

Peter Benner and Lihong Feng

3 Model order reduction based on moment-matching

Abstract: This is a survey of model order reduction (MOR) methods based on moment-matching. Moment-matching methods for linear non-parametric and parametric systems are reviewed in detail. Extensions of moment-matching methods to nonlinear systems are also discussed. Efficient algorithms for computing the reduced-order models (ROMs) are presented.

Keywords: moment-matching, linear time-invariant systems, parametric systems, error estimation, greedy-type algorithm

MSC 2010: 37M05, 65P99, 65L80

3.1 Introduction

In this chapter, we focus on linear time-invariant (LTI) systems

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= A(\mu)\mathbf{x}(t, \mu) + B(\mu)\mathbf{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu)\mathbf{x}(t, \mu) + D(\mu)\mathbf{u}(t), \end{aligned} \quad (3.1)$$


and mildly nonlinear systems

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= \mathbf{f}(\mathbf{x}(t, \mu), \mu) + B(\mu)\mathbf{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu)\mathbf{x}(t, \mu) + D(\mu)\mathbf{u}(t), \end{aligned}$$

with and without parameters. Here $\mathbf{x}(t, \mu) \in \mathbb{R}^n$ is the state vector, and its entries are called state variables. n is often referred to as the *order* of the system. The vector $\mu \in \mathbb{R}^m$ includes all of the geometrical and physical parameters. The system matrices $E(\mu), A(\mu) \in \mathbb{R}^{n \times n}$, and $B(\mu) \in \mathbb{R}^{n \times n_i}$, $C(\mu) \in \mathbb{R}^{n_o \times n}$, $D(\mu) \in \mathbb{R}^{n_o \times n_i}$ may depend on the parameters. The vector $\mathbf{f}(\mathbf{x}, \mu) \in \mathbb{R}^n$ is a nonlinear function. The system in (3.1) is called the *state-space* representation of the system. It may result from the spatial discretization of partial differential equations (PDEs) describing certain processes like fluid dynamics, temperature distribution in devices, electric circuits, etc.

For most MOR methods, the term $D(\mu)\mathbf{u}(t)$ remains unchanged during the process of MOR. For simplicity, we therefore assume that $D(\mu) = 0$, a zero matrix. There are

Peter Benner, Lihong Feng, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg, Germany, e-mails: benner@mpi-magdeburg.mpg.de, feng@mpi-magdeburg.mpg.de

Open Access. © 2021 Peter Benner and Lihong Feng, published by De Gruyter.  This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

n_I input terminals and n_O output terminals. When $n_I = n_O = 1$, the system is called a *single-input single-output (SISO) system*. Otherwise, if $n_I, n_O > 1$, it is called a *multi-input multi-output (MIMO) system*.

The basic idea of (P)MOR methods is as follows. Find a low-dimensional trial subspace S_1 which well approximates the manifold where the state vector $\mathbf{x}(t, \mu)$ resides. $\mathbf{x}(t, \mu)$ is approximated by a vector $\hat{\mathbf{x}}(t, \mu)$ in S_1 , which causes a residual of the state equation. The reduced-order model (ROM) is obtained by a (Petrov)–Galerkin projection of the residual onto a test subspace S_2 . In particular, one computes an orthonormal matrix $V = (v_1, v_2, \dots, v_r)$ whose columns span S_1 . The ROM is derived by the following two steps.

1. By replacing $\mathbf{x}(t, \mu)$ in (3.1) with $V\mathbf{z}(t, \mu)$, we obtain

$$\begin{aligned} E(\mu) \frac{dV\mathbf{z}(t, \mu)}{dt} &\approx A(\mu)V\mathbf{z}(t, \mu) + B(\mu)\mathbf{u}(t), \\ \mathbf{y}(t) &\approx C(\mu)V\mathbf{z}(t, \mu). \end{aligned} \quad (3.2)$$

2. Notice that the equations in (3.1) do not hold any longer. Therefore, we can only use “ \approx ” in (3.2). Denote the residual as $\mathbf{e}(t, \mu) = AV\mathbf{z}(t, \mu) + B(\mu)\mathbf{u}(t) - E(\mu) \frac{dV\mathbf{z}(t, \mu)}{dt}$, which in general is nonzero over the whole vector space \mathbb{R}^n . However, it is possible to force $\mathbf{e} = \mathbf{0}$ in a properly chosen subspace S_2 of \mathbb{R}^n . If we have computed a matrix $W \in \mathbb{R}^{n \times r}$, whose columns span S_2 , then $\mathbf{e} = \mathbf{0}$ in S_2 means \mathbf{e} is orthogonal to each column in W , i. e. $W^T \mathbf{e} = \mathbf{0} \iff W^T E \frac{dV\mathbf{z}(t, \mu)}{dt} = W^T AV\mathbf{z}(t, \mu) + W^T B\mathbf{u}(t)$. Finally, we obtain the ROM

$$\begin{aligned} \hat{E}(\mu) \frac{d\mathbf{z}(t, \mu)}{dt} &= \hat{A}(\mu)\mathbf{z}(t, \mu) + \hat{B}(\mu)\mathbf{u}(t), \\ \hat{\mathbf{y}}(t, \mu) &= \hat{C}\mathbf{z}(t, \mu), \end{aligned} \quad (3.3)$$

where $\hat{E}(\mu) = W^T E(\mu)V \in \mathbb{R}^{r \times r}$, $\hat{A}(\mu) = W^T A(\mu)V \in \mathbb{R}^{r \times r}$, $\hat{B}(\mu) = W^T B(\mu) \in \mathbb{R}^{r \times n_I}$, $\hat{C}(\mu) = C(\mu)V \in \mathbb{R}^{n_O \times r}$. $\mathbf{z}(\mu) \in \mathbb{R}^r$ is a vector of length $r \ll n$. Then $\mathbf{x}(t, \mu)$ can be approximated by $\mathbf{x}(t, \mu) \approx V\mathbf{z}(t, \mu)$. The system in (3.3) is referred to as the *reduced-order model* (ROM), since it is of much smaller order than the original system in (3.1), i. e. $r \ll n$. The ROM can then replace the original system for fast simulation.

MOR methods differ in computing the two matrices W and V . One common goal of all methods is that the input-output behavior of the ROM should be sufficiently “close” to that of the original model. The error between the transfer functions (see (3.6)) is also used to measure the accuracy of the ROM.

Moment-matching relates to a class of methods which construct the ROM by building the projection matrices W, V from the system information in the frequency domain. The early moment-matching methods are only applicable to linear non-parametric systems. Later on, these methods were extended to linear parametric systems. Based on nonlinear system theory [52], multi-moment-matching methods based on variational analysis were proposed and are successful in reducing weakly

nonlinear systems. In contrast to the snapshot based time-domain methods, e. g., proper orthogonal decomposition or the reduced basis method, moment-matching methods can be considered as frequency-domain methods, and are independent of the inputs. Therefore, these methods are robust for systems with varying inputs. The rest of this chapter is organized as follows. In Section 3.2, we introduce moment-matching methods for linear non-parametric systems, where methods based on rational interpolation are particularly discussed. The extension of those methods to linear parametric systems is introduced in Section 3.3. Methods based on moment-matching and multi-moment-matching for nonlinear systems are reviewed in Section 3.4, and their extension to parametric nonlinear systems is discussed in Section 3.5. Conclusions are drawn in the end.

3.2 Moment-matching for linear non-parametric systems

This section reviews moment-matching methods for linear non-parametric systems, so that the vector of parameters μ can be dropped from the system (3.1). Among the early works of moment-matching MOR, the method of Asymptotic Waveform Evaluation (AWE) in [48] was shown to be able to reduce large-scale interconnected electrical circuit models, which stimulated broad interests in this kind of methods. The AWE method tries to find a Padé approximation of the transfer function $H(s)$, which can be computed much more quickly than computing $H(s)$ itself.

Transfer function

For all the methods introduced in this chapter, the transfer function of the system is used to either derive the ROM, or to perform the error estimation. The transfer function of the system in (3.1) is the input/output relation of the system in the frequency domain. By applying the Laplace transform to both sides of the equations in (3.1), we obtain

$$sEX(s) - E\mathbf{x}(0) = AX(s) + BU(s), \quad (3.4)$$

$$Y(s) = CX(s). \quad (3.5)$$

Here, $X(s)$ is the Laplace transform of $\mathbf{x}(t)$, and $\mathbf{x}(0)$ is the initial state of the system. Assuming that $\mathbf{x}(0) = \mathbf{0}$, we obtain the expression for the transfer function

$$H(s) = Y(s)/U(s) = C(sE - A)^{-1}B, \quad (3.6)$$

where the right division “/” has to be understood in a formal way for MIMO systems. For a SISO system, the transfer function $H(s)$ is a scalar function. The Padé approximation of a scalar function can be defined as follows.

Padé approximation

The Padé approximation of a function $H(s)$ is a rational function $H_{p,q}(s)$ whose Taylor series at $s = 0$ agrees with that of $H(s)$ in at least the first $p + q + 1$ terms [21].

For a MIMO system, the transfer function $H(s)$ is a matrix function and each entry of it can be approximated by the above Padé approximation. For clarity of explanation, we use a SISO system as an example to briefly describe the method.

From the definition of the Padé approximation, we know that, if $H(s) = H_{p,q}(s_0 + \sigma) = P_p(\sigma)/Q_q(\sigma)$ is a Padé approximation of the transfer function $H(s_0 + \sigma)$, then we have

$$H_{p,q}(s_0 + \sigma) = H(s_0 + \sigma) + O(\sigma^{p+q+1}). \quad (3.7)$$

The derivatives of $H(s_0 + \sigma)$ at $\sigma = 0$ are actually the derivatives of $H(s)$ at $s = s_0$, they are also the moments $m_i(s_0)$, $i = 0, 1, \dots$, (see the definition in Section 3.2.1.1) of the transfer function. By definition, the Padé approximation $H_{p,q}(s_0 + \sigma)$ matches the first $p + q + 1$ moments of the transfer function.

If the coefficients of the two polynomials $P_p(\sigma)$ and $Q_q(\sigma)$ in $H_{p,q}(s_0 + \sigma)$ are computed, then $H_{p,q}(s_0 + \sigma)$ is obtained. The coefficients can be obtained by solving two groups of equations which are derived from equating the coefficients of the Taylor series expansion (at $\sigma = 0$) on both sides of (3.7).

Since the moments are the coefficients of the Taylor series expansion of $H(s_0 + \sigma)$ at $\sigma = 0$, they are involved in solving the equations to obtain the coefficients of $P_p(\sigma)$ and $Q_q(\sigma)$. However, in the AWE method, the moments are computed explicitly, which can cause serious numerical instability.

In order to overcome the numerical instability of AWE, a more robust method “Padé via Lanczos” (PVL) [21] (see also [32]) was proposed. PVL also computes the Padé approximation of $H(s) = H(s_0 + \sigma)$, however, the moments of $H(s)$ do not have to be computed explicitly. Instead, an orthonormal basis of the subspace spanned by the moment vectors is computed, which constitutes the projection matrix V , and the projection matrix W is also computed simultaneously. Both of them are computed by the nonsymmetric Lanczos process. It is proved in [21] that the transfer function of the ROM produced by W and V is the Padé approximation of the original transfer function $H(s)$. The PVL method avoids explicit computation of the moments, and therefore avoids the possible numerical instability.

Unfortunately, PVL does not necessarily preserve passivity of the original system, which is a problem in some engineering applications, especially in Integrated Circuit (IC) design. For this target, the method “Passive and Reduced-order Interconnect Macromodeling Algorithm” (PRIMA) [45] was proposed. The resulting ROM preserves the passivity of the original system, under certain assumptions on the system matrices. The trade-off is that only half the number of moments can be matched by PRIMA as compared to PVL, if the matrix V in both methods expands the same subspace. Such an approximation of the transfer function is called a *Padé-type approximation*.

The rational interpolation method proposed in [35] extended the Padé-approximation method and the Padé-type-approximation method based on a single expansion point to the case of multiple expansion points. All these methods can be called moment-matching methods, because all the ROMs match the moments of the original system to different extents. For survey papers on moment-matching model reduction; see [4, 31] and [3, 35].

Moment-matching MOR methods try to derive a ROM whose transfer function matches the moments of the transfer function of the original system. Generally speaking, the more moments matched, the more accurate the ROM will be. In the following, we first introduce the definition of the moments and moment vectors, then we show how to compute the matrices W and V based on moment-matching.

3.2.1 Basic idea

3.2.1.1 Moments and moment vectors

If we expand the transfer function $H(s)$ into its Taylor series about an expansion point s_0 so that $s_0E - A$ is a nonsingular matrix,

$$\begin{aligned} H(s) &= C^T [(s - s_0 + s_0)E - A]^{-1} B \\ &= C^T [(s - s_0)E + (s_0E - A)]^{-1} B \\ &= C^T [I + (s_0E - A)^{-1}E(s - s_0)]^{-1} (s_0E - A)^{-1} B \\ &= \sum_{i=0}^{\infty} \underbrace{C^T [-(s_0E - A)^{-1}E]^i (s_0E - A)^{-1} B}_{=: m_i(s_0)} (s - s_0)^i, \end{aligned} \quad (3.8)$$

and if the system is a SISO system, then $m_i(s_0)$, $i = 0, 1, 2, \dots$, are called the moments of the transfer function $H(s)$. If the system is a MIMO system, then $m_i(s_0)$, $i = 0, 1, 2, \dots$, are matrices and they are called block moments [45]. In the field of circuit design, the entry in the j th row, k th column of $m_i(s_0)$ is called the i th moment of the current that flows into port j when the voltage source at port k is the only nonzero source [45]. Analogies exist for other application areas, such as mechanics. In this chapter, we only consider $m_i(s_0)$ as a whole, and do not consider its entries individually. This means, when we talk about moments of the transfer function, we mean $m_i(s_0)$, $i = 0, 1, \dots$, which refers either to the moments of a SISO system or to the block moments of a MIMO system.

From (3.4) and (3.8), it is straightforward to obtain the corresponding Taylor series expansion of $X(s)$,

$$X(s) = \sum_{i=0}^{\infty} [-(s_0E - A)^{-1}E]^i (s_0E - A)^{-1} B U(s) (s - s_0)^i. \quad (3.9)$$

Here, we call the vectors $[-(s_0E - A)^{-1}E]^i (s_0E - A)^{-1} B$, $i = 0, 1, \dots$, moment vectors which are to be used to compute the projection matrices W , V . Notice that when the system

in (3.1) has multiple inputs, i. e. $n_I > 1$, then $[-(s_0E - A)^{-1}E]^i (s_0E - A)^{-1}B$, $i = 0, 1, \dots$, are matrices rather than vectors. For simplicity, we still call them moment vectors.

3.2.1.2 Computation of the projection matrices W and V

Computation of V

Approximating $X(s)$ by the truncated series in (3.9) means that $X(s) \approx VZ(s)$. The columns of $V \in \mathbb{R}^{n \times r}$ constitute an orthonormal basis of S_1 , which is a subspace spanned by the moment vectors in the truncated series. After inverse Laplace transform, we obtain the corresponding approximation of $\mathbf{x}(t)$ in S_1 , i. e. $\mathbf{x}(t) \approx V\mathbf{z}(t)$, where $\mathbf{z}(t)$ is the inverse Laplace transform of $Z(s)$. This means that $\mathbf{x}(t)$ in the time domain can be approximated by $V\mathbf{z}(t)$. Usually, the more moment vectors included in S_1 , the more accurate the approximation $V\mathbf{z}(t)$ will be. However, in order to keep the ROM small, we usually choose a small number of moment vectors starting from $i = 0$, i. e. the columns of the orthonormal matrix V span the subspace

$$\text{range}\{V\} = \text{span}\{\tilde{B}(s_0), \tilde{A}(s_0)\tilde{B}(s_0), \dots, \tilde{A}^{q-1}(s_0)\tilde{B}(s_0)\}, \quad (3.10)$$

where $\tilde{A}(s_0) = (s_0E - A)^{-1}E$, $\tilde{B}(s_0) = (s_0E - A)^{-1}B$ and $q \ll n$.

