Optimized Domain Desomposition Methodes with HPDDM Library

Ryadh Haferssas [1] **Pierre Jolivet** [2] **Frédéric Nataf** [1]

[2]IRIT-CNRS

[1]LJLL-CNRS-INRIA(Alpines)

December 9, 2016

Outline

Why Domain Decomposition Methods ?

Find $u \in V_h$ such that : $\forall v \in V_h$

$$a_\Omega(u, v) = \ell(v)$$

Why Domain Decomposition Methods ?

$$\text{Find } u \in V_h \text{ such that : } \forall v \in V_h \quad a_\Omega(u,v) = \ell(v) \Longrightarrow \boxed{\boldsymbol{A}u = F} \in \mathbb{R}^n.$$

Why Domain Decomposition Methods ?

$$\text{Find } u \in V_h \text{ such that : } \forall v \in V_h \quad a_\Omega(u,v) = \ell(v) \Longrightarrow \boxed{\boldsymbol{A}u = F} \in \mathbb{R}^n.$$

- Darcy $\quad a_\Omega(u,v) = \int_\Omega \kappa \underline{\nabla}\, u \cdot \underline{\nabla}\, v \quad dx$
- Elasticty $\quad a_\Omega(u,v) = \int_\Omega \underline{\underline{\underline{\underline{C}}}} : \underline{\underline{\varepsilon}}(u) : \underline{\underline{\varepsilon}}(v) \quad dx$

Why Domain Decomposition Methods ?

$$\text{Find } u \in V_h \text{ such that : } \forall v \in V_h \quad a_\Omega(u,v) = \ell(v) \Longrightarrow \boxed{\boldsymbol{A}u = F} \in \mathbb{R}^n.$$

- Darcy $\quad a_\Omega(u,v) = \int_\Omega \kappa \underline{\nabla}\, u \cdot \underline{\nabla}\, v \quad dx$
- Elasticty $\quad a_\Omega(u,v) = \int_\Omega \underline{\underline{\underline{\underline{C}}}} : \underline{\underline{\varepsilon}}(u) : \underline{\underline{\varepsilon}}(v) \quad dx$

Purpose

- Solve $\boldsymbol{A}U = F$. Achieve robustness with regard to the irregularity of the coefficient distribution
- Achieve a strong and weak scalability when solving with thousands of processors

Why Domain Decomposition Methods ?

$$\text{Find } u \in V_h \text{ such that : } \forall v \in V_h \quad a_\Omega(u,v) = \ell(v) \Longrightarrow \boxed{\boldsymbol{A}u = F} \in \mathbb{R}^n.$$

- Darcy $\quad a_\Omega(u,v) = \int_\Omega \kappa \underline{\nabla}\, u \cdot \underline{\nabla}\, v \quad dx$
- Elasticty $\quad a_\Omega(u,v) = \int_\Omega \underline{\underline{C}} : \underline{\underline{\varepsilon}}(u) : \underline{\underline{\varepsilon}}(v) \quad dx$

Purpose

- Solve $\boldsymbol{A}U = F$. Achieve robustness with regard to the irregularity of the coefficient distribution
- Achieve a strong and weak scalability when solving with thousands of processors

Tools

- A good mathematical foundation theory & HPDDM Library: a parallel framework

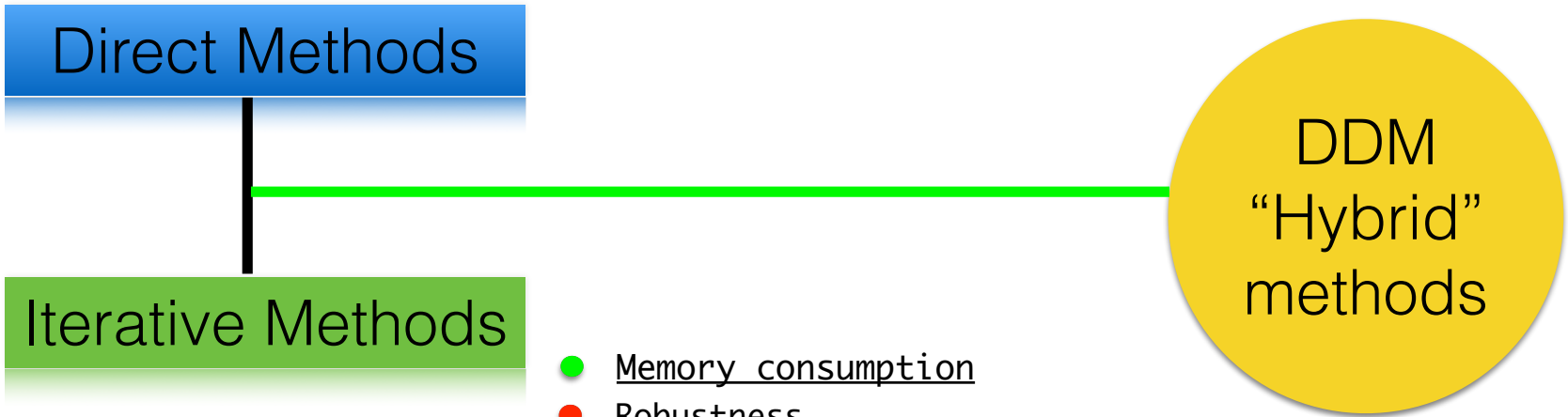
Why Domain Decomposition Methods ?

How can we solve a large sparse system $\boldsymbol{A}u = F \in \mathbb{R}^n$?

Why Domain Decomposition Methods ?

How can we solve a large sparse system $\boldsymbol{A}u = F \in \mathbb{R}^n$?

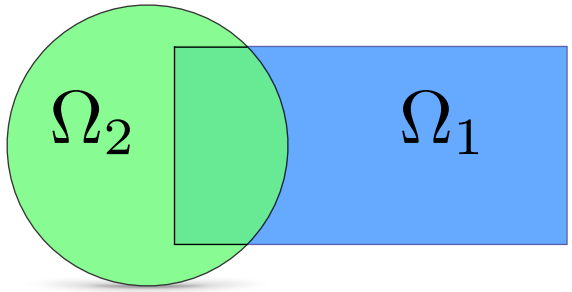
- Memory consumption
- Robustness
- Parallelizable



- Memory consumption
- Robustness
- Parallelizable

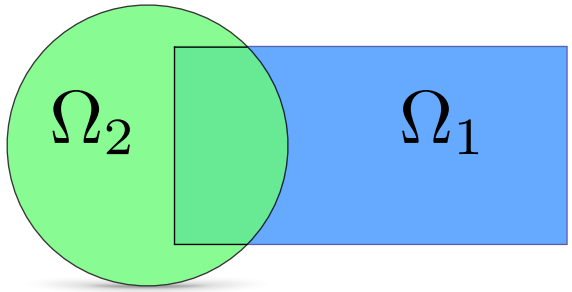
Original Schwarz DDM (H.A. Schwarz, 1870)

$$\begin{cases} -\Delta u = 0 \text{ in } \Omega \\ u = g \text{ on } \partial\Omega \end{cases} \Longrightarrow \boldsymbol{A}u = F \in \mathbb{R}^n.$$



Original Schwarz DDM (H.A. Schwarz, 1870)

$$\begin{cases} -\Delta u = 0 \text{ in } \Omega \\ \quad u = g \text{ on } \partial\Omega \end{cases} \Longrightarrow \boldsymbol{A}u = F \in \mathbb{R}^n.$$

