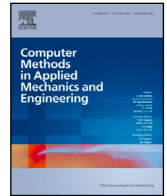Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Accurate error estimation for model reduction of nonlinear dynamical systems via data-enhanced error closure

Sridhar Chellappa *, Lihong Feng, Peter Benner

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

A R T I C L E   I N F O

A B S T R A C T

Accurate error estimation is crucial in model order reduction, to obtain small reduced-order models as well as to certify their accuracy when deployed in downstream applications such as digital twins. In existing a posteriori error estimation approaches, knowledge about the time integration scheme is mandatory, e.g., the residual-based error estimators proposed for the reduced basis method. This poses a challenge when automatic ordinary differential equation solver libraries are used to perform the time integration. To address this, we present a data-enhanced approach for *a posteriori* error estimation. Our new formulation enables residual-based error estimators to be independent of any time integration method. To achieve this, we introduce a corrected reduced-order model which takes into account a data-driven closure term for improved accuracy. The closure term, subject to mild assumptions, is related to the local truncation error of the corresponding time integration scheme. We propose efficient computational schemes for approximating the closure term, at the cost of a modest amount of training data. Furthermore, the new error estimator is incorporated within a greedy process to obtain parametric reduced-order models. Numerical results on three different systems show the accuracy of the proposed error estimation approach and its ability to produce ROMs that generalize well.

1. Introduction

Model order reduction (MOR) has become an important enabling technology that facilitates the rapid and reliable simulation of large-scale systems in a number of scientific disciplines. The central goal of MOR is to replace an *expensive-to-compute* full-order model (FOM) with a surrogate, called the reduced-order model (ROM). The ROM has fewer degrees of freedom which often enables real-time simulation — an important requirement in recent developments such as *digital twins* [1,2]. We refer to the recent books [3–5] for a detailed background and the state-of-the-art in MOR literature.

In this work, we are interested in obtaining ROMs for parametric, nonlinear dynamical systems that arise from the numerical discretization of partial differential equations (PDEs). For such systems, the Reduced Basis Method (RBM) [6–8] is a commonly used MOR approach. The RBM relies on an *a posteriori* error estimator to perform an iterative, greedy sampling of the parameter space to compute solution snapshots, in order to build a linear subspace that serves as an approximation for the solution space. The FOM equations are then projected onto this subspace to obtain the final ROM. *A posteriori*, residual-based error bounds/estimators are an indispensable part of the RBM; such estimators have been proposed for a variety of system classes such as coercive and non-coercive elliptic systems [9,10], linear parabolic systems [11], non-linear or non-affine systems [12–14]. While initial development in the

* Corresponding author.
E-mail addresses: chellappa@mpi-magdeburg.mpg.de (S. Chellappa), feng@mpi-magdeburg.mpg.de (L. Feng), benner@mpi-magdeburg.mpg.de (P. Benner).

RBM community focused on deriving *a posteriori* error estimators in the variational setting, more recent work has also focused on deriving such estimators for PDEs that are discretized using the finite volume method or the finite difference method; see [15–19].

1.1. Motivation

For dynamical systems, all of the above works on *a posteriori* error estimation assume prior knowledge of the time integration scheme. The details of the time integration method are needed to compute the residual vector, which is obtained by substituting an approximate solution in the residual expression. For instance, if a Runge–Kutta method were used, the Butcher tableau of the corresponding scheme needs to be known. However, there are scenarios where no (or only incomplete) knowledge of the time integration scheme is available. One such example is when automatic ordinary differential equation (ODE) solver libraries, e.g., SUNDIALS [20,21], ODEPACK [22], ARKODE [23] are used to perform the time integration. Such ODE solvers are often available readily in computational software such as MATLAB® (ode45, ode15s, etc.) or Python (scipy.integrate.odeint). Another popular ODE solver library is the TS library [24] available in PETSc [25]. Typically, automatic ODE solver packages are computationally efficient as they employ adaptive order selection (e.g., ode15s can switch between order 1 to order 5 methods) and/or adaptive choice of the time step, which is beneficial in case the system being considered exhibits *stiffness*. Moreover, much time is saved by re-using efficient and robust software which is already available and well-maintained. Whenever automatic ODE solvers are used, the standard RBM can no longer be used in a straightforward manner to obtain a ROM. This is owing to the fact that the exact expression of the time integration method used within such ODE solvers is unknown and deriving the corresponding error estimator is an open question. This serves as the fundamental motivation for our work.

1.2. Main contributions

Our work introduces a new data-enhanced paradigm for *a posteriori*, residual-based error estimation for the RBM, in the absence of any knowledge about the time integration scheme used. In addition to the discretized system matrices, our approach only requires access to solution snapshots at different time instances obtained from any black-box ODE solver in a library, at a small set of system parameter samples. By black-box we mean that the details of the integration scheme used within the ODE solver, viz., its expression, step size, etc. are not known. In our approach, we fit a (simpler) time integration scheme of our choice to the available snapshots data. Since the time integration method we impose on the data does not coincide with the actual scheme used to generate the data in first place, there is a mismatch or *defect* at each time step. Subject to a mild assumption, this defect corresponds to the local truncation error (LTE) of the imposed time integration scheme. We then learn this defect term as a function of time and the system parameters. Learning the defect using data allows us to formulate a new *corrected* ROM that takes into account the LTE at each time step. Taking into account the LTE as a closure term means that the corrected ROM can *recover* the solution obtained from a ROM solved by a solver from an ODE library. We use the *corrected* ROM to compute a good approximation to the true residual vector at each time step, leading to accurate estimation of the true error. The computation of the defect term is a key element of our proposed methodology. To do this efficiently, we rely on two observations. First, we demonstrate with numerical evidence that the defect term possesses a certain low-rank structure in space that allows us to efficiently project it on a low-dimensional subspace. Second, if the solution to the FOM is fairly smooth over the parameter space, then this smoothness carries over to the defect term and it can be efficiently approximated with respect to parameter variations using a suitable surrogate model.

1.3. Prior work

A number of recent works have sought to extend the RBM to a non-intrusive framework, mainly for steady systems; see [26–28]. In [27,28], the non-intrusive reduced basis (NIRB) method makes use of FEM solutions computed on two different meshes – one fine and one coarse – to estimate the state error due to the ROM. A different approach is proposed in [26] that involves the empirical interpolation method (EIM) to precompute an affine decomposition consisting of parameter-dependent and parameter-independent quantities. More recently, the method in [29] has sought to extend the NIRB approach to time-dependent systems. All of these methods assume no knowledge of the FOM system matrices and rely purely on snapshots of the state variable. But, they need knowledge of the time integration scheme used to numerically integrate the spatially discretized system. In our new approach, we assume knowledge of the system matrices. Therefore, our approach is an intrusive method in terms of access to the model or the system matrices. However, our approach is non-intrusive in terms of the time integration scheme (ODE solver) used. Nevertheless, we do envisage an extension of our method to the case when there is no access to the system matrices; this will be a subject for future investigation.

To eliminate the dependence on the time integration scheme, our approach aims to increase the accuracy of a user-imposed time integration method by addition of a data-driven closure term. The problem of improving time integration accuracy is an emerging topic and has received attention in the machine learning community [30–32]. The deep Euler method (DEM) is introduced in [30] where the authors' motivation is to improve the first-order accuracy of the explicit Euler time integration scheme by a factor of η, i.e., to $\mathcal{O}(\eta \delta t)$ ($\eta \ll 1$). They achieve this by approximating the LTE using a feed-forward neural network (FNN). Extension of the approach to other time integration schemes beyond the explicit Euler method are also illustrated. In the same spirit of the DEM approach, the work [31] introduces hypersolvers, which are targeted at Neural ODEs [33]. While both DEM and hypersolvers seek to improve the accuracy of the time integration scheme by approximating the LTE, they differ in the assumptions made about the model. DEM assumes the model of the PDE is known and is exact, while for hypersolvers, the model is unknown and is

Table 1
List of FOM variables.

Quantity	Variable	Vector space
State vector	$\mathbf{x}$	$\mathbb{R}^N$
Input vector	$\mathbf{u}$	$\mathbb{R}^{N_I}$
Output vector	$\mathbf{y}$	$\mathbb{R}^{N_O}$
Initial condition	$\mathbf{x}_0$	$\mathbb{R}^N$
Nonlinear vector	$\mathbf{f}$	$\mathbb{R}^N$
Approximate state vector	$\widetilde{\mathbf{x}} := \mathbf{V}\widehat{\mathbf{x}}$	$\mathbb{R}^N$
System matrix	$\mathbf{A}$	$\mathbb{R}^{N \times N}$
Input matrix	$\mathbf{B}$	$\mathbb{R}^{N \times N_I}$
Output matrix	$\mathbf{C}$	$\mathbb{R}^{N_O \times N}$

Table 2
List of ROM variables.

Quantity	Variable	Vector space
Left projection matrix	$\mathbf{W}$	$\mathbb{R}^{N \times n}$
Right projection matrix	$\mathbf{V}$	$\mathbb{R}^{N \times n}$
State vector	$\widehat{\mathbf{x}}$	$\mathbb{R}^n$
Output vector	$\widehat{\mathbf{y}}$	$\mathbb{R}^{N_O}$
Initial condition	$\widehat{\mathbf{x}}_0 := \mathbf{W}^\mathsf{T}\mathbf{x}_0$	$\mathbb{R}^n$
Nonlinear vector	$\widehat{\mathbf{f}} := \mathbf{W}^\mathsf{T}\mathbf{f}(\widetilde{\mathbf{x}})$	$\mathbb{R}^n$
System matrix	$\widehat{\mathbf{A}} := \mathbf{W}^\mathsf{T}\mathbf{A}\mathbf{V}$	$\mathbb{R}^{n \times n}$
Input matrix	$\widehat{\mathbf{B}} := \mathbf{W}^\mathsf{T}\mathbf{B}$	$\mathbb{R}^{n \times N_I}$
Output matrix	$\widehat{\mathbf{C}} := \mathbf{C}\mathbf{V}$	$\mathbb{R}^{N_O \times n}$

approximated by a separate neural network. Our work here differs from DEM and hypersolvers in several ways. Firstly, our aim is to obtain a corrected ROM for better error estimation whereas, in the aforementioned two methods the aim is simply to improve the FOM accuracy during simulation. Secondly, our targets are parametric nonlinear systems. Therefore, we want to learn the LTE for different time instances for a range of system parameters. Both DEM and hypersolvers are limited to non-parametric systems. While either method can potentially be extended to the parametric case, this is expected to be computationally very expensive. Furthermore, both these methods learn a closure term which is a function of the past states. Our approach treats the closure term as a function of time and any additional system parameters. This allows us to take advantage of the special structure and smoothness properties of the closure term.

The rest of this paper is organized as follows. In Section 2, we present the mathematical preliminaries of MOR and provide a short recap of the RBM along with *a posteriori* error estimation. We also illustrate the pitfalls of the current error estimation approach used in RBM, through the example of the heat equation. Section 3 contains the main contributions of this manuscript. We introduce a user-imposed time integration scheme which incorporates the data-driven closure term and derive the *a posteriori* output error estimator suited for situations where ODE solver libraries are used within the RBM. The algorithmic aspects of the proposed approach are discussed in Section 4 while numerical results are presented in Section 5 to support our new method. We conclude with a summary and topics for future research in Section 6.

2. Mathematical background

Consider the following parametric system of ODEs:

$$\frac{d}{dt}\mathbf{x}(\boldsymbol{\mu}) = \mathbf{A}(\boldsymbol{\mu})\mathbf{x}(\boldsymbol{\mu}) + \mathbf{f}(\mathbf{x}, \boldsymbol{\mu}) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{x}(0) = \mathbf{x}_0, \tag{1a}$$

$$\mathbf{y}(\boldsymbol{\mu}) = \mathbf{C}(\boldsymbol{\mu})\mathbf{x}(\boldsymbol{\mu}). \tag{1b}$$

Such systems often arise upon discretizing a PDE using numerical discretization schemes. Table 1 lists each variable and its corresponding dimension. In subsequent discussions, we refer to (1) as the FOM. The state vector $\mathbf{x}(\boldsymbol{\mu})$ can be obtained for any time instance for a given parameter $\boldsymbol{\mu} \in \mathcal{P} \subset \mathbb{R}^p$ by time integration of the FOM using any desired method like Runge–Kutta methods or linear multi-step methods. The set $\mathcal{P}$ denotes the parameter space and p is its dimension. The number of equations N in (1) is often very large to ensure a high-fidelity solution of the underlying physical process. This poses a major challenge for its numerical solution, especially for many instances of the parameter $\boldsymbol{\mu}$ in applications such as control and uncertainty quantification.

2.1. Model order reduction

Projection-based MOR techniques offer a systematic approach to obtain a ROM for (1) in the following form:

$$\frac{d}{dt}\widehat{\mathbf{x}}(\boldsymbol{\mu}) = \widehat{\mathbf{A}}(\boldsymbol{\mu})\widehat{\mathbf{x}}(\boldsymbol{\mu}) + \widehat{\mathbf{f}}(\widetilde{\mathbf{x}}, \boldsymbol{\mu}) + \widehat{\mathbf{B}}\mathbf{u}(t), \qquad \widehat{\mathbf{x}}(0) = \mathbf{W}^\mathsf{T}\mathbf{x}_0, \tag{2a}$$

$$\widehat{\mathbf{y}}(\boldsymbol{\mu}) = \widehat{\mathbf{C}}(\boldsymbol{\mu})\widehat{\mathbf{x}}(\boldsymbol{\mu}). \tag{2b}$$

The above ROM is derived based on the ansatz $\mathbf{x} \approx \widetilde{\mathbf{x}} := \mathbf{V}\widehat{\mathbf{x}}$ applied to (1). The resulting over-determined system of equations is reduced by a Petrov–Galerkin projection, leading to (2). Table 2 lists the reduced variables and their corresponding dimensions. The number of equations n in (2a) is often much smaller than N in (1a), i.e., $n \ll N$. Therefore, the ROM can be readily used for repeated simulations given any new values of the parameter $\boldsymbol{\mu}$. Different MOR techniques differ in how they compute the projection matrices $\mathbf{V}, \mathbf{W}$. When $\mathbf{V} = \mathbf{W}$, it is referred to as Galerkin projection. In the sequel, we will limit ourselves to ROMs obtained through a Galerkin projection.

Remark 1. Although (2a) is a ROM of dimension $n \ll N$, evaluating $\widehat{\mathbf{f}}(\widetilde{\mathbf{x}}, \boldsymbol{\mu})$ involves operations that scale with the FOM dimension N (since $\widetilde{\mathbf{x}} = \mathbf{V}\widehat{\mathbf{x}}$ needs to be evaluated for each time step). Thus, a direct evaluation of (2a) may not offer any computational speedup over evaluating the FOM. Hyperreduction techniques [34–36] can be used to address this issue. We employ the discrete empirical interpolation method (DEIM) for the hyperreduction in our numerical results in Section 5.

Remark 2. While we limit our focus to ROMs obtained using Galerkin projection, we emphasize that this is not a restriction of our proposed error estimation framework. It will be straightforward to use the new error estimator in a Petrov–Galerkin setting as well, given a right projection matrix $\mathbf{W}$ is available. The motivation for limiting ourselves to Galerkin projection is mainly due to its simplicity as only one matrix $\mathbf{V}$ needs to be constructed. Also, for the numerical examples we consider in Section 5, Galerkin projection is sufficient to provide stable ROMs. A Petrov–Galerkin formulation of the RBM for steady systems has been considered in several works [8,10,37,38]. In [39], a Petrov–Galerkin projection has been used in a space–time framework. We further note that in existing literature about the RBM applied to parametric dynamical systems, Galerkin projection is typically adopted. The Petrov–Galerkin approach has been preferred in some problems of fluid dynamics owing to its better stability for such applications [36,40]. In this approach, the ROM solution is obtained based on a residual minimization formulation, e.g., the Least-squares Petrov–Galerkin (LSPG) method [36]. To the best of our knowledge, we are not aware of the LSPG applied to parametric dynamical systems in a greedy setting. However, we note that the LSPG in a greedy framework has been done already for steady problems [41].

2.2. Reduced basis method

The RBM is a greedy approach that builds a global projection matrix $\mathbf{V}$ to obtain a parametric ROM for (1). Since its introduction, the RBM has been a highly successful approach to obtain ROMs for parametric systems in a variety of applications such as process engineering [19,42], geosciences [43], data assimilation [44,45], uncertainty quantification [46] to name just a few.

When RBM is applied to dynamical systems, the POD-Greedy method [47] is adopted. It consists of a greedy sampling in the parameter space and a compression of the time trajectory through singular value decomposition (SVD). Algorithm 1 sketches the pseudo-code for the standard RBM using the POD-Greedy algorithm. In Step 1, `solver` denotes the time integration scheme used to solve Eqs. (1) and (2). This is determined *a priori* by the user based on the ODE library used, e.g. `scipy.ode.integrate`.

Consider a fine discretization of the parameter space $\mathcal{P}$ in the form of a training set $\Xi := \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_{N_p}\}$ consisting of samples of the parameter $\boldsymbol{\mu}$. The POD-Greedy method starts by solving the FOM at a randomly chosen parameter $\boldsymbol{\mu}^* \in \Xi$. Performing the SVD of the solution snapshot matrix

$$\mathbf{X}(\boldsymbol{\mu}^*) := \big[\mathbf{x}(t_0), \mathbf{x}(t_1), \ldots, \mathbf{x}(t_K)\big] \in \mathbb{R}^{N \times N_t} \tag{3}$$

with $N_t = K + 1$ yields the projection basis $\mathbf{V}$. More precisely, we first obtain

$$\mathbf{X}(\boldsymbol{\mu}^*) = \mathbf{V}_{\boldsymbol{\mu}^*} \boldsymbol{\Sigma}_{\boldsymbol{\mu}^*} (\mathbf{W}_{\boldsymbol{\mu}^*})^\mathsf{T}$$

with $\mathbf{V}_{\boldsymbol{\mu}^*} \in \mathbb{R}^{N \times N}, \boldsymbol{\Sigma}_{\boldsymbol{\mu}^*} \in \mathbb{R}^{N \times N_t}$ and $\mathbf{W}_{\boldsymbol{\mu}^*} \in \mathbb{R}^{N_t \times N_t}$. The matrix $\boldsymbol{\Sigma}_{\boldsymbol{\mu}^*}$ is a rectangular diagonal matrix and contains the singular values σ_i in the locations $\boldsymbol{\Sigma}_{\boldsymbol{\mu}^*}(i, i), i \in \{1, 2, \ldots, \min(N, N_t)\}$.

We then update $\mathbf{V}$ by enriching it with the first r_c left singular vectors of $\mathbf{X}(\boldsymbol{\mu}^*)$, i.e., $\mathbf{V}_{\boldsymbol{\mu}^*}(:, 1 : r_c)$. In subsequent iterations, snapshots of the FOM are similarly collected at different values of $\boldsymbol{\mu}^*$ and the matrix $\mathbf{V}$ is updated with new information. Note, however, that for all iterations after the first, we perform an SVD of the matrix $\overline{\mathbf{X}}$ obtained after removing from $\mathbf{X}(\boldsymbol{\mu}^*)$ the information already represented in $\mathbf{V}$, i.e., we set

$$\overline{\mathbf{X}} := \mathbf{X}(\boldsymbol{\mu}^*) - \mathbf{V}(\mathbf{V}^\mathsf{T}\mathbf{X}(\boldsymbol{\mu}^*)).$$

To ensure good conditioning, it is recommended to perform a Gram–Schmidt orthonormalization after each update of $\mathbf{V}$. The choice of $\boldsymbol{\mu}^*$ at each iteration is determined via an error estimator $\Delta(\boldsymbol{\mu})$ as follows:

$$\boldsymbol{\mu}^* = \underset{\boldsymbol{\mu} \in \Xi}{\arg\min} \, \Delta(\boldsymbol{\mu}).$$

The error estimator serves as an upper bound for the true state error $\|\mathbf{x}(t, \boldsymbol{\mu}) - \widetilde{\mathbf{x}}(t, \boldsymbol{\mu})\|$ (or true output error $\|\mathbf{y}(t, \boldsymbol{\mu}) - \widehat{\mathbf{y}}(t, \boldsymbol{\mu})\|$). That is,

$$\|\mathbf{x}(t, \boldsymbol{\mu}) - \widetilde{\mathbf{x}}(t, \boldsymbol{\mu})\| \leq \Delta(\boldsymbol{\mu}), \qquad \text{or}$$

$$\|\mathbf{y}(t, \boldsymbol{\mu}) - \widehat{\mathbf{y}}(t, \boldsymbol{\mu})\| \leq \Delta(\boldsymbol{\mu}).$$

All that is needed to evaluate $\Delta(\boldsymbol{\mu})$ is to solve the ROM (2a) which can be formulated at each greedy iteration using $\mathbf{V}$.