Computation of W

To obtain the ROM, we also need to compute the (Petrov-)Galerkin projection matrix W . The columns of the matrix W span the subspace below, i. e.

$$\text{range}\{W\} = \text{span}\{\tilde{C}(s_0), \tilde{A}_c(s_0)\tilde{C}(s_0), \dots, \tilde{A}_c^{q-1}(s_0)\tilde{C}(s_0)\}, \quad (3.11)$$

where $\tilde{A}_c(s_0) = (s_0E - A)^{-T}E^T$, $\tilde{C}(s_0) = (s_0E - A)^{-T}C^T$. Note that the two subspaces in (3.10) and (3.11) are actually two Krylov subspaces $K_q(\tilde{A}(s_0), \tilde{B}(s_0))$ and $K_q(\tilde{A}_c(s_0), \tilde{C}(s_0))$, respectively. Moment-matching methods based on computing W , V from Krylov subspaces are often called *Krylov-based methods*. If the above two matrices W and V are used to obtain the ROM (3.3), the transfer function of the ROM matches the first $2q$ moments of the transfer function of the original model [35]. We summarize this in the following theorem.

Theorem 3.1. *If V and W span the subspaces in (3.10) and (3.11), respectively, then the transfer function $\hat{H}(s) = \hat{C}(s\hat{E} + \hat{A})^{-1}\hat{B}$ of the ROM (3.3) matches the first $2q$ moments of the transfer function of the original system, i. e.*

$$m_i(s_0) = \hat{m}_i(s_0), \quad i = 0, 1, \dots, 2q - 1,$$

where $\hat{m}_i(s_0) = \hat{C}[-(s_0\hat{E} - \hat{A})\hat{E}]^{-i}(s_0\hat{E} - \hat{A})^{-1}\hat{B}$, $i = 0, 1, \dots, 2q - 1$, are the i th-order moments of \hat{H} .

Note that in order to ensure the projector property of VW^T , one also needs to enforce the bi-orthogonality condition $W^T V = I$ (assuming here that the subspace basis ma-

trices V, W are formed over \mathbb{R}). The moment-matching MOR method PRIMA [45] uses $W = V$. In this case, only q moments of the transfer function are matched.

The orthonormal matrices W and V in (3.10), (3.11) can be computed by rational Krylov subspace algorithms (rational Lanczos algorithm, rational Arnoldi algorithm) [35]. Only (sparse) matrix factorizations, (sparse) forward/backward solves, and matrix-vector multiplications are used in these algorithms, such that the complexity of the moment-matching MOR is in $O(nq^2)$ for sparse matrices E, A .

3.2.2 Stability

In general, the moment-matching methods do not preserve stability of the original system. Only for systems with special structures, there exist several approaches based on Galerkin projection where the ROM are guaranteed to be stable and passive; see, e. g., [45]. For details on passivity of LTI systems, we refer to Chapter 5 of this volume. The passivity preservation of the moment-matching method for RLC circuits can be mathematically described as follows [45].

Theorem 3.2. *If the system matrices E and A satisfy $E^T + E \geq 0$ and $A^T + A \leq 0$, respectively, and if $C = B$, then the ROM obtained by Galerkin projection, i. e., $W = V$ preserves the passivity of the original system in (3.1).*

Stability is naturally guaranteed by passivity, therefore the ROM obtained by moment-matching with Galerkin projection preserves stability as well. Benefiting from the preservation of passivity and low computational complexity, the moment-matching method is very popular in circuit simulation and in micro-electro-mechanical systems (MEMS) simulation as well.

3.2.3 Multiple expansion points

The accuracy of the moment-matching methods depends not only on the number of moments matched, but also on the expansion points. Since the Taylor expansion in (3.8) is only accurate within a certain radius around the expansion point s_0 , the ROM becomes inaccurate beyond this radius.

To increase the accuracy of a single-point expansion, one may use more than one expansion point. Moment-matching by multi-point expansion is also known as rational interpolation [35]. For example, if using a set of q distinct expansion points $\{s_1, \dots, s_q\}$, the ROM obtained by, e. g.,

$$\begin{aligned} \text{range}\{V\} &= \text{span}\{\tilde{B}(s_1), \dots, \tilde{B}(s_l)\}, \\ \text{range}\{W\} &= \text{span}\{\tilde{C}(s_1), \dots, \tilde{C}(s_l)\}, \end{aligned}$$

matches the first two moments $m_0(s_i), m_1(s_i)$ at each $s_i, i = 1, \dots, l$ [35]. Here, $\tilde{B}(s_i) = (s_i E - A)^{-1} B$, $\tilde{C}(s_i) = (s_i E - A)^{-T} C^T, i = 1, \dots, l$.

More generally, if we use

$$\text{range}\{V\} = \text{span}\{\tilde{B}(s_1), \dots, \tilde{A}^{q_1-1}(s_1)\tilde{B}(s_1), \dots, \tilde{B}(s_l), \dots, \tilde{A}^{q_l-1}(s_l)\tilde{B}(s_l)\}, \quad (3.12)$$

$$\text{range}\{W\} = \text{span}\{\tilde{C}(s_1), \dots, \tilde{A}_c^{q_1-1}(s_1)\tilde{C}(s_1), \dots, \tilde{C}(s_l), \dots, \tilde{A}_c^{q_l-1}(s_l)\tilde{C}(s_l)\}, \quad (3.13)$$

where $\tilde{A}(s_k) = (s_k E - A)^{-1}E$, $\tilde{A}_c(s_k) = (s_k E - A)^{-T}E^T$, $k = 1, \dots, l$, then we have the following moment-matching property.

Theorem 3.3 ([35]). *If*

$$\text{range}\{V\} \supseteq \text{span}\{\tilde{B}(s_1), \dots, \tilde{A}^{q_1-1}(s_1)\tilde{B}(s_1), \dots, \tilde{B}(s_l), \dots, \tilde{A}^{q_l-1}(s_l)\tilde{B}(s_l)\},$$

and

$$\text{range}\{W\} \supseteq \text{span}\{\tilde{C}(s_1), \dots, \tilde{A}_c^{q_1-1}(s_1)\tilde{C}(s_1), \dots, \tilde{C}(s_l), \dots, \tilde{A}_c^{q_l-1}(s_l)\tilde{C}(s_l)\},$$

then the transfer function $\hat{H}(s) = \hat{C}(s\hat{E} + \hat{A})^{-1}\hat{B}$ of the ROM (3.3) matches the first $2q_k$ moments of the transfer function of the original system at each expansion point s_k , i. e.

$$m_i(s_k) = \hat{m}_i(s_k), \quad i = 0, 1, \dots, 2q_k - 1, \quad k = 1, \dots, l,$$

where $\hat{m}_i(s_k) = \hat{C}[-(s_k\hat{E} - \hat{A})\hat{E}]^{-i}(s_k\hat{E} - \hat{A})^{-1}\hat{B}$, $i = 0, 1, \dots, 2q_k - 1$, are the i th-order moments of \hat{H} at s_k .

Given expansion points s_1, \dots, s_l , Algorithm 3.1 presents a procedure for computing the projection matrix V in (3.12).

The matrix W can also be computed using Algorithm 3.1, only by replacing $\tilde{B}(s_k)$ with $\tilde{C}(s_k)$, and $\tilde{A}(s_k)$ with $\tilde{A}_c(s_k)$. Algorithm 3.1 is also applicable to SISO systems, as we can see from Step 7. However, for SISO systems, the algorithm can be further simplified, and the two matrices W , V can be easily computed in parallel. Algorithm 3.2 is a version for SISO systems. In fact, the computed V , $W \in \mathbb{R}^{n \times r}$ from either Algorithm 3.1, or Algorithm 3.2, are not bi-orthogonal, which is not required by the moment-matching Theorem 3.3. However, for systems with $E = I$, the identity matrix, it is preferred that the reduced matrix $\hat{E} = I_r$, the identity matrix of dimension of r . Then we can use the transform $W \leftarrow W(V^T W)^{-1}$ to obtain a new W , so that $\hat{E} = W^T E V = W^T V = I_r$. In the final steps of both algorithms, we need to orthogonalize the columns of the intermediate matrices using the modified Gram–Schmidt process, which is an algorithm for orthogonalizing any given group of vectors. The details of the algorithm are given in Algorithm 3.3. The finally obtained orthogonal vectors are actually orthonormal, i. e., their norms are all 1. The number of orthogonal vectors are $\tilde{l} \leq l$, because once deflation ($\|a_k\|_2 \leq \varepsilon$) in Step 7 occurs, \tilde{l} will not be increased.

Remark 3.1. Let $\text{size}(M, 2)$ be the MATLAB notation for the number of columns in a matrix M . Then it could happen that $\text{size}(V, 2) \neq \text{size}(W, 2)$. In this situation, more computations should be done as follows. Denote $r_V = \text{size}(V, 2)$, $r_W = \text{size}(W, 2)$, if

Algorithm 3.1: Compute V in (3.12) for a non-parametric MIMO system (3.1).

Input: System matrices E, A, B, C , expansion points s_1, \dots, s_l .

Output: Projection matrix V .

```

1: Initialize  $a_1 = 0, a_2 = 0, sum = 0, col = 0$ .
2: for  $k = 1, \dots, l$  do
3:   if (multiple input) then
4:     Orthogonalize the columns in  $\tilde{B}(s_k)$  using the modified Gram–Schmidt process:  $[v_1, v_2, \dots, v_{m_k}] = \text{orth}\{\tilde{B}(s_k)\}$ ,
5:      $sum = m_k$ . ( $m_k$  is the number of remaining columns after deflation.)
6:   else
7:     Compute the first column in  $V$ :  $v_1 = \tilde{B}(s_k) / \|\tilde{B}(s_k)\|_2$ ,
8:      $sum = 1$ .
9:   end if
10:  Orthogonalize the columns in  $\tilde{A}(s_k)\tilde{B}(s_k), \dots, \tilde{A}(s_k)^{q_k-1}\tilde{B}(s_k)$  iteratively as follows:
11:  for  $i = 1, 2, \dots, q_k - 1$  do
12:     $a_2 = sum$ .
13:    if  $a_1 = a_2$  then
14:      break; go to Step 2
15:    else
16:      for  $j = a_1 + 1, \dots, a_2$  do
17:         $w = \tilde{A}(s_k)v_j, col = sum + 1$ .
18:        for  $d = 1, 2, \dots, col - 1$  do
19:           $h = v_d^T w, w = w - hv_d$ .
20:        end for
21:        if  $\|w\|_2 > \varepsilon$  ( $\varepsilon > 0$  is a small value indicating deflation, e. g.,  $\varepsilon = 10^{-7}$ ) then
22:           $v_{col} = \frac{w}{\|w\|_2}, sum = col$ .
23:        end if
24:      end for ( $j$ )
25:    end if
26:     $a_1 = a_2$ .
27:  end for ( $i$ )
28:   $V_k = [v_1, \dots, v_{sum}]$ ,
29: end for ( $k$ )
30: Orthogonalize the columns in  $[V_1, \dots, V_l]$  by the modified Gram–Schmidt process to obtain  $V$ , i. e.  $V := \text{orth}\{V_1, \dots, V_l\}$ .

```

$r_V < r_W$, then add $r_W - r_V$ random orthogonal columns to V , and vice versa. This way, the moment-matching property of the ROM remains unchanged due to the definitions of V, W in Theorem 3.3.

Algorithm 3.2: Compute V in (3.12) and W in (3.13) for a non-parametric SISO system (3.1).

Input: System matrices E, A, B, C , expansion points s_1, \dots, s_l .

Output: Projection matrices V, W .

```

1: Initialize  $col = 0, col_v = 0, col_w = 0$ .
2: for  $k = 1, \dots, l$  do
3:   Compute the first column of  $V_k$ :  $v_1 = \tilde{B}(s_k) / \|\tilde{B}(s_k)\|_2$ .
4:   Compute the first column of  $W_k$ :  $w_1 = \tilde{C}(s_k) / \|\tilde{C}(s_k)\|_2$ .
5:    $col = col + 1$ .
6:   Orthogonalize the vectors  $\tilde{A}(s_k)\tilde{B}(s_k), \dots, \tilde{A}(s_k)^{q_k-1}\tilde{B}(s_k)$  against  $v_1$  iteratively, orthogonalize the vectors  $\tilde{A}_c(s_k)\tilde{C}(s_k), \dots, \tilde{A}_c(s_k)^{q_k-1}\tilde{C}(s_k)$  against  $w_1$  iteratively, as follows:
7:   for  $i = 1, 2, \dots, q_k - 1$  do
8:      $v = \tilde{A}(s_k)v_i$ ,
9:      $w = \tilde{A}_c(s_k)w_i$ 
10:    for  $j = 1, 2, \dots, col$  do
11:       $h = v_j^T v$ ,  $v = v - hv_j$ .
12:       $h = w_j^T w$ ,  $w = w - hw_j$ .
13:    end for
14:     $col = col + 1$ .
15:    if  $\|v\|_2 > \varepsilon$  ( $\varepsilon > 0$  is a small value indicating deflation, e. g.,  $\varepsilon = 10^{-7}$ ) then
16:       $v_{col} = \frac{v}{\|v\|_2}$ ,
17:    else
18:       $col_v = col - 1$ , stop updating  $V_k$ .
19:    end if
20:    if  $\|w\|_2 > \varepsilon$  then
21:       $w_{col} = \frac{w}{\|w\|_2}$ ,
22:    else
23:       $col_w = col - 1$ , stop updating  $W_k$ .
24:    end if
25:  end for
26:   $V_k = [v_1, \dots, v_{col_v}]$ ,  $W_k = [w_1, \dots, w_{col_w}]$ .
27: end for
28: Orthogonalize the columns in  $[V_1, \dots, V_l]$  by the modified Gram–Schmidt process to obtain  $V$ , i. e.  $V := \text{orth}\{V_1, \dots, V_l\}$ .
29: Orthogonalize the columns in  $[W_1, \dots, W_l]$  by the modified Gram–Schmidt process to obtain  $W$ , i. e.  $W := \text{orth}\{W_1, \dots, W_l\}$ .

```

The issue is then how to (adaptively) choose the multiple expansion points. Many adaptive techniques have been proposed during the last years [8, 14, 25, 40, 39, 38, 30, 28, 56], where some are more or less heuristic [8, 14, 25, 40]. Based on system theory, an

Algorithm 3.3: Modified Gram–Schmidt process.

Input: A group of nonzero vectors a_1, \dots, a_l , a deflation tolerance $\varepsilon > 0$.

