Physics of Fluids

Comparing different nonlinear dimensionality reduction techniques for data-driven unsteady fluid flow modeling

Cite as: Phys. Fluids **34**, 117119 (2022); https://doi.org/10.1063/5.0127284
Submitted: 20 September 2022 • Accepted: 02 November 2022 • Published Online: 16 November 2022

Hunor Csala, Scott T. M. Dawson and Amirhossein Arzani

Physics of Fluids ARTICLE scitation.org/journal/phf

Comparing different nonlinear dimensionality reduction techniques for data-driven unsteady fluid flow modeling

Cite as: Phys. Fluids **34**, 117119 (2022); doi: 10.1063/5.0127284
Submitted: 20 September 2022 · Accepted: 2 November 2022 ·
Published Online: 16 November 2022

Hunor Csala,[1,2] Scott T. M. Dawson,[3] and Amirhossein Arzani[1,2,a)]

AFFILIATIONS

[1]Department of Mechanical Engineering, University of Utah, Salt Lake City, Utah 84112, USA
[2]Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, Utah 84112, USA
[3]Department of Mechanical, Materials and Aerospace Engineering, Illinois Institute of Technology, Chicago, Illinois 60616, USA

[a)]**Author to whom correspondence should be addressed**: amir.arzani@sci.utah.edu

ABSTRACT

Computational fluid dynamics (CFD) is known for producing high-dimensional spatiotemporal data. Recent advances in machine learning (ML) have introduced a myriad of techniques for extracting physical information from CFD. Identifying an optimal set of coordinates for representing the data in a low-dimensional embedding is a crucial first step toward data-driven reduced-order modeling and other ML tasks. This is usually done via principal component analysis (PCA), which gives an optimal linear approximation. However, fluid flows are often complex and have nonlinear structures, which cannot be discovered or efficiently represented by PCA. Several unsupervised ML algorithms have been developed in other branches of science for nonlinear dimensionality reduction (NDR), but have not been extensively used for fluid flows. Here, four manifold learning and two deep learning (autoencoder)-based NDR methods are investigated and compared to PCA. These are tested on two canonical fluid flow problems (laminar and turbulent) and two biomedical flows in brain aneurysms. The data reconstruction capabilities of these methods are compared, and the challenges are discussed. The temporal vs spatial arrangement of data and its influence on NDR mode extraction is investigated. Finally, the modes are qualitatively compared. The results suggest that using NDR methods would be beneficial for building more efficient reduced-order models of fluid flows. All NDR techniques resulted in smaller reconstruction errors for spatial reduction. Temporal reduction was a harder task; nevertheless, it resulted in physically interpretable modes. Our work is one of the first comprehensive comparisons of various NDR methods in unsteady flows.

I. INTRODUCTION

Many fluid mechanics modeling methods deal with, and indeed require, large amounts of data from experiments or numerical simulations. Recently, well-known techniques in data science and machine learning (ML) have been applied to complex fluid flow problems to extract underlying information about the flow structures,[1] speed up numerical simulations,[2] and to enable real-time prediction and control.[3] Typical computational fluid dynamics (CFD) simulations even for laminar flows can produce numerical results with millions of degrees of freedom in space and thousands of timesteps. However, such spatiotemporal high-dimensionality is often an artifact of CFD's resolution. That is, it is necessary to design high spatiotemporal resolution strategies to achieve desired accuracy on discrete mesh and timepoints. Interestingly, even some seemingly complex fluid flow problems live in a hidden low-dimensional manifold, which once identified and exposed could drastically reduce the high-dimensionality of the data. Reduced-order models (ROMs) aim to discover such low-dimensionality and therefore capture the most important flow features.[4] ROMs have been a fundamental part of the fluid dynamics research for a long time from classical projection-based ROMs[5–9] to more recent purely data-driven ROMs[10–14] and manifold learning approaches.[15]

Finding a suitable set of reduced coordinates is a crucial first step in developing ROMs. The most common approach in fluid dynamics is finding a new low-dimensional coordinate system using principal component analysis (PCA) also known as proper orthogonal decomposition (POD),[16] and then deriving the dynamics of the system by Galerkin projection of the Navier–Stokes equations onto these POD modes.[8,17] Both steps can be improved by using more sophisticated ML models, instead of POD. Specifically, nonlinear dimensionality

reduction (NDR) methods, such as manifold learning or autoencoders (AEs), can be applied.[18] Galerkin projection can be replaced by more generic regression models, such as sparse identification of nonlinear dynamics (SINDy) or long-short-term-memory (LSTM) neural networks (NNs).[2] Another closely related data-driven ROM method is the dynamic mode decomposition (DMD),[11,19] which has been recently frequently used for fluid dynamics, but similar to POD it is inherently linear. DMD in a sense is even more "linear" than POD as not only the modes are linear but also the evolution equation is linear, whereas POD with Galerkin projection could produce a nonlinear evolution model. The main advantage of manifold learning and neural network-based approaches is that they can uncover the nonlinear flow behavior.

AEs have been extensively applied in recent years for finding low-dimensional embeddings of CFD data. Various different AE types and architectures have been studied such as fully connected networks,[20,21] variational autoencoders (VAE),[22–24] convolutional neural networks (CNN),[25–27] 3D CNNs,[28] graph convolutional neural networks (GCNN),[29,30] sparse convolutional neural networks (SCNN),[31] physics-informed autoencoders,[32,33] symmetry-aware autoencoders,[34] adversarial autoencoders (AAE),[35] and advection-aware autoencoders.[36] Most of them have been tested on canonical fluid dynamics problems, for example, flow over a cylinder or turbulent channel flow. Gruber *et al.*[29] compared a fully connected network, a CNN, and a GCNN for a 2D flow over a cylinder. Results suggested that the fully connected network had the best overall performance, GCNN worked better only when the latent space was relatively high-dimensional (e.g., Ref. 64), while CNN did not perform well due to the irregularity of the domain. The full understanding of the performance of various types of deep learning models on such problems is still an open question.

Apart from AEs, manifold learning (also known as generalized principal component analysis–gPCA) methods are also suitable for nonlinear dimensionality reduction.[37,38] These include kernel principal components analysis (KPCA), locally linear embedding (LLE), Laplacian eigenmap (LEM), isometric mapping (Isomap), multi-dimensional scaling (MDS), self-organizing map (SOM), t-distributed stochastic neighbor embedding (t-SNE), maximum variance unfolding (MVU) or semidefinite embedding (SDE), uniform manifold approximation projection (UMAP), local tangent space alignment (LTSA), diffusion map, Sammon mapping, and curvilinear component analysis (CCA), among others. However, manifold learning methods have been less frequently studied in modeling physical systems such as fluid flow. Pyta and Abel[39] compared the Sammon mapping, LEM, and an AE to POD for the transient 2D lid-driven cavity flow. LEM had the best performance for this problem; however, surprisingly the improvement of LEM with respect to POD became less apparent when increasing the nonlinearity of the system, that is, higher Reynolds (Re) numbers. Ehlert *et al.*[15] used LLE for the transient flow over a cylinder at Re = 100 and showed that two LLE modes achieved similar performance to ten POD modes. Farzamnik *et al.*[40] used Isomap for dimensionality reduction and k-nearest neighbors (k-NN) for data reconstruction of jets and flow over multiple cylinders. Finally, these methods have also been explored in experimental fluid mechanics.[41,42] Recently, network science has also been exploited for analyzing fluid flows.[43–45] Some of these methods (e.g., Lagrangian transport networks) share similarities with manifold learning methods (e.g., LEM, Isomap) as they both use neighborhood or graph-based techniques for finding dominant and coherent flow patterns.

The manifold learning methods can also be used for parametric studies where instead of different time steps, simulations with different parameters take the place of different snapshots. Parametric steady transonic flows around an airfoil were studied using Isomap and LLE.[46,47] Halder *et al.*[48] developed a non-intrusive ROM for a parametric steady incompressible lid-driven cavity problem where Isomap was used for dimensionality reduction. Diez *et al.*[49] used KPCA for parametric mass transport problems. Rovira *et al.*[50] used t-SNE and UMAP for dimensionality reduction of turbulent reactive flow data. Instead of the temporal dimension of data, different physical quantities (e.g., velocity, temperature, and species concentrations) were used for constructing the data matrix. A similar study was done in Ref. 51 to compare PCA to Isomap, MDS, and a physically derived reduced-order manifold method for turbulent jet flames, showing spatial modes as well. Wu *et al.*[52] used t-SNE for dimensionality reduction of 2D turbulent simulations, where nine different flow features (e.g., turbulence intensity and Q-criterion) were reduced to two for visualization.

Despite growing interest in ROM modeling of fluid flow, a detailed understanding of qualitative and quantitative differences among various NDR methods is lacking. In this study, we provide a detailed qualitative (mode patterns) and quantitative (reconstruction error) comparison of these methods. Specifically, PCA is compared to various NDR techniques (LLE, KPCA, LEM, Isomap, and AE) for two canonical 2D fluid flow problems (flow over a cylinder and turbulent channel flow). Additionally, 3D patient-specific blood flow in the internal carotid artery (ICA) and middle cerebral artery (MCA) aneurysms are also investigated as practical real-world examples. Linear data-driven ROM approaches have been used to study blood flow physics[53–55] and perform data assimilation[56] in cardiovascular flows.

Herein, we demonstrate that with proper settings the NDR methods can achieve superior results in terms of data reconstruction. The spatial vs temporal arrangement of the data and their effect on mode extraction is also discussed. The first few dominant spatial modes are visualized and qualitatively compared between the methods, and their relevance to flow physics is discussed. Finally, the methods are compared quantitatively via the relative reconstruction error. The main contribution of our work is the detailed comparison of several nonlinear dimensionality reduction techniques applied to unsteady fluid flow problems.

II. METHODS

A. Dimensionality reduction

All unsteady CFD simulation (or experimental) results can be arranged in a data matrix $\mathbf{X}$, where the columns $\mathbf{x}_i \in \mathbb{R}^n$ correspond to flattened velocity data at different timesteps

$$\mathbf{X}_{(n\times m)} = \begin{bmatrix} | & | & \dots & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_m \\ | & | & \dots & | \end{bmatrix} = \begin{bmatrix} - - - & \boldsymbol{\chi}_1 & - - - \\ - - - & \boldsymbol{\chi}_2 & - - - \\ \vdots & \vdots & \vdots \\ - - - & \boldsymbol{\chi}_n & - - - \end{bmatrix}. \quad (1)$$

$\mathbf{X} \in \mathbb{R}^{n\times m}$, where n is the number of spatial locations where the velocity is sampled (i.e., mesh points or measurement locations), and m is the number of saved temporal snapshots. It is possible to think in terms of the rows of $\mathbf{X}$ as well, where each row $\boldsymbol{\chi}_i \in \mathbb{R}^m$ corresponds to the temporal evolution of velocity values at a given spatial location. For simplicity, the magnitude of the velocity vector is used in this

study, but alternatively, the different velocity components could also be stacked on top of each other, as discussed in Appendix A 1.

In Secs. II A 1–II A 7, seven dimensionality reduction algorithms are shortly presented. More detailed descriptions can be found in Refs. 37, 38, and 57. The goal for all methods is to construct a low-dimensional embedding $\mathbf{Y}$, which captures the most dominant flow features, then map back to the original high-dimensional space with the least amount of error. Two broad approaches are possible. We can reduce the spatial size of the data, leading to an embedding $\mathbf{Y} \in \mathbb{R}^{(r\times m)}$ where $r \ll n$, or reduce the temporal size of the data, resulting in an embedding $\mathbf{Y} \in \mathbb{R}^{(n\times r)}$ where $r \ll m$.

To assess the quality of the algorithms, the relative reconstruction error using the Frobenius norm is used

$$\varepsilon_r(\mathbf{X}, \mathbf{X}_r) = \frac{\|\mathbf{X} - \mathbf{X}_r\|_F}{\|\mathbf{X}\|_F}, \tag{2}$$

where $\mathbf{X}_r$ is the reconstruction of $\mathbf{X}$ from an r-dimensional latent space. More complicated error metrics have been developed,[57,58] mostly in the image processing community, but have been rarely used for physical simulations. The Scikit-learn Python library[59] was used for computing the manifold learning methods, while the AE models were created in PyTorch.

1. Principal component analysis (PCA)

First, we overview principal component analysis (PCA),[60] which is the standard linear approach. The basic idea behind PCA was developed in multiple branches of science, and thus, it has many different names such as Karhunen–Loève transform, Hotelling transform, empirical orthogonal eigenfunctions, and proper orthogonal decomposition (POD). The POD terminology is used most frequently in the fluid dynamics community.[8] The underlying mathematical algorithm is singular value decomposition (SVD), which is one of the most powerful matrix factorization algorithms and a cornerstone of linear algebra. PCA constructs an optimal basis such that the greatest variance of the data corresponds to the direction of the first coordinate, the second greatest to the second coordinate, and so on. PCA is an orthogonal linear transformation, and thus, it cannot identify low-dimensional structures, which are highly nonlinear.

To compute PCA, first, the temporal mean of the data at each spatial location is subtracted from each row

$$\tilde{\mathbf{X}} = \sqrt{\frac{1}{m}}(\mathbf{X} - \bar{\mathbf{x}}\mathbf{1}), \tag{3}$$

where $\mathbf{1} = [1, 1\ldots, 1] \in \mathbb{R}^{(1\times m)}$ is a vector of ones and $\bar{\mathbf{x}} = \frac{1}{m}\sum_{j=1}^{m} \mathbf{X}_{ij} \in \mathbb{R}^{(n\times 1)}$ is a column vector containing the mean of each row. The SVD can be written as

$$\tilde{\mathbf{X}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{4}$$

where $\mathbf{\Sigma}$ contains the singular values along its diagonal, $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors, and $()^T$ denotes the transpose. It is computationally faster to perform the "thin" or "economy" SVD instead of the "full" SVD, meaning that the columns of $\mathbf{U}$ not corresponding to rows of $\mathbf{V}^T$ are discarded, assuming $m < n$.[37] Therefore, the matrices will have the following size: $\mathbf{U} \in \mathbb{R}^{(n\times m)}$, $\mathbf{\Sigma} \in \mathbb{R}^{(m\times m)}$, and $\mathbf{V} \in \mathbb{R}^{(m\times m)}$. PCA is often written in terms of the covariance matrix $\mathbf{C}$,

$$\mathbf{C}\mathbf{v}_j = \tilde{\mathbf{X}}^T\tilde{\mathbf{X}}\mathbf{v}_j = \sigma_j^2\mathbf{v}_j, \tag{5}$$

where σ_j is the jth diagonal entry of $\mathbf{\Sigma}$ and $\mathbf{v}_j$ is the jth column of $\mathbf{V}$. $\mathbf{C}$ is also called time-correlation matrix and has a size of $\mathbf{C} = \tilde{\mathbf{X}}^T\tilde{\mathbf{X}} \in \mathbb{R}^{(m\times m)}$. The space-correlation matrix can also be formed, which is called the similarity matrix $\mathbf{G}$

$$\mathbf{G}\mathbf{u}_j = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\mathbf{u}_j = \sigma_j^2\mathbf{u}_j, \tag{6}$$

where $\mathbf{u}_j$ is the jth column of $\mathbf{U}$. The size of $\mathbf{G} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T \in \mathbb{R}^{(n\times n)}$ is usually much larger than $\mathbf{C} \in \mathbb{R}^{(m\times m)}$. Therefore, for computational efficiency, usually $\mathbf{C}$ is used for computing the singular values. However, both approaches lead exactly to the same results. This difference between spatial vs temporal correlation will be further discussed later for the NDR methods, where unlike PCA very different results are obtained for these two choices.

