MORLAB – The Model Order Reduction LABoratory

Peter Benner[*] Steffen W. R. Werner[†]

Abstract

For an easy use of model order reduction techniques in applications, software solutions are needed. In this paper, we describe the MORLAB, Model Order Reduction LABoratory, toolbox as an efficient implementation of model reduction techniques for dense, medium-scale linear time-invariant systems. Giving an introduction to the underlying programming principles of the toolbox, we show the basic idea of spectral splitting and present an overview about implemented model reduction techniques. Two numerical examples are used to illustrate different use cases of the MORLAB toolbox.

1 Introduction

For the modeling of natural processes as, e.g., fluid dynamics, chemical reactions or the behavior of electronic circuits, power or gas transportation networks, dynamical input-output systems are used

$$G : \begin{cases} 0 = f(x(t), Dx(t), \ldots, D^k x(t), u(t)), \\ y(t) = h(x(t), Dx(t), \ldots, D^k x(t), u(t)), \end{cases} \tag{1}$$

with states $x(t) \in \mathbb{R}^n$, inputs $u(t) \in \mathbb{R}^p$ and outputs $y(t) \in \mathbb{R}^p$. The operator D^j denotes the derivative or shift operator of order $j \in \mathbb{N}$ in case of underlying continuous- or discrete-time dynamics. Due to the demand for increasing the accuracy of models, the number of states describing (1) is drastically increasing and, consequently, there is a high demand for computational resources (time and memory) when using (1) in simulations or controller design. A solution to this problem is given by model order reduction, which aims for the construction of a surrogate model $\widehat{G}$, with a much smaller number of internal states $\hat{x}(t) \in \mathbb{R}^r$, $r \ll n$, which approximates the input-to-output behavior of (1) such that

$$\|y - \hat{y}\| \leq \mathrm{tol}\cdot\|u\|,$$

for an appropriately defined norm, a given tolerance tol and all admissible inputs u, where $\hat{y}$ is the output of the reduced-order system.

A software solution for model order reduction of dynamical systems is the **MORLAB**, **M**odel **O**rder **R**eduction **LAB**oratory, toolbox. Originating from [6], the toolbox is mainly developed as efficient open source implementation of established matrix equation-based model reduction methods. Nowadays, it is one of the most efficient model reduction toolboxes for dense, medium-scale, linear time-invariant systems, with an implementation compatible with MathWorks MATLAB and

[*]Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg, Germany.
E-mail: benner@mpi-magdeburg.mpg.de
Otto von Guericke University, Faculty of Mathematics, Universitätsplatz 2, 39106 Magdeburg, Germany.
E-mail: peter.benner@ovgu.de
[†]Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg, Germany.
E-mail: werner@mpi-magdeburg.mpg.de

Table 1: Code meta data of the latest MORLAB version [13].

name (shortname)	Model Order Reduction LABoratory (MORLAB)
version (release-date)	5.0 (2019-08-23)
identifier (type)	doi: 10.5281/zenodo.3332716 (doi)
authors	Peter Benner, Steffen W. R. Werner
orcids	0000-0003-3362-4103, 0000-0003-1667-4862
topic (type)	Model Reduction (Toolbox)
license (type)	GNU Affero General Public License v3.0 (open)
languages	MATLAB
dependencies	MATLAB ($\geq$ 2012b), Octave ($\geq$ 4.0.0)
systems	Linux, MacOS, Windows
website	`http://www.mpi-magdeburg.mpg.de/projects/morlab`

GNU Octave. In the latest version [13], MORLAB gives a large variety of balancing-based model reduction methods and also some non-projective methods. Most of those are not known to be implemented somewhere else. In contrast to other software solutions, the general philosophy of MORLAB is to work on invariant subspaces rather than with spectral decompositions or projections on hidden manifolds, which results in fast and accurate implementations. Mainly the two spectral projection methods, the matrix sign function and the right matrix pencil disk function, are used in the underlying implementations. Therefore, MORLAB is suited as backend source code for multi-step model reduction approaches, for example, using a pre-reduction step; see, e.g., [25, 35]. Additionally to model order reduction methods, the toolbox implements efficient matrix equation solvers, system-theoretic subroutines and evaluation routines to examine original and reduced-order systems in the frequency and time domain. Due to the brevity of the paper, the additional main features are not further considered in detail.

In this paper, we will describe the underlying principles and structures of the MORLAB toolbox and give some applications of the software. The meta data of the latest MORLAB version [13] can be found in Table 1. In the following, Section 2 starts with an introduction of the programming principles that were used in MORLAB. Afterwards, Section 3 gives the underlying ideas of the spectral splitting, on which the toolbox bases, followed by Section 4 with an overview about the implemented model reduction methods. In Section 5, two applications of using MORLAB as backend software are presented. The paper is concluded by Section 6.

2 Code design principles

The main aim of the MORLAB toolbox is to give efficient and comparable implementations of many different model reduction methods. Following certain design principles, which will be explained in more detail in the upcoming subsections, the following list of main features briefly summarizes the MORLAB toolbox.

Feature checklist

Open source and free	The toolbox is running under the GNU Affero General Public License v3.0 and is freely available on the project website and on Zenodo.
Fast and exact	Using spectral projection methods, the toolbox can outperform other established software in terms of accuracy and speed.
Unified framework	All model reduction routines share the same interface and allow for quick exchange and easy comparison between the methods.
Modular	Each subroutine can be called on its own by the user to be used and combined in varies ways.

Table 4: Currently supported system classes.

Class	System equations	Routine name
Continuous-time standard systems	$\dot{x}(t) = Ax(t) + Bu(t),$ $y(t) = Cx(t) + Du(t)$	`ct_ss`
Discrete-time standard systems	$x_{k+1} = Ax_k + Bu_k,$ $y_k = Cx_k + Du_k$	`dt_ss`
Continuous-time descriptor systems	$E\dot{x}(t) = Ax(t) + Bu(t),$ $y(t) = Cx(t) + Du(t)$	`ct_dss`
Discrete-time descriptor systems	$Ex_{k+1} = Ax_k + Bu_k,$ $y_k = Cx_k + Du_k$	`dt_dss`
Continuous-time second-order systems	$M\ddot{x}(t) = -Kx(t) - E\dot{x}(t) + B_u u(t),$ $y(t) = C_p x(t) + C_v \dot{x}(t) + Du(t)$	`ct_soss`

Portable	No binary extensions are required, which allows for running the toolbox with bare MATLAB or Octave installations.

In general, MORLAB uses spectral projection methods for all steps of the model reduction procedure. Fig. 1 shows the different stages in MORLAB from the full-order to the reduced-order model. First, the full-order model is decomposed into (at most) three subsystems that can usually be considered independently of each other for the application of model reduction techniques. This first main step, the additive system decomposition, is discussed in more detail in Section 3. Afterwards, the model reduction methods are applied to the resulting subsystems. An overview of those can be found in Section 4. At the end, the reduced subsystems are coupled for the resulting reduced-order model. Based on this basic workflow, the different design principles applied in MORLAB are explained in the following. For the sake of brevity, mainly the model reduction routines are considered.

2.1 Toolbox structure

The routines in MORLAB follow a strict structure and naming scheme to make them easy to find and interpret in terms of their objective. Describing first the general structure, the routines of the toolbox are divided by their purpose into the following subdirectories:

`checks/`	Contains subroutines that are used for internal checks of data, e.g., if the system structures fit to the model reduction methods.
`demos/`	Contains example scripts showing step-by-step explanations of the different main features of the toolbox.
`eqn_solvers/`	Contains the matrix equation solvers.
`evaluation/`	Contains functions to evaluate the full-order or reduced-order models in the time or frequency domain.
`mor/`	Contains the model reduction routines.
`subroutines/`	Contains auxiliary and system-theoretic routines that are used by the model reduction techniques, matrix equation solvers or evaluation functions.