Output: A group of orthogonalized vectors $v_1, \dots, v_{\tilde{l}}$, $\tilde{l} \leq l$.

```

1:  $v_1 = a_1 / \|a_1\|_2$ ,  $\tilde{l} = 1$ .
2: for  $k = 2, \dots, l$  do
3:   for  $i = 1, \dots, \tilde{l}$  do
4:      $h = v_i^T a_k$ ,
5:      $a_k = a_k - hv_i$ .
6:   end for
7:   if  $\|a_k\|_2 > \varepsilon$  then
8:      $\tilde{l} = \tilde{l} + 1$ ,
9:      $v_{\tilde{l}} = \frac{a_k}{\|a_k\|_2}$ ,
10:  end if
11: end for

```

error bound is derived in [56], but it faces high computational complexity. The residual of the state vector is simply used in [39] as the error estimator of the ROM. In the next subsection, we introduce several typical techniques of adaptivity [38, 25, 30, 28].

3.2.4 Selection of expansion points

3.2.4.1 \mathcal{H}_2 -optimal iterative rational Krylov algorithm

The iterative rational Krylov algorithm (IRKA) is proposed in [38]. Given a group of initial expansion points, IRKA adaptively updates the expansion points, and upon convergence, IRKA produces a ROM satisfying \mathcal{H}_2 -optimal necessary conditions. (See Equation (5) in Chapter 1 of this volume for the definition of the \mathcal{H}_2 -norm.) The expansion points are selected as the mirror images of the poles of the updated ROM at each iteration. The algorithm is presented as Algorithm 3.4.

Moment-matching property

For single-input single-output (SISO) systems, IRKA leads to the following interpolation property upon convergence:

$$\begin{aligned} \hat{H}(-\hat{\lambda}_i) &= H(-\hat{\lambda}_i), \\ \frac{\partial \hat{H}(-\hat{\lambda}_i)}{\partial s} &= \frac{\partial H(-\hat{\lambda}_i)}{\partial s}. \end{aligned} \tag{3.14}$$

Here $\hat{\lambda}_i$, $i = 1, \dots, r$, are the eigenvalues of the ROM defined in Step 3(b) of Algorithm 3.4. They are also the poles of the reduced transfer function $\hat{H}(s)$.

Algorithm 3.4: Iterative Rational Krylov Algorithm (IRKA).

- 1: Make an initial selection of the expansion points closed under conjugation, i. e. $s_1, \dots, s_i, \bar{s}_i, \dots, s_r$, if s_i is a complex variable. Fix a tolerance ϵ for the accuracy of the ROM. Choose initial directions $\tilde{b}_1, \dots, \tilde{b}_r, \tilde{c}_1, \dots, \tilde{c}_r$.
- 2: Choose V_r, W_r so that

$$\begin{aligned} \text{range}(V_r) &= \text{span}\{\tilde{B}(s_1)\tilde{b}_1, \dots, \tilde{B}(s_i)\tilde{b}_i, \tilde{B}(\bar{s}_i)\tilde{b}_{i+1}, \dots, \tilde{B}(s_r)\tilde{b}_r\}, \\ \text{range}(W_r) &= \text{span}\{\tilde{C}(s_1)\tilde{c}_1, \dots, \tilde{C}(s_i)\tilde{c}_i, \tilde{C}(\bar{s}_i)\tilde{c}_{i+1}, \dots, \tilde{C}(s_r)\tilde{c}_r\}, \end{aligned}$$

and $W_r = W_r(V_r^T W_r)^{-1}$. Here $\tilde{B}(s_i) = (s_i E - A)^{-1} B$, $\tilde{C}(s_i) = (s_i E - A)^{-T} C^T$, $i = 1, \dots, r$.

- 3: While $(\max_{j=1, \dots, r} \{\frac{s_j - s_j^{\text{old}}}{s_j}\} > \epsilon)$
 - (a) $\hat{E} = W_r^T E V_r$, $\hat{A} = W_r^T A V_r$, $\hat{B} = W_r^T B$, $\hat{C} = C V_r$.
 - (b) Compute eigenvalues, -vectors of $\lambda E - A$ so that $(\lambda_i \hat{E} - \hat{A})y_i = \lambda_i y_i$, $i = 1, \dots, r$.
 - (c) Assign $s_i \leftarrow -\lambda_i$ for $i = 1, \dots, r$, $Y = (y_1, \dots, y_r)$.
 - (d) $\tilde{B} = \hat{B}^T Y^{-T}$, $\tilde{C} = \hat{C} Y$, $(\tilde{b}_1, \dots, \tilde{b}_r) \leftarrow \tilde{B}$, $(\tilde{c}_1, \dots, \tilde{c}_r) \leftarrow \tilde{C}$.
 - (e) Update V_r and W_r :

$$\text{range}(V_r) = \text{span}\{\tilde{B}(s_1)\tilde{b}_1, \dots, \tilde{B}(s_r)\tilde{b}_r\},$$

$$\text{range}(W_r) = \text{span}\{\tilde{C}(s_1)\tilde{c}_1, \dots, \tilde{C}(s_r)\tilde{c}_r\},$$

and $W_r = W_r(V_r^T W_r)^{-1}$.

- 4: $\hat{E} = W_r^T E V_r$, $\hat{A} = W_r^T A V_r$, $\hat{B} = W_r^T B$, $\hat{C} = C V_r$.

It is easy to see that the images of the poles of the ROM are selected as the expansion points, and are updated every time the ROM is updated. In IRKA, \bar{s}_i is the conjugate of s_i . From the definition of the moments of the transfer function, we know that the first-order derivative of the transfer function at s_i is the first-order moment $m_1(s_i)$. The value of the transfer function at s_i is the zeroth-order moment $m_0(s_i)$. Therefore, IRKA generates ROMs matching the first two moments of the transfer function at each expansion point s_i , $i = 1, \dots, r$.

Optimality property [38]

The ROM computed by IRKA satisfies the following necessary conditions of optimality.

Theorem 3.4. *Let $H(s)$ be the transfer function of a stable SISO system, and \hat{H} be a local minimizer of dimension r for the optimal \mathcal{H}_2 -model reduction problem*

$$\|H - \hat{H}\|_{\mathcal{H}_2} = \min_{\substack{\dim(\tilde{H})=r, \\ \tilde{H}:\text{stable}}} \|H - \tilde{H}\|_{\mathcal{H}_2},$$

and suppose that $\hat{H}(s)$ has simple poles at $\hat{\lambda}_i, i = 1, \dots, r$, then $\hat{H}(s)$ interpolates $H(s)$ and its first derivative at $\hat{\lambda}_i, i = 1, \dots, r$:

$$\hat{H}(-\hat{\lambda}_i) = H(-\hat{\lambda}_i), \quad \frac{\partial \hat{H}(-\hat{\lambda}_i)}{\partial s} = \frac{\partial H(-\hat{\lambda}_i)}{\partial s}.$$

Comparing Theorem 3.4 with the moment-matching property in (3.14), we see that IRKA constructs a ROM that satisfies the necessary condition of the local optimal property in Theorem 3.4.

3.2.4.2 A heuristic technique

In [25], an adaptive scheme for both choosing expansion points and deciding the number of moments is delineated. Generally speaking, the expansion points are chosen based on a binary principle. The number of moments matched at each expansion point is determined by a tested point which is known to cause the largest error in the interval of each pair of neighboring expansion points. Using this technique, the projection matrix V in (3.12) is adaptively computed, and the ROM is obtained by Galerkin projection using $W = V$. The only inputs of the algorithm are an acceptable dimension of the ROM, say r_{\max} , as well as the acceptable accuracy of the ROM, tol . r_{\max} will be adjusted to a proper number during the adaptive scheme if it was selected too small. The details of the algorithm can be found in [25]. From the numerical examples in [25], the method shows its success in automatically obtaining ROMs for several circuit examples. It is nevertheless clarified in [25] that the proposed method has difficulty in dealing with multi-input and multi-output (MIMO) models with many resonances in the output responses. The proposed method may obtain good results for a single-input and single-output (SISO) system with many resonances in the output, but it will fail when the system is MIMO and possesses multiple resonances in all the output responses of all the I-O ports. For such systems, an efficient error estimation may help to construct more robust and reliable ROMs. In the next subsection, we introduce a greedy-type algorithm which adaptively selects the expansion points using a recently developed *a posteriori* error bound.

3.2.4.3 Scheme based on a posteriori error estimation

In [28], an *a posteriori* error bound $\Delta(\tilde{\mu})$ for the transfer function of the ROM is proposed. This will be discussed in Section 3.3.4, where the error bound is defined for linear parametric systems, and can straightforwardly treat linear non-parametric systems as a special case. For linear non-parametric systems, the error bound $\Delta(\tilde{\mu})$ actually depends only on s , i. e. $\tilde{\mu} = s$. $\Delta(s)$ can be computed following (3.25) and (3.26), except that $\tilde{\mu}$ is replaced by s .

Similar to the reduced basis method (see Chapter 4 of Volume 2), the next expansion point s_i is iteratively selected as the point at which the error bound is maximized. Using the error bound, the ROM can be automatically generated by Algorithm 3.5. The projection matrices W and V for constructing the ROM are extended iteratively by the matrices $W_{\hat{s}}$ and $V_{\hat{s}}$ generated at the selected expansion point \hat{s} , until the error bound is below the error tolerance ϵ_{tol} . The so-called training set Ξ_{train} is a set of samples of s , which is given by the user, and which should cover the interesting range of the frequency axis. The expansion points are selected from Ξ_{train} . The matrices W^{du} and V^{du} are used to compute the error bound; see (3.25).

Algorithm 3.5: Automatic generation of a reduced model by adaptively selecting expansion points \hat{s} for non-parametrized LTI systems.

Input: System matrices $E, A, B, C, \epsilon_{\text{tol}} > 0, \Xi_{\text{train}}$: a large set of samples of s , taken over the interesting range of the frequency.

Output: The projection matrices W, V .

- 1: $W = [], V = [],$ set $\epsilon = \epsilon_{\text{tol}} + 1$.
 - 2: Initial expansion point: $\hat{s} \in \Xi_{\text{train}}$.
 - 3: **while** $\epsilon > \epsilon_{\text{tol}}$ **do**
 - 4: $\text{range}(V_{\hat{s}}) = \text{span}\{\tilde{B}(\hat{s}), \tilde{A}(\hat{s})\tilde{B}(\hat{s}), \dots, \tilde{A}^{q-1}(\hat{s})\tilde{B}(\hat{s})\},$
 - 5: $\text{range}(W_{\hat{s}}) = \text{span}\{\tilde{C}(\hat{s}), \tilde{A}_c(\hat{s})\tilde{C}(\hat{s}), \dots, \tilde{A}_c^{q-1}(\hat{s})\tilde{C}(\hat{s})\}.$
 - 6: $V = \text{orth}\{V, V_{\hat{s}}\}, W^{du} = V.$
 - 7: $W = \text{orth}\{W, W_{\hat{s}}\}, V^{du} = W.$
 - 8: $\hat{s} = \arg \max_{s \in \Xi_{\text{train}}} \Delta(s).$
 - 9: $\epsilon = \Delta(\hat{s}).$
 - 10: **end while**
-

Either $V_{\hat{s}}$ in Step 6 or $W_{\hat{s}}$ in Step 7 in Algorithm 3.5 can be computed by Step 1, Steps 3–29 plus Step 31 in Algorithm 3.1. Step 6 or Step 7 of Algorithm 3.5 implements the modified Gram–Schmidt process, Algorithm 3.3.

3.2.4.4 Complex expansion points

Note that the projection matrices V, W computed by the moment-matching method, as well as the multi-moment-matching method in the next section, could be complex, if the expansion point for the variable s is taken as a complex number. The ROM then has complex system matrices, even if the original system matrices are real.

In order to obtain real reduced system matrices, each complex matrix should be separated into its real part and its imaginary part, which should then be combined to obtain a real projection matrix for MOR, i. e. we need to do the following extra

step:

$$\begin{aligned} V &\leftarrow \text{orth}\{\text{Re}(V), \text{Im}(V)\}. \\ W &\leftarrow \text{orth}\{\text{Re}(W), \text{Im}(W)\}. \end{aligned}$$

Here and below, $\text{Re}(\cdot)$ and $\text{Im}(\cdot)$ is the real and imaginary part of a complex variable, respectively. Since, in \mathbb{C}^n ,

$$\text{range}\{V\} = \text{colspan}\{\text{Re}(V), \text{Im}(V)\}, \quad \text{range}\{W\} = \text{colspan}\{\text{Re}(W), \text{Im}(W)\},$$

over \mathbb{C} , the moment-matching property in Theorem 3.3 remains unchanged.

The algorithm IRKA in Section 3.2.4.1 can also introduce complex interpolation points s_i and \bar{s}_i , where \bar{s}_i is the conjugate of s_i . If s_i is complex, then in Step 2 of IRKA, $(s_i E - A)B\tilde{B}_i$ and $(\bar{s}_i E - A)B\tilde{B}_{i+1}$ are both complex vectors, which may produce complex matrices V_r, W_r . It is nevertheless not difficult to verify that the conjugate of $(s_i E - A)B\tilde{B}_i$ is $(\bar{s}_i E - A)B\tilde{B}_{i+1}$, so that they have the same real and imaginary (up to the sign) parts. Therefore, in Step 2, we can replace $(s_i E - A)B\tilde{B}_i$ and $(\bar{s}_i E - A)B\tilde{B}_{i+1}$ with $\text{Re}[(s_i E - A)B\tilde{B}_i]$ and $\text{Im}[(s_i E - A)B\tilde{B}_i]$ for any complex s_i , without changing the subspace.

3.3 Multi-moment-matching for linear parametric systems

Some parametric model order reduction (PMOR) methods are basically extensions of MOR methods for non-parametric systems. PMOR methods can be used to compute the ROM of the parametric system in (3.1), where the vector of parameters μ should be symbolically preserved in the ROM as follows:

$$\begin{aligned} \hat{E}(\mu) \frac{d\mathbf{z}(t, \mu)}{dt} &= \hat{A}(\mu)\mathbf{z}(t, \mu) + \hat{B}(\mu)\mathbf{u}(t), \\ \hat{\mathbf{y}}(t, \mu) &= \hat{C}(\mu)\mathbf{z}. \end{aligned}$$

Here $\hat{E}(\mu) = W^T E(\mu) V$, $\hat{A}(\mu) = W^T A(\mu) V$, $\hat{B}(\mu) = W^T B(\mu)$ and $\hat{C}(\mu) = C(\mu) V$. A survey of PMOR methods can be found in [13].

3.3.1 A robust algorithm

Multi-moment matching PMOR methods are reported in [17, 24], which are generalizations of the moment-matching method [35]. In this section, the robust PMOR algorithm proposed in [24] is reviewed. For ease of notation, we call this method PMOR-MM. Both methods in [17, 24] are based on Galerkin projection, i. e. $W = V$. Note that the method

in [24] is already extended to Petrov–Galerkin in [2]. For clarity and simplicity, we use Galerkin projection to explain the idea and the algorithm. Assume that $E(\mu)$, $A(\mu)$ are either in the affine form defined as

$$\begin{aligned} E(\mu) &= E_0 + E_1\mu_1 + \cdots + E_m\mu_m, \\ A(\mu) &= A_0 + A_1\mu_1 + \cdots + A_m\mu_m, \end{aligned}$$

or can be approximated in the affine form above.

