An introduction to Schwarz methods

Victorita Dolean

Université de Nice and University of Geneva

École thematique CNRS
Décomposition de domaine
November 15-16, 2011

Outline

Motivation: pro and cons of direct solvers

Complexity of the Gauss factorization

Gauss	$d = 1$	$d = 2$	$d = 3$
dense matrix	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
using band structure	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^{7/3})$
using sparsity	$\mathcal{O}(n)$	$\mathcal{O}(n^{3/2})$	$\mathcal{O}(n^2)$

Different sparse direct solvers

- PARDISO (`http://www.pardiso-project.org`)
- SUPERLU (`http://crd.lbl.gov/~xiaoye/SuperLU`)
- SPOOLES (`www.netlib.org/linalg/spooles/spooles.2.2.html`)
- MUMPS (`http://graal.ens-lyon.fr/MUMPS/`)
- UMFPACK (`http://www.cise.ufl.edu/research/sparse/umfpack`)

Why iterative solvers?

Limitations of direct solvers

In practice all direct solvers work well until a certain barrier:

- **two-dimensional problems** (100K unknowns)
- **three-dimensional problems** (10K unknowns).

Beyond, the factorization cannot be stored in memory any more.

To summarize:

- below a certain size, direct solvers are chosen.
- beyond the critical size, iterative solvers are needed.

Why domain decomposition?

Natural iterative/direct trade-off

- Parallel processing is the only way to have faster codes, new generation processors are parallel: dual, quadri core.
- Large scale computations need for an ”artificial” decomposition
- Memory requirements, direct solvers are too costly
- Iterative solvers are not robust enough.

New iterative/direct solvers are welcome : these are domain decomposition methods

In some situations, the decomposition is natural

- Moving domains (rotor and stator in an electric motor)
- Strongly heterogeneous media
- Different physics in different subdomains

Linear Algebra from the End User point of view

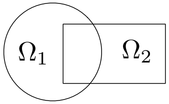
Direct	DDM	Iterative
Cons: Memory Difficult to $\|\|$ Pros: Robustness	Pro: Flexible Naurally $\|\|$	Pros: Memory Easy to $\|\|$ Cons: Robustness
solve(MAT,RHS,SOL)	Few black box routines Few implementations of efficient DDM	solve(MAT,RHS,SOL)

Multigrid methods: very efficient but may lack robustness, not always applicable (Helmholtz type problems, complex systems) and difficult to parallelize.

Outline

The First Domain Decomposition Method

The original Schwarz Method (H.A. Schwarz, 1870)



$$-\Delta(u) = f \quad \text{in } \Omega$$
$$u = 0 \quad \text{on } \partial\Omega.$$

Schwarz Method : $(u_1^n, u_2^n) \to (u_1^{n+1}, u_2^{n+1})$ with

$$\begin{array}{ll} -\Delta(u_1^{n+1}) = f \quad \text{in } \Omega_1 & -\Delta(u_2^{n+1}) = f \quad \text{in } \Omega_2 \\ u_1^{n+1} = 0 \text{ on } \partial\Omega_1 \cap \partial\Omega & u_2^{n+1} = 0 \text{ on } \partial\Omega_2 \cap \partial\Omega \\ u_1^{n+1} = u_2^n \quad \text{on } \partial\Omega_1 \cap \overline{\Omega_2}. & u_2^{n+1} = u_1^{n+1} \quad \text{on } \partial\Omega_2 \cap \overline{\Omega_1}. \end{array}$$

Parallel algorithm, converges but very slowly, overlapping subdomains only.
The parallel version is called **Jacobi Schwarz method (JSM)**.

Continuous ASM and RAS - I

The algorithm acts on the local functions $(u_i)_{i=1,2}$.
To make things global, we need:

- extension operators, E_i, s.t. for a function $w_i : \Omega_i \mapsto \mathbb{R}$, $E_i(w_i) : \Omega \mapsto \mathbb{R}$ is the extension of w_i by zero outside Ω_i.
- partition of unity functions $\chi_i : \Omega_i \mapsto \mathbb{R}$, $\chi_i \geq 0$ and $\chi_i(x) = 0$ for $x \in \partial\Omega_i$ and s.t.

$$w = \sum_{i=1}^{2} E_i(\chi_i \, w_{|\Omega_i}) \, .$$

Let u^n be an approximation to the solution to the global Poisson problem and u^{n+1} is computed by solving first local subproblems and then gluing them together.

Continuous ASM and RAS - II

Local problems to solve

$$\begin{array}{rll} -\Delta(u_i^{n+1}) = f & \text{in} & \Omega_i \\ u_i^{n+1} = 0 & \text{on} & \partial\Omega_i \cap \partial\Omega \\ u_i^{n+1} = u^n & \text{on} & \partial\Omega_i \cap \overline{\Omega}_{3-i}. \end{array}$$

Two ways to ”glue” solutions

- Using the partition of unity functions **Restricted Additive Schwarz (RAS)**

$$u^{n+1} := \sum_{i=1}^{2} E_i(\chi_i\, u_i^{n+1})\,.$$

- Not based on the partition of unity **Additive Schwarz (ASM)**

$$u^{n+1} := \sum_{i=1}^{2} E_i(u_i^{n+1})\,.$$

Block Jacobi methods - I

Let us consider a linear system:

$$AU = F$$

with a matrix A of size $m \times m$, a right handside $F \in \mathbb{R}^m$ and a solution $U \in \mathbb{R}^m$ where m is an integer. Let D be the diagonal of A, the Jacobi algorithm reads:

$$DU^{n+1} = DU^n + (b - AU^n)\,,$$

or equivalently,

$$U^{n+1} = U^n + D^{-1}(b - AU^n) = U^n + D^{-1}r^n\,,$$

where r^n is the residual of the equation.

Block Jacobi methods - II

We now define a block Jacobi algorithm. The set of indices $\{1, \ldots, m\}$ is partitioned into two sets

$$\mathcal{N}_1 := \{1, \ldots, m_s\} \text{ and } \mathcal{N}_2 := \{m_s + 1, \ldots, m\}\,.$$

Let $U_1 := U_{|\mathcal{N}_1}$, $U_2 := U_{|\mathcal{N}_2}$ and similarly $F_1 := F_{|\mathcal{N}_1}$, $F_2 := F_{|\mathcal{N}_2}$. The linear system has the following block form:

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

where $A_{ij} := A_{|\mathcal{N}_i \times \mathcal{N}_j}$, $1 \leq i, j \leq 2$.

Block Jacobi methods - III

The block-Jacobi algorithm reads:

$$\begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} U_1^{n+1} \\ U_2^{n+1} \end{pmatrix} = \begin{pmatrix} F_1 - A_{12}\, U_2^n \\ F_2 - A_{21}\, U_1^n \end{pmatrix}. \tag{1}$$

Let $U^n = (U_1^n, U_2^n)^T$, algorithm (1) becomes

$$\begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix} U^{n+1} = F - \begin{pmatrix} 0 & A_{12} \\ A_{21} & 0 \end{pmatrix} U^n. \tag{2}$$

On the other hand, it can be rewritten equivalently

$$\begin{pmatrix} U_1^{n+1} \\ U_2^{n+1} \end{pmatrix} = \begin{pmatrix} U_1^n \\ U_2^n \end{pmatrix} + \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix}^{-1} \begin{pmatrix} r_1^n \\ r_2^n \end{pmatrix} \tag{3}$$

where

$$r^n := F - AU^n\,, r_i^n := r^n_{|\mathcal{N}_i},\ i = 1, 2\,.$$

Block-Jacobi compact form

In order to have a more compact form, let us introduce

- R_1 the restriction operator from $\mathcal{N}$ into $\mathcal{N}_1$
- R_2 the restriction operator from $\mathcal{N}$ into $\mathcal{N}_2$.

The transpose operator R_1^T is an extension operator from $\mathcal{N}_1$ into $\mathcal{N}$ and the same holds for R_2^T.
Notice that $A_{ii} = R_i A R_i^T$.

Block-Jacobi in compact form

$$U^{n+1} = U^n + (R_1^T(R_1 A R_1^T)^{-1} R_1 + R_2^T(R_2 A R_2^T)^{-1} R_2) r^n \,. \quad (4)$$

where

$$r^n := F - AU^n \,, r_i^n := r^n_{|\mathcal{N}_i}, \; i = 1, 2 \,.$$

Schwarz algorithms as block Jacobi methods - I

Let $\Omega := (0,1)$ and consider the following BVP

$$\begin{array}{l} -\Delta u = f \text{ in } \Omega \\ u(0) = u(1) = 0\,. \end{array}$$

discretized by a three point finite difference scheme on the grid $x_j := j\,h$, $1 \leq j \leq m$ where $h := 1/(m+1)$.
Let $u_j \simeq u(x_j)$, $f_j := f(x_j)$, $1 \leq j \leq m$ and the discrete problem

$$AU = F,\; U = (u_j)_{1\leq j\leq m},\; F = (f_j)_{1\leq j\leq m}.$$

where $A_{j\,j} := 2/h^2$ and $A_{j\,j+1} = A_{j+1\,j} := -1/h^2$.
Let domains $\Omega_1 := (0, (m_s + 1)\,h)$ and $\Omega_2 := (m_s\,h, 1)$ define an overlapping decomposition with a minimal overlap of width h.

