Machine learning for fluid flow reconstruction from limited measurements

Pierre Dubois, Thomas Gomez, Laurent Planckaert, Laurent Perret

Machine learning for fluid flow reconstruction from limited measurements

Pierre Dubois[a,*], Thomas Gomez[a], Laurent Planckaert[a], Laurent Perret[b]

[a] *Univ. Lille, CNRS, ONERA, Arts et Metiers Institute of Technology, Centrale Lille, UMR 9014 - LMFL - Laboratoire de Mécanique des fluides de Lille - Kampé de Fériet, F-59000 Lille, France*
[b] *Centrale Nantes, LHEEA UMR CNRS 6598, Nantes, France*

Abstract

This paper investigates the use of data-driven methods for the reconstruction of unsteady fluid flow fields. The proposed framework is based on the combination of machine learning tools: dimensionality reduction to extract dominant spatial directions from data, reconstruction algorithm to recover encoded data by limited measurements and cross-validation for hyperparameter optimization. For the encoding part, linear and nonlinear extraction of patterns are considered: proper orthogonal decomposition (POD), linear autoencoder (LAE) and variational autoencoder (VAE). For the reconstruction part, regressive reconstruction (neural network, linear, support vector, gradient boosting) and library-based reconstruction are compared, each method being cross-validated to ensure good generalization on testing data. The position of sensors is optimized using an enhanced clustering algorithm. The robustness of regressive reconstructions to noise measurements is also addressed, showing the benefits of variational approaches in the reduction phase. The strategy is tested for three increasing complexity flows: 2D vortex shedding ($Re = 200$), 2D spatial mixing layer and 3D vortex shedding ($Re = 20000$). The results suggest that proper machine learning approaches to fluid flow data can lead to effective reconstruction models that can be used for the rapid estimation of complex flows.

Keywords: data-driven, machine learning, dimensionality reduction

1. Introduction

1.1. Flow reconstruction

Understanding fluid physics is possible through numerical or experimental analysis of canonical flows. Both approaches are however limited: accurate numerical solving of Navier Stokes equations is computationally expensive [1] and experimental flow visualization requires significant resources [2]. As a consequence, reconstruction methods have emerged: limited measurements at time t are used to estimate the full velocity field at the same time [3]. Mathematically, the problem consists in finding the relationship $\mathcal{G}$ between punctual measurements $\boldsymbol{y}(\boldsymbol{x_c}, t) \in \mathbb{R}^p$ and the complete flow field $\boldsymbol{U}(\boldsymbol{x}, t) \in \mathbb{R}^n$ where p denotes the number of sensors, $\boldsymbol{x}_c$ their locations and n is the dimension of the field (see Fig 1). The measurement operator $\mathcal{H}$ being likely ill conditioned thus not invertible, data-driven procedures have been developed to estimate the $\mathcal{G}$ operator from data [4].

A first possible approach is the regressive reconstruction. Supervised learning methods [5] are used to infer the mapping between the measurement space and the field space. Given a formulation for the $\mathcal{G}$ operator, a cost function evaluating the error between training examples (snapshots with associated measurements) and their reconstruction is minimized. Earlier investigations of regressive reconstruction include the stochastic estimation (SE) where the reconstructed field is a multi-linear function of

*Corresponding author: pierre.dubois@onera.fr

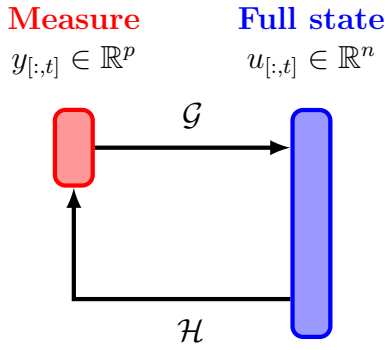


Figure 1: Representation of the reconstruction problem: recover the high dimensional velocity field from limited measurements

available measurements [6]. SE has been applied to a wide variety of flows, including isotropic turbulence [7], boundary layers [8], open cavities [9], backward-facing steps [10], jets [11] and urban flows [12]. A recent extension is proposed by Erichson *et al.* [13] with the regressive function being nonlinear and defined by a shallow neural network. The authors have successfully estimated the vorticity field in the wake of a 2D cylinder using very few wall pressure measurements.

A second possibility is the direct reconstruction based on an optimization problem $\mathcal{P}$ [14]. In this model-free method, limited measurements are used to estimate the flow field as a linear combination of reference modes [15]. These modes are often tailored to the considered flow and found by modal decomposition [16]. Gappy POD [17] is a popular algorithm: reference modes are dominant orthogonal directions of the flow (POD modes) and the best linear combination is found by solving a least-square problem built on masked data. Lusseyran *et al.* [18] used this technique to reconstruct the 3D flow past a cavity, choosing as many sensors as reference modes to obtain a unique solution. A recent extension is GOBAL [19] where reference modes balance the controllability and the observability of the considered flow. Deep learning is also an emerging tool to build libraries: taking advantage of neural networks flexibility, nonlinear manifolds capturing dominant flow structures can be leveraged [20] [21]. Finally, enforcing sparsity in the solution (compressed sensing and sparse reconstruction) is a usual technique to improve the robustness to measurement corruption. This however requires prohibitively many measurements and expensive computations, leading to the development of local reconstruction strategies [3].

The last category of flow field estimation is data-assimilation [22]. In this approach, a dynamical model evolves the field estimate while measurements improve the forecasts. The dynamical model may be a reduced-order approximation of Navier-Stokes equations, found by a Galerkin projection onto a data-driven basis [23] or by model identification [24]. Typical applications of data-assimilation include the use of Kalman filters for mean flow reconstruction [25] and full flow field estimates [26].

The three approaches to obtain a data-driven estimation (denoted $\widehat{\boldsymbol{U}}$) of the velocity field $\boldsymbol{U}$ are summarized in table 1. For the regressive reconstruction, the operator $\mathcal{G}$ is directly inferred from data, thus requiring a machine learning procedure. For the direct reconstruction, the $\mathcal{G}$ operator is evaluated at a given measurement vector via the use of convex optimization tools. For the data assimilation, measurements update predictions of a dynamical model via a least square combination of prediction and measurements errors.

1.2. Dimensionality reduction

The field to estimate is $\mathbf{U} \in \mathbb{R}^n$ where n is the dimension. Despite complex spatio-temporal dynamics, fluid systems exhibit low dimensional features making relevant the use of dimensionality reduction. The objective is to extract spatial patterns characterizing the fluid flow: vortex shedding for wake flows, Kelvin Helmotz instability for shear layers, coherent structures for boundary layers, etc [27]. This extraction may be performed by linear or nonlinear encoding transformations e. If the decoding transformation d is known, the reconstruction problem reduces to the estimation of latent structures dynamics $a(t) \in \mathbb{R}^r$ via the

Method	Idea	Description	Formulation
Direct reconstruction	evaluate $\mathcal{G}(\boldsymbol{y})$	solve optimisation problem $\mathcal{P}$	$\widehat{\boldsymbol{U}} = \min_{\boldsymbol{U}} \mathcal{P}[\mathcal{H}(\boldsymbol{U})]$
Regressive reconstruction	estimate $\mathcal{G}$	learn operator $\mathcal{G}$	$\widehat{\boldsymbol{U}} = \mathcal{G}(\boldsymbol{y})$
Data assimilation	update predictions using $\boldsymbol{y}$	propagate state (get background $\boldsymbol{U}^b$) and analyze the forecast (kalman gain K)	$\begin{cases} \text{Prediction} \rightarrow \boldsymbol{U}^b \\ \text{Analysis} \rightarrow \widehat{\boldsymbol{U}} = \boldsymbol{U}^b + K[\boldsymbol{y} - \mathcal{H}(\boldsymbol{U})] \end{cases}$

Table 1: Reconstruction algorithms from the literature. Data assimilation is not used in this paper.

f operator, as illustrated in figure 2. The $\mathcal{G}$ operator is then decomposed into $d \circ f$.

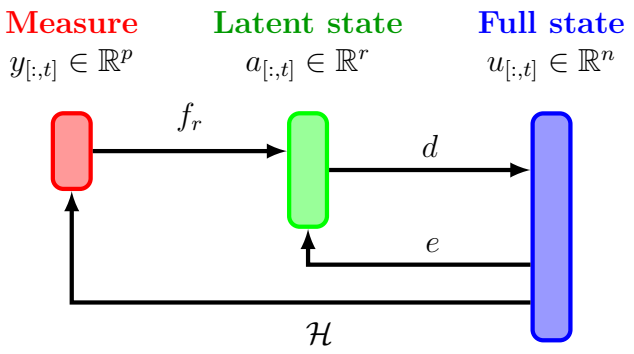


Figure 2: Representation of the reconstruction problem with dimensionality reduction: measurements are used to recover dynamics of dominant structures

The most common reduction technique is the proper orthogonal decomposition (POD) which is the name given to principal components analysis (PCA) applied to fluid flow fields. POD modes are uncorrelated directions that optimally describe variability in data, massively used for reconstructive [28] and predictive tasks [29]. Deep learning [30] has made possible considerable progress in dimensionality reduction by using autoencoders. It consists of two neural networks trained together to minimize a reconstruction error: an encoder encodes high dimensional data to low dimensional data (called latent data living in latent space) and a decoder decodes latent data to recover the high dimensional state. Xu *et al.* [20] took advantage of convolutional autoencoders to leverage nested nonlinear manifolds and predict the transient flow over a cylinder and in the wake of a ship. Morton *et al* [31] used variational autoencoders to learn a reduced-order model (ROM) for the flow behind two counter rotating cylinders. Using the power of generative modeling, the authors were able to generate relevant flow data for non-simulated cases.

Given a dimensionality reduction method, it is possible to learn a reduced-order model for latent variables. Some examples from the literature include the approximation of Koopman operator eigenfunctions with dynamical mode decomposition (DMD [32]) or neural networks [33] and the approximation of Perron Frobenius operator using clustering algorithm (CROM [34]).

1.3. Sensors placement

The position of sensors plays a crucial role in the reconstruction performance. Supposing that measurements are known discrete positions of the field to reconstruct, the measurement operator $\mathcal{H}$ is a matrix. For regressive methods, sensors must measure non-redundant signals, otherwise multicollinearities in explaining variables will affect the estimation [35]. For direct reconstruction methods, the measurement matrix

C must be incoherent with the reference library ϕ. Most algorithms therefore search the product $C\phi$ with the lowest condition number to easily solve the minimization problem [36]. Main approaches include data-driven sparse placement such as QR sensing so that $C\phi$ satisfies the restricted isometry principle [4].

1.4. Objectives

This paper compares regressive and direct reconstruction methods for the estimation of increasing complexity flow fields: wake of a 2D circular cylinder ($Re = 200$, 2D domain, case 1), spatial mixing layer (Reynolds based on initial vorticity thickness $Re = 500$, 2D domain, case 2) and wake of a square cylinder ($Re = 20000$, 3D domain, case 3). The estimation is performed on the two or three components of the fluctuating velocity field, which was reduced using linear (POD, linear autoencoder LAE or linear variational autoencoder LVAE) or nonlinear (variational autoencoder VAE) reduction techniques. Measurements are velocity signals at known positions, with sensor placement being performed by an unsupervised algorithm. There is a focus on reduction and reconstruction errors for all methods. All models are cross-validated using grid search, randomized grid search or genetic optimization. The sensitivity to noisy measurements is also investigated. The main interests of this work lie in the complete machine learning strategy used, the investigation of variational autoencoders to reduce fluid flow data and an analysis of the regressive reconstructions robustness.

2. Methodology

The proposed strategy is purely data-driven: simulation data are used to define the grid of sensors, extract low-dimensional patterns, create the library of reference elements for direct reconstruction and learn the regressive function for regressive reconstruction. The global framework for fluid flow estimation from measurements is summarized in figure 3. This section provides mathematical details for each block.

2.1. Notations

Simulation data are written as a matrix $U \in \mathbb{R}^{n\times m}$ where n is the dimension (number of cells n_c multiplied by number of velocity components n_v) and m is the number of snapshots. Considering a 70/30 split, $m_{\text{train}} = 0.7m$ and $m_{\text{test}} = 0.3m$ are respectively the number of training and testing snapshots. The fluctuating velocity field matrix is defined as $u = U - \bar{U}$ where $\bar{U}$ is the mean flow computed over all snapshots. This tall but skinny matrix ($n \gg m$) is split into zero mean matrices u^{train} and u^{test}. Considering that measurements are p known locations in the fluctuating field to estimate, the measurement matrix is $C \in \mathbb{R}^{p\times n}$ with $C_{i,j} = 1$ if $y_i = u_j$ and 0 otherwise. Training and testing measurements are therefore $y^{\text{train}} = Cu^{\text{train}}$ and $y^{\text{test}} = Cu^{\text{test}}$. The high dimensional field u is reduced to the low dimensional field $a \in \mathbb{R}^{r\times m}$ where $n \gg r$ and r is the number of modes. Given a measurement vector at time t (denoted $y_{[:,t]}$ and not necessary in the testing set), the estimated reduced state is written $\widehat{a}_{[:,t]}$ and the estimated fluctuant velocity field is $\widehat{u}_{[:,t]}$.

2.2. Metrics

The determination coefficient R^2 is used to compare all training or testing reduced snapshots with their estimation. It is defined by the ratio between the variance recovered in estimated snapshots and the true variance. For the $i_r \in [1..r]$ component of the reduced state and all $m_e = m_{\text{train}}$ or $m_e = m_{\text{test}}$ estimations, the coefficient reads:

$$R^2_{i_r} = 1 - \frac{\sum_{t=1}^{m_e} \left[a_{[i_r,t]} - \widehat{a}_{[i_r,t]}\right]^2}{\sum_{t=1}^{m_e} \left[a_{[i_r,t]} - \overline{a}_{[i_r]}\right]^2}$$

The reduced state being computed from a fluctuating field, $\overline{a}$ is the null vector. The normalized mean square error is used to compare all training or testing high dimensional flow fields with their estimation. It is a normalized version of the mean square error between true snapshots and reconstructed ones. For the $i_n \in [1..n]$ component of the field and m_e estimations, it is computed as:

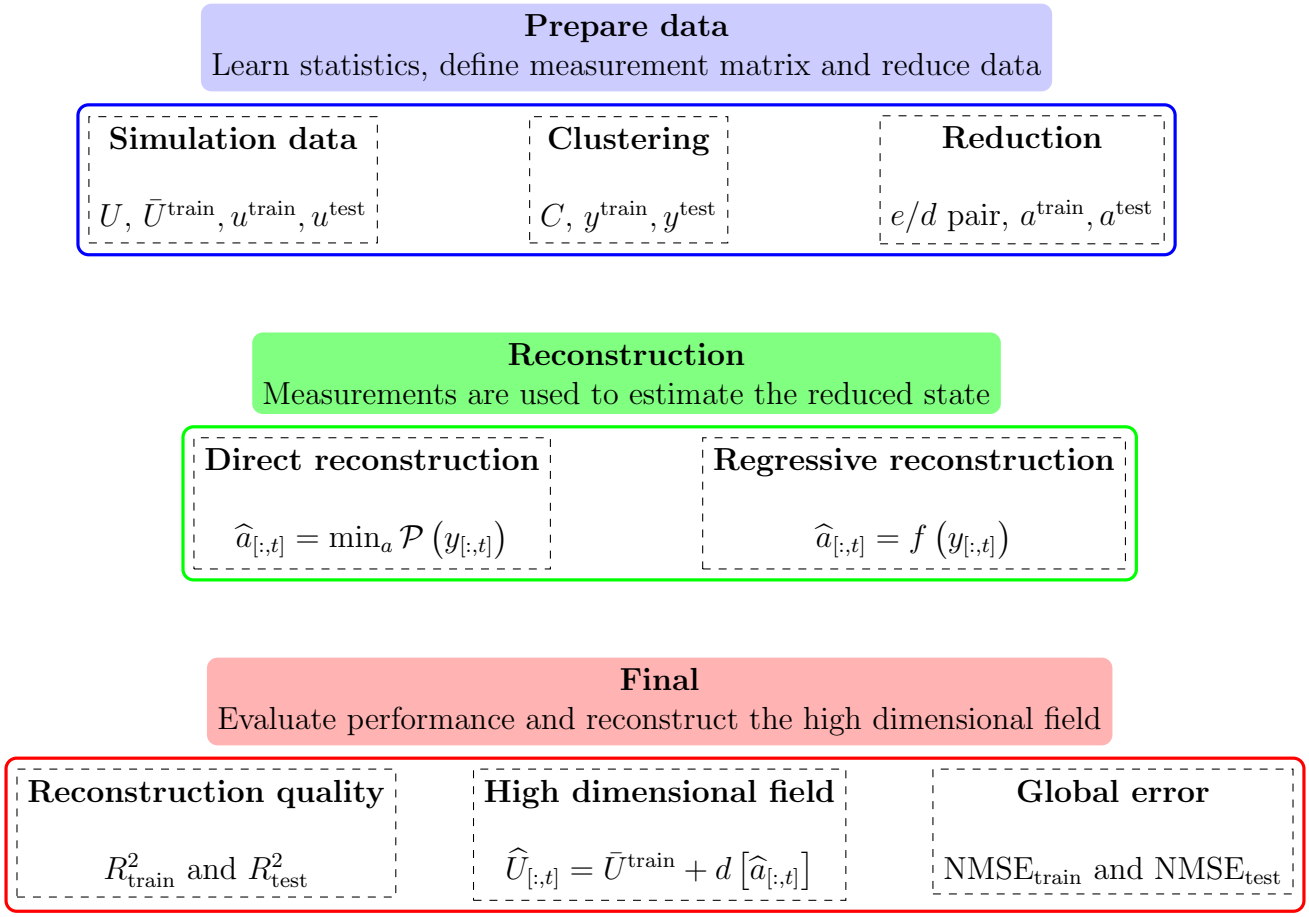


Figure 3: Global framework for the reconstruction of a fluid flow field using measurements.

$$\text{NMSE}_{i_n} = \frac{\sum_{t=1}^{m_e} \left[U_{[i_n,t]} - \widehat{U}_{[i_n,t]}\right]^2}{\sum_{t=1}^{m_e} \left[U_{[i_n,t]} - \overline{U}_{[i_n]}\right]^2}$$

By averaging over the number of modes or the number of cells, it is possible to appreciate the quality of the reconstruction via a single number. Therefore, the following metrics are also computed:

$$R^2 = \frac{1}{r} \sum_{i_r=1}^{r} R^2_{i_r}$$

$$\text{NMSE} = \frac{1}{n} \sum_{i_n=1}^{n} \text{NMSE}_{i_n}$$

These metrics are complementary: the determination coefficient scores the $y \to \hat{a}$ step while NMSE quantifies the $y \to \hat{u}$ error. For R^2 close to unity, estimated reduced states recover nearly all the variance observed in expected reduced states. For NMSE close to zero, all estimated fields are on average close to real ones. Results should be close for both training and testing estimations, otherwise reconstruction or reduction methods were badly designed (overfitting or underfitting).

