Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Adaptive Basis Construction and Improved Error Estimation for Parametric Nonlinear Dynamical Systems

Sridhar Chellappa | Lihong Feng | Peter Benner

Max Planck Institute for Dynamics of Complex Technical Systems, 1 Sandtorstraße, 39106 Magdeburg, Germany

Correspondence
Sridhar Chellappa. Email: chellappa@mpi-magdeburg.mpg.de

Summary

An adaptive scheme to generate reduced-order models for parametric nonlinear dynamical systems is proposed. It aims to automatize the POD-Greedy algorithm combined with empirical interpolation. At each iteration, it is able to adaptively determine the number of the reduced basis vectors and the number of the interpolation basis vectors for basis construction. The proposed technique is able to derive a suitable match between the reduced basis and the interpolation basis vectors, making the generation of a stable, compact and reliable reduced-order model possible. This is achieved by adaptively adding new basis vectors or removing unnecessary ones, at each iteration of the greedy algorithm. An efficient output error estimator plays a key role in the adaptive scheme. We also propose an improved output error estimator based on previous work. Upon convergence of the POD-Greedy algorithm, the new error estimator is shown to be sharper than the existing ones, implicating that a more reliable reduced-order model can be constructed. The proposed method is tested on several nonlinear dynamical systems, namely, the viscous Burgers' equation and two other models from chemical engineering.

KEYWORDS:
Model Order Reduction, Error Estimation, Adaptivity

1 | INTRODUCTION

Large-scale mathematical models have become common in detailed modelling of complex physical and chemical processes. Very often, these models need to be evaluated repeatedly, for different sets of parameters. To avoid the huge computational burden, model order reduction (MOR) typically seeks a small-scale system that faithfully approximates the original large-scale system with parameter variations. The original large-scale system is referred to as the full-order model (FOM) while the small-scale system is called the reduced-order model (ROM) in the following.

MOR for linear systems has been under investigation for several decades and is well-established [3]. However, methods for nonlinear systems are still under active research. Existing MOR methods for linear systems could be extended to weakly-nonlinear systems or systems with structured non-linearities [5, 7]. For general strong nonlinear systems, the snapshot based methods, e.g., Proper Orthogonal Decomposition (POD), Reduced Basis Method (RBM) are chosen most often. The use of POD/RBM is often accompanied by an interpolatory strategy for efficient evaluation of the nonlinear function of the ROM. The idea is instead of evaluating the vector of nonlinearities with the full dimension, only several elements in the vector are evaluated. The interpolation indices decide which elements should be evaluated. Several interpolation methods are proposed,

e.g., the Empirical Interpolation Method (EIM)[2], the Discrete Empirical Interpolation Method (DEIM) [8, 13], the hyper-reduction method in [1]. POD with DEIM, i.e. POD-DEIM, was proposed for MOR of non-parametric nonlinear systems [8], while RBM combined with EIM (RBM-EIM for short) is often applied to parametric nonlinear systems.

A standard implementation of POD-DEIM is to separately generate the reduced bases (RB) for the state vector and the interpolation bases for the nonlinear vector using POD and DEIM, respectively. This technique nevertheless does not guarantee that the ROM and the interpolation bases are as small as possible. In the worst case, the ROM might be unstable. To avoid these issues, the authors in [11] introduced a method of adaptively constructing the RB and the DEIM bases. The adaptivity is guided by an a posteriori output error estimator for the ROM. Finally, a compact and reliable ROM is obtained. The algorithm in [11] has several drawbacks. Firstly, it is only applicable to non-parametric systems. Secondly, the adaptive scheme is only *one-way*. This means, the interpolation bases can only be extended but cannot be shrunk when necessary. Therefore, the number of the initial basis vectors must be small enough, which could cause more iterations until convergence. Thirdly, the error estimator used in [11] is not ideally sharp and could be further improved.

For parametric systems, direct implementation of RBM-EIM by separately implementing EIM could give rise to similar issues as for non-parametric systems caused by POD-DEIM. In addition, the EIM needs FOM simulations at all samples in a training set, which is time consuming. This issue is also pointed out in [4, 9]. In [10], RBM-EIM is implemented such that both the reduced basis and the interpolation basis are updated simultaneously. The update was done by trivially adding a single new member to each of the basis spaces at each iteration step of the POD-Greedy algorithm [14] designed for parametric nonlinear systems.

Recently, adaptive schemes are proposed in [4, 9] for RB and EIM bases construction. The authors in [9] propose a SER method of simultaneously enriching the RB and EIM bases for nonlinear but stationary problems. The idea is to start with an initial FOM solution at one parameter sample to initialize the EIM. The initial EIM is then used for FOM simulation, from which an initial RB vector is computed. The RB then yields an initial ROM. The ROM solutions at all samples of the parameter are then projected back to produce the approximate FOM solutions. The EIM is updated from the approximate snapshots of the nonlinear function being evaluated at the approximate FOM solutions. In the subsequent iterations, the FOM is never simulated. Instead, only the ROM is updated with the updated EIM, and is then simulated to get the updated approximate FOM solutions, so that a new loop of updates is triggered. Since only the ROM is simulated for EIM and RB updates, the snapshots used for EI and RB construction are approximate snapshots. SER can be considered as an approximate RB-EIM method.

A progressive EIM (PREIM) method for nonlinear, dynamical (time-dependent) systems is proposed in [4]. A special case of the method is a natural extension of the SER method in [9]. The PREIM method evaluates the nonlinear function not only at the approximate FOM solution, but also at the high-fidelity FOM solution, whenever available. More precisely, if a new sample of the parameter is selected for RB enrichment, the high-fidelity solution at this sample and the corresponding many time instances need to be computed to enrich the RB bases. The nonlinear function evaluated at the high-fidelity solution of this sample can be readily obtained. For other samples in the training set, the nonlinear function is evaluated at the approximate FOM solutions computed from the ROM. Therefore, this method can be referred to as semi-approximate RB-EIM.

In this work, we propose a new adaptive scheme of RB-(D)EIM basis enrichment for parametric nonlinear systems. As compared with the adaptive POD-DEIM in [11],

- we have extended the technique to parametric nonlinear systems. The extension is nontrivial, since our adaptive scheme is constructed based on POD-Greedy [14], tailored for parametric systems. Furthermore, the DEIM/EIM does not need to be pre-implemented, avoiding solving the FOM at all samples in the training set.

- We have made the adaptivity more flexible. The RB and (D)EIM bases enrichment is now a *two-way* technique. i.e., it is not only for adaptive basis extension but also for adaptive basis shrinking according to a user-defined error tolerance of the ROM, and the user-given initial basis dimensions.

- An improved and sharper output error estimator is derived for both non-parametric and parametric systems, which can be further used in the proposed adaptive schemes. Radial basis function interpolation is applied to compute the parameter-dependent inf-sup constant cheaply and fast, so that singular value decompositions of a full dimensional matrix at all the samples of the parameter are avoided. In [16] the authors propose a Kriging interpolation based method to estimate the inf-sup constant and also remark on the connections between Kriging and radial basis interpolation. In this work, we apply the more straightforward method of radial basis interpolation as proposed in [21], where the interpolation points are adaptively constructed. Adaptivity is nevertheless not considered in [16].

As compared with SER, PREIM in [4, 9],

- we do not use the approximate state to evaluate the nonlinear function during the RB and EIM construction so that no extra errors are introduced. For DEIM/EIM, we only use the *high-fidelity* solutions that are needed for RB enrichment and are computed by FOM simulation at the parameter samples selected iteratively by the POD-Greedy algorithm. The nonlinear function is evaluated only at those high-fidelity FOM solutions that are available *for free*.
- For both methods, SER and PREIM, a new sample is selected for basis enrichment according to the approximation quality of EIM rather than the ROM quality. Furthermore, the selection process might be time consuming for dynamical systems, since it is done at every parameter sample in the training set and at each time instance corresponding to each sample. Our adaptive scheme is based on an efficient output error estimator for the ROM. At each iteration, a new sample is selected by considering both the EIM error and the RB error. The parameter selection follows POD-Greedy, but the adaptive bases enrichment is based on the separate contributions of the RB and EIM errors to the whole ROM error, thanks to the error estimator.
- At each iteration of either SER or PREIM, only one candidate vector is computed for EIM basis enrichment. Our proposed adaptive scheme makes adaptive EIM construction possible, meaning the number of the new EI basis vectors for basis extension could vary at each iteration. Moreover, the dimension of both RB and EI basis space can also be adaptively shrunk according to the error estimator.
- Beyond those aspects, we have made the RB enrichment adaptive, which is not considered in either SER, PREIM or standard POD-Greedy. This means the number of POD modes to be added to the RB space is adaptively varied at each POD-Greedy iteration.

Note that in [23] an adaptive POD-Greedy algorithm is also proposed, but with different adaptivity. There, the algorithm focuses on adaptively enriching the training set of the parameters.

Summary of contributions:

- In Section 3, we propose an improved primal-dual based output error estimator. The error estimator can be seen as being composed of two parts. The first part is the product of the norms of two residuals: the dual residual and the primal residual. The second part is associated with the approximate state of the dual system. We show that the norm of the dual residual (residual of the dual system) can be further reduced by introducing more efficient solvers for the dual system. The second part of the error estimator can also be reduced by introducing a modified output. Upon convergence of the POD-Greedy algorithm, the new error estimator is shown to be much sharper than the existing ones.
- In Section 4, an adaptive process for RB-(D)EIM basis generation is proposed, which we call adaptive POD-Greedy-(D)EIM. Since we improved the existing adaptive POD-DEIM using the improved error estimator and the proposed two-way approach, we also show the improved adaptive POD-DEIM in this section.
- To efficiently compute the inf-sup constant for error estimation, we apply the radial basis interpolation approach [21] to approximate the inf-sup constant for parametric systems. It shows good accuracy and speed-up.
- The proposed ideas are tested on several examples in Section 5 including two examples from chemical engineering - a fluidized bed crystallizer model and a batch chromatographic model.

The remaining part of the paper is organized as follows. In Section 2, the standard algorithms, on which the proposed adaptive algorithm is built, are reviewed: POD, POD-Greedy, EIM and DEIM. Since the adaptive algorithm is based on an efficient error estimator, we introduce an improved output error estimator in Section 3. The adaptive POD-Greedy-(D)EIM is proposed in Section 4. Simulation results are presented in Section 5, and conclusions are given in Section 6.

2 | PRELIMINARIES

In this section, we review existing algorithms for model order reduction of nonlinear parametric systems, which are the building blocks of the proposed adaptive algorithm. Consider a parametric, nonlinear dynamical system of the following form,

$$\begin{aligned} E(\mu)\dot{x}(t,\mu) &= A(\mu)x(t,\mu) + f(x(t,\mu),\mu) + B(\mu)u(t), \\ y(t,\mu) &= C(\mu)x(t,\mu), \end{aligned} \tag{1}$$

where

- $\mu \in \mathbb{R}^d$ is a vector of parameters in a parameter domain $\mathscr{P} \subset \mathbb{R}^d$,
- $x(t,\mu), f(x(t,\mu),\mu) \in \mathbb{R}^N$ are the state and the state dependent nonlinear vectors respectively,
- $u(t) \in \mathbb{R}^{N_I}$ is the input signal,
- $y(t,\mu) \in \mathbb{R}^{N_O}$ is the output/quantity of interest,
- $E(\mu), A(\mu) \in \mathbb{R}^{N\times N}$ are the system matrices,
- $B(\mu) \in \mathbb{R}^{N\times N_I}, C(\mu) \in \mathbb{R}^{N_O\times N}$ are the input and output matrices, respectively.

Such systems typically arise from the spatial discretization of parametric partial differential equation (PDEs) through finite difference, finite element or finite volume methods. Since N is typically large, we wish to find a system of reduced order $r \ll N$, that accurately approximates the original system solution.

2.1 | Parametric Model Order Reduction via projection

Assume the solution to (1) lies in a low dimensional subspace, then it is possible to construct a matrix $V \in \mathbb{R}^{N\times r}$ with low rank ($r \ll N$), whose columns constitute an orthogonal basis that spans this subspace, such that the solution can be approximated by the basis vectors. Then a ROM can be given via Petrov-Galerkin projection,

$$\begin{aligned} E_r(\mu)\dot{x}_r(t,\mu) &= A_r(\mu)x_r(t,\mu) + f_r(Vx_r(t,\mu),\mu) + B_r(\mu)u(t), \\ y_r(t,\mu) &= C_r(\mu)x_r(t,\mu), \end{aligned} \tag{2}$$

where

- $W \in \mathbb{R}^{N\times r}$ is a matrix, whose columns span a test subspace,
- $x_r(t,\mu) \in \mathbb{R}^r$, is the reduced state vector,
- $f_r(Vx_r(t,\mu),\mu) := W^T f(Vx_r(t,\mu),\mu) \in \mathbb{R}^r$, is the reduced nonlinear vector,
- $A_r(\mu) := W^T A(\mu)V \in \mathbb{R}^{r\times r}$, $E_r(\mu) := W^T E(\mu)V \in \mathbb{R}^{r\times r}$ are the reduced system matrices,
- $B_r(\mu) := W^T B(\mu) \in \mathbb{R}^{r\times N_I}, C_r(\mu) := C(\mu)V \in \mathbb{R}^{N_O\times r}$ are the reduced input and output matrices respectively.

For parametric nonlinear dynamical systems, snapshot based methods are widely used to construct the basis vectors in V and Galerkin projection is often chosen to construct the ROM, i.e. $W = V$. POD is either used as an independent method or as an intermediate step in POD-Greedy [14] for ROM construction. Throughout the paper, $\|\cdot\|$ refers to the vector 2-norm or the matrix spectral norm.

2.2 | POD and POD-Greedy

Proper orthogonal decomposition (POD) is a procedure of constructing an optimal orthonormal basis that is used to approximate a given dataset. Consider a matrix $X \in \mathbb{R}^{N\times K}$, consisting of the given data. Let $\operatorname{rank}(X) = r_X$. For any $\ell \le r_X$, POD gives rise to an orthonormal basis $\{u_i\}_{i=1}^{\ell}$ satisfying the following optimality criterion,

$$\begin{aligned} \{u_i\}_{i=1}^{\ell} = \arg\min_{\{\tilde{u}_i\}_{i=1}^{\ell}} \sum_{j=1}^{K} \left\| x_j - \sum_{k=1}^{l} \langle x_j, \tilde{u}_k\rangle \tilde{u}_k \right\|^2, \\ \text{s.t}, \langle \tilde{u}_i, \tilde{u}_j\rangle = \delta_{ij}, 1 \le i,j \le \ell. \end{aligned} \tag{3}$$

Algorithm 1 Proper orthogonal decomposition (POD)

Input: Snapshots $X = \left[x(t_1), x(t_2), \dots, x(t_K)\right]$, tolerance ϵ_{POD} (a heuristically chosen small value).
Output: POD basis matrix V.

