SIAM J. SCI. COMPUT.
Vol. 37, No. 4, pp. A2123–A2150

ONLINE ADAPTIVE MODEL REDUCTION FOR NONLINEAR SYSTEMS VIA LOW-RANK UPDATES*

BENJAMIN PEHERSTORFER† AND KAREN WILLCOX†

Abstract. This work presents a nonlinear model reduction approach for systems of equations stemming from the discretization of partial differential equations with nonlinear terms. Our approach constructs a reduced system with proper orthogonal decomposition and the discrete empirical interpolation method (DEIM); however, whereas classical DEIM derives a linear approximation of the nonlinear terms in a static DEIM space generated in an offline phase, our method adapts the DEIM space as the online calculation proceeds and thus provides a nonlinear approximation. The online adaptation uses new data to produce a reduced system that accurately approximates behavior not anticipated in the offline phase. These online data are obtained by querying the full-order system during the online phase, but only at a few selected components to guarantee a computationally efficient adaptation. Compared to the classical static approach, our online adaptive and nonlinear model reduction approach achieves accuracy improvements of up to three orders of magnitude in our numerical experiments with time-dependent and steady-state nonlinear problems. The examples also demonstrate that through adaptivity, our reduced systems provide valid approximations of the full-order systems outside of the parameter domains for which they were initially built in the offline phase.

Key words. adaptive model reduction, nonlinear systems, empirical interpolation, proper orthogonal decomposition

AMS subject classifications. 65M22, 65N22

DOI. 10.1137/140989169

1. Introduction.

Model reduction derives reduced systems of large-scale systems of equations, typically using an offline phase in which the reduced system is constructed from solutions of the full-order system, and an online phase in which the reduced system is executed repeatedly to generate solutions for the task at hand. In many situations, the reduced systems yield accurate approximations of the full-order solutions but with orders of magnitude reduction in computational complexity. Model reduction exploits that often the solutions are not scattered all over the high-dimensional solution space, but instead they form a low-dimensional manifold that can be approximated by a low-dimensional (linear) reduced space. In some cases, however, the manifold exhibits a complex and nonlinear structure that can only be approximated well by the linear reduced space if its dimension is chosen sufficiently high. Thus, solving the reduced system can become computationally expensive. We therefore propose a nonlinear approximation of the manifold. The nonlinear approximation is based on adapting the reduced system while the computation proceeds in the online phase, using newly generated data through limited queries to the full-order system at a few selected components. Our online adaptation leads to a reduced system that can more efficiently capture nonlinear structure in the manifold, it ensures computational efficiency by performing low-rank updates, and through the use of new

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 29, 2014; accepted for publication (in revised form) June 9, 2015; published electronically August 18, 2015. This work was partially supported by AFOSR grant FA9550-11-1-0339 under the Dynamic Data-Driven Application Systems (DDDAS) Program.
http://www.siam.org/journals/sisc/37-4/98916.html
†Department of Aeronautics & Astronautics, MIT, Cambridge, MA 02139 (pehersto@mit.edu, kwillcox@mit.edu).

data it avoids relying on precomputed quantities that restrict the adaptation to those situations that were anticipated in the offline phase.

We focus on systems of equations stemming from the discretization of nonlinear partial differential equations (PDEs). Projection-based model reduction employs Galerkin or Petrov–Galerkin projection of the equations onto a low-dimensional reduced space that is spanned by a set of basis vectors. Proper orthogonal decomposition (POD) is one popular method to construct such a set of basis vectors [41]. Other methods include truncated balanced realization [33] and Krylov subspace methods [21, 23]. In the case of nonlinear systems, however, projection alone is not sufficient to obtain a computationally efficient method, because then the nonlinear terms of the PDE entail computations that often render solving the reduced system almost as expensive as solving the full-order system. One solution to this problem is to approximate the nonlinear terms with sparse sampling methods. Sparse sampling methods sample the nonlinear terms at a few components and then approximately represent them in a low-dimensional space. In [3], the approximation is derived via gappy POD. The Gauss–Newton with approximated tensors (GNAT) method [10] approximates the nonlinear terms in the low-dimensional space by solving a low-cost least-squares problem. We consider here the discrete empirical interpolation method (DEIM) [12], which is the discrete counterpart of the empirical interpolation method [4]. It samples the nonlinear terms at previously selected DEIM interpolation points and then combines interpolation and projection to derive an approximation in a low-dimensional DEIM space. The approximation quality and the costs of the DEIM interpolant directly influence the overall quality and costs of the reduced system. We therefore propose to adapt this DEIM interpolant online to better capture the nonlinear structure of the manifold induced by the solutions of the nonlinear system.

Adaptivity has attracted much attention in the context of model reduction. Offline adaptive methods extend [42, 26] or weight [14, 15] snapshot data while the reduced system is constructed in the offline phase; however, once the reduced system is generated, it stays fixed and is kept unchanged online. Online adaptive methods change the reduced system during the computations in the online phase. Most of the existing online adaptivity approaches rely on precomputed quantities that restrict the way the reduced system can be updated online. They do not incorporate new data that become available online and thus must anticipate offline how the reduced system might change. Interpolation between reduced systems [1, 17, 34, 46], localization approaches [20, 18, 2, 36, 19], and dictionary approaches [27, 31] fall into this category of online adaptive methods.

In contrast, we consider here online adaptivity that does not solely rely on precomputed quantities but incorporates new data online and thus allows changes to the reduced system that were not anticipated offline. There are several approaches that incorporate new data by rebuilding the reduced system [16, 32, 39, 44, 38, 35, 45, 29]; however, even if an incremental basis generation or an h-refinement of the basis [9] is employed, assembling the reduced system with the newly generated basis often still entails expensive computations online. An online adaptive and localized approach that takes new data into account efficiently was presented in [43]. To increase accuracy and stability, a reference state is subtracted from the snapshots corresponding to localized reduced systems in the online phase. This adaptivity approach incorporates the reference state as new data online, but it is a limited form of adaptivity because each snapshot receives the same update.

We develop an online adaptivity approach that adapts the DEIM space and the DEIM interpolation points with additive low-rank updates and thus allows more com-

plex updates, including translations and rotations of the DEIM space. We sample the nonlinear terms at more points than specified by DEIM to obtain a nonzero residual at the sampling points. From this residual, we derive low-rank updates to the basis of the DEIM space and to the DEIM interpolation points. This introduces online computational costs that scale linearly in the number of degrees of freedom of the full-order system, but it allows the adaptation of the DEIM approximation while the computation proceeds in the online phase. To avoid the update being limited by precomputed quantities, our method queries the full-order system during the online phase; however, to achieve a computationally efficient adaptation, we query the full-order system at a few components only. Thus, our online adaptivity approach explicitly breaks with the classical offline/online splitting of model reduction and allows online costs that scale linearly in the number of degrees of freedom of the full-order system.

Section 2 briefly summarizes model reduction for nonlinear systems. It then motivates online adaptive model reduction with a synthetic optimization problem and gives a detailed problem formulation. The DEIM basis and the DEIM interpolation points adaptivity procedures follow in sections 3 and 4, respectively. The numerical results in section 5 demonstrate reduced systems based on our online adaptive DEIM approximations on parametrized and time-dependent nonlinear systems. Conclusions are drawn in section 6.

2. Model reduction for nonlinear systems. We briefly discuss model reduction for nonlinear systems. A reduced system with POD and DEIM is derived in sections 2.1 and 2.2, respectively. Sections 2.3 and 2.4 demonstrate on a synthetic optimization problem that the approximation quality of the reduced system can be significantly improved by incorporating data that become available online but that the classical model reduction procedures do not allow a computationally efficient modification of the reduced system in the online phase.

2.1. Proper orthogonal decomposition. We consider the discrete system of nonlinear equations

$$\boldsymbol{A}\boldsymbol{y}(\boldsymbol{\mu}) + \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) = 0 \tag{2.1}$$

stemming from the discretization of a nonlinear PDE depending on the parameter $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_d]^T \in \mathcal{D}$ with parameter domain $\mathcal{D} \subset \mathbb{R}^d$. The solution or state vector $\boldsymbol{y}(\boldsymbol{\mu}) = [y_1(\boldsymbol{\mu}), \ldots, y_N(\boldsymbol{\mu})]^T \in \mathbb{R}^N$ is an N-dimensional vector. We choose the linear operator $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ and the nonlinear function $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^N$ such that they correspond to the linear and the nonlinear terms of the PDE, respectively. We consider here the case where the function $\boldsymbol{f}$ is evaluated componentwise at the state vector $\boldsymbol{y}(\boldsymbol{\mu})$, i.e., $\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) = [f_1(y_1(\boldsymbol{\mu})), \ldots, f_N(y_N(\boldsymbol{\mu}))]^T \in \mathbb{R}^N$, with the nonlinear functions $f_1, \ldots, f_N : \mathbb{R} \to \mathbb{R}, y \mapsto f(y)$. The Jacobian of (2.1) is $\boldsymbol{J}(\boldsymbol{\mu}) = \boldsymbol{A} + \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{y}(\boldsymbol{\mu}))$ with

$$\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{y}(\boldsymbol{\mu})) = \operatorname{diag}(f_1'(y_1(\boldsymbol{\mu})), \ldots, f_N'(y_N(\boldsymbol{\mu})))$$

and the first derivatives $f_1', \ldots, f_N'$ of $f_1, \ldots, f_N$ with respect to y. Note that the following DEIM adaptivity scheme can be extended to nonlinear functions $\boldsymbol{f}$ with component functions $f_1, \ldots, f_N$ that depend on multiple components of the state vector with the approach discussed for DEIM in [12, section 3.5]. Note further that (2.1) is a steady-state system but that all of the following discussion is applicable also to time-dependent problems. We also note that we assume well-posedness of (2.1).

We derive a reduced system of the full-order system (2.1) by computing a reduced basis with POD. Let $\boldsymbol{Y} = [\boldsymbol{y}(\boldsymbol{\mu}_1), \ldots, \boldsymbol{y}(\boldsymbol{\mu}_M)] \in \mathbb{R}^{N\times M}$ be the snapshot matrix. Its columns are the $M \in \mathbb{N}$ solution vectors, or snapshots, of (2.1) with

parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M \in \mathcal{D}$. Selecting the snapshots, i.e., selecting the parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M \in \mathcal{D}$, is a widely studied problem in the context of model reduction. Many selection algorithms are based on greedy approaches, see, e.g., [42, 40, 8, 37] and especially for time-dependent problems [25]. We do not further consider how to best select the parameters of the snapshots here, but we emphasize that the selection of snapshots can significantly impact the quality of the reduced system. POD constructs an orthonormal basis $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n] \in \mathbb{R}^{N \times n}$ of an n-dimensional space that is a solution to the minimization problem

$$\min_{\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n \in \mathbb{R}^N} \sum_{i=1}^{M} \left\| \boldsymbol{y}(\boldsymbol{\mu}_i) - \sum_{j=1}^{n} (\boldsymbol{v}_j^T \boldsymbol{y}(\boldsymbol{\mu}_i)) \boldsymbol{v}_j \right\|_2^2 .$$

The norm $\|\cdot\|_2$ is the Euclidean norm. The POD basis vectors in the matrix $\boldsymbol{V} \in \mathbb{R}^{N \times n}$ are the n left-singular vectors of $\boldsymbol{Y}$ corresponding to the n largest singular values. The POD-Galerkin reduced system of (2.1) is obtained by Galerkin projection as

$$\tilde{\boldsymbol{A}} \tilde{\boldsymbol{y}}(\boldsymbol{\mu}) + \boldsymbol{V}^T \boldsymbol{f}(\boldsymbol{V} \tilde{\boldsymbol{y}}(\boldsymbol{\mu})) = 0 \tag{2.2}$$

with the reduced linear operator $\tilde{\boldsymbol{A}} = \boldsymbol{V}^T \boldsymbol{A} \boldsymbol{V} \in \mathbb{R}^{n \times n}$, the reduced state vector $\tilde{\boldsymbol{y}}(\boldsymbol{\mu}) \in \mathbb{R}^n$, and the reduced Jacobian $\tilde{\boldsymbol{A}} + \boldsymbol{V}^T \boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{V} \tilde{\boldsymbol{y}}(\boldsymbol{\mu})) \boldsymbol{V} \in \mathbb{R}^{n \times n}$. For many problems, the solution $\boldsymbol{y}(\boldsymbol{\mu})$ of (2.1) is well approximated by $\boldsymbol{V} \tilde{\boldsymbol{y}}(\boldsymbol{\mu})$, even if the number of POD basis vectors n is chosen much smaller than the number of degrees of freedom N of system (2.1). However, in the case of nonlinear systems, solving the reduced system (2.2) instead of (2.1) does not necessarily lead to computational savings because the nonlinear function $\boldsymbol{f}$ is still evaluated at all N components of $\boldsymbol{V} \tilde{\boldsymbol{y}}(\boldsymbol{\mu}) \in \mathbb{R}^N$.

2.2. Discrete empirical interpolation method. DEIM approximates the nonlinear function $\boldsymbol{f}$ in a low-dimensional space by sampling $\boldsymbol{f}$ at only $m \ll N$ components and then approximating all other components. This can significantly speed up the computation time of solving the reduced system to determine the reduced state vector $\tilde{\boldsymbol{y}}(\boldsymbol{\mu}) \in \mathbb{R}^n$.

DEIM computes $m \in \mathbb{N}$ basis vectors by applying POD to the set of nonlinear snapshots

$$\{\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu}_1)), \ldots, \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu}_M))\} \subset \mathbb{R}^N . \tag{2.3}$$

This leads to the DEIM basis vectors that are stored as columns in the DEIM basis $\boldsymbol{U} \in \mathbb{R}^{N \times m}$. DEIM selects m pairwise distinct interpolation points $p_1, \ldots, p_m \in \{1, \ldots, N\}$ and assembles the DEIM interpolation points matrix $\boldsymbol{P} = [\boldsymbol{e}_{p_1}, \ldots, \boldsymbol{e}_{p_m}] \in \mathbb{R}^{N \times m}$, where $\boldsymbol{e}_i \in \{0, 1\}^N$ is the ith canonical unit vector. The interpolation points are constructed with a greedy approach inductively on the basis vectors in $\boldsymbol{U}$ [12, Algorithm 1]. Thus, the ith interpolation point p_i can be associated with the basis vector in the ith column of the DEIM basis $\boldsymbol{U}$. The DEIM interpolant of $\boldsymbol{f}$ is defined by the tuple $(\boldsymbol{U}, \boldsymbol{P})$ of the DEIM basis $\boldsymbol{U}$ and the DEIM interpolation points matrix $\boldsymbol{P}$. The DEIM approximation of the nonlinear function $\boldsymbol{f}$ evaluated at the state vector $\boldsymbol{y}(\boldsymbol{\mu})$ is given as

$$\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) \approx \boldsymbol{U} (\boldsymbol{P}^T \boldsymbol{U})^{-1} \boldsymbol{P}^T \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) , \tag{2.4}$$

where $\boldsymbol{P}^T \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu}))$ samples the nonlinear function at m components only. The DEIM interpolation points matrix $\boldsymbol{P}$ and the DEIM basis $\boldsymbol{U}$ are selected such that the matrix $(\boldsymbol{P}^T \boldsymbol{U})^{-1} \in \mathbb{R}^{m \times m}$ has full rank.

