



ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE

Exercices

---

MATH-250 — Analyse numérique

---

Professeur — Buffa Annalisa  
École Polytechnique Fédérale de Lausanne  
2017

**Exercices MATLAB - Analyse Numérique - 2017**  
**Section MA**  
**Prof. A. Quarteroni**  
**Séance 1 - Introduction à Matlab**

**Exercice 1**

On considère les matrices

$$A = \begin{bmatrix} 5 & 3 & 0 \\ 1 & 1 & -4 \\ 3 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 4 & 3 & 2 \\ 0 & 1 & 0 \\ 5 & 0 & 1/2 \end{bmatrix}$$

Créer un répertoire de travail et écrire un fichier ".m" dans lequel placer les instructions pour calculer (sans utiliser de boucles), la matrice  $C = AB$  (produit matriciel) et la matrice  $D$  qui a comme éléments  $D_{ij} = A_{ij}B_{ij}$  (produit composante par composante).

**Solution 1**

Dans le script `ex1.m`, nous avons :

```
clc
clear all
close all

A= [5 3 0; 1 1 -4; 3 0 0];
B= [4 3 2; 0 1 0; 5 0 1/2];

C = A*B
D = A.*B
```

On obtient

```
>> ex1

C =
    20      18      10
   -16       4       0
    12       9       6

D =
    20       9       0
     0       1       0
    15       0       0
```

**Exercice 2**

Définir (sans utiliser de boucles) la matrice diagonale de taille  $n = 5$  dont la diagonale est un vecteur de points équirépartis entre 3 et 6 (i.e.  $[3, 3.75, 4.5, 5.25, 6]$ ).

## Solution 2

```
>> M = diag(linspace(3,6,5))  
  
M =  
  
3.0000      0      0      0  
0    3.7500      0      0  
0      0    4.5000      0  
0      0      0    5.2500  
0      0      0      0    6.0000
```

## Exercice 3

Écrire une fonction pour calculer :

- le produit, composante par composante, entre deux vecteurs  $x$  et  $y$ ;
- le produit scalaire entre les mêmes vecteurs  $x$  et  $y$ ;
- un vecteur dont les éléments sont définis par :

$$v_1 = x_1y_n, \quad v_2 = x_2y_{n-1}, \quad \dots, \quad v_{n-1} = x_{n-1}y_2, \quad v_n = x_ny_1.$$

Utiliser et compléter la définition suivante :

```
function [ElByElProd, ScalProd, v] = operations(x,y)  
%  
% [ELBYELPROD] = OPERATIONS(X,Y) is the element by element product of vectors X  
% and Y. NOTE: X and Y can be row or column vectors.  
% [ELBYELPROD, SCALPROD] = OPERATIONS(X,Y) returns also the scalar product of  
% vectors X and Y.  
% [ELBYELPROD, SCALPROD, V] = OPERATIONS(X,Y) returns also the vector V  
% which is defined as: V(1) = X(1)Y(n)  
%                      V(2) = X(2)Y(n-1)  
%                      ...  
%                      V(N) = X(N)Y(1)  
  
...  
  
return
```

Tester la fonction avec MATLAB.

## Solution 3

```
function [ElByElProd, ScalProd, v] = operations(x,y)  
%  
% [ELBYELPROD] = OPERATIONS(X,Y) is the element by element product of vectors X  
% and Y. NOTE: X and Y can be row or column vectors.  
% [ELBYELPROD, SCALPROD] = OPERATIONS(X,Y) returns also the scalar product of  
% vectors X and Y.  
% [ELBYELPROD, SCALPROD, V] = OPERATIONS(X,Y) returns also the vector V  
% which is defined as: V(1) = X(1)Y(n)  
%                      V(2) = X(2)Y(n-1)  
%                      ...  
%                      V(N) = X(N)Y(1)
```

```

size_x = size(x);
size_y = size(y);

if (size_x(1) ≠ size_y(1) || size_x(2) ≠ size_y(2))
    disp('!!! ERROR: vectors size is different !!!')
    return
end

if (min(size_x) > 1)
    disp('!!! ERROR: X and Y are matrices !!!')
    return
end

ElByElProd = x.*y;

if (size_x(2) ≥ size_x(1))
    ScalProd = x*y';
else
    ScalProd = x'*y;
end

n = length(x);
v = x.*y(end:-1:1);
%% On peut aussi utiliser une boucle for:
% v = [ ];
% for i = 1:n
% v = [v x(i)*y(n-i+1)];
% end

return

```

```

>> x = [1 4 7 2 1 2];
>> y = [0 9 1 4 3 0];
>> [ElByElProd, ScalProd, v] = operations(x,y)

ElByElProd =
0     36      7      8      3      0

ScalProd =
54

v =
0     12     28      2      9      0

```

#### Exercice 4

En utilisant la commande `diag`, définir en MATLAB la matrice  $A \in \mathbb{R}^{n \times n}$  avec  $n = 10$

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 \end{bmatrix}$$

Ensuite, calculer les quantités suivantes :

- a) le déterminant de  $A$  ;
- b) les normes  $\|A\|_1$ ,  $\|A\|_2$ ,  $\|A\|_\infty$  (tapez `help norm` pour voir les options) ;
- c) le rayon spectral de  $A$ , noté  $\rho(A)$ . On rappelle que  $\rho(A) = \max_{j=1,\dots,n} |\lambda_j(A)|$ , où  $\lambda_j(A)$  sont les valeurs propres de  $A$ . Vérifier que, puisque  $A$  est symétrique et définie positive, on a  $\rho(A) = \|A\|_2$  ;

Visualiser les vecteurs propres  $\mathbf{v}_j, j \in \{1, \dots, 10\}$  en utilisant les commandes `[v, lambda]=eig(A)` et `plot(v)`.

En utilisant MATLAB, vérifier que la matrice  $V$  (dont les colonnes sont égales aux vecteurs propres de  $A$ ) permet de diagonaliser la matrice  $A$ . En particulier, vérifier que

$$V^{-1}AV = D = \text{diag}\{\lambda_1, \dots, \lambda_n\}.$$

Visualiser finalement la structure des matrices  $A$ ,  $V$ ,  $D$  (avec la commande `spy`).

#### Solution 4

On peut définir la matrice  $A$  avec la commande suivante

```
n=10;
A=2*diag(ones(1,n))-diag(ones(1,n-1),1)-diag(ones(1,n-1),-1);
A
```

Après, on calcule son déterminant par

```
det(A)
```

```
ans = 11
```

et les normes  $\|A\|_1$ ,  $\|A\|_2$ ,  $\|A\|_\infty$  par

```
nrm1=norm(A,1)
nrm2=norm(A,2)
nrminf=norm(A,inf)
```

et on obtient

```
nrm1 = 4
nrm2 = 3.9190
nrminf = 4
```

Pour évaluer le rayon spectral de  $A$ , il faut calculer ses valeurs propres et en prendre le maximum, en valeur absolue :

```
rho=max(abs(eig(A)))
```

```
rho = 3.9190
```

On peut directement vérifier que,  $A$  étant symétrique et définie positive, on a  $\rho(A) = \|A\|_2$ . Pour calculer et visualiser les vecteurs propres  $\mathbf{v}_j, j \in \{1, \dots, 10\}$ , il faut utiliser les commandes

```
[V, D]=eig(A);
figure
plot(V)
```

ou, si on veut visualiser chaque vecteur propre individuellement, la boucle suivante :

```
figure
for i=1:10
    plot(V(:,i))
    pause
end
```

Avec la première option, on a le graphe de la Figure 1.

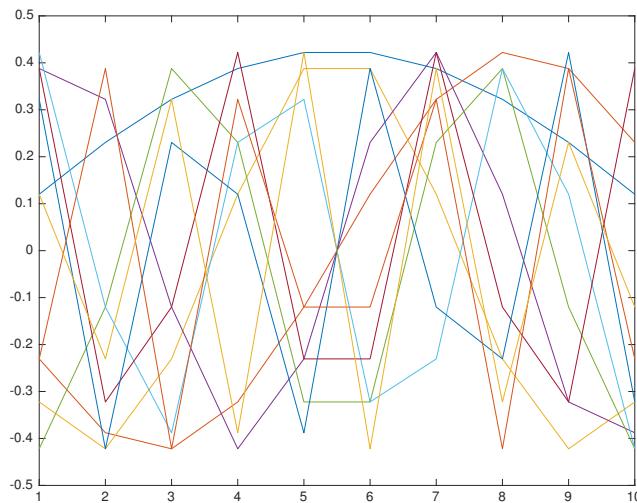


FIGURE 1 – Composantes des vecteurs propres de la matrice  $A$ .

Pour vérifier que la matrice  $V$  (dont les colonnes sont égales aux vecteurs propres de  $A$ ) permet de diagonaliser la matrice  $A$ , c'est-à-dire  $V^{-1}AV = D = \text{diag}\{\lambda_1, \dots, \lambda_n\}$ , il faut utiliser les commandes suivantes :

```
[V,D]=eig(A)
inv(V)*A*V % ou mieux (V\A)*V
D
dif = norm(D-inv(V)*A*V); % ou mieux dif = norm(D-(V\A)*V);
```

on obtient

```
dif = 3.5003e-15
```

Enfin, pour visualiser la structure des matrices  $A$ ,  $V$ ,  $D$ , il faut taper

```
spy(A)
spy(D)
spy(V)
```

On peut observer à la Figure 2 que la matrice  $A$  est tri-diagonale, la matrice  $D$  est diagonale alors que la matrice  $V$  des vecteurs propres est pleine : tous ses éléments sont différentes de zéro.

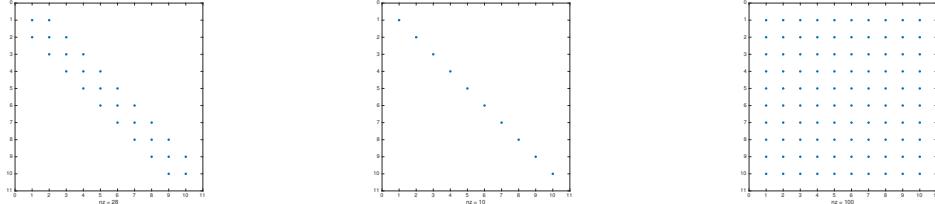


FIGURE 2 – Structure des matrices  $A$ ,  $D$  et  $V$  de gauche à droite.

