MORLAB – A Model Order Reduction Framework in MATLAB and Octave

Peter Benner^{1,2} and Steffen W. R. Werner¹ ( )

¹ Max Planck Institute for Dynamics of Complex Technical Systems,
Sandtorstr. 1, 39106 Magdeburg, Germany
{benner,werner}@mpi-magdeburg.mpg.de

² Faculty of Mathematics, Otto von Guericke University,
Universitätsplatz 2, 39106 Magdeburg, Germany
peter.benner@ovgu.de

Abstract. When synthesizing feedback controllers for large-scale dynamical systems, often a reduction of the plant model by model order reduction is required. This is a typical task in computer-aided control system design environments. Therefore, in the last years, model order reduction became an essential tool for the practical use of mathematical models in engineering processes. For the integration of established model order reduction methods into those processes, software solutions are needed. In this paper, we describe the MORLAB (Model Order Reduction LABoratory) toolbox as such a software solution in MathWorks MATLAB[®] and GNU Octave, and its featured integration into established software tools used in simulations and controller design. We give benchmark examples for two important extensions of the toolbox.

Keywords: Model order reduction · Dynamical systems · MATLAB · Octave

1 Introduction

Dynamical input-output systems are a usual way of modeling natural phenomena as, e.g., fluid dynamics, mechanical systems or the behavior of electrical circuits. In general, they are described by differential and algebraic equations

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

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

© Springer Nature Switzerland AG 2020

A. M. Bigatti et al. (Eds.): ICMS 2020, LNCS 12097, pp. 432–441, 2020.

https://doi.org/10.1007/978-3-030-52200-1_43

where the inputs, $u(t) \in \mathbb{R}^m$, are used to influence the internal states, $x(t) \in \mathbb{R}^n$, to get the desired outputs, $y(t) \in \mathbb{R}^p$. Nowadays, dynamical systems (1) are an essential tool in simulation and the design of controllers. But due to the use of highly accurate models, involving a large number of differential equations in (1), the demand for computational resources, e.g., time and memory, is often too high for a practical usage. The aim of model order reduction is the construction of a surrogate model $\hat{G}$, with a much smaller number of internal states, $\hat{x}(t) \in \mathbb{R}^r$, and differential equations, $r \ll n$, which approximates the input-to-output behavior of (1), i.e.,

$$\|y - \hat{y}\| \leq \text{tolerance} \cdot \|u\|,$$

for an appropriately defined norm and all admissible inputs u , where $\hat{y}$ denotes the outputs of the reduced-order model.

For the integration into established engineering processes, software solutions for the model reduction problem are needed. One such software solution, compatible with MathWorks MATLAB® and GNU Octave, is the **MORLAB (Model Order Reduction LABoratory)** toolbox [6, 8]. As a free and open source software, the main aim of the toolbox is the model order reduction of linear, medium-scale dynamical systems.

In Sect. 2, we will briefly describe the fundamentals of the MORLAB toolbox and afterwards, in Sect. 3, provide the ideas behind the integration of MORLAB in other MATLAB and Octave software, also presenting some numerical examples.

2 The MORLAB Toolbox

The MORLAB toolbox originated in [2] as a repository of MATLAB codes for model order reduction of linear standard systems. The current version [6] comes with 10 different model reduction techniques for continuous- and discrete-time standard, descriptor and second-order systems. The implementation of the toolbox is based on spectral projection methods, like the matrix sign function [11] and the right matrix pencil disk function [3], which are used for the solution of underlying matrix equations. Therefore, the toolbox comes with many different solvers for different types of matrix equations, e.g., for continuous- and discrete-time algebraic Riccati or Lyapunov equations. To support the work with linear dynamical systems, the toolbox also implements a number of system-theoretic subroutines, e.g., the additive decomposition of dynamical systems, and evaluation tools for the time and frequency domain with support for the different implemented system structures.

