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

1 Motivation
   Why Domain Decomposition Methods?

2 State of the Arts
   Classical Methods
   Algebraic Formulation
   Modified Schwarz Method
   Schwarz Methods vs. Many Subdomains
   State of the Arts Summary

3 New Methods and Techniques
   ADDM-lib for parallel computing
   Partitioning with weights
   Overlap via extended matrix
   Modified Schwarz Method
     Diagonal Approximation of the Optimal Interface Conditions
   Two-level preconditioner

4 Conclusion and Prospects
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**

- **Direct Solvers**
  - Difficult to parallelize
  - Use a lot of memory
  - But are robust
  - And work as black-box solver

- **DDM**
  - Naturally parallel
  - Flexible compromise
  - Can be no efficient
  - Flexible compromise

- **Krylov Methods**
  - Low memory usage
  - Need good preconditioner to be robust
  - Also work as black-box solver

**Comparison**

- **Direct Solvers**
  - Difficult to parallelize
  - Use a lot of memory
  - But are robust
  - And work as black-box solver

- **Krylov Methods**
  - Easy to parallelize
  - Low memory usage
  - Need good preconditioner to be robust
  - Also work as black-box solver

**Conclusion**

- Lack of black-box routines

**Prospects**

- Future developments
  - Improved algorithms
  - Enhanced scalability
  - Integration with new technologies
**DDM vs. other methods**

**Direct Solvers**
- Difficult to parallelize
- Use a lot of memory
- But are robust
- And work as black-box solver

**Krylov Methods**
- Easy to parallelize
- Low memory usage
- Need good preconditioner to be robust
- Also work as black-box solver

**DDM**
- Naturally parallel
- Flexible compromise
- Can be no efficient
- Lack of black-box routines

**Motivation** State of the Arts New Methods and Techniques Conclusion and Prospects
DDM vs. other methods

Direct Solvers
- difficult to parallelize
- use a lot of memory
- but are robust
- and work as black-box solver

DDM
- naturally parallel
- flexible compromise
- can be no efficient

Krylov Methods
- low memory usage
- need good preconditioner to be robust
- easy to parallelize
- also work as black-box solver

- lack of black-box routines
State of the Arts
Original Method

\[ \begin{align*}
-\Delta(u) &= f \quad \text{in} \quad \Omega \\
\mathbf{u} &= \mathbf{g} \quad \text{on} \quad \partial\Omega
\end{align*} \]

Alternating Schwarz Method

\[ \begin{align*}
-\Delta(u_1^{n+1}) &= f \quad \text{in} \quad \Omega_1 \\
u_1^{n+1} &= g \quad \text{on} \quad \partial\Omega_1 \setminus \Gamma_1 \\
u_1^{n+1} &= u_2^n \quad \text{on} \quad \Gamma_1 \\
-\Delta(u_2^{n+1}) &= f \quad \text{in} \quad \Omega_2 \\
u_2^{n+1} &= g \quad \text{on} \quad \partial\Omega_2 \setminus \Gamma_2 \\
u_2^{n+1} &= u_1^{n+1} \quad \text{on} \quad \Gamma_2.
\end{align*} \]

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

\[-\Delta(u) = f \text{ in } \Omega\]
\[u = g \text{ on } \partial\Omega\]

\[-\Delta(u_{n+1}^1) = f \text{ in } \Omega_1\]
\[u_{n+1}^1 = g \text{ on } \partial\Omega_1 \setminus \Gamma_1\]
\[u_{n+1}^1 = u_n^2 \text{ on } \Gamma_1.\]

\[-\Delta(u_{n+1}^2) = f \text{ in } \Omega_2\]
\[u_{n+1}^2 = g \text{ on } \partial\Omega_2 \setminus \Gamma_2\]
\[u_{n+1}^2 = u_n^1 \text{ on } \Gamma_2.\]
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 $\kappa$ of operator $A$, preconditioned by $P_{as}$ i.e., ASM with the coarse grid correction, satisfies

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

where the constant $C$ is independent of, $H$ and $\delta$. 
Algebraic Formulation (Jacobi and Schwarz)

Let's consider a discretized problem which yields a linear system.

\[-\Delta(u) = f \quad \text{in} \quad \Omega\]
\[u = g \quad \text{on} \quad \partial \Omega\]

For the set of indices \(\Omega\) partitioned into two sets \(\Omega_1\) and \(\Omega_2\) 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)

Let's consider a discretized problem which yields a linear system.

\[-\Delta(u) = f \quad \text{in} \quad \Omega \]
\[u = g \quad \text{on} \quad \partial\Omega\]

For the set of indices \(\Omega\) partitioned into two sets \(\Omega_1\) and \(\Omega_2\) 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 \(\implies\) \textbf{Schwarz method with minimal overlap}. 
Preconditioned Krylov space

\[
\begin{bmatrix}
U_1^{n+1} \\
U_2^{n+1}
\end{bmatrix} =
\begin{bmatrix}
U_1^n \\
U_2^n
\end{bmatrix} +
\begin{bmatrix}
A_1^{-1} & 0 \\
0 & A_2^{-1}
\end{bmatrix}
\begin{bmatrix}
F_1 \\
F_2
\end{bmatrix} -
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
U_1^n \\
U_2^n
\end{bmatrix}
\]

\[u_{n+1} = u_n + M^{-1}(f - Au_n)\]

Let \(r_0 := M^{-1}(f - Au_0)\), we have (fixed point method):

\[u_n = \sum_{i=0}^{n} (I - M^{-1}A)^i r_0 + u_0\]

A preconditioned Krylov solve will generate an optimal solution \(u^K_n\) in:

\[\mathcal{K}_m (M^{-1}A, r_0) := u_0 + \text{SPAN} \{r_0, M^{-1}Ar_0, \ldots, (M^{-1}A)^{m-1}r_0\}\]

where \(u_n \in \mathcal{K}_m\) but with “frozen” coefficients \(\Rightarrow u_n\) is less optimal then \(u^K_n\).
Modified Schwarz Method

