Accélération de la méthode BEM pour l'équation de Helmholtz 2D : algorithmes de factorisation de rang faible

TP3 : à rendre le 03/11/2024 (environ 2 pages sans les pseudo-codes)

Rappel de la Philosophie des TPs : Le but de la série de TPs est de mettre en oeuvre un solveur BEM rapide pour l'équation de Hemholtz 2D. Nous choisissons le cas de la diffraction d'une onde incidente plane, i.e. $u^{inc} = e^{-i\mathbf{k}.\mathbf{x}}$, par un disque de rayon a centré en $\mathbf{0}$ et de frontière Γ. $\mathbf{k}$ est le vecteur qui permet de déterminer l'angle d'incidence de l'onde. Le nombre d'onde du problème est donné par $k = |\mathbf{k}|$. Le domaine extérieur est noté Ω^+. Nous allons considérer le cas d'une condition à la frontière de type Dirichlet. Ce cas test a l'avantage de présenter une solution analytique simple. Le champ diffracté est donné par :

$$u^+(r,\theta) = -\sum_{n\in\mathbb{Z}} (-i)^n \frac{J_n(ka)}{H_n^{(1)}(ka)} H_n^{(1)}(kr) e^{in\theta}, \quad r \geq a. \tag{1}$$

Les J_n sont les fonctions de Bessel de première espèce et les $H_n^{(1)}$ les fonctions de Hankel du premier type.

Grandes étapes de la BEM :

1. Reformulation de l'EDP sous forme intégrale ;
2. Résolution d'une équation intégrale pour obtenir les traces des champs sur la frontière ;
3. Application de la représentation intégrale pour obtenir le champ dans tout le domaine (non-borné pour les problèmes extérieurs).

Lors du TP1, nous avons vu que le champ diffracté dans le cas de conditions à la frontière de type Dirichlet est donné par la représentation intégrale

$$u^+(\mathbf{x}) = \int_\Gamma G(\mathbf{x},\mathbf{y}) p(\mathbf{y}) d\Gamma(\mathbf{y}) \tag{2}$$

où $p = -\partial_{\mathbf{n}} u^+ - \partial_{\mathbf{n}} u^{inc}$ avec $\mathbf{n}$ la normale extérieure au disque.
Dans le TP2, p a été déterminé numériquement en résolvant l'équation intégrale

$$\begin{array}{l} \text{Trouver } p \in H^{-1/2}(\Gamma) \text{ tel que} \\ \displaystyle\int_\Gamma G(\mathbf{x},\mathbf{y}) p(\mathbf{y}) d\Gamma(\mathbf{y}) = -u^{\text{inc}} \end{array} \tag{3}$$

TP3 : Algorithmes de factorisation de rang faibl

Dans ce TP, nous allons tester les algorithmes de factorisation par des matrices de rang faible qui sont utilisés pour accélérer la résolution numérique de l'équation intégrale pour les formulations directes.
Les objectifs qui devront être bien expliqués dans votre compte-rendu sont les suivants :

- Comprendre le concept de rang numérique et d'approximation de rang faible.
- Comparer différentes méthodes de compression de matrices.
- Évaluer leurs performances en précision et en coût de calcul.

Principales étapes du travail (à détailler dans votre compte-rendu)

1. Générer des matrices de rang faible (en entrée vous donnerez la taille de la matrice et le rang).
2. Utiliser la méthode SVD et tracer les valeurs singulières de vos matrices de rang faible. Est-ce que le comportement observé est celui prévu ? Vous pouvez observer l'erreur obtenue avec la SVD tronquée en fonction du rang numérique choisi. Quels sont les avantages et les incovénients de cette approche ?
3. Coder la méthode ACA avec pivotage complet (Algo 1). Est-ce que le comportement observé est celui prévu ? Ici aussi vous pouvez étudier l'influence du rang sur la précision. Quels sont les avantages et les incovénients de cette approche ?