To compute the matrix V , a series expansion of the state \mathbf{x} in the frequency domain is used. After Laplace transform (with zero initial condition), the original parametric system in (3.1) can be written as

$$\begin{aligned} G(\tilde{\mu})X(\tilde{\mu}) &= B(\tilde{\mu})U(\tilde{\mu}), \\ Y(\tilde{\mu}) &= C(\tilde{\mu})X(\tilde{\mu}), \end{aligned} \tag{3.15}$$

where the entries in $\tilde{\mu} = (\tilde{\mu}_1, \dots, \tilde{\mu}_p)$ are sufficiently smooth functions of the parameters μ_1, \dots, μ_m and the Laplace variable s . $U(\tilde{\mu})$ is the Laplace transform of $u(t)$. Due to the affine form of $E(\mu)$ and $A(\mu)$, $G(\tilde{\mu})$ can also be written in affine form as

$$G(\tilde{\mu}) = G_0 + G_1\tilde{\mu}_1 + \cdots + G_p\tilde{\mu}_p.$$

As a result, the state vector in the frequency domain can be written as

$$\begin{aligned} X(\tilde{\mu}) &= [G(\tilde{\mu})]^{-1}B(\tilde{\mu})U(\tilde{\mu}) \\ &= [G_0 + G_1\tilde{\mu}_1 + \cdots + G_p\tilde{\mu}_p]^{-1}B(\tilde{\mu})U(\tilde{\mu}). \end{aligned} \tag{3.16}$$

Given an expansion point $\tilde{\mu}^0 = [\tilde{\mu}_1^0, \dots, \tilde{\mu}_p^0]$, $X(\tilde{\mu})$ in (3.16) can be expanded as

$$\begin{aligned} X(\tilde{\mu}) &= [I - (\sigma_1 M_1 + \cdots + \sigma_p M_p)]^{-1} \tilde{B}_M U(\tilde{\mu}) \\ &= \sum_{i=0}^{\infty} (\sigma_1 M_1 + \cdots + \sigma_p M_p)^i \tilde{B}_M U(\tilde{\mu}), \end{aligned} \tag{3.17}$$

where $\tilde{B}_M = [G(\tilde{\mu}^0)]^{-1}B(\tilde{\mu})$, $M_i = -[G(\tilde{\mu}^0)]^{-1}G_i$, $i = 1, 2, \dots, p$, and $\sigma_i = \tilde{\mu}_i - \tilde{\mu}_i^0$, $i = 1, 2, \dots, p$. We call the coefficients in the above series expansion the *moment vectors (matrices)* of the parametrized system. The corresponding multi-moments of the transfer function are those moment vectors multiplied by C from the left.

To obtain the projection matrix V , instead of directly computing the moment vectors [17], a numerically robust method is proposed in [29], and a detailed algorithm is presented in [24]. The method combines the recursions in (3.18), with a repeated modified Gram–Schmidt process so that the moment vectors are computed implicitly. We

have

$$\begin{aligned}
R_0 &= B_M, \\
R_1 &= [M_1 R_0, \dots, M_p R_0], \\
R_2 &= [M_1 R_1, \dots, M_p R_1], \\
&\vdots \\
R_q &= [M_1 R_{q-1}, \dots, M_p R_{q-1}], \\
&\vdots
\end{aligned} \tag{3.18}$$

Here, $B_M = \tilde{B}_M$, if $B(\tilde{\mu})$ does not depend on μ , i.e. $B(\tilde{\mu}) = B$. Otherwise, $B_M = [\tilde{B}_{M_1}, \dots, \tilde{B}_{M_p}]$, $\tilde{B}_{M_i} = [G(\tilde{\mu}^0)]^{-1} B_i$, $i = 1, \dots, p$, if $B(\tilde{\mu})$ can be approximated in affine form, e. g., $B(\tilde{\mu}) \approx B_1 \tilde{\mu}_1 + \dots + B_p \tilde{\mu}_p$.

Then $V := V_{\tilde{\mu}^0}$ is computed as

$$\text{range}(V_{\tilde{\mu}^0}) = \text{span}\{R_0, R_1, \dots, R_q\}_{\tilde{\mu}^0}, \tag{3.19}$$

with the sub-index denoting the dependance on the expansion point $\tilde{\mu}^0$. It is proved in [24] that the leading multi-moments of the original system match those of the ROM. The accuracy of the ROM can be improved by increasing the number of terms in (3.19), whereby more multi-moments can be matched. To be self-contained, we present the method in Algorithm 3.6.

It is noticed that the dimensions of R_j , $j = 1, \dots, q$, increase exponentially. If the number p of the parameters is larger than 2, it is advantageous to use multiple point expansion, such that only the low order moment matrices, e. g., R_j , $j \leq 1$, have to be computed for each expansion point. As a result, the order of the ROM can be kept small. Given a group of expansion points $\tilde{\mu}^i$, $i = 0, \dots, \ell$, a matrix $V_{\tilde{\mu}^i}$ can be computed from (3.19) for each $\tilde{\mu}^i$ as

$$\text{range}(V_{\tilde{\mu}^i}) = \text{span}\{R_0, R_1, \dots, R_q\}_{\tilde{\mu}^i}, \tag{3.20}$$

where R_j , $j = 1, \dots, q$, are defined as in (3.18), with $R_0 = B_M$, $B_M = [G(\tilde{\mu}^i)]^{-1} B$, or $B_M = [\tilde{B}_{M_1}, \dots, \tilde{B}_{M_p}]$, $\tilde{B}_{M_j} = [G(\tilde{\mu}^i)]^{-1} B_j$, $M_j = -[G(\tilde{\mu}^i)]^{-1} G_j$, $j = 1, 2, \dots, p$. The final projection matrix V is a combination (orthogonalization) of all the matrices $V_{\tilde{\mu}^i}$,

$$V = \text{orth}\{V_{\tilde{\mu}^0}, \dots, V_{\tilde{\mu}^\ell}\}. \tag{3.21}$$

The multi-moment-matching PMOR method is very efficient for linear parametric systems, especially for systems with affine matrices $E(\mu)$, $A(\mu)$ [22]. The method also performs well if the matrices are not affine, but it can be well approximated in affine form [13, 30].

Algorithm 3.6: Compute $V = [v_1, v_2, \dots, v_q]$ for a parametric system (3.1), where $B(\mu)$ is generally considered as a matrix.

Input: Expansion point $\tilde{\mu}^0$, moment vectors in R_1, R_2, \dots, R_q .

Output: Projection matrix V .

- 1: Initialize $a_1 = 0, a_2 = 0, \text{sum} = 0$.
 - 2: **if** (multiple input) **then**
 - 3: Orthogonalize the columns in R_0 using the modified Gram–Schmidt process: $[v_1, v_2, \dots, v_{q_1}] = \text{orth}\{R_0\}$,
 - 4: $\text{sum} = q_1$. (q_1 is the number of remaining columns after orthogonalization.)
 - 5: **else**
 - 6: Compute the first column in V : $v_1 = R_0 / \|R_0\|_2, \text{sum} = 1$.
 - 7: **end if**
 - 8: Orthogonalize the columns in R_1, R_2, \dots, R_q iteratively as follows:
 - 9: **for** $i = 1, 2, \dots, q$ **do**
 - 10: $a_2 = \text{sum}$;
 - 11: **for** $d = 1, 2, \dots, p$ **do**
 - 12: **if** $a_1 = a_2$ **then**
 - 13: stop
 - 14: **else**
 - 15: **for** $j = a_1 + 1, \dots, a_2$ **do**
 - 16: $w = \tilde{G}^{-1}(\tilde{\mu}^0)G_d v_j, \text{col} = \text{sum} + 1$.
 - 17: **for** $k = 1, 2, \dots, \text{col} - 1$ **do**
 - 18: $h = v_k^T w, w = w - h v_k$.
 - 19: **end for**
 - 20: **if** $\|w\|_2 > \varepsilon$ (a small value indicating deflation, e. g., $\varepsilon = 10^{-7}$) **then**
 - 21: $v_{\text{col}} = \frac{w}{\|w\|_2}, \text{sum} = \text{col}$.
 - 22: **end if**
 - 23: **end for** (j)
 - 24: **end if**
 - 25: **end for** (d)
 - 26: $a_1 = a_2$;
 - 27: **end for** (i)
 - 28: Orthogonalize the columns in V by the modified Gram–Schmidt process.
-

3.3.2 Applicability to steady systems

The above PMOR method computes ROMs of the dynamical systems in (3.1). It is easy to see that the method can be naturally applied to steady systems:

$$\begin{aligned} (E_0 + E_1\mu_1 + \dots + E_m\mu_m)\mathbf{x}(\mu) &= B(\mu)u(\mu), \\ \mathbf{y}(\mu) &= C(\mu)\mathbf{x}(\mu). \end{aligned} \tag{3.22}$$

Comparing (3.22) with the Laplace transformed system (3.15), we see that they have an identical form. Consequently, the series expansion of \mathbf{x} in (3.22) can be obtained similarly to (3.17). The corresponding moment vectors can also be defined according to (3.18). Algorithm 3.6 can then be used to compute a projection matrix V . Then the ROM of (3.22) is constructed by a Galerkin projection,

$$\begin{aligned} (V^T E_0 V + V^T E_1 V \mu_1 + \cdots + V^T E_m V \mu_q) \mathbf{x}(\mu) &= V^T B(\mu) u(\mu), \\ \hat{\mathbf{y}}(\mu) &= C(\mu) V \mathbf{x}(\mu). \end{aligned}$$

3.3.3 Structure-preserving (P)MOR for second-order systems

For the second-order systems

$$\begin{aligned} M(\mu) \frac{d^2 \mathbf{x}(t, \mu)}{dt^2} + D(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} + K(\mu) \mathbf{x}(t, \mu) &= B(\mu) u(t), \\ \mathbf{y}(t, \mu) &= C(\mu) \mathbf{x}(t, \mu), \end{aligned} \quad (3.23)$$

often arising from, e. g., mechanical engineering, it is desired that the ROM preserves the second-order structure, i. e.,

$$\begin{aligned} W^T M(\mu) V \frac{d^2 \mathbf{z}(t, \mu)}{dt^2} + W^T D(\mu) V \frac{d\mathbf{z}(t, \mu)}{dt} + W^T K(\mu) V \mathbf{z}(t, \mu) &= W^T B(\mu) u(t), \\ \mathbf{y}(t, \mu) &= C(\mu) V \mathbf{z}(t, \mu). \end{aligned} \quad (3.24)$$

Note that PMOR-MM computes the projection matrix using the series expansion of the state vector \mathbf{x} in the frequency domain. After a Laplace transformation (assume $\mathbf{x}(t = 0, \mu) = \mathbf{0}$), the first equation in (3.23) becomes

$$\begin{aligned} s^2 M(\mu) X(s, \mu) + s D(\mu) X(s, \mu) + K(\mu) X(s, \mu) &= B(\mu) U(s), \\ Y(s, \mu) &= C(\mu) X(s, \mu). \end{aligned}$$

Thus

$$[s^2 M(\mu) + s D(\mu) + K(\mu)] X(s, \mu) = B(\mu) U(s),$$

where $U(s)$ is the Laplace transform of the input signal $u(t)$. Defining $G(\tilde{\mu}) := s^2 M(\mu) + s D(\mu) + K(\mu)$, $\tilde{\mu} = (\tilde{\mu}_1, \tilde{\mu}_2, \tilde{\mu}_3) := (s^2, s, \mu)$, a projection matrix V can be computed following (3.15)–(3.21) in Section 3.3.1. Applying a Petrov-Galerkin projection to the second-order system, we can obtain a second-order ROM as in (3.24). Note that with a Galerkin projection, also the symmetry and definiteness properties of the coefficient matrices can be preserved in the ROM.

3.3.4 Selecting expansion points based on a posteriori error estimation

Note that the projection matrix V in (3.21) depends on the multiple expansion points $\tilde{\mu}^i, i = 1, \dots, \ell$.

In this section, we introduce an *a posteriori* error bound proposed in [28], which is an error bound for the transfer function $\hat{H}(\tilde{\mu})$ of the ROM. Given the error bound $\Delta(\tilde{\mu})$ for the ROM, the expansion points $\tilde{\mu}^i$ can be adaptively selected, and the projection matrix V can be automatically computed as shown in Algorithm 3.7.

Algorithm 3.7: Adaptively selecting expansion points $\tilde{\mu}^i$, and automatically computing V .

Input: ε_{tol} , set $\varepsilon = \varepsilon_{\text{tol}} + 1$, Ξ_{train} : a set of samples of $\tilde{\mu}$ covering the interesting domain.

Output: V .

- 1: $V = []$, $V^{du} = []$.
 - 2: Choose an initial expansion point: $\tilde{\mu}^0 \in \Xi$, $i = 0$.
 - 3: **while** $\varepsilon > \varepsilon_{\text{tol}}$ **do**
 - 4: $\text{range}(V_{\tilde{\mu}^i}) = \text{span}\{R_0, R_1, \dots, R_q\}_{\tilde{\mu}^i}$. (By applying Algorithm 3.6 at expansion point $\tilde{\mu}^i$.)
 - 5: $\text{range}(V_{\tilde{\mu}^i}^{du}) = \text{span}\{R_0^{du}, R_1^{du}, \dots, R_q^{du}\}_{\tilde{\mu}^i}$. (By applying Algorithm 3.6 at expansion point $\tilde{\mu}^i$, and replacing R_0, R_1, \dots, R_q with $R_0^{du}, R_1^{du}, \dots, R_q^{du}$ in (3.27).)
 - 6: $V = \text{orth}\{V, V_{\tilde{\mu}^i}\}$, $W = V$.
 - 7: $V^{du} = \text{orth}\{V^{du}, V_{\tilde{\mu}^i}^{du}\}$, $W^{du} = V^{du}$.
 - 8: $i = i + 1$.
 - 9: $\tilde{\mu}^i = \arg \max_{\tilde{\mu} \in \Xi_{\text{train}}} \Delta(\tilde{\mu})$.
 - 10: $\varepsilon = \Delta(\tilde{\mu}^i)$.
 - 11: **end while**.
-

For a MIMO system, the error bound $\Delta(\tilde{\mu})$ is defined as

$$\Delta(\tilde{\mu}) = \max_{ij} \Delta_{ij}(\tilde{\mu}).$$

Here $\Delta_{ij}(\tilde{\mu})$ is the error bound for the (i, j) th entry of the transfer function (it is a matrix for MIMO systems) of the ROM, i. e.,

$$|H_{ij}(\tilde{\mu}) - \hat{H}_{ij}(\tilde{\mu})| \leq \Delta_{ij}(\tilde{\mu}).$$

For a SISO system, there is no need to take the maximum. $\Delta_{ij}(\tilde{\mu})$ can be computed as

$$\Delta_{ij}(\tilde{\mu}) = \frac{\|\mathbf{r}_i^{du}(\tilde{\mu})\|_2 \|\mathbf{r}_j^{pr}(\tilde{\mu})\|_2}{\beta(\tilde{\mu})} + |(\hat{\mathbf{x}}^{du})^* \mathbf{r}_j^{pr}(\tilde{\mu})|. \quad (3.25)$$

Here and below, $(\cdot)^*$ is the conjugate transpose of a vector or a matrix. Let \mathbf{c}_i be the i th row of $C(\tilde{\mu})$ and \mathbf{b}_j be the j th column of $B(\tilde{\mu})$ in (3.15),

$$\begin{aligned} \mathbf{r}_j^{pr}(\tilde{\mu}) &= \mathbf{b}_j - G(\tilde{\mu})\hat{\mathbf{x}}_j^{pr}, \\ \hat{\mathbf{x}}_j^{pr} &= V[W^T G(\tilde{\mu})V]^{-1}W^T \mathbf{b}_j, \end{aligned} \quad (3.26)$$

$$\begin{aligned}\mathbf{r}_i^{du}(\tilde{\mu}) &= -\mathbf{c}_i^T - G^*(\tilde{\mu})\hat{\mathbf{x}}_i^{du}, \\ \hat{\mathbf{x}}_i^{du} &= -V^{du}[(W^{du})^T G^*(\tilde{\mu})V^{du}]^{-1}(W^{du})^T \mathbf{c}_i^T,\end{aligned}$$

where $\hat{\mathbf{x}}_j^{pr}$ is the approximate solution to the primal system

$$G(\tilde{\mu})\mathbf{x}_j^{pr} = \mathbf{b}_j,$$

and can be computed from the ROM of the primal system obtained with the projection matrices W, V . $\hat{\mathbf{x}}_i^{du}$ is the approximate solution to the dual system

$$G^*(\tilde{\mu})\mathbf{x}_i^{du} = -\mathbf{c}_i^T,$$

and can be computed from the ROM of the dual system obtained with the projection matrices W^{du}, V^{du} . The variable $\beta(\tilde{\mu})$ is the smallest singular value of the matrix $G(\tilde{\mu})$. The matrix V^{du} can be computed, for example, using (3.20) and (3.21), by replacing R_0, \dots, R_{qr} with $R_0^{du}, R_1^{du}, \dots, R_{qr}^{du}$, where the matrices $G(\tilde{\mu}^i)$ in R_0, \dots, R_{qr} are substituted by $G^*(\tilde{\mu}^i)$, and B is replaced with $-C^T$. More specifically,

$$\begin{aligned}R_0^{du} &= C_M^{du}, \\ R_1^{du} &= [M_1^{du} R_0^{du}, \dots, M_p^{du} R_0^{du}], \\ R_2^{du} &= [M_1^{du} R_1^{du}, \dots, M_p^{du} R_1^{du}], \\ &\vdots \\ R_q^{du} &= [M_1^{du} R_{q-1}^{du}, \dots, M_p^{du} R_{q-1}^{du}], \\ &\vdots\end{aligned}\tag{3.27}$$

$R_0^{du} = C_M^{du} = -[G^*(\tilde{\mu}^i)]^{-1}C^T$, $M_j = [G^*(\tilde{\mu}^i)]^{-1}G_j^T$, $j = 1, 2, \dots, p$. We can take $W^{du} = V^{du}$. The derivation of $\Delta(\tilde{\mu})$ is detailed in [28].