We combine DEIM and POD-Galerkin to obtain the POD-DEIM-Galerkin reduced system

$$\tilde{\boldsymbol{A}}\tilde{\boldsymbol{y}}(\boldsymbol{\mu}) + \boldsymbol{V}^T\boldsymbol{U}(\boldsymbol{P}^T\boldsymbol{U})^{-1}\boldsymbol{P}^T\boldsymbol{f}(\boldsymbol{V}\tilde{\boldsymbol{y}}(\boldsymbol{\mu})) = 0\,. \tag{2.5}$$

We assume the well-posedness of (2.5). The Jacobian is

$$\tilde{\boldsymbol{J}}(\boldsymbol{\mu}) = \underbrace{\tilde{\boldsymbol{A}}}_{n\times n} + \underbrace{\boldsymbol{V}^T\boldsymbol{U}(\boldsymbol{P}^T\boldsymbol{U})^{-1}}_{n\times m}\underbrace{\tilde{\boldsymbol{J}}_{\boldsymbol{f}}(\boldsymbol{P}^T\boldsymbol{V}\tilde{\boldsymbol{y}}(\boldsymbol{\mu}))}_{m\times m}\underbrace{\boldsymbol{P}^T\boldsymbol{V}}_{m\times n},$$

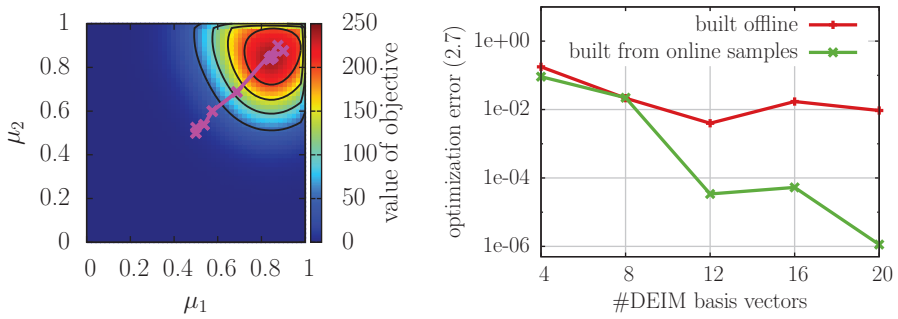
where we use the fact that the nonlinear function is evaluated componentwise at the state vector and follow [12] to define

$$\tilde{\boldsymbol{J}}_{\boldsymbol{f}}(\boldsymbol{P}^T\boldsymbol{V}\tilde{\boldsymbol{y}}(\boldsymbol{\mu})) = \tilde{\boldsymbol{J}}_{\boldsymbol{f}}(\boldsymbol{P}^T\boldsymbol{y}^r(\boldsymbol{\mu})) = \operatorname{diag}(f'_{p_1}(y^r_{p_1}(\boldsymbol{\mu})), \ldots, f'_{p_m}(y^r_{p_m}(\boldsymbol{\mu})))$$

with $\boldsymbol{y}^r(\boldsymbol{\mu}) = [y^r_1(\boldsymbol{\mu}), \ldots, y^r_N(\boldsymbol{\mu})]^T = \boldsymbol{V}\tilde{\boldsymbol{y}}(\boldsymbol{\mu})$. Solving (2.5) with, e.g., the Newton method evaluates the nonlinear function $\boldsymbol{f}$ at the interpolation points given by $\boldsymbol{P}$ only, instead of at all N components. The corresponding computational procedure of the POD-DEIM-Galerkin method is split into an offline phase where the POD-DEIM-Galerkin reduced system is constructed and an online phase where it is evaluated. The one-time high computational costs of building the DEIM interpolant and the reduced system in the offline phase are compensated during the online phase where the reduced (2.5) instead of the full-order system (2.1) is solved for a large number of parameters.

2.3. Problem formulation. Let $(\boldsymbol{U}_0, \boldsymbol{P}_0)$ be the DEIM interpolant of the nonlinear function $\boldsymbol{f}$, with m DEIM basis vectors and m DEIM interpolation points, that is built in the offline phase from the nonlinear snapshots $\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu}_1)), \ldots, \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu}_M)) \in \mathbb{R}^N$ with parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_M \in \mathcal{D}$. We consider the situation where in the online phase, the application at hand (e.g., optimization, uncertainty quantification, or parameter inference) requests $M' \in \mathbb{N}$ solutions $\tilde{\boldsymbol{y}}(\boldsymbol{\mu}_{M+1}), \ldots, \tilde{\boldsymbol{y}}(\boldsymbol{\mu}_{M+M'}) \in \mathbb{R}^n$ of the POD-DEIM-Galerkin reduced system (2.5), with parameters $\boldsymbol{\mu}_{M+1}, \ldots, \boldsymbol{\mu}_{M+M'} \in \mathcal{D}$. Solving the POD-DEIM-Galerkin reduced system requires DEIM approximations of the nonlinear function at the vectors $\boldsymbol{V}\tilde{\boldsymbol{y}}(\boldsymbol{\mu}_{M+1}), \ldots, \boldsymbol{V}\tilde{\boldsymbol{y}}(\boldsymbol{\mu}_{M+M'}) \in \mathbb{R}^N$. Note that for the sake of exposition, we ignore that an iterative solution method (e.g., Newton method) might also require DEIM approximations at intermediate iterates of the reduced state vectors. We define $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_i) = \boldsymbol{y}(\boldsymbol{\mu}_i)$ for $i = 1, \ldots, M$ and $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_i) = \boldsymbol{V}\tilde{\boldsymbol{y}}(\boldsymbol{\mu}_i)$ for $i = M+1, \ldots, M+M'$. Then, the online phase consists of $k = 1, \ldots, M'$ steps, where, at step k, the nonlinear function $\boldsymbol{f}(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k}))$ is approximated with DEIM. We therefore aim to provide at step k a DEIM interpolant that approximates $\boldsymbol{f}(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k}))$ well.

The quality of the DEIM approximation of $\boldsymbol{f}(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k}))$ depends on how well the nonlinear function $\boldsymbol{f}(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k}))$ can be represented in the DEIM basis and how well the components selected by the DEIM interpolation points represent the overall behavior of the nonlinear function at $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k})$; however, when the DEIM interpolant is built offline, the reduced state vectors $\tilde{\boldsymbol{y}}(\boldsymbol{\mu}_{M+1}), \ldots, \tilde{\boldsymbol{y}}(\boldsymbol{\mu}_{M+M'})$ are not known, and thus the DEIM basis and the DEIM interpolation points cannot be constructed to explicitly take the vectors $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+1}), \ldots, \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+M'})$ into account. Rebuilding the interpolant online would require evaluating the nonlinear function at full-order state vectors and computing the singular value decomposition (SVD) of the snapshot matrix, which would entail high computational costs. We therefore present in the following a computationally efficient online adaptivity procedure to adapt a DEIM interpolant with only a few samples of the nonlinear function that can be cheaply computed in the online phase.



(a) path to the approximate optimum $\tilde{\boldsymbol{\mu}}^*$ (b) optimization error

FIG. 1. *The plot in* (a) *shows the path of the optimization algorithm to the optimum* $\tilde{\boldsymbol{\mu}}^*$ *of* $\mathbf{1}_n^T\tilde{\boldsymbol{g}}(\boldsymbol{\mu})$ *with* $\tilde{\boldsymbol{g}}$ *using* $m = 100$ *DEIM basis vectors. The DEIM interpolant* $\tilde{\boldsymbol{g}}$ *is evaluated at only a few selected points, but these points are not known when the DEIM interpolant is constructed. The results in* (b) *show that if the DEIM interpolant is rebuilt from snapshots corresponding to those points, the optimization algorithm converges faster to the optimum than with the original DEIM interpolant built from snapshots corresponding to a uniform grid in the parameter domain* $\mathcal{D}$.

2.4. Motivating example. Before presenting our adaptivity approach, we motivate online adaptive DEIM interpolants by illustrating on a synthetic optimization problem that incorporating data from the online phase can increase the DEIM approximation accuracy. Let $\Omega = [0,1]^2 \subset \mathbb{R}^2$ be the spatial domain and let $\mathcal{D} = [0,1]^2 \subset \mathbb{R}^2$ be the parameter domain of the nonlinear function $g : \Omega \times \mathcal{D} \to \mathbb{R}$ that is defined as

$$g(\boldsymbol{x}, \boldsymbol{\mu}) = \frac{\mu_1 \mu_2 \exp(x_1 x_2)}{\exp(20\|\boldsymbol{x} - \boldsymbol{\mu}\|_2^2)}\,. \tag{2.6}$$

We discretize g in the spatial domain on an equidistant 40×40 grid with $N = 1600$ grid points with spatial coordinates $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \Omega$ and obtain the vector-valued function $\boldsymbol{g} : \mathcal{D} \to \mathbb{R}^N$ with the ith component $g_i(\boldsymbol{\mu}) = g(\boldsymbol{x}_i, \boldsymbol{\mu})$. We are then interested in the parameter $\boldsymbol{\mu}^*$ that maximizes $\mathbf{1}_N^T\boldsymbol{g}(\boldsymbol{\mu})$, where $\mathbf{1}_N = [1, \ldots, 1]^T \in \mathbb{R}^N$. We do not directly evaluate $\boldsymbol{g}$ but first derive a DEIM interpolant $\tilde{\boldsymbol{g}}$ of $\boldsymbol{g}$ and then search for $\tilde{\boldsymbol{\mu}}^*$ that maximizes the approximate objective $\mathbf{1}_n^T\tilde{\boldsymbol{g}}(\boldsymbol{\mu})$ with $\mathbf{1}_n^T = [1, \ldots, 1]^T \in \mathbb{R}^n$.

In the offline phase, we neither know the optimal parameter $\boldsymbol{\mu}^*$ nor the path of the optimization algorithm to the optimum and thus cannot use this information when constructing the DEIM interpolant. Thus, we build the interpolant from $M = 400$ snapshots corresponding to the parameters on a 20×20 equidistant grid in the parameter domain $\mathcal{D}$. We run an optimization algorithm, here Nelder–Mead [28], which evaluates the DEIM interpolant at M' parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{M'} \in \mathcal{D}$; see Figure 1(a). The starting point is $[0.5, 0.5] \in \mathcal{D}$. To demonstrate the gain of including information from the online phase into the DEIM interpolant, we generate new nonlinear snapshots by evaluating the nonlinear function $\boldsymbol{g}$ at those M' parameters and then construct a DEIM interpolant from them. We rerun the Nelder–Mead algorithm with the new DEIM interpolant and report the optimization error,

$$\frac{\|\boldsymbol{\mu}^* - \tilde{\boldsymbol{\mu}}^*\|_2}{\|\boldsymbol{\mu}^*\|_2}\,, \tag{2.7}$$

for the original and the new DEIM interpolant in Figure 1(b). The new interpolant, which takes online data into account, achieves an accuracy improvement by four orders of magnitude compared to the original DEIM interpolant built from offline data

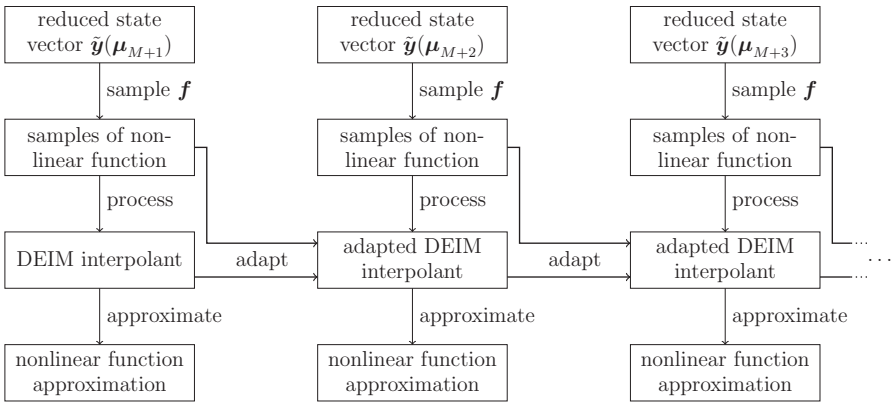


FIG. 2. *The figure shows the workflow of the online phase of model reduction with online adaptive DEIM interpolants. The DEIM interpolant is adapted with the samples of the nonlinear function of the previous evaluations. The adaptation requires neither additional snapshots nor nonlinear function evaluations at full state vectors.*

only. This is certainly not a practical approach, because it requires solving the problem twice, but it shows that the DEIM approximation accuracy can be significantly improved by incorporating data from the online phase. We therefore present in the following a more practical way to adapt the DEIM interpolant online.

3. Online basis adaptivity. We adapt the DEIM interpolant at each step $k = 1, \ldots, M'$ of the online phase by deriving low-rank updates to the DEIM basis and the DEIM interpolation points matrix; see Figure 2. The adaptation is initialized at the first step $k = 1$ in the online phase with the DEIM interpolant $(\boldsymbol{U}_0, \boldsymbol{P}_0)$ from the offline phase, from which the adapted interpolant $(\boldsymbol{U}_1, \boldsymbol{P}_1)$ is derived. This process is continued to construct at step k the adapted interpolant $(\boldsymbol{U}_k, \boldsymbol{P}_k)$ from $(\boldsymbol{U}_{k-1}, \boldsymbol{P}_{k-1})$. At each step $k = 1, \ldots, M'$, the adapted interpolant $(\boldsymbol{U}_k, \boldsymbol{P}_k)$ is then used to provide an approximation of the nonlinear function at the vector $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k}) = \boldsymbol{V}\tilde{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k})$.

This section introduces the DEIM basis update. Section 3.1 proposes a residual relating to the DEIM approximation quality of the nonlinear function. This residual is exploited in sections 3.2 and 3.3 to construct a basis update. The computational procedure and its computational complexity are discussed in section 3.4. We close, with section 3.5, on remarks on the properties of the basis update.

