Algebraic Domain Decomposition Methods for Darcy flow in heterogeneous media

Mikolaj SZYDLARSKI

IFP Energies nouvelles, LJLL

December 29, 2010

Supervisors: M. Pascal Havé *IFP Energies nouvelles*
M. Roland Masson *IFP Energies nouvelles*
M. Frédéric Nataf *Laboratoire J.L. Lions (UPMC)*

Outline

Motivation

Why Domain Decomposition Method ?

- The term Domain Decomposition has slightly different meaning to specialist within the discipline of PDEs.
  - process of distributing data among the processors
  - process of subdividing the solution of large linear system into smaller problem
- Ease of parallelization
  - parallel processing is one way to have a faster codes
  - new generation processors are parallel (multi cores)
- In some situation, the domain decomposition is natural
  - strong heterogeneous media
  - different physics in different subdomains

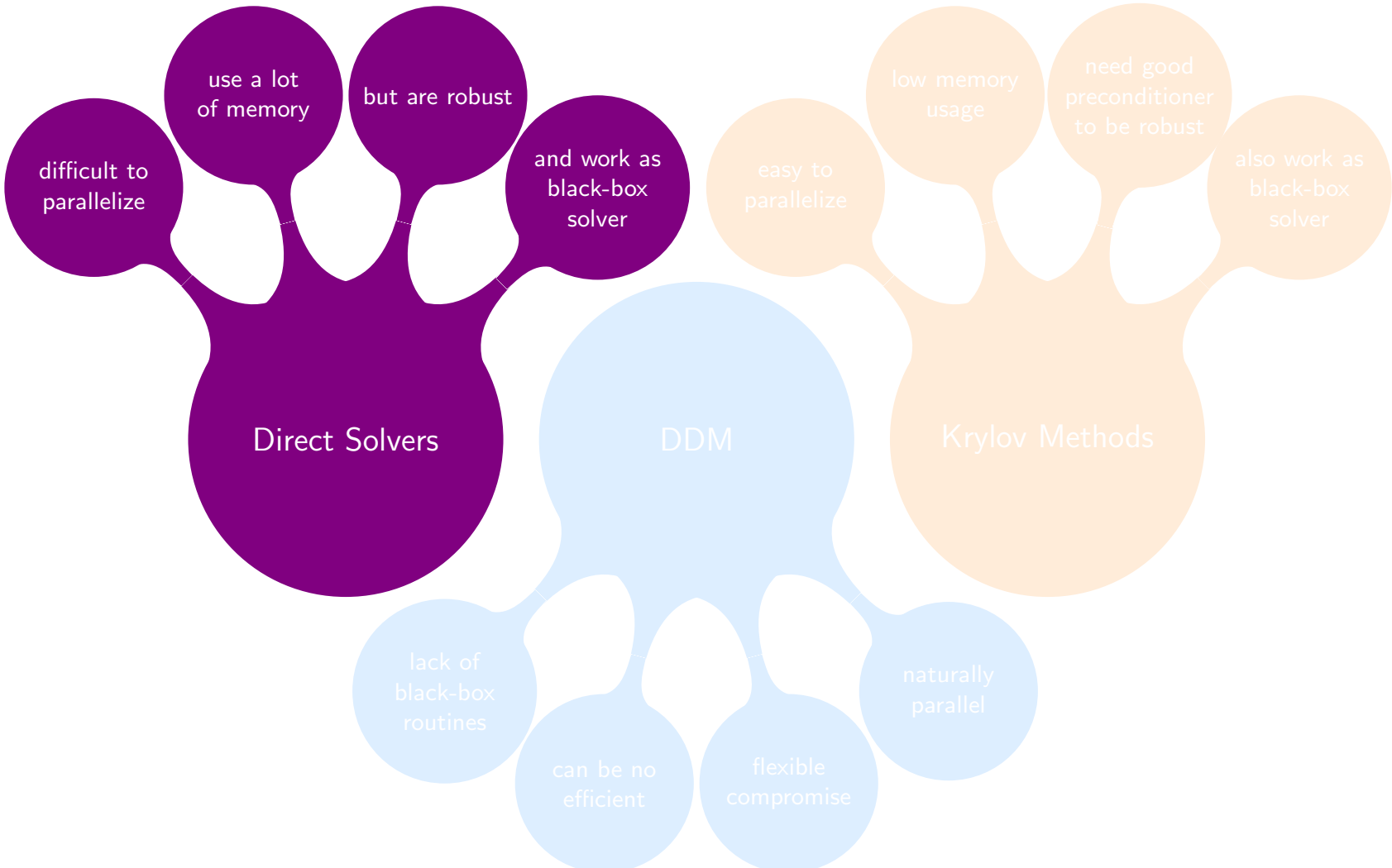
Why Domain Decomposition Method ?

- The term Domain Decomposition has slightly different meaning to specialist within the discipline of PDEs.
  - process of distributing data among the processors
  - process of subdividing the solution of large linear system into smaller problem
- Ease of parallelization
  - parallel processing is one way to have a faster codes
  - new generation processors are parallel (multi cores)
- In some situation, the domain decomposition is natural
  - strong heterogeneous media
  - different physics in different subdomains

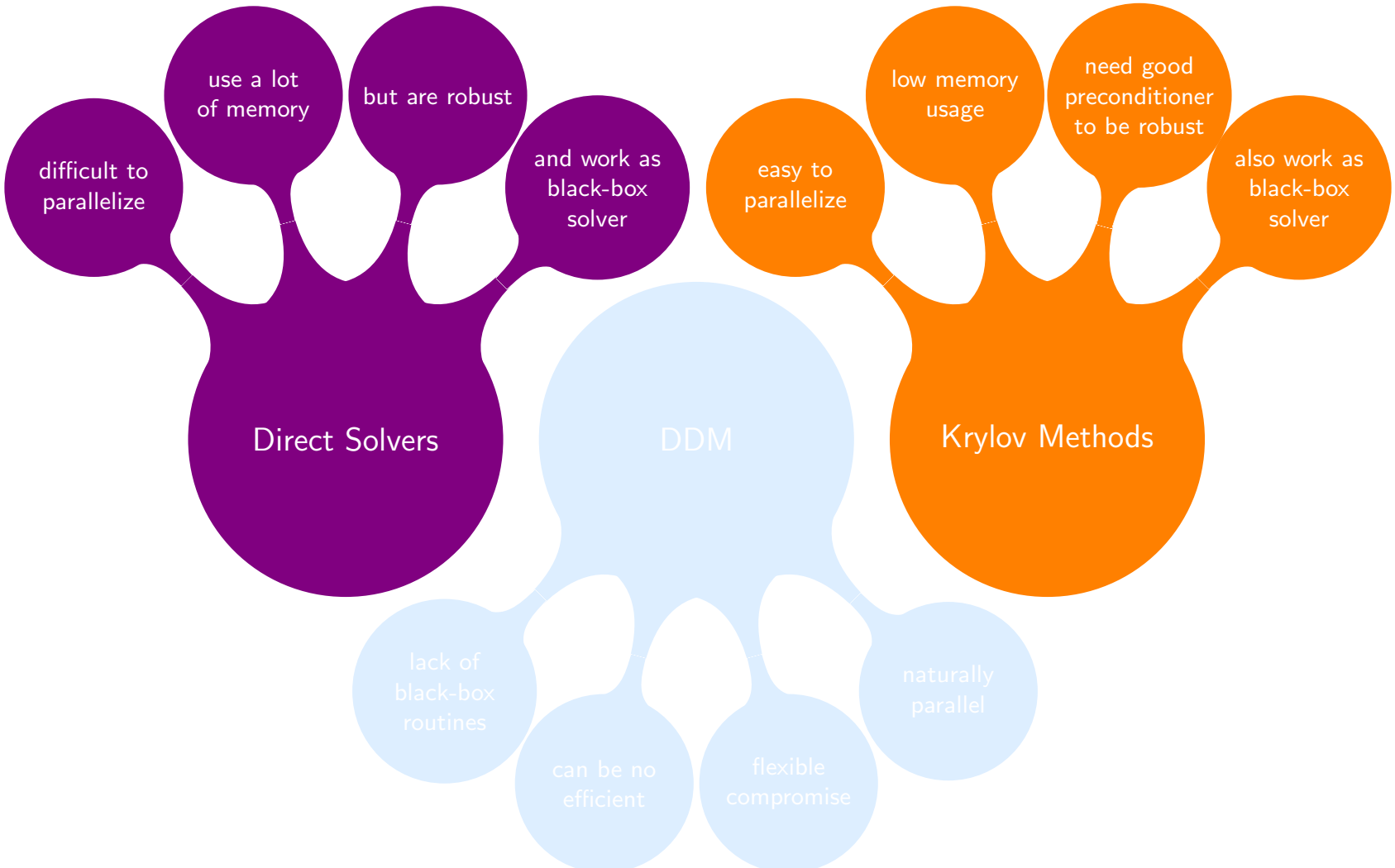
Why Domain Decomposition Method ?

- The term Domain Decomposition has slightly different meaning to specialist within the discipline of PDEs.
  - process of distributing data among the processors
  - process of subdividing the solution of large linear system into smaller problem
- Ease of parallelization
  - parallel processing is one way to have a faster codes
  - new generation processors are parallel (multi cores)
- In some situation, the domain decomposition is natural
  - strong heterogeneous media
  - different physics in different subdomains

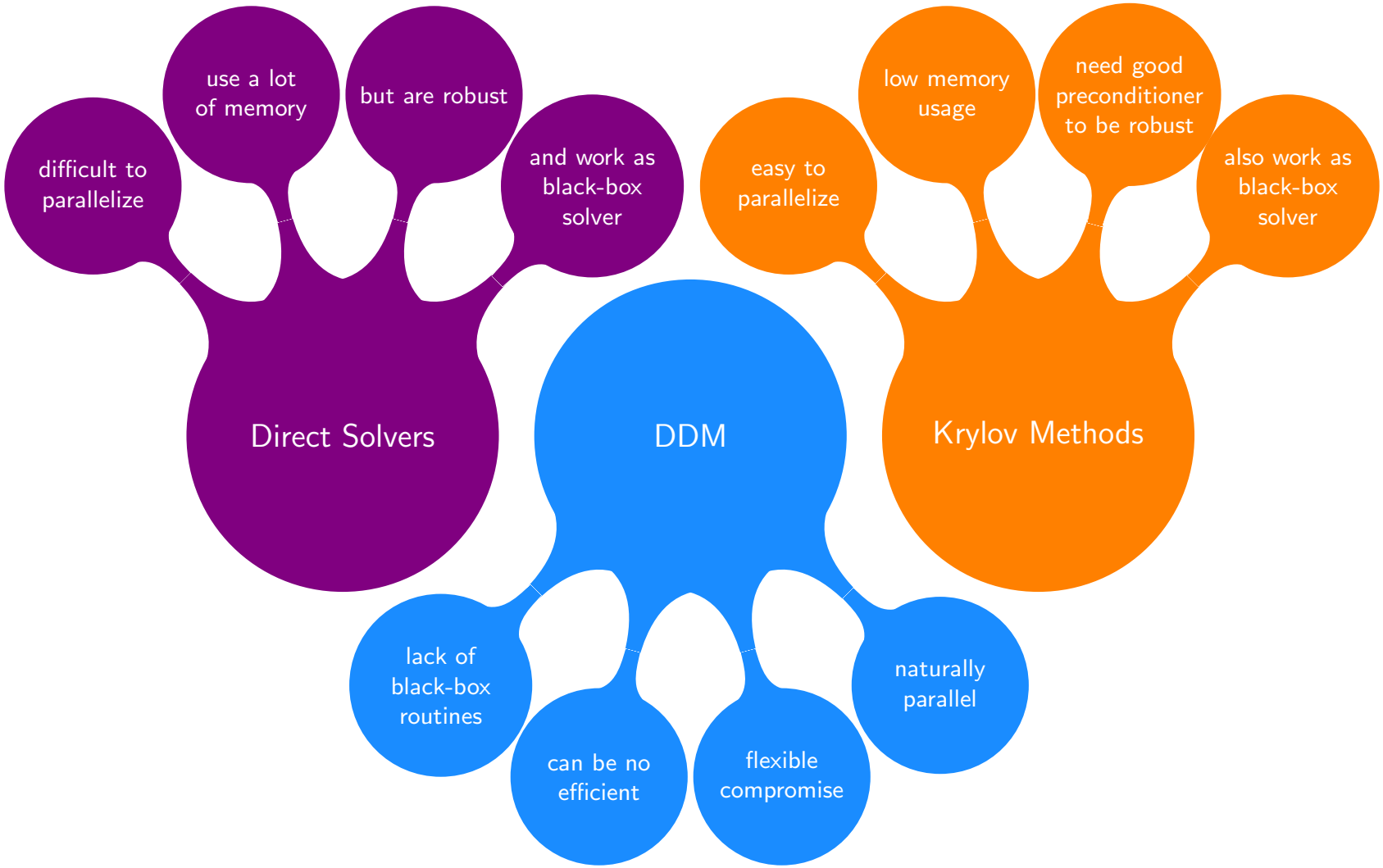
DDM vs. other methods



DDM vs. other methods

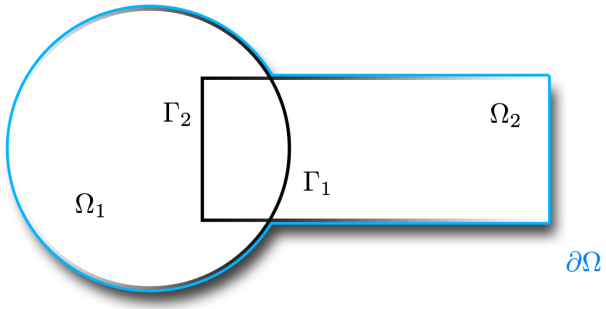


DDM vs. other methods



State of the Arts

Original Method



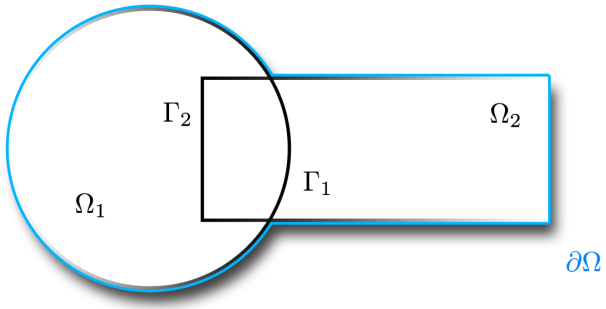
$$\begin{aligned} -\Delta(u) &= f \quad \text{in} \quad \Omega \\ u &= g \quad \text{on} \quad \partial\Omega \end{aligned}$$

Alternating Schwarz Method

$$\begin{aligned} -\Delta(u_1^{n+1}) &= f \quad \text{in} \quad \Omega_1 \\ u_1^{n+1} &= g \quad \text{on} \quad \partial\Omega_1 \backslash \Gamma_1 \\ u_1^{n+1} &= u_2^n \quad \text{on} \quad \Gamma_1. \end{aligned} \qquad \begin{aligned} -\Delta(u_2^{n+1}) &= f \quad \text{in} \quad \Omega_2 \\ u_2^{n+1} &= g \quad \text{on} \quad \partial\Omega_2 \backslash \Gamma_2 \\ u_2^{n+1} &= u_1^{n+1} \quad \text{on} \quad \Gamma_2. \end{aligned}$$

"As $n \to \infty$, $(u_1^n, u_2^n) \to (u_{sol|\Omega_1}^n, u_{sol|\Omega_2}^n)$, where u_{sol} is a solution of continuous problem [Schwarz, 1870]."

Parallel Schwarz Method



$$\begin{aligned} -\Delta(u) &= f \quad \text{in} \quad \Omega \\ u &= g \quad \text{on} \quad \partial\Omega \end{aligned}$$

Parallel Schwarz Method

$$\begin{aligned} -\Delta(u_1^{n+1}) &= f \quad \text{in} \quad \Omega_1 \\ u_1^{n+1} &= g \quad \text{on} \quad \partial\Omega_1 \backslash \Gamma_1 \\ u_1^{n+1} &= u_2^n \quad \text{on} \quad \Gamma_1. \end{aligned} \qquad \begin{aligned} -\Delta(u_2^{n+1}) &= f \quad \text{in} \quad \Omega_2 \\ u_2^{n+1} &= g \quad \text{on} \quad \partial\Omega_2 \backslash \Gamma_2 \\ u_2^{n+1} &= u_1^n \quad \text{on} \quad \Gamma_2. \end{aligned}$$

Drawbacks of original methods

Original algorithms:

- are parallel but converges slowly
- need overlap in order to converge
- convergence speed depend on size of overlap

Improvements:

- Schwarz methods as a precondition for Krylov methods
- more general interface conditions

All of them can be apply at the algebraic level !

Example: The condition number κ of operator A, preconditioned by $\mathcal{P}_{as}$ i.e., ASM with the coarse grid correction, satisfies

$$\kappa(\mathcal{P}_{as}A) \leq C\left(1+\frac{H}{\delta}\right),$$

where the constant C is independent of, H and δ.

Algebraic Formulation (Jacobi and Schwarz)

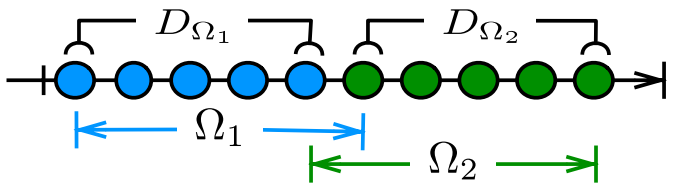
Lets consider a discretized problem which yields a linear system.

$$\begin{aligned} -\Delta(u) &= f \quad \text{in} \quad \Omega \\ u &= g \quad \text{on} \quad \partial\Omega \end{aligned}$$

$$AU = F$$

For the set of indices Ω partitioned into two sets Ω_1 and Ω_2 we have:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$$



The **block-Jacobi algorithm** reads:

$$\begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \end{bmatrix} = \begin{bmatrix} U_1^{n} \\ U_2^{n} \end{bmatrix} + \begin{bmatrix} A_{11}^{-1} & 0 \\ 0 & A_{22}^{-1} \end{bmatrix} \left(\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} - \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} U_1^{n} \\ U_2^{n} \end{bmatrix} \right)$$

It corresponding to solving a Dirichlet boundary value problem in each subdomain with Dirichlet data taken from the other one at the previous step $\Longrightarrow$ **Schwarz method with minimal overlap**.

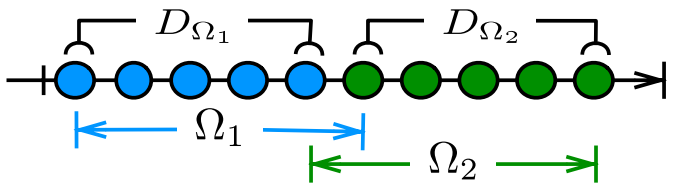
Algebraic Formulation (Jacobi and Schwarz)

Lets consider a discretized problem which yields a linear system.

$$\begin{aligned} -\Delta(u) &= f \quad \text{in} \quad \Omega \\ u &= g \quad \text{on} \quad \partial\Omega \end{aligned} \qquad AU = F$$

For the set of indices Ω partitioned into two sets Ω_1 and Ω_2 we have:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$$



The **block-Jacobi algorithm** reads:

$$\begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \end{bmatrix} = \begin{bmatrix} U_1^{n} \\ U_2^{n} \end{bmatrix} + \begin{bmatrix} A_{11}^{-1} & 0 \\ 0 & A_{22}^{-1} \end{bmatrix} \left(\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} - \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} U_1^{n} \\ U_2^{n} \end{bmatrix} \right)$$

It corresponding to solving a Dirichlet boundary value problem in each subdomain with Dirichlet data taken from the other one at the previous step $\Longrightarrow$ **Schwarz method with minimal overlap**.

Preconditioned Krylov space

$$\underbrace{\begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \end{bmatrix} = \begin{bmatrix} U_1^{n} \\ U_2^{n} \end{bmatrix} + \begin{bmatrix} A_{11}^{-1} & 0 \\ 0 & A_{22}^{-1} \end{bmatrix} \left(\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} - \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} U_1^{n} \\ U_2^{n} \end{bmatrix} \right)}_{\mathbf{u}_{n+1} = \mathbf{u}_n + M^{-1}(\mathbf{f} - A\mathbf{u}_n)}$$

Let $\mathbf{r}_0 := M^{-1}(\mathbf{f} - A\mathbf{u}_0)$, we have (fixed point method):

$$\mathbf{u}_n = \sum_{i=0}^{n} \left(\mathbb{I} - M^{-1}A\right)^i \mathbf{r}_0 + \mathbf{u}_0$$

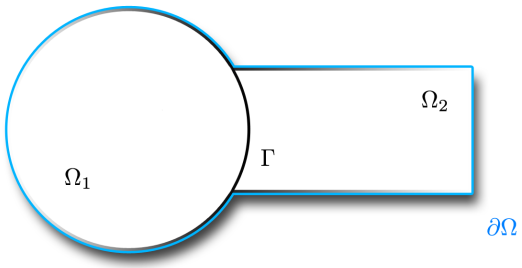
A preconditioned Krylov solve will generate an optimal solution $\mathbf{u}_n^{\mathcal{K}}$ in:

$$\mathcal{K}_m\left(M^{-1}A, \mathbf{r}_0\right) := \mathbf{u}_0 + \mathtt{SPAN}\left\{\mathbf{r}_0, M^{-1}A\mathbf{r}_0, \ldots, (M^{-1}A)^{m-1}\mathbf{r}_0\right\}$$

where $\mathbf{u}_n \in \mathcal{K}_m$ but with "frozen" coefficients $\Longrightarrow$ **$\mathbf{u}_n$ is less optimal then $\mathbf{u}_n^{\mathcal{K}}$.**

Modified Schwarz Method

Another improvement arise from usage of more general interface conditions for a non-overlapping ($\Longleftrightarrow \ \alpha > 0$) decomposition [Lions, 1990]:



$$\begin{aligned} -\Delta(u) &= f \quad \text{in} \quad \Omega \\ u &= g \quad \text{on} \quad \partial\Omega \end{aligned} \tag{1}$$

Modified Schwarz Method

$$\begin{aligned} -\Delta(u_1^{n+1}) &= f && \text{in} && \Omega_1 \\ u_1^{n+1} &= g && \text{on} && \partial\Omega_1 \backslash \Gamma \\ \left(\frac{\partial}{\partial n_1} + \alpha\right)(u_1^{n+1}) &= \left(\frac{\partial}{\partial n_2} + \alpha\right)(u_n^{n+1}) && \text{on} && \partial\Omega_1 \cap \overline{\Omega}_2 \\ -\Delta(u_2^{n+1}) &= f && \text{in} && \Omega_2 \\ u_2^{n+1} &= g && \text{on} && \partial\Omega_2 \cap \partial\Gamma \\ \left(\frac{\partial}{\partial n_2} + \alpha\right)(u_2^{n+1}) &= \left(\frac{\partial}{\partial n_1} + \alpha\right)(u_n^{n+1}) && \text{on} && \partial\Omega_2 \cap \overline{\Omega}_1 \end{aligned}$$

Optimal Choice

J. L. Lions:

"First of all, it is possible to replace the constants in the Robin condition by two proportional function on the interface, or even by local or nonlocal operators [Lions, 1990]."

F. Nataf, F. Rogier and E. de Sturler:

"The rate of convergence of Schwarz and Schur type algorithms is very sensitive to the choice of interface condition. The original Schwarz method is based on the use of Dirichlet boundary conditions. In order to increase the efficiency of the algorithm, it has been proposed to replace the Dirichlet boundary condition with more general boundary conditions. . . . It has been remarked that absorbing (or artificial) boundary conditions are a good choice. In this report, we try to clarify the question of the interface condition [Nataf et al., 1994]."

Optimal Choice

J. L. Lions:

> *"First of all, it is possible to replace the constants in the Robin condition by two proportional function on the interface, or even by local or nonlocal operators [Lions, 1990]."*

F. Nataf, F. Rogier and E. de Sturler:

> *"The rate of convergence of Schwarz and Schur type algorithms is very sensitive to the choice of interface condition. The original Schwarz method is based on the use of Dirichlet boundary conditions. In order to increase the efficiency of the algorithm, it has been proposed to replace the Dirichlet boundary condition with more general boundary conditions. . . . It has been remarked that absorbing (or artificial) boundary conditions are a good choice. In this report, we try to clarify the question of the interface condition [Nataf et al., 1994]."*

Optimal Choice

The authors proved that use of non-local DtN (*Dirichlet to Neumann*) map (a.k.a. *Steklov-Poincaré*) as interface condition in problem (1) leads to (*exact*) convergence in two iterations.

Definition (DtN map)

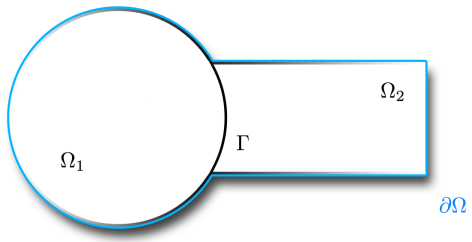
Let

$$u_0 : \Gamma_1 \to \mathbb{R}$$
$$\mathrm{DtN}_2(u_0) := \nabla v \cdot n_{2|\partial\Omega_1 \cap \overline{\Omega}_2},$$

where n_2 is the outward normal to $\Omega_2 \setminus \overline{\Omega}_1$, and v satisfies the following boundary value problem:

$$\begin{aligned} \mathcal{L}(v) &= 0 \quad \text{in} \quad \Omega_2 \setminus \overline{\Omega}_1 \\ v &= 0 \quad \text{on} \quad \partial\Omega_2 \cap \partial\Omega \\ v &= u_0 \quad \text{on} \quad \partial\Omega_1 \cap \overline{\Omega}_2. \end{aligned}$$

Optimal Interface Condition at the matrix level



$$\begin{bmatrix} A_{11} & A_{1\Gamma} & 0 \\ A_{\Gamma 1} & A_{\Gamma\Gamma} & A_{\Gamma 2} \\ 0 & A_{2\Gamma} & A_{22} \end{bmatrix} \begin{bmatrix} U_1 \\ U_\Gamma \\ U_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ F_\Gamma \\ F_2 \end{bmatrix}$$

In order to write a "modified" Schwarz method we need to introduce two square matrixes S_1 and S_2 which acts on vector of the type U_Γ:

$$\begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & A_{\Gamma\Gamma} + S_2 \end{pmatrix} \begin{pmatrix} U_1^{n+1} \\ U_{\Gamma,1}^{n+1} \end{pmatrix} = \begin{pmatrix} F_1 \\ F_\Gamma + S_2 U_{\Gamma,2}^n - A_{\Gamma 2} U_2^n \end{pmatrix}$$

$$\begin{pmatrix} A_{22} & A_{2\Gamma} \\ A_{\Gamma 2} & A_{\Gamma\Gamma} + S_1 \end{pmatrix} \begin{pmatrix} U_2^{n+1} \\ U_{\Gamma,2}^{n+1} \end{pmatrix} = \begin{pmatrix} F_2 \\ F_\Gamma + S_1 U_{\Gamma,1}^n - A_{\Gamma 1} U_1^n \end{pmatrix}$$

Lemma

If $A_{\Gamma\Gamma} + S_1 + S_2$ is invertible and problem (1) is well-posed. Then above algorithm converges to the solution of (1) $\Longrightarrow$ $U_i^\infty = U_i$ and $U_{\Gamma,1}^\infty = U_{\Gamma,2}^\infty = U_\Gamma$.