Algorithm 1 Pseudo-code : ACA avec pivotage complet

Initialization : $\mathbb{R}_0 := \mathbb{A}$, $k = 0$
repeat
  $k := k + 1$
  Find the pivot $(i^*, j^*) := \text{argmax}_{i,j} |\mathbb{R}_{k-1}(i, j)|$
  $\delta_k := \mathbb{R}_{k-1}(i^*, j^*)$
  $\mathbf{u}_k := \mathbb{R}_{k-1}(:, j^*)$
  $\mathbf{v}_k := \mathbb{R}_{k-1}(i^*, :)/\delta_k$
  $\mathbb{R}_k = \mathbb{R}_{k-1} - \mathbf{u}_k \mathbf{v}_k$
until $||\mathbb{R}_k||_F \leq \varepsilon ||\mathbb{A}||_F$

4. L'étape suivante est de coder la méthode ACA avec pivotage partiel (Algo 2). Nous allons décomposer cette étape en quatre étapes pour bien comprendre l'importance de chaque changement.
   (a) Améliorer l'ACA avec pivotage complet en ne cherchant plus le pivot dans toute la matrice mais juste sur une ligne ou colonne.
   (b) Ne pas assembler $\mathbb{R}$ mais mettre à jour juste les parties utiles de la matrice en se rappelant que $\mathbb{R}_k(i, j) = \mathbb{A}(i, j) - \sum_{\ell=1}^{k} \mathbf{u}_\ell(i)\mathbf{v}_\ell(j)$.
   (c) Ajouter le test sur les lignes ou colonnes déjà utilisées (cf Algo 2).
   (d) Remplacer le critère de convergence par le critère de non stagnation pour ne pas avoir de partie de complexité quadratique dans votre code, i.e., le critère est

$||\mathbf{u}_k||_2||\mathbf{v}_k||_2 \leq \varepsilon||\mathbb{A}_k||_F$. Il faudra aussi utiliser le calcul astucieux de la norme de Frobenius :

$$||\mathbf{A}_k||_F^2 = ||\mathbf{A}_{k-1}||_F^2 + 2\sum_{\ell=1}^{k-1} \mathbf{u}_k^T \mathbf{u}_\ell \mathbf{v}_\ell^T \mathbf{v}_k + ||\mathbf{u}_k||_2^2||\mathbf{v}_k||_2^2$$

Algorithm 2 Pseudo-code : ACA avec pivotage partiel mais avec calcul explicite du résidu

Initialization : $\mathbb{R}_0 := \mathcal{A}$, $\mathcal{P}_r = \emptyset$; $\mathcal{P}_c = \emptyset$, $k = 1$, $i^* = 1$
repeat
  $j^* := \text{argmax}_j|\mathbb{R}_{k-1}(i^*, j)|$
  $\delta_k := \mathbb{R}_{k-1}(i^*, j^*)$
  if $\delta_k = 0$ **then**
    if $|\mathcal{P}_r| = N - 1$ **then**
      STOP
    end if
  else
    $\mathbf{u}_k := \mathbb{R}_{k-1}(:, j^*)$
    $\mathbf{v}_k := \mathbb{R}_{k-1}(i^*, :)/\delta_k$
    $\mathbb{R}_k = \mathbb{R}_{k-1} - \mathbf{u}_k\mathbf{v}_k$
    $k := k + 1$
  end if
  $\mathcal{P}_r = \mathcal{P}_r \cup \{i^*\}, \quad \mathcal{P}_c = \mathcal{P}_c \cup \{j^*\}$
  $i^* := \text{argmax}_{i \notin P_r}|u_k(i)|$
until $||\mathbb{R}_k||_F \leq \varepsilon||\mathbb{A}||_F$

Bien détailler les pseudo-codes des différentes étapes et expliquer l'utilité de chaque étape. Est-ce que le comportement observé final est celui prévu ? Quels sont les avantages et les incovénients de cette approche ?

5. Maintenant que vous avez trois approches : comparer les temps de calcul, les complexités et les rangs obtenus (pour des matrices dont vous connaissez le rang exact ou le rang numérique). Expliquez bien la différence entre rang exact et rang numérique.
6. Tester ces trois approches pour la matrice BEM. Ici aussi il y a trois étapes :
   (a) La matrice est assemblée totalement avant de rentrer dans cette routine.
   (b) Vous appelez au vol juste la routine qui calcule la fonction de Green.
   (c) Vous calculez juste au vol une ligne ou une colonne de la matrice BEM - c'est un peu plus difficile à faire. Il faut réfléchir sur le papier à quoi correspond une ligne ou une colonne de la matrice.

   Qu'observez-vous ? Est-ce qu'on peut utiliser cette approche pour accélerer les BEMs ?

BON COURAGE !