Chapitre 1 : Utilisation de la console.

1. Introduction.

A l'ouverture de Scilab, une fenêtre de commande ou console s'ouvre et une invite de commande `→` indique que le logiciel attend les instructions.

Cette première fenêtre fonctionne comme une calculatrice. On valide les instructions en tapant sur "entrée"

Si l'instruction se termine par un point virgule, l'instruction est effectuée mais n'est pas affichée.

Pour effacer la console, on peut taper "clc" (clear console) ou aller dans le menu "edition" puis "effacer console".

Remarques :

- Les signes `//` indiquent à Scilab que tout ce qui est à droite de ces signes ne doit pas être lu. On l'utilisera donc pour écrire des commentaires.
- On a possibilité d'écrire plusieurs instructions sur une même ligne en les séparant par des "," ou des ";".

2. Variables, affectations, constantes,

2.1. Variables et affectations.

Les variables correspondent à un espace de stockage.

Elles sont créées dès qu'elles sont introduites par l'instruction d'affectation `=` et ces variables prennent le type de l'expression.

Affectation : `Nom = expression .`

L'expression peut-être du type numérique (réel ou complexe), matriciel ou de type chaîne de caractère.

2.2. Constantes

Scilab connaît deux constantes réelles prédéfinies π et e notées `%pi` et `%e`.

Le nombre imaginaire i est noté `%i`.

Scilab connaît deux constantes booléennes prédéfinies `vrai` et `faux` notées `%t` et `%f`.

2.3 Chaînes de caractères

Une chaîne de caractères est une suite de caractères écrits entre apostrophes (`'`) ou guillemets droits doubles (`"`).

2.4. Variable logique.

Une variable logique ou booléenne correspond à une expression logique. Elle peut prendre deux valeurs : vraie (`%t`) ou fausse (`%f`).

3. Les opérateurs en Scilab.

3.1. Opérateurs arithmétiques .

+ - * / ^

addition soustraction multiplication division puissance

Exercice 1 : Taper :

`x = 5; y = 2;`

`x + y, z = x - y`

`x * z, x/y, x^y`

3.2. Opérateurs relationnels et logiques.

Logiques : & | ~
 et ou non

Comparaisons et tests : == < > >= <= <>
 égalité < > ≥ ≤ ≠

Exercice 2 :

Taper :

a=2>3, b=2<3, c=%t;

a&b, a|b

d=~a&c, d& b

3.3. Les fonctions usuelles prédéfinies.

log	exp	floor	abs	sqrt	sin	cos
logarithme	exponentielle	partie entière	valeur absolue	$\sqrt{\quad}$	sinus	cosinus

4. Instructions d'entrée et de sortie.

4.1. Instruction d'entrée.

L'entrée par l'utilisateur d'une variable numérique ou d'une matrice se fait par la commande `input`.

L'instruction `x = input(' Donner une valeur à x : ')` affiche le texte entre apostrophe puis Scilab attend une réponse.

Si elle est correcte, la réponse est affectée à la variable x.

Cette instruction permet donc de demander avec une phrase la valeur à attribuer à une variable.

Exemple :

Taper :

`e = input(" somme en euros : ");`

4.2. Instructions de sortie.

L'instruction `disp(x)` affiche le contenu de la variable x.

L'instruction `disp('texte')` la chaîne de caractères entre apostrophes.

Exemple :

Taper

`a = 5;`

`disp(' la variable a est égale à ',a)`

`disp(a, ' la variable a est égale à :')`

Chapitre 2 : Calcul matriciel.

1. Définition d'une matrice .

Pour une matrice de petite dimension, il suffit d'écrire les coefficients de cette matrice entre crochets de la façon suivante :

- les séparateurs des lignes sont des points virgules
- les séparateurs entre les éléments de chaque ligne étant soit des virgules soit des espaces.

Exemple : La matrice $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ s'écrit en Scilab : $A = [1\ 2\ 3; 4\ 5\ 6]$.

Le vecteur ligne $L = (1\ 2\ 3)$ s'écrit en Scilab : $L = [1\ 2\ 3]$.

Le vecteur colonne $C = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ s'écrit en Scilab : $C = [1; 2; 3]$.

Exercice 1 : Taper les matrices suivantes : $X = \begin{pmatrix} -1 & 8 \\ 3 & 5 \end{pmatrix}$, $Y = \begin{pmatrix} -6 & 7 \\ 2 & 8 \\ 5 & 9 \end{pmatrix}$

2. Matrices particulières.

Matrice nulle : La matrice $0_{n,p}$ de $M_{n,p}(\mathbb{R})$ est notée en Scilab : $M = \text{zeros}(n,p)$.

Matrice unité ou identité : La matrice unité I_n de $M_n(\mathbb{R})$ est notée en Scilab : $M = \text{eye}(n,n)$.

La matrice M à n lignes et p colonnes ne possédant que des 1 est notée : $M = \text{ones}(n,p)$

Remarque :

Ces commandes permettent d'initialiser les matrices avant de pouvoir faire des calculs de manière rapide.

3. Vecteurs particuliers.

Pour introduire des vecteurs lignes dont les coefficients sont régulièrement espacés (exemple : $(1, 3, 5, 7)$) on dispose de deux commandes .

La première commande " $L = a : h : b$ " donne un vecteur ligne L contenant des valeurs allant de a à b par pas de h .

Ainsi la commande " $L = 1 : 2 : 7$ " donne le vecteur ligne : $L = (1, 3, 5, 7)$.

