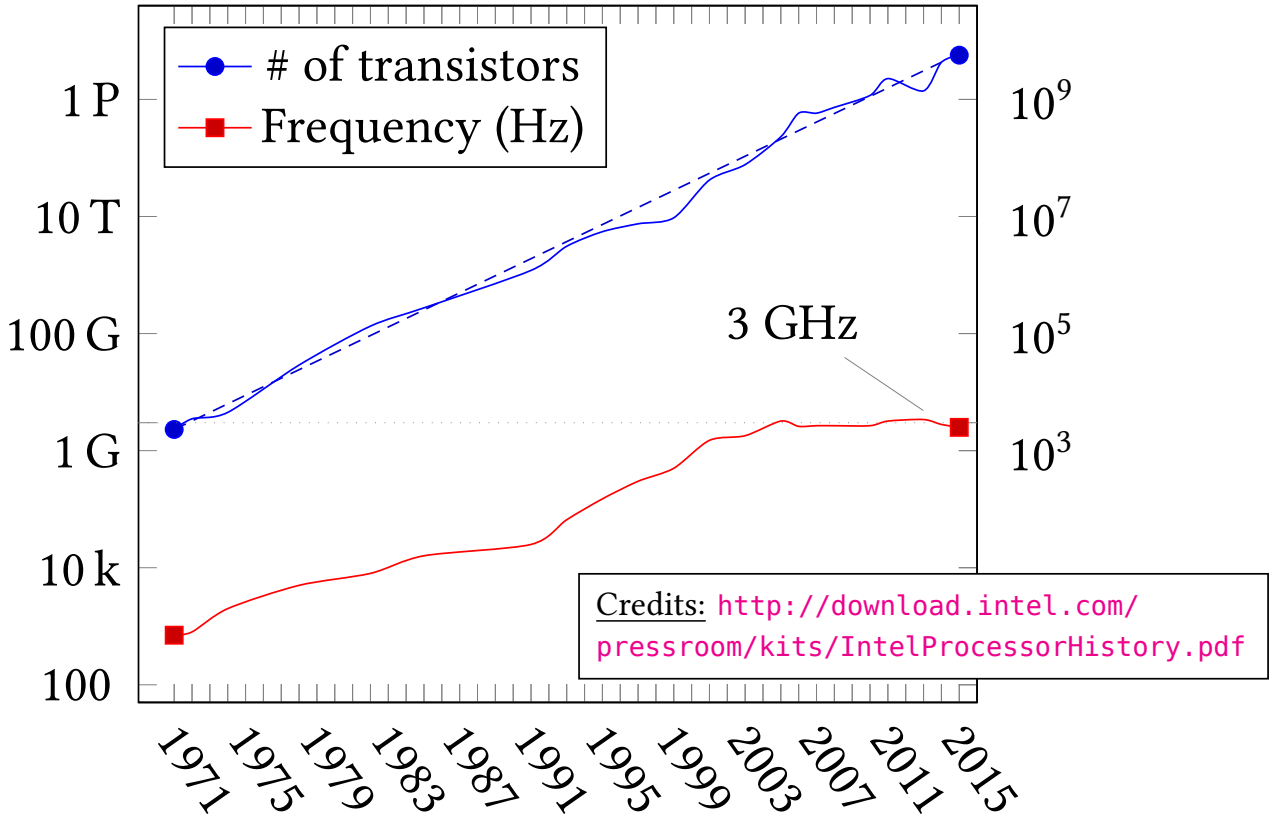
FreeFem++ and HPDDM

Groupe calcul – 2017

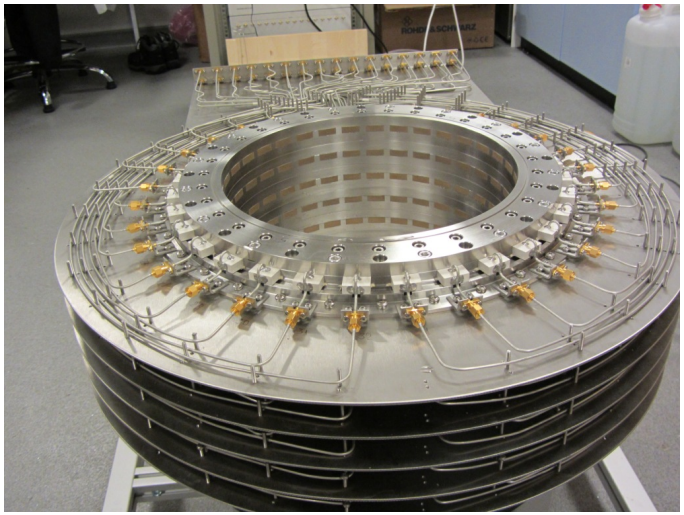
Need for massively parallel computing



Since year 2004 :

