A multiprocess and multithreaded strategy for parallel proper orthogonal decomposition in a reduced order modeling framework

Florian Krötz

MathLab, Mathematics Area, SISSA International School for Advanced Studies, Trieste, Italy

Ulm University, Ulm, Germany

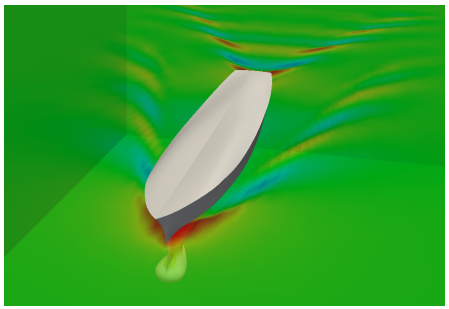
19.06.2020

Agenda

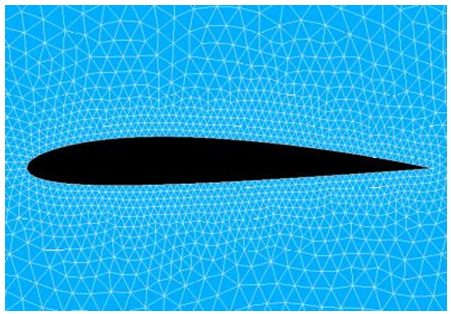
1. Reduced Basis Method
2. Parallel POD algorithms
3. Implementation and Benchmarks
4. Integration in `EZyRB` with an example

Physical Problems

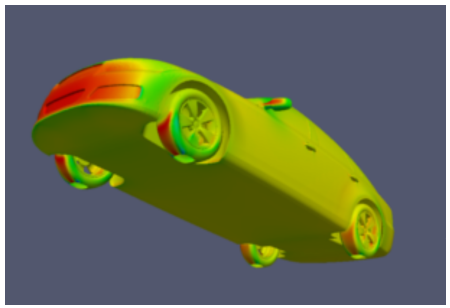
Complex phenomena described using **parametrized** **Partial Differential Equations (PDE)**



Naval Engineering



Aeronautics



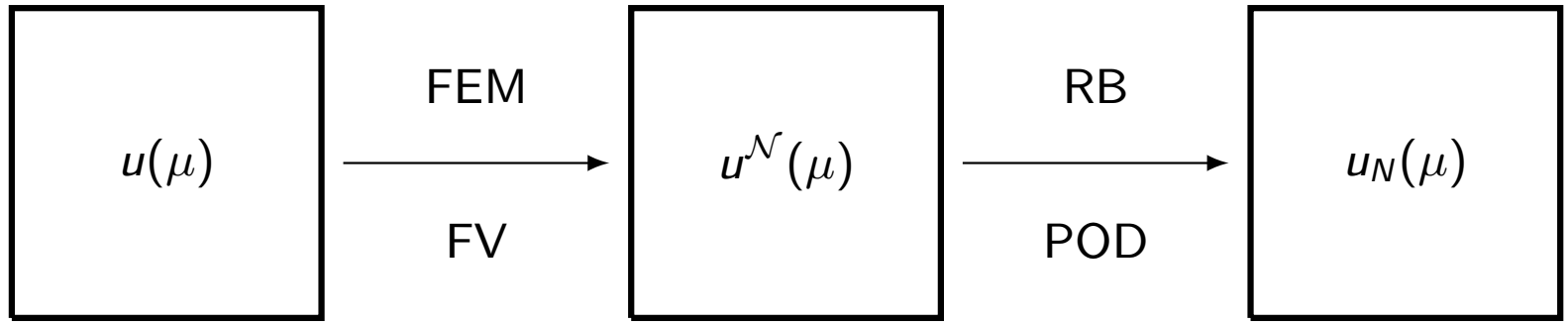
Automotive

Solving the PDEs using fine discretization like **Finite Element**, **Finite Volume**, ...

Problem: Many evaluations are expensive.

- **optimizing** the parameter (**multi query** application)
- **controlling** the parameter (**real time** application)

RB workflow



$$\text{find } u(\mu) \in \mathbb{X}: \quad a(u(\mu), v; \mu) = f(v; \mu) \quad \forall v \in \mathbb{X}.$$

Truth problem

$$\text{find } u^{\mathcal{N}}(\mu) \in \mathbb{X}^{\mathcal{N}}: \quad a(u^{\mathcal{N}}(\mu), v; \mu) = f(v; \mu) \quad \forall v \in \mathbb{X}^{\mathcal{N}}.$$

Truth solutions

$$u^{\mathcal{N}}(\mu) = \sum_{i=1}^{\mathcal{N}} u_i^{\mathcal{N}}(\mu) \cdot \varphi_i, \quad \mathbb{X}^{\mathcal{N}} = \text{span}(\varphi_1, ..., \varphi_{\mathcal{N}})$$

Goal: Approximate solution manifold

$$\mathbb{M}^{\mathcal{N}} = \{u^{\mathcal{N}}(\mu), \quad \mu \in \mathbb{P}\}$$

Reduced Space

$$\mathbb{X}_N = \text{span}(\xi_1, ..., \xi_N), \qquad N \ll \mathcal{N}$$

Reduced solutions

$$u_N(\mu) = \sum_{i=1}^{N} u_{N,i}(\mu) \cdot \xi_i$$

Reduced problem

$$\text{find } u_N(\mu) \in \mathbb{X}_N : \quad a(u_N(\mu), v; \mu) = f(v; \mu) \quad \forall v \in \mathbb{X}_N$$

POD basis generation

The POD is a method to find the optimal orthogonal space in a linear manner.

- Set based approach $\Xi_{train} = \{\mu_1, ..., \mu_{n_{train}}\}$
- $\mathbb{X}_{n_{train}} = \text{span}\left\{u^{\mathcal{N}}(\mu) | \mu \in \Xi_{train}\right\}$

Eigen value problem (EVP)

$$C(v) = \frac{1}{n_{train}} \sum_{m=1}^{n_{train}} \left(v, u^{\mathcal{N}}(\mu_m)\right)_{\mathbb{X}} u^{\mathcal{N}}(\mu_m), \quad v \in \mathbb{X}_{n_{train}}$$

$$\left(C(\xi_n), u^{\mathcal{N}}(\mu_m)\right)_{\mathbb{X}} = \lambda_n \left(\xi_n, u^{\mathcal{N}}(\mu_m)\right)_{\mathbb{X}}, \quad 1 \leq m \leq n_{train}$$

POD error

$$\frac{1}{n_{train}} \sum_{\mu \in \Xi_{train}} \inf_{w_N \in \mathbb{X}_{n_{train}}} \|u^{\mathcal{N}}(\mu) - w_N\|_{\mathbb{X}}^2 = \sum_{n=N+1}^{n_{train}} \lambda_n$$

- Truncate all modes $> N$

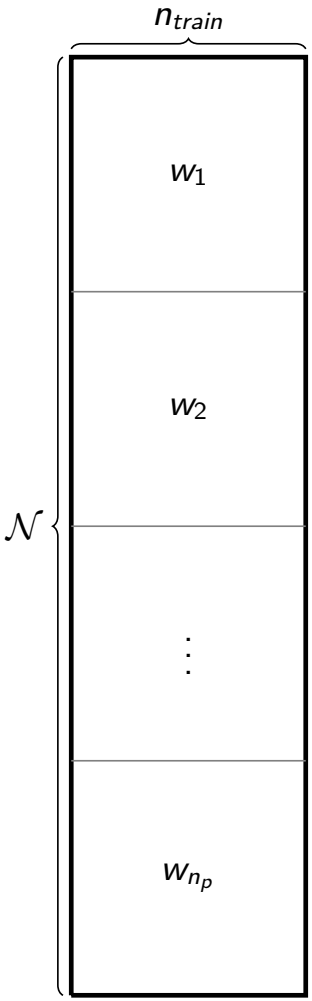
Snapshot Matrix

$$\mathcal{W} = \begin{bmatrix} | & | & & | \\ \underline{u}^{\mathcal{N}}(\mu_1) & \underline{u}^{\mathcal{N}}(\mu_2) & \dots & \underline{u}^{\mathcal{N}}(\mu_{n_{train}}) \\ | & | & & | \end{bmatrix}$$

