Adaptive and Robust Coarse Space construction for Domain Decomposition Methods

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

[1]IRIT, ENSEEIHT

[2]LJLL-CNRS-INRIA(Alpines)

December 10th, 2015

Outline

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

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

- Darcy $\quad a_\Omega(u,v) = \int_\Omega \kappa \underline{\nabla}\, u \cdot \underline{\nabla}\, v \quad dx$
- Elasticty $\quad a_\Omega(u,v) = \int_\Omega \underline{\underline{\underline{\underline{C}}}} : \underline{\underline{\varepsilon}}(u) : \underline{\underline{\varepsilon}}(v) \quad dx$
- Helmholtz $\quad a_\Omega(u,v) = \int_\Omega k^2 uv + \underline{\nabla}\, u \cdot \underline{\nabla}\, v \quad dx$

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

- Darcy $\quad a_\Omega(u,v) = \int_\Omega \kappa \underline{\nabla} u \cdot \underline{\nabla} v \quad dx$
- Elasticty $\quad a_\Omega(u,v) = \int_\Omega \underline{\underline{\underline{\underline{C}}}} : \underline{\underline{\varepsilon}}(u) : \underline{\underline{\varepsilon}}(v) \quad dx$
- Helmholtz $\quad a_\Omega(u,v) = \int_\Omega k^2 uv + \underline{\nabla} u \cdot \underline{\nabla} v \quad dx$

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 a thousands of processors.

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

- Darcy $\quad a_\Omega(u,v) = \int_\Omega \kappa \underline{\nabla} u \cdot \underline{\nabla} v \quad dx$
- Elasticty $\quad a_\Omega(u,v) = \int_\Omega \underline{\underline{\underline{\underline{C}}}} : \underline{\underline{\varepsilon}}(u) : \underline{\underline{\varepsilon}}(v) \quad dx$
- Helmholtz $\quad a_\Omega(u,v) = \int_\Omega k^2 uv + \underline{\nabla} u \cdot \underline{\nabla} v \quad dx$

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 a thousands of processors.

Tools

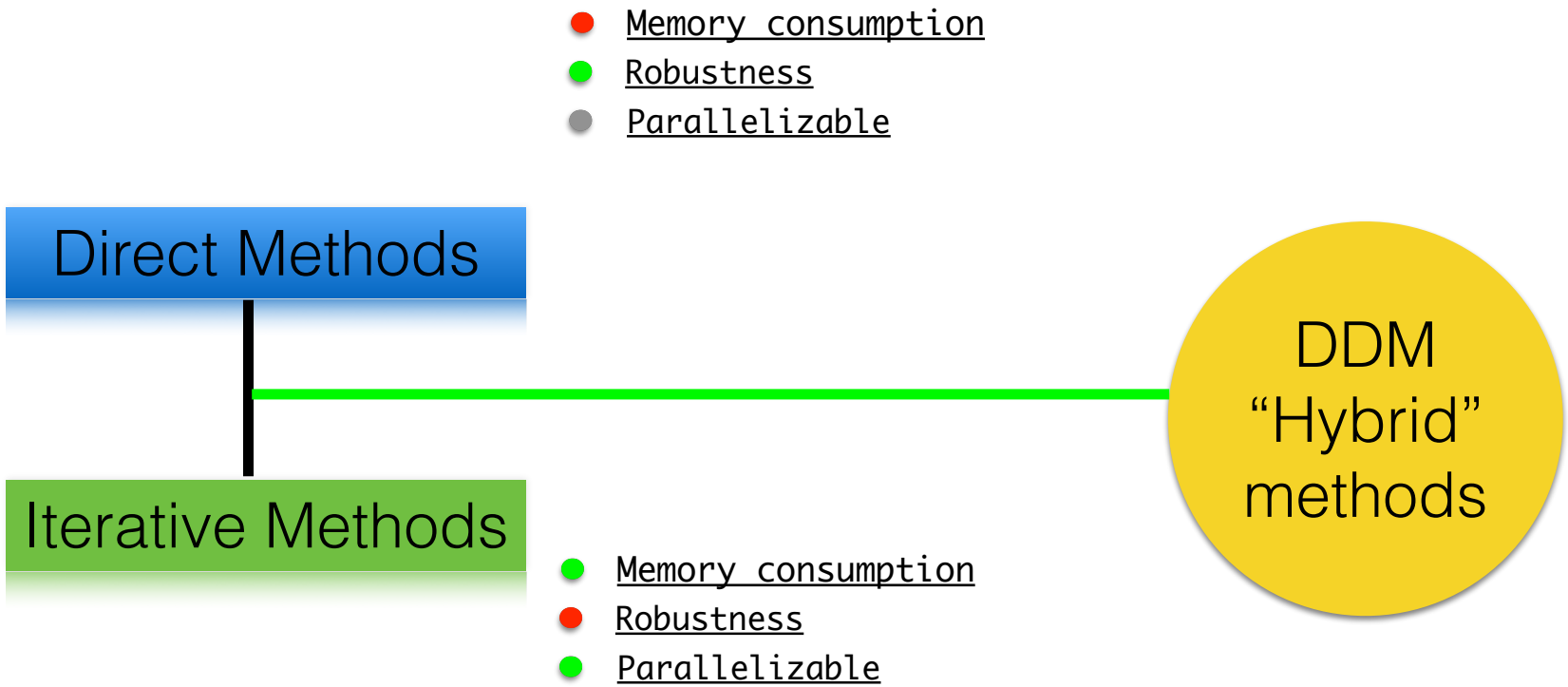
- A good mathematical foundation theory & HPDDM Library: a parallel framework .

