UNIVERSITY OF TWENTE. | THERMAL ENGINEERING

# Application of POD and DMD in Fluid Dynamics Analysis

Alireza Ghasemi

MAGISTER

See more at https://www.magister-itn.eu/

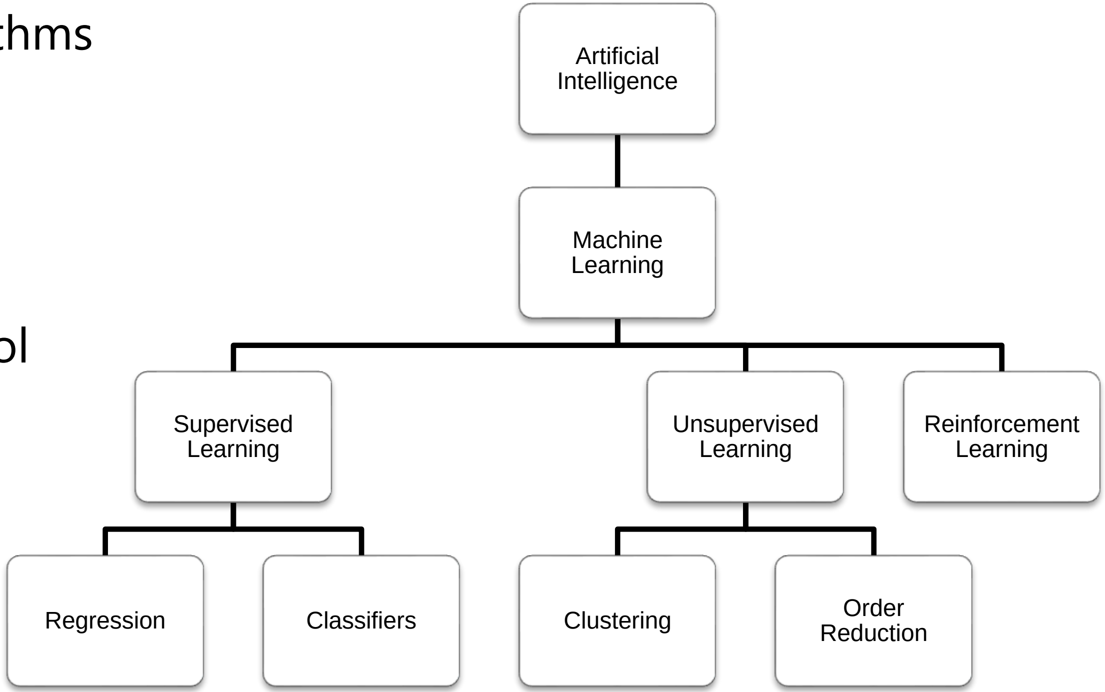
This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie Grant Agreement No 766264.

# OUTLINE

- Introduction to POD
- Simple Demo
- Practical Example
- Short Q&A
- Introduction to DMD
- Practical Example
- Q&A