1: Perform $X \xrightarrow{\text{SVD}} U\Sigma W^T$, $\Sigma := \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$, $D := diag(\sigma_1, \sigma_2, \dots, \sigma_{r_X})$.
2: Find r, s.t., $\sum_{i=r+1}^{r_X} \sigma_i \big/ \sum_{i=1}^{r_X} \sigma_i < \epsilon_{\text{POD}}$, r_X is the number of non-zero singular values from Σ, $V = U(:, 1:r)$.

The basis vectors $\{u_i\}_{i=1}^{\ell}$ are called the POD basis, and they are obtained through the Singular Value Decomposition (SVD) of X. When POD is used to compute the projection matrix V for MOR of a dynamical system, the system is first simulated to obtain solutions $x(t_1)$, $x(t_2)$, …, $x(t_K)$ at (selected) time instances t_1, t_2, …, t_K, which are called *snapshots*, then the POD basis of the snapshot matrix $X := [x(t_1), x(t_2), \dots, x(t_K)]$ defines the projection matrix V. Algorithm 1 demonstrates the construction of V through POD.

For parametric systems, one needs a suitable method to capture the variation in the solution manifold due to the parameter variations. For this purpose, we adopt the POD-Greedy approach [14]. It relies on POD in time domain and a greedy selection in the parameter domain; it is a standard implementation of the RBM for parametric dynamical systems. The details are given in Algorithm 2. A crucial ingredient of the algorithm is the availability of a cheap and sharp error estimator Δ. At each iteration, the error estimator must be evaluated n_{train} times, where n_{train} is the cardinality of the parameter training set Ξ [24]. Therefore, it is important that the error estimator is computed in a rapid and reliable manner.

2.3 | EIM and DEIM

From (2), we observe that the complexity of evaluating the nonlinear term f_r, still depends on the FOM, since we need to first evaluate it at $Vx_r \in \mathbb{R}^N$ for a given parameter μ. Some interpolation techniques are proposed to reduce this complexity.

Empirical interpolation.

The Empirical Interpolation Method was introduced in the context of the Reduced Basis Method, in order to reduce the complexity in evaluating nonaffine parameter dependence or nonlinear dependance in the ROM [2, 13]. Both the interpolation indices and the interpolation basis vectors are chosen in a greedy manner.

Discrete empirical interpolation.

The discrete variant of EIM, i.e. DEIM was introduced in [8]. The main differences of DEIM from EIM include two aspects. One is the interpolation basis construction: DEIM uses the *pre-computed POD basis* of the nonlinear snapshot matrix via SVD, as the interpolation basis, whereas EIM constructs the interpolation basis iteratively through a greedy algorithm and from the nonlinear snapshots. The other aspect is the interpolation indices selection: DEIM selects the interpolation indices by looking at the distance between the current *interpolation basis* vector and its approximation obtained via interpolation using the previous interpolation basis vectors. However, EIM chooses the interpolation indices based on the distance between the current *nonlinear snapshot* vector and its approximation obtained via interpolation using the previous interpolation basis vectors.

Algorithm 2 Standard POD-Greedy [14]

Input: Parameter training set $\Xi \subset \mathscr{P}$, tolerance `tol`.
Output: Basis matrix V.

1: Initialize. $V = [\,]$, $\mu^* \in \Xi$.
2: **while do** $\Delta(\mu^*) >$ `tol`
3: Simulate FOM at μ^* and obtain snapshots, $X = [x(t_1, \mu^*), x(t_2, \mu^*), \dots, x(t_K, \mu^*)]$.
4: Update the projection matrix $\bar{X} := \big(X - \text{Proj}_{\mathcal{V}}(X)\big) \xrightarrow{\text{SVD}} U\Sigma W^T$, $\mathcal{V}$ is the subspace spanned by V.
5: $V \leftarrow$ `orth`$\{V, U(:, 1)\}$.
6: $\mu^* := \arg\max_{\mu \in \Xi} \Delta(\mu)$.
7: **end while**

Algorithm 3 Empirical interpolation method (EIM)[2]

Input: Snapshots of the nonlinear vector at a set of parameter samples, $F = \left[f(x(t_1,\mu_1),\mu_1),\dots,f(x(t_K,\mu_1),\mu_1),\dots,f(x(t_1,\mu_{n_s}),\mu_{n_s}),\dots,f(x(t_K,\mu_{n_s}),\mu_{n_s})\right] := [f_1,\dots,f_{Kn_s}] \in \mathbb{R}^{N\times K\cdot n_s}$, maximal iteration steps `max_iter`, tolerance ϵ_{def}, n_s is the total number of parameter samples.

Output: EIM basis $U_f = \left[\zeta_1,\zeta_2,\dots,\zeta_\ell\right]$, index matrix $P = [e_{\wp_1}, e_{\wp_2},\dots,e_{\wp_\ell}]$.

1: Initialize $U_f = [\,]$, $P = [\,]$, $m = 1$.
2: Select the snapshot that maximizes the norm. $\eta_1 = \arg\max_{\substack{f_i \\ 1\le i\le K\cdot n_s}} \|f_i\|$, the interpolation index is given as the position of the row element of η_1 with maximal magnitude. $\left[\sim,\wp_1\right] = \texttt{max}(|\eta_1|)$, where $\eta_1 = [\eta_{11},\eta_{12},\dots,\eta_{1N}]^T$. Here, `max()` refers to the MATLAB® function.
3: The first interpolation basis vector, $\zeta_1 = \eta_1/\eta_{1,\wp_1}$.
4: Update the interpolation matrix $U_f \leftarrow \left[U_f,\zeta_1\right]$, $P \leftarrow [P, e_{\wp_1}]$.
5: **while** $m \le$ `max_iter` **do**
6: $m = m + 1$.
7: Form the m^{th} EIM interpolation for each snapshot vector in F: $\mathcal{I}_m\left[f_i\right] = U_f\left(P^TU_f\right)^{-1}P^Tf_i$, $i = 1,\dots,K\cdot n_s$.
8: Find $f^* = \arg\max_{\substack{f_i \\ 1\le i\le K\cdot n_s}} \left\|f_i - \mathcal{I}_m[f_i]\right\|$. Determine residual $\eta_m = f^* - \mathcal{I}_m[f^*]$.
9: **if** $\|\eta_m\| < \epsilon_{\text{def}}$ **then**
10: $m = m - 1$.
11: stop.
12: **end if**
13: $\left[\sim,\wp_m\right] = \texttt{max}(|\eta_m|)$. Set $\zeta_m = \eta_m/\eta_{m,\wp_m}$
14: $U_f \leftarrow \left[U_f,\zeta_m\right]$, $P \leftarrow \left[P, e_{\wp_m}\right]$.
15: **end while**

Both EIM and DEIM consider the following approximation for the nonlinear term,

$$f(x(t,\mu),\mu) \approx U_f c(t,\mu), \tag{4}$$

where U_f is the matrix of interpolation vectors and $c(t,\mu)$ is the vector of the time-parameter dependent coefficients. The interpolation is achieved by choosing a few interpolation indices where the approximation matches the original function, i.e.

$$P^Tf(x(t,\mu),\mu) = P^TU_fc(t,\mu), \tag{5}$$

where $P = [e_{\wp_1}, e_{\wp_2},\dots,e_{\wp_\ell}]$ is a column permutation of the identity matrix, such that the i^{th} column $e_{\wp_i}$, is all zeros except for the $\wp_i$-th row where the value is 1. It stores all the interpolation indices $\wp_1,\dots,\wp_\ell$. We provide both algorithms as Algorithm 3 and Algorithm 4 for the sake of completeness.

ROM after interpolation.

Using either EIM or DEIM, the ROM in (2) can now be evaluated as,

$$\begin{aligned} E_r(\mu)\dot{x}_r(t,\mu) &= A_r(\mu)x_r(t,\mu) + V^TU_f\left(P^TU_f\right)^{-1}P^Tf(Vx_r(t,\mu),\mu) + B_r(\mu)u(t),\\ y_r(t,\mu) &= C_r(\mu)x_r(t,\mu). \end{aligned} \tag{6}$$

Remark 1. The term $V^TU_f\left(P^TU_f\right)^{-1}$ in (6) can be precomputed. In evaluating $P^Tf(Vx_r(t,\mu),\mu)$, only a few terms (say, $\ell_{\text{EI}} \ll N$ terms) of the nonlinear vector $f(Vx_r(t,\mu),\mu)$ needs to be evaluated, thereby removing the bottleneck in computing the nonlinear term of the ROM.

Remark 2. Algorithm 2 can be combined with either EIM or DEIM for MOR of nonlinear systems. The interpolation bases are precomputed before starting the greedy loop. Algorithm 5 illustrates the Standard POD-Greedy-(D)EIM algorithm. Clearly, the nonlinear vector needs to be evaluated at all samples of $\mu \in \Xi$, at which the FOM needs to be simulated.

Algorithm 4 Discrete empirical interpolation method (DEIM)[8]

Input: Snapshots of the nonlinear vector at a set of parameter samples, $F = \left[f(x(t_1,\mu_1),\mu_1),\dots,f(x(t_K,\mu_1),\mu_1),\dots,f(x(t_1,\mu_{n_s}),\mu_{n_s}),\dots,f(x(t_K,\mu_{n_s}),\mu_{n_s})\right]$, ϵ_{POD}, n_s is as defined in Algorithm 3.

Output: DEIM basis U_f, index matrix $P = [e_{\wp_1}, e_{\wp_2},\dots,e_{\wp_\ell}]$.

1: Initialize $U_f = [\,]$, $P = [\,]$.
2: Perform $F \xrightarrow{\text{SVD}} U\Sigma W^T$, where $U := [u_1^f, u_2^f,\dots,u_{r_F}^f]$, $\Sigma := \begin{bmatrix} D_f & 0 \\ 0 & 0 \end{bmatrix}$,
3: $D_f := diag(\sigma_1^f,\sigma_2^f,\dots,\sigma_{r_F}^f)$.
4: Find ℓ, s.t., $\sum_{i=\ell+1}^{r_F}\sigma_i^f \Big/ \sum_{i=1}^{r_F}\sigma_i^f < \epsilon_{\text{POD}}$, r_F is the number of non-zero singular values in Σ.
5: Select the first interpolation index as the position of the row element with maximal magnitude in the first column of U: $\wp_1 = \arg\max_{j\in\{1,2,\dots,N\}} |u_{1j}^f|$, where $u_1^f = [u_{11}^f, u_{12}^f,\dots,u_{1N}^f]^T$.
6: $U_f \leftarrow u_1^f$, $P \leftarrow [e_{\wp_1}]$.
7: **for** $i = 2$ to ℓ **do**
8: Solve $(P^T U_f)c = P^T u_i^f$, for c.
9: Form the residual, $r_i = u_i^f - U_f c$.
10: $\wp_i = \arg\max_{j\in\{1,2,\dots,N\}} |r_{ij}|$. Here, $r_i = [r_{i1}, r_{i2},\dots,r_{iN}]^T$.
11: $U_f \leftarrow [U_f, u_i^f]$, $P \leftarrow [P, e_{\wp_i}]$.
12: **end for**

3 | IMPROVED *A POSTERIORI* OUTPUT ERROR ESTIMATION

Error estimation plays a crucial role in both standard POD-Greedy and the proposed adaptive algorithms. Error estimators for RBM were mostly proposed based on the weak form of the PDE arising from the finite element discretization [27, 12, 14]. In [29], an efficient output error estimator was proposed in the discretized vector space, which makes the error estimator straightforwardly applicable to the already discretized systems. There, the authors propose an *a posteriori* output error estimator for the ROM in (6). It avoids the accumulation of the residual over time, a phenomenon often seen in other error estimation approaches [12, 14, 15], and therefore is often much sharper than the other error estimators. Moreover, the error estimator is applicable to nonlinear dynamical systems. However, we observed that it is still possible to further improve the sharpness and computational efficiency of the error estimator from different aspects. In the following we briefly review the output error estimation in [29] and propose an improved output error estimator, as well as a more efficient way of computing the error estimator. For the sake of concise notation, we do not explicitly show the parameter dependance of the system matrices (E, A, B, C) and vectors $x(t,\mu)$, $f(x(t,\mu),\mu)$, $y(t,\mu)$. The same shall also be followed by the corresponding dual system matrices and vectors that will be introduced below.

3.1 | Output error estimator from [29]

In this subsection, we briefly review the output error estimator in [29]. As in [29], we consider systems with single output, i.e. $C \in \mathbb{R}^{1\times N}$ and y is a scalar in (1). We will address the error estimation for multiple outputs in Remark 4. Consider a semi-implicit

Algorithm 5 Standard POD-Greedy-(D)EIM [24]

Input: Parameter training set $\Xi \subset \mathscr{P}$, tolerance `tol`, maximal number of iterations `max_iter`, snapshot matrix of the nonlinear vector, $F = \left[f(x(t_1,\mu_1),\mu_1),\dots,f(x(t_K,\mu_1),\mu_1),\dots,f(x(t_1,\mu_{n_{\text{train}}}),\mu_{n_{\text{train}}}),\dots,f(x(t_K,\mu_{n_{\text{train}}}),\mu_{n_{\text{train}}})\right]$, recall that n_{train} is the cardinality of the training set Ξ.

Output: Basis matrix V.