It is worth pointing out that, although the error bound is dependent on the parameter $\tilde{\mu}$, many $\tilde{\mu}$ -independent terms constituting the error bound need to be pre-computed only once, and they are repeatedly used in Algorithm 3.7 for the many samples of $\tilde{\mu}$ in Ξ_{train} . For example, when computing $\hat{\mathbf{x}}_j^{pr}$ in \mathbf{r}_j^{pr} , $W^T G_k V$, $k = 1, \dots, m$, are μ -independent, and need to be computed only once, $W^T G(\tilde{\mu})V$ is then derived by assembling $W^T G_k V$ for any value of $\tilde{\mu}$.

Algorithm 3.7 is similar to Algorithm 3.5, except that the projection matrix V is constructed for system (3.1) in the parametric case using Algorithm 3.6. At the i th iteration step, the expansion point $\tilde{\mu}^i$ is selected as the one maximizing the error bound $\Delta(\tilde{\mu})$. The projection matrix V is enriched by the matrix $V_{\tilde{\mu}^i}$ corresponding to $\tilde{\mu}^i$. The matrix V^{du} aids in computing the error bound. The most costly part of the error bound is $\beta(\tilde{\mu})$, since we need to compute the smallest singular value of the matrix $G(\tilde{\mu})$ of full dimension. The smallest singular value of the projected matrix $W^T G(\tilde{\mu})V$ could be

heuristically used to approximate $\beta(\bar{\mu})$. In [44], a method of interpolation is proposed to compute an approximation of $\beta(\bar{\mu})$, which has been shown to be accurate and cheap.

At the end of this section, we mention a method from [9], which deals with linear parametric systems with time-varying parameters. There, it is shown that these parametric systems can be equivalently considered as bilinear systems. Then suitable MOR methods for bilinear systems can be applied. MOR for bilinear systems will be discussed in the next section, where MOR based on multi-moment matching or bilinear IRKA (BIRKA) [9, 10, 12] are introduced.

3.4 Moment-matching MOR for nonlinear systems

In this section we consider mildly nonlinear systems without parameter variations,

$$\begin{aligned} E \frac{d\mathbf{x}(t)}{dt} &= \mathbf{f}(\mathbf{x}(t)) + Bu(t), \\ \mathbf{y}(t) &= C\mathbf{x}(t), \end{aligned} \quad (3.28)$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $f(\cdot) \in \mathbb{R}^n$ is a nonlinear, vector-valued function depending on $\mathbf{x}(t)$. These nonlinear systems usually come from spatial discretizations of nonlinear PDEs. The ROM via Petrov–Galerkin projection is obtained as follows:

$$W^T E V \frac{d\mathbf{z}(t)}{dt} = W^T \mathbf{f}(V\mathbf{z}(t)) + W^T Bu(t),$$

where $W \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{n \times r}$ with $W^T V = I$.

In the literature, some MOR methods for nonlinear systems are based on moment-matching. The quadratic method [16] is the simplest one. The bilinearization method [5, 46] is more accurate than the quadratic method. Methods based on variational analysis [11, 27, 37, 47, 52], in general, yield smaller errors than the previous two. A method based on a piece-wise linear approximation of the nonlinear function $f(\cdot)$ [50] could be used when dealing with strong nonlinearities. These methods are extensions of the moment-matching methods for linear systems.

3.4.1 Quadratic method

We first analyze the quadratic MOR method proposed in [16]. This method approximates the nonlinear function $\mathbf{f}(\cdot)$ by its power series expansion at, e. g., $\mathbf{x}(0) = \mathbf{0}$, which can be rewritten into a Kronecker product formulation of $\mathbf{x}(t)$ [52],

$$\begin{aligned} \mathbf{f}(\mathbf{x}(t)) &= \mathbf{f}(\mathbf{0}) + A_1 \mathbf{x}(t) + A_2 (\mathbf{x}(t) \otimes \mathbf{x}(t)) \\ &\quad + A_3 (\mathbf{x}(t) \otimes \mathbf{x}(t) \otimes \mathbf{x}(t)) + \dots, \end{aligned} \quad (3.29)$$

where $A_1 \in \mathbb{R}^{n \times n}$ is the Jacobian of \mathbf{f} and, in general, $A_j \in \mathbb{R}^{n \times n^j}$ denotes a matrix whose entries correspond to the j th-order partial derivatives of f_i w. r. t. x_1, \dots, x_n , $1 \leq i \leq n$. Here f_i, x_k are the i th and k th entry of \mathbf{f} and $\mathbf{x}(t)$, respectively. A quadratic system can be obtained by a truncation of (3.29),

$$\begin{aligned} E \frac{d\mathbf{x}(t)}{dt} &= A_1 \mathbf{x}(t) + A_2(\mathbf{x}(t) \otimes \mathbf{x}(t)) + Bu(t) + \mathbf{f}(\mathbf{0}), \\ \mathbf{y}(t) &= C\mathbf{x}(t). \end{aligned} \quad (3.30)$$

If $\mathbf{f}(\mathbf{0}) = \mathbf{0}$, the projection matrix V is computed as an orthonormal basis of the Krylov subspace $K_q(A_1^{-1}, A_1^{-1}B)$ as follows:

$$\text{range}(V) = \text{span}\{A_1^{-1}B, A_1^{-2}B, \dots, A_1^{-q}B\}.$$

Note that V is constructed only by use of the linear part of the quadratic system. A ROM is derived as

$$\begin{aligned} V^T E V \frac{d\mathbf{z}(t)}{dt} &= V^T A_1 V \mathbf{z}(t) + V^T A_2 (V \mathbf{z}(t) \otimes V \mathbf{z}(t)) + V^T B u(t), \\ \mathbf{y}(t) &= C V \mathbf{z}(t). \end{aligned}$$

It can be seen that the idea of the quadratic method comes from the moment-matching method for linear systems. The projection matrix V is computed in the same way as in the previous moment-matching methods, but is applied to the quadratic system.

If $\mathbf{f}(\mathbf{0}) \neq \mathbf{0}$, then the system in (3.30) can be reformulated into

$$\begin{aligned} E \frac{d\mathbf{x}(t)}{dt} &= A_1 \mathbf{x}(t) + A_2(\mathbf{x}(t) \otimes \mathbf{x}(t)) + [B, \mathbf{f}(\mathbf{0})][u(t), 1]^T, \\ \mathbf{y}(t) &= C\mathbf{x}(t). \end{aligned}$$

The input matrix B in (3.30) is replaced by the matrix $[B, \mathbf{f}(\mathbf{0})]$, which means $\mathbf{f}(\mathbf{0})$ can always be treated as a part of the input matrix of the system, therefore, for simplicity, we assume below that $\mathbf{f}(\mathbf{0}) = \mathbf{0}$.

3.4.2 Bilinearization method

For a nonlinear system with $E = I$, and with single input, i. e. B is a vector \mathbf{b} , a bilinear system can be obtained by applying the Carleman linearization process to the nonlinear system (3.28) [52]. In [5, 46], the bilinear system is derived by approximating $\mathbf{f}(\mathbf{x}(t))$ with a degree-2 polynomial in the Carleman linearization process. More specifically, by use of the first three terms in (3.29), we obtain the following approximation of $\mathbf{f}(\mathbf{x}(t))$:

$$\mathbf{f}(\mathbf{x}(t)) \approx A_1 \mathbf{x}(t) + A_2(\mathbf{x}(t) \otimes \mathbf{x}(t)).$$

With the definitions

$$\begin{aligned} \mathbf{x}_\otimes &= \begin{pmatrix} \mathbf{x}(t) \\ \mathbf{x}(t) \otimes \mathbf{x}(t) \end{pmatrix}, \quad B_\otimes = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix}, \quad C_\otimes = (C, \mathbf{0}), \\ A_\otimes &= \begin{pmatrix} A_1 & A_2 \\ 0 & A_1 \otimes I + I \otimes A_1 \end{pmatrix}, \\ N_\otimes &= \begin{pmatrix} 0 & 0 \\ \mathbf{b} \otimes I + I \otimes \mathbf{b} & 0 \end{pmatrix}, \end{aligned}$$

the nonlinear system (3.28) ($E = I$) can be approximated by the following bilinear system:

$$\begin{aligned} \frac{d\mathbf{x}_\otimes}{dt} &= A_\otimes \mathbf{x}_\otimes + N_\otimes \mathbf{x}_\otimes u(t) + B_\otimes u(t), \\ \mathbf{y}(t) &= C_\otimes \mathbf{x}_\otimes. \end{aligned} \tag{3.31}$$

The derivation can be easily extended to multi-input systems with $B \in \mathbb{R}^{n \times n_I}$ being a matrix. After a few more calculations, the following bilinear system with multiple inputs can be obtained:

$$\begin{aligned} \frac{d\mathbf{x}_\otimes}{dt} &= A_\otimes \mathbf{x}_\otimes + \sum_{i=1}^{n_I} N_\otimes^{(i)} \mathbf{x}_\otimes u_i(t) + B_\otimes u(t), \\ \mathbf{y}(t) &= C_\otimes \mathbf{x}_\otimes, \end{aligned} \tag{3.32}$$

where $u(t) := (u_1(t), \dots, u_{n_I}(t))^T$, $B := (\mathbf{b}_1, \dots, \mathbf{b}_{n_I})$ and

$$B_\otimes = \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix}, \quad N_\otimes^{(i)} = \begin{pmatrix} 0 & 0 \\ \mathbf{b}_i \otimes I + I \otimes \mathbf{b}_i & 0 \end{pmatrix}.$$

Given a bilinear system with E singular, several modified MOR schemes are proposed in [1, 12, 33, 34], but will not be discussed in this chapter due to space limitations. We can see that the above bilinear system is of much larger state-space dimension than the original nonlinear system (3.28). In the following we will introduce the process of constructing the projection matrix V for MOR.

Once the nonlinear system is approximated by the bilinear system (3.31) or (3.32), there are several choices for applying MOR. Multi-moment-matching methods extend the moment-matching methods for linear systems to bilinear systems by studying the transfer function of the bilinear system. Gramian-based bilinear MOR methods construct the matrices W and V by exploring the Gramian matrices of the bilinear systems. We focus on multi-moment-matching methods in this chapter.

3.4.2.1 Constructing W and V

Multivariate transfer functions and multi-moment-matching

The bilinearization MOR methods in [5, 46, 23] construct the matrices $W, V, W = V$ by approximating the transfer function of the bilinear system. Note that only SISO systems are considered in [5, 46]. For MIMO systems, the expression of the transfer function will be different; see [43]. In [43], a method similar to that in [46] was extended to MIMO systems. In the following description, we consider only SISO bilinear systems.

Under the assumption $E = I$, the identity matrix, the output response of the bilinear system (3.31) can be expressed by a Volterra series [52],

$$y(t) = \sum_{k=1}^{\infty} y_k(t),$$

where $y_k(t)$ is described in (3.33) and (3.34). In (3.34), $h_k^{(\text{reg})}$ is called the *regular kernel of degree k* . The multivariate Laplace transform of this kernel defines the k th multivariate transfer function $H_k^{(\text{reg})}$ in (3.35). We have

$$y_k(t) = \int_0^t \int_0^{t_1} \dots \int_0^{t_{k-1}} h_k^{(\text{reg})}(t_1, \dots, t_k) u(t - t_1 - \dots - t_k) \dots u(t - t_k) dt_k \dots dt_1, \quad (3.33)$$

$$h_k^{(\text{reg})}(t_1, \dots, t_k) = C_{\otimes} e^{A_{\otimes} t_k} N_{\otimes} e^{A_{\otimes} t_{k-1}} \dots N_{\otimes} e^{A_{\otimes} t_1} B_{\otimes}, \quad (3.34)$$

$$H_k^{(\text{reg})}(s_1, \dots, s_k) = C_{\otimes} (s_k I - A_{\otimes})^{-1} N_{\otimes} (s_{k-1} I - A_{\otimes})^{-1} \dots N_{\otimes} (s_1 I - A_{\otimes})^{-1} B_{\otimes}. \quad (3.35)$$

By using the Neumann expansion,

$$(I - sA_{\otimes}^{-1})^{-1} = I + sA_{\otimes}^{-1} + s^2 A_{\otimes}^{-2} + s^3 A_{\otimes}^{-3} + \dots,$$

$H_k^{(\text{reg})}(s_1, s_2, \dots, s_k)$ can be expanded into a multivariable Maclaurin series in (3.36), with the so-called k th-order multi-moments $m(l_1, \dots, l_k)$ being defined in (3.37). We have

$$H_k^{(\text{reg})}(s_1, \dots, s_k) = \sum_{l_k=1}^{\infty} \dots \sum_{l_1=1}^{\infty} m(l_1, \dots, l_k) s_1^{l_1-1} s_2^{l_2-1} \dots s_k^{l_k-1}, \quad (3.36)$$

$$m(l_1, \dots, l_k) = (-1)^k C_{\otimes} A_{\otimes}^{-l_k} N_{\otimes} \dots A_{\otimes}^{-l_2} N_{\otimes} A_{\otimes}^{-l_1} B_{\otimes}, \quad l_1, \dots, l_k = 1, 2, \dots \quad (3.37)$$