Schwarz algorithms as block Jacobi methods - II

The discretization of the **JSM** for domain Ω_1 reads

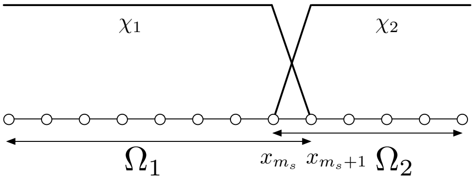
$$\begin{cases} -\dfrac{u_{1,j-1}^{n+1} - 2u_{1,j}^{n+1} + u_{1,j+1}^{n+1}}{h^2} = f_j, \ 1 \le j \le m_s \\ u_{1,0}^{n+1} = 0 \\ u_{1,m_s+1}^{n+1} = u_{2,m_s+1}^{n} \end{cases} .$$

Solving for $U_1^{n+1} = (u_{1,j}^{n+1})_{1 \le j \le m_s}$ corresponds to solving a Dirichlet boundary value problem in subdomain Ω_1 with Dirichlet data taken from the other subdomain at the previous step. Then, U_1^{n+1} and U_1^{n+1} satisfy

$$\begin{aligned} A_{11} U_1^{n+1} + A_{12} U_2^n = F_1 \,, \\ A_{22} U_2^{n+1} + A_{21} U_1^n = F_2 \,. \end{aligned}$$

Schwarz as block Jacobi methods - III

The discrete counterpart of the extension operator E_1 (resp. E_2) is defined by $E_1(U_1) = (U_1, 0)^T$ (resp. $E_2(U_2) = (0, U_2)^T$).



then $E_1(U_1) + E_2(U_2) = E_1(\chi_1 U_1) + E_2(\chi_2 U_2) = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$.

When the overlap is minimal, **the discrete counterparts of the three Schwarz methods are equivalent to the same block Jacobi algorithm.**

Continuous level

- Ω and an overlapping decomposition $\Omega = \cup_{i=1}^N \Omega_i$.
- A function $u : \Omega \to \mathbb{R}$.
- Restriction of $u : \Omega \to \mathbb{R}$ to Ω_i, $1 \leq i \leq N$.
- The extension E_i of a function $\Omega_i \to \mathbb{R}$ to a function $\Omega \to \mathbb{R}$.
- Partition of unity functions χ_i, $1 \leq i \leq N$.

Discrete level

- A set of d.o.f. $\mathcal{N}$ and a decomposition $\mathcal{N} = \cup_{i=1}^N \mathcal{N}_i$.
- A vector $U \in \mathbb{R}^{\#\mathcal{N}}$.
- The restriction $R_i\, U$ where $U \in \mathbb{R}^{\#\mathcal{N}}$ and R_i is a rectangular $\#\mathcal{N}_i \times \#\mathcal{N}$ boolean matrix.
- Extension R_i^T.
- Diagonal matrices with positive entries, of size $\#\mathcal{N}_i \times \#\mathcal{N}_i$ s. t. $Id = \sum_{i=1}^N R_i^T\, D_i\, R_i$.

Restrictions operators

Let $\mathcal{T}_h$ be a mesh of a domain Ω and u_h some discretization of a function u which is the solution of an elliptic Dirichlet BVP. This yields a linear algebra problem

$$\text{Find } U \in \mathbb{R}^{\#\mathcal{N}} \text{s.t. } A\,U = F\,.$$

Define the restriction operator $r_i = E_i^T$:

$$r_i : \quad u_h \mapsto u_{h|\Omega_i}$$

Let R_i be the boolean matrix corresponding to the restriction operator r_i:

$$R_i := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \dots \end{bmatrix}$$

$$R_i : \ \mathbb{R}^{\#\mathcal{N}} \longmapsto \mathbb{R}^{\#\mathcal{N}_i}$$

Partition of unity

We have

$$R_i : \mathbb{R}^{\#\mathcal{N}} \longmapsto \mathbb{R}^{\#\mathcal{N}_i}$$

and the transpose is a prolongation operator

$$R_i^T : \mathbb{R}^{\#\mathcal{N}_i} \longmapsto \mathbb{R}^{\#\mathcal{N}} .$$

The local Dirichlet matrices are given by

$$A_i := R_i A R_i^T .$$

We also need a kind of partition of unity defined by matrices D_i

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

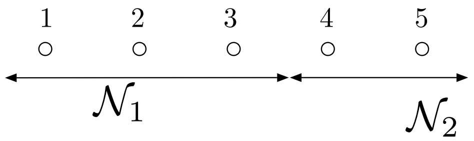
so that we have:

$$\sum_{i=1}^{N} R_i^T D_i R_i = Id$$

Two subdomain case: 1d algebraic setting

Let $\mathcal{N} := \{1, \dots, 5\}$ be partitioned into

$$\mathcal{N}_1 := \{1, 2, 3\} \text{ and } \mathcal{N}_2 := \{4, 5\}\,.$$



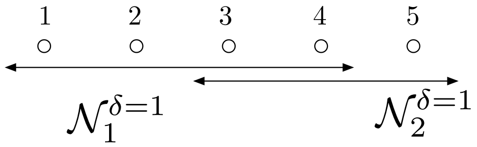
Then, matrices R_1, R_2, D_1 and D_2 are:

$$R_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \text{ and } R_2 = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$D_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } D_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Consider now the case overlapping case

$$\mathcal{N}_1^{\delta=1} := \{1,2,3,4\} \text{ and } \mathcal{N}_2^{\delta=1} := \{3,4,5\}\,.$$



Then, matrices R_1, R_2, D_1, D_2 are:

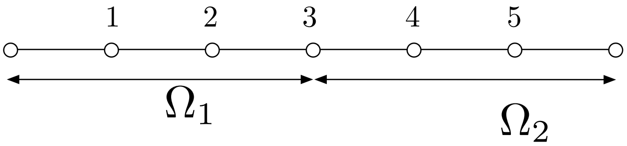
$$R_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \text{ and } R_2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$D_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 \end{pmatrix} \text{ and } D_2 = \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Two subdomain case: 1d finite element decomposition

Partition of the 1D mesh corresponds to an ovr. decomp. of $\mathcal{N}$:

$$\mathcal{N}_1 := \{1,2,3\} \text{ and } \mathcal{N}_2 := \{3,4,5\}\,.$$



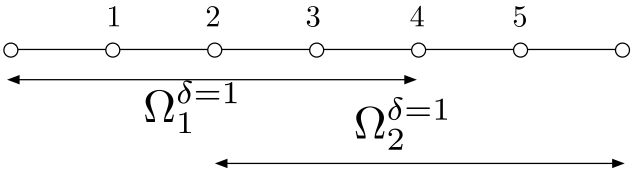
Then, matrices R_1, R_2, D_1, D_2 are:

$$R_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \text{ and } R_2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$D_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/2 \end{pmatrix} \text{ and } D_2 = \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Consider now the situation of an overlapping partition.

$$\mathcal{N}_1^{\delta=1} := \{1,2,3,4\} \text{ and } \mathcal{N}_2^{\delta=1} := \{2,3,4,5\}\,.$$



Then, matrices R_1, R_2, D_1, D_2 are:

$$R_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \text{ and } R_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$D_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 \end{pmatrix} \text{ and } D_2 = \begin{pmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Multi-D and many subdomains: General procedure

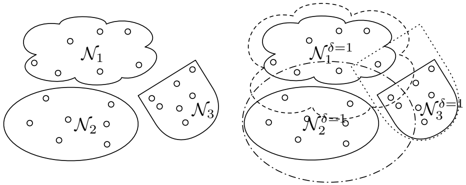
The set of indices $\mathcal{N}$ can be partitioned by an automatic graph partitioner such as **METIS** or **SCOTCH**.

- From the input matrix A, a connectivity graph is created.
- Two indices $i, j \in \mathcal{N}$ are connected if the matrix coefficient $A_{ij} \neq 0$.
- Even if matrix A is not symmetric, the connectivity graph is symmetrized.
- Algorithms that find a good partitioning of highly unstructured graphs are used.
- This distribution must be done so that the number of elements assigned to each processor is roughly the same (balance the computations among the processors).
- The number of adjacent elements assigned to different processors is minimized (minimize the communication between different processors).

Multi-D algebraic setting

Let us consider a partition into N subsets

$$\mathcal{N} := \cup_{i=1}^{N} \mathcal{N}_i, \quad \mathcal{N}_i \cap \mathcal{N}_j = \emptyset \text{ for } i \neq j\,.$$


$\mathcal{N}_1$
$\mathcal{N}_2$
$\mathcal{N}_3$
$\mathcal{N}_1^{\delta=1}$
$\mathcal{N}_2^{\delta=1}$
$\mathcal{N}_3^{\delta=1}$

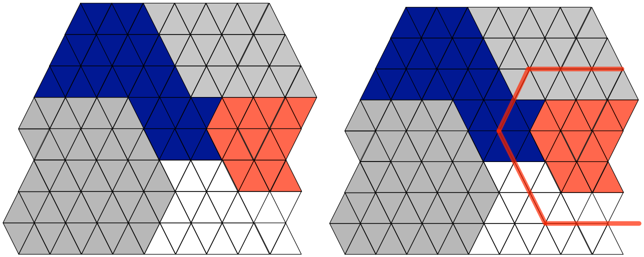
Extend each subset $\mathcal{N}_i$ with its direct neighbors to form $\mathcal{N}_i^{\delta=1}$.
Let R_i be the restriction matrix from set $\mathcal{N}$ to the subset $\mathcal{N}_i^{\delta=1}$
and D_i be a diagonal matrix of size $\#\mathcal{N}_i^{\delta=1} \times \#\mathcal{N}_i^{\delta=1}$, $1 \leq i \leq N$
such that for

$$\mathcal{M}_j := \{1 \leq i \leq N |\, j \in \mathcal{N}_i^{\delta=1}\}\,.$$

and $j \in \mathcal{N}_i^{\delta=1}$, we define $(D_i)_{jj} := 1/\#\mathcal{M}_j$.

Multi-D algebraic finite element decomposition

In a FE setting, the computational domain is the union of elements of the finite element mesh $\mathcal{T}_h$.