Why Domain Decomposition Methods ?

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

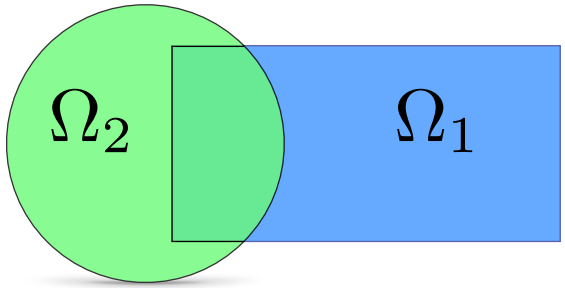
Why Domain Decomposition Methods ?

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



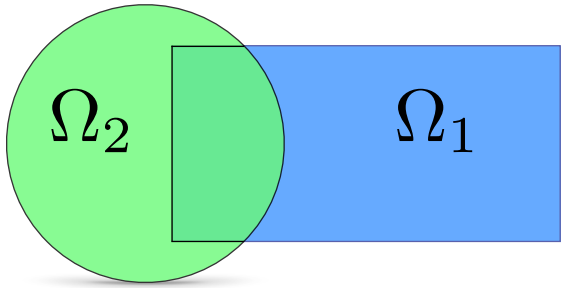
Original Schwarz domain decomposition method (H.A. Schwarz, 1870)

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



Original Schwarz domain decomposition method (H.A. Schwarz, 1870)

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



With $\Omega := \Omega_1 \cup \Omega_2$, $(u_1^n, u_2^n) \rightarrow (u_1^{n+1}, u_2^{n+1})$

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

From the partition $\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$. where $\mathcal{N} = dof(\Omega)$ Define

From the partition $\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$. where $\mathcal{N} = dof(\Omega)$ Define

- R_i A restriction operator $R_i : \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$.

From the partition $\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$. where $\mathcal{N} = dof(\Omega)$ Define

- R_i A restriction operator $R_i : \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$.
- $R_i^\intercal$ An extension operator by zero $R_i^\intercal : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}}$.

From the partition $\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$. where $\mathcal{N} = dof(\Omega)$ Define

- R_i A restriction operator $R_i : \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$.
- $R_i^\intercal$ An extension operator by zero $R_i^\intercal : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}}$.
- Define a partition of unity $D_i : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$

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

From the partition $\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$. where $\mathcal{N} = dof(\Omega)$ Define

- R_i A restriction operator $R_i : \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$.
- $R_i^\intercal$ An extension operator by zero $R_i^\intercal : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}}$.
- Define a partition of unity $D_i : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$

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

To solve $\boldsymbol{A}u = F$ Schwarz methods can be viewed as preconditionners for a fixed point algorithm:

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

From the partition $\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$. where $\mathcal{N} = dof(\Omega)$ Define