Algorithm 1 POD-Greedy algorithm

Input: Training set Ξ, tolerance (tol), discretized model $(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{f}, \mathbf{x}_0)$
Output: $\mathbf{V}$

1: Initialize $\mathbf{V} = [\ ]$, $\epsilon = 1 + \texttt{tol}$, choose $\texttt{solver}$, greedy parameter $\boldsymbol{\mu}^*$

2: **while** $\epsilon > \texttt{tol}$ **do**

3:    Obtain FOM snapshots $\mathbf{X}(\boldsymbol{\mu}^*)$ at $\boldsymbol{\mu}^*$ with $\texttt{solver}$

4:    Determine $\mathbf{V}_{\boldsymbol{\mu}^*}$ through an SVD of $\overline{\mathbf{X}} := \mathbf{X}(\boldsymbol{\mu}^*) - \mathbf{V}(\mathbf{V}^\mathsf{T}\mathbf{X}(\boldsymbol{\mu}^*))$

5:    Update $\mathbf{V}$ as $\mathbf{V} := \texttt{orth}\big(\mathbf{V}, \mathbf{V}_{\boldsymbol{\mu}^*}(:, 1 : r_c)\big)$
      with $\texttt{orth}(\,\cdot\,)$ denoting an orthogonalization process which can be implemented using the modified Gram–Schmidt process, or QR
      factorization

6:    Get $(\widehat{\mathbf{E}}, \widehat{\mathbf{A}}, \widehat{\mathbf{B}}, \widehat{\mathbf{C}}, \widehat{\mathbf{f}}, \widehat{\mathbf{x}}_0)$ in (2) by Galerkin projection (+ hyperreduction)

7:    Solve ROM with $\texttt{solver}$ for all $\boldsymbol{\mu} \in \Xi$

8:    $\boldsymbol{\mu}^* := \arg\max_{\boldsymbol{\mu} \in \Xi} \Delta(\boldsymbol{\mu})$

9:    Set $\epsilon = \Delta(\boldsymbol{\mu}^*)$

10: **end while**

2.3. A posteriori error estimation for the RBM

The standard error estimation approach in the RBM literature is residual-based [7,8]. In order to derive the error estimator $\Delta(\boldsymbol{\mu})$, knowledge of the time discretization scheme used to integrate the FOM and the ROM is assumed, e.g., using implicit Euler, Crank–Nicolson method, or an implicit–explicit (IMEX) method. Computing $\Delta(\boldsymbol{\mu})$ for a given parameter involves determining the residual vector $\mathbf{r} \in \mathbb{R}^N$ (or its norm) at each time instance; some approaches to error estimation also require the residual of a dual or adjoint system. Let us illustrate this by means of an example.

Suppose (1) is discretized in time using a first-order IMEX scheme [48]. The linear part (involving $\mathbf{A}(\boldsymbol{\mu})$) is discretized implicitly, while the nonlinear vector $\mathbf{f}(\mathbf{x})$ is evaluated explicitly. The resulting discretized system reads

$$\mathbf{E}_{\mathrm{im}}\mathbf{x}_{\mathrm{im}}^k = \mathbf{A}_{\mathrm{im}}\mathbf{x}_{\mathrm{im}}^{k-1} + \delta t\big(\mathbf{f}(\mathbf{x}_{\mathrm{im}}^{k-1}) + \mathbf{B}\mathbf{u}^k\big), \tag{4a}$$

$$\mathbf{y}_{\mathrm{im}}^{k+1} = \mathbf{C}\mathbf{x}_{\mathrm{im}}^k \tag{4b}$$

with $\mathbf{E}_{\mathrm{im}} := \mathbf{I}_N - \delta t \mathbf{A}$, $\mathbf{A}_{\mathrm{im}} := \mathbf{I}_N$, where $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ is the identity matrix. For better clarity, we have not shown the parameter dependence of the system matrices and vectors. Also note that in (4) and hereafter, the subscript 'im' shall denote the IMEX time discretization. The ROM (2) can be discretized in the same way and reads

$$\widehat{\mathbf{E}}_{\mathrm{im}}\widehat{\mathbf{x}}_{\mathrm{im}}^k = \widehat{\mathbf{A}}_{\mathrm{im}}\widehat{\mathbf{x}}_{\mathrm{im}}^{k-1} + \delta t\big(\widehat{\mathbf{f}}(\widetilde{\mathbf{x}}_{\mathrm{im}}^{k-1}) + \widehat{\mathbf{B}}\mathbf{u}^k\big), \tag{5a}$$

$$\widehat{\mathbf{y}}_{\mathrm{im}}^{k+1} = \widehat{\mathbf{C}}\widehat{\mathbf{x}}_{\mathrm{im}}^k \tag{5b}$$

with $\widehat{\mathbf{E}}_{\mathrm{im}} := \mathbf{I}_n - \delta t \widehat{\mathbf{A}}$, $\widehat{\mathbf{A}}_{\mathrm{im}} := \mathbf{I}_n$ with $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ being the identity matrix. The residual arising due to the ROM approximation can be computed by substituting the approximate state vector $\widetilde{\mathbf{x}}_{\mathrm{im}}^k$ into (4a). The resulting residual at the kth time step, $\mathbf{r}^k$, reads

$$\mathbf{r}^k := \mathbf{A}_{\mathrm{im}}\widetilde{\mathbf{x}}_{\mathrm{im}}^{k-1} + \delta t\big(\mathbf{f}(\widetilde{\mathbf{x}}_{\mathrm{im}}^{k-1}) + \mathbf{B}\mathbf{u}^k\big) - \mathbf{E}_{\mathrm{im}}\widetilde{\mathbf{x}}_{\mathrm{im}}^k. \tag{6}$$

It is clear that in order to obtain the residual $\mathbf{r}^k$, the time discretization scheme for the ROM should be the same as that for the FOM so that $\widetilde{\mathbf{x}}_{\mathrm{im}}^k$ in (6) and $\mathbf{x}_{\mathrm{im}}^k$ in (4) correspond to the same time instance t^k. An *a posteriori* error bound $\Delta(\boldsymbol{\mu})$ for the approximation error $\mathbf{e}^k(\boldsymbol{\mu}) := \mathbf{x}_{\mathrm{im}}^k(\boldsymbol{\mu}) - \widetilde{\mathbf{x}}_{\mathrm{im}}^k(\boldsymbol{\mu})$ for a given parameter $\boldsymbol{\mu}$ can be computed based on the residual as below.

Theorem 2.1 (*Residual-based Error Bound*). *Suppose that the nonlinear quantity* $\mathbf{f}(\mathbf{x}, \boldsymbol{\mu})$ *is Lipschitz continuous in the first argument for all* $\boldsymbol{\mu}$ *such that there exists a constant* $L_{\mathbf{f}}$ *for which*

$$\|\mathbf{f}(\mathbf{x}, \boldsymbol{\mu}) - \mathbf{f}(\widetilde{\mathbf{x}}_{\mathrm{im}}, \boldsymbol{\mu})\| \le L_{\mathbf{f}} \|\mathbf{x} - \widetilde{\mathbf{x}}_{\mathrm{im}}\|.$$

Further assume that for any parameter $\boldsymbol{\mu}$ *the projection error at the first time step is*

$$\|\mathbf{e}^0(\boldsymbol{\mu})\| = \|\mathbf{x}_{\mathrm{im}}^0(\boldsymbol{\mu}) - \widetilde{\mathbf{x}}_{\mathrm{im}}^0(\boldsymbol{\mu})\| = \|\mathbf{x}_{\mathrm{im}}^0(\boldsymbol{\mu}) - \mathbf{V}\mathbf{V}^\mathsf{T}\mathbf{x}^0(\boldsymbol{\mu})\|.$$

The error in the state variable at the kth *time step,* $\|\mathbf{e}^k(\boldsymbol{\mu})\| = \|\mathbf{x}_{\mathrm{im}}^k(\boldsymbol{\mu}) - \widetilde{\mathbf{x}}_{\mathrm{im}}^k(\boldsymbol{\mu})\|$ *is given by the following expression:*

$$\|\mathbf{e}^k(\boldsymbol{\mu})\| \le \Delta^k(\boldsymbol{\mu}) := \xi(\boldsymbol{\mu})^k \left\|\mathbf{e}^0(\boldsymbol{\mu})\right\|_2 + \sum_{i=1}^{k} \zeta(\boldsymbol{\mu}) \cdot \big(\xi(\boldsymbol{\mu})\big)^{k-i} \cdot \|\mathbf{r}^i(\boldsymbol{\mu})\|_2 \tag{7}$$

where $\zeta(\boldsymbol{\mu}) := \|\big(\mathbf{E}(\boldsymbol{\mu})\big)^{-1}\|_2$ *and* $\xi(\boldsymbol{\mu}) := \left(\|\big(\mathbf{E}(\boldsymbol{\mu})\big)^{-1}\mathbf{A}(\boldsymbol{\mu})\|_2 + \delta t\, L_{\mathbf{f}}\|\big(\mathbf{E}(\boldsymbol{\mu})\big)^{-1}\|_2\right)$.

Proof. See Appendix A. □

Residual-based error bounds such as the one above are already available in the RBM literature [47,49] for both linear and nonlinear systems.

In many applications, only a small set of variables which are obtained as a linear combination of the state variables are of interest. These are typically called the output of the system or quantities of interest (QoI). Goal-oriented *a posteriori* error estimators for the QoI have been discussed in several works [15,18,19,49]. They typically involve the residual of a dual or an adjoint system. The general form of the goal-oriented error estimators in [15,49] is

$$\|\mathbf{y}^k(\boldsymbol{\mu}) - \widehat{\mathbf{y}}^k(\boldsymbol{\mu})\| \leq \Gamma(\boldsymbol{\mu}) \cdot \left(\sum_{i=1}^{k} \|\mathbf{r}^i\|^2 \right)^{\frac{1}{2}} \cdot \left(\sum_{j=0}^{k-1} \|\mathbf{r}_{\mathrm{du}}^{K-k+j}\|^2 + \delta_{\mathrm{du}}^K \right)^{\frac{1}{2}}, \tag{8}$$

where $\|\mathbf{r}_{\mathrm{du}}^i\|$ denotes the residual of an appropriately defined dual system at the ith time instance, δ_{du}^K is a scaled upper bound for the dual ROM state error at the last time step and $\Gamma(\boldsymbol{\mu})$ is some parameter-dependent constant. Improving upon this expression, a different goal-oriented error estimator was proposed in [18,19] having the following form:

$$\|\mathbf{y}^k(\boldsymbol{\mu}) - \widehat{\mathbf{y}}^k(\boldsymbol{\mu})\| \leq \breve{\Gamma}(\boldsymbol{\mu}) \cdot \|\mathbf{r}^k\| \cdot \|\mathbf{r}_{\mathrm{du}}^k\| \tag{9}$$

with $\breve{\Gamma}(\boldsymbol{\mu})$ being a parameter-dependent constant. This error estimator avoids the accumulation of the residuals over time, which is a major drawback for the goal-oriented error estimator in (8) and also the state error estimator in (7). It is also to be noted that the ROM resulting from the use of a goal-oriented error estimator often turns out to be of a smaller size.

Having reviewed the main ideas of RBM and the standard *a posteriori* error estimator, we next illustrate the disadvantage of this standard approach when ODE solver libraries are used to solve the FOM/ROM within Algorithm 1.

2.4. RBM with ODE solver libraries

Automatic ODE solver packages are implemented and readily available in a number of open-source and proprietary computational software packages. The MATLAB® ODE Suite [50], for instance, implements both linear multi-step (`ode15s`, `ode23s`, etc.) and Runge–Kutta-type solvers (`ode45`, `ode23`, etc.). In Python, using the `scipy` module one can access the `odeint` and `solve_ivp` submodules, both providing access to a variety of standard time integration schemes. In addition to these, there are several stand-alone libraries to solve ODEs such as SUNDIALS [21], ODEPACK [22], ARKODE [23]. All of the mentioned libraries implement adaptive order and adaptive time-stepping which makes them highly efficient for a variety of problems, e.g., problems exhibiting stiffness.

When adaptive ODE solver packages are used within the RBM (i.e., in Steps 3 and 7 of Algorithm 1, `solver` is replaced by the chosen method from the package), the standard error estimation approach (see Section 2.3) becomes less straightforward. One can no longer write the corresponding expression of the residual resulting from the ROM (e.g., (6)). The reason for this is that the exact expression of the time integration method used is unknown, as the `solver` adaptively varies the time step and/or the order of the scheme. If a residual expression obtained from a user-imposed time discretization scheme, e.g., (6) is used, the resulting error estimator (7) no longer gives an efficient bound for the true error, i.e., the error between $\mathbf{V}\widehat{\mathbf{x}}^k(\boldsymbol{\mu})$ and $\mathbf{x}^k(\boldsymbol{\mu})$ obtained from simulating the ROM (2) and the FOM (1), respectively, using an ODE solver from a library. We illustrate this next by obtaining a ROM for the simple non-parametrized linear heat diffusion model and estimating its error.

2.5. Example: ROM for the linear heat equation

We consider the linear heat equation in 1-D over the domain $\Omega = [0, 1]$ and time $t \in [0, 1]$

$$\frac{\partial}{\partial t} v(z, t; \mu) - \mu \frac{\partial^2}{\partial z^2} v(z, t; \mu) = 0 \tag{10}$$

where $v(z, t; \mu)$ is the state, $z \in \Omega$ is the spatial variable and the thermal diffusivity is $\mu = 0.06$. We further impose Dirichlet boundary conditions $v(0, t; \mu) = v(1, t; \mu) = 0$, $\forall t \in [0, 1]$. We also fix the output variable of interest as the value of the state at the node next to the right boundary. Employing the finite difference method, we discretize the domain Ω in an equidistant fashion, with a grid size of 2^{-8}. With this, the resulting discretized ODE can be written in a form similar to (1) as

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x}, \qquad \mathbf{x}(0) = \mathbf{x}_0, \tag{11}$$

where $\mathbf{x} \in \mathbb{R}^{255}$ is the discretized state vector and $\mathbf{A} \in \mathbb{R}^{255 \times 255}$. We let the initial condition $\mathbf{x}_0$ be a normal distribution, i.e.,

$$\mathbf{x}_0 := \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \cdot \left(\frac{z - \mathrm{m}}{\sigma} \right)^2}$$

with a mean $\mathrm{m} = 0.5$ and a standard deviation of $\sigma = 0.15$. Our aim is to obtain a ROM for (11) and use the state and output error estimators in Eqs. (7) and (8), respectively, to quantify the error of the ROM. For the output error estimator, we use the one proposed in [19]. To this end, we consider the standard POD-based ROM. This involves obtaining snapshots of the state vector in (11) for
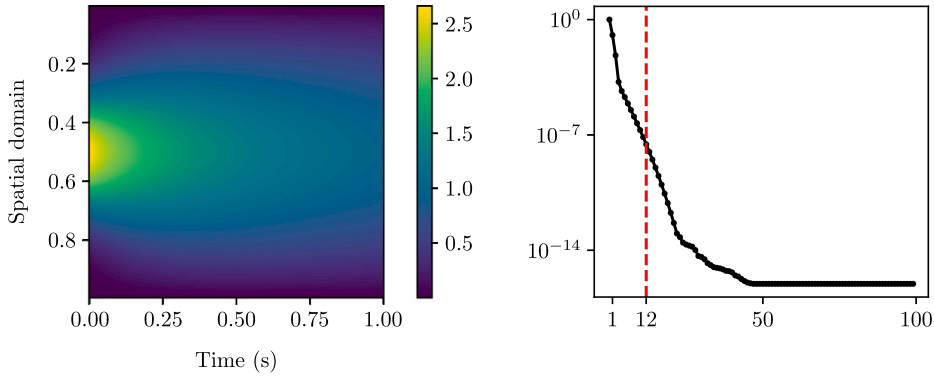
Fig. 1. Heat equation. Left: solution to the parametrized heat equation (11) at $\mu = 0.06$; Right: normalized singular values $\sigma_i/\sigma_1, i \in \{1, 2, \dots, 100\}$.

different time instances. The SVD of the resulting snapshot matrix is used to obtain the projection matrix $\mathbf{V}$. Fig. 1 illustrates the solution to (11) at $\mu = 0.06$. We also see the exponential decay of the singular values of the snapshot matrix.

For this example, we compute $\mathbf{V}$ consisting of the first 12 columns of the left singular vector matrix. For integrating (11), we use the `odeint` function available in the `scipy` package for Python. We note that `odeint` is a wrapper around the LSODA solver available in the Fortran library `odepack`. LSODA implements adaptive time-stepping. Moreover, it switches between methods for non-stiff and stiff problems automatically. We use the same solver to integrate the ROM. Once the reduced matrices are obtained through a Galerkin projection using $\mathbf{V}$, the first step involved in estimating the error is to integrate the ROM using `odeint` to evaluate and substitute the approximate solution $\widetilde{\mathbf{x}}$ at each time instance, into the `odeint` time discretization scheme to get the residual. The residual operator at the kth time instance has the general expression:

$$\mathcal{R}^k\left[\widetilde{\mathbf{x}}^k, \widetilde{\mathbf{x}}^{k-1}, \dots, \widetilde{\mathbf{x}}^{k-s}\right] \tag{12}$$

where the arguments for $\mathcal{R}^k[\cdot]$ could be the solutions at the current and past s time steps (in case the scheme used is a linear multi-step method) or at s different stage solutions (in case an s-stage Runge–Kutta scheme is used). The exact form of this expression, naturally, is dependent on the time integration method that was used to obtain the snapshots. As we are using a solver from the `odepack` library, knowing the residual operator expression is complicated and often impossible. To circumvent this, one may choose to use a *different, but known* time integration method in order to compute the residual, e.g., via (6). However, this will lead to erroneous results, as we will demonstrate next. We denote by $\mathbf{r}^k$ the output of the residual operator $\mathcal{R}^k[\cdot]$ at a given set of arguments.

Suppose, we use a different time integration scheme, say a first-order backward Euler method as an "approximation" to the true method used within `odeint`. In this case, the approximation to the true residual reads

$$\widetilde{\mathbf{r}}^k = \widetilde{\mathbf{x}}^{k-1} - \left(\mathbf{I}_N - \delta t \mathbf{A}\right)\widetilde{\mathbf{x}}^k \tag{13}$$

with the approximate residual operator $\widetilde{\mathcal{R}}^k[*, \star] = \mathbf{I}_N * - \left(\mathbf{I}_N - \delta t \mathbf{A}\right)\star$ and $*, \star$ being placeholders for the arguments of $\widetilde{\mathcal{R}}$. In general, $\mathcal{R}^k[\cdot]$ and $\widetilde{\mathcal{R}}^k[\cdot]$ are quite different, leading to rather inaccurate estimation of the error when the latter is used in (7). This is illustrated in Fig. 2. The top figure shows the estimated state error obtained using (7) for every time step and the corresponding true error, both measured in the 2-norm. The bottom figure illustrates the estimated error for the output variable and its true error.

The estimated error measured using the wrong expression of the residual overestimates by 2 orders of magnitude in the best case. In this work, we propose a scheme to suitably modify $\widetilde{\mathcal{R}}^k[\cdot]$ using a closure term, such that the resulting expression for the residual is close to the one evaluated by $\mathcal{R}^k[\cdot]$. We limit our focus to the case of output error estimation though the proposed closure technique for correcting the residual can be straight-forwardly applied to any residual-based error estimator.

3. Improving output error estimation via a data-enhanced closure approach

In this section, we introduce a data-enhanced closure strategy to ensure that the residual resulting from the user-imposed time integration scheme, viz. $\widetilde{\mathcal{R}}^k\left[\widetilde{\mathbf{x}}^k, \widetilde{\mathbf{x}}^{k-1}, \dots, \widetilde{\mathbf{x}}^{k-s}\right]$ is close to the true residual $\mathcal{R}^k\left[\widetilde{\mathbf{x}}^k, \widetilde{\mathbf{x}}^{k-1}, \dots, \widetilde{\mathbf{x}}^{k-s}\right]$ such that the estimated output error is accurate.

3.1. Defect-corrected FOM and ROM

In our proposed approach, we first add a closure term to the FOM resulting from the user-imposed time integration scheme. This closure term is derived based on the snapshots of the true solution obtained using the ODE solver library. More precisely, suppose
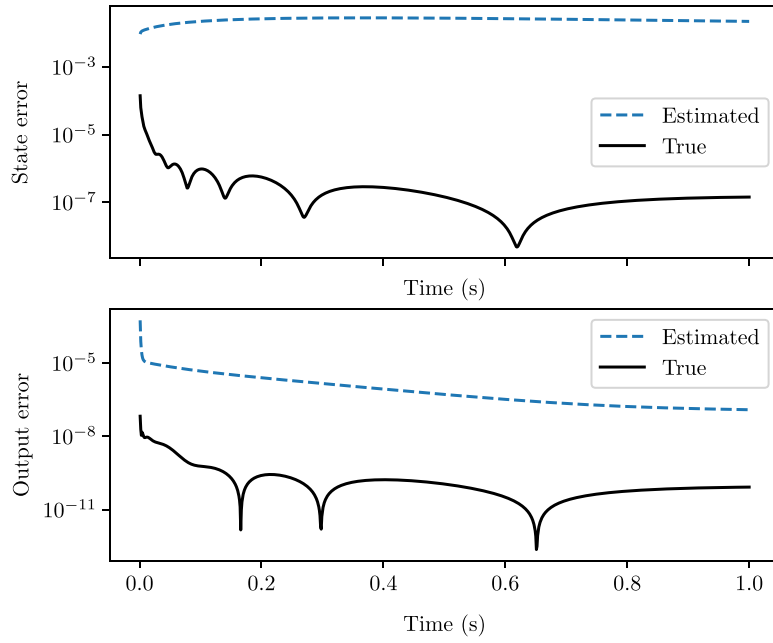
Fig. 2. Estimated and true errors for the heat equation (11) for $\mu = 0.06$ obtained by imposing a backward Euler method on the ROM snapshots obtained from `odeint`. Top: the estimated state error and the corresponding true error (see (7)); Bottom: the estimated output error and the corresponding true error (see (8)).

we have the snapshots of the solution (3) at any given parameter μ obtained from an ODE solver or some legacy code:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^0 & \mathbf{x}^1 & \dots & \mathbf{x}^K \end{bmatrix} \in \mathbb{R}^{N \times N_t}. \tag{14}$$

For purpose of illustration, we explain the details of the new method by considering a first-order IMEX scheme as the user-imposed time integration scheme. The FOM resulting from this is exactly (4).

