Bernard Haasdonk

13 MOR software

Abstract: This chapter is devoted to an important requirement of successful model order reduction (MOR) application, namely, the software aspect. The most common situation is the existence of a so-called full model, i. e., a high-fidelity, high-dimensional simulation model, that needs to be accelerated by MOR techniques, optimally without reimplementing the partially complex reduction techniques, as presented in the first volume of this handbook.

Initially, as neither full simulation models nor MOR algorithms are to be reprogrammed, but ideally are reused from existing implementations, we concentrate on the aspect of the interplay of such packages. We will discriminate, discuss, and exemplify different levels of solver "intrusiveness" that allow corresponding reduction techniques to be applied. On the one hand, most effective MOR techniques require deep access into the full model's simulation code. On the other hand, application-specific full model simulators may only offer very restricted access to internals, especially in case of commercial packages. This gap in requirements and practical accessibility motivates the discrimination into "white-box," "gray-box," and "black-box" simulation scenarios. In particular, we exemplify the ideal case of MOR for white-box situations on two examples, namely, parametric linear elliptic PDE and parametric nonlinear ODE systems. Depending on those access classes, different corresponding reduction techniques can be applied.

The second part of the current chapter then discusses existing MOR software. Several program packages exist which provide MOR techniques. They differ in availability, licensing, programming language, system types, physical application domains, external simulator bindings, etc. We give an overview of the most relevant of those MOR packages, such that applicants can identify potential suitable software library candidates.

Keywords: Model order reduction, reduced basis methods, software

MSC 2010: 65D15, 65-04, 93C15, 68N30

Acknowledgement: The author acknowledges feedback from P. Buchfink, D. Wittwar, S. Shuva, A. Schmidt, J. Rommes, N. Walker, J. Saak, C. Himpe, F. Ballarin, S. Werner, M. Cruz Varona, J. Wollner, and S. Rave which helped to improve the presentation and pointed to further software packages.

Bernard Haasdonk, Institute of Applied Analysis and Numerical Simulation, University of Stuttgart, Stuttgart, Germany, e-mail: haasdonk@mathematik.uni-stuttgart.de

https://doi.org/10.1515/9783110499001-013

13.1 Introduction

Following the main motivation of this handbook, we recall that model order reduction (MOR) is a key technique to enable high-level simulation scenarios. By reducing the dimensionality or order of a high-fidelity or "full" model, the generation of a "reduced" model is expected to give approximate but still accurate results while decreasing runtime, memory, and ideally financial costs for simulation and product development. By reduced models, more complex simulation settings are possible beyond single forward solves: "Multiquery" sampling is possible enabling uncertainty quantification or surrogate-based optimization, and "real-time" response is ideally achieved enabling interactive design or real-time control applications. We want to refer to the multitude of textbooks that have appeared during the last two decades such as [35, 4, 2, 30]. The key of efficient model reduction is an "offline-online decomposition," which relates to the separation of the MOR into two phases: The reduced model construction phase (possibly computationally intensive, performed only once) is denoted "offline phase," while the execution of the reduced model (ideally computationally cheap, performed multiple times) is very fast due to small memory and computational demands.

In this chapter we want to discuss from a rather general level some perspectives on the state and perspectives of the interplay between high-fidelity simulation software packages and MOR algorithms and code. Currently, many commercial simulation software packages exist, but few offer MOR technology or access to required internals. By this, such existing software packages cannot straightforwardly be combined with MOR and can thus not be used for modern simulation tasks, potentially representing an economic disadvantage.

On the other hand, MOR researchers have a high interest to make use of these commercial packages due to efficiency, accessibility to industrial-sized relevant applications, etc. As a result, MOR researchers frequently struggle or fail to get their algorithms to work with such commercial tools. Typically, they then either are content with "toy examples," and implement the high-dimensional solvers by themselves, or try to use the existing commercial packages by workarounds or "hacks." The ideal way, though, of developing elegant and goal-oriented interface access between solvers and MOR technology is rarely realized.

The ideal and frequent goal of MOR is to have reduced models that are computationally independent of the full dimension during the online phase. This is called "ideal online efficiency." In order to obtain this, a suitable offline-online decomposition must realize suitable interaction between the full-order and the reduced-order model. In particular for obtaining ideal online efficiency, MOR techniques are typically very code-intrusive in the sense that specific details of the full model must be accessed in an efficient manner. If a full simulator provides all of this necessary information to the outside by suitable functionality, we call such reduction scenario "white-box," as

from an MOR viewpoint the full model is fully transparent. As mentioned earlier, high-fidelity simulation packages are frequently not fully capable to provide the required ingredients, because of "closed-source" policies, code inaccessibility, code complexity, or algorithm strategy. For instance, if a full model performs discretizations in a matrix-free fashion by full mesh traverse, or a package only can assemble full finite element matrices, then a rapid partial evaluation of discretization operators or single matrix entries is difficult without (costly) traversing the mesh or assembling the global matrix, which is relevant for many MOR procedures.

In particular, beside the clean, transparent, and optimal white-box scenarios mentioned earlier, we want to introduce notions of "gray-box" and "black-box" scenarios, where the high-fidelity model provides partial or none of the required information. For such situations, there exist several practical solution strategies, workarounds, or "hacks" to enable MOR without full access to required ingredients. We will particularly also address such strategies.

As a general observation, the tedious and time-intensive development of software is not warranted suitable acknowledgement from the scientific communities in applied sciences. Regrettably, this is always just expected and accepted as a by-product of some main disciplinary scientific work/progress. This is reflected in this chapter, as we will not have many references to journal articles but mostly students theses, proceedings articles, presentations, or PhD theses that are partially devoted to the aspect of MOR software.

By nature, many reduction methods will be mentioned in this chapter, and most of these techniques are described in detail in Volumes 1 and 2 of the current handbook. In order to avoid an abundant reference list, at first use of the methods, we solely refer to the corresponding chapters.

This contribution should serve both applicants as well as MOR researchers: Applicants can identify which kind of internals they could or are willing to provide for the most elementary reduction techniques. Correspondingly, they can decide about which software packages to use and which "hacks" exist to enable MOR for their problem. Scientists working on MOR software should get an incentive to extend or contribute to existing academic MOR software packages.

This chapter is structured as follows. In the next section, we exemplify the interplay between full simulation packages and MOR algorithms. For this we choose two model examples that in certain sense represent different "corners" in the space of MOR model problems. The interplay of full and MOR software packages needs to be realized for white-box and gray-box, as well as black-box scenarios, and we comment on some main coupling techniques. In Section 13.3 we then give an overview of existing MOR software packages, both toolboxes of existing commercial simulator packages and packages developed at academic institutions. We give short characterizations of the packages such that users have some guideline on which package to choose in their respective application. We conclude in Section 13.4 with a summary and some recom-

mendations for both commercial simulator developers/companies and academic MOR researchers.

13.2 Interplay of full and reduced solvers

In order to clarify and illustrate the needs and challenges in interplay of MOR software with full solvers, we remain conceptional in this section, but still are very specific in terms of two model types. We address both the reduced basis (RB) methods community, aiming at solving parameterized partial differential equations (PDEs) (Chapters 1, 4, and 6 in Volume 2 of *Model order reduction*), and control theory researchers, aiming at ordinary differential equation (ODE) systems. Therefore, we choose two model examples representing those fields. For those models we exemplify each a plain vanilla state-of-the-art reduction scheme, which despite simplicity is intrusive in the sense that it requires deep access into the full solver. Therefore, we denote these as white-box approaches. Correspondingly, we exemplify how, in realistic cases with limited information about or restricted access to the full model, MOR can still be realized in gray-box or black-box scenarios.

As our first model we select a parametric variational form, e. g., as appearing by finite element discretizations of a linear elliptic PDE. As our second model we use a nonlinear ODE system. This covers both a linear and a nonlinear, a steady and an unsteady, and a parametric and a nonparametric case.

For these models and reduction scenarios, we necessarily must repeat some notation and terminology which appears at various places within this handbook, but which we consider essential to make the point. In the last subsection, we comment on possible coupling techniques of full and reduced solver packages. Readers who are familiar with those concepts or who are primarily interested in MOR software packages may skip this section and directly continue with Section 13.3.

13.2.1 Model 1: parametric stationary variational problem

We assume to have a Hilbert space X of functions, which is assumed to be finite-dimensional of high dimension n and spanned by basis functions $\psi_i, i = 1, \dots, n$. We want to solve a linear parametric variational problem as is typical in RB methods (Chapters 1, 4, and 6 in Volume 2 of *Model order reduction* or [14, 17]), omitting the output for ease of presentation: For a parameter $\mu \in \mathcal{P}$ from a parameter domain $\mathcal{P} \subset \mathbb{R}^p$, find a solution $u(\mu) \in X$ such that

$$a(u(\mu), v; \mu) = f(v; \mu), \quad v \in X. \tag{13.1}$$

Here, for any μ, the form $a(\cdot,\cdot;\mu) : X \times X \to \mathbb{R}$ is bilinear and $f(\cdot;\mu) : X \to \mathbb{R}$ is linear. Introducing the system matrix and right-hand side

$$\mathbf{A}(\mu) := \left(a(\psi_j, \psi_i; \mu)\right)_{i,j=1}^{n} \in \mathbb{R}^{n\times n}, \quad \mathbf{f}(\mu) := \left(f(\psi_i; \mu)\right)_{i=1}^{n} \in \mathbb{R}^{n}, \tag{13.2}$$

the solution is obtained by solving the corresponding linear system for the degree of freedom vector $\mathbf{x}(\mu) = (x_i(\mu))_{i=1}^{n} \in \mathbb{R}^n$ and subsequent linear combination

$$\mathbf{A}(\mu)\mathbf{x}(\mu) = \mathbf{f}(\mu), \quad u(\mu) = \sum_{i=1}^{n} x_i(\mu)\psi_i. \tag{13.3}$$

The problem is typically assumed to be solvable, e. g., by ellipticity or inf-sup stability assumptions. Note that by choosing $X = \mathbb{R}^n$, ψ_i being the standard basis, we obtain $u(\mu) = \mathbf{x}(\mu)$ and hence this model formulation also simply covers parametric linear equation systems of the form (13.3) without the need for a variational form (although such can easily be constructed [14]).