Considering to the naming scheme of MORLAB, each function starts with `ml_` as assignment to the toolbox. This makes MORLAB routines easier to distinguish from other source codes and also allows for easy searching. Mainly the model reduction routines, but also some subroutines are

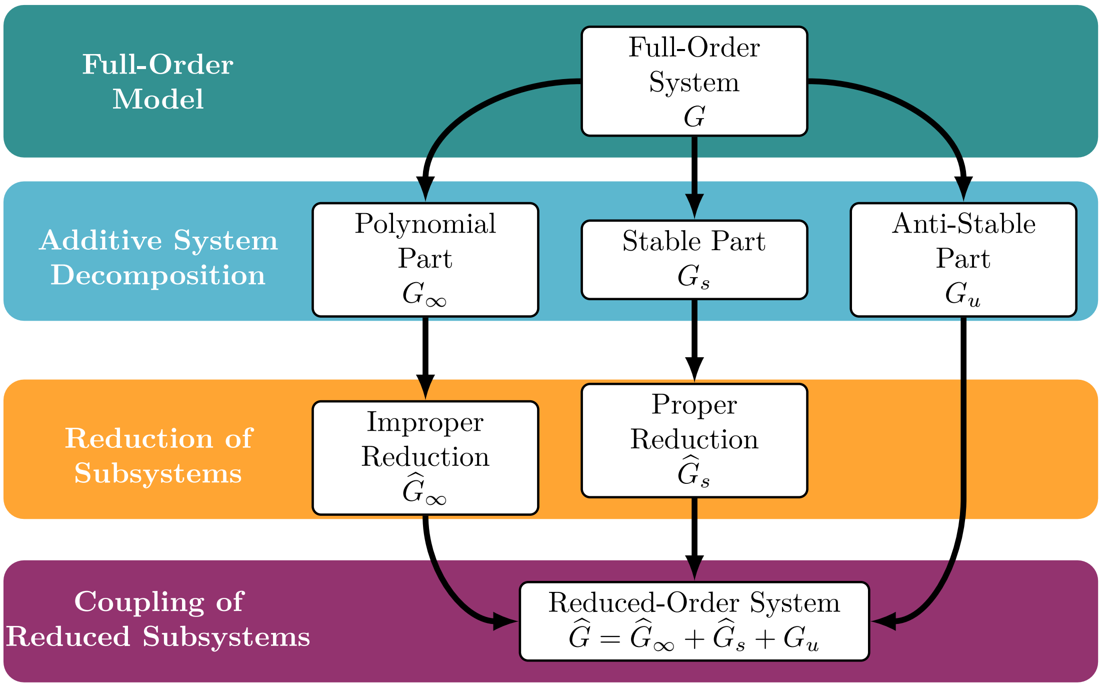

Full-Order Model
Full-Order System G
Additive System Decomposition
Polynomial Part G_∞
Stable Part G_s
Anti-Stable Part G_u
Reduction of Subsystems
Improper Reduction $\widehat{G}_\infty$
Proper Reduction $\widehat{G}_s$
Coupling of Reduced Subsystems
Reduced-Order System $\widehat{G} = \widehat{G}_\infty + \widehat{G}_s + G_u$

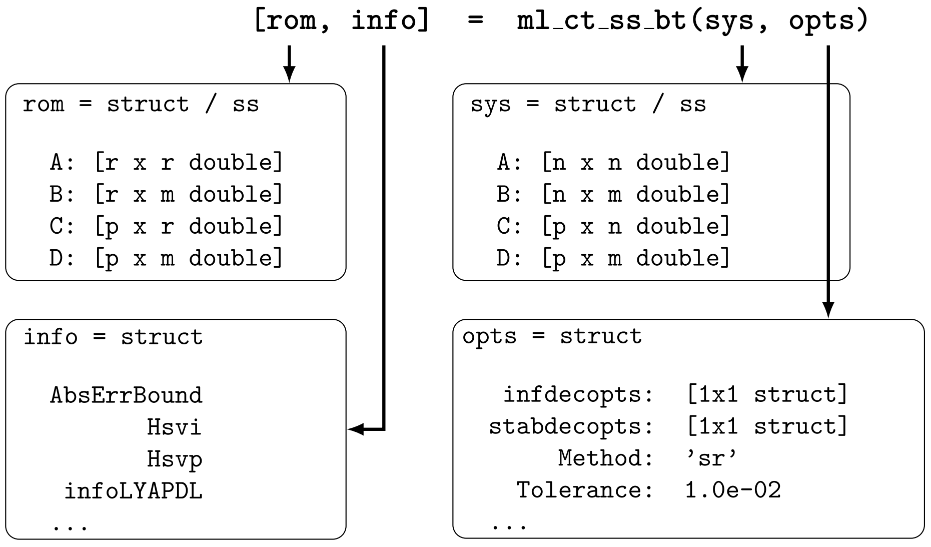
Figure 1: General MORLAB workflow.

additionally named after the system classes they can be applied to. Currently, there are routines for continuous- (`ct`) and discrete-time (`dt`) dynamical system with equations that describe standard (`ss`), descriptor (`dss`) or second-order state spaces (`soss`). The resulting different system classes, supported in the latest MORLAB version, are summarized in Table 4 with their names, system equations and the corresponding naming schemes.

2.2 Function interfaces

A typical function call in MORLAB can be seen in Fig. 2. From before, we know that the called function is a MORLAB routine for continuous-time standard systems (see Table 4). The actual function name, `bt`, stands for the balanced truncation method. Fig. 2 shows the principle idea in MORLAB to give an easy interface to the user. Here, `sys` contains the data of the original system, while `rom` gives the resulting reduced-order model in exactly the same format as the original model was given, indicating the purpose of using reduced-order models as surrogates for the original system. In general, MORLAB supports three different interfaces for model reduction methods. It is possible to pass directly the system matrices to the function (e.g., `ml_ct_ss_bt(A, B, C, D, opts)`) or to construct the system as an object by using the native data type `struct`, with appropriate naming of fields, or the state-space object (`ss`) introduced by the Control System Toolbox$^{\text{TM}}$ in MATLAB or the 'control' package in Octave. The latter format allows for easy interconnection to other model reduction software and also for using system-theoretic routines implemented in the two mentioned software libraries.