### Exercice 5

a) Soit

$$f(x) = \frac{x^2}{2} \sin(x), \quad x \in [1, 20]$$

une fonction qu'on veut représenter graphiquement en choisissant 10 points, 20 points et 100 points dans l'intervalle de définition. Écrire un fichier ".m" pour réaliser les trois graphiques sur la même figure et avec trois couleurs différentes. Quelle est la meilleure représentation ?

b) Faire la même chose pour les fonctions :

$$g(x) = \frac{x^3}{6} \cos(\sin(x)) \exp\{-x\} + \left(\frac{1}{1+x}\right)^2, \quad x \in [1, 20]$$

$$h(x) = x(1-x) + \frac{\sin(x)\cos(x)}{x^3}, \quad x \in [1, 20]$$

### Solution 5

a) Le script `ex5.m` est

```
f=@(x) x.^2/2.*sin(x);
x1 = linspace(1,20,10);
x2 = linspace(1,20,20);
x3 = linspace(1,20,100);

figure
plot(x1,f(x1),'r')
hold on
plot(x2,f(x2),'g')
plot(x3,f(x3),'m')
legend('10 points','20 points','100 points')
```

```

xlabel('x')
ylabel('f(x)')
saveas(gcf, 'f5', 'epsc')

```

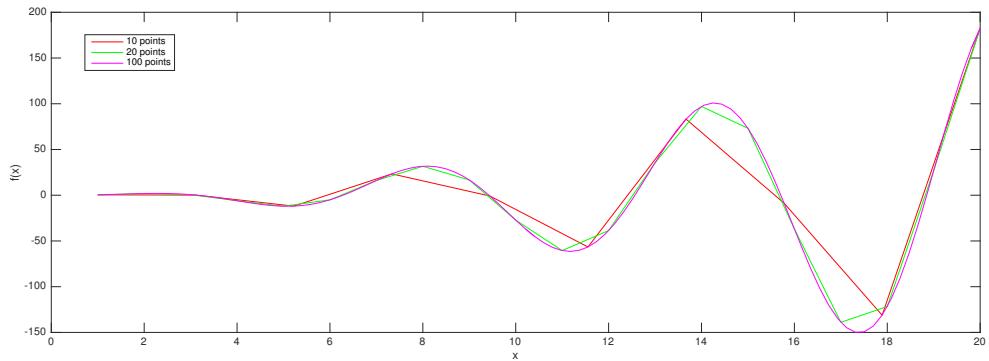


FIGURE 3 – Graphe de la fonction  $f$ .

b) On remplace la fonction  $f$  par les fonctions  $g$  et  $h$  :

```

g=@(x) x.^3/6.*cos(sin(x)).*exp(-x)+(1./(1+x)).^2;
h=@(x) x.* (1-x)+sin(x).*cos(x)./(x.^3);

```

## Exercices Théoriques - Analyse Numérique - 2017

Section MA

Prof. A. Quarteroni

**Séance 2 - Normes matricielles et conditionnement des systèmes linéaires**

### **Exercice 1**

Soit  $\|\cdot\|$  une norme vectorielle. Prouver que la fonction

$$\|A\| = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} \quad (1)$$

est une norme matricielle, en remarquant que la relation (1) est équivalente<sup>1</sup> à

$$\|A\| = \sup_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|.$$

### **Solution 1**

En utilisant l'astuce, on montre directement que cette définition forme une norme matricielle.

a) Si  $\|A\mathbf{x}\| \geq 0$ , alors  $\|A\| = \sup_{\|\mathbf{x}\|=1} \|A\mathbf{x}\| \geq 0$ . De plus

$$\|A\| = \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} = 0 \Leftrightarrow \|A\mathbf{x}\| = 0 \quad \forall \mathbf{x} \neq \mathbf{0},$$

ou encore

$$A\mathbf{x} = \mathbf{0} \quad \forall \mathbf{x} \neq \mathbf{0} \Leftrightarrow A = \mathbf{0}.$$

Donc,  $\|A\| = 0 \Leftrightarrow A = \mathbf{0}$ .

b) Soit un scalaire  $\alpha$ , on a

$$\|\alpha A\| = \sup_{\|\mathbf{x}\|=1} \|\alpha A\mathbf{x}\| = |\alpha| \sup_{\|\mathbf{x}\|=1} \|A\mathbf{x}\| = |\alpha| \|A\|.$$

c) Vérifions enfin l'inégalité triangulaire. Par définition du supremum, si  $\mathbf{x} \neq \mathbf{0}$  alors

$$\frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A\| \Rightarrow \|A\mathbf{x}\| \leq \|A\| \|\mathbf{x}\|,$$

ainsi, en prenant  $\mathbf{x}$  de norme 1, on obtient

$$\|(A + B)\mathbf{x}\| \leq \|A\mathbf{x}\| + \|B\mathbf{x}\| \leq \|A\| + \|B\|,$$

d'où on déduit  $\|A + B\| = \sup_{\|\mathbf{x}\|=1} \|(A + B)\mathbf{x}\| \leq \|A\| + \|B\|$ .

1. En effet, on peut définir pour tout  $\mathbf{x} \neq \mathbf{0}$  un vecteur unitaire  $\mathbf{u} \equiv \mathbf{x}/\|\mathbf{x}\|$  de sorte que (1) s'écrive

$$\|A\| = \sup_{\|\mathbf{u}\|=1} \|A\mathbf{u}\| = \|A\mathbf{w}\| \quad \text{avec } \|\mathbf{w}\| = 1.$$

### Exercice 2

Soit  $\|\cdot\|$  une norme matricielle subordonnée à une norme vectorielle  $\|\cdot\|$ . Prouver que

- a)  $\|Ax\| \leq \|A\| \|x\|$ , i.e.  $\|\cdot\|$  est une norme compatible (ou consistante) avec  $\|\cdot\|$  ;
- b)  $\|I\| = 1$  ;
- c)  $\|AB\| \leq \|A\| \|B\|$ , i.e.  $\|\cdot\|$  est sous-multiplicative.

### Solution 2

- a) Par définition du supremum, si  $x \neq 0$  alors

$$\|A\| = \sup_{y \neq 0} \frac{\|Ay\|}{\|y\|} \geq \frac{\|Ax\|}{\|x\|} \Rightarrow \|Ax\| \leq \|A\| \|x\|.$$

- b) Par la définition de la norme matricielle subordonnée à une norme vectorielle

$$\|I\| = \sup_{x \neq 0} \frac{\|Ix\|}{\|x\|} = 1.$$

- c) Par la compatibilité de la norme, on déduit

$$\|AB\| = \sup_{\|x\|=1} \|ABx\| \leq \sup_{\|x\|=1} \|A\| \|Bx\| = \|A\| \sup_{\|x\|=1} \|Bx\| = \|A\| \|B\|.$$

### Exercice 3

Montrer que, si  $\|\cdot\|$  est une norme matricielle consistante avec une norme vectorielle  $\|\cdot\|$ , alors  $\rho(A) \leq \|A\| \forall A \in \mathbb{C}^{n \times n}$ .

### Solution 3

Si  $\lambda$  est une valeur propre de  $A$ , alors il existe  $v \neq 0$ , vecteur propre de  $A$ , tel que  $Av = \lambda v$ . Ainsi, puisque  $\|\cdot\|$  est consistante,

$$|\lambda| \|v\| = \|\lambda v\| = \|Av\| \leq \|A\| \|v\|$$

et donc  $|\lambda| \leq \|A\|$ . Cette inégalité étant vraie pour toute valeur propre de  $A$ , elle l'est en particulier quand  $|\lambda|$  est égale au rayon spectral.

### Exercice 4

Étant donnée la matrice  $A \in \mathbb{R}^{2 \times 2}$ ,  $a_{11} = a_{22} = 1$ ,  $a_{12} = \gamma$ ,  $a_{21} = 0$ , vérifier que pour  $\gamma \geq 0$ ,  $K_\infty(A) = K_1(A) = (1 + \gamma)^2$ . Soit  $Ax = b$  le système linéaire où  $b$  est tel que  $x = (1 - \gamma, 1)^\top$  soit la solution. Trouver une majoration du type

$$\frac{\|\delta x\|_\infty}{\|x\|_\infty} \leq C \frac{\|\delta b\|_\infty}{\|b\|_\infty}$$

avec  $C > 0$  une constante qui ne dépend que de  $\|A^{-1}\|_\infty$ ,  $\|b\|_\infty$  et  $\|x\|_\infty$ . Le vecteur  $(x + \delta x)$  est la solution du système perturbé  $A(x + \delta x) = (b + \delta b)$  avec  $\delta b$  une perturbation du vecteur  $b$ . Le problème est-il bien conditionné par rapport à  $\gamma \rightarrow \infty$  ?

