TP 1 : Introduction à Scilab

Étant donné un entier $n \geq 3$, on considère la matrice $A = (a_{i,j})_{1 \leq i,j \leq n} \in \mathcal{M}_n(\mathbb{R})$, dite *laplacienne*, définie par

$$\forall (1 \leq i, j \leq n) \qquad a_{i,j} = \begin{cases} 2 & \text{si } i = j, \\ -1 & \text{si } |i-j| = 1, \\ 0 & \text{sinon.} \end{cases}$$

1. Définir dans la console de Scilab la matrice A obtenue lorsque $n = 5$.
2. Entrer dans la console les instructions `A'==A`. Quel résultat obtient-on ? Comment peut-on l'interpréter ?
3. À l'aide de la commande `spec`, vérifier que cette matrice est définie positive.
4. Créer une fonction `A = f1(n)`, dont l'argument d'entrée est l'entier n et l'argument de sortie est la matrice A. Pour construire la matrice, cette fonction utilisera uniquement des boucles `for`.
5. Afficher le déterminant de A pour quelques valeurs de n (en utilisant la fonction précédente `f1`). Que remarque-t-on ? Proposer une formule pour le déterminant de A, pour tout n.
6. Créer une autre fonction `A = laplacien(n)`, qui fournira le même résultat que la fonction `f1`, mais cette fois sans utiliser de boucle `for`. On utilisera pour cela les commandes `diag` et `ones` (pour comprendre leur utilisation on pourra utiliser l'aide intégrée au logiciel via la commande `help`, ou simplement taper `fonction diag dans Scilab` dans un moteur de recherche).
7. *Le coût des boucles for.* Comparer l'efficacité des fonctions `f1` et `laplacien` en termes de temps de calcul pour de grandes valeurs de n ($n = 200, 700, \dots$). Que constatez-vous ? Afin de mesurer précisément ce temps de calcul, vous utiliserez la fonction `timer`, qui permet de mesurer (approximativement) le temps d'execution d'une commande, de la façon suivante :

   ```
   timer(); \\  Sert à ‘‘enclencher le chronomètre’’
   INSTRUCTIONS;
   timer(), \\  Renvoie le temps d'exécution des INSTRUCTIONS.
   Noter la virgule!
   ```

   Ce temps de calcul évolue-t-il en fonction de n de la même façon pour les deux fonctions ?
8. *Inverser une matrice vs résoudre un système linéaire 1.* Etant donné $n \in \mathbb{N}$, on définit le vecteur $b = (1, \cdots, 1)^T \in \mathbb{R}^n$ (on rappelle l'existence de la commande `ones`). On veut calculer avec Scilab la solution du

système linéaire $Ax = b$, pour différentes valeurs de n (par exemple, $n = 10, n = 100, n = 1000$, etc [1]). Pour cela on comparera deux approches, en utilisant :

a) soit la commande `inv(A)` qui inverse la matrice A,

b) soit la commande `A\b` qui résoud le système linéaire particulier $Ax = b$.

Les vecteurs x obtenus par les deux méthodes sont-ils identiques ? On pourra utiliser la commande `norm` pour le vérifier [2]. Comment peut-on interpréter ces résultats ?

9. *Inverser une matrice vs résoudre un système linéaire 2.* Reprendre la question précédente et comparer l'efficacité en termes de temps de calcul des deux méthodes (on prendra de grandes valeurs pour n)avec la commande `timer`. Que constatez-vous ?

   On pourra plotter l'évolution du temps de calcul des méthodes a) et b) en fonction de n :

```
N=[500:50:1500]';
for i=1:length(N),
  A=laplacien(N(i)); b=ones(N(i),1);
  timer(); inv(A); T1(i)=timer(); A \b; T2(i)=timer();
end,
plot2d(N,[T1,T2],style=[color('red'),color('blue')]);
```

1. Attention, pour $n > 5000$ le temps de calcul peut devenir long.
2. Rappelons que $\|x - y\|$ permet de mesurer la distance entre deux vecteurs x et y.

Démarrage rapide avec Scilab

Une matrice $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ se définit avec `A = [1 2 3;4 5 6];` Noter que cette commande n'affiche pas de résultat, car elle se termine par un point-virgule "`;`". Si on veut voir le résultat d'une fonction, ou d'une expression, il faut conclure la commande avec une virgule "`,`".
La définition d'un vecteur se fait de manière similaire, car Scilab considère que les vecteurs de taille n sont des matrices de taille $n \times 1$.

Quelques commandes de base pour la manipulation de matrices :

- Transposée d'une matrice : `A'` (Utile pour passer d'un vecteur ligne à colonne).
- Produit de deux matrices (ou matrice fois vecteur) : `A*B`
- Puissance d'une matrice : `A^3`
- Inverse d'une matrice : `A^-1` ou `inv(A)`
- Solution de l'équation $Ax = b$: `A\b` (Attention : le vecteur doit être en colonne !)
- Déterminant d'une matrice : `det(A)`
- Spectre d'une matrice : `S=spec(A)` retourne un vecteur contenant toutes les valeurs propres de A rangées par ordre croissant.
- Norme euclidienne d'un vecteur : `norm(x)`

Quelques commandes pour générer des matrices :

- Matrice identité de taille $n \times n$: `eye(n,n)`
- Matrice aléatoire de taille $n \times m$: `rand(n,m)`
- Vecteur défini par progression arithmétique : `I=[3:0.1:4]` renverra le vecteur ligne `[3 3.1 3.2 ... 3.9 4]`

Syntaxe de la boucle `for` :

```
for i=1:100,
   INSTRUCTIONS;
end
```

Écrire et utiliser une fonction dans un fichier :

1. Utiliser le navigateur de fichiers de Scilab pour se placer dans un dossier dans lequel on va travailler.
2. Depuis la console Scilab, cliquer sur l'icône "Démarrer Scinotes".
3. Écrire sa fonction dans Scinotes. La syntaxe est la suivante :

   ```
   function [variables-de-sortie] = nom-de-la-fonction(variables-entrantes)
      INSTRUCTIONS;
   endfunction
   ```

   Si besoin est, on commentera une ligne en la faisant commencer par \\
4. Enregistrer le fichier dans le dossier, en lui donnant le nom de la fonction.
5. Cliquer sur l'icône "Exécuter".
6. On peut alors faire appel à la fonction depuis la console ou d'autres fonctions.

Astuce : si Scilab mouline sur un calcul trop difficile, on peut tuer l'exécution depuis la console via Contrôle > Abandonner.