1: (D)EIM interpolation basis calculated using Algorithm 3 or Algorithm 4.
2: Call Algorithm 2, where instead of the ROM in (2), the ROM in (6) is simulated at each iteration.

scheme for the time integration of (1),

$$\begin{aligned} \tilde{E}^k x^{k+1} &= \tilde{A}^k x^k + \Delta t_k f(x^k) + \Delta t_k B^k u^k, \\ y^{k+1} &= C x^{k+1}. \end{aligned}$$

It is called the primal system. For a sharp estimation of the output error, the following dual system is needed,

$$(\tilde{E}^k)^T x_{\mathrm{du}}^{k+1} = -C^T.$$

In general, the system matrices may be time-dependent if the time step Δt_k changes over time, and therefore they are associated with the superscript k. If we consider constant time steps for simplicity, i.e. $\Delta t_k = \Delta t$, then the superscript k can be removed, and the primal and dual systems can be simplified to

$$\begin{aligned} \tilde{E} x^{k+1} &= \tilde{A} + \Delta t f(x^k) + \Delta t B u^k, \\ y^{k+1} &= C x^{k+1}. \end{aligned} \tag{7}$$

and

$$\tilde{E}^T x_{\mathrm{du}} = -C^T, \tag{8}$$

respectively. Note that the dual system becomes a steady system in the simplified case. For clarity we use the simplified case to describe the error estimator, though it is well defined for the general case, too.

Applying the same time integration scheme to the ROM (2) results in,

$$\begin{aligned} E_r(\mu) x_r^{k+1} &= A_r(\mu) x_r^k + \Delta t f_r(V x_r^k) + \Delta t B_r u^k, \\ y_r^{k+1} &= C_r x_r^{k+1}. \end{aligned} \tag{9}$$

It is clear that the time-discrete ROM in (9) is exactly the ROM of the primal system. In [29], a ROM of the dual system is obtained by Galerkin projection using the same projection matrix V as for (2), i.e.

$$(V^T \tilde{E} V)^T x_r^{\mathrm{du}} = -V^T C^T. \tag{10}$$

The approximate solutions, $\hat{x}^{k+1} := V x_r^{k+1}$ to the primal system and $\hat{x}_{\mathrm{du}} := V x_r^{du}$ to the dual system, introduce their residuals, respectively,

$$\begin{aligned} r_{\mathrm{pr}}^{k+1} &= \tilde{A}\hat{x}^k + \Delta t f(\hat{x}^k) + \Delta t B u^k - \tilde{E}\hat{x}^{k+1}, \\ r_{\mathrm{du}} &= -C^T - \tilde{E}^T \hat{x}_{\mathrm{du}}. \end{aligned} \tag{11}$$

Using Eqs. (7), (8) and (9) the error in the output can be shown to be bounded as

$$|y^{k+1} - y_r^{k+1}| \le \phi^{k+1} \|r_{\mathrm{pr}}^{k+1}\|. \tag{12}$$

Here, $\phi^{k+1} := \rho^{k+1} (\|\tilde{E}^{-1}\| \|r_{\mathrm{du}}\| + \|\hat{x}_{\mathrm{du}}\|)$. The term

$$\rho^{k+1} := \frac{\|\tilde{r}_{\mathrm{pr}}^{k+1}\|}{\|r_{\mathrm{pr}}^{k+1}\|}, \tag{13}$$

where

$$\begin{aligned} \tilde{r}_{\mathrm{pr}}^{k+1} &= \tilde{A} x^k + \Delta t f(x^k) + \Delta t B u^k - \tilde{E}\hat{x}^{k+1}, \\ &= \tilde{E}(x^{k+1} - \hat{x}^{k+1}). \end{aligned} \tag{14}$$

is an auxiliary residual obtained by replacing $\hat{x}^{k+1}$ in the "right-hand-side" part ($\tilde{A}\hat{x}^k + \Delta t f(\hat{x}^k) + \Delta t B u^k$) of r_{pr}^{k+1} (see (11)) with the true solution x^{k+1}. It leads to a relation to the state error $x^{k+1} - \hat{x}^{k+1}$.

Remark 3. Note that the error bound only depends on the residuals at the current time step t_{k+1}, and avoids the error accumulation over time. The detailed explanation and proof of the error estimator can be found in [29]. It is proved in [29] that, under mild assumptions, ρ^{k+1} is lower and upper bounded. Upon convergence of the POD-Greedy algorithm, ρ^{k+1} should tend to be 1. This conclusion will be demonstrated numerically in the section on simulation results.

From (12), (14), we note that, at each time instance, the true solution x^{k+1} is required for computing $\tilde{r}_{\mathrm{pr}}^{k+1}$ in the expression of ρ^{k+1}. This can be avoided by approximating ρ^{k+1} with a time-averaged value $\bar{\rho}$ obtained as,

$$\bar{\rho} = \frac{1}{K} \sum_{i=1}^{K} \rho^i, \tag{15}$$

where ρ^i corresponds to $\tilde{r}_{\mathrm{pr}}^i$, which requires the true solution x^i at time instance t_i. If we take t_i as the time instances at which the snapshots are computed for bases enrichment, then x^i are exactly the snapshots, which are available for free.

Furthermore, $\bar{\rho}$ is not available for all μ. Therefore, when used inside a greedy algorithm, at each iteration, we simply approximate the value of $\bar{\rho}$ with,

$$\bar{\rho} \approx \bar{\rho}(\mu^*),$$

where μ^* is the parameter chosen at the current iteration of the POD-Greedy algorithm, so that the snapshots at μ^* are available to compute $\bar{\rho}(\mu^*)$.

With the approximations in Remark 3, the error bound becomes an error estimator, i.e.

$$|y^{k+1} - y_r^{k+1}| \lesssim \bar{\Phi}\|r_{\text{pr}}^{k+1}\| =: \bar{\Delta}, \tag{16}$$

where $\bar{\Phi} := \bar{\rho}\,(\|\tilde{E}^{-1}\|\|r_{\text{du}}\| + \|\hat{x}_{\text{du}}\|)$.

It can be seen that the error estimator $\bar{\Delta}$ in (16) consists of two parts:

$$\bar{\Delta}_1 := \bar{\rho}(\|\tilde{E}^{-1}\|\|r_{\text{du}}\|\|r_{\text{pr}}^{k+1}\|)$$

and

$$\bar{\Delta}_2 := \bar{\rho}\|\hat{x}_{\text{du}}\|\|r_{\text{pr}}^{k+1}\|.$$

The decay rate of $\bar{\Delta}_1$ is decided by the two residuals and the decay speed of $\bar{\Delta}_2$ depends on the product of the primal residual norm and the norm of the approximate dual state $\|\hat{x}_{\text{du}}\|$. We aim to improve the efficiency of the error estimator by considering each of them. On the one hand, we seek a corrected output so that the second part $\bar{\Delta}_2$ is modified to a form with faster decay rate. On the other hand, we try to use more suitable methods to obtain smaller $\|r_{\text{du}}\|$ than that in [29]. In this way, both $\bar{\Delta}_1$ and $\bar{\Delta}_2$ could decay faster, which results in sharper error estimation.

3.2 | Modified $\bar{\Delta}_2$ with corrected output

We consider a correction term to the estimated output quantity given as,

$$\bar{y}_r^{k+1} = y_r^{k+1} - \left(\hat{x}_{\text{du}}\right)^T r_{\text{pr}}^{k+1}. \tag{17}$$

Theorem 1. Given the discrete FOM in (7) and the discrete ROM in (9), assuming $\tilde{E}$ is non-singular at all values of μ, we have the following error bound for the modified output term in (17),

$$|y^{k+1} - \bar{y}_r^{k+1}| \leq \|\tilde{E}^{-1}\|\|r_{\text{du}}\|\|\tilde{r}_{\text{pr}}^{k+1}\| + \|\hat{x}_{\text{du}}\|\|r_{\text{pr}}^{k+1} - \tilde{r}_{\text{pr}}^{k+1}\|, \tag{18}$$

Proof. The error in the modified output can be represented as,

$$y^{k+1} - \bar{y}_r^{k+1} = \boldsymbol{C}(x^{k+1} - \hat{x}^{k+1}) + (\hat{x}_{\text{du}})^T r_{\text{pr}}^{k+1}. \tag{19}$$

Multiplying $(x^{k+1} - \hat{x}^{k+1})^T$ on both sides of (8) we get,

$$(x^{k+1} - \hat{x}^{k+1})^T \tilde{E}^T x_{\text{du}} = -(x^{k+1} - \hat{x}^{k+1})^T \boldsymbol{C}^T. \tag{20}$$

Transposing the above equation we obtain,

$$(x_{\text{du}})^T \tilde{r}_{\text{pr}}^{k+1} = -\boldsymbol{C}\left(x^{k+1} - \hat{x}^{k+1}\right), \tag{21}$$

where we have made use of (14). Next, we simply substitute (21) into (19), followed by addition and subtraction of the term $(\hat{x}_{\text{du}})^T \tilde{r}_{\text{pr}}^{k+1}$ to obtain,

$$\begin{aligned} y^{k+1} - \bar{y}_r^{k+1} &= -(x_{\text{du}})^T \tilde{r}_{\text{pr}}^{k+1} + (\hat{x}_{\text{du}})^T r_{\text{pr}}^{k+1} \\ &= -(x_{\text{du}})^T \tilde{r}_{\text{pr}}^{k+1} + (\hat{x}_{\text{du}})^T r_{\text{pr}}^{k+1} + (\hat{x}_{\text{du}})^T \tilde{r}_{\text{pr}}^{k+1} - (\hat{x}_{\text{du}})^T \tilde{r}_{\text{pr}}^{k+1} \\ &= -(x_{\text{du}} - \hat{x}_{\text{du}})^T \tilde{r}_{\text{pr}}^{k+1} + (\hat{x}_{\text{du}})^T \left(r_{\text{pr}}^{k+1} - \tilde{r}_{\text{pr}}^{k+1}\right). \end{aligned} \tag{22}$$

Consider now the dual system residual as given in (11). It can be shown that

$$\begin{aligned} r_{\text{du}} &= \tilde{E}^T \left(x_{\text{du}} - \hat{x}_{\text{du}}\right), \\ \left(x_{\text{du}} - \hat{x}_{\text{du}}\right) &= \tilde{E}^{-T} r_{\text{du}}. \end{aligned} \tag{23}$$

Inserting (23) into (22) yields

$$y^{k+1} - \bar{y}_r^{k+1} = -(r_{\text{du}})^T (\tilde{E})^{-1} \tilde{r}_{\text{pr}}^{k+1} + (\hat{x}_{\text{du}})^T \left(r_{\text{pr}}^{k+1} - \tilde{r}_{\text{pr}}^{k+1}\right). \tag{24}$$

From the triangle and Cauchy-Schwarz inequalities we obtain,

$$\begin{aligned} |y^{k+1} - \bar{y}_r^{k+1}| &= \| - (r_{\mathrm{du}})^T \tilde{E}^{-1} \tilde{r}_{\mathrm{pr}}^{k+1} + (\hat{x}_{\mathrm{du}})^T (r_{\mathrm{pr}}^{k+1} - \tilde{r}_{\mathrm{pr}}^{k+1})\|, \\ &\leq \|\tilde{E}^{-1}\| \|r_{\mathrm{du}}\| \|\tilde{r}_{\mathrm{pr}}^{k+1}\| + \|\hat{x}_{\mathrm{du}}\| \|r_{\mathrm{pr}}^{k+1} - \tilde{r}_{\mathrm{pr}}^{k+1}\|. \end{aligned} \tag{25}$$

□

In order to remove the quantity $\|\tilde{r}_{\mathrm{pr}}^{k+1}\|$ from the error bound to avoid computing the true solution, we propose the following error estimator. Instead of applying the upper bound $rb_2 := \|r_{\mathrm{pr}}^{k+1}\| + \|\tilde{r}_{\mathrm{pr}}^{k+1}\|$ to $nr := \|r_{\mathrm{pr}}^{k+1} - \tilde{r}_{\mathrm{pr}}^{k+1}\|$ in (25), we propose to use $rb_1 := \left| \|r_{\mathrm{pr}}^{k+1}\| - \|\tilde{r}_{\mathrm{pr}}^{k+1}\| \right|$ to approximate nr, i.e.

$$\|r_{\mathrm{pr}}^{k+1} - \tilde{r}_{\mathrm{pr}}^{k+1}\| \approx \left| \|r_{\mathrm{pr}}^{k+1}\| - \|\tilde{r}_{\mathrm{pr}}^{k+1}\| \right|.$$

This is motivated by the fact that, firstly, applying the upper bound rb_2 will result in an upper bound $(1 + \bar{\rho})\|\hat{x}_{\mathrm{du}}\| \|r_{\mathrm{pr}}^{k+1}\|$ for $\|\hat{x}_{\mathrm{du}}\| r_{\mathrm{pr}}^{k+1} - \tilde{r}_{\mathrm{pr}}^{k+1}\|$ in (25), which is even larger than $\bar{\Delta}_2$ in (16). Here we have used the relation in (13), (15). Secondly, we find that

$$|rb_1 - nr| \leq \begin{cases} 2\rho^{k+1} \|r_{\mathrm{pr}}^{k+1}\|, & \rho^{k+1} > 1 \\ 2\|r_{\mathrm{pr}}^{k+1}\|, & \rho^{k+1} \leq 1, \end{cases} \tag{26}$$

whereas,

$$|rb_2 - nr| \leq [2 + 2\rho^{k+1}] \|r_{\mathrm{pr}}^{k+1}\|.$$

This shows that $|rb_1 - nr|$ poses a smaller upper bound than $|rb_2 - nr|$, implicating that nr could be better approximated by rb_1 than by rb_2. Therefore, we have the following error estimation

$$\begin{aligned} |y^{k+1} - \bar{y}_r^{k+1}| &\lesssim \|\tilde{E}^{-1}\| \|r_{\mathrm{du}}\| \|\tilde{r}_{\mathrm{pr}}^{k+1}\| + \|\hat{x}_{\mathrm{du}}\| \left| \|r_{\mathrm{pr}}^{k+1}\| - \|\tilde{r}_{\mathrm{pr}}^{k+1}\| \right| \\ &= \rho^{k+1} \|\tilde{E}^{-1}\| \|r_{\mathrm{du}}\| \|r_{\mathrm{pr}}^{k+1}\| + |1 - \rho^{k+1}| \|\hat{x}_{\mathrm{du}}\| \|r_{\mathrm{pr}}^{k+1}\| \\ &= \left(\rho^{k+1} \|\tilde{E}^{-1}\| \|r_{\mathrm{du}}\| + |1 - \rho^{k+1}| \|\hat{x}_{\mathrm{du}}\| \right) \|r_{\mathrm{pr}}^{k+1}\| \\ &\approx \left(\bar{\rho} \|\tilde{E}^{-1}\| \|r_{\mathrm{du}}\| + |1 - \bar{\rho}| \|\hat{x}_{\mathrm{du}}\| \right) \|r_{\mathrm{pr}}^{k+1}\|, \end{aligned} \tag{27}$$

where the relation in (13) is used to remove $\|\tilde{r}_{\mathrm{pr}}^{k+1}\|$ from the error estimation. Similarly as for the original error estimator, ρ^{k+1} is approximated with the mean value $\bar{\rho}$ in (15). We then define $\bar{\Psi} := \left(\bar{\rho} \|\tilde{E}^{-1}\| \|r_{\mathrm{du}}\| + |1 - \bar{\rho}| \|\hat{x}_{\mathrm{du}}\| \right)$ to get the following error estimator:

$$|y^{k+1} - \bar{y}_r^{k+1}| \lesssim \bar{\Psi} \|r_{\mathrm{pr}}^{k+1}\|. \tag{28}$$

Note that $\bar{\Delta}_2$ for the error estimator (16) now becomes $|1 - \bar{\rho}| \|\hat{x}_{\mathrm{du}}\| \|r_{\mathrm{pr}}^{k+1}\|$ in (28). Although $\|\hat{x}_{\mathrm{du}}\| \|r_{\mathrm{pr}}^{k+1}\|$ remains unchanged, the coefficient changes from $\bar{\rho}$ to $|1 - \bar{\rho}|$. As has been analyzed in Remark 3, when the POD-Greedy algorithm converges, $\bar{\rho}$ tends to be 1, so that $|1 - \bar{\rho}|$ goes to 0, leading to a bound with faster decay rate.

