# Algorithmes parallèles et méthodes de décomposition de domaine

## Méthodes numériques avancées en mécanique

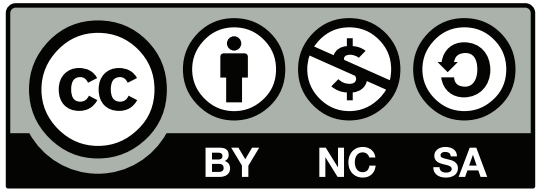
Christophe Bovet

christophe.bovet@onera.fr

03/03/2021

ONERA
THE FRENCH AEROSPACE LAB

Cours téléchargeable ici :
https://christophe-bovet.fr/enseignements/




Cette œuvre est mise à disposition sous licence Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions 3.0 France.
Pour voir une copie de cette licence, visitez http://creativecommons.org/licenses/by-nc-sa/3.0/fr/ ou écrivez à Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

# Sommaire

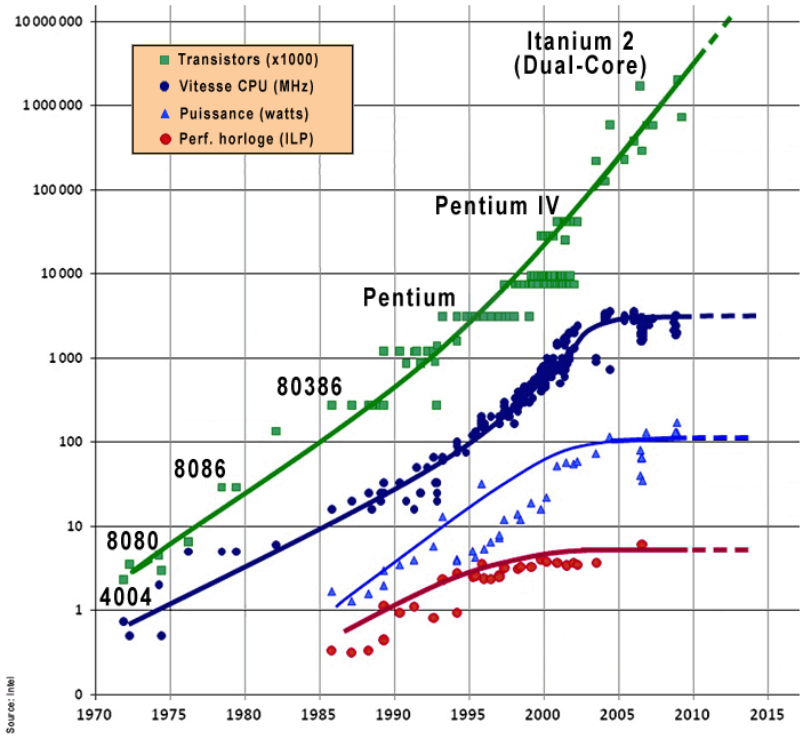
1 Introduction
- Informatique et parallélisme
- Problème type du mécanicien
- Cadre et objectifs du cours

2 Rappels sur la résolution de systèmes linéaires

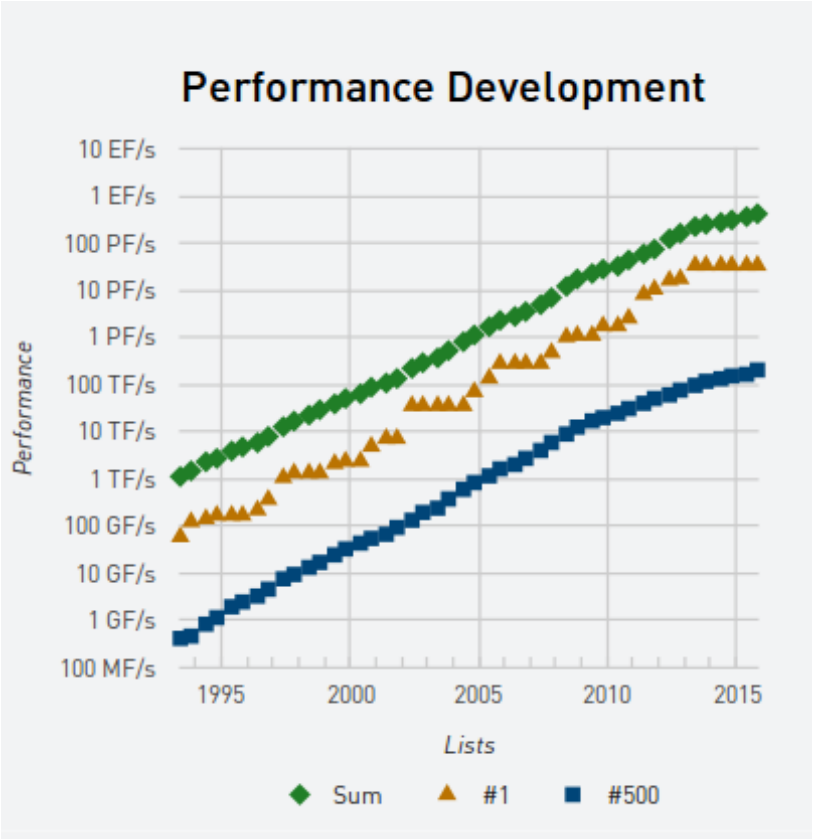
3 Méthodes de type Schur

4 Problèmes ouverts et connexes

# Architecture parallèle et loi de Moore




(www.astrosurf.com/luxorion/loi-moore.html)

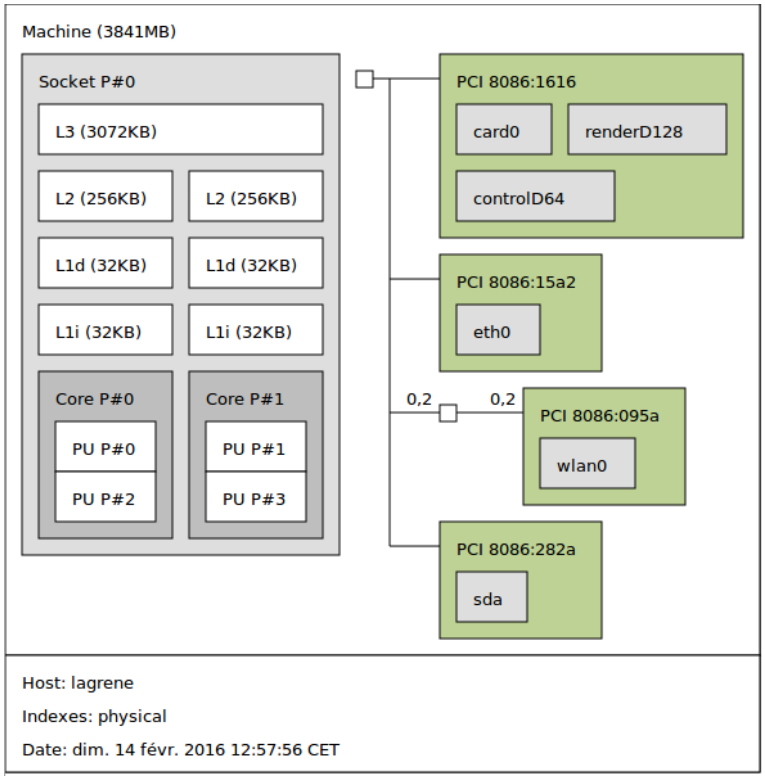



(www.top500.org)

- 1 Flop/s = 1 *floating-point operation per second*
- TOP 500 #1 : Sunway TaihuLight, 10 649 600 cœurs, **1.45GHz** 125 000 TFlops/s

# Architecture d'un processeur grand public

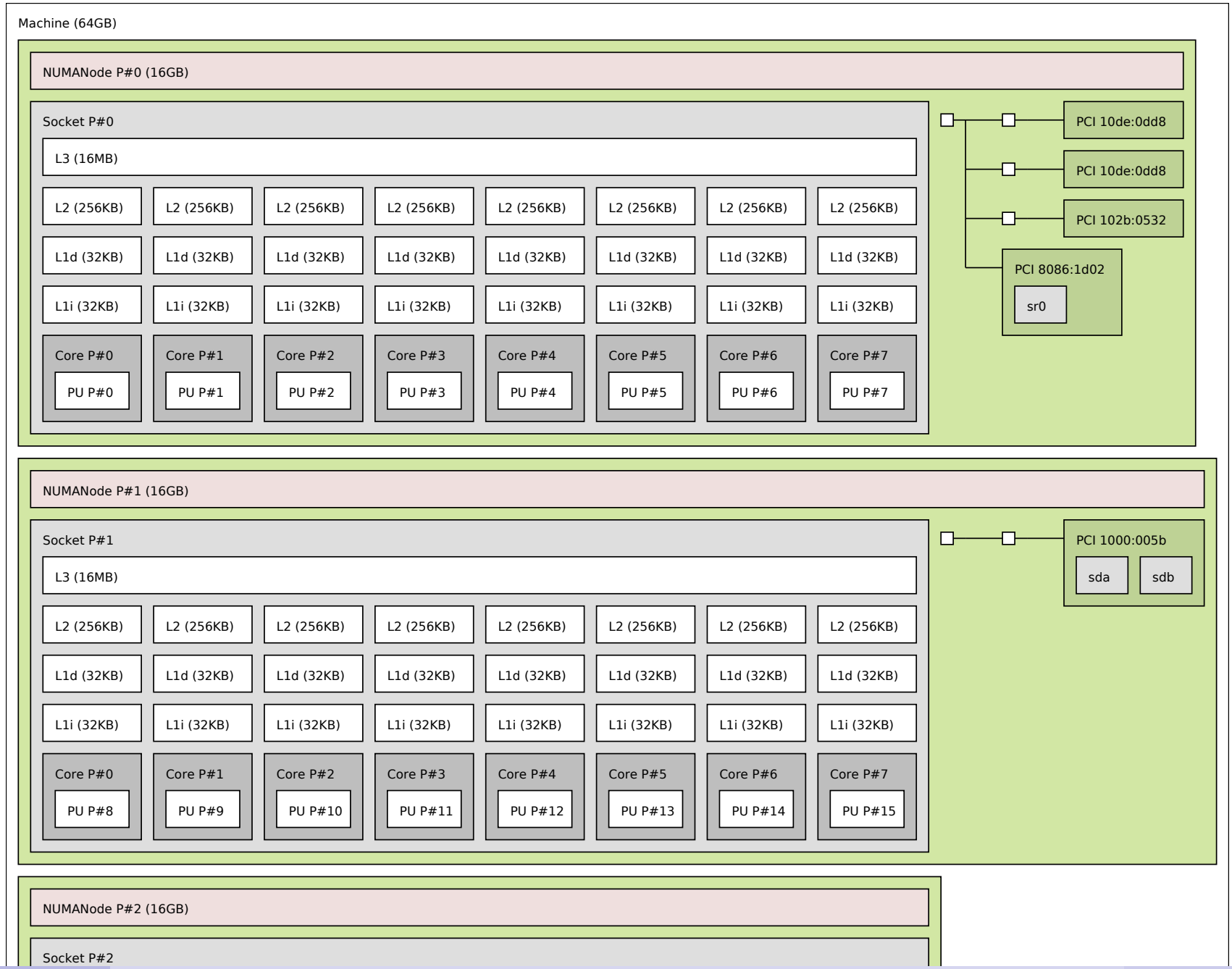
- **Unités de calculs** :
  coeur / processeur / noeud
- **Mémoire** :
  cache L1/L2/L3, RAM, disque dur
- **Communication** :
  bus / réseau (gigabit, InfiniBand)




- Latence (Core i7 Xeon 5500)

| | |
|---|---|
| L1 CACHE hit, | ∼4 cycles (2.1 - 1.2 ns) |
| L2 CACHE hit, | ∼10 cycles (5.3 - 3.0 ns) |
| L3 CACHE hit, line unshared | ∼40 cycles (21.4 - 12.0 ns) |
| L3 CACHE hit, shared line in another core | ∼65 cycles (34.8 - 19.5 ns) |
| L3 CACHE hit, modified in another core | ∼75 cycles (40.2 - 22.5 ns) |
| local DRAM | ∼60 ns |
| remote DRAM | ∼100 ns |

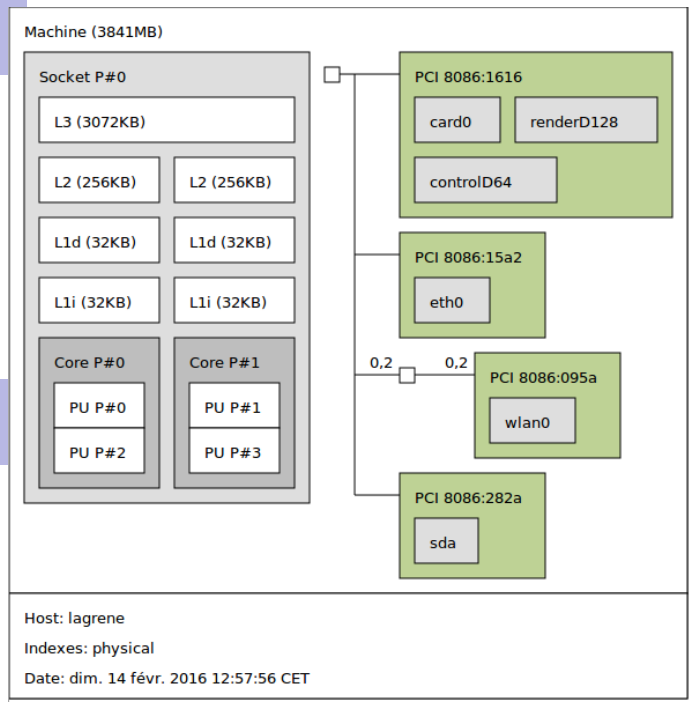
# Architecture d'un proc de calcul (Xeon E5-4620 32 cores, 64 Go)



# Différents types de parallélisme

## Mémoire partagée (symmetric multiprocessing SMP)

Échange rapide (accès direct)
Besoin de protéger les données (mutex/semaphore)
⇒ multithreading, protocole OpenMP




## Mémoire séparée

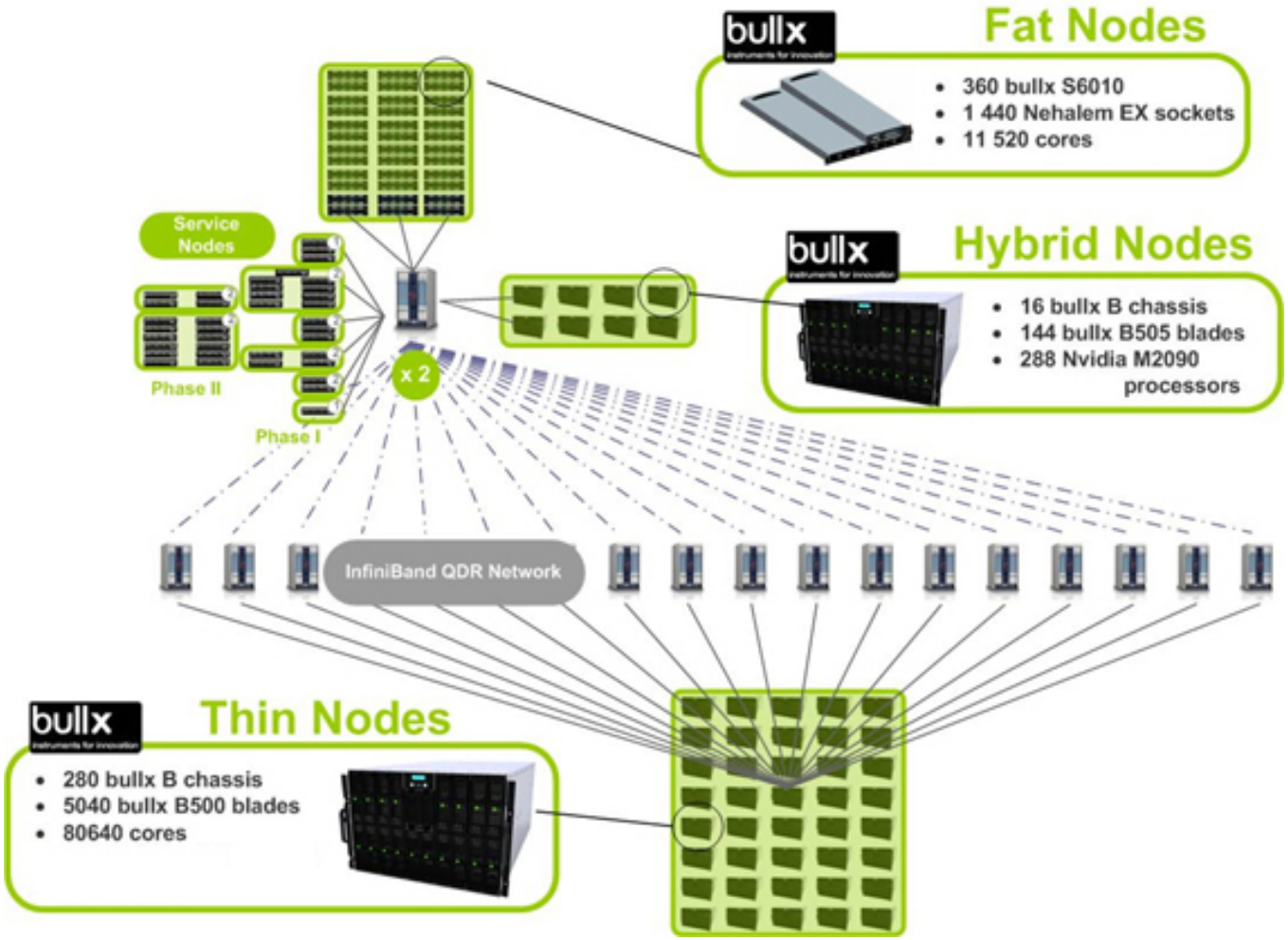
Données protégées
Il faut s'occuper d'échanger les données.
⇒ Protocole MPI
(message passing interface)

## Architectures actuelles

Architectures mixtes plusieurs noeuds en réseau avec plusieurs coeurs à mémoire partagée.

# Présentation d'un super calculateur

## Le calculateur Curie du TGCC (version 2016)

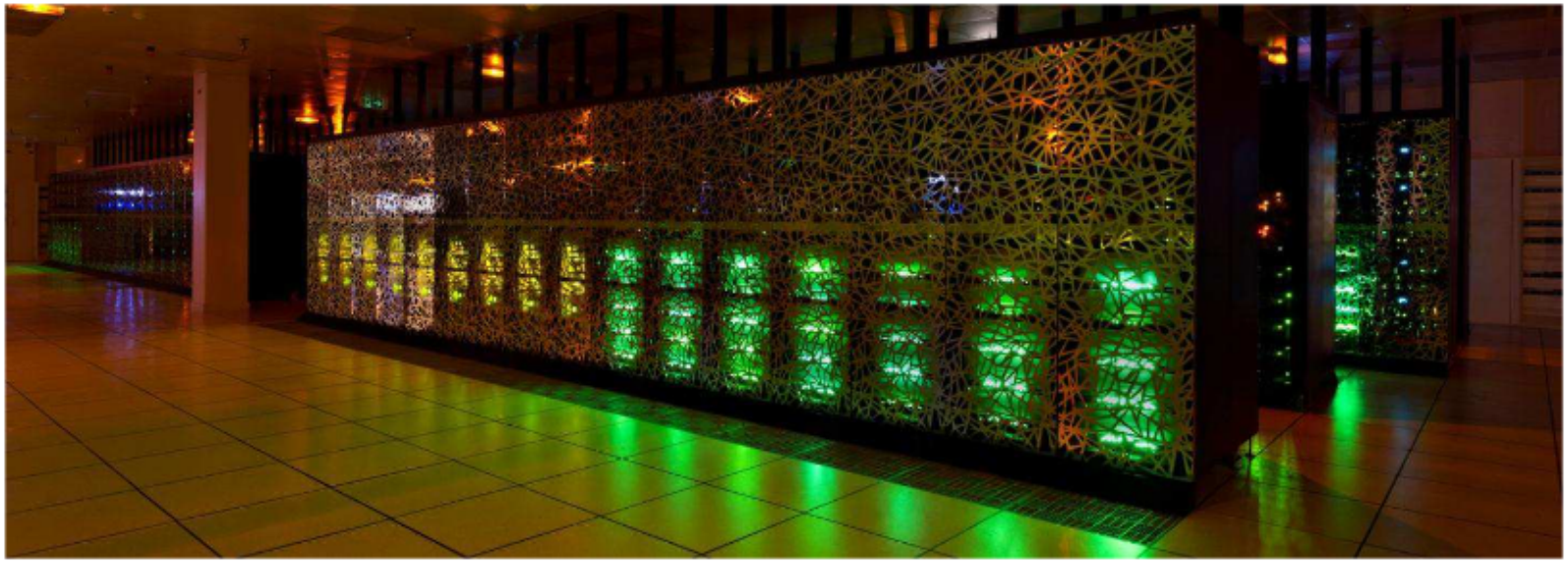



Source : http ://www-hpc.cea.fr/fr/complexe/tgcc-curie.htm

# Présentation d'un super calculateur

## Le calculateur Curie du TGCC (version 2016)

| Fat nodes | Thin nodes |
|---|---|
| 360 BullX-S6010 | 5040 BullX B500 |
| Intel NH EX 2,26 Ghz | Intel Sandy Bridge 2,7Ghz |
| 11 520 coeurs | 80 640 coeurs |
| 128 coeurs/noeud | 16 coeurs/noeuds |
| 512 Go/noeud | 4 Go/coeurs |
| 105 TFlop/s | 1 740 TFlop/s |



Source : http ://www-hpc.cea.fr/fr/complexe/tgcc-curie.htm

# Problème type du mécanicien

Résoudre un problème non linéaire d'évolution à l'aide d'une méthode de discrétisation. Par exemple pour la méthode des éléments finis :

- Équations d'équilibre (EDP, EDO)
- Formulation variationnelle
- Discrétisation éléments finis

Si le problème est non linéaire, un algorithme incrémental de type Newton permet de se ramener à une succession de problèmes linéaires

**Pour traiter efficacement ces problèmes il faut exploiter ces architectures parallèles**

# Spécificité de nos problèmes de mécaniciens

Pour résoudre un problème mécanique il y a plusieurs phases qui peuvent être plus ou moins facilement parallélisables

- Génération du maillage
- Création d'opérateur
- Résolution de problèmes non linéaires
- Résolution de systèmes linéaires
- Écriture des résultats
- Visualisation des résultats

Certaines sont *embarrassingly parallel*[1] d'autres beaucoup moins...