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

\[
\begin{align*}
-\Delta(u) &= f \quad \text{in} \quad \Omega \\
u &= g \quad \text{on} \quad \partial\Omega
\end{align*}
\]

\[
\begin{align*}
-\Delta(u_{1}^{n+1}) &= f & \text{in} \quad \Omega_1 \\
u_1^{n+1} &= g & \text{on} \quad \partial\Omega_1 \setminus \Gamma \\
\left( \frac{\partial}{\partial n_1} + \alpha \right) (u_1^{n+1}) &= \left( \frac{\partial}{\partial n_2} + \alpha \right) (u_n^{n+1}) & \text{on} \quad \partial\Omega_1 \cap \Omega_2 \\
-\Delta(u_2^{n+1}) &= f & \text{in} \quad \Omega_2 \\
u_2^{n+1} &= g & \text{on} \quad \partial\Omega_2 \cap \partial\Gamma \\
\left( \frac{\partial}{\partial n_2} + \alpha \right) (u_2^{n+1}) &= \left( \frac{\partial}{\partial n_1} + \alpha \right) (u_n^{n+1}) & \text{on} \quad \partial\Omega_2 \cap \Omega_1
\end{align*}
\]
J. L. Lions:

“First of all, it is possible to replace the constants in the Robin condition by two proportional functions 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].”
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

\[ u_0 : \Gamma_1 \to \mathbb{R} \]

\[ \text{DtN}_2(u_0) := \nabla v \cdot n_2|_{\partial \Omega_1 \cap \Omega_2}, \]

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

\[ \mathcal{L}(v) = 0 \quad \text{in} \quad \Omega_2 \setminus \overline{\Omega}_1 \]

\[ v = 0 \quad \text{on} \quad \partial \Omega_2 \cap \partial \Omega \]

\[ v = u_0 \quad \text{on} \quad \partial \Omega_1 \cap \overline{\Omega}_2. \]
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 $U_{\Gamma}$:

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

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) $\Rightarrow U_{i}^{\infty} = U_{i}$ and $U_{\Gamma,1}^{\infty} = U_{\Gamma,2}^{\infty} = U_{\Gamma}$. 

\[
\begin{pmatrix}
U_{1}^{n+1} \\
U_{\Gamma,1}^{n+1}
\end{pmatrix}
=
\begin{pmatrix}
F_1 \\
F_{\Gamma} + S_2 U_{\Gamma,2}^{n} - A_{\Gamma 2} U_2^{n}
\end{pmatrix}
\]

\[
\begin{pmatrix}
U_{2}^{n+1} \\
U_{\Gamma,2}^{n+1}
\end{pmatrix}
=
\begin{pmatrix}
F_2 \\
F_{\Gamma} + S_1 U_{\Gamma,1}^{n} - A_{\Gamma 1} U_1^{n}
\end{pmatrix}
\]
Optimal 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 \( \Rightarrow \) \( A_{\Gamma \Gamma} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma} \) 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

\[
\begin{pmatrix}
A_{11} & A_{1\Gamma} \\
A_{\Gamma1} & A_{\Gamma\Gamma} + S_2
\end{pmatrix}
\begin{pmatrix}
U^{n+1}_1 \\
U^{n+1}_{\Gamma,1}
\end{pmatrix}
= 
\begin{pmatrix}
F_1 \\
F_{\Gamma} + S_2U^n_{\Gamma,2} - A_{\Gamma2}U^n_2
\end{pmatrix}
\]

\[
\begin{pmatrix}
A_{22} & A_{2\Gamma} \\
A_{\Gamma2} & A_{\Gamma\Gamma} + S_1
\end{pmatrix}
\begin{pmatrix}
U^{n+1}_2 \\
U^{n+1}_{\Gamma,2}
\end{pmatrix}
= 
\begin{pmatrix}
F_2 \\
F_{\Gamma} + S_1U^n_{\Gamma,1} - A_{\Gamma1}U^n_1
\end{pmatrix}
\]

**Optimal choice**

Taking \( S_1 = -A_{\Gamma1}A_{11}^{-1}A_{1\Gamma} \) and \( S_2 = -A_{\Gamma1}A_{22}^{-1}A_{2\Gamma} \) yields a convergence in two steps \( \Rightarrow A_{\Gamma\Gamma} - A_{\Gamma i}A_{i i}^{-1}A_{i \Gamma} \) 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.

The mean value of the solution in domain $\Omega_i$ depends on the value of $f$ on all subdomains.

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

- can be incorporate as additional preconditioner $\implies$ “two-level preconditioning”.

\[-\Delta(u) = f \text{ in } \Omega \]
\[u = g \text{ on } \partial \Omega\]
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.

\[-\Delta(u) = f \text{ in } \Omega \]
\[u = g \text{ on } \partial\Omega\]

The mean value of the solution in domain \(\Omega_i\) depends on the value of \(f\) on all subdomains.

**A classical remedy:** \(\Rightarrow\) coarse grid problem that couples all subdomains.

- can be incorporate as additional preconditioner \(\Rightarrow\) “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 $Q$.