2.3. Enhanced clustering

The spatial location of sensors is determined by enhanced clustering [37]. To be consistent with the definition of the state $u_{[:,t]} \in \mathbb{R}^{n_v \times n_c}$, the simulation mesh with n_c centres is repeated n_v times. This gives the set $\{x_{[i_n,:]} \in \mathbb{R}^{n_v}\}_{i_n \in [1..n]}$ to be clustered into K_c voronoi cells $\{V_k\}_{k \in [1..K_c]}$ defined by their centroids $\{\mu_{[k,:]} \in \mathbb{R}^{n_v}\}_{k \in [1..K_c]}$. The partitions are solutions of the following optimization problem:

$$\underset{V_1,\dots,V_{K_c}}{\arg\min} \sum_{k=1}^{K_c} \sum_{x_{[i_n,:]} \in V_k} ||x_{[i_n,:]} - \mu_{[k,:]}||_2^2$$

where $||.||_2^2$ is the square of the l_2 norm and represents the sum of squared components. Most energetic clusters (in terms of velocity variance) are then defined as sensors. The complete procedure is presented in algorithm 1. In this paper, the number of clusters is set to $K_c = 500$ and sensors correspond to Voronoi centroids recovering 20%, 40% or 80% of the total training field variance.

2.4. Dimensionality reduction

An encoder/decoder procedure for dimensionality reduction is considered. The encoder e compresses the data from initial space (dimension n) to a latent space (dimension r) and the decoder d decompresses encoded data. The objective is to find the e/d pair that maximizes the information when encoding and minimizes the reconstruction error ϵ when decoding. The latent space must also be smaller than the physical space ($r \ll n$) to avoid the trivial but irrelevant solution $e = d = I_{n\times n}$. Given a family of candidate encoders E and a family of candidate decoders D, the best e/d pair is:

$$(e^*, d^*) = \underset{(e,d)\in E\times D}{\arg\min}\ \epsilon(u_{[:,t]}, d\left[e(u_{[:,t]})\right])$$

In proper orthogonal decomposition (POD), the encoder and decoder are unitary matrices obtained from the spectral decomposition of the training covariance matrix $C_u = u^{\text{train}}[u^{\text{train}}]^T$. This decomposition yields $C_u\Phi = \Phi\Lambda$ where the transfer matrix $\Phi \in \mathbb{R}^{n\times n}$ transforms initial basis vectors into uncorrelated directions. These modes are hierarchically sorted according to the variance Λ_{ii} they recover. When truncating the transformation to first r modes, initial data are written in the best r dimensional subspace to describe variability in u^{train}.

Algorithm 1 Enhanced clustering

Require: K_c number of clusters **and** $\{x_{[i_n,:]} \in \mathbb{R}^{n_v}\}_{i_n\in[1..n]}$ mesh points **and** u^{train} training velocity field

1: define K_c initial centroids:

$$\{\mu^{(0)}_{[k,:]} \in \mathbb{R}^{n_v}\}_{k\in[1..K_c]}$$

2: **while** convergence is not reached **do**

3: find Voronoi cells

$$V_k^{(l)} = \left\{ \begin{array}{c} x_{[i_n,:]} \in \mathbb{R}^{n_v} : \left|\left|x_{[i_n,:]} - \mu^{(l)}_{[k,:]}\right|\right|_2^2 \\ \leq \left|\left|x_{[i_n,:]} - \mu^{(l)}_{[k^*,:]}\right|\right|_2^2 \\ \forall k^* \in [1..K_c] \end{array} \right\}$$

4: compute new centroids

$$\mu^{(l+1)}_{[k,:]} = \frac{1}{|V_k^{(l)}|} \sum_{\substack{i_n \text{ with} \\ x_{[i_n,:]}\in V_k}} x_{[i_n,:]}$$

where $|V_k^{(l)}|$ is the cardinality of $V_k^{(l)}$.

5: **end while**

6: compute the variance of training velocity fields in each cluster

$$\sigma^2_{V_k} = \frac{1}{m_{\text{train}}} \sum_{t=1}^{m_{\text{train}}} \sum_{\substack{i_n \text{ with} \\ x_{[i_n,:]}\in V_k}} \left[u^{\text{train}}_{[i_n,t]}\right]^2$$

7: keep p most energetic clusters

In recent applications, encoders and decoders are neural networks. They consist in interconnected units (called neurons) propagating the input information via weighted connections. With a good choice for hyperparameters (number of layers, activation functions, regularization) and optimal parameters (weights between neurons and biais for each neuron), a neural network can approximate any function [38]. When considering neurons with linear activations, encoding and decoding transformations are matrices but no orhogonality constraint is imposed. Optimal weights and biases minimize the mean square

error between training samples and autoencoded ones. For one training sample, the cost function reads:

$$\epsilon = ||u^{\text{train}}_{[:,t]} - \hat{u}^{\text{train}}_{[:,t]}||_2^2 = \sum_{i_n=1}^{n} \left[u^{\text{train}}_{[i_n,t]} - \hat{u}^{\text{train}}_{[i_n,t]}\right]^2$$

In the variational formulation of autoencoders, inputs $u_{[:,t]}$ are encoded as multivariate Gaussian distributions $\mathcal{N}(a_{\mu_{[:,t]}}, a_{\sigma[:,t]})$ instead of single points. The encoded distributions are encouraged to be close to standard Gaussian $\mathcal{N}(0, I_{r\times r})$, thus enforcing the regularity (in terms of continuity and completeness) in the latent space and uncorrelatedness between latent features. The cost function therefore encompasses two errors to minimize: the reconstruction error and the Kullback Leibler divergence between the encoded input distribution and a standard Gaussian. This error reads [39]:

$$\begin{aligned}\epsilon &= ||u^{\text{train}}_{[:,t]} - \hat{u}^{\text{train}}_{[:,t]}||_2^2 \\ &\quad + D_{\text{KL}}\left[\mathcal{N}(a_{\mu}^{\text{train}}_{[:,t]}, a_{\sigma}^{\text{train}}_{[:,t]}) \;||\; \mathcal{N}(0, I_{r\times r})\right] \\ &= \sum_{i_n=1}^{n} \left[u^{\text{train}}_{[i_n,t]} - \hat{u}^{\text{train}}_{[i_n,t]}\right]^2 \\ &\quad - \frac{1}{2}\sum_{i_r=1}^{r}\left[1 + \log\left(a^2_{\sigma[i_r,t]}\right) - a^2_{\sigma[i_r,t]} - a^2_{\mu[i_r,t]}\right]\end{aligned}$$

In this paper, POD, linear autoencoder (LAE), linear variational autoencoder (LVAE) and variational autoencoder (VAE) are implemented for dimensionality reduction. The latent space dimension is chosen according to an energetic criterion: the number of neurons in the latent layer is the number of POD modes required to recover 40%, 80% and 95% (for mixing layer and vortex shedding $Re = 20000$) or 99% (for vortex shedding $Re = 200$) of the variability observed in training data. Chosen architectures for autoencoders are given in figures 4 and 5. Mathematical formulations of the encoding/decoding process are detailed in table 2.

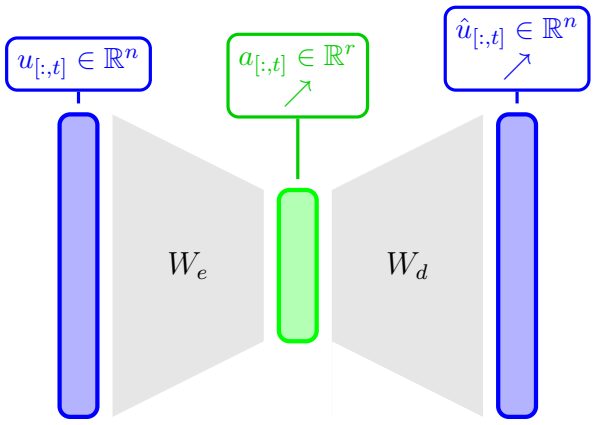


Figure 4: The linear autoencoder (LAE) is composed of one hidden layer and linear (↗) activation functions

2.5. Direct reconstruction

Given a measurement matrix C, the field is measured by $y_{[:,t]} = Cu_{[:,t]}$. Using the known (lossy) decoder d, one can approximate the measurement via $\hat{y}_{[:,t]} = Cd\left[a_{[:,t]}\right]$. The objective is to find the reduced state $a_{[:,t]}$ that minimizes the error between $y_{[:,t]}$ and $\hat{y}_{[:,t]}$. This optimization problem is denoted $\mathcal{P}(y_{[:,t]})$. For linear decoders such as POD, LAE or LVAE, the direct reconstruction (DR) can be performed using linear algebra tools. For nonlinear decoders such as VAE, nonlinear optimization is required, significantly increasing the complexity. The present paper therefore limits investigations to linear decoders. Following this restriction, general forms for the decoding and the recovered measurement are adopted:

$$\begin{cases} d\left[a_{[:,t]}\right] = Wa_{[:,t]} + b \\ \hat{y}_{[:,t]} = CWa_{[:,t]} + Cb \end{cases}$$

Given the number of sensors (p equations) and the number of modes (r unknowns), the optimization strategy is different. If the problem is underdetermined ($r \geq p$), there exists an infinity of solutions and the one with minimal l_1 norm is selected. The problem reads:

$$\widehat{a}_{[:,t]} = \mathcal{P}(y_{[:,t]}) = \begin{cases} \min\limits_{a} ||a||_1 \\ y_{[:,t]} = CWa + Cb \end{cases}$$

	POD	LAE	LVAE	VAE
Find	$\Phi_r \in \mathbb{R}^{n\times r}$	$W_e \in \mathbb{R}^{n\times r}$ and $b_e \in \mathbb{R}^r$ $W_d \in \mathbb{R}^{r\times n}$ and $b_d \in \mathbb{R}^n$	$W_1 \in \mathbb{R}^{n\times r}$ and $b_1 \in \mathbb{R}^r$ $W_2 \in \mathbb{R}^{r\times r}$ and $b_2 \in \mathbb{R}^r$ $W_3 \in \mathbb{R}^{r\times r}$ and $b_3 \in \mathbb{R}^r$ $W_4 \in \mathbb{R}^{r\times n}$ and $b_4 \in \mathbb{R}^n$	Same as LVAE
Encoding	$a_{[:,t]} = \Phi_r^T u_{[:,t]}$	$a_{[:,t]} = W_e u_{[:,t]} + b_e$	$\begin{cases} h_{[:,t]} = W_1 u_{[:,t]} + b_1 \\ a_{\mu[:,t]} = W_2 h_{[:,t]} + b_2 \\ a_{\sigma[:,t]} = \sigma\left[W_3 h_{[:,t]} + b_3\right] \end{cases}$	$\begin{cases} h_{[:,t]} = \tanh\left[W_1 u_{[:,t]} + b_1\right] \\ a_{\mu[:,t]} = W_2 h_{[:,t]} + b_2 \\ a_{\sigma[:,t]} = \sigma\left[W_3 h_{[:,t]} + b_3\right] \end{cases}$
Decoding	$\hat{u}_{[:,t]} = \Phi_r a_{[:,t]}$	$\hat{u}_{[:,t]} = W_d a_{[:,t]} + b_d$	$\begin{cases} a_{[:,t]} \sim \mathcal{N}\left(a_{\mu[:,t]}, a_{\sigma[:,t]}\right) \\ \hat{u}_{[:,t]} = W_4 a_{[:,t]} + b_4 \end{cases}$	$\begin{cases} a_{[:,t]} \sim \mathcal{N}\left(a_{\mu[:,t]}, a_{\sigma[:,t]}\right) \\ \hat{u}_{[:,t]} = \tanh\left[W_4 a_{[:,t]} + b_4\right] \end{cases}$
Noteworthy	Orthogonal modes $\Phi_r^T \Phi_r = I_{r\times r}$	Non orthogonal modes $W_e = W_d^\dagger$	-	Nonlinear reduction
Modes	Φ_r	W_2	W_4	Extracted via SVD on W_4

Table 2: Encoding and decoding formula for considered reduction methods. Parameters W are weights while parameters b are biases.

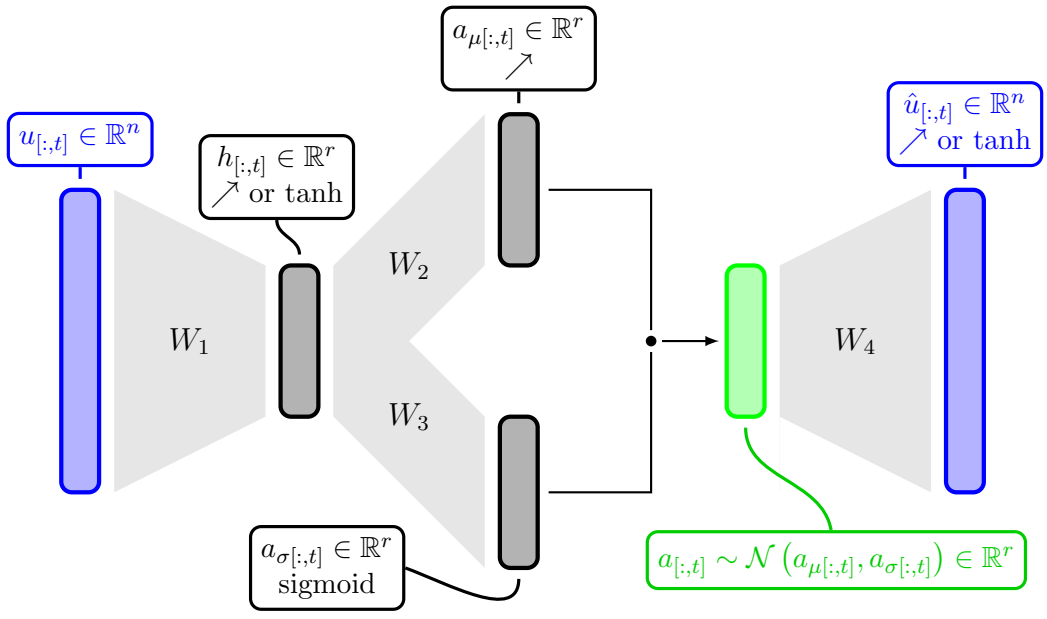


Figure 5: The variational autoencoder encodes the input data as a Gaussian distribution. Before computing the mean and the standard deviation of the input, a first hidden layer linearly (LVAE, ↗) or nonlinearly (VAE, tanh) compresses the data. In the training phase, a sample drawn from the encoded distribution is decoded via a linear (LVAE) or nonlinear (VAE) decoder. When the autoencoder is deployed for a reconstruction task, only the decoder is of interest.

If the problem is overdetermined ($r \leq p$), no solutions exist. A least squares approach is used to find the closest solution in the subspace spanned by colums of the reference library CW. A regularization term $\lambda||a||_1$ may be added to promote robustness to outliers in data. Several λ

hyperparameters are tested when reconstructing training snapshots and the one leading to the maximum $R^{2^{\text{train}}}$ score is kept in the testing phase. The overdetermined problem therefore reads:

$$\widehat{a}_{[:,t]} = \min_{a} ||y_{[:,t]} - [CWa + Cb]||_2^2 + \lambda||a||_1$$

All optimization problems for direct reconstruction are solved using the cvxpy library [40].

2.6. Regressive reconstruction

In regressive reconstruction methods, the objective is to learn the mapping between the measurement space and the latent space. Suppose the true relationship is $a_{[:,t]} = f\left[y_{[:,t]}\right] + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ an irreductible error. In the context of machine learning, f is approximated by $\hat{f}$ using training data points $\mathcal{D}$. The model $\hat{f}$ may combines measurements (parametrized method) or make a decision (non parametrized) and perform multi-task (simultaneous reconstruction of all modes) or not (modes reconstructed independantly). To ensure *good* performance on training data points but also on unseen (testing) data points, the model must meet a bias-variance tradeoff. These terms appear when decomposing the quadratic error on a testing point with an explicit dependance on $\mathcal{D}$:

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[\left[a_{[:,t]} - \hat{f}\left[y_{[:,t]}; \mathcal{D}\right]\right]^2\right] = &\text{Bias}\left[\hat{f}\left[y_{[:,t]}\right]\right]^2 \\ &+ \text{Var}\left[\hat{f}\left[y_{[:,t]}\right]\right] \\ &+ \sigma^2\end{aligned}$$

The bias of the learning method indicates if the model is a good candidate for approximating the true function. The more flexible the candidate, the lower the bias. Its expression reads:

$$\text{Bias}\left[\hat{f}\left[y_{[:,t]}\right]\right] = \mathbb{E}_{\mathcal{D}}\left[\hat{f}\left[y_{[:,t]}; \mathcal{D}\right] - f\left[y_{[:,t]}; \mathcal{D}\right]\right]$$

The variance measures the difference in fits when changing the data set. Its expression reads:

$$\begin{aligned}\text{Var}\left[\hat{f}\left[y_{[:,t]}\right]\right] = \\ \mathbb{E}_{\mathcal{D}}\left[\left\{\hat{f}\left[y_{[:,t]}; \mathcal{D}\right] - \mathbb{E}_{\mathcal{D}}\left[\hat{f}\left[y_{[:,t]}; \mathcal{D}\right]\right]\right\}^2\right]\end{aligned}$$

The bias-variance tradeoff is presented in figure 6. This decomposition has mainly theoritical uses because one can rarely afford to have different training sets. In the following, four regressive models are presented: linear regression, support vector regression, artificial neural network and gradient boosted decision trees. Cross-validation strategies are also discussed. Note that a visual comparison of all investigated regressive methods is given in figure 7.

Linear regression (LM). The model assumes a linear relationship between the measurement and all components of the reduced state, yielding a reconstruction $\widehat{a}_{[:,t]} = \beta^T y_{[:,t]}$. The parameter to optimize is the $\beta \in \mathbb{R}^{p \times r}$ matrix. The cost function is the mean square error which is Lasso penalized using a l_{21} norm [41].

$$\mathcal{C}(\beta) = \frac{1}{2m_{\text{train}}}||R(\beta)||_F^2 + \alpha||\beta||_{21}$$

In doing so, most relevant measurements are selected thus promoting sparsity in the solution vector β. Here, the residual R, the Frobenius and the l_{21} norms are defined as follows:

$$\begin{cases} R(\beta) = \left\{a^{\text{train}}\right\}^T - \left\{y^{\text{train}}\right\}^T \beta \\ ||R(\beta)||_F = \sqrt{\sum_{t,i_r}\left[R(\beta)_{[t,i_r]}\right]^2} \\ ||\beta||_{12} = \sum_{i_p=1}^{p}\sqrt{\sum_{i_r=1}^{r}\left[\beta_{[i_p,i_r]}\right]^2} \end{cases}$$

In the single-task formulation (LS), the

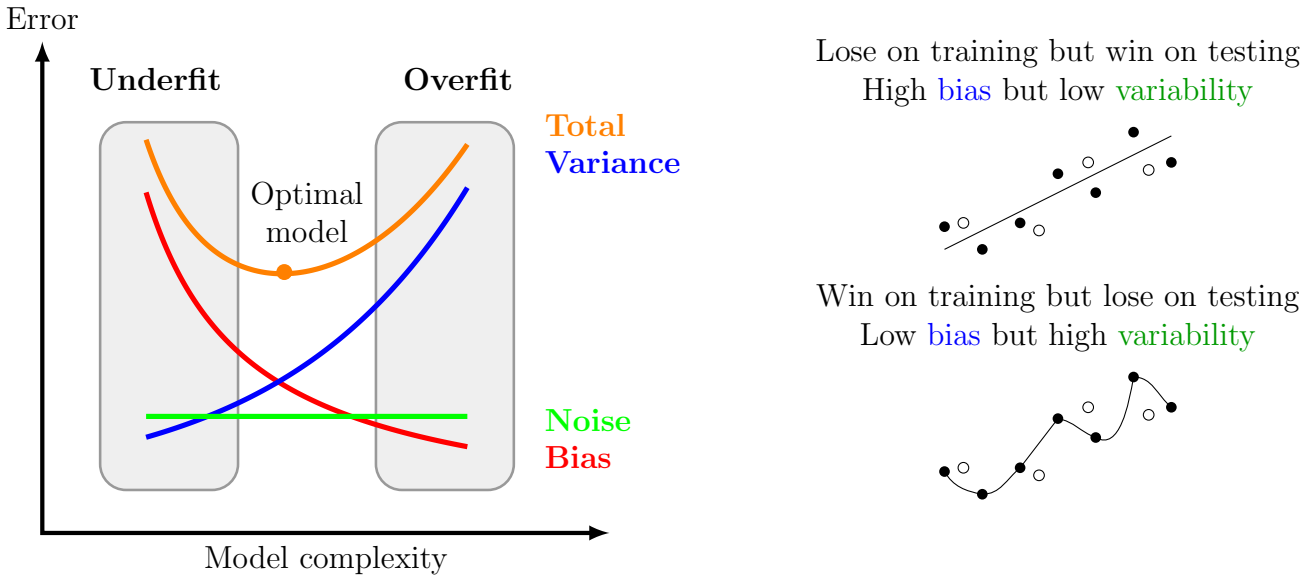


Figure 6: Illustration of the bias variance tradeoff. Filled circles represent training data while empty circles represent testing data.

reconstruction yields $\widehat{a}_{[i_r,t]} = \beta^T_{[i_r,:]} y_{[:,t]}$ with $\beta_{[i_r,:]} \in \mathbb{R}^{p\times 1}$ for each mode i_r. Selected measurements are different for each target and cross-validation must be performed r times. Without the regularization term, the method is known as *linear stochastic estimation* [6].

