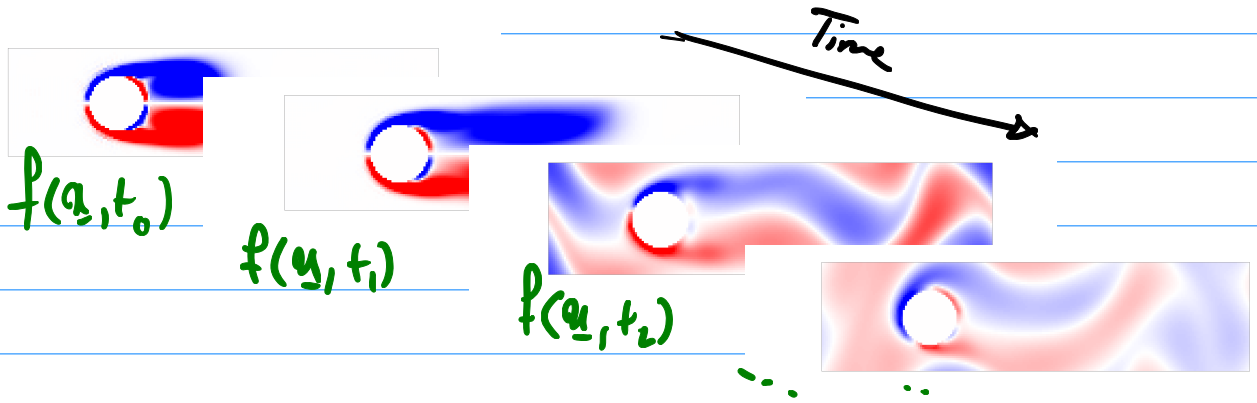
Lecture 11: From PCA & SVD to Proper Orthogonal decompositions & Dynamic Mode Decomposition

11.1 POD

So far we saw the connection between SVD & PCA & also discussed the statistical foundations of PCA. However, in many applications, specially in engineering, vanilla PCA is insufficient

ex Flow Past a Cylinder. (show GIF)



In such problems we are clearly dealing with functions (fields) of the form $f(x,t)$ where $f: \mathbb{R}^d \times [0,\infty) \to \mathbb{R}$.

However, you may observe that for fixed $t \geq 0$ the function $f(\cdot, t)$ may exhibit low dimensional structure. ex in the cylinder example. The vorticity fields are very smooth.

This observation suggests that a dimension reduction should be possible that takes the dynamic nature of data into account.

The Proper Orthogonal Decomposition (POD) method does one version of this. POD assumes

$$f(\underline{x},t) = \sum_{j=0}^{\infty} c_j(t)\psi_j(\underline{x})$$

Coeff. in time ↔ bases in $\underline{x}$

POD is the same linear model/idea we have seen before in DFT & DWT except that the time component is now separate & accounted for by the coeffs. c_k.

In this context ψ_j are called the POD Modes & the c_j are called the POD coeffs.

Luckily, & perhaps unsurprisingly, there is an easy way of computing POD using SVD.

Suppose dynamics $f(x,t)$ are observed over a discrete set, ie, $(\underline{x}_j, t_k)$, $j=0,\dots,N-1$, $k=0,\dots,T-1$

$$d_{0j} = f(\underline{x}_j, t_0) \quad D = \left[\, \underline{d}_0 = f(\underline{x}_j, t_0) \;\middle|\; \underline{d}_1 = f(\underline{x}_j, t_1) \;\middle|\; \cdots \;\middle|\; \underline{d}_{T-1} = f(\underline{x}_j, t_{T-1}) \,\right]$$

Then we simply compute the SVD of D

$$D = U \Sigma V^T$$

Then the columns of U are precisely the POD modes & the rows of ΣV^T are precisely the POD coeffs.

Note: sometimes the rows of V^T are called the POD coeffs - but in our terminology those are normalized POD coeffs.