Remark 4. For systems with multiple outputs, we define the dual system as

$$\tilde{E}^T x_{\mathrm{du}} = -C_i^T,$$

where C_i represents the i^{th} row of the output matrix $C \in \mathbb{R}^{N_O \times N}$. For error estimation, we consider $|y_i - (\bar{y}_r)_i|$ for each element i of y. Then we have,

$$\|y^{k+1} - \bar{y}_r^{k+1}\|_\infty = \max_{i \in \{1,2,\ldots,N_O\}} |y_i^{k+1} - (\bar{y}_r^{k+1})_i|.$$

3.3 | Improving the decay rate of $\bar{\Delta}_1$

Recall that the error estimator consists of two parts $\bar{\Delta}_1$ and $\bar{\Delta}_2$, where $\bar{\Delta}_1 = \rho^{k+1} \|\tilde{E}^{-1}\| \|r_{\mathrm{du}}\| \|r_{\mathrm{pr}}^{k+1}\|$. In [29], $\|r_{\mathrm{du}}\|$ in $\bar{\Delta}_1$ is obtained by reducing the dual system using the projection matrix V for MOR of the primal system, leading to a slowly decaying $\|r_{\mathrm{du}}\|$. To achieve faster decay, we propose to use more suitable methods to compute the approximate solution $\hat{x}_{\mathrm{du}}$ to the dual system, according to different problems. The motivation is based on the observation that $\hat{x}_{\mathrm{du}}$ is *not* necessarily computed by reducing the dual system using MOR. Any appropriate method which can give a good $\hat{x}_{\mathrm{du}}$ should be applicable.

Parametric dual system.

When the dual system has parameter dependence, instead of using the primal system projection matrix V to reduce the dual system, we construct the dual reduced basis separately. With the dual reduced basis, the dual system solution can be much better approximated than using the primal reduced basis. Consequently, the computed $\hat{x}_{\text{du}}$ will give a residual with smaller norm. Certainly, this will lead to higher computational cost.

Non-parametric dual system.

In case the dual system is non-parametric (this can happen when the matrix $\tilde{E}$ is constant), instead of reducing the dual system, we can use Krylov-space methods, e.g. `GMRES, MINRES` [22, 25], to iteratively solve the linear dual system and $\hat{x}_{\text{du}}$ is just the approximate solution computed by those methods. While this is done only once, the computational cost is similar to computing one snapshot and we do not expect much extra computations. This approach leads to much smaller $\|r_{\text{du}}\|$ than using the primal reduced basis to reduce the dual system.

3.4 | Efficiently computing the inf-sup constant

In (16), (28), the term $\|\tilde{E}^{-1}\|$ needs to be calculated. In case of matrix spectral norm $\|\cdot\|_2$,

$$\|\tilde{E}^{-1}\|_2 = \sigma_{\max}(\tilde{E}^{-1}) = \frac{1}{\sigma_{\min}(\tilde{E})}.$$

$\sigma_{\min}(\tilde{E})$ is actually the so-called inf-sup constant commonly used in the RBM. Since the matrix $\tilde{E}$ is parameter-dependent, it can become expensive to evaluate the smallest singular value for each parameter in the training set Ξ, as a large-scale eigenvalue problem needs to be solved for every parameter. Some methods have been proposed to make this computationally efficient. A first attempt was through the Successive Constraints Method (SCM) and its improvement, proposed in [17, 18]. However, SCM often suffers from a very slow convergence [26]. In this work, we make use of the radial basis interpolation method proposed in [21]. This approach avoids the slow convergence rates as seen from SCM, while reducing the computational costs drastically.

Radial basis interpolation for the inf-sup constant.

When considering parameter spaces of high dimensions, the radial basis functions are a good candidate for interpolation basis. To achieve the goal of interpolating the smallest singular values of the matrix $\tilde{E}$, $\forall \mu \in \Xi$, we start with an initial coarse training set of points $Y \subset \Xi$. The large-scale eigenvalue problem is solved for the parameters in this coarse training set and a coarse radial basis interpolant is formed, following the procedure outlined in [21]. The coarse parameter set Y is then enriched in a greedy manner. At each step, a new parameter from Ξ is chosen and added to Y in order to improve the radial basis interpolant. The new parameter is chosen as the one that maximizes a pre-defined *criterion function*, $\mathfrak{C}$ defined over Ξ. The criterion function is such that it promotes adding new points in locations with highly varying response and ensures positivity of the interpolant. It was originally proposed in [19] and further adapted in [21] for the purpose of interpolating the smallest singular values. At the end of each iteration, the relative error (in the $\mathcal{L}_\infty$ norm) between the current and the previous interpolations for all parameters in Ξ is computed. This defines the termination condition. Such an adaptive procedure offsets the need for performing SVD computations on large-scale matrices at all the parameters in Ξ.

4 | ADAPTIVITY

In this section, we propose an adaptive scheme for automatically generating the reduced basis and the interpolation basis. The scheme is based on another separation of the error estimator as detailed below.

4.1 | Error estimation by considering interpolation

In practice, while calculating the residual of the primal system, we approximate the nonlinear term using (D)EIM to avoid the full dimensional computation. Considering this aspect, the residual of the primal system from (11) is now expressed as

$$\begin{aligned} r_{\text{pr}}^{k+1} &= \tilde{A}\hat{x}^k + \Delta t f(\hat{x}^k) + \Delta t \mathcal{I}[f(\hat{x}^k)] - \Delta t \mathcal{I}[f(\hat{x}^k)] + \Delta t B u^k - \tilde{E}\hat{x}^{k+1} \\ &= \underbrace{\left(\tilde{A}\hat{x}^k + \Delta t \mathcal{I}[f(\hat{x}^k)] + \Delta t B u^k - \tilde{E}\hat{x}^{k+1}\right)}_{:= r_{\text{pr},\mathcal{I}}^{k+1}} + \underbrace{\left((\Delta t f(\hat{x}^k) - \Delta t \mathcal{I}[f(\hat{x}^k)])\right)}_{:= e_{\mathcal{I}}^k}. \end{aligned}$$

Here, $\mathcal{I}[f(\cdot)]$, is the interpolation of the function $f(\cdot)$. By considering the separation of r_{pr}^{k+1} in the above equation, the error estimator in (16) or (28) can be split into two contributions - one from approximating the state by reduced basis vectors (RB error) and the other from approximating the nonlinear term by interpolation ((D)EIM error), i.e.

$$|y^{k+1} - y_r^{k+1}| \lesssim \underbrace{\bar{\Phi}\|r_{\text{pr},\mathcal{I}}^{k+1}\|}_{:=\bar{\Delta}_{\text{RB}}^{k+1}} + \underbrace{\bar{\Phi}\|e_{\mathcal{I}}^k\|}_{:=\bar{\Delta}_{\mathcal{I}}^k}, \tag{29}$$

or

$$|y^{k+1} - \bar{y}_r^{k+1}| \lesssim \underbrace{\bar{\Psi}\|r_{\text{pr},\mathcal{I}}^{k+1}\|}_{:=\bar{\Delta}_{\text{RB}}^{k+1}} + \underbrace{\bar{\Psi}\|e_{\mathcal{I}}^k\|}_{:=\bar{\Delta}_{\mathcal{I}}^k}, \tag{30}$$

where for convenience of explanation, we use the same names $\bar{\Delta}_{\text{RB}}^{k+1}$, $\bar{\Delta}_{\mathcal{I}}^k$ for the two parts of either the original error estimator or the modified one, though we will mainly focus on the modified error estimator in the later analysis.

Mean error estimate:

Note that the above error estimator measures the output error at each time step, which corresponds to many values. For the proposed adaptive scheme, we need a single value to measure the actual error. Therefore, we define the mean error as below,

$$\frac{1}{K}\sum_{i=1}^{K}\|y^i - \bar{y}_r^i\| \leq \frac{1}{K}\sum_{i=1}^{K}\left(\bar{\Delta}_{\text{RB}}^i + \bar{\Delta}_{\mathcal{I}}^i\right) = \bar{\Delta}_{\text{RB}} + \bar{\Delta}_{\mathcal{I}} =: \bar{\Delta}. \tag{31}$$

Here, $i = 1, 2, \ldots, K$ indicates the time instances t_i where the snapshots are taken, $\bar{\Delta}_{\text{RB}} = \frac{1}{K}\sum_{i=1}^{K}\bar{\Delta}_{\text{RB}}^i$ and $\bar{\Delta}_{\mathcal{I}} = \frac{1}{K}\sum_{i=1}^{K}\bar{\Delta}_{\mathcal{I}}^i$. With the mean output error estimator, we are ready to propose the adaptive scheme as below.

4.2 | Adaptively increasing/reducing the number of RB and EI basis vectors

Consider a user defined tolerance `tol` for the ROM. At every iteration, we check the relative changes of error estimators $\bar{\Delta}_{\text{RB}}$ and $\bar{\Delta}_{\mathcal{I}}$ in (31) w.r.t the error tolerance `tol`, respectively, i.e.

$$\frac{\bar{\Delta}_{\text{RB}}}{\texttt{tol}} \quad \text{and} \quad \frac{\bar{\Delta}_{\mathcal{I}}}{\texttt{tol}}. \tag{32}$$

It is easy to see that a value in (32) being bigger than 1 implicates the current basis is not accurate enough, and needs to be extended; otherwise, a value being smaller than 1 means that the current bases are accurate enough, and no new basis vectors need to be computed. If possible, some old basis vectors could be removed to make the reduced basis/interpolation basis space as compact as possible. Note that the actual values in (32) could vary from, e.g., 10^{-10} to 10^{10}. In order to decide how many basis vectors need to be added/removed, we use the function log, which maps the actual values in (32) to some moderate values, typically falling into a subinterval of $[-10, 10]$. We then use the rounded log-mapped values as the indicators for basis enriching or shrinking. Let $\ell_{\text{RB}}, \ell_{\text{EI}}$ be the number of basis vectors in the RB, (D)EIM projection matrices at the current iteration, respectively.

For the ROM considered in (2) we update the RB, (D)EIM projection matrices at each iteration of the greedy algorithm based on the following rule,

$$\begin{aligned} \Delta\ell_{\text{RB}} &:= \pm 1 + \left\lfloor \log\left(\frac{\bar{\Delta}_{\text{RB}}}{\texttt{tol}}\right)\right\rfloor, \\ \Delta\ell_{\text{EI}} &:= \pm 1 + \left\lfloor \log\left(\frac{\bar{\Delta}_{\mathcal{I}}}{\texttt{tol}}\right)\right\rfloor. \end{aligned} \tag{33}$$

Algorithm 6 Adaptive POD-(D)EIM (two-way)

Input: POD, (D)EIM projection basis and interpolation points from Algorithms 1 and 4, respectively:
$V \in \mathbb{R}^{N\times \ell_{\text{RB}}}$, $U_f \in \mathbb{R}^{N\times \ell_{\text{EI}}}$, Index matrix $P = [e_{\wp_1}, e_{\wp_2}, \ldots, e_{\wp_\ell}]$, initial choice of basis $\ell^0_{\text{RB}}, \ell^0_{\text{EI}}$, Tolerance `tol`, `zoa`.
Output: ROM with updated size $(\ell^*_{\text{RB}}, \ell^*_{\text{EI}})$.