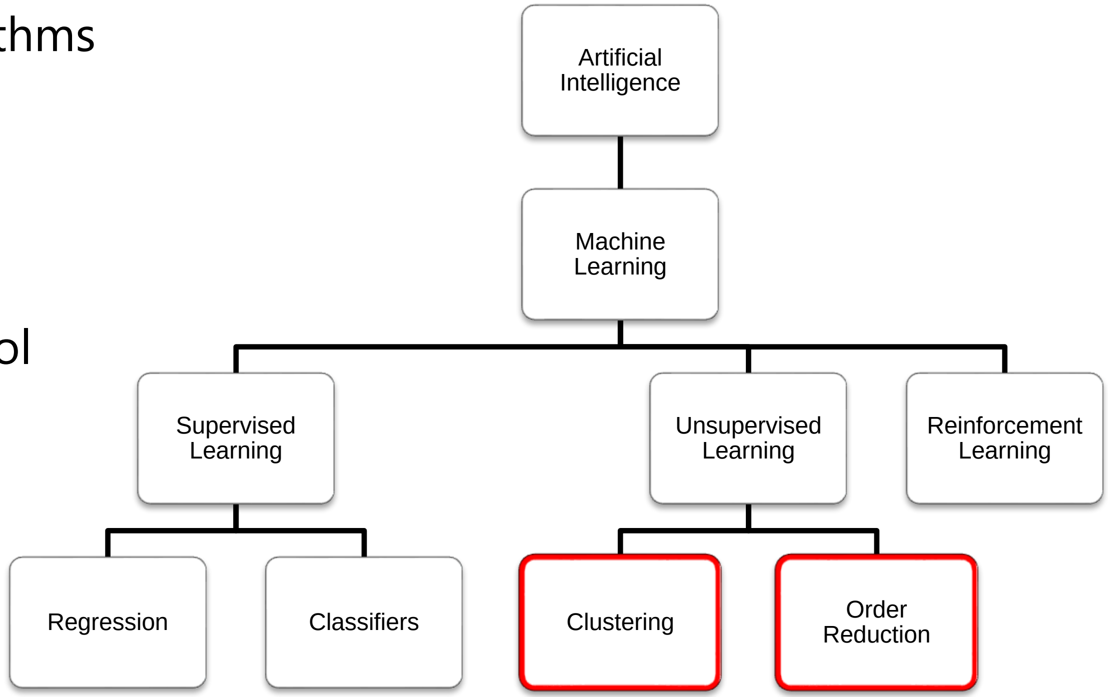
# ALGORITHM CLASSIFICATION

- Unsupervised Algorithms
- Data driven
- Patter recognition
- ROM
- Dynamics and Control



# ALGORITHM CLASSIFICATION

- Unsupervised Algorithms
- Data driven
- Patter recognition
- ROM
- Dynamics and Control



# POD: PROPER ORTHOGONAL DECOMPOSITION

- Also known as PCA: Principal Component Analysis
- Original formulation in 1967 by J. Lumley
- Widely used in Fluid Dynamics
- Many variations exist:
  - Missing data
  - Noisy or corrupted data
  - Multiple datasets
  - Multiscale

# DECOMPOSITION: MAIN IDEA

$$y(x,t) = \sum_{j=1}^{m} u_j(x) a_j(t)$$

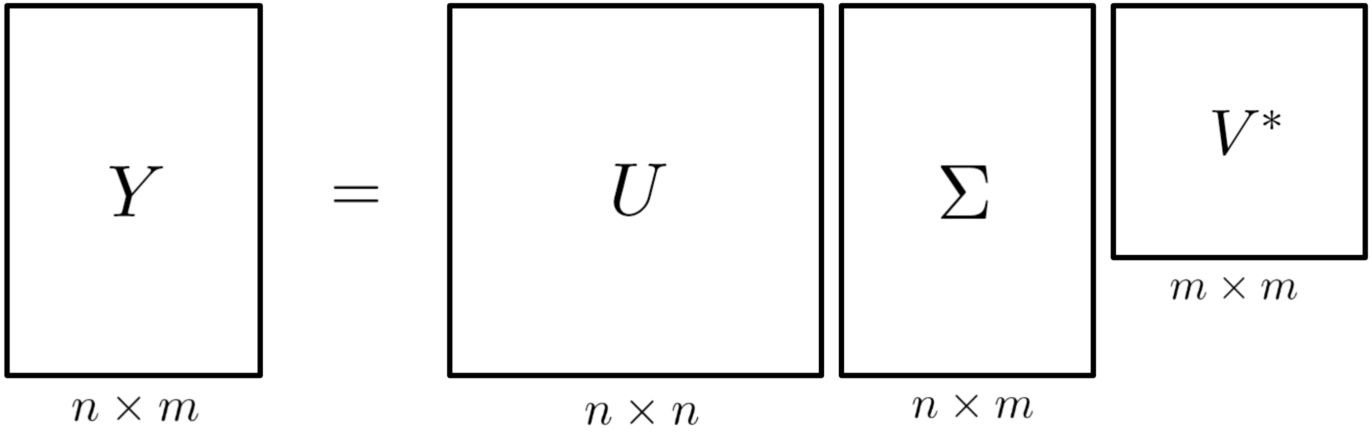
- Decomposing time variant systems into spatial and temporal parts
- In general, not a unique decomposition
- POD finds this decomposition with smallest “m”

