Accurate and robust predictions for model order reduction via an adaptive, hybrid FOM/ROM approach

Sébastien Riffaud

▶ **To cite this version:**

Sébastien Riffaud. Accurate and robust predictions for model order reduction via an adaptive, hybrid FOM/ROM approach. 2023. hal-04361506

HAL Id: hal-04361506
https://inria.hal.science/hal-04361506

Preprint submitted on 22 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Graphical Abstract

Accurate and robust predictions for model order reduction via an adaptive, hybrid FOM/ROM approach

Sébastien Riffaud

Highlights

Accurate and robust predictions for model order reduction via an adaptive, hybrid FOM/ROM approach

Sébastien Riffaud

- An approach based on the principal directions of the reduced-order residual and reduced-order Jacobian matrix is proposed to improve the accuracy and computational efficiency of the ECSW method.
- A hybrid strategy that alternates between the FOM and ROM is employed to accelerate numerical simulations while maintaining accurate approximations.
- A residual-based error indicator is developed to determine when the ROM is not sufficiently accurate and the FOM needs to be employed.
- An adaptive-extended version of the hybrid approach is proposed to update the ROM with the solution snapshots generated by the FOM when the ROM was not sufficiently accurate.

Accurate and robust predictions for model order reduction via an adaptive, hybrid FOM/ROM approach

Sébastien Riffaud[a]

[a]*Sorbonne Université, Inria & CNRS, UMR 7598, Laboratoire Jacques-Louis Lions, 75012 Paris, France*

Abstract

In this paper, we introduce a hybrid approach that alternates between a high-fidelity model and a reduced-order model to speedup numerical simulations while maintaining accurate approximations. In particular, a residual-based error indicator is developed to determine when the reduced-order model is not sufficiently accurate and the high-fidelity model needs to be solved. Then, we propose an adaptive-extended version of the hybrid approach to update the reduced-order model with the solution snapshots generated by the high-fidelity model when the reduced-order model was not sufficiently accurate. In this way, we expect the reduced-order model to become more robust for predicting new out-of-sample solutions. The performance of the proposed method is evaluated on parametrized, time-dependent, nonlinear problems governed by the 1D Burgers' equation and 2D compressible Euler equations. The results demonstrate the accuracy and computational efficiency of the adaptive hybrid approach with respect to the high-fidelity model.

Keywords: Reduced-order model, Proper orthogonal decomposition, Hyper-reduction, Error estimation, Adaptive model reduction

1. Introduction

Many applications in science and engineering require efficient numerical simulations, either due to runtime constraints in the case of large-scale computational models or due to the large number of simulations to run in the case of many-query problems. Projection-based reduced-order models (ROMs) [1, 2, 3, 4, 5, 6, 7, 8, 9] have been developed to dramatically decrease the computational cost associated with numerical solutions of parametric problems

governed by partial differential equations (PDEs). They typically consists of an offline stage in which training high-fidelity solutions are collected to define a data-driven approximation space. A popular dimensionality reduction technique for constructing the reduced basis that spans this approximation space is the proper orthogonal decomposition (POD) [10, 2]. Then, the low-dimensionality of the resulting reduced basis is exploited during an online stage to enable fast or real-time predictions of new out-of-sample solutions.

To construct accurate ROMs over a wide range of input parameters, high-fidelity solution snapshots are collected for different training time-instances and input parameters. However, there are two major difficulties associated with this offline procedure. First, the number of high-fidelity simulations for sampling the solution manifold is limited in practice due to the expensive computational cost of the high-fidelity model (or high-dimensional model (HDM)). Second, if the training solution snapshots are too different from the predicted solution, the ROM may fail to deliver accurate approximations.

Several approaches have been proposed in the literature to address the aforementioned issues. Peherstorfer et al. [11, 12] developed an online adaptive bases and adaptive sampling for transport-dominated problems. This method uses the adaptive discrete empirical interpolation method and rank-one updates to adapt the reduced basis with the locality in space and time of propagating coherent structures. In [13, 14, 15], a hybrid HDM/ROM approach is presented. The main idea is to alternate between the HDM and ROM on the fly to achieve a prescribed accuracy level. An *a posteriori* error estimation is also developed to determine when the ROM is not sufficiently accurate and the HDM needs to be solved. Similarly, Zucatti et al. [16] introduced an adaptive, training-free ROM for convection-dominated problems based on hybrid snapshots. The latter are generated by combining space-local HDM and ROM solutions. An error strategy is employed to identify regions where the ROM is inaccurate and the HDM needs to be solved. In [17, 18], an adaptive h-refinement for ROMs is proposed. In this approach, the basis vectors that most contributes to the error are identified using an adjoint-based error indicator. These vectors are then refined so that the estimated error is lower than a prescribed tolerance.

In this work, we focus on least-squares Petrov-Galerkin (LSPG) projection-based ROMs [19, 20] equipped with the energy-conserving sampling and weighting (ECSW) hyper-reduction method [21, 22]. In particular, we propose an accurate and efficient approach for the ECSW method based on the principal directions of the reduced-order residual and reduced-order Jacobian

matrix [23]. Then, we adopt a hybrid HDM/ROM approach to accelerate numerical simulations while maintaining accurate approximations. There are two main differences with the method presented in [15]. First, they consider only Galerkin projection-based ROMs equipped with the discrete empirical interpolation method (DEIM) [24, 25]. Second, we propose a different residual-based error indicator for determining when the ROM is not sufficiently accurate. Instead of fixing a tolerance directly on the residual norm, we derive an *a posteriori* estimation that connects the time-history of the residual norm to the error norm. This interpolation is then fitted at some training time-parameter instances in a machine-learning fashion. Finally, we prescribe an accuracy level on the estimated error, which automatically determines the corresponding tolerance on the residual norm. The computational efficiency of the resulting method depends on the ROM accuracy to predict new out-of-sample solutions. If the HDM is used in a large part of the computations, the hybrid model will be accurate but computationally expensive to solve. Conversely, if the ROM is sufficient to approximate the solution most of the time, this approach allows to significantly reduce the computational cost of numerical simulations. For this reason, we also propose an adaptive-extended version of the hybrid model which aims at reducing the use of the HDM. The main idea is to enrich the ROM with the solution snapshots generated by the HDM when the ROM was not sufficiently accurate. In this way, we expect the ROM to become more robust for predicting new out-of-sample solutions.

The remainder of the paper is organized as follows. In Section 2, we introduce the HDM and ROM used in this work. Then, Section 3 presents in details the hybrid HDM/ROM approach and the underlying residual-based error indicator. In Section 4, we describe the adaptive-extended version of the hybrid model. Section 5 demonstrates the accuracy of the proposed method and the significant reduction of the computational cost for two different applications. Finally, Section 6 draws some conclusions and perspectives.

2. Projection-based reduced-order model

2.1. High-dimensional model

Let the parameter domain $\mathcal{D}$ be a closed and bounded subset of the Euclidean space $\mathbb{R}^{N_p}$. In this work, we consider parametrized, time-dependent,

nonlinear, discrete problems of the form

$$\begin{cases} \mathbf{r}(\mathbf{u}(t_k;\boldsymbol{\theta}),t_k;\boldsymbol{\theta}) = \mathbf{0} & \forall k \in \{1,\dots,N_t\} \\ \mathbf{u}(t_0;\boldsymbol{\theta}) = \mathbf{u}_0(\boldsymbol{\theta}), \end{cases} \tag{1}$$

where $t_k \geqslant 0$ denotes the k-th time-instance, $\boldsymbol{\theta} \in \mathcal{D}$ denotes the input parameters, $\mathbf{u} : \mathbb{R}_+ \times \mathcal{D} \to \mathbb{R}^{N_{\text{dof}}}$ denotes the discrete solution, and $\mathbf{r} : \mathbb{R}^{N_{\text{dof}}} \times \mathbb{R}_+ \times \mathcal{D} \to \mathbb{R}^{N_{\text{dof}}}$ denotes the nonlinear residual resulting from the discretization of the PDE. This computational model will be referred to as the HDM in the following.

2.2. Reduced-order model

In the ROM, the high-dimensional solution $\mathbf{u}(t_k;\boldsymbol{\theta})$ is approximated on a low-dimensional subspace to reduce the number of degrees of freedom (i.e., $N_{\text{v}} \ll N_{\text{dof}}$):

$$\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}) = \mathbf{u}_{\text{off}}(\boldsymbol{\theta}) + \mathbf{V}\mathbf{y}(t_k;\boldsymbol{\theta}),$$

where the offset $\mathbf{u}_{\text{off}}(\boldsymbol{\theta}) \in \mathbb{R}^{N_{\text{dof}}}$ is set to the initial condition $\mathbf{u}_0(\boldsymbol{\theta})$, the reduced basis $\mathbf{V} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{v}}}$ is construted by POD, and the reduced coordinates $\mathbf{y}(t;\boldsymbol{\theta}) \in \mathbb{R}^{N_{\text{v}}}$ are determined by the LSPG method equipped with the ECSW method.

2.2.1. Proper orthogonal decomposition

During the offline stage, solution snapshots of the HDM are collected for different time-parameter instances. Let the resulting snapshot database be $\mathbf{S} = [\mathbf{u}(t_1^{\text{train}},\boldsymbol{\theta}_1^{\text{train}}),\dots,\mathbf{u}(t_{N_{\text{s}}}^{\text{train}},\boldsymbol{\theta}_{N_{\text{s}}}^{\text{train}})] \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{s}}}$. The reduced basis $\mathbf{V}$ is computed by POD to minimize the projection error of the snapshots onto this basis. In particular, the reduced basis dimension N_{v} is defined as the minimum integer such that the relative projection error is less than a user-defined tolerance $\varepsilon_{\text{pod}} \in [0,1]$:

$$\left\| \mathbf{S} - \mathbf{S}^{\text{off}} - \mathbf{V}\mathbf{V}^T(\mathbf{S} - \mathbf{S}^{\text{off}}) \right\|_F \leqslant \varepsilon_{\text{pod}} \left\| \mathbf{S} \right\|_F ,$$

where $\mathbf{S}^{\text{off}} = [\mathbf{u}_{\text{off}}(\boldsymbol{\theta}_1^{\text{train}}),\dots,\mathbf{u}_{\text{off}}(\boldsymbol{\theta}_{N_{\text{s}}}^{\text{train}})] \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{s}}}$, and $\|\cdot\|_F$ stands for the Frobenius norm.

2.2.2. Least-squares Petrov-Galerkin method

During the online stage, the reduced coordinates $\mathbf{y}(t_k;\boldsymbol{\theta})$ are determined by LSPG projection:

$$\begin{cases} \min\limits_{\mathbf{y}(t_k;\boldsymbol{\theta})\in\mathbb{R}^{N_\mathrm{v}}} \|\mathbf{r}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}),t_k;\boldsymbol{\theta})\|_2^2 & \forall k\in\{1,\dots,N_t\} \\ \mathbf{y}(t_0;\boldsymbol{\theta}) = \mathbf{V}^T\big(\mathbf{u}_0(\boldsymbol{\theta}) - \mathbf{u}_\mathrm{off}(\boldsymbol{\theta})\big). \end{cases} \tag{2}$$

Notably, the first optimality condition leads to the equivalent problem:

$$\begin{cases} \mathbf{W}(t_k;\boldsymbol{\theta})^T\mathbf{r}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}),t_k;\boldsymbol{\theta}) = \mathbf{0} & \forall k\in\{1,\dots,N_t\} \\ \mathbf{y}(t_0;\boldsymbol{\theta}) = \mathbf{V}^T\big(\mathbf{u}_0(\boldsymbol{\theta}) - \mathbf{u}_\mathrm{off}(\boldsymbol{\theta})\big), \end{cases}$$

where $\mathbf{W}(t_k;\boldsymbol{\theta}) = \mathbf{J}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}),t_k;\boldsymbol{\theta})\mathbf{V} \in \mathbb{R}^{N_\mathrm{dof}\times N_\mathrm{v}}$ with $\mathbf{J}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}),t_k;\boldsymbol{\theta}) \in \mathbb{R}^{N_\mathrm{dof}\times N_\mathrm{dof}}$ the Jacobian of the residual $\mathbf{r}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}),t_k;\boldsymbol{\theta})$ with respect to $\widetilde{\mathbf{u}}$.

