Space Weather Programming Summer School – Lecture on August 3, 2023

Data-driven Model Reduction

Lecturer: Boris Kramer, University of California San Diego

This module will cover nonlinear data-driven model reduction via operator inference. Specifically, we will cover:

- The definition and computation of the proper orthogonal decomposition (POD) basis.
- Method of snapshots for high-dimensional data
- Operator inference (OpInf) model learning
- Properties of the OpInf reduced-order models.

1 Proper Orthogonal Decomposition (POD)

Proper Orthogonal Decomposition (POD) is a data reduction technique, which extracts dominant structures from experimental or simulated data. POD determines the optimal basis for representing a given dataset with respect to the mean squared error.[1] Based on the extracted coherent structures, POD has been employed to produce ROMs via classical Galerkin projection, which have been used for simulation, design, control and uncertainty quantification. POD is not limited to a certain model structure, and can apply to general nonlinear systems. As Volkwein [11, p.3] notes "*the POD method is a universal tool that is applicable also to problems with time dependent coefficients and nonlinear systems .... This, and its ease of use makes POD very competitive in practical use, despite of a certain heuristic flavor*". We introduce POD for general finite dimensional dynamical systems, and refer to [3, 11, 12] for more information, in particular the infinite dimensional case.

1.1 Computation of the POD basis

Consider the high-dimensional dynamical (think of $n \gg 10,000$) system

$$\dot{\mathbf{q}}(t) = \mathbf{f}(\mathbf{q}(t)) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{q} \in \mathbb{R}^n, \tag{1}$$

with initial condition $\mathbf{q}(0) = \mathbf{q}_0$, a source term (external forcing) $\mathbf{B}\mathbf{u}(t)$, where $\mathbf{B} \in \mathbb{R}^{n\times m}$ and $\mathbf{u}(t) \in \mathbb{R}^m$. Such systems often arise in discretizations of partial differential equations which are very common in space weather modeling (e.g., the MHD equations). The system has a solution $\mathbf{q}(t_k) = \mathbf{q}_k$ for $k = 1, \dots, s$, which are often called *snapshots*. The snapshots are stored in the *snapshot matrix*

$$\mathbf{q} = [\mathbf{q}_1 \ \mathbf{q}_2 \dots \ \mathbf{q}_s] \ \in \mathbb{R}^{n\times s}. \tag{2}$$

The POD problem:
Given solution data $\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_s$ of (1), find an r-dimensional subspace with basis $\{\mathbf{v}_1, \dots, \mathbf{v}_r\}$ such that

$$\min_{\mathbf{v}_1,\dots,\mathbf{v}_r} \sum_{j=1}^{s} \left\| \mathbf{q}_j - \sum_{i=1}^{r} (\mathbf{q}_j, \mathbf{v}_i)\mathbf{v}_i \right\|_2^2 \quad \text{s.t.} \quad (\mathbf{v}_i, \mathbf{v}_j) = \delta_{ij}, \tag{3}$$

where $(\mathbf{q}_i, \mathbf{q}_j)_{\mathbb{R}^n} = \mathbf{q}_j^\top \mathbf{q}_i$ is the inner product in $\mathbb{R}^n$. Thus, POD yields a basis that optimally represents the given solution data in the least-squares sense.

[1] The method has been independently rediscovered many times, see Pearson (1901), Hotelling (1933), Loeve (1945) and Karhuenen (1946) and therefore is known under multiple names, such as Karhuenen-Loeve expansion, Principal Component Analysis and Hotelling transformation.

Theorem 1. *[12, Thm 1.1.1.] Let $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_s] \in \mathbb{R}^{n\times s}$ be a snapshot matrix with rank $d \le \min\{n, s\}$. Further, let $\mathbf{Q} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{W}^\top$ be the short singular value decomposition of $\mathbf{Q}$ where $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_d] \in \mathbb{R}^{n\times d}$ and $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_d] \in \mathbb{R}^{d\times s}$ are orthogonal matrices and $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ is diagonal. Then for any $r \in \{1, 2, \dots, d\}$ the solution to the POD problem, which can equivalently be written as*

$$\max_{\tilde{\mathbf{v}}_1,\dots,\tilde{\mathbf{v}}_r} \sum_{j=1}^{s}\sum_{i=1}^{r} |(\mathbf{q}_j, \tilde{\mathbf{v}}_i)|^2 \quad s.t. \quad (\tilde{\mathbf{v}}_i, \tilde{\mathbf{v}}_j) = \delta_{ij}, \tag{4}$$