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

Some properties:

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

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

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

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

Some properties:

- $PA = AP^T$
- $P^TZ = 0, \quad P^TQ$
- $QA = I - P^T, \quad QAZ = Z, \quad QAQ = Q$

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

Example: $\mathcal{P}_{AD} := M^{-1} + Q$, $\mathcal{P}_{BNN} := P^TM^{-1}P + Q$, $\mathcal{P}_{A-DEF} := P^TM^{-1} + Q$
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 be 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 $\Omega$. 
Motivation

State of the Arts

New Methods and Techniques

Conclusion and Prospects

ADDMlib - parallel library

**Figure:** Decomposition of global linear system into Partial Vectors and Operators (very similar idea introduced independently in [Buluç et al., 2009]).
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]).
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 $\bigcup_i V_i = V$, 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 $\bigcup V_i = V$, 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.
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 $\bigcup_i V_i = V$, 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])

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

\(\lfloor x \rfloor\) is the floor function rounds the element \(x\) to the nearest integer toward minus infinity and \(\gamma_{\text{const}}\) 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])

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

\([x]\) is the floor function rounds the element \(x\) to the nearest integer toward minus infinity and \(\gamma_{const}\) is an arbitrary constant.
Let us consider following, anisotropic problem: \(-\kappa \Delta (u) = f\), discretized (FreeFem++) on 2D unit square in size \(N_x \times N_y\), where \(N_x = N_y = 128\).

\[
\kappa = \begin{bmatrix}
\kappa_{xx} & 0 \\
0 & \kappa_{yy}
\end{bmatrix} = \begin{bmatrix}
1 \times 10^{-6} & 0 \\
0 & 1
\end{bmatrix}
\]
Partitioning with weights - Numerical Experiment 2

<table>
<thead>
<tr>
<th>Paritioner</th>
<th>n-iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>(⋆) SCOTCH</td>
<td>26</td>
</tr>
<tr>
<td>(b) SCOTCH + W</td>
<td>26</td>
</tr>
<tr>
<td>(⋆) SCOTCH</td>
<td>45</td>
</tr>
<tr>
<td>(d) SCOTCH + W</td>
<td>34</td>
</tr>
<tr>
<td>(⋆) SCOTCH</td>
<td>42</td>
</tr>
<tr>
<td>(f) SCOTCH + W</td>
<td>32</td>
</tr>
</tbody>
</table>

where $|\kappa| = 1$ and $|\kappa| = 10^{-6}$.
Partitioning with weights - real test case

Matrix IvaskMULTI_p_only.mtx

<table>
<thead>
<tr>
<th>nparts</th>
<th>nrows</th>
<th>nnz</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>49,572</td>
<td>480,612</td>
</tr>
</tbody>
</table>
Partitioning with weights - real test case

-6
-5
-4
-3
-2
-1
0
10 20 30 40 50 60 70 80 90
log10(residual norm)
number of iterations
IvaskMULTI (32p) +I (0)
IvaskMULTI (32pW) +I (0)
29 / 71
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 \textit{inflate} operator

Bigger overlap $\implies$ faster convergence !!
### Enlarge overlap \( \Rightarrow \) algebraically *inflate* operator

Before inflation

\[
\begin{pmatrix}
A_{1,1} & A_{1,2} & A_{1,3} \\
A_{2,1} & A_{2,2} & A_{2,3} \\
A_{3,1} & A_{3,2} & A_{3,3}
\end{pmatrix}
\begin{pmatrix}
U_1 \\
U_2 \\
U_3
\end{pmatrix}
= 
\begin{pmatrix}
F_1 \\
F_2 \\
F_3
\end{pmatrix}
\]
Enlarge overlap $= \text{algebraically inflate operator}$

After inflation

\[
\begin{bmatrix}
A_{1i,1i} & A_{1i, \Gamma_1^2} & 0 & 0 & 0 & 0 & 0 \\
A_{\Gamma_1^{21},i} & A_{\Gamma_1^{21}, \Gamma_1^2} & A_{\Gamma_1^{21}, \Gamma_2^1} & 0 & 0 & 0 & 0 \\
0 & A_{\Gamma_2^{11}, \Gamma_2^1} & A_{\Gamma_2^{11}, \Gamma_2^2} & A_{\Gamma_2^{11}, \Gamma_2^2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
A_{\Gamma_2^{12},i} & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
A_{2i,2i} & A_{2i, \Gamma_2^1} & A_{2i, \Gamma_2^3} & 0 & 0 & 0 \\
A_{\Gamma_2^{12},i} & A_{\Gamma_2^{12}, \Gamma_2^1} & A_{\Gamma_2^{12}, \Gamma_2^3} & 0 & 0 & 0 \\
A_{\Gamma_3^{22},i} & 0 & A_{\Gamma_3^{22}, \Gamma_3^2} & 0 & 0 & 0 \\
0 & A_{\Gamma_1^{21}, \Gamma_2^1} & A_{\Gamma_1^{21}, \Gamma_2^2} & 0 & 0 & 0 \\
0 & 0 & A_{\Gamma_2^{22}, \Gamma_2^3} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
A_{\Gamma_3^{23},i} & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
A_{3i,3i} & A_{3i, \Gamma_3^2} & 0 \\
A_{\Gamma_3^{23},i} & A_{\Gamma_3^{23}, \Gamma_3^2} & A_{\Gamma_3^{23}, \Gamma_3^3} \\
0 & A_{\Gamma_3^{23}, \Gamma_3^2} & A_{\Gamma_3^{23}, \Gamma_3^3} \\
A_{\Gamma_3^{32},i} & 0 & 0 & 0 & 0 \\
0 & A_{\Gamma_3^{32}, \Gamma_3^2} & A_{\Gamma_3^{32}, \Gamma_3^3} \\
0 & 0 & 0 & 0 & 0 \\
A_{\Gamma_3^{33},i} & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
Inflation vs Computational Time

Matrix L3D4x4x16n10.mtx

<table>
<thead>
<tr>
<th>nparts</th>
<th>nrows</th>
<th>nnz</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>270,641</td>
<td>3,941,521</td>
</tr>
</tbody>
</table>

\[ \Omega_{\text{size}} = 10^3 \]

\[ \delta \Omega = 1.0 \]

\[ \log_{10}(\text{residual norm}) \]

\[ \text{number of iterations} \]