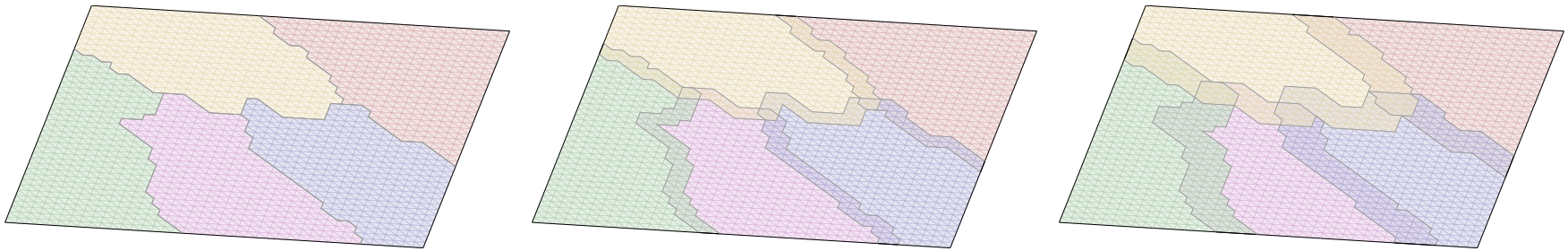


Given (u_1^n, u_2^n), find (u_1^{n+1}, u_2^{n+1}) such that

$$\left| \begin{array}{rr} -\Delta u_1^{n+1} = 0 & \text{in } \Omega_1 \\ u_1^{n+1} = g & \text{on } \partial\Omega_1 \cap \partial\Omega \\ u_1^{n+1} = u_2^n & \text{on } \Omega \backslash \Omega_1 \end{array} \right| \quad \left| \begin{array}{rr} -\Delta u_2^{n+1} = 0 & \text{in } \Omega_2 \\ u_2^{n+1} = g & \text{on } \partial\Omega_2 \cap \partial\Omega \\ u_2^{n+1} = u_1^{n+1} & \text{on } \Omega \backslash \Omega_2 \end{array} \right|$$

[Martin J. Gander and Wanner 2014]

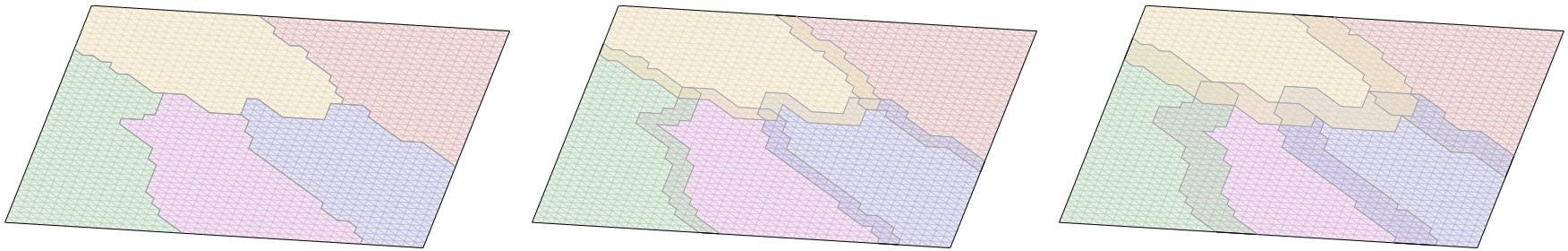
Schwarz DDM Notations & definitions



From finite elements spaces :

- Global $V_h = \text{span}\{\phi_k : k = 1, \ldots, n\} \rightarrow \mathcal{N}$
- Local $V_h(\Omega_i) = \text{span}\left\{\phi_{k|\Omega_i} : \text{mes}(\text{supp}(\phi_k)) \cap \overline{\Omega}_i \neq 0\right\} \rightarrow \mathcal{N}_i$

Schwarz DDM Notations & definitions



From finite elements spaces :

- Global $V_h = \text{span}\{\phi_k : k = 1, \ldots, n\} \rightarrow \mathcal{N}$
- Local $V_h(\Omega_i) = \text{span}\left\{\phi_{k|\Omega_i} : \text{mes}(\text{supp}(\phi_k)) \cap \overline{\Omega}_i \neq 0\right\} \rightarrow \mathcal{N}_i$

$$\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$$

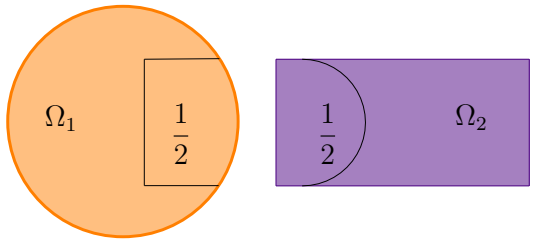
Interpolation Operators :

- Global to Local $R_i : \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$ (Restriction)
- Local to Global $R_i^T : \ R_i^\intercal : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}}$ (Prolongation)

Duplicated unknowns are coupled via a partition of unity:

$$D_i : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$$

$$\sum_{i=1}^{N} R_i^\intercal \, D_i \, R_i = Id.$$



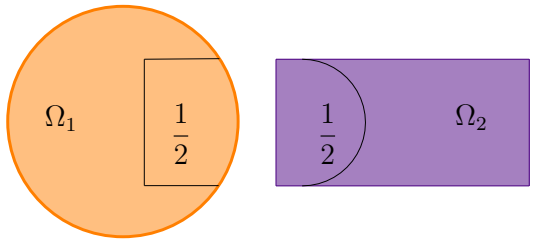
Interpolation Operators :

- Global to Local $R_i : \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$ (Restriction)
- Local to Global $R_i^T : \ R_i^\intercal : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}}$ (Prolongation)

Duplicated unknowns are coupled via a partition of unity:

$$D_i : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$$

$$\sum_{i=1}^{N} R_i^\intercal \, D_i \, R_i = Id.$$



Then

$$u^{n+1} = \sum_{i=1}^{N} R_i^\intercal \, D_i u_i^{n+1}$$

To solve $\boxed{Au = F}$ Schwarz methods can be viewed as preconditioners for a fixed point algorithm:

$$u^{n+1} = u^n + M^{-1}(f - Au^n).$$

Where the Schwarz preconditioner can be

Classical :

- $M^{-1}_{\text{ASM},1} := \sum_{i=1}^{N} R_i^\intercal {A_i}^{-1} R_i$ with $A_i = R_i A R_i^\intercal$ [Toselli and Widlund 2005]

To solve $\boxed{Au = F}$ Schwarz methods can be viewed as preconditioners for a fixed point algorithm:

$$u^{n+1} = u^n + M^{-1}(f - Au^n).$$

Where the Schwarz preconditioner can be

Classical :

- $M^{-1}_{\text{ASM},1} := \sum_{i=1}^{N} R_i^\intercal {A_i}^{-1} R_i$ with $A_i = R_i A R_i^\intercal$ [Toselli and Widlund 2005]
- $M^{-1}_{RAS,1} := \sum_{i=1}^{N} R_i^\intercal D_i A_i^{-1} R_i$ [Cai and Sarkis 1999]

P-L. Lions algorithm

$$
\begin{aligned}
-\Delta u_1^{n+1} &= f && \text{in } \Omega_1 \\
u_1^{n+1} &= 0 && \text{on } \partial\Omega_1 \cap \partial\Omega \\
\left(\frac{\partial}{\partial \mathbf{n}_1} + \alpha\right) u_1^{n+1} &= \left(\frac{\partial}{\partial \mathbf{n}_1} + \alpha\right) u_2^{n} && \text{on } \partial\Omega_1 \cap \overline{\Omega_2}\,,
\end{aligned}
$$

and

$$
\begin{aligned}
-\Delta u_1^{n+1} &= f && \text{in } \Omega_2 \\
u_1^{n+1} &= 0 && \text{on } \partial\Omega_2 \cap \partial\Omega \\
\left(\frac{\partial}{\partial \mathbf{n}_2} + \alpha\right) u_2^{n+1} &= \left(\frac{\partial}{\partial \mathbf{n}_2} + \alpha\right) u_1^{n} && \text{on } \partial\Omega_2 \cap \overline{\Omega_1}
\end{aligned}
$$

P-L. Lions algorithm

$$
\begin{aligned}
-\Delta u_1^{n+1} &= f && \text{in } \Omega_1 \\
u_1^{n+1} &= 0 && \text{on } \partial\Omega_1 \cap \partial\Omega \\
\left(\frac{\partial}{\partial \mathbf{n}_1} + \alpha\right) u_1^{n+1} &= \left(\frac{\partial}{\partial \mathbf{n}_1} + \alpha\right) u_2^{n} && \text{on } \partial\Omega_1 \cap \overline{\Omega_2},
\end{aligned}
$$

and

$$
\begin{aligned}
-\Delta u_1^{n+1} &= f && \text{in } \Omega_2 \\
u_1^{n+1} &= 0 && \text{on } \partial\Omega_2 \cap \partial\Omega \\
\left(\frac{\partial}{\partial \mathbf{n}_2} + \alpha\right) u_2^{n+1} &= \left(\frac{\partial}{\partial \mathbf{n}_2} + \alpha\right) u_1^{n} && \text{on } \partial\Omega_2 \cap \overline{\Omega_1}
\end{aligned}
$$