*is given by the singular vectors $\{\mathbf{v}_i\}_{i=1}^r$, i.e., by the first r columns of $\mathbf{V}$. This is called the **POD basis** of rank r. Moreover, the POD basis retains the maximum energy, measured via*

$$\mathrm{EN}(r) = \sum_{i=1}^{r} \sigma_i^2 \tag{5}$$

Proof. For a proof of this result, see [12]. □

1.2 Method of Snapshots for practical computation of POD basis

For high-dimensional applications, such as those that arise from discretized PDEs, the state-space dimension n can be in the order of millions, so that much fewer snapshots can be stored (due to memory requirements) which leads to $s \ll n$. A computation of POD modes as outlined in the previous section becomes prohibitively expensive, since it requires a singular decomposition of the $n \times s$ matrix $\mathbf{Q}$. To remedy this computational burden, Sirovich [9] introduced the *method of snapshots*, which only requires an SVD of a square matrix of size $s \times s$. To understand the concept, let

$$\mathbf{Q} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{W}^\top \tag{6}$$

be the SVD of the snapshot matrix, so that $\mathbf{V}$ and $\mathbf{W}$ are orthogonal and its respective columns satisfy

$$\mathbf{Q}\mathbf{W} = \mathbf{V}\boldsymbol{\Sigma}, \qquad \mathbf{Q}^\top\mathbf{V} = \mathbf{W}\boldsymbol{\Sigma}.$$

Using the SVD from equation (6), a direct calculation shows that

$$\begin{aligned} \mathbf{Q}\mathbf{Q}^\top &= \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top \quad \in \mathbb{R}^{n\times n}, \\ \mathbf{Q}^\top\mathbf{Q} &= \mathbf{W}\boldsymbol{\Lambda}\mathbf{W}^\top \quad \in \mathbb{R}^{s\times s}, \end{aligned} \tag{7}$$

where $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^2$ contains the POD eigenvalues. Consequently, the columns of $\mathbf{W}$ are the eigenvectors of $\mathbf{Q}^\top\mathbf{Q}$ and the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{Q}\mathbf{Q}^\top$. The method of snapshots computes the decomposition (7) either via eigenvalue decomposition or the more stable SVD. Since $\mathbf{Q}\mathbf{W} = \mathbf{V}\boldsymbol{\Sigma} = \mathbf{V}\boldsymbol{\Lambda}^{1/2}$ we have

$$\mathbf{V} = \mathbf{Q}\mathbf{W}\boldsymbol{\Lambda}^{-1/2},$$

and therefore $\mathbf{V} = [\mathbf{v}_1,\ \mathbf{v}_2,\ \dots, \mathbf{v}_s]$ contains the desired POD modes.

Remark 2. *The computation of the product $\mathbf{Q}^\top\mathbf{Q}$ or $\mathbf{Q}\mathbf{Q}^\top$ can introduce numerical errors. In some cases, such as when the scaling of entries in $\mathbf{Q}$ varies by several orders of magnitude, or when they are small altogether (say $a = 10^{-9}$ then $a^2 = 10^{-18} = 0$ in most machines where machine zero is 1E-16, so we lost this information completely before even computing the SVD), a direct singular value decomposition on $\mathbf{Q}$ should be used. Alternatively, mean subtraction can remedy the scaling problem.*

Algorithm 1 POD basis of rank r.

```
Input: Snapshot matrix $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_s] \in \mathbb{R}^{n\times s}$, rank $r$ and flag for solver.
Output: POD basis $\{\mathbf{v}_i\}_{i=1}^r \in \mathbb{R}^n$ and POD eigenvalues $\{\lambda_i\}_{i=1}^r$.
 1: if flag =0 then
 2:     Compute SVD $\mathbf{Q} = \mathbf{V\Sigma W}$
 3:     Set $\mathbf{v}_i = \mathbf{V}(:,i) \in \mathbb{R}^n$ and $\lambda_i = \mathbf{\Sigma}_{ii}$
 4: else if flag =1 then
 5:     Set $\mathbf{R} = \mathbf{QQ}^\top \in \mathbb{R}^{n\times n}$
 6:     Compute EVD $\mathbf{R} = \mathbf{V\Lambda V}^\top$
 7:     Set $\mathbf{v}_i = \mathbf{V}(:,i) \in \mathbb{R}^n$ and $\lambda_i = \mathbf{\Lambda}_{ii}$
 8: else if flag =2 then
 9:     Determine $\mathbf{K} = \mathbf{Q}^\top\mathbf{Q} \in \mathbb{R}^{s\times s}$
10:     Compute the eigenvalue decomposition (or SVD) $\mathbf{K} = \mathbf{W\Lambda W}^\top$
11:     Set $\mathbf{V} = \mathbf{QW\Lambda}^{-1/2}$ so that $\mathbf{v}_i = \mathbf{V}(:,i)$
12: end if
```

