université de BORDEAUX

Soutenance de thèse de doctorat

Approximations de rang faible et modèles d'ordre réduit appliqués à quelques problèmes de la mécanique des fluides

Low rank approximation and reduced order modeling applied to some fluid dynamics problems

Lucas Lestandi

Sous la direction de
Mejdi Azaïez et Tomás Chacón

October 16, 2018

I2M BORDEAUX

Scientific computing ?

- **Physics & Engineering**

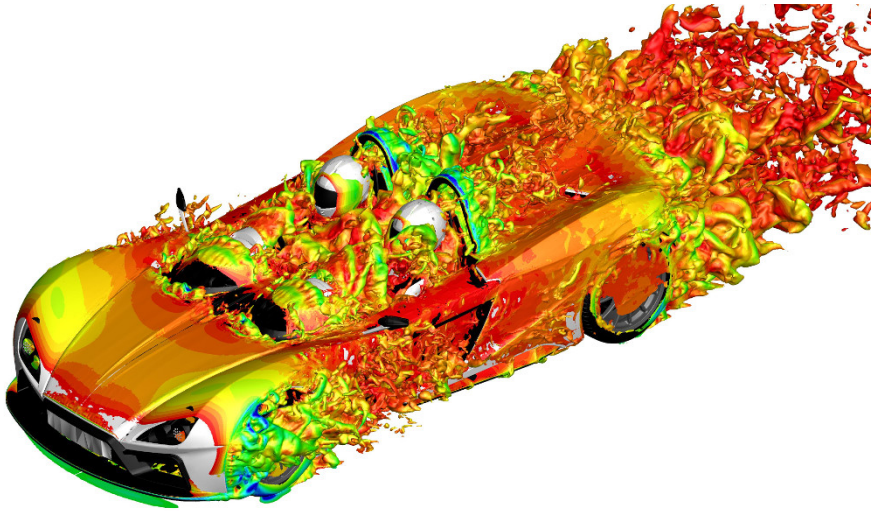


Figure: Full car FE simulation. Courtesy: Nektar++.

Figure: Water drop falling into water. Courtesy: Notus CFD.

Scientific computing ?

- Physics & Engineering
- **Energy**

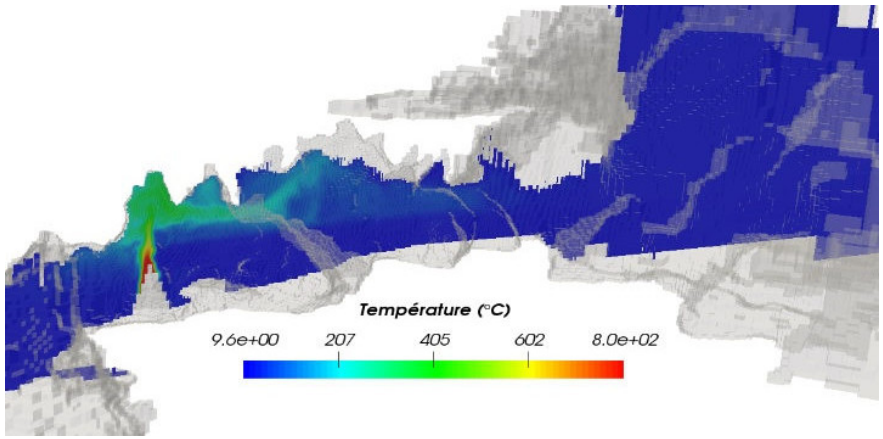


Figure: Cave fire simulation. Courtesy of F. Salmon.

Scientific computing ?

- Physics & Engineering
- Energy
- **Weather, earth sciences**

Figure: Hurricane Irma trajectory forecast. Courtesy: NOAA.

Scientific computing ?

- Physics & Engineering
- Energy
- Weather, earth sciences
- **Biology**

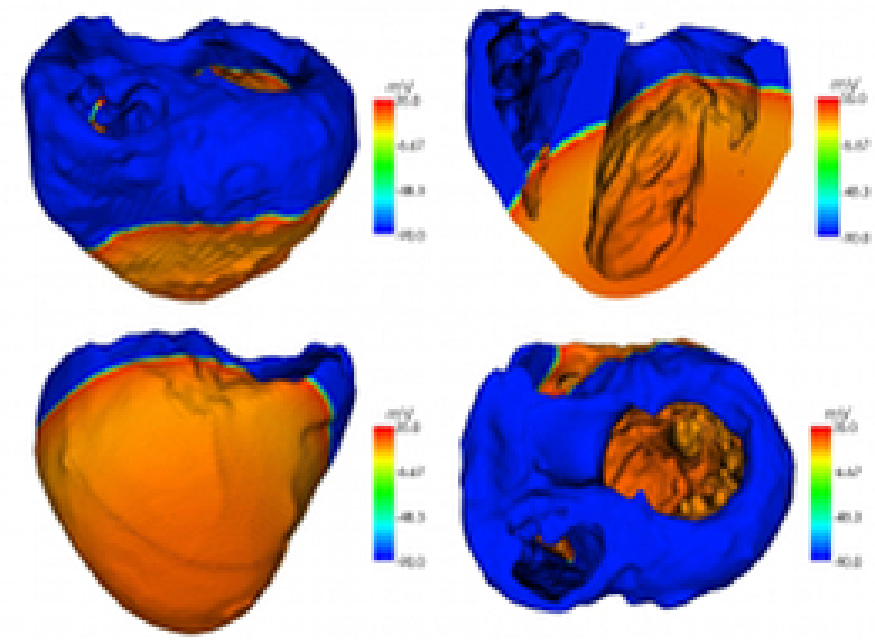


Figure: Heart depolarization wave. Courtesy: Team CARMEN Inria.

Scientific computing ?

- Physics & Engineering
- Energy
- Weather, earth sciences
- Biology
- **Finance**

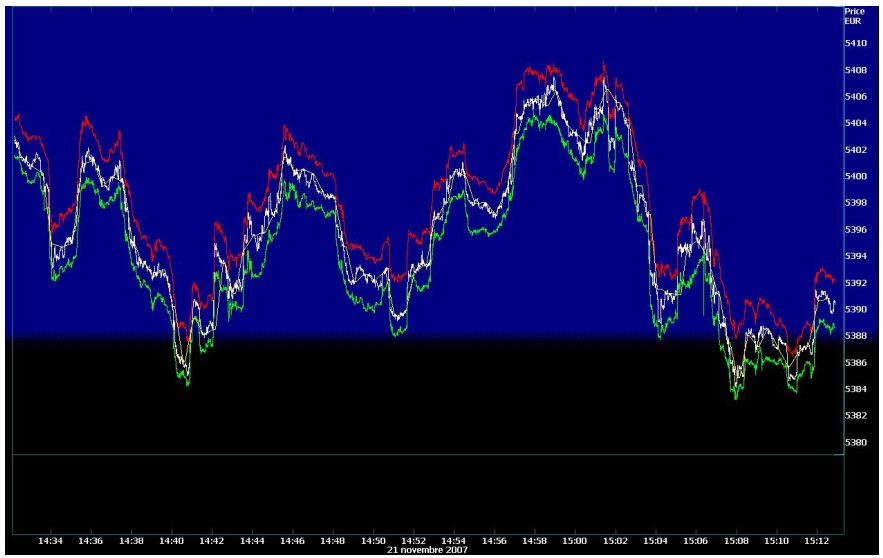


Figure: CAC 40 modeling. Courtesy: FiQuant.

Scientific computing ?

- Physics & Engineering
- Energy
- Weather, earth sciences
- Biology
- Finance
- **Everything that can be written as equations !**

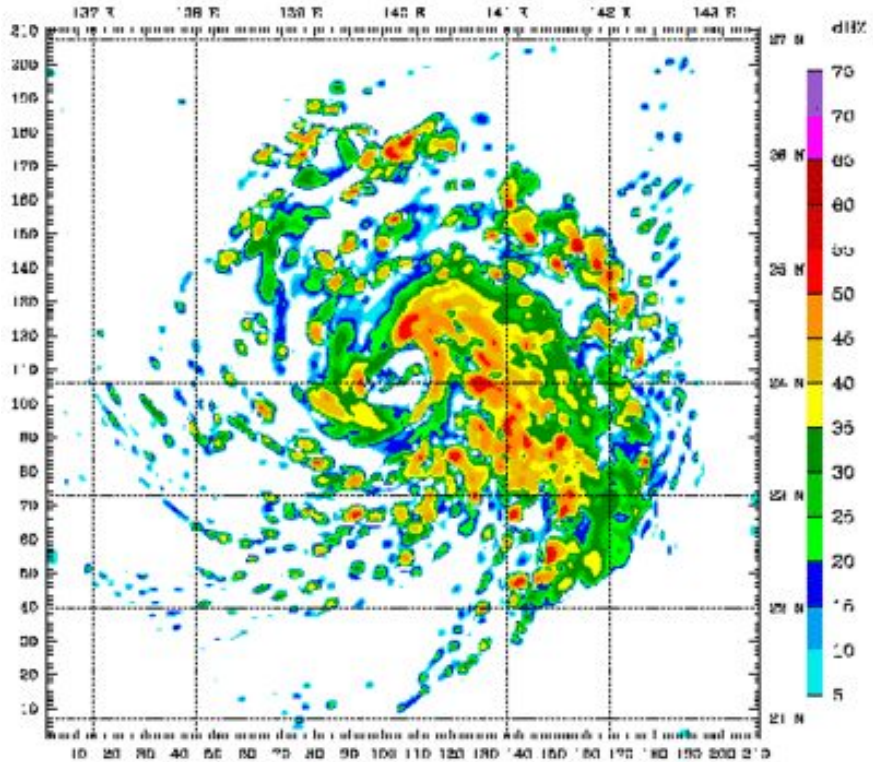
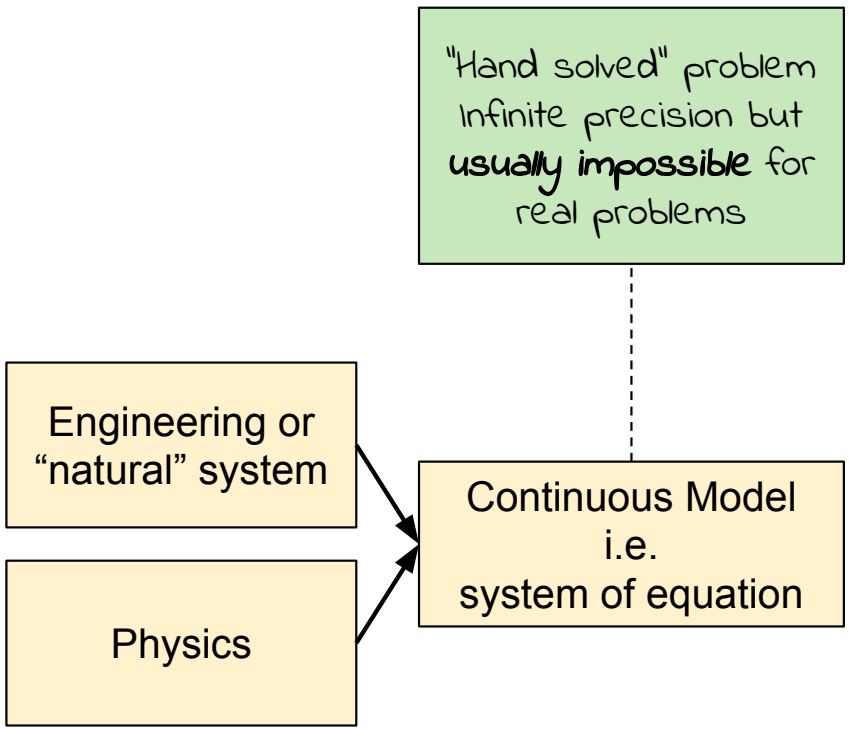
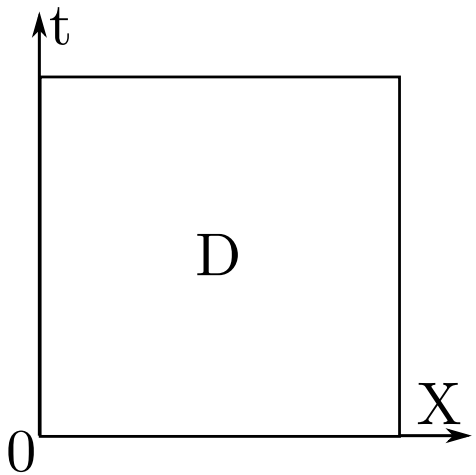


Figure: Typhoon Mawar.

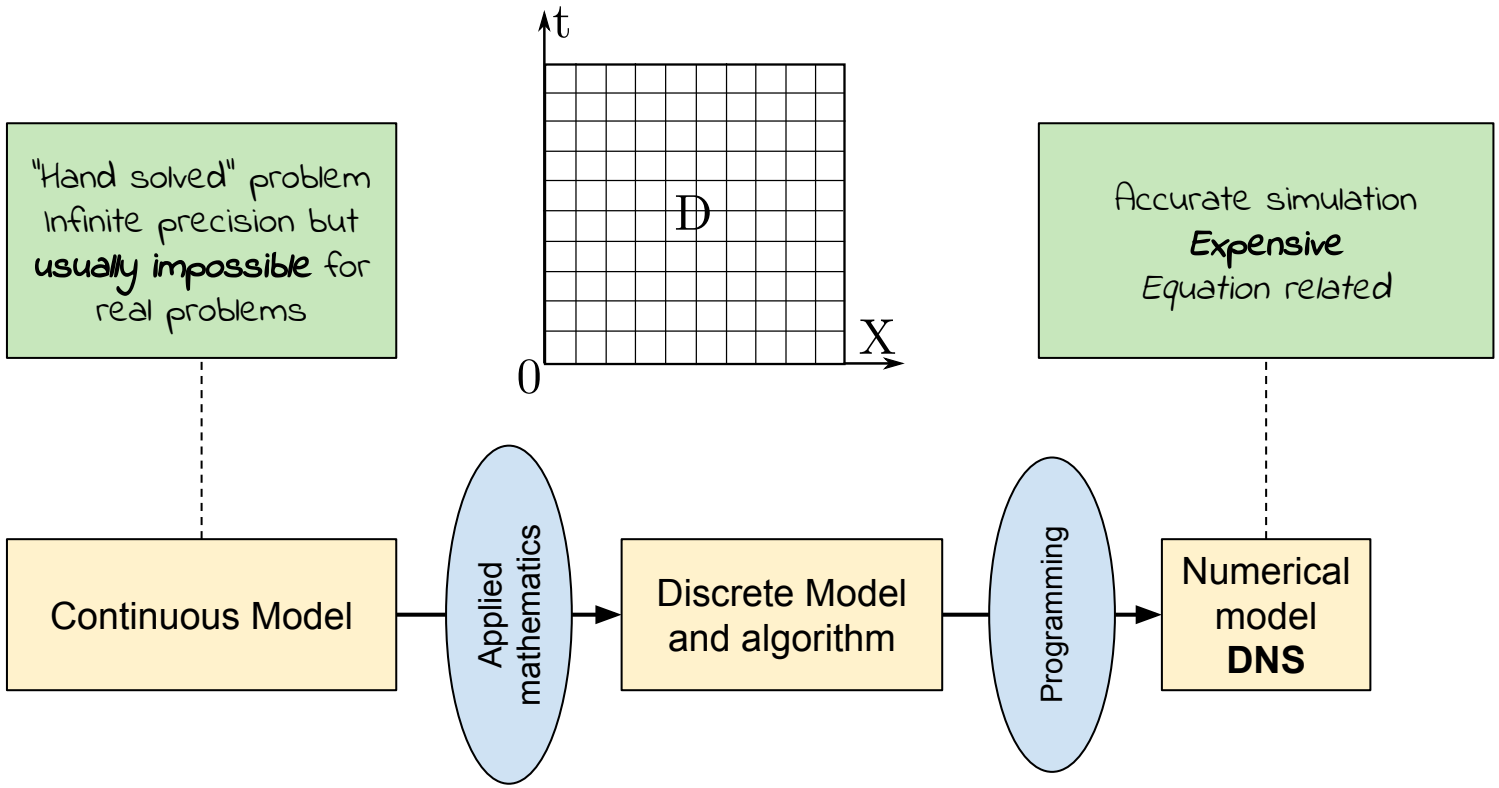
Numerical simulation in fluids





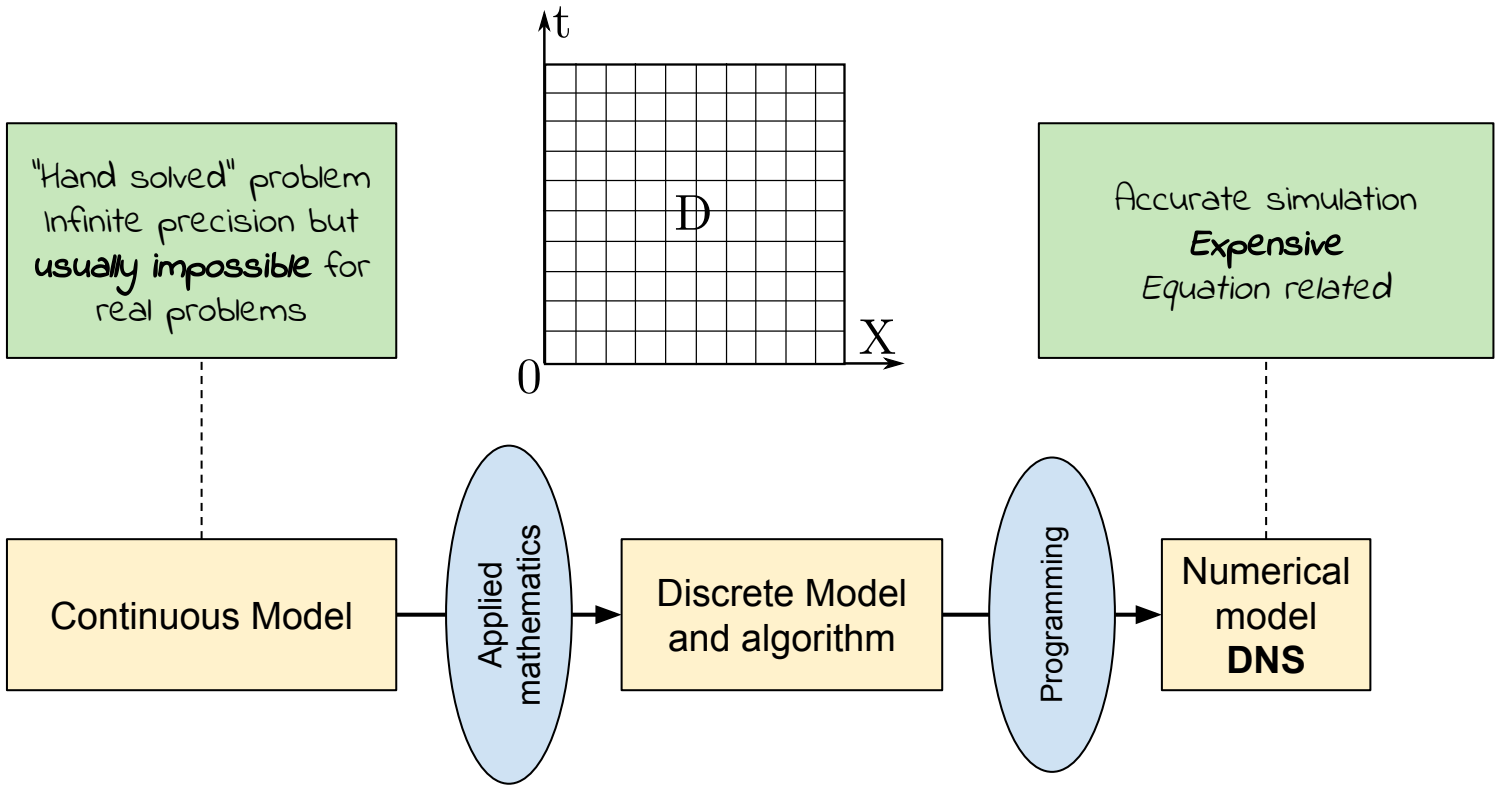
$$\frac{\partial v}{\partial t} + c\frac{\partial v}{\partial x} = 0$$

Numerical simulation in fluids



$$\frac{\partial v}{\partial t} + c\frac{\partial v}{\partial x} = 0 \Longrightarrow \frac{v_i^{n+1} - v_i^n}{\delta t} + c\frac{v_{i+1}^n - v_{i-1}^n}{\delta x} = 0$$

Numerical simulation in fluids



$$\frac{\partial v}{\partial t} + c\frac{\partial v}{\partial x} = 0 \Longrightarrow \frac{v_i^{n+1} - v_i^n}{\delta t} + c\frac{v_{i+1}^{n+1} - v_{i-1}^{n+1}}{\delta x} = 0 \Longrightarrow \boldsymbol{v^{n+1}} = A^{-1}\boldsymbol{v^n} \quad \text{with } A \in \mathbb{R}^{N\times N}$$

Numerical simulation is expensive

Solve n_t times linear system of size N:

$$Ax = b, \quad x \in \mathbb{R}^N$$

Numerical simulation is expensive

Solve n_t times linear system of size N:

$$Ax = b, \quad x \in \mathbb{R}^N$$

Costs $\mathcal{O}(N \log(N))$ FLOPS.

Numerical simulation is expensive

Solve n_t times linear system of size N:

$$Ax = b, \quad x \in \mathbb{R}^N$$

Costs $\mathcal{O}(N \log(N))$ FLOPS.

The curse of dimensionality

Let $N = 1000$,

d	shape	DoF	Mem
1	1000	10^3	7.8 kB
2	1000^2	10^6	7.6 MB
3	1000^3	10^9	7.4 GB
3D+time	1000^4	10^{12}	7.3 TB

Numerical simulation is expensive

Solve n_t times linear system of size N:

$$Ax = b, \quad x \in \mathbb{R}^N$$

Costs $\mathcal{O}(N \log(N))$ FLOPS.

The curse of dimensionality

Let $N = 1000$,

d	shape	DoF	Mem
1	1000	10^3	7.8 kB
2	1000^2	10^6	7.6 MB
3	1000^3	10^9	7.4 GB
3D+time	1000^4	10^{12}	7.3 TB

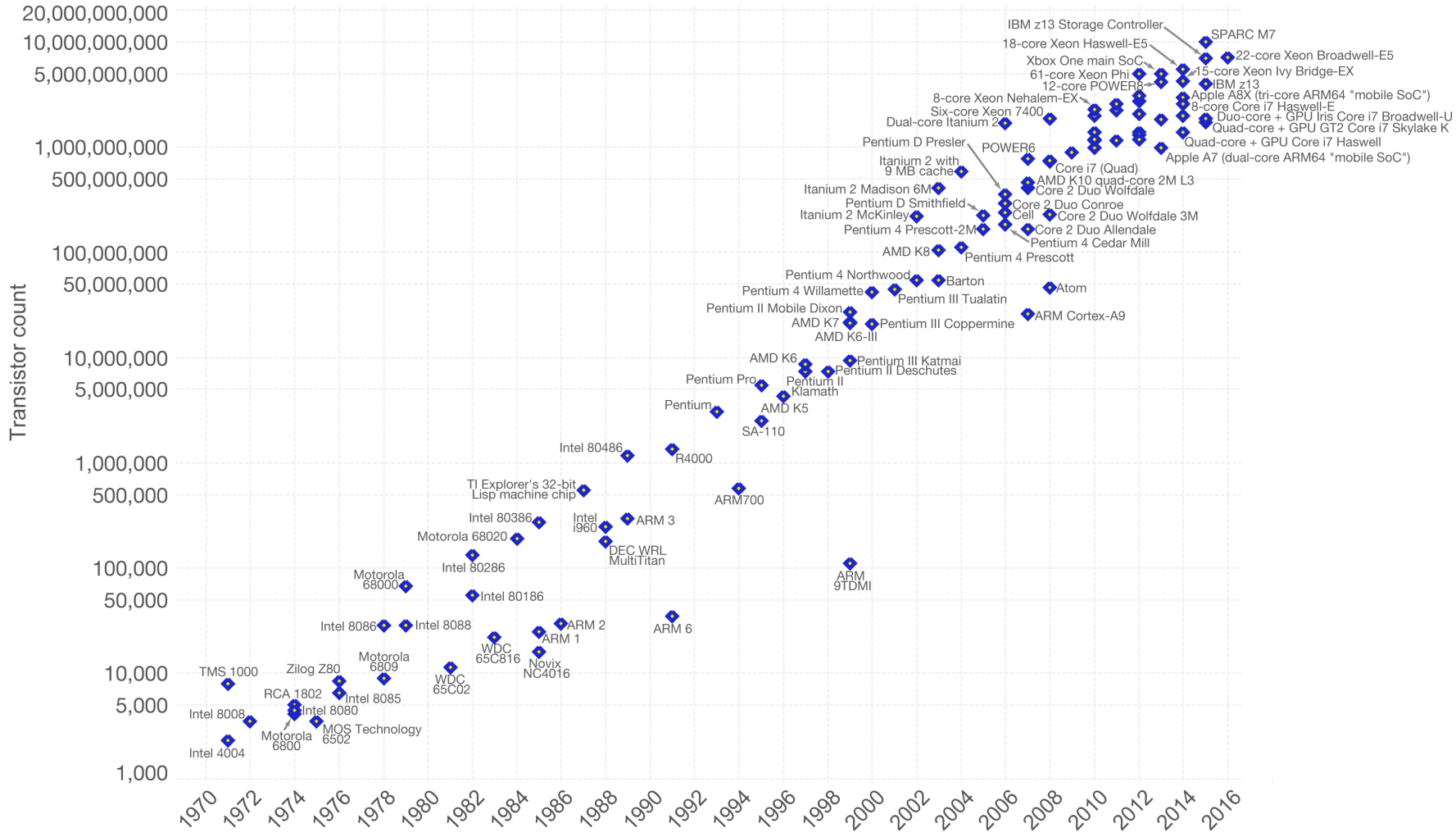
- Other parameters : viscosity, stiffness, shape,... $\Longrightarrow$ many dimensions
- Control, optimization $\Longrightarrow$ many queries

Rise of the computing power

Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

Big Data ?

Modern Super Computers

Scale	Name	Loc.	Country	Cores	Pflops	Power (kW)
Local	Condor	TREFLE,I2M	FR	400	NA	NA
National	Occigen	CIMES	FR	80,000	2,5	1400
World	Summit	Oak Ridge	USA	2,282,544	122	8806

Big Data ?

Modern Super Computers

Scale	Name	Loc.	Country	Cores	Pflops	Power (kW)
Local	Condor	TREFLE,I2M	FR	400	NA	NA
National	Occigen	CIMES	FR	80,000	2,5	1400
World	Summit	Oak Ridge	USA	2,282,544	122	8806

Large Numerical Simulations

Water drop simulation (2017, F. Desmons)

- 70M points
- time step 10^{-5}, 200 snapshots
- OCCIGEN : 1500 CPU, 6 days
- 2 TB of data

Full Universe (2012, CEA, Curie)

- 550 billion particles, 2 trillion points
- 300TB MEM, 80 000 CPU
- 10M CPU hours
- 50 PB of data

Reduced order modeling and data reduction

Reduced order modeling and data reduction

Change of paradigm

- Break the curse of dimensionality to open new areas

Reduced order modeling and data reduction

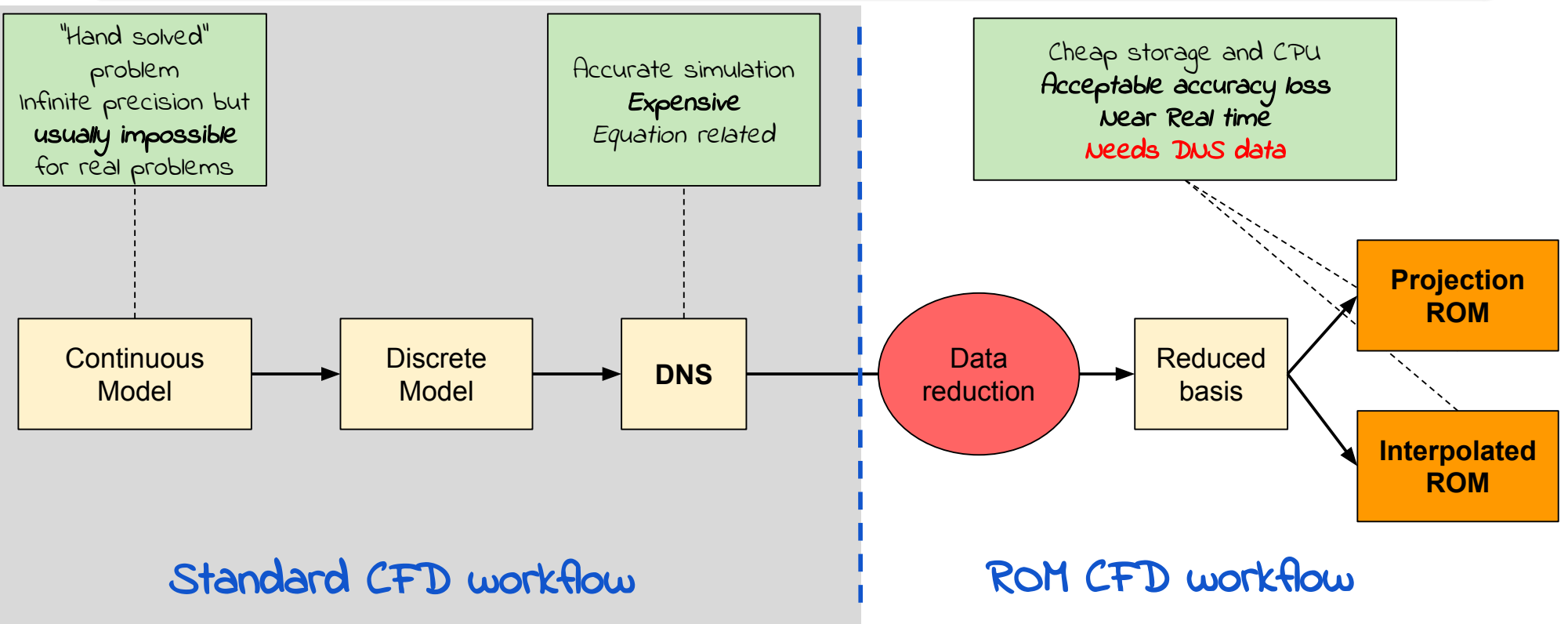
Change of paradigm

- Break the curse of dimensionality to open new areas
- Cost from N^d to $\boldsymbol{rNd}$ with $r \ll N$

Reduced order modeling and data reduction

Change of paradigm

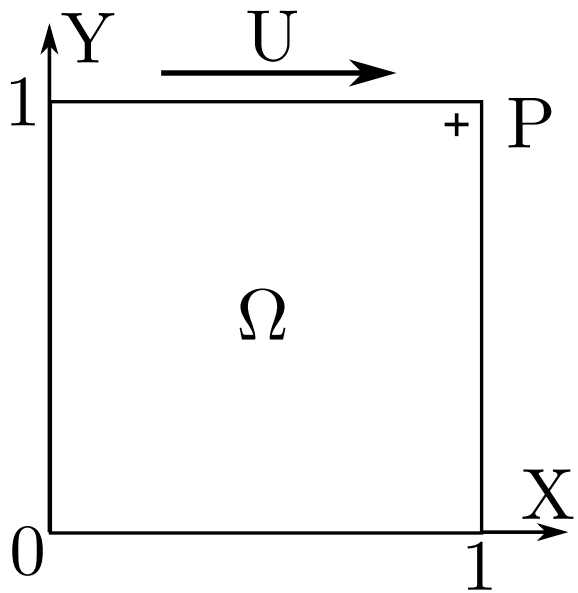
- Break the curse of dimensionality to open new areas
- Cost from N^d to ***rNd*** with $r \ll N$



Outline

Singular Lid Driven Cavity

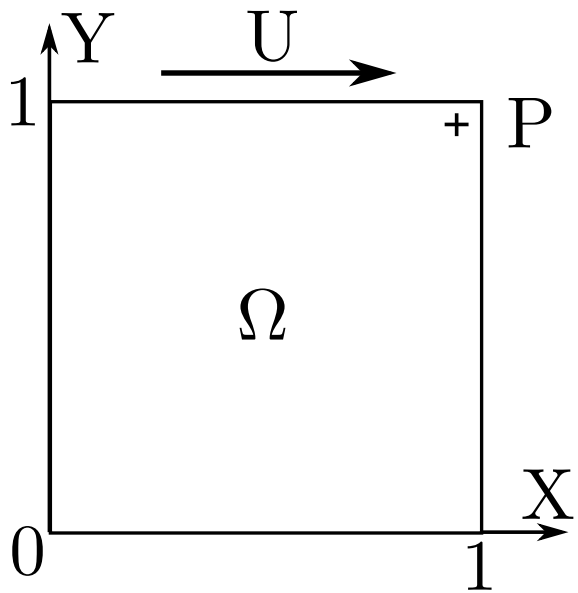
2D Singular Lid Driven Cavity



Numerical setup

- $Re \in [8000, 12000]$
- Unstable flow with bifurcations

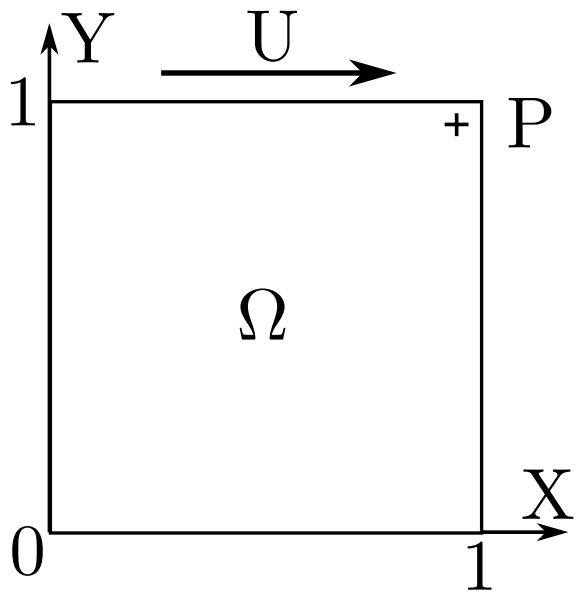
2D Singular Lid Driven Cavity



Numerical setup

- $Re \in [8000, 12000]$
- Unstable flow with bifurcations
- 2D NS vorticity formulation
- High accuracy NCCD solver
- $257x257$ cartesian grid, $dt = 10^{-3}$

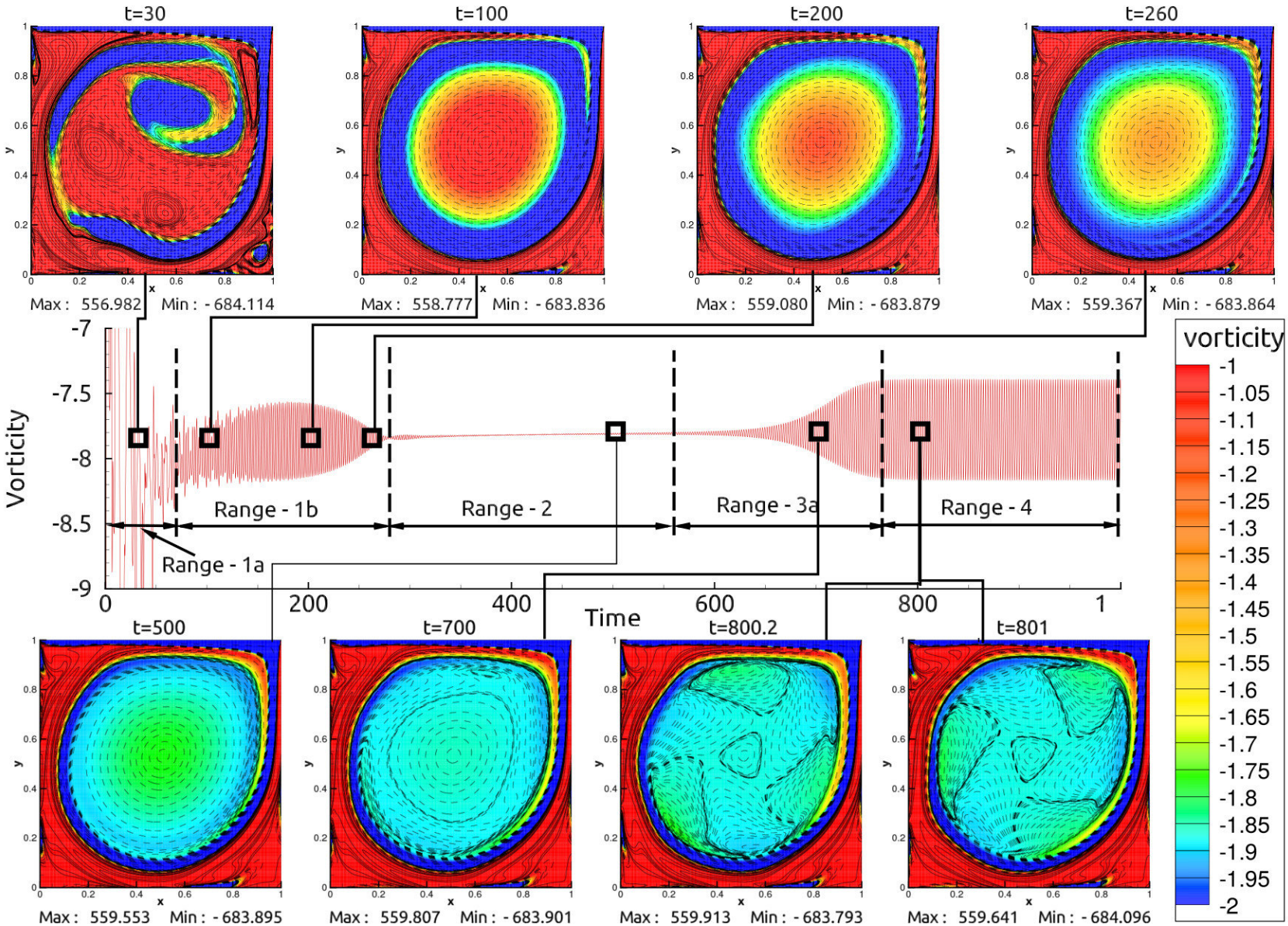
2D Singular Lid Driven Cavity



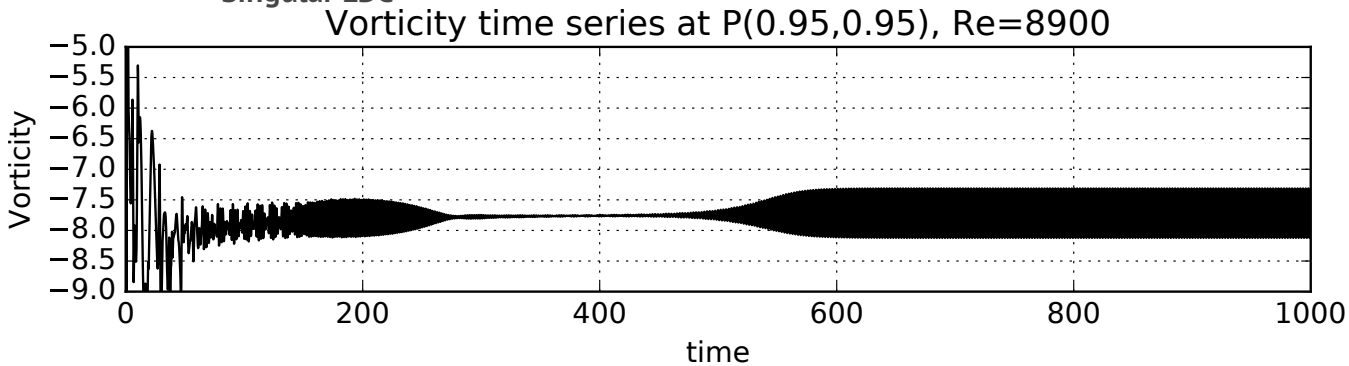
Numerical setup

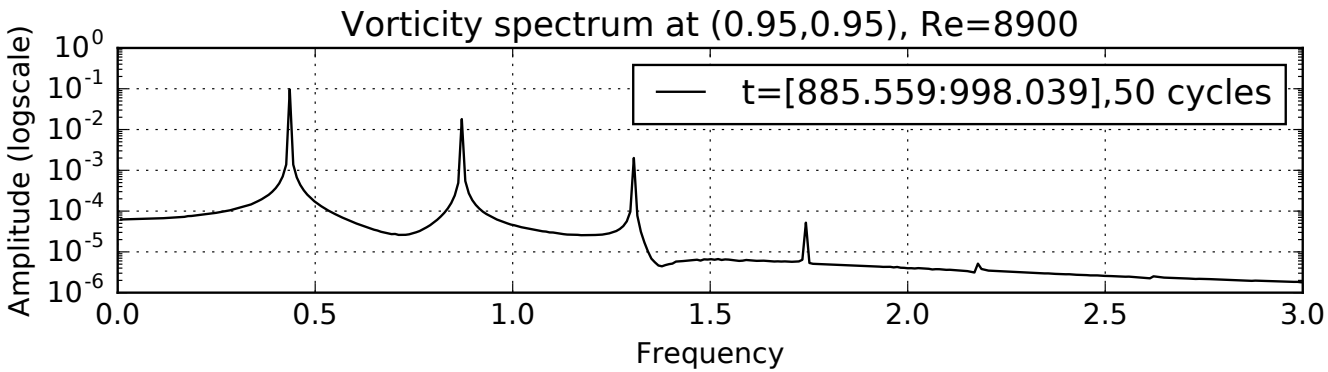
- $Re \in [8000, 12000]$
- Unstable flow with bifurcations
- 2D NS vorticity formulation
- High accuracy NCCD solver
- $257x257$ cartesian grid, $dt = 10^{-3}$
- Extremely sensitive problem