Truth solution

$$u^{\mathcal{N}}(\mu) = \sum_{i=1}^{\mathcal{N}} u_i^{\mathcal{N}}(\mu)\varphi_i, \qquad \underline{u}^{\mathcal{N}}(\mu) = \begin{pmatrix} u_1^{\mathcal{N}}(\mu) \\ \vdots \\ u_{\mathcal{N}}^{\mathcal{N}}(\mu) \end{pmatrix}$$

Structure of the snapshot matrix



Partitioning cases

- Domain decomposition
- Manually blocked

Proper Orthogonal Decomposition

EVP

- Solving eigen value problem (EVP) of the snapshot matrix

$$\mathcal{W}^T\mathcal{W} = VDV^T$$

$$U = \mathcal{W}VD^{-\frac{1}{2}}$$

SVD

- Singular Value Decomposition (SVD) of the snapshot matrix

$$\mathcal{W} = USV^T$$

Relation singular values and eigen values

$$\mathcal{W}^T\mathcal{W} = (USV^T)^T USV^T = VSU^T USV^T = VS^2V^T$$

The left singular are computed with

$$U = \mathcal{W}VS^{-1}$$

Parallel SVD approach introduced by Beattie, Borggaard and Iliescu in 2007

Singular Value Decomposition (SVD) of the snapshot Matrix W

$$\mathcal{W} = USV^T$$

U are the POD modes.

Compute SVD of the local snapshot matrices $\mathcal{W} = [W_1^T ... W_{n_p}^T]^T$

$$\text{svd}(W_k) = [U, S, \boldsymbol{V_k}]$$

Compress local right singular vectors $\boldsymbol{V_k}$ using SVD

$$\hat{\mathcal{V}} = [V_1 ... V_{n_p}] \qquad \text{svd}(\hat{\mathcal{V}}) = [\mathcal{T}, \mathcal{M}, \boldsymbol{\mathcal{V}}]$$

Compute left singular vectors

$$\text{svd}(\mathcal{W}\mathcal{V}) = [\boldsymbol{\mathcal{U}}, \mathcal{S}, \mathcal{Z}]$$

Parallel EVP

We compute

$$\mathcal{W}^T\mathcal{W} = VDV^T$$

The left singular are computed with

$$U = \mathcal{W}VD^{-\frac{1}{2}}$$

Expensive operations
dimension $\mathcal{N}$

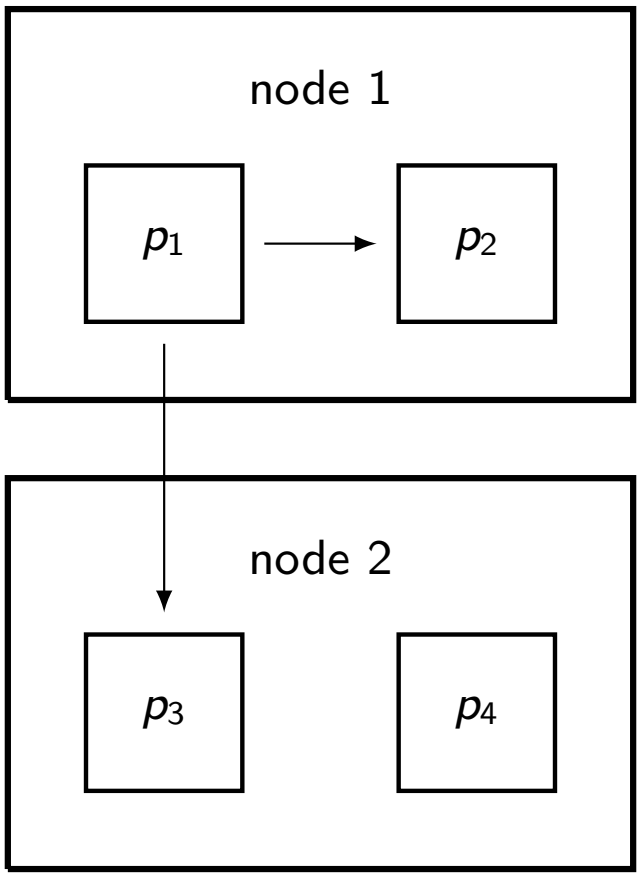
- $\mathcal{W}^T\mathcal{W}$
- $U = \mathcal{W}VD^{-\frac{1}{2}}$

Cheap operations
dimension n_{train}

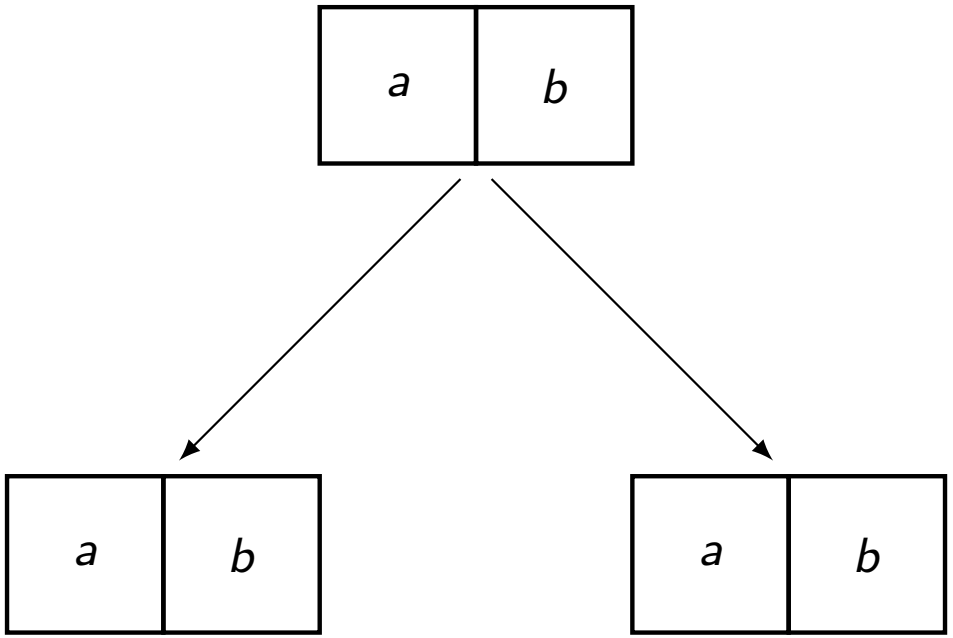
- $\mathcal{W}^T\mathcal{W} = VDV^T$ (solving EVP)
- $VD^{-\frac{1}{2}}$

MPI - Message Passing Interface

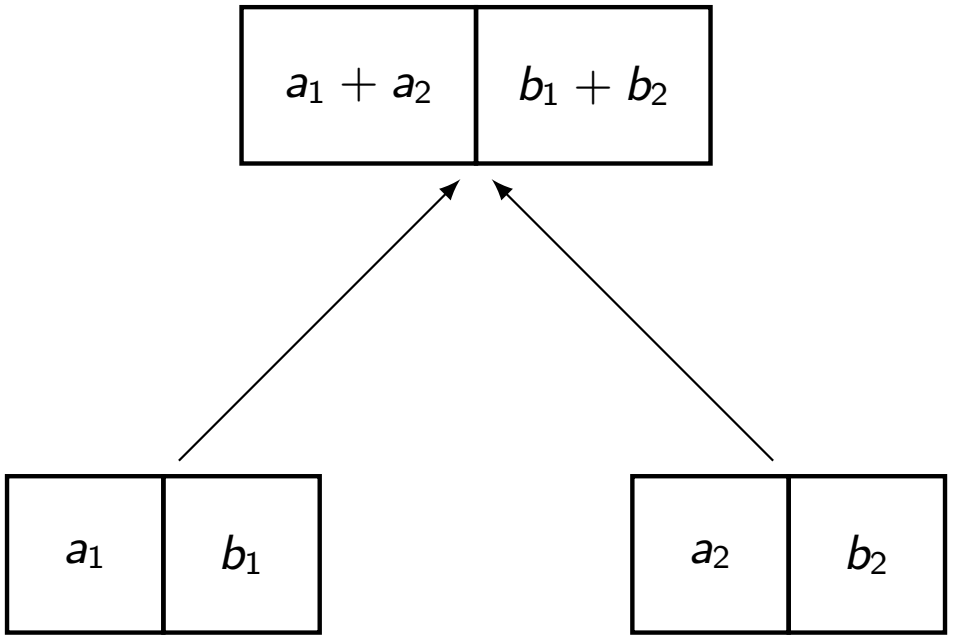
- A standard for a library interface for parallel programs.
- **process-based**: interface to send message from one process to another process.
- **distributed memory**: every memory has its own virtual memory.



