Reduced Order Modeling

Zulkeefal Dar, Joan Baiges and Ramon Codina

Zulkeefal Dar
International Center for Numerical Methods in Engineering, e-mail: zulkeefal.dar@upc.edu

Joan Baiges
Universitat Politècnica de Catalunya e-mail: joan.baiges@upc.edu

Ramon Codina
Universitat Politècnica de Catalunya e-mail: ramon.codina@upc.edu

Contents

List of Acronyms

AE	AutoEncoder
AMR	Adaptive Mesh-Refinement
ANN	Artificial Neural Network
BiLSTM	Bidirectional Long Short-Term Memory
DEIM	Discrete Empirical Interpolation Method
DMD	Dynamic Mode Decomposition
FE	Finite Element
FNN	Feedforward Neural Network
FOM	Full Order Model
LSTM	Long Short-Term Memory
ML	Machine Learning
NIROM	Non-Intrusive Reduced Order Model
NN	Neural Network
PDE	Partial Differential Equation
PG	Petrov-Galerkin
POD	Proper Orthogonal Decomposition
POD-G	Proper Orthogonal Decomposition based Galerkin projection
PROM	Parametric Reduced Order Model
RIC	Relative Information Content
ROM	Reduced Order Model
SGS	Sub-Grid Scales
SINDy	Sparse Identification of Nonlinear Dynamics
SUPG	Streamline-Upwind Petrov–Galerkin
SVD	Singular Value Decomposition
VMS	Variational MultiScale

Abstract This chapter presents an overview of the most popular reduced order models found in the approximation of partial differential equations and their connection with machine learning techniques. Although the presentation is applicable to many problems in science and engineering, the focus is first order evolution problems in time and, more specifically, flow problems. Particular emphasis is put in the distinction between intrusive models, that make use of the physical problem being modeled, and non-intrusive models, purely designed from data using machine learning strategies. For the former, models based on proper orthogonal decomposition and Galerkin projection are described in detail, whereas alternatives are only mentioned. Likewise, some modifications that are crucial in the applications are detailed. The progressive incorporation of machine learning methods is described, yielding first hybrid formulations and ending with pure data-driven approaches. An effort has been made to include references with applications of the methods being described.

1 Introduction

Partial Differential Equations (PDEs) provide a mathematical model to represent the many processes occurring in nature of significant importance. Hence, solving these PDEs is of prime interest to researchers and engineers. Many high-fidelity solution techniques have been developed which can solve the PDEs with the desired accuracy. However, these methods are very computationally expensive in general. The computational expense can become particularly prohibitive in the case of optimization problems as they require a large number of simulations to be performed. Moreover, a control problem might require solutions to PDEs in real-time, which is seldom achievable given the high computational cost associated with solving PDEs. *Reduced order models* (ROMs) offer an alternative approach to the high fidelity models to obtain solutions with reasonable accuracy and at a reduced computational cost. ROMs reduce the computational cost by approximating the large-scale systems by much smaller ones.

Broadly speaking, any model which reduces the computational expense can be regarded as a ROM. In the context of finite difference, finite element, or finite volume discretizations, a ROM could mean using a coarser discretizing mesh [18, 60]. Similarly, using a larger time step for a time integration scheme could also imply a ROM [60]. Furthermore, using simplified physics could be considered a ROM. However, in the scientific community, ROMs are understood to represent a particular class of model order reduction, called *projection based* methods (see Remark 1). These methods involve finding a latent *low-dimensional space* to represent the actual *full order model* (FOM) dynamics. The motivation of such methods relies on the observation that even nonlinear dynamical systems can exhibit patterns that can be used to characterize their behavior. Unsurprisingly the origin of these methods can be traced back to identifying and studying coherent structures in turbulent flows in fluid mechanics [83]. These methods are perhaps still the most commonly used in flow problems, and hence, most of the explanations and literature in this chapter will refer to flow problems.

Projection based ROMs consist of two steps:

1. *Offline step*: Finding the low-dimensional representation of the FOM
2. *Online step*: Solving for the unknowns in the reduced ordered space

where the offline step is very computationally intensive but needs to be performed only once or a few times. Once the offline step has been performed, the low-cost online step of the ROM can be performed several times to solve an optimization problem or a real-time control problem, thus providing large computational savings and real-time solutions.

The implementation strategy of the online step of ROMs is used to classify them into *intrusive* and *non-intrusive* ROMs. The intrusive ROMs are *physics* based in the sense that they require the *governing equations*, and possible access to the computational code, to project the FOM onto a reduced order space to solve for the unknowns. Non-intrusive ROMs, on the other hand, are purely *data-driven* and do not require access to governing equations or the computational code.

In the current data-centric era, Machine Learning (ML) has emerged as a viable tool for reduced order modeling. ML has revolutionized a wide range of fields over the past few decades. Scientific computing, and in particular reduced order modeling, is no exception. Although the interest in exploring ML techniques for reduced order modeling is relatively new, it has already shown great potential by replacing in part, or entirely, the offline and online steps.

A variety of *conventional* (see Remark 2) and *ML-based* techniques have been developed for offline and online phases till date and applied successfully in a variety of contexts, e.g., solid mechanics [53, 147], material science [80], fluid mechanics [15, 16, 32, 64, 68, 87, 98, 114, 130, 141], shape optimization [5, 25, 94, 93, 121] and flow control [11, 70, 103, 111] problems. The *proper orthogonal decomposition based Galerkin projection* (POD-G) method can be considered to be the most well-established and commonly used method for reduced order modeling. This method uses proper orthogonal decomposition (POD) to find the *basis*, called *POD modes*, of the low-dimensional space. The FOM is then projected in an *intrusive* manner onto these POD modes using mostly *Galerkin projection* to solve for the unknowns.

The organization of the chapter is as follows. Section 2 describes POD along with its essential ingredient, the *singular value decomposition* (SVD). Section 3 describes the Galerkin projection, hyperreduction, and stabilization of POD-ROMs.Section4 describes the non-intrusive ROMs with a brief explanation of *dynamic mode decomposition* (DMD).Section 5 deals with the description of parametric ROMs. Finally,Section 6 describes the ML techniques used for the online and offline phases of reduced order modeling.

Remark 1 In literature, the term *reduced basis* is also sometimes used for *projection based* methods. However, more commonly reduced basis methods are meant to refer to a particular class of projection based methods based on greedy algorithms[1]. This later usage is also applicable in the context of this chapter.

Remark 2 All data-driven dimension reduction techniques can be classified as ML techniques in the broader sense. For example, POD and DMD can be classified as unsupervised ML techniques. However, these techniques were originally developed for dynamical systems based on mathematical arguments. Therefore, such techniques are referred to as *conventional* and we do not group them under the umbrella of ML techniques.

2 Proper Orthogonal Decomposition

The basis commonly used, e.g., piecewise-linear basis in the finite element (FE) method, Fourier modes, etc., can solve a large number of dynamical systems, but

[1] Greedy algorithms are a class of algorithm which are based on choosing the option which produces the largest immediate reward with the expectation that the successive application of greedy sampling will lead to a global optimum. Greedy algorithms may use an error estimator to guide the sampling.

these basis are generic and do not correspond to the *intrinsic* modes of the systems which they solve. Hence, a large number of such basis functions need to be used to capture the solution. The intrinsic modes which form the basis of the solution space can be found using proper generalized decomposition, reduced basis method or proper orthogonal decomposition (POD), among others. Proper generalized decomposition and reduced basis methods are commonly based on a greedy approach and a comprehensive review on them can be found in [42, 43] and [76], respectively. POD [37] is perhaps the most commonly used method to find the basis, called *POD modes* in the context of POD.

For clarity, let us first introduce the concept of *function* and *vector* based description of a variable in the context of numerical methods. Suppose that the analysis domain is spatially discretized using an *interpolation based method* like finite elements, finite volumes, spectral elements, etc. The variable of interest can then either be represented in the *vectorial* form as the collection of coefficients that multiply the basis functions, or in a corresponding *functional* form which relies on the interpolation to define the variable over the entire domain. Throughout this chapter, the functional representation is denoted using the lowercase letters, like $\boldsymbol{q}$, and the vectorial representation using the uppercase letters, like $\boldsymbol{Q}$. In the case of Greek alphabets, where the case of alphabets is not obvious, functional representation is denoted by showing the dependence on the spatial coordinates $\boldsymbol{x}$ explicitly, like $\boldsymbol{\zeta}(\boldsymbol{x})$. Also, a variable with underbar $\underline{\boldsymbol{a}}$ represents the variable in a general form which could be either functional or vectorial based on the context. In the case of the FE method, the vectorial and functional representations of a variable $\underline{\boldsymbol{u}}$ are related as

$$\boldsymbol{u}(\boldsymbol{x},t) = \sum_{k=1}^{nn} \boldsymbol{\chi}^k(\boldsymbol{x}) U^k(t)$$

where $\boldsymbol{u}(\boldsymbol{x},t)$ is the *functional* representation which depends on the spatial coordinates $\boldsymbol{x}$ in addition to time t, U^k the $k-$th element of the *vector* $\boldsymbol{U}$, $\boldsymbol{\chi}^k(\boldsymbol{x})$ the FE interpolation function for the $k-$th node and nn the total number of nodes.

Now, POD consists of representing a variable $\underline{\boldsymbol{u}}$ as a linear combination of the POD modes as

$$\underline{\boldsymbol{u}}(t,\mu) = \sum_{k=1}^{nb} \underline{\boldsymbol{\Psi}}^k(\mu) U_r^k(t,\mu) \tag{1}$$

where $\boldsymbol{\Psi}^k$ is the k-th POD mode, U_r^k the k-th *ROM coefficient*, nb the number of basis vectors and μ a parameter that characterizes the behavior of the system. $\underline{\boldsymbol{u}}$ and $\underline{\boldsymbol{\Psi}}$ could be the functional or vectorial representation, as required, but the same representation must be used for both variables.

POD relies on a *singular value decomposition* (SVD) to find the basis. Let us describe how the basis is determined using the SVD of the data generated using PDEs.

2.1 Proper Orthogonal Decomposition Applied to Partial Differential Equations

Let us consider a general unsteady nonlinear PDE describing the behavior of a real-valued function $\boldsymbol{u}$ of n components and dependent on a parameter μ. The evolution of $\boldsymbol{u}$ in the spatial domain $\Omega \subset \mathbb{R}^d$, d denoting the dimensions of the problem, and time interval $]0, t_\mathrm{f}[$ is given by

$$\partial_t \boldsymbol{u}(\boldsymbol{x}, t, \mu) + \mathcal{N}(\boldsymbol{u}(\boldsymbol{x}, t, \mu)) = \boldsymbol{f}(\boldsymbol{x}, t, \mu) \quad \text{in } \Omega, \quad t \in]0, t_\mathrm{f}[, \tag{2}$$

where $\mathcal{N}$ is a nonlinear operator, $\boldsymbol{f}$ the forcing term and ∂_t the time derivative. Equation (2) is further provided with suitable boundary and initial conditions so that the problem is well-posed. For simplicity, μ is considered to be fixed for now, and hence, $\boldsymbol{u}$ will not be stated explicitly as a function of the parameter μ from now on tillSection 5, when parametric ROMs are discussed.

After the advent of the computational era, the most commonly used technique to solve (2) is to discretize it in space using a discretization technique, e.g., the FE method. This discretization leads to the a system of *ordinary differential equations* (ODEs) which reads: find $\boldsymbol{U} :]0, t_\mathrm{f}[\rightarrow \mathbb{R}^{np}$ such that

$$\boldsymbol{M}\partial_t \boldsymbol{U} + \boldsymbol{N}(\boldsymbol{U})\boldsymbol{U} = \boldsymbol{F}, \tag{3}$$

where $\boldsymbol{U}$ is the vector of unknowns, $\boldsymbol{M} \in \mathbb{R}^{np \times np}$ the mass-matrix, $\boldsymbol{N}(\boldsymbol{U}) \in \mathbb{R}^{np \times np}$ the matrix representation of the nonlinear differential operator $\mathcal{N}$, $\boldsymbol{F} \in \mathbb{R}^{np}$ the resulting forcing vector, and np the number of degrees of freedom. We shall denote as $\mathcal{B}_f = \mathbb{R}^{np}$ the FOM space.

Finally, a time-integration technique can be applied to (3) to obtain a fully-discrete system, which for a given time-step reads

$$\boldsymbol{A}(\boldsymbol{U})\boldsymbol{U} = \boldsymbol{R}, \tag{4}$$

where $\boldsymbol{A}(\boldsymbol{U}) \in \mathbb{R}^{np \times np}$ is the nonlinear system matrix and $\boldsymbol{R} \in \mathbb{R}^{np}$ is the right-hand-side which takes into account the contributions of the previous values of $\boldsymbol{U}$ as well. Equation (4) can then be solved for nt time-steps to get nt solution vectors.

To find the POD basis, all solution vectors are not generally required. Rather it is desired to select a minimum, but sufficient, number of solution vectors that contain all the important dynamic features of the system. A simple approach for a uniform step time-integration scheme is to use solution vectors after every i-th time-step, where i is a natural number [17]. Another approach could be to capture each cycle of a periodic phenomenon using a certain number of solution vectors [107]. Suppose, that we are able to gather a set of ns solution vectors, called *snapshots*, $\{\boldsymbol{U}^1, \boldsymbol{U}^2, ..., \boldsymbol{U}^{ns}\}$ carrying all the important features of the system dynamics. To simplify the exposition, we have assumed that the snapshots correspond to the first ns consecutive solution vectors, but this can be easily generalized. Note that the term *snapshots* will be interchangeably used for the solution vectors, as well as

their mean-subtracted form discussed in Section 2.2. Once the solution set has been gathered, the SVD is used to find the basis or POD modes.

2.2 Singular Value Decomposition

SVD, also known as *principal component analysis*, is one of the most important matrix factorization techniques used across many fields. The SVD of a matrix is guaranteed to exist and can be considered unique for the basis generation purposes. To perform the SVD, it is customary to first subtract the mean value $\overline{\boldsymbol{U}} \in \mathbb{R}^{np}$ from the solution vectors $\boldsymbol{U}^j$ to obtain $\boldsymbol{S}^j = \boldsymbol{U}^j - \overline{\boldsymbol{U}}$, for $j = 1, 2, ..., ns$. The mean-subtracted snapshots are then arranged into a matrix $\boldsymbol{S} \in \mathbb{R}^{np \times ns}$ as follows:

$$\boldsymbol{S} = \begin{bmatrix} | & | & & | \\ \boldsymbol{S}^1 & \boldsymbol{S}^2 & \dots & \boldsymbol{S}^{ns} \\ | & | & & | \end{bmatrix},$$

a tall skinny matrix as, in general, $ns \ll np$. It is now desired to find the basis of the space $\mathcal{B} \subset \mathcal{B}_f$ to which these snapshots belong using SVD. The SVD of $\boldsymbol{S}$ gives

$$\boldsymbol{S} = \boldsymbol{\Psi}' \boldsymbol{\Lambda}' \boldsymbol{V}^T, \tag{5}$$

where $\boldsymbol{\Psi}' \in \mathbb{R}^{np \times np}$ contains the left singular vectors, $\boldsymbol{\Lambda}' \in \mathbb{R}^{np \times ns}$ contains the singular values and $\boldsymbol{V} \in \mathbb{R}^{ns \times ns}$ contains the right singular vectors. Some important properties of the SVD are listed below:

- Matrices $\boldsymbol{\Psi}'$ and $\boldsymbol{V}$ are *orthogonal* i.e.

$$\boldsymbol{\Psi}'^T \boldsymbol{\Psi}' = \boldsymbol{\Psi}' \boldsymbol{\Psi}'^T = \boldsymbol{I}_{np} \quad \text{and} \quad \boldsymbol{V}^T \boldsymbol{V} = \boldsymbol{V} \boldsymbol{V}^T = \boldsymbol{I}_{ns}, \tag{6}$$

  where $\boldsymbol{I}_k$ is the identity matrix in $\mathbb{R}^k$.
- Matrix $\boldsymbol{\Lambda}'$ contains non-negative values along the diagonal, arranged in the decreasing order, and zeros elsewhere, i.e.,

$$\Lambda'_{ii} \geq \Lambda'_{jj} \geq 0, \quad \forall\, i < j \quad \text{and} \quad \Lambda'_{ij} = 0, \quad \forall\, i \neq j \tag{7}$$

Now, $\boldsymbol{\Lambda}'$ has at most ns non-zero values. Assuming such a case, it is possible to write $\boldsymbol{\Lambda}' = \begin{bmatrix} \boldsymbol{\Lambda} \\ \boldsymbol{0} \end{bmatrix}$, where $\boldsymbol{\Lambda} \in \mathbb{R}^{ns \times ns}$. Using this, the *full* SVD (5) can be converted to its *economy* or *reduced* SVD form as

$$\boldsymbol{S} = \boldsymbol{\Psi}' \begin{bmatrix} \boldsymbol{\Lambda} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{V}^T = \boldsymbol{\Psi} \boldsymbol{\Lambda} \boldsymbol{V}^T, \tag{8}$$

where $\boldsymbol{\Psi} \in \mathbb{R}^{np \times ns}$. The *full* and *reduced* versions of SVD are shown in Fig. 1. Note that (8) represents the exact decomposition of $\boldsymbol{S}$. The set of columns $\{\boldsymbol{\Psi}_j\}_{j=1}^{ns}$ of the

matrix $\mathbf{\Psi}$, represents the basis vectors of $\mathcal{B}$. So

$$\mathcal{B} = \text{span}\{\mathbf{\Psi}_1, \mathbf{\Psi}_2, ..., \mathbf{\Psi}_{ns}\}, \quad \text{where } \mathbf{\Psi}_j \in \mathbb{R}^{np}, \text{ for } j = 1, ..., ns.$$

So, the dimension of the solution space has been reduced from np to ns, with $ns \ll np$. However, ns still could be of the order of hundreds or even thousands and could still be considered computationally demanding. So, to unlock the full potential of ROMs in terms of computational savings, truncation is performed to yield a smaller number of basis vectors than the one provided by the *economy* SVD.