# POD: DATA ASSEMBLY

$$Y = \begin{bmatrix} u(x_1, t_1) & u(x_1, t_2) & \cdots & u(x_1, t_m) \\ u(x_2, t_1) & u(x_2, t_2) & & u(x_2, t_m) \\ \vdots & \vdots & \ddots & \vdots \\ u(x_n, t_1) & u(x_n, t_2) & \cdots & u(x_n, t_m) \end{bmatrix}$$

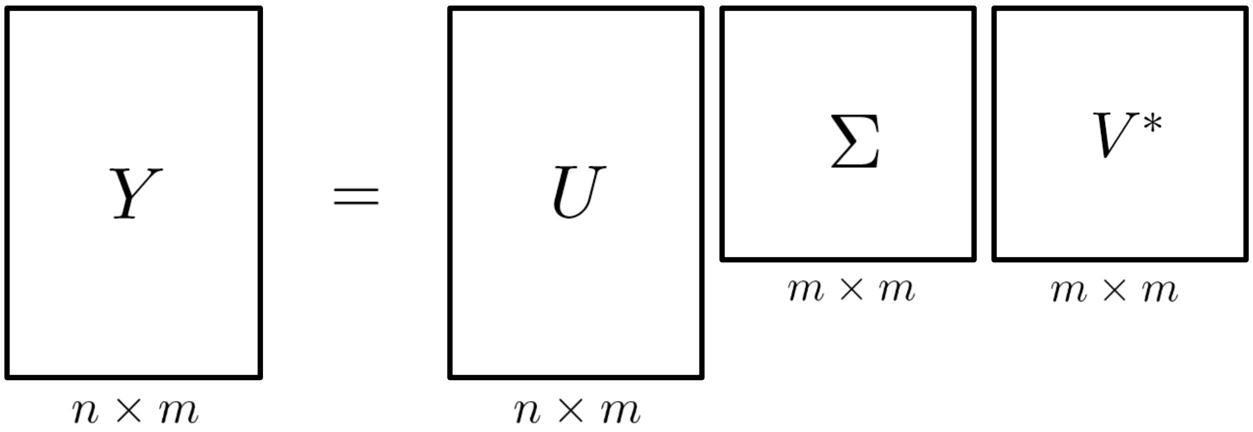
- Spatial dimension: n
- Temporal dimension: m
- Various assembly techniques are available