It is possible to create overlapping subdomains resolved by the finite element meshes:

$$\Omega_i = \bigcup_{\tau \in \mathcal{T}_{i,h}} \tau \quad \text{for } 1 \le i \le N. \qquad (5)$$

Let $\{\phi_k\}_{k\in\mathcal{N}}$ be a basis of the finite element space. For $1 \le i \le N$, we define

$$\mathcal{N}_i := \{k \in \mathcal{N} : \text{supp}(\phi_k) \cap \Omega_i \ne \emptyset\} .$$

For all degree of freedom $k \in \mathcal{N}$, let

$$\mu_k := \# \{j : 1 \le j \le N \text{ and supp}(\phi_k) \cap \Omega_j \ne \emptyset\} .$$

Let R_i be the restriction matrix from set $\mathcal{N}$ to the subset $\mathcal{N}_i$ and D_i be a diagonal matrix of size $\#\mathcal{N}_i \times \#\mathcal{N}_i$, $1 \le i \le N$. Then, for $k \in \mathcal{N}_i$, we define $(D_i)_{kk} := 1/\mu_k$.

Algebraic formulation - JSM

Define local unknowns $U_i := R_i\, U$ for $i = 1, \ldots, N$ and local right handside $F_i := R_i\, F$.

$$\begin{aligned} R_i\, A\, U &= R_i\, A\, R_i^T\, (R_i\, U) + R_i\, A\, (Id - R_i^T\, R_i) U = F_i \\ &= R_i\, A\, R_i^T\, U_i + R_i\, A\, (Id - R_i^T\, R_i) \textstyle\sum_{j=1}^N R_j^T\, D_j\, R_j\, U \\ &= R_i\, A\, R_i^T\, U_i + \textstyle\sum_{j=1}^N R_i\, A\, (Id - R_i^T\, R_i) R_j^T\, D_j U_j \end{aligned}$$

Notice that $(Id - R_i^T\, R_i) R_i^T\, R_i = 0$ so we have

$$R_i\, A\, R_i^T\, U_i + \sum_{j \neq i} R_i\, A\, (Id - R_i^T\, R_i) R_j^T\, D_j U_j = F_i \qquad (6)$$

Algebraic formulation - JSM

Let us define the block matrix $\widetilde{A}$ (extended matrix)

$$(\widetilde{A})_{ii} := R_i\, A\, R_i^T,\ (\widetilde{A})_{ij} := R_i\, A\, (Id - R_i^T\, R_i) R_j^T\, D_j,\ 1 \le i \ne j \le N$$

Define (extended) unknown vector and right-hand side

$$\begin{aligned} \widetilde{U} &:= (U_1, \ldots, U_i, \ldots, U_N)^T \in \mathbb{R}^{\sum_{i=1}^N \#\mathcal{N}_i}, \\ \widetilde{F} &:= (R_1\, F, \ldots, R_i\, F, \ldots, R_N\, F)^T \in \mathbb{R}^{\sum_{i=1}^N \#\mathcal{N}_i}. \end{aligned}$$

Let $(M_{JSM})_{ii} := (\widetilde{A})_{ii} = R_i\, A\, R_i^T$. The block Jacobi method applied to the (extended) system

$$\widetilde{A}\, \widetilde{U} = \widetilde{F}$$

is the JSM:

$$\widetilde{U}^{n+1} = \widetilde{U}^n + M_{JSM}^{-1} \widetilde{r}^n\, , \widetilde{r}^n := \widetilde{F} - \widetilde{A}\, \widetilde{U}^n. \tag{7}$$

Algebraic formulation - RAS and ASM

As for (RAS), we give the following definition

$$M_{RAS}^{-1} := \sum_{i=1}^{N} R_i^T \, D_i \, (R_i \, A \, R_i^T)^{-1} \, R_i \tag{8}$$

so that the iterative RAS algorithm reads:

$$U^{n+1} = U^n + M_{RAS}^{-1} r^n \, , \, r^n := F - A \, U^n .$$

For (ASM), we give the following definition

$$M_{ASM}^{-1} := \sum_{i=1}^{N} R_i^T \, (R_i \, A \, R_i^T)^{-1} \, R_i \tag{9}$$

so that the iterative ASM algorithm reads:

$$U^{n+1} = U^n + M_{RAS}^{-1} r^n \, .$$

Geometrical analysis in 1d

Let $L > 0$ and the domain $\Omega = (0, L)$ be decomposed into two subodmains $\Omega_1 := (0, L_1)$ and $\Omega_2 := (l_2, L)$ with $l_2 \leq L_1$. The error $e_i^n := u_i^n - u_{|\Omega_i}$, $i = 1, 2$ satisfies

$$\begin{array}{l} -\dfrac{d^2 e_1^{n+1}}{dx^2} = 0 \quad \text{in } (0, L_1) \\ e_1^{n+1}(0) = 0 \\ e_1^{n+1}(L_1) = e_2^n(L_1) \end{array} \quad \text{then,} \quad \begin{array}{l} -\dfrac{d^2 e_2^{n+1}}{dx^2} = 0 \quad \text{in } (l_2, L) \\ e_2^{n+1}(l_2) = e_1^{n+1}(l_2) \\ e_2^{n+1}(L) = 0\,. \end{array} \tag{10}$$

Thus the errors are affine functions in each subdomain:

$$e_1^{n+1}(x) = e_2^n(L_1)\,\frac{x}{L_1} \text{ and } e_2^{n+1}(x) = e_1^{n+1}(l_2)\,\frac{L-x}{L-l_2}\,.$$

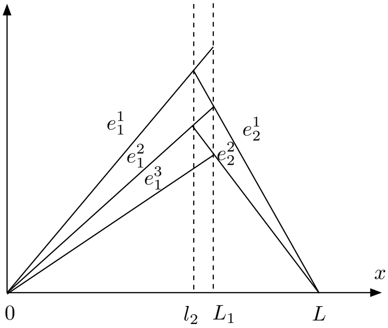
Thus, we have

$$e_2^{n+1}(L_1) = e_1^{n+1}(l_2)\,\frac{L-L_1}{L-l_2} = e_2^n(L_1)\,\frac{l_2}{L_1}\,\frac{L-L_1}{L-l_2}\,.$$

Let $\delta := L_1 - l_2$ denote the size of the overlap, we have

$$e_2^{n+1}(L_1) = \frac{l_2}{l_2+\delta}\,\frac{L-l_2-\delta}{L-l_2}\,e_2^n(L_1) = \frac{1-\delta/(L-l_2)}{1+\delta/l_2}\,e_2^n(L_1)\,.$$

It is clear that $\delta > 0$ is sufficient and necessary to have convergence.



Fourier analysis in 2d - I

Let $\mathbb{R}^2$ decomposed into two half-planes $\Omega_1 = (-\infty, \delta) \times \mathbb{R}$ and $\Omega_2 = (0, \infty) \times \mathbb{R}$ with an overlap of size $\delta > 0$ and the problem

$$\begin{aligned} (\eta - \Delta)(u) &= f \quad \text{in} \quad \mathbb{R}^2, \\ u & \quad \text{is bounded at infinity}, \end{aligned}$$

By linearity, the errors $e_i^n := u_i^n - u|_{\Omega_i}$ satisfy the JSM $f = 0$:

$$\begin{aligned} (\eta - \Delta)(e_1^{n+1}) &= 0 \quad \text{in} \quad \Omega_1 \\ e_1^{n+1} & \quad \text{is bounded at infinity} \\ e_1^{n+1}(\delta, y) &= e_2^n(\delta, y), \end{aligned} \tag{11}$$

$$\begin{aligned} (\eta - \Delta)(e_2^{n+1}) &= 0 \quad \text{in} \quad \Omega_2 \\ e_2^{n+1} & \quad \text{is bounded at infinity} \\ e_2^{n+1}(0, y) &= e_1^n(0, y). \end{aligned} \tag{12}$$

Fourier analysis in 2d - II

By taking the partial Fourier transform of the equation in the y direction we get:

$$\left(\eta - \frac{\partial^2}{\partial x^2} + k^2\right)(\hat{e}_1^{n+1}(x,k)) = 0 \quad \text{in} \quad \Omega_1.$$

For a given k, the solution

$$\hat{e}_1^{n+1}(x,k) = \gamma_+^{n+1}(k)\exp(\lambda^+(k)x) + \gamma_-^{n+1}(k)\exp(\lambda^-(k)x).$$

must be bounded at $x = -\infty$. This implies

$$\hat{e}_1^{n+1}(x,k) = \gamma_+^{n+1}(k)\exp(\lambda^+(k)x)$$

and similarly,

$$\hat{e}_2^{n+1}(x,k) = \gamma_-^{n+1}(k)\exp(\lambda^-(k)x)$$

Fourier analysis in 2d - III

From the interface conditions we get

$$\gamma_+^{n+1}(k) = \gamma_-^{n}(k)\exp(\lambda^-(k)\delta),\ \gamma_-^{n+1}(k) = \gamma_+^{n}(k)\exp(-\lambda^+(k)\delta).$$

Combining these two and denoting $\lambda(k) = \lambda^+(k) = -\lambda^-(k)$, we get for $i = 1, 2$,

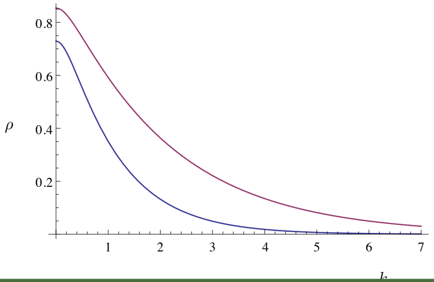
$$\gamma_\pm^{n+1}(k) = \rho(k;\alpha,\delta)^2\,\gamma_\pm^{n-1}(k)$$

with ρ the convergence rate given by:

$$\rho(k;\alpha,\delta) = \exp(-\lambda(k)\delta), \tag{13}$$

where $\lambda(k) = \sqrt{\eta + k^2}$.

Fourier analysis in 2d - IV



Remark

We have the following properties:

- *For all* $k \in \mathbb{R}$, $\rho(k) < \exp(-\sqrt{\eta}\,\delta) < 1$ *so that* $\gamma_i^n(k) \to 0$ *uniformly as* n *goes to infinity.*
- $\rho \to 0$ *as* k *tends to infinity, high frequency modes of the error converge very fast.*
- *When there is no overlap (*$\delta = 0$*),* $\rho = 1$ *and there is stagnation of the method.*

About FreeFem++ (survival kit)

`FreeFem++` allows a very simple and natural way to solve a great variety of variational problems (FEM, DG).