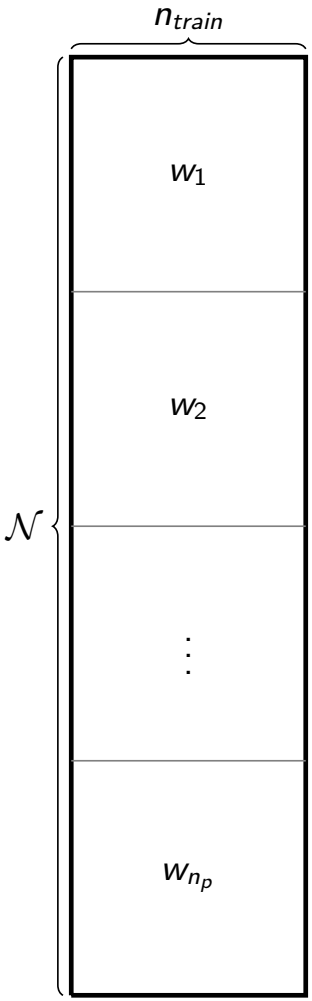
MPI_Bcast



MPI_Reduce with MPI_sum

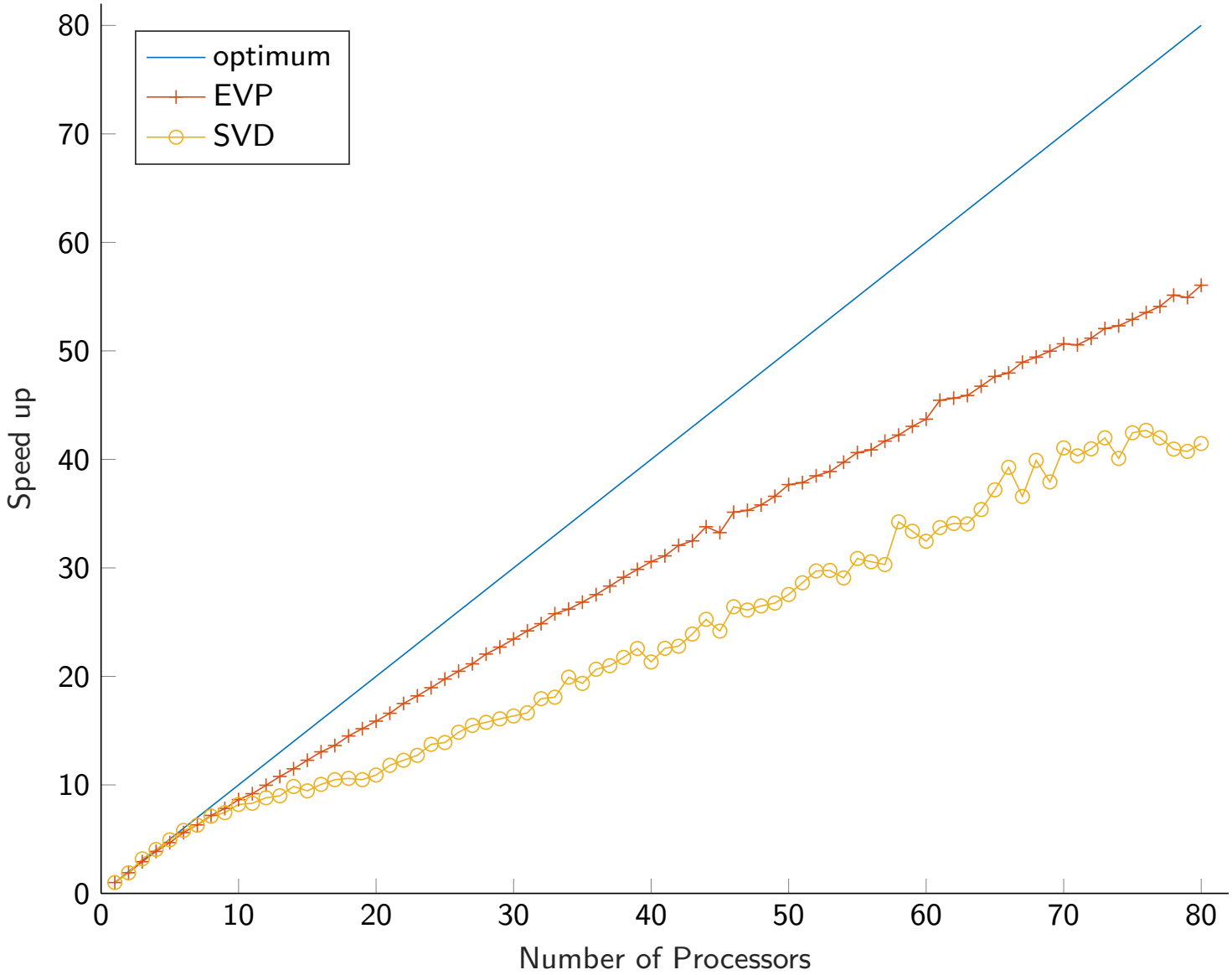


Partitioning of the Snapshot matrix

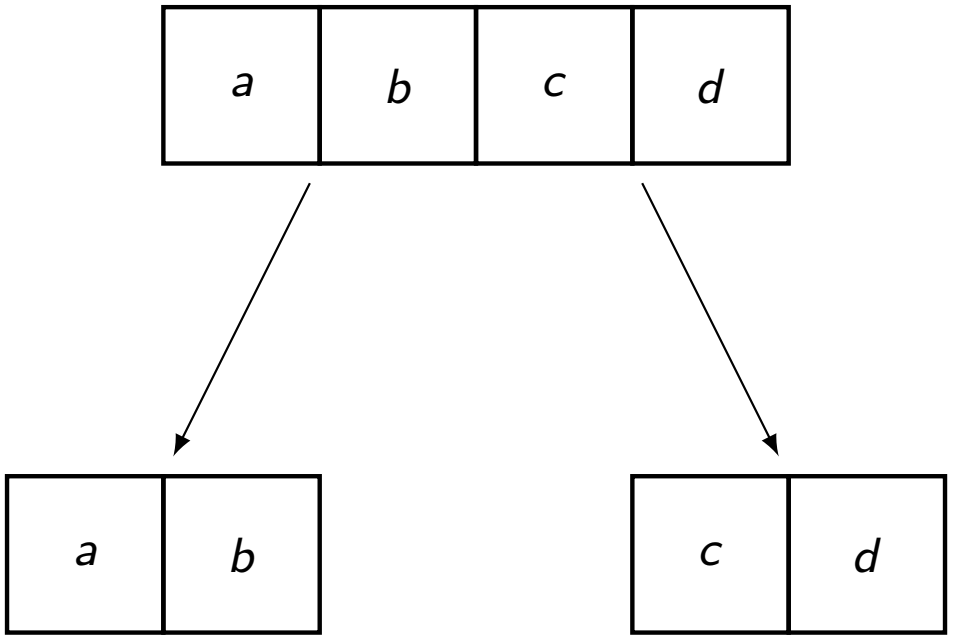


1. compute local correlation matrix $w_i^T w_i$
2. reduce local correlation matrices to get global correlation matrix C
3. Solve EVP $C = VDV^T$
4. compute $VD^{-\frac{1}{2}}$
5. Broadcast $VD^{-\frac{1}{2}}$ to every processor
6. $u_i = w_i \cdot VD^{-\frac{1}{2}}$