Optimal Interface Condition at the matrix level

$$\begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & A_{\Gamma\Gamma} + S_2 \end{pmatrix} \begin{pmatrix} U_1^{n+1} \\ U_{\Gamma,1}^{n+1} \end{pmatrix} = \begin{pmatrix} F_1 \\ F_\Gamma + S_2 U_{\Gamma,2}^n - A_{\Gamma 2} U_2^n \end{pmatrix}$$

$$\begin{pmatrix} A_{22} & A_{2\Gamma} \\ A_{\Gamma 2} & A_{\Gamma\Gamma} + S_1 \end{pmatrix} \begin{pmatrix} U_2^{n+1} \\ U_{\Gamma,2}^{n+1} \end{pmatrix} = \begin{pmatrix} F_2 \\ F_\Gamma + S_1 U_{\Gamma,1}^n - A_{\Gamma 1} U_1^n \end{pmatrix}$$

Optimal choice

Taking $S_1 = -A_{\Gamma 1} A_{11}^{-1} A_{1\Gamma}$ and $S_2 = -A_{\Gamma 1} A_{22}^{-1} A_{2\Gamma}$ yields a convergence in two steps $\Longrightarrow$ $A_{\Gamma\Gamma} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma}$ is a Schur complement.

The matrices S_1 and S_2 are full, therefore

- they are costly to build
- the subdomain matrix is partially full

However it is possible to approximate them by sparse matrices e.g., via local Schur complement on successive reduced "outer" domain, which we call **patches** [Magoulès et al., 2006].

Optimal Interface Condition at the matrix level

$$\begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & A_{\Gamma\Gamma} + S_2 \end{pmatrix} \begin{pmatrix} U_1^{n+1} \\ U_{\Gamma,1}^{n+1} \end{pmatrix} = \begin{pmatrix} F_1 \\ F_\Gamma + S_2 U_{\Gamma,2}^n - A_{\Gamma 2} U_2^n \end{pmatrix}$$

$$\begin{pmatrix} A_{22} & A_{2\Gamma} \\ A_{\Gamma 2} & A_{\Gamma\Gamma} + S_1 \end{pmatrix} \begin{pmatrix} U_2^{n+1} \\ U_{\Gamma,2}^{n+1} \end{pmatrix} = \begin{pmatrix} F_2 \\ F_\Gamma + S_1 U_{\Gamma,1}^n - A_{\Gamma 1} U_1^n \end{pmatrix}$$

Optimal choice

Taking $S_1 = -A_{\Gamma 1} A_{11}^{-1} A_{1\Gamma}$ and $S_2 = -A_{\Gamma 1} A_{22}^{-1} A_{2\Gamma}$ yields a convergence in two steps $\Longrightarrow$ $A_{\Gamma\Gamma} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma}$ is a Schur complement.

The matrices S_1 and S_2 are full, therefore

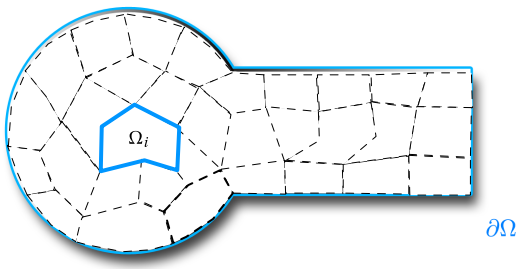
- they are costly to build
- the subdomain matrix is partially full

However it is possible to approximate them by sparse matrices e.g., via local Schur complement on successive reduced "outer" domain, which we call **patches** [Magoulès et al., 2006].

Schwarz method vs. Many subdomains

It is well known that performance may deteriorate with large number of subdomains i.e., plateaus appear in the convergence of the Krylov methods.

They are due to the lack of a global exchange of information in the preconditioner.



$$\begin{aligned} -\Delta(u) &= f \quad \text{in} \quad \Omega \\ u &= g \quad \text{on} \quad \partial\Omega \end{aligned}$$

The mean value of the solution in domain Ω_i depends on the value of f on all subdomains.

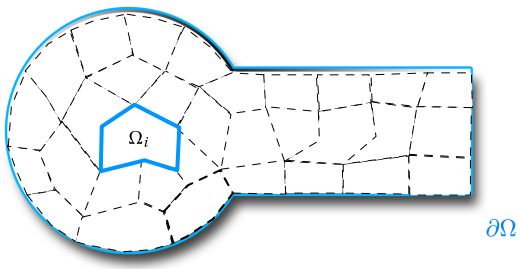
A classical remedy: $\Longrightarrow$ coarse grid problem that couples all subdomains.

- can be incorporate as additional preconditioner $\Longrightarrow$ "two-level preconditioning".

Schwarz method vs. Many subdomains

It is well known that performance may deteriorate with large number of subdomains i.e., plateaus appear in the convergence of the Krylov methods.

They are due to the lack of a global exchange of information in the preconditioner.



$$\begin{aligned} -\Delta(u) &= f \quad \text{in} \quad \Omega \\ u &= g \quad \text{on} \quad \partial\Omega \end{aligned}$$

The mean value of the solution in domain Ω_i depends on the value of f on all subdomains.

A classical remedy: $\Longrightarrow$ coarse grid problem that couples all subdomains.

- can be incorporate as additional preconditioner $\Longrightarrow$ "two-level preconditioning".

Two-level preconditioner

From an abstract point of view, all two-level preconditioners of the method consists of an arbitrary preconditioner M, combined with one or more matrices P and Q.

$$P := \mathbb{I} - AQ, \quad Q := ZE^{-1}Z^T, \quad E := Z^T AZ$$

Some properties:

- $PA = AP^T$
- $P^T Z = \mathbf{0}$, $P^T Q$
- $QA = \mathbb{I} - P^T$, $QAZ = Z$, $QAQ = Q$

$A, M, P, Q \in \mathbb{R}^{n \times n}$
$Z \in \mathbb{R}^{n \times m}$
$E \in \mathbb{R}^{m \times m}$, $m \ll n$

The matrix Z consists of so-called projection vectors, whose columns span the projection space (More detail in [Tang et al., 2009]).

Example: $\mathcal{P}_{AD} := M^{-1} + Q$, $\mathcal{P}_{BNN} := P^T M^{-1} P + Q$, $\mathcal{P}_{A-DEF} := P^T M^{-1} + Q$

Two-level preconditioner

From an abstract point of view, all two-level preconditioners of the method consists of an arbitrary preconditioner M, combined with one or more matrices P and Q.

$$P := \mathbb{I} - AQ, \quad Q := ZE^{-1}Z^T, \quad E := Z^T AZ$$

Some properties:

- $PA = AP^T$
- $P^T Z = \mathbf{0}$, $P^T Q$
- $QA = \mathbb{I} - P^T$, $QAZ = Z$, $QAQ = Q$

$A, M, P, Q \in \mathbb{R}^{n \times n}$
$Z \in \mathbb{R}^{n \times m}$
$E \in \mathbb{R}^{m \times m}$, $m \ll n$

The matrix Z consists of so-called projection vectors, whose columns span the projection space (More detail in [Tang et al., 2009]).

Example: $\mathcal{P}_{AD} := M^{-1} + Q$, $\mathcal{P}_{BNN} := P^T M^{-1} P + Q$, $\mathcal{P}_{A-DEF} := P^T M^{-1} + Q$

What we know so far?

Schwarz methods:

- are very suitable for parallel computing
- are easy to use at the algebraic level
  - and its iterative process can by accelerated by Krylov methods
- they also "work" with general interface conditions
  - for which we know "optimal" choice
- we can modify interface conditions at the algebraic level
- in case of many subdomains, we can incorporate Schwarz preconditioner with the coarse grid correction
  - in order to construct two-level preconditioner

New Methods and Techniques

ADDMlib - parallel library

ADDMlib - Algebraic Domain Decomposition Methods (library)

- carefully design object oriented library
- written in modern C++ (Boost + STL)
- provides (via MPI) many of the mechanism needed within parallel application code
  - parallel vectors and sparse matrices in several sparse formats
- we put stress on:
  - easiness of implementing preconditioners of DDM type
  - Krylov subspace methods (GMRES,FGMRES,BiCGstab)
- it provides convenient interface for chosen functionality from other libraries like *METIS*, *SCOTCH* or *PETSc*

ADDMlib - parallel library

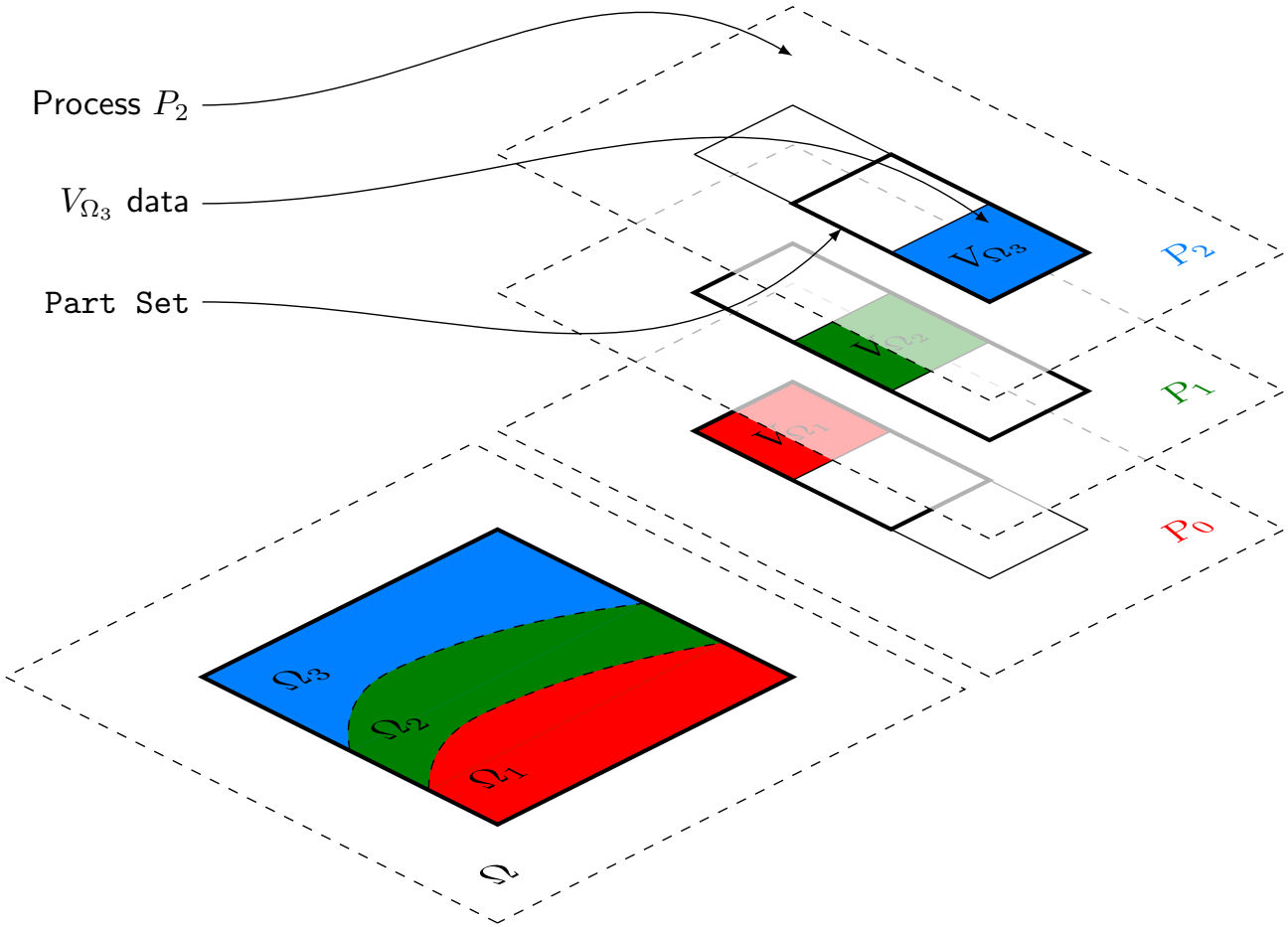


Figure: `DDMVector` structure and its division into `Partial Vectors` according to decomposition of domain Ω.

ADDMlib - parallel library

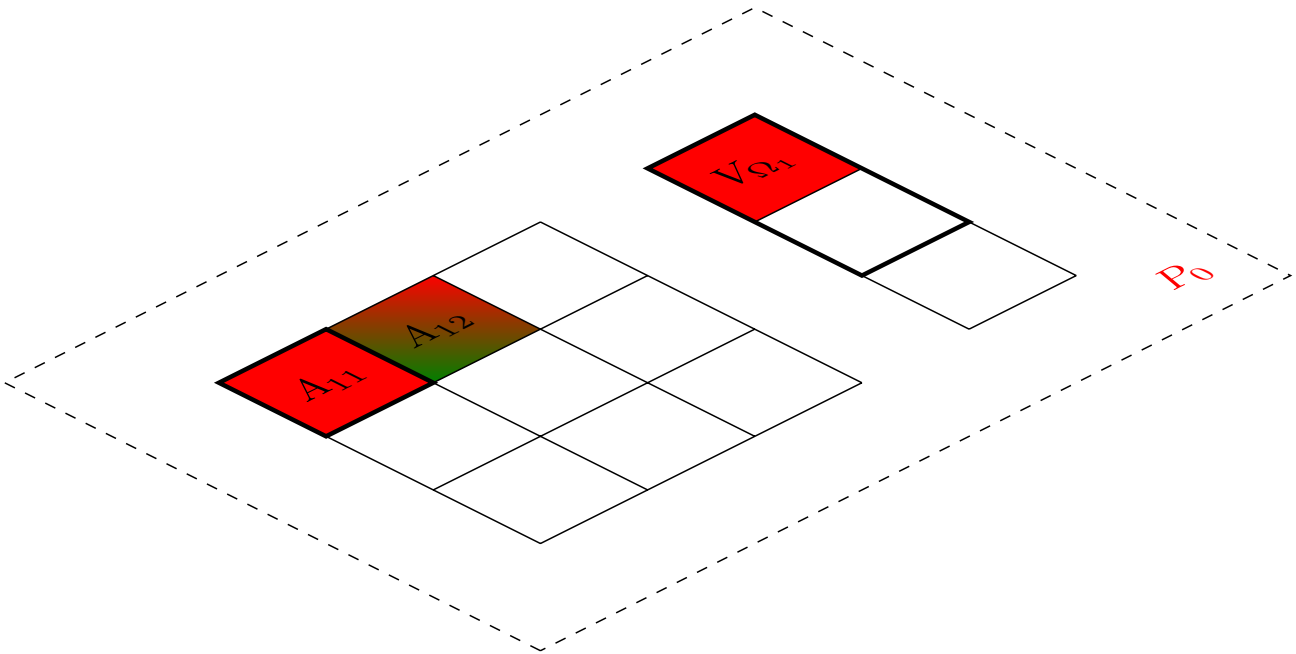


Figure: Decomposition of global linear system into `Partial Vectors` and `Operators` (very similar idea introduced independently in [Buluç et al., 2009]).

ADDMlib - parallel library

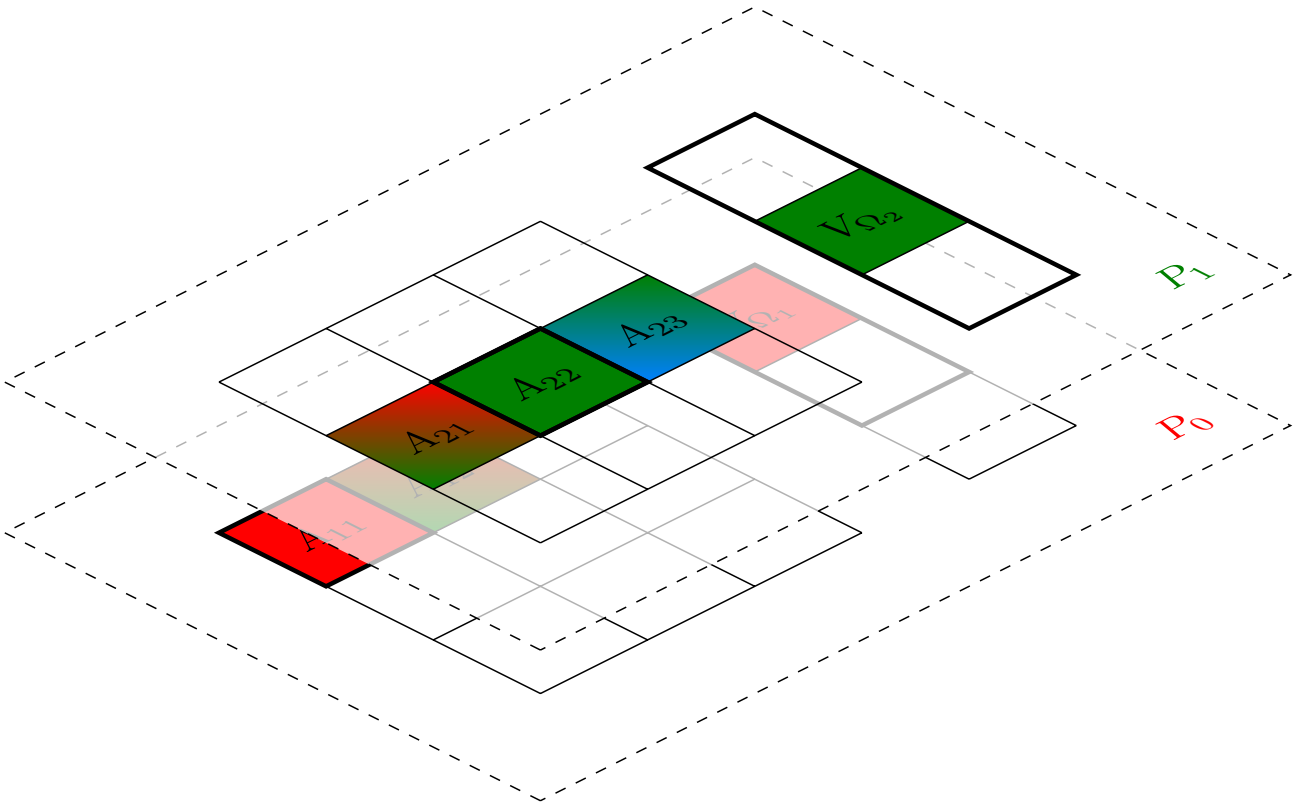


Figure: Decomposition of global linear system into `Partial Vectors` and `Operators` (very similar idea introduced independently in [Buluç et al., 2009]).

ADDMlib - parallel library

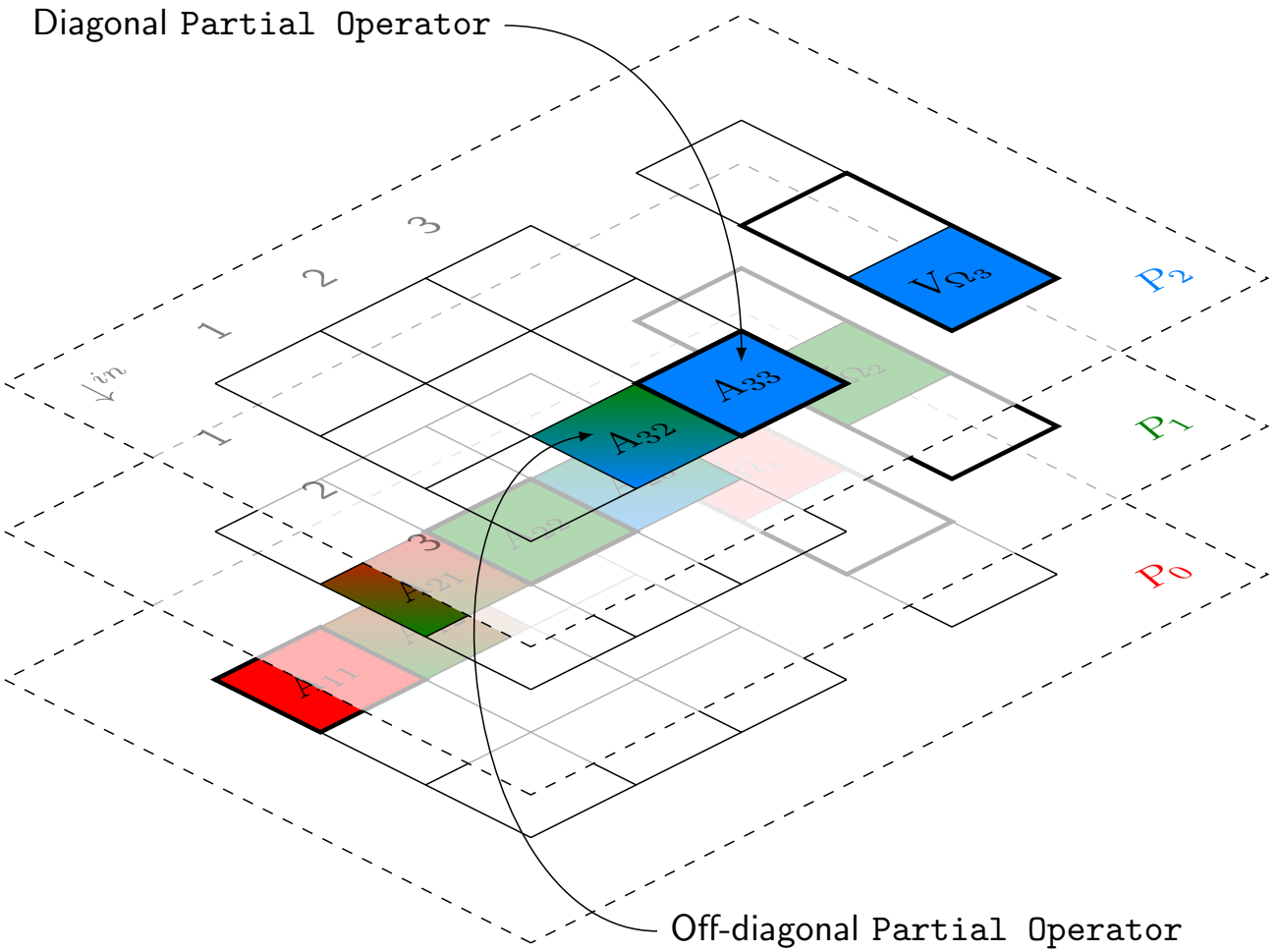


Figure: Decomposition of global linear system into `Partial Vectors` and `Operators` (very similar idea introduced independently in [Buluç et al., 2009]).

Partitioning

But how to subdivide and map data into processors ?

- in arbitrary fashion
- or we can use adjacency graph partitioners (SCOTCH, METIS)

Definition (Graph Partitioning)

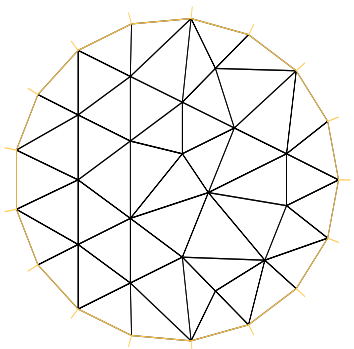
For a general sparse linear system whose adjacency graph is $G = (V, E)$, the k-way graph partitioning problem is defined as follows: given a graph $G = (V, E)$ with $|V| = n$, partition V into k subsets, $V_1, V_2, \ldots, V_k$ such that $V_i \cap V_j = \emptyset$ for $i \neq j$, $|V_i| = n/k$, and $\cup_i V_i = V$, and the number of edges of E whose incident vertices belong to different subset is minimized.

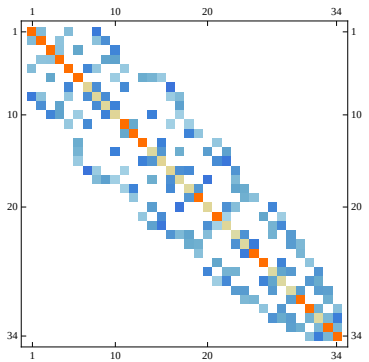
During our experiments we have noticed that the way how the adjacency graph is partitioned has strong influence on overall performance of algebraic DDM.

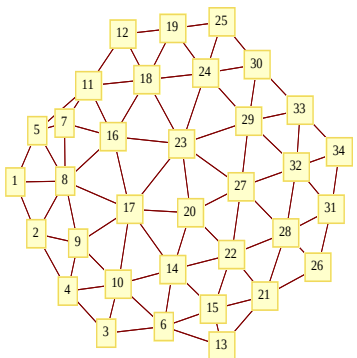
Partitioning

But how to subdivide and map data into processors ?

- in arbitrary fashion
- or we can use adjacency graph partitioners (SCOTCH, METIS)







During our experiments we have noticed that the way how the adjacency graph is partitioned has strong influence on overall performance of algebraic DDM.

Partitioning

But how to subdivide and map data into processors ?

- in arbitrary fashion
- or we can use adjacency graph partitioners (SCOTCH, METIS)

Definition (Graph Partitioning)

For a general sparse linear system whose adjacency graph is $G = (V, E)$, the k-way graph partitioning problem is defined as follows: given a graph $G = (V, E)$ with $|V| = n$, partition V into k subsets, $V_1, V_2, \ldots, V_k$ such that $V_i \cap V_j = \emptyset$ for $i \neq j$, $|V_i| = n/k$, and $\cup_i V_i = V$, and the number of edges of E whose incident vertices belong to different subset is minimized.

During our experiments we have noticed that the way how the adjacency graph is partitioned has strong influence on overall performance of algebraic DDM.

Partitioning

But how to subdivide and map data into processors ?

- in arbitrary fashion
- or we can use adjacency graph partitioners (SCOTCH, METIS)

Definition (Graph Partitioning)

For a general sparse linear system whose adjacency graph is $G = (V, E)$, the k-way graph partitioning problem is defined as follows: given a graph $G = (V, E)$ with $|V| = n$, partition V into k subsets, $V_1, V_2, \ldots, V_k$ such that $V_i \cap V_j = \emptyset$ for $i \neq j$, $|V_i| = n/k$, and $\cup_i V_i = V$, and the number of edges of E whose incident vertices belong to different subset is minimized.

During our experiments we have noticed that the way how the adjacency graph is partitioned has strong influence on overall performance of algebraic DDM.

Partitioning with weights

There is a certain number of problems for which "smart" partitioning can increase robustness (e.g., anisotropic problems)

Is it possible to extract algebraically some information about physical properties of the problem to solve, and use them to obtain better partition ?

. . . yes we can define weights for edges of adjacency graph using values of the underlaying matrix using following formula adapted from AMG methods (see for example [Stüben, 2001])

Automatic weight labelling

$$c = \left\lfloor \left(\frac{|a_{ij}|}{|a_{ii}| + |a_{jj}|} \times \gamma_{const} \right) \right\rfloor$$

$\lfloor x \rfloor$ is the floor function rounds the element x to the nearest integer toward minus infinity and γ_{const} is an arbitrary constant.