- CPU frequency stalls at 3 GHz due to the heat dissipation wall. The only way to improve the performance of computer is to go parallel

Parallel Software tools : HPDDM and FreeFem++



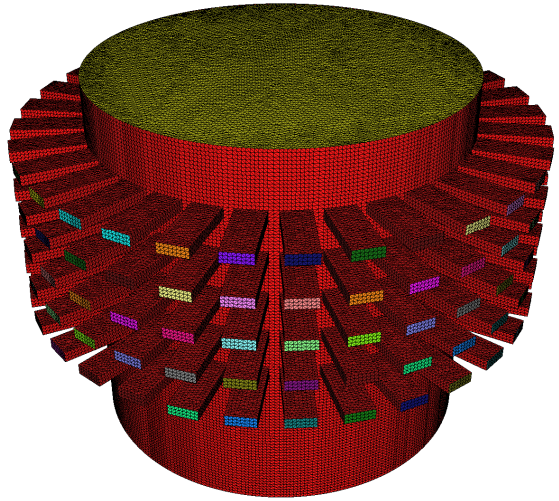


FIGURE – Antennas and mesh – interior diameter 28,5 cm

Two in-house open source libraries (LGPL) linked to many third-party libraries :

- HPDDM (High Performance Domain Decomposition Methods) for massively parallel computing
- FreeFem++(-mpi) for the parallel simulation of equations from physics by the finite element method (FEM).

$Au = f$? Panorama of parallel linear solvers

Multi-frontal sparse direct solver (I. Duff et al.)

MUMPS (J.Y. L'Excellent), SuperLU (Demmel, . . .), PastiX, UMFPACK, PARDISO (O. Schenk),

Iterative Methods

- Fixed point iteration : Jacobi, Gauss-Seidel, SSOR
- Krylov type methods : Conjuguate Gradient (Stiefel-Hestenes), GMRES (Y. Saad), QMR (R. Freund), MinRes, BiCGSTAB (van der Vorst)

"Hybrid Methods"

- Multigrid (A. Brandt, Ruge-Stüben, Falgout, McCormick, A. Ruhe, Y. Notay, . . .)
- Domain decomposition methods (O. Widlund, C. Farhat, J. Mandel, P.L. Lions,) are a naturally parallel compromise

Why iterative solvers ?

Limitations of direct solvers

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

- **two-dimensional problems** (10^6 unknowns)
- **three-dimensional problems** (10^5 unknowns).

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

To summarize :

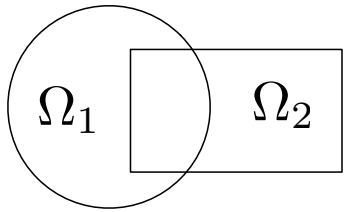
- below a certain size, direct solvers are chosen.
- beyond the critical size, iterative solvers are needed.

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)	Some black box routines Some 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.

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

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

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

Parallel algorithm, converges but very slowly, overlapping subdomains only.

The parallel version is called **Jacobi Schwarz method (JSM)**.

Consider the discretized Poisson problem : $Au = f \in \mathbb{R}^n$.

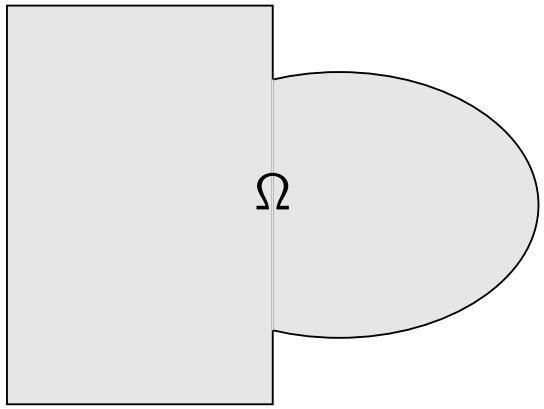
Given a decomposition of $[\![1; n]\!]$, $(\mathcal{N}_1, \mathcal{N}_2)$, define :

- the restriction operator R_i from $\mathbb{R}^{[\![1;n]\!]}$ into $\mathbb{R}^{\mathcal{N}_i}$,
- R_i^T as the extension by 0 from $\mathbb{R}^{\mathcal{N}_i}$ into $\mathbb{R}^{[\![1;n]\!]}$.