#### Solution 4

On a

$$A = \begin{bmatrix} 1 & \gamma \\ 0 & 1 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} 1 & -\gamma \\ 0 & 1 \end{bmatrix}.$$

Ainsi, puisque  $\gamma \geq 0$ ,

$$\begin{aligned} \|A\|_1 &= \max_{j=1,2} \sum_{i=1}^2 |a_{ij}| = \max \{1, 1 + \gamma\} = 1 + \gamma, \\ \|A\|_\infty &= \max_{i=1,2} \sum_{j=1}^2 |a_{ij}| = \max \{1 + \gamma, 1\} = 1 + \gamma, \\ \|A^{-1}\|_1 &= \max \{1, 1 + \gamma\} = 1 + \gamma, \\ \|A^{-1}\|_\infty &= \max \{1 + \gamma, 1\} = 1 + \gamma. \end{aligned}$$

Par conséquent,

$$K_1(A) = \|A\|_1 \|A^{-1}\|_1 = K_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty = (1 + \gamma)^2.$$

On a

$$\mathbf{b} = A\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

en particulier  $\|\mathbf{b}\|_\infty = 1$ . En perturbant le second membre du système  $A\mathbf{x} = \mathbf{b}$  (on ne perturbe pas la matrice), on obtient un système perturbé de la forme

$$A(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b},$$

donc

$$\delta\mathbf{x} = A^{-1}\delta\mathbf{b}.$$

On en tire que

$$\|\delta\mathbf{x}\|_\infty \leq \|A^{-1}\|_\infty \|\delta\mathbf{b}\|_\infty.$$

En divisant par  $\|\mathbf{x}\|_\infty$ , on trouve

$$\frac{\|\delta\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \leq \frac{\|A^{-1}\|_\infty}{\|\mathbf{x}\|_\infty} \|\delta\mathbf{b}\|_\infty.$$

En plus, puisque  $\|\mathbf{b}\|_\infty = 1$ , on peut écrire

$$\frac{\|\delta\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \leq \underbrace{\frac{\|A^{-1}\|_\infty}{\|\mathbf{x}\|_\infty}}_C \frac{\|\delta\mathbf{b}\|_\infty}{\|\mathbf{b}\|_\infty}$$

avec  $C = \frac{\|A^{-1}\|_\infty}{\|\mathbf{x}\|_\infty}$  la constante cherchée. On a donc que

$$C = \frac{1 + \gamma}{\max \{1, |1 - \gamma|\}}.$$

On voit bien que  $C \rightarrow 1$  quand  $\gamma \rightarrow \infty$ . Ainsi, pour le cas particulier de  $\mathbf{b} = (1, 1)^\top$ , le problème est bien conditionné. Remarquons que, dans le cas général ( $\mathbf{b}$  arbitraire), le problème est mal conditionné pour  $\gamma$  grand. En effet,  $K_\infty(A) \rightarrow \infty$  quand  $\gamma \rightarrow \infty$ . Cet exercice met en évidence que le fait d'avoir une matrice avec un grand conditionnement n'empêche pas nécessairement le système global d'être bien conditionné pour des choix particuliers du second membre  $\mathbf{b}$ .

### Exercice 5

Supposons que  $\|\delta A\| \leq \gamma \|A\|$ ,  $\|\delta \mathbf{b}\| \leq \gamma \|\mathbf{b}\|$  avec  $\gamma \in \mathbb{R}^+$  et  $\delta A \in \mathbb{R}^{n \times n}$ ,  $\delta \mathbf{b} \in \mathbb{R}^n$ . On veut montrer que, si  $\gamma K(A) < 1$ , on a les inégalités suivantes :

$$\frac{\|\mathbf{x} + \delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{1 + \gamma K(A)}{1 - \gamma K(A)}, \quad (2)$$

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{2\gamma}{1 - \gamma K(A)} K(A). \quad (3)$$

- a) Si  $C$  est une matrice carrée telle que  $\rho(C) < 1$ , on sait (voir le Théorème 1.5 du livre) que  $I - C$  est inversible. Montrer que dans ce cas on a

$$\frac{1}{1 + \|C\|} \leq \|(I - C)^{-1}\| \leq \frac{1}{1 - \|C\|}. \quad (4)$$

où  $\|\cdot\|$  est une norme matricielle subordonnée à une norme vectorielle telle que  $\|C\| \leq 1$ .

- b) Montrer l'inégalité (2) en utilisant le résultat du point 1 et le fait que  $(A + \delta A)(\mathbf{x} + \delta \mathbf{x}) = \mathbf{b} + \delta \mathbf{b}$ .  
c) Montrer l'inégalité (3). Suggestion : utiliser l'inégalité triangulaire  $\|\mathbf{x} + \delta \mathbf{x}\| \leq \|\mathbf{x}\| + \|\delta \mathbf{x}\|$ .

### Solution 5

- a) Puisque<sup>2</sup>  $\|I\| = 1$ , on a (Exercice 2, Série 2)

$$1 = \|I\| \leq \|I - C\| \|(I - C)^{-1}\| \leq (1 + \|C\|) \|(I - C)^{-1}\|,$$

ce qui donne la première égalité de (4). Pour la seconde, en remarquant que  $I = I - C + C$  et en multipliant à droite les deux membres par  $(I - C)^{-1}$ , on a  $(I - C)^{-1} = I + C(I - C)^{-1}$ . En prenant les normes, on obtient

$$\|(I - C)^{-1}\| \leq 1 + \|C\| \|(I - C)^{-1}\|,$$

d'où on déduit la seconde inégalité, puisque  $\|C\| < 1$ .

- b) Soit

$$(A + \delta A)(\mathbf{x} + \delta \mathbf{x}) = \mathbf{b} + \delta \mathbf{b}.$$

Alors, on a

$$(I + A^{-1}\delta A)(\mathbf{x} + \delta \mathbf{x}) = \mathbf{x} + A^{-1}\delta \mathbf{b}.$$

---

2. On suppose (toujours) que  $\|\cdot\|$  est une norme matricielle subordonnée à une norme vectorielle.

De plus, puisque  $\gamma K(A) < 1$  et  $\|\delta A\| \leq \gamma \|A\|$  on a

$$\|A^{-1}\delta A\| \leq \|A^{-1}\| \|\delta A\| \leq \gamma \|A^{-1}\| \|A\| = \gamma K(A) < 1.$$

Alors,  $\rho(A^{-1}\delta A) < 1$  et  $I + A^{-1}\delta A$  est inversible. En prenant l'inverse de cette matrice et en passant aux normes, on obtient

$$\|\mathbf{x} + \delta \mathbf{x}\| \leq \left\| (I + A^{-1}\delta A)^{-1} \right\| (\|\mathbf{x}\| + \gamma \|A^{-1}\| \|\mathbf{b}\|).$$

Alors, l'inégalité du point 1 donne

$$\|\mathbf{x} + \delta \mathbf{x}\| \leq \frac{1}{1 - \|\delta A\|} (\|\mathbf{x}\| + \gamma \|A^{-1}\| \|\mathbf{b}\|),$$

ce qui implique

$$\frac{\|\mathbf{x} + \delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{1 + \gamma K(A)}{1 - \gamma K(A)},$$

puisque  $\|\delta A\| \leq \gamma K(A)$  et  $\|\mathbf{b}\| \leq \|A\| \|\mathbf{x}\|$ .

- c) Montrons à présent que l'inéquation (3) est correcte. En retranchant  $A\mathbf{x} = \mathbf{b}$  de (2), on a

$$A\delta \mathbf{x} = -\delta A(\mathbf{x} + \delta \mathbf{x}) + \delta \mathbf{b}.$$

En prenant l'inverse de  $A$  et en passant aux normes, on obtient l'inégalité suivante

$$\begin{aligned} \|\delta \mathbf{x}\| &\leq \|A^{-1}\delta A\| \|\mathbf{x} + \delta \mathbf{x}\| + \|A^{-1}\| \|\delta \mathbf{b}\| \\ &\leq \gamma K(A) \|\mathbf{x} + \delta \mathbf{x}\| + \gamma \|A^{-1}\| \|\mathbf{b}\| \end{aligned}$$

En divisant les deux membres par  $\|\mathbf{x}\|$  et en utilisant l'inégalité triangulaire  $\|\mathbf{x} + \delta \mathbf{x}\| \leq \|\mathbf{x}\| + \|\delta \mathbf{x}\|$ , on obtient finalement (3).

## Exercice 6

- a) Soit  $A \in \mathbb{R}^{n \times n}$  une matrice symétrique définie positive et soient  $\lambda_i$  et  $\mathbf{v}_i$ ,  $i = 1, \dots, n$ , les valeurs propres et les vecteurs propres de  $A$ . Montrer que si  $\mathbf{x}$  est la solution du système linéaire  $A\mathbf{x} = \mathbf{b}$ , alors

$$\mathbf{x} = \sum_{i=1}^n (c_i / \lambda_i) \mathbf{v}_i,$$

où  $c_i$  est la  $i$ -ème composante de  $\mathbf{b}$  dans la base des vecteurs propres de  $A$ .

- b) On se donne maintenant le système linéaire  $A\mathbf{x} = \mathbf{b}$  suivant

$$\begin{bmatrix} 1001 & 1000 \\ 1000 & 1001 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

où  $A$  est mal-conditionnée avec les valeurs propres  $\lambda_1 = 1$ ,  $\lambda_2 = 2001$ . En décomposant le second membre sur la base des vecteurs propres de la matrice  $A$ , expliquer pourquoi,

lorsque  $\mathbf{b} = (2001, 2001)^\top$ , une petite perturbation  $\delta\mathbf{b} = (1, 0)^\top$  produit de grandes variations dans la solution, et réciproquement, si  $\mathbf{b} = (1, -1)^\top$ , une petite variation  $\delta\mathbf{x} = (0.001, 0)^\top$  dans la solution induit de grandes variations dans  $\mathbf{b}$ .

### Solution 6

- a) Puisque  $A$  est une matrice symétrique, il existe une matrice  $V$  orthogonale et une matrice  $D$  diagonale dont tous les coefficients sont réels, telles que

$$V^{-1}AV = D = \text{diag}\{\lambda_1, \dots, \lambda_n\}$$

ou, de façon équivalente,  $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$  pour  $i = 1, \dots, n$ , de sorte que les vecteurs colonnes de  $V$  soient les vecteurs propres de  $A$ . De plus, les vecteurs propres sont deux à deux orthogonaux (et peuvent être normalisés : donc, on a que  $\mathbf{v}_j^\top \mathbf{v}_l = \delta_{jl}$ , où  $\delta_{jl}$  est le symbole de Kronecker) et on déduit que les vecteurs propres d'une matrice symétrique sont orthogonaux et engendrent l'espace  $\mathbb{R}^n$  tout entier.

