TP 1-Révisions

Avant de commencer

0.1 Enregistrez votre travail

Lorsque vous fermez Scilab, vous perdez instantanément tout travail non enregistré.

1. Dans le dossier « Mes documents » créez un dossier « Info_2a » puis un sous dossier « TP_1 ».
2. Durant la séance sauvegardez régulièrement vos fichiers dans le dossier « TP_1 ».

A chaque nouvelle feuille de TP, il faudra créer un dossier correspondant.

0.2 SciNotes

On utilisera systématiquement **SciNotes**, l'éditeur de texte intégré à Scilab. Pour y accéder, une fois Scilab ouvert :

1. placez-vous sur la console,
2. cliquez sur l'icône  .
3. dans fichier, sélectionnez « Enregistrez-sous » pour sauvegarder ce fichier dans le dossier du TP1 sous le format *.sci* : ce fichier contiendra toutes les fonctions du TP1,
4. dans fichier, sélectionnez « Nouveau » pour créer un nouveau fichier et enregistrez-le dans le dossier du TP1 sous le format *.sce* : ce fichier contiendra tous les scripts du TP1.

Pour exécuter un script ou une fonction, dans SciNotes vous pouvez, au choix sélectionner « Exécuter fichier sans écho » ou cliquer sur l'icône  qui enregistre et exécute.

0.3 Commentaires

Beaucoup de TP dureront plusieurs séances; il est donc important, en plus de sauvegarder votre travail, que votre code soit « propre » :

- donnez des noms explicites à vos fonctions et vos variables;
- commentez votre code : la commande // permet d'apposer des commentaires (les expressions qui suivent // sont ignorées par Scilab).

Exemple 1

```
// Exemple 1

function y = exemple1(x)    //fonction de l'exemple 1
    y  = 2x
endfunction
```

0.4 Aide

En cas de doute, vous pouvez consulter l'aide de Scilab grâce à la commande (dans la console) :

```
help fonction
```

où `fonction` est le nom de la commande dont on cherche les fonctionnalités.

1 Vecteurs, matrices

Définir une matrice

- Un **vecteur ligne** est défini par une liste d'éléments entre crochets séparés par des virgules.
- Un **vecteur colonne** est défini par une liste d'éléments entre crochets séparés par des points-virgules.
- Pour définir une matrice de taille $n \times p$ on combine les deux syntaxes.

Exercice 1

1. *Dans la console, taper*
$$\mathrm{A1} = [1,2,3;4,5,6].$$
Expliquer ce que cette ligne de commande réalise.
2. *Créer la matrice* $\mathrm{B2} = \begin{pmatrix} 1 & -1 & 0 \\ 3 & 2 & -1 \\ 1 & 1 & 1 \end{pmatrix}$.
3. *Avec l'aide Scilab, déterminer le rôle des commandes* `zeros`, `ones`, `eye`, `linspace`.

On peut aussi construire des matrices « par blocs » en concaténant (c-à-d juxtaposant) des vecteurs et/ou matrices, si leur taille le permet, verticalement ou horizontalement.

Par exemple, si A et B sont des matrices avec le même nombre de colonnes, `[A;B]` est la concaténation verticale de A et B (la matrice $\begin{pmatrix} \mathrm{A} \\ \mathrm{B} \end{pmatrix}$).

Exercice 2 (*Définition par blocs*)

On considère les matrices suivantes :

$$\mathrm{A2} = \begin{pmatrix} 1 & 3 & 5 & 7 \end{pmatrix} \quad , \quad \mathrm{B2} = \begin{pmatrix} 4 & 2 & -1 \end{pmatrix} \quad , \quad \mathrm{C2} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad \mathrm{D2} = \begin{pmatrix} 0 \\ -2 \\ 0 \\ 0 \end{pmatrix}.$$