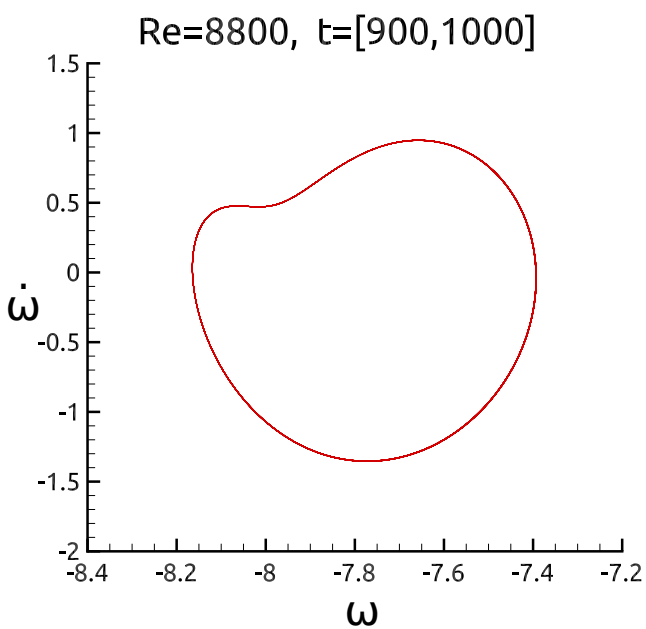
Lid driven cavity DNS at Re=8800

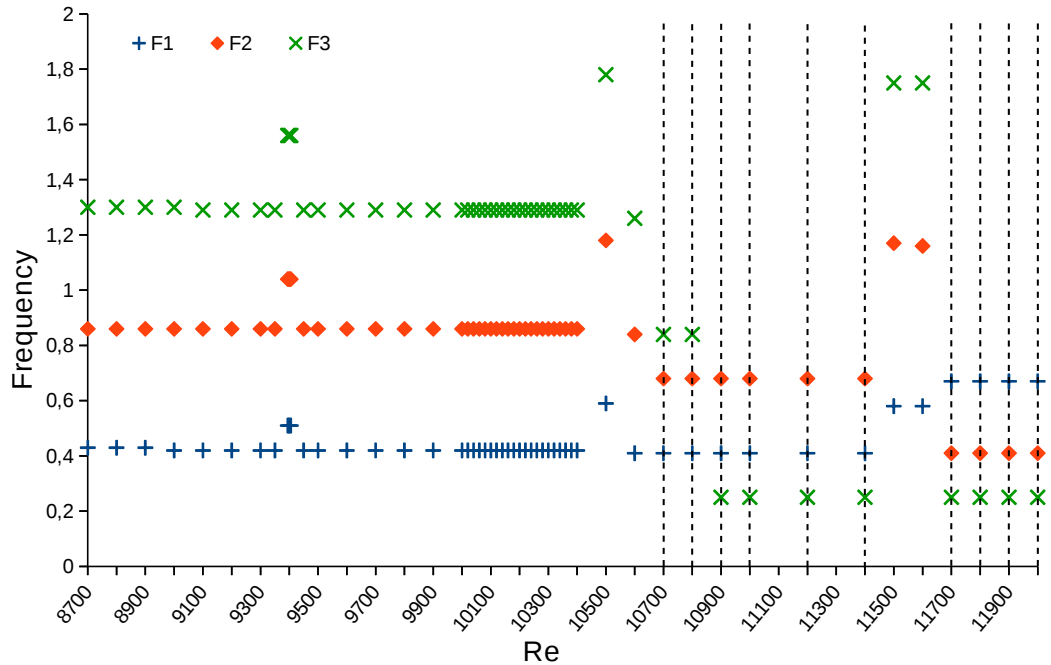


LDC Analysis

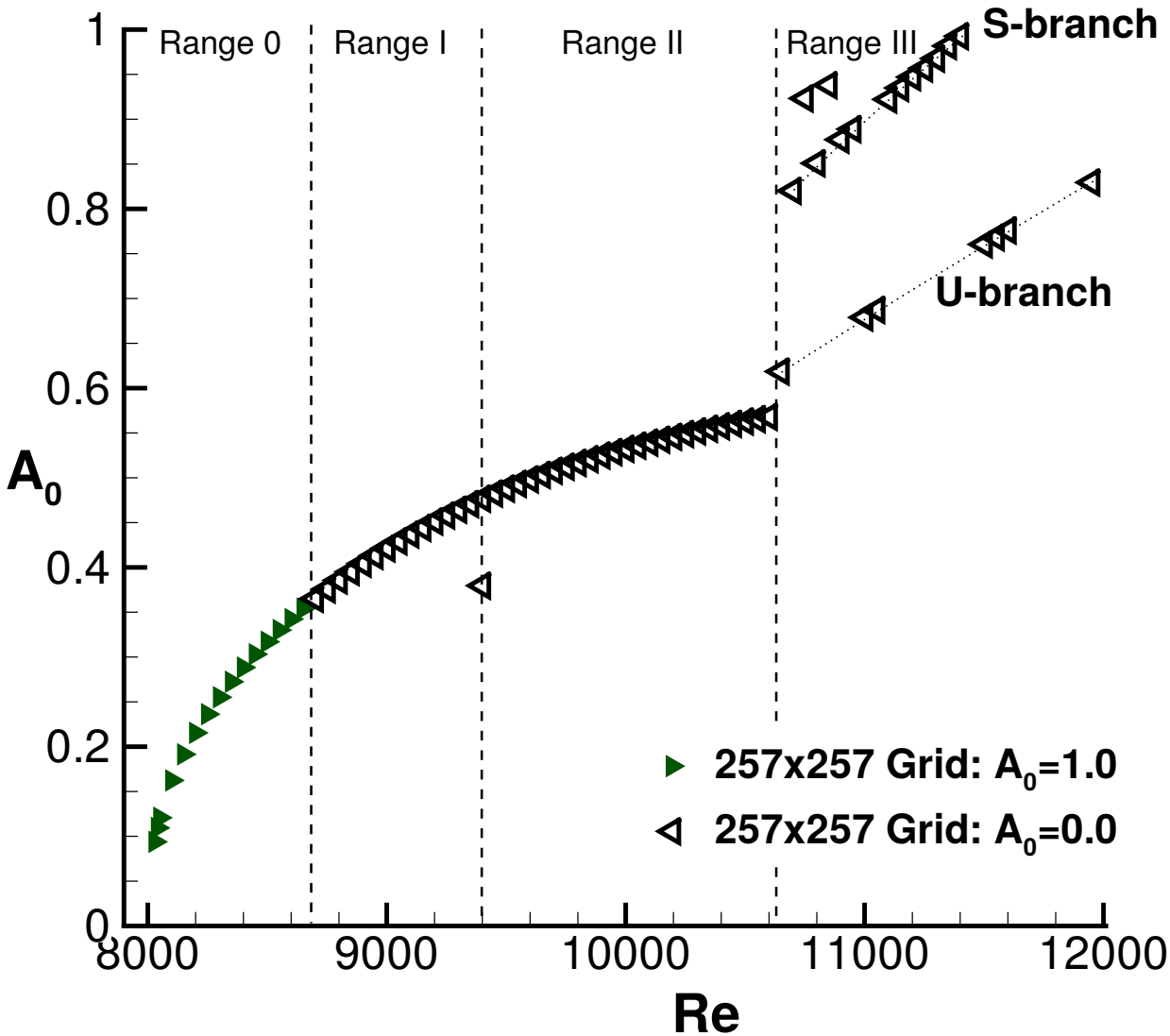




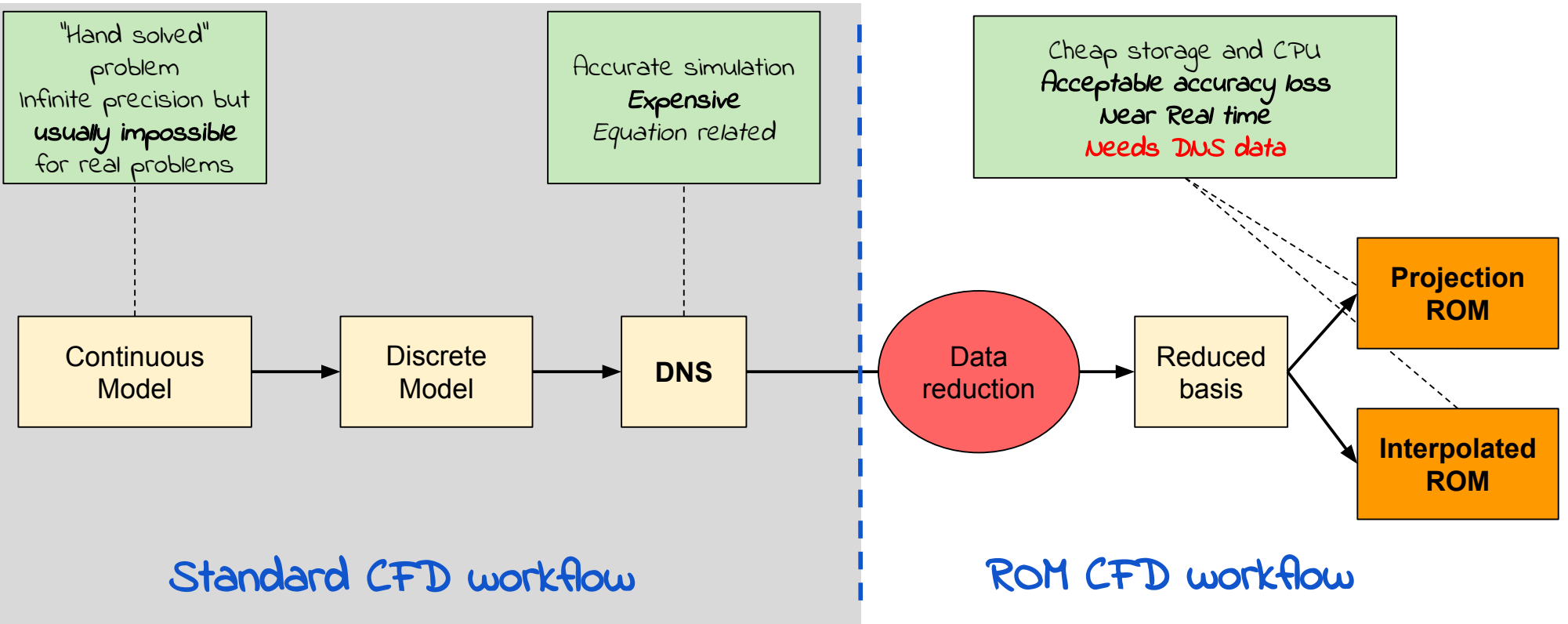




LDC Flow behavior : Hopf bifurcations

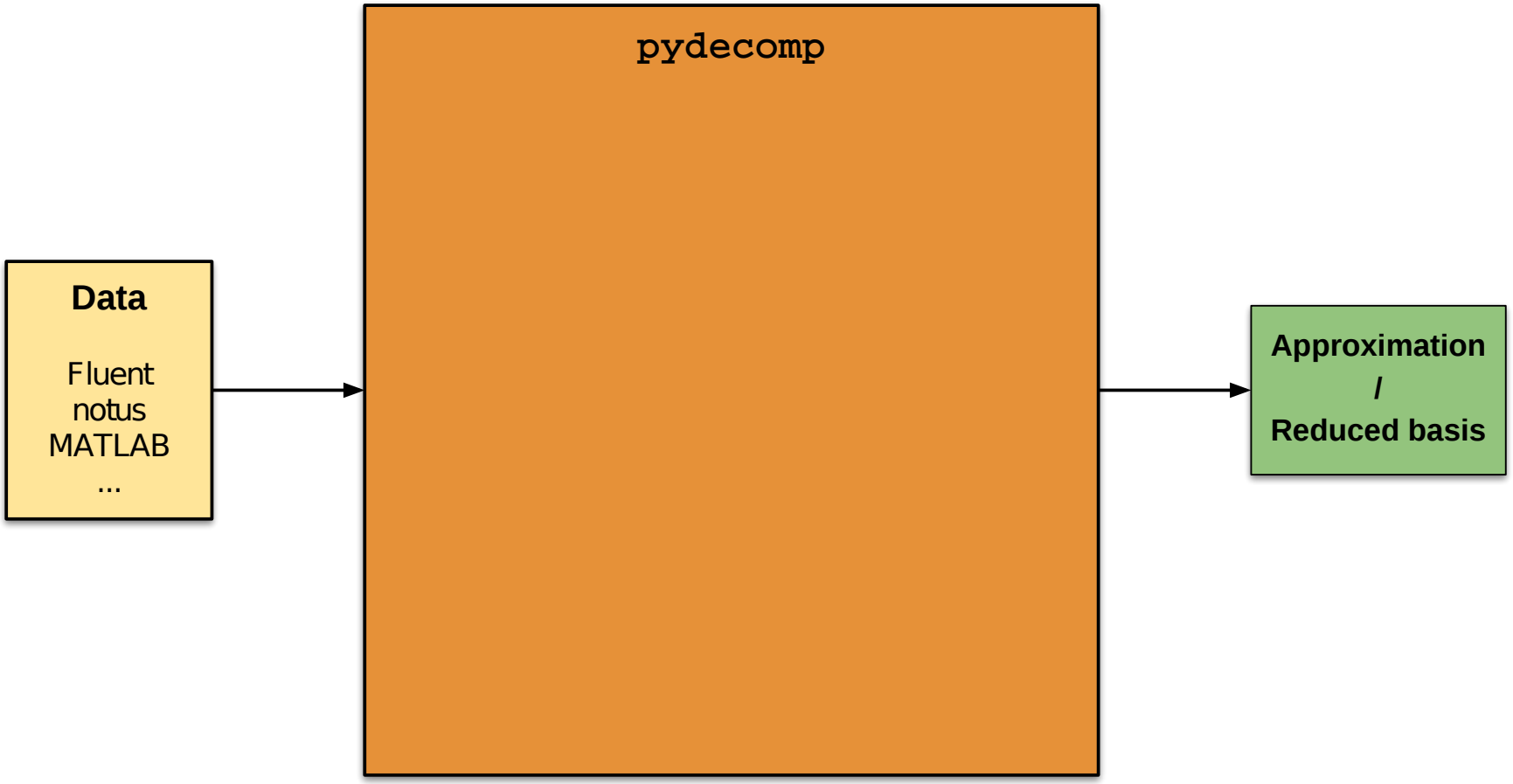


Data decomposition



Data decomposition

Build a library that computes data decomposition:



SVD, a General Matrix Decomposition

Theorem (Singular Value Decomposition)

For any matrix $A \in \mathbb{R}^{n \times m}$, there are orthogonal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ so that

$$A = U \Sigma V^\top$$

where Σ is a diagonal matrix of size $n \times m$ with diagonal elements $\sigma_{ii} \geq 0$ called singular values.

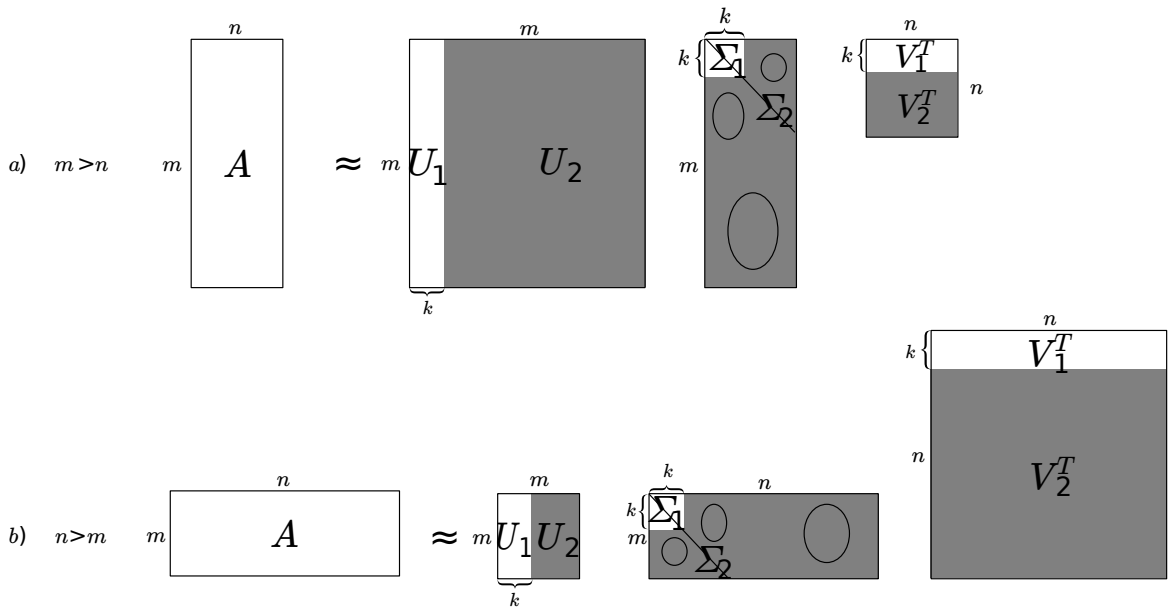
SVD, a General Matrix Decomposition

Theorem (Singular Value Decomposition)

For any matrix $A \in \mathbb{R}^{n\times m}$, there are orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ so that

$$A = U\Sigma V^\top$$

where Σ is a diagonal matrix of size $n \times m$ with diagonal elements $\sigma_{ii} \geq 0$ called singular values.



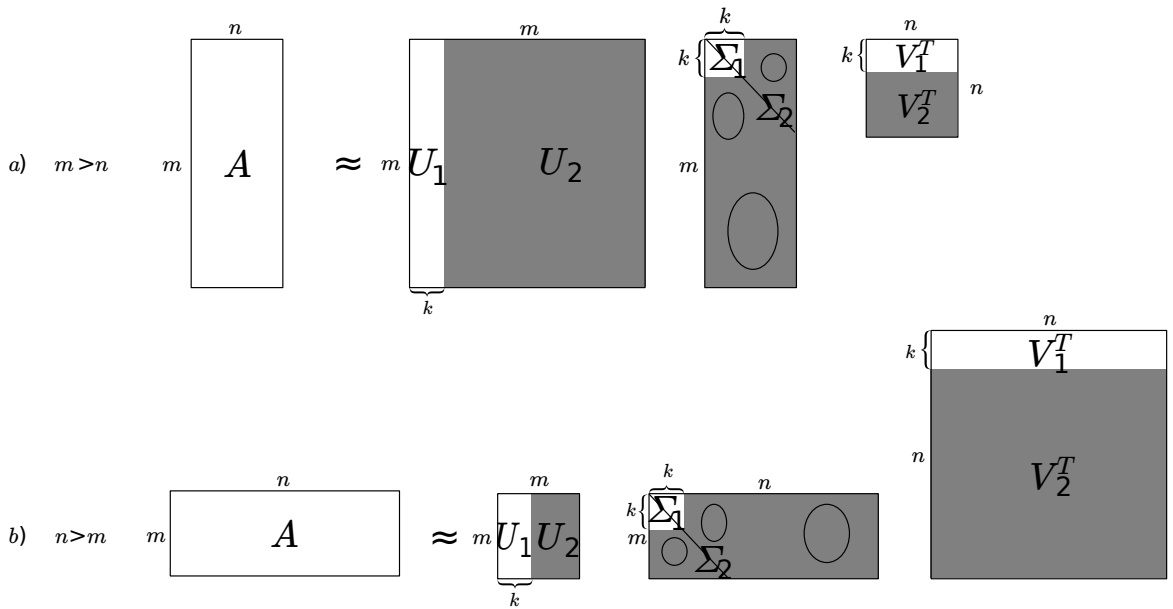
SVD, a General Matrix Decomposition

Theorem (Singular Value Decomposition)

For any matrix $A \in \mathbb{R}^{n\times m}$, there are orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ so that

$$A = U\Sigma V^\top$$

where Σ is a diagonal matrix of size $n \times m$ with diagonal elements $\sigma_{ii} \geq 0$ called singular values.



- **Eckart-Young theorem:** $\min_{rank(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}$
- The SVD can be computed by EVD.

SVD for image compression

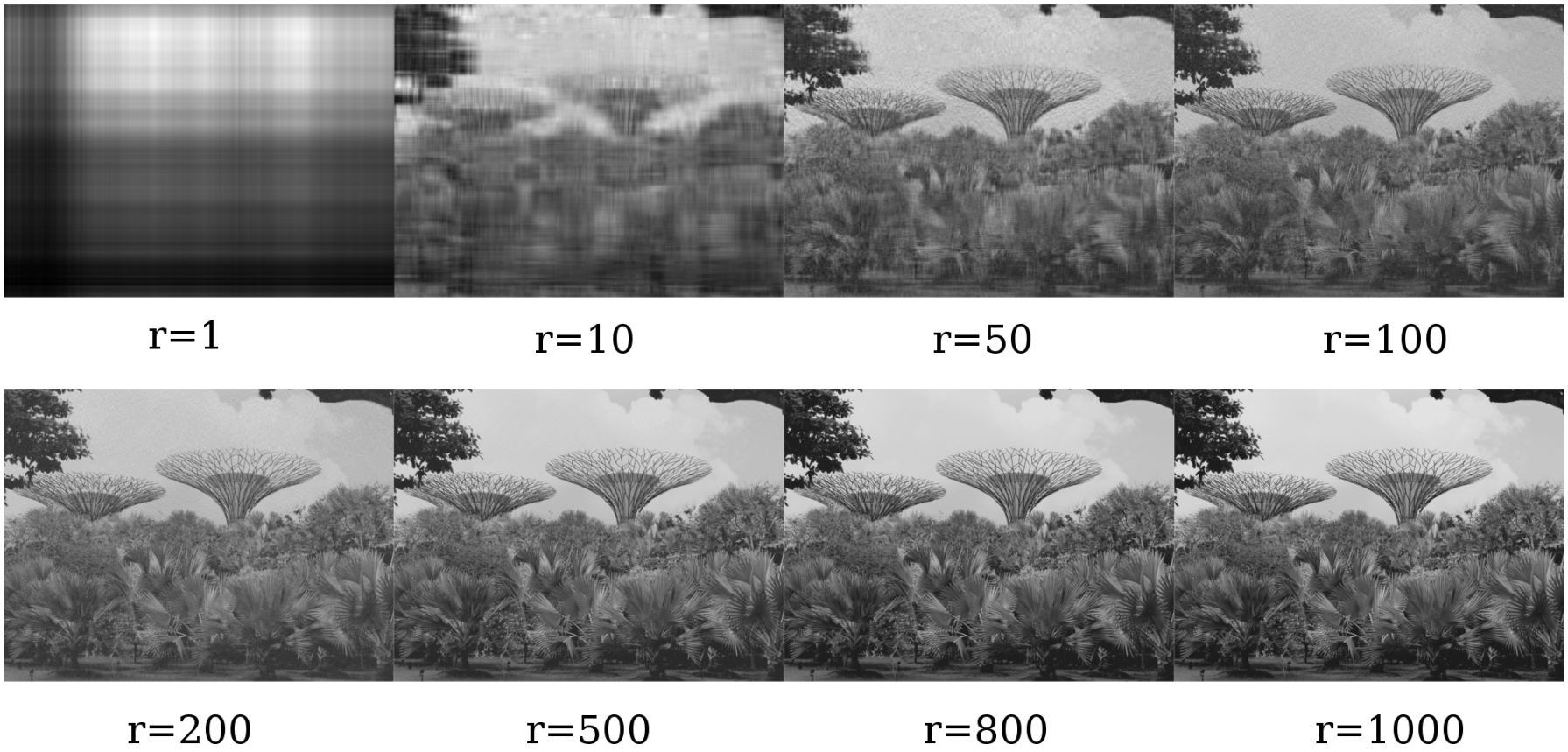
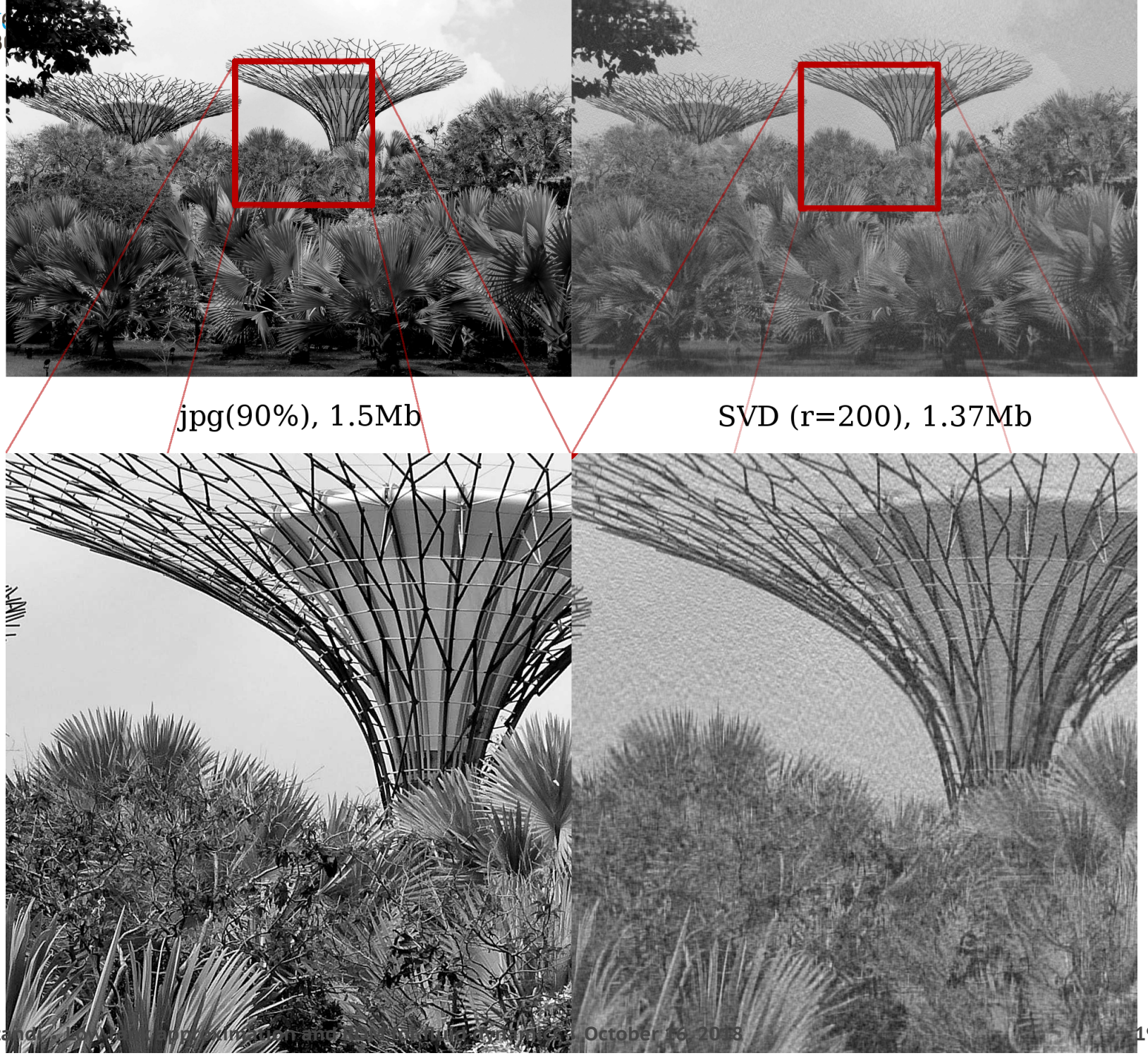


Figure: Singapore garden by the sea SVD reconstruction


jpg(90%), 1.5Mb
SVD (r=200), 1.37Mb

Proper Orthogonal Decomposition

POD

- **Goal:** Approximate $u(x,t)$ by a finite sum

$$u(x,t) \simeq u_K(x,t) = \sum_{k=1}^{K} a_k(t)\phi_k(x)$$

Proper Orthogonal Decomposition

POD

- **Goal:** Approximate $u(x,t)$ by a finite sum

$$u(x,t) \simeq u_K(x,t) = \sum_{k=1}^{K} a_k(t)\phi_k(x)$$

- Functional space approach of SVD
- Low-dimensional approximate of high-dimensional process

Properties

- Spectral theory : POD is solution of an eigenvalue problem

Proper Orthogonal Decomposition

POD

- **Goal:** Approximate $u(x,t)$ by a finite sum

$$u(x,t) \simeq u_K(x,t) = \sum_{k=1}^{K} a_k(t)\phi_k(x)$$

- Functional space approach of SVD
- Low-dimensional approximate of high-dimensional process

Properties

- Spectral theory : POD is solution of an eigenvalue problem
- POD bases are orthonormal

Proper Orthogonal Decomposition

POD

- **Goal:** Approximate $u(x,t)$ by a finite sum

$$u(x,t) \simeq u_K(x,t) = \sum_{k=1}^{K} a_k(t)\phi_k(x)$$

- Functional space approach of SVD
- Low-dimensional approximate of high-dimensional process

Properties

- Spectral theory : POD is solution of an eigenvalue problem
- POD bases are orthonormal
- Eckart-Young theorem

Proper Orthogonal Decomposition

POD

- **Goal:** Approximate $u(x,t)$ by a finite sum

$$u(x,t) \simeq u_K(x,t) = \sum_{k=1}^{K} a_k(t)\phi_k(x)$$

- Functional space approach of SVD
- Low-dimensional approximate of high-dimensional process

Properties

- Spectral theory : POD is solution of an eigenvalue problem
- POD bases are orthonormal
- Eckart-Young theorem
- POD bases are Optimal : $\|u - u_M\|_{L^2(X\times Y)} \leq \|u - s_M\|_{L^2(X\times Y)}. \quad \forall \text{ basis } s_M$

Proper Orthogonal Decomposition

POD

- **Goal:** Approximate $u(x,t)$ by a finite sum

$$u(x,t) \simeq u_K(x,t) = \sum_{k=1}^{K} a_k(t)\phi_k(x)$$

- Functional space approach of SVD
- Low-dimensional approximate of high-dimensional process

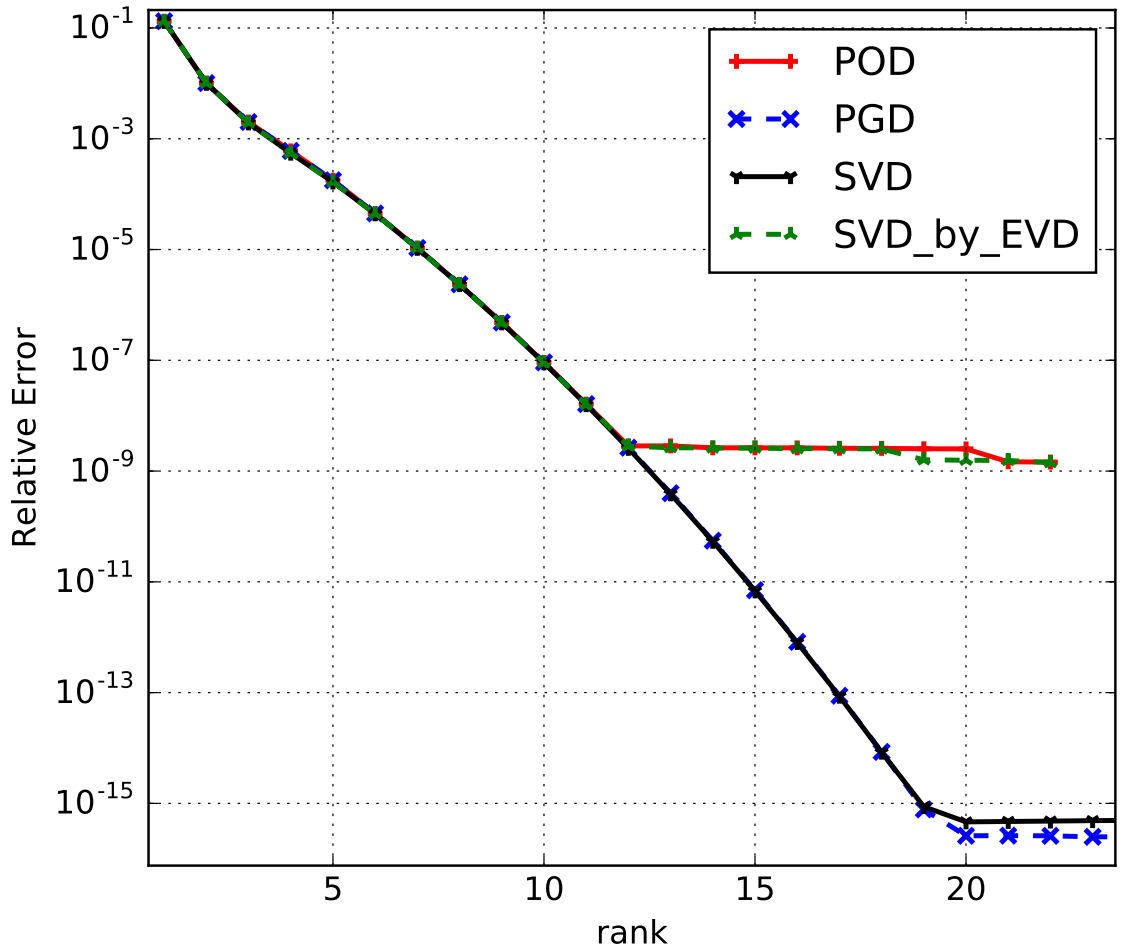
Properties

- Spectral theory : POD is solution of an eigenvalue problem
- POD bases are orthonormal
- Eckart-Young theorem
- POD bases are Optimal : $\|u - u_M\|_{L^2(X\times Y)} \leq \|u - s_M\|_{L^2(X\times Y)}. \quad \forall \text{ basis } s_M$
- Choice of the inner product : $l^2, L^2, H^1, \ldots$

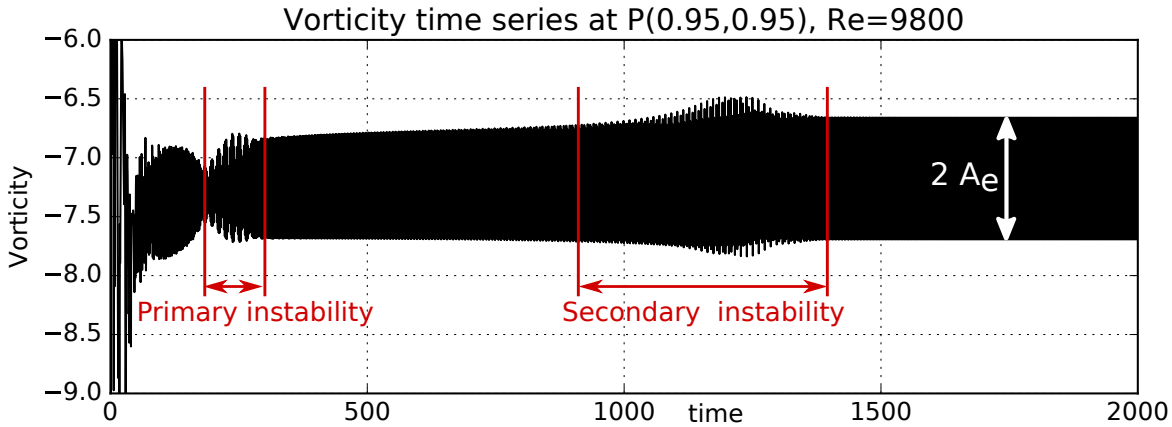
Example: synthetic data

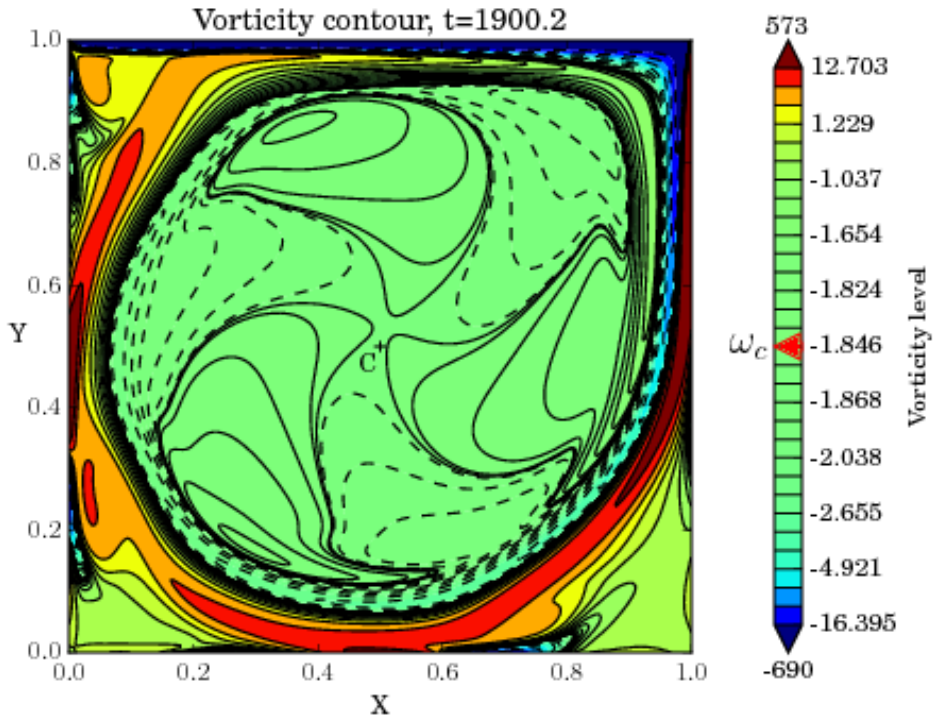
Let f_3 defined on $\Omega = [0,1] \times [0,1]$ with $f_3(x,y) = \sin(\sqrt{x^2+y^2})$

$$\mathcal{E} = \frac{||f_3 - \widetilde{f_3}||_{L^2}}{||f_3||_{L^2}}$$



Example: POD applied to Lid driven cavity (Re=9800)





LDC space modes

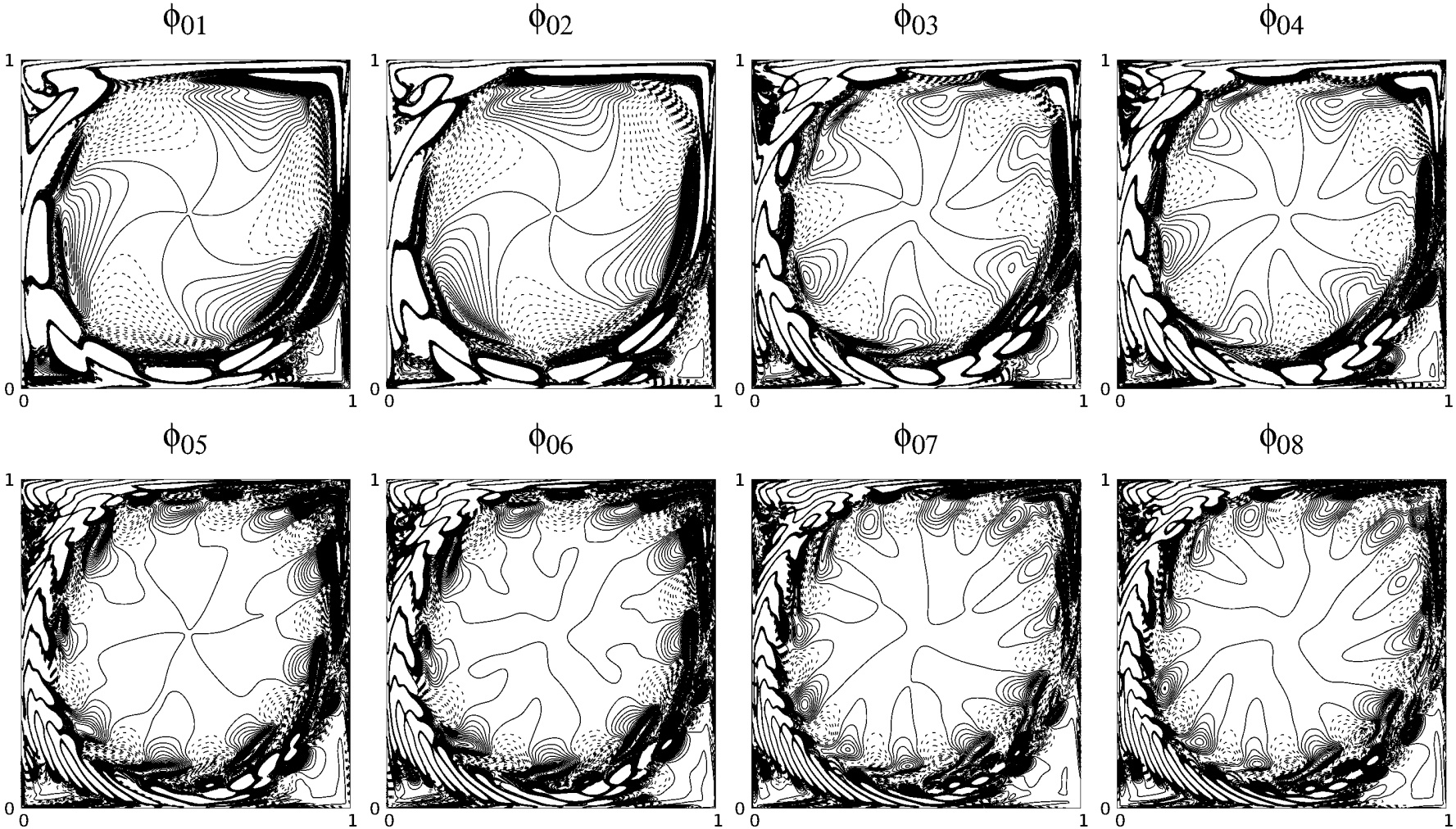


Figure: Re=9800, first 8 POD space modes

LDC time modes

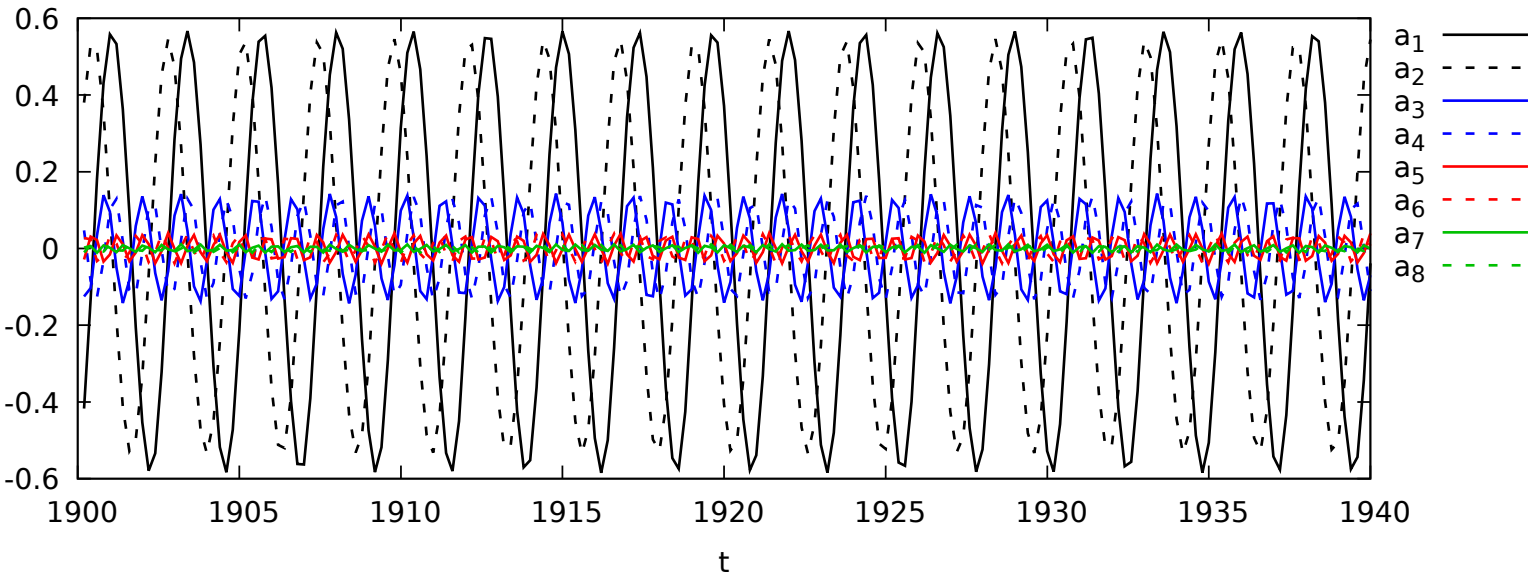
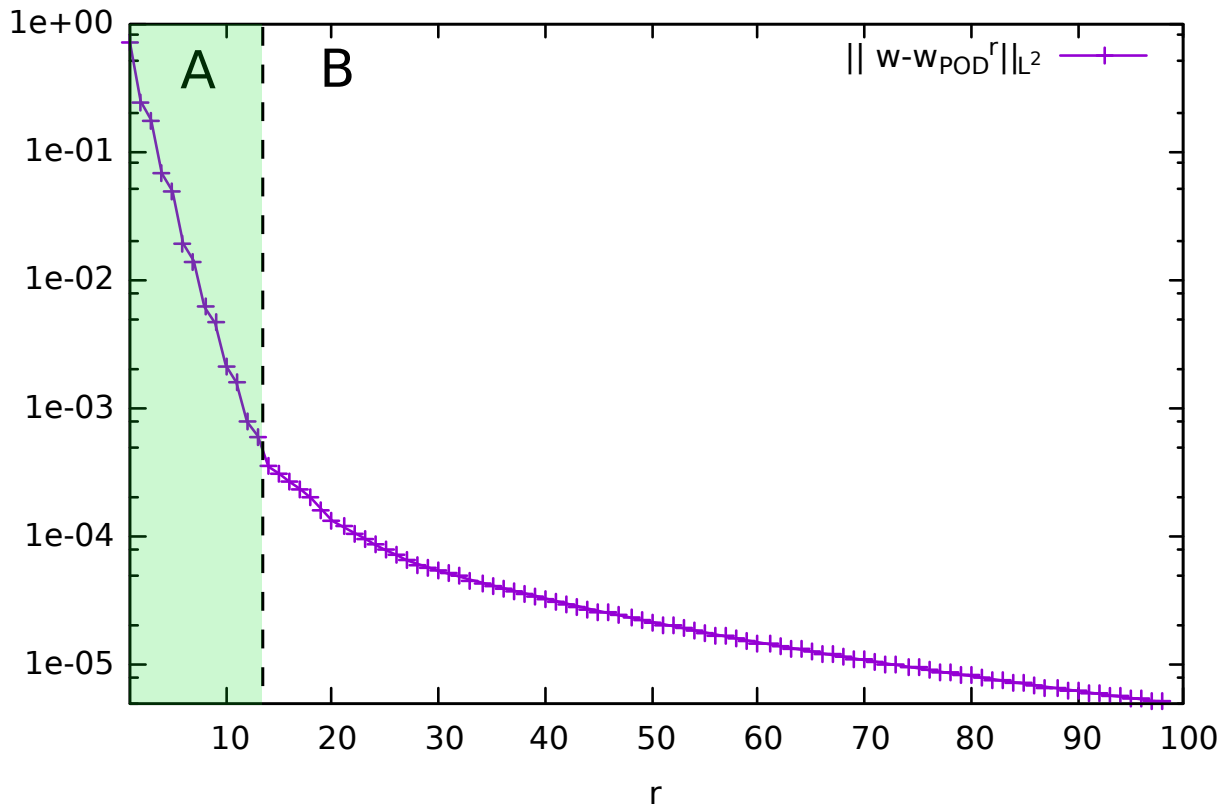


Figure: Re=9800, first 8 POD time modes

Error decay


1e+00
1e-01
1e-02
1e-03
1e-04
1e-05
A
B
|| w-w_POD^r||_L2
10
20
30
40
50
60
70
80
90
100
r

Tensors

1. A vector space special case. $E = U \otimes V$

Tensors

1. A vector space special case. $E = U \otimes V$
2. A generalisation of matrices to higher dimensions. i.e. $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.

