Piecewise Dynamic Mode Decomposition for Fluid Flow Simulations

Matheus B. Seidel[1], Gabriel F. Barros[1], Renato N. Elias[1], Alvaro L. G. A. Coutinho[1], Michèle S. Pfeil[1]

[1]*Programa de Engenharia Civil - COPPE, Universidade Federal do Rio de Janeiro*
Rio de Janeiro, 21941-853, RJ, Brazil
matheus.seidel@coc.ufrj.br, gabriel.barros@coc.ufrj.br, rnelias@coc.ufrj.br, alvaro@coc.ufrj.br, mpfeil@coc.ufrj.br

Abstract. The computational simulation of fluid flows over structures is still a major fluid mechanics research area. Because of the multiscale nature of many applications and the consequent large amount of data, these simulations are computationally expensive but can benefit from modern Reduced Order Models and data-driven methods. In this work, Dynamic Mode Decomposition (DMD) and its recent variation, Piecewise DMD (pDMD), are presented and compared in terms of dynamic modes extracted from the data, accuracy in reconstructing an approximation for the original dataset as a reduced order model and, most importantly, computational cost. The pDMD method is shown to be a variation of the traditional DMD that aims to improve and overcome some of the caveats of the standard version. This variation consists of decomposing the entire data into smaller datasets and applying a linear mapping independently on each subset instead of calculating a global linear fitting. The preliminary results presented in this work show how DMD can capture the dynamics and accurately reconstruct the simulation data and how pDMD can provide more accurate results when traditional DMD reaches its limitations, capture the specific dynamics of different stages of transient flows, and reduce the computational cost by 90% for a two-dimensional flow over a cylinder when compared to standard DMD.

Keywords: Dynamic Mode Decomposition, Reduced order model, Fluid flow simulation.

1 Introduction

The computational simulation of fluid flow over structures is a major research area in fluid mechanics due to its large amount of data and demanded computational power. Even with recent developments in computer processing, Direct Numerical Simulation (DNS) for solving fluid flows is still not feasible for most practical engineering problems [1] [2] [3]. On the other hand, recent studies in machine learning and data-driven methods applied to Computational Fluid Dynamics (CFD) can take advantage of the high-order nature of the problem and develop Reduced Order Models (ROM) to simulate fluid flows and solve real engineering problems using less time and computational resources. One way to develop a ROM is by using Dynamic Mode Decomposition (DMD), a non-intrusive data-driven method first developed by Schmid [4]. DMD does not require physical information about the system and may be used for future state predictions, computationally cheap parametric simulations, and qualitative dynamic analysis of fluid flows. A variation of DMD, recently proposed by Alla et al. [5], is called Piecewise DMD (pDMD) and aims to improve and overcome some of the caveats of the traditional method. Essentially, pDMD decomposes the entire data into smaller datasets and applies a linear mapping independently on each subset instead of calculating a global linear fitting. It is a simple and elegant idea that is, according to its authors, based on the "divide and conquer" approach well known in the numerical analysis literature, and that can significantly reduce the computational cost of DMD. This work aims to investigate, using simulation data of a two-dimensional flow around a cylinder, how DMD can capture the dynamics and accurately reconstruct the simulation data and how pDMD can provide more accurate results, capture the specific dynamics of different stages of transient flows, and reduce the computational cost when compared to standard DMD. The remainder of this paper is organized as follows: Section 2 presents a brief formulation of Dynamic Mode Decomposition; Section 3 introduces the piecewise variation of DMD; Section 4 shows the preliminary results of our research, where the benchmark of a two-dimensional cylinder at Reynolds number 100 is used to test DMD and pDMD by extracting the dynamic modes, reconstructing the original data and investigating the computational cost; and Section 5 concludes this work.

2 Dynamic Mode Decomposition

DMD starts with collecting the data as snapshots and organizing them into matrices. Each one of the m snapshots must be organized as a column vector x_k, and the matrix X will represent the snapshot sequence. A matrix X' must be similarly defined but shifted one time step forward. Matrices X and X' are shown in eq. (1) and are considered "tall and skinny" for CFD applications since the number of columns is equal to the number of snapshots and the number of lines is equal to the number of mesh nodes.

$$X = \begin{pmatrix} | & | & | & | \\ x_1 & x_2 & ... & x_{m-1} \\ | & | & | & | \end{pmatrix}, \qquad X' = \begin{pmatrix} | & | & | & | \\ x_2 & x_3 & ... & x_m \\ | & | & | & | \end{pmatrix}. \tag{1}$$