Support vector regression (SVR). In its linear formulation, each component of the reduced state is linearly explained by the measurements. For the mode i_r, the regression yields $\hat{a}_{[:,t]} = \beta^T_{i_r} y_{[:,t]}$ with $\beta_{i_r} \in \mathbb{R}^{p\times 1}$ ensuring at most an ϵ deviation from true targets. Optimal parameters are found by solving the primal formula:

$$\begin{cases} \min\limits_{\beta_{i_r},\xi,\xi^*} \dfrac{1}{2}\beta^T_{i_r}\beta + C_B \sum\limits_{t=1}^{m_{\text{train}}} (\xi_t + \xi^*_t) \\ a^{\text{train}}_{[i_r,t]} - \beta^T_{i_r} y^{\text{train}}_{[:,t]} \leq \epsilon + \xi_t \ \forall t \\ \beta^T_{i_r} y^{\text{train}}_{[:,t]} - a^{\text{train}}_{[i_r,t]} \leq \epsilon + \xi^*_t \ \forall t \\ \xi_t, \xi^*_t \geq 0 \ \forall t \end{cases}$$

where slack variables ξ and ξ^* penalyze observations out of the ϵ tube. This regularization is controlled by the box constraint C_B. Introducing α and α^* as Lagrange multipliers, the dual formula reads:

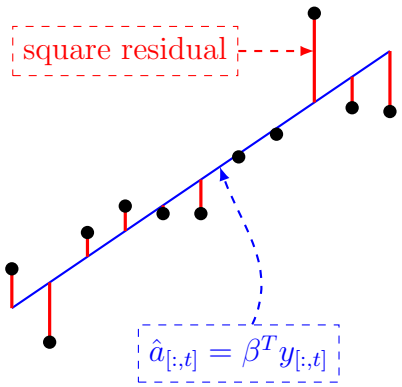
$$\begin{cases} \min\limits_{\alpha} \mathcal{C}(\alpha) \\ \sum\limits_{t=1}^{m_{\text{train}}} (\alpha_t - \alpha^*_t) = 0 \text{ with } 0 \leq (\alpha_t, \alpha^*_t) \leq C_B \end{cases}$$

Denoting e the vector of all ones and $Q = \left\{y^{\text{train}}\right\}^T y^{\text{train}} \in \mathbb{R}^{p\times p}$ the semidefinite matrix computing relationships between measurements in the measurement space, the cost function reads:

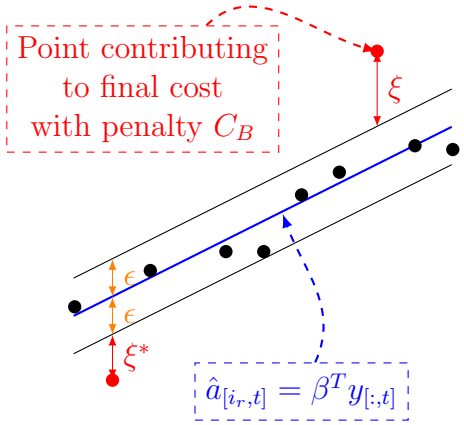
$$\begin{aligned} \mathcal{C}(\alpha) = \ & \frac{1}{2}(\alpha - \alpha^*)^T Q (\alpha - \alpha^*) \\ & + \epsilon e^T(\alpha + \alpha^*) - y^{\text{train}}(\alpha - \alpha^*) \end{aligned}$$

According to Karush-Kuhn-Tucker conditions [42], observations inside the ϵ tube have a zero multiplier. Other observations are refered as support vectors. These conditions are written:

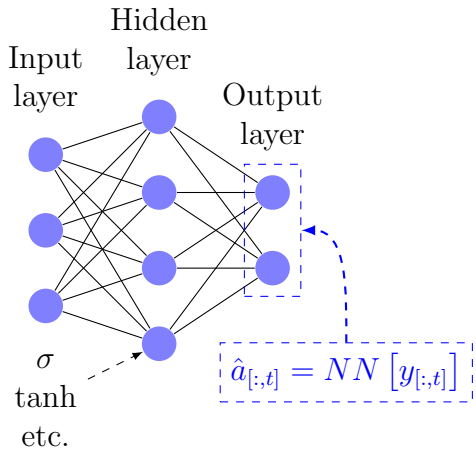
$$\forall t \begin{cases} \alpha_t \left(\epsilon + \xi_t - a^{\text{train}}_{[i-r,t]} + \hat{a}^{\text{train}}_{[i-r,t]}\right) = 0 \\ \alpha^*_t \left(\epsilon + \xi^*_t + a^{\text{train}}_{[i-r,t]} - \hat{a}^{\text{train}}_{[i-r,t]}\right) = 0 \\ \xi_t (C_B - \alpha_t) = 0 \\ \xi^*_t (C_B - \alpha^*_t) = 0 \end{cases}$$



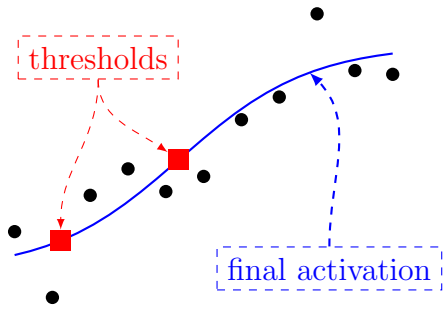
(a) Linear regression: minimize the total sum of square residuals



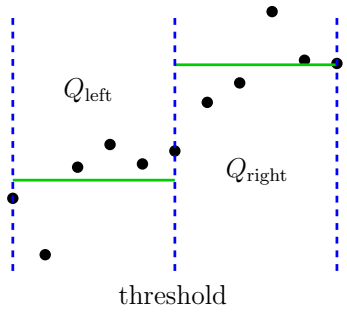
(b) SVR: ensure at most ϵ deviation



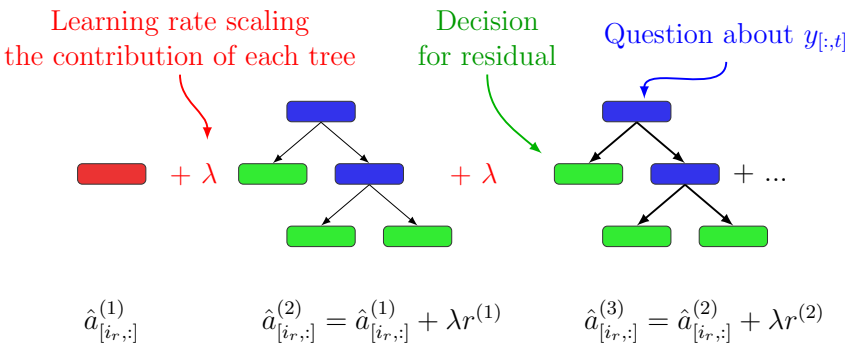
(c) Neural network: find weights and biases minimizing the mean square error



(d) Neural network with one hidden layer composed of two σ neurons: output with two s-shapes



(e) Decision tree with max depth 2



(f) Gradient boosting. A decision tree predicts residuals to update a first guess

Figure 7: Illustrations of the supervised learning functions investigated in this paper. Black dots represent training data.

Linear ϵ-SVR can also be used to perform a nonlinear regression in the measurement space. To do so, training measurements are implicitely mapped to a higher dimensional space where a linear ϵ-SVR is relevant. A kernel function G is used to compute higher dimensional interactions without actually transforming variables. The estimation for mode i_r reads:

$$\widehat{a}_{[i_r,t]} = \sum_{t=1}^{m_{\text{train}}} (\alpha_t^* - \alpha_t) G(y_{[:,t]}^{\text{train}}, y_{[:,t]})$$

The Q matrix in the cost function is therefore modified to $Q_{[t_1,t_2]} = G\left(y_{[:,t_1]}^{\text{train}}, y_{[:,t_2]}^{\text{train}}\right)$ where the G function may be polynomial or Gaussian. The reader can find the complete derivation of equations in the tutorial of Smola *et. al* [43].

Neural networks (NN). The reduced state is explained by the measurement vector passing through units organized in layers. The propagation operator is built on activation functions (sigmoid $\sigma(x) = \dfrac{1}{1+e^{-x}}$, hyperbolic tangent tanh, rectified linear unit relu$(x) = \max(0, x)$, etc.), weights and biases. The interest in neural networks is largely attributed to their flexibility [44]. Any function can indeed be approximated by a sufficiently large and deep network: a lot of neurons in hidden layers increase nonlinearities while a lot of hidden layers create a hierarchical representation of features. The neural network estimate being $\widehat{a}_{[:,t]} = \text{NN}\left[y_{[:,t]}\right]$, the mean square error to minimize is:

$$\mathcal{C}(W) = \frac{1}{2m_{\text{train}}} || \left\{a^{\text{train}}\right\}^T - NN\left[\left\{y^{\text{train}}\right\}^T\right] ||_F^2$$

where W refers to all weights and biases in the network. To avoid overfitting the training data, neurons are randomly dropped out at each parameter update [45]. This improves generalization by enforcing the network to learn how to use all of its inputs. To speed up the learning process, batch gradient descent is performed: the global gradient is the average gradient evaluated on batches of data. Denoting B the number of batches and $\mathcal{C}_b$ the cost for batch b, a gradient update then reads:

$$W \leftarrow W - \frac{\eta}{B} \sum_{b=1}^{B} \left(\frac{\partial \mathcal{C}_b}{\partial W}\right)$$

Backpropagation is used to compute gradients. To be concise, a training sample is denoted (y_t, a_t). When reaching layer l in the network, the input i^l is treated by neurons according to $o^l = n_l(i^l)$ where n_l is the activation function. At the final layer L, the cost for one sample is evaluated via $\mathcal{C}_t = \frac{1}{2}||a_t - o_t^L||^2$. The contribution of neuron j in layer l to this error is $[\delta_j^l]_t$. Finally, weights between layer $l-1$ and l are w^l while neuron biases for layer l are b^l. The complete backpropagation [46] procedure for gradient computation is in algorithm 2. The interested reader can refer to [47] for more details.

Algorithm 2 Backpropagation

Require: Prior parameters and a sample (y_t, a_t)

1: Set corresponding activation of first layer

$$o^1 = y_t$$

2: Forward propagation - $\forall l$ compute

$$o^l = n_l(i^l) = n_l(w^l o^{l-1} + b^l)$$

3: Errors in the last layer - compute

$$\delta^L = \nabla_a C_t \odot n_l'\left(i^L\right) = (a_t - o^L) \odot n_l'\left(i^L\right)$$

4: Backpropagation of errors - $\forall l$ compute

$$[\delta^l]_t = \left([w^{l+1}]^T \delta^{l+1}\right) \odot n_l'\left(i^l\right)$$

where n_l' is the derivative of the activation function

5: Gradient estimation - compute

$$\frac{\partial C_t}{\partial w_{jk}^l} = o_k^{l-1}[\delta_j^l]_t \text{ and } \frac{\partial C_t}{\partial b_j^l} = [\delta_j^l]_t$$

Gradient Boosting (GB). For each mode

i_r, decision trees are combined to decide the value of $\widehat{a}_{[i_r,t]}$ from simple decision rules applied to $y_{[:,t]}$. At the opposite of adaptative boosting where the decision is made from very short trees (called stumps) with different amount of say (see [48] for details), gradient boost builds bigger and fixed-sized trees scaled by the same parameter. A visual explanation is given in figure 7.

Decision trees are non parametric supervised learning methods where the measurement space is recursively split by minimizing an inpurity function (see algorithm 3 and [49]). Gradient boosted trees combine such trees while considering previous trees' errors in the learning process [50]. The detailed process is presented in algorithm 4. Here, the differentiable loss function is the mean square error:

$$\mathcal{L}\left[a_{[i_r,t]}, F(y_{[:,t]})\right] = \frac{1}{2}\left(a_{[i_r,t]} - F\left[y_{[:,t]}\right]\right)^2$$

With this loss function, F_0 is the mean value of $a^{\text{train}}_{[i_r,:]}$, corresponding to an initial guess for any measurement $y_{[:,t]}$. The residual for each training sample (when building the m-th tree) reads $r_{tm} = F\left[y^{\text{train}}_{[:,t]}\right] - a^{\text{train}}_{[i_r,t]}$. Terminal regions R_{jm} are created, corresponding to leaves in the tree. Leaves may contain several residuals so the mean of residuals γ_{jm} is computed. The final estimation combines the M trees scaled by same amount, yielding $\widehat{a}_{[i_r,t]} = F_M\left[y_{[:,t]}\right]$. The hyperparameter λ is called the learning rate and empirical evidence shows that the smaller the λ (i.e. a lot of small steps in the right direction), the lower the variance of the model. The learning rate is therefore a regularization parameter.

Cross-validation. For each reconstruction methods, hyperparameters must be specified. Cross-validation (CV) consists in learning as many models as hyperparameter candidates and select the one leading to the best generalization score. The procedure is presented in algorithm 5. In this paper, 3 fold CV is performed, with different strategies for choosing candidates.

Algorithm 3 Decision tree

Require: maximum depth d_{max}

1: At node k, represent N_k available data by $\mathcal{Q}$

2: Select parameters $\theta = (\text{feature}, \text{threshold})$ minimizing the **impurity function** $\mathcal{I}$ defined by:

$$\mathcal{I}(\mathcal{Q},\theta) = \frac{k_{\text{left}}}{N_k}\mathcal{X}\left(\mathcal{Q}_{\text{left}}(\theta)\right) + \frac{k_{\text{right}}}{N_k}\mathcal{X}\left(\mathcal{Q}_{\text{right}}(\theta)\right)$$

where k_{left} is the number of samples placed in left node, $\mathcal{X}$ is the mean square error function and $\mathcal{Q}_{\text{left}(\theta)}$ are training data in the left node.

3: Best parameters are $\theta^* = \arg\min_\theta \mathcal{I}(\mathcal{Q},\theta)$

4: Recurse for $\mathcal{Q}_{\text{left}(\theta^*)}$ and $\mathcal{Q}_{\text{right}(\theta^*)}$ until d_{max} is reached

For Lasso linear regression, a simple grid search is carried out. The only hyperparameter to optimize is α and candidates are taken in a discretized $[0, 1]$ interval.

For SVR and gradient boosting, a randomized grid search is performed [51]. A random search with $N = 25$ trials will find the 5% optimal region of hyperparameters with a probability greater than 0.7, as explained in figure 8. SVR hyperparameters are the box constraint (C_B, sampled from an exponential distribution with scale 15) and the kernel (polynomial with degree $q = 2$ to $q = 10$ or Gaussian with γ drawn from an exponential distribution with scale 0.01). Gradient boosting hyperparameters are the learning rate (λ, sampled from uniform distribution in $[0, 1]$), the number of estimators (M, random integer between 10 and 50), the minimum samples in a leaf (random integer between 5 and 15) and the maximum depth (d_{max}, integer between $0.1p$ and $0.8p$).

For neural network, a genetic optimization based on a selection/breeding/mutation scheme is implemented. Hyperparameters leading to best generalization scores survive and evolve. In this paper, a population of 15 networks evolves

Algorithm 4 Gradient boosting

Require: data $(y^{\text{train}}, a^{\text{train}}_{[i_r,:]})$, differentiable loss function $\mathcal{L}$

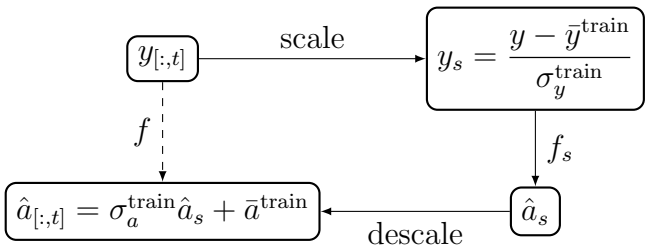
1: Initialize reconstruction function $F_0(y) = \arg\min_\gamma \sum_{t=1}^{m_{\text{train}}} \mathcal{L}\left[a^{\text{train}}_{[i_r,t]}, \gamma\right]$
2: **for** $m = 1$ **to** M **do**
3: $\quad \forall t \in [1, m_{\text{train}}]$ compute $r_{t,m} = -\left\{\dfrac{\partial \mathcal{L}\left[a^{\text{train}}_{[i_r,t]}, F(y)\right]}{\partial F(y)}\right\}_{F(y)=F_{m-1}\left(y^{\text{train}}_{[:,t]}\right)}$
4: $\quad$ Fit a regression tree to residuals values r_{tm} (**algorithm 3**)
5: $\quad$ Create terminal regions R_{jm} for $j = [1, J_m]$
6: $\quad \forall j \in [1, J_m]$ compute $\gamma_{jm} = \arg\min_\gamma \sum_{y^{\text{train}}_{[:,t]} \in R_{jm}} \mathcal{L}\left[a^{\text{train}}_{[i_r,t]}, F_{m-1}(y_t) + \gamma\right]$
7: $\quad$ Update reconstruction function $F_m(y) = F_{m-1}(y) + \lambda \sum_{j=1}^{J_m} \gamma_{jm} I(y \in R_{jm})$
8: **end for**

through 7 generations. Hyperparameters are the number of hidden layers (1,2 or 3), the number of neurons in each layer (r, $2r$, p, $p/2$ or $r+p$), the activation function (relu, tanh, σ or linear), the dropout rate (0.0, 0.1, 0.2 or 0.3) and the learning rate ($10^{-4}, 10^{-3}, 10^{-2}$ or 0.1).

Algorithm 5 K-fold cross-validation

Require: training data, a form of a model with **given** hyperparameters
1: divide **all training data** into K_v folds of equal size
2: define K_v different train - validation partitionings
3: **for** $k = 1$ **to** K_v **do**
4: $\quad$ learn model k on train data
5: $\quad$ evaluate model k on validation data via $\text{score}_{\text{val}}$ k e.g. -MSE or R^2
6: **end for**
7: compute the generalization score as the average of validation scores
8: train the final model using **all training data** and hyperparameters leading to best cross-validation score

Normalization. To speed up learning and convergence, regressive models are learned on normalized data (zero mean, unit variance). Defining f_s the fitted function, the reconstruction procedure is:



A summary of all investigated regressive methods is given in table 3. Models are implemented using scikit-learn [52] and keras [53] libraries.