This is straightforward to see since each column of D takes the form (assuming $N > T$)

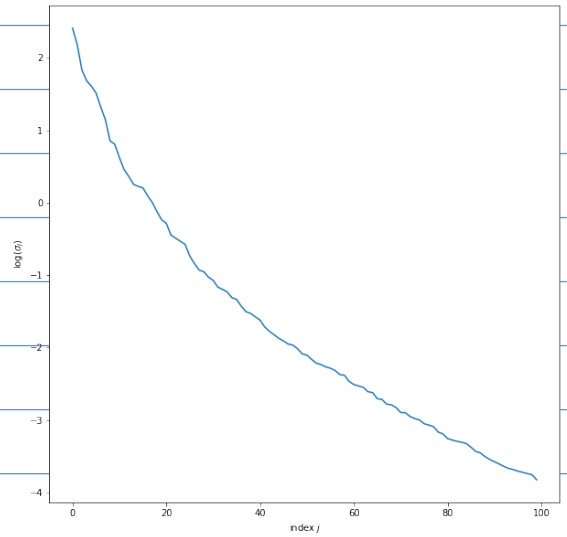
$$d_k = \sum_{j=0}^{T} u_j \sigma_j v_{jk}$$

(σ_j: singular values; u_j: cols. of U; v_{jk}: entries of V)

$$f(x, t_k) = \sum_j \psi_j(x)\, c_j(t_k)$$

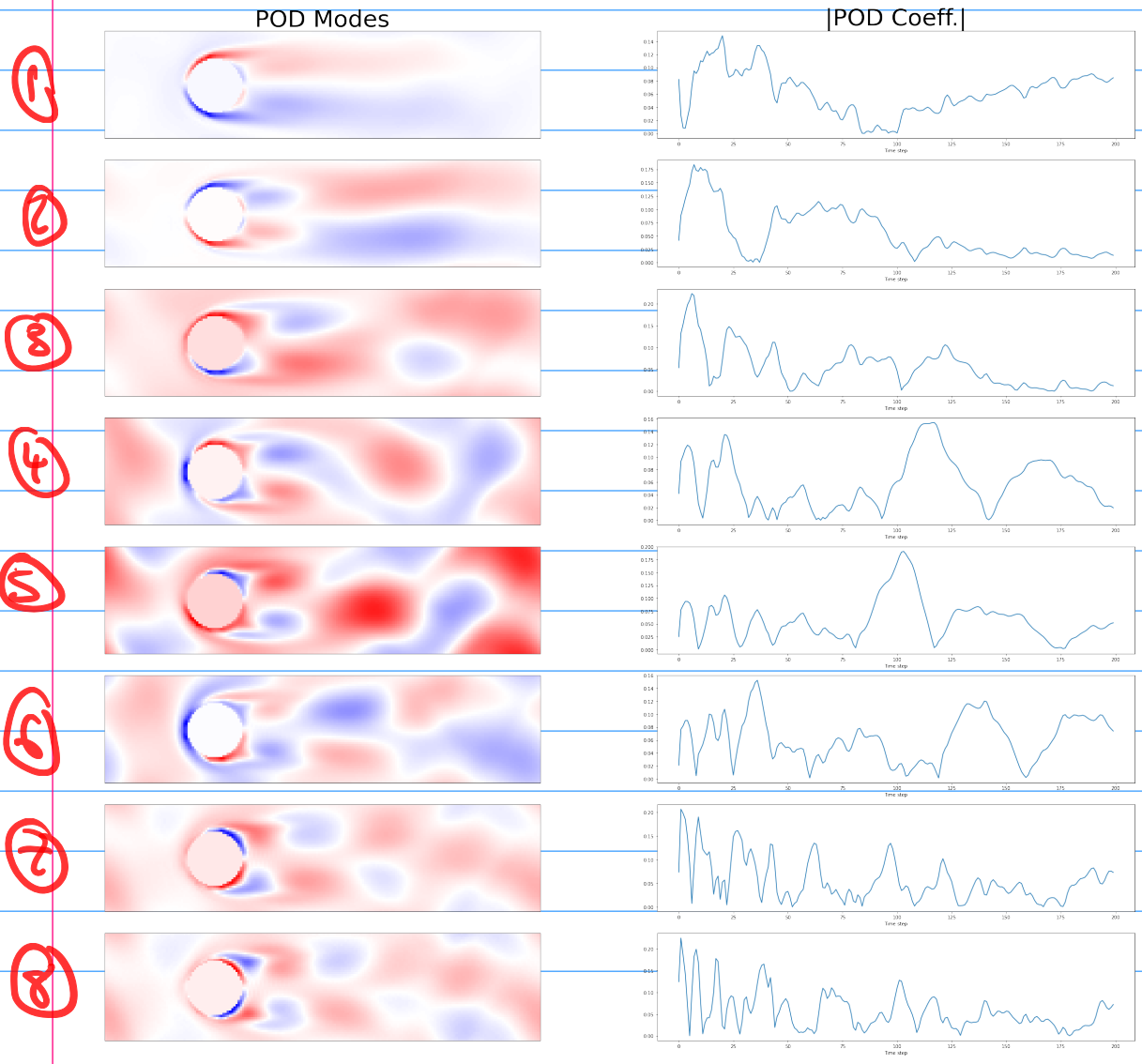
In summary, implementation is very simple!