Since the user-imposed time integration scheme differs from the one used to generate the snapshots in $\mathbf{X}$, we have a defect or a mismatch when we insert $\mathbf{x}^k$ in (14) into the first-order IMEX scheme. This reads

$$\mathbf{d}^k := \mathbf{E}_{im}\mathbf{x}^k - \left(\mathbf{A}_{im}\mathbf{x}^{k-1} + \delta t \big(\mathbf{f}(\mathbf{x}^{k-1}) + \mathbf{B}\mathbf{u}^k \big) \right). \tag{15}$$

Remark 3. We note that the quantity $\mathbf{d}^k$ converges to the local truncation error (LTE) [51] of the first-order IMEX method as the solution $\mathbf{x}^k$ converges to the true solution to (1) for a small enough step size.

We seek to modify the time-discrete FOM (4) such that its solution recovers the solution of (1) computed by an ODE solver from a library. To this end, consider the following corrected FOM (C-FOM for short) obtained by adding the defect vector $\mathbf{d}^k$ as a closure term:

$$\mathbf{E}_{im}\mathbf{x}^k_{im,c} = \mathbf{A}_{im}\mathbf{x}^{k-1}_{im,c} + \delta t \big(\mathbf{f}(\mathbf{x}^{k-1}_{im,c}) + \mathbf{B}\mathbf{u}^k \big) + \mathbf{d}^k, \tag{16a}$$

$$\mathbf{y}^k_{im,c} = \mathbf{C}\mathbf{x}^k_{im,c}. \tag{16b}$$

In (16), $\mathbf{x}^k_{im,c}$ is the solution obtained after introducing the closure term and, as such, it differs from the solution $\mathbf{x}^k_{im}$ to (4). In fact, without adding $\mathbf{d}^k$ as a closure term the local truncation error of the first-order IMEX method is $\mathcal{O}((\delta t)^2)$. However, based on the discussion in [30, Theorem 1] the local truncation error of the solution from the corrected IMEX method (16) is of the same order as that resulting from the ODE solver. It can be seen from Fig. 3 that the solution to the heat equation obtained using (16) is visually identical to the one obtained using an ODE solver.

We emphasize that the defect vector $\mathbf{d}^k$ represents one choice for the closure term. In general, any other form of the closure term may be used. Since we use the defect vector as our choice for the closure term to recover the true residual, we use the two terms interchangeably.

Remark 4. For any general time integration scheme, the defect vector $\mathbf{d}^k$ can be shown to have the following equivalence:

$$\mathbf{d}^k = -\mathcal{R}^k\big[\mathbf{x}^k, \mathbf{x}^{k-1}, \dots, \mathbf{x}^{k-s} \big]. \tag{17}$$
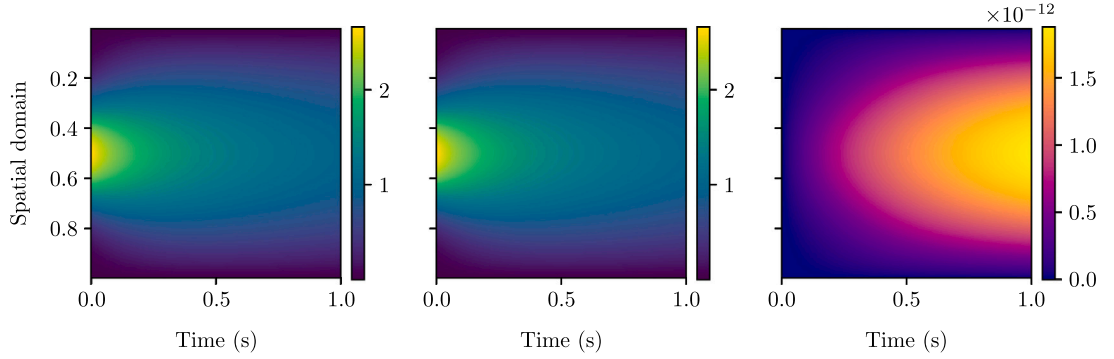
Fig. 3. Solution to the heat equation (11); Left: solution obtained using the ODE solver; Middle: solution using C-FOM (16); Right: pointwise errors between the solutions from the solver and C-FOM.

A corrected ROM (C-ROM) corresponding to the C-FOM can be defined by projecting the defect vector $\mathbf{d}^k$ to the reduced space. The reduced defect vector is defined as $\widehat{\mathbf{d}}^k := \mathbf{V}^\mathsf{T}\mathbf{d}^k$ and the C-ROM is:

$$\widehat{\mathbf{E}}_{\text{im}}\widehat{\mathbf{x}}_{\text{im,c}}^k = \widehat{\mathbf{A}}_{\text{im}}\widehat{\mathbf{x}}_{\text{im,c}}^{k-1} + \delta t\left(\widehat{\mathbf{f}}(\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1}) + \widehat{\mathbf{B}}\mathbf{u}^k\right) + \widehat{\mathbf{d}}^k, \tag{18a}$$

$$\widehat{\mathbf{y}}_{\text{im,c}}^k = \widehat{\mathbf{C}}\widehat{\mathbf{x}}_{\text{im,c}}^k. \tag{18b}$$

3.2. An error estimator using the C-ROM

Next, we make use of the C-ROM to derive a new error estimator $\overline{\Delta}^k(\boldsymbol{\mu})$ that accurately estimates the true error $\|\mathbf{y}^k(\boldsymbol{\mu}) - \widehat{\mathbf{y}}^k(\boldsymbol{\mu})\|$, where $\mathbf{y}^k(\boldsymbol{\mu})$ and $\widehat{\mathbf{y}}^k(\boldsymbol{\mu})$ are the output of the FOM (1) and the ROM (2) at time t^k, respectively. The FOM and the ROM can be solved using any ODE solver. As mentioned, to derive the residual correctly, the same ODE solver must be applied to both the FOM and ROM simulations. Our proposed error estimator makes use of a dual system. We begin by deriving the dual system for (16).

3.2.1. Dual system

We derive the dual system corresponding to the C-FOM (16) using the method of Lagrange multipliers. The Lagrangian can be formulated as

$$\mathcal{L} := \mathbf{C}\mathbf{x}_{\text{im,c}}^k + \left(\Lambda^k\right)^\mathsf{T}\left(\mathbf{E}_{\text{im}}\mathbf{x}_{\text{im,c}}^k - \mathbf{A}_{\text{im}}\mathbf{x}_{\text{im,c}}^{k-1} - \delta t\mathbf{f}(\mathbf{x}_{\text{im,c}}^{k-1}) - \delta t\mathbf{B}\mathbf{u}^k - \mathbf{d}^k\right), \tag{19}$$

with $\Lambda \in \mathbb{R}^N$ being the vector of Lagrange coefficients. The dual system can be obtained by setting $\frac{\partial \mathcal{L}}{\partial \mathbf{x}_{\text{im,c}}^k} \equiv \mathbf{0}$. This yields the system

$$\mathbf{E}_{\text{du}}\mathbf{x}_{\text{du}} = \mathbf{C}_{\text{du}} \tag{20}$$

with $\mathbf{E}_{\text{du}} := \mathbf{E}_{\text{im}}^\mathsf{T}$ and $\mathbf{C}_{\text{du}} := -\mathbf{C}^\mathsf{T}$. Note that the Lagrange multipliers Λ are the dual state variables; for better clarity we denote them by $\mathbf{x}_{\text{du}}$. Note that the defect vector $\mathbf{d}^k$ is treated as a function of time and parameter $\boldsymbol{\mu}$. Therefore, it does not depend on the solution $\mathbf{x}_{\text{im,c}}^k$ of the C-ROM.

We further define the dual ROM as

$$\widehat{\mathbf{E}}_{\text{du}}\widehat{\mathbf{x}}_{\text{du}} = \widehat{\mathbf{C}}_{\text{du}} \tag{21}$$

obtained by making the ansatz $\mathbf{x}_{\text{du}} \approx \widetilde{\mathbf{x}}_{\text{du}} = \mathbf{V}_{\text{du}}\widehat{\mathbf{x}}_{\text{du}}$. Here, $\mathbf{V}_{\text{du}}$ is the projection matrix corresponding to the dual system and $\widehat{\mathbf{E}}_{\text{du}} := \mathbf{V}_{\text{du}}^\mathsf{T}\mathbf{E}_{\text{du}}\mathbf{V}_{\text{du}}$, $\widehat{\mathbf{C}}_{\text{du}} := \mathbf{V}_{\text{du}}^\mathsf{T}\mathbf{C}_{\text{du}}$.

The residuals corresponding to the ROM (18) and the dual ROM (21) are, respectively,

$$\mathbf{r}_{\text{im,c}}^k := \mathbf{A}_{\text{im}}\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1} + \delta t\left(\mathbf{f}(\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1}) + \mathbf{B}\mathbf{u}^k\right) + \mathbf{d}^k - \mathbf{E}_{\text{im}}\widetilde{\mathbf{x}}_{\text{im,c}}^k \tag{22}$$

and

$$\mathbf{r}_{\text{du}} := \mathbf{C}_{\text{du}} - \mathbf{E}_{\text{du}}\widetilde{\mathbf{x}}_{\text{du}}. \tag{23}$$

Following the approach in [18,19], we define an auxiliary residual $\check{\mathbf{r}}_{\text{im,c}}$ as

$$\check{\mathbf{r}}_{\text{im,c}}^k := \mathbf{A}_{\text{im}}\mathbf{x}_{\text{im,c}}^{k-1} + \delta t\left(\mathbf{f}(\mathbf{x}_{\text{im,c}}^{k-1}) + \mathbf{B}\mathbf{u}^k\right) + \mathbf{d}^k - \mathbf{E}_{\text{im}}\widetilde{\mathbf{x}}_{\text{im,c}}^k. \tag{24}$$

The auxiliary residual defined above will be required in the derivation of the data-enhanced output error estimator.

3.2.2. Modified output term

In anticipation of the new error estimator we propose next, we introduce a modified output variable $\overline{\mathbf{y}}_{\text{im,c}}^{k}$ defined as

$$\overline{\mathbf{y}}_{\text{im,c}}^{k} := \widehat{\mathbf{y}}_{\text{im,c}}^{k} - \widetilde{\mathbf{x}}_{\text{du}}^{\top} \mathbf{r}_{\text{im,c}}^{k}. \tag{25}$$

Adding a correction term to the output in the form of a dual-weighted residual is an established practice [10,19,49] and serves to derive a tighter estimate of the true error.

3.2.3. Data-enhanced error estimation

Denoting the ODE solver applied to solve the FOM (1) and the ROM (2) as `solver`, the norm of the true error we desire viz., $\|\mathbf{y}^{k} - \widehat{\mathbf{y}}^{k}\|$ can be written as:

$$\|\mathbf{y}^{k} - \widehat{\mathbf{y}}^{k}\| = \|\mathbf{y}^{k} - \overline{\mathbf{y}}_{\text{im,c}}^{k} + \overline{\mathbf{y}}_{\text{im,c}}^{k} - \widehat{\mathbf{y}}^{k}\| \tag{26}$$

$$\leq \|\mathbf{y}^{k} - \overline{\mathbf{y}}_{\text{im,c}}^{k}\| + \|\overline{\mathbf{y}}_{\text{im,c}}^{k} - \widehat{\mathbf{y}}^{k}\|. \tag{27}$$

The following theorem bounds the first summand in (27):

Theorem 3.1 (*A Posteriori Error Bound for the Corrected ROM*). *Given the FOM in* (1), *the C-FOM in* (16) *and the C-ROM* (18), *assuming that* $\mathbf{E}_{\text{im}}$ *is non-singular for all* $\boldsymbol{\mu} \in \mathcal{P}$, *we have the following error bound for the modified output vector in* (25):

$$\|\mathbf{y}^{k} - \overline{\mathbf{y}}_{\text{im,c}}^{k}\| \leq \|\mathbf{E}_{\text{im}}^{-1}\| \, \|\mathbf{r}_{\text{du}}\| \, \|\check{\mathbf{r}}_{\text{im,c}}^{k}\| + \|\widetilde{\mathbf{x}}_{\text{du}}\| \, \|\mathbf{r}_{\text{im,c}}^{k} - \check{\mathbf{r}}_{\text{im,c}}^{k}\|. \tag{28}$$

Proof. See Appendix B. □

Although the bound above is rigorous, it is not computable owing to the quantity $\check{\mathbf{r}}^{k}$. Recall from (24) that its computation requires that the FOM solution $\mathbf{x}_{\text{im,c}}^{k}$ is available for any parameter $\boldsymbol{\mu}$, which is not the case. To derive a computable error estimator, we make use of the arguments used in [19] to get the following error indicator:

$$\|\mathbf{y}^{k} - \overline{\mathbf{y}}_{\text{im,c}}^{k}\| \lesssim \left(\overline{\rho} \, \|\mathbf{E}_{\text{im}}^{-1}\| \, \|\mathbf{r}_{\text{du}}\| + |1 - \overline{\rho}| \, \|\widetilde{\mathbf{x}}_{\text{du}}\| \right) \|\mathbf{r}_{\text{im,c}}^{k}\|. \tag{29}$$

The quantity $\overline{\rho}$ is a measure for how close the residual $\mathbf{r}_{\text{im,c}}^{k}$ is to the auxiliary residual $\check{\mathbf{r}}_{\text{im,c}}^{k}$ and is defined as

$$\overline{\rho} = \frac{1}{K} \sum_{k=1}^{K} \rho^{k}, \qquad \rho^{k} = \frac{\|\check{\mathbf{r}}_{\text{im,c}}^{k}(\boldsymbol{\mu}^{*})\|}{\|\mathbf{r}_{\text{im,c}}^{k}(\boldsymbol{\mu}^{*})\|}. \tag{30}$$

It is evaluated only at the greedy parameter $\boldsymbol{\mu}^{*}$ for which snapshots of the true solution $\mathbf{x}^{k}$ are available. For additional details, we refer to [19]. Substituting (29) into (27) results in

$$\|\mathbf{y}^{k} - \widehat{\mathbf{y}}^{k}\| \lesssim \left(\overline{\rho} \, \|\mathbf{E}_{\text{im}}^{-1}\| \, \|\mathbf{r}_{\text{du}}\| + |1 - \overline{\rho}| \, \|\widetilde{\mathbf{x}}_{\text{du}}\| \right) \|\mathbf{r}_{\text{im,c}}^{k}\| + \|\overline{\mathbf{y}}_{\text{im,c}}^{k} - \widehat{\mathbf{y}}^{k}\| =: \overline{\Delta}_{a}^{k}(\boldsymbol{\mu}). \tag{31}$$

The second quantity in the above inequality is the norm of the error between the output resulting from the user-imposed C-ROM and that obtained by solving the ROM (2) with `solver`. It can be obtained cheaply as only two ROMs with small sizes need to be solved. Typically, our numerical experiments show that this quantity is very small and less than the magnitude of the first quantity. Therefore, we can safely neglect it so that only the first quantity can act as an alternative form of the proposed data-enhanced error estimator, i.e.,

$$\|\mathbf{y}^{k}(\boldsymbol{\mu}) - \widehat{\mathbf{y}}^{k}(\boldsymbol{\mu})\| \lesssim \|\mathbf{y}^{k}(\boldsymbol{\mu}) - \overline{\mathbf{y}}_{\text{im,c}}^{k}(\boldsymbol{\mu})\| \tag{32}$$

$$\lesssim \left(\overline{\rho} \, \|\mathbf{E}_{\text{im}}^{-1}\| \, \|\mathbf{r}_{\text{du}}(\boldsymbol{\mu})\| + |1 - \overline{\rho}| \, \|\widetilde{\mathbf{x}}_{\text{du}}(\boldsymbol{\mu})\| \right) \|\mathbf{r}_{\text{im,c}}^{k}(\boldsymbol{\mu})\| =: \overline{\Delta}_{b}^{k}(\boldsymbol{\mu}). \tag{33}$$

3.2.4. Error estimation in presence of hyperreduction

For nonlinear systems, the efficient computation of the ROM is impeded by the presence of the nonlinear function $\mathbf{f}(\widetilde{\mathbf{x}}_{\text{im,c}}^{k})$ whose evaluation scales with the dimension N of the FOM. To tackle this, the DEIM approach [35] is used in this work. Using DEIM, the residual expression in (22) gets modified as below:

$$\mathbf{r}_{\text{im,c}}^{k} = \mathbf{A}_{\text{im}}\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1} + \delta t \big(\mathbf{f}(\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1}) + \mathcal{I}[\mathbf{f}(\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1})] - \mathcal{I}[\mathbf{f}(\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1})] + \mathbf{B}\mathbf{u}^{k} \big) + \mathbf{d}^{k} - \mathbf{E}_{\text{im}}\widetilde{\mathbf{x}}_{\text{im,c}}^{k},$$

$$= \underbrace{\left(\mathbf{A}_{\text{im}}\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1} + \delta t \big(\mathcal{I}[\mathbf{f}(\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1})] + \mathbf{B}\mathbf{u}^{k} \big) + \mathbf{d}^{k} - \mathbf{E}_{\text{im}}\widetilde{\mathbf{x}}_{\text{im,c}}^{k} \right)}_{\mathbf{r}_{\text{im,c},\mathcal{I}}^{k}} + \underbrace{\delta t \left(\mathbf{f}(\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1}) - \mathcal{I}[\mathbf{f}(\widetilde{\mathbf{x}}_{\text{im,c}}^{k-1})] \right)}_{\mathbf{e}_{H}^{k}}. \tag{34}$$

In the equation above, $\mathcal{I}[\cdot]$ denotes the hyperreduction operator. Further, $\mathbf{e}_{H}^{k}$ refers to the error introduced by hyperreduction at the kth time step. For a detailed discussion on the computational aspects, we refer to the work [19], where the simultaneous adaptive construction of the RBM and the DEIM bases vectors are also discussed. This adaptive basis construction approach is also implemented in our numerical experiments.
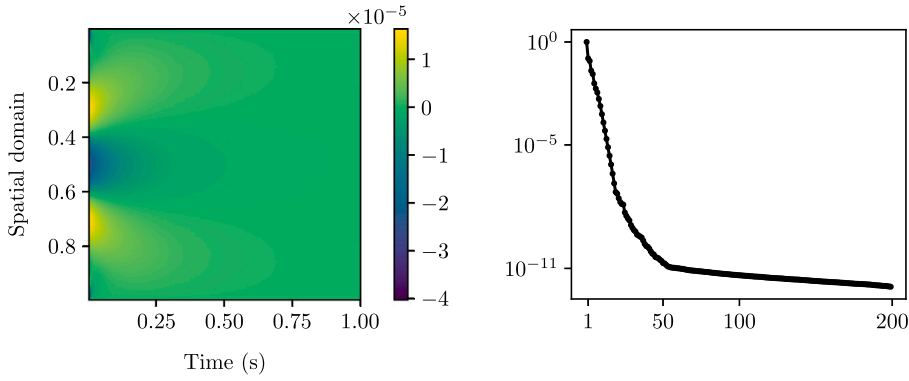
Fig. 4. Defect/local truncation error of the heat equation. Left: space time variation of the defect at $\mu = 0.06$; Right: singular values of the matrix $\mathbf{D} := \{\mathbf{d}^k\}_{k=0}^{K}$. The singular values exhibit an exponential decay illustrating the existence of a low-dimensional subspace.

We make use of the new data-enhanced error estimators $(\overline{\Delta}_a^k(\boldsymbol{\mu})$ or $\overline{\Delta}_b^k(\boldsymbol{\mu}))$ to choose the next parameter within the greedy algorithm. For each parameter $\boldsymbol{\mu}$, we determine the average estimated error over time as:

$$\overline{\Delta}_z(\boldsymbol{\mu}) = \frac{1}{N_t} \sum_{k=0}^{N_t} \overline{\Delta}_z^k(\boldsymbol{\mu}), \tag{35}$$

where z stands for a or b. In the numerical results, we use $\overline{\Delta}_b^k(\boldsymbol{\mu})$ in (35) in Line 11 of Algorithm 2 so that during the greedy iterations, the ROM (2) does not have to be repeatedly solved with `solver` in order to evaluate the second quantity in (31).

Remark 5. While we have illustrated the data-enhanced error estimator using a first-order IMEX time integration scheme, any consistent time integration scheme can be used, including higher order ones. We will also demonstrate the use of a second-order IMEX scheme in the numerical examples. Of course, one should be cautious of the fact that a higher order time integration scheme comes with a larger computational cost.

Remark 6. When the defect vector $\mathbf{d}^k$ present in the residual (see (22)) is known exactly, i.e., the C-ROM uses the same time integration method used for the FOM, we recover the *a posteriori* output error estimator proposed in [19]. This shows that our newly proposed error estimator is consistent with the case where the time integration method used is known.

Remark 7. Note that the C-ROM is not the finally derived ROM, but is only used within the greedy algorithm to derive the error estimator $\overline{\Delta}_z^k(\boldsymbol{\mu})$ (see Steps 9–10 in Algorithm 2) which estimates the error between the solution of the FOM (1) and that of the ROM (2), both being computed using any `solver`. The error estimator can also be used in the online stage where the user may wish to use any preferred `solver` to compute the ROM solutions. Moreover, given a good approximation of the defect vector, the solution of the C-ROM and the solution $\hat{\mathbf{x}}^k$ from the ODE solver are nearly the same. Therefore, $\overline{\Delta}_b^k(\boldsymbol{\mu})$ is almost as accurate as $\overline{\Delta}_a^k(\boldsymbol{\mu})$.