# ECONOMY SVD & TRUNCATION



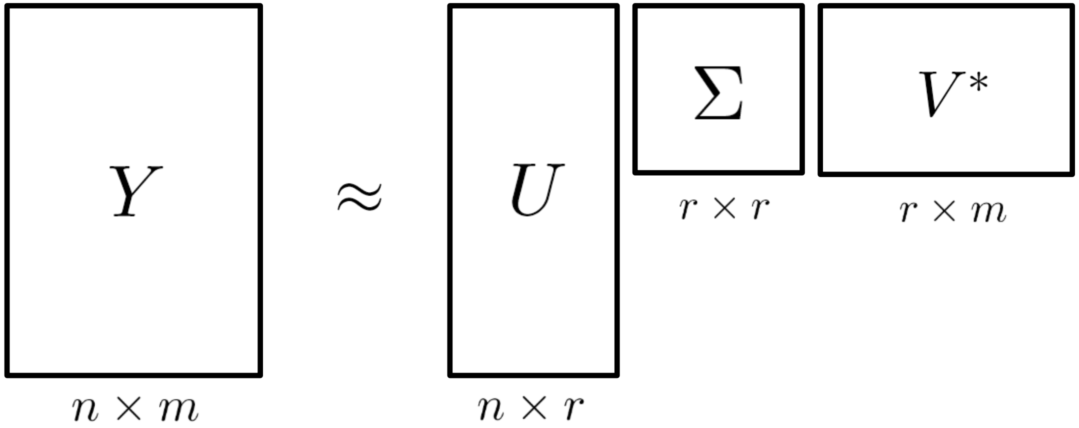

- We can trim U and Σ while keeping the same results
- U and V are unitary. Σ is diagonal with decreasing positive entries
- We can further trim for an approximation to Y

# ECONOMY SVD & TRUNCATION



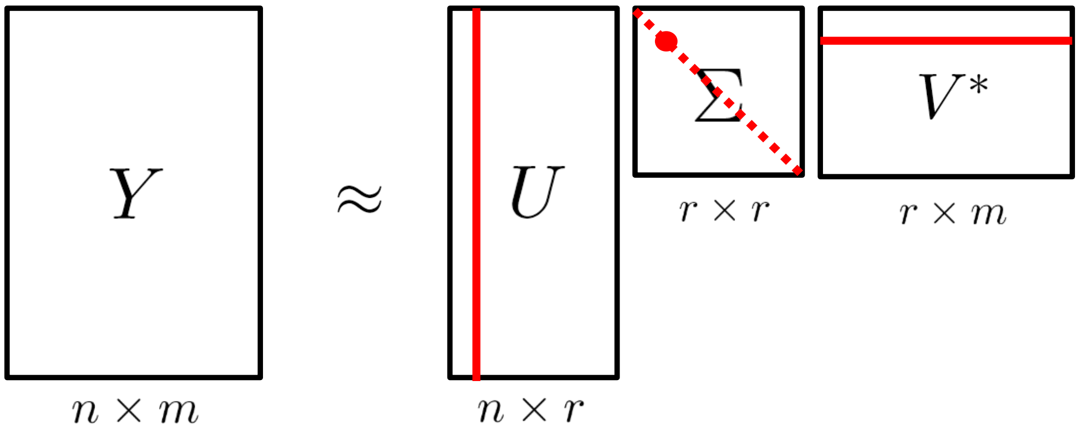

- We can trim U and Σ while keeping the same results
- U and V are unitary. Σ is diagonal with decreasing positive entries
- We can further trim for an approximation to Y

# ECONOMY SVD & TRUNCATION




- We can trim U and Σ while keeping the same results
- U and V are unitary. Σ is diagonal with decreasing positive entries
- We can further trim for an approximation to Y

# ECONOMY SVD & TRUNCATION




- We can trim U and Σ while keeping the same results
- U and V are unitary. Σ is diagonal with decreasing positive entries
- We can further trim for an approximation to Y

# POD: MODES AND COEFFICIENTS

$$Y = U \Sigma V^* \approx U_r \Sigma_r V_r^*$$

- Optimality of POD: Rank-r approximation minimizes the Frobenius norm

$$\|Y - Y_r\|_F = \sqrt{\sum_{j=1}^{n} \sum_{k=1}^{m} (Y - Y_r)_{jk}^2}$$

- U contains the POD modes
- Σ contains POD singular values
- V contains the right singular vector or POD coefficients