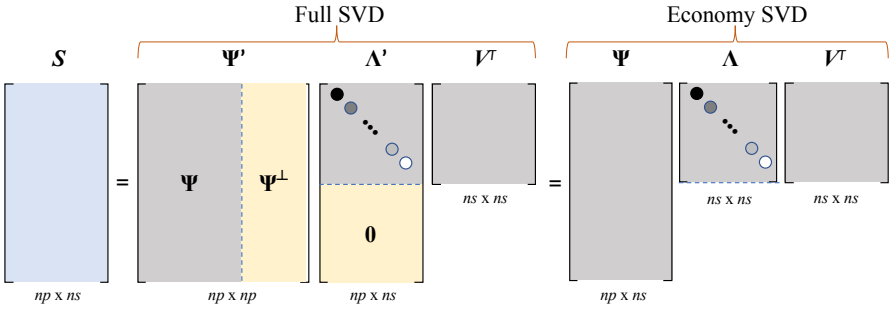

Fig. 1 Matrix representation of full and economy SVD. The lightening of the color of the circles represents the ordered decreasing of the diagonal values of $\mathbf{\Lambda}'$ and $\mathbf{\Lambda}$.

2.2.1 Truncated SVD

As discussed above, in practice, the basis is truncated to get $r < ns$ number of basis vectors. This leads to a *reduced space* $\mathcal{B}_r \subset \mathcal{B} \subset \mathcal{B}_f$, where $\dim\mathcal{B}_r = r$, $\dim\mathcal{B} = ns$ and $\dim\mathcal{B}_f = np$ and $r < ns \ll np$. The truncation is motivated by the ordered decreasing singular values in $\mathbf{\Lambda}$, Property (7). A singular value Λ_{ii} represents the amount of energy or information with which the corresponding basis vector $\mathbf{\Psi}_i$ contributes towards the solution. This contribution can be quantified using the relative information content (RIC) given by

$$\text{RIC} = \frac{\sum_{k=1}^{r} \Lambda_{kk}}{\sum_{k=1}^{ns} \Lambda_{kk}}.$$

The singular and RIC values for the classical problem of the flow over a cylinder approximated using the FE method are shown in Fig. 2. It can be seen that only a few initial values contain the most of the energy. So, instead of using 150 basis vectors, it is possible to use just a few of them to describe the flow dynamics around the cylinder. Based on the reasoning discussed above, RIC is widely used as a truncation criterion. The number of POD modes can then be decided such that RIC is equal to a desired value, e.g., 0.9, meaning that the POD modes will retain 90% of the

information. This truncation can be represented as a *truncated* SVD as

$$\boldsymbol{S} \approx \hat{\boldsymbol{S}} = \hat{\boldsymbol{\Psi}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{V}}^T, \tag{9}$$

where $\hat{\boldsymbol{\Psi}} \in \mathbb{R}^{np\times r}$ contains the first r columns of $\boldsymbol{\Psi}$, $\hat{\boldsymbol{\Lambda}} \in \mathbb{R}^{r\times r}$ contains the top left $r \times r$ block of $\boldsymbol{\Lambda}$ and $\hat{\boldsymbol{V}} \in \mathbb{R}^{ns\times r}$ contains first r columns of $\boldsymbol{V}$. Note that the truncated SVD only approximates matrix $\boldsymbol{S}$, as shown in (9). However, the truncated SVD is guaranteed to give the optimal approximation of $\boldsymbol{S}$ in the low-dimensional space as guaranteed by the Eckart–Young theorem [57].

Theorem 1 *The rank-r truncated SVD $\hat{\boldsymbol{S}}$ provides the best rank-r approximation to $\boldsymbol{S}$ in the L^2 sense, i.e.,*

$$\underset{\hat{\boldsymbol{S}} \text{ of rank } r}{\arg\min} \|\boldsymbol{S} - \hat{\boldsymbol{S}}\|_2 = \hat{\boldsymbol{\Psi}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{V}}^T.$$

Thus, the presence of patterns in the high dimensional data, shown by rapidly decreasing singular values, and the optimality of SVD guaranteed by the Eckart–Young theorem have resulted in the wide usage of POD to find the basis of the reduced spaces.

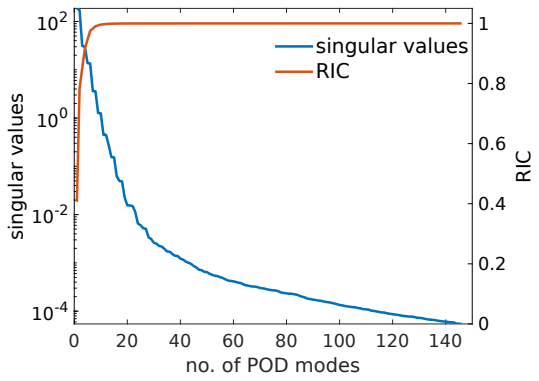

Fig. 2 Singular and RIC values for the flow over a cylinder. The singular values are represented on a log scale for better visualization.

2.2.2 SVD for Functions

The SVD (9) corresponds to solving the problem of minimizing

$$J(\hat{\boldsymbol{\Psi}}_1, ..., \hat{\boldsymbol{\Psi}}_r) = \sum_{i=1}^{ns} \left\| \boldsymbol{S}^i - \sum_{j=1}^{r} (\boldsymbol{S}^{iT}\hat{\boldsymbol{\Psi}}_j)\hat{\boldsymbol{\Psi}}_j \right\|^2_{\mathbb{R}^{np}}, \quad \text{subject to} \quad \hat{\boldsymbol{\Psi}}_i^T\hat{\boldsymbol{\Psi}}_j = \delta_{ij}. \tag{10}$$

Many times we are dealing with functions, e.g., when using FE methods, defined over the entire domain Ω, and not vectors corresponding to the degrees of freedom

of the approximation. In such cases, it could be desired for the properties to hold in the *functional (continuous)* sense rather than in the *algebraic (discrete)* sense. So, let us suppose that instead of solving the minimization problem (10), it is desired to minimize its *functional* counterpart

$$j(\hat{\phi}_1(\boldsymbol{x}), ..., \hat{\phi}_r(\boldsymbol{x})) = \sum_{i=1}^{ns} \left\| s^i(\boldsymbol{x}) - \sum_{j=1}^{r} \left(\int_\Omega s^i(\boldsymbol{x}) \hat{\phi}_j(\boldsymbol{x}) \right) \hat{\phi}_j(\boldsymbol{x}) \right\|^2_{L^2(\Omega)},$$

$$\text{subject to} \quad \int_\Omega \hat{\phi}_i(\boldsymbol{x}) \hat{\phi}_j(\boldsymbol{x}) = \delta_{ij}, \quad (11)$$

where $s^i(\boldsymbol{x})$ is the *functional form* of the vectors $\boldsymbol{S}^i$, $i = 1, ..., ns$, and $\hat{\phi}_j(\boldsymbol{x})$, $j = 1, ..., r$, are the *basis functions*, which are $L^2(\Omega)$-*orthogonal*. Note that (11) minimizes the difference over the entire domain Ω. For the sake of clarity, we have considered that the unknown of the problem is a scalar function, and so are the snapshots and the basis, but the extension to the vector case is straightforward. Also note that the difference between $\hat{\boldsymbol{\Psi}}$ and $\hat{\phi}(\boldsymbol{x})$ is not only of the vectorial and functional representation. Rather $\hat{\boldsymbol{\Psi}}$ and $\hat{\phi}(\boldsymbol{x})$ are two different bases having different orthogonal properties. The functional SVD (11) can be shown to be the same as minimizing

$$J(\hat{\boldsymbol{\Phi}}_1, ..., \hat{\boldsymbol{\Phi}}_r) = \sum_{i=1}^{ns} \left\| \boldsymbol{M}^{1/2} \boldsymbol{S}^i - \sum_{j=1}^{r} \left((\boldsymbol{M}^{1/2} \boldsymbol{S}^i)^T \boldsymbol{M}^{1/2} \hat{\boldsymbol{\Phi}}_j \right) \boldsymbol{M}^{1/2} \hat{\boldsymbol{\Phi}}_j \right\|^2_{\mathbb{R}^{nP}},$$

$$\text{subject to} \quad \hat{\boldsymbol{\Phi}}^T \boldsymbol{M} \hat{\boldsymbol{\Phi}} = \boldsymbol{I}_r, \quad (12)$$

where $\boldsymbol{M}$ is the mass-matrix as in (3) and $\boldsymbol{I}_r \in \mathbb{R}^{r \times r}$ is the identity matrix. Note that the $L^2(\Omega)$-*orthogonality* of *basis functions* $\hat{\boldsymbol{\phi}}(\boldsymbol{x})$ translates to *orthogonality* of the corresponding *basis vectors* $\hat{\boldsymbol{\Phi}}$ with respect to the mass-matrix $\boldsymbol{M}$ as $\hat{\boldsymbol{\Phi}}^T \boldsymbol{M} \hat{\boldsymbol{\Phi}} = \boldsymbol{I}_r$. To find $\hat{\boldsymbol{\Phi}}$, we perform the SVD of $\widetilde{\boldsymbol{S}} = \boldsymbol{M}^{1/2} \boldsymbol{S}$:

$$\widetilde{\boldsymbol{S}} = \widetilde{\boldsymbol{\Phi}} \hat{\boldsymbol{\Lambda}} \hat{\boldsymbol{V}}^T,$$

and the desired basis can be recovered as

$$\hat{\boldsymbol{\Phi}} = \boldsymbol{M}^{-1/2} \widetilde{\boldsymbol{\Phi}}$$

Using the *functional* SVD produced more accurate results in [54]. Note, however, that throughout this chapter the SVD term will refer to the one that solves problem (10), unless stated otherwise, e.g., in Section 3.3.3.

3 Reduced Order Modeling Using Proper Orthogonal Decomposition

As discussed in Section 1, reduced order modeling consists of offline and online steps. The offline step of finding the reduced order basis is discussed in Section 2. Now we describe the most commonly used method for the online step, the *Galerkin projection*, to find the *ROM coefficients* in (1).

3.1 Galerkin Projection

As $\hat{\boldsymbol{\Psi}}$ forms the basis of the reduced solution space $\mathcal{B}_r$, and was calculated from mean-subtracted snapshots, decomposition (1) can be written as

$$\boldsymbol{U} \approx \hat{\boldsymbol{\Psi}}\boldsymbol{U}_r + \overline{\boldsymbol{U}}, \tag{13}$$

where the ROM coefficients $\boldsymbol{U}_r \in \mathbb{R}^r$ are the components of $\boldsymbol{U}$ in $\mathcal{B}_r$ expressed in the reference system defined by $\hat{\boldsymbol{\Psi}}$. Given $\boldsymbol{U}_r$, $\boldsymbol{U}$ can be found using (13). Let us insert (13) in the original matrix system (4). Omitting the explicit dependence of $\boldsymbol{A}$ on $\boldsymbol{U}$ from the notation, we can write (4) as

$$\boldsymbol{A}\boldsymbol{U} \approx \boldsymbol{A}(\hat{\boldsymbol{\Psi}}\boldsymbol{U}_r + \overline{\boldsymbol{U}}) = \boldsymbol{R}.$$

Taking knowns to the right-hand-side, we get

$$\boldsymbol{A}\hat{\boldsymbol{\Psi}}\boldsymbol{U}_r = \boldsymbol{R} - \boldsymbol{A}\overline{\boldsymbol{U}}. \tag{14}$$

This leads to a $np \times r$ over-determined system. Let us assume $\boldsymbol{A}$ to be symmetric and positive definite (SPD). A least-square strategy for approximating (14) with respect to the norm induced by $\boldsymbol{A}^{-1}$, as described in [30, 35], leads to

$$\hat{\boldsymbol{\Psi}}^T\boldsymbol{A}\hat{\boldsymbol{\Psi}}\boldsymbol{U}_r = \hat{\boldsymbol{\Psi}}^T(\boldsymbol{R} - \boldsymbol{A}\overline{\boldsymbol{U}}). \tag{15}$$

which is the *Galerkin projection* of the full order system (4) onto the reduced space. Let us write (15) compactly as

$$\boldsymbol{A}_r\boldsymbol{U}_r = \boldsymbol{R}_r \tag{16}$$

where

$$\begin{aligned}\boldsymbol{A}_r &:= \hat{\boldsymbol{\Psi}}^T\boldsymbol{A}\hat{\boldsymbol{\Psi}} \quad \in \mathbb{R}^{r\times r},\\ \boldsymbol{R}_r &:= \hat{\boldsymbol{\Psi}}^T(\boldsymbol{R} - \boldsymbol{A}\overline{\boldsymbol{U}}) \quad \in \mathbb{R}^r.\end{aligned}$$

Applicable for the general matrices $\boldsymbol{A}$, the so-called *Petrov-Galerkin* (PG) projection is found to provide more stable results, as compared to Galerkin projection, in the case of $\boldsymbol{A}$ not being a SPD matrix [35]. Using $\hat{\boldsymbol{\Psi}}^T\boldsymbol{A}$ as a suitable PG projector,

the PG reduced order form of (4) is given by

$$\boldsymbol{A}_{rA}\boldsymbol{U}_r = \boldsymbol{R}_{rA}, \tag{17}$$

where now

$$\begin{aligned}\boldsymbol{A}_{rA} &:= \hat{\boldsymbol{\Psi}}^T \boldsymbol{A}^T \boldsymbol{A} \hat{\boldsymbol{\Psi}} \quad \in \mathbb{R}^{r\times r},\\ \boldsymbol{R}_{rA} &:= \hat{\boldsymbol{\Psi}}^T \boldsymbol{A}^T (\boldsymbol{R} - \boldsymbol{A}\overline{\boldsymbol{U}}) \quad \in \mathbb{R}^{r}.\end{aligned}$$

This corresponds to a least squares strategy for solving (14) with respect to the standard Euclidean norm in $\mathbb{R}^{np}$. Irrespective of the type of projection used, both the final reduced order systems (16) and (17) are $r \times r$ systems as opposed to the full order system (4) of size $np \times np$, with $r \ll np$. Thus the reduced order system can be *solved* at a fraction of the cost of the full order system. All the concepts described later apply to both the Galerkin and PG ROMs; however for simplicity, we will describe them using the Galerkin-ROM (15).

3.2 Hyperreduction

The ROM discussed above can be *solved* at a reduced computational expense. However, assembling the system matrices has a cost of the same order as that of the FOM. For linear problems, the assembling of matrices needs to be done once, and hence, is not considered a bottle-neck to achieving reduced computation times. However, for nonlinear problems the system matrices need to be assembled for every nonlinear iteration, i.e., multiple times for every time-step in general, and will lead to a significant cost. Thus, it is important to use some techniques to determine the nonlinear terms at a reduced cost. This is achieved using *hyperreduction* techniques and the resulting models are called *hyper-ROMs*. There are many methods used for hyperreduction including, but not limited to, empirical interpolation method [23] or its discrete version discrete empirical interpolation method (DEIM) [38], Gauss-Newton with approximate tensors [35], missing point estimator approach [12], cubature based approximation method [9], energy conserving sampling and weighting method [61] and adaptive mesh-refinement (AMR) based hyperreduction [118]. Here we briefly describe DEIM and AMR based hyperreduction.

Remark 3 In the case of a polynomial nonlinearity in general, and quadratic nonlinearity in particular, the reduced nonlinear operator can be written as a tensor which is not a function of $\boldsymbol{U}_r$, and hence, needs to be computed just once. However, hyperreduction techniques, like DEIM and AMR, described in this chapter are applicable in a broader context.

3.2.1 Discrete Empirical Interpolation Method

DEIM is a greedy algorithm and its origin can be traced back to the gappy POD method [59] which was originally designed for image reconstruction. Just as ROM approximates the *solution space* by a subspace, DEIM does the same but for *nonlinear terms* only. However, DEIM uses *interpolation indices* to find the temporal coefficients instead of solving the reduced system.

Let us denote the vector of nonlinear terms as $\boldsymbol{N}(\theta) \in \mathbb{R}^{np}$, depending on θ. θ can represent time t or any other parameter in the case of parametric ROMs. However, here we explain DEIM in the context of non-parametric nonlinear ROMs with $\theta = t$. For DEIM applied to parametric ROMs, see [10]. DEIM proposes *approximating* the space to which the $\boldsymbol{N}$ belongs by a subspace of lower dimension s, i.e., $s \ll np$ and not necessarily equal to the dimension r of the ROM space. Let this subspace be spanned by the basis $\boldsymbol{B} = [\boldsymbol{B}_1, ..., \boldsymbol{B}_s] \in \mathbb{R}^{np \times s}$. Thus we can write

$$\boldsymbol{N}(t) \approx \boldsymbol{B}\boldsymbol{d}(t) \tag{18}$$

where $\boldsymbol{d}(t)$ is the vector of coefficients. For simplicity, from now on the dependence on t will be omitted from the notation.