Donc, soient  $\mathbf{b} = \sum_{i=1}^n c_i \mathbf{v}_i$  le membre de droite du système linéaire  $A\mathbf{x} = \mathbf{b}$  et  $\mathbf{x}$  sa solution ; en écrivant aussi  $\mathbf{x}$  dans la base des vecteurs propres de  $A$ , on a :

$$A\mathbf{x} = A \sum_{i=1}^n x_i \mathbf{v}_i = \sum_{i=1}^n c_i \mathbf{v}_i.$$

Et, puisque  $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$ , on trouve

$$\sum_{i=1}^n x_i \lambda_i \mathbf{v}_i = \sum_{i=1}^n c_i \mathbf{v}_i.$$

Donc on trouve

$$\sum_{i=1}^n (\lambda_i x_i - c_i) \mathbf{v}_i = 0,$$

c'est-à-dire  $\lambda_i x_i = c_i$  et

$$\mathbf{x} = \sum_{i=1}^n (c_i / \lambda_i) \mathbf{v}_i.$$

- b) Les vecteurs propres de la matrice  $A$  sont  $\mathbf{v}_1 = (1, -1)^\top$  (qui correspond à  $\lambda_1 = 1$ ) et  $\mathbf{v}_2 = (1, 1)^\top$  (qui correspond à  $\lambda_2 = 2001$ ). Soit  $\mathbf{b} = (2001, 2001)^\top$  et  $\delta\mathbf{b} = (1, 0)^\top$ . Alors,

$$\mathbf{b} + \delta\mathbf{b} = \begin{bmatrix} 2001 \\ 2001 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 2001\mathbf{v}_2 + \frac{1}{2}(\mathbf{v}_1 + \mathbf{v}_2) = \frac{1}{2}\mathbf{v}_1 + \frac{4003}{2}\mathbf{v}_2.$$

Si on écrit la solution  $\mathbf{x}$  comme combinaison linéaire des vecteurs propres, on trouve

$$\mathbf{x} = \frac{1}{2}\mathbf{v}_1 + \frac{\frac{4003}{2}}{2001}\mathbf{v}_2 = \frac{1}{2}\mathbf{v}_1 + \frac{4003}{4002}\mathbf{v}_2 \approx \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}.$$

Ainsi on voit que l'erreur  $\delta\mathbf{x}$  par rapport à la solution exacte  $\mathbf{x} = (1, 1)^\top$  est  $\delta\mathbf{x} \approx (0.5, -0.5)^\top$ .

Réiproquement, soit  $\mathbf{b} = (1, -1)^\top$ . La solution exacte du système est  $\mathbf{x} = (1, -1)^\top$ . On exprime la solution perturbée par rapport aux vecteurs propres :

$$\mathbf{x} + \delta\mathbf{x} = \begin{bmatrix} 1.001 \\ -1 \end{bmatrix} = \frac{2.001}{2}\mathbf{v}_1 + \frac{0.001}{2}\mathbf{v}_2,$$

d'où  $c_1 = 2.001/2$  et  $c_2 = 0.001/2$ . Donc

$$\mathbf{b} + \delta\mathbf{b} = \begin{bmatrix} 2.001 \\ 0 \end{bmatrix}$$

et  $\delta\mathbf{b} = (1.001, 1)^\top$ .

*Remarque.* Le système linéaire de cet exercice pourrait être obtenu de l'analyse d'une barre rigide attachée dans sa partie central à un ressort de raideur 4000 et connectée aux extrémités à deux ressorts de raideur 1 (voir Figure 1 ci-dessous). On applique des forces  $b_1$  et  $b_2$  aux extrémités de la barre et on observe ses déplacements verticaux  $x_1$  et  $x_2$ . Si les forces  $b_1$  et  $b_2$  sont équilibrées (par exemple  $b_1 = 2001$ ,  $b_2 = 2001$ ), alors de petits changements  $\delta\mathbf{b}$  engendrent des mouvements significatifs de la barre (grand  $\delta\mathbf{x}$ ). A l'inverse, si les forces ne sont pas équilibrées (par exemple  $b_1 = 1$ ,  $b_2 = -1$ ), alors on peut obtenir de petits déplacements  $\delta\mathbf{x}$  même si on impose de forts changements  $\delta\mathbf{b}$  sur les forces exercées.

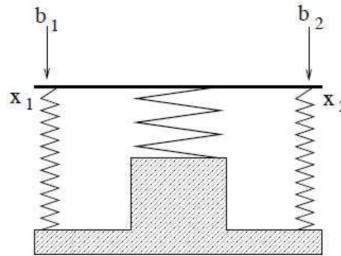


FIGURE 1 – Ressort

**Exercices Théoriques - Analyse Numérique - 2017**  
**Section MA**  
**Prof. A. Quarteroni**  
**Séance 3 - Systèmes linéaires : méthodes directes**

**Exercice 1**

Considérons le cas particulier d'un système linéaire dont la matrice est tri-diagonale et inversible :

$$A = \begin{bmatrix} a_1 & c_1 & & 0 \\ b_2 & a_2 & \ddots & \\ \ddots & \ddots & c_{n-1} & \\ 0 & b_n & a_n & \end{bmatrix}.$$

- a) Montrer qu'il existe deux matrices bi-diagonales  $L$  et  $U$  de la forme

$$L = \begin{bmatrix} 1 & & & 0 \\ \beta_2 & 1 & & \\ & \ddots & \ddots & \\ 0 & & \beta_n & 1 \end{bmatrix}, \quad U = \begin{bmatrix} \alpha_1 & \gamma_1 & & 0 \\ & \alpha_2 & \ddots & \\ & & \ddots & \gamma_{n-1} \\ 0 & & & \alpha_n \end{bmatrix},$$

telles que  $A = LU$ , et donner les expressions des coefficients  $\alpha_i$ ,  $\beta_i$  et  $\gamma_i$  en fonction des coefficients de  $A$ . Ces formules sont connues sous l'appellation d'*Algorithme de Thomas*.

- b) Obtenir les formules résultantes de l'extension de l'algorithme de Thomas à la résolution du système  $A\mathbf{x} = \mathbf{f}$ , avec  $\mathbf{f} = (f_i)_{i=1}^n \in \mathbb{R}^n$ , donnée par
- (a) trouver  $\mathbf{y}$  tel que  $L\mathbf{y} = \mathbf{f}$ ;
  - (b) trouver  $\mathbf{x}$  tel que  $U\mathbf{x} = \mathbf{y}$ .
- c) Combien d'opérations virgule flottante requiert l'algorithme précédent ?

**Solution 1**

- a) On appelle  $L_i$  la  $i$ -ième ligne de  $L$  et  $U_j$  la  $j$ -ième colonne de  $U$ . Il faut vérifier que

$$L_i U_j = [A]_{ij}.$$

Pour la matrice  $A$ , on a des éléments non nuls seulement pour  $i = j$  ou  $i = j \pm 1$ . On obtient les équations suivantes :

- si  $i = j$ , on a  $\beta_i \gamma_{i-1} + \alpha_i = a_i$  ;
- si  $i = j - 1$ , on a  $\gamma_i = c_i$  ;
- si  $i = j + 1$ , on a  $\beta_i \alpha_j = b_i$ .

Donc,  $\alpha_i$ ,  $\beta_i$  et  $\gamma_i$  s'obtiennent facilement avec les relations suivantes :

$$\alpha_1 = a_1, \quad \beta_i = \frac{b_i}{\alpha_{i-1}}, \quad \alpha_i = a_i - \beta_i c_{i-1}, \quad \gamma_i = c_i. \quad (1)$$

- b) La résolution du système  $A\mathbf{x} = \mathbf{f}$  revient à résoudre deux systèmes bidiagonaux,  $L\mathbf{y} = \mathbf{f}$  et  $U\mathbf{x} = \mathbf{y}$ , pour lesquels on a les formules suivantes :

$$\begin{aligned}(L\mathbf{y} = \mathbf{f}) : \quad & y_1 = f_1, \quad y_i = f_i - \beta_i y_{i-1}, \quad i = 2, \dots, n, \\ (U\mathbf{x} = \mathbf{y}) : \quad & x_n = \frac{y_n}{\alpha_n}, \quad x_i = \frac{y_i - \gamma_i x_{i+1}}{\alpha_i}, \quad i = n-1, \dots, 1.\end{aligned}\tag{2}$$

- c) L'algorithme requiert  $8n - 7$  flops :  $3(n-1)$  flops pour la factorisation (1) et  $5n - 4$  flops pour la substitution (2).

## Exercice 2

On considère un câble élastique qui occupe au repos le segment  $[0, 1]$ , fixé aux extrémités, sur lequel on applique une force d'intensité  $f(x)$ . Son déplacement au point  $x$ ,  $u(x)$ , est la solution de l'équation différentielle suivante :

$$\begin{aligned}-u''(x) &= f(x), \quad x \in (0, 1), \\ u(0) &= 0, \quad u(1) = 0.\end{aligned}\tag{3}$$

Soit  $N \in \mathbb{N}$ ,  $h = 1/N$  et  $x_i = ih$  pour  $i = 0, \dots, N$ ; pour approcher la solution  $u(x)$ , on considère la discrétisation de l'intervalle  $(0, 1)$  en  $N$  sous-intervalles  $(x_i, x_{i+1})$ , et on construit une approximation  $u_i$  de  $u(x_i)$  par la méthode des différences finies. Cette méthode requiert de résoudre numériquement le système linéaire tridiagonal  $A\mathbf{u} = \mathbf{b}$  qui suit :

$$\frac{1}{h^2} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-2}) \\ f(x_{N-1}) \end{bmatrix}\tag{4}$$

où  $\mathbf{u} = [u_1, u_2, \dots, u_{N-1}]^\top$  et  $\mathbf{b} = [f(x_1), f(x_2), \dots, f(x_{N-1})]^\top$ . Plus  $N$  est grand, plus l'approximation sera précise et plus la taille du système linéaire à résoudre sera élevée.