2 Operator Inference for Data Driven Model Order Reduction

Given that we have a low-dimensional basis that represents the data, we next introduce a data-driven model learning method for nonlinear systems. Specifically, we will cover learning polynomial dynamical systems via operator inference [6]. We would like to learn the reduced-order model directly from the data without knowing the full-order model.

2.1 Polynomial Dynamical Systems

Consider now the case that the full-order nonlinear dynamical system is in polynomial form:

$$\dot{\mathbf{q}} = \underbrace{\mathbf{Aq}}_{\text{linear}} + \underbrace{\mathbf{H}(\mathbf{q}\otimes\mathbf{q})}_{\text{quadratic}} + \underbrace{\mathbf{G}(\mathbf{q}\otimes\mathbf{q}\otimes\mathbf{q})}_{\text{cubic}} + \underbrace{\mathbf{Bu}}_{\text{linear input}} + \underbrace{\mathbf{c}}_{\text{constant}} + \text{HOT}, \tag{8}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{H} \in \mathbb{R}^{n\times n^2}$, $\mathbf{G} \in \mathbb{R}^{n\times n^3}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{c} \in \mathbb{R}^{n\times 1}$, and HOT means higher-order polynomial terms. Here, $\otimes$ denotes the usual Kronecker product, which is best understood with an example:

Example 3.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \otimes \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1y_1 & x_1y_2 & x_2y_1 & x_2y_2 \end{bmatrix}^\top \in \mathbb{R}^4,$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \otimes \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1^2 & x_1x_2 & x_2x_1 & x_2^2 \end{bmatrix}^\top \in \mathbb{R}^4,$$

Note the redundancy with the terms x_1x_2 and x_2x_1. Therefore, we also introduce a Kronecker product without redundant terms, i.e.,

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \otimes' \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1^2 & x_1x_2 & x_2^2 \end{bmatrix}^\top \in \mathbb{R}^3.$$

2.2 Reduced-order Model

Projection-based model reduction leverages the Galerkin projection framework. This means that we project (8) onto a subspace spanned by the basis $\mathbf{V} \in \mathbb{R}^{n \times r}$. This means we approximate $\mathbf{q}$ in (8) as $\mathbf{V}\mathbf{q}_r$ where $\mathbf{q}_r \in \mathbb{R}^r$ is now the state vector in the reduced dimension (much smaller than n). However, this means that

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{V}\mathbf{q}_r = \mathbf{A}\mathbf{V}\mathbf{q}_r + \mathbf{H}(\mathbf{V}\mathbf{q}_r \otimes \mathbf{V}\mathbf{q}_r) + \mathbf{G}(\mathbf{V}\mathbf{q}_r \otimes \mathbf{V}\mathbf{q}_r \otimes \mathbf{V}\mathbf{q}_r) + \mathbf{B}\mathbf{u} + \mathbf{R},$$

where we have a residual term $\mathbf{R}$ since otherwise equality would not hold true anymore. The Galerkin projection framework now enforces that the residual is orthogonal to $\mathbf{V}$, so that $\mathbf{V}^\top\mathbf{R}$. This really means that error terms live outside of the subspace. Performing this projection yields

$$\mathbf{V}^\top \left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{V}\mathbf{q}_r\right) = \mathbf{V}^\top \left[\mathbf{A}\mathbf{V}\mathbf{q}_r + \mathbf{H}(\mathbf{V}\mathbf{q}_r \otimes \mathbf{V}\mathbf{q}_r) + \mathbf{G}(\mathbf{V}\mathbf{q}_r \otimes \mathbf{V}\mathbf{q}_r \otimes \mathbf{V}\mathbf{q}_r) + \mathbf{B}\mathbf{u} + \mathbf{c}\right],$$

