

CentraleSupélec

DE LA RECHERCHE À L'INDUSTRIE



ST7 - HPC
CAMPAGNE
D'EXPLORATION
SISMIQUE

Vendredi 28 mars 2024

Professeurs encadrants

M. COLVEZ, F. Gatti, F. LEHMANN

Alexandre FAURE, Paul HUGUET,
Gabriel SOUZA, Vitor OPSFELDER

INTRODUCTION & ENJEUX

Campagne d'exploration sismique non intrusive pour préserver les ouvrages sensibles.

Objectif 1

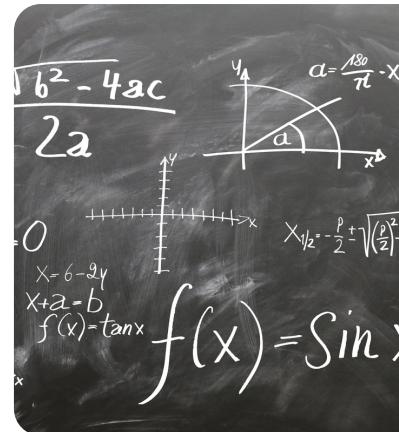
Mettre en pratique



- Mettre en œuvre les concepts d'optimisation et de HPC

Objectif 2

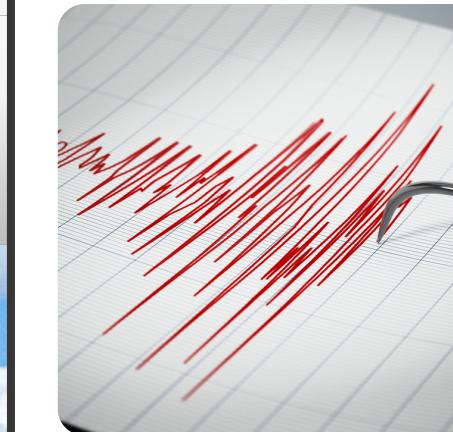
Comprendre et adopter les méthodes



- Comprendre la modélisation physique
- Maîtriser la résolution d'un problème adjoint

Objectif 3

Mener une campagne d'exploration



- Retrouver les paramètres de Lamé sur un site donné

SOMMAIRE

- 01** Formalisation du problème
- 02** Mise en œuvre et réalisations
- 03** Performances et parallélisation