Partitioning with weights

There is a certain number of problems for which "smart" partitioning can increase robustness (e.g., anisotropic problems)

Is it possible to extract algebraically some information about physical properties of the problem to solve, and use them to obtain better partition ?

. . . yes we can define weights for edges of adjacency graph using values of the underlaying matrix using following formula adapted from AMG methods (see for example [Stüben, 2001])

Automatic weight labelling

$$c = \left\lfloor \left(\frac{|a_{ij}|}{|a_{ii}| + |a_{jj}|} \times \gamma_{const} \right) \right\rfloor$$

$\lfloor x \rfloor$ is the floor function rounds the element x to the nearest integer toward minus infinity and γ_{const} is an arbitrary constant.

Partitioning with weights - Numerical Experiment 1

Let us consider following, anisotropic problem: $-\kappa\Delta(\mathbf{u}) = f$, discretized (FreeFem++) on 2D unit square in size $N_x \times N_y$, where $N_x = N_y = 128$

$$\kappa = \begin{bmatrix} \kappa_{xx} & 0 \\ 0 & \kappa_{yy} \end{bmatrix} = \begin{bmatrix} 1 \times 10^{-6} & 0 \\ 0 & 1 \end{bmatrix}$$

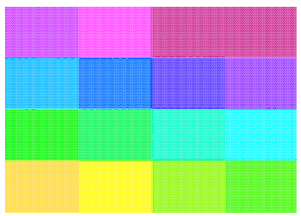


Figure: a

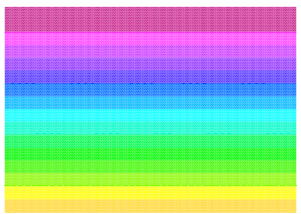


Figure: b

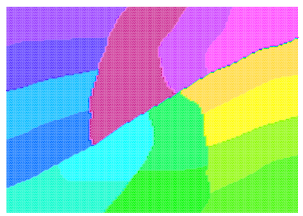


Figure: c

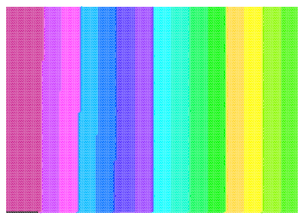
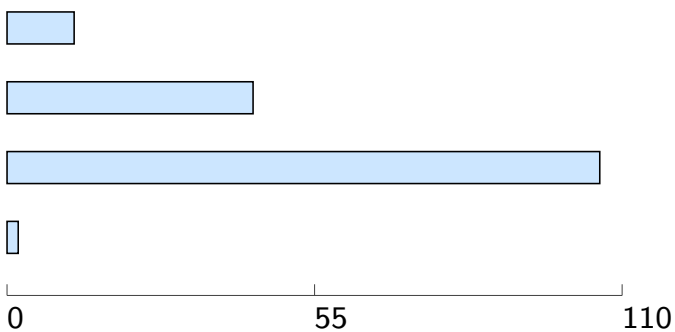
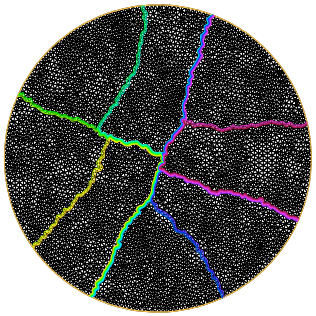


Figure: d

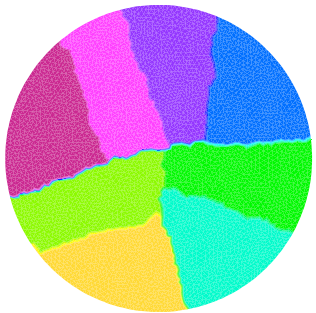
Paritioner	*n-iter*
(a) Manual	12
(b) Manual	44
(e) SCOTCH	106
(d) SCOTCH + W	**2**



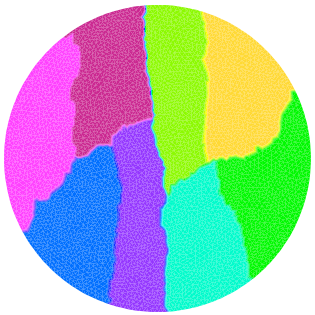
Partitioning with weights - Numerical Experiment 2



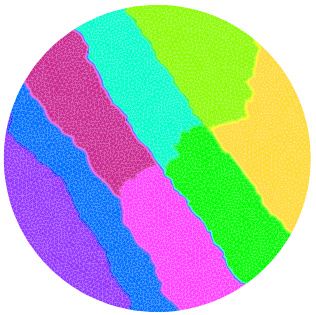
(⋆)



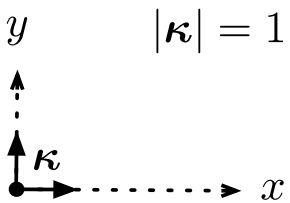
(b)

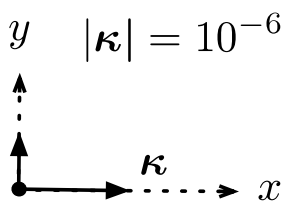


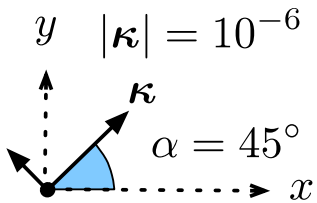
(d)



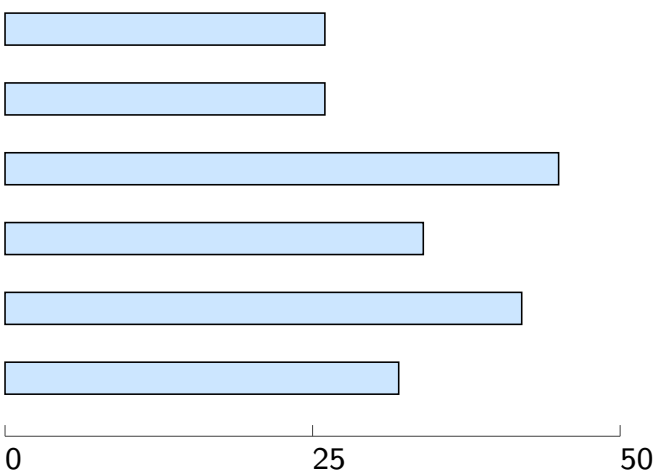
(f)







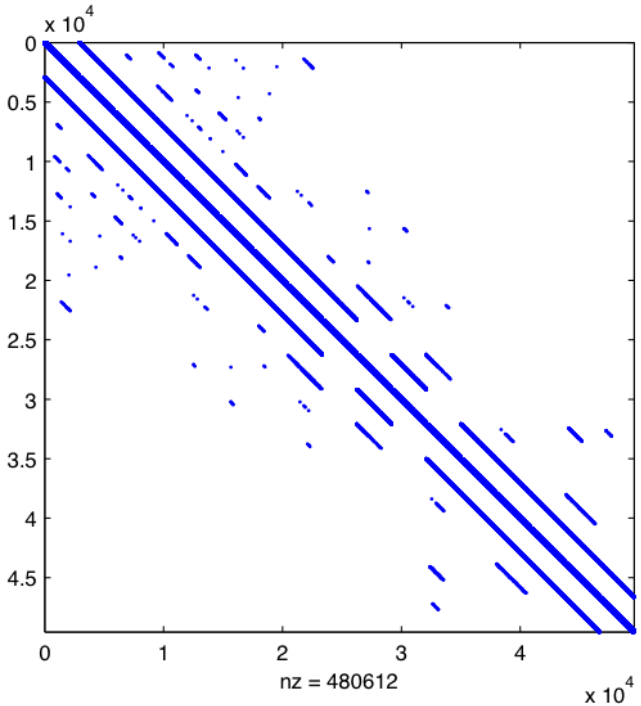
Paritioner	*n-iter*
(⋆) SCOTCH	26
(b) SCOTCH + W	26
(⋆) SCOTCH	45
(d) SCOTCH + W	34
(⋆) SCOTCH	42
(f) SCOTCH + W	32



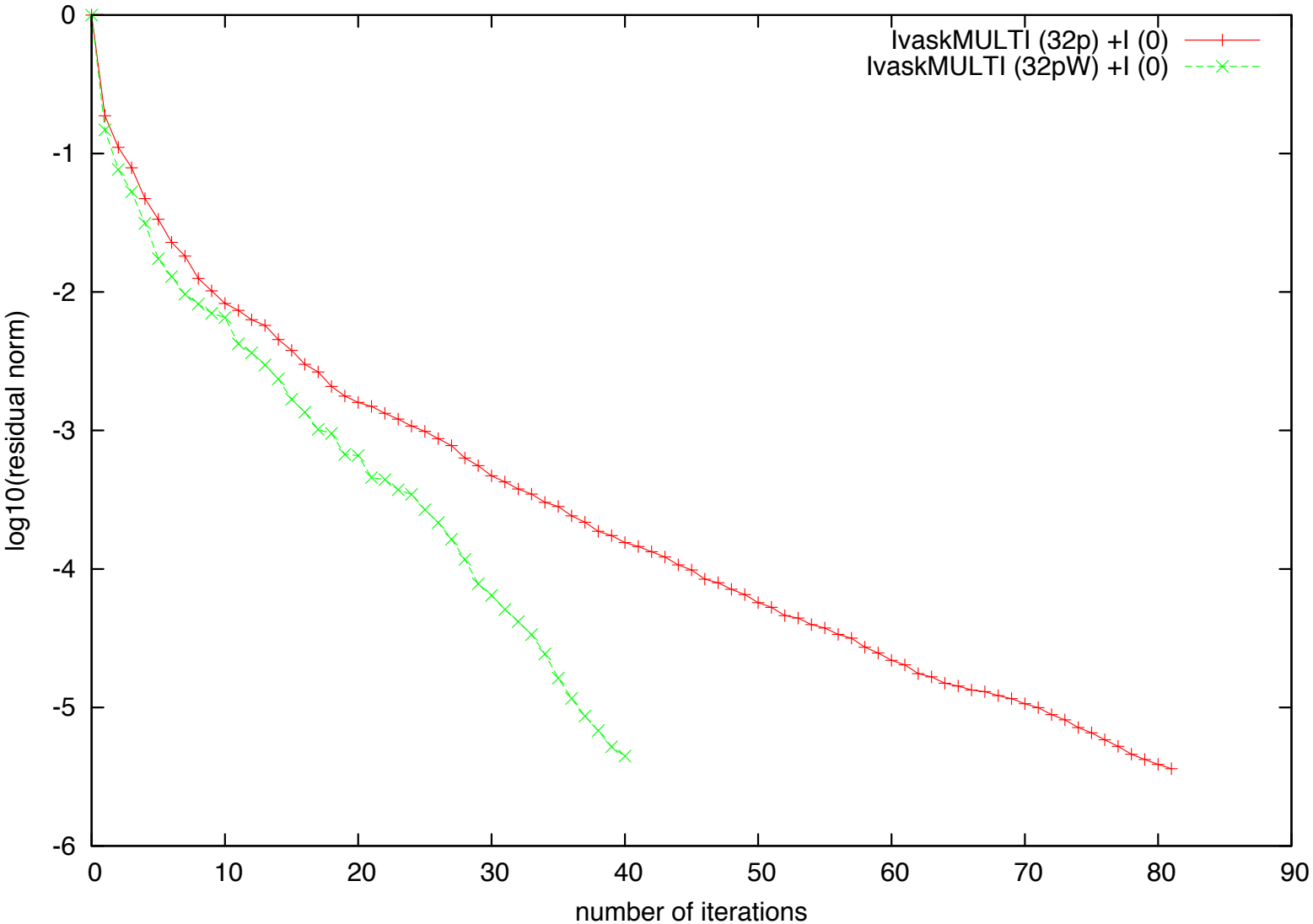
Partitioning with weights - real test case

Matrix IvaskMULTI_p_only.mtx

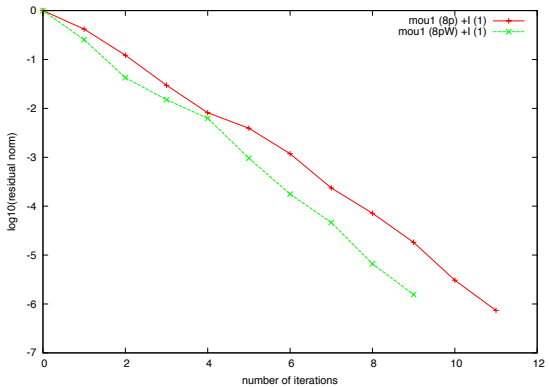
nparts	nrows	nnz
32	49,572	480,612

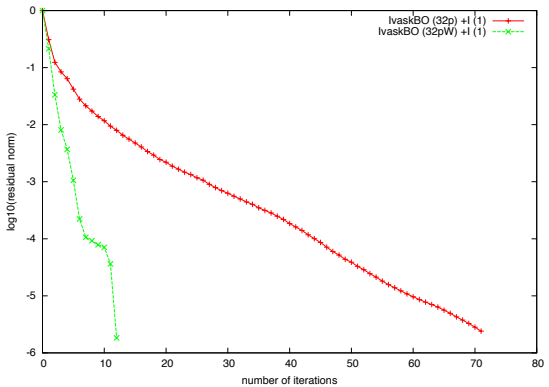


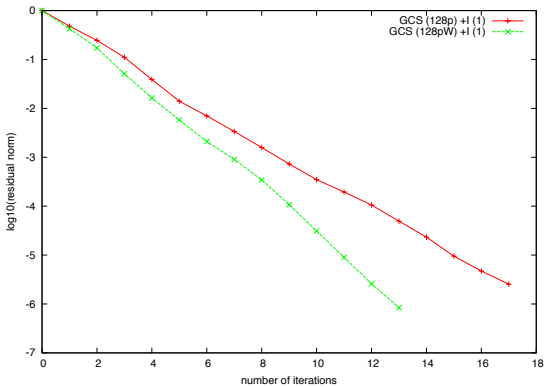
Partitioning with weights - real test case

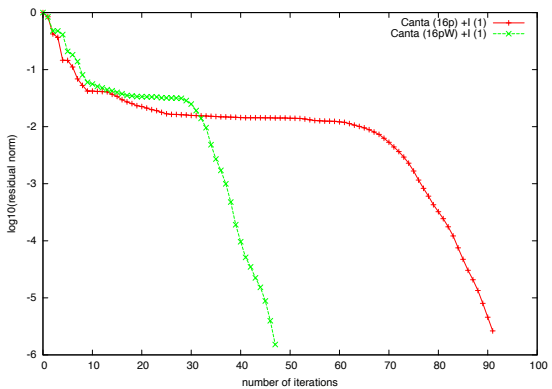


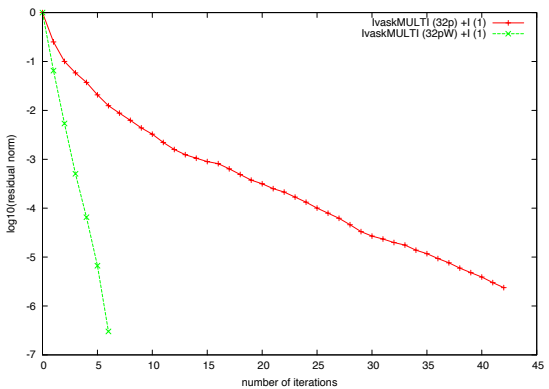
Partitioning with weights - real test case (ALL)

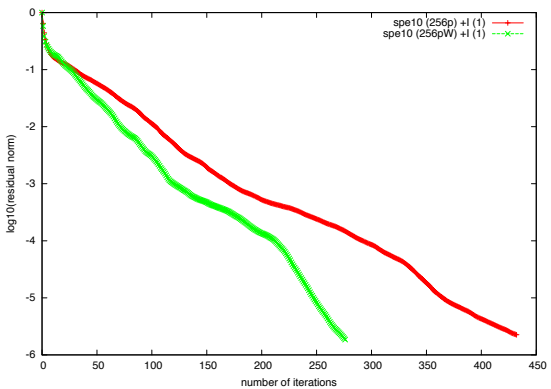






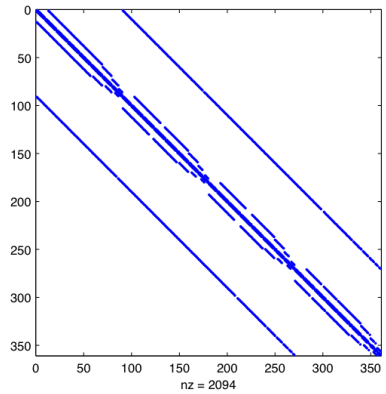


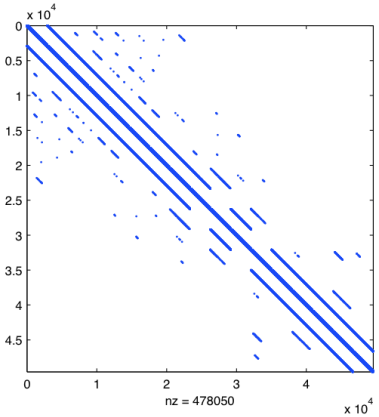


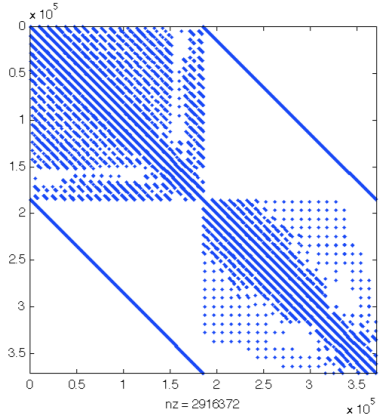


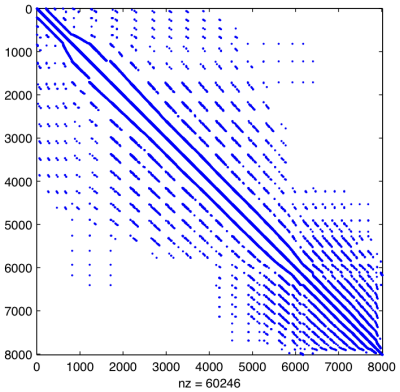
Cost of one iteration and time of partitioning is about the same in both cases !!

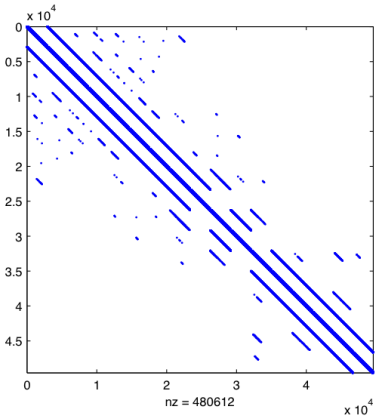
Partitioning with weights - real test case (ALL)

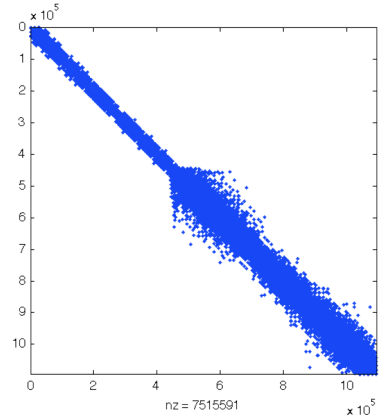








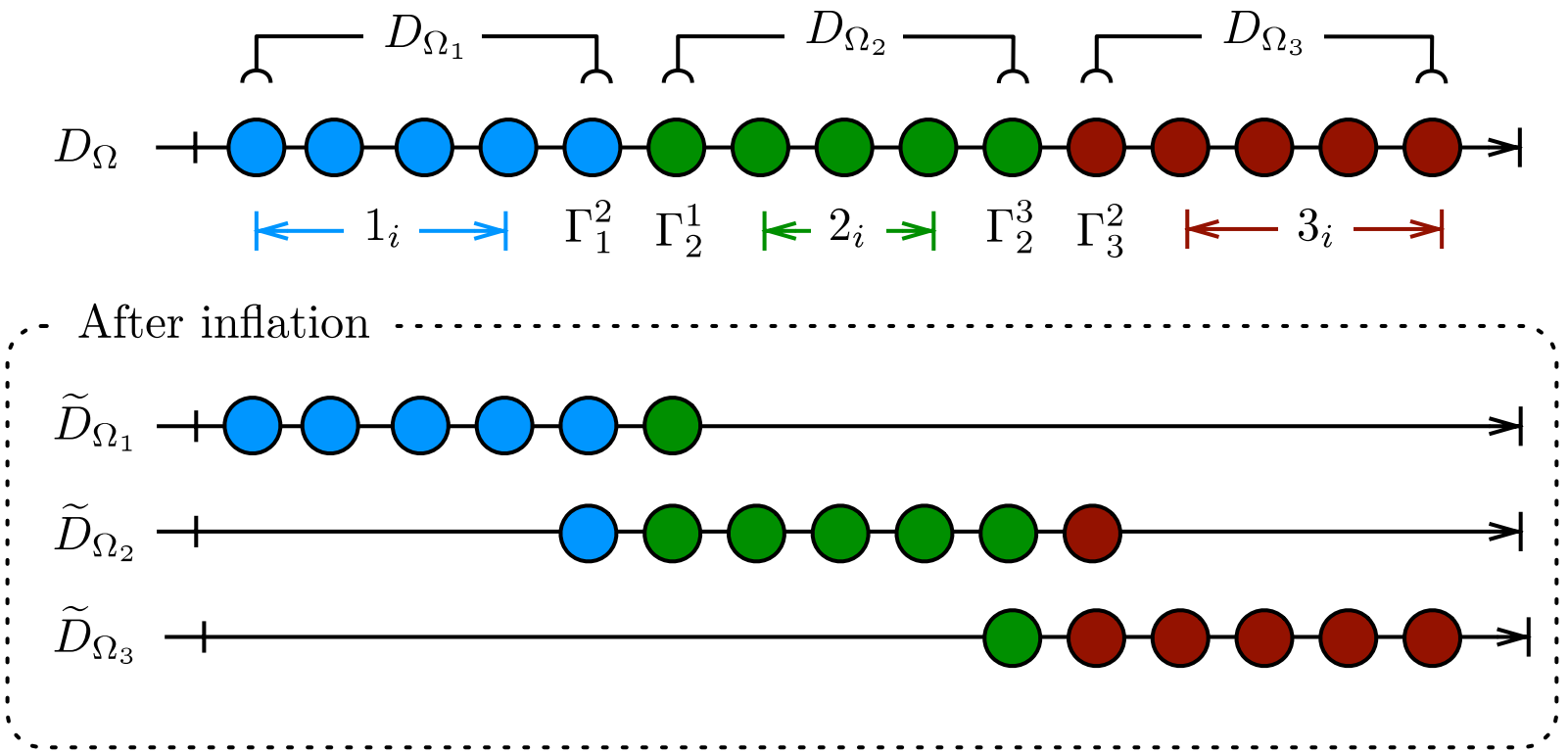




Cost of one iteration and time of partitioning is about the same in both cases !!

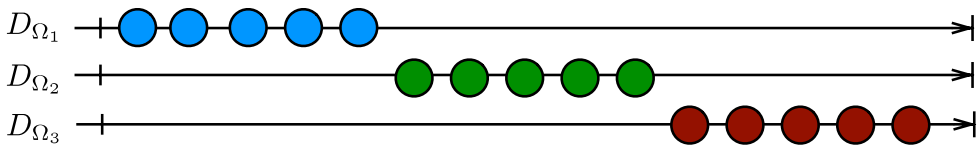
Enlarge overlap = algebraically *inflate* operator

Bigger overlap $\Longrightarrow$ faster convergence !!