Tensors

1. A vector space special case. $E = U \otimes V$
2. A generalisation of matrices to higher dimensions. i.e. $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.

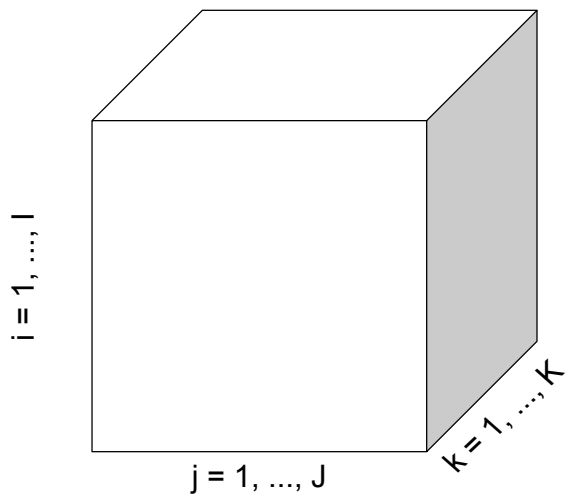


Figure: Order 3 tensor

Tensor related Key Properties

- **Tensor product,** $\otimes$:

$$\begin{aligned} \otimes : \quad \mathbb{R}^{\mathcal{I}} \times \mathbb{R}^{\mathcal{J}} &\to \quad \mathbb{R}^{\mathcal{I}\times\mathcal{J}} \\ (\mathcal{X}, \mathcal{Y}) &\mapsto \quad \mathcal{X} \otimes \mathcal{Y} \end{aligned}$$

Entry-wise $\mathcal{T} = \mathcal{X} \otimes \mathcal{Y}$ reads

$$T_{ij} = x_i y_j$$

Tensor related Key Properties

- **Tensor product,** $\otimes$:

$$\begin{aligned} \otimes : \quad \mathbb{R}^{\mathcal{I}} \times \mathbb{R}^{\mathcal{J}} &\to \quad \mathbb{R}^{\mathcal{I}\times\mathcal{J}} \\ (\mathcal{X}, \mathcal{Y}) &\mapsto \quad \mathcal{X} \otimes \mathcal{Y} \end{aligned}$$

  Entry-wise $\mathcal{T} = \mathcal{X} \otimes \mathcal{Y}$ reads

$$T_{ij} = x_i y_j$$

- **Tensor rank**: $rank(A)$ is the minimum number of rank one tensor that generates A.

Tensor related Key Properties

- **Tensor product,** $\otimes$:

$$\begin{aligned} \otimes : \quad \mathbb{R}^{\mathcal{I}} \times \mathbb{R}^{\mathcal{J}} &\to \quad \mathbb{R}^{\mathcal{I}\times\mathcal{J}} \\ (\mathcal{X}, \mathcal{Y}) &\mapsto \quad \mathcal{X} \otimes \mathcal{Y} \end{aligned}$$

  Entry-wise $\mathcal{T} = \mathcal{X} \otimes \mathcal{Y}$ reads

$$T_{ij} = x_i y_j$$

- **Tensor rank**: $rank(A)$ is the minimum number of rank one tensor that generates A.
- **Matricisation**: Ordering the elements of a tensor into a matrix.

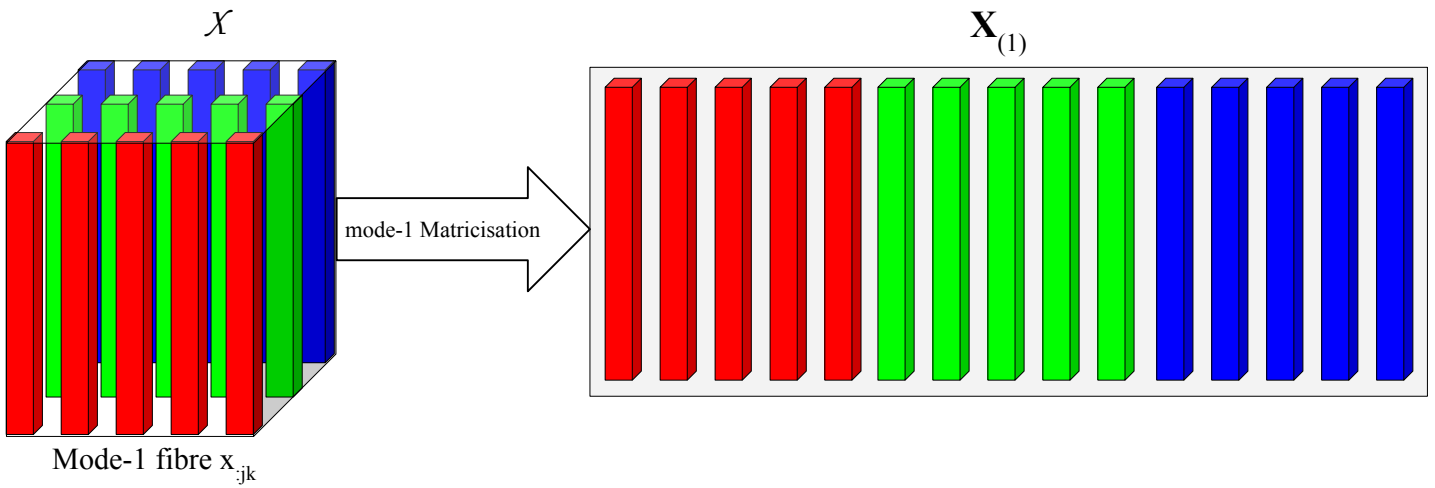


Figure: Mode one matricisation of third order tensor with $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$.

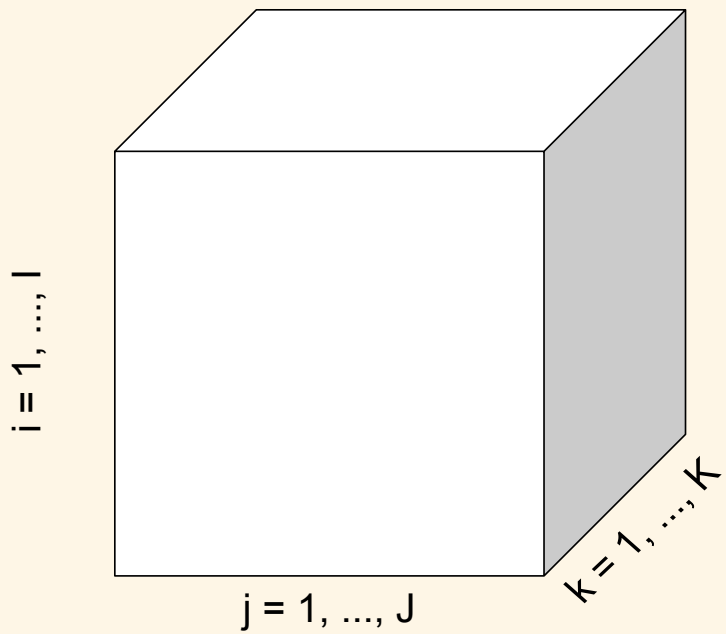
Full Tensor Format

A d-way array.

- High storage cost : $\mathcal{O}(n^d)$
- No evaluation cost

Can be used if n^d remains small

$$\mathcal{X} = \sum_{\boldsymbol{i} \in \mathcal{I}} x_{\boldsymbol{i}} \, \boldsymbol{e}_{1,i_1} \otimes \cdots \otimes \boldsymbol{e}_{d,i_d}$$



Canonical Tensor Format and CP Decomposition

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ is a finite sum of rank-1 tensors.

$$\mathcal{T} = \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} \boldsymbol{t}_{\mu,i} \quad \textit{where } \mathbf{t}_{\mu,i} \in V_\mu = \mathbb{R}^{n_\mu} \tag{1}$$

Canonical Tensor Format and CP Decomposition

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ is a finite sum of rank-1 tensors.

$$\mathcal{T} = \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} \boldsymbol{t}_{\mu,i} \quad \text{where } \mathbf{t}_{\mu,i} \in V_\mu = \mathbb{R}^{n_\mu} \tag{1}$$

- **Storage cost :** $\mathcal{O}(drn)$

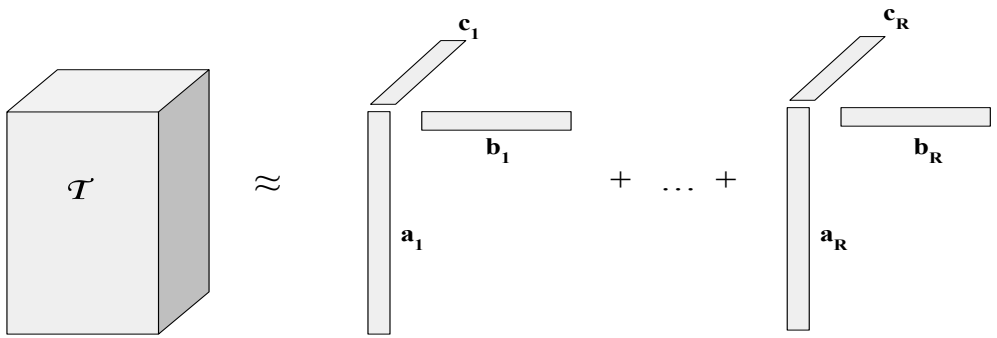


Figure: CP decomposition of third order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

Canonical Tensor Format and CP Decomposition

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ is a finite sum of rank-1 tensors.

$$\mathcal{T} = \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} \boldsymbol{t}_{\mu,i} \quad \text{where } \mathbf{t}_{\mu,i} \in V_\mu = \mathbb{R}^{n_\mu} \tag{1}$$

- Storage cost : $\mathcal{O}(drn)$
- **NP-hard problem to compute the rank of a tensor**

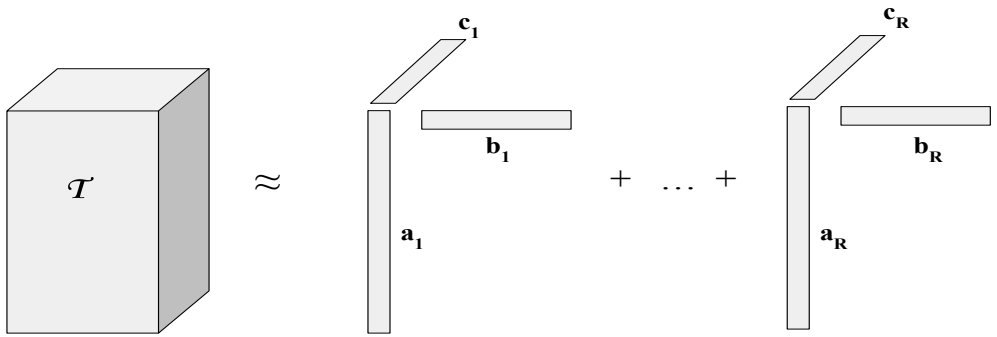


Figure: CP decomposition of third order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

Canonical Tensor Format and CP Decomposition

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ is a finite sum of rank-1 tensors.

$$\mathcal{T} = \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} \boldsymbol{t}_{\mu,i} \quad \text{where } \mathbf{t}_{\mu,i} \in V_\mu = \mathbb{R}^{n_\mu} \tag{1}$$

- Storage cost : $\mathcal{O}(drn)$
- NP-hard problem to compute the rank of a tensor
- **Find the best canonical decomposition: ill-posed problem.**

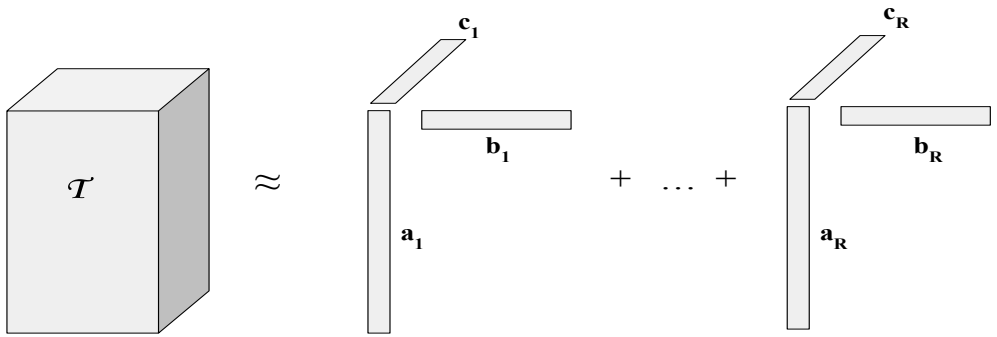


Figure: CP decomposition of third order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

Canonical Tensor Format and CP Decomposition

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ is a finite sum of rank-1 tensors.

$$\mathcal{T} = \sum_{i=1}^{r} \bigotimes_{\mu=1}^{d} \boldsymbol{t}_{\mu,i} \quad \textit{where } \mathbf{t}_{\mu,i} \in V_\mu = \mathbb{R}^{n_\mu} \tag{1}$$

- Storage cost : $\mathcal{O}(drn)$
- NP-hard problem to compute the rank of a tensor
- Find the best canonical decomposition: ill-posed problem.
- **Discrete PGD $\approx$ CP decomposition**

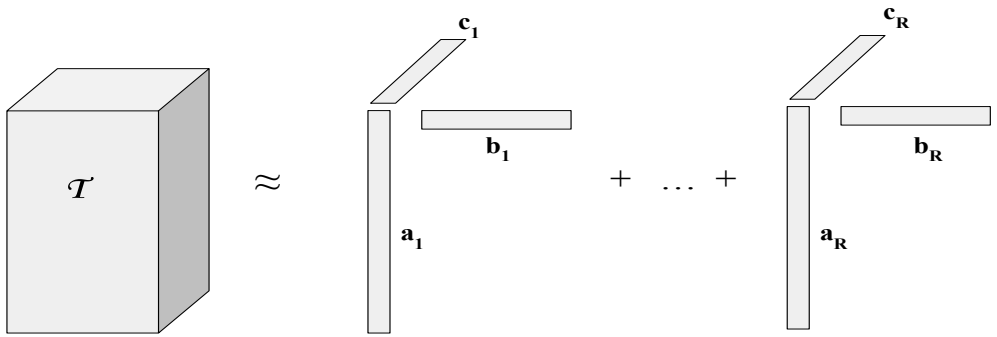


Figure: CP decomposition of third order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

Tucker Tensor Format

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ in Tucker format.

$$\mathcal{T} = \sum_{i_1=1}^{k_1} \cdots \sum_{i_d=1}^{k_d} w_{i_1,\ldots,i_d} u_{1,i_1} \otimes \cdots \otimes u_{d,i_d} \tag{2}$$

with the core tensor $\mathcal{W} \in \mathbb{R}^{k_1 \times \cdots \times k_d}$.
$\boldsymbol{k}$ is the tucker rank of $\mathcal{T}$.

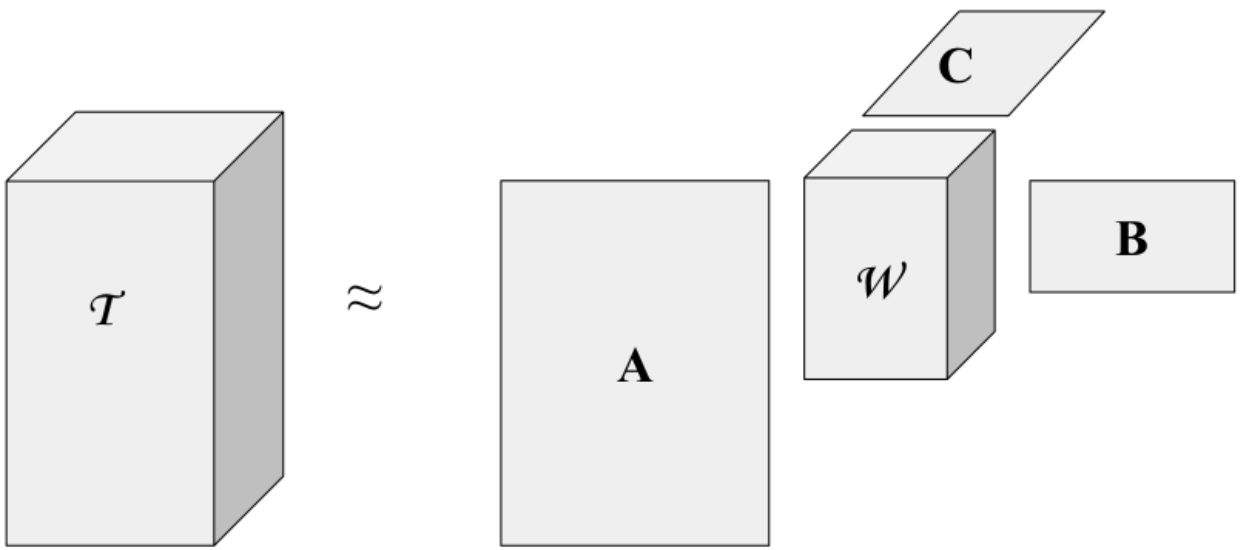
Tucker Tensor Format

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ in Tucker format.

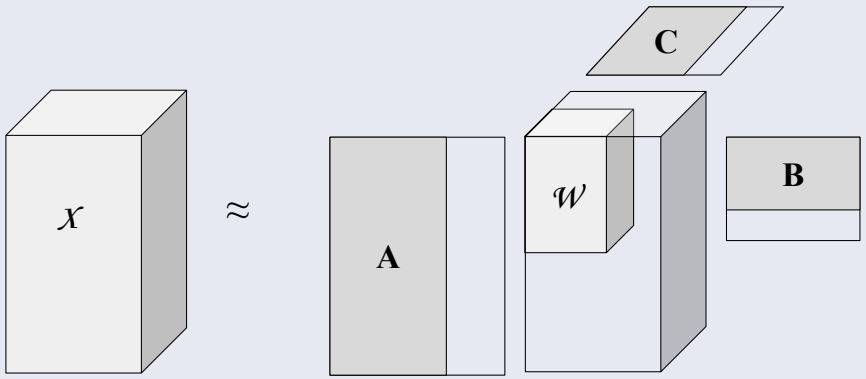
$$\mathcal{T} = \sum_{i_1=1}^{k_1} \cdots \sum_{i_d=1}^{k_d} w_{i_1,\ldots,i_d} u_{1,i_1} \otimes \cdots \otimes u_{d,i_d} \tag{2}$$

with the core tensor $\mathcal{W} \in \mathbb{R}^{k_1 \times \cdots \times k_d}$.
$\boldsymbol{k}$ is the tucker rank of $\mathcal{T}$.

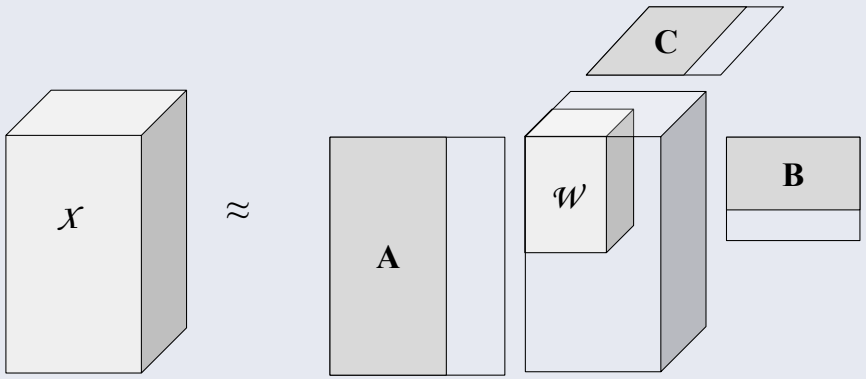
Storage cost : $\mathcal{O}(k^d + dkn) \Rightarrow$ intractable if d is large.



T-HOSVD

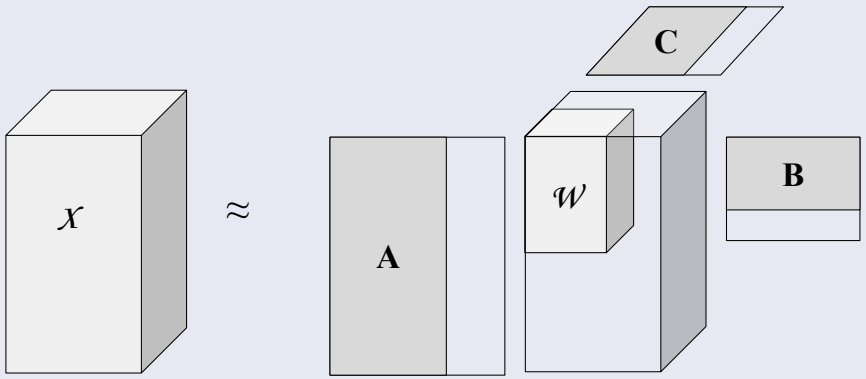


T-HOSVD



Idea Compute the singular vectors of each dimension separately.

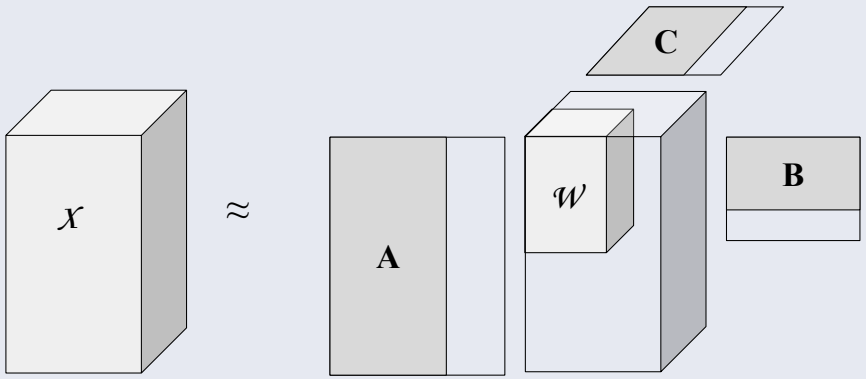
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$

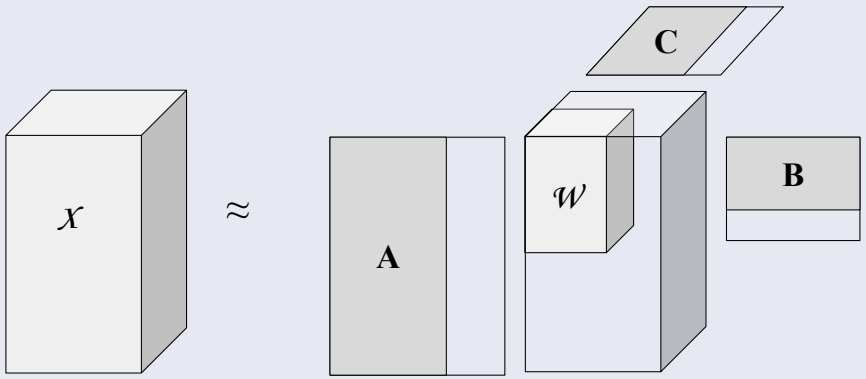
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \implies B.$

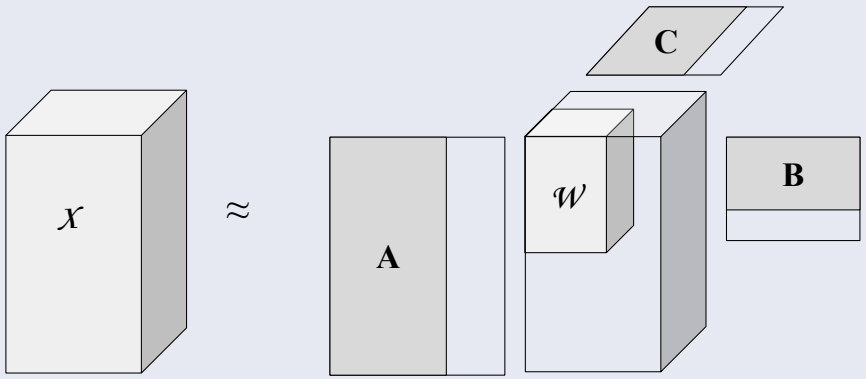
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \Longrightarrow A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \Longrightarrow B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \Longrightarrow C.$

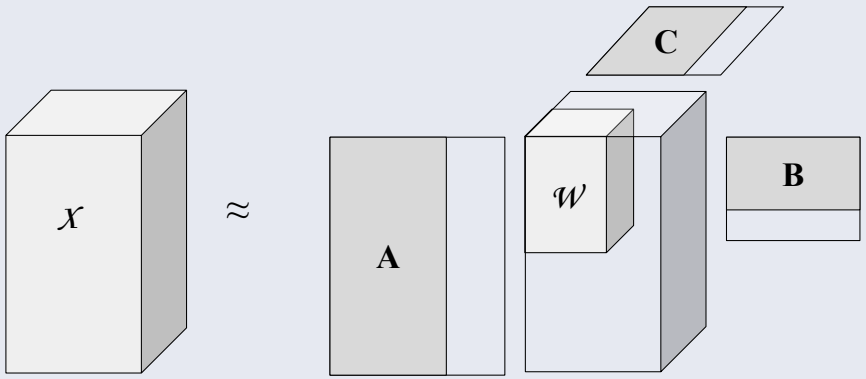
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \implies B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \implies C.$
4. **Projection** $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

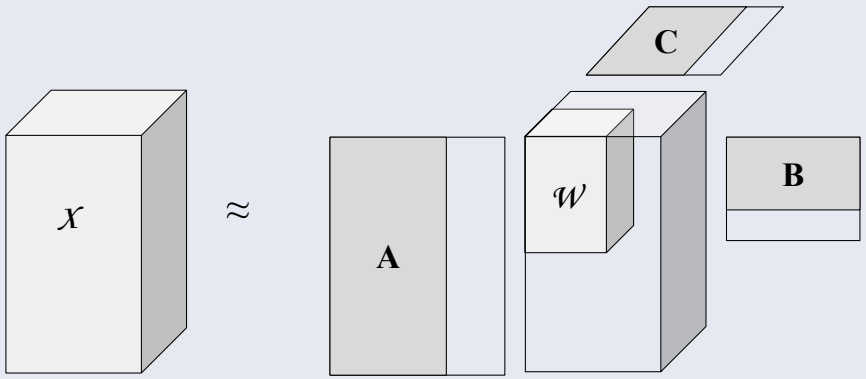
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \implies B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \implies C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

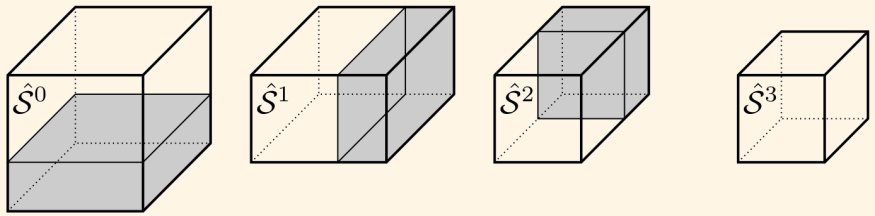
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

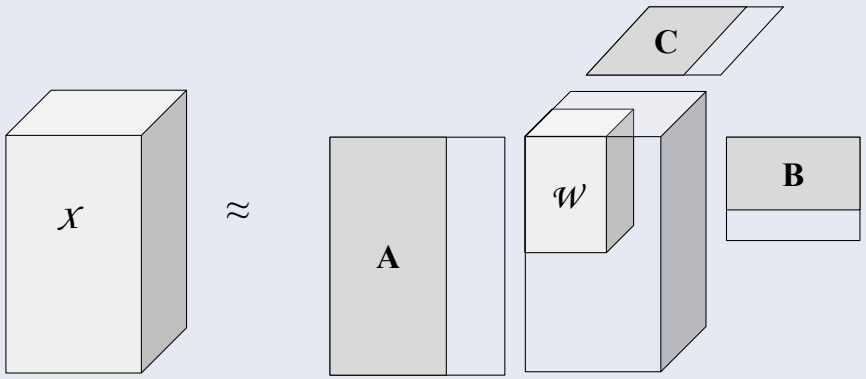
1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \implies B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \implies C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD



Goal Improve T-HOSVD

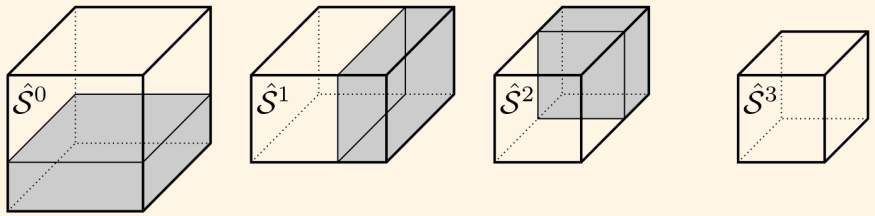
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \implies B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \implies C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

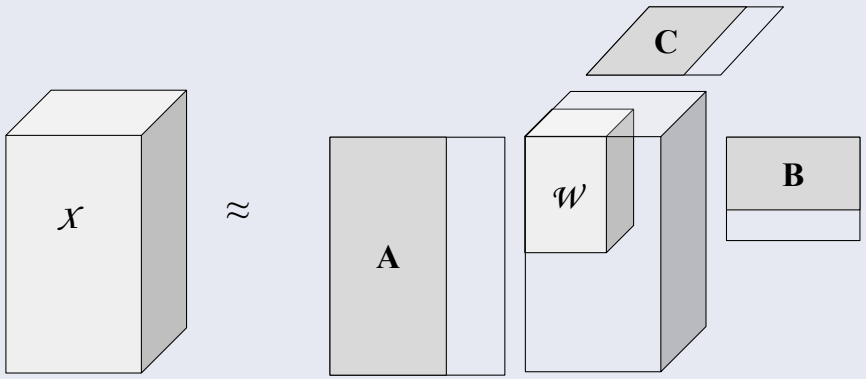
ST-HOSVD



Goal Improve T-HOSVD

Idea Use current approximation for next SVD

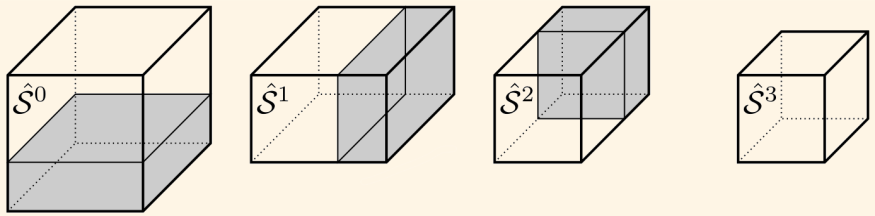
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \implies B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \implies C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD

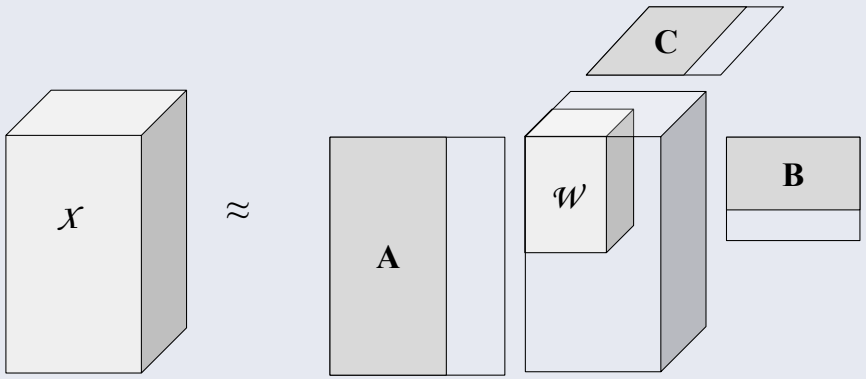


Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$

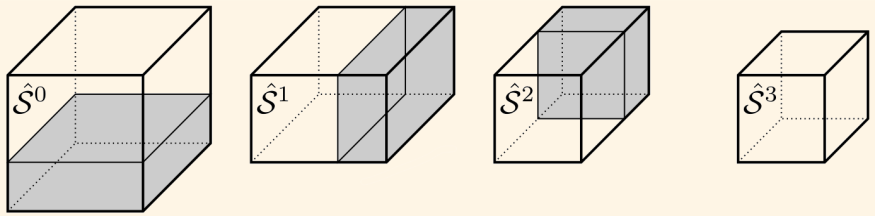
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \quad \Longrightarrow A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \quad \Longrightarrow B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \quad \Longrightarrow C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD

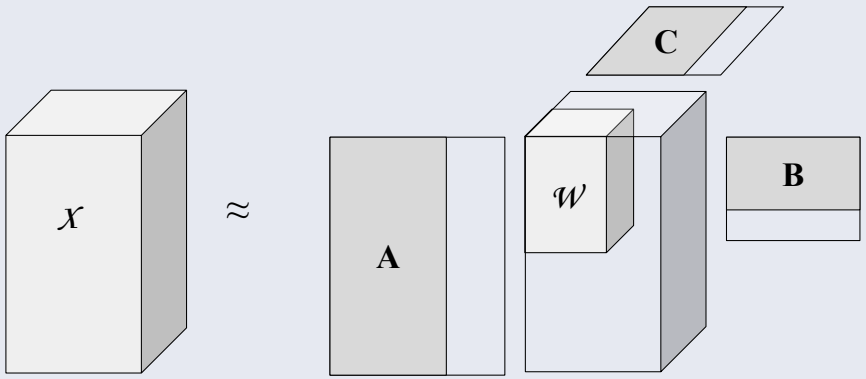


Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$
2. Truncated SVD : $\widehat{\mathcal{S}}_{(1)} \approx \widehat{U}_1 \widehat{\Sigma}_1 \widehat{V}_1^\intercal$

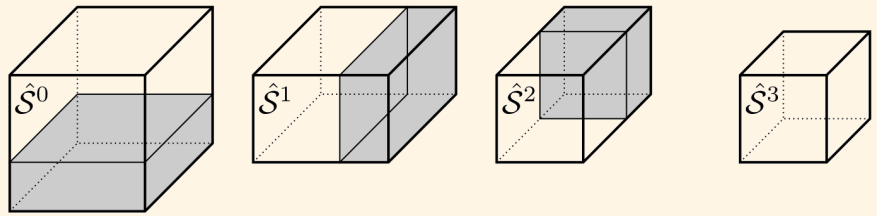
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \implies B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \implies C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD

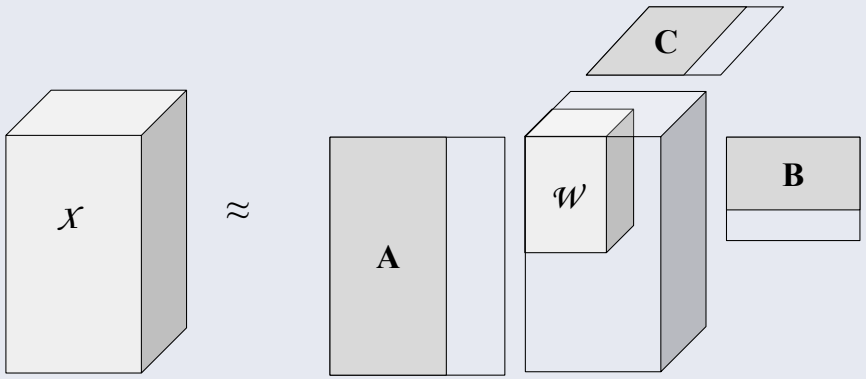


Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$
2. Truncated SVD : $\widehat{\mathcal{S}}_{(1)} \approx \widehat{U}_1 \widehat{\Sigma}_1 \widehat{V}_1^\intercal$
3. $\widehat{\mathcal{S}}_{(1)} = \widehat{\Sigma}_1 \widehat{V}_1^\intercal$; $\boldsymbol{A} = \widehat{U}_1$

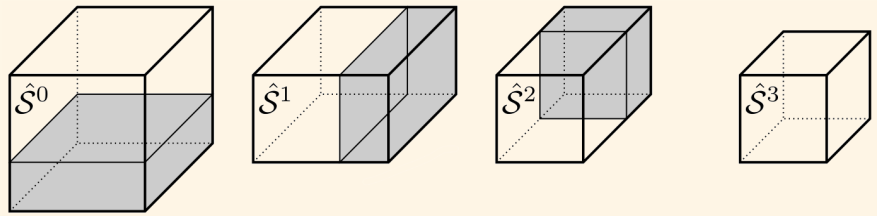
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \quad \Longrightarrow A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \quad \Longrightarrow B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \quad \Longrightarrow C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD

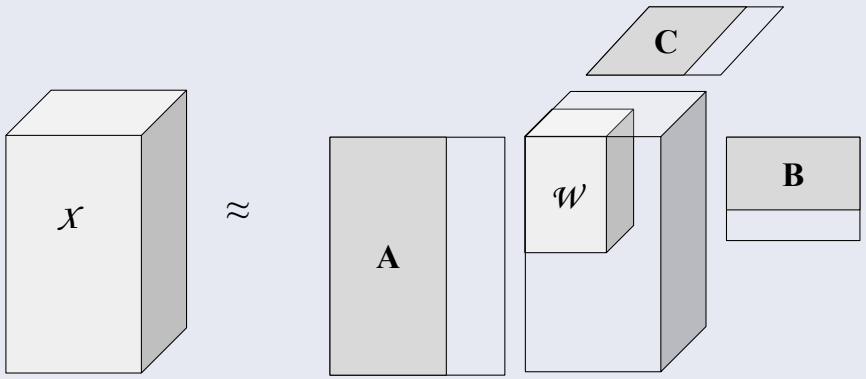


Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$
2. Truncated SVD : $\widehat{\mathcal{S}}_{(1)} \approx \widehat{U}_1 \widehat{\Sigma}_1 \widehat{V}_1^\intercal$
3. $\widehat{\mathcal{S}}_{(1)} = \widehat{\Sigma}_1 \widehat{V}_1^\intercal$; $\boldsymbol{A} = \widehat{U}_1$
4. Truncated SVD : $\widehat{\mathcal{S}}_{(2)} \approx \widehat{U}_2 \widehat{\Sigma}_2 \widehat{V}_2^\intercal$

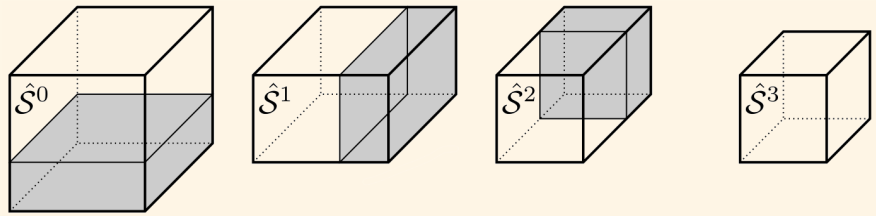
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \quad \Longrightarrow A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \quad \Longrightarrow B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \quad \Longrightarrow C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD

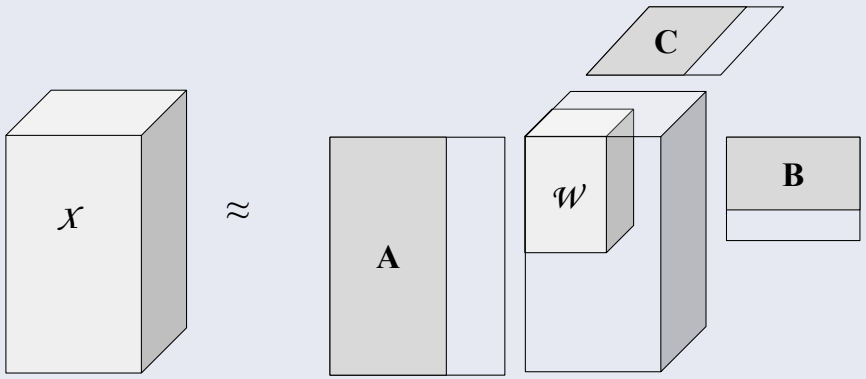


Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$
2. Truncated SVD : $\widehat{\mathcal{S}}_{(1)} \approx \widehat{U}_1 \widehat{\Sigma}_1 \widehat{V}_1^\intercal$
3. $\widehat{\mathcal{S}}_{(1)} = \widehat{\Sigma}_1 \widehat{V}_1^\intercal$; $\boldsymbol{A} = \widehat{U}_1$
4. Truncated SVD : $\widehat{\mathcal{S}}_{(2)} \approx \widehat{U}_2 \widehat{\Sigma}_2 \widehat{V}_2^\intercal$
5. $\widehat{\mathcal{S}}_{(2)} = \widehat{\Sigma}_2 \widehat{V}_2^\intercal$; $\boldsymbol{B} = \widehat{U}_2$

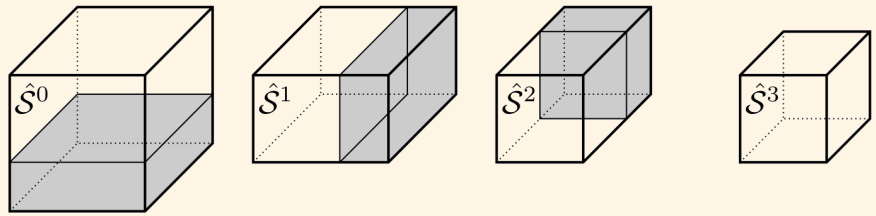
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \quad \Longrightarrow A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \quad \Longrightarrow B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \quad \Longrightarrow C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD

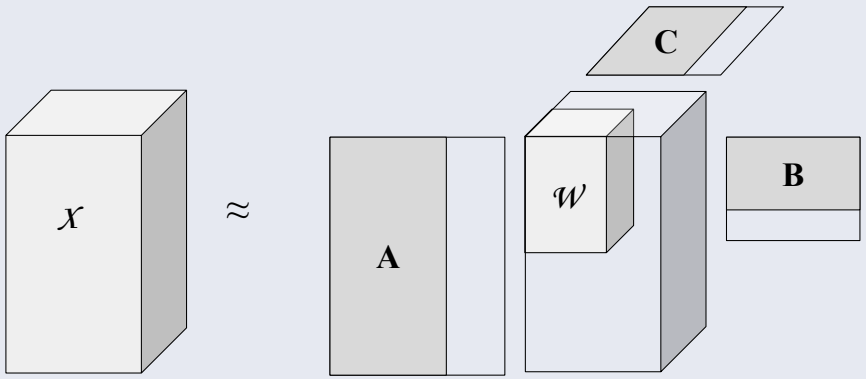


Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$
2. Truncated SVD : $\widehat{\mathcal{S}}_{(1)} \approx \widehat{U}_1 \widehat{\Sigma}_1 \widehat{V}_1^\intercal$
3. $\widehat{\mathcal{S}}_{(1)} = \widehat{\Sigma}_1 \widehat{V}_1^\intercal ; \boldsymbol{A} = \widehat{U}_1$
4. Truncated SVD : $\widehat{\mathcal{S}}_{(2)} \approx \widehat{U}_2 \widehat{\Sigma}_2 \widehat{V}_2^\intercal$
5. $\widehat{\mathcal{S}}_{(2)} = \widehat{\Sigma}_2 \widehat{V}_2^\intercal ; \boldsymbol{B} = \widehat{U}_2$
6. Truncated SVD : $\widehat{\mathcal{S}}_{(3)} \approx \widehat{U}_3 \widehat{\Sigma}_3 \widehat{V}_3^\intercal$

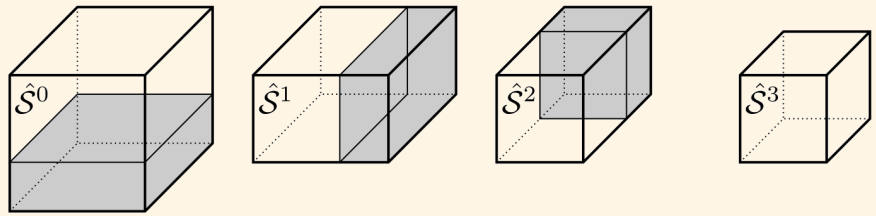
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \quad \Longrightarrow A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \quad \Longrightarrow B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \quad \Longrightarrow C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD

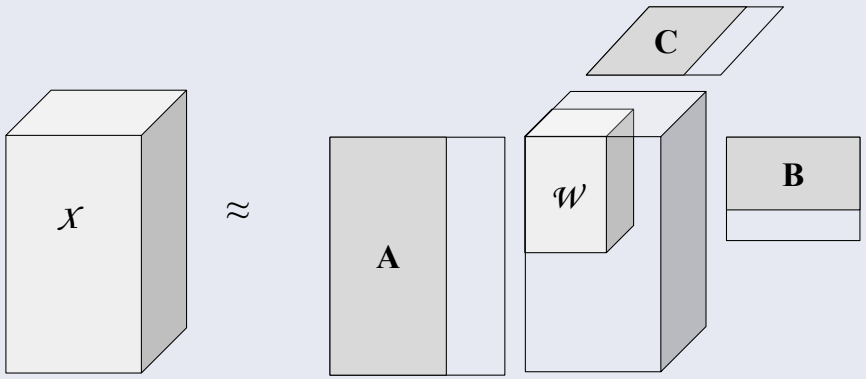


Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$
2. Truncated SVD : $\widehat{\mathcal{S}}_{(1)} \approx \widehat{U}_1 \widehat{\Sigma}_1 \widehat{V}_1^\intercal$
3. $\widehat{\mathcal{S}}_{(1)} = \widehat{\Sigma}_1 \widehat{V}_1^\intercal ; \boldsymbol{A} = \widehat{U}_1$
4. Truncated SVD : $\widehat{\mathcal{S}}_{(2)} \approx \widehat{U}_2 \widehat{\Sigma}_2 \widehat{V}_2^\intercal$
5. $\widehat{\mathcal{S}}_{(2)} = \widehat{\Sigma}_2 \widehat{V}_2^\intercal ; \boldsymbol{B} = \widehat{U}_2$
6. Truncated SVD : $\widehat{\mathcal{S}}_{(3)} \approx \widehat{U}_3 \widehat{\Sigma}_3 \widehat{V}_3^\intercal$
7. $\mathcal{W}_{(3)} = \widehat{\Sigma}_3 \widehat{V}_3^\intercal ; \boldsymbol{C} = \widehat{U}_3$

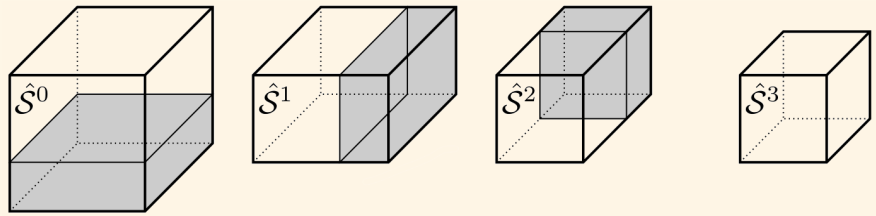
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \quad \Longrightarrow A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \quad \Longrightarrow B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \quad \Longrightarrow C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD

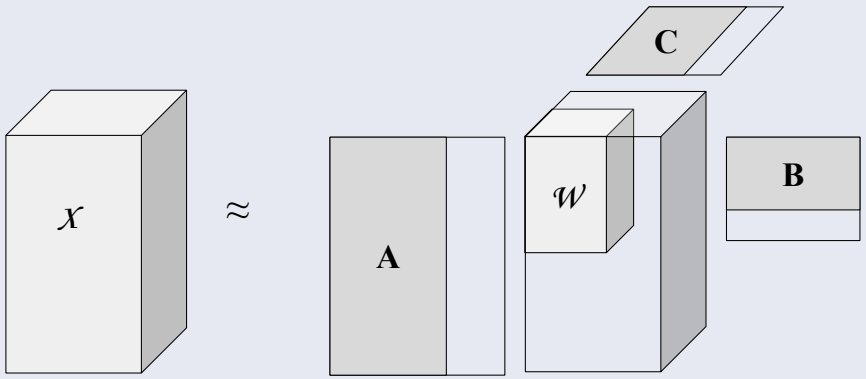


Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$
2. Truncated SVD : $\widehat{\mathcal{S}}_{(1)} \approx \widehat{U}_1 \widehat{\Sigma}_1 \widehat{V}_1^\intercal$
3. $\widehat{\mathcal{S}}_{(1)} = \widehat{\Sigma}_1 \widehat{V}_1^\intercal ; \boldsymbol{A} = \widehat{U}_1$
4. Truncated SVD : $\widehat{\mathcal{S}}_{(2)} \approx \widehat{U}_2 \widehat{\Sigma}_2 \widehat{V}_2^\intercal$
5. $\widehat{\mathcal{S}}_{(2)} = \widehat{\Sigma}_2 \widehat{V}_2^\intercal ; \boldsymbol{B} = \widehat{U}_2$
6. Truncated SVD : $\widehat{\mathcal{S}}_{(3)} \approx \widehat{U}_3 \widehat{\Sigma}_3 \widehat{V}_3^\intercal$
7. $\mathcal{W}_{(3)} = \widehat{\Sigma}_3 \widehat{V}_3^\intercal ; \boldsymbol{C} = \widehat{U}_3$

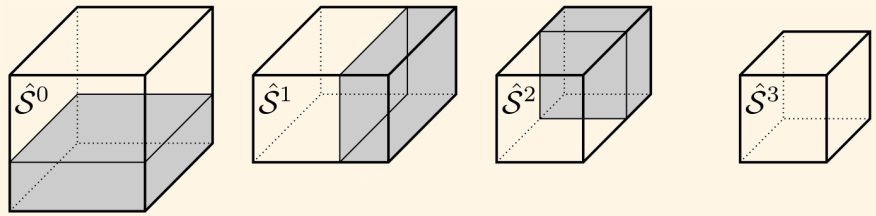
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \implies B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \implies C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD



Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$
2. Truncated SVD : $\widehat{\mathcal{S}}_{(1)} \approx \widehat{U}_1 \widehat{\Sigma}_1 \widehat{V}_1^\intercal$
3. $\widehat{\mathcal{S}}_{(1)} = \widehat{\Sigma}_1 \widehat{V}_1^\intercal ; \boldsymbol{A} = \widehat{U}_1$
4. Truncated SVD : $\widehat{\mathcal{S}}_{(2)} \approx \widehat{U}_2 \widehat{\Sigma}_2 \widehat{V}_2^\intercal$
5. $\widehat{\mathcal{S}}_{(2)} = \widehat{\Sigma}_2 \widehat{V}_2^\intercal ; \boldsymbol{B} = \widehat{U}_2$
6. Truncated SVD : $\widehat{\mathcal{S}}_{(3)} \approx \widehat{U}_3 \widehat{\Sigma}_3 \widehat{V}_3^\intercal$
7. $\mathcal{W}_{(3)} = \widehat{\Sigma}_3 \widehat{V}_3^\intercal ; \boldsymbol{C} = \widehat{U}_3$

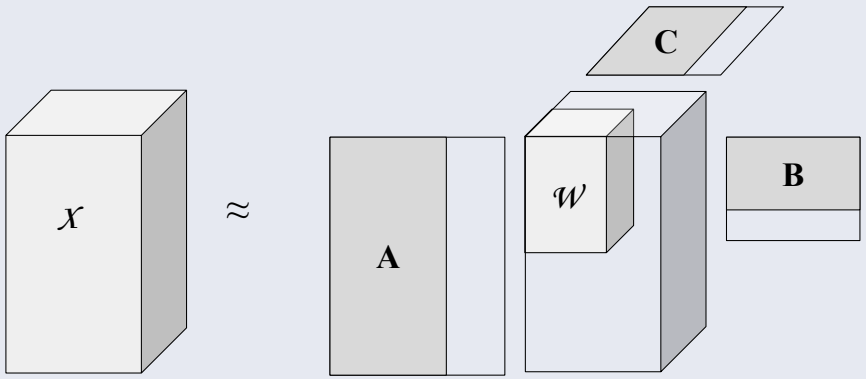
- Very efficient CPU-wise

Shared properties :

Pros Easy to implement, *almost* best approximation i.e. $\varepsilon \leq \sum_{\mu=1}^{d} \|\tilde{\Sigma}_\mu\|_F^2$

Cons $\mathcal{W}$ may become big.

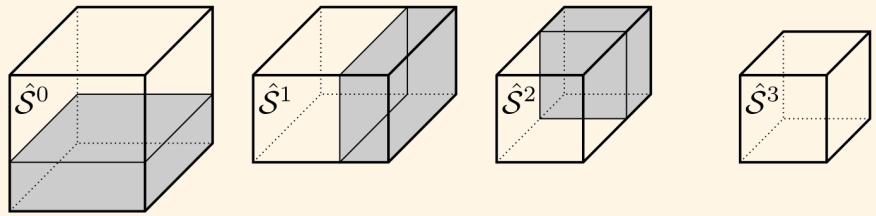
T-HOSVD



Idea Compute the **singular vectors** of each dimension separately.

1. $SVD(\mathcal{M}_1(\mathcal{X})) \implies A.$
2. $SVD(\mathcal{M}_2(\mathcal{X})) \implies B.$
3. $SVD(\mathcal{M}_3(\mathcal{X})) \implies C.$
4. Projection $w_{ijk} = \langle \mathcal{X}, \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k \rangle$

ST-HOSVD



Goal Improve T-HOSVD

Idea Use current approximation for next SVD

1. $\widehat{\mathcal{S}} = \mathcal{T}$
2. Truncated SVD : $\widehat{\mathcal{S}}_{(1)} \approx \widehat{U}_1 \widehat{\Sigma}_1 \widehat{V}_1^\intercal$
3. $\widehat{\mathcal{S}}_{(1)} = \widehat{\Sigma}_1 \widehat{V}_1^\intercal ; \boldsymbol{A} = \widehat{U}_1$
4. Truncated SVD : $\widehat{\mathcal{S}}_{(2)} \approx \widehat{U}_2 \widehat{\Sigma}_2 \widehat{V}_2^\intercal$
5. $\widehat{\mathcal{S}}_{(2)} = \widehat{\Sigma}_2 \widehat{V}_2^\intercal ; \boldsymbol{B} = \widehat{U}_2$
6. Truncated SVD : $\widehat{\mathcal{S}}_{(3)} \approx \widehat{U}_3 \widehat{\Sigma}_3 \widehat{V}_3^\intercal$
7. $\mathcal{W}_{(3)} = \widehat{\Sigma}_3 \widehat{V}_3^\intercal ; \boldsymbol{C} = \widehat{U}_3$

- Very efficient CPU-wise
- As accurate as T-HOSVD

Shared properties :

Pros Easy to implement, *almost* best approximation i.e. $\varepsilon \leq \sum_{\mu=1}^{d} \|\tilde{\Sigma}_\mu\|_F^2$

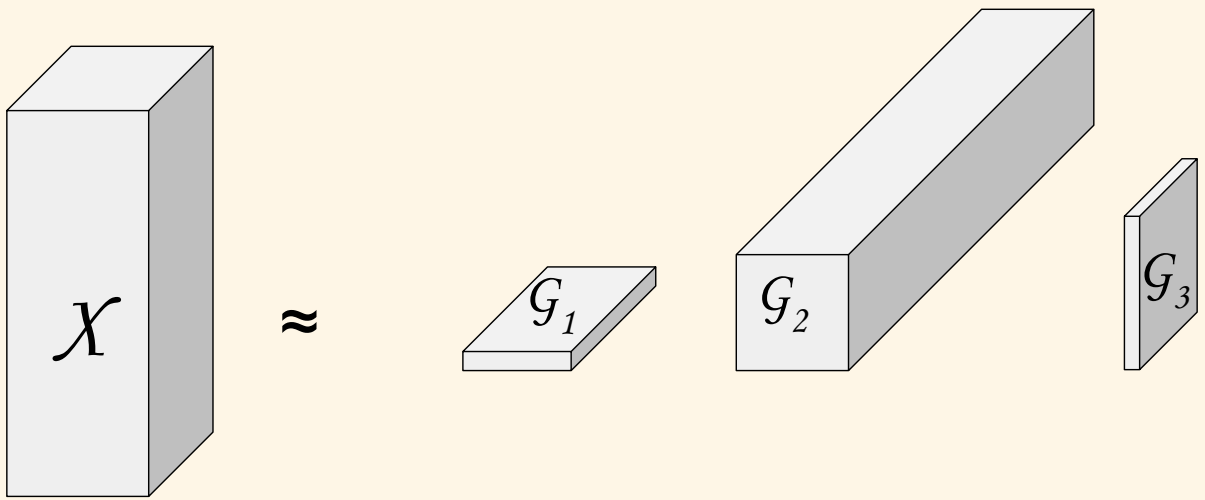
Cons $\mathcal{W}$ may become big.

Tensor train

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ in TT format.

$x_{i_1,\ldots,i_d} = \boldsymbol{G}_1(i_1)\boldsymbol{G}_2(i_2)\cdots\boldsymbol{G}_d(i_d)$, with $\boldsymbol{G}_\mu \in \mathbb{R}^{k_{\mu-1}\times k_\mu}$

Storage cost (TT) : $\mathcal{O}(k^2 dn)$



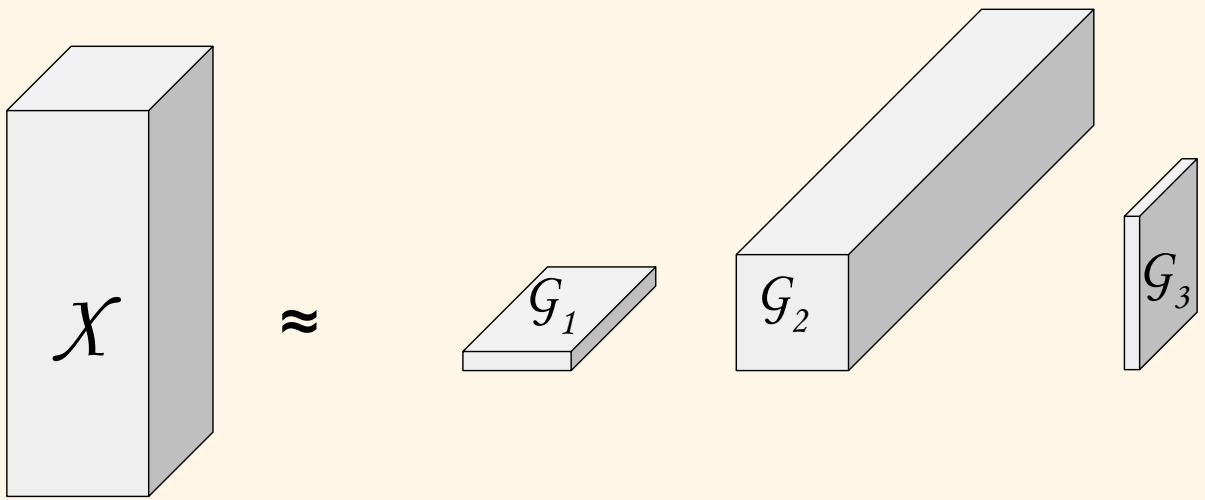
Goal Get rid of the core tensor.

Tensor train

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ in TT format.

$x_{i_1,\ldots,i_d} = \boldsymbol{G}_1(i_1)\boldsymbol{G}_2(i_2)\cdots\boldsymbol{G}_d(i_d)$, with $\boldsymbol{G}_\mu \in \mathbb{R}^{k_{\mu-1}\times k_\mu}$

Storage cost (TT) : $\mathcal{O}(k^2 dn)$



Goal Get rid of the core tensor.

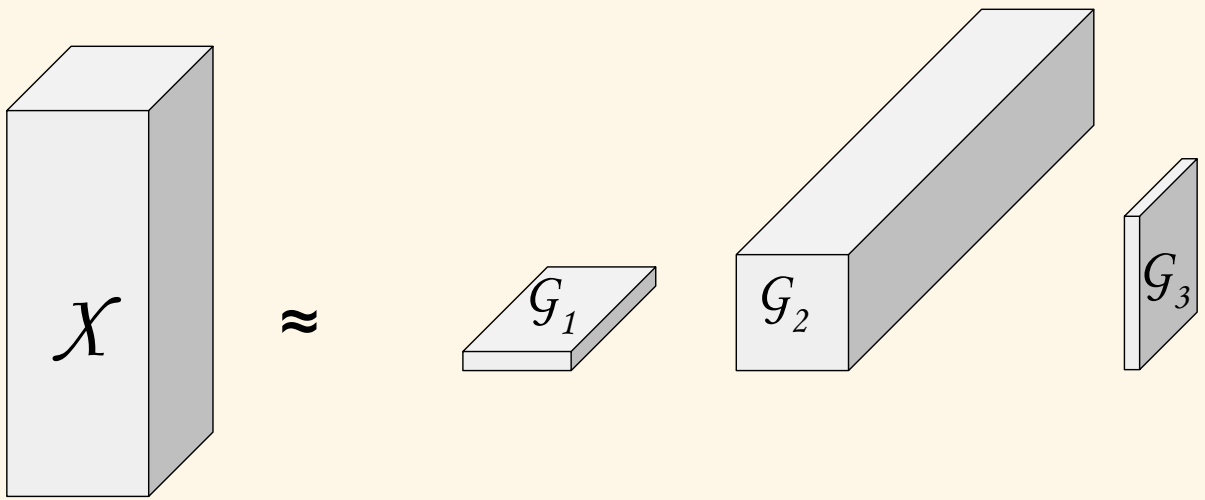
Idea **Encode the modes and their relations into matrix products (MPS) for each element.**

Tensor train

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ in TT format.

$x_{i_1,\ldots,i_d} = \boldsymbol{G}_1(i_1)\boldsymbol{G}_2(i_2)\cdots\boldsymbol{G}_d(i_d)$, with $\boldsymbol{G}_\mu \in \mathbb{R}^{k_{\mu-1}\times k_\mu}$

Storage cost (TT) : $\mathcal{O}(k^2 dn)$



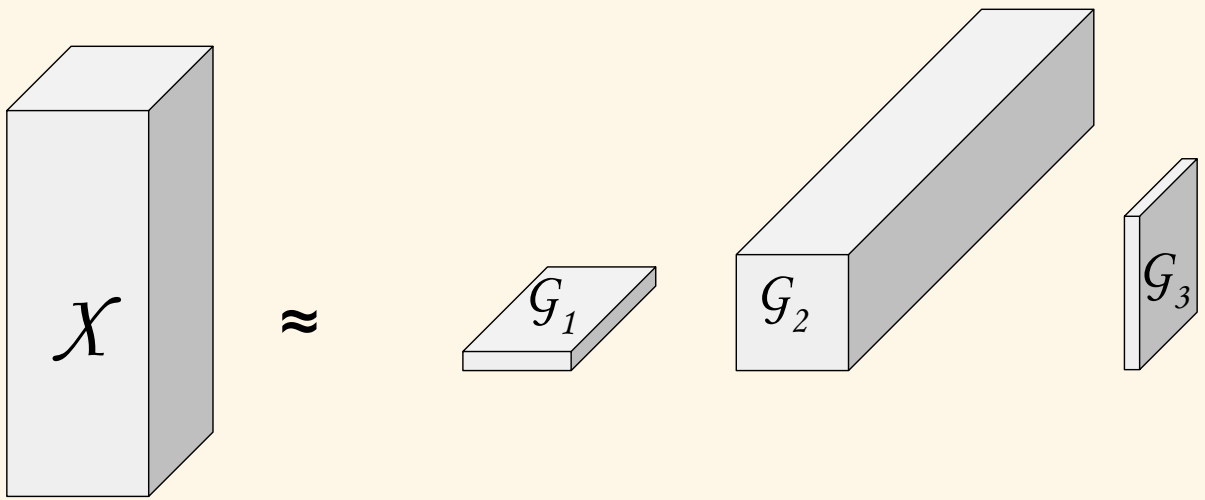
Goal Get rid of the core tensor.
Idea Encode the modes and their relations into matrix products (MPS) for each element.
Pros **Efficient for high d. Easy implementation. Continuous version**

Tensor train

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ in TT format.

$x_{i_1,\ldots,i_d} = \boldsymbol{G}_1(i_1)\boldsymbol{G}_2(i_2)\cdots\boldsymbol{G}_d(i_d)$, with $\boldsymbol{G}_\mu \in \mathbb{R}^{k_{\mu-1}\times k_\mu}$

Storage cost (TT) : $\mathcal{O}(k^2 dn)$



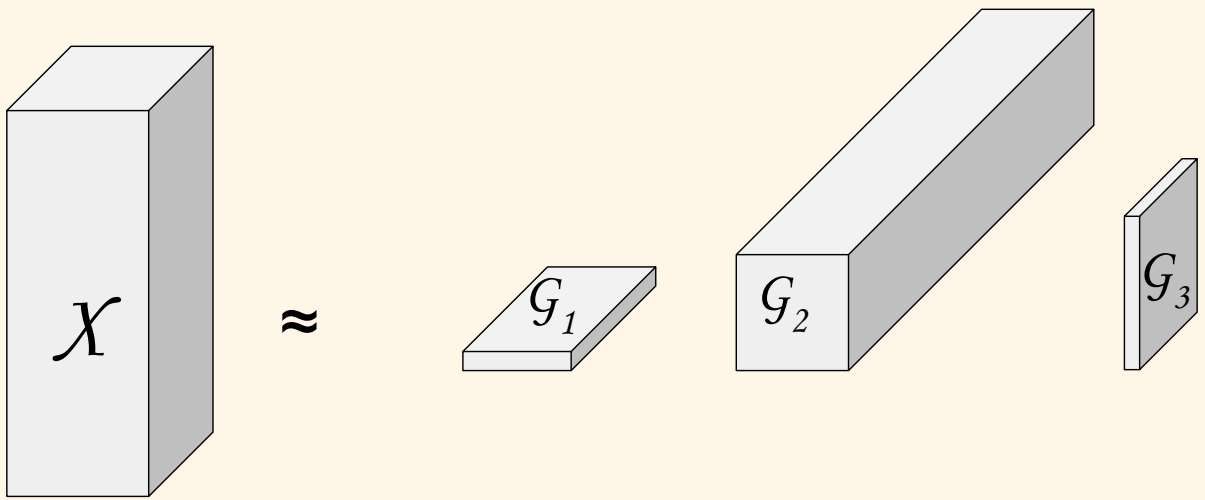
Goal Get rid of the core tensor.
Idea Encode the modes and their relations into matrix products (MPS) for each element.
Pros Efficient for high d. Easy implementation. Continuous version
Cons **Basis is not orthonormal.**

Tensor train

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ in TT format.

$x_{i_1,\ldots,i_d} = \boldsymbol{G}_1(i_1)\boldsymbol{G}_2(i_2)\cdots\boldsymbol{G}_d(i_d)$, with $\boldsymbol{G}_\mu \in \mathbb{R}^{k_{\mu-1}\times k_\mu}$

Storage cost (TT) : $\mathcal{O}(k^2 dn)$

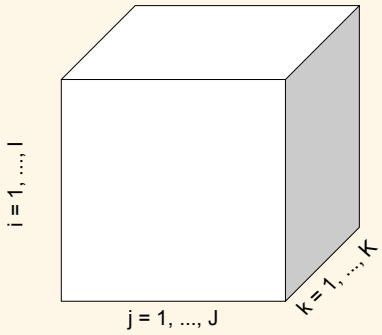


Goal Get rid of the core tensor.
Idea Encode the modes and their relations into matrix products (MPS) for each element.
Pros Efficient for high d. Easy implementation. Continuous version
Cons Basis is not orthonormal.
Algo **TT-SVD, same idea as ST-HOSVD but different format.**

Tensor decomposition recap

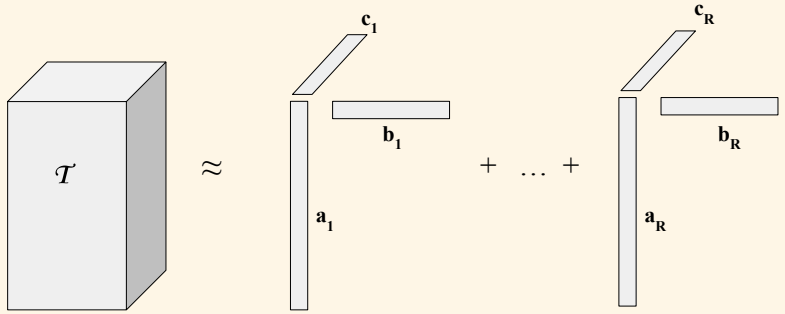
Full

- Original data format.
- Exponential storage cost : $\mathcal{O}(n^d)$



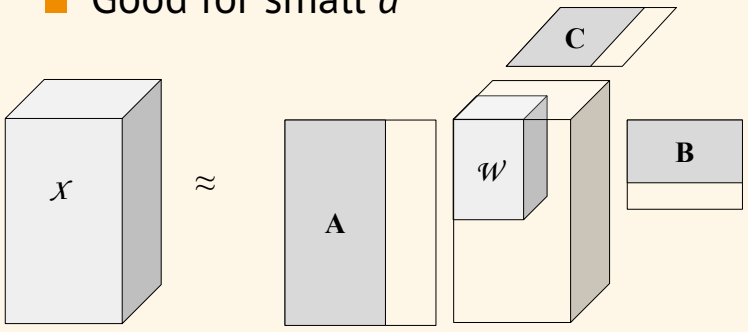
Canonical

- Linear storage cost : $\mathcal{O}(rnd)$
- Easy programming and No limit on d
- Approximation issues !



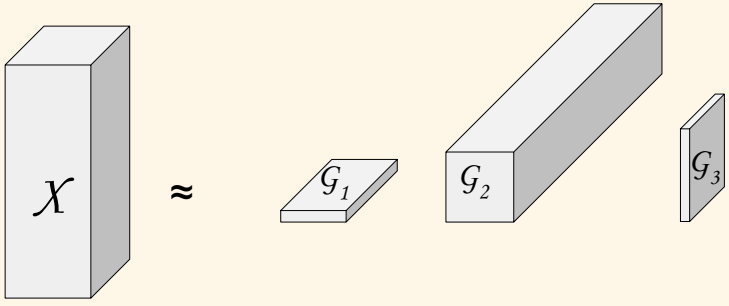
Tucker

- Exponential storage cost on rank : $\mathcal{O}(k^d + dkn)$
- Easy approximation with (ST-)HOSVD.
- Good for small d



Tensor Train

- d linear storage cost $\mathcal{O}(dk^2n)$, Great for large d
- TT-SVD + sampling algorithms.
- Easy to use



Introduction to Recursive-POD

- The POD expansion to multi-parametric functions.

$$T(x, y, z) = \sum_{m \geq 0} \sigma_m V_m(y, z)\, \varphi_m(x),$$

Introduction to Recursive-POD

- The POD expansion to multi-parametric functions.

$$T(x, y, z) = \sum_{m \geq 0} \sigma_m V_m(y, z)\, \varphi_m(x),$$

- We apply the POD expansion to each mode $V_m(y, z)$.

$$V_m(y, z) = \sum_{k \geq 0} \sigma_m^{(k)}\, w_m^{(k)}(z)\, v_m^{(k)}(y),$$

Introduction to Recursive-POD

- The POD expansion to multi-parametric functions.

$$T(x, y, z) = \sum_{m \geq 0} \sigma_m V_m(y, z)\, \varphi_m(x),$$

- We apply the POD expansion to each mode $V_m(y, z)$.

$$V_m(y, z) = \sum_{k \geq 0} \sigma_m^{(k)}\, w_m^{(k)}(z)\, v_m^{(k)}(y),$$

$\Longrightarrow$ The function $T \in L^2(X \times Y \times Z)$ admits the expansions

$$T_{RPOD} = \sum_{m \geq 0} \sum_{k \geq 0} \sigma_m\, \sigma_m^{(k)}\, \boldsymbol{\varphi}_m \otimes \boldsymbol{v}_m^{(k)} \otimes \boldsymbol{w}_m^{(k)}$$

Introduction to Recursive-POD

- The POD expansion to multi-parametric functions.

$$T(x, y, z) = \sum_{m \geq 0} \sigma_m V_m(y, z)\, \varphi_m(x),$$

- We apply the POD expansion to each mode $V_m(y, z)$.

$$V_m(y, z) = \sum_{k \geq 0} \sigma_m^{(k)}\, w_m^{(k)}(z)\, v_m^{(k)}(y),$$

$\Longrightarrow$ The function $T \in L^2(X \times Y \times Z)$ admits the expansions

$$T_{RPOD} = \sum_{m \geq 0} \sum_{k \geq 0} \sigma_m\, \sigma_m^{(k)}\, \boldsymbol{\varphi}_m \otimes \boldsymbol{v}_m^{(k)} \otimes \boldsymbol{w}_m^{(k)}$$

Introduction to Recursive-POD

- The POD expansion to multi-parametric functions.

$$T(x,y,z) = \sum_{m\geq 0} \sigma_m V_m(y,z)\,\varphi_m(x),$$

- We apply the POD expansion to each mode $V_m(y,z)$.

$$V_m(y,z) = \sum_{k\geq 0} \sigma_m^{(k)}\, w_m^{(k)}(z)\, v_m^{(k)}(y),$$

$\Longrightarrow$ The function $T \in L^2(X \times Y \times Z)$ admits the expansions

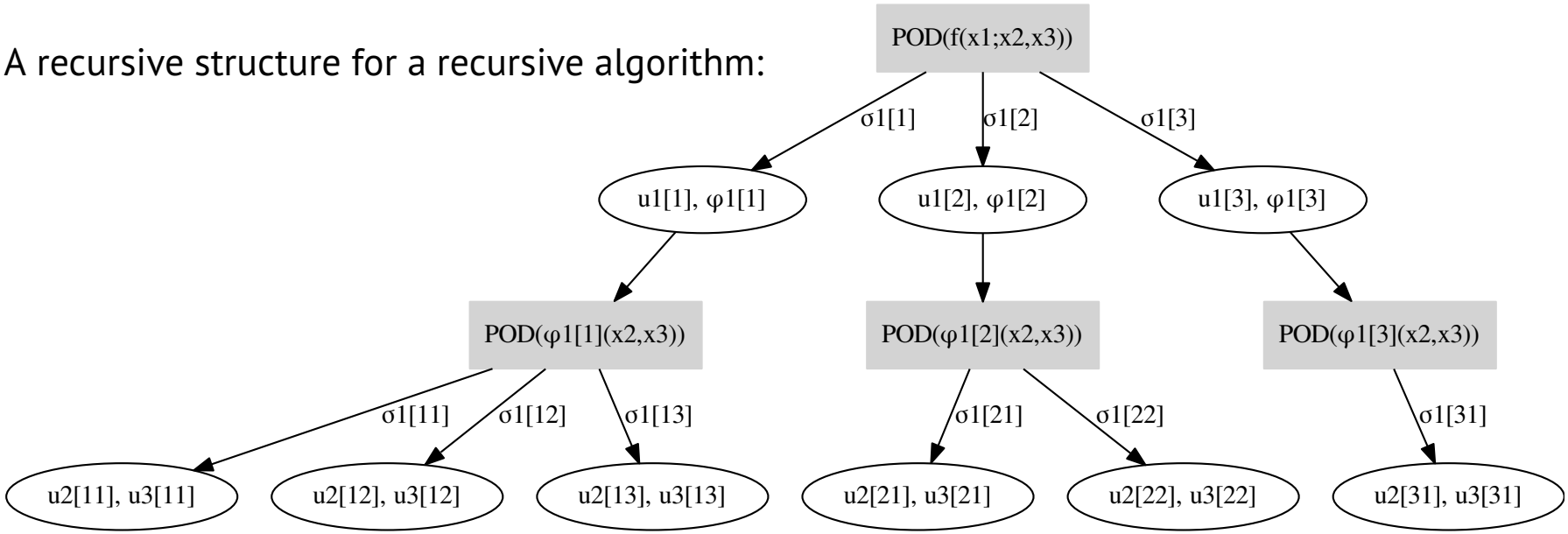
$$T_{RPOD} = \sum_{m\geq 0}\sum_{k\geq 0} \sigma_m\, \sigma_m^{(k)}\, \boldsymbol{\varphi}_m \otimes \boldsymbol{v}_m^{(k)} \otimes \boldsymbol{w}_m^{(k)}$$

To be compared with

$$T_{\text{canonical}} = \sum_{m\geq 0} X_m \otimes Y_m \otimes Z_m \quad \text{and} \quad T_{\text{tucker}} = \sum_{i,j,k\geq 0} w_{ijk} X_i \otimes Y_j \otimes Z_k$$

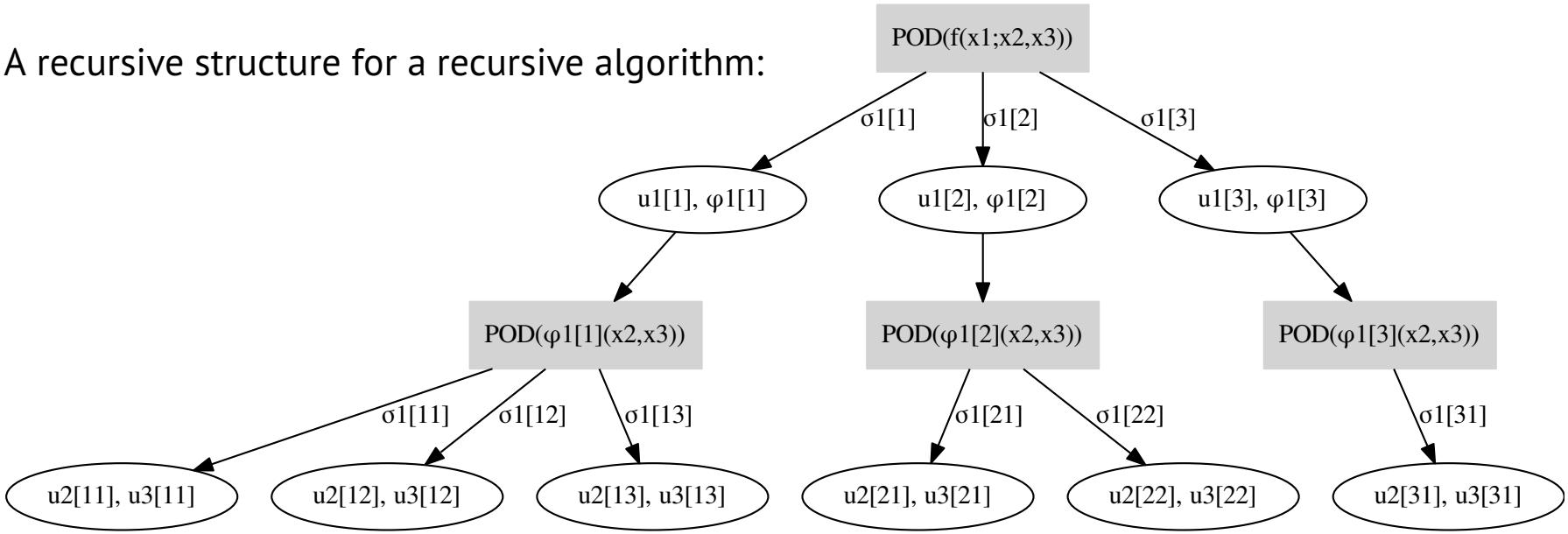
RPOD

A recursive structure for a recursive algorithm:



RPOD

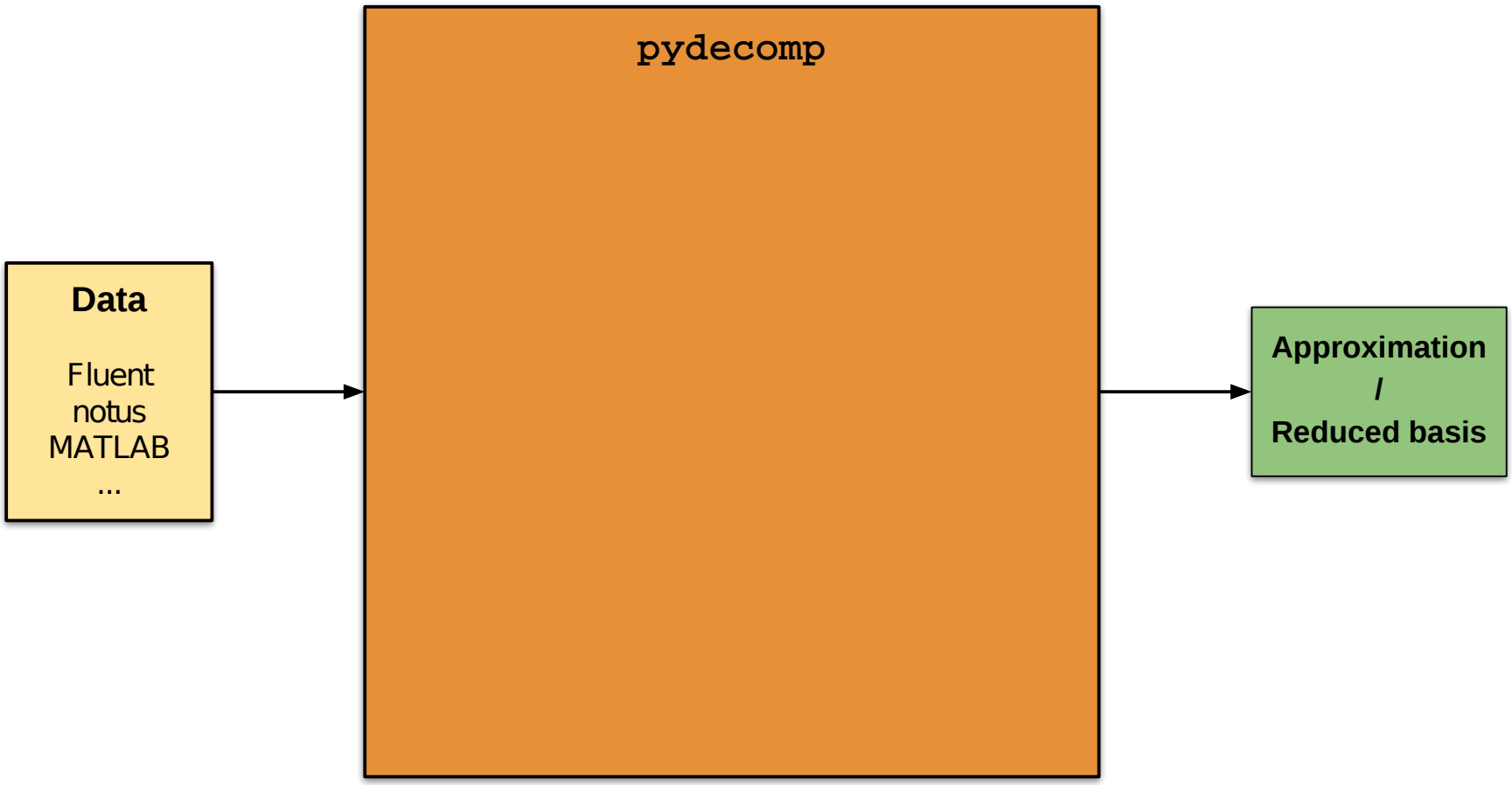
A recursive structure for a recursive algorithm:



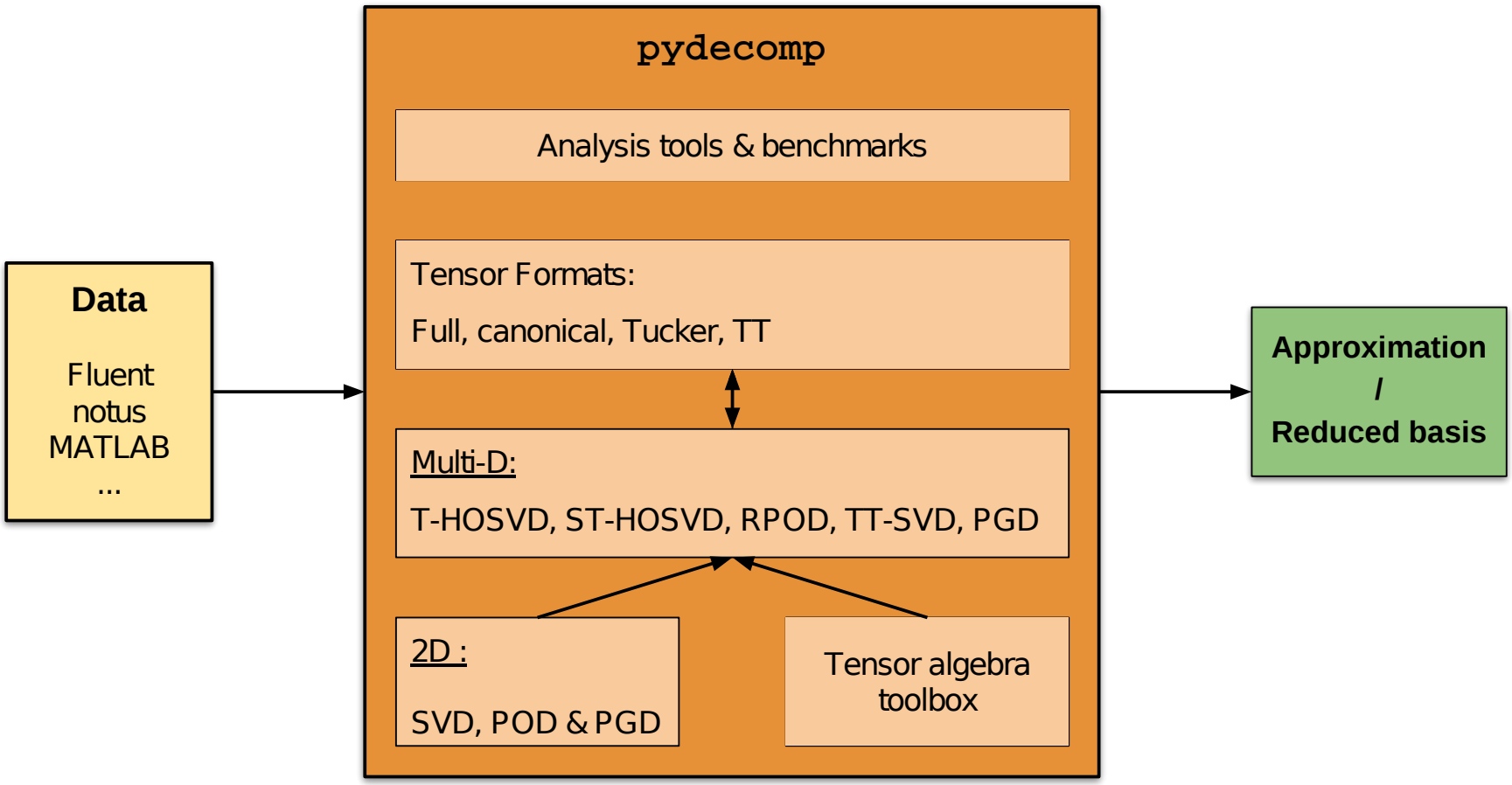
Properties

- Quasi Optimality property
- Almost linear in d
- Data structure is different

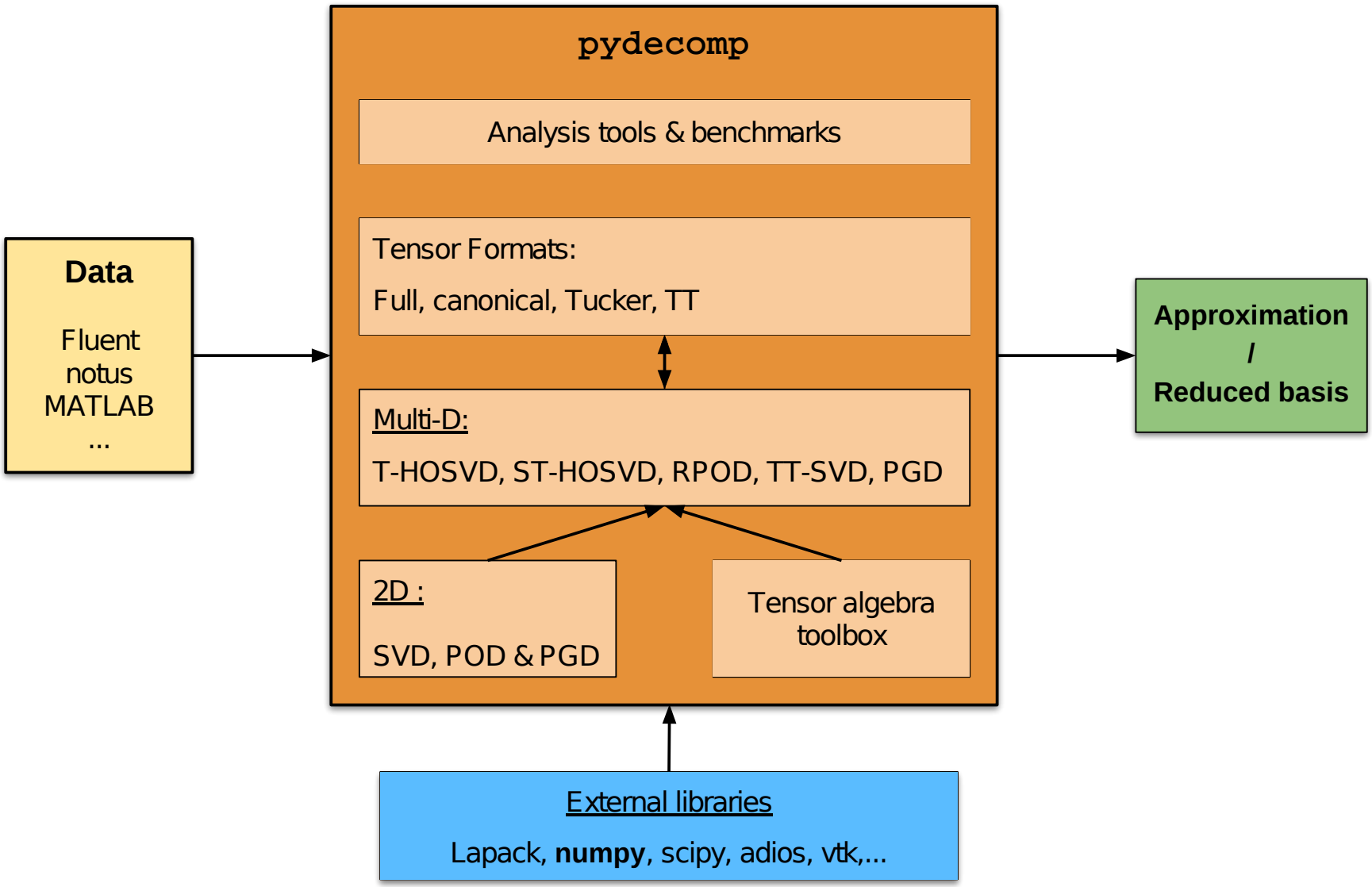
pydecomp: a data decomposition library



pydecomp: a data decomposition library



pydecomp: a data decomposition library



Synthetic data 3D

$$\mathcal{E} = \frac{||\mathcal{T}_{\text{exact}} - \mathcal{T}_{\text{approx}}||_F}{||\mathcal{T}_{\text{exact}}||_F} \quad \text{or} \quad \mathcal{E} = \frac{||f_{\text{exact}} - f_{\text{approx}}||_{L^2}}{||f_{\text{exact}}||_{L^2}}$$

Synthetic data 3D

$$\mathcal{E} = \frac{||\mathcal{T}_{\text{exact}} - \mathcal{T}_{\text{approx}}||_F}{||\mathcal{T}_{\text{exact}}||_F} \quad \text{or} \quad \mathcal{E} = \frac{||f_{\text{exact}} - f_{\text{approx}}||_{L^2}}{||f_{\text{exact}}||_{L^2}}$$

$$CR = \frac{\texttt{Mem\_cost}(\mathcal{T}_{\text{decomp}})}{\texttt{Mem\_cost}(\mathcal{T}_{\text{exact}})} (\times 100 \text{ for } \%).$$

Higher dimensions

$$f_2(\boldsymbol{x}) = \sin(||\boldsymbol{x}||_2)$$

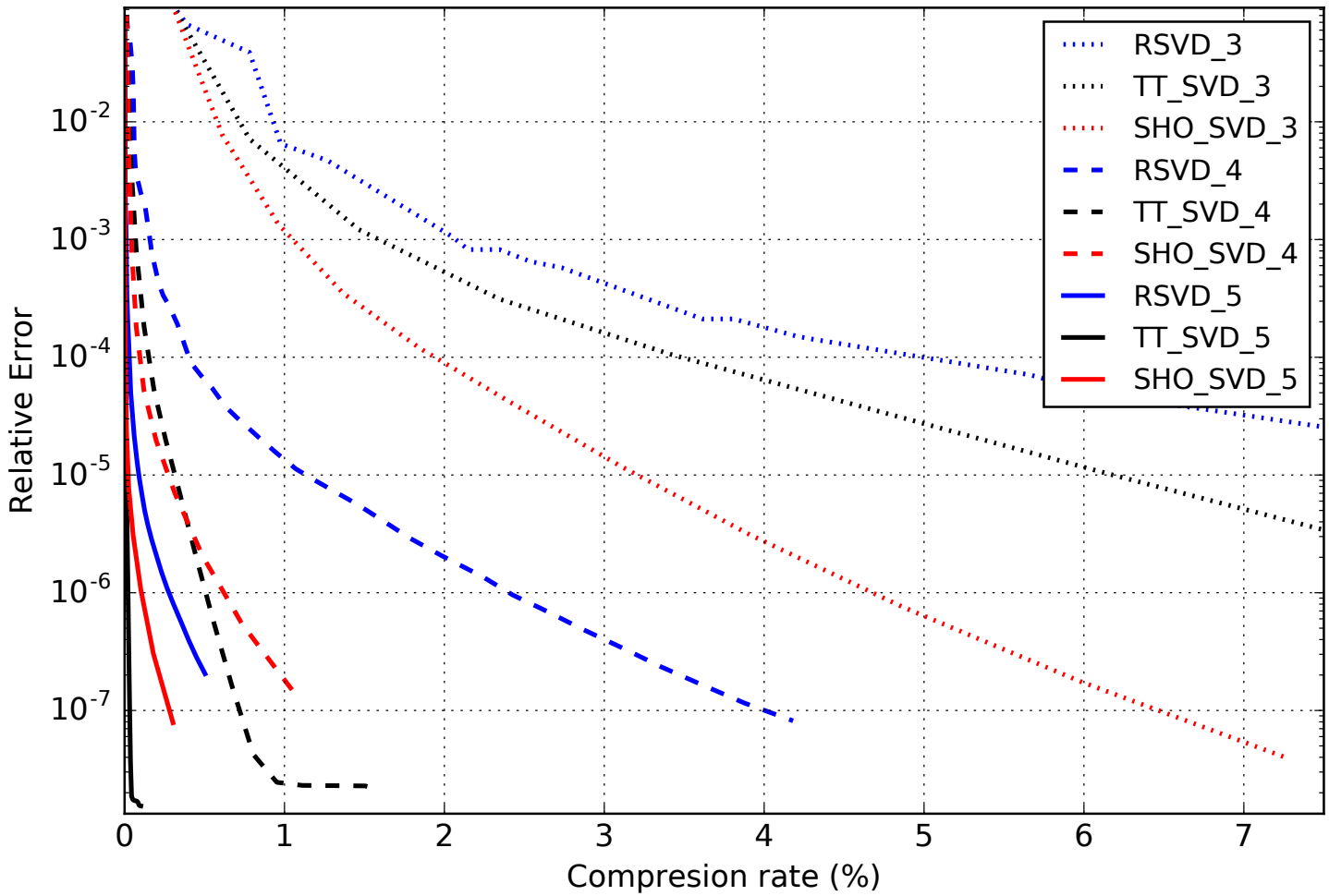


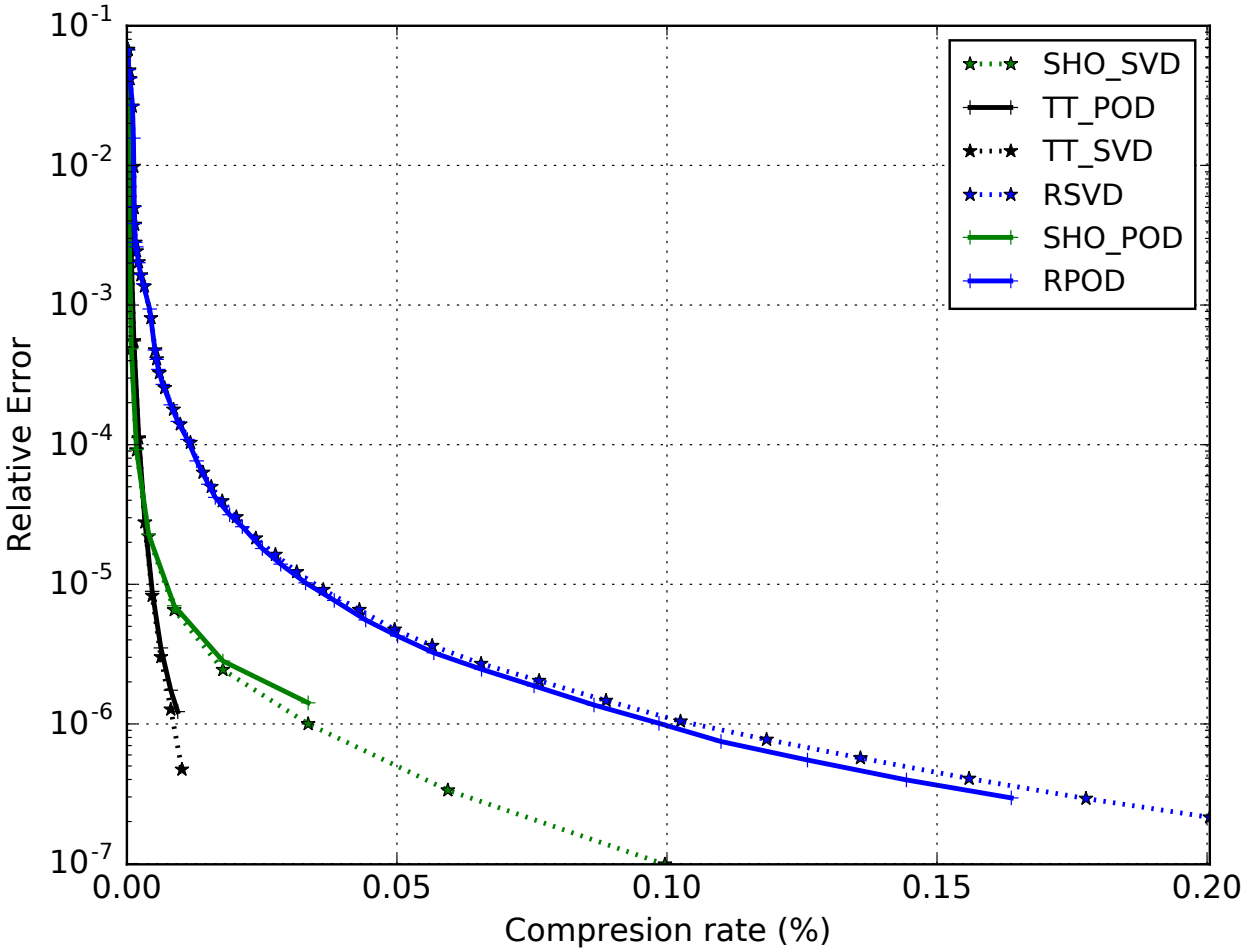
Figure: f_2 decomposition with $d = 3$ to 5 on a 32^d grid with 3 methods, L^2 scalar product.

Scalar product influence

$$
\begin{aligned}
f_s(x_1, x_2, x_3, x_4, x_5) \quad = \quad & x_1^2\{\sin[5x_2\pi + 3\log(x_1^3 + x_2^2 + x_4^3 + x_3 + \pi^2)] - 1\}^2 \\
& +(x_1 + x_3 - 1)(2x_2 - x_3)(4x_5 - x_4)\cos[30(x_1 + x_3 + x_4 + x_5)] \\
& \log(6 + x_1^2 x_2^2 + x_3^3) - 4x_1^2 x_2 x_5^3(-x_3 + 1)^{3/2}
\end{aligned}
$$

$\Omega = [0, 1]^5$, cartesian grid, $n = 40$

Scalar product influence



method	comp.	eval.
RPOD	9.535	31.80
RSVD	7.964	32.20
ST-HOSVD	1.096	1.23
ST-HOPOD	2.378	0.98
TT-SVD	1.205	1.19
TT-POD	2.206	1.13

Table: CPU times (s) on f_v for $n = 40$, $d = 5$, $\epsilon = 10^{-12}$

Experimental data: 3 parameter LDC

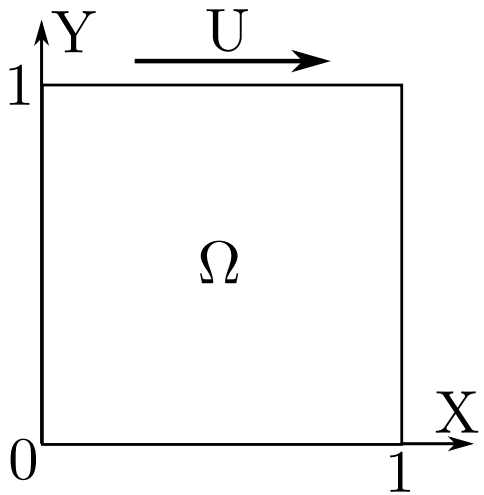


Figure: LDC schematic view

Parameter domain

- $\boldsymbol{x} \in [0,1]^2$, on a 257×257 grid
- $t \in [1900, 1940, 0.2]$
- $Re \in \{10000, 10020, \ldots, 10100\}$

Experimental data: 3 parameter LDC

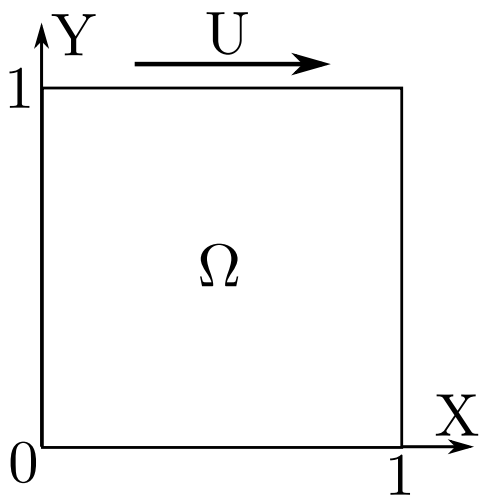


Figure: LDC schematic view

Parameter domain

- $\boldsymbol{x} \in [0,1]^2$, on a 257×257 grid
- $t \in [1900, 1940, 0.2]$
- $Re \in \{10000, 10020, \ldots, 10100\}$

Tensor

- Order 3
- $66049 \times 200 \times 6$

or

- Order 4
- $257 \times 257 \times 200 \times 6$

3 parameter LDC decomposition

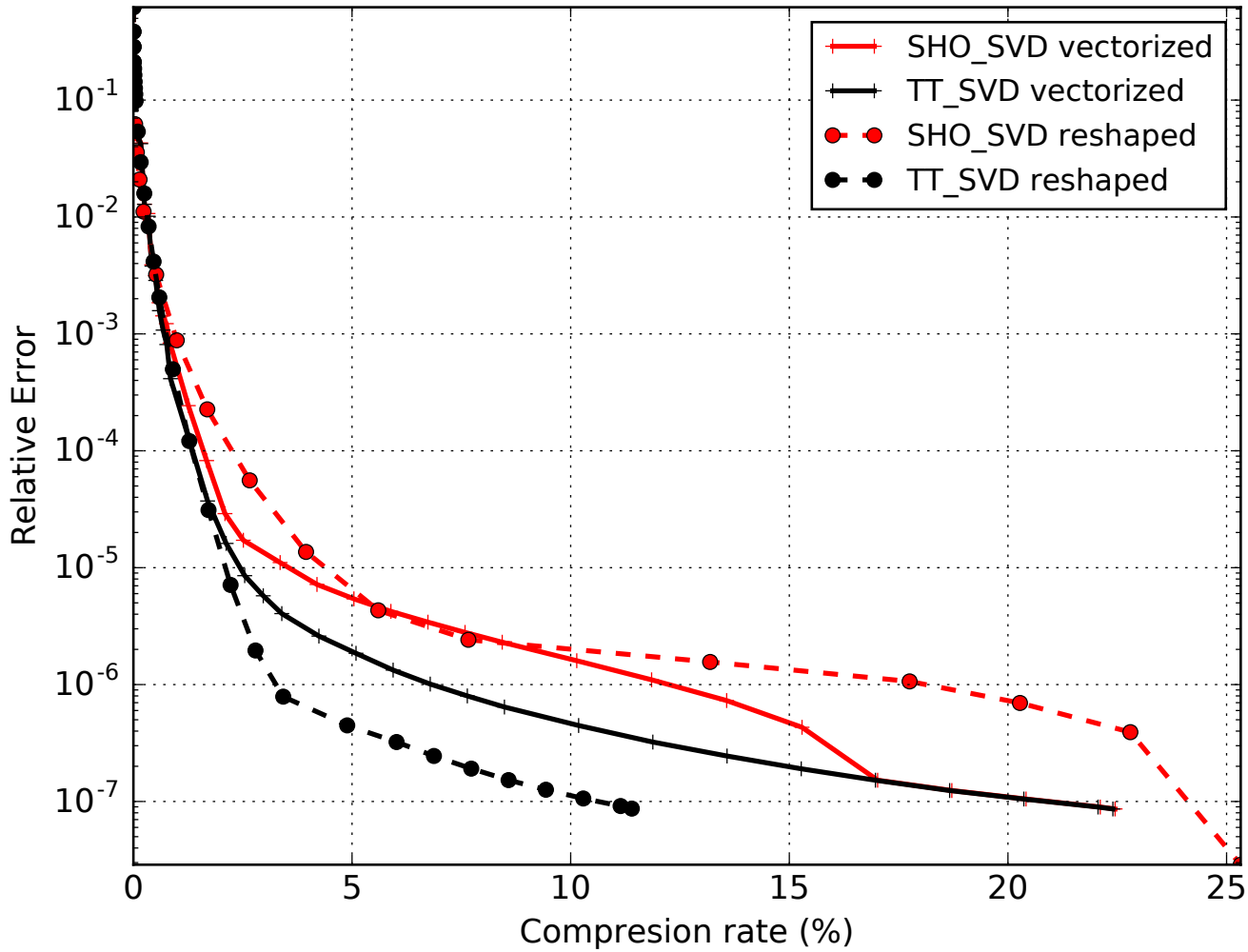


Figure: LDC decomposition within stable limit cycle time range. Input tensor shape : $6 \times 201 \times 66049$. Cutoff tolerance : $\varepsilon = 10^{-4}$

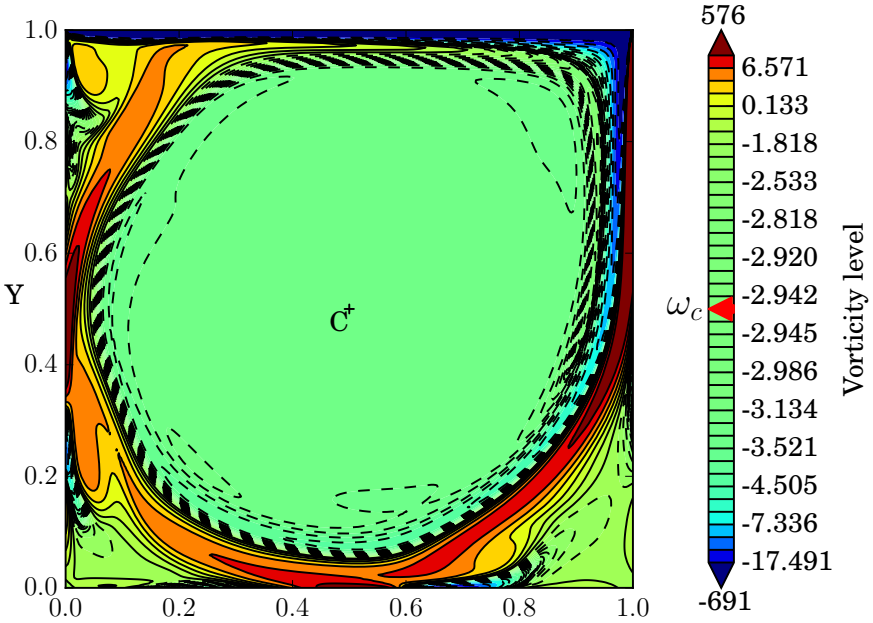


Figure: Original data

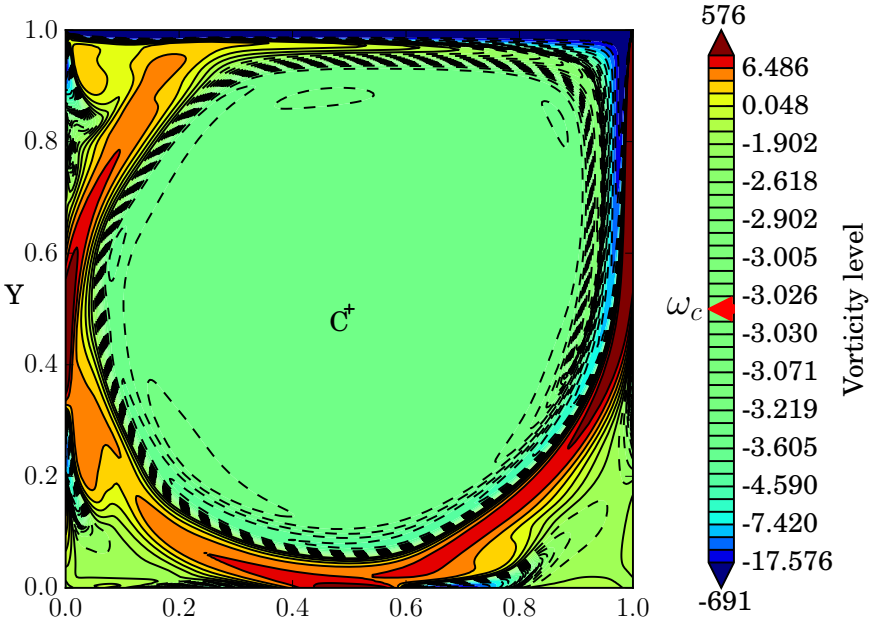


Figure: Reconstructed from ST-HOSVD, 1% error.

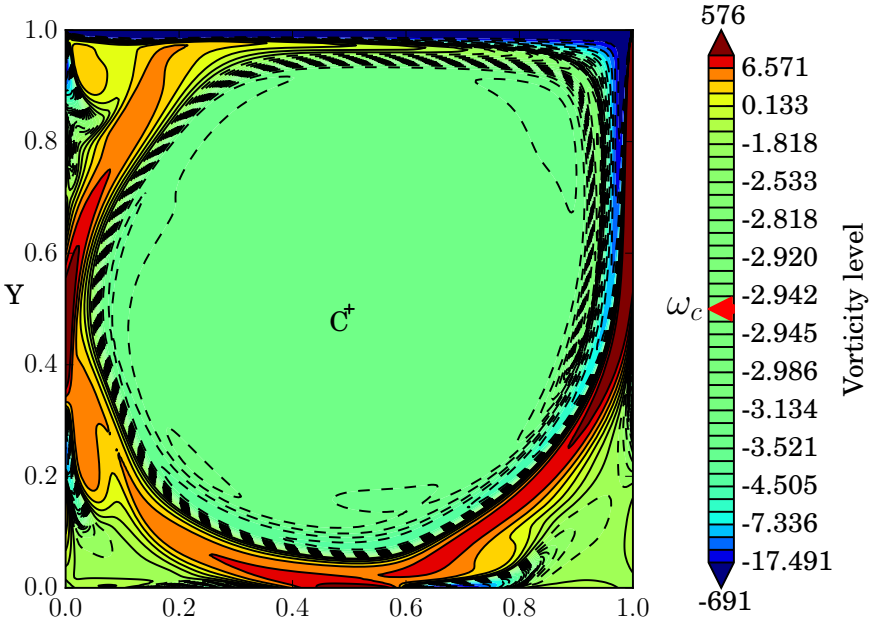


Figure: Original data

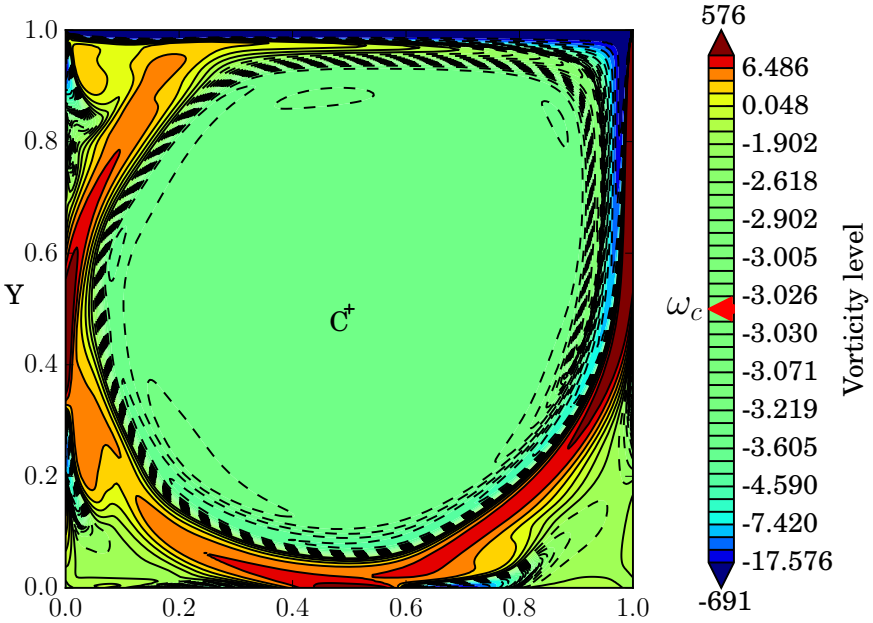
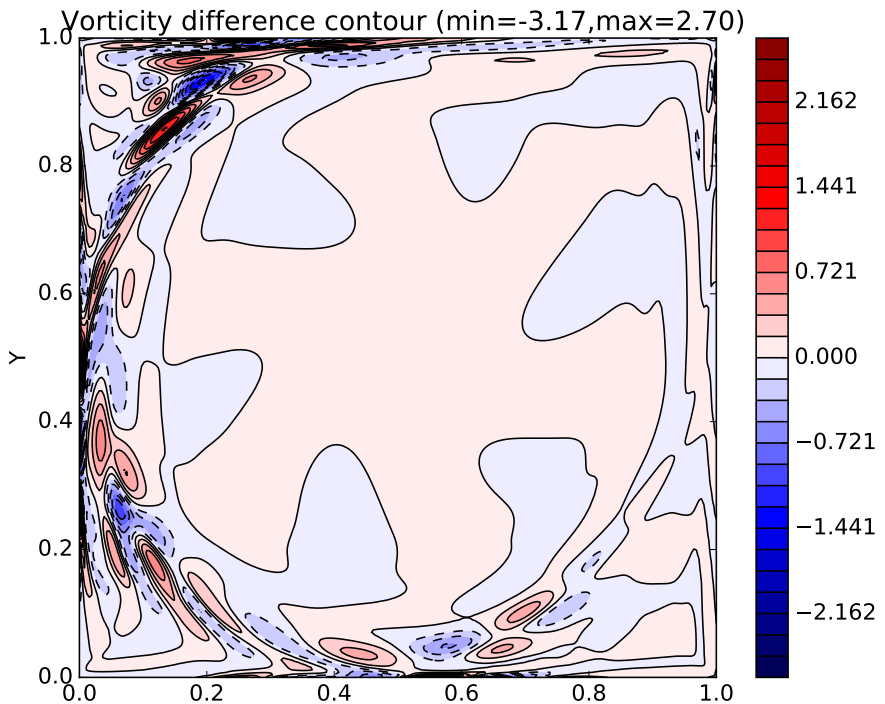


Figure: Reconstructed from ST-HOSVD, 1% error.



Experimental data: Droplet evaporation

Input data

- 320×356 camera spatial resolution
- 51 wavelength
- 29 snapshots

Experimental data: Droplet evaporation

Input data

- 320×356 camera spatial resolution
- 51 wavelength
- 29 snapshots

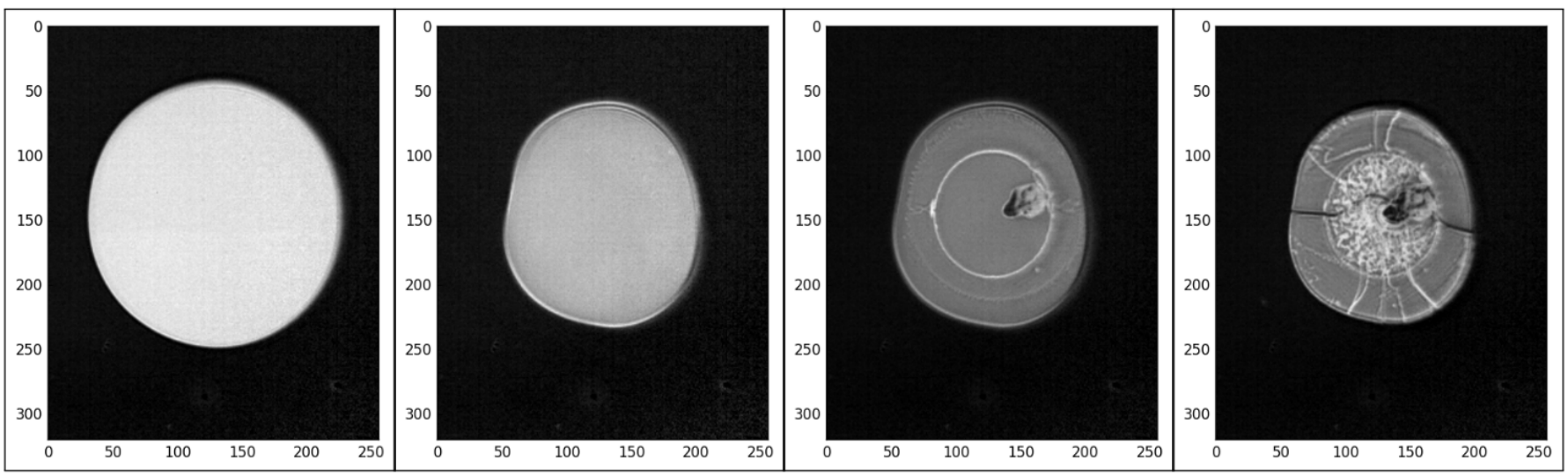
$\Longrightarrow$ Tensor of size $29 \times 51 \times 320 \times 356$ in a 800MB MATLAB file.

Experimental data: Droplet evaporation

Input data

- 320 × 356 camera spatial resolution
- 51 wavelength
- 29 snapshots

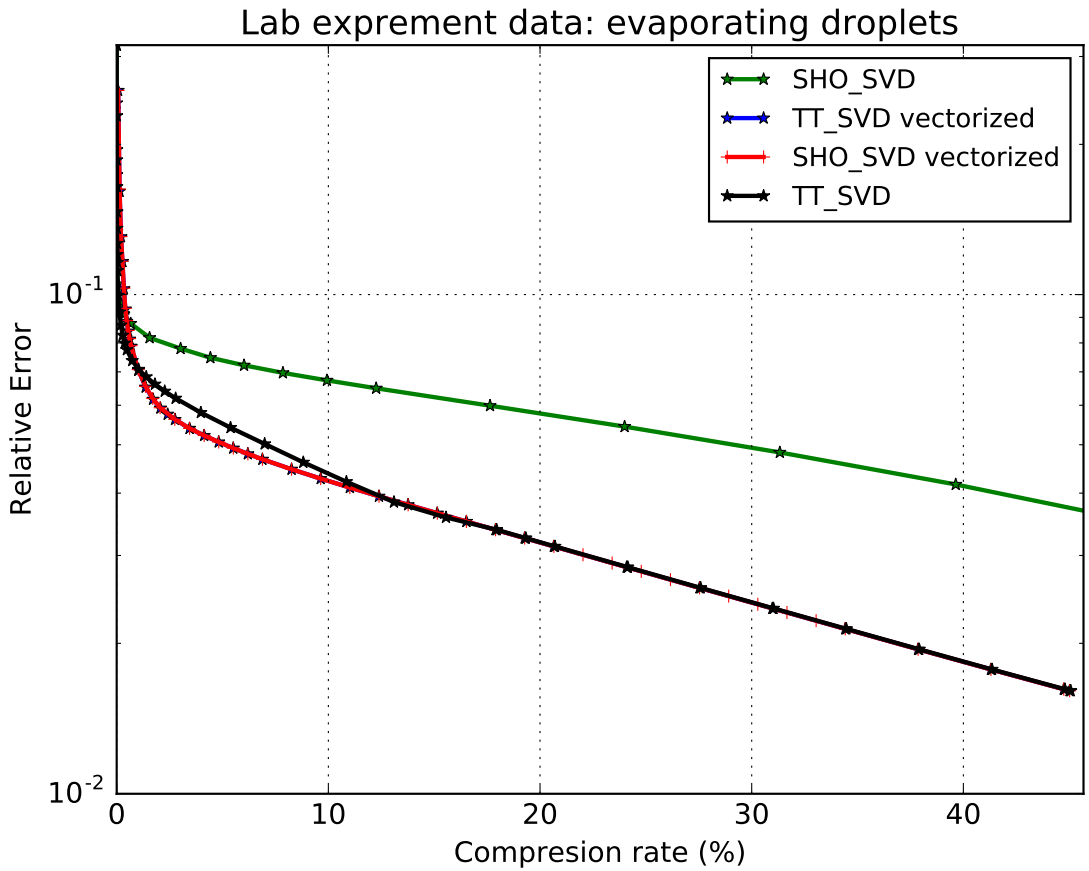
$\Longrightarrow$ Tensor of size $29 \times 51 \times 320 \times 356$ in a 800MB MATLAB file.