$u^m \longrightarrow u^{m+1}$ by solving concurrently :

$$u_1^{m+1} = u_1^m + A_1^{-1} R_1 (f - Au^m) \qquad u_2^{m+1} = u_2^m + A_2^{-1} R_2 (f - Au^m)$$

where $u_i^m = R_i u^m$ and $A_i := R_i A R_i^T$.



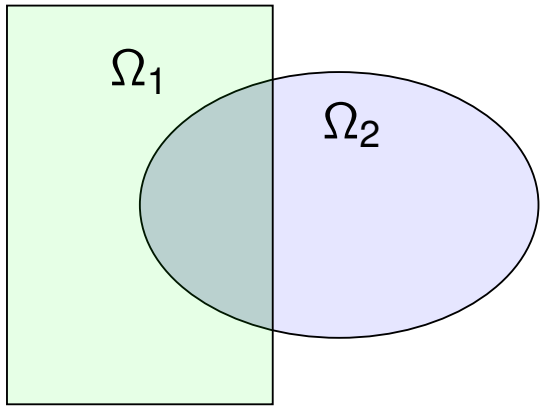
Consider the discretized Poisson problem : $Au = f \in \mathbb{R}^n$.
Given a decomposition of $[\![1; n]\!]$, $(\mathcal{N}_1, \mathcal{N}_2)$, define :

- the restriction operator R_i from $\mathbb{R}^{[\![1;n]\!]}$ into $\mathbb{R}^{\mathcal{N}_i}$,
- R_i^T as the extension by 0 from $\mathbb{R}^{\mathcal{N}_i}$ into $\mathbb{R}^{[\![1;n]\!]}$.

$u^m \longrightarrow u^{m+1}$ by solving concurrently :

$$u_1^{m+1} = u_1^m + A_1^{-1} R_1 (f - Au^m) \qquad u_2^{m+1} = u_2^m + A_2^{-1} R_2 (f - Au^m)$$

where $u_i^m = R_i u^m$ and $A_i := R_i A R_i^T$.



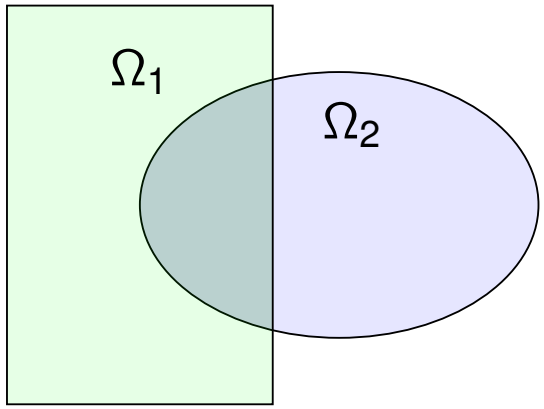
Consider the discretized Poisson problem : $Au = f \in \mathbb{R}^n$.
Given a decomposition of $[\![1; n]\!]$, $(\mathcal{N}_1, \mathcal{N}_2)$, define :

- the restriction operator R_i from $\mathbb{R}^{[\![1;n]\!]}$ into $\mathbb{R}^{\mathcal{N}_i}$,
- R_i^T as the extension by 0 from $\mathbb{R}^{\mathcal{N}_i}$ into $\mathbb{R}^{[\![1;n]\!]}$.

$u^m \longrightarrow u^{m+1}$ by solving concurrently :

$$u_1^{m+1} = u_1^m + A_1^{-1} R_1 (f - Au^m) \qquad u_2^{m+1} = u_2^m + A_2^{-1} R_2 (f - Au^m)$$

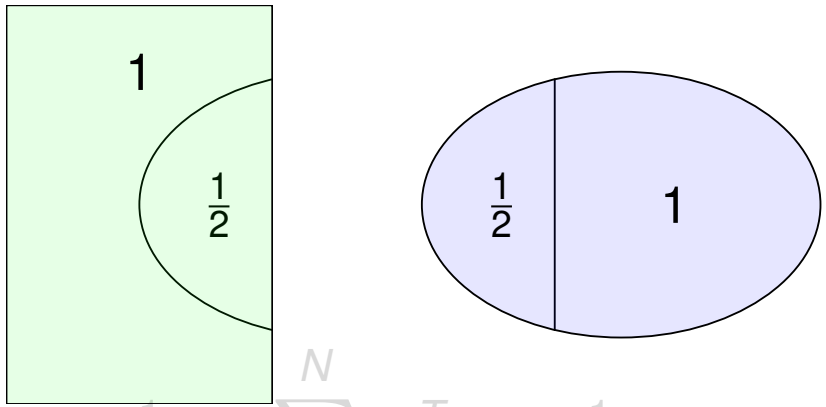
where $u_i^m = R_i u^m$ and $A_i := R_i A R_i^T$.