Cylinder flow demo again



singular values decay rapidly. 6 orders of magnitude drop in first 40 singular vals!

Normalized



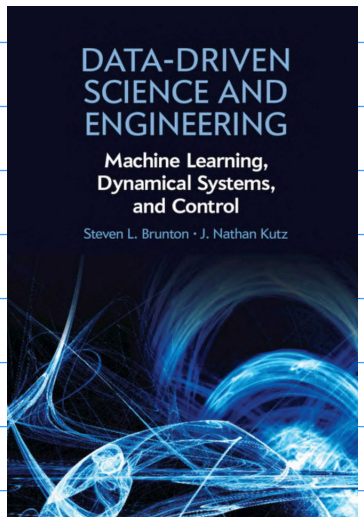
- Observe how the POD Modes summarize the coarse → fine features present in the flow!
- The |POD Coeff.| show us how active each POD mode is at any given time!

For ex mode 2 is mostly active in first half of simulation. Modes 4&5 are mostly active in the second half!

The POD is not the only approach out there for analysis of dynamic data. In fact, this field is very much the subject of intense research with major active teams at UW & our department. We will now discuss an advanced variant called Dynamic Mode Decomposition (DMD).

11.2 DMD

This is a very quick overview & demo for more details see ch7 of →



Look back at the POD coeffs. of the cylinder flow example. Observe that while the POD modes are smooth & structured the POD coeffs - are rough & chaotic.

This is because POD only sees spatial structure & ignores temporal dynamics. DMD attempts to address this issue. By modeling the dynamics of the data:

$$\underline{a}_{k+1} = A\underline{a}_k$$

$$f(x, t_{k+1}) \xleftarrow{A} f(x, t_k)$$

Let us restructure our data as follows:

$$\mathbb{R}^{N\times T-1} \ni D = \left[\underline{d}_0 \,\middle|\, \underline{d}_1 \,\middle|\, \cdots \,\middle|\, \underline{d}_{T-2}\right]$$

$$\mathbb{R}^{N\times T-1} \ni D' = \left[\underline{d}_1 \,\middle|\, \underline{d}_2 \,\middle|\, \cdots \,\middle|\, \underline{d}_{T-1}\right]$$

How we assume the snapshots d_k are obtained at uniform instances in time, ie, $t_k = t_{k-1} + \delta t$.

More generally you can assume data is given as pairs of snapshots $\{d_k, d'_k\}_{k=0}^{T}$ where $d_k = f(x, t_k)$, $d'_k = f(x, t_k + \delta t)$.