1. *Définir les matrices* A2, B2 *et* D2.
2. *Construire la matrice* C2 *en concaténant les colonnes* $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, $\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ *et* $\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$.
3. *On considère les matrices*
$$\mathrm{M2} = \begin{pmatrix} 1 & 3 & 5 & 7 \\ 4 & 2 & -1 & 0 \\ 1 & 0 & 1 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad et \quad \mathrm{N2} = \begin{pmatrix} 0 & -2 & 0 & 0 \\ 1 & 0 & 1 & 4 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix}.$$
Construire les matrices M2 *et* N2 *à l'aide des matrices* A2, B2, C2 *et* D2 *et, éventuellement, de leurs transposées (voir encadré ci-dessous).*

Exercice 3 (*EDHEC 2015*)

On considère la matrice suivante

$$\mathrm{C3} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Recopier et compléter, à l'aide des matrices `zeros` *et* `ones`*, les deux espaces libres pour que la commande construise la matrice* C3.

```
C3=[ones(1,5);---------,---------;ones(1,5)]
```

Manipuler des matrices

Soient $A, B \in \mathcal{M}_{n,p}(\mathbb{R})$.

- Extraction et modification des éléments.
  1. L'instruction $A(i, j)$ extrait le coefficient de la $i^{\text{ème}}$ ligne et de la $j^{\text{ème}}$ colonne de A.
     L'instruction $A(i, j) = a$ remplace le coefficient de la $i^{\text{ème}}$ ligne et de la $j^{\text{ème}}$ colonne de A par a.
  2. L'instruction $A(i, :)$ extrait la $i^{\text{ème}}$ ligne de A.
     L'instruction $A(i, :) = L$ remplace la $i^{\text{ème}}$ ligne de A par L.
  3. L'instruction $A(:, j)$ extrait la $j^{\text{ème}}$ colonne de A.
     L'instruction $A(:, j) = C$ remplace la $j^{\text{ème}}$ colonne de A par C.
  4. L'instruction `size(A)` renvoie la taille de la matrice et `rank(A)` son rang (voir le chapitre 5).
- Opérations sur les matrices.

Syntaxe	`A+B`	`A-B`	`A*B`	`inv(A)`	`A'`	`A^k`
Matrice renvoyée	$A+B$	$A-B$	AB	A^{-1}	${}^tA$	A^k

Exercice 4 (*EDHEC 2017*)

On considère la matrice :

$$A4 = \begin{pmatrix} 1 & \frac{2}{3} & \frac{1}{3} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Compléter et recopier, les commandes suivantes pour que soit affichée la matrice A^n pour une valeur n entrée par l'utilisateur :

```
n=input('Entrez une valeur pour n:')
A4=[-------]
disp(-------)
```

2 Boucles `while` et `for`

2.1 Boucle `for`

Boucle `for`

La syntaxe est la suivante :

```
for var=vect
    instruction
end
```

où

- `vect` est un vecteur de valeurs,
- `var` est une variable qui prend successivement les valeurs contenues dans `vect`,
- `instruction` est un bloc d'instructions qui va être exécuté successivement pour chaque valeurs de `var`.

En général, `var` est de la forme `deb:pas:fin` c'est-à-dire le vecteur prenant les valeurs entre `deb` et `fin` espacées de `pas`.

Exercice 5 (*Ecricome 2015*)

Soit F la fonction définie sur $\mathbb{R}$ par

$$\forall x \in \mathbb{R},\ F(x) = \begin{cases} 0 & si\ x < 0 \\ 1 - e^{-x} & si\ x \geqslant 0 \end{cases}$$

On considère la suite $(u_n)_{n\geqslant 1}$ définie par $u_1 = 1$ et pour tout $n \geqslant 1$, $u_{n+1} = F(u_n)$.