The second important part of the MORLAB interface for nearly all routines are the `opts` and `info` structs, as shown in Fig. 2. Supporting the feature of configurability, the `opts` struct allows the user the rearrangement of all computational parameters, which would be usually set by the function itself during runtime. In general, each MORLAB function that allows the user to change optional parameters for the computations has an `opts` struct for that purpose. As result, higher level routines can contain nested structs to change computational parameters of used subroutines. Fig. 3 shows an example `opts` struct for the `ml_ct_ss_bt`. This struct again contains entries ending on `opts` denoting also `opts` structs for subroutines that are called by the main function. Beside changing computational parameters, a second aim of the `opts` struct is the a priori determination of



```
[rom, info]  =  ml_ct_ss_bt(sys, opts)
```

```
rom = struct / ss

  A: [r x r double]
  B: [r x m double]
  C: [p x r double]
  D: [p x m double]
```

```
sys = struct / ss

  A: [n x n double]
  B: [n x m double]
  C: [p x n double]
  D: [p x m double]
```

```
info = struct

  AbsErrBound
         Hsvi
         Hsvp
   infoLYAPDL
  ...
```

```
opts = struct

   infdecopts:  [1x1 struct]
  stabdecopts:  [1x1 struct]
       Method:  'sr'
    Tolerance:  1.0e-02
  ...
```

Figure 2: Example function call of a model reduction routine in MORLAB.

```
       lyapdlopts: [ 1x1 struct | {ml_morlabopts('ml_lyapdl_sgn_fac')} ]
           Method: [ 'bfsr' | {'sr'} ]
            Order: [ positive integer | {min(10,length(Hsv)) + Nu} ]
 OrderComputation: [ 'order' | {'tolerance'} ]
    stabsignmopts: [ 1x1 struct | {ml_morlabopts('ml_signm')} ]
     stabsylvopts: [ 1x1 struct | {ml_morlabopts('ml_sylv_sgn')} ]
  StoreProjection: [ 1 | {0} ]
        Tolerance: [ nonnegative scalar | {1.0e-02} ]
       UnstabDim: [ integer | {-1} ]

For more details see ml_ct_ss_bt.
```

Figure 3: Example `opts` struct for the `ml_ct_ss_bt` function.

system information. For example, if a system is known to be stable, the additive decomposition into the stable and anti-stable subsystems can be turned off using the `opts` struct to avoid unnecessary computations. For easy application, only entries, which the user wants to change, need to be existing in the struct. Also, the toolbox comes with an option constructor (`ml_morlabopts`), which creates a complete but empty `opts` struct for a given function name. The consistent naming of optional parameters between different routines allows the easy reuse of `opts` structs for different functions.

The counterpart of the `opts` struct is the `info` struct. Here, information about the performance and results of the routine are collected. As for `opts`, the `info` struct can be nested as it contains structs starting with `info`, which give information about used subroutines. Also, this struct is used for optional outputs, e.g., projection matrices of a model reduction method can be stored in here.

2.3 Documentation

MORLAB comes with an extensive documentation that is accessible in several ways. Each routine has a complete inline documentation, which can be displayed by the `help` command, containing the syntax, description and literature references for background information. Besides, a complete overview about the existing MORLAB routines with short description can be generated by `help morlab`. As usual for MATLAB toolboxes, a full HTML documentation is provided in the toolbox and demo scripts can be used as a starting how-to to get into the main features of the toolbox.

3 Additive system decomposition approach

Most model order reduction methods are in a certain sense restricted with respect to the spectrum of the underlying system matrices, e.g., the classical balanced truncation method can only be applied to first-order systems with finite stable matrix pencils. Other software solutions use therefor either an eigendecomposition of the system matrices in the beginning or apply projections onto the hidden manifolds. In MORLAB, this problem is solved by working directly with the corresponding invariant subspaces of the matrix pencil. As shown in Fig. 1, this results in the additive decomposition of the full-order system into independent reducable subsystems, in the literature known as additive decomposition of the transfer function, which will be coupled at the end again. MORLAB has two different approaches for this additive decomposition based on either the solution of a Sylvester equation or on a block wise projection approach. This gives MORLAB the advantage of handling unstructured systems, while staying efficient and accurate due to only computing the necessary deflating subspaces. For both approaches, the matrix sign and disk functions are used, as quickly defined below.

Let $Y \in \mathbb{R}^{n \times n}$ be a matrix with no purely imaginary eigenvalues, then the Jordan canonical form of Y can be written as

$$Y = S \begin{bmatrix} J_- & 0 \\ 0 & J_+ \end{bmatrix} S^{-1}, \tag{2}$$

where S is an invertible transformation matrix, J_- contains the k eigenvalues of Y with negative real parts and J_+ the $n-k$ eigenvalues with positive real parts. The *matrix sign function* is then defined as

$$\operatorname{sign}(Y) = S \begin{bmatrix} -I_k & 0 \\ 0 & I_{n-k} \end{bmatrix} S^{-1}, \tag{3}$$

with S the transformation matrix from (2); see, e.g., [34]. Efficient computations can be based on a Newton scheme.

Let $\lambda X - Y$, with $X, Y \in \mathbb{R}^{n \times n}$, be a regular matrix pencil with no eigenvalues on the unit circle and its Weierstrass canonical form be written as

$$\lambda X - Y = W \begin{bmatrix} \lambda I_k - J_0 & 0 \\ 0 & \lambda N - J_\infty \end{bmatrix} T^{-1}, \tag{4}$$

where W, T are invertible transformation matrices, $\lambda I_k - J_0$ contains the k eigenvalues inside the unit disk and $\lambda N - J_\infty$ the $n-k$ eigenvalues outside the unit disk. The *right matrix pencil disk function* is then defined by

$$\operatorname{disk}(Y, X) = T \left(\lambda \begin{bmatrix} 0 & 0 \\ 0 & I_{n-k} \end{bmatrix} - \begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix} \right) T^{-1}, \tag{5}$$

with T, the right transformation matrix from (4). The computation follows the inverse-free iteration [1,5] and a subspace extraction method [7,36].

In the following subsections, the ideas of the additive decomposition for two general system classes are quickly summarized.

3.1 Standard system case

Assume a continuous-time standard system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{6}$$

with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times m}$, $D \in \mathbb{R}^{p \times m}$, A having no eigenvalues on the imaginary axis and its representation in the frequency domain by the corresponding transfer function

$$G(s) = C(sI_n - A)^{-1}B + D,$$

for $s \in \mathbb{C}$. Most model reduction methods can only be applied to asymptotically stable systems, which means in case of (6) that A has only eigenvalues with negative real parts. Nevertheless, model reduction methods can be applied by decomposing the system (6) into two subsystems, where the system matrices contain either the stable or anti-stable system part, i.e., we search for a transformation matrix T such that

$$T^{-1}AT = \begin{bmatrix} A_s & 0 \\ 0 & A_u \end{bmatrix},$$

where A_s contains only the stable and A_u the anti-stable eigenvalues. Using T as state-space transformation and partitioning accordingly the input and output matrices yields the additive system decomposition of the system's transfer function

$$G(s) = G_s(s) + G_u(s).$$

Applying the matrix sign function (3) to A gives the appropriate spectral splitting, where the spectral projectors onto the deflating subspaces are given as

$$\mathcal{P}_s = \frac{1}{2}(I_n - \operatorname{sign}(A)) \quad \text{and} \quad \mathcal{P}_u = \frac{1}{2}(I_n + \operatorname{sign}(A)).$$

Let $QR\Pi^{\mathsf{T}} = I_n - \operatorname{sign}(A)$ be a pivoted QR decomposition, the dimension of the deflating subspace corresponding to the eigenvalues with negative real part is given by $0.5(n + \operatorname{tr}(\operatorname{sign}(A)))$ and we get

$$Q^{\mathsf{T}}AQ = \begin{bmatrix} A_s & W_A \\ 0 & A_u \end{bmatrix}.$$

By solving the standard Sylvester equation

$$-A_uX + XA_s - W_A = 0, \tag{7}$$

the final transformation matrix and its inverse are given by

$$T = Q\begin{bmatrix} I_k & X \\ 0 & I_{n-k} \end{bmatrix} \quad \text{and} \quad T^{-1} = \begin{bmatrix} I_k & -X \\ 0 & I_{n-k} \end{bmatrix} Q^{\mathsf{T}}. \tag{8}$$

The MORLAB implementation uses the Newton iteration with Frobenius norm scaling for the computation of the matrix sign function as well as a matrix sign function-based solver for the Sylvester equation (7). Note that the actual transformation matrix (8) is never setup completely but only applied block wise on the original system to avoid unnecessary computations.