Deriving W and V

In [23, 7], the BICOMB method constructs the projection matrix V from a series of Krylov subspaces in the following steps:

$$\text{range}(V^{(1)}) = K_{q_1}(A_{\otimes}^{-1}, A_{\otimes}^{-1} B_{\otimes}), \quad (3.38)$$

and for $j > 1$

$$\text{range}(V^{(j)}) = K_{q_j}(A_{\otimes}^{-1}, A_{\otimes}^{-1} N_{\otimes} V^{(j-1)}). \quad (3.39)$$

The final projection matrix V is

$$\text{range}(V) = \text{orth}\{V^{(1)}, \dots, V^{(J)}\}. \quad (3.40)$$

Here,

$$K_{q_j}(A^{-1}, R) := \{R, A^{-1}R, \dots, A^{-q_j+1}R\}$$

is a block Krylov subspace generated by $R = A_{\otimes}^{-1}B_{\otimes}$, $j = 1$, or $R = A_{\otimes}^{-1}N_{\otimes}V^{(j-1)}$, $j > 1$. Note that each $V^{(k)}$ actually tries to match the multi-moments of the k th transfer function $H_k^{(\text{reg})}$; the multi-moment-matching property of the ROM can be found in, e. g., [23]. Applying $x_{\otimes} \approx Vz_{\otimes}$ to (3.31), the ROM of the nonlinear system (3.28) is given by

$$\begin{aligned} \frac{dz_{\otimes}}{dt} &= \hat{A}_{\otimes}z_{\otimes} + \hat{N}_{\otimes}z_{\otimes}u(t) + \hat{B}_{\otimes}u(t), \\ \hat{y}(t) &= \hat{C}_{\otimes}z_{\otimes}, \end{aligned} \quad (3.41)$$

where $\hat{A}_{\otimes} = V^T A_{\otimes} V$, $\hat{N}_{\otimes} = V^T N_{\otimes} V$, $\hat{B}_{\otimes} = V^T B_{\otimes}$, $\hat{C}_{\otimes} = C_{\otimes} V$. For simulation results of the BICOMB method, we refer to [7]. Algorithm 3.1 can be mildly modified to Algorithm 3.8 to compute $V^{(j)}$ in (3.38)–(3.40). Algorithm 3.8 computes a matrix V from the block Krylov subspaces defined as follows:

$$\text{range}(V^{(j)}) = K_{q_j}(M^{-1}, R_j), \quad j = 1, \dots, J. \quad (3.42)$$

The final projection matrix V is computed as in (3.40). Let $M = A_{\otimes}$, $R_1 = A_{\otimes}^{-1}B_{\otimes}$ and $R_j = A_{\otimes}^{-1}N_{\otimes}V^{(j-1)}$ for $j > 1$, Algorithm 3.8 can be used to compute the matrix V in (3.40) for the ROM (3.41) of the bilinear system.

3.4.3 Variational analysis method

The third set of nonlinear MOR methods [11, 27, 37, 42, 47] originates from variational analysis of nonlinear system theory [52].

3.4.3.1 Methods using polynomial approximation

In [27, 42, 47], the original nonlinear system is first approximated by a polynomial system, then variational analysis is applied to the polynomial system to obtain a reduced polynomial system. In the following, we describe the method developed in [27]. Its main difference from the method in [47] is the construction of the projection matrices V_2 and V_3 , and will be explained later.

Algorithm 3.8: Compute V in (3.40).

Input: Matrices of the block Krylov subspace in (3.42): $M, R_j, j = 1, 2, \dots, J$.

Output: Matrix V in (3.40) with orthonormal columns.

```

1: Initialize  $a_1 = 0, a_2 = 0, sum = 0$ .
2: for  $j = 1, \dots, J$  do
3:   if  $R_j$  is a matrix then
4:     Orthogonalize the columns in  $R_j$  using a modified Gram–Schmidt
       process:  $[v_1, v_2, \dots, v_{m_j}] = \text{orth}\{R_j\}$ .
5:      $sum = m_j$ . ( $m_j$  is the number of remaining columns after deflation.)
6:   else
7:     Compute the first column in  $V_j$ :  $v_1 = R_j / \|R_j\|_2$ . ( $R_j$  is a vector.)
8:      $sum = 1$ .
9:   end if
10:  for  $i = 1, 2, \dots, q_j - 1$  do
11:     $a_2 = sum$ .
12:    if  $a_1 = a_2$  then
13:      break, go to 2.
14:    else
15:      for  $d = a_1 + 1, \dots, a_2$  do
16:         $w = Mv_d, col = sum + 1$ .
17:        for  $k = 1, 2, \dots, col - 1$  do
18:           $h = v_k^T w, w = w - hv_k$ .
19:        end for
20:        if  $\|w\|_2 > \varepsilon$  (a small value indicating deflation, e. g.,  $\varepsilon = 10^{-7}$ ) then
21:           $v_{col} = \frac{w}{\|w\|_2}, sum = col$ .
22:        end if
23:      end for ( $d$ )
24:    end if
25:     $a_1 = a_2$ .
26:  end for ( $i$ )
27:   $V_j = [v_1, \dots, v_{sum}]$ ,
28: end for ( $j$ )
29: Orthogonalize the columns in  $[V_1, \dots, V_J]$  by the modified Gram–Schmidt process
    to obtain  $V$ , i. e.  $V := \text{orth}\{V_1, \dots, V_J\}$ .

```

With the power series expansion of $\mathbf{f}(\mathbf{x}(t))$ in (3.29), the original nonlinear system (3.28) is first approximated by a degree-2 polynomial system

$$\begin{aligned} \frac{d\mathbf{x}(t)}{dt} &= A_1 \mathbf{x}(t) + A_2 (\mathbf{x}(t) \otimes \mathbf{x}(t)) + B u(t), \\ \mathbf{y}(t) &= C \mathbf{x}(t), \end{aligned} \tag{3.43}$$

or by a degree-3 polynomial system

$$\begin{aligned}\frac{d\mathbf{x}(t)}{dt} &= A_1\mathbf{x}(t) + A_2(\mathbf{x}(t) \otimes \mathbf{x}(t)) \\ &\quad + A_3(\mathbf{x}(t) \otimes \mathbf{x}(t) \otimes \mathbf{x}(t)) + Bu(t), \\ \mathbf{y}(t) &= C\mathbf{x}(t).\end{aligned}\tag{3.44}$$

Consider the response of (3.28) to a variation of the input $au(t)$,

$$\begin{aligned}\frac{d\mathbf{x}(t)}{dt} &= f(\mathbf{x}(t)) + B(au(t)), \\ \mathbf{y}(t) &= C\mathbf{x}(t),\end{aligned}\tag{3.45}$$

where α is an arbitrarily small-valued variable. Assuming that the response to $u(t) = 0$ is $\mathbf{x}(t) = \mathbf{0}$ (in [52], it is called a forced response), then $\mathbf{x}(t)$, as a function of α , can be expanded into a power series of α around $\alpha_0 = 0$,

$$\mathbf{x}(t) = \alpha\mathbf{x}_1(t) + \alpha^2\mathbf{x}_2(t) + \alpha^3\mathbf{x}_3(t) + \cdots,\tag{3.46}$$

where the first term of the series is $\mathbf{x}_0(t) = \mathbf{x}(t, \alpha_0 = 0) = \mathbf{0}$, since when $\alpha_0 = 0$, $\alpha_0 u(t) = 0$. The corresponding response $\mathbf{x}(t, \alpha_0 = 0) = \mathbf{0}$ is then removed from the above expansion. Substituting both (3.46) and (3.29) into the right hand side and (3.46) into the left hand side of (3.45), we get

$$\begin{aligned}\alpha\frac{d\mathbf{x}_1(t)}{dt} + \alpha^2\frac{d\mathbf{x}_2(t)}{dt} + \alpha^3\frac{d\mathbf{x}_3(t)}{dt} + \cdots &= \alpha A_1\mathbf{x}_1(t) \\ &\quad + \alpha^2[A_1\mathbf{x}_2(t) + A_2(\mathbf{x}_1(t) \otimes \mathbf{x}_1(t))] + \cdots + B(au(t)).\end{aligned}$$

Since this equation holds for all α , coefficients of powers of α can be equated. This gives the variational equations:

$$\frac{d\mathbf{x}_1(t)}{dt} = A_1\mathbf{x}_1(t) + Bu(t),\tag{3.47}$$

$$\frac{d\mathbf{x}_2(t)}{dt} = A_1\mathbf{x}_2(t) + A_2(\mathbf{x}_1(t) \otimes \mathbf{x}_1(t)),\tag{3.48}$$

$$\begin{aligned}\frac{d\mathbf{x}_3(t)}{dt} &= A_1\mathbf{x}_3(t) + A_2(\mathbf{x}_1(t) \otimes \mathbf{x}_2(t) + \mathbf{x}_2(t) \otimes \mathbf{x}_1(t)) \\ &\quad + A_3(\mathbf{x}_1(t) \otimes \mathbf{x}_1(t) \otimes \mathbf{x}_1(t)),\end{aligned}\tag{3.49}$$

⋮

It is worth pointing out that the assumptions on the forced response can be relaxed, and similar variational equations of $\mathbf{x}_\delta = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$ can be derived. Here, $\hat{\mathbf{x}}(t)$ is the response to a certain input $\hat{u}(t)$ for a fixed initial state $\mathbf{x}(t = 0) = \mathbf{x}_0^*$. For a detailed discussion; see Section 3.4 in [52].

Deriving W and V

We notice that all of these variational equations are linear systems of order n for the vectors of the unknowns $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, \dots , respectively. Since $\mathbf{x}(t)$ is a linear combination of $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, \dots (see (3.46)), it is in the subspace spanned by $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, \dots . The projection matrix V can be computed from the subspace containing $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, \dots .

Building upon this observation, the method in [27] constructs V based on the linear variational equations (3.47)–(3.49) rather than from the nonlinear system. From the moment-matching MOR for linear systems, a projection matrix V_1 for $\mathbf{x}_1(t)$ of the first linear system (3.47) is constructed as

$$\text{range}(V_1) = \text{span}\{A_1^{-1}B, A_1^{-2}B, \dots, A_1^{-q_1}B\}. \quad (3.50)$$

Then $\mathbf{x}_1(t)$ can be approximated by $\mathbf{x}_1(t) \approx V_1 \mathbf{z}_1(t)$. A projection matrix V_2 for $\mathbf{x}_2(t)$ of the second linear system (3.48) is similarly constructed by

$$\text{range}(V_2) = \text{span}\{A_1^{-1}A_2, A_1^{-2}A_2, \dots, A_1^{-q_2}A_2\}, \quad (3.51)$$

such that $\mathbf{x}_2(t) \approx V_2 \mathbf{z}_2(t)$. A projection matrix V_3 for $\mathbf{x}_3(t)$ in (3.49) can be derived in a similar way. From (3.46), we have

$$\mathbf{x}(t) \approx \alpha V_1 \mathbf{z}_1(t) + \alpha^2 V_2 \mathbf{z}_2(t),$$

or

$$\mathbf{x}(t) \approx \alpha V_1 \mathbf{z}_1(t) + \alpha^2 V_2 \mathbf{z}_2(t) + \alpha^3 V_3 \mathbf{z}_3(t),$$

which indicates that the solution $\mathbf{x}(t)$ to (3.43) or (3.44) can be approximated by a linear combination of the columns of V_1 , V_2 or V_1 , V_2 , V_3 . Therefore the final projection matrix V can be computed as

$$\text{range}(V) = \text{orth}\{V_1, \dots, V_J\}, \quad J = 2 \text{ or } 3. \quad (3.52)$$

The ROM is thus derived from the polynomial system (3.43) or (3.44) as follows:

$$\begin{aligned} \frac{d\mathbf{z}(t)}{dt} &= V^T A_1 V \mathbf{z}(t) + V^T A_2 (V \mathbf{z}(t) \otimes V \mathbf{z}(t)) + V^T B u(t), \\ y(t) &= C V \mathbf{z}(t), \end{aligned} \quad (3.53)$$

or

$$\begin{aligned} \frac{d\mathbf{z}(t)}{dt} &= V^T A_1 V \mathbf{z}(t) + V^T A_2 (V \mathbf{z}(t) \otimes V \mathbf{z}(t)) \\ &\quad + V^T A_3 (V \mathbf{z}(t) \otimes V \mathbf{z}(t) \otimes V \mathbf{z}(t)) + V^T B u(t), \\ y(t) &= C V \mathbf{z}(t). \end{aligned} \quad (3.54)$$

The advantage of this method is that it has the flexibility of using a more accurate polynomial system (3.44) to approximate the original nonlinear system. It is possible that for the quadratic method, the system (3.30) can also be replaced by a more accurate polynomial system. However, the projection matrix V computed by the quadratic method might be less accurate than the matrix V in (3.52), because it is computed using only the linear part of the nonlinear system. For an approximation of the original nonlinear system (3.28), the bilinear system is less accurate than the polynomial system (3.44). Moreover, since the bilinear system is derived by approximating the nonlinear function $\mathbf{f}(\mathbf{x})$ by its power series expansion up to second order, the projection matrix V also only uses the information of the series expansion of $\mathbf{f}(\mathbf{x})$ at most to the second order, which is less accurate than the matrix V computed by the variational analysis method.

Again, let $M = A_1$, $R_1 = A_1^{-1}B$, $R_2 = A_1^{-1}A_2$ in (3.50), (3.51), then Algorithm 3.8 can be used to compute V in (3.52). Since there are many columns in A_2 , it is not possible to use all the columns. Instead, one may take only the first several columns, e. g., $R_2 = A_1^{-1}A_2(:, 1:q)$, $q \ll n$, where MATLAB notation is used.

In [47], the second projection matrix \tilde{V}_2 is constructed from the approximate system by replacing \mathbf{x}_1 with $V_1\mathbf{z}_1$ in (3.48) to get

$$\frac{d\mathbf{x}_2(t)}{dt} = A_1\mathbf{x}_2(t) + A_2(V_1\mathbf{z}_1(t) \otimes V_1\mathbf{z}_1(t)).$$

Then

$$\text{range}(\tilde{V}_2) = \text{span}\{A_1^{-1}A_2(V_1 \otimes V_1), A_1^{-2}A_2(V_1 \otimes V_1), \dots, A_1^{-q_2}A_2(V_1 \otimes V_1)\}. \quad (3.55)$$

The advantage of this approach is that there are much fewer columns in $A_2(V_1 \otimes V_1)$ than in A_2 in (3.51). Thus, \tilde{V}_2 matches more moments than V_2 in (3.51) if the matrices have the same number of columns. However, \tilde{V}_2 only matches approximate moments because the input matrix A_2 in (3.48) is approximated by $A_2(V_1 \otimes V_1)$ in (3.55). Therefore, although \tilde{V}_2 matches more moments, its accuracy is impaired by the approximate moments. The accuracy of the two methods is compared in [7].

At the end of this subsection, we would like to mention another method [42] which is based on both the Volterra series expansion of the output response and variational analysis. In [42], the original system (3.28) is approximated by the polynomial system in (3.44). Then the Volterra series representation of the output response of the polynomial system is employed to introduce the nonlinear transfer functions of (3.44). The k th-order nonlinear transfer function is similar to the k th transfer function $H_k^{(\text{reg})}(s_1, s_2, \dots, s_k)$ for the bilinear system. The projection matrix V is constructed based on the moments of the nonlinear transfer functions. Instead of performing the Laplace transform of the Volterra kernels as in (3.35), the nonlinear transfer functions are computed from the variational linear systems (3.47)–(3.49), whose transfer functions are equivalent with the first-order, second-order and third-order nonlinear

transfer functions, respectively. The basic idea of [42] is quite similar to the methods in [5] and [46]. The main difference is that [5] and [46] are based on a bilinear approximation of the original nonlinear system, whereas [42] is based on the more accurate approximation (3.44).