which in turn gives us the ROM in the following form

$$\dot{\mathbf{q}}_r = \mathbf{A}_r\mathbf{q}_r + \mathbf{H}_r(\mathbf{q}_r \otimes \mathbf{q}_r) + \mathbf{G}_r(\mathbf{q}_r \otimes \mathbf{q}_r \otimes \mathbf{q}_r) + \mathbf{B}_r\mathbf{u} + \mathbf{c}_r, \tag{9}$$

where $\mathbf{A}_r = \mathbf{V}^\top\mathbf{A}\mathbf{V} \in \mathbb{R}^{r \times r}$, $\mathbf{H}_r = \mathbf{V}^\top\mathbf{H}(\mathbf{V} \otimes \mathbf{V}) \in \mathbb{R}^{r \times r^2}$, $\mathbf{G}_r = \mathbf{V}^\top\mathbf{G}(\mathbf{V} \otimes \mathbf{V} \otimes \mathbf{V}) \in \mathbb{R}^{r \times r^3}$, $\mathbf{c}_r = \mathbf{V}^\top\mathbf{c} \in \mathbb{R}^2$. Note the following rule applies to the Kronecker product: $\mathbf{V}\mathbf{q} \otimes \mathbf{V}\mathbf{q} = (\mathbf{V} \otimes \mathbf{V})(\mathbf{q} \otimes \mathbf{q})$, which allows for precomputation of the reduced polynomial operators as shown above.

2.3 Operator Inference (OPINF)

We would like to learn the ROM in (9) directly from the data (i.e., we want to compute $\mathbf{A}_r$, $\mathbf{H}_r$, $\mathbf{B}_r$, and $\mathbf{c}_r$ given the data $\mathbf{Q}$) *without* knowing the operators of the FOM given in (8). This will lead to the method of operator inference [6]. Given the state and input data in the snapshot form

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \dots & \mathbf{q}_s \end{bmatrix} \in \mathbb{R}^{n \times s}, \qquad \mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_s \end{bmatrix} \in \mathbb{R}^{n \times s}$$

where $\mathbf{q}_i \approx \mathbf{q}(t_i)$ are solutions to FOM (8) computed from a time-stepping scheme. Then, operator inference method proceeds as follows:

1. Compute a POD basis $\mathbf{V} \in \mathbb{R}^{n \times r}$ by Algorithm 1 above.
2. Project the high-dimensional data onto basis $\mathbf{V}$:
$$\mathbf{Q}_r = \mathbf{V}^\top\mathbf{Q} = \begin{bmatrix} \mathbf{q}_{1,r} & \mathbf{q}_{2,r} & \dots & \mathbf{q}_{s,r} \end{bmatrix} \in \mathbb{R}^{r \times s}.$$
3. Approximate the time derivative of $\mathbf{q}_r$ via a time-differentiation scheme
$$\dot{\mathbf{Q}}_r = \boldsymbol{\mathcal{D}}(\mathbf{Q}_r),$$
where $\boldsymbol{\mathcal{D}}$ is a numerical differentiation operator.
4. Store the derivative data in
$$\dot{\mathbf{Q}}_r = \begin{bmatrix} \dot{\mathbf{q}}_{1,r} & \dot{\mathbf{q}}_{2,r} & \dots & \dot{\mathbf{q}}_r \end{bmatrix} \in \mathbb{R}^{r \times s}.$$

5. Solve an optimization problem for the operators of the ROM (9) as

$$\begin{pmatrix}\widetilde{\mathbf{A}}_r & \widetilde{\mathbf{B}}_r & \widetilde{\mathbf{H}}_r & \tilde{\mathbf{c}}_r\end{pmatrix} = \underset{\substack{\mathbf{A}_r\in\mathbb{R}^{r\times r}\\ \mathbf{B}_r\in\mathbb{R}^{r\times m}\\ \mathbf{H}_r\in\mathbb{R}^{r\times r^2}\\ \mathbf{c}_r\in\mathbb{R}^{r\times 1}}}{\operatorname{argmin}} \left\|\mathbf{Q}_r^\top\mathbf{A}_r^\top + (\mathbf{Q}_r\otimes'\mathbf{Q}_r)\mathbf{H}_r^\top + \mathbf{U}^\top\mathbf{B}_r^\top + \mathbf{1}_s\mathbf{c}_r^\top - \dot{\mathbf{Q}}_r^\top\right\|_2^2,$$

where $\mathbf{1}_s = [1,\dots,1]\in\mathbb{R}^{1\times s}$.