2.2.3. Energy-conserving sampling and weighting method

In this work, the residual minimization problem (2) is solved by the Gauss-Newton algorithm, which requires to evaluate the reduced-order residual and reduced-order Jacobian matrix, defined respectively as

$$\mathbf{W}(t_k;\boldsymbol{\theta})^T\mathbf{r}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}),t_k;\boldsymbol{\theta}) \quad \text{and} \quad \mathbf{W}(t_k;\boldsymbol{\theta})^T\mathbf{W}(t_k;\boldsymbol{\theta}).$$

However, if the residual is nonlinear with respect to the solution, then the evaluation of the resulting reduced-order quantities typically scales with the size of the HDM, unless hyper-reduction techniques are employed. For this reason, we consider the ECSW method, in which the standard scalar product is approximated by the weighted sum:

$$\mathbf{a}^T\mathbf{b} \approx \sum_{i=1}^{N_\mathrm{dof}} \omega_i a_i b_i,$$

where $\mathbf{a},\mathbf{b}\in\mathbb{R}^{N_\mathrm{dof}}$, and only a few weights $\omega_i \geqslant 0$ are nonzero to speedup computations (i.e., $\|\boldsymbol{\omega}\|_0 \ll N_\mathrm{dof}$, where $\|\cdot\|_0$ stands for the ℓ_0 pseudo-norm). The weights $\boldsymbol{\omega} = [\omega_1,\dots,\omega_{N_\mathrm{dof}}] \in \mathbb{R}^{N_\mathrm{dof}}$ are determined empirically during the offline stage to best approximate the exact scalar product, which typically leads to an approximation problem of the form:

$$\mathbf{C}\boldsymbol{\omega} \approx \mathbf{d},$$

where $\mathbf{C} \in \mathbb{R}^{N_c \times N_{dof}}$ and $\mathbf{d} \in \mathbb{R}^{N_c}$ are defined in details in Appendix A. The weights $\boldsymbol{\omega}$ are then solution to the non-negative least-squares (NNLS) problem

$$\min_{\boldsymbol{\omega} \geqslant 0} \|\mathbf{C}\boldsymbol{\omega} - \mathbf{d}\|_2^2 , \tag{3}$$

which is solved by the Matlab implementation of the Lawson-Hanson algorithm [26]. This algorithm promotes sparsity in the solution and terminates when the stopping criterion $\|\mathbf{C}\boldsymbol{\omega} - \mathbf{d}\|_2 \leqslant \varepsilon_{\text{ecsw}} \|\mathbf{d}\|_2$ is satisfied for a user-defined tolerance $\varepsilon_{\text{ecsw}} \in [0, 1]$.

3. Hybrid HDM/ROM model

Given the previous HDM and ROM, we now introduce the hybrid HDM/ROM approach. The objective is to accelerate numerical simulation while maintaining accurate approximations. The main idea is to compute the solution using the HDM only when the ROM is not sufficiently accurate. Specifically, we want the relative space-time error to be smaller than a user-defined tolerance $\varepsilon_{\text{err}} \in [0, 1]$:

$$\sqrt{\sum_{k=1}^{N_t} \|\mathbf{u}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2^2} \leqslant \varepsilon_{\text{err}} \sqrt{\sum_{k=1}^{N_t} \|\mathbf{u}(t_k;\boldsymbol{\theta})\|_2^2}.$$

For this purpose, we impose that

$$\|\mathbf{u}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2 \leqslant \varepsilon_{\text{err}} \|\mathbf{u}(t_k;\boldsymbol{\theta})\|_2 \leqslant \frac{\varepsilon_{\text{err}}}{1 - \varepsilon_{\text{err}}} \|\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2 ,$$

where the last inequality comes from

$$\begin{aligned} \|\mathbf{u}(t_k;\boldsymbol{\theta})\|_2 &\leqslant \|\mathbf{u}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2 + \|\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2 \\ &\leqslant \varepsilon_{\text{err}} \|\mathbf{u}(t_k;\boldsymbol{\theta})\|_2 + \|\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2 \\ &\leqslant \frac{1}{1 - \varepsilon_{\text{err}}} \|\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2 . \end{aligned}$$

The right-hand side can be efficiently evaluated since it scales with the size of the ROM:

$$\|\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2^2 = \|\mathbf{u}_{\text{off}}(\boldsymbol{\theta})\|_2^2 + 2 \left\langle \mathbf{V}^T \mathbf{u}_{\text{off}}(t_k;\boldsymbol{\theta}), \mathbf{y}(t_k;\boldsymbol{\theta}) \right\rangle + \|\mathbf{y}(t_k;\boldsymbol{\theta})\|_2^2 ,$$

where $\|\mathbf{u}_{\text{off}}(\boldsymbol{\theta})\|_2^2$ and $\mathbf{V}^T\mathbf{u}_{\text{off}}(t_k;\boldsymbol{\theta})$ can be precomputed once for all. However, computing the error $\|\mathbf{u}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2$ is computationally prohibitive as it requires the evaluation of the high-dimensional solution $\mathbf{u}(t_k;\boldsymbol{\theta})$. For this reason, we employ an error indicator $\delta_{\text{err}}(t_k;\boldsymbol{\theta}) \approx \|\mathbf{u}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2$. The latter is presented in Section 3.1, and the resulting hybrid model is described in Algorithm 1.

Algorithm 1 Hybrid HDM/ROM model

Input: $\boldsymbol{\theta} \in \mathcal{D}$, $\varepsilon_{\text{pod}} \in [0,1]$, $\varepsilon_{\text{ecsw}} \in [0,1]$, and $\varepsilon_{\text{err}} \in [0,1]$.
Output: $\{\widehat{\mathbf{u}}(t_1;\boldsymbol{\theta}), \ldots, \widehat{\mathbf{u}}(t_{N_t};\boldsymbol{\theta})\} \in \mathbb{R}^{N_{\text{dof}}}$.

```
Initialize û(t_0; θ) = u_0(θ);
for k ∈ {1, ..., N_t} do
    Compute ũ(t_k; θ) by solving the ROM equation (2);
    if δ_err(t_k; θ) ⩽ ε_err/(1−ε_err) ‖ũ(t_k; θ)‖_2 then
        Set û(t_k; θ) = ũ(t_k; θ);
    else
        Compute u(t_k; θ) by solving the HDM equation (1);
        Set û(t_k; θ) = u(t_k; θ);
    end
end
```

3.1. Residual-based error estimation

The objective of this section is to develop an error indicator that depends on the time-history of the residual norm $\|\mathbf{r}(\mathbf{u}(t_k;\boldsymbol{\theta}), t_k;\boldsymbol{\theta})\|_2$. For this purpose, we follow the classical approach used to derive *a posteriori* error estimation [27, 28, 29, 30, 31].

Let $\mathbf{f}(\mathbf{u}(t_{k-1};\boldsymbol{\theta}), t_k;\boldsymbol{\theta}) = \mathbf{u}(t_k;\boldsymbol{\theta})$ be the function that pushes forward the solution to the next time-step by solving the high-dimensional residual (1). Note that this function depends only on the previous solution $\mathbf{u}(t_{k-1};\boldsymbol{\theta})$ since the time is integrated in this work using single-step methods. Furthermore, let's define $\mathbf{f}(\widetilde{\mathbf{u}}(t_{k-1};\boldsymbol{\theta}), t_k;\boldsymbol{\theta}) = \overline{\mathbf{u}}(t_k;\boldsymbol{\theta})$ as illustrated in Figure 1.

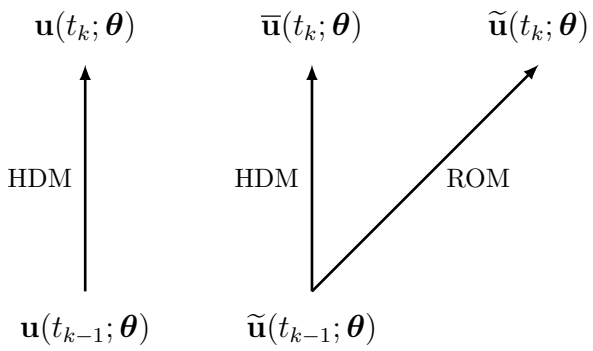


Figure 1: Illustration of the different quantities involved in the error estimation

Assuming $\mathbf{f}(\mathbf{u}(t_{k-1};\boldsymbol{\theta}), t_k;\boldsymbol{\theta})$ is Lipschitz continuous with respect to $\mathbf{u}(t_{k-1};\boldsymbol{\theta})$ and $\mathbf{r}(\mathbf{u}(t_k;\boldsymbol{\theta}), t_k;\boldsymbol{\theta})$ is inverse Lipschitz continuous with respect to $\mathbf{u}(t_k;\boldsymbol{\theta})$ yields respectively to

$$\left\|\mathbf{u}(t_k;\boldsymbol{\theta}) - \overline{\mathbf{u}}(t_k;\boldsymbol{\theta})\right\|_2 \leqslant \alpha(t_k,\boldsymbol{\theta}) \left\|\mathbf{u}(t_{k-1};\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_{k-1};\boldsymbol{\theta})\right\|_2$$

$$\left\|\overline{\mathbf{u}}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\right\|_2 \leqslant \beta(t_k,\boldsymbol{\theta}) \left\|\mathbf{r}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}), t_k;\boldsymbol{\theta})\right\|_2,$$

where $\alpha(t_k,\boldsymbol{\theta})$ and $\beta(t_k,\boldsymbol{\theta})$ denote the corresponding Lipschitz and inverse Lipschitz constants. An upper bound of the error is then given by

$$\begin{aligned}
&\left\|\mathbf{u}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\right\|_2 \\
&\quad\leqslant \left\|\mathbf{u}(t_k;\boldsymbol{\theta}) - \overline{\mathbf{u}}(t_k;\boldsymbol{\theta})\right\|_2 + \left\|\overline{\mathbf{u}}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\right\|_2 \\
&\quad\leqslant \alpha(t_k,\boldsymbol{\theta}) \left\|\mathbf{u}(t_{k-1};\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_{k-1};\boldsymbol{\theta})\right\|_2 + \beta(t_k,\boldsymbol{\theta}) \left\|\mathbf{r}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}), t_k;\boldsymbol{\theta})\right\|_2 \\
&\quad= \lambda_0(t_k,\boldsymbol{\theta}) \left\|\mathbf{u}(t_0;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_0;\boldsymbol{\theta})\right\|_2 + \sum_{i=1}^{k} \lambda_i(t_k,\boldsymbol{\theta}) \left\|\mathbf{r}(\widetilde{\mathbf{u}}(t_i;\boldsymbol{\theta}), t_i;\boldsymbol{\theta})\right\|_2,
\end{aligned}$$

where $\lambda_i(t_k,\boldsymbol{\theta}) = \left(\prod_{j=i+1}^{k} \alpha(t_j,\boldsymbol{\theta})\right) \beta(t_i,\boldsymbol{\theta})$ with the definition $\beta(t_0,\boldsymbol{\theta}) = 1$. Unfortunately, evaluating the true constants $\lambda_i(t_k,\boldsymbol{\theta})$ is difficult in practice. For this reason, we propose to fit them at some training time-parameter

instances $(t_k, \boldsymbol{\theta}) \in \{(t_1^{\text{train}}, \boldsymbol{\theta}_1^{\text{train}}), \ldots, (t_{N_\text{s}}^{\text{train}}, \boldsymbol{\theta}_{N_\text{s}}^{\text{train}})\}$ as follows

$$\begin{aligned}
\tau(t_k, \boldsymbol{\theta}) &= \frac{\|\mathbf{u}(t_k;\boldsymbol{\theta}) - \overline{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2 \, \|\overline{\mathbf{u}}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2}{\|\mathbf{u}(t_k;\boldsymbol{\theta}) - \overline{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2 + \|\overline{\mathbf{u}}(t_k;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2}, \\
\alpha^*(t_k, \boldsymbol{\theta}) &= \frac{\tau(t_k, \boldsymbol{\theta})}{\|\mathbf{u}(t_{k-1};\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_{k-1};\boldsymbol{\theta})\|_2}, \\
\beta^*(t_k, \boldsymbol{\theta}) &= \begin{cases} 1 & \text{if } k = 0 \\ \frac{\tau(t_k,\boldsymbol{\theta})}{\|\mathbf{r}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}),t_k;\boldsymbol{\theta})\|_2} & \text{otherwise,} \end{cases} \\
\lambda_i^*(t_k, \boldsymbol{\theta}) &= \left(\prod_{j=i+1}^{k} \alpha^*(t_j, \boldsymbol{\theta}) \right) \beta^*(t_i, \boldsymbol{\theta}).
\end{aligned}$$

The constants $\lambda_i^*(t_k, \boldsymbol{\theta})$ are then interpolated for new out-of-sample time-parameter instances $(t_k, \boldsymbol{\theta}) \notin \{(t_1^{\text{train}}, \boldsymbol{\theta}_1^{\text{train}}), \ldots, (t_{N_s}^{\text{train}}, \boldsymbol{\theta}_{N_s}^{\text{train}})\}$ by

$$\lambda_i^*(t_k, \boldsymbol{\theta}) = \mathcal{I}\left(t_k, \boldsymbol{\theta}; \{\boldsymbol{\theta}_n^{\text{train}}\}_{n \in \Theta_k}, \{\lambda_i^*(t_k, \boldsymbol{\theta}_n^{\text{train}})\}_{n \in \Theta_k}\right),$$

where $\Theta_k = \left\{n \in \{1, \ldots, N_\text{s}\} : t_n^{\text{train}} = t_k\right\}$, and $\mathcal{I}$ denotes the interpolation method. In particular, we employ here a radial basis function (RBF) interpolation based on polyharmonic splines (see Appendix B). The resulting error indicator is finally given by

$$\delta_{\text{err}}(t_k; \boldsymbol{\theta}) = \lambda_0^*(t_k, \boldsymbol{\theta}) \, \|\mathbf{u}(t_0;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_0;\boldsymbol{\theta})\|_2 + \sum_{i=1}^{k} \lambda_i^*(t_k, \boldsymbol{\theta}) \, \|\mathbf{r}(\widetilde{\mathbf{u}}(t_i;\boldsymbol{\theta}), t_i; \boldsymbol{\theta})\|_2 \, .$$

Note that since the offset is defined in this work as $\mathbf{u}_{\text{off}}(\boldsymbol{\theta}) = \mathbf{u}(t_0;\boldsymbol{\theta})$, the initial error $\|\mathbf{u}(t_0;\boldsymbol{\theta}) - \widetilde{\mathbf{u}}(t_0;\boldsymbol{\theta})\|_2$ vanishes in the expression of $\delta_{\text{err}}(t_k;\boldsymbol{\theta})$.