3. Presentation of cases

This section presents the three cases investigated in this paper. Simulations were performed using OpenFOAM.

3.1. Case 1

A 2D cylinder is placed in a uniform flow. The diameter is $D = 2m$ and the reference velocity is $U_\infty = 1\ m/s$. The Reynolds number based on D and U_∞ is 200 and the computational domain is $x \in [-10D; +15D]$ and $z \in [-10D; +10D]$ with x the longitudinal direction and z the vertical. The domain is discretized into 9200

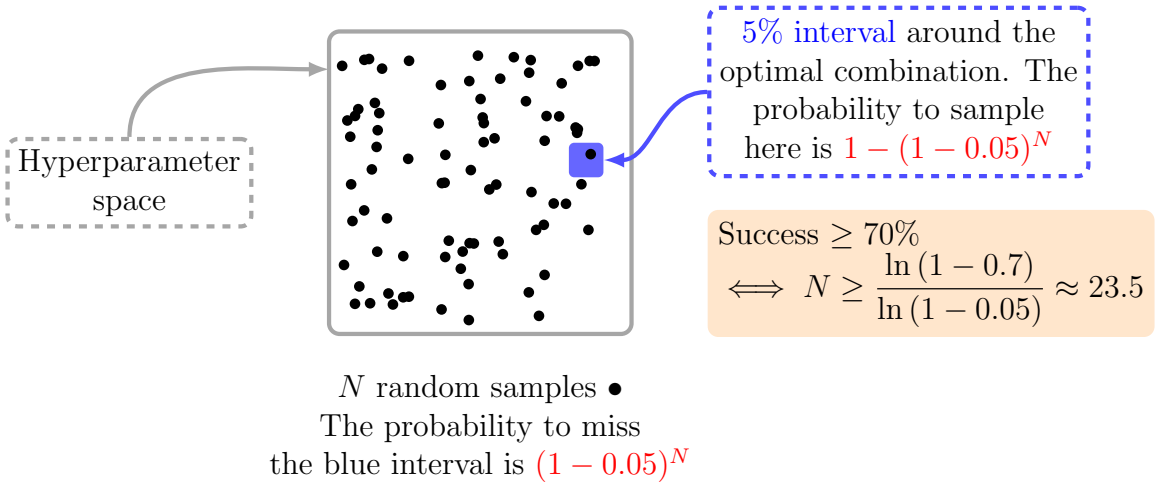


Figure 8: Randomized grid search. To hit the optimal hyperparameters space region with 70% of success and 95% confidence, a random exploration of the grid space with $N \approx 24$ points is enough

Method	**Linear regression**	**Support Vector Regression**	**Neural Network**	**Boosted Trees**
Description	Latent state linearly explained by measurements	Linear regression performed in higher dimension	Interconnected neurons with nonlinear activations	Decision trees trained based on previous trees' errors
Multitask	Possible	No	Yes	No
Parameters	β	α	weights and biases	Non parametric
Regularization	Lasso	Box constraint	Dropout	Learning rate
Hyperparameters	Lasso coefficient	Kernel Box constraint	Dropout rate Number of hidden layers Activation functions Learning rate	Number of estimators Maximum depth of trees Learning rate
Hyperparameters optimization	Grid search	Randomized grid search	Genetic algorithm	Randomized grid search
Benefits	One regression Simple method	Kernel trick	Flexibility	Input space partionning
Drawbacks	Too simple?	r regressions	Black box to tune	Instability r regressions

Table 3: Table of regressive functions to recover the reduced state from limited measurements.

cells and a DNS simulation is performed. The time step is $\Delta t = 0.05s$ and 450s are simulated. To reach periodic vortex shedding, around 200s are required. The remaining 250s are kept for training and testing reconstruction methods. The solution is sampled each $\Delta t^* = 1$ where the non-dimensional time is defined as $t^* = tU_\infty/D$. The 250s of simulation therefore accounts for 125 snapshots. The Strouhal number is estimated from the power spectrum of the instantaneous lift coefficient (Fig. 9b). It is found equal to 0.191, which is close to values observed in the literature

[54]. One shedding cycle corresponds to a non-dimensional period $T^* = 5.23$ and approximately 24 shedding cycles are observed in simulation data. Given the 70/30 splitting strategy, 16 shedding cycles are observed in training data. This seems reasonnable to fully characterize the dynamics considering the simplicity of the flow. Reduction and reconstruction are performed in the region centered around the cylinder and the wake i.e. $x \in [-5D; +15D]$ and $z \in [-3.75D; +3.75D]$ which accounts for 5840 cells.

3.2. Case 2

A DNS of a 2D spatial mixing layer is performed. The upper (fast) and lower (slow) stream velocity are respectively $U_1 = 30m/s$ and $U_2 = 10m/s$. The initial vorticity thickness is $\delta_{\omega,0} = 1m$ and the inlet profile is a hyperbolic tangent reading:

$$U(x = 0, z) = U_c + \frac{1}{2}\Delta U \tanh\left(\frac{2z}{\delta_{\omega,0}}\right)$$

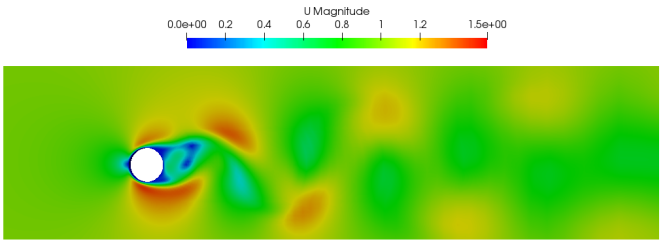
where the velocity difference is $\Delta U = U_1 - U_2 = 20m/s$ and the convective velocity is $U_c = (U_1 + U_2)/2 = 20m/s$. The Reynolds number based on $\delta_{\omega,0}$ and ΔU is $Re = 500$. Stochastic perturbation is added to the inlet profile to trigger the Kelvin Helmotz instability. The flow is simulated with a physical time step $dt = 0.01s$ and two runs are performed: first, 250s are simulated to ensure convergence of statistics (mean flow and standard deviation); second, 135s are simulated from the first solution to produce relevant snapshots. For validation purposes, the streamwise momentum thickness with respect to initial vorticity thickness is plotted in fig. 9d. The linear growth region (least square fit) with $d\delta_\theta(x)/dx \approx 0.020$ is in the range of values from the literature [55]. The non-dimensional time is defined as $t^* = t\Delta U/\delta_{\omega,0}$ and the solution is sampled each nondimensional time unit, leading to a total of 2700 snapshots. This number is consistent with the choices from the literature [34, 3]. The complete domain $x \in [0; 180\delta_{\omega,0}]$ and $z \in [-28\delta_{\omega,0}; 28\delta_{\omega,0}]$ is reconstructed, corresponding to 13680 cells. Given the unsteady location of the mixing layer, one convection cycle takes at most $L/U_c = 9s$ where L is the length of the domain. This corresponds to 180 non dimensional time units, leading to an ensemble of snapshots that covers at least 15 convection cycles with possible vortex pairing.

3.3. Case 3

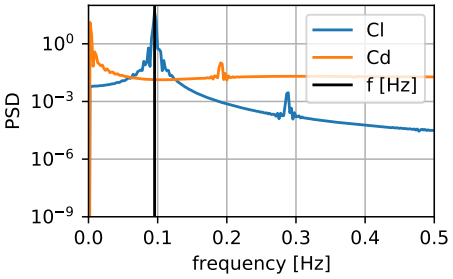
Reconstruction methods are applied in the wake of a square cylinder. This flow is computed with a large eddy simulation (LES) performed in a three-dimensional domain. A Smagorinsky model is used to account for the subgrid scales. The diameter is $H = 0.04m$ and the reference velocity is $U_\infty = 0.5\ m/s$. The Reynolds number based on H and U_∞ is 20000 and the computational domain is $x \in [-4H; 15H + H/2]$, $y \in [-5H; +5H]$ and $z \in [-7H; +7H]$ with the cylinder defined by the box $(-H/2; -5H; -H/2) \rightarrow (+H/2; +5H; +H/2)$. The computation is initialized with a RANS $k-\omega$ solution and periodic boundary conditions are applied in the y direction. The time step for the LES is $\Delta t = 0.0005s$ and $120s$ are simulated. Only the last $105s$ are kept for reconstruction. The total number of cells is 271 040. The nondimensional time is defined as $t^* = tU_\infty/H$ and the solution is sampled each nondimensional time unit, leading to a total number of 1312 snapshots. From the spectra in fig. 9f, St $= 0.132$ which is close to values observed in several experiments [56]. One shedding cycle therefore corresponds to $T^* = 7.58$ and approximately 173 shedding cycles are observed in simulation data. With the 70/30 splitting strategy, 120 cycles are then used to learn dominant patterns and reconstruction models. Reduction and reconstruction are performed in the box delimited by $(-1.25H; -1.25H; -3H)$ and $(+12.5H; +1.25H; +3H)$, accounting for 48023 cells.

4. Results

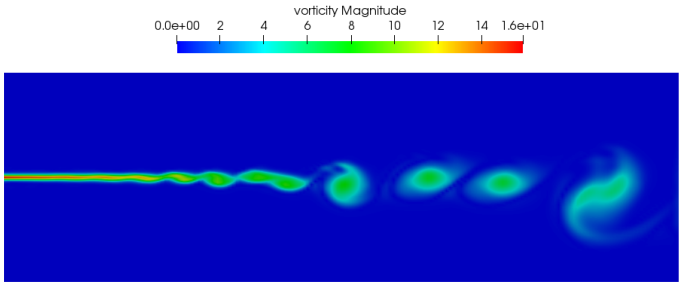
The enhanced clustering algorithm gives 500 clusters that optimally describe variability in the training data. Sensors are defined as centroids



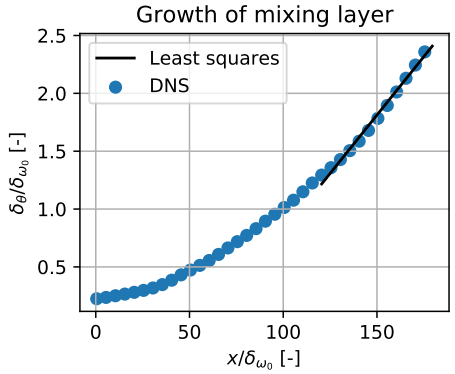
(a) Velocity magnitude (m/s) for case 1



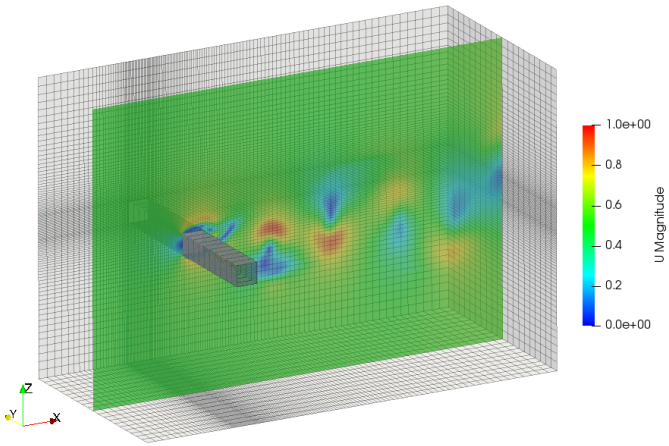
(b) PSD of lift and drag coefficients (case 1)



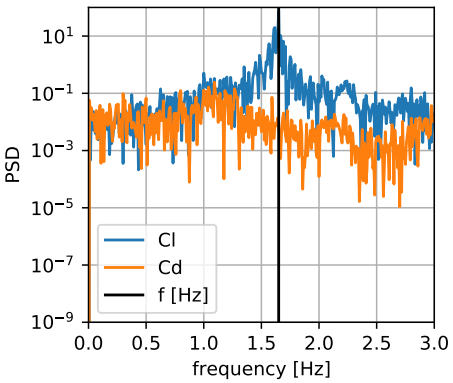
(c) Vorticity magnitude (s^{-1}) for case 2



(d) Growth rate (case 2)



(e) Slice colored by velocity magnitude (m/s) for case 3



(f) PSD of lift and drag coefficients (case 3)

Figure 9: Visualizations of the three cases considered in this paper. For all cases, x is the longitudinal direction, z is the vertical direction and y is the spanwise direction.

Case	$p_{0.2}$	$p_{0.4}$	$p_{0.8}$
1	8	22	92
2	18	42	122
3	39	99	372

Table 4: Number of sensors for each case

recovering 20%, 40% or 80% of the variance. Considering the two or three components of the velocity field to estimate, the total number of sensors p is summarized in table 4. The choice of $K_c = 500$ is arbitrary and could be optimized using the heuristic elbow criterion [57]. Figure 10 gives some visualizations of clustering results. For case 1, Voronoi cells close to the cylinder are displayed. The four sensors corresponding to $p_{0.2} = 8$ are also indicated in red. For the mixing layer, sensors for $p_{0.8}$ are indeed located where the variability (i.e. the root mean square) of training data is the most important. For the 3D cylinder, sensors for $p_{0.8}$ are placed in the 3D domain, in the near wake of the square cylinder.

To reduce the dimension from n to $r \ll n$, POD and autoencoders are considered. The latent space dimension is determined by the number of POD modes required to recover 40%, 80% and 99% (case 1) or 95% (other cases) of the training data variance. This choice may overestimate the optimal latent space dimension for the VAE but this prevents from heavy cross validation. Latent space dimensions are summarized in table 5.

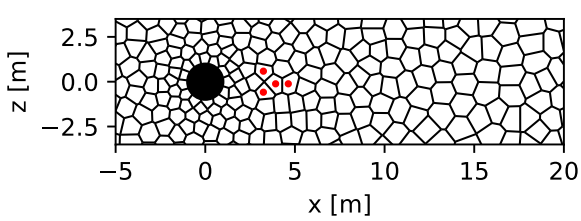
Case	n	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$ (case 1) or $r_{0.95}$
1	11 680	1	2	5
2	27 360	3	11	32
3	144 069	2	21	177

Table 5: Latent space dimension for each case

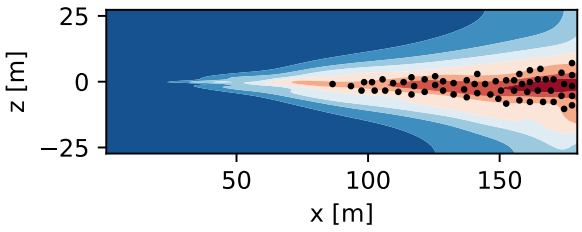
Given a dimension for the latent space and a reduction method, a set of spatial modes with associated dynamics is extracted from training data. For the POD method, spatial patterns are hierarchically sorted by the recovered variance and first modes are defined as coherent structures. For iterative autoencoders methods, no order and no orthogonality constraint is imposed. Fig. 11 compares first and fifth POD modes for the mixing layer and supports that most (respectively lower) energetic modes are associated to low (respectively high) frequency dynamics. Fig. 12 shows an LAE mode and confirms that nonorthogonality makes more visible the vortex shedding process as argued in [13]. The evolution of the cost function is also shown to verify that learned modes are converged and relevant for testing data. Fig. 13 shows two dominant modes of 3D cylinder with the VAE reduction. Despite close resemblances between the first dominant VAE mode and the first POD mode, higher order modes are different because the VAE focus both on an energy constraint and a regularity constraint.

Interestingly for cylinder cases, most of the modes captured by autoencoders are associated with the vortex shedding period. To support this result, the power spectra for each component of the latent state is performed. Histograms of dominant frequencies are plotted in figure 14. While the POD algorithm extracts hierarchical structures with different frequencies, most of the time series found by neural encoders have a dominant frequency corresponding to the one used for the Strouhal computation i.e. $f = 0.0952$ Hz for case 1 and $f = 1.65$ Hz for case 3. As an illustration, using VAE on the square cylinder yields more than a hundred modes out of 177 fundamentally oscillating at the vortex shedding frequency.

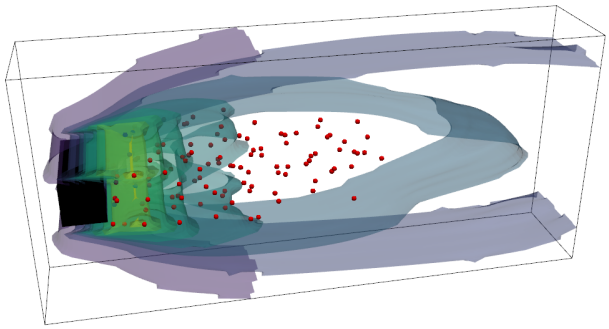
To quantify the error in the reduction process, the NMSE of auto-encoded velocity fields is computed. Results are summarized in table 6. As expected, the reduction error is lower when increasing the dimension of the latent space. Besides, spatial modes learned on training



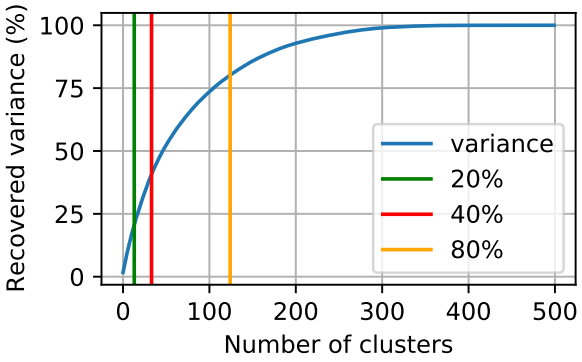
(a) Voronoi cells and $p_{0.2}$ sensors for case 1



(b) $p_{0.8}$ sensors and contours of $[u_{\text{rms}}]_x$ for case 2



(c) $p_{0.8}$ sensors and isosurfaces of $[u_{\text{rms}}]_x$ for case 3



(d) Cumulative variance for case 3

Figure 10: Some visualizations of the enhanced clustering algorithm results

data generalize to testing data with a good extent (testing errors are close to training errors). Concerning the reduction strategy, POD and LAE yield close reduction errors which is legit given that LAE and POD modes span the same subspace [58]. With variational approaches, reduction errors are much higher but this is compensated by a better organized latent space. To perform visualizations, the first two dominant signals (in terms of variance) are extracted by the singular value decomposition of latent time series. The associated 2D scatter plot is colored according to a three cluster membership with clusters being learned on POD dynamics. Ellipses correspond to $\pm 2a_\sigma$ intervals projected onto the latent space (see fig. 15). These visualizations show that the range for variationnal spaces is narrower compared to the LAE space. Besides, samples located in the confidence intervals should have a meaningful decoding by construction which is sought for robustness. To verify this, the impact of noisy latent states during the decoding process is investigated. Noise is applied to each latent time series following:

$$a_{[i_r,:]} \leftarrow a_{[i_r,:]} + \epsilon \quad \text{with } \epsilon \sim \mathcal{N}\left(0, I_a \left[\max a^{\text{train}}_{[i_r,:]}\right]\right)$$

where I_a is the noisy instensity. Noisy time series are then decoded and reconstruction errors are computed. The robustness ratio is defined as:

$$\gamma = \frac{\text{NMSE}_0}{\text{NMSE}_{I_a}}$$

where NMSE_0 is the error when no noise is applied and NMSE_{I_a} is the error with noisy time series. When γ is close to 1, noise has no influence in the decoding process while γ close