Form $V^* := V(:, 1:\ell^0_{\text{RB}})$, $U^*_f := U_f(:, 1:\ell^0_{\text{EI}})$, $P^* := P(:, 1:\ell^0_{\text{EI}})$.
Determine the dual system solution $\hat{x}_{\text{du}}$, using `GMRES`.
Simulate the ROM constructed using V^*, U^*_f, P^* and compute the error $\bar{\Delta} := \bar{\Delta}_{\text{RB}} + \bar{\Delta}_{\mathcal{I}}$.
while $\bar{\Delta} \notin$ `zoa` **do**
    Determine $\text{p} = \left\lfloor \log\left(\frac{\bar{\Delta}_{\text{RB}}}{\texttt{tol}}\right)\right\rfloor$, $\text{d} = \left\lfloor \log\left(\frac{\bar{\Delta}_{\mathcal{I}}}{\texttt{tol}}\right)\right\rfloor$.
    Trivial updates $p_0, d_0 = \pm 1$, in case p or $d = 0$, to ensure at least one basis vector is added/removed.
    PODinc $= p_0 + p$, (D)EIMinc $= d_0 + d$.
    $\ell^*_{\text{RB}} = \ell^*_{\text{RB}} +$ PODinc.
    $\ell^*_{\text{EI}} = \ell^*_{\text{EI}} +$ (D)EIMinc.
    Ensure $\ell^*_{\text{EI}} > \ell^*_{\text{RB}}$, for stability reasons.
    Update projection matrices, $V^* := V(:, 1:\ell^*_{\text{RB}})$, $U^*_f := U_f(:, 1:\ell^*_{\text{EI}})$, $P^* := P(:, 1:\ell^*_{\text{EI}})$.
    Simulate the updated ROM constructed using V^*, U^*_f, P^* and compute the updated error $\bar{\Delta} = \bar{\Delta}_{\text{RB}} + \bar{\Delta}_{\mathcal{I}}$.
end while

Here, $\bar{\Delta}_{\text{RB}}$, $\bar{\Delta}_{\mathcal{I}}$ are defined as in (31). Based on the above update, the number of basis vectors for RB, (D)EIM basis enriching/shrinking is given as

$$\begin{aligned} \ell_{\text{RB}} &= \ell_{\text{RB}} + \Delta\ell_{\text{RB}}, \\ \ell_{\text{EI}} &= \ell_{\text{EI}} + \Delta\ell_{\text{EI}}. \end{aligned} \tag{34}$$

Remark 5. Recall that a one-way definition of $\Delta\ell_{\text{RB}}, \Delta\ell_{\text{EI}}$ was proposed in [11], where basis shrinking was not considered. Finally, the adaptive algorithm must start from a small number of basis vectors to be able to continue. Whereas, both basis extension and shrinking are covered by (33). The value ± 1 tries to ensure at least one basis vector is added (+1)/removed (-1), in case the rounded value $p = \left\lfloor \log\left(\frac{\bar{\Delta}_{\text{RB}}}{\texttt{tol}}\right)\right\rfloor$ or $d = \left\lfloor \log\left(\frac{\bar{\Delta}_{\mathcal{I}}}{\texttt{tol}}\right)\right\rfloor$ is zero, but the values $\log\left(\frac{\bar{\Delta}_{\text{RB}}}{\texttt{tol}}\right)$ or $\log\left(\frac{\bar{\Delta}_{\mathcal{I}}}{\texttt{tol}}\right)$ are nonzero (either slightly smaller than zero or slightly larger than zero).

4.3 | Dealing with stagnation in adaptivity

The goal for adaptivity is to add/delete basis vectors to/from the current RB, EI bases so that the ROM meets the given tolerance while being kept as compact as possible. It is observed that in some cases, the convergence of the adaptive algorithm becomes slow when the estimated error is close to the tolerance. In such situations, the number of additional basis vectors to be added/deleted is usually one, resulting in a slow convergence. A second issue is that the error estimator could keep oscillating (below and above the error tolerance) upon basis enriching/shrinking. To avoid the above phenomena, we propose to define a *zone-of-acceptance* (`zoa`) for the output error. In particular, we set a new value $\epsilon^* <$ `tol`. ϵ^* and `tol` then define a *zone-of-acceptance*: $[\epsilon^*, \texttt{tol}]$. Whenever the estimated error falls into $[\epsilon^*, \texttt{tol}]$, the algorithm will terminate. We typically take $\epsilon^* = 0.1\,\texttt{tol}$.

In the following, we first extend the one-way adaptive algorithm in [11] for non-parametric systems to a two-way algorithm, then we propose the adaptive POD-Greedy-(D)EIM algorithm for parametric systems.

4.4 | Adaptive algorithm for non-parametric systems

In this section, we extend the one-way adaptive scheme in [11] to a two-way process. This means we have the flexibility of starting our algorithm given any possible number of initial basis vectors. The method is able to suggest the proper number of basis vectors to be added to or removed from the current basis, and yields a compact and stable ROM, for the given tolerance. We detail this in Algorithm 6. The constants p_0, d_0 in Algorithm 6 act as ± 1 in Equation (34). See Remark 5.

Algorithm 7 Adaptive POD-Greedy-(D)EIM

Input: Parameter training set $\Xi \subset \mathscr{P}$, Tolerance $\texttt{tol}$, $\epsilon_{\texttt{POD}}$ (or $\epsilon_{\texttt{def}}$).

Output: RB Basis $V \in \mathbb{R}^{N\times \ell_{\text{RB}}}$, (D)EIM basis and interpolation points $U_{\text{EI}}, P_{\text{EI}} \in \mathbb{R}^{N\times \ell_{\text{EI}}}$.

1: In case of non-parametric dual system, precompute the approximate solution to the dual system ($\hat{x}_{\text{du}}$), using `GMRES`.
2: Initialize, $V = [\,]$, $V_{\text{du}} = [\,]$, $U_{\text{EI}} = [\,]$, $P_{\text{EI}} = [\,]$, $\ell_{\text{RB}} = 1$, $\ell_{\text{EI}} = 1$. Initial μ^*: a random parameter from Ξ.
3: **while do** $\bar{\Delta}(\mu^*) \notin \texttt{zoa}$
4: Compute FOM at μ^* and obtain snapshots, $\text{X} = [x(t_1, \mu^*), x(t_2, \mu^*), \dots, x(t_K, \mu^*)]$.
5: Update the projection matrix. $\bar{\text{X}} :=$ X - $\text{Proj}_{\mathcal{V}}(\text{X}) \xrightarrow{\text{SVD}} U\Sigma W^T$, $\mathcal{V}$ is the subspace spanned by the columns of V.
6: $V \leftarrow \texttt{orth}\{V, U(:, 1:\ell_{\text{RB}})\}$.
7: `Update_EI`.
8: `Update_`V_{du}, in case of parametric dual system.
9: $\mu^* := \arg\max_{\mu\in\Xi} \bar{\Delta}(\mu)$.
10: `Adapt_basis_update`.
11: **end while**

Remark 6. In our numerical experiments, we noticed that in some cases, when $\ell^*_{\text{EI}} < \ell^*_{\text{RB}}$, the ROM is no longer stable. Therefore, in Step 10 of Algorithm 6 we set the number of EI basis vectors (ℓ^*_{EI}) to be larger than that of the RB basis vectors (ℓ^*_{RB}).

Remark 7. Note that the inputs of Algorithm 6 are the POD and DEIM basis computed by standard algorithms, where both bases are conservatively computed to guarantee accuracy of the ROM. The adaptive algorithm chooses proper basis vectors from the POD and DEIM basis, respectively, in order to construct a more compact and stable ROM. One starts from an initial choice for the POD and DEIM basis combination. Both basis vectors are then iteratively updated, based on the output error estimator.

4.5 | Adaptive algorithm for parametric systems

For standard implementation of the POD-Greedy algorithm combined with interpolation of the nonlinear part, the (D)EIM basis and interpolation indices are usually pre-computed outside of the greedy loop. This separate basis generation often leads to a less compact ROM. In addition, (D)EIM needs FOM simulations at all samples in a training set, which is time consuming especially for problems needing many time steps for one simulation, e.g., the batch chromatographic model presented in Section Section 5. In [10], POD-Greedy and EIM are implemented such that at each iteration of the greedy algorithm, a single basis vector is added to the current reduced bases and the interpolation bases, respectively. The most recent work that addresses this issue can be found in [4, 9]. As has been discussed in the introduction, there are major differences between those existing algorithms and our proposed algorithm.

Algorithm 7 is the proposed adaptive POD-Greedy-(D)EIM algorithm. Algorithms 8, 9 and 10 are the supporting functions needed. The basic idea of Algorithm 7 is that starting from the first selected parameter μ^*, the RB basis and the (D)EIM basis are updated simultaneously but not trivially. The number of the RB and (D)EIM basis vectors ℓ_{RB} and ℓ_{EI} are determined by (34), and updated in the subroutine `Adapt_basis_update`. The reduced basis matrix V is updated in Step 5 of Algorithm 7, where instead of using $\ell_{\text{RB}} = 1$ as in the standard case, ℓ_{RB} is adaptively computed by `Adapt_basis_update`. The interpolation basis matrix U_{EI} is updated by the subroutine `Update_EI`, which essentially implements the interpolation algorithm DEIM or EIM. The dual system is also reduced by the reduced basis method implemented in the subroutine `Update_`V_{du}. It is clear that FOM simulations are only needed for those selected parameters μ^*, not for all parameters in the training set Ξ. Algorithm 10 determines the number of RB and (D)EIM basis vectors ($\ell_{\text{RB}}, \ell_{\text{EI}}$), that will be added/removed at each iteration.

Remark 8. $\texttt{orth}\{V, U(:, 1:\ell_{\text{RB}})\}$ in Step 5 of Algorithm 7 is an orthogonalization step for the updated RB matrix V. Using a modified Gram-Schmidt (MGS) procedure is recommended, where the last p columns in V represent the most recently added basis vectors in the previous iteration. In case of $l_{RB} < 0$, it allows direct removal of p columns from V, without influencing the previous basis vectors.

Remark 9. In practice, we can make use of separate tolerances for the RB, (D)EIM error in Algorithm 10, i.e. Step 1 can be modified as, $p = \left\lfloor \log\left(\frac{\bar{\Delta}_{\text{RB}}(\mu^*)}{\texttt{tol}_{\texttt{RB}}}\right)\right\rfloor$, $d = \left\lfloor \log\left(\frac{\bar{\Delta}_{I}(\mu^*)}{\texttt{tol}_{\texttt{EI}}}\right)\right\rfloor$. The tolerance for (D)EIM approximation is usually set a little lower

Algorithm 8 `Update_EI`

Input: Snapshots of the nonlinear vector at all parameters so far selected by the greedy algorithm, $F = \left[f(x(t_1,\mu),\,\mu),\, f(x(t_2,\mu),\,\mu),\, \ldots\,, f(x(t_K,\mu),\,\mu)\right]$ for all selected μ, ℓ_{EI}, $\epsilon_{\texttt{POD}}$ (or $\epsilon_{\texttt{def}}$).
Output: (D)EIM basis and interpolation points U_{EI}, P_{EI}.
1: `max_iter` $= \ell_{\mathrm{EI}}$ for Algorithm 3 or set $\ell = \ell_{\mathrm{EI}}$ in Step 6 of Algorithm 4 and ignore Step 3.
2: Call $[U_{\mathrm{f}},\, P_{\mathrm{f}}] =$ `EIM`$(F,$ `max_iter`$, \epsilon_{\texttt{def}})$ or $[U_{\mathrm{f}},\, P_{\mathrm{f}}] =$ `DEIM`$(F, \epsilon_{\texttt{svd}})$.

Algorithm 9 `Update_`V_{du}

Input: V_{du}, μ^*_{du}, `tol`.
Output: Updated dual projection matrix V_{du}.
1: **if** $\bar{\Delta}_{\mathrm{du}}(\mu^*_{\mathrm{du}}) >$ `tol` **then**
2: Solve full order dual system for chosen parameter μ^*_{du} and obtain $x_{\mathrm{du}}(\mu^*_{\mathrm{du}})$.
3: Update V_{du}, $V_{\mathrm{du}} :=$ `orth`$\{V_{\mathrm{du}}\,, x_{\mathrm{du}}(\mu^*_{\mathrm{du}})\}$.
4: $\mu^*_{\mathrm{du}} := \arg\max_{\mu\in\Xi} \bar{\Delta}_{\mathrm{du}}(\mu)$.
5: **end if**

Algorithm 10 `Adapt_basis_update`

Input: l_{RB}, l_{EI}, $\bar{\Delta}_{\mathrm{RB}}(\mu^*)$, $\bar{\Delta}_{\mathcal{I}}(\mu^*)$, `tol`.
Output: Updated ℓ_{RB}, ℓ_{EI}.
1: $p = \left\lfloor \log\left(\frac{\bar{\Delta}_{\mathrm{RB}}(\mu^*)}{\texttt{tol}}\right)\right\rfloor$, $d = \left\lfloor \log\left(\frac{\bar{\Delta}_{\mathcal{I}}(\mu^*)}{\texttt{tol}}\right)\right\rfloor$.
2: In case $p = 0$ or $d = 0$, enforce trivial update, $p_0 = \pm 1$ or $d_0 = \pm 1$.
3: $\ell_{\mathrm{RB}} = p_0 + p$.
4: $\ell_{\mathrm{EI}} = \ell_{\mathrm{EI}} + d_0 + d$.
5: Ensure $\ell_{\mathrm{EI}} > (\mathrm{rank}(V) + \ell_{\mathrm{RB}})$, for stability reasons.

than that for the RB approximation. This follows from the observation that the nonlinear vector needs to be sufficiently well-approximated to enable a good state approximation.

5 | NUMERICAL RESULTS

In this section we test the proposed adaptive algorithm on three examples. All the examples we consider can be represented in the general form of (1). We use the same non-parametric model as in [11] to test the extended two-way adaptive Algorithm 6. Algorithm 7 is tested for the other two parametric models.