- a) On suppose que la force appliquée soit  $f(x) = x(1-x)$  et on prend  $N = 20$  intervalles. Construire la matrice  $A$  et le vecteur  $\mathbf{b}$  correspondants, à l'aide des commandes suivantes :

```
f = @(x) x.*(1-x);
N = 20; h = 1/N;
x = linspace(h, 1-h, N-1)';
% on transpose pour avoir un vecteur colonne
b = f(x);
A = (N^2)*(diag(2*ones(N-1, 1), 0) - diag(ones(N-2, 1)) - diag(ones(N-2, 1), -1));
```

Calculer la factorisation  $LU$  de  $A$  avec la commande MATLAB `lu`. Vérifier que la matrice de permutation  $P$  est l'identité (on sait de la théorie qu'aucune permutation de lignes n'est effectuée dès que la matrice est symétrique définie positive, ce qui est le cas de  $A$ ).

Calculer également la factorisation de Cholesky de  $A$  (commande `chol`) et remarquer qu'elle diffère de la précédente.

- b) Mettre en oeuvre l'algorithme de *substitution directe* pour la résolution d'un système triangulaire inférieure et l'algorithme de *substitution rétrograde* pour la résolution d'un système triangulaire supérieure. Utiliser et compléter les fonctions suivantes :

```
function [x] = subs_directe(L, b)
%
% [X] = SUBS_DIRECTE(L, B) resout le systeme triangulaire inferieur L*X = B
%
...
return
```

```
function [x] = subs_retrograde(U, b)
%
% [X] = SUBS_RETROGRADE(L, B) resout le systeme triangulaire superieur ...
% U*X = B
%
...
return
```

Calculer la solution du système linéaire  $A\mathbf{u} = \mathbf{b}$  à partir de la factorisation  $A = LU$ , en utilisant les fonctions `subs_directe` et `subs_retrograde` pour résoudre les deux systèmes triangulaires.

- c) A l'aide de la commande `plot`, représenter le déplacement  $\mathbf{u}$  du câble aux noeuds  $x_i$  définis au point 1.  
d) Etudier le comportement du conditionnement de la matrice  $A$ ,  $K(A)$ , lorsque  $N$  augmente, en traçant le graphe des valeurs de  $K(A)$  pour  $N = 10, 20, \dots, 120$ . Tracer le graphe bilogarithmique des mêmes valeurs avec la commande `loglog`. Quel type de courbe obtient-on ? Si on suppose une relation linéaire  $\log_{10} K(A) = m \log_{10} N + c$  pour le graphe bilogarithmique, alors on a  $K(A) = CN^m$  (avec  $C = 10^c$ ) : calculer les constantes  $m$  et  $C$ . De combien  $K(A)$  croît-il lorsque on double le nombre  $N$  des sous-intervalles ?

## Solution 2

- a) On définit la matrice  $A$  et le vecteur  $\mathbf{b}$  avec les commandes suggérées dans l'énoncé :

```
f = @(x) x.*(1-x);
N = 20; h = 1/N;
x = linspace(h, 1-h, N-1)';
b = f(x);
A = (N^2)*(diag(2*ones(N-1, 1),0) - diag(ones(N-2,1),1) - diag(ones(N-2,1),-1));
```

Ensuite, on calcule la factorisation  $LU$  de  $A$ . En général, MATLAB peut décider d'effectuer des permutations de lignes pendant le processus de factorisation, ce qui mène à une factorisation  $PA = LU$ . Ainsi, la syntaxe de la commande `lu` à utiliser est la suivante :

```
[L,U,P] = lu(A);
```

De cette façon, on a stocké dans la variable `P` la matrice de permutation  $P$ . Dans notre cas, on peut vérifier que `P` est la matrice identité : par exemple, on peut calculer l'écart maximal entre les éléments de  $P$  et de  $I$ , et on obtient :

```

max(max(abs(P - eye(N-1))))
ans =
0

```

Donc, MATLAB calcule la factorisation  $LU$  sans permutation. Les facteurs ont été stockés dans les variables  $L$  et  $U$ . Ces facteurs diffèrent du facteur  $H$  de la factorisation de Cholesky  $A = HH^\top$ , que l'on calcule avec la commande

```
H = chol(A)';
```

car

```

max(max(abs(L - H)))
ans =
27.2843

```

b) On trouve

```

function [x] = subs_directe(L,b)
%
% [X] = SUBS_DIRECTE(L,B) resout le systeme triangulaire inferieure L*X = B
%

x = zeros(size(b,1), 1);

[m,n]=size(L);

if m ~= n
    disp('Error: the matrix is not square!');
    x = [];
    return
end

if m ~= length(b)
    disp(['Error: the dimension of the matrix and of the vector' ...
        ' are not consistent!']);
    x = [];
    return
end

l = min(diag(abs(L)));

if l == 0
    disp('Error: the matrix is singular');
    x = [];
    return
end

for j=1:n
    x(j)=(b(j) - L(j,1:j-1)*x(1:j-1))/L(j,j);
end

return
end

```

```

function [x] = subs_retrograde(U,b)
%
% [X] = SUBS_RETROGRADE(L,B) resout le systeme triangulaire superieur U*X ...
%      = B
%
x = zeros(size(b,1), 1);

[m,n]=size(U);

if m ~= n
    disp('Error: the matrix is not square!');
    x = [];
    return
end

if m ~= length(b)
    disp(['Error: the dimension of the matrix and of the vector' ...
        ' are not consistent!']);
    x = [];
    return
end

l = min(diag(abs(U)));

if l == 0
    disp('Error: the matrix is singular');
    x = [];
    return
end

for j=n:-1:1
    x(j)=(b(j) - U(j,j+1:n)*x(j+1:n))/U(j,j);
end

return
end

```

D'après le cours, on exploite la factorisation  $LU$  de la façon suivante :

```

y = subs_directe(L,b);
u = subs_retrograde(U,y);

```

- c) La Figure 1 montre les déplacements  $u_i$  aux noeuds  $x_i$  (cercles rouges). Si  $f(x)$  est un polynôme, il est facile de trouver la solution exacte du problème différentiel : dans notre cas<sup>3</sup>, on a  $u(x) = x^4/12 - x^3/6 + x/12$ . On a ajouté le graphe de  $u(x)$  afin de montrer qu'avec  $N = 20$  sous-intervalles, on obtient déjà une solution approchée assez précise. La Figure 1 a été obtenue par le script **ex2exact.m** suivant.

```

clc; clear all; close all

```

- 
3. Comme  $u''(x) = -x + x^2$ , on intègre deux fois et on trouve  $u(x) = x^4/12 - x^3/6 + ax + b$ , où  $a$  et  $b$  sont deux constantes que l'on trouve en imposant les conditions aux bords :

$$u(0) = 0 \implies b = 0, \quad u(1) = 0 \implies a = 1/12.$$

```

f = @(x) x.*(1-x);
N = 20; h = 1/N;
x = linspace(h, 1-h, N-1)';
b = f(x);
A = (N^2)*(diag(2*ones(N-1, 1), 0) - diag(ones(N-2, 1), 1) - ...
    diag(ones(N-2, 1), -1));
[L,U,P] = lu(A);
y = subs_directe(L,b);
u = subs_retrograde(U,y);

u_exact = 'x.^4 / 12 - x.^3 / 6 + x / 12';
fplot(u_exact, [0,1]);
hold on;

plot(x, u, 'or');
legend('u_{exact}', 'u_i')
saveas(gcf, 'ex2_uexact', 'epsc')

```

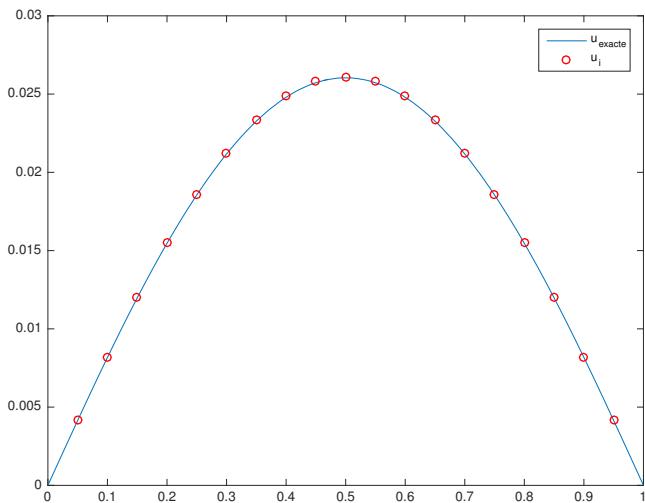


FIGURE 1 – Solution exacte  $u(x)$  et déplacements approchés  $u_i$

Il faut remarquer que la taille du vecteur  $\mathbf{u}$  est  $N - 1$ ; en effet seules les valeurs des déplacements aux noeuds  $x_i$  pour  $i = 1, \dots, N - 1$  sont inconnues, car  $u_0 = u_N = 0$  (conditions aux bords).