It is possible to have access to the underlying linear algebra such as the stiffness or mass matrices.

Tutorial: `http://www.cmap.polytechnique.fr/spip.php?article239`.

A very detailed documentation of `FreeFem++` is available on the official website `http://www.freefem.org/ff++`

`http://www.freefem.org/ff++/ftp/freefem++doc.pdf`

Let a homogeneous Dirichlet boundary value problem for a Laplacian defined on a unit square $\Omega =]0,1[^2$:

$$\begin{cases} -\Delta u = f & \text{dans } \Omega \\ u = 0 & \text{sur } \partial\Omega \end{cases} \tag{14}$$

The variational formulation of the problem

Find $u \in H_0^1(\Omega) := \{w \in H^1(\Omega) \ : \ w = 0, \text{ on } \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4\}$

such that

$$\int_\Omega \nabla u . \nabla v dx - \int_\Omega f \, v \, dx = 0, \forall v \in H_0^1(\Omega) \, .$$

Feature of `Freefem++`: penalization of Dirichlet BC. Let *TGV* (*Très Grande Valeur* in French) be a very large value, the above variational formulation is approximated by
Find $u \in H^1(\Omega)$ such that

$$\int_\Omega \nabla u . \nabla v dx + TGV \int_{\cup_{i=1,\dots,4}\Gamma_i} u \, v - \int_\Omega fv \, dx = 0, \forall v \in H^1(\Omega) \, .$$

The following `FreeFem++` script is solving this problem

```
// Number of mesh points in x and y directions
int Nbnoeuds=10;
```

The text after `//` symbols are comments ignored by the `FreeFem++` language.
Each new variable must be declared with its type (here `int` designs integers).

```
//Mesh definition
mesh Th=square(Nbnoeuds,Nbnoeuds,[x,y]);
```

The function `square` returns a structured `mesh` of the square, the sides of the square are labelled from 1 to 4 in trigonometrical sense.

Define the function representing the right hand side

```
// Function of x and y
func f=x*y;
```

and the P_1 finite element space `Vh` over the mesh `Th`.

```
// Finite element space on the mesh Th
fespace Vh(Th,P1);
//uh and vh are of type Vh
Vh uh,vh;
```

The functions u_h and v_h belong to the P_1 finite element space V_h which is an approximation to $H^1(\Omega)$.

```
// variational problem definition
problem heat(uh,vh,solver=LU)=
        int2d(Th)(dx(uh)*dx(vh)+dy(uh)*dy(vh))
        -int2d(Th)(f*vh)
        +on(1,2,3,4,uh=0);
```

The keyword `problem` allows the definition of a variational problem (without solving it)

$$\int_{\Omega} \nabla u_h . \nabla v_h dx + TGV \int_{\cup_{i=1,\dots,4}\Gamma_i} u_h\, v_h - \int_{\Omega} f v_h dx = 0, \forall v_h \in V_h\,.$$

where TGV is equal to 10^{30}.
The parameter `solver` sets the method that will be used to solve the resulting linear system. To solve the problem we need

```
//Solving the problem
heat;
// Plotting the result
plot(uh,wait=1);
```

The `Freefem++` script can be saved with your favourite text editor (e.g. under the name `heat.edp`). In order to execute the script write the shell command

```
FreeFem++ heat.edp
```

The result will be displayed in a graphic window.

Solve Neumann or Fourier boundary conditions such as

$$\begin{cases} -\Delta u + u = f & \text{dans } \Omega \\ \frac{\partial u}{\partial n} = 0 & \text{sur } \Gamma_1 \\ u = 0 & \text{sur } \Gamma_2 \\ \frac{\partial u}{\partial n} + \alpha u = g & \text{sur } \Gamma_3 \cup \Gamma_4 \end{cases} \tag{15}$$

The new variational formulation consists in determining $u_h \in V_h$ such that

$$\int_\Omega \nabla u_h . \nabla v_h dx + \int_{\Gamma_3 \cup \Gamma_4} \alpha u_h v_h + TGV \int_{\Gamma_2} u_h . v_h$$
$$- \int_{\Gamma_3 \cup \Gamma_4} g v_h - \int_\Omega f v_h dx = 0, \forall v_h \in V_h.$$

The `Freefem++` definition of the problem

```
problem heat(uh,vh)=
int2d(Th)(dx(uh)*dx(vh)+dy(uh)*dy(vh))
+int1d(Th,3,4)(alpha*uh*vh)
-int1d(Th,3,4)(g*vh)
-int2d(Th)(f*vh)
+on(2,uh=0);
```

In order to use some **linear algebra** package, we need the matrices. The keyword `varf` allows the definition of a variational formulation

```
varf heat(uh,vh)=
int2d(Th)(dx(uh)*dx(vh)+dy(uh)*dy(vh))
+int1d(Th,3,4)(alpha*uh*vh)
-int1d(Th,3,4)(g*vh)
-int2d(Th)(f*vh)
+on(2,uh=0);
matrix Aglobal; // stiffness sparse matrix
Aglobal = heat(Vh,Vh,solver=UMFPACK);// UMFPACK solver
Vh rhsglobal;  //right hand side vector
rhsglobal[] = heat(0,Vh);
```

Here `rhsglobal` is a FE function and the associated vector of d.o.f. is `rhsglobal[]`.
The linear system is solved by using UMFPACK

```
// Solving the problem by a sparse LU sover
uh[] = Aglobal^-1*rhsglobal[];
```

Decomposition into overlapping domains I

Suppose we want a decomposition of a rectangle Ω into `nn`$\times$`mm` domains with approximately `nloc` points in one direction.

```
int nn=4,mm=4;
int npart= nn*mm;
int nloc = 20;
real allong = 1;
allong = real(nn)/real(mm);

mesh Th=square(nn*nloc*allong,mm*nloc,[x*allong,y]);
fespace Vh(Th,P1);
fespace Ph(Th,P0);
Ph  part;
Ph xx=x,yy=y;
part = int(xx/allong*nn)*mm + int(yy*mm);
plot(part,fill=1,value=1,wait=1,ps="decompunif.eps");
```

For arbitrary decompositions, use METIS or SCOTCH.

```
int nn=4,mm=4;
int npart= nn*mm;
int nloc = 20;
real allong = 1;
allong = real(nn)/real(mm);

mesh Th=square(nn*nloc*allong,mm*nloc,[x*allong,y]);
fespace Vh(Th,P1);
fespace Ph(Th,P0);
Ph  part;
bool withmetis = 1;
if(withmetis)  // Metis partition
  {
    load "metis";
    int[int] nupart(Th.nt);
    metisdual(nupart,Th,npart);
    for(int i=0;i<nupart.n;++i)
      part[][i]=nupart[i];
  }
plot(part,fill=1,value=1,wait=1,ps="decompMetis.eps");
```

Decomposition into overlapping domains II

To build the overlapping decomposition and the associated algebraic call the routine `SubdomainsPartitionUnity`.

Output:

- overlapping meshes `aTh[i]`
- the restriction/interpolation operators `Rih[i]` from the local finite element space `Vh[i]` to the global one `Vh`
- the diagonal local matrices `Dih[i]` from the partition of unity.

```
include "createPartition.edp";
include "decompMetis.edp";

// overlapping partition
int sizeovr = 3;
mesh[int] aTh(npart);    // sequence of ovr. meshes
matrix[int] Rih(npart); // local restriction operators
matrix[int] Dih(npart); // partition of unity operators

SubdomainsPartitionUnity(Th,part[],sizeovr,aTh,Rih,Dih);
```

RAS and ASM: global data

We first need to define the global data.

```
// Solve Dirichlet subproblem Delta (u) = f
// u = 1 on the global boundary
int[int] chlab=[1,1  ,2,1  ,3,1  ,4,1  ];
Th=change(Th,refe=chlab);

macro Grad(u) [dx(u),dy(u)]          // EOM
func f = 1;                          // right hand side
func g = 1;                          // Dirichlet data

// global problem
Vh rhsglobal,uglob;
varf vaglobal(u,v) =
    int2d(Th)(Grad(u)'*Grad(v))
    +on(1,u=g) + int2d(Th)(f*v);
matrix Aglobal;
Aglobal = vaglobal(Vh,Vh,solver = UMFPACK);  // matrix
rhsglobal[] = vaglobal(0,Vh);                // rhs
uglob[] = Aglobal^-1*rhsglobal[];
```

RAS and ASM: local data

And then the local problems

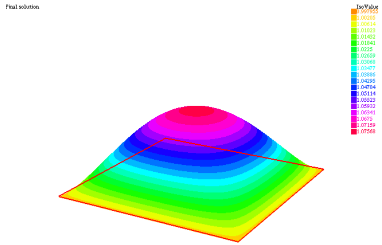
```
// overlapping partition
int sizeovr = 4;
mesh[int] aTh(npart);                // overlapping meshes
matrix[int] Rih(npart);              // restriction operators
matrix[int] Dih(npart);              // partition of unity
SubdomainsPartitionUnity(Th,part[],sizeovr,aTh,Rih,Dih);

matrix[int] aA(npart);               // Dirichlet matrices
for(int i = 0;i<npart;++i)
  {
    cout << " Domain :" << i << "/" << npart << endl;
    matrix aT = Aglobal*Rih[i]';
    aA[i] =  Rih[i]*aT;
    set(aA[i],solver = UMFPACK);// direct solvers
  }
```

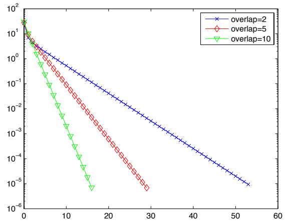
RAS and ASM : Schwarz iteration

```
int nitermax = 1000;
Vh un = 0, rn = rhsglobal;                // initial guess
for(int iter = 0;iter<nitermax;++iter)
 {real err = 0;
  Vh er = 0;
  for(int i = 0;i<npart;++i)
     {real[int] bi = Rih[i]*rn[];         // restriction
      real[int] ui = aA[i] ^-1 * bi;      // local solve
      bi = Dih[i]*ui;                     // bi = ui;  ASM
        er[] += Rih[i]'*bi; }
    un[] += er[];                  // build new iterate
    rn[] = Aglobal*un[];           // global residual
    rn[] = rn[] - rhsglobal[];
    rn[] *= -1;
    err = sqrt(er[]'*er[]);
    cout << "Iter: " << iter << " Err = " << err << endl
    if(err < 1e-5) break;
    plot(un,wait=1,value=1,fill=1,dim=3);}
plot(un,wait=1,value=1,fill=1,dim=3,ps = "solution.eps")
```

Convergence

Convergence history of the RAS solver for different values of the overlapping parameter.





