NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

IHPC Seminar, 04/02/2020

Breaking the curse of dimensionality from large data to Reduced Order Models

Dr. Lucas Lestandi

Research Fellow, SPMS-MAS, Nanyang Technological University, Supervisor : LiLian Wang

This work was conducted with

Mejdi Azaïez (U. Bordeaux), Tomás Chacón (U. Sevilla), Tapan Sengupta (IIT Kanpur) and Li-Lian Wang (NTU)

Agency for Science, Technology and Research SINGAPORE

Context

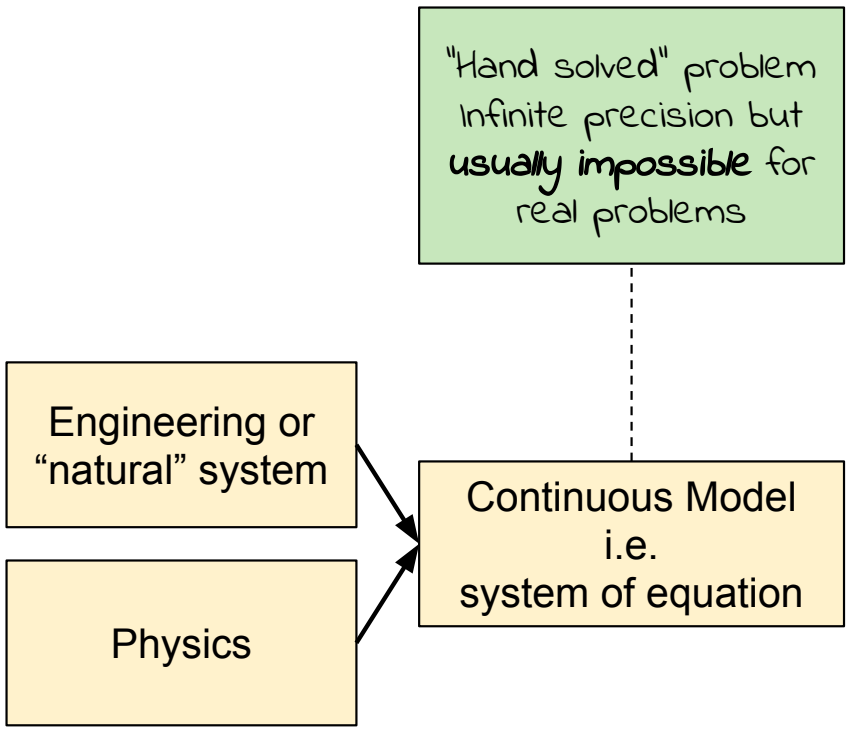
Why do we need Reduced Order Modeling and Low Rank Approximation?

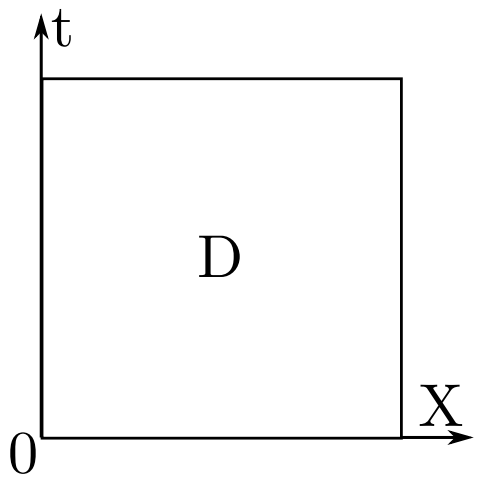
Context

Why do we need Reduced Order Modeling and Low Rank Approximation?

- the curse of dimensionality
- reduce data
- reduce model

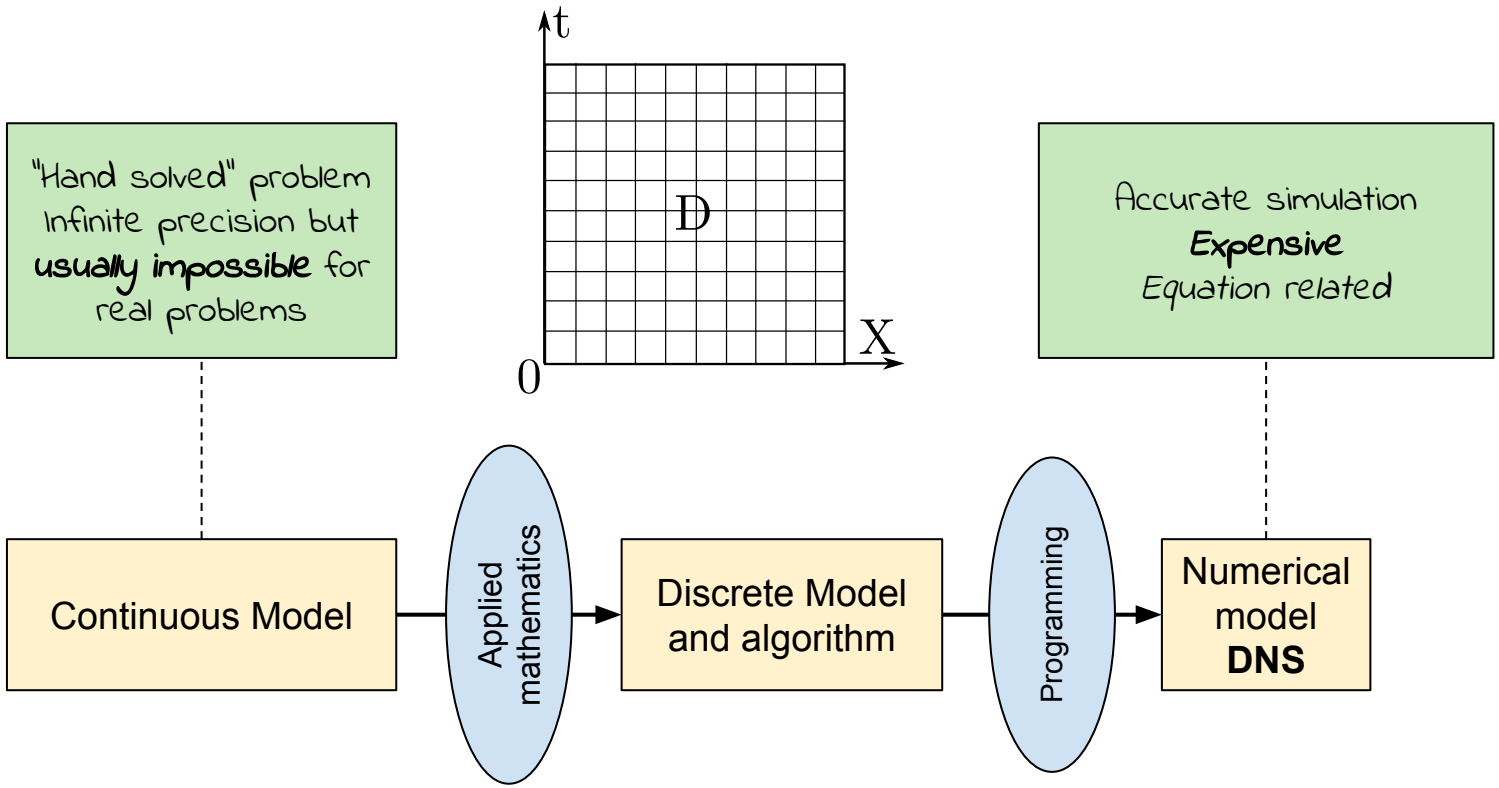
Numerical simulation in fluids





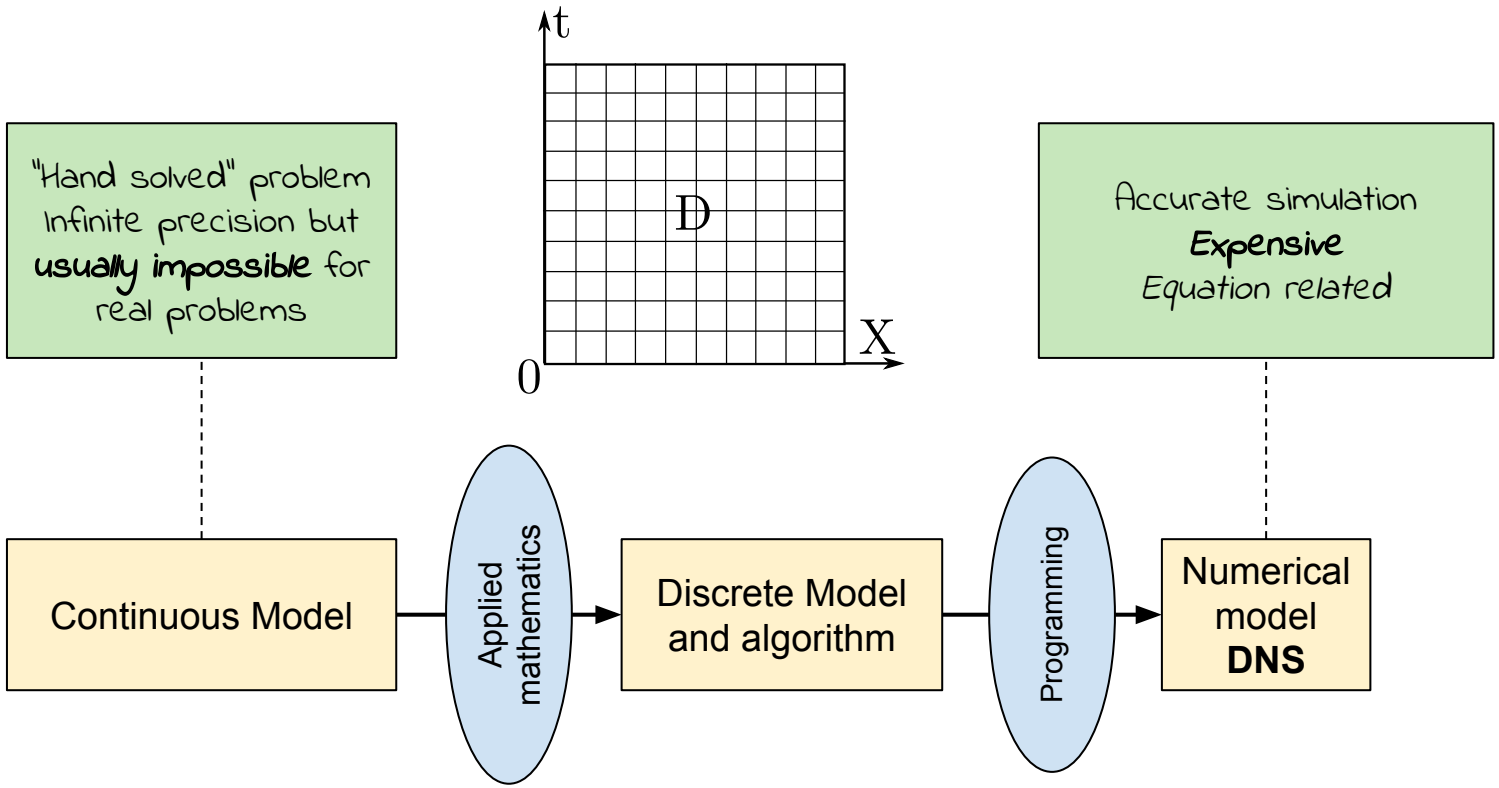
$$\frac{\partial v}{\partial t} + c\frac{\partial v}{\partial x} = 0$$

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

Big Data ?

Modern Super Computers

Scale	Name	Loc.	Country	Cores	Pflops	Power (kW)
Local	Condor	TREFLE,I2M	FR	400	NA	NA
National	ASPIRE 1	NSCC	SG	30,912	1	NA
National	Occigen	CIMES	FR	80,000	2,5	1400
World	Summit	Oak Ridge	USA	2,414,592	148	10,096

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Big Data ?

Modern Super Computers

Scale	Name	Loc.	Country	Cores	Pflops	Power (kW)
Local	Condor	TREFLE,I2M	FR	400	NA	NA
National	ASPIRE 1	NSCC	SG	30,912	1	NA
National	Occigen	CIMES	FR	80,000	2,5	1400
World	Summit	Oak Ridge	USA	2,414,592	148	10,096

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

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Reduced order modeling and data reduction

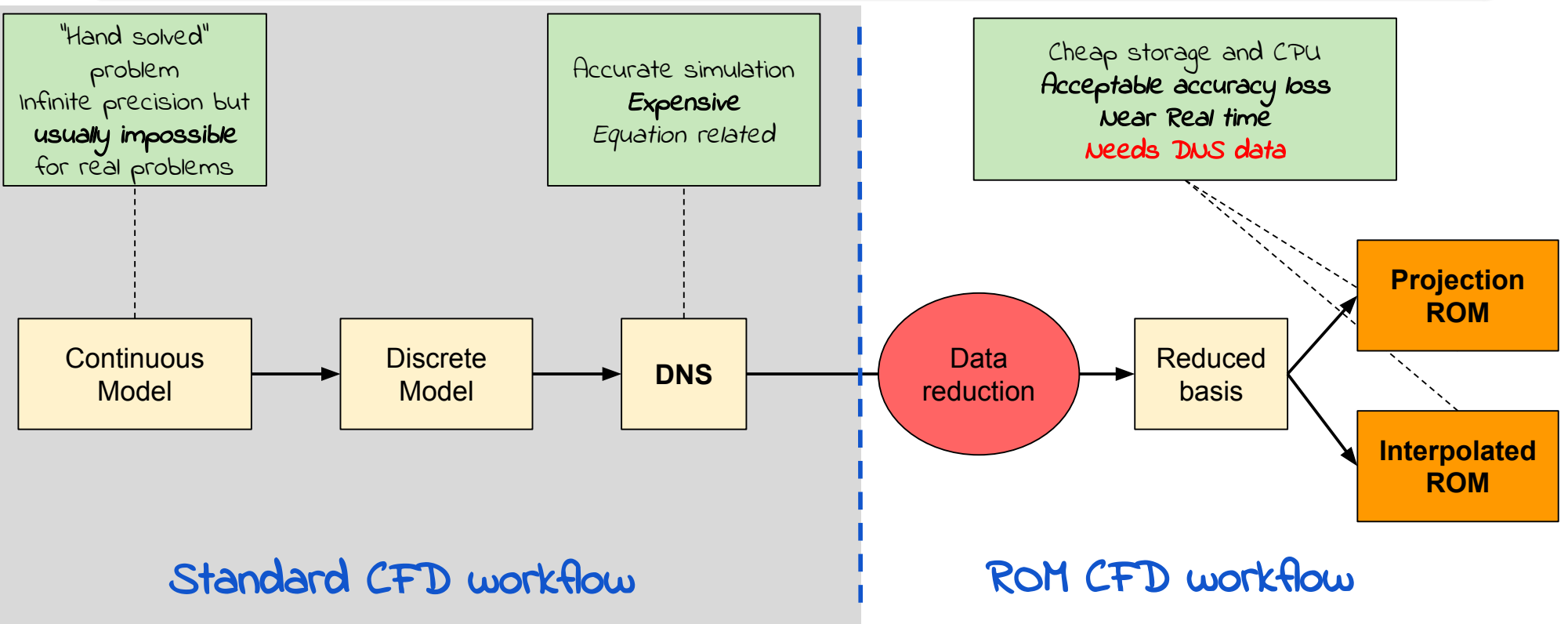
Change of paradigm

- 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 $r \ll N$



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Outline

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Data decomposition

- Bivariate decomposition
- Tensor formats and approximation
- Numerics

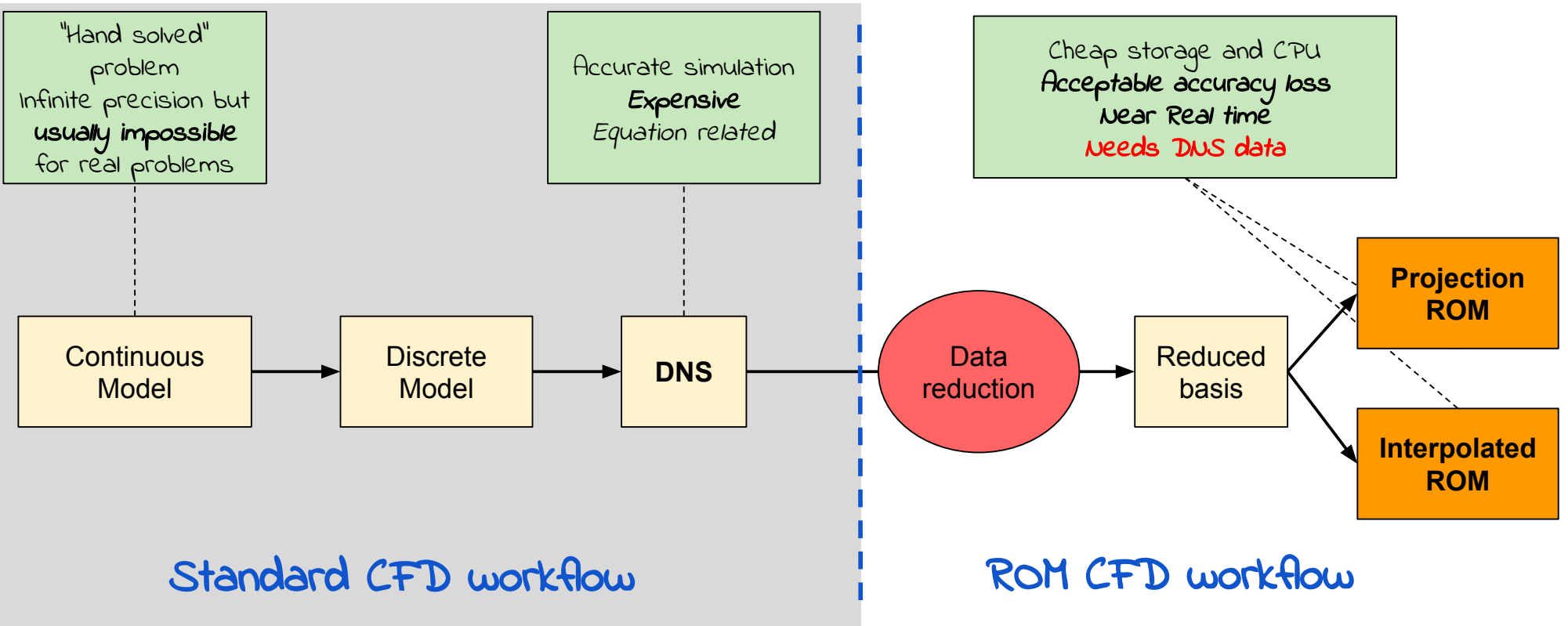
Reduced order models for complex flows

Research Project: Deep Reduced Order Model

Conclusion and perspectives

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

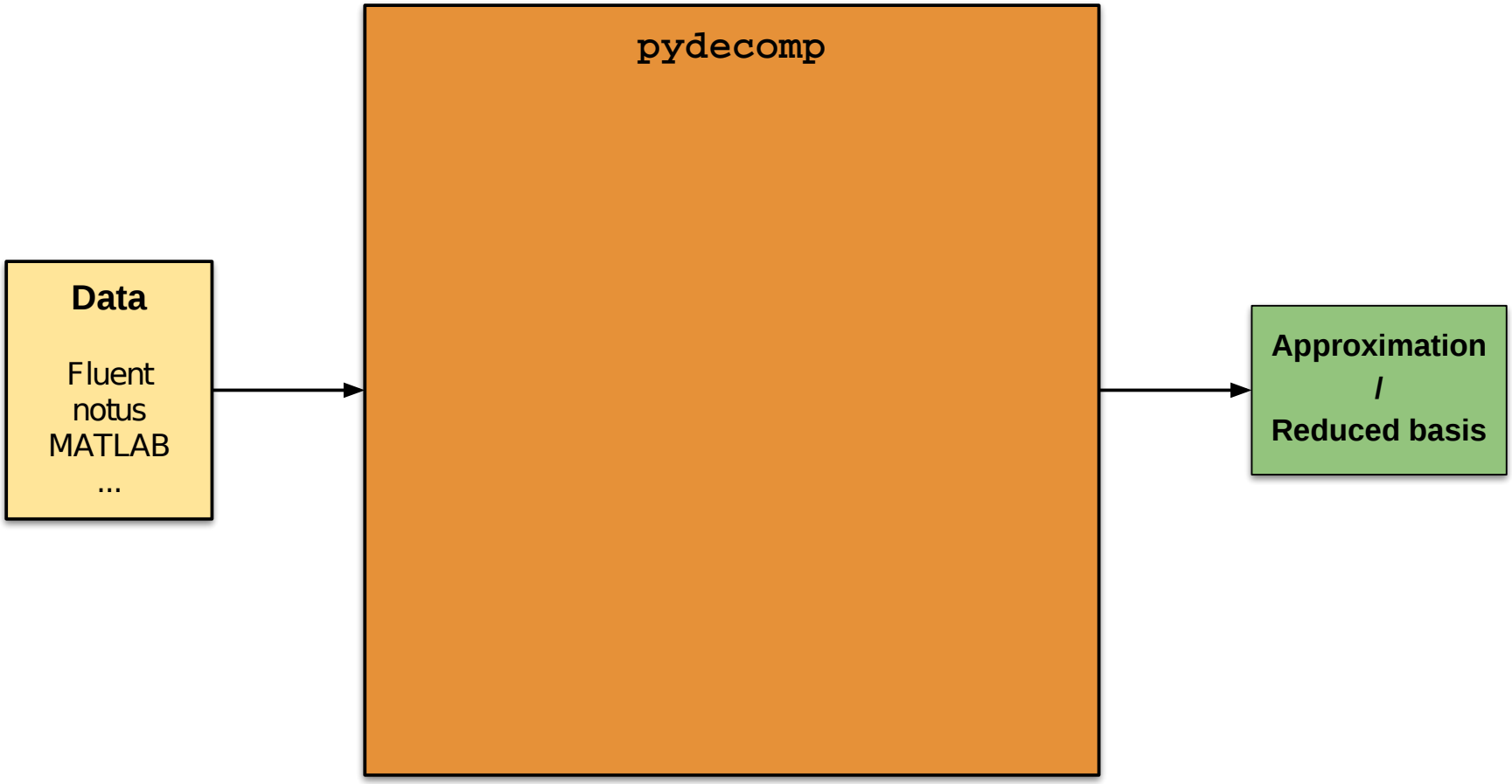
Data decomposition



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

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.

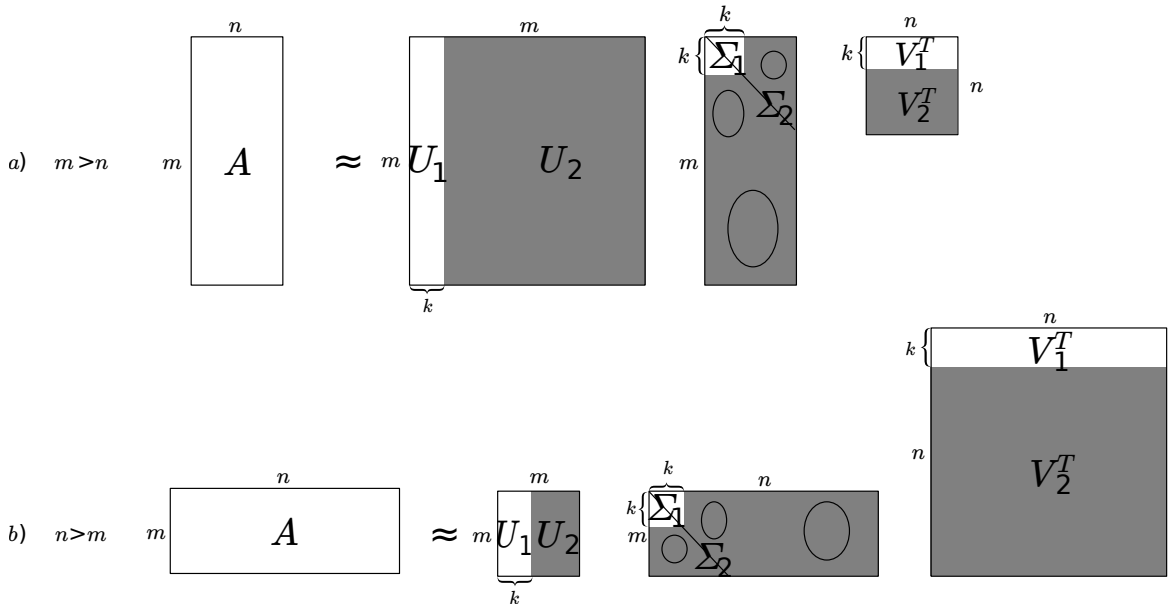
SVD, a General Matrix Decomposition

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.