4. Computational aspects

While we have derived an accurate data-enhanced *a posteriori* output error estimator, there remain a few computational challenges. In this section we highlight these challenges and propose efficient solutions to address them.

In (33), the residual term $\mathbf{r}_{\text{im,c}}^k$ involves the defect vector $\mathbf{d}^k$ (see (22)). However, determining this term involves knowing the true solution to (1) obtained from the ODE solver at any parameter $\boldsymbol{\mu}$ (see (15)). Without a cheap method to approximate $\mathbf{d}^k$ for every parameter $\boldsymbol{\mu}$, the error estimator is not efficient. To alleviate this, we make several observations about the function $\mathbf{d}(t, \boldsymbol{\mu})$, which will lead to its efficient approximation. These observations relate to

1. a certain low-rank structure (over space) that $\mathbf{d}(t, \boldsymbol{\mu})$ possesses and
2. a smoothness over parameter variations that it inherits from the underlying parametric PDE.

4.1. Low-rank structure of the defect

For a large class of problems, there exist a low-dimensional subspace onto which the solution snapshots can be projected, without incurring a large error. Indeed, this forms the underlying motivation for performing model order reduction using POD and other methods. Leveraging this fact, we assume that the defect trajectory at a given parameter $\boldsymbol{\mu}$, $\mathbf{d}^k(\boldsymbol{\mu}), \forall k \in \{0, 1, \ldots, K\}$ can also be efficiently approximated in a suitable low-dimensional subspace.

Fig. 4 plots the values of the defect vector $\mathbf{d}^k$ in (15) for the heat equation. It is evident that the defect trajectory has a certain regularity over the spatial domain (see left figure); evidence for the low-rank structure is also seen through the SVD performed (see right figure) on the snapshot matrix $\mathbf{D} \in \mathbb{R}^{N \times N_t}$ whose columns consist of snapshots of $\mathbf{d}^k$ at different values of the time $t^k, \forall k \in \{0, \dots, K\}$. We see an exponential decay of the relative quantity σ_i / σ_1, where σ_i is the ith singular value. A similar exercise will be repeated in Section 5 for the three numerical examples we consider. In each of the cases, it will be seen that the defect trajectory is reducible spatially. We will use this fact to compute $\mathbf{d}^k(\boldsymbol{\mu})$ efficiently.

The concept of Kolmogorov n-width is used in the RBM literature [7,52,53] to quantify the approximability of the solution manifold $\mathcal{M}$ corresponding to a parametrized system of equations, with a linear subspace of dimension n denoted as $\mathcal{V}^n$. Consider the solution manifold $\mathcal{M}$ for the FOM in (1) defined as

$$\mathcal{M} = \{\mathbf{x}(t, \boldsymbol{\mu}) \; : \; (t, \boldsymbol{\mu}) \in \mathcal{T} \times \mathcal{P}\} \subset \mathbb{R}^N. \tag{36}$$

The Kolmogorov n-width of $\mathcal{M}$ using $\mathcal{V}^n$ can be defined as

$$d_n(\mathcal{M}) := \inf_{\dim(\mathcal{V}^n) = n} \sup_{\mathbf{x} \in \mathcal{M}} \inf_{\widetilde{\mathbf{x}} \in \mathcal{V}^n} \|\mathbf{x} - \widetilde{\mathbf{x}}\|. \tag{37}$$

For the parametrized defect function $\mathbf{d}(t, \boldsymbol{\mu})$, we define the following manifold:

$$\mathcal{M}_d = \{\mathbf{d}(t, \boldsymbol{\mu}) \; : \; (t, \boldsymbol{\mu}) \in \mathcal{T} \times \mathcal{P}\} \subset \mathbb{R}^N. \tag{38}$$

From our numerical examples we have observed that $\mathcal{M}_d$ can be well-approximated by low-dimensional subspaces when the original parametric problem has fast Kolmogorov n-width decay. This implicates that the Kolmogorov n-width of $\mathcal{M}_d$ may inherit the behaviour of $d_n(\mathcal{M})$. At present we cannot strictly prove this, but from the definition of the defect vector in (15), the defect vector $\mathbf{d}(t, \boldsymbol{\mu})$ can be seen as being in the image of the operator $\mathcal{D} : \mathbb{R}^n \times \mathcal{P} \mapsto \mathbb{R}^n$, where $\mathcal{D}[\mathbf{x}(t, \boldsymbol{\mu})] = \mathbf{E}_{\text{im}}\mathbf{x}(t, \boldsymbol{\mu}) - \left(\mathbf{A}_{\text{im}}\mathbf{x}(t - \delta t, \boldsymbol{\mu}) + \delta t(\mathbf{f}(\mathbf{x}(t - \delta t, \boldsymbol{\mu})) + \mathbf{B}\mathbf{u}(t))\right)$. With this observation, the inheritance property of the Kolmogorov n-width of $\mathcal{M}_d$ might be proved based on Theorem 4.1 in [54]. We leave this as our future work.

4.2. Strategies to approximate the defect

As discussed in the previous section, the defect vector is assumed to possess a certain low-rank structure in space and it inherits the smoothness of the solution $\mathbf{x}(t, \boldsymbol{\mu})$ over parameter variations, from the underlying parametric PDE. We use these two observations to make the approximation of $\mathbf{d}^k(\boldsymbol{\mu})$ computationally efficient, such that the output error estimator defined in (33) can be used in the POD-Greedy algorithm. To this end, we adopt a two-stage approximation strategy.

Starting from the observation about the low-rank structure of the defect, we can approximate the defect vector at a given time and parameter using the basis expansion

$$\mathbf{d}(t, \boldsymbol{\mu}) \approx \widetilde{\mathbf{d}}(t, \boldsymbol{\mu}) = \sum_{i=1}^{n_d} \mathbf{v}_{d,i} \, \widehat{d}_i(t, \boldsymbol{\mu}) \tag{39}$$

where $\mathbf{v}_{d,i} \in \mathbb{R}^N$ are the expansion basis vectors and

$$\widehat{\mathbf{d}}(t, \boldsymbol{\mu}) := [\widehat{d}_1(t, \boldsymbol{\mu}), \dots, \widehat{d}_{n_d}(t, \boldsymbol{\mu})]^\top \in \mathbb{R}^{n_d}$$

is the vector of expansion coefficients. We denote with

$$\mathbf{V}_d := \left[\mathbf{v}_{d,1}, \mathbf{v}_{d,2}, \dots, \mathbf{v}_{d,n_d}\right] \in \mathbb{R}^{N \times n_d}$$

the basis matrix. If the observation regarding the rapid decay of the singular values holds, then n_d will be small. Given such a basis expansion for the defect vector, we can approximate it for any given parameter $\boldsymbol{\mu}$ and a time instance t if $\widehat{\mathbf{d}}(t, \boldsymbol{\mu})$ can be evaluated cheaply. Our two-stage approach involves:

- identifying a suitable basis matrix $\mathbf{V}_d$ using a POD/SVD-based approach and
- learning the map $(t, \boldsymbol{\mu}) \mapsto \widehat{\mathbf{d}}(t, \boldsymbol{\mu})$ for which we propose two different approaches: one based on radial basis function interpolation and the other using a feed-forward neural network.

4.2.1. SVD-based spatial reduction

In the first stage of approximation, we collect snapshots of the defect vector $\mathbf{d}(t, \boldsymbol{\mu}), \forall t \in \mathcal{T}_d := \{t^1, t^2, \dots, t^{N_t}\}$ and $\boldsymbol{\mu} \in \Xi_{\text{defect}}$ where Ξ_{defect} is a set containing d_s parameter samples, with d_s typically small. Doing so involves solving (1) with `solver` to obtain the FOM solution snapshots. Following this, the solution snapshots can be used to obtain the defect vector from e.g., (15).[1] We denote by $\mathfrak{D} \in \mathbb{R}^{N \times N_t \times d_s}$ the third-order tensor arranged such that each of its frontal slices corresponds to the matrix $\mathbf{D}(\boldsymbol{\mu}) := \left[\mathbf{d}(t^1, \boldsymbol{\mu}), \mathbf{d}(t^2, \boldsymbol{\mu}), \dots, \mathbf{d}(t^{N_t}, \boldsymbol{\mu})\right] \in \mathbb{R}^{N \times N_t}$ and $\boldsymbol{\mu} \in \Xi_{\text{defect}}$. We refer to the ith frontal slice as $\mathfrak{D}^{(i)}$. Next, we apply a two-step SVD reduction [55] to $\mathfrak{D}$ which will result in $\widehat{\mathfrak{D}} \in \mathbb{R}^{n_d \times N_t \times d_s}$ as follows:

---

[1] Note that the defect is obtained here for a particular user-imposed time integration scheme.

- In step 1, we perform SVDs for each frontal slice separately, i.e., we perform a SVD for each $\mathfrak{D}^{(i)}$ with $i \in \{1, 2, \ldots, d_s\}$. For every such SVD, we make use of a fixed tolerance $\texttt{tol}_{\text{SVD},t}$ to truncate the singular values and collect the first ℓ_i left singular vectors in the matrix $\mathbf{U}_{d,i} \in \mathbb{R}^{N \times \ell_i}$.

- In step 2, we form a matrix $\mathbf{R}$ defined as

$$\mathbf{R} := \left[\mathbf{U}_{d,1} \, \middle| \, \mathbf{U}_{d,2} \, \middle| \, \cdots \, \middle| \, \mathbf{U}_{d,d_s} \right] \in \mathbb{R}^{N \times (\ell_1 + \cdots + \ell_{d_s})}$$

whose columns consist of the truncated left singular vectors for each parameter obtained in step 1. We then perform the SVD of $\mathbf{R}$, using a tolerance $\texttt{tol}_{\text{SVD},\boldsymbol{\mu}}$ to obtain the projection matrix $\mathbf{V}_d \in \mathbb{R}^{N \times n_d}$ as the first n_d columns of the left singular vectors of $\mathbf{R}$.

Finally, the reduced tensor $\widehat{\mathfrak{D}}$ can be obtained via a *mode-1* tensor–matrix product as

$$\widehat{\mathfrak{D}} = \mathfrak{D} \times_1 \mathbf{V}_d^\top.$$

We note that each mode-1 fiber in $\widehat{\mathfrak{D}}$ corresponds to the reduced defect vector $\widehat{\mathbf{d}}(t^k, \boldsymbol{\mu}) \in \mathbb{R}^{n_d}$.

Thus far, we have reduced the dimension of the first mode of the tensor $\mathfrak{D}$ from N to n_d, with $n_d \ll N$. Next, we detail the two approaches used to approximate $\widehat{\mathbf{d}}(t, \boldsymbol{\mu})$.

Remark 8. The motivation for using the two-step SVD approach is to reduce the overall computational costs. While we have not pursued it in this work, another valid approach to reduce the computational cost would be a randomized-SVD.

4.2.2. Interpolation using radial basis functions

Radial basis functions (RBFs) are a popular class of kernel methods which are used in scattered-data approximation [56,57]. We use RBFs to learn an approximation $\widehat{d}_{\text{RBF}}(\boldsymbol{\mu})$ to each entry of the reduced defect vector at each time instance, such that for any given (j, k) pair, $\widehat{d}_{\text{RBF}}(\boldsymbol{\mu})$ interpolates $\widehat{d}_j(t^k, \boldsymbol{\mu})$, the jth entry of the reduced defect $\widehat{\mathbf{d}}(t^k, \boldsymbol{\mu})$ at parameters $\boldsymbol{\mu} \in \Xi_{\text{defect}}$.

The RBF approximation reads

$$\widehat{d}_{\text{RBF},j}^k(\boldsymbol{\mu}) = \sum_{i=1}^{d_s} w_i \Phi(\|\boldsymbol{\mu} - \boldsymbol{\mu}_i\|) \tag{40}$$

with $\{w_i\}_{i=1}^{d_s}$ denoting the weights and $\Phi(\cdot)$ being the radial basis functions. The weights are obtained by imposing the interpolation condition $\widehat{d}_{\text{RBF},j}^k(\boldsymbol{\mu}_i) = \widehat{d}_j(t^k, \boldsymbol{\mu}_i)$, $i = 1, 2, \ldots, d_s$, $\boldsymbol{\mu}_i \in \Xi_{\text{defect}}$. This leads to the following system of linear equations

$$\begin{bmatrix} \Phi(\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_1\|) & \Phi(\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|) & \cdots & \Phi(\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_{d_s}\|) \\ \Phi(\|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\|) & \Phi(\|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_2\|) & \cdots & \Phi(\|\boldsymbol{\mu}_2 - \boldsymbol{\mu}_{d_s}\|) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi(\|\boldsymbol{\mu}_{d_s} - \boldsymbol{\mu}_1\|) & \Phi(\|\boldsymbol{\mu}_{d_s} - \boldsymbol{\mu}_2\|) & \cdots & \Phi(\|\boldsymbol{\mu}_{d_s} - \boldsymbol{\mu}_{d_s}\|) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{d_s} \end{bmatrix} = \begin{bmatrix} \widehat{d}_j(t^k, \boldsymbol{\mu}_1) \\ \widehat{d}_j(t^k, \boldsymbol{\mu}_2) \\ \vdots \\ \widehat{d}_j(t^k, \boldsymbol{\mu}_{d_s}) \end{bmatrix}. \tag{41}$$

Based on the RBF interpolation, the defect vector $\mathbf{d}(t^k, \boldsymbol{\mu})$ in (39) can be approximated as

$$\mathbf{d}(t^k, \boldsymbol{\mu}) \approx \widetilde{\mathbf{d}}_{\text{RBF}}(t^k, \boldsymbol{\mu}) := \sum_{j=1}^{n_d} \mathbf{v}_{d,j} \, \widehat{d}_{\text{RBF},j}^k(\boldsymbol{\mu}). \tag{42}$$

We obtain an RBF interpolant for each time instance $k \in \{0, 1, 2, \ldots, K\}$ and each coordinate $j \in \{1, 2, \ldots, n_d\}$, resulting in a total of $(n_d \cdot N_t)$ RBF interpolants. Fig. 5 graphically illustrates the approach. Theoretically, $\widetilde{\mathbf{d}}_{\text{RBF}}(t^k, \boldsymbol{\mu})$ in (42) is valid for any $\boldsymbol{\mu} \in \mathcal{P}$. Therefore, N_p in Fig. 5 can be arbitrarily large. We denote it by N_p as we are only interested in obtaining the reduced defect coefficients corresponding to the N_p parameter samples present in the training set Ξ.

In our numerical results in Section 5, we denote this method of approximating the defect vector using a SVD spatial reduction followed by an RBF interpolation as SVD+RBF.

Remark 9. In this work, we have considered separate RBF interpolants for each time step and each generalized spatial coordinate, leading to potentially many interpolants. While the number of interpolants scales as $\mathcal{O}(n_d N_t)$, this can be efficiently implemented in one step by solving a linear system with the small $d_s \times d_s$ coefficient matrix in (41), and with multiple right-hand sides (totalling $n_d N_t$). The runtime of solving such a linear system is usually much faster than separately solving $n_d N_t$ linear systems with the same coefficient matrix.

4.2.3. Approximation using artificial neural networks

The second approach we consider to approximate the expansion coefficients $\widehat{\mathbf{d}}(t, \boldsymbol{\mu})$ for different time and parameter values is based on artificial neural networks (ANNs). A widely used architecture to implement ANNs are the feed-forward neural networks (FNNs). FNNs have been shown to be efficient for both regression and classification tasks in a variety of applications. The basic architecture of an FNN consists of three components: an input layer, hidden layer(s) and an output layer. The core element of any
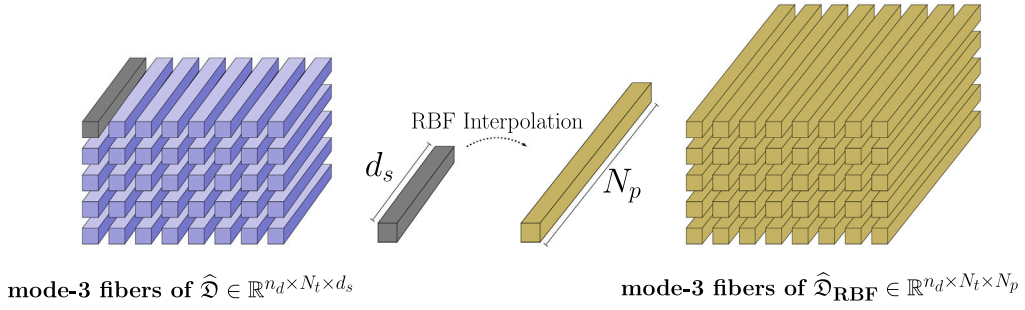
Fig. 5. Approximation of the defect as a function of the parameter $\boldsymbol{\mu}$. The RBF interpolant learns an approximation of the defect vector over N_p samples, with interpolation occurring at d_s samples. We construct an individual RBF interpolant for each time and generalized spatial coordinate.
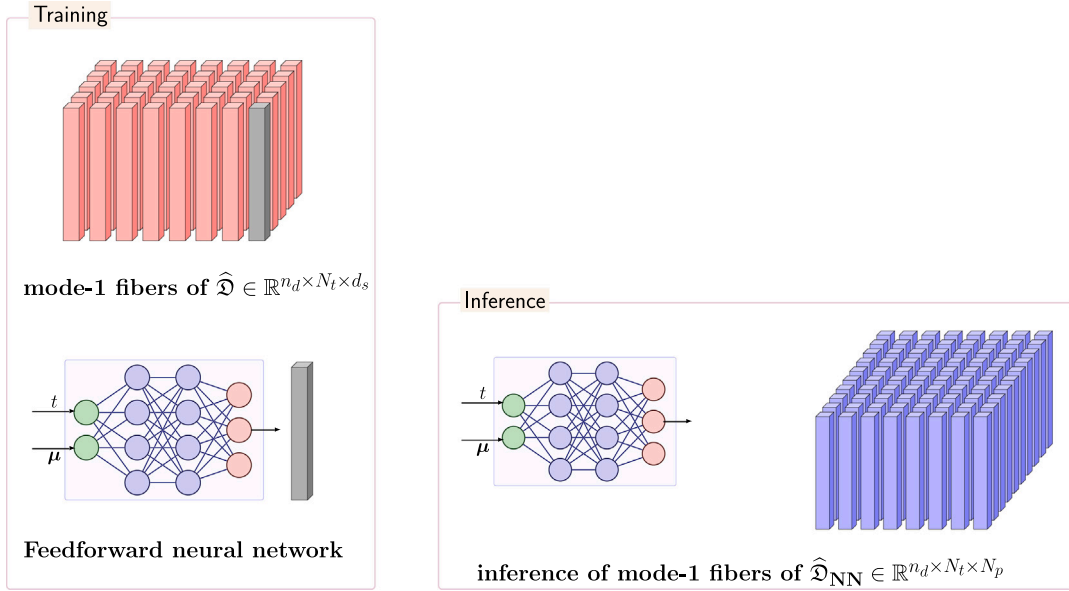

Fig. 6. Approximation of the defect coefficients as a function of the inputs $(t, \boldsymbol{\mu})$. The neural network is trained based on data available at d_s parameter samples. In the inference stage, the neural network learns the approximation of the defect for all N_p parameter samples.

NN in general and FNNs in particular are artificial neurons. The hidden layer(s) in an FNN consists of neurons stacked together. Each neuron can receive inputs from a previous layer. The output corresponds to a nonlinear function of the weighted sum of its input signals. For a detailed overview of ANNs and FNNs, we refer to [58].

In this work, we consider the FNN to learn an approximation to the map between the inputs $(t, \boldsymbol{\mu})$ and the output $\widehat{\mathbf{d}}(t, \boldsymbol{\mu}) \in \mathbb{R}^{n_d}$. That is,

$$\widehat{\mathbf{d}}_{\mathrm{NN}} : (t, \boldsymbol{\mu}) \mapsto \widehat{\mathbf{d}}_{\mathrm{NN}}(t, \boldsymbol{\mu}) \approx \widehat{\mathbf{d}}(t, \boldsymbol{\mu}).$$

Fig. 6 graphically illustrates the approach. To train the FNN, our training data consists of the dataset $\tau_{\mathrm{train}} := (\Lambda_{\mathrm{NN}}, D_{\mathrm{NN}})$ where $\Lambda_{\mathrm{NN}} := \{(t_i, \boldsymbol{\mu}_j)\}_{i,j=1}^{N_t, d_s}$ is the input set, containing time and parameter samples and $D_{\mathrm{NN}} := \{\widehat{\mathbf{d}}(t_i, \boldsymbol{\mu}_j)\}_{i,j=1}^{N_t, d_s}$ is the output data set which consists of the reduced defect $\widehat{\mathbf{d}}(t_i, \boldsymbol{\mu}_j) \in \mathbb{R}^{n_d}$ at each time and parameter sample of the input. The neural network is implemented in PyTorch; more details regarding the number of layers used and other hyperparameters will be provided in the numerical section. The loss function is the mean square loss, viz.,

$$L_{\mathrm{mse}} := \frac{1}{2} \sum_{i=1}^{d_s} \sum_{k=1}^{N_t} \left\| \widehat{\mathbf{d}}(t^k, \boldsymbol{\mu}_i) - \widehat{\mathbf{d}}_{\mathrm{NN}}(t^k, \boldsymbol{\mu}_i) \right\|_2^2.$$

Once the neural network is trained, it can infer the values of the defect vector $\widehat{\mathbf{d}}_{\mathrm{NN}}(t, \boldsymbol{\mu})$ at any chosen $(t, \boldsymbol{\mu})$. The original defect vector at a given parameter $\boldsymbol{\mu}$ and at any time instance $\mathbf{d}(t, \boldsymbol{\mu})$ can be approximated using the FNN-based approach as

$$\mathbf{d}(t, \boldsymbol{\mu}) \approx \widetilde{\mathbf{d}}_{\mathrm{NN}}(t, \boldsymbol{\mu}) := \mathbf{V}_d \, \widehat{\mathbf{d}}_{\mathrm{NN}}(t, \boldsymbol{\mu}). \tag{43}$$

Algorithm 2 POD-Greedy algorithm for ODE solver libraries

Input: Training set Ξ, tolerance (tol), discretized model $(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{f}, \mathbf{x}_0)$

Input for defect approximation: Training set for defect approximation $\Xi_{\text{defect}} \subset \Xi$, SVD tolerances $(\text{tol}_{\text{SVD},t}, \text{tol}_{\text{SVD},\boldsymbol{\mu}})$, user-imposed time integration method $\text{solver}_{\text{imp}}$

Output: V.