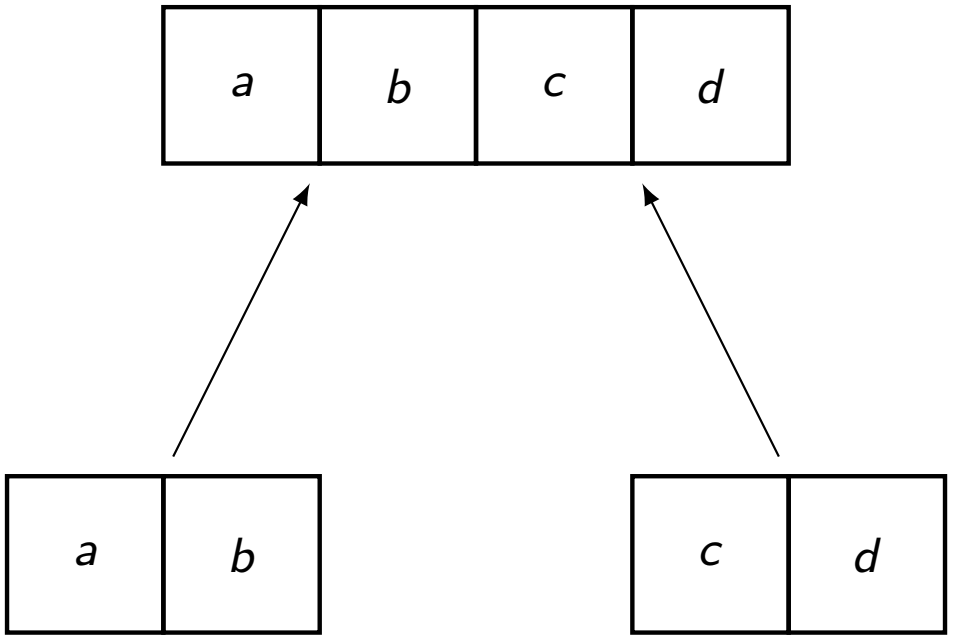
Snapshot matrix is already distributed random $10^7 \times 300$ matrix



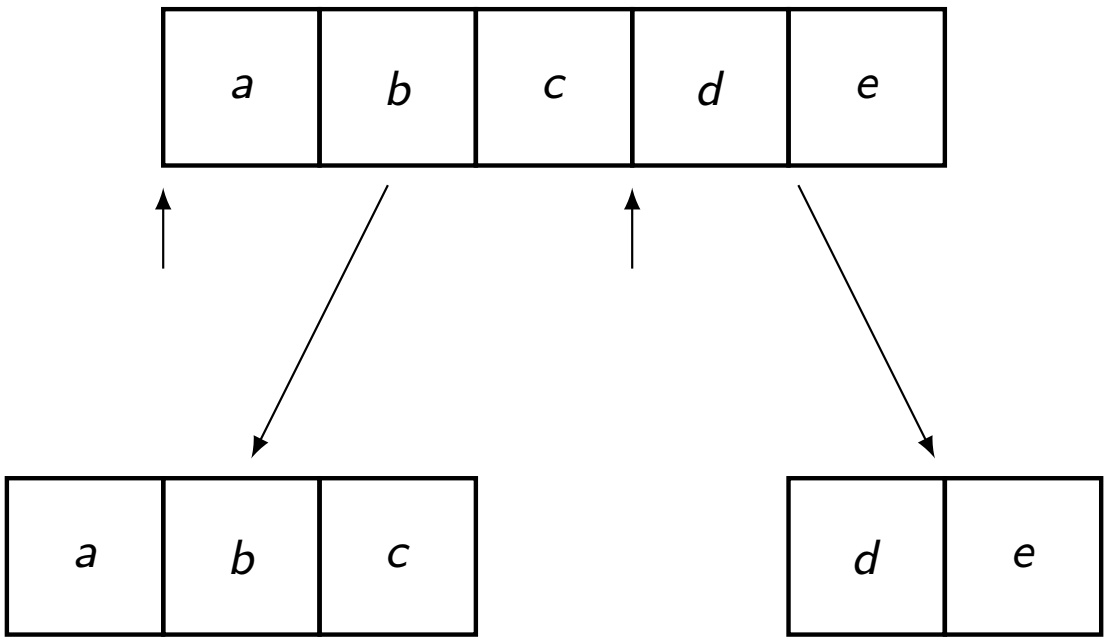
MPI_Scatter



MPI_Gather

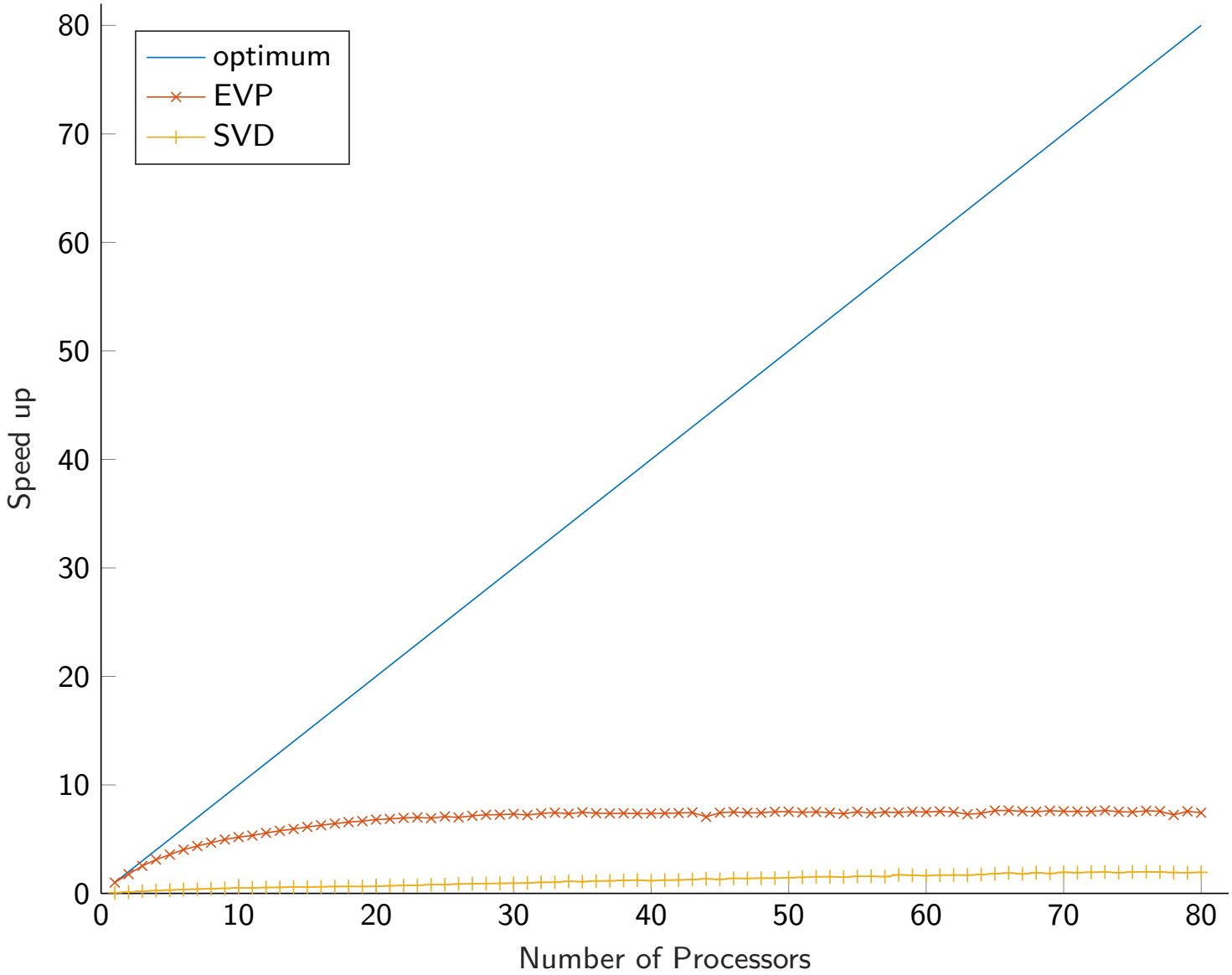


MPI_Scatterv and MPI_Gatherv



sendcount = [3,2], displacement = [0,3]