Optimized :

- $M_{\text{ORAS},1}^{-1} := \sum_{i=1}^{N} R_i^\intercal D_i {B_i}^{-1} R_i$ [St-Cyr, M. J. Gander, and Thomas 2007]

P-L. Lions algorithm

$$\begin{aligned} -\Delta u_1^{n+1} &= f && \text{in } \Omega_1 \\ u_1^{n+1} &= 0 && \text{on } \partial\Omega_1 \cap \partial\Omega \\ \left(\frac{\partial}{\partial \mathbf{n}_1} + \alpha\right) u_1^{n+1} &= \left(\frac{\partial}{\partial \mathbf{n}_1} + \alpha\right) u_2^{n} && \text{on } \partial\Omega_1 \cap \overline{\Omega_2}\,, \end{aligned}$$

and

$$\begin{aligned} -\Delta u_1^{n+1} &= f && \text{in } \Omega_2 \\ u_1^{n+1} &= 0 && \text{on } \partial\Omega_2 \cap \partial\Omega \\ \left(\frac{\partial}{\partial \mathbf{n}_2} + \alpha\right) u_2^{n+1} &= \left(\frac{\partial}{\partial \mathbf{n}_2} + \alpha\right) u_1^{n} && \text{on } \partial\Omega_2 \cap \overline{\Omega_1} \end{aligned}$$

Optimized :

- $M_{\text{ORAS},1}^{-1} := \sum_{i=1}^{N} R_i^\intercal D_i B_i{}^{-1} R_i$ [St-Cyr, M. J. Gander, and Thomas 2007]
- $M_{\text{SORAS},1}^{-1} := \sum_{i=1}^{N} R_i^\intercal D_i B_i{}^{-1} D_i R_i$ [Haferssas, Jolivet, and Nataf 2015]

One-Level methods are not scalable

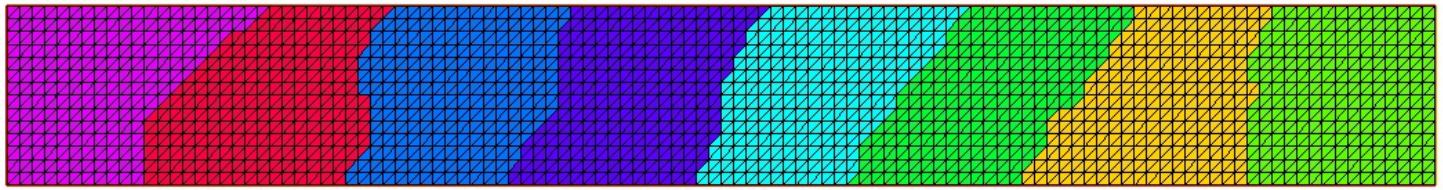


Figure: 2D domain partitioned with METIS

One-Level methods are not scalable

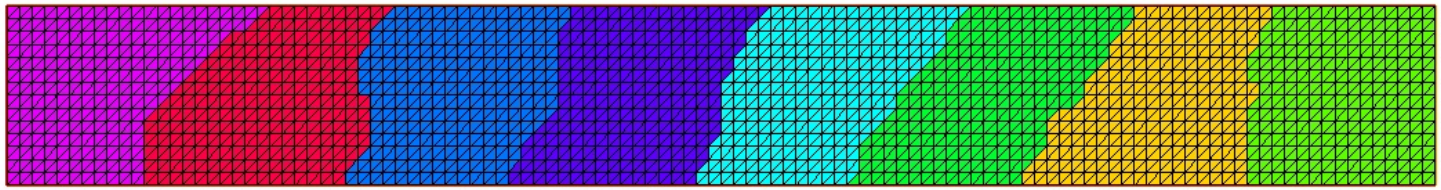


Figure: 2D domain partitioned with METIS

		AS	RAS	ORAS	SORAS
# DOFs	# subdom	# iterations	# iterations	# iterations	# iterations
66703	16	140	140	96	57
130433	32	245	216	146	109
266812	64	383	312	245	203
541838	128	566	460	480	398

Table: 2D Elasticity: number of GMRES iterations for compressible case with $E = 10^7$ and $\nu = 0.3$

Two Level Domain Decomposition Methods

Given

$$V_H := \textbf{span}(Z)$$

$$\Downarrow$$

$$R_H = Z^\intercal : \ \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#Z}$$

$$\underbrace{E := R_H A R_H^\intercal}_{\text{coarse problem, } E \text{ much smaller than } A}$$

Enrich the one level preconditioner with Z (ie $Q = R_H^\intercal E^{-1} R_H$)

Two Level Domain Decomposition Methods

Given

$$V_H := \textbf{span}(Z)$$

$$\Downarrow$$

$$R_H = Z^\intercal : \ \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#Z}$$

$$\underbrace{E := R_H A R_H^\intercal}_{\text{coarse problem, } E \text{ much smaller than } A}$$

Enrich the one level preconditioner with Z (ie $Q = R_H^\intercal E^{-1} R_H$) :

- $M_{2,AD}^{-1} := Q + M^{-1}$

Two Level Domain Decomposition Methods

Given

$$V_H := \textbf{span}(Z)$$

$$\Downarrow$$

$$R_H = Z^\intercal : \ \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#Z}$$

$$\underbrace{E := R_H A R_H^\intercal}_{\text{coarse problem, } E \text{ much smaller than } A}$$

Enrich the one level preconditioner with Z (ie $Q = R_H^\intercal E^{-1} R_H$) :

- $M_{2,AD}^{-1} := Q + M^{-1}$
- $M_{2,A-DEF_1}^{-1} := M^{-1}(I - AQ) + Q$

Two Level Domain Decomposition Methods

Given

$$V_H := \textbf{span}(Z)$$

$$\Downarrow$$

$$R_H = Z^\intercal : \ \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#Z}$$

$$\underbrace{E := R_H A R_H^\intercal}_{\text{coarse problem, } E \text{ much smaller than } A}$$

Enrich the one level preconditioner with Z (ie $Q = R_H^\intercal E^{-1} R_H$) :

- $M_{2,AD}^{-1} := Q + M^{-1}$
- $M_{2,A-DEF_1}^{-1} := M^{-1}(I - AQ) + Q$
- $M_{2,A-DEF_2}^{-1} := (I - QA)M^{-1} + Q$ [Tang, Nabben, Vuik, and Erlangga 2009]

Two Level Domain Decomposition Methods

Given

$$V_H := \textbf{span}(Z)$$

$$\Downarrow$$

$$R_H = Z^\intercal : \ \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#Z}$$

$$\underbrace{E := R_H A R_H^\intercal}_{\text{coarse problem, } E \text{ much smaller than } A}$$

Enrich the one level preconditioner with Z (ie $Q = R_H^\intercal E^{-1} R_H$) :

- $M_{2,AD}^{-1} := Q + M^{-1}$
- $M_{2,A-DEF_1}^{-1} := M^{-1}(I - AQ) + Q$
- $M_{2,A-DEF_2}^{-1} := (I - QA)M^{-1} + Q$ [Tang, Nabben, Vuik, and Erlangga 2009]
- $M_{2,BNN}^{-1} := (I - QA)M^{-1}(I - AQ) + Q$ [Mandel 1992]

Two Level Domain Decomposition Methods

Given

$$V_H := \textbf{span}(Z)$$

$$\Downarrow$$

$$R_H = Z^\intercal : \ \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#Z}$$

$$\underbrace{E := R_H A R_H^\intercal}_{\text{coarse problem, } E \text{ much smaller than } A}$$

Enrich the one level preconditioner with Z (ie $Q = R_H^\intercal E^{-1} R_H$) :

- $M_{2,AD}^{-1} := Q + M^{-1}$
- $M_{2,A-DEF_1}^{-1} := M^{-1}(I - AQ) + Q$
- $M_{2,A-DEF_2}^{-1} := (I - QA)M^{-1} + Q$ [Tang, Nabben, Vuik, and Erlangga 2009]
- $M_{2,BNN}^{-1} := (I - QA)M^{-1}(I - AQ) + Q$ [Mandel 1992]

what does the space V_H contain?

