates MIMO ROMs in the Loewner representation, the importance of realization in ROM interpolation was demonstrated.

Abbreviations

BT: balanced truncation; FOM: full-order model; FRF: frequency response function; MIMO: multiple-input multiple-output; MOR: model order reduction; PMOR: parametric model order reduction; pROM: parametric reduced-order model; ROM: reduced-order model; SIMO: single-input multiple-output; SISO: single-input single-out; SVD: singular value decomposition.

Acknowledgements

The model of the footbridge was developed within the frame of the research project IWT 090137 ("TRICON Prediction and control of human-induced vibrations of civil engineering structures") and kindly provided as a numerical test case for the present research by Dr. Katrien Van Nimmen (katrien.vannimmen@kuleuven.be). We would also like to thank Professor Athanasios C. Antoulas for bestowing the MATLAB code for the Loewner framework and the "FOM" model on us generously.

Authors' contributions

All authors participated in the development of the methods and techniques. YY conducted the numerical experiment and wrote the draft paper. All authors did revisions and approved the final manuscript. All authors read and approved the final manuscript.

Funding

The research is funded by the Max Planck institute.

Availability of data and materials

The microthruster model and the "FOM" model is available at MOR Wiki: https://morwiki.mpi-magdeburg.mpg.de/morwiki/index.php/Main_Page. The footbridge model is not open to the public.

Competing interests

The authors declare that they have no competing interests.

Appendices

Appendix A: A Proof of Theorem 2

Proof

$$
\begin{aligned}
[E_{\mathrm{a}}(p)]_{i,j} &= \sum_{l=1}^{n_p} [E_l]_{i,j}\,\phi_l(p) = -\sum_{l=1}^{n_p} [\mathbb{L}_l]_{i,j}\,\phi_l(p) = -\sum_{l=1}^{n_p} \frac{v_i(p_l)r_j - \ell_i w_j(p_l)}{\mu_i - \lambda_j}\phi_l(p) \\
&= \frac{\left(\sum_{l=1}^{n_p} v_i(p_l)\phi_l(p)\right) r_j - \ell_i \left(\sum_{l=1}^{n_p} w_j(p_l)\phi_l(p)\right)}{\mu_i - \lambda_j} \\
&= -\left[\widetilde{\mathbb{L}}(p)\right] = [E_{\mathrm{b}}(p)]_{i,j},
\end{aligned}
$$

where $\widetilde{\mathbb{L}}(p)$ denotes the Loewner matrix computed with the interpolated data at p, namely (47). This proves $E_{\mathrm{a}}(p) = E_{\mathrm{b}}(p)$. Similarly, we prove $A_{\mathrm{a}}(p) = A_{\mathrm{b}}(p)$.

$$
B_{\mathrm{a}}(p) = \sum_{l=1}^{n_p} B_l\phi_l(p) = \sum_{l=1}^{n_p} V_l\phi_l(p) = \left[\sum_{l=1}^{n_p} v_1(p_l)\phi_l(p), \ldots, \sum_{l=1}^{n_p} v_{n_L}(p_l)\phi_l(p)\right] = B_{\mathrm{b}}(p).
$$

Similarly, we can prove $C_a(p) = C_b(p)$.

Therefore, the pROM

$$(E_a(p), A_a(p), B_a(p), C_a(p))$$

and the pROM

$$(E_b(p), A_b(p), B_b(p), C_b(p))$$

are equal. □

Appendix B: A Proof of Theorem 3

Proof Define $I_l = [0_{n\times n(l-1)}, I_{n\times n}, 0_{n\times n(n_p-1)}]^T$. Then,

$$s_l\mathbb{L} - \mathbb{L}_{\sigma l} = Y\Sigma_H X_H I_l = I_l^T Y_V \Sigma_V X.$$