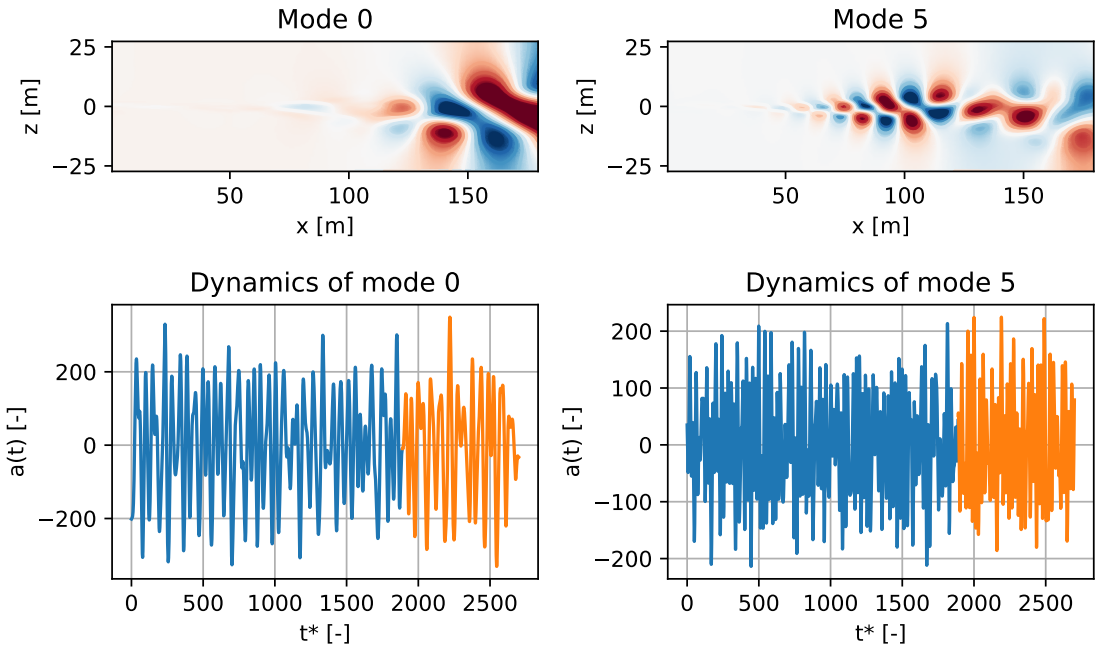
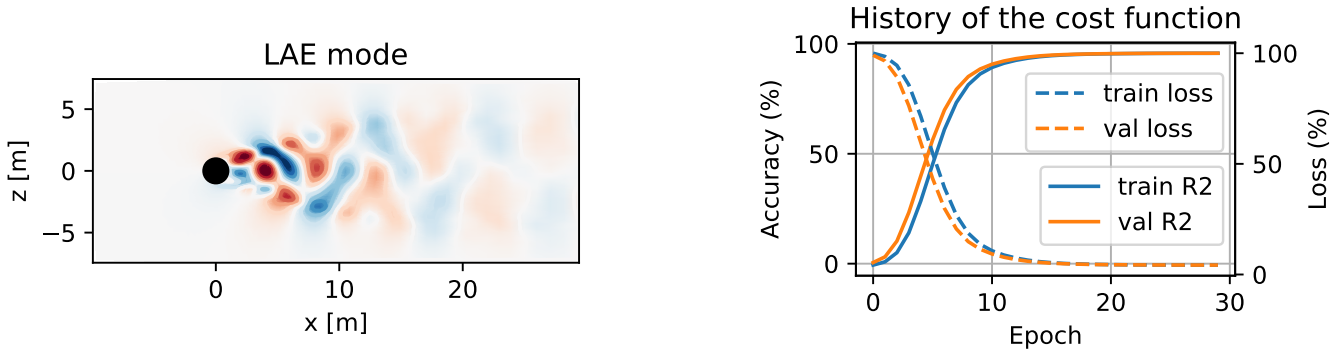


Figure 11: Two POD modes (x component) for the mixing layer and their associated dynamics. The latent variable is colored in blue for the training part and orange for the testing part.



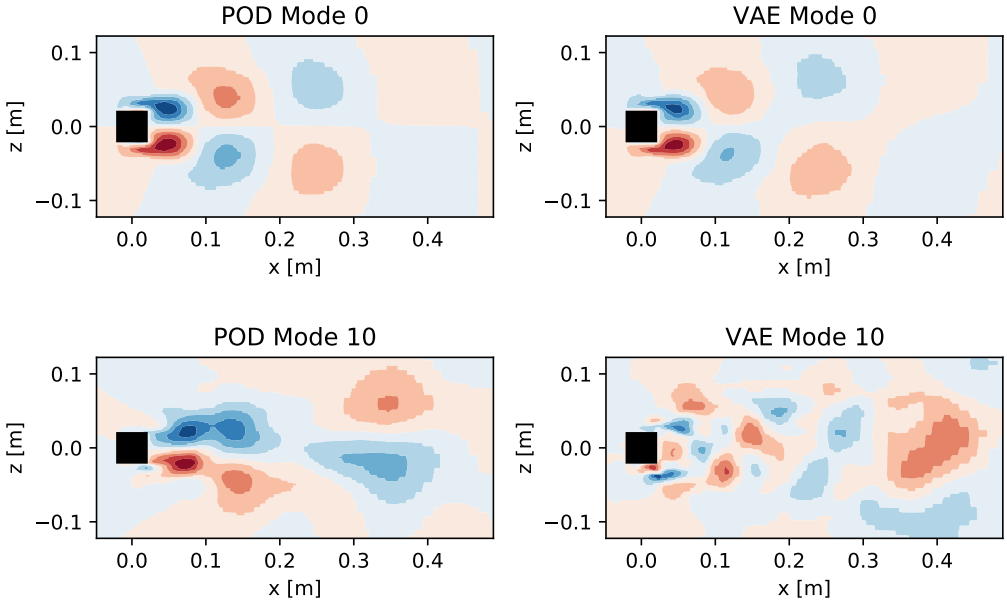
(a) One LAE mode

(b) Learning history of the LAE

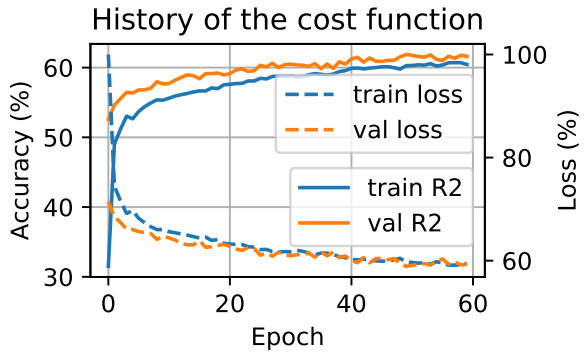
Figure 12: Some LAE results for case 1. For the history of the cost function, "accuracy" (continued lines) refers to the determination coefficient. Reaching 1.0 for the training and the validation score is not surprising given the simple periodic behaviour of the flow. Note that the validation set corresponds to the last 20% of training data and serve as a monitor to early stop the learning.

to 0 means a high sensitivity to noise. In the latter case, it means that decoding directions are not robust and lead to meaningless velocity fields $\hat{u}$. Testing results for the mixing layer and the 3D cylinder reduced with $r_{0.8}$ modes are shown in figure 16. Despite higher global errors when autoencoding velocity fields with VAE, the nonlinear decoder proves to more robust when decoding noisy latent dynamics.

Reduction methods can also be compared in terms of recovered spectra. Fig. 17 shows the singular values of testing auto-encoded velocity fields with $r_{0.8}$ modes. For linear reduction



(a) Two dominant VAE modes (extracted with singular value decomposition) and the POD modes of same order.

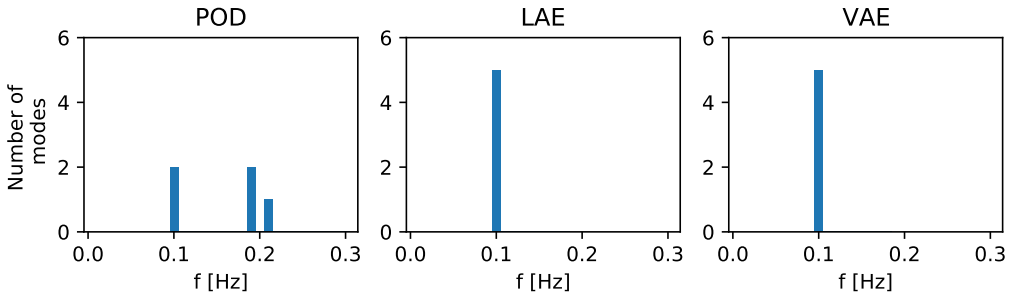


(b) Learning history of the VAE.

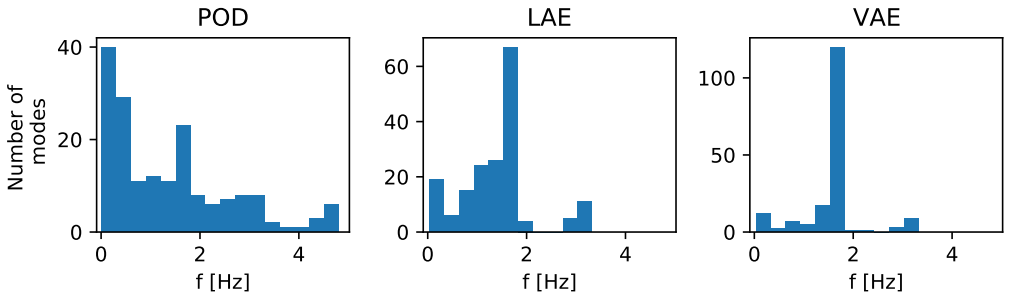
Figure 13: Some VAE results for case 3. POD modes and dominant VAE modes are different because VAE modes are learned with an energetic criterion **and** a regularity constraint (as opposed to POD modes, solely based on the variance).

methods, only high singular values are recovered, the lowest ones being cut according to the latent space dimension (low-pass filter). The VAE seeks to reproduce the entire spectra but this is to the detriment of the coherent structures estimation. Results could possibily be improved by considering deep variational autoencoders but this is out of the scope of this paper. Finally, some NMSE maps are shown in fig. 18. For the 2D cylinder reduced with the VAE, the auto-encoding error is non zero in the wake of the cylinder which is not surprising given that the mean square error is not the reduction criterion. For the mixing layer linearly reduced at 80% of kinetic energy, the error is maximum at the origin of the mixing layer. For the 3D cylinder reduced at 95% of energy, POD and LAE errors are nearly identical because both set of modes span the same subspace.

Latent variables are now estimated using measurements. A total number of 540 regressive models were trained and cross-validated while 81 direct reconstructions with λ optimization



(a) Dominant frequencies for case 1 and $r_{0.99} = 5$

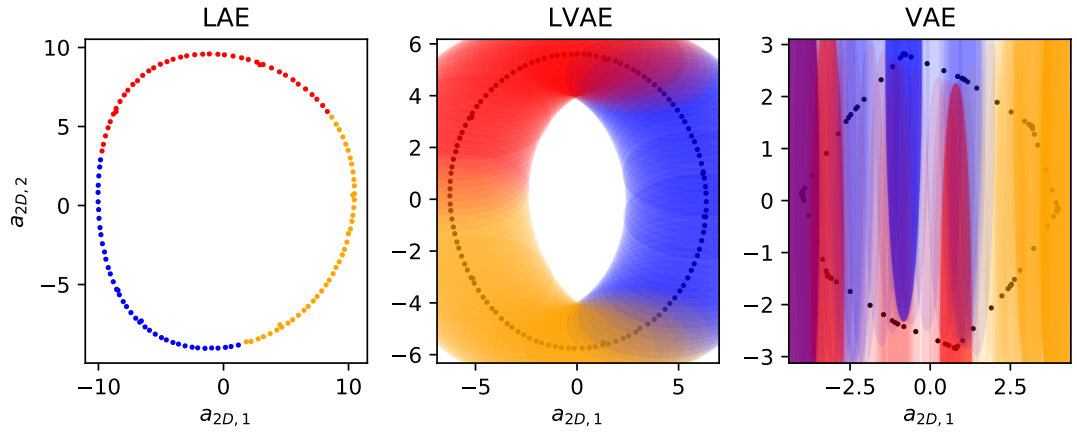


(b) Dominant frequencies for case 3 and $r_{0.95} = 177$

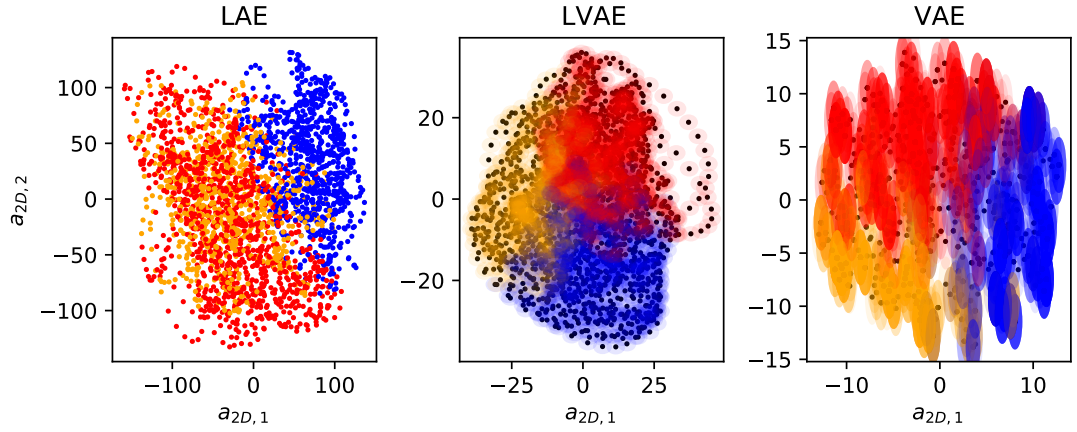
Figure 14: Histograms of dominant frequencies in training latent time series. For the POD reduction, each mode oscillates at a different fundamental frequency. For the LAE and VAE reductions, most of the modes fundamentaly oscillate at the vortex shedding frequency which is a consequence of non orthogonality.

Method	**Case**	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$ (case 1) or $r_{0.95}$
POD	1	[12.88 12.60]	[1.34 1.32]	[0.12 0.12]
	2	[22.79 23.50]	[8.88 9.59]	[2.40 3.13]
	3	[45.05 43.26]	[29.14 31.17]	[9.37 15.40]
LAE	0	[13.00 12.68]	[1.34 1.32]	[0.12 0.12]
	1	[22.83 23.57]	[9.22 9.95]	[2.87 3.61]
	2	[45.65 44.06]	[30.29 32.11]	[11.65 16.9]
LVAE	0	[12.69 12.51]	[1.53 1.50]	[1.74 1.72]
	1	[22.84 23.60]	[9.02 9.80]	[2.80 3.55]
	2	[45.76 44.17]	[32.13 33.75]	[25.40 28.35]
VAE	0	[25.09 24.79]	[16.6 16.31]	[3.72 3.69]
	1	[29.77 30.65]	[23.38 24.26]	[19.43 20.66]
	2	[51.59 50.43]	[41.16 41.41]	[32.86 35.55]

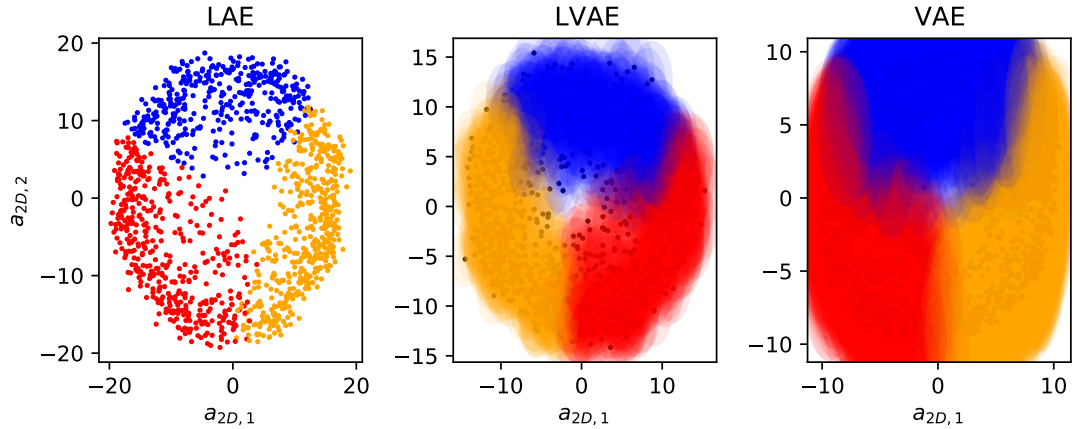
Table 6: NMSE results when autoencoding velocity fields. Values are in percentage and given as [train, test] errors.



(a) 2D cylinder - With $r_{0.99} = 5$

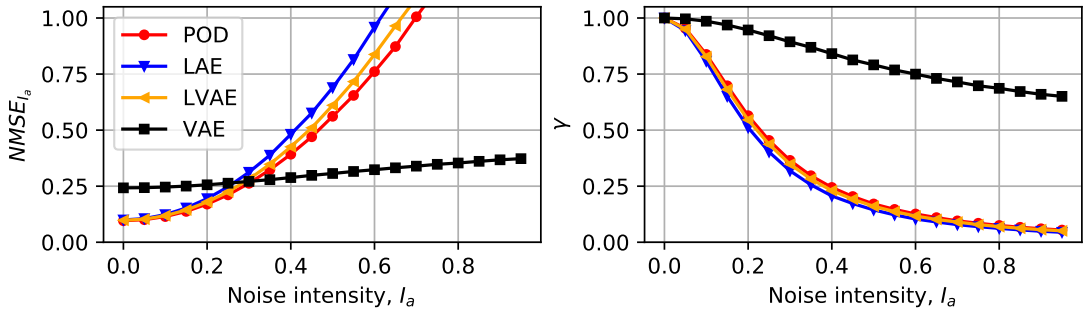


(b) Mixing layer - With $r_{0.80} = 11$

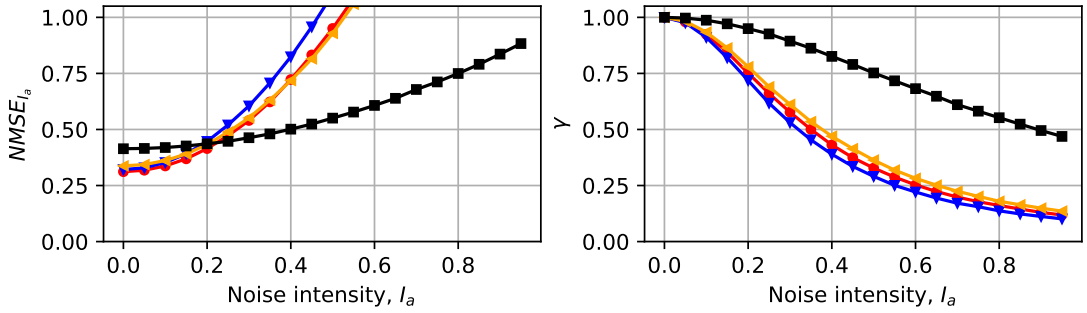


(c) 3D cylinder - With $r_{0.95} = 177$

Figure 15: Latent space visualization for all cases. Ellipses in variational spaces correspond to the 95% interval around mean encodings. Variational spaces have a narrower range compared to LAE spaces which is consitent with the constraint of overlapping distributions.