1. In parallel computing, an embarrassingly parallel workload or problem (also called perfectly parallel or pleasingly parallel) is one where little or no effort is needed to separate the problem into a number of parallel tasks.This is often the case where there is little or no dependency or need for communication between those parallel tasks, or for results between them. (source Wikipedia)

# Spécificité de nos problèmes de mécaniciens

Pour résoudre un problème mécanique il y a plusieurs phases qui peuvent être plus ou moins facilement parallélisables

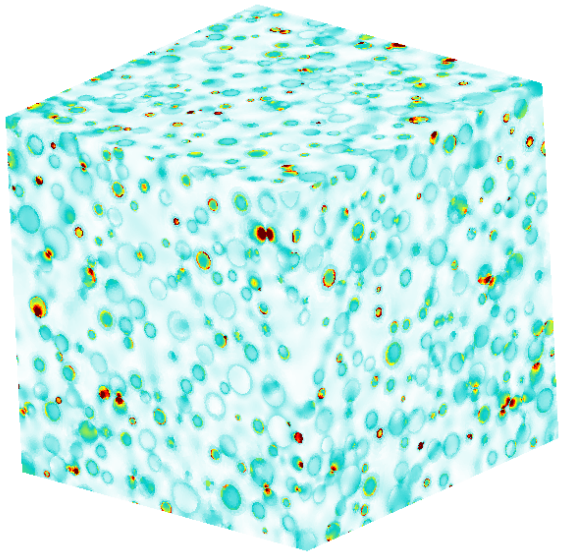
- Génération du maillage
- Création d'opérateur
- Résolution de problèmes non linéaires
- Résolution de systèmes linéaires
- Écriture des résultats
- Visualisation des résultats

Certaines sont *embarrassingly parallel*[1] d'autres beaucoup moins...

1. In parallel computing, an embarrassingly parallel workload or problem (also called perfectly parallel or pleasingly parallel) is one where little or no effort is needed to separate the problem into a number of parallel tasks.This is often the case where there is little or no dependency or need for communication between those parallel tasks, or for results between them. (source Wikipedia)

# Spécificité de nos problèmes de mécaniciens

- Complexité algorithmique d'une factorisation LU $O\left(\frac{2}{3}n^3\right)$



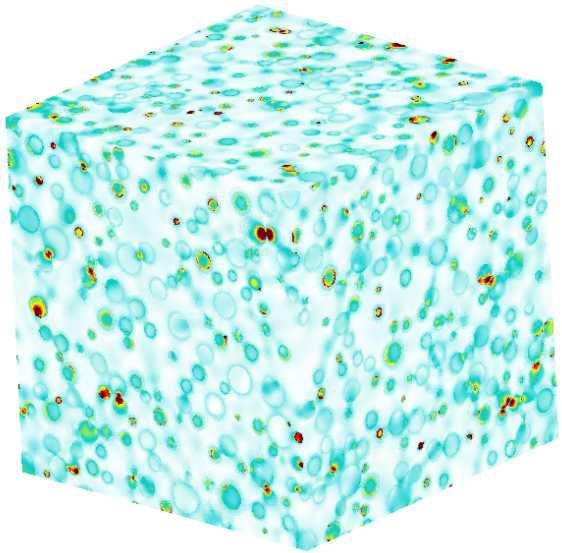
$\geq 10^8$ *ddls*

| | Flop/s de l'ordinateur | | |
|---|---|---|---|
| n | $10^9$ | $10^{12}$ | $10^{15}$ |
| $10^4$ | 10 min | 1 sec | 1 $\mu s$ |
| $10^6$ | 20 ans | 7 mois | 10 min |
| $10^8$ | ... | ... | 20 ans |

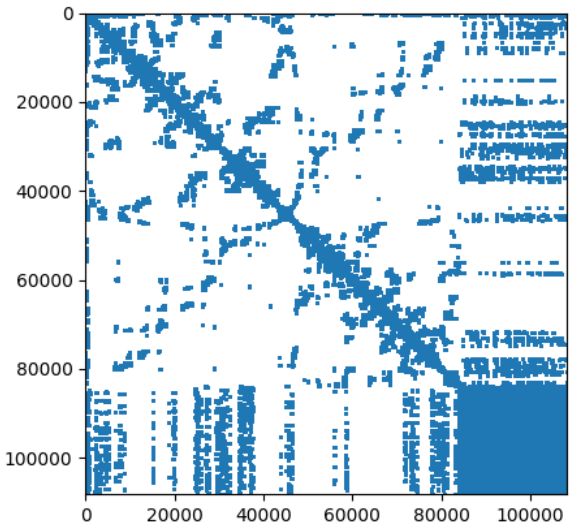
- Il faut exploiter les spécificités du problème à résoudre
- Sans propriété favorable la parallélisation est difficile, utilisation de techniques informatiques pures (BLAS, ScaLAPACK).
- En mécanique (et beaucoup de physiques), la méthode de discrétisation conduit à des matrices creuses

# Spécificité de nos problèmes de mécaniciens

- Complexité algorithmique d'une factorisation LU $O\left(\frac{2}{3}n^3\right)$ (en dense)



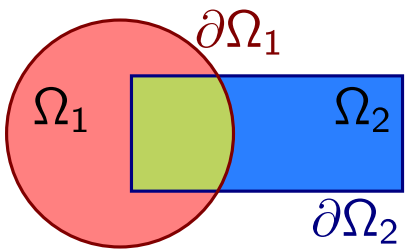
$\geq 10^8$ *ddls*




- Il faut exploiter les spécificités du problème à résoudre
- Sans propriété favorable la parallélisation est difficile, utilisation de techniques informatiques pures (BLAS, ScaLAPACK).
- En mécanique (et beaucoup de physiques), la méthode de discrétisation conduit à des matrices creuses

- Résoudre le problème

$$-\Delta(u) = f \text{ dans } \Omega$$
$$u = 0 \text{ sur } \partial\Omega$$




- Méthode de Schwarz alternée : $(u_1^n, u_2^n) \to (u_1^{n+1}, u_2^{n+1})$

$$-\Delta(u_1^{n+1}) = f \text{ dans } \Omega_1 \qquad -\Delta(u_2^{n+1}) = f \text{ dans } \Omega_2$$
$$u_1^{n+1} = 0 \text{ sur } \partial\Omega_1 \cap \partial\Omega \quad \text{puis} \quad u_2^{n+1} = 0 \text{ sur } \partial\Omega_2 \cap \partial\Omega$$
$$u_1^{n+1} = u_2^n \text{ sur } \partial\Omega_1 \cap \overline{\Omega}_2 \qquad u_2^{n+1} = u_1^{n+1} \text{ sur } \partial\Omega_2 \cap \overline{\Omega}_1$$

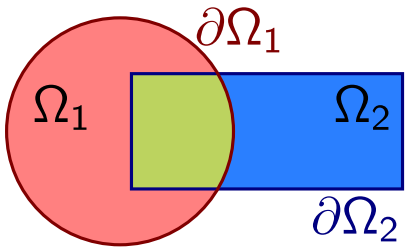
- Algorithme séquentiel avec ou sans recouvrement
- En général, une grande zone de recouvrement accélère la convergence du schéma itératif

# Un peu d'histoire

## Méthode de Schwarz parallèle (P.L. Lions, 1988)

- Résoudre le problème

$$-\Delta(u) = f \text{ dans } \Omega$$
$$u = 0 \text{ sur } \partial\Omega$$



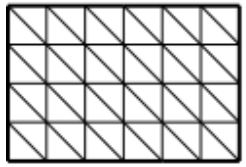

- Méthode de Schwarz parallèle : $(u_1^n, u_2^n) \to (u_1^{n+1}, u_2^{n+1})$

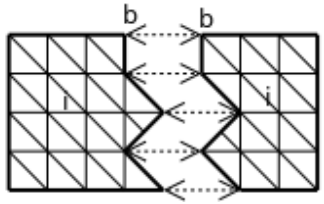
$$-\Delta(u_1^{n+1}) = f \text{ dans } \Omega_1 \qquad -\Delta(u_2^{n+1}) = f \text{ dans } \Omega_2$$
$$u_1^{n+1} = 0 \text{ sur } \partial\Omega_1 \cap \partial\Omega \quad \text{puis} \quad u_2^{n+1} = 0 \text{ sur } \partial\Omega_2 \cap \partial\Omega$$
$$u_1^{n+1} = u_2^n \text{ sur } \partial\Omega_1 \cap \overline{\Omega}_2 \qquad u_2^{n+1} = u_1^n \text{ sur } \partial\Omega_2 \cap \overline{\Omega}_1$$

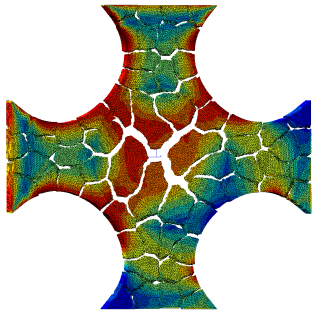
- Algorithme parallèle avec ou sans recouvrement (convergence plus lente)
- Il existe des versions purement algébriques $\Rightarrow$ solveur boîte noire

# Un peu d'histoire

## Méthode de sous-structuration ou Schur (Przemieniecki, 1963)





- Renumérotation des *ddls*
  $i$ = intérieur, $b$ = interface (boundary)

$$\begin{pmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ib} \\ \boldsymbol{K}_{bi} & \boldsymbol{K}_{bb} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i \\ \boldsymbol{u}_b \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{f}_b \end{pmatrix}$$

- Décomposition

$$\boldsymbol{f} = \boldsymbol{f}^\alpha + \boldsymbol{f}^\beta = \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{f}_b^\alpha \end{pmatrix} + \begin{pmatrix} 0 \\ \boldsymbol{f}_b^\beta \end{pmatrix}$$

$$\boldsymbol{u} = \boldsymbol{u}^\alpha + \boldsymbol{u}^\beta = \begin{pmatrix} \boldsymbol{u}_i^\alpha \\ 0 \end{pmatrix} + \begin{pmatrix} \boldsymbol{u}_i^\beta \\ \boldsymbol{u}_b^\beta \end{pmatrix}$$

- Par linéarité, on obtient deux systèmes

$$\begin{pmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ib} \\ \boldsymbol{K}_{bi} & \boldsymbol{K}_{bb} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^\alpha \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{f}_b^\alpha \end{pmatrix} \qquad \begin{pmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ib} \\ \boldsymbol{K}_{bi} & \boldsymbol{K}_{bb} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^\beta \\ \boldsymbol{u}_b \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{f}_b^\beta \end{pmatrix}$$

# Un peu d'histoire

## Méthode de sous-structuration ou Schur (Przemieniecki, 1963)

- Par linéarité, on obtient deux systèmes

$$\begin{pmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ib} \\ \boldsymbol{K}_{bi} & \boldsymbol{K}_{bb} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^{\alpha} \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{f}_b^{\alpha} \end{pmatrix} \qquad \begin{pmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ib} \\ \boldsymbol{K}_{bi} & \boldsymbol{K}_{bb} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^{\beta} \\ \boldsymbol{u}_b \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{f}_b^{\beta} \end{pmatrix}$$

1. Calcul du dépl. intérieur produit par les efforts internes : $\boldsymbol{u}_i^{\alpha} = \boldsymbol{K}_{ii}^{-1} \boldsymbol{f}_i$
2. Effort d'interface associé à ces déplacements : $\boldsymbol{f}_b^{\alpha} = \boldsymbol{K}_{bi} \boldsymbol{K}_{ii}^{-1} \boldsymbol{f}_i$
3. Effort restant pour le syst. de droite : $\boldsymbol{f}_b^{\beta} = \boldsymbol{f}_b - \boldsymbol{f}_b^{\alpha} = \boldsymbol{f}_b - \boldsymbol{K}_{bi} \boldsymbol{K}_{ii}^{-1} \boldsymbol{f}_i$
4. Élimination de $\boldsymbol{u}_i^{\beta}$ : $\boldsymbol{u}_i^{\beta} = -\boldsymbol{K}_{ii}^{-1} \boldsymbol{K}_{ib} \boldsymbol{u}_b$
5. Système final

$$(\boldsymbol{K}_{bb} - \boldsymbol{K}_{bi} \boldsymbol{K}_{ii}^{-1} \boldsymbol{K}_{ib}) \boldsymbol{u}_b = \boldsymbol{f}_b - \boldsymbol{K}_{bi} \boldsymbol{K}_{ii}^{-1} \boldsymbol{f}_i$$

- Résolution de systèmes plus petits ($\boldsymbol{K}_{ii}$ est diagonale par blocs), $(\boldsymbol{K}_{bb} - \boldsymbol{K}_{bi} \boldsymbol{K}_{ii}^{-1} \boldsymbol{K}_{ib})$ est de la taille de l'interface

# Cadre et objectifs du cours

**Objectifs**

- Avoir un aperçu des méthodes de décomposition de domaine existantes
- Connaître les notions importantes (complexité, extensibilité, problème grossier, ...)

**Limites :**

- On se restreint à la partie résolution de systèmes linéaires en mécanique du solide
- On se limite aux méthodes de type Schur simple (pas de FETI-DP, ni BDDC)
- On survole la partie informatique

# Plan


1. Introduction
   - Informatique et parallélisme
   - Problème type du mécanicien
   - Cadre et objectifs du cours
2. Rappels sur la résolution de systèmes linéaires
   - Méthodes directes
   - Méthodes itératives
3. Méthodes de type Schur
   - Problème modèle & sous-structuration
   - Méthode de Schur Primal
   - Méthode Neumann-Neumann
   - Méthode BDD
   - Méthode FETI
4. Problèmes ouverts et connexes
   - Robustesse des méthodes DD
   - Passage à l'échelle sur cas industriel

# Sommaire

1 Introduction

2 Rappels sur la résolution de systèmes linéaires
- Méthodes directes
- Méthodes itératives

3 Méthodes de type Schur

4 Problèmes ouverts et connexes

# Méthodes directes

- Les méthodes de résolution directes remplacent le système initial par des systèmes faciles à résoudre. Le calcul se fait en trois étapes :

1. Calcul de la factorisation $\boldsymbol{LU}$

$$\boldsymbol{LU} = \boldsymbol{PA}$$

Alors

$$\boldsymbol{LUx} = \boldsymbol{PAx} = \boldsymbol{Pb}$$

2. Résolution d'un système tri. inf. à diagonale unitaire

$$\boldsymbol{Ly} = \boldsymbol{Pb} \qquad \text{(descente, } \textit{forward substitution}\text{)}$$

3. Résolution d'un système tri. supérieur

$$\boldsymbol{Ux} = \boldsymbol{y} \qquad \text{(remontée, } \textit{backward substitution}\text{)}$$

# Méthodes itératives

- Les solveurs directs sont robustes mais très gourmand en mémoire
- La parallélisation est possible mais pas évidente
- Pour les très gros problèmes il faut penser aux méthodes itératives ou hybride
- **En dense**, un produit MV en $O(n^2) \Rightarrow$ uniquement utile pour le creux
- Solveurs itératifs bien adapté au parallèle

$$\boldsymbol{A}\boldsymbol{x} \quad \Rightarrow \quad \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{A}_{11}\boldsymbol{x}_1 + \boldsymbol{A}_{12}\boldsymbol{x}_2 \\ \boldsymbol{A}_{21}\boldsymbol{x}_1 + \boldsymbol{A}_{22}\boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{11}\boldsymbol{x}_1 + \boldsymbol{A}_{12}\boldsymbol{x}_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{A}_{21}\boldsymbol{x}_1 + \boldsymbol{A}_{22}\boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{11}\boldsymbol{x}_1 \\ \boldsymbol{A}_{21}\boldsymbol{x}_1 \end{bmatrix} + \begin{bmatrix} \boldsymbol{A}_{12}\boldsymbol{x}_2 \\ \boldsymbol{A}_{22}\boldsymbol{x}_2 \end{bmatrix}$$

## Principe

- On souhaite résoudre $\boldsymbol{Ax} = \boldsymbol{b}$ avec $\boldsymbol{A}$ très grande et creuse
- Construire une suite de vecteurs tels que $\lim_{k \to +\infty} \boldsymbol{x}_k = \boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{x}^\star$
- Soit $\boldsymbol{x}_k$ l'approximation au pas $k$

$$\boldsymbol{x}^\star = \boldsymbol{x}_k + \boldsymbol{e}_k \quad \text{(erreur)}$$

$$\boldsymbol{A}\boldsymbol{e}_k = \boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{A}\boldsymbol{x}_k := \boldsymbol{r}_k \quad \text{(résidu)}$$

On a donc

$$\boldsymbol{x}^\star = \boldsymbol{x}_k + \boldsymbol{A}^{-1}\boldsymbol{r}_k$$

- L'idée est de remplacer $\boldsymbol{A}$ par une matrice proche mais facilement inversible

## Principe

- Soit $\boldsymbol{M}$ une matrice inversible qui
  - est une bonne approximation de $\boldsymbol{A}$
  - soit facile à calculer
  - permette de résoudre facilement le système $\boldsymbol{Mz} = \boldsymbol{r}$
- $\boldsymbol{M}$ est appelé **préconditionneur**
- Au lieu de résoudre $\boldsymbol{x}^\star = \boldsymbol{x}_k + \boldsymbol{A}^{-1}\boldsymbol{r}_k$, on itère :

$$\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$$
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{M}^{-1}\boldsymbol{r}_k$$

- Trois étapes :
  calcul de résidu $\Rightarrow$ résolution du pb préconditionné $\Rightarrow$ maj solution

# Préconditionnement

## Comment construire un préconditionneur

- Décomposer la matrice $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$ où $\boldsymbol{M}$ est (facilement) inversible
- On a l'équivalence

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{M}^{-1}\boldsymbol{r}_k \Leftrightarrow \boldsymbol{M}\boldsymbol{x}_{k+1} = \boldsymbol{N}\boldsymbol{x}_k + \boldsymbol{b}$$

- Quelques manipulations algébriques mènent à la formulation sous forme de point fixe

$$\boldsymbol{x}_{k+1} = \boldsymbol{B}\boldsymbol{x}_k + \boldsymbol{c} \quad \text{avec } \boldsymbol{B} = \boldsymbol{M}^{-1}\boldsymbol{N} \text{ et } \boldsymbol{c} = \boldsymbol{M}^{-1}\boldsymbol{b}$$

- $\boldsymbol{B}$ est la matrice d'itération $\boldsymbol{e}_{k+1} = \boldsymbol{B}\boldsymbol{e}_k$

# Méthodes itératives stationnaires

## Définition

- Une méthode itérative est convergente ssi la suite $(\boldsymbol{x}_n)_n \rightarrow \boldsymbol{A}^{-1}\boldsymbol{f}\ \forall$ init. $\boldsymbol{x}_0$
- Dans notre cas, la méthode converge uniquement si le rayon spectral de la matrice d'itération

$$\rho(\boldsymbol{B}) = \rho(\boldsymbol{M}^{-1}\boldsymbol{N}) < 1$$

- Suivant le choix de $\boldsymbol{M}$ on obtient les méthodes Jacobi, Gauss-Seidel, de relaxation ...
- ... et des propriétés de convergence différentes.

# Méthodes itératives de type Krylov

- Avec les méthodes précédentes

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \boldsymbol{M}^{-1}\boldsymbol{r}_{k-1} = \boldsymbol{x}_{k-1} + \boldsymbol{z}_{k-1}$$

- On construit successivement $\boldsymbol{z}_k = Q_k(\boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{z}_0$ où $Q_k$ est un polynôme de degré $k$.
- Déf : Espace de Krylov :

$$\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0) = \text{Vect}(\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{r}_0)$$

- Les méthodes de projection ajoutent une contrainte sur $\boldsymbol{r}_m$ tel que

$$\begin{cases} \boldsymbol{x}_m \in \ \boldsymbol{x}_0 + \mathcal{K}_m(\boldsymbol{K}, \boldsymbol{r}_0) \\ \boldsymbol{r}_m \perp_? \ \mathcal{K}_m(\boldsymbol{K}, \boldsymbol{r}_0) \end{cases}$$

- Dans le cas précédent $\boldsymbol{K} \leftarrow \boldsymbol{M}^{-1}\boldsymbol{A}$ et $\boldsymbol{r}_0 \leftarrow \boldsymbol{z}_0$

# Méthodes itératives de type Krylov

- Les méthodes de projection ajoutent une contrainte sur $\boldsymbol{r}_m$ tel que

$$\begin{cases} \boldsymbol{x}_m \in \boldsymbol{x}_0 + \mathcal{K}_m(\boldsymbol{K}, \boldsymbol{r}_0) \\ \boldsymbol{r}_m \perp_? \mathcal{K}_m(\boldsymbol{K}, \boldsymbol{r}_0) \end{cases}$$

- Suivant les propriétés de $\boldsymbol{K}$, le choix du type l'orthogonalité permet de définir plusieurs approches (CG, GMRes, OrthoDir, etc.)
- Construction d'une base de l'espace de Krylov (procédé d'Arnoldi)
- Si $\boldsymbol{A}$ est symétrique définie positive le gradient conjugué est la meilleure solution

$$\begin{cases} \boldsymbol{x}_m \in \boldsymbol{x}_0 + \mathcal{K}_m(\boldsymbol{K}, \boldsymbol{r}_0) \\ \boldsymbol{r}_m \perp \mathcal{K}_m(\boldsymbol{K}, \boldsymbol{r}_0) \end{cases}$$

- Si on connait des informations sur la solution, on peut utiliser des techniques d'augmentation ou de déflation

$$\begin{cases} \boldsymbol{x}_m \in \boldsymbol{x}_0 + \mathcal{K}_m(\boldsymbol{K}, \boldsymbol{r}_0) + \mathrm{Vect}(\boldsymbol{C}) \\ \boldsymbol{r}_m \perp \mathcal{K}_m(\boldsymbol{K}, \boldsymbol{r}_0) + \mathrm{Vect}(\boldsymbol{C}) \end{cases}$$

# Sommaire

1 Introduction

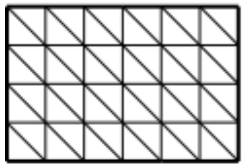
2 Rappels sur la résolution de systèmes linéaires

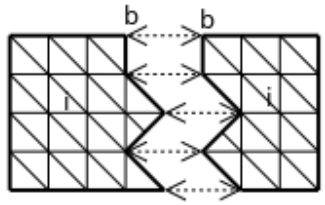
3 Méthodes de type Schur
- Problème modèle & sous-structuration
- Méthode de Schur Primal
- Méthode Neumann-Neumann
- Méthode BDD
- Méthode FETI

4 Problèmes ouverts et connexes

# Problème modèle & sous-structuration

- Système global $\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f}$ avec $\boldsymbol{K}$ SDP
- Découpage virtuel en deux domaines avec renumerotation des inconnues $i$ = intérieur, $b$ = bord






- Opérateur & second membre indépendants par sous-domaine

$$\begin{pmatrix} \boldsymbol{K}^1_{ii} & \boldsymbol{K}^1_{ib} \\ \boldsymbol{K}^1_{bi} & \boldsymbol{K}^1_{bb} \end{pmatrix}, \begin{pmatrix} \boldsymbol{f}^1_i \\ \boldsymbol{f}^1_b \end{pmatrix} \qquad \begin{pmatrix} \boldsymbol{K}^2_{ii} & \boldsymbol{K}^2_{ib} \\ \boldsymbol{K}^2_{bi} & \boldsymbol{K}^2_{bb} \end{pmatrix}, \begin{pmatrix} \boldsymbol{f}^2_i \\ \boldsymbol{f}^2_b \end{pmatrix}$$

- Après assemblage on obtient le système global

$$\begin{pmatrix} \boldsymbol{K}^1_{ii} & 0 & \boldsymbol{K}^1_{ib} \\ 0 & \boldsymbol{K}^2_{ii} & \boldsymbol{K}^2_{ib} \\ \boldsymbol{K}^1_{bi} & \boldsymbol{K}^2_{bi} & \boldsymbol{K}_{bb} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^1_i \\ \boldsymbol{u}^2_i \\ \boldsymbol{u}_b \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}^1_i \\ \boldsymbol{f}^2_i \\ \boldsymbol{f}_b \end{pmatrix} \quad \text{avec} \begin{cases} \boldsymbol{K}_{bb} = \boldsymbol{K}^1_{bb} + \boldsymbol{K}^2_{bb} \\ \boldsymbol{f}_b = \boldsymbol{f}^1_b + \boldsymbol{f}^2_b \end{cases}$$