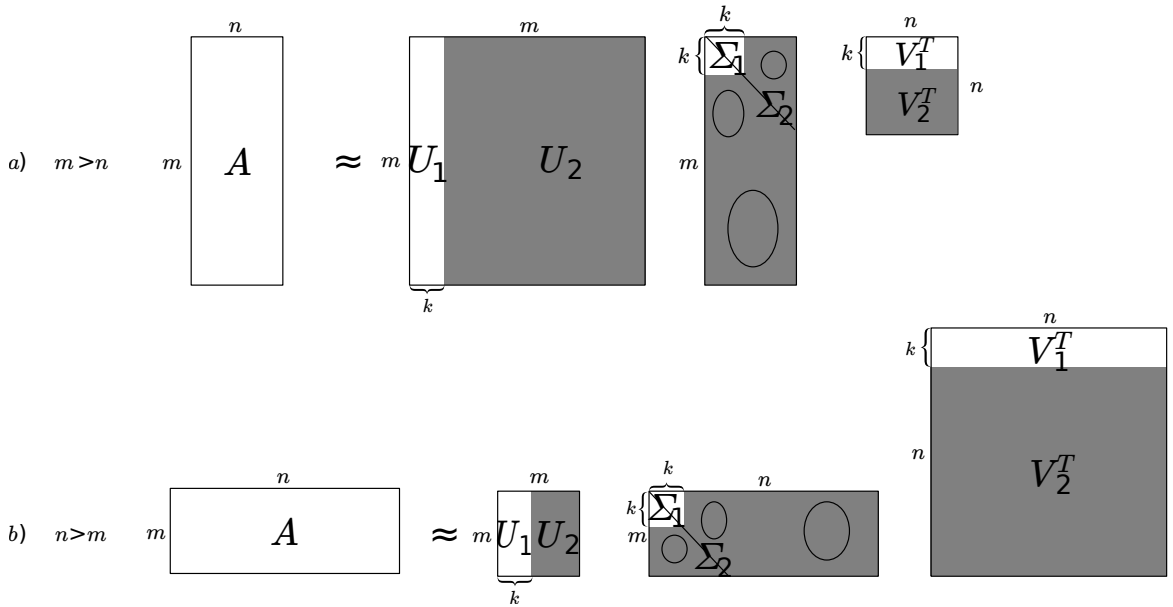
SVD, a General Matrix Decomposition

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

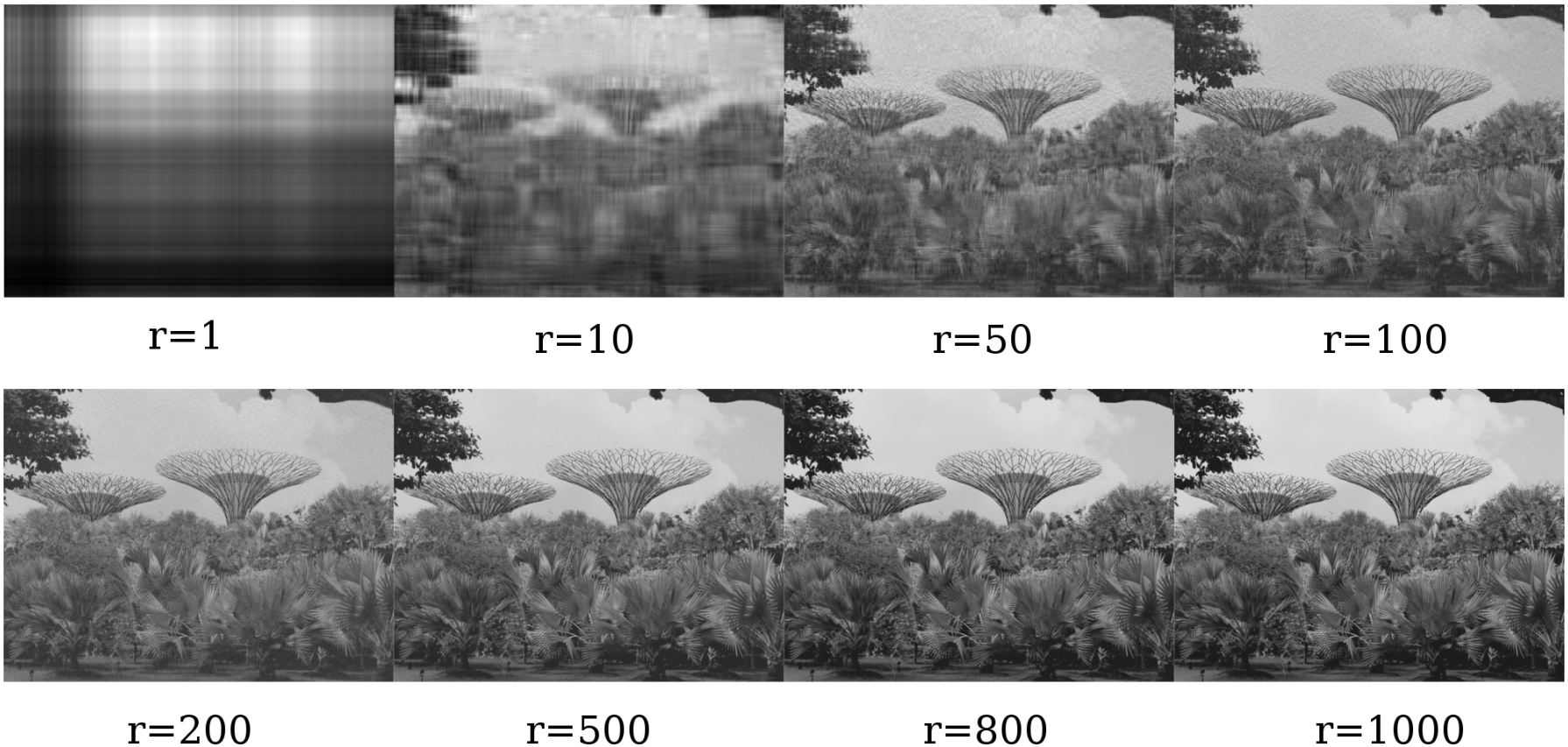


Figure: Singapore garden by the sea SVD reconstruction

jpg(90%), 1.5Mb

SVD (r=200), 1.37Mb

Proper Orthogonal Decomposition

POD

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

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

Proper Orthogonal Decomposition

POD

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

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

- 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

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

- 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 $\{\phi_k\}_{k=1}^{r}$

Proper Orthogonal Decomposition

POD

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

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

- 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 $\{\phi_k\}_{k=1}^{r}$
- Eckart-Young theorem

Proper Orthogonal Decomposition

POD

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

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

- 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 $\{\phi_k\}_{k=1}^{r}$
- 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

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

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

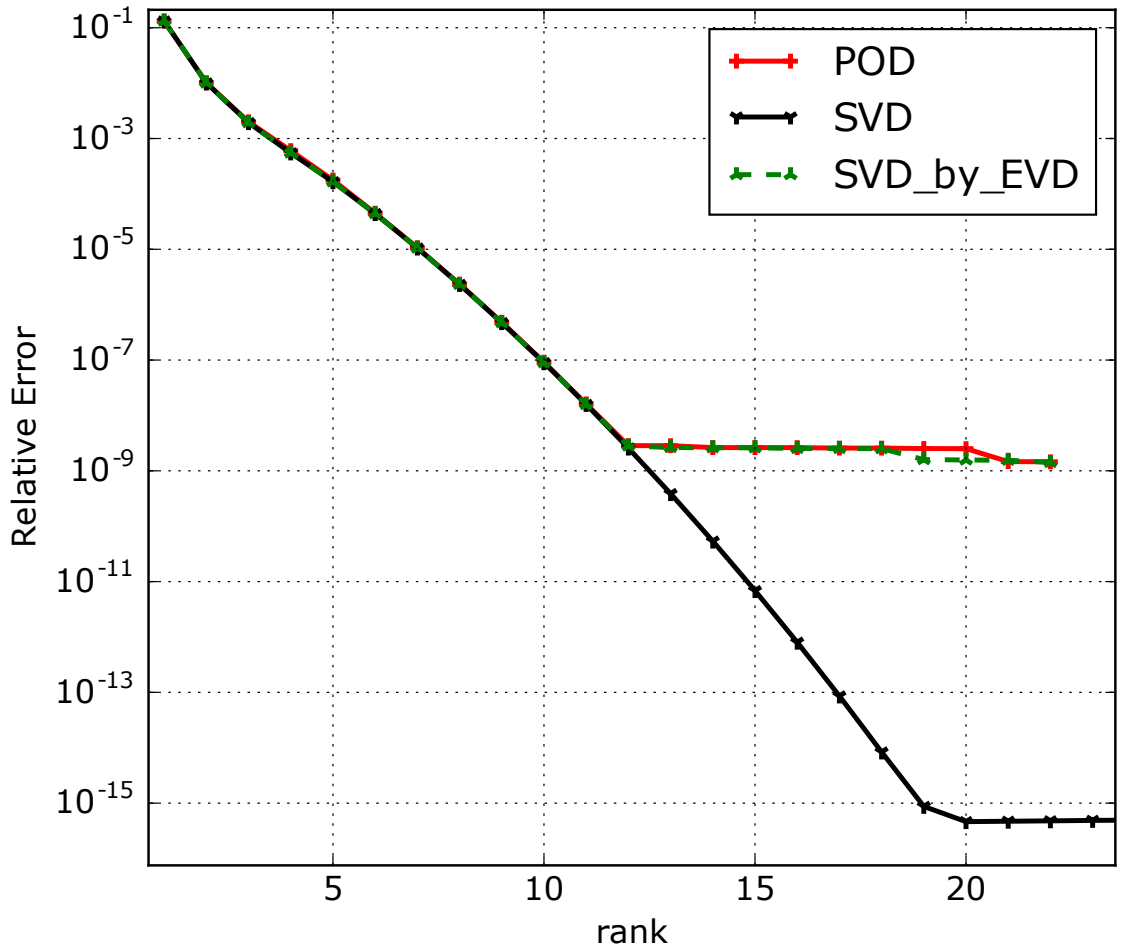
Properties

- Spectral theory : POD is solution of an eigenvalue problem
- POD bases are orthonormal $\{\phi_k\}_{k=1}^{r}$
- 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 defined on $\Omega = [0,1] \times [0,1]$ with $f(x,t) = \sin(\sqrt{x^2 + t^2}) \approx \widetilde{f} = \sum_{k=1}^{r} \phi_k(x) a_k(t)$

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



High Reynolds number Lid driven cavity

Joint

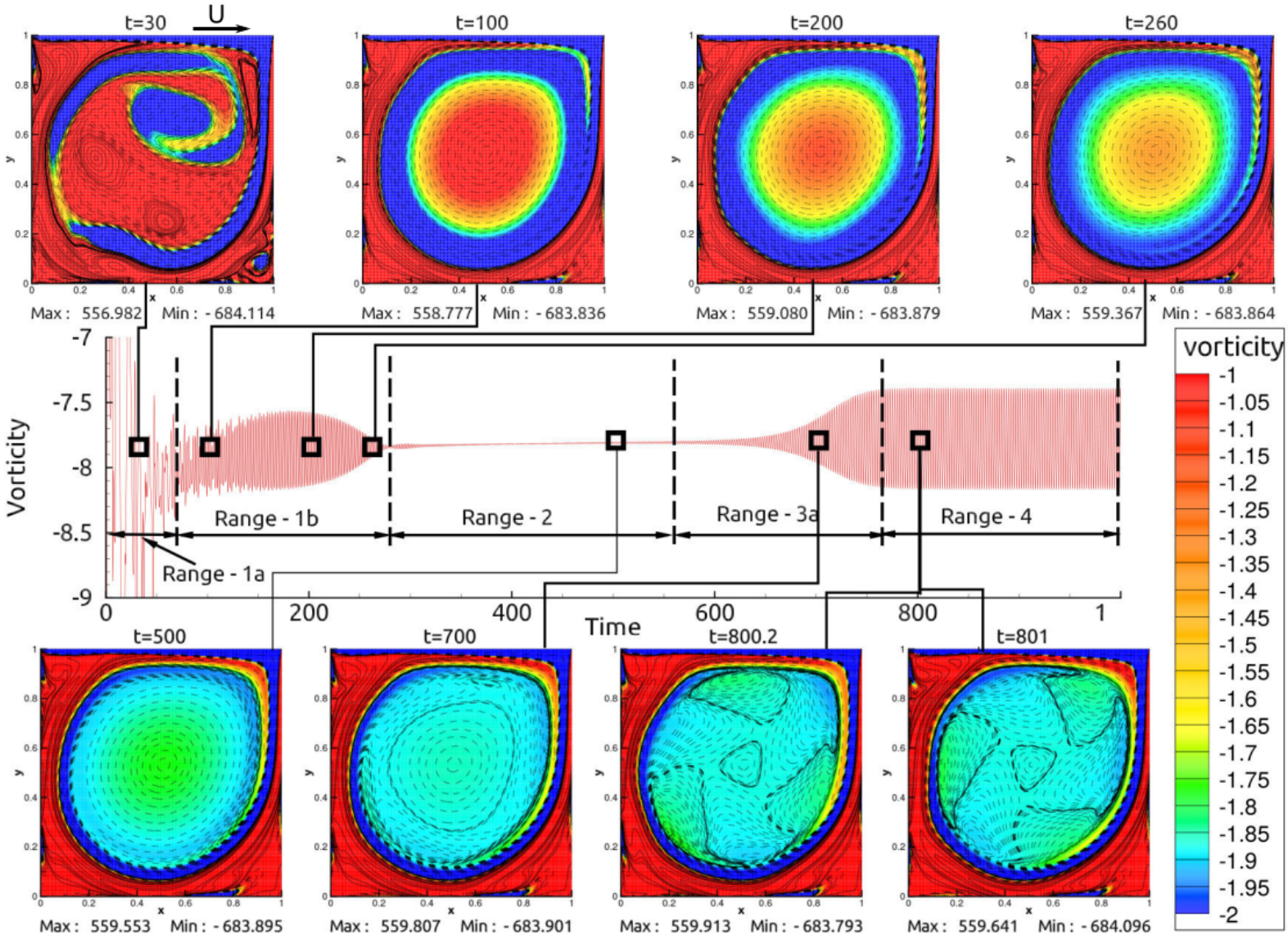


Figure: Vorticity time series at (0.95,0.95) for Re=8800

LDC space modes

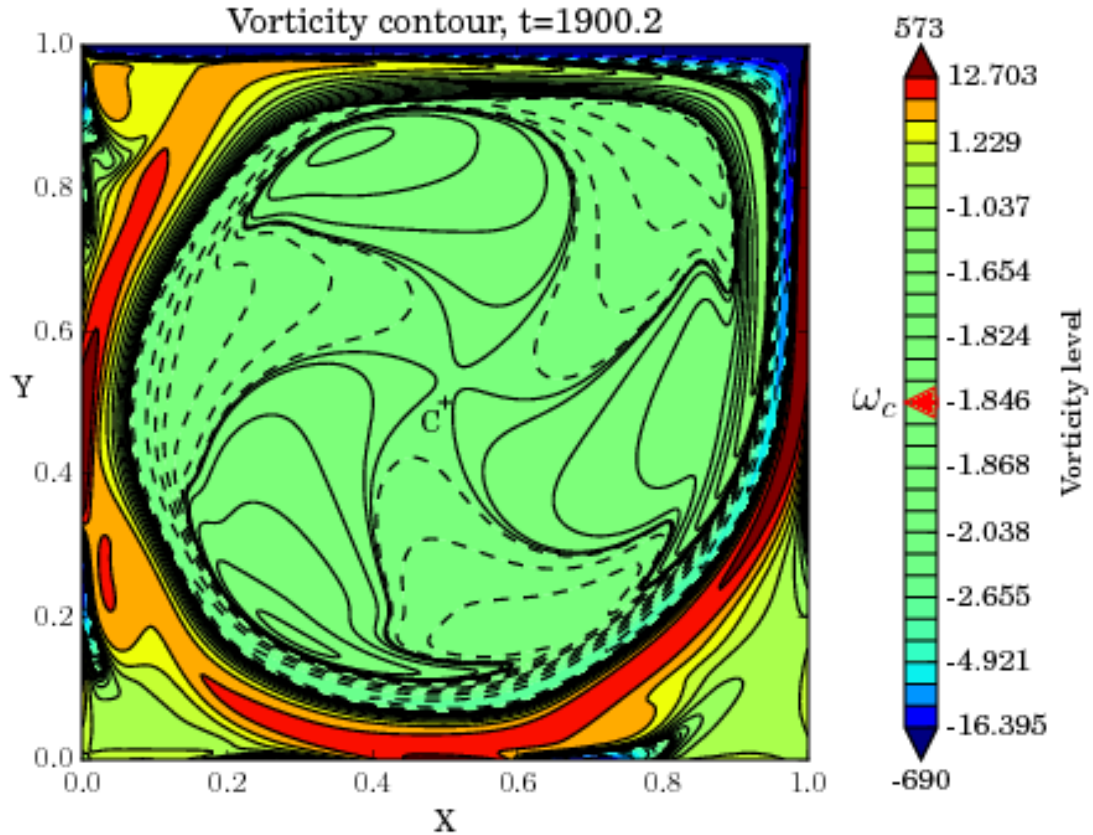


Figure: One snapshot of vorticity contours at t=1900,2 for Re=9800 in the limit cycle

LDC space modes

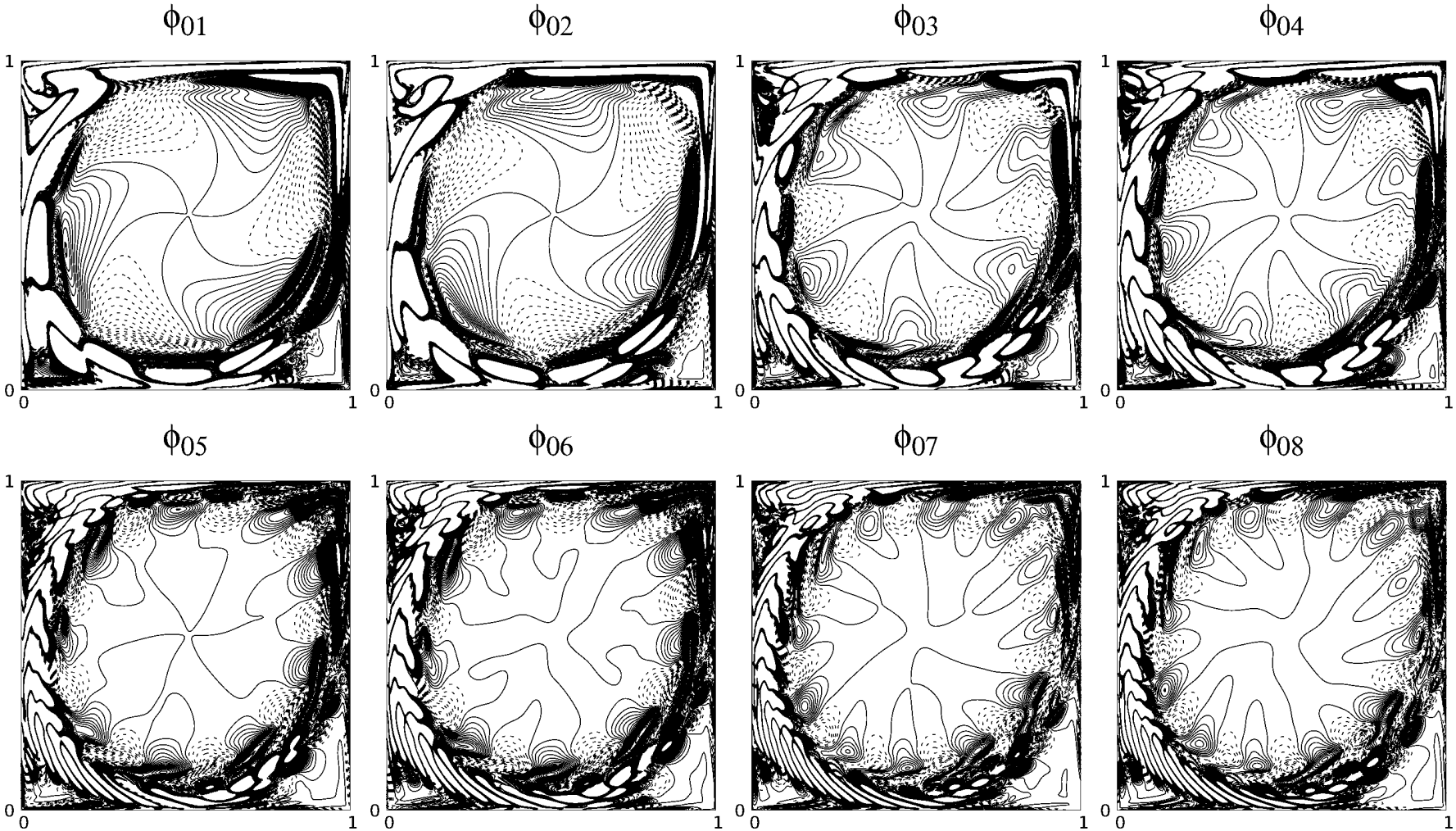


Figure: Re=9800, first 8 POD space modes

LDC time modes

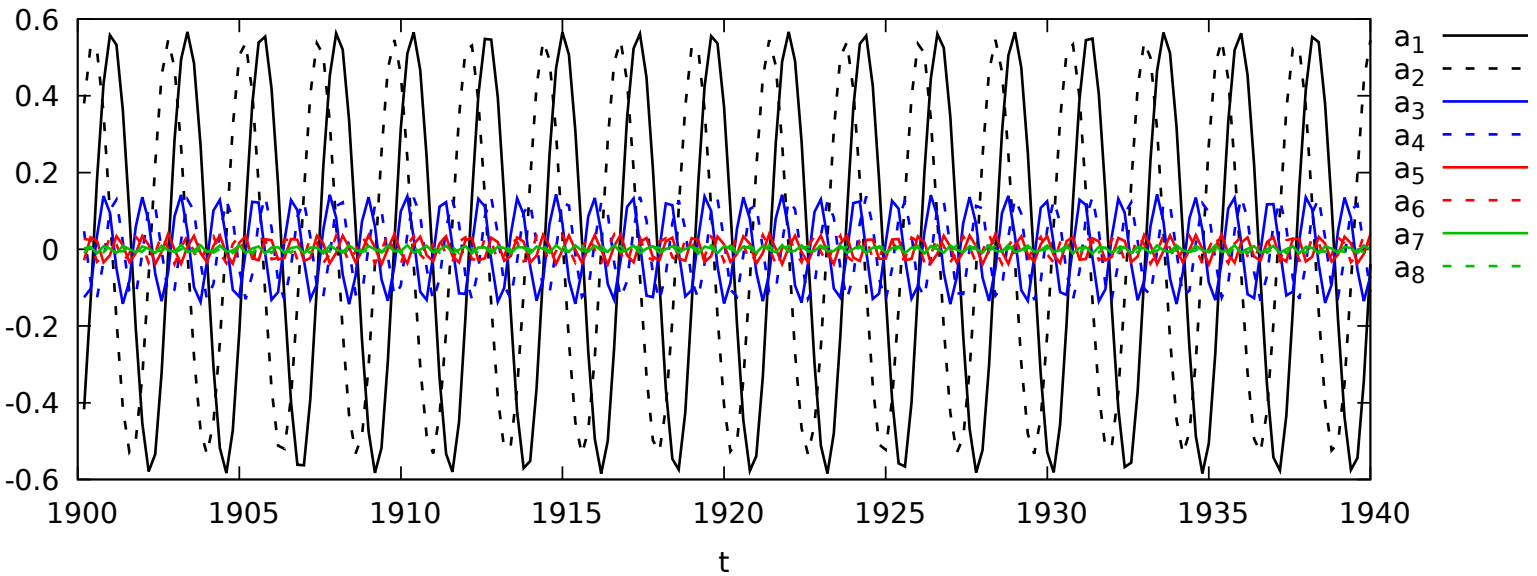
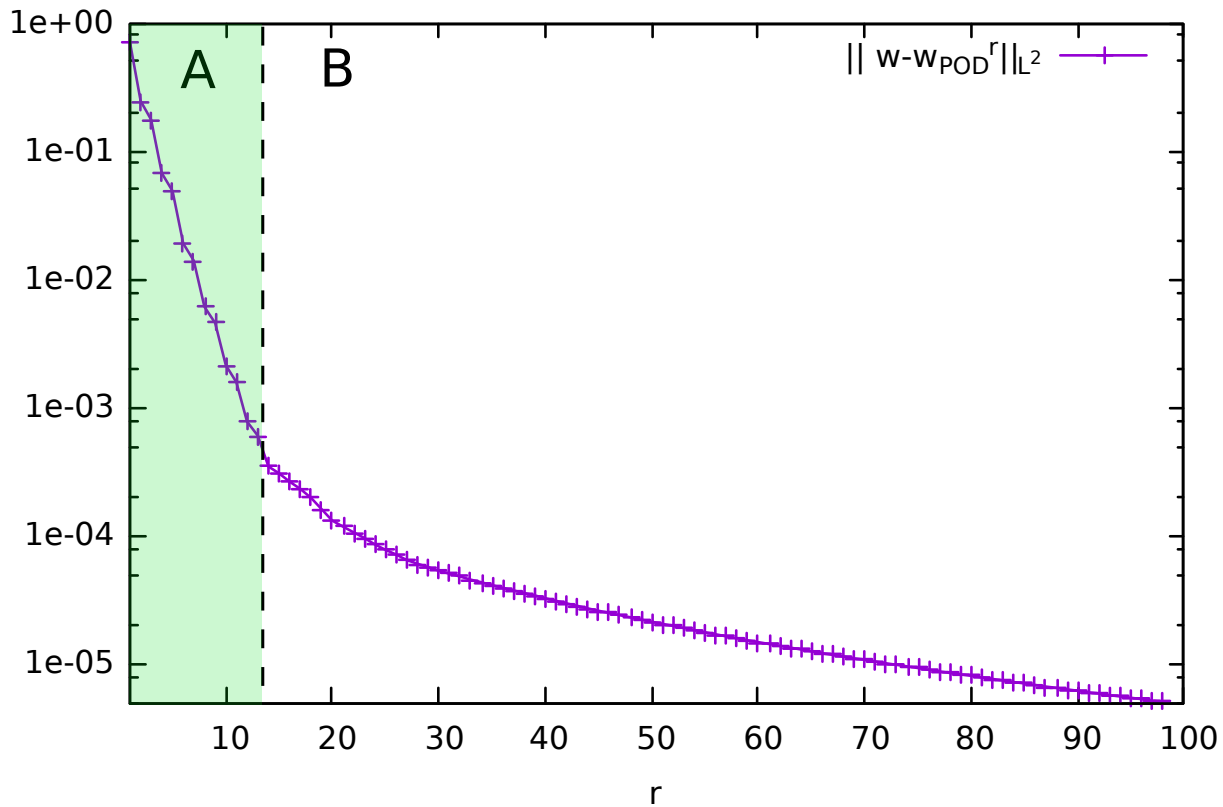


Figure: Re=9800, first 8 POD time modes

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

Data decomposition

- Bivariate decomposition
- Tensor formats and approximation
- Numerics

Reduced order models for complex flows

Research Project: Deep Reduced Order Model

Conclusion and perspectives

Tensors

A generalisation of matrices to higher dimensions. i.e. $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.