# DEMO

Let’s try it with the following synthetic signal

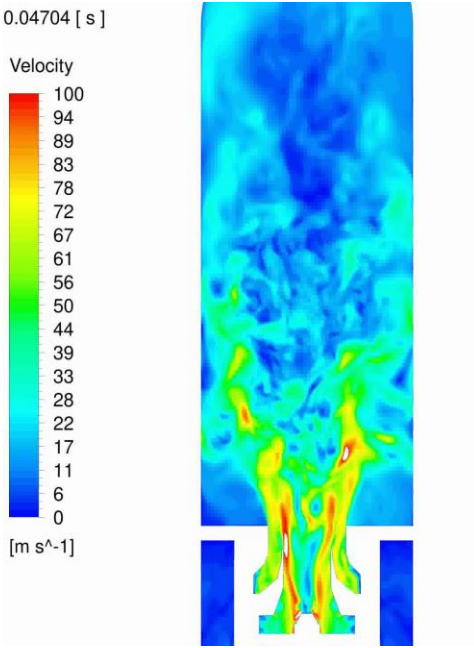
$$s_1 = a_1 \exp\left(-\frac{(x-b_1)^2}{2c_1^2}\right)$$

$$s_2 = a_2 \exp\left(-\frac{(x-b_2)^2}{2c_2^2}\right)$$

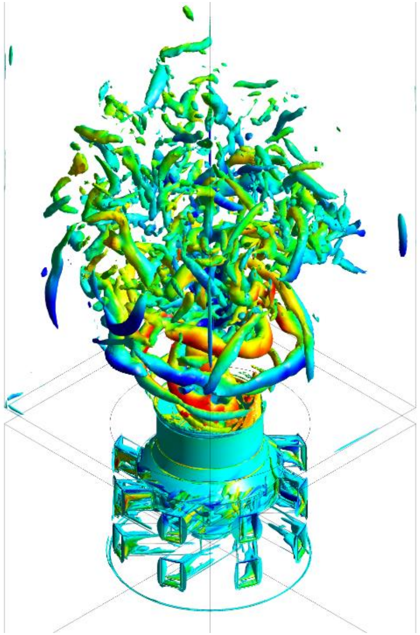
$$y(x,t) = s_1 \sin(2\pi\omega_1 t) + s_2 \sin\ (2\pi\omega_2 t)$$