Enlarge overlap = algebraically *inflate* operator

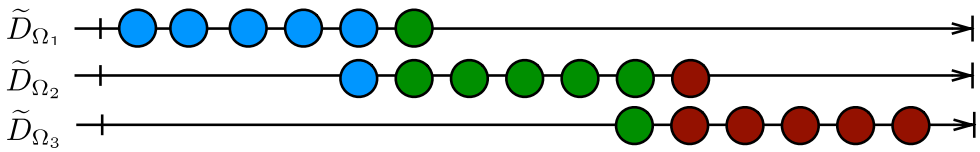
Before inflation



$$\left[\begin{array}{cc|ccc|cc}
A_{1_i1_i} & A_{1_i\Gamma_1^2} & 0 & 0 & 0 & 0 & 0 \\
A_{\Gamma_1^21_i} & A_{\Gamma_1^2\Gamma_1^2} & 0 & A_{\Gamma_1^2\Gamma_2^1} & 0 & 0 & 0 \\
\hline
0 & 0 & A_{2_i2_i} & A_{2_i\Gamma_2^1} & A_{2_i\Gamma_2^3} & 0 & 0 \\
0 & A_{\Gamma_2^1\Gamma_1^2} & A_{\Gamma_2^12_i} & A_{\Gamma_2^1\Gamma_2^1} & 0 & 0 & 0 \\
0 & 0 & A_{\Gamma_2^32_i} & 0 & A_{\Gamma_2^3\Gamma_2^3} & 0 & A_{\Gamma_2^3\Gamma_3^2} \\
\hline
0 & 0 & 0 & 0 & 0 & A_{3_i3_i} & A_{3_i\Gamma_3^2} \\
0 & 0 & 0 & 0 & A_{\Gamma_3^2\Gamma_2^3} & A_{\Gamma_3^23_i} & A_{\Gamma_3^2\Gamma_3^2}
\end{array}\right]
\left[\begin{array}{c}
U_{1_i} \\ U_{\Gamma_1^2} \\ \hline U_{2_i} \\ U_{\Gamma_2^1} \\ U_{\Gamma_2^3} \\ \hline U_{3_i} \\ U_{\Gamma_3^2}
\end{array}\right]
=
\left[\begin{array}{c}
F_{1_i} \\ F_{\Gamma_1^2} \\ \hline F_{2_i} \\ F_{\Gamma_2^1} \\ F_{\Gamma_2^3} \\ \hline F_{3_i} \\ F_{\Gamma_3^2}
\end{array}\right]$$

Enlarge overlap = algebraically *inflate* operator

After inflation

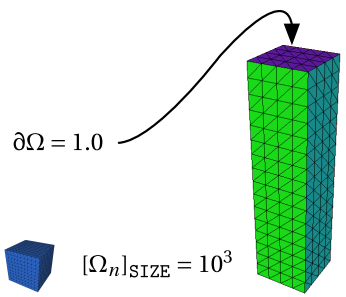


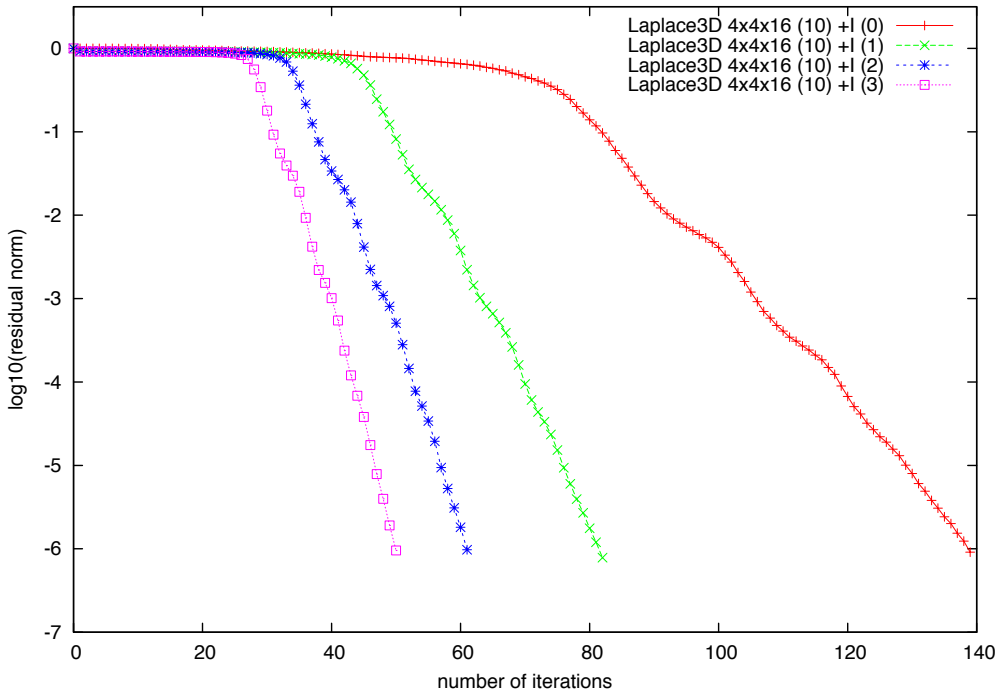
$$
\left[\begin{array}{ccc|ccccc|ccc}
A_{1_i1_i} & A_{1_i\Gamma_1^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
A_{\Gamma_1^21_i} & A_{\Gamma_1^2\Gamma_1^2} & A_{\Gamma_1^2\Gamma_2^1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & A_{\Gamma_2^1\Gamma_1^2} & A_{\Gamma_2^1\Gamma_2^1} & A_{\Gamma_2^12_i} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & A_{2_i2_i} & A_{2_i\Gamma_2^1} & A_{2_i\Gamma_2^3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & A_{\Gamma_2^12_i} & A_{\Gamma_2^1\Gamma_2^1} & 0 & A_{\Gamma_2^1\Gamma_1^2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & A_{\Gamma_2^32_i} & 0 & A_{\Gamma_2^3\Gamma_2^3} & 0 & A_{\Gamma_2^3\Gamma_3^2} & 0 & 0 & 0 \\
A_{\Gamma_1^21_i} & 0 & 0 & 0 & A_{\Gamma_1^2\Gamma_2^1} & 0 & A_{\Gamma_1^2\Gamma_1^2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & A_{\Gamma_3^2\Gamma_2^3} & 0 & A_{\Gamma_3^2\Gamma_3^2} & A_{\Gamma_3^23_i} & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_{3_i3_i} & A_{3_i\Gamma_3^2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_{\Gamma_3^23_i} & A_{\Gamma_3^2\Gamma_3^2} & A_{\Gamma_3^2\Gamma_2^3} \\
0 & 0 & 0 & A_{\Gamma_2^32_i} & 0 & 0 & 0 & 0 & 0 & A_{\Gamma_2^3\Gamma_3^2} & A_{\Gamma_2^3\Gamma_2^3}
\end{array}\right]
$$

Inflation vs Computational Time

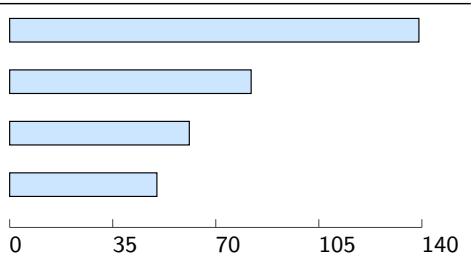
Matrix L3D4x4x16n10.mtx

nparts	nrows	nnz
256	270,641	3,941,521

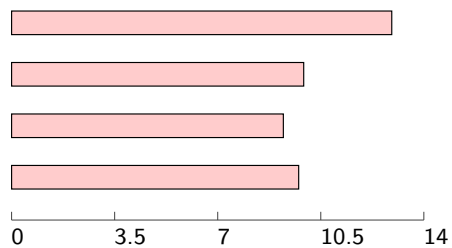




	n-iter
I(0)	139
I(1)	82
I(2)	61
I(3)	50



inf$[s]$ + LU$[s]$ + sol$[s]$	$\sum[s]$
0.00+0.22+12.69	12.91
0.34+0.31+9.27	9.92
0.79+0.59+7.85	9.23
1.53+1.69+6.56	9.75



Modified Schwarz Method

New interface condition $\Longrightarrow$ additional augmented matrixes defined on the interface between sub-domains.

$$
\left[\begin{array}{ccc|ccccc|ccc}
A_{1_i1_i} & A_{1_i\Gamma_1^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
A_{\Gamma_1^2 1_i} & A_{\Gamma_1^2\Gamma_1^2} & A_{\Gamma_1^2\Gamma_2^1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & A_{\Gamma_2^1\Gamma_1^2} & A_{\Gamma_2^1\Gamma_2^1}+S_2^1 & A_{\Gamma_2^1 2_i} & -S_2^1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & A_{2_i2_i} & A_{2_i\Gamma_2^1} & A_{2_i\Gamma_2^3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & A_{\Gamma_2^1 2_i} & A_{\Gamma_2^1\Gamma_2^1} & 0 & A_{\Gamma_2^1\Gamma_1^2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & A_{\Gamma_2^3 2_i} & 0 & A_{\Gamma_2^3\Gamma_2^3} & 0 & A_{\Gamma_2^3\Gamma_3^2} & 0 & 0 & 0 \\
A_{\Gamma_1^2 1_i} & -S_1^2 & 0 & 0 & A_{\Gamma_1^2\Gamma_2^1} & 0 & A_{\Gamma_1^2\Gamma_1^2}+S_1^2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & A_{\Gamma_3^2\Gamma_2^3} & 0 & A_{\Gamma_3^2\Gamma_3^2}+S_3^2 & A_{\Gamma_3^2 3_i} & -S_3^2 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_{3_i3_i} & A_{3_i\Gamma_3^2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_{\Gamma_3^2 3_i} & A_{\Gamma_3^2\Gamma_3^2} & A_{\Gamma_3^2\Gamma_2^3} \\
0 & 0 & 0 & A_{\Gamma_2^3 2_i} & 0 & -S_2^3 & 0 & 0 & 0 & A_{\Gamma_2^3\Gamma_3^2} & A_{\Gamma_2^3\Gamma_2^3}+S_2^3
\end{array}\right]
$$

Modified Schwarz Method - Optimal Interface Conditions

$$\left[\begin{array}{ccc|ccc} A_{1_i1_i} & A_{1_i1_\Gamma} & 0 & 0 & 0 & 0 \\ A_{1_\Gamma1_i} & A_{1_\Gamma1_\Gamma} & A_{1_\Gamma2_\Gamma} & 0 & 0 & 0 \\ 0 & A_{2_\Gamma1_\Gamma} & A_{2_\Gamma2_\Gamma}+S_1 & A_{2_\Gamma2_i} & -S_1 & 0 \\ \hline 0 & 0 & 0 & A_{2_i2_i} & A_{2_i2_\Gamma} & 0 \\ 0 & 0 & 0 & A_{2_\Gamma2_i} & A_{2_\Gamma2_\Gamma} & A_{2_\Gamma1_\Gamma} \\ A_{1_\Gamma1_i} & -S_2 & 0 & 0 & A_{1_\Gamma2_\Gamma} & A_{1_\Gamma1_\Gamma}+S_2 \end{array}\right] \left[\begin{array}{c} U_{1_i} \\ U_{1_\Gamma} \\ U_{2_\Gamma} \\ \hline U_{2_i} \\ U_{2_\Gamma} \\ U_{1_\Gamma} \end{array}\right] = \left[\begin{array}{c} F_{1_i} \\ F_{1_\Gamma} \\ F_{2_\Gamma} \\ \hline F_{2_i} \\ F_{2_\Gamma} \\ F_{1_\Gamma} \end{array}\right]$$

Optimal choice for two domain case

The choice of S_1 and S_2 can be "adjusted" in such a way that Schur complements appears in inflated operator $\widetilde{A}$ i.e.

$$S_1^{opt} = -A_{2_\Gamma2_i}A_{2_i2_i}^{-1}A_{2_i2_\Gamma}$$

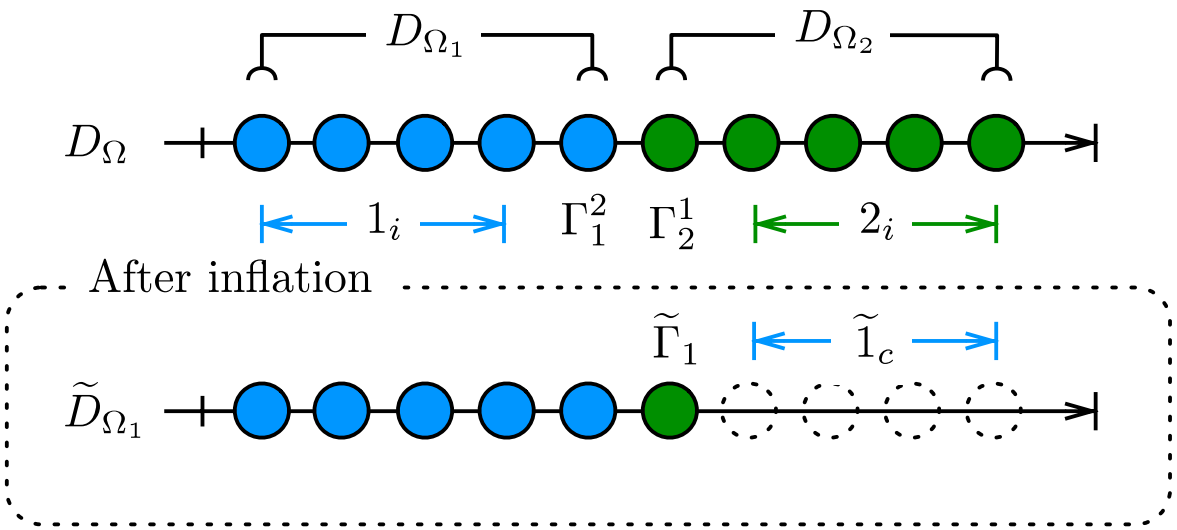
$$S_2^{opt} = -A_{1_\Gamma1_i}A_{1_i1_i}^{-1}A_{1_i1_\Gamma}$$

is optimal, and the ASM in form of preconditioner in an iterative Krylov solver, converges in two steps.

Optimal Interface Conditions Approximation

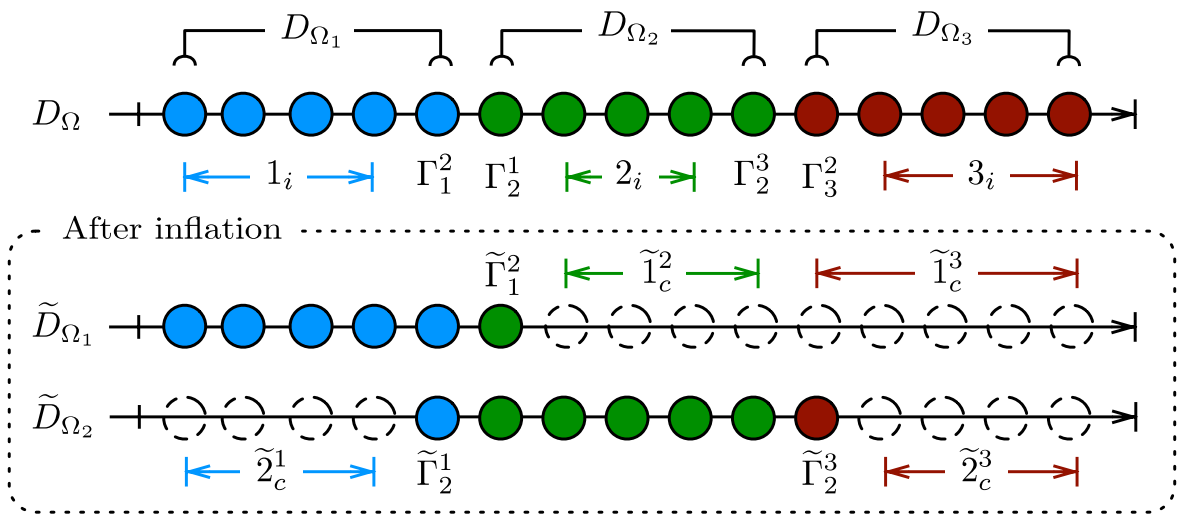
For a sake of simplicity we consider only two subdomains and we focus on domain Ω_1 which we simply denote by 1 and its inflated counterpart by $\widetilde{1}$.

$$\begin{bmatrix} A_{1_i 1_i} & A_{1_i \widetilde{\Gamma}_1} & 0 \\ A_{\widetilde{\Gamma}_1 1_i} & A_{\widetilde{\Gamma}_1 \widetilde{\Gamma}_1} & A_{\widetilde{\Gamma}_1 \widetilde{1}_c} \\ 0 & A_{\widetilde{1}_c \widetilde{\Gamma}_1} & A_{\widetilde{1}_c \widetilde{1}_c} \end{bmatrix} \begin{bmatrix} U_{1_i} \\ U_{\widetilde{\Gamma}_1} \\ U_{\widetilde{1}_c} \end{bmatrix} = \begin{bmatrix} F_{1_i} \\ F_{\widetilde{\Gamma}_1} \\ F_{\widetilde{1}_c} \end{bmatrix}$$



Optimal Interface Conditions Approximation - General Case

What about general case ($2 < N$ - subdomains) ?



Diagonal Approximation

GOAL: approximate optimal interface conditions by a sparse matrix keeping some filtering properties.

$$S^{opt}_{\widetilde{\Gamma}_1\widetilde{\Gamma}_1} := -A_{\widetilde{\Gamma}_1\widetilde{1}_c} A^{-1}_{\widetilde{1}_c\widetilde{1}_c} A_{\widetilde{1}_c\widetilde{\Gamma}_1}$$

More precisely we seek an approximation to $S^{opt}_{\widetilde{\Gamma}_1\widetilde{\Gamma}_1}$ in form:

The optimal interface conditions approximation

$$S^{\widetilde{\approx}}_{\widetilde{\Gamma}_1\widetilde{\Gamma}_1} := -A_{\widetilde{\Gamma}_1\widetilde{1}_c} \beta_{\widetilde{1}_c\widetilde{1}_c} A_{\widetilde{1}_c\widetilde{\Gamma}_1}$$

such that, the optimality condition is verified on the vector $V_{\widetilde{\Gamma}_1}$

$$-A_{\widetilde{\Gamma}_1\widetilde{1}_c} \beta_{\widetilde{1}_c\widetilde{1}_c} A_{\widetilde{1}_c\widetilde{\Gamma}_1} V_{\widetilde{\Gamma}_1} = S^{opt}_{\widetilde{\Gamma}_1\widetilde{\Gamma}_1} V_{\widetilde{\Gamma}_1}$$

where V is a harmonic vector i.e.,

$$A_{\widetilde{1}_c\widetilde{1}_c} V_{\widetilde{1}_c} + A_{\widetilde{1}_c\widetilde{\Gamma}_1} V_{\widetilde{\Gamma}_1} = 0$$

Sparse matrix $\beta_{\widetilde{1}_c\widetilde{1}_c}$

If V is a harmonic vector in $\widetilde{1}_c$, we take $\beta_{\widetilde{1}_c\widetilde{1}_c}$ to be a diagonal matrix defined by

$\beta_{\widetilde{1}_c\widetilde{1}_c}$ operator

$$\beta_{\widetilde{1}_c\widetilde{1}_c} := \mathit{diag}\left(-V_{\widetilde{1}_c}./A_{\widetilde{1}_c\widetilde{\Gamma}_1}V_{\widetilde{\Gamma}_1}\right)$$

and $\beta_{\widetilde{1}_c\widetilde{1}_c} = 0$ otherwise.

"./" - element wise division

$$\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} ./ \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} v_1 / w_1 \\ \vdots \\ v_n / w_n \end{bmatrix}$$

Harmonic Vector - Facts

- Due to the block preconditioner M^{-1} (the Schwarz method) the vectors in the *Krylov* space $\mathcal{K}_m(\widetilde{M}^{-1}\widetilde{A}, r_0)$ are sub-domain wise harmonic.
- Many iterative methods use Krylov space for computation (selected) eigenvalues.
- **Our choice of harmonic vector is an approximated eigenvector of $\widetilde{M}^{-1}\widetilde{A}$ assosiated with the smallest eigenvalue λ (we use Krylov subspaces created via GMRES).**

Harmonic Vector - Facts

- Due to the block preconditioner M^{-1} (the Schwarz method) the vectors in the *Krylov* space $\mathcal{K}_m(\widetilde{M}^{-1}\widetilde{A}, r_0)$ are sub-domain wise harmonic.
- Many iterative methods use Krylov space for computation (selected) eigenvalues.
- **Our choice of harmonic vector is an approximated eigenvector of $\widetilde{M}^{-1}\widetilde{A}$ assosiated with the smallest eigenvalue λ (we use Krylov subspaces created via GMRES).**

Harmonic Vector - Facts

- Due to the block preconditioner M^{-1} (the Schwarz method) the vectors in the *Krylov* space $\mathcal{K}_m(\widetilde{M}^{-1}\widetilde{A}, r_0)$ are sub-domain wise harmonic.
- Many iterative methods use Krylov space for computation (selected) eigenvalues.
- **Our choice of harmonic vector is an approximated eigenvector of $\widetilde{M}^{-1}\widetilde{A}$ assosiated with the smallest eigenvalue λ (we use Krylov subspaces created via GMRES).**

Approximate eigenvector from GMRES solver

1. The computational kernel of GMRES is the Arnoldi process which computes the orthonormal basis W_m for the Krylov subspace $\mathcal{K}_m(\widetilde{M}^{-1}\widetilde{A}, r_0)$.
2. Since the Arnoldi basis is orthonormal, $W_m = (w_1\ w_2\ \ldots\ w_m)$ is an orthogonal matrix ($W_m \in \mathbb{R}^{n\times m}$).
3. In the orthogonalisation process the scalars h_{ij} are computed so that the square upper *Hessenberg* matrix $H_m \in \mathbb{R}^{m\times m}$ satisfies the fundamental relation:

$$H_m = W_m^H \widetilde{M}^{-1}\widetilde{A}W_m$$

Approximate eigenvector from GMRES solver

1. The computational kernel of GMRES is the Arnoldi process which computes the orthonormal basis W_m for the Krylov subspace $\mathcal{K}_m(\widetilde{M}^{-1}\widetilde{A}, r_0)$.
2. Since the Arnoldi basis is orthonormal, $W_m = (w_1\ w_2\ \ldots\ w_m)$ is an orthogonal matrix ($W_m \in \mathbb{R}^{n\times m}$).
3. In the orthogonalisation process the scalars h_{ij} are computed so that the square upper *Hessenberg* matrix $H_m \in \mathbb{R}^{m\times m}$ satisfies the fundamental relation:

$$H_m = W_m^H \widetilde{M}^{-1}\widetilde{A}W_m$$

Approximate eigenvector from GMRES solver

1. The computational kernel of GMRES is the Arnoldi process which computes the orthonormal basis W_m for the Krylov subspace $\mathcal{K}_m(\widetilde{M}^{-1}\widetilde{A}, r_0)$.
2. Since the Arnoldi basis is orthonormal, $W_m = (w_1\ w_2\ \ldots\ w_m)$ is an orthogonal matrix ($W_m \in \mathbb{R}^{n\times m}$).
3. In the orthogonalisation process the scalars h_{ij} are computed so that the square upper *Hessenberg* matrix $H_m \in \mathbb{R}^{m\times m}$ satisfies the fundamental relation:

$$H_m = W_m^H \widetilde{M}^{-1}\widetilde{A}W_m$$

Approximate eigenvector from GMRES solver

4. The eigenvalues of H_m are called Ritz values and they approximate the eigenvalues of $\widetilde{M}^{-1}\widetilde{A}$.

if $z_\star$ is an chosen eigenvector of H_m, then $\mathcal{V}_\star = W_m z_\star$ is almost an eigenvector of $\widetilde{M}^{-1}\widetilde{A}$, for the same eigenvalue λ i.e.,

$$\begin{array}{rcccl} \widetilde{M}^{-1}\widetilde{A}\mathcal{V}_\star & \simeq & W_m H_m W_m^H W_m z_\star & = & \\ & = & W_m H_m z_\star & = & \\ & = & W_m \lambda z_\star & = & \lambda \mathcal{V}_\star \end{array}$$

5. In practice a specific *Lapack* procedure can be used to compute the eigenelements of H_m.

Approximate eigenvector from GMRES solver

4. The eigenvalues of H_m are called Ritz values and they approximate the eigenvalues of $\widetilde{M}^{-1}\widetilde{A}$.

if $z_\star$ is an chosen eigenvector of H_m, then $\mathcal{V}_\star = W_m z_\star$ is almost an eigenvector of $\widetilde{M}^{-1}\widetilde{A}$, for the same eigenvalue λ i.e.,

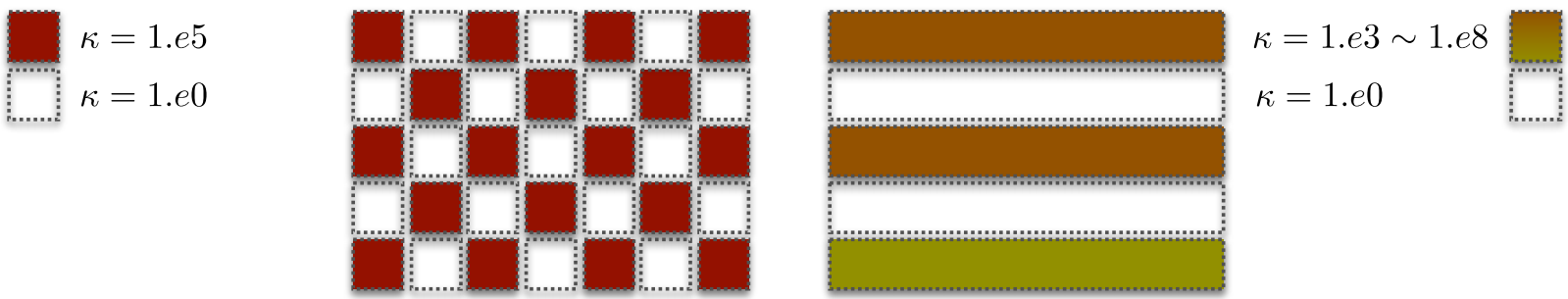
$$\begin{array}{rcccl} \widetilde{M}^{-1}\widetilde{A}\mathcal{V}_\star & \simeq & W_m H_m W_m^H W_m z_\star & = & \\ & = & W_m H_m z_\star & = & \\ & = & W_m \lambda z_\star & = & \lambda \mathcal{V}_\star \end{array}$$

5. In practice a specific *Lapack* procedure can be used to compute the eigenelements of H_m.

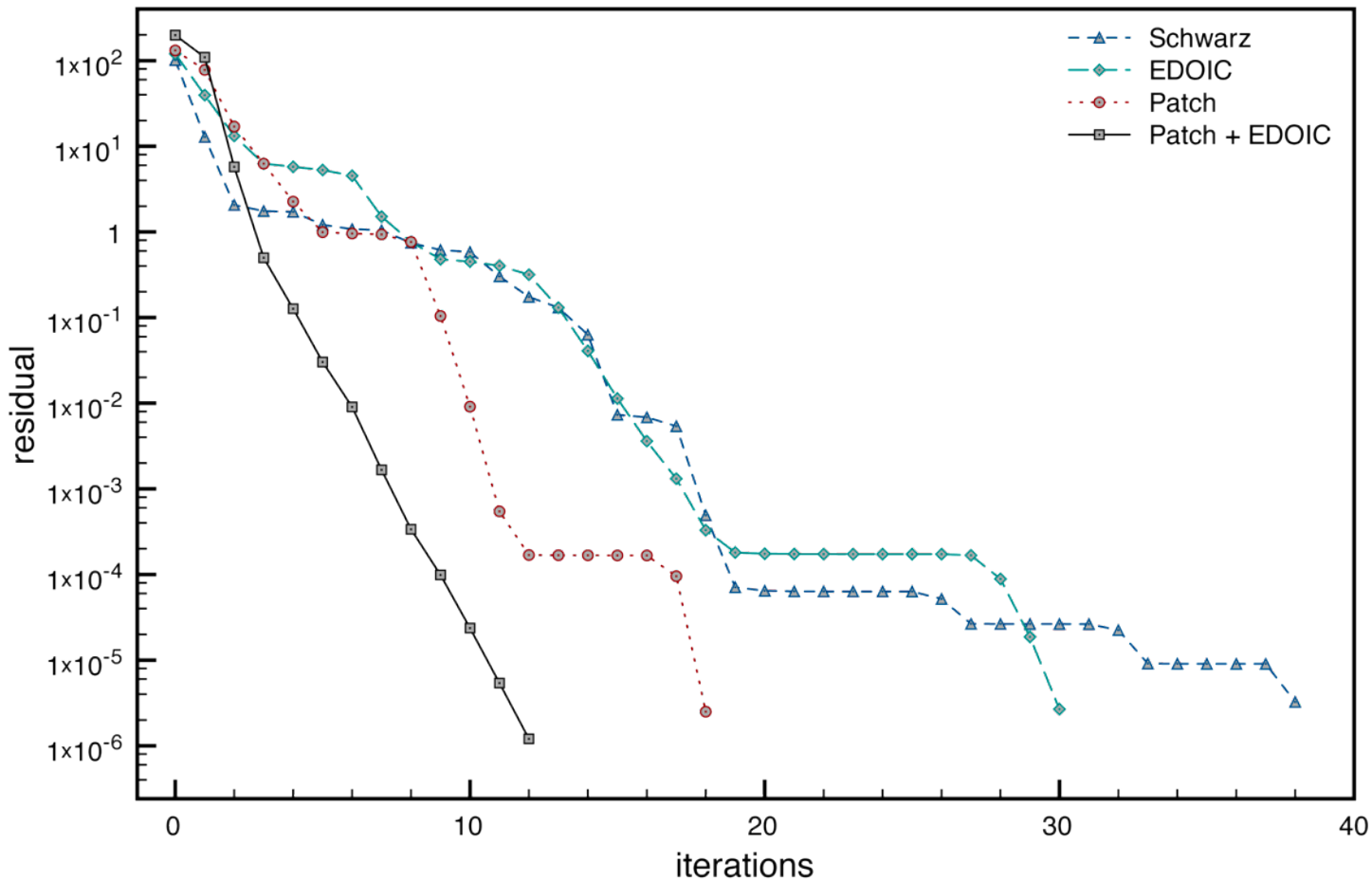
Numerical Experiments with EDOIC

Two sub-domain (complex) case:

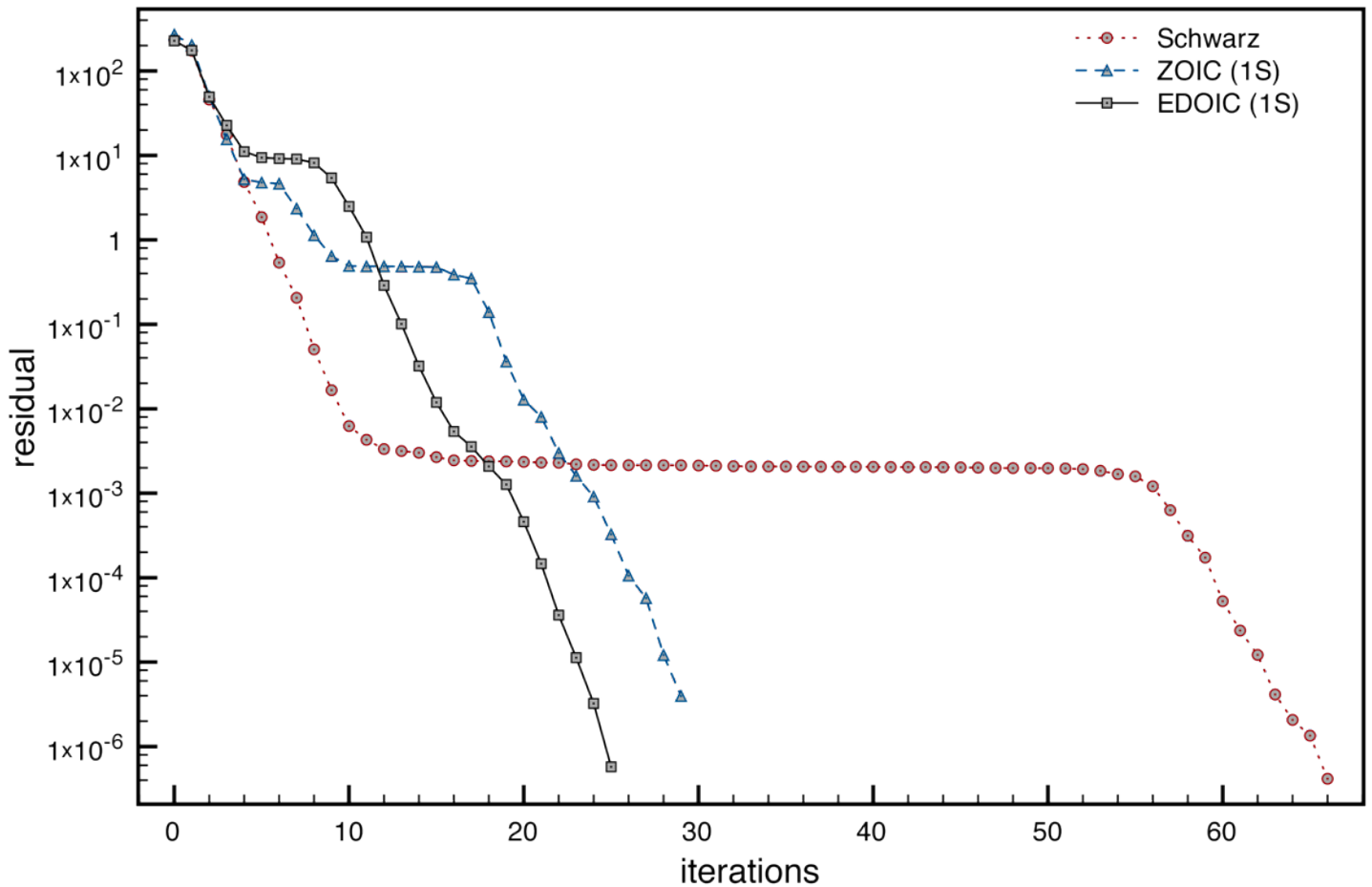
$$\begin{cases} \left(\eta(x,y) - div(\kappa(x,y)\vec{\nabla})\right) u(x,y) = f(x,y) & \text{in} \quad \Omega \\ u(x,y) = 0 & \text{on} \quad \partial\Omega_D \\ \frac{\partial u}{\partial n} = 0 & \text{on} \quad \partial\Omega_N \end{cases}$$



Numerical Experiments with EDOIC

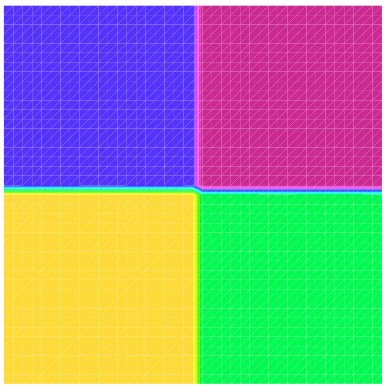


Numerical Experiments with EDOIC

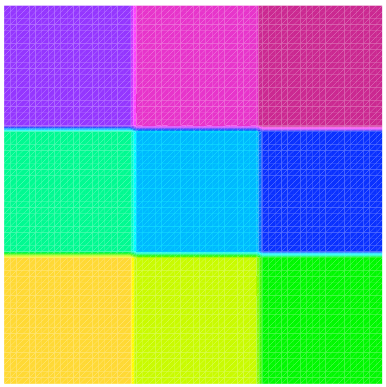


Numerical Experiment (fixed size problem)

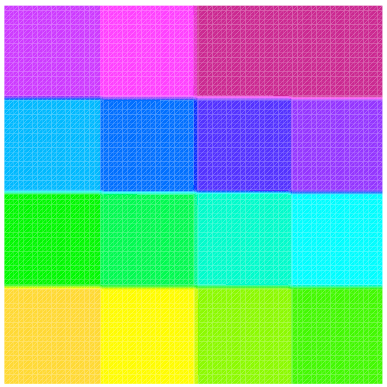
Fixed size problem $n_x = n_y = 50$



(a) $M_x = M_y = 2$

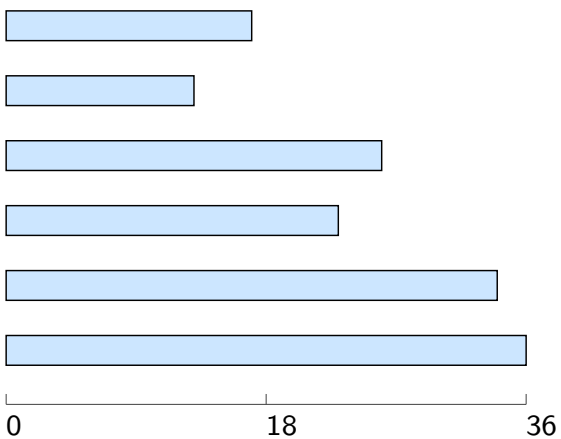


(b) $M_x = M_y = 3$



(c) $M_x = M_y = 4$

Method	*n-iter*
(a) ASM	17
(a) MSM+EDOIC(3)	13
(b) ASM	26
(b) MSM+EDOIC(3)	23
(c) ASM	34
(c) MSM+EDOIC(3)	36



Two-level preconditioner

PROBLEM: Convergence of the Schwarz method deteriorates with increasing number of subdomains.

SOLUTION: "Remove" smallest eigenvalues that slow down the Schwarz method.

It leads us to construction of two-level preconditioner using

$$P := \mathbb{I} - A\left(ZE^{-1}Z^T\right) \quad E := Z^T AZ$$

which are common ingredients of the coarse grid, deflation and AMG preconditioners (see [Tang et al., 2009]).

An effective two-level preconditioner is highly dependent on the choice of coarse grid subspace $Z \in \mathbb{R}^{n \times m}$.

How to choose Z ?

Two-level preconditioner

PROBLEM: Convergence of the Schwarz method deteriorates with increasing number of subdomains.

SOLUTION: "Remove" smallest eigenvalues that slow down the Schwarz method.

It leads us to construction of two-level preconditioner using

$$P := \mathbb{I} - A\left(ZE^{-1}Z^T\right) \quad E := Z^T AZ$$

which are common ingredients of the coarse grid, deflation and AMG preconditioners (see [Tang et al., 2009]).

An effective two-level preconditioner is highly dependent on the choice of coarse grid subspace $Z \in \mathbb{R}^{n \times m}$.

How to choose Z ?

Two-level preconditioner

PROBLEM: Convergence of the Schwarz method deteriorates with increasing number of subdomains.

SOLUTION: "Remove" smallest eigenvalues that slow down the Schwarz method.

It leads us to construction of two-level preconditioner using

$$P := \mathbb{I} - A\left(ZE^{-1}Z^T\right) \quad E := Z^T AZ$$

which are common ingredients of the coarse grid, deflation and AMG preconditioners (see [Tang et al., 2009]).

An effective two-level preconditioner is highly dependent on the choice of coarse grid subspace $Z \in \mathbb{R}^{n\times m}$.

How to choose Z ?

Two-level preconditioner

PROBLEM: Convergence of the Schwarz method deteriorates with increasing number of subdomains.

SOLUTION: "Remove" smallest eigenvalues that slow down the Schwarz method.

It leads us to construction of two-level preconditioner using

$$P := \mathbb{I} - A\left(ZE^{-1}Z^T\right) \quad E := Z^T AZ$$

which are common ingredients of the coarse grid, deflation and AMG preconditioners (see [Tang et al., 2009]).

An effective two-level preconditioner is highly dependent on the choice of coarse grid subspace $Z \in \mathbb{R}^{n\times m}$.

How to choose Z ?

Two-level preconditioner

PROBLEM: Convergence of the Schwarz method deteriorates with increasing number of subdomains.

SOLUTION: "Remove" smallest eigenvalues that slow down the Schwarz method.

It leads us to construction of two-level preconditioner using

$$P := \mathbb{I} - A\left(ZE^{-1}Z^T\right) \quad E := Z^T AZ$$

which are common ingredients of the coarse grid, deflation and AMG preconditioners (see [Tang et al., 2009]).

An effective two-level preconditioner is highly dependent on the choice of coarse grid subspace $Z \in \mathbb{R}^{n\times m}$.

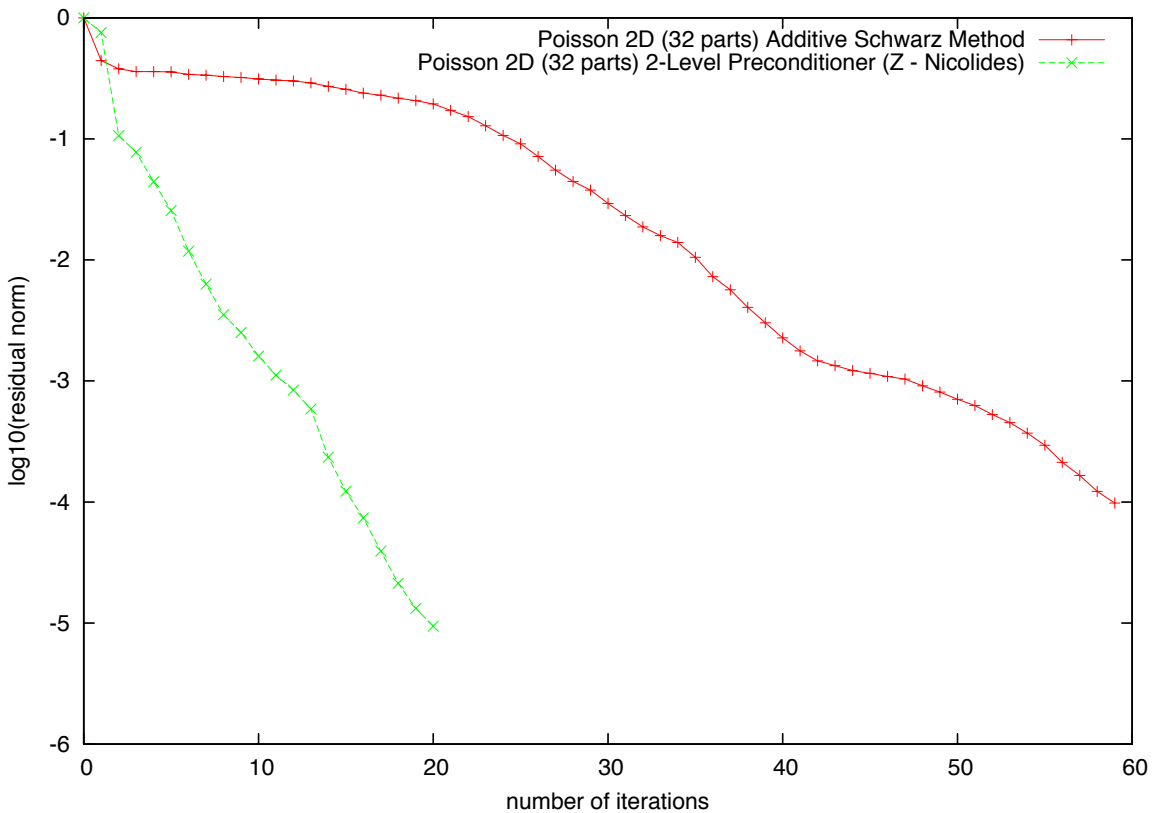
How to choose Z ?

Coarse grid correction for smooth problems

For a Poisson like problem, Nicolaides proposed [Nicolaides, 1987]:

$$Z = \begin{bmatrix} \mathbf{1}_{\Omega_1} & 0 & \cdots & 0 \\ \vdots & \mathbf{1}_{\Omega_2} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \mathbf{1}_{\Omega_J} \end{bmatrix}$$

$$(z_k)_l = \begin{cases} 1 & l \in \Omega_j \\ 0 & l \notin \Omega_j \end{cases}$$

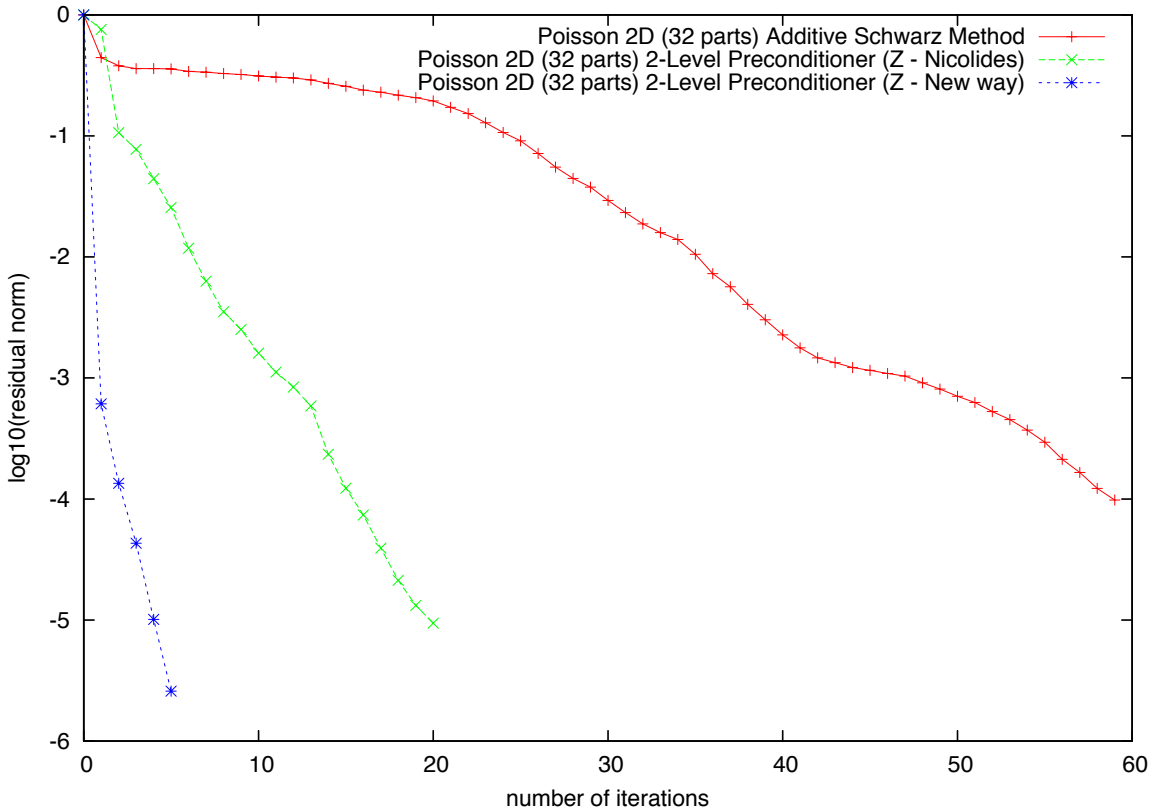


Coarse grid correction for smooth problems

For a Poisson like problem, Nicolaides proposed [Nicolaides, 1987]:

$$Z = \begin{bmatrix} \mathbf{1}_{\Omega_1} & 0 & \cdots & 0 \\ \vdots & \mathbf{1}_{\Omega_2} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \mathbf{1}_{\Omega_J} \end{bmatrix}$$

$$(z_k)_l = \begin{cases} 1 & l \in \Omega_j \\ 0 & l \notin \Omega_j \end{cases}$$



Our choice of Z

In deflation techniques Z consists of eigenvectors or approximations of eigenvectors (which we know how to find: $\mathcal{V}_\star = W_m z_\star$)

$$Z^\star := [\mathcal{V}_1\ \mathcal{V}_2\ \cdots\ \mathcal{V}_{n_\mathcal{V}}] = \begin{bmatrix} [\mathcal{V}_1]_{\widetilde{D}_{\Omega_1}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_1}} & \cdots & [\mathcal{V}_{n_\mathcal{V}}]_{\widetilde{D}_{\Omega_1}} \\ [\mathcal{V}_1]_{\widetilde{D}_{\Omega_2}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_2}} & \cdots & [\mathcal{V}_{n_\mathcal{V}}]_{\widetilde{D}_{\Omega_2}} \\ \vdots & \vdots & & \vdots \\ [\mathcal{V}_1]_{\widetilde{D}_{\Omega_N}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_N}} & \cdots & [\mathcal{V}_{n_\mathcal{V}}]_{\widetilde{D}_{\Omega_N}} \end{bmatrix}$$

We can apply a part wise spliting to $Z^\star$ in order to construct a coarse subspace similar in structure to one proposed by Nicolaides.

Our choice of Z

In deflation techniques Z consists of eigenvectors or approximations of eigenvectors (which we know how to find: $\mathcal{V}_\star = W_m z_\star$)

Coarse grid subspace $Z \in \mathbb{R}^{n \times (n_v \times N)}$

$$Z := \begin{bmatrix} [\mathcal{V}_1]_{\widetilde{D}_{\Omega_1}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_1}} & \cdots & [\mathcal{V}_{n\mathcal{V}}]_{\widetilde{D}_{\Omega_1}} & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & [\mathcal{V}_1]_{\widetilde{D}_{\Omega_2}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_2}} & \cdots & [\mathcal{V}_{n\mathcal{V}}]_{\widetilde{D}_{\Omega_2}} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & [\mathcal{V}_1]_{\widetilde{D}_{\Omega_N}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_N}} & \cdots & [\mathcal{V}_{n\mathcal{V}}]_{\widetilde{D}_{\Omega_N}} \end{bmatrix}$$

We can apply a part wise spliting to $Z^\star$ in order to construct a coarse subspace similar in structure to one proposed by Nicolaides.

Our choice of Z

In deflation techniques Z consists of eigenvectors or approximations of eigenvectors (which we know how to find: $\mathcal{V}_\star = W_m z_\star$)

Coarse grid subspace $Z \in \mathbb{R}^{n\times(2\times 3)}$

$$Z^\star := \begin{bmatrix} [\mathcal{V}_1]_{\widetilde{D}_{\Omega_1}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_1}} \\ [\mathcal{V}_1]_{\widetilde{D}_{\Omega_2}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_2}} \\ [\mathcal{V}_1]_{\widetilde{D}_{\Omega_3}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_3}} \end{bmatrix} \rightarrow \begin{bmatrix} [\mathcal{V}_1]_{\widetilde{D}_{\Omega_1}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_1}} & & \\ & & [\mathcal{V}_1]_{\widetilde{D}_{\Omega_2}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_2}} \\ & & & \\ & & [\mathcal{V}_1]_{\widetilde{D}_{\Omega_3}} & [\mathcal{V}_2]_{\widetilde{D}_{\Omega_3}} \end{bmatrix} = Z$$

We can apply a part wise spliting to $Z^\star$ in order to construct a coarse subspace similar in structure to one proposed by Nicolaides.

Our choice of two-level preconditioner

Our choice = The two-level hybrid Schwarz preconditioner [Smith et al., 1996].

Two-level preconditioner $\mathcal{P}_{L\&R}$

$$\mathcal{P}_L := \left[\mathbb{I} - \left(ZE_L^{-1}Z^T\right)M^{-1}A + \left(ZE_L^{-1}Z^T\right)\right]$$

$$\mathcal{P}_R := \left[\mathbb{I} - \left(ZE_R^{-1}Z^T\right)AM^{-1} + \left(ZE_R^{-1}Z^T\right)\right]$$

$$M^{-1} := \begin{bmatrix} A_{D\Omega_1}^{-1} & 0 & 0 \\ 0 & A_{D\Omega_2}^{-1} & 0 \\ 0 & 0 & A_{D\Omega_3}^{-1} \end{bmatrix}$$

$$E_L := Z^T M^{-1} A Z$$
$$E_R := Z^T A M^{-1} Z$$

Left preconditioner

$$\mathcal{P}_L M^{-1} A\mathbf{u} = \mathcal{P}_L M^{-1}\mathbf{b}$$

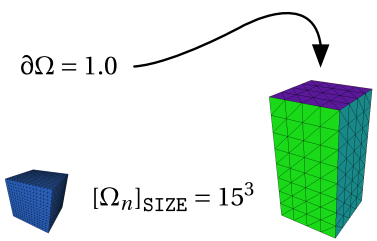
Right preconditioner

$$AM^{-1}\mathcal{P}_R\bar{\mathbf{u}} = \mathbf{b}$$
$$\mathbf{u} = M^{-1}\mathcal{P}_R\bar{\mathbf{u}}$$

Academic Problem - 3D Laplace

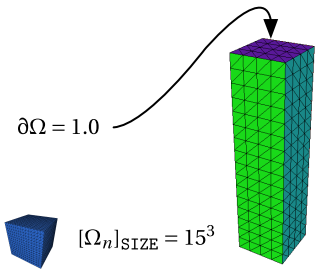
Matrix L3D4x4x8n15.mtx

nparts	nrows	nnz
128	450,241	6,606,721



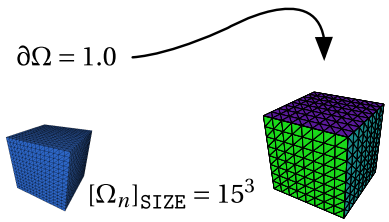
Matrix L3D4x4x16n15.mtx

nparts	nrows	nnz
256	896,761	13,187,881



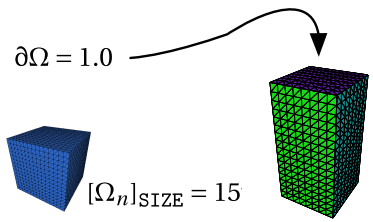
Matrix L3D8x8x8n15.mtx

nparts	nrows	nnz
512	1,771,561	26,223,481



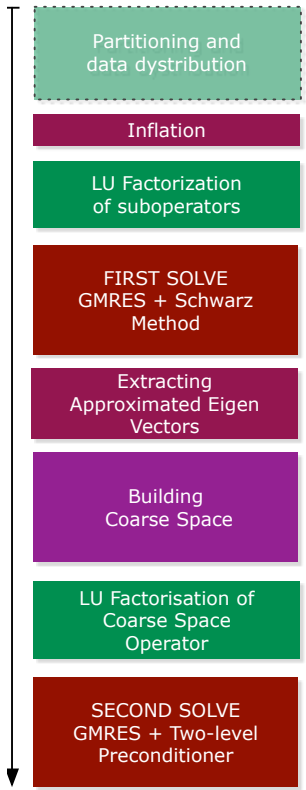
Matrix L3D8x8x16n15.mtx

nparts	nrows	nnz
1024	3,528,481	52,345,441



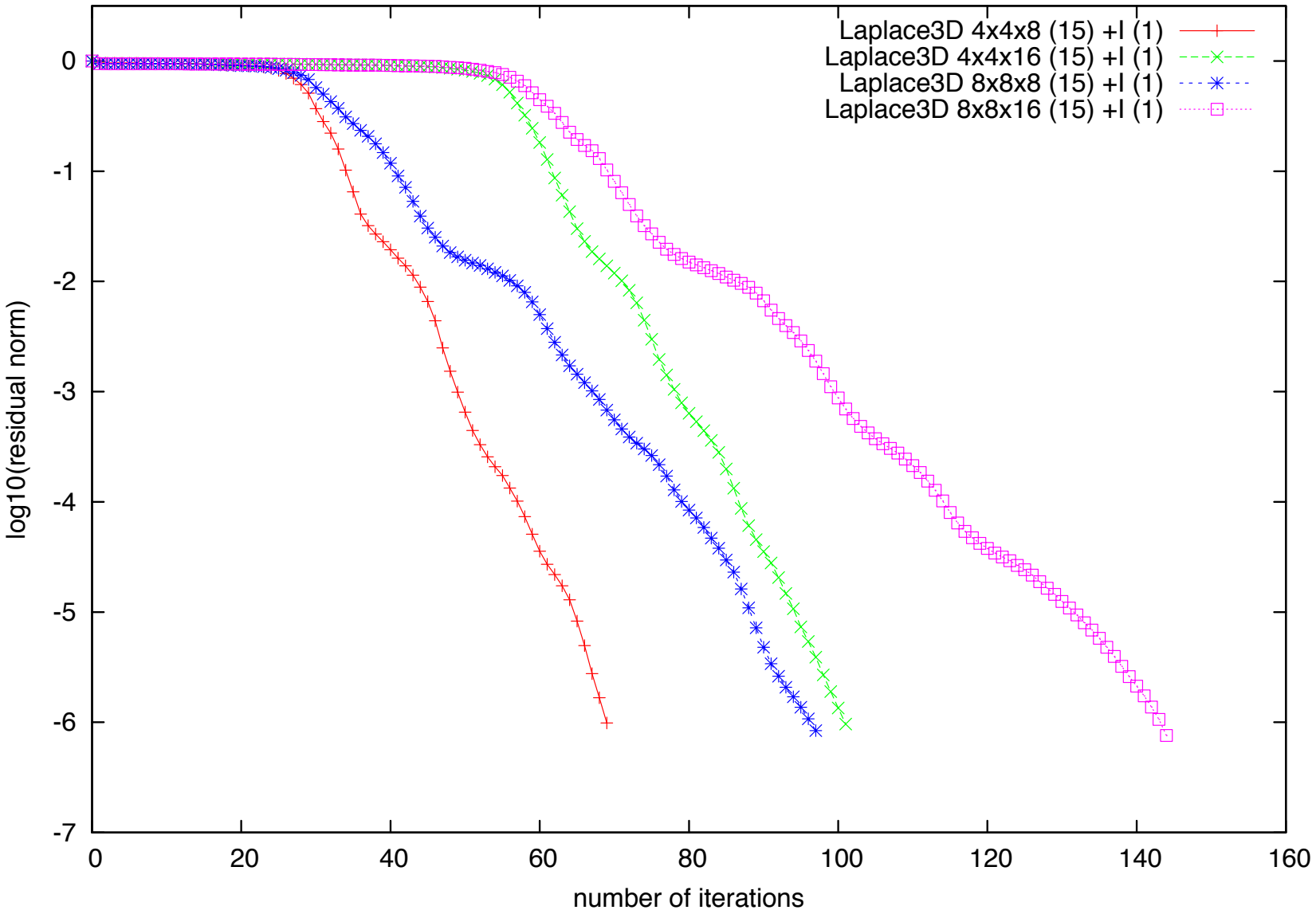
- all experiments performed on IFP cluster
  - 114 nodes equipped with 4 processes AMD Barcelona 2.3 Ghz (quad-core socket)
  - interconnected by Infiniband switched fabric (type of network topology)
- max number of available process **256**
  - thus only in two first variants we dedicated one part per one process

What we have measured ?