Note that this convergence, not very fast even in a simple configuration of 4 subdomains.
The iterative version of ASM does not converge. For this reason, the ASM method is always used a preconditioner for a Krylov method such as CG, GMRES or BiCGSTAB.

Outline

Fixed point method

Consider a well-posed but difficult to solve linear system

$$A\,x = b$$

and B an "easy to invert" matrix of the same size than A. A possible iterative method is a fixed point algorithm

$$x^{n+1} = x^n + B^{-1}(b - A\,x^n)$$

and x is a fixed point of the operator:

$$x \longmapsto x + B^{-1}(b - A\,x)\,.$$

Let $r_0 := b - Ax^0$ and $C := B^{-1}\,A$, a direct computation shows that we have:

$$x^n = \sum_{i=0}^{n} (I_d - C)^i\, B^{-1} r_0 + x^0\,. \qquad (16)$$

We have convergence iff the spectral radius of the matrix $I_d - C$ is smaller than one.

Why Krylov methods I

Consider now a preconditioned Krylov applied to the linear system:

$$B^{-1} A x = B^{-1} b$$

Let us denote x^0 an initial guess and $r^0 := B^{-1} b - C x^0$ the initial residual. Then $y := x - x^0$ solves

$$C y = r^0 .$$

The basis for Krylov methods is the following

Lemma

Let C be an invertible matrix of size $N \times N$.
Then, there exists a polynomial $\mathcal{P}$ of degree $p < N$ such that

$$C^{-1} = \mathcal{P}(C) .$$

Proof.

Let be a minimal polynomial of C of degree $d \leq N$:

$$\mathcal{M}(X) := \sum_{i=0}^{d} a_i X^i$$

We have $\sum_{i=0}^{d} a_i C^i = 0$ and there is no non zero polynomial of lower degree that annihilates C. Thus, a_0 cannot be zero since

$$C \sum_{i=1}^{d} a_i C^{i-1} = 0 \Rightarrow \sum_{i=1}^{d} a_i C^{i-1} = 0 \, .$$

Then, $\sum_{i=0}^{d-1} a_{i+1} X^i$ would be an annihiling polynomial of C of degree lower than d. This implies

$$I_d + C \sum_{i=1}^{d} \frac{a_i}{a_0} C^{i-1} = 0 \Rightarrow C^{-1} := -\sum_{i=1}^{d} \frac{a_i}{a_0} C^{i-1} \, .$$

Coming back to the linear system, we have

$$x = x^0 + \sum_{i=1}^{d} (-\frac{a_i}{a_0}) C^{i-1}\, r^0\,.$$

Thus, it makes sense to introduce Krylov spaces, $\mathcal{K}^n(C, r^0)$

$$\mathcal{K}^n(C, r^0) := Span\{r^0,\, Cr^0, \dots, C^{n-1} r^0\},\; n \geq 1.$$

to seek y^n an approximation to y.
Example: The CG methods applies to symmetric positive definite (SPD) matrices and minimizes the A^{-1}-norm of the residual when solving $Ax = b$:

$$CG \left\{ \begin{array}{l} \text{Find } y^n \in \mathcal{K}^n(A, r^0) \text{ such that} \\ \|A\,y^n - r^0\|_{A^{-1}} = \min\limits_{w \in \mathcal{K}^n(A, r^0)} \|A\,w - r^0\|_{A^{-1}}\,. \end{array} \right.$$

A detailed analysis reveals that $x^n = y^n + x_0$ can be obtained by the quite cheap recursion formula:

```
for $i = 1, 2, \dots$ do
    $\rho_{i-1} = (r_{i-1}, r_{i-1})_2$
    if $i = 1$ then
        $p_1 = r_0$
    else
        $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
        $p_i = r_{i-1} + \beta_{i-1} p_{i-1}$
    end if
    $q_i = A p_{i-1}$
    $\alpha_i = \frac{\rho_{i-1}}{(p_i, q_i)_2}$
    $x_i = x_{i-1} + \alpha_i p_i$
    $r_i = r_{i-1} - \alpha_i q_i$
    check convergence; continue if necessary
end for
```

Preconditioned Krylov

By solving an optimization problem:

$$GMRES \left\{ \begin{array}{l} \text{Find } y^n \in \mathcal{K}^n(C, r^0) \text{ such that} \\ \|C\,y^n - r^0\|_2 = \min\limits_{w \in \mathcal{K}^n(C, r^0)} \|C\,w - r^0\|_2 \end{array} \right.$$

a preconditioned Krylov solve will generate an optimal x_K^n in

$$\mathcal{K}^n(C, B^{-1} r_0) := x_0 + Span\{B^{-1} r_0,\; C\,B^{-1} r_0, \ldots, C^{n-1}\,B^{-1} r_0\}\,.$$

This minimization problem is of size n. When n is small w.r.t. N, its solving has a marginal cost. Thus, x_K^n has a computing cost similar to that of x^n. But, since $x^n \in \mathcal{K}^n(B^{-1}A, B^{-1} r_0)$ as well but with “frozen” coefficients, we have that x_n is less optimal (actually much much less) than x_K^n.

Schwarz methods as preconditioners

In the previous Krylov methods we can use as preconditioner

- RAS (in conjunction with BiCGStab or GMRES)

$$B^{-1} := M_{RAS}^{-1} = \sum_{i=1}^{N} R_i^T \, D_i \, (R_i \, A \, R_i^T)^{-1} \, R_i$$

- ASM (in a CG methods)

$$B^{-1} := M_{ASM}^{-1} = \sum_{i=1}^{N} R_i^T \, (R_i \, A \, R_i^T)^{-1} \, R_i$$

Preconditioner in CG

We use

- M_{ASM}^{-1} as a preconditioner
- a Krylov method: conjugate gradient since M_{ASM}^{-1} and A are symmetric.

At iteration m the error for the PCG method is bounded by:

$$||\bar{x} - x_m||_{M_{ASM}^{-\frac{1}{2}} A M_{ASM}^{-\frac{1}{2}}} \leq 2 \left[\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right]^m ||\bar{x} - x_0||_{M_{ASM}^{-\frac{1}{2}} A M_{ASM}^{-\frac{1}{2}}}.$$

where κ is the condition number of $M_{ASM}^{-1} A$ and $\bar{x}$ is the exact solution.

The CG with the ASM preconditioner becomes:

```
for $i = 1, 2, \ldots$ do
    $\rho_{i-1} = (r_{i-1}, M_{ASM}^{-1} r_{i-1})_2$
    if $i = 1$ then
        $p_1 = M_{ASM}^{-1} r_0$
    else
        $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
        $p_i = M_{ASM}^{-1} r_{i-1} + \beta_{i-1} p_{i-1}$
    end if
    $q_i = A p_{i-1}$
    $\alpha_i = \frac{\rho_{i-1}}{(p_i, q_i)_2}$
    $x_i = x_{i-1} + \alpha_i p_i$
    $r_i = r_{i-1} - \alpha_i q_i$
    check convergence; continue if necessary
end for
```

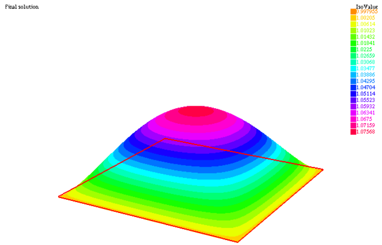
The action of the global operator is given by

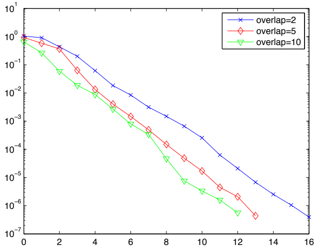
```
Vh rn, s;
func real[int] A(real[int] &l)                    // A*u
{
  rn[]= Aglobal*l;
  return rn[];
}
```

The preconditioning method can be Additive Schwarz (ASM)

```
func real[int] Mm1(real[int] &l)
{
   s = 0;
   for(int i=0;i<npart;++i)
     {
       mesh Thi = aTh[i];
       real[int] bi = Rih[i]*l;          // restricts rhs
       real[int] ui = aA[i] ^-1 * bi;    // local solves
       s[] += Rih[i]'*ui;                // prolongation
      }
   return s[];
}
```

The Krylov method applied in this case is the CG. The performance is now less sensitive to the overlap size.





We can also use RAS as a preconditioner, by taking into account the partition of unity

```
func real[int] Mm1(real[int] &l)
{
   s = 0;
   for(int i=0;i<npart;++i)
     {
       mesh Thi = aTh[i];
       real[int] bi = Rih[i]*l;         // restricts rhs
       real[int] ui = aA[i] ^-1 * bi; // local solves
       bi = Dih[i]*ui;
       s[] += Rih[i]'*bi;               // prolongation
     }
   return s[];
}
```

this time in conjuction with BiCGStab since we deal with non-symmetric problems.

Weak scalability

How to evaluate the efficiency of a domain decomposition?

Weak scalability – definition

”How the solution time varies with the number of processors for a fixed problem size per processor.”