To enable efficient model reduction, the parametric dependency is assumed to be based on parameter separable forms ("affine assumption"), i. e., there exist Q_a coefficient functions $\theta_a^q : \mathcal{P} \to \mathbb{R}, q = 1, \ldots, Q_a$, which can be rapidly evaluated, and parameter-independent system components $\mathbf{A}^q \in \mathbb{R}^{n\times n}$ such that

$$\mathbf{A}(\mu) = \sum_{q=1}^{Q_a} \theta_a^q(\mu)\mathbf{A}^q$$

and a similar expansion for $\mathbf{f}$ using coefficients $\theta_f^q(\mu)$ and components $\mathbf{f}^q, q = 1, \ldots, Q_f$. We denote the inner product matrix of the space X as

$$\mathbf{K} := \left(\langle \psi_i, \psi_j \rangle\right)_{i,j=1}^{n} \in \mathbb{R}^{n\times n}, \tag{13.4}$$

which allows to compute norms (or errors, projections, orthogonalizations, Riesz representers for error estimation, etc.), e. g., $\|u(\mu)\|_X = \sqrt{\mathbf{x}(\mu)^T\mathbf{K}\mathbf{x}(\mu)}$.

13.2.1.1 White-box reduction scenario

In parametric problems we discriminate between the offline phase (reduced model construction, potentially computationally expensive and involving full system components or solves) and the online phase (assembly and solve of reduced system, rapidly executable). The latter can then be evaluated in many-query contexts. As reduction technique we consider Galerkin projection. For this, in the offline phase, a matrix $\mathbf{V} \in \mathbb{R}^{n\times r}$ is constructed, where $r \ll n$ indicates the reduced dimension. In RB methods, $\mathbf{V}$ can be obtained by various techniques, e. g., as concatenation of solution snapshots

(Lagrangian basis), Gram–Schmidt orthogonalization of such, proper orthogonal decomposition (POD) (Chapter 2 in Volume 2 of *Model order reduction*), or greedy procedures. All of those have in common that several full system solves (for the snapshots) must be executed and, as mentioned earlier, the inner product matrix $\mathbf{K}$ may be required for orthogonalization.

After basis matrix generation, the full system components are projected,

$$\mathbf{A}_r^q := \mathbf{V}^T \mathbf{A}^q \mathbf{V} \in \mathbb{R}^{r\times r}, \quad \mathbf{f}_r^q := \mathbf{V}^T \mathbf{f}^q \in \mathbb{R}^r \tag{13.5}$$

for q ranging from 1 to Q_a and Q_f, respectively. This concludes the offline step.

Then in the online step, for any given parameter $\mu \in \mathcal{P}$ only the coefficient functions need to be evaluated and the reduced system is assembled by linear combination,

$$\mathbf{A}_r(\mu) := \sum_{q=1}^{Q_a} \theta_a^q(\mu)\mathbf{A}_r^q, \quad \mathbf{f}_r(\mu) := \sum_{q=1}^{Q_f} \theta_f^q(\mu)\mathbf{f}_r^q. \tag{13.6}$$

The reduced solution then results via its r-dimensional coefficient vector $\mathbf{x}_r(\mu) \in \mathbb{R}^r$ from solving

$$\mathbf{A}_r(\mu)\mathbf{x}_r(\mu) = \mathbf{f}_r(\mu).$$

If reconstruction of the approximate solution is desired (e. g., for visualization, etc.), the approximation $\tilde{\mathbf{x}}(\mu) \in \mathbb{R}^n$ of $\mathbf{x}(\mu)$ is obtained as

$$\tilde{\mathbf{x}}(\mu) = \mathbf{V}\mathbf{x}_r(\mu).$$

By realizing this reduction strategy, an ideal online efficiency is obtained, meaning that the reduced computations in the online stage do not involve any operations of high complexity n, but only of complexity depending on small quantities r, Q_a, Q_f. If a full discretization scheme provides all the required internals to realize the above reduction and reduced solution steps, we call it a white-box reduction scenario.

Figure 13.1 illustrates and summarizes this decomposition and white-box interplay of full-order and reduced-order simulation software.

Note that instead of providing the full-dimensional matrices $\mathbf{K}, \mathbf{A}^q$, it can as well be sufficient that the solver provides suitable matrix–vector multiplication routines. By this, also PDE discretization schemes that work with matrix-free implementations can be used in a white-box fashion.

13.2.1.2 Gray-box reduction scenarios

Now, in practice, the full solver package may not give access to all of the details. In particular, we are not aware of any commercial simulator package giving explicit access

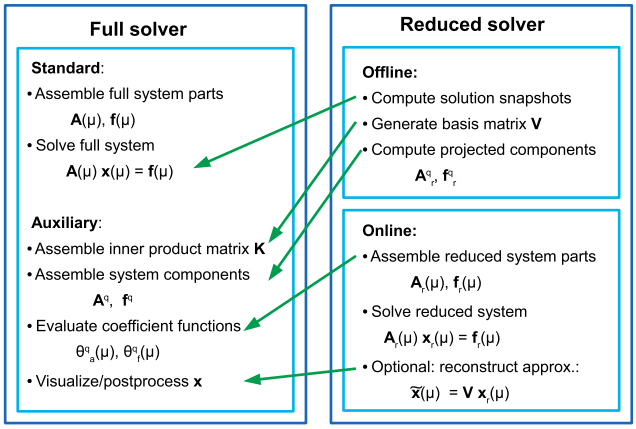

Figure 13.1: Required functionality and interplay of full (high-dimensional) and reduced solver in white-box MOR scenario for stationary variational problems.

to parametric matrix components and coefficient functions. We thus call this scenarios gray-box, as the full solver does not give complete insight into the problem, but still MOR is desired.

Missing K

A first problem may consist in the situation that the full model does not provide an inner product matrix (13.4). This prevents measuring errors, computing projections, and orthogonalizations in the correct function space norms.

If geometry information of the underlying mesh and information about the discrete function spaces (polynomial degree, order of the nodes) is available, this matrix may be constructable "by hand" outside of the simulator code. Still this is of the same technical complexity as assembling a finite element matrix, which may require considerable effort.

Alternatively, instead of working with the correct $\mathbf{K}$, one could choose certain alternative matrices. This corresponds to choosing another inner product on the solution space. A common choice is $\mathbf{K} := \mathbf{I}_n$, the identity, i. e., choosing $X = \mathbb{R}^n$ as standard Euclidean space. Note that this approach is simple but may lead to severely suboptimal bases as the meaning/weighting of the different entries of the degree of freedom vector is not respected. For example, if those degrees of freedom relate to values on grid cells of largely varying size, the small cells are given the same weight as the large cells, which may mislead basis generation (e. g., when computing a POD without proper weighted inner product) and thus deteriorate the global accuracy. This choice is common in engineering practice, but should be applied with care. Another choice for the inner product matrix could be $\mathbf{K} = \mathbf{A}(\bar{\mu})$ for a fixed reference parameter $\bar{\mu}$ if the system provides a symmetric, positive definite system matrix. This choice actually cor-

responds to the "energy inner product" in case of thermal diffusion problems, which is known to be beneficial for error estimation [14].

Missing parameter separable decomposition

Most prominently the parameter separable decomposition may not be available. A first modification of the procedure could consist of the following. One can omit to work with components $\mathbf{A}_r^q, \mathbf{f}_r^q$, and instead directly assemble the reduced system by projection of the system matrices during the online phase,

$$\mathbf{A}_r(\mu) = \mathbf{V}^T\mathbf{A}(\mu)\mathbf{V}, \quad \mathbf{f}_r(\mu) = \mathbf{V}^T\mathbf{f}(\mu). \tag{13.7}$$

We remark that the definitions of the reduced system and the parameter separable decomposition in the previous subsection exactly yield this equivalent representation in the case of parameter separable decomposition. Thus this is no approximation step but rather a reformulation of computing the reduced system. However, in (13.7) there is a computational bottleneck, as the assembly of the full system $\mathbf{A}(\mu)$ and $\mathbf{f}(\mu)$ is expensive, in particular polynomial in n. Hence, the online efficiency is sacrificed.

A second way of solving the missing parameter separability is to produce an approximate parameter separable approximation of the problem. Several approaches can be found in the literature that help to solve the issue of the missing separable decomposition in the system components. For the right-hand side vector the discrete empirical interpolation method (DEIM) [8] can be applied, which will be treated in more detail in the context of nonlinear unsteady problems in Section 13.2.2. A variant of this procedure has been formulated for matrices, which is called the matrix DEIM (MDEIM) [40, 28]. In the online phase this method only requires the evaluation of a few matrix entries $(\mathbf{A}(\mu))_{i_m,j_m}$ for a set of M "magic index pairs" $(i_m, j_m), m = 1, \ldots, M$. Still, this pointwise assembly might not be possible or highly inefficient with a given solver package, for instance, if it only can assemble the complete system matrix and not single entries. Then obviously, extraction of single matrix entries has complexity polynomial in n due to the required assembly of the complete matrix. But if this local entry assembly is possible in an effective way, then this procedure can be online-efficient. Also, the procedure can generate an exact parametric representation: If the parametric matrices lie within a finite-dimensional space, this MDEIM procedure will find such an exact representation, where M is exactly the dimension of this covering finite dimensional space.

An alternative can be parametric regression or interpolation of system matrices [41], denoted as "operator extraction": Based on a given set of system matrix snapshots, a polynomial approximation or interpolation in the parameters is realized and successfully used. In the online phase, access to neither $\mathbf{A}(\mu)$ nor its entries is required, recovering the ideal online efficiency. However, no error control for assessing the parametric approximation quality is possible by this approach.