We have effectively divided, but we have yet to conquer.

Duplicated unknowns coupled via a *partition of unity* :

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



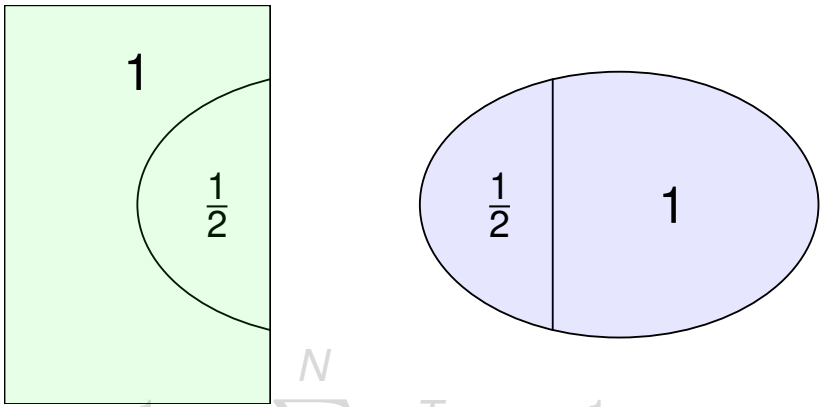
Then, $u^{m+1} = \sum_{i=1}^{N} R_i^T D_i u_i^{m+1}$. $M_{RAS}^{-1} = \sum_{i=1}^{N} R_i^T D_i A_i^{-1} R_i$.

RAS algorithm (Cai & Sarkis, 1999). Weighted Overlapping Block Jacobi method

We have effectively divided, but we have yet to conquer.

Duplicated unknowns coupled via a *partition of unity* :

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



Then, $u^{m+1} = \sum_{i=1}^{N} R_i^T D_i u_i^{m+1}$.

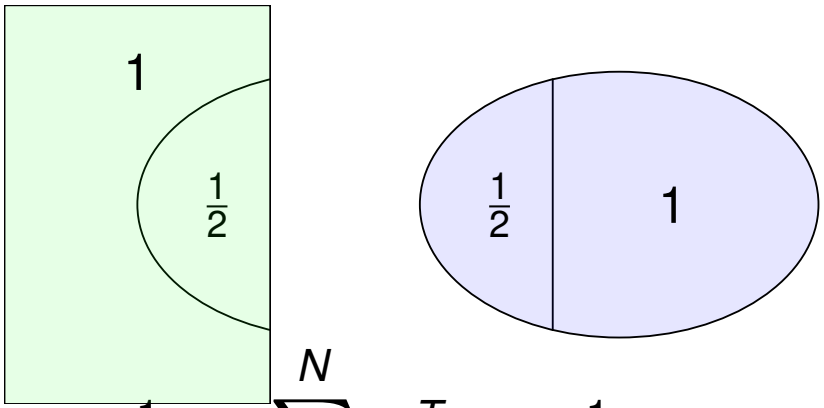
$$M_{RAS}^{-1} = \sum_{i=1}^{N} R_i^T D_i A_i^{-1} R_i.$$

RAS algorithm (Cai & Sarkis, 1999). Weighted Overlapping Block Jacobi method

We have effectively divided, but we have yet to conquer.

Duplicated unknowns coupled via a *partition of unity* :

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



Then, $u^{m+1} = \sum_{i=1}^{N} R_i^T D_i u_i^{m+1}$.

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

RAS algorithm (Cai & Sarkis, 1999). Weighted Overlapping Block Jacobi method

Algebraic formulation - RAS and ASM

Discrete Schwarz algorithm iterates on a pair of local functions (u_m^1, u_m^2)
RAS algorithm iterates on the global function u^m

Schwarz and RAS

Discretization of the classical Schwarz algorithm and the iterative RAS algorithm :

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

are equivalent

$$U^n = R_1^T D_1 U_1^n + R_2^T D_2 U_2^n \, .$$

(Efstathiou and Gander, 2002).

Operator M_{RAS}^{-1} is used as a preconditioner in Krylov methods for non symmetric problems.