5.1 | A non-parametric example

The model Fluidized Bed Crystallizer (FBC) is from the field of chemical engineering. Enantiomers are molecules that have the same physical, chemical properties but occur as mirror images of one another. Due to their similar propoerties, separation of the two components is not easily achieved using simple techniques, but requires sophisticated methods such as adsorption, crystallization, etc. For a more in-depth treatment, the reader is referred to [20]. Fig. 1 shows a FBC, which is a long cylindrical column, with the walls tapering inwards as one approaches the bottom. The chemical mixture that has the two enantiomers dissolved in it (called racemate) is injected from the bottom. Some seed crystals need to be introduced into the crystallizer before it begins operation. Seed crystals are essentially the pure crystals of the enantiomer we want to isolate. The seeds are necessary for triggering the precipitation of the crystals in the racemate. During its operation, the smaller crystals move to the top of the crystallizer along with the fluid flow. Bigger crystals sink to the bottom from where they are collected and sent to a crushing device (such as an ultrasonic attentuator) to be crushed to an appropriate size and reintroduced as seed crystals. The

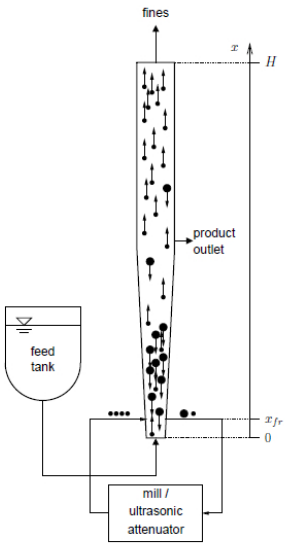


FIGURE 1 Model of the Fluidized bed crystallizer [11].

crystallization process is governed by a set of conservation formulas, called the population balance equations which are PDEs. The PDE governing the FBC is given as:

$$\begin{aligned} A_c(x)\frac{\partial n}{\partial t} = & -\frac{\partial}{\partial x}\left(A_c(x)v_p(x,L,t)n(x,L,t)\right) + D\frac{\partial}{\partial x}\left(A_c(x)\frac{\partial n}{\partial x}\right) - A_c(x)G\frac{\partial n}{\partial L} \\ & + \dot{V}_{\text{us}}\left(n_{\text{us}}(L)\frac{\int_0^\infty nl^3 dl}{\int_0^\infty n_{us}l^3 dl} - n(x,L,t)\right)\hat{\delta}(x - x_{\text{us}}), \end{aligned} \tag{35}$$

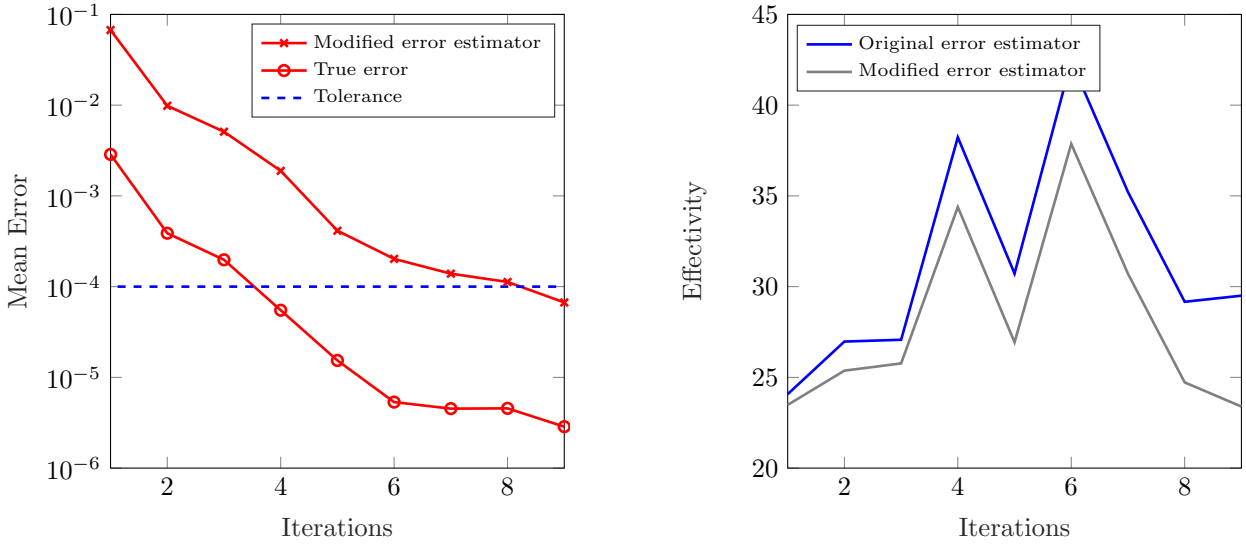
where

- x denotes the spatial coordinate, L denotes the particle size coordinate,
- $A_c(\cdot)$ is the area of cross-section, n is the number size density, i.e., the number of particles per volume of size L at coordinate x at time t,
- v_p is the plug flow velocity,
- D is the dispersion constant, G is the crystal growth factor,
- $\dot{V}_{\text{us}}$ is the volume flow to/from the attenuator,
- $n_{\text{us}}(L)$ is a constant describing the distribution of the crystals coming from the attenuator.

Equation (35) can be discretized and rewritten into a form as in (7). Table 1 gives the model parameters that we consider for the full order simulation. After discretization in space, the original system is of a size $N = 18400$. We use Algorithm 6, where DEIM is used to compute the interpolation basis. The tolerance for the ROM is set as $\texttt{tol} = 10^{-4}$. The model of the crystallizer involves a very long time to reach a cyclic state, usually 5000 seconds. However, for snapshot generation, we need only the transient portion and the first cycle of the steady state since the latter cycles behave very similarly. As a result we only need to simulate the FOM till 500 seconds for snapshot generation. Algorithm 6 has been used to implement the two-way adaptivity scheme. We make use of `GMRES` to solve the associated dual system. It is implemented via the MATLAB® function `gmres`. Moreover, we use the Incomplete LU (ILU) factorization with a drop tolerance of 10^{-3} as a preconditioner. The `GMRES` tolerance is set to be 10^{-6}.

TABLE 1 Simulation data for the FBC.

Interpolation	N	$[0, T]$ (s)	tolerance	ϵ_{POD}
DEIM	18400	[0, 500]	10^{-4}	10^{-10}



(a) Convergence of estimated error.

(b) Effectivity: Original vs Modified estimator.

FIGURE 2 FBC adaptive increment.

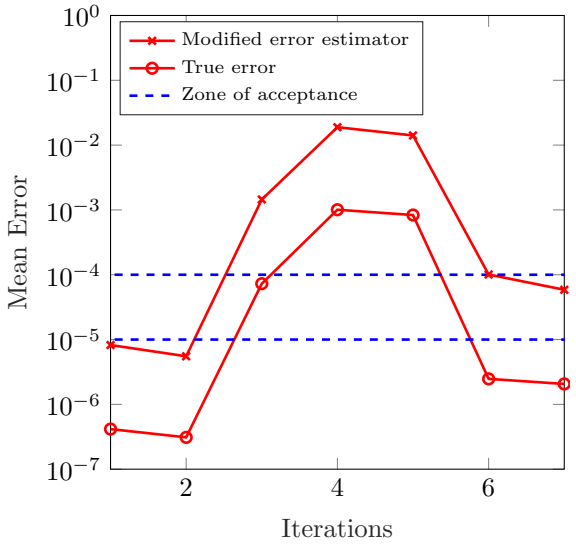
TABLE 2 Simulation results for the FBC example.

Process	Initial		Final		Iterations
	ℓ_{RB}	ℓ_{EI}	ℓ_{RB}	ℓ_{EI}	
Increase	3	8	16	20	9
Decrease	31	39	17	28	7

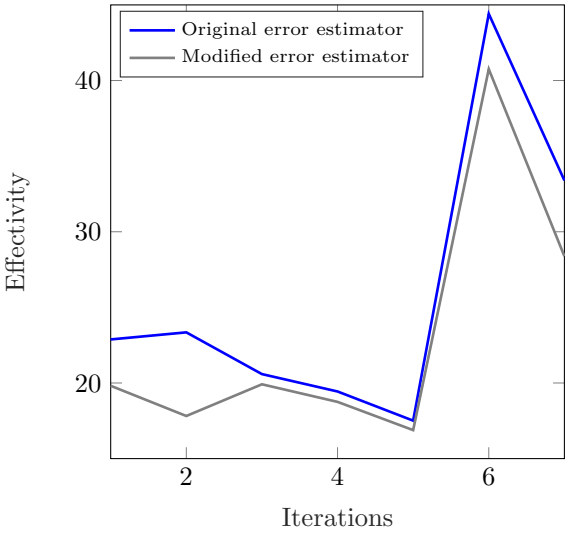
Fig. 2a shows the adaptive generation of POD and DEIM basis starting from small initial numbers of $\ell_{\text{RB}}, \ell_{\text{EI}} = (3, 8)$. The error estimator is below the tolerance after 9 iterations, showing that Algorithm 6 terminates. The adaptive process results in a final ROM of $(\ell_{\text{RB}}, \ell_{\text{EI}}) = (16, 20)$ basis vectors. The size of the ROM would be $r = 60$, with $(\ell_{\text{RB}}, \ell_{\text{EI}}) = (60, 61)$, without adaptivity. Moreover, using the Standard POD algorithm and setting $\epsilon_{\text{POD}} = 10^{-4}$ in Algorithm 1 does not guarantee that the output error of the ROM is also below the tolerance.

For a pair of big initial values: $(\ell_{\text{RB}}, \ell_{\text{EI}}) = (31, 39)$, the iterations of Algorithm 6 are shown in Fig. 3a. In the beginning, the error estimator is below 10^{-5}, indicating that the ROM is very accurate and there is possibility to further reduce the size of the ROM. After 7 iterations, the reduced basis vectors from POD as well as the DEIM basis vectors are adaptively adjusted to $(\ell_{\text{RB}}, \ell_{\text{EI}}) = (17, 28)$. These results have been summarised in Table 2. In Figs. 2a and 3a, the true error is the mean error defined by the left hand side of (31) and the corresponding error estimator is defined by the right hand side of the same inequality.

In Figs. 2b and 3b, we compare the effectivities of the original and the modified error estimators. On the one hand, both error estimators show good effectivities and are relatively sharp. On the other hand, the modified error estimator clearly outperforms the original estimator, especially in the final step of the algorithm. Finally, Figs. 4a and 4b compare the modified error estimator for the final ROM over all time steps t_k, with the true error, in the increasing and decreasing cases, respectively. Fig. 4 not only shows the sharpness of the modified error estimator, especially for the cyclic state in the time interval [200, 500]s, but also verifies the reliability of the error estimator.

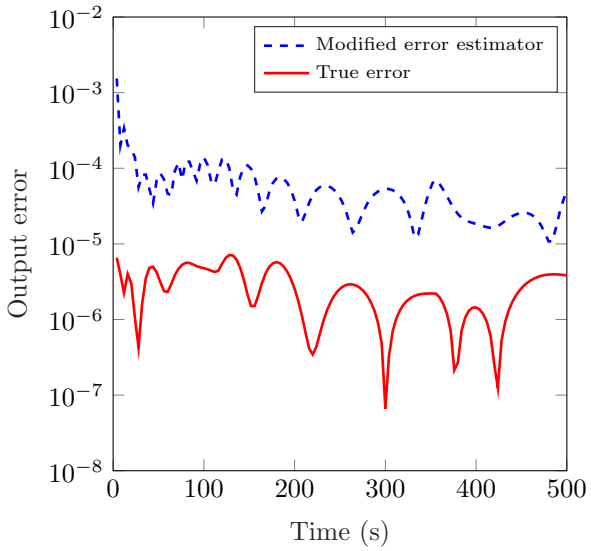


(a) Convergence of estimated error.

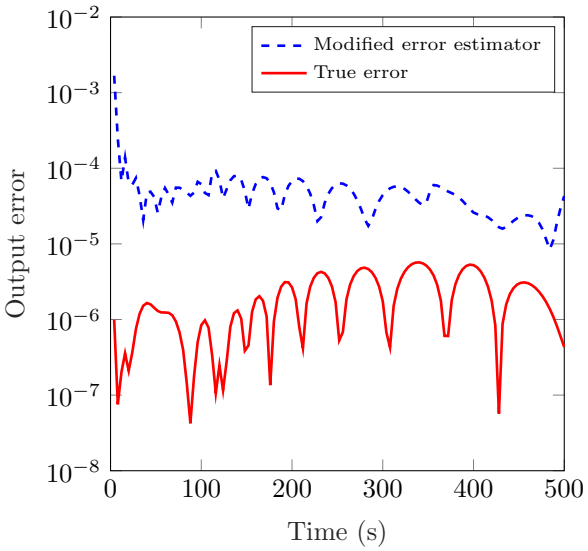


(b) Effectivity: Original vs Modified estimator.

FIGURE 3 FBC adaptive decrement.



(a) Increasing.



(b) Decreasing.

FIGURE 4 Final ROM error over all time for the FBC.

5.2 | Parametric example

We consider two examples of parametric systems. The first is the viscous Burgers' equation and the second is a chemical process called batch chromatography. The former is an example with a parametric dual system while the latter has a non-parametric dual system.

TABLE 3 Simulation parameters for the Burgers' equation.

Interpolation	N	$[0, T]$ (s)	Parameter training set (Ξ)	tolerance	ϵ_{def}
EIM	500	[0,2]	100 log-uniformly distributed samples in [0.0005,1]	10^{-3}	10^{-10}

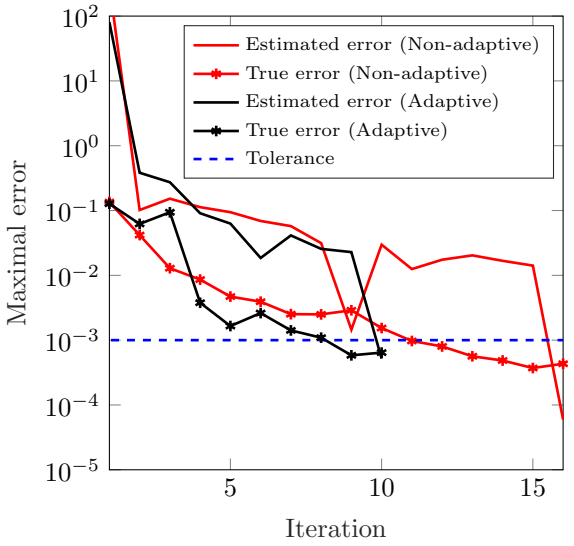


FIGURE 5 Convergence of the greedy algorithms: Algorithm 5 vs Algorithm 7.

TABLE 4 Runtime comparison for the Burgers' equation.

Method	runtime (s)
Adaptive	606
Non-adaptive	933

(a) Adaptive vs non-adaptive.

Method	runtime (s)
SVD	3.7
RBF	0.4

(b) Inf-sup constant computed over training set.

Burgers' equation

We test the proposed Adaptive POD-(D)EIM-Greedy algorithm on the one-dimensional viscous Burgers' equation defined in the domain $\sigma \in \Omega := [0, 1]$. The parameter that varies is the viscosity. The equation and initial boundary conditions are given as

$$\begin{aligned} \frac{\partial w}{\partial t} + w\frac{\partial w}{\partial \sigma} &= \mu \frac{\partial^2 w}{\partial \sigma^2} + s(\sigma, t), \\ w(\sigma, 0) &= 0, \\ \frac{\partial w(1,t)}{\partial \sigma} &= 0, \end{aligned} \tag{36}$$

where $w := w(\sigma, t) \in \mathbb{R}^N$ is the state variable. $s(\sigma, t)$ is the source/input term, μ is the viscosity. The output is taken at the last spatial point in the domain: $y = w(1, t)$. We consider $s(\sigma, t) \equiv 1$.