La commande " $L = a : b$ " est équivalente à la commande " $L = a : 1 : b$ ", la longueur du pas lorsqu'elle est égale à 1 pouvant être omise.

La deuxième commande " $L = \text{linspace}(a,b,n)$ " donne un vecteur ligne de longueur n et d'extrémité a et b . Le pas est donc égal à $\left(\frac{b-a}{n-1}\right)$.

Ainsi la commande " $L = \text{linspace}(1,7,4)$ " donne le vecteur ligne : $L = (1, 3, 5, 7)$.

Exercice 2 : Ecrire le vecteur ligne $(0, 2, 4, 6, 8)$ à l'aide des deux commandes.

4. Opérations sur les matrices.

4.1. Opérations usuelles

L'instruction " $\text{size}(A)$ " donne le type de la matrice A . Cette instruction renvoie une matrice ligne à deux coefficients correspondant au nombre de ligne et au nombre de colonne de A .

L'instruction " $\text{rank}(A)$ " donne le rang de A .

Ecriture en Scilab

Addition	$A + B$	$A + B$
Multiplication	$A * B$	$A * B$
Soustraction	$A - B$	$A - B$
Multiplication par un réel	$\alpha \cdot A$	$\alpha * A$
Puissance	A^n	$A^{\wedge}n$
Inverse	A^{-1}	$inv(A)$
Transposée	tA	A'

4.2. Opérations coefficients par coefficients.

Les opérations pointées suivantes $\cdot *$, $\cdot /$, $\cdot ^$ permettent d'effectuer respectivement un produit coefficient par coefficient, une division coefficient par coefficient et une puissance coefficient par coefficient.

Soient $A = (a_{ij})$ et $B = (b_{ij})$ deux matrices de $M_{n,p}(\mathbb{R})$.

L'instruction " $A \cdot *B$ " donne la matrice de $M_{n,p}(\mathbb{R})$ de coefficients $a_{ij} \times b_{ij}$.

L'instruction " $A \cdot /B$ " donne la matrice de $M_{n,p}(\mathbb{R})$ de coefficients a_{ij}/b_{ij} .

L'instruction " $A \cdot ^n$ " donne la matrice de $M_{n,p}(\mathbb{R})$ de coefficients $(a_{ij})^n$.

Plus généralement, si f est une fonction, la fonction f appliquée à la matrice $A = (a_{ij})$ donne une matrice C dont les coefficients sont égaux aux $f(a_{ij})$.

Exercice 3 : Soient $A = \begin{pmatrix} 1 & -1 & 1 \\ 2 & 1 & 0 \end{pmatrix}$ et $B = \begin{pmatrix} 6 & 7 & -3 \\ 1 & 1 & -1 \end{pmatrix}$.

Taper : $A \cdot *B$, $A \cdot /B$, $A \cdot ^4$, $abs(A)$.

Remarque :

Scilab permet d'ajouter ou de soustraire un scalaire à une matrice mais c'est une facilité de programmation.

Ainsi l'instruction " $\rightarrow A + 2$ " permet d'ajouter le réel 2 à tous les coefficients de A donc est identique à l'instruction

" $\rightarrow A + 2 * ones(n,p)$ "

5. Extraction ou modification.

5.1. Extraction.

Pour un vecteur u ligne ou colonne, la commande " $x = u(k)$ " permet d'extraire le $k^{i\grave{e}me}$ élément de u .

Pour une matrice $A = (a_{ij})$ de $M_{n,p}(\mathbb{R})$,

l'instruction " $x = A(i,j)$ " permet d'extraire le coefficient a_{ij} de A .

l'instruction " $L = A(i,:)$ " permet d'extraire la $i^{i\grave{e}me}$ ligne de A .

l'instruction " $C = A(:,j)$ " permet d'extraire la $j^{i\grave{e}me}$ colonne de A .

Remarque :

L'opérateur ":" sert donc à désigner une ligne ou une colonne.

Exercice 4 : Soit $A = \begin{pmatrix} 1 & -1 & 1 \\ 2 & 1 & 0 \end{pmatrix}$. Taper $x = A(2,3); L = A(2,:); C = A(:,2)$.

5.2. Modification.

Soient u un vecteur ligne ou colonne et une matrice $A = (a_{ij})$ de $M_{n,p}(\mathbb{R})$.

Soient $b \in \mathbb{R}, L \in M_{1,p}(\mathbb{R})$ et $C \in M_{n,1}(\mathbb{R})$.

L'instruction " $u(k) = b$ " remplace le $k^{i\grave{e}me}$ élément de u par b .

L'instruction " $A(i,j) = b$ " remplace le coefficient a_{ij} de A par b .

L'instruction " $A(i,:) = L$ " remplace la $i^{i\grave{e}me}$ ligne de A par L .

L'instruction " $A(:,j) = C$ " remplace la $j^{i\grave{e}me}$ colonne de A par C .

Exercice 5 : Soit $A = \begin{pmatrix} 1 & -1 & 1 \\ 2 & 1 & 0 \end{pmatrix}$. Taper $A(2,2) = 4; A(1,:) = ones(1,3); A(:,3) = ones(2,1)$.

Chapitre 3 : graphe en dimension deux.

1. Instruction plot et plot2d.

Pour tracer la courbe représentative d'une fonction f de une variable, Scilab trace une ligne brisée entre les points $(x_1, f(x_1)), \dots, (x_2, f(x_2))$.