An efficient way to determine $\boldsymbol{d}$ is to sample s spatial points and use them to determine $\boldsymbol{d}$. This can be achieved using a sampling matrix $\boldsymbol{H}$ defined as

$$\boldsymbol{H} = [\boldsymbol{H}_{s_1}, ..., \boldsymbol{H}_{s_s}] \in \mathbb{R}^{np \times ns}$$

where $\boldsymbol{H}_{s_j} = [0, ..., 0, 1, 0, ...0]^T \in \mathbb{R}^{np}$, for $j = 1, ..., s$, is the s_j-th column of the identity matrix $\boldsymbol{I}_{np} \in \mathbb{R}^{np \times np}$. Using the sampling matrix $\boldsymbol{H}$ we can write

$$\boldsymbol{H}^T\boldsymbol{N} = \boldsymbol{H}^T\boldsymbol{B}\boldsymbol{d}.$$

Suppose $\boldsymbol{H}^T\boldsymbol{B}$ is non-singular, thus leading to a unique solution of $\boldsymbol{d}$ as

$$\boldsymbol{d} = (\boldsymbol{H}^T\boldsymbol{B})^{-1}\boldsymbol{H}^T\boldsymbol{N}. \tag{19}$$

Using (19), we can write (18) as

$$\boldsymbol{N} \approx \boldsymbol{B}(\boldsymbol{H}^T\boldsymbol{B})^{-1}\boldsymbol{H}^T\boldsymbol{N}. \tag{20}$$

Now we need to define the basis $\boldsymbol{B}$ and the sampling points s_j, $j = 1, ..., s$, called *interpolation indices* in DEIM, to approximate $\boldsymbol{N}$ using (20) at a reduced cost. The basis $\boldsymbol{B}$ is found using POD for the *nonlinear vector* $\boldsymbol{N}$. During the simulations, the nonlinear vectors at different time-steps are gathered to form a snapshot matrix $\boldsymbol{S}_N$ of nonlinear terms as

$$\boldsymbol{S}_N = \begin{bmatrix} | & | & & | \\ \boldsymbol{N}^1 & \boldsymbol{N}^2 & \dots & \boldsymbol{N}^{ns} \\ | & | & & | \end{bmatrix}.$$

The *truncated* SVD of rank-s is then performed as

$$\hat{\boldsymbol{S}}_N = \boldsymbol{B}\boldsymbol{\Lambda}_N \boldsymbol{V}_N^T \tag{21}$$

to obtain the basis $\boldsymbol{B}$ of the desired order. The interpolation indices are then selected iteratively using the basis $\boldsymbol{B}$. This approach is shown in Algorithm 1, where $[|\rho|\ y]$ = **max**$\{|\boldsymbol{X}|\}$ means that y is the index of the maximum value of the components of vector $\boldsymbol{X} = [X_1, ..., X_{np}]$, i.e., $X_y \geq X_z$, for $z = 1, ..., np$. The smallest y is taken if more than one component corresponds to the maximum value.

Algorithm 1 DEIM algorithm for the selection of *interpolation indices*

INPUT : Basis vectors $\{\boldsymbol{B}_1, ..., \boldsymbol{B}_s\}$
OUTPUT : *Interpolation indices* $\boldsymbol{s} = s_1, ..., s_s$

1: $[|\rho|\ s_1]$ = **max**$\{|\boldsymbol{B}_1|\}$ ▷ find the first index
2: $\boldsymbol{B} = [\boldsymbol{B}_1], \boldsymbol{H} = [\boldsymbol{H}_{s_1}]$ ▷ initialize matrices based on the first value
3: **for** $k = 2; k \leq s; k + 1$ **do** ▷ loop to find successively the remaining indices
4: $\quad \boldsymbol{H}^T\boldsymbol{B}\boldsymbol{d} = \boldsymbol{H}^T\boldsymbol{B}_k$ ▷ solve for $\boldsymbol{d}$
5: $\quad \boldsymbol{R}_k = \boldsymbol{B}_k - \boldsymbol{B}\boldsymbol{d}$ ▷ calculate residual $\boldsymbol{R}_k$
6: $\quad [|\rho|\ s_k]$ = **max**$\{|\boldsymbol{R}_k|\}$ ▷ find the index s_k where the residual has the maximum value
7: $\quad \boldsymbol{B} \leftarrow [\boldsymbol{B}\ \boldsymbol{B}_k], \boldsymbol{H} \leftarrow [\boldsymbol{H}\ \boldsymbol{H}_{s_k}]$ ▷ update the matrices for the next iteration
8: **end for**

3.2.2 Adaptive Mesh-Refinement Based Hyperreduction

AMR based hyperreduction proposes calculating the nonlinear terms on a mesh coarser than the one used for the FOM. The points of the coarser mesh are located using AMR. AMR based hyperreduction aims at concentrating the mesh in the regions of higher physics and coarsening it everywhere else such that the overall degrees of freedom are reduced. AMR uses a posteriori error-estimator to decide these areas of higher physics based activity. In [118] the mesh was coarsened such that the total error, in a certain norm, remained approximately the same before and after hyperreduction. An a posteriori residual-based error estimator was used and a coarse mesh containing 80% less degrees of freedom was achieved giving results with a negligible error. Numerical analysis of the error estimator was also performed in [49] and it was shown that the estimator provides an upper bound for the true error and has the correct numerical behavior.

3.3 Stabilization Using Variational Multiscale Methods

Instabilities can arise when PDEs are solved using numerical methods, usually in singular perturbation problems or when the approximation spaces of the different unknowns need to satisfy compatibility conditions. This issue is further exacerbated when POD-G is used to develop a ROM. This has to do with the fact that the ROM

does not account for the impact of the FOM scales that are not captured by the low-order space. This problem is well-known in other computational mechanics settings, such as finite elements, where stabilized formulations have been developed to address the instability of the Galerkin projection. The *Variational Multiscale* (VMS) framework, originally proposed in [79], is a popular framework used to develop stabilized formulations taking into account the effect of the discarded scales in a multi-scale problem. A comprehensive review of VMS-based stabilization methods developed for fluid problems is provided in [51]. Inspired by this, VMS based stabilization methods have been developed for projection based ROMs [118] and successfully applied in the context of flow problems [119], fluid-structure interaction [133, 134] and adaptive-mesh based hyperreduction [118]. A comprehensive description of it is provided in [49, 118]. However, a summary of the method, which uses the same VMS formulation to stabilize both FOM and ROM, is presented here for completeness. Let us describe the formulation using a general unsteady nonlinear convection-diffusion-reaction transport equation.

3.3.1 Variational Problem

Let us consider again problem (2) and write it in a slightly modified form, along with the boundary and initial conditions. Let the boundary Γ of the domain Ω be split into non-overlapping Dirichlet, Γ_D, and Neumann, Γ_N, parts. Given the initial condition for the unknown $\boldsymbol{u}^0$, the problem aims at finding $\boldsymbol{u}$ of n components that satisfies

$$\begin{aligned}
\partial_t \boldsymbol{u} + \mathcal{N}(\boldsymbol{u};\boldsymbol{u}) &= \boldsymbol{f} && \text{in } \Omega, && t \in]0, t_\mathrm{f}[,\\
\mathcal{D}\boldsymbol{u} &= \mathcal{D}\boldsymbol{u}_0 && \text{on } \Gamma_D, && t \in]0, t_\mathrm{f}[,\\
\mathcal{F}(\boldsymbol{u};\boldsymbol{u}) &= \boldsymbol{f}_N && \text{on } \Gamma_N, && t \in]0, t_\mathrm{f}[,\\
\boldsymbol{u} &= \boldsymbol{u}^0 && \text{in } \Omega, && t = 0,
\end{aligned}$$

where $\boldsymbol{u}_0$ is the prescribed Dirichlet boundary condition, $\mathcal{D}$ the Dirichlet operator, $\boldsymbol{f}_N$ the prescribed Neumann boundary condition and $\mathcal{F}$ the flux operator. Let us define $\mathcal{N}$ as a general nonlinear operator of second order using Einstein's notation:

$$\mathcal{N}(\boldsymbol{u};\boldsymbol{y}) := -\partial_i(\boldsymbol{K}_{ij}(\boldsymbol{u})\partial_j\boldsymbol{y}) + \boldsymbol{A}_{f,i}(\boldsymbol{u})\partial_i\boldsymbol{y} + \boldsymbol{A}_{c,i}(\boldsymbol{u})\partial_i\boldsymbol{y} + \boldsymbol{S}(\boldsymbol{u})\boldsymbol{y},$$

where $\boldsymbol{K}_{ij}$, $\boldsymbol{A}_{c,i}$, $\boldsymbol{A}_{f,i}$ and $\boldsymbol{S}$ are matrices in $\mathbb{R}^{n\times n}$ and are a function of $\boldsymbol{u}$, ∂_i denotes differentiation with respect to the i-th Cartesian coordinate x_i and indexes $i, j = 1, ..., d$. Let us also define the flux operator $\mathcal{F}$ using Einstein's notation as

$$\mathcal{F}(\boldsymbol{u};\boldsymbol{y}) := n_i\boldsymbol{K}_{ij}(\boldsymbol{u})\partial_j\boldsymbol{y} - n_i\boldsymbol{A}_{f,i}(\boldsymbol{u})\boldsymbol{y},$$

where $\boldsymbol{n}$ is the external unit normal to the boundary Γ with n_i being its i-th component.

To write the weak form of the problem, let the integral of the product of two functions, $\boldsymbol{f}$ and $\boldsymbol{g}$ over the domain ω be defined by $\langle \boldsymbol{f}, \boldsymbol{g}\rangle_\omega$. For simplicity the

subscript ω is omitted in the case $\omega = \Omega$. Let us also introduce the form B_ω and the linear form L_ω as

$$\begin{aligned} B_\omega(\boldsymbol{u};\boldsymbol{y},\boldsymbol{v})_\omega &:= \langle \partial_i \boldsymbol{v}, \boldsymbol{K}_{ij}(\boldsymbol{u})\partial_j \boldsymbol{y}\rangle_\omega + \langle \boldsymbol{v}, \boldsymbol{A}_{c,i}(\boldsymbol{u})\partial_i \boldsymbol{y}\rangle_\omega + \langle \partial_i(\boldsymbol{A}^T_{f,i}(\boldsymbol{u})\boldsymbol{v}), \boldsymbol{y}\rangle_\omega \\ &\quad + \langle \boldsymbol{v}, \boldsymbol{S}(\boldsymbol{u})\boldsymbol{y}\rangle_\omega, \\ L_\omega(\boldsymbol{v}) &:= \langle \boldsymbol{v}, \boldsymbol{f}\rangle_\omega + \langle \boldsymbol{v}, \boldsymbol{f}_N\rangle_{\Gamma_N}. \end{aligned}$$

Let $\boldsymbol{u}(.,t)$ and $\boldsymbol{v}$ belong to the space $\mathcal{B}_c$, the solution space of the continuous problem. The weak form of the problem (in space) consists of finding $\boldsymbol{u} :]0, t_f[\rightarrow \mathcal{B}_c$ such that

$$\langle \partial_t \boldsymbol{u}, \boldsymbol{v}\rangle + B(\boldsymbol{u};\boldsymbol{u},\boldsymbol{v}) = L(\boldsymbol{v}), \tag{22}$$

$$\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \langle \boldsymbol{u}^0, \boldsymbol{v}\rangle, \quad \text{at } t = 0 \tag{23}$$

for all $\boldsymbol{v} \in \mathcal{B}_{c,0}$, where $\mathcal{B}_{c,0}$ is the space of time independent test functions that satisfy $\mathcal{D}\boldsymbol{v} = \boldsymbol{0}$ on Γ_D. For simplicity, we assume in what follows homogeneous Dirichlet conditions so that $\boldsymbol{v} \in \mathcal{B}_c = \mathcal{B}_{c,0}$.

3.3.2 Variational Multiscale Full Order Model Approximation

VMS method can be applied to other discretization techniques, but in what follows we shall concentrate on the FE method. Thus, let us discretize the domain using FEs. Let $\mathcal{P}_h = \{K\}$ be a FE partition of the domain Ω, assumed quasi-uniform for simplicity, with elements of size h. From this, a conforming FE space $\mathcal{B}_h \subset \mathcal{B}_c$ may be constructed using a standard approach. Note that now $\mathcal{B}_h = \mathcal{B}_f$, i.e., the FE space is a particular realization of the FOM space introduced earlier.

Any time integration scheme may be used for the time discretization. For conciseness, we shall assume that a backward difference scheme is employed with a uniform time step Δt and the time discretization is represented by replacing ∂_t with δ_t, where δ_t involves $\boldsymbol{u}_h^n, \boldsymbol{u}_h^{n-1}, ...$, depending on the order of the scheme used. Using a superscript n for the time step counter, and a subscript h for FE quantities, the fully discretized Galerkin approximation of problem (22) is to find $\{\boldsymbol{u}_h^n\} \in \mathcal{B}_h$, for $n = 1, ..., nt$, nt being the number of time steps, that satisfy

$$\langle \delta_t \boldsymbol{u}_h, \boldsymbol{v}_h\rangle + B(\boldsymbol{u}_h;\boldsymbol{u}_h,\boldsymbol{v}_h) = L(\boldsymbol{v}_h) \quad \forall\, \boldsymbol{v}_h \in \mathcal{B}_h,$$

where we have omitted the initial conditions and the time step superscript for simplicity. This problem may suffer from instabilities and, hence, it requires the use of stabilization methods like those based on VMS method.

The core idea of the VMS approach lies in the splitting $\mathcal{B}_c = \mathcal{B}_h \oplus \mathcal{B}'$, where $\mathcal{B}'$is any space that completes $\mathcal{B}_h$ in $\mathcal{B}_c$. We call $\mathcal{B}'$ the space of *sub-grid scales* or *subscales* (SGS), and the functions in the SGS spaces will be identified with the superscript $'$. Using the splitting $\boldsymbol{u} = \boldsymbol{u}_h + \boldsymbol{u}'$ and similarly for the test function $\boldsymbol{v} = \boldsymbol{v}_h + \boldsymbol{v}'$, the continuous problem (22) splits into

$$\langle \delta_t(\boldsymbol{u}_h + \boldsymbol{u}'), \boldsymbol{v}_h \rangle + B(\boldsymbol{u}_h + \boldsymbol{u}'; \boldsymbol{u}_h + \boldsymbol{u}', \boldsymbol{v}_h) = L(\boldsymbol{v}_h), \qquad \forall\, \boldsymbol{v}_h \in \mathcal{B}_h, \tag{24}$$

$$\langle \delta_t(\boldsymbol{u}_h + \boldsymbol{u}'), \boldsymbol{v}' \rangle + B(\boldsymbol{u}_h + \boldsymbol{u}'; \boldsymbol{u}_h + \boldsymbol{u}', \boldsymbol{v}') = L(\boldsymbol{v}'), \qquad \forall\, \boldsymbol{v}' \in \mathcal{B}', \tag{25}$$

which is exactly equivalent to (22) as no assumptions have been made so far. We want to find the SGSs $\boldsymbol{u}'$ using (25) and plug them into (24) to account for their effect on $\boldsymbol{u}_h$. To achieve this, several assumptions are made, which are briefly discussed below (see [51]).

Important Considerations/Assumptions

- Choosing the subscale space $\mathcal{B}'$. In fact, the approximation to $\mathcal{B}'$ will be a consequence of the approximation to $\boldsymbol{u}'$. The choice of SGS space leads to *algebraic subgrid scales* [44] or *orthogonal subgrid scales*, among other possibilities [45].
- While expanding (25), we come across the application of the operator $\mathcal{N}$ to subscales as $\mathcal{N}(\boldsymbol{u}; \boldsymbol{u}')$. Because the subscale problem is infinite dimensional, the following *key* approximation is used

$$\mathcal{N}(\boldsymbol{u}; \boldsymbol{u}')|_K \approx \boldsymbol{\tau}_K^{-1}(\boldsymbol{u})\boldsymbol{u}'|_K,$$

  where $\boldsymbol{\tau}_K$ is a matrix of *stabilization parameters* that approximates the inverse of the differential operator on each element K. Different approximations for $\boldsymbol{\tau}_K$ yield different VMS methods.
- Taking $\delta_t \boldsymbol{u}'$ into account or not. Taking it into consideration yields *dynamic* SGSs [50], whereas, assuming $\delta_t \boldsymbol{u}' = 0$ results in *quasi-static* SGSs.
- For $B(\boldsymbol{u}; \boldsymbol{y}, \boldsymbol{v})$, $\boldsymbol{u}$ can be approximated as $\boldsymbol{u} \approx \boldsymbol{u}_h$ instead of $\boldsymbol{u} = \boldsymbol{u}_h + \boldsymbol{u}'$ (this can be relaxed, see [46]). The first approach is known as *linear* SGSs and the later as *nonlinear* SGGs, in accordance with the inclusion of SGSs in only linear or both, linear and nonlinear, terms, respectively. $\boldsymbol{u}$ representing the nonlinearity will be replaced by $\boldsymbol{u}^*$ to represent any of the above possibilities in a general way.
- Approximate the SGSs in the interior of the elements only or consider their contributions on the interelement boundaries as well [14, 47]. The use of interelement boundary SGSs becomes crucial when discontinuous interpolations are used for some components of the unknown [48] or when SGSs are used as a posteriori error estimator [49].

The final problem can be written as finding $\boldsymbol{u}_h^n \in \mathcal{B}_h$, for $n = 1, ..., nt$, that satisfy

$$\langle \delta_t \boldsymbol{u}_h, \boldsymbol{v}_h \rangle + B_h(\boldsymbol{u}^*; \boldsymbol{u}_h, \boldsymbol{v}_h) = L_h(\boldsymbol{u}^*; \boldsymbol{v}_h), \quad \forall\, \boldsymbol{v}_h \in \mathcal{B}_h, \tag{26}$$

with the forms B_h and L_h given as

$$B_h(\boldsymbol{u}^*; \boldsymbol{u}_h, \boldsymbol{v}_h) = B(\boldsymbol{u}^*; \boldsymbol{u}_h, \boldsymbol{v}_h) + B'(\boldsymbol{u}^*; \boldsymbol{u}_h, \boldsymbol{v}_h) \tag{27}$$

$$L_h(\boldsymbol{u}^*; \boldsymbol{v}_h) = L(\boldsymbol{v}_h) + L'(\boldsymbol{u}^*; \boldsymbol{v}_h) \tag{28}$$

where $B'(\boldsymbol{u}^*;\boldsymbol{u}_h,\boldsymbol{v}_h)$ and $L'(\boldsymbol{u}^*;\boldsymbol{v}_h)$ are defined based on the choices made regarding the considerations discussed above. $B'(\boldsymbol{u}^*;\boldsymbol{u}_h,\boldsymbol{v}_h)$ and $L'(\boldsymbol{u}^*;\boldsymbol{v}_h)$ for different combination of choices can be found in [118].