The rank-r PCA approximation of the original data is given by

$$\tilde{\mathbf{X}}_r = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^T, \tag{7}$$

where $\mathbf{U}_r \in \mathbb{R}^{(n\times r)}$ and $\mathbf{V}_r \in \mathbb{R}^{(m\times r)}$ contain the first r columns of $\mathbf{U}$ and $\mathbf{V}$, and $\mathbf{\Sigma}_r \in \mathbb{R}^{(r\times r)}$ contains the first r rows and columns of $\mathbf{\Sigma}$.

For temporal reduction, the low-dimensional embedding $\mathbf{Y} \in \mathbb{R}^{(n\times r)}$ can be simply expressed as

$$\mathbf{Y} = \mathbf{U}_r\mathbf{\Sigma}_r. \tag{8}$$

The columns of $\mathbf{U}$ represent an ordered set of spatial functions, the so-called modes. The importance of each mode is given by its associated singular value. These modes can be visualized as velocity values. That is, each mode $\mathbf{u}_j \in \mathbb{R}^{(n\times 1)}$ has the same size as the velocity data in one snapshot $\mathbf{x}_j \in \mathbb{R}^{(n\times 1)}$. If spatial reduction of the data is sought, the low-dimensional embedding $\mathbf{Y} \in \mathbb{R}^{(r\times m)}$ can be expressed with the right singular vectors

$$\mathbf{Y} = \mathbf{\Sigma}_r\mathbf{V}_r^T. \tag{9}$$

2. Locally linear embedding (LLE)

LLE, first introduced in Ref. 61, lies on the assumption that the local neighborhood of a data point $\mathbf{x} \in \mathbb{R}^n$ can be well approximated by the linear subspace spanned by the k nearest neighbors (k-NN) of $\mathbf{x}$. Subsequently, these locally linear approximations can be used to construct a low-dimensional embedding $\mathbf{y} \in \mathbb{R}^r$, which preserves the local neighborhood relationships.

The LLE algorithm can be summarized in the following steps:

(1) Find the k-NN for each data point $\mathbf{x}_j \in \mathbb{R}^n$, where $\mathbf{x}_j$ is the jth column of $\tilde{\mathbf{X}}$, that is, the jth velocity snapshot. The neighborhood of $\mathbf{x}_j$ will correspond to snapshots that are similar to $\mathbf{x}_j$. To calculate the k-NN, the pairwise distances are calculated between $\mathbf{x}_j$ and all other data points. The distance measure used for the k-NN can be chosen, but mostly the L_2 norm is used. Then, the k closest points to $\mathbf{x}_j$ will be assigned as its k nearest neighbors.
(2) Approximately each data point as a linear combination of its k-NN with coefficients w_{ij}: $\mathbf{x}_j \approx \sum_{i=1}^{k} w_{ij}\mathbf{x}_i$

- To find the coefficients (or weights), the following minimization problem needs to be solved:

$$\min_{w_{ij}} \sum_{j=1}^{m} \left\| \mathbf{x}_j - \sum_{i=1}^{m} w_{i,j} \mathbf{x}_i \right\|^2, \quad \text{subject to} \sum_{i=1}^{m} w_{ij} = 1. \tag{10}$$

If $\mathbf{x}_i$ is not a k-NN of $\mathbf{x}_j$, then $w_{ij} = 0$.

- The solution to this constrained optimization problem has the following form for the weights of $\mathbf{x}_j$:

$$\tilde{\mathbf{w}}_j = \frac{\mathbf{G}_j^{-1}\mathbf{1}}{\mathbf{1}^T \mathbf{G}_j^{-1}\mathbf{1}} \in \mathbb{R}^k, \tag{11}$$

where $\mathbf{G}_j \in \mathbb{R}^{(k\times k)}$ is the local Gram matrix at $\mathbf{x}_j$, which is defined as

$$\mathbf{G}_j = \mathbf{G}_{il}^j = (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_l - \mathbf{x}_j) \tag{12}$$

with $\mathbf{x}_i$ and $\mathbf{x}_l$ being the k-NN of $\mathbf{x}_j$. $\mathbf{1} \in \mathbb{R}^{(k\times 1)}$ is a vector of ones. Note that $\mathbf{w}_j \in \mathbb{R}^m$, but $w_{i,j} = 0$ when $\mathbf{x}_i$ is not k-NN of $\mathbf{x}_j$; therefore, $\tilde{\mathbf{w}}_j \in \mathbb{R}^k$ contains the k non-zero components of $\mathbf{w}_j$.

(3) Compute the low-dimensional embedding $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m] \in \mathbb{R}^{(r\times m)}$ using the weights found in step 2.

- The low-dimensional representation should minimize the following error:

$$\min_{\mathbf{Y}} \sum_{j=1}^{m} \left\| \mathbf{y}_j - \sum_{i=1}^{m} w_{ij} \mathbf{y}_i \right\|^2, \quad \text{subject to} \sum_{j=1}^{m} \mathbf{y}_j = \mathbf{0} \quad \text{and} \quad \frac{1}{m} \sum_{j=1}^{m} \mathbf{y}_j \mathbf{y}_j^T = \mathbf{I}. \tag{13}$$

The first constraint requires the embedding to be centered at the origin, while the second constraint forces the embedding to have unit covariance and set the rank arbitrarily to r, the dimension of the desired latent space.

- The solution to this optimization problem is the r eigenvectors of the matrix

$$\mathcal{L} = (\mathbf{I} - \mathbf{W})(\mathbf{I} - \mathbf{W})^T \in \mathbb{R}^{(m\times m)} \tag{14}$$

corresponding to the second to $(r+1)$ smallest eigenvalues. $\mathbf{I} \in \mathbb{R}^{(m\times m)}$ is the identity matrix, and $\mathbf{W} \in \mathbb{R}^{(m\times m)}$ is the matrix of weights composed by the weight vectors $\mathbf{w}_j$ as columns. The rows of $\mathbf{Y}$ are set to be equal to the eigenvectors of $\mathcal{L}$. The smallest eigenvalue will be zero with a corresponding eigenvector equal to $\mathbf{1} \in \mathbb{R}^m$ due to the first constraint; therefore, it is omitted. Note that these eigenvectors are of size m, which is the number of snapshots. Therefore, these are not spatial modes as they have been in case of the PCA, but temporal modes. This means that visualizing these eigenvectors in the spatial domain is not possible. This is further discussed in Sec. II B.

In certain cases, it can be numerically challenging to solve Eq. (10), as the Gram matrix may be close to singular.[62] Therefore, a small regularization term λ is added to the trace of $\mathbf{G}$,

$$\mathbf{G} \leftarrow \mathbf{G} + \lambda \mathbf{I}. \tag{15}$$

In manifold learning methods, reconstructing the original data based on low-dimensional embedding is not as simple as in the case of PCA. The LLE reconstruction is based on the inverse LLE presented in Ref. 63 and the non-parametric model in Ref. 64. The same method is used in Refs. 15 and 47 for reconstructing fluid flow data, and in Ref. 51, a simplified version is applied.

Using inverse LLE, first, the k-NN are found for each point $\mathbf{y}_i \in \mathbb{R}^r$ in the embedded space denoted with $\mathbf{y}_{ij}$, where j goes from 1 to k. Then, the reconstruction weights can be computed by minimizing the following:

$$\min_{w_{ij}} \left\| \mathbf{y}_i - \sum_{j=1}^{k} w_{ij} \mathbf{y}_{ij} \right\|_2^2, \quad \text{subject to} \sum_{j=1}^{k} w_{ij} = 1. \tag{16}$$

Equation (16) can be solved similarly to Eq. (10). The λ regularization parameter is also used here to overcome the numerical issues regarding singularity. When the weights w_{ij} are obtained, data point $\mathbf{x}_i \in \mathbb{R}^n$ can be approximated as

$$\mathbf{x}_i \approx \sum_{j=1}^{k} w_{ij} \mathbf{x}_j. \tag{17}$$

Therefore, the reconstructed data $\mathbf{X}_r$ will be

$$\mathbf{X}_r = \mathbf{W}\tilde{\mathbf{X}}. \tag{18}$$

3. Kernel principal component analysis (KPCA)

A nonlinear extension of PCA using kernels is called kernel PCA.[65] This method computes the eigenvectors of the so-called kernel matrix. The main idea behind KPCA is that the given data may not lie in a linear subspace, so first the data are embedded in a higher-dimensional space with dimension $D > n$ such that the data approximately lie in a linear subspace. Then, PCA is applied to the embedded data. The high-dimensional embedding $\mathbf{\Phi} \in \mathbb{R}^{(D\times m)}$ is never actually computed thanks to the "kernel trick."[65] A kernel matrix is defined as

$$\mathcal{K} = \mathbf{\Phi}^T \mathbf{\Phi}, \tag{19}$$

$$\mathcal{K}_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j), \tag{20}$$

where κ is the kernel function and $\mathbf{x}_i$ is the ith column of $\tilde{\mathbf{X}}$. Therefore, the kernel matrix has dimensions $\mathcal{K} \in \mathbb{R}^{(m\times m)}$. Popular kernel functions are polynomial $\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y})^p$, radial basis function (RBF) $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\gamma\|\mathbf{x} - \mathbf{y}\|^2)$, and hyperbolic tangent $\kappa(\mathbf{x}, \mathbf{y}) = \tanh(\mathbf{x}^T\mathbf{y})$. In this study, the RBF kernel is used.

The centered kernel matrix $\tilde{\mathcal{K}}$ can be expressed as

$$\tilde{\mathcal{K}} = \mathbf{H}\mathcal{K}\mathbf{H} = \left(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T\right)\mathcal{K}\left(\mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T\right), \tag{21}$$

where $\mathbf{H} \in \mathbb{R}^{(m\times m)}$ is the centering matrix. The eigendecomposition of the centered kernel matrix and the low-dimensional embedding $\mathbf{Y} \in \mathbb{R}^{(r\times m)}$ can be written as

$$\tilde{\mathcal{K}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T, \tag{22}$$

$$\mathbf{Y} = \mathbf{\Lambda}_r^{1/2}\mathbf{V}_r^T, \tag{23}$$

where $\mathbf{V}$ and $\mathbf{\Lambda}$ are matrices containing the eigenvectors and eigenvalues and the subscript r refers to the r largest eigenvalues and corresponding eigenvectors. Reconstructing the data from the embedded space is not trivial as the reverse mapping generally does not exist, which is known as the pre-image problem.[66] Here, a method proposed in Ref. 67 is used, which relies on kernel ridge regression. A regularization ridge parameter α is added to the trace of the kernel matrix, similar to the λ parameter of the LLE method.

It has been shown in Refs. 68 and 69 that LLE, LEM, and Isomap all can be viewed as KPCA with a specially constructed kernel matrix; therefore, a unified framework for these nonlinear dimensionality reduction methods can be constructed.

4. Laplacian eigenmaps (LEM)

A similar manifold learning method to LLE is LEM, introduced in Ref. 70. LEM is also called spectral embedding due to its close relation to spectral clustering.[71] LEM tries to find a low-dimensional representation where nearby points in the original manifold are also nearby, therefore preserving local information.

The LEM algorithm can be summarized in the following steps:

(1) Find the k-NN of each data point $\mathbf{x}_j \in \mathbb{R}^n$. It should also be noted that instead of finding the k-nearest neighbors, an ε-neighborhood can be computed, where points are considered neighbors if $\|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq \varepsilon$ according to some distance metric. Both approaches are investigated in Ref. 39 for the lid-driven cavity problem. This second approach is more geometrically motivated, but it can lead to severely disconnected graphs, and choosing ε is more difficult than choosing k.

(2) Define a matrix of weights $\mathbf{W} \in \mathbb{R}^{(m\times m)}$ whose entries w_{ij} measure the affinity between to data points $\mathbf{x}_i$ and $\mathbf{x}_j$. The simplest approach would be to simply assign $w_{ij} = 1$ if they are neighbors and 0 otherwise. This would correspond to the so-called adjacency matrix from graph theory

$$w_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is in the k} - \text{NN of } \mathbf{x}_j \text{ or vice-versa,} \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

The more popular technique is to use an RBF kernel for defining different weights to different point pairs, based on their distance

$$w_{ij} = e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2}. \tag{25}$$

These two methods can also be combined as the following:

$$w_{ij} = \begin{cases} e^{-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2} & \text{if } \mathbf{x}_i \text{ is in the k-NN of } \mathbf{x}_j \text{ or vice versa,} \\ 0 & \text{otherwise.} \end{cases} \tag{26}$$

(3) Construct the low-dimensional embedding $\mathbf{Y} \in \mathbb{R}^{(r\times m)}$ by solving the following minimization problem:

$$\min_{\mathbf{Y}} \sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \tag{27}$$

with appropriate constraints explained below to avoid getting a $\mathbf{Y} = \mathbf{0}$ trivial solution, where $\mathbf{y}_j$ is the jth column of $\mathbf{Y}$.

A diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{(m\times m)}$ can be defined as the row-wise sum of the weights

$$d_{ii} = \sum_{j=1}^{m} w_{ij}. \tag{28}$$

The constraints needed for the optimization problem are $\mathbf{YD1} = \mathbf{0}$ and $\mathbf{YDY}^T = \mathbf{I} \in \mathbb{R}^{(r\times r)}$. The first one avoids getting a constant embedding, and the second one ensures that $\text{rank}(\mathbf{Y}) = r$. The Laplacian $\mathcal{L} \in \mathbb{R}^{(m\times m)}$ matrix can be constructed as

$$\mathcal{L} = \mathbf{D} - \mathbf{W}. \tag{29}$$

The solution to the optimization problem will be the r eigenvectors of $\mathcal{L}$ corresponding to the r smallest eigenvalues, excluding the first one which is equal to zero. The size of these eigenvectors is m, which is equal to the number of snapshots. These eigenvectors will be the rows of $\mathbf{Y} \in \mathbb{R}^{(r\times m)}$.

A possible reconstruction method for LEM was suggested in Ref. 72; however, in this study, the inverse LLE reconstruction method [Eq. (16)] will be used for LEM as well for simplicity.

5. Isometric mapping (Isomap)

The isometric feature mapping algorithm was developed by Tenenbaum *et al.*[73] It extends the classical idea of preserving interpoint distances of the multi-dimensional scaling (MDS) technique[74] by using geodesic distances instead of the Euclidian distance. Note that the classical MDS with Euclidean distances is a linear subspace learning method.[75] The Isomap algorithm can be summarized as follows:

(1) Construct the k-NN graph of each data point $\mathbf{x}_i \in \mathbb{R}^n$ and weight the graph by labeling each edge with its Euclidean length.

(2) Estimate the geodesic distances between all pairs of points by computing the shortest path distances in the graph and constructing the distance matrix $\mathbf{R} \in \mathbb{R}^{(m\times m)}$. The two most commonly used methods are Dijkstra's algorithm and the Floyd–Warshall algorithm.[76] Subsequently, create the matrix of squared distances: $S_{ij} = R_{ij}^2$.

(3) Construct the low-dimensional embedding based on the matrix of graph distances using MDS. First, define the double centered dissimilarity matrix $\mathbf{K} \in \mathbb{R}^{(m\times m)}$ as

$$\mathbf{K} = -\frac{1}{2}\mathbf{HSH}, \tag{30}$$

where $\mathbf{H} = \mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}^T \in \mathbb{R}^{(m\times m)}$ is a centering matrix. The eigenvalue decomposition of the symmetric matrix $\mathbf{K}$ can be written as

$$\mathbf{K} = \mathbf{V\Lambda V}^T, \tag{31}$$

where $\mathbf{V}$ and $\mathbf{\Lambda}$ contain the eigenvectors and eigenvalues of $\mathbf{K}$. The low-dimensional embedding $\mathbf{Y} \in \mathbb{R}^{(r\times m)}$ can be written as

$$\mathbf{Y} = \mathbf{\Lambda}_\mathbf{r}^{\mathbf{1/2}}\mathbf{V}_r^T, \tag{32}$$

where the subscript r refers to the first r largest eigenvalues and corresponding eigenvectors.

