Université de Pau et des Pays de l'Adour **Année 2018-2019**
Département de Mathématiques

L2-info - Calcul Scientifique

TP-3: Matrices et résolution de systèmes linéaires

Les programmes de chaque exercices sont à déposer sur Elearn, dans le module prévu à cet effet. Vous rendrez également un document texte contenant les réponses aux questions.

Exercice 1. Écrire un programme permettant d'allouer une matrice carrée de taille $n \times n$ à partir d'un entier n saisi par l'utilisateur. Le programme allouera également deux vecteurs u et v tous deux de taille n.

Le programme initialisera la matrice A sous la forme d'une matrice de Vandermonde:

$$A_n = \begin{pmatrix} 1^0 & 1^1 & 1^2 & \dots & 1^{n-1} \\ 2^0 & 2^1 & 2^2 & \dots & 2^{n-1} \\ 3^0 & 3^1 & 3^2 & \dots & 3^{n-1} \\ \vdots & \vdots & & \vdots & \\ n^0 & n^1 & n^2 & \dots & n^{n-1} \end{pmatrix}$$

Le programme devra réaliser l'affichage de la matrice sous la forme d'un simple tableau de nombres.

Le programme calcule ensuite le produit $v = Au$ et affiche le vecteur résultat v. Le vecteur u est un vecteur unité :

$$u = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

Exercice 2. Écrire un programme qui réalise la décomposition LU d'une matrice, en utilisant l'algorithme vu en cours, sans le calcul du second membre.

Tester le programme sur la matrice A puis sur la matrice B :

$$A = \begin{pmatrix} 1 & 2 & 1 & 2 \\ -2 & -3 & 0 & -5 \\ 4 & 8 & 6 & 7 \\ 1 & -1 & 0 & 5 \end{pmatrix}$$

Vérifier le calcul en comparant le résultat du produit LU avec la matrice initiale.

Exercice 3. Reprendre et compléter le programme de l'exercice 2 afin de résoudre le système linéaire. Vous ajouterez le calcul du vecteur intermédiaire y à l'algorithme de décomposition LU.

Résoudre le système $Ax = b$, avec la matrice A de l'exercice 2 et b le vecteur $b = (-1, 1, -1, 1)$

On ajoutera au programme une étape de vérification du résultat: le produit Ax, avec x la solution calculée doit bien être égal à b.

Résoudre le système $Bx = b$ avec B la matrice:

$$B = \begin{pmatrix} 1 & -2 & 4 & 1 \\ 2 & -4 & 7 & -1 \\ 1 & 0 & 6 & 0 \\ 2 & -5 & 7 & 5 \end{pmatrix}$$

Pour quelle raison, le calcul avec B n'est pas possible ? Adapter le programme pour réaliser le calcul.

Exercice 4. Écrire un programme qui calcule les itérés de l'exemple du cours sur la méthode de Jacobi selon les deux réécritures. Calculer le résidu à chaque étape par rapport à la solution $(1, 2)$.

$$\text{Version 1: } \begin{cases} a_{n+1} = \dfrac{5 - b_n}{3} \\ b_{n+1} = \dfrac{5 - a_n}{2} \end{cases} \qquad \text{Version 2: } \begin{cases} b_{n+1} = 5 - 3a_n \\ a_{n+1} = 5 - 2b_n \end{cases}$$

Exercice 5. Écrire un programme qui implémente la méthode de Jacobi pour le système :

$$\begin{pmatrix} 3 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 3 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \\ 1 \\ 1 \\ 2 \end{pmatrix}$$

Votre programme utilisera uniquement un nombre d'étapes $N = 10$ puis $N = 100$.

Le programme devra afficher à l'issue des itérations l'erreur entre la solution obtenue et la solution exacte qui est $\bar{x} = (1, 1, 1, 1, 1)$.

Exercice 6. Écrire un programme qui permet de stocker une matrice creuse au format CSR, vu en cours, puis qui calcule un produit matrice-vecteur. Le programme devra :

1. Allouer la structure de donnée pour le stockage CSR
2. Initialiser la matrice avec la matrice de l'exercice 5
3. Afficher la matrice
4. Réaliser le produit entre cette matrice et le vecteur $(1, 1, 1, 1, 1)$. Vérifier le résultat obtenu. Le calcul du produit sera fait dans une fonction dont la signature donnée ci-dessous avec `b` le vecteur utilisé pour stocker le résultat:

```
void prodSpMatVec(spmat A, double *x, double *b);
```

Indication: pour simplifier la manipulation d'une matrice sous forme CSR, vous utiliserez la structure suivante :

```
typedef struct sparse_matrix {
  double *A; int *IA; int *JA;
  int n;
} spmat;
```

Exercice 7. Reprendre l'exercice 5 et l'adapter pour travailler avec une matrice A stockée en format CSR. La matrice A de taille $n \times n$ est définie par:

$$\begin{pmatrix} 3 & -1 & 0 & & & & 0 \\ -1 & 3 & -1 & 0 & & & \\ 0 & -1 & 3 & -1 & 0 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 0 & -1 & 3 & -1 & 0 \\ 0 & & & & 0 & -1 & 3 \end{pmatrix}$$

Quel est le gain en mémoire en fonction de n dans ce cas, par rapport à un stockage plein?

Pour cela il faut implémenter l'algorithme de la méthode de Jacobi sous la forme matricielle afin de pouvoir utiliser un produit matrice-vecteur adapté aux matrices creuses (vous réutiliserez la fonction développée à l'exercice 6).

Attention: l'algorithme utilise la matrice $\tilde{A}$ et le vecteur d vus en cours.

Adapter la méthode afin de calculer une solution à une précision de 10^-5. Afficher $\max_i |r_i|$ ainsi que le nombre d'étapes réalisées. Comparer les résultats pour $n = 10$, $n = 100$ et $n = 10000$. Que pouvez vous conclure ?