- R_i A restriction operator $R_i : \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$.
- $R_i^\intercal$ An extension operator by zero $R_i^\intercal : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}}$.
- Define a partition of unity $D_i : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$

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

To solve $\boldsymbol{A}u = F$ Schwarz methods can be viewed as preconditionners for a fixed point algorithm:

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

Where :

- $M_{RAS,1}^{-1} := \sum_{i=1}^{N} R_i^T D_i A_i^{-1} R_i$

From the partition $\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$. where $\mathcal{N} = dof(\Omega)$ Define

- R_i A restriction operator $R_i : \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$.
- $R_i^\intercal$ An extension operator by zero $R_i^\intercal : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}}$.
- Define a partition of unity $D_i : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$

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

To solve $\boldsymbol{A}u = F$ Schwarz methods can be viewed as preconditionners for a fixed point algorithm:

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

Where :

- $M_{RAS,1}^{-1} := \sum_{i=1}^{N} R_i^T D_i A_i^{-1} R_i$
- $M_{\text{ASM},1}^{-1} := \sum_{i=1}^{N} R_i^\intercal A_i{}^{-1} R_i$ with $A_i = R_i A R_i^\intercal$.

From the partition $\mathcal{N} = \bigcup_{i=1}^{N} \mathcal{N}_i$. where $\mathcal{N} = dof(\Omega)$ Define

- R_i A restriction operator $R_i : \mathbb{R}^{\#\mathcal{N}} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$.
- $R_i^\intercal$ An extension operator by zero $R_i^\intercal : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}}$.
- Define a partition of unity $D_i : \mathbb{R}^{\#\mathcal{N}_i} \longrightarrow \mathbb{R}^{\#\mathcal{N}_i}$

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

To solve $\boldsymbol{A}u = F$ Schwarz methods can be viewed as preconditionners for a fixed point algorithm:

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

Where :

- $M_{RAS,1}^{-1} := \sum_{i=1}^{N} R_i^T D_i A_i^{-1} R_i$
- $M_{\text{ASM},1}^{-1} := \sum_{i=1}^{N} R_i^\intercal {A_i}^{-1} R_i$ with $A_i = R_i A R_i^\intercal$.
- $M_{\text{SORAS},1}^{-1} := \sum_{i=1}^{N} R_i^\intercal D_i B_i^{-1} D_i R_i$ Optimized conditions

Why One-Level methods are not so good?

		AS	RAS	ORAS	SORAS
Nbr DOFs	Nbr subdom	iteration	iteration	iteration	iteration
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 of GMRES iteration for compressible case with $E = 10^7$ and $\nu = 0.3$

Why One-Level methods are not so good?

		AS	RAS	ORAS	SORAS
Nbr DOFs	Nbr subdom	iteration	iteration	iteration	iteration
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 of GMRES iteration for compressible case with $E = 10^7$ and $\nu = 0.3$

For a one-level preconditioner M_1, the condition number is given

$$\kappa(M^{-1}A) \leqslant C \frac{1}{H^2} \left(1 + \frac{H}{\delta}\right) \qquad \text{[Widlund \& Dryija]}$$

where :

- δ size of the overlap
- H size of subdomain.

Two Level Domain Decomposition Methods

Given $V_H := \mathbf{span} Z$ an additive space. Z a set of vectors.

Define $R_H = Z^\intercal$ and $E := R_H A R_H^\intercal$ where E is 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 $V_H := \mathbf{span} Z$ an additive space. Z a set of vectors.

Define $R_H = Z^\intercal$ and $E := R_H A R_H^\intercal$ where E is 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 $V_H := \mathbf{span} Z$ an additive space. Z a set of vectors.

Define $R_H = Z^\intercal$ and $E := R_H A R_H^\intercal$ where E is 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 $V_H := \mathbf{span} Z$ an additive space. Z a set of vectors.

Define $R_H = Z^\intercal$ and $E := R_H A R_H^\intercal$ where E is much smaller than A

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

- $M_{2,AD}^{-1} := Q + M^{-1}$.
- $M_{2,A-DEF_1}^{-1} := M^{-1}(I - AQ) + Q$.
- $M_{2,A-DEF_2}^{-1} := (I - QA)M^{-1} + Q$.