The idea of DMD is to find a linear mapping A such that:

$$X' \approx AX \Rightarrow x_{k+1} = Ax_k. \tag{2}$$

The matrix A is a linear approximation that is chosen to minimize $\parallel x_{k+1} - Ax_k \parallel_2$ and can be calculated by:

$$A = X'X^{\dagger} \tag{3}$$

where $X^{\dagger}$ is the Moore-Penrose pseudoinverse of X. Because A is usually a large matrix, its eigenvectors and eigenvalues are calculated by taking the Singular Value Decomposition (SVD) of X:

$$X = U\Sigma V^* \tag{4}$$

where * denotes the conjugate transpose, U is the left singular vector, Σ is the singular values matrix and V^* is the right singular matrix. Substituting eq. (4) in eq. (2):

$$X' = AU\Sigma V^*. \tag{5}$$

The dominant coherent patterns are the columns of the U matrix, which are hierarchically organized from most to least important to capture the variance of X. Multiplying eq. (5) by U^* on the left and by $V\Sigma^{-1}$ on the right of each term, we get:

$$U^*X'V\Sigma^{-1} = U^*AU = \tilde{A} \tag{6}$$

where $\tilde{A}$ is the projection of the A matrix into the singular vectors of U. Because of how the columns of U are organized, from most important to least important, a good approximation can be achieved only using the first r columns of U, and therefore $\tilde{A}$ can be much smaller than A and still provide an accurate approximation. It can be proved that the eigenvalues of $\tilde{A}$ are the same of the A matrix [6], such that:

$$\tilde{A}W = W\Lambda \tag{7}$$

where Λ stores the eigenvalues of A and its eigenvectors are Φ given by:

$$\Phi = X'V\Sigma^{-1}W. \tag{8}$$

The eigenvectors are called modes of the system because they are spatial-temporal coherent mode shapes. These modes allow the data reconstruction and future state predictions:

$$\tilde{X}(k\Delta t) = \Phi\Lambda^t b_o \tag{9}$$

where $\tilde{X}$ is snapshot being reconstructed or predicted, k is an integer, Δt is the time step size, t is the time, b_o is the initial condition, and Φ and Λ are the eigenvectors and eigenvalues of A, respectively. Notice that DMD is a very simple method and consists of a best fit linear model that can be used for nonlinear and linear, dynamical systems. Like most scientific machine learning algorithms, it is essentially a function that maps inputs to outputs in the form $y = f(x;\theta)$, similarly as in eq. (2), where θ represents the system parameters that can be used.

3 Piecewise DMD

Piecewise DMD (pDMD) was recently proposed by Alla et al. [5] and consists of a variation of the traditional DMD method that aims to improve and overcome some of the DMD caveats. The authors present examples of simulations where DMD fails to capture the dynamics, make future-state predictions, and even reconstruct the data in which it was trained, such as the FitzHugh-Nagumo model, λ - ω system, Turing instability, and Turing-Hopf instability. Instead of calculating a "global" linear fitting (as in eq. (2)) over the entire snapshots matrix, pDMD decomposes the whole time interval $[0, t_f]$ into N smaller datasets and applies a linear mapping independently on each one of these subsets. Consider the simulation data as a snapshot matrix S. Traditional DMD would construct the X and X' matrices, as in eq. (1), and calculate the linear mapping A such that $X' \approx AX$. Piecewise DMD performs the decomposition $S^N = \cup_{i=1}^{N} S_i$, where S_i is the submatrix of ν columns of S defined by $S_i = [S_{:,(i-1)\nu+1}, ..., S_{:,i\nu}] \in \mathbb{R}^{2n \times \nu}$ for $i = 1, ..., N$. Notice that S_i consists of the snapshot matrix for time interval $[t_{(i-1)\nu}, t_{i\nu-1}]$. The point of pDMD is to apply a similar linear mapping $X_i' \approx A_i X_i$ for each S_i using rank r_i, which can be defined *a priori* for all the data or calculated for each subset (in this work, the same r is used for all subsets). Our version of pDMD is summarized in Algorithm 1, and Fig. 1 shows the schematic of the method. The original algorithm proposed by Alla et al. [5] calculates a relative error for each subset and establishes a threshold above which N must be increased and the process restarted, whereas our code uses a constant N defined *a priori*. No computational cost analysis for the author's algorithm has yet been performed.