Note that this expression is for quadratic model only, but the procedure obviously applies to cubic and higher-order polynomial models as well. For simplicity, let

$$\mathbf{O} = \begin{bmatrix}\mathbf{A}_r & \mathbf{H}_r & \mathbf{B}_r & \mathbf{c}_r\end{bmatrix} \in \mathbb{R}^{r\times(r+r^2+m+1)}$$
$$\mathbf{D} = \begin{bmatrix}\mathbf{Q}_r^\top & (\mathbf{Q}_r\otimes'\mathbf{Q}_r)^\top & \mathbf{U}^\top & \mathbf{1}_s\end{bmatrix} \in \mathbb{R}^{s\times(r+r^2+m+1)}$$

so that

$$\widetilde{\mathbf{O}} = \underset{\mathbf{O}\in\mathbb{R}^{r\times(r+r^2+m+1)}}{\operatorname{argmin}} \left\|\mathbf{D}\mathbf{O}^\top - \dot{\mathbf{Q}}_r^\top\right\|_2^2.$$

After solving this, we get the r-dimensional DD ROM

$$\dot{\tilde{\mathbf{q}}}_r = \widetilde{\mathbf{A}}_r\tilde{\mathbf{q}}_r + \widetilde{\mathbf{H}}_r(\tilde{\mathbf{q}}_r\otimes'\tilde{\mathbf{q}}_r) + \widetilde{\mathbf{B}}_r\mathbf{u} + \tilde{\mathbf{c}}_r. \tag{10}$$

Here we make some observations about this problem:

1. If $r > r + r^2 + m + 1$, this is an over-determined linear least-square problem and therefore it has a unique solution.

2. We can break this optimization problem up into r independent LS problems of the form

$$\min_{\mathbf{O}_i\in\mathbb{R}^{r+r^2+m+1}} \|\mathbf{D}\mathbf{o}_i - \mathbf{r}_i\|, \quad i = 1,\dots,r,$$

   where $\mathbf{o}_i$ is a column of $\mathbf{O}^\top$ (row of $\mathbf{O}$) and $\mathbf{r}_i$ is a column of $\dot{\mathbf{q}}_r^\top$. Therefore we get an efficient and scalable implementation.

3. Approximation of the time derivative $\dot{\mathbf{q}}_r(t_k)$ can be done via time-derivative approximation schemes [2]. Alternatively, the OPINF problem can also be formulated in discrete-time.

4. Learning the output operator $\mathbf{C}_r$ in "$\mathbf{y} = \mathbf{C}_r\mathbf{q}_r$" is also possible via

$$\min_{\mathbf{C}_r\in\mathbb{R}^{p\times r}} \left\|\mathbf{Q}_r\mathbf{C}_r^\top - \mathbf{Y}^\top\right\|_2^2.$$

To get some theoretical results, we need to make the following two assumptions. Those relate to the high-fidelity numerical model that generated the training data.

Assumption 4. *The time-stepping scheme for the FOM in (8) is convergent, i.e.,*

$$\max_{i\in\{1,\dots,\frac{T}{\Delta t}\}} \left\|\mathbf{q}_i - \mathbf{q}(t_i)\right\|_2^2 \to 0 \quad \textit{as} \quad \Delta t\to 0.$$

Assumption 5. *The derivatives $\dot{\mathbf{q}}_{r,i}$ approximated from projected states converge to $\frac{d}{dt}\mathbf{q}_r(t_k)$ as*

$$\max_{i\in\{1,\dots,\frac{T}{\Delta t}\}}\left\|\dot{\mathbf{q}}_{r,i}-\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}_r(t_i)\right\|_2 \to 0 \quad \textit{as} \quad \Delta t\to 0.$$