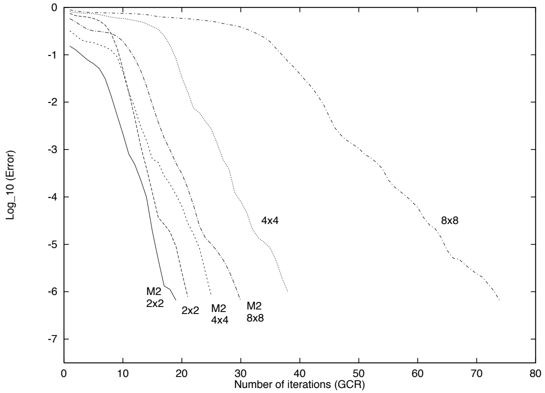
It is not achieved with the one level method

Number of subdomains	8	16	32	64
ASM	18	35	66	128

The iteration number increases linearly with the number of subdomains in one direction.

Convergence curves- more subdomains

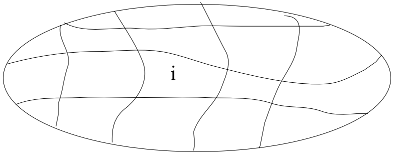
Plateaus appear in the convergence of the Krylov methods.



How to achieve scalability

Stagnation corresponds to a few very low eigenvalues in the spectrum of the preconditioned problem. They are due to the lack of a global exchange of information in the preconditioner.

$$-\Delta u = f \text{ in } \Omega$$
$$u = 0 \text{ on } \partial\Omega$$



The mean value of the solution in domain *i* depends on the value of *f* on all subdomains.
A classical remedy consists in the introduction of a coarse grid problem that couples all subdomains. This is closely related to deflation technique classical in linear algebra (see Nabben and Vuik's papers in SIAM J. Sci. Comp, 200X).

Outline

Adding a coarse grid

We add a coarse space correction (*aka* second level)
Let V_H be the coarse space and z be a basis, $V_H = \mathrm{span}\, z$, writing $R_0 = Z^T$ we define the two level preconditioner as:

$$M_{ASM,2}^{-1} := R_0^T (R_0 A R_0^T)^{-1} R_0 + \sum_{i=1}^{N} R_i^T A_i^{-1} R_i.$$

The Nicolaides approach is to use the kernel of the operator as a coarse space, this is the constant vectors, in local form this writes:

$$Z := (R_i^T D_i R_i \mathbf{1})_{1 \le i \le N}$$

where D_i are chosen so that we have a partition of unity:

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

Theoretical convergence result

Theorem (Widlund, Sarkis)

Let $M_{ASM,2}^{-1}$ be the two-level additive Schwarz method:

$$\kappa(M_{ASM,2}^{-1}A) \leq C\left(1+\frac{H}{\delta}\right)$$

where δ is the size of the overlap between the subdomains and H the subdomain size.

This does indeed work very well

Number of subdomains	8	16	32	64
ASM	18	35	66	128
ASM + Nicolaides	20	27	28	27

Idea of the proof (Upper bound)

Lemma

If each point in Ω belongs to at most k_0 of the subdomains Ω_j, then the largest eigenvalue of $M_{ASM,2}^{-1}A$ satisfies

$$\lambda_{max}(M_{ASM,2}^{-1}A) \;\leq\; k_0 + 1 .$$

Assumption (Stable decomposition)

There exists a constant C_0, such that every $u \in V$ admits a decomposition $u = \sum_{i=0}^{N} R_i^T u_i,\; u_i \in V_i,\; i = 0, \ldots, N$ that satisfies:

$$\sum_{i=0}^{N} \tilde{a}_i(u_i, u_i) \leq C_0^2 a(u, u).$$

Idea of the proof (Lower bound)

Theorem

If every $v \in V$ admits a C_0-stable decomposition (with uniform C_0), then the smallest eigenvalue of $M_{AS,2}^{-1} A$ satisfies

$$\lambda_{min}(M_{ASM,2}^{-1} A) \geq C_0^{-2}.$$

Therefore, the condition number of the two-level Schwarz preconditioner can be bounded by

$$\kappa(M_{ASM,2}^{-1} A) \leq C_0^2 (k_0 + 1).$$

Deflation and Coarse grid correction

Let A be a SPD matrix, we want to solve

$$Ax = b$$

with a preconditioner M (for example the Schwarz method). Let Z be a rectangular matrix so that the "bad eigenvectors" belong to the space spanned by its columns. Define

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

Examples of coarse grid preconditioners

$$\mathcal{P}_{A-DEF2} := P^TM^{-1}+Q\,, \quad \mathcal{P}_{BNN} := P^TM^{-1}P+Q \text{ (Mandel, 1993)}$$

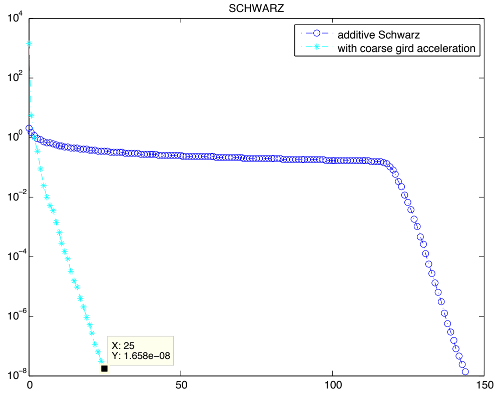
Some properties: $QAZ = Z$, $P^TZ = 0$ and $P^T\,Q = 0$.
Let r_n be the residual at step n of the algorithm: $Z^T\,r_n = 0$.

How to choose Z?

Coarse grid correction for smooth problems

For a Poisson like problem, Nicolaides (1987), Sarkis (2002).
Let $(\chi_i)_{1\le i\le N}$ denote a partition of unity :

$$Z = \begin{bmatrix} \chi_1 & 0 & \cdots & 0 \\ \vdots & \chi_2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \chi_N \end{bmatrix}$$



Coarse grid implementation - I

It is enough to replace the Schwarz preconditioner by P_{BNN} as follows. First build $E = Z^T AZ$

```
Vh[int]  Z(npart);
for(int i=0;i<npart;++i)
{ Z[i]=1.;
real[int] zit = Rih[i]*Z[i][];
real[int] zitemp = Dih[i]*zit;
Z[i][]=Rih[i]'*zitemp;
}
real[int,int] Ef(npart,npart);   // E = Z^T*A*Z
for(int i=0;i<npart;++i)
{  real[int] vaux = A(Z[i][]);
   for(int j=0;j<npart;++j)
  Ef(j,i) = Z[j][]'*vaux;
}
matrix E;
E = Ef;
set(E,solver=UMFPACK);
```

Coarse grid implementation - II

Then the coarse space correction $Q = ZE^{-1}Z^T$:

```
func real[int] Q(real[int] &l)    // Q = Z*E^-1*Z^T
{
   real[int] res(l.n);
   res=0.;
   real[int] vaux(npart);
   for(int i=0;i<npart;++i)
   {
      vaux[i]=Z[i][]'*l;
   }
   real[int] zaux=E^-1*vaux;      // zaux=E^-1*Z^T*l
   for(int i=0;i<npart;++i)       // Z*zaux
   {
      res +=zaux[i]*Z[i][];
   }
   return res;
}
```

Coarse grid implementation - III

The projector out of the coarse space $P = I - QA$ and its transpose P^T:

```
func real[int] P(real[int] &l)    // P = I - A*Q
{
   real[int] res=Q(l);
   real[int] res2=A(res);
   res2 -= l;
   res2 *= -1.;
   return res2;
}
func real[int] PT(real[int] &l)  // P^T = I-Q*A
{
   real[int] res=A(l);
   real[int] res2=Q(res);
   res2 -= l;
   res2 *= -1.;
   return res2;
}
```

Coarse grid implementation - IV

And finally the preconditioner $P_{BNN} = P^T M^{-1} P + Q$:

```
int j;
func real[int] BNN(real[int] &u)    // precond BNN
{
    real[int] aux1 = Q(u);
    real[int] aux2 = P(u);
    real[int] aux3 = Mm1(aux2);
    aux2 = PT(aux3);
    aux2 += aux1;
    ++j;
return aux2;
}
```

Outline

Motivation

Large discretized system of PDEs
strongly heterogeneous coefficients
(high contrast, nonlinear, multiscale)

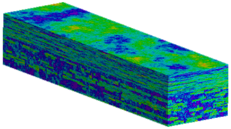
E.g. Darcy pressure equation,
P^1-finite elements:

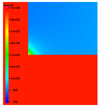
$$\mathbf{A}\mathbf{u} = \mathbf{f}$$

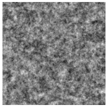
$$\text{cond}(\mathbf{A}) \sim \frac{\alpha_{\max}}{\alpha_{\min}} h^{-2}$$

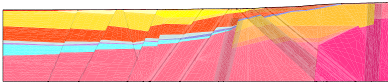
Goal:
iterative solvers
robust in size and heterogeneities

Applications:
flow in heterogeneous /
stochastic / layered media
structural mechanics
electromagnetics
etc.