Two Level Domain Decomposition Methods

Given $V_H := \mathbf{span} Z$ an additive space. Z a set of vectors.

Define $R_H = Z^\intercal$ and $E := R_H A R_H^\intercal$ where E is much smaller than A

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

- $M_{2,AD}^{-1} := Q + M^{-1}$.
- $M_{2,A-DEF_1}^{-1} := M^{-1}(I - AQ) + Q$.
- $M_{2,A-DEF_2}^{-1} := (I - QA)M^{-1} + Q$.
- $M_{2,BNN}^{-1} := (I - QA)M^{-1}(I - AQ) + Q$.

Two Level Domain Decomposition Methods

Given $V_H := \mathbf{span} Z$ an additive space. Z a set of vectors.

Define $R_H = Z^\intercal$ and $E := R_H A R_H^\intercal$ where E is much smaller than A

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

- $M_{2,AD}^{-1} := Q + M^{-1}$.
- $M_{2,A-DEF_1}^{-1} := M^{-1}(I - AQ) + Q$.
- $M_{2,A-DEF_2}^{-1} := (I - QA)M^{-1} + Q$.
- $M_{2,BNN}^{-1} := (I - QA)M^{-1}(I - AQ) + Q$.

But what does the space V_H contain?

GenEO I approach to construct V_H

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

$$A_i^N U_{i,k} = \lambda_{i,k} D_i R_{i,0}^\intercal R_{i,0} A_i^N R_{i,0}^\intercal R_{i,0} D_i U_{i,k}$$

Solved by ARPACK

(Concurrently)

GenEO I approach to construct V_H

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

$$A_i^N U_{i,k} = \lambda_{i,k} D_i R_{i,0}^\intercal R_{i,0} A_i^N R_{i,0}^\intercal R_{i,0} D_i U_{i,k}$$

Solved by ARPACK (Concurrently)

With :

- A_i^N the local unassembled 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 construct V_H

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

$$A_i^N U_{i,k} = \lambda_{i,k} D_i R_{i,0}^\intercal R_{i,0} A_i^N R_{i,0}^\intercal R_{i,0} D_i U_{i,k}$$

Solved by ARPACK (Concurrently)

With :

- A_i^N the local unassembled 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 construct V_H

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

$$A_i^N U_{i,k} = \lambda_{i,k} D_i R_{i,0}^\intercal R_{i,0} A_i^N R_{i,0}^\intercal R_{i,0} D_i U_{i,k}$$

Solved by ARPACK (Concurrently)

With :

- A_i^N the local unassembled 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: $0 < \lambda_{j,m_{j+1}} < \infty$*:*

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

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.

SORAS-GenEO II Coarse space (P.L. Lions' algorithm)

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 Coarse space (P.L. Lions' algorithm)

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 $(\mu_{i,k}, V_{i,k})$

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

Solved by ARPACK (Concurrently)

SORAS-GenEO II Coarse space (P.L. Lions' algorithm)

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 $(\mu_{i,k}, V_{i,k})$

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

Solved by ARPACK (Concurrently)

where :

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

SORAS-GenEO II Coarse space (P.L. Lions' algorithm)

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 $(\mu_{i,k}, V_{i,k})$

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

where :

- B_i the local unassembled 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 Coarse space (P.L. Lions' algorithm)

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 $(\mu_{i,k}, V_{i,k})$

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

where :

- B_i the local unassembled 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 Coarse space (P.L. Lions' algorithm)

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 $(\mu_{i,k}, V_{i,k})$

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

where :

- B_i the local unassembled 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 Coarse space (P.L. Lions' algorithm)

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 $(\mu_{i,k}, V_{i,k})$

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

where :

- B_i the local unassembled 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 (Haferssas, Jolivet and N., 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 neighbors.
- k_1 the maximum multiplicity.
- τ and γ a user-defined thresholds.

Outline

Fields of Application

Fields of Application

- Elasticity

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

Fields of Application

- Elasticity

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

- Nearly-Incompressible

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

Fields of Application

- Elasticity

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

- Nearly-Incompressible

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

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

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

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

- Stokes

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

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

$\mathcal{L}$ is builded from the fluid coefficient $\mathcal{L}(\alpha, \mu) := \frac{4}{3}\alpha\mu$.