We want to mention another approach in a PDE context, which was coined the "two-grid finite element/reduced basis approach" [7]: One constructs a RB on the given (fine) grid in a traditional way, e. g., Lagrangian, POD, or greedy basis. Then in the online phase, for a new parameter, a full problem is solved but on a coarser mesh. Finally, this coarse mesh solution is projected on the obtained (fine mesh) RB. By this, missing details on the coarse mesh are potentially included in the projected solution, as such fine details are contained in the fine mesh snapshots/RB. The computational cost in the online phase clearly is not online-efficient as it depends on the coarse mesh resolution. But it still can be computationally faster than solving the full problem on the fine mesh. In this method, internal details on the geometry, in particular the nesting of the two meshes, are required, and details on the discrete function spaces, in order to compute the prolongation operator for mapping the coarse mesh solution degree of freedom vector to a fine mesh degree of freedom vector, one further needs the inner product matrix for projection onto the RB.

So overall, in the case of the missing parameter separable decomposition gray-box scenario, either one sacrifices the ideal online efficiency while maintaining accuracy of the reduced model, or one must accept another approximation stage in the reduction chain for obtaining the optimal online runtime complexity.

These are the most common approaches of workarounds or "hacks" around missing system information for stationary problems.

13.2.1.3 Black-box reduction scenario

We denoted the previous section as gray-box, even if some references rather call their approaches black-box. The reason for our nomenclature is that the mentioned approaches still require some insight into the discretization or sampling of system components, i. e., knowledge of and access to the system structure. In contrast to this, the real notion black-box would be even more restrictive as not allowing any insight into the system components or discretization. The most limiting situation would be that the system only is observable via input–output pairs $(\mu_i, \mathbf{x}(\mu_i))$, $i = 1, \ldots, n_{\text{train}}$. Then, based on these training data, machine learning approaches could be used to infer a functional relation between the input parameters and the solution (degree of freedom vector), e. g., kernel methods (Chapter 9 in Volume 1 of *Model order reduction*), or neural networks. As a learning of a high-dimensional quantity (degree of freedom vector $\mathbf{x}(\mu)$) may suffer from the curse of dimensionality, we recommend to first identify a subspace (e. g., by POD) with suitable basis matrix $\mathbf{V}$ corresponding to an orthogonal basis; then the training data can be projected orthogonally to the reduced space $(\mu_i, \mathbf{V}^T\mathbf{K}\mathbf{x}(\mu_i))$ and the mapping from input parameters to reduced state vectors can be learned. Such an approach, however, does not involve any knowledge about the system structure. Alternative approaches would consist of assuming a certain (parametric) system structure and inferring the missing parameters purely from the observed

data. These approaches in MOR are denoted to be "data-driven." In control theory such problems already have a long tradition in the area of system identification. As this, however, is not so much related to MOR software and interplay between full and reduced solvers, we do not comment on such options.

13.2.2 Model 2: time-dependent ODE systems

As second model we assume to have a time-dependent nonlinear ODE system,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{13.8}$$

on a time interval $[0, T]$ with final time $T \in \mathbb{R} \cup \{\infty\}$, unknown state $\mathbf{x} : [0, T] \to \mathbb{R}^n$, input $\mathbf{u} : [0, T] \to \mathbb{R}^{n_u}$, and system nonlinearity $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^{n_u} \to \mathbb{R}^n$. Continuity of the right-hand side then implies existence of a solution. Additionally, there may be an output $\mathbf{y} : [0, T] \to \mathbb{R}^{n_y}$, which, for simplicity, we assume to be linear in the state

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t)$$

for $\mathbf{C} \in \mathbb{R}^{n_y \times n}$. We now focus on local evaluations of the system nonlinearity, as this will be the key for full online-efficient reduction. For a given set of few indices $I = \{i_1, \dots i_M\} \subset \{1, \dots n\}$ with typically $M \ll n$ (e. g., DEIM "magic indices," Chapter 5 in Volume 2 of *Model order reduction*), we define

$$\mathbf{f}_I := (f_{i_1}, \dots, f_{i_M})^T \tag{13.9}$$

as local evaluation of $\mathbf{f} = (f_i)_{i=1}^n$. We now require the following, which are typically nontrivial for given off-the-shelf simulation packages:

(i) The local evaluation $\mathbf{f}_I$ can be computed without assembling the full nonlinearity $\mathbf{f}$.[1]

(ii) The evaluation of those components can be performed rapidly based on only a subset of the state vector $\mathbf{x} = (x_i)_{i=1}^n$ in the following sense: There exists an input index subset $\bar{I} := \{\bar{i}_1, \dots \bar{i}_{\bar{M}}\}$, typically with $n \gg \bar{M} \geq M$, that defines $\mathbf{x}_{\bar{I}} := (x_{\bar{i}_1}, \dots, x_{\bar{i}_{\bar{M}}})^T \in \mathbb{R}^{\bar{M}}$ as the restriction of $\mathbf{x}$ to the indices $\bar{I}$. Then we assume that there exist functions $\bar{f}_{i_m} : \mathbb{R}^{\bar{M}} \times \mathbb{R}^{n_u} \to \mathbb{R}, m = 1, \dots M$, such that $f_{i_m}(\mathbf{x}, \mathbf{u}) = \bar{f}_{i_m}(\mathbf{x}_{\bar{I}}, \mathbf{u})$. This means that overall there is a function $\bar{\mathbf{f}}_I := (\bar{f}_{i_1}, \dots, \bar{f}_{i_M})^T : \mathbb{R}^{\bar{M}} \times \mathbb{R}^{n_u} \to \mathbb{R}^M$ satisfying

$$\bar{\mathbf{f}}_I(\mathbf{x}_{\bar{I}}, \mathbf{u}) = \mathbf{f}_I(\mathbf{x}, \mathbf{u}) \tag{13.10}$$

---

1 Note that in the DEIM literature frequently the sampling matrix $\mathbf{P} := [\mathbf{e}_{i_1}, \dots \mathbf{e}_{i_M}] \in \mathbb{R}^{n \times M}$ is defined, where $\mathbf{e}_i \in \mathbb{R}^n$ denote the standard Euclidean basis vectors. Then we verify that $\mathbf{f}_I = \mathbf{P}^T\mathbf{f}$, but the latter equation may give the wrong understanding that the sampling matrix multiplication is a computational recipe, which it is not, as this is of complexity n. Thus, we refrain from adopting this DEIM notation here and use the definition (13.9).

for any $\mathbf{x}$ and $\mathbf{u}$. We emphasize that $M, \bar{M}, n_u$ typically are small, thus this function $\bar{\mathbf{f}}_I$ can be evaluated in complexity independent of n, and hence is online-efficient. If the system corresponds to a discretized PDE these properties are useful (and realistic): Property (i) corresponds to a local evaluation of the system nonlinearity in a few grid points. Property (ii) means that such a pointwise evaluation, e. g., of a finite difference pencil, only requires the knowledge of the state in a local neighborhood around the evaluation grid points, also understandable as a submesh. This property is typical for discretizations of differential operators using basis functions of local support such as finite difference, finite element, finite volume, or discontinuous Galerkin bases. If the model has some physical meaning, there may exist a weight matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ that represents a physically relevant inner product and norm $\|\mathbf{x}\|_{\mathbf{K}} := \sqrt{\mathbf{x}^T\mathbf{K}\mathbf{x}}$.

13.2.2.1 White-box reduction scenario

Note that for linear time-invariant (LTI) systems, i. e., $\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, there are plenty of reduction strategies, e. g., approximating the state trajectories for special input signals (snapshot-based such as POD, POD-greedy using the proper inner product given by $\mathbf{K}$) or approximating the input–output behavior in some sense (e. g., optimal $\mathcal{H}_2$-approximation, moment-matching, Hankel norm approximation, etc. [2]). Again for simplicity, we assume to have some basis $\mathbf{V} \in \mathbb{R}^{n\times r}$ with $r \ll n$ of orthogonal columns obtained by orthogonalizing any basis matrix of any of the mentioned procedures.

Then, a straightforward Galerkin projection of the system results in an approximation $\tilde{\mathbf{x}} := \mathbf{V}\mathbf{x}_r$ with $\mathbf{x}_r : [0, T] \to \mathbb{R}^r$ being the solution of

$$\dot{\mathbf{x}}_r(t) = \mathbf{V}^T\mathbf{f}(\mathbf{V}\mathbf{x}_r(t), \mathbf{u}(t)), \quad \mathbf{x}_r(0) = \mathbf{x}_{r,0}, \tag{13.11}$$

with initial data projection $\mathbf{x}_{r,0} := \mathbf{V}^T\mathbf{x}_0$ and potential output approximation $\tilde{\mathbf{y}}(t) = \mathbf{C}\tilde{\mathbf{x}}(t) = \mathbf{C}\mathbf{V}\mathbf{x}_r(t)$. Even though this is a low-dimensional model, it is not online-efficient, as (a) the reconstruction $\mathbf{V}\mathbf{x}_r$, (b) the nonlinearity evaluation, and (c) the projection $\mathbf{V}^T\mathbf{f}$ are operations of the original complexity n. This is a well-known computational bottleneck of reduction methods such as POD. In fact, the integration of (13.11) may even be more expensive than integrating the original system (13.8), rendering the reduced model practically useless. To decrease this computational complexity, sampling-based approximations of the nonlinearity are applied, for example by gappy POD [11] or POD-DEIM [8]. Here, a further approximation stage is employed: An additional basis matrix $\mathbf{U}$ is assumed that approximates $\mathbf{f}$ well by $\tilde{\mathbf{f}} := \mathbf{U}\mathbf{c}$. The coefficient vector $\mathbf{c}$ is computed by a linear transformation of a local nonlinearity evaluation $\mathbf{c} := \mathbf{M}\mathbf{f}_I$ as introduced in (13.9). This can both be an interpolation-type