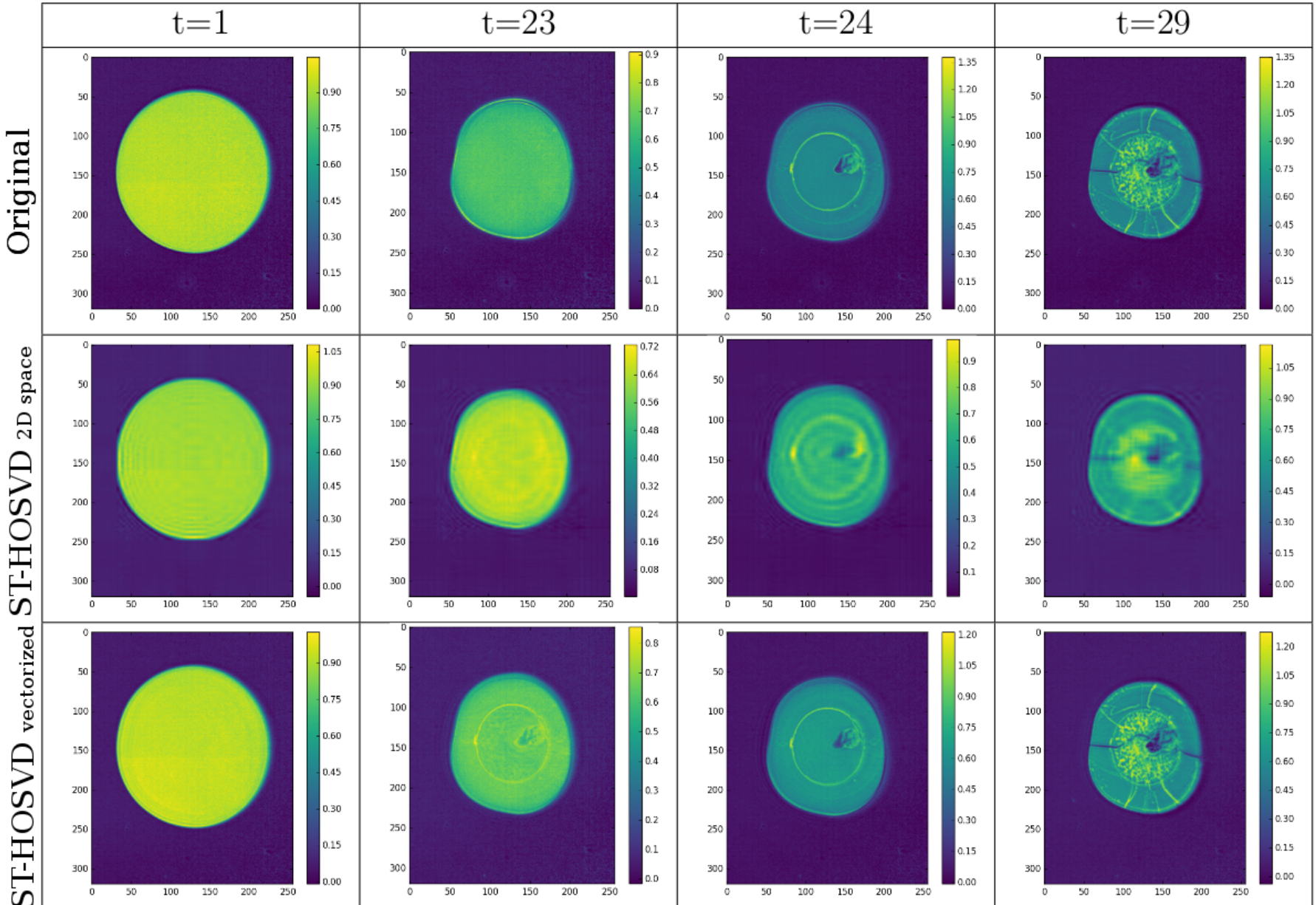
(a) t= 1/29 (b) t= 23/29 (c) t= 24/29 (d) t= 29/29

Figure: Evaporating droplets. Data provided by C. Pradère

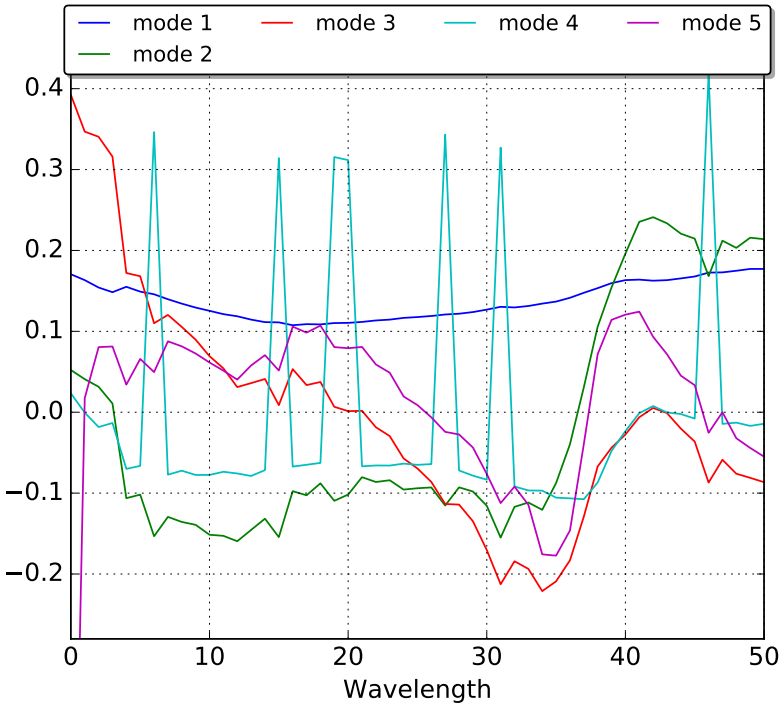
Droplet evaporation data approximation

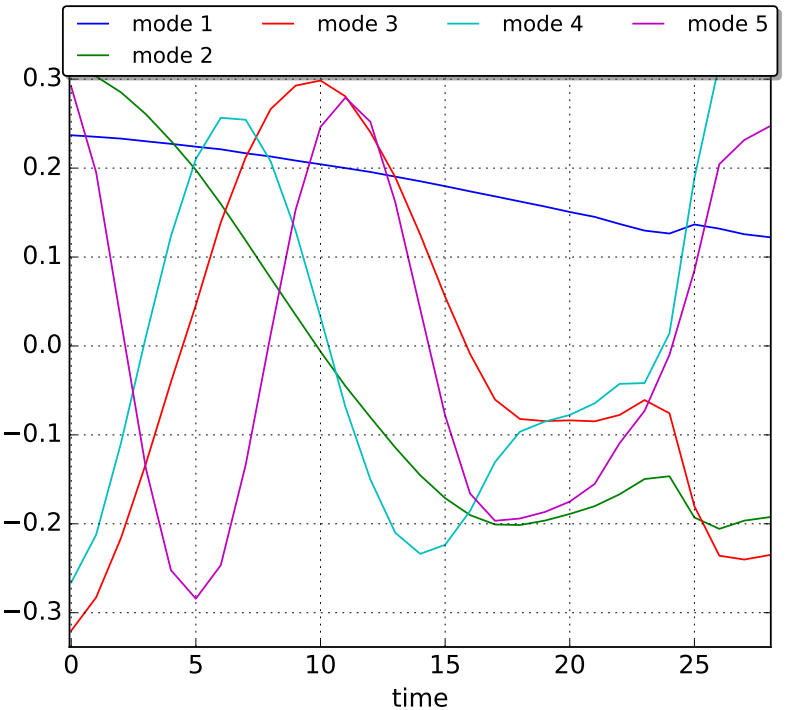


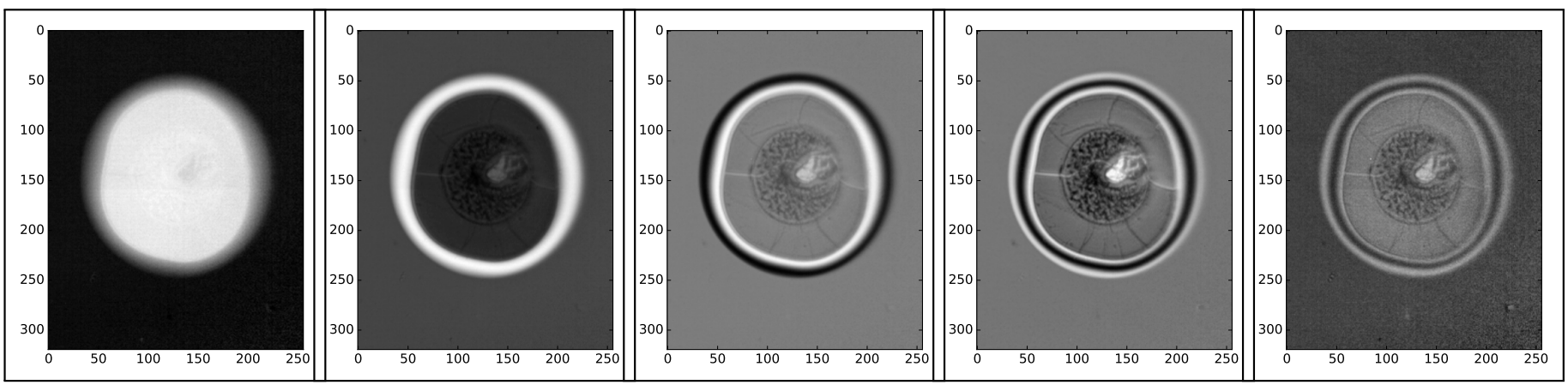
Droplet evaporation data approximation



Droplet evaporation data modes



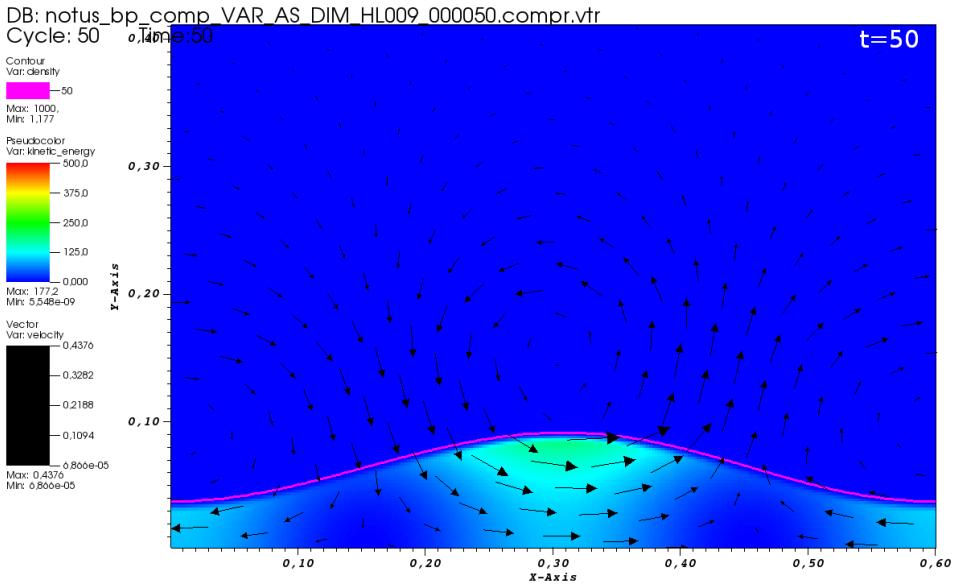


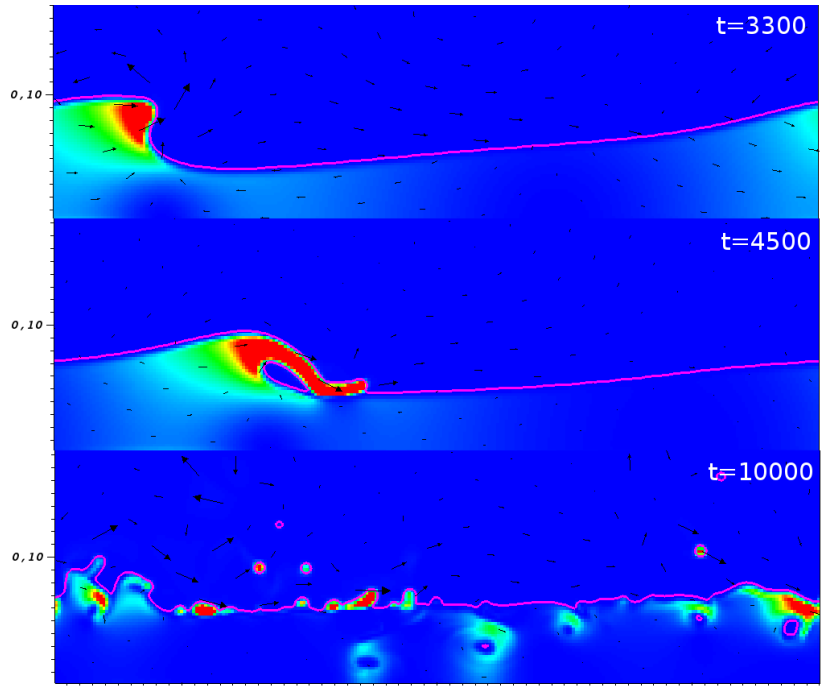


Spatial modes from vectorized data layout (1 to 5, left to right).

Experimental data: Breaking wave

Breaking wave, notus simulation
F. Desmons, 2018



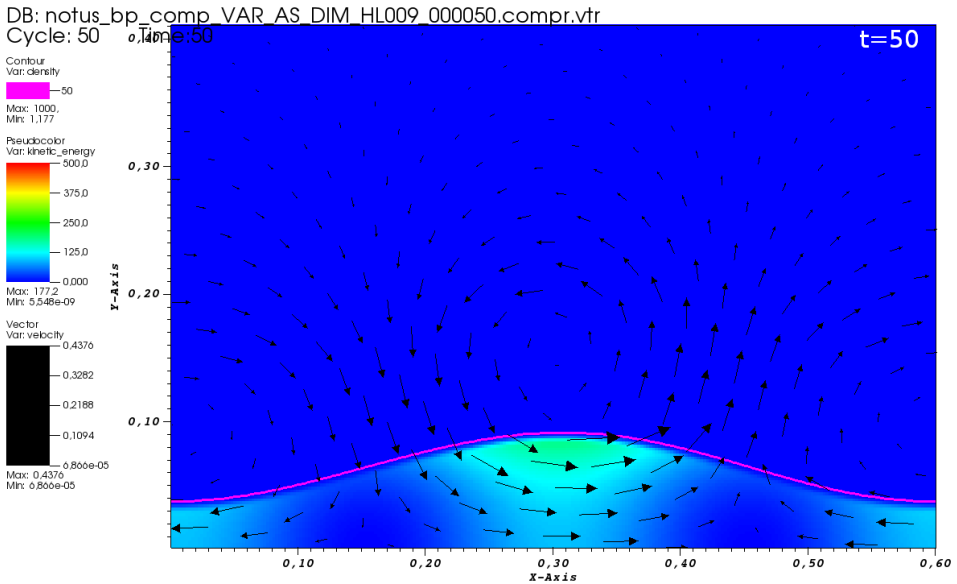


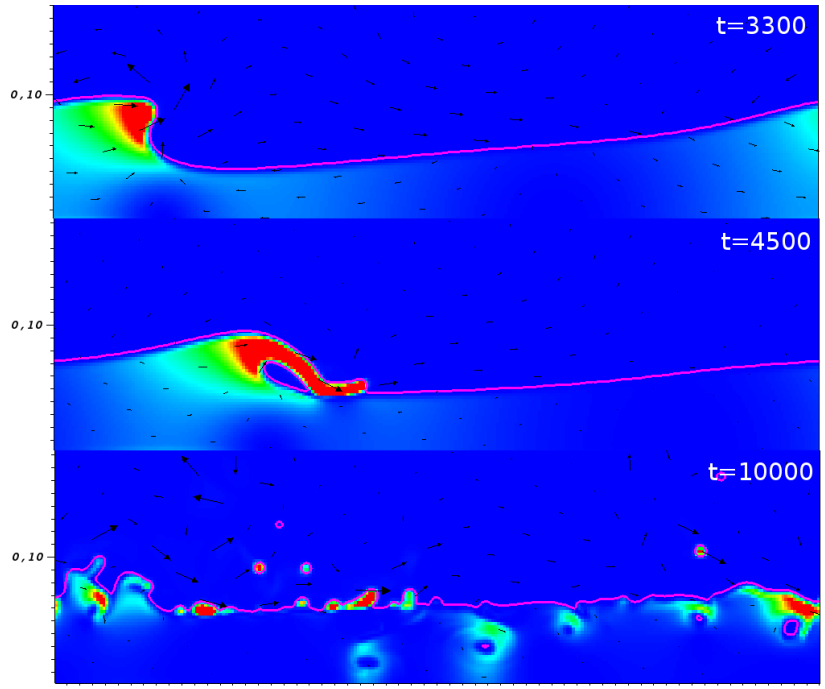
Input data from notus CFD

- $\Omega = [0, 0.6]^2$ discretized on a 256×256 Cartesian grid

Experimental data: Breaking wave

Breaking wave, notus simulation
F. Desmons, 2018



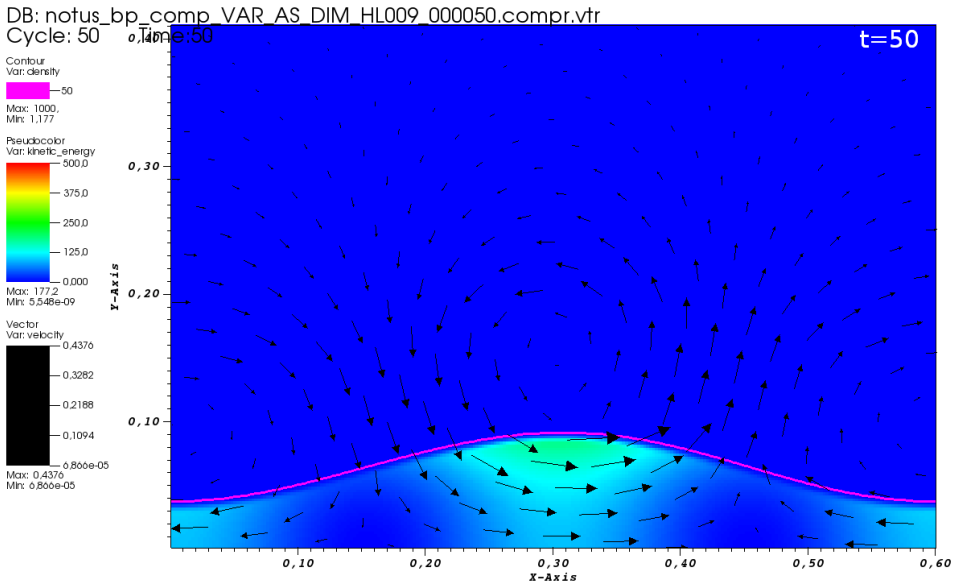


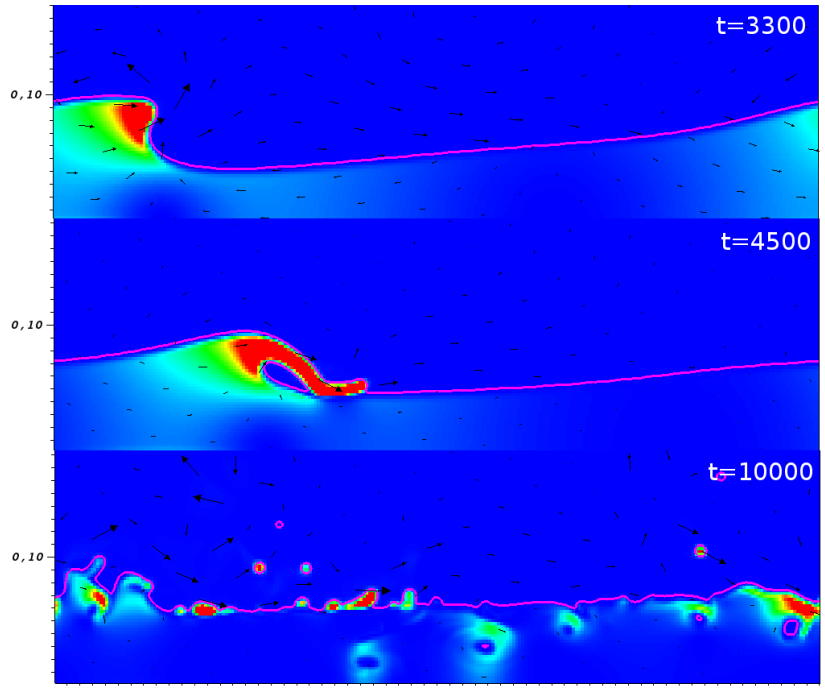
Input data from notus CFD

- $\Omega = [0, 0.6]^2$ discretized on a 256×256 Cartesian grid
- 201 snapshots

Experimental data: Breaking wave

Breaking wave, notus simulation
F. Desmons, 2018



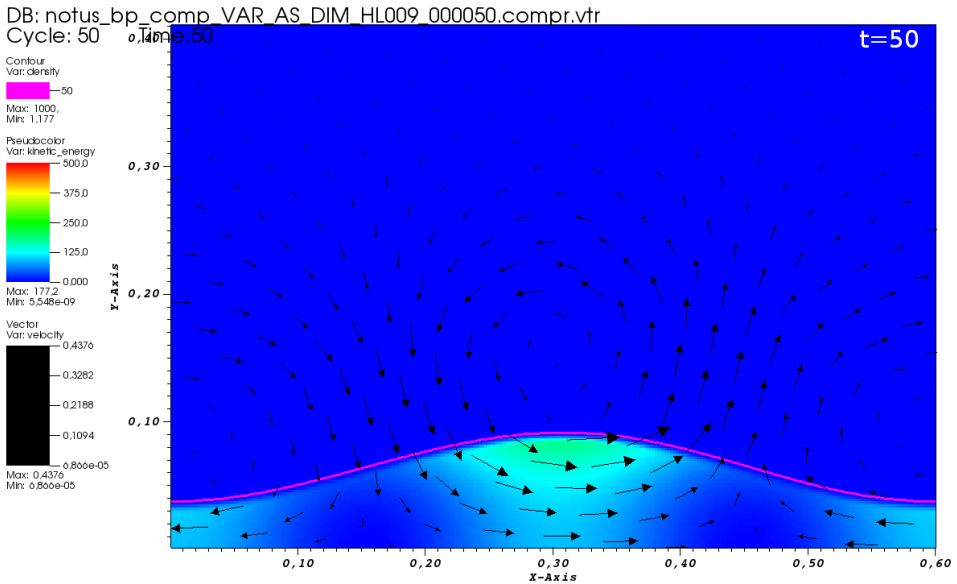


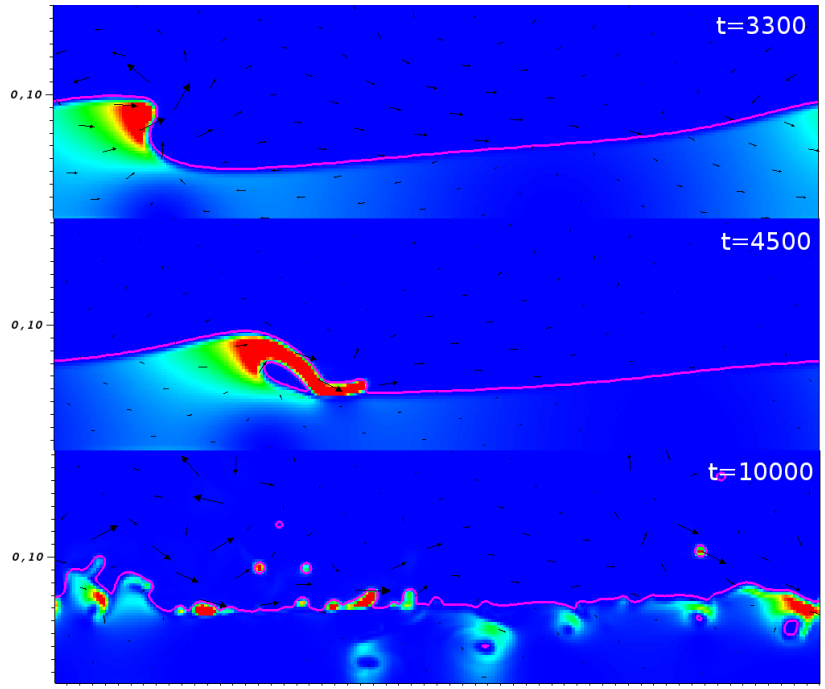
Input data from notus CFD

- $\Omega = [0, 0.6]^2$ discretized on a 256×256 Cartesian grid
- 201 snapshots
- 3 height / wavelength ratio (9,10,11cm)

Experimental data: Breaking wave

Breaking wave, notus simulation
F. Desmons, 2018



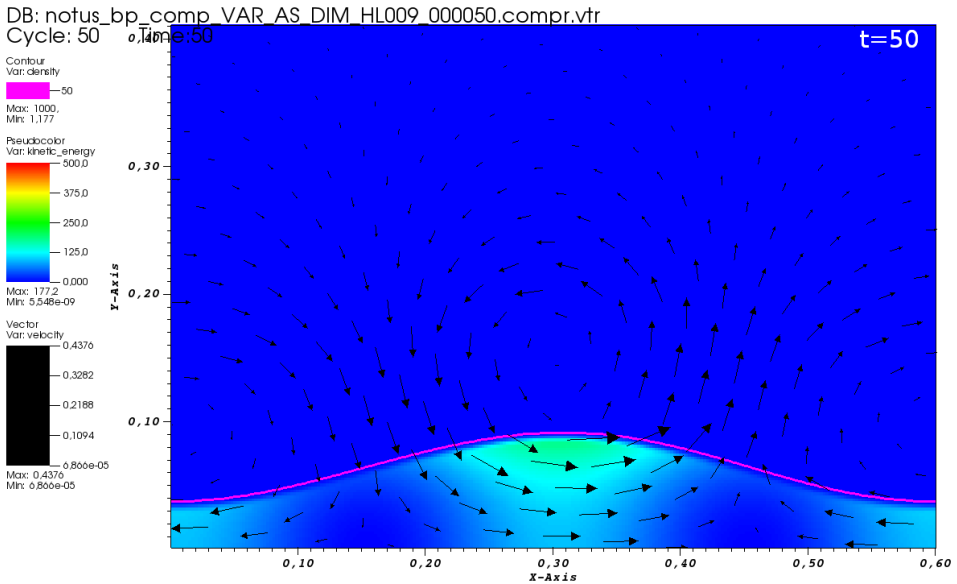


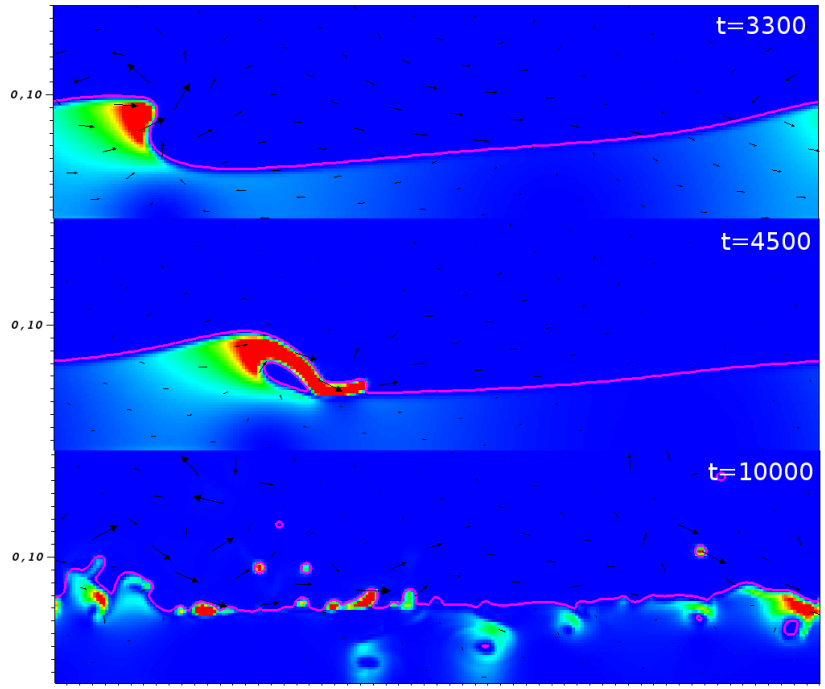
Input data from notus CFD

- $\Omega = [0, 0.6]^2$ discretized on a 256×256 Cartesian grid
- 201 snapshots
- 3 height / wavelength ratio (9,10,11cm)
- 5 output param : u, v, ω, p, ρ

Experimental data: Breaking wave

Breaking wave, notus simulation
F. Desmons, 2018



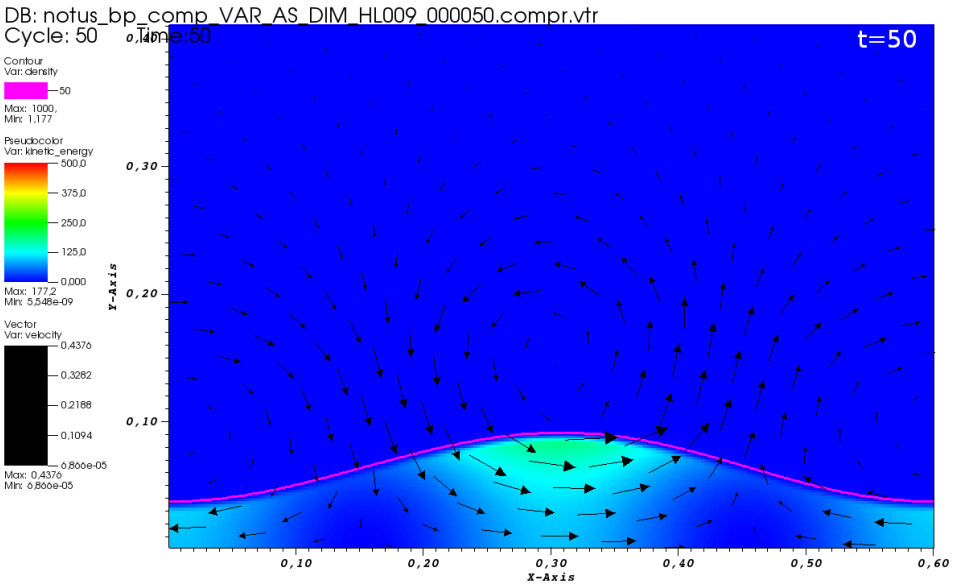


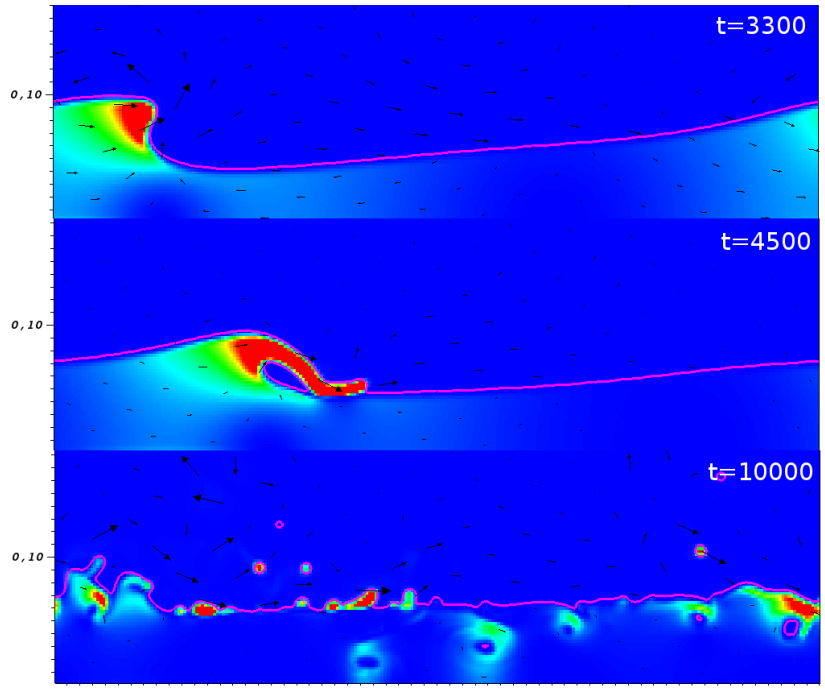
Input data from notus CFD

- $\Omega = [0, 0.6]^2$ discretized on a 256×256 Cartesian grid
- 201 snapshots
- 3 height / wavelength ratio (9,10,11cm)
- 5 output param : u, v, ω, p, ρ

Experimental data: Breaking wave

Breaking wave, notus simulation
F. Desmons, 2018



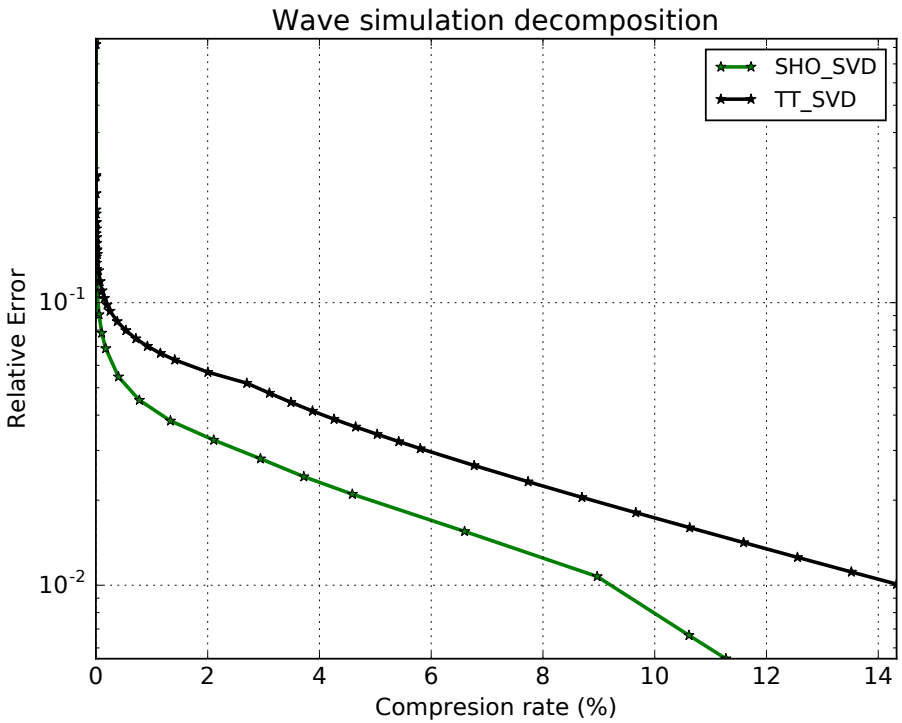


Input data from notus CFD

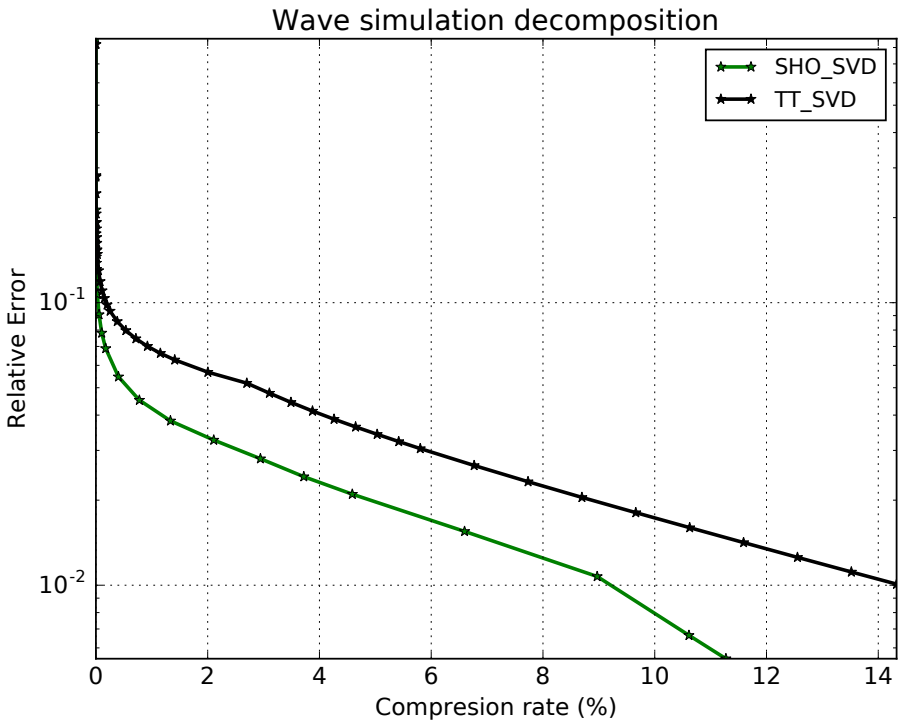
- $\Omega = [0, 0.6]^2$ discretized on a 256×256 Cartesian grid
- 201 snapshots
- 3 height / wavelength ratio (9,10,11cm)
- 5 output param : u, v, ω, p, ρ

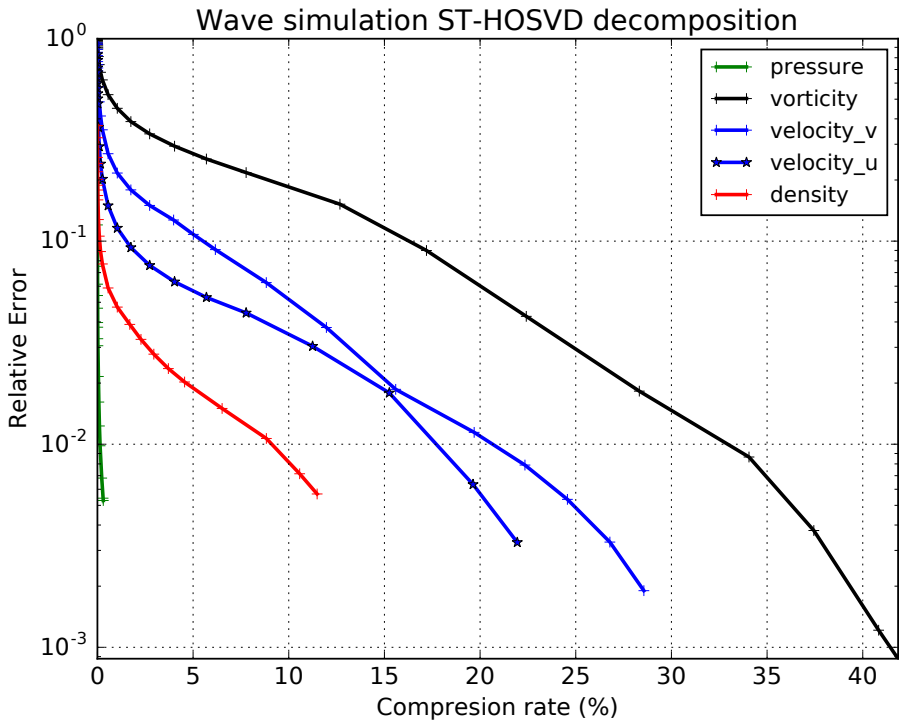
1. Five order 4 tensor $(3 \times 201 \times 256 \times 256)$
2. 1 order 5 tensor $(5 \times 3 \times 201 \times 256 \times 256)$

Wave decomposition error, $\varepsilon = 10^{-3}$

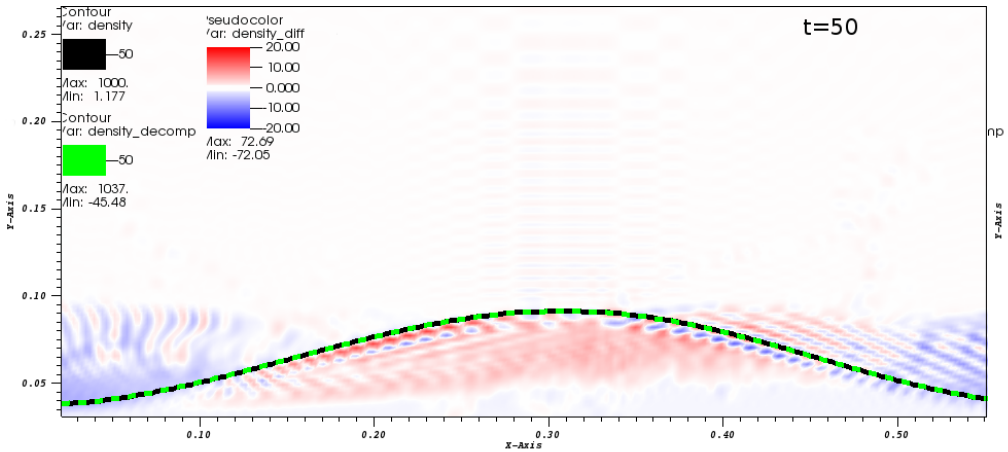


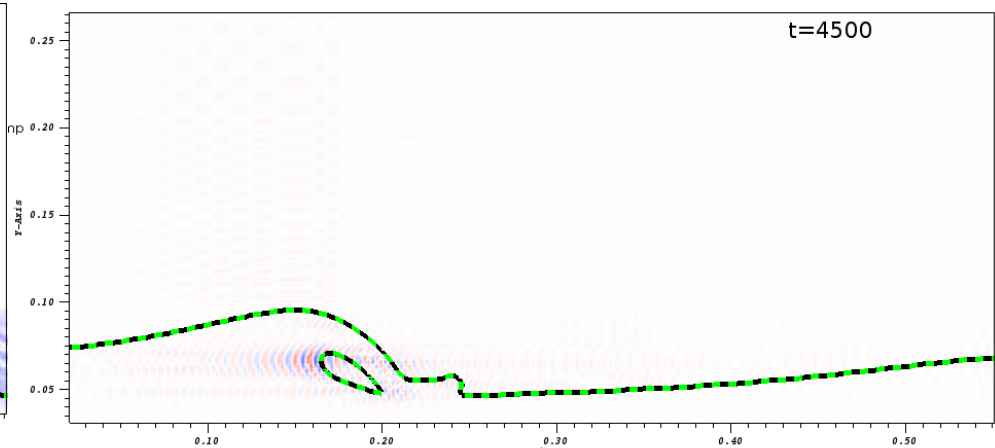
Wave decomposition error, $\varepsilon = 10^{-3}$

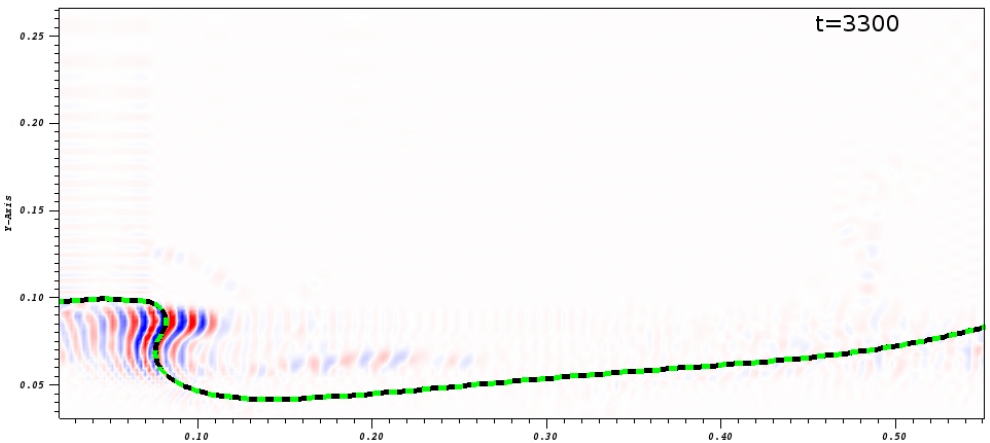


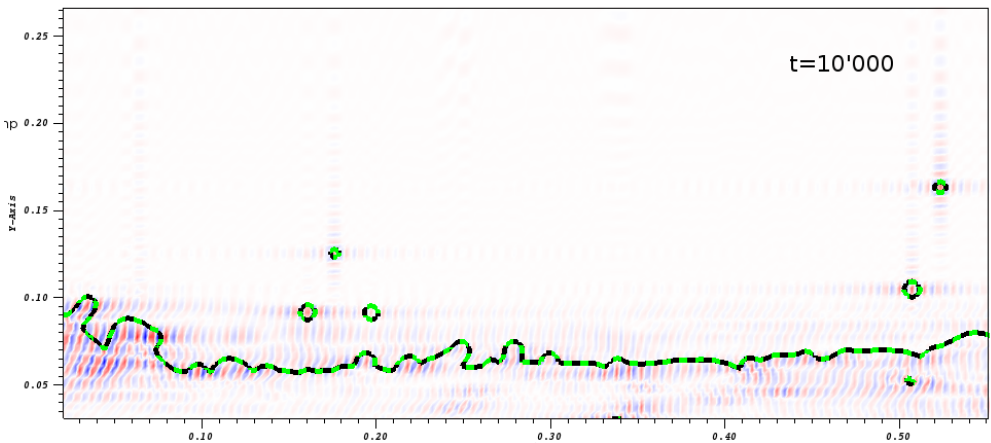


Is $\varepsilon = 10^{-3}$ accurate enough for interpretation?