An overview of the basic principles of the MORLAB toolbox can be found in [8] and is also given in the following:

Open source and free	The toolbox is licensed under the GNU Affero General Public License v3.0 and is freely available on the project website and Zenodo.
Fast and accurate	Based on the spectral projection methods, the toolbox can outperform other available model reduction and system-theoretic software.
Unified framework	All model reduction routines share the same interface and allow for quick exchange and easy comparison between the methods.
Configurable	All routines can be configured separately using option structs.
Modular	Each subroutine can be called on its own by the user to be used and combined in various ways.
Portable	No binary extensions are required, which allows for running the toolbox with bare MATLAB or Octave installations.
Documentation	The toolbox comes with an extensive documentation for every function in HTML and MATLAB inline format.
Dependencies	MATLAB ($\geq 2012b$), Octave ($\geq 4.0.0$).

See [8] for a more detailed discussion of the toolbox structure, implementation details and overviews about the currently supported system classes.

3 Toolbox Integration in MATLAB and Octave

An important point for the design decisions in MORLAB was the compatibility with other highly-used system-theoretic software tools. Two of the most important ones are Simulink® and the Control System Toolbox™ in MATLAB. In the following, we describe first the integration of MORLAB into those toolboxes using the state-space object and give then an overview of the extensions that MORLAB introduces itself.

The numerical experiments, used here for illustration, have been executed on a machine with 2 Intel(R) Xeon(R) Silver 4110 CPU processors running at 2.10 GHz and equipped with 192 GB total main memory. The computer runs on CentOS Linux release 7.5.1804 (Core) with MATLAB 9.7.0.1190202 (R2019b).

3.1 Standard First-Order Systems and the State-Space Object

The system class corresponding to state-space models as (1), which is fully supported in MATLAB, are standard linear time-invariant systems, e.g., in the continuous-time case

Fig. 1. Idea of integrating MORLAB into Simulink and the Control System Toolbox in MATLAB.

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t),\end{aligned}\tag{2}$$

with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$ and $D \in \mathbb{R}^{p \times m}$. The way in MATLAB to efficiently deal with (2) are so-called *state-space model objects* (**ss** objects). Those objects are, in principle, a collection of the most important properties of (2), e.g., the matrices A, B, C, D , corresponding measured data, units of the state variables and so forth.

The Control System Toolbox is the main toolbox for dealing with dynamical systems in MATLAB. It provides a lot of important system-theoretic routines for the analysis and simulation of (2). The toolbox uses the **ss** object as its main interface. On the other hand, Simulink is a modeling toolkit for hierarchical system design and numerical experiments using graphical blocks. Complete dynamical processes are usually modeled here, especially embedding **LTI system** blocks, which correspond to (2) and use the **ss** object in its backend.

Now, the task is the integration of model order reduction into the computer-aided control system design workflow. Concerning the model order reduction of (2), the reduced-order surrogate model reads as

$$\begin{aligned}\hat{x}(t) &= \hat{A}\hat{x}(t) + \hat{B}u(t), \\ \hat{y}(t) &= \hat{C}\hat{x}(t) + \hat{D}u(t),\end{aligned}$$

with $\hat{A} \in \mathbb{R}^{r \times r}$, $\hat{B} \in \mathbb{R}^{r \times m}$, $\hat{C} \in \mathbb{R}^{p \times r}$ and $\hat{D} \in \mathbb{R}^{p \times m}$, which belongs to the same system class as (2). Therefore, MORLAB provides consistent interfaces for the model order reduction routines, i.e., the reduced-order models will be of the same data type as the original models. Besides an interface for the single matrices

of (2) and one using `structs`, MORLAB fully supports `ss` objects. An example of the resulting workflow is given in Fig. 1. Let `sys` be the original `ss` object, e.g., in a Simulink LTI `system` block. This object can be directly put into a MORLAB routine (here `ml_ct_ss_bt`), which produces a new `ss` object, namely `rom`. Due to the consistent data type of `rom`, the reduced-order model can now directly replace the original `sys` object for the acceleration of the surrounding process. The important point here is that the model reduction with MORLAB only concerns the `ss` object to be exchanged but the surrounding process is untouched. The example function used in Fig. 1 can be replaced by any model order reduction routine of choice from MORLAB. Overviews about supported methods for (2) can be found in [4, 8].