GenEO I approach to build V_H

Find the eigenpairs $(\lambda_{i,k}, U_{i,k})$

$$A_i^N U_{i,k} = \lambda_{i,k} D_i A_i D_i U_{i,k}$$

Solved by ARPACK (Concurrently)

With :

- A_i^N local matrix, resulting from the local bilinear form, with Neumann boundary condition on the interfaces.
- $R_{i,0} : \Omega_i \longrightarrow \Omega_i^\circ = \bigcup_{j \neq i} (\Omega_i \bigcap \Omega_j)$ a restriction operator.

GenEO I approach to build V_H

Find the eigenpairs $(\lambda_{i,k}, U_{i,k})$

$$A_i^N U_{i,k} = \lambda_{i,k} D_i A_i D_i U_{i,k}$$

Solved by ARPACK (Concurrently)

With :

- A_i^N local matrix, resulting from the local bilinear form, with Neumann boundary condition on the interfaces.
- $R_{i,0} : \Omega_i \longrightarrow \Omega_i^\circ = \bigcup_{j \neq i} (\Omega_i \bigcap \Omega_j)$ a restriction operator.

Choose a number of eigenmodes τ_i for each subdomain then define

$$W_i = [D_i U_{i,1}, D_i U_{i,2}, \ldots, D_i U_{i,\tau_i}]$$

GenEO I approach to build V_H

Find the eigenpairs $(\lambda_{i,k}, U_{i,k})$

$$A_i^N U_{i,k} = \lambda_{i,k} D_i A_i D_i U_{i,k}$$

Solved by ARPACK (Concurrently)

With :

- A_i^N local matrix, resulting from the local bilinear form, with Neumann boundary condition on the interfaces.
- $R_{i,0} : \Omega_i \longrightarrow \Omega_i^\circ = \bigcup_{j \neq i} (\Omega_i \bigcap \Omega_j)$ a restriction operator.

Choose a number of eigenmodes τ_i for each subdomain then define

$$W_i = [D_i U_{i,1}, D_i U_{i,2}, \ldots, D_i U_{i,\tau_i}]$$

Finally

$$Z = [W_1, W_2, \ldots, W_N]$$

Theoretical result for GenEO I

Theorem (Spillane, Dolean, Hauret, Nataf, Pechstein, Scheichl)

If for all j: $0 < \lambda_{j,m_{j+1}} < \infty$*:*

$$\kappa(M_{AS,2}^{-1}A) \leqslant (1+k_0)\Big[2+k_0\,(2k_0+1)\,\Big(1+\tau\Big)\Big]$$

where :

- k_0 the maximum multiplicity of the interaction between subdomains.
- Parameter τ can be chosen arbitrarily small at the expense of a large coarse space.

[Spillane et al. 2014]

ORAS: Optimized RAS

Optimized RAS :

- P.L. Lions algorithm at the continuous level (partial differential equation)

ORAS: Optimized RAS

Optimized RAS :

- P.L. Lions algorithm at the continuous level (partial differential equation)
- Algebraic formulation for overlapping subdomains

ORAS: Optimized RAS

Optimized RAS :

- P.L. Lions algorithm at the continuous level (partial differential equation)
- Algebraic formulation for overlapping subdomains
  $\Rightarrow$ Let B_i be the matrix of the Robin subproblem in each subdomain $1 \leqslant i \leqslant N$
  define $M_{ORAS}^{-1} := \sum_{i=1}^{N} R_i^T D_i B_i^{-1} R_i$,
  [St-Cyr, M. J. Gander, and Thomas 2007]

ORAS: Optimized RAS

Optimized RAS :

- P.L. Lions algorithm at the continuous level (partial differential equation)
- Algebraic formulation for overlapping subdomains
  $\Rightarrow$ Let B_i be the matrix of the Robin subproblem in each subdomain $1 \leqslant i \leqslant N$
  define $M_{ORAS}^{-1} := \sum_{i=1}^{N} R_i^T D_i B_i^{-1} R_i$,
  [St-Cyr, M. J. Gander, and Thomas 2007]
- Symmetric variant $M_{\text{SORAS},1}^{-1} := \sum_{i=1}^{N} R_i^\intercal D_i {B_i}^{-1} D_i R_i$

ORAS: Optimized RAS

Optimized RAS :

- P.L. Lions algorithm at the continuous level (partial differential equation)
- Algebraic formulation for overlapping subdomains
  $\Rightarrow$ Let B_i be the matrix of the Robin subproblem in each subdomain $1 \leqslant i \leqslant N$
  define $M_{ORAS}^{-1} := \sum_{i=1}^{N} R_i^T D_i B_i^{-1} R_i$,
  [St-Cyr, M. J. Gander, and Thomas 2007]
- Symmetric variant $M_{\text{SORAS},1}^{-1} := \sum_{i=1}^{N} R_i^\intercal D_i {B_i}^{-1} D_i R_i$
- Adaptive Coarse space with prescribed targeted convergence rate
  $\Rightarrow$

ORAS: Optimized RAS

Optimized RAS :

- P.L. Lions algorithm at the continuous level (partial differential equation)
- Algebraic formulation for overlapping subdomains
  $\Rightarrow$ Let B_i be the matrix of the Robin subproblem in each subdomain $1 \leqslant i \leqslant N$
  define $M_{ORAS}^{-1} := \sum_{i=1}^{N} R_i^T D_i B_i^{-1} R_i$,
  [St-Cyr, M. J. Gander, and Thomas 2007]
- Symmetric variant $M_{\text{SORAS},1}^{-1} := \sum_{i=1}^{N} R_i^\intercal D_i {B_i}^{-1} D_i R_i$
- Adaptive Coarse space with prescribed targeted convergence rate
  $\Rightarrow$???

SORAS-GenEO II approach to build V_H

Find the eigenpairs $(\lambda_{i,k}, U_{i,k})$

$$A_i^N U_{i,k} = \lambda_{i,k} B_i U_{i,k}$$

SORAS-GenEO II approach to build V_H

Find the eigenpairs $(\lambda_{i,k}, U_{i,k})$

$$A_i^N U_{i,k} = \lambda_{i,k} B_i U_{i,k}$$

Solved by ARPACK (Concurrently)

Find the eigenpairs $(\mu_{i,k}, V_{i,k})$

$$B_i V_{i,k} = \mu_{i,k} D_i A_i D_i V_{i,k}$$

where :

- B_i the local matrix, resulting from the local bilinear form with Optimized interface conditions

SORAS-GenEO II approach to build V_H

Find the eigenpairs $(\lambda_{i,k}, U_{i,k})$

$$A_i^N U_{i,k} = \lambda_{i,k} B_i U_{i,k}$$

Find the eigenpairs $(\mu_{i,k}, V_{i,k})$

$$B_i V_{i,k} = \mu_{i,k} D_i A_i D_i V_{i,k}$$

Solved by ARPACK (Concurrently)

where :

- B_i the local matrix, resulting from the local bilinear form with Optimized interface conditions

Choose τ_i and γ_i for each subdomain then define

$$W_i = [D_i U_{i,1} \quad D_i U_{i,2} \quad \dots D_i U_{i,\tau_i}]$$

SORAS-GenEO II approach to build V_H

Find the eigenpairs $(\lambda_{i,k}, U_{i,k})$

$$A_i^N U_{i,k} = \lambda_{i,k} B_i U_{i,k}$$

Solved by ARPACK (Concurrently)

Find the eigenpairs $(\mu_{i,k}, V_{i,k})$

$$B_i V_{i,k} = \mu_{i,k} D_i A_i D_i V_{i,k}$$

where :

- B_i the local matrix, resulting from the local bilinear form with Optimized interface conditions

Choose τ_i and γ_i for each subdomain then define

$$W_i = [D_i U_{i,1} \quad D_i U_{i,2} \quad \dots D_i U_{i,\tau_i}] \text{ And } H_i = [D_i V_{i,1} \quad D_i V_{i,2} \dots D_i V_{i,\gamma_i}]$$