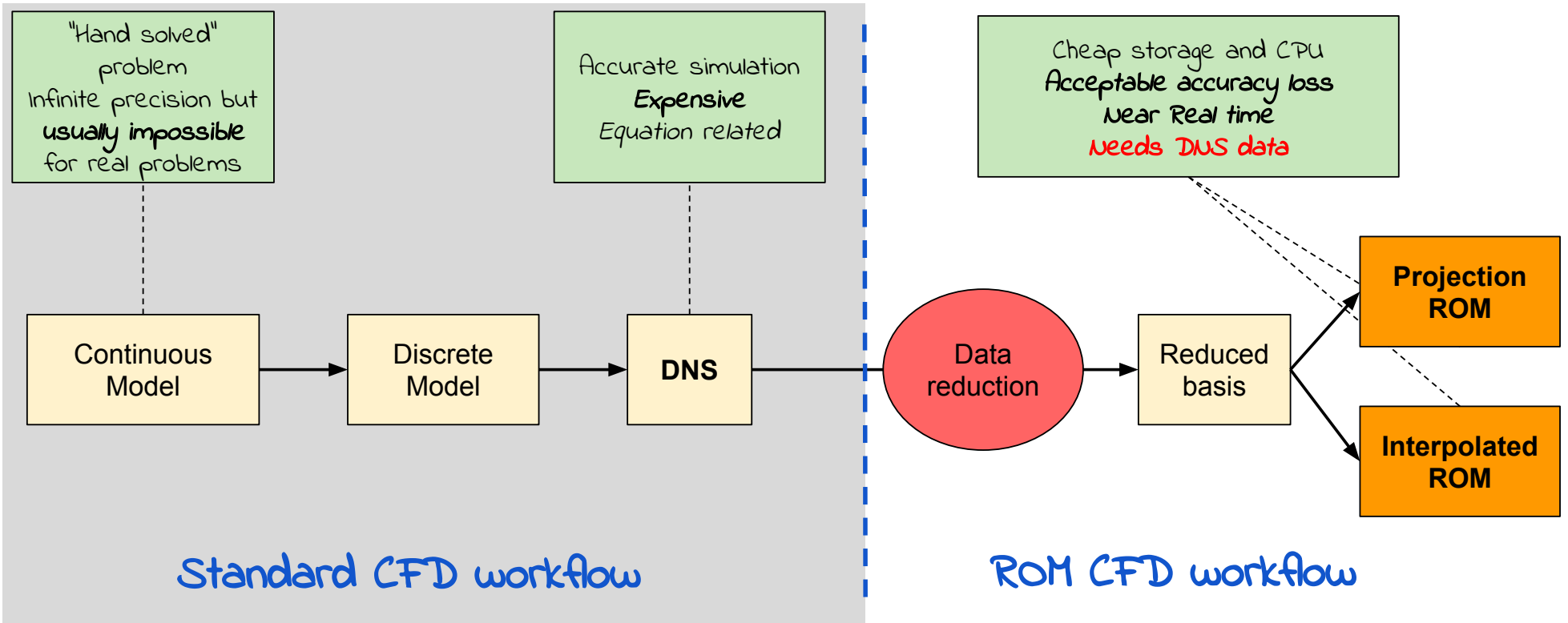




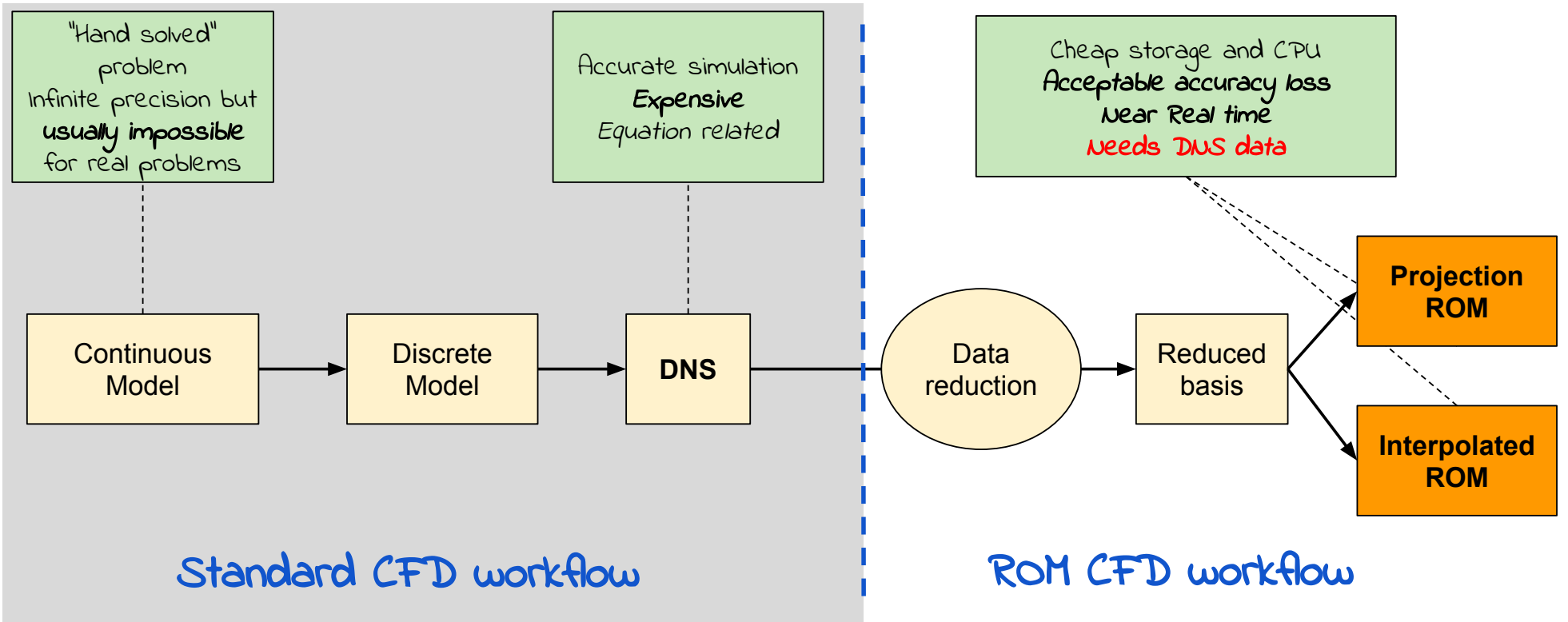




Reduced order modeling



Reduced order modeling



ROM for complex flows

Objectives

- **Input:** LDC DNS data, $Re \in [8000, 12000]$

ROM for complex flows

Objectives

- **Input:** LDC DNS data, $Re \in [8000, 12000]$
- **Analysis:** Limit cycle, within Hopf bifurcations

ROM for complex flows

Objectives

- **Input:** LDC DNS data, $Re \in [8000, 12000]$
- **Analysis:** Limit cycle, within Hopf bifurcations
- **Goal:** Evaluate $\omega(\cdot, \cdot, Re_t)$ from few Re_{donor}

ROM for complex flows

Objectives

- **Input:** LDC DNS data, $Re \in [8000, 12000]$
- **Analysis:** Limit cycle, within Hopf bifurcations
- **Goal:** Evaluate $\omega(\cdot, \cdot, Re_t)$ from few Re_{donor}
- **Ex:** $Re_d \in \{10000, 10020, 10060, 10080, 10100\}$; $Re_t = 10040$

ROM for complex flows

Objectives

- **Input:** LDC DNS data, $Re \in [8000, 12000]$
- **Analysis:** Limit cycle, within Hopf bifurcations
- **Goal:** Evaluate $\omega(\cdot, \cdot, Re_t)$ from few Re_{donor}
- **Ex:** $Re_d \in \{10000, 10020, 10060, 10080, 10100\}$; $Re_t = 10040$

ROM for complex flows

Objectives

- **Input:** LDC DNS data, $Re \in [8000, 12000]$
- **Analysis:** Limit cycle, within Hopf bifurcations
- **Goal:** Evaluate $\omega(\cdot, \cdot, Re_t)$ from few Re_{donor}
- **Ex:** $Re_d \in \{10000, 10020, 10060, 10080, 10100\}$; $Re_t = 10040$

Error definition (RMS)

$$E = \frac{||\omega_{DNS}(Re_t) - \omega_{ROM}(Re_t)||_F}{||\omega_{DNS}(Re_t)||_F}$$

A simple interpolated ROM

Basis interpolation ROM

- pydecomp gives $\omega(\boldsymbol{x}, t, Re) \approx [\mathcal{W}, \boldsymbol{\Phi}, \boldsymbol{A}, \boldsymbol{U}]$ (ex: ST-HOSVD)

A simple interpolated ROM

Basis interpolation ROM

- pydecomp gives $\omega(\boldsymbol{x}, t, Re) \approx [\mathcal{W}, \boldsymbol{\Phi}, \boldsymbol{A}, \boldsymbol{U}]$ (ex: ST-HOSVD)
- Interpolate $\boldsymbol{U}$ with standard methods (Lagrange, linear, spline)

A simple interpolated ROM

Basis interpolation ROM

- pydecomp gives $\omega(\boldsymbol{x}, t, Re) \approx [\mathcal{W}, \Phi, \boldsymbol{A}, \boldsymbol{U}]$ (ex: ST-HOSVD)
- Interpolate $\boldsymbol{U}$ with standard methods (Lagrange, linear, spline)
- Reconstruct $\omega(\boldsymbol{x_i}, t_j, Re_t)\ \forall i, j$ at target Reynolds Re_t

A simple interpolated ROM

Basis interpolation ROM

- pydecomp gives $\omega(\boldsymbol{x}, t, Re) \approx [\mathcal{W}, \Phi, \boldsymbol{A}, \boldsymbol{U}]$ (ex: ST-HOSVD)
- Interpolate $\boldsymbol{U}$ with standard methods (Lagrange, linear, spline)
- Reconstruct $\omega(\boldsymbol{x_i}, t_j, Re_t)\ \forall i, j$ at target Reynolds Re_t

A simple interpolated ROM

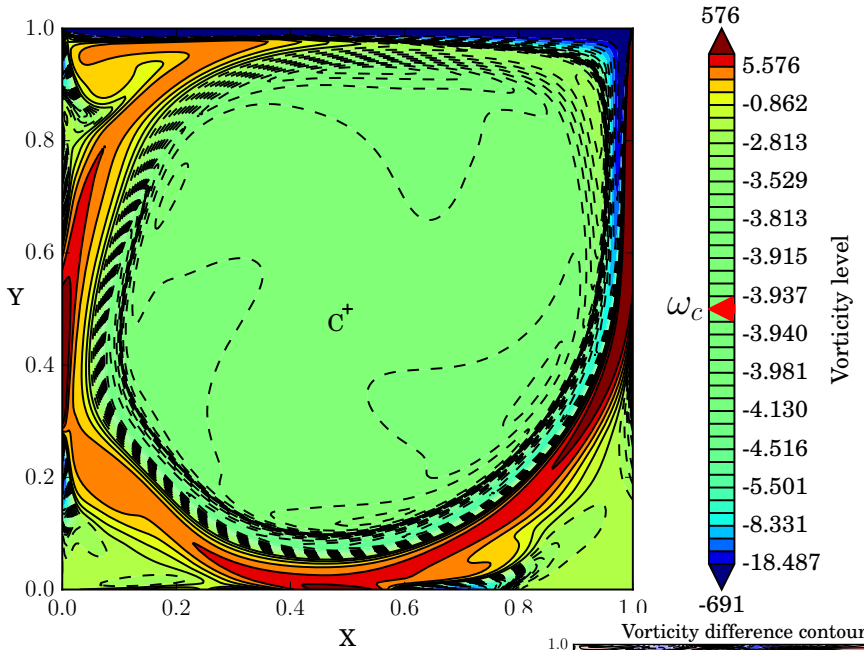
Basis interpolation ROM

- pydecomp gives $\omega(\boldsymbol{x}, t, Re) \approx [\mathcal{W}, \Phi, \boldsymbol{A}, \boldsymbol{U}]$ (ex: ST-HOSVD)
- Interpolate $\boldsymbol{U}$ with standard methods (Lagrange, linear, spline)
- Reconstruct $\omega(\boldsymbol{x_i}, t_j, Re_t)\ \forall i, j$ at target Reynolds Re_t

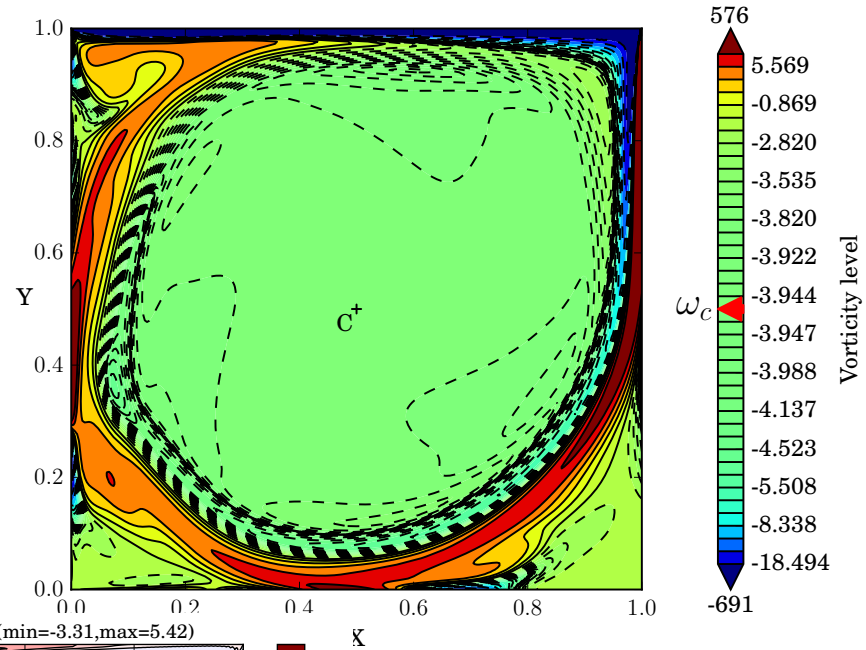
Results (RMS reconstruction error)

Linear	2.94%
Lagrange	1.49%
Spline	1.45%

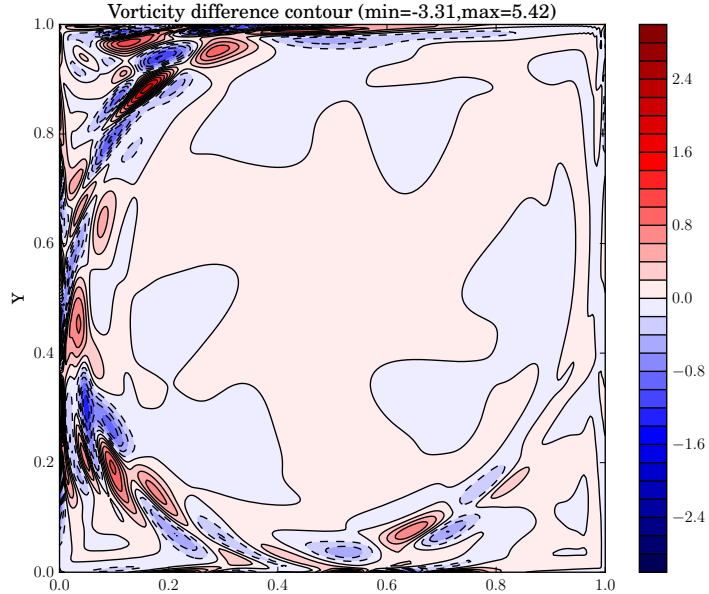
Lagrange basis interpolation ROM



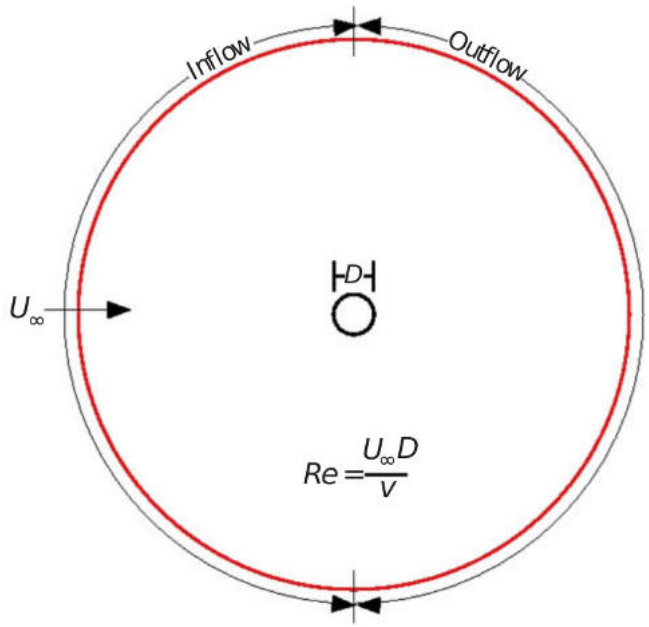
(a) DNS



b) Lagrange

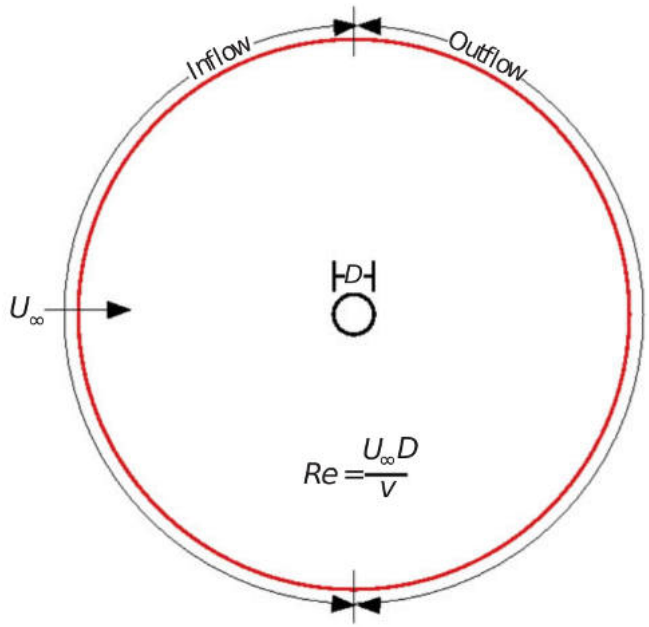


Flow past a circular cylinder



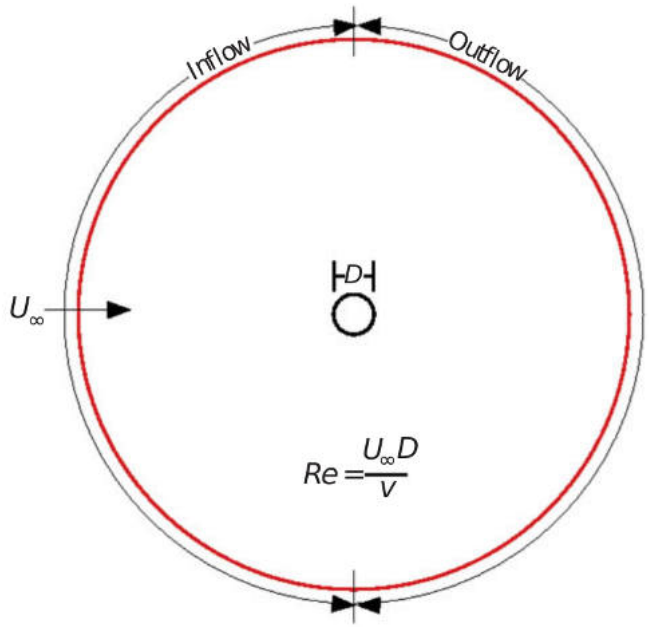
- $Re \in [55, 200]$
- Unstable flow with bifurcations

Flow past a circular cylinder



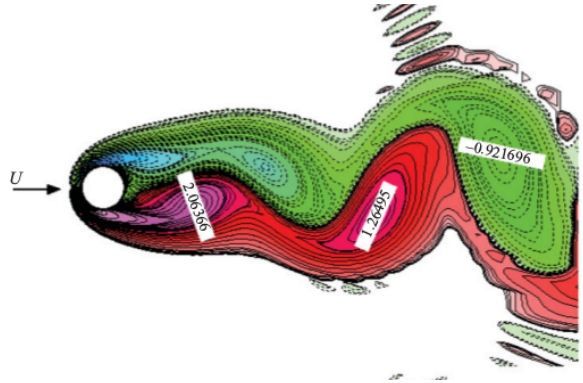
- $Re \in [55, 200]$
- Unstable flow with bifurcations
- 2D NS vorticity formulation
- High accuracy OUCS3 solver

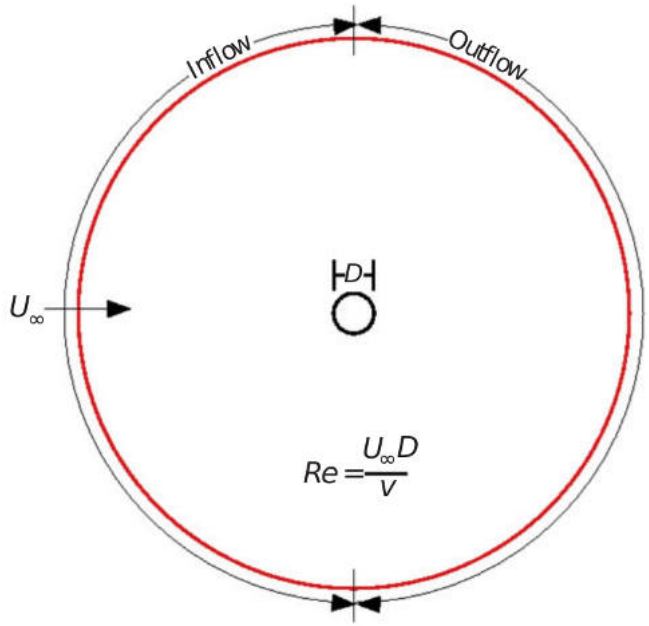
Flow past a circular cylinder

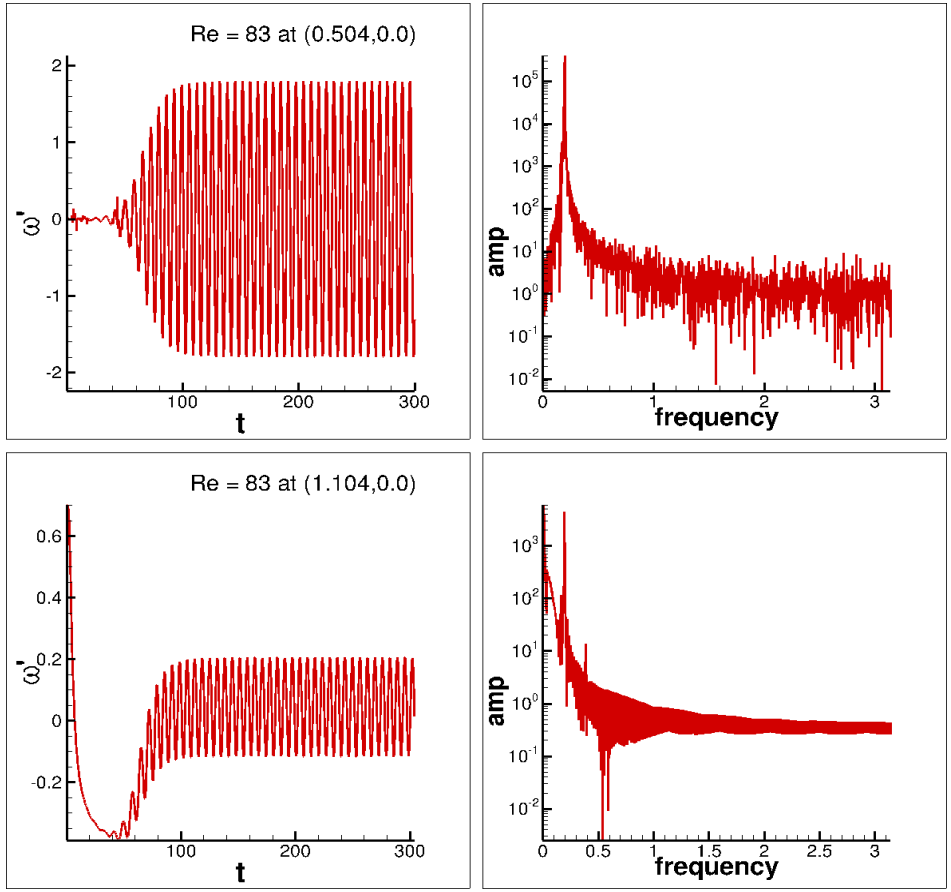


- $Re \in [55, 200]$
- Unstable flow with bifurcations
- 2D NS vorticity formulation
- High accuracy OUCS3 solver
- Cylindrical grid (400×1000)

Flow past a circular cylinder







- $Re \in [55, 200]$
- Unstable flow with bifurcations
- 2D NS vorticity formulation
- High accuracy OUCS3 solver
- Cylindrical grid (400×1000)

Time series and associated FFT at $Re = 83$ at two wake points $(0.504, 0.0)$ and $(1.104, 0.0)$.

Need for time-scaled interpolation

At point $\boldsymbol{x} = (0, 0.504)$ we compare

- DNS : $\omega(\cdot, Re_t)$
- Lagrange interp ROM : $\widetilde{\omega}(Re_t) = \mathrm{Lagrange}(\omega(Re_1), \omega(Re_2))$

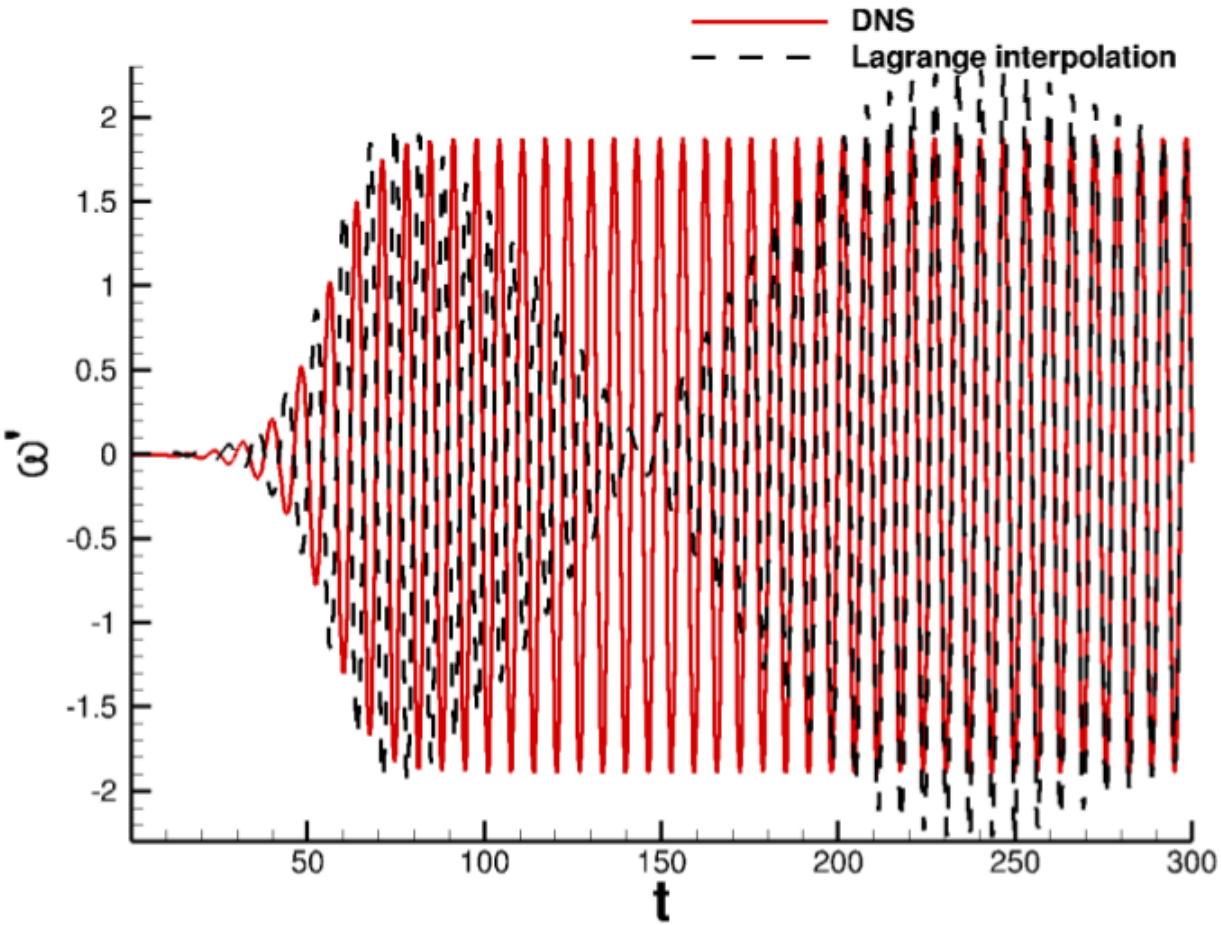


Figure: Direct interpolation of DNS vorticity time series at point (0,0.504) for flow past a circular cylinder.

Time scales in FPCC

- Strouhal number :

$$St = \frac{f\,D}{U_\infty}$$

Time scales in FPCC

- Strouhal number :

$$St = \frac{f\,D}{U_\infty}$$

- For FPCC, Fey, König and Eckelmann (1998) give the following empirical relation (on experimental data for $47 \leq Re \leq 2 \times 10^5$)

$$St = St^* + m/\sqrt{Re}$$

Time scales in FPCC

- Strouhal number :

$$St = \frac{f\,D}{U_\infty}$$

- For FPCC, Fey, König and Eckelmann (1998) give the following empirical relation (on experimental data for $47 \leq Re \leq 2 \times 10^5$)

$$St = St^* + m/\sqrt{Re}$$

- Proposed power law scaling:

$$\frac{St(Re_s)}{St(Re_b)} = \left(\frac{Re_b}{Re_s}\right)^n$$

FPCC Flow behavior

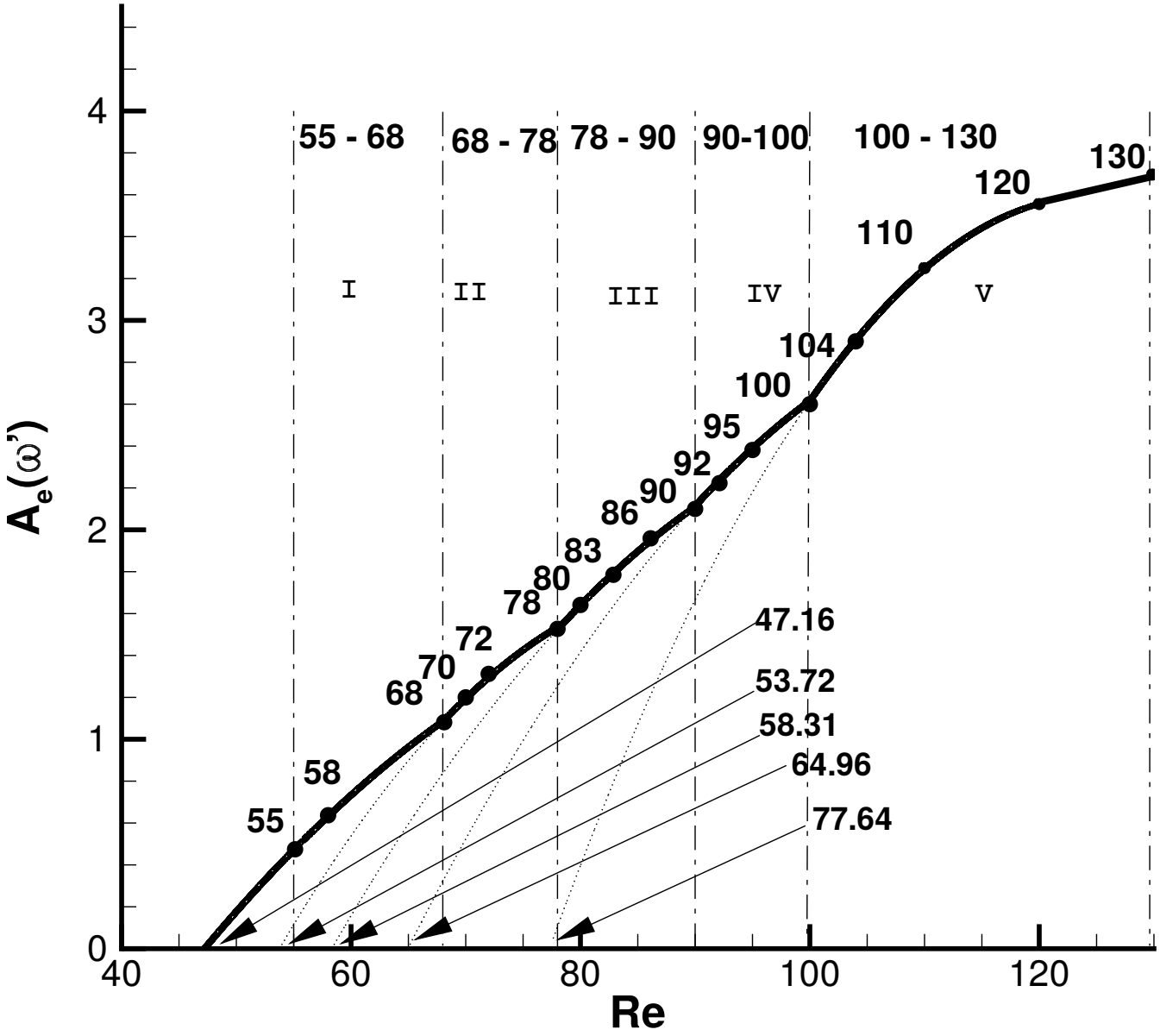


Figure: Bifurcation sequence explained via equilibrium amplitude variation with Re

FPCC Flow behavior

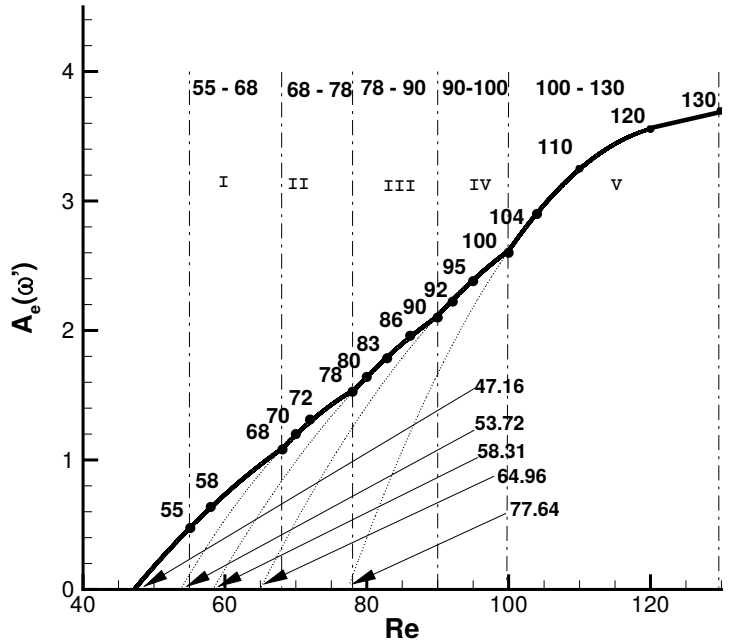


Table: Scaling Constant and Base Re_b for Different Range of Re_s for FPCC

Re Range	Scaling Constant (n)	Basic Re (Re_b)
$55 - 68$	-0.49 ± 0.02	60
$68 - 78$	-0.41 ± 0.02	72
$78 - 90$	-0.37 ± 0.02	80
$90 - 100$	-0.32 ± 0.02	95
$100 - 130$	-0.28 ± 0.02	110

Time scaling

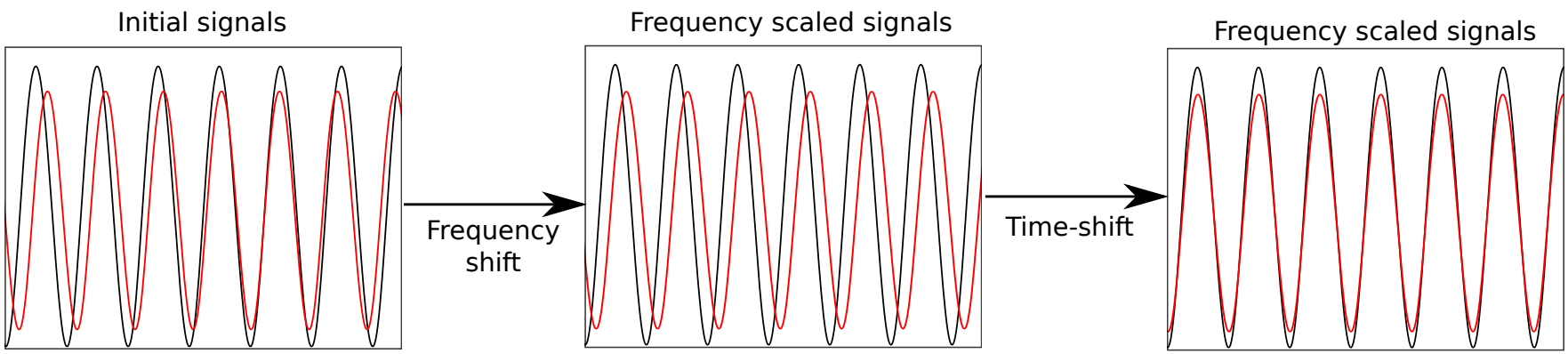
Having fixed n on a segment, time scaling reads:

$$t_s = t_b \left(\frac{Re_b}{Re_s} \right)^n + t_0(Re_b, Re_s)$$

Time scaling

Having fixed n on a segment, time scaling reads:

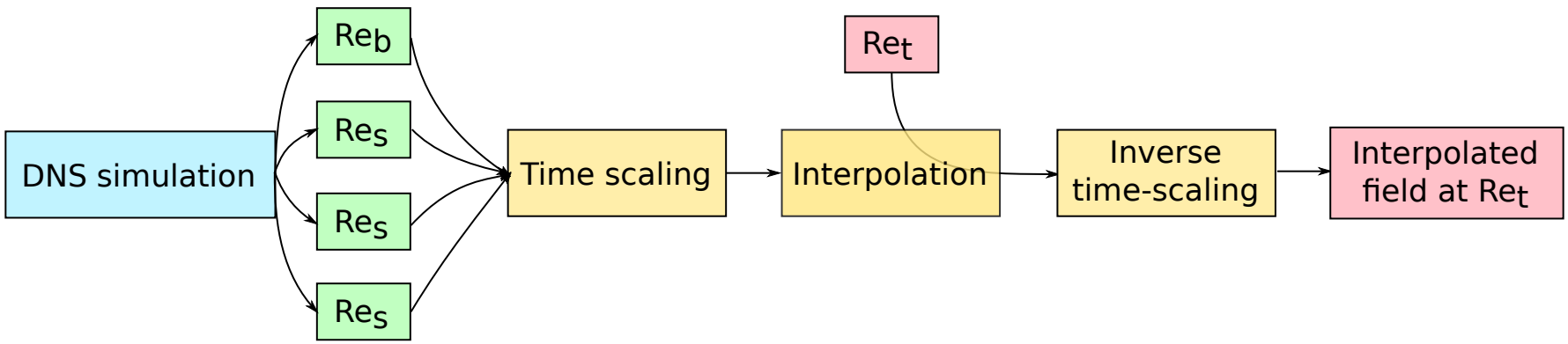
$$t_s = t_b \left(\frac{Re_b}{Re_s} \right)^n + t_0(Re_b, Re_s)$$



Time scaling

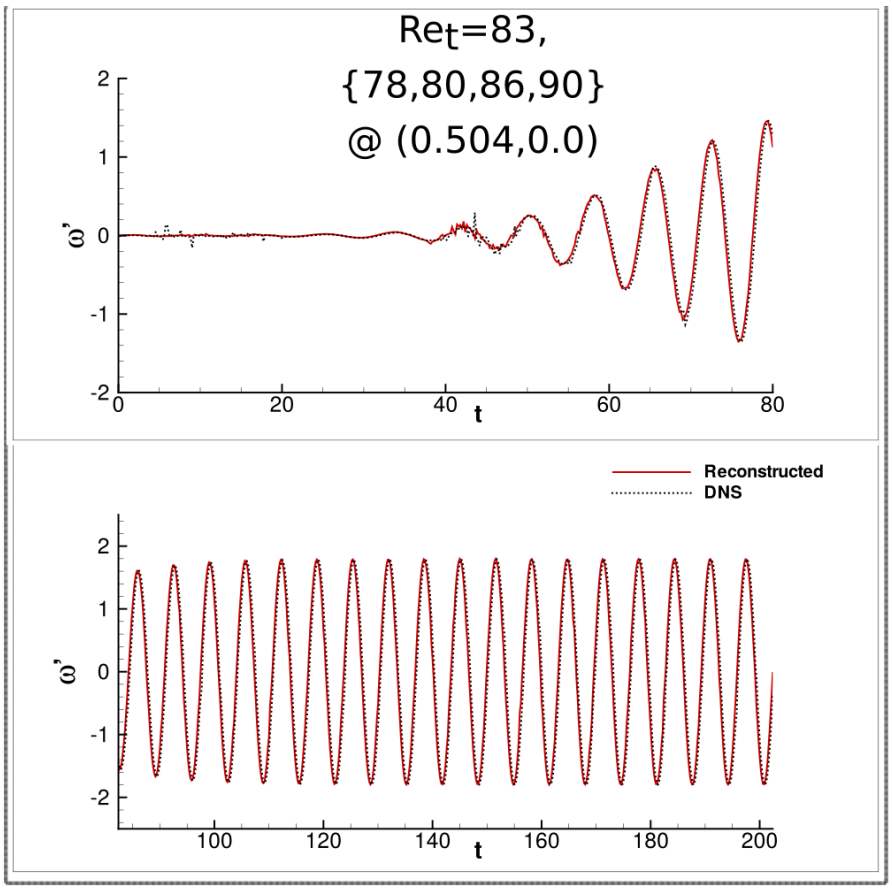
Having fixed n on a segment, time scaling reads:

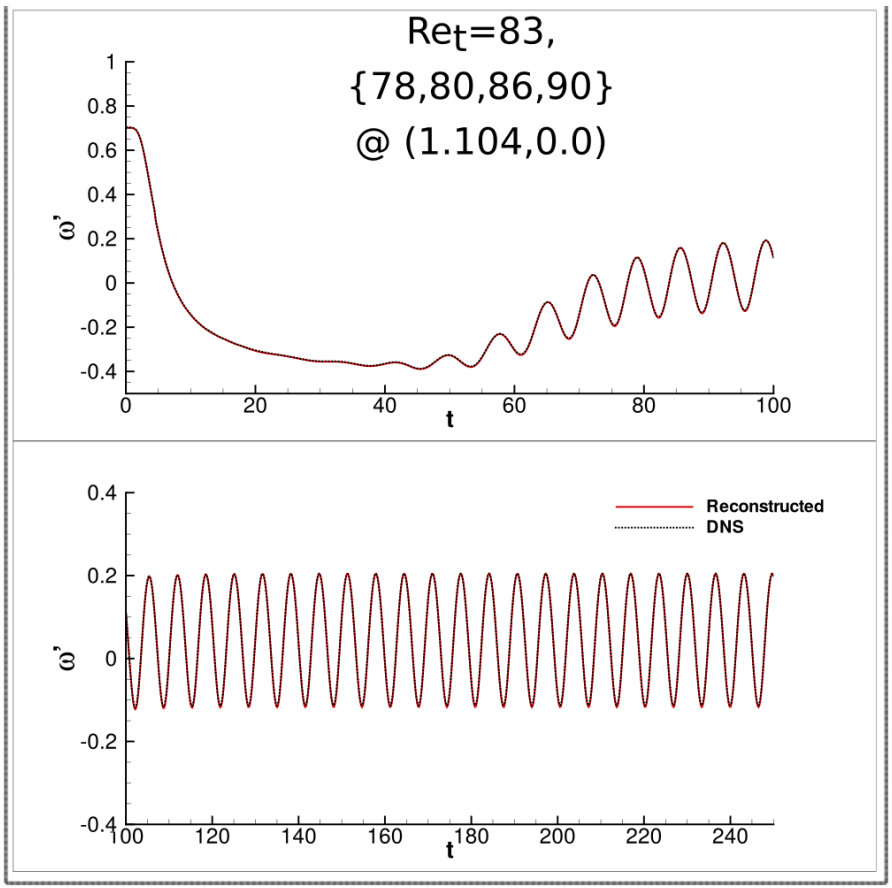
$$t_s = t_b \left(\frac{Re_b}{Re_s} \right)^n + t_0(Re_b, Re_s)$$



Schematic of the time-scaled interpolation ROM

Time-scaling applied on FPCC





Time-scaling applied on FPCC

Table: RMS Error estimates of interpolation for Re = 83

Re of Donor Points	Interpolation Error (%)
(78,80,86,90)	4.3
(72,80,86,90)	4.3
(55,80,86,90)	6.2
(55,80,86,130)	14
(55,68,72,86)	131

Time scaling overview

Time-scaling ROM

- Physics based ROM for supercritical flows (internal/external)
- Allows Lagrange interpolation directly on DNS data
- Only 3-4 snapshots required

Conclusion and perspectives

Conclusion

Data reduction

Features implemented in **`pydecomp`**:

- Tensor formats and decomposition

ROM for fluid dynamics

Conclusion

Data reduction

Features implemented in **`pydecomp`**:

- Tensor formats and decomposition
- Functional decomposition

ROM for fluid dynamics

Conclusion

Data reduction

Features implemented in **`pydecomp`**:

- Tensor formats and decomposition
- Functional decomposition
- Numerical insight and rule of thumb

ROM for fluid dynamics

Conclusion

Data reduction

Features implemented in **`pydecomp`**:

- Tensor formats and decomposition
- Functional decomposition
- Numerical insight and rule of thumb

ROM for fluid dynamics

- Study of complex LDC flow

Conclusion

Data reduction

Features implemented in **`pydecomp`**:

- Tensor formats and decomposition
- Functional decomposition
- Numerical insight and rule of thumb

ROM for fluid dynamics

- Study of complex LDC flow
- Multidimensional basis interpolation

Conclusion

Data reduction

Features implemented in **`pydecomp`**:

- Tensor formats and decomposition
- Functional decomposition
- Numerical insight and rule of thumb

ROM for fluid dynamics

- Study of complex LDC flow
- Multidimensional basis interpolation
- Time-scaling interpolation