- Stokes

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

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

$\mathcal{L}$ is builded from the fluid coefficient $\mathcal{L}(\alpha, \mu) := \frac{4}{3}\alpha\mu$.

- Diffusion

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

- Stokes

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

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

$\mathcal{L}$ is builded from the fluid coefficient $\mathcal{L}(\alpha, \mu) := \frac{4}{3}\alpha\mu$.

- Diffusion

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

Outline

HPDDM Library

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 with MPI and :

- Linked with graph partitioners (METIS & SCOTCH).
- Linked with BLAS & LAPACK.
- Linked with direct solvers (MUMPS, SuiteSparse, MKL PARDISO, PASTIX).
- Linked with eigenvalue solver (ARPACK).
- **Interfaced with FreeFem++ & FEEL++**
- **C++, Python and C interfaces**

Outline

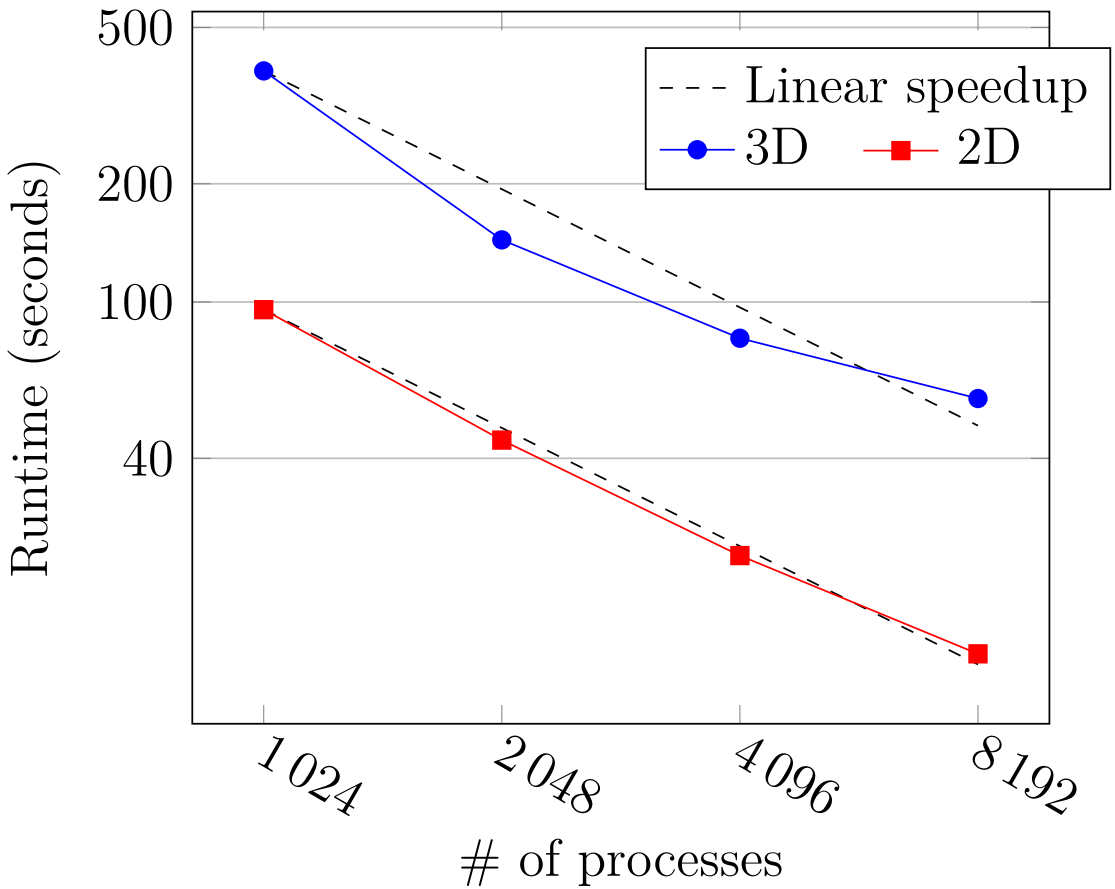
Machine used for scaling tests

Turing, IDRIS-Genci project