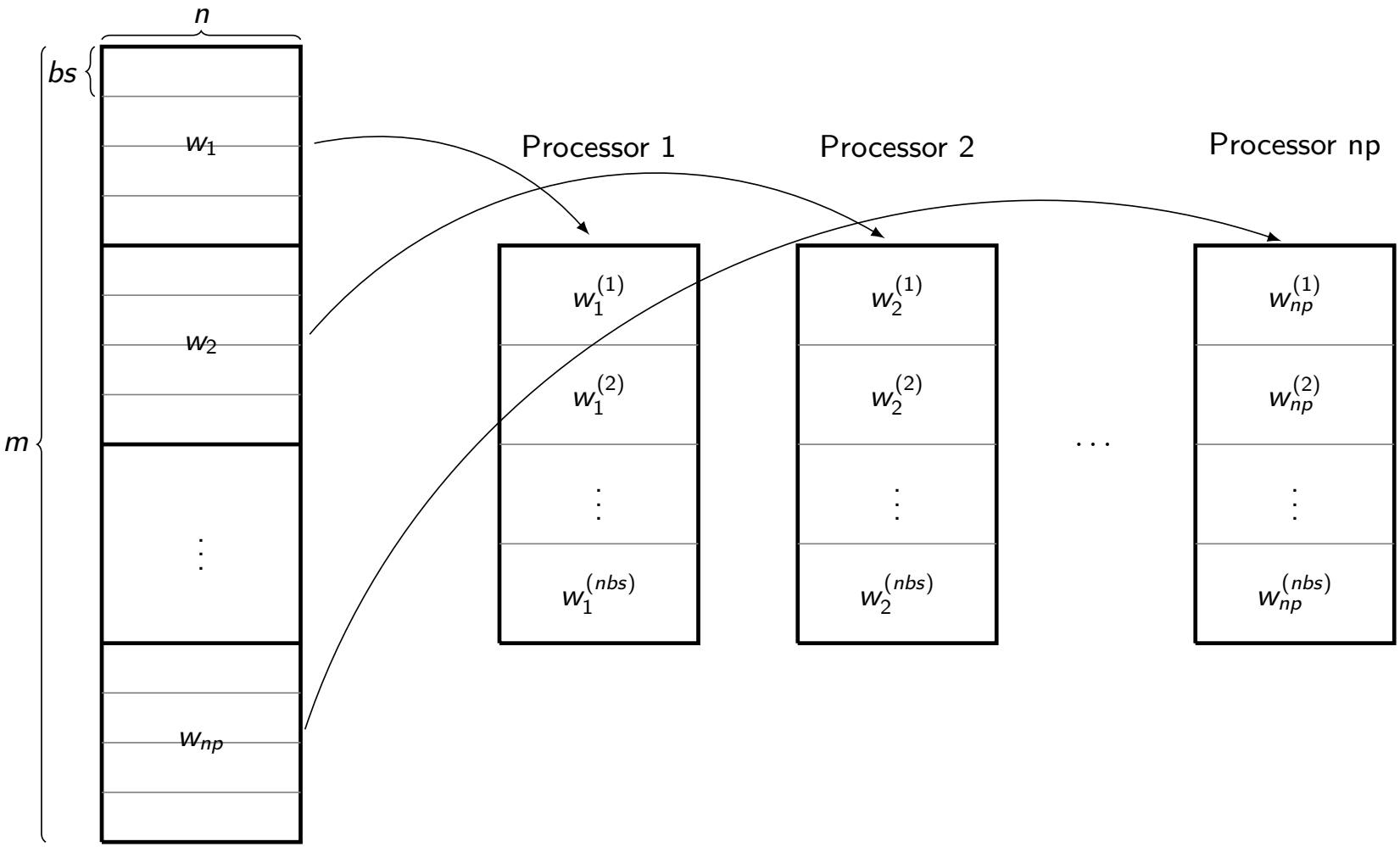
With scattering and gathering of $\mathcal{W}$ random $10^7 \times 300$ matrix



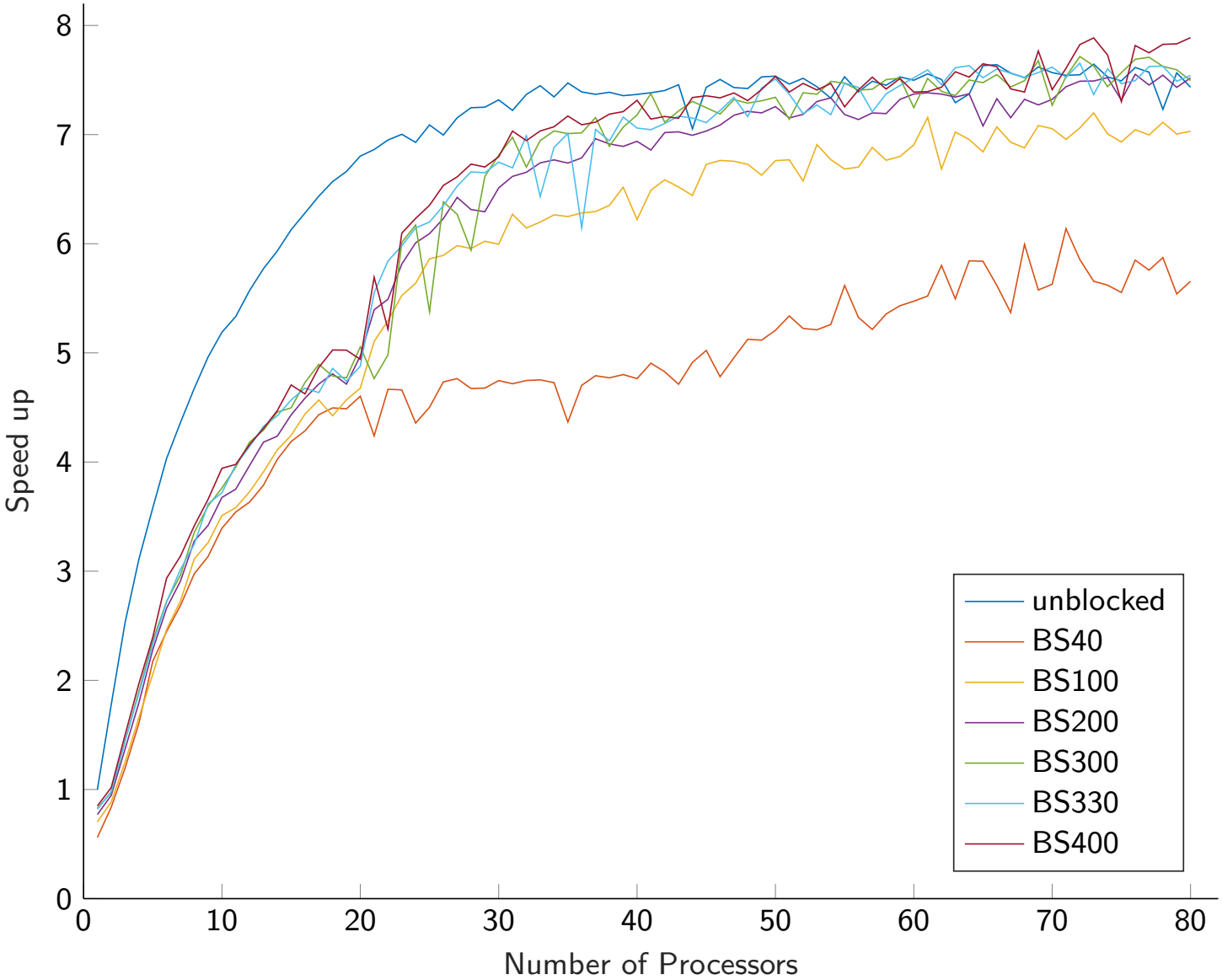
MPI I functions

- I functions provide non bocking communication
- communication is not blocking processor
- communication buffer should not be used while communication
- MPI_Scatterv ⇒ MPI_Iscatterv
- MPI_Gatherv ⇒ MPI_Igatherv