Future work

Data reduction: pydecomp

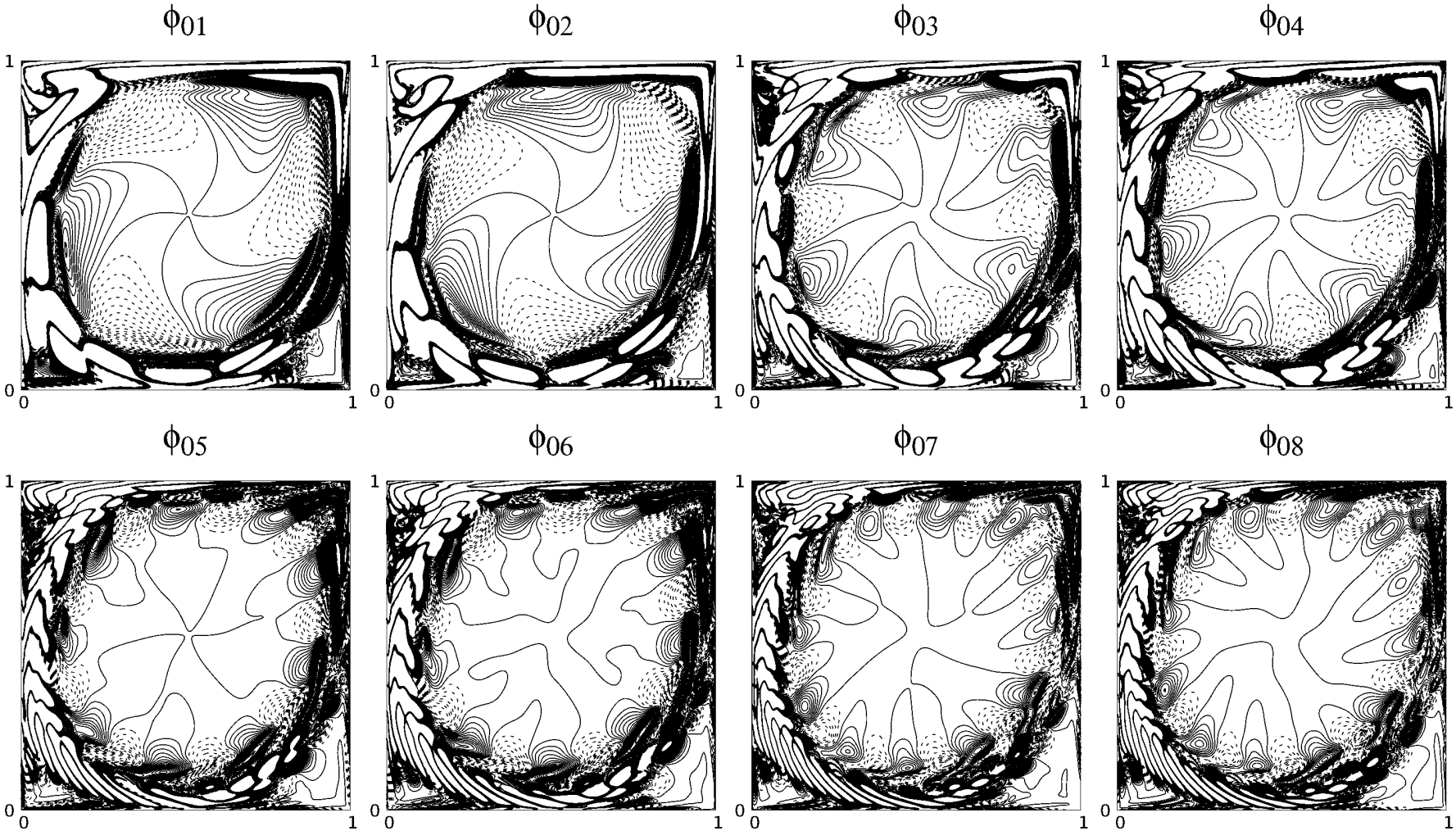
- Publicly available
- Blackbox algorithms
- Parallelism and distributed memory

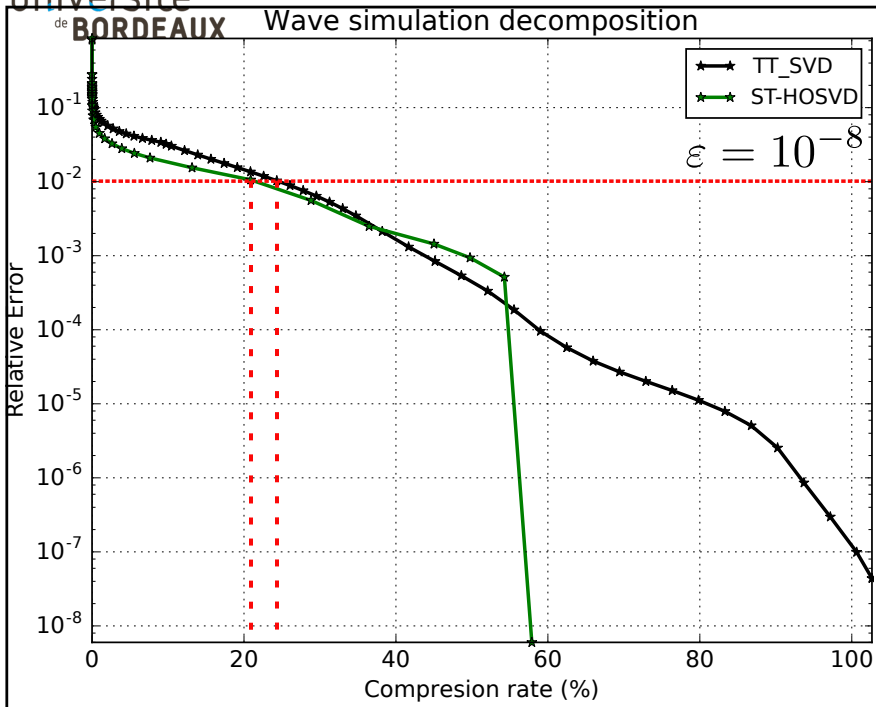
ROM

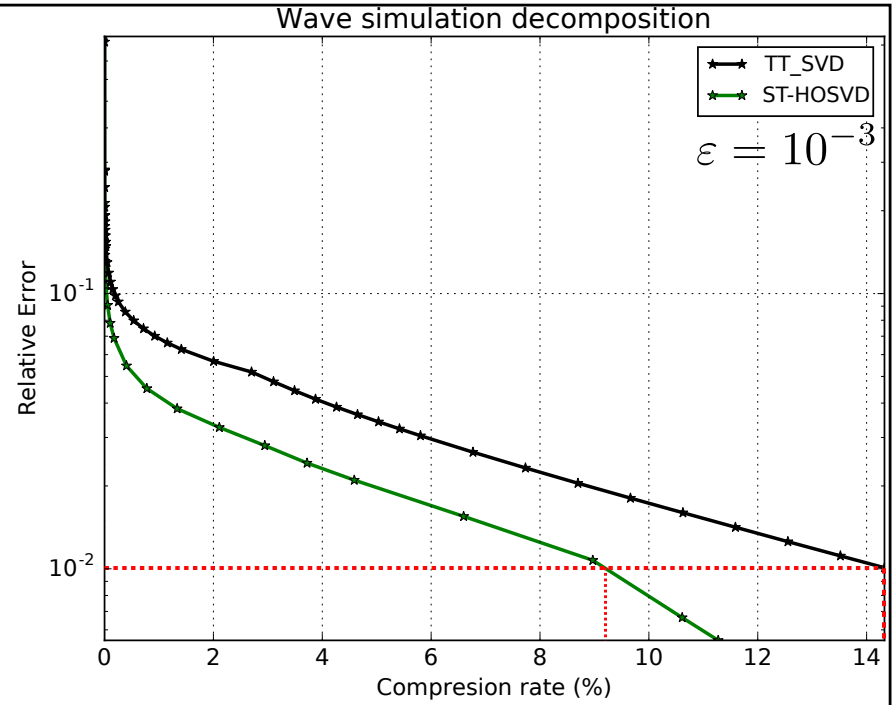
- Time-scaling on POD mode
- Non linear interpolation
- Projected ROM (Stabilized POD-G)

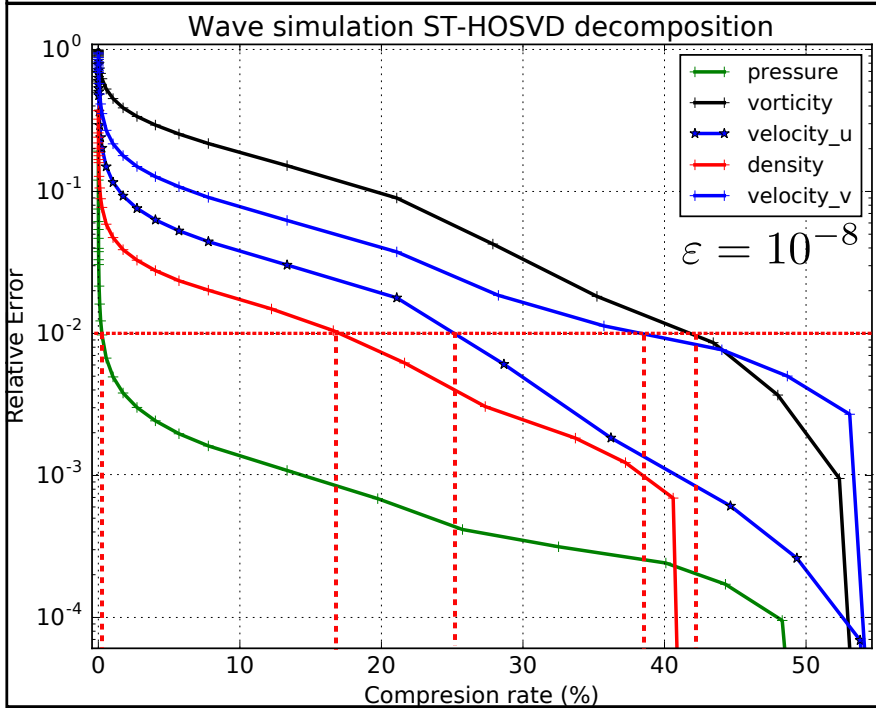
université de BORDEAUX

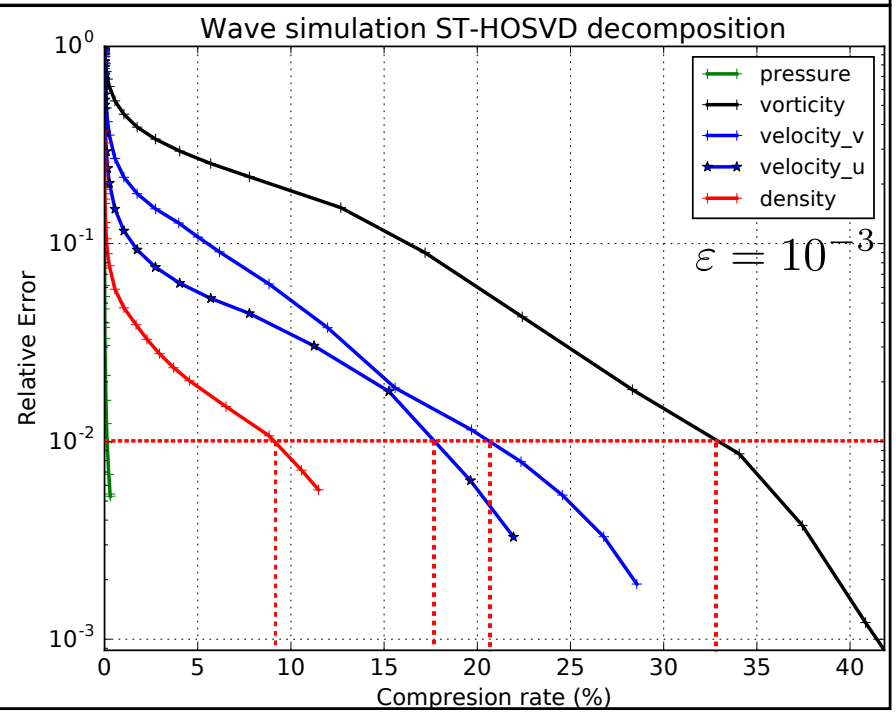
Thank you for your attention!




Wave simulation decomposition
TT_SVD
ST-HOSVD
$\varepsilon = 10^{-8}$
Relative Error
Compresion rate (%)

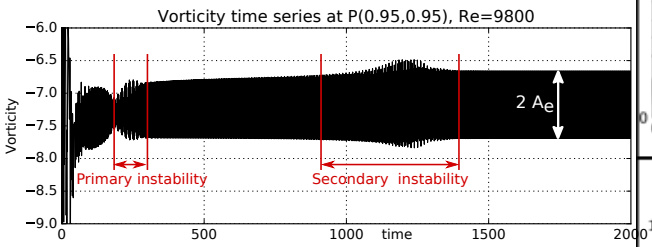

Wave simulation decomposition
TT_SVD
ST-HOSVD
$\varepsilon = 10^{-3}$
Relative Error
Compresion rate (%)


Wave simulation ST-HOSVD decomposition
pressure
vorticity
velocity_u
density
velocity_v
$\varepsilon = 10^{-8}$
Relative Error
Compresion rate (%)


Wave simulation ST-HOSVD decomposition
pressure
vorticity
velocity_v
velocity_u
density
$\varepsilon = 10^{-3}$
Relative Error
Compresion rate (%)

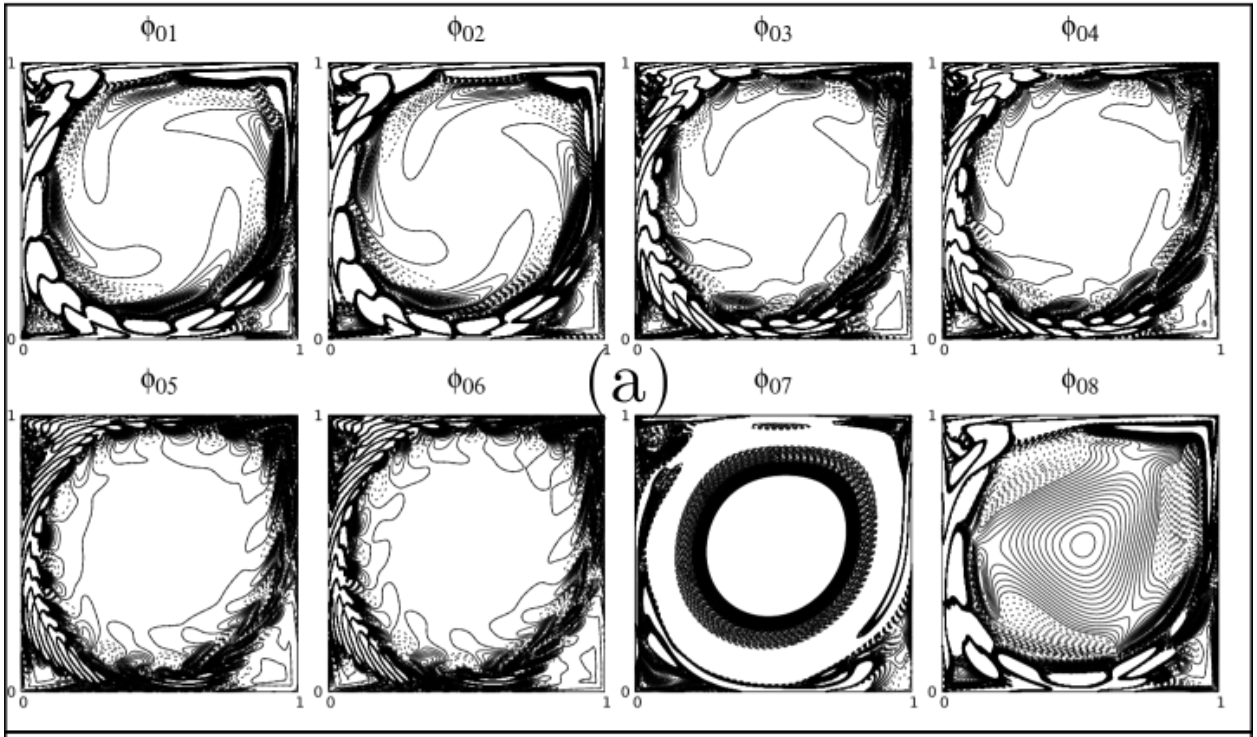
POD, an analysis tool

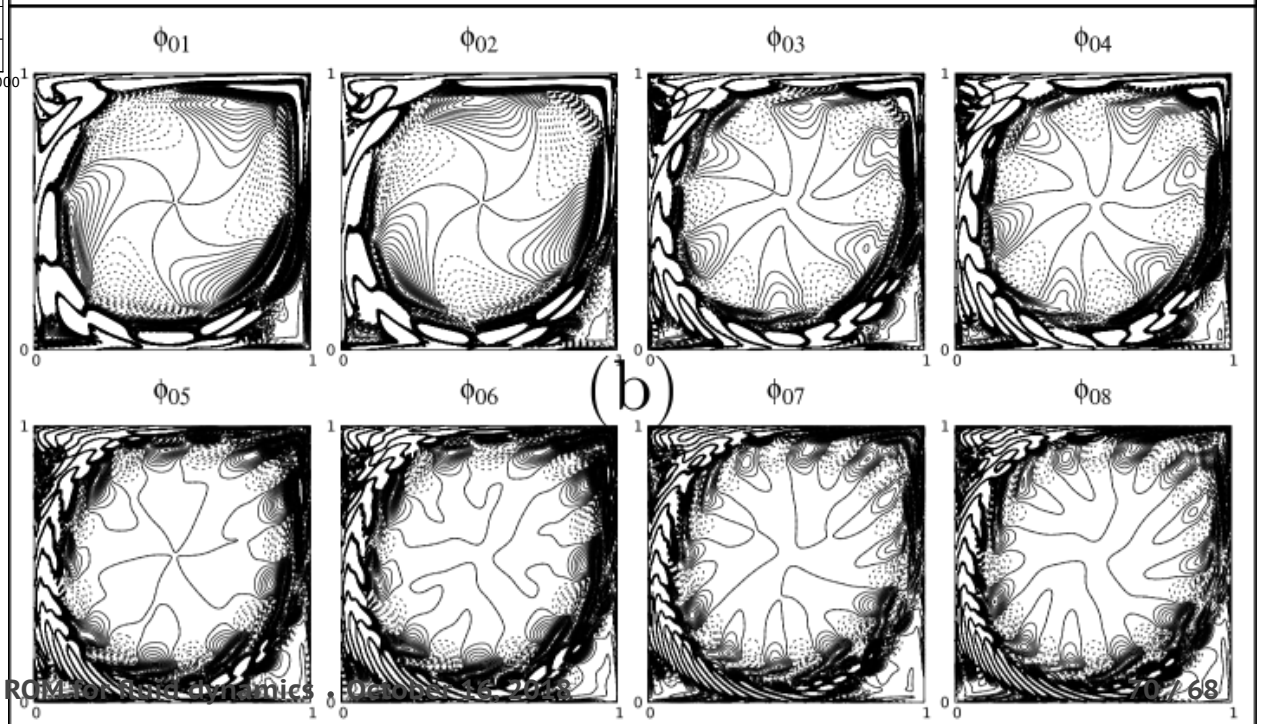
POD, an analysis tool



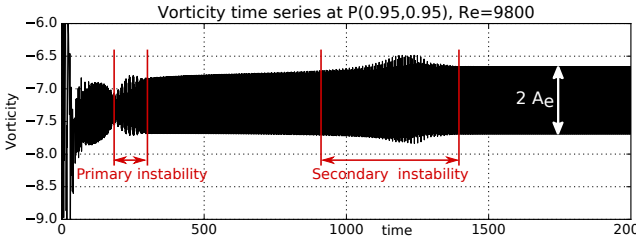
(a) $t \in [500, 600]$

(b) $t \in [1900, 2000]$



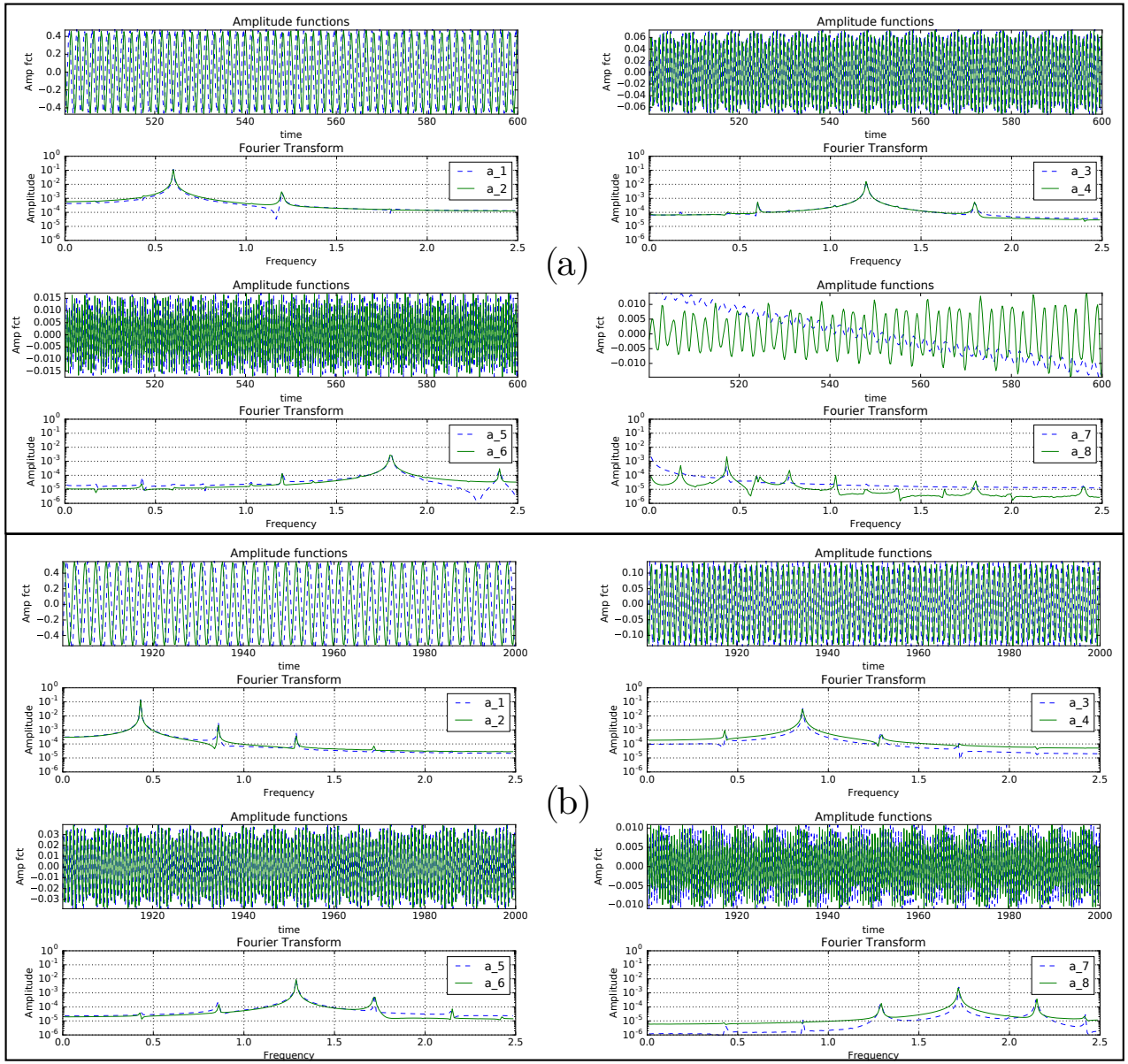


POD, an analysis tool

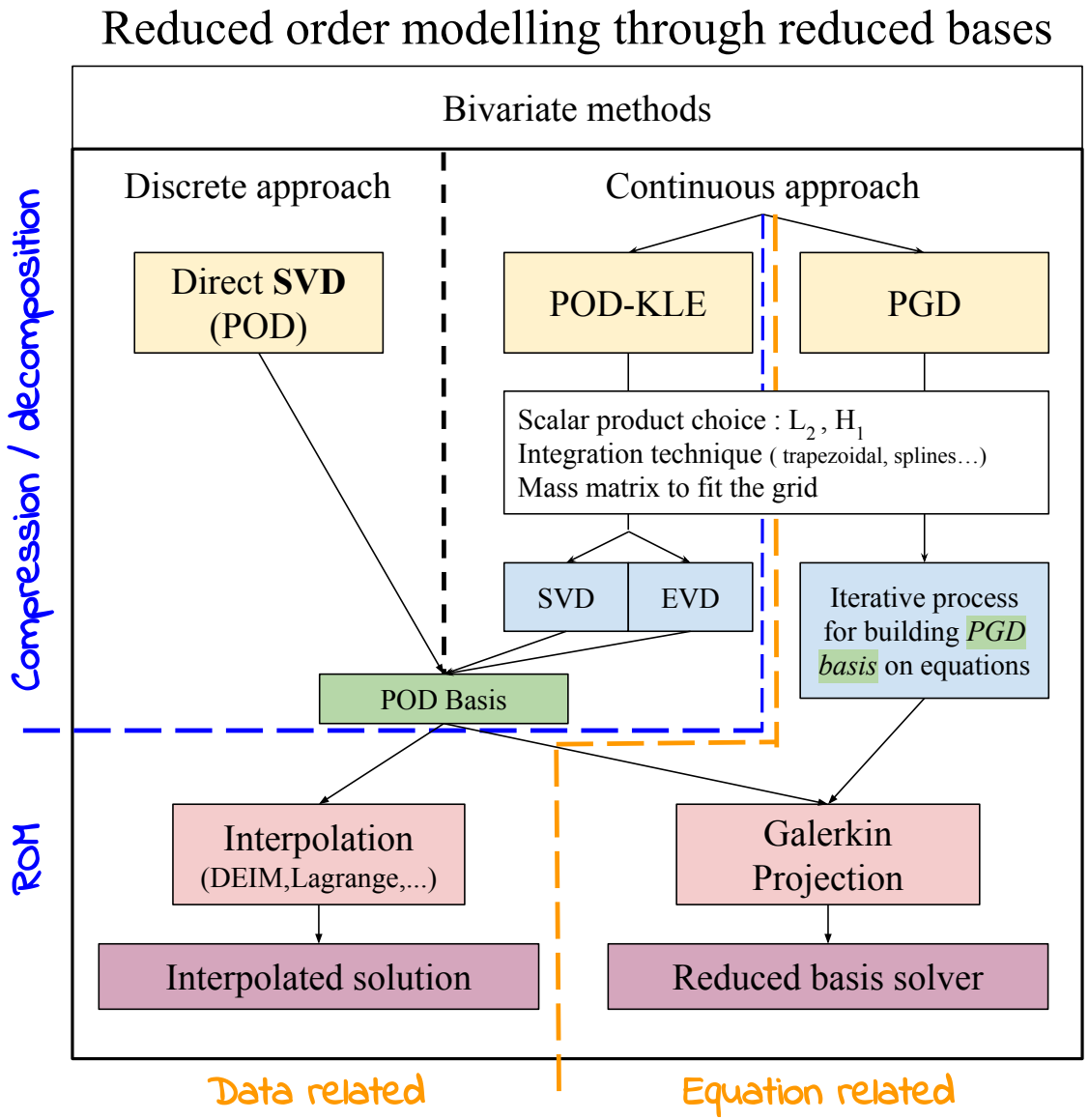


(a) $t \in [500, 600]$

(b) $t \in [1900, 2000]$



Standard 2D ROM building process



Stabilized POD-Galerkin

Navier-Stokes weak form

$$\begin{cases} \dfrac{d}{dt}(\boldsymbol{u},\boldsymbol{v}) + b(\boldsymbol{u},\boldsymbol{u},\boldsymbol{v}) + \nu(\nabla\boldsymbol{u},\nabla\boldsymbol{v}) - (p,\nabla\cdot\boldsymbol{v}) &= \langle \boldsymbol{f},\boldsymbol{v}\rangle & \forall \boldsymbol{v}\in\boldsymbol{x},\ \text{in } \mathcal{D}'(0,T), \\ (\nabla\cdot\boldsymbol{u},q) &= 0 & \forall q\in Q,\ \text{a.e. in } (0,T), \end{cases}$$

We consider the following space for the POD setting:

$$\boldsymbol{x}^r = \operatorname{span}\{\varphi_1,\ldots,\varphi_r\}.$$

Stabilized POD-Galerkin

Navier-Stokes weak form

$$\begin{cases} \dfrac{d}{dt}(\boldsymbol{u},\boldsymbol{v}) + b(\boldsymbol{u},\boldsymbol{u},\boldsymbol{v}) + \nu(\nabla\boldsymbol{u},\nabla\boldsymbol{v}) - (p,\nabla\cdot\boldsymbol{v}) &= \langle \boldsymbol{f},\boldsymbol{v}\rangle & \forall \boldsymbol{v}\in\boldsymbol{x},\ \text{in } \mathcal{D}'(0,T), \\ (\nabla\cdot\boldsymbol{u},q) &= 0 & \forall q\in Q,\ \text{a.e. in } (0,T), \end{cases}$$

We consider the following space for the POD setting:

$$\boldsymbol{x}^r = \text{span}\left\{\varphi_1,\ldots,\varphi_r\right\}.$$

We look for $\boldsymbol{u}(\boldsymbol{x},t) \approx \boldsymbol{u}_r(\boldsymbol{x},t) = \boldsymbol{u}_D(\boldsymbol{x}) + \widetilde{\boldsymbol{u}}_r(\boldsymbol{x},t) = \boldsymbol{u}_D(\boldsymbol{x}) + \sum_{i=1}^{r} a_i(t)\varphi_i(\boldsymbol{x})$,

Stabilized POD-Galerkin

Navier-Stokes weak form

$$\begin{cases} \dfrac{d}{dt}(\boldsymbol{u},\boldsymbol{v}) + b(\boldsymbol{u},\boldsymbol{u},\boldsymbol{v}) + \nu(\nabla\boldsymbol{u},\nabla\boldsymbol{v}) - (p,\nabla\cdot\boldsymbol{v}) &= \langle \boldsymbol{f},\boldsymbol{v}\rangle & \forall \boldsymbol{v}\in\boldsymbol{x},\ \text{in } \mathcal{D}'(0,T), \\ (\nabla\cdot\boldsymbol{u},q) &= 0 & \forall q\in Q,\ \text{a.e. in } (0,T), \end{cases}$$

We consider the following space for the POD setting:

$$\boldsymbol{x}^r = \text{span}\left\{\varphi_1,\ldots,\varphi_r\right\}.$$

We look for $\boldsymbol{u}(\boldsymbol{x},t) \approx \boldsymbol{u}_r(\boldsymbol{x},t) = \boldsymbol{u}_D(\boldsymbol{x}) + \widetilde{\boldsymbol{u}}_r(\boldsymbol{x},t) = \boldsymbol{u}_D(\boldsymbol{x}) + \sum_{i=1}^{r} a_i(t)\varphi_i(\boldsymbol{x})$,
We introduce the stabilization term

$$(P_R'(\widetilde{\boldsymbol{u}}_r\cdot\ \nabla\widetilde{\boldsymbol{u}}_r), P_R'(\widetilde{\boldsymbol{u}}_r\cdot\nabla\varphi))_\tau$$

Stabilized POD-Galerkin

Navier-Stokes weak form

$$\begin{cases} \dfrac{d}{dt}(\boldsymbol{u},\boldsymbol{v}) + b(\boldsymbol{u},\boldsymbol{u},\boldsymbol{v}) + \nu(\nabla\boldsymbol{u},\nabla\boldsymbol{v}) - (p,\nabla\cdot\boldsymbol{v}) &= \langle \boldsymbol{f},\boldsymbol{v}\rangle & \forall \boldsymbol{v}\in\boldsymbol{x},\ \text{in } \mathcal{D}'(0,T), \\ (\nabla\cdot\boldsymbol{u},q) &= 0 & \forall q\in Q,\ \text{a.e. in } (0,T), \end{cases}$$

We consider the following space for the POD setting:

$$\boldsymbol{x}^r = \text{span}\left\{\varphi_1,\ldots,\varphi_r\right\}.$$

We look for $\boldsymbol{u}(\boldsymbol{x},t) \approx \boldsymbol{u}_r(\boldsymbol{x},t) = \boldsymbol{u}_D(\boldsymbol{x}) + \widetilde{\boldsymbol{u}}_r(\boldsymbol{x},t) = \boldsymbol{u}_D(\boldsymbol{x}) + \sum_{i=1}^{r} a_i(t)\varphi_i(\boldsymbol{x})$,
We introduce the stabilization term

$$(P'_R(\widetilde{\boldsymbol{u}}_r\cdot\ \nabla\widetilde{\boldsymbol{u}}_r), P'_R(\widetilde{\boldsymbol{u}}_r\cdot\nabla\varphi))_\tau$$

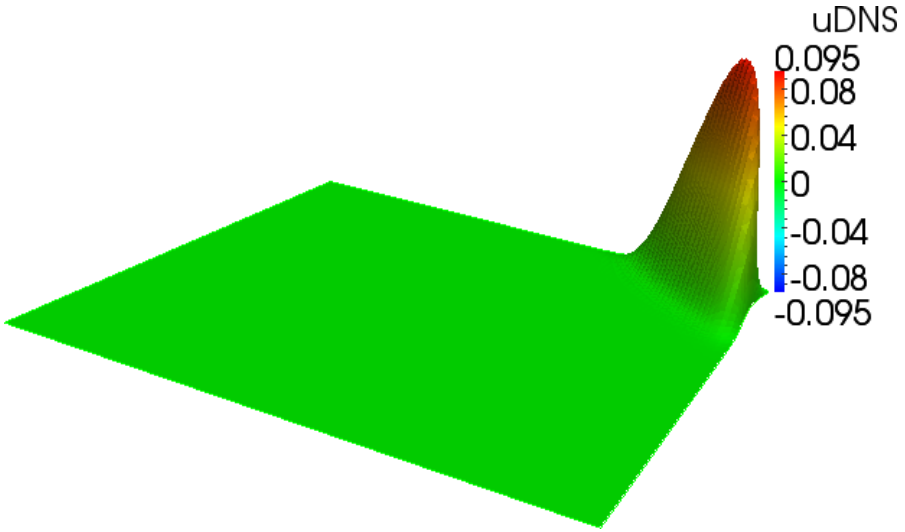
Then the Streamline-Derivative PODG reads:

$$\begin{aligned} & \frac{d}{dt}(\widetilde{\boldsymbol{u}}_r,\varphi) + b(\boldsymbol{u}_D,\widetilde{\boldsymbol{u}}_r,\varphi) + b(\widetilde{\boldsymbol{u}}_r,\boldsymbol{u}_D,\varphi) + b(\widetilde{\boldsymbol{u}}_r,\widetilde{\boldsymbol{u}}_r,\varphi) + \nu(\nabla\widetilde{\boldsymbol{u}}_r,\nabla\varphi) \\ + \quad & (P'_R(\widetilde{\boldsymbol{u}}_r\cdot\ \nabla\widetilde{\boldsymbol{u}}_r), P'_R(\widetilde{\boldsymbol{u}}_r\cdot\nabla\varphi))_\tau = \langle \boldsymbol{F},\varphi\rangle, \end{aligned}$$

Stabilized PODG in action

A convection-dominated convection-diffusion-reaction problem

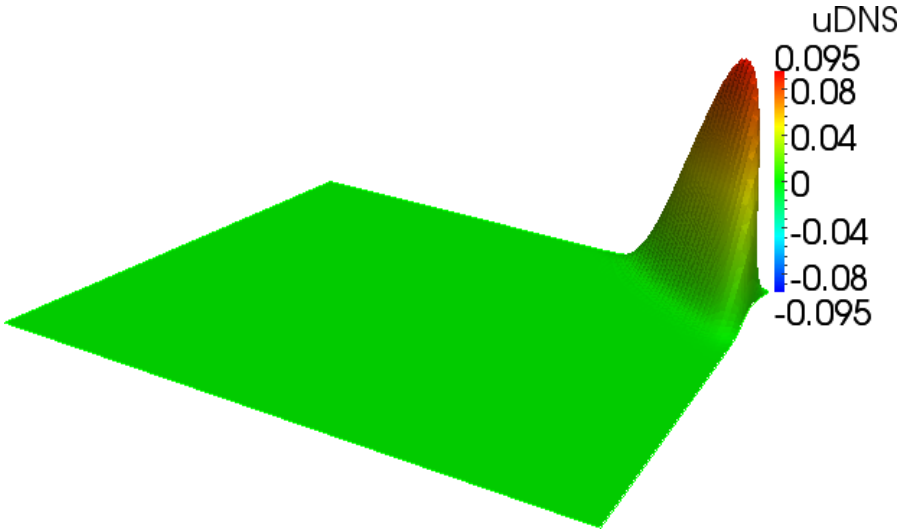
$$\begin{cases} \partial_t u - \nu \Delta u + \boldsymbol{b} \cdot \nabla u + g u = f & \text{in } [0,T] \times \Omega \\ u(x,0) = u_0(x) & \text{in } \Omega \\ u(x,t) = 0 & \text{on } [0,T] \times \partial\Omega \end{cases}$$

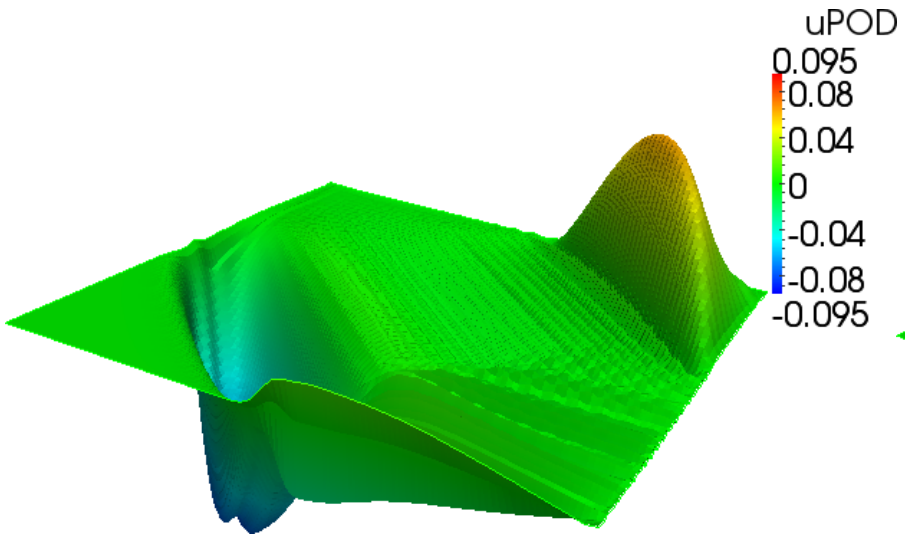


Stabilized PODG in action

A convection-dominated convection-diffusion-reaction problem

$$\begin{cases} \partial_t u - \nu \Delta u + \boldsymbol{b} \cdot \nabla u + g u = f & \text{in } [0,T] \times \Omega \\ u(x,0) = u_0(x) & \text{in } \Omega \\ u(x,t) = 0 & \text{on } [0,T] \times \partial\Omega \end{cases}$$





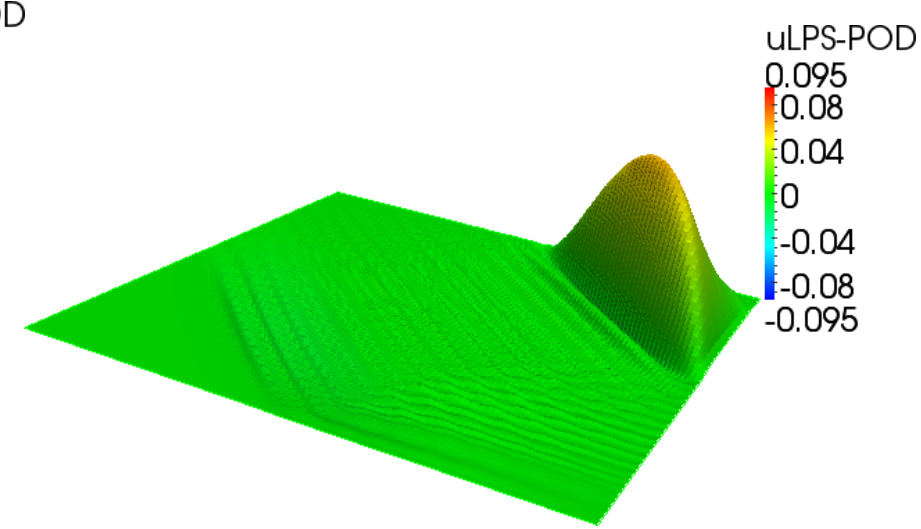
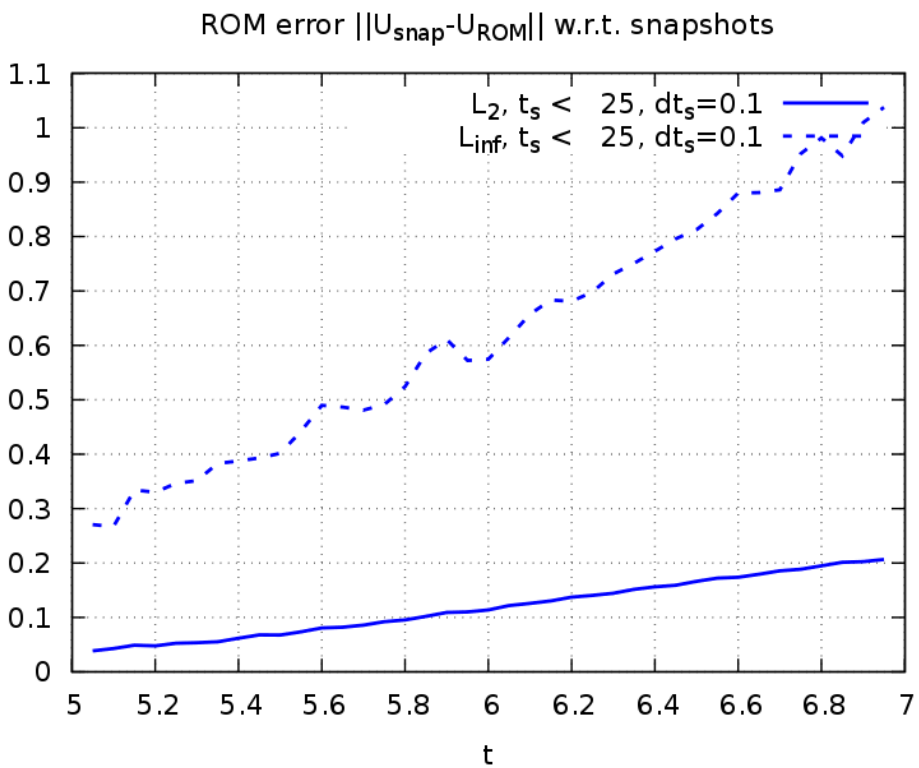
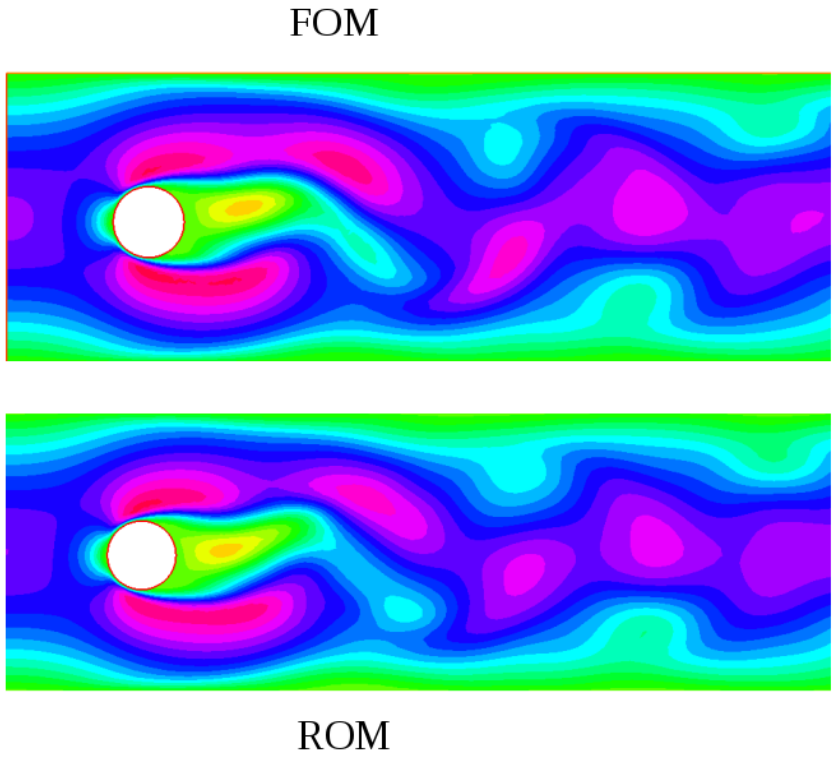


Figure: Numerical solution at $T = 1.25$:
DNS (top), POD model (bottom left), LPS-POD model (bottom right).

NS SD-PODG preliminary results





université de BORDEAUX

Thank you for your attention!