Remark 1 (Splitting of discrete-time standard systems)**.** In case of discrete-time standard systems, the implementation involves the matrix sign function of $(A + I_n)^{-1}(A - I_n)$ and the solution of the discrete-time Sylvester equation $A_u^{-1}XA_s - X - A_u^{-1}W_A = 0$ for doing the spectral splitting with respect to the unit circle.

3.2 Descriptor system case

Now, we consider the case of continuous-time descriptor systems

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{9}$$

with $E, A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times m}$, $D \in \mathbb{R}^{p \times m}$, $\lambda E - A$ having no finite eigenvalues on the imaginary axis and its representation in the frequency domain by the corresponding transfer function

$$G(s) = C(sE - A)^{-1}B + D, \quad s \in \mathbb{C}.$$

In contrast to the previous section, an additional splitting for the algebraic part corresponding to the infinite eigenvalues of $\lambda E - A$ is necessary, i.e., we search for transformation matrices W, T such that

$$W(\lambda E - A)T = \lambda \begin{bmatrix} E_s & 0 & 0 \\ 0 & E_u & 0 \\ 0 & 0 & E_\infty \end{bmatrix} - \begin{bmatrix} A_s & 0 & 0 \\ 0 & A_u & 0 \\ 0 & 0 & A_\infty \end{bmatrix}, \tag{10}$$

where $\lambda E_s - A_s$ contains the finite stable eigenvalues, $\lambda E_u - A_u$ the finite anti-stable eigenvalues and $\lambda E_\infty - A_\infty$ only infinite eigenvalues. Then, the system and its transfer function accordingly decouple into the different parts

$$G(s) = G_s(s) + G_u(s) + G_\infty(s),$$

as shown in Fig. 1. For this purpose, the Theorem 3 from [11] is used to construct block wise orthogonal transformation matrices.

First, the splitting of the algebraic part is performed as $G = G_{su} + G_\infty$ by using the matrix disk function. In fact, the inverse-free iteration is applied to the matrix pencil $\lambda(\alpha A) - E$ for appropriate scaling parameter α to compute matrices $\widetilde{A}$ and $\widetilde{E}$, whose null spaces are the deflating subspaces of $\lambda(\alpha A) - E$ corresponding to the eigenvalues inside and outside the unit circle, respectively; see [1,5]. Using a stabilized subspace extraction method [7,36], the orthogonal projection matrices can be obtained and according to [11] combined into appropriate transformation matrices to get

$$\widetilde{W}(\lambda E - A)\widetilde{T} = \lambda \begin{bmatrix} E_{su} & 0 \\ 0 & E_\infty \end{bmatrix} - \begin{bmatrix} A_{su} & 0 \\ 0 & A_\infty \end{bmatrix},$$

where $\lambda E_{su} - A_{su}$ contains all the finite eigenvalues. Afterwards, the generalized matrix sign function, working implicitly on the spectrum of $E_{su}^{-1} A_{su}$, is used such that the null spaces of $E_{su} - \operatorname{sign}(A_{su}, E_{su})$ and $E_{su} + \operatorname{sign}(A_{su}, E_{su})$ are the deflating subspaces corresponding to the eigenvalues left and right of the imaginary axis, respectively. Using the same subspace extraction method and block transformation, the block diagonalization (10) is accomplished.

Remark 2 (Splitting of discrete-time descriptor systems)**.** In the discrete-time descriptor case, the second splitting with respect to the imaginary axis needs to be replaced by a splitting with respect to the unit disk. Although, this is the actual nature of the matrix disk function, for performance reasons, the generalized matrix sign function is used as $\operatorname{sign}(A_{su} - E_{su}, A_{su} + E_{su})$ replaces the sign functions above.

4 Model reduction with the MORLAB toolbox

Most of the model reduction methods in MORLAB belong to the class of projection-based model reduction, i.e., we are searching for truncation matrices $W, T \in \mathbb{R}^{n \times r}$, which are used to project the state-space, $x \approx T\hat{x}$, and the corresponding equations. For example, given a continuous-time descriptor system (9), the reduced-order system is computed by

$$\begin{aligned} \underbrace{W^{\mathsf{T}} E T}_{\widehat{E}} \dot{\hat{x}}(t) &= \underbrace{W^{\mathsf{T}} A T}_{\widehat{A}} \hat{x}(t) + \underbrace{W^{\mathsf{T}} B}_{\widehat{B}} u(t), \\ \hat{y}(t) &= \underbrace{C T}_{\widehat{C}} \hat{x}(t) + \underbrace{D}_{\widehat{D}} u(t), \end{aligned} \tag{11}$$

with $\widehat{E}, \widehat{A} \in \mathbb{R}^{r \times r}$, $\widehat{B} \in \mathbb{R}^{r \times m}$, $\widehat{C} \in \mathbb{R}^{p \times r}$ and $\widehat{D} = D$. In the following, a very brief overview about the implemented model reduction methods in MORLAB is provided.

4.1 First-order methods

For the sake of generality in the MORLAB setting, only the method abbreviations are mentioned here. According to the naming scheme, see Section 2 and Fig. 2, the abbreviations have to be connected with the system classes to give the actual MORLAB function.

Table 5: First-order model reduction methods.

Method	Routine name	Comment	References
Balanced truncation	`bt`	- preserves stability	[26, 28]
Balanced stochastic truncation	`bst`	- preserves minimal phase	[9, 21]
Frequency-limited balanced truncation	`flbt`	- local frequency approx.	[19, 23]
Time-limited balanced truncation	`tlbt`	- local time approx.	[19, 22]
LQG balanced truncation	`lqgbt`	- unstable system reduction	[9, 24]
$\mathcal{H}_\infty$ balanced truncation	`hinfbt`	- unstable system reduction	[29]
Positive-real balanced truncation	`prbt`	- preserves passivity	[18, 33]
Bounded-real balanced truncation	`brbt`	- preserves contractivity	[31, 33]
Modal truncation	`mt`	- preserves spectrum parts	[8, 17]
Hankel-norm approximation	`hna`	- best approx. in Hankel-norm	[11, 20]

One of the oldest ideas for model reduction, and fitting with the spectral splitting approach from before, is modal truncation. While originally a part of the eigenvector basis was used for the projection [17], the deflating subspaces from Section 3 are an appropriate choice when using shifting and scaling on the spectrum of the system matrices.

A large part of the model reduction methods in MORLAB are so-called balancing-related methods. In classical balanced truncation [28], the continuous-time Lyapunov equations

$$\begin{aligned} AP + PA^\mathsf{T} + BB^\mathsf{T} &= 0, \\ A^\mathsf{T}Q + QA + C^\mathsf{T}C &= 0, \end{aligned} \tag{12}$$

are solved for the system Gramians P and Q, which are then used by, e.g., the square root or balancing-free square root method to compute the reduced-order projection matrices; see, e.g., [26, 38]. The balancing-related methods are based on the idea of balanced truncation but replace the Lyapunov equations (12) by other matrix equations, which infuse different properties to the resulting methods. Some comments on the implementation of balancing-related methods in MORLAB are given for previous versions in [10] for the standard system case and the general idea of the implementation of model reduction for descriptor systems is given in [12].