3.3.3 Variational Multiscale Reduced Order Model Approximation

A ROM for the FOM discussed above can be developed by constructing a ROM space $\mathcal{B}_r \subset \mathcal{B}_h \subset \mathcal{B}_c$. Using the POD relying on SVD for functions, described in Section 2.2.2, we may obtain a ROM space of dimension r

$$\mathcal{B}_r = \text{span}\{\hat{\boldsymbol{\Phi}}_1, \hat{\boldsymbol{\Phi}}_2, ..., \hat{\boldsymbol{\Phi}}_r\},$$

such that $r = \dim \mathcal{B}_r \ll \dim \mathcal{B}_h$.

The proposed VMS formulation for ROM is exactly the same as the one used for the FOM with one key difference, i.e., the functions are now approximated in $\mathcal{B}_r$ instead of $\mathcal{B}_h$. The rationale behind this is the fact that $\mathcal{B}_r \subset \mathcal{B}_h$ and, specifically, it is possible to write the ROM basis functions as a linear combination of the basis of the FE space $\mathcal{B}_h$, and therefore the ROM basis functions are piecewise polynomials defined on the FE partition $\mathcal{P}_h$. The use of the same stabilization parameters for the FOM and the ROM is also justified based on the above reasoning. Applying the VMS concept to the ROM we will have the decomposition

$$\mathcal{B} = \mathcal{B}_h \oplus \mathcal{B}' = \mathcal{B}_r \oplus \mathcal{B}'',$$

where $\mathcal{B}''$ is any space that completes $\mathcal{B}_r$ in $\mathcal{B}$.

Remark 4 An interesting observation can be made when $L^2(\Omega)$-orthogonal SGSs are used in conjunction with $L^2(\Omega)$-orthogonal basis $\hat{\boldsymbol{\Phi}}$ obtained by solving (12). Suppose that we were able to construct a POD basis of $\mathcal{B}_h$, with $np = \dim \mathcal{B}_h$, i.e.,

$$\mathcal{B}_h = \text{span}\{\hat{\boldsymbol{\Phi}}_1, \hat{\boldsymbol{\Phi}}_2, ..., \hat{\boldsymbol{\Phi}}_{np}\}.$$

Then, since the basis vectors obtained from the POD are $L^2(\Omega)$-orthogonal, choosing orthogonal subgrid scales allows us to write

$$\mathcal{B}'' = \text{span}\{\hat{\boldsymbol{\Phi}}_{r+1}, \hat{\boldsymbol{\Phi}}_{r+2}, ..., \hat{\boldsymbol{\Phi}}_{np}\} \oplus \mathcal{B}',$$

i.e., we have an explicit representation of the ROM space of SGSs. So, when VMS-ROM is used to approximate ROM SGSs, it accounts for the FOM subscales, present in the subspace $\mathcal{B}'$, as well as the SGSs arising as a result of ROM trunctaion, present in the subspace spanned by $\{\hat{\boldsymbol{\Phi}}_{r+1}, \hat{\boldsymbol{\Phi}}_{r+2}, ..., \hat{\boldsymbol{\Phi}}_{np}\}$.

Having in mind the previous discussion, the final reduced order problem can be written as finding $\boldsymbol{u}_r^n \in \mathcal{B}_r$, for $n = 1, ..., nt$, that satisfy

$$\langle \delta_t \boldsymbol{u}_r, \boldsymbol{v}_r \rangle + B_r(\boldsymbol{u}^*;\boldsymbol{u}_r,\boldsymbol{v}_r) = L_r(\boldsymbol{u}^*;\boldsymbol{v}_r), \quad \forall\, \boldsymbol{v}_r \in \mathcal{B}_r, \tag{29}$$

with the forms B_r and L_r given as

$$B_r(\boldsymbol{u}^*; \boldsymbol{u}_r, \boldsymbol{v}_r) = B(\boldsymbol{u}^*; \boldsymbol{u}_r, \boldsymbol{v}_r) + B'(\boldsymbol{u}^*; \boldsymbol{u}_r, \boldsymbol{v}_r) \tag{30}$$

$$L_r(\boldsymbol{u}^*; \boldsymbol{v}_r) = L(\boldsymbol{v}_r) + L'(\boldsymbol{u}^*; \boldsymbol{v}_r). \tag{31}$$

It can be seen that the Equations (29)-(31) look exactly the same as (26)-(28). Furthermore, the expressions for B' and L' are also the same for the FOM and the ROM if the same choices are made for both, regarding the considerations discussed in Section 3.3.2. The only difference between the FOM and the ROM formulation is that in the case of ROM, functions are approximated in $\mathcal{B}_r$ instead of $\mathcal{B}_h$. $B'(\boldsymbol{u}^*; \boldsymbol{u}_r, \boldsymbol{v}_r)$ and $L'(\boldsymbol{u}^*; \boldsymbol{v}_r)$ for different combination of choices can be found in [118].

3.3.4 Other Stabilized Reduced Order Models

Streamline-Upwind Petrov–Galerkin (SUPG), a popular scheme for stabilized FE methods introduced in [27], has been used to deal with the instabilities in the context of projection based ROMs as well [31, 67, 66, 82, 113]. In the case of existence of compatibility conditions between the approximation spaces for difference unknowns, enrichment of approximation spaces using the so called *supremizers* has been used to provide the required stability to ROMs [22, 120]. A grad-div stabilization was used for the POD-G method and an error analysis was carried out for the resulting formulation in [65]. A *streamline derivative* stabilization term, as well as an a posteriori stabilization method, were used in [13]. *Least square Petrov-Galerkin* has also been used to stabilize the ROMs at the fully-discrete system level [35, 52] and compared with the Galerkin projection in [34]. A nonintrusive stabilization method for projection based ROMs was proposed in [8] which can be applied as a black-box post processing step. A POD mode dependent eddy viscosity stabilization scheme was developed and applied to quasigeostrophic ocean circulation in [123]. In the domain of hyperreduction, algorithms like DEIM have been found to exhibit numerical instabilities for second-order dynamical systems [61]. In such cases energy conserving sampling and weighting method can be used, which provides the required stability by conserving the energy.

4 Non-Intrusive Reduced Order Models

4.1 The General Concept

The Galerkin projection discussed in Section 3.1 is an intrusive approach, i.e., it requires knowledge of the governing equations and/or access to the code used for the FOM. There is another class of purely data-driven ROMs called non-intrusive reduced order models (NIROMs) which provide solutions based only on the data and without using the governing equations. NIROMs allow the decoupling of the

FOM and the ROM implementations completely and are particularly useful in the cases where the code used for the FOM is not open-source. NIROMs can be obtained using *conventional* or *ML-based* techniques and the recent large-scale adoption of NIROMs can be attributed to the increasing popularity of ML in scientific computing. The ML-based NIROMs are later discussed in Section 6.2. For now, we describe dynamic mode decomposition (DMD), which can be considered a *conventional* non-intrusive extension of the POD.

4.2 Dynamic Mode Decomposition

Dynamic mode decomposition (DMD), originally introduced in [127] in the context of fluid dynamics, aims at identifying spatio-temporal coherent structures in high-dimensional data. DMD computes a set of spatial modes, as well as their temporal evolution. It characterizes the temporal evolution simply as an oscillation of fixed frequency with a growth or decay rate. So, while POD-G relies on solving a Galerkin projected system to find the temporal evolution of the modes, DMD only requires the data to define the temporal behavior of modes. An important consideration regarding DMD modes is that they are not orthogonal by construction, and hence, may require more modes than POD-G to capture the same phenomena. Recursive DMD [110], a variant of the original DMD, has been developed to produce orthogonal modes in a recursive manner. Furthermore, there is a lack of consensus regarding the best criteria to be used for selecting the dominant DMD modes [111, 135]. A number of other specialized DMD algorithms have been developed, including, but not limited to, exact DMD [136], optimized DMD [39], extended DMD [143] and time-delayed DMD [29]. DMD has been successfully applied across a wide range of applications in fluid mechanics [6, 85, 122, 128].

Let us describe how a basic DMD algorithm can be used to obtain DMD modes and DMD eigenvalues (which describe the temporal evolution) from data. The same symbols and terminologies are used, where applicable, as was used to describe SVD in Section 2.2. First, two different sets of snapshots are gathered; let us denote them by $\{\boldsymbol{S}^i\}_{i=1}^{ns}$ and $\{\boldsymbol{S}^{*i}\}_{i=1}^{ns}$. The snapshot pairs are such that $\boldsymbol{S}^{*i}$ is obtained by evolving the system state by time-step Δt using $\boldsymbol{S}^i$ as the initial conditions, for $i = 1, ..., ns$, and with Δt small enough to resolve the temporal dynamics to the smallest desired scale. The gathered snapshots are then arranged in matrices, $\boldsymbol{S}$ and $\boldsymbol{S}^*$, given by

$$\boldsymbol{S} = \begin{bmatrix} | & | & & | \\ \boldsymbol{S}^1 & \boldsymbol{S}^2 & \dots & \boldsymbol{S}^{ns} \\ | & | & & | \end{bmatrix} \quad \text{and} \quad \boldsymbol{S}^* = \begin{bmatrix} | & | & & | \\ \boldsymbol{S}^{*1} & \boldsymbol{S}^{*2} & \dots & \boldsymbol{S}^{*ns} \\ | & | & & | \end{bmatrix}.$$

Let $\boldsymbol{T} \in \mathbb{R}^{np \times np}$ be a best-fit linear operator which relates $\boldsymbol{S}$ and $\boldsymbol{S}^*$ as

$$\boldsymbol{S}^* = \boldsymbol{T}\boldsymbol{S} \tag{32}$$

i.e. $\boldsymbol{T}$ acts as a time-integrator.

Now, it is desired to find the eigenvalues and eigenvectors of matrix $\boldsymbol{T}$. This matrix $\boldsymbol{T}$ is a $np \times np$ matrix, and hence, it is impractical to perform its eigendecomposition. DMD provides an efficient way of finding the r leading eigenvalues and eigenvectors of matrix $\boldsymbol{T}$. Using $\boldsymbol{S}^+$ as the pseudo-inverse of $\boldsymbol{S}$, (32) can be written as

$$\boldsymbol{T} = \boldsymbol{S}^* \boldsymbol{S}^+ \tag{33}$$

SVD can be used approximate the pseudo-inverse $\boldsymbol{S}^+$. Using a *truncated* SVD, it can be written

$$\boldsymbol{S} \approx \hat{\boldsymbol{\Psi}} \hat{\boldsymbol{\Lambda}} \hat{\boldsymbol{V}}^T. \tag{34}$$

As $\hat{\boldsymbol{\Psi}}$ and $\hat{\boldsymbol{V}}$ are orthogonal and satisfy $\hat{\boldsymbol{\Psi}}^T \hat{\boldsymbol{\Psi}} = \boldsymbol{I}_{np}$ and $\hat{\boldsymbol{V}}^T \hat{\boldsymbol{V}} = \boldsymbol{I}_{ns}$, using (34) allows us to write (33) as

$$\boldsymbol{T} \approx \boldsymbol{S}^* \hat{\boldsymbol{V}} \hat{\boldsymbol{\Lambda}}^{-1} \hat{\boldsymbol{\Psi}}^T,$$

where $\hat{\boldsymbol{\Lambda}}$ is a diagonal matrix and can be easily inverted. We are only interested in the first r eigenvalues and eigenvectors of matrix $\boldsymbol{T}$. The r-rank approximation of $\boldsymbol{T}$, denoted by $\boldsymbol{T}_r \in \mathbb{R}^{r \times r}$, is achieved by projecting $\boldsymbol{T}$ on to the reduced space using basis $\hat{\boldsymbol{\Psi}}$ as

$$\begin{aligned} \boldsymbol{T}_r &= \hat{\boldsymbol{\Psi}}^T \boldsymbol{T} \hat{\boldsymbol{\Psi}} \\ &= \hat{\boldsymbol{\Psi}}^T \boldsymbol{S}^* \hat{\boldsymbol{V}} \hat{\boldsymbol{\Lambda}}^{-1} \hat{\boldsymbol{\Psi}}^T \hat{\boldsymbol{\Psi}} \\ &= \hat{\boldsymbol{\Psi}}^T \boldsymbol{S}^* \hat{\boldsymbol{V}} \hat{\boldsymbol{\Lambda}}^{-1}. \end{aligned}$$

Now, the DMD eigenvalues can be found by performing the eigendecomposition of the reduced operator $\boldsymbol{T}_r$ as

$$\boldsymbol{T}_r \boldsymbol{E} = \boldsymbol{E} \Upsilon$$

where the entries of the diagonal matrix Υ are the eigenvalues of the low-dimensional $\boldsymbol{T}_r$, as well as the high-dimensional $\boldsymbol{T}$. $\boldsymbol{E}$ contains the eigenvectors of $\boldsymbol{T}_r$ and allows us to obtain the eigenvectors of $\boldsymbol{T}$, denoted as $\boldsymbol{\varphi}$, as

$$\boldsymbol{\varphi} = \boldsymbol{S}^* \hat{\boldsymbol{V}} \hat{\boldsymbol{\Lambda}}^{-1} \boldsymbol{E}$$

where the columns of $\boldsymbol{\varphi} \in \mathbb{R}^{np \times r}$, called DMD modes, are the eigenvectors of $\boldsymbol{T}$. Once DMD eigenvalues and eigenvectors have been determined, the state of the system at the k-th time-step, $\boldsymbol{U}^k$, is given by

$$\boldsymbol{U}^k = \boldsymbol{\varphi} \Upsilon^{k-1} \boldsymbol{D},$$

where $\boldsymbol{D} \in \mathbb{R}^r$ is the vector of mode amplitudes that can be computed using initial conditions. The DMD of a flow over a cylinder is illustrated in Fig. 3.

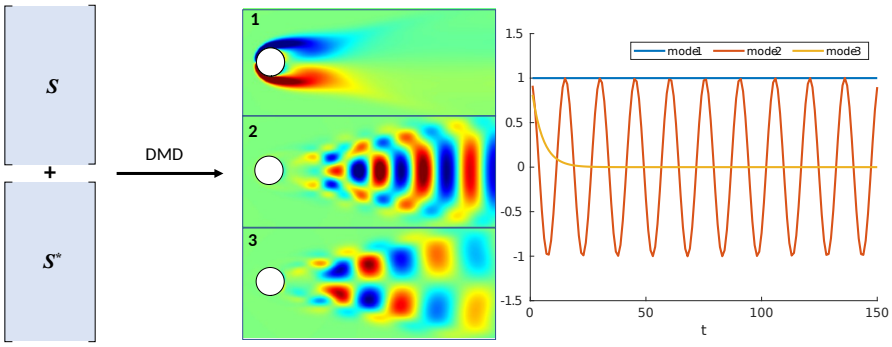

Fig. 3 Illustration of DMD applied to a flow over a cylinder. Three DMD modes and the temporal evolution of their coefficients is shown.

5 Parametric Reduced Order Models

In the previous sections, we have discussed how to build a ROM during an *offline* stage and how to use it for getting results quickly during an *online* stage. So far, we have assumed that the unknown $\boldsymbol{U}(t,\mu)$ was a function of t only and the parameter $\mu \in \mathcal{D} \subset \mathbb{R}$, was kept constant. So, in essence, the ROM was used to solve exactly the same problem whose solution was used to generate the snapshots to be used for the ROM basis generation. The aim of reduced order modeling is to perform the computationally expensive offline stage *once* (or a *few times*) and then use the generated ROM to perform *many* simulations in the cheap online phase for the new values of the parameter μ. This situation arises routinely in optimization and control problems governed by parametric PDEs. The parameter can represent anything including boundary conditions, geometry, viscosity, Reynold's number, etc. For simplicity, we assume that the parameter represents a scalar and its different values, $\mu_1, ..., \mu_{ps}$, represent different *configurations*, however, the subsequent discussion is equally valid where μ represents more than one parameters. The difficulty with parametric reduced order models (PROMs) lies in the fact that the basis $\boldsymbol{\Psi}_{\mu_1}$ obtained for μ_1 is unlikely to perform well for μ_2 as the behavior captured by the basis $\boldsymbol{\Psi}_{\mu_1}$ might be different from the behavior exhibited by the system for μ_2, i.e.

$$\boldsymbol{U}(\mu_1) \approx \boldsymbol{\Psi}_{\mu_1} \boldsymbol{U}_r(\mu_1),$$

but

$$\boldsymbol{U}(\mu_2) \not\approx \boldsymbol{\Psi}_{\mu_1} \boldsymbol{U}_r(\mu_2).$$

Several techniques have been developed to obtain a suitable basis for PROMs. Hyperreduction techniques, like DEIM described in Section 3.2.1, can also be used for PROMS [10] with $\theta = \mu$. Here, we describe two popular techniques to obtain a basis for PROMs, the global basis method and the local basis with interpolation method. These techniques commonly use a greedy approach to sample suitable

parameter values to obtain the snapshots. Thus, they are commonly referred to as POD-greedy approaches.

