Faculté des sciences et ingénierie (Toulouse III) **Année universitaire**
Département de mathématiques – L3 ESR **2017-2018**
U.E. Analyse Numérique

Feuille de TP1
Introduction à Scilab, Algèbre linéaire

1 Introduction à Scilab

Aide en ligne : Utiliser la commande **help** suivie du nom de la commande pour savoir sa syntaxe ou ce qu'elle renvoie. On peut aussi, avec un mot clef de ce que vous cherchez utiliser **apropos** suivie du mot clef.

Syntaxe de base, commandes élémentaires :
- Le point virgule permet de ne pas afficher les calculs effectués par Scilab.
- Si l'on effectue plusieurs commandes en même temps il faut les séparer d'une virgule.
- Les décimaux sont définis avec un point : x=1.2.
- Les opérations classiques : $+$, $-$, $*$, /, x^2, %**pi**, **log**, **cos**, **sin**, **floor**...
- Tirer un réel de $[0, 1]$ au hasard : **rand**
- valeur absolue = **abs**.

Calcul vectoriel, calcul matriciel : Pour définir un vecteur, il faut mettre les idfférents éléments entre crochets et séparés :
- soit par une virgule ou un espace pour les vecteurs ligne,
- soit par un point virgule pour les vecteurs colonne.

On peut créer un vecteur vide sans mettre de valeur entre les crochets.

```
x=[1 2 3 4 5], y=[1,2,3,4,5], z=[0;2;0;4;8] t=[].
```

Pour définir une matrice c'est la même chose : ligne par ligne, les éléments d'une même ligne séparés par des espaces ou virgules et on change d ligne avec un point virgule.

```
A=[0,1,0,-1; -2,1,3,2; 3,1,0,-1; -1,1/3,0,1]
```

Opérations sur les vecteurs et les matrices :
- Calcul vectoriel et matriciel "classique" avec les opérateurs usuels : $+$, $-$, $*$, ^. A noter que A\b résoud le système linéaire $AX = b$.
- Calcul "élément par élément" : les opérateurs classiques précédés d'un .
  .*, ./, .^
- les fonctions usuelles s'appliquent aux vecteurs et matrices élément par élement : **log**, **exp**, **cos**...
- calcul de la longueur d'un vecteur : **length**, taille d'une matrice : **size**
- extraction de sous-vecteurs, sous-matrices :

```
x(i), A(i,j), x(i:j), x(i:2:j), A(i:j,k:l), A(:,j), A(i,:).
```

désignent respectivement l'élément i de x, l'élément i, j de A, le sous-vecteur de x des éléments i à j, le sous vecteur de x de i à j de 2 en 2, la sous matrice de A formée des lignes i à j et colonnes k à l, la $j$ème colonne de A et la $i$eme ligne de A...

— le plus grand/petit élément d'un vecteur : **max**, **min**, norme euclidienne d'un vecteur : **norm**

— la somme des éléments dun vecteur x : **sum**(x)

Créer des vecteurs/matrices particuliers :

— discrétisation de N termes de pas uniforme de 1er terme a, de dernier terme b : **linspace**(a,b,N)

— discrétisation de pas h de 1er terme a et de dernier terme b (si c'est possible) : a:h:b

— matrice/vecteur de zeros : **zeros**(n,p), matrice/vecteur de 1 : ones(n,p)

— matrice aléatoire de termes entre 0 et 1 de taille n, p : **rand**(n,p)

— concaténation de 2 vecteurs lignes x et y : [x y], deux vecteurs colonnes : [x;y]

Graphiques : Pour tracer le graphe d'une fonction, Scilab a besoin d'un vecteur d'abscisses et d'un vecteur d'ordonnées, et, en gros, il relie les points... comme votre vieille TI du lycée... Mais comme un exemple vaut mieux que de longs discours :

```
x=linspace(0,2*%pi,100);
y=sin(x);
z=cos(x);
plot(x,y)
plot(x,z)
clf
plot(x,y,x,z)
legend('sin','cos')
clf
plot(x,z)
plot(x,y,'r')
clf
plot(x,y,%pi/2,1,'o')
clf
subplot(1,2,1)
plot(x,y)
title('graphe de $\sin$')
subplot(1,2,2)
plot(x,z)
title('graphe de $\cos$')
```

Explications :