approximation, e. g., DEIM,[2] but as well a least-squares approximation by oversampling of $\mathbf{f}$. In both cases, $\mathbf{U}$ must approximate the range of $\mathbf{f}$ well, such as obtained by an additional POD of snapshots of the nonlinearity [8], or a greedy basis based on $\mathbf{f}$ snapshots (cf. the early references [16, 15]). If we replace $\mathbf{f}$ by $\tilde{\mathbf{f}}$ in (13.11) and use $\mathbf{V}^T\tilde{\mathbf{f}} = \mathbf{V}^T\mathbf{U}\mathbf{M}\mathbf{f}_I = \mathbf{W}\mathbf{f}_I$ with $\mathbf{W} := \mathbf{V}^T\mathbf{U}\mathbf{M} \in \mathbb{R}^{r\times M}$ and $(\mathbf{V}\mathbf{x}_r)_{\bar{I}} = \mathbf{V}_{\bar{I}}\mathbf{x}_r$ with $\mathbf{V}_{\bar{I}} \in \mathbb{R}^{\bar{M}\times r}$ containing the rows of $\mathbf{V}$ corresponding to the indices in $\bar{I}$, we obtain a "hyperreduced" model as alternative to (13.11):

$$\dot{\mathbf{x}}_r(t) = \mathbf{V}^T\tilde{\mathbf{f}}\big(\mathbf{V}\mathbf{x}_r(t), \mathbf{u}(t)\big) = \mathbf{W}\mathbf{f}_I\big(\mathbf{V}\mathbf{x}_r(t), \mathbf{u}(t)\big) = \mathbf{W}\bar{\mathbf{f}}_I\big(\mathbf{V}_{\bar{I}}\mathbf{x}_r(t), \mathbf{u}(t)\big), \tag{13.12}$$

assigned with the identical initial conditions and a potential output as (13.11).

We give some intuition about the operations involved in solving such a system: In the offline phase the initial state is projected, i. e., $\mathbf{x}_{r,0}$ is computed, and the small matrix $\mathbf{W}$ is assembled. Also we identify the appearance of a restriction of an RB $\mathbf{V}_{\bar{I}}$. This means, for example in the case of a finite difference discretization, that the RB needs to be stored nodewise on "all" finite difference nodes in the stencils around the sampling points I. This matrix is of small size. Now, when turning to the online phase, i. e., integration of (13.12), we clearly see the requirement of repeated evaluation of the local nonlinearity evaluation function $\bar{\mathbf{f}}_I$ that we assumed to be available by the high-fidelity model and should be rapidly evaluated in a complexity independent of n. Also, instead of reconstructing the full state approximation $\mathbf{V}\mathbf{x}_r$ as required in the model (13.11), the hyperreduced model only requires a local reconstruction of the approximated state $\mathbf{V}_{\bar{I}}\mathbf{x}_r$ which corresponds to the values of the reduced solution on a subgrid defined by the indices $\bar{I}$. This local reconstruction of the state then is sufficient to exactly evaluate the system nonlinearity in the local sampling points required for the evolution of the reduced model.

This concludes the "clean" white-box reduction scenario. We are not aware of implementation of all of the full model requirements in commercial packages to realize this. In particular the rapid local evaluation of the system nonlinearity is an extremely intrusive – and mostly also code-intrusive – requirement of making complex discretization packages suitable for online-efficient model reduction. Some publications that presented this ideal reduction for nonlinear problems include [16, 10, 8].

13.2.2.2 Gray-box reduction scenario

Now again, as white-box scenarios are rarely implemented in usual high-fidelity simulation packages, we again must refrain to gray-box scenarios.

---

2 Note that in the case of the DEIM, the matrix $\mathbf{U}$ is of size $n \times M$ and $\mathbf{M} = (\mathbf{P}^T\mathbf{U})^{-1}$. Then obviously the interpolation property is verified:

$$\mathbf{P}^T\tilde{\mathbf{f}} = \mathbf{P}^T\mathbf{U}\mathbf{M}\mathbf{f}_I = \mathbf{P}^T\mathbf{U}\big(\mathbf{P}^T\mathbf{U}\big)^{-1}\mathbf{f}_I = \mathbf{f}_I = \mathbf{P}^T\mathbf{f}.$$

.

Missing inner product weighting matrix K

If no weighting matrix $\mathbf{K}$ is available by the full model, the same workarounds as in Section 13.2.1.2 apply, i. e., working with the Euclidean inner product or reconstructing $\mathbf{K}$ by potential knowledge about the underlying PDE discretization.

Missing local evaluation routine $\bar{\mathbf{f}}_I$

In the context of nonlinear problems, the main problem is a potential missing local evaluation routine. If no access to the full model apart from full samples of $\mathbf{f}$ are possible, then one could sacrifice online efficiency for accuracy and just accept the (inefficient) nonhyperreduced model (13.11). As mentioned above, conceptually a dimension reduction will be obtained by this, but no or limited computational gain is to be expected.

If aiming at online efficiency, one can add yet another approximation stage, e. g., realized by the Kernel-DEIM approach [39, p. 83ff], [22]: The idea is to provide kernel-based approximants for the local function evaluations: $f_{i_m}(\mathbf{V}\mathbf{x}_r) \approx \hat{f}_{i_m}(\mathbf{V}_{\bar{I}}\mathbf{x}_r)$, where $\hat{f}_{i_m}$ is for instance a kernel interpolant or a more general approximant (Chapter 9 in Volume 1 of *Model order reduction*). This can be constructed purely based on state and nonlinearity snapshots.

Missing local evaluation set $\bar{I}$

If the discretization/interpretation of the full model is unknown, it may be a priori unclear which state entries influence the required sampling entries of the nonlinearity, i. e., the index set $\bar{I}$ may be unknown. A general approach for this is suggested: If the full nonlinearity of the system can be sampled, snapshot-based POD can be computed and a DEIM can be executed resulting in a possible choice of sampling points I. Now, the set of indices $\bar{I}$ that the magic index evaluations depend on must be generated. One possibility for this is analysis of the Jacobian of $\mathbf{f}$ (which clearly requires this as additional functionality from the full solver): The nonzero entries in row i_m of the Jacobian indicate those entries of the input state vector that induce changes in the value of $\mathbf{f}$. Thus, analysis of the sparsity pattern of $\nabla\mathbf{f}$ is sufficient for identification of the set $\bar{I}$.

Alternatives

Now clearly, also other gray-box approaches can be followed if one refrains from the above Galerkin projection structure. For instance by working with state and velocity snapshots of a linear system, dynamic mode decomposition (DMD) [25] allows to infer an approximate linear model. Note that velocity, i. e., nonlinearity snapshots, require a sort of intrusiveness as this is more than just state observations, and therefore not black-box. If a system can be observed in terms of input and output observations in frequency space, then the Loewner framework [20] enables to efficiently find an approximate LTI realization of a system with the observed behavior. For parametric un-

steady problems, e. g., LTI systems, the parametric matrix interpolation idea can also be applied (e. g., [13]).

13.2.2.3 Black-box reduction scenario

The mentioned gray-box approaches still assumed to have access to the systems nonlinearity (or frequency response measurements). Now, what can be done if the only way of accessing the system consists of sampling state trajectories and outputs? First, a direct mapping of input state to output quantities can be approximated with machine learning techniques. However, such techniques do not involve any model knowledge nor model assumptions. If one assumes a certain model structure, i. e., linear or polynomial nonlinearity for $\mathbf{f}$, then the system coefficients of a reduced model can be inferred from observed data [31]. In control theory this field is long known as system identification.

If one is interested in approximating $\mathbf{x}$ instead of an output, the prediction of a high-dimensional target quantity may suffer from the curse of dimensionality. So blended approaches combining model reduction and machine learning exist. For example [38] suggest a nonintrusive reduced-order model (ROM) approach for nonlinear problems combining POD and neural networks: By sampling state trajectories, a suitable approximating space/basis $\mathbf{V}$ can be computed by POD. Then each of the state snapshots can be mapped to the POD space, obtaining POD coefficients. Then a neural network can be learned to map time to the POD coefficients, which thus directly predicts an approximation of the state even without integration of a dynamical system.

13.2.3 Coupling techniques

In the above we have seen that online efficiency in MOR is based on a tight interplay of full-order solvers and MOR implementation. This requires information exchange between those levels in suitable ways. Therefore, we now discuss the main two different coupling options for MOR software. As additional reference on the coupling aspect, we want to point to a presentation that also discusses various approaches for MOR interfaces,[3] with a focus on motivating the package pyMOR as mentioned in the next section.

Write high-dimensional data to disk

The principle is the following: The full solver is agnoscent of the reduction scheme and reduction package but provides required high-dimensional data by file output, e. g., finite element method system matrices, or system matrix components (in the case of

3 https://www.stephanrave.de/talks/cse_2015.pdf

parameter separability) L^2 or H^1 scalar product weight matrices, grid description for visualization, etc. Also, the user or the full solver then must provide or export the coefficient functions of the separable parameter decompositions as function strings, etc.

The reduction scheme or MOR package then needs to import these data, project the high-dimensional objects to low-dimensional quantities, assemble the reduced system, and rapidly solve the reduced system. Potentially, the MOR code exports then high-dimensional state approximations for visualization or postprocessing by the full solver package. We refer again to Figure 13.1 for illustration of these exchange steps.

This approach by file exchange is applicable for linear problems or even problems with low polynomial structure or suitably Taylor-approximated systems.

There are several advantages or beneficial options resulting from this decomposition: One can realize a straightforward coupling of different programming platforms for the full and MOR solvers by only agreeing on a joint file format. By this, the MOR code can be written in language of choice and can be agnoscent of the programming language of the full system. Also, full solvers can be easily exchanged. One can realize a full decoupling of high-dimensional offline and online operations.