Remark 1. Parts of the Control System Toolbox are also available in Octave implemented in the ‘control’ package. There, the same interfaces and integration options are provided as for the MATLAB software. Due to the lack of system-theoretic subroutines in Octave, MORLAB additionally provides analysis and simulation tools that are compatible with MATLAB and Octave, which highly extend the functionality of Octave for dynamical systems in its current state.

Remark 2. Besides the continuous-time version (2), the `ss` object supports discrete-time linear time-invariant standard systems, which are also implemented in the MORLAB toolbox with appropriate model reduction routines.

It should be noted that also the default MATLAB toolboxes provide some model reduction routines for small-scale dense standard systems in the Control System Toolbox and the Robust Control ToolboxTM. In comparison, MORLAB can be applied to way larger systems due to its efficient underlying spectral projection routines, which also often allow MORLAB to outperform the routines from the default MATLAB toolboxes by far. Also, the variety of the MORLAB model order reduction routines is so far not known to be matched by any other comparable model order reduction software for any of the provided system classes. Especially, the following two sections describe significant extensions of the MORLAB toolbox, which, in large parts, cannot be found in any other published model reduction software.

3.2 Extension for Descriptor Systems

The first extension of the standard MATLAB functionality for linear dynamical systems that was done in MORLAB are descriptor systems, e.g., in the continuous-time case

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{3}$$

with $E \in \mathbb{R}^{n \times n}$ and A, B, C, D as in (2). In principle, the `ss` object supports general E matrices but most of the routines in the system-theoretic toolboxes reformulate (3) as (2) by explicitly inverting the E matrix to the right. Beside

Fig. 2. Frequency response results for the PI-circuit example.

explicit inversion often has a negative influence on numerical accuracy, in case of differential-algebraic equations describing the system dynamics, the E matrix might not be invertible anymore. In that case, a lot of the routines in the Control System Toolbox and Simulink are unusable. But MORLAB supports this case in corresponding model reduction and system-theoretic routines by proper handling of possible algebraic system parts. The idea of model order reduction of descriptor systems with MORLAB can be found in [5] and the underlying additive decomposition of descriptor systems also in [8].

In many cases, the use of the MORLAB toolbox makes the integration of descriptor system models into other MATLAB software possible as we will see in the following example. We consider the model of an RLC circuit described by cascaded PI-circuits from [9]. The chosen `S40PI.n` benchmark system has the form (3) with $n = 2182$, $m = 1$ and $p = 1$ and we apply an α -shift with $\alpha = 10^{-4}$ as in [10] to stabilize the system. By construction, the E matrix is not invertible and the overall system has a complicated nested structure of differential-algebraic equations. For this system, we are going to use the balanced truncation method for descriptor systems as it is implemented in MORLAB by

$$[\text{rom}, \text{info}] = \text{ml_ct_dss_bt}(\text{sys}, \text{opts}),$$

where `sys` contains the original model as `ss` object, `opts` is an option struct that could be used to adjust parameters of the subroutines, `rom` is again the reduced-order model and `info` contains information about the subroutines and the full- and reduced-order systems.

First, we take a closer look at the resulting `info` struct. Here, we can see the resulting sizes of additive decomposed system parts. The matrix pencil $\lambda E - A$ of the original model had $n_f = 2028$ finite eigenvalues in the left open half-plane and $n_\infty = 154$ infinite eigenvalues, which correspond to the algebraic equations of the system. The reduced-order model instead has $r = 262$ states, where the matrix pencil $\lambda \hat{E} - \hat{A}$ has only finite stable eigenvalues left, i.e., the $\hat{E}$ matrix

is now invertible and the reduced-order system can be treated as (2). In other words, the reduced-order model generated by MORLAB can now be used in all the system-theoretic routines in MATLAB, which was not possible for the original descriptor system.