— ligne 4 (resp 5) : on trace x en abscisse, y (resp z) en ordonnées (graphe de sin sur $[0, 2\pi]$ avec 100 points donc). Comme on n'a rien dit entre les 2 "plot", ben les graphiques se superposent sur le même avec la même couleur
— ligne 5 : on efface la fenêtre graphique
— ligne 6 : on trace *sur le même graphe mais pas de la même couleur* les graphes de cos et sin
— ligne 7 : on va étiqueter chacun des 2 graphes : le premier avec cos et le 2eme avec sin
— lignes 10-11 : pareil que 4 et 5 sauf que cette fois on a demandé la couleur rouge pour sin.
— ligne 13 : on trace le graphe de sin et on met un rond sur le point de coordonnées $(\pi/2, 1)$.
— lignes 15à 20 : ici on fait une ligne de 2 graphes l'un à coté de l'autre. En effet, **subplot**(i,j,k) crée un tableau de graphes de i lignes, j colonnes. On définit chaque graphique comme d'habitude, case par case (du coup on définit la case numéro k, c'est numéroté de gauche à droite de la ligne 1 à la ligne i). Donc ici on fait une ligne de 2 graphes : celui de gauche est le graphe de sin (on met même un titre avec **title**) et celui de droite est le graphe de cos.

Scripts, Fonctions :Écrire un script ou une fonction permet d'itérer une succession d'instructions pour des valeurs de paramètres ou des données différentes sans avoir à retaper toutes les instructions à chaque fois.
Pour écrire un script ou une fonction, on doit créer un fichier d'extension .sci dans le répertoire courant (Attention, ce doit être celui où Scilab est ouvert). Pour plus de simplicité dans vos programmes, on conseille que le nom du fichier corresponde au nom de la fonction qu'il contient, ou au nom que l'on veut donner au programme que l'on a rédigé (par exemple 'monscrip.sci'). Ainsi, en tapant en ligne de commande

```
exec('monscript.sci')
```

on exécute le script contenu dans le fichier 'monscrip.sci'. S'il s'agit d'une fonction, en tapant **monscript** (comme si on utilisait une fonction Scilab prédéfinie) suivie des arguments nécessaires, Scilab est capable de savoir dans quel fichier aller chercher la procédure correspondant au script ou à la fonction qu'on lui a demandé d'éxecuter.

Un exemple Cet exemple décrit deux manières de calculer la somme et le produit des éléments d'un vecteur $v \in \mathbb{R}^n$: avec un script ou avec une fonction :

```
v= ...
n=length(v);
S=0; P=1;
for i=1:n
  S=S+v(i); P=P*v(i);
end
S
P
```

```
function [S,P]=somprod(v)

n=length(v);
S=0; P=1;
for i=1:n
  S=S+v(i); P=P*v(i);
end
```

Structures de contrôle : boucles, tests : On en rappelle ici que les syntaxes pour les boucles et les tests en Scilab.

1. **Syntaxe boucles "for" :**

```
for i=imin :pas: imax
        <commandes>;
end
```

NB : Si vous écrivez tout sur une ligne, il faut une virgule après imax
Exemple :

```
for i=1:2:10, X(i)=1; end
```

2. **Syntaxe boucles while :**

```
    while <condition>,
        <commandes>;
    end
```

3. **Syntaxe tests :**

```
    if <test>,
        <commandes>;
   elseif <test>,
        <commandes>;
   else <commandes>;
   end
```

2 Mise en pratique : Un peu d'algèbre linéaire vue en TD

Exercice 1. (Petits systèmes, stabilité et conditionnement)
On considère les systèmes linéaires perturbés suivants étudiés en TD :

$$A = \begin{pmatrix} 7 & 10 \\ 5 & 7 \end{pmatrix}, \quad Y_1 = \begin{pmatrix} 10 \\ 7 \end{pmatrix} \quad \text{et} \quad Y_2 = \begin{pmatrix} 10,1 \\ 6,9 \end{pmatrix}.$$

Résoudre les équations $AX = Y_1$ et $AX = Y_2$ grâce à la commande \ de Scilab. Calculer le conditionnement de la matrice A et conclure.

Exercice 2. (Normes subordonnées)

En considérant la matrice suivante $A = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{pmatrix}$, utiliser la commande **norm** de Scilab pour calculer ses normes $\|\cdot\|_1, \|\cdot\|_2, \|\cdot\|_\infty$ et retrouver les formules démontrées en TD en calculant les membres de droite de manière la plus concise possible grâce aux commandes pré-programmées de Scilab.

1. $\|A\|_\infty = \max_{i\in\{1,\dots,N\}} \sum_{j=1}^{N} |a_{i,j}|$
2. $\|A\|_1 = \max_{j\in\{1,\dots,N\}} \sum_{i=1}^{N} |a_{i,j}|$
3. $\|A\|_2 = (\rho(A^T A))^{\frac{1}{2}}$