Some drawbacks and limitations of this approach are apparent: The reduced model simulation might be slow due to the disk access bottleneck. Further, the MOR code needs to process high-dimensional data. File export routines for the full solver must be implemented, which might be nontrivial (e. g., mass matrices in matrix free full solvers). This file exchange does not work for nonlinear problems or problems with complex parametric system components unless gray-box strategies, i. e., "hacks" or approximations of the preceding sections, are applied. Also, this full decomposition of offline and online phases prevents modern adaptive simulation schemes, where offline and online phases are blended (e. g., optimization with reduced model adaptation, local MOR with adaptive basis enrichment, etc.). It may be that iteration of full solver steps, reduced system creation, and solve are impossible or difficult.

Alternatively, the file exchange can also be based only on reduced quantities, if the full solver package is extended by MOR basis generation and projection functionality. Then the MOR package only needs to read reduced quantities, solve reduced systems, and communicate reduced coefficient vectors to the full solver package that is responsible for visualization/postprocessing.

Communication of full and reduced solver by function interfaces

The principle here is that the full solver is extended in order to provide function access to high-dimensional quantities used for the reduction, e. g., inner product matrices, system nonlinearities, etc. An improved approach even consists in avoiding access to matrices, but only providing access to matrix–vector multiplication for inner-product matrices or system matrices.

This offers some clear advantages: It is potentially very fast if optimal and minimally invasive connection to the full solver is realized. Nonlinear problems can be

treated by giving the MOR package access to the exact system nonlinearity. The provisioning of matrix–vector multiplication routines allows matrix-free operations, and hence MOR for discretizations that do not rely on matrix representations (e. g., finite volume or finite difference discretizations). A potential tight and repeated interaction between full and reduced solvers is possible in modern simulation scenarios with adaptive ROMs such as in optimization or local MOR. If those function interfaces are designed in a minimal and generic way, very general model reduction is possible, as various full solvers can implement those interfaces and various MOR packages can use those different full solver interface implementations. If the interface classes make use of (references to) high-dimensional objects stored in the full solver's memory, no communication of high-dimensional data is required, and the MOR code can be very fast and memory-efficient.

Again, also clear disadvantages appear: This approach is more complex as coupling via online bindings, shared libraries, mex-interfaces, or network communication, etc., is required. The realization of the required internal access may be difficult or impossible in certain full solver packages (e. g., local system matrix access to matrix-free discretization operators).

13.3 MOR software packages

After these conceptual aspects of MOR software, we want to become very specific in this section and give an overview of existing MOR software packages. First we address commercial packages, which we understand as being developed by companies with commercial interest. Then, we give a brief survey of academic packages, developed at universities or public research institutions. By the overview of this section, users may be guided (to some extent) to select the suitable packages for their application case.

13.3.1 Commercial packages with MOR functionality

Many simulator packages have realized the need and usefulness of model reduction on top of the traditional simulation chain. The following is a short and certainly incomplete list of such commercial packages. We do not go into details but refer to the corresponding websites. Mostly – but with notable exception of Scilab – the following packages are closed-source and subject to license fees.

ANSYS® (CADFem):[4] The package *Model Reduction inside ANSYS* provides reduction techniques for linear systems, in particular three-dimensional finite element from ANSYS Mechanical™, enabling piezoelectrical and thermomechanical models.

4 https://www.cadfem.de/produkte/cadfem-ansys-extensions/model-reduction-inside-ansys.html

The resulting low-dimensional matrices can be imported in ANSYS Simplorer®, MATLAB/Simulink, or other system simulation languages VHDL-AMS or SPICE. An early reference to the initial version "MOR for ANSYS" has been published [3].

Akselos Integra™(Akselos):[5] This package enables numerical simulation of large-scale mechanical assets. Among others, the underlying technique is the static condensation RB element method [19]. Being based on reduction of components, this technique is very suitable for technical component-based structures. Linear systems can be treated, as well as systems with local nonlinear subsystems.

CST MICROWAVE STUDIO® (CST):[6] In order to obtain network models, that are compatible with the circuit simulator SPICE and have the same port behavior as three-dimensional structures, suitable MOR techniques can be applied. In particular stability and passivity preservation is obtained by the reduction techniques.

MATLAB®[7] (The Mathworks, Inc.):[8] In its *control system toolbox* various control-theoretic reduction techniques are provided, in particular pole zero simplification, mode selection or balanced truncation (BT) (Chapter 2 in Volume 1 of *Model order reduction*). These methods are available in the Model Reducer App, which can be interactively used.

MOR toolbox (MOR DIGITAL SYSTEMS):[9] The MOR toolbox is a MATLAB-based toolbox gathering algorithms for (i) reduction of large-scale linear dynamical models and (ii) creation of linear dynamical models from input–output frequency data. The algorithms gathered in the MOR toolbox generate a linear state-space model whose input–output behavior is close to the initial model. Some methods take advantage of the sparse nature of the models and can therefore be applied to very large-scale models with several thousands of states. Such models arise very often in physics, biology, etc. The MOR toolbox is free for academic use, and licenses are available for commercial purposes.

SLICOT (Niconet e.V.):[10] This collection of MATLAB toolboxes also contains a *SLICOT Model and Controller Reduction Toolbox*, providing algorithms for many control-theoretic reduction methods, e. g., BT, singular perturbation approximation, frequency-weighted balancing, Hankel norm approximation, or co-prime factorization.

SciMOR (ESI Group, Scilab Enterprises SAS):[11] This model reduction toolbox has recently been developed and provides modern techniques for reduction of paramet-

5 https://akselos.com

6 https://perso.telecom-paristech.fr/begaud/intra/MWS_Getting_Started.pdf

7 We acknowledge MATLAB to be a registered trademark throughout this chapter, but for readability refrain from indicating this at various further occasions.

8 https://de.mathworks.com/help/control/ug/about-model-order-reduction.html

9 http://mordigitalsystems.fr/en/

10 http://slicot.org/matlab-toolboxes/model-reduction

11 https://scilab.io/scilab-model-reduction-toolbox

ric PDEs. In particular, POD in combination with parameter interpolation is implemented. This toolbox is part of the Scilab open-source project with the goal of democratizing computational science and is free of charge.

There are some common limitations with most of those MOR implementations: Apart from very recent developments, those commercial packages typically do not provide the most efficient latest MOR methods. This in particular holds true for packages that are already more than 10 years old, as the last decade has shown tremendous improvement in MOR technology. In these packages, in particular the full models are typically not fully accessible. As discussed in the previous sections, this prevents white-box implementation of modern reduction techniques, which mostly require a high level of intrusiveness.

13.3.2 Academic MOR software packages

Next, we want to give short descriptions of existing MOR software packages developed by groups from academia. The packages are free of charge, but may be subject to some licensing options. The packages are mostly open-source, except two of them, which are not publicly available for download. Typically, those packages are developed at universities or public research institutions.

The software packages differ in various aspects. In addition to the open-source policy, the main aspect is the programming language. Differences can also be observed with respect to the ease of installation, whether paper references are given, and whether documentation, benchmark models, or tutorial examples are provided. Some packages are under active development by large development teams, while some are single-programmer projects that resulted from PhD theses and are "frozen" or only updated in a minimal fashion. Large differences exist in the provisioning of full-scale solvers within the package and the extent of the coupling to external high-fidelity models/solvers. The purpose of those packages can be either fundamental research, teaching, or even industrial application. The physical application fields also vary widely: Some packages only support one application domain (e. g., electronics or mechanical systems) while others do not restrict the type of applications, but allow all kinds of physical domains, including electronics, fluid dynamics, biology, chemistry, finance, etc. Most discriminating are the types of systems that can be reduced such as LTI systems, nonlinear ODE systems, parametric problems, elliptic PDEs, parabolic PDEs, hyperbolic PDEs, first-order systems, second-order systems, differential algebraic equations (DAEs), and parametric, dense, or sparse systems. Finally, the implemented reduction techniques are very different in the packages, e. g., BT, moment-matching (Chapter 3 in Volume 1 of *Model order reduction*), RB methods, POD, Hankel norm approximation, optimal $\mathcal{H}_2$-norm reduction, etc.

In Table 13.1 we give some metadata for several packages from academic development teams that are available at the time of finalization of this overview, i. e., June

Table 13.1: Metadata of academic MOR software packages (as of Aug. 2019).

Acronym	Ref.	Language	License type	Latest version
DPA	[33]	MATLAB/Octave	–	Nov. 2015
emgr	[18]	MATLAB/Octave	BSD-2-Clause	5.8, May. 2020
ITHACA	[36]	C++	LGPL 3.0 / MIT License	Mar. 2020
KerMor	[39]	MATLAB	GNU GPL & BSD	0.9, Aug. 2015
M.E.S.S.	[34]	MATLAB/Octave, C	GNU GPL 2	2.0.1, Feb. 2020
MOREMBS	[12]	MATLAB, C++	–	on request
MORLAB	[5]	MATLAB	GNU Affero GPL 3	5.0, Dec. 2019
MORPACK	[26]	MATLAB	–	on request
pyMOR	[27]	Python	BSD-2-Clause	2019.2, Dec. 2019
RBmatlab	[14]	MATLAB	–	1.16.09, Sept. 2016
RBniCS	[17]	Python	GNU lesser GPL 3	v0.1.0, Jun. 2019
SparseRC	[21]	MATLAB	–	Nov. 2011
sssMOR	[6]	MATLAB	BSD-2-Clause	v2.00, Sept. 2017

2020. We do not express any preference by the order of the packages but present them in alphabetic order. For each of the packages we give a reference that either is specifically devoted to that software package or is a reference using that package. We specify the corresponding (main) programming languages, not excluding that some of the packages provide bindings or some optimized routines for some other language. Most of the packages are MATLAB-based, some using as little MATLAB-specific functionality that they are also executable from Octave. The packages devoted to MATLAB or Python do mainly not have restrictions on the operating system, as long as MATLAB or Python is installed in corresponding versions. Only the package versions providing C/C++ versions typically are limited to either Windows or Linux operating systems. About half of the packages specify some open-source GNU- or BSD-type license. The column "latest version" lists version numbers and release dates if provided by the supporting websites. Some packages do not provide access by download but only on request, in particular packages that are used for simulation of multibody systems (Chapter 2 in this volume). The recent release dates indicate that most of the packages are under active development and we recommend to consult the corresponding webpages and github sites for most recent information. Especially the packages maintained at github frequently have most recent commits that yield functional versions more recent than the latest release tags given in the table. In particular we want to emphasize that the following detailed descriptions may relate to more recent git-commits than the versions mentioned in the table. In the context of MOR software, we want also to refer to the excellent software list at the MOR Wiki.[12] Note that, by nature, our package list has a considerable overlap to that online reference list.