# EXAMPLE: LES OF UT BURNER

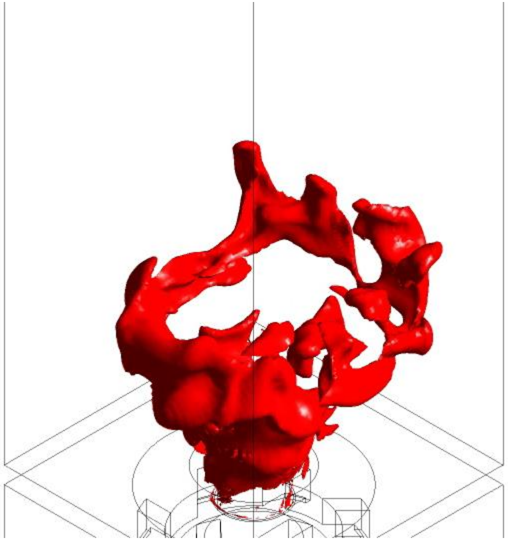
Velocity Contour




Q-Criterion

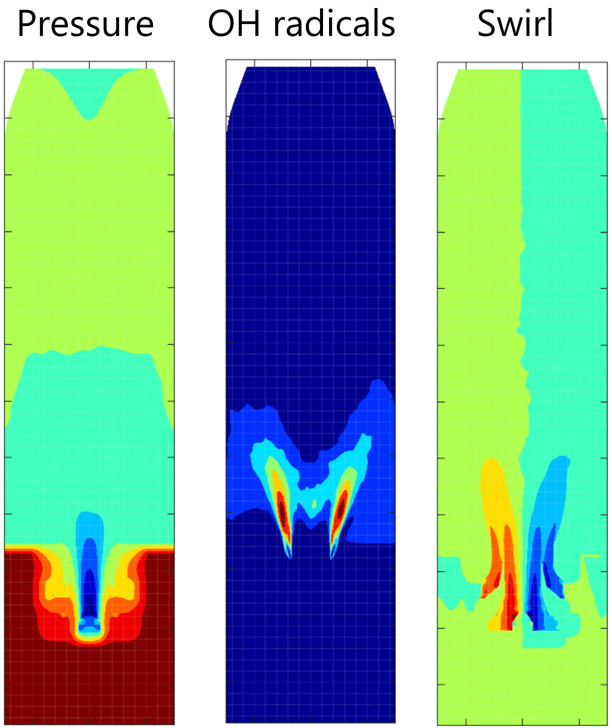


Flame Front



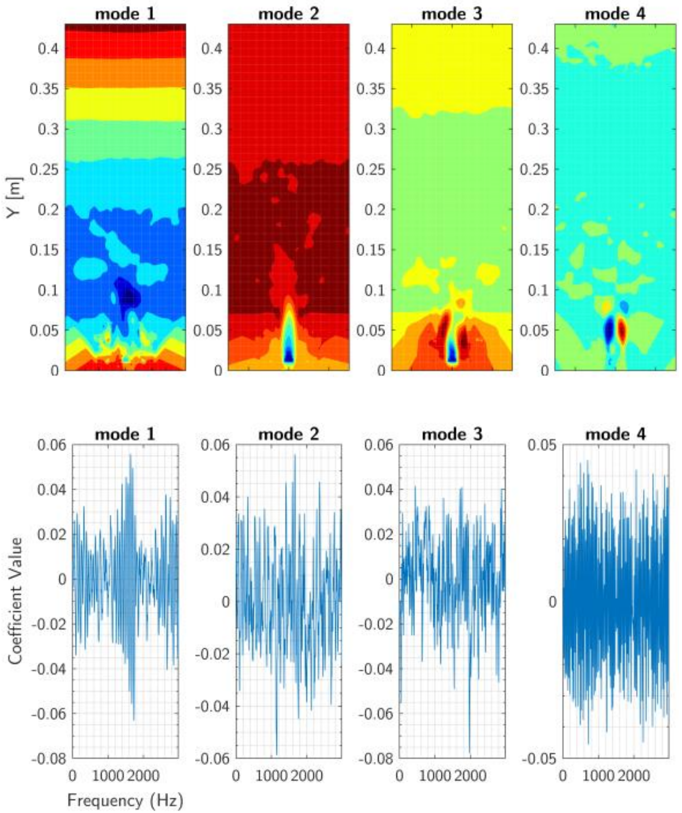
# POD APPLIED TO UT BURNER

- Identify mean values
- Breakdown into modes
- Extract flow structures
- Able to identify PVC
- Energy breakdown of modes



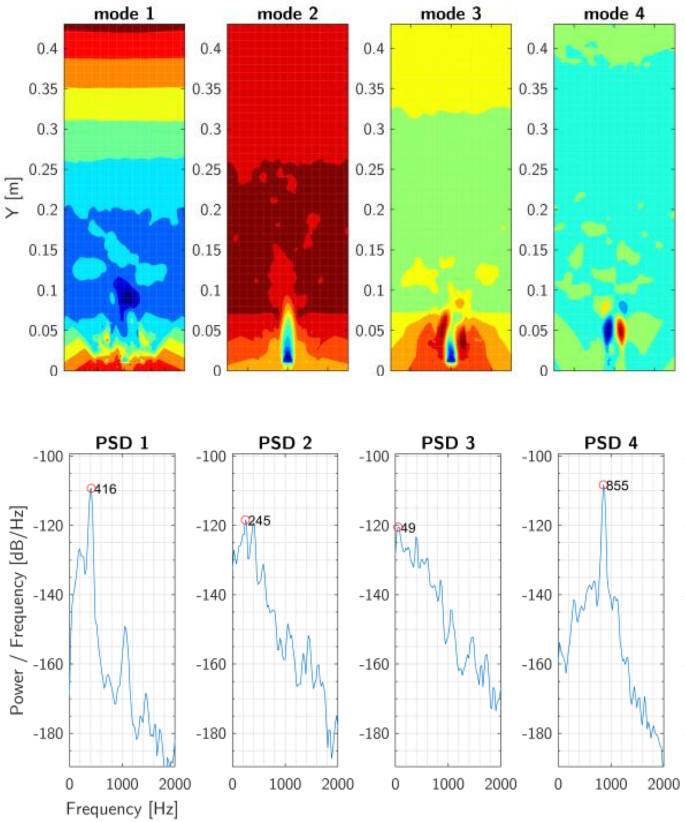
# POD APPLIED TO UT BURNER

- Identify mean values
- Breakdown into modes
- Extract flow structures
- Able to identify PVC
- Energy breakdown of modes