# Problème modèle & sous-structuration

- Après assemblage on obtient le système global

$$\begin{pmatrix} \boldsymbol{K}_{ii}^1 & 0 & \boldsymbol{K}_{ib}^1 \\ 0 & \boldsymbol{K}_{ii}^2 & \boldsymbol{K}_{ib}^2 \\ \boldsymbol{K}_{bi}^1 & \boldsymbol{K}_{bi}^2 & \boldsymbol{K}_{bb} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^1 \\ \boldsymbol{u}_i^2 \\ \boldsymbol{u}_b \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^1 \\ \boldsymbol{f}_i^2 \\ \boldsymbol{f}_b \end{pmatrix} \qquad \text{avec} \begin{cases} \boldsymbol{K}_{bb} = \boldsymbol{K}_{bb}^1 + \boldsymbol{K}_{bb}^2 \\ \boldsymbol{f}_b = \boldsymbol{f}_b^1 + \boldsymbol{f}_b^2 \end{cases}$$

- Systèmes sous-structurés équivalents

$$\begin{pmatrix} \boldsymbol{K}_{ii}^1 & \boldsymbol{K}_{ib}^1 \\ \boldsymbol{K}_{bi}^1 & \boldsymbol{K}_{bb}^1 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^1 \\ \boldsymbol{u}_b^1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^1 \\ \boldsymbol{f}_b^1 + (\boldsymbol{f}_b^2 - (\boldsymbol{K}_{bi}^2 \boldsymbol{u}_i^2 + \boldsymbol{K}_{bb}^2 \boldsymbol{u}_i^2)) \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{K}_{ii}^2 & \boldsymbol{K}_{ib}^2 \\ \boldsymbol{K}_{bi}^2 & \boldsymbol{K}_{bb}^2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^2 \\ \boldsymbol{u}_b^2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^2 \\ \boldsymbol{f}_b^2 + (\boldsymbol{f}_b^1 - (\boldsymbol{K}_{bi}^1 \boldsymbol{u}_i^1 + \boldsymbol{K}_{bb}^1 \boldsymbol{u}_i^1)) \end{pmatrix}$$

$$\boldsymbol{u}_b^1 = \boldsymbol{u}_b^2 = \boldsymbol{u}_b$$

# Problème modèle & sous-structuration

- Systèmes sous-structurés équivalents

$$\begin{pmatrix} \boldsymbol{K}_{ii}^1 & \boldsymbol{K}_{ib}^1 \\ \boldsymbol{K}_{bi}^1 & \boldsymbol{K}_{bb}^1 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^1 \\ \boldsymbol{u}_b^1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^1 \\ \boldsymbol{f}_b^1 + (\boldsymbol{f}_b^2 - (\boldsymbol{K}_{bi}^2 \boldsymbol{u}_i^2 + \boldsymbol{K}_{bb}^2 \boldsymbol{u}_i^2)) \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{K}_{ii}^2 & \boldsymbol{K}_{ib}^2 \\ \boldsymbol{K}_{bi}^2 & \boldsymbol{K}_{bb}^2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^2 \\ \boldsymbol{u}_b^2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^2 \\ \boldsymbol{f}_b^2 + (\boldsymbol{f}_b^1 - (\boldsymbol{K}_{bi}^1 \boldsymbol{u}_i^1 + \boldsymbol{K}_{bb}^1 \boldsymbol{u}_i^1)) \end{pmatrix}$$

$$\boldsymbol{u}_b^1 - \boldsymbol{u}_b^2 = 0$$

- Systèmes sous-structurés équivalents

$$\begin{pmatrix} \boldsymbol{K}_{ii}^1 & \boldsymbol{K}_{ib}^1 \\ \boldsymbol{K}_{bi}^1 & \boldsymbol{K}_{bb}^1 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^1 \\ \boldsymbol{u}_b^1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^1 \\ \boldsymbol{f}_b^1 + \boldsymbol{\lambda}_b^1 \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{K}_{ii}^2 & \boldsymbol{K}_{ib}^2 \\ \boldsymbol{K}_{bi}^2 & \boldsymbol{K}_{bb}^2 \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^2 \\ \boldsymbol{u}_b^2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^2 \\ \boldsymbol{f}_b^2 + \boldsymbol{\lambda}_b^2 \end{pmatrix}$$

$$\boldsymbol{u}_b^1 - \boldsymbol{u}_b^2 = 0$$

$$\boldsymbol{\lambda}_b^1 + \boldsymbol{\lambda}_b^2 = 0$$

# Problème modèle & sous-structuration

- Système global

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} \quad \text{avec} \quad \boldsymbol{K} \text{ SDP}$$

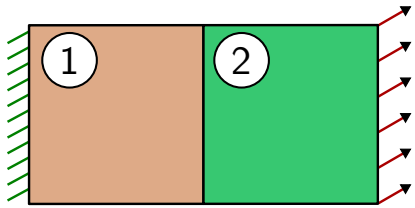
- Décompo. sans recouvrement

$$\boldsymbol{K}^s\boldsymbol{u}^s = \boldsymbol{f}^s + \boldsymbol{\lambda}^s,\ s = 1, 2$$

$$\boldsymbol{\lambda}_b^1 + \boldsymbol{\lambda}_b^2 = 0$$

$$\boldsymbol{u}_b^1 - \boldsymbol{u}_b^2 = 0$$

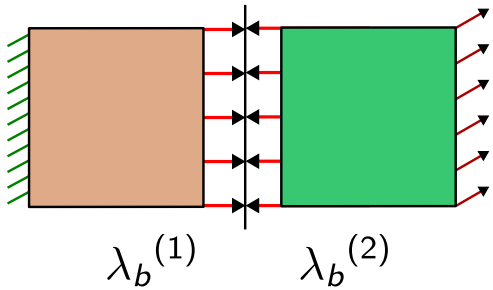
- Problème global $\Leftrightarrow$
  - Équilibre locaux
  - Équilibre interface
  - Continuité interface




Problème init.

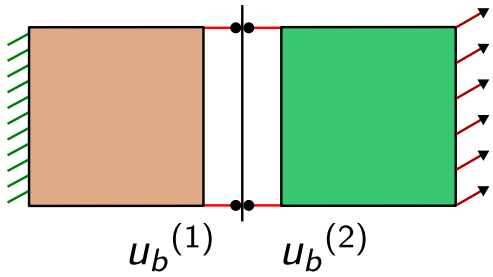
⇩

Décomposition




Équilibre

$\lambda_b{}^{(1)} = -\lambda_b{}^{(2)}$




Continuité

$u_b{}^{(1)} = u_b{}^{(2)}$

# Problème modèle & sous-structuration

- Retour sur les équilibres locaux

$$\begin{pmatrix} \boldsymbol{K}_{ii}^s & \boldsymbol{K}_{ib}^s \\ \boldsymbol{K}_{bi}^s & \boldsymbol{K}_{bb}^s \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^s \\ \boldsymbol{u}_b^s \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^s \\ \boldsymbol{f}_b^s + \boldsymbol{\lambda}_b^s \end{pmatrix}$$

- Les inconnues internes peuvent être éliminées

$$\begin{cases} \boldsymbol{u}_i^s = \boldsymbol{K}_{ii}^{s\,-1}(\boldsymbol{f}_i^s - \boldsymbol{K}_{ib}^s \boldsymbol{u}_b^s) \\ (\boldsymbol{K}_{bb}^s - \boldsymbol{K}_{bi}^s \boldsymbol{K}_{ii}^{s\,-1} \boldsymbol{K}_{ib}^s)\ \boldsymbol{u}_b^s = \boldsymbol{\lambda}_b^s + \boldsymbol{f}_b^s - \boldsymbol{K}_{bi}^s \boldsymbol{K}_{ii}^{s\,-1} \boldsymbol{f}_i^s \end{cases}$$

$$\begin{cases} \boldsymbol{u}_i^s = \boldsymbol{K}_{ii}^{s\,-1}(\boldsymbol{f}_i^s - \boldsymbol{K}_{ib}^s \boldsymbol{u}_b^s) \\ \boldsymbol{S}^s \boldsymbol{u}_b^s = \boldsymbol{\lambda}_b^s + \boldsymbol{b}^s \end{cases}$$

- $\boldsymbol{S}^s$ est le complément de Schur primal de $\boldsymbol{K}_{ii}^s$ dans $\boldsymbol{K}^s$. C'est une matrice carré d'ordre $n_b$ (taille de l'interface)
- $\boldsymbol{b}^s$ est second membre condensé

# Remarque sur le complément de Schur

- Le complément de schur $\boldsymbol{S}^s$ est une matrice dense
- Elle hérite des nombreuses propriétés de $\boldsymbol{K}$ (symétrie, profil triangulaire, M-matrices, H-matrices, ...)
- Le complément de schur apparaît naturellement lors d'une élimination de Gauss par blocs
- D'un point de vue mécanique, le complément de schur relie efforts et déplacements d'interface

$$\boldsymbol{S}^s \boldsymbol{u}_b^s = \boldsymbol{\lambda}_b^s + \boldsymbol{b}^s$$

- Additivité du rang (Guttman 1946)

$$\text{rang}(\boldsymbol{K}^s) = \text{rang}(\boldsymbol{K}_{ii}^s) + \text{rang}(\boldsymbol{S}^s)$$

# Quelques définitions

**Optimalité théorique (*optimality*)**

- Une méthode de résolution itérative est *optimale* si son taux de convergence est indépendant de la taile du système à résoudre.

**Extensibilité théorique (*scalability*)**

- Une méthode itérative de type décomposition de domaine pour résoudre des systèmes linéaires est *extensible* si son taux de convergence ne se détériore pas quand le nombre de sous-domaines augmentent.

En pratique deux versions de l'extensibilité :

**Extensibilité forte (*strong scalability*)**

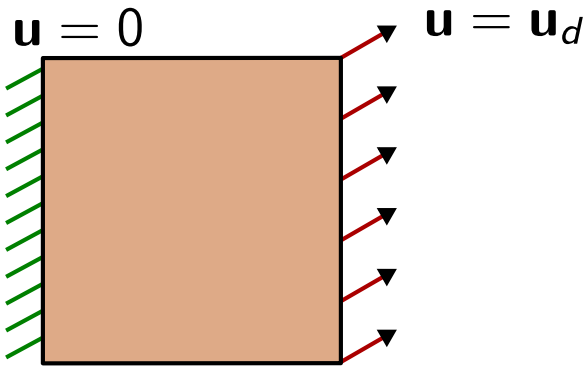
- Pour un problème de taille fixée (h fixé), le temps d'exécution est inversement proportionnel au nombre de processeurs.

**Extensibilité faible (*weak scalability*)**

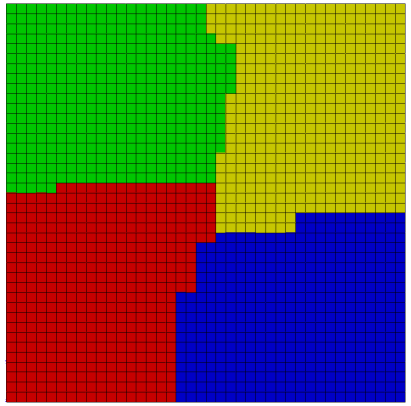
- Le temps d'exécution ne varie pas lorsque l'on augmente la taille du problème global, en augmentant le nombre de processeurs dans la même proportion (H/h fixé).

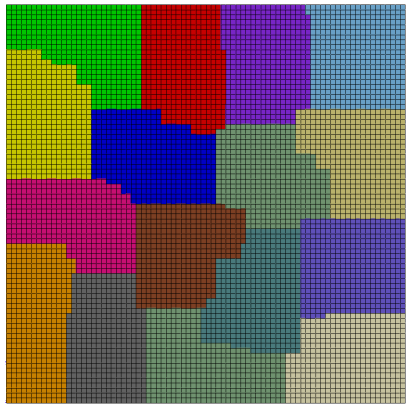
# Cas test 1 : carre 2D élastique (contrainte plane)

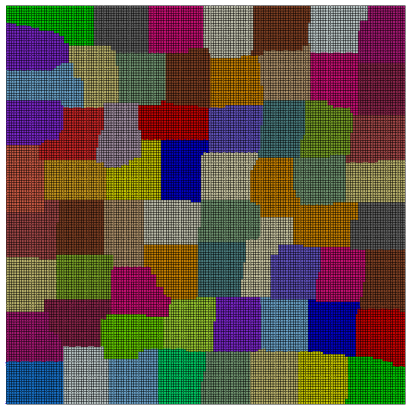
- Problème physique




- Étude de type extensibilité faible ($H/h = 20$)
- Maillage quadratique

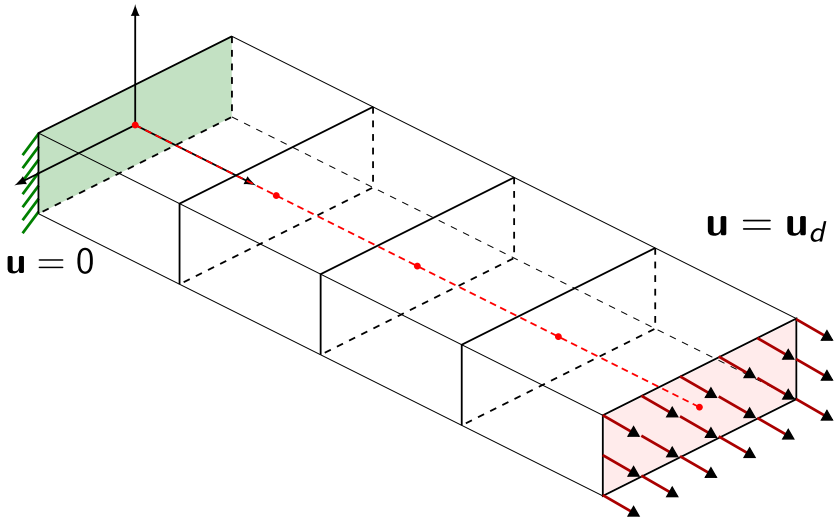


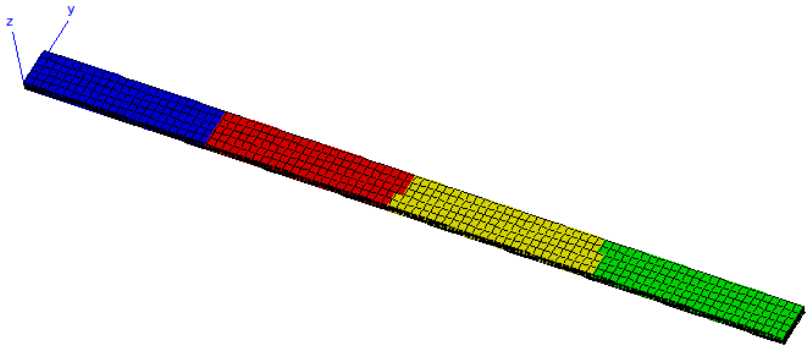




# Cas test 2 : poutre en flexion

- Problème physique
- Maillage quadratique



# Méthode de Schur primal (Przemieniecki, 1963)

- L'idée originale est de résoudre $\boldsymbol{S}\boldsymbol{u}_b = \boldsymbol{b}$ avec un solveur itératif (Gradient conjugué)
- D'un point de vue parallélisme
  - Factorisation parallèle de $\boldsymbol{K}_{ii}^s$ et substitutions en parallèle
  - Calcul parallèle des $2^{nd}$ membre
  - Le produit $\boldsymbol{S}\boldsymbol{u}_b$ est effectué en parallèle car $\boldsymbol{S}\boldsymbol{u}_b = \boldsymbol{S}^1\boldsymbol{u}_b + \boldsymbol{S}^2\boldsymbol{u}_b$

**Algorithme 1 :** PGC

Init. $\boldsymbol{u}_{b0}, \boldsymbol{r}_0 = \boldsymbol{S}\boldsymbol{u}_{b0} - \boldsymbol{b}$

$\boldsymbol{z}_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0, \boldsymbol{w}_0 = \boldsymbol{z}_0$

**for** $j = 0 \dots n$ **do**

$\boldsymbol{q}_j = \boldsymbol{S}\boldsymbol{w}_j$

$\alpha_j = (\boldsymbol{z}_j, \boldsymbol{r}_j)/(\boldsymbol{q}_j, \boldsymbol{w}_j)$

$\boldsymbol{u}_{bj+1} = \boldsymbol{u}_{bj} + \alpha_j \boldsymbol{w}_j$

$\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j \boldsymbol{q}_j$

$\boldsymbol{z}_{j+1} = \boldsymbol{M}^{-1}\boldsymbol{r}_{j+1}$

$\beta_j = (\boldsymbol{z}_{j+1}, \boldsymbol{q}_j)/(\boldsymbol{q}_j, \boldsymbol{w}_j)$

$\boldsymbol{w}_{j+1} = \boldsymbol{z}_{j+1} - \beta_j \boldsymbol{w}_j$

$\boldsymbol{u}_b \Rightarrow \boldsymbol{u}_i^s = \boldsymbol{K}_{ii}^{s\,-1}(\boldsymbol{f}_i^s - \boldsymbol{K}_{bi}^s \boldsymbol{u}_b)$

- Quantités échangés
- Taux de convergence du PCG

$$\|\boldsymbol{e}_k\|_{\boldsymbol{S}} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|\boldsymbol{e}_0\|_{\boldsymbol{S}}$$

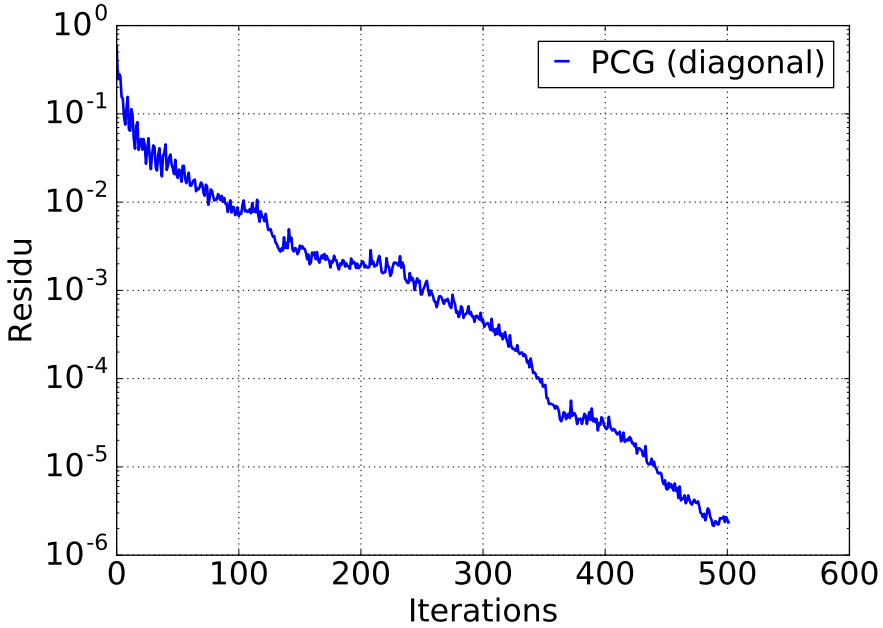
où $\kappa = \mathrm{Cond}(\boldsymbol{M}, \boldsymbol{S})$

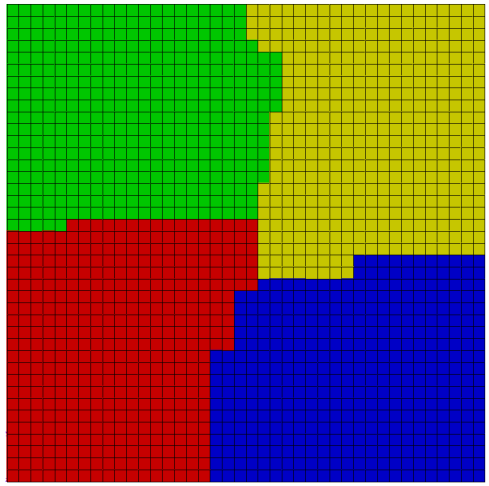
- Il faut trouver un bon préconditionneur !

$$\mathrm{Cond}(\boldsymbol{S}) \text{ en } O(1/h)$$

# Méthode de Schur primal (Przemieniecki, 1963)

- Cas du carré avec 4 domaines
- Préconditionneur diagonal



- Résultats peu satisfaisants
- Il faut trouver un meilleur préconditionneur

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{I} & 0 & 0 \\ 0 & \boldsymbol{I} & 0 \\ \boldsymbol{K}_{bi}^1 \boldsymbol{K}_{ii}^{1\,-1} & \boldsymbol{K}_{bi}^2 \boldsymbol{K}_{ii}^{2\,-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{K}_{ii}^1 & 0 & 0 \\ 0 & \boldsymbol{K}_{ii}^2 & 0 \\ 0 & 0 & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & 0 & \boldsymbol{K}_{ii}^{1\,-1} \boldsymbol{K}_{ib}^1 \\ 0 & \boldsymbol{I} & \boldsymbol{K}_{ii}^{2\,-1} \boldsymbol{K}_{ib}^2 \\ 0 & 0 & \boldsymbol{I} \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{K}_{ii}^1 & 0 & 0 \\ 0 & \boldsymbol{K}_{ii}^2 & 0 \\ 0 & 0 & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & 0 & \boldsymbol{K}_{ii}^{1\,-1} \boldsymbol{K}_{ib}^1 \\ 0 & \boldsymbol{I} & \boldsymbol{K}_{ii}^{2\,-1} \boldsymbol{K}_{ib}^2 \\ 0 & 0 & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^1 \\ \boldsymbol{u}_i^2 \\ \boldsymbol{u}_b \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & 0 & 0 \\ 0 & \boldsymbol{I} & 0 \\ \boldsymbol{K}_{bi}^1 \boldsymbol{K}_{ii}^{1\,-1} & \boldsymbol{K}_{bi}^2 \boldsymbol{K}_{ii}^{2\,-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_i^1 \\ \boldsymbol{f}_i^2 \\ \boldsymbol{f}_b \end{pmatrix}$$

**Algorithme 2 :** Solveurs directs parallèles, principe de base

Factorisation parallèle de $\boldsymbol{K}_{ii}^{(s)}$

Calcul parallèle des compléments de Schur $\boldsymbol{S}^{(s)}$ et des $2^{nd}$ membre $\boldsymbol{b}^s$

$\boldsymbol{b}^s = \boldsymbol{f}_b^s - \boldsymbol{K}_{bi}^s \boldsymbol{K}_{ii}^{s\,-1} \boldsymbol{f}_i^s$

Assemblage du complément de Schur $\boldsymbol{S} = \sum_s \boldsymbol{S}^{(s)}$ et du $2^{nd}$ membre

Factorisation simultanée de $\boldsymbol{S} = \sum_s \boldsymbol{S}^{(s)}$

Descente remontée simultanée de $\boldsymbol{S}^{-1} \quad \Rightarrow \quad \boldsymbol{u}_b$

Descente remontée parallèle de

$(\boldsymbol{K}_{ii}^{(s)})^{-1} \quad \Rightarrow \quad \boldsymbol{u}_i^s = \boldsymbol{K}_{ii}^{s\,-1}(\boldsymbol{f}_i^s - \boldsymbol{K}_{bi}^s \boldsymbol{u}_b)$

# Méthodes de Schur

- Système condensé global

$$\boldsymbol{S}\boldsymbol{u}_b = \boldsymbol{b}$$

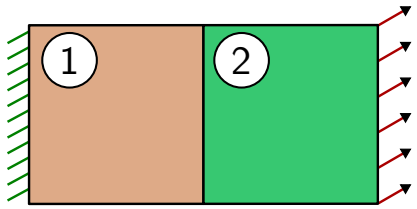
- Formulation sous-structurée condensée

$$\boldsymbol{S}^s \boldsymbol{u}_b^s = \boldsymbol{b}^s + \boldsymbol{\lambda}^s,\ s = 1, 2$$

$$\boldsymbol{\lambda}_b^1 + \boldsymbol{\lambda}_b^2 = 0$$

$$\boldsymbol{u}_b^1 - \boldsymbol{u}_b^2 = 0$$

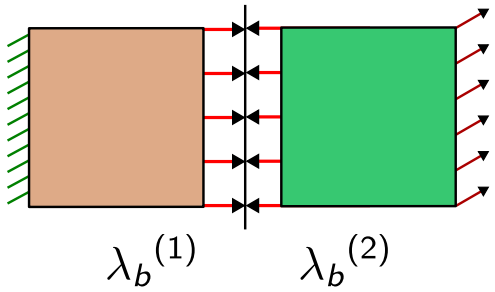
- Les méthodes de type Schur s'appuient sur ces 4 équations




Problème init.