<table>
<thead>
<tr>
<th>n-iter</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I(0)</td>
<td>139</td>
<td>139</td>
<td>139</td>
</tr>
<tr>
<td>I(1)</td>
<td>82</td>
<td>82</td>
<td>82</td>
</tr>
<tr>
<td>I(2)</td>
<td>61</td>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>I(3)</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

\[ \begin{align*}
\inf[s] + LU[s] + \sum[s] \end{align*} \]

\[ \begin{align*}
0.00 + 0.22 + 12.69 & = 12.91 \\
0.34 + 0.31 + 9.27 & = 9.92 \\
0.79 + 0.59 + 7.85 & = 9.23 \\
1.53 + 1.69 + 6.56 & = 9.75
\end{align*} \]
**Modified Schwarz Method**

A new interface condition implies additional augmented matrixes defined on the interface between sub-domains. The augmented matrices are defined as follows:

\[
\begin{bmatrix}
A_{1_{i1}}, & A_{1_{i1}}^{\Gamma_2} & 0 \\
A_{1_{i1}}, & A_{1_{i1}}^{\Gamma_2} & A_{1_{i1}}^{\Gamma_1} + S_1^2 \\
0, & A_{1_{i1}}^{\Gamma_1} & A_{1_{i1}}^{\Gamma_1} + S_2^1
\end{bmatrix}
\]

\[
\begin{bmatrix}
A_{2_{i2}}, & A_{2_{i2}}^{\Gamma_2} & A_{2_{i2}}^{\Gamma_1} & 0 \\
A_{2_{i2}}, & A_{2_{i2}}^{\Gamma_2} & A_{2_{i2}}^{\Gamma_1} + S_2^1 \\
0, & A_{2_{i2}}^{\Gamma_1} & A_{2_{i2}}^{\Gamma_1} + S_2^1
\end{bmatrix}
\]

\[
\begin{bmatrix}
A_{3_{i3}}, & A_{3_{i3}}^{\Gamma_2} & A_{3_{i3}}^{\Gamma_3} & A_{3_{i3}}^{\Gamma_3} + S_3 \\
A_{3_{i3}}, & A_{3_{i3}}^{\Gamma_2} & A_{3_{i3}}^{\Gamma_3} + S_3 \\
0, & A_{3_{i3}}^{\Gamma_3} & A_{3_{i3}}^{\Gamma_3} + S_3
\end{bmatrix}
\]
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 $\tilde{A}$ i.e.

$$S_1^{opt} = -A_{2i2i}^{-1} A_{2i2i}^- A_{2i2i}$$

$$S_2^{opt} = -A_{1i1i}^{-1} A_{1i1i}^{-1} A_{1i1i}$$

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 $\Omega_1$ which we simply denote by 1 and its inflated counterpart by $\tilde{1}$.

\[
\begin{bmatrix}
    A_{1i} & 0 \\
    A_{1c} & 0
\end{bmatrix}
\begin{bmatrix}
    U_{1i} \\
    U_{1c}
\end{bmatrix}
= 
\begin{bmatrix}
    F_{1i} \\
    F_{1c}
\end{bmatrix}
\]
What about general case ($2 < N$ - subdomains)?
Diagonal Approximation

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

\[
S^\text{opt}_{\tilde{\Gamma}_1 \tilde{\Gamma}_1} := -A_{\tilde{\Gamma}_1 \tilde{1}_c} A^{-1}_{\tilde{1}_c \tilde{1}_c} A_{\tilde{1}_c \tilde{\Gamma}_1}
\]

More precisely we seek an approximation to \(S^\text{opt}_{\tilde{\Gamma}_1 \tilde{\Gamma}_1}\) in form:

\[
S_{\tilde{\Gamma}_1 \tilde{\Gamma}_1} \approx S^\text{opt}_{\tilde{\Gamma}_1 \tilde{\Gamma}_1} = -A_{\tilde{\Gamma}_1 \tilde{1}_c} \beta_{\tilde{1}_c \tilde{1}_c} A_{\tilde{1}_c \tilde{\Gamma}_1}
\]

such that, the optimality condition is verified on the vector \(V_{\tilde{\Gamma}_1}\):

\[
-A_{\tilde{\Gamma}_1 \tilde{1}_c} \beta_{\tilde{1}_c \tilde{1}_c} A_{\tilde{1}_c \tilde{\Gamma}_1} V_{\tilde{\Gamma}_1} = S^\text{opt}_{\tilde{\Gamma}_1 \tilde{\Gamma}_1} V_{\tilde{\Gamma}_1}
\]

where \(V\) is a harmonic vector i.e.,

\[
A_{\tilde{1}_c \tilde{1}_c} V_{\tilde{1}_c} + A_{\tilde{1}_c \tilde{\Gamma}_1} V_{\tilde{\Gamma}_1} = 0
\]
Sparse matrix $\beta_{\tilde{1}_c \tilde{1}_c}$

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

$$\beta_{\tilde{1}_c \tilde{1}_c} := \text{diag} \left( -V_{\tilde{1}_c} \cdot / A_{\tilde{1}_c \tilde{1}_c} V_{\tilde{1}_c} \right)$$

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

"./" - element wise division

$$\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \cdot / \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} v_1 / w_1 \\ \vdots \\ v_n / w_n \end{bmatrix}$$
Harmonic Vector - Facts

- Due to the block preconditioner $M^{-1}$ (the **Schwarz method**) the vectors in the **Krylov space** $K_m(\tilde{M}^{-1}\tilde{A}, r_0)$ 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 $\tilde{M}^{-1}\tilde{A}$ associated with the **smallest eigenvalue** $\lambda$ (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 $\mathcal{K}_m(\widetilde{M}^{-1}\tilde{A}, r_0)$ 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 $\widetilde{M}^{-1}\tilde{A}$ associated with the smallest eigenvalue $\lambda$ (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 $K_m(\tilde{M}^{-1}\tilde{A}, r_0)$ 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 $\tilde{M}^{-1}\tilde{A}$ associated with the smallest eigenvalue $\lambda$ (we use Krylov subspaces created via GMRES).
The computational kernel of GMRES is the Arnoldi process which computes the orthonormal basis $W_m$ for the Krylov subspace $K_m(\tilde{M}^{-1}\tilde{A}, r_0)$.