3.1. Residual. A DEIM interpolant $(\boldsymbol{U}, \boldsymbol{P})$ computes an approximation of the nonlinear function $\boldsymbol{f}$ at a state vector $\boldsymbol{y}(\boldsymbol{\mu})$ by sampling $\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu}))$ at the components defined by the DEIM interpolation points; see section 2.2. The DEIM approximation interpolates $\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu}))$ at the DEIM interpolation points, and thus the residual,

$$\boldsymbol{U}(\boldsymbol{P}^T\boldsymbol{U})^{-1}\boldsymbol{P}^T\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) - \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu}))\,,$$

is zero at the interpolation points, i.e.,

$$\left\| \boldsymbol{P}^T \left(\boldsymbol{U}(\boldsymbol{P}^T\boldsymbol{U})^{-1}\boldsymbol{P}^T\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) - \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) \right) \right\|_2 = 0\,.$$

We now extend the set of m interpolation points $\{p_1, \ldots, p_m\}$ to a set of $m + m_s$ pairwise distinct sampling points $\{s_1, \ldots, s_{m+m_s}\} \subset \{1, \ldots, N\}$ with $m_s \in \mathbb{N}$ and $m_s > 0$. The first m sampling points $s_1, \ldots, s_m$ coincide with the DEIM interpolation

points, and the other m_s points are drawn randomly with a uniform distribution from $\{1, \ldots, N\} \setminus \{p_1, \ldots, p_m\}$. Note that we remark on the selection of the sampling points in section 3.5. The corresponding sampling points matrix,

$$\boldsymbol{S} = [\boldsymbol{e}_{s_1}, \ldots, \boldsymbol{e}_{s_{m+m_s}}] \in \mathbb{R}^{N \times (m+m_s)} ,$$

is derived similarly to the DEIM interpolation points matrix $\boldsymbol{P}$; see section 2.2. The nonlinear function $\boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu}))$ is then approximated by $\boldsymbol{U}\boldsymbol{c}(\boldsymbol{y}(\boldsymbol{\mu}))$, where the coefficient $\boldsymbol{c}(\boldsymbol{y}(\boldsymbol{\mu})) \in \mathbb{R}^m$ is the solution of the overdetermined regression problem

$$\boldsymbol{S}^T \boldsymbol{U} \boldsymbol{c}(\boldsymbol{y}(\boldsymbol{\mu})) = \boldsymbol{S}^T \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) . \tag{3.1}$$

With the Moore–Penrose pseudoinverse $(\boldsymbol{S}^T\boldsymbol{U})^+ \in \mathbb{R}^{m \times (m+m_s)}$, the solution of (3.1) is the coefficient

$$\boldsymbol{c}(\boldsymbol{y}(\boldsymbol{\mu})) = (\boldsymbol{S}^T \boldsymbol{U})^+ \boldsymbol{S}^T \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) . \tag{3.2}$$

In general, the $m + m_s$ sampling points lead to a residual

$$\boldsymbol{r}(\boldsymbol{y}(\boldsymbol{\mu})) = \boldsymbol{U}\boldsymbol{c}(\boldsymbol{y}(\boldsymbol{\mu})) - \boldsymbol{f}(\boldsymbol{y}(\boldsymbol{\mu})) \tag{3.3}$$

that is nonzero at the sampling points, i.e., $\|\boldsymbol{S}^T \boldsymbol{r}(\boldsymbol{y}(\boldsymbol{\mu}))\|_2 > 0$.

3.2. Adapting the DEIM basis. For the basis adaptation at step k, we define a window of size $w \in \mathbb{N}$ that contains the vector $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k})$ and the vectors $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k-w+1}), \ldots, \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{M+k-1})$ of the previous $w-1$ steps. If $k < w$, then the previous $w-1$ vectors also include snapshots from the offline phase;[1] see section 2.3. For the sake of exposition, we introduce for each step k a vector $\boldsymbol{k} = [k_1, \ldots, k_w]^T \in \mathbb{N}^w$ with $k_i = M + k - w + i$, such that $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_1}), \ldots, \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_w})$ are the vectors in the window.

At each step k, we generate $m + m_s$ sampling points and assemble the corresponding sampling points matrix $\boldsymbol{S}_k$. The first m sampling points correspond to the DEIM interpolation points given by $\boldsymbol{P}_{k-1}$. The remaining sampling points are chosen randomly, as discussed in section 3.1. We then construct approximations of the nonlinear function $\boldsymbol{f}$ at the vectors $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_1}), \ldots, \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_w})$ with the DEIM basis $\boldsymbol{U}_{k-1}$ but with the sampling points matrix $\boldsymbol{S}_k$ instead of $\boldsymbol{P}_{k-1}$. For $i = 1, \ldots, w$, the coefficient $\boldsymbol{c}_k(\boldsymbol{y}(\boldsymbol{\mu}_{k_i})) \in \mathbb{R}^m$ of the approximation $\boldsymbol{U}_{k-1}\boldsymbol{c}_k(\boldsymbol{y}(\boldsymbol{\mu}_{k_i}))$ of $\boldsymbol{f}(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_i}))$ is derived following (3.2) as

$$\boldsymbol{c}_k(\boldsymbol{y}(\boldsymbol{\mu}_{k_i})) = (\boldsymbol{S}_k^T \boldsymbol{U}_{k-1})^+ \boldsymbol{S}_k^T \boldsymbol{f}(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_i})) . \tag{3.4}$$

The coefficients $\boldsymbol{c}_k(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_1})), \ldots, \boldsymbol{c}_k(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_w}))$ are put as columns in the coefficient matrix $\boldsymbol{C}_k \in \mathbb{R}^{m \times w}$.

We then derive two vectors, $\boldsymbol{\alpha}_k \in \mathbb{R}^N$ and $\boldsymbol{\beta}_k \in \mathbb{R}^m$, such that the adapted basis $\boldsymbol{U}_k = \boldsymbol{U}_{k-1} + \boldsymbol{\alpha}_k \boldsymbol{\beta}_k^T$ minimizes the Frobenius norm of the residual at the sampling points given by $\boldsymbol{S}_k$

$$\left\| \boldsymbol{S}_k^T \left(\boldsymbol{U}_k \boldsymbol{C}_k - \boldsymbol{F}_k \right) \right\|_F^2 , \tag{3.5}$$

where the right-hand side matrix $\boldsymbol{F}_k = [\boldsymbol{f}(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_1})), \ldots, \boldsymbol{f}(\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_w}))] \in \mathbb{R}^{N \times w}$ contains as columns the nonlinear function evaluated at the state vectors $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_1}), \ldots, \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_w})$.

[1] In this case, the ordering of the snapshots affects the online adaptivity process.

Note that only $\boldsymbol{S}_k^T\boldsymbol{F}_k \in \mathbb{R}^{(m+m_s)\times w}$ is required in (3.5), and not the complete matrix $\boldsymbol{F}_k \in \mathbb{R}^{N\times w}$. Note further that $\boldsymbol{F}_k$ may contain snapshots from the offline phase if $k < w$; see the first paragraph of this subsection. We define the residual matrix $\boldsymbol{R}_k = \boldsymbol{U}_{k-1}\boldsymbol{C}_k - \boldsymbol{F}_k$ and transform (3.5) into

$$\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{S}_k^T\boldsymbol{\alpha}_k\boldsymbol{\beta}_k^T\boldsymbol{C}_k\|_F^2 .$$

Thus, the vectors $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ of the update $\boldsymbol{\alpha}_k\boldsymbol{\beta}_k^T \in \mathbb{R}^{N\times m}$ are a solution of the minimization problem

$$\operatorname*{arg\,min}_{\boldsymbol{\alpha}_k\in\mathbb{R}^N,\boldsymbol{\beta}_k\in\mathbb{R}^m} \|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{S}_k^T\boldsymbol{\alpha}_k\boldsymbol{\beta}_k^T\boldsymbol{C}_k\|_F^2 . \tag{3.6}$$

3.3. Optimality of basis update. We show in this section that an optimal update $\boldsymbol{\alpha}_k\boldsymbol{\beta}_k^T$, i.e., a solution of the minimization problem (3.6), can be computed from an eigenvector corresponding to the largest eigenvalue of a generalized eigenproblem. We first consider five auxiliary lemmata and then derive the optimal update $\boldsymbol{\alpha}_k\boldsymbol{\beta}_k^T$ in Theorem 3.6. In the following, we exclude the trivial case where the matrices $\boldsymbol{S}_k^T\boldsymbol{R}_k$ and $\boldsymbol{C}_k$ have zero entries only.

LEMMA 3.1. *Let $\boldsymbol{S}_k^T\boldsymbol{R}_k \in \mathbb{R}^{(m+m_s)\times w}$ and let $\boldsymbol{\alpha} \in \mathbb{R}^{m+m_s}$ be the left and $\boldsymbol{\beta} \in \mathbb{R}^w$ be the right singular vector of $\boldsymbol{S}_k^T\boldsymbol{R}_k$ corresponding to the singular value $\sigma > 0$. For a vector $\boldsymbol{z} \in \mathbb{R}^w$, we have*

$$\|\boldsymbol{S}_k^T\boldsymbol{R}_k - \boldsymbol{\alpha}\boldsymbol{z}^T\|_F^2 = \|\boldsymbol{S}_k^T\boldsymbol{R}_k - \boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T\|_F^2 + \|\boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T - \boldsymbol{\alpha}\boldsymbol{z}^T\|_F^2 . \tag{3.7}$$

Proof. We have

$$\|\boldsymbol{S}_k^T\boldsymbol{R}_k - \boldsymbol{\alpha}\boldsymbol{z}^T\|_F^2 = \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2 - 2\boldsymbol{\alpha}^T\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{z} + \|\boldsymbol{\alpha}\boldsymbol{z}^T\|_F^2$$

and

$$\|\boldsymbol{S}_k^T\boldsymbol{R}_k - \boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T\|_F^2 = \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2 - 2\boldsymbol{\alpha}^T\boldsymbol{S}_k^T\boldsymbol{R}_k\sigma\boldsymbol{\beta} + \|\boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T\|_F^2$$

and

$$\|\boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T - \boldsymbol{\alpha}\boldsymbol{z}^T\|_F^2 = \|\boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T\|_F^2 - 2\boldsymbol{\alpha}^T\boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T\boldsymbol{z} + \|\boldsymbol{\alpha}\boldsymbol{z}^T\|_F^2 .$$

We show

$$-2\boldsymbol{\alpha}^T\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{z} = -2\boldsymbol{\alpha}^T\boldsymbol{S}_k^T\boldsymbol{R}_k\sigma\boldsymbol{\beta} + 2\|\boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T\|_F^2 - 2\boldsymbol{\alpha}^T\boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T\boldsymbol{z} . \tag{3.8}$$

Using $(\boldsymbol{S}_k^T\boldsymbol{R}_k)^T\boldsymbol{\alpha} = \sigma\boldsymbol{\beta}$ and $\boldsymbol{\alpha}^T\boldsymbol{\alpha} = 1$, we find

$$\|\boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T\|_F^2 = \sigma^2\boldsymbol{\alpha}^T\boldsymbol{\alpha}\boldsymbol{\beta}^T\boldsymbol{\beta} = \boldsymbol{\alpha}^T\boldsymbol{S}_k^T\boldsymbol{R}_k\sigma\boldsymbol{\beta}$$

and $\boldsymbol{\alpha}^T\boldsymbol{\alpha}\sigma\boldsymbol{\beta}^T\boldsymbol{z} = \boldsymbol{\alpha}^T\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{z}$, which shows (3.8) and therefore (3.7). ∎

LEMMA 3.2. *Let $r \in \mathbb{N}$ be the rank of $\boldsymbol{S}_k^T\boldsymbol{R}_k \in \mathbb{R}^{(m+m_s)\times w}$, and let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0 \in \mathbb{R}$ be the singular values of $\boldsymbol{S}_k^T\boldsymbol{R}_k$. Let further $\boldsymbol{\alpha}_i' \in \mathbb{R}^{m+m_s}$ and $\boldsymbol{\beta}_i' \in \mathbb{R}^w$ be the left and the right singular vector, respectively, that correspond to a singular value σ_i. Set $\boldsymbol{a} = -\boldsymbol{\alpha}_i' \in \mathbb{R}^{m+m_s}$ and let $\boldsymbol{b} \in \mathbb{R}^m$ be a solution of the minimization problem*

$$\operatorname*{arg\,min}_{\boldsymbol{b}\in\mathbb{R}^m} \|\sigma_i\boldsymbol{\beta}_i' - \boldsymbol{C}_k^T\boldsymbol{b}\|_2^2 , \tag{3.9}$$

then $\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 \leq \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2$ *holds, and* $\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 < \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2$ *holds if* $\|\boldsymbol{C}_k\boldsymbol{\beta}_i'\|_2 > 0$.

Proof. Since $\boldsymbol{a} = -\boldsymbol{\alpha}_i'$ and because of Lemma 3.1, we find

$$\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 = \|\boldsymbol{S}_k^T\boldsymbol{R}_k - \boldsymbol{\alpha}_i'\sigma_i\boldsymbol{\beta}_i'^T\|_F^2 + \|\boldsymbol{\alpha}_i'\sigma_i\boldsymbol{\beta}_i'^T - \boldsymbol{\alpha}_i'\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2\,. \tag{3.10}$$

The first term of the right-hand side of (3.10) equals $\sum_{j\neq i}^r \sigma_j^2$. For the second term of the right-hand side of (3.10), we have

$$\|\boldsymbol{\alpha}_i'\sigma_i\boldsymbol{\beta}_i'^T - \boldsymbol{\alpha}_i'\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 = \|\sigma_i\boldsymbol{\beta}_i' - \boldsymbol{C}_k^T\boldsymbol{b}\|_2^2 \leq \sigma_i^2\,,$$

because $\|\boldsymbol{\alpha}_i'\|_2^2 = \|\boldsymbol{\beta}_i'\|_2^2 = 1$. This shows that $\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 \leq \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2$ because $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2 = \sum_{j=1}^r \sigma_j^2$. If $\|\boldsymbol{C}_k\boldsymbol{\beta}_i'\|_2 > 0$, then the rows of $\boldsymbol{C}_k$, and thus the columns of $\boldsymbol{C}_k^T$, cannot all be orthogonal to $\boldsymbol{\beta}_i'$, and therefore a $\boldsymbol{b} \in \mathbb{R}^m$ exists with $\|\sigma_i\boldsymbol{\beta}_i' - \boldsymbol{C}_k^T\boldsymbol{b}\|_2^2 < \sigma_i^2$, which shows $\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 < \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2$. ∎

We note that in [22] a similar update as in Lemma 3.2 is used to derive a low-rank approximation of a matrix.

LEMMA 3.3. *There exist* $\boldsymbol{a} \in \mathbb{R}^{m+m_s}$ *and* $\boldsymbol{b} \in \mathbb{R}^m$ *with* $\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 < \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2$ *if and only if* $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F > 0$.

Proof. Let $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F = 0$, which leads to

$$\begin{aligned}\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 &= \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2 + 2\boldsymbol{a}^T\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\boldsymbol{b} + \|\boldsymbol{a}\|_2^2\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2^2 \\ &= \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2 + \|\boldsymbol{a}\|_2^2\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2^2\,,\end{aligned}$$

and thus $\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 < \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2$ cannot hold. See Lemma 3.2 for the case $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F > 0$ and note that the right singular vectors span the row space of $\boldsymbol{S}_k^T\boldsymbol{R}_k$. ∎

LEMMA 3.4. *Let* $\boldsymbol{C}_k \in \mathbb{R}^{m\times w}$ *have rank* $r < m$, *i.e.,* $\boldsymbol{C}_k$ *does not have full row rank. There exists a matrix* $\boldsymbol{Z}_k \in \mathbb{R}^{r\times w}$, *with rank* r, *and a matrix* $\boldsymbol{Q}_k \in \mathbb{R}^{m\times r}$ *with orthonormal columns such that*

$$\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 = \|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{z}^T\boldsymbol{Z}_k\|_F^2$$

for all $\boldsymbol{a} \in \mathbb{R}^{m+m_s}$ *and* $\boldsymbol{b} \in \mathbb{R}^m$, *where* $\boldsymbol{z}^T = \boldsymbol{b}^T\boldsymbol{Q}_k \in \mathbb{R}^r$.

