Programmation pour le calcul scientifique
Année : 2018-2019
Formation : L3 Ingénieurie Mathématique

TP7 : DS

A lire attentivement avant de commencer :

L'utilisation de documents est autorisé. A la fin de la séance vous devrez m'envoyez les programmes que vous aurez écrits par email **avec accusé de réception**.

Commentez systématiquement vos programmes. Une question sera considérée comme traitée si les intructions correspondantes et l'affichage des résultats de ces instructions ont été programmés, sont compilables et fonctionnent à l'exécution. Donc tous les programmes doivent être validés par un test même si ce n'est pas demandé explicitement dans la question.

Si vous avez traité certaines questions mais que le code résultant ne produit pas les résultats escomptés, rajoutez à côté des lignes de code correspondantes des commentaires pour décrire le problème rencontré.

Chaque exercice sera traité dans un programme indépendant. L'exercice 1 sera traité dans un seul fichier. Les trois exercices suivants seront structuré en trois fichiers : un fichier contenant le programme principal, un autre les fonctions associées et un troisième (header) les éventuelles classes et prototypes des fonctions.

1 Pointeurs, références, allocation de mémoire

1. Définir un entier k et un pointeur sur un entier p. Initialiser la valeur de k.
2. Modifier la valeur de k en utilisant le pointeur p.
3. Définir q un autre pointeur sur un entier en lui allouant directement une place mémoire.
4. Attribuer une valeur à cette place mémoire.
5. Faire pointer p vers cette place mémoire, et modifier la valeur contenue dans cette place mémoire en utilisant p.
6. Détruire la place mémoire allouée à q.
7. Rajouter le code ci après à votre programme. Créer un pointeur (de type string) qui pointera sur "Euler" ou "Runge-Kutta" suivant la réponse donnée au clavier.

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
  string repA("Euler"), repB("Runge-Kutta");
  cout << "Quel schéma en temps souhaitez-vous utiliser ? " << endl;
  cout << "A) " << repA << endl;
  cout << "B) " << repB << endl;

  char votre_reponse;
  cout << "Votre reponse (A ou B) : ";
  cin >> votre_reponse; //Récupère la réponse de l'utilisateur
```

2 Résolution numérique du Laplacien

On considère la discrétisation du Laplacien en une dimension, avec un pas d'espace $h = 1/N$, pour un nombre total d'inconnues dans le domaine égal à $N - 1$:

$$\frac{2u_i - u_{i+1} - u_{i-1}}{h^2} = f(x_i), \quad 1 \leq i \leq N-1,$$

avec les conditions aux limites $u_0 = a$ et $u_N = b$, où a et b sont deux réels quelconques. On peut montrer que la solution de ce système linéaire vérifie :

$$u_n = nu_1 - (n-1)u_0 + \sum_{i=1}^{n-1}(n-i)h^2 f(x_i), \quad \forall n \geq 2,$$

avec $x_i = ih$ pour tout $0 \leq i \leq N$.

1. Ecrire une fonction qui prend en argument un entier N, deux réels a et b et une fonction f et qui renvoie la valeur de u_1 correspondant au système linéaire ci-dessus.
2. Ecrire une fonction qui prend en argument un entier N, deux réels a et b et une fonction f et qui renvoie un vecteur de type `vector` contenant la solution numérique du système ci-dessus.
3. Dans le programme principal, tester ces fonctions pour le cas où la solution exacte est la fonction $u(x) = sin(2\pi x)$ pour $x \in [0, 1]$.
4. Ecrire une fonction qui calcule l'erreur en norme max entre la solution numérique et la solution exacte dans le cas de la question précédente.

3 Méthode de la puissance itérée

On considère la matrice A_h de la discrétisation du Laplacien en une dimension, avec un pas d'espace $h = 1/N$, pour un nombre total d'inconnues dans le domaine égal à $N - 1$.

1. Ecrire une fonction qui prend en argument un vecteur x de taille $N - 1$ et renvoie un nouveau vecteur qui est le résultat de la multiplication de x par la matrice A_h. Pour simplifier le programme et économiser de la mémoire, on pourra tirer parti de la structure particulière de la matrice A_h.
2. Ecrire une fonction qui prend en argument un vecteur x, un entier *nite*, et calcule les *nite* premières itérations de l'algorithme de la puissance itérée appliqué à la matrice A_h et au vecteur x. Cette fonction renvoie un réel λ qui représente l'approximation à la *nite*-ième itération de la valeur propre dominante de la matrice, et modifie le vecteur x afin qu'il prenne comme valeur l'approximation correspondante du vecteur propre associé.
3. Ecrire une fonction qui prend en argument un vecteur x et un réel λ et renvoie la norme max de $A_h x - \lambda x$.
4. Ecrire un programme qui demande à l'utilisateur la valeur de N et un entier *nite* et affiche à l'écran le résultat de l'algorithme de la puissance itérée appliqué à la matrice A_h au bout de *nite* itérations et une estimation de l'erreur correspondante en norme max.

4 Une classe de nombres complexes

1. Dans un fichier header, déclarer une classe `complexe` avec comme données privées `x` et `y` deux réels représentant la partie réelle et la partie imaginaire d'un nombre complexe.
2. Définissez une fonction `init` membre public de la classe qui prend en argument deux réels `u` et `v` et qui modifie l'instance de la classe en attribuant à `x` la valeur de `u` et à `y` la valeur de `v`.
3. Définissez une fonction `initbis` membre public de la classe qui prend en argument deux réels `u` et `v` et qui renvoie comme résultat l'instance de la classe elle-même en attribuant à `x` la valeur de `u` et à `y` la valeur de `v`, à l'aide du pointeur `this`.
4. Définissez une fonction membre public de la classe qui prend en argument une autre instance de la classe et qui renvoie le produit des deux instances considérées.
5. Définissez une fonction amie de la classe qui prend en argument deux instances de la classe et qui renvoie leur produit.

Théorème — Soit A une matrice carrée d'ordre n et $(\lambda_1,\lambda_2,...,\lambda_n)$ ses valeurs propres. On suppose :

$$|\lambda_1| > \max\left(|\lambda_2|, \ldots, |\lambda_n|\right).$$

On considère la somme directe $\mathbb{C}^n = E \oplus F$ où E est le sous-espace caractéristique de A associé à la valeur propre λ_1, et F est le sous-espace caractéristique de A associé aux autres valeurs propres.

Alors si $w^{(0)} \notin F$, la suite de vecteurs $(w^{(n)})$ définie par la relation de récurrence

$$\forall n \in \mathbb{N},\ w^{(n+1)} = \frac{1}{||Aw^{(n)}||} Aw^{(n)}$$

vérifie

- $w^{(n)} \neq 0$ pour tout $n \in \mathbb{N}$.
- $||Aw^{(n)}|| \longrightarrow |\lambda_1|$ lorsque $n \to \infty$.
- $\left(\frac{\overline{\lambda_1}}{|\lambda_1|}\right)^n w^{(n)} \longrightarrow v$ lorsque $n \to \infty$, où v est un vecteur unitaire de A associé à la valeur propre λ_1.
- Si j est une composante non nulle du vecteur v, alors $\frac{(Aw^{(n)})_j}{w_j^{(n)}} \longrightarrow \lambda_1$ lorsque $n \to \infty$.

FIGURE 1 – *Rappel : Algorithme de la puissance itérée (source : Wikipedia)*

5 Makefile

1. Créer un makefile pour compiler les programmes des exercices précédent.