Since the Arnoldi basis is orthonormal, $W_m = (w_1 w_2 \ldots w_m)$ is an orthogonal matrix ($W_m \in \mathbb{R}^{n \times m}$).

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:

$$H_m = W_m^H \tilde{M}^{-1}\tilde{A}W_m$$
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 $K_m(\tilde{M}^{-1} \tilde{A}, r_0)$.

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:

$$H_m = W_m^H \tilde{M}^{-1} \tilde{A} W_m$$
The computational kernel of GMRES is the Arnoldi process which computes the orthonormal basis $W_m$ for the Krylov subspace $\mathcal{K}_m(\tilde{M}^{-1}\tilde{A}, r_0)$.

Since the Arnoldi basis is orthonormal, $W_m = (w_1 \ w_2 \ldots \ w_m)$ is an orthogonal matrix ($W_m \in \mathbb{R}^{n \times m}$).

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:

$$H_m = W_m^H \tilde{M}^{-1} \tilde{A} W_m$$
The eigenvalues of $H_m$ are called Ritz values and they approximate the eigenvalues of $\tilde{M}^{-1}\tilde{A}$.

If $z_\star$ is an chosen eigenvector of $H_m$, then $V_\star = W_m z_\star$ is almost an eigenvector of $\tilde{M}^{-1}\tilde{A}$, for the same eigenvalue $\lambda$, i.e.,

$$\tilde{M}^{-1}\tilde{A} V_\star \approx W_m H_m W_m^H W_m z_\star = W_m H_m W_m z_\star = W_m \lambda z_\star = \lambda V_\star$$

In practice a specific $\text{Lapack}$ procedure can be used to compute the eigenelements of $H_m$. 

Approximate eigenvector from GMRES solver
The eigenvalues of $H_m$ are called Ritz values and they approximate the eigenvalues of $\tilde{M}^{-1}\tilde{A}$.

If $z_\star$ is a chosen eigenvector of $H_m$, then $V_\star = W_m z_\star$ is almost an eigenvector of $\tilde{M}^{-1}\tilde{A}$, for the same eigenvalue $\lambda$ i.e.,

$$\tilde{M}^{-1}\tilde{A}V_\star \approx W_m H_m W_m^H W_m z_\star = W_m H_m z_\star = W_m \lambda z_\star = \lambda V_\star$$

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:

\[
\begin{cases}
\left( \eta(x, y) - \text{div}(\kappa(x, y) \vec{\nabla}) \right) u(x, y) = f(x, y) \quad \text{in} \quad \Omega \\
u(x, y) = 0 \quad \text{on} \quad \partial\Omega_D \\
\frac{\partial u}{\partial n} = 0 \quad \text{on} \quad \partial\Omega_N
\end{cases}
\]
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) $M_x = M_y = 4$

<table>
<thead>
<tr>
<th>Method</th>
<th>n-iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) ASM</td>
<td>17</td>
</tr>
<tr>
<td>(a) MSM+EDOIC(3)</td>
<td>13</td>
</tr>
<tr>
<td>(b) ASM</td>
<td>26</td>
</tr>
<tr>
<td>(b) MSM+EDOIC(3)</td>
<td>23</td>
</tr>
<tr>
<td>(c) ASM</td>
<td>34</td>
</tr>
<tr>
<td>(c) MSM+EDOIC(3)</td>
<td>36</td>
</tr>
</tbody>
</table>
**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

\[ P := I - A \left( Z E^{-1} Z^T \right) \quad E := Z^T A Z \]

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

\[
P := I - A \left( Z E^{-1} Z^T \right) \quad E := Z^T A Z
\]

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

\[ P := I - A \left( Z E^{-1} Z^T \right) \quad E := Z^T A Z \]

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

\[ P := I - A \left( Z E^{-1} Z^T \right) \quad E := Z^T A Z \]

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

\[
P := I - A \left( Z E^{-1} Z^T \right) \quad E := Z^T A Z
\]

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]:

\[
Z = \begin{bmatrix}
1_{\Omega_1} & 0 & \cdots & 0 \\
\vdots & 1_{\Omega_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1_{\Omega_J}
\end{bmatrix}
\]

\[
(z_k)_l = \begin{cases} 
1 & l \in \Omega_j \\
0 & l \notin \Omega_j 
\end{cases}
\]
For a Poisson like problem, Nicolaides proposed [Nicolaides, 1987]:

\[
Z = \begin{bmatrix}
1_{\Omega_1} & 0 & \cdots & 0 \\
\vdots & 1_{\Omega_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1_{\Omega_J}
\end{bmatrix}
\]

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

\[
\log_{10}(\text{residual norm})
\]

Number of iterations

Poisson 2D (32 parts) Additive Schwarz Method
Poisson 2D (32 parts) 2-Level Preconditioner (Z - Nicolaides)
Poisson 2D (32 parts) 2-Level Preconditioner (Z - New way)
Our choice of $Z$

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

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

We can apply a part wise splitting to $Z^*$ 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: $V_\star = W_m z_\star$)

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

$Z := \begin{bmatrix}
[V_1]_{\tilde{D}_\Omega_1} & [V_2]_{\tilde{D}_\Omega_1} & \cdots & [V_{n_v}]_{\tilde{D}_\Omega_1} & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & [V_1]_{\tilde{D}_\Omega_2} & [V_2]_{\tilde{D}_\Omega_2} & \cdots & [V_{n_v}]_{\tilde{D}_\Omega_2} & 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & [V_1]_{\tilde{D}_\Omega_N} & [V_2]_{\tilde{D}_\Omega_N} & \cdots & [V_{n_v}]_{\tilde{D}_\Omega_N}
\end{bmatrix}$