Also, the Hankel-norm approximation is implemented. This method is non-projection-based, i.e., by construction, there are no W, T fulfilling (11) and also $\widehat{D} = D$ does not hold anymore. This method solves the optimal approximation problem in the Hankel semi-norm and is also a good guess for the $\mathcal{H}_\infty$ approximation problem [11, 20]. It can be seen as a refinement of the balanced truncation method, since it is also based on the solution of (12).

As an overview for the current MORLAB version, Table 5 shows all the implemented model reduction methods for first-order continuous-time systems, with their routine abbreviation, a comment on their properties and references for the standard and descriptor versions.

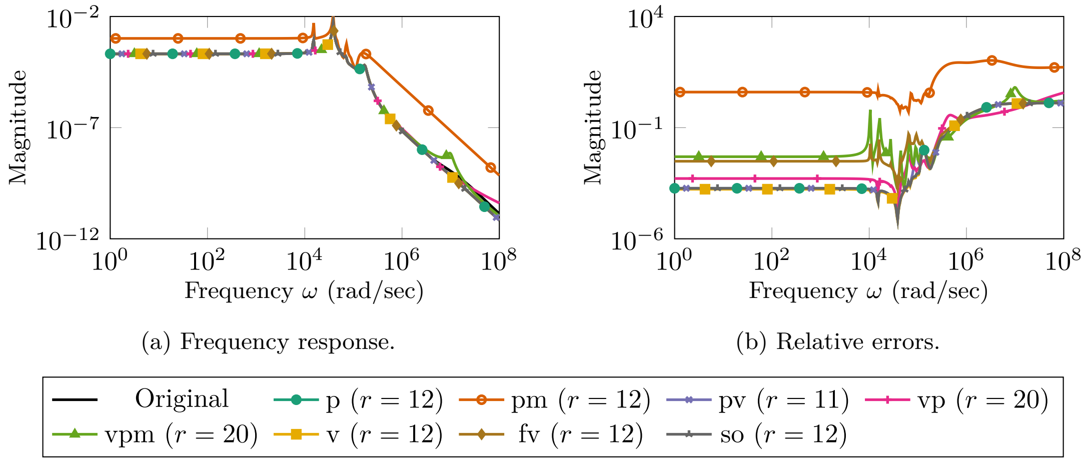
Remark 3 (Discrete-time model reduction methods)**.** Currently, only the methods `mt`, `bt` and `lqgbt` have discrete-time implementations for the standard and descriptor system case. Discrete-time equivalents of the continuous-time matrix equations are solved for those methods.

4.2 Second-order methods

In case of systems with second-order time derivatives, the toolbox implements different structure-preserving approaches. Given the system structure from Table 4, the reduced-order models will also have the form

$$\begin{aligned} \widehat{M}\ddot{\hat{x}}(t) &= -\widehat{K}\hat{x}(t) - \widehat{E}\dot{\hat{x}}(t) + \widehat{B}_u u(t), \\ \hat{y}(t) &= \widehat{C}_p\hat{x}(t) + \widehat{C}_v\dot{\hat{x}}(t) + \widehat{D}u(t), \end{aligned} \tag{13}$$

with $\widehat{M}\,\widehat{E}, \widehat{K} \in \mathbb{R}^{r \times r}$, $\widehat{B}_u \in \mathbb{R}^{r \times m}$, $\widehat{C}_p, \widehat{C}_v \in \mathbb{R}^{p \times r}$ and $\widehat{D} \in \mathbb{R}^{p \times m}$. MORLAB implements the second-order balanced truncation and balancing-related methods for this purpose. Originating



(a) Frequency response.

(b) Relative errors.

Figure 4: Frequency domain results for the butterfly gyroscope.

in [16, 27, 32], the second-order balanced truncation approach uses a first-order realization of the original second-order system and then restricts to parts of the system Gramians to result in (13). In [14], a collection of the different construction formulas can be found that are all implemented in MORLAB, as well as the frequency- and time-limited second-order balanced truncation methods, which are also implemented in MORLAB. The naming of the methods follows the previous subsection.

5 Numerical examples

In the following, two benchmark examples are shown to demonstrate possible applications of the MORLAB toolbox. The experiments reported here have been executed on a machine with 2 Intel(R) Xeon(R) Silver 4110 CPU processors running at 2.10GHz and equipped with 192 GB total main memory. The computer is running on CentOS Linux release 7.5.1804 (Core) and using MATLAB 9.4.0.813654 (R2018a) and the MORLAB toolbox version 5.0 [13].

The source code of the implementations used to compute the presented results can be obtained from

`https://doi.org/10.5281/zenodo.3678213`

and is authored by Jens Saak and Steffen W. R. Werner.

5.1 Butterfly gyroscope

As a first numerical example, we consider the butterfly gyroscope benchmark example from [30]; see [15] for the background. We will use the MORLAB toolbox as backend software for a two-step model reduction approach. Thereby, a fast pre-reduction step is used to create an accurate, medium-scale approximation of the original model and, afterwards, more sophisticated model reduction methods are used to construct the final reduced-order model, see, e.g., [25, 35]. The model we consider now involves second-order time derivatives as it has the form

$$\begin{aligned} M\ddot{x}(t) + E\dot{x}(t) + Kx(t) &= B_u u(t), \\ y(t) &= C_p x(t), \end{aligned}$$

with a state-space dimension $n = 17\,361$ and $m = 1$, $p = 12$ inputs and outputs, respectively.

As in [35], we use the structure-preserving interpolation framework from [4] as efficient pre-reduction method that preserves the system structure in the intermediate medium-scale approximation. We compute a single projection basis the same way as in [35] using the sampling points

$\pm$`logspace(0, 8, 100)`i. After orthogonalization by the economy size QR decomposition, the intermediate reduced-order model has the state-space dimension 2 600. Now, we apply the second-order balanced truncation methods from MORLAB to the intermediate model. The toolbox supports an all-at-once approach for balancing-related model reduction, i.e., the underlying Gramians are computed once and then used for several different reduced-order models. Therefore, we can compute all 8 different second-order balancing formulas from [14] at the same time and compare them afterwards.

Fig. 4 shows the resulting reduced-order models in the frequency domain. The relative error was computed

$$\frac{\|G(i\omega) - \widehat{G}(i\omega)\|_2}{\|G(i\omega)\|_2},$$

in the frequency range $\omega \in [10^0, 10^8]$. Both plots were directly generated with the MORLAB routine `ml_sigmaplot`, which computes sigma and error plots for an arbitrary number of given models. The notation in the legend follows the formulas from [14]. Except for the pm and vpm models, all other reduced-order models are stable. Clearly, the winners are p, v, pv and so, which all have basically the same size and error behavior.

5.2 Parametric thermal block model

As second example, we consider the parametric thermal block model as described in [37] with the single parameter setup. Following this description, we consider the first-order generalized state-space system

$$\begin{aligned} E\dot{x}(t;\mu) &= A(\mu)x(t;\mu) + Bu(t), \\ y(t;\mu) &= Cx(t;\mu), \end{aligned} \tag{14}$$

where $A(\mu) = A_0 + \mu\,(0.2A_1 + 0.4A_2 + 0.6A_3 + 0.8A_4)$, with the parameter $\mu \in [10^{-6}, 10^2]$, the state-space dimension $n = 7\,488$ and $m = 1$, $p = 4$ inputs and outputs, respectively. The matrix pencil $\lambda E - A(\mu)$ is finite and stable for all parameter values μ in the range of interest.