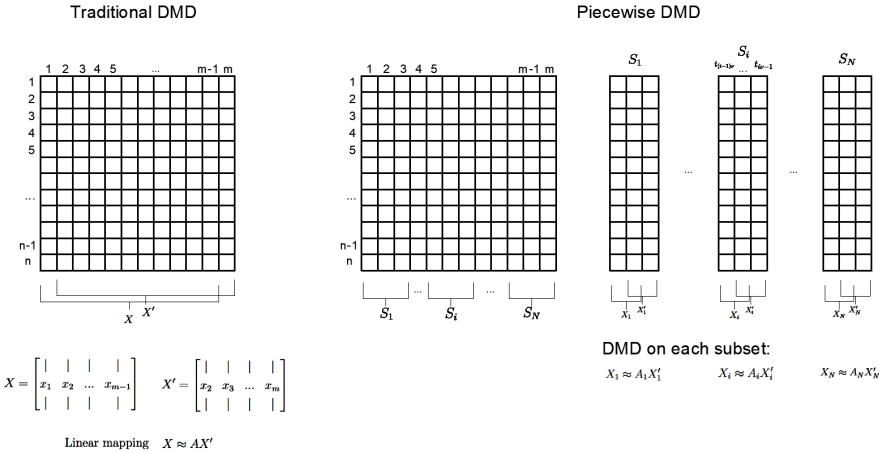


Figure 1. Schematic of piecewise DMD

Algorithm 1 Piecewise DMD (pDMD)

INPUT: Dataset as snapshots $\mathbf{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$
OUTPUT: Piecewise reconstruction $\tilde{S^N} \approx S$ in $[0, t_f]$
1: Choose number N of subsets
2: Choose rank r used for all subsets
3: Split the dataset $S^N = \cup_{i=1}^{N} S_i$
4: for $i = 1, \ldots, N$ do
5: Set X_i and X_i'
6: Compute the SVD $X_i = U_i \Sigma_i V_i^*$
7: $X_i' \approx A_i X_i \Rightarrow X_i' = A_i U_i \Sigma_i V_i^*$
8: Apply truncation with rank r: $U_i^* X_i' V_i \Sigma_i^{-1} = U_i^* A_i U_i = \tilde{A}_i$
9: Compute eigenvalues $\tilde{A}_i W_i = W_i \Lambda_i$
10: Compute eigenvectors by $\Phi_i = X_i' V_i \Sigma_i^{-1} W_i$
11: Approximately reconstruct the dataset $\tilde{S}_i$ as in eq. (9)
12: Make $\tilde{S^N} = \cup_{i=1}^{N} \tilde{S}_i$
13: Calculate the relative error ε in Frobenius norm between the original data S and the pDMD approximate reconstruction $\tilde{S}$.

4 Two-dimensional flow around a cylinder with $Re = 100$

The simulation of incompressible flow with a Reynolds number equal to 100 over a two-dimensional cylinder is performed using the FEniCS Project software. FEniCS is an open-source code with a high-level Python interface that solves partial differential equations using the finite element method. The geometry of the considered problem is shown in Fig. 2.

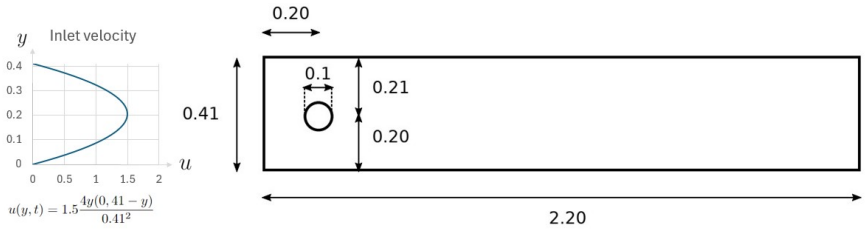


Figure 2. Two-dimensional fluid domain in the flow

The boundary conditions are the following: the side walls and surface of the cylinder are defined with no-slip condition, the left face is defined with velocity inlet as shown in Fig. 2, and the right face is defined as zero relative pressure output. The initial condition of the fluid is uniform, with null velocity over the entire domain. The mesh is generated using FEniCS and is presented in Fig. 3. The flow simulation is performed from $t = 0.001s$ to $t = 5.0s$ with time step = 0.001, and all 5000 snapshots are saved.