1. *Monter que $\forall n \geqslant 1, u_n > 0$.*

2. *Recopier et compléter le programme suivant qui permet de représenter les cent premiers termes de la suite.*

```
U=zeros(1,100);
U(1)=1
for n=1:99
    U(n+1)=-----------
end
plot(U,"+")
```

3. *A l'aide de la représentation graphique obtenue, que peut-on conjecturer quant à la monotonie et la limite de la suite?*

Les fonctions

La syntaxe d'une fonction est la suivante :

```
function sortie=maFonction(entree)
    corps
endfunction
```

où

- `sortie` : le ou les éléments renvoyés par la fonction (zéro, un ou plusieurs variables),
- `maFonction` : le nom de la fonction,
- `entree` : le ou les arguments d'entrée de la fonction (zéro, un ou plusieurs variables),
- `corps` : la suite finie d'instructions qui permet à la fonction de calculer `sortie` à partir de `entree`

Exercice 6 (*Ecricome 2018*)

On considère les matrices $A = \begin{pmatrix} 2 & 1 & -2 \\ 0 & 3 & 0 \\ 1 & -1 & 5 \end{pmatrix}$ *et* $B = \begin{pmatrix} 1 & -1 & -1 \\ -3 & 3 & -3 \\ -1 & 1 & 1 \end{pmatrix}$. *On pose* $X_0 = \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}$, $X_1 = \begin{pmatrix} 3 \\ 0 \\ -2 \end{pmatrix}$ *et pour tout entier naturel* n, $X_{n+2} = \frac{1}{6}AX_{n+1} + \frac{1}{6}BX_n$.

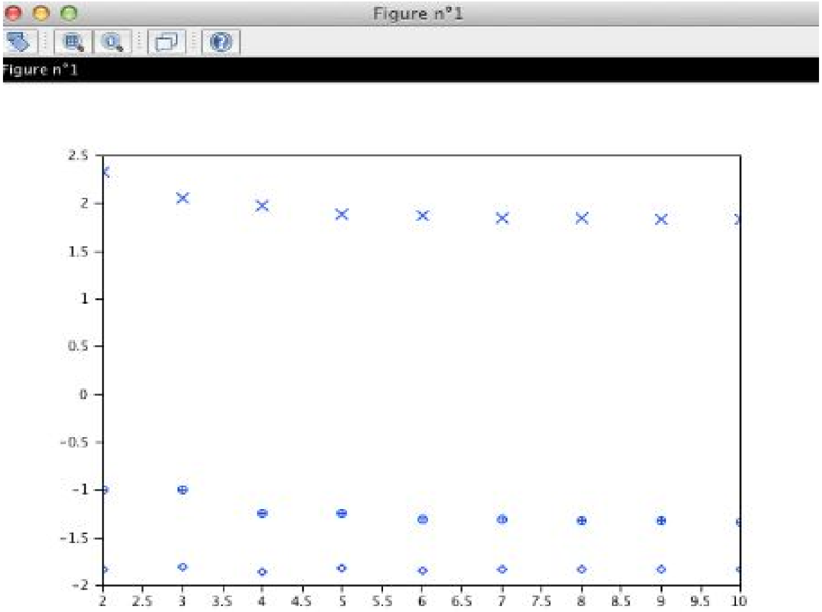
1. *Recopier et compléter (dans le fichier .sci) la fonction ci-dessous qui prend en argument un entier supérieur ou égal à 2 et qui renvoie* X_n :

```
function res=X(n)
    Xold=[3;0;-1]
    Xnew=[3;0;-2]
    A=[2,1,-2;0,3,0;1,-1,5]
    B=[1,-1,-1;-3,3,-3;-1,1,1]
    for i=2:n
        Aux= ---------
        Xold= ---------
        Xnew= ---------
    end
    res= ---------
endfunction
```

2. *Pour tout* $n \in \mathbb{N}$, *on note* $X_n = \begin{pmatrix} \alpha_n \\ \beta_n \\ \gamma_n \end{pmatrix}$ *et on admet que* $\beta_n = \left(\frac{1}{2}\right)^{n-1} - \frac{2}{3}\left(-\frac{1}{2}\right)^n - \frac{4}{3}$.