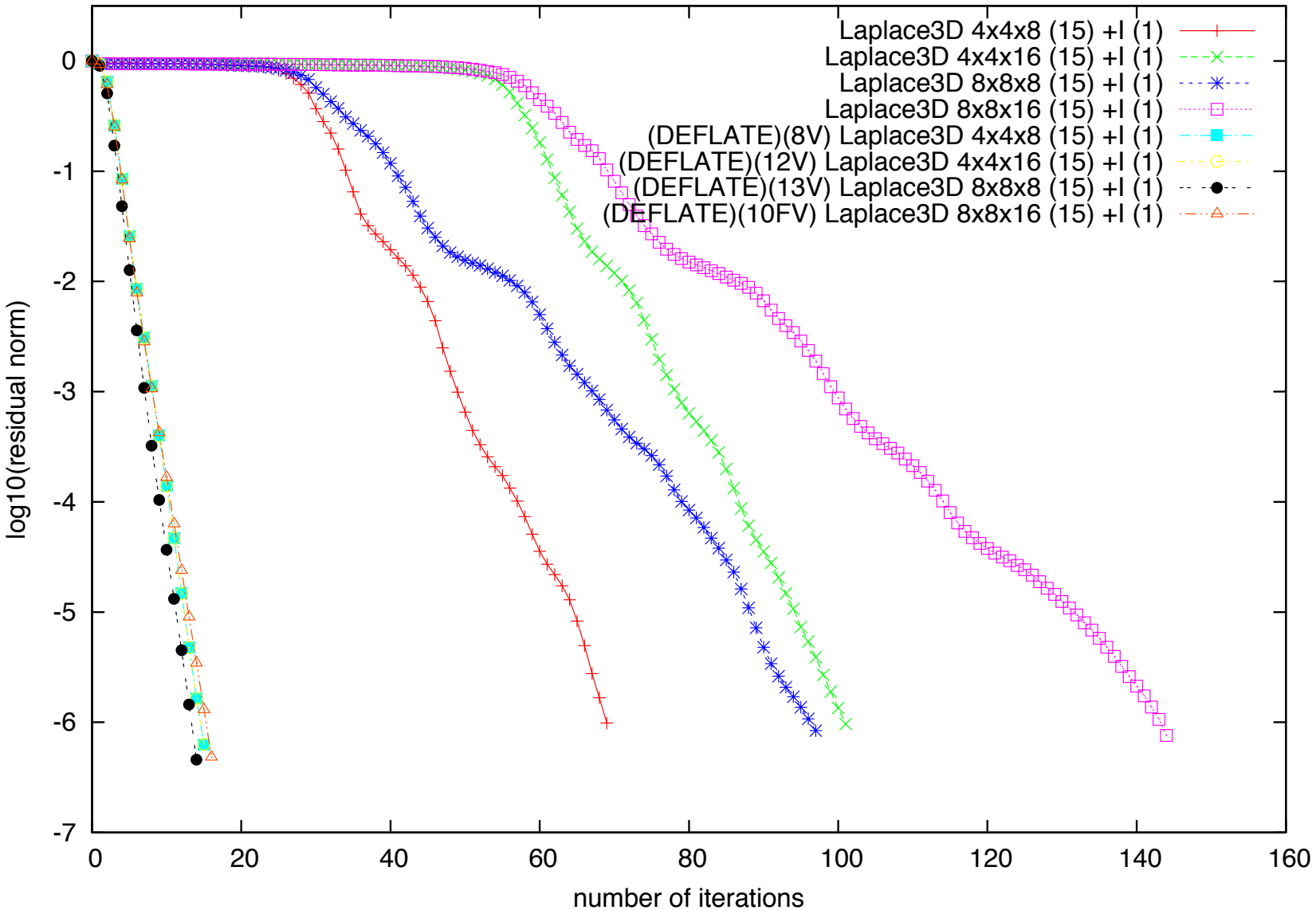


niter	number of iterations
$\kappa_{\approx}$	roughly estimated condition number given as $\kappa_{\approx} = \lambda_{max}/\lambda_{min}$ where $\lambda_{\{min,max\}}$ are the approximated, extreme eigenvalues of $(\widetilde{M}^{-1}\widetilde{A})$
nV	number of approximated eigenvectors used in construction of coarse space
$\|\|r_{sol}\|\|$	standard norm of final residual i.e., $\|\|r_{sol}\|\| = \|\|Au_{sol} - b\|\|_2/\|\|b\|\|_2$
CS$[s]$	time of "construction" coarse space operator
Inf$[s]$	time of inflation process for each level
LU$[s]$	time of LU factorisation of endomorphic `Partial Operators` in `DDMOperator`
sol$[s]$	time of iterative process (in case of varian with two-level preconditioner **sol** consist also LU factorisation time of coarse operator)

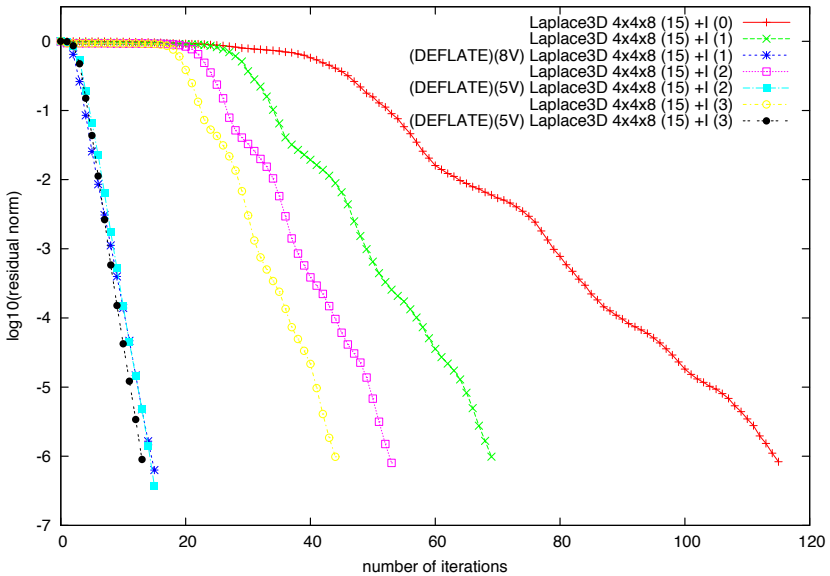
Academic Problem - 3D Laplace (SCALABILITY)

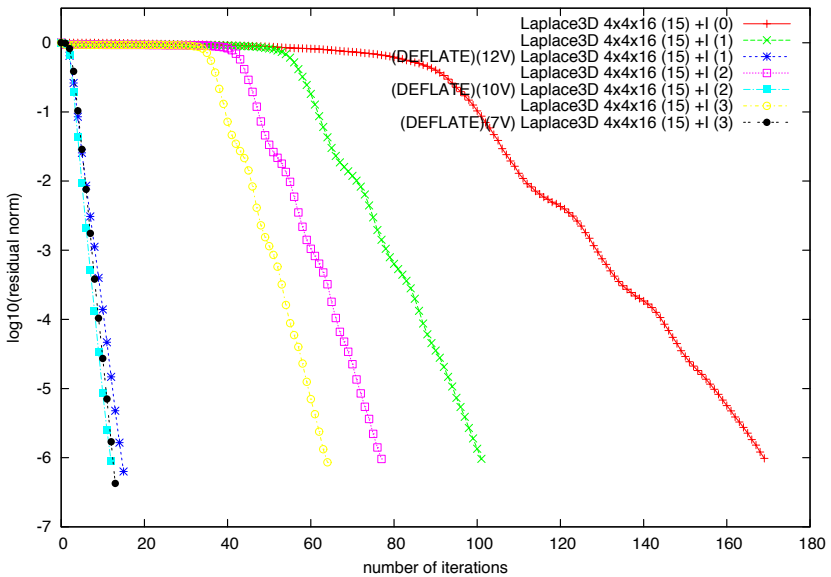


Academic Problem - 3D Laplace (SCALABILITY)



Academic Problem - 3D Laplace (TIME COST)

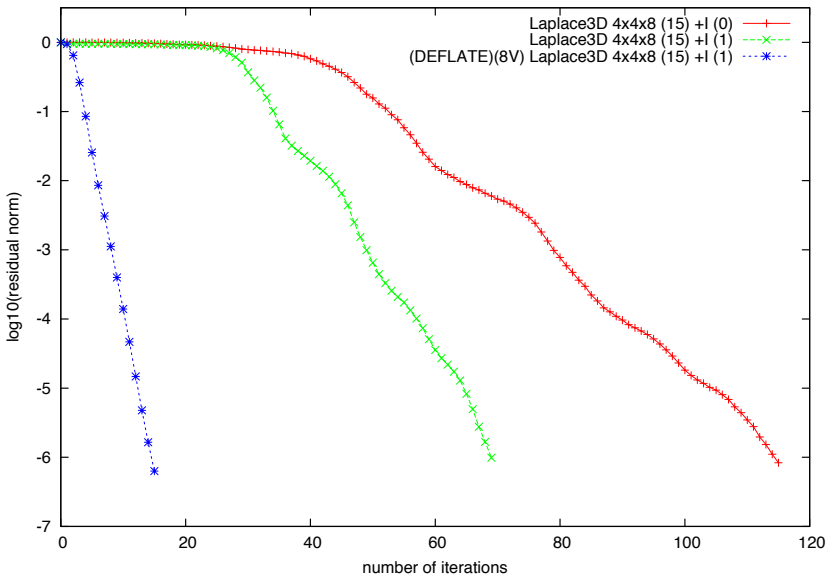


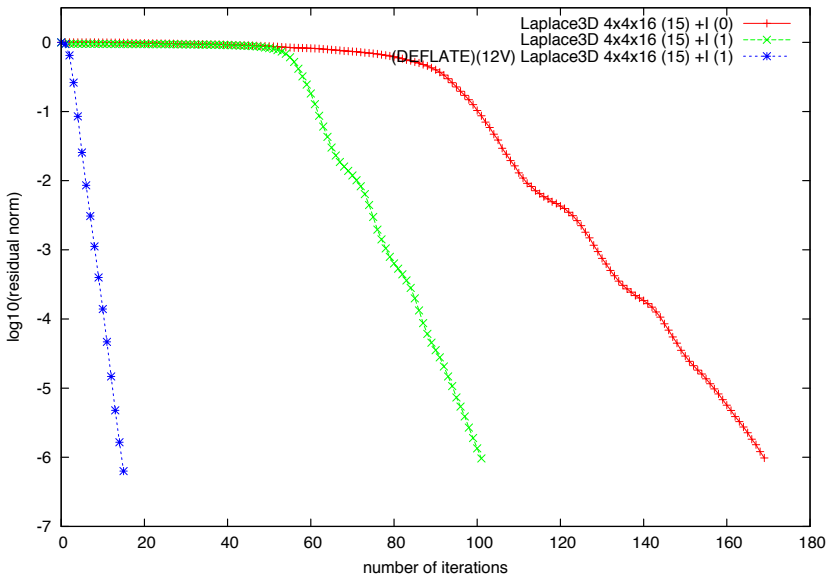


expvar	niter	$\kappa_{\approx}$	nV	$\|r_{sol}\|$	CS$[s]$	inf$[s]$	LU$[s]$	sol$[s]$
+I(0)	115	3694.10		$8.31e-07$			2.06	8.84
+I(1)	69	1174.49		$1.92e-10$		1.18	4.39	8.35
D..+I(1)	15	3.61	10	$2.55e-09$	1.18			3.36
+I(2)	53	662.34		$3.09e-09$		1.16+1.73	6.35	6.37
D..+I(2)	15	2.78	5	$1.07e-09$	0.76			4.41
+I(3)	44	444.15		$2.59e-09$		1.19+1.76+2.60	14.59	8.03
D..+I(3)	13	2.37	5	$5.46e-09$	1.66			4.91

expvar	niter	$\kappa_{\approx}$	nV	$\|r_{sol}\|$	CS$[s]$	inf$[s]$	LU$[s]$	sol$[s]$
+I(0)	169	15158.70		$9.78e-07$			2.04	17.44
+I(1)	101	4818.07		$1.68e-09$		1.29	3.56	10.30
D..+I(1)	14	5.82	13	$2.18e-06$	2.12			5.17
+I(2)	77	2715.89		$2.12e-09$		1.29+1.89	7.40	12.63
D..+I(2)	12	2.30	10	$9.43e-07$	2.20			5.81
+I(3)	64	1820.48		$2.64e-09$		1.27+1.87+2.74	13.81	14.57
D..+I(3)	13	2.18	7	$1.09e-08$	2.00			6.79

Academic Problem - 3D Laplace (TIME COST)





	n-iter
I(0)	115
I(1)	69
I(1)+D	15

	n-iter
I(0)	169
I(1)	101
I(1)+D	14

	$CS[s] + inf[s] + LU[s] + sol[s]$	$\sum[s]$
I(0)	0.00+0.00+2.06+8.84	10.90
I(1)	0.00+1.18+4.39+**8.35**	13.92
I(1)+D	1.18+1.18+4.39+**3.36**	10.11

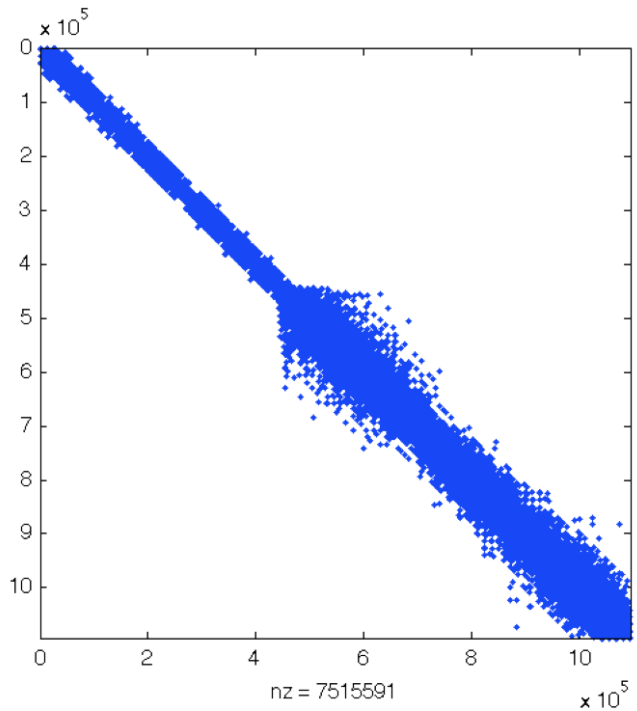
	$CS[s] + inf[s] + LU[s] + sol[s]$	$\sum[s]$
I(0)	0.00+0.00+2.04+17.84	19.88
I(1)	0.00+1.29+3.56+**10.30**	15.15
I(1)+D	2.12+1.29+3.56+**5.17**	12.14

Real Test Case - SPE10 Benchmark

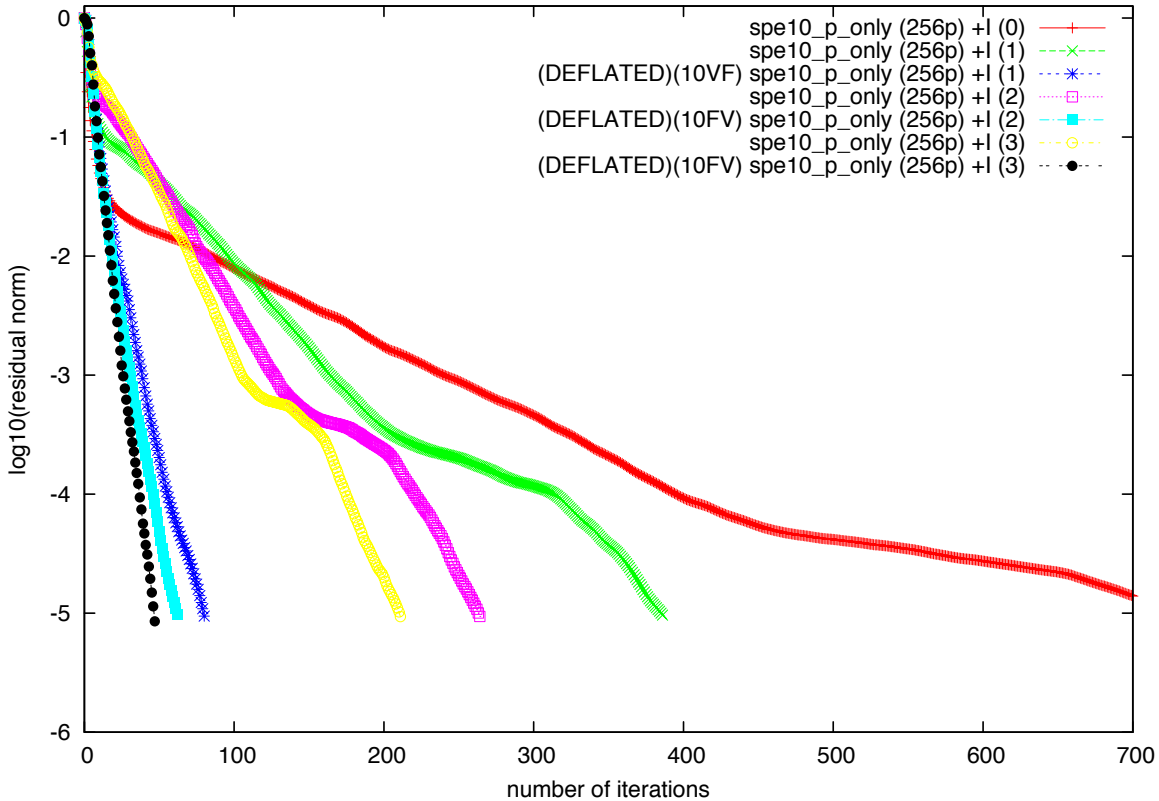
[from SPE10 description] .. the aim of the SPE10 is to simulate porous media flow in a highly heterogeneous black oil reservoir that is described by a fine-scale 1 million cell geological model.

Matrix spe10_p_only.mtx

nparts	nrows	nnz
256	1,094,421	7,515,591

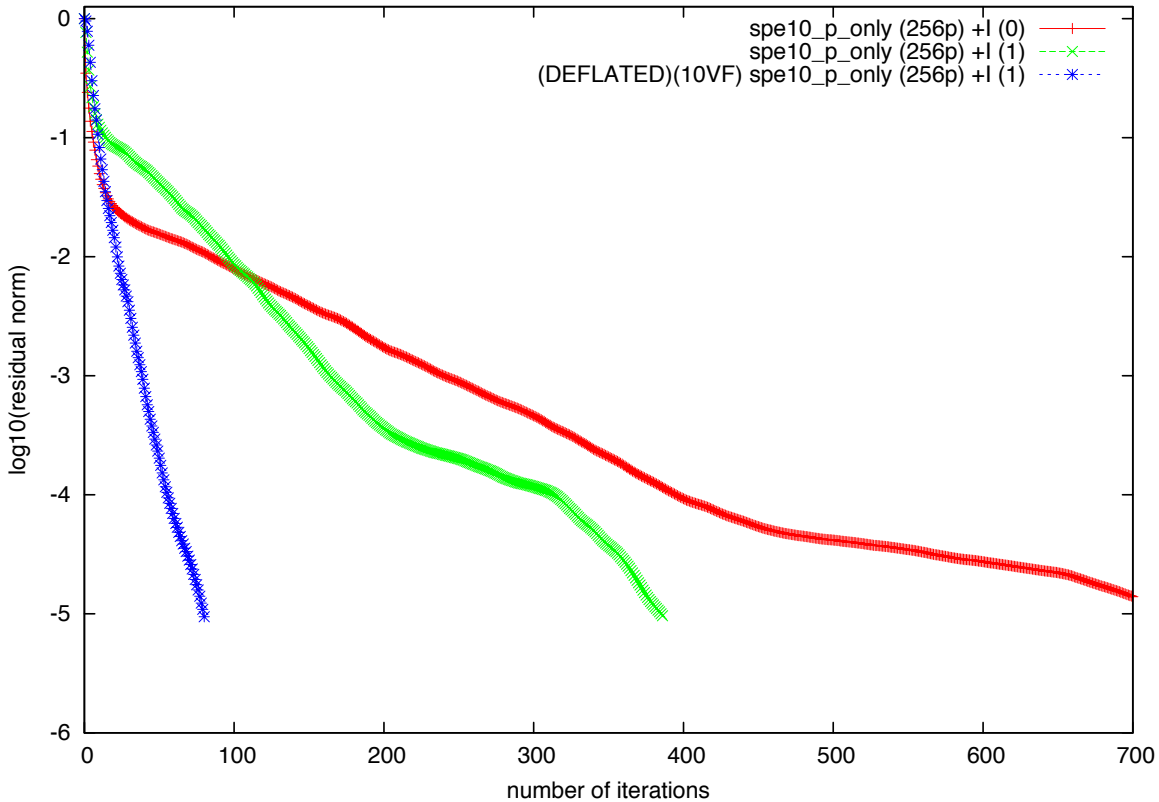


IFP Matrices Collection (spe10)

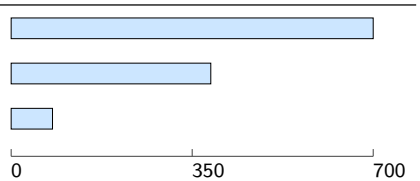


expvar	niter	$\kappa_{\approx}$	nV	$\|r_{sol}\|$	CS$[s]$	inf$[s]$	LU$[s]$	sol$[s]$
+I(0)	700	83147.80		$6.40e-05$			1.36	93.34
+I(1)	386	15169.30		$3.32e-08$		1.09	2.24	42.14
D..+I(1)	80	185.10	10F	$1.83e-09$	1.30			22.43
+I(2)	264	7825.55		$3.84e-08$		1.09+1.54	3.38	22.76
D..+I(2)	62	97.08	10F	$1.13e-08$	1.48			14.39
+I(3)	211	5055.91		$5.02e-09$		1.09+1.54+2.20	5.06	17.32
D..+I(3)	47	50.12	10F	$2.22e-09$	1.61			14.22

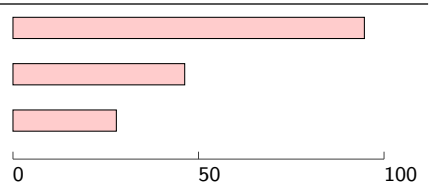
IFP Matrices Collection (spe10)



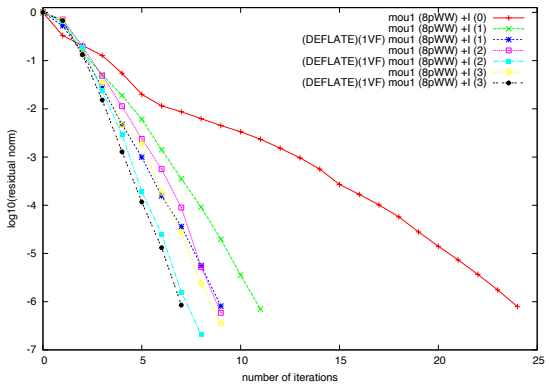
	n-iter
I(0)	700
I(1)	386
I(1)+D	80

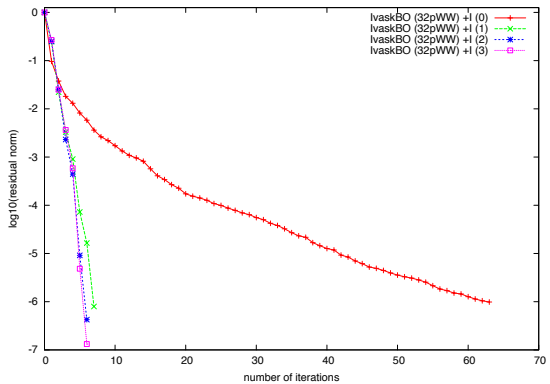


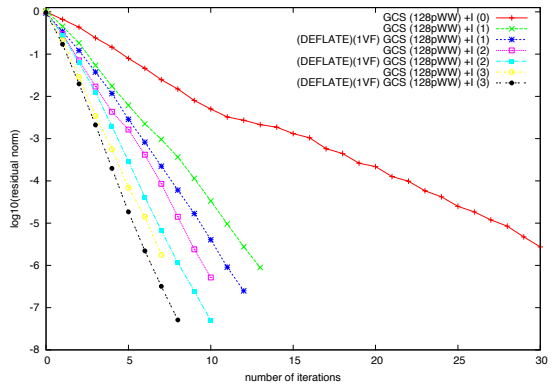
	$CS[s] + inf[s] + LU[s] + sol[s]$	$\sum[s]$
I(0)	0.00+0.00+1.36+93.34	94.70
I(1)	0.00+1.90+2.24+**42.14**	46.28
I(1)+D	1.30+1.90+2.24+**22.43**	27.87

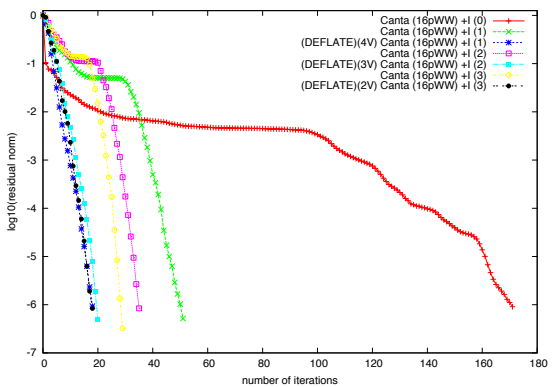


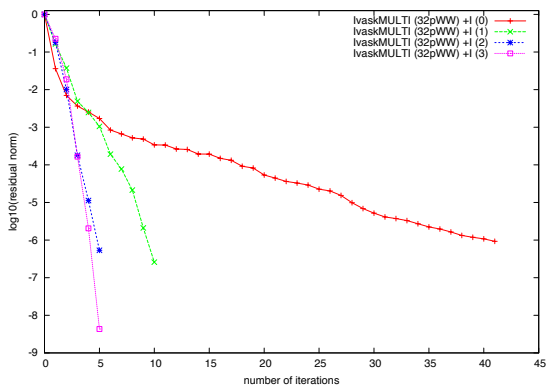
IFP Matrices Collection (ALL)