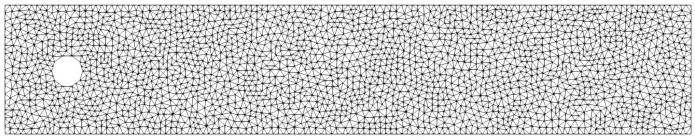
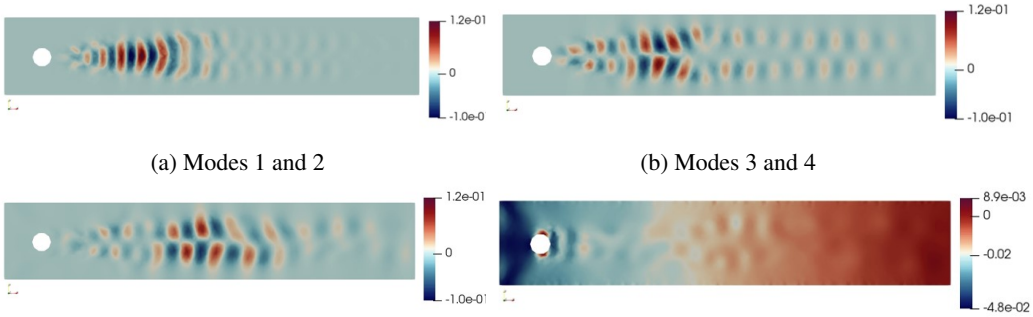


Figure 3. Finite element mesh with 2,445 nodes and 4,584 triangular elements

4.1 DMD Modes

After generating the simulation results, the PyDMD library [7] [8] is used to perform the Dynamic Mode Decomposition over 5000 pressure snapshots, which are read using the h5py library[1]. The code developed uses the traditional SVD (default in PyDMD), $r = 40$, and exact DMD. The mesh is read using the meshio library[2], and the results are exported as .vtk files. Figure 4 shows the dynamic modes obtained for the pressure data.



(a) Modes 1 and 2

(b) Modes 3 and 4

(c) Modes 5 and 6

(d) Modes 7 and 8

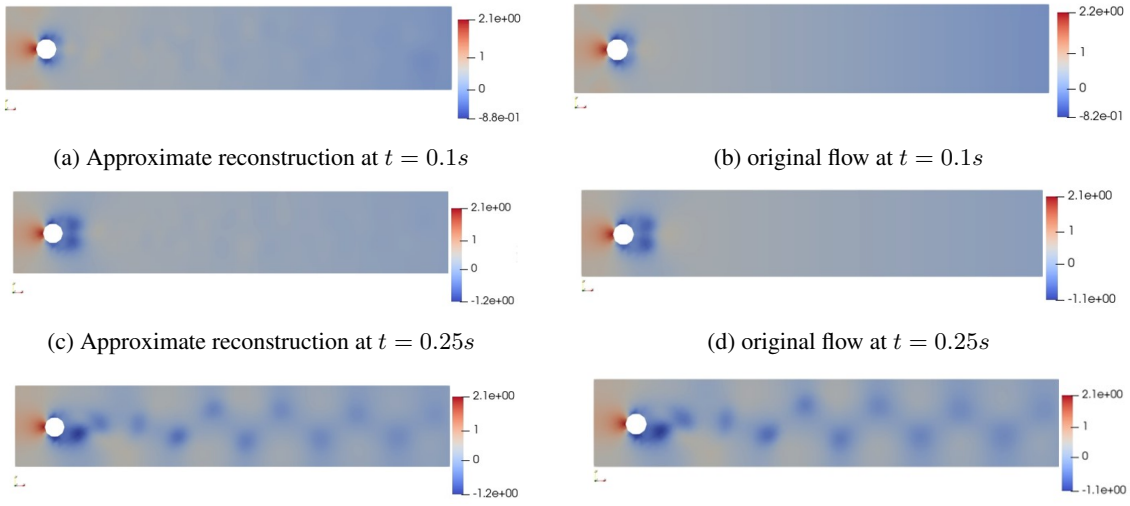
Figure 4. Dynamic modes for pressure data

The dynamic modes obtained show how DMD is able to capture the vortex street formed on the wake of the cylinder. These modes enable the reconstruction of the data as well as future state prediction.

[1] Available in https://h5py.org
[2] Available in https://pypi.org/project/meshio/1.2.0

4.2 Approximate reconstruction of the data

The dynamic modes previously shown are used in PyDMD to approximately reconstruct the flow simulation. Figure 5 shows the comparison between the original flow and the approximate reconstruction of the pressure field. In the initial phase of the flow, the pressure fields are similar, but the approximation shows residues of the vortex street that can be seen as fluctuations in the pressure field downwind of the cylinder. The same pattern is seen at $t = 0.25s$, where the two leeward vortices are still in approximate equilibrium, but the pressure field shows some fluctuations on the wake.