5.1 Global Basis

Probably the most obvious approach is to sample different parameter values, obtain snapshots corresponding to them and perform the SVD on all the snapshots to obtain a single global basis $\hat{\boldsymbol{\Psi}}$ such that

$$\boldsymbol{U}(\mu) \approx \hat{\boldsymbol{\Psi}}\boldsymbol{U}_r(\mu), \quad \forall\, \mu \in \mathcal{D}. \tag{35}$$

A greedy approach can be used to sample ps parameter values to obtain the snapshots. The global basis approach can provide a compact r dimensional basis satisfying (35) if the solution is not very sensitive to the parameter μ, i.e., the solution manifold has rapidly decaying Kolmogorov n-width. If the solution manifold has slow decaying Kolmogorov n-width, it might require obtaining snapshots at a lot of sampled parameter values, which can lead to a prohibitively expensive offline phase. Even if the computational expense of the offline phase is completely ignored, achieving a reasonable accuracy in the online phase will require a lot of POD modes. Hence, truncating the global basis to a rank r, ensuring a real-time execution of the online phase with reasonable accuracy, will not be possible.

5.2 Local Basis with Interpolation

In the case that the global basis approach is not feasible, local basis can be developed and used with interpolation. Similar to the global basis approach, ps parameter values are sampled and the snapshots are obtained for them. However, instead of performing a SVD of the matrix containing all the snapshots, a separate SVD is performed for the snapshot matrix for every sampled parameter value μ to obtain a corresponding *local* basis $\boldsymbol{\Psi}_{\mu_i}$, for $i = 1, ..., ps$. Now, the basis $\boldsymbol{\Psi}_{\mu^*}$ can be obtained at a requested, but unsampled, parameter value μ^* using interpolation.

If conventional interpolation techniques are used, the interpolated basis $\boldsymbol{\Psi}_{\mu^*}$ is likely to lose the key properties, e.g., orthogonality, after interpolation. Hence, interpolation using property preserving matrix manifolds is recommended to preserve the key properties. Let $\mathcal{G}$ be such a manifold of orthogonal matrices. Also, let $\{\mu_i\}_{i=1}^{ps}$ be the set of sampled parameter values and $\{\boldsymbol{\Psi}_{\mu_i}\}_{i=1}^{ps}$ be the set of corresponding bases. The basis $\boldsymbol{\Psi}_{\mu^*}$ for the unsampled point $\mu^* \notin \{\mu_i\}_{i=1}^{ps}$ can be obtained as follows. First, a tangent space $\mathcal{T}(\boldsymbol{\Psi}_{\tilde{\mu}})$ is defined such that it is tangent to $\mathcal{G}$ at a reference point $\boldsymbol{\Psi}_{\tilde{\mu}} \in \{\boldsymbol{\Psi}_{\mu_i}\}_{i=1}^{ps}$. Now, $\boldsymbol{\Psi}_{\mu_i}$, $i = 1, ..., ps$, except the reference point $\boldsymbol{\Psi}_{\tilde{\mu}}$, are projected to the tangent space using a *logarithmic map* defined as

$$T(\boldsymbol{\Psi}_{\mu_i}) = \boldsymbol{R}_{\mu_i} \tan^{-1}(\boldsymbol{\Xi}_{\mu_i})\boldsymbol{W}_{\mu_i}^T$$

with $\boldsymbol{R}_{\mu_i}$, $\boldsymbol{\Xi}_{\mu_i}$ and $\boldsymbol{W}^T_{\mu_i}$ obtained from the following SVD:

$$(\boldsymbol{\Psi}_{\mu_i} - \boldsymbol{\Psi}_{\tilde{\mu}}\boldsymbol{\Psi}^T_{\tilde{\mu}}\boldsymbol{\Psi}_{\mu_i})(\boldsymbol{\Psi}^T_{\tilde{\mu}}\boldsymbol{\Psi}_{\mu_i})^{-1} = \boldsymbol{R}_{\mu_i}\boldsymbol{\Xi}_{\mu_i}\boldsymbol{W}^T_{\mu_i},$$

where $T(\boldsymbol{\Psi}_{\mu_i})$ is the projection of $\boldsymbol{\Psi}_{\mu_i}$ on the tangent space $\mathcal{T}(\boldsymbol{\Psi}_{\tilde{\mu}})$, so that after projecting $\{\boldsymbol{\Psi}_{\mu_i}\}_{i=1}^{ps}$ we obtain $\{T(\boldsymbol{\Psi}_{\mu_i})\}_{i=1}^{ps}$. At this point, the standard interpolation (for example using Lagrange interpolation) is performed using $\{T(\boldsymbol{\Psi}_{\mu_i})\}_{i=1}^{ps}$, the projections on the tangent space, to obtain $T(\boldsymbol{\Psi}_{\mu^*})$. Now, $T(\boldsymbol{\Psi}_{\mu^*})$ needs to be projected back to the manifold $\mathcal{G}$. This can be done using an *exponential map* defined as

$$\boldsymbol{\Psi}_{\mu^*} = \left(\boldsymbol{\Psi}_{\tilde{\mu}}\boldsymbol{W}_{\mu^*}\cos(\boldsymbol{\Xi}_{\mu^*}) + \boldsymbol{R}_{\mu^*}\sin(\boldsymbol{\Xi}_{\mu^*})\right)\boldsymbol{W}^T_{\mu^*},$$

where $\boldsymbol{R}_{\tilde{\mu}}$, $\boldsymbol{\Xi}_{\tilde{\mu}}$ and $\boldsymbol{W}^T_{\tilde{\mu}}$ are obtained from the following SVD:

$$T(\boldsymbol{\Psi}_{\mu^*}) = \boldsymbol{R}_{\mu^*}\boldsymbol{\Xi}_{\mu^*}\boldsymbol{W}^T_{\mu^*}.$$

An illustrative implementation of matrix manifold interpolation is shown in Fig. 4.

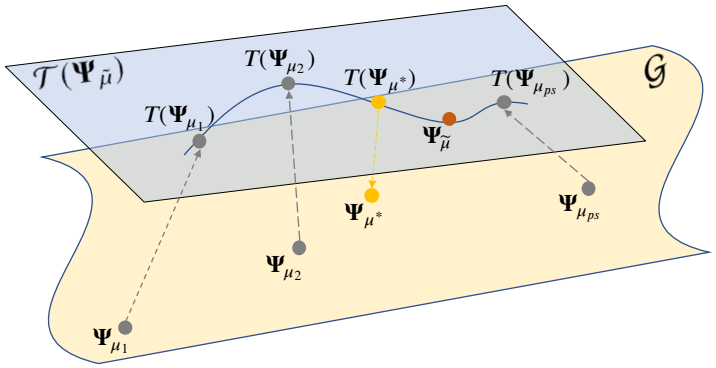

Fig. 4 Interpolation of a set of matrices $\{\boldsymbol{\Psi}_{\mu_i}\}_{i=1}^{ps}$ using the matrix manifold $\mathcal{G}$ and the tangent plane $\mathcal{T}(\boldsymbol{\Psi}_{\tilde{\mu}})$.

Remark 5 The above described manifold based interpolation has been shown to be applicable to the direct interpolation of reduced order system matrices/vectors as well for linear systems [7]. Consider a spatially discretized reduced order parametric linear system

$$\boldsymbol{M}_r(\mu^*)\partial_t\boldsymbol{U}_r(\mu^*) + \boldsymbol{L}_r(\mu^*)\boldsymbol{U}_r(\mu^*) = \boldsymbol{F}_r(\mu^*).$$

If $\{\boldsymbol{M}_r(\mu_i)\}_{i=1}^{ps}$, $\{\boldsymbol{L}_r(\mu_i)\}_{i=1}^{ps}$ and $\{\boldsymbol{F}_r(\mu_i)\}_{i=1}^{ps}$ are obtained *offline*, $\boldsymbol{M}_r(\mu^*)$, $\boldsymbol{L}_r(\mu^*)$ and $\boldsymbol{F}_r(\mu^*)$ can be obtained during the *online* phase using the manifold based interpolation. This ensures that the key properties, e.g., symmetric positive definiteness (SPD), is preserved after the interpolation. This direct interpolation is more efficient than first finding the interpolated basis $\boldsymbol{\Psi}_{\mu^*}$ and then finding the reduced matrix $\boldsymbol{X}_r(\mu^*)$ using $\boldsymbol{\Psi}^T_{\mu^*}\boldsymbol{X}\boldsymbol{\Psi}_{\mu^*}$. However, this direct interpolation has been shown to work

for linear problems only so far. The appropriate logarithmic and exponential maps to be used to preserve different matrix properties can be found in [7].

6 Machine Learning Based Reduced Order Models

The impact of ML has been profound on scientific computing. In this section, the applications of ML in the context of *projection based* ROMs are explored. However, it is pertinent to mention the natural suitability of ML techniques to develop extremely computationally inexpensive models, even beyond the context of projection based ROMs. A lot of the success of ML techniques can be attributed to their ability to find and learn nonlinear mappings that govern a physical system. For any system, a few key inputs can be selected and a ML technique can be applied to learn the (non)linear mapping that exists between its outputs and the selected inputs. Since the *online (testing)* phase of ML algorithms is very fast, any such application would result in a computationally inexpensive model, i.e., a ROM.

In the context of projection based ROMs, ML techniques have been applied to achieve higher accuracy, improvement in speeds or a combination of both. ML techniques have been applied to obtain nonlinear reduced spaces for ROMs which offer a more compact representation than linear POD spaces. ML techniques have also been used to obtain NIROMs by directly learning the nonlinear evolution of *reduced coordinates*, previously referred to as ROM coefficients in the context of POD. The term *reduced coordinates* is more popular in the literature in the context of ML-based ROMs, and hence, it will be used from here on. ML can also improve the accuracy of the intrusive Galerkin ROMs by providing closure models or corrections based on finer solutions. Purely data-driven ML techniques can be very data hungry. In order to reduce the reliance on data, and improve the generalization of the ML models, physics has been embedded in ML techniques and then applied to reduced order modeling. ML has even been used for system identification to discover simple equations for the evolution of reduced coordinates. Let us describe the state-of-art in the above-mentioned application domains.

6.1 Nonlinear Dimension Reduction

POD (or DMD) provides modes that approximate a linear subspace. However, the evolution of many dynamical systems lies in nonlinear spaces. The linear approximation can lead to two issues. First, the POD modes might be unable to capture highly nonlinear phenomena. Second, in the case that the linear representation can capture the dynamics with reasonable accuracy, using POD may require more *reduced coordinates* than the nonlinear representation. So, instead of the linear mapping provided by the POD (13), a more robust mapping would be using a nonlinear function $\vartheta_D : \mathbb{R}^r \rightarrow \mathbb{R}^{np}$ given by

$$\boldsymbol{U} \approx \boldsymbol{U}_D = \vartheta_D(\boldsymbol{U}_r), \tag{36}$$

where $\boldsymbol{U}_D \in \mathbb{R}^{np}$ is the mapped value and $\boldsymbol{U}$ is the FOM solution. This nonlinear mapping can be achieved using *autoencoders* (AEs) [21].

AEs are artificial neural networks (ANNs) used widely for dimension reduction. The simplest AE, called undercomplete AE, consists of input and output layers of the same size as the size of the FOM, np in this case. Furthermore, it has a *bottleneck* layer in the middle of the desired size r, the same as the size of the reduced space. The architecture of an undercomplete AE is shown in Fig. 5. In general, AEs are quite deep, i.e., they consist of many layers, and use nonlinear activation functions.

Based on the task performed, an AE can be subdivided into two sub-parts: an *encoder* and a *decoder*. The encoder compresses the high-dimensional data successively through its many layers to produce the low-dimensional representation. The encoder can be represented by a function $\vartheta_E : \mathbb{R}^{np} \rightarrow \mathbb{R}^r$ as

$$\boldsymbol{U}_r = \vartheta_E(\boldsymbol{U}).$$

The decoder reproduces the high dimensional representation from the low dimensional one as per the mapping (36). To train an AE, $\boldsymbol{U}$ is given both as the input and the desired output and the loss function $\|\boldsymbol{U}_D - \boldsymbol{U}\|_2^2$ is minimized, i.e., AE as a whole is expected to behave like an identity map. Interestingly, the optimal solution using the encoder and decoder of single layers with linear activation functions is shown to be closely related to POD [20], i.e., POD can be considered to be a linear AE. AEs have been used with simple feedforward [102], as well as convolutional [69, 92, 108] layers and have shown performance enhancement as compared to the linear dimension reduction techniques. Furthermore, *time-lagged* AEs have been shown to capture the slowly evolving dynamics of chemical processes with higher precision [142].

An important issue of AEs is that they do not provide a systematic way of determining the *suitable* dimension of the reduced space as they do not provide hierarchical reduced coordinates, as POD does based on the RIC index. The number of reduced coordinates needs to be provided a priori to an AE as the size of the bottleneck. A smaller number of reduced coordinates cannot be selected as the coordinates are not distributed hierarchically and each coordinate may correspond to roughly the same RIC. A trial-and-error approach can be used to find the optimal dimension of the reduced space, but this is not an efficient approach. *Variational autoencoders* [90] can resolve this issue and provide a parsimonious set of the reduced coordinates. In [58], β-Variational autoencoders [77] uses β as a hyperparameter to promote the sparsity of reduced coordinates by deactivating the unimportant nodes of the bottleneck. β-Variational autoencoders were shown to represent $\sim 90\%$ of the energy for the problem of a flow over an urban environment, using five modes only, in contrast to just $\sim 30\%$ captured by the first five POD modes. AEs have also been applied to discover nonlinear coordinate spaces for DMD [99, 112, 132].

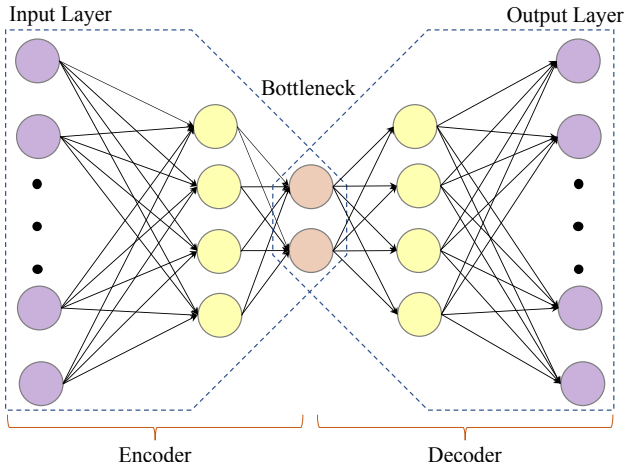

Fig. 5 Architecture of an undercomplete autoencoder with encoder-decoder parts. The number of layers and the size of the bottleneck is set to three and two, respectively, for illustration purposes.

6.2 Machine Learning Based Non-Intrusive Reduced Order Models

As discussed in Section 4.1, the non-intrusive approach to reduced order modeling relies on modeling the dynamics of reduced coordinates using data only, i.e., without accessing the governing equations. ML techniques, like ANNs, can be used to learn the nonlinear dynamics of the reduced coordinates to get a NIROM. A generic schematic of such an approach is shown in Fig. 6. Once the reduced representation is obtained, using POD or AEs, a trained ML model ϑ_{ML} can be used to evolve the reduced coordinates from $\boldsymbol{U}_r^n$ to $\boldsymbol{U}_r^{n+1}$, where n is the time-step counter. Inputs additional to $\boldsymbol{U}_r^n$ can also be provided to ϑ_{ML} to better learn the mapping $\boldsymbol{U}_r^n \rightarrow \boldsymbol{U}_r^{n+1}$.

ML-based NIROMs have been successfully used for a variety of applications. Deep *feedforward neural networks* (FNNs), combined with POD for dimensionality reduction, were applied to get accurate results for the differentially heated cavity flow at various Rayleigh numbers in [114]. The *least squares support vector machine* [131] was used in [41] to relate reduced coordinates with the applied excitations for predicting hypersonic aerodynamic performance. FNN were used to develop a NIROM for the industrial thermo-mechanical phenomena arising in blast furnace hearth walls in [129]. The *multi-output support vector machine* [145] was used to model the dynamics of POD coefficients to predict the stress and displacement for geological and geotechnical processes in [148].

Long short-term memory (LSTM) [78] and its variant *bidirectional long short-term memory* (BiLSTM) [71] *neural networks* (NNs) have a memory effect and can capture sequences like the time evolution of a process with higher accuracy than a FNN. LSTM/BiLSTMs accepts a history of q time-steps, $\{\boldsymbol{U}_r^n, \boldsymbol{U}_r^{n-1}, ..., \boldsymbol{U}_r^{n-q+1}\}$, to predict the future value $\boldsymbol{U}_r^{n+1}$. Thus

$$\boldsymbol{U}_r^{n+1} = \vartheta_{LSTM}(\boldsymbol{U}_r^n, \boldsymbol{U}_r^{n-1}, ..., \boldsymbol{U}_r^{n-q+1}).$$

LSTM and BiLSTM NNs have been widely used to predict the temporal evolution of systems based on the past values. LSTM and BiLSTM NNs were used to model isotropic and magnetohydrodynamic turbulence in [103], where the *Hurst Exponent* [81] was employed to study and quantify the memory effects captured by the LSTM/BiLSTM NNs. LSTM NNs were used in [137] to predict high-dimensional chaotic systems and were shown to outperform Gaussian processes. The improved performance of LSTM NNs was also shown for reduced order modeling of near-wall turbulence as compared to FNNs in [130]. Finally, LSTM NNs were also used to model a synthetic jet and three dimensional flow in the wake of a cylinder in [1].

Training ML algorithms in general, and LSTM/BiLSTM NNs in particular, can be very computationally demanding. *Transfer learning* can be used to speed up the training phase. Instead of initializing the weights of a network randomly, transfer learning relies on using weights of a network previously trained for a closely related problem for initialization. Providing better initial weights allows the training to converge faster to the optimal weights. Transfer learning was used to speed-up the training of LSTM and BiLSTM NNs modeling the three-dimensional turbulent wake of a finite wall-mounted square cylinder in [146]. The flow was analyzed on 2D planes at different heights, each modeled using a separate LSTM/BiLSTM NN. After the first LSTM/BiLSTM NN was trained, the others were initialized using its weights, as the flow in different planes is correlated.