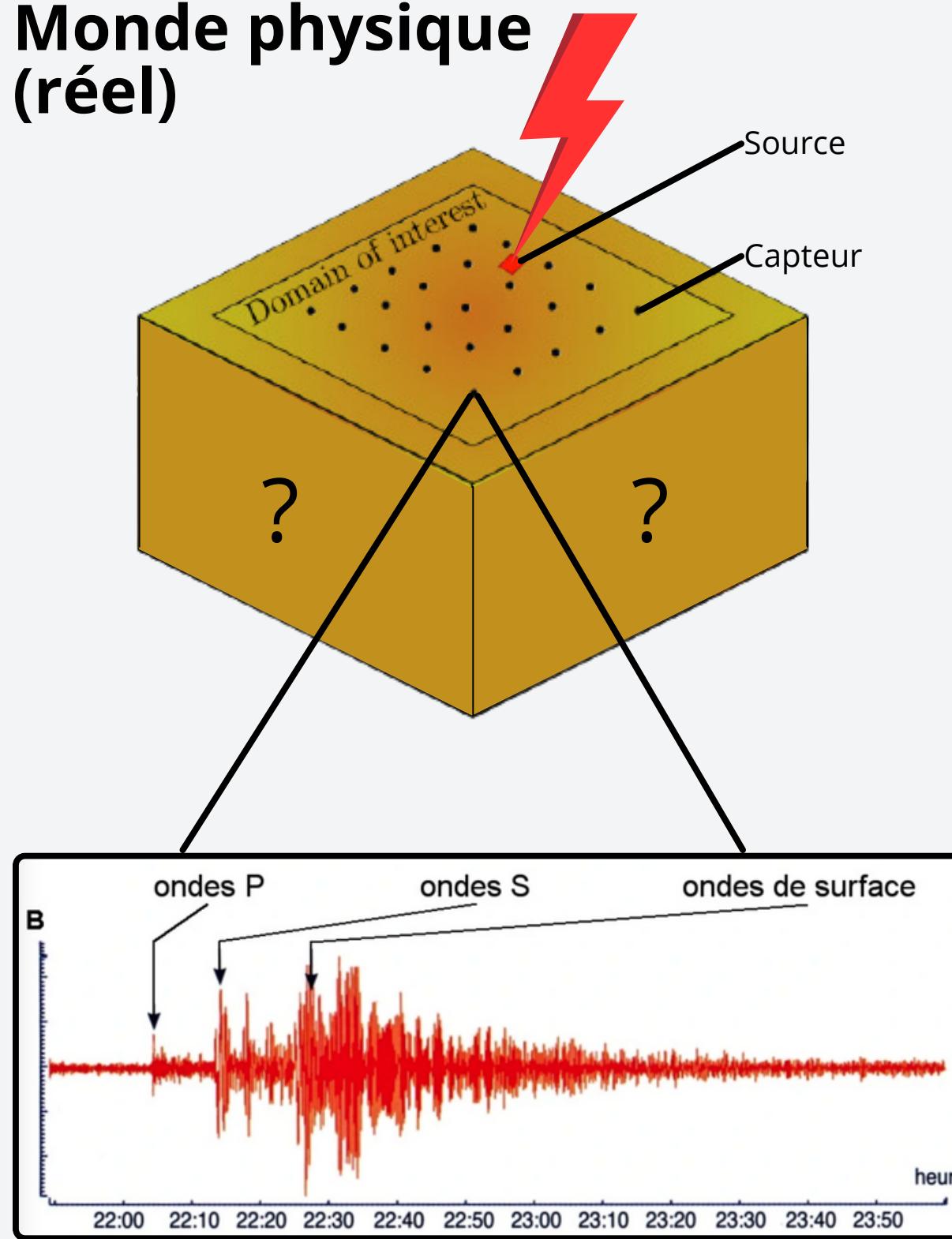


FORMALISATION DU PROBLÈME

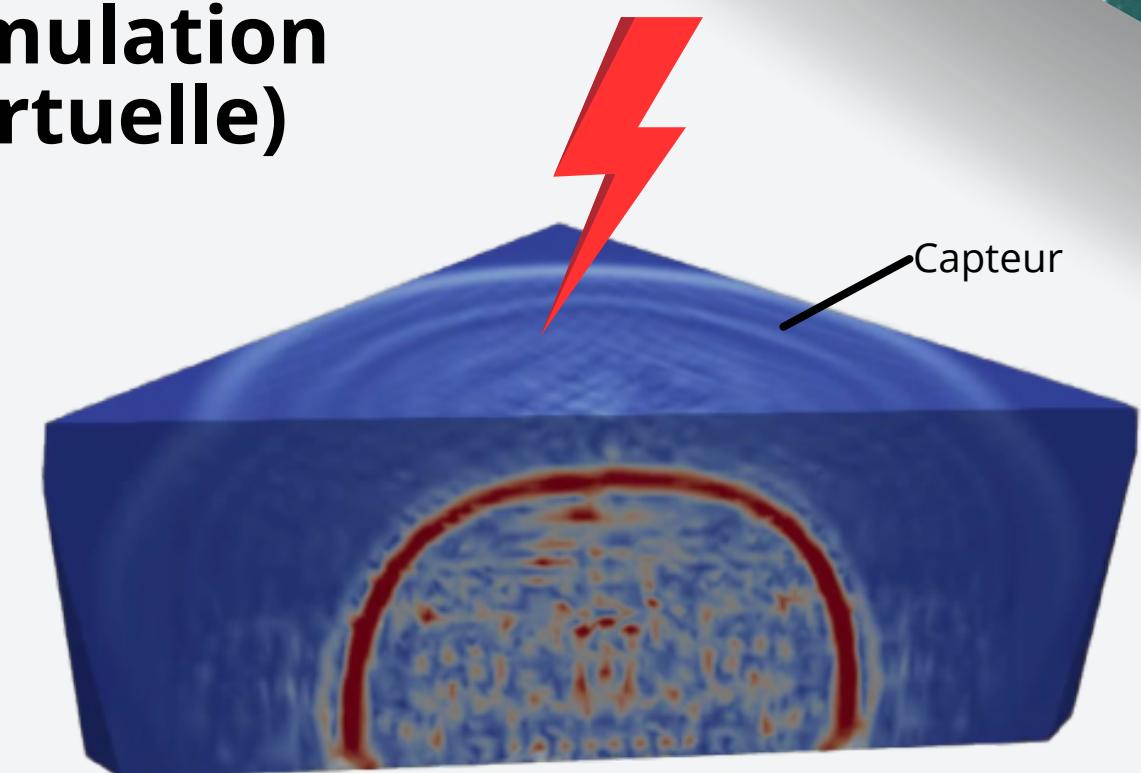
1.

ÉCOLE POLYTECHNIQUE

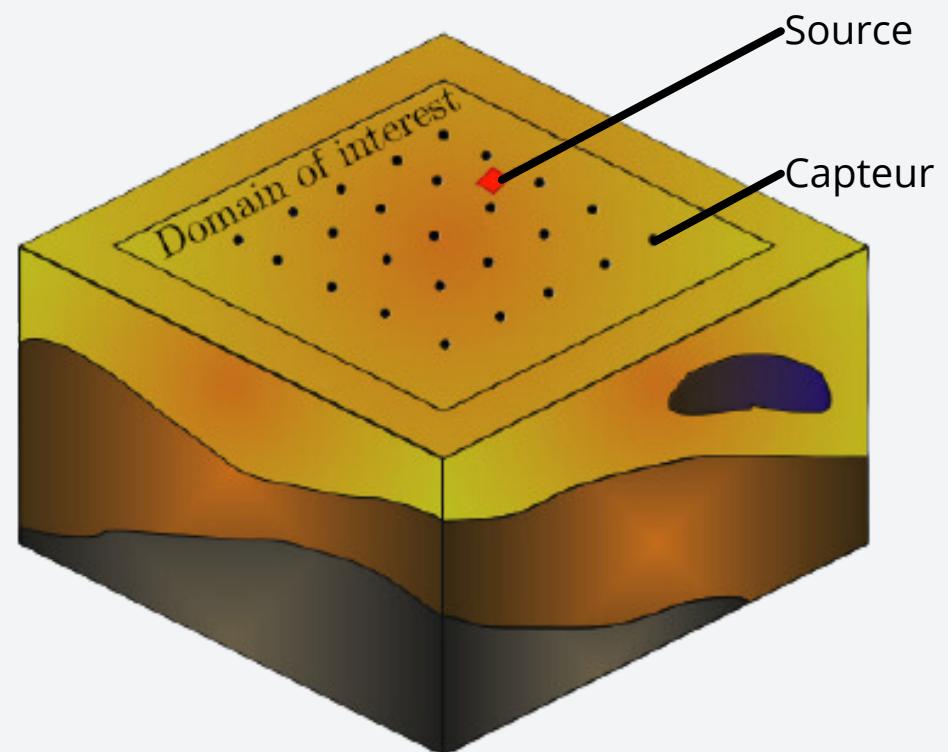
MONDE PHYSIQUE (réel)



Simulation (virtuelle)