The simulation parameters are listed in Table 3. A training set, Ξ is formed by 100 log-uniformly distributed samples in the parameter domain $\mathcal{P} := [0.0005\,, 1]$. The model has $N = 500$ equations after discretization in space. We make use of EIM to treat the nonlinear term. We set $\epsilon_{\texttt{def}}$ to be 10^{-10} in Algorithm 7. A time step of $\Delta t = 4 \cdot 10^{-4}$ was used, with the snapshots collected every 10^{th} time step. In Fig. 5, we compare the Standard POD-Greedy-(D)EIM with the proposed Adaptive POD-Greedy-(D)EIM algorithm. It can be seen that using the latter leads to a much quicker convergence of the greedy loop: 10 iterations as compared to 16 iterations that the standard greedy algorithm needs. We show the convergence of the modified error

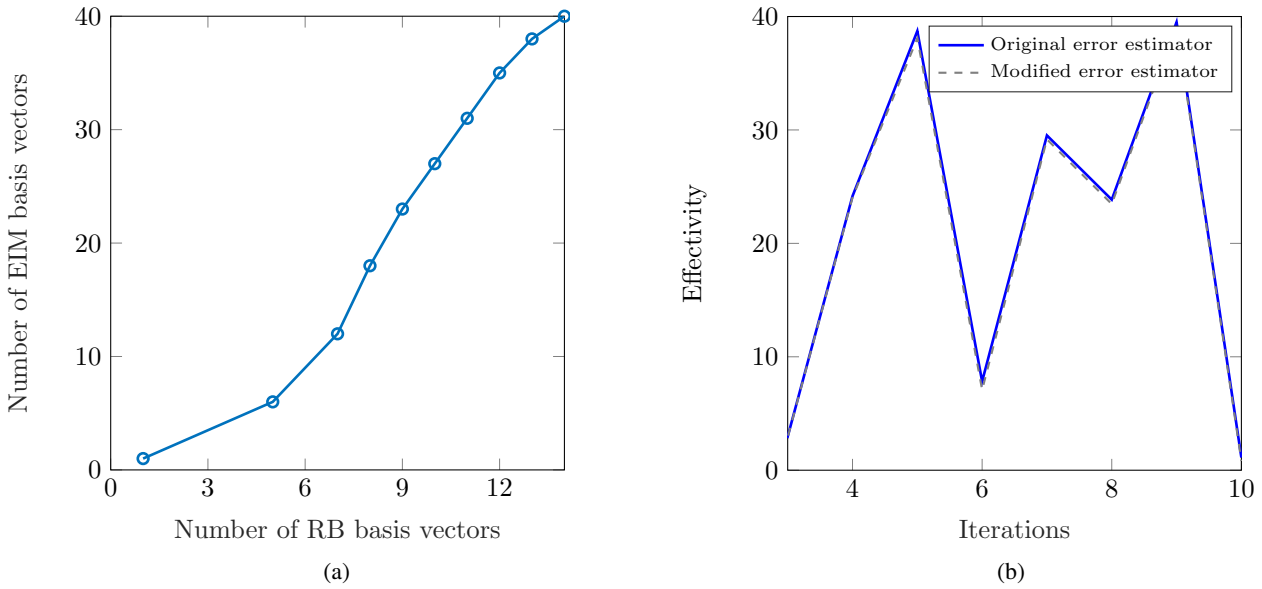


FIGURE 6 Algorithm 7. (a) Adaptive increment of RB vs EIM basis vectors. (b) Effectivity: Original vs modified estimator.

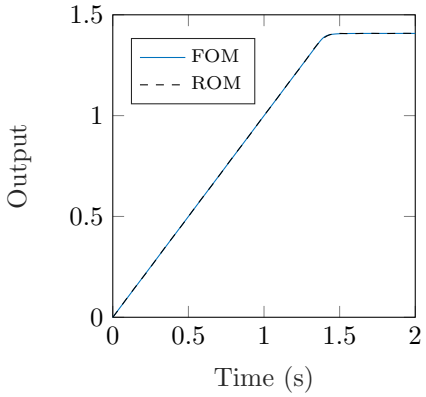


FIGURE 7 Burgers' equation output at $\mu = 5.10^{-4}$.

estimator and the true error for both algorithms. In Fig. 5 and Fig. 9a the maximal errors are defined over all the parameters in the training set Ξ. In particular, $\bar{\Delta}_{\max} := \max_{\mu \in \Xi} \bar{\Delta}(\mu)$, where $\bar{\Delta}(\mu)$ is defined in (31). The maximal true error is defined as: $\max_{\mu \in \Xi} \frac{1}{K} \sum_{i=1}^{K} \|y^{t_i} - \bar{y}_r^{t_i}\|$, where t_i, with $i = 1, 2, \ldots, K$, are the time instances where the snapshots are taken. The improved convergence of Algorithm 7 is a direct consequence of enriching the basis in an adaptive manner.

In Fig. 6a, we plot the successive increments of the RB, EIM basis vectors. Starting from a value of 1 for each, we can see that the biggest jumps are at the first few steps when the output error is estimated to be large. Subsequent steps moderate the number of basis vectors to be added, as the algorithm converges. We end up with a final value of $(\ell_{\mathrm{RB}}, \ell_{\mathrm{EI}}) = (14, 40)$ for the RB, EIM basis respectively. As for the standard implementation, where the EIM basis is precomputed outside the greedy loop, the resulting ROM has dimension $(\ell_{\mathrm{RB}}, \ell_{\mathrm{EI}}) = (16, 154)$. Thus, our proposed algorithm not only produces a ROM that meets a certain tolerance, but also leads to a more compact ROM.

In Table 4a, we show the runtime taken for the Adaptive and Non-adaptive greedy algorithms till convergence. The adaptive algorithm needs much less time. The reduced runtime of the adaptive approach is mainly contributed by the reduced number of FOM simulations. For the inf-sup constant (the smallest singular value of the system matrix) we apply radial basis function interpolation. From Table 4b, it is clear that the RBF approach is much faster as compared to using SVD to determine the

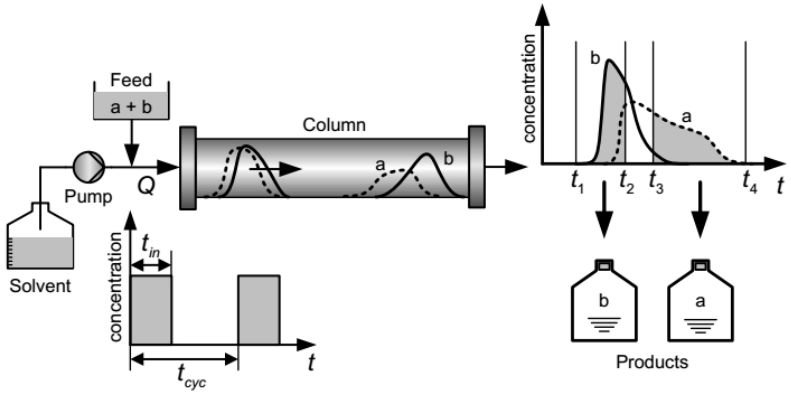


FIGURE 8 The schematic of a batch chromatographic process [29].

smallest singular value of the system matrix. One can imagine, the savings in time would be much more significant for large-scale systems with $N \gg 500$. Fig. 7 shows the output $y(t, \mu)$ of the FOM and $y_r(t, \mu)$ of the ROM at $\mu = 5.10^{-4}$. The ROM solution is nearly indistinguishable from the FOM solution.

Batch chromatography model

The last example is from batch chromatography, a process used to separate components of a mixture. We give the schematic of the process in Fig. 8. The mixture containing two components that need to be separated is periodically injected at one end of the column. In the column, a static bed of a substance called the stationary phase is present. The injected mixture has to pass through this stationary phase. Batch chromatography relies on the phenomenon of adsorption. The components that need to be separated have different adsorption affinities towards the stationary phase and hence tend to move through the column with varying velocities. The separated components are then collected at the end of the column. The time of collecting the two components of the original mixture is determined based on the required purity specifications. The dynamic equations for the process can be given as,

$$\begin{aligned} \frac{\partial c_z}{\partial t} + \frac{1-\epsilon}{\epsilon}\frac{\partial q_z}{\partial t} &= -\frac{\partial c_z}{\partial x} + \frac{1}{Pe}\frac{\partial^2 c_z}{\partial x^2}, \\ \frac{\partial q_z}{\partial t} &= \frac{L}{Q/\epsilon A_c}\kappa_z(q_z^{\text{eq}} - q_z), \end{aligned} \tag{37}$$

where

- c_z, q_z are the concentrations of the two components ($z = a, b$) to be separated, in the solid and the liquid phase, respectively,
- ϵ is the column porosity, Pe is the Peclet number, L is the length of the column, Q is the volumetric feed flow rate, A_c is the cross-section area and κ_z the mass transfer coefficient,
- q_z^{eq} is the adsorption equilibrium and it is the nonlinear term in the equation. It can be given as,

$$q_z^{\text{eq}} = \frac{H_{z1}c_z}{1 + K_{a1}c_a^f c_a + K_{b1}c_b^f c_b} + \frac{H_{z2}c_z}{1 + K_{a2}c_a^f c_a + K_{b2}c_b^f c_b},$$

- H_{z1}, H_{z2} are Henry's constants, K_{z1}, K_{z2} are thermodynamic coefficients and c_z^f is the feed concentration of each component ($z = a, b$).

The PDE defining the batch chromatographic process is discretized using the finite volume method in space and a Crank-Nicolson scheme in time. Thus we have

$$\begin{aligned} Ec_z^{k+1} &= Ac_z^k + d_z^k - \frac{1-\epsilon}{\epsilon}\Delta t h_z^k, \\ q_z^{k+1} &= q_z^k + \Delta t h_z^k. \end{aligned} \tag{38}$$

TABLE 5 Simulation parameters for the batch chromatography equation.

Interpolation	N	Parameter training set Ξ	tolerance	ϵ_{POD}
DEIM	1000	60 samples uniformly distributed in $[0.0667, 0.1667] \times [0.5, 2.0]$	10^{-4}	10^{-10}

Here, E, A are both non-parametric tri-diagonal matrices. The injection time $t_{in} \in [0.5, 2.0]$ and the volumetric feed flow rate $Q \in [0.0667, 0.1667]$ are the two parameters of interest. d_z^k depends on t_{in}. In particular,

$$d_z^k = d_0^k (1, 0, 0, \dots, 0)^T,$$

where $d_0^k := \Delta x Pe \left(\frac{\Delta t}{2\Delta x} + \frac{\Delta t}{Pe \Delta x^2} \right) \chi(t^k)$. The term $\chi(t^k)$ can be given as

$$\chi(t^k) = 1 \text{ if } t^k \in [0\,, t_{in}], \text{else } 0.$$

h_z^k just corresponds to the right hand side of the second equation in (37), showing its dependency on Q. Additionally, we note that for this example, we use the Adaptive Snapshot Selection (AdSS) technique [6] to reduce the computational cost of SVD inside the DEIM algorithm and the greedy loop.

Adaptive Snapshot Selection (AdSS).

For models where the number of time steps is large, it is often cumbersome to perform the SVD on such a large snapshot matrix. AdSS serves as a pre-treatment step to avoid this difficulty. It is essentially an algorithm to determine the linear dependency of successive vectors in the snapshot matrix. The angle between a new vector and the last selected vector is evaluated. If it falls below a tolerance it means that the new vector is almost linearly dependent on the last selected vector and thus can be discarded. For more details on this approach the reader is referred to [28]. Finally, a much thinner snapshot matrix is obtained, reducing the costs of SVD in either the DEIM or the POD-Greedy algorithm.

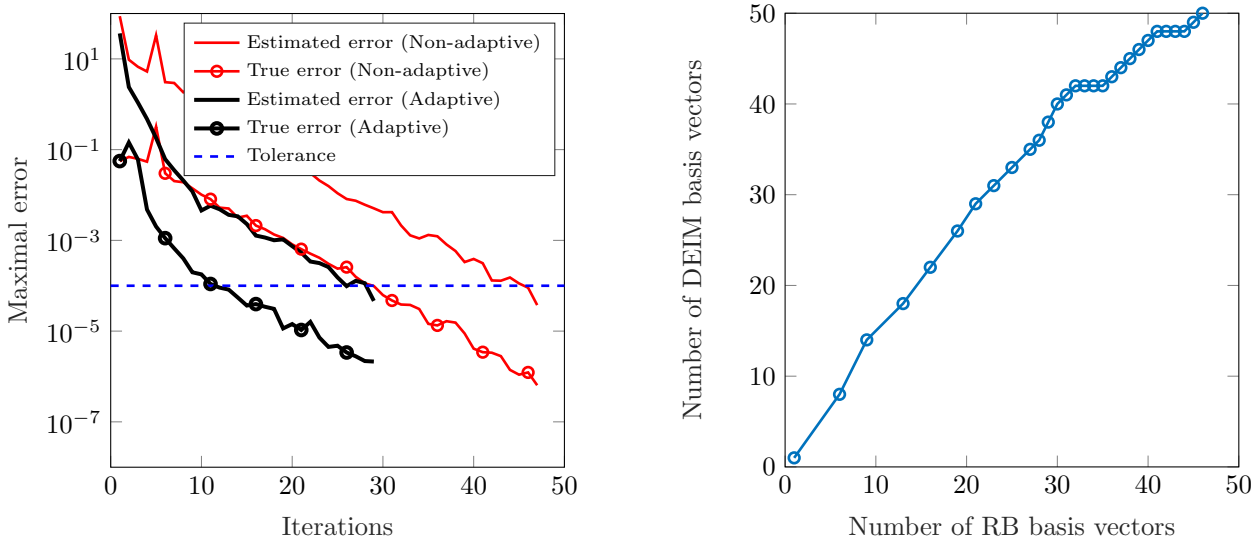
The parameters used for the simulations are shown in Table 5. Since EIM is used for the Burger's equation, for this example, we use DEIM as the interpolation method, to show the flexibility of the adaptive algorithm. We set ϵ_{POD} to be 10^{-10}. The system dimension is $N = 1000$. We take 60 uniformly distributed parameters $\mu := (Q, t_{in})$ from the parameter domain. The simulation time varies depending on the choice of Q. This is due to the fact that the volumetric feed flow (Q) determines the speed of flow of the components through the column. This speed in turn determines the time for which the model needs to be simulated per switching cycle. The tolerance for the ROM error is set as 10^{-4}.