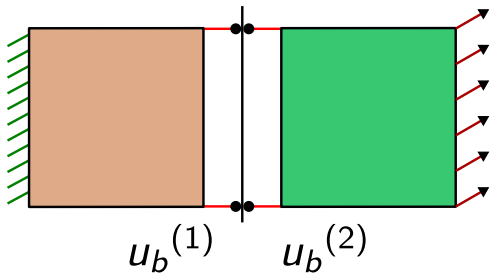
⇩

Décomposition




Équilibre

$\lambda_b^{(1)} = -\lambda_b^{(2)}$




Continuité

$u_b^{(1)} = u_b^{(2)}$

# Méthode Neumann-Neumann [Le Tallec et al., 1991]

- Idée : proposer un $\boldsymbol{M}$ qui a un sens physique
- Scaling du déséquilibre

$$\boldsymbol{r}_i^s = \delta_s \ \boldsymbol{r}_i \ s = 1, 2$$

avec $\delta_1 + \delta_2 = 1$

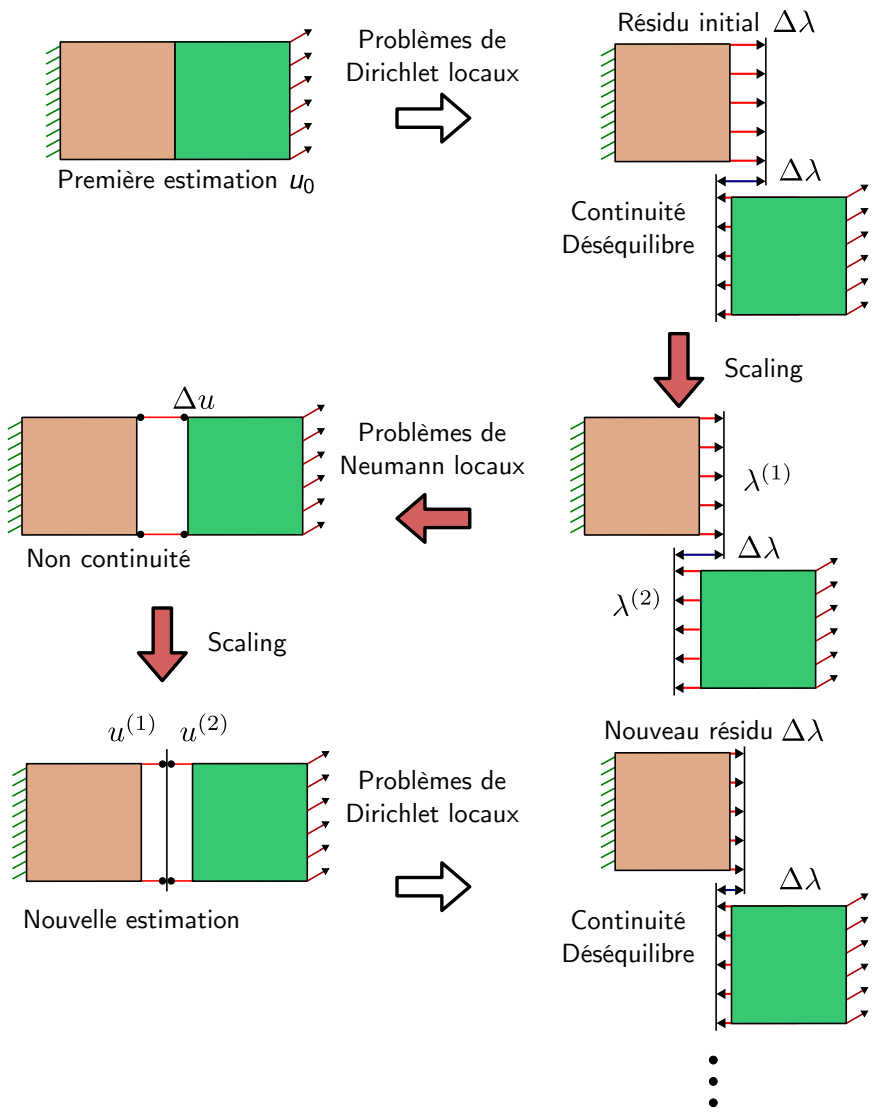
- Résolution de problèmes Neumann locaux

$$\boldsymbol{z}_i^s = \boldsymbol{S}^{s+} \boldsymbol{r}_i^s \ s = 1, 2$$

- Scaling du déplacement

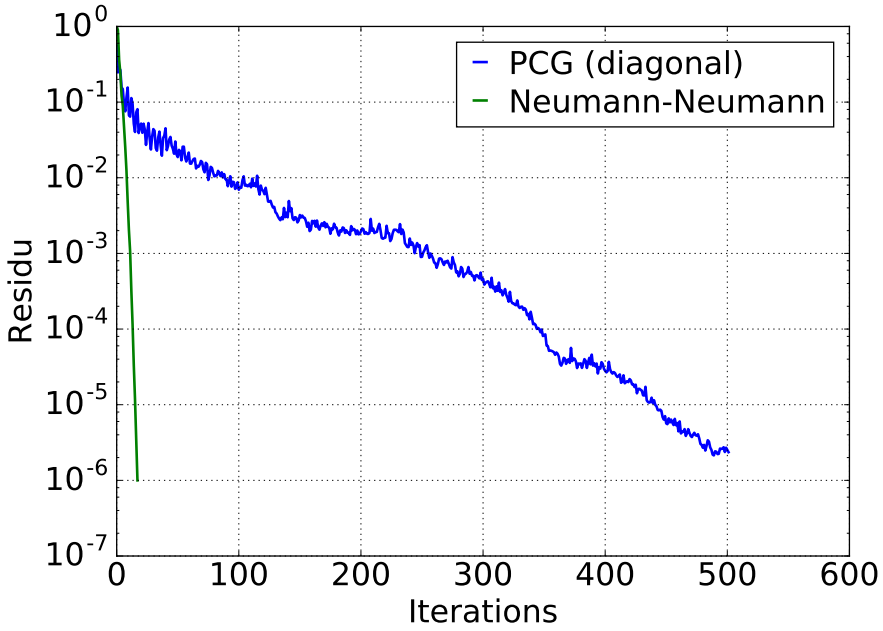
$$\boldsymbol{u}_{b,i+1} = \boldsymbol{u}_{b,i} + \sum_s \delta_s \boldsymbol{z}_i^s$$

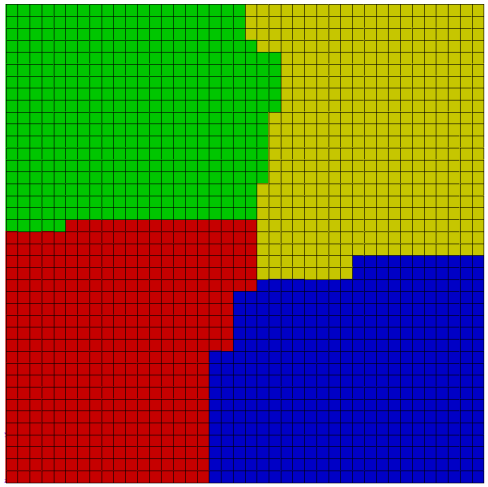
- Pourquoi des pseudo-inverses ?



# Méthode Neumann-Neumann [Le Tallec et al., 1991]

- Cas du carré avec 4 domaines
- Neumann-Neumann vs CG

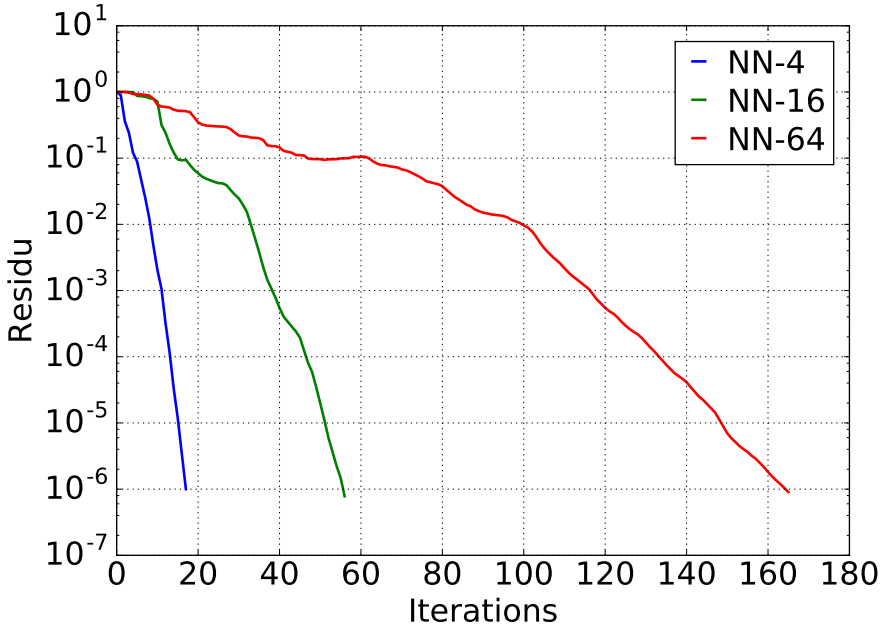


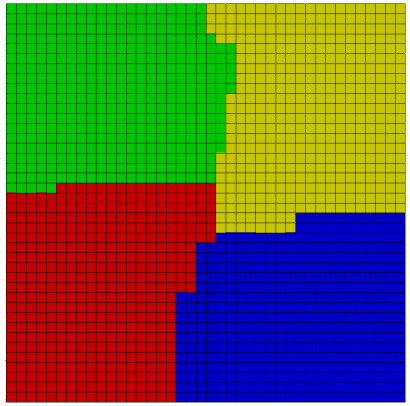



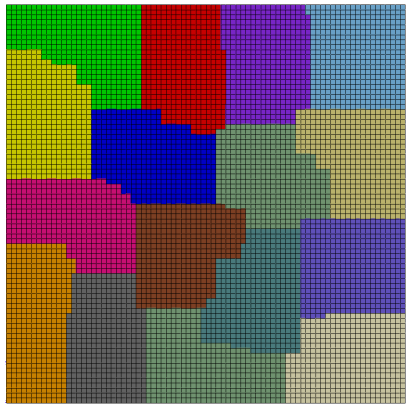
- C'est bien mieux !
- Est-ce extensible ?

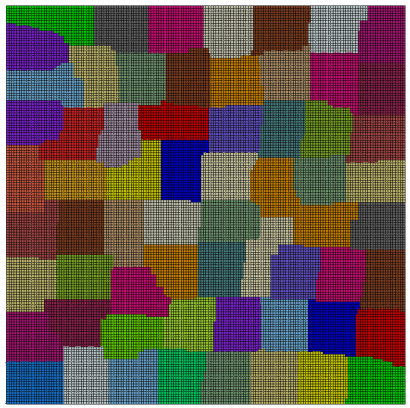
# Méthode Neumann-Neumann [Le Tallec et al., 1991]

- Extensibilité faible







- Convergence de plus en plus lente
- Conditionnement du problème d'interface avec Neumann-Neumann

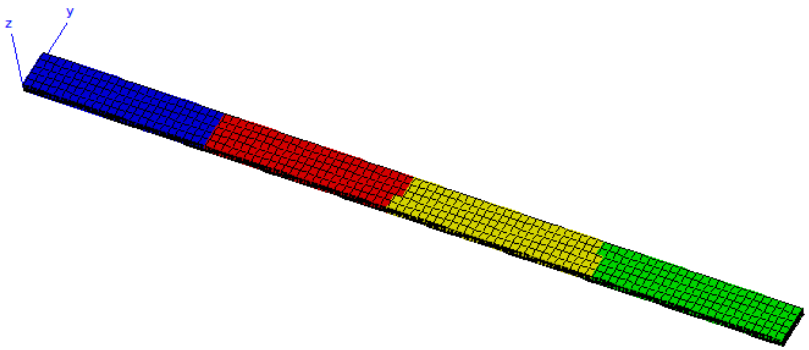
$$\kappa = O\left(\frac{1}{H^2}\left(1+\log\frac{H}{h}\right)^q\right)$$

# Nécessité d'un problème grossier

- Conditionnement du problème d'interface avec Neumann-Neumann

$$\kappa = O\left(\frac{1}{H^2}\left(1 + \log\frac{H}{h}\right)^q\right)$$

- La convergence se dégrade lorsque le nombre de sous-domaines augmente car le chargement se propage de proche en proche
- Exemple pour la poutre (NN-step.z7p)




- Besoin d'un solveur à grille grossière qui permet de propager rapidement l'information entre tous les sous-domaines

# Méthode Balancing Domain Decomposition [Mandel, 1993]

- Dans Neumann-Neumann

$$z_i^s = S^{s+} r_i^s \quad s = 1, 2$$

- Faire d'une contrainte un atout
- Rappel sur les pseudo-inverses :

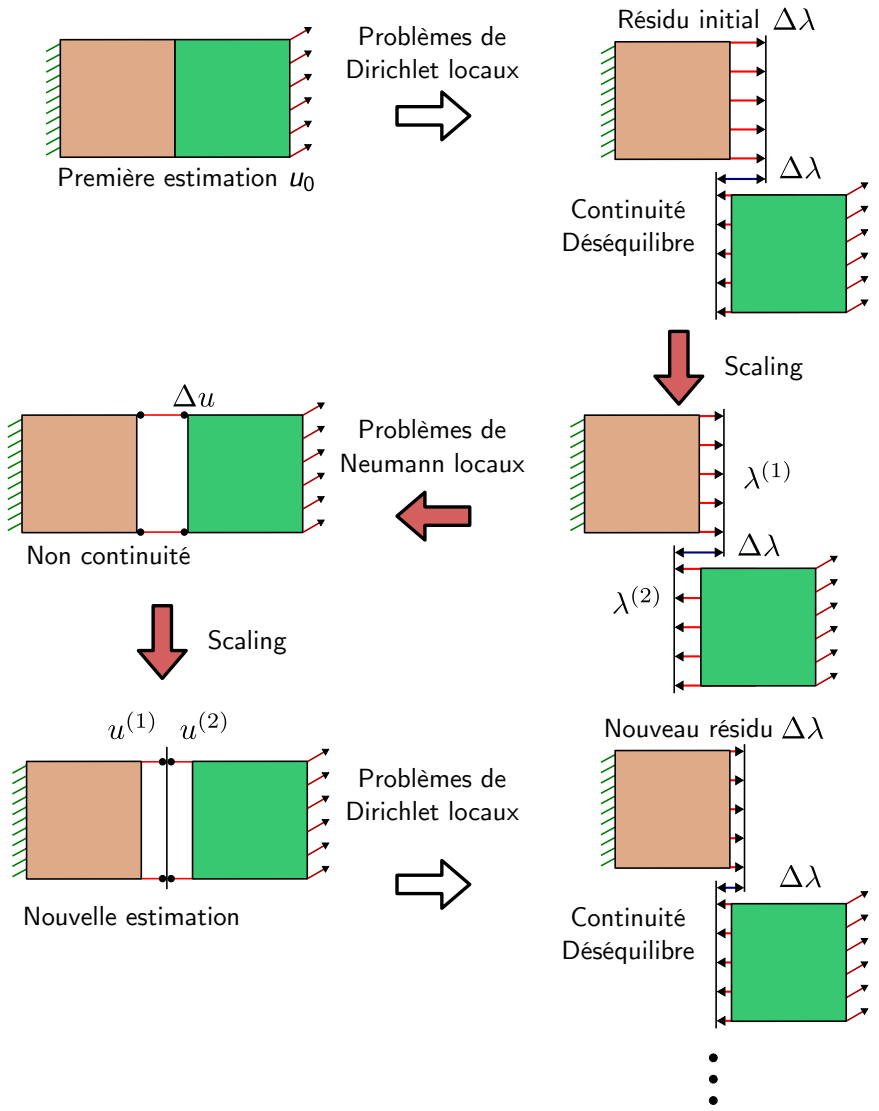
On appelle pseudo-inverse de $A$ une matrice notée $A^+$ qui vérifie

$$\forall y \in \mathrm{Im}(A),\ AA^+ y = y$$

On a : $\ker(A^T) \oplus \mathrm{Im}(A) = \mathbb{R}^n$

Soit $R$ une base de $\ker(A)$ :

$$y \in \mathrm{Im}(A) \Leftrightarrow R^T y = 0$$




Empêcher $r$ d'exciter les MCRs fournit une grille grossière naturelle

# Notations pour n domaines

- On pose,

$$\boldsymbol{x}^{\boxplus} = \begin{pmatrix} \vdots \\ \boldsymbol{x}^s \\ \vdots \end{pmatrix} \qquad \boldsymbol{x}^{\boxminus} = \begin{pmatrix} \cdots & \boldsymbol{x}^s & \cdots \end{pmatrix} \qquad \boldsymbol{x}^{\boxtimes} = \begin{pmatrix} \ddots & & \\ & \boldsymbol{x}^s & \\ & & \ddots \end{pmatrix}$$

- On a alors :

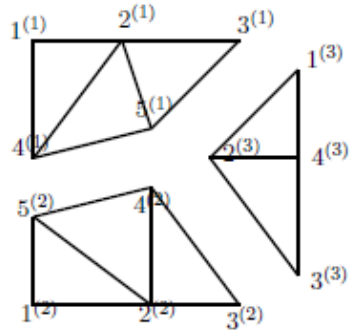
$$\begin{cases} s = 1,2 \quad \boldsymbol{S}^s \boldsymbol{u}^s = \boldsymbol{b}^s + \boldsymbol{\lambda}_b^s \\ \boldsymbol{\lambda}_b^1 + \boldsymbol{\lambda}_b^2 = \boldsymbol{0} \\ \boldsymbol{u}_b^1 - \boldsymbol{u}_b^2 = \boldsymbol{0} \end{cases} \Leftrightarrow \begin{cases} \boldsymbol{S}^{\boxtimes} \boldsymbol{u}_b^{\boxplus} = \boldsymbol{b}^{\boxplus} + \boldsymbol{\lambda}_b^{\boxplus} \\ \boldsymbol{A}^{\boxminus} \boldsymbol{\lambda}_b^{\boxplus} = \boldsymbol{0} \\ \boldsymbol{B}^{\boxminus} \boldsymbol{u}_b^{\boxplus} = \boldsymbol{0} \end{cases}$$

- Propriété fondamentale : orthogonalité des assemblages

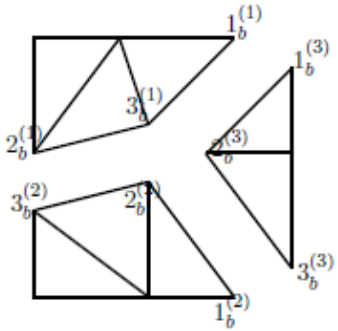
$$\boldsymbol{B}^{\boxminus} \boldsymbol{A}^{\boxminus\, T} = \sum_s \boldsymbol{B}^s \boldsymbol{A}^{s\, T} = \boldsymbol{0}$$

- D'où : $\forall \boldsymbol{x}_b^{\boxplus}, \quad \exists!(\boldsymbol{y}, \boldsymbol{z})$ tel que $\boldsymbol{x}_b^{\boxplus} = \boldsymbol{B}^{\boxminus\, T} \boldsymbol{y} + \boldsymbol{A}^{\boxminus\, T} \boldsymbol{z}$

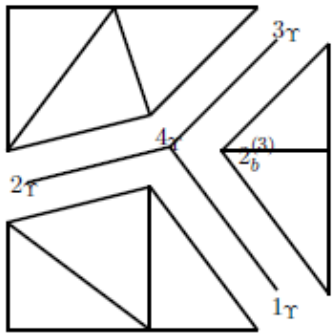
# Notations pour n domaines



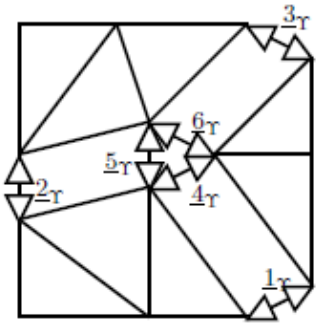
(a) Subdomains



(b) Local interface



(c) Primal interface



(d) Dual interface

$$\boldsymbol{t}^1 = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \end{pmatrix} \quad \boldsymbol{A}^1 = \begin{pmatrix} 0&0&0 \\ 0&1&0 \\ 1&0&0 \\ 0&0&1 \end{pmatrix} \quad \boldsymbol{B}^1 = \begin{pmatrix} 0&0&0 \\ 0&1&0 \\ 1&0&0 \\ 0&0&0 \\ 0&0&1 \\ 0&0&1 \end{pmatrix}$$

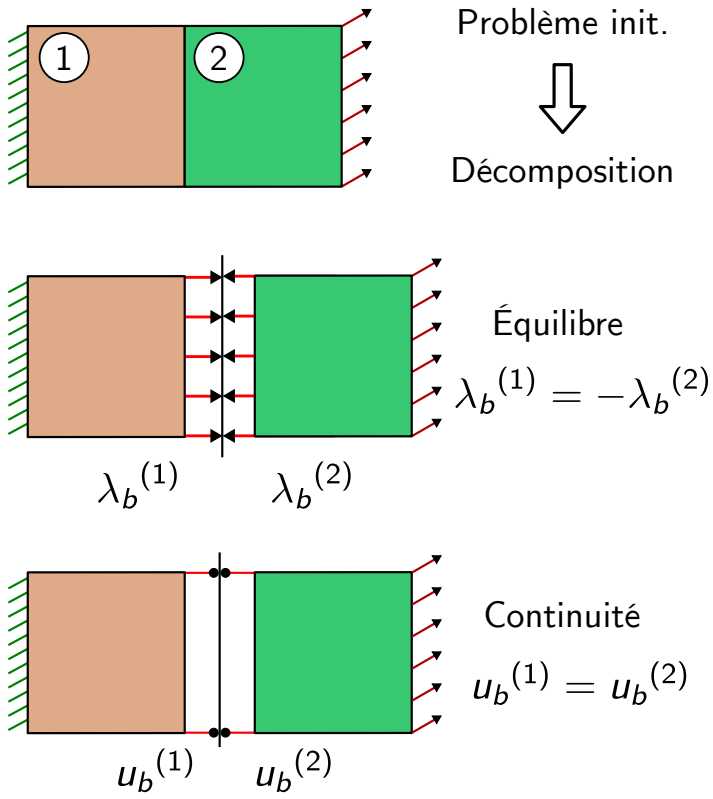
$$\boldsymbol{t}^2 = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \end{pmatrix} \quad \boldsymbol{A}^2 = \begin{pmatrix} 1&0&0 \\ 0&0&1 \\ 0&0&0 \\ 0&1&0 \end{pmatrix} \quad \boldsymbol{B}^2 = \begin{pmatrix} 1&0&0 \\ 0&0&-1 \\ 0&0&0 \\ 0&1&0 \\ 0&-1&0 \\ 0&0&0 \end{pmatrix}$$