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

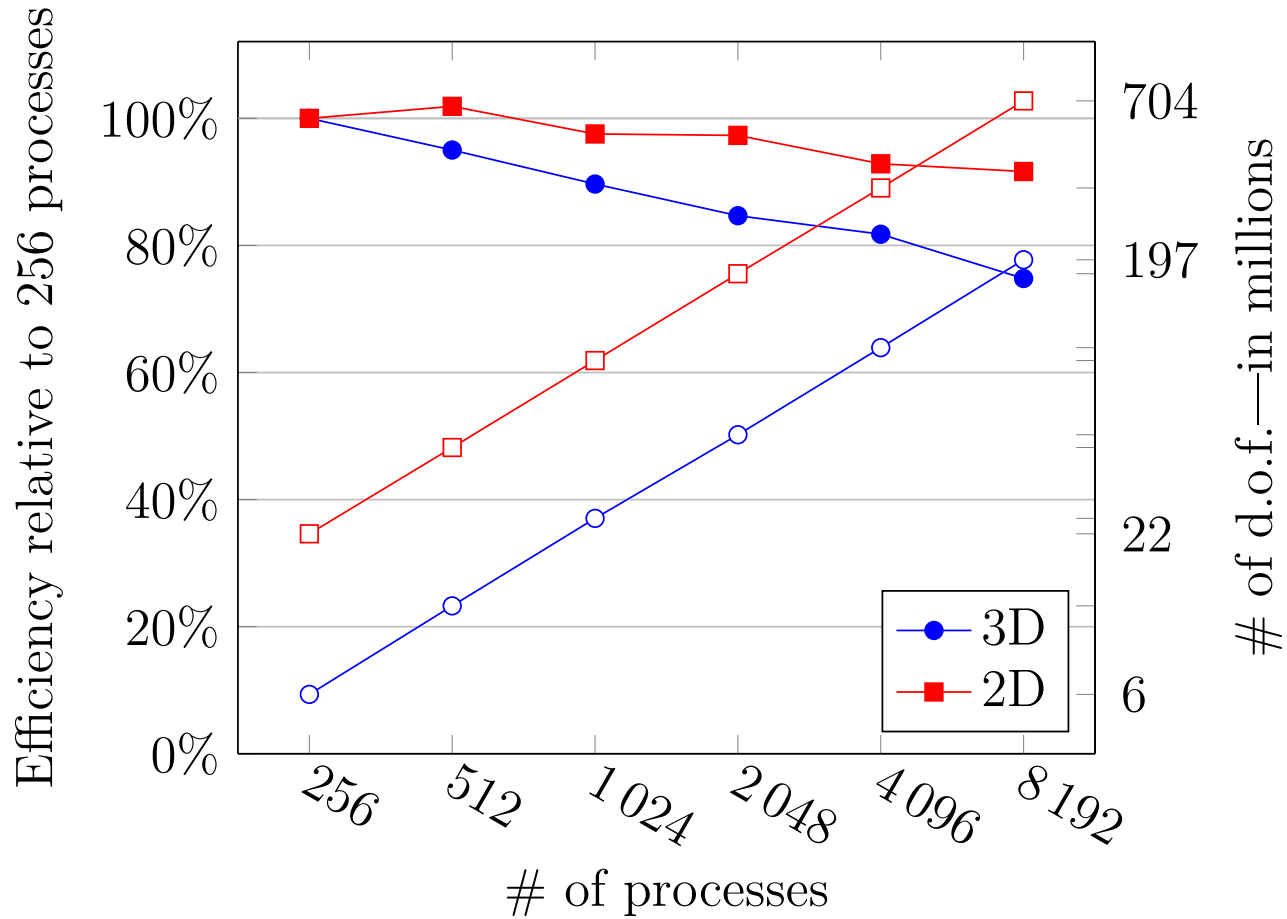
Strong scaling (Stokes problem), (with FreeFem++ and HPDDM)

Stokes problem with automatic mesh partition. Driven cavity Discretized by $\mathbb{P}_2\backslash\mathbb{P}_1$ FE 100M d.o.f. (2D), 50M d.o.f. (3D)



Weak scaling (Nearly-Incompressible linear elasticity), (with FreeFem++ and HPDDM)

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



Conclusion

Summary :

- Using two generalized eigenvalue problems and projection preconditioning we are able to achieve a targeted convergence rate.
- Available in HPDDM C++/MPI library https://www.github.org/hpddm/hpddm
- Available in the public release of FreeFem++

Future :

- Multilevel extension of the coarse operator,
- Nonlinear time dependent problem (Reuse of the coarse space)

Bibliography


N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, R. Scheichl, "Abstract Robust Coarse Spaces for Systems of PDEs via Generalized Eigenproblems in the Overlaps", *Numerische Mathematik*, 2013.