itérations successives



FORMALISATION DU PROBLÈME

$\tilde{m} \in \mathbb{M}$ l'espace des paramètres de Lamé sur le maillage

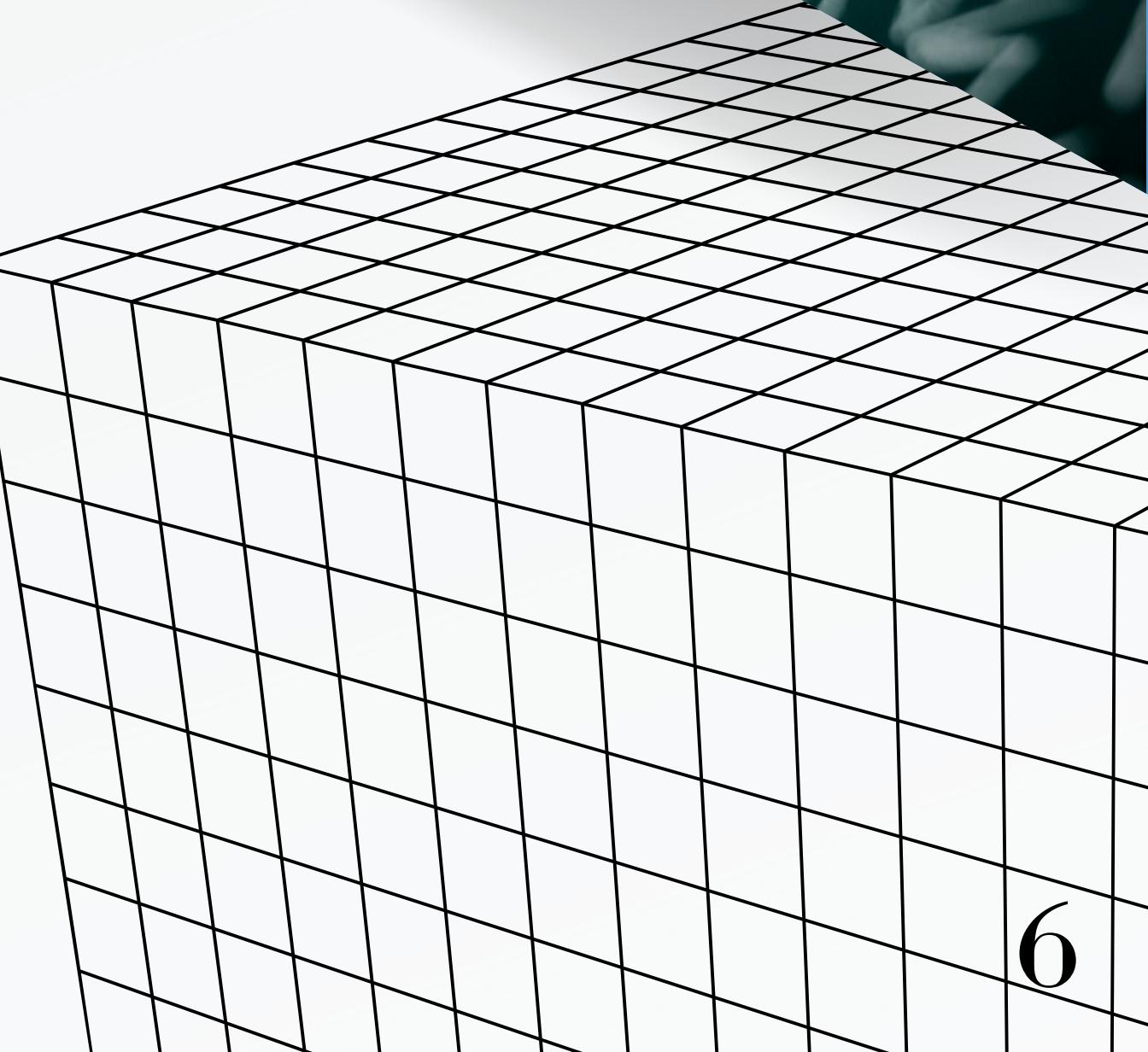
$\tilde{u} \in \mathbb{U}$ l'espace des réalisations possibles de la propagation de l'onde

Problème direct

On cherche $u(\tilde{m}) \in \mathbb{U}$ t.q. $\mathcal{F}(u(\tilde{m}), \tilde{m}) = 0$

Problème inverse

On cherche $m \in \mathbb{M}$ t.q. $\mathcal{F}(\tilde{u}, m) = 0$



FONCTION DE COÛT

Pour obtenir les paramètres de Lamé, on minimise une **fonction de coût** :

$$\tilde{m} = \arg_{m \in \mathbb{M}} \mathcal{J}(m) = \arg_{m \in \mathbb{M}} \mathcal{H}(u(m), m)$$

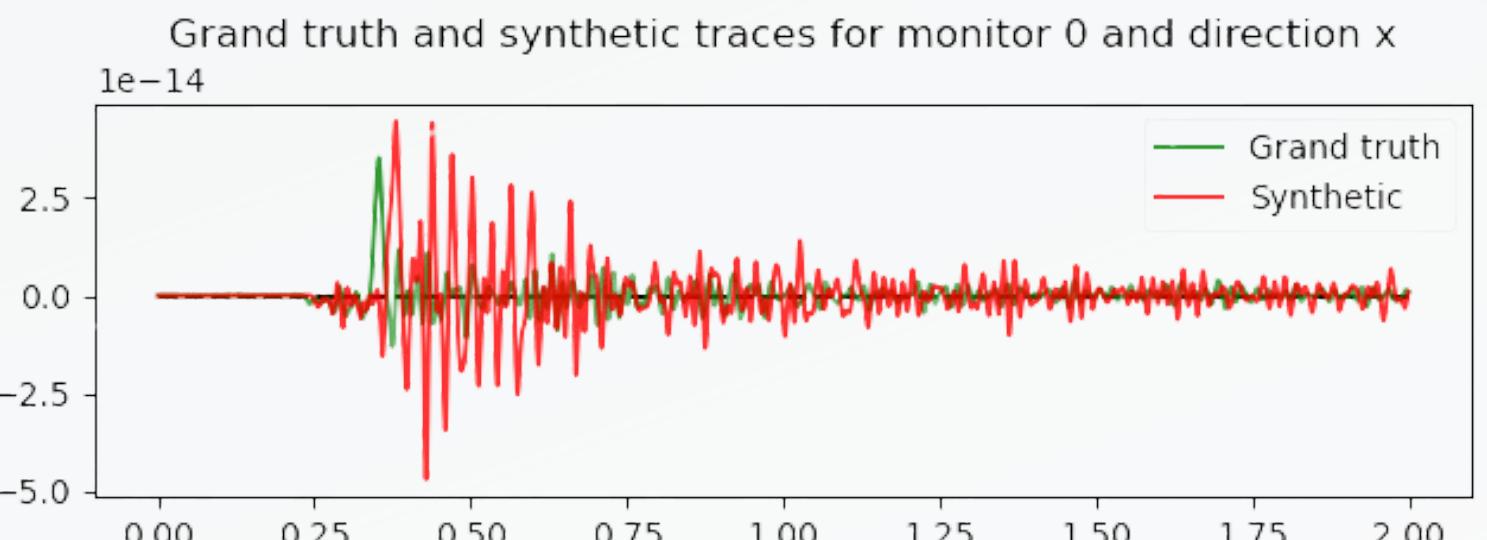
Voici son expression :

$$\mathcal{J}(m) = \frac{1}{2} \sum_{i=1}^3 \int_0^{T=2s} \|\tilde{u}(x_i^r, t) - u_i^{obs}(t)\|^2 dt + \mathcal{R}(m)$$

Et l'expression du terme de **régularisation** :

$$\mathcal{R}(m) = \frac{\mathcal{R}_\lambda}{2} \int_\Omega \langle \nabla_x \lambda, \nabla_x \lambda \rangle dv + \frac{\mathcal{R}_\mu}{2} \int_\Omega \langle \nabla_x \mu, \nabla_x \mu \rangle dv$$

pour pénaliser les matériaux inhomogènes.