Algebraic formulation - RAS and ASM

Discrete Schwarz algorithm iterates on a pair of local functions (u_m^1, u_m^2)
RAS algorithm iterates on the global function u^m

Schwarz and RAS

Discretization of the classical Schwarz algorithm and the iterative RAS algorithm :

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

are equivalent

$$U^n = R_1^T D_1 U_1^n + R_2^T D_2 U_2^n \, .$$

(Efstathiou and Gander, 2002).

Operator M_{RAS}^{-1} is used as a preconditioner in Krylov methods for non symmetric problems.

Many cores : Strong and Weak scalability

How to evaluate the efficiency of a domain decomposition ?

Strong scalability (Amdahl)

"How the solution time varies with the number of processors for a fixed *total* problem size"

Weak scalability (Gustafson)

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

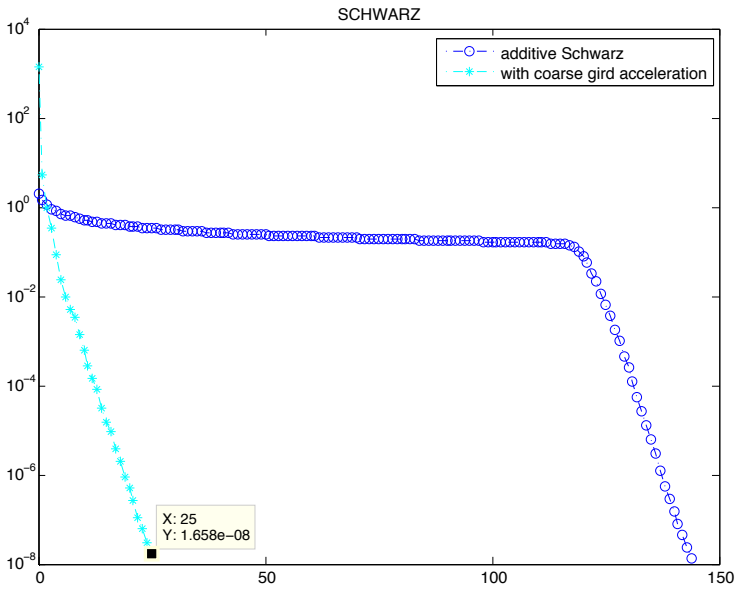
Not achieved with the one level method

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.



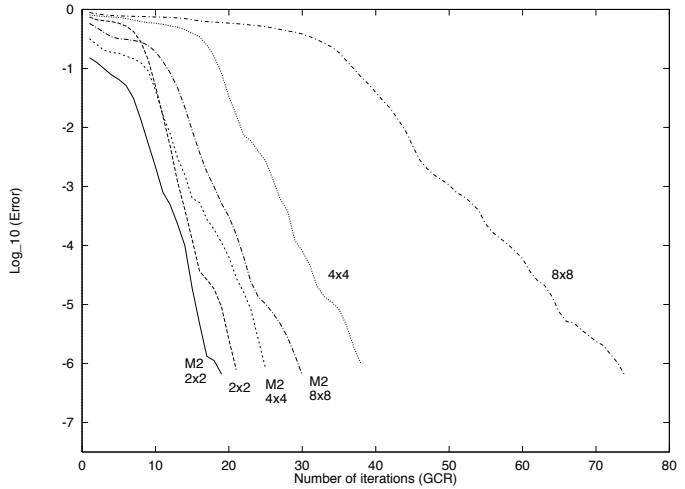


FIGURE – Decomposition into 64 subdomains and into $m \times m$ squares

Solution of a Poisson problem $-\Delta u = f$

Number of subdomains	2x2	4x4	8x8
Number of iterations	20	36	64

Adding a coarse space

One level methods are not scalable for steady state problems.

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

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

The Nicolaides approach (1987) is to use the kernel of the operator as a coarse space, this is the constant vectors, in local form this writes :

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

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, Dryija)

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

Fails for highly heterogeneous problems

You need a larger and adaptive coarse space, see later

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 space must be

- Local (calculated on each subdomain) $\rightarrow$ parallel
- Adaptive (calculated automatically)
- Easy and cheap to compute
- Robust (must lead to an algorithm whose convergence is proven not to depend on the partition nor the jumps in coefficients)

Adaptive Coarse space for highly heterogeneous Darcy and (compressible) elasticity problems :
Geneo .EVP per subdomain :

Find $V_{j,k} \in \mathbb{R}^{\mathcal{N}_j}$ and $\lambda_{j,k} \geq 0$:

$$D_j R_j A R_j^T D_j V_{j,k} = \lambda_{j,k} A_j^{Neu} V_{j,k}$$

In the two-level ASM, let τ be a user chosen parameter :
Choose eigenvectors $\lambda_{j,k} \geq \tau$ per subdomain :

$$Z := \left(R_j^T D_j V_{j,k}\right)_{\lambda_{j,k} \geq \tau}^{j=1,\ldots,N}$$

This automatically includes Nicolaides CS made of Zero Energy Modes.