R. Haferssas, P. Jolivet and F Nataf, "A robust coarse space for optimized Schwarz methods: SORAS-GenEO-2", C. R. Math. Acad. Sci. Paris , 2015.


V. Dolean, P. Jolivet and F Nataf, "An Introduction to Domain Decomposition Methods: algorithms, theory and parallel implementation", https://hal.archives-ouvertes.fr/cel-01100932 , Lecture Notes, SIAM, 2015.


P. Jolivet and F Nataf, "HPDDM: high-performance unified framework for domain decomposition methods", https://github.com/hpddm/hpddm , MPI-C++, 2014.

Bibliography


N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, R. Scheichl, "Abstract Robust Coarse Spaces for Systems of PDEs via Generalized Eigenproblems in the Overlaps", *Numerische Mathematik*, 2013.


R. Haferssas, P. Jolivet and F Nataf, "A robust coarse space for optimized Schwarz methods: SORAS-GenEO-2", C. R. Math. Acad. Sci. Paris , 2015.


V. Dolean, P. Jolivet and F Nataf, "An Introduction to Domain Decomposition Methods: algorithms, theory and parallel implementation", https://hal.archives-ouvertes.fr/cel-01100932 , Lecture Notes, SIAM, 2015.


P. Jolivet and F Nataf, "HPDDM: high-performance unified framework for domain decomposition methods", https://github.com/hpddm/hpddm , MPI-C++, 2014.

THANK YOU

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	

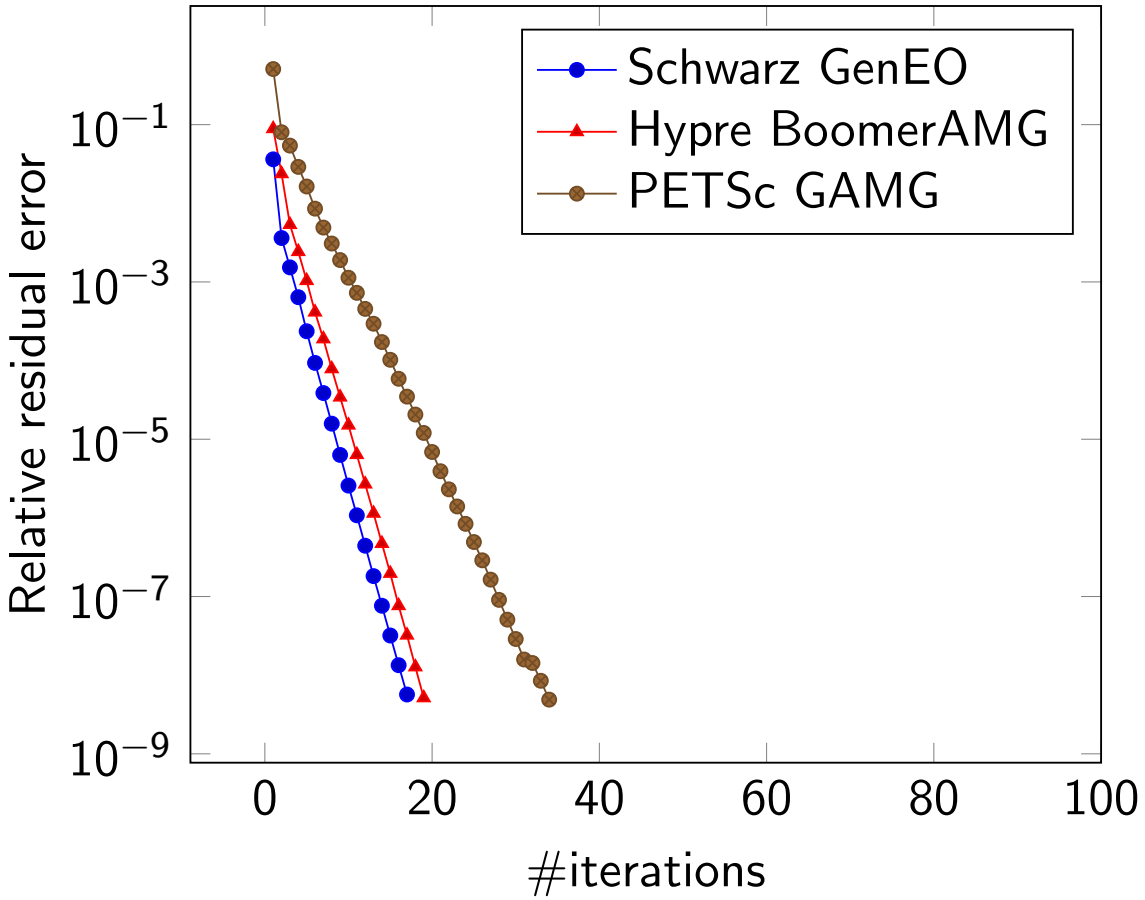
Weak scaling (Nearly-Incompressible linear elasticity), -time