The LLE reconstruction method is used for the Isomap embedding as well, similarly to Ref. 47. Constructing the low-dimensional embedding is similar to the KPCA approach, where $\mathbf{S}$ takes the role of the kernel matrix. It is important to note that the LLE and KPCA reconstruction methods use the original data $\mathbf{X}$ as well for the reconstruction, not just the embedded data $\mathbf{Y}$. This is a significant difference from PCA and AE, which can reconstruct data just from $\mathbf{Y}$.

Hyperparameters for all manifold learning methods are listed in Table I in the Appendix.

6. Autoencoder (AE)

Another approach for nonlinear dimensionality reduction is the use of neural networks, namely, autoencoders (AE). They have a specific architecture, where the output is designed to be the same as the input, but there is a so-called bottleneck layer in the middle, which has significantly fewer neurons than the input. This forces the network to learn a meaningful low-dimensional representation of the data. It was shown that a shallow AE with only one fully connected hidden layer and a linear activation function leads to an embedding that spans the same subspace as PCA.[1,77] Deep AEs with nonlinear activation functions can outperform PCA for fluid flow problems,[25] since they usually involve complex nonlinear behavior. Here, a fully connected deep neural network, also called multilayer perceptron (MLP), will be used for dimensionality reduction. A clear advantage of AEs over the manifold learning methods is that reconstructing the data is straightforward through the decoder network. In most cases, the encoder and decoder networks share the same structure, but this is not a requirement.

7. Mode decomposing autoencoder (MDAE)

To obtain interpretable spatial modes, a special AE structure was introduced by Murata *et al.*[78] The main idea was that after obtaining the low-dimensional representation in the bottleneck layer, separate decoders are then connected to each of the latent vectors and their final output is summed. It was called mode decomposing autoencoder (MDAE), even though the "modes" or decomposed fields attained this way are inherently different from modes obtained from temporal data reduction. The main distinction is that the decomposed fields here are time-dependent, while modes obtained by other methods are not. Nevertheless, these temporal decomposed fields can still help us understand the inner workings of the AE as they are physically meaningful. For the 2D flow over a cylinder case, it was shown that by performing POD on two decomposed fields, the orthogonal bases corresponding to the first six POD modes of the original data can be recovered.[78] The method has also been applied to 3D data.[79] In the original paper, a CNN architecture was used; however, here a fully connected network is utilized.

In all AE and MDAE cases, the temporal mean of the data was subtracted and the data were rescaled to $[0, 1]$. The ReLU activation function was used to impose nonlinearity. The Adamax optimizer was selected. Recently, it was suggested in Ref. 80 that this optimizer is the best-suited choice for AEs, which was confirmed by tests on the fluid datasets used during this study. A variable learning rate was chosen using a learning rate scheduler. The initial learning rate was 10^{-3}, and then, it was reduced by a factor of 0.1 after reaching 1/3 and 2/3 of the final number of 9000 epochs, leading to a final learning rate of 10^{-5}. The network structure and other hyperparameters are listed in Tables II–V in the Appendix for the different test cases. An overview of the AE and MDAE architectures is shown in Fig. 1.

B. The choice of data arrangement

The conventional way to look at dimensionality reduction of fluid flow data is to take the columns of $\tilde{\mathbf{X}}$, corresponding to velocity snapshots $\mathbf{x}_i \in \mathbb{R}^n$ as data points.[2] Therefore, dimensionality reduction will lead to m samples of embedded vectors $\mathbf{y}_i \in \mathbb{R}^r$, where $r < n$. This is shown in the top part of Fig. 1 for the AE and MDAE architectures. This is the established way for creating ROMs[17] by modeling the dynamics of the system in the low-dimensional representation, then mapping back to the original space. However, in general, this does not lead to interpretable spatial modes, since the latent space is inherently

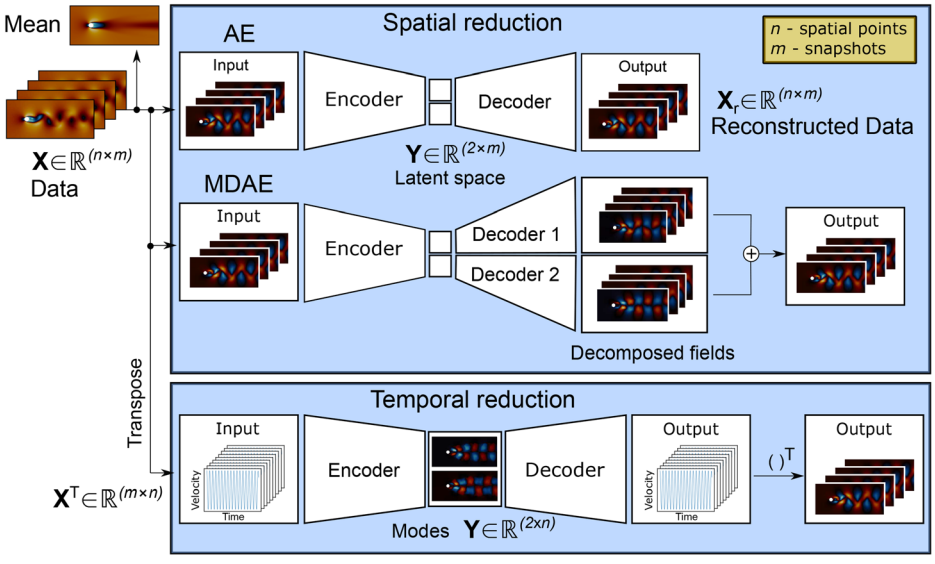

FIG. 1. AE architectures are shown. Top row: spatial dimensionality reduction using AE and MDAE. Bottom row: temporal dimensionality reduction leading to spatial modes in the latent space. For simplicity, a latent space of size $r = 2$ is shown.

smaller than the physical domain as the spatial size of the data is reduced from $\mathbb{R}^n$ to $\mathbb{R}^r$. It is only a special property of PCA that allows achieving both spatial modes (using the left singular vectors) and it is also suitable for ROMs (with the embedding obtained from the right singular vectors). Nevertheless, if velocity snapshots $\mathbf{x}_i$ are used as input, the NDR methods will not provide flow structures that can be visualized as a scalar or vector field.

To overcome this difficulty, the rows of $\tilde{\mathbf{X}}$ are taken as data points $\boldsymbol{\chi}_i \in \mathbb{R}^m$ (or equivalently the columns of $\tilde{\mathbf{X}}^T$). This arrangement is shown at the bottom part of Fig. 1 for the AE case. Usually, the number of snapshots is much smaller than the number of spatial points where velocity is sampled $m \ll n$, so the resulting matrices for eigenvector calculation will be $(n \times n)$ instead of $(m \times m)$, resulting in significantly higher computational requirements. Traditionally, these manifold learning methods are tested and presented on datasets of images, where a data matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ contains m images of n pixels each.[38,57] In that case, it is straightforward that different data points should be different images. However, in the case of fluid flow, the different data points can also be thought of as temporal velocity values at a fixed spatial location. In this context, two "neighboring" data points $\boldsymbol{\chi}_i$, $\boldsymbol{\chi}_j \in \mathbb{R}^m$ would have similar velocities over time, even though they are not necessarily close in the physical domain. This approach corresponds to reducing the temporal size of the data to $r < m$ but keeping the spatial size fixed at n. Thus, the results of the dimensionality reduction can be visualized as spatial modes. In PCA, this approach simply corresponds to swapping the left and right singular vectors and keeping the singular values unchanged.

This duality issue is briefly addressed in Ref. 39. Additionally, in Refs. 81 and 82, this duality is called the duality between data and feature manifold structures, and it is argued that both views make sense and contain valuable information. Gu and Zhou[81] developed a co-clustering algorithm that tries to leverage information from both manifolds. In the context of fluid dynamics, the feature manifold corresponds to the velocity values at different spatial locations and the data manifold is spanned by the different snapshots.

Typically, the spatial reduction of data is preferred as dimensionality reduction is used as a first step toward ROMs (e.g., Refs. 46–48). In some other studies (e.g., reactive flows with multiple thermochemical variables[50,51]), the second approach was used and the number of scalar field variables was reduced while keeping the spatial size constant, therefore obtaining visual modes. In this study, both approaches are considered and compared. The spatial reduction can lead to powerful low-dimensional representations suitable for ROMs, while physically interpretable modes and coherent structures can be found through temporal reduction.

C. Test cases

Four test cases were investigated with varying complexity to assess the efficiency of the dimensionality reduction methods for unsteady fluid flow modeling. The first problem is the 2D flow over a cylinder at Re = 100, which is a canonical benchmark problem for unsteady fluid flows. It has been widely studied for developing low-dimensional models[83] and extensively used as benchmark for data-driven unsteady fluid flow models (e.g., Refs. 15, 22, 24, and 25). Second, a 2D slice of 3D turbulent channel flow is considered, which is another widely used benchmark case of notably higher complexity (e.g., Refs. 26–28). Finally, pulsatile blood flow in internal carotid artery (ICA) and middle cerebral artery (MCA) brain aneurysms are considered as real-world examples of unsteady fluid flow. These test cases are summarized in Fig. 2.

1. Flow over a cylinder

The first test case is 2D incompressible flow over a cylinder at Re = 100, which is larger than the critical value for the onset of vortex shedding. The cylinder is centered at (0, 0) with a diameter of $D = 1$. The length of the domain is $50D$, and the width is $20D$. The domain spans $-10 \leq x \leq 40$ and $-10 \leq y \leq 10$. The no-slip boundary condition is enforced on the cylinder. At the front and lateral sides of the domain, a uniform incoming flow $\mathbf{u}_{in} = (1, 0)$ is applied. At the outflow boundary, a zero traction condition is employed. The Reynolds number was set to $\mathrm{Re} = \frac{|\mathbf{u}_{in}|D}{\nu} = 100$ by choosing the viscosity to be $\nu = 0.01$. The initial condition was set to zero.

The Navier–Stokes equations were solved using the finite-element method with FEniCS.[84] The total number of triangular cells

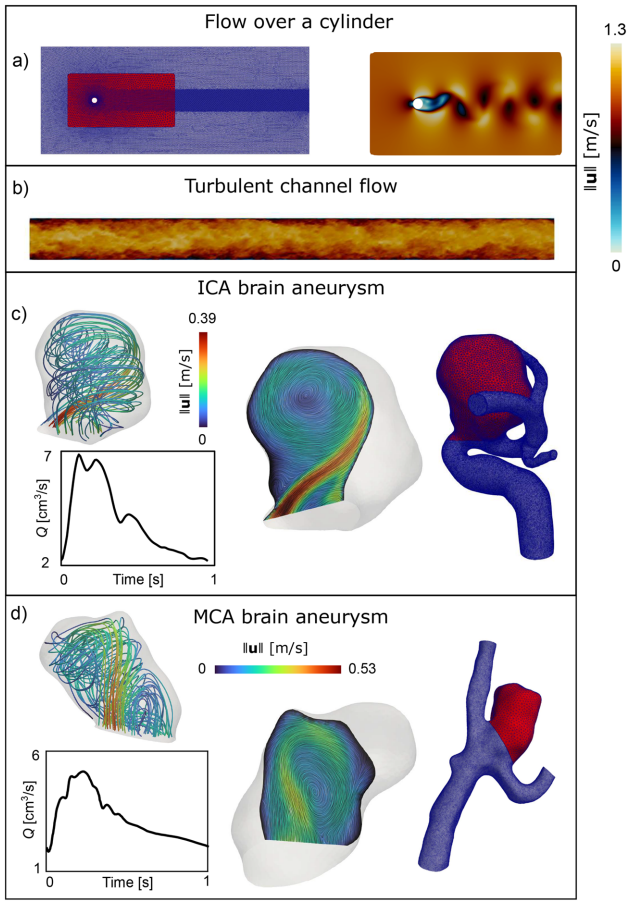

FIG. 2. (a) Flow over a cylinder case is shown. The domain and mesh are shown on the left. The cropped and downsampled region is shown in red, and a sample velocity snapshot is shown on the right. (b) A velocity snapshot of the turbulent channel flow is shown. (c) ICA and (d) MCA aneurysm cases with 3D streamlines, 2D streamlines in a cross section, inlet flow waveform, and mesh are shown. The cropped and downsampled region of interest is shown in red.

was 102k (second-order elements). The solution was computed until $t = 200$, with a time step size of $\Delta t = 0.01$. The initial transient phase was discarded. After the fully developed vortex shedding started, 1000 snapshots were saved uniformly in $t \in [100, 200)$. The spatial domain was cropped to $-5 \leq x \leq 15$ and $-5 \leq y \leq 5$ and downsampled to a coarser mesh, containing 3961 points [highlighted in red in Fig. 2(a)]. Sample flow results depicting the von Kármán vortex street are shown at the first saved snapshot on the right side of Fig. 2(a). For further analysis, the magnitude of the velocity vector is used, and the data matrix $\mathbf{X}_{(n\times m)}$ for this case is of size 3961×1000.

2. Turbulent channel flow

The second test case is a direct numerical simulation (DNS) of turbulent channel flow,[85] obtained from the Johns Hopkins turbulence databases.[86] In this dataset, incompressible turbulent channel flow was simulated in a domain of size $8\pi \times 2 \times 3\pi$, using $2048 \times 512 \times 1536$ nodes. The friction Reynolds number was $\mathrm{Re}_\tau \approx 1000$, and 4000 snapshots were saved. A 2D slice in the spanwise (z direction) midplane of the dataset was used for this study. Data were downsampled by taking every eighth data point in both spatial directions, leading to a grid of $256 \times 64 = 16\,384$ points. In the temporal dimension, every second snapshot was used, adding up to a total of 2000 snapshots. An example snapshot can be seen in Fig. 2(b), colored by velocity magnitude. For the dimensionality reduction algorithms, the magnitude of the velocity vector is used; therefore, the data matrix $\mathbf{X}_{(n\times m)}$ is of size $16\,384 \times 2000$ for this case.

3. ICA brain aneurysm

An internal carotid artery (ICA) brain aneurysm model is selected as the third test case. The geometry is taken from the Aneurisk dataset[87] (Aneurisk model ID: C0035). The meshing and CFD simulation was carried out with SimVascular.[88] The mesh consisted of 6.6M elements with a boundary layer mesh. A population-averaged pulsatile inflow waveform[89] was used for the inlet, scaled according to Ref. 90. The inlet flow rate Q is shown in Fig. 2(c). Split-resistance boundary condition was applied for the outlets using Murray's law. No-slip condition was applied for the vessel walls, which were considered rigid. Newtonian blood rheology model was assumed with a density of $\rho = 1060\,\mathrm{kg/m^3}$ and dynamic viscosity of $\mu = 0.004$ kg/ms. The Reynolds number using the average inlet velocity and inlet diameter is $\mathrm{Re} = 322$. The simulation time step is chosen to divide each cardiac cycle into 10 000 timesteps. The simulation was run for three cardiac cycles, and 1000 snapshots were saved from the last cycle. Finally, results were cropped and downsampled to coarser mesh consisting of 40 500 points, while keeping the no-slip condition at the wall. This region is highlighted in red in Fig. 2(c). Blood flow streamlines are also shown in Fig. 2(c) for the first saved time step. The plane shown in the middle of Fig. 2(c) is chosen for visualizing the modes later on. The velocity magnitude is used for the data-driven methods, and the data matrix $\mathbf{X}_{(n\times m)}$ is of size $40\,500 \times 1000$.