We can apply a part wise splitting 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: \( \mathcal{V}_* = W_m z_* \))

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

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

We can apply a part wise splitting to \( Z^* \) 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** \( \mathcal{P}_{L&R} \)

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

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

\[
M^{-1} := \begin{bmatrix}
A_{D_{\Omega_1}}^{-1} & 0 & 0 \\
0 & A_{D_{\Omega_2}}^{-1} & 0 \\
0 & 0 & A_{D_{\Omega_3}}^{-1}
\end{bmatrix}
\]

\[
E_L := Z^T M^{-1} A Z \\
E_R := Z^T A M^{-1} Z
\]

**Left preconditioner**

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

**Right preconditioner**

\[
A M^{-1} \mathcal{P}_R \bar{\mathbf{u}} = \mathbf{b} \\
\mathbf{u} = M^{-1} \mathcal{P}_R \bar{\mathbf{u}}
\]
Academic Problem - 3D Laplace

- 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?

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>niter</strong></td>
<td>number of iterations</td>
</tr>
<tr>
<td><strong>κ_≈</strong></td>
<td>roughly estimated condition number given as $κ_≈ ≈ \lambda_{max}/\lambda_{min}$ where $\lambda_{{\text{min, max}}}$ are the approximated, extreme eigenvalues of $(\tilde{\mathbf{M}}^{-1}\tilde{\mathbf{A}})$</td>
</tr>
<tr>
<td><strong>nV</strong></td>
<td>number of approximated eigenvectors used in construction of coarse space</td>
</tr>
<tr>
<td>$</td>
<td></td>
</tr>
<tr>
<td><strong>CS[s]</strong></td>
<td>time of “construction” coarse space operator</td>
</tr>
<tr>
<td><strong>Inf[s]</strong></td>
<td>time of inflation process for each level</td>
</tr>
<tr>
<td><strong>LU[s]</strong></td>
<td>time of LU factorisation of endomorphic Partial Operators in DDMOperator</td>
</tr>
<tr>
<td><strong>sol[s]</strong></td>
<td>time of iterative process (in case of variational two-level preconditioner sol consist also LU factorisation time of coarse operator)</td>
</tr>
</tbody>
</table>
Academic Problem - 3D Laplace (SCALABILITY)
Academic Problem - 3D Laplace (SCALABILITY)

![Graph showing the comparison of different configurations of Laplace3D and their scalability]

- Log10(residual norm) vs. number of iterations for various configurations of Laplace3D.
- Configurations include: Laplace3D 4x4x8 (15) +I (1), Laplace3D 4x4x16 (15) +I (1), Laplace3D 8x8x8 (15) +I (1), Laplace3D 8x8x16 (15) +I (1), (DEFLATE)(8V) Laplace3D 4x4x8 (15) +I (1), (DEFLATE)(12V) Laplace3D 4x4x16 (15) +I (1), (DEFLATE)(13V) Laplace3D 8x8x8 (15) +I (1), (DEFLATE)(10FV) Laplace3D 8x8x16 (15) +I (1).
Academic Problem - 3D Laplace (TIME COST)
Academic Problem - 3D Laplace (TIME COST)
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

<table>
<thead>
<tr>
<th>nparts</th>
<th>nrows</th>
<th>nnz</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>1,094,421</td>
<td>7,515,591</td>
</tr>
</tbody>
</table>
IFP Matrices Collection (spe10)

\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
\text{expvar} & \text{niter} & \kappa_\infty & \mathbf{nV} & \|\mathbf{r}_{sol}\| & \text{CS[s]} & \text{inf[s]} & \text{LU[s]} & \text{sol[s]} \\
\hline
+I(0) & 700 & 83147.80 & 6.40e - 05 & & 1.36 & 1.36 & 93.34 \\
+I(1) & 386 & 15169.30 & 3.32e - 08 & & 1.09 & 2.24 & 42.14 \\
D..+I(1) & 80 & 185.10 & 10F & 1.83e - 09 & 1.30 & & 22.43 \\
+I(2) & 264 & 7825.55 & 3.84e - 08 & & 1.09+1.54 & 3.38 & 22.76 \\
D..+I(2) & 62 & 97.08 & 10F & 1.13e - 08 & 1.48 & & 14.39 \\
+I(3) & 211 & 5055.91 & 5.02e - 09 & & 1.09+1.54+2.20 & 5.06 & 17.32 \\
D..+I(3) & 47 & 50.12 & 10F & 2.22e - 09 & 1.61 & & 14.22 \\
\hline
\end{array}
\]
IFP Matrices Collection (spe10)

-6
-5
-4
-3
-2
-1
0
0  100  200  300  400  500  600  700
log10(residual norm)

number of iterations

spe10_p_only (256p) +I (0)
spe10_p_only (256p) +I (1)
(DEFLATED)(10VF) spe10_p_only (256p) +I (1)

<table>
<thead>
<tr>
<th>n-iter</th>
<th>I(0)</th>
<th>I(1)</th>
<th>I(1)+D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>700</td>
<td>386</td>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>CS[s] + inf[s] + LU[s] + sol[s]</th>
<th>∑ [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>I(0)</td>
<td>0.00+0.00+1.36+93.34 94.70</td>
<td></td>
</tr>
<tr>
<td>I(1)</td>
<td>0.00+1.90+2.24+42.14 46.28</td>
<td></td>
</tr>
<tr>
<td>I(1)+D</td>
<td>1.30+1.90+2.24+22.43 27.87</td>
<td></td>
</tr>
</tbody>
</table>
## Motivation