Adaptive Coarse space for highly heterogeneous Darcy and (compressible) elasticity problems :
Geneo .EVP per subdomain :

Find $V_{j,k} \in \mathbb{R}^{\mathcal{N}_j}$ and $\lambda_{j,k} \geq 0$:

$$D_j R_j A R_j^T D_j V_{j,k} = \lambda_{j,k} A_j^{Neu} V_{j,k}$$

In the two-level ASM, let τ be a user chosen parameter :
Choose eigenvectors $\lambda_{j,k} \geq \tau$ per subdomain :

$$Z := \left(R_j^T D_j V_{j,k}\right)_{\lambda_{j,k} \geq \tau}^{j=1,\ldots,N}$$

This automatically includes Nicolaides CS made of Zero Energy Modes.

Two technical assumptions.

Theorem (Spillane, Dolean, Hauret, N., Pechstein, Scheichl (Num. Math. 2013))

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

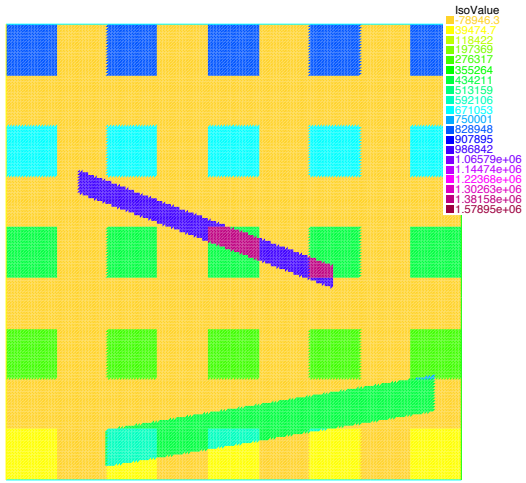
$$\kappa(M_{ASM,2}^{-1}A) \leq (1+k_0)\left[2+k_0(2k_0+1)\left(1+\tau\right)\right]$$

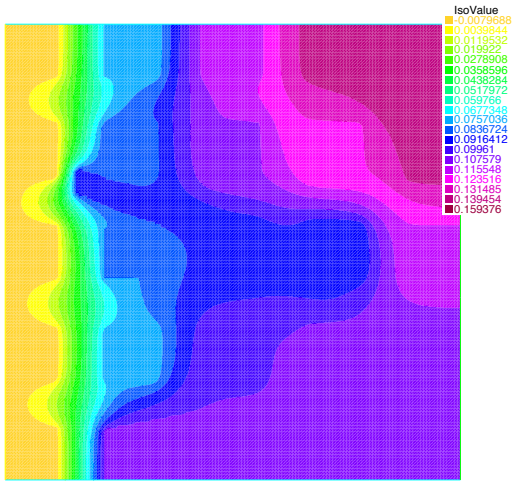
Possible criterion for picking τ : (used in our Numerics)

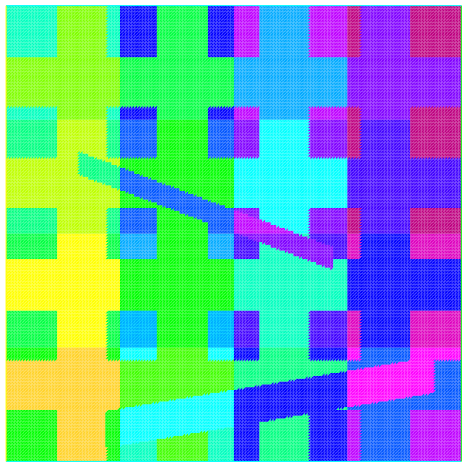
$$\tau := \min_{j=1,\ldots,N} \frac{H_j}{\delta_j}$$