Proof. With the rank revealing QR decomposition [24, Theorem 5.2.1] of the matrix $\boldsymbol{C}_k$, we obtain a matrix $\boldsymbol{Q}_k \in \mathbb{R}^{m\times r}$ with orthonormal columns and a matrix $\boldsymbol{Z}_k \in \mathbb{R}^{r\times w}$ with rank r, such that $\boldsymbol{C}_k = \boldsymbol{Q}_k\boldsymbol{Z}_k$. This leads to

$$\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 = \|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{Q}_k\boldsymbol{Z}_k\|_F^2 = \|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{z}^T\boldsymbol{Z}_k\|_F^2\,.$$ ∎

LEMMA 3.5. *Let* $\boldsymbol{C}_k \in \mathbb{R}^{m\times w}$ *have rank* m, *i.e., full row rank, and assume* $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F > 0$. *Let* $\boldsymbol{\beta}_k' \in \mathbb{R}^m$ *be an eigenvector corresponding to the largest eigenvalue* $\lambda \in \mathbb{R}$ *of the generalized eigenvalue problem*

$$\boldsymbol{C}_k(\boldsymbol{S}_k^T\boldsymbol{R}_k)^T(\boldsymbol{S}_k^T\boldsymbol{R}_k)\boldsymbol{C}_k^T\boldsymbol{\beta}_k' = \lambda\boldsymbol{C}_k\boldsymbol{C}_k^T\boldsymbol{\beta}_k'\,, \tag{3.11}$$

and set $\boldsymbol{\alpha}_k' = -1/\|\boldsymbol{C}_k^T\boldsymbol{\beta}_k'\|_2^2\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\boldsymbol{\beta}_k'$. *The vectors* $\boldsymbol{\alpha}_k'$ *and* $\boldsymbol{\beta}_k'$ *are a solution of the minimization problem*

$$\operatorname*{arg\,min}_{\boldsymbol{a}\in\mathbb{R}^{m+m_s},\boldsymbol{b}\in\mathbb{R}^m} \|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2\,. \tag{3.12}$$

Proof. We have $\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2 = 0$ if and only if $\boldsymbol{b} = \boldsymbol{0}_m = [0,\dots,0]^T \in \mathbb{R}^m$ because $\boldsymbol{C}_k$ has full row rank. Furthermore, since $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F > 0$, the vector $\boldsymbol{b} = \boldsymbol{0}_m$ cannot be a solution of the minimization problem (3.12); see Lemma 3.3. We therefore have in the following $\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2 > 0$. The gradient of the objective of (3.12) with respect to $\boldsymbol{a}$ and $\boldsymbol{b}$ is

$$\begin{bmatrix} 2\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\boldsymbol{b} + 2\boldsymbol{a}\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2^2 \\ 2\boldsymbol{C}_k(\boldsymbol{S}_k^T\boldsymbol{R}_k)^T\boldsymbol{a} + 2\|\boldsymbol{a}\|_2^2\boldsymbol{C}_k\boldsymbol{C}_k^T\boldsymbol{b} \end{bmatrix} \in \mathbb{R}^{m+m_s+m}. \tag{3.13}$$

By setting the gradient (3.13) to zero, we obtain from the first $m + m_s$ components of (3.13) that

$$\boldsymbol{a} = \frac{-1}{\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2^2}\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\boldsymbol{b}. \tag{3.14}$$

We plug (3.14) into the remaining m components of the gradient (3.13) and find that the gradient (3.13) is zero if for $\boldsymbol{b}$ the following equality holds:

$$\boldsymbol{C}_k(\boldsymbol{S}_k^T\boldsymbol{R}_k)^T(\boldsymbol{S}_k^T\boldsymbol{R}_k)\boldsymbol{C}_k^T\boldsymbol{b} = \frac{\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\boldsymbol{b}\|_2^2}{\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2^2}\boldsymbol{C}_k\boldsymbol{C}_k^T\boldsymbol{b}. \tag{3.15}$$

Let us therefore consider the eigenproblem (3.11). First, we show that all eigenvalues of (3.11) are real. The left matrix of (3.11) is symmetric and cannot be the zero matrix because $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F > 0$. The right matrix $\boldsymbol{C}_k\boldsymbol{C}_k^T$ of (3.11) is symmetric positive definite because $\boldsymbol{C}_k$ has full row rank. Therefore, the generalized eigenproblem (3.11) can be transformed into a real symmetric eigenproblem, for which all eigenvalues are real. Second, this also implies that the eigenvalues of (3.11) are insensitive to perturbations (well-conditioned) [24, section 7.2.2]. Third, for an eigenvector $\boldsymbol{b}$ of (3.11) with eigenvalue λ, we have $\lambda = \|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\boldsymbol{b}\|_2^2/\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2^2$. Therefore, all eigenvectors of the generalized eigenproblem (3.11) lead to the gradient (3.13) being zero if the vector $\boldsymbol{a}$ is set as in (3.14).

If $\boldsymbol{b}$ does not satisfy (3.15) and thus cannot be an eigenvector of (3.11), $\boldsymbol{b}$ cannot lead to a zero gradient and therefore cannot be part of a solution of the minimization problem (3.12). Because for an eigenvector $\boldsymbol{b}$, we have that

$$\|\boldsymbol{S}_k^T\boldsymbol{R}_k + \boldsymbol{a}\boldsymbol{b}^T\boldsymbol{C}_k\|_F^2 = \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2 - 2\frac{\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\boldsymbol{b}\|_2^2}{\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2^2} + \|\boldsymbol{a}\|_2^2\|\boldsymbol{b}^T\boldsymbol{C}_k\|_2^2 = \|\boldsymbol{S}_k^T\boldsymbol{R}_k\|_F^2 - \lambda,$$

and because of (3.14), we obtain that an eigenvector of (3.11) corresponding to the largest eigenvalue leads to a global optimum of (3.12). This shows that $\boldsymbol{\alpha}_k'$ and $\boldsymbol{\beta}_k'$ are a solution of the minimization problem (3.12). ∎

THEOREM 3.6. *If $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F > 0$, and after the transformation of Lemma* 3.4, *an optimal update $\boldsymbol{\alpha}_k\boldsymbol{\beta}_k^T$ with respect to* (3.6) *is given by setting $\boldsymbol{\alpha}_k = \boldsymbol{S}_k\boldsymbol{\alpha}_k'$ and $\boldsymbol{\beta}_k = \boldsymbol{Q}_k\boldsymbol{\beta}_k'$, where $\boldsymbol{\alpha}_k'$ and $\boldsymbol{\beta}_k'$ are defined as in Lemma* 3.5, *and where $\boldsymbol{Q}_k \in \mathbb{R}^{m\times r}$ is given as in Lemma* 3.4. *If $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F = 0$, an optimal update with respect to* (3.6) *is $\boldsymbol{\alpha}_k = \boldsymbol{0}_N \in \mathbb{R}^N$ and $\boldsymbol{\beta}_k = \boldsymbol{0}_w \in \mathbb{R}^w$, where $\boldsymbol{0}_N$ and $\boldsymbol{0}_w$ are the N- and w-dimensional null vectors, respectively.*

Proof. The case $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F > 0$ follows from Lemmata 3.4 and 3.5. The case $\|\boldsymbol{S}_k^T\boldsymbol{R}_k\boldsymbol{C}_k^T\|_F = 0$ follows from Lemma 3.3. ∎

Algorithm 1. Adapt interpolation basis with rank-one update.

procedure adaptBasis($\boldsymbol{U}_{k-1}$, $\boldsymbol{P}_{k-1}$)
  Select randomly m_s points from $\{1, \ldots, N\}$ which are not points in $\boldsymbol{P}_{k-1}$
  Assemble sampling points matrix $\boldsymbol{S}_k$ by combining $\boldsymbol{P}_{k-1}$ and sampling points
  Evaluate the components of the right-hand side matrix $\boldsymbol{F}_k$ selected by $\boldsymbol{S}_k$
  Compute the coefficient matrix $\boldsymbol{C}_k$, and $\boldsymbol{C}_k = \boldsymbol{Q}_k \boldsymbol{Z}_k$ following Lemma 3.4
  Compute residual at the sampling points $\boldsymbol{S}_k^T \boldsymbol{R}_k = \boldsymbol{S}_k^T (\boldsymbol{U}_{k-1} \boldsymbol{C}_k - \boldsymbol{F}_k)$
  Compute an $\boldsymbol{\alpha}_k'$ and $\boldsymbol{\beta}_k'$ following Lemma 3.5
  Set $\boldsymbol{\alpha}_k = \boldsymbol{S}_k \boldsymbol{\alpha}_k'$
  Set $\boldsymbol{\beta}_k = \boldsymbol{Q}_k \boldsymbol{\beta}_k'$
  Update basis $\boldsymbol{U}_k = \boldsymbol{U}_{k-1} + \boldsymbol{\alpha}_k \boldsymbol{\beta}_k^T$
return $\boldsymbol{U}_k$
end procedure

3.4. Computational procedure. The computational procedure of the online basis update is summarized in Algorithm 1. The procedure in Algorithm 1 is called at each step $k = 1, \ldots, M'$ in the online phase. The input parameters are the DEIM basis $\boldsymbol{U}_{k-1}$ and the DEIM interpolation points matrix $\boldsymbol{P}_{k-1}$ of the previous step $k-1$. First, m_s random and uniformly distributed points are drawn from the set $\{1, \ldots, N\} \subset \mathbb{N}$ such that the points are pairwise distinct from the interpolation points given by $\boldsymbol{P}_{k-1}$. They are then combined with the interpolation points into the sampling points, from which the sampling points matrix $\boldsymbol{S}_k$ is assembled. With the sampling points matrix $\boldsymbol{S}_k$, the matrix $\boldsymbol{S}_k^T \boldsymbol{F}_k$ is constructed. We emphasize that we only need $\boldsymbol{S}_k^T \boldsymbol{F}_k \in \mathbb{R}^{(m+m_s) \times w}$ and not all components of the right-hand side matrix $\boldsymbol{F}_k \in \mathbb{R}^{N \times w}$. The coefficient matrix $\boldsymbol{C}_k$ is constructed with respect to the basis $\boldsymbol{U}_{k-1}$ and the right-hand side $\boldsymbol{S}_k^T \boldsymbol{F}_k$. Then the residual at the sampling points $\boldsymbol{S}_k^T \boldsymbol{R}_k$ is computed and the update $\boldsymbol{\alpha}_k \boldsymbol{\beta}_k^T$ is derived from an eigenvector of the generalized eigenproblem (3.11) corresponding to the largest eigenvalue. Finally, the additive update $\boldsymbol{\alpha}_k \boldsymbol{\beta}_k^T$ is added to $\boldsymbol{U}_{k-1}$ to obtain the adapted basis $\boldsymbol{U}_k = \boldsymbol{U}_{k-1} + \boldsymbol{\alpha}_k \boldsymbol{\beta}_k^T$.

Algorithm 1 has linear runtime complexity with respect to the number of degrees of freedom N of the full-order system. Selecting the sampling points is in $\mathcal{O}(m_s m)$ and assembling the matrix $\boldsymbol{S}_k$ is in $\mathcal{O}(N(m + m_s))$. The costs of assembling the coefficient matrix are in $\mathcal{O}((m + m_s)^3 w)$, which do not include the costs of sampling the nonlinear function. If the nonlinear function is expensive to evaluate, then sampling the nonlinear function can dominate the overall computational costs. In the worst case, the nonlinear function is sampled at $w(m + m_s)$ components at each step $k = 1, \ldots, M'$; however, the runtime results of the numerical examples in section 5 show that in many situations these additional costs are compensated for by the online adaptivity that allows a reduction of the number of DEIM basis vectors m without loss of accuracy. We emphasize once more that the nonlinear function is only sampled at the $m + m_s$ components of the vectors $\boldsymbol{S}_k^T \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_1}), \ldots, \boldsymbol{S}_k^T \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_w}) \in \mathbb{R}^{m+m_s}$ and not at all N components of $\hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_1}), \ldots, \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_w}) \in \mathbb{R}^N$; cf. the definition of the coefficients in (3.4). Computing the vectors $\boldsymbol{S}_k^T \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_1}), \ldots, \boldsymbol{S}_k^T \hat{\boldsymbol{y}}(\boldsymbol{\mu}_{k_w}) \in \mathbb{R}^{m+m_s}$ from the reduced state vectors $\tilde{\boldsymbol{y}}(\boldsymbol{\mu}_{k_1}), \ldots, \tilde{\boldsymbol{y}}(\boldsymbol{\mu}_{k_w}) \in \mathbb{R}^n$ is in $\mathcal{O}((N + mn)w)$. The transformation described in Lemma 3.4 relies on a QR decomposition that requires $\mathcal{O}(mw^2)$ operations [24, section 5.2.9]. The matrices of the eigenproblem (3.11) have size $m \times m$, the left-hand side matrix is symmetric, and the right-hand side matrix is symmetric positive definite. Therefore, the costs to obtain the generalized eigenvector are $\mathcal{O}(m^3)$ [24,

p. 391]. Since usually $m \ll N, m_s \ll N$ as well as $w \ll N$, it is computationally feasible to adapt the DEIM basis with Algorithm 1 during the online phase.

3.5. Remarks on the online basis update. At each step $k = 1, \ldots, M'$, new sampling points are generated. Changing the sampling points at each step prevents overfitting of the basis update to only a few components of the nonlinear function.

The greedy algorithm to select the DEIM interpolation points takes the growth of the L_2 error of the DEIM approximation into account [12, Algorithm 1, Lemma 3.2]. In contrast, the sampling points in our adaptivity scheme are selected randomly, without such an objective; however, the sampling points serve a different purpose than the DEIM interpolation points. First, sampling points are only used for the adaptation, whereas ultimately the DEIM interpolation points are used for the DEIM approximation; see the adaptivity scheme outlined at the beginning of section 3. Second, a new set of sampling points is generated at each adaptivity step, and therefore a poor selection of sampling points is quickly replaced. Third, many adaptivity steps are performed. An update that targets the residual at a poor selection of sampling points is therefore compensated for quickly. Fourth, the adaptation is performed online, and therefore a computationally expensive algorithm to select the sampling points is often infeasible. The numerical experiments in section 5 demonstrate that the effect of the random sampling on the accuracy is small compared to the gain achieved by the adaptivity.

Theorem 3.6 guarantees that the update $\boldsymbol{\alpha}_k \boldsymbol{\beta}_k^T$ is a global optimum of the minimization problem (3.6); however, the theorem does not state that the update is unique. If multiple linearly independent eigenvectors corresponding to the largest eigenvalue exist, all of them lead to the same residual (3.5) and thus lead to an optimal update with respect to (3.6).

The DEIM basis computed in the offline phase from the SVD of the nonlinear snapshots contains orthonormal basis vectors. After adapting the basis, the orthonormality of the basis vectors is lost. Therefore, to obtain a numerically stable method, it is necessary to keep the condition number of the basis matrix $\boldsymbol{U}_k$ low, e.g., by orthogonalizing the basis matrix $\boldsymbol{U}_k$ after several updates or by monitoring the condition number and orthogonalizing if a threshold is exceeded. Note that monitoring the condition number can be achieved with an SVD of the basis matrix $\boldsymbol{U}_k$, with costs $\mathcal{O}(Nm^2 + m^3)$, and thus this is feasible in the online phase. Our numerical results in section 5 show, however, that even many adaptations lead to only a slight increase of the condition number, and therefore we do not orthogonalize the basis matrix in the following. Furthermore, in our numerical examples, the same window size is used for problems that differ with respect to degrees of freedom and type of nonlinearity, which shows that fine-tuning the window size to the current problem at hand is often unnecessary.

4. Online interpolation points adaptivity. After having adapted the DEIM basis at step k, we also adapt the DEIM interpolation points. The standard DEIM greedy method is too computationally expensive to apply in the online phase, because it recomputes all m interpolation points. We propose an adaptivity strategy that exploits that it is often unnecessary to change all m interpolation points after a single rank-one update to the DEIM basis. Section 4.1 describes a strategy that selects at each step $k = 1, \ldots, M'$ one interpolation point to be replaced by a new interpolation point. The corresponding efficient computational procedure is presented in section 4.2.