Gaussian process regression [117] can be used to build NIROMs alongside providing uncertainty quantification. Gaussian process regression has been used to develop NIROM for shallow water equations [101], chaotic systems like climate forecasting models [138] and nonlinear structural problems [73]. In the domain of unsupervised learning, *cluster reduced order modeling* has been applied to mixing layers [86]. The cluster reduced order modeling groups the ensemble of data (snapshots) into clusters and the transitions between the states are dynamically modeled using a Markov process.

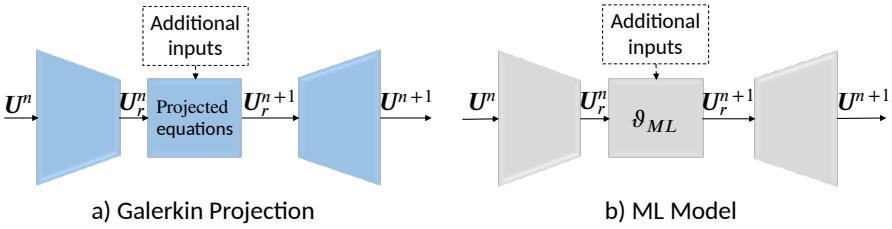

Fig. 6 Non-intrusive ROM obtained by replacing the Galerkin projection (a) with a ML approach (b) to model the dynamics of the reduced coordinates $\boldsymbol{U}_r$.

6.3 Closure Modeling

Another common application of ML techniques is to provide closure modeling in the context of intrusive ROMs. Galerkin projection is the most commonly used intrusive technique to solve for the reduced coordinates. However, performing the Galerkin projection can lead to inaccuracy and instability issues. These issue were highlighted in Section 3.3, where the primary concern was to improve the *stability* of ROMs. Here, the closure modeling of ROMs is presented with the primary objective of improving their *accuracy*. Note that the stabilization and closure modeling are related, yet they are distinct issues [13].

Let us present the closure problem in the context of truncated ROMs. Let us suppose that numerical simulations are performed to gather snapshots $\boldsymbol{U} \in \mathbb{R}^{np}$ for ns time-steps, grouped together to form a snapshot matrix $\boldsymbol{S} \in \mathbb{R}^{np \times ns}$. Let the snapshots belong to the space $\mathcal{B}$ with $\boldsymbol{\Psi} \in \mathbb{R}^{np \times ns}$ being the basis of $\mathcal{B}$. To obtain a computationally efficient ROM, the basis $\boldsymbol{\Psi}$ is truncated to r modes. This results in a truncated basis $\hat{\boldsymbol{\Psi}} \in \mathbb{R}^{np \times r}$ for the *resolved* reduced space and a basis $\widetilde{\boldsymbol{\Psi}} \in \mathbb{R}^{np \times (ns-r)}$ for the *unresolved* reduced space. If we assume for a moment that ns is large enough ($ns \geq np$), we may consider that $\mathcal{B} = \mathcal{B}_f$ and thus, $\boldsymbol{U}$ can be written as

$$\boldsymbol{U} = \hat{\boldsymbol{\Psi}} \boldsymbol{U}_r + \widetilde{\boldsymbol{\Psi}} \widetilde{\boldsymbol{U}}, \tag{37}$$

where $\boldsymbol{U}_r \in \mathbb{R}^r$ is the vector of reduced coordinates captured by the reduced space and $\widetilde{\boldsymbol{U}} \in \mathbb{R}^{(np-r)}$ contains the unresolved coordinates.

Let us assume that (3) represents the semi-discretized (in space) form of the governing equations describing the behavior of $\boldsymbol{U}$. A Galerkin projection of (3) onto resolved and unresolved spaces, using $\hat{\boldsymbol{\Psi}}$ and $\widetilde{\boldsymbol{\Psi}}$, respectively, along with using (37), results in the following system:

$$\begin{bmatrix} \partial_t \boldsymbol{U}_r \\ \partial_t \widetilde{\boldsymbol{U}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{G}(\boldsymbol{U}_r, \widetilde{\boldsymbol{U}}) \\ \widetilde{\boldsymbol{G}}(\boldsymbol{U}_r, \widetilde{\boldsymbol{U}}) \end{bmatrix},$$

where both, $\boldsymbol{G} \in \mathbb{R}^r$ and $\widetilde{\boldsymbol{G}} \in \mathbb{R}^{(np-r)}$, are functions of $\boldsymbol{U}_r$ and $\widetilde{\boldsymbol{U}}$. The objective is to solve for the dynamics of the resolved part $\boldsymbol{U}_r$ only, i.e.,

$$\partial_t \boldsymbol{U}_r = \boldsymbol{G}(\boldsymbol{U}_r, \widetilde{\boldsymbol{U}}).$$

Using the truncated basis to build the ROM implicitly implies, abusing of the notation,

$$\partial_t \boldsymbol{U}_r = \boldsymbol{G}(\boldsymbol{U}_r, \boldsymbol{0}) = \boldsymbol{G}(\boldsymbol{U}_r),$$

which is not true in the nonlinear cases, as the behavior of the resolved scales is governed by their interaction with the unresolved ones as well. So, it is desired to model this interaction as a term $\boldsymbol{C}(\boldsymbol{U}_r)$, which is a function of the resolved scales $\boldsymbol{U}_r$, so that

$$\partial_t \boldsymbol{U}_r = \boldsymbol{G}(\boldsymbol{U}_r) + \boldsymbol{C}(\boldsymbol{U}_r). \tag{38}$$

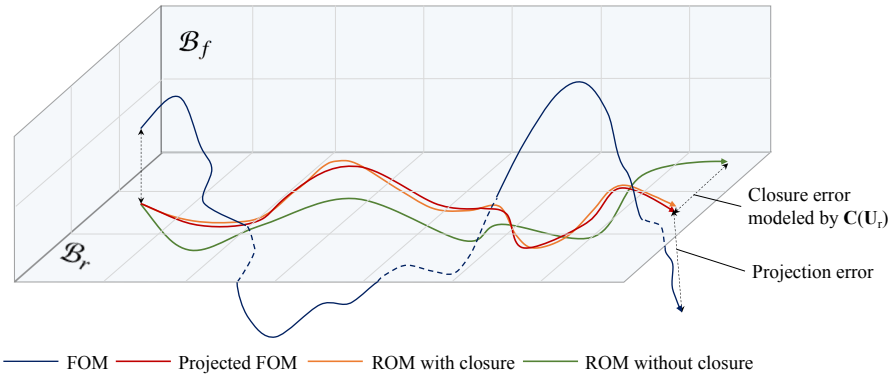

Fig. 7 An illustration of the ROM closure problem. The projection error is due to the use of the space $\mathcal{B}_r \subset \mathcal{B}_f$ for approximating the unknowns. The closure error is due to neglecting the effect of nonlinear interaction on the evolution of the resolved coordinates.

Fig. 7 represents the closure problem graphically. A variety of *conventional* closure models have been developed for ROMs, see [4] for a review and [141] for a comparison of results obtained using popular conventional techniques. Lately, ML techniques have been widely used to obtain the closure models for ROMs. In [125], a single layer FNN, trained using *Bayesian regularization* and *extreme learning machine* approaches, was used to model ROM closure terms for flow problems governed by viscous Burgers equations. The ROM closure terms were modeled as a function of the Reynolds number and resolved reduced coordinates. An extreme learning approach was also used in [124] to determine *eddy viscosity* for a *LES-inspired* closure model. An *uplifting ROM* with closure was propsoed in [3]. LSTMs were used to provide the closure term, as well as to determine the reduced coordinates $\widetilde{\boldsymbol{U}}$ of the unresolved space. Since the basis $\widetilde{\boldsymbol{\Psi}}$ was already known via POD, (37) was used to approximate $\boldsymbol{U}$. A similar approach was used in [2] to develop a closure model for pressure modes to accurately predict the hydrodynamic forces. A *residual neural network*, a hundreds of layers deep FNN, was used to develop a closure model for 1D Burgers equation in [144] .

ROM closure based on *Mori-Zwanzig formalism* [106, 150] are also popular. Such approaches, in general, use closure models consisting of two terms

$$\text{closure term} = \text{memory integral} + \text{contribution of initial conditions}, \tag{39}$$

where the *memory integral* is a non-Markovian [2] term and takes into account the contribution of the past values of the resolved coordinates $\boldsymbol{U}_r$ to model the unresolved ones $\widetilde{\boldsymbol{U}}$. This memory integral is very computationally expensive to compute. To evaluate it efficiently, neural closure models using *neural delay differential equations* were proposed in [149]. The number of past values required to accurately determine

[2] A non-Markovian term implies that the future state depends not only on the current values, but also on the past value, i.e., such processes have memory effects of the past values.

the closure term was also determined. A conditioned LSTM NN was used to model the memory term in [140]. To ensure the computational efficiency, the authors further used explicit time integration of the memory term, while using implicit integration for the remaining terms of the discrete system. The Mori-Zwanzig formalism and LSTM NNs were shown to have a natural analogy between them in [100]. This analogy was also used to develop a systematic approach for constructing memory based ROMs. Recently, *reinforcement learning* techniques have also been applied to build unsupervised reward-oriented closure models for ROMs [24, 126].

6.4 Correction Based on Fine Solutions

Reduced order modeling errors can be improved by introducing a correction term in the *fully discrete* system, i.e., discretized in both space and time, and a general concept to correct fully discrete reduced order problems based on the knowledge of *fine* solutions can be developed. A fine solution is any solution that is considered more accurate than the ROM solution. This is applicable to projection based ROMs, as well as to the cases in which ROM represents a model with a coarser spatial discretization or a larger time step.

Let $\boldsymbol{U}_c$ be the solution of the *coarse* ROM system given by

$$\boldsymbol{A}\boldsymbol{U}_c = \boldsymbol{R}. \tag{40}$$

Let the fine solution $\boldsymbol{U}_f$ be also available for the given problem. Let us assume that the *projection* of the fine solution onto the coarse solution space, denoted by $\boldsymbol{U}^{cf}$, is the best possible coarse solution. In this case, a correction vector $\boldsymbol{C}$ can be added to modify system (40) to obtain a new system

$$\boldsymbol{A}\boldsymbol{U}^{cf} + \boldsymbol{C} = \boldsymbol{R}, \tag{41}$$

with $\boldsymbol{U}^{cf}$ as its solution. When the fine solution is not known, $\boldsymbol{C}$ needs to be modeled as a function of the coarse solution, i.e., $\boldsymbol{C} = \boldsymbol{C}(\boldsymbol{U}_c)$.

A correction term using a *least-squares* (LS) approach was proposed for POD-ROM in [17]. Special considerations regarding gathering training data and using the appropriate initial conditions were also addressed with least-squares providing a linear model of the correction term. A nonlinear correction term was determined using ANN in [18]. The correction term was determined for a coarse mesh based ROM using the solution of an AMR based FOM, and applied to fluid, structure and FSI problems . A comparison of linear least-squares and nonlinear ANNs based corrections was carried out for the wave equation in [60]. A nonlinear ANN based correction for POD-ROM was used in [54]. Different combination of features to be provided as the inputs to the ANNs were evaluated to develop an accurate model while minimizing the complexity. The implicit and explicit treatment of the ANN based correction was also evaluated. It was shown that the ROM was able to produce good results for parametric interpolation, as well as temporal extrapolation. All of

the above mentioned works relied on significantly less training data as compared to NIROMs. A training-free correction was further proposed in [19] to account for the loss of information due to the adaptive coarsening of the coarse mesh based ROM. The correction was based solely on the data generated within the same simulation and did not require any *external* data.

Remark 6 The *correction based on fine solutions* discussed in Section 6.4 and the *closure modeling* discussed in Section 6.3 are similar to some extent. However, there is a difference in their motivation, as well as the accepted definition in the literature. Closure modeling for ROMs is understood to account for the error generated due to the Galerkin projection, i.e., spatial discretization as given by (38). On the other hand, the correction based on fine solutions works by introducing a correction at the fully discrete level given by (41). Hence, the two approaches have been discussed separately.

6.5 Machine Learning Applied to Parametric Reduced Order Models

Just as ML can make non-parametric ROMs robust by offering a non-intrusive determination of reduced coordinates and/or obtaining a nonlinear approximation to the reduced solution space, it offers the same for the parametric ROMs. One way of applying ML to facilitate parametric reduced order modeling is to develop a non-intrusive model of the reduced coordinates as a function of the parameter μ, possibly including time, to get

$$\boldsymbol{U}_r(t,\mu) = \vartheta_{ML}(t,\mu),$$

where ϑ_{ML} is the desired mapping capturing the parametric dependency as well. Here again, it is assumed that $\mu \in \mathcal{D} \subset \mathbb{R}$ without loss of generality, as explained in Section 5. The dynamics of the reduced coordinates can be modeled using gaussian process regression [73, 74, 89] or ANNs [17, 54, 60, 75, 139] trained using the parameter value as an additional input. ANNs have also been used to predict results for times beyond the training time interval [54, 107].

The set of parameter values for which the given system's behavior is *similar*, such that it can be captured by the same basis, is not always obvious. ML-based *clustering* algorithms can help in grouping this *similar* behavior and relating it with appropriate parameter values. A shock sensor and clustering algorithm was used in [56] to decompose the domain, used for aerodynamic simulations, into regions with and without shocks. Accordingly, the snapshots were also decomposed into the two groups and two separate ROMs were built for shock and shock-free subdomains, leading to improved results. Optimal trees [26] were used to build an *interpretable classifier* for unmanned air vehicle operation conditions in [88]. During the operation, sensors provided the data related to the operating conditions. This data was then used by the classifier to suggest the relevant ROM to be used from the library of ROMs prepared in the offline stage.

ML has also been used to obtain nonlinear solution manifolds for parametric ROMs [92]. Deep *convolutional auto-encoders* were used in [69] to generate a reduced parametric trial manifold. It was then combined with a LSTM NN to model the dynamics of the reduced coordinates. Nonlinear reduced parametric manifolds were also learned using AEs in [62] and a variation of it was developed in [63] to speed up the offline training process.

6.6 Physics Informed Machine Learning for Reduced Order Models

ML tools, in general, require a large amount of data to provide accurate results. Moreover, developing a model for generalized cases further increases the data requirement. Generating and storing such amount of data is not always possible. An efficient way of reducing the reliance on data, without affecting the accuracy or generalizability, is to embed physics in the ML tools. Embedding physics in ML is being increasingly used in the broader field of scientific computing; however, limited work is done so far in the domain of reduced order modeling.

One way of employing physics is to use ANNs to solve PDEs directly by minimizing the residual of the governing equations without using any FOM data. Such ANNs are called *physics informed neural networks* [116] and can be used to directly solve the reduced order system without relying on training-testing phases. *Physics reinforced neural networks*, a variation of physics informed neural networks, were proposed in [40] in the context of ROMs. In general, incorporating physics in ML models involves a loss function consisting of two terms: a data-driven loss function $\mathcal{J}_D$ and a physics-based loss function $\mathcal{J}_P$. Let $\boldsymbol{A}$ be a general operator that describes the desired physics such that

$$\boldsymbol{A}(\boldsymbol{U}_r) = \boldsymbol{0}, \tag{42}$$

where $\boldsymbol{A}$ may account for temporal derivatives, nonlinearity, etc. The physics-based loss function $\mathcal{J}_P$ is given by

$$\mathcal{J}_P = \|\boldsymbol{A}(\boldsymbol{U}_{ML})\|_2, \tag{43}$$

where $\boldsymbol{U}_{ML}$ is the output of the ML model. Furthermore, if the snapshots projected onto the reduced space, denoted as $\mathcal{R}(\boldsymbol{U})$, are known, $\mathcal{J}_D$ is given by

$$\mathcal{J}_D = \|\boldsymbol{U}_{ML} - \mathcal{R}(\boldsymbol{U})\|_2. \tag{44}$$

In general, the training phase involves minimizing the mean of (43) and (44) for multiple time-steps and/or parameter values. The total loss function is given by

$$\mathcal{J} = \mathcal{J}_D + \varepsilon \mathcal{J}_P, \tag{45}$$

where ε is a hyperparameter that decides the weightage to be given to adherence to the physics. Physics reinforced neural networks used the residual of the entire governing equation to embed physics. Physics knowledge was embedded in [91] by

incorporating the residual, arising from violating the conservation laws using finite volume discretization, in the loss function. Embedding physics in a data-driven ROM closure model reduced the data requirement by 50% in [105]. The physics was incorporated by requiring some terms of the closure model to be energy dissipating, while others to be energy conserving.

Introducing physics using (45), so that the training phase becomes a *constrained* optimization problem, can be considered as applying *weak constraints* (note that $\mathcal{J}_P = 0$ if $\boldsymbol{U}_{ML}$ is replaced by $\boldsymbol{U}_r$ in Eq. (43)). The violation of the physics leads to a large loss function, and hence, in an effort to minimize the loss function, the ANN tries to adhere to the physics as well. The physics embedded in such a way is prone to be violated in the testing phase when the ANN is exposed to cases beyond the training phase. This is because *architecturally*, the ANN is still unaware of the physics. Furthermore, ε acts as an additional hyperparameter which needs to be tuned.