Tensors

A generalisation of matrices to higher dimensions. i.e. $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$.

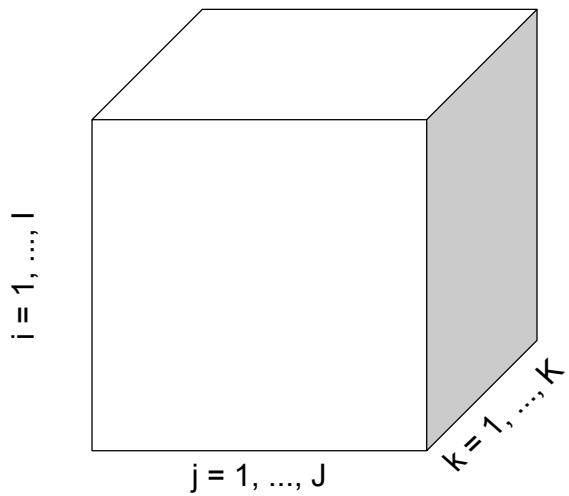


Figure: Order 3 tensor

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

$$T_{ij} = x_i y_j$$

- **Matricisation**: Ordering the elements of a tensor into a matrix **allows SVD**.

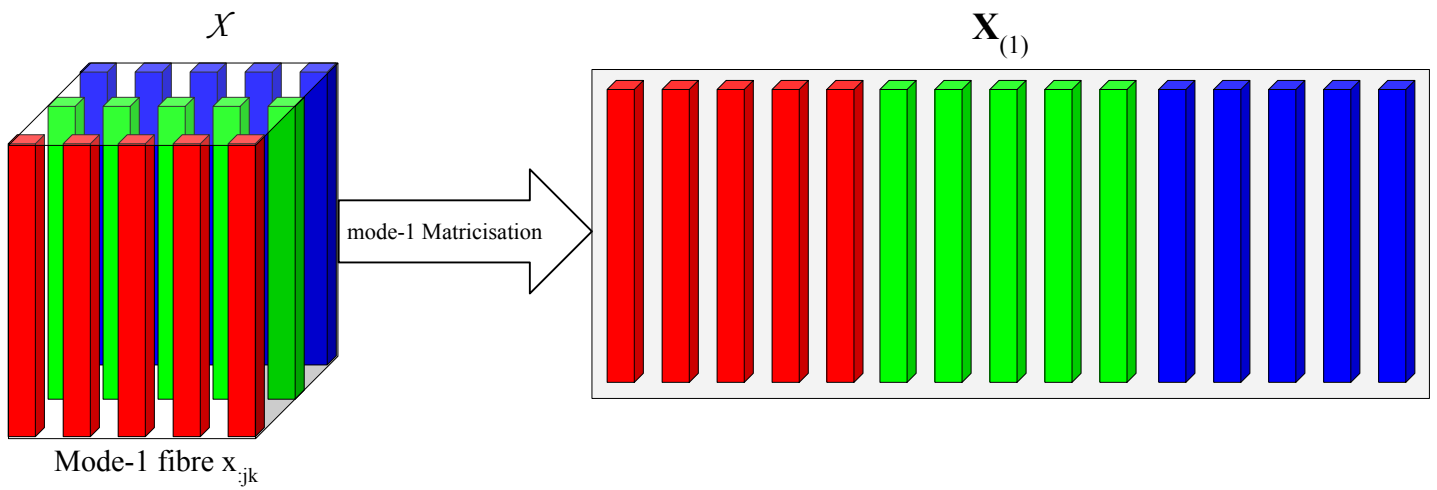


Figure: Mode one matricisation of third order tensor with $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$.

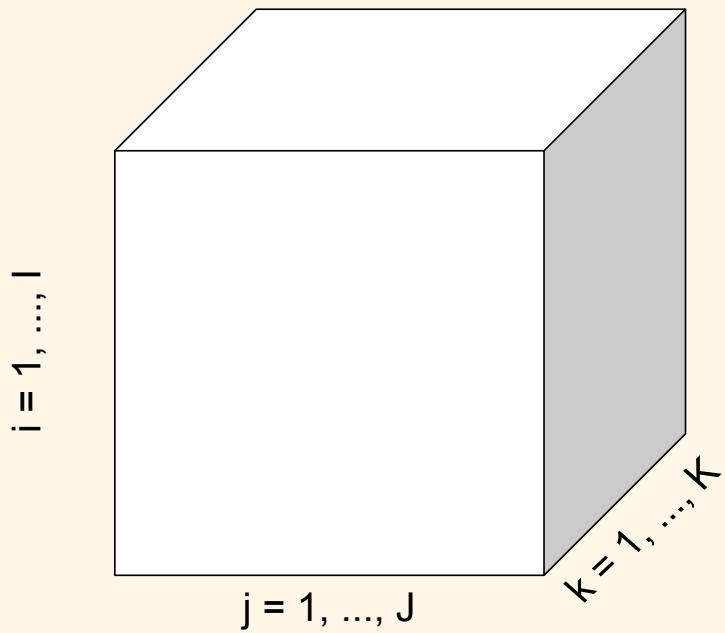
Full Tensor Format

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.

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

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

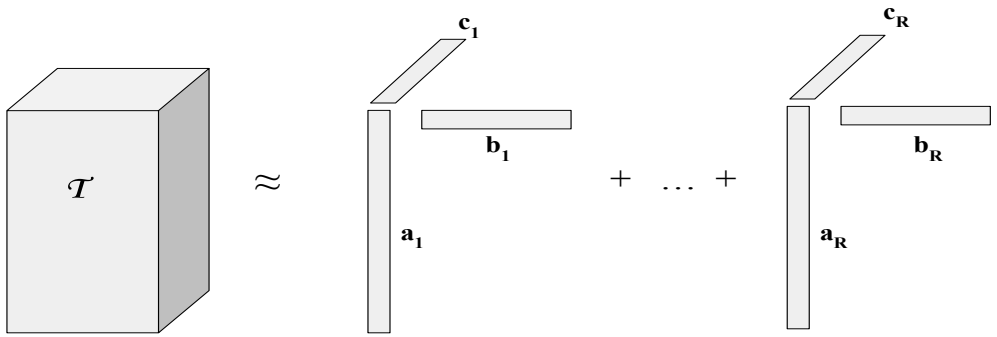


Figure: CP decomposition of third order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

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.

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

- Storage cost : $\mathcal{O}(drn)$
- **Tensor rank:** $rank(A)$ **is the minimum number of rank one tensor that generates A.**

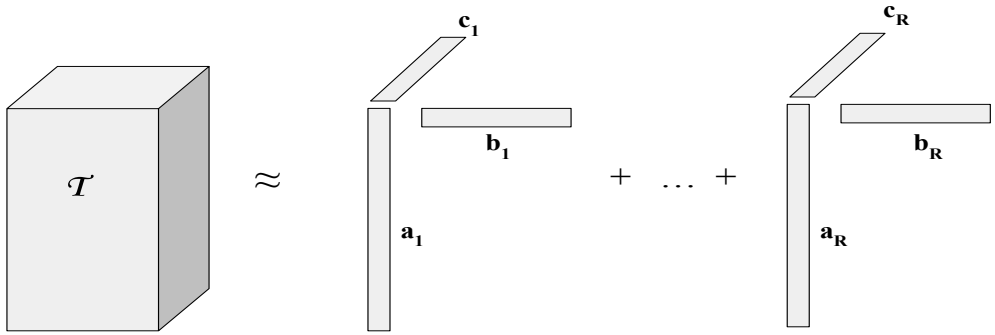


Figure: CP decomposition of third order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

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 : $\mathcal{O}(drn)$
- **Tensor rank**: $rank(A)$ is the minimum number of rank one tensor that generates A.
- **NP-hard problem to compute the rank of a tensor**

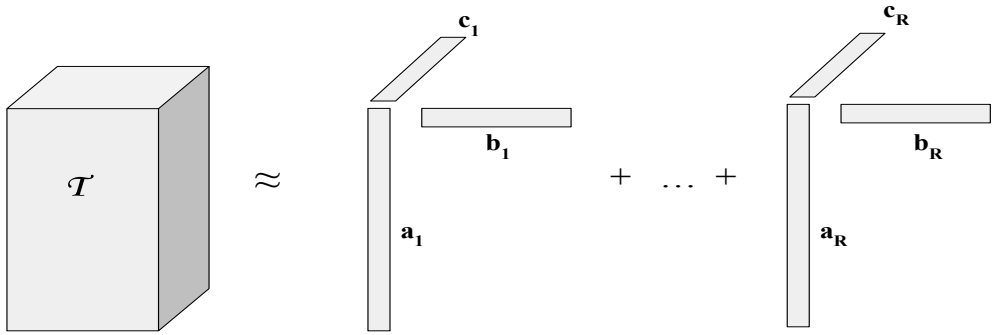


Figure: CP decomposition of third order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

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.

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

- Storage cost : $\mathcal{O}(drn)$
- **Tensor rank**: $rank(A)$ is the minimum number of rank one tensor that generates A.
- NP-hard problem to compute the rank of a tensor
- **Find the best canonical decomposition: ill-posed problem.**

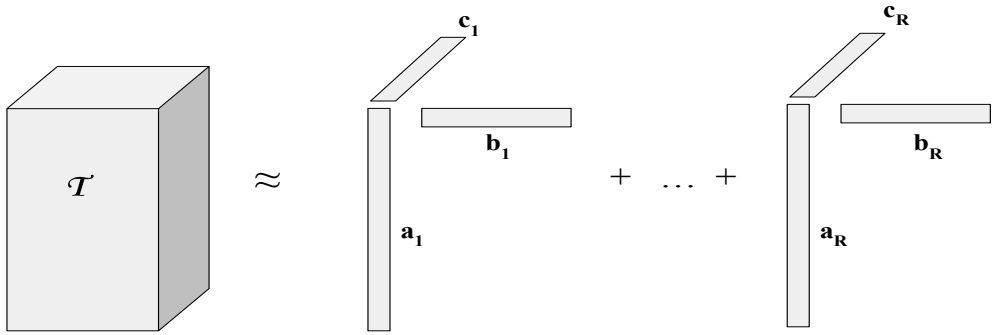


Figure: CP decomposition of third order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

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 : $\mathcal{O}(drn)$
- **Tensor rank**: $rank(A)$ is the minimum number of rank one tensor that generates A.
- NP-hard problem to compute the rank of a tensor
- Find the best canonical decomposition: ill-posed problem.
- **Discrete PGD $\approx$ CP decomposition**

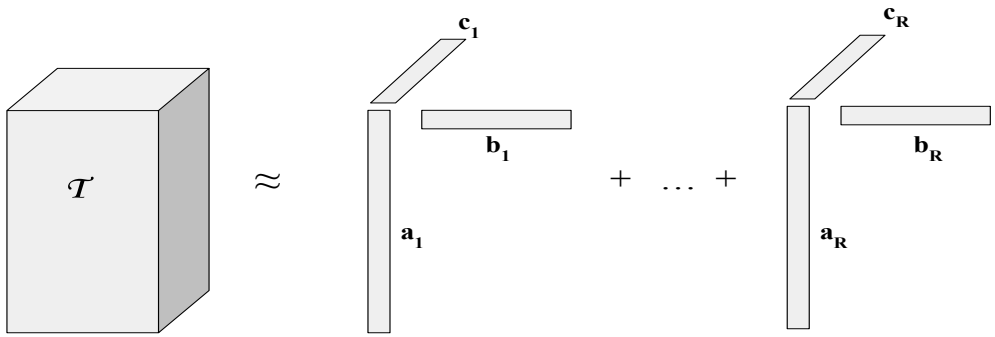


Figure: CP decomposition of third order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$.

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 $\mathcal{T}$.

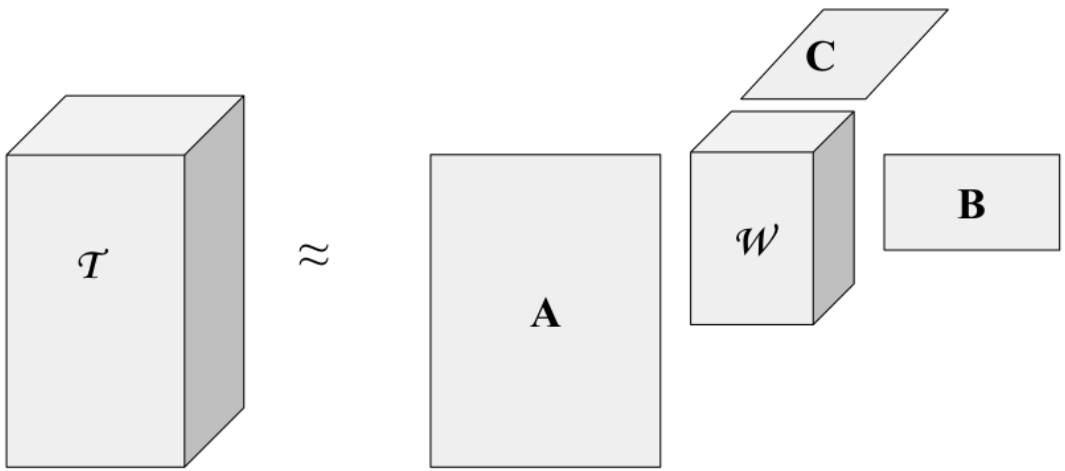
Tucker Tensor Format

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ 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.
Algorithms: ST-HOSVD, T-HOSVD with almost best approx.
Heavily relies on matricization.

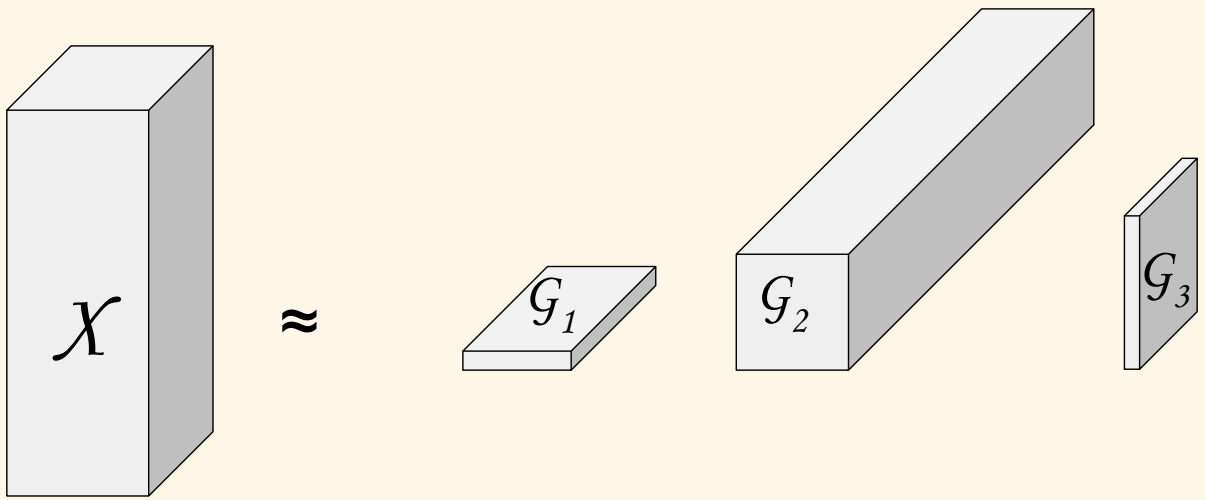


Tensor train

$\mathcal{T} \in \mathbb{R}^{n_1 \times \cdots n_d}$ 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) : $\mathcal{O}(k^2 dn)$



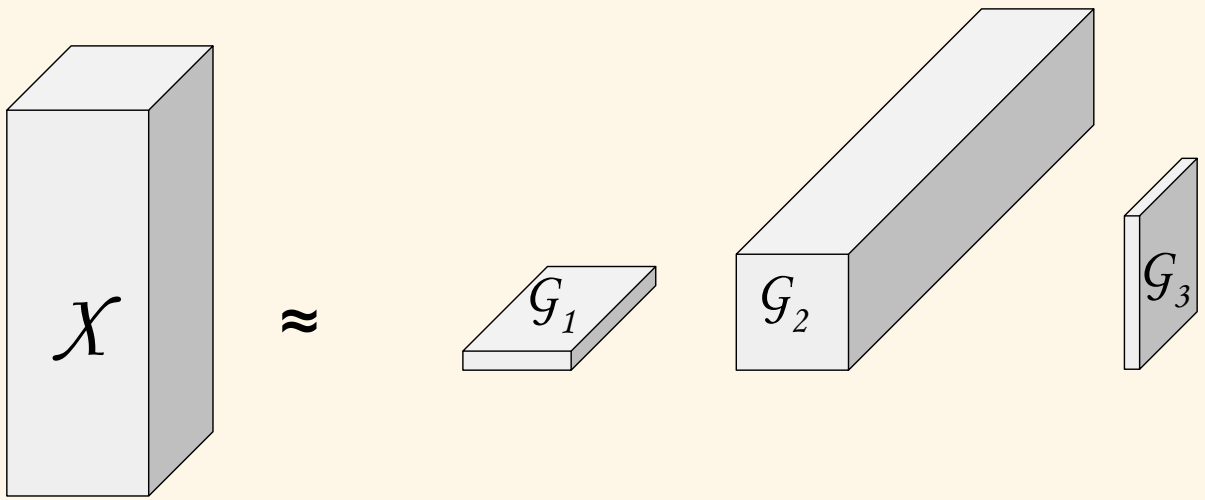
Goal Get rid of the core tensor.

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.

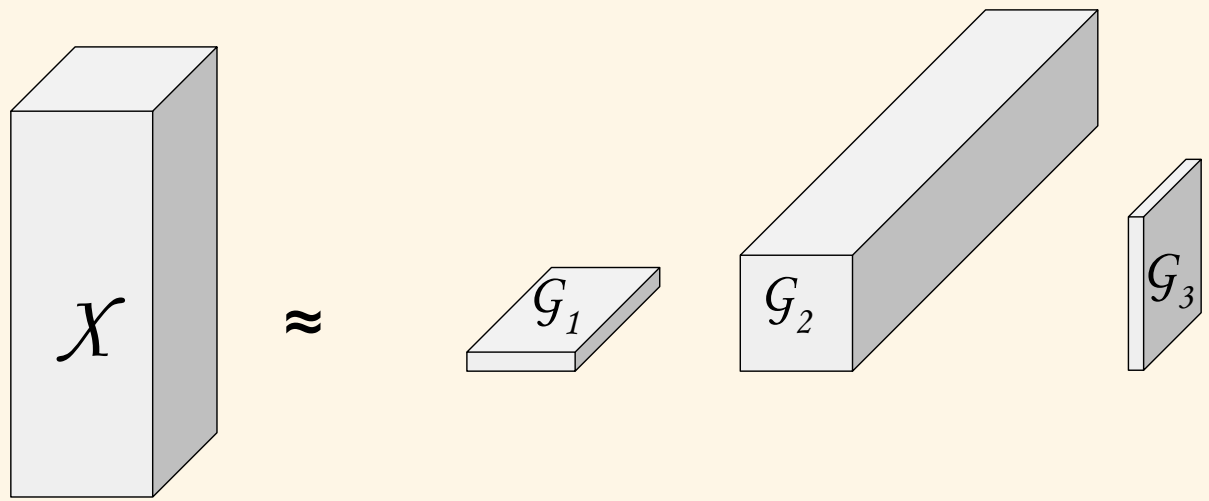
Idea **Encode the modes and their relations into matrix products (MPS) for each element.**

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.