Then DMD seeks to find a matrix $A \in \mathbb{R}^{N\times N}$ such that

$$D' \approx AD,$$

That is, A is the linear operator that best describes the incremental dynamics of our data.

$$\underline{d}_{k+1} \approx A\underline{d}_k$$

This approx. is then done in a best-fit sense

$$(*) \qquad A = \underset{B\in\mathbb{R}^{N\times N},\ \mathrm{rank}(B)\le r}{\arg\min} \|BD - D'\|_F^2$$

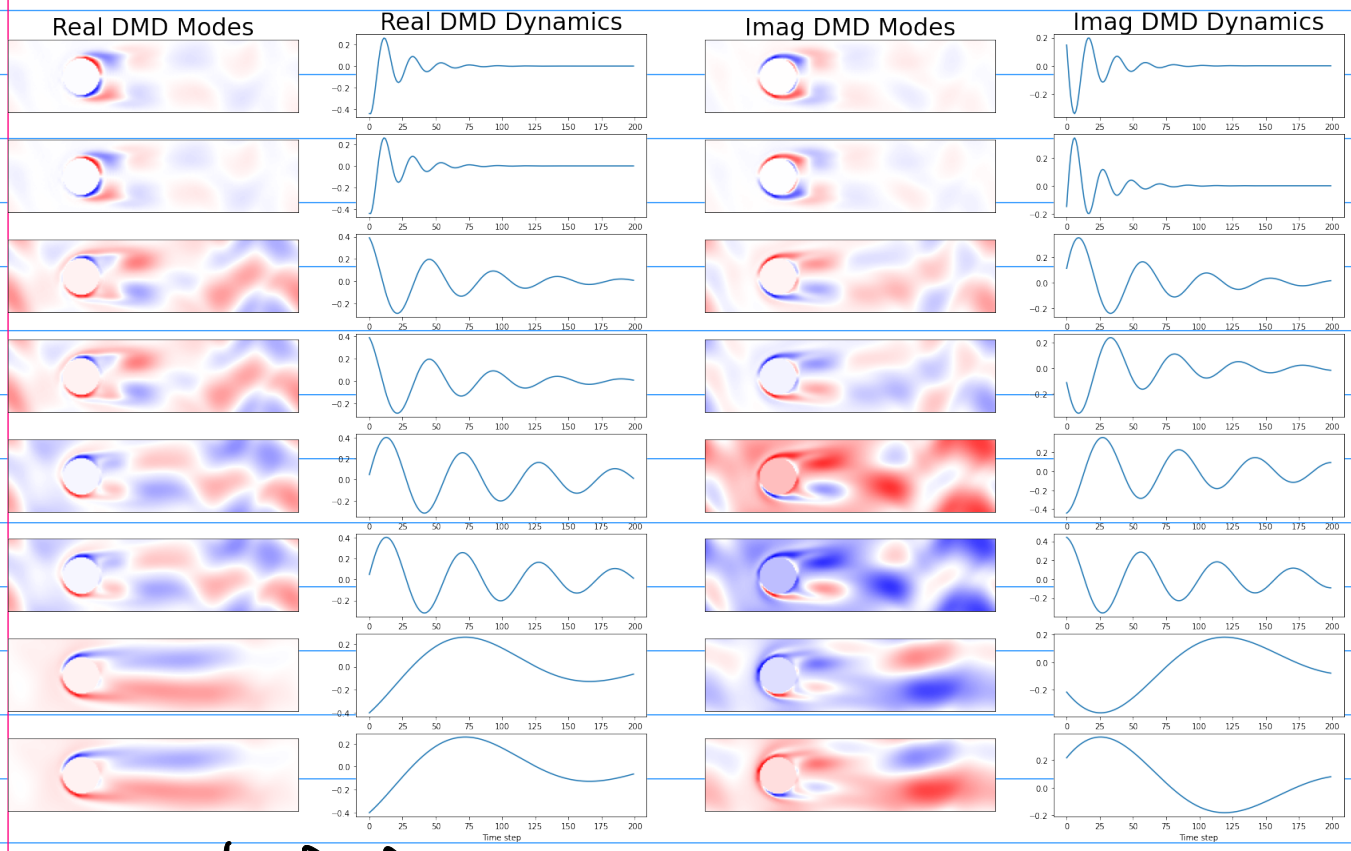
* Compare this to our discussion around optimality of low-rank matrix approx. via SVD in Lec10!

We won't get into the details of the algorithm. But DMD algorithms solve problem (*) to find a low rank matrix A that approximates the dynamics. In doing so, DMD returns appropriate orthogonal modes that approximate the spatial variations & the temporal dynamics in D!