4.1. Adapting the interpolation points. Let k be the current step with the adapted DEIM basis $\boldsymbol{U}_k$, and let $\boldsymbol{U}_{k-1}$ be the DEIM basis and $\boldsymbol{P}_{k-1}$ be the DEIM

interpolation points matrix of the previous step. Further let $\{p_1^{k-1}, \ldots, p_m^{k-1}\} \subset \{1, \ldots, N\}$ be the interpolation points corresponding to $\boldsymbol{P}_{k-1}$. We adapt the interpolation points by replacing the ith interpolation point p_i^{k-1} by a new interpolation point $p_i^k \in \{1, \ldots, N\} \setminus \{p_1^{k-1}, \ldots, p_m^{k-1}\}$. We therefore construct the adapted interpolation points matrix

$$\boldsymbol{P}_k = \boldsymbol{P}_{k-1} + (\boldsymbol{e}_{p_i^k} - \boldsymbol{e}_{p_i^{k-1}})\boldsymbol{d}_i^T \tag{4.1}$$

from the interpolation points matrix $\boldsymbol{P}_{k-1}$ with the rank-one update $(\boldsymbol{e}_{p_i^k} - \boldsymbol{e}_{p_i^{k-1}})\boldsymbol{d}_i^T \in \mathbb{R}^{N \times m}$. The N-dimensional vectors $\boldsymbol{e}_{p_i^k} \in \{0,1\}^N$ and $\boldsymbol{e}_{p_i^{k-1}} \in \{0,1\}^N$ are the p_i^kth and p_i^{k-1}th canonical unit vectors, respectively. The vector $\boldsymbol{d}_i \in \{0,1\}^m$ is the ith canonical unit vector of dimension m. The update $(\boldsymbol{e}_{p_i^k} - \boldsymbol{e}_{p_i^{k-1}})\boldsymbol{d}_i^T$ replaces the ith column $\boldsymbol{e}_{p_i^{k-1}}$ of $\boldsymbol{P}_{k-1}$ with $\boldsymbol{e}_{p_i^k}$ and thus replaces point p_i^{k-1} with point p_i^k.

Each column of $\boldsymbol{P}_{k-1}$, and thus each interpolation point, is selected with respect to the basis vector in the corresponding column in $\boldsymbol{U}_{k-1}$. The standard DEIM greedy procedure ensures this for $\boldsymbol{P}_0$ built in the offline phase [12, Algorithm 1], and the following adaptivity procedure ensures this recursively for the adapted DEIM interpolation points matrix $\boldsymbol{P}_{k-1}$. We replace the point p_i^{k-1} corresponding to the basis vector that was rotated most by the basis update from $\boldsymbol{U}_{k-1}$ to $\boldsymbol{U}_k$. We therefore first compute the dot product between the previous and the adapted basis vectors

$$\operatorname{diag}(\boldsymbol{U}_k^T \boldsymbol{U}_{k-1})\,. \tag{4.2}$$

If the dot product of two normalized basis vectors is one, then they are colinear and the adapted basis vector has not been rotated with respect to the previous vector at step $k-1$. If it is zero, they are orthogonal. We select the basis vector $\boldsymbol{u}_k \in \mathbb{R}^N$ of $\boldsymbol{U}_k$ that corresponds to the component of (4.2) with the lowest absolute value. Note that after the adaptation, the adapted basis vectors are not necessarily normalized and therefore need to be normalized before (4.2) is computed. The new interpolation point p_i^k is derived from $\boldsymbol{u}_k$ following the standard DEIM greedy procedure. It then replaces the interpolation point p_i^{k-1}. All other interpolation points are unchanged, i.e., $p_j^k = p_j^{k-1}$ for all $j = 1, \ldots, m$ with $j \neq i$.

4.2. Computational procedure. Algorithm 2 summarizes the computational procedure to adapt the DEIM interpolation points at step k. The inputs are the DEIM basis $\boldsymbol{U}_{k-1}$ and the DEIM interpolation points matrix $\boldsymbol{P}_{k-1}$ as well as the adapted DEIM basis $\boldsymbol{U}_k$. The algorithm first selects the index i of the column of the basis vector that was rotated most by the basis update. This basis vector is denoted as $\boldsymbol{u}_k \in \mathbb{R}^N$. Then, the DEIM interpolant $(\hat{\boldsymbol{U}}_k, \hat{\boldsymbol{P}}_k)$ is built using available information from $\boldsymbol{U}_k$ and $\boldsymbol{P}_{k-1}$. The basis $\hat{\boldsymbol{U}}_k \in \mathbb{R}^{N \times (m-1)}$ contains all $m-1$ columns of $\boldsymbol{U}_k$ except for the ith column. The matrix $\hat{\boldsymbol{P}}_k$ is assembled similarly from the interpolation points matrix $\boldsymbol{P}_{k-1}$. The DEIM approximation $\hat{\boldsymbol{u}}_k$ of $\boldsymbol{u}_k$ is constructed with the interpolant $(\hat{\boldsymbol{U}}_k, \hat{\boldsymbol{P}}_k)$. The interpolation point p_i^k is set to the component with the largest absolute residual $|\hat{\boldsymbol{u}}_k - \boldsymbol{u}_k| \in \mathbb{R}^N$. If p_i^k is not already an interpolation point, the DEIM interpolation points matrix $\boldsymbol{P}_k$ is constructed with the update (4.1), else $\boldsymbol{P}_k = \boldsymbol{P}_{k-1}$.

The runtime costs of Algorithm 2 scale linearly with the number of degrees of freedom N of the full-order system (2.1). The dot product between the normalized basis vectors is computed in $\mathcal{O}(Nm)$. The matrices $\hat{\boldsymbol{U}}_k$ and $\hat{\boldsymbol{P}}_k$ are assembled in $\mathcal{O}(Nm)$, and the DEIM approximation $\hat{\boldsymbol{u}}_k$ is derived in $\mathcal{O}(m^3)$. Computing the

Algorithm 2. Adapt interpolation points after basis update.

procedure adaptInterpPoints($\boldsymbol{U}_{k-1}$, $\boldsymbol{P}_{k-1}$, $\boldsymbol{U}_k$)
  Normalize basis vectors
  Take dot product diag($\boldsymbol{U}_k^T \boldsymbol{U}_{k-1}$) between the basis vectors of $\boldsymbol{U}_k$ and $\boldsymbol{U}_{k-1}$
  Find the index i of the pair of basis vectors which are nearest to orthogonal
  Let $\boldsymbol{u}_k \in \mathbb{R}^N$ be the ith column of the adapted basis $\boldsymbol{U}_k$
  Store all other $m-1$ columns of $\boldsymbol{U}_k$ in $\hat{\boldsymbol{U}}_k \in \mathbb{R}^{N\times(m-1)}$
  Store all other $m-1$ columns of $\boldsymbol{P}_{k-1}$ in $\hat{\boldsymbol{P}}_k \in \mathbb{R}^{N\times(m-1)}$
  Approximate $\boldsymbol{u}_k$ with DEIM interpolant $(\hat{\boldsymbol{U}}_k, \hat{\boldsymbol{P}}_k)$ as

$$\hat{\boldsymbol{u}}_k = \hat{\boldsymbol{U}}_k (\hat{\boldsymbol{P}}_k^T \hat{\boldsymbol{U}}_k)^{-1} \hat{\boldsymbol{P}}_k^T \boldsymbol{u}_k$$

  Compute residual $|\hat{\boldsymbol{u}}_k - \boldsymbol{u}_k| \in \mathbb{R}^N$
  Let p_i^k be the index of the largest component of the residual
  if $\boldsymbol{e}_{p_i^k}$ is not a column of $\boldsymbol{P}_{k-1}$, **then**
    Replace interpolation point p_i^{k-1} of the ith basis vector with p_i^k
    Assemble updated interpolation points matrix $\boldsymbol{P}_k$ with (4.1)
  else
    Do not change interpolation points and set $\boldsymbol{P}_k = \boldsymbol{P}_{k-1}$
  end if
return $\boldsymbol{P}_k$
end procedure

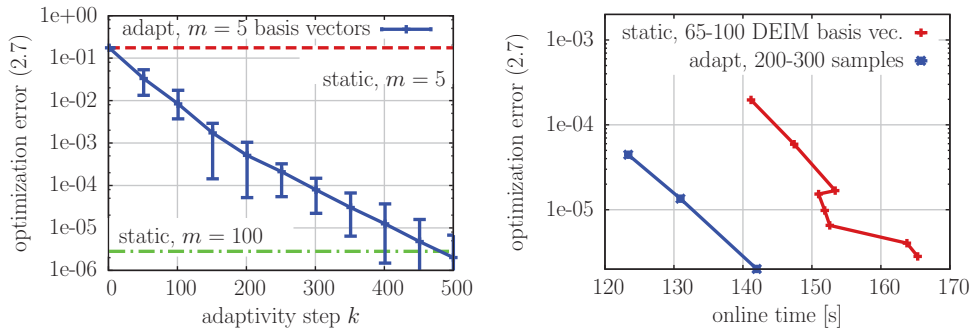
residual $|\hat{\boldsymbol{u}}_k - \boldsymbol{u}_k| \in \mathbb{R}^N$ and finding the component with the largest residual has linear runtime costs in N. Assembling the adapted interpolation points matrix $\boldsymbol{P}_k$ is in $\mathcal{O}(N)$ because only one column has to be replaced.

5. Numerical results. We present numerical experiments to demonstrate our nonlinear model reduction approach based on online adaptive DEIM interpolants. The optimization problem introduced in section 2.3 is revisited in section 5.1, and the time-dependent FitzHugh–Nagumo system is discussed in section 5.2. Section 5.3 applies online adaptive DEIM interpolants to a simplified model of a combustor governed by a reacting flow of a premixed H_2-Air flame. The reduced system is evaluated at a large number of parameters to predict the expected failure of the combustor.

All of the following experiments and runtime measurements were performed on compute nodes with Intel Xeon E5-1620 and 32GB RAM on a single core using a MATLAB implementation. The nonlinear functions in this section can all be evaluated componentwise; see section 2.1.

5.1. Synthetic optimization problem. In section 2.4, we introduced the function $\boldsymbol{g} : \mathcal{D} \to \mathbb{R}^N$ and the parameter $\boldsymbol{\mu}^* \in \mathcal{D}$, which is the maximum of the objective $\mathbf{1}_N^T \boldsymbol{g}(\boldsymbol{\mu})$. We built the DEIM interpolant $\tilde{\boldsymbol{g}}$ of $\boldsymbol{g}$ and searched for the maximum $\tilde{\boldsymbol{\mu}}^* \in \mathcal{D}$ of the approximate objective $\mathbf{1}_n^T \tilde{\boldsymbol{g}}(\boldsymbol{\mu})$ to approximate $\boldsymbol{\mu}^*$. The reported optimization error (2.7) showed that an accuracy of about 10^{-2} is achieved by a DEIM interpolant with 20 DEIM basis vectors built from nonlinear snapshots corresponding to parameters at an equidistant 20×20 grid in the parameter domain $\mathcal{D}$.

Let us now consider online adaptive DEIM interpolants of $\boldsymbol{g}$. We set the window size to $w = 50$ and the number of sampling points to $m_s = 300$, and we do not orthogonalize the DEIM basis matrix $\boldsymbol{U}_k$ after the adaptation. Note that we need a

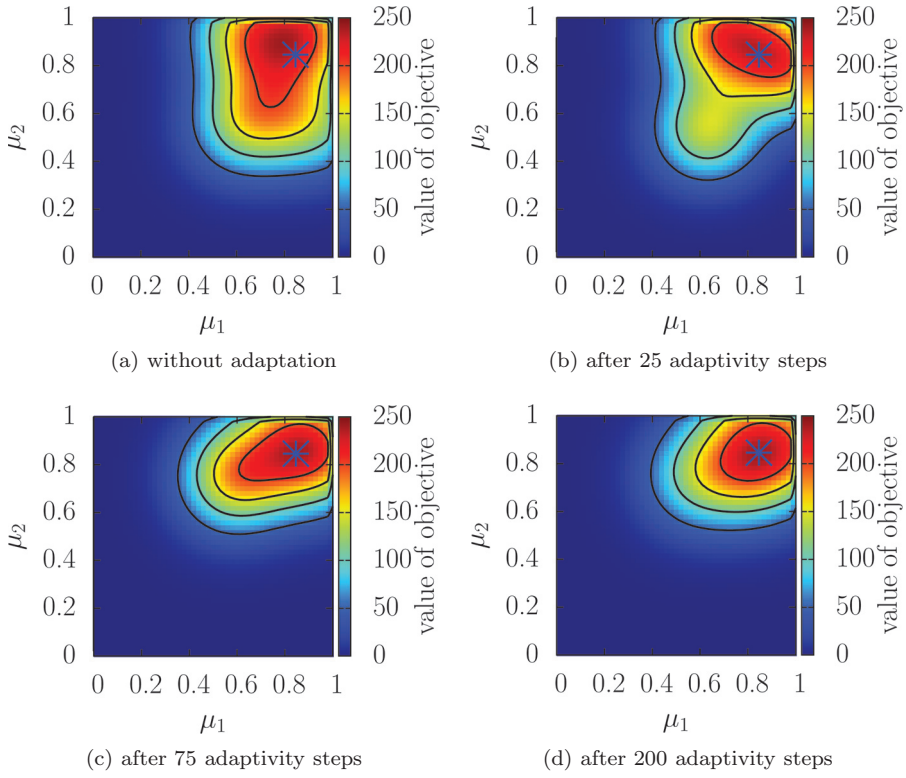


(a) error $\|\boldsymbol{\mu}^* - \tilde{\boldsymbol{\mu}}^*\|_2 / \|\boldsymbol{\mu}^*\|_2$ (b) runtime (optimization + online adaptivity)

FIG. 3. *Optimization problem. The results in* (a) *show that our online adaptive DEIM interpolant with $m = 5$ DEIM basis vectors (solid, blue) improves the optimization error of the static DEIM interpolant with $m = 5$ basis vectors (dashed, red) by five orders of magnitude. The online adaptive interpolant achieves a similar accuracy as a static interpolant with $m = 100$ basis vectors (dash-dotted, green). The reported error of the online adaptive interpolant is the mean error over ten runs. The bars indicate the minimal and the maximal error over these ten runs. The plot in* (b) *shows that for an accuracy of about 10^{-6}, the online runtime of the optimization method with the online adaptive DEIM interpolant is lower than the runtime with the static interpolant.*

large number of samples m_s here because the function $\boldsymbol{g}$ has one sharp peak and is zero elsewhere in the spatial domain Ω. Note that $m_s = 300$ is still less than the number of degrees of freedom $N = 1600$. The optimization with Nelder–Mead consists of $k = 1, \ldots, 500$ steps. In each step k, the intermediate result $\boldsymbol{\mu}_{M+k}$ found by Nelder–Mead is used to initiate the adaptation of the DEIM interpolant following sections 3 and 4. The adapted DEIM interpolant is then used in the subsequent iteration of the Nelder–Mead optimization method. We compare the optimization error (2.7) obtained with the adaptive interpolants to the error of the static interpolants. The static interpolants are constructed in the offline phase and are not adapted during the online phase; see section 2.3. The online adaptive DEIM interpolants are initially constructed in the offline phase from the same set of snapshots as the static interpolants.