- Building Coarse Space
- Partitioning and data distribution (mat1)
- LU Factorization of sub-operators (mat1)
- SOLVE (mat1) GMRES + Modified Schwarz Method

## State of the Arts

- Extracting Approximated Eigen Vectors
- Partitioning and data distribution (mat2)
- LU Factorization of sub-operators (mat2)
- SOLVE (mat2) GMRES + Two-Level preconditioner

## New Methods and Techniques

### Matrix BO_60x60x32_matX.mtx

<table>
<thead>
<tr>
<th>mat_name</th>
<th>nparts</th>
<th>nrows</th>
<th>nnz</th>
</tr>
</thead>
<tbody>
<tr>
<td>BO_60x60x32_mat1.mtx</td>
<td>80</td>
<td>115,200</td>
<td>791,520</td>
</tr>
<tr>
<td>BO_60x60x32_mat2.mtx</td>
<td>80</td>
<td>115,200</td>
<td>791,572</td>
</tr>
<tr>
<td>BO_60x60x32_mat3.mtx</td>
<td>80</td>
<td>115,200</td>
<td>791,598</td>
</tr>
<tr>
<td>BO_60x60x32_mat4.mtx</td>
<td>80</td>
<td>115,200</td>
<td>791,500</td>
</tr>
<tr>
<td>BO_60x60x32_mat5.mtx</td>
<td>80</td>
<td>115,200</td>
<td>791,512</td>
</tr>
</tbody>
</table>

### Matrix BO_120x120x64_matX.mtx

<table>
<thead>
<tr>
<th>mat_name</th>
<th>nparts</th>
<th>nrows</th>
<th>nnz</th>
</tr>
</thead>
<tbody>
<tr>
<td>BO_120x120x64_mat1.mtx</td>
<td>160</td>
<td>921,600</td>
<td>6,391,680</td>
</tr>
<tr>
<td>BO_120x120x64_mat2.mtx</td>
<td>160</td>
<td>921,600</td>
<td>6,391,680</td>
</tr>
<tr>
<td>BO_120x120x64_mat3.mtx</td>
<td>160</td>
<td>921,600</td>
<td>6,391,680</td>
</tr>
<tr>
<td>BO_120x120x64_mat4.mtx</td>
<td>160</td>
<td>921,600</td>
<td>6,390,986</td>
</tr>
<tr>
<td>BO_120x120x64_mat5.mtx</td>
<td>160</td>
<td>921,600</td>
<td>6,387,222</td>
</tr>
</tbody>
</table>
Black Oil Simulation

![Graphs showing log10(residual norm) vs. number of iterations for different Black Oil Simulation configurations.]

**sol[s]**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM+I(1)</td>
<td>0.98</td>
</tr>
<tr>
<td>ASM+I(1)+D</td>
<td>0.63</td>
</tr>
</tbody>
</table>

**∑[s]**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM+I(1)</td>
<td>$5 \times 0.98$</td>
</tr>
<tr>
<td>ASM+I(1)+D</td>
<td>$1 \times 0.98 + 0.12 + 4 \times 0.63$</td>
</tr>
<tr>
<td>ASM+I(1)</td>
<td>$5 \times 7.58$</td>
</tr>
<tr>
<td>ASM+I(1)+D</td>
<td>$1 \times 7.58 + 1.27 + 4 \times 3.67$</td>
</tr>
</tbody>
</table>
**Black Oil Simulation - Reduced Formula**

\[
\tilde{P}_R := \left[ \mathbb{I} - (ZE_R^{-1}Z^T)AM^{-1} + (ZE_R^{-1}Z^T) \right]
\]
Conclusion and Prospects
We have considered the extended and the original linear system arising from the domain decomposition method with overlapping.

We applied the two-level preconditioner using Schwarz algorithm and the coarse grid correction.

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

All presented methods are as algebraic as possible which paves the way to extension to systems of equations e.g. multiphase flows.

Proposed two-level preconditioner is scalable and can be very robust in respect to number of iteration.

Both methods are adaptive and can be used during first solve that is even the first solve is not completed.

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!
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.

On the Schwarz alternating method. III: a variant for nonoverlapping subdomains.
Algebraic approximation of dirichlet-to-neumann maps for equations of linear elasticity.

Optimal interface conditions for domain decomposition methods.

Deflation of conjugate gradients with applications to boundary value problems.

Schwarz, H. A. (1870).
Über einen Grenzübergang durch alternierendes Verfahren.


Additional Numerical Experiments

Additional Numerical Experiments
IFP Matrices Collection (spe10 - system of equations)

Matrix spe10.sys.mtx

<table>
<thead>
<tr>
<th>nparts</th>
<th>nrows</th>
<th>nnz</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>2,188,842</td>
<td>21,554,641</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>expvar</th>
<th>niter</th>
<th>$|r_{sol}|$</th>
<th>$|s|$</th>
<th>$\inf[s]$</th>
<th>LU$[s]$</th>
<th>sol$[s]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+I(0)</td>
<td>663</td>
<td>$1.43e-06$</td>
<td></td>
<td>$\approx 4$</td>
<td>343.13</td>
<td></td>
</tr>
<tr>
<td>+I(1)</td>
<td>361</td>
<td>$2.12e-09$</td>
<td>$4.36$</td>
<td>$\approx 6$</td>
<td>70.76</td>
<td></td>
</tr>
<tr>
<td>+I(2)</td>
<td>252</td>
<td>$3.81e-08$</td>
<td>$4.37 + 7.23$</td>
<td>$\approx 13$</td>
<td>70.88</td>
<td></td>
</tr>
<tr>
<td>+I(3)</td>
<td>183</td>
<td>$2.03e-08$</td>
<td>$4.52 + 7.42 + 11.00$</td>
<td>$\approx 24$</td>
<td>78.14</td>
<td></td>
</tr>
</tbody>
</table>
Adaptive Solver

\[ u_0 = 0 \]
\[ u_{STOP} \]
\[ Z^* \in \mathcal{K}_{m=STOP}^{ASM} \]
\[ Z^* \rightarrow Z \]
\[ u_0 = u_{STOP} \in \mathcal{K}_m^{ASM} \]

\[ \mathcal{K}_m^{ASM}(AM^{-1}, r_0) := u_0 + \text{SPAN}\{r_0, (AM^{-1})^{m-1}r_0\} \]

\[ \mathcal{K}_m^{2lvl}(AM^{-1} \mathcal{P}_R, r_0) := u_0 + \text{SPAN}\{r_0, (AM^{-1} \mathcal{P}_R)^{m-1}r_0\} \]
Adaptive Solver - Stop after XX