# Méthode Balancing Domain Decomposition [Mandel, 1993]

On choisit : $\boldsymbol{u}_b^{\boxplus} = \boldsymbol{A}^{\boxminus T} \boldsymbol{u}_b$

$$\begin{cases} \boldsymbol{S}^{\boxtimes} \boldsymbol{u}_b^{\boxplus} = \boldsymbol{b}^{\boxplus} + \boldsymbol{\lambda}_b^{\boxplus} \\ \boldsymbol{A}^{\boxminus} \boldsymbol{\lambda}_b^{\boxplus} = \boldsymbol{0} \\ \boldsymbol{B}^{\boxminus} \boldsymbol{u}_b^{\boxplus} = \boldsymbol{0} \end{cases}$$

Eq. (3) vérifiée par construction $\boldsymbol{A}^{\boxminus} \times (1)$




## Système BDD

$$\boldsymbol{A}^{\boxminus} \boldsymbol{S}^{\boxtimes} \boldsymbol{A}^{\boxminus T} \boldsymbol{u}_b = \boldsymbol{A}^{\boxminus} \boldsymbol{b}^{\boxplus} + \boldsymbol{A}^{\boxminus} \boldsymbol{\lambda}_b^{\boxplus}$$

# Méthode Balancing Domain Decomposition [Mandel, 1993]

## Système BDD

$$\boldsymbol{A}^{\boxminus}\boldsymbol{S}^{\boxslash}{\boldsymbol{A}^{\boxminus}}^{T}\boldsymbol{u}_b = \boldsymbol{A}^{\boxminus}\boldsymbol{b}^{\boxplus} + \boldsymbol{A}^{\boxminus}\boldsymbol{\lambda}_b^{\boxplus}$$

$$\boldsymbol{S}\boldsymbol{u}_b = \boldsymbol{b} + \boldsymbol{A}^{\boxminus}\boldsymbol{\lambda}_b^{\boxplus}$$

En résolvant $\boldsymbol{S}\boldsymbol{u}_b = \boldsymbol{b}$, on a l'équilibre des efforts à convergence

Le résidu $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{S}\boldsymbol{u}_b$ est directement le saut d'effort à l'interface

## Préconditionnement

Preconditionneur de la forme $\boldsymbol{M}^{-1} = \tilde{\boldsymbol{A}}^{\boxminus}\boldsymbol{S}^{+\boxslash}\tilde{\boldsymbol{A}}^{\boxminus T}$

$\tilde{\boldsymbol{A}}^{s}$ correspond au scaling (cf $\delta_1, \delta_2$)

Le préconditionneur est appliqué au résidu

$$\boldsymbol{z}_j = \boldsymbol{M}^{-1}\boldsymbol{r}_j = \tilde{\boldsymbol{A}}^{\boxminus}\boldsymbol{S}^{+\boxslash}\tilde{\boldsymbol{A}}^{\boxminus T}\boldsymbol{r}_j$$

Le résidu doit être dans l'image de $\boldsymbol{S}^{\boxslash}$. Il faut garantir

$${\boldsymbol{R}_b}^{\boxslash T}\tilde{\boldsymbol{A}}^{\boxminus T}\boldsymbol{r}_j = \boldsymbol{G}^{T}\boldsymbol{r}_j = \boldsymbol{0}$$

## Contrainte sur les MCR gérée par initialisation/projection ($\boldsymbol{G}^T \boldsymbol{r}_j = \boldsymbol{0}$)

- Définition d'un projecteur $\boldsymbol{\Pi}$ tel que $\boldsymbol{G}^T \boldsymbol{S}_p \boldsymbol{\Pi} = \boldsymbol{0}$

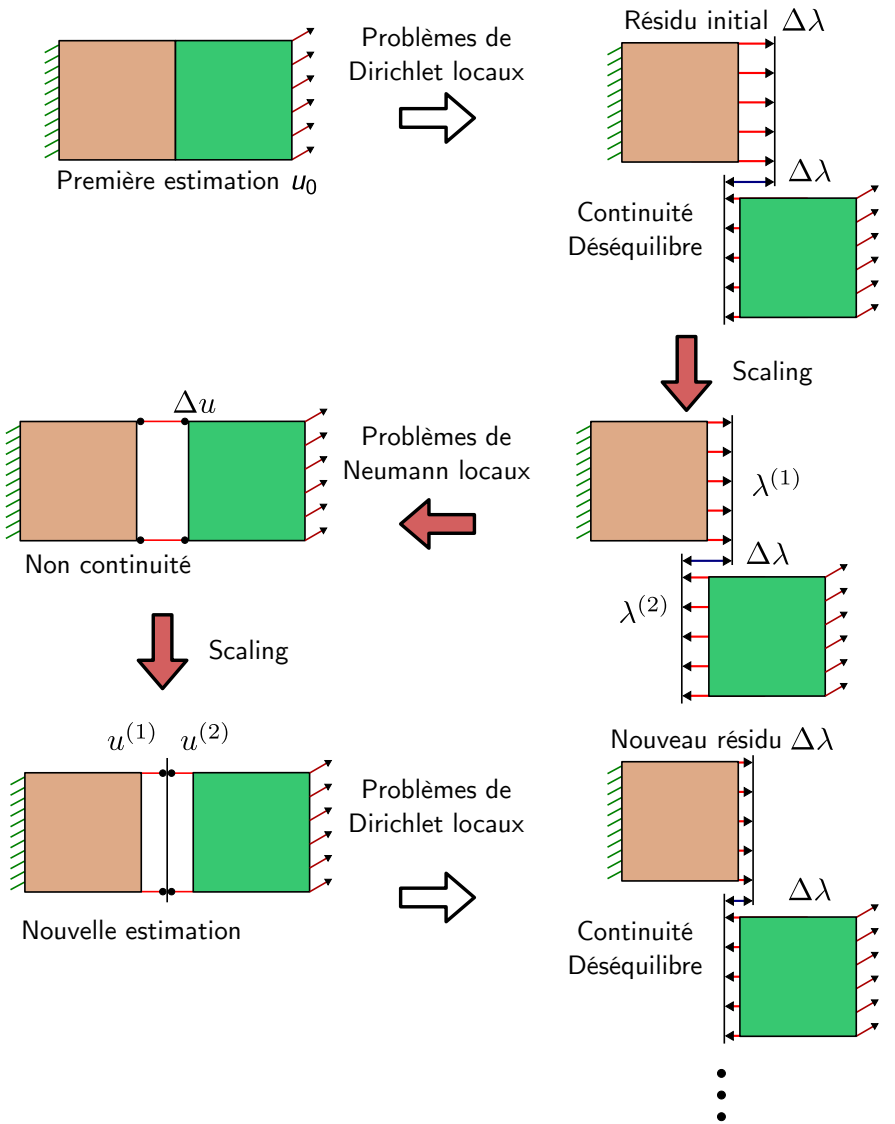
$$\begin{aligned} \boldsymbol{u}_b &= \boldsymbol{u}_0 + \boldsymbol{\Pi} \boldsymbol{u} \\ \boldsymbol{u}_0 &= \boldsymbol{G}(\boldsymbol{G}^T \boldsymbol{S}_p \boldsymbol{G})^{-1} \boldsymbol{G}^T \boldsymbol{b}_p \\ \boldsymbol{\Pi} &= \boldsymbol{I} - \boldsymbol{G}(\boldsymbol{G}^T \boldsymbol{S}_p \boldsymbol{G})^{-1} \boldsymbol{G}^T \boldsymbol{S}_p \end{aligned}$$

- La méthode BDD utilise un gradient conjugué augmenté

$$\begin{cases} \boldsymbol{u}_b \in \boldsymbol{u}_0 + \mathcal{K}_m + \mathrm{Vect}(\boldsymbol{G}) \\ \boldsymbol{r}_m \perp \mathcal{K}_m + \mathrm{Vect}(\boldsymbol{G}) \end{cases}$$

**Algorithme 3 :** BDD

$$u_0 = G(G^T S_p G)^{-1} G^T b_p$$
$$r_0 = b_p - S_p u_0 = \Pi^T b_p$$
$$z_0 = M^{-1} r_0$$
$$w_0 = z_0$$
**for** $i = 0 \ldots m$ **do**
$$q_i = S_p \Pi w_i$$
$$\alpha_i = (z_i, r_i)/(q_i, w_i)$$
$$u_{i+1} = u_i + \alpha_i w_i$$
$$r_{i+1} = r_i - \alpha_i q_i$$
$$z_{i+1} = M^{-1} r_{i+1}$$
$$w_{i+1} = z_{i+1}$$
$$w_{i+1} - = (z_{i+1}, q_i)/(w_i, q_i) w_i$$

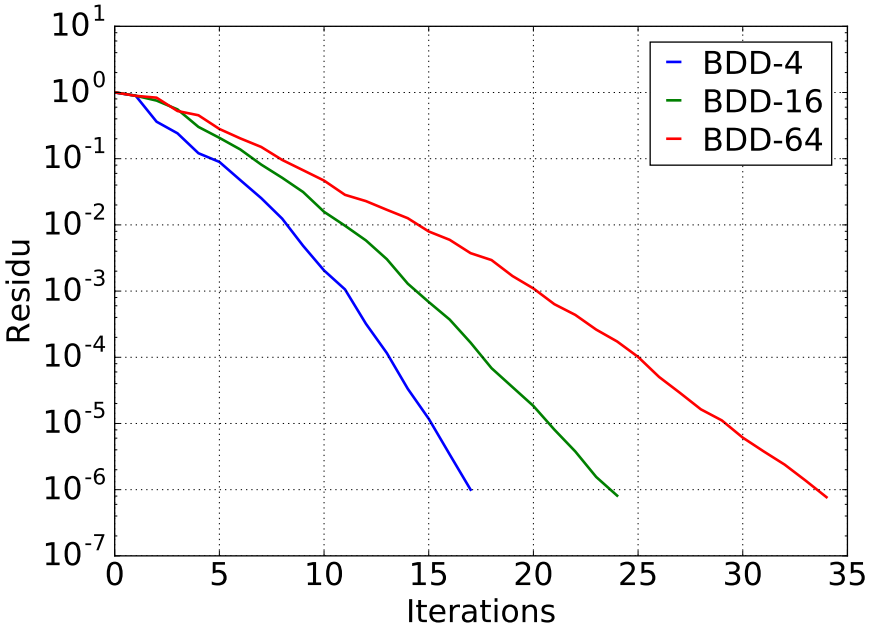


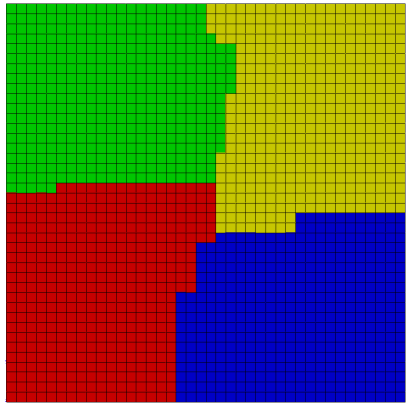
# Méthode Balancing Domain Decomposition [Mandel, 1993]

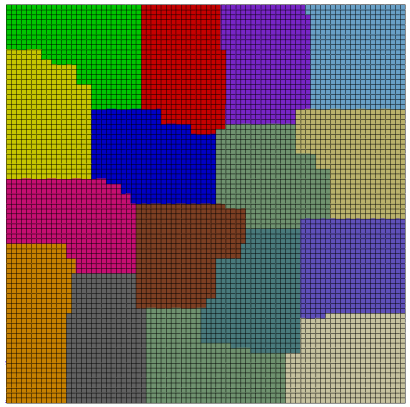
- Conditionnement du problème d'interface

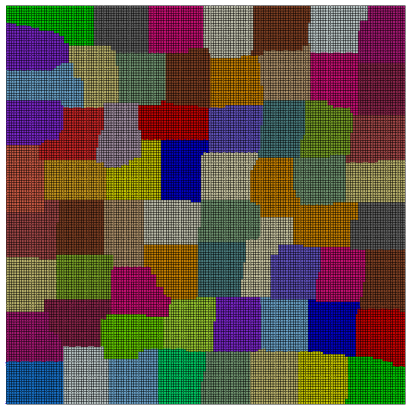
$$\kappa = O\left(\left(1 + \log \frac{H}{h}\right)^2\right)$$

- Extensibilité faible





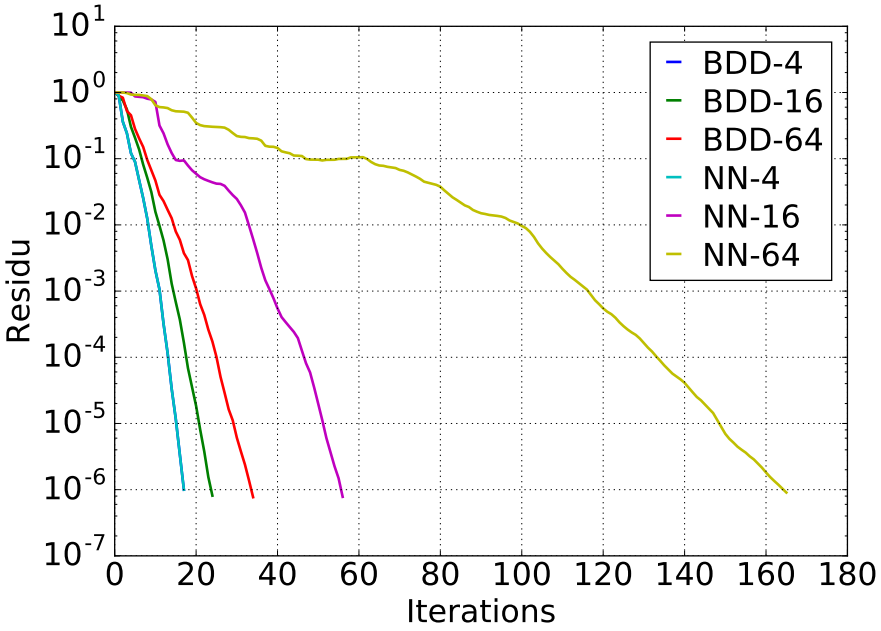


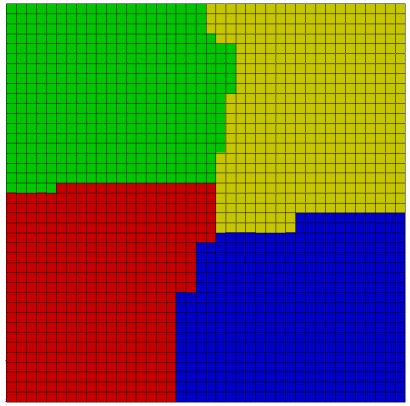
# Méthode Balancing Domain Decomposition [Mandel, 1993]

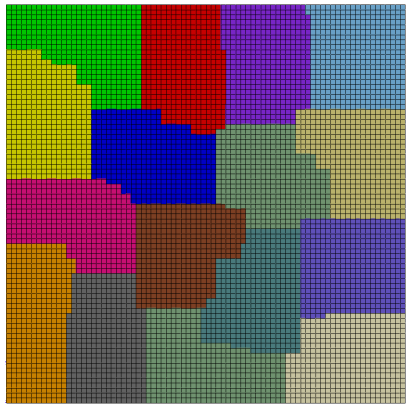
- Conditionnement du problème d'interface

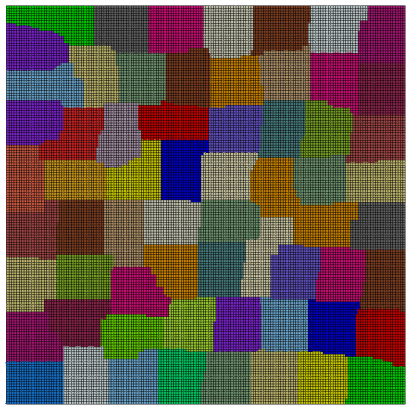
$$\kappa = O\left(\left(1 + \log \frac{H}{h}\right)^2\right)$$

- Extensibilité faible



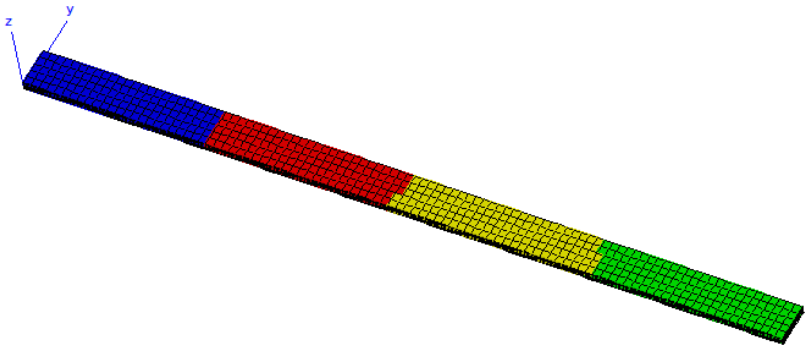







# Apport du problème grossier

- Exemple pour la poutre (BDD-step.z7p)




- Le solveur à grille grossière permet de propager rapidement l'information entre tous les sous-domaines

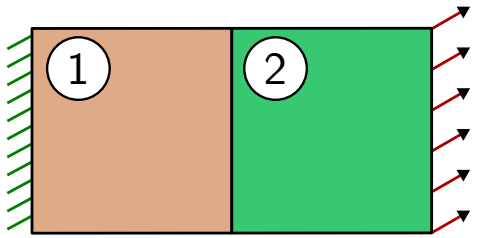
# Méthode Finite Element Tearing and Interconnecting

[Farhat and Roux, 1991]

- On choisit : $\lambda_b^{\square} = \boldsymbol{B}^{\boxminus T} \lambda_b$

$$\begin{cases} \boldsymbol{S}^{\boxtimes} \boldsymbol{u}_b^{\square} = \boldsymbol{b}^{\square} + \lambda_b^{\square} \\ \boldsymbol{A}^{\boxminus} \lambda_b^{\square} = \boldsymbol{0} \\ \boldsymbol{B}^{\boxminus} \boldsymbol{u}_b^{\square} = \boldsymbol{0} \end{cases}$$

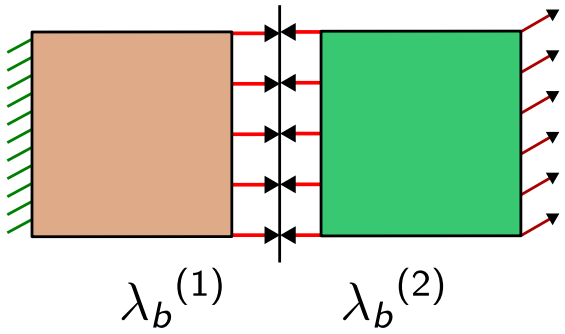
- L'équilibre d'interface est vérifié par construction
- Recherche itérative de l'effort d'interface qui annule le saut de déplacement




Problème init.

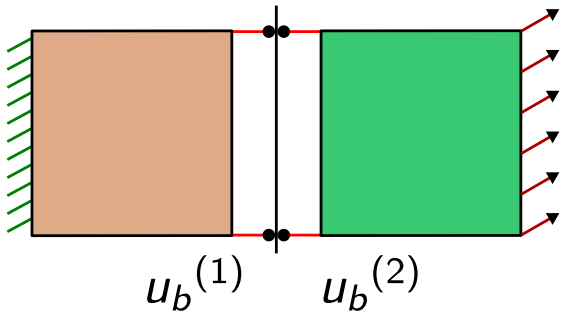
⇩

Décomposition




Équilibre

$\lambda_b^{(1)} = -\lambda_b^{(2)}$




Continuité

$u_b^{(1)} = u_b^{(2)}$

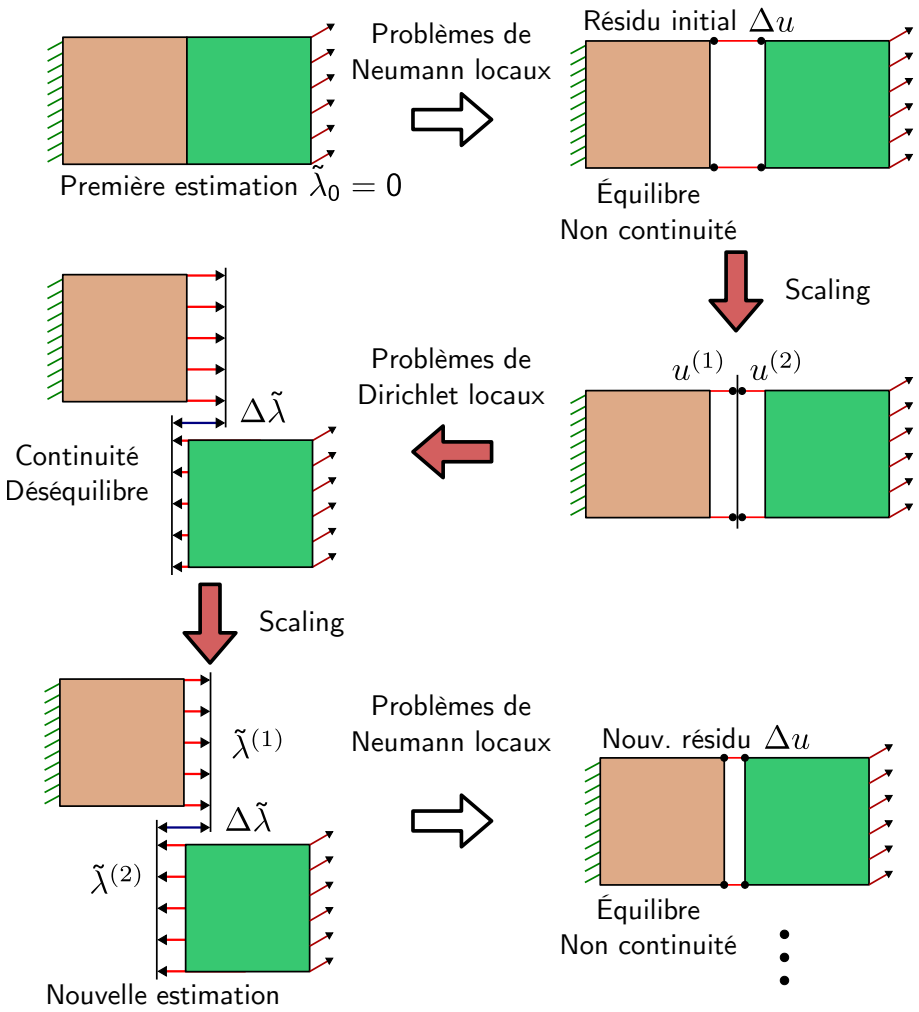
# Méthode Finite Element Tearing and Interconnecting