Il faut donc d'une manière ou d'une autre fournir :

- un vecteur ligne $x = [x_1, \dots, x_n]$

- un vecteur ligne $y = [f(x_1), \dots, f(x_n)]$.

- Pour déclarer le vecteur x des abscisses, on a deux possibilités (voir chapitre 2, paragraphe 3).

Par exemple pour découper le segment $[0,1]$ en 100 sous-segments, on dispose des solutions suivantes :

$x = 0 : 0.01 : 1$

$x = \text{linspace}(0, 1, 101)$.

- Pour déclarer le vecteur y des ordonnées, on construit ce vecteur en utilisant les opérations coefficients par coefficients (voir chapitre 2, paragraphe 4.2)

- Pour tracer le graphe, on utilise ensuite les fonctions plot ou plot2d.

L'instruction $\text{plot}(x, y)$ ou $\text{plot2d}(x, y)$ trace donc une courbe en reliant les points (x_k, y_k) entre eux.

Exemple : Tracer la courbe de la fonction $f : x \mapsto x^2$ sur $[0; 1]$ avec un pas de 0,1.

- $\rightarrow x = 0 : 0.1 : 1;$

- $\rightarrow y = x \cdot ^2;$

- $\rightarrow \text{plot2d}(x, y)$

Exercice 1 : Tracer la courbe de la fonction $f : x \mapsto \cos(x)$ sur $[0; 2\pi]$ avec un pas de 0,01.

Exercice 2 : Tracer la courbe de la fonction $f : x \mapsto \frac{1}{1+x^2}$ sur $[-3, 3]$ pour 11 valeurs.

2. Tracés de plusieurs courbes sur un même graphique.

Par défaut, les différents tracés viennent se superposer les uns par dessus les autres.

Pour n'avoir qu'un seul tracé, il faut effacer le précédent avec la commande : *clf*.

A chaque appel de *plot2d* les valeurs maximales et minimales sont évaluées pour offrir un affichage qui permet de voir la courbe en entier.

Dans le cas de plusieurs appels successifs de la commande *plot2d*, les courbes sont superposées mais de plus, en cas de changement de valeurs extrêmes, les courbes sont ré-affichées pour permettre une visualisation simultanée.

Exercice 3 :

Tracer la fonction $f : x \mapsto \left(\frac{x}{1+x^3} \right)$ sur $[0; 1]$ et $g : x \mapsto \frac{x}{\exp(x)}$ sur $[-1; 2]$.

Mais si on veut imposer les coordonnées de la fenêtre d'affichage, on utilise l'instruction :

$\text{rect} = [x_{\min}, y_{\min}, x_{\max}, y_{\max}]$, qu'on place dans l'instruction *plot2d* : l'appel est fait par $\text{plot2d}(x, y, \text{rect} = \dots)$.

Exercice 4 :

Tracer sur une même figure la courbe de $f : x \mapsto \frac{x^2}{2x-1}$ et la courbe de l'asymptote d'équation $y = \frac{1}{2}x + \frac{1}{4}$ sur $[1; 4]$.

On définit pour fenêtre d'affichage : x varie de 1 à 4 et y de 0 à 3.

L'instruction *plot2d* permet d'afficher simultanément plusieurs courbes, pour cela il suffit d'utiliser en argument des matrices à la place des vecteurs lignes x et y .

L'appel est fait par $\text{plot2d}(xx, yy)$ où xx et yy sont deux matrices de même dimensions $m \times n$ et revient à superposer sur un même graphique n courbes représentant chacune des n colonnes de yy en fonction de la colonne correspondante de xx .

On a deux méthodes possibles :

- si le vecteur d'abscisses est commun à l'ensemble des ordonnées :

$\text{plot2d}([x; x; \dots]', [f_1(x); f_2(x), \dots]')$

- s'il y a autant de vecteurs d'abscisses et d'ordonnées : $plot2d([x_1;x_2,\dots]', [f_1(x_1);f_2(x_2),\dots]')$.

Exemple :

Taper les instructions :

- $\rightarrow x = linspace(1, 10, 101);$

- $\rightarrow plot2d([x ; x ; x]', [\sin(x); \cos(x); \sin(x) .* \cos(x)]')$

Exercice 5 :

1. Tracer les courbes des fonctions $x \mapsto \sin(x)$, $x \mapsto \sin(2x)$ et $x \mapsto \sin(3x)$ sur $[0; 2\pi]$.

2. Tracer les courbes des fonctions $x \mapsto \sin(x)$, $x \mapsto \sin(2x)$ et $x \mapsto \sin(3x)$ respectivement sur $[0; 2\pi]$, $[2\pi; 4\pi]$ et $[4\pi, 6\pi]$.

3. Choix de la couleur et du trait.

Pour obtenir une couleur ou un mode de tracé différent, on utilise l'instruction $style = (\text{vecteur ligne})$: l'appel est fait par $plot2d([x_1;x_2]', [y_1;y_2]', style = [n_1, n_2])$.

La taille du vecteur ligne correspond au nombre de vecteurs des ordonnées.

Ce tableau doit impérativement être en troisième position et on peut omettre d'écrire $style$.

Les valeurs numériques de ce vecteur ligne correspondent à une couleur ou à un symbole.

Plus précisément une valeur positive correspond à une couleur et une valeur négative ou nulle remplace l'affichage continu par un symbole.