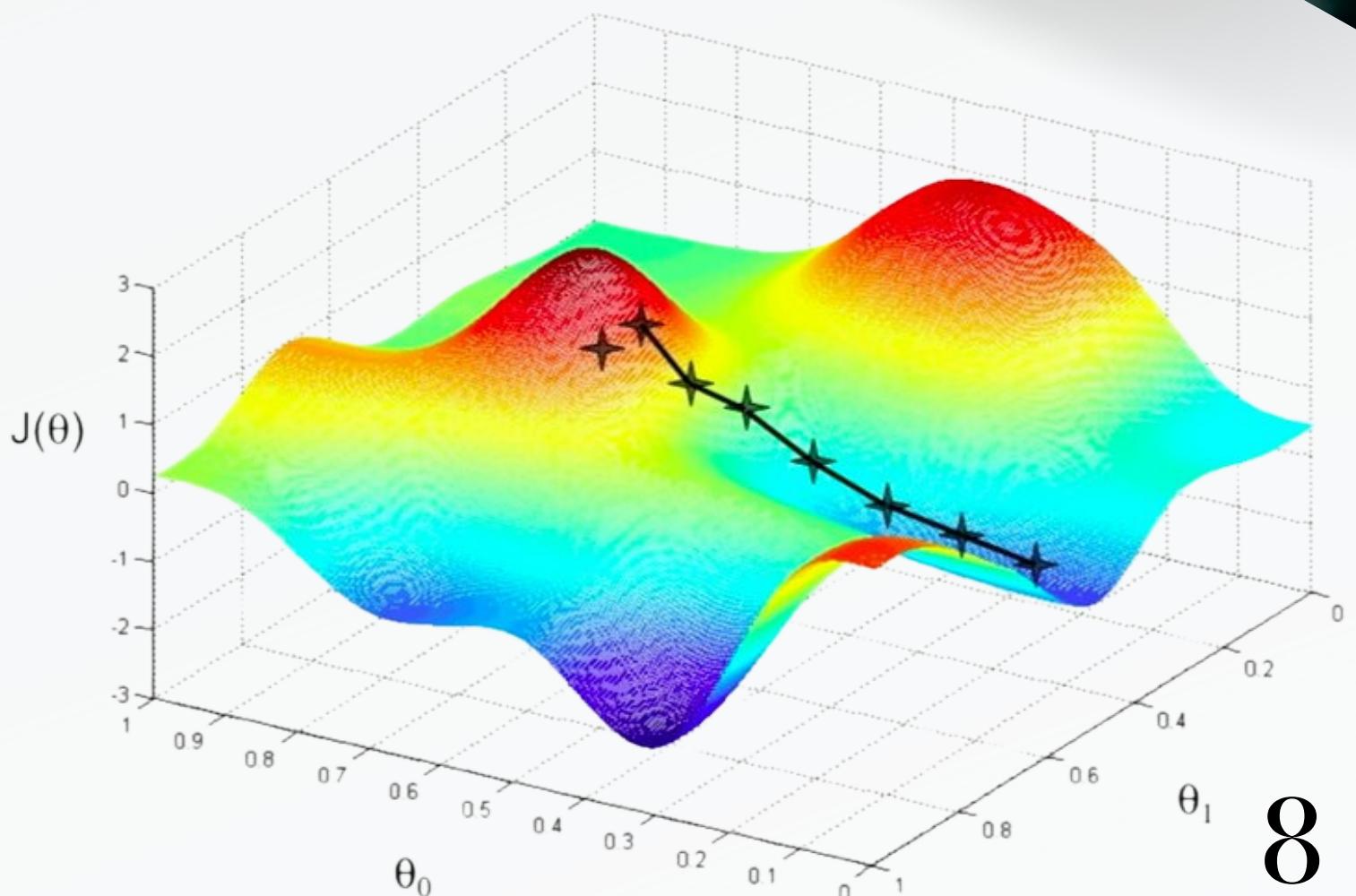
DESCENTE DE GRADIENT

Le matériau optimal est déterminé par itération successive, par **descente de gradient** de la fonction coût.

On résout un problème à $2.N \approx 7\,000\,000$ paramètres.

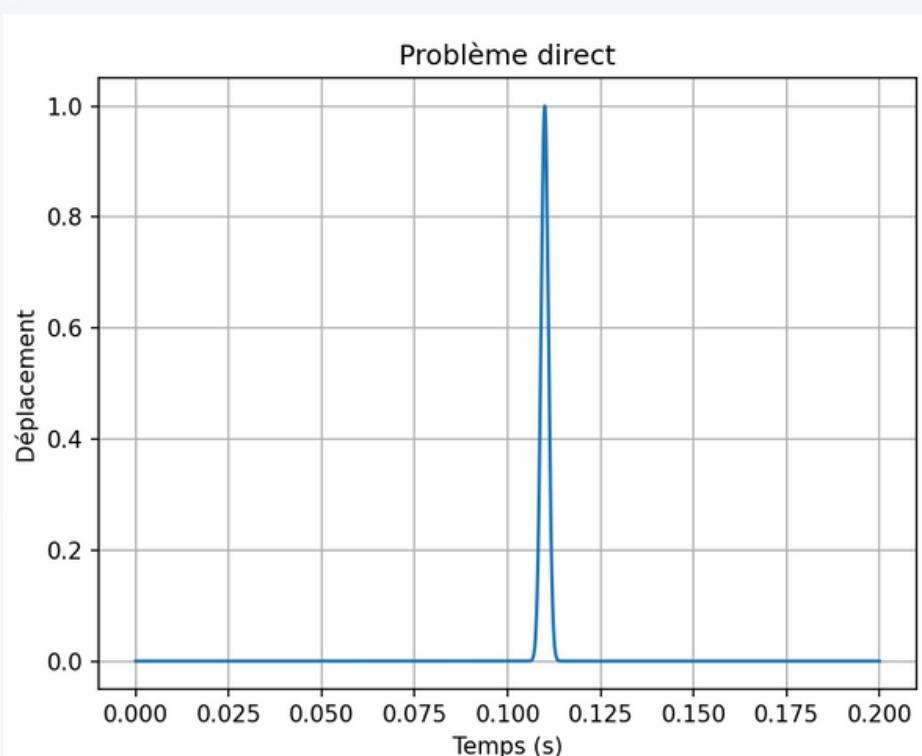
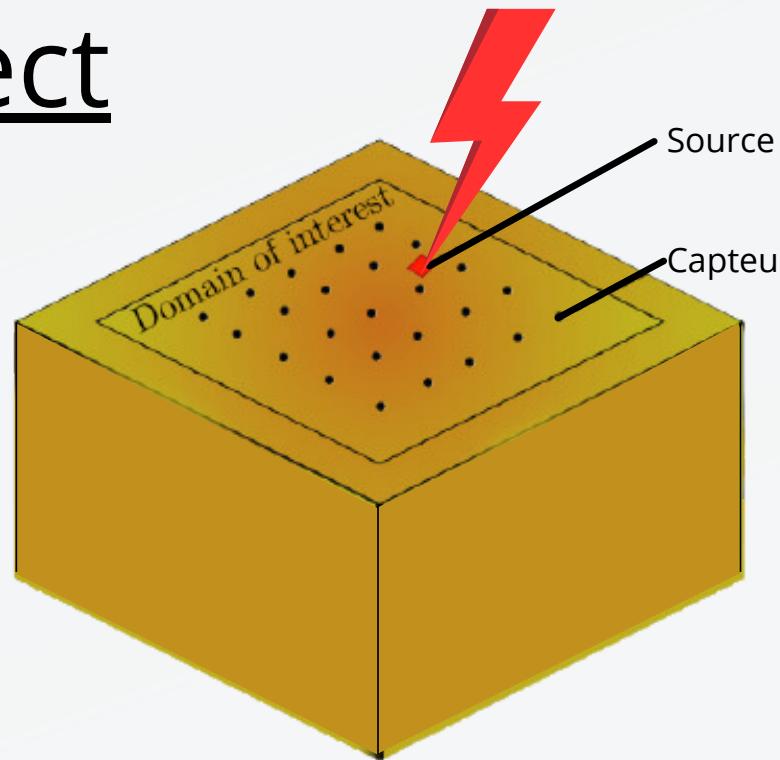
- trop de directions à calculer pour le gradient