SORAS-GenEO II approach to build V_H

Find the eigenpairs $(\lambda_{i,k}, U_{i,k})$

$$A_i^N U_{i,k} = \lambda_{i,k} B_i U_{i,k}$$

Solved by ARPACK (Concurrently)

Find the eigenpairs $(\mu_{i,k}, V_{i,k})$

$$B_i V_{i,k} = \mu_{i,k} D_i A_i D_i V_{i,k}$$

where :

- B_i the local matrix, resulting from the local bilinear form with Optimized interface conditions

Choose τ_i and γ_i for each subdomain then define

$$W_i = [D_i U_{i,1} \quad D_i U_{i,2} \quad \dots D_i U_{i,\tau_i}] \text{ And } H_i = [D_i V_{i,1} \quad D_i V_{i,2} \dots D_i V_{i,\gamma_i}]$$

$$Z^\tau = [W_1 \quad W_2 \dots \quad W_N] \text{ And } Z^\gamma = [H_1 \quad H_2 \quad \dots \quad H_N]$$

SORAS-GenEO II approach to build V_H

Find the eigenpairs $(\lambda_{i,k}, U_{i,k})$

$$A_i^N U_{i,k} = \lambda_{i,k} B_i U_{i,k}$$

Solved by ARPACK (Concurrently)

Find the eigenpairs $(\mu_{i,k}, V_{i,k})$

$$B_i V_{i,k} = \mu_{i,k} D_i A_i D_i V_{i,k}$$

where :

- B_i the local matrix, resulting from the local bilinear form with Optimized interface conditions

Choose τ_i and γ_i for each subdomain then define

$$W_i = [D_i U_{i,1} \quad D_i U_{i,2} \quad \dots D_i U_{i,\tau_i}] \text{ And } H_i = [D_i V_{i,1} \quad D_i V_{i,2} \dots D_i V_{i,\gamma_i}]$$

$$Z^\tau = [W_1 \quad W_2 \dots \quad W_N] \text{ And } Z^\gamma = [H_1 \quad H_2 \quad \dots \quad H_N]$$

$$Z = [Z^\tau \quad Z^\gamma]$$

Theoretical results for SORAS-GenEO II

Theorem (H., Jolivet and Nataf, 2015)

Let γ and τ be user-defined targets. Then, the eigenvalues of the two-level SORAS-GenEO II preconditioned system satisfy the following estimate

$$\frac{1}{1+\frac{k_1}{\tau}} \leqslant \kappa(M_{SORAS,2}^{-1} A) \leqslant \max(1, k_0\,\gamma)$$

where :

- k_0 the maximum number of neighbors.
- k_1 the maximum multiplicity.
- τ and γ a user-defined thresholds.

Outline

Fields of Application

- Compressible elasticity equation

$$\int_\Omega 2\mu\, \underline{\underline{\varepsilon}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{v})dx + \int_\Omega \lambda\, \underline{\nabla} \cdot (\underline{u})\, \underline{\nabla} \cdot (\underline{v})dx = \int_\Omega \underline{f}dx\, \underline{v}dx + \int_{\Gamma^N} \underline{g} \cdot \underline{v}$$

- Nearly-incompressible elasticity equations

$$\begin{cases} 2\displaystyle\int_\Omega \mu \underline{\underline{\varepsilon}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{v})dx & - \displaystyle\int_\Omega p\underline{\nabla} \cdot (\underline{v})dx = \int_\Omega \underline{f}\underline{v}dx + \int_{\Gamma^N} \underline{g} \cdot \underline{v}, \\ -\displaystyle\int_\Omega \underline{\nabla} \cdot (\underline{u})qdx & - \displaystyle\int_\Omega \frac{1}{\lambda} pqdx = 0 \end{cases}$$

Fields of Application

- Compressible elasticity equation

$$\int_\Omega 2\mu\, \underline{\underline{\varepsilon}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{v})dx + \int_\Omega \lambda\, \underline{\nabla} \cdot (\underline{u})\, \underline{\nabla} \cdot (\underline{v})dx = \int_\Omega \underline{f}dx\, \underline{v}dx + \int_{\Gamma^N} \underline{g} \cdot \underline{v}$$

- Nearly-incompressible elasticity equations

$$\begin{cases} 2\int_\Omega \mu \underline{\underline{\varepsilon}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{v})dx & -\int_\Omega p\underline{\nabla} \cdot (\underline{v})dx = \int_\Omega \underline{f}\underline{v}dx + \int_{\Gamma^N} \underline{g} \cdot \underline{v}, \\ -\int_\Omega \underline{\nabla} \cdot (\underline{u})qdx & -\int_\Omega \frac{1}{\lambda}pqdx = 0 \end{cases}$$

$$\underline{\underline{\sigma}}_S(\boldsymbol{u}).n + \mathcal{L}(\alpha)\,\boldsymbol{u} = 0. \text{ on } \partial\Omega_i \backslash \partial\Omega$$

Where $\mathcal{L}$ is constructed from the Lamé coefficient of the material and it is defined as follows

$$\mathcal{L}(\alpha, \lambda, \mu) := \frac{2\alpha\mu(2\mu + \lambda)}{\lambda + 3\mu}$$

- Stokes equations

$$\begin{cases} \int_\Omega 2\mu \underline{\underline{\varepsilon}}(\underline{u}) : \underline{\underline{\varepsilon}}(\underline{v}) dx & - \int_\Omega p \nabla \cdot (\underline{v}) dx = \int_\Omega \underline{f} \underline{v} dx, \\ \int_\Omega \nabla \cdot (\underline{u}) q dx & = 0 \end{cases}$$

$$\underline{\underline{\sigma}}_F(\boldsymbol{u}).n + \mathcal{L}(\alpha)\,\boldsymbol{u} = 0. \text{ on } \partial\Omega_i \backslash \partial\Omega$$

Where $\mathcal{L}$ is constructed from viscosity coefficient of the fluid and it is defined as follows

$$\mathcal{L}(\alpha, \lambda, \mu) := \frac{2\alpha\mu(2\mu + \lambda)}{\lambda + 3\mu} \text{ with } \lambda = 10^8$$

- Diffusion equation

$$\int_\Omega \kappa \underline{\nabla} u \cdot \underline{\nabla} v \quad dx = \int_\Omega f \quad v dx$$

HPDDM Library - P. Jolivet & F. Nataf

An implementation of several Domain Decomposition Methods :

- One-and two-level Schwarz methods
- The Finite Element Tearing and Interconnecting (FETI) method
- Balancing Domain Decomposition (BDD) method

Library written in C++11 MPI and :

- Linked with automatic partitioners (METIS & SCOTCH).
- Linked with BLAS & LAPACK.
- Linked with direct solvers (MUMPS, SuiteSparse, MKL PARDISO, PASTIX).
- Linked with eigenvalue solver (ARPACK).
- Interfaced with discretisation kernel FreeFem++ & FEEL++