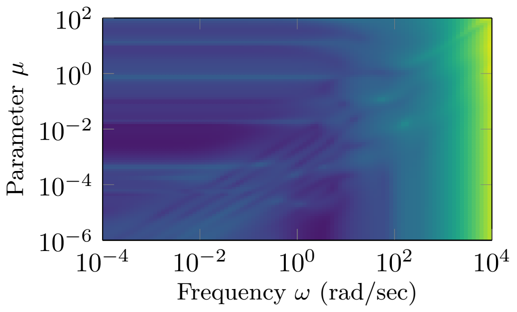
Although MORLAB does not implement parametric system classes yet, we want to use the toolbox as model reduction backend for two-step parametric model reduction methods. The first idea is taken from [3]. Given some non-parametric reduced-order models G_j computed for parameter samples μ_j, $j = 1, \ldots, k$, a global parameter interpolating system can be constructed in the frequency domain using Lagrange interpolation as

$$\widehat{G}(s,\mu) = \sum_{j=1}^{k} \ell_j(\mu) G_j(s), \tag{15}$$

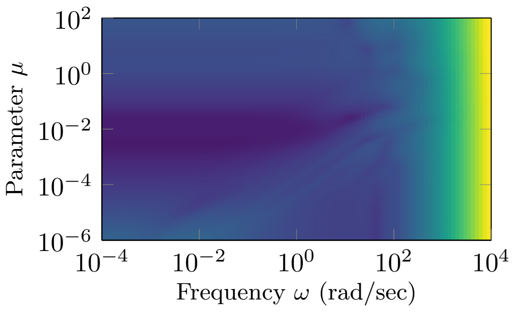
with $\ell_j(\mu)$ Lagrange basis functions in the parameter μ with the knot vector $\mu_1, \ldots, \mu_k$. Rewriting the sum (15) gives a realization for the interpolating reduced-order model

$$\widehat{E} = \begin{bmatrix} \widehat{E}_1 & & \\ & \ddots & \\ & & \widehat{E}_k \end{bmatrix}, \quad \widehat{A} = \begin{bmatrix} \widehat{A}_1 & & \\ & \ddots & \\ & & \widehat{A}_k \end{bmatrix},$$
$$\widehat{B} = \begin{bmatrix} \widehat{B}_1 \\ \vdots \\ \widehat{B}_k \end{bmatrix}, \qquad \widehat{C} = \begin{bmatrix} \ell_1(\mu)\widehat{C}_1, & \ldots, & \ell_k(\mu)\widehat{C}_k \end{bmatrix},$$

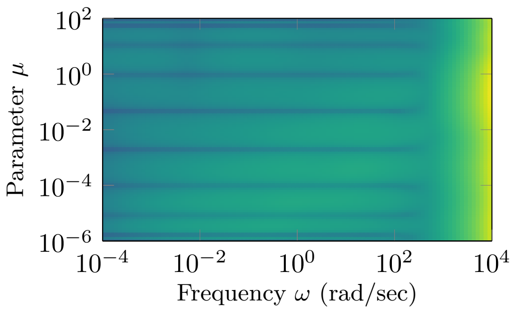
where $\widehat{E}_j, \widehat{A}_j, \widehat{B}_j, \widehat{C}_j$ are the matrices of the local reduced-order models. Thinking of other scalar function approximation methods, easy extensions of (15) come into mind. Replacing the Lagrange basis functions $\ell_j(\mu)$ by linear B-splines $b_{1,j}(\mu)$ over the knot vector $\mu_1, \ldots, \mu_k$, we can construct a piecewise linear interpolating reduced-order model. Another idea would be to use the variation diminishing B-spline approximation, which just needs some modifications of the knot vector used



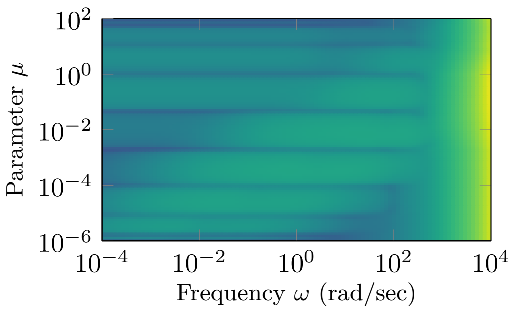
(a) TwoPW ($r = 63$).



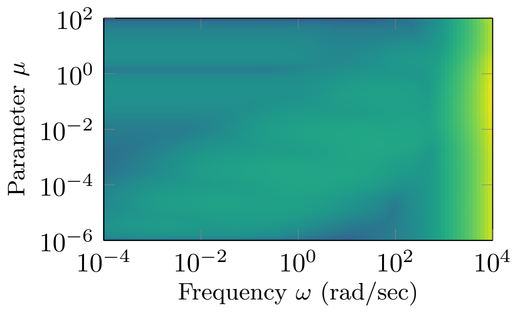
(b) OnePW ($r = 107$).



(c) InterpLag ($r = 108$).



(d) InterpBspline ($r = 108$).



(e) VarDABspline ($r = 108$).

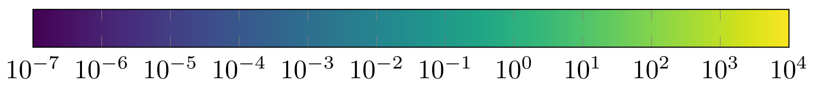
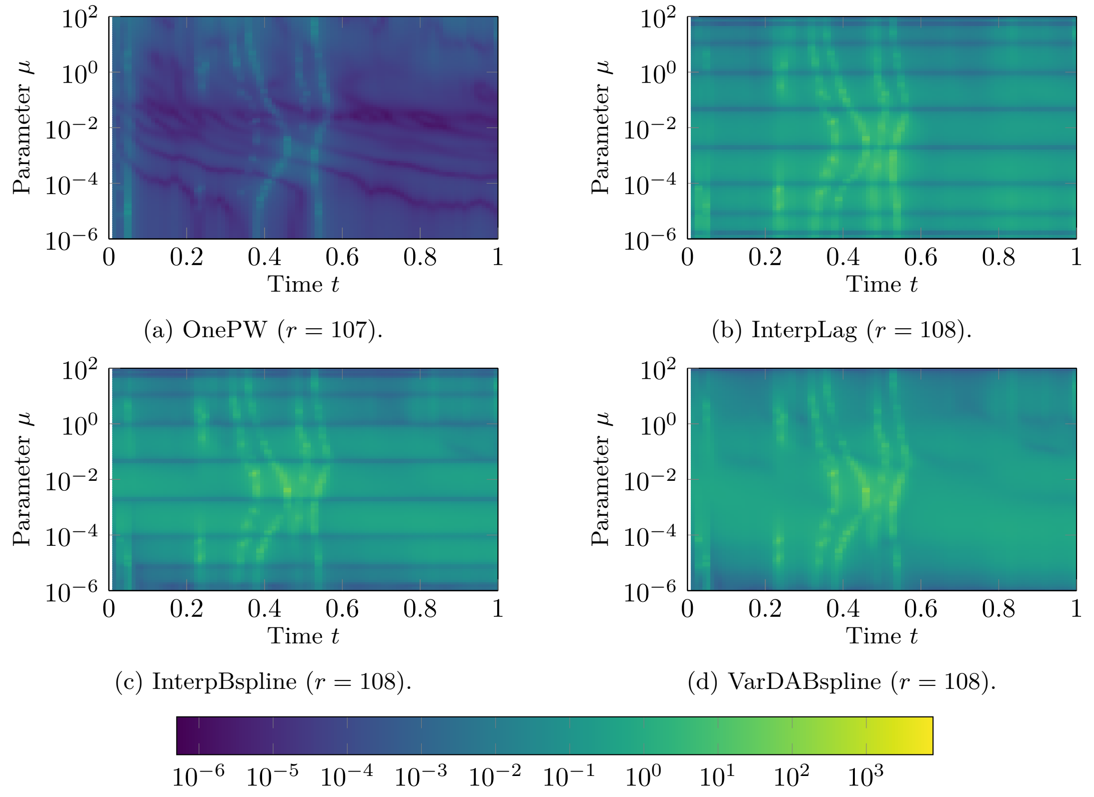


Figure 5: Relative errors in the frequency domain of different parametric extensions for the thermal block model.

for the basis functions. In general, this transfer function interpolation-based approach comes with several advantages. First, it does not matter how the local reduced-order models were computed or which size they have. If all local reduced-order models were stable, the global interpolating one will be stable by construction, too. Also, instead of setting up the complete reduced-order model, it can be advantageous to use the local reduced-order models for simulations in parallel and combine the results at the end by the parametric output matrix.