1: Initialize $\mathbf{V} = [\ ]$, $\epsilon = 1 + \text{tol}$, choose solver from a library, greedy parameter $\boldsymbol{\mu}^*$

2: Compute FOM snapshots for all $\boldsymbol{\mu} \in \Xi_{\text{defect}}$; determine the defect data tensor $\mathfrak{D}$ (see Section 4.2.1)

3: Perform SVD on $\mathfrak{D}$ to get $\mathbf{V}_d$; obtain an approximation $\widetilde{\mathbf{d}}_{\text{RBF}}(t, \boldsymbol{\mu})$ (42) or $\widetilde{\mathbf{d}}_{\text{NN}}(t, \boldsymbol{\mu})$ (43) for $\mathbf{d}(t, \boldsymbol{\mu})$

4: **while** $\epsilon > \text{tol}$ **do**

5:    Obtain FOM snapshots $\mathbf{X}_{\boldsymbol{\mu}^*}$ at $\boldsymbol{\mu}^*$ with solver

6:    Compute true defect $\{\mathbf{d}(t^k, \boldsymbol{\mu})\}_{k=1}^{N_t}$ at $\boldsymbol{\mu}^*$ based on $\mathbf{X}^*$

7:    Determine $\mathbf{V}^*$ through an SVD of $\overline{\mathbf{X}} := \mathbf{X}^* - \mathbf{V}(\mathbf{V}^\mathsf{T}\mathbf{X}^*)$

8:    Update $\mathbf{V}$ as $\mathbf{V} := \text{orth}\big(\mathbf{V}, \mathbf{V}^*(:, 1 : r_c)\big)$
       with $\text{orth}(\cdot)$ denoting an orthogonalization process which can be implemented using the modified Gram–Schmidt process, or QR factorization

9:    Obtain the C-ROM (e.g., (18)) corresponding to $\text{solver}_{\text{imp}}$ by Galerkin projection (+ hyperreduction)

10:    Solve the C-ROM for all $\boldsymbol{\mu} \in \Xi$ to obtain the residual and compute the error estimator $\overline{\Delta}_a(\boldsymbol{\mu})$ (31) or $\overline{\Delta}_b(\boldsymbol{\mu})$ (33)

11:    $\boldsymbol{\mu}^* := \arg\max_{\boldsymbol{\mu} \in \Xi} \overline{\Delta}_z(\boldsymbol{\mu})$, $z = a$ or $z = b$

12:    Set $\epsilon = \overline{\Delta}_z(\boldsymbol{\mu}^*)$, $z = a$ or $z = b$

13: **end while**

In our numerical results in Section 5, we denote this method of approximating the defect vector first with a SVD spatial reduction followed by an approximation of the coefficients $\widehat{\mathbf{d}}$ with a feedforward neural network as SVD+FNN.

4.3. POD-Greedy with black-box ODE solvers

We present the POD-Greedy algorithm that supports black-box ODE solvers and incorporates the new data-enhanced *a posteriori* error estimator as Algorithm 2. It requires some additional inputs compared to Algorithm 1. These include a separate training set $\Xi_{\text{defect}} \subset \Xi$ for the defect approximation and two separate tolerances for the SVD, $\text{tol}_{\text{SVD},t}, \text{tol}_{\text{SVD},\boldsymbol{\mu}}$ which are required for the two-step SVD method to compute $\mathbf{V}_d$. Before starting the greedy algorithm in Step 4, Steps 2–3 in Algorithm 2 are targeted towards learning the defect vector, which is added as a closure term to get the C-ROM. In Step 2, FOM solutions of (1) are obtained for the d_s parameter samples in the training set Ξ_{defect}. Using this data, the defect vectors (see (15)) induced by a user-imposed time integration scheme (denoted $\text{solver}_{\text{imp}}$) are collected in the tensor $\mathfrak{D}$. Then, in Step 3, the data in $\mathfrak{D}$ are first compressed into a low-dimensional space and the reduced defect vectors $\widehat{\mathbf{d}}(t, \boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi$ are learned via RBF or FNN. Afterwards, the defect vector $\mathbf{d}(t, \boldsymbol{\mu})$ is approximated via the decoded vectors $\widetilde{\mathbf{d}}_{\text{RBF}}(t, \boldsymbol{\mu})$ or $\widetilde{\mathbf{d}}_{\text{NN}}(t, \boldsymbol{\mu})$. The approximation $\widetilde{\mathbf{d}}_{\text{RBF}}(t, \boldsymbol{\mu})$ or $\widetilde{\mathbf{d}}_{\text{NN}}(t, \boldsymbol{\mu})$ can be updated (replaced) by the true defect vector once new FOM data is available at $\boldsymbol{\mu}^*$ (Step 6, Step 11). Updating the approximate defect vector with the available true defect vector at $\boldsymbol{\mu}^*$ at each iteration leads to considerable improvements in the performance of Algorithm 2, as we shall demonstrate in Section 5. Furthermore, it reduces the amount of initial training samples needed in Ξ_{defect}. Typically, the user may not know, *a priori*, the number of FOM samples needed to get a good approximation of the closure term. Therefore, Ξ_{defect} can be coarsely sampled to keep the computational cost low. Once the greedy algorithm starts, the FOM solution snapshots at $\boldsymbol{\mu}^*$ chosen at each greedy iteration are readily available. Those snapshots can be further used to compute the true defect vector $\mathbf{d}(t^k, \boldsymbol{\mu}^*)$ via (15). Since the snapshots are anyway available, the only computational costs incurred are those corresponding to the evaluation of $\mathbf{d}(t^k, \boldsymbol{\mu}), \forall k \in \{0, 1, \ldots, K\}$.

4.3.1. Computational cost

We now analyse the additional computational cost incurred by the proposed Algorithm 2 when compared to the standard POD-Greedy method in Algorithm 1.

To simplify things, we define the cost of solving the nonlinear FOM (1) to be $C_{\text{FOM}} := N_{\text{newton}} N_L N_t$; the cost of the linear solve at each Newton iteration, viz. N_L, will be at least $O(N)$ depending on the particular method being implemented in the solver. The cost of obtaining the defect vector at each time step (see (15)) is denoted by $C_{\mathbf{d}}$. It depends on the user-defined time-stepping scheme. The main contributions to $C_{\mathbf{d}}$ are matrix–vector products and evaluations of the nonlinearity. Thus, it evaluates to $C_{\mathbf{d}} := (\mathcal{O}(N^2) + \mathcal{O}(N))$ in the worst-case . However, the matrix–vector multiplications typically involve sparse matrices and can be done cheaply.

We denote by $C_{\text{SVD}} := \mathcal{O}(\min(p^2 q, pq^2))$ the cost of an SVD for a matrix of dimension $\mathbb{R}^{p \times q}$. We let $C_{\text{RBF}} := \mathcal{O}(d_s^3)$ be the cost of obtaining one RBF interpolant. The cost of training a neural network is difficult to estimate owing to its architecture and the use

of specialized hardware. Therefore, for simplicity, we define it as $C_{NN} := \sigma \cdot (C_{FP} + C_{BP})$ with C_{FP} denoting the cost of one forward pass, C_{BP} denoting the cost of one backward pass and σ being a constant that depends on the number of layers, epochs, the batch size and other hyperparameters.

The overall factors contributing to the proposed algorithm are listed below:

- Step 2 in Algorithm 2 requires the FOM solution at d_s parameters. This has cost that scales as $d_s \cdot C_{FOM}$.
- In Step 2, evaluating the defect vector at each time step and for all d_s parameter samples incurs cost that scales as $(d_s N_t) \cdot C_{\mathbf{d}}$.
- The cost of the two-stage SVD in Step 3 to obtain $\mathbf{V}_d$ is $(d_s + 1) \cdot C_{SVD}$.
- Based on the method used for approximating the map $(t, \boldsymbol{\mu}) \mapsto \hat{\mathbf{d}}(t, \boldsymbol{\mu})$, the costs differ:
    - For the RBF-based approach, the RBF coefficient matrix can be factorized once and reused for subsequent solves. The cost is thus $C_{RBF} + (n_d N_t - 1) \cdot \mathcal{O}(d_s^2)$
    - For the NN-based approach, the cost is C_{NN}
- In the inference stage where the defect vector is approximated for all $\boldsymbol{\mu} \in \Xi$,
    - the RBF-based approach has a cost $(N_p N_t) \cdot d_s$
    - the NN-based approach incurs a cost scaling as $(N_p N_t) \cdot C_{FP}$

    The cost of the matrix tensor product to obtain $\widetilde{\mathfrak{D}}$ is $d_s N_t \cdot n_d N$ for both RBF and NN-based approaches
- To update the approximate defect vector in Step 6, the cost of evaluating the true defect vector at the current greedy parameter scales as $N_t \cdot C_{\mathbf{d}}$

The major cost in approximating the defect will be the cost of solving the FOM at d_s parameter samples. From our experience, the RBF-based approach performs better than the NN-based approach. Comparisons of corresponding run times are detailed in Section 5.

5. Numerical results

To demonstrate the validity of the proposed data-enhanced output error estimation approach, we test it on three numerical examples. These are:

1. the viscous Burgers' equation with one parameter
2. the FitzHugh–Nagumo equations with two parameters and
3. the batch chromatography equations with one parameter.

All reported numerical results were performed on a desktop computer running Ubuntu 20.04, installed with a 12th generation INTEL® CORE™i5 processor, 32 GB of RAM and a NVIDIA RTX A4000 GPU with 16 GB of memory. The simulations were carried out in the Spyder IDE with Python 3.9.12 (miniconda). Where required, MATLAB® v2019b was used to run some simulations. The radial basis interpolation was performed using the RBF package [59].

For the two greedy algorithms (Algorithms 1 and 2), we plot the maximum estimated errors computed using $\overline{\Delta}_b(\boldsymbol{\mu})$ over the training set at every iteration. We define this as:

$$\varepsilon_{\max} := \max_{\boldsymbol{\mu} \in \Xi} \overline{\Delta}_b(\boldsymbol{\mu})$$

where $\overline{\Delta}_b(\boldsymbol{\mu})$ is defined in (33). Additionally, to illustrate the performances of the ROM over the test set Ξ_{test}, we plot the mean estimated error $\overline{\Delta}_b(\boldsymbol{\mu})$ for every parameter $\boldsymbol{\mu} \in \Xi_{test}$.

5.1. Code availability

The companion Python code to reproduce the numerical results is available at https://doi.org/10.5281/zenodo.8169490.

5.2. Burgers' equation

Model description. The viscous Burgers' equation defined in the 1-D domain $\Omega := [0, 1]$ is given by

$$\frac{\partial v}{\partial t} + v \frac{\partial v}{\partial z} = \mu \frac{\partial^2 v}{\partial^2 z},$$

$$v(z, 0) = \sin(2\pi z),$$

$$v(0, t) = v(1, t) = 0$$

(44)

with $v := v(z, t) \in \mathbb{R}$ denoting the state variable, where $z \in \Omega$ is the spatial variable and the time variable $t \in [0, 2]$. We spatially discretize (44) with the finite difference method. The mesh size is $\Delta z = 0.001$, which results in a discretized FOM of dimension $N = 1000$. As the variable parameter, we consider the viscosity $\mu \in \mathcal{P} := [0.005, 1]$. We sample 100 logarithmically-spaced samples from $\mathcal{P}$ and divide the samples randomly into a training set Ξ and a testing set Ξ_{test} in the ratio $80 : 20$. To solve the ODE, the solver in Algorithm 2 is the scipy library odeint. To have a uniform comparison, we compute solutions of the FOM on a uniformly spaced time step of $\delta t = 0.01$. The output variable of interest is the value of the state at the node just before the right boundary.
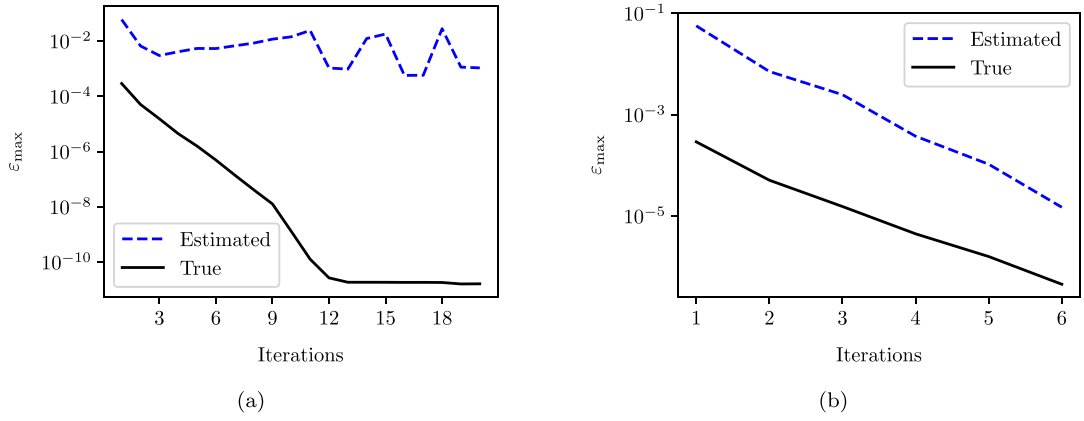
Fig. 7. Burgers' equation, Algorithm 1: (a) error (estimator) decay when using a first-order IMEX method without any closure term; (b) error (estimator) decay when the true residual is known.
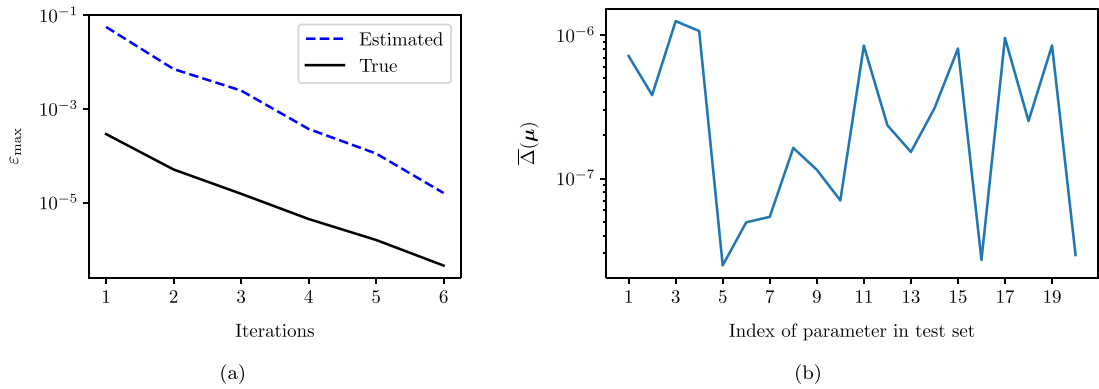

Fig. 8. Burgers' equation, Algorithm 2: (a) error (estimator) decay when using the SVD+RBF method to approximate the closure term and Step 6 is not included; (b) performance of the ROM over a test set.

POD-Greedy with Algorithm 1. We first apply Algorithm 1 to the Burgers' equation with a tolerance `tol` $= 10^{-4}$. The `solver` used is the `odeint` library in `scipy`. Since the exact expression of the residual is unknown, we use a first-order IMEX method (IMEX1) to approximate the residual and estimate the output error. As the true residual is incorrectly approximated by the user-imposed time integration scheme, the resulting estimated error is severely overestimated. This results in the stagnation of the greedy algorithm as seen in Fig. 7(a). For comparison, we also plot in Fig. 7(b) the convergence of the greedy algorithm when the exact residual is known. To remedy this, we next apply the proposed Algorithm 2.

POD-Greedy with Algorithm 2 and SVD+RBF closure approximation. First, we collect $d_s = 16$ uniformly-spaced parameter samples from the training set to construct Ξ_{defect} and obtain the corresponding defect vector in Step 2. The SVD tolerances $\text{tol}_{\text{SVD},t}, \text{tol}_{\text{SVD},\mu}$ are both set to 10^{-4} so that $n_d = 47$. The user-defined solver $\text{solver}_{\text{imp}}$ is IMEX1. The resulting convergence of the greedy algorithm using the RBF-based approximation of the defect vector is shown in Fig. 8(a). As shown, the maximum estimated error converges exponentially to the desired tolerance. The dimension of the ROM obtained is $n = 7$. To demonstrate the performance of the ROM, we show in Fig. 8(b) the mean estimated errors for the parameters in the test set Ξ_{test}. The obtained errors are smaller than the desired tolerance 10^{-4}, showing the reliability of our error estimation approach. Furthermore, Fig. 9(a) shows the singular value decays of the defect vector trajectories at the 16 parameter samples in Ξ_{defect}. Fig. 9(b) shows the singular value decay obtained from the SVD of $\mathbf{R}$ (Section 4.2.1). The singular value decay in Fig. 9(b) indicates the fast Kolmogorov n-width decay of the defect manifold and hence good performance of Algorithm 2. In terms of runtime, this approach requires 17 seconds for generating the training data for learning the defect vector and obtaining the RBF interpolants.

POD-Greedy with Algorithm 2 and SVD+FNN closure approximation. Next, we repeat Algorithm 2 but now with the neural network-based approximation of the defect. The feed-forward neural network has 3 hidden layers with $16, 64, 64$ neurons, respectively. The activation function are the SiLU function for the first three layers and Tanh for the last layer. We have normalized the input data to be between $[0, 1]$ and the output data is between $[-1, 1]$. The neural network is implemented in PyTorch. It is trained using the Adam optimizer for 2000 epochs, with learning rate 0.005. Initially, we do not implement Step 6 and do not update $\widetilde{\mathbf{d}}_{\text{NN}}(t, \mu^*)$ with $\mathbf{d}(t, \mu^*)$ at μ^* (selected from the previous iteration) when computing the error estimator $\overline{\Delta}_b(\mu)$ in Step 11 at the current iteration. Using
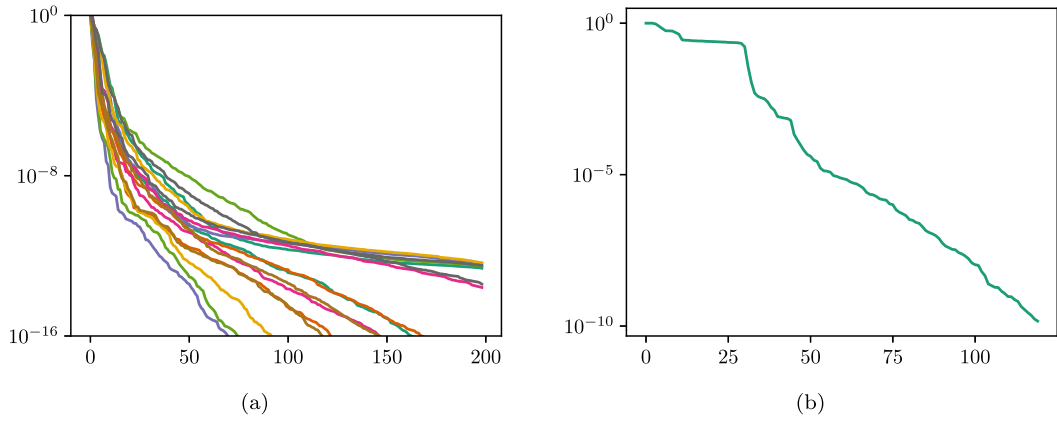
Fig. 9. Burgers' equation: (a) normalized singular values of the defect matrix $\mathbf{D}(\boldsymbol{\mu})$ for $\boldsymbol{\mu} \in \Xi_{\text{defect}}$; (b) normalized singular values of $\mathbf{R}$.
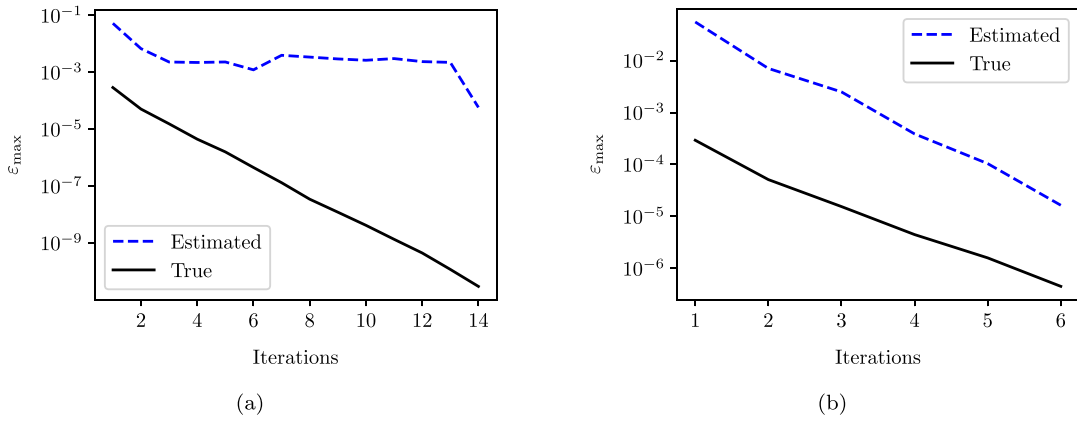

Fig. 10. Burgers' equation, Algorithm 2: (a) error (estimator) decay when using the SVD+FNN method without closure updates; (b) error (estimator) decay when using the SVD+FNN method with closure updates.

the same Ξ_{defect} with $|\Xi_{\text{defect}}| = 16$ (as done for the SVD+RBF approach), we did not obtain convergence of the greedy algorithm. Therefore, we use a Ξ_{defect} with 24 samples. The SVD tolerances for this case are both 0.1 resulting in $n_d = 6$. Using enriched training data results in the successful convergence of Algorithm 2 as seen in Fig. 10(a). However, it requires up to 14 iterations for this convergence. Evidently, the NN-based approach does not yield a satisfactory performance even with more training data. Then, we implement Step 6 and update the defect vector approximation at Step 11, which results in a significant improvement in performance. This is shown in Fig. 10(b).