4. Adaptive, hybrid HDM/ROM model

The computational cost of the hybrid model depends on the number of time-steps in which the HDM is solved. For this reason, we propose to adopt an adaptive approach which aims at reducing the use of the HDM. The main idea is to enrich the ROM with the new solution snapshots generated by the HDM when the ROM was not sufficiently accurate. In this way, we expect the ROM to be more robust for predicting new solutions.

Let $\mathbf{S}_{\text{old}}, \mathbf{S}_{\text{old}}^{\text{off}} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{s}_{\text{old}}}}$ be the snapshot databases, $\mathbf{V}_{\text{old}} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{v}_{\text{old}}}}$ be the corresponding reduced basis, and $\boldsymbol{\omega}_{\text{old}} \in \mathbb{R}^{N_{\text{dof}}}$ be the weights associated with the approximation problem $\mathbf{C}_{\text{old}}\boldsymbol{\omega}_{\text{old}} \approx \mathbf{d}_{\text{old}}$. Now, assume that new

snapshots $\mathbf{S}_{\text{new}}, \mathbf{S}^{\text{off}}_{\text{new}} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{s}_{\text{new}}}}$ become available, the ROM is updated using the new reduced basis $\mathbf{V} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{v}}}$ and weights $\boldsymbol{\omega} \in \mathbb{R}^{N_{\text{dof}}}$ described in Sections 4.1 and 4.2 respectively. The resulting adaptive-extended version of the hybrid model is presented in Algorithm 2.

Algorithm 2 Adaptive, hybrid HDM/ROM model

Input: $\boldsymbol{\theta} \in \mathcal{D}$, $\varepsilon_{\text{pod}} \in [0,1]$, $\varepsilon_{\text{ecsw}} \in [0,1]$, and $\varepsilon_{\text{err}} \in [0,1]$.
Output: $\widehat{\mathbf{u}}(t_k;\boldsymbol{\theta}) \in \mathbb{R}^{N_{\text{dof}}}$.

```
Initialize û(t_0; θ) = u_0(θ);
Initialize S_new ∈ R^{N_dof × 0}, S^off_new ∈ R^{N_dof × 0}, and N_{s_new} = 0;
for k ∈ {1, ..., N_t} do
    Compute ũ(t_k; θ) by solving the ROM equation (2);
    if δ_err(t_k; θ) ⩽ ε_err / (1 − ε_err) ‖ũ(t_k; θ)‖_2 then
        Set û(t_k; θ) = ũ(t_k; θ);
    else
        Compute u(t_k; θ) by solving the HDM equation (1);
        Set û(t_k; θ) = u(t_k; θ);
        Set S_new = [S_new u(t_k; θ)] and S^off_new = [S^off_new u_off(θ)];
        Set N_{s_new} = N_{s_new} + 1;
    end
end
if N_{s_new} > 0 then
    Update the reduced basis V using Algorithm 3;
    Update the weights ω using Algorithm 4;
end
```

4.1. Update of the reduced basis

We want the new reduced basis $\mathbf{V}$ to verify

$$\left\|\mathbf{S} - \mathbf{S}^{\text{off}} - \mathbf{V}\mathbf{V}^T(\mathbf{S} - \mathbf{S}^{\text{off}})\right\|_F \leqslant \varepsilon_{\text{pod}} \left\|\mathbf{S}\right\|_F ,$$

where $\mathbf{S} = [\mathbf{S}_{\text{old}}\ \mathbf{S}_{\text{new}}] \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{s}}}$, $\mathbf{S}^{\text{off}} = [\mathbf{S}^{\text{off}}_{\text{old}}\ \mathbf{S}^{\text{off}}_{\text{new}}] \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{s}}}$, and $N_{\text{s}} = N_{\text{s}_{\text{old}}} + N_{\text{s}_{\text{new}}}$. To this end, we impose that

$$\left\|\mathbf{S}_{\text{old}} - \mathbf{S}^{\text{off}}_{\text{old}} - \mathbf{V}\mathbf{V}^T(\mathbf{S}_{\text{old}} - \mathbf{S}^{\text{off}}_{\text{old}})\right\|_F \leqslant \varepsilon_{\text{pod}} \left\|\mathbf{S}_{\text{old}}\right\|_F \tag{4a}$$

$$\left\|\mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}} - \mathbf{V}\mathbf{V}^T(\mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}})\right\|_F \leqslant \varepsilon_{\text{pod}} \left\|\mathbf{S}_{\text{new}}\right\|_F , \tag{4b}$$

and we define the new reduced basis as $\mathbf{V} = [\mathbf{V}_{\text{old}}\ \mathbf{V}_{\text{new}}]$, where $\mathbf{V}_{\text{new}} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{v}_{\text{new}}}}$ contains the additional basis vectors necessary to enforce equation (4b). In this way, the snapshot matrices $\mathbf{S}_{\text{old}}$ and $\mathbf{S}^{\text{off}}_{\text{old}}$ do not need to be stored in memory, which would be computationally prohibitive. Algorithm 3 presents in detail the method used to obtain $\mathbf{V}$. First, the POD vectors $\mathbf{V}_{\text{new}}$ associated with $\mathbf{S}_{\text{new}}$ that are not already in $\mathbf{V}_{\text{old}}$ are extracted by the method of snapshots in Lines 1–3. The dimension of $\mathbf{V}_{\text{new}}$ is then chosen in Line 4 such that equation (4b) is satisfied:

$$
\begin{aligned}
&\left\|\mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}} - \mathbf{V}\mathbf{V}^T(\mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}})\right\|_F^2 \\
&\quad = \left\|\mathbf{S}^{\perp}_{\text{new}} + \mathbf{V}_{\text{old}}\mathbf{V}^T_{\text{old}}(\mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}}) - \mathbf{V}\mathbf{V}^T(\mathbf{S}^{\perp}_{\text{new}} + \mathbf{V}_{\text{old}}\mathbf{V}^T_{\text{old}}(\mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}}))\right\|_F^2 \\
&\quad = \left\|\mathbf{S}^{\perp}_{\text{new}} + \mathbf{V}_{\text{old}}\mathbf{V}^T_{\text{old}}(\mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}}) - \mathbf{V}_{\text{new}}\mathbf{V}^T_{\text{new}}\mathbf{S}^{\perp}_{\text{new}} - \mathbf{V}_{\text{old}}\mathbf{V}^T_{\text{old}}(\mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}}))\right\|_F^2 \\
&\quad = \left\|\mathbf{S}^{\perp}_{\text{new}} - \mathbf{V}_{\text{new}}\mathbf{V}^T_{\text{new}}\mathbf{S}^{\perp}_{\text{new}}\right\|_F^2 \\
&\quad = \sum_{i=N_{\text{v}_{\text{new}}}+1}^{N_{\text{s}_{\text{new}}}} \sigma_i^2 \qquad \text{(according to the Eckart-Young theorem [32])} \\
&\quad \leqslant \varepsilon^2_{\text{pod}} \left\|\mathbf{S}_{\text{new}}\right\|_F^2 .
\end{aligned}
$$

Next, the first $N_{\text{v}_{\text{new}}}$ columns of $\mathbf{V}_{\text{new}} = \mathbf{S}^{\perp}_{\text{new}}\mathbf{U}\mathbf{\Sigma}^{-1}$ are added to $\mathbf{V}_{\text{old}}$ in Line 5 to obtain $\mathbf{V}$. If the matrix $\mathbf{V}$ is not unitary due to numerical error, then a re-orthogonalization step can finally be performed in Line 6.

Algorithm 3 Update of the reduced basis

Input: $\mathbf{V}_{\text{old}} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{v}_{\text{old}}}}$, $\mathbf{S}_{\text{new}} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{s}_{\text{new}}}}$, $\mathbf{S}^{\text{off}}_{\text{new}} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{s}_{\text{new}}}}$, and $\varepsilon_{\text{pod}} \in [0, 1]$.

Output: $\mathbf{V} \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{v}}}$.

1: Define $\mathbf{S}^{\perp}_{\text{new}} = \mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}} - \mathbf{V}_{\text{old}}\mathbf{V}^T_{\text{old}}(\mathbf{S}_{\text{new}} - \mathbf{S}^{\text{off}}_{\text{new}})$;
2: Compute the correlation matrix $\mathbf{C} = (\mathbf{S}^{\perp}_{\text{new}})^T\mathbf{S}^{\perp}_{\text{new}}$;
3: Compute the SVD (or eigendecomposition) of $\mathbf{C} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T$;
4: Find the smallest integer $N_{\text{v}_{\text{new}}}$ such that $\sum_{i=N_{\text{v}_{\text{new}}}+1}^{N_{\text{s}_{\text{new}}}} \sigma_i^2 \leqslant \varepsilon^2_{\text{pod}} \left\|\mathbf{S}_{\text{new}}\right\|_F^2$;
5: Set $\mathbf{V} = \left[\mathbf{V}_{\text{old}} \quad \mathbf{S}^{\perp}_{\text{new}}\left[\sigma_1^{-1}\mathbf{u}_1 \quad \dots \quad \sigma^{-1}_{N_{\text{v}_{\text{new}}}}\mathbf{u}_{N_{\text{v}_{\text{new}}}}\right]\right]$;
6: *Optional:* orthonormalize $\mathbf{V}$.

4.2. Update of the reduced mesh

Given $\mathbf{S}_{\text{new}}$, $\mathbf{S}^{\text{off}}_{\text{new}}$, and $\mathbf{V}_{\text{new}}$ defined in Section 4.1, we can assemble the additional approximation problem associated with $\mathbf{C}_{\text{new}}$ and $\mathbf{d}_{\text{new}}$. We then

want the weights $\boldsymbol{\omega} \in \mathbb{R}^{N_{\text{dof}}}$ solution to the NNLS problem (3) given by

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_{\text{old}} \\ \mathbf{C}_{\text{new}} \end{bmatrix} \qquad \text{and} \qquad \mathbf{d} = \begin{bmatrix} \mathbf{d}_{\text{old}} \\ \mathbf{d}_{\text{new}} \end{bmatrix}.$$

To this end, the main idea is to introduce new nonzero elements in $\boldsymbol{\omega}_{\text{old}}$ to obtain the weights $\boldsymbol{\omega}$ that satisfy the new approximation problem $\mathbf{C}\boldsymbol{\omega} \approx \mathbf{d}$. As the NNLS solver [26] iteratively improves the solution until the stopping criterion is satisfied, the weights $\boldsymbol{\omega}$ can be efficiently updated by initializing the NNLS solver with $\boldsymbol{\omega}_{\text{old}}$ as initial guess. The resulting method is presented in Algorithm 4.