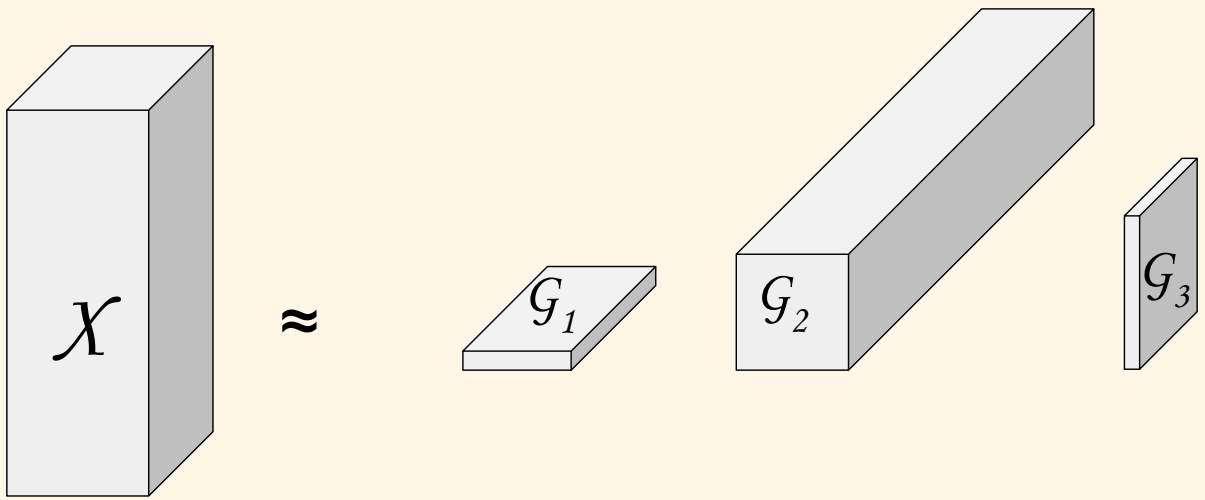
Pros **Efficient for high d. Easy implementation. Continuous version**

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



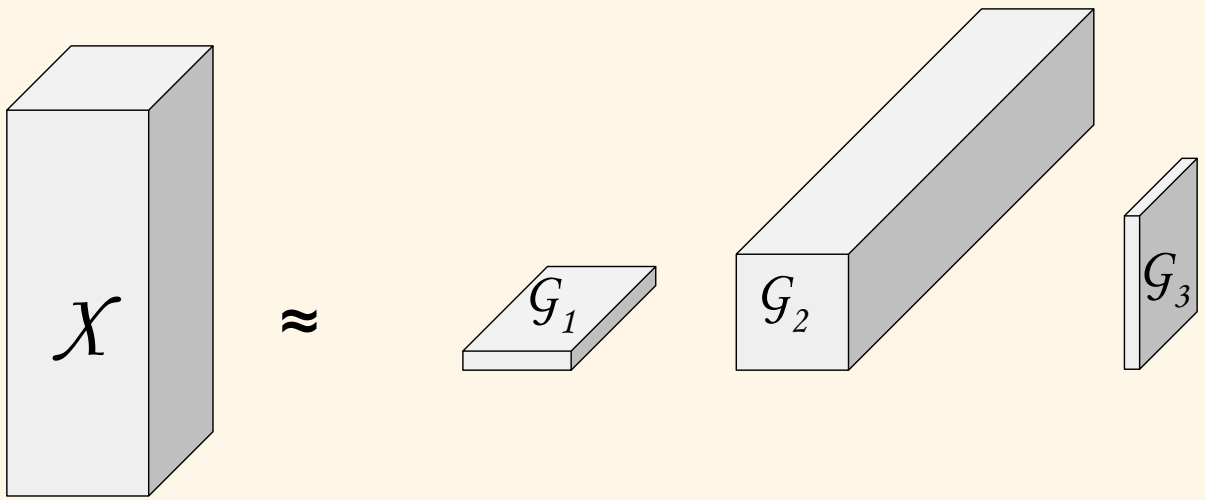
Goal Get rid of the core tensor.
Idea Encode the modes and their relations into matrix products (MPS) for each element.
Pros Efficient for high d. Easy implementation. Continuous version
Cons **Basis is not orthonormal.**

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

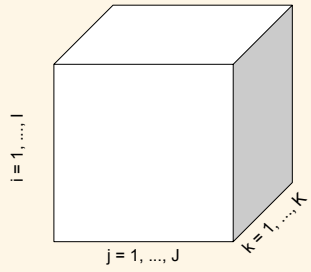


Goal Get rid of the core tensor.
Idea Encode the modes and their relations into matrix products (MPS) for each element.
Pros Efficient for high d. Easy implementation. Continuous version
Cons Basis is not orthonormal.
Algo **TT-SVD, same idea as ST-HOSVD but different format.**

Tensor decomposition recap

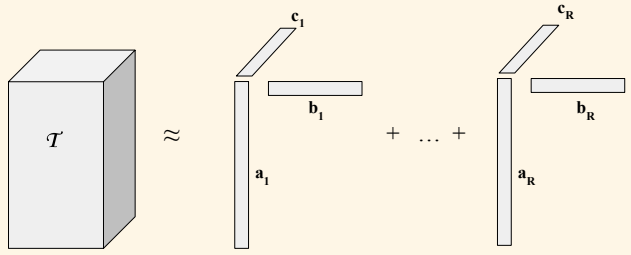
Full

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



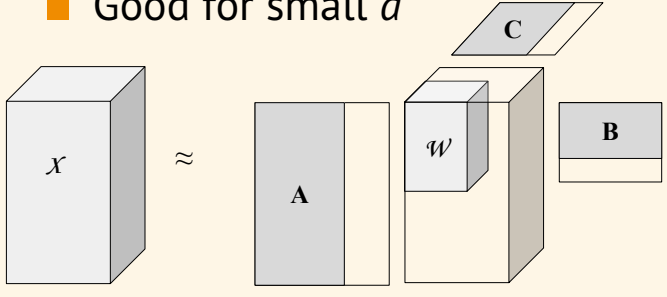
Canonical

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



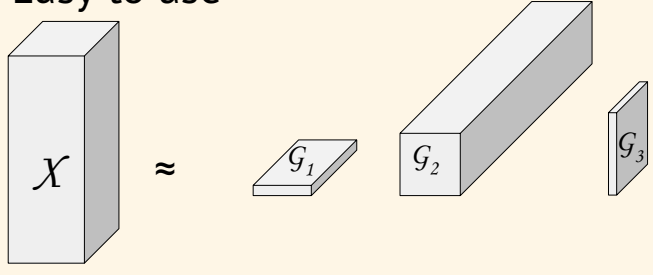
Tucker

- 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



Lestandi 2018

Azaïez, Lestandi, and Rebollo 2019

Data decomposition

- Bivariate decomposition
- Tensor formats and approximation
- Numerics

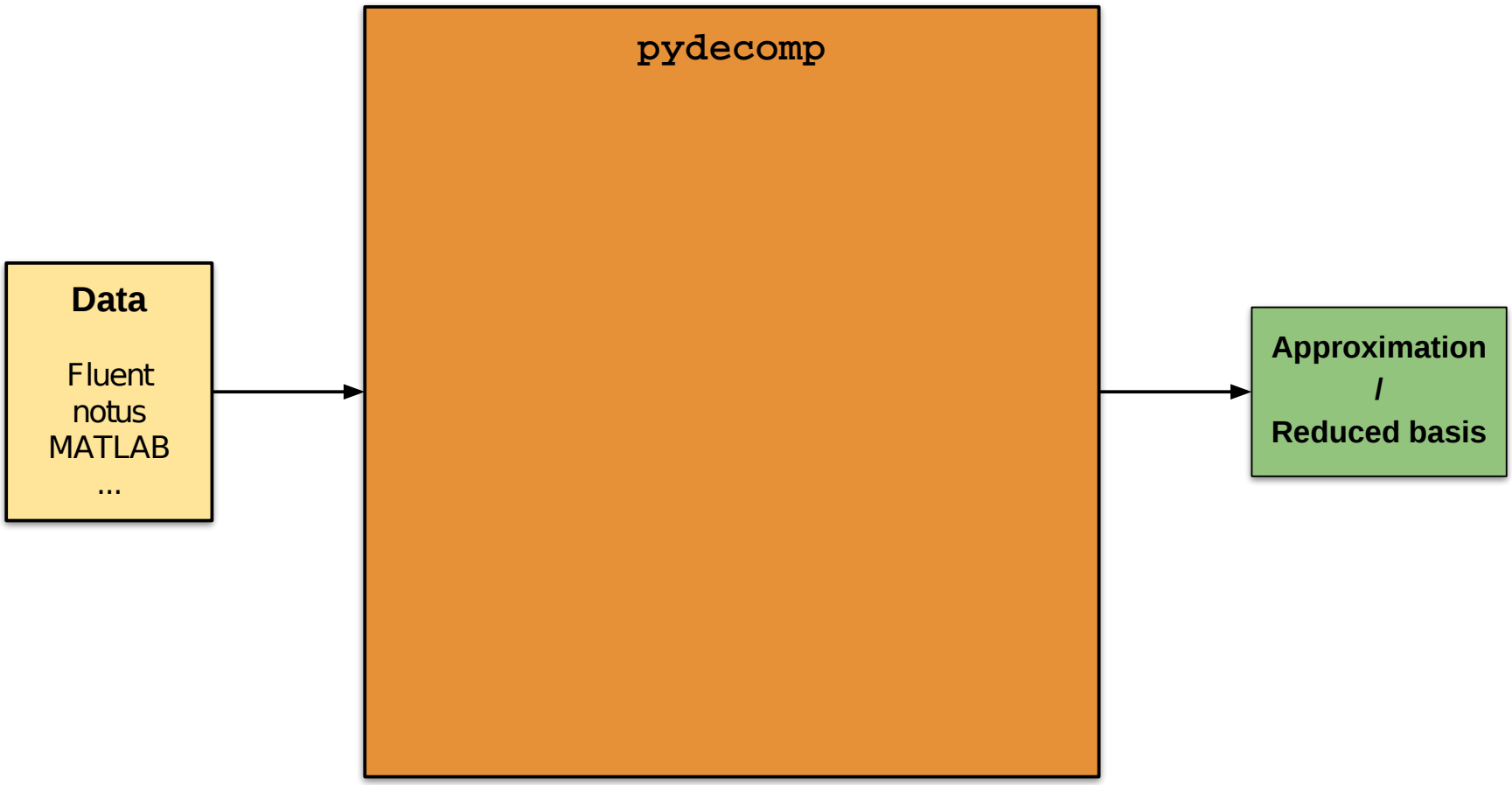
Reduced order models for complex flows

Research Project: Deep Reduced Order Model

Conclusion and perspectives

pydecomp: a data decomposition library

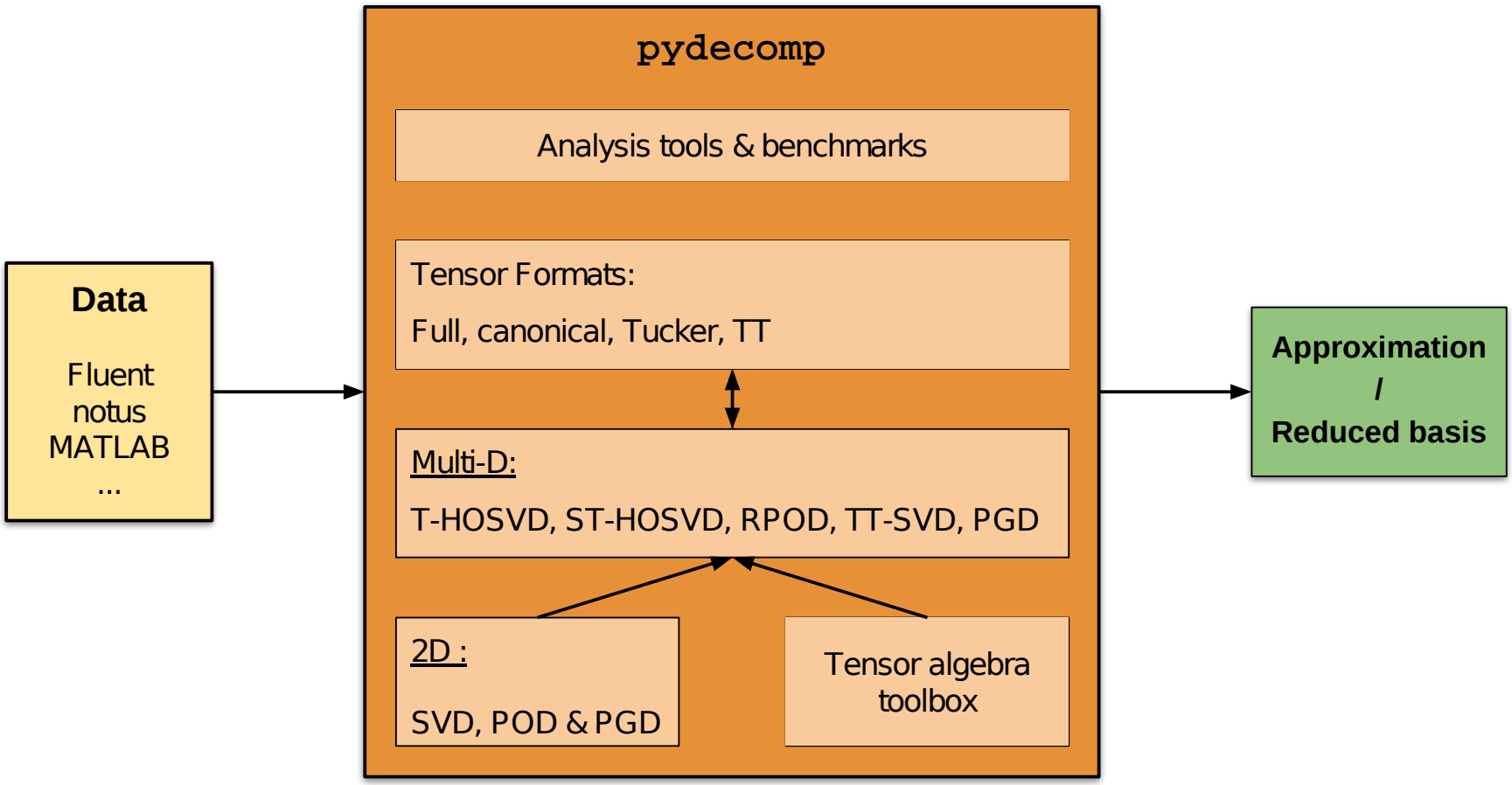
`https://git.notus-cfd.org/llestandi/python_decomposition_library`



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

pydecomp: a data decomposition library

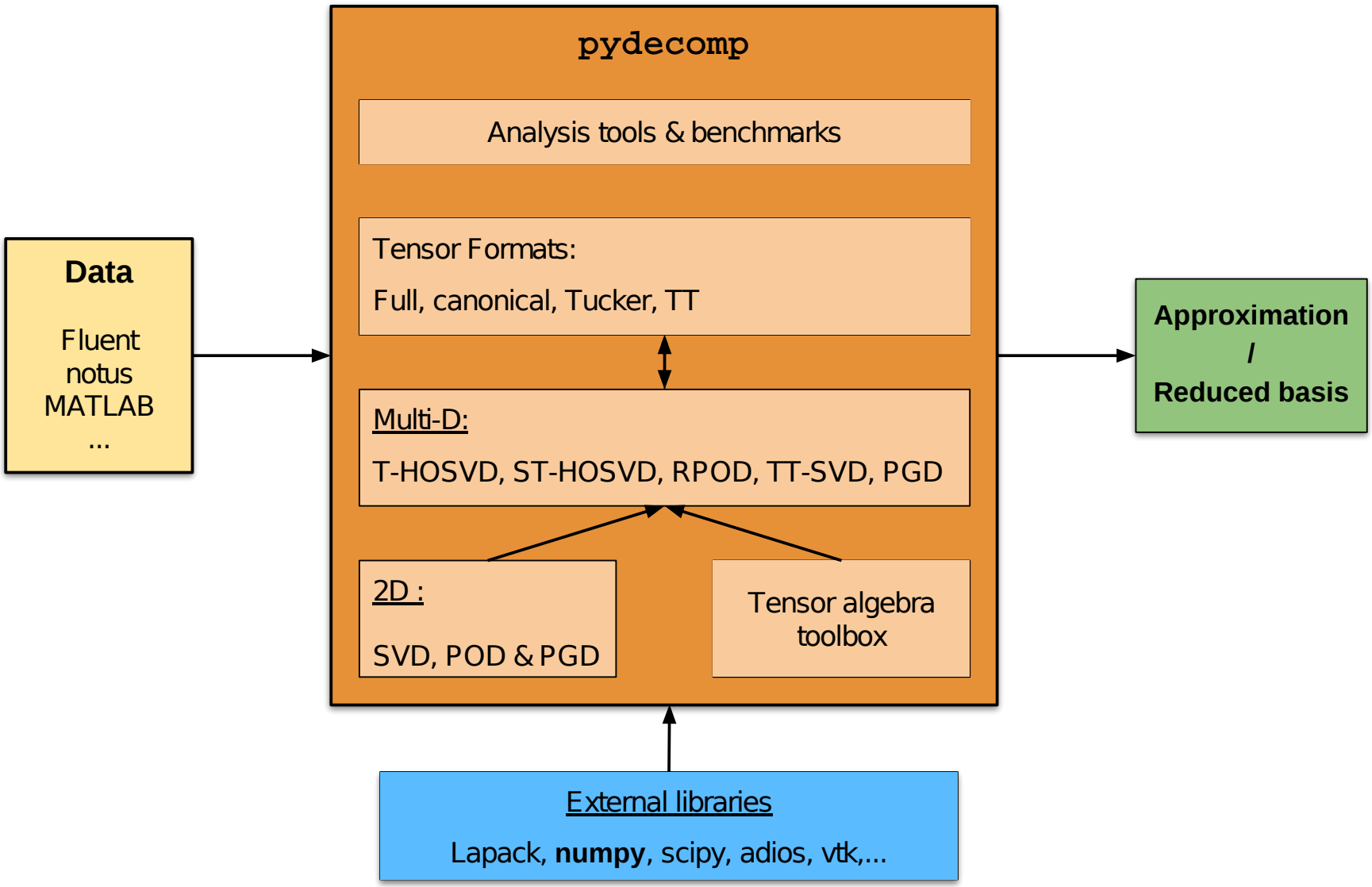
`https://git.notus-cfd.org/llestandi/python_decomposition_library`



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

pydecomp: a data decomposition library

`https://git.notus-cfd.org/llestandi/python_decomposition_library`



Synthetic data 3D

$$\mathcal{E} = \frac{||\mathcal{T}_{\text{exact}} - \mathcal{T}_{\text{approx}}||}{||\mathcal{T}_{\text{exact}}||}$$

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

Synthetic data 3D

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

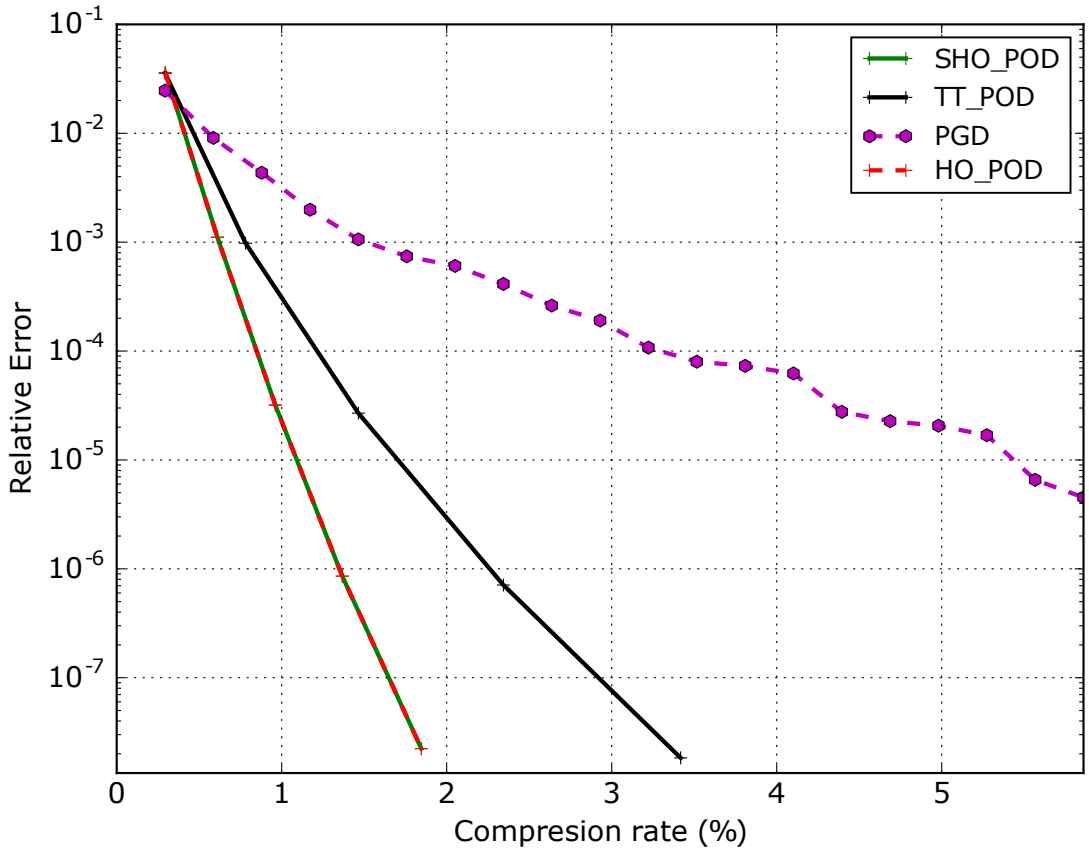


Figure: Approximation error of $f(x, y, z) = \frac{1}{1+x+y+z}$ with 32 grid points per dimension.

Higher dimensions

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

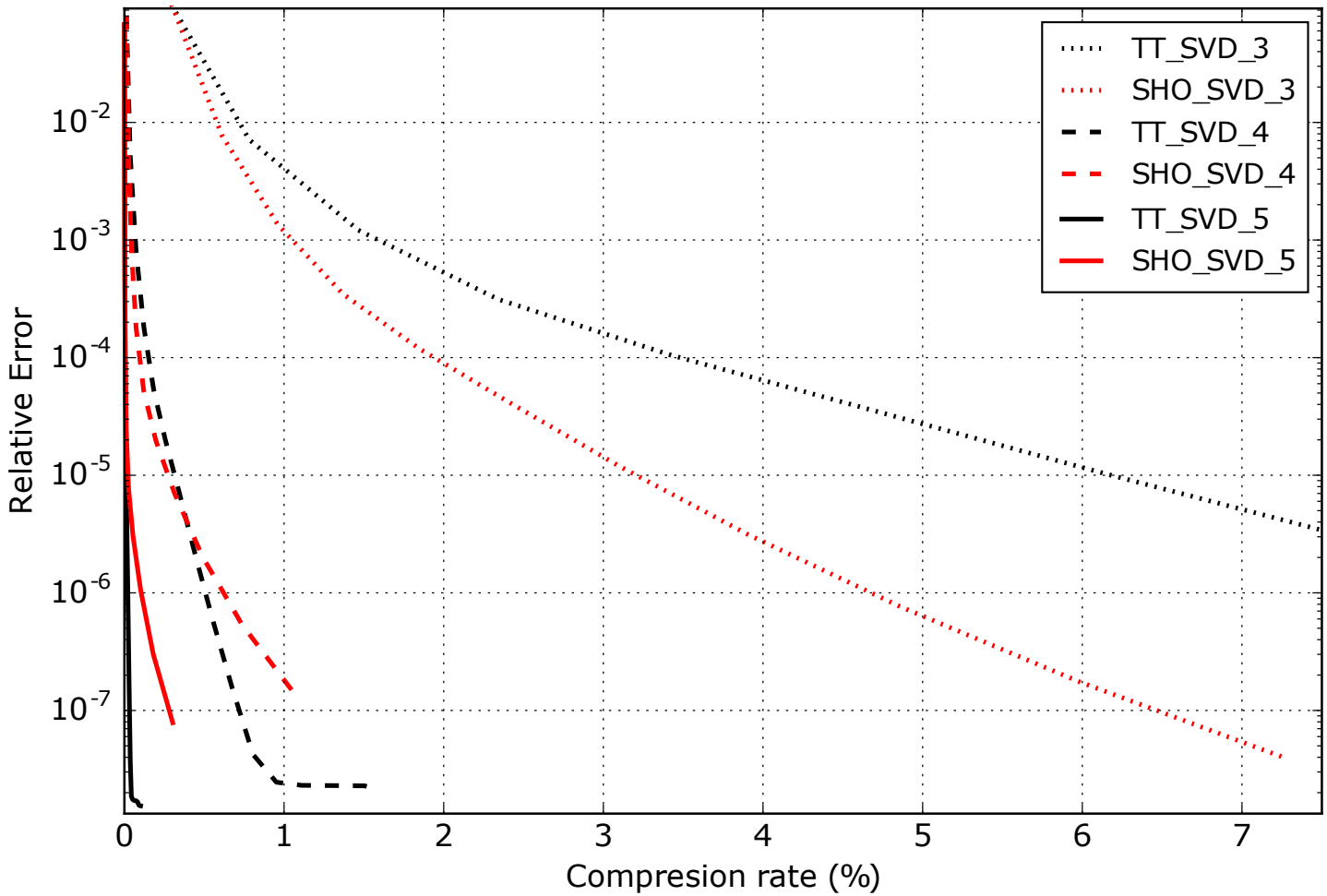


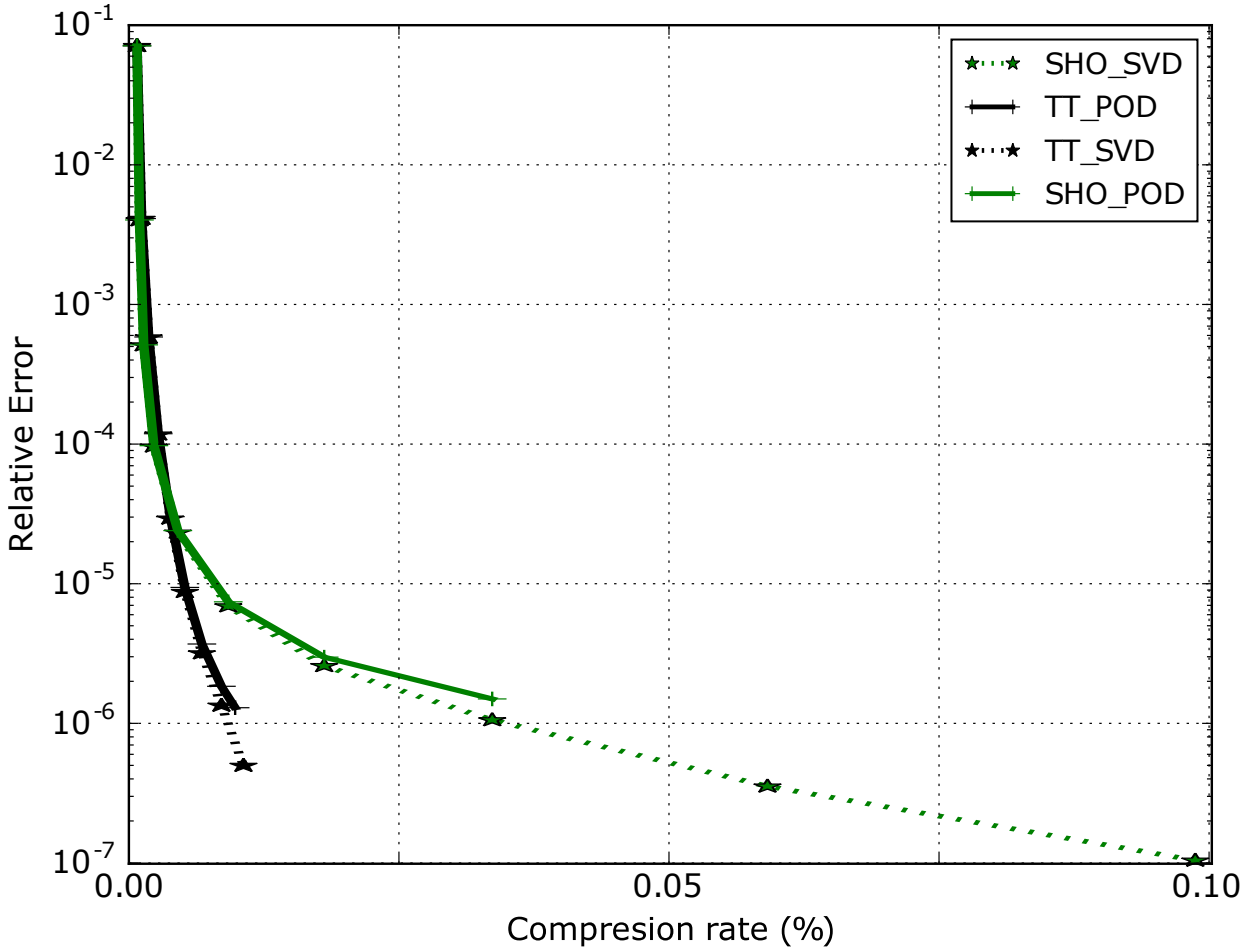
Figure: f_2 decomposition with $d = 3$ to 5 on a 32^d grid with 3 methods, L^2 scalar product.