Machine used for scaling tests

Curie

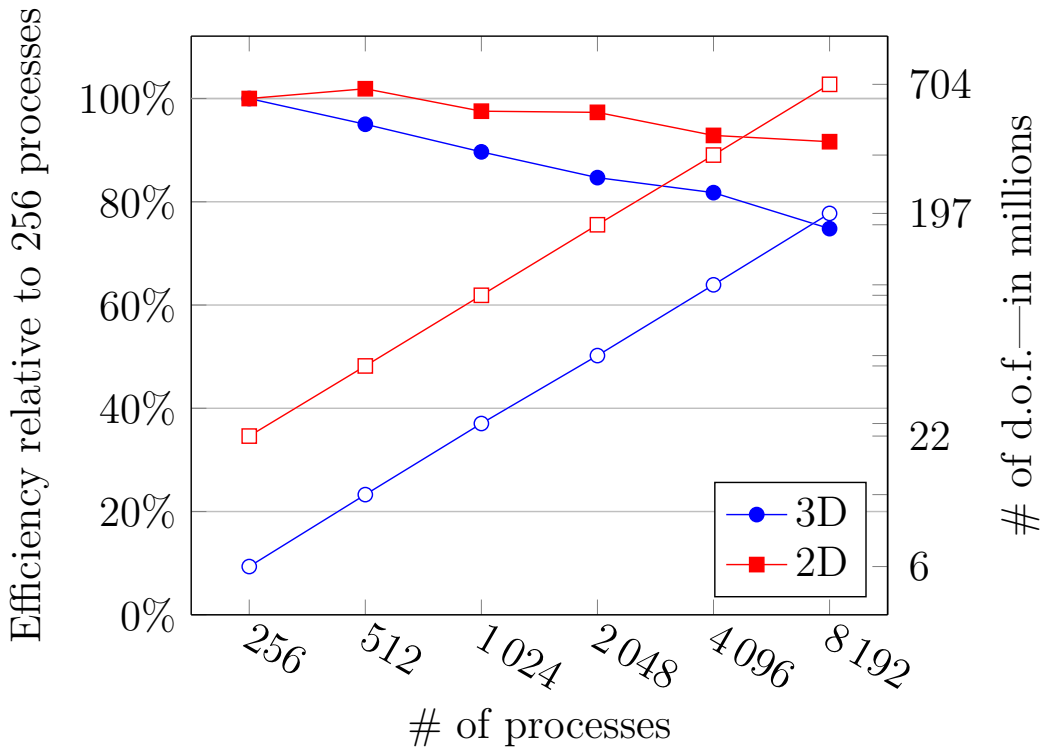
- 5,040 compute nodes
- 2 eight-core Intel Sandy Bridge@2.7 GHz per node
- 1.7 PFLOPs peak performance

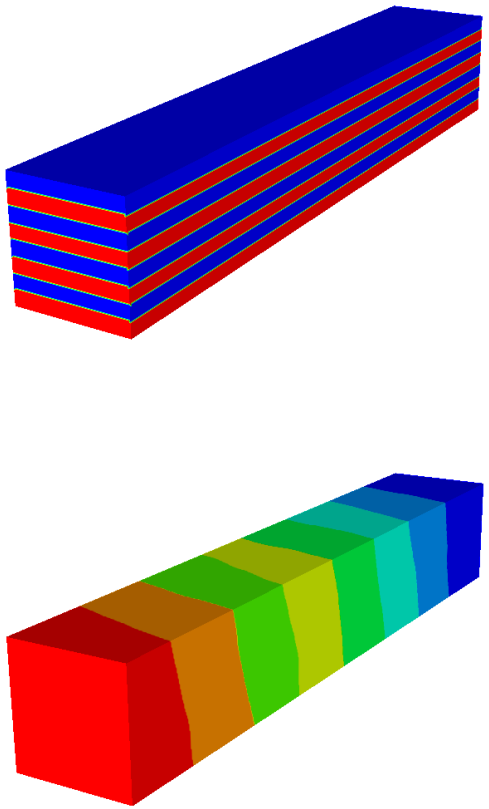
Turing, IDRIS-Genci project

- IBM Blue Gene/Q
- 6144 compute nodes (16 core per node @1.6 GHZ)
- 1.258 PFLOPs peak performance

Weak scalability (Nearly-Incompressible linear elasticity)

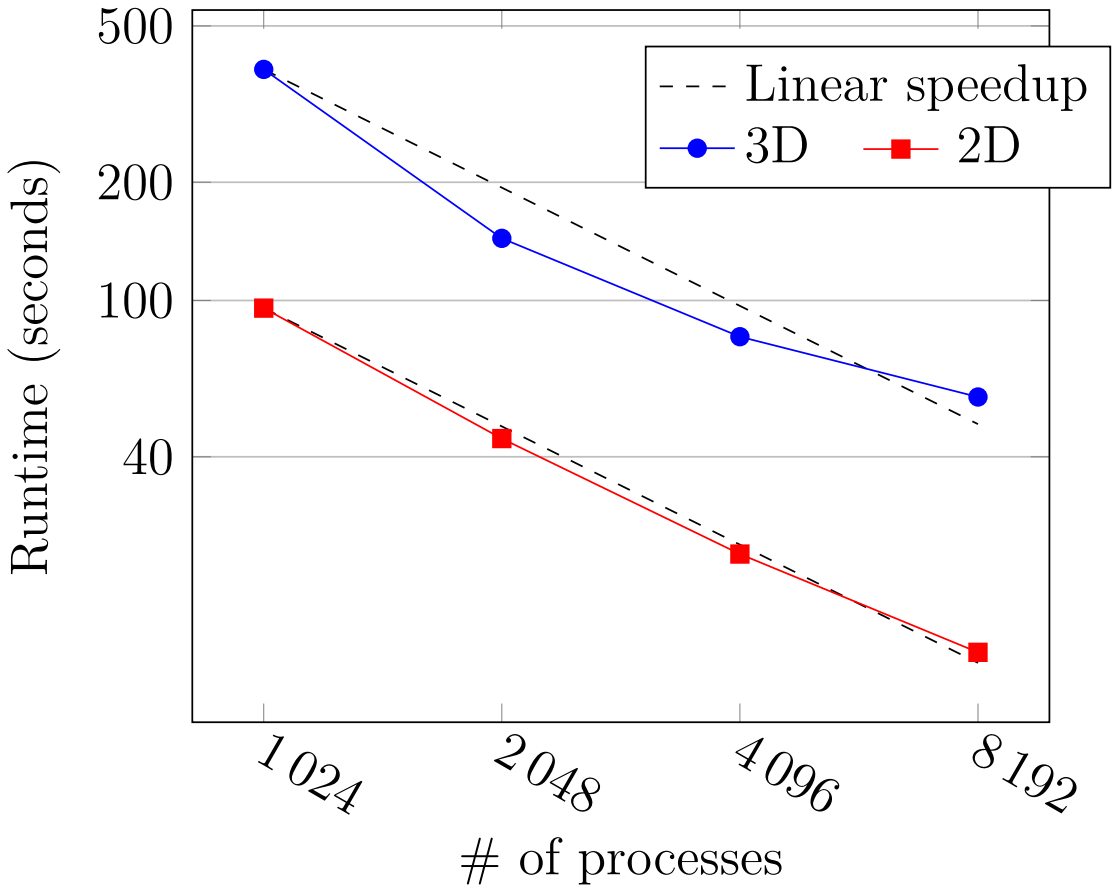
"Sandwich" problem discretized by $\mathbb{P}_3\backslash\mathbb{P}_2$(2D), $\mathbb{P}_2\backslash\mathbb{P}_1$(3D)
Automatic mesh partition with METIS, 1 subdomain/MPI process, 2 OpenMP threads/MPI process





Strong scalability test (Stokes problem)

Led Driven cavity discretized by $\mathbb{P}_2\backslash\mathbb{P}_1$ FE 50M d.o.f. (3D), 100M d.o.f. (2D)
Automatic mesh partition with METIS



Outline

Incompressible Navier-Stokes in Turek benchmark

$$\begin{cases} \rho\dfrac{\partial \underline{u}}{\partial t} + \rho\,\underline{u}\cdot\nabla\underline{u} - \underline{\nabla}\cdot\underline{\underline{\sigma}}_F(\underline{u},p) = \underline{\mathrm{f}} \text{ in } \Omega\times(0,T) \\ \underline{\nabla}\cdot\underline{u} = 0 \text{ in } \Omega\times(0,T) \\ \underline{u} = \underline{\mathrm{g}} \text{ on } \Gamma_D\times(0,T) \\ \underline{\underline{\sigma}}_F(\underline{u},p)\underline{n} = \mathrm{h} \text{ on } \Gamma_N\times(0,T) \\ \underline{u}(0) = u_0 \text{ in } \Omega\times 0 \end{cases}$$

with BDF2

$$\begin{cases} \rho\frac{3\underline{u}^{n+1}-4\underline{u}^{n+1}+\underline{u}^{n+1}}{2dt} + \rho(2\underline{u}^n - \underline{u}^{n-1})\cdot\nabla\underline{u}^{n+1} - \mu\Delta\underline{u}^{n+1} + \nabla p^{n+1} = \underline{\mathrm{f}} \text{ in } \Omega \\ \nabla\cdot\underline{u}^{n+1} = 0 \text{ in } \Omega \\ \text{Boundary conditions} \end{cases}$$

$$A = \begin{bmatrix} H & L^T \\ L & 0 \end{bmatrix}\begin{bmatrix} U \\ P \end{bmatrix}^{n+1} = F,$$

with $H = \frac{1}{\Delta t}M + A + C(U^n)$

DDM preconditionning & Krylov recyling

$$\mathbf{A}^n\mathbf{U}^n = \mathbf{F}^n \quad n = 1, 2, \ ...$$

$$M^{-1}_{SORAS,1} := \sum_{i=1}^{N} R_i^T D_i B_i^{-1} D_i R_i$$

$$M^{-1}_{ORAS,1} := \sum_{i=1}^{N} R_i^T D_i B_i^{-1} R_i$$

Use GCRO-DR in [Parks et al. 2006] as implemented in [Jolivet and Tournier 2016]