Remark 1. *In order to accelerate the resolution of the NNLS problem* (3)*, the number of rows in* $\mathbf{C}$ *is reduced in practice. Given the notations introduced in Appendix A, let* $\mathbf{S}'_{\text{old}}$ *be the constraints of the old approximation problem (i.e.,* $\mathbf{S}'_{\text{old}} = \mathbf{C}_3^{\text{old}}$*),* $\mathbf{V}'_{\text{old}}$ *be the associated reduced basis (i.e.,* $\mathbf{V}'_{\text{old}} = \mathbf{C}_4^{\text{old}}$*), and* $\boldsymbol{\Sigma}'_{\text{old}}$ *be the associated singular values. The new constraints* $\mathbf{S}'_{\text{new}}$ *are approximated on the reduced basis* $\mathbf{V}' = [\mathbf{V}'_{\text{old}}\ \mathbf{V}'_{\text{new}}]$*, where the additional basis vectors* $\mathbf{V}'_{\text{new}}$ *are extracted using Algorithm 3. Then, the matrix* $\mathbf{C}_{new}$ *can be written as* $\mathbf{C}_{new} = \boldsymbol{\Sigma}'_{\text{add}}\mathbf{V}'_{\text{old}} + \boldsymbol{\Sigma}'_{\text{new}}\mathbf{V}'_{\text{new}}$ *with* $\boldsymbol{\Sigma}'_{\text{add}} = \text{diag}(\mathbf{1}^T(\mathbf{V}'_{\text{old}})^T\mathbf{S}'_{\text{new}})$*. The matrix* $\mathbf{C} = \boldsymbol{\Sigma}'\mathbf{V}'$ *is finally given by*

$$\boldsymbol{\Sigma}' = \begin{bmatrix} \boldsymbol{\Sigma}'_{\text{old}} + \boldsymbol{\Sigma}'_{\text{add}} & \\ & \boldsymbol{\Sigma}'_{\text{new}} \end{bmatrix} \quad \textit{and} \quad \mathbf{V}' = \begin{bmatrix} \mathbf{V}'_{\text{old}} \\ \mathbf{V}'_{\text{new}} \end{bmatrix}.$$

Algorithm 4 Update of the reduced mesh [26]

Input: $\boldsymbol{\omega}_{\text{old}} \in \mathbb{R}^{N_{\text{dof}}}$, $\mathbf{C} \in \mathbb{R}^{N_{\text{c}} \times N_{\text{dof}}}$, $\mathbf{d} \in \mathbb{R}^{N_{\text{c}}}$, and $\varepsilon_{\text{ecsw}} \in [0, 1]$.
Output: $\boldsymbol{\omega} \in \mathbb{R}^{N_{\text{dof}}}$.

$\boldsymbol{\omega} = \boldsymbol{\omega}_{\text{old}}$;
$\mathcal{Z} = \{i \in \{1, \ldots, N_{\text{dof}}\} \mid \omega_i = 0\}$;
$\mathcal{P} = \{i \in \{1, \ldots, N_{\text{dof}}\} \mid \omega_i > 0\}$;
$\mathbf{w} = \mathbf{C}^T(\mathbf{d} - \mathbf{C}\boldsymbol{\omega})$;
while $\mathcal{Z} \neq \varnothing$ and $\max_{i \in \mathcal{Z}}(w_i) > \varepsilon_{\text{ecsw}}$ **do**
  $j = \arg\max_{i \in \mathcal{Z}}(w_i)$;
  $\mathcal{Z} = \mathcal{Z} \setminus \{j\}$;
  $\mathcal{P} = \mathcal{P} \cup \{j\}$;
  $\mathbf{s} = \mathbf{0}$;
  Let $\mathbf{C}_{:,\mathcal{P}}$ be the submatrix of $\mathbf{C}$ with columns from $\mathcal{P}$;
  Let $\mathbf{s}_{\mathcal{P}}$ be the subvector of $\mathbf{s}$ with indexes from $\mathcal{P}$;
  Find $\mathbf{s}_{\mathcal{P}}$ minimizing $\|\mathbf{C}_{:,\mathcal{P}}\,\mathbf{s}_{\mathcal{P}} - \mathbf{d}\|_2$ using a QR solver;
  while $\min_{i \in \mathcal{P}}(s_i) \leqslant 0$ **do**
    $\mathcal{Q} = \{i \in \{1, \ldots, N_{\text{dof}}\} \mid s_i \leqslant 0\} \cap \mathcal{P}$;
    $\alpha = \min_{i \in \mathcal{Q}} \left(\frac{\omega_i}{\omega_i - s_i}\right)$;
    $\boldsymbol{\omega} = \boldsymbol{\omega} + \alpha(\mathbf{s} - \boldsymbol{\omega})$;
    $\mathcal{Z} = (\{i \in \{1, \ldots, N_{\text{dof}}\} : |\omega_i| < \varepsilon_{\text{ecsw}}\} \cap \mathcal{P}) \cup \mathcal{Z}$;
    $\mathcal{P} = \{1, \ldots, N_{\text{dof}}\} \setminus \mathcal{Z}$;
    $\mathbf{s} = \mathbf{0}$;
    Find $\mathbf{s}_{\mathcal{P}}$ minimizing $\|\mathbf{C}_{:,\mathcal{P}}\,\mathbf{s}_{\mathcal{P}} - \mathbf{d}\|_2$ using a QR solver;
  end while
  $\boldsymbol{\omega} = \mathbf{s}$;
  $\mathbf{w} = \mathbf{C}^T(\mathbf{d} - \mathbf{C}\boldsymbol{\omega})$;
end while

5. Numerical experiments

In this section, the performance of the proposed approach is evaluated for parametrized, time-dependent, nonlinear problems based on the 1D Burger's equation and 2D compressible Euler equations. In each case, the accuracy of the method with respect to the HDM is evaluated using the relative space-

time error:

$$\text{Error} = \sqrt{\frac{\sum_{k=1}^{N_t} \|\mathbf{u}(t_k;\boldsymbol{\theta}) - \widehat{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2^2}{\sum_{k=1}^{N_t} \|\mathbf{u}(t_k;\boldsymbol{\theta})\|_2^2}},$$

and the computational speedup factor with respect to the HDM is evaluated in order to quantify the reduction in computational cost.

5.1. One-dimensional Burgers' equation

The first application focuses on the prediction of a single hump in 1D. The input parameter $\boldsymbol{\theta} = [\nu, A]$ corresponds to the diffusion coefficient $\nu \in [0.1, 1]$ and a constant $A \in [1, 5]$. The spatial domain is $[-2, 2.5]$, and we consider the time-interval $[10^{-3}, 0.2]$. The fluid velocity $u(x, t; \boldsymbol{\theta}) \in \mathbb{R}$ is governed by the Burgers' equation:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2},$$

where the initial and boundary conditions are supplied by the exact solution

$$u(x, t; \boldsymbol{\theta}) = \sqrt{\frac{\nu}{t}} \left[\frac{\left(e^{\frac{A}{2\nu}} - 1\right) e^{\frac{-x^2}{4\nu t}}}{\sqrt{\pi} + \frac{\sqrt{\pi}}{2}\left(e^{\frac{A}{2\nu}} - 1\right) \text{erfc}(\frac{x}{\sqrt{4\nu t}})} \right]$$

with $\text{erfc}(x)$ the complimentary error function.

The HDM is constructed using a second-order finite difference method. The time-interval is divided into $N_t = 200$ uniform subintervals using the fixed time-step size $\Delta t = 10^{-3}$, and the spatial domain is equally partitioned by 500 interior grid points, leading to $N_{\text{dof}} = 500$ degrees of freedom (DOFs). For constructing the ROM, solution snapshots are collected every time-step at the 9 training parameters shown in Figure 2, leading to a snapshot database of size $N_{\text{s}} = 200 \times 9 = 1,800$. The reduced basis is built using the tolerance $\varepsilon_{\text{pod}} = 10^{-3}$, which yields to $N_{\text{v}} = 36$ basis vectors. The hyper-reduction tolerance is chosen as $\varepsilon_{\text{ecsw}} = 10^{-6}$, resulting in a reduced mesh of size $\|\boldsymbol{\omega}\|_0 = 86$. In Figure 3, we show snapshots of the Mach number solution at final time computed using the HDM.

Figure 4 reports the performance of the hybrid HDM/ROM model for predicting the solution at the 40 queried parameters shown in Figure 2. The error tends to decrease as the tolerance ε_{err} decreases, since the solution is

computed by the HDM for a larger number of time-steps. In particular, the error for $\varepsilon_{\text{err}} = 10^{-1}$ (resp. $\varepsilon_{\text{err}} = 10^{-4}$) is almost the same as that of the full ROM (resp. HDM). Moreover, the runtime tends to increase as the tolerance ε_{err} decreases, since the HDM is used in a larger part of the computations. In particular, the runtime for $\varepsilon_{\text{err}} = 10^{-1}$ (resp. $\varepsilon_{\text{err}} = 10^{-4}$) corresponds approximately to that of the full ROM (resp. ROM+HDM). In the following, we choose the tolerance $\varepsilon_{\text{err}} = 10^{-2}$ because the associated error is less than 1% and the median computational speedup factor is 5.41.

Figures 5 and 6 compare the performance of the hybrid approach and its adaptive-extended version. The error of the adaptive, hybrid model is lower since the underlying ROM is updated with the new snapshots generated by the HDM. Regarding the computational cost, the runtime of the hybrid approach and its adaptive version is about the same, because on the one hand, the cost of the adaptive ROM is higher since the reduced basis dimension and reduced mesh size are larger, but on the other hand, the HDM is solved less frequently.

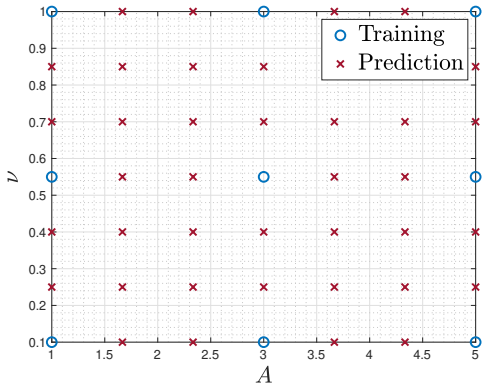
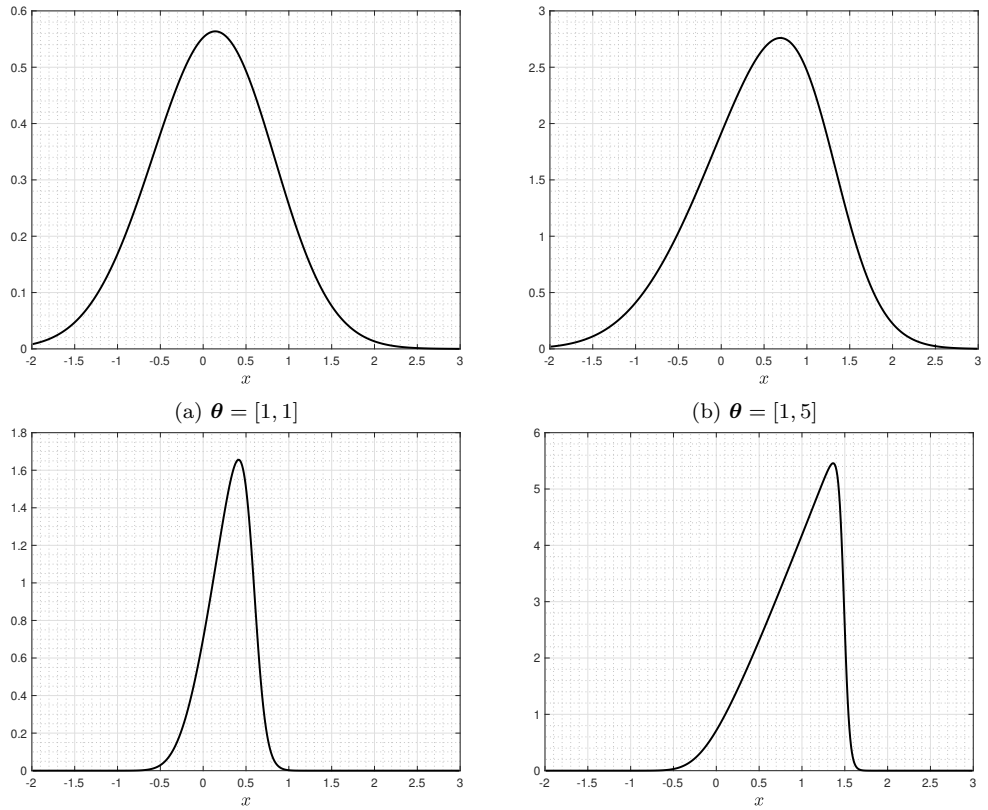


Figure 2: Sampling of the parameter domain $\mathcal{D}$



(a) $\boldsymbol{\theta} = [1, 1]$ (b) $\boldsymbol{\theta} = [1, 5]$

(c) $\boldsymbol{\theta} = [0.1, 1]$ (d) $\boldsymbol{\theta} = [0.1, 5]$

Figure 3: High-dimensional solution at final time for different parameters $\boldsymbol{\theta}$