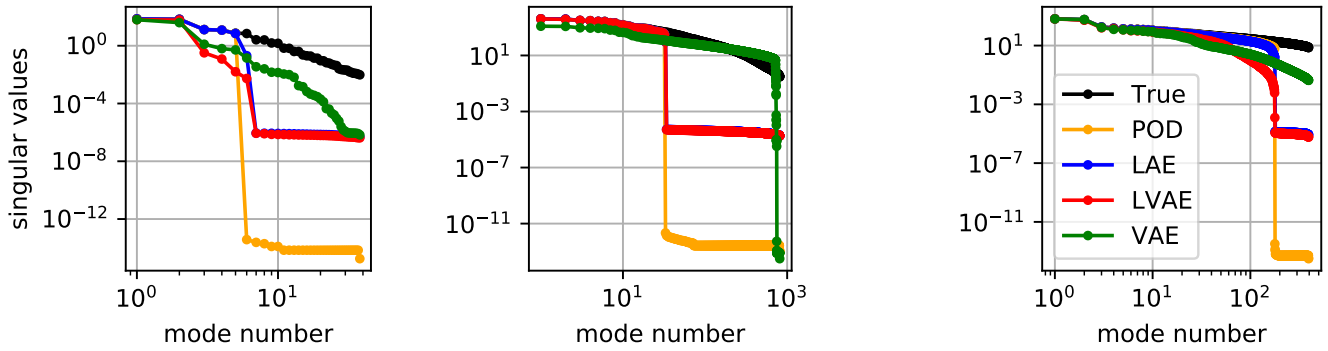


(a) Mixing layer - With $r_{0.80} = 11$



(b) 3D cylinder - With $r_{0.80} = 21$

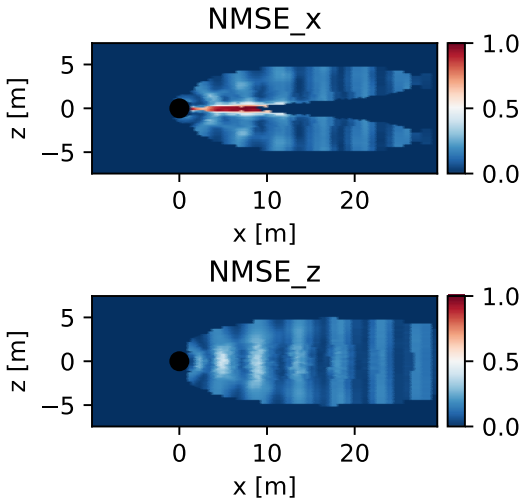
Figure 16: Influence of noise applied on the latent state when auto-encoding high-dimensional data. Decoding directions of the VAE appear more robust to noisy latent dynamics.



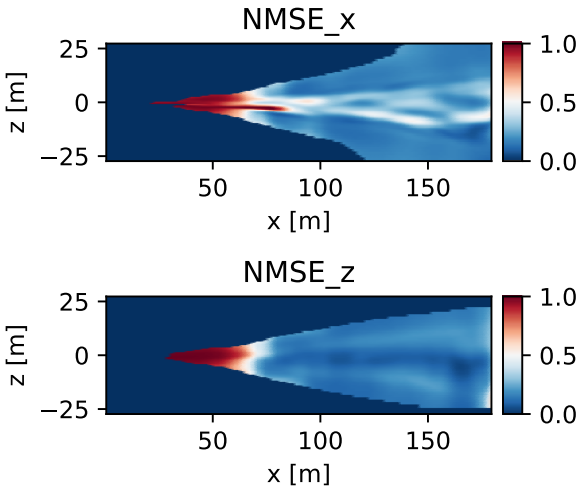
(a) 2D cylinder (b) Mixing layer (c) 3D Cylinder

Figure 17: Spectra of autoencoded testing velocity fields. Linear reduction methods act as low-pass filters while the nonlinear reduction seeks to reproduce the whole spectra.

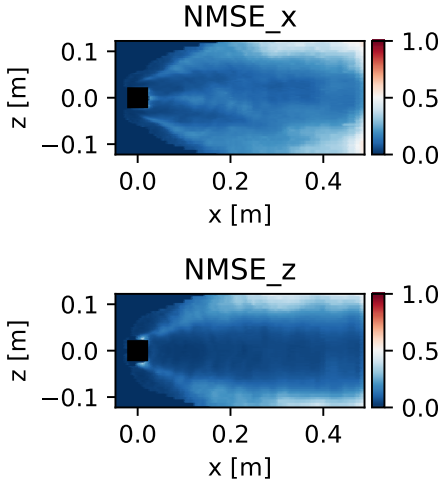
were performed. Results for all cases, all reconstruction methods and all reduction methods are summarized in tables A.8, A.9 and A.10. Both training and testing errors are given, the first being of interest for analysis tasks, the second being more relevant from a practical viewpoint. Some examples of cross-validation for the mixing layer are given in figure 19. The linear



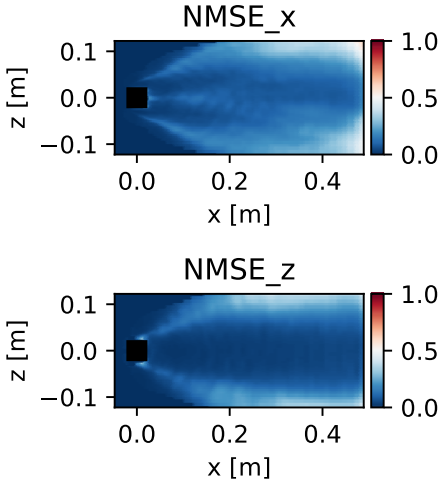
(a) 2D cylinder with $r_{0.99} = 5$ modes and VAE reduction. The error in the wake is non zero as opposed to other reduction methods.



(b) Mixing layer with $r_{0.8} = 11$ modes and POD reduction. The error at the origin of the mixing layer vanishes for $r_{0.95}$.



(c) 3D cylinder with $r_{0.95} = 177$ modes and LAE reduction.



(d) 3D cylinder with $r_{0.95} = 177$ modes and POD reduction. Same as LAE reduction because both modes span the same subspace.

Figure 18: NMSE maps of auto-encoded testing velocity fields.

regression requires the α hyperparameter which is optimized by grid search. In its multitask formulation, α is unique for all the modes, leading to a β^T matrix with some columns equal to zero. In its single-task formulation, α may be different for all the modes and β^T is sparse. For the support vector regression and gradient boosted trees, each component of the latent space is regressed independently. For each mode, 25 combinations of hyperparameters are randomly sampled in the hyperparameters space, yielding a matrix of cross-validation scores. Combinations with the highest R^2 score (black dots) are used to train the final models. For the neural network, candidate hyperparameters evolve for several generations and only best

candidates survive. The final population contains best neural networks, trained with best hyperparameters. The direct reconstruction optimization is illustrated in figure 20 for case 3: several λ hyperparameters are tested when reconstructing training data and the one leading to the best determination score is kept for testing.

A deeper analysis of the results tables that all reconstruction methods yield close errors for a given case, a latent dimension and a grid of sensors. Some outliers for the direct reconstruction can be noticed for case 2 and case 3; this is attributed to the simplex algorithm used for solving the underdetermined system of equations. Not surprisingly, increasing the number of sensors and the dimension of the latent space leads to lower testing errors. As an example, the influence of sensors density is visualised in fig. 21 for the mixing layer.

The effect of noise in measurements is also explored. Considering a noise intensity I_y, each component of the measurement vector is modified according to:

$$y_{[i_p,:]} \leftarrow y_{[i_p,:]} + \epsilon \quad \text{with } \epsilon \sim \mathcal{N}\left(0, I_y \max[\, y_{[i_p,:]}\,]\right)$$

Reconstructions are then performed and determination coefficients are computed. Results (averaged over several runs) are illustrated in figure 22 with the mixing layer and the 3D cylinder. Not surprisingly, the higher the noise level, the higher the impact on the global determination score. Besides, the noise preferentially impacts high order modes when considering POD reduction. It can also be noted that despite differences in formulations (cost function, hyperparameters), each model gives similar results which is attributed to the systematic cross validation of hyperparameters. The choice of one method towards another therefore depends on the user specifications: combination of measurements or decision from the measurements? black box or white box? few or lots of parameters? heavy cross-validation or not? multitask or not? etc. The choice may also be guided by the computational complexity as shown in table 7 with the CPU time to learn regressive models of the turbulent flow around the cylinder[1].

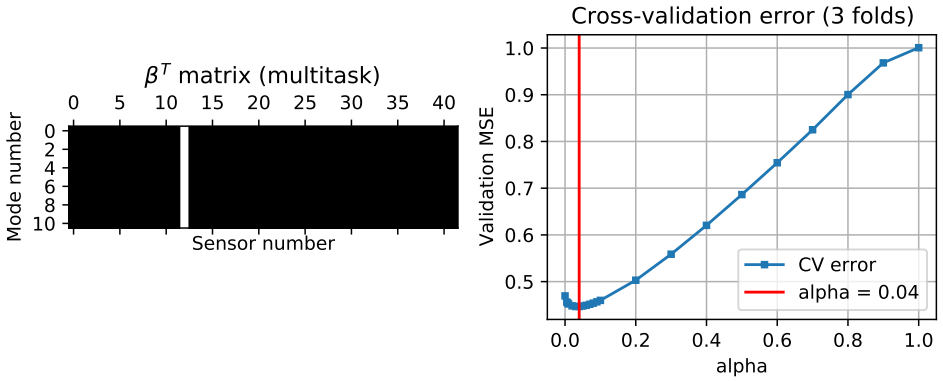
5. Discussion and perspectives

Overall, the results suggest that machine learning methods can be used for fluid flow reconstruction. The choice of one strategy towards another depends on the data availability, the model's interpretability, the implementation cost and the computational cost.

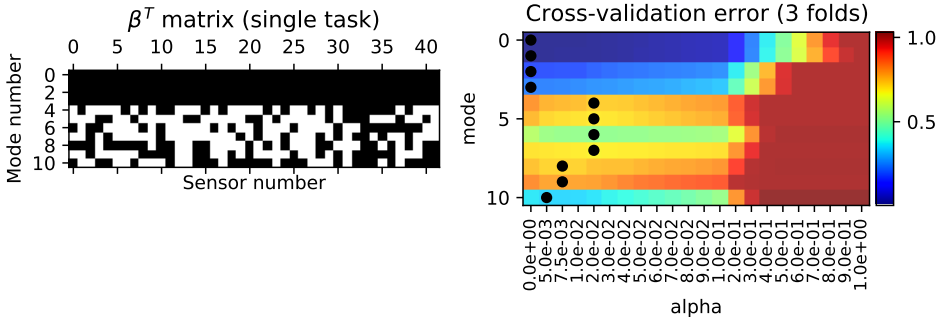
The **data availibility** refers to the snapshots ensemble used for learning methods. Snapshots must be representative of the flow and the number of available samples can play a decisive role during training. Neural networks are known to require a tremendeous amount of data to be efficient, while linear regressions can yield reasonable performance with a small snapshots ensemble. In this paper, the number of snapshots was chosen empirically, based on a macrocospic scale. This number may be optimised with a parametric study (do the scores change with an enriched dataset?) or with a deeper analysis of the flow. One may consider the use of the Hurst exponent [29], integral time scales or the Lyapunov exponent [59].

The **interpretability** is the degree to which a machine decision is understandable. Decision trees are very interpretable since the decision path to explain the final output is human readable. At the opposite, deep neural networks are black boxes where the final output is obtained by a complex feedforward process. To make machine learning more interpretable, physical constraints can be added in the formulation [60, 61]. A good machine learning model should be interpretable (few active terms i.e. parcimonious) and generalisable (robust to new configurations).

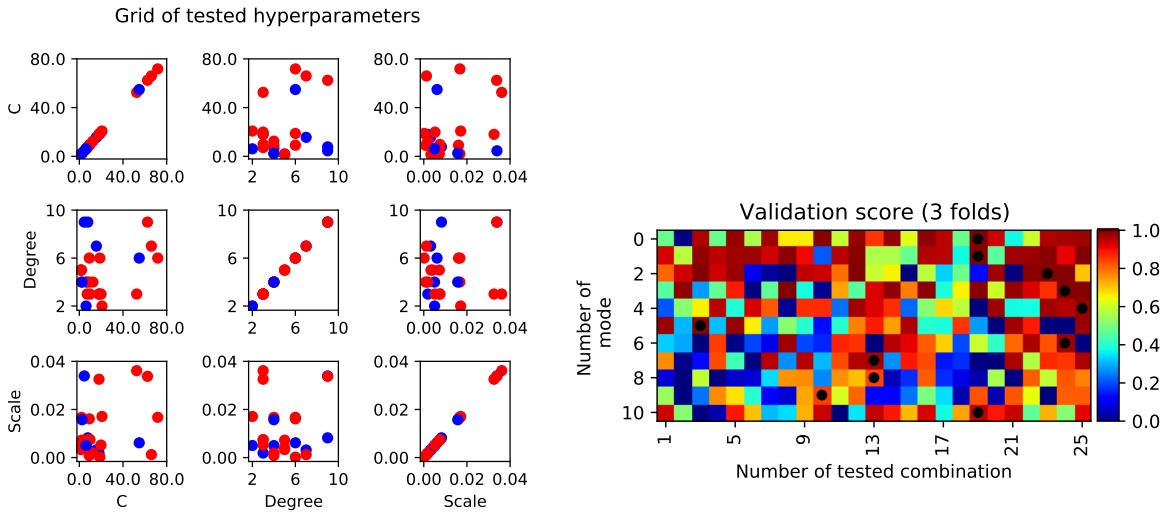
[1] Note that learning all the models to produce tables in appendix took around 326 CPU hours.



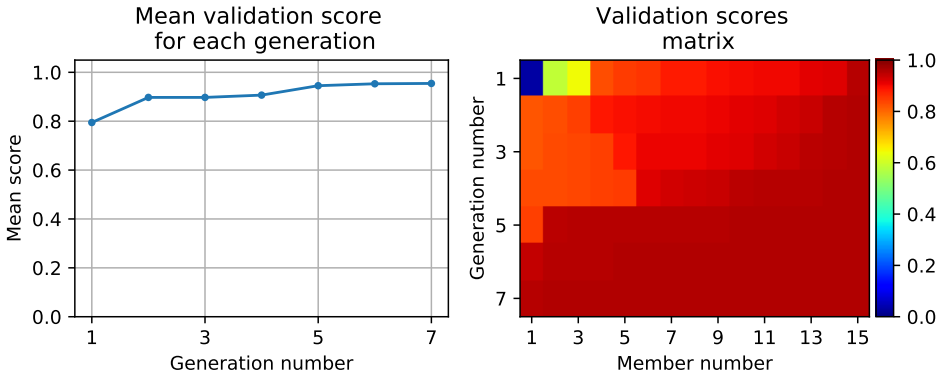
(a) Lasso cross-validation (multitask formulation). The contribution of one sensor is set to zero for all modes (blank column).



(b) Lasso cross-validation (single task formulation). The selection of sensors is different for each mode.



(c) Support Vector Regression cross-validation. The procedure is identical when validating Gradient Boosted Trees. At the opposite of the Lasso optimization, hyper-parameters are randomly tested hence the chaotic organization of the validation score matrix. On the left figure, blue dots indicate a polynomial kernel while red dots indicate a Gaussian kernel.



(d) Genetic optimisation of neural network hyperparameters. At the end of the process, each neural network has a validation score close to unit meaning that bad hyperparameters didn't survive across generations.

Figure 19: Examples of cross-validation to optimize hyperparameters of regressive methods. Results are illustrated on the mixing layer reduced with $r_{0.80} = 11$ POD modes and reconstructed with $p_{0.40} = 42$ sensors.

Method	Noteworthy	CPU time
POD	SVD computation	1 minute
LAE **LVAE** **VAE**	Iterative procedure No hyperparameter optimisation	3 minutes 6 minutes 12 minutes
Direct reconstruction	Reconstruction of all training and testing data One optimisation per reconstruction	[5, 5, 5, -] minutes
Neural network	Genetic optimisation Neural network fitting	[15, 17, 30, 21] minutes
Linear single task	One regression per mode Lasso with l_1 norm	[3, 3, 3, 3] minutes
Linear multi-task	One regression for all modes Lasso with l_{21} norm	[43, 39, 36, 23] minutes
Support vectors **Gradient Boosting**	Random grid search One optimisation per mode	[6, 6, 6, 12] minutes [26, 25, 26, 24] minutes

Table 7: Computational time to perform $r_{0.8}$ reductions and $p_{0.8}$ reconstructions for case 3. Values are given as [POD, LAE, LVAE, VAE] CPU times. Computations were performed sequentially with one node from NEC SATOR HPC.

The **implementation cost** refers to the mathematical formulation of the method. In general, the more flexible the method, the more complex the implementation.

The **computational cost** refers to the computational ressources required for learning. Depending on the cross validation strategy and the number of parameters to learn, the learning phase can be heavy. Neural networks are particularly greedy during learning but once weights and biases are known, the evaluation of the model is fast. Applied to case 1, linear regression coupled to POD reduction is a way more reasonable strategy than neural network coupled to VAE reduction: both strategies capture a low dimensional representation of the vortex shedding but the linear procedure

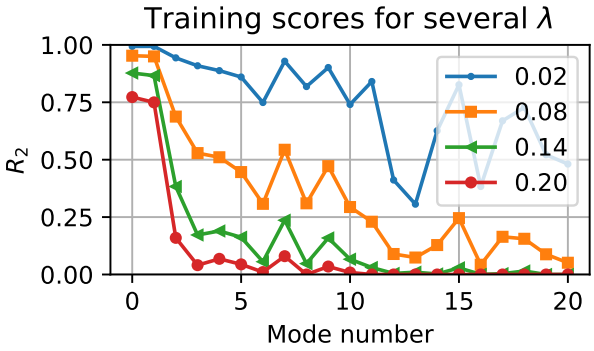
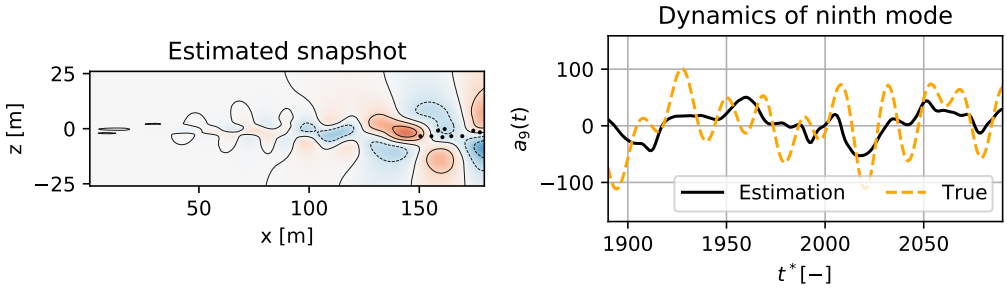


Figure 20: Selection of the λ hyperparameter for the direct estimation of case 3 (POD reduction with $r_{0.8}$ modes and $p_{0.8}$ sensors)

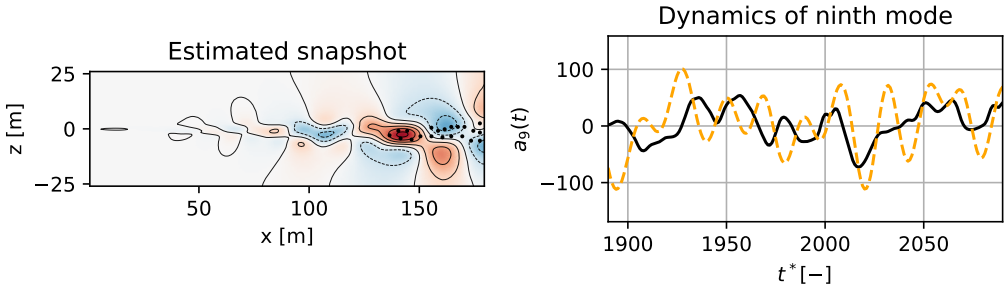
is much faster and interpretable than the nonlinear one. For case 3, a linear regression is more discussable if the model is to be used for real-time estimation. The choice of LVAE (for robustness to noise estimate) with a neural network (for rapid evaluation) is more tailored for such task. The computational cost is probably the main obstacle for applying the procedure to industrial flow fields: storing and processing the matrix of snapshots would require lots of memory unless reduction and reconstruction are locally performed[2].