To analyze the quality of the resulting reduced-order model, we plotted the frequency response and the absolute error of the original and reduced-order system using the `ml_sigmaplot` function from MORLAB. The results can be seen in Fig. 2. The two transfer functions are indistinguishable in the eye ball norm and the absolute error shows the good approximation behavior of the reduced-order model. Additionally, we added the absolute error bound to the plot that was provided in the `info` struct.

Remark 3. As for the standard system case, MORLAB supports discrete-time descriptor systems with appropriate model reduction methods.

3.3 Extension for Second-Order Systems

A very important but currently unsupported system class in the Control System Toolbox and Simulink are linear systems involving second-order time derivatives

$$\begin{aligned} M\ddot{x}(t) + E\dot{x}(t) + Kx(t) &= B_u u(t), \\ y(t) &= C_p x(t) + C_v \dot{x}(t) + Du(t), \end{aligned} \quad (4)$$

with $M, E, K \in \mathbb{R}^{n \times n}$, $B_u \in \mathbb{R}^{n \times m}$, $C_p, C_v \in \mathbb{R}^{p \times n}$, $D \in \mathbb{R}^{p \times m}$. Those systems usually arise in the modeling process of mechanical or electro-mechanical systems. In principle, every second-order system (4) can be rewritten as a first-order system (3), e.g., by introducing an extended state vector $q(t) = [x(t)^T, \dot{x}(t)^T]^T$. The drawback of this process is the resulting first-order system of order $2n$, for which most computational methods ignore the internal second-order structure. This makes computations with second-order systems potentially more expensive than they need to be. This can be avoided by directly working on the second-order system structure (4). In MORLAB, the struct data type “`ssoss`” is used to handle second-order system objects. Appropriate evaluation tools for the time and frequency evaluation that work directly with the second-order structure are implemented in the toolbox to give, as addition to established software, the opportunity to efficiently work with second-order systems.

Corresponding to the direct use of the second-order structure in analysis and evaluation of dynamical systems, the application of model order reduction should preserve the system structure, i.e., it is beneficial if a reduced-order model of (4) has the same structure

$$\begin{aligned} \widehat{M}\ddot{\hat{x}}(t) + \widehat{E}\dot{\hat{x}}(t) + \widehat{K}\hat{x}(t) &= \widehat{B}_u u(t), \\ y(t) &= \widehat{C}_p \hat{x}(t) + \widehat{C}_v \dot{\hat{x}}(t) + \widehat{D}u(t), \end{aligned} \quad (5)$$

with $\widehat{M}, \widehat{E}, \widehat{K} \in \mathbb{R}^{r \times r}$, $\widehat{B}_u \in \mathbb{R}^{r \times m}$, $\widehat{C}_p, \widehat{C}_v \in \mathbb{R}^{p \times r}$, $\widehat{D} \in \mathbb{R}^{p \times m}$. This not only allows to use (5) directly as a surrogate model that can be handled by the same

Fig. 3. Time simulation of the artificial fishtail example.

tools as (4) but usually a structure-preserving approximation is more accurate than an unstructured one and also might allow for interesting re-interpretations of the reduced-order system quantities. MORLAB provides several structure-preserving model reduction methods for (4).

As example, we take a look at the benchmark model [13], describing the movement of an artificial fishtail designed for autonomous underwater vehicles. The model has a mechanical second-order structure (4) with $C_v = 0$, $D = 0$ and $n = 779\,232$, $m = 1$, $p = 3$. It is crucial to employ the second-order structure since the corresponding first-order realization would be of order $\approx 1.5 \cdot 10^6$. Due to the size of the model, the dense model reduction methods from MORLAB cannot be applied directly to the system. Therefore, we use the same approach as in [7, 12] by applying a structure-preserving pre-reduction following the theory from [1] with the same setup as in [7]. The second-order frequency-limited balanced truncation method (`ml_ct_soss_flb`) was then applied to the pre-reduced model of order 100 for the frequency interval $[0, 20]$ Hz. As this MORLAB routine allows for computing several reduced-order models at once, we created reduced-order models for the 8 different balancing formulas from [7]. Independent of the balancing formula, the reduced-order models were of order 1 and stable.