1	2	3	4	5	6	7
<i>noir</i>	<i>bleu</i>	<i>vert</i>	<i>cyan</i>	<i>rouge</i>	<i>magenta</i>	<i>jaune</i>
0	-1	-2	-3	-4	-5	-6
•	+	×	⊕	*	◇	△

Exercice 6 :

Tracer la courbe de la fonction $x \mapsto x^2$ en rouge et de $x \mapsto \ln(1 + 2x)$ en vert sur $[0; 1]$ (pas de 0.01)

Exercice 7 : Tracer la courbe de la fonction f définie sur I avec un pas de 0,01 :

1. $f : x \mapsto x - \lfloor x \rfloor$ sur $I = [-10; 10]$.

2. $f : x \mapsto \lfloor \ln(x + 2) - 1 \rfloor$ sur $I = [-1; 1]$.

3. $f : x \mapsto \ln(\sqrt{x^4 + 2})$ sur $I = [0; 4]$.

Lors du tracé de $y = f(x)$, si le vecteur x est omis, alors les points tracés sont (k, y_k) .

Exercice 8 : Tracer les 21 premières valeurs de la suite u de terme général $u_n = (-2 + 0,2 \times n)^2$

Chapitre 4 : l'instruction répétitive *for*.

1. Utilisation de l'éditeur Scinotes.

Dès que l'on a besoin d'écrire des programmes un peu développés, on utilisera la fenêtre d'édition Scinotes, qui permet d'écrire, de sauvegarder, de modifier des programmes complets.

Pour ouvrir la fenêtre Scinotes, on clique sur l'icône située en haut à gauche de l'écran "démarrer Scinotes" qui ouvre une seconde fenêtre dans laquelle on pourra écrire le programme considéré.

On tape alors les commandes que l'on veut dans la fenêtre Scinotes, en enregistrant régulièrement grâce au menu "fichier" puis "enregistrer sous" pour le premier enregistrement, puis "enregistrer" pour les suivants (fichier enregistré avec l'extension *.sce*).

Afin de rendre un programme plus compréhensible, il est judicieux de placer des commentaires qui apparaissent à droite des caractères " // ".

2. Rédaction de la boucle *for*.

L'instruction *for* permet de faire une itération sur les composantes d'un vecteur ligne donné.

Rédaction de cette instruction :

Considérons le programme suivant :

```
for  $x = v$ ; //  $v$  est un vecteur ligne  
    suite d'instructions portant ou non sur la variable  $x$ ;  
end
```

Dans la commande ci-contre , l'instruction sera exécutée pour x prenant successivement comme valeurs les composantes de v .

Exemple :

```
For  $i = 1 : 5$ ; //  $i$  prend successivement les valeurs 1,2,3,4 et 5.  
     $disp(i.^2)$  // afficher  $i^2$  pour  $i$  variant de 1 à 5.  
end
```

3. Exercices sur l'instruction *for*.

Exercice 1 :

Ecrire un programme qui calcule et affiche la somme des N premiers nombres entiers pour un N non nul choisi par l'utilisateur.

Exercice 2 :

Ecrire un programme qui calcule et affiche le factoriel $n!$ pour un n choisi par l'utilisateur.

Exercice 3 :

Ecrire un programme qui calcule et affiche $\sum_{k=1}^n \frac{1}{k!}$ pour un n non nul choisi par l'utilisateur.

Exercice 4 :

Soit la suite u définie par : $u_{n+1} = 3u_n - 2$.

Ecrire un programme qui calcule et affiche les termes successifs de cette suite pour une valeur initiale u_1 et jusqu'à un rang n , ces deux valeurs étant entrées par l'utilisateur et représenter les termes successifs de la suite .

Exercice 5:

On considère la suite de Fibonacci définie par: $u_n = u_{n-1} + u_{n-2}$, $u_1 = 1$ et $u_2 = 1$.

Ecrire un programme qui calcule et affiche la valeur de u_n pour un n choisi par l'utilisateur.

Exercice 6 :

On considère les suites (a_n) et (b_n) définies par : $a_1 = 1$, $b_1 = 1$ et $\forall n \in \mathbb{N}^*$, $a_{n+1} = \frac{a_n}{n+1} - \frac{b_n}{(n+1)^2}$ et $b_{n+1} = \frac{b_n}{n+1}$

Ecrire un programme qui calcule et affiche les n premiers termes de chacune des suites (a_n) et (b_n) pour une valeur de n entrée par l'utilisateur et qui permet de représenter sur un même graphique les termes successifs de ces suites .

Exercice 7 :

On considère les suites définies par :
$$\begin{cases} u_1 = 1 & v_1 = 2 \\ \forall n \in \mathbb{N}^* & v_{n+1} = \frac{u_n + v_n}{2} & u_{n+1} = \sqrt{u_n v_n} \end{cases} .$$

1. Ecrire un programme qui calcule et affiche les n premiers termes de chacune des suites (u_n) et (v_n) pour une valeur de n entrée par l'utilisateur et qui permet de représenter sur un même graphique les termes successifs de ces suites .

2. En déduire le comportement de ces suites (monotonie, convergence, limite...)

Exercice 8 : Soit la suite $(u_n)_{n \in \mathbb{N}}$ définie par $u_0 = 1$ et $\forall n \in \mathbb{N} : u_{n+1} = u_n + \frac{1}{4}(2 - u_n^2)$.

On pose $f : x \mapsto x + \frac{1}{4}(2 - x^2)$.

1. Etudier les variations de f et déterminer ses points fixes.

Tracer sur une même figure la courbe de f et la courbe de la droite d'équation $y = x$ sur $[1; \sqrt{2}]$.