It is important to note that models were learned for fixed inlet conditions: the Reynolds number for each case and the stochastic perturbation for the mixing layer. An interesting extension would consist in learning models covering several regimes. Following the idea of Morton *et. al* [31], generative models could be used to generate relevant data for an unseen case, given a context parameter. Applied to the mixing layer, such a strategy could be used to learn the sensitivity to inlet conditions and control the vortex pairing [62]. Another comment concerns the scalability of the method. The proposed framework was applied to increasing complexity, yet canonical flows. For highly turbulent flows where nonlinearities dominate, linear reconstructions may be limited. Given the flexibility of neural networks, deep learning models with a good choice for hyperparameters should yield reasonable results. This however requires a tremendeous amount of relevant data and a good learning procedure to avoid vanishing or exploding gradients [63]. Concerning the measurement matrix, sensors were positioned according to an energetic criterion. Instead of clustering the mesh as a whole (thus putting all mesh refinements on an equal footing), one may improve the method by considering local clusterings. In doing so, the enhanced clustering algorithm could be more relevant near walls. Other strategies based on the controllobability and the observability of the sensors may also be investigated as this could give better insights for measuring the flow [19]. Finally, direct and regressive reconstructions are more likely to be used for offline applications and flow analysis. For real-time estimation, data assimilation should be preferred [64] but this requires a dynamical model and the estimation of uncertainties. To learn a probabilistic [34] or deterministic [28, 29] dynamical model of the latent state, data-driven procedures may be considered. This requires the latent state to have a temporal regularity which was not observed for the variationnal auto-encoded mixing layer (see fig. 23). Even if it is not a hurdle when learning reconstruction models (used for analysis tasks with non sequential reconstructions), it is necessary to address this drawback. This could be done by adding a temporal constraint when learning the VAE *e.g* via the use of reccurent units [65]. Another difficulty to apply data assimilation schemes lies in the accurate estimation of prediction errors. Gaussian processes are promising tools for such tasks [66].

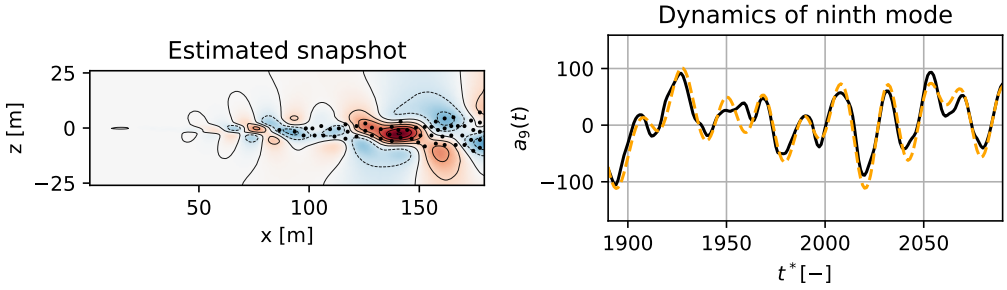
[2]We could then benefit from parallelization and dynamic allocation. When investigating the flow around an isolated building, we found that working with a 1391728×1600 snapshots matrix approximately requires 50 GiB of RAM. In such a case, estimation tools may be learned on a subdomain or a coarser mesh on which the validated solution is interpolated to.



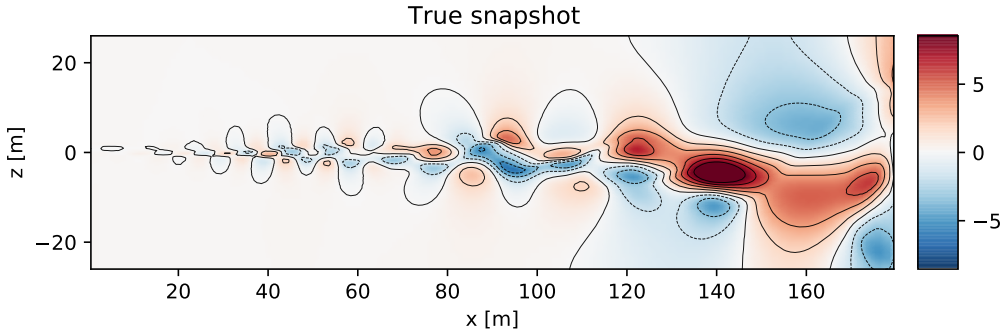
(a) Grid of sensors $p_{0.2} = 18$



(b) Grid of sensors $p_{0.4} = 42$

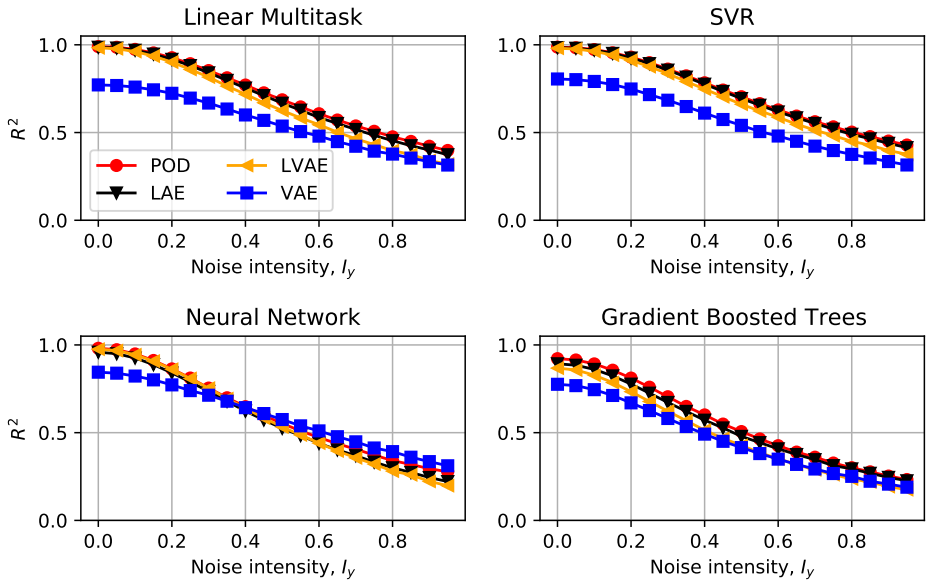


(c) Grid of sensors $p_{0.8} = 122$

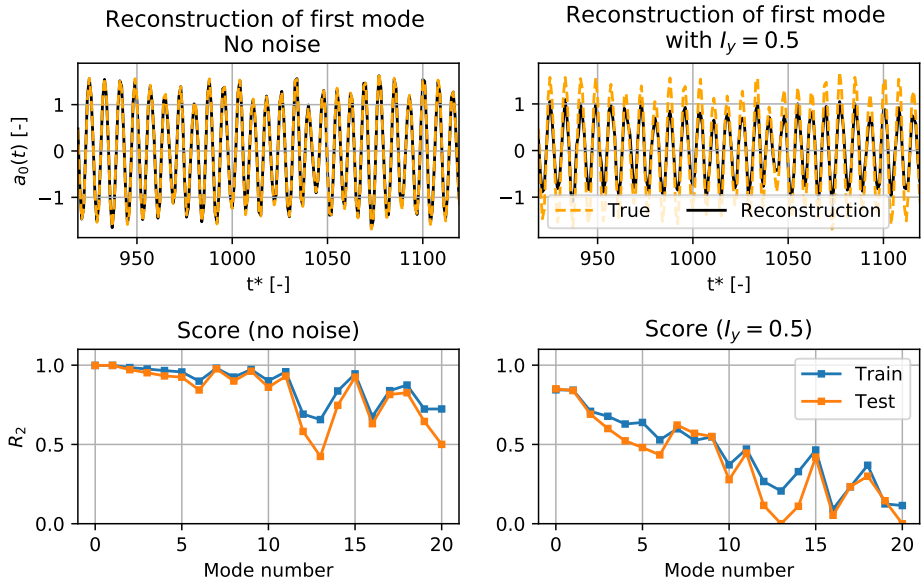


(d) Reference snapshot colored by the longitudinal fluctuating velocity.

Figure 21: Linear Multitask reconstruction of a mixing layer. The estimation is visualised in the physical space (one testing snapshot) and in the latent state (dynamics of the ninth POD mode). Increasing the density of sensors leads to a better estimate of the snapshot.



(a) Determination coefficient as a function of noise level in measurements. Illustrated on the mixing layer with $r_{0.80} = 11$ and $p_{0.80} = 122$.



(b) Linear Multitask reconstruction of the first POD mode for the 3D cylinder with $r_{0.80} = 21$ and $p_{0.8} = 372$.

Figure 22: Influence of noisy measurements in the reconstruction of testing latent dynamics.

6. Conclusion

This paper investigates the use of machine learning tools for fluid flow field estimation from limited measurements. First, the field is reduced using reduction techniques. Second, the reduced state is recovered from limited measurements. Finally, the estimated latent state is decoded to reconstruct the full velocity field. The strategy is applied to three canonical flows: a 2D cylinder at small Reynolds number, a spatial mixing layer and a 3D square cylinder at moderate Reynolds. For the reduction process, linear or nonlinear

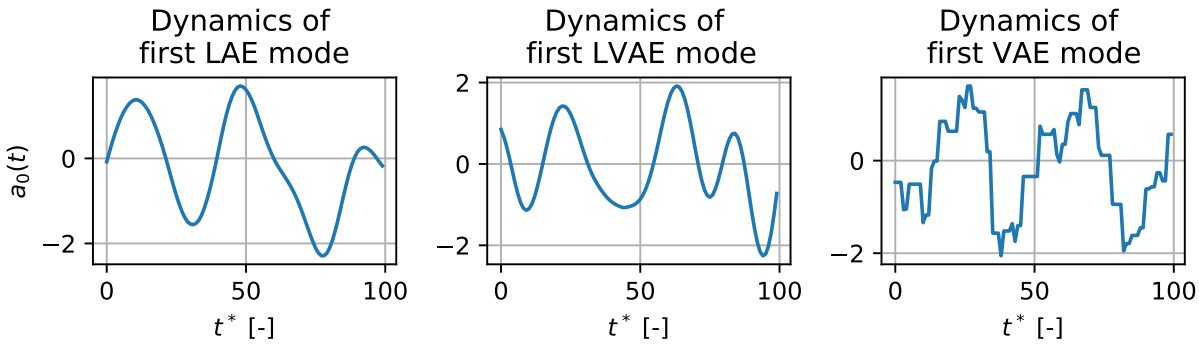


Figure 23: Comparison of latent state time series for the mixing layer. The dynamics of the VAE state shows some temporal irregularities. This may be an obstacle for real time estimation.

transformations can be used. Linear autoencoders encode and decode data with high fidelity but the method is not robust to noise applied to the latent state. Using a variational approach can increase the robustness to the detriment of the reduction mean square error. Besides, the use of nonlinearities in the reduction phase leads to a faster identification of dominant frequencies compared to a classical proper orthogonal decomposition. For the latent state estimation, cross-validation during the learning phase enables good generalization to test data for all methods. The choice of one reconstruction technique towards another therefore depends on initial specifications: data availability, interpretability, implementation cost and computational cost. All in all, results on the three canonical flow fields provide valuable informations for the development of new reconstruction techniques based on machine learning and their deployment on complex geometries that can be encoutered in industrial issues.

7. Acknowledgments

The authors wish to thank ONERA and Hauts-De-France region for their funding.

8. References

[1] K. T. Carlberg, A. Jameson, M. J. Kochenderfer, J. Morton, L. Peng, F. D. Witherden, Recovering missing cfd data for high-order discretizations using deep neural networks and dynamics learning, Journal of Computational Physics 395 (2019) 105–124.

[2] Z. Deng, Y. Chen, Y. Liu, K. C. Kim, Time-resolved turbulent velocity field reconstruction using a long short-term memory (lstm)-based artificial intelligence framework, Physics of Fluids 31 (7) (2019) 075108.

[3] J. L. Callaham, K. Maeda, S. L. Brunton, Robust flow reconstruction from limited measurements via sparse representation, Physical Review Fluids 4 (10) (2019) 103907.

[4] K. Manohar, B. W. Brunton, J. N. Kutz, S. L. Brunton, Data-driven sparse sensor placement for reconstruction: Demonstrating the benefits of exploiting known patterns, IEEE Control Systems Magazine 38 (3) (2018) 63–86.

[5] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics 52 (2020) 477–508.

[6] A. Arnault, J. Dandois, J.-M. Foucaut, Comparison of stochastic estimation methods with conditional events optimization for the reconstruction of the flow around a supercritical airfoil in transonic conditions, Computers & Fluids 136 (2016) 436–455.

[7] R. J. Adrian, Conditional eddies in isotropic turbulence, The Physics of Fluids 22 (11) (1979) 2065–2070.

[8] Y. Guezennec, Stochastic estimation of coherent structures in turbulent boundary layers, Physics of Fluids A: Fluid Dynamics 1 (6) (1989) 1054–1060.

[9] N. E. Murray, L. S. Ukeiley, Estimation of the flowfield from surface pressure measurements in an open cavity, AIAA journal 41 (5) (2003) 969–972.

[10] D. R. Cole, M. N. Glauser, Applications of stochastic estimation in the axisymmetric sudden expansion, Physics of Fluids 10 (11) (1998) 2941–2949.

[11] J. P. Bonnet, D. R. Cole, J. Delville, M. N. Glauser, L. S. Ukeiley, Stochastic estimation and proper orthogonal decomposition: complementary techniques for identifying structure, Experiments in

fluids 17 (5) (1994) 307–314.
[12] L. Perret, K. Blackman, E. Savory, Combining wind-tunnel and field measurements of street-canyon flow via stochastic estimation, Boundary-Layer Meteorology 161 (3) (2016) 491–517.
[13] N. B. Erichson, L. Mathelin, Z. Yao, S. L. Brunton, M. W. Mahoney, J. N. Kutz, Shallow learning for fluid flow reconstruction with limited sensors and limited data, arXiv preprint arXiv:1902.07358.
[14] J.-C. Loiseau, B. R. Noack, S. L. Brunton, Sparse reduced-order modeling: sensor-based dynamics to full-state estimation, arXiv preprint arXiv:1706.03531.
[15] S. Al Mamun, C. Lu, B. Jayaraman, Extreme learning machines as encoders for sparse reconstruction, Fluids 3 (4) (2018) 88.
[16] K. Taira, S. L. Brunton, S. T. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, L. S. Ukeiley, Modal analysis of fluid flows: An overview, AIAA Journal 55 (12) (2017) 4013–4041.
[17] R. Everson, L. Sirovich, Karhunen–loeve procedure for gappy data, JOSA A 12 (8) (1995) 1657–1664.
[18] F. Lusseyran, P. Gougat, A reconstruction method for the flow past an open cavity, Journal of Fluids Engineering 128 (2006) 531.
[19] L. Mathelin, K. Kasper, H. Abou-Kandil, Observable dictionary learning for high-dimensional statistical inference, Archives of Computational Methods in Engineering 25 (1) (2018) 103–120.
[20] J. Xu, K. Duraisamy, Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics, arXiv preprint arXiv:1912.11114.
[21] L. Sun, J.-X. Wang, Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data, arXiv preprint arXiv:2001.05542.
[22] V. Mons, J.-C. Chassaing, T. Gomez, P. Sagaut, Reconstruction of unsteady viscous flows using data assimilation schemes, Journal of Computational Physics 316 (2016) 255–280.
[23] B. R. Noack, K. Afanasiev, M. Morzyński, G. Tadmor, F. Thiele, A hierarchy of low-dimensional models for the transient and post-transient cylinder wake, Journal of Fluid Mechanics 497 (2003) 335–363.
[24] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the National Academy of Sciences (2016) 201517384.
[25] C. Colburn, J. Cessna, T. Bewley, State estimation in wall-bounded flow systems. part 3. the ensemble kalman filter, Journal of Fluid Mechanics 682 (2011) 289–303.
[26] T. Suzuki, Reduced-order kalman-filtered hybrid simulation combining particle tracking velocimetry and direct numerical simulation, Journal of Fluid Mechanics 709 (2012) 249–288.
[27] K. Taira, M. S. Hemati, S. L. Brunton, Y. Sun, K. Duraisamy, S. Bagheri, S. T. Dawson, C.-A. Yeh, Modal analysis of fluid flows: Applications and outlook, AIAA journal 58 (3) (2020) 998–1022.
[28] C. W. Rowley, S. T. Dawson, Model reduction for flow analysis and control, Annual Review of Fluid Mechanics 49 (2017) 387–417.
[29] A. T. Mohan, D. V. Gaitonde, A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks, arXiv preprint arXiv:1804.09269.
[30] J. N. Kutz, Deep learning in fluid dynamics, Journal of Fluid Mechanics 814 (2017) 1–4.
[31] J. Morton, F. D. Witherden, M. J. Kochenderfer, Parameter-conditioned sequential generative modeling of fluid flows, arXiv preprint arXiv:1912.06752.
[32] P. J. Schmid, Dynamic mode decomposition of numerical and experimental data, Journal of fluid mechanics 656 (2010) 5–28.
[33] B. Lusch, J. N. Kutz, S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nature communications 9 (1) (2018) 4950.
[34] E. Kaiser, B. R. Noack, L. Cordier, A. Spohn, M. Segond, M. Abel, G. Daviller, J. Östh, S. Krajnović, R. K. Niven, Cluster-based reduced-order modelling of a mixing layer, Journal of Fluid Mechanics 754 (2014) 365–414.
[35] A. Arnault, J. Dandois, J.-C. Monnier, J. Delva, J.-M. Foucaut, Analysis of the filtering effect of the stochastic estimation and accuracy improvement by sensor location optimization, Experiments in Fluids 57 (12) (2016) 185.
[36] B. Jayaraman, S. Al Mamun, C. Lu, Interplay of sensor quantity, placement and system dimension in pod-based sparse reconstruction of fluid flows, Fluids 4 (2) (2019) 109.
[37] B. Jayaraman, C. Lu, J. Whitman, G. Chowdhary, Sparse feature map-based markov models for nonlinear fluid flows, Computers & Fluids 191 (2019) 104252.
[38] K. Hornik, M. Stinchcombe, H. White, et al., Multilayer feedforward networks are universal approximators., Neural networks 2 (5) (1989) 359–366.
[39] S. Odaibo, Tutorial: Deriving the standard variational autoencoder (vae) loss function, arXiv preprint arXiv:1907.08956.
[40] S. Diamond, S. Boyd, CVXPY: A Python-embedded modeling language for convex optimization, Journal of Machine Learning Research 17 (83) (2016) 1–5.
[41] R. Li, X. Wang, L. Lei, Y. Song, $l\_\{21\}$-norm based loss function and regularization extreme learning

machine, IEEE Access 7 (2018) 6575–6586.
[42] H. Kuhn, A. Tucker, Nonlinear programming. proceedings of the 2nd berkeley symposium on mathematical statistics and probability, university of california.
[43] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and computing 14 (3) (2004) 199–222.
[44] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems 2 (4) (1989) 303–314.
[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (1) (2014) 1929–1958.
[46] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533–536.
[47] J. Friedman, T. Hastie, R. Tibshirani, et al., The elements of statistical learning, Vol. 1, Springer series in statistics New York, 2001.
[48] G. James, D. Witten, T. Hastie, R. Tibshirani, An introduction to statistical learning, Vol. 112, Springer, 2013.
[49] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, Classification and regression trees, CRC press, 1984.
[50] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.
[51] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, The Journal of Machine Learning Research 13 (1) (2012) 281–305.
[52] 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, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.
[53] F. Chollet, keras, https://github.com/fchollet/keras (2015).
[54] M. Braza, P. Chassaing, H. H. Minh, Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder, Journal of fluid mechanics 165 (1986) 79–130.
[55] A. Attili, F. Bisetti, Statistics and scaling of turbulence in a spatially developing mixing layer at Re_λ= 250, Physics of Fluids 24 (3) (2012) 035109.
[56] H. Bai, M. M. Alam, Dependence of square cylinder wake on reynolds number, Physics of Fluids 30 (1) (2018) 015102.
[57] T. M. Kodinariya, P. R. Makwana, Review on determining number of cluster in k-means clustering, International Journal 1 (6) (2013) 90–95.
[58] E. Plaut, From principal subspaces to principal components with linear autoencoders, arXiv preprint arXiv:1804.10253.
[59] P. Mohan, N. Fitzsimmons, R. D. Moser, Scaling of lyapunov exponents in homogeneous isotropic turbulence, Physical Review Fluids 2 (11) (2017) 114606.
[60] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics 807 (2016) 155–166.
[61] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.
[62] J. Ko, D. Lucor, P. Sagaut, Sensitivity of two-dimensional spatially developing mixing layers with respect to uncertain inflow conditions, Physics of Fluids 20 (7) (2008) 077102.
[63] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International conference on machine learning, 2013, pp. 1310–1318.
[64] K. Loh, P. S. Omrani, R. van der Linden, Deep learning and data assimilation for real-time production prediction in natural gas wells, arXiv preprint arXiv:1802.05141.
[65] P. Dubois, T. Gomez, L. Planckaert, L. Perret, Data-driven predictions of the lorenz system, Physica D: Nonlinear Phenomena 408 (2020) 132495.
[66] X. Qiu, E. Meyerson, R. Miikkulainen, Quantifying point-prediction uncertainty in neural networks via residual estimation with an i/o kernel, arXiv preprint arXiv:1906.00588.

