Université de Franche-Comté
Licence de Mathématiques, Semestre 6, 2019-20

Discrétisation des EDPs
Feuille de TP n^{o}1
Notions de base sous `Scilab`

1 Principes généraux

• Les deux fenêtres `Scilab`

1. Fenêtre `console` : elle sert à communiquer avec `Scilab` : on peut y taper des instructions et elle contiendra les résultats des différentes exécutions.
   Quelques commandes :

`clc`	:	efface tout ce qui est affiché dans la fenêtre
`clear`	:	efface le contenu de toutes les variables
`cd`	:	identique unix/linux/DOS (`cd ../` : remonter d'un niveau ; chemin entre ' ' s'il contient des espaces ou caractères spéciaux)
`pwd`	:	affiche le chemin courant
`ls`	:	affiche les fichiers et répertoires
↑	:	recopie la dernière instruction entrée (modifiable avant d'être validée)
`help NomDeCmde`	:	affiche l'aide concernant la commande `NomDeCmde`

2. Fenêtre `SciNotes` : éditeur de texte de scilab. On l'utilisera pour manipuler les fichiers textes créés lors des TP (extensions `.sce` pour les programmes aussi appelés scripts ou `.sci` pour les fonctions)[1].

• Environnement de travail
- créer un répertoire dans lequel placer les fichiers `Scilab` que vous écrirez lors des différents TP.
- préciser à `SciNotes` que ce répertoire sera votre répertoire de travail
  (Menus "*Fichiers* → *Répertoires de travail* → *Ajouter un répertoire*").
- préciser à `console` que ce répertoire sera le répertoire où `Scilab` cherchera vos fonctions
  (Menus "*Fichiers* → *Changer de répertoire courant*").

• Exécutions de scripts
Pour exécuter une succession de commandes contenues dans le fichier `script.sce`, deux possibilités :
- soit cliquer sur le symbole ▷ dans la barre d'outils de `SciNotes`
- soit taper dans la `console` la commande : `exec('script.sce')`.
Pour exécuter seulement une partie du script, on peut effectuer un copier-coller des instructions concernées dans la fenêtre de commande, ou bien sélectionner les lignes concernées puis faire un clic droit : "*Evaluer la sélection*".

• Mode de travail
Vous avez alors 2 possibilités pour faire exécuter des taches à `Scilab`, la seconde étant celle à laquelle il faudra vous habituer :

1. soit n'utiliser que la fenêtre `console` pour y taper vos instructions et voir les résultats ;
2. soit mettre votre script dans un fichier `TP1.sce` et vos fonctions dans un fichier `MesFonctions.sci`
   (a) Les deux premières lignes du script `TP1.sce` devront être :

```
1. clear  // nettoyage complet de la memoire (optionnel mais conseille)
2. exec('MesFonctions.sci') // mise en memoire des fonctions du fichier
```

   (b) On modifie le fichier `TP1.sce` et si nécessaire `MesFonctions.sci`, on sauvegarde et on exécute ensuite le script (voir Exécutions de scripts ci-dessus). Dans ce cas, l'affichage éventuel des résultats se fait dans la fenêtre `console`.

Contrairement au premier cas, les résultats ne sont pas affichés dans ce deuxième cas lors de l'exécution du script. Il faut le demander explicitement à `Scilab` et pour cela il y a de nouveau 2 façons de faire :

- soit directement dans la `console` si la variable est en mémoire ;
- soit en insérant les commandes `disp` ou `printf` dans le fichier (la première est plus simple mais aussi plus limitée).

[1] Si `SciNotes` n'est pas ouvert, taper `edit` puis Entrée dans la fenêtre `console`

2 Vecteurs et matrices

2.1 Définitions de vecteurs et opérations

Exercice 1 Définir les vecteurs suivants :

$u_1 = (1, 2, 3, 4)$, $v_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, $u_3 = (1 + i, 3 + 2i, 4i)$, $u_4 = (1, 2, \ldots, 20)$, $u_5 = (0, \frac{1}{10}, \frac{2}{10}, \ldots, \frac{20}{10})$.

Exercice 2 Afficher les quantités suivantes <u>à l'aide des vecteurs de l'exercice 1</u> :

$2u_1$, $(1^5, 2^5, 3^5, 4^5)$, $\|u_1\|_2$, $(1, 2, 3, 4, 1 + i, 3 + 2i, 4i)$, $v_1 = \begin{pmatrix} 1+i \\ 3+2i \\ 4i \end{pmatrix}$, $(\frac{2}{10}, \ldots, \frac{19}{10})$, la longueur de u_4.

2.2 Définitions de matrices et opérations

Exercice 3 Construire les matrices suivantes :

1. $A_1 = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$, $A_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$, l'identité 5×5, $A_3 = 0_{2,3}$, A_4 de taille 6×2 d'éléments valant tous 1.
2. A de taille 10×10 telle que $a_{ij} = 25$ si $i \neq j$ et $a_{ii} = 30$ puis B de taille 12×3 telle que $b_{ij} = 5j$.