- d) Pour chaque valeur de  $N$ , on a une matrice  $A$  différente. Donc, il faut coder une boucle qui, pour  $N = 10, 20, 30, \dots, 120$ , construit cette matrice et calcule son conditionnement. Cette valeur sera ensuite mémorisée dans un vecteur  $\mathbf{k}$ . La boucle peut se coder comme suit :

```

for N = 10:10:120
    h = 1/N;
    A=(N^2)*(diag(2*ones(N-1, 1), 0)-diag(ones(N-2, 1), 1)-diag(ones(N-2, 1), -1));
    k(N/10) = cond(A);
    disp(sprintf('N = %i: K(A) = %e', N, k(N/10)));
    % ceci pour afficher les valeurs calculees
end

```

et on obtient

```

N = 10: K(A) = 3.986346e+01
N = 20: K(A) = 1.614476e+02

(...)

N = 110: K(A) = 4.903279e+03
N = 120: K(A) = 5.835434e+03

```

Le graphe de  $K(A)$  en fonction de  $N$  (commande `plot([10:10:120], k)`) et le graphe bi-logarithmique (commande `loglog([10:10:120],k); grid on`) sont affichés en Figure 2. On peut trouver cette figure avec le script `ex2bilog.m`.

```

for N = 10:10:120
    h = 1/N;
    A=(N-2)*(diag(2*ones(N-1,1),0)-diag(ones(N-2,1),1)-diag(ones(N-2,1),-1));
    k(N/10) = cond(A);
    disp(sprintf('N = %i: K(A) = %e',N,k(N/10)));
    % ceci pour afficher les valeurs calculees
end

figure()
plot([10:10:120], k)
xlabel('N') % x-axis label
ylabel('K(A)') % y-axis label
saveas(gcf,'ex2_lin','epsc')

figure()
loglog([10:10:120],k);
xlabel('log(N)') % x-axis label
ylabel('log(K(A))') % y-axis label
grid on
saveas(gcf,'ex2_log','epsc')

```

On voit que le graphe bi-logarithmique est bien celui d'une droite, donc du type

$$\log_{10} K = m \log_{10} N + c.$$

On calcule  $m$  et  $c$  directement sur le graphe, en mesurant la pente entre les abscisses 1 ( $N = 10$ ) et 2 ( $N = 100$ ), ou bien en utilisant MATLAB :

```

m = ( log10(k(10)) - log10(k(1)) ) / (log10(120) - log10(10))
m =
    2.0066

c = log10(k(1)) - m*log10(10)
c =
   -0.4060

C = 10^c
C =
    0.3926

```

La pente est presque égale à 2, donc la formule à proposer sera bien

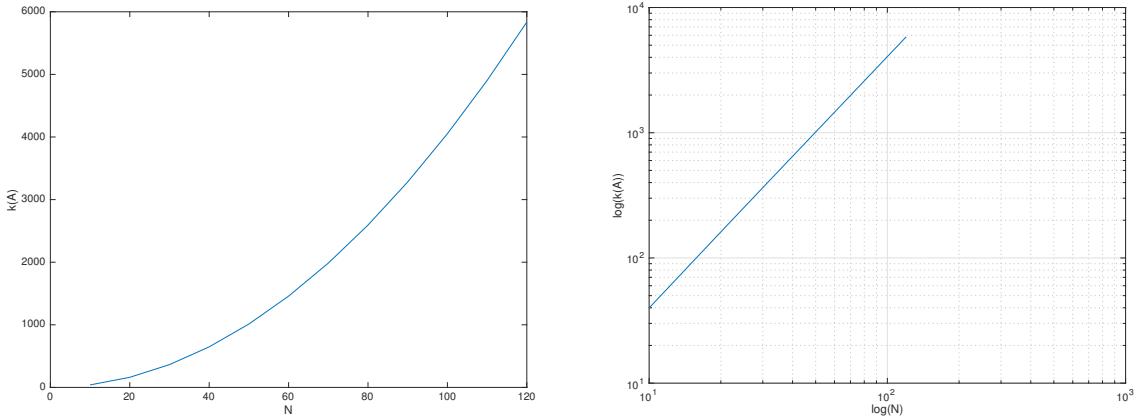


FIGURE 2 – Nombre de conditionnement  $K(A)$  en fonction de  $N$ , graphes linéaire (à gauche) et bi-logarithmique (à droite)

$$K(A) = CN^2$$

avec  $C \simeq 0.3926$ . Donc,  $K(A)$  croît quadratiquement avec  $N$ , ce qui signifie que la solution du système linéaire par la méthode de factorisation  $LU$  devient de plus en plus sensible aux perturbations sur les données et aux erreurs d'arrondi.

### Exercice 3

Etudier l'existence et l'unicité de la factorisation  $LU$  des matrices suivantes :

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 \\ 0 & 2 \end{bmatrix}.$$

### Solution 3

On rappelle que si  $M \in \mathbb{R}^{n \times n}$ , la factorisation  $LU$  de  $M$  avec  $l_{ii} = 1$  pour  $i = 1, \dots, n$  existe et est unique si et seulement si les sous-matrices principales  $M_i$  de  $M$  d'ordre  $i = 1, \dots, n - 1$  sont inversibles (voir Théorème 3.4 à la page 77 du livre). Dans ce cas :

$$M = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} \end{bmatrix}.$$

La matrice singulière  $A$ , dont la sous-matrice principale  $A_1 = 1$  est inversible, admet une unique factorisation  $LU$ . La matrice inversible  $B$  dont la sous-matrice  $B_1$  est singulière n'admet pas de factorisation, tandis que la matrice (singulière)  $C$ , dont la sous-matrice  $C_1$  est singulière, admet une infinité de factorisations de la forme  $C = L_\beta U_\beta$ , avec  $l_{11}^\beta = 1$ ,  $l_{21}^\beta = \beta$ ,  $l_{22}^\beta = 1$ , et  $u_{11}^\beta = 0$ ,  $u_{12}^\beta = 1$ ,  $u_{22}^\beta = 2 - \beta$ ,  $\forall \beta \in \mathbb{R}$ .

### Exercice 4

Soit  $A$  la matrice donnée par

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 5 & 1 & 10 \\ 3 & 1 & 35 & 5 \\ 4 & 10 & 5 & 45 \end{bmatrix},$$

avec  $\det(A) > 0$ . Effectuer la factorisation de Cholesky de la matrice  $A$ , après avoir remarqué qu'une telle factorisation existe.

#### Solution 4

La matrice est symétrique et définie positive. En effet, la matrice est symétrique et tous les mineurs principaux dominants de  $A$  sont positifs (critère de Sylvester). On a vu au cours que les coefficients  $h_{ij}$  de  $H^\top$  (triangulaire inférieure), avec  $A = H^\top H$ , peuvent être calculés comme suit :  $h_{11} = \sqrt{a_{11}}$  et, pour  $i = 2, \dots, n$ , on a

$$\begin{aligned} h_{ij} &= \frac{1}{h_{jj}} \left( a_{ij} - \sum_{k=1}^{j-1} h_{ik} h_{jk} \right), \quad j = 1, \dots, i-1; \\ h_{ii} &= \left( a_{ii} - \sum_{k=1}^{i-1} h_{ik}^2 \right)^{1/2}. \end{aligned} \tag{5}$$

On obtient ainsi

$$H^\top = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & -5 & 1 & 0 \\ 4 & 2 & 3 & 4 \end{bmatrix}.$$

# Exercices Théoriques - Analyse Numérique - 2017

## Section MA

**Prof. A. Quarteroni**

### Séance 4 - Systèmes linéaires : méthodes directes et itératives

#### **Exercice 1**

Etudier l'existence et l'unicité de la factorisation  $LU$  de Gauss des matrices suivantes :

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 \\ 0 & 2 \end{bmatrix}.$$

Répéter l'exercice dans le cas où il est possible d'utiliser une matrice de pivots.

#### **Solution 1**

On rappelle que si  $M \in \mathbb{R}^{n \times n}$ , la factorisation  $LU$  de  $M$  avec  $l_{ii} = 1$  pour  $i = 1, \dots, n$  existe et est unique si et seulement si les sous-matrices principales  $M_i$  de  $M$  d'ordre  $i = 1, \dots, n-1$  sont inversibles (voir Théorème 3.4 à la page 77 du livre). Dans ce cas :

$$M = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ l_{21}u_{11} & l_{21}u_{12} + u_{22} \end{bmatrix}.$$

La matrice singulière  $A$ , dont la sous-matrice principale  $A_1 = 1$  est inversible, admet une unique factorisation  $LU$ . La matrice inversible  $B$  dont la sous-matrice  $B_1$  est singulière n'admet pas de factorisation, tandis que la matrice (singulière)  $C$ , dont la sous-matrice  $C_1$  est singulière, admet une infinité de factorisations de la forme  $C = L_\beta U_\beta$ , avec  $l_{11}^\beta = 1$ ,  $l_{21}^\beta = \beta$ ,  $l_{22}^\beta = 1$ , et  $u_{11}^\beta = 0$ ,  $u_{12}^\beta = 1$ ,  $u_{22}^\beta = 2 - \beta$ ,  $\forall \beta \in \mathbb{R}$ .

On se donne la possibilité d'utiliser une matrice de permutation pour les matrices  $B$  et  $C$  (la matrice  $A$  n'en a pas besoin). Pour  $B$ , si on utilise une matrice de permutation

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

on a que  $PB = I$ . Par conséquent, la factorisation de  $PB$  est triviale.

Pour  $C$ , il n'existe pas de matrice  $P$  telle que le mineur principal d'ordre 1 de  $PC$  est non nul. En revanche, il existe une matrice

$$Q = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

qui permute les colonnes telle que

$$CQ = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix}.$$

Le mineur principal d'ordre 1 de  $CQ$  est non nul, donc il existe une unique factorisation  $LU$  de  $CQ$ .

#### **Exercice 2**

Soit  $A$  la matrice donnée par

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 5 & 1 & 10 \\ 3 & 1 & 35 & 5 \\ 4 & 10 & 5 & 45 \end{bmatrix},$$

avec  $\det(A) > 0$ . Effectuer la factorisation de Cholesky de la matrice  $A$ , après avoir remarqué qu'une telle factorisation existe.

### Solution 2

La matrice est symétrique et définie positive. En effet, la matrice est symétrique et tous les mineurs principaux dominants de  $A$  sont positifs (critère de Sylvester). On a vu au cours que les coefficients  $h_{ij}$  de  $H^\top$  (triangulaire inférieure), avec  $A = H^\top H$ , peuvent être calculés comme suit :  $h_{11} = \sqrt{a_{11}}$  et, pour  $i = 2, \dots, n$ , on a

$$\begin{aligned} h_{ij} &= \frac{1}{h_{jj}} \left( a_{ij} - \sum_{k=1}^{j-1} h_{ik} h_{jk} \right), \quad j = 1, \dots, i-1; \\ h_{ii} &= \left( a_{ii} - \sum_{k=1}^{i-1} h_{ik}^2 \right)^{1/2}. \end{aligned} \tag{1}$$

On obtient ainsi

$$H^\top = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & -5 & 1 & 0 \\ 4 & 2 & 3 & 4 \end{bmatrix}.$$

### Exercice 3

a) En prenant  $n = 1000$ , écrire en MATLAB la matrice  $n \times n$  suivante :

$$A = \begin{bmatrix} 1 & \mathbf{1}^\top \\ \mathbf{1} & -I_{n-1} \end{bmatrix}$$

où  $\mathbf{1}$  est un vecteur colonne unitaire de longueur  $n-1$  et  $I_{n-1}$  est la matrice identité  $(n-1) \times (n-1)$ . Ensuite calculer la factorisation  $LU$  de la matrice  $A$  avec MATLAB (en utilisant la commande `lu` avec  $L$ ,  $U$  et  $P$  comme sorties ; écrire `help lu` pour apprendre comme utiliser la commande).