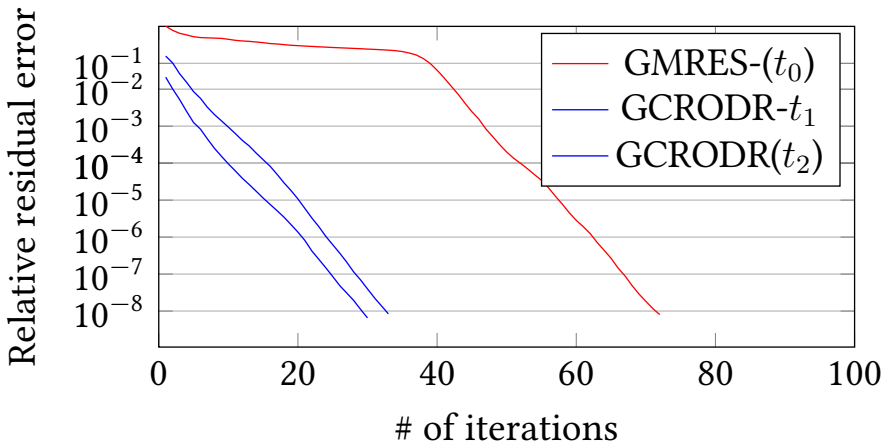
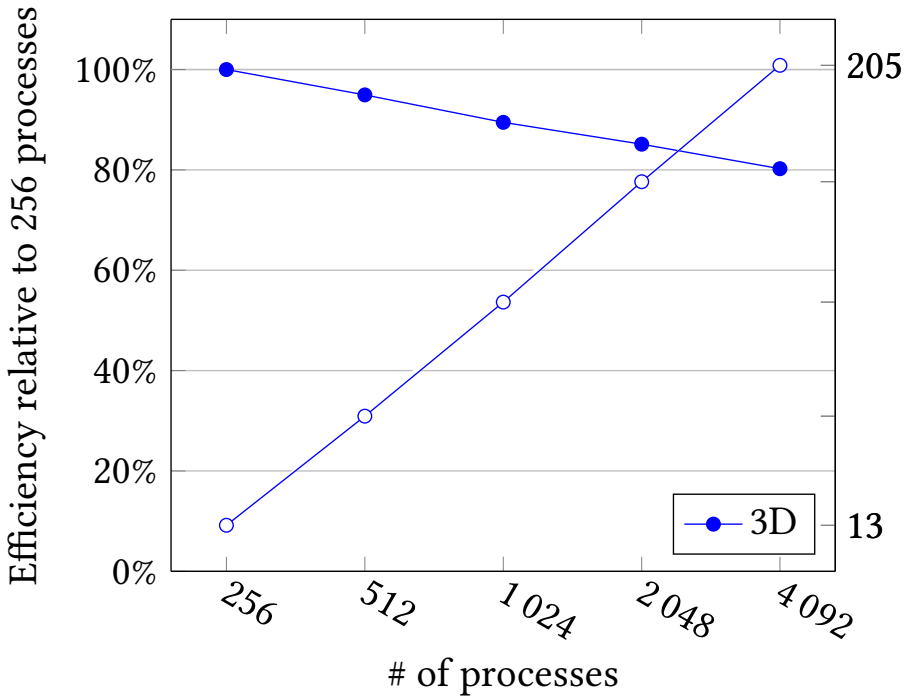
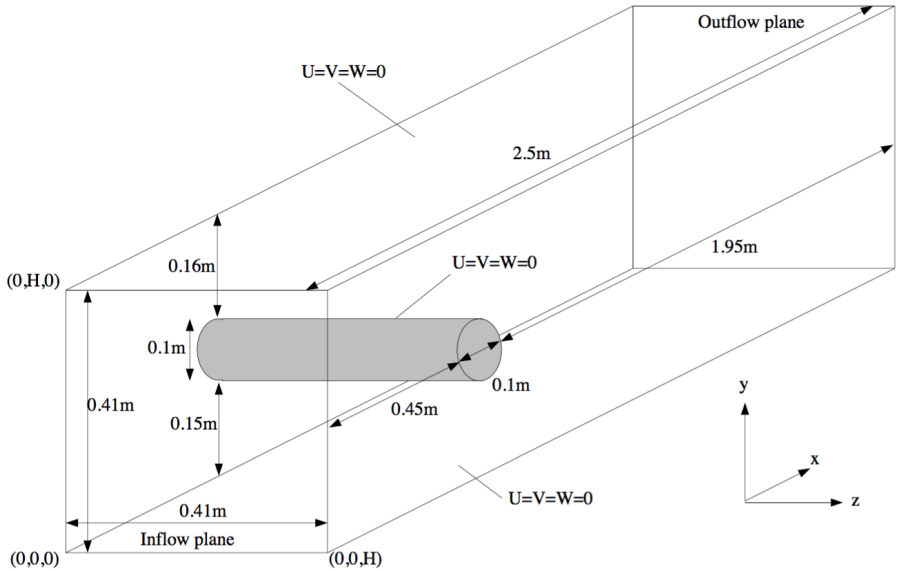


Figure: GMRES iterations - Navier-Stokes problem with 2 millions d.o.f solved on 128 proc for 3 time steps.

Weak scalability - Incompressible Navier-Stokes in Turek benchmark

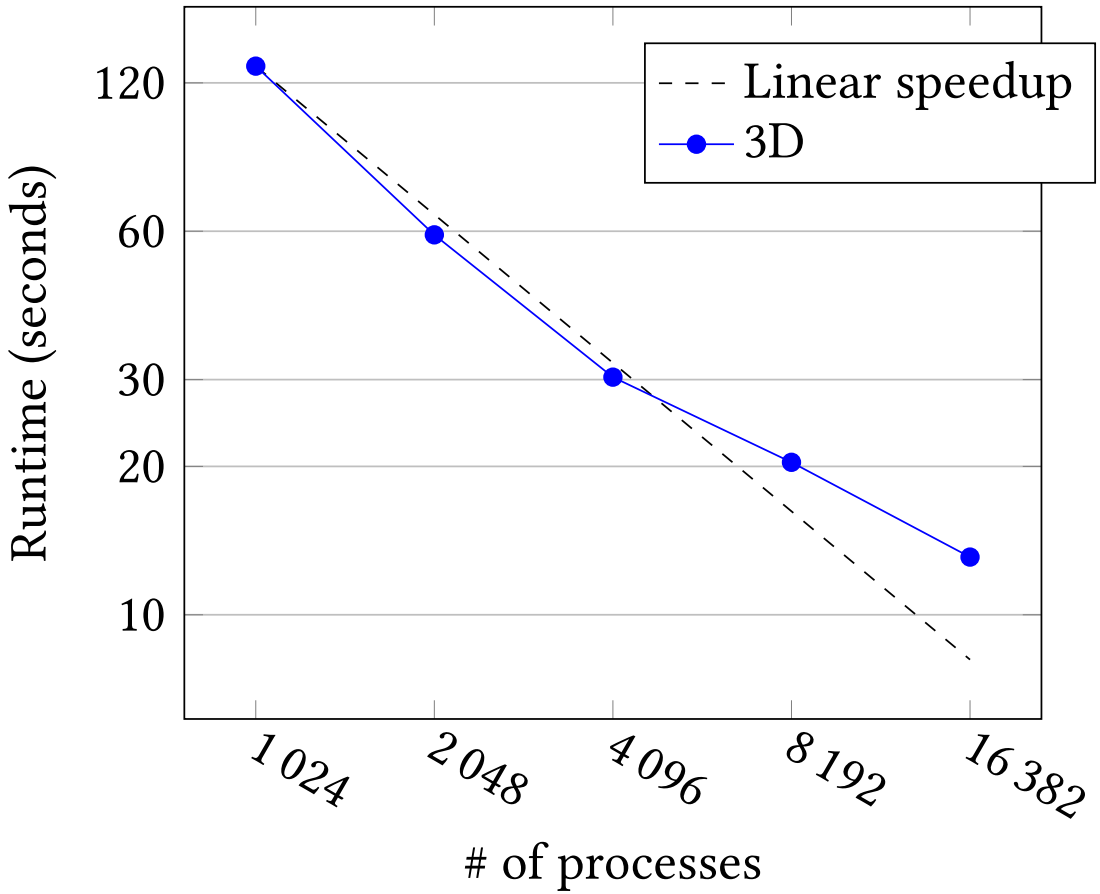




$C_D = 0.31 \in [0.29, 0.31] \quad C_L = -0.01 \in [-0.01, -0.008] \quad Strouhal = 0.36$

