MA261. Introduction au calcul scientifique
Corrigé 1 : Introduction à Matlab.

Exercice 1 Soient les vecteurs colonnes et la matrice suivants

$$\vec{u}_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix},\ \vec{u}_2 = \begin{pmatrix} -5 \\ 2 \\ 1 \end{pmatrix},\ \vec{u}_3 = \begin{pmatrix} -1 \\ -3 \\ 7 \end{pmatrix},\ A = \begin{pmatrix} 2 & 3 & 4 \\ 7 & 6 & 5 \\ 2 & 8 & 7 \end{pmatrix}.$$

1. **Structures Matlab**
   (a) Entrer ces données sous Matlab.
   (b) Calculer $\vec{u}_1 + 3\vec{u}_2 - \vec{u}_3/5$.
   (c) Calculer le produit scalaire entre les vecteurs $\vec{u}_1$ et $\vec{u}_2$.
   (d) Calculer le produit $A\vec{u}_1$.
2. **Commandes Matlab**
   Trouver les commandes Matlab permettant de :
   (a) calculer $\|\vec{u}_1\|_2$, $\|\vec{u}_2\|_1$, $\|\vec{u}_3\|_\infty$;
   (b) déterminer les dimensions de la matrice A, en extraire le nombre de colonnes ;
   (c) calculer le déterminant et l'inverse de A.
3. **Résolution de systèmes linéaires**
   Proposer deux méthodes permettant de résoudre le problème $A\vec{x} = \vec{u}_1$, et déterminer les commandes Matlab associées.

Corrigé 1 Taper en ligne :

```
u1 = [ 1 ; 2 ; 3 ]
u2 = [ -5 ; 2 ; 1 ]
u3 = [ -1 ; -3 ; 7 ]
A = [ 2 3 4 ; 7 6 5 ; 2 8 7 ]
u1+3*u2-u3/5
u1'*u2
A*u1
norm(u1,2)
norm(u2,1)
norm(u3,inf)
size(A)
size(A,2)
det(A)
inv(A)
x = inv(A)*u1
x = A\u1
```

Exercice 2 Soient la matrice et les vecteurs colonnes suivants

$$A = \begin{pmatrix} 5/8 & -1/4 & 1/8 \\ 1/4 & 0 & 1/4 \\ 1/8 & -1/4 & 5/8 \end{pmatrix}, \vec{b} = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, \vec{u}_1 = \begin{pmatrix} 5 \\ 2 \\ -4 \end{pmatrix}.$$

On définit, pour $n \geq 1$, la suite de vecteurs $\vec{u}_{n+1} = A\vec{u}_n + \vec{b}$.

1. Construire une fonction `suite.m` calculant les premiers termes de la suite $\vec{u}_n$. Cette fonction aura comme arguments d'entrée les données suivantes : la matrice A, le second membre $\vec{b}$, le terme initial $\vec{u}_1$, et le nombre de termes voulus nb_{it}.
2. Représenter graphiquement l'évolution de chacune des composantes.
   Qu'observe-t-on ?
3. Soient
$$\vec{u}_{1b} = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}, A_b = \begin{pmatrix} 5 & 6 & 3 \\ -1 & 5 & -1 \\ 1 & 2 & 0 \end{pmatrix}.$$
   Observe-t-on le même comportement si on remplace $\vec{u}_1$ par $\vec{u}_{1b}$? Que se passe-t-il si on remplace A par A_b (quel que soit le terme initial) ?

Corrigé 2 Dans le fichier `suite.m`, créer la fonction :

```
function u = suite(A,u1,b,nb_it)
u = zeros(3,nb_it+1);
u(:,1) = u1;
for k = 1:nb_it
  u(:,k+1) = A*u(:,k)+b;
end
```

En sortie, le tableau `u` contient les itérés successifs $(\vec{u}_n)_{1\leq n\leq nb_{it}}$.
Puis, taper en ligne :

```
A = [ 5/8 -1/4 1/8 ; 1/4 0 1/4 ; 1/8 -1/4 5/8 ]
b = [ 1 ; -1 ; 1 ]
u1 = [ 5 ; 2 ; -4 ]
u = suite(A,u1,b,30);
```

Pour étudier le comportement des itérés, taper en ligne :

```
hold on
plot(u(1,:),'g')
plot(u(2,:),'r')
plot(u(3,:),'y')
hold off
```