To explain the poorer performance of the SVD+FNN approach, we plot in Fig. 11 the approximation of the defect vector at $\mu = 0.123$ and 4 different time instances, viz., $t \in \{0.01, 0.2, 0.5, 1.5\}$ s. We notice that the approximation from the FNN, while qualitatively capturing the true defect, fails to produce a very close match to the true value. This is especially the case for latter time instances, as the magnitude of the defect vector gets smaller. However, the RBF-based approach results in a significantly better approximation. This might be because of the fact that each entry of the reduced defect vector is separately learned by an individually-trained RBF interpolation, while the whole reduced defect vector is learned by a single and uniformly-trained FNN. However, if we use n_d FNNs to learn the n_d entries of the defect vector separately, the training will become much more expensive when n_d is not very small.

5.3. FitzHugh–Nagumo equations

Model description. The FitzHugh–Nagumo system models the response of an excitable neuron or cell under an external stimulus. It finds applications in a variety of fields such as cardiac electrophysiology and brain modelling. The nonlinear coupled system of two partial differential equations defined in the domain $\Omega := [0, L]$ is given below:

$$\epsilon \frac{\partial v_1(z, t)}{\partial t} = \epsilon^2 \frac{\partial^2 v_1(z, t)}{\partial z^2} + f(v_1(z, t)) - v_2(z, t) + c, \tag{45a}$$
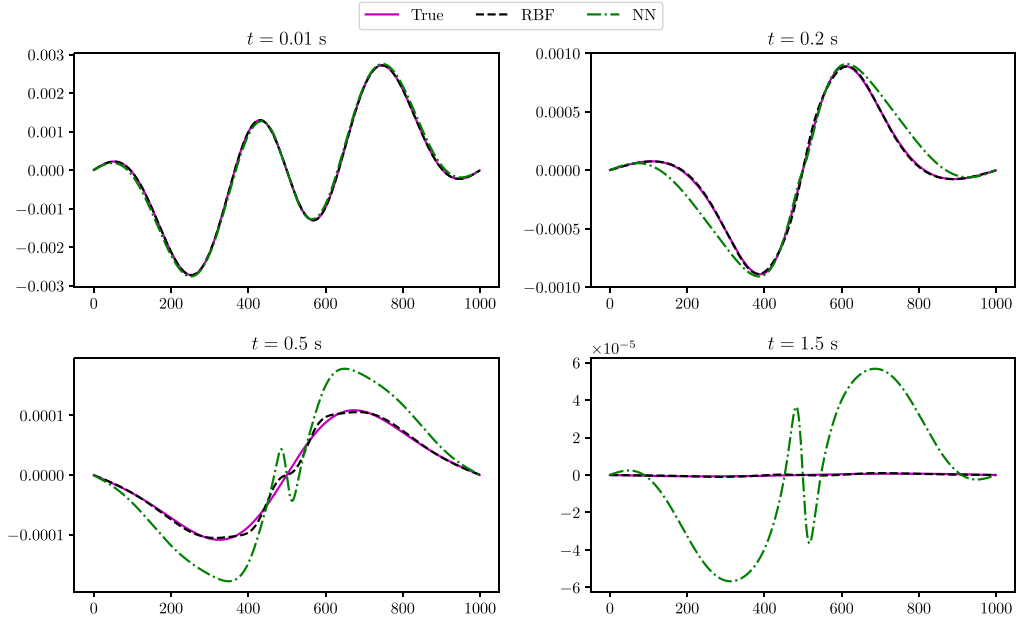
Fig. 11. Burgers' equation: approximation of the true defect by the RBF-based and NN-based approaches at $t \in \{0.01, 0.2.0.5, 1.5\}$ s for $\mu = 0.123$.

$$\frac{\partial v_2(z,t)}{\partial t} = b\, v_1(z,t) - \gamma v_2(z,t) + c, \tag{45b}$$

with boundary conditions

$$\frac{\partial}{\partial z} v_1(0,t) = -I_{\text{ext}}(t), \quad \frac{\partial}{\partial z} v_1(L,t) = 0, \tag{46}$$

and initial conditions

$$v_1(z,0) = 0.001, \quad v_2(z,0) = 0.001. \tag{47}$$

In the above equations, $v_1(z,t)$ and $v_2(z,t)$ represent the electric potential and the recovery rate of the potential, respectively. The spatial variable is denoted by $z \in \Omega$ and the time $t \in [0,5]$. The nonlinear term is represented by $f(v_1(z,t)) := v_1(v_1 - 0.1)(1 - v_1)$. The external stimulus is $I_{\text{ext}}(t) = 50000 t^3 e^{-15t}$. The system has the four free parameters ϵ, c, b, γ. We fix $b = 0.5$ and $\gamma = 2$ while the two free parameters are $\mu = [\epsilon, c] \in \mathcal{P} := [0.01, 0.04] \times [0.025, 0.075]$. A finite difference scheme is employed to spatially discretize Eqs. (45a) and (45b) with 512 nodes used for each variable leading to a FOM of dimension $N = 1024$. We sample 100 parameters uniformly from the domain $\mathcal{P}$ and randomly divide them into the training set Ξ and the test set Ξ_{test} in the ratio 70 : 30. To solve the ODE, we use ode15s from MATLAB®. The time discretization is done on a uniform grid with $\delta t = 0.01$. The QoIs are the values of the two state variables at the node next to the leftmost boundary.

We apply Algorithms 1 and 2 to the FitzHugh–Nagumo system. This is a particularly challenging example for both algorithms as the system exhibits a slow singular value decay. Of particular interest is the approximation of the limit cycle behaviour of the system for certain combinations of the two free parameters, (ϵ, c). The RBM tolerance is set as tol $= 10^{-3}$.

POD-Greedy with Algorithm 1. Applying Algorithm 1 to this example and using a second-order IMEX scheme (IMEX2) to compute the residual does not result in the convergence of the greedy algorithm, as see in Fig. 13. This stems from the fact that the actual numerical scheme solver used in Algorithm 1 is the ode15s solver from MATLAB®, therefore, the residual we compute using IMEX2 scheme is incorrect.

POD-Greedy with Algorithm 2 and SVD+RBF closure approximation. We apply Algorithm 2 to this example using $\widetilde{\mathbf{d}}_{\text{RBF}}(t, \mu)$ in Step 3 and with an RBM tolerance tol $= 10^{-3}$. The user-imposed time integration scheme solver$_{\text{imp}}$ is IMEX2. Due to the challenging nature of the problem, $d_s = 21$ uniformly-spaced samples are chosen from the training set to obtain Ξ_{defect}. The tolerance tol$_{\text{SVD},t}$ = tol$_{\text{SVD},\mu}$ = 10^{-6}, resulting in $n_d = 311$. Fig. 12(a) plots the singular value decays of $\mathbf{D}(\mu)$ at all $\mu \in \Xi_{\text{defect}}$ while Fig. 12(b) shows the decay of the singular values of $\mathbf{R}$ (see Section 4.2.1). Similar to the case of the Burgers' equation, an exponential decay of the singular values is observed in Fig. 12(a). However, the second SVD shown in Fig. 12(b) has a relatively slower decay of the singular values. This shows that the solution manifold of the FitzHugh–Nagumo system with respect to the parameter variations is more difficult to be approximated by a low-dimensional linear subspace. We incur 25 seconds to compute the training data for the defect trajectories and to obtain the SVD+RBF approximation of the defect vectors. The greedy algorithm takes 11 iterations to reach the desired tolerance; Fig. 14(a) shows the error convergence of Algorithm 2. The dimension of the ROM obtained is $n = 33$.
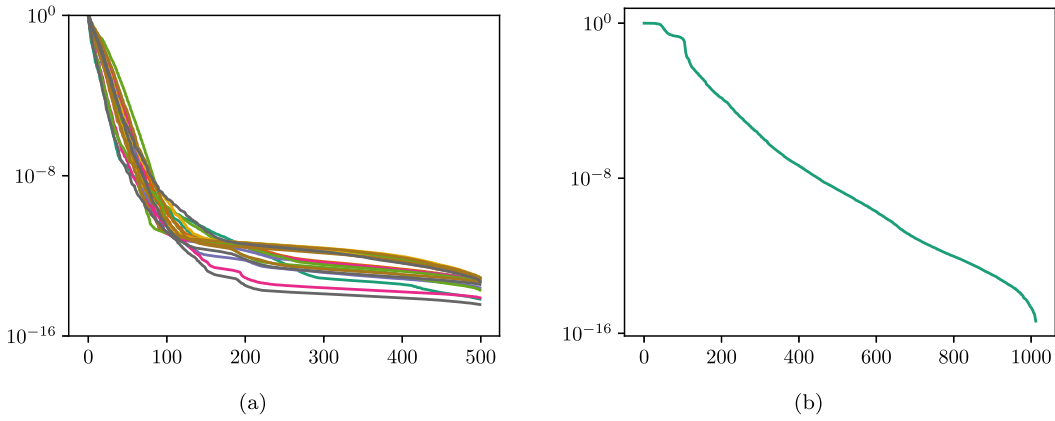
(a)

(b)

Fig. 12. FitzHugh–Nagumo equations: (a) Stage 1 - normalized singular values of the defect matrix $\mathbf{D}(\boldsymbol{\mu})$ for $\boldsymbol{\mu} \in \Xi_{\mathrm{defect}}$; (b) Stage 2 - normalized singular values of $\mathbf{R}$.
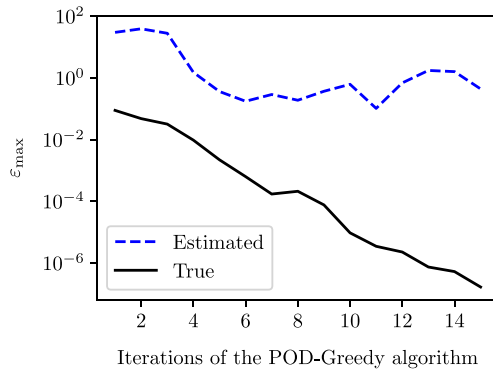

Fig. 13. FitzHugh–Nagumo equations, Algorithm 1: error (estimator) decay when using a second-order IMEX method without any closure term.
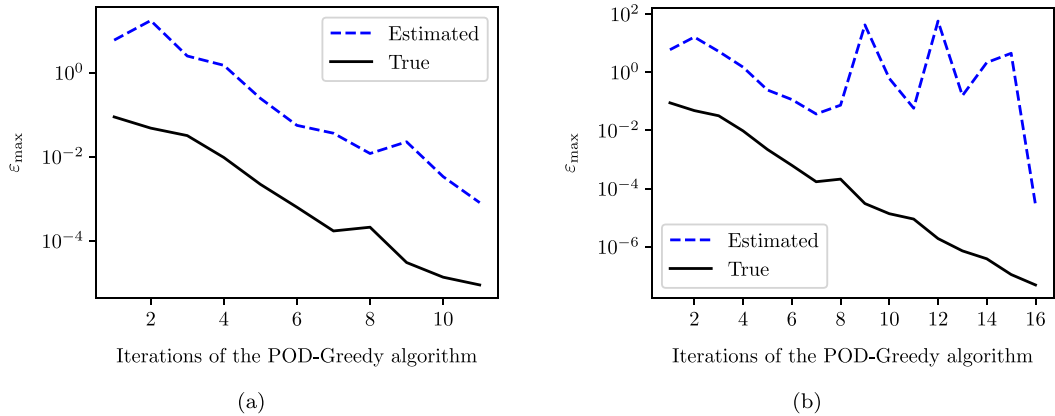

(a)

(b)

Fig. 14. FitzHugh–Nagumo equations, Algorithm 2: (a) error (estimator) decay when using the SVD+RBF method with Step 6 included; (b) error (estimator) decay when using the SVD+RBF method without Step 6.

Note that we have implemented Step 6 in Algorithm 2 to update the RBF approximation $\widetilde{\mathbf{d}}_{\mathrm{RBF}}(t, \boldsymbol{\mu}^*)$ with $\mathbf{d}(t, \boldsymbol{\mu}^*)$ when we compute the error estimator $\overline{\Delta}_b(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \Xi$ in Step 11. Without doing so, the greedy algorithm converges nevertheless, but takes 16 iterations (Fig. 14(b)) and the ROM has a larger size $n = 48$.

POD-Greedy with Algorithm 2 and SVD+FNN closure approximation. Next, we use $\widetilde{\mathbf{d}}_{\mathrm{NN}}(t, \boldsymbol{\mu})$ in Step 3 of Algorithm 2. The FNN is a 3-layer network having, respectively, $64, 64, 32$ neurons in its hidden layers. The SiLU function is used for activation in all but the last layer. In the last layer, Tanh is the activation function. The training is carried out for 2000 epochs using the Adam optimizer.
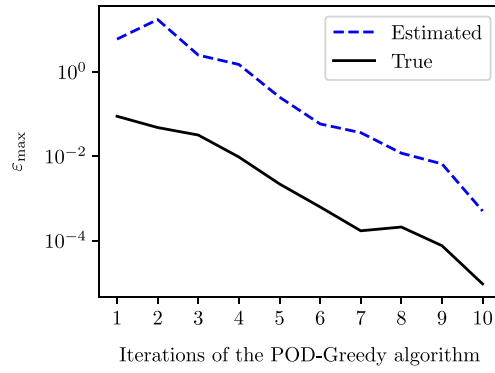
Fig. 15. FitzHugh–Nagumo equations, Algorithm 2: error (estimator) decay when using the SVD+FNN method with Step 6 included.
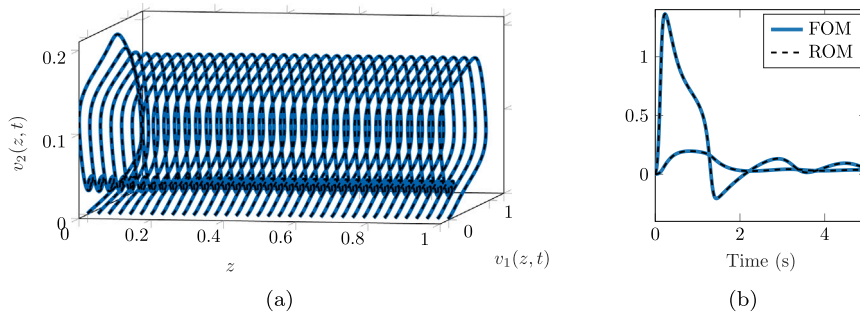

Fig. 16. FitzHugh–Nagumo equations, Algorithm 2 with SVD+RBF: performance at the test parameter $\boldsymbol{\mu} = (0.0267, 0.0367)$ (a) Limit cycle behaviour; (b) output quantities.
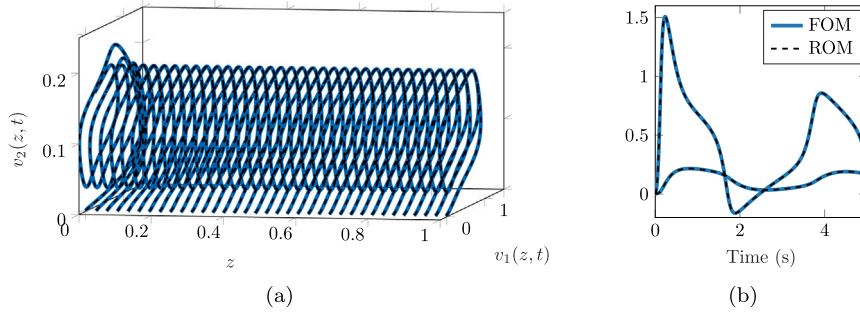

Fig. 17. FitzHugh–Nagumo equations, Algorithm 2 with SVD+FNN: performance at the test parameter $\boldsymbol{\mu} = (0.04, 0.0472)$ (a) Limit cycle behaviour; (b) output quantities.

The learning rate is 0.002. No special tuning was done to calibrate the hyperparameters of the FNN. A detailed investigation on this is left for future work. We set $\mathtt{tol}_{\mathrm{SVD},t} = \mathtt{tol}_{\mathrm{SVD},\boldsymbol{\mu}} = 10^{-3}$, such that $n_d = 57$. The total time for computing the training data at all the samples in Ξ_{defect} and for training the FNN is 106 seconds, where training the FNN dominates the total runtime. Fig. 15 plots the convergence of the greedy algorithm. It takes 10 iterations to converge. The resulting ROM has dimension $n = 30$.

The finally derived ROM is then simulated at two parameter samples $\boldsymbol{\mu} = (0.0267, 0.0367)$ and $\boldsymbol{\mu} = (0.04, 0.0472)$ taken from the test set. Fig. 16 shows the results of the ROM for the parameter $\boldsymbol{\mu} = (0.0267, 0.0367)$ obtained from Algorithm 2 using the SVD+RBF approximation of the closure term. We see that the ROM is able to successfully capture both the state and the output dynamics of the FOM at the test parameter. At this parameter, the limit cycle behaviour is not very strong. At a different test parameter ($\boldsymbol{\mu} = (0.04, 0.0472)$) shown in Fig. 17, the ROM is able to successfully recover the stronger limit cycle behaviour as well. For this case, we show results using the SVD+FNN approach. But, we note that similar accuracy is also achieved with the SVD+RBF approach. The corresponding results are not shown to avoid repetition.
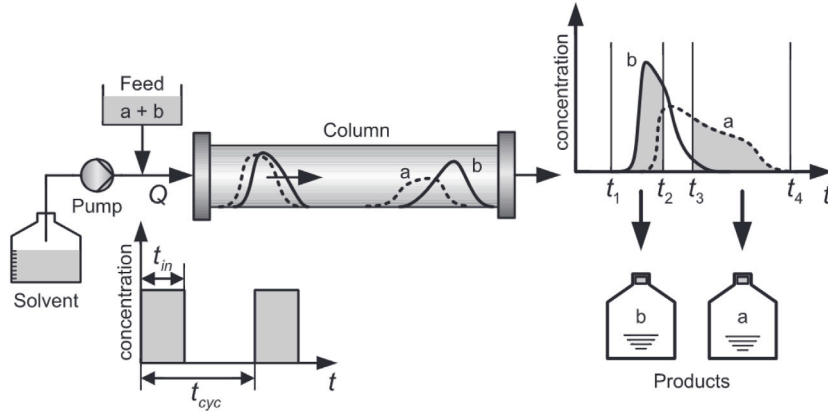
Fig. 18. Schematic of the batch chromatography process. The mobile phase Solvent and the Feed mixture consisting of the two chemicals (a, b) to be separated are periodically injected into the Column at a volumetric feed flow rate of Q and with injection intervals t_{in}. The Column contains an adsorbent bed through which the two chemicals a, b travel at different velocities, owing to their differing adsorption affinities. The different velocities ensure that the separated Products a and b can be collected at the outlet at different time instances.

5.4. Batch chromatography

The last example we consider is the model of the batch chromatography purification process. Being a coupled, nonlinear system of four PDEs, this example poses a considerable challenge for our proposed approach.

Model description. Batch chromatography is an important purification process for separation of chemicals in food and pharmaceutical industries. We consider the governing equations for the batch chromatographic process for binary separation, i.e., the separation of two chemical components from a mixture. A schematic of the complete process is shown in Fig. 18.