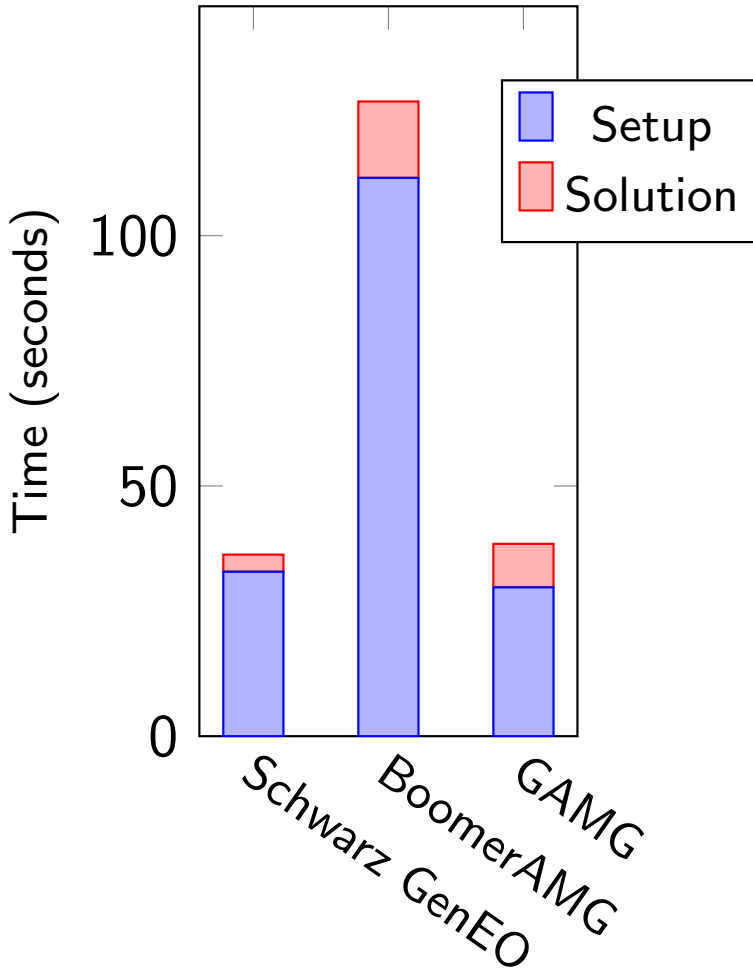
1 subdomain/MPI process, 2 OpenMP threads/MPI process.

	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	$704.1 \cdot 10^6$

Homogeneous 3D Poisson, P.Jolivet

Discretized by $\mathbb{P}_1$ FE solved on 4,096 MPI processes 262M d.o.f.





3D linear elasticity, P.Jolivet

Heterogeneous 3D linear elasticity equation discretized by $\mathbb{P}_2$ FE solved on 4,096 MPI processes, 262M d.o.f.