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.
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: Droplet evaporation

Lestandi 2020

Input data

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

Experimental data: Droplet evaporation

Lestandi 2020

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.

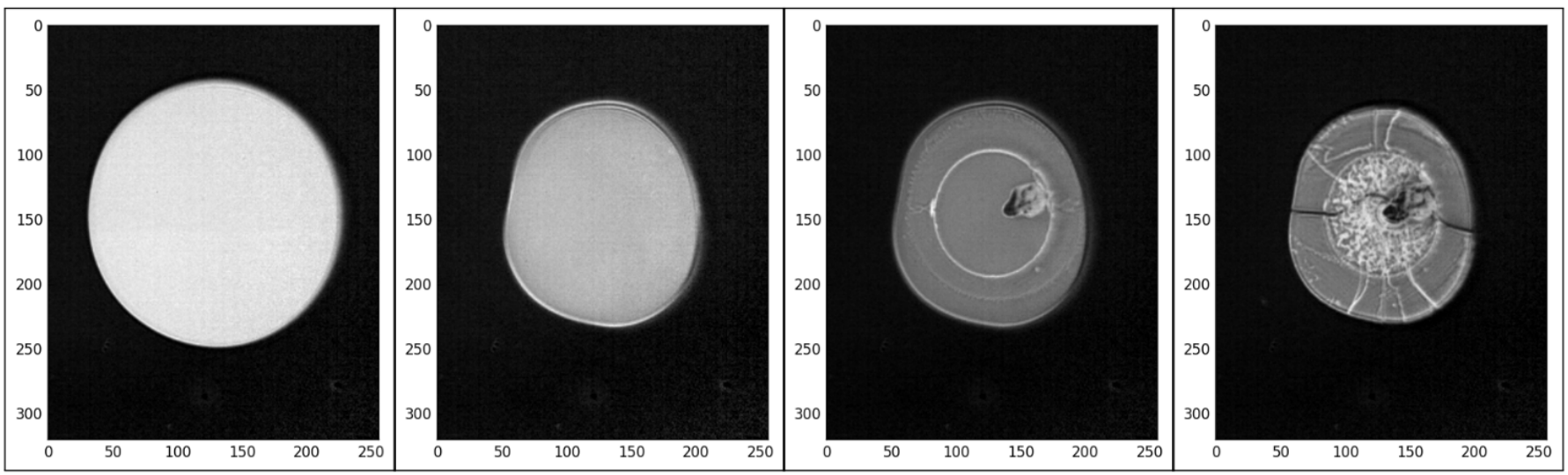
Experimental data: Droplet evaporation

Lestandi 2020

Input data

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

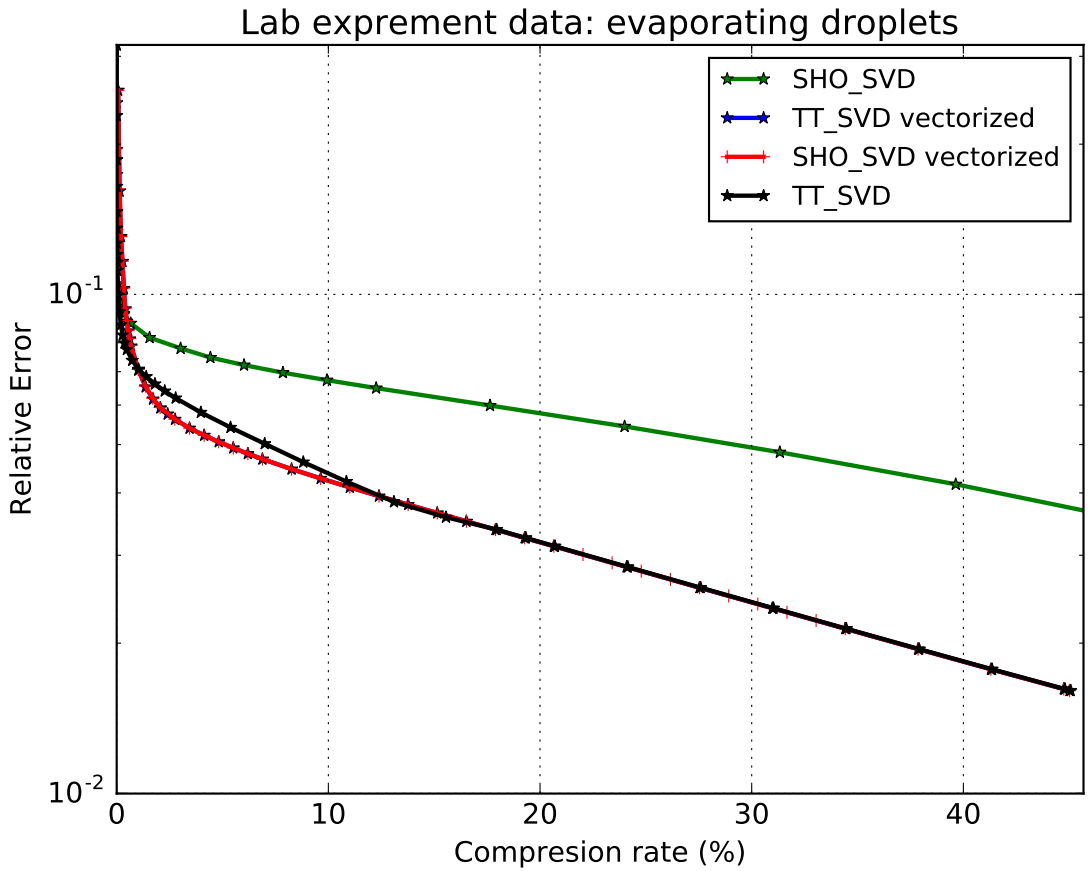
$\Longrightarrow$ Tensor of size $29 \times 51 \times 320 \times 356$ in a 800MB MATLAB file.



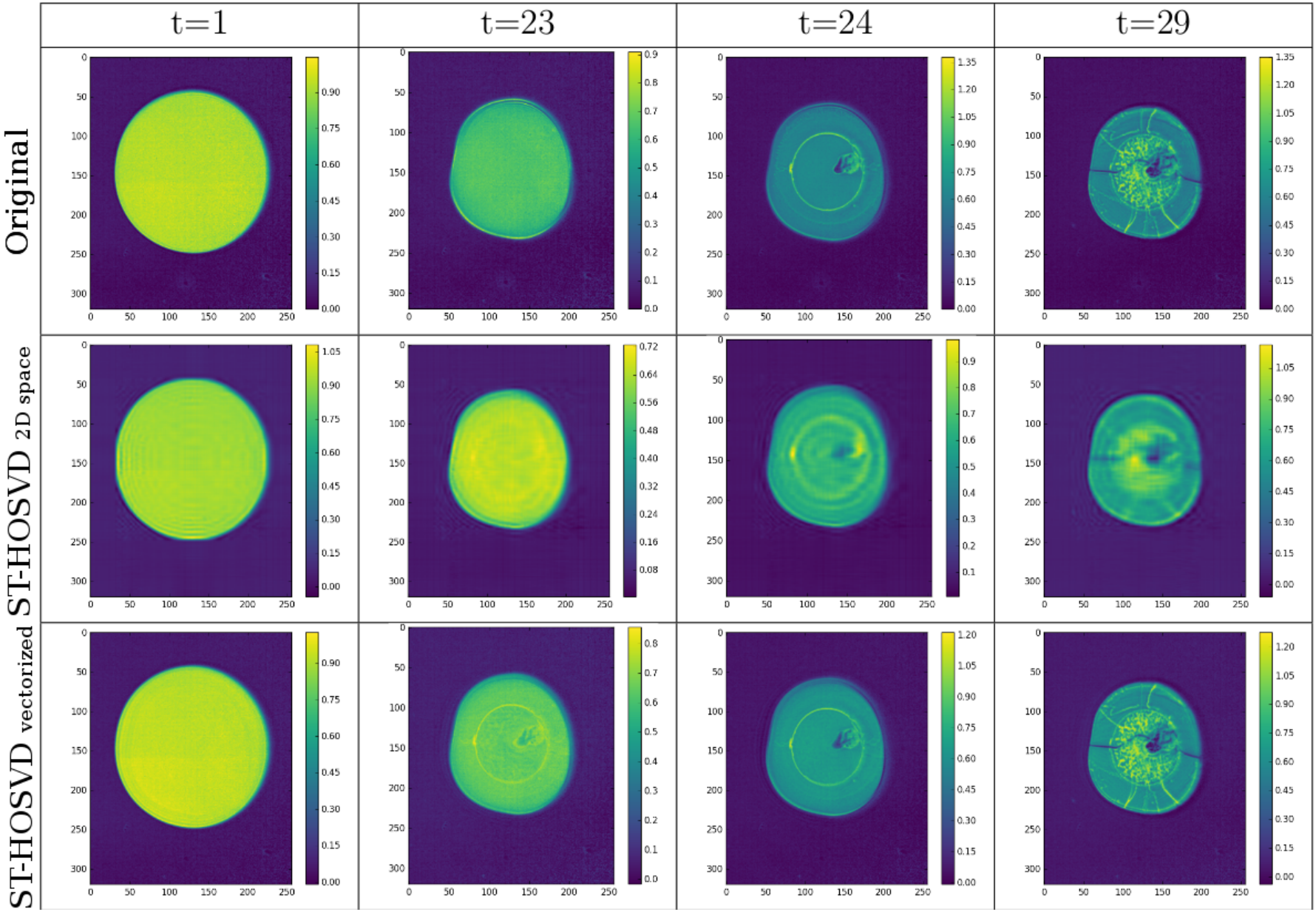
(a) t= 1/29 (b) t= 23/29 (c) t= 24/29 (d) t= 29/29

Figure: Evaporating droplets. Data provided by C. Pradère

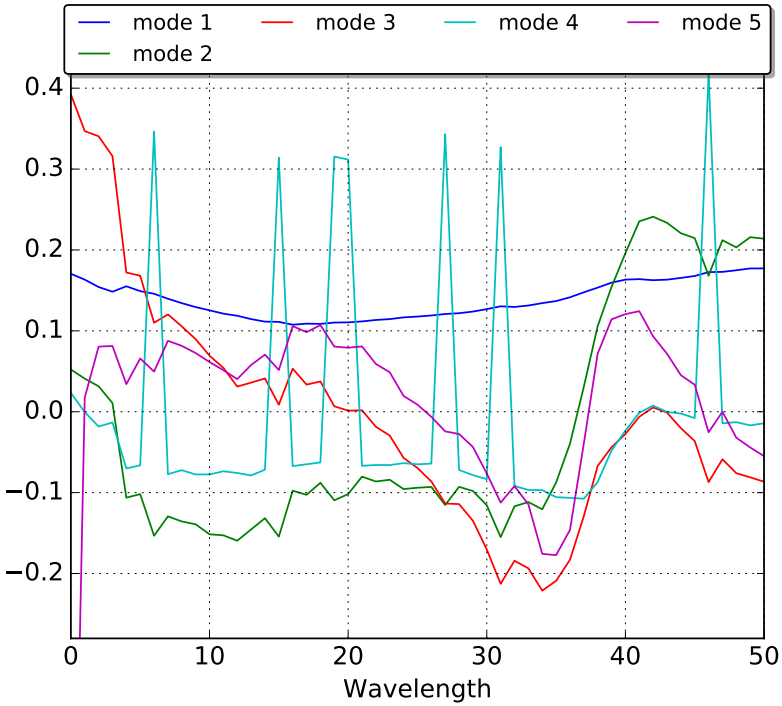
Droplet evaporation data approximation

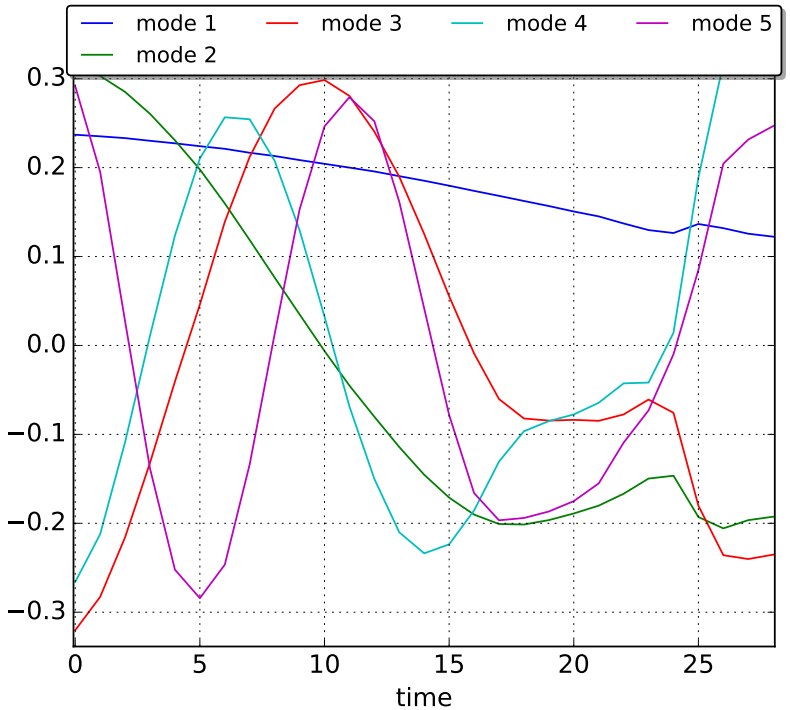


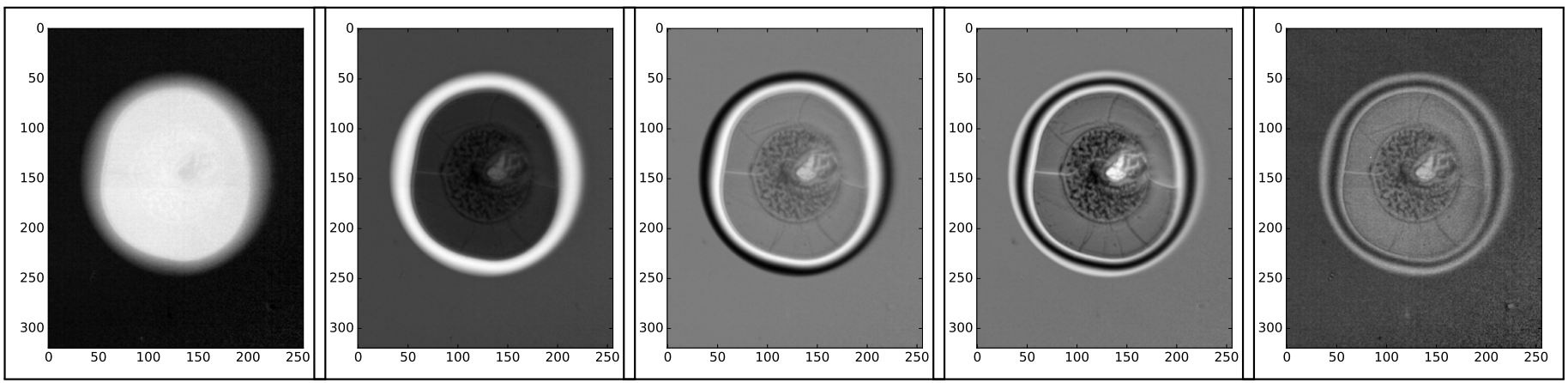
Droplet evaporation data approximation



Droplet evaporation data modes



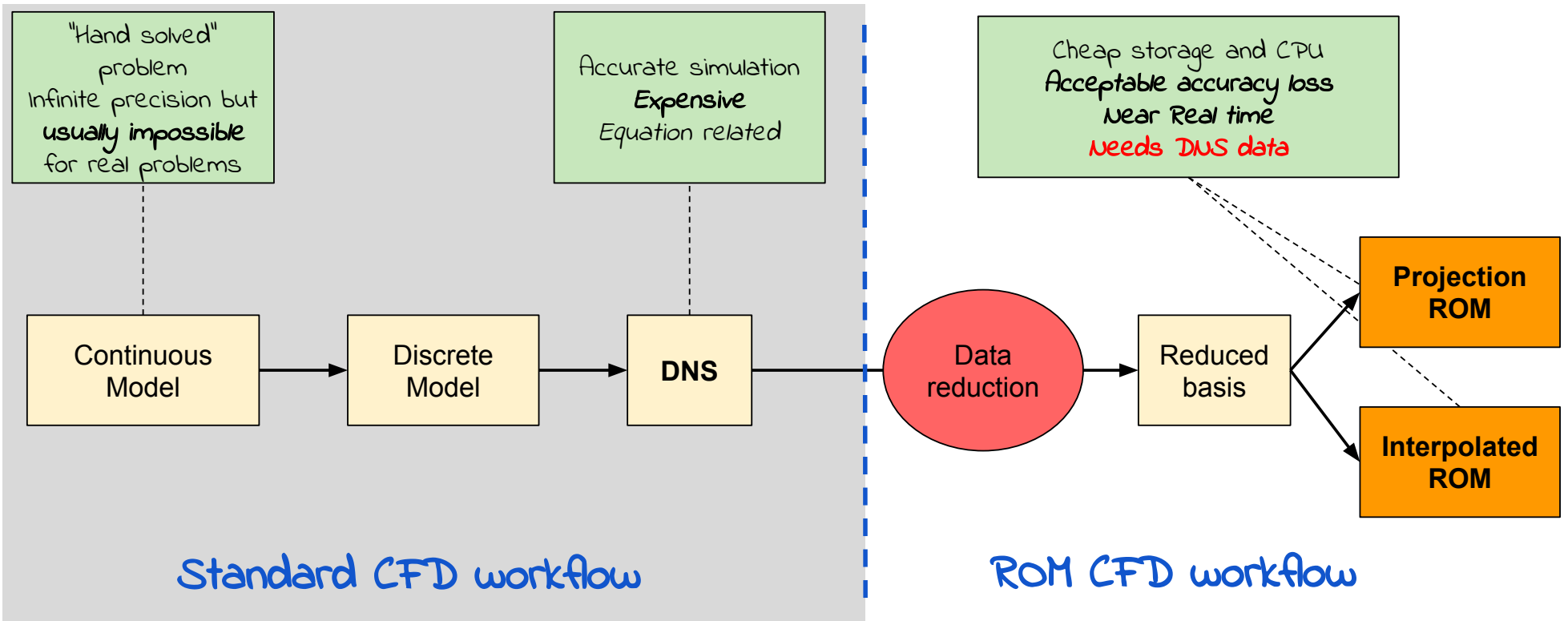




Spatial modes from vectorized data layout (1 to 5, left to right).

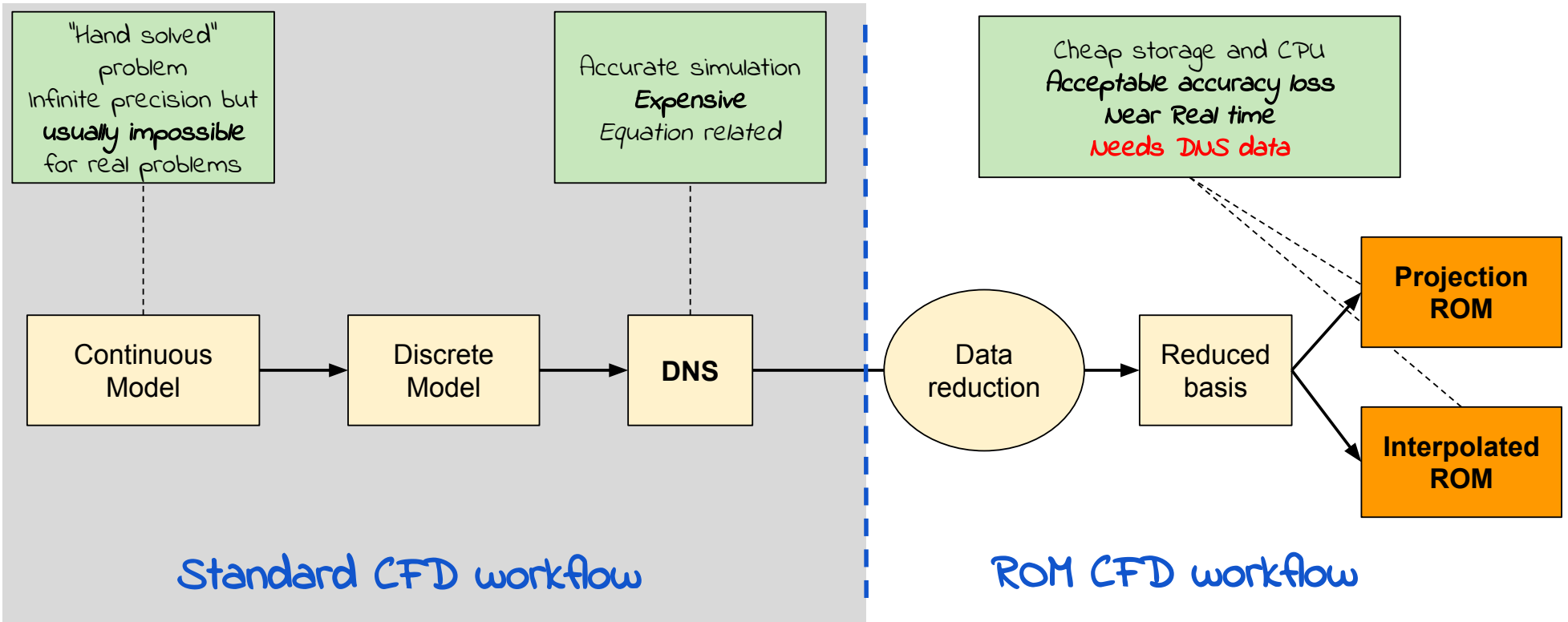
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Reduced Order Modeling (ROM)



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Reduced Order Modeling (ROM)



The goal of ROM

Reduce the computing time by orders of magnitude.

Typical applications

- Parametric study
- Shape optimization
- Control (flow, heat transfer...)

The goal of ROM

Reduce the computing time by orders of magnitude.