3.4.3.2 Methods based on quadratic–bilinearization

All those previous nonlinear model reduction methods first approximate the nonlinear function $\mathbf{f}(\mathbf{x}(t))$ by a polynomial, then reduce the approximate polynomial system to a small dimension. When the function $\mathbf{f}(\mathbf{x}(t))$ is weakly nonlinear, it is usually sufficient to approximate it by a degree-2 polynomial or degree-3 polynomial. Meanwhile, when $\mathbf{f}(\mathbf{x}(t))$ is strongly nonlinear, the low-degree polynomial approximation is not accurate. It is possible to employ higher order polynomials to improve the accuracy, but with much more complexity. Furthermore, the storage requirement for higher order polynomials is prohibitive if the matrix dimension is very large. Therefore, these methods are more suitable for weakly nonlinear systems.

The methods based on quadratic–bilinearization provide a solution to the above issues of polynomial approximation. Instead of approximating the nonlinear part $\mathbf{f}(\mathbf{x})$ by a polynomial function, equivalent transformations are applied to the nonlinear system in (3.28). The nonlinear system is first “lifted” to a polynomial system by adding polynomial algebraic equations or by taking Lie derivatives and adding more differential equations. The polynomial system is then transformed into a quadratic–bilinear system by either adding quadratic algebraic equations or taking Lie derivatives again. No accuracy is lost during the transformations. The detailed explanation can be found in [36, 37].

The equivalent quadratic–bilinear system is

$$G_0 \dot{\tilde{\mathbf{x}}} = G_1 \tilde{\mathbf{x}} + G_2 (\tilde{\mathbf{x}} \otimes \tilde{\mathbf{x}}) + D_1 \tilde{\mathbf{x}} u + D_2 (\tilde{\mathbf{x}} \otimes \tilde{\mathbf{x}}) u + \tilde{B} u(t), \quad (3.56)$$

where $\tilde{\mathbf{x}}$ is the lifted state vector after more state variables are added to the state vector \mathbf{x} . Notice that in [36, 37], the system (3.56) is called quadratic-linear differential algebraic equation (QLDAE). However, the system above obviously includes the bilinear term $D_1 \tilde{\mathbf{x}} u$ and the quadratic–bilinear term $D_2 (\tilde{\mathbf{x}} \otimes \tilde{\mathbf{x}}) u$. Therefore, the notion quadratic–bilinear differential algebraic equations (QBDAEs) introduced in [11] is used in this paper.

Once the QBDAEs are derived after several steps of transformations, the variational analysis (3.45)–(3.49) in the previous subsection can be applied to the QBDAEs. The projection matrix V can also be computed likewise. Then a Galerkin projection can be applied to (3.56) to get the reduced QBDAEs, which is considered as the ROM for the original nonlinear system in (3.28).

Recall that, from the second variational equation (3.48), the input matrix has many vectors, which makes the computation of the projection matrix V_2 tricky.

In [36, 37], a different way of computing the projection matrix V is proposed based on the transfer functions of the QBDAEs (3.56). The expression of the transfer functions of the QBDAEs can be originally found in [52]. For example, assuming for simplicity $G_0 = I$, the first two transfer functions are

$$\begin{aligned} H_1(s) &= L^T (sI - G_1)^{-1} B, \\ H_2(s_1, s_2) &= \frac{1}{2!} L^T [(s_1 + s_2)I - G_1]^{-1} \\ &\quad \times \{G_2[H_1(s_1) \otimes H_1(s_2) + H_1(s_2) \otimes H_1(s_1)] + D_1[H_1(s_1) + H_2(s_2)]\}. \end{aligned} \quad (3.57)$$

Using Taylor series expansions of the transfer functions, the matrix V can be recursively computed from the coefficients of the series expansions. The series expansions of H_1 and H_2 about zero (adaptation to nonzero expansion points is straightforward) are given as

$$\begin{aligned} H_1(s) &= L^T \sum_{k=0}^{\infty} A^k \tilde{B} s^k, \\ H_2(s_1, s_2) &= \frac{1}{2!} L^T \sum_{i=0}^k A^{k+1} (s_1 + s_2)^k \left\{ G_2 \left[\left(\sum_{k=0}^{\infty} A^k \tilde{B} s_1^k \right) \otimes \left(\sum_{k=0}^{\infty} A^k \tilde{B} s_2^k \right) \right. \right. \\ &\quad \left. \left. + \left(\sum_{k=0}^{\infty} A^k \tilde{B} s_2^k \right) \otimes \left(\sum_{k=0}^{\infty} A^k \tilde{B} s_1^k \right) \right] \right. \\ &\quad \left. + D_1 \left[\sum_{k=0}^{\infty} A^k \tilde{B} s_1^k + \sum_{k=0}^{\infty} A^k \tilde{B} s_2^k \right] \right\}, \end{aligned}$$

where $A = G_1^{-1}$, $\tilde{B} = -G_1^{-1}B$. In [36, 37], the projection matrix V is constructed as

$$\begin{aligned} \text{range}(V_1) &= \text{span}\{A^i \tilde{B}, i \leq q\}, \\ \text{range}(V_2) &= \text{span}\{A^{i+1} D_1 A^j \tilde{B}, i + j \leq q\}, \\ \text{range}(V_3) &= \text{span}\{A^{i+1} G_2 (A^j \tilde{B}) \otimes (A^k \tilde{B}), i + j + k \leq q, k \leq j\}, \\ \text{range}(V) &= \text{span}\{V_1, V_2, V_3\}. \end{aligned} \quad (3.58)$$

It can be seen that, if the system matrix B is a vector, the Kronecker product $(A^j \tilde{B}) \otimes (A^k \tilde{B})$ is also a vector so that the construction of V_3 is easy. In general, if B has m columns, $(A^j \tilde{B}) \otimes (A^k \tilde{B})$ has m^2 columns. The number of columns in $(A^j \tilde{B}) \otimes (A^k \tilde{B})$ is still moderate if m is small. This is an advantage over the way of computing V through variational analysis.

Algorithm 3.8 can also be used to compute V in (3.58), where we need to let $M = G_1$, $R_1 = \tilde{B}$, $R_2 = D_1 V_1$, $R_3 = G_2 A^j V_2 \otimes V_2$. Note that in order to compute V_2 , we use V_1 instead of $A^j \tilde{B}$ in R_2 , since V_1 is already the basis of the subspace spanned by $A^j \tilde{B}$, $j \leq q$. Similarly, we use $V_2 \otimes V_2$ rather than $A^j \tilde{B} \otimes A^k \tilde{B}$. This way, we avoid the issues of how to choose proper values of j, k . When $A^j \tilde{B}$ is replaced by V_1 in R_2 , V_1 is a matrix instead of

a vector. However, usually there are a few columns in V_1 , which still keeps the column dimensions of R_2, R_3 moderate.

In [11], the method is extended to two-sided projection based on the transfer functions (3.57) of the QBDAEs. It is proved that by using two-sided projection, the reduced transfer function matches almost twice as many moments of the original transfer functions as with the one-sided projection used in [36, 37]. Simulation results also show better accuracy than the one-sided projection. However, the two-sided projection sometimes causes numerical instability, which may produce unstable reduced models [11].

Note that the subspace dimension in (3.58) will grow exponentially if the coefficients of the series expansion of the higher order transfer functions, e. g. $H_3(s_1, s_2, s_3)$, are also included to compute the projection matrix V . This easily leads to a ROM with no reduced number of equations. In [58], the higher order multivariate transfer functions $H_2(s_1, s_2), H_3(s_1, s_2, s_3), \dots$, are transformed to single- s transfer functions $H_2(s), H_3(s), \dots$ by association of variables without losing accuracy. The series expansion of $H_2(s)$ or $H_3(s)$ only depends on the single variable s , such that the exponential growth of the subspace dimension can be avoided. Compared with the method in [37], a more compact ROM with the same accuracy can be obtained. The theory on association of variables can be found in [52].

Recall that, if the original nonlinear system is a system of ODEs, the QBDAEs usually constitute a system of differential-algebraic equations after quadratic–bilinearization, i. e., G_0 could be singular. In general, it is still unclear how to determine the index of the QBDAEs which may cause problems when the ROM is solved.

3.4.3.3 Other variants

Algorithm IRKA has been extended to the *bilinear iterative rational Krylov algorithm (BIRKA)* in [10] to compute the ROMs of bilinear systems, which iteratively updates a set of interpolation points such that the ROM satisfies the necessary conditions of \mathcal{H}_2 -optimality. Upon convergence, the BIRKA method produces a ROM whose Volterra series interpolates that of the original bilinear system at the mirror images of the poles of the ROM. However, the computational cost of BIRKA for large-scale systems is high and it is also not possible to match higher-order derivatives. Regarding computational cost of BIRKA, efforts have been done in [12] to reduce the computational cost for some special systems.

3.4.4 Trajectory piece-wise linear method

The trajectory piece-wise linear method in [49, 50] is proposed to deal with strongly nonlinear systems. An error bound for this method is proposed in [51], where the

stability and passivity of the ROM are also discussed. The trajectory piece-wise linear method first linearizes the nonlinear function $\mathbf{f}(\mathbf{x}(t))$ at a number of linearization points \mathbf{x}_i , $i = 0, 1, 2, \dots, k$, then approximates $\mathbf{f}(\mathbf{x}(t))$ by the weighted sum of these linearizations, $\mathbf{f}(\mathbf{x}_i) + \mathbf{A}_i(\mathbf{x} - \mathbf{x}_i)$. Finally, the original nonlinear system is approximated by the following weighted sum of linear systems:

$$\begin{aligned} \frac{d\mathbf{x}(t)}{dt} &= \sum_{i=0}^{s-1} \tilde{w}_i(\mathbf{x}) \mathbf{f}(\mathbf{x}_i) + \sum_{i=0}^{s-1} \tilde{w}_i(\mathbf{x}) \mathbf{A}_i(\mathbf{x} - \mathbf{x}_i) + \mathbf{B}u(t), \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t). \end{aligned}$$

Once a projection matrix V is obtained, the ROM can be obtained using a Galerkin projection. In [49, 50], V is obtained by applying the moment-matching method to each linearized system.

The linearization points are chosen by selecting a training input and simulating the original nonlinear system. The procedure is simply as follows: (1) A linearized model around state \mathbf{x}_i (initially $i = 0$) is generated. (2) The original nonlinear system is simulated while $\min_{0 \leq j \leq i} \|\mathbf{x} - \mathbf{x}_j\| < \delta$, i. e. while the current state \mathbf{x} is close enough to any of the previous linearization points. (3) A new linearization point \mathbf{x}_{i+1} is taken as the first state violating $\|\mathbf{x} - \mathbf{x}_i\| < \delta$, then return to step (1). Note that in order to get the linearization points, the original full system has to be simulated. Instead of simulating the full system, a fast algorithm for computing an approximate trajectory is also proposed in the paper.

The weak point of this method is that training inputs have to be chosen. In general, it is unclear how to choose the optimal training inputs so that the trajectory represents the behavior of the state vector $\mathbf{x}(t)$. If the training inputs are chosen far away from the actual inputs, then the computed trajectory of the unknown vector will depart from the real behavior of the state vector $\mathbf{x}(t)$ and the ROM will lose accuracy. Computation of the weight functions \tilde{w}_i in the above linear system is also more or less heuristic. Some related papers based on piece-wise linear ideas are [15, 18, 19, 20, 53, 54, 55].

3.5 Extension to parametric nonlinear systems

Some of the above nonlinear MOR methods could be extended to deal with parametric nonlinear systems, though little relevant work has been done. Often the nonlinear system also involves parameter variations, i. e.,

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= \mathbf{f}(\mathbf{x}(t, \mu), \mu) + \mathbf{B}(\mu)u(t), \\ \mathbf{y}(t, \mu) &= \mathbf{C}(\mu)\mathbf{x}(t, \mu), \end{aligned} \tag{3.59}$$

where $\mu \in \mathbb{R}^p$ is the vector of geometrical or physical parameters. When $\mathbf{f}(\mathbf{x}(t, \mu), \mu)$ is mildly nonlinear, many of the above introduced methods can be extended to solving (3.59).

The nonlinear system in (3.59) can be transformed to a parametric bilinear system using the same technique as introduced in Section 3.4.2, so the resulting system could be considered as a linear parametric system, where the input $u(t)$ could be taken as an extra parameter. The PMOR-MM method can then be applied to obtain the ROM. Extension of both the quadratic method and the variational analysis approach introduced in Section 3.4.1 and Section 3.4.3 is straightforward. In the following, we discuss these extensions in more detail.

3.5.1 Quadratic PMOR

The quadratic method in Section 3.4.1 depends on the power series expansion of the nonlinear function $\mathbf{f}(\mathbf{x}(t))$. For parameter dependent $\mathbf{f}(\mathbf{x}(t, \mu), \mu)$, the corresponding power series expansion may be written as

$$\begin{aligned} \mathbf{f}(\mathbf{x}(t, \mu), \mu) &= \mathbf{f}(\mathbf{0}) + A_1(\mu)\mathbf{x}(t, \mu) + A_2(\mu)(\mathbf{x}(t, \mu) \otimes \mathbf{x}(t, \mu)) \\ &\quad + A_3(\mu)(\mathbf{x}(t, \mu) \otimes \mathbf{x}(t, \mu) \otimes \mathbf{x}(t, \mu)) + \dots \end{aligned} \quad (3.60)$$

The approximated quadratic system is

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= A_1(\mu)\mathbf{x}(t, \mu) + A_2(\mu)(\mathbf{x}(t, \mu) \otimes \mathbf{x}(t, \mu)) + \tilde{B}(\mu)\tilde{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu)\mathbf{x}(t, \mu), \end{aligned} \quad (3.61)$$

where $\tilde{B}(\mu) = [B(\mu), \mathbf{f}(\mathbf{0})]$, $\tilde{u}(t) = (u(t)^T, 1)^T$. We seek a projection matrix V , which is used to reduce the linear parametric system

$$\begin{aligned} E(\mu) \frac{d\mathbf{x}(t, \mu)}{dt} &= A_1(\mu)\mathbf{x}(t, \mu) + \tilde{B}(\mu)\tilde{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu)\mathbf{x}(t, \mu). \end{aligned} \quad (3.62)$$

Once V is computed from the linear system in (3.62), the ROM is then obtained by applying a Galerkin projection with V to the *quadratic* system in (3.63), i. e. the ROM of (3.59) is

$$\begin{aligned} V^T E(\mu) V \frac{d\mathbf{z}(t, \mu)}{dt} &= V^T A_1(\mu) V \mathbf{z}(t, \mu) + V^T A_2(\mu) V (\mathbf{z}(t, \mu) \otimes \mathbf{z}(t, \mu)) \\ &\quad + V^T \tilde{B}(\mu) \tilde{u}(t), \\ \mathbf{y}(t, \mu) &= C(\mu) V \mathbf{z}(t, \mu). \end{aligned} \quad (3.63)$$

Since V is computed from (3.62), the PMOR-MM method can be directly used to compute V . PMOR-MM is shown to be accurate for MOR of quadratic parametric systems [6, 26, 57], when the magnitude of the input signal is relatively small. The extension to two-sided (Petrov-Galerkin) projection is straightforward.