Theorem 6. *(Convergence of the learned and projected ROMs [6]) Let Assumptions 2-3 hold and let a POD basis matrix $\mathbf{V}\in\mathbb{R}^{n\times r}$ be given. Let $\mathbf{A}_r, \mathbf{B}_r, \mathbf{H}_r$ be the intrusively projected ROM operators from (9). If the data matrix $\mathbf{D}=\begin{bmatrix}\mathbf{Q}_r & \mathbf{Q}_r\otimes'\mathbf{Q}_r & \mathbf{U}\end{bmatrix}$ has full column rank, then for every $\varepsilon>0$ there exists a $r\le n$ and a time step size $\Delta t>0$ such that for the difference between the learned ROM operators $\tilde{\mathbf{A}}_r, \tilde{\mathbf{B}}_r, \tilde{\mathbf{H}}_r$, and the intrusively projected operators $\mathbf{A}_r, \mathbf{B}_r, \mathbf{H}_r$ from (9) we have*

$$\left\|\mathbf{A}_r-\tilde{\mathbf{A}}_r\right\|\le\varepsilon,\quad \left\|\mathbf{B}_r-\tilde{\mathbf{B}}_r\right\|\le\varepsilon,\quad \left\|\mathbf{H}_r-\tilde{\mathbf{H}}_r\right\|\le\varepsilon.$$

2.4 Extensions and Code for Operator Inference

Here we show some extensions beyond the basic algorithm applied to polynomial structure:

1. If the NL model is of the form $\dot{\mathbf{q}}=\mathbf{A}\mathbf{q}+\mathbf{H}(\mathbf{q}\otimes\mathbf{q})+f(\mathbf{q},t)+\mathbf{B}\mathbf{u}(t)$ and $f(\mathbf{q},t)$ is known (to some extent), then we can also use OPINF [1].

2. We can combine OPINF with nonlinear state transformations (called lifting) and apply OPINF for a general nonlinear system $\dot{\mathbf{q}}=f(\mathbf{q},t)$. We can turn it into a nice polynomial structure and learn a model [8, 10, 7].

3. More theories on non-Markovian properties of OPINF can be referred to in [5].

4. A python package for operator inference can be found at `https://pypi.org/project/opinf/`.

5. Operator inference has been used for space weather modeling in [4].

References

[1] P. Benner, P. Goyal, B. Kramer, P. Peherstorfer, and K. Willcox. Operator inference for non-intrusive model reduction of systems with non-polynomial nonlinear terms. *Computer Methods in Applied Mechanics and Engineering*, 372:113433, 2020.

[2] R. Chartrand. Numerical differentiation of noisy, nonsmooth, multidimensional data. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 244–248. IEEE, 2017.

[3] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge Monographs on Mechanics. Cambridge University Press, Cambridge, second edition, 2012.

[4] O. Issan and B. Kramer. Predicting solar wind streams from the inner-heliosphere to earth via shifted operator inference. *Journal of Computational Physics*, 473:111689, 2023.

[5] B. Peherstorfer. Sampling low-dimensional Markovian dynamics for pre-asymptotically recovering reduced models from data with operator inference. *SIAM Journal on Scientific Computing*, 42(5):A3489–A3515, 2020.

[6] B. Peherstorfer and K. Willcos. Data-driven operator inference for nonintrusive projection-based model reduction. *Computer Methods in Applied Mechanics and Engineering*, 306:196–215, 2016.

[7] E. Qian, B. Kramer, A. N. Marques, and K. E. Willcox. Transform & learn: A data-driven approach to nonlinear model reduction. In *AIAA Aviation 2019 Forum*, page 3707, 2019.

[8] E. Qian, B. Kramer, B. Peherstorfer, and K. Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020.

[9] L. Sirovich. Turbulence and the dynamics of coherent structures. i-coherent structures. ii-symmetries and transformations. iii-dynamics and scaling. *Quarterly of Applied Mathematics*, 45:561–571, 1987.

[10] R. Swischuk, B. Kramer, C. Huang, and K. Willcox. Learning physics-based reduced-order models for a single-injector combustion process. *AIAA Journal*, 58:6:2658–2672, 2020.

[11] S. Volkwein. Proper orthogonal decomposition for linear-quadratic optimal control. Lecture Notes, University of Konstanz, available at https://urldefense.com/v3/__http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/scripts.php__;!!Mih3wA!VE5BxPjjUwMfbd1A31L8WPO6k_wQKhO86S2le_VjeOoGUuxQt-66w6_2FJx4s_AZw8k$, 2013.

[12] S. Volkwein. Proper orthogonal decomposition: Theory and reduced-order modelling. Lecture Notes, University of Konstanz, available at https://urldefense.com/v3/__http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/scripts.php__;!!Mih3wA!VE5BxPjjUwMfbd1A31L8WPO6k_wQKhO86S2le_VjeOoGUuxQt-66w6_2FJx4s_AZw8k$, 2013.