4. MCA brain aneurysm

A middle cerebral artery (MCA) brain aneurysm model is taken as the last test case (Aneurisk dataset, model ID: C0051), as shown in Fig. 2(d). While the ICA case was a lateral aneurysm caused by the high curvature of the vessel, the MCA case is a bifurcation type aneurysm near a branching point. The simulation was prepared using SimVascular, and all settings were identical to the ICA case. The only difference was the inlet flow profile, which was taken from Ref. 91 as shown in Fig. 2(d). The Reynolds number using the average inlet velocity and inlet diameter is $\mathrm{Re} = 432$. The mesh consisted of 7.3M elements with a boundary layer mesh. For the dimensionality reduction, the results are cropped and downsampled to a coarser mesh of 10 167 points, as shown in Fig. 2(d). The blood flow streamlines are also shown. The plane shown in the middle of Fig. 2(d) will be used for visualizing the modes later on. The velocity magnitude is used for the data-driven methods, and the data matrix $\mathbf{X}_{(n\times m)}$ is of size $10\,167 \times 1000$.

III. RESULTS

The deep learning cases were run on a GPU (Tesla V100-SXM2–16GB, A100, and P100-PCIE-16GB models for different runs) with average computational times ranging between 0.2 and 5 h for each case. The manifold learning cases were run on a single CPU where spatial reduction needed only up to 5 min of computing time, while temporal reduction took up to 8 h. The precise run times depend on the test case (i.e., input size) and hyperparameters (e.g., nearest neighbors). For manifold learning, the reconstruction error can be computed for different number of modes without the need for recomputing the embedding, while for the autoencoders each case with a different latent space size required a completely separate run.

The different dimensionality reduction methods were applied to the four test cases considering both data arrangement scenarios, leading to spatial and temporal reduction of the data. To assess the complexity of each flow, the PCA singular values are shown in Fig. 3 where for a better comparison each data matrix was normalized. It is clear that the singular values of the turbulent channel flow problem decay significantly slower than the other cases, while the cylinder flow shows the fastest decay. The normalized cumulative energy is defined as the cumulative sum of singular values normalized by the sum of all singular values. The flow over a cylinder case reaches a normalized cumulative energy value close to 100% after less than 50 singular values, while for the turbulent case even 200 modes only correspond to around 80% of the energy. Note that for the turbulent channel flow case, there are 2000 modes in total, but only the first 1000 are shown. The two brain aneurysm cases show similar complexity. Even though the Reynolds number is higher for the MCA case ($\mathrm{Re} = 432$) than the ICA ($\mathrm{Re} = 322$), the ICA case shows slightly slower decay in singular values. This could be because of a more complex temporal behavior due to the larger ratio of systolic to diastolic inlet velocities. In Secs. III A–III C, a detailed comparison between NDR modes and data reconstruction is presented.

A. Flow over a cylinder

The relative reconstruction error for the flow over a cylinder is shown in Fig. 4 as a function of mode numbers on a semi-log plot. Spatial and temporal reduction cases are shown. MDAE can only be used for spatial reduction; therefore, it is absent from the temporal reduction figures. The PCA curve shows a near log-linear decrease in relative reconstruction error with an increasing number of modes. However, none of the nonlinear methods follow this behavior. For the spatial reduction [Fig. 4(a)], KPCA, LEM, and Isomap have a sharp

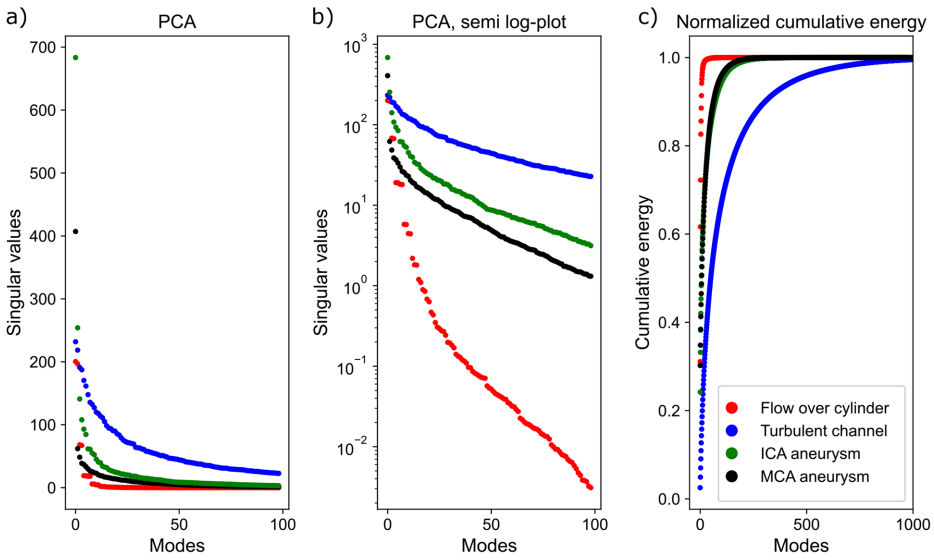

FIG. 3. (a) Singular values for different principal components (modes). (b) Singular values on a semi-log plot for better visualization. (c) Normalized cumulative energy, defined as the cumulative sum of singular values normalized by the sum of all singular values.

drop in error at two modes but then adding subsequent modes does not result in significantly better reconstruction. The two AEs can achieve the best performance with only two modes, but with more than six modes LLE outperforms them. LLE shows a sharp drop at four modes, and adding higher modes further reduces the reconstruction error up to eight modes. In spatial reconstruction, all nonlinear techniques outperform PCA with a fewer number of modes.

The relative reconstruction error for the temporal reduction is shown in Fig. 4(b). The reconstruction error is calculated on the same dataset that is used for identifying the NDR models, as the main interest here is not model evolution (extrapolation), but rather mode extraction. Note that for PCA there is no difference between the spatial and temporal arrangement of the data; therefore, the PCA results are exactly the same as in Fig. 4(a). For both data arrangement scenarios, the temporal-mean-subtracted data are used. All the nonlinear methods achieve inferior performance in this case compared to the spatial arrangement. The AE is distinctly better than the other methods, but LLE, KPCA, and Isomap have comparable results to PCA. The LEM algorithm has the worst performance and does not reach the accuracy of PCA at any number of modes.

The benefit of the temporal arrangement of the data is that interpretable spatial modes could be obtained, which facilitate the understanding of the underlying flow structures. These modes are shown in Fig. 5. For each method, four representative modes are chosen. Sometimes these modes come in pairs, with similar structures but opposite signs; therefore, only modes with distinct patterns are shown. Higher order modes are less significant, so only the first eight modes are chosen for visualization. The AE modes are not ranked and come

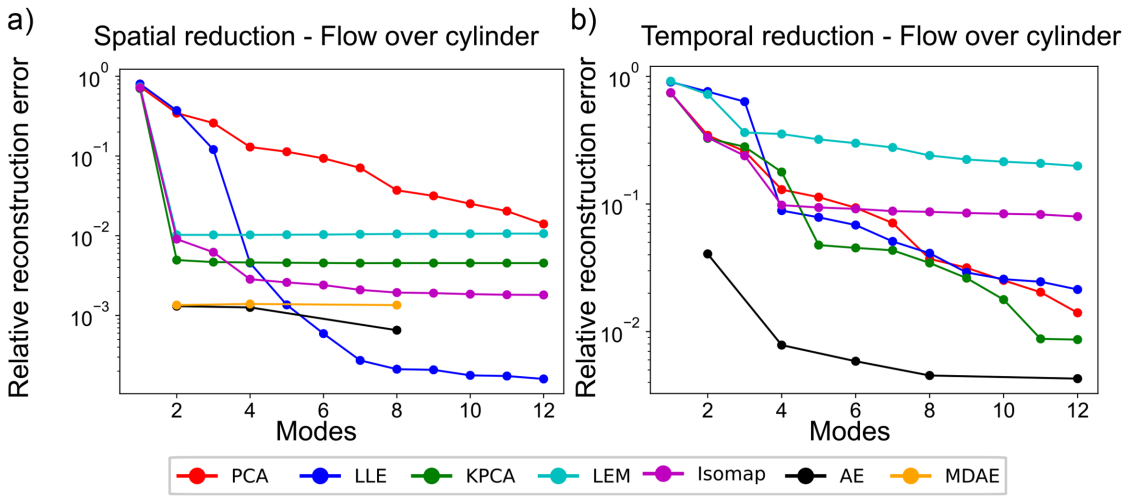

FIG. 4. Relative reconstruction error with a different number of modes for (a) spatial and (b) temporal reduction of the flow over a cylinder case. Note that MDAE is only used for spatial reduction.

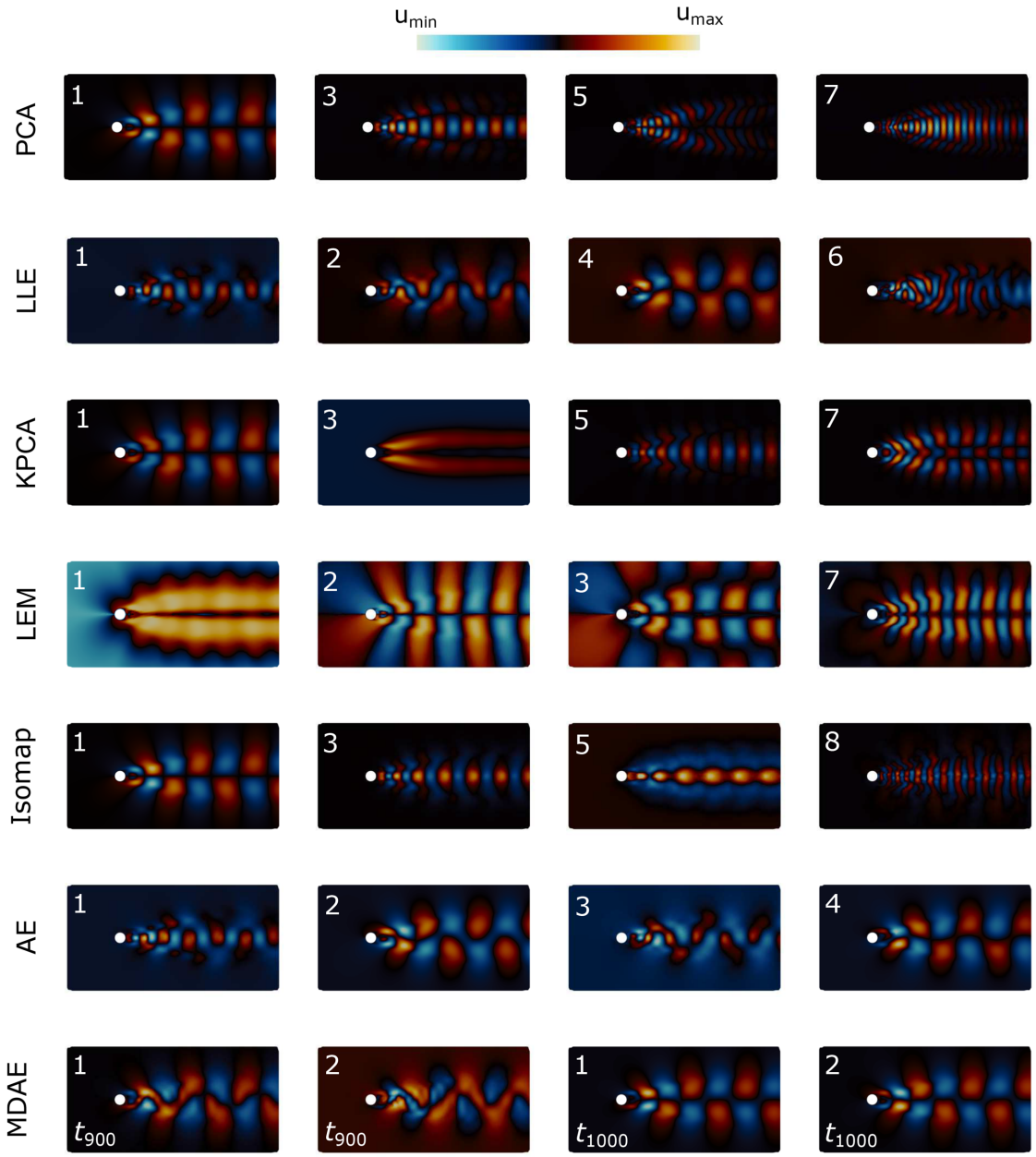

FIG. 5. Representative set of spatial modes visualized for the flow over a cylinder case. The inset number refers to the mode number, where higher number modes are less significant. The exceptions are the AEs, where the obtained modes are in an arbitrary order of importance. In MDAE, the snapshot number is specified. The color bar range is different for each mode, but only the qualitative structure of the modes is important.

in an arbitrary order; for example, mode 1 is not necessarily more significant than mode 4; therefore, their arrangement in the figure is arbitrary and is just numbered from 1 through 4. Additionally, in the AE, each setup with different latent space sizes results in different modes. The two modes of an AE with a latent space of size two might not be the same as the first two (or any two) modes of an AE with a latent space of different sizes. In Fig. 5, modes are shown from an AE with a latent space of size 4 for the convenience of visualization. In MDAE, the modes are time dependent; therefore, these decomposed fields are shown for an MDAE model with a latent space size of two during two different timesteps (snapshot numbers 900 and 1000 as indicated in each subfigure). The color scale has a different range for each method and mode, but these values are omitted since the focus is only on the qualitative structure of the modes.

In Fig. 5, similar patterns can be observed throughout the different methods; for example, PCA-1, LLE-4, KPCA-1, LEM-3, Isomap-1, AE-2,4, and MDAE-1,2-t_{1000} all show the main underlying structure behind the vortex shedding, that is, alternating regions of low and high velocity, which are anti-symmetric along the horizontal mid-plane. Interestingly, LLE starts worse than PCA for 1–3 modes (in terms of reconstruction), but shows a sharp drop in error at four modes, corresponding to the common mode described, which suggests the importance of this pattern. Some of the higher order modes are also comparable; for example, PCA-7, LLE-6, KPCA-7, LEM-7, and Isomap-8 show thinner vertical regions alternating between low velocity and high velocity. In the case of the nonlinear techniques, the borders between high velocity and low velocity are often complicated and not just straight lines as in the case of PCA. The AE modes (especially 1 and 3) are more complicated than the other methods, which seems to be contributing to the one order of magnitude better reconstruction. LLE modes 1 and 2 resemble the AE modes 1 and 3, all showing a zigzag pattern of low and high velocities. Similar structures can be seen for the MDAE decomposed fields at snapshot t_{900}. Another interesting feature is that KPCA-3 and LEM-1 modes seem to separate the cylinder wake region from the other parts of the domain; therefore, they provide little information regarding the phase of the limit cycle.

B. Turbulent channel

The relative reconstruction error is plotted in Fig. 6 for the 2D turbulent channel flow up to 100 modes. Similarly to the flow over a cylinder case, all nonlinear methods perform better than PCA for the turbulent case in spatial reduction [Fig. 6(a)]. The differences here are even larger due to the highly nonlinear nature of turbulent flows. It has to be noted that, unlike PCA, the nonlinear dimensionality reduction methods do not necessarily show a monotonically decreasing behavior with an increasing number of modes. For example, the LLE, KPCA, Isomap, and AE curves all show non-monotonic behavior. For a fewer number of modes, the AE provides the smallest error, while for a higher number of modes, the LLE gives the best results for spatial reduction. Unlike the flow over cylinder case where LEM had the highest reconstruction error out of the manifold learning methods, LEM performed well for the turbulent channel flow case. One reason behind this could be the high temporal resolution of the turbulent channel flow, meaning that nearby snapshots are very similar to each other, which makes the embedding and reconstruction easier for LEM.

