AMS-304 Résolution des problèmes de diffraction par équations intégrales (2017-2018)

Approximations par des matrices de rang faible

Le but de ce TP est de mettre en évidence la structure particulière de la matrice de la BEM et la possibilité de l'approcher par une matrice de rang faible. Pour simplifier le problème, on ne considère pas la matrice BEM complète mais la matrice $\mathbb{G}$ correspondant à la discrétisation du noyau de Green. En effet, la possibilité de faire une approximation de rang faible provient essentiellement de la structure de $\mathbb{G}$.
Pour simplifier encore, on considère un domaine 1D. Soit X un nuage de points défini comme la discrétisation uniforme de l'intervalle $[-1, 1]$. La fonction de Green G de l'équation de Helmholtz pour l'espace infini 3D est donnée par

$$G(\mathbf{x}-\mathbf{y}) = \frac{\exp(ik|\mathbf{x}-\mathbf{y}|)}{|\mathbf{x}-\mathbf{y}|}$$

où k est le nombre d'onde. Pour éviter le traitement des singularités, on étudie plutôt la fonction G_α définie par

$$G_\alpha(\mathbf{x}-\mathbf{y}) = \frac{\exp(ik|\mathbf{x}-\mathbf{y}|)}{|\mathbf{x}-\mathbf{y}|+\alpha}.$$

Le but est de calculer de manière rapide, quelque soit le vecteur $\boldsymbol{\rho}$, le vecteur $\mathbb{V}$ défini par

$$\mathbb{V}_i = \sum_j G_\alpha(\mathbf{x}_i - \mathbf{x}_j)\boldsymbol{\rho}_j, \quad \forall \mathbf{x_i} \in X.$$

Matriciellement, on cherche à effectuer le produit matrice-vecteur $\mathbb{V} = \mathbb{G}_\alpha\boldsymbol{\rho}$.

Question 1 (Génération du nuage de points).
Créer une fonction pour générer un nuage de points X, i.e. la discrétisation de l'intervalle $[-1, 1]$ avec $N+1$ points. On choisit le paramètre α de l'ordre d'un pas d'espace (pour simuler que $\mathbf{x}_i \neq \mathbf{x}_j$), i.e. $\alpha = 2/N$.

Afin d'utiliser les capacités de Matlab, il faut chercher à éviter de faire des boucles. Pour créer l'ensemble des couples de points $(\mathbf{x}, \mathbf{y})$, on utilise la fonction Meshgrid :

```
[X,Y]=meshgrid(x,x)
```

Décomposition en valeurs singulières tronquée

La SVD de la matrice $\mathbb{M}$ de taille $n \times n$ est donnée par :

$$\mathbb{M} = \mathbb{U}\mathbf{\Sigma}\mathbb{V}^t.$$

$\mathbb{M}_r$ est la SVD de $\mathbb{M}$ tronquée aux r plus grandes valeurs singulières

$$\mathbb{M}_r = \sum_{i=1}^{r} \mathbb{U}_i \mathbf{\Sigma}_{ii} \mathbb{V}_i^*.$$

Le résultat principal du cours sur la SVD est donné par

$$||\mathbb{M} - \mathbb{M}_k||_F^2 = \sum_{i=k+1}^{n} \sigma_i^2 \text{ et } ||\mathbb{M} - \mathbb{M}_k||_2 = \sigma_{k+1}. \quad (1)$$

Question 2 (Etude des valeurs singulières de la matrice complète).

(a) Assembler la matrice $\mathbb{G}_\alpha$ correspondant à l'évaluation de la fonction G_α pour tous les couples de points du nuage X pour $N = 1000$ et $k = 0,\ 0.1,\ 100$.

(b) Calculer la Décomposition en Valeurs Singulières de $\mathbb{G}_\alpha$ et représenter sur une figure la décroissance des valeurs singulières (utiliser la fonction Matlab loglog). Quelle est l'influence de k ?

(c) Vérifier numériquement le résultat (1) en faisant varier le nombre de valeurs singulières conservées.