On passe par le problème adjoint.



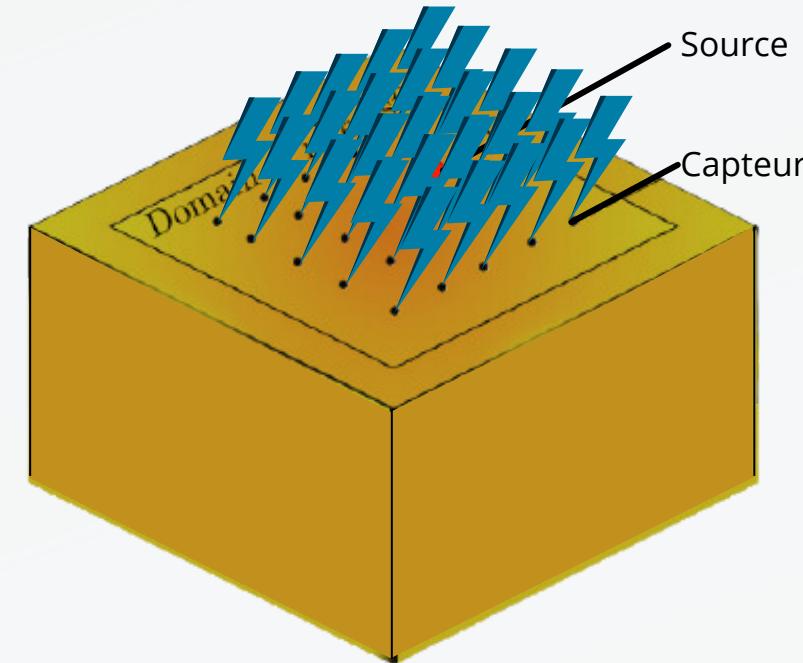
PASSAGE PAR LE PROBLÈME ADJOINT

Direct



Input

i Adjoint



Les **récepteurs** (capteurs) deviennent les **sources**.

Le signal émis est le misfit entre le signal réel (grand truth) et le signal reçu dans la simulation).

GRADIENT DE LA FONCTION COÛT

Par discréttisation du Lagrangien, on calcule le gradient de la fonction de coût.

$$\frac{\partial \mathcal{L}}{\partial u} = 0$$

Le **gradient** est calculé grâce à la résolution du problème direct et du problème adjoint :

$$\begin{cases} \tilde{g}_\lambda = \tilde{M}^{-1} [\mathcal{R}_\lambda \tilde{g}_\lambda^{reg} + \tilde{g}_{mis}^\lambda] \\ \tilde{g}_\mu = \tilde{M}^{-1} [\mathcal{R}_\mu \tilde{g}_\mu^{reg} + \tilde{g}_{mis}^\mu] \end{cases}$$

Une correction est ensuite appliquée.

L-BFGS

- Des étapes plus importantes lorsque la courbure est forte.
- Approximer la matrice inverse du Hessien.
- Ajuster la taille du pas pour converger plus rapidement.
- En ne stockant qu'un historique limité des itérations passées, le L-BFGS préserve les ressources mémoire.

Two Loop Recursion

```
q ←  $\nabla f_k$ ;
for  $i = k - 1, k - 2, \dots, k - m$ 
     $\alpha_i \leftarrow \rho_i s_i^T q$ ;
     $q \leftarrow q - \alpha_i y_i$ ;
end (for)
 $r \leftarrow H_k^0 q$ ;
for  $i = k - m, k - m + 1, \dots, k - 1$ 
     $\beta \leftarrow \rho_i y_i^T r$ ;
     $r \leftarrow r + s_i (\alpha_i - \beta)$ 
end (for)
stop with result  $H_k \nabla f_k = r$ .
```

BACKTRACKING LINE SEARCH

- Pour garantir une diminution suffisante de la fonction objective à chaque itération de l'inversion, nous employons une recherche linéaire de type backtracking d'Armijo.

```
while  $\mathcal{J}\left(\lambda^{(k)} + \alpha_\lambda \mathbf{s}_\lambda^{(k)}, \mu^{(k)} + \alpha_\mu \mathbf{s}_\mu^{(k)}\right) \geq \mathcal{J}\left(\lambda^{(k)}, \mu^{(k)}\right) + c_1 \left(\alpha_\lambda \mathbf{s}_\lambda^{(k)} \cdot \tilde{\mathbf{g}}_\lambda^{(k)} + \alpha_\mu \mathbf{s}_\mu^{(k)} \cdot \tilde{\mathbf{g}}_\mu^{(k)}\right)$  do  
    |  $\alpha_\lambda = \xi \alpha_\lambda$   
    |  $\alpha_\mu = \xi \alpha_\mu$   
end
```

$$\mu_{k+1} = \mu_k + \alpha_{\mu_k} H_k \nabla_\mu J_k$$

$$\lambda_{k+1} = \lambda_k + \alpha_{\lambda_k} H_k \nabla_\lambda J_k$$

2.