Figure 3(a) summarizes the optimization error (2.7) corresponding to the static and the online adaptive DEIM interpolants. To account for the random selection of the sampling points, the optimization based on the adaptive DEIM interpolant is repeated ten times, and the mean, the minimal, and the maximal optimization error over these ten runs are reported. After 500 updates, the online adaptive DEIM interpolant with five DEIM basis vectors achieves a mean error below 10^{-6}. This is an improvement of five orders of magnitude compared to the static DEIM interpolant with five DEIM basis vectors. The static DEIM interpolant requires 100 basis vectors for a comparable accuracy. The spread of the optimization error due to the random sampling is small if considered relative to the significant gain achieved by the online adaptation here. In Figure 3(b), we report the online runtime of the Nelder–Mead optimization method. For the static DEIM interpolant, the runtime for $65, 70, 75, 80, 85, 90, 95, 100$ DEIM basis vectors is reported. For the adaptive interpolant, the runtime corresponding to $200, 250$, and 300 samples is reported. The runtime of the optimization procedure with the static DEIM interpolants quickly increases with the number of DEIM basis vectors m. The optimization procedure based on our proposed online adaptive DEIM interpolants is fastest and leads to the most accurate result in this example. Figure 4 shows the approximate objective $\mathbf{1}_n^T \tilde{\boldsymbol{g}}(\boldsymbol{\mu})$ with an online adaptive DEIM interpolant.



(a) without adaptation

(b) after 25 adaptivity steps

(c) after 75 adaptivity steps

(d) after 200 adaptivity steps

FIG. 4. *Optimization problem. The plots show the approximate objective function* $\mathbf{1}_n^T \tilde{\boldsymbol{g}}(\boldsymbol{\mu})$ *with the online adaptive DEIM interpolant* $\tilde{\boldsymbol{g}}$ *with* $m_s = 300$ *sample points and* $m = 5$ *DEIM basis vectors. The interpolant adapts to the function behavior near the optimal parameter* $\boldsymbol{\mu}^* \in \mathcal{D}$ *marked by the cross.*

The blue cross indicates the optimum $\boldsymbol{\mu}^*$. After 75 updates, the online adaptive interpolant approximates the nonlinear function $\boldsymbol{g}$ well near the optimum $\boldsymbol{\mu}^*$.

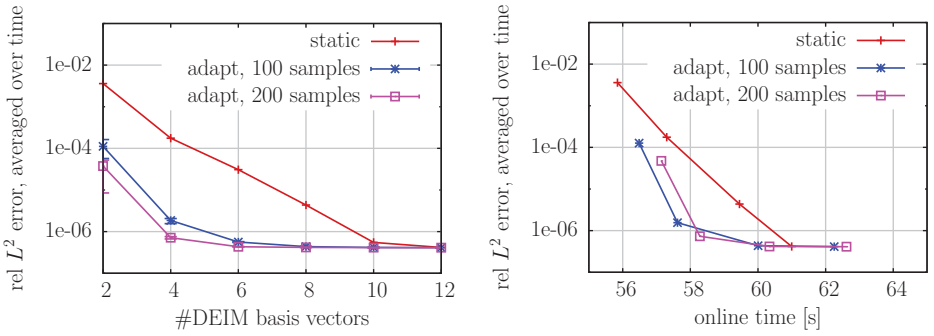
5.2. Time-dependent FitzHugh–Nagumo system. We consider the FitzHugh–Nagumo system to demonstrate our online adaptive approach on a time-dependent problem. The FitzHugh–Nagumo system was used as a benchmark model in the original DEIM paper [12]. It is a one-dimensional time-dependent nonlinear system of PDEs modeling the electrical activity in a neuron. We closely follow [12] and define the FitzHugh–Nagumo system as

$$\begin{aligned} \epsilon \partial_t y^v(x,t) &= \epsilon^2 \partial_x^2 y^v(x,t) + f(y^v(x,t)) - y^w(x,t) + c\,, \\ \partial_t y^w(x,t) &= b y^v(x,t) - \gamma y^w(x,t) + c \end{aligned}$$

with spatial coordinate $x \in \Omega = [0, L] \subset \mathbb{R}$, length $L \in \mathbb{R}$, time $t \in [0, T] \subset \mathbb{R}$, state functions $y^v, y^w : \Omega \times [0, T] \to \mathbb{R}$, the second derivative operator ∂_x^2 in the spatial coordinate x, and the first derivative operator ∂_t in time t. The state function y^v and y^w are voltage and recovery of voltage, respectively. The nonlinear function $f : \mathbb{R} \to \mathbb{R}$ is $f(y^v) = y^v(y^v - 0.1)(1 - y^v)$ and the initial and boundary conditions are

$$\begin{aligned} y^v(x,0) &= 0, & y^w(x,0) &= 0\,, & x &\in [0, L]\,, \\ \partial_x y^v(0,t) &= -i_0(t)\,, & \partial_x y^v(L,t) &= 0, & t &\geq 0\,, \end{aligned}$$

with $i_0 : [0, T] \to \mathbb{R}$ and $i_0(t) = 50000 t^3 \exp(-15t)$. We further set $L = 1, \epsilon =$



(a) accuracy (b) online runtime

FIG. 5. *FitzHugh–Nagumo system. The plots show that adapting the DEIM interpolant of the FitzHugh–Nagumo reduced system at every* 200*th time step in the online phase improves the overall accuracy of the solution of the reduced system by up to two orders of magnitude. The online runtime is not dominated by the DEIM approximation here, and thus the online adaptive DEIM interpolants improve the runtime only slightly.*

$0.015, T = 1, b = 0.5, \gamma = 2$, and $c = 0.05$. More details on the implementation of the FitzHugh–Nagumo system, including a derivation of the linear and nonlinear operators, can be found in [11].

We discretize the FitzHugh–Nagumo system with finite differences on an equidistant grid with 1024 grid points in the spatial domain and with the forward Euler method at 10^6 equidistant time steps in the temporal domain. The full-order system with the state vector $\boldsymbol{y}(t) \in \mathbb{R}^N$ has $N = 2 \times 1024 = 2048$ degrees of freedom. We derive a POD basis $\boldsymbol{V} \in \mathbb{R}^{N \times n}$ and a POD-DEIM-Galerkin reduced system following [12]. The POD basis and the DEIM interpolant are built from $M = 1000$ snapshots, which are the solutions of the full-order system at every 1000th time step. The state vector $\boldsymbol{y}(t)$ is approximated by $\boldsymbol{V}\tilde{\boldsymbol{y}}(t)$, where $\tilde{\boldsymbol{y}}(t) \in \mathbb{R}^n$ is the solution of the reduced system.

We report the average of the relative L^2 error of the approximation $\boldsymbol{V}\tilde{\boldsymbol{y}}(t)$ at every 1000th time step but starting with time step 500. Thus, the error is measured at the time steps halfway between those at which the snapshots were taken. We again consider static and online adaptive DEIM interpolants. The static interpolants are built in the offline phase and are not adapted online. The online adaptive DEIM interpolants are adapted at every 200th time step. The window size is $w = 5$ to reduce the computational costs of the online adaptation. The DEIM basis matrix is not orthogonalized after an adaptation. We set the number of POD basis vectors to $n = 10$. The experiment is repeated ten times, and the mean, the minimal, and the maximal averaged relative L^2 errors over these ten runs are reported. The results in Figure 5(a) show that adapting the DEIM interpolant online improves the accuracy by up to two orders of magnitude compared to the static interpolant. The accuracy of the solution obtained with the adaptive reduced system is limited by the POD basis, which stays fixed during the online phase. The spread of the error due to the random selection of the sampling points is small. We report the online runtime of the forward Euler method for DEIM basis vectors $2, 4, 6$, and 8 in Figure 5(b). It can be seen that the runtimes corresponding to the static interpolants increase only slightly as the number of DEIM basis vectors is increased. This shows that, for this problem, the runtime is not dominated by the DEIM approximation and explains why online adaptivity improves the runtime only slightly here.

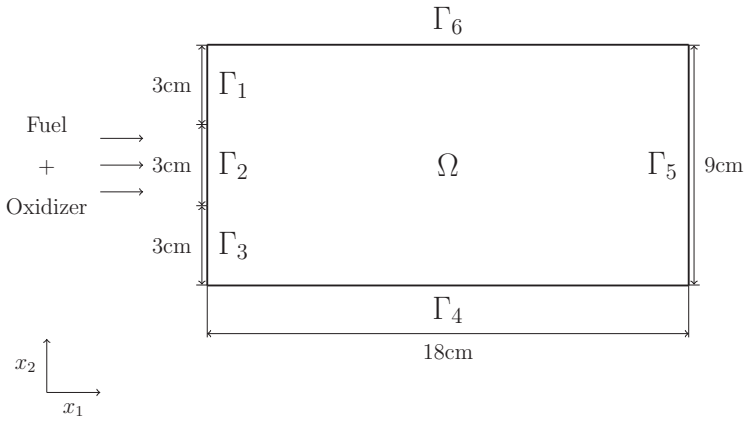


FIG. 6. *Combustor. The figure shows the geometry of the spatial domain of the combustor problem.*

5.3. Expected failure rate. In this section, we compute the expected failure rate of a combustor with Monte Carlo and importance sampling. We employ a POD-DEIM-Galerkin reduced system to speed up the computation. Reduced systems have been extensively studied for computing failure rates and rare event probabilities [6, 5, 30, 13]; however, we consider here POD-DEIM-Galerkin reduced systems based on online adaptive DEIM interpolants that are adapted while they are evaluated by the Monte Carlo method during the online phase. Online adaptivity is well-suited for computing failure rates because it adapts the reduced system to the failure boundaries where a high accuracy is required. Note that those regions are not known during the offline phase.

5.3.1. Combustor model. Our simplified model of a combustor is based on a steady premixed H_2-Air flame. The one-step reaction mechanism underlying the flame is

$$2H_2 + O_2 \to 2H_2O\,,$$

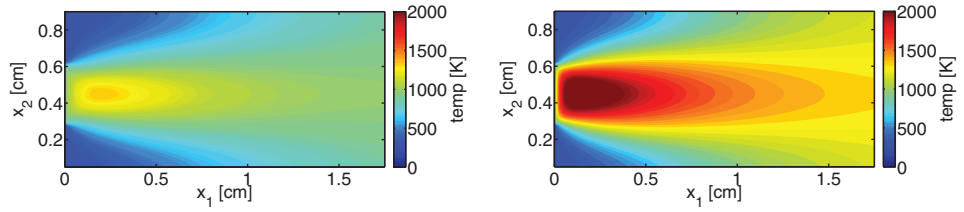
where H_2 is the fuel, O_2 is the oxidizer, and H_2O is the product. Let $\Omega \subset \mathbb{R}^2$ be the spatial domain with the geometry shown in Figure 6, and let $\mathcal{D} \subset \mathbb{R}^2$ be the parameter domain. The governing equation is a nonlinear advection-diffusion-reaction equation

$$\kappa \Delta y(\boldsymbol{\mu}) - \omega \nabla y(\boldsymbol{\mu}) + f(y(\boldsymbol{\mu}), \boldsymbol{\mu}) = 0\,, \tag{5.1}$$

where $\boldsymbol{\mu} \in \mathcal{D}$ is a parameter and where $y(\boldsymbol{\mu}) = [y_{H_2}, y_{O_2}, y_{H_2O}, T]^T$ contains the mass fractions of the species, H_2, O_2, and H_2O, and the temperature. The constant $\kappa = 2.0\text{cm}^2/\text{sec}$ is the molecular diffusivity and $\omega = 50\text{cm/sec}$ is the velocity in x_1 direction. The function $f(y(\boldsymbol{\mu}), \boldsymbol{\mu}) = [f_{H_2}(y(\boldsymbol{\mu}), \boldsymbol{\mu}), f_{O_2}(y(\boldsymbol{\mu}), \boldsymbol{\mu}), f_{H_2O}(y(\boldsymbol{\mu}), \boldsymbol{\mu}), f_T(y(\boldsymbol{\mu}), \boldsymbol{\mu})]^T$ is defined by its components

$$f_i(y(\boldsymbol{\mu}), \boldsymbol{\mu}) = -\nu_i \left(\frac{\eta_i}{\rho}\right) \left(\frac{\rho y_{H_2}}{\eta_{H_2}}\right)^2 \left(\frac{\rho y_{O_2}}{\eta_{O_2}}\right) \mu_1 \exp\left(-\frac{\mu_2}{RT}\right), \quad i = H_2, O_2, H_2O\,,$$
$$f_T(y(\boldsymbol{\mu}), \boldsymbol{\mu}) = Q f_{H_2O}(y(\boldsymbol{\mu}), \boldsymbol{\mu})\,.$$

The vector $\boldsymbol{\nu} = [2, 1, 2]^T \in \mathbb{N}^3$ is constant and derived from the reaction mechanism, $\rho = 1.39 \times 10^{-3}$ gr/cm^3 is the density of the mixture, $\boldsymbol{\eta} = [2.016, 31.9, 18]^T \in \mathbb{R}^3$ are the molecular weights in gr/mol, $R = 8.314472\text{J}/(\text{mol K})$ is the universal gas



(a) parameter $\boldsymbol{\mu} = [5.5 \times 10^{11}, 1.5 \times 10^3]$ (b) parameter $\boldsymbol{\mu} = [1.5 \times 10^{13}, 9.5 \times 10^3]$

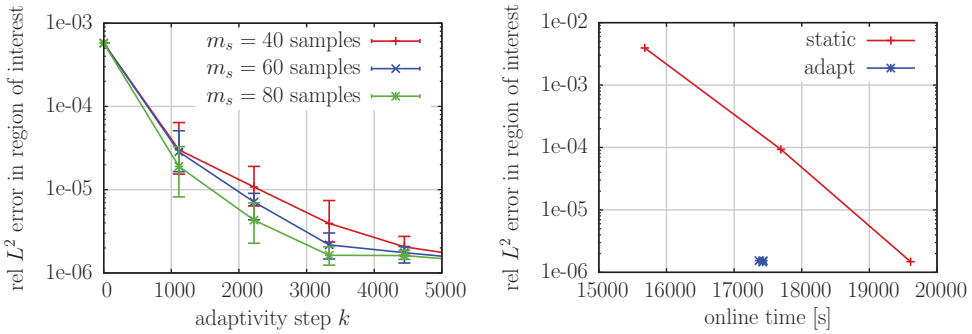
FIG. 7. *Combustor. The plots show the temperature in the spatial domain Ω for two parameters near the boundary of the parameter domain $\mathcal{D}$.*

constant, and Q = 9800K is the heat of reaction. The parameter $\boldsymbol{\mu} = [\mu_1, \mu_2] \in \mathcal{D}$ is in the domain $\mathcal{D} = [5.5 \times 10^{11}, 1.5 \times 10^{13}] \times [1.5 \times 10^3, 9.5 \times 10^3] \subset \mathbb{R}^2$, where μ_1 is the preexponential factor and μ_2 is the activation energy. With the notation introduced in Figure 6, we impose homogeneous Dirichlet boundary conditions on the mass fractions on Γ_1 and Γ_3, and homogeneous Neumann conditions on the temperature and mass fractions on Γ_4, Γ_5, and Γ_6. We further have Dirichlet boundary conditions on Γ_2 with $y_{\mathrm{H}_2} = 0.0282, y_{\mathrm{O}_2} = 0.2259, y_{\mathrm{H}_2\mathrm{O}} = 0, y_T = 950\mathrm{K}$ and on Γ_1, Γ_3 with $y_T = 300\mathrm{K}$.