To test the results, the original system and one of the reduced-order models, namely the one created by the `so` formula (see [7] for details), were simulated with the input signal

$$u(t) = 2500(\sin(10\pi(t - 1.35)) + 1)$$

using the MORLAB function `ml_ct_soss_simulate_ss22`. Figure 3 shows the second output entry with the displacement of the fishtail tip in y-direction. In the simulation, no difference between the full- and reduced-order models is visible. To compare, the simulation of the full-order model took around 4 h.

The main computational work for the model order reduction was the intermediate model, which took 21 h followed by the computation of the final reduced-order models with 0.45 s. The simulation of the reduced-order model took then 0.06 s using the same time discretization scheme as for the original model, i.e., the reduced-order model allows now for real time simulations, which could be even implemented on an onboard chip in a fish drone.

References

1. Beattie, C.A., Gugercin, S.: Interpolatory projection methods for structure-preserving model reduction. *Syst. Control Lett.* **58**(3), 225–232 (2009). <https://doi.org/10.1016/j.sysconle.2008.10.016>
2. Benner, P.: 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, pp. 19–24, October 2006. <https://doi.org/10.1109/CACSD-CCA-ISIC.2006.4776618>
3. Benner, P., Byers, R.: Disk functions and their relationship to the matrix sign function. In: Proceedings of European Control Conference, ECC 97, p. 936. BELWARE Information Technology, Waterloo, Belgium (1997). CD-ROM
4. Benner, P., Werner, S.W.R.: Balancing related model reduction with the MORLAB toolbox. *Proc. Appl. Math. Mech.* **18**(1), e201800083 (2018). <https://doi.org/10.1002/pamm.201800083>
5. Benner, P., Werner, S.W.R.: Model reduction of descriptor systems with the MORLAB toolbox. *IFAC-PapersOnLine* **51**(2), 547–552 (2018). <https://doi.org/10.1016/j.ifacol.2018.03.092>. 9th Vienna International Conference on Mathematical Modelling MATHMOD 2018, Vienna, Austria, 21–23 February 2018
6. Benner, P., Werner, S.W.R.: MORLAB - Model Order Reduction LABoratory (version 5.0) (2019). <https://doi.org/10.5281/zenodo.3332716>. <https://www.mpi-magdeburg.mpg.de/projects/morlab>
7. Benner, P., Werner, S.W.R.: Frequency- and time-limited balanced truncation for large-scale second-order systems. e-print 2001.06185, arXiv (2020). <http://arxiv.org/abs/2001.06185>. math.OC
8. Benner, P., Werner, S.W.R.: MORLAB - The Model Order Reduction LABoratory. e-print 2002.12682, arXiv (2020). <https://arxiv.org/abs/2002.12682>. cs.MS
9. Freitas, F., Martins, N., Varricchio, S.L., Rommes, J., Veliz, F.C.: Reduced-order transfer matrices from RLC network descriptor models of electric power grids. *IEEE Trans. Power Syst.* **26**(4), 1905–1916 (2011). <https://doi.org/10.1109/TPWRS.2011.2136442>
10. Freitas, F., Rommes, J., Martins, N.: Gramian-based reduction method applied to large sparse power system descriptor models. *IEEE Trans. Power Syst.* **23**(3), 1258–1270 (2008). <https://doi.org/10.1109/TPWRS.2008.926693>
11. Roberts, J.D.: Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Int. J. Control* **32**(4), 677–687 (1980). <https://doi.org/10.1080/0020717800892288>. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971)

12. Saak, J., Siebelts, D., Werner, S.W.R.: A comparison of second-order model order reduction methods for an artificial fishtail. *at-Automatisierungstechnik* **67**(8), 648–667 (2019). <https://doi.org/10.1515/auto-2019-0027>
13. Siebelts, D., Kater, A., Meurer, T., Andrej, J.: Matrices for an artificial fishtail. Hosted at MORwiki - Model Order Reduction Wiki (2019). <https://doi.org/10.5281/zenodo.2558728>