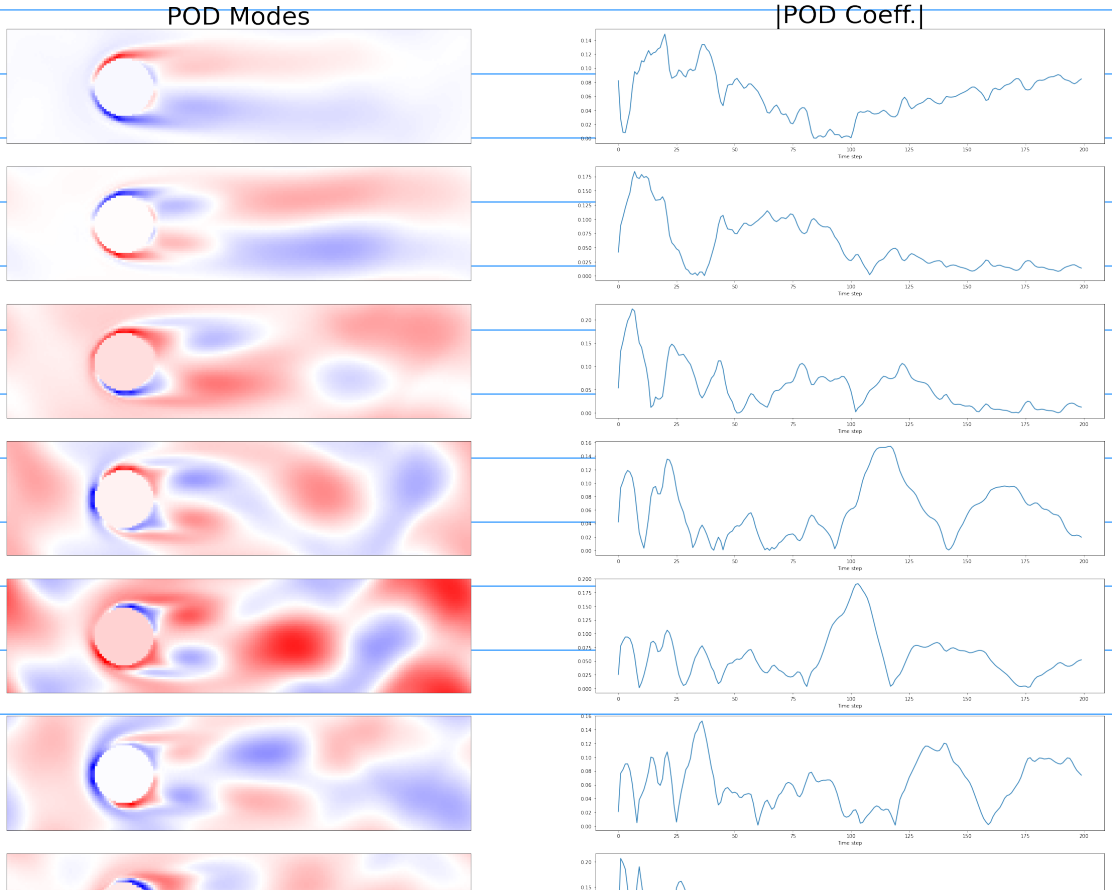
A very nice implementation exists in PyDMD which we now demo.

Note, the DMD algorithm in PyDMD relies on eigen decomposition of A & so returns complex eigenvalues & DMD modes/dynamics as a result!

* DMD modes & temporal dynamics associated to the dominant eigenvalues of A for Cyl. Flow.



* Compare to POD.



so what does DMD actually do?

$$\underline{d}_k = A\underline{d}_{k-1} = A(A\underline{d}_{k-2}) \ldots$$

$$\underline{d}_k = A^{k-1}\underline{d}_0 = Q\Lambda^{k-1}Q^{-1}\underline{d}_0$$

$$= Q\Lambda^{k-1}\underline{b} = \sum_{j=1}^{r} \underline{q}_j \cdot \lambda_j^{k-1} b_j$$

DMD modes

DMD dynamics

- DMD fits a linear model to temporal dynamics.
- It is good for stationary/steady state data
- Not so good for nonlinear & transient data ie, when linear model is not good.