The governing PDEs for the batch chromatography system are:

$$\frac{\partial v_{1,z}}{\partial t} + \frac{1-\epsilon}{\epsilon}\frac{\partial v_{2,z}}{\partial t} = -\frac{\partial v_{1,z}}{\partial z} + \frac{1}{\text{Pe}}\frac{\partial^2 v_{1,z}}{\partial z^2}, \tag{48}$$

$$\frac{\partial v_{2,z}}{\partial t} = \frac{L}{Q/\epsilon A_c}\kappa_z\left(v_{2,z}^{\text{Eq}} - v_{2,z}\right), \tag{49}$$

where the state variables $v_{1,z}, v_{2,z}$ refer to the concentrations of the chemical component z in the liquid and solid phase, respectively. Since we are interested in binary separation, $z \in \{a, b\}$. The boundary conditions are:

$$\frac{\partial v_{1,z}(0,t)}{\partial z} = \text{Pe}\left(v_{1,z}(0,t) - u(t)\right), \quad \frac{\partial v_{1,z}(1,t)}{\partial z} = 0 \tag{50}$$

and the initial conditions are:

$$v_{1,z}(z,0) = 0, \quad v_{2,z}(z,0) = 0. \tag{51}$$

The quantity $v_{2,z}^{\text{Eq}}$ in (48) is the source of nonlinearity and it denotes the adsorption equilibrium:

$$v_{2,z}^{\text{Eq}} = f_z(v_{1,a}, v_{2,b}) := \frac{H_{z1}v_{1,z}}{1 + K_{a1}v_{1,a}^{\text{f}}v_{1,a} + K_{b1}v_{2,b}^{\text{f}}v_{2,b}} + \frac{H_{z2}v_{1,z}}{1 + K_{a2}v_{1,a}^{\text{f}}v_{1,a} + K_{b2}v_{2,b}^{\text{f}}v_{2,b}}.$$

The discretization of the above PDE is performed using a second-order finite volume method. Each of the four PDEs is spatially discretized into 800 volume elements, resulting in a FOM of dimension $N = 3200$. For full details regarding the batch chromatography PDE discretization and the terms involved, we refer to [60]. The output variables are the concentrations of the liquid phases $(v_{1,a}, v_{1,b})$ at the rightmost node. The batch chromatography model has two free parameters, Q and t_{in}, which denote, respectively, the volumetric feed flow of the solvent injected into the column and the injection frequency of the solvent. We fix $t_{\text{in}} = 0.5$ while $Q \in [0.0667, 0.1667]$. We collect 60 uniformly-spaced samples of Q and divide them into the training set Ξ and the test set Ξ_{test} in the ratio $80 : 20$. The solver used for both algorithms is ode15s from MATLAB® while the user-imposed time integration method solver$_{\text{imp}}$ in Algorithm 2 is IMEX2. The time discretization divides the time range $t \in [0, 10]$ into a uniform grid with $\delta t = 0.005$.

POD-Greedy with Algorithm 1. First, we show the results of applying Algorithm 1 to the batch chromatography equations. The RBM tolerance is set to be tol $= 10^{-3}$. Since the exact form of the residual expression of batch chromatography equations is unknown, (i.e., $\mathcal{R}[\cdot]$ corresponding to ode15s is not available) we impose IMEX2 to get an approximate residual operator $\widetilde{\mathcal{R}}[\cdot]$. However, $\widetilde{\mathcal{R}}[\cdot]$ corresponding to IMEX2 is different from $\mathcal{R}[\cdot]$ so that the estimated error is inaccurate, resulting in stagnation of the greedy algorithm, see Fig. 19. To address this situation, we next apply the proposed approach which involves adding a closure term.
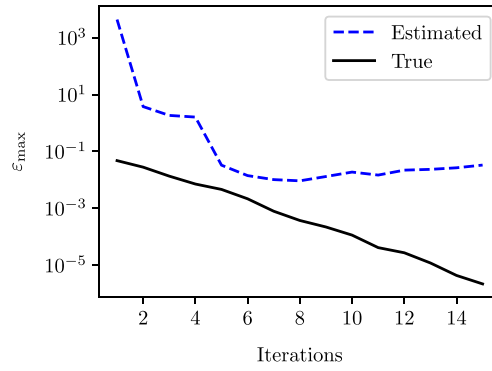
Fig. 19. Batch chromatography, Algorithm 1: error (estimator) decay when using a second-order IMEX method without any closure term.
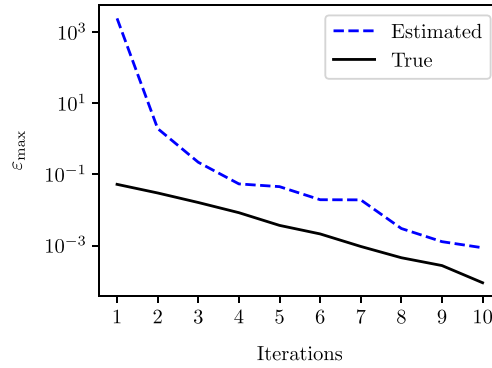

Fig. 20. Batch chromatography, Algorithm 2: convergence when using the SVD+RBF method with Step 6 included to approximate the closure term.

POD-Greedy with Algorithm 2 and SVD+RBF closure approximation. As done for the last two examples, we first apply Algorithm 2 to the current example and employ the RBF-based approach to approximate the closure term. The greedy algorithm successfully converges in 10 iterations to the desired tolerance of $\mathtt{tol} = 10^{-3}$. The dimension of the ROM is $n = 134$. The convergence is shown in Fig. 20. To learn the closure term, $d_s = 10$ uniformly-spaced samples are chosen from the training set $\varXi$. The SVD tolerances are $\mathtt{tol}_{\mathrm{SVD},t} = \mathtt{tol}_{\mathrm{SVD},\mu} = 10^{-5}$, leading to $n_d = 291$. The singular value decay from the SVD of each defect snapshot matrix $\mathbf{D}(\mu)$ for all $\mu \in \varXi_{\mathrm{defect}}$ is presented in Fig. 21(a). It can be noticed that the singular values decay even slower than those of the FitzHugh–Nagumo model (see Fig. 12(a)). In Fig. 21(b), the singular values of $\mathbf{R}$ also exhibit a much slower decay. This indicates that for the batch chromatography example, both the dynamics at a given parameter and the solution manifold with respect to the parameter variations are much more difficult to be captured by a low-dimensional linear space. As a result, this problem is likely to have a slow Kolmogorov n-width decay. The batch chromatography equations are in fact a system of first-order hyperbolic PDEs [61]. It is known that for problems exhibiting hyperbolic characteristics or convection-dominance, the Kolmogorov n-width decay is slow [62]. The runtime for obtaining the training data for the SVD+RBF approach and for obtaining the RBF interpolant is 186 seconds. To demonstrate the quality of the ROM resulting from Algorithm 2, we compare the output obtained from the FOM and ROM in Fig. 22. The results are shown for the parameter sample $Q = 0.0803$ taken from $\varXi_{\mathrm{test}}$. We can see that the ROM is able to successfully recover the dynamics of both output quantities. For this sample, we obtain the mean error $\overline{\Delta}(Q) = 2.082 \cdot 10^{-5}$, which is below the desired tolerance. Additionally, to show the approximation of the state vector, we plot the space–time values and corresponding approximation errors of the liquid phase concentration $v_{1,a}, v_{1,b}$. The results are shown in Figs. 23 and 24. It can be inferred that the ROM delivers a sharp approximation of both these state quantities for the entire duration of the simulation at an unseen parameter during training. Note that the state vector has error larger than the tolerance. The reason for this is that our error estimator aims to estimate the output error rather than the whole state error.

The quality of the defect vector approximation using the SVD+FNN approach was not satisfactory for this example. This owes to the particularly non-smooth nature of the defect snapshots for different time instances and parameters. Fig. 25 shows the true reduced defect vector $\hat{\mathbf{d}}(t, \mu)$ and its approximation using RBF interpolation $\hat{\mathbf{d}}_{\mathrm{RBF}}(t, \mu)$ corresponding to $\mu = 0.0769$ at the time instances $t \in \{0.05, 0.25, 2.5, 5.0\}$ s. While the RBF interpolants recover a good approximation, the FNN was not successful in capturing the entire complexity of the defect snapshots. The FNN-based approximation was very inaccurate and we do not show those results. The neural network struggles to capture the fast changing nature of the reduced defect vector in the reduced coordinate space and also its wide range of magnitudes ($\pm(10^{-3} - 10^{-8})$). The fact that we used separate RBF interpolants for each coordinate and time instance led to a much better approximation than the FNN.
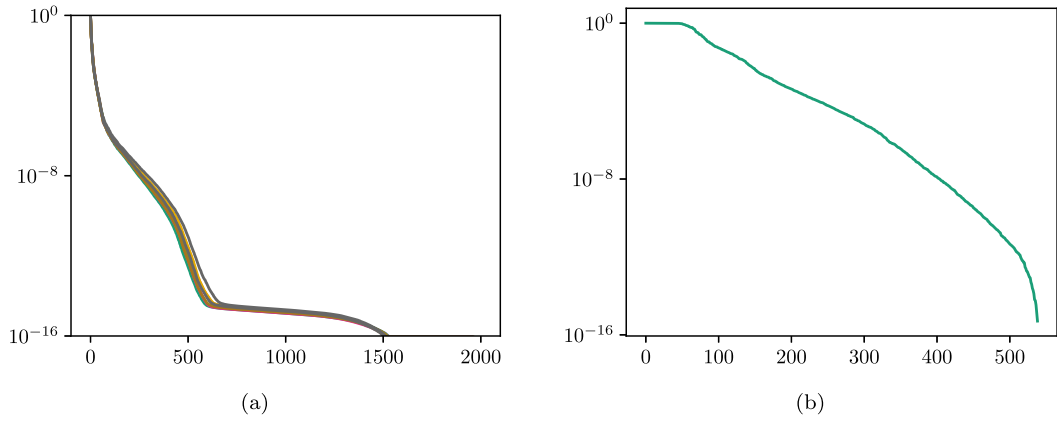
Fig. 21. Batch chromatography: (a) Stage 1 - normalized singular values of the defect matrix $\mathbf{D}(\boldsymbol{\mu})$ for $\boldsymbol{\mu} \in \Xi_{\text{defect}}$; (b) Stage 2 - normalized singular values of $\mathbf{R}$.
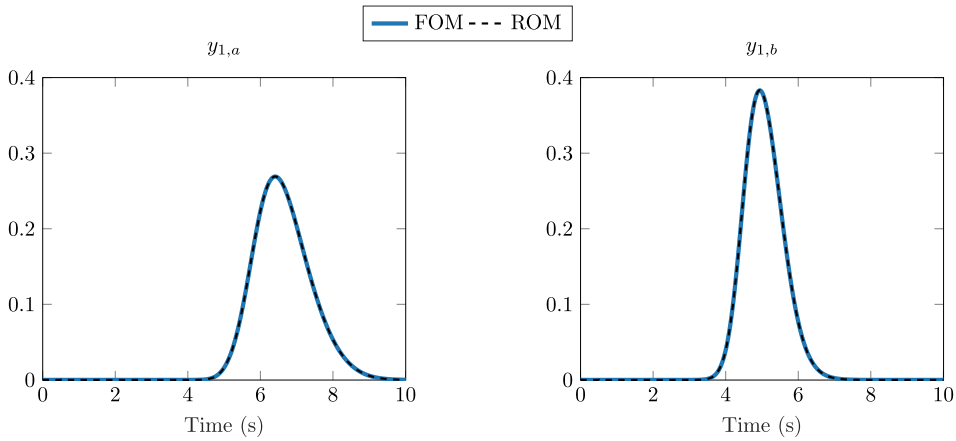

Fig. 22. Batch chromatography, Algorithm 2: performance of the ROM at test parameter sample $Q = 0.0803$ for the two output quantities $y_{1,a}, y_{1,b}$.
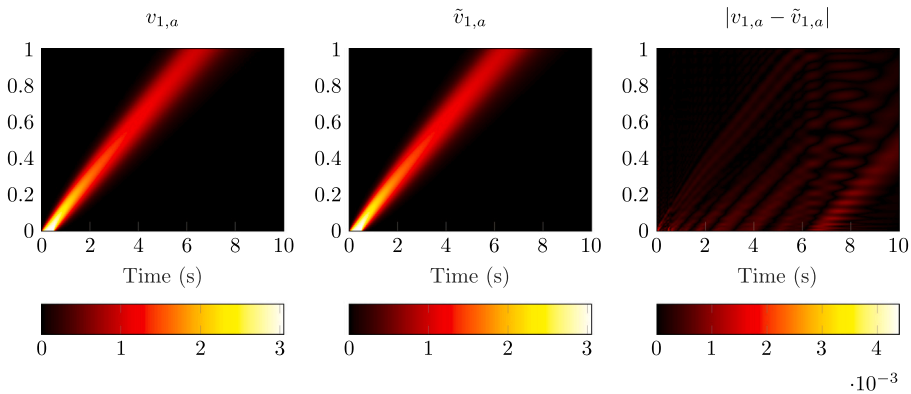

Fig. 23. Batch chromatography, Algorithm 2: performance of the ROM at test parameter sample $Q = 0.0803$ for the state $v_{1,a}$.

6. Conclusions

In this work, we have introduced a data-enhanced *a posteriori* output error estimator for model reduction of general parametric nonlinear dynamical systems. The proposed error estimator does not require any knowledge of the underlying time integration scheme used to integrate the given ODE. Applied to the reduced basis method, the new approach enables the direct use of ODE solver libraries, a feature that was not considered so far, to the best of our knowledge. While it demands a modest amount of
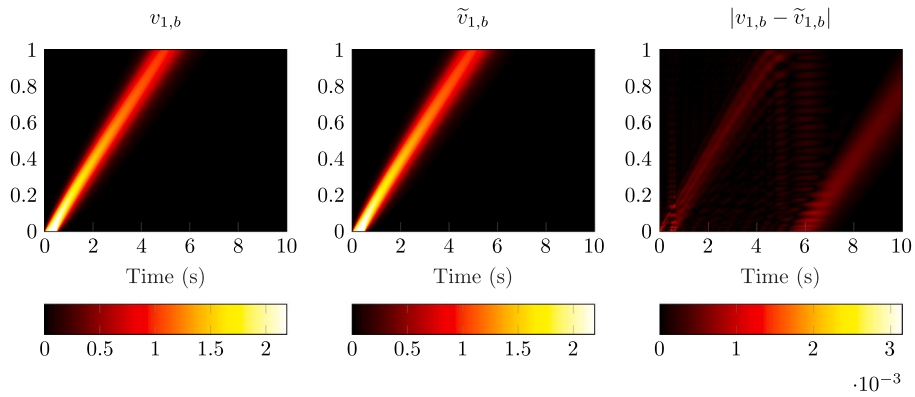
Fig. 24. Batch chromatography, Algorithm 2: performance of the ROM at test parameter sample $Q = 0.0803$ for the state $v_{1,b}$.
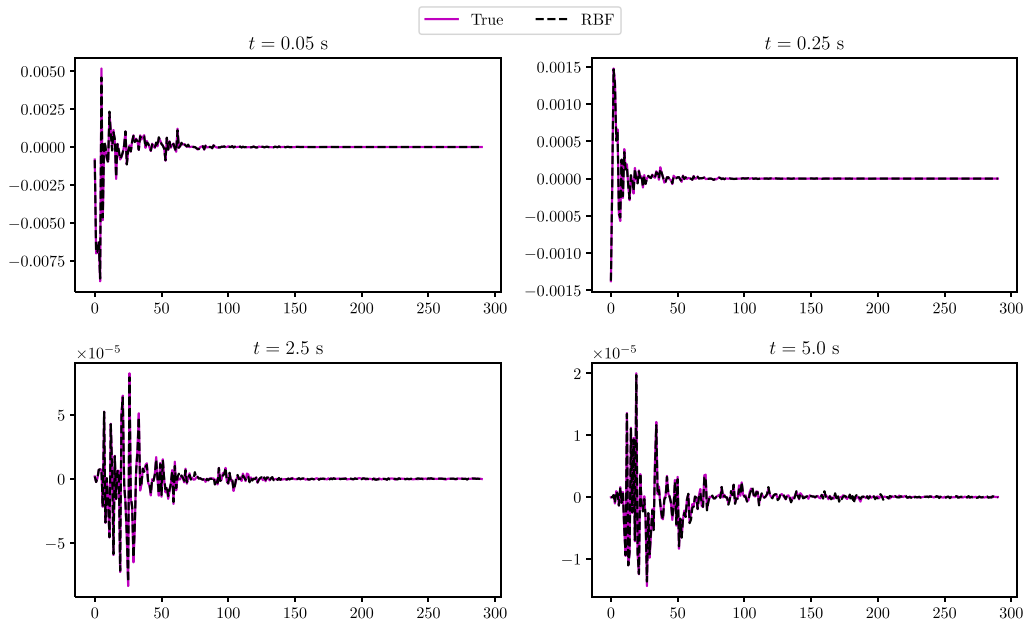

Fig. 25. Batch chromatography: approximation of the reduced defect vector by the RBF-based approach at $t \in \{0.05, 0.25, 2.5, 5.0\}$ s at $Q = 0.0769$.

extra training data, the proposed error estimator is efficient and can also be used in the online stage to certify the accuracy of the ROMs. Numerical experiments performed on three challenging examples demonstrate the benefits offered by the new approach. We observed that the RBF-based approach performed better, compared to a NN-based approach. An immediate extension of the proposed approach is to consider an adaptive sampling of the training set Ξ. New strategies to accurately approximate the closure term need to be considered for this. Another fruitful line of future work could involve integrating the proposed methodology in the Neural ODE framework [33] to make it fully non-intrusive.

CRediT authorship contribution statement

Sridhar Chellappa: Conceptualization, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Lihong Feng:** Formal analysis, Supervision, Writing – review & editing, Methodology. **Peter Benner:** Formal analysis, Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Acknowledgements

Appendix A

Proof of Theorem 2.1. Consider the FOM in (4) and its residual (6) based on the ROM (5). Subtracting (6) from (4) and rearranging yields

$$
\mathbf{E}_{\mathrm{im}}\mathbf{e}^{k} = \mathbf{A}_{\mathrm{im}}\mathbf{e}^{k-1} + \delta t\left(\mathbf{f}(\mathbf{x}_{\mathrm{im}}^{k-1}) - \mathbf{f}(\widetilde{\mathbf{x}}_{\mathrm{im}}^{k-1})\right) + \mathbf{r}^{k},
$$

$$
\mathbf{e}^{k} = \mathbf{E}_{\mathrm{im}}^{-1}\mathbf{A}_{\mathrm{im}}\mathbf{e}^{k-1} + \delta t\mathbf{E}_{\mathrm{im}}^{-1}\left(\mathbf{f}(\mathbf{x}_{\mathrm{im}}^{k-1}) - \mathbf{f}(\widetilde{\mathbf{x}}_{\mathrm{im}}^{k-1})\right) + \mathbf{E}_{\mathrm{im}}^{-1}\mathbf{r}^{k} \tag{A.1}
$$

where $\mathbf{e}^{k} = \|\mathbf{x}_{\mathrm{im}}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im}}^{k}\|$ is the error in the state vector at the kth time step.

Taking the norm on both sides and enforcing the Lipschitz condition results in

$$
\begin{aligned}
\left\|\mathbf{e}^{k}\right\|_{2} &\le \left\|\mathbf{E}_{\mathrm{im}}^{-1}\mathbf{A}_{\mathrm{im}}\right\|_{2}\left\|\mathbf{e}^{k-1}\right\|_{2} + \delta t L_{\mathbf{f}}\left\|\mathbf{E}_{\mathrm{im}}^{-1}\right\|_{2}\left\|\mathbf{e}^{k-1}\right\|_{2} + \left\|\mathbf{E}_{\mathrm{im}}^{-1}\right\|_{2}\left\|\mathbf{r}^{k}\right\|_{2} \\
&\le \left(\left\|\mathbf{E}_{\mathrm{im}}^{-1}\mathbf{A}_{\mathrm{im}}\right\|_{2} + \delta t L_{\mathbf{f}}\left\|\mathbf{E}_{\mathrm{im}}^{-1}\right\|_{2}\right)\left\|\mathbf{e}^{k-1}\right\|_{2} + \left\|\mathbf{E}_{\mathrm{im}}^{-1}\right\|_{2}\left\|\mathbf{r}^{k}\right\|_{2}.
\end{aligned} \tag{A.2}
$$

For notational convenience, we define $\zeta := \left\|\mathbf{E}_{\mathrm{im}}^{-1}\right\|_{2}$ and $\xi := \left(\left\|\mathbf{E}_{\mathrm{im}}^{-1}\mathbf{A}_{\mathrm{im}}\right\|_{2} + \delta t L_{\mathbf{f}}\left\|\mathbf{E}_{\mathrm{im}}^{-1}\right\|_{2}\right)$ and rewrite (A.2) as

$$
\left\|\mathbf{e}^{k}\right\|_{2} \le \xi\left\|\mathbf{e}^{k-1}\right\|_{2} + \zeta\left\|\mathbf{r}^{k}\right\|_{2}. \tag{A.3}
$$

At $k = 0$, the initial error is

$$
\left\|\mathbf{e}^{0}\right\|_{2} = \left\|\mathbf{x}_{\mathrm{im}}^{0} - \mathbf{V}\mathbf{V}^{\top}\mathbf{x}_{\mathrm{im}}^{0}\right\|_{2}. \tag{A.4}
$$

Using (A.4), the recursion in (A.3) can be resolved for every k to obtain the expression for the error bound in (7). □

Appendix B

Proof of Theorem 3.1. The output error for the modified output in (25) is

$$
\mathbf{y}^{k} - \overline{\mathbf{y}}_{\mathrm{im,c}}^{k} = \mathbf{C}\left(\mathbf{x}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}\right) + \widetilde{\mathbf{x}}_{\mathrm{du}}^{\top}\mathbf{r}_{\mathrm{im,c}}^{k}. \tag{B.1}
$$

Multiplying by $\left(\mathbf{x}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}\right)^{\top}$ on both sides of (20) yields

$$
\left(\mathbf{x}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}\right)^{\top}\mathbf{E}_{\mathrm{im}}^{\top}\mathbf{x}_{\mathrm{du}} = -\left(\mathbf{x}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}\right)^{\top}\mathbf{C}^{\top},
$$

where we have made use of the fact that $\mathbf{E}_{\mathrm{du}} = \mathbf{E}_{\mathrm{im}}^{\top}$. Taking the transpose on both sides of the above equality leads to

$$
\mathbf{x}_{\mathrm{du}}^{\top}\mathbf{E}_{\mathrm{im}}\left(\mathbf{x}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}\right) = -\mathbf{C}\left(\mathbf{x}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}\right). \tag{B.2}
$$

We recall the auxiliary residual introduced in (24) which can be written as

$$
\begin{aligned}
\check{\mathbf{r}}_{\mathrm{im,c}}^{k} &= \mathbf{A}_{\mathrm{im}}\mathbf{x}_{\mathrm{im,c}}^{k-1} + \delta t\left(\mathbf{f}(\mathbf{x}_{\mathrm{im,c}}^{k-1}) + \mathbf{B}\mathbf{u}^{k}\right) + \mathbf{d}^{k} - \mathbf{E}_{\mathrm{im}}\widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}, \\
&= \mathbf{E}_{\mathrm{im}}\left(\mathbf{x}_{\mathrm{im,c}}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}\right), \\
&= \mathbf{E}_{\mathrm{im}}\left(\mathbf{x}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}\right).
\end{aligned} \tag{B.3}
$$

The third equality above follows from our assumption that the corrected solution $(\mathbf{x}_{\mathrm{im,c}}^{k})$ obtained from (16) actually recovers the FOM solution $\mathbf{x}^{k}$ given an accurate closure term $\mathbf{d}^{k}$.