12 http://www.modelreduction.org

We now give some more details on the packages, including nonabbreviated name, open-source policy, online availability, license type, download (or project) URL, programming language, installation procedure, documentation, main reduction methods, application fields, and further individual comments.

DPA:[13] A collection of MATLAB algorithms related to versions of the *Dominant Pole Algorithm* [33] are provided for download as source code without license restrictions. No installation steps are required, and operation is performed via simple execution of the *.m files. Documentation is available as comments in the program files. Reduction by those methods is related to modal reduction with connections to moment-matching (Chapter 3 in Volume 1 of *Model order reduction*). The algorithms allow reduction of first- and second-order single-input, single-output systems and first-order multiple-input, multiple-output systems. The motivating field of application is power system electronics, but the code can be used for reduction and modal analysis of dynamical systems from other domains as well, including electronics (RLC parasitics), acoustics, and mechanics. Quite a number of test systems and (large-scale) system matrices are provided.

emgr:[14] The MATLAB/Octave package provides an *Empirical Gramian Framework* [18] for model reduction of nonlinear input–output systems. The program files are available for download under the open-source BSD-2-clause license. No installation steps are required, as the single MATLAB file can directly be executed without dependency on further packages. The empirical Gramians basically extend the concept of system Gramians for first-order LTI systems (Chapter 2 in Volume 1 of *Model order reduction*) to nonlinear systems. Overall, these reduction techniques can be related to BT. For parametric problems with high-dimensional parameter spaces, empirical Gramians also allow combined reduction of parameter and system order. Due to its generality, there is no restriction to the field of application: Models from neural science, mechanical systems, electrical networks, or discretized PDEs are contained as benchmark systems on the website. Extensive documentation of the programs is also provided online.

ITHACA:[15] This C++ package on *In real Time Highly Advanced Computational Applications* comes in several versions: ITHACA-FV with finite volume full-order solver https://mathlab.sissa.it/ITHACA-FV [36, 37] based on OpenFOAM, ITHACA-SEM with spectral element detailed solver https://mathlab.sissa.it/ITHACA-SEM based on Nektar++ and ITHACA-DG https://mathlab.sissa.it/ITHACA-DG based on discontinuous Galerkin full-order solver based on HopeFOAM. All versions are open-source, the former under a LGPL license, the latter two under an MIT License. The packages' code and documentation are available on corresponding

13 https://sites.google.com/site/rommes/software

14 https://gramian.de

15 http://mathlab.sissa.it/

github pages. The packages ITHACA-SEM and ITHACA-DG are at an early development stage, thus we focus on the more mature ITHACA-FV package in the following: The installation of ITHACA-FV requires an existing implementation of OpenFOAM 5.0, 6.0, or 1812, and then the cloning and compilation of the package are realized with few commands. The package provides several well-documented tutorial examples. The documentation is extracted from the code by doxygen. The package provides the implementation of several reduced-order modeling techniques for parameterized problems. In particular, the thermal block, steady and unsteady Navier–Stokes, additionally coupling with an energy equation, and the Boussinesq equation are contained. As reduction techniques, POD, nonintrusive POD-interpolation, DEIM (Chapter 5 in Volume 2 of *Model order reduction*), and DMD (Chapter 7 in Volume 2 of *Model order reduction*) are provided.

KerMor:[16] The MATLAB package provides *Kernel and MOR methods* for surrogate modeling of nonlinear systems [39]. It is open-source partly under the GNU GPL 3 and BSD license. The source files are maintained and are accessible freely at a corresponding github repository. The package is using sophisticated object-oriented features of MATLAB, and hence is not suitable to be used under Octave. Program documentation is provided online but can as well be generated offline by the `mtoc++` and `doxygen` documentation tools. After download or cloning of the package, some installation steps are required for compiling suitable mex functions and setting environment variables. Then a single startup file needs to be executed in MATLAB in order to use the package. The considered model classes are simple IO function maps, LTI and parametric nonlinear systems. As surrogate modeling techniques, mainly projection-based methods (POD-DEIM) (Chapter 5 in Volume 2 of *Model order reduction*) and kernel methods (VKOGA, SVR) (Chapter 9 in Volume 1 of *Model order reduction*) are provided. The field of application is not limited, demo examples contain electric circuit as well as system-biological (programmed cell death) models, discretized PDEs (Burgers), or biomechanics (nonlinear elasticity for muscle models).

M.E.S.S.:[17] The *Matrix Equation Sparse Solver* library [34] provides algorithms for approximate matrix equation solving such as large-scale Lyapunov or (differential) Riccati equations. Since the solution of such matrix equations represents the core of many algorithms in model reduction and control, this package is essential for reduction of large-scale problems. The package consists of a version for MATLAB/Octave and a version for C; additionally it provides Python bindings. The code is accessible via a public git repository and covered by a GNU GPL 2 license with some exceptions. The MOR functions comprise BT and the iterative rational Krylov algorithm (IRKA) (Chapter 3 in Volume 1 of *Model order reduction*)

16 https://www.morepas.org/software/kermor/index.html

17 http://www.mpi-magdeburg.mpg.de/projects/mess

for first-order LTI systems. Documentation is provided within the MATLAB source files. Installation is straightforward by simple unpacking and a startup script for initialization at each MATLAB session. Due to its generality, model examples both contain discretized transport PDEs (heat equation, advection-diffusion) and mass-spring-damper systems for structure-preserving second-order techniques. Further demonstration examples explain the extension to structured DAE systems that allow for implicit index reduction.

MOREMBS:[18] This package on *Model order reduction for elastic multibody systems* [12] consists of both a MATLAB version (MatMorembs) and a C++ version (Morembs++). The package is not available for download, but can be provided on request for academic use. The supported reduction techniques for second-order systems comprise modal techniques such as the Craig–Bampton method for component mode synthesis (CMS), as well as Krylov subspace techniques (Chapter 3 in Volume 1 of *Model order reduction*) or SVD/Gramian-based techniques (Chapter 2 in Volume 1 of *Model order reduction*). The field of application is clearly focused on elastic mechanical systems. The strength of the package lies in a multitude of coupling options, in particular importing system matrices from various commercial finite element programs (ABAQUS, ANSYS, PERMAS, Nastran) and exporting reduced system descriptions to multibody simulator programs (MATLAB Simulink, SIMPACK, Neweul-M^2, Adams, LMS).

MORLAB:[19] This *Model Order Reduction LABoratory* package [5] is aiming for spectral-projection-based model reduction of dynamical systems. It is freely available for download as a MATLAB toolbox, Octave package, or zip archive. The package is subject to the GNU Affero General Public License 3. Installation is simple by executing a startup file for adding paths or by using the automatic mechanisms for MATLAB toolboxes and Octave packages.

There is an extensive HTML documentation included in the package. The package aims at dense first-order LTI, descriptor, or second-order systems. The spectrum of methods is based on the solution of matrix equations [29], in particular modal truncation, BT with many variants (frequency-limited BT, time-limited BT, bounded-real BT, positive-real BT, balanced stochastic truncation, linear quadratic-Gaussian BT, and $\mathcal{H}_\infty$-BT), as well as a variant of the Hankel norm approximation. Due to its generality of systems, the package does not focus on special application fields.

MORPACK:[20] This *Model Order Reduction PACKage* is a MATLAB library for reducing elastic multibody systems [26, 24]. The package is not available for download but test versions for academic purposes can be provided on request. The appli-

18 http://www.itm.uni-stuttgart.de/research/morembs/MOREMBS_en.php

19 http://www.mpi-magdeburg.mpg.de/projects/morlab

20 https://tu-dresden.de/ing/maschinenwesen/ifkm/dmt/forschung/projekte/morpack

cation field is limited to second-order mechanical systems for elastic multibody dynamics. The package acts as a general interface between finite element software (ANSYS, ABAQUS, NASTRAN, LS-DYNA) and the multibody simulators (SIMPACK, ANSYS, EMBS-Matlab). Therefore, validation of the reduction and choice of the master degrees of freedom have a high priority. Over 60 correlation criteria are available to compare reduced-order models against the original model as well as measurement data. Likewise, there are many methods to select optimal master degrees of freedom. The reduction methods that are available comprise Guyan, CMS, Krylov subspace methods, (second-order) BT, and minimal model generation by mode truncation. Also multistep reduction processes can be performed. The package is steered by a graphical user interface. Several benchmark models are provided ranging from 100,000 to 1,200,000 degrees of freedom.

pyMOR:[21] The *Model Order Reduction with Python* package is an open-source library under active development [27]. It is accessible by a repository at github under a (modified) BSD-2-Clause license. Installation basically works via `pip`, and detailed installation instructions are given on the website. Extensive program documentation is provided online and within the program source code. The package aims at covering all types of reduction techniques, ranging from RB methods for parameterized PDEs up to MOR for control systems, e. g., BT or Krylov subspace methods. Also general nonlinearities can be treated by empirical interpolation (Chapters 1 and 5 in Volume 2 of *Model order reduction*). Due to its generality, no application fields are excluded. By using abstract interfaces, coupling of external high-fidelity solvers is possible and several of such dockers exist, e. g., for Dune, FEniCS, deal.II, or NGSolve. Also, some finite element and finite volume discretizations are included based on NumPy/SciPy. Two jupyter notebooks are provided for interactive exploration of corresponding models (heat equation, spring) and reduction techniques. Many demo applications are contained within the package such as PDE-based models (Burgers equation or elliptic and parabolic equations).