[Farhat and Roux, 1991]

- On choisit : $\lambda_b^{\square} = B^{\square\,T} \lambda_b$

$$\begin{cases} S^{\square} u_b^{\square} = b^{\square} + \lambda_b^{\square} \\ A^{\square} \lambda_b^{\square} = 0 \\ B^{\square} u_b^{\square} = 0 \end{cases}$$

- L'équilibre d'interface est vérifié par construction
- Recherche itérative de l'effort d'interface qui annule le saut de déplacement



# Méthode FETI [Farhat and Roux, 1991]

- On doit résoudre :

$$\begin{cases} \boldsymbol{S}^{\boxtimes}\boldsymbol{u}_b^{\boxplus} = \boldsymbol{b}^{\boxplus} + \boldsymbol{B}^{\boxminus\,T}\boldsymbol{\lambda}_b & \text{(équilibres locaux)} \\ \boldsymbol{B}^{\boxminus}\boldsymbol{u}_b^{\boxplus} = \boldsymbol{0} & \text{(continuité du déplacement)} \end{cases}$$

- Expression en fonction de $\boldsymbol{\lambda}_b$

$$\begin{cases} \boldsymbol{u}_b^{\boxplus} = \boldsymbol{S}^{+\boxtimes}\left(\boldsymbol{b}^{\boxplus} + \boldsymbol{B}^{\boxminus\,T}\boldsymbol{\lambda}_b\right) + \boldsymbol{R}^{\boxtimes}\alpha^{\boxplus} \\ \boldsymbol{0} = \boldsymbol{B}^{\boxminus}\boldsymbol{u}_b^{\boxplus} \\ \boldsymbol{0} = \boldsymbol{R}^{T\boxtimes}\left(\boldsymbol{b}^{\boxplus} + \boldsymbol{B}^{\boxminus\,T}\boldsymbol{\lambda}_b\right) & \text{(contrainte de solvabilité)} \end{cases}$$

- Comme pour BDD, la contrainte de solvabilité est due aux sous-domaines flottants
- On injecte (1) dans (2)

$$\begin{cases} \boldsymbol{B}^{\boxminus}\boldsymbol{u}_b^{\boxplus} = \boldsymbol{0} = \boldsymbol{B}^{\boxminus}\boldsymbol{S}^{+\boxtimes}\boldsymbol{B}^{\boxminus\,T}\boldsymbol{\lambda}_b + \boldsymbol{B}^{\boxminus}\boldsymbol{R}^{\boxtimes}\alpha^{\boxplus} + \boldsymbol{B}^{\boxminus}\boldsymbol{S}^{+\boxtimes}\boldsymbol{b}^{\boxplus} \\ \boldsymbol{0} = \boldsymbol{R}^{T\boxtimes}\boldsymbol{B}^{\boxminus\,T}\boldsymbol{\lambda}_b + \boldsymbol{R}^{T\boxtimes}\boldsymbol{b}^{\boxplus} \end{cases}$$

# Méthode FETI [Farhat and Roux, 1991]

$$\begin{cases} \boldsymbol{B}^{\boxminus}\boldsymbol{u}_b^{\boxplus} = \mathbf{0} = \boldsymbol{B}^{\boxminus}\boldsymbol{S}^{+\boxtimes}\boldsymbol{B}^{\boxminus\,T}\boldsymbol{\lambda}_b + \boldsymbol{B}^{\boxminus}\boldsymbol{R}^{\boxtimes}\alpha^{\boxplus} + \boldsymbol{B}^{\boxminus}\boldsymbol{S}^{+\boxtimes}\boldsymbol{b}^{\boxplus} \\ \mathbf{0} = \boldsymbol{R}^{T\boxtimes}\boldsymbol{B}^{\boxminus\,T}\boldsymbol{\lambda}_b + \boldsymbol{R}^{T\boxtimes}\boldsymbol{b}^{\boxplus} \end{cases}$$

- On obtient le système FETI

$$\begin{pmatrix} \boldsymbol{F} & \boldsymbol{G} \\ \boldsymbol{G}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_b \\ \alpha^{\boxplus} \end{pmatrix} = \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{e} \end{pmatrix}$$

avec

$$\boldsymbol{F} = \boldsymbol{B}^{\boxminus}\boldsymbol{S}^{+\boxtimes}\boldsymbol{B}^{T\boxminus} \quad \text{(opérateur FETI)}$$

$$\boldsymbol{G} = \boldsymbol{B}^{\boxminus}\boldsymbol{R}^{\boxtimes} \quad \text{(trace des MCRs)}$$

$$\boldsymbol{d} = -\boldsymbol{B}^{\boxminus}\boldsymbol{S}^{+\boxtimes}\boldsymbol{b}^{\boxplus} \quad (2^{nd} \text{ membre condensé})$$

$$\boldsymbol{e} = -\boldsymbol{R}^{T\boxtimes}\boldsymbol{b}^{\boxplus} \quad \text{(travail du chargement sur les noyaux)}$$

- La contrainte de solvabilité fournit un problème grossier naturel

# Méthode FETI [Farhat and Roux, 1991]

**Contrainte sur les MCR gérée par initialisation/projection ($\boldsymbol{G}^T\boldsymbol{\lambda} = \boldsymbol{e}$)**

- Définition d'un projecteur $\boldsymbol{\Pi}$ tel que $\boldsymbol{G}^T\boldsymbol{\Pi} = \boldsymbol{0}$

$$\boldsymbol{\lambda}_b = \boldsymbol{\lambda}_0 + \boldsymbol{\Pi}\tilde{\boldsymbol{\lambda}} \quad \text{avec} \quad \boldsymbol{\lambda}_0 \text{tel que} (\boldsymbol{G}^T\boldsymbol{\lambda}_0) = \boldsymbol{e}$$

- $\tilde{\boldsymbol{\lambda}}$ est recherchée à l'aide d'un solveur itératif (Gradient Conjugué)

$$\boldsymbol{\Pi}^T\boldsymbol{F}\boldsymbol{\Pi}\tilde{\boldsymbol{\lambda}} = \boldsymbol{\Pi}^T(\boldsymbol{d} - \boldsymbol{F}\boldsymbol{\lambda}_0)$$

# Préconditionnement

- Comme pour BDD et Neumann-Neumann, il faut trouver un bon préconditionneur
- L'opérateur FETI est une somme de compléments de Schur duaux locaux

$$\boldsymbol{F} = \boldsymbol{B}^{\boxminus} \boldsymbol{S}^{+\boxslash} \boldsymbol{B}^{\boxminus\, T}$$

- Un bon préconditionneur est une somme pondérée de compléments de Schur primaux

$$\boldsymbol{M}^{-1} = \tilde{\boldsymbol{B}}^{\boxminus} \boldsymbol{S}^{\boxslash} \tilde{\boldsymbol{B}}^{\boxminus\, T}$$

- Les matrices $\tilde{\boldsymbol{B}}$ représentent le *scalings* (cf. $\delta_1, \delta_2$)
- On peut approcher plus grossièrement $\boldsymbol{S}^s$ *(dirichlet, lumped, superlumped)*

# Méthode FETI [Farhat and Roux, 1991]

**Algorithme 4 :** FETI-I

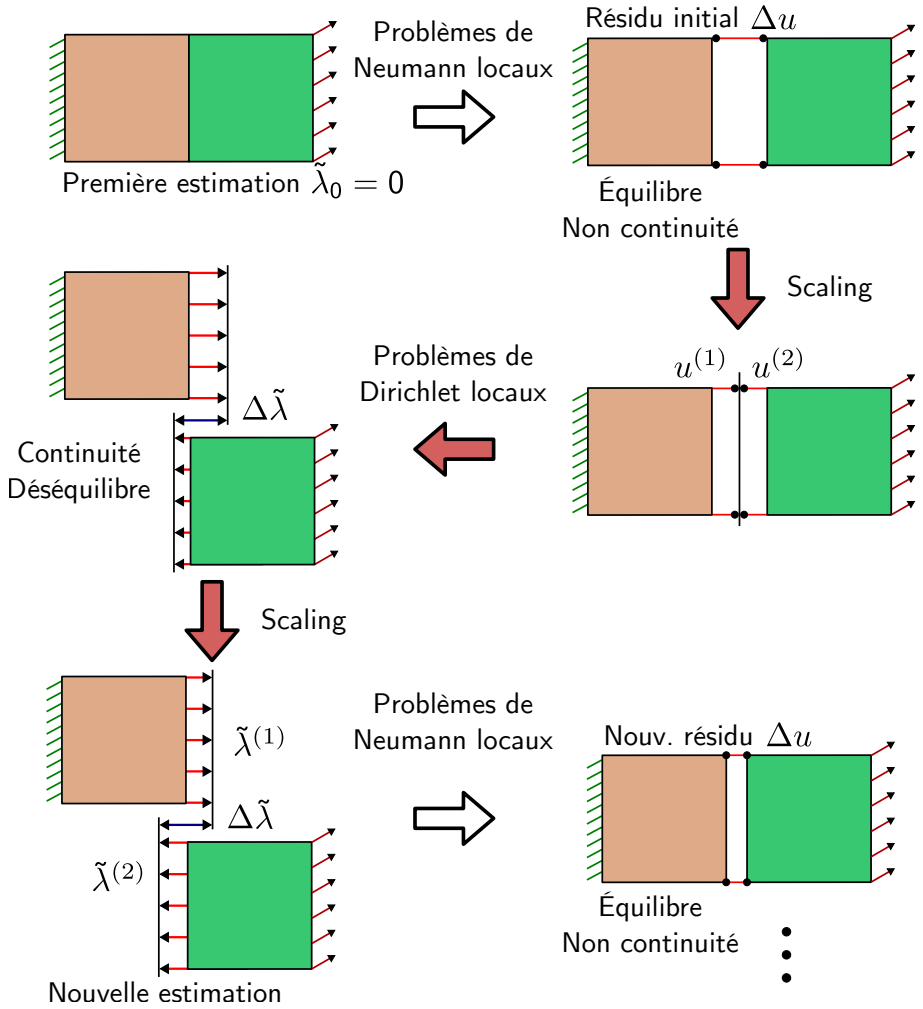
$r_0 = \Pi^T(d - F\lambda_0)$; $z_0 = M^{-1} r_0$

$w_0 = \Pi z_0$; $\tilde{\lambda}_0 = 0$; $i = 0$

**while** $\sqrt{r_i^T z_i}$ **do**

- $q_i = F w_i$
- $\delta_i = q_i^T w_i$
- $\gamma_i = z_i^T r_i$
- $\alpha_i = \gamma_i / \delta_i$
- $\tilde{\lambda}_{i+1} = \tilde{\lambda}_i + w_i \alpha_i$
- $r_{i+1} = r_i - \Pi^T q_i \alpha_i$
- $z_{i+1} = M^{-1} r_{i+1}$
- $w_{i+1} = \Pi z_{i+1}$
- **for** $0 \leq j \leq i$ **do**
  - $\Phi_{i,j} = q_j^T w_{i+1}$
  - $w_{i+1} \leftarrow w_{i+1} - (\Phi_{i,j}/\delta_j) w_j$
- $i \leftarrow i + 1$

$\lambda = \lambda_0 + \tilde{\lambda}_i$



# Méthode FETI [Farhat and Roux, 1991]

**Algorithme 5 :** FETI-I

$r_0 = \Pi^T(d - F\lambda_0)$; $z_0 = M^{-1}r_0$
$w_0 = \Pi z_0$; $\tilde{\lambda}_0 = 0$; $i = 0$
**while** $\sqrt{r_i^T z_i}$ **do**
  $q_i = Fw_i$ // Parallel Neumann solve
  $\delta_i = q_i^T w_i$ // MPI exchange
  $\gamma_i = z_i^T r_i$ // MPI exchange
  $\alpha_i = \gamma_i / \delta_i$
  $\tilde{\lambda}_{i+1} = \tilde{\lambda}_i + w_i \alpha_i$ // Solution update
  $r_{i+1} = r_i - \Pi^T q_i \alpha_i$ // Residual update
  $z_{i+1} = M^{-1} r_{i+1}$ // Parallel Dirichlet solve
  $w_{i+1} = \Pi z_{i+1}$
  **for** $0 \leq j \leq i$ **do**
   // Re-orthogonalization
    $\Phi_{i,j} = q_j^T w_{i+1}$ // MPI exchange
    $w_{i+1} \leftarrow w_{i+1} - (\Phi_{i,j}/\delta_j) w_j$
  $i \leftarrow i + 1$
$\lambda = \lambda_0 + \tilde{\lambda}_i$

## Remarque

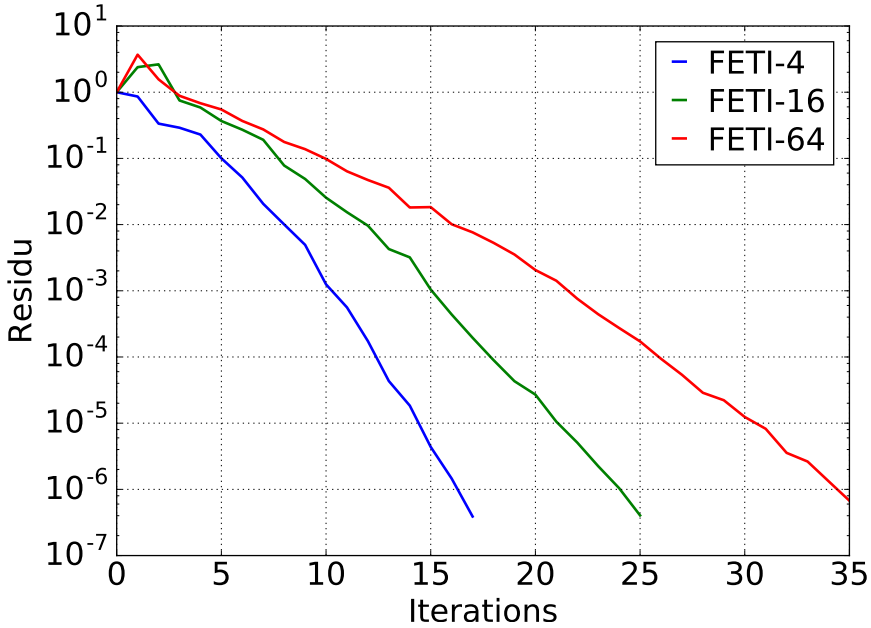
$\tilde{\lambda}_i$ solution
$r_i$ résidu
$z_i$ résidu précond.
$w_i$ dir. de recherche

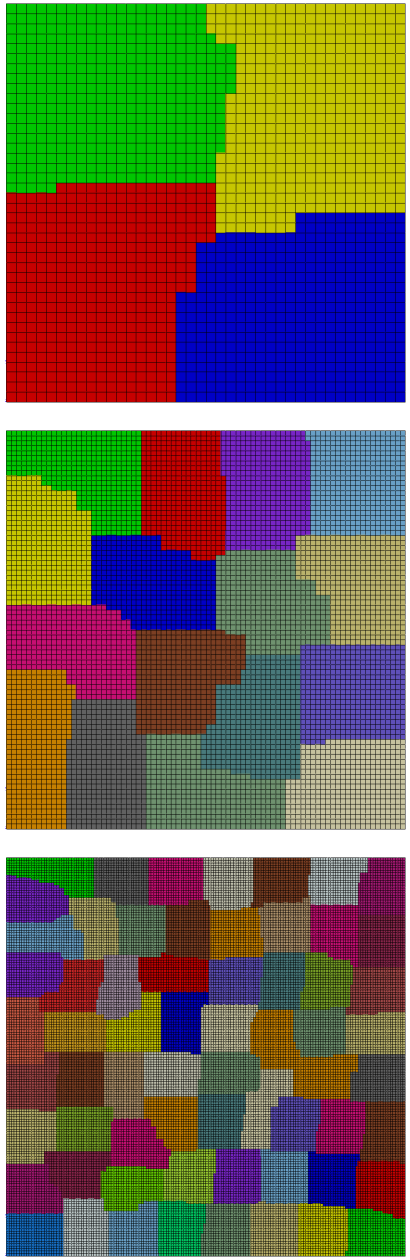
## Propriétés

$$(r_i, z_j) = 0 \ \forall \ i \neq j$$
$$(q_i, w_j) = 0 \ \forall \ i \neq j$$
$$(r_i, w_j) = 0 \ si \ i > j$$

# Méthode FETI [Farhat and Roux, 1991]

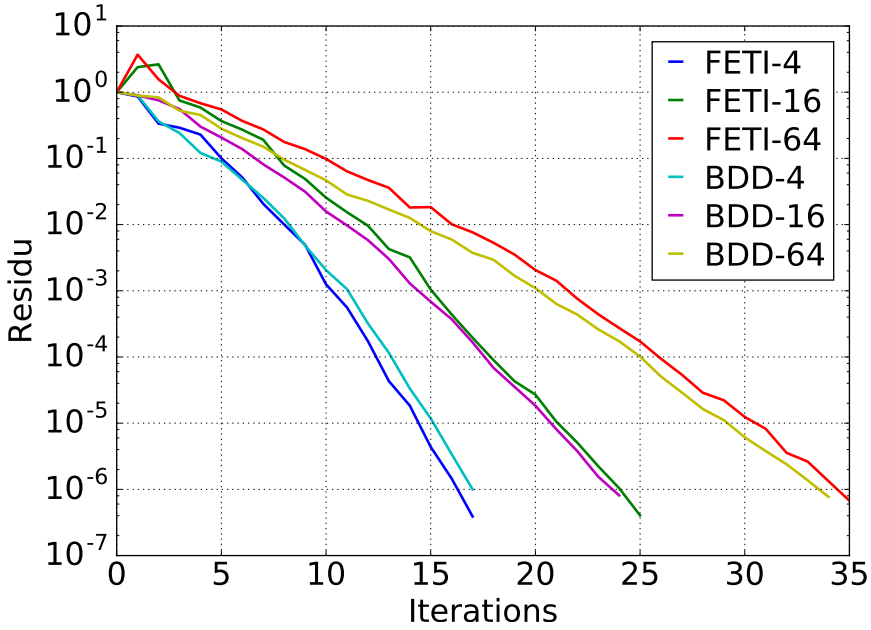
- Extensibilité faible

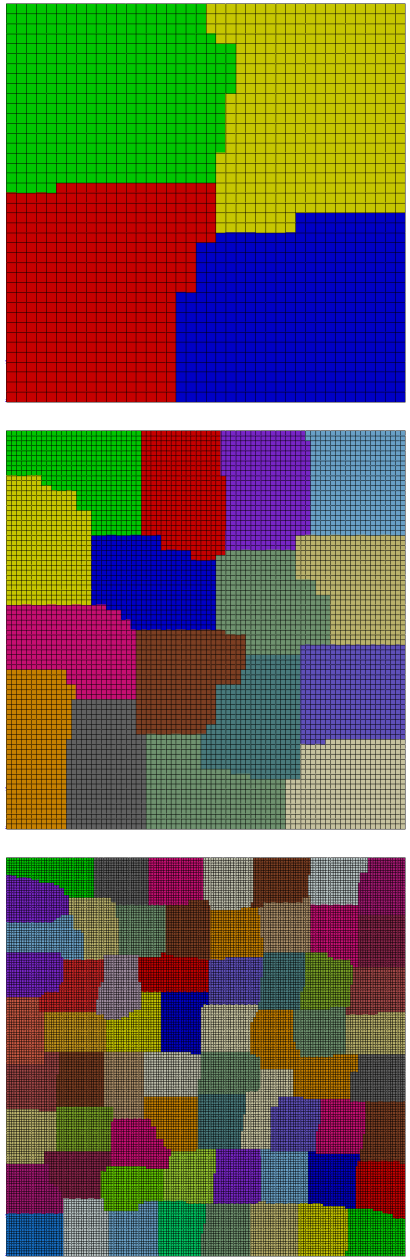


# Méthode FETI [Farhat and Roux, 1991]

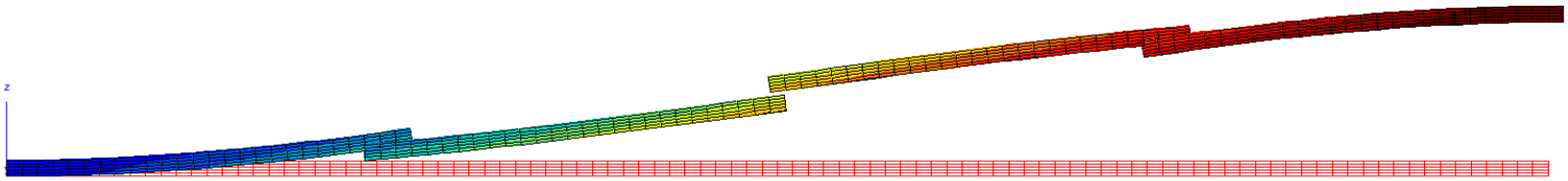
- Extensibilité faible



# Apport du problème grossier

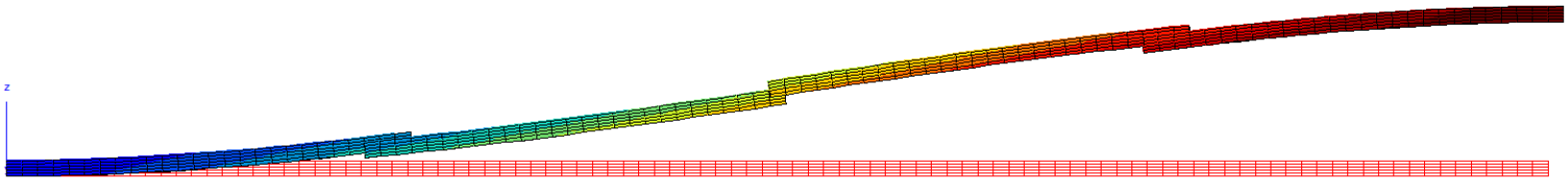
- Exemple pour la poutre (feti-step.z7p)
- Itération 0



- Avec la méthode FETI, les interfaces sont toujours équilibrées
- Le saut de déplacement diminue au cours des itérations

# Apport du problème grossier

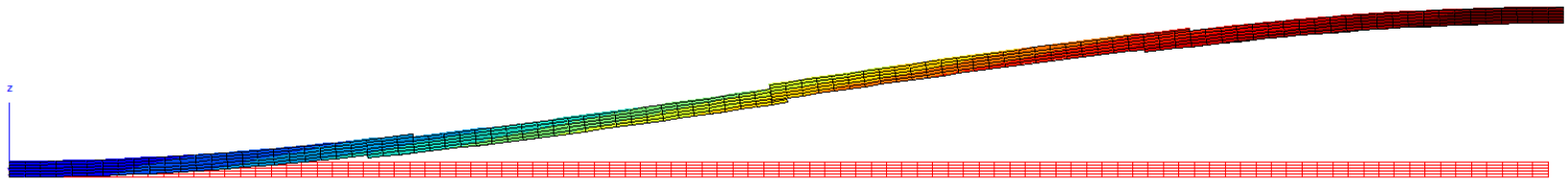
- Exemple pour la poutre (feti-step.z7p)
- Itération 1



- Avec la méthode FETI, les interfaces sont toujours équilibrées
- Le saut de déplacement diminue au cours des itérations