![Graph showing log10(residual norm) vs. number of iterations for Laplace3D 4x4x16 (15) +I (1)]
Adaptive Solver - Stop after 10

Laplace3D 4x4x16 (15) +I (1)
(STOP 10)(DEFLATE)(2V)
Adaptive Solver - Stop after 20
Adaptive Solver - Stop after (40 and 50)

Laplace3D 4x4x16 (15) +I (1)
(STOP 10)(DEFLATE)(2V)
(STOP 20)(DEFLATE)(3V)
(STOP 40)(DEFLATE)(6V)
(STOP 50)(DEFLATE)(8V)
Adaptive Solver vs Time of Computation

<table>
<thead>
<tr>
<th>expvar</th>
<th>niter</th>
<th>(\kappa_\infty)</th>
<th>nV</th>
<th>(|r_{sol}|)</th>
<th>CS[s]</th>
<th>inf[s]</th>
<th>LU[s]</th>
<th>sol[s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+I(1)</td>
<td>101</td>
<td>4818.07</td>
<td>1.68e-09</td>
<td>1.29</td>
<td>3.56</td>
<td>10.30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>S10</td>
<td>10 + 85</td>
<td>2427.91</td>
<td>2.34e-09</td>
<td>0.25</td>
<td>1.38</td>
<td>4.21</td>
<td>3.43 + 12.15</td>
</tr>
<tr>
<td></td>
<td>S20</td>
<td>20 + 56</td>
<td>648.32</td>
<td>1.22e-09</td>
<td>0.37</td>
<td>1.31</td>
<td>4.17</td>
<td>3.52 + 11.77</td>
</tr>
<tr>
<td></td>
<td>S40</td>
<td>40 + 23</td>
<td>7.68</td>
<td>9.53e-10</td>
<td>1.15</td>
<td>1.32</td>
<td>5.47</td>
<td>3.73 + 7.70</td>
</tr>
<tr>
<td></td>
<td>S50</td>
<td>50 + 18</td>
<td>4.12</td>
<td>2.69e-10</td>
<td>1.01</td>
<td>1.29</td>
<td>3.76</td>
<td>3.91 + 3.82</td>
</tr>
</tbody>
</table>
Adaptive Solver vs Time of Computation

![Graph showing the comparison between Adaptive Solver and Time of Computation.

The graph plots the log10(residual norm) on the y-axis against the number of iterations on the x-axis. The data points are colored to represent different cases, with key labels indicating the specific configurations.

Table:

<table>
<thead>
<tr>
<th>n-iter</th>
<th>CS[s] + inf[s] + LU[s] + sol[s]</th>
<th>∑[s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>I(1)</td>
<td>0.00 + 1.29 + 3.56 + 10.30</td>
<td>15.15</td>
</tr>
<tr>
<td>AI(1)</td>
<td>1.01 + 1.29 + 3.76 + 7.73</td>
<td>13.79</td>
</tr>
</tbody>
</table>
Improved Diagonal Approximation (EDOIC)
Improved Diagonal Approximation (EDOIC)

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

$$\beta_{\tilde{1}_c\tilde{1}_c} A_{\tilde{1}_c\tilde{1}_c} \tilde{V}_{\tilde{1}_c} = \tilde{V}_{\tilde{1}_c},$$

or equivalently

$$-\beta_{\tilde{1}_c\tilde{1}_c} A_{\tilde{1}_c\tilde{1}_c} \tilde{V}_{\tilde{1}_c} = \tilde{V}_{\tilde{1}_c}.$$

The optimal interface conditions operator $S_{\tilde{\Gamma}_1\tilde{\Gamma}_1}^{opt}$ is approximated by

$$S_{\tilde{\Gamma}_1\tilde{\Gamma}_1}^{opt} \approx S_{\tilde{\Gamma}_1\tilde{\Gamma}_1}^{edoic} := -A_{\tilde{1}_c\tilde{1}_c} \left(2\beta_{\tilde{1}_c\tilde{1}_c} - \beta_{\tilde{1}_c\tilde{1}_c} A_{\tilde{1}_c\tilde{1}_c} \beta_{\tilde{1}_c\tilde{1}_c}\right) A_{\tilde{1}_c\tilde{1}_c}.$$

The idea of this improvement originates from the following calculations:

$\|(BA - I)\| \leq \epsilon < 1$ leads to $\|(BA - I)^2\| \leq \epsilon^2 < \epsilon$. Then, remarking that $(BA - I)^2 = BAB + I = I - (2B - BAB)A$, one concludes that

$C = 2B - BAB$ is better approximation of $A^{-1}$ than $B$ since $\|C - I\| \leq \epsilon^2 < \epsilon$. 

Sparse Patch Method
Sparse Patch Method

\[ A^j_P = \begin{pmatrix} \overline{A}_{jj} & \overline{A}_{j\Gamma} \\ \overline{A}_{\Gamma j} & \overline{A}_{\Gamma\Gamma} \end{pmatrix} \]

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