In the case of the temporal reduction, the results are shown in Fig. 6(b). LEM and Isomap give similar results to PCA until 20 modes, but unlike PCA further modes do not produce significantly better reconstructions. LLE seems to achieve worse reconstruction than PCA; however, it gets better with more modes. KPCA is better than PCA, and finally, the AE results have notably smaller errors than all the other methods. It is important to note that the AE used for spatial reduction achieved an order of magnitude better results than the AE for temporal reduction. The spatial modes are visualized in Fig. 7. The overall observations made for the cylinder case hold here as well. The MDAE decomposed fields are not shown here since they failed to uncover any features and the two modes were simply decomposing the velocity field in half. For PCA, KPCA, LEM, and Isomap, it can be seen that the first modes capture the large-scale coherent structure and the higher order modes capture the smaller scales. Similar patterns can be identified in PCA-1, KPCA-1, LEM-1, Isomap-1, and AE-2 modes. In general, PCA, LEM, and Isomap have similar reconstruction errors for less than ten modes; therefore, it is no surprise that the modes are similar. The KCPA modes are also similar to these, even though they produce smaller reconstruction errors. This might be because of the different reconstruction method used for KPCA. The LLE modes are dissimilar to all of the other modes and seem to capture much smaller scales. Interestingly, they have the largest reconstruction error, so these modes are not able to sufficiently represent the flow dynamics. It is interesting to note that the AE modes are visually not extremely distinct from the PCA, KPCA, LEM, and Isomap modes, even though

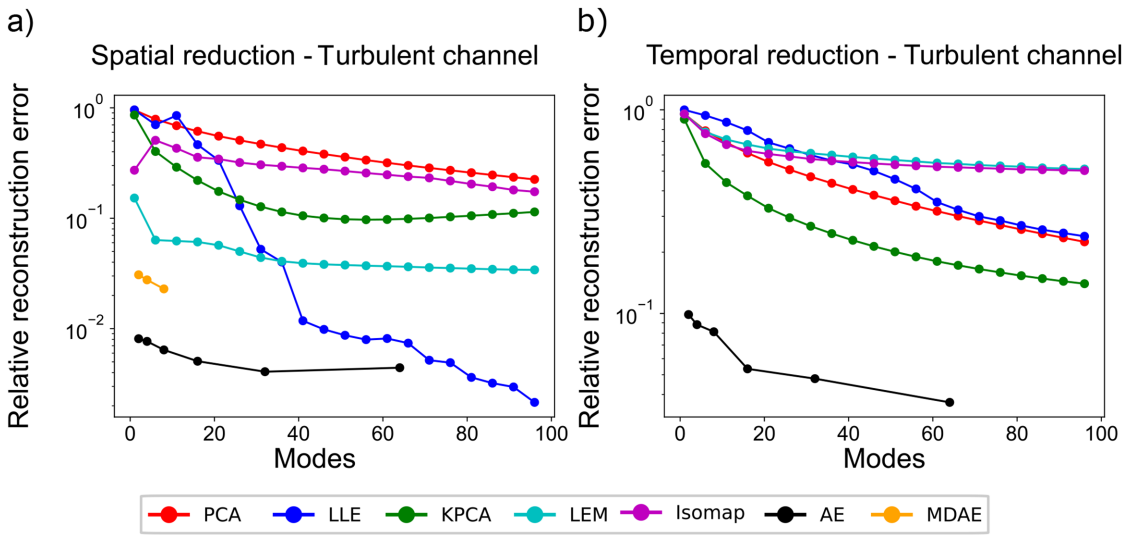

FIG. 6. Relative reconstruction error with different number of modes for (a) spatial and (b) temporal reduction of the turbulent channel case. Note that MDAE is only used for spatial reduction.

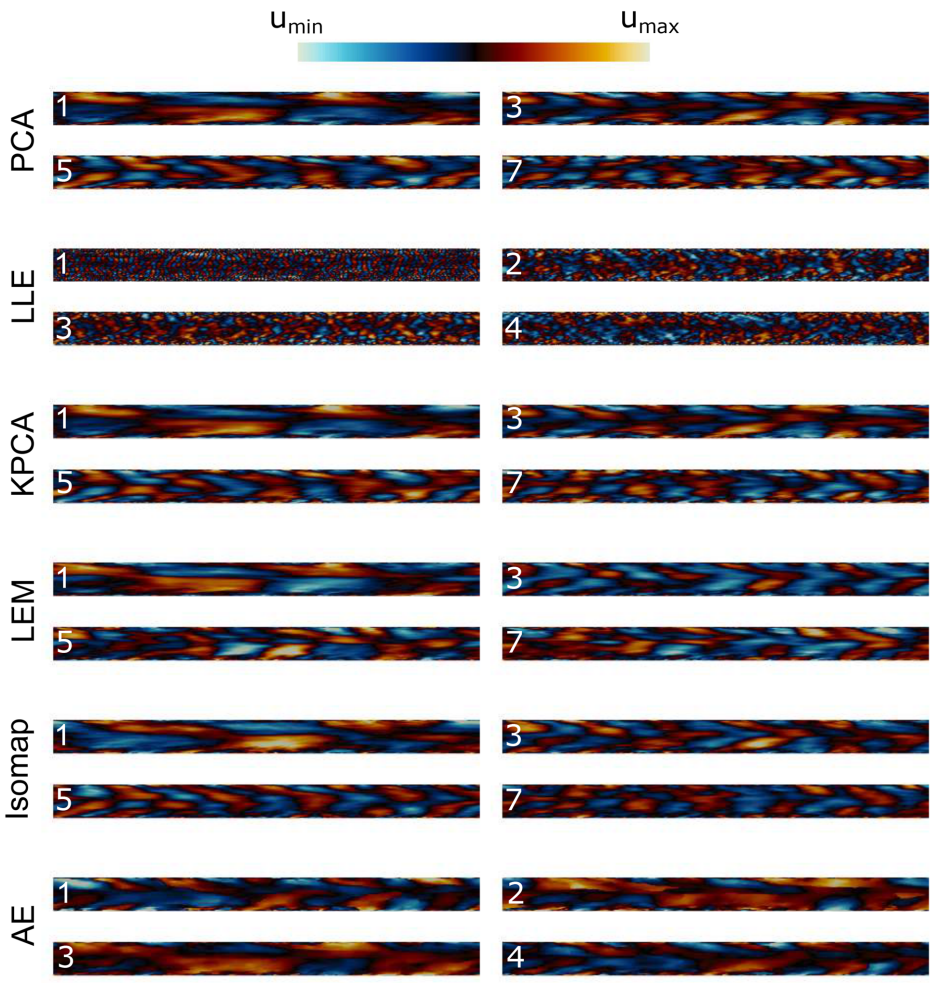

FIG. 7. Representative set of spatial modes visualized for the turbulent channel case. The inset number refers to the mode number, where higher number modes are less significant. The exceptions are the AEs, where the obtained modes are in an arbitrary order of importance. The color bar range is different for each mode, but only the qualitative structure of the modes is important.

they achieve an order of magnitude better spatial reconstruction. This suggests that the AE reconstruction using neural networks is much more powerful than the other methods.

C. Brain aneurysms

Reconstruction results for the ICA and MCA brain aneurysm cases are shown in Fig. 8 for the first 40 modes. In the ICA case for spatial reduction [Fig. 8(a)], KPCA achieves quite similar reconstructions to PCA, while LLE, LEM, and Isomap have significantly better results. LLE, LEM, and Isomap have a sharp drop in error at 2–3 modes, but additional higher modes lead to only minor improvements in reconstruction error. AE and MDAE outperform the other methods. The temporal reduction results for the ICA aneurysm case are plotted in Fig. 8(b). Initially, LEM and Isomap give better reconstruction than PCA, but higher modes worsen their results. LLE performs better than PCA, but the AE gives the smallest reconstruction error among all methods. To visualize the changes in reconstruction error during the cardiac cycle, Figs. 8(c) and 8(d) plot the reconstruction error throughout one cardiac cycle as a function of time, reconstructed with eight modes for the spatial and temporal ICA cases, respectively. It can be seen that for some NDR cases, the error follows the shape of the inlet waveform (e.g., LLE, Isomap, and LEM in the spatial case), while in other cases, it is more constant (e.g., PCA, KPCA, and MDAE in the spatial case). In the temporal case, all curves except the AE have a similar shape. In the MCA case, similar patterns were observed in the variation of error in time, and therefore, the results are not shown for brevity.

Results for the MCA aneurysm case are shown in Figs. 8(e) and 8(f). In spatial reconstruction, KCPA has the worst performance in nonlinear methods with only marginally better results than PCA. LLE, LEM, and Isomap perform well with a few modes, but further modes do not significantly improve their reconstruction. AE achieves the best performance but a latent space size of 16 and 32 does not bring improvement over a size of 8. Even though the singular value decay in Fig. 3 suggested that the ICA data have higher complexity than MCA, the manifold learning methods perform better in the ICA case. In the temporal data arrangement scenario, KPCA and LEM do not provide a notable improvement over PCA. Isomap gives good results with a latent space size of around 10, but reconstruction worsens with a high number of modes. AE has an order of magnitude better results than all

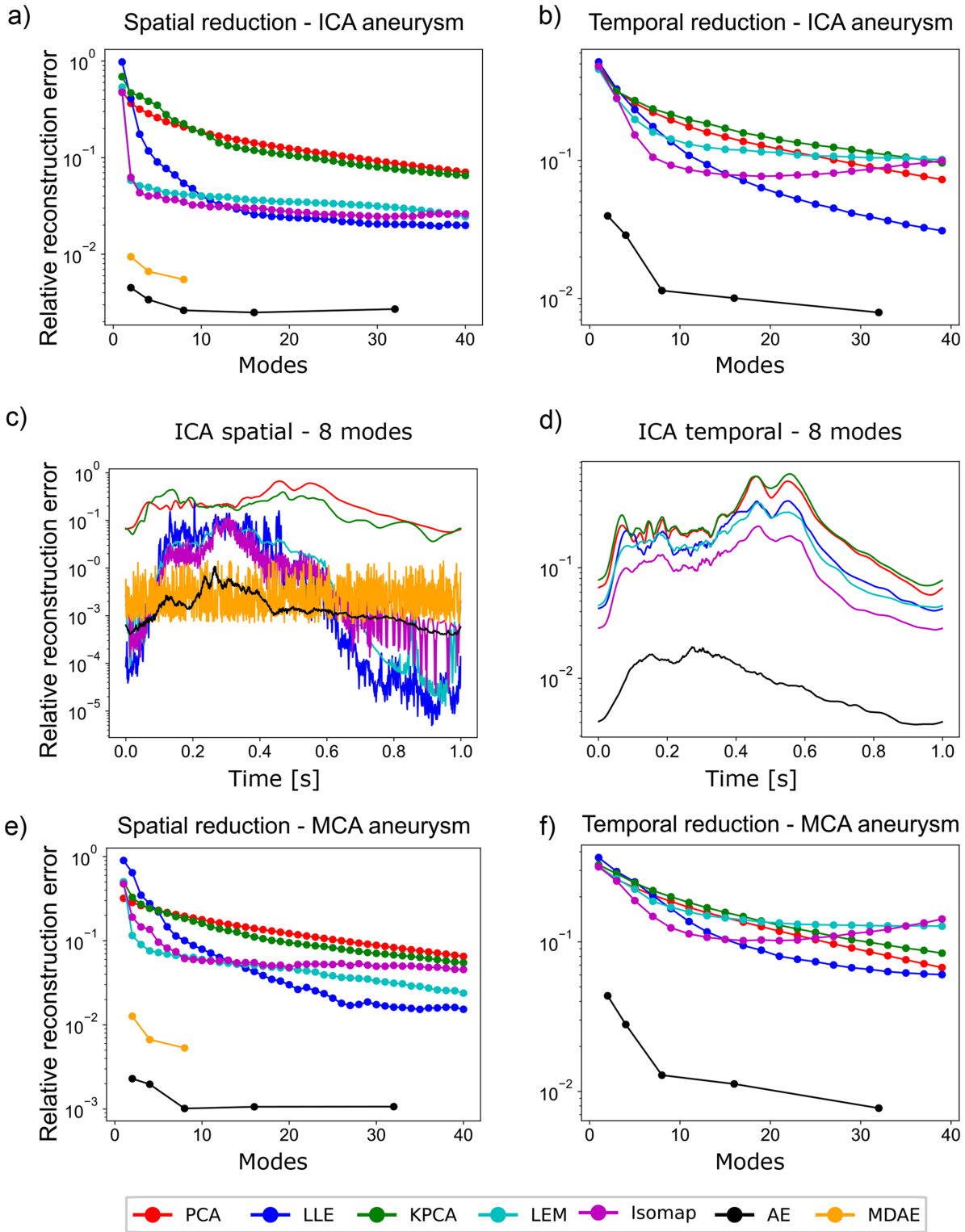

FIG. 8. Relative reconstruction error with a different number of modes is shown for (a) spatial and (b) temporal reduction of the ICA aneurysm case. Relative reconstruction error in the ICA case as a function of time is shown for (c) spatial and (d) temporal reduction using eight modes. Relative reconstruction error is shown for the MCA aneurysm for the (e) spatial and (f) temporal case. Note that MDAE is only used for spatial reduction. The temporal changes in error are not shown for the MCA case for the sake of brevity.

the other methods. Contrary to the spatial reduction, increasing latent space in AE leads to increasingly better reconstruction.

The spatial modes are shown in Fig. 9 for the ICA aneurysm. A representative plane [same as shown in Fig. 2(c)] is chosen for visualization of the modes. Two decomposed fields are shown for an MDAE model with a latent space size of two with two different snapshots (t_0 and t_{101}). PCA-1 highlights the main incoming jet creating an anticlockwise vortex structure in the aneurysm. The same structures are shown in LLE-1, KPCA-1, LEM-2, Isomap-1, AE-1, and MDAE-2-t_1. The core of the vortex is often emphasized, sometimes with low values (PCA-1, LLE-4, KPCA-1), other times with high values (LEM-3, Isomap-4). LEM-1 is highly distinct from all other modes as it separates the near-wall regions from the inside of the aneurysm. Interestingly, the AE modes 2–4 are not significantly different from each other, and they are similar to the AE-1 patterns with an opposite relative magnitude. The spatial modes are shown in Fig. 10 for a 2D cross section of the MCA aneurysm. In MDAE, the decomposed fields are shown for snapshot number 1 and 201. Flow enters the aneurysm on the left side of the cross section and creates a clockwise vortex. This vortex is highlighted in PCA-1, LLE-1, KPCA-1, Isomap-1, and AE-2. Interestingly, PCA-1 has the opposite relative magnitude pattern compared to the others. Similarly to the ICA case, LEM-1 distinguishes between the near-wall regions and the interior of the aneurysm. Some of the higher modes show more detailed flow structures (e.g., PCA-8, LLE-5,6, KPCA-5,6, and Isomap-3), which are harder to interpret, especially since they are 2D slices of 3D features.

IV. DISCUSSION

Nonlinear dimensionality reduction techniques have the capability of outperforming linear methods, such as PCA, for fluid flow data.[15,25,39,78] In this manuscript, we provided a detailed comparison of these methods in reconstructing unsteady fluid flow data. For the spatial reduction, AE and LLE achieved the best overall performance among all data sets. There were methods, like LEM, which performed well on one data set (turbulent channel flow) but poorly on others (cylinder flow). On the other hand, an evident advantage of PCA over nonlinear methods is the lack of hyperparameters.

Designing neural network architectures requires careful planning and investigation of multiple options (e.g., network size and depth, activation function, optimizer, and weight initialization). Manifold learning methods also have at least a few hyperparameters (e.g., number of nearest neighbors, kernel parameters, and regularization parameters). Each physical problem requires its own hyperparameter tuning. There have been methods proposed for the automatic choice of these parameters,[92] but well-developed and standard techniques are missing. These hyperparameters can affect the results presented in this study. For example, in AEs, if they are not sufficiently deep, manifold learning methods can achieve better performance. Also, different latent space sizes might lead to different optimal hyperparameters in AEs; therefore, the network has to be tuned separately for each latent space. Nevertheless, with proper parameter setup, the nonlinear methods can find better low-dimensional embeddings that describe the dynamics with fewer modes.