Partitioning for non-blocking communication



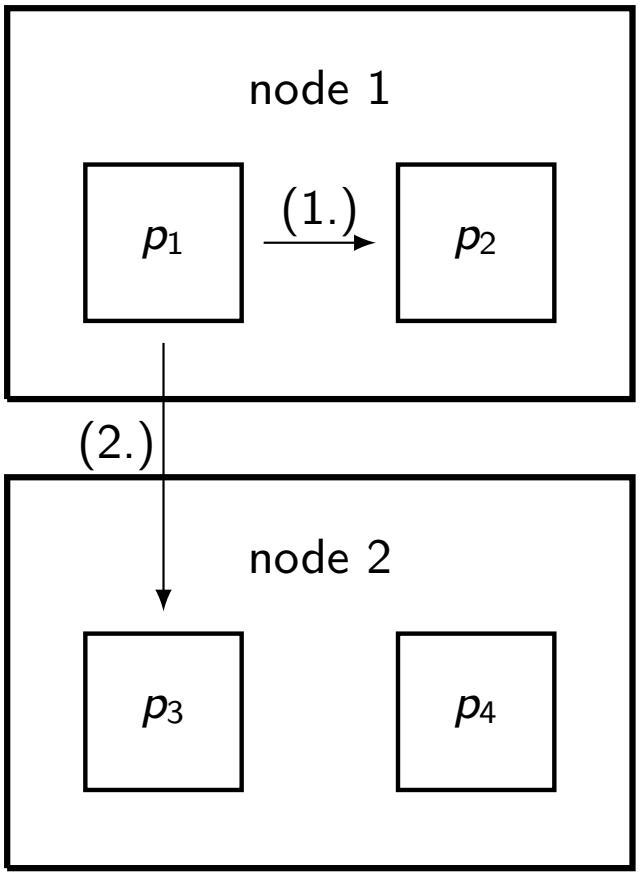
Different block sizes random $10^7 \times 300$ matrix



Increasing waiting time

- We measured the waiting time
- Waiting time increased from 0.0062 to 0.0263 (magnitude of 10)
- Possible reason: **too many communications**
- Possible solution: **reduce** MPI communications

MPI communication



1. **Copy** operation in memory
2. **Sending** operation via network

MPI & OpenMP

MPI

- Process Based
- Communication on one node is copying

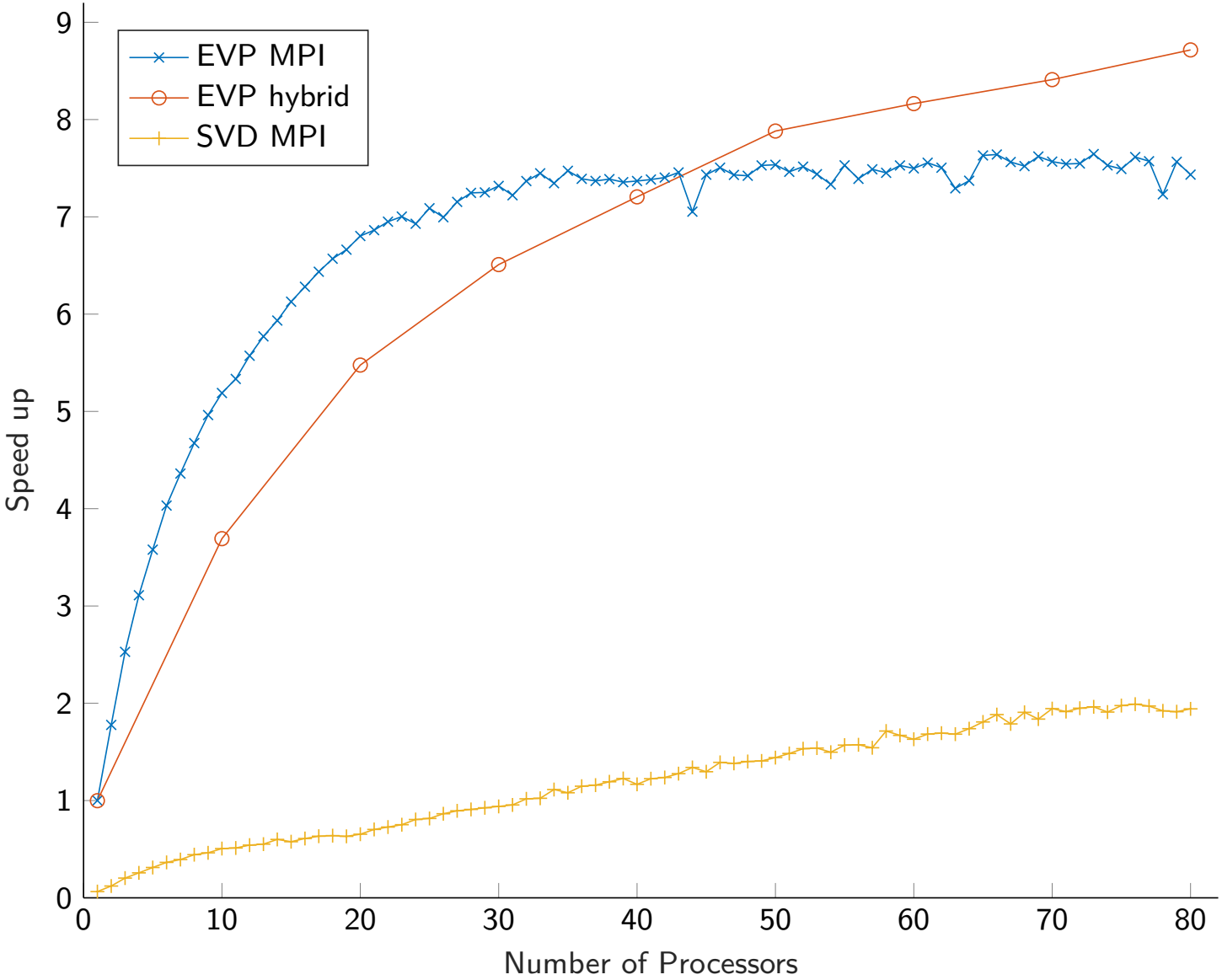
OpenMP

- Thread based
- shared memory

Hybrid implementation

- MPI for communication between nodes and processors
- OpenMP parallelization on nodes