Typical applications

- Parametric study
- Shape optimization
- Control (flow, heat transfer...)

Example

ROM of standard flow for several Re

- Compute DNS for Several $\{Re_i\} \in [Re_l, Re_u]$ (and know the physics)
- Train ROM
- Compute ROM solutions for any $Re \in [Re_l, Re_u]$

Interpolated ROM
Why not interpolate directly?

Goal: Create a ROM for $Re \in [55 - 130]$, precompute a dozen of Re_i

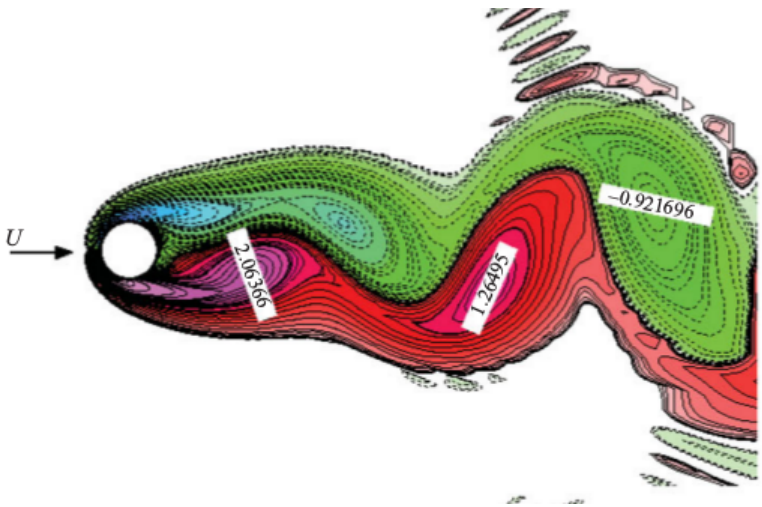


Figure: Flow past a circular cylinder, Re=75

Interpolated ROM
Why not interpolate directly?

Goal: Create a ROM for $Re \in [55 - 130]$, precompute a dozen of Re_i

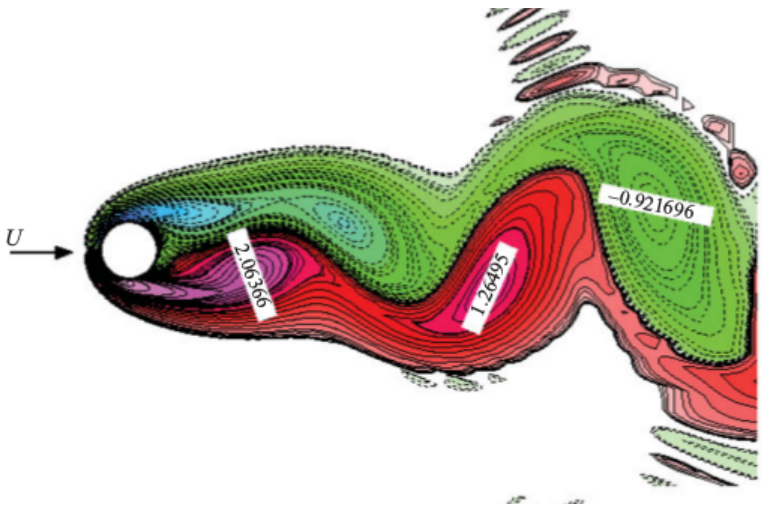


Figure: Flow past a circular cylinder, Re=75

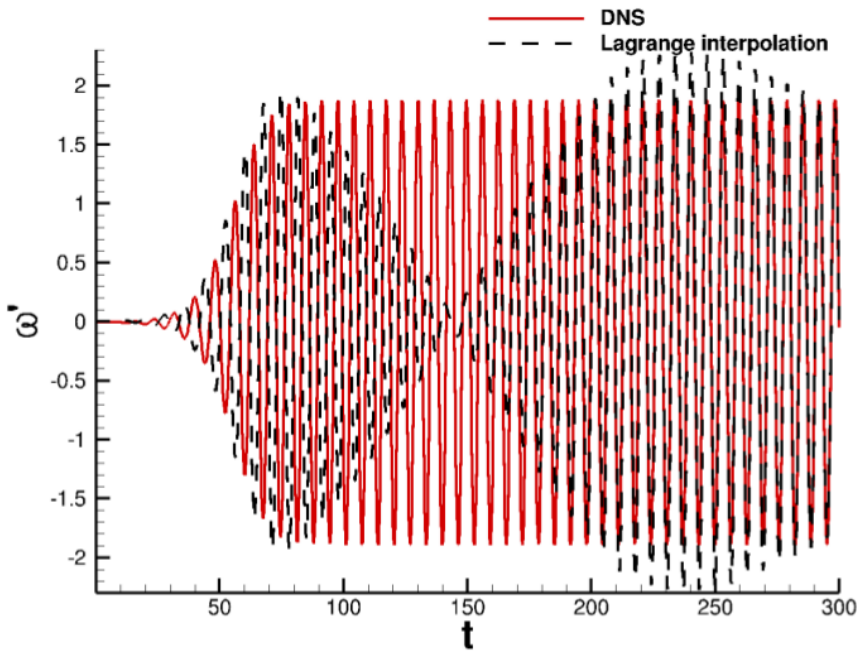
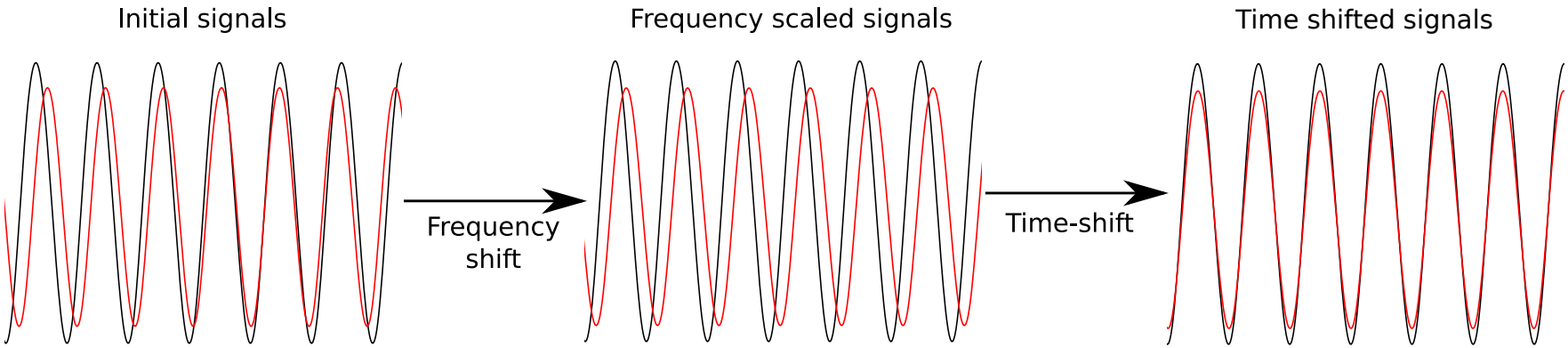


Figure: Direct Lagrange interpolation of the local time series in the cylinder wake

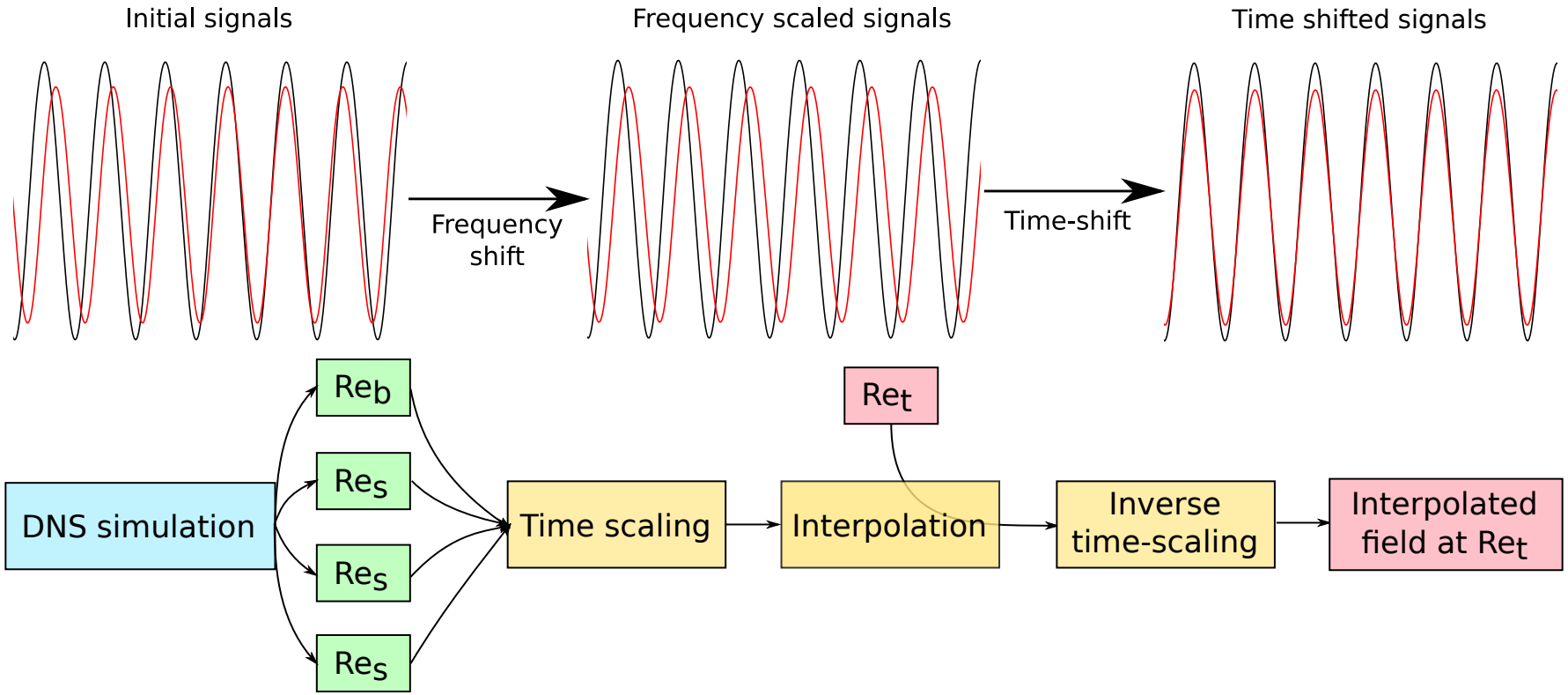
Using the physics : time-scaling interpolation

We take advantage of a relationship between *St* and *Re*.



Using the physics : time-scaling interpolation

We take advantage of a relationship between *St* and *Re*.



Time scaling interpolation in action

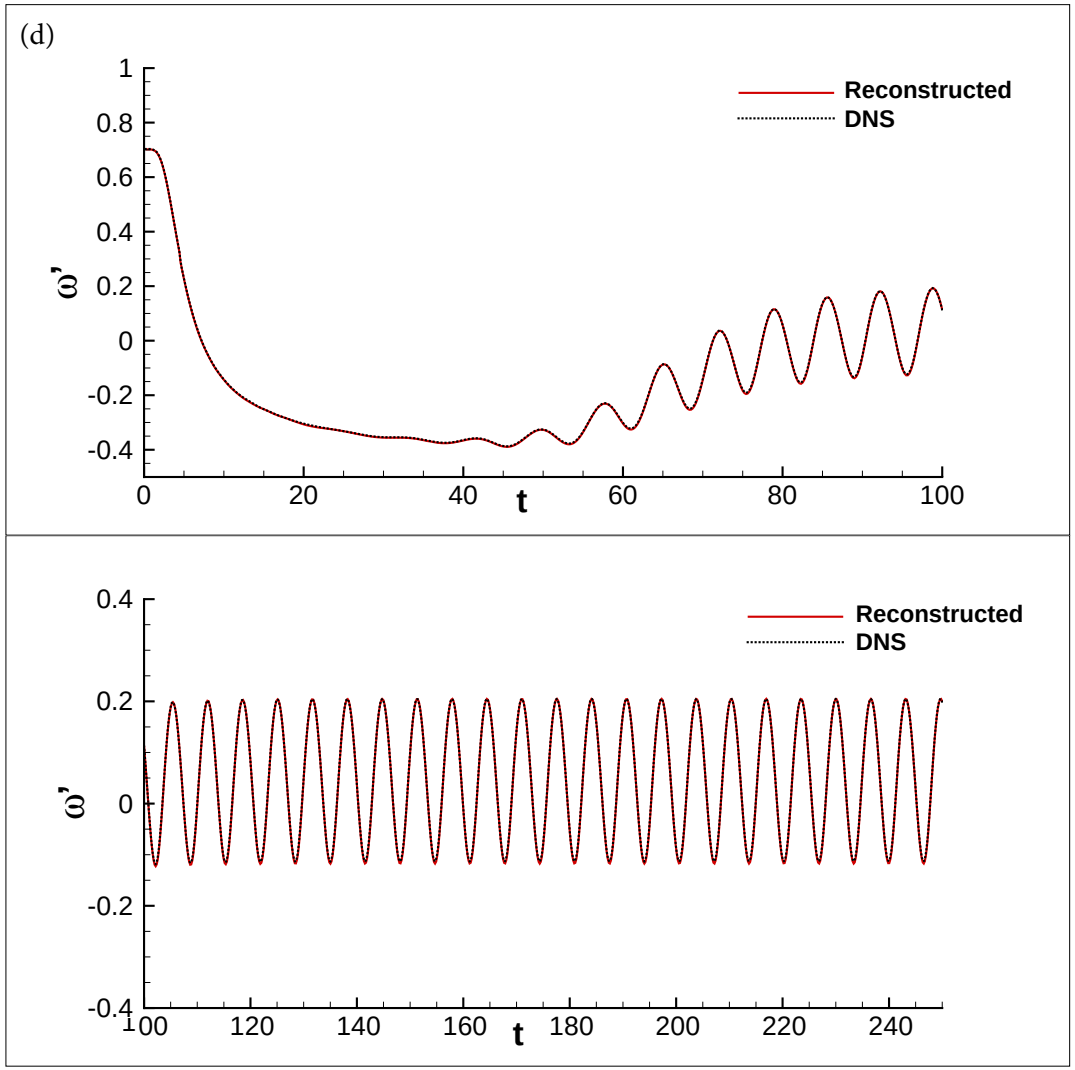
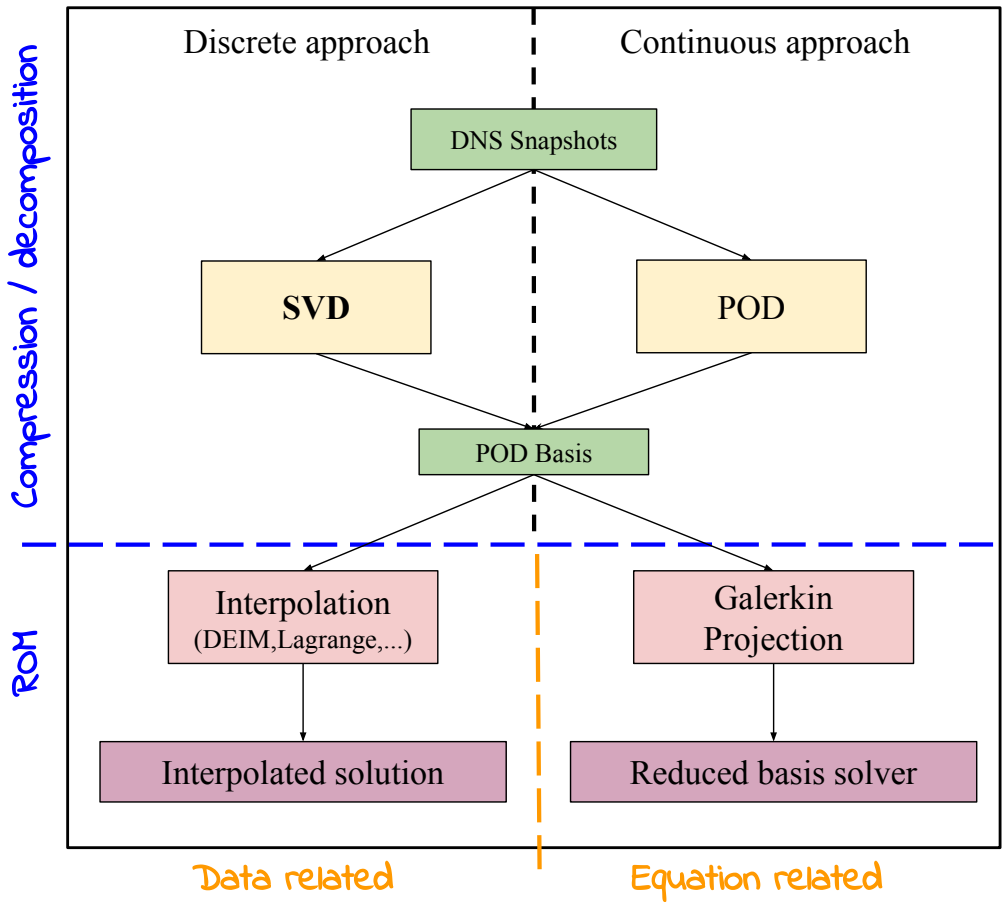


Figure: Reconstructed disturbance at (1.104,0) for Re=83, with donor $Re = \{78, 80, 86, 90\}$

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

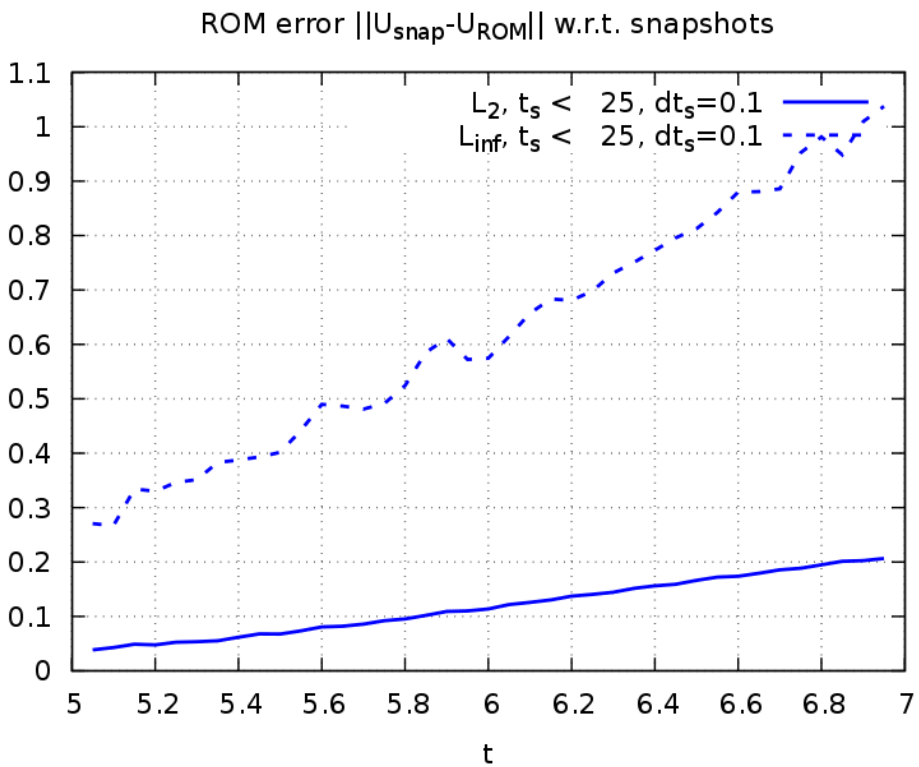
POD-Galerkin

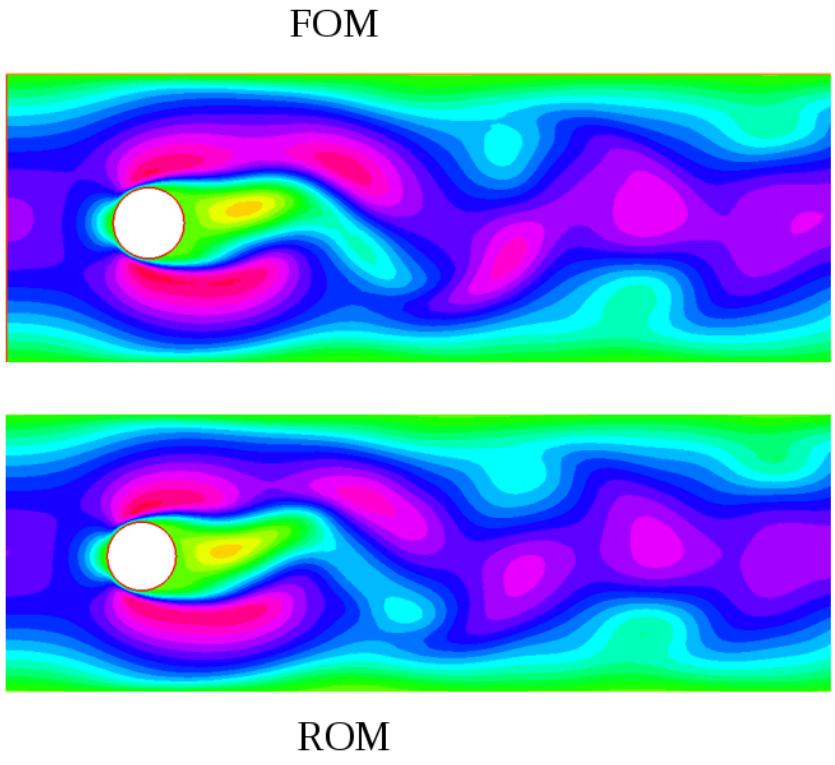
1. PDE in weak form
2. Full Order Model, $n_{dof} = O(10^6)$
3. Compute a reduced basis (RB) from snapshots e.g. with POD
4. Project PDE onto RB : typical POD-Galerkin ROM $n_{dof} = O(10)$
5. Stabilization term?, Tuning....