   (a) *Vérifier, sur quelques valeurs de* n, *que votre fonction renvoie un résultat cohérent avec cette formule.*

   (b) *La fonction de la question 1 a été utilisée dans un script pour obtenir la représentation graphique des suites* (α_n), (β_n) *et* (γ_n) :



Reconnaître la représenation graphique de (β_n).

Exercice 7

Expliquer ce que fait le script suivant

```
n=input('Entrez un entier naturel n')
S=0
for i=1:n
    S=S+i
end
```

Exercice 8 (*Bonus à faire à la fin si le temps le permet*)

On considère la suite $(u_n)_{n\in\mathbb{N}}$ définies par

$$u_0 = 1 \quad \text{et} \quad \forall n \in \mathbb{N}\ u_{n+1} = 2u_n + n^2.$$

1. *Écrire une fonction qui prend en argument un entier naturel n et qui renvoie le vecteur $[u_0, \dots, u_n]$.*
2. *Avec la commande `sum` et la fonction de la question 1, écrire une fonction qui prend en argument un entier naturel n et qui renvoie la somme partielle $\sum_{k=0}^{n} u_k$.*
3. *Avec la commande `cumsum` et la fonction de la question 1, écrire une fonction qui prend en argument un entier naturel n et qui renvoie le vecteur $[S_0, \dots, S_n]$ où $S_i = \sum_{k=0}^{i} u_k$.*

2.2 Boucle `while`

Boucle `while`

La syntaxe est la suivante :

```
while condition
    instruction
end
```

où

- `condition` est une condition (c'est-à-dire une expression qui prend la valeur vraie ou fausse),
- `instruction` est un bloc d'instructions qui va être exécuté tant que `condition` est vraie.

Exercice 9 (*EDHEC 2016*)

On considère une suite $(u_n)_{n\in\mathbb{N}}$ telle que

$$\forall n \in \mathbb{N},\ e^{-\sqrt{u_n}} \leqslant u_n - n \leqslant e^{-\sqrt{n}}.$$

1. Recopier et compléter le script Scilab suivant afin qu'il affiche un entier n pour lequel $u_n - n$ est inférieur ou égal à 10^{-4}.

```
n=0
while -----
    n = -----
end
disp(n)
```

2.

Exercice 10 (*d'après EML 2016*)

On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par

$$\begin{cases} u_0 = \frac{1}{2} \\ \forall n\in\mathbb{N}\ u_{n+1} = f(u_n) \end{cases} \quad \text{où} \quad f : t\in\mathbb{R}\mapsto \begin{cases} t^2 - t\ln t & \text{si } t\neq 0 \\ 0 & \text{si } t=0 \end{cases}$$

On admettre que $\forall n\in\mathbb{N}$, $u_n > 0$.

1. *Écrire une fonction qui prend en argument un entier n et qui renvoie le vecteur $[u_0,\dots,u_n]$.*
2. *Modifier la fonction précédente pour qu'elle affiche la représentation graphique des n premiers termes de la suite (on pourra s'inspirer de l'exercice 5).*
3. *A l'aide de la fonction précédente, afficher la représentation graphique des 100 premiers termes. Que pouvez-vous conjecturer quant à la monotonie de $(u_n)_{n\in\mathbb{N}}$ et sa limite ?*
4. *On admet que $(u_n)_{n\in\mathbb{N}}$ converge vers 1. Écrire un script qui calcule et affiche un entier naturel n tel que $|1-u_n| < 10^{-3}$.*
5. *(une fois que tout le TP est fini, s'il reste du temps)*
   (a) *Justifier que f est $\mathscr{C}^2$, calculer f'', en déduire les variations de f' puis son signe, en déduire les variations de f.*
   (b) *Monter que pour tout n, $u_n\in[0,\frac{1}{2}]$.*
   (c) *Montrer que $(u_n)_{n\in\mathbb{N}}$ est croissante.*
   (d) *Montrer que $(u_n)_{n\in\mathbb{N}}$ converge et déterminer sa limite (on pourra étudier la fonction $t\mapsto t-\ln t$).*