The PDE (5.1) is discretized with finite differences on an equidistant 73×37 grid in the spatial domain Ω. The corresponding discrete system of nonlinear equations has $N = 10,804$ degrees of freedoms and is solved with the Newton method. The state vector $\boldsymbol{y}(\boldsymbol{\mu}) \in \mathbb{R}^N$ contains the mass fractions and temperature at the grid points. Figure 7 shows the temperature field for parameters $\boldsymbol{\mu} = [5.5 \times 10^{11}, 1.5 \times 10^3]$ and $\boldsymbol{\mu} = [1.5 \times 10^{13}, 9.5 \times 10^3]$. We follow the implementation details in [7], where a POD-DEIM-Galerkin reduced system [12] is derived.

5.3.2. Combustor: Region of interest. We demonstrate our online adaptivity procedures by adapting the DEIM interpolant to a region of interest $\mathcal{D}_{\mathrm{RoI}} = [2 \times 10^{12}, 6 \times 10^{12}] \times [5 \times 10^3, 7 \times 10^3] \subset \mathcal{D}$. We therefore first construct a POD basis with $n = 30$ basis vectors from snapshots corresponding to parameters at an 10×10 equidistant grid in the whole parameter domain $\mathcal{D}$ and derive a DEIM interpolant with the corresponding nonlinear snapshots and $m = 15$ DEIM basis vectors. We then adapt the DEIM interpolant online at state vectors corresponding to parameters drawn randomly with a uniform distribution from the region of interest $\mathcal{D}_{\mathrm{RoI}}$. After each adaptivity step, the reduced system is evaluated at parameters stemming from a 24×24 equidistant grid in $\mathcal{D}$ from which 3×3 parameters are in the region of interest $\mathcal{D}_{\mathrm{RoI}}$. We report the relative L^2 error of the temperature with respect to the full-order solution at the parameters in $\mathcal{D}_{\mathrm{RoI}}$.

Figure 8(a) shows the error of the temperature field computed with a POD-DEIM-Galerkin reduced system based on an online adaptive DEIM interpolant versus the adaptivity step. Reported is the mean L^2 error over ten runs, where the bars indicate the minimal and the maximal error. The window size is set to $w = 50$. Online adaptivity improves the accuracy of the solution of the POD-DEIM-Galerkin reduced system by about three orders of magnitude in the region of interest after 3000 updates. The random selection of the sampling point leads only to a small spread of the error here. The online runtimes corresponding to static and online adaptive DEIM interpolants are reported in Figure 8(b). The static interpolants are built in the offline phase from the same set of nonlinear snapshots as the adaptive interpolants but they are not changed in the online phase. The reported runtimes are for 10,000



(a) online adaptive DEIM interpolant (b) error w.r.t. online runtime

FIG. 8. *Combustor. The figure in* (a) *shows that adapting the DEIM interpolant to the region of interest $\mathcal{D}_{RoI}$ improves the relative L^2 error of solutions with parameters in $\mathcal{D}_{RoI}$ by up to three orders of magnitude. The plot in* (b) *shows that our online adaptive approach is more efficient than the static interpolant with respect to the online runtime.*

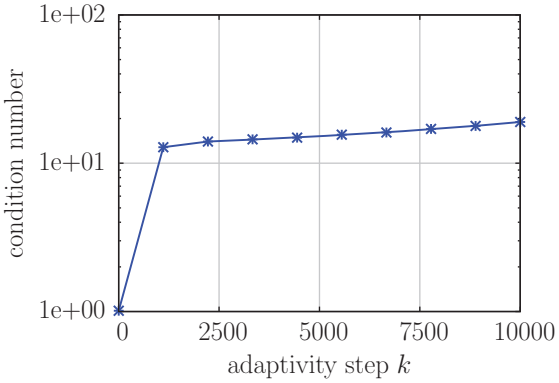
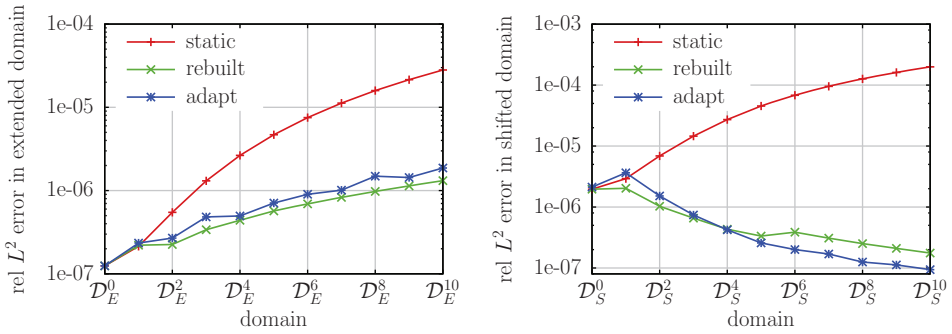


FIG. 9. *Combustor. The condition number of the basis matrix increases only slightly after* 10,000 *adaptivity steps. It is therefore not necessary to orthogonalize the basis online in this example.*

online steps. In the case of a static DEIM interpolant, each step consists of solving the POD-DEIM-Galerkin reduced system for a given parameter and computing the error. In the case of adaptivity, in addition to solving the POD-DEIM-Galerkin reduced system and computing the error, the DEIM interpolant is adapted. For the static DEIM interpolants, the runtimes are reported for $10, 20$, and 30 DEIM basis vectors. For the online adaptive DEIM interpolants, the number of DEIM basis vectors $m = 15$ is fixed, but the number of samples m_s is set to $40, 60$, and 80. Overall, the reduced systems based on the online adaptive DEIM interpolants lead to the lowest online runtime with respect to the L^2 error in the region of interest $\mathcal{D}_{\text{RoI}}$.

After an online basis update, the orthogonality of the basis vectors is lost. Figure 9 shows that even after 10,000 updates, the condition number of the DEIM basis matrix is still low, and thus it is not necessary to orthogonalize the basis during the online adaptation here. Note that the eigenproblem (3.11), from which the updates are derived, is well-conditioned; see Lemma 3.5.

5.3.3. Combustor: Extended and shifted parameter domains. We now consider an online adaptive reduced system of the combustor problem that is built



(a) extended parameter domains $\mathcal{D}_E^0, \ldots, \mathcal{D}_E^{10}$ (b) shifted parameter domains $\mathcal{D}_S^0, \ldots, \mathcal{D}_S^{10}$

FIG. 10. *Combustor. Online adaptive reduced systems provide valid approximations of the full-order solutions also for parameters outside of the domain $\mathcal{D}_{offline}$ for which they were initially built. Accuracy improvements over static reduced systems by up to three orders of magnitude are achieved.*

from snapshots with parameters in a domain $\mathcal{D}_{\text{offline}} \subset \mathcal{D}$ in the offline phase but which is then evaluated at parameters outside of $\mathcal{D}_{\text{offline}}$ in the online phase. We set $\mathcal{D}_{\text{offline}} = [2 \times 10^{12}, 6 \times 10^{12}] \times [5 \times 10^3, 7 \times 10^3] \subset \mathcal{D}$ and build a POD-DEIM-Galerkin reduced system from snapshots with parameters coinciding with an equidistant 10×10 grid in $\mathcal{D}_{\text{offline}}$. The number of POD basis vectors is set to $n = 30$. In the case of online adaptive DEIM interpolants, the window size is $w = 50$, and the number of samples is $m_s = 60$; see section 5.3.2.

Consider now the online phase, where the reduced system is used to approximate the full-order solutions with parameters at the equidistant 24×24 grid in the domains

$$\mathcal{D}_E^i = \left[2 \times 10^{12}, 6 \times 10^{12} + i\delta\mu_1\right] \times \left[5 \times 10^3, 7 \times 10^3 + i\delta\mu_2\right]$$

with $\boldsymbol{\delta\mu} = [9 \times 10^{11}, 2.5 \times 10^2]^T \in \mathbb{R}^2$ and $i = 0, \ldots, 10$. At each step $i = 0, \ldots, 10$, the domain is equidistantly extended. Figure 10(a) reports the relative L^2 error of the temperature field obtained with a static, a rebuilt, and an online adaptive reduced system with $m = 15$ DEIM basis vectors. The static interpolant is fixed and is not changed in the online phase. For the rebuilt interpolant, a DEIM basis is constructed from snapshots corresponding to parameters at an equidistant 10×10 grid in the domain $\mathcal{D}_E^i$ in each step $i = 0, \ldots, 10$. The online adaptive reduced system is adapted 5000 times at step i with respect to the domain $\mathcal{D}_E^i$; see section 5.3.2. The results show that the static reduced system quickly fails to provide valid approximations of the solutions of the full-order system as the domain is extended. In contrast, the online adaptive approach is able to capture the behavior of the full-order system also in the domains $\mathcal{D}_E^1, \ldots, \mathcal{D}_E^{10}$ that cover regions outside of $\mathcal{D}_{\text{offline}}$. An accuracy improvement by about one order of magnitude is achieved. This is about the same accuracy improvement that is achieved with the interpolant that is rebuilt in each step; however, our online adaptive interpolant avoids the additional costs of rebuilding from scratch. Note that the full-order system becomes harder to approximate as the domain is increased, and thus also the errors corresponding to the online adaptive and the rebuilt reduced system grow with the size of the domain.

Figure 10(b) shows the accuracy results for the static, the rebuilt, and the online adaptive reduced system if the parameter domain is shifted instead of extended. The shifted domains are

$$\mathcal{D}_S^i = \left[2 \times 10^{12} + i\delta\mu_1, 6 \times 10^{12} + i\delta\mu_1\right] \times \left[5 \times 10^3 + i\delta\mu_2, 7 \times 10^3 + i\delta\mu_2\right]$$

for $i = 0, \ldots, 10$. The number of DEIM basis vectors is $m = 10$. The online adaptive reduced system provides approximations that are up to three orders of magnitude more accurate than the approximations obtained by the static system. The adaptive interpolant achieves about the same accuracy as the rebuilt interpolant again. Note that the full-order system becomes simpler to approximate as the domain is shifted toward the upper-right corner of the parameter domain $\mathcal{D}$, and thus the errors corresponding to the online adaptive and the rebuilt system decrease.

The rebuilt and the online adaptive DEIM interpolant achieve a similar accuracy in Figures 10(a) and 10(b). In Figure 10(b), the online adaptive DEIM interpolant achieves a slightly higher accuracy than the rebuilt interpolant. This underlines that rebuilding the interpolant from scratch, from snapshots corresponding to the shifted domain, does not necessarily lead to an optimal DEIM interpolant with respect to accuracy. For the experiment presented in Figure 10(a), the online adaptive interpolant performs slightly worse than the rebuilt interpolant. Overall, the differences between the rebuilt and the online adaptive interpolant are small here, almost insignificant, and thus a more extensive study is necessary for a general comparison of rebuilt and online adaptive DEIM interpolants. Also note that other approaches based on rebuilding the DEIM interpolant might be feasible in certain situations. For example, the DEIM bases could be enriched with new basis vectors at each adaptivity step; however, note also that this would lead to a different form of adaptivity than what we consider here, because we keep the number of DEIM basis vectors fixed in the online phase.

5.3.4. Combustor: Expected failure rate. We now compute the expected failure rate of the combustor modeled by the advection-diffusion-reaction equation of section 5.3.1. We assume that the combustor fails if the maximum temperature exceeds 2290K in the spatial domain Ω. This is a value near the maximum temperature obtained with the parameters in the domain $\mathcal{D}$. We then define the random variable

$$T = \begin{cases} 1\,, \text{ if temperature } > 2290\text{K}\,, \\ 0\,, \text{else}\,, \end{cases}$$

where we assume that the parameters $\boldsymbol{\mu} \in \mathcal{D}$ are drawn from a normal distribution with mean

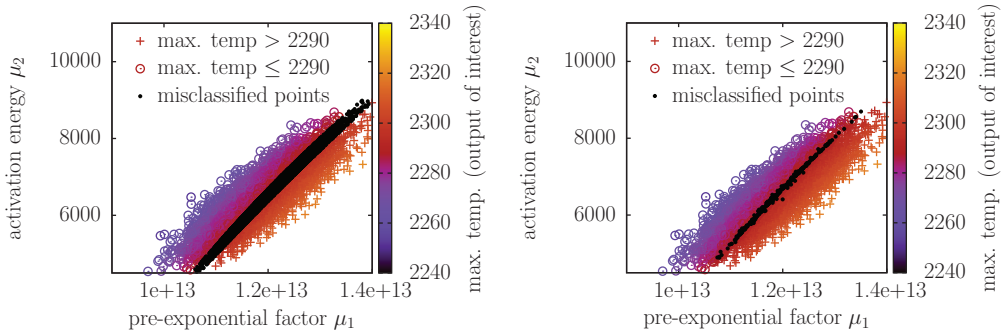
$$[8.67 \times 10^{12}, 5.60 \times 10^{3}] \in \mathcal{D} \tag{5.2}$$

and covariance matrix

$$\begin{bmatrix} 2.08 \times 10^{24} & 8.67 \times 10^{14} \\ 8.67 \times 10^{14} & 6.40 \times 10^{5} \end{bmatrix}. \tag{5.3}$$

Drawing the parameters from the given normal distribution leads to solutions with a maximum temperature near 2290K. The indicator variable T evaluates to one if a failure occurs, and to 0 else. We are therefore interested in the expected failure rate $\mathbb{E}[T]$.

We construct a POD-DEIM-Galerkin reduced system with $n = 10$ POD basis vectors and $m = 4$ DEIM basis vectors from the $M = 100$ snapshots of the full-order system with parameters stemming from an 10×10 equidistant grid in $\mathcal{D}$. We solve the reduced system instead of the full-order system to speed up the computation. Note that we use fewer POD and DEIM basis vectors here than in sections 5.3.2



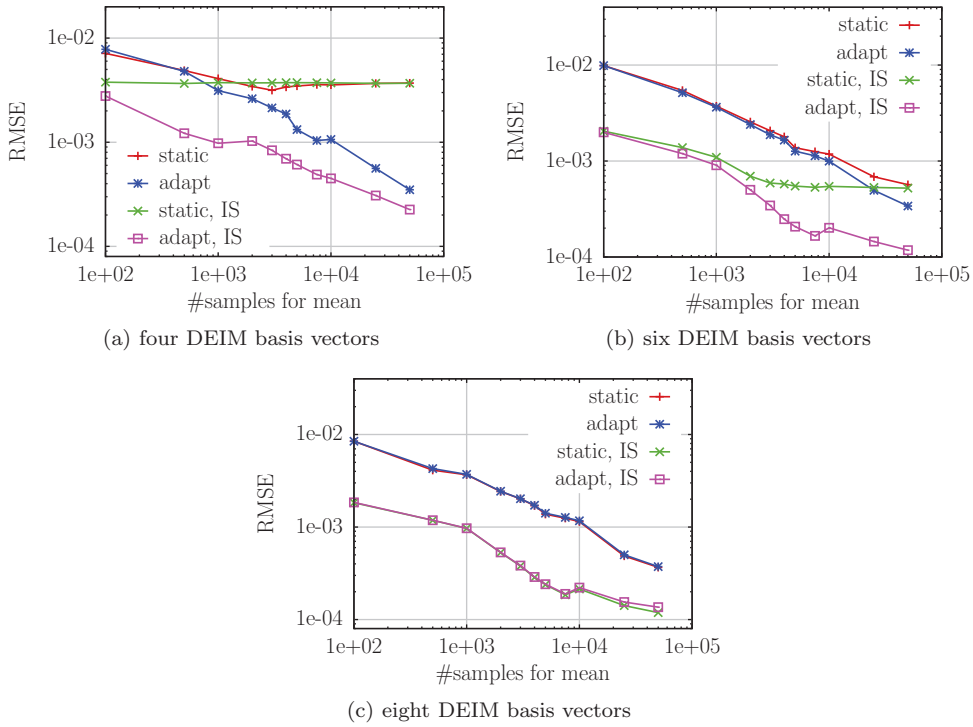
(a) static DEIM interpolant (b) online adaptive DEIM interpolant

FIG. 11. *Combustor. With the reduced system based on the static DEIM interpolant* 11,761 *out of* 50,000 *data points are misclassified. Online adaptivity reduces the number of misclassified points to* 326.

and 5.3.3 to reduce the online runtime of the up to one million reduced system solves. The failure rate $\mathbb{E}[T]$ is computed with Monte Carlo and with Monte Carlo enhanced by importance sampling. The biasing distribution of the importance sampling is obtained in a preprocessing step by evaluating the reduced system at a large number of parameters and then fitting a normal distribution to the parameters that led to a temperature above 2280K.