RBmatlab:[22] The *Reduced Basis Matlab* package is an open-source library for numerical approximation of parameterized problems. The code is publicly available for download via the Model Reduction of Parametrized Systems (MoRePaS) website without license restrictions. The master branch is maintained as a git repository, for which access can be granted on request. Documentation is provided online and within the MATLAB function headers. This documentation can be generated offline by the `mtoc++` and `doxygen` tools. Installation is simple by unzipping, setting two environment variables, and optionally extending the MATLAB startup script to get RBmatlab started automatically during the initialization of each

21 https://github.com/pymor/pymor

22 https://www.morepas.org/software/rbmatlab

MATLAB session. As system types parametric PDEs (elliptic, parabolic, hyperbolic) mostly motivated by transport problems (heat equation, Burgers equation, two-phase flow), mechanics (elasticity), or finance (variational inequalities) are supported, as well as parametric control systems, e. g., the chemical master equation. Snapshot-based reduction methods (POD, greedy, POD-greedy) are implemented, including certification by error estimators. Coupling to the scientific computing packages Dune, Alberta, and COMSOL is realized. Additionally, the package contains PDE discretization techniques (finite element method, finite volume method, discontinuous Galerkin) to be used as high-fidelity solvers for reduction procedures. Use of those reduced models in parameter optimization and feedback control are some of the implemented multiquery settings. Many demos are implemented and can be interactively accessed for getting insight into the functionality of the package. A pedagogical model of the well-known thermal block model is provided as tutorial example [14].

RBniCS:[23] This package on *Reduced Order Modelling in FEniCS* is understood to be accompanying the book [17]. Installation prerequisites are the availability of FEniCS (with PETSc, SLEPc, petsc4py, and slepc4py), numpy, and scipy. The remaining installation of RBniCS is then easily done by cloning the git repository and requesting python3 to install the package. As model classes the package comprises parametric elliptic and parabolic problems. Both linear and nonlinear problems (using empirical interpolation) are considered. Advection-diffusion as well as Stokes and Navier–Stokes problems are readily available. The code uses clear naming, and is hence sufficiently comprehensive. A documentation can be generated but is not available online. As basis generation procedures POD, greedy, and Gram–Schmidt algorithms are realized. The successive constraint method for rapid computation of stability factor lower bounds is implemented. Almost 20 tutorials are available and suitable to be used in model reduction courses. A particular feature of this package is the ease of specifying new problems, i. e., not only changing coefficient functions but also specifying and changing the differential operators of the PDE by high-level FEniCS commands.

SparseRC:[24] SparseRC [21] is a collection of MATLAB routines which performs partitioning/reordering-based model reduction for RC netlists with nodes up to hundreds of thousands, and terminals up to tens of thousands. The motivating field of application is analog circuit design, where parasitic extraction of the physical layout may result in large-scale networks of resistors (R) and capacitors (C). These networks can be modeled by dynamical systems and SparseRC can be used to reduce these systems, exploiting and preserving properties specific to such networks.

23 https://mathlab.sissa.it/rbnics

24 https://sites.google.com/site/rionutiu2/research/software

sssMOR:[25] The *Sparse State-Space and Model Order Reduction Toolbox* [6] and the extension `psssMOR`[26] for parametric problems are MATLAB libraries distributed under the BSD-2-Clause license. The code is accessible via a git repository at github. Installation is realized by unzipping the packages and executing a few installation commands as specified on the webpage. The MATLAB functions are very well documented, and hence accessible by the MATLAB help functionality. The particular motivation of those packages is extending the MATLAB-inherent state-space toolbox, which is restricted to dense matrices and thus only allows treatment of moderately sized problems. The `sss` toolbox provides this functionality using sparse matrices, and hence can treat considerably larger system orders. The `sssMOR` library then implements MOR techniques using those sparse system representations. The system types considered are LTI control systems. Basic reduction techniques implemented are modal truncation, BT, and rational Krylov subspace methods. At the same time, it provides the IRKA as well as some more recent algorithms such as the CUmulative REduction framework (CURE), the Stability-Preserving, Adaptive Rational Krylov algorithm (SPARK), and the confined IRKA (CIRKA). Some model samples from well-known benchmark collections (CD player, building, gyro) are provided. In principle, the application scope is not limited as long as the systems can be cast as (parametric) LTI control systems. The toolboxes are currently being extended to cope with nonlinear sparse state-space systems, such as bilinear (bsssMOR) and quadratic-bilinear (qbsssMOR) models. Extensions for other system classes, e. g., port-Hamiltonian (spHMOR) and second-order (ssoMOR) systems, are naturally also conceivable under the same guiding principle.

We do not claim completeness of this above package list, as many researchers have their private code collection, libraries, or repositories. But the list covers the main currently available software packages that we are aware of. Several smaller packages exist, such as `pydmd`[27] on DMD and `ezyrb`[28] on POD with interpolation. Some academic PDE discretization packages also include MOR functionality, e. g., `libmesh`,[29] which has RB capabilities via `rbOOmit` [23], or `Feel++`,[30] which also provides MOR methods. Further packages exist which however are no longer under active development and not provided by download. Among those we want to mention `dune-rb`, whose capabilities and principles of coupling with RBmatlab have been explained in [9]. The thesis [1] describes the extension to localized model reduction approaches. Also the package

25 https://www.rt.mw.tum.de/?sssmor
26 https://www.rt.mw.tum.de/?psssmor
27 https://mathlab.sissa.it/pydmd
28 http://mathlab.sissa.it/ezyrb
29 http://libmesh.github.io/
30 http://www.feelpp.org

rbMIT is a software package that provides the most elementary RB algorithms. This software package was awarded with the Springer Computational Science and Engineering Prize in 2009. The package is currently not available by a website, but can be accessed via the internet archive.[31] Similarly the package PABTEC on BT for electronics applications is listed at the swMATH portal[32] but seems no longer to be publicly accessible. The academic MORE package[33] [32] is a precursor of the commercial MOR toolbox mentioned in the previous subsection.

Disadvantages of academic packages (which more or less applies to the different packages mentioned above) must certainly also be stated. The level of documentation of those code packages may be incomplete, development of packages may be discontinued, and support can typically not be offered in an extensive manner or instantaneously. So using these packages typically requires some self-study, reproducing of running models and reduction techniques, or even some reverse engineering, or trial and error with changing parameters.

13.4 Conclusions and recommendations

We argued that online-efficient and accurate MOR algorithms require access to data, functionality, or other internals of the full solver. In particular, in many cases, a special design, decomposition or structure of the full-order model needs to be established in order to optimally apply corresponding MOR techniques. Elementary routines from the full solver or high-fidelity simulation package that are required for white-box MOR may comprise the following: For snapshot-based MOR algorithms, e. g., POD, greedy, and POD-greedy, a triggering of a full-order simulation with specified input parameters and return of state snapshots must be available. Information about nodal interpretation of the state vector is required. In the simplest case of linear finite element or finite difference discretization this is equivalent to enumeration of the mesh nodes. In general, a routine for assembly or matrix–vector multiplication with the inner product (mass) matrix is very helpful. This enables computing L^2-norms, orthonormalization, and projections. Similarly, a subroutine for assembly or matrix–vector multiplication with the stiffness matrix, which enables computing Sobolev H^1-semi-norms, is helpful. Export of other system matrices, e. g., Jacobian matrices of nonlinear terms, may be required. For parametric problems an export of parameter separable decompositions is essential; this means access to (nonparametric) system matrix and vector components and parametric coefficient functions. For sampling-based methods for

31 https://web.archive.org/web/20171110212818/http://augustine.mit.edu:80/methodology/methodology_rbMIT_System.htm

32 https://swmath.org/software/4061

33 https://w3.onera.fr/more

nonlinear problems, local evaluation of system nonlinearities based on local reconstruction of the state vector needs to be provided. Optionally, visualization or postprocessing of a given state vector by the full solver is useful. If those functionalities are not provided by the full solver, either a loss of online efficiency is unavoidable, or some gray-box or black-box workarounds or hacks might be possible as we have exemplified for general stationary and unsteady, linear, and nonlinear problems.

A multitude of software packages from groups from academia have been developed. Typically those packages not only provide the latest MOR technology, but also implementations of some full system solvers. Only by control of all implementational details of the full models, which are typically code-intrusive, optimal online efficiency can be realized. This is mostly not possible by full-order models from commercial packages. This full control of the high-fidelity models enables independence of scientists from external simulator packages, but at the same time is very time-intensive, distracting one from main disciplinary research.

We want to close with some recommendations. Addressing commercial software developers, we think that simulation packages will not remain competitive if MOR technology is not included as essential enabler for modern higher-level simulation tasks such as uncertainty quantification, parametric studies, optimization, design, etc. Simulation software engineers should be aware that their solver is no longer the last element in the simulation analysis pipeline but those are embedded in more complex simulation tasks. Inclusion of MOR algorithms in simulator packages can thus be a competitive advantage. When planning to include MOR technology, one should be clearly aware that access to more internals than just the solution/state degree of freedom vector is required. Access to system matrices, components, local nonlinearity evaluations, local subgrid geometry, etc., can be useful or required for modern and efficient MOR algorithms as listed above. Creating suitable interfaces or other means of access to those internals will enable applying efficient MOR algorithms. Apart from exporting system parameters, matrices, or geometry information, also importing facilities for effective bases, error estimation routines, etc., would be recommended for expanding the scope of applicability of commercial simulation packages. In particular, by realizing error estimation techniques, a certification and therefore guarantee of reliability of methods/packages is obtained. The last decade has enabled hundreds of PhD students to specialize and graduate in the field of MOR. These would be excellent candidates for transfer of those technologies into business and industry. A further option for realizing MOR algorithms is the foundation of public–private partnerships or joint programs, e. g., European Industrial Training Networks, etc.