2. Ecrire un programme qui calcule et affiche les n premiers termes de la suite (u_n) pour une valeur de n entrée par l'utilisateur et qui permet de représenter sur un même graphique les termes successifs de cette suite .

3. Etudier le comportement asymptotique de la suite (u_n) et donner en cas de convergence la limite de u .

Exercice 9 :

Ecrire un programme qui demande à l'utilisateur un entier naturel non nul n et qui permet d'afficher une matrice triangulaire supérieure de $M_n(\mathbb{R})$ dont les coefficients diagonaux sont égaux à 4 et tous les coefficients au-dessus de la diagonale égaux à -1.

Chapitre 5 : Instruction conditionnelle If.

1. Rédaction de l' instruction conditionnelle if.

L' instruction conditionnelle de base permet de donner des instructions différentes selon qu'un test est vrai ou faux.

if cond 1 then

Première rédaction : *Instruction A*

end

Si la condition booléenne 1 est vraie alors l' instruction A est réalisée.

if cond 2 then

Instruction A

Deuxième rédaction : *else*

Instruction B

end

Si la condition booléenne 2 est vraie, l' instruction A est exécutée sinon l' instruction B est exécutée.

Remarques :

- Il existe une autre syntaxe de *if* utile pour écrire des instructions tenant sur une ligne , qui utilise des virgules ",," :

if condition then, Instruction A, else, Instruction B, end

- Quand *else* est suivi d'un *if* , on peut utiliser la forme *elseif* de la façon suivante :

if cond 1 then

Instruction A1

elseif cond 2 then

Instruction A2

else Instruction A3

end

L' utilisation de *if elseif* permet de spécifier plusieurs conditions. Seules les instructions correspondant à la première condition vérifiée sont exécutées.

- La seule contrainte est que chaque mot-clé *then* doit être sur la même ligne que le *if* ou le *elseif* correspondant.

2. Exercices .

Exercice 1 :

On considère le programme suivant :

a=input("donner la valeur de a");

b=input("donner la valeur de b");

if a < b then

aux = a

a = b

b= aux

end

disp(a,"a=");

disp("b=",b);

1. Pour *a = 2* et *b = 6*, donner les valeurs de *a* et *b* à la fin du programme. Expliquer l' action de ce programme sur *a* et *b*.

2. Idem pour *a = 9* et *b = 3*.

Exercice 2 :

On considère le programme suivant :

```
a=input("donner la valeur de a");
b=input("donner la valeur de b");
c=input("donner la valeur de c");
    if a > b then d = a
        else d = b
    end
    if c > d then d = c
    end
disp(d)
```

1. Pour $a = -1$, $b = 6$, $c = 9$, donner la valeur affichée par l'ordinateur. Expliquer l'objectif de ce programme.
2. Idem pour $a = 8$, $b = 17$, $c = -4$

Exercice 3 :

1. Ecrire un programme qui demande à l'utilisateur un entier n et qui affiche sa parité.
2. Ecrire un programme qui permet de savoir si une année est bissextile.

Les années bissextiles sont :

- soit divisibles par 4 mais non divisibles par 100
- soit divisibles par 400.

Rappel : la fonction notée "floor" correspond en Scilab à la fonction partie entière.

Exercice 4 :

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

$$f(x) = \begin{cases} \exp(-x) + x - 2 & \text{si } x \leq 0 \\ x \cos x - 1 & \text{si } x \in]0; 3] \\ \frac{x^2+2}{x-1} & \text{si } x > 3 \end{cases} .$$

Ecrire un programme qui demande à l'utilisateur une valeur x et qui affiche la valeur de $f(x)$.

Exercice 5 :

Ecrire un programme qui demande à l'utilisateur un entier naturel non nul n et qui permet d'afficher la matrice de $M_n(\mathbb{R})$ de la forme suivante :

$$A_n = \begin{pmatrix} 2 & -1 & -1 & \cdots & -1 \\ 3 & 2 & \ddots & & \vdots \\ 3 & \ddots & \ddots & \ddots & -1 \\ \vdots & & \ddots & 2 & -1 \\ 3 & \cdots & 3 & 3 & 2 \end{pmatrix} .$$

Exercice 6 :

Ecrire un programme prenant en entrée un trinôme $aX^2 + bX + c$, avec $a \neq 0$, représenté par un tableau de ses coefficients et affichant ses racines réelles si elles existent ou "aucune racine réelle" sinon.

Chapitre 6 : l'instruction while.

1. Définition de l'instruction while.

1.1. Rédaction.

La boucle while correspond à "tant que ".

while cond 1

Instruction 1;

Rédaction : :

Instruction p;

end

Elle fonctionne de la façon suivante .

Si la condition booléenne cond 1 est vraie, le bloc d'instruction est exécuté, puis on recommence la boucle.

Sinon on s'arrête.

Il est clair que le bloc d'instructions doit pouvoir modifier l'environnement de manière à ce que l'expression booléenne puisse au bout d'un nombre fini d'étapes prendre la valeur fausse (%F).

1.2. Propriétés.

Quelle boucle utiliser ?

On utilise les boucles for lorsqu'on connaît à priori le nombre d'itérations. Dans le cas contraire on utilise une boucle while.

Comment sortir d'une boucle ?

La commande break permet de quitter une boucle for ou while.

Lorsque break est exécutée dans une boucle, cela arrête la boucle et on passe à l'instruction qui suit la boucle.

Comment compter la nombre d'itérations ?