Since Y is orthonormal,

$$\Sigma_H X_H I_l = Y^T I_l^T Y_V \Sigma_V X.$$

Therefore,

$$s_l\mathbb{L} - \mathbb{L}_{\sigma l} = YY^T I_l^T Y_V \Sigma_V X.$$

Compute the SVD of $Y^T I_l^T Y_V \Sigma_V$ as

$$Y^T I_l^T Y_V \Sigma_V = Y_l' \Sigma_l' X_l',$$

and the SVD of $s_l\mathbb{L} - \mathbb{L}_{\sigma l}$ is

$$\text{svd}(s_l\mathbb{L}_l - \mathbb{L}_{\sigma l}) = (YY_l')\Sigma_l'(X_l'X)$$

because both YY_l' and $X_l'X$ are orthonormal and Σ' is diagonal with non-negative diagonal entries. Therefore,

$$\text{rowspan}\{X_l\} = \text{rowspan}\{X_l'X\} \subseteq \text{rowspan}\{X\},$$
$$\text{colspan}\{Y_l\} = \text{colspan}\{YY_l'\} \subseteq \text{colspan}\{Y\}.$$

□

Received: 1 February 2019 Accepted: 31 July 2019
Published online: 08 August 2019

References

1. Feldmann P, Freund RW. Efficient linear circuit analysis by Padé approximation via the Lanczos process. IEEE Trans Comput Aided Design Integr Circuits Syst. 1995;14:639–49.
2. Odabasioglu A, Celik M, Pileggi LT. PRIMA: passive reduced-order interconnect macromodeling algorithm. In: ICCAD '97: Proceedings of the 1997 IEEE/ACM international conference on computer-aided design. Washington, DC: IEEE Computer Society; 1997. p. 58–65. https://doi.org/10.1109/ICCAD.1997.643366.
3. Feng L, Yue Y, Banagaaya N, Meuris P, Schoenmaker W, Benner P. Parametric modeling and model order reduction for (electro-)thermal analysis of nanoelectronic structures. J Math Ind. 2016;6(1):1–10. https://doi.org/10.1186/s13362-016-0030-8.
4. Meerbergen K. Fast frequency response computation for Rayleigh damping. Int J Numer Methods Eng. 2008;73(1):96–106. https://doi.org/10.1002/nme.2058.