3.5.2 PMOR after bilinearization

Following the bilinearization method introduced in Section 3.4.2, the parametric nonlinear system in (3.59) can be approximated by a parametric bilinear system as

$$\begin{aligned}\frac{dx_{\otimes}}{dt} &= A_{\otimes}(\mu)x_{\otimes} + N_{\otimes}(\mu)x_{\otimes}u(t) + B_{\otimes}(\mu)u(t), \\ \mathbf{y}(t, \mu) &= C_{\otimes}(\mu)x_{\otimes}.\end{aligned}$$

By considering $u(t)$ associated with the bilinear term $N_{\otimes}(\mu)x_{\otimes}u(t)$ to be an extra parameter, say $u(t) = \mu_{m+1}(t)$, the above system can be viewed as a linear parametric system,

$$\begin{aligned}E_{\otimes}(\mu)\frac{dx_{\otimes}}{dt} &= A_{\otimes}(\mu)x_{\otimes} + N_{\otimes}(\mu)x_{\otimes}\mu_{m+1}(t) + B_{\otimes}(\mu)u(t), \\ \mathbf{y}(t, \mu) &= C_{\otimes}(\mu)x_{\otimes}.\end{aligned}\tag{3.64}$$

Note that the parameter $\mu_{m+1}(t)$ is time-varying. Strictly speaking, the PMOR-MM method in Section 3.3 cannot be directly used, since the Laplace transform of (3.64) cannot be applied as in (3.15). However, it is found that directly applying PMOR-MM to some systems with time-varying parameters [24, 41] or to the bilinear system [2] may also produce accurate results.

3.5.3 PMOR based on variational analysis

The variational analysis method in Section 3.4.3 can easily be extended to deal with parametric nonlinear systems. It can be seen that from the power series expansion of $\mathbf{f}(\mathbf{x}(t, \mu), \mu)$ in (3.60), one can obtain the parametric variational equations

$$\frac{d\mathbf{x}_1(t, \mu)}{dt} = A_1(\mu)\mathbf{x}_1(t, \mu) + B(\mu)u(t),\tag{3.65}$$

$$\frac{d\mathbf{x}_2(t, \mu)}{dt} = A_1(\mu)\mathbf{x}_2(t, \mu) + A_2(\mu)(\mathbf{x}_1(t, \mu) \otimes \mathbf{x}_1(t, \mu)),\tag{3.66}$$

$$\begin{aligned}\frac{d\mathbf{x}_3(t, \mu)}{dt} &= A_1(\mu)\mathbf{x}_3(t, \mu) + A_2(\mu)(\mathbf{x}_1(t, \mu) \otimes \mathbf{x}_2(t, \mu) + \mathbf{x}_2(t, \mu) \otimes \mathbf{x}_1(t, \mu)) \\ &\quad + A_3(\mu)(\mathbf{x}_1(t, \mu) \otimes \mathbf{x}_1(t, \mu) \otimes \mathbf{x}_1(t, \mu)),\end{aligned}\tag{3.67}$$

⋮

For each linear parametric system in (3.65)–(3.67), the PMOR-MM method can be used to compute the projection matrices V_1, V_2, V_3 corresponding to (3.65)–(3.67), respectively. The final projection matrix V for the parametric nonlinear system is then the combination of V_1, V_2, V_3 , and can be computed following (3.52). The ROM is of a form

similar to (3.54):

$$\begin{aligned}\frac{d\mathbf{z}(t, \mu)}{dt} &= V^T A_1(\mu) V \mathbf{z}(t, \mu) + V^T A_2(\mu) (V \mathbf{z}(t, \mu) \otimes V \mathbf{z}(t, \mu)) \\ &\quad + V^T A_3(\mu) (V \mathbf{z}(t, \mu) \otimes V \mathbf{z}(t, \mu) \otimes V \mathbf{z}(t, \mu)) + V^T B(\mu) u(t), \\ \mathbf{y}(t, \mu) &= C(\mu) V \mathbf{z}(t, \mu).\end{aligned}$$

3.6 Conclusions

This chapter reviews moment-matching methods for MOR of a wide range of systems, including standard linear time-invariant (LTI) systems, parametric LTI systems, nonlinear non-parametric and nonlinear parametric systems. Sufficient algorithms are provided to enable most of the methods to be realizable and the results in the literature to be reproducible. Some algorithms, e. g., Algorithms 3.1–3.2 and Algorithm 3.8 have not appeared elsewhere. The discussions in some sections, e. g. Sections 3.3.2, 3.3.3, and 3.5 are also new. It has been demonstrated in numerous publications that moment-matching methods are powerful MOR tools for many systems and are useful in many application areas.

Bibliography

- [1] M. I. Ahmad, P. Benner, and P. Goyal. Krylov subspace-based model reduction for a class of bilinear descriptor systems. *J. Comput. Appl. Math.*, 315:303–318, 2017.
- [2] M. I. Ahmad, P. Benner, and L. Feng. Interpolatory model reduction for quadratic–bilinear systems using error estimators. *Eng. Comput.*, 36(1):25–44, 2018.
- [3] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM Publications, Philadelphia, PA, 2005.
- [4] Z. Bai. Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. *Appl. Numer. Math.*, 43(1–2):9–44, 2002.
- [5] Z. Bai and D. Skoogh. A projection method for model reduction of bilinear dynamical systems. *Linear Algebra Appl.*, 415(2–3):406–425, 2006.
- [6] N. Banagaaya, P. Benner, and L. Feng. Parametric model order reduction for electro-thermal coupled problems with many inputs. In *Proceedings of European Consortium for Mathematics in Industry ECMI 2016: Progress in Industrial Mathematics at ECMI 2016*, volume 26, pages 263–270. Springer, 2018.
- [7] U. Baur, P. Benner, and L. Feng. Model order reduction for linear and nonlinear systems: a system-theoretic perspective. *Arch. Comput. Methods Eng.*, 21:331–358, 2014.
- [8] T. Bechtold, E. B. Rudnyi, and J. G. Korvink. Error indicators for fully automatic extraction of heat-transfer macromodels for MEMS. *J. Micromech. Microeng.*, 15(3):430–440, 2005.
- [9] P. Benner and T. Breiten. On H_2 -model reduction of linear parameter-varying systems. *Proc. Appl. Math. Mech.*, 11(1):805–806, 2011.
- [10] P. Benner and T. Breiten. Interpolation-based \mathcal{H}_2 -model reduction of bilinear control systems. *SIAM J. Matrix Anal. Appl.*, 33(3):859–885, 2012.

- [11] P. Benner and T. Breiten. Two-sided projection methods for nonlinear model order reduction. *SIAM J. Sci. Comput.*, 37(2):B239–B260, 2015.
- [12] P. Benner and P. Goyal. Multipoint interpolation of Volterra series and \mathcal{H}_2 -model reduction for a family of bilinear descriptor systems. *Syst. Control Lett.*, 97:1–11, 2016.
- [13] P. Benner, S. Gugercin, and K. Willcox. A survey of model reduction methods for parametric systems. *SIAM Rev.*, 57(4):483–531, 2015.
- [14] A. Bodendiek and M. Bollhöfer. Adaptive-order rational Arnoldi-type methods in computational electromagnetism. *BIT Numer. Math.*, 54:357–380, 2014.
- [15] B. Bond and L. Daniel. Parameterized model order reduction of nonlinear dynamical systems. In *Proc. IEEE/ACM Conference on Computer-Aided Design, ICCAD-2005, November*, pages 487–494, 2005.
- [16] C.-T. Chen. *Linear System Theory and Design*. Oxford University Press, New York/Oxford, 1999.
- [17] L. Daniel, O. C. Siong, L. S. Chay, K. H. Lee, and J. White. A multiparameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 23(5):678–693, 2004.
- [18] N. Dong and J. Roychowdhury. Piecewise polynomial nonlinear model reduction. In *Proc. Design Automation Conference*, pages 484–489, 2003.
- [19] N. Dong and J. Roychowdhury. Automated extraction of broadly applicable nonlinear analog macromodels from SPICE-level descriptions. In *Proc. IEEE Custom Integrated Circuits Conference, 2004*, pages 117–120, 2004.
- [20] N. Dong and J. Roychowdhury. Automated nonlinear macromodelling of output buffers for high-speed digital applications. In *Proc. Design Automation Conference*, pages 51–56, 2005.
- [21] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 14(5):639–649, 1995.
- [22] L. Feng, A. C. Antoulas, and P. Benner. Automatic generation of reduced order models for linear parametric systems. In *European Conference on Mathematics for Industry (ECMI). The European Consortium for Mathematics in Industry*, volume 22, 2016.
- [23] L. Feng and P. Benner. A note on projection techniques for model order reduction of bilinear systems. In *Numerical Analysis and Applied Mathematics: International Conference of Numerical Analysis and Applied Mathematics*, pages 208–211, 2007.
- [24] L. Feng and P. Benner. A robust algorithm for parametric model order reduction based on implicit moment matching. In *Reduced Order Methods for Modeling and Computational Reduction. MS&A Series*, volume 9, pages 159–186. Springer, Berlin, Heidelberg, New York, 2014.
- [25] L. Feng, J. G. Korvink, and P. Benner. A fully adaptive scheme for model order reduction based on moment-matching. *IEEE Trans. Compon. Packag. Manuf. Technol.*, 5(12):1872–1884, 2015.
- [26] 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.*, 6(1):1–10, 2016.
- [27] L. Feng, X. Zeng, C. Chiang, D. Zhou, and Q. Fang. Direct nonlinear order reduction with variational analysis. In *Proc. Design Automation and Test in Europe*, pages 1316–1321, 2004.
- [28] L. Feng, A. C. Antoulas, and P. Benner. Some a posteriori error bounds for reduced order modelling of (non-)parametrized linear systems. *ESAIM: M2AN*, 51(6):2157–2158, 2017.
- [29] L. Feng and P. Benner. A robust algorithm for parametric model order reduction. In *Proceedings in Applied Mathematics and Mechanics (ICIAM)*, volume 7(1), pages 10215.01–10215.02, 2007.
- [30] L. Feng, P. Benner, P. Meuris, and W. Schoenmaker. Parametric and reduced-order modeling for the thermal analysis of nanoelectronic structures. In *Proceedings of SCEE-2014, Scientific Computing in Electrical Engineering*, pages 115–163. Springer, 2015.

- [31] R. W. Freund. Model reduction methods based on Krylov subspaces. *Acta Numer.*, 12:267–319, 2003.
- [32] K. Gallivan, E. Grimme, and P. Van Dooren. Asymptotic waveform evaluation via a Lanczos method. *Appl. Math. Lett.*, 7(5):75–80, 1994.
- [33] P. Goyal, M. I. Ahmad, and P. Benner. Model reduction of quadratic–bilinear descriptor systems via Carleman bilinearization. In *Proc. European Control Conf. 2015, Linz*, pages 1177–1182. IEEE, 2015.
- [34] P. Goyal and P. Benner. An iterative model order reduction scheme for a special class of bilinear descriptor systems appearing in constraint circuit simulation. In *ECCOMAS Congress 2016, VII European Congress on Computational Methods in Applied Sciences and Engineering*, volume 2, pages 4196–4212, 2016.
- [35] E. J. Grimme. Krylov projection methods for model reduction. PhD thesis, Univ. Illinois, Urbana-Champaign 1997.
- [36] C. Gu. QLMOR: A new projection-based approach for nonlinear model order reduction. In *Proc. International Conference on Computer-Aided Design*, November, pages 389–396, 2009.
- [37] C. Gu. QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 30(9):1307–1320, 2011.
- [38] S. Gugercin, A. C. Antoulas, and C. Beattie. \mathcal{H}_2 model reduction for large-scale linear dynamical systems. *SIAM J. Matrix Anal. Appl.*, 30(2):609–638, 2008.
- [39] U. Hetmaniuk, R. Tezaur, and C. Farhat. An adaptive scheme for a class of interpolatory model reduction methods for frequency response problems. *Int. J. Numer. Methods Eng.*, 93:1109–1124, 2013.
- [40] H-J. Lee, C.-Ch. Chu, and W-Sh. Feng. An adaptive-order rational Arnoldi method for model-order reductions of linear time-invariant systems. *Linear Algebra Appl.*, 235–261, 2006.
- [41] L. Feng, D. Koziol, E. B. Rudnyi, and J. G. Korvink. Parametric model reduction for fast simulation of cyclic voltammograms. *Sens. Lett.*, 4:165–173, 2006.
- [42] P. Li and L. T. Pileggi. NORM: Compact model order reduction of weakly nonlinear systems. In *Proc. Design Automation Conference*, pages 472–477, 2003.
- [43] Y. Lin, L. Bao, and Y. Wei. A model-order reduction method based on Krylov subspace for MIMO bilinear dynamical systems. *J. Appl. Math. Comput.*, 25(1–2):293–304, 2007.
- [44] A. Manzoni and F. Negri. Heuristic strategies for the approximation of stability factors in quadratically nonlinear parametrized PDEs. *Adv. Comput. Math.*, 41(5):1255–1288, 2015.
- [45] A. Odabasioglu, M. Celik, and L. T. Pileggi. PRIMA: passive reduced-order interconnect macromodeling algorithm. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 17(8):645–654, 1998.
- [46] J. R. Phillips. Projection frameworks for model reduction of weakly nonlinear systems. In *Proc. Design Automation Conference*, pages 184–189, 2000.
- [47] J. R. Phillips. Projection-based approaches for model reduction of weakly nonlinear time-varying systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 22(2):171–187, 2003.
- [48] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. Comput.-Aided Des.*, 9(4):352–366, 1990.
- [49] M. J. Rewieński. A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems. Ph.D. Thesis, Massachusetts Institute of Technology 2003.
- [50] M. J. Rewieński and J. White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 22(2):155–170, 2003.
- [51] M. J. Rewieński and J. White. Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations. *Linear Algebra Appl.*, 415(2–3):426–454, 2006.

- [52] W. J. Rugh. *Nonlinear System Theory*. The Johns Hopkins University Press, Baltimore, 1981.
- [53] S. Tiwary and R. Rutenbar. Scalable trajectory methods for on-demand analog macromodel extraction. In *Proc. Design Automation Conference*, pages 403–408, 2005.
- [54] D. Vasilyev, M. Rewienski, and J. White. A TBR-based trajectory piecewise-linear algorithm for generating accurate low-order models for nonlinear analog circuits and MEMS. In *Proc. Design Automation Conference*, pages 490–495, 2003.
- [55] T. Voß, R. Pulch, E. J. W. Maten, and A. El Guennouni. Trajectory piecewise linear approach for nonlinear differential-algebraic equations in circuit simulation. In *Scientific Computing in Electrical Engineering SCEE 2006, Mathematics in Industry*, volume 11, pages 167–174, Springer, Berlin, Heidelberg, 2007.
- [56] T. Wolf, H. Panzer, and B. Lohmann. Gramian-based error bound in model reduction by Krylov subspace methods. In *Proceedings of IFAC World Congress*, pages 3587–3591, 2011.
- [57] Y. Yue, L. Feng, P. Meuris, W. Schoenmaker, and P. Benner. Application of Krylov-type parametric model order reduction in efficient uncertainty quantification of electro-thermal circuit models. In *Progress In Electromagnetics Research Symposium (PIERS 2015)*, pages 379–384, 2015.
- [58] Y. Zhang, H. Liu, Q. Wang, N. Fong, and N. Wong. Fast nonlinear model order reduction via associated transforms of high-order Volterra transfer functions. In *Proc. Design Automation Conference*, pages 289–294, 2012.