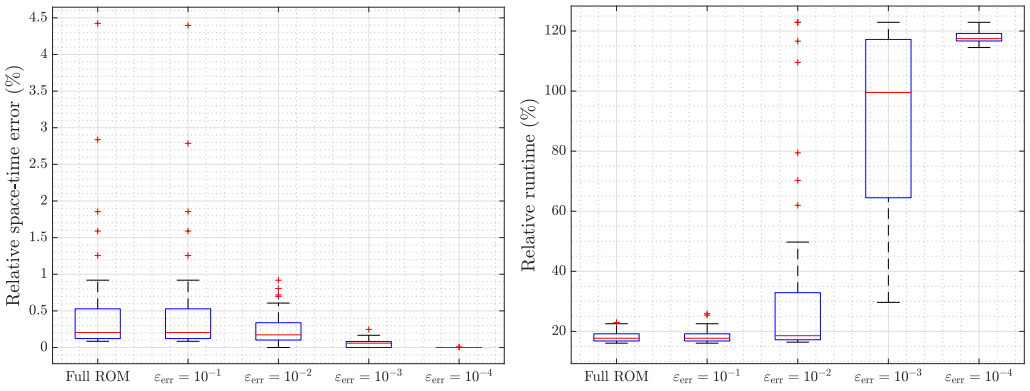


Figure 4: Performance of the hybrid model for different tolerances ε_{err}

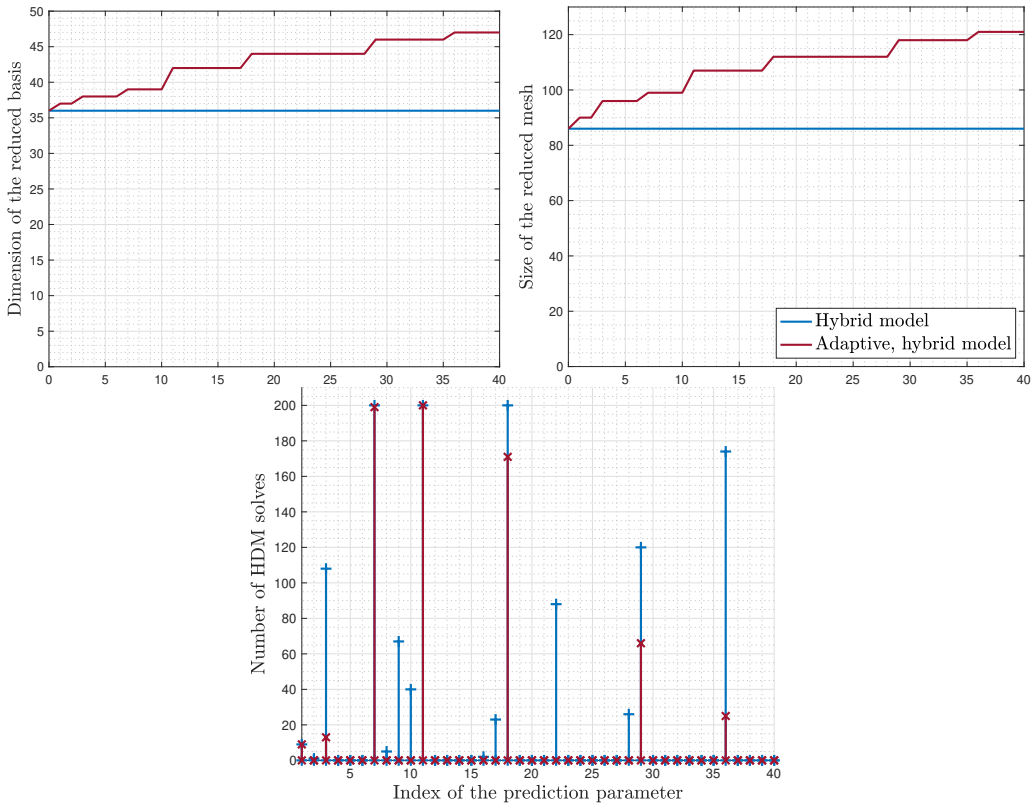


Figure 5: Comparaison of the evolution of the reduced basis, reduced mesh, and number of HDM solves for the hybrid model and its adaptive version. In particular, the prediction parameters are ordered to minimize the distance to the input parameter $\boldsymbol{\theta} = [0.1, 5]$, and in case of equality, from bottom to top and right to left

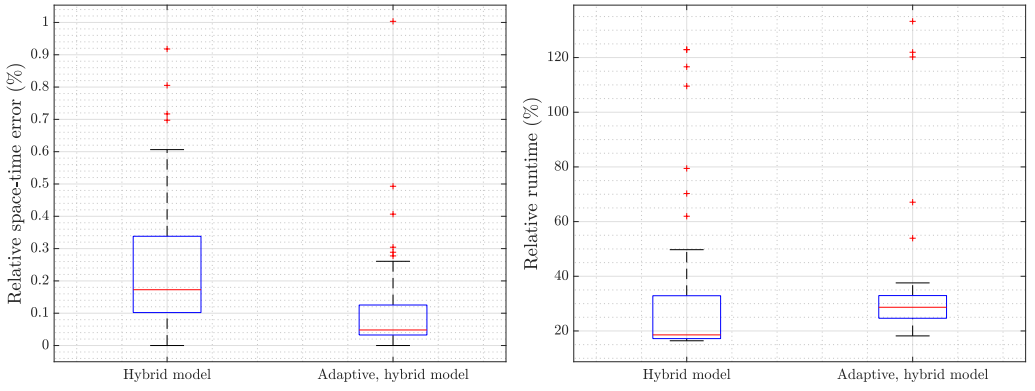


Figure 6: Comparaison of the performance of the hybrid model and its adaptive version

5.2. Two-dimensional compressible Euler equations

The second application focuses on the prediction of subsonic and transonic flows over a NACA 0012 airfoil in 2D. The input parameter $\boldsymbol{\theta} = [M_\infty, \alpha]$ corresponds here to the free-stream Mach number $M_\infty \in [0.3, 0.8]$ and the angle of attack $\alpha \in [0, 8]$. The spatial domain $\Omega \subset \mathbb{R}^2$ is shown in Figure 7, and we consider the time-interval $[0, 10]$. The fluid density $\rho(\boldsymbol{x}, t; \boldsymbol{\theta}) \geq 0$, velocity $\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\theta}) = [u(\boldsymbol{x}, t; \boldsymbol{\theta}), v(\boldsymbol{x}, t; \boldsymbol{\theta})] \in \mathbb{R}^2$, and total energy $E(\boldsymbol{x}, t; \boldsymbol{\theta}) \geq 0$ are governed by the compressible Euler equations:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \dfrac{\partial \rho u}{\partial x} + \dfrac{\partial \rho v}{\partial y} = 0 \\ \dfrac{\partial \rho u}{\partial t} + \dfrac{\partial (\rho u^2 + p)}{\partial x} + \dfrac{\partial \rho uv}{\partial y} = 0 \\ \dfrac{\partial \rho v}{\partial t} + \dfrac{\partial \rho uv}{\partial x} + \dfrac{\partial (\rho v^2 + p)}{\partial y} = 0 \\ \dfrac{\partial E}{\partial t} + \dfrac{\partial u(E + p)}{\partial x} + \dfrac{\partial v(E + p)}{\partial y} = 0, \end{cases}$$

where $p(\boldsymbol{x}, t; \boldsymbol{\theta}) \geq 0$ is the fluid pressure, given by the equation of state

$$p = (\gamma - 1) \left(E - \rho \frac{u^2 + v^2}{2} \right)$$

with γ, the specific heat ratio, taken as $\gamma = 1.4$ in the following. The initial condition is a uniform flow at Mach M_∞:

$$\rho_0 = \gamma, \quad u_0 = M_\infty, \quad v_0 = 0, \quad p_0 = 1.$$

Slip boundary conditions are applied at the airfoil surface, and the far-field boundary condition is set to be a uniform flow at Mach M_∞.

The HDM is constructed using a first-order finite volume method equipped with the local Lax-Friedrichs flux in space and the implicit Euler scheme in time. The time-interval is divided into $N_t = 1,000$ uniform subintervals using the fixed time-step size $\Delta t = 10^{-2}$, and the spatial domain is partitioned by 1,718 vertices and 2,300 triangular cells, leading to $N_{\text{dof}} = 4 \times 2,300 = 9,200$ DOFs. For constructing the ROM, solution snapshots are collected every time-step at the 9 training parameters shown in Figure 2, leading to a snapshot database of size $N_{\text{s}} = 1,000 \times 9 = 9,000$. The reduced basis is built using the tolerance $\varepsilon_{\text{pod}} = 10^{-2}$, which yields to $N_{\text{v}} = 29$ basis vectors. The

hyper-reduction tolerance is chosen as $\varepsilon_{\text{ecsw}} = 10^{-4}$, resulting in a reduced mesh of size $\|\boldsymbol{\omega}\|_0 = 251$. In Figure 9, we show snapshots of the Mach number solution at final time computed using the HDM.

Figure 10 reports the performance of the hybrid HDM/ROM model for predicting the solution at the 40 queried parameters shown in Figure 8. As previously, the error tends to decrease while the runtime tends to increase as the tolerance ε_{err} decreases, since the HDM is solved in a larger part of the computations. In particular, the error for $\varepsilon_{\text{err}} = 10^{-1}$ (resp. $\varepsilon_{\text{err}} = 10^{-3}$) is almost the same as that of the full ROM (resp. HDM), and the runtime for $\varepsilon_{\text{err}} = 10^{-1}$ (resp. $\varepsilon_{\text{err}} = 10^{-4}$) corresponds approximately to that of the full ROM (resp. ROM+HDM). In the following, we choose the tolerance $\varepsilon_{\text{err}} = 10^{-2}$ because the associated error is less than 1% and the median computational speedup factor is 11.95.

In Figures 5 and 6, we compare the performance of the hybrid approach and its adaptive-extended version. The error of the adaptive, hybrid model is lower since the underlying ROM is updated with the new high-dimensional snapshots. The runtime of the hybrid approach and its adaptive version is about the same, because on the one hand, the cost of the adaptive ROM is higher since the reduced basis dimension and reduced mesh size are larger, but on the other hand, the HDM is solved less frequently.

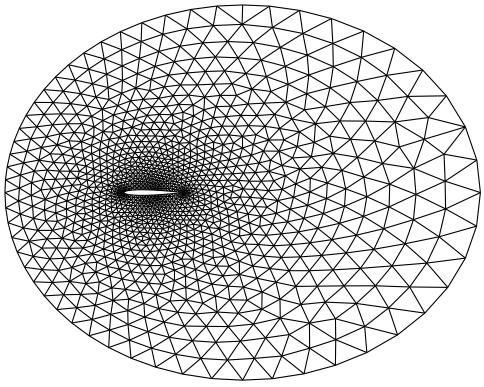


Figure 7: Computational domain

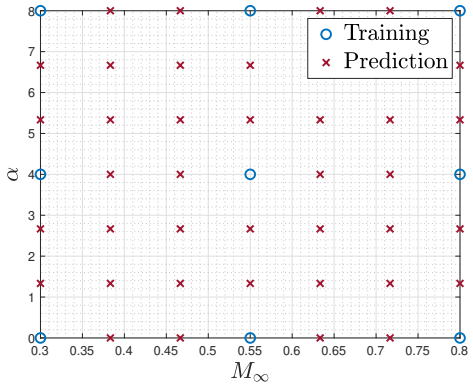
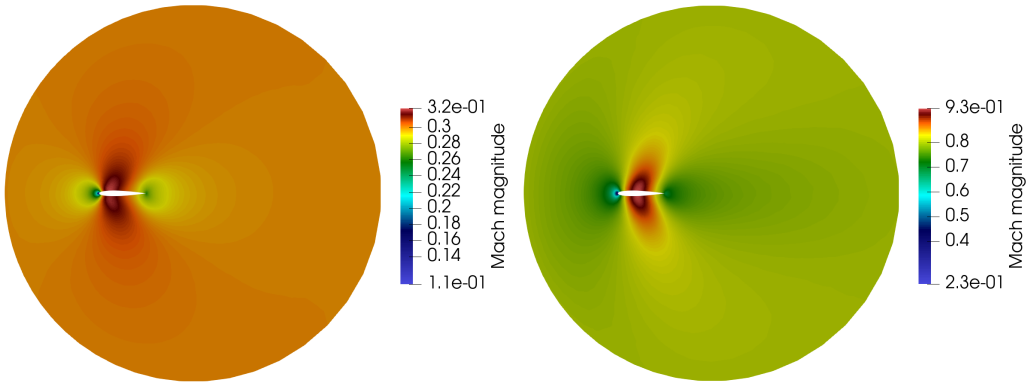
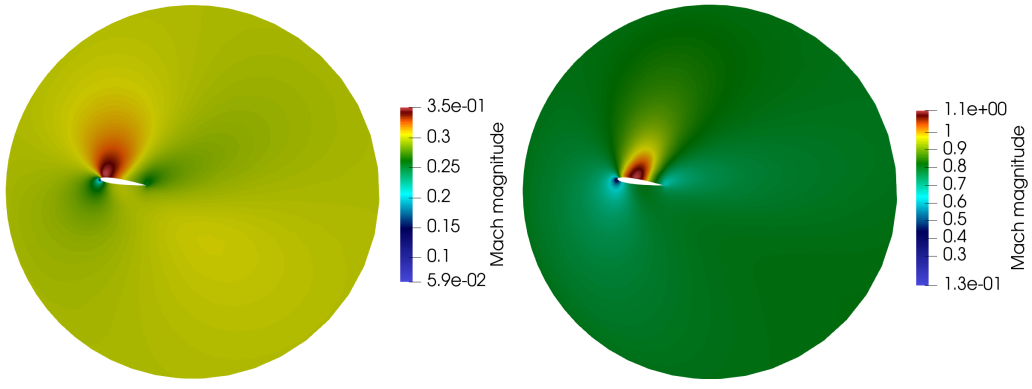