3 Graphiques

La commande `plot2d`

- Si X et Y sont deux vecteurs lignes ou deux vecteurs colonnes de même taille, l'instruction `plot2d(X,Y)` trace la ligne brisée reliant les points dont les abscisses sont données par X et les ordonnées par Y.
- Si X, Y et Z sont trois **vecteurs colonnes de même taille**, l'instruction `plot2d(X,[Y,Z])` trace Y en fonction de X puis Z en fonction de X. Cela se généralise avec la commande `plot2d(X,[Y_1,Y_2,...,Y_n])`
- La commande `clf` permet de supprimer le contenu de la fenêtre graphique courante.

Tracer de fonctions

Si X est un vecteur ligne et `f` une fonction préalablement créée, l'instruction `fplot2d(x,f)` permet de tracer la ligne brisée reliant les points $\{(x,f(x)),\ x$ coefficient de X$\}$.

Exercice 11

Soit f la fonction définie sur $[-1,4]$ par

$$\forall x\in[-1,4],\ f(x)=\sqrt{1+x}.$$

1. *Définir une fonction `f11` qui prend en argument un nombre x dans $[-1,4]$ et qui renvoie $\sqrt{x+1}$.*
2. *Définir une fonction `graphe11` qui prend en argument un entier naturel n (et ne possède pas de sortie) et*

qui affiche la représentation graphique de `f11` avec n points (on pourra utiliser la commande `linspace` pour obtenir un vecteur de taille n constitué de nombre uniformément espacés entre -1 et 4).

3. *Modifier la fonction `graphe11` pour qu'elle affiche aussi, sur la même fenêtre graphique, la courbe de la fonction identité.*
4. *On s'intéresse à la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = -0.5$ et $\forall n \in \mathbb{N}$, $u_{n+1} = f(u_n)$.*
   (a) *On considère les points $A_0 = \begin{pmatrix} u_0 \\ u_0 \end{pmatrix}$, $A'_0 = \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}$, $A_1 = \begin{pmatrix} u_1 \\ u_1 \end{pmatrix}$, $A'_1 = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$,..., $A_n = \begin{pmatrix} u_n \\ u_n \end{pmatrix}$, $A'_n = \begin{pmatrix} u_n \\ u_{n+1} \end{pmatrix}$. Écrire une fonction qui prend en argument un entier naturel n et qui renvoie la matrice*

   $$\begin{pmatrix} A_0 & A'_0 & \dots & A_n & A'_n \end{pmatrix} = \begin{pmatrix} u_0 & u_0 & u_1 & u_1 & \dots & u_n & u_n \\ u_0 & u_1 & u_1 & u_2 & \dots & u_n & u_{n+1} \end{pmatrix}$$

   (b) *Écrire une fonction qui prend en argument un entier naturel n et qui affiche, sur une même fenêtre graphique, la courbe représentative de f, de la fonction identité et la ligne brisée reliant les points A_0, A'_0, ..., A_n, A'_n.*

   (c) *Tester la fonction avec $n = 100$ ou $n = 1000$. Quel résultat du cours cela illustre-t-il?*

4 Éléments du programme officiel

1. Les commandes suivantes, rencontrées durant ce TP, ainsi que leurs arguments sont exigibles :

   `sum` , `cumsum` , `zeros` , `ones` , `eye`

2. Les commandes suivantes, rencontrées durant ce TP, doivent savoir être manipulées mais la connaissance de leurs arguments n'est pas exigible :

   `plot2d` , `fplot2d`

3. Savoir-faire exigibles (première année) :
   - calcul des termes d'une suite
   - calcul de la valeur approchée de la limite d'une suite ou de la somme d'une série.