PROJET DE PROGRAMMATION PARALLÈLE ET DISTRIBUÉE (IATIC4-ISTY)

MÉTHODES DES PUISSANCES

Nahid Emad et France Boillod-Cerneux

1 Description du problème

Soit A une matrice carrée d'ordre n (de n lignes et n colonnes). On considère le problème de *valeur propre* suivant :

$$Au_i = \lambda_i u_i, \text{pour } i = 1, \cdots, n. \tag{1}$$

On suppose que les valeurs propres λ_i sont ordonées de la manière suivante :

$$| \lambda_1 | > | \lambda_2 | \geq \cdots \geq | \lambda_n |$$

Soient $u_1, u_2, \cdots, u_n$ les vecteurs propres associés aux valeurs propres $\lambda_1, \lambda_2, \cdots, \lambda_n$. Le problème consiste à calculer un nombre petit s ($1 \leq s << n$) de valeurs propres λ_i et leur vecteurs propres associés u_i (pour $i = 1, \cdots, s$).

2 Objectif du projet

L'objectif du projet est d'implémenter la méthode des puissances pour le calcul de la plus grande valeur propre λ_1 de la matrice A du problème ci-dessus et son vecteur propres associé u_1. Plus particulièrement, il s'agit d'effectuer une programmation parallèle de cette méthode afin de la rendre plus efficace en termes du temps d'exécution. L'architecture parallèle visée est à mémoire partagée et l'interface de programmation pour le calcul parallèle sur ce type d'architecture est OpenMP (Open Multi-Processing).

3 Méthode des puissances

Soit v un vecteur avec au moins un composant non-nul et, de norme 1. La suite de vecteurs $v, Av, A^2v, A^3v, A^4v, \cdots$ est appelée la suite des puissances A de v. Le calcul de cette suite est à la base de la méthode des puissances qui permet de calculer la plus grande valeur propre de A et son vecteur propre associé. En effet, quand $k \to +\infty$, cette suite converge vers le plus grand vecteur propre de A : $A^k v \to u_1$. Le calcul de la plus grande valeur propre λ_1 est alors effectué selon le processus décrit dans l'algorithme ci-dessous.

Algorithme de la méthode des puissances

Input: A, v, n
Result: λ_1
initialization : $v_1 = \frac{v}{\|v\|_\infty}$
for $k = 1, 2, \cdots$ *jusqu'à la convergence* **do**
  $v_k \leftarrow \frac{1}{\alpha_k} A v_{k-1}$
  $\alpha_k \leftarrow argmax_{i=1,\cdots,n} |(A v_{k-1})_i|$
end
$\lambda_1 \leftarrow \alpha_k$

La valeur α_k est la plus grande en module, des composantes de Av_{k-1}. Dans la pratique, on normalise le vecteur $v_k = Av_{k-1}$.

4 Bonus : La déflation

On peut utiliser la *déflation* dans la méthode des puissances pour calculer d'autres éléments (valeurs, vecteurs) propres dominants.

- Utiliser la version déflatée de la méthode : appliquer la méthode des puissances à la matrice $A + \sigma I$ avec σ un nombre positif (appelé *shift* d'origine).
- Utiliser la version déflatée de la méthode : appliquer la méthode des puissances à la matrice $A + \sigma I$ avec σ un nombre positif (appelé *shift* d'origine).
- Si les valeurs propres sont réelles et ordonnées comme : $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \lambda_n$, le meilleur shift d'origine est : $\sigma_{best} = \frac{\lambda_2 + \lambda_n}{2}$.
- Plus précisément, après avoir calculé le pair (λ_1, u_1), la méthode de déflation permet de calculer la seconde valeur propre de plus grand module. En appliquant la même méthode à la matrice A^T qui a les même valeurs propres que A, on peut calculer son vecteur propre dominant w_1 associé à λ_1. La matrice B suivante a les valeurs propres $\lambda_2, \cdots, \lambda_n$ et zéro. Pour calculer λ_2, u_2, il suffit alors d'appliquer la méthode à la matrice : $B = A - \frac{\lambda_1 u_1^T w_1}{(u_1, w_1)}$

5 Travail à réaliser

1. Le projet doit être fait en binôme.
2. Il s'agit dans un premier temps d'écrire l'algorithme parallèle détaillé. Proposez une analyse apriori de ses performances. Cette analyse consiste à calculer le facteur d'accélération (l'efficacité), le débit, le débit asymptotique, le débit infini, le $N_{1/2}$, etc. ainsi que ses complexités en temps et en espace.

3. Implémentez l'algorithme proposé en utilisant le langage C et les directives de compilation OpenMP. En utilisant les métriques de performances vues en cours (temps d'exécution, débit, etc.), présentez les courbes de performances de votre programme en fonction des paramètres comme la taille du problème, le nombre de cœurs, etc (scalabilité fort et scalabilité faible).
4. Nous considérons que votre programme est séquentiel si le nombre de threads est un. Mesurez l'efficacité de votre programme en calculant le rapport entre le temps de la version parallèle et celui de la version séquentielle.
5. Vérifiez la cohérence entre les performances apriori (point 2) et à posteriori (points 3-4).

6 Rendu du projet

— Un rapport de plus de 6 pages (moins de 10 pages) en format word ou pdf doit être rédigé contenant un résumé de votre travail, les cas tests, les courbes de performances, des charges des threads et des cœurs, votre conclusion ainsi que toutes informations complémentaires que vous jugerez utiles à la compréhension de votre travail.
— Les codes source ainsi que les données des cas test (représentant les courbes de performances présentées dans le rapport) devront être fournis avec le rapport. Attention de ne pas m'envoyer des fichiers binaires (auquel cas une sanction de 3 points vous sera appliquée).
— Les fichiers doivent respecter le format suivant :
  — Nom1Prénom1-nom2Prénom2_codeSourcePUISS.c et
  — Nom1Prénom1-nom2-prénom2_rapportPUISS.pdf (ou word).
— La date limite de l'envoi du projet (rapport et le code source) la date limite pour le rendu du projet est le 9 janvier 2022 avant minuit.