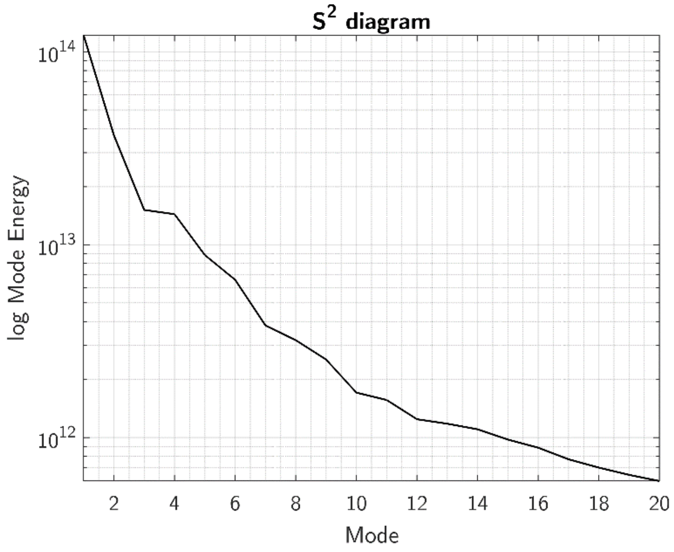
# POD APPLIED TO UT BURNER

- Identify mean values
- Breakdown into modes
- Extract flow structures
- Able to identify PVC
- Energy breakdown of modes



# POD APPLIED TO UT BURNER

- Identify mean values
- Breakdown into modes
- Extract flow structures
- Able to identify PVC
- Energy breakdown of modes



# DMD: DYNAMIC MODE DECOMPOSITIONS

- Think of it as POD + DTFT
- Developed in 2010 by P. Schmid
- DMD modes are not orthogonal like POD
- Many extensions exist:
  - GPU CUDA implementation
  - Compressed sensing
  - Noisy data
  - Multiresolution
  - Robustness

# KOOPMAN ANALYSIS BASICS

- Formalism for dynamic systems: Koopman Theory 1931
- Imagine the following discrete time system with state vector x

$$x_{n+1} = F(x_n)$$

- Observables of the state vector or functions of x is defined as $g(x)$
- Koopman operator evolves the system in the following manner

$$g(x_{n+1}) = Kg(x_n)$$

- Koopman operator turns a finite nonlinear dynamical system in vector space into an infinite dimensional linear system in function space $g(x)$

# DMD: MAIN IDEA

- DMD gets us the best linear approximation to a dynamical system by approximating the infinite-dimensional Koopman operator
- Let's assemble our Y matrix in the two following forms

$$\text{Y}=\begin{bmatrix} | & | & & | \\ y_1 & y_2 & \dots & y_{m-1} \\ | & | & & | \end{bmatrix} \qquad \text{Y}'=\begin{bmatrix} | & | & & | \\ y_2 & y_3 & \dots & y_m \\ | & | & & | \end{bmatrix}$$

- Looking for an optimally locally linear approximation in the following form

$$Y' \approx AY$$

# DMD: ALGORITHM

- Best-fit A matrix is given by $A = Y'Y^{\dagger}$
- This, again, minimizes the aforementioned Frobenius norm
- By using SVD or $Y = U\Sigma V^*$ we can work out the following

$$A = Y'V\Sigma^{-1}U^*$$

- For efficiency, instead of $A$, we compute $\tilde{A}$ which is the projection of A onto POD modes in the following manner