Les itérations convergent visuellement vers $\begin{pmatrix} 10/3 \\ 2/3 \\ 10/3 \end{pmatrix}$.

```
u1b = [ 2 ; 1 ; 0 ];
ub = suite(A,u1b,b,30);
hold on
plot(ub(1,:),'g')
plot(ub(2,:),'r')
plot(ub(3,:),'y')
hold off
```

Les itérations convergent visuellement vers $\begin{pmatrix} 10/3 \\ 2/3 \\ 10/3 \end{pmatrix}$.

```
Ab = [ 5 6 3 ; -1 5 -1 ; 1 2 0 ]
```

Pour les deux termes initiaux $\vec{u}_1$ et $\vec{u}_{1b}$, la suite diverge.
Pour des itérations du type $\vec{u}_{n+1} = A\vec{u}_n + \vec{b}$, il y a convergence si, et seulement si, le rayon spectral de A est strictement inférieur à un. On rappelle que si $(\lambda_i(A))_i$ sont les valeurs propres de A (dans $\mathbb{C}$), alors on a par définition

$$\rho(A) = \max_i |\lambda_i(A)|.$$

Or, on trouve $\rho(A) = 1/2$ et $\rho(A_b) \approx 5.9$.

Exercice 3 Soit $A \in \mathbb{R}^{n\times n}$. On introduit le vecteur p_L^A (resp. p_C^A), appartenant à $\mathbb{R}^n$, des indices des colonnes (resp. des lignes) du premier coefficient non nul de chaque ligne (resp. de chaque colonne). Par convention, si tous les coefficients d'une ligne (resp. d'une colonne) sont nuls, le nombre reporté est $n+1$. Par exemple, pour la matrice

$$A = \begin{pmatrix} 5 & 6 & 3 & 0 \\ 0 & 5 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \end{pmatrix} \in \mathbb{R}^{4\times 4}, \text{on a } p_L^A = \begin{pmatrix} 1 \\ 2 \\ 5 \\ 3 \end{pmatrix} \text{ et } p_C^A = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 4 \end{pmatrix}.$$

Les profils en ligne P_L^A et en colonne P_C^A peuvent alors se définir comme étant :

$$P_L^A = \left\{(i,j) \in \{1,\dots,n\}^2 \ : \ (p_L^A)_i \leq j\right\}.$$

$$P_C^A = \left\{(i,j) \in \{1,\dots,n\}^2 \ : \ (p_C^A)_j \leq i\right\}.$$

Leur utilité première est d'éviter de stocker les composantes nulles en début de chaque ligne ou colonne.

1. Écrire une fonction qui pour toute matrice calcule son profil ligne et colonne, i.e. renvoie les vecteurs p_L^A et p_C^A.
   `Aide : utiliser la fonction Matlab retournant le minimum d'un ensemble.`
2. Comment pourrait-on simplement améliorer les profils en ligne et en colonne ?
3. Que peut-on dire des profils des matrices symétriques et symétriques définies-positives ? Modifier en conséquence votre fonction.

Corrigé 3 1. Dans `Profil.m`, créer la fonction `Profil` :

```
function [pL,pC] = Profil(A)
N = size(A,1);
pL = zeros(N,1);
pC = zeros(N,1);
for I = 1:N
  pL(I) = min(find(A(I,:)));
  pC(I) = min(find(A(:,I)));
end
```

En sortie, les vecteurs colonnes `pL` et `pC` contiennent les valeurs de p_L^A et p_C^A avec la correspondance `pL(I) = 0` $\iff (p_L^A)_i = n+1$ (resp. `pC(I) = 0` $\iff (p_C^A)_i = n+1$.)

2. Il suffit de définir deux vecteurs q_L^A (resp. q_C^A) des indices des colonnes (resp. des lignes) du dernier coefficient non nul de chaque ligne (resp. de chaque colonne), avec un nombre reporté égal à 0 si tous les coefficients sont nuls. Les profils correspondants sont :
$$P_L^A = \left\{(i,j) \in \{1,\dots,n\}^2 \ : \ (p_L^A)_i \leq j \leq (q_L^A)_i\right\}.$$
$$P_C^A = \left\{(i,j) \in \{1,\dots,n\}^2 \ : \ (p_C^A)_j \leq i \leq (q_C^A)_j\right\}.$$

3. Lorsqu'une matrice A est symétrique, on a toujours $p_L^A = p_C^A$: il suffit d'introduire un unique vecteur p^A.
Si de plus, la matrice est définie-positive, on remarque que $A_{i,i} = (A\vec{e}_i, \vec{e}_i) > 0$, et ainsi $(p^A)_i \leq i$ pour tout i.