A different approach is given by the piecewise approximation; see, e.g., [2]. For this method, let the local reduced-order models be computed by projection methods and the projection matrices be collected as $W = [W_1, \ldots, W_k]$ and $T = [T_1, \ldots, T_k]$. The parametric reduced-order system is then computed using W, T as projection matrices on the original system, as in (11). Concerning



(a) OnePW ($r = 107$).

(b) InterpLag ($r = 108$).

(c) InterpBspline ($r = 108$).

(d) VarDABspline ($r = 108$).

Figure 6: Relative errors in the time simulation of different parametric extensions for the thermal block model.

the parametric matrix $A(\mu)$ in (14), we note that

$$\begin{aligned} W^{\mathsf{T}}A(\mu)T &= W^{\mathsf{T}}A_0T + \mu\left(0.2W^{\mathsf{T}}A_1T + 0.4W^{\mathsf{T}}A_2T + 0.6W^{\mathsf{T}}A_3T + 0.8W^{\mathsf{T}}A_4T\right) \\ &= \widehat{A}_0 + \mu\left(0.2\widehat{A}_1 + 0.4\widehat{A}_2 + 0.6\widehat{A}_3 + 0.8\widehat{A}_4\right). \end{aligned}$$

Using this method, we can preserve the exact parameter dependency in the reduced-order model. Variants of it, for example, use column compression of T and W to control the size of the resulting reduced-order model. Also, it needs to be noted that by concatenation of the projection matrices, original properties like stability preservation can be lost. Therefore, modifications like a one-sided projection by combining $[W, T]$ into a single basis can be used to handle most systems.

For our numerical example, we will use the following setup. For the parameter sampling points, we use 10 logarithmically distributed Chebyshev roots, i.e., let $\nu_1, \ldots, \nu_k$ be the Chebyshev roots in the interval $[-6, 2]$, the sampling points are given as $\mu_j = 10^{\nu_j}$. The local reduced-order models are computed by the balanced truncation routine from MORLAB (`ml_ct_dss_bt`) using 10^{-4} for the absolute error bound and we save the reduced-order models as well as the projection matrices for the parametric approaches. The following different parametric reduced-order models are then computed:

- two-sided piecewise approximation (TwoPW), where the final truncated projection matrices were compressed using singular value decompositions and a relative truncation tolerance of 10^{-4},
- one-sided piecewise approximation (OnePW), where the final truncated projection matrix was compressed using the basis concatenation and the singular value decomposition with relative truncation tolerance 10^{-4},
- transfer function interpolation using Lagrange basis functions (InterpLag),

- transfer function interpolation using linear B-splines (InterpBspline),
- transfer function approximation using the variation diminishing approximation with quadratic B-spline basis functions (VarDABspline).

Fig. 5 shows the results in the frequency domain, where we computed the point wise relative errors as

$$\frac{\|G(i\omega,\mu) - \widehat{G}(i\omega,\mu)\|_2}{\|G(i\omega,\mu)\|_2},$$

in the ranges $\omega \in [10^{-4}, 10^4]$ and $\mu \in [10^{-6}, 10^2]$. The piecewise methods, TwoPW and OnePW, are the clear winners of the comparison. We note that TwoPW is unstable for all parameters, while OnePW is stable. Also, the interpolation approaches work nicely, where the interpolation property is clearly visible in the plots. The variation diminishing B-spline result, VarDABspline, seems to be a smoother version of InterpBspline.

In the time domain, we simulate the parametric systems with using a pre-sampled white noise input signal. The relative errors shown in Fig. 6 are computed by

$$\sqrt{\sum_{j=1}^{4} \frac{|y_j(t;\mu) - \hat{y}_j(t;\mu)|^2}{|y_j(t;\mu)|^2}}$$

in the ranges $t \in [0, 1]$ and $\mu \in [10^{-6}, 10^2]$. The TwoPW is not shown in Fig. 6, since due to the instability in all parameters, no useful results were computed during the simulation. For the rest, we see that again OnePW performs overall very good. Also we see that the B-spline approaches and classical Lagrange interpolation give more or less the same results.

6 Conclusions

We presented the MORLAB toolbox as efficient software solution for model reduction of dense, medium-scale linear time-invariant systems. We gave an overview about the main features and structure of the toolbox, as well as underlying programming principles. An important point when considering unstructured systems is the spectral splitting, which we showed in MORLAB to be based on spectral projection methods. Following the computational steps led to an overview about the implemented model reduction methods in MORLAB. We gave two numerical examples to illustrate how MORLAB can be used as backend software for different system types. In the first example, MORLAB provided the efficient, structure-preserving implementation of sophisticated model reduction methods that are used in two-step approaches. In the second example, we used MORLAB to generate local reduced-order models that were afterwards combined by different techniques to construct parametric reduced-order systems.

Acknowledgment

This work was supported by the German Research Foundation (DFG) Research Training Group 2297 “MathCoRe”, Magdeburg.

References

[1] Z. Bai, J. Demmel, and M. Gu. An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems. *Numer. Math.*, 76(3):279–308, 1997. doi:10.1007/s002110050264.

[2] U. Baur, C. A. Beattie, P. Benner, and S. Gugercin. Interpolatory projection methods for parameterized model reduction. *SIAM J. Sci. Comput.*, 33(5):2489–2518, 2011. doi:10.1137/090776925.

[3] U. Baur and P. Benner. Modellreduktion für parametrisierte Systeme durch balanciertes Abschneiden und Interpolation (Model Reduction for Parametric Systems Using Balanced Truncation and Interpolation). *at-Automatisierungstechnik*, 57(8):411–420, 2009. doi:10.1524/auto.2009.0787.

[4] C. A. Beattie and S. Gugercin. Interpolatory projection methods for structure-preserving model reduction. *Syst. Control Lett.*, 58(3):225–232, 2009. doi:10.1016/j.sysconle.2008.10.016.

[5] P. Benner. *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*. Logos–Verlag, Berlin, Germany, 1997. *Also:* Dissertation, Fakultät für Mathematik, TU Chemnitz–Zwickau, 1997.

[6] P. Benner. A MATLAB repository for model reduction based on spectral projection. In *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, pages 19–24, October 2006. doi:10.1109/CACSD-CCA-ISIC.2006.4776618.

[7] P. Benner. Partial stabilization of descriptor systems using spectral projectors. In P. Van Dooren, S. P. Bhattacharyya, R. H. Chan, V. Olshevsky, and A. Routray, editors, *Numerical Linear Algebra in Signals, Systems and Control*, volume 80 of *Lect. Notes Electr. Eng.*, pages 55–76. Springer Netherlands, 2011. doi:10.1007/978-94-007-0602-6_3.