We further use the expression in (B.3) to write (B.2) as

$$
\mathbf{x}_{\mathrm{du}}^{\top}\check{\mathbf{r}}_{\mathrm{im,c}}^{k} = -\mathbf{C}\left(\mathbf{x}^{k} - \widetilde{\mathbf{x}}_{\mathrm{im,c}}^{k}\right). \tag{B.4}
$$

Now, we substitute (B.4) into (B.1) followed by addition and subtraction of the term $\widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \check{\mathbf{r}}_{\mathrm{im,c}}^{k}$ to get

$$
\begin{aligned}
\mathbf{y}^{k} - \overline{\mathbf{y}}_{\mathrm{im,c}}^{k} &= -\mathbf{x}_{\mathrm{du}}^{\mathsf{T}} \check{\mathbf{r}}_{\mathrm{im,c}}^{k} + \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \mathbf{r}_{\mathrm{im,c}}^{k}, \\
&= -\mathbf{x}_{\mathrm{du}}^{\mathsf{T}} \check{\mathbf{r}}_{\mathrm{im,c}}^{k} + \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \mathbf{r}_{\mathrm{im,c}}^{k} + \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \check{\mathbf{r}}_{\mathrm{im,c}}^{k} - \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \check{\mathbf{r}}_{\mathrm{im,c}}^{k}, \\
&= -\left(\mathbf{x}_{\mathrm{du}} - \widetilde{\mathbf{x}}_{\mathrm{du}}\right)^{\mathsf{T}} \check{\mathbf{r}}_{\mathrm{im,c}}^{k} + \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \left(\mathbf{r}_{\mathrm{im,c}}^{k} - \check{\mathbf{r}}_{\mathrm{im,c}}^{k}\right).
\end{aligned}
\tag{B.5}
$$

Subsequent to this, we use the expression for the dual system (20) and its residual (23) to obtain

$$
\begin{aligned}
\mathbf{r}_{\mathrm{du}} &= \mathbf{E}_{\mathrm{du}}\mathbf{x}_{\mathrm{du}} - \mathbf{E}_{\mathrm{du}}\widetilde{\mathbf{x}}_{\mathrm{du}} = \mathbf{E}_{\mathrm{du}}\left(\mathbf{x}_{\mathrm{du}} - \widetilde{\mathbf{x}}_{\mathrm{du}}\right), \\
&\implies \left(\mathbf{x}_{\mathrm{du}} - \widetilde{\mathbf{x}}_{\mathrm{du}}\right) = \mathbf{E}_{\mathrm{du}}^{-1}\mathbf{r}_{\mathrm{du}} = \mathbf{E}_{\mathrm{im}}^{-\mathsf{T}}\mathbf{r}_{\mathrm{du}}.
\end{aligned}
\tag{B.6}
$$

Substituting (B.6) into (B.5) yields

$$
\mathbf{y}^{k} - \overline{\mathbf{y}}_{\mathrm{im,c}}^{k} = -\mathbf{r}_{\mathrm{du}}^{\mathsf{T}} \mathbf{E}_{\mathrm{im}}^{-1} \check{\mathbf{r}}_{\mathrm{im,c}}^{k} + \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \left(\mathbf{r}_{\mathrm{im,c}}^{k} - \check{\mathbf{r}}_{\mathrm{im,c}}^{k}\right).
\tag{B.7}
$$

Taking the norm on either sides and using the triangle and Cauchy–Schwarz inequalities, we obtain the error bound as

$$
\left\| \mathbf{y}^{k} - \overline{\mathbf{y}}_{\mathrm{im,c}}^{k} \right\| \leq \| -\mathbf{r}_{\mathrm{du}}^{\mathsf{T}} \mathbf{E}_{\mathrm{im}}^{-1} \check{\mathbf{r}}_{\mathrm{im,c}}^{k} \| + \| \widetilde{\mathbf{x}}_{\mathrm{du}}^{\mathsf{T}} \left(\mathbf{r}_{\mathrm{im,c}}^{k} - \check{\mathbf{r}}_{\mathrm{im,c}}^{k}\right) \|,
\tag{B.8}
$$

$$
\leq \| \mathbf{E}_{\mathrm{im}}^{-1} \| \| \mathbf{r}_{\mathrm{du}} \| \| \check{\mathbf{r}}_{\mathrm{im,c}}^{k} \| + \| \widetilde{\mathbf{x}}_{\mathrm{du}} \| \| \left(\mathbf{r}_{\mathrm{im,c}}^{k} - \check{\mathbf{r}}_{\mathrm{im,c}}^{k}\right) \|. \quad \square
\tag{B.9}
$$

References

[1] D. Hartmann, M. Herz, U. Wever, Model order reduction a key technology for digital twins, in: W. Keiper, A. Milde, S. Volkwein (Eds.), Reduced-Order Modeling (ROM) for Simulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing, Springer-Verlag, Cham, 2018, pp. 167–179, http://dx.doi.org/10.1007/978-3-319-75319-5_8.

[2] M.G. Kapteyn, D.J. Knezevic, D.B.P. Huynh, M. Tran, K.E. Willcox, Data-driven physics-based digital twins via a library of component-based reduced-order models, Internat. J. Numer. Methods Engrg. 123 (13) (2022) 2986–3003, http://dx.doi.org/10.1002/nme.6423.

[3] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, L.M. Silveira (Eds.), Model Order Reduction. Volume 1: System- and Data-Driven Methods and Algorithms, De Gruyter, 2021, http://dx.doi.org/10.1515/9783110499001.

[4] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, L.M. Silveira (Eds.), Model Order Reduction. Volume 2: Snapshot-Based Methods and Algorithms, De Gruyter, 2021, http://dx.doi.org/10.1515/9783110671490.

[5] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, L.M. Silveira (Eds.), Model Order Reduction. Volume 3: Applications, De Gruyter, 2021, http://dx.doi.org/10.1515/9783110499001.

[6] G. Rozza, D.B.P. Huynh, A.T. Patera, Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: application to transport and continuum mechanics, Arch. Comput. Methods Eng. 15 (3) (2008) 229–275, http://dx.doi.org/10.1007/s11831-008-9019-9.

[7] A. Quarteroni, A. Manzoni, F. Negri, Reduced Basis Methods for Partial Differential Equations, in: La Matematica per il 3+2, vol. 92, Springer International Publishing, 2016, http://dx.doi.org/10.1007/978-3-319-15431-2.

[8] J.S. Hesthaven, G. Rozza, B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations, in: SpringerBriefs in Mathematics, Springer International Publishing, 2016, http://dx.doi.org/10.1007/978-3-319-22470-1.

[9] L. Machiels, Y. Maday, A.T. Patera, Output bounds for reduced-order approximations of elliptic partial differential equations, Comput. Methods Appl. Mech. Engrg. 190 (26–27) (2001) 3413–3426, http://dx.doi.org/10.1016/S0045-7825(00)00275-9.

[10] D.V. Rovas, Reduced-Basis Output Bound Methods for Parametrized Partial Differential Equations (Ph.D. thesis), Massachussetts Institute of Technology (MIT), Cambridge, USA, 2003, URL https://dspace.mit.edu/handle/1721.1/16956.

[11] M.A. Grepl, A.T. Patera, A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations, ESAIM: Math. Model. Numer. Anal. 39 (1) (2005) 157–181, http://dx.doi.org/10.1051/m2an:2005006.

[12] K. Veroy, C. Prud'Homme, D.V. Rovas, A.T. Patera, A posteriori error bounds for reduced-basis approximation of parametrized noncoercive and nonlinear elliptic partial differential equations, in: 16th AIAA Computational Fluid Dynamics Conference, Orlando, United States, 2003, URL https://hal.archives-ouvertes.fr/hal-01219051.

[13] C. Canuto, T. Tonn, K. Urban, A posteriori error analysis of the reduced basis method for nonaffine parametrized nonlinear PDEs, SIAM J. Numer. Anal. 47 (3) (2009) 2001–2022, http://dx.doi.org/10.1137/080724812.

[14] M.A. Grepl, Certified reduced basis methods for nonaffine linear time-varying and nonlinear parabolic partial differential equations, Math. Models Methods Appl. Sci. 22 (3) (2012) 1150015, 40, http://dx.doi.org/10.1142/S0218202511500151.

[15] B. Haasdonk, M. Ohlberger, Efficient reduced models and a posteriori error estimation for parametrized dynamical systems by offline/online decomposition, Math. Comput. Model. Dyn. Syst. 17 (2) (2011) 145–161, http://dx.doi.org/10.1080/13873954.2010.514703.

[16] M. Drohmann, B. Haasdonk, M. Ohlberger, Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation, SIAM J. Sci. Comput. 34 (2) (2012) A937–A969, http://dx.doi.org/10.1137/10081157X.

[17] D. Wirtz, D.C. Sorensen, B. Haasdonk, A posteriori error estimation for DEIM reduced nonlinear dynamical systems, SIAM J. Sci. Comput. 36 (2) (2014) A311–A338, http://dx.doi.org/10.1137/120899042.

[18] Y. Zhang, L. Feng, S. Li, P. Benner, An efficient output error estimation for model order reduction of parametrized evolution equations, SIAM J. Sci. Comput. 37 (6) (2015) B910–B936, http://dx.doi.org/10.1137/140998603.

[19] S. Chellappa, L. Feng, P. Benner, Adaptive basis construction and improved error estimation for parametric nonlinear dynamical systems, Internat. J. Numer. Methods Engrg. 121 (23) (2020) 5320–5349, http://dx.doi.org/10.1002/nme.6462.

[20] D.J. Gardner, D.R. Reynolds, C.S. Woodward, C.J. Balos, Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers, ACM Trans. Math. Softw. (2022) http://dx.doi.org/10.1145/3539801.

[21] A.C. Hindmarsh, P.N. Brown, K.E. Grant, S.L. Lee, R. Serban, D.E. Shumaker, C.S. Woodward, SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers, ACM Trans. Math. Softw. 31 (3) (2005) 363–396, http://dx.doi.org/10.1145/1089014.1089020.

[22] A.C. Hindmarsh, ODEPACK, a systematized collection of ODE solvers, in: R.S. Stepleman (Ed.), Scientific Computing : Applications of Mathematics and Computing to the Physical Sciences, Elsevier, 1983.

[23] D.R. Reynolds, D.J. Gardner, C.S. Woodward, R. Chinomona, ARKODE: A flexible IVP solver infrastructure for one-step methods, ACM Trans. Math. Software 49 (2) (2023) http://dx.doi.org/10.1145/3594632.

[24] S. Abhyankar, J. Brown, E.M. Constantinescu, D. Ghosh, B.F. Smith, H. Zhang, PETSc/TS: A modern scalable ODE/DAE solver library, 2018, http://dx.doi.org/10.48550/arXiv.1806.01437, e-prints arXiv:1806.01437.

[25] S. Balay, S. Abhyankar, M.F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E.M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W.D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M.G. Knepley, F. Kong, S. Kruger, D.A. May, L.C. McInnes, R.T. Mills, L. Mitchell, T. Munson, J.E. Roman, K. Rupp, P. Sanan, J. Sarich, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, J. Zhang, PETSc web page, 2023, URL https://petsc.org/.

[26] F. Casenave, A. Ern, T. Lelièvre, A nonintrusive reduced basis method applied to aeroacoustic simulations, Adv. Comput. Math. 41 (5) (2015) 961–986, http://dx.doi.org/10.1007/s10444-014-9365-0.

[27] R. Chakir, J. Hammond, A non-intrusive reduced basis method for elastoplasticity problems in geotechnics, J. Comput. Appl. Math. 337 (2018) 1–17, http://dx.doi.org/10.1016/j.cam.2017.12.044.

[28] R. Chakir, Y. Maday, A two-grid finite-element/reduced basis scheme for the approximation of the solution of parameter dependent PDE, in: 9e Colloque National en Calcul des Structures, CSMA, Giens, France, 2009, URL https://hal.archives-ouvertes.fr/hal-01420726.

[29] E. Grosjean, Y. Maday, Error estimate of the Non-Intrusive Reduced Basis (NIRB) two-grid method with parabolic equations, 2022, http://dx.doi.org/10.48550/arXiv.2211.08897, e-prints arXiv:2211.08897.

[30] X. Shen, X. Cheng, K. Liang, Deep Euler method: solving ODEs by approximating the local truncation error of the Euler method, 2020, http://dx.doi.org/10.48550/arXiv.2003.09573, e-prints arXiv:2003.09573.

[31] M. Poli, S. Massaroli, A. Yamashita, H. Asama, J. Park, Hypersolvers: Toward fast continuous-depth models, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 21105–21117, URL https://proceedings.neurips.cc/paper/2020/file/f1686b4badcf28d33ed632036c7ab0b8-Paper.pdf.

[32] Z. Huang, S. Liang, H. Zhang, H. Yang, L. Lin, Accelerating numerical solvers for large-scale simulation of dynamical system via neurvec, 2022, http://dx.doi.org/10.48550/arXiv.2208.03680, e-prints arXiv:2208.03680.

[33] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018, URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

[34] M. Barrault, Y. Maday, N.C. Nguyen, A.T. Patera, An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations, C. R. Math. Acad. Sci. Paris 339 (9) (2004) 667–672, http://dx.doi.org/10.1016/j.crma.2004.08.006.

[35] S. Chaturantabut, D.C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM J. Sci. Comput. 32 (5) (2010) 2737–2764, http://dx.doi.org/10.1137/090766498.

[36] K. Carlberg, C. Bou-Mosleh, C. Farhat, Efficient non-linear model reduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations, Internat. J. Numer. Methods Engrg. 86 (2) (2011) 155–181, http://dx.doi.org/10.1002/nme.3050.

[37] A. Abdulle, O. Budáč, A Petrov–Galerkin reduced basis approximation of the Stokes equation in parameterized geometries, C.R. Math. 353 (7) (2015) 641–645, http://dx.doi.org/10.1016/j.crma.2015.03.019.

[38] W. Dahmen, C. Plesken, G. Welper, Double greedy algorithms: Reduced basis methods for transport dominated problems, ESAIM: Math. Model. Numer. Anal. 48 (3) (2014) 623–663, http://dx.doi.org/10.1051/m2an/2013103.

[39] M. Yano, A space-time Petrov–Galerkin certified reduced basis method: Application to the Boussinesq equations, SIAM J. Sci. Comput. 36 (1) (2014) A232–A266, http://dx.doi.org/10.1137/120903300.

[40] J. Barnett, C. Farhat, Quadratic approximation manifold for mitigating the Kolmogorov barrier in nonlinear projection-based model order reduction, J. Comput. Phys. 464 (2022) http://dx.doi.org/10.1016/j.jcp.2022.111348, Paper No. 111348, 20.

[41] A. Ferrero, T. Taddei, L. Zhang, Registration-based model reduction of parameterized two-dimensional conservation laws, J. Comput. Phys. 457 (2022) http://dx.doi.org/10.1016/j.jcp.2022.111068, Paper No. 111068, 22.

[42] Y. Zhang, L. Feng, S. Li, P. Benner, Accelerating PDE constrained optimization by the reduced basis method: application to batch chromatography, Internat. J. Numer. Methods Engrg. 104 (11) (2015) 983–1007, http://dx.doi.org/10.1002/nme.4950.

[43] D. Degen, K. Veroy, F. Wellmann, Certified reduced basis method in geosciences: addressing the challenge of high-dimensional problems, Comput. Geosci. 24 (1) (2020) 241–259, http://dx.doi.org/10.1007/s10596-019-09916-6.

[44] M. Dihlmann, B. Haasdonk, A reduced basis Kalman filter for parametrized partial differential equations, ESAIM Control Optim. Calc. Var. 22 (3) (2016) 625–669, http://dx.doi.org/10.1051/cocv/2015019.

[45] M. Kärcher, S. Boyaval, M.A. Grepl, K. Veroy, Reduced basis approximation and a posteriori error bounds for 4D-Var data assimilation, Optim. Eng. 19 (3) (2018) 663–695, http://dx.doi.org/10.1007/s11081-018-9389-2.

[46] P. Chen, A. Quarteroni, G. Rozza, Reduced basis methods for uncertainty quantification, SIAM/ASA J. Uncertain. Quantif. 5 (1) (2017) 813–869, http://dx.doi.org/10.1137/151004550.

[47] B. Haasdonk, M. Ohlberger, Reduced basis method for finite volume approximations of parametrized linear evolution equations, ESAIM: Math. Model. Numer. Anal. 42 (2) (2008) 277–302, http://dx.doi.org/10.1051/m2an:2008001.

[48] U.M. Ascher, S.J. Ruuth, B.T.R. Wetton, Implicit-explicit methods for time-dependent partial differential equations, SIAM J. Numer. Anal. 32 (3) (1995) 797–823, http://dx.doi.org/10.1137/0732037.

[49] M.A. Grepl, Reduced-Basis Approximation a Posteriori Error Estimation for Parabolic Partial Differential Equations (Ph.D. thesis), Massachussetts Institute of Technology (MIT), Cambridge, USA, 2005, URL http://dspace.mit.edu/handle/1721.1/7582.

[50] L.F. Shampine, M.W. Reichelt, The MATLAB ODE suite, SIAM J. Sci. Comput. 18 (1) (1997) 1–22, http://dx.doi.org/10.1137/S1064827594276424, Dedicated to C. William Gear on the occasion of his 60th birthday.

[51] J.C. Butcher, Numerical Methods for Ordinary Differential Equations, John Wiley & Sons, Ltd., Chichester, 2008, p. xx+463, http://dx.doi.org/10.1002/9780470753767.

[52] A. Buffa, Y. Maday, A.T. Patera, C. Prud'homme, G. Turinici, A priori convergence of the greedy algorithm for the parametrized reduced basis method, ESAIM Math. Model. Numer. Anal. 46 (3) (2012) 595–603, http://dx.doi.org/10.1051/m2an/2011056.

[53] A. Pinkus, *N*-Widths in Approximation Theory, in: Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)], vol. 7, Springer-Verlag, Berlin, 1985, p. x+291, http://dx.doi.org/10.1007/978-3-642-69894-1.

[54] A. Cohen, R. DeVore, Kolmogorov widths under holomorphic mappings, IMA J. Numer. Anal. 36 (1) (2015) 1–12, http://dx.doi.org/10.1093/imanum/dru066.

[55] Q. Wang, J.S. Hesthaven, D. Ray, Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem, J. Comput. Phys. 384 (2019) 289–307, http://dx.doi.org/10.1016/j.jcp.2019.01.031.

[56] M.D. Buhmann, Radial Basis Functions: Theory and Implementations, in: Cambridge Monographs on Applied and Computational Mathematics, vol. 12, Cambridge University Press, Cambridge, 2003.

[57] H. Wendland, Scattered Data Approximation, in: Cambridge Monographs on Applied and Computational Mathematics, vol. 17, Cambridge University Press, Cambridge, 2005.

[58] C.M. Bishop, Pattern Recognition and Machine Learning, in: Information Science and Statistics, Springer, New York, 2006, p. xx+738, http://dx.doi.org/10.1007/978-0-387-45528-0.

[59] T. Hines, Python package containing tools for radial basis function (RBF) applications, 2023, https://github.com/treverhines/RBF.

[60] S. Chellappa, A Posteriori Error Estimation and Adaptivity for Model Order Reduction of Large-Scale Systems (Ph.D. thesis), Otto-von-Guericke-Universität, Magdeburg, Germany, 2023, http://dx.doi.org/10.25673/101396.

[61] J.W. Lee, A. Seidel-Morgenstern, Solving hyperbolic conservation laws with active counteraction against numerical errors: Isothermal fixed-bed adsorption, Chem. Eng. Sci. 207 (2019) 1309–1330, http://dx.doi.org/10.1016/j.ces.2019.07.053.

[62] C. Greif, K. Urban, Decay of the Kolmogorov N-width for wave problems, Appl. Math. Lett. 96 (2019) 216–222, http://dx.doi.org/10.1016/j.aml.2019.05.013.