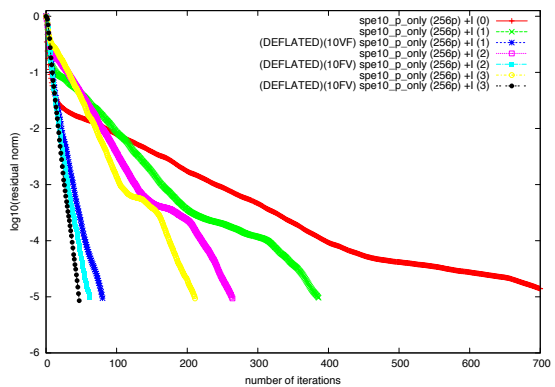




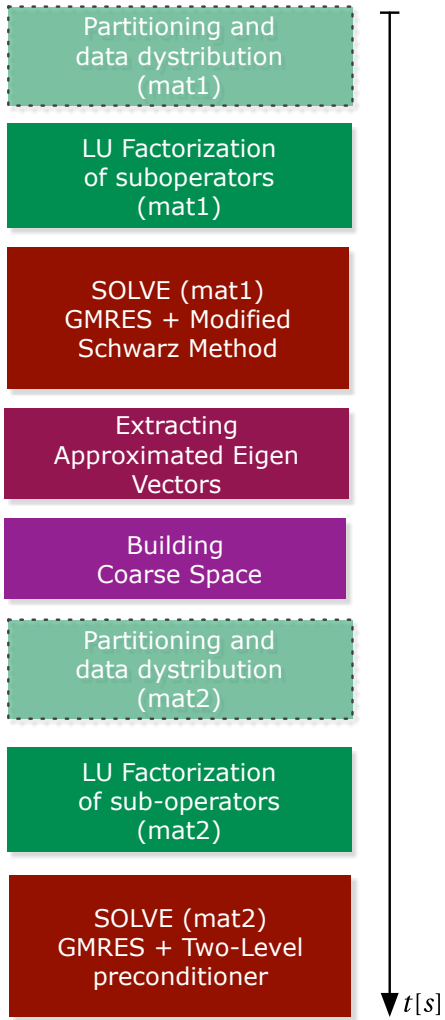








Black Oil Simulation



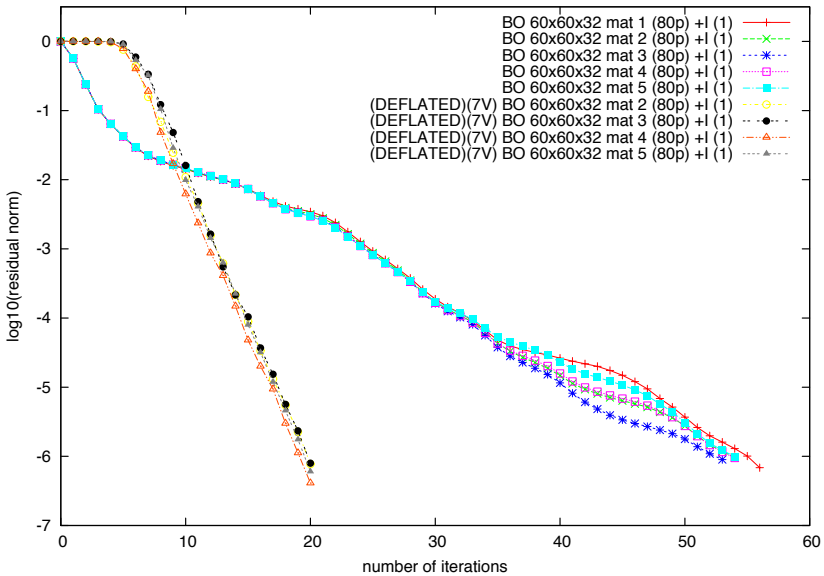
Matrix BO_60x60x32_matX.mtx

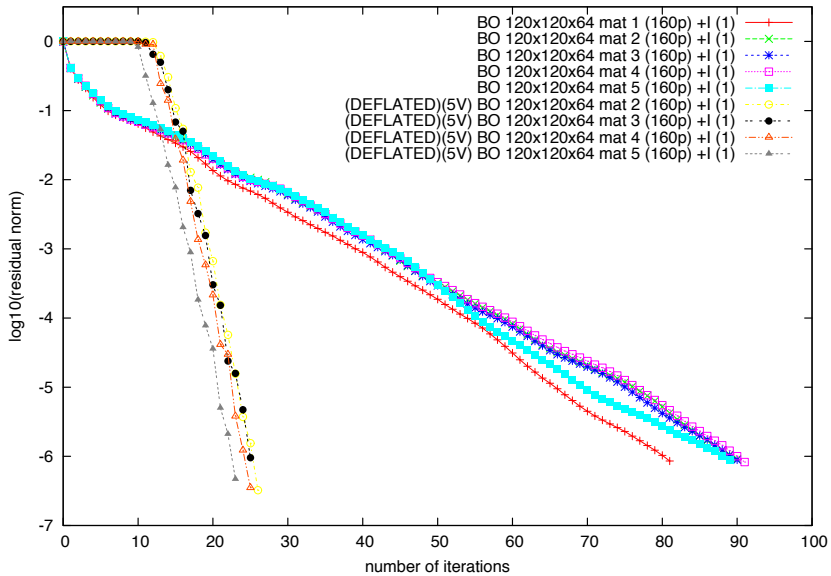
mat_name	nparts	nrows	nnz
BO_60x60x32_mat1.mtx	80	115,200	791,520
BO_60x60x32_mat2.mtx	80	115,200	791,572
BO_60x60x32_mat3.mtx	80	115,200	791,598
BO_60x60x32_mat4.mtx	80	115,200	791,500
BO_60x60x32_mat5.mtx	80	115,200	791,512

Matrix BO_120x120x64_matX.mtx

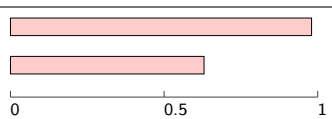
mat_name	nparts	nrows	nnz
BO_120x120x64_mat1.mtx	160	921,600	6,391,680
BO_120x120x64_mat2.mtx	160	921,600	6,391,680
BO_120x120x64_mat3.mtx	160	921,600	6,391,680
BO_120x120x64_mat4.mtx	160	921,600	6,390,986
BO_120x120x64_mat5.mtx	160	921,600	6,387,222

Black Oil Simulation

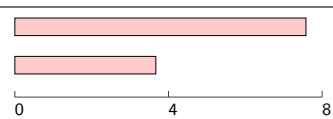




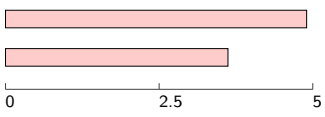
	$sol[s]$
ASM+I(1)	0.98
ASM+I(1)+D	0.63



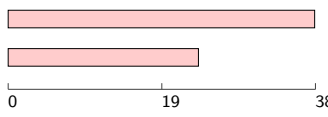
	$sol[s]$
ASM+I(1)	7.58
ASM+I(1)+D	3.67



		$\sum[s]$
ASM+I(1)	5×0.98	4.90
ASM+I(1)+D	1×0.98 + 0.12 + 4×0.63	3.62

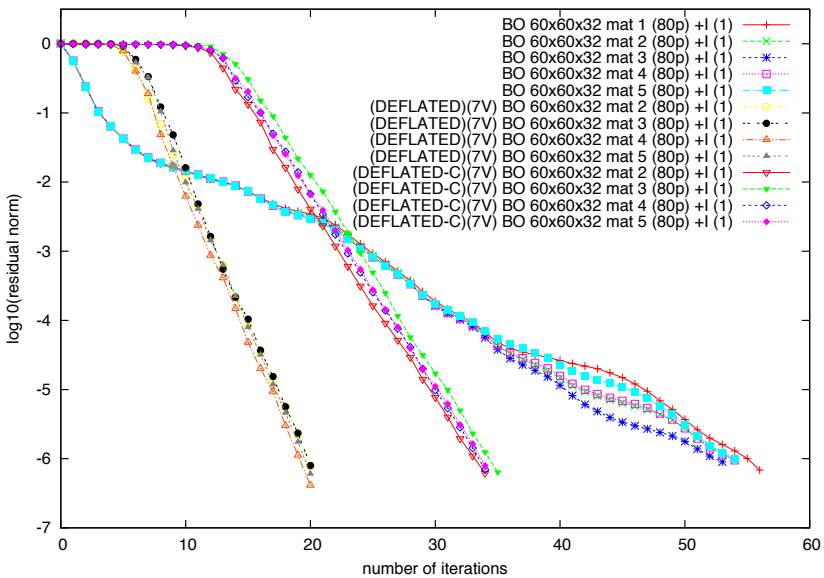


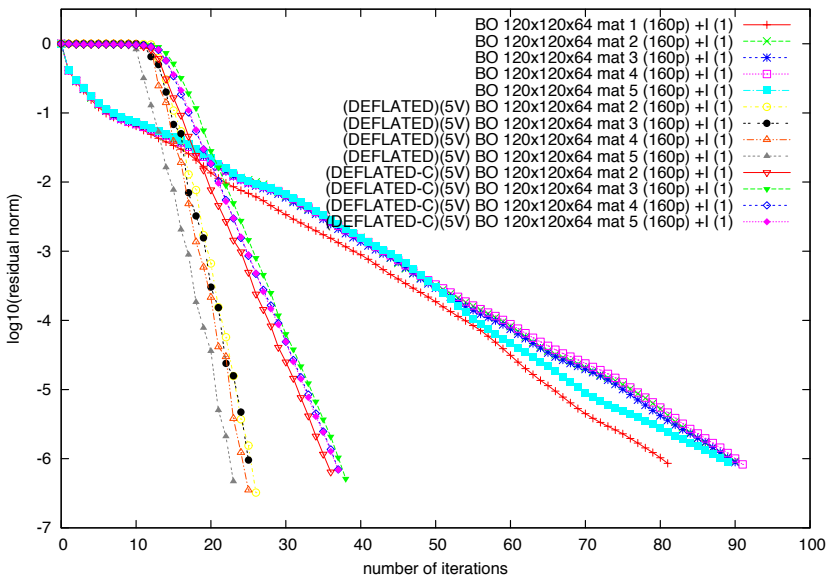
		$\sum[s]$
ASM+I(1)	5×7.58	37.90
ASM+I(1)+D	1×7.58 + 1.27 + 4×3.67	23.53

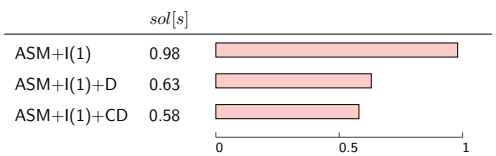


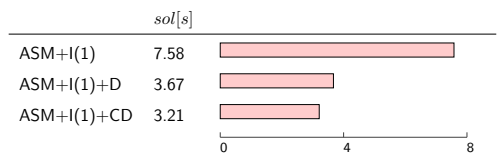
Black Oil Simulation - Reduced Formula

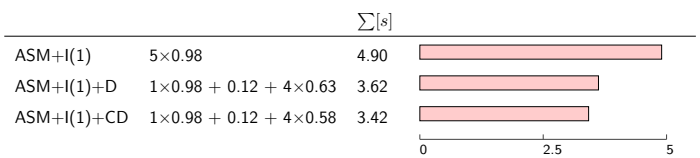
$$\widetilde{\mathcal{P}}_R := \left[\mathbb{I} - \cancel{\left(ZE_R^{-1}Z^T\right)AM^{-1}} + \left(ZE_R^{-1}Z^T\right)\right]$$

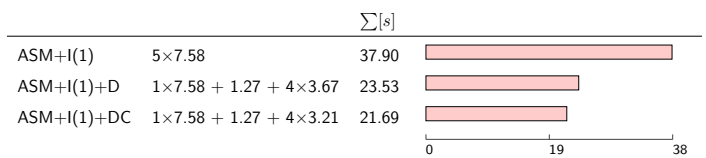












Conclusion and Prospects

Conclusion and Prospects

1. We have considered the extended and the original linear system arising from the domain decomposition method with overlapping.
2. We applied the two-level preconditioner using Schwarz algorithm and the coarse grid correction
3. The coarse grid space is based on the approximated (sub-domain wise split) eigenvectors
   - its size can be adapted to the difficulty of the problem
4. All presented methods are as algebraic as possible which paves the way to extension to systems of equations e.g. multiphase flows
5. Proposed two-level preconditioner is scalable and can be very robust in respect to number of iteration
6. Both methods are adaptive and can be used during first solve that is even the first solve is not completed
7. All methods work for arbitrary decomposition
   - which quality we can improve using weighted graph partitioning

Conclusion and Prospects

1. We have considered the extended and the original linear system arising from the domain decomposition method with overlapping.
2. We applied the two-level preconditioner using Schwarz algorithm and the coarse grid correction
3. The coarse grid space is based on the approximated (sub-domain wise split) eigenvectors
   - its size can be adapted to the difficulty of the problem
4. All presented methods are as algebraic as possible which paves the way to extension to systems of equations e.g. multiphase flows
5. Proposed two-level preconditioner is scalable and can be very robust in respect to number of iteration
6. Both methods are adaptive and can be used during first solve that is even the first solve is not completed
7. All methods work for arbitrary decomposition
   - which quality we can improve using weighted graph partitioning

Conclusion and Prospects

1. We have considered the extended and the original linear system arising from the domain decomposition method with overlapping.
2. We applied the two-level preconditioner using Schwarz algorithm and the coarse grid correction
3. The coarse grid space is based on the approximated (sub-domain wise split) eigenvectors
   - its size can be adapted to the difficulty of the problem
4. All presented methods are as algebraic as possible which paves the way to extension to systems of equations e.g. multiphase flows
5. Proposed two-level preconditioner is scalable and can be very robust in respect to number of iteration
6. Both methods are adaptive and can be used during first solve that is even the first solve is not completed
7. All methods work for arbitrary decomposition
   - which quality we can improve using weighted graph partitioning

Conclusion and Prospects

1. We have considered the extended and the original linear system arising from the domain decomposition method with overlapping.
2. We applied the two-level preconditioner using Schwarz algorithm and the coarse grid correction
3. The coarse grid space is based on the approximated (sub-domain wise split) eigenvectors
   - its size can be adapted to the difficulty of the problem
4. All presented methods are as algebraic as possible which paves the way to extension to systems of equations e.g. multiphase flows
5. Proposed two-level preconditioner is scalable and can be very robust in respect to number of iteration
6. Both methods are adaptive and can be used during first solve that is even the first solve is not completed
7. All methods work for arbitrary decomposition
   - which quality we can improve using weighted graph partitioning

Conclusion and Prospects

1. We have considered the extended and the original linear system arising from the domain decomposition method with overlapping.
2. We applied the two-level preconditioner using Schwarz algorithm and the coarse grid correction
3. The coarse grid space is based on the approximated (sub-domain wise split) eigenvectors
   - its size can be adapted to the difficulty of the problem
4. All presented methods are as algebraic as possible which paves the way to extension to systems of equations e.g. multiphase flows
5. Proposed two-level preconditioner is scalable and can be very robust in respect to number of iteration
6. Both methods are adaptive and can be used during first solve that is even the first solve is not completed
7. All methods work for arbitrary decomposition
   - which quality we can improve using weighted graph partitioning

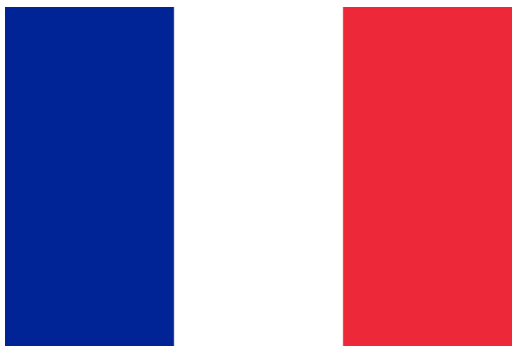
Conclusion and Prospects

1. We have considered the extended and the original linear system arising from the domain decomposition method with overlapping.
2. We applied the two-level preconditioner using Schwarz algorithm and the coarse grid correction
3. The coarse grid space is based on the approximated (sub-domain wise split) eigenvectors
   - its size can be adapted to the difficulty of the problem
4. All presented methods are as algebraic as possible which paves the way to extension to systems of equations e.g. multiphase flows
5. Proposed two-level preconditioner is scalable and can be very robust in respect to number of iteration
6. Both methods are adaptive and can be used during first solve that is even the first solve is not completed
7. All methods work for arbitrary decomposition
   - which quality we can improve using weighted graph partitioning

Conclusion and Prospects

1. We have considered the extended and the original linear system arising from the domain decomposition method with overlapping.
2. We applied the two-level preconditioner using Schwarz algorithm and the coarse grid correction
3. The coarse grid space is based on the approximated (sub-domain wise split) eigenvectors
   - its size can be adapted to the difficulty of the problem
4. All presented methods are as algebraic as possible which paves the way to extension to systems of equations e.g. multiphase flows
5. Proposed two-level preconditioner is scalable and can be very robust in respect to number of iteration
6. Both methods are adaptive and can be used during first solve that is even the first solve is not completed
7. All methods work for arbitrary decomposition
   - which quality we can improve using weighted graph partitioning

Dziekuje za wasza uwage!



Merci de votre attention!

Thank you for your attention!

Bibliography I


Buluç, A., Fineman, J. T., Frigo, M., Gilbert, J. R., and Leiserson, C. E. (2009).
Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks.
In *SPAA '09: Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 233–244, New York, NY, USA. ACM.


Lions, P.-L. (1990).
On the Schwarz alternating method. III: a variant for nonoverlapping subdomains.
In Chan, T. F., Glowinski, R., Périaux, J., and Widlund, O., editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations ,held in Houston, Texas, March 20-22, 1989*, Philadelphia, PA. SIAM.

Bibliography II


Magoulès, F., Roux, F. X., and Series, L. (2006).
Algebraic approximation of dirichlet-to-neumann maps for equations of linear elasticity.
Comp. Meth. Appl. Mech. Engrg., 195:3742–3759.


Nataf, F., Rogier, F., and de Sturler, E. (1994).
Optimal interface conditions for domain decomposition methods.
Technical Report 301, CMAP (Ecole Polytechnique).


Nicolaides, R. A. (1987).
Deflation of conjugate gradients with applications to boundary value problems.
SIAM J. Matrix Anal. Appl., 24:355–365.


Schwarz, H. A. (1870).
Über einen Grenzübergang durch alternierendes Verfahren.
Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15:272–286.

Bibliography III


Smith, B. F., Bjørstad, P. E., and Gropp, W. (1996).
Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.
Cambridge University Press.


Stüben, K. (2001).
A review of algebraic multigrid.
Journal of Computational and Applied Mathematics, 128:281–309.

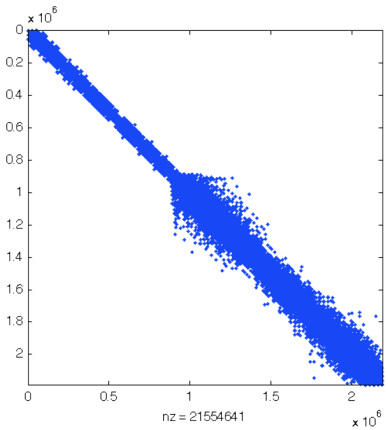

Tang, J. M., Nabben, R., Vuik, C., and Erlangga, Y. A. (2009).
Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods.
J. Sci. Comput., 39:340–370.

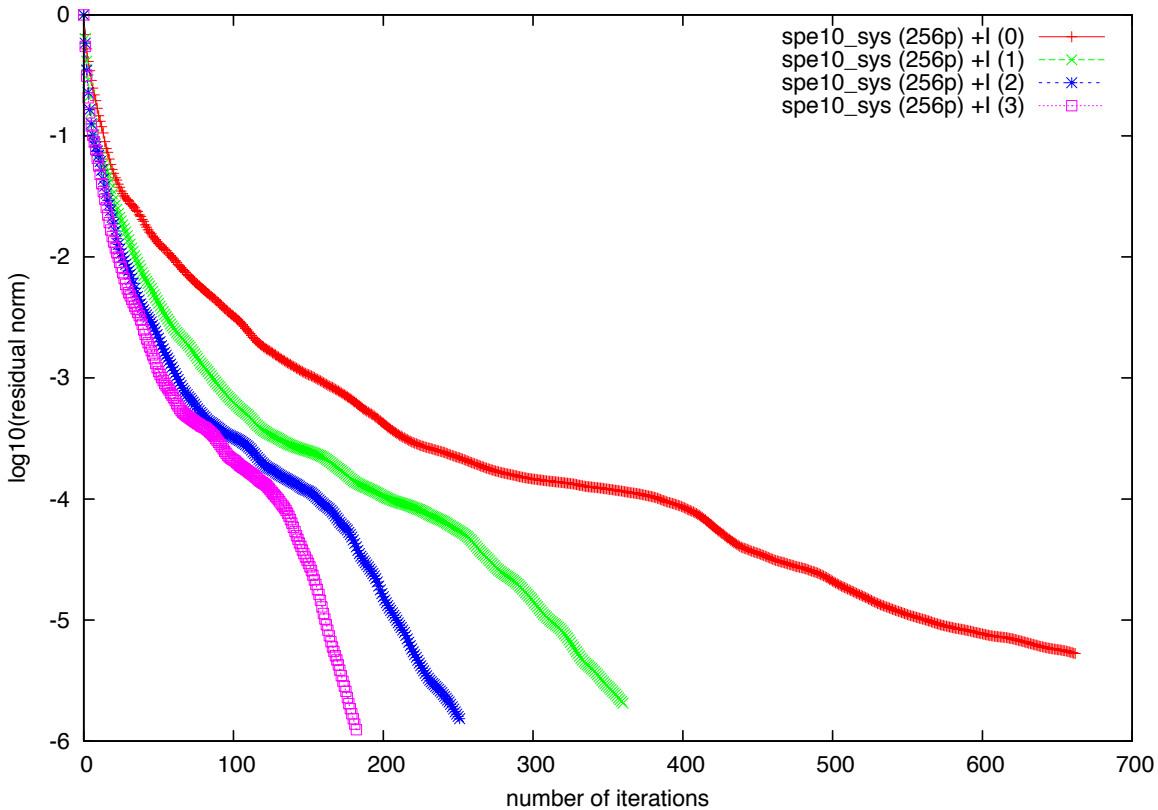
Additional Numerical Experiments

IFP Matrices Collection (spe10 - system of equations)

Matrix `spe10_sys.mtx`

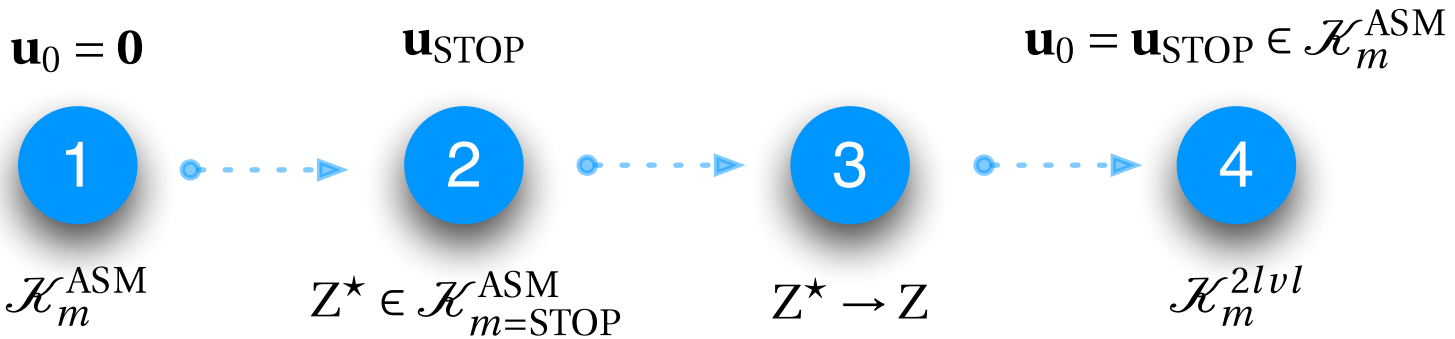
nparts	nrows	nnz
256	2,188,842	21,554,641





expvar	niter	$\|r_{sol}\|$	inf$[s]$	LU$[s]$	sol$[s]$
+I(0)	663	$1.43e-06$		≈ 4	343.13
+I(1)	361	$2.12e-09$	4.36	≈ 6	70.76
+I(2)	252	$3.81e-08$	4.37 + 7.23	≈ 13	70.88
+I(3)	183	$2.03e-08$	4.52 + 7.42 + 11.00	≈ 24	78.14