[8] P. Benner and E. S. Quintana-Ortí. Model reduction based on spectral projection methods. In P. Benner, V. Mehrmann, and D. Sorensen, editors, *Dimension Reduction of Large-Scale Systems*, volume 45, pages 5–45, Berlin/Heidelberg, Germany, 2005. Springer. doi:10.1007/3-540-27909-1_1.

[9] P. Benner and T. Stykel. Model order reduction for differential-algebraic equations: A survey. In Achim Ilchmann and Timo Reis, editors, *Surveys in Differential-Algebraic Equations IV*, Differential-Algebraic Equations Forum, pages 107–160. Springer International Publishing, Cham, March 2017. doi:10.1007/978-3-319-46618-7_3.

[10] P. Benner and S. W. R. Werner. Balancing related model reduction with the MORLAB toolbox. *Proc. Appl. Math. Mech.*, 18(1):e201800083, 2018. doi:10.1002/pamm.201800083.

[11] P. Benner and S. W. R. Werner. Hankel-norm approximation of large-scale descriptor systems. e-print 1612.06205, arXiv, 2018. To appear in Adv. Comput. Math. URL: http://arxiv.org/abs/1612.06205.

[12] P. Benner and S. W. R. Werner. Model reduction of descriptor systems with the MORLAB toolbox. *IFAC-PapersOnLine 9th Vienna International Conference on Mathematical Modelling MATHMOD 2018, Vienna, Austria, 21–23 February 2018*, 51(2):547–552, 2018. doi:10.1016/j.ifacol.2018.03.092.

[13] P. Benner and S. W. R. Werner. MORLAB – Model Order Reduction LABoratory (version 5.0), 2019. see also: http://www.mpi-magdeburg.mpg.de/projects/morlab. doi:10.5281/zenodo.3332716.

[14] P. Benner and S. W. R. Werner. Frequency- and time-limited balanced truncation for large-scale second-order systems. e-print 2001.06185, arXiv, 2020. math.OC. URL: http://arxiv.org/abs/2001.06185.

[15] D. Billger. The butterfly gyro. In P. Benner, V. Mehrmann, and D. C. Sorensen, editors, *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*, pages 349–352. Springer-Verlag, Berlin/Heidelberg, Germany, 2005. doi:10.1007/3-540-27909-1_18.

[16] Y. Chahlaoui, D. Lemonnier, A. Vandendorpe, and P. Van Dooren. Second-order balanced truncation. *Linear Algebra Appl.*, 415(2–3):373–384, 2006. doi:10.1016/j.laa.2004.03.032.

[17] E. J. Davison. A method for simplifying linear dynamic systems. *IEEE Trans. Autom. Control*, AC–11:93–101, 1966. doi:10.1109/TAC.1966.1098264.

[18] U. B. Desai and D. Pal. A transformation approach to stochastic model reduction. *IEEE Trans. Autom. Control*, 29(12):1097–1100, 1984. doi:10.1109/TAC.1984.1103438.

[19] W. Gawronski and J.-N. Juang. Model reduction in limited time and frequency intervals. *Int. J. Syst. Sci.*, 21(2):349–376, 1990. doi:10.1080/00207729008910366.

[20] K. Glover. All optimal Hankel-norm approximations of linear multivariable systems and their L^{∞}-error norms. *Internat. J. Control*, 39(6):1115–1193, 1984. doi:10.1080/00207178408933239.

[21] M. Green. A relative error bound for balanced stochastic truncation. *IEEE Trans. Autom. Control*, 33(10):961–965, 1988. doi:10.1109/9.7255.

[22] K. Haider, A. Ghafoor, M. Imran, and F. M. Malik. Model reduction of large scale descriptor systems using time limited Gramians. *Asian J. Control*, 19(3):1217–1227, 2017. doi:10.1002/asjc.1444.

[23] M. Imran and A. Ghafoor. Model reduction of descriptor systems using frequency limited Gramians. *J. Franklin Inst.*, 352(1):33–51, 2015. doi:10.1016/j.jfranklin.2014.10.013.

[24] E. A. Jonckheere and L. M. Silverman. A new set of invariants for linear systems – application to reduced order compensator design. *IEEE Trans. Autom. Control*, 28(10):953–964, 1983. doi:10.1109/TAC.1983.1103159.

[25] M. Lehner and P. Eberhard. A two-step approach for model reduction in flexible multibody dynamics. *Multibody Syst. Dyn.*, 17(2-3):157–176, 2007. doi:10.1007/s11044-007-9039-5.

[26] V. Mehrmann and T. Stykel. Balanced truncation model reduction for large-scale systems in descriptor form. In P. Benner, V. Mehrmann, and D. C. Sorensen, editors, *Dimension Reduction of Large-Scale Systems*, volume 45, pages 83–115. Springer-Verlag, Berlin/Heidelberg, Germany, 2005. doi:10.1007/3-540-27909-1_3.

[27] D. G. Meyer and S. Srinivasan. Balancing and model reduction for second-order form linear systems. *IEEE Trans. Autom. Control*, 41(11):1632–1644, 1996. doi:10.1109/9.544000.

[28] B. C. Moore. Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Trans. Autom. Control*, AC–26(1):17–32, 1981. doi:10.1109/TAC.1981.1102568.

[29] D. Mustafa and K. Glover. Controller reduction by $\mathcal{H}_\infty$-balanced truncation. *IEEE Trans. Autom. Control*, 36(6):668–682, 1991. doi:10.1109/9.86941.

[30] Oberwolfach Benchmark Collection. Butterfly gyroscope. hosted at MORwiki – Model Order Reduction Wiki, 2004. URL: http://modelreduction.org/index.php/Butterfly_Gyroscope.

[31] P. C. Opdenacker and E. A. Jonckheere. A contraction mapping preserving balanced reduction scheme and its infinity norm error bounds. *IEEE Trans. Circuits Syst.*, 35(2):184–189, 1988. doi:10.1109/31.1720.

[32] T. Reis and T. Stykel. Balanced truncation model reduction of second-order systems. *Math. Comput. Model. Dyn. Syst.*, 14(5):391–406, 2008. doi:10.1080/13873950701844170.

[33] T. Reis and T. Stykel. Positive real and bounded real balancing for model reduction of descriptor systems. *Internat. J. Control*, 83(1):74–88, 2010. doi:10.1080/00207170903100214.

[34] J. D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Internat. J. Control*, 32(4):677–687, 1980. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971). doi:10.1080/00207178008922881.

[35] J. Saak, D. Siebelts, and S. W. R. Werner. A comparison of second-order model order reduction methods for an artificial fishtail. *at-Automatisierungstechnik*, 67(8):648–667, 2019. `doi:10.1515/auto-2019-0027`.

[36] X. Sun and E. S. Quintana-Ortí. Spectral division methods for block generalized Schur decompositions. *Mathematics of Computation*, 73(248):1827–1847, 2004. `doi:10.1090/S0025-5718-04-01667-9`.

[37] The MORwiki Community. Thermal block. hosted at MORwiki – Model Order Reduction Wiki, 2020. URL: `https://morwiki.mpi-magdeburg.mpg.de/morwiki/index.php/Thermal_Block`.

[38] A. Varga. Efficient minimal realization procedure based on balancing. In *Prepr. of the IMACS Symp. on Modelling and Control of Technological Systems*, volume 2, pages 42–47, 1991.