In Figure 11, we plot parameters drawn from the normal distribution with mean (5.2) and covariance matrix (5.3) and color them according to the maximum temperature estimated with the reduced system. The black dots indicate misclassified parameters, i.e., parameters for that the reduced system predicts a failure but the full-order system does not, and vice versa. The reduced system with a static DEIM interpolant with four DEIM basis vectors leads to 11,761 misclassified parameters out of 50,000 overall parameters. This is a misclassification rate of about 23 percent.

Let us now consider a reduced system with an online adaptive DEIM interpolant. We adapt the interpolant after every 25th evaluation. The number of samples is set again to $m_s = 60$, and the window size is $w = 50$. The misclassification rate of 23 percent obtained with the static interpolant drops to 0.65 percent if online adaptivity is employed. This shows that the DEIM interpolant quickly adapts to the nonlinear function evaluations corresponding to the parameters drawn from the specified distribution of $\boldsymbol{\mu}$. We evaluate the root-mean-square error (RMSE) of the expected failure rate predicted by the reduced system with a static DEIM interpolant and by a reduced system with an online adaptive interpolant. The RMSE is computed with respect to the expected rate obtained from one million evaluations of the full-order system. The averaged RMSEs over 30 runs are reported in Figure 12. In the case of the static DEIM interpolant, the reduced system cannot predict the expected rate if only four DEIM basis vectors are used. Even with six DEIM basis vectors, the RMSE for the static interpolant does not reduce below 10^{-3}. A static DEIM interpolant with eight DEIM basis vectors is necessary so that the number of samples used in the computation of the mean limits the RMSE rather than the accuracy of the POD-DEIM-Galerkin reduced system. In the case of the online adaptive DEIM interpolant, an RMSE between one and two orders of magnitude lower than with the static reduced system is achieved if the accuracy of the POD-DEIM-Galerkin reduced system limits the RMSE. This can be seen particularly well for Monte Carlo enhanced by importance sampling. Note that the same POD-DEIM-Galerkin reduced system



(a) four DEIM basis vectors

(b) six DEIM basis vectors

(c) eight DEIM basis vectors

FIG. 12. *Combustor. The plots in* (a) *and* (b) *show that the RMSE of the expected failure* $\mathbb{E}[T]$ *can be reduced by up to almost two orders of magnitude if the reduced system is adapted online. In* (c)*, the number of DEIM basis vectors is increased such that the RMSE is limited by the number of samples used for the mean computation rather than by the accuracy of the POD-DEIM-Galerkin reduced system. Therefore, improving the accuracy of the POD-DEIM-Galerkin reduced system by using the online adaptive DEIM interpolant cannot improve the RMSE. The expected failure is computed with Monte Carlo and Monte Carlo enhanced by importance sampling (IS).*

and the same setting for the adaptive DEIM interpolant as in section 5.3.2 is used. Therefore, the runtime comparison between the static and adaptive DEIM interpolant shown in Figure 8(b) applies here too.

6. Conclusions. We presented an online adaptive model reduction approach for nonlinear systems where the DEIM interpolant is adapted during the online phase. We have shown that our DEIM basis update is the optimal rank-one update with respect to the Frobenius norm. In our numerical experiments, the online adaptive DEIM interpolants improved the overall accuracy of the solutions of the reduced systems by orders of magnitude compared to the solutions of the corresponding reduced systems with static DEIM interpolants. Furthermore, our online adaptivity procedures have a linear runtime complexity in the number of degrees of freedom of the full-order system and thus are faster than rebuilding the DEIM interpolant from scratch.

Our adaptivity approach shows that it is unnecessary to solve the full-order system or to fully evaluate it in order to adapt the reduced system in the online phase. We directly adapt the reduced system with data that are generated by sampling the nonlinear function at a few additional components. A natural extension of our adaptivity approach would be to include other data such as the reduced state vector during Newton iterations or time stepping. Our approach is particularly useful for situations

in which it may be desired to solve the reduced system for parameters that lead to a solution outside the span of the snapshots generated in the offline phase. This is often the case in outer loop applications—optimization, inverse problem and control—where the ultimate solution path may be difficult to anticipate before the problem is solved. The adaptivity also offers a path to robustness of the reduced system by mitigating against a potential poor choice of snapshots in the initial construction of the reduced system.

A topic of future research is an indicator that helps to decide how many sampling points should be used. Depending on the computational costs of the indicator, it would then also become feasible to decide adaptively during the online phase how many sampling points should be used at the current adaptivity step. Also of interest could be replacing the random selection of the sampling points with a deterministic algorithm. In the offline phase, randomly selecting snapshots has been successfully replaced with greedy algorithms; see, e.g., [42, 40, 25, 8]. The sampling points in our adaptivity scheme, however, are repeatedly generated during the online phase, and therefore the challenge will be deriving an algorithm that is computationally feasible to be run many times in the online phase.

Acknowledgments. Several examples were computed on the computer clusters of the Munich Centre of Advanced Computing. The authors thank Florian Augustin, Ralf Zimmermann, and Alessio Spantini for very helpful discussions and comments.

REFERENCES

[1] D. Amsallem and C. Farhat, *An online method for interpolating linear parametric reduced-order models*, SIAM J. Sci. Comput., 33 (2011), pp. 2169–2198.

[2] D. Amsallem, M. Zahr, and C. Farhat, *Nonlinear model order reduction based on local reduced-order bases*, Internat. J. Numer. Methods Engrg., 92 (2012), pp. 891–916.

[3] P. Astrid, S. Weiland, K. Willcox, and T. Backx, *Missing point estimation in models described by proper orthogonal decomposition*, IEEE Trans. Automat. Control, 53 (2008), pp. 2237–2251.

[4] M. Barrault, Y. Maday, N.-C. Nguyen, and A. Patera, *An "empirical interpolation" method: Application to efficient reduced-basis discretization of partial differential equations*, C. R. Math., 339 (2004), pp. 667–672.

[5] A. Basudhar and S. Missoum, *An improved adaptive sampling scheme for the construction of explicit boundaries*, Struct. Multidiscip. Optim., 42 (2010), pp. 517–529.

[6] B. Bichon, M. Eldred, L. Swiler, S. Mahadevan, and J. McFarland, *Efficient global reliability analysis for nonlinear implicit performance functions*, AIAA J., 46 (2008), pp. 2459–2468.

[7] M. Buffoni and K. Willcox, *Projection-based model reduction for reacting flows*, in Proceedings of the 40th Fluid Dynamics Conference and Exhibit, Fluid Dynamics and Co-located Conferences, AIAA paper 2010-5008, 2010.

[8] T. Bui-Thanh, K. Willcox, and O. Ghattas, *Model reduction for large-scale systems with high-dimensional parametric input space*, SIAM J. Sci. Comput., 30 (2008), pp. 3270–3288.

[9] K. Carlberg, *Adaptive h-refinement for reduced-order models*, Internat. J. Numer. Methods Engrg., 102 (2015), pp. 1192–1210.

[10] K. Carlberg, C. Bou-Mosleh, and C. Farhat, *Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations*, Internat. J. Numer. Methods Engrg., 86 (2011), pp. 155–181.

[11] S. Chaturantabut, *Nonlinear Model Reduction via Discrete Empirical Interpolation*, Ph.D. thesis, Computational and Applied Mathematics, Rice University, Houston, 2011.

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

[13] P. Chen and A. Quarteroni, *Accurate and efficient evaluation of failure probability for partial different equations with random input data*, Comput. Methods Appl. Mech. Engrg., 267 (2013), pp. 233–260.

[14] P. Chen, A. Quarteroni, and G. Rozza, *A weighted reduced basis method for elliptic partial differential equations with random input data*, SIAM J. Numer. Anal., 51 (2013), pp. 3163–3185.

[15] P. Chen, A. Quarteroni, and G. Rozza, *A weighted empirical interpolation method: A priori convergence analysis and applications*, ESAIM Math. Model. Numer. Anal., 48 (2014), pp. 943–953.

[16] D. Ryckelynck, *A priori hyperreduction method: An adaptive approach*, J. Comput. Phys., 202 (2005), pp. 346–366.

[17] J. Degroote, J. Vierendeels, and K. Willcox, *Interpolation among reduced-order matrices to obtain parameterized models for design, optimization and probabilistic analysis*, Internat. J. Numer. Methods Fluids, 63 (2010), pp. 207–230.

[18] M. Dihlmann, M. Drohmann, and B. Haasdonk, *Model reduction of parametrized evolution problems using the reduced basis method with adaptive time-partitioning*, in Proceedings of the International Conference on Adaptive Modeling and Simulation, D. Aubry, P. Díez, B. Tie, and N. Parés, eds., 2011, pp. 156–167.

[19] J. Eftang and A. Patera, *Port reduction in parametrized component static condensation: Approximation and a posteriori error estimation*, Internat. J. Numer. Methods Engrg., 96 (2013), pp. 269–302.

[20] J. Eftang and B. Stamm, *Parameter multi-domain hp empirical interpolation*, Internat. J. Numer. Methods Engrg., 90 (2012), pp. 412–428.

[21] P. Feldmann and R. Freund, *Efficient linear circuit analysis by Padé approximation via the Lanczos process*, IEEE Trans. Computer-Aided Design Integrated Circuits Syst., 14 (1995), pp. 639–649.

[22] S. Friedland and A. Torokhti, *Generalized rank-constrained matrix approximations*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 656–659.

[23] K. Gallivan, E. Grimme, and P. Van Dooren, *Padé approximation of large-scale dynamic systems with Lanczos methods*, in Proceedings of the 33rd IEEE Conference on Decision and Control, 1994.

[24] G. H. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 2013.

[25] B. Haasdonk, *Convergence rates of the POD-Greedy method*, ESAIM Math. Model. Numer. Anal., 47 (2013), pp. 859–873.

[26] B. Haasdonk and M. Ohlberger, *Adaptive basis enrichment for the reduced basis method applied to finite volume schemes*, in Proceedings of the Fifth International Symposium on Finite Volumes for Complex Applications, R. Eymard and J.-M. Hérard, eds., 2008, pp. 471–479.

[27] S. Kaulmann and B. Haasdonk, *Online greedy reduced basis construction using dictionaries*, in Proceedings of the Sixth International Conference on Adaptive Modeling and Simulation (ADMOS 2013), J. P. Moitinho de Almeida, P. Díez, C. Tiago, and N. Parés, eds., 2013, pp. 365–376.

[28] J. Lagarias, J. Reeds, M. Wright, and P. Wright, *Convergence properties of the Nelder–Mead simplex method in low dimensions*, SIAM J. Optim., 9 (1998), pp. 112–147.

[29] O. Lass, *Reduced Order Modeling and Parameter Identification for Coupled Nonlinear PDE Systems*, Ph.D. thesis, University of Konstanz, Konstanz, Germany, 2014.

[30] J. Li and D. Xiu, *Evaluation of failure probability via surrogate models*, J. Comput. Phys., 229 (2010), pp. 8966–8980.

[31] Y. Maday and B. Stamm, *Locally adaptive greedy approximations for anisotropic parameter reduced basis spaces*, SIAM J. Sci. Comput., 35 (2013), pp. A2417–A2441.

[32] R. Markovinović and J. Jansen, *Accelerating iterative solution methods using reduced-order models as solution predictors*, Internat. J. Numer. Methods Engrg., 68 (2006), pp. 525–541.

[33] B. Moore, *Principal component analysis in linear systems: Controllability, observability, and model reduction*, IEEE Trans. Automat. Control, 26 (1981), pp. 17–32.

[34] H. Panzer, J. Mohring, R. Eid, and B. Lohmann, *Parametric model order reduction by matrix interpolation*, Automatisierungstechnik, 58 (2010), pp. 475–484.

[35] A. Paul-Dubois-Taine and D. Amsallem, *An adaptive and efficient greedy procedure for the optimal training of parametric reduced-order models*, Internat. J. Numer. Methods Engrg., 102 (2015), pp. 1262–1292.

[36] B. Peherstorfer, D. Butnaru, K. Willcox, and H.-J. Bungartz, *Localized discrete empirical interpolation method*, SIAM J. Sci. Comput., 36 (2014), pp. A168–A192.

[37] B. Peherstorfer, S. Zimmer, and H.-J. Bungartz, *Model reduction with the reduced basis method and sparse grids*, in Sparse Grids and Applications, J. Garcke and M. Griebel, eds., Lect. Notes Comput. Sci. Eng. 88, Springer, New York, 2013, pp. 223–242.

[38] L. Peng and K. Mohseni, *An online manifold learning approach for model reduction of dynamical systems*, SIAM J. Numer. Anal., 52 (2014), pp. 1928–1952.

[39] M.-L. Rapùn and J. Vega, *Reduced order models based on local POD plus Galerkin projection*, J. Comput. Phys., 229 (2010), pp. 3046–3063.

[40] G. Rozza, D. Huynh, and A. Patera, *Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations*, Arch. Comput. Methods Eng., 15 (2007), pp. 1–47.

[41] L. Sirovich, *Turbulence and the dynamics of coherent structures*, Quart. Appl. Math., 1987, pp. 561–571.

[42] K. Veroy and A. Patera, *Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: Rigorous reduced-basis a posteriori error bounds*, Internat. J. Numer. Methods Fluids, 47 (2005), pp. 773–788.

[43] K. Washabaugh, D. Amsallem, M. Zahr, and C. Farhat, *Nonlinear model reduction for CFD problems using local reduced-order bases*, in Proceedings of the 42nd Fluid Dynamics Conference and Exhibit, Fluid Dynamics and Co-located Conferences, AIAA paper 2012-2686, 2012.

[44] G. Weickum, M. Eldred, and K. Maute, *A multi-point reduced-order modeling approach of transient structural dynamics with application to robust design optimization*, Struct. Multidiscip. Optim., 38 (2009), pp. 599–611.

[45] M. Zahr and C. Farhat, *Progressive construction of a parametric reduced-order model for pde-constrained optimization*, Internat. J. Numer. Methods Engrg., 102 (2015), pp. 1111–1135.

[46] R. Zimmermann, *A locally parametrized reduced-order model for the linear frequency domain approach to time-accurate computational fluid dynamics*, SIAM J. Sci. Comput., 36 (2014), pp. B508–B537.