In spatial dimensionality reduction, all NDR methods achieved a smaller error compared to PCA for all test cases. For the flow over a cylinder case, KPCA, LEM, and Isomap seem to identify the previously reported underlying dynamics corresponding to a limit cycle.[83] However, an increase in the number of modes does not lead to better reconstruction errors. The same behavior was reported in Ref. 93 for LLE applied to the 1D Burgers' equation. This is a significant difference between PCA and NDR methods. In PCA, the reconstruction error decreases strictly monotonically (if all singular values are nonzero). However, in nonlinear methods, the reconstruction is not exact, even with the maximum number of modes. This ties back to the pre-image problem described in Ref. 66, which is usually an ill-defined problem, and only approximate solutions can be used.[94,95] Therefore, in some manifold learning techniques (NDR methods), a robust reconstruction method often cannot be achieved.

Our results show that the temporal reduction of unsteady fluid flow data seems to be a harder task compared to spatial reduction. In some cases, there is an order of magnitude difference in relative reconstruction error between temporal and spatial reconstruction. To achieve reasonable results for the neighborhood-based methods (LLE, LEM, and Isomap), the number of k-nearest neighbors used for the embedding had to be significantly higher (at least double, but in some cases an order of magnitude larger) than in the case of spatial reduction. This is because the number of spatial points in fluid flow data derived from computational fluid dynamics is usually much higher than the number of snapshots $n \gg m$.

Due to their high flexibility, the AE models had the best overall performance for all cases. LLE appeared to be comparable with AE for the 2D models (cylinder and channel flows), especially at a higher number of modes, but not for the 3D brain aneurysm cases. The AE model could even be further improved by using more recently proposed AE architectures.[29] Other extensions of manifold learning methods have also been developed (e.g., Hessian LLE,[96] discriminant kernel LLE (DKLLE),[97] landmark MDS using the Nystrom method,[98] and successive 1D LEM[99]). The relative success of these methods in fluid flow data still needs to be investigated. Additionally, most studies use the Euclidean distance to measure similarity between data points in manifold learning; however, other distance metrics can also be applied, which can influence the NDR methods' results.[100] More physically inspired similarity metrics could also be used (e.g., kinematic similarity[101]) for constructing the adjacency matrix, especially in the case of temporal dimensionality reduction.

Even though the temporal approach to dimensionality reduction leads to inferior results compared to spatial reduction, it has the immense advantage of producing visualizable modes, which were rarely leveraged for nonlinear methods in the past.[51,78] In fluid flow, these spatial modes highlight the main underlying coherent structures and flow features, which facilitate physical understanding and interpretability. All of the methods successfully captured the main dynamics of the flow around the cylinder as previously reported.[15,78] AE and LLE find notably more complicated structures than PCA. For the turbulent case, most methods (except LLE) produce visually similar modes, where higher order modes correspond to the smaller scale features. The differences in relative reconstruction error are likely due to the different reconstruction methods used. In the brain aneurysm cases, most methods identify the main flow structures, namely, the incoming blood stream and the intra-aneurysmal vortex.

NDR methods also have several drawbacks and limitations, which makes the use of these methods challenging in practice. As was discussed earlier, they need extensive hyperparameter tuning, data reconstruction can be difficult for certain methods, and they are computationally more expensive than PCA. For all cases, the velocity data

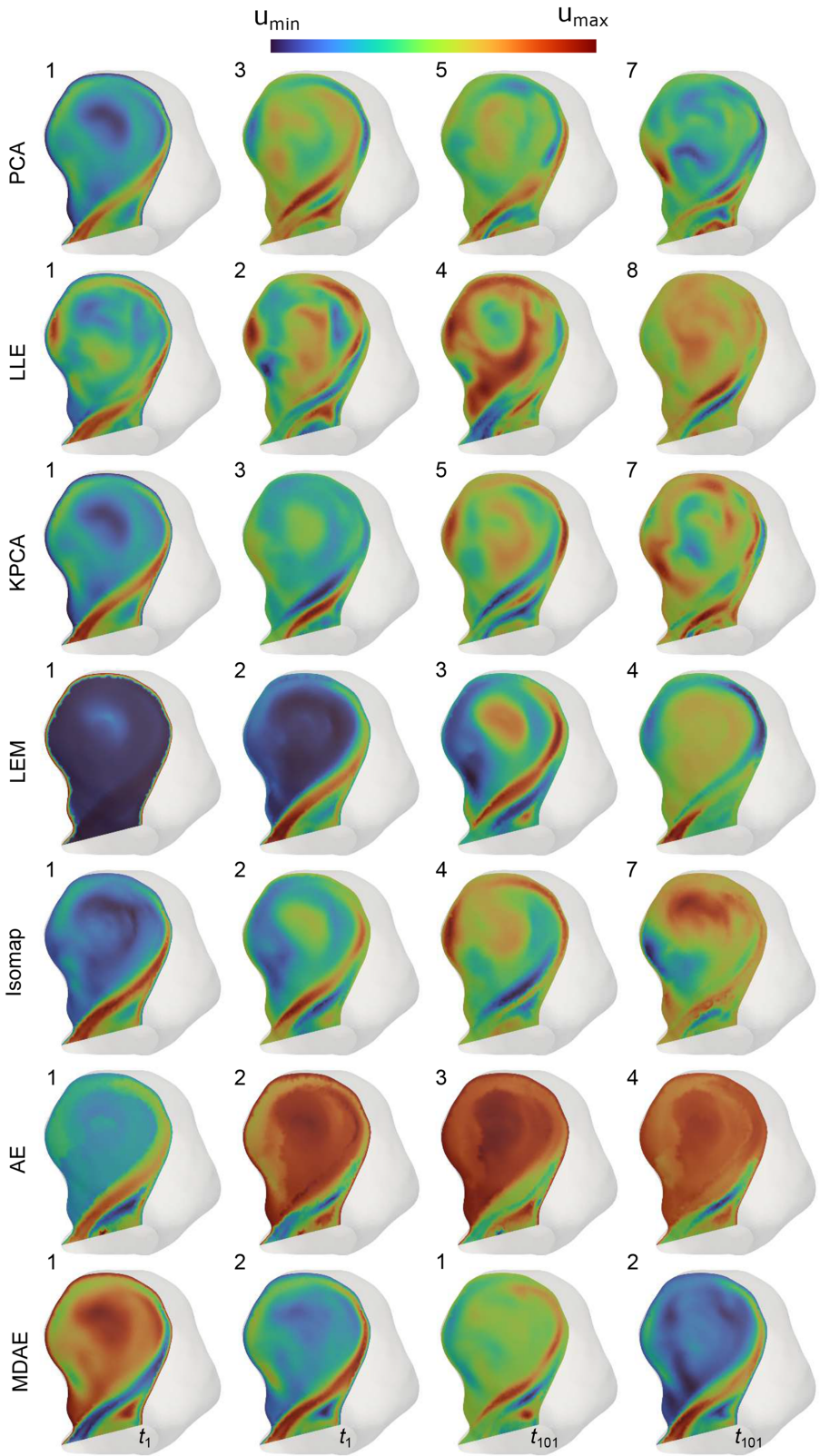

FIG. 9. Representative set of spatial modes visualized for the ICA aneurysm case. The inset number refers to the mode number, where higher number modes are less significant. The exceptions are the AEs, where the obtained modes are in an arbitrary order of importance. In MDAE, the snapshot number is specified. The color bar range is different for each mode, but only the qualitative structure of the modes is important.

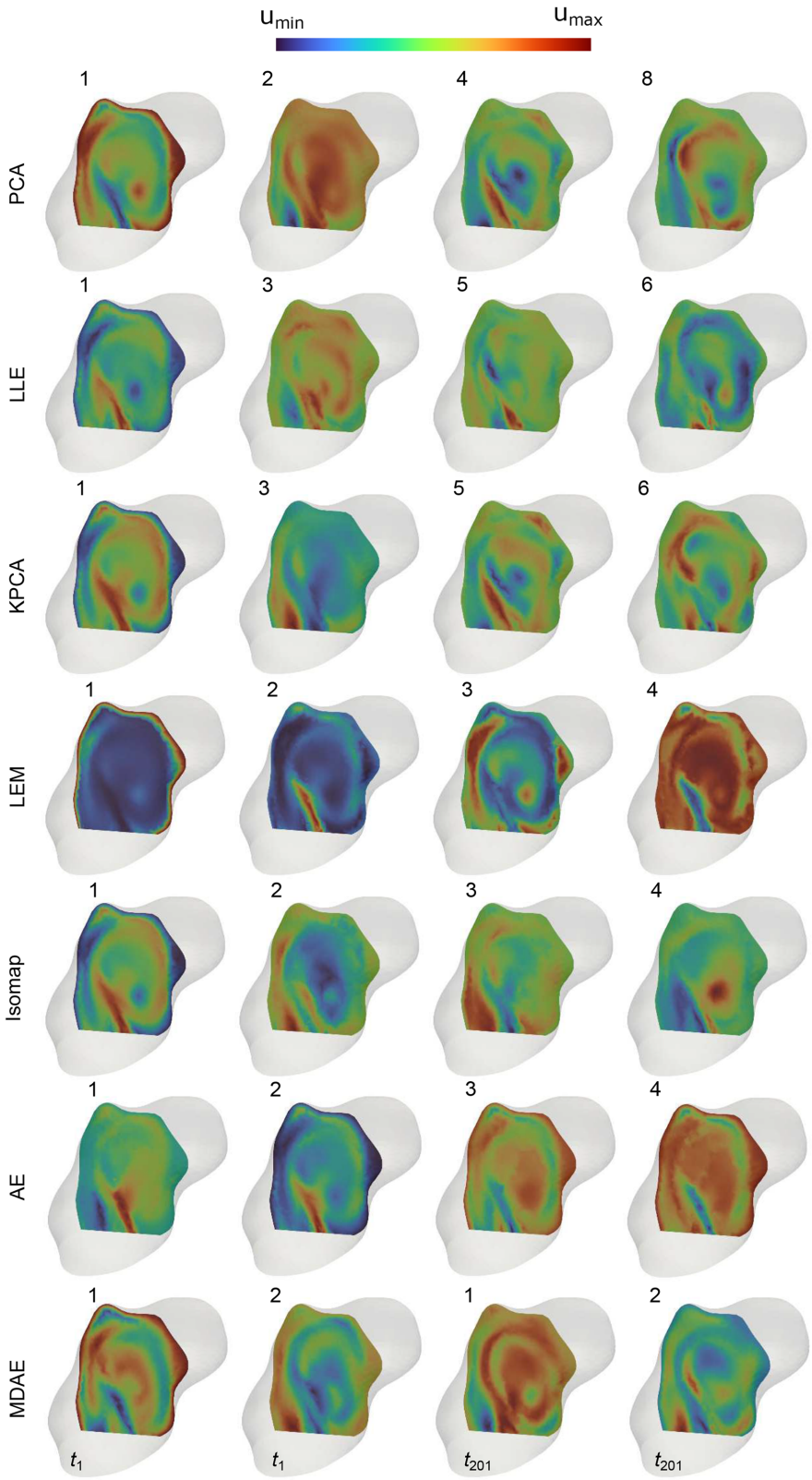

FIG. 10. Representative set of spatial modes visualized for the MCA aneurysm case. The inset number refers to the mode number, where higher number modes are less significant. The exceptions are the AEs, where the obtained modes are in an arbitrary order of importance. In MDAE, the snapshot number is specified. The color bar range is different for each mode, but only the qualitative structure of the modes is important.

obtained from the CFD simulations had to be downsampled to a coarser mesh in order to make the algorithms work in reasonable computational time, as also seen in other studies.[28] This is especially the case for temporal data arrangement and reduction, where the matrices obtained for eigenvector calculation are of size $n \times n$, where n is the number of mesh points. Another disadvantage compared to PCA is that NDR methods do not produce orthogonal modes, which could be an issue in a certain class of sparse data-driven modeling techniques (e.g., compressed sensing) that require the orthogonality property.[102] Ideas to overcome this issue for AEs and obtain near-orthogonal modes were presented in Ref. 103. In the case of AEs, another limitation is that the spatial modes come in an arbitrary order not sorted by importance as in the other methods. This was addressed in Ref. 104 by using a hierarchical AE to rank the modes. Finally, with the exception of autoencoders,[105] it is not clear how the other NDR methods could be used along with the governing equations to extrapolate the training data.

In conclusion, unsupervised learning models were presented and tested on reducing the spatial and temporal size of CFD simulation data. NDR methods clearly achieved better reconstruction error than traditional PCA for spatial reduction. However, reducing the temporal size proved to be more challenging and computationally demanding, where only AEs resulted in significantly improved embeddings. On the other hand, the temporal arrangement of the data led to spatial modes, which were qualitatively compared and physically interpreted. Some NDR methods discovered more complex mode structures compared to PCA. Our study presented one of the first comprehensive comparisons of various NDR methods for fluid flow datasets. The results encourage further studies to investigate the practical utility of these methods in data-driven ROM modeling of unsteady fluid flow.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Hunor Csala: Data curation (lead); Formal analysis (lead); Investigation (lead); Methodology (lead); Software (lead); Writing – original draft (lead); Writing – review & editing (supporting). **Scott Dawson:** Conceptualization (supporting); Investigation (supporting); Methodology (supporting); Writing – review & editing (supporting). **Amirhossein Arzani:** Conceptualization (lead); Methodology (supporting); Supervision (lead); Writing – original draft (supporting); Writing – review & editing (lead).

DATA AVAILABILITY

The Python source codes used to generate the results in the manuscript are available on GitHub https://github.com/amir-cardiolab/Nonlinear-dimensionality-reduction.

APPENDIX A: VECTOR DATA AND HYPERPARAMETERS

1. Vector data arrangement

In Sec. II A, the dimensionality reduction methods were presented and used with a data matrix $\mathbf{X}_{mag} \in \mathbb{R}^{n\times m}$ containing the velocity magnitudes at each mesh point. However, we can define the data matrix for vectorial data differently

$$\mathbf{X}_{mag} = \begin{bmatrix} \|\mathbf{u}_{11}\| & \|\mathbf{u}_{12}\| & \dots \\ \|\mathbf{u}_{21}\| & \|\mathbf{u}_{22}\| & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}, \quad \mathbf{X}_{col} = \begin{bmatrix} u_{11} & u_{12} & \dots \\ v_{11} & v_{12} & \dots \\ u_{21} & u_{22} & \dots \\ v_{21} & v_{22} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}, \quad \mathbf{X}_{row} = \begin{bmatrix} u_{11} & v_{11} & u_{12} & v_{12} & \dots \\ u_{21} & v_{21} & u_{22} & v_{22} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{A1}$$

The structure used in this paper based on velocity magnitude ($\mathbf{X}_{mag} \in \mathbb{R}^{n\times m}$) could be seen in the left side of Eq. (A1), where $\mathbf{u}_{ij}$ corresponds to velocity vector at spatial location i and time step j. The velocity vector $\mathbf{u} = (u, v, w)$ can also be used by stacking the velocity components below each other in the same column, resulting in a data matrix $\mathbf{X}_{col} \in \mathbb{R}^{dn\times m}$, where $d \in \{2, 3\}$ is the dimensionality of the velocity data. The structure of this matrix is shown in Eq. (A1) for $d = 2$. This is a well-established way of dealing with vectorial data using POD or DMD,[106] but it cannot be simply extended for all cases of manifold learning. When the goal is temporal dimensionality reduction, the rows of the data matrix are compared for finding the neighborhood relations. However, if vectorial data are stacked below each other, some rows will contain u velocities, while others v velocities; therefore, this will not lead to a fair comparison. To overcome this issue, the velocity components could be stacked row-wise to get a matrix $\mathbf{X}_{row} \in \mathbb{R}^{n\times dm}$, shown on the right side of Eq. (A1) for $d = 2$. This will lead to a fair comparison of the manifold learning methods, but after temporal dimensionality reduction, the u and v modes will be mixed, without a simple way to separate them. Therefore, this arrangement will not be suitable either.