Hybrid implementation random $10^7 \times 300$ matrix

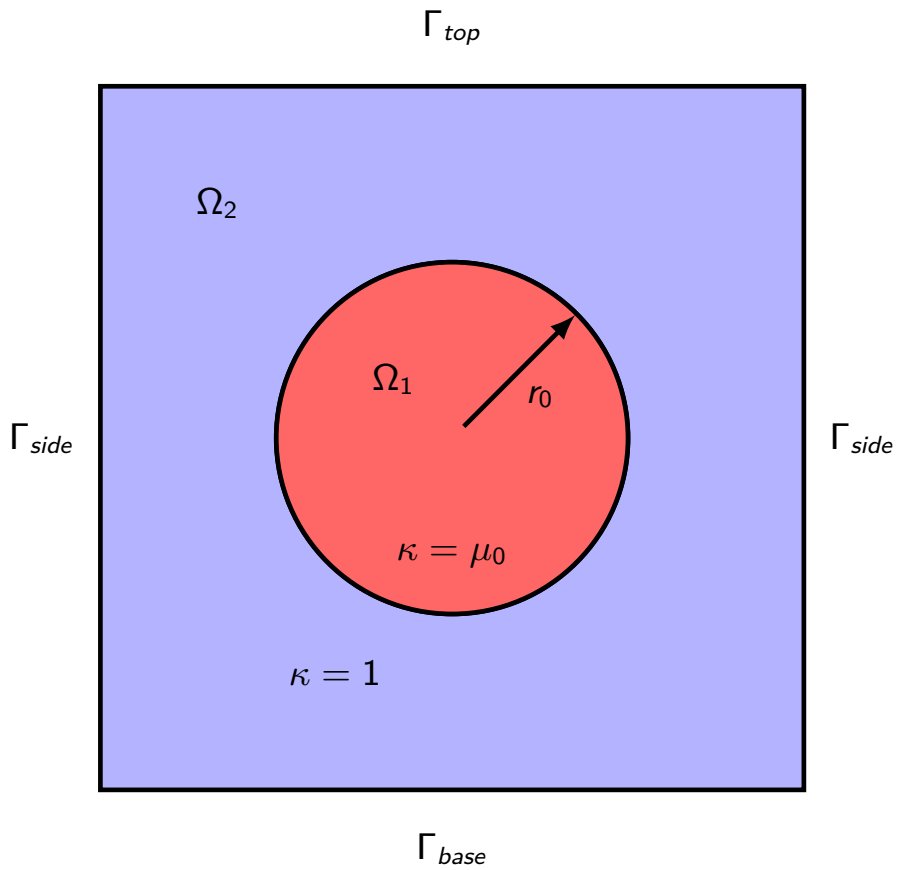


EZyRB: Easy Reduced Basis method

- Open source on `GitHub`
- Data driven reduced order modeling framework in python
- POD with interpolation (PODI)

Example Heat Conduction

$$\begin{cases} -\nabla \cdot (\kappa(x,\mu)\nabla u(x,\mu)) = 0 & \text{in } \Omega \\ u(x,\mu) = 0 & \text{on } \Gamma_{top} \\ \kappa(\mu)\nabla u(x,\mu) \cdot \mathbf{n} = 0 & \text{on } \Gamma_{side} \\ \kappa(\mu)\nabla u(x,\mu) \cdot \mathbf{n} = \mu_1 & \text{on } \Gamma_{base} \end{cases}$$



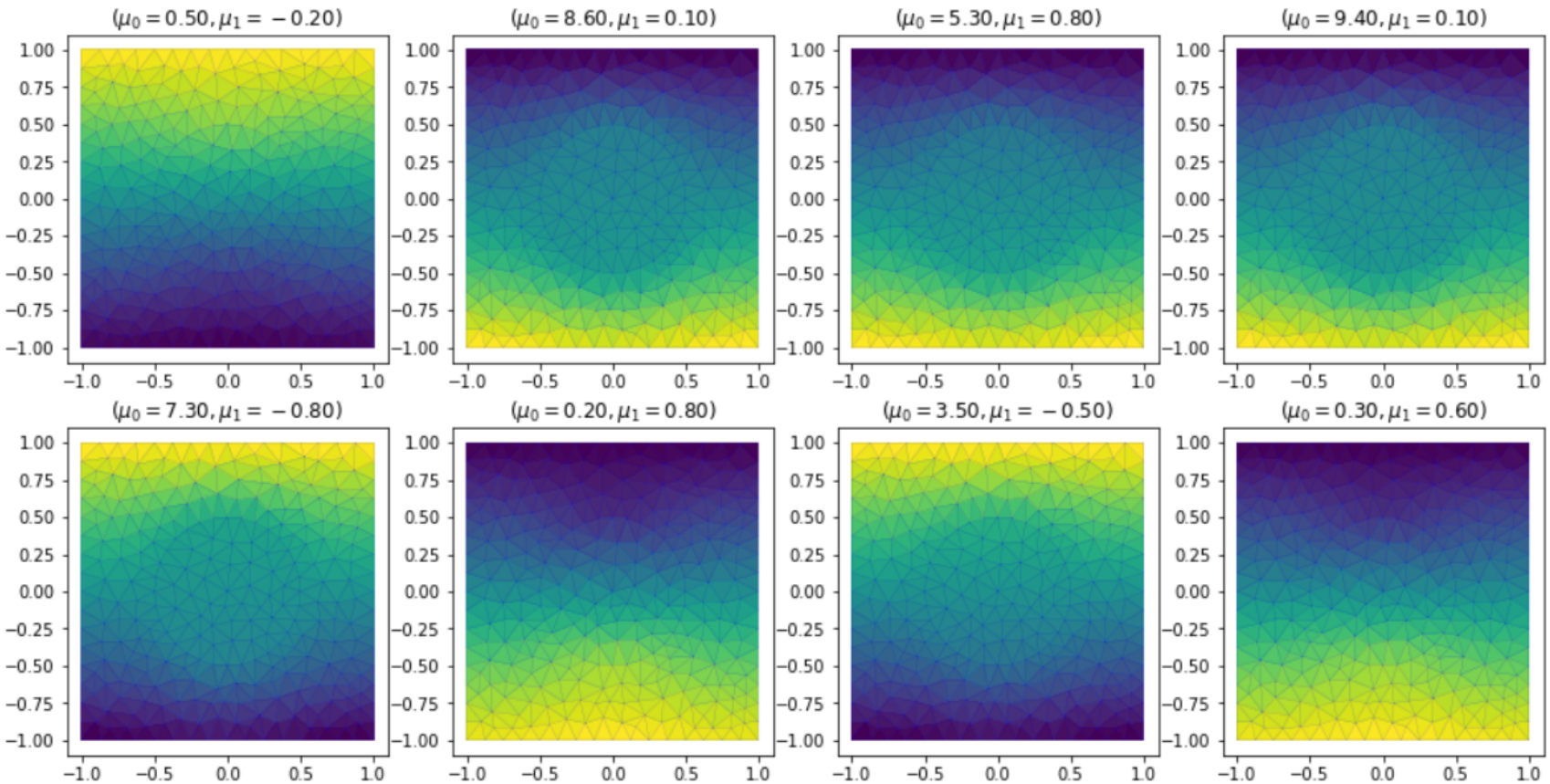
find $u(\mu) \in \mathbb{X}$:

$a(u(\mu), v; \mu) = f(v; \mu) \quad \forall v \in \mathbb{X}.$

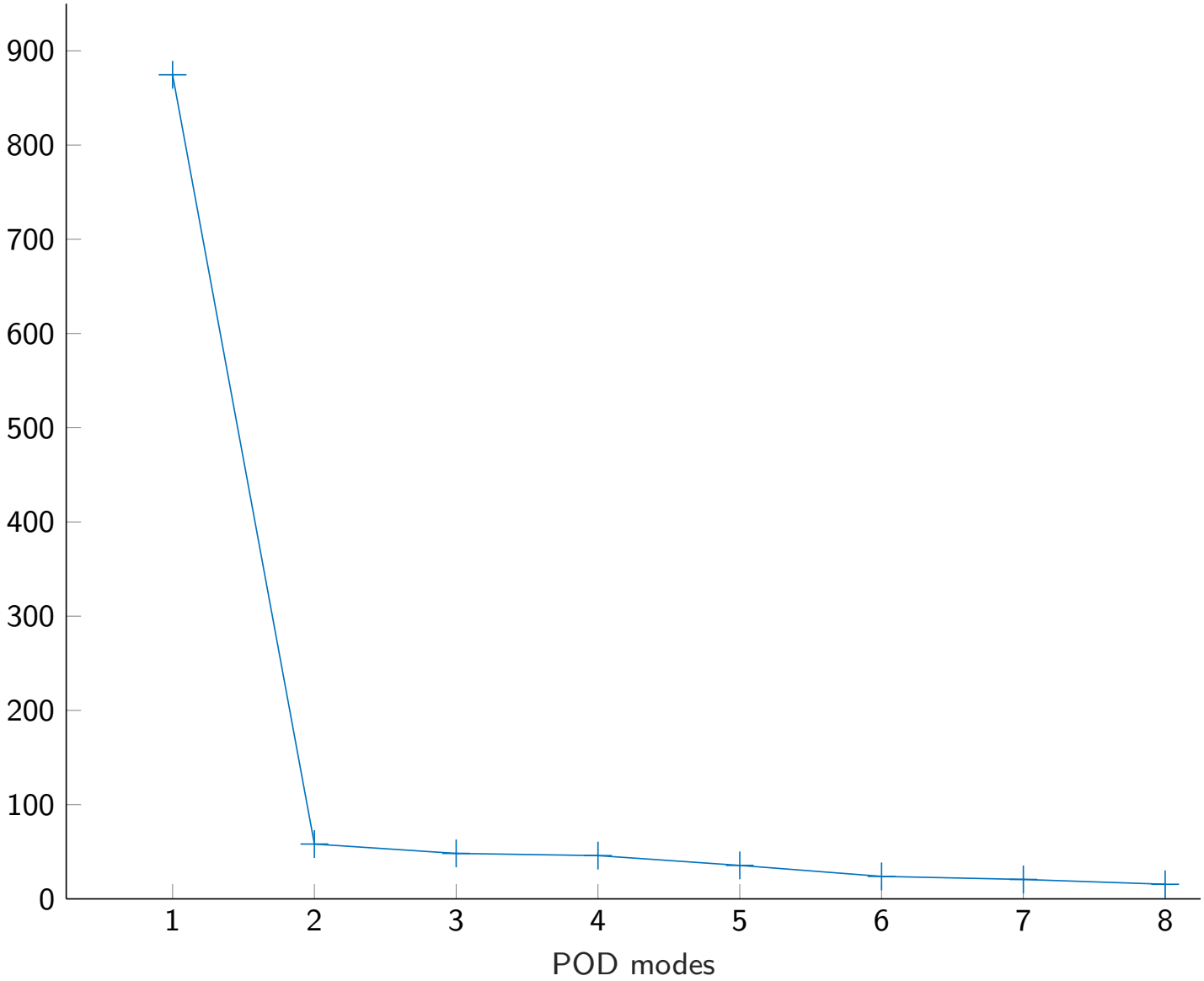
$$a(u, v; \mu) = \int_\Omega \kappa(x,\mu)\nabla u(x,\mu)\nabla v(x) dx, \qquad f(v;\mu) = \mu_1 \int_{\Gamma_{base}} v \; ds$$

Snapshots

$$\Xi_{train} = \left\{ \begin{pmatrix} 0.5 \\ -0.2 \end{pmatrix}, \begin{pmatrix} 8.6 \\ 0.1 \end{pmatrix}, \begin{pmatrix} 5.3 \\ 0.8 \end{pmatrix}, \begin{pmatrix} 9.4 \\ 0.1 \end{pmatrix}, \begin{pmatrix} 7.3 \\ -0.8 \end{pmatrix}, \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix}, \begin{pmatrix} 3.5 \\ -0.5 \end{pmatrix}, \begin{pmatrix} 0.3 \\ 0.6 \end{pmatrix} \right\}.$$



Singular values



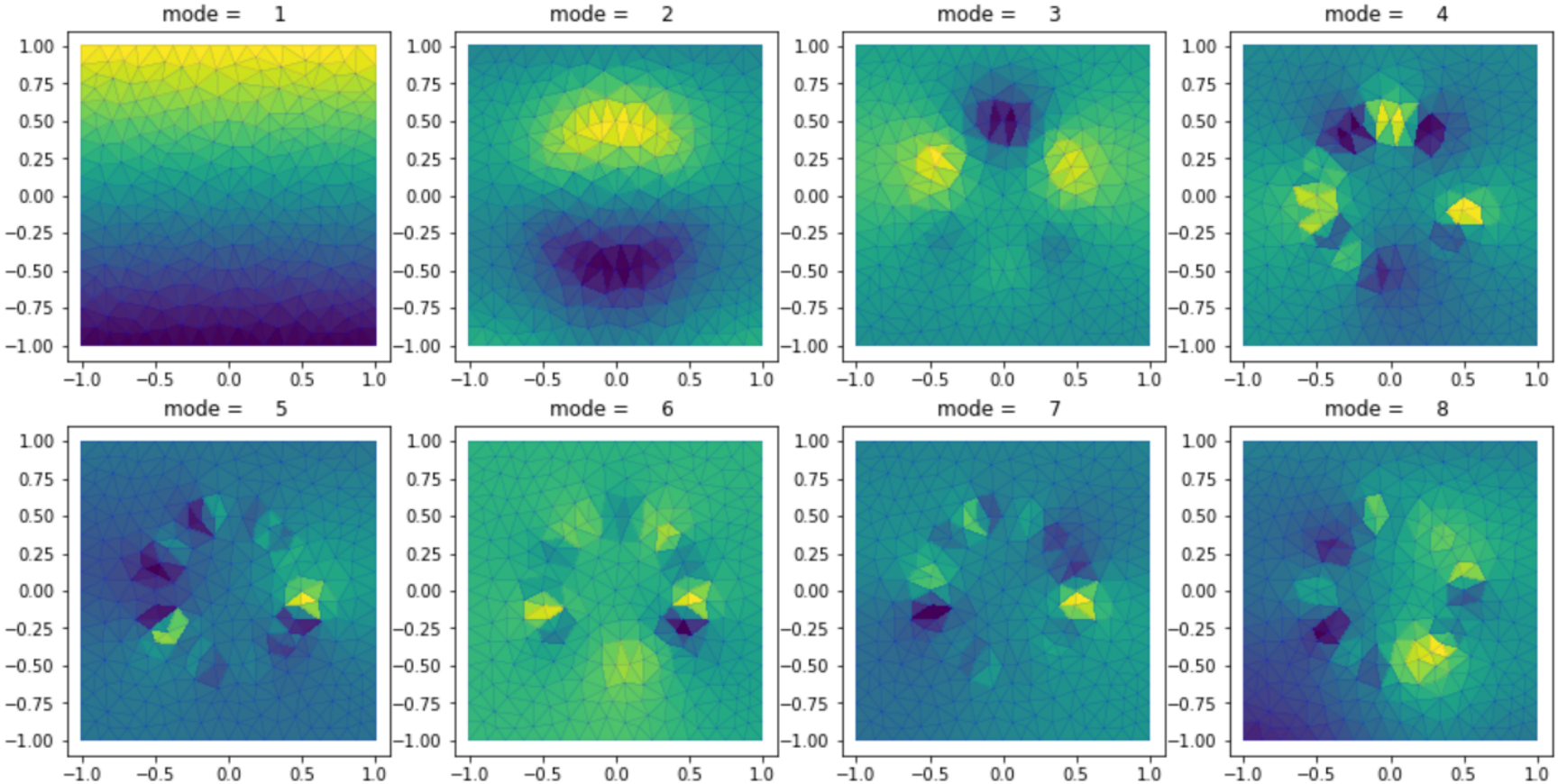
Conclusion

- We presented two algorithms to compute the POD in parallel
- EVP perfomes better than SVD due to its limited number of operations
- Distributing the snapshot matrix is a bottle neck
- We presented hybrid parallel strategy for the POD

Outlook

- The performance of the computation of the **correlation matrix** could be improved by using graphics processing units (**GPU**).

PODmodes


mode = 1
mode = 2
mode = 3
mode = 4
mode = 5
mode = 6
mode = 7
mode = 8
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0