# Apport du problème grossier

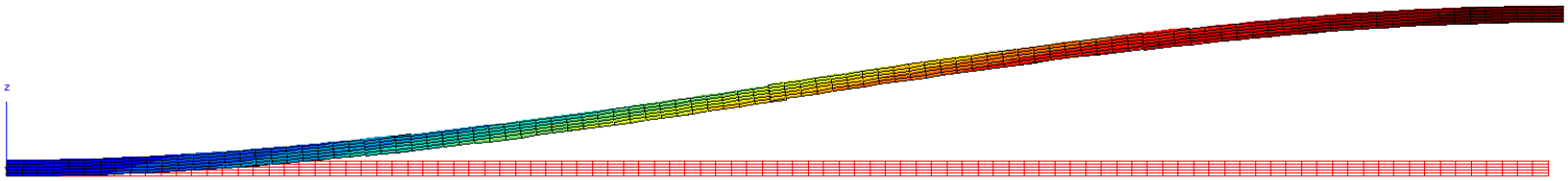
- Exemple pour la poutre (feti-step.z7p)
- Itération 2



- Avec la méthode FETI, les interfaces sont toujours équilibrées
- Le saut de déplacement diminue au cours des itérations

# Apport du problème grossier

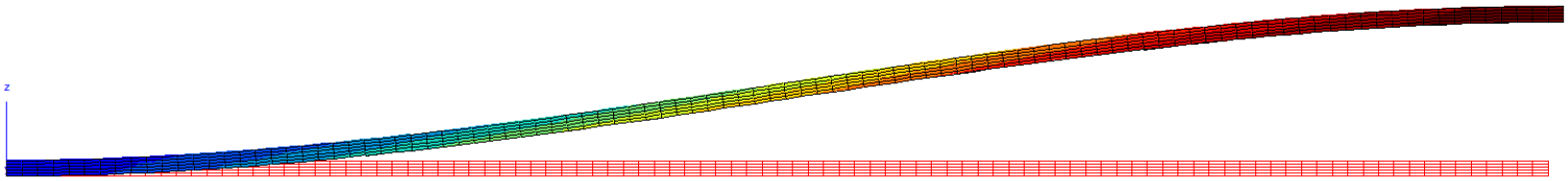
- Exemple pour la poutre (feti-step.z7p)
- Itération 3



- Avec la méthode FETI, les interfaces sont toujours équilibrées
- Le saut de déplacement diminue au cours des itérations

# Apport du problème grossier

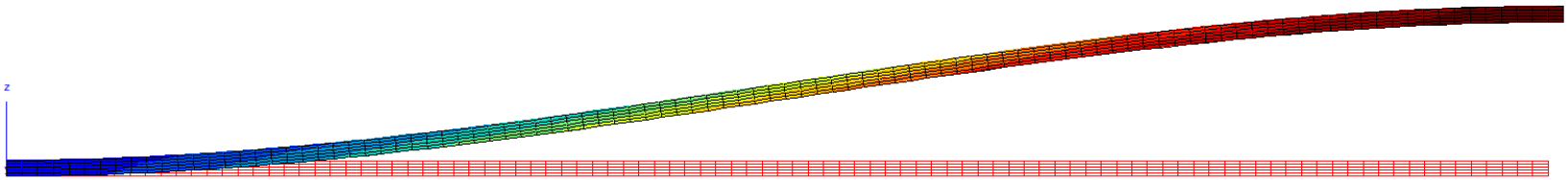
- Exemple pour la poutre (feti-step.z7p)
- Itération 4



- Avec la méthode FETI, les interfaces sont toujours équilibrées
- Le saut de déplacement diminue au cours des itérations

# Apport du problème grossier

- Exemple pour la poutre (feti-step.z7p)
- Itération 13

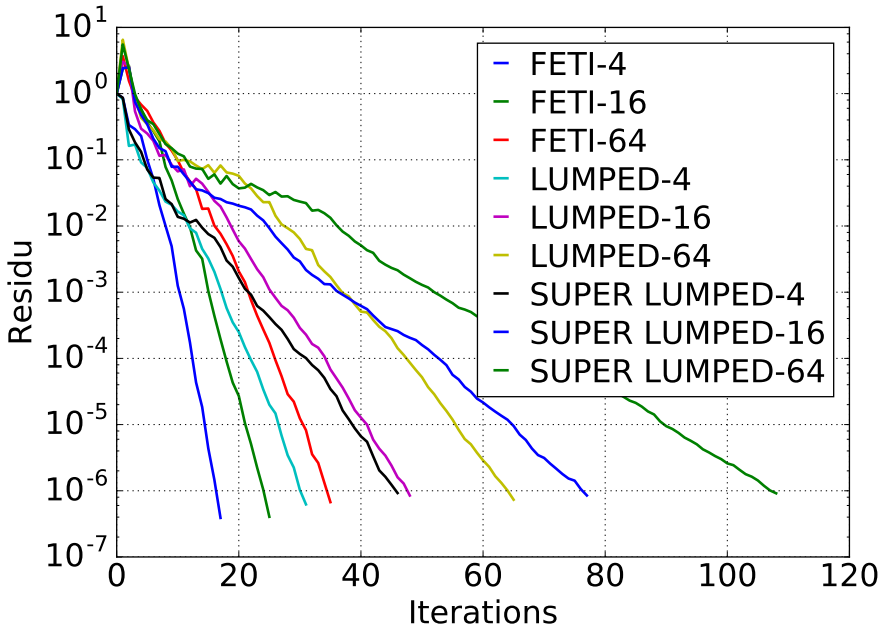


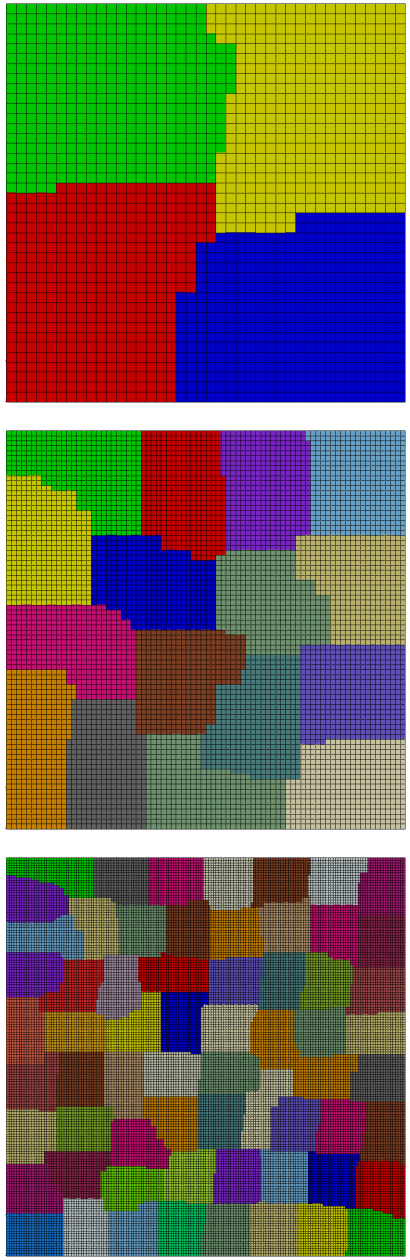
- Avec la méthode FETI, les interfaces sont toujours équilibrées
- Le saut de déplacement diminue au cours des itérations

# Méthode FETI [Farhat and Roux, 1991]

- Avantage de la méthode FETI le choix du préconditionneur :

| | |
|---|---|
| de Dirichlet | $\boldsymbol{S}^s = \boldsymbol{K}^s_{bb} - \boldsymbol{K}^s_{bi}\boldsymbol{K}^{s\,-1}_{ii}\boldsymbol{K}^s_{ib}$ |
| *lumped* | $\tilde{\boldsymbol{S}}^s = \boldsymbol{K}^s_{bb}$ |
| *super lumped* | $\tilde{\boldsymbol{S}}^s = \mathrm{diag}(\boldsymbol{K}^s_{bb})$ |

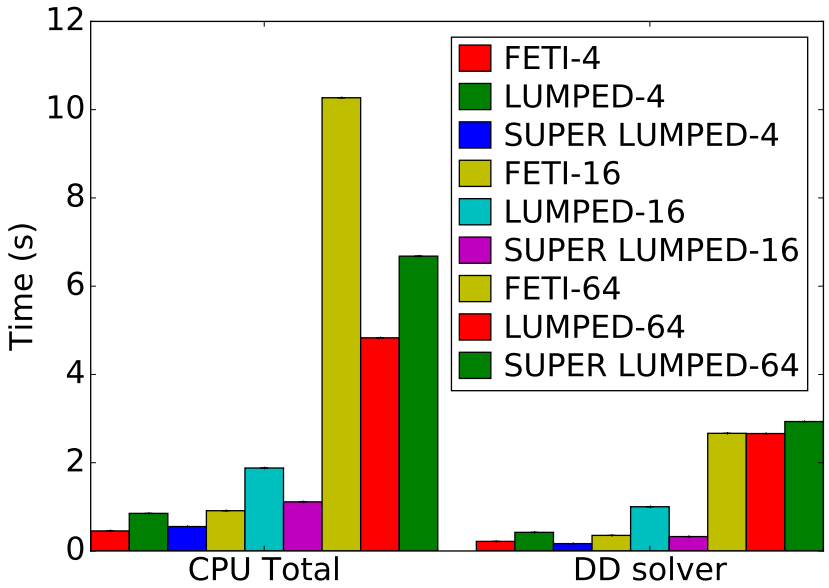


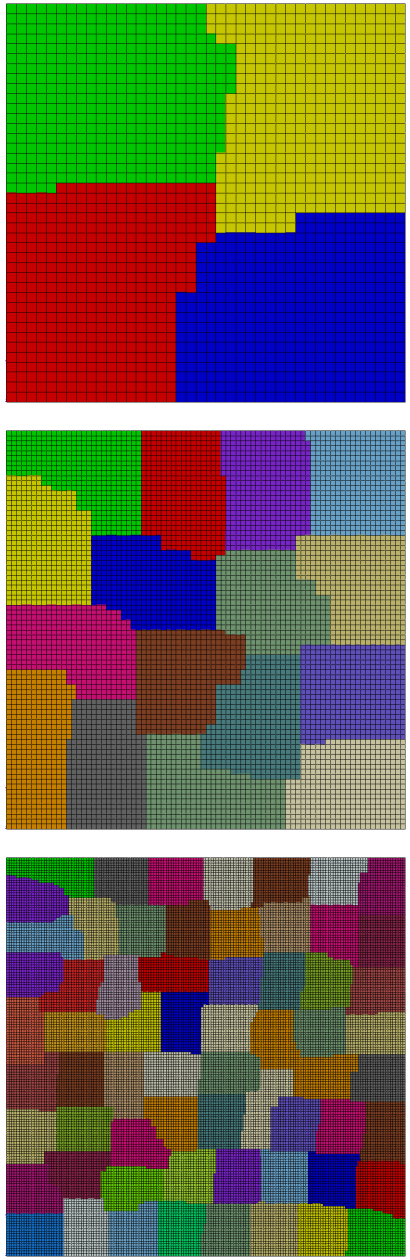



# Méthode FETI [Farhat and Roux, 1991]

- Avantage de la méthode FETI le choix du préconditionneur :

| | |
|---|---|
| de Dirichlet | $\boldsymbol{S}^s = \boldsymbol{K}^s_{bb} - \boldsymbol{K}^s_{bi}\boldsymbol{K}^{s\,-1}_{ii}\boldsymbol{K}^s_{ib}$ |
| *lumped* | $\tilde{\boldsymbol{S}}^s = \boldsymbol{K}^s_{bb}$ |
| *super lumped* | $\tilde{\boldsymbol{S}}^s = \mathrm{diag}(\boldsymbol{K}^s_{bb})$ |



# Sommaire

1 Introduction

2 Rappels sur la résolution de systèmes linéaires

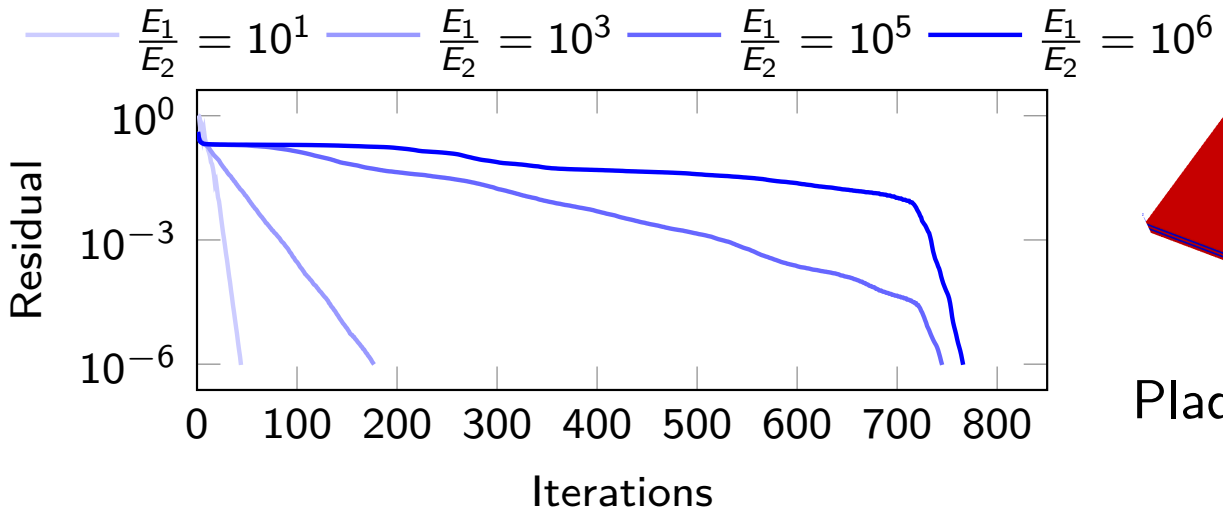
3 Méthodes de type Schur

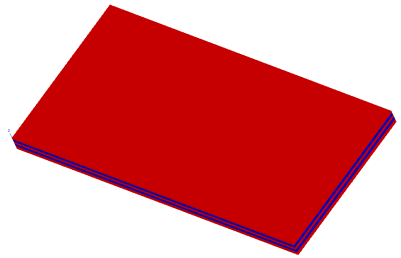
4 Problèmes ouverts et connexes
- Robustesse des méthodes DD
- Passage à l'échelle sur cas industriel

# Amélioration de la robustesse des méthodes DD

- Les méthodes DD sont capables de résoudre de très grands problèmes
- Elles restent sensibles au conditionnement du problème à résoudre (découpage auto , hétérogénéité, incompressibilité, ...)
- Conditionnement du problème d'interface FETI & BDD

$$\kappa = O\left(\left(1 + \log \frac{H}{h}\right)^2\right)$$



Plaque hétérogène

# Amélioration de la robustesse des méthodes DD

L'idée est d'augmenter la taille de l'espace de recherche

## Approches existantes

- Augmentation (problème grossier supplémentaire bien choisi)
  [Mandel and Sousedík, 2007, Klawonn et al., 2016,
  Spillane and Rixen, 2013, Spillane et al., 2014, Kim et al., 2016]
- Solveurs de Krylov multipréconditionnés
  [Bridson and Greif, 2006, Greif et al., 2011, Greif et al., 2012,
  Spillane, 2016, Gosselet et al., 2015]
- Solveurs de Krylov à seconds membres multiples

## Adaptive Multipreconditioned FETI

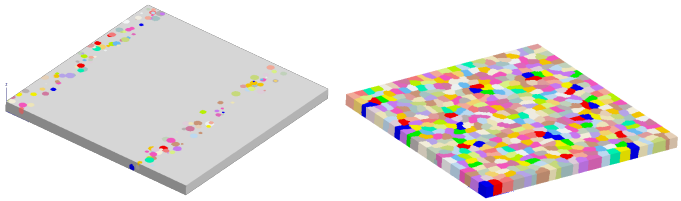
- Voir [Gosselet et al., 2015, Spillane et al., 2014, Bovet et al., 2017]
- Idée de base : perte d'information lors du préconditionnement $\boldsymbol{z}_i = \boldsymbol{M}^{-1}\boldsymbol{r}_i = \sum_s \tilde{\boldsymbol{B}}^s \tilde{\boldsymbol{S}}^s \tilde{\boldsymbol{B}}^{sT} \boldsymbol{r}_i$
- Laisser les contributions locales à part $\Rightarrow$ croissance rapide de l'espace de recherche ($\propto N_d$)

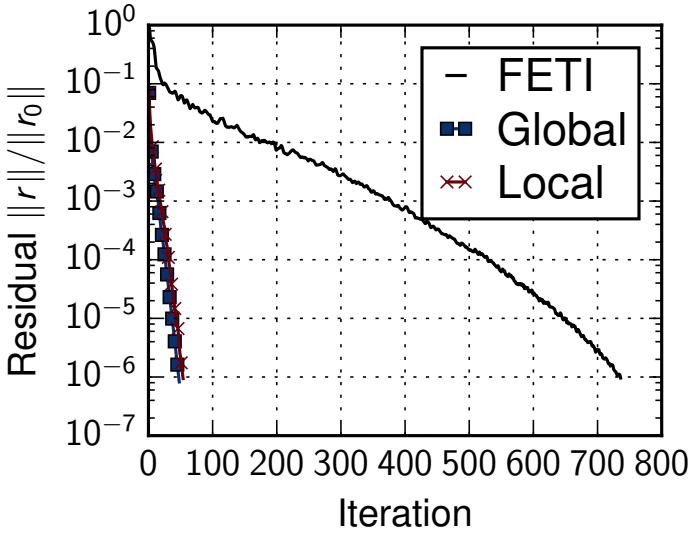
$$\boldsymbol{Z}_i = \left( \ldots, \tilde{\boldsymbol{B}}^s \hat{\boldsymbol{S}}^s \tilde{\boldsymbol{B}}^{sT} \boldsymbol{r}_i, \ldots \right)$$

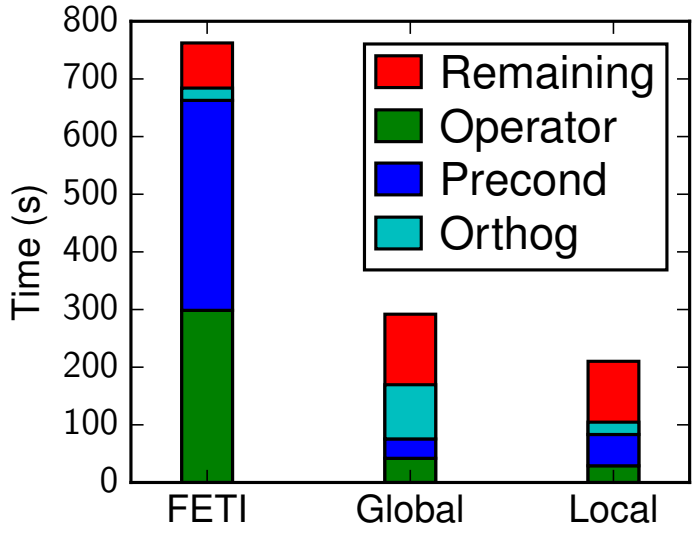
- Utilisation d'un Gradient conjugué multipréconditionné
- Un procédé d'adaptativité permet de garder uniquement les contributions utiles à la convergence

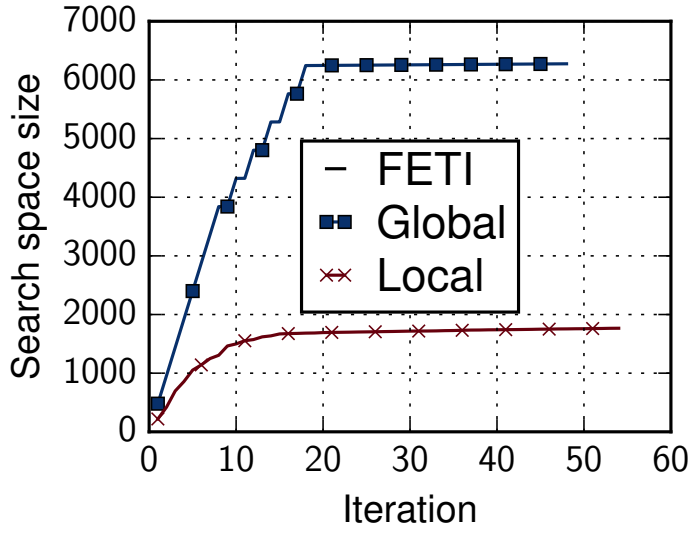
# Amélioration de la robustesse des méthodes DD

- Heterogeneous plate (400 stiff inclusions, linear elasticity)
- Test parameters $\tau = 0.01$, $E_{max}/E_0 \leq 10^5$, 480 subdomains, 32 Mdofs (Stelvio cluster, $4Gb/core$)



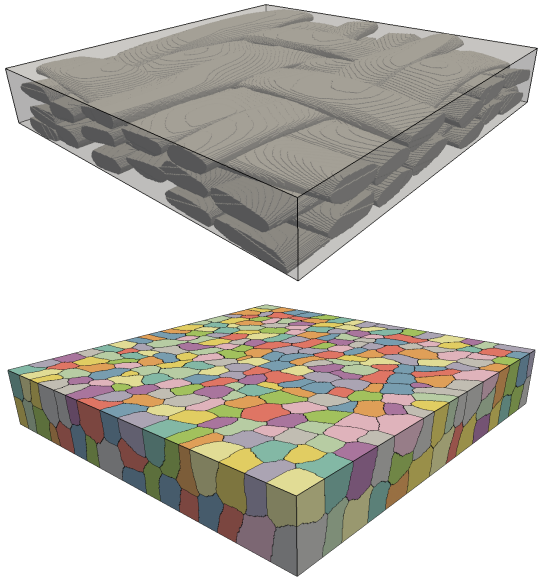


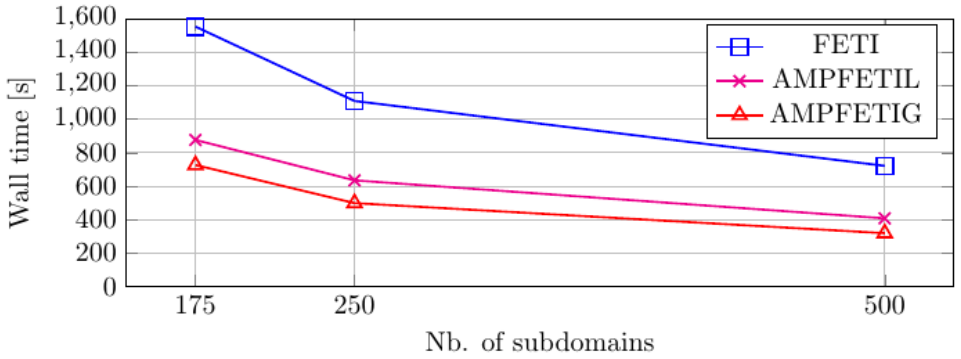



# Amélioration de la robustesse des méthodes DD

- Résolution de problèmes de complexité industrielle
- Étude d'extensibilité forte (100Mddls)



Source [Bovet et al., 2017]




(a) Total wall time

| $N = 175$ | # iter. | Time (s) |
|---|---|---|
| FETI | 371 | 1553.00 |
| AMPFETIL | 103 | 876.30 |
| AMPFETIG | 120 | 726.60 |
| $N = 250$ | # iter. | Time (s) |
| FETI | 405 | 1109.00 |
| AMPFETIL | 103 | 635.10 |
| AMPFETIG | 121 | 499.10 |
| $N = 500$ | # iter. | Time (s) |
| FETI | 544 | 721.60 |
| AMPFETIL | 104 | 408.20 |
| AMPFETIG | 122 | 319.52 |