   Par ailleurs, si A est symétrique définie-positive, on peut utiliser la factorisation de Cholesky pour résoudre les systèmes linéaires du type $A\vec{x} = \vec{b}$.
   Dans un premier temps, on calcule la matrice triangulaire inférieure L telle que $A = LL^\mathsf{T}$. Dans un second temps, on résout successivement $L\vec{y} = \vec{b}$ et $L^\mathsf{T}\vec{x} = \vec{y}$ par un algorithme de descente-remontée.
   Bien sûr, pour une matrice L triangulaire inférieure, on sait que $i, j = 0$ dès que $j > i$. Le *profil optimisé* d'une telle matrice est donc
$$P_{opt}^L = \left\{(i,j) \in \{1,\dots,n\}^2 \ : \ (p^L)_i \leq j \leq i\right\}.$$
   Et, pour conclure, si on définit le *profil optimisé* d'une matrice SDP comme étant :
$$P_{opt}^A = \left\{(i,j) \in \{1,\dots,n\}^2 \ : \ (p^A)_i \leq j \leq i\right\},$$
   son intérêt principal est d'être conservé par la factorisation de Cholesky ! En d'autres termes, on a la propriété : $P_{opt}^L = P_{opt}^A$.

   Dans le fichier `ProfSDP.m`, créér la fonction `ProfSDP` :

```
function p = ProfSPD(A)
% On suppose que A est SPD.
N = size(A,1);
p = zeros(N,1);
for I = 1:N
  p(I) = min(find(A(:,I)));
end
```

En sortie, le vecteur colonne `p` contient les valeurs de p.

Exercice 4 1. Écrire une fonction Matlab `GenereMatrice` générant une matrice d'ordre quelconque avec des éléments aléatoires.

2. Écrire une fonction Matlab `GenereSysteme` construisant des systèmes linéaires aléatoires, et les résolvant (lorsque c'est possible).

Corrigé 4 1. Dans le fichier `GenereMatrice.m`, créer la fonction :

```
function A = GenereMatrice(m,n,Amax)
A = Amax*rand(m,n);
```

A est une matrice $\mathbb{R}^{m\times n}$, dont les éléments sont compris entre 0 et A_{max}. On peut aussi utiliser `randn` pour avoir des éléments négatifs, ou `sprand` et `sprandn` lorsque m et n sont très grands.

2. Dans le fichier `GenereSysteme.m`, créer la fonction :

```
function [A,b,x,msg] = GenereSysteme(m,Amax,bmax)
A = GenereMatrice(m,m,Amax);
b = bmax*rand(m,1);
if det(A)~=0
  x = A\b;
  msg = 'ok';
else
  x = 0;
  msg = 'pas de solution';
end
```

Mise en œuvre informatique à l'aide de Matlab

1. *Points à retenir* :
   - Lancement de Matlab.
   - Commandes `help` et `lookfor` ; utilisation de la *notice*.
   - Création d'un répertoire pour ranger les fichiers Matlab.
   - Ecriture d'une fonction `xxx` et sauvegarde dans un fichier `xxx.m`.
2. *Fonctionnalités Matlab à maîtriser* :
   - Initialisation (structures *pleines* par défaut) pour vecteurs et matrices : `u = zeros(N,1), A = zeros(N,M)`.
   - Générateur *aléatoire* pour construction d'exemples : `u = rand(N,1), A = rand(N,M)`.
   - Boucles : `for`.
   - Tests : `if, else`.
   - Remise à zéro des variables : `clear all`.
   - Commandes pour l'affichage
     en ligne : `'message'` ;
     sous forme graphique : `plot(u)` (ligne brisée), `plot(u,'o')` (points).
   - Ecriture *symbolique*, sous la forme `u'*v, A*u, A\b`, etc.
   - Commandes pour les normes : `norm(u,1), norm(u,2), norm(u,inf)`.