(d) Vérifier numériquement la complexité théorique de SVD (on prendra $N = 100,\ 500,\ 1000,\ 2000,\ 3000$).

En déduire si l'on peut approcher $\mathbb{G}_\alpha$ par une matrice de rang faible ; et si l'on peut utiliser directement la SVD pour effectuer un produit matrice-vecteur rapide.

La SVD tronquée fournit la meilleure approximation de rang faible d'une matrice mais cette méthode est coûteuse en nombre d'opérations. On va maintenant s'intéresser aux méthodes Adaptive Cross Approximation (ACA) qui permettent également de trouver des approximations de rang faible mais en effectuant moins d'opérations. Dans les deux prochaines questions, on ne considère plus la matrice correspondant à la discrétisation de la fonction de Green mais des matrices dont on connait le rang. Pour construire une matrice de rang r, il suffit de sommer r matrices de rang 1 (produit de 2 vecteurs).
La méthode ACA est basée sur la décomposition : $\mathbb{M} = \mathbb{R}_k + \mathbb{B}_k$ où $\mathbb{B}_k$ est l'approximation à l'itération k.

Algorithme ACA avec pivotage complet

On rappelle l'itération k de la méthode ACA avec pivotage complet :

— Trouver le pivot (i^*, j^*) tel que $(i^*, j^*) = \text{argmax}_{ij}|(\mathbb{R}_k)_{ij}|$

— Calculer les vecteurs $\mathbf{u}_{k+1} := \frac{(\mathbb{R}_k)_{ij^*}}{(\mathbb{R}_k)_{i^*j^*}}$ et $\mathbf{v}_{k+1} := (\mathbb{R}_k)_{i^*j}$

— Mettre à jour l'approximation : $\mathbb{B}_{k+1} = \mathbb{B}_k + \mathbf{u}_{k+1}\mathbf{v}_{k+1}^T$

— Mettre à jour le résidu : $\mathbb{R}_{k+1} = \mathbb{R}_k - \mathbf{u}_{k+1}\mathbf{v}_{k+1}^T$

Le critère d'arrêt de la méthode est donné par

$$||\mathbb{M} - \mathbb{B}_k||_F \leq \varepsilon_f ||\mathbb{M}||_F.$$

Question 3 (ACA avec pivotage complet).

(a) Implémenter la méthode ACA avec pivotage complet. Pour une matrice dont vous connaissez le rang exact (par exemple 2 ou 6), en combien d'itérations obtenez vous la décomposition ? Est-ce que la taille de la matrice a une influence ? Quelle est la précision de l'approximation ?

(b) Pour une matrice dont vous connaissez le rang exact, comment évolue le temps de calcul (à rang fixé) mais avec une taille qui augmente ? Comment évolue le temps de calcul à taille fixée mais avec un rang qui augmente ?

(c) Estimez (en justifiant) la complexité théorique de l'algorithme. La vérifier numériquement (tracer la courbe temps de calcul en fonction de la taille de la matrice, pour un rang numérique égal à 2 ou 6).

(d) Que se passe-t-il pour une matrice qui n'a pas un rang **exact** *faible mais un rang* **numérique** *faible ? Faire varier le critère d'arrêt ε_f et comparer la précision de l'approximation obtenue à la précision désirée, i.e. la précision demandée dans le critère d'arrêt.*

(e) Quels sont les avantages et inconvénients de la méthode ?

Pour palier les inconvénients de l'ACA avec pivotage complet, on utilise l'ACA avec pivotage partiel. Le principe est de ne pas maximiser le pivot en parcourant toute la matrice mais seulement une ligne ou une colonne. Le critère d'arrêt de la méthode est donné par