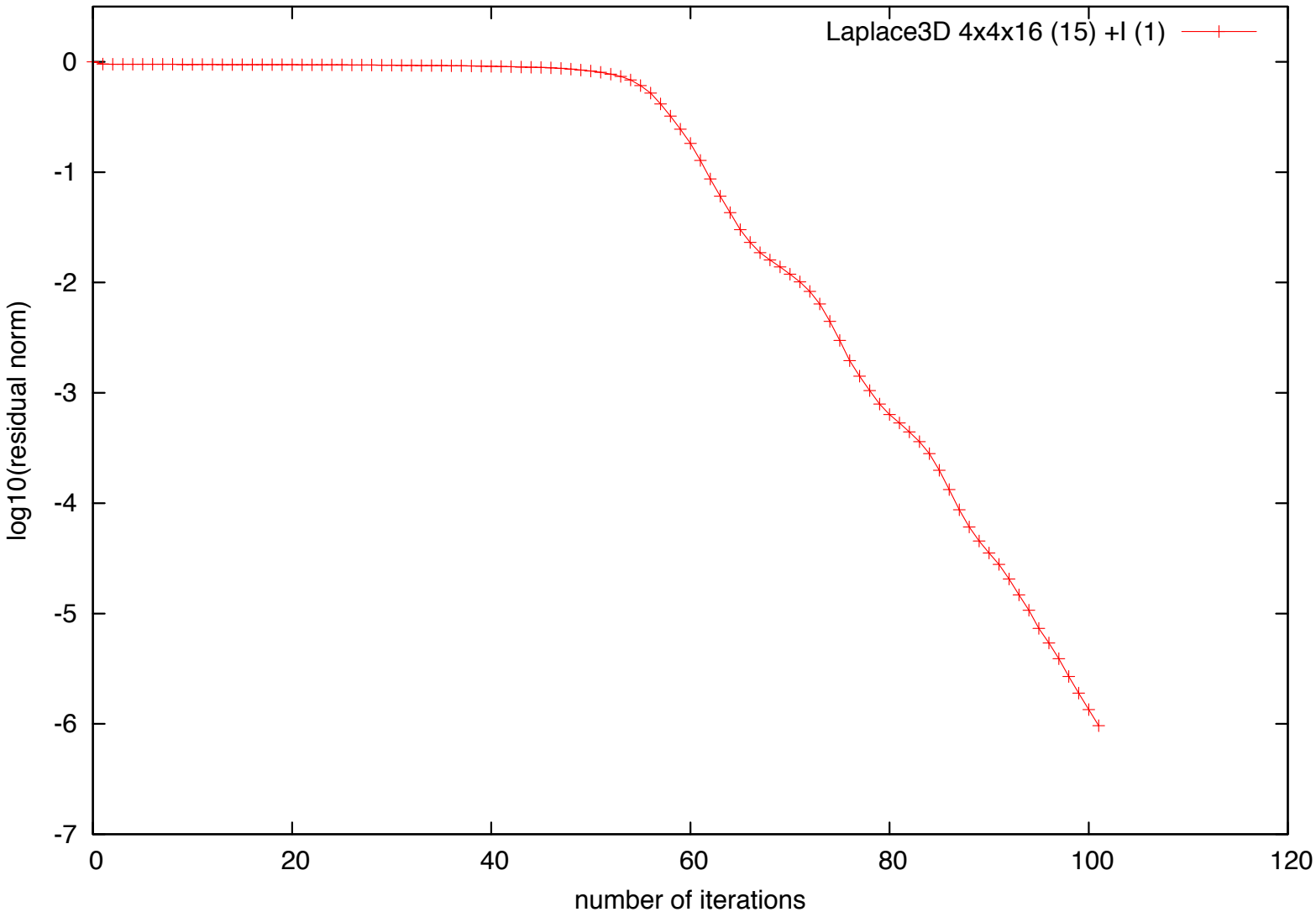
Adaptive Solver



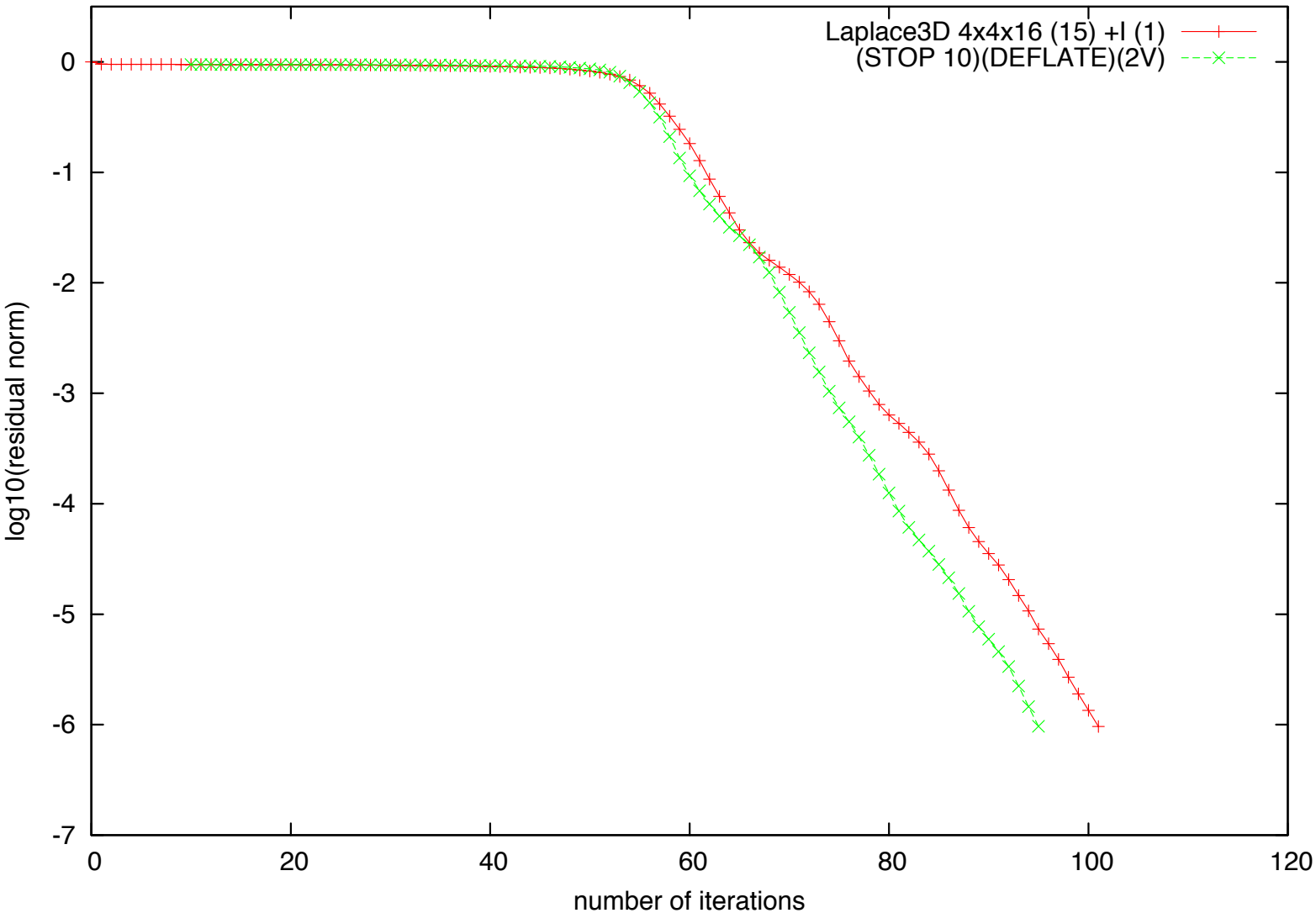
$$\mathscr{K}_m^{\text{ASM}}\left(\mathrm{AM}^{-1}, \mathbf{r}_0\right) := \mathbf{u}_0 + \mathtt{SPAN}\left\{\mathbf{r}_0, (\mathrm{AM}^{-1})^{m-1}\mathbf{r}_0\right\}$$

$$\mathscr{K}_m^{2lvl}\left(\mathrm{AM}^{-1}\mathscr{P}_{\mathrm{R}}, \mathbf{r}_0\right) := \mathbf{u}_0 + \mathtt{SPAN}\left\{\mathbf{r}_0, (\mathrm{AM}^{-1}\mathscr{P}_{\mathrm{R}})^{m-1}\mathbf{r}_0\right\}$$

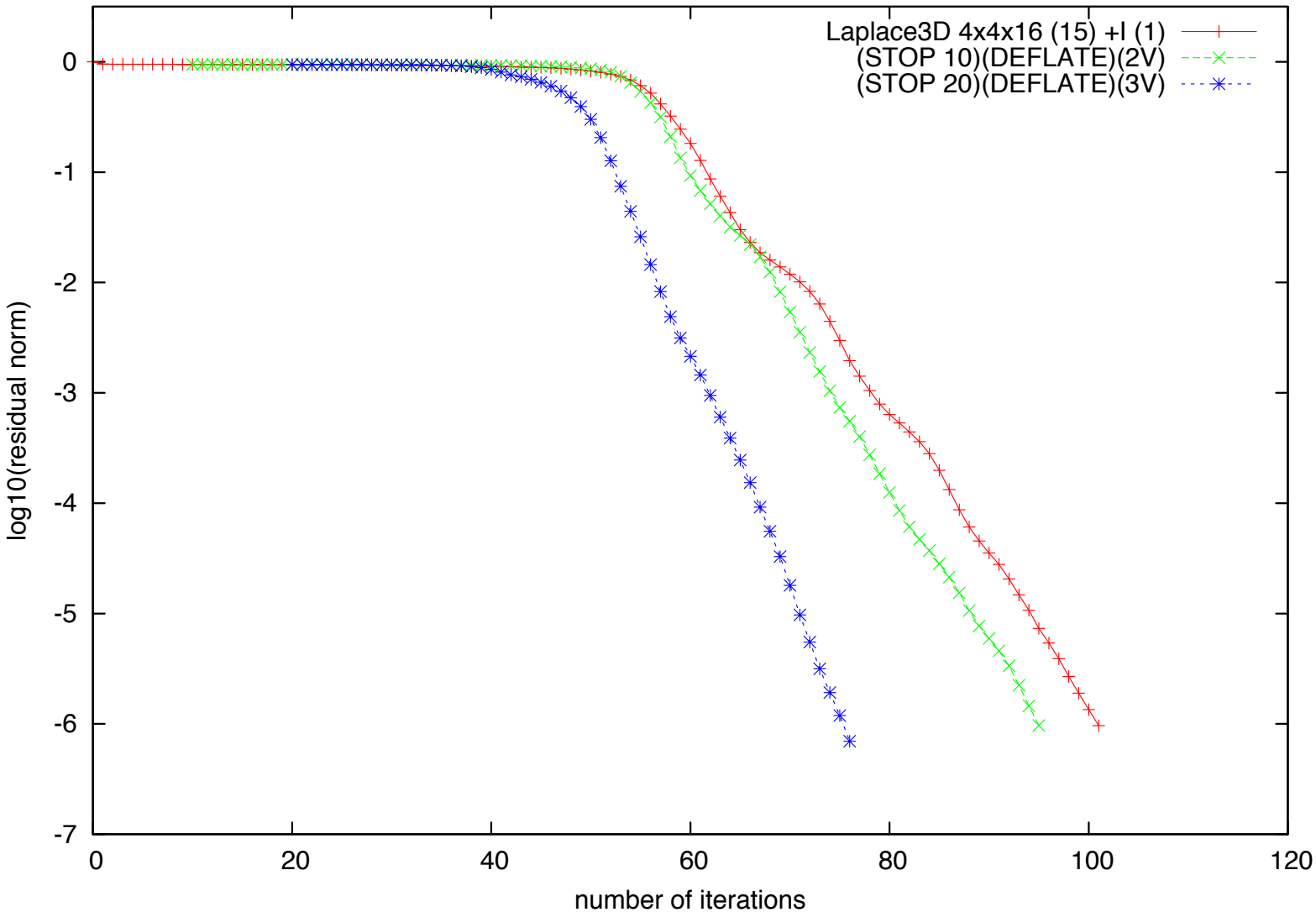
Adaptive Solver - Stop after XX



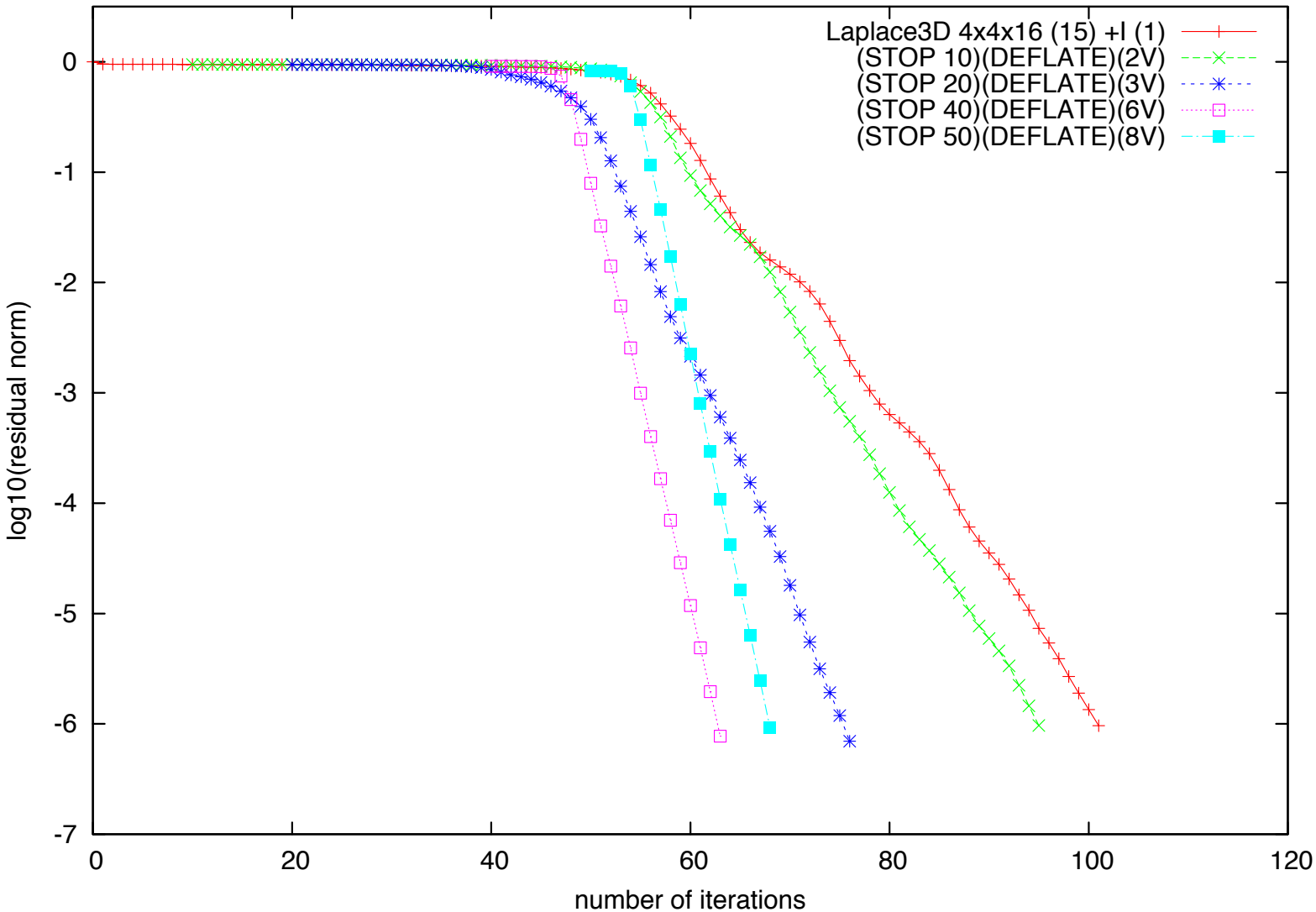
Adaptive Solver - Stop after 10



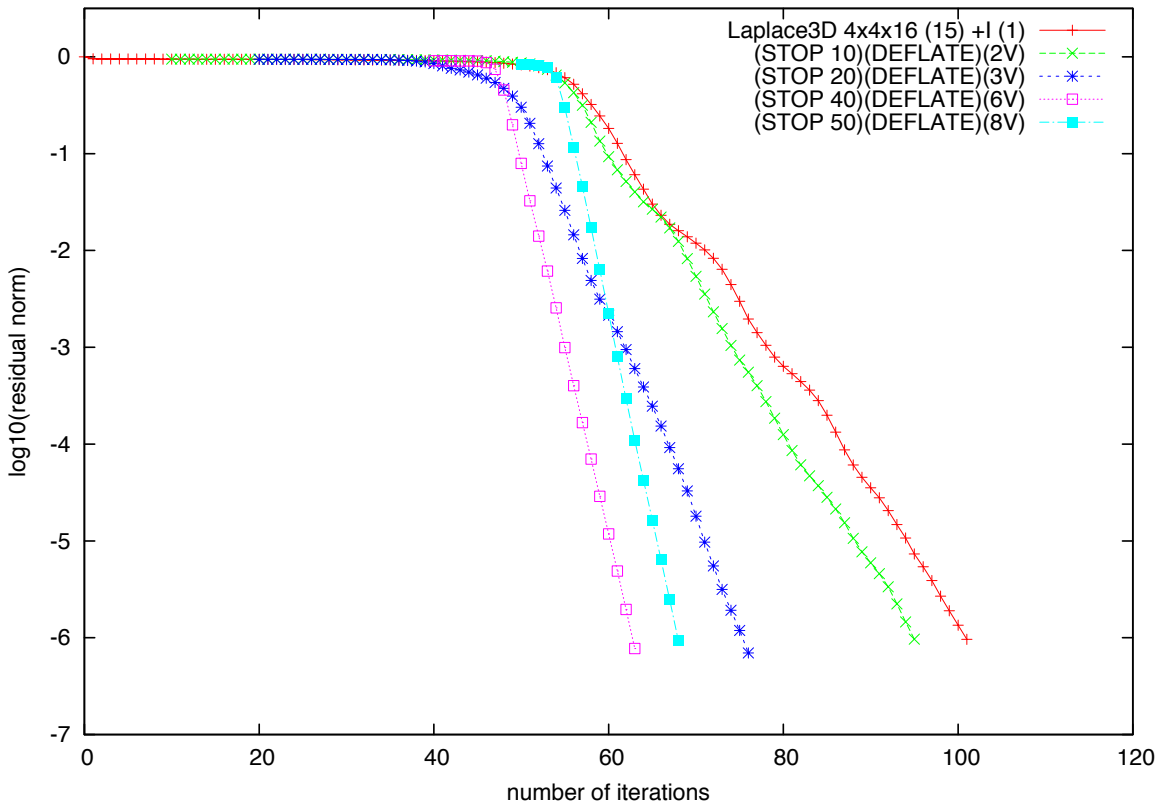
Adaptive Solver - Stop after 20



Adaptive Solver - Stop after (40 and 50)

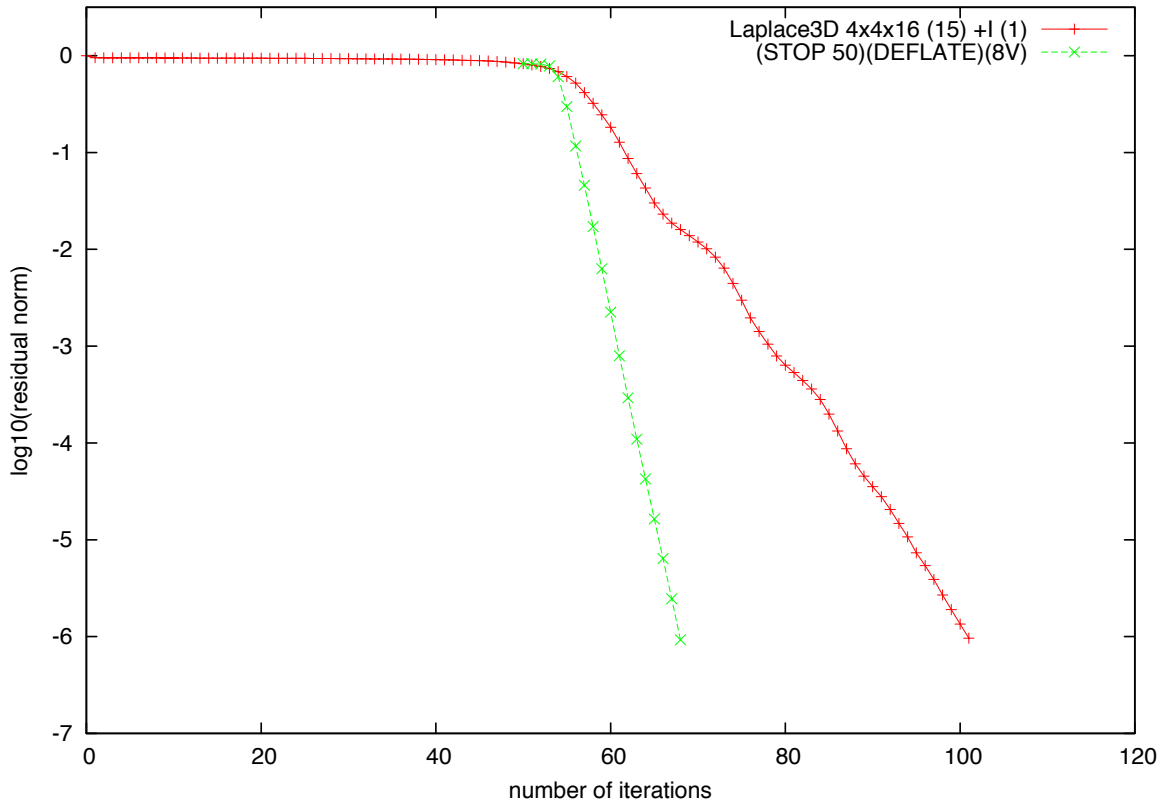


Adaptive Solver vs Time of Computation

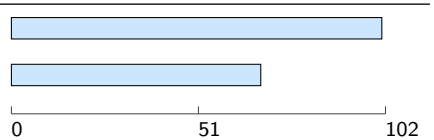


expvar	niter	$\kappa_{\approx}$	nV	$\|r_{sol}\|$	CS$[s]$	inf$[s]$	LU$[s]$	sol$[s]$
+I(1)	101	4818.07		$1.68e-09$		1.29	3.56	10.30
S10	10 + 85	2427.91	2	$2.34e-09$	0.25	1.38	4.21	3.43 + 12.15
S20	20 + 56	648.32	3	$1.22e-09$	0.37	1.31	4.17	3.52 + 11.77
S40	40 + 23	7.68	6	$9.53e-10$	1.15	1.32	5.47	3.73 + 7.70
S50	50 + 18	4.12	8	$2.69e-10$	1.01	1.29	3.76	3.91 + 3.82

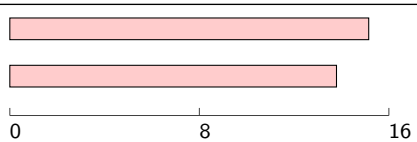
Adaptive Solver vs Time of Computation



	n-iter
I(1)	101
AI(1)	68



	$CS[s] + inf[s] + LU[s] + sol[s]$	$\sum[s]$
I(1)	0.00+1.29+3.56+10.30	15.15
AI(1)	1.01+1.29+3.76+7.73	13.79



Improved Diagonal Approximation (EDOIC)

Improved Diagonal Approximation (EDOIC)

Let $\beta_{\widetilde{1}_c\widetilde{1}_c}$ be a symmetric sparse operator which satisfies

$$\beta_{\widetilde{1}_c\widetilde{1}_c} A_{\widetilde{1}_c\widetilde{1}_c} V_{\widetilde{1}_c} = V_{\widetilde{1}_c}, \quad \text{or equivalently} \quad -\beta_{\widetilde{1}_c\widetilde{1}_c} A_{\widetilde{1}_c\widetilde{\Gamma}_1} V_{\widetilde{\Gamma}_1} = V_{\widetilde{1}_c}.$$

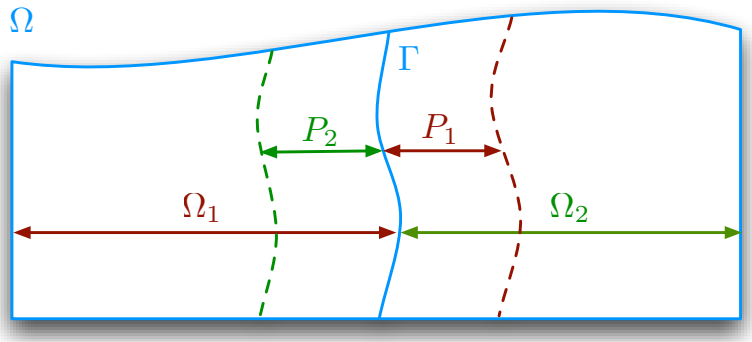
The optimal interface conditions operator $S^{opt}_{\widetilde{\Gamma}_1\widetilde{\Gamma}_1}$ is approximated by

$$S^{opt}_{\widetilde{\Gamma}_1\widetilde{\Gamma}_1} \approx S^{edoic}_{\widetilde{\Gamma}_1\widetilde{\Gamma}_1} := -A_{\widetilde{\Gamma}_1\widetilde{1}_c} \left(2\beta_{\widetilde{1}_c\widetilde{1}_c} - \beta_{\widetilde{1}_c\widetilde{1}_c} A_{\widetilde{1}_c\widetilde{1}_c} \beta_{\widetilde{1}_c\widetilde{1}_c}\right) A_{\widetilde{1}_c\widetilde{\Gamma}_1}$$

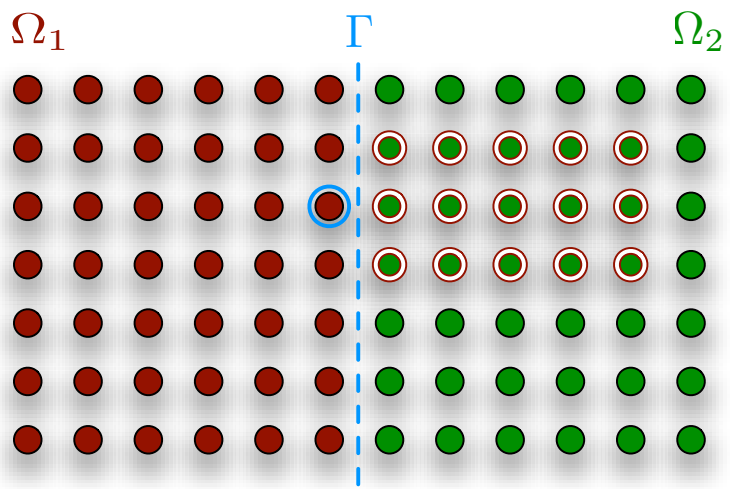
The idea of this improvement originates from the following calculations: $\|(\mathcal{B}\mathcal{A}-I)\| \leq \epsilon < 1$ *leads to* $\|(\mathcal{B}\mathcal{A}-I)^2\| \leq \epsilon^2 < \epsilon$. *Then, remarking that* $(\mathcal{B}\mathcal{A}-I)^2 = \mathcal{B}\mathcal{A}\mathcal{B}\mathcal{A} - 2\mathcal{B}\mathcal{A} + I = I - (2\mathcal{B} - \mathcal{B}\mathcal{A}\mathcal{B})\mathcal{A}$, *one concludes that* $\mathcal{C} = 2\mathcal{B} - \mathcal{B}\mathcal{A}\mathcal{B}$ *is better approximation of* $\mathcal{A}^{-1}$ *than* $\mathcal{B}$ *since* $\|\mathcal{C} - I\| \leq \epsilon^2 < \epsilon$.

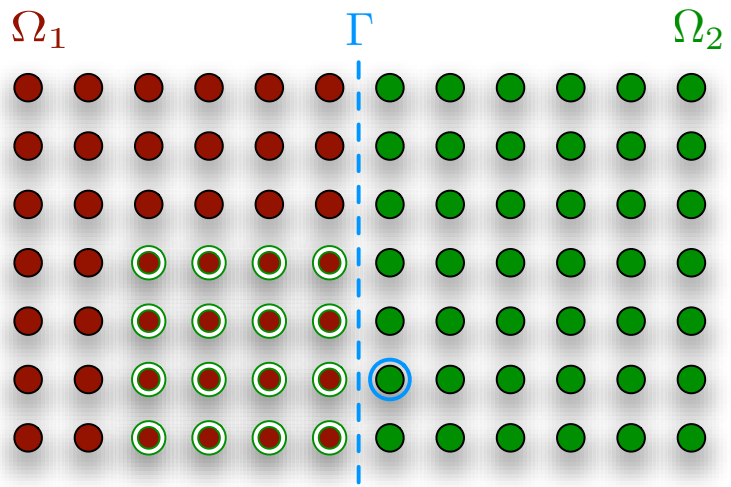
Sparse Patch Method

Sparse Patch Method



$$A_P^j = \begin{pmatrix} \overline{A}_{jj} & \overline{A}_{j\Gamma} \\ \overline{A}_{\Gamma j} & \overline{A}^j_{\Gamma\Gamma} \end{pmatrix}$$





See [Magoulès et al., 2006] for more informations.