(a) Approximate reconstruction at $t = 0.1s$

(b) original flow at $t = 0.1s$

(c) Approximate reconstruction at $t = 0.25s$

(d) original flow at $t = 0.25s$

(e) Approximate reconstruction at $t = 5.0s$

(f) original flow at $t = 5.0s$

Figure 5. Approximate reconstruction and original data of the flow pressure field

Note that the approximation is visually indistinguishable when compared to the original data when the wake is fully developed. The von Karman vortex street is clearly formed in the wake of the cylinder, and no spurious fluctuations can be detected. The relative error is computed to assess the approximation's accuracy. The relative error ε_p in Frobenius norm between the original data S and the DMD approximate reconstruction $\tilde{S}$ is given by:

$$\varepsilon_p(\tilde{S}) = \frac{||S - \tilde{S}||_F}{||S||_F}. \tag{10}$$

Table 1 shows the error for the entire pressure dataset and for the specific snapshots shown previously. Figure 6 shows the relative error calculated for each node of the domain as given by eq. (10).

Table 1. Relative error for the pressure approximation

Data	ε_p
$[0, t_f]$	0.10403
$t = 0.10s$	0.07545
$t = 0.25s$	0.09479
$t = 5.00s$	0.09053

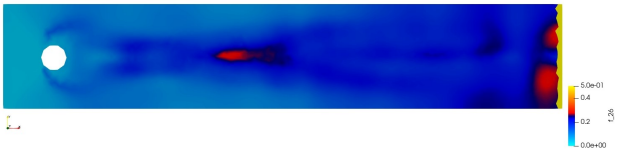
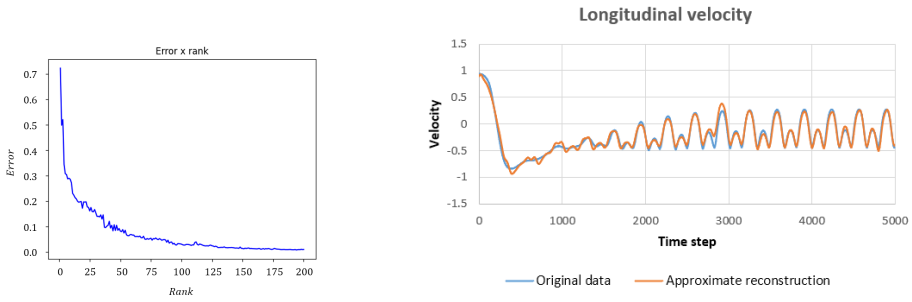


Figure 6. Relative error for the pressure approximation calculated for each node of the domain

The error for the entire reconstructed data is 0.10 when rank $r = 40$ is used. Increasing the rank captures more of the dynamics, leading to reduced errors. Figure 7a shows how the relative error is reduced exponentially

as the SVD rank increases. The error assessment also uses the longitudinal velocity at a given point one diameter leeward of the cylinder. Figure 7b presents the time history of the velocity for the original data and approximate reconstruction with rank 40. The graph shows an accurate reconstruction of the data and calculating the velocity approximation error ε_v using eq. (10) yields $\varepsilon_v = 0.03$.



(a) (b)

Figure 7. (a) Relative error and rank of the SVD. (b) Time history of the longitudinal velocity one diameter leeward of the cylinder for the original data and approximate reconstruction with $r = 40$.

4.3 Piecewise DMD

The Piecewise DMD code used in this work is also based on PyDMD, where multiple dynamic mode decompositions are performed, once for each subset. The number N of subsets in which the snapshot matrix is decomposed varies, and the rank is kept constant and defined *a priori* for each test case. For each N and rank, it is calculated the relative error ε_p for the approximate reconstruction $\tilde{S}^N$.

Piecewise DMD is applied to the pressure data with N varying from 1 to 500 and for four fixed ranks: 10, 20, 40, and 100. For each rank and value of N, the relative error is calculated by eq. (10). Figure 8 presents the error as a function of N for each rank.