Strong scalability - Incompressible Navier-Stokes

Turek benchmark discretized by $\mathbb{P}_2\backslash\mathbb{P}_1$ FE, 87M d.o.f. (3D)
automatic mesh partition with METIS



Outline

Conclusion

Summary

- SORAS preconditioner

$$M_{\text{SORAS}}^{-1} := \sum_{i=1}^{N} R_i^T D_i B_i^{-1} D_i R_i$$

  is amenable to a fruitful theory for OSM
- Using two generalized eigenvalue problems, we are able to achieve a targeted convergence rate for OSM
- Nonlinear time dependent problem
- Freely available via HPDDM library or FreeFem++

Future work

- Another look at parameter α optimization
- Multilevel extension of the coarse operator
- Recycle coarse space


Cai, Xiao-Chuan and Marcus Sarkis (1999). "A restricted additive Schwarz preconditioner for general sparse linear systems". In: *SIAM Journal on Scientific Computing* 21, pp. 239–247.


St-Cyr, A., M. J. Gander, and S. J. Thomas (2007). "Optimized multiplicative, additive, and restricted additive Schwarz preconditioning". In: *SIAM J. Sci. Comput.* 29.6, 2402–2425 (electronic). ISSN: 1064-8275. DOI: 10.1137/060652610. URL: http://dx.doi.org/10.1137/060652610.


Gander, Martin J. and Gerhard Wanner (2014). "The Origins of the Alternating Schwarz Method". In: *Domain Decomposition Methods in Science and Engineering XXI*. Springer, pp. 487–495.


Haferssas, Ryadh, Pierre Jolivet, and Frdric Nataf (2015). "A robust coarse space for optimized Schwarz methods: SORAS-GenEO-2". In: *C. R. Math. Acad. Sci. Paris* 353.10, pp. 959–963. ISSN: 1631-073X. DOI: 10.1016/j.crma.2015.07.014. URL: http://dx.doi.org/10.1016/j.crma.2015.07.014.


Jolivet, Pierre and Pierre-Henri Tournier (2016). "Block Iterative Methods and Recycling for Improved Scalability of Linear Solvers". In: *Proceedings of the 2016 International Conference for High Performance Computing, Networking, Storage and Analysis*. SC16. IEEE.


Mandel, Jan (1992). "Balancing domain decomposition". In: *Comm. on Applied Numerical Methods* 9, pp. 233–241.


Parks, Michael L., Eric de Sturler, Greg Mackey, Duane D. Johnson, and Spandan Maiti (2006). "Recycling Krylov subspaces for sequences of linear systems". In: *SIAM J. Sci. Comput.* 28.5, 1651–1674 (electronic). ISSN: 1064-8275. DOI: 10.1137/040607277. URL: http://dx.doi.org/10.1137/040607277.


Spillane, N., V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl (2014). "Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps". In: *Numer. Math.* 126.4, pp. 741–770. ISSN: 0029-599X. DOI: 10.1007/s00211-013-0576-y. URL: http://dx.doi.org/10.1007/s00211-013-0576-y.


Tang, J.M., R. Nabben, C. Vuik, and Y.A. Erlangga (2009). "Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods". In: *Journal of Scientific Computing* 39.3, pp. 340–370.


Toselli, Andrea and Olof Widlund (2005). *Domain Decomposition Methods - Algorithms and Theory*. Vol. 34. Springer Series in Computational Mathematics. Springer.

Weak scaling (Nearly-Incompressible linear elasticity)-time

	N	Factorization	Deflation	Solution	# of it.	Total	# of d.o.f.
3D	1 024	79.2 s	229.0 s	76.3 s	45	387.5 s	$50.63 \cdot 10^6$
	2 048	29.5 s	76.5 s	34.8 s	42	143.9 s	
	4 096	11.1 s	45.8 s	19.8 s	42	80.9 s	
	8 192	4.7 s	26.1 s	14.9 s	41	56.8 s	
2D	1 024	5.2 s	37.9 s	51.5 s	51	95.6 s	$100.13 \cdot 10^6$
	2 048	2.4 s	19.3 s	22.1 s	42	44.5 s	
	4 096	1.1 s	10.4 s	10.2 s	35	22.6 s	
	8 192	0.5 s	4.6 s	6.9 s	38	12.7 s	

Weak scaling (Nearly-Incompressible linear elasticity), -time

1 subdomain/MPI process, 2 OpenMP threads/MPI process.

	N	Factorization	Deflation	Solution	# of it.	Total	# of d.o.f.
3D	256	25.2 s	76.0 s	37.2 s	46	145.2 s	$6.1 \cdot 10^6$
	512	26.5 s	81.1 s	39.8 s	47	155.1 s	$12.4 \cdot 10^6$
	1 024	29.2 s	82.6 s	41.7 s	45	165.5 s	$25.0 \cdot 10^6$
	2 048	26.9 s	83.5 s	46.3 s	47	171.0 s	$48.8 \cdot 10^6$
	4 096	28.3 s	88.8 s	54.5 s	53	177.7 s	$97.9 \cdot 10^6$
	8 192	29.0 s	78.3 s	79.8 s	60	196.1 s	$197.6 \cdot 10^6$
2D	256	4.8 s	72.9 s	39.9 s	46	123.9 s	$22.1 \cdot 10^6$
	512	4.7 s	65.9 s	45.0 s	51	121.3 s	$44.0 \cdot 10^6$
	1 024	4.8 s	70.0 s	46.1 s	51	127.0 s	$88.3 \cdot 10^6$
	2 048	4.8 s	69.0 s	46.5 s	51	127.4 s	$176.8 \cdot 10^6$
	4 096	4.8 s	65.8 s	52.8 s	56	132.6 s	$351.0 \cdot 10^6$
	8 192	4.8 s	65.4 s	53.0 s	54	134.8 s	$704.1 \cdot 10^6$

Weak scaling (Incompressible Navier-Stokes in Turek benchmark, with FreeFem++ and HPDDM)

	N	Factorization	GCRODR	# of it.	Total	# of d.o.f.
3D	256	28.2 s	17.0 s	29	67.9 s	$13.7 \cdot 10^6$
	512	30.5 s	16.7 s	28	68.1 s	$26.0 \cdot 10^6$
	1 024	30.6 s	19.7 s	35	70.8 s	$50.9 \cdot 10^6$
	2 048	30.9 s	24.0 s	41	75.7 s	$103.6 \cdot 10^6$
	4 092	28.6 s	28.5 s	57	79.8 s	$205.8 \cdot 10^6$

Strong scaling (Incompressible Navier-Stokes in Turek benchmark, with FreeFem++ and HPDDM)

	N	Factorization	GCRODR	# of it.	Total Step	# of d.o.f.
3D	1 024	70.4 s	27.2 s	36	129.7 s	$87.11 \cdot 10^6$
	2 048	23.7 s	16.7 s	46	59.0 s	
	4 096	8.1 s	10.3 s	57	30.3 s	
	8 192	3.6 s	7.0 s	66	20.4 s	
	16 384	1.5 s	4.7 s	75	13.1 s	