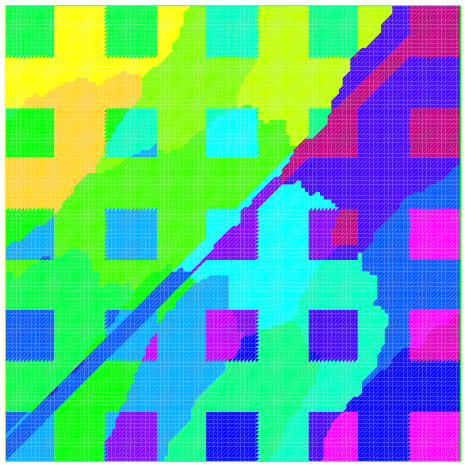
H_j . . . subdomain diameter, δ_j . . . overlap

Numerical results (Darcy)



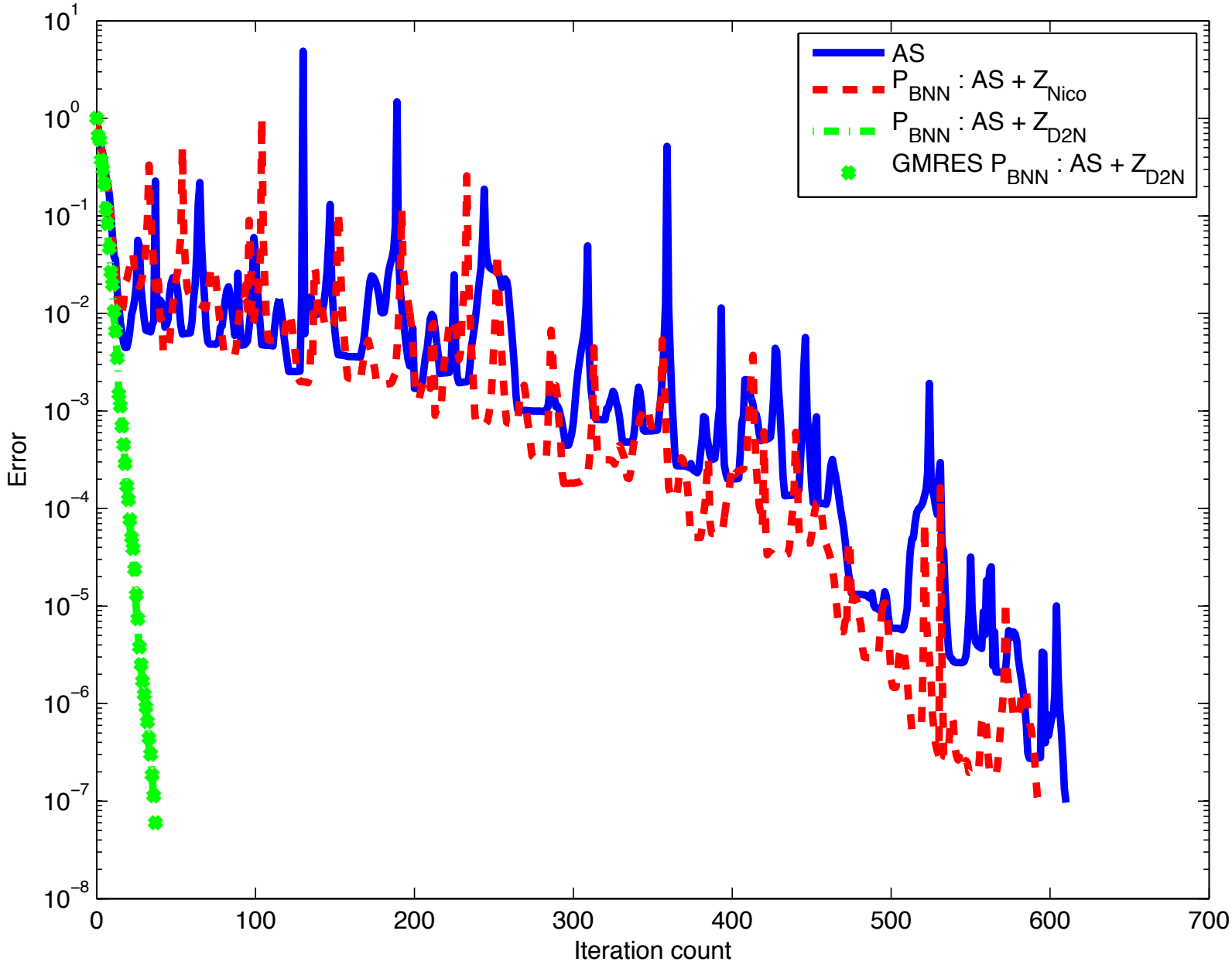






Channels and inclusions : $1 \leq \alpha \leq 1.5 \times 10^6$, the solution and partitionings (Metis or not)

Convergence



HPDDM Library (P. Jolivet and N.)