## Ajout d'un espace d'augmentation bien choisi

$$\begin{cases} \boldsymbol{x}_m \in \boldsymbol{x}_O + \mathcal{K}_m + \mathcal{C} \\ \boldsymbol{r}_m \perp_? \mathcal{K}_m + \mathcal{C} \end{cases}$$

- Différentes approches possibles [Mandel and Sousedík, 2007, Klawonn et al., 2016, Spillane and Rixen, 2013, Spillane et al., 2014]
- Espace optimal $\mathcal{C} = \text{range}(\boldsymbol{C})$ : $\overrightarrow{vps}$ associées aux vps maximales de

$$\boldsymbol{\Pi}^T \left( \boldsymbol{F}\boldsymbol{v} - \mu \tilde{\boldsymbol{S}}^{-1} \boldsymbol{v} \right) = 0, \quad \boldsymbol{v} \in \text{range}(\boldsymbol{\Pi})$$

- Problèmes aux vps généralisées locaux (Geneo) [Spillane et al., 2014, Spillane and Rixen, 2013]

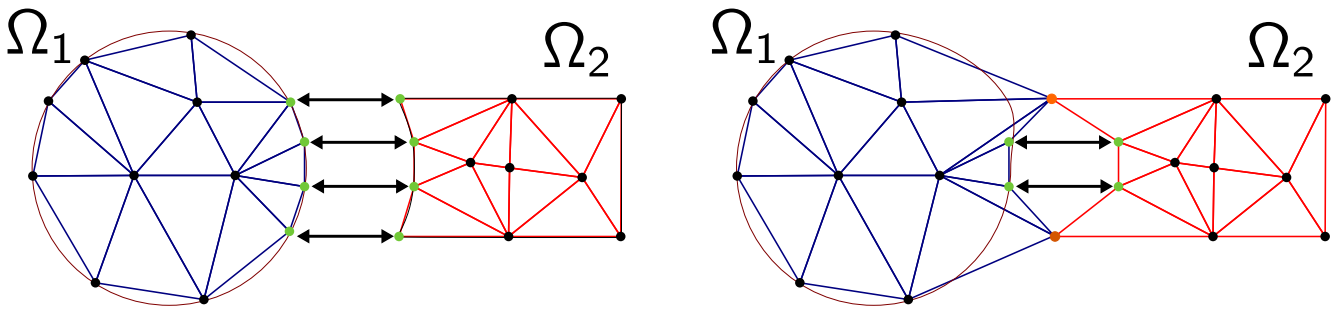
$$\boldsymbol{S}^s \boldsymbol{v}^s - \mu^s \tilde{\boldsymbol{B}}^{sT} \tilde{\boldsymbol{S}} \tilde{\boldsymbol{B}}^s \boldsymbol{v}^s = \boldsymbol{0} \qquad \boldsymbol{C} = \{ \boldsymbol{\Pi} \tilde{\boldsymbol{S}} \boldsymbol{B}^s \boldsymbol{v}^s ;\ 0 < \mu^s < \tau \}$$

# Problématiques

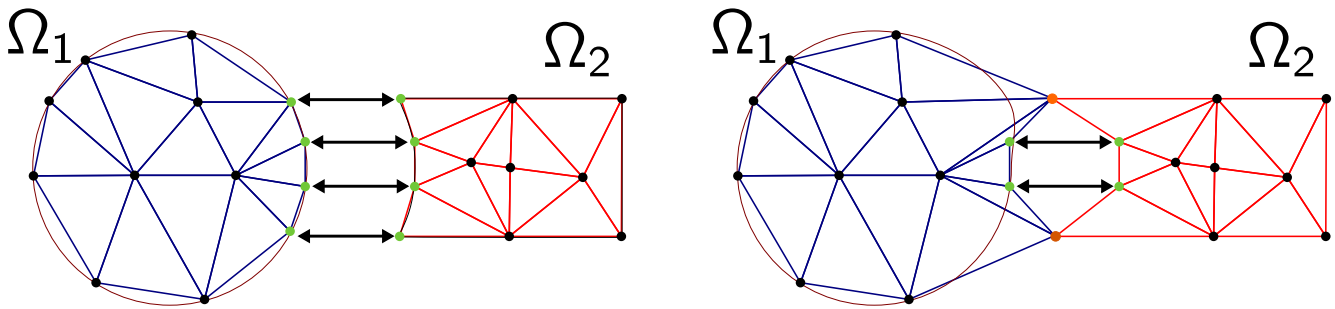
- Appliquer ces méthodes sur des cas industriels de grande taille fait apparaître de nouvelles problématiques :
  1. Les cas industriels sont souvent mal conditionnés et la robustesse des méthodes classiques (FETI, BDD) n'est parfois pas suffisante (convergence lente ou stagnation)
  2. Le problème grossier (indispensable pour avoir une méthode extensible) peut devenir un goulot d'étranglement lorsque le nombre de processeurs est très élevé.
- Il existe des remèdes
  1. Réduction de la taille du problème grossier par hybridation

# Contrôle de la taille du problème grossier

- Pour les méthodes FETI et BDD, la taille du problème grossier est déterminée par les noyaux des opérateurs locaux
- En supposant les sous-domains connexes, un ordre de grandeur est :
  - Mécanique des solides en statique $3 \times N_s$ (2D), $6 \times N_s$ (3D)
  - Thermique stationnaire $N_s$
- La résolution du problème grossier peut dominer le temps de calcul si $N_s \gg 1$ et les sous-domaines sont petits.
- La méthode FETI-DP [Farhat et al., 2001] permet de réduire la taille du problème grossier



# Méthode FETI-Dual Primal [Farhat et al., 2001]




- Idée de base : imposer la continuité de la quantité primale ($\boldsymbol{u}$) sur certains noeuds de l'interface appelé *corners*
- Grâce à ces *corners*, les opérateurs n'ont plus de noyau
- Imposer la continuité de la quantité primale sur ces *corners* mènent à un problème grossier naturel
- Comparé à la méthode FETI, la taille du problème grossier est approx. divisée par 2.
- Le choix des corners n'est pas toujours aisé ...
- Il existe l'équivalent pour la méthode BDD (BDDC [Dohrmann, 2003])

Exemple de résultats [Klawonn and Rheinbach, 2010]

| **Elasticity 2D – FETI-DP** ($H/h = 160$) | | | | | | | |
|---|---|---|---|---|---|---|---|
| #Cores | $N$ | D.o.f. | It. | Time | Speedup | Ideal Speedup | Efficiency |
| | | | Assembly + Solve | | | | |
| 16 | 16 | 821 762 | 14 | 35s | 1.0 | 1 | 100% |
| 64 | 64 | 3 281 922 | 21 | 38s | 3.7 | 4 | 92% |
| 256 | 256 | 13 174 442 | 22 | 38s | 14.7 | 16 | 92% |
| 1 024 | 1 024 | 52 449 282 | 23 | 39s | 57.4 | 64 | 90% |
| 4 096 | 4 096 | 209 756 162 | 24 | 41s | 218.5 | 256 | 85% |
| 16 384 | 16 384 | 838 942 722 | $\infty$ | $\infty$ | **0.0** | 1 024 | **0**% |
| | | | Solve | | | | |
| 16 | | | | 21s | 1.0 | 1 | 100% |
| 64 | | | | 24s | 3.5 | 4 | 88% |
| 256 | see | above | | 24s | 14.0 | 16 | 88% |
| 1 024 | | | | 25s | 53.8 | 64 | 84% |
| 4 096 | | | | 26s | 206.8 | 256 | 81% |
| 16 384 | | | | $\infty$ | **0.0** | 1 024 | **0**% |

TABLE 7.1

*FETI-DP on JUGENE for 2D linear elasticity without coefficient jumps on* $[0,1]^2$*, discretized with linear finite elements. Stopping criterion* $\mathrm{rtol} = 1\mathrm{e}-6$*. Upper: Assembly + Solver Time. Lower: Solver Time. The solution phase fails on 16 384 cores.*

# Hybrid Total FETI [Ríha et al., 2016, Markopoulos et al., 2018]

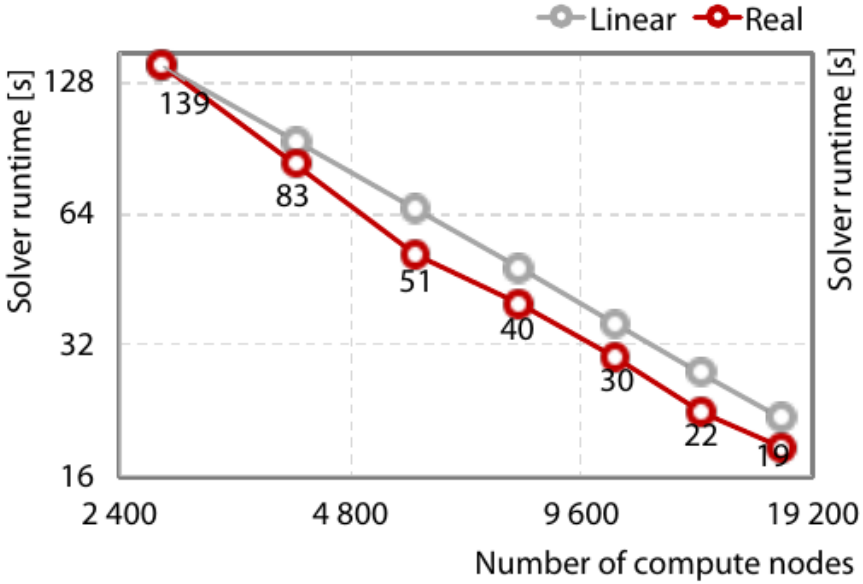
- Total FETI = toutes les conditions aux limites de type Dirichlet sont imposées par multiplicateur de Lagrange
- Hybrid = regrouper les sous-domaines en clusters
- Stratégie FETI-DP (par corner ou raccord en moyenne) dans les clusters
- Méthode FETI classique entre les clusters
- Avec cette approche, les sous-domaines dans un même cluster n'ont pas de mouvement de corps rigide entre eux
- Problème grossier à deux échelles :
  - $N_c$ problèmes grossiers locaux (= internes aux clusters)
  - Un problème grossier de taille $6N_c \times 6N_c$ (élasticité 3D) couple les clusters

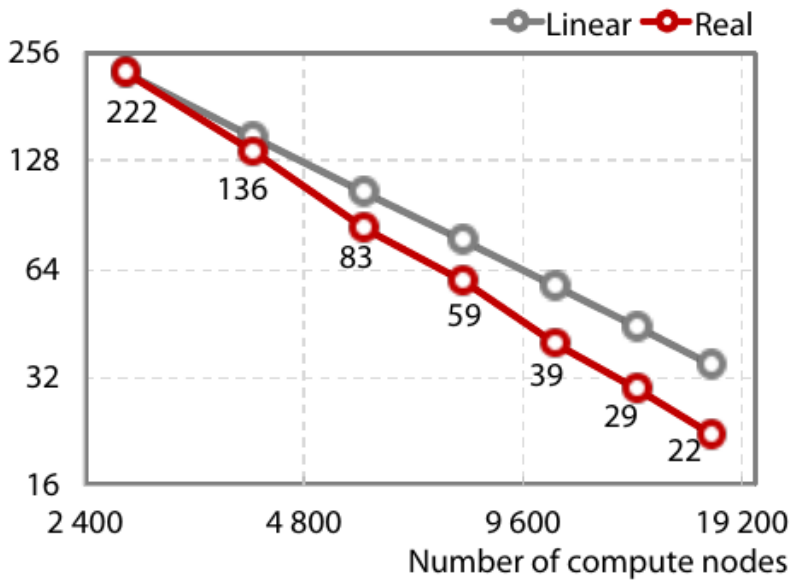
# Hybrid Total FETI (résultats)

Exemple de résultats [Ríha et al., 2016]

Test performed on ORNL Titan (18,688 compute nodes)

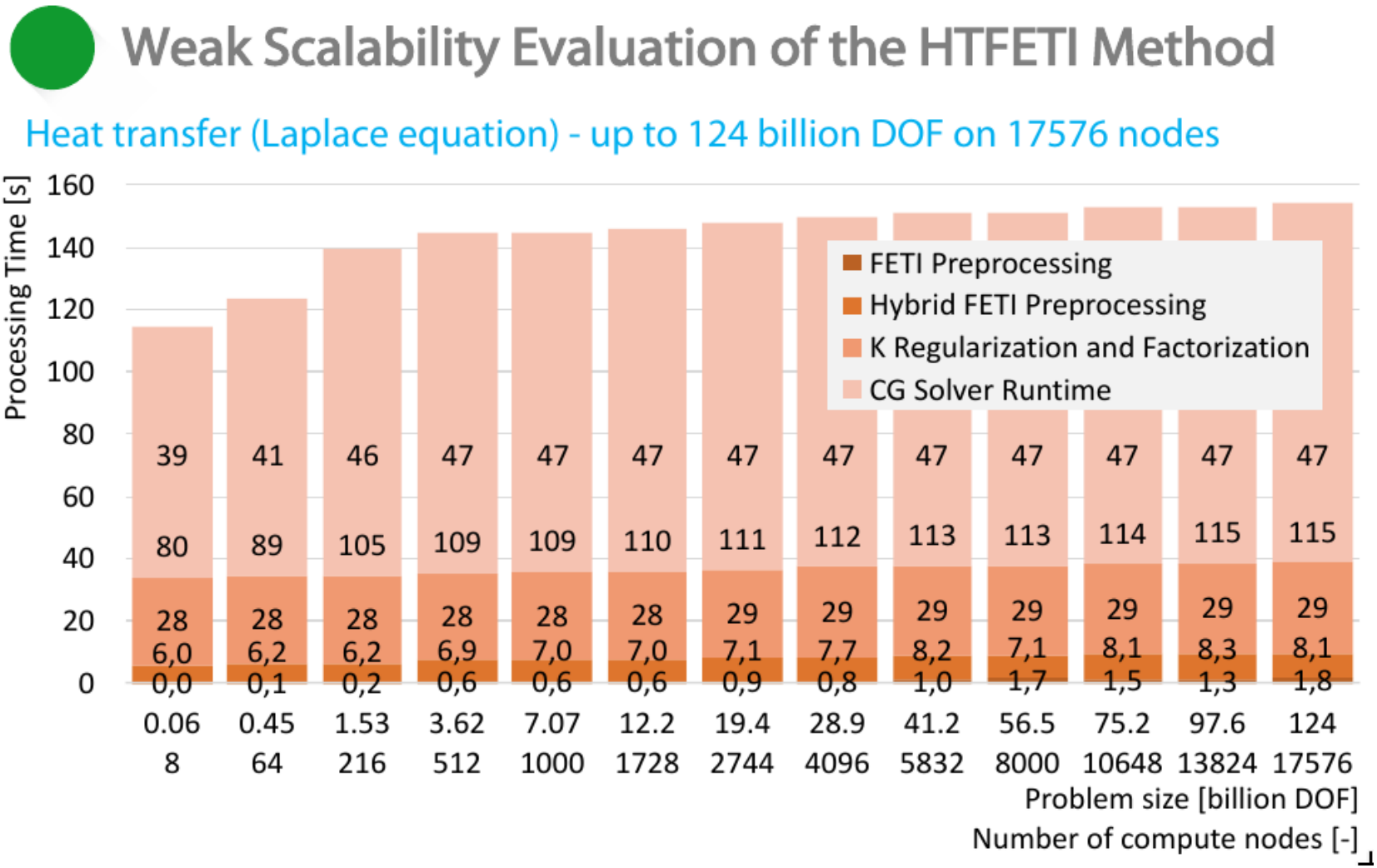
Structural mechanics - 11 billion DOF on up to 17 576 nodes (281 216 cores)
and Heat transfer (Laplace equation) - 20 billion DOF on up to 17 576 nodes



# Hybrid Total FETI (résultats)

Exemple de résultats [Ríha et al., 2016]



# Liens pratiques

- J'ai besoin de faire des calculs parallèles ?
- Solveurs en décomposition de domaine open-source
  - PETSc `https://www.mcs.anl.gov/petsc/`
  - hpddm `https://github.com/hpddm`
  - ESPRESO `http://espreso.it4i.cz/`
- Solveurs directs parallèles open-source
  - MUMPS `http://mumps.enseeiht.fr/`
  - SuperLU `http://crd-legacy.lbl.gov/~xiaoye/SuperLU/`
  - PaStiX `http://pastix.gforge.inria.fr/`
- Site web de la communauté : `http://www.ddm.org/`

# Un peu de lecture

- Introduction aux méthodes de décomposition de domaine
  [Gander and Halpern, 2012]
- Présentation théorique des méthodes DD (Schwarz & Schur)
  [Toselli and Widlund, 2005] [Pechstein, 2012]
- Approche plus mécanicienne [Gosselet and Rey, 2006]
- Papiers fondateurs
  FETI [Farhat and Roux, 1991]
  BDD [Mandel, 1993]
- Complément sur les solveurs de Krylov [Saad, 2003]

Bovet, C., Parret-Fréaud, A., Spillane, N., and Gosselet, P. (2017).
Adaptive multipreconditioned FETI : Scalability results and robustness assessment.
*Computers & Structures*, 193 :1–20.

Bridson, R. and Greif, C. (2006).
A multipreconditioned conjugate gradient algorithm.
*SIAM J. Matrix Anal. Appl.*, 27(4) :1056–1068 (electronic).

Dohrmann, C. R. (2003).
A preconditionner for substructuring based on constrained energy minimization.
*SIAM Journal for Scientific Computing*, 25 :246.

Dryja, M. and Hackbusch, W. (1997).
On the nonlinear domain decomposition method.
*BIT Numerical Mathematics*, 37(2) :296–311.

Farhat, C., Lesoinne, M., LeTallec, P., Pierson, K., and Rixen, D. (2001).
FETI-DP : a Dual-Primal Unified FETI Method - Part I : a Faster Alternative to the Two-Level FETI Method.
*International Journal for Numerical Methods in Engineering*, 50(7) :1523–1544.

Farhat, C. and Roux, F.-X. (1991).
A method of finite element tearing and interconnecting and its parallel solution algorithm.
*International Journal for Numerical Methods in Engineering*, 32(6) :1205.

Gander, M. J. and Halpern, L. (2012).
Méthodes de décomposition de domaine.
*Encyclopédie électronique pour les ingénieurs*.

Garaud, J.-D. (2008).
*Coupled aero-thermo-mechanical algorithms for non-linear simulations of critically hot aerospace structures*.
Theses, Université Pierre et Marie Curie - Paris VI.

Gosselet, P. and Rey, C. (2006).
Non-overlapping domain decomposition methods in structural mechanics.
*Archives of Computational Methods in Engineering*, 13(4) :515–572.

Gosselet, P., Rixen, D., Roux, F.-X., and Spillane, N. (2015).
Simultaneous FETI and block FETI : Robust domain decomposition with multiple search directions.
*International Journal for Numerical Methods in Engineering*, 104(10) :905–927.
nme.4946.

Greif, C., Rees, T., and Szyld, D. B. (2011).
MPGMRES : a generalized minimum residual method with multiple preconditioners.
Technical Report 11-12-23, Department of Mathematics, Temple University.
Revised September 2012 and January 2014. Also available as Technical Report TR-2011-12, Department of Computer Science, University of British Columbia.

Greif, C., Rees, T., and Szyld, D. B. (2012).
Additive Schwarz with Variable Weights.
Technical Report 12-11-30, Department of Mathematics, Temple University.

Kim, H. H., Chung, E., and Wang, J. (2016).
BDDC and FETI-DP algorithms with adaptive coarse spaces for three-dimensional elliptic problems with oscillatory and high contrast coefficients.
*arXiv preprint arXiv :1606.07560*.

Klawonn, A., Kühn, M., and Rheinbach, O. (2016).
Adaptive coarse spaces for FETI-DP in three dimensions.
*SIAM J. Sci. Comput.*, 38(5) :A2880–A2911.

Klawonn, A., Lanser, M., and Rheinbach, O. (2014).
Nonlinear FETI-DP and BDDC Methods.
*SIAM Journal on Scientific Computing*, 36(2) :A737–A765.

Klawonn, A. and Rheinbach, O. (2010).
Highly scalable parallel domain decomposition methods with an application to biomechanics.
*ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik : Applied Mathematics and Mechanics*, 90(1) :5–32.

Le Tallec, P., De Roeck, Y. H., and Vidrascu, M. (1991).
Domain decomposition methods for large linearly elliptic three-dimensional problems.
*Journal of Computational and Applied Mathematics*, 34(1) :93.

Mandel, J. (1993).
Balancing domain decomposition.
*Communications in Numerical Methods in Engineering*, 9(3) :233.

Mandel, J. and Sousedík, B. (2007).
Adaptive selection of face coarse degrees of freedom in the BDDC and the FETI-DP iterative substructuring methods.
*Comput. Methods Appl. Mech. Engrg.*, 196(8) :1389–1399.

Markopoulos, A., Říha, L., Brzobohatý, T., Meca, O., Kučera, R., and Kozubek, T. (2018).
The htfeti method variant gluing cluster subdomains by kernel matrices representing the rigid body motions.
In Bjørstad, P. E., Brenner, S. C., Halpern, L., Kim, H. H., Kornhuber, R., Rahman, T., and Widlund, O. B., editors, *Domain Decomposition Methods in Science and Engineering XXIV*, Lecture Notes in Computational Science and Engineering, pages 543–551, Cham. Springer International Publishing.

Negrello, C. (2017).
*Highly parallel methods for numerical simulation in nonlinear structural mechanics*.
Theses, Université Paris-Saclay.

Negrello, C., Gosselet, P., Rey, C., and Pebrel, J. (2016).
Substructured formulations of nonlinear structure problems – influence of the interface condition.
*International Journal for Numerical Methods in Engineering*, 107(13) :1083–1105.
nme.5195.

Pebrel, J., Rey, C., and Gosselet, P. (2008).
A Nonlinear Dual-Domain Decomposition Method : Application to Structural Problems with Damage.
*International Journal for Multiscale Computational Engineering*, 6(3) :251.

Pechstein, C. (2012).
*Finite and boundary element tearing and interconnecting solvers for multiscale problems*, volume 90.
Springer Science & Business Media.

Říha, L., Brzobohatý, T., Markopoulos, A., Meca, O., and Kozubek, T. (2016).
Massively parallel hybrid total feti (htfeti) solver.
In *PASC '16*.

Saad, Y. (2003).
*Iterative methods for sparse linear systems*.
Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition.

Spillane, N. (2016).
An Adaptive Multipreconditioned Conjugate Gradient Algorithm.
*SIAM J. Sci. Comput.*, 38(3) :A1896–A1918.

Spillane, N., Dolean, V., Hauret, P., Nataf, F., Pechstein, C., and Scheichl, R. (2014).
Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps.
*Numer. Math.*, 126(4) :741–770.

Spillane, N. and Rixen, D. J. (2013).
Automatic spectral coarse spaces for robust FETI and BDD algorithms.
*International Journal for Numerical Methods in Engineering*, 95(11) :953–990.

Toselli, A. and Widlund, O. (2005).
*Domain decomposition methods—algorithms and theory*, volume 34 of *Springer Series in Computational Mathematics*.
Springer-Verlag, Berlin.