Exercice 4

1. Afficher la taille de A ; sa première puis sa dernière ligne ; la sous-matrice (A_{ij}), $i = 3, .., 5$, $j = 5, .., 9$.
2. Afficher la matrice bloc $\begin{pmatrix} A_1 & 0 \\ 0 & A_1 \end{pmatrix}$.
3. Calculer A^2 et ${}^tA$. Utiliser `Scilab` pour montrer que A est symétrique puis qu'elle est inversible
4. Pour $b = (1, \ldots, 10)$, calculer la solution X de $AX = {}^t b$, puis celle $XA = {}^t b$

Important : les références des éléments d'un vecteur ou d'une matrice commencent toujours à 1 (par exemple $u(0)$ n'aura pas de sens et provoquera un message d'erreur)

3 Fonctions

Remarque générale
Lorsque le cas se présente, il est vivement recommandé d'appliquer une fonction directement à un vecteur (ou une matrice) plutôt que d'effectuer une boucle (`Scilab` est optimisé pour ce premier cas). Dès qu'on écrira une fonction, il faudra donc systématiquement se demander si l'écriture permet une utilisation pour des entrées vectorielles. Parfois, une adaptation est inutile : le produit $2X$ a un sens si X est un vecteur ; parfois elle est nécessaire: le produit XZ n'a pas de sens si X et Z sont des vecteurs. Le sens à donner à ce produit (pour généraliser le cas scalaire), serait un produit composante par composante : $x_i z_i$. L'opérateur effectuant le produit composante par composante est `.*`. Appliqué à des scalaires, il correspond au produit classique.

Exercice 5

1. Sans utiliser aucune boucle, écrire une fonction F calculant pour $x, y \in \mathbb{R}^n$, la quantité $z = F(x, y) \in \mathbb{R}^n$ où $z_i = 1 + x_i y_i$ (attention au 1 pour $n \geq 2$!).
2. Calculer $F(x, y)$ pour x et y deux colonnes distinctes de B définie à l'exercice 3.

Généralisation d'opérateurs classiques à des opérations composante par composante :

- pour permettre la compatibilité entre vecteurs, le produit `*` doit être remplacé par `.*`, la puissance `^` par `.^` et l'inverse ou le quotient `/` par `./`
- les fonctions `sqrt`, `log`, `exp`, `cos`, `sin`, ... sont directement utilisables avec des entrées vectorielles le résultat étant du même type que l'entrée.

Exercice 6

1. Ecrire une fonction ayant pour entrée l'entier $n \geq 2$ et construisant le vecteur contenant les n premiers termes de la suite de Fibonacci, définie par la formule de récurrence : $a_1 = a_2 = 1$, $a_k = a_{k-1} + a_{k-2}$, pour $k \geq 3$.
2. Ecrire une fonction ayant pour entrée deux entiers n et m et construisant la matrice C de taille $n \times m$ telle que $c_{ij} = \frac{1}{i+j}$. Utiliser `Scilab` pour montrer que si $n = m$, la matrice C n'est pas nécessairement définie positive.

4 Graphiques

1. Commande `plot`
   On représente toujours des vecteurs (y compris pour les fonctions classiques).
   Par exemple :
   ```
   X1 = -%pi:0.1:%pi;
   Y1 = cos(X1);
   plot(X1,Y1)
   ```
   Attention : $X1$ et $Y1$ doivent être de même taille.

2. Options de la commande `plot`
   C'est un troisième argument pouvant cumuler 3 symboles :

   (a) couleur du tracé : `r` rouge, `b` bleu, `g` vert, etc...

   (b) style de la ligne joignant les points : `-` continue, `:` pointillé

   (c) symbole marquant l'emplacement de chaque point : `o`, `x`, `*`, `v`, `<`, `d`

   Exemple : comparer
   ```
   X = 1:10;
   Y = X .^2;
   plot(X,Y)
   plot(X,Y,'d')
   plot(X,Y,':')
   plot(X,Y,'d:')
   plot(X,Y,'rd-.')
   ```

3. Représentation simultanée de plusieurs courbes : `plot(X,Y,'*:',X1,Y1,'o')`

4. Attributs d'un graphique :

   (a) titre : `title`

   (b) légende : `legend`

   (c) étiquettes d'axes : `xlabel`, `ylabel`

   Il est parfois nécessaire de définir un attribut contenant une partie fixe et une partie variable.
   Exemple :
   ```
   disp('Graphe de sin sur [a,b]')
   a=input("Donner la valeur de a :  ") // choix des bornes par l'utilisateur
   b=input("Donner la valeur de b :  ")
   X2=a:(b-a)/100:b;
   Y2=cos(X2);
   plot(X2,Y2);
   titre = msprintf('Fonction sinus sur [%i,%i]',0,10) //  %i pour format entier
   title(titre)
   ```

5. Juxtaposition de plusieurs graphiques : `subplot`
   Exemple :
   ```
   subplot(1,2,1) // 1 ligne avec deux graphiques celui etant le premier
   plot(X,Y)
   subplot(1,2,2) // et celui-ci le second
   plot(X1,Y1)
   ```

Exercice 7

1. Représenter graphiquement les $n \geq 2$ premiers termes de la suite de Fibonacci (exercice 6) où l'entier n sera choisi par l'utilisateur.

2. Ajouter des attributs au graphique dont un titre mentionnant n.

3. Utilisez alors une échelle plus adaptée à cette représentation.

Exercice 8

1. Représenter graphiquement le cercle de centre $(0, 0)$ et de rayon 2.

2. Ajouter sa tangente au point de coordonnées $(\sqrt{2}, \sqrt{2})$.

3. Taper alors les commandes : `a=gca();` puis `a.isoview="on";` et ajouter des attributs.