Figure 8: Sampling of the parameter domain $\mathcal{D}$



(a) $\boldsymbol{\theta} = [0.3, 0]$ (b) $\boldsymbol{\theta} = [0.8, 0]$



(c) $\boldsymbol{\theta} = [0.3, 8]$ (d) $\boldsymbol{\theta} = [0.8, 8]$

Figure 9: High-dimensional solution at final time for different parameters $\boldsymbol{\theta}$

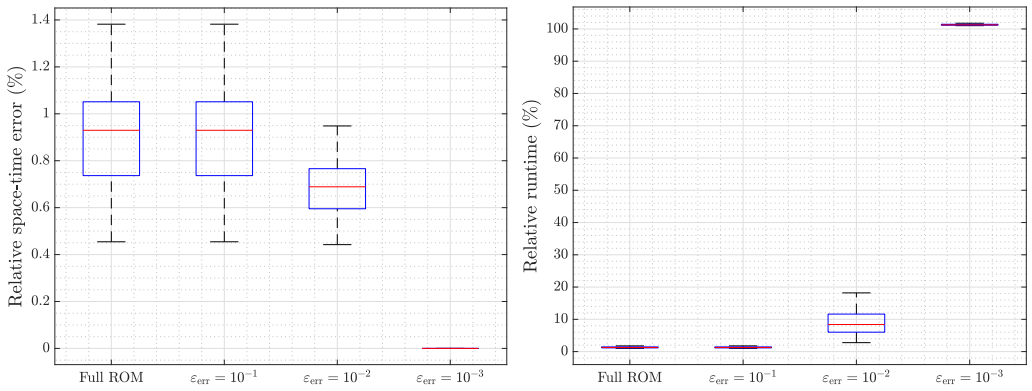


Figure 10: Performance of the hybrid model for different tolerances ε_{err}

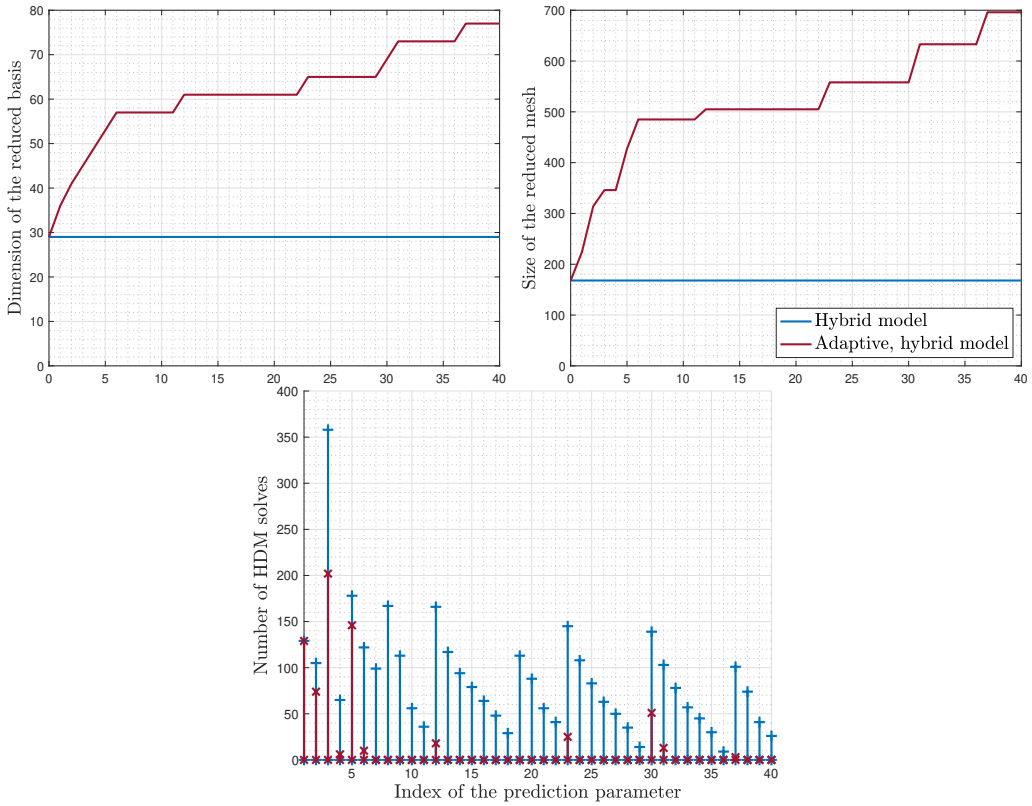


Figure 11: Comparaison of the evolution of the reduced basis, reduced mesh, and number of HDM solves for the hybrid model and its adaptive version. In particular, the prediction parameters are ordered to minimize the distance to the input parameter $\boldsymbol{\theta} = [0.8, 8]$, and in case of equality, from top to bottom and right to left

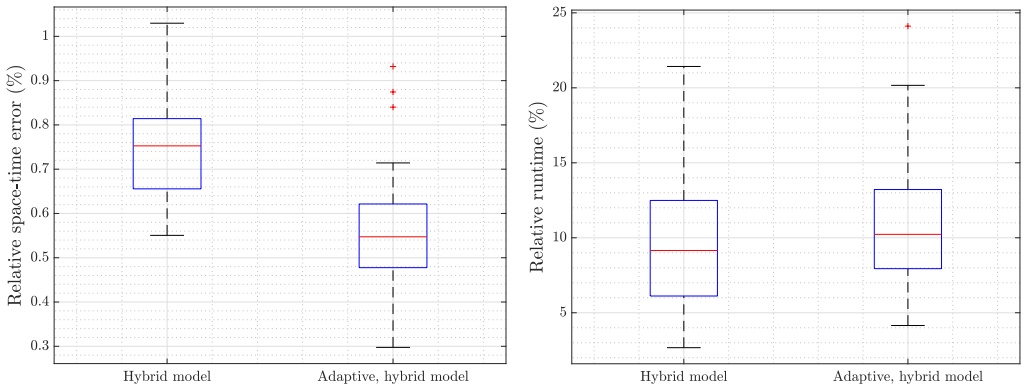


Figure 12: Comparaison of the performance of the hybrid model and its adaptive version

6. Conclusion

In this paper, we have proposed an accurate and efficient approach for the ECSW method based on the principal directions of the reduced-order residual and reduced-order Jacobian matrix. Then, we have adopted a hybrid HDM/ROM approach to accelerate numerical simulations while maintaining accurate approximations. In particular, we have developed a residual-based error indicator to determine when the ROM is not sufficiently accurate and the HDM needs to be solved. Finally, we have proposed an adaptive-extended version of the hybrid model to enrich the ROM with the solution snapshots generated by the HDM.

The performance of the adaptive, hybrid HDM/ROM approach has been evaluated for parametrized, time-dependent, nonlinear problems based on the 1D Burger's equation and 2D compressible Euler equations. The results demonstrated the accuracy and robustness of the proposed method, capable of delivering less than 1% of error, while delivering a significant computational speedup factor. In particular, the adaptive version of the hybrid model allows to further improve the accuracy and robustness of the method for predicting new out-of-sample solutions.

In perspective, the computational efficiency of the proposed method could also be improve. To this end, the computational cost associated with the HDM solution could be reduced by considering hybrid snapshots. The latter are generated by employing the HDM in regions where the ROM is not sufficiently accurate, and space-local ROMs [33, 34, 35] elsewhere. In addition, the performance of the ROM could also be further improved by using

ROMs based on nonlinear manifolds [36, 37, 38]. Finally, another perspective could be to employ the resulting adaptive, hybrid HDM/ROM model to dramatically reduce the computational cost associated with the offline stage of ROMs. These topics will be addressed in future work.

Appendix A.

The objective of this section is to describe the construction of $\mathbf{C} \in \mathbb{R}^{N_\mathrm{c} \times N_\mathrm{dof}}$ and $\mathbf{d} \in \mathbb{R}^{N_\mathrm{c}}$ involved in the NNLS problem (3). Let the training time-parameter instances be $\{(t_1^{\mathrm{train}}, \boldsymbol{\theta}_1^{\mathrm{train}}), \ldots, (t_{N_\mathrm{s}}^{\mathrm{train}}, \boldsymbol{\theta}_{N_\mathrm{s}}^{\mathrm{train}})\}$. We start to present different strategies for constructing $\mathbf{C}$ and consequently $\mathbf{d}$ since $\mathbf{d} = \mathbf{C}\mathbf{1}$.

Residual. The evaluation of the reduced-order residual $\mathbf{W}(t_k;\boldsymbol{\theta})^T \mathbf{r}(\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta}), t_k;\boldsymbol{\theta})$ leads to the approximation problem:

$$\mathbf{C}_1 \boldsymbol{\omega} \approx \mathbf{d}_1, \tag{A.1}$$

where

$$\mathbf{C}_1 = \begin{bmatrix} \mathbf{C}_1^{(1)} \\ \vdots \\ \mathbf{C}_1^{(N_\mathrm{s})} \end{bmatrix} \in \mathbb{R}^{(N_\mathrm{s} N_\mathrm{v}) \times N_\mathrm{dof}} \quad \text{and} \quad \mathbf{C}_1^{(n)} \in \mathbb{R}^{N_\mathrm{v} \times N_\mathrm{dof}}$$

with $\left(\mathbf{C}_1^{(n)}\right)_{i,j} = \left(\mathbf{W}(t_n^{\mathrm{train}};\boldsymbol{\theta}_n^{\mathrm{train}})\right)_{j,i} \left(\mathbf{r}(\widetilde{\mathbf{u}}(t_n^{\mathrm{train}};\boldsymbol{\theta}_n^{\mathrm{train}}), t_n^{\mathrm{train}};\boldsymbol{\theta}_n^{\mathrm{train}})\right)_j$.

Jacobian. In the same way, the evaluation of the reduced-order Jacobian matrix $\mathbf{W}(t_k;\boldsymbol{\theta})^T \mathbf{W}(t_k;\boldsymbol{\theta})$ leads to the approximation problem:

$$\mathbf{C}_2 \boldsymbol{\omega} \approx \mathbf{d}_2, \tag{A.2}$$

where

$$\mathbf{C}_2 = \begin{bmatrix} \mathbf{C}_2^{(1)} \\ \vdots \\ \mathbf{C}_2^{(N_\mathrm{s})} \end{bmatrix} \in \mathbb{R}^{(N_\mathrm{s} N_\mathrm{v}(N_\mathrm{v}+1)/2) \times N_\mathrm{dof}} \quad \text{and} \quad \mathbf{C}_2^{(n)} \in \mathbb{R}^{(N_\mathrm{v}(N_\mathrm{v}+1)/2) \times N_\mathrm{dof}}$$

with $\left(\mathbf{C}_2^{(n)}\right)_{i_1+i_2(i_2-1)/2,j} = \left(\mathbf{W}(t_n^{\mathrm{train}};\boldsymbol{\theta}_n^{\mathrm{train}})\right)_{j,i_1} \left(\mathbf{W}(t_n^{\mathrm{train}};\boldsymbol{\theta}_n^{\mathrm{train}})\right)_{j,i_2}$ for $1 \leq i_1 \leq i_2 \leq N_\mathrm{v}$. Note that the symmetry of the problem has been exploited to remove the duplicate rows of $\mathbf{C}_2$.