Lors de l'utilisation de la boucle while, même si l'on ne connaît pas à l'avance le nombre d'itérations à effectuer, on peut avoir besoin de les décompter ou d'en connaître le nombre final.

C'est le cas lorsque l'on effectue des calculs sur des suites récurrentes.

Pour ce faire, il suffit d'introduire un compteur qui compte 1 à chaque passage dans la boucle.

On définit donc une variable k, à initialiser en dehors de la boucle, et on écrit dans la boucle l'instruction " $k = k + 1$ ".

2. Exercices.

Exercice 1 :

1. On considère le programme Scilab suivant :

```
v=input('donner un entier naturel v= ');
```

```
  s = 1;
```

```
  i = 1;
```

```
  while s < v
```

```
    i = 1+i;
```

```
    s= s + 1/i;
```

```
  end
```

```
  disp(s, 'la valeur est ');
```

Pour $v = 2$, donner la valeur affichée par l'ordinateur.

2. On démontre que la suite u définie pour tout $n \geq 1$ par $u_n = \sum_{k=1}^n \frac{1}{k} - \ln n$ est une suite décroissante,

minorée par 0 et donc convergente. Sa limite est appelée constante d'Euler C avec $C \approx 0.577215$.

Modifier le programme précédent pour qu'il détermine la valeur de n à partir de laquelle la valeur de u_n est une approximation de C à ε près, la valeur de ε étant choisie par l'utilisateur.

Exercice 2 :

On considère la suite réelle (u_n) définie par $u_0 = -0.01$ et $\forall n \in \mathbb{N}, u_{n+1} = u_n - u_n^2$.

1. Ecrire un programme Scilab qui demande à l'utilisateur de saisir un entier N et qui affiche tous les termes de la suite jusqu'au rang N .

2. Ecrire un programme qui calcule et affiche le premier rang n à partir duquel $u_n \leq -1000$.

Exercice 3 :

On considère la suite (u_n) définie par : $u_0 = 1, u_1 = 2$ et $u_{n+2} = u_{n+1} + 2u_n$.

Ecrire un programme qui calcule et affiche le premier rang n à partir duquel $u_n \geq 10$.

Exercice 4 :

(Etude de la suite de Syracuse)

On considère la suite u définie par : $u_0 \in \mathbb{N}$ et $\forall n \in \mathbb{N}, u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{sinon} \end{cases}$.

1. Ecrire un programme qui demande deux entiers naturels u_0 et n , qui renvoie la valeur de u_n et qui représente les termes de u .

2. Quel est le comportement de la suite u si $u_0 = 1$?

3. a. Ecrire un script qui calcule la plus petite valeur de n pour laquelle $u_n = 1$.

b. Compléter ce script pour qu'il calcule et affiche la plus grande valeur prise par la suite.

(On parle d'altitude de la suite)

Chapitre 7 : les fonctions en Scilab.

1. Les fonctions.

1.1. Fonctions renvoyant une seule valeur.

Scilab dispose d'un langage qui permet de définir de nouvelles commandes grâce à l'écriture des fonctions.

Une fonction est une suite d'instructions qui, après exécution, rend une valeur.

La déclaration d'une fonction renvoyant une seule valeur se présente de la manière suivante :

function $y = \text{nom-de-la fonction}(x_1, x_2, \dots, x_m)$

Bloc d'instructions

endfunction

Toutes les variables $y, x_1, \dots, x_m$ utilisées à l'intérieur de la fonction sont par défaut locales. Ces variables sont des variables muettes et leurs noms peuvent être réutilisés dans la définition d'autres fonctions ou dans Scilab.

L'identificateur y joue un rôle particulier. Lors de l'appel d'une fonction, le bloc d'instructions est exécuté et, à la fin, on affecte à la variable y une valeur. C'est cette valeur qui sera rendue par la fonction, après avoir affecté aux variables $x_1, \dots, x_m$ des expressions.

Exemple 1 :

Pour définir la fonction $f : f(x) = \frac{3}{8 + \exp(-x)}$, nous définissons une fonction Scilab dans la console

dont le nom est f , avec un argument

x et une valeur de retour y (l'image $f(x)$).

– $\rightarrow$ *function* $y = f(x)$

– $\rightarrow$ $y = 3/(8 + \exp(-x));$

– $\rightarrow$ *endfunction;*

Pour appeler la fonction et calculer une valeur, il suffit par exemple de taper :

– $\rightarrow$ $f(0)$

ans =

0.3333333

ou

d'utiliser la commande *feval* :

– $\rightarrow$ *feval*(0, f)

ans =

0.3333333

Remarque sur la fonction *feval* :

l'instruction $y = \text{feval}(x, f)$ renvoie le vecteur y définie par $y(i) = f(x(i))$.

1.2. Fonctions retournant plusieurs valeurs.

Rédaction :

function $[y_1, \dots, y_p] = \text{nom-de-la fonction}(x_1, x_2, \dots, x_m)$

Bloc d'instructions

endfunction

Pour récupérer toutes les valeurs, on utilise les crochets " [] ". Sinon on ne récupère qu'une seule valeur, c'est la première qui est retournée. (la plus à gauche)

Exemple 2 :

– $\rightarrow$ *function* $[m, M] = \text{min max}(x, y)$

– $\rightarrow$ $m = \text{min}(x, y); M = \text{max}(x, y);$

– $\rightarrow$ *endfunction;*

Pour appeler la fonction, on utilise l'instruction :

$[m, M] = \text{min max}(x, y)$.

ou

`feval([x,y],minmax);`

1.3. Charger une fonction.

Pour que la fonction soit reconnue et exécutée par Scilab, il faut la "charger". En effet les fonctions sous Scilab ont un statut de variables et donc nécessitent d'être initialisées.