An implementation of several Domain Decomposition Methods and Multiple RHS solver

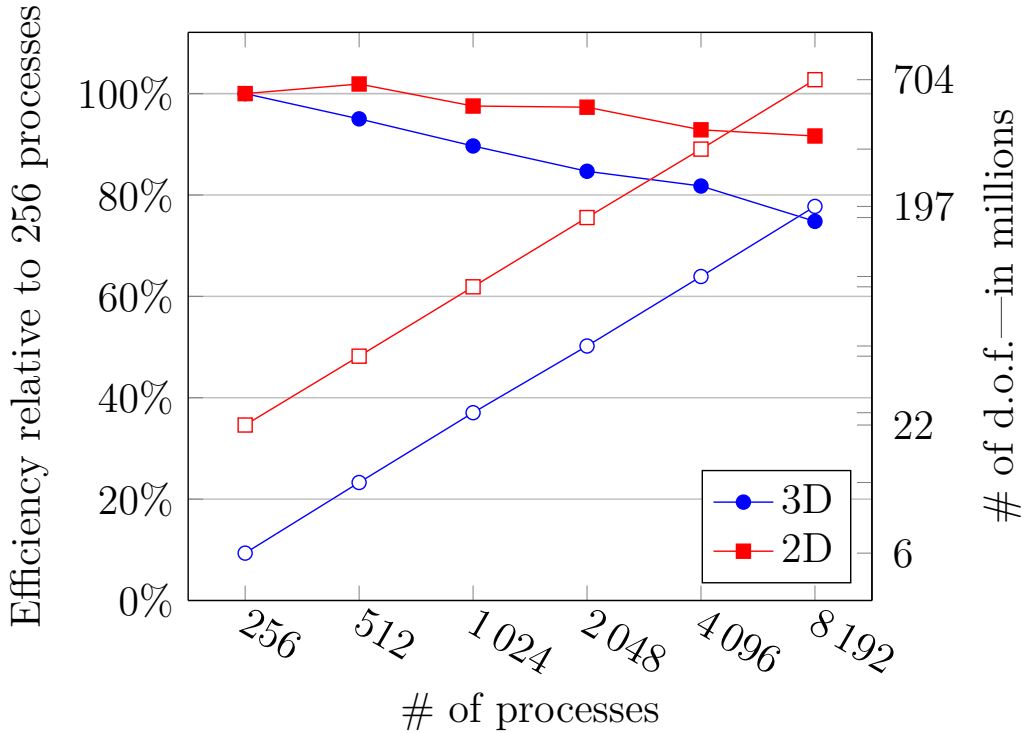
- One-and two-level Schwarz methods
- The Finite Element Tearing and Interconnecting (FETI) method
- Balancing Domain Decomposition (BDD) method
- Implements parallel algorithms : Domain Decomposition methods and Block solvers
- 2 billions unknowns in three dimension solved in 210 seconds on 8100 cores

Library

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

Weak scalability for heterogeneous elasticity (with FreeFem++ and HPDDM)

Rubber Steel sandwich with automatic mesh partition



(a) Timings of various simulations

200 millions unknowns in 3D wall-clock time : 200. sec.
IBM/Blue Gene Q machine with 1.6 GHz Power A2 processors. Hours provided by an IDRIS-GENCI project.

Strong scalability in two and three dimensions (with FreeFem++ and HPDDM)

Stokes problem with automatic mesh partition. Driven cavity problem

	N	Factorization	Deflation	Solution	# of it.	Total	# of d.o.f.
3D	1 024	79.2 s	229.0 s	76.3 s	45	387.5 s	$50.63 \cdot 10^6$
	2 048	29.5 s	76.5 s	34.8 s	42	143.9 s	
	4 096	11.1 s	45.8 s	19.8 s	42	80.9 s	
	8 192	4.7 s	26.1 s	14.9 s	41	56.8 s	

Peak performance : 50 millions d.o.f's in 3D in 57 sec.

IBM/Blue Gene Q machine with 1.6 GHz Power A2 processors. Hours provided by an IDRIS-GENCI project.

HPDDM `https://github.com/hpddm/hpddm` is a framework in C++/MPI for high-performance domain decomposition methods with a Plain Old Data (POD) interface

Maxwell in the frequency domain

- Mesh with 2.3M degrees of freedom ;
- Domain decomposition methods with impedance interface conditions, twice as fast as Dirichlet interface conditions ;
- Parallel computing on 64 cores on SGI UV2000 at UPMC : 3s per emitter, 5 mn as a whole.