The other option is to simply perform dimensionality reduction separately for u and v data, and this approach was previously used, for example, in Refs. 31 and 48. Here, we present a comparison of these different approaches in the case of PCA for flow over a cylinder as shown in Fig. 11. Namely, we compare the results based on the velocity magnitude analysis as performed in Sec. III of this paper [Fig. 11(a)], the results based on column-wise stacked velocity component [Fig. 11(b)], and the results based on a separate PCA analysis done on different components [Fig. 11(c)]. The top section of the figure shows the mode magnitude obtained by assembling the u and v modes shown in the bottom rows into a data array and calculating the magnitude.

The first difference observed between the velocity magnitude and vector-based modes is the background. In Fig. 11(a), the black background corresponds to zero velocity, and there are both positive and negative velocity features. In the other two columns, the gray background corresponds to zero, which is the lowest value since this is a magnitude calculated from the u and v modes; therefore, all features have a positive sign. To account for the difference in scale of the u and v modes, the singular vectors were multiplied with their corresponding singular values before plotting. The two velocity vector cases have almost identical mode structures, and some modes are shifted (e.g., mode u-3). However, the velocity magnitude-based modes are clearly distinct from the vector-based

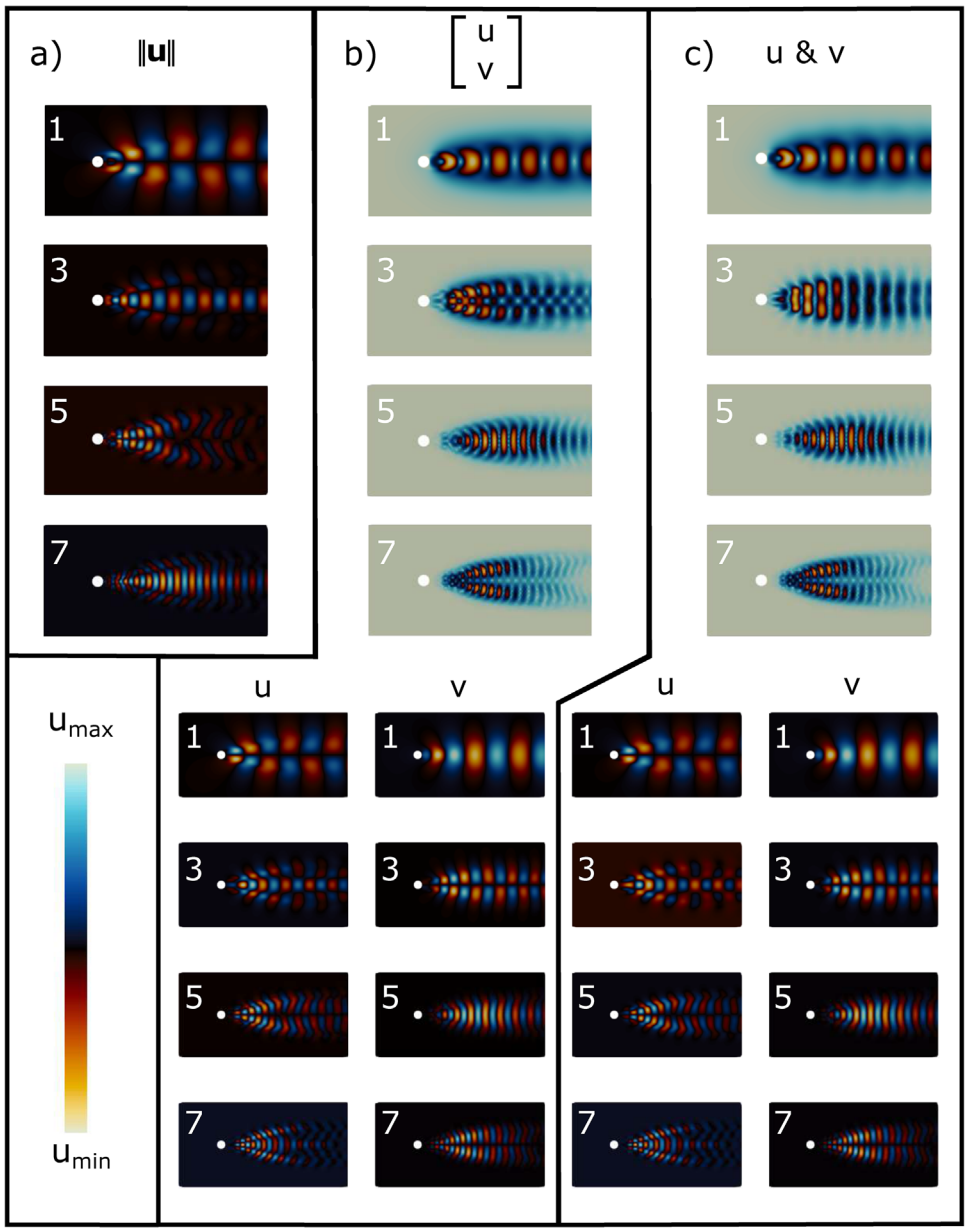

FIG. 11. The first four odd-numbered PCA modes for flow over a cylinder, (a) using the velocity magnitude, (b) using the velocity vector and stacking the components column-wise, (c) using the *u* and *v* velocity components separately. The top part shows the mode magnitudes, and the bottom part shows the *u* and *v* modes.

modes. The $\|\mathbf{u}\|$-based modes are dominated by the *u* velocity, while the vector-based modes are influenced more by the *v* velocity.

Stacking the velocity components column-wise is very convenient and can be tied to shared matrix factorization.[107] Nevertheless, it is not logical for most nonlinear methods during temporal dimensionality reduction.

2. Hyperparameters

In this section, the hyperparameters used in different NDR methods are presented. Table I lists the parameters used in the different manifold learning approaches, while Tables II–V list the autoencoder parameters for different test cases.

TABLE I. Manifold learning hyperparameters: k—number of nearest neighbors, λ and α—regularization parameters, γ—RBF kernel parameter.

NDR	LLE		KPCA		LEM			Isomap	
Parameters	k	λ	γ	α	k	γ	λ	k	λ
Cylinder—spatial	30	10^{-9}	10	10^{-3}	30	100	10^{-9}	50	10^{-9}
Cylinder—temporal	300	10^{-9}	5	10^{-3}	200	100	10^{-9}	200	10^{-9}
Turbulent—spatial	15	10^{-9}	20	1	15	10	10^{-9}	30	10^{-6}
Turbulent—temporal	700	10^{-9}	5	1	10^3	100	10^{-9}	800	10^{-9}
ICA—spatial	30	10^{-9}	6×10^{-4}	10^{-6}	20	0.01	10^{-9}	15	10^{-6}
ICA—temporal	300	10^{-6}	10^{-3}	10^{-3}	100	10^{-3}	10^{-6}	50	10^{-6}
MCA—spatial	50	10^{-9}	10^{-3}	10^{-6}	20	0.01	10^{-9}	15	10^{-6}
MCA—temporal	100	10^{-6}	10^{-3}	10^{-3}	50	10^{-3}	10^{-6}	50	10^{-6}

TABLE II. AE hyperparameters—cylinder case. Case name: AE type (AE/MDAE)—spatial (S) or temporal reduction (T)—latent space size, layers: encoder layers—bottleneck layers (always 1)—decoder layers.

Case	Encoder structure	Layers	Batch size
AE-S-2	3961-1024-256-64-8-2	5-1-5	128
AE-S-4	3961-1024-256-64-16-4	5-1-5	128
AE-S-8	3961-1024-256-64-8	4-1-4	128
MDAE-S-2	3961-1024-256-64-2	4-1-4	128
MDAE-S-4	3961-1024-256-64-4	4-1-4	128
MDAE-S-8	3961-1024-256-64-8	4-1-4	128
AE-T-2	1000-256-64-16-8-2	5-1-5	256
AE-T-4	1000-256-64-16-4	4-1-4	256
AE-T-6	1000-256-64-16-6	4-1-4	256
AE-T-8	1000-256-64-32-8	4-1-4	256
AE-T-12	1000-256-64-32-12	4-1-4	256

TABLE III. AE hyperparameters—turbulent channel case. Case name: AE type (AE/MDAE)—spatial (S) or temporal reduction (T)—latent space size, layers: encoder layers—bottleneck layers (always 1)—decoder layers.

Case	Encoder structure	Layers	Batch size
AE-S-2	16 384-4096-1024-256-32-8-2	6-1-6	256
AE-S-4	16 384-4096-1024-256-32-4	5-1-5	256
AE-S-8	16 384-4096-1024-256-32-8	5-1-5	256
AE-S-16	16 384-4096-1024-256-64-16	5-1-5	256
AE-S-32	16 384-4096-1024-256-32	4-1-4	256
AE-S-64	16 384-4096-1024-256-64	4-1-4	256
MDAE-S-2	16 384-4096-1024-256-64-16-2	6-1-6	256
MDAE-S-4	16 384-4096-1024-256-64-16-4	6-1-6	256
MDAE-S-8	16 384-4096-512-128-32-8	5-1-5	256
AE-T-2	2000-1000-500-128-16-2	5-1-5	256
AE-T-4	2000-500-200-100-32-4	5-1-5	256
AE-T-8	2000-500-200-100-32-8	5-1-5	256
AE-T-16	2000-1000-500-256-128-64-32-16	7-1-7	256
AE-T-32	2000-1000-500-256-128-64-32	6-1-6	256
AE-T-64	2000-1000-500-256-128-64	5-1-5	256

TABLE IV. AE hyperparameters—ICA case. Case name: AE type (AE/MDAE)—spatial (S) or temporal reduction (T)—latent space size, layers: encoder layers—bottleneck layers (always 1)—decoder layers.

Case	Encoder structure	Layers	Batch size
AE-S-2	40 500-2048-512-64-8-2	5-1-5	128
AE-S-4	40 500-8192-2048-512-128-32-8-4	7-1-7	128
AE-S-8	40 500-8192-2048-512-128-32-8	6-1-6	128
AE-S-16	40 500-8192-2048-512-128-32-16	6-1-6	128
AE-S-32	40 500-8192-2048-512-128-64-32	6-1-6	128
MDAE-S-2	40 500-4096-1024-256-64-16-2	6-1-6	128
MDAE-S-4	40 500-8192-2048-512-128-32-8-4	7-1-7	128
MDAE-S-8	40 500-8192-2048-512-128-32-8	6-1-6	128
AE-T-2	1000-512-128-32-8-2	5-1-5	256
AE-T-4	1000-512-256-128-64-32-16-8-4	8-1-8	256
AE-T-8	1000-512-256-128-64-32-16-8	7-1-7	256
AE-T-16	1000-512-256-128-64-32-16	6-1-6	256
AE-T-32	1000-512-256-128-64-32	5-1-5	256

TABLE V. AE hyperparameters—MCA case. Case name: AE type (AE/MDAE)—spatial (S) or temporal reduction (T)—latent space size, layers: encoder layers—bottleneck layers (always 1)—decoder layers.

Case	Encoder structure	Layers	Batch size
AE-S-2	10 167-2048-512-128-32-8-2	6-1-6	128
AE-S-4	10 167-2048-512-128-32-4	5-1-5	128
AE-S-8	10 167-2048-512-64-8	4-1-4	128
AE-S-16	10 167-2048-512-128-16	4-1-4	128
AE-S-32	10 167-2048-512-128-32	4-1-4	128
MDAE-S-2	10 167-4096-1024-256-64-16-2	6-1-6	128
MDAE-S-4	10 167-4096-1024-256-64-16-4	6-1-6	128
MDAE-S-8	10 167-2048-512-64-8	4-1-4	128
AE-T-2	1000-512-128-32-8-2	5-1-5	256
AE-T-4	1000-512-256-128-64-32-16-8-4	8-1-8	256
AE-T-8	1000-512-256-128-64-32-16-8	7-1-7	256
AE-T-16	1000-512-256-128-64-32-16	6-1-6	256
AE-T-32	1000-512-256-128-64-32	5-1-5	256