Même s'il est possible de définir les fonctions dans la console comme dans les deux exemples précédents, il vaut toujours mieux les écrire dans Scinotes, les sauver dans un fichier et les charger ensuite dans Scilab.

Il sera ainsi beaucoup plus facile de réaliser des corrections ou des modifications.

1.4. Fonction "deff".

La fonction "deff" permet de construire simplement des fonctions mathématiques. Elle prend en argument deux chaînes de caractères la première correspondant à l'en-tête de la fonction définie et la seconde correspondant à sa "définition mathématique".

Exemple 3 :

Soit $f : x \rightarrow \exp(x) + \ln(x + 1)$.

– $\rightarrow$ `deff('y = f(x)', 'y = exp(x) + log(x + 1)');`

2. Exercices .

Exercice 1 :

1. Créer une fonction factorielle (nommée *fact*) qui prend en argument n et qui renvoie la valeur $n!$.
2. Créer ensuite un programme qui demande deux entiers n et p et qui affiche la valeur $\binom{n}{p}$.

Exercice 2* :

On considère la suite u définie par :
$$\begin{cases} u_0 = 1 \\ u_{n+1} = \frac{1}{2} \ln(1 + 2u_n) \end{cases}$$

1. Créer une fonction $h : x \mapsto \frac{1}{2} \ln(1 + 2x)$ sous scilab afin de calculer tout terme de rang n de la suite u , n étant saisi par l'utilisateur.
2. Représenter graphiquement les 20 premiers termes de la suite. Conjecturer sur le comportement de cette suite.
3. Déterminer le rang N à partir duquel $|u_N| \leq 10^{-4}$.

Exercice 3 :

Critère des séries alternées.

Si u est une suite décroissante et convergente de limite nulle, alors la série $S = \sum (-1)^n u_n$ converge.

De plus, si on note, pour tout entier $p \in \mathbb{N}$ $R_p = \sum_{n=p+1}^{+\infty} (-1)^n u_n$ le reste d'ordre p de la série S , on a

l'encadrement : $|R_p| \leq u_{p+1}$.

1. Justifier la convergence de la série de terme général : $v_n = (-1)^n \ln(1 + \exp(-n))$.
2. Écrire un programme qui donne une approximation à 10^{-6} près de la somme de cette série. La fonction $k : x \mapsto \ln(1 + \exp(-x))$ sera définie en tête de programme.

Exercice 4 : Tracer des graphes en Scilab.

Objectif : Représenter la courbe de la fonction $g : x \mapsto x \exp(-x)$ sur $[-1; 5]$ (avec un pas de 0.01).

Il faut dans un premier temps déclarer le vecteur ligne ou colonne x des abscisses: $x = [-1 : 0.01 : 5];$

Nous allons maintenant voir deux méthodes pour déclarer le vecteur y des ordonnées .

1. En utilisant les opérations pointées sur les vecteurs, construire le vecteur y des ordonnées puis représenter la courbe de g .

2. En utilisant la définition d'une fonction réelle sous Scilab et la fonction f_{eval} .

Créer la fonction réelle g sous Scilab, définir le vecteur y des ordonnées en utilisant f_{eval} puis représenter la courbe de g .

Remarque : une dernière méthode serait d'utiliser les opérations pointées à l'intérieur de la fonction et de définir le vecteur y des ordonnées par l'instruction : $y = g(x)$.

Exercice 5 :

On considère la fonction numérique f définie sur $[1, 2]$ par:

$$f(x) = 2(1 - e^{-x}) - x.$$

1. Etudier les variations de f sur $[1, 2]$.

Représenter la courbe de la fonction f sur $[1; 2]$ (avec un pas de 0.01).

2. Montrer que l'équation $f(x) = 0$, admet une unique solution dans $[1, 2]$, notée L .

On définit deux suites numériques $(a_n)_{n \geq 0}$ et $(b_n)_{n \geq 0}$ par : $a_0 = 1, b_0 = 2$

et pour tout entier naturel n , si $f(a_n)f(\frac{a_n+b_n}{2}) > 0$, alors, $a_{n+1} = \frac{a_n+b_n}{2}$ sinon, $a_{n+1} = a_n$
 $b_{n+1} = b_n$ $b_{n+1} = \frac{a_n+b_n}{2}$

3. On donne les valeurs de f suivantes : $f(\frac{3}{2}) \simeq 0,05$ et $f(\frac{7}{4}) \simeq -0,1$. Déterminer les valeurs de a_1 et b_1 puis de a_2 et b_2 .

4. Montrer, par récurrence que, pour tout entier n : $a_n \leq L \leq b_n$.

5. Montrer que, pour tout entier n , $b_n - a_n = (\frac{1}{2})^n$ et étudier le sens de variation des suites $(a_n)_{n \geq 0}$ et $(b_n)_{n \geq 0}$.

6. En déduire la convergence de ces deux suites, et préciser leur limite.

7. Ecrire un programme en Scilab, donnant une valeur approchée de L à 10^{-5} près et indiquant le nombre d'itérations nécessaires. La fonction f sera définie en tête de programme.

Chapitre 8 : Simulation de lois de probabilités avec Scilab.