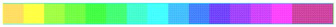




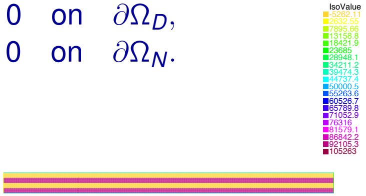


Darcy equation with heterogeneities

$$\begin{aligned} -\nabla \cdot (\alpha(x,y)\nabla u) &= 0 \quad \text{in} \quad \Omega \subset \mathbb{R}^2, \\ u &= 0 \quad \text{on} \quad \partial\Omega_D, \\ \frac{\partial u}{\partial n} &= 0 \quad \text{on} \quad \partial\Omega_N. \end{aligned}$$



Decomposition



$\alpha(x, y)$

Jump	1	10	10^2	10^3	10^4
ASM	39	45	60	72	73
ASM + Nicolaides	30	36	50	61	65

Objectives

Strategy

Define an appropriate coarse space $V_{H2} = \mathrm{span}(z_2)$ and use the framework previously introduced, writing $R_0 = Z_2^T$ the two level preconditioner is:

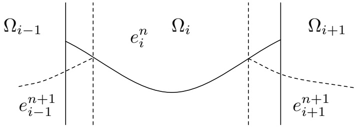
$$P_{ASM2}^{-1} := R_0^T (R_0 A R_0^T)^{-1} R_0 + \sum_{i=1}^{N} R_i^T A_i^{-1} R_i.$$

The coarse grid must be

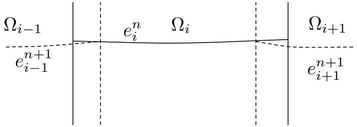
- Local (calculated on each subdomain) $\rightarrow$ parallel
- Adaptive (calculated automatically)
- Easy and cheap to compute (on the boundary for instance)
- Robust (must lead to an algorithm whose convergence does not depend on the partition or the jumps in coefficients)

Heuristic approach: what functions should be in Z_2?

The error satisfies the Schwarz algorithm, it is harmonic, so it satisfies a maximum principle.



Fast convergence



Slow convergence

Idea

Ensure that the error decreases quickly on the subdomain boundaries which translates to making $\left.\frac{\partial e}{\partial n_i}\right|_{\Gamma_i}$ big.

Ensuring that the error decreases quickly on the subdomain boundaries

The Dirichlet to Neumann operator is defined as follows: Let $g : \Gamma_i \mapsto \mathbb{R}$,

$$\mathrm{DtN}_{\Omega_i}(g) = \alpha \left.\frac{\partial v}{\partial n_i}\right|_{\Gamma_i},$$

where v satisfies

$$\begin{cases} (-\mathrm{div}(\alpha\nabla))v = 0, & \text{in } \Omega_i, \\ v = g, & \text{on } \partial\Omega_i. \end{cases}$$

To construct the coarse space, we use the low frequency modes associated with the DtN operator:

$$\mathrm{DtN}_{\Omega_i}(v_i^\lambda) = \lambda\,\alpha\, v_i^\lambda$$

with λ small. The functions v_i^λ are extended harmonically to the subdomains.

Theoretical convergence result

Suppose we have $(v_i^{\lambda_k}, \lambda_i^k)_{1 \le k \le n_{\Gamma_i}}$ the eigenpairs of the local DtN maps ($\lambda_i^1 \le \lambda_i^2 \le \ldots$) and that we have selected m_i in each subdomain. Then let Z be the coarse space built via the local DtN maps:

$$Z := (R_i^T \, D_i \, \tilde{V}_i^{\lambda_i^k})_{1 \le i \le N;\, 1 \le k \le m_i}$$

Theorem (D., Nataf, Scheichl and Spillane 2010)

Under the monotonicity of α in the overlapping regions:

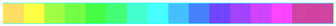
$$\kappa(M_{ASM,2}^{-1} A) \le C(1 + \max_{1 \le i \le N} \frac{1}{\delta_i \, \lambda_i^{m_i+1}})$$

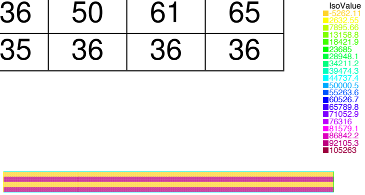
where δ_i is the size of the overlap of domain Ω_i and C is independent of the jumps of α.

If m_i is chosen so that, $\lambda_i^{m_i+1} \ge 1/H_i$ the convergence rate will be analogous to the constant coefficient case.

Results with the new DtN method

Jump	1	10	10^2	10^3	10^4
ASM	39	45	60	72	73
ASM + Nicolaides	30	36	50	61	65
ASM + DtN	31	35	36	36	36

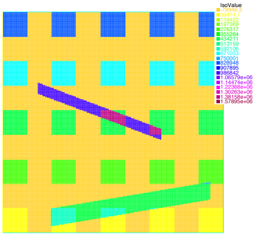

Decomposition

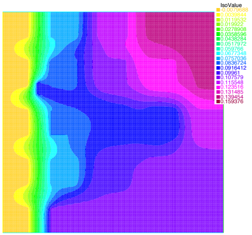


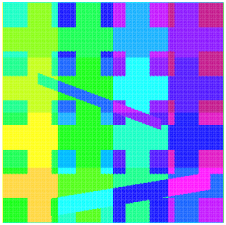
$\alpha(x, y)$

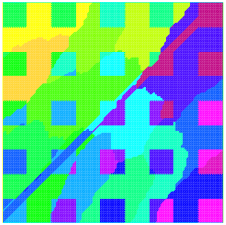
With DtN the jumps do not affect convergence
We put at most two modes per subdomain in the coarse grid (using the automatic selection process)

Numerical results



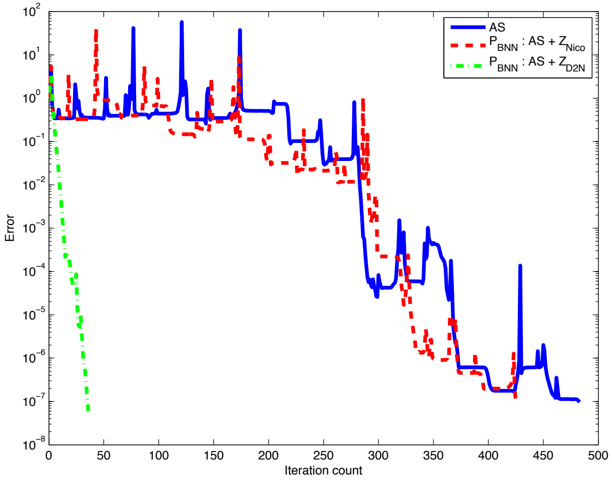






Channels and inclusions: $1 \leq \alpha \leq 1.5 \times 10^6$, the solution and partitionings (Metis or not)

Numerical results



ASM convergence for channels and inclusions – 4×4 Metis partitioning

Numerical results

subdomain i	m_i	total number of eigenvalues
1	3	155
2	1	109
3	5	175
10	4	174
11	2	71
12	2	128
13	3	166
14	3	127
15	3	188
16	3	106

Metis 4 by 4 decomposition

Numerical results

	ASM	ASM+Nico	ASM+DtN
2×2	103	110	22
2×2 Metis	76	76	22
4×4	603	722	26
4×4 Metis	483	425	36
8×8	461	141	34
8×8 Metis	600	542	31

Convergence results for the ”hard” test case

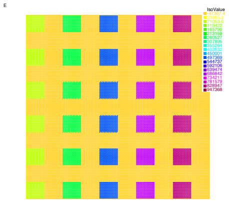
Numerical results – Optimality

$\#Z$ per subd.	ASM	ASM+Z_{Nico}	ASM+Z_{D2N}
$\max(m_i - 1, 1)$			273
m_i	614	543	36
$m_i + 1$			32

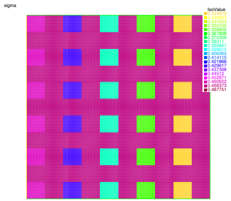
m_i is given automatically by the strategy.

- Taking one fewer eigenvalue has a huge influence on the iteration count
- Taking one more has only a small influence

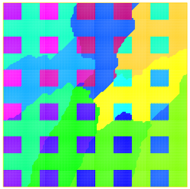
Results for elasticity (Problem)



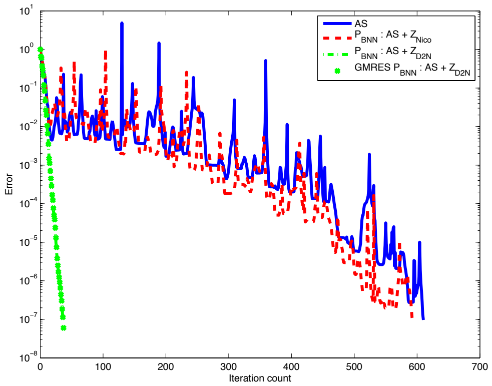
Young's modulus ($1 \leq E \leq 10^6$)



Poisson's ratio ($0.35 \leq \nu \leq 0.48$)



Results for 2d elasticity (Solution)



Overlap is two grid cells

Outline

Problem setting – I

Given $f \in (V^h)^*$ find $u \in V^h$

$$\begin{aligned} a(u, v) &= \langle f, v\rangle \qquad \forall v \in V^h \\ \iff \qquad \mathbf{A}\,\mathbf{u} &= \mathbf{f} \end{aligned}$$

Assumption throughout: $\mathbf{A}$ *symmetric positive definite (SPD)*

Examples:

- Darcy $\quad a(u, v) = \int_\Omega \boldsymbol{\kappa} \nabla u \cdot \nabla v \, dx$
- Elasticity $\quad a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{C}\, \varepsilon(\boldsymbol{u}) : \varepsilon(\boldsymbol{v})\, dx$
- Eddy current $\quad a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \boldsymbol{\nu}\, \mathrm{curl}\, \boldsymbol{u} \cdot \mathrm{curl}\, \boldsymbol{v} + \boldsymbol{\sigma}\, \boldsymbol{u} \cdot \boldsymbol{v}\, dx$

Heterogeneities / high contrast / nonlinearities in parameters

Problem setting – II

1. $V^h \dots$ FE space of functions in Ω based on mesh $\mathcal{T}^h = \{\tau\}$
2. $\mathbf{A}$ given as set of **element stiffness matrices**
   + connectivity (list of DOF per element)

   Assembling property:

$$a(v, w) \; = \sum_{\tau} a_\tau(v_{|\tau}, w_{|\tau})$$

   where $a_\tau(\cdot,\cdot)$ *symm. pos. semi-definite*
3. $\{\phi_k\}_{k=1}^n$ (FE) basis of V^h
   on each element: *unisolvence*

   set of non-vanishing basis functions linearly independent

   fulfilled by standard FE

   continuous, Nédélec, Raviart-Thomas of low/high order
4. Two more assumptions on $a(\cdot,\cdot)$ later!