$$||\mathbf{u}_k||_2||\mathbf{v}_k||_2 \leq \varepsilon_p ||\mathbb{B}_k||_F.$$

function AcaPartial(f)
  $k \leftarrow 1,\ \mathcal{P}_l \leftarrow \emptyset,\ \mathcal{P}_c \leftarrow \emptyset$
  $i^* \leftarrow 1$ ▷ La première ligne est choisie comme pivot
  repeat
    $\mathcal{P}_l \leftarrow \mathcal{P}_l \cup \{i^*\}$
    $b_j^k \leftarrow M_{i^*j} - \sum_{\nu=1}^{k-1} a_{i^*}^\nu b_j^\nu$ ▷ Calcul et mise à jour
    $j^* \leftarrow \arg\max_{j \in \tau - \mathcal{P}_c} |b^\nu|,\ \delta \leftarrow b_{j^*}^k$ ▷ Recherche de la colonne pivot
    if $\delta = 0$ **then**
      if $\sigma - \mathcal{P}_l = \emptyset$ **then** ▷ Il n'est plus possible de trouver une ligne pivot.
        return ▷ Pas de convergence
      end if
      $i^* = \min\{i \in \sigma \mid i \notin \mathcal{P}_l\}$ ▷ Ou tout autre choix dans $\sigma - \mathcal{P}_l$
    else ▷ Un pivot non nul est trouvé
      $\mathcal{P}_c \leftarrow \mathcal{P}_c \cup j^*$
      $b \leftarrow b/\delta$
      $a_i^k \leftarrow M_{ij^*} - \sum_{\nu=1}^{k-1} a_i^\nu b_{j^*}^\nu$ ▷ Calcul et mise à jour
      $i^* \leftarrow \arg\max_{i \in \sigma - \mathcal{P}_l} |a^\nu|$ ▷ Recherche de la ligne pivot
      $k \leftarrow k + 1$
    end if
  until Convergence
  return $A := (a^\nu)_{\nu=1,\dots,k}$ $B := (b^\nu)_{\nu=1,\dots,k}$
end function

FIGURE 1 – Algorithme de l'ACA avec pivotage partiel (d'après la thèse de B. Lizé).

Exemple d'application de l'ACA avec pivotage partiel

On considère la matrice $\mathbb{M}$ de rang exact 2 :

$$\mathbb{M} = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ 9.1 & -3 & 11 & 31 \\ 17.6 & -16 & 28 & 80 \\ 26.2 & 50 & -7 & -26 \end{bmatrix}$$

- A la première itération : $i^\star = 1$
- Ligne 1 de $\mathbb{M}$: $a = [6.5\ 31\ -14\ -43]$ et de $\mathbb{R}_0$: $[6.5\ 31\ -14\ -43]$
- Choix du pivot colonne : $j^\star = 4$, $\gamma_1 = -1/43$. On a bien un pivot non nul
- Colonne 4 de $\mathbb{M}$: $b = [-43\ 31\ 80\ -26]^T$ et de $\mathbb{R}_0$: $b = [-43\ 31\ 80\ -26]^T$
- Mise à jour des vecteurs : $u_1 = [1\ -0.7209\ -1.8605\ 0.6047]^T$
- et $v_1 = [6.5\ 31\ -14\ -43]$

$$\mathbb{B}_1 = \begin{bmatrix} 6.5 & 31 & -14 & -43 \\ -4.6860 & -22.3488 & 10.0930 & 31 \\ -12.0930 & -57.6744 & -26.0465 & 80 \\ 3.9302 & 18.7442 & -8.4651 & -26 \end{bmatrix} \text{ et } \mathbb{R}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 13.7860 & 19.3488 & 0.9070 & 0 \\ 29.6930 & 41.6744 & 1.9535 & 0 \\ 22.2698 & 31.25581 & 1.4651 & 0 \end{bmatrix}$$