MISE EN OEUVRE & RÉALISATIONS

```
tab.color = colors
def on_key_press(self, symbol, modifiers):
    if self.context_index == -1:
        if symbol == key.UP and not self.active_index ==
            self.menu_labels[self.active_index].color =
                self.mags_dt = self.get_act_color_mag()
        if symbol == key.DOWN and not self.active_index ==
            self.menu_labels[self.active_index].color =
                self.active_index += 1
                self.mags_dt = self.get_act_color_mag()
    if symbol == key.ENTER:
        if self.active_index == 3:
            app.exit()
    if self.active_index == 0:
        index = self.active_index
    if symbol == key.ESCAPE:
        if self.active_index == -1:
            self.active_index = 0
            self.mags_dt = self.get_act_color_mag()
            self.menu_labels[0].color = colors
```

ÉTAPES DE L'ALGORITHME

$k \leftarrow 1$

Résolution du problème direct

tant que $k \leq N_{iter}$ **et** $\mathcal{J}(\lambda^{(k)}, \mu^{(k)}) > tol$

Calcul du misfit aux récepteurs \mathcal{F}

Résolution du problème adjoint \mathcal{A}

Calcul de la fonction de coût $\mathcal{J}(\lambda^{(k)}, \mu^{(k)})$

Calcul des gradients selon les paramètres $\tilde{g}_\lambda^{(k)}$ et $\tilde{g}_\mu^{(k)}$

Recherche des directions de descente $s_\lambda^{(k)}$ et $s_\mu^{(k)}$

Calcul du pas de descente $\alpha_\lambda^{(k)}$ et $\alpha_\mu^{(k)}$ (*inclus la mise à jour des paramètres du matériau et la résolution de problèmes directs*)

$k \leftarrow k + 1$

fin du tant que

```
def on_key_press(self, symbol, modifiers):
    """Delegate key press"""
    if self.context_index == -1:
        if symbol == key.UP and not self.active_index == 0:
            self.menu_labels[self.active_index].color = [255, 255, 255, 255]
            self.active_index -= 1
            self.mags_dt = self.get_act_color_mag()
        elif symbol == key.DOWN and not self.active_index == 3:
            self.menu_labels[self.active_index].color = [255, 255, 255, 255]
            self.active_index += 1
            self.get_act_color_mag()
    else:
        self.mags_dt = self.get_act_color_mag()
```

ORGANISATION DU TRAVAIL

26 fév. - Début du projet

Familiarisation avec les concepts théoriques

13 mar. - Lancement du projet en groupe

Travail en groupe pour

- approfondir les aspects théoriques pour l'étude
- se familiariser avec SEM3D et les outils
- constituer les premières briques élémentaires du code

19 mar. - Mise en cohérence de l'algorithme final

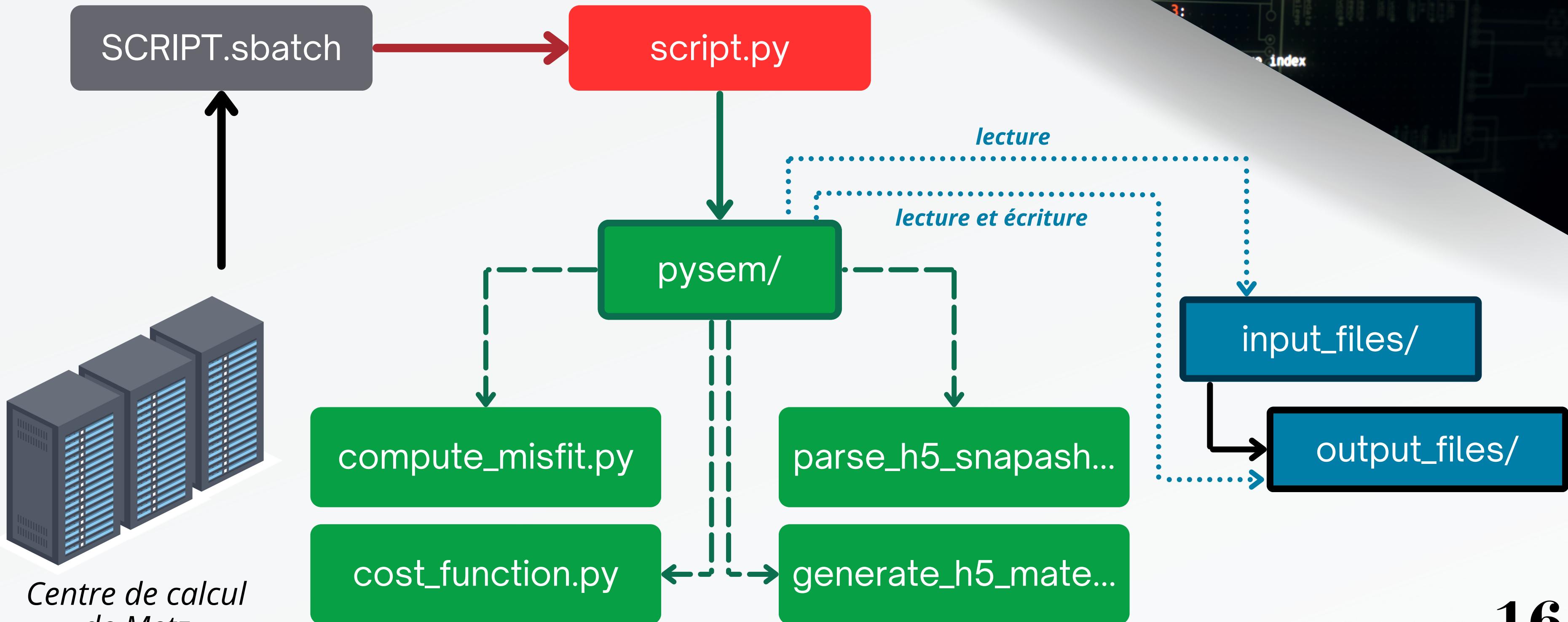
Finalisation des briques élémentaires

- Vitor : fonction de coût et régularisation
- Gabriel : direction de descente
- Alexandre : script global, mise à jour du matériau, parallélisation partielle
- Paul : Backtracking line search