Residual+Jacobian. Combining the contributions of (A.1) and (A.2) leads to the approximation problem:

$$\mathbf{C}_3\boldsymbol{\omega} \approx \mathbf{d}_3, \tag{A.3}$$

where

$$\mathbf{C}_3 = \begin{bmatrix}\mathbf{C}_1\\ \mathbf{C}_2\end{bmatrix} \in \mathbb{R}^{(N_\mathrm{s}N_\mathrm{v}(N_\mathrm{v}+3)/2)\times N_\mathrm{dof}}.$$

Principal directions. In practice, the number of rows of $\mathbf{C}_3$ can rapidly become very large, which is problematic for solving the NNLS problem (3). In order to drastically reduce the number of rows, we replace $\mathbf{C}_3$ by its principal directions. Let $\mathbf{C}_3 \approx \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ be the truncated SVD of $\mathbf{C}_3$, where $\mathbf{U} \in \mathbb{R}^{(N_\mathrm{s}N_\mathrm{v}(N_\mathrm{v}+3)/2)\times N_\mathrm{c}}$ is unitary, $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_{N_\mathrm{c}}) \in \mathbb{R}^{N_\mathrm{c}\times N_\mathrm{c}}$ is diagonal, $\mathbf{V} \in \mathbb{R}^{N_\mathrm{dof}\times N_\mathrm{c}}$ is unitary, and the rank N_c is defined as the smallest integer such that the relative approximation error is less than the tolerance ε_pod (i.e., $\left\|\mathbf{C}_3 - \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\right\|_F \leqslant \varepsilon_\mathrm{pod}\left\|\mathbf{C}_3\right\|_F$). The approximation problem (A.3) becomes

$$\mathbf{C}_4\boldsymbol{\omega} \approx \mathbf{d}_4, \tag{A.4}$$

where

$$\mathbf{C}_4 = \mathbf{V}^T \in \mathbb{R}^{N_\mathrm{c}\times N_\mathrm{dof}} \quad \text{and} \quad N_\mathrm{c} \ll N_\mathrm{s}N_\mathrm{v}(N_\mathrm{v}+3)/2.$$

Weighted principal directions. In the approximation problem (A.4), the principal directions contribute equally to the approximation error, which may not be representative of the original approximation problem (A.3). For this reason, we introduce a weighting of the principal directions to approximately recover the original problem (A.3):

$$\begin{aligned}\left\|\mathbf{C}_3\boldsymbol{\omega} - \mathbf{d}_3\right\|_2 &\approx \left\|\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\boldsymbol{\omega} - \mathbf{d}_3\right\|_2\\ &= \left\|\mathbf{U}\boldsymbol{\Sigma}(\mathbf{C}_4\boldsymbol{\omega} - \mathbf{d}_4)\right\|_2\\ &= \left\|\boldsymbol{\Sigma}(\mathbf{C}_4\boldsymbol{\omega} - \mathbf{d}_4)\right\|_2.\end{aligned}$$

The approximation problem (A.4) therefore becomes

$$\mathbf{C}_5\boldsymbol{\omega} \approx \mathbf{d}_5, \tag{A.5}$$

where

$$\mathbf{C}_5 = \boldsymbol{\Sigma}\mathbf{C}_4 \in \mathbb{R}^{N_\mathrm{c}\times N_\mathrm{dof}}.$$

Remark 2. *Note that the solution $\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})$ is replaced in practice by the projected snapshot $\mathbf{u}_\mathrm{off}(\boldsymbol{\theta}) + \mathbf{V}\mathbf{V}^T(\mathbf{u}(t_k;\boldsymbol{\theta}) - \mathbf{u}_\mathrm{off}(\boldsymbol{\theta}))$ in order to save the time associated with the evaluation of the reduced-order solution.*

Numerical results. The different strategies for constructing $\mathbf{C}$ are now evaluated for the two applications presented in Section 5. In each case, we report the size of $\mathbf{C}$ involved in the NNLS problem (3). Moreover, Figures A.13 and A.14 show the runtime for computing the weights $\boldsymbol{\omega}$ and the accuracy of the associated ROM, as a function of the reduced mesh size $\|\boldsymbol{\omega}\|_0$.

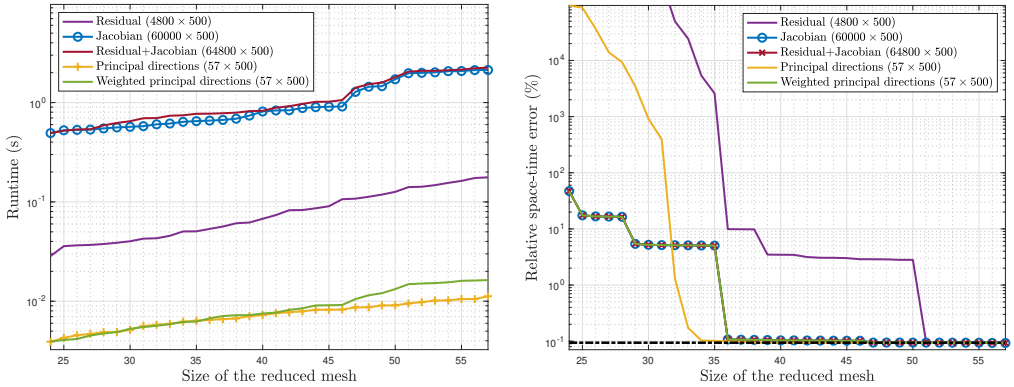


Figure A.13: Results for the application presented in Section 5.1. We consider the reproduction of the solution corresponding to the input parameter $\boldsymbol{\theta} = [0.1, 5]$. Solution snapshots are collected every time-step, leading to $N_s = 200$ snapshots. The reduced basis is built using the tolerance $\varepsilon_{\text{pod}} = 10^{-3}$, which yields to $N_{\text{v}} = 24$ basis vectors.

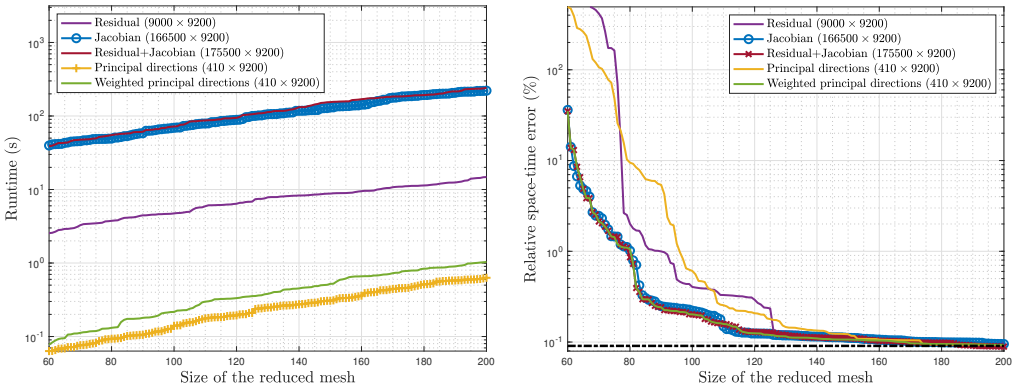


Figure A.14: Results for the application presented in Section 5.2. We consider the reproduction of the solution corresponding to the input parameter $\boldsymbol{\theta} = [0.8, 8]$. Solution snapshots are collected every 4 time-steps, leading to $N_s = 250$ snapshots. The reduced basis is built using the tolerance $\varepsilon_{\text{pod}} = 10^{-3}$, which yields to $N_{\text{v}} = 36$ basis vectors.

The runtime of the "principal directions" and "weighted principal directions" approaches are almost the same and are significantly lower than that

of the other approaches, as the number of rows of the corresponding matrix $\mathbf{C}$ is drastically smaller. In addition, the error of the "jacobian", "residual+jacobian", and "weighted principal direction" approaches are about the same and are generally lower that that of the other approaches, since the "residual+jacobian" approach contains all the available information for computing the weights $\boldsymbol{\omega}$. For these reasons, the "weighted principal direction" approach is used in this work as it provides the most accurate ROM, while drastically reducing the runtime to solve the NNLS problem (3).

Appendix B.

The objective of this section is to describe the interpolation used in the error indicator $\delta_{\text{err}}(t_k;\boldsymbol{\theta})$. Let the training parameters corresponding to the time-instance t_k be $\{\boldsymbol{\theta}_n^{\text{train}}\}_{n\in\Theta_k} = \{\boldsymbol{\theta}_1^{\text{train},k},\dots,\boldsymbol{\theta}_{N_{\text{s},k}}^{\text{train},k}\}$. The RBF interpolation based on polyharmonic splines is then defined as

$$\mathcal{I}\left(t_k,\boldsymbol{\theta};\{\boldsymbol{\theta}_n^{\text{train}}\}_{n\in\Theta_k},\{\lambda_i^*(t_k,\boldsymbol{\theta}_n^{\text{train}})\}_{n\in\Theta_k}\right) = \sum_{m=1}^{N_{\text{s},k}} w_m \left\|\boldsymbol{\theta}-\boldsymbol{\theta}_m^{\text{train},k}\right\|_2 + \mathbf{v}^T\begin{bmatrix}1\\ \boldsymbol{\theta}\end{bmatrix},$$

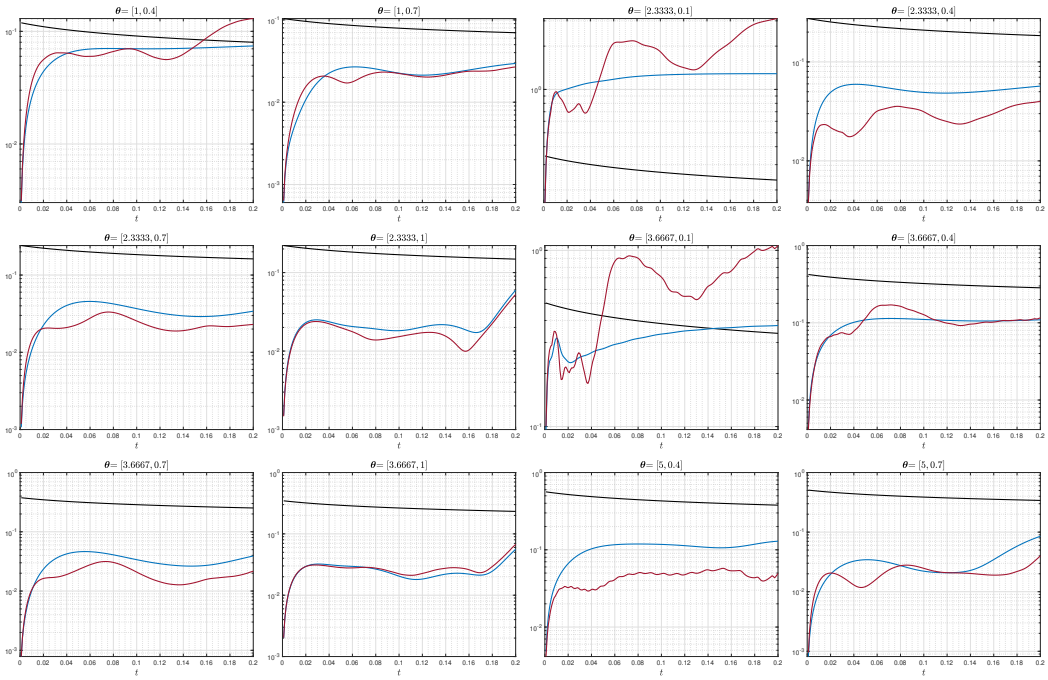
where $\mathbf{w}=[w_1,\dots,w_{N_{\text{s},k}}]\in\mathbb{R}^{N_{\text{s},k}}$ and $\mathbf{v}\in\mathbb{R}^{N_p+1}$ are solutions to the linear system

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{B}^T & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{w}\\ \mathbf{v}\end{bmatrix} = \begin{bmatrix}\mathbf{f}\\ \mathbf{0}\end{bmatrix}$$

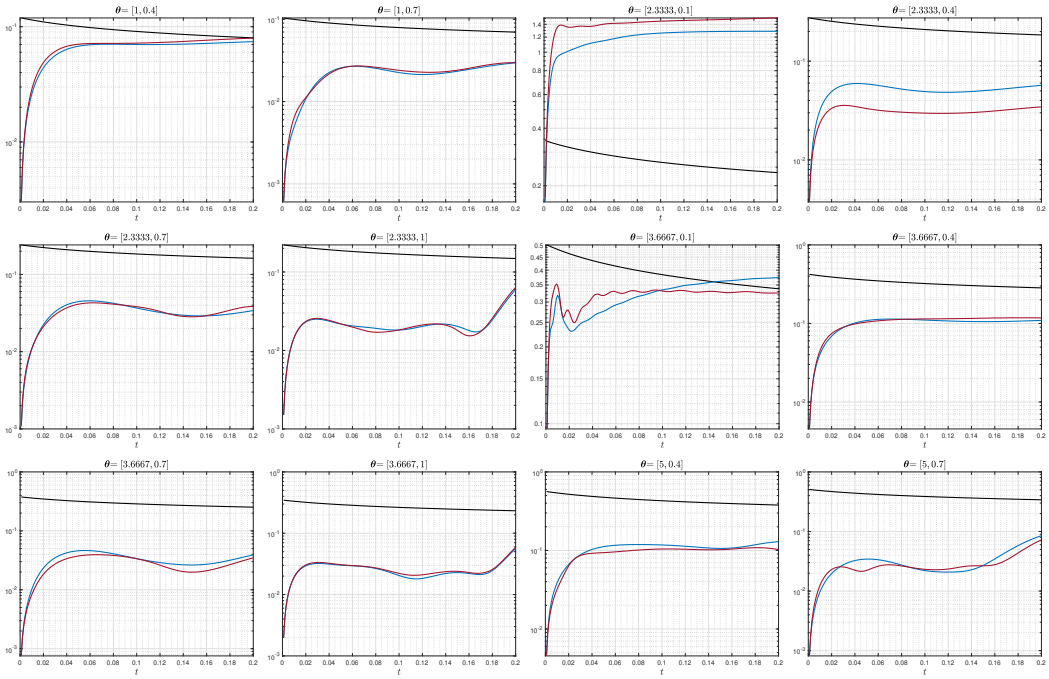
with

$$A_{n,m} = \left\|\boldsymbol{\theta}_n^{\text{train},k}-\boldsymbol{\theta}_m^{\text{train},k}\right\|_2,\ \mathbf{B}^T = \begin{bmatrix}1 & \dots & 1\\ \boldsymbol{\theta}_1^{\text{train},k} & \dots & \boldsymbol{\theta}_{N_{\text{s},k}}^{\text{train},k}\end{bmatrix},\ \mathbf{f} = \begin{bmatrix}\lambda_i^*(t_k,\boldsymbol{\theta}_1^{\text{train},k})\\ \vdots\\ \lambda_i^*(t_k,\boldsymbol{\theta}_{N_{\text{s},k}}^{\text{train},k})\end{bmatrix}.$$

Numerical examples. Figures B.15 and B.16 show the proposed error estimation for the two applications presented in Section 5. In each case, we compare the interpolation resulting from a coarse and fine training grid. It can be observed that the error indicator $\delta_{\text{err}}(t_k;\boldsymbol{\theta})$ becomes more accurate to approximate the error norm $\|\mathbf{u}(t_k;\boldsymbol{\theta})-\widetilde{\mathbf{u}}(t_k;\boldsymbol{\theta})\|_2$ as the number of training parameters $\boldsymbol{\theta}_n^{\text{train}}$ increases.