An alternative strategy is to amend the ANN structurally so that it enforces the physical laws *strongly*. Such an ANN is hoped to be more robust in the testing phase as it is not blind to the physics. Embedded physics in a coarse grained ROM for 3D turbulence via *hard* constraints was achieved in [104]. The divergence free condition was enforced using curl-computing layers which formed a part of the backpropagation process as well. Backpropagation through the curl operator ensured that the network is not *blind* and has intimate knowledge of the constraints through which it must make predictions. Another way of embedding physics in the layers was proposed in [115]. A physics-guided machine learning framework was introduced which *injected* the desired physics in one of the layers of the LSTM NN. By incorporating physics, an ANN applicable to more generalized cases was achieved as compared to the purely data-driven approach.

6.7 Reduced System Identification

ML can also be used to find the equations representing the dynamics of a system using data. An equation comprising of different terms with adjustable coefficients is assumed to model the behavior of a system. Data is then used to find the value of these adjustable coefficients. This technique is known as *system identification* and is of particular interest in two scenarios. First, if the equations are not known, as in the case of modeling climate, epidemiology, neuroscience, etc. Second, when the equations describing the behavior of the reduced coordinates are required. The resulting equation based representation of a system provides generalizability and interpretability, not achievable by simply constructing a regression model based on data. System identification is a broad field and many techniques have been applied in this context, see [95] and [84] for details.

Reduced system identification aims to obtain *sparse* equations (consisting of a few simple terms) to describe the evolution of reduced coordinates of a projection based ROM. The sparse identification of nonlinear dynamics (SINDy) [28] algorithm can

be used to get a simplistic dynamic model for the reduced coordinates. A library of simple nonlinear terms, like polynomials or trigonometric functions, is provided. SINDy then tries to find a mapping for the provided input-output data using the minimum number of terms of the library, thus providing a minimalistic interpretable model offering a balance of efficiency and accuracy.

SINDy has been applied to recover models for a variety of flow behaviors including shear layers, laminar and turbulent wakes, and convection [33, 55, 96, 97]. The vortex shedding behind a cylinder, for example, can be captured using three modes only [109], first two POD modes and a *shift mode* as

$$\begin{aligned} \partial_t U_{r1} &= \mu U_{r1} - U_{r2} - U_{r1} U_{r3}, \\ \partial_t U_{r2} &= U_{r1} + \mu U_{r2} - U_{r2} U_{r3}, \\ \partial_t U_{r3} &= U_{r1}^2 + U_{r2}^2 - U_{r3}, \end{aligned} \tag{46}$$

where U_{r1}, U_{r2} and U_{r3} are the reduced coordinates related to first two POD modes and the shift mode. SINDy was able to recover this minimalistic model using data, identifying the dominant terms and the associated coefficients correctly [28]. SINDy was also combined with an autoencoder to find the low dimensional nonlinear representation, as well as to model the dynamics of the corresponding reduced coordinates [36]. To improve the performance of SINDy, physics was also embedded in it in the form of symmetry in [72] and of conservation laws in [97].

7 Concluding Remarks

Reduced order modeling can significantly accelerate numerical simulations. Thus ROMs permit the solution of optimization and control problems, requiring many simulation runs, at a reasonable computational expense. Traditionally, POD-G has been the most popular approach used to obtain ROMs. Overtime, additional ingredients have been incorporated to improve the performance of these ROMs. For example, hyperreduction techniques have been developed to efficiently deal with nonlinear terms, stabilization techniques have been incorporated to deal with instabilities, DMD has been introduced as a non-intrusive variant of POD-G and suitable basis generation methods have been developed for parametric ROMs.

Like any other field, availability of data and open-access to ML models has led to the increased popularity of ML techniques for reduced order modeling. ML has been used to obtain ROMs with lower computational expense and enhanced accuracy as compared to the conventional techniques. Not only this, but ML has also opened new avenues of reduced order modeling which the conventional techniques were unable to pursue, nonlinear dimension reduction being one of them. The reduced order modeling community has been using ML in all the possible ways. On one hand, ML has been used to improve primarily *physics based* ROMs by providing ML-based closure models and correction terms. On the other hand, physics has been embedded in primarily *ML-based* ROMs to improve their performance. Finally, NIROMs have

been developed based *entirely* on ML. Thus, a range of reduced order modeling techniques are at disposal with purely conventional techniques at one end of the spectrum to purely ML techniques at the other end, with the hybrid techniques lying in-between.

References

[1] R. Abadía-Heredia et al. "A Predictive Hybrid Reduced Order Model Based on Proper Orthogonal Decomposition Combined with Deep Learning Architectures". In: *Expert Systems with Applications* 187 (2022), p. 115910.

[2] H. F. Ahmed et al. "Machine Learning–Based Reduced-Order Modeling of Hydrodynamic Forces Using Pressure Mode Decomposition". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 235.16 (2021), pp. 2517–2528.

[3] S. E. Ahmed et al. "A Long Short-Term Memory Embedding for Hybrid Uplifted Reduced Order Models". In: *Physica D: Nonlinear Phenomena* 409 (2020), p. 132471.

[4] S. E. Ahmed et al. "On Closures for Reduced Order Models—A Spectrum of First-Principle to Machine-Learned Avenues". In: *Physics of Fluids* 33.9 (2021), p. 091301.

[5] I. Akhtar, J. Borggaard, and A. Hay. "Shape Sensitivity Analysis in Flow Models Using a Finite-Difference Approach". In: *Mathematical Problems in Engineering* 2010 (2010).

[6] A. Alla and J. N. Kutz. "Nonlinear Model Order Reduction via Dynamic Mode Decomposition". In: *SIAM Journal on Scientific Computing* 39.5 (2017), B778–B796.

[7] D. Amsallem and C. Farhat. "An Online Method for Interpolating Linear Parametric Reduced-Order Models". In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2169–2198.

[8] D. Amsallem and C. Farhat. "Stabilization of Projection-Based Reduced-Order Models". In: *International Journal for Numerical Methods in Engineering* 91.4 (2012), pp. 358–377.

[9] S. S. An, T. Kim, and D. L. James. "Optimizing Cubature for Efficient Integration of Subspace Deformations". In: *ACM Transactions on Graphics* 27.5 (2008), 165:1–165:10.

[10] H. Antil, M. Heinkenschloss, and D. C. Sorensen. "Application of the Discrete Empirical Interpolation Method to Reduced Order Modeling of Nonlinear and Parametric Systems". In: *Reduced Order Methods for Modeling and Computational Reduction*. Ed. by A. Quarteroni and G. Rozza. MS&A - Modeling, Simulation and Applications. Cham: Springer International Publishing, 2014, pp. 101–136.

[11] E. Arian, M. Fahl, and E. W. Sachs. *Trust-Region Proper Orthogonal Decomposition for Flow Control*. Tech. rep. Institute for Computer Applications in Science and Engineering, Hampton VA, 2000.

[12] P. Astrid et al. "Missing Point Estimation in Models Described by Proper Orthogonal Decomposition". In: *IEEE Transactions on Automatic Control* 53.10 (2008), pp. 2237–2251.

[13] M. Azaïez, T. Chacón Rebollo, and S. Rubino. "A Cure for Instabilities Due to Advection-Dominance in POD Solution to Advection-Diffusion-Reaction Equations". In: *Journal of Computational Physics* 425 (2021), p. 109916.

[14] J. Baiges and R. Codina. "A Variational Multiscale Method with Subscales on the Element Boundaries for the Helmholtz Equation". In: *International Journal for Numerical Methods in Engineering* 93.6 (2013), pp. 664–684.

[15] J. Baiges, R. Codina, and S. Idelsohn. "A Domain Decomposition Strategy for Reduced Order Models. Application to the Incompressible Navier–Stokes Equations". In: *Computer Methods in Applied Mechanics and Engineering* 267 (2013), pp. 23–42.

[16] J. Baiges, R. Codina, and S. Idelsohn. "Explicit Reduced-Order Models for the Stabilized Finite Element Approximation of the Incompressible Navier–Stokes Equations". In: *International Journal for Numerical Methods in Fluids* 72.12 (2013), pp. 1219–1243.

[17] J. Baiges, R. Codina, and S. Idelsohn. "Reduced-Order Subscales for POD Models". In: *Computer Methods in Applied Mechanics and Engineering* 291 (2015), pp. 173–196.

[18] J. Baiges et al. "A Finite Element Reduced-Order Model Based on Adaptive Mesh Refinement and Artificial Neural Networks". In: *International Journal for Numerical Methods in Engineering* 121.4 (2020), pp. 588–601.

[19] J. Baiges et al. "An Adaptive Finite Element Strategy for the Numerical Simulation of Additive Manufacturing Processes". In: *Additive Manufacturing* 37 (2021), p. 101650.

[20] P. Baldi and K. Hornik. "Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima". In: *Neural Networks* 2.1 (1989), pp. 53–58.

[21] D. H. Ballard. "Modular Learning in Neural Networks". In: *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*. AAAI'87. Seattle, Washington: AAAI Press, 1987, pp. 279–284.

[22] F. Ballarin et al. "Supremizer Stabilization of POD–Galerkin Approximation of Parametrized Steady Incompressible Navier–Stokes Equations". In: *International Journal for Numerical Methods in Engineering* 102.5 (2015), pp. 1136–1161.

[23] M. Barrault et al. “An ‘Empirical Interpolation’Method: Application to Efficient Reduced-Basis Discretization of Partial Differential Equations”. In: *Comptes Rendus Mathematique* 339.9 (2004), pp. 667–672.

[24] M. Benosman, A. Chakrabarty, and J. Borggaard. “Reinforcement Learning-based Model Reduction for Partial Differential Equations”. In: *IFAC-PapersOnLine*. 21st IFAC World Congress 53.2 (2020), pp. 7704–7709.

[25] M. Bergmann, L. Cordier, and J.-P. Brancher. “Drag Minimization of the Cylinder Wake by Trust-Region Proper Orthogonal Decomposition”. In: *Active Flow Control*. Springer, 2007, pp. 309–324.

[26] D. Bertsimas and J. Dunn. “Optimal Classification Trees”. In: *Machine Learning* 106.7 (2017), pp. 1039–1082.

[27] A. N. Brooks and T. J. R. Hughes. “Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations”. In: *Computer Methods in Applied Mechanics and Engineering* 32.1 (1982), pp. 199–259.

[28] S. L. Brunton, J. L. Proctor, and J. N. Kutz. “Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems”. In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937.

[29] S. L. Brunton et al. “Chaos as an Intermittently Forced Linear System”. In: *Nature Communications* 8.1 (2017), p. 19.

[30] T. Bui-Thanh, K. Willcox, and O. Ghattas. “Model Reduction for Large-Scale Systems with High-Dimensional Parametric Input Space”. In: *SIAM Journal on Scientific Computing* 30.6 (2008), pp. 3270–3288.

[31] S. Buoso et al. “Stabilized Reduced-Order Models for Unsteady Incompressible Flows in Three-Dimensional Parametrized Domains”. In: *Computers & Fluids* 246 (2022), p. 105604.

[32] J. Burkardt, M. Gunzburger, and H.-C. Lee. “POD and CVT-based Reduced-Order Modeling of Navier–Stokes Flows”. In: *Computer methods in applied mechanics and engineering* 196.1-3 (2006), pp. 337–355.

[33] J. L. Callaham et al. “An Empirical Mean-Field Model of Symmetry-Breaking in a Turbulent Wake”. In: *Science Advances* 8.19 (2022), eabm4786.

[34] K. Carlberg, M. Barone, and H. Antil. “Galerkin v. Least-Squares Petrov–Galerkin Projection in Nonlinear Model Reduction”. In: *Journal of Computational Physics* 330 (2017), pp. 693–734.

[35] K. Carlberg, C. Bou-Mosleh, and C. Farhat. “Efficient Non-Linear Model Reduction via a Least-Squares Petrov–Galerkin Projection and Compressive Tensor Approximations”. In: *International Journal for Numerical Methods in Engineering* 86.2 (2011), pp. 155–181.

[36] K. Champion et al. “Data-Driven Discovery of Coordinates and Governing Equations”. In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22445–22451.

[37] A. Chatterjee. “An Introduction to the Proper Orthogonal Decomposition”. In: *Current Science* 78.7 (2000), pp. 808–817.

[38] S. Chaturantabut and D. C. Sorensen. “Nonlinear Model Reduction via Discrete Empirical Interpolation”. In: *SIAM Journal on Scientific Computing* 32.5 (2010), pp. 2737–2764.

[39] K. K. Chen, J. H. Tu, and C. W. Rowley. “Variants of Dynamic Mode Decomposition: Boundary Condition, Koopman, and Fourier Analyses”. In: *Journal of Nonlinear Science* 22.6 (2012), pp. 887–915.

[40] W. Chen et al. “Physics-Informed Machine Learning for Reduced-Order Modeling of Nonlinear Problems”. In: *Journal of Computational Physics* 446 (2021), p. 110666.

[41] Z. Chen, Y. Zhao, and R. Huang. “Parametric Reduced-Order Modeling of Unsteady Aerodynamics for Hypersonic Vehicles”. In: *Aerospace Science and Technology* 87 (2019), pp. 1–14.

[42] F. Chinesta, A. Ammar, and E. Cueto. “Recent Advances and New Challenges in the Use of the Proper Generalized Decomposition for Solving Multidimensional Models”. In: *Archives of Computational Methods in Engineering* 17.4 (2010), pp. 327–350.

[43] F. Chinesta, P. Ladeveze, and E. Cueto. “A Short Review on Model Order Reduction Based on Proper Generalized Decomposition”. In: *Archives of Computational Methods in Engineering* 18.4 (2011), p. 395.

[44] R. Codina. “On Stabilized Finite Element Methods for Linear Systems of Convection–Diffusion-Reaction Equations”. In: *Computer Methods in Applied Mechanics and Engineering* 188.1 (2000), pp. 61–82.

[45] R. Codina. “Stabilization of Incompressibility and Convection through Orthogonal Sub-Scales in Finite Element Methods”. In: *Computer methods in applied mechanics and engineering* 190.13-14 (2000), pp. 1579–1599.

[46] R. Codina. “Stabilized Finite Element Approximation of Transient Incompressible Flows Using Orthogonal Subscales”. In: *Computer methods in applied mechanics and engineering* 191.39-40 (2002), pp. 4295–4321.

[47] R. Codina and J. Baiges. “Finite Element Approximation of Transmission Conditions in Fluids and Solids Introducing Boundary Subgrid Scales”. In: *International Journal for Numerical Methods in Engineering* 87.1-5 (2011), pp. 386–411.

[48] R. Codina, J. Principe, and J. Baiges. “Subscales on the Element Boundaries in the Variational Two-Scale Finite Element Method”. In: *Computer methods in applied mechanics and engineering* 198.5-8 (2009), pp. 838–852.

[49] R. Codina, R. Reyes, and J. Baiges. “A Posteriori Error Estimates in a Finite Element VMS-based Reduced Order Model for the Incompressible Navier-Stokes Equations”. In: *Mechanics Research Communications*. Special Issue Honoring G.I. Taylor Medalist Prof. Arif Masud 112 (2021), p. 103599.

[50] R. Codina et al. “Time Dependent Subscales in the Stabilized Finite Element Approximation of Incompressible Flow Problems”. In: *Computer Methods in Applied Mechanics and Engineering* 196.21-24 (2007), pp. 2413–2430.

[51] R. Codina et al. “Variational Multiscale Methods in Computational Fluid Dynamics”. In: *Encyclopedia of computational mechanics* (2018), pp. 1–28.

[52] N. Dal Santo et al. “An Algebraic Least Squares Reduced Basis Method for the Solution of Nonaffinely Parametrized Stokes Equations”. In: *Computer Methods in Applied Mechanics and Engineering* 344 (2019), pp. 186–208.

[53] T. Daniel et al. “Model Order Reduction Assisted by Deep Neural Networks (ROM-net)”. In: *Advanced Modeling and Simulation in Engineering Sciences* 7.1 (2020), p. 16.

[54] Z. Dar, J. Baiges, and R. Codina. “Artificial neural network based correction models for reduced order models in computational fluid mechanics”. In: *Submitted* (2022).

[55] N. Deng et al. “Low-Order Model for Successive Bifurcations of the Fluidic Pinball”. In: *Journal of Fluid Mechanics* 884 (2020), A37.

[56] R. Dupuis, J.-C. Jouhaud, and P. Sagaut. “Surrogate Modeling of Aerodynamic Simulations for Multiple Operating Conditions Using Machine Learning”. In: *AIAA Journal* 56.9 (2018), pp. 3622–3635.

[57] C. Eckart and G. Young. “The Approximation of One Matrix by Another of Lower Rank”. In: *Psychometrika* 1.3 (1936), pp. 211–218.

[58] H. Eivazi et al. “Towards Extraction of Orthogonal and Parsimonious Non-Linear Modes from Turbulent Flows”. In: *Expert Systems with Applications* 202 (2022), p. 117038.

[59] R. Everson and L. Sirovich. “Karhunen–Loève Procedure for Gappy Data”. In: *JOSA A* 12.8 (1995), pp. 1657–1664.

[60] A. Fabra, J. Baiges, and R. Codina. “Finite Element Approximation of Wave Problems with Correcting Terms Based on Training Artificial Neural Networks with Fine Solutions”. In: *Computer Methods in Applied Mechanics and Engineering* 399 (2022), p. 115280.

[61] C. Farhat, T. Chapman, and 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”. In: *International Journal for Numerical Methods in Engineering* 102.5 (2015), pp. 1077–1110.

[62] S. Fresca, L. Dede’, and A. Manzoni. “A Comprehensive Deep Learning-Based Approach to Reduced Order Modeling of Nonlinear Time-Dependent Parametrized PDEs”. In: *Journal of Scientific Computing* 87.2 (2021), p. 61.

[63] S. Fresca and A. Manzoni. “POD-DL-ROM: Enhancing Deep Learning-Based Reduced Order Models for Nonlinear Parametrized PDEs by Proper Orthogonal Decomposition”. In: *Computer Methods in Applied Mechanics and Engineering* 388 (2022), p. 114181.