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Navier Stokes-PODG is unstable





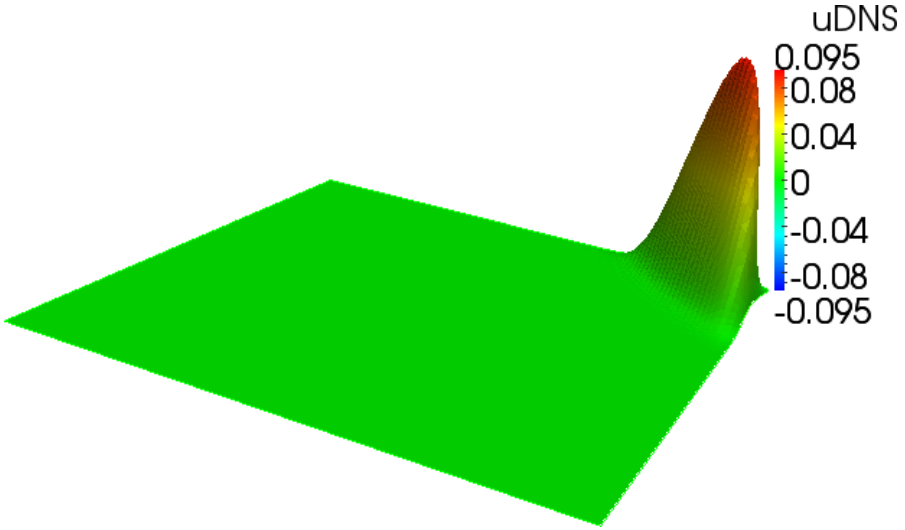
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Stabilized PODG in action

Joint work with S. Rubino and T. Chacon

A convection-dominated convection-diffusion-reaction problem

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



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Stabilized PODG in action

Joint work with S. Rubino and T. Chacon

A convection-dominated convection-diffusion-reaction problem

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

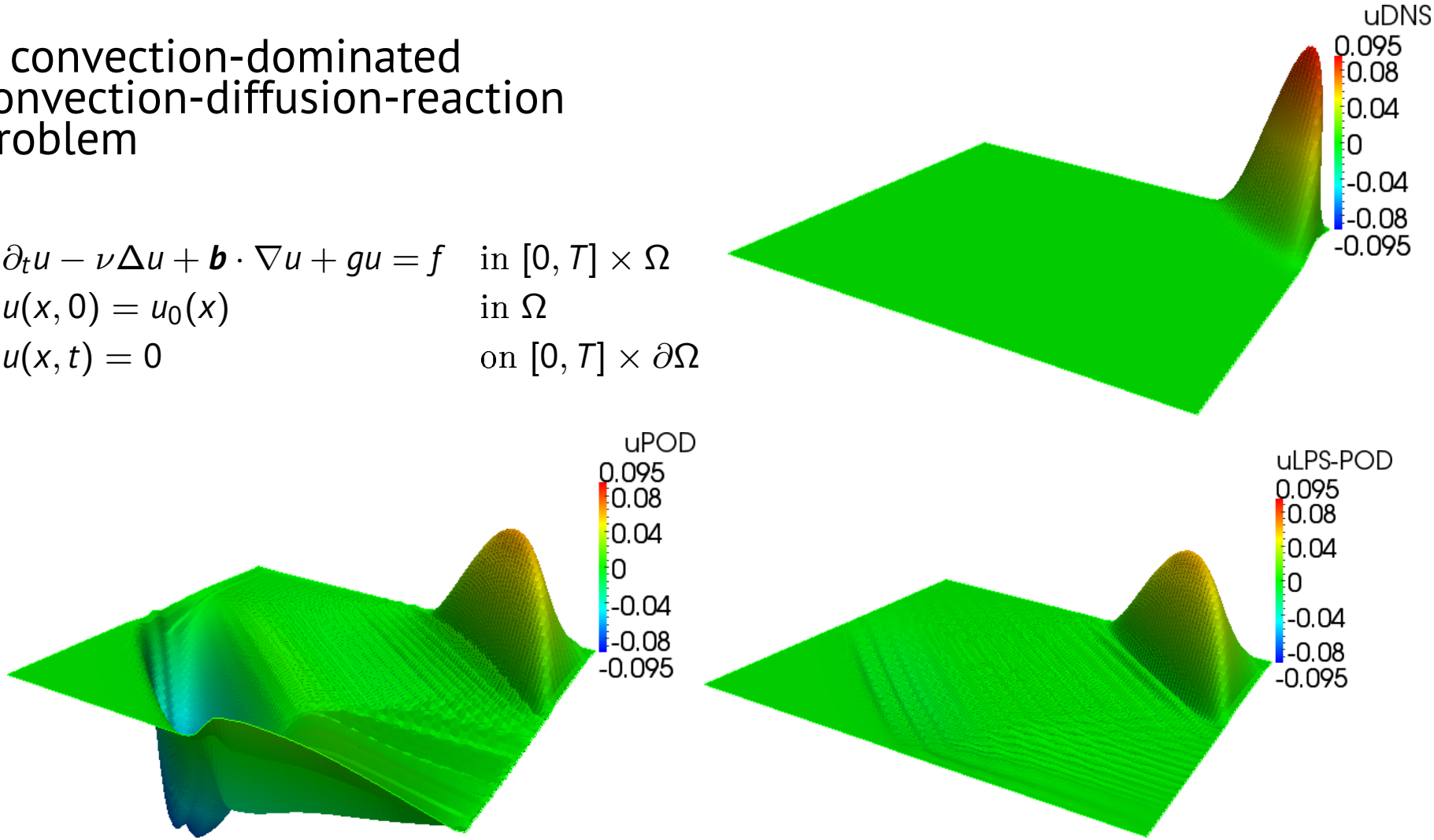


Figure: Numerical solution at $T = 1.25$:
DNS (top), POD model (bottom left), LPS-POD model (bottom right).

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Opening new horizons to ROM with Deep Learning

Idea: PDEs solution live in a manifold (w.r.t. parameters) which is inherently of **low dimension**.

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Opening new horizons to ROM with Deep Learning

Idea: PDEs solution live in a manifold (w.r.t. parameters) which is inherently of **low dimension**.

Breaking the curse of dimensionality

- **low dimensional** representation of non-linear features, including function approximation, blackbox approximation.
- Uncovering non linear features through DNN encoding
- Mixing the two previous items into a solver and applying ROM ideas i.e. building DeepROM.

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Opening new horizons to ROM with Deep Learning

Idea: PDEs solution live in a manifold (w.r.t. parameters) which is inherently of **low dimension**.

Breaking the curse of dimensionality

- **low dimensional** representation of non-linear features, including function approximation, blackbox approximation.
- Uncovering non linear features through DNN encoding
- Mixing the two previous items into a solver and applying ROM ideas i.e. building DeepROM.

Goals.

1. Construct a non-linear approximator (blackbox NN) to enable complete uncoupling of ROMs.

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Opening new horizons to ROM with Deep Learning

Idea: PDEs solution live in a manifold (w.r.t. parameters) which is inherently of **low dimension**.

Breaking the curse of dimensionality

- **low dimensional** representation of non-linear features, including function approximation, blackbox approximation.
- Uncovering non linear features through DNN encoding
- Mixing the two previous items into a solver and applying ROM ideas i.e. building DeepROM.

Goals.

1. Construct a non-linear approximator (blackbox NN) to enable complete uncoupling of ROMs.
2. Construct a fluid dynamics analysis/compressor system. Generating (very) low rank representations and feature map.

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Opening new horizons to ROM with Deep Learning

Idea: PDEs solution live in a manifold (w.r.t. parameters) which is inherently of **low dimension**.

Breaking the curse of dimensionality

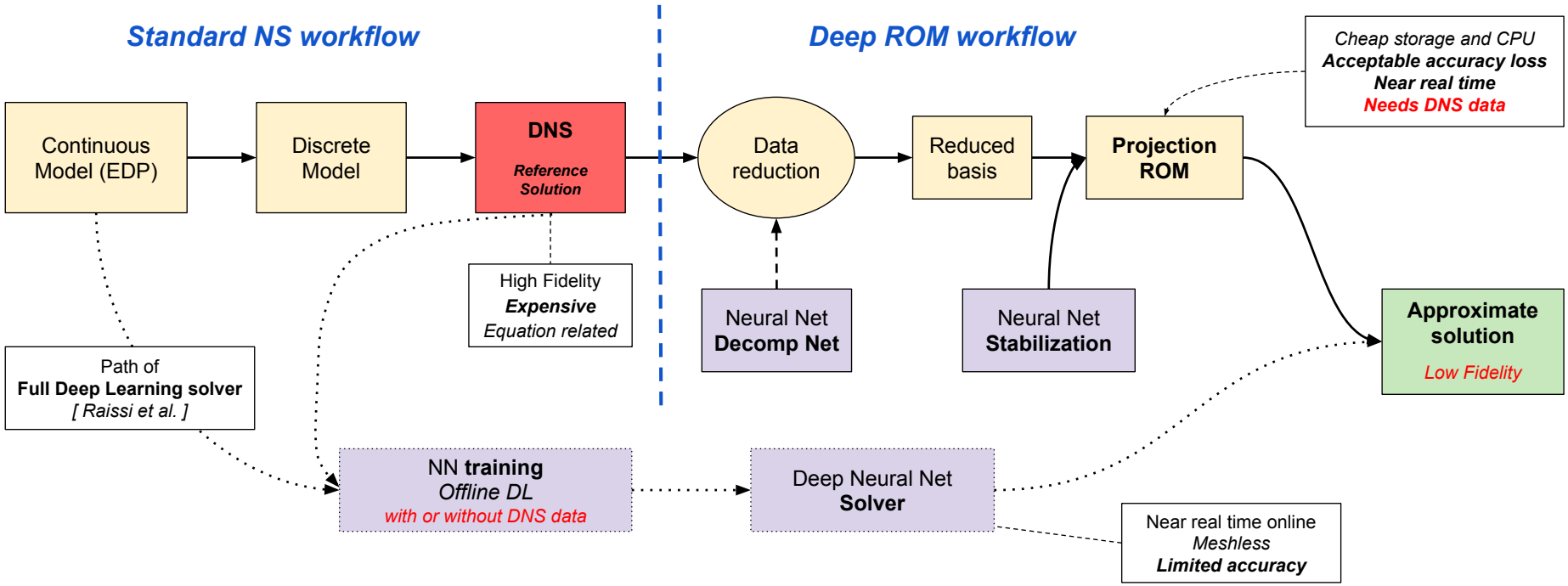
- **low dimensional** representation of non-linear features, including function approximation, blackbox approximation.
- Uncovering non linear features through DNN encoding
- Mixing the two previous items into a solver and applying ROM ideas i.e. building DeepROM.

Goals.

1. Construct a non-linear approximator (blackbox NN) to enable complete uncoupling of ROMs.
2. Construct a fluid dynamics analysis/compressor system. Generating (very) low rank representations and feature map.
3. PDE solving using Deep learning and make appropriate use of ROM knowledge.

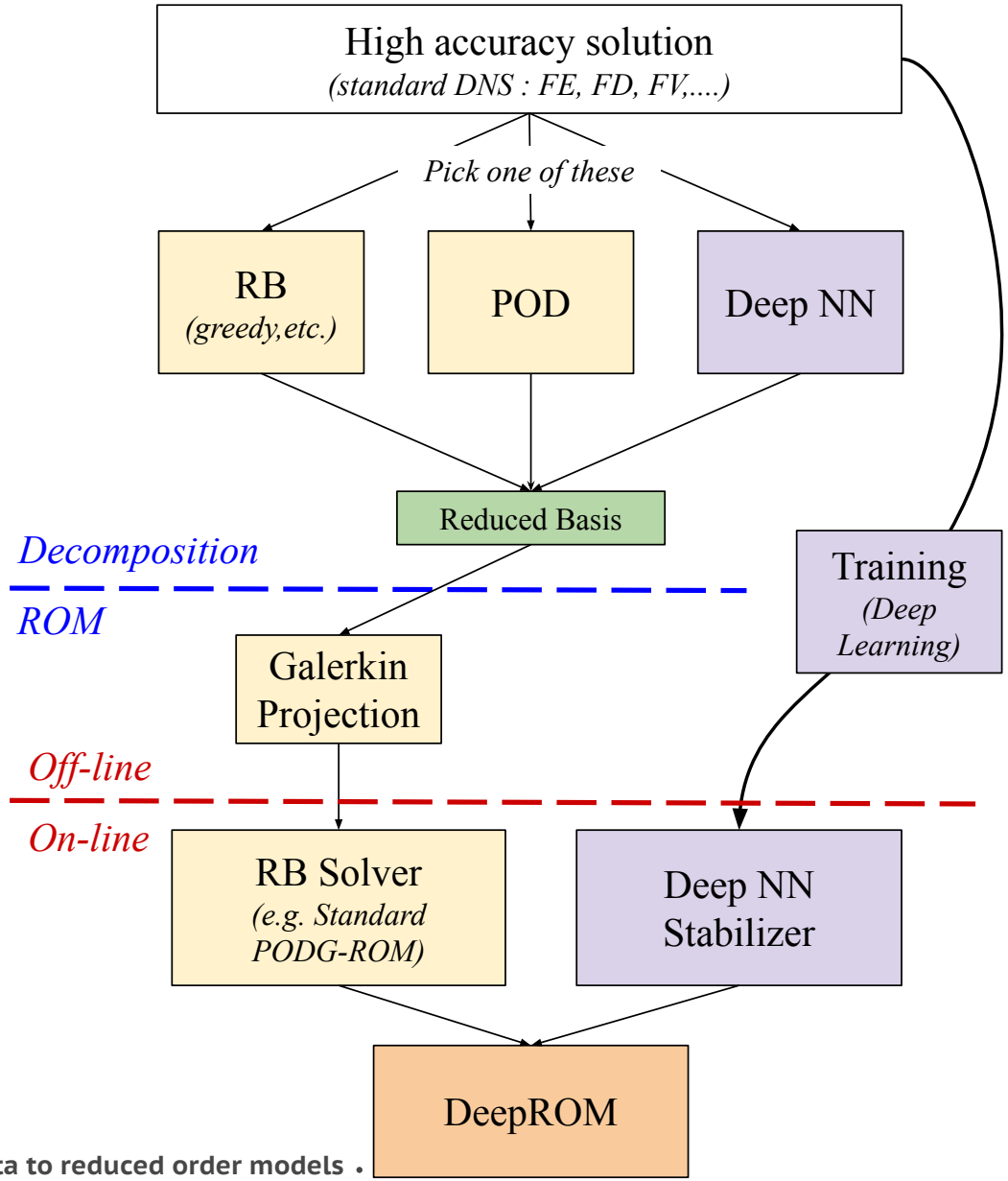
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Making up for ROM shortcomings



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

A DeepROM diagram



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Data decomposition
- Bivariate decomposition
- Tensor formats and approximation
- Numerics

Reduced order models for complex flows

Research Project: Deep Reduced Order Model

Conclusion and perspectives

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

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
- Time-scaling interpolation
- State of the art of DNN with PDEs and DeepROM plan

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Future work

DeepROM

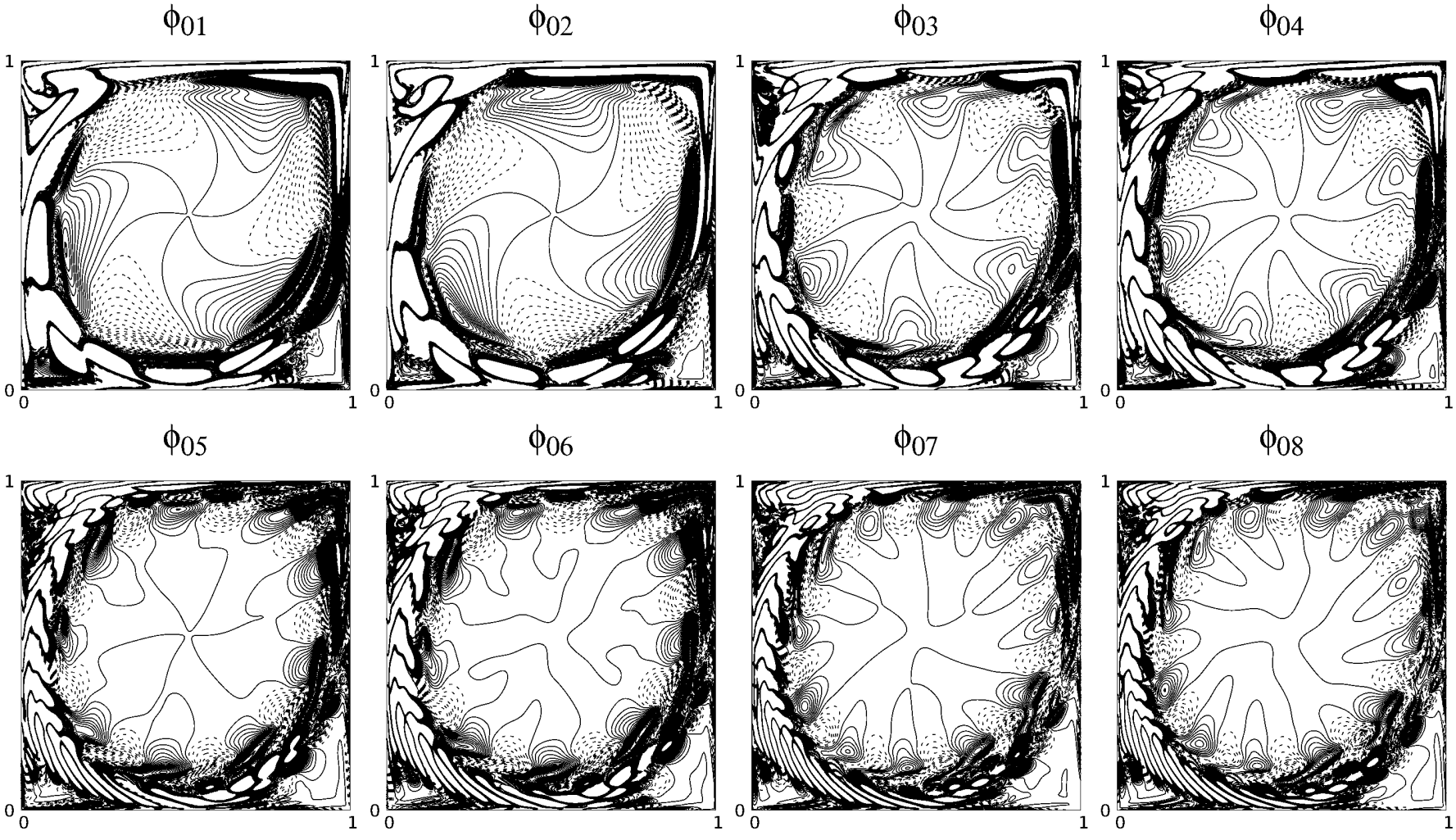
- DNN non-linear interpolation
- Projected ROM (Stabilized POD-G)
- DNN basis generation (GAN,LSTM...)
- Explore DNN architectures

Explore DNN and PDE synergies

- Meshless solving of PDE
- PDE + data hyper reduction
- PDE aided interpolation

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Thank you for your attention!



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

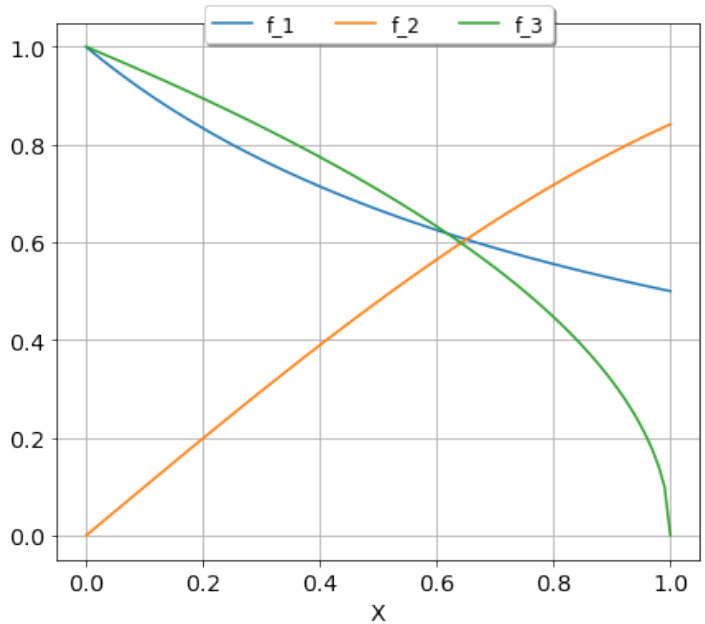
MLP approximator

In order to be able to increase the dimension, we introduce the following functions defined on $\Omega = [0,1]^d$:

$$f_1(\mathbf{x}) = \frac{1}{1+\sum_i x_i}$$

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

$$f_3(\mathbf{x}) = \sqrt{1-\prod_i x_i}$$



As usual the error is measured relative to the exact function f in L^2 or l^2 norm. As the difference is usually negligible for regular grid, we will simply use l^2:

$$\mathcal{E} = \frac{||f - f_{NN}||_2}{||f||_2}$$

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

MLP example

Following *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition* by **Aurélien Géron** (Chap 10 for now) , we will use the sequential `keras` API on our simple "interpolation" problem.

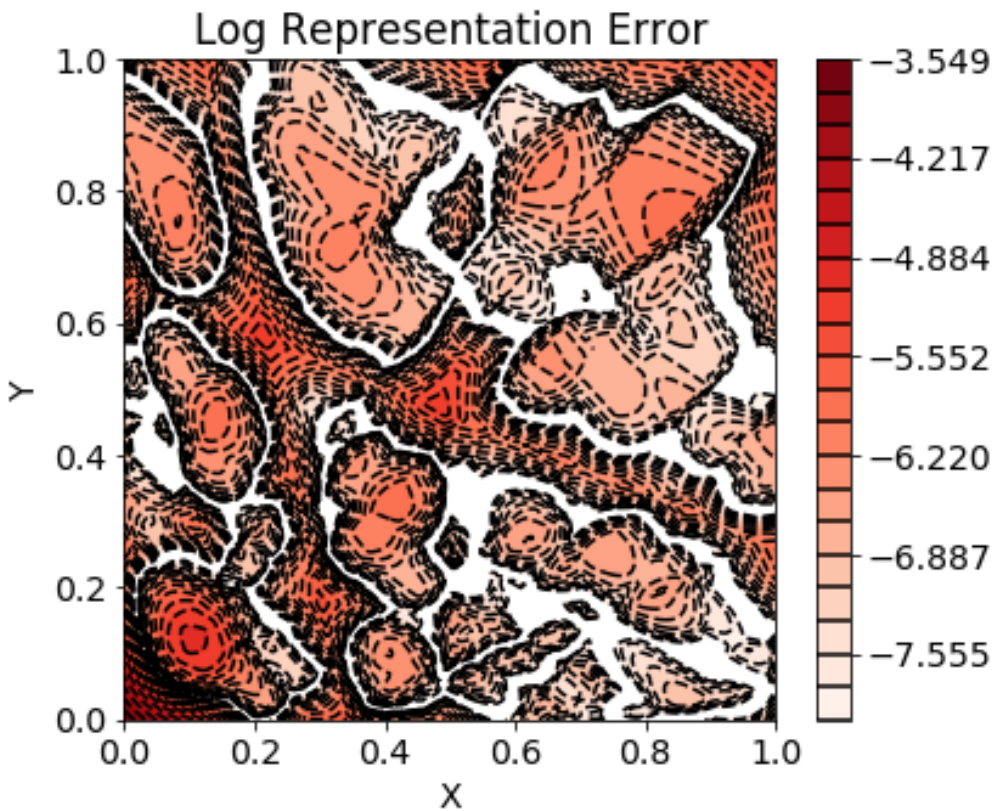
Entrée [158]:
```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[d,]))
N=10
L=2
for i in range(L):
    model.add(keras.layers.Dense(N, activation="relu")) # L deep layers
model.add(keras.layers.Dense(1, activation="linear"))  # output layer
# See Book for more elegant writing
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_9 (Flatten)          (None, 2)                 0
_________________________________________________________________
dense_19 (Dense)             (None, 100)               300
_________________________________________________________________
dense_20 (Dense)             (None, 100)               10100
_________________________________________________________________
dense_21 (Dense)             (None, 1)                 101
=================================================================
Total params: 10,501
Trainable params: 10,501
Non-trainable params: 0
_________________________________________________________________
```

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

MLP example

```
L2err= 0.0031758773723378868
Linf_error= 0.028756439685821533
f= <function f1 at 0x000001D5EE136E18>
```



Parameters:

N_train	501
N_test	172
N_valid	33
loss	MSE
batch size	1
epoch	10

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Optimized search on hyperparameters

We can also do an optimized search on parameters using Keras. Then we use less trainig data (N_ train=101,N_ valid=44,N_ test=32).

Entrée [213]:
```
from scipy.stats import reciprocal
from sklearn.model_selection import RandomizedSearchCV

param_distribs = {
    "n_hidden": [0, 1, 2, 3],
    "n_neurons": np.arange(1, 100),
    "learning_rate": reciprocal(3e-4, 3e-2),
}

rnd_search_cv = RandomizedSearchCV(keras_reg, param_distribs, n_iter=10, cv=
rnd_search_cv.fit(X_train, f_train, epochs=100,
                  validation_data=(X_valid, f_valid),
                  callbacks=[keras.callbacks.EarlyStopping(patience=10)])
```

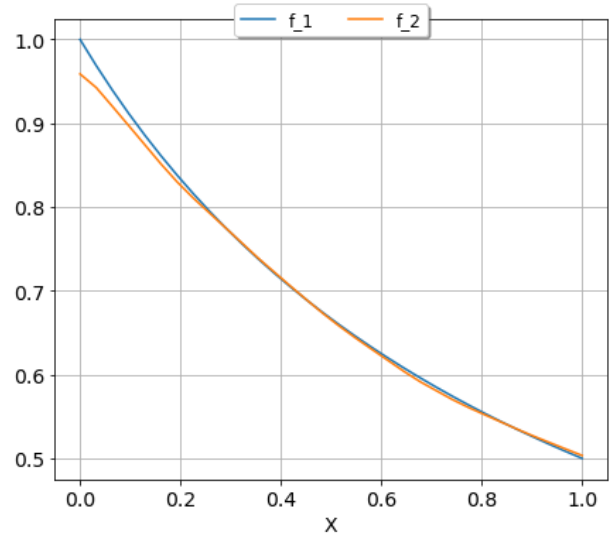
...