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

# DMD: ALGORITHM

- Now we compute the eigen decomposition of $\tilde{A}$ or simply: $\tilde{A}W = W\Lambda$
- Eigenvalues of $\tilde{A}$ are the same as $A$
- It can be shown that eigenvectors of $\Phi = \mathrm{UW}$ are the same as $A$

$$\Phi = Y' V \Sigma^{-1} W$$

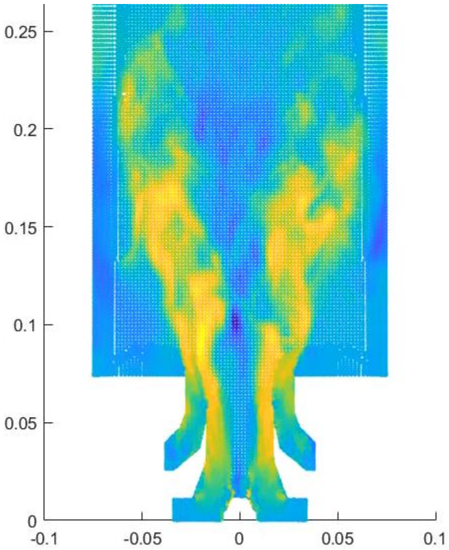
- We compute eigenvectors (modes) and eigenvalues (time dynamics) of $A$ without ever actually computing $A$
- We can still adopt a low rank approach like before

# DMD: CODE

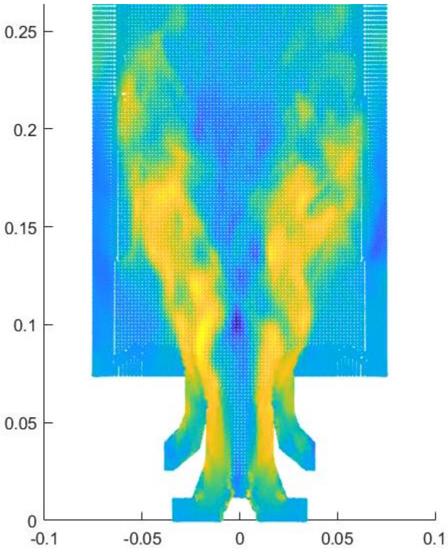
1. Prepare the data matrix $Y$ and $Y'$ in the correct form
2. Run economy SVD `[U ,S, V]=svd(Y,'econ');`
3. Find $\tilde{A}$ `Atilde = U'*Yp*V/S;`
4. Find $W$ and $D$ `[W, D] = eig(Atilde);`
5. Find time eigenvalues `omega = log(diag(D))/dt;`
6. Find DMD modes `Phi = Yp*V/S*W;`
7. Find DMD mode coefficients `b = Phi\Y(:, 1);`
8. Construct your approximation `Ydmd = Phi*(b.exp(omega*time));`

# EXAMPLE: LES OF UT BURNER

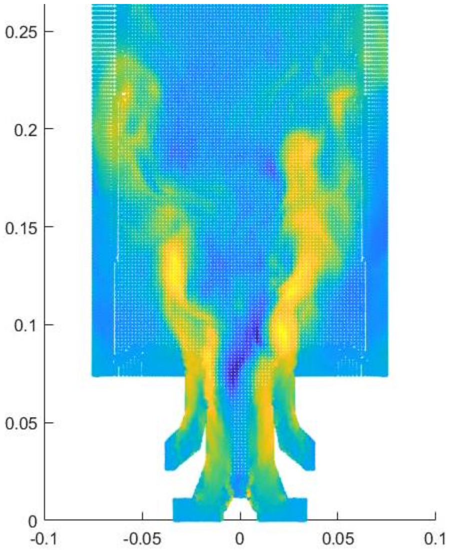
DMD




DMD ROM




Prediction



Thank you for your attention!

Stay healthy :)