Préparation des livrables finaux

- Paul & Alexandre : diaporama et tests de performances
- Tous: rapport

STRUCTURE FINALE DES CODES



DIFFICULTÉS RENCONTRÉES

- Maillage SEM3D vs matériau h5
 - gestion des points d'intégrations, cellules, éléments...
- Mise en place de l'algorithme L-BFGS
 - modification de la routine de calcul
 - manque de temps pour le rendre fonctionnel
- Prise en main des scripts initiaux
 - parse_h5_snapshots.py
 - parse_h5_trace.py

```
def on_key_press(self, symbol, modifiers):
    """Delegate key press"""
    if self.context_index == -1:
        if symbol == key.UP and not self.active_index == 0:
            self.menu_labels[self.active_index].color = [255, 255, 255, 255]
            self.active_index -= 1
            self.mags_dt = self.get_act_color_mag()
        elif symbol == key.DOWN and not self.active_index == 3:
            self.menu_labels[self.active_index].color = [255, 255, 255, 255]
            self.active_index += 1
            self.get_act_color_mag()
    else:
        self.context_index = -1
```

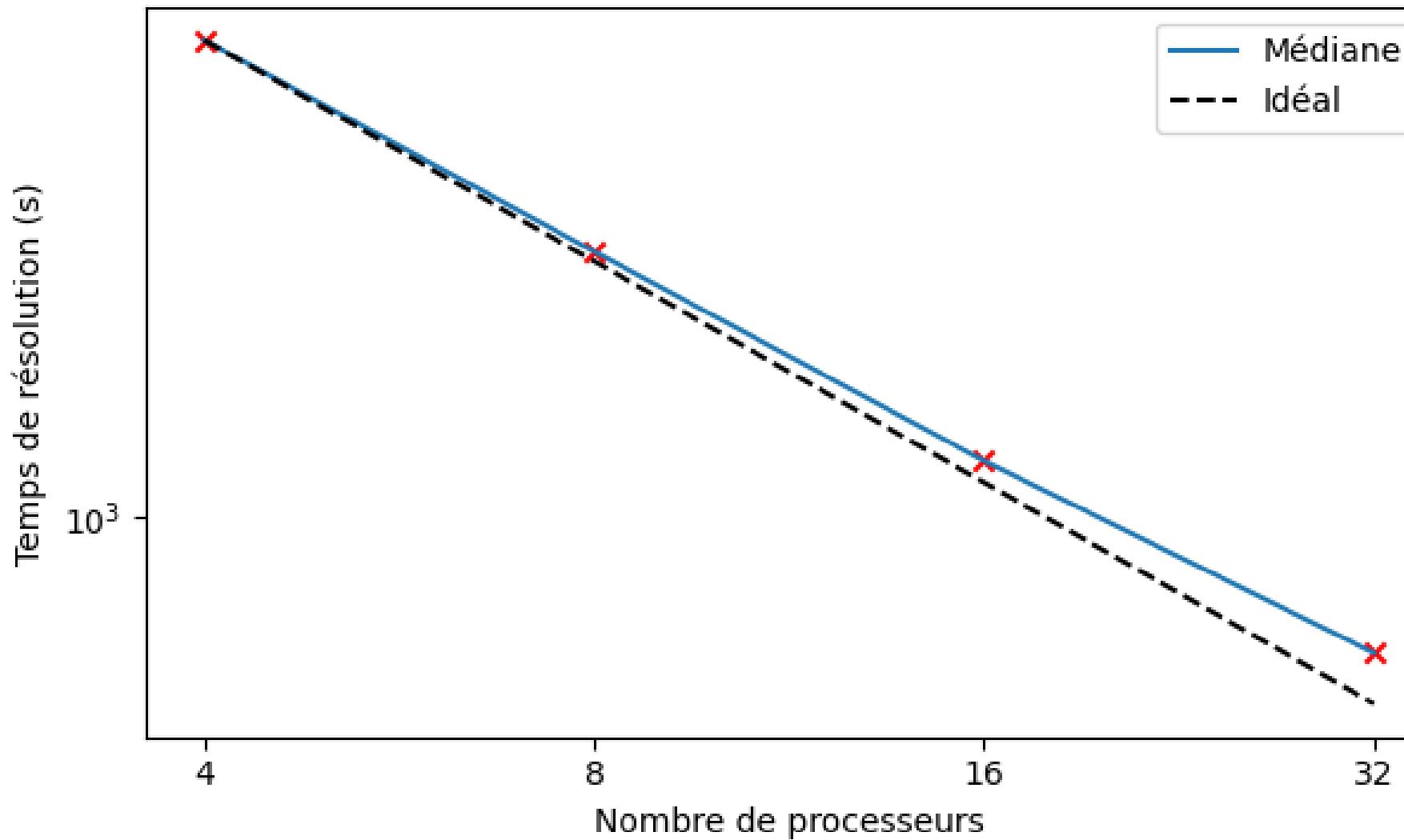
3. PERFORMANCES & PARALLELISATION



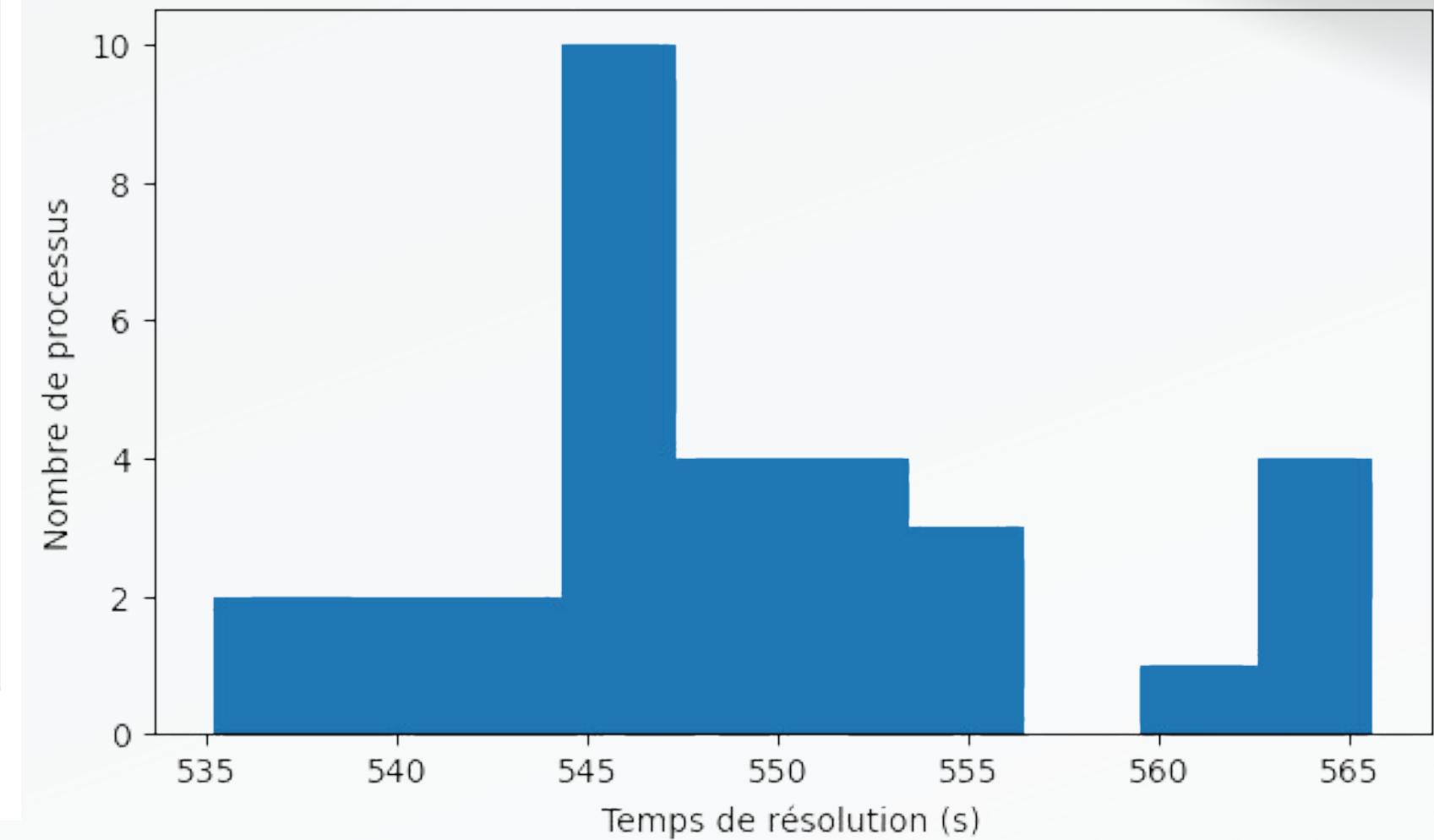
PERFORMANCES SOLVER

- Bon speed-up
- Bon load-balancing (3% d'écart vs médiane)

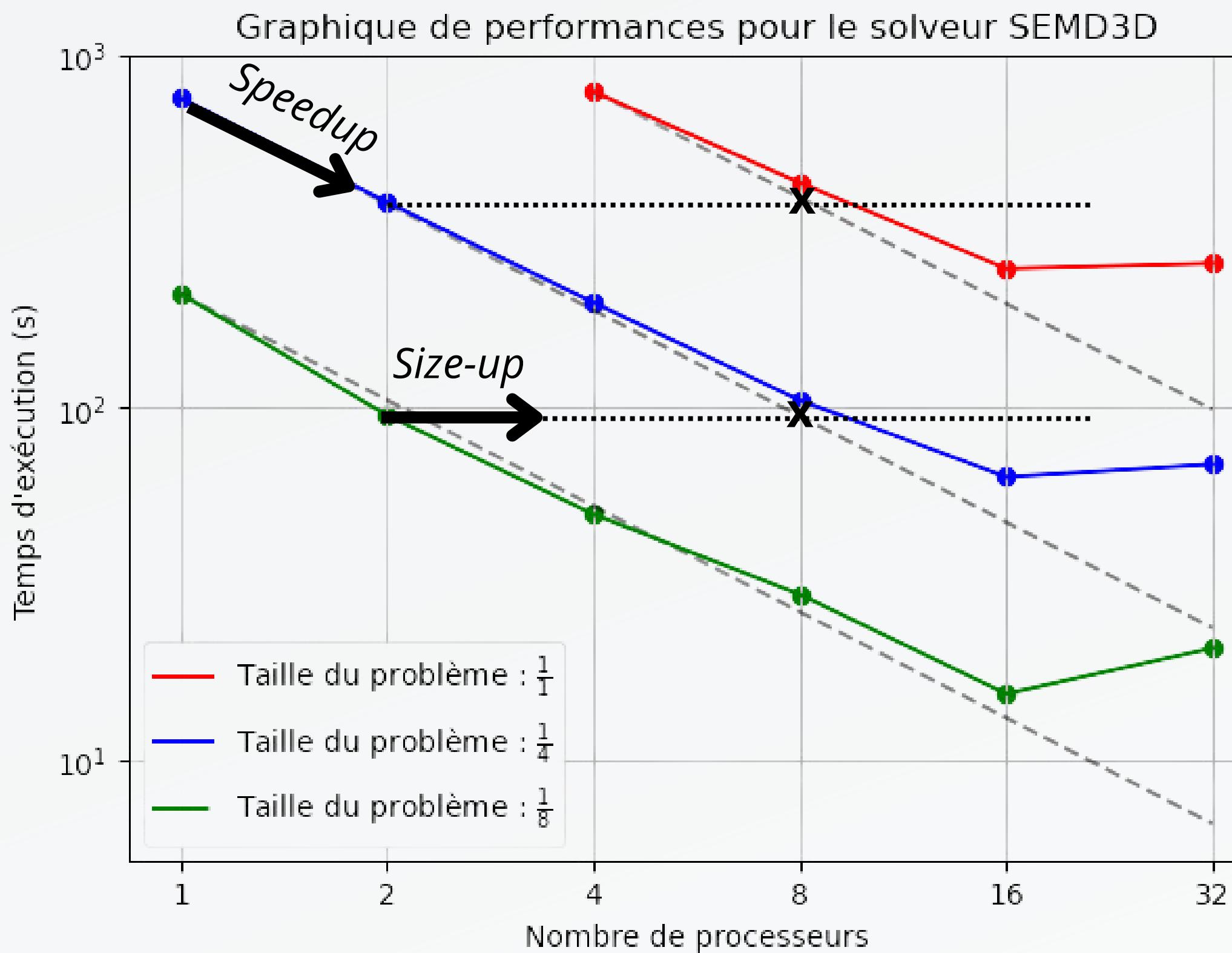
Temps de résolution en fonction du nombre de processeurs pour le solveur de SEM3D



Histogramme des temps de résolution pour 32 processeurs pour le solveur SEM3D



PERFORMANCES SOLVER



Pour un matériau plus homogène

- Bon speedup
 - jusqu'à 16 proc
- Bon size-up
 - jusqu'à 16 proc

CONCLUSION

- Bonne appropriation : méthode des éléments finis, problème adjoint, algorithme L-BFGS
- Implémentation de l'algorithme quasi complète (intégration et débogage non terminé)
- Parallélisation d'une partie du code réalisée
- Premiers résultats de performances





CentraleSupélec



**MERCI DE VOTRE
ATTENTION**