- $||\mathbb{B}_1||_F = 127.6636$ et $||v_1||_2||u_1||_2$=127.6636, le critère n'est pas satisfait.
- On sait que $i_2^\star = 3$
- Ligne de $\mathbb{M}$: $a = [17.6\ -1628\ 80]$
- Ligne de $\mathbb{R}_1$: $[29.693\ 41.6744\ 1.9536\ 0]$
- Choix pivot $j_2^\star = 2$, $\gamma_1 = 1/41.6744$. On a bien un pivot non nul
- Colonne de $\mathbb{M}$: $b = [31\ -3\ -16\ 50]^T$
- Colonne de $\mathbb{R}_0$: $b = [0\ 19.3488\ 41.6744\ 31.2558]^T$
- Choix du nouveau pivot : $i_4^\star = 4$ (on ne doit pas reprendre une ligne pivot déjà parcourue)
- Nouveaux vecteurs $u_2 = [0\ 0.4643\ 1\ 0.75]^T$
- et $v_2 = [29.6930\ 41.6744\ 1.9535\ 0]$
- $\mathbb{B}_2 = \mathbb{M}$, $||\mathbb{B}_2||_F = 126.0288$ et $||v_2||_2||u_2||_2$=68.2826
- On ne peut plus trouver de pivot non nul.

(a) *Implémenter la méthode ACA avec pivotage partiel. La différence principale avec la version totalement pivotée est la nécessité de stocker la liste des colonnes et lignes déjà choisies comme pivot. Pour une matrice dont vous connaissez le rang exact (par exemple 2 ou 6), en combien d'itérations obtenez vous la décomposition ? Est-ce que la taille de la matrice a une influence ? Quelle est la précision de l'approximation ?*

(b) *Pour une matrice dont vous connaissez le rang exact, comment évolue le temps de calcul (à rang fixé) mais avec une taille qui augmente ? Comment évolue le temps de calcul à taille fixée mais avec un rang qui augmente ?*

(c) *Estimez (en justifiant) la complexité théorique de l'algorithme. La vérifier numériquement (tracer la courbe temps de calcul en fonction de la taille de la matrice, pour un rang numérique égal à 2 et à 6).*

(d) *Que se passe-t-il pour une matrice qui n'a pas un rang* **exact** *faible mais un rang* **numérique** *faible ? Faire varier le critère d'arrêt ε_f et comparer la précision de l'approximation obtenue à la précision désirée, i.e. la précision demandée dans le critère d'arrêt.*

(e) *Quels sont les avantages et inconvénients de la méthode ?*

Suite aux observations faites à la question 2, on subdivise la matrice $\mathbb{G}_\alpha$ en 4 sous-matrices en séparant la liste des points en 2 sous-listes : X_1 et X_2. Sur la Figure 1, le bloc 1 correspond aux interactions entre les points de X_1, le bloc 2 aux interactions entre les points de X_1 et les points de X_2, ...

1	2
3	4

FIGURE 2 – Décomposition de $\mathbb{G}_\alpha$ en quatre sous-blocs.

Question 5 (Partitionment de la matrice en 4 sous-blocs).

(a) *Faire une fonction qui calcule la SVD de chaque sous-bloc. Pour les mêmes valeurs de k et de N que dans la question 2, représenter sur une figure la décroissance des valeurs singulières pour chaque sous-bloc. Que remarquez-vous ? Justifier ce phénomène en utilisant l'expression de la fonction de Green.*

(b) *Quand peut-on considérer que l'on peut effectuer une approximation de rang faible ? On utilisera ce même critère par la suite pour savoir quand tronquer la SVD.*

Question 6 (Partitionment récursif de la matrice).

(a) *Continuer à subdiviser récursivement les blocs de la matrice qui ne peuvent pas être approchés par une matrice de rang faible (ils sont de rang plein). Comment arrêtez vous le processus récursif ?*

(b) Pour les mêmes valeurs de k et de N que dans la question 2, représenter sur un schéma le rang numérique (obtenu en utilisant le critère déterminé lors de la question 5) de chaque bloc à chaque niveau (comme sur la Figure 1 mais en remplaçant le numéro du bloc par son rang), en allant jusqu'à 3 niveaux de subdivision.

(c) Comparer le coût de stockage de la matrice approchée à celui de la matrice pleine. Qu'en déduisez vous ?

Question 7 (Algorithme rapide basé sur ce partitionnement). *Déterminer et implémenter l'algorithme rapide qui permet de calculer le produit matrice-vecteur en utilisant le partitionnement introduit dans la question précédente ainsi que la méthode ACA avec pivotage partiel.*