- b) Visualiser la structure des facteurs  $L$  et  $U$  et de la matrice de permutation  $P$  (en utilisant la commande `spy`).
- c) Soient  $\tilde{P}$  et  $\tilde{Q}$  deux matrices de permutation. On considère maintenant la matrice  $\tilde{A} = \tilde{P} A \tilde{Q}$  obtenue en effectuant une permutation des lignes avec  $\tilde{P}$  et une permutation des colonnes avec  $\tilde{Q}$  et on note par  $\tilde{A} = \tilde{L} \tilde{U}$  la factorisation  $LU$  de la matrice  $\tilde{A}$ . Trouver une permutation des lignes de  $\tilde{P}$  et des colonnes de  $\tilde{Q}$  telles que les facteurs  $\tilde{L}$  et  $\tilde{U}$  soient creux.

- d) Calculer une factorisation  $\tilde{A} = \tilde{L}\tilde{U}$  de la matrice  $\tilde{A}$  à l'aide de la commande `lu` et visualiser la structure des facteurs  $\tilde{L}$  et  $\tilde{U}$ .
- e) Transformer au format creux (en utilisant la commande `sparse`) les matrices  $L$ ,  $U$  du point (a), et les matrices  $\tilde{L}$ ,  $\tilde{U}$  du point (d). En utilisant la commande `whos` comparer la taille en mémoire de ces matrices. Que peut-on remarquer ?

### Solution 3

- a) Pour construire la matrice

$$A = \begin{bmatrix} 1 & \mathbf{1}^\top \\ \mathbf{1} & -I_{n-1} \end{bmatrix}$$

on utilise en MATLAB les commandes suivantes

```
n = 1000;
a11 = 1;
a12 = ones(1, n-1);
a21 = ones(n-1, 1);
a22 = -eye(n-1,n-1);
A = [a11,a12;a21,a22];
```

Ensuite, pour calculer la factorisation  $LU$ , on utilise la commande :

```
[L,U,P] = lu(A);
```

- b) Pour visualiser la structure des facteurs  $L$  et  $U$ , on écrit

```
figure(1)
spy(L)
figure(2)
spy(U)
```

On voit sur la Figure 1 que les facteurs sont pleins :

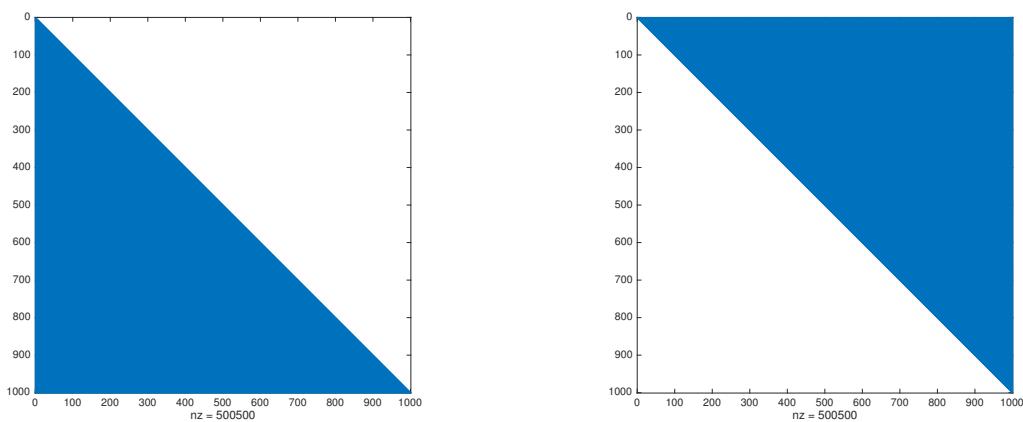


FIGURE 1 – Structure des matrices  $L$  (à gauche) et  $U$  (à droite) telles que  $A = LU$ .

- c) On peut permute la première ligne de  $A$  avec la dernière, ainsi que la première colonne de  $A$  avec la dernière, en utilisant la matrice de permutation  $P$  donnée par

$$P = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix},$$

et en exécutant le produit  $\tilde{A} = PAP$  :

```
P = zeros(n,n);
P(1, n) = 1;
P(n, 1) = 1;
P(2:n-1, 2:n-1) = eye(n-2,n-2);
tildeA = P*A*P;
```

- d) La factorisation  $LU$  de  $\tilde{A}$ , telle que  $\tilde{A} = \tilde{L}\tilde{U}$ , ainsi que les structures des matrices  $\tilde{L}$  et  $\tilde{U}$ , sont données par :

```
[tildeL, tildeU] = lu(tildeA);
figure(3)
spy(tildeL)
figure(4)
spy(tildeU)
```

On trouve les structures présentées dans la Figure 2.

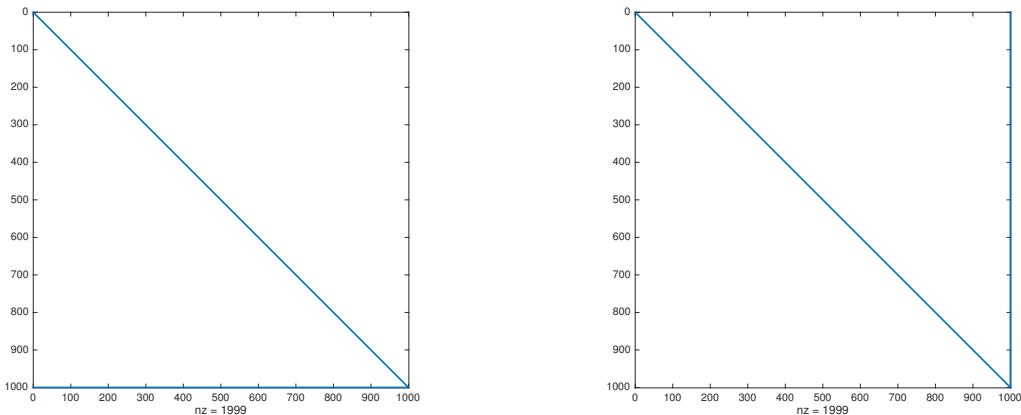


FIGURE 2 – Structure des matrices  $\tilde{L}$  (à gauche) et  $\tilde{U}$  (à droite) telles que  $\tilde{A} = \tilde{L}\tilde{U}$ .

- e) On peut noter que le nombre d'éléments non nuls passe de 500500 à 1999, donc il est 250 fois plus petit ! Cette différence devient importante surtout quand on utilise un format de représentation des matrices creuses qui ne garde que les entrées non-nulles. Cela est fait en MATLAB à l'aide de la commande **sparse**. Normalement, si on travaille avec des matrices creuses et qu'on les converties au format creux, on gagne en mémoire et en performance. Pour cet exemple, on peut voir la différence en tapant

```
sparseU = sparse(U);
```

```

sparseL = sparse(L);
sparse_tildeU = sparse(tildeU);
sparse_tildeL = sparse(tildeL);
whos

```

On trouve

Name	Size	Bytes	Class	Attributes
A	1000x1000	8000000	double	
L	1000x1000	8000000	double	
P	1000x1000	8000000	double	
U	1000x1000	8000000	double	
a11	1x1	8	double	
a12	1x999	7992	double	
a21	999x1	7992	double	
a22	999x999	7984008	double	
n	1x1	8	double	
sparseL	1000x1000	8016008	double	sparse
sparseU	1000x1000	8016008	double	sparse
sparse_tildeL	1000x1000	39992	double	sparse
sparse_tildeU	1000x1000	39992	double	sparse
tildeA	1000x1000	8000000	double	
tildeL	1000x1000	8000000	double	
tildeU	1000x1000	8000000	double	

On voit que l'utilisation de la mémoire pour des matrices pleines est de 8 MB (en fait, comme les entrées sont des **double**, donc 8 Bytes,  $8 * 1.000 * 1.000 = 8.000.000$  Bytes = 8 MB). Pour la première factorisation ( $LU$ ), l'utilisation est même plus grande avec le format **sparse**, tandis que si on utilise la deuxième factorisation ( $\tilde{L}\tilde{U}$ ), chaque facteur occupe moins de 40 KB, c'est-à-dire plus de 200 fois moins que dans le premier cas.