5. Han JS, Rudnyi EB, Korvink JG. Efficient optimization of transient dynamic problems in MEMS devices using model order reduction. J Micromech Microeng. 2005;15(4):822–32. https://doi.org/10.1088/0960-1317/15/4/021.
6. Li S, Yue Y, Feng L, Benner P, Seidel-Morgenstern A. Model reduction for linear simulated moving bed chromatography systems using Krylov-subspace methods. AIChE J. 2014;60(11):3773–83.
7. Benner P, Gugercin S, Willcox K. A survey of model reduction methods for parametric systems. SIAM Rev. 2015;57(4):483–531. https://doi.org/10.1137/130932715.
8. Baur U, Beattie CA, Benner P, Gugercin S. Interpolatory projection methods for parameterized model reduction. SIAM J Sci Comput. 2011;33(5):2489–518. https://doi.org/10.1137/090776925.
9. Rozza G, Huynh DBP, Patera AT. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. Arch Comput Methods Eng. 2008;15(3):229–75. https://doi.org/10.1007/s11831-008-9019-9.
10. Amsallem D, Farhat C. Interpolation method for the adaptation of reduced-order models to parameter changes and its application to aeroelasticity. AIAA J. 2008;46:1803–13.
11. Panzer H, Mohring J, Eid R, Lohmann B. Parametric model order reduction by matrix interpolation. at-Automatisierungstechnik. 2010;58(8):475–84.
12. Amsallem D, Farhat C. An online method for interpolating linear parametric reduced-order models. SIAM J Sci Comput. 2011;33(5):2169–98.
13. Baur U, Benner P. Modellreduktion für parametrisierte Systeme durch balanciertes Abschneiden und Interpolation (Model Reduction for Parametric Systems Using Balanced Truncation and Interpolation). at-Automatisierungstechnik. 2009;57(8):411–20.
14. Baur U, Benner P, Greiner A, Korvink JG, Lienemann J, Moosmann C. Parameter preserving model reduction for MEMS applications. Math Comput Model Dyn Syst. 2011;17(4):297–317.
15. Mayo AJ, Antoulas AC. A framework for the solution of the generalized realization problem. Linear Algebra Appl. 2007;425(2–3):634–62.
16. Fu Z-F, He J. Modal Analysi. Waltham: Butterworth-Heinemann; 2001.
17. Kutz JN, Brunton SL, Brunton BW. Dynamic mode decomposition: data-driven modeling of complex systems. Philadelphia: Society of Industrial and Applied Mathematics; 2016. https://doi.org/10.1137/1.9781611974508.
18. Antoulas AC. Approximation of large-scale dynamical systems. Advances in design and control, vol. 6. Philadelphia: SIAM Publications; 2005. https://doi.org/10.1137/1.9780898718713.
19. Feng L, Rudnyi EB, Korvink JG. Preserving the film coefficient as a parameter in the compact thermal model for fast electro-thermal simulation. IEEE Trans Comput Aided Design Integr Circuits Syst. 2005;24(12):1838–47.
20. Yue Y, Meerbergen K. Using Krylov–Padé model order reduction for accelerating design optimization of structures and vibrations in the frequency domain. Int J Numer Methods Eng. 2012;90(10):1207–32.
21. Ionita AC, Antoulas AC. Data-driven parametrized model reduction in the Loewner framework. SIAM J Sci Comput. 2014;36(3):984–1007. https://doi.org/10.1137/130914619.
22. Yue Y, Feng L, Benner P. 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. https://doi.org/10.1109/NEMO.2018.8503114.
23. Gugercin S, Antoulas AC, Beattie C. $\mathcal{H}_2$ model reduction for large-scale linear dynamical systems. SIAM J Matrix Anal Appl. 2008;30(2):609–38. https://doi.org/10.1137/060666123.
24. Benner P, Stykel T. Model order reduction for differential-algebraic equations: a survey. In: Ilchmann A, Reis T, editors. Surveys in differential-algebraic equations IV. Differential-algebraic equations forum. Cham: Springer; 2017. p. 107–60. https://doi.org/10.1007/978-3-319-46618-7_3.
25. Golub GH, van Van Loan CF. Matrix computations. 3rd ed. London: The Johns Hopkins University Press; 1996.
26. Friswell MI. Candidate reduced order models for structural parameter estimation. J Vib Acoust. 1990;1:93–7.
27. Martins N, Lima LTG, Pinto HJCP. Computing dominant poles of power system transfer functions. IEEE Trans Power Syst. 1996;11(1):162–70. https://doi.org/10.1109/59.486093.
28. Zhang Z, Hu X, Cheng CK, Wong N. A block-diagonal structured model reduction scheme for power grid networks. In: Design, automation & test in Europe (DATE), Grenoble, France. 2011. p. 1–6. https://doi.org/10.1109/DATE.2011.5763014.
29. Yue Y, Meerbergen K. Accelerating optimization of parametric linear systems by model order reduction. SIAM J Optim. 2013;23(2):1344–70. https://doi.org/10.1137/120869171.
30. Chahlaoui Y, Van Dooren P. A collection of benchmark examples for model reduction of linear time invariant dynamical systems. Technical report 2002–2, SLICOT Working Note. 2002; http://slicot.org/20-site/126-benchmark-examples-for-model-reduction. Accessed 06 Aug 2019.
31. Ferranti F, Knockaert L, Dhaene T. Passivity-preserving parametric macromodeling by means of scaled and shifted state-space systems. IEEE Trans Microw Theory Tech. 2011;59(10):2394–403.
32. Ljung L. System identification: theory for the user. 2nd ed. Englewood Cliffs: Prentice Hall Information and System Sciences Series. Prentice Hall PTR; 1999.
33. Feldman P, Freund RW. Efficient linear circuit analysis by Padé approximation via the Lanczos process. IEEE Trans Comput Aided Des. 1995;14(5):639–49. https://doi.org/10.1109/43.384428.
34. Bai Z. Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. Appl Numer Math. 2002;43(1–2):9–44.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.