[64] B. Galletti et al. “Low-Order Modelling of Laminar Flow Regimes Past a Confined Square Cylinder”. In: *Journal of Fluid Mechanics* 503 (2004), pp. 161–170.

[65] B. García-Archilla, J. Novo, and S. Rubino. “Error Analysis of Proper Orthogonal Decomposition Data Assimilation Schemes with Grad–Div Stabilization for the Navier–Stokes Equations”. In: *Journal of Computational and Applied Mathematics* 411 (2022), p. 114246.

[66] S. Giere and V. John. “Towards Physically Admissible Reduced-Order Solutions for Convection–Diffusion Problems”. In: *Applied Mathematics Letters* 73 (2017), pp. 78–83.

[67] S. Giere et al. “SUPG Reduced Order Models for Convection-Dominated Convection–Diffusion–Reaction Equations”. In: *Computer Methods in Applied Mechanics and Engineering* 289 (2015), pp. 454–474.

[68] B. Glaz, L. Liu, and P. P. Friedmann. “Reduced-Order Nonlinear Unsteady Aerodynamic Modeling Using a Surrogate-Based Recurrence Framework”. In: *AIAA journal* 48.10 (2010), pp. 2418–2429.

[69] F. J. Gonzalez and M. Balajewicz. “Deep Convolutional Recurrent Autoencoders for Learning Low-Dimensional Feature Dynamics of Fluid Systems”. In: *arXiv preprint arXiv:1808.01346* (2018). arXiv: `1808.01346 [physics]`.

[70] W. R. Graham, J. Peraire, and K. Y. Tang. “Optimal Control of Vortex Shedding Using Low-Order Models. Part I—Open-Loop Model Development”. In: *International Journal for Numerical Methods in Engineering* 44.7 (1999), pp. 945–972.

[71] A. Graves and J. Schmidhuber. “Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures”. In: *Neural Networks*. IJCNN 2005 18.5 (2005), pp. 602–610.

[72] Y. Guan, S. L. Brunton, and I. Novosselov. “Sparse Nonlinear Models of Chaotic Electroconvection”. In: *Royal Society Open Science* 8.8 (2021), p. 202367.

[73] M. Guo and J. S. Hesthaven. “Reduced Order Modeling for Nonlinear Structural Analysis Using Gaussian Process Regression”. In: *Computer methods in applied mechanics and engineering* 341 (2018), pp. 807–826.

[74] M. Guo and J. S. Hesthaven. “Data-Driven Reduced Order Modeling for Time-Dependent Problems”. In: *Computer Methods in Applied Mechanics and Engineering* 345 (2019), pp. 75–99.

[75] J. S. Hesthaven and S. Ubbiali. “Non-Intrusive Reduced Order Modeling of Nonlinear Problems Using Neural Networks”. In: *Journal of Computational Physics* 363 (2018), pp. 55–78.

[76] J. S. Hesthaven, G. Rozza, and B. Stamm. *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*. SpringerBriefs in Mathematics. Cham: Springer International Publishing, 2016.

[77] I. Higgins et al. “Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework”. In: *International Conference on Learning Representations*. 2022.

[78] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[79] T. J. R. Hughes et al. “The Variational Multiscale Method—a Paradigm for Computational Mechanics”. In: *Computer Methods in Applied Mechanics and Engineering*. Advances in Stabilized Methods in Computational Mechanics 166.1 (1998), pp. 3–24.

[80] A. Hunter et al. “Reduced-Order Modeling through Machine Learning and Graph-Theoretic Approaches for Brittle Fracture Applications”. In: *Computational Materials Science* 157 (2019), pp. 87–98.

[81] H. E. Hurst. “Long-Term Storage Capacity of Reservoirs”. In: *Transactions of the American Society of Civil Engineers* 116.1 (1951), pp. 770–799.

[82] V. John, B. Moreau, and J. Novo. “Error Analysis of a SUPG-stabilized POD-ROM Method for Convection-Diffusion-Reaction Equations”. In: *Computers & Mathematics with Applications* 122 (2022), pp. 48–60.
[83] L. John Leask. “The Structure of Inhomogeneous Turbulent Flows”. In: *The structure of inhomogeneous turbulent flows* (1967), pp. 166–178.
[84] J.-N. Juang. *Applied System Identification*. Prentice Hall, 1994.
[85] K. Kaheman, S. L. Brunton, and J. N. Kutz. “Automatic Differentiation to Simultaneously Identify Nonlinear Dynamics and Extract Noise Probability Distributions from Data”. In: *Machine Learning: Science and Technology* 3.1 (2022), p. 015031.
[86] E. Kaiser et al. “Cluster-Based Reduced-Order Modelling of a Mixing Layer”. In: *Journal of Fluid Mechanics* 754 (2014), pp. 365–414.
[87] I. Kalashnikova and M. Barone. “Stable and Efficient Galerkin Reduced Order Models for Non-Linear Fluid Flow”. In: *6th AIAA Theoretical Fluid Mechanics Conference*. 2011, p. 3110.
[88] M. G. Kapteyn, D. J. Knezevic, and K. Willcox. “Toward Predictive Digital Twins via Component-Based Reduced-Order Models and Interpretable Machine Learning”. In: *AIAA Scitech 2020 Forum*. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, 2020.
[89] M. Kast, M. Guo, and J. S. Hesthaven. “A Non-Intrusive Multifidelity Method for the Reduced Order Modeling of Nonlinear Problems”. In: *Computer Methods in Applied Mechanics and Engineering* 364 (2020), p. 112947.
[90] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations*. 2013.
[91] K. Lee and K. Carlberg. “Deep Conservation: A Latent-Dynamics Model for Exact Satisfaction of Physical Conservation Laws”. In: *arXiv preprint arXiv:1909.09754* (2020). arXiv: 1909.09754 [physics].
[92] K. Lee and K. T. Carlberg. “Model Reduction of Dynamical Systems on Nonlinear Manifolds Using Deep Convolutional Autoencoders”. In: *Journal of Computational Physics* 404 (2020), p. 108973.
[93] P. LeGresley and J. Alonso. “Airfoil Design Optimization Using Reduced Order Models Based on Proper Orthogonal Decomposition”. In: *Fluids 2000 Conference and Exhibit*. American Institute of Aeronautics and Astronautics, 2000.
[94] J. Li, X. Du, and J. R. R. A. Martins. “Machine Learning in Aerodynamic Shape Optimization”. In: *Progress in Aerospace Sciences* 134 (2022), p. 100849.
[95] L. Ljung. *System Identification: Theory for the User*. 2nd edition. Upper Saddle River, NJ: Pearson, 1998.
[96] J.-C. Loiseau. “Data-Driven Modeling of the Chaotic Thermal Convection in an Annular Thermosyphon”. In: *Theoretical and Computational Fluid Dynamics* 34 (2020), pp. 339–365.
[97] J.-C. Loiseau and S. L. Brunton. “Constrained Sparse Galerkin Regression”. In: *Journal of Fluid Mechanics* 838 (2018), pp. 42–67.
[98] D. J. Lucia and P. S. Beran. “Projection Methods for Reduced Order Models of Compressible Flows”. In: *Journal of Computational Physics* 188.1 (2003), pp. 252–280.
[99] B. Lusch, J. N. Kutz, and S. L. Brunton. “Deep Learning for Universal Linear Embeddings of Nonlinear Dynamics”. In: *Nature Communications* 9.1 (2018), p. 4950.
[100] C. Ma, J. Wang, and W. E. “Model Reduction with Memory and the Machine Learning of Dynamical Systems”. In: *Communications in Computational Physics* 25.4 (2019).
[101] R. Maulik et al. “Latent-Space Time Evolution of Non-Intrusive Reduced-Order Models Using Gaussian Process Emulation”. In: *Physica D: Nonlinear Phenomena* 416 (2021), p. 132797.
[102] M. Milano and P. Koumoutsakos. “Neural Network Modeling for Near Wall Turbulent Flow”. In: *Journal of Computational Physics* 182.1 (2002), pp. 1–26.
[103] A. T. Mohan and D. V. Gaitonde. “A Deep Learning Based Approach to Reduced Order Modeling for Turbulent Flow Control Using LSTM Neural Networks”. In: *arXiv preprint arXiv:1804.09269* (2018). arXiv: 1804.09269 [physics].
[104] A. T. Mohan et al. “Embedding Hard Physical Constraints in Neural Network Coarse-Graining of 3D Turbulence”. In: *arXiv preprint arXiv:2002.00021* (2020). arXiv: 2002.00021 [physics].
[105] M. Mohebujjaman, L. Rebholz, and T. Iliescu. “Physically Constrained Data-Driven Correction for Reduced-Order Modeling of Fluid Flows”. In: *International Journal for Numerical Methods in Fluids* 89.3 (2019), pp. 103–122.
[106] H. Mori. “Transport, Collective Motion, and Brownian Motion*)”. In: *Progress of Theoretical Physics* 33.3 (1965), pp. 423–455.
[107] C. Mou et al. “Data-Driven Variational Multiscale Reduced Order Models”. In: *Computer Methods in Applied Mechanics and Engineering* 373 (2021), p. 113470.
[108] T. Murata, K. Fukami, and K. Fukagata. “Nonlinear Mode Decomposition with Convolutional Neural Networks for Fluid Dynamics”. In: *Journal of Fluid Mechanics* 882 (2020), A13.
[109] B. R. Noack et al. “A Hierarchy of Low-Dimensional Models for the Transient and Post-Transient Cylinder Wake”. In: *Journal of Fluid Mechanics* 497 (2003), pp. 335–363.
[110] B. R. Noack et al. “Recursive Dynamic Mode Decomposition of Transient and Post-Transient Wake Flows”. In: *Journal of Fluid Mechanics* 809 (2016), pp. 843–872.
[111] B. R. Noack et al., eds. *Reduced-Order Modelling for Flow Control*. Vol. 528. CISM International Centre for Mechanical Sciences. Vienna: Springer, 2011.
[112] S. E. Otto and C. W. Rowley. “Linearly Recurrent Autoencoder Networks for Learning Dynamics”. In: *SIAM Journal on Applied Dynamical Systems* 18.1 (2019), pp. 558–593.
[113] P. Pacciarini and G. Rozza. “Stabilized Reduced Basis Method for Parametrized Advection–Diffusion PDEs”. In: *Computer Methods in Applied Mechanics and Engineering* 274 (2014), pp. 1–18.

[114] S. Pawar et al. “A Deep Learning Enabler for Nonintrusive Reduced Order Modeling of Fluid Flows”. In: *Physics of Fluids* 31.8 (2019), p. 085101.
[115] S. Pawar et al. “Model Fusion with Physics-Guided Machine Learning: Projection-based Reduced-Order Modeling”. In: *Physics of Fluids* 33.6 (2021), p. 067123.
[116] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations”. In: *Journal of Computational Physics* 378 (2019), pp. 686–707.
[117] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. 2005.
[118] R. Reyes and R. Codina. “Projection-Based Reduced Order Models for Flow Problems: A Variational Multiscale Approach”. In: *Computer Methods in Applied Mechanics and Engineering* 363 (2020), p. 112844.
[119] R. Reyes et al. “Reduced Order Models for Thermally Coupled Low Mach Flows”. In: *Advanced Modeling and Simulation in Engineering Sciences* 5.1 (2018), pp. 1–20.
[120] G. Rozza, D. B. P. Huynh, and A. Manzoni. “Reduced Basis Approximation and a Posteriori Error Estimation for Stokes Flows in Parametrized Geometries: Roles of the Inf-Sup Stability Constants”. In: *Rozza* (2013).
[121] G. Rozza, T. Lassila, and A. Manzoni. “Reduced Basis Approximation for Shape Optimization in Thermal Flows with a Parametrized Polynomial Geometric Map”. In: *Spectral and High Order Methods for Partial Differential Equations*. Springer, 2011, pp. 307–315.
[122] S. Sahba et al. “Dynamic Mode Decomposition for Aero-Optic Wavefront Characterization”. In: *Optical Engineering* 61.1 (2022), p. 013105.
[123] O. San and T. Iliescu. “A Stabilized Proper Orthogonal Decomposition Reduced-Order Model for Large Scale Quasigeostrophic Ocean Circulation”. In: *Advances in Computational Mathematics* 41.5 (2015), pp. 1289–1319.
[124] O. San and R. Maulik. “Extreme Learning Machine for Reduced Order Modeling of Turbulent Geophysical Flows”. In: *Physical Review E* 97.4 (2018), p. 42322.
[125] O. San and R. Maulik. “Neural Network Closures for Nonlinear Model Order Reduction”. In: *Advances in Computational Mathematics* 44.6 (2018), pp. 1717–1750.
[126] O. San, S. Pawar, and A. Rasheed. “Variational Multiscale Reinforcement Learning for Discovering Reduced Order Closure Models of Nonlinear Spatiotemporal Transport Systems”. In: *arXiv preprint arXiv:2207.12854* (2022). arXiv: 2207.12854 [physics].
[127] P. J. Schmid. “Dynamic Mode Decomposition of Numerical and Experimental Data”. In: *Journal of Fluid Mechanics* 656 (2010), pp. 5–28.
[128] P. J. Schmid, D. Violato, and F. Scarano. “Decomposition of Time-Resolved Tomographic PIV”. In: *Experiments in Fluids* 52.6 (2012), pp. 1567–1579.
[129] N. V. Shah et al. “Finite Element Based Model Order Reduction for Parametrized One-Way Coupled Steady State Linear Thermo-Mechanical Problems”. In: *Finite Elements in Analysis and Design* 212 (2022), p. 103837.
[130] P. A. Srinivasan et al. “Predictions of Turbulent Shear Flows Using Deep Neural Networks”. In: *Physical Review Fluids* 4.5 (2019), p. 054603.
[131] J. A. K. Suykens et al. *Least Squares Support Vector Machines*. WORLD SCIENTIFIC, 2002.
[132] N. Takeishi, Y. Kawahara, and T. Yairi. “Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 1130–1140.
[133] A. Tello and R. Codina. “Field-to-Field Coupled Fluid Structure Interaction: A Reduced Order Model Study”. In: *International Journal for Numerical Methods in Engineering* 122.1 (2021), pp. 53–81.
[134] A. Tello, R. Codina, and J. Baiges. “Fluid Structure Interaction by Means of Variational Multiscale Reduced Order Models”. In: *International Journal for Numerical Methods in Engineering* 121.12 (2020), pp. 2601–2625.
[135] G. Tissot et al. “Model Reduction Using Dynamic Mode Decomposition”. In: *Comptes Rendus Mécanique*. Flow Separation Control 342.6 (2014), pp. 410–416.
[136] J. H. Tu et al. “On Dynamic Mode Decomposition: Theory and Applications”. In: *Journal of Computational Dynamics* 1.2 (Mon Dec 01 01:00:00 CET 2014), pp. 391–421.
[137] P. R. Vlachas et al. “Data-Driven Forecasting of High-Dimensional Chaotic Systems with Long Short-Term Memory Networks”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2213 (2018), p. 20170844.
[138] Z. Y. Wan and T. P. Sapsis. “Reduced-Space Gaussian Process Regression for Data-Driven Probabilistic Forecast of Chaotic Dynamical Systems”. In: *Physica D: Nonlinear Phenomena* 345 (2017), pp. 40–55.
[139] Q. Wang, J. S. Hesthaven, and D. Ray. “Non-Intrusive Reduced Order Modeling of Unsteady Flows Using Artificial Neural Networks with Application to a Combustion Problem”. In: *Journal of computational physics* 384 (2019), pp. 289–307.
[140] Q. Wang, N. Ripamonti, and J. S. Hesthaven. “Recurrent Neural Network Closure of Parametric POD-Galerkin Reduced-Order Models Based on the Mori-Zwanzig Formalism”. In: *Journal of Computational Physics* 410 (2020), p. 109402.
[141] Z. Wang et al. “Proper Orthogonal Decomposition Closure Models for Turbulent Flows: A Numerical Comparison”. In: *Computer Methods in Applied Mechanics and Engineering* 237 (2012), pp. 10–26.
[142] C. Wehmeyer and F. Noé. “Time-Lagged Autoencoders: Deep Learning of Slow Collective Variables for Molecular Kinetics”. In: *The Journal of Chemical Physics* 148.24 (2018), p. 241703.
[143] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. “A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition”. In: *Journal of Nonlinear Science* 25.6 (2015), pp. 1307–1346.

[144] X. Xie, C. Webster, and T. Iliescu. “Closure Learning for Nonlinear Model Reduction Using Deep Residual Neural Network”. In: *Fluids* 5.1 (2020), p. 39.
[145] S. Xu et al. “Multi-Output Least-Squares Support Vector Regression Machines”. In: *Pattern Recognition Letters* 34.9 (2013), pp. 1078–1084.
[146] M. Z. Yousif and H.-C. Lim. “Reduced-Order Modeling for Turbulent Wake of a Finite Wall-Mounted Square Cylinder Based on Artificial Neural Network”. In: *Physics of Fluids* 34.1 (2022), p. 015116.
[147] J. Yvonnet and Q.-C. He. “The Reduced Model Multiscale Method (R3M) for the Non-Linear Homogenization of Hyperelastic Media at Finite Strains”. In: *Journal of Computational Physics* 223.1 (2007), pp. 341–368.
[148] H. Zhao. “A Reduced Order Model Based on Machine Learning for Numerical Analysis: An Application to Geomechanics”. In: *Engineering Applications of Artificial Intelligence* 100 (2021), p. 104194.
[149] Q. Zhu, Y. Guo, and W. Lin. “Neural delay differential equations”. In: *The International Conference on Learning Representations*. 2021, p. 20.
[150] R. Zwanzig. “Ensemble Method in the Theory of Irreversibility”. In: *The Journal of Chemical Physics* 33.5 (1960), pp. 1338–1341.