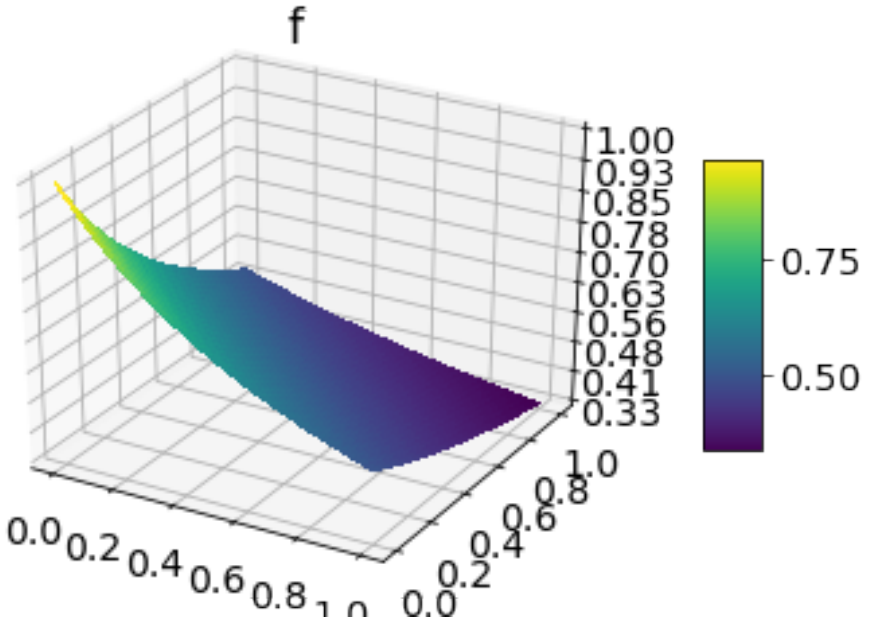
Entrée [214]:
```
rnd_search_cv.best_params_
```

Out[214]:
```
{'learning_rate': 0.024827442804036565, 'n_hidden': 3, 'n_neurons': 74}
```

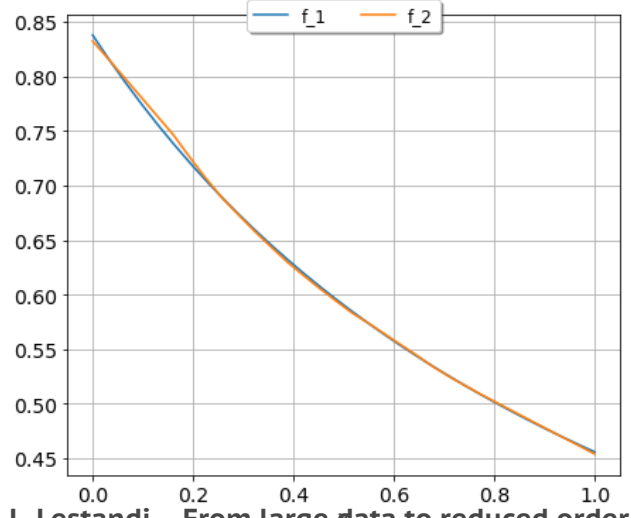
Optimized MLP approximation

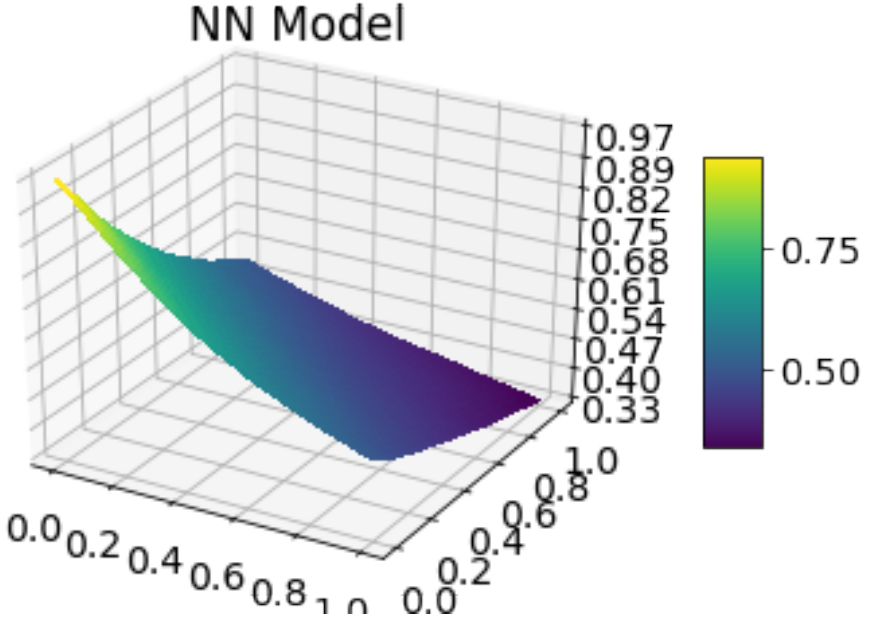
```
y= 0.0
grid size: 32
```





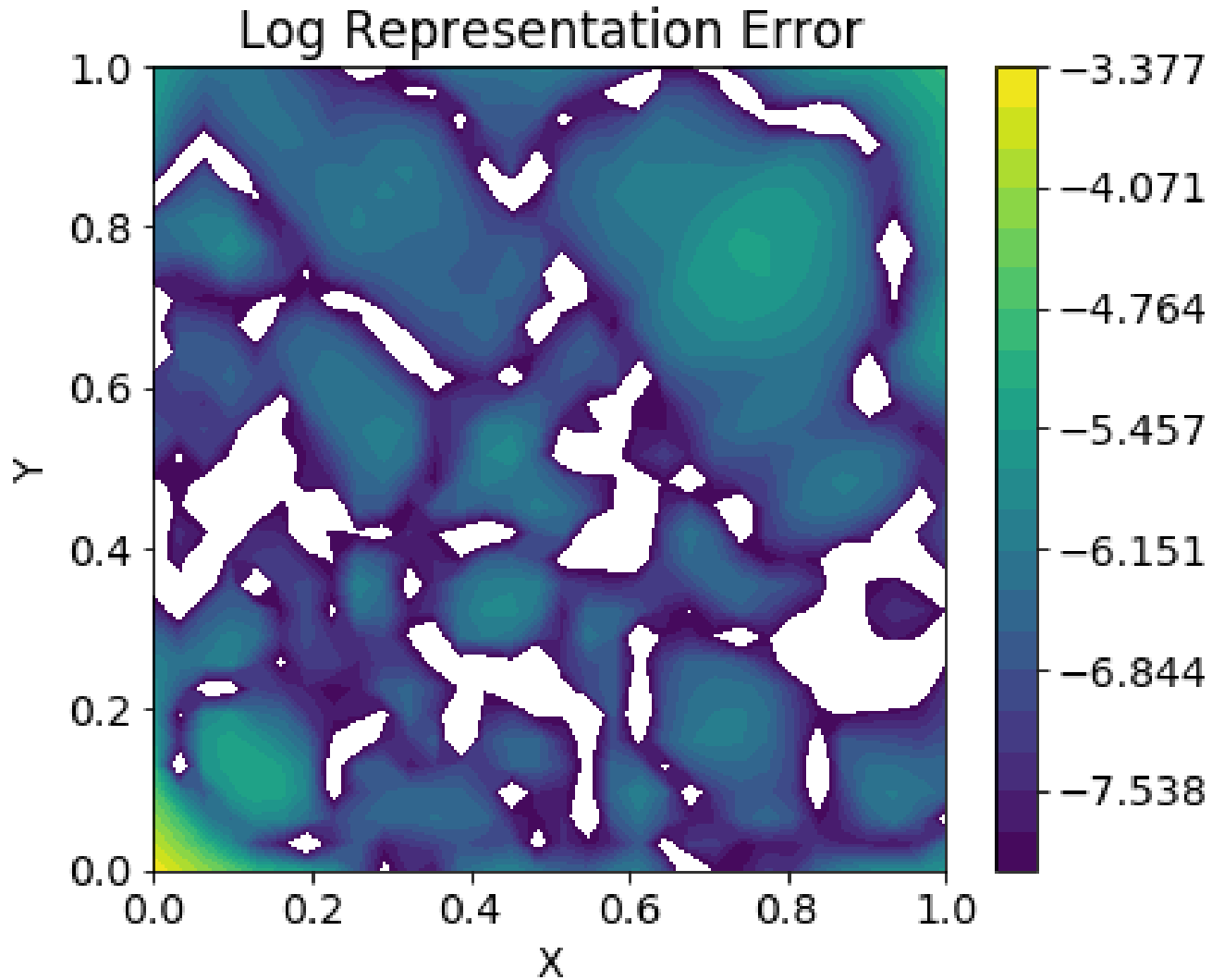

```
y= 0.1935483870967742
grid size: 32
```





NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Optimized MLP approximation



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Short term roadmap to improve approximation

- Improve understanding of tensor flow and MLP engineering
  - loss function
  - training algorithm
  - batch size
  - architecture
  - ...
- training points (from Cartesian to greedy/random sampling of space)
- number of dimensions influence
- preprocessing : centering, spread, etc.
- literature examples and existing solutions

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:

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

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:

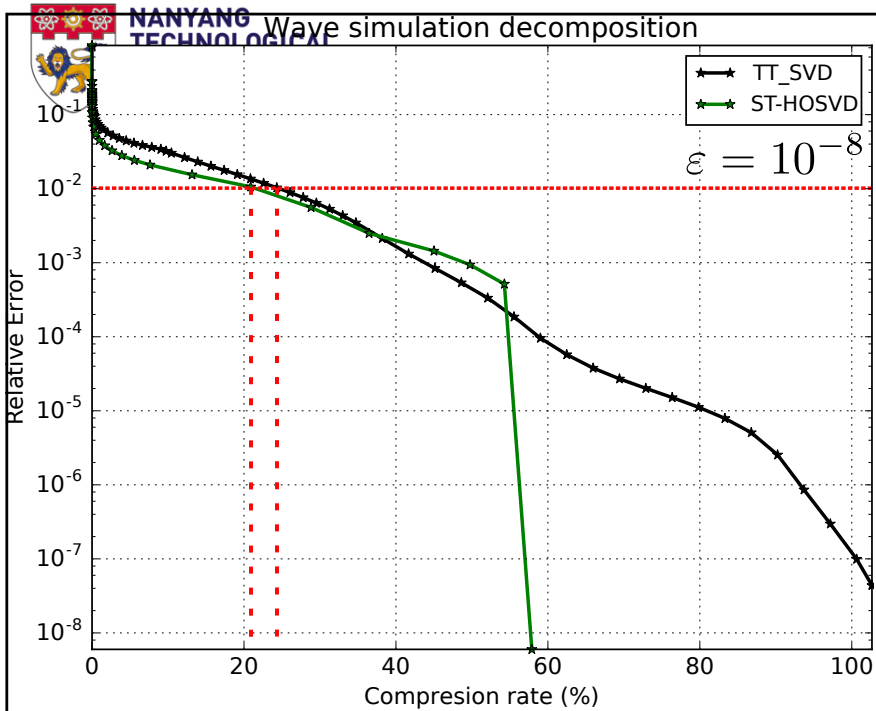
$$\boldsymbol{x}^r = \text{span}\left\{\varphi_1,\dots,\varphi_r\right\}.$$

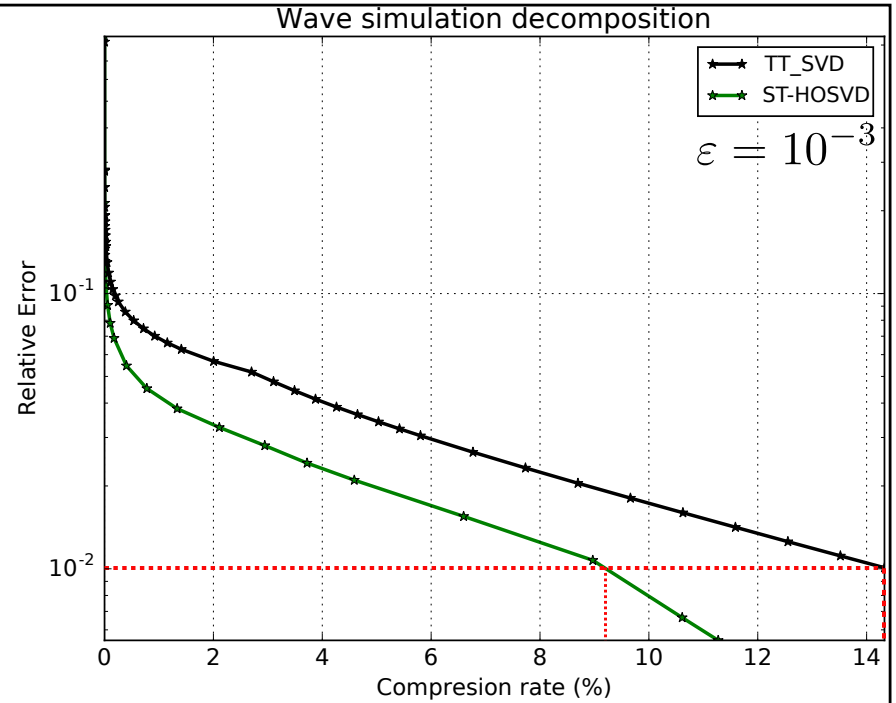
We look for $\boldsymbol{u}(\boldsymbol{x},t) \approx \boldsymbol{u}_r(\boldsymbol{x},t) = \boldsymbol{u}_D(\boldsymbol{x}) + \widetilde{\boldsymbol{u}}_r(\boldsymbol{x},t) = \boldsymbol{u}_D(\boldsymbol{x}) + \sum_{i=1}^r a_i(t)\varphi_i(\boldsymbol{x})$,
We introduce the stabilization term

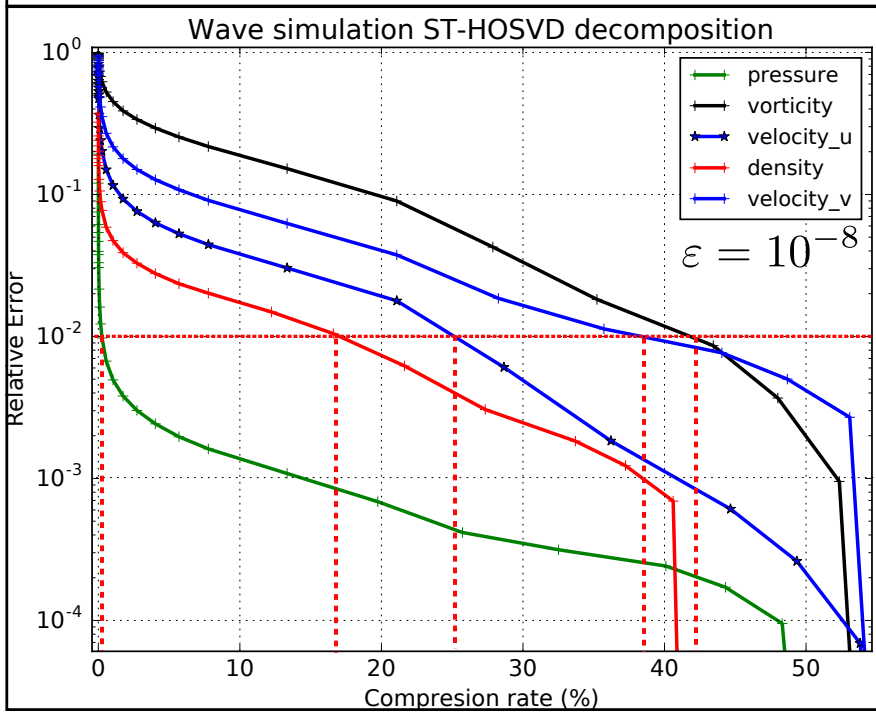
$$(P_R'(\widetilde{\boldsymbol{u}}_r\cdot\ \nabla\widetilde{\boldsymbol{u}}_r), P_R'(\widetilde{\boldsymbol{u}}_r\cdot\nabla\varphi))_\tau$$

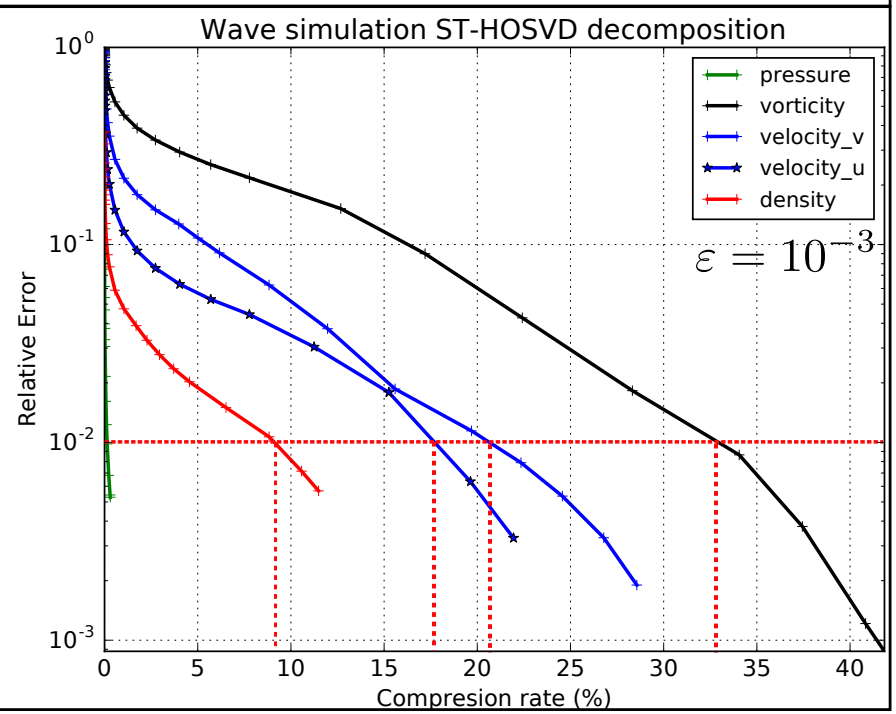
Then the Streamline-Derivative PODG reads:

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


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 (%)

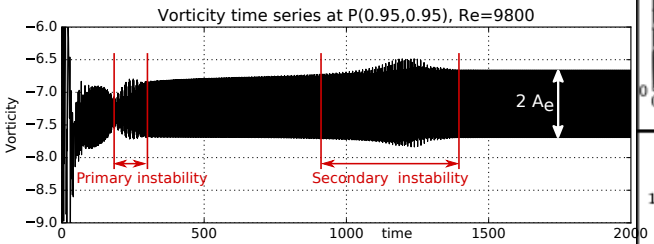

Wave simulation ST-HOSVD decomposition
pressure
vorticity
velocity_v
velocity_u
density
$\varepsilon = 10^{-3}$
Relative Error
Compresion rate (%)

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

POD, an analysis tool

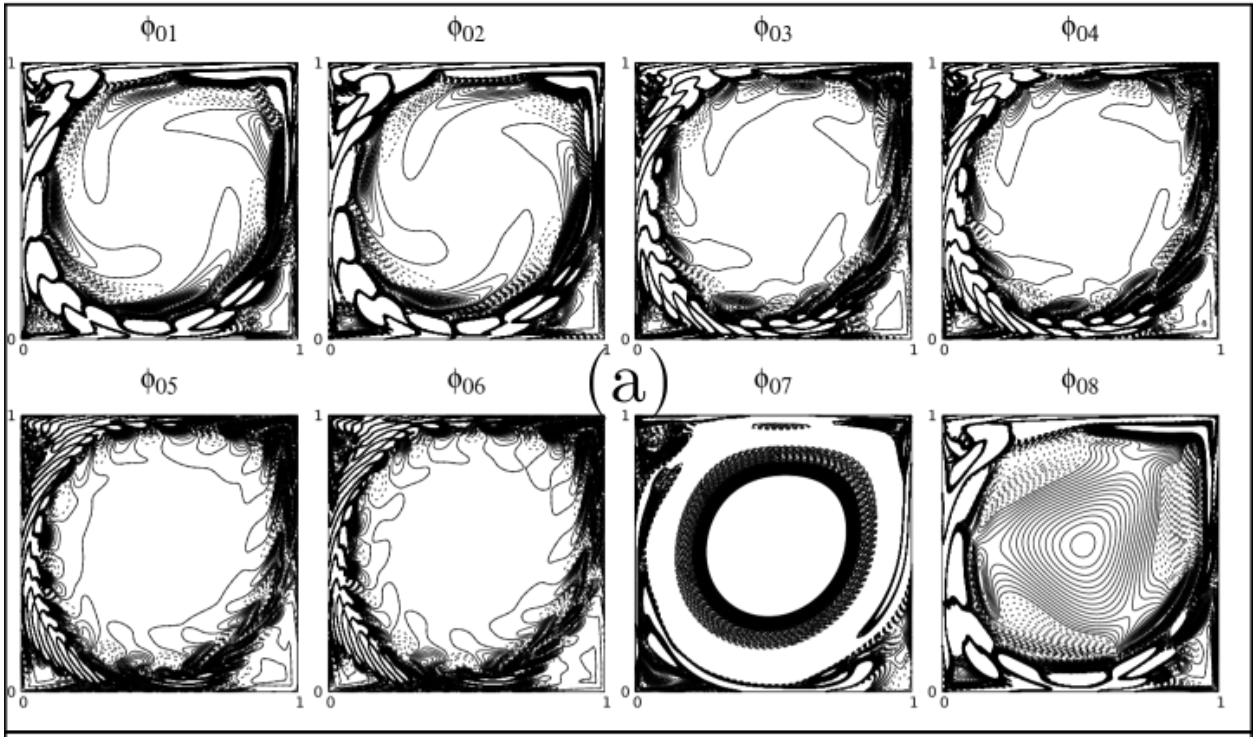
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

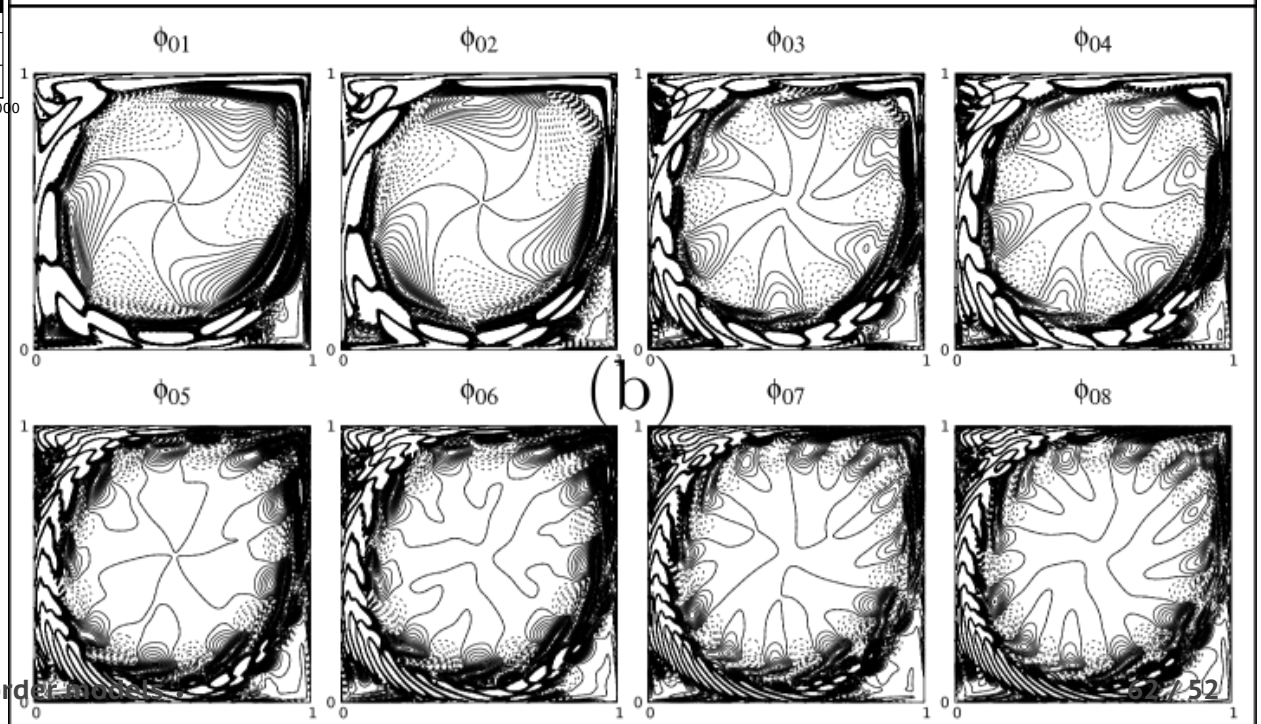
POD, an analysis tool



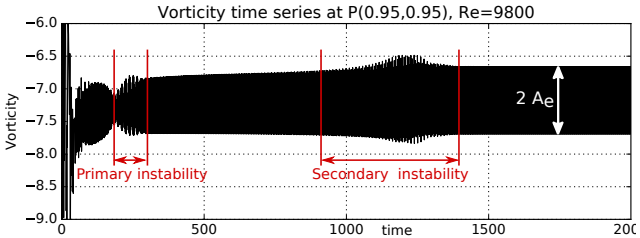
(a) $t \in [500, 600]$

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





POD, an analysis tool



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

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