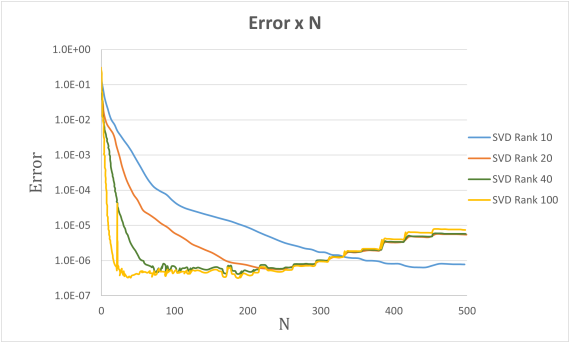


Figure 8. Error $\times$ number of subsets

The error decreases with the piecewise approach, and the graphs show a relevant inversely proportional trend between the relative error and the number of subsets. Except for the case with rank 10, the error starts increasing approximately for $N \geqslant 250$. Nonetheless, the final error for $N = 500$ in all cases is of the order of 10^{-6}, which can be considered very low. When the number of subsets is $N = 1$, pDMD is reduced to the traditional DMD method. As expected in this case, the relative error presented in Table 1 is exactly equal to the first point in the graph of Fig. 8 for SVD rank 40, where $\varepsilon_p = 0.104$.

4.4 Computational cost analysis

The computational cost of the pDMD analysis of the previous section is assessed by measuring the elapsed time when running the code with SVD rank 40 for a varying number of subsets. Figure 9 shows that using pDMD significantly increases the method's efficiency up to $N = 40$. Notice that, once again, traditional DMD is equal to pDMD with $N = 1$ and, therefore, has the computational cost of the first point in the graph, where $t \approx 1000s$.

Using pDMD reduces the elapsed time up to around 90%, taking $t \approx 98s$ for $N \geq 40$. Further increasing the number of subsets beyond 40 shows no decrease in computational cost for this case.

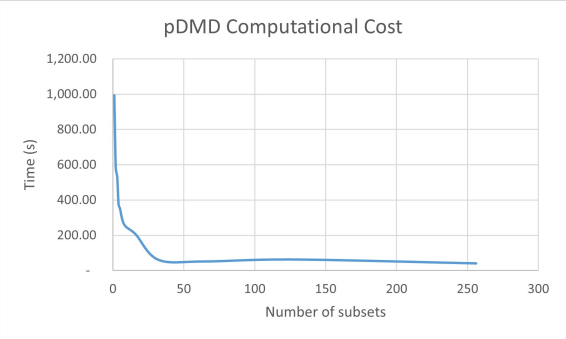


Figure 9. Computational cost analysis

5 Conclusion

This work has shown the capabilities of implementing Machine Learning techniques in CFD using DMD and pDMD in a two-dimensional flow around cylinder data. We have shown that the dynamic modes successfully capture the dynamics and reconstruct the approximation of the simulation. The use of pDMD allows the refinement of the results by decreasing the error of the approximation and capturing the modes in more detail in each flow stage and the formation of the vortex wake. Applying pDMD reduced up to 90%

Acknowledgements. We thank Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES for financing this research.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] G. V. Iungo, C. Santoni-Ortiz, M. Abkar, F. Porté-Agel, M. A. Rotea, and S. Leonardi. Data-driven reduced order model for prediction of wind turbine wakes. *Journal of Physics: Conference Series*, vol. 625, pp. 1–10, 2015.

[2] A. Sheidani, S. Salavatidezfouli, G. Stabile, and G. Rozza. Assessment of URANS and LES methods in predicting wake shed behind a vertical axis wind turbine. *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 232, pp. 1–15, 2023.

[3] R. Vinuesa and S. Brunton. Emerging trends in machine learning for computational fluid dynamics. *Computing in Science and Engineering*, vol. 24, pp. 33–41, 2022.

[4] P. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, vol. 656, pp. 5–28, 2010.

[5] A. Alla, A. Monti, and I. Sgura. Piecewise dmd for oscillatory and turing spatio-temporal dynamics. *Computers Mathematics with Applications*, vol. 160, pp. 108–124, 2024.

[6] J. N. Kutz, S. L. S. L. Brunton, B. W. Brunton, and J. L. Procor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM Society for Industrial and Applied Mathematics, 2016.

[7] S. M. Ichinaga, F. Andreuzzi, N. Demo, M. Tezzele, K. Lapo, G. Rozza, S. L. Brunton, and J. N. Kutz. Pydmd: A python package for robust dynamic mode decomposition. `https://doi.org/10.48550/arXiv.2402.07463`, 2024.

[8] N. Demo, M. Tezzele, and G. Rozza. Pydmd: Python dynamic mode decomposition. *The Journal of Open Source Software*, vol. 3, pp. 530, 2018.