Some recommendations for academic MOR researchers might be to develop further "hacks" of industrial software, i. e., using solvers as black-box or gray-box leading to new algorithms and analysis. For reproducibility of results, we strongly encourage to provide open-source and online accessible program code. If possible, it is advisable to contribute to existing MOR libraries instead of reinventing the wheel by developing new packages from scratch. If it is required or desired to reinvent MOR code or solvers,

practitioners or simulation engineers should think of useful central interfaces, which will ease exchange and coupling of the code to other packages. This is particularly relevant for bridging different programming platforms such as C++, MATLAB, Python, Fortran, etc.

A general wish for the development in computational sciences is the increased acknowledgment of and respect for the effort in development, maintenance, and documentation of software packages. Typically, applications and computational approaches are so involved, that software packages cannot be set up from scratch by scientists within the typical scientific "lifetime," e. g., a PhD thesis. In particular, existing open-source packages directly enable a high entrance level for further developments. Software development thus is crucial and one core scientific service for the community. Scientists and students who develop, maintain, and provide well-designed software should more easily obtain scientific credits by accepting such results as major scientific results in theses as well as being able to publish journal articles on such software – in contrast to the current state of expecting such developments as minor by-product of disciplinary scientific contributions. But also in funding agencies such major achievements or proposals on scientific software development should be accepted as foundation for disciplinary progress. Only slowly this awareness in the scientific communities is developing, e. g., reflected by the recent Software branch at the SIAM journal on Scientific Computing that now also enables to publish journal articles on software. But certainly this general awareness can be further strengthened by each individual researcher.

Bibliography

[1] S. Alebrand, *Efficient Schemes for Parametrized Multiscale Problems*. PhD thesis, University of Stuttgart, 2015.

[2] A. C. Antoulas, *Approximation of Large–Scale Dynamical Systems*, SIAM Publications, Philadelphia, PA, 2005.

[3] T. Bechtold, E. B. Rudnyi, and J. Korvink, Selected model reduction software, in *Fast Simulation of Electro-Thermal MEMS: Efficient Dynamic Compact Models*, Springer, 2007.

[4] P. Benner, M. Ohlberger, A. Cohen, and K. Willcox (eds.), *Model Reduction and Approximation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

[5] P. Benner and S. W. R. Werner, MORLAB – Model Order Reduction LABoratory (version 4.0), December 2018. See also: http://www.mpi-magdeburg.mpg.de/projects/morlab.

[6] A. Castagnotto, M. Cruz Varona, L. Jeschek, and B. Lohmann, sss & sssMOR: analysis and reduction of large-scale dynamical systems in MATLAB, *Automatisierungstechnik*, **65** (2) (2017), 134–150.

[7] R. Chakir and Y. Maday, A two-grid finite-element/reduced basis scheme for the approximation of the solution of parameter dependent PDE, in *9e Colloque National en Calcul des Structures*, 2009.

[8] S. Chaturantabut and D. Sorensen, Nonlinear model reduction via discrete empirical interpolation, *SIAM J. Sci. Comput.*, **32** (5) (2010), 2737–2764.

[9] M. Drohmann, B. Haasdonk, and M. Ohlberger, A software framework for reduced basis methods using DUNE-RB and RBMATLAB, in A. Dedner, B. Flemisch, and R. Klöfkorn (eds.), *Advances in DUNE: Proceedings of the DUNE User Meeting, Held in October 6th–8th 2010 in Stuttgart*, Springer, Germany, 2012.

[10] M. Drohmann, B. Haasdonk, and M. Ohlberger, Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation, *SIAM J. Sci. Comput.*, **34** (2) (2012), A937–A969.

[11] R. Everson and L. Sirovich, Karhunen–Loeve procedure for gappy data, *J. Opt. Soc. Am. A*, **12** (1995), 1657–1664.

[12] J. Fehr, D. Grunert, P. Holzwarth, B. Fröhlich, N. Walker, and P. Eberhard, MOREMBS – a model order reduction package for elastic multibody systems and beyond, in *Reduced-Order Modeling (ROM) for Simulation and Optimization*, pp. 141–166, Springer, 2018.

[13] M. Geuss, *A Black-Box Method for Parametric Model Order Reduction based on Matrix Interpolation with Application to Simulation and Control*. PhD thesis, Technische Universität München, 2015.

[14] B. Haasdonk, Reduced basis methods for parametrized PDEs – a tutorial introduction for stationary and instationary problems, in P. Benner, A. Cohen, M. Ohlberger, and K. Willcox (eds.), *Model Reduction and Approximation: Theory and Algorithms*, pp. 65–136, SIAM, Philadelphia, 2017.

[15] B. Haasdonk and M. Ohlberger, Reduced basis method for explicit finite volume approximations of nonlinear conservation laws, in *Hyperbolic Problems: Theory, Numerics and Applications*. Proc. Sympos. Appl. Math., vol. 67, pp. 605–614, Amer. Math. Soc., Providence, RI, 2009.

[16] B. Haasdonk, M. Ohlberger, and G. Rozza, A reduced basis method for evolution schemes with parameter-dependent explicit operators, *Electron. Trans. Numer. Anal.*, **32** (2008), 145–161.

[17] J. S. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, SpringerBriefs in Mathematics, Springer International Publishing, 2015.

[18] C. Himpe, emgr – The empirical gramian framework, *Algorithms*, **11** (7) (2018), 91.

[19] D. B. P. Huynh, D. J. Knezevic, and A. T. Patera, A static condensation reduced basis element method: complex problems, *Comput. Methods Appl. Mech. Eng.*, **259** (2013), 197–216.

[20] A. Ionita and A. Antoulas, Data-driven parametrized model reduction in the Loewner framework, *SIAM J. Sci. Comput.*, **36** (3) (2014), A984–A1007.

[21] R. Ionutiu, J. Rommes, and W. H. A. Schilders, SparseRC: sparsity preserving model reduction for RC circuits with many terminals, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, **30** (12) (2011), 1828–1841.

[22] L. Kazaz, Black box model order reduction of nonlinear systems with kernel and discrete empirical interpolation. Bachelor's thesis, University of Stuttgart, 2014.

[23] D. J. Knezevic and J. W. Peterson, A high-performance parallel implementation of the certified reduced basis method, *Comput. Methods Appl. Mech. Eng.*, **200** (13–16) (2011), 1455–1466.

[24] P. Koutsovasilis and M. Beitelschmidt, MORPACK toolbox for coupling rigid and elastic multi-body dynamics, in *Proc. of NAFEMS World Congress*, 2009.

[25] N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*, SIAM, 2016.

[26] C. Lein and M. Beitelschmidt, MORPACK-Schnittstelle zum Import von FE-Strukturen nach SIMPACK, *Automatisierungstechnik*, **60** (9) (2012), 547–559.

[27] R. Milk, S. Rave, and F. Schindler, pyMOR – generic algorithms and interfaces for model order reduction, *SIAM J. Sci. Comput.*, **38** (5) (2016), S194–S216.

[28] F. Negri, A. Manzoni, and D. Amsallem, Efficient model reduction of parametrized systems by matrix discrete empirical interpolation, *J. Comput. Phys.*, **303** (2015), 431–454.
[29] S. W. R. Werner and P. Benner, Model reduction of descriptor systems with the MORLAB toolbox, in *Proc. 9th Vienna International Conference on Mathematical Modelling MATHMOD 2018*, IFAC-PapersOnLine, vol. 51, pp. 547–552, 2018.
[30] A. T. Patera and G. Rozza, *Reduced Basis Approximation and a Posteriori Error Estimation for Parametrized Partial Differential Equations*, to appear in (tentative) MIT Pappalardo Graduate Monographs in Mechanical Engineering, MIT, 2007.
[31] B. Peherstorfer and K. Willcox, Data-driven operator inference for nonintrusive projection-based model reduction, *Comput. Methods Appl. Mech. Eng.*, **306** (2016), 196–215.
[32] C. Poussot-Vassal and P. Vuillemin, Introduction to MORE: a MOdel REduction toolbox, in *Proc. IEEE International Conference on Control Applications*, pp. 776–781, 2012.
[33] J. Rommes and N. Martins, Efficient computation of transfer function dominant poles using subspace acceleration, *IEEE Trans. Power Syst.*, **21** (3) (2006), 1218–1226.
[34] J. Saak, M. Köhler, and P. Benner, M-M.E.S.S.-1.0.1 – the matrix equations sparse solvers library. DOI:10.5281/zenodo.50575, April 2016. See also: www.mpi-magdeburg.mpg.de/projects/mess.
[35] W. H. A. Schilders, H. A. Van der Vorst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, vol. 13, Springer, 2008.
[36] G. Stabile, S. Hijazi, A. Mola, S. Lorenzi, and G. Rozza, POD-Galerkin reduced order methods for CFD using finite volume discretisation: vortex shedding around a circular cylinder, *Commun. Appl. Ind. Math.*, **8** (1) (2017), 210–236.
[37] G. Stabile and G. Rozza, Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier-Stokes equations, *Comput. Fluids*, (2018).
[38] Q. Wang, J. S. Hesthaven, and D. Ray, Non-intrusive reduced order modelling of unsteady flows using artificial neural networks with application to a combustion problem, *J. Comput. Phys.*, **384** (2019), 289–307.
[39] D. Wirtz, *Model Reduction for Nonlinear Systems: Kernel Methods and Error Estimation*. PhD Thesis, University of Stuttgart, October 2013.
[40] D. Wirtz, D. C. Sorensen, and B. Haasdonk, A posteriori error estimation for DEIM reduced nonlinear dynamical systems, *SIAM J. Sci. Comput.*, **36** (2) (2014), A311–A338.
[41] O. Zeeb, *A numerical framework for semi-automated Reduced Basis Methods with blackbox solvers*. PhD thesis, University of Ulm, 2015.