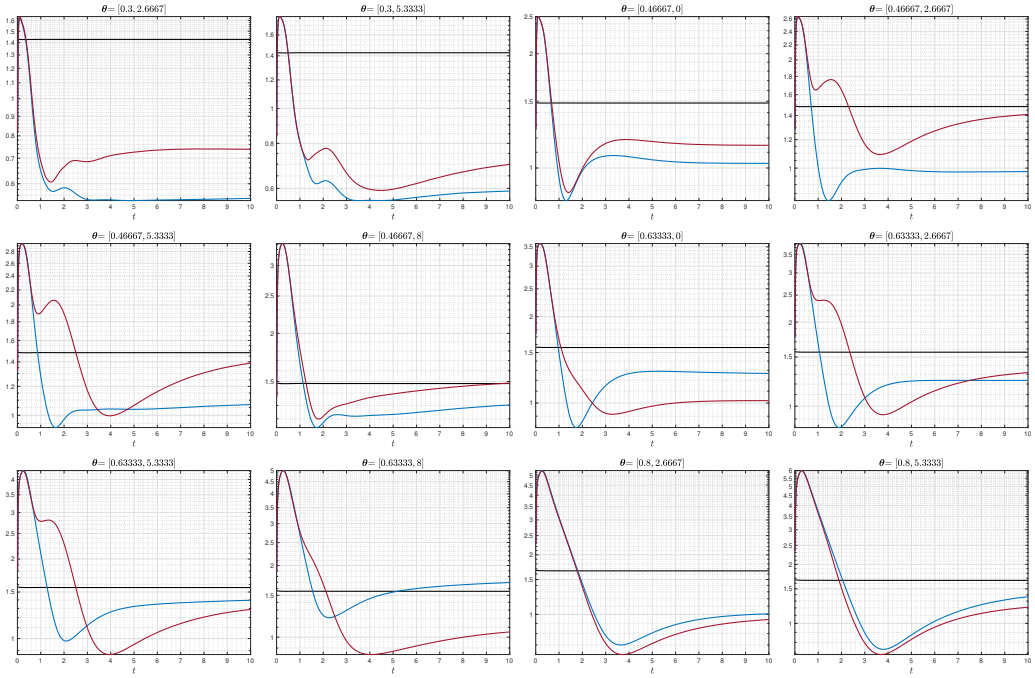


(a) Coarse 3×3 uniform grid for sampling the training parameters $\boldsymbol{\theta}_n^{\text{train}}$

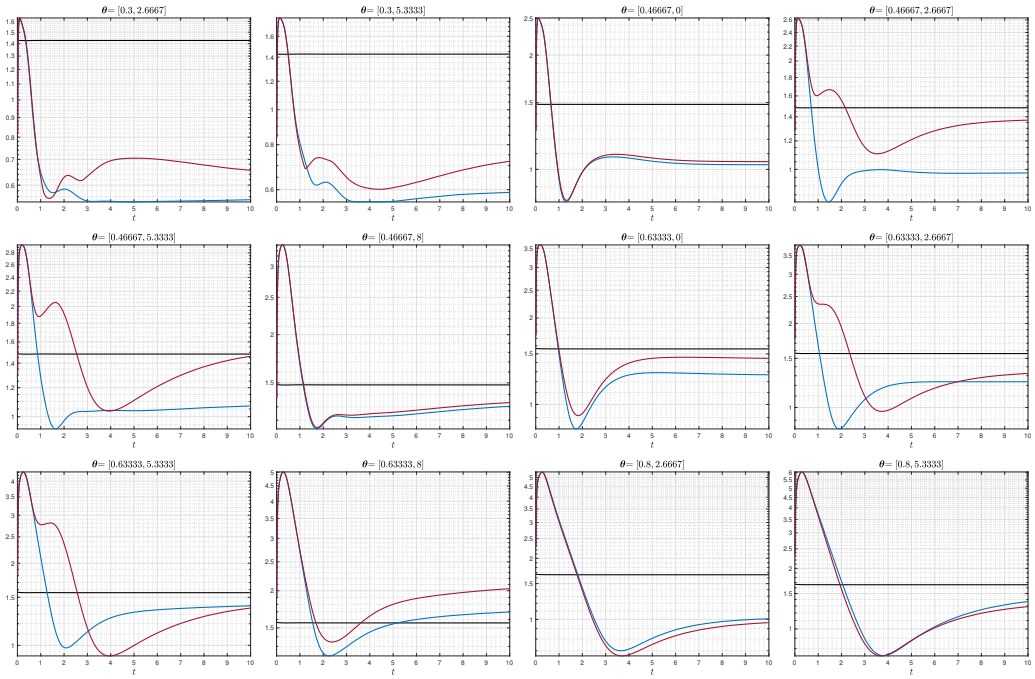


(b) Fine 6×6 uniform grid for sampling the training parameters $\boldsymbol{\theta}_n^{\text{train}}$

Figure B.15: Comparaison of the error norm in blue and the error indicator in red for the application presented in Section 5.1. The error tolerance criterion $\frac{\varepsilon_{\text{err}}}{1-\varepsilon_{\text{err}}} \left\| \widetilde{\mathbf{u}}(t_k; \boldsymbol{\theta}) \right\|_2$ is plotted in black.



(a) Coarse 3×3 uniform grid for sampling the training parameters $\boldsymbol{\theta}_n^{\text{train}}$



(b) Fine 6×6 uniform grid for sampling the training parameters $\boldsymbol{\theta}_n^{\text{train}}$

Figure B.16: Comparaison of the error norm in blue and the error indicator in red for the application presented in Section 5.2 The error tolerance criterion $\frac{\varepsilon_{\text{err}}}{1-\varepsilon_{\text{err}}} \left\| \widetilde{\mathbf{u}}(t_k; \boldsymbol{\theta}) \right\|_2$ is plotted in black.

References

[1] B. Moore, Principal component analysis in linear systems: Controllability, observability, and model reduction, IEEE transactions on automatic control 26 (1) (1981) 17–32.

[2] G. Berkooz, P. Holmes, J. L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Annual review of fluid mechanics 25 (1) (1993) 539–575.

[3] K. Willcox, J. Peraire, Balanced model reduction via the proper orthogonal decomposition, AIAA journal 40 (11) (2002) 2323–2330.

[4] C. W. Rowley, T. Colonius, R. M. Murray, Model reduction for compressible flows using pod and galerkin projection, Physica D: Nonlinear Phenomena 189 (1-2) (2004) 115–129.

[5] N. Ngoc Cuong, K. Veroy, A. T. Patera, Certified real-time solution of parametrized partial differential equations, Handbook of Materials Modeling: Methods (2005) 1529–1564.

[6] M. Bergmann, C.-H. Bruneau, A. Iollo, Enablers for robust pod models, Journal of Computational Physics 228 (2) (2009) 516–538.

[7] K. Carlberg, C. Farhat, J. Cortial, D. Amsallem, The gnat method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows, Journal of Computational Physics 242 (2013) 623–647.

[8] A. Iollo, D. Lombardi, Advection modes by optimal mass transfer, Physical Review E 89 (2) (2014) 022923.

[9] F. Bernard, A. Iollo, S. Riffaud, Reduced-order model for the bgk equation based on pod and optimal transport, Journal of Computational Physics 373 (2018) 545–570.

[10] L. Sirovich, Turbulence and the dynamics of coherent structures. i. coherent structures, Quarterly of applied mathematics 45 (3) (1987) 561–571.

[11] B. Peherstorfer, Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling, SIAM Journal on Scientific Computing 42 (5) (2020) A2803–A2836.

[12] B. Peherstorfer, K. Willcox, Online adaptive model reduction for nonlinear systems via low-rank updates, SIAM Journal on Scientific Computing 37 (4) (2015) A2123–A2150.

[13] F. Bai, Y. Wang, Deim reduced order model constructed by hybrid snapshot simulation, SN Applied Sciences 2 (12) (2020) 2165.

[14] F. Bai, Y. Wang, Reduced-order modeling based on hybrid snapshot simulation, International Journal of Computational Methods 18 (01) (2021) 2050029.

[15] L. Feng, G. Fu, Z. Wang, A fom/rom hybrid approach for accelerating numerical simulations, Journal of Scientific Computing 89 (2021) 1–16.

[16] V. Zucatti, M. J. Zahr, An adaptive, training-free reduced-order model for convection-dominated problems based on hybrid snapshots, arXiv preprint arXiv:2301.01718 (2023).

[17] K. Carlberg, Adaptive h-refinement for reduced-order models, International Journal for Numerical Methods in Engineering 102 (5) (2015) 1192–1210.

[18] P. A. Etter, K. T. Carlberg, Online adaptive basis refinement and compression for reduced-order models via vector-space sieving, Computer Methods in Applied Mechanics and Engineering 364 (2020) 112931.

[19] T. Bui-Thanh, K. Willcox, O. Ghattas, Model reduction for large-scale systems with high-dimensional parametric input space, SIAM Journal on Scientific Computing 30 (6) (2008) 3270–3288.

[20] K. Carlberg, C. Bou-Mosleh, C. Farhat, Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations, International Journal for numerical methods in engineering 86 (2) (2011) 155–181.

[21] C. Farhat, P. Avery, T. Chapman, J. Cortial, Dimensional reduction of nonlinear finite element dynamic models with finite rotations and

energy-based mesh sampling and weighting for computational efficiency, International Journal for Numerical Methods in Engineering 98 (9) (2014) 625–662.

[22] C. Farhat, T. Chapman, P. Avery, Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models, International journal for numerical methods in engineering 102 (5) (2015) 1077–1110.

[23] R. Tezaur, F. As'ad, C. Farhat, Robust and globally efficient reduction of parametric, highly nonlinear computational models and real time online performance, Computer Methods in Applied Mechanics and Engineering 399 (2022) 115392.

[24] M. Barrault, Y. Maday, N. C. Nguyen, A. T. Patera, An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations, Comptes Rendus Mathematique 339 (9) (2004) 667–672.

[25] S. Chaturantabut, D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM Journal on Scientific Computing 32 (5) (2010) 2737–2764.

[26] C. L. Lawson, R. J. Hanson, Solving least squares problems, SIAM, 1995.

[27] C. Prud'Homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, G. Turinici, Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods, J. Fluids Eng. 124 (1) (2002) 70–80.

[28] M. A. Grepl, A. T. Patera, A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations, ESAIM: Mathematical Modelling and Numerical Analysis 39 (1) (2005) 157–181.

[29] 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, Archives of Computational Methods in Engineering 15 (3) (2008) 229–275.

[30] Y. Zhang, L. Feng, S. Li, P. Benner, An efficient output error estimation for model order reduction of parametrized evolution equations, SIAM Journal on Scientific Computing 37 (6) (2015) B910–B936.

[31] B. A. Freno, K. T. Carlberg, Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations, Computer Methods in Applied Mechanics and Engineering 348 (2019) 250–296.

[32] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.

[33] S. Riffaud, M. Bergmann, C. Farhat, S. Grimberg, A. Iollo, The dgdd method for reduced-order modeling of conservation laws, Journal of Computational Physics 437 (2021) 110336.

[34] S. Anderson, C. White, C. Farhat, Space-local reduced-order bases for accelerating reduced-order models through sparsity, International Journal for Numerical Methods in Engineering 124 (7) (2023) 1646–1671.

[35] A. Iollo, G. Sambataro, T. Taddei, A one-shot overlapping schwarz method for component-based model reduction: application to nonlinear elasticity, Computer Methods in Applied Mechanics and Engineering 404 (2023) 115786.

[36] K. Lee, K. T. Carlberg, Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders, Journal of Computational Physics 404 (2020) 108973.

[37] S. Fresca, A. Manzoni, Pod-dl-rom: Enhancing deep learning-based reduced order models for nonlinear parametrized pdes by proper orthogonal decomposition, Computer Methods in Applied Mechanics and Engineering 388 (2022) 114181.

[38] J. Barnett, C. Farhat, Y. Maday, Neural-network-augmented projection-based model order reduction for mitigating the kolmogorov barrier to reducibility, Journal of Computational Physics 492 (2023) 112420.