Appendix A. Reconstruction results

Tables A.8, A.9 and A.10 give training and testing scores for the three investigated cases. Different reduction and reconstruction strategies are combined to estimate the full velocity field from limited measurements.

Method	Reduction	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$
DR	POD	[14.42 12.84 12.91]	[1.53 1.37 1.35]	[1.44 0.16 0.13]
	LAE	[14.19 12.96 13.06]	[1.53 1.37 1.35]	[0.73 0.16 0.13]
	LVAE	[15.03 12.71 12.70]	[1.73 1.49 1.46]	[10.32 9.73 1.32]
	VAE	**Not implemented**		
LM	POD	[12.88 12.88 12.88]	[1.44 1.34 1.34]	[0.28 0.12 0.12]
	LAE	[13.01 13.00 13.00]	[1.44 1.34 1.34]	[0.27 0.12 0.12]
	LVAE	[12.70 12.69 12.69]	[1.63 1.53 1.53]	[1.84 1.74 1.74]
	VAE	[25.09 25.09 25.09]	[16.46 16.52 16.52]	[3.61 3.47 3.49]
LS	POD	[12.88 12.88 12.88]	[1.44 1.34 1.34]	[0.28 0.12 0.12]
	LAE	[13.01 13.00 13.00]	[1.44 1.34 1.34]	[0.27 0.12 0.12]
	LVAE	[12.70 12.69 12.69]	[1.63 1.53 1.53]	[1.84 1.74 1.74]
	VAE	[25.09 25.09 25.09]	[16.46 16.52 16.53]	[3.61 3.46 3.46]
SVR	POD	[12.99 12.95 12.94]	[1.46 1.47 1.50]	[0.24 0.26 0.29]
	LAE	[13.11 13.07 13.08]	[1.45 1.47 1.50]	[0.21 0.20 0.20]
	LVAE	[12.89 12.87 12.87]	[1.82 1.81 1.91]	[2.08 2.15 2.25]
	VAE	[25.11 25.10 25.09]	[16.62 16.95 16.86]	[3.55 3.67 3.70]
NN	POD	[12.88 12.89 12.86]	[1.38 1.65 1.34]	[0.43 0.13 0.33]
	LAE	[13.02 13.02 13.00]	[1.75 1.42 1.36]	[0.31 0.13 0.13]
	LVAE	[12.68 12.87 12.72]	[2.13 1.54 1.54]	[1.76 1.76 1.79]
	VAE	[25.15 25.09 25.08]	[16.62 16.67 16.72]	[3.77 4.48 3.73]
GB	POD	[12.89 12.88 12.89]	[1.44 1.41 1.36]	[0.25 0.2 0.15]
	LAE	[13.03 13.02 13.01]	[1.44 1.42 1.37]	[0.25 0.17 0.18]
	LVAE	[12.75 12.73 12.71]	[1.71 1.60 1.73]	[1.93 1.92 1.93]
	VAE	[25.10 25.10 25.10]	[17.42 16.98 16.73]	[3.95 3.83 3.86]

(a) Training errors.

Method	Reduction	$r_{0.4}$	$r_{0.8}$	$r_{0.99}$
DR	POD	[14.08 12.56 12.63]	[1.52 1.35 1.33]	[1.55 0.16 0.13]
	LAE	[13.81 12.64 12.74]	[1.52 1.35 1.33]	[0.78 0.16 0.13]
	LVAE	[14.78 12.54 12.53]	[1.73 1.46 1.44]	[10.06 9.66 1.29]
	VAE	**Not implemented**		
LM	POD	[12.6 12.6 12.6]	[1.44 1.32 1.32]	[0.30 0.12 0.12]
	LAE	[12.69 12.68 12.68]	[1.44 1.32 1.32]	[0.29 0.12 0.12]
	LVAE	[12.53 12.51 12.51]	[1.61 1.50 1.50]	[1.82 1.72 1.72]
	VAE	[24.79 24.79 24.79]	[16.13 16.25 16.22]	[3.49 3.45 3.48]
LS	POD	[12.6 12.6 12.6]	[1.44 1.32 1.32]	[0.30 0.12 0.12]
	LAE	[12.69 12.68 12.68]	[1.44 1.32 1.32]	[0.29 0.12 0.12]
	LVAE	[12.53 12.51 12.51]	[1.61 1.5 1.5]	[1.82 1.72 1.72]
	VAE	[24.79 24.79 24.79]	[16.13 16.23 16.24]	[3.49 3.45 3.42]
SVR	POD	[12.71 12.68 12.65]	[1.44 1.45 1.48]	[0.25 0.26 0.29]
	LAE	[12.79 12.75 12.75]	[1.43 1.45 1.47]	[0.21 0.2 0.21]
	LVAE	[12.72 12.71 12.69]	[1.80 1.80 1.89]	[2.06 2.12 2.22]
	VAE	[24.80 24.80 24.79]	[16.29 16.65 16.56]	[3.51 3.65 3.67]
NN	POD	[12.60 12.61 12.58]	[1.36 1.63 1.32]	[0.37 0.14 0.35]
	LAE	[12.69 12.71 12.68]	[1.73 1.40 1.34]	[0.30 0.14 0.13]
	LVAE	[12.52 12.71 12.55]	[2.13 1.53 1.52]	[1.75 1.74 1.78]
	VAE	[24.85 24.79 24.78]	[16.32 16.36 16.42]	[3.76 4.51 3.69]
GB	POD	[12.61 12.60 12.62]	[1.55 1.44 1.37]	[0.40 0.26 0.19]
	LAE	[12.70 12.73 12.68]	[1.49 1.42 1.37]	[0.32 0.19 0.2]
	LVAE	[12.59 12.57 12.55]	[1.8 1.61 1.72]	[1.96 1.96 1.92]
	VAE	[24.80 24.79 24.80]	[17.18 16.7 16.42]	[4.05 3.80 3.78]

(b) Testing errors.

Table A.8: NMSE values for the 2D cylinder. Values are in percentage and given as [$p_{0.2}$, $p_{0.4}$, $p_{0.8}$] errors.

Method	Reduction	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[26.76 24.98 23.14]	[26.44 22.6 10.78]	[112. 21.61 7.26]
	LAE	[27.3 25.1 23.17]	[31.64 26.28 11.28]	[108.03 27.99 9.86]
	LVAE	[28.19 25.32 23.18]	[30.67 31.12 10.94]	[123.56 22.29 8.00]
	VAE	**Not implemented**		
LM	POD	[25.10 23.34 22.80]	[21.11 17.61 9.52]	[19.96 15.32 5.40]
	LAE	[25.13 23.36 22.84]	[21.37 17.77 9.89]	[20.26 15.77 5.8 0]
	LVAE	[25.16 23.38 22.85]	[21.18 17.70 9.63]	[20.15 15.59 5.62]
	VAE	[29.72 29.32 29.28]	[27.66 25.97 22.17]	[27.05 24.89 19.77]
LS	POD	[25.08 23.34 22.80]	[21.17 17.28 9.52]	[20.27 15.39 5.41]
	LAE	[25.1 23.36 22.84]	[21.5 17.61 9.88]	[20.39 15.81 6.01]
	LVAE	[25.12 23.38 22.85]	[21.19 17.45 9.67]	[20.26 15.54 5.88]
	VAE	[29.72 29.33 29.28]	[27.55 25.92 22.15]	[26.96 24.65 19.71]
SVR	POD	[24.59 23.21 22.87]	[19.94 16.57 9.32]	[18.39 14.07 3.20]
	LAE	[24.63 23.26 22.91]	[20.63 16.38 9.67]	[19.19 14.19 4.39]
	LVAE	[24.89 23.26 22.92]	[20.5 16.59 9.46]	[18.63 14.00 4.55]
	VAE	[29.58 29.42 29.53]	[27.16 25.52 22.55]	[26.22 24.05 19.14]
NN	POD	[25.16 23.41 22.95]	[21.3 17.82 9.66]	[21.61 16.08 6.58]
	LAE	[24.67 23.68 22.98]	[20.88 17.76 10.21]	[20.52 15.75 6.38]
	LVAE	[24.9 23.31 22.93]	[21.76 18.20 9.82]	[19.54 15.03 6.11]
	VAE	[29.72 29.63 29.63]	[27.72 26.10 23.72]	[27.77 25.01 19.33]
GB	POD	[23.8 22.95 22.82]	[16.84 12.10 9.09]	[13.61 7.23 2.88]
	LAE	[24.01 22.95 22.92]	[17.02 11.36 9.48]	[14.06 5.89 3.42]
	LVAE	[23.65 23.02 22.93]	[15.57 12.12 9.32]	[13.41 7.08 3.46]
	VAE	[29.65 29.62 29.66]	[25.46 23.94 22.97]	[23.57 20.46 19.12]

(a) Training errors.

Method	Reduction	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[28.31 26.18 23.89]	[29.29 25.06 11.27]	[125.56 23.60 8.60]
	LAE	[28.91 26.29 23.94]	[33.36 27.56 11.58]	[110.76 29.68 11.07]
	LVAE	[29.67 26.47 23.97]	[32.17 33.40 11.40]	[126.37 24.90 9.20]
	VAE	**Not implemented**		
LM	POD	[25.90 24.22 23.51]	[23.13 20.05 10.26]	[22.15 18.10 6.30]
	LAE	[26.00 24.28 23.58]	[23.41 20.26 10.64]	[22.43 18.49 6.76]
	LVAE	[26.04 24.33 23.62]	[23.26 20.16 10.44]	[22.35 18.39 6.58]
	VAE	[30.47 30.19 30.17]	[28.80 27.41 22.96]	[28.68 26.93 20.55]
LS	POD	[25.86 24.22 23.51]	[23.10 19.76 10.25]	[22.23 18.06 6.31]
	LAE	[25.95 24.28 23.58]	[23.63 20.07 10.64]	[22.73 18.61 6.97]
	LVAE	[25.96 24.33 23.62]	[23.24 20.06 10.48]	[22.54 18.58 6.77]
	VAE	[30.46 30.20 30.18]	[28.73 27.44 22.98]	[28.62 26.90 20.51]
SVR	POD	[25.68 24.13 23.63]	[22.92 19.76 10.39]	[21.72 17.80 5.91]
	LAE	[25.88 24.18 23.71]	[23.21 19.75 10.73]	[22.28 18.15 6.48]
	LVAE	[25.87 24.25 23.73]	[23.12 19.84 10.51]	[22.07 17.81 6.35]
	VAE	[30.48 30.32 30.47]	[28.85 27.38 23.50]	[28.42 26.78 20.34]
NN	POD	[25.87 24.45 23.95]	[22.91 19.79 10.53]	[22.99 18.11 7.87]
	LAE	[25.94 24.57 23.78]	[23.22 19.99 11.32]	[22.11 18.12 9.46]
	LVAE	[25.86 24.35 23.80]	[23.29 20.16 10.59]	[21.70 17.50 6.91]
	VAE	[30.52 30.51 30.55]	[28.72 27.37 24.64]	[28.91 26.58 20.71]
GB	POD	[26.24 24.70 23.97]	[22.98 19.63 12.09]	[21.89 17.93 8.68]
	LAE	[26.52 24.75 24.18]	[23.51 20.07 13.13]	[22.17 18.09 10.61]
	LVAE	[26.52 25.11 24.41]	[23.82 20.27 13.09]	[22.31 18.55 10.11]
	VAE	[30.82 30.71 30.64]	[28.9 27.27 23.82]	[28.10 25.82 20.99]

(b) Testing errors.

Table A.9: NMSE values for the mixing layer. Values are in percentage and given as [$p_{0.2}$, $p_{0.4}$, $p_{0.8}$] errors.

Method	Reduction	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[47.36 46.59 45.45]	[48.84 42.86 33.76]	[62.91 74.23 27.44]
	LAE	[47.81 47.07 46.]	[52.07 45.57 34.67]	[78.36 86.67 29.95]
	LVAE	[47.71 47.07 46.02]	[48.95 44.12 34.48]	[70.5 97.72 43.51]
	VAE	**Not implemented**		
LM	POD	[46.8 45.85 45.17]	[40.47 36.91 31.16]	[38.39 31.78 16.41]
	LAE	[47.35 46.42 45.74]	[41.6 38.1 32.15]	[38.92 32.82 18.53]
	LVAE	[47.33 46.41 45.79]	[42.19 38.95 33.63]	[41.5 37.5 28.99]
	VAE	[49.55 48.98 48.45]	[44.8 42.9 40.44]	[42.28 38.97 33.66]
LS	POD	[46.82 45.84 45.19]	[40.57 36.98 31.26]	[38.66 32.48 17.3]
	LAE	[47.33 46.42 45.75]	[41.58 38.19 32.28]	[38.98 32.99 19.16]
	LVAE	[47.31 46.42 45.81]	[42.21 38.98 33.77]	[41.46 37.3 29.05]
	VAE	[49.55 49. 48.41]	[44.71 42.79 40.43]	[42.3 39.04 33.9]
SVR	POD	[45.63 45.43 45.26]	[33.7 31.25 29.56]	[28.31 19.05 10.61]
	LAE	[46.24 45.98 45.75]	[34.92 32.76 30.73]	[28.61 21.58 13.17]
	LVAE	[46.16 45.98 45.84]	[36.21 34.92 32.47]	[33.93 30.21 26.24]
	VAE	[50.04 49.99 50.96]	[42.12 41.31 40.41]	[36.54 34.43 32.58]
NN	POD	[45.78 45.36 45.91]	[38.14 35.41 31.35]	[39.22 32.03 18.52]
	LAE	[46.83 46.51 45.72]	[42. 38.45 31.52]	[32.38 31.06 21.42]
	LVAE	[47.16 45.99 45.84]	[39.21 38.23 33.5]	[40.85 36.33 29.98]
	VAE	[53.04 50.75 51.54]	[43.96 42.47 41.32]	[38.96 37.6 33.85]
GB	POD	[45.48 45.45 45.23]	[33.12 31.62 30.37]	[26.47 22.6 17.71]
	LAE	[46.3 45.88 45.77]	[33.58 32.42 31.35]	[24.71 20.31 16.89]
	LVAE	[46.3 46.11 45.93]	[35.32 34.28 33.39]	[31.44 29.17 27.57]
	VAE	[50.68 51.29 51.08]	[41.91 41.36 41.08]	[35.54 34.29 33.31]

(a) Training errors.

Method	Reduction	$r_{0.4}$	$r_{0.8}$	$r_{0.95}$
DR	POD	[45.12 44.43 43.56]	[47.98 42.36 35.2]	[65.96 96.11 30.79]
	LAE	[45.97 45.18 44.31]	[51.51 45.75 36.56]	[80.78 107.44 34.43]
	LVAE	[45.81 45.21 44.35]	[48.85 44.81 36.24]	[77.93 124.96 58.26]
	VAE	**Not implemented**		
LM	POD	[44.77 43.91 43.38]	[40.81 38.28 33.51]	[39.83 36. 25.51]
	LAE	[45.5 44.65 44.14]	[41.98 39.68 34.53]	[40.32 36.77 26.47]
	LVAE	[45.55 44.72 44.21]	[42.39 40.19 35.57]	[42.22 39.51 32.61]
	VAE	[48.29 47.89 47.2]	[44.27 42.71 40.65]	[42.59 40.21 35.59]
LS	POD	[44.8 43.93 43.39]	[40.9 38.48 33.68]	[40.1 36.49 25.69]
	LAE	[45.49 44.66 44.15]	[42.06 39.85 34.71]	[40.48 37.12 26.91]
	LVAE	[45.58 44.74 44.23]	[42.41 40.34 35.75]	[42.21 39.5 32.68]
	VAE	[48.31 47.9 47.13]	[44.23 42.63 40.62]	[42.54 40.23 35.76]
SVR	POD	[44.28 43.8 43.94]	[38.91 37.22 33.81]	[37.6 34.6 24.98]
	LAE	[45.03 44.56 44.14]	[40.09 38.45 34.51]	[38.63 35.86 25.93]
	LVAE	[45.07 44.66 44.28]	[40.65 39.09 35.35]	[39.89 37.86 32.2]
	VAE	[49.51 48.6 49.01]	[43.19 42.2 40.65]	[40.59 38.93 35.68]
NN	POD	[44.31 43.92 44.29]	[39.7 38.64 34.74]	[41.19 37.61 27.94]
	LAE	[45.17 44.87 44.21]	[42.53 40.72 35.42]	[38.94 38.63 29.77]
	LVAE	[45.8 44.69 44.27]	[41.17 39.98 36.18]	[42.14 39.84 33.45]
	VAE	[52.29 49.5 49.95]	[43.7 42.73 41.64]	[41.33 40.57 36.77]
GB	POD	[44.47 44.22 43.88]	[39.42 38.1 35.31]	[38.74 36.84 31.81]
	LAE	[45.22 44.82 44.48]	[40.88 39.3 36.29]	[40.13 38.12 33.1]
	LVAE	[45.38 45.09 44.63]	[41.3 40.02 37.47]	[40.23 38.58 35.33]
	VAE	[49.65 49.66 49.49]	[43.35 42.44 41.35]	[40.96 39.63 37.02]

(b) Testing errors.

Table A.10: NMSE values for the 3D cylinder. Values are in percentage and given as [$p_{0.2}$, $p_{0.4}$, $p_{0.8}$] errors.