Les figures de cet exercice sont obtenues avec le script **ex3.m**.

```

n = 1000;
a11 = 1;
a12 = ones(1, n-1);
a21 = ones(n-1, 1);
a22 = -eye(n-1,n-1);
A = [a11,a12;a21,a22];

[L,U,P] = lu(A);

figure(1)
spy(L)
saveas(gcf, 'L', 'epsc')
figure(2)
spy(U)
saveas(gcf, 'U', 'epsc')

P = zeros(n,n);
P(1, n) = 1;
P(n, 1) = 1;
P(2:n-1, 2:n-1) = eye(n-2,n-2);
tildeA = P*A*P;

[tildeL, tildeU] = lu(tildeA);
figure(3)

```

```

spy(tildeL)
saveas(gcf, 'tildeL', 'epsc')
figure(4)
spy(tildeU)
saveas(gcf, 'tildeU', 'epsc')

sparseU = sparse(U);
sparseL = sparse(L);
sparse_tildeU = sparse(tildeU);
sparse_tildeL = sparse(tildeL);
whos

```

#### Exercice 4

En considérant la matrice de Hilbert  $A \in \mathbb{R}^{n \times n}$

$$A = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 & \dots \\ 1/2 & 1/3 & 1/4 & \dots & \\ 1/3 & \vdots & & & \\ \vdots & & & & \end{bmatrix} \quad A_{ij} = \frac{1}{i+j-1};$$

- a) Calculer avec MATLAB les éléments de la matrice  $A$  avec  $n = 10$  (en utilisant la commande `hilb`) et le nombre de conditionnement  $K_2(A)$  avec la commande `cond`. Enregistrer les nombres de conditionnement calculés pour  $n = 1, \dots, 10$  dans un vecteur avec 10 composantes.
- b) Visualiser sur un graphe en échelle semilogarithmique (linéaire sur les abscisses et logarithmique sur les ordonnées) la valeur de  $K_2(A)$  en fonction de  $n$ , pour  $1 \leq n \leq 10$  en utilisant la commande `semilogy`. En déduire que  $K_2(A)$  se comporte comme  $e^{\alpha n}$ .
- c) Construire un vecteur colonne  $\mathbf{x}_{ex}$  aléatoire avec  $n$  composantes en utilisant la commande `rand` et calculer le vecteur  $\mathbf{b} = A\mathbf{x}_{ex}$ . Pour chaque  $n = 1, 2, \dots, 10$  résoudre le système linéaire  $A\mathbf{x} = \mathbf{b}$  avec la commande `\` qui met en oeuvre une méthode directe, et calculer l'erreur relative

$$\varepsilon^r = \frac{\|\mathbf{x} - \mathbf{x}_{ex}\|}{\|\mathbf{x}_{ex}\|}. \quad (2)$$

Visualiser l'erreur sur un graphe en échelle semilogarithmique et en déduire que cette erreur se conduit comme  $K_2(A)$ . Visualiser ensemble l'erreur relative  $\varepsilon^r$  et son estimation  $\tilde{\varepsilon}^r$  en échelle semilogarithmique, où  $\mathbf{r}$  est le résidu et  $\tilde{\varepsilon}^r$  est défini comme

$$\tilde{\varepsilon}^r = K_2(A) \frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}. \quad (3)$$

#### Solution 4

- a) D'abord on apprend que

```

help hilb
HILB    Hilbert matrix.
HILB(N) is the N by N matrix with elements 1/(i+j-1),
which is a famous example of a badly conditioned matrix.

```

On peut calculer les quantités nécessaires avec les commandes suivantes, dans une boucle `for` pour  $n = 1, 2, \dots, 10$  :

```
for n=1:10
    A = hilb(n);
    KA(n) = cond(A);
end
```

- b) Le conditionnement de la matrice  $A$  est enregistré dans le vecteur `KA`, qui vaut

```
KA =
1.0e+13 *
Columns 1 through 3
    0.00000000000100    0.000000000001928    0.0000000000052406
Columns 4 through 6
    0.000000001551374    0.000000047660725    0.000001495105864
Columns 7 through 9
    0.000047536735690    0.001525757558921    0.049315373368211
Column 10
    1.602517099202631
```

et son comportement en fonction de  $n$  peut être visualisé avec les commandes suivantes :

```
figure(1);
semilogy([1:1:10],KA,'r');
grid;
xlabel('n');
ylabel('K(A)');
```

On peut observer à la Figure 3 que le conditionnement a une comportement linéaire sur le graphique, c'est-à-dire  $K_2(A)$  croît (presque) exponentiellement en fonction de  $n$ .

- c) On veut construire un vecteur colonne  $\mathbf{x}_{ex}$  aléatoire avec  $n$  composantes, calculer le vecteur  $\mathbf{b} = A\mathbf{x}_{ex}$ , résoudre (pour chaque  $n = 1, 2, \dots, 10$ ) le système linéaire  $A\mathbf{x} = \mathbf{b}$  et calculer l'erreur relative

$$\varepsilon^r = \frac{\|\mathbf{x} - \mathbf{x}_{ex}\|}{\|\mathbf{x}_{ex}\|};$$

on peut utiliser le code suivant :

```
for n=1:10
    A = hilb(n);
    KA(n) = cond(A);
    xex = rand(n,1);
    b = A * xex;
```

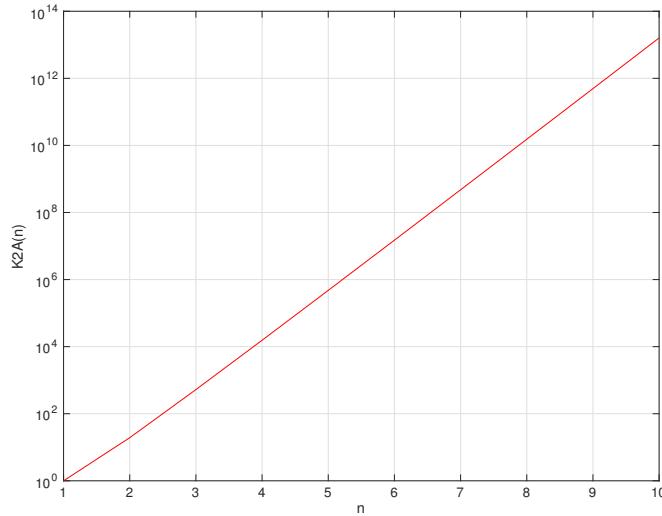


FIGURE 3 – Graphique du conditionnement de la matrice de Hilbert en fonction de  $n$ ; l'échelle est linéaire sur les abscisses et logarithmique sur les ordonnées.

```

x = A\b;
err(n) = norm(x-xex)/norm(xex);
end

```

Ensuite, pour visualiser le comportement du conditionnement et de l'erreur relative  $\varepsilon^r$  sur le même graphique, on peut taper :

```

figure(2);
semilogy([1:1:10], KA, 'r', [1:1:10], err, 'b');
grid;
xlabel('n');
ylabel('KA(n) & err(n)');

```

Le graphique à la Figure 4 montre le conditionnement (ligne rouge) et l'erreur (ligne bleue) en fonction de  $n$  dans une échelle semi-logarithmique. On peut observer que l'erreur relative se comporte de la même façon que le conditionnement, même si elle est plus petite. Cela confirme le fait que, plus le conditionnement de la matrice est grand, plus les erreurs d'arrondi sont amplifiées dans la résolution du système linéaire, conduisant à une erreur relative toujours plus grande, en accord avec l'inégalité suivante :

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{K(A)}{1 - K(A)\|\delta A\|/\|A\|} \left( \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\delta A\|}{\|A\|} \right).$$

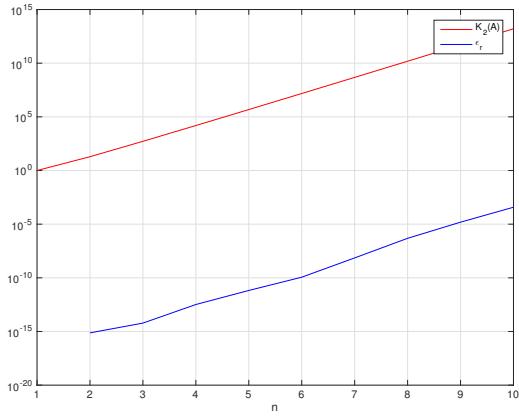
Les figures de cet exercice sont obtenues avec le script `ex4.m`.

```

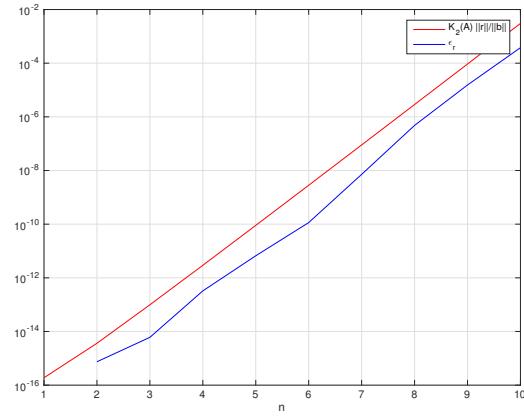
clc
clear all
close all

for n=1:10
    A = hilb(n);
    KA(n) = cond(A);
end

```



(a)  $K$  et  $\varepsilon^r$  en fonction de  $n$ .



(b)  $\tilde{\varepsilon}^r$  et  $\varepsilon^r$  en fonction de  $n$ .

FIGURE 4 – A gauche, Figure 4(a) : Graphique du conditionnement de la matrice de Hilbert,  $K$ , et de l'erreur relative,  $\varepsilon^r$ , définie à l'équation (2), en fonction de  $n$ .

A droite, Figure 4(b) : Graphique de l'estimation de l'erreur relative,  $\tilde{\varepsilon}^r$ , définie à l'équation (3), et de l'erreur relative,  $\varepsilon^r$ , en fonction de  $n$ .

L'échelle est linéaire sur les abscisses et logarithmique sur les ordonnées.

```

figure();
semilogy([1:1:10],KA,'r');
grid;
xlabel('n');
ylabel('K2A(n)');
saveas(gcf,'KA','epsc')

for n=1:10
    A = hilb(n);
    KA(n) = cond(A);
    xex = rand(n,1);
    b = A * xex;
    x = A\b;
    err(n) = norm(x-xex)/norm(xex);
    r(n) = norm(A*x - b);
end

figure();
semilogy([1:1:10], KA, 'r', [1:1:10], err, 'b');
grid;
xlabel('n');
%ylabel('K2A(n) & err(n)');
legend('K_{2}(A)', '\varepsilon_{r}')
saveas(gcf,'KVSErr','epsc')

figure();
semilogy([1:1:10], KA*norm(r)/norm(b), 'r', [1:1:10], err, 'b');
grid;
xlabel('n');
%ylabel('KA*norm(r)/norm(b) & err(n)');
legend('K_{2}(A) ||r|| / ||b||', '\varepsilon_{r}')
saveas(gcf,'ErrVSTildeErr','epsc')

```