The dual system is a linear algebraic system without parameters, therefore, we propose to use the Krylov method `GMRES` (with the same configurations as done for the FBC example in Section 5.1) to compute the approximate solution to the dual system. The results are to be compared with the one obtained using the primal reduced basis to reduce the dual system.

As for the Burgers' equation, for the batch chromatography example, we show the convergence of the modified error estimator and the corresponding true error, taken as the maximum over all parameters in Ξ, for each iteration. As can be seen from Fig. 9a, the proposed Adaptive POD-Greedy-(D)EIM algorithm results in a quicker convergence as compared to the Standard POD-Greedy. The Standard POD-Greedy results in a ROM of size $(\ell_{\text{RB}}, \ell_{\text{EI}}) = (47, 109)$ in 47 iterations, whereas the proposed Adaptive POD-Greedy method leads to a ROM of size $(\ell_{\text{RB}}, \ell_{\text{EI}}) = (46, 50)$ in 29 iterations. In Fig. 9b, the adaptive increase of the RB, DEIM basis vectors is shown. Similar to the Burgers' equation, the largest jumps are in the first few steps when the error is large. In Fig. 10a, we compare the effectivities of the original and the modified error estimators. The modified estimator offers more efficient results. This is due to the combination of two facts,

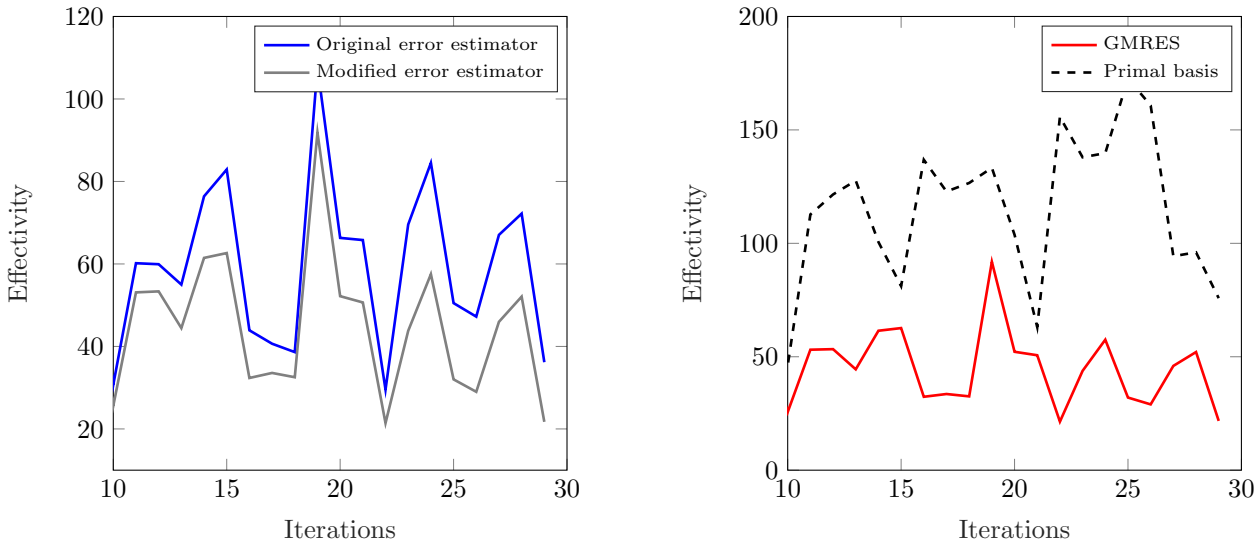
- the use of `GMRES` for solving the dual system leads to a faster decay of the dual residual norm.
- the second term of the modified error estimator is multiplied by the term $1 - \bar{\rho}$. From Fig. 11a we can see that $\bar{\rho}$ tends to one as the iteration proceeds.

In Fig. 10b, we compare the effectivities of the modified error estimator in two cases: in one case, we use the primal reduced basis to reduce the dual system and get the approximate dual solution ($\hat{x}_{\text{du}}$) from the reduced dual system; while in the other case, we use the `GMRES` method to solve the dual system. It is clear that the case of using `GMRES` results in a smaller effectivity. This is due to the fact that the residual of the dual system is very small when using a Krylov space method to solve the dual



(a) (b)

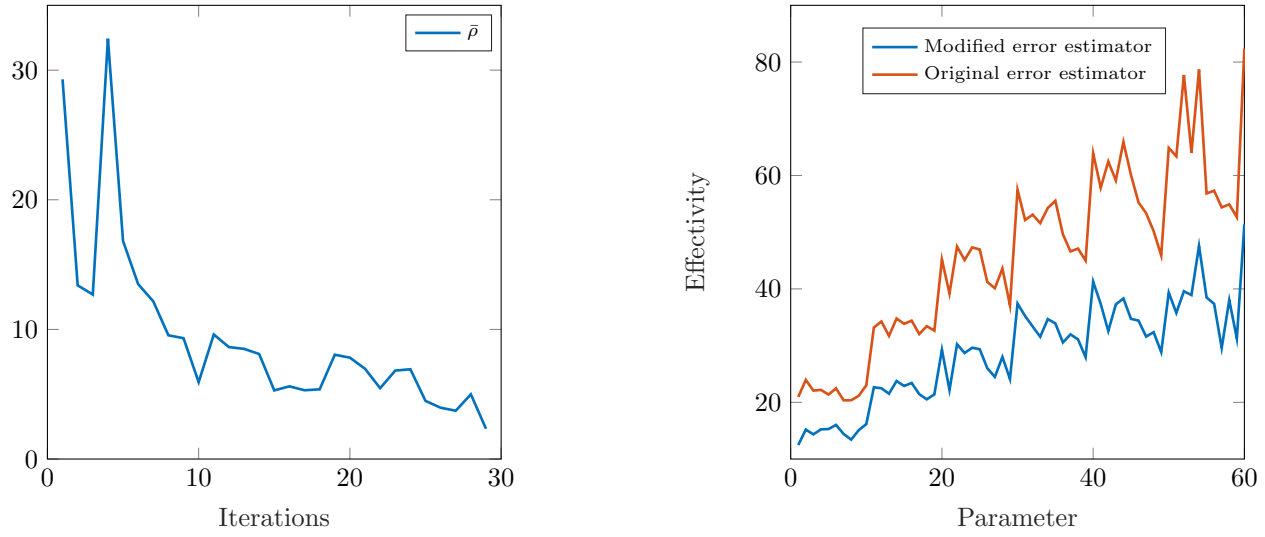
FIGURE 9 Batch chromatography example. (a) Convergence of the greedy algorithms: Algorithms 5 and 7. (b) Adaptive increment of RB vs DEIM basis vectors.



(a) (b)

FIGURE 10 Effectivity comparison inside Algorithm 7 for the batch chromatography example. (a) Effectivity: original vs modified, (b) Modified error estimator using the primal reduced basis vs `GMRES` for the dual system.

system. In Fig. 11b, we compare the effectivity of the original error estimator with the modified error estimator, for estimating the final ROM at the final iteration of Algorithm 7. This shows that the modified error estimator is much sharper for the final ROM. Finally, it can be seen from Table 6 that the proposed adaptive approach is able to find a more compact ROM in a much shorter time.



(a) (b)

FIGURE 11 Results of the adaptive process for batch chromatography. (a) The value of $\bar{\rho}(\mu^*)$ at selected μ^* at each iteration of Algorithm 7. (b) Effectivities of the two error estimators in (29) and (30), respectively.

TABLE 6 Runtime comparison for the batch chromatography example.

Method	runtime (s)
Adaptive	7140
Non-adaptive	11260

6 | CONCLUSIONS

In this work, we propose an adaptive scheme to generate the RB and (D)EIM basis. A good balance between the approximations of the state and the nonlinear term is achieved, while obtaining a ROM of desired tolerance. Whenever necessary, the proposed scheme adaptively adds new basis vectors to, or removes redundant basis vectors from the existing RB, (D)EIM projection matrices, in order to obtain a compact ROM. Expensive FOM simulations at all samples of the training set is avoided. The adaptive scheme is driven by a suitable *a posteriori* error estimator, which makes use of an appropriate dual system solver. We have tested the adaptive approach on several examples from applications. The method is shown to work successfully for both non-parametric and parametric systems. The Adaptive POD-Greedy-(D)EIM scheme is able to deliver a ROM with much fewer iterations when compared to the standard approach. Also, we demonstrated that the proposed modified error estimator offers a better effectivity when compared to a similar estimator. The current error estimator is based on a semi-implicit time discretization. Future work would be to derive an error estimator for implicit time integration schemes involving nonlinear solvers, such as the Newton method.

References

[1] D. Amsallem, M. Zahr, Y. Choi, and C. Farhat, *Design optimization using hyper-reduced-order models*, Struct. Multidisc. Optim. **51** (2015), 919–940.

[2] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera, *An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations*, C. R. Math. Acad. Sci. Paris **339** (2004), no. 9, 667–672.

[3] U. Baur, P. Benner, and L. Feng, *Model order reduction for linear and nonlinear systems: A system-theoretic perspective*, Arch. Comput. Methods Eng. **21** (2014), no. 4, 331–358.

[4] A. Benaceur, V. Ehrlacher, A. Ern, and S. Meunier, *A progressive reduced basis/empirical interpolation method for nonlinear parabolic problems*, SIAM J. Sci. Comput. **40** (2018), no. 5, A2930–A2955.

[5] P. Benner and T. Breiten, *Interpolation-based $\mathcal{H}_2$-model reduction of bilinear control systems*, SIAM J. Matrix Anal. Appl. **33** (2012), no. 3, 859–885.

[6] P. Benner, L. Feng, S. Li, and Y. Zhang, *Reduced-order modeling and rom-based optimization of batch chromatography*, Numerical Mathematics and Advanced Applications - ENUMATH 2013, Springer International Publishing, 2015, pp. 427–435.

[7] P. Benner, P. Goyal, and S. Gugercin, *$\mathcal{H}_2$-quasi-optimal model order reduction for quadratic-bilinear control systems*, SIAM J. Matrix Anal. Appl. **39** (2018), no. 2, 983–1032.

[8] S. Chaturantabut and D. C. Sorensen, *Nonlinear model reduction via discrete empirical interpolation*, SIAM J. Sci. Comput. **32** (2010), no. 5, 2737–2764.

[9] C. Daversin and C. Prud'homme, *Simultaneous empirical interpolation and reduced basis method for non-linear problems*, C.R. Acad. Sci. Paris **353** (2015), no. 12, 1105 – 1109.

[10] M. Drohmann, B. Haasdonk, and M. Ohlberger, *Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation*, SIAM J. Sci. Comput. **34** (2012), no. 2, A937–A969.

[11] L. Feng, M. Mangold, and P. Benner, *Adaptive POD-DEIM basis construction and its application to a nonlinear population balance system*, AIChE J. **63** (2017), no. 9, 3832–3844.

[12] M. Grepl, *Reduced-basis approximation a posteriori error estimation for parabolic partial differential equations*, Ph.D. thesis, Massachussetts Institute of Technology (MIT), Cambridge, USA, 2005.

[13] M. Grepl, Y Maday, N. C. Nguyen, and A. T. Patera, *Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations*, ESAIM: Math. Model. Numer. Anal. **41** (2007), no. 3, 575–605.

[14] B. Haasdonk and M. Ohlberger, *Reduced basis method for finite volume approximations of parametrized linear evolution equations*, ESAIM: Math. Model. Numer. Anal. **42** (2008), no. 2, 277 – 302.

[15] B. Haasdonk and M. Ohlberger, *Efficient reduced models and a posteriori error estimation for parametrized dynamical systems by offline/online decomposition*, Math. Comput. Model. Dyn. Syst. **17** (2011), no. 2, 145–161.

[16] M. W. Hess, S. Grundel, and P. Benner, *Estimating the inf-sup constant in reduced basis methods for time-harmonic Maxwell's equations*, IEEE Trans. Microw. Theory Techn. **63** (2015), no. 11, 3549–3557.

[17] D.B.P. Huynh, D.J. Knezevic, Y. Chen, J.S. Hesthaven, and A.T. Patera, *A natural-norm successive constraint method for inf-sup lower bounds*, Comp. Meth. Appl. Mech. Eng. **199** (2010), no. 29, 1963 – 1975.

[18] D.B.P. Huynh, G. Rozza, S. Sen, and A.T. Patera, *A successive constraint linear optimization method for lower bounds of parametric coercivity and inf-sup lower bounds*, C.R. Acad. Sci. Paris **345** (2007), no. 8, 473 – 478.

[19] T. J. Mackman and C. B. Allen, *Investigation of an adaptive sampling method for data interpolation using radial basis functions*, Internat. J. Numer. Methods Engrg. **83** (2010), no. 7, 915–938.

[20] M. Mangold, L. Feng, D. Khlopov, S. Palis, P. Benner, D. Binev, and A. Seidel-Morgenstern, *Nonlinear model reduction of a continuous fluidized bed crystallizer*, J. Comput. Appl. Math. **289** (2015), 253 – 266.

[21] A. Manzoni and F. Negri, *Heuristic strategies for the approximation of stability factors in quadratically nonlinear parametrized PDEs*, Adv. Comput. Math. **41** (2015), no. 5, 1255–1288.

[22] C. Paige and M. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal. **12** (1975), no. 4, 617–629.

[23] 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), 1262–1292.

[24] A. Quarteroni, A. Manzoni, and F. Negri, *Reduced basis methods for partial differential equations*, La Matematica per il 3+2, vol. 92, Springer International Publishing, 2016, ISBN: 978-3-319-15430-5.

[25] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput. **7** (1986), no. 3, 856–869.

[26] P. Sirković and D. Kressner, *Subspace acceleration for large-scale parameter-dependent hermitian eigenproblems*, SIAM J. Matrix Anal. Appl. **37** (2016), no. 2, 695–718.

[27] K. Veroy, C. Prud'Homme, D. V. Rovas, and A. T. Patera, *A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations*, 16th AIAA Computational Fluid Dynamics Conference (Orlando, United States), 2003.

[28] Y. Zhang, L. Feng, S. Li, and P. Benner, *Accelerating PDE constrained optimization by the reduced basis method: application to batch chromatography*, Internat. J. Numer. Methods Engrg. **104** (2015), no. 11, 983–1007.

[29] _____, *An efficient output error estimation for model order reduction of parametrized evolution equations*, SIAM J. Sci. Comput. **37** (2015), no. 6, B910–B936.