REFERENCES

[1]S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," Annu. Rev. Fluid Mech. **52**, 477–508 (2020).
[2]R. Vinuesa and S. L. Brunton, "The potential of machine learning to enhance computational fluid dynamics," arXiv:2110.02085 (2021).
[3]F. Ren, H. B. Hu, and H. Tang, "Active flow control using machine learning: A brief review," J. Hydrodyn. **32**(2), 247–253 (2020).
[4]K. Taira, S. L. Brunton, S. T. M. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley, "Modal analysis of fluid flows: An overview," AIAA J. **55**(12), 4013–4041 (2017).
[5]A. E. Deane, I. G. Kevrekidis, G. E. Karniadakis, and S. A. Orszag, "Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders," Phys. Fluids A: Fluid Dyn. **3**(10), 2337–2354 (1991).
[6]T. Lassila, A. Manzoni, A. Quarteroni, and G. Rozza, "Model order reduction in fluid dynamics: Challenges and perspectives," *Reduced Order Methods for Modeling and Computational Reduction* (Springer, 2014), pp. 235–273.
[7]G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," Annu. Rev. Fluid Mech. **25**(1), 539–575 (1993).
[8]P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley, *Turbulence, Coherent Structures, Dynamical Systems and Symmetry* (Cambridge University Press, 2012).
[9]N. Aubry, P. Holmes, J. L. Lumley, and E. Stone, "The dynamics of coherent structures in the wall region of a turbulent boundary layer," J. Fluid Mech. **192**, 115–173 (1988).
[10]J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems* (SIAM, 2016).
[11]P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," J. Fluid Mech. **656**, 5–28 (2010).
[12]L. Perret, E. Collin, and J. Delville, "Polynomial identification of POD based low-order dynamical system," J. Turbul. **7**, N17 (2006).
[13]D. J. Lucia, P. S. Beran, and W. A. Silva, "Aeroelastic system development using proper orthogonal decomposition and Volterra theory," J. Aircr. **42**(2), 509–518 (2005).
[14]B. Glaz, L. Liu, and P. P. Friedmann, "Reduced-order nonlinear unsteady aerodynamic modeling using a surrogate-based recurrence framework," AIAA J. **48**(10), 2418–2429 (2010).
[15]A. Ehlert, C. N. Nayeri, M. Morzynski, and B. R. Noack, "Locally linear embedding for transient cylinder wakes," arXiv:1906.07822 (2019).
[16]J. L. Lumley, "The structure of inhomogeneous turbulent flows," *Atmospheric Turbulence and Radio Wave Propagation* (Nauka, Moscow, Russia, 1967).
[17]S. E. Ahmed, S. Pawar, O. San, A. Rasheed, T. Iliescu, and B. R. Noack, "On closures for reduced order models—A spectrum of first-principle to machine-learned avenues," Phys. Fluids **33**(9), 091301 (2021).
[18]J. Yu, C. Yan, and M. Guo, "Non-intrusive reduced-order modeling for fluid problems: A brief review," Proc. Inst. Mech. Eng., Part G: J. Aerosp. Eng. **233**(16), 5896–5912 (2019).
[19]P. J. Schmid, "Dynamic mode decomposition and its variants," Annu. Rev. Fluid Mech. **54**, 225–254 (2022).
[20]H. Eivazi, H. Veisi, M. H. Naderi, and V. Esfahanian, "Deep neural networks for nonlinear model order reduction of unsteady flows," Phys. Fluids **32**(10), 105104 (2020).
[21]N. B. Erichson, L. Mathelin, Z. Yao, S. L. Brunton, M. W. Mahoney, and J. N. Kutz, "Shallow neural networks for fluid flow reconstruction with limited sensors," Proc. R. Soc. A **476**(2238), 20200097 (2020).
[22]M. Cheng, F. Fang, C. C. Pain, and I. M. Navon, "An advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modelling," Comput. Methods Appl. Mech. Eng. **372**, 113375 (2020).
[23]R. Lopez and J. Atzberger, "Variational autoencoders for learning nonlinear dynamics of physical systems," arXiv:2012.03448 (2020).
[24]J. Qu, W. Cai, and Y. Zhao, "Deep learning method for identifying the minimal representations and nonlinear mode decomposition of fluid flows," Phys. Fluids **33**(10), 103607 (2021).
[25]L. Agostini, "Exploration and prediction of fluid dynamical systems using auto-encoder technology," Phys. Fluids **32**(6), 067103 (2020).
[26]K. Fukami, K. Hasegawa, T. Nakamura, M. Morimoto, and K. Fukagata, "Model order reduction with neural networks: Application to laminar and turbulent flows," SN Comput. Sci. **2**(6), 467 (2021).
[27]T. Nakamura, K. Fukami, K. Hasegawa, Y. Nabae, and K. Fukagata, "Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow," Phys. Fluids **33**(2), 025116 (2021).
[28]P. Pant, R. Doshi, P. Bahl, and A. Barati Farimani, "Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations," Phys. Fluids **33**(10), 107101 (2021).
[29]A. Gruber, M. Gunzburger, L. Ju, and Z. Wang, "A comparison of neural network architectures for data-driven reduced-order modeling," Comput. Methods Appl. Mech. Eng. **393**, 114764 (2022).
[30]J. Tencer and K. Potter, "A tailored convolutional neural network for nonlinear manifold learning of computational physics data using unstructured spatial discretizations," SIAM J. Sci. Comput. **43**(4), A2581–A2613 (2021).
[31]W. Obayashi, H. Aono, T. Tatsukawa, and K. Fujii, "Feature extraction of fields of fluid dynamics data using sparse convolutional autoencoder," AIP Adv. **11**(10), 105211 (2021).
[32]N. B. Erichson, M. Muehlebach, and M. W. Mahoney, "Physics-informed autoencoders for Lyapunov-stable fluid flow prediction," arXiv:1905.10866 (2019).
[33]H. Gao, L. Sun, and J.-X. Wang, "Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels," Phys. Fluids **33**(7), 073603 (2021).
[34]S. Kneer, T. Sayadi, D. Sipp, P. Schmid, and G. Rigas, "Symmetry-aware autoencoders: s-PCA and s-nlPCA," arXiv:2111.02893 (2021).
[35]C. E. Heaney, Z. Wolffs, J. A. Tómasson, L. Kahouadji, P. Salinas, A. Nicolle, O. K. Matar, I. M. Navon, N. Srinil, and C. C. Pain, "An AI-based domain-decomposition non-intrusive reduced-order model for extended domains applied to multiphase flow in pipes," arXiv:2202.06170 (2022).
[36]S. Dutta, P. Rivera-Casillas, B. Styles, and M. W. Farthing, "Reduced order modeling using advection-aware autoencoders," Math. Comput. Appl. **27**(3), 34 (2022).
[37]J. A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction* (Springer, 2007).
[38]R. Vidal, Y. Ma, and S. S. Sastry, *Generalized Principal Components Analysis* (Springer, 2016).
[39]L. Pyta and D. Abel, "Nonlinear model reduction of the Navier-Stokes-equations," in *Proceedings of the American Control Conference (ACC)* (IEEE, 2016), pp. 5249–5254.
[40]E. Farzamnik, A. Ianiro, S. Discetti, N. Deng, K. Oberleithner, B. R. Noack, and V. Guerrero, "From snapshots to manifolds—A tale of shear flows," arXiv:2203.14781 (2022).
[41]M. A. Mendez, J. Dominique, M. Fiore, F. Pino, P. Sperotto, and J. Berghe, "Challenges and opportunities for machine learning in fluid mechanics," arXiv:2202.12577 (2022).
[42]F. Tauro, S. Grimaldi, and M. Porfiri, "Unraveling flow patterns through nonlinear manifold learning," PLoS One **9**(3), e91131 (2014).
[43]K. Taira and A. G. Nair, "Network-based analysis of fluid flows: Progress and outlook," Prog. Aerosp. Sci. **131**, 100823 (2022).
[44]G. Iacobello, L. Ridolfi, and S. Scarsoglio, "A review on turbulent and vortical flow analyses via complex networks," Physica A **563**, 125476 (2021).
[45]A. G. Nair, S. L. Brunton, and K. Taira, "Networked-oscillator-based modeling and control of unsteady wake flows," Phys. Rev. E **97**(6), 063107 (2018).
[46]K. Decker, H. D. Schwartz, and D. Mavris, "Dimensionality reduction techniques applied to the design of hypersonic aerial systems," AIAA Paper No. 2020-3003, 2020, p. 3003.
[47]T. Franz, R. Zimmermann, S. Görtz, and N. Karcher, "Interpolation-based reduced-order modelling for steady transonic flows via manifold learning," Int. J. Comput. Fluid Dyn. **28**(3–4), 106–121 (2014).
[48]R. Halder, K. Fidkowski, and K. Maki, "Local non-intrusive reduced order modeling using isomap," AIAA Paper No. 2022-0081, 2022, p. 0081.
[49]P. Díez, A. Muixí, S. Zlotnik, and A. García-González, "Nonlinear dimensionality reduction for parametric problems: A kernel proper orthogonal decomposition," Int. J. Numer. Methods Eng. **122**(24), 7306–7327 (2021).

[50]M. Rovira, K. Engvall, and C. Duwig, "Identifying key features in reactive flows: A tutorial on combining dimensionality reduction, unsupervised clustering, and feature correlation," Chem. Eng. J. **438**, 135250 (2022).
[51]A. C. Nunno, B. A. Perry, J. F. Macart, and M. E. Mueller, "Data-driven dimension reduction in turbulent combustion: Utility and limitations," AIAA Paper No. 2019-2010, 2019, p. 2010.
[52]J. Wu, J. Wang, H. Xiao, and J. Ling, "Visualization of high dimensional turbulence simulation data using t-SNE," AIAA Paper No. 2017-1770, 2017, p. 1770.
[53]G. Di Labbio and L. Kadem, "Reduced-order modeling of left ventricular flow subject to aortic valve regurgitation," Phys. Fluids **31**(3), 031901 (2019).
[54]M. Habibi, S. Dawson, and A. Arzani, "Data-driven pulsatile blood flow physics with dynamic mode decomposition," Fluids **5**(3), 111 (2020).
[55]T. B. Le, "Dynamic modes of inflow jet in brain aneurysms," J. Biomech. **116**, 110238 (2021).
[56]M. Habibi, R. M. D'Souza, S. T. M. Dawson, and A. Arzani, "Integrating multi-fidelity blood flow data with reduced-order data assimilation," Comput. Biol. Med. **135**, 104566 (2021).
[57]L. Van Der Maaten, E. Postma, J. Van den Herik *et al.*, "Dimensionality reduction: A comparative review," J. Mach. Learn. Res. **10**(66–71), 13 (2009).
[58]Q. Zhang, Y. Shang, and G. Zhang, "pyDRmetrics—A python toolkit for dimensionality reduction quality assessment," Heliyon **7**(2), e06199 (2021).
[59]F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," J. Mach. Learn. Res. **12**, 2825–2830 (2011).
[60]K. Pearson, "On lines and planes of closest fit to systems of points in space," London Edinburgh Dublin Philos. Mag. J. Sci. **2**(11), 559–572 (1901).
[61]S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," Science **290**(5500), 2323–2326 (2000).
[62]L. K. Saul and S. T. Roweis, see http://www.cs.columbia.edu/~jebara/6772/papers/lleintro.pdf for "An Introduction to Locally Linear Embedding," 2000.
[63]B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Locally linear embedding and its variants: Tutorial and survey," arXiv:2011.10925 (2020).
[64]L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifolds," J. Mach. Learn. Res. **4**, 119–155 (2003).
[65]B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," Neural Comput. **10**(5), 1299–1319 (1998).
[66]P. Honeine and C. Richard, "Preimage problem in kernel-based machine learning," IEEE Signal Process. Mag. **28**(2), 77–88 (2011).
[67]G. H. Bakir, J. Weston, and B. Schölkopf, "Learning to find pre-images," Adv. Neural Inf. Process. Syst. **16**, 449–456 (2004).
[68]B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Unified framework for spectral dimensionality reduction, maximum variance unfolding, and kernel learning by semidefinite programming: Tutorial and survey," arXiv:2106.15379 (2021).
[69]J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *Proceedings of the Twenty-First International Conference on Machine Learning* (ICML, 2004), p. 47.
[70]M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," Neural Comput. **15**(6), 1373–1396 (2003).
[71]B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Laplacian-based dimensionality reduction including spectral clustering, Laplacian eigenmap, locality preserving projection, graph embedding, and diffusion map: Tutorial and survey," arXiv:2106.02154 (2021).
[72]A. Cloninger, W. Czaja, and T. Doster, "The pre-image problem for Laplacian eigenmaps utilizing l1 regularization with applications to data fusion," Inverse Probl. **33**(7), 074006 (2017).
[73]J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," Science **290**(5500), 2319–2323 (2000).
[74]W. S. Torgerson, "Multidimensional scaling. I. Theory and method," Psychometrika **17**(4), 401–419 (1952).
[75]B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Multidimensional scaling, sammon mapping, and isomap: Tutorial and survey," arXiv:2009.08136 (2020).
[76]T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (MIT Press, 2022).
[77]E. Plaut, "From principal subspaces to principal components with linear autoencoders," arXiv:1804.10253 (2018).
[78]T. Murata, K. Fukami, and K. Fukagata, "Nonlinear mode decomposition with convolutional neural networks for fluid dynamics," J. Fluid Mech. **882**, A13 (2020).
[79]K. Hasegawa, K. Fukami, and K. Fukagata, "Visualization of nonlinear modal structures for three-dimensional unsteady fluid flows with customized decoder design," in Proceedings of the Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS) (2021).
[80]A. Muneer, S. M. Taib, S. M. Fati, A. O. Balogun, and I. A. Aziz, "A hybrid deep learning-based unsupervised anomaly detection in high dimensional data," Comput. Mater. Continua **70**(3), 6073–6088 (2022).
[81]Q. Gu and J. Zhou, "Co-clustering on manifolds," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2009), pp. 359–368.
[82]M. Rezaei-Ravari, M. Eftekhari, and F. Saberi-Movahed, "Regularizing extreme learning machine by dual locally linear embedding manifold learning for training multi-label neural network classifiers," Eng. Appl. Artif. Intell. **97**, 104062 (2021).
[83]B. R. Noack, K. Afanasiev, M. Morzyński, G. Tadmor, and F. Thiele, "A hierarchy of low-dimensional models for the transient and post-transient cylinder wake," J. Fluid Mech. **497**, 335–363 (2003).
[84]A. Logg, K.-A. Mardal, and G. Wells, *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book* (Springer Science and Business Media, 2012), Vol. 84.
[85]J. Graham, K. Kanov, X. I. A. Yang, M. Lee, N. Malaya, C. C. Lalescu, R. Burns, G. Eyink, A. Szalay, R. D. Moser *et al.*, "A web services accessible database of turbulent channel flow and its use for testing a new integral wall model for LES," J. Turbul. **17**(2), 181–215 (2016).
[86]Y. Li, E. Perlman, M. Wan, Y. Yang, C. Meneveau, R. Burns, S. Chen, A. Szalay, and G. Eyink, "A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence," J. Turbul. **9**, N31 (2008).
[87]L. M. Sangalli, P. Secchi, and S. Vantini, "AneuRisk65: A dataset of three-dimensional cerebral vascular geometries," Electron. J. Stat. **8**(2), 1879–1890 (2014).
[88]A. Updegrove, N. M. Wilson, J. Merkow, H. Lan, A. L. Marsden, and S. C. Shadden, "Simvascular: An open source pipeline for cardiovascular simulation," Ann. Biomed. Eng. **45**(3), 525–541 (2017).
[89]Y. Hoi, B. A. Wasserman, Y. J. Xie, S. S. Najjar, L. Ferruci, E. G. Lakatta, G. Gerstenblith, and D. A. Steinman, "Characterization of volumetric flow rate waveforms at the carotid bifurcations of older adults," Physiol. Meas. **31**(3), 291 (2010).
[90]K. Valen-Sendstad, M. Piccinelli, R. KrishnankuttyRema, D. Steinman *et al.*, "Estimation of inlet flow rates for image-based aneurysm CFD models: Where and how to begin?," Ann. Biomed. Eng. **43**(6), 1422–1431 (2015).
[91]K. Valen-Sendstad, K.-A. Mardal, M. Mortensen, B. A. P. Reif, and H. P. Langtangen, "Direct numerical simulation of transitional flow in a patient-specific intracranial aneurysm," J. Biomech. **44**(16), 2826–2832 (2011).
[92]J. Valencia-Aguirre, A. Álvarez-Mesa, G. Daza-Santacoloma, and G. Castellanos-Domínguez, "Automatic choice of the number of nearest neighbors in locally linear embedding," in *Iberoamerican Congress on Pattern Recognition* (Springer, 2009), pp. 77–84.
[93]J. C. Winstead, "Nonlinear model reduction based on manifold learning with application to the Burgers' equation," Master's thesis (University of Tennessee, 2017).
[94]J. T.-Y. Kwok and I. W.-H. Tsang, "The pre-image problem in kernel methods," IEEE Trans. Neural Networks **15**(6), 1517–1525 (2004).
[95]N. D. Monnig, B. Fornberg, and F. G. Meyer, "Inverting nonlinear dimensionality reduction with scale-free radial basis function interpolation," Appl. Comput. Harmon. Anal. **37**(1), 162–170 (2014).
[96]D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," Proc. Natl. Acad. Sci. **100**(10), 5591–5596 (2003).

[97]X. Zhao and S. Zhang, "Facial expression recognition using local binary patterns and discriminant kernel locally linear embedding," EURASIP J. Adv. Signal Process. **2012**(1), 1–9.
[98]C. J. C. Burges, *Dimension Reduction: A Guided Tour* (Now Publishers, Inc., 2010).
[99]S. Gerber, T. Tasdizen, and R. Whitaker, "Robust non-linear dimensionality reduction using successive 1-dimensional Laplacian eigenmaps," in *Proceedings of the 24th International Conference on Machine Learning* (ICML, 2007), pp. 281–288.
[100]M. Naseer and S.-Y. Qin, "Performance comparison of nonlinear dimensionality reduction methods for image data using different distance measures," in *Proceedings of the International Conference on Computational Intelligence and Security* (IEEE, 2008).
[101]K. L. Schlueter-Kuck and J. O. Dabiri, "Coherent structure colouring: Identification of coherent structures from sparse data using graph theory," J. Fluid Mech. **811**, 468–486 (2017).
[102]A. Arzani and S. T. M. Dawson, "Data-driven cardiovascular flow modelling: Examples and opportunities," J. R. Soc. Interface **18**(175), 20200802 (2021).
[103]H. Eivazi, S. Le Clainche, S. Hoyas, and R. Vinuesa, "Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows," Expert Syst. Appl. **202**, 117038 (2022).
[104]K. Fukami, T. Nakamura, and K. Fukagata, "Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data," Phys. Fluids **32**(9), 095110 (2020).
[105]K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," Proc. Natl. Acad. Sci. **116**(45), 22445–22451 (2019).
[106]S. L. Brunton and J. N. Kutz, *Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2022).
[107]C. C. Aggarwal, *Linear Algebra and Optimization for Machine Learning* (Springer, 2020), Vol. 156.