Schwarz setting – I

Overlapping partition: $\Omega = \bigcup_{j=1}^{N} \Omega_j$ (Ω_j union of elements)

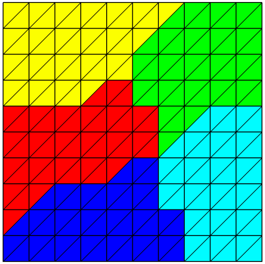
$$V_j := \mathrm{span}\{\phi_k : \mathrm{supp}(\phi_k) \subset \overline{\Omega}_j\}$$

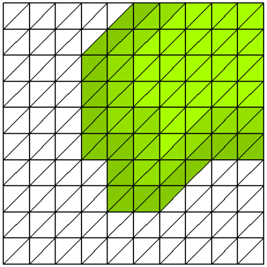
such that every ϕ_k contained in one of those spaces, i.e.

$$V^h = \sum_{j=1}^{N} V_j$$

Example: adding “layers” to non-overlapping partition

(partition and adding layers based on matrix information only!)





Schwarz setting – II

Local subspaces:

$$V_j \subset V^h \qquad j = 1, \ldots, N$$

Coarse space (defined later):

$$V_0 \ \subset V^h$$

Additive Schwarz preconditioner:

$$\mathbf{M}_{ASM,2}^{-1} \ = \ \sum_{j=0}^{N} \mathbf{R}_j^\top \mathbf{A}_j^{-1} \mathbf{R}_j$$

where $\mathbf{A}_j = \mathbf{R}_j^\top \mathbf{A} \mathbf{R}_j$
and $\mathbf{R}_j^\top \leftrightarrow R_j^\top : V_j \to V^h$ natural embedding

Partition of unity

Definitions:

$$\begin{aligned} dof(\Omega_j) &:= \{k : \mathrm{supp}(\phi_k) \cap \Omega_j \neq \emptyset\} \\ idof(\Omega_j) &:= \{k : \mathrm{supp}(\phi_k) \subset \overline{\Omega}_j\} \qquad V_j = \mathrm{span}\{\phi_k\}_{k \in idof(\Omega_j)} \\ imult(k) &:= \#\{j : k \in idof(\Omega_j)\} \end{aligned}$$

Partition of unity:

(used for design of coarse space and for stable splitting)

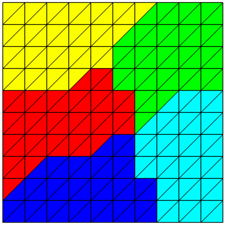
$$\Xi_j v = \sum_{k \in idof(\Omega_j)} \frac{1}{imult(k)} v_k \, \phi_k \qquad \text{for } v = \sum_{k=1}^{n} v_k \phi_k$$

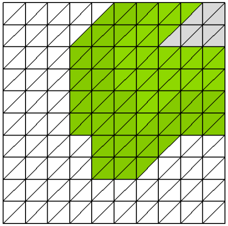
Properties:

$$\sum_{j=1}^{N} \Xi_j v = v \qquad \Xi_j v \in V_j$$

Overlapping zone / Choice of coarse space

Overlapping zone: $\Omega_j^\circ = \{x \in \Omega_j : \exists i \neq j : x \in \Omega_i\}$





Observation: $\Xi_{j|\Omega_j \setminus \Omega_j^\circ} = \mathrm{id}$

Coarse space should be **local**:

$$V_0 = \sum_{j=1}^{N} V_{0,j} \qquad \text{where } V_{0,j} \subset V_j$$

E.g. $V_{0,j} = \mathrm{span}\{\Xi_j p_{j,k}\}_{k=1}^{m_j}$

Abstract eigenvalue problem

Gen.EVP per subdomain:

Find $p_{j,k} \in V_{h|\Omega_j}$ and $\lambda_{j,k} \geq 0$:

$$a_{\Omega_j}(p_{j,k},\, v) \;=\; \lambda_{j,k}\, a_{\Omega_j^\circ}(\Xi_j p_{j,k},\, \Xi_j v) \qquad \forall v \in V_{h|\Omega_j}$$

$$\mathbf{A}_j \mathbf{p}_{j,k} \;=\; \lambda_{j,k}\, \mathbf{X}_j \mathbf{A}_j^\circ \mathbf{X}_j\, \mathbf{p}_{j,k} \qquad (\mathbf{X}_j \ldots \text{diagonal})$$

(properties of eigenfunctions discussed soon) $a_D \ldots$ restriction of a to D

In the two-level ASM:

Choose first m_j eigenvectors per subdomain:

$$V_0 \;=\; \operatorname{span}\{\Xi_j p_{j,k}\}_{k=1,\ldots,m_j}^{j=1,\ldots,N}$$

Theory

Two technical assumptions.

Theorem (D., Hauret, Nataf, Pechstein, Scheichl, Spillane)

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

$$\kappa(\mathbf{M}_{ASM,2}^{-1}\mathbf{A}) \leq (1+k_0)\left[2 + k_0\,(2k_0+1)\max_{j=1}^{N}\left(1+\frac{1}{\lambda_{j,m_j+1}}\right)\right]$$

Possible criterion for picking m_j: (used in our Numerics)

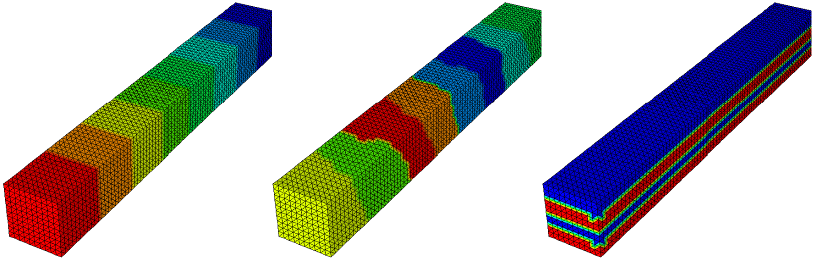
$$\lambda_{j,m_j+1} < \frac{\delta_j}{H_j}$$

$H_j \ldots$ subdomain diameter, $\delta_j \ldots$ overlap

Numerics – Darcy – I

Domain & Partitions

Coefficient



Iterations (CG) vs. jumps

Code: Matlab & FreeFem++

κ_2	ASM-1	ASM-2-low	$dim(V_H)$	GenEO	$dim(V_H)$
1	22	16	*(8)*	16	*(8)*
10^2	31	24	*(8)*	17	*(15)*
10^4	37	30	*(8)*	21	*(15)*
10^6	36	29	*(8)*	18	*(15)*

ASM-1: 1-level ASM ASM-2-low: $m_j = 1$ NEW: $\lambda_{j,m_j+1} < \delta_j/H_j$

Numerics – Darcy – II

Iterations (CG) vs. number of subdomains

regular partition

subd.	dofs	ASM-1	ASM-2-low	$dim(V_H)$	GenEO	$dim(V$
4	4840	14	15	*(4)*	10	*(6)*
8	9680	26	22	*(8)*	11	*(14)*
16	19360	51	36	*(16)*	13	*(30)*
32	38720	$>$ 100	61	*(32)*	13	*(62)*

METIS partition

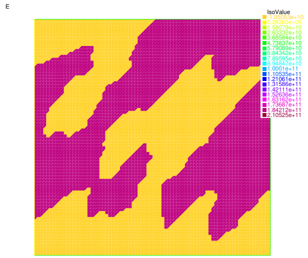
subd.	dofs	ASM-1	ASM-2-low	$dim(V_H)$	GenEO	$dim(V$
4	4840	21	18	*(4)*	15	*(7)*
8	9680	36	29	*(8)*	18	*(15)*
16	19360	65	45	*(16)*	22	*(31)*
32	38720	$>$100	79	*(32)*	34	*(63)*

Numerics – Darcy – III

Iterations (CG) vs. overlap

(added) layers	ASM-1	ASM-2-low	*(V_H)*	GenEO	*(V_H)*
1	26	22	*(8)*	11	*(14)*
2	22	18	*(8)*	9	*(14)*
3	16	15	*(8)*	9	*(14)*

Numerics – 2D Elasticity



$$E_1 = 2 \cdot 10^{11}$$
$$\nu_1 = 0.3$$

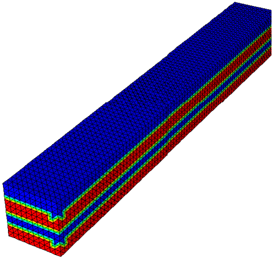
$$E_2 = 2 \cdot 10^{7}$$
$$\nu_2 = 0.45$$

METIS partitions with 2 layers added

subd.	dofs	ASM-1	ASM-2-low	*(V_H)*	GenEO	*(V_H)*
4	13122	93	134	*(12)*	42	*(42)*
16	13122	164	165	*(48)*	45	*(159)*
25	13122	211	229	*(75)*	47	*(238)*
64	13122	279	167	*(192)*	45	*(519)*

Numerics – 3D Elasticity

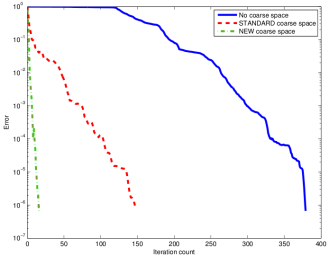
Iterations (CG) vs. number of subdomains



$$E_1 = 2 \cdot 10^{11}$$
$$\nu_1 = 0.3$$

$$E_2 = 2 \cdot 10^{7}$$
$$\nu_2 = 0.45$$

Relative error vs. iterations
16 regular subdomains



subd.	dofs	ASM-1	ASM-2-low	(V_H)	GenEO	(V_H)
4	1452	79	54	*(24)*	16	*(46)*
8	29040	177	87	*(48)*	16	*(102)*
16	58080	378	145	*(96)*	16	*(214)*