1. La commande `rand`.

Rappel : Déf : On dit que X suit une loi uniforme sur $[a, b[$ si X est une va à densité de densité f définie par :

$$\begin{cases} f(x) = \frac{1}{b-a} \text{ si } x \in [a, b[\\ f(x) = 0 \text{ si } x \notin [a, b[\end{cases} .$$

On note $X \hookrightarrow U([a, b[)$.

Proposition : La fonction de répartition d'une loi uniforme sur $[a, b]$ est définie par :

$$\begin{cases} F(x) = 0 \text{ si } x < a \\ F(x) = \frac{x-a}{b-a} \text{ si } x \in [a, b[\\ F(x) = 1 \text{ si } x \geq b \end{cases} .$$

Définition de la fonction `rand( )` :

- la fonction `rand( )` donne un résultat choisi "au hasard" dans $[0; 1[$ donc suivant la loi uniforme sur $[0; 1[$.

- Pour tous entiers m et n , la fonction `rand(m,n)` donne une matrice à m lignes et n colonnes, chaque coefficient étant choisi selon une loi uniforme sur $[0; 1[$.

Exemple 1 : Taper dans la console les lignes d'instructions :

```
-> x = rand()
-> x = floor(10 * rand())
-> x = rand(3, 2)
-> x = floor(10 * rand(3, 2))
```

2. Histogrammes.

Si y est un vecteur, la commande `bar(y)` trace un histogramme dont la hauteur de chaque barre est donnée par les coefficients de y . On peut aussi préciser les abscisses des barres en donnant un vecteur x de même longueur (même nombre de colonne) que y : `bar(x,y)`. L'instruction `bar` suppose qu'on a déjà ordonné les données par classe : y correspond aux effectifs et x à la liste des différentes classes.

Ce travail fastidieux peut être automatisé par l'instruction `histplot`. Scilab dispose de la commande `histplot` pour tracer des histogrammes avec deux utilisations possibles :

- si x est une matrice dont les coefficients sont les données, la commande `histplot(n,x)`, répartit les données dans n classes de même largeur et trace l'histogramme correspondant, l'effectif de chaque classe étant normalisé par l'effectif total. (de telle sorte que l'aire totale soit égale à 1)

- si $b=[b_1, \dots, b_p]$ est un vecteur ligne définissant les classes $[b_1, b_2[, \dots, [b_{p-1}, b_p[$ et si x est une matrice dont les coefficients sont les données, la commande `histplot(b,x)`, répartit les données dans les classes définies par b et trace l'histogramme correspondant, l'effectif de chaque classe étant normalisé par l'effectif total.

Exemple 2 : Taper dans la console les lignes d'instructions :

```
-> y = floor(10 * rand(1, 5))
-> bar(y)
-> z = linspace(5, 21, 5)
-> bar(z, y)
-> x = floor(10 * rand(4, 2))
-> histplot(4, x).
```

Remarque : La fonction `histplot(b,x)` dessine un histogramme des données contenues dans le vecteur x en utilisant les classes b . Quand le nombre de classes n est fourni au lieu de b , celles-ci sont définies telles que : $b(1) = \min(x) < b(2) = b(1) + db < \dots < b(n+1) = \max(x)$ avec $db = (b(n+1) - b(1))/n$. Les classes sont définies par $C1 = [b(1), b(2)]$ puis $Ci =] b(i), b(i+1)]$ pour $i = 2, 3, \dots, n$.

En notant $N_{\max}$ le nombre total de données ($N_{\max} = \text{longueur}(x)$) et N_i le nombre de données se situant dans C_i , la valeur de l'histogramme pour x appartenant à C_i est égale à $N_i / (N_{\max} (x(i+1) - x(i)))$

quand on normalise (comportement par défaut) et sinon elle vaut simplement N_i .

3. Commande grand et simulations des variables aléatoires discrètes usuelles.

La fonction `grand` permet de simuler des lois usuelles. Ainsi

$Y = \text{grand}(\text{nombre} - \text{lignes}, \text{nombres} - \text{colonnes}, \text{nom} - \text{loi}, \text{paramètres})$ permet de créer une matrice de taille $r \times s$, chaque coefficient étant choisi suivant la loi.

Soient r et s deux entiers naturels non nuls.

La commande $A = \text{grand}(r, s, 'uin', 1, n)$ crée une matrice de taille $r \times s$, chaque coefficient étant choisi selon la loi uniforme sur $[1; n]$ de manières indépendantes.

La commande $A = \text{grand}(r, s, 'bin', n, p)$ crée une matrice de taille $r \times s$, chaque coefficient étant choisi selon la loi binômiale de paramètres (n, p) de manières indépendantes.

La commande $A = \text{grand}(r, s, 'geom', p)$ crée une matrice de taille $r \times s$, chaque coefficient étant choisi selon la loi géométrique de paramètre p de manières indépendantes.

La commande $A = \text{grand}(r, s, 'poi', \lambda)$ crée une matrice de taille $r \times s$, chaque coefficient étant choisi selon la loi de Poisson de paramètre λ de manières indépendantes.

Exemple 3 : Taper dans la console les lignes d'instructions :

```
-> x = grand(1000, 1, "bin", 30, 0.5);  
-> c = linspace(0, 30, 31);  
-> histplot(c, x).
```

Exercice 1 : On considère des lancers d'un dé cubique équilibré.

1) Que fait la séquence suivante :

```
S = 0;  
for i = 1 : 10  
    d = floor(6*rand()+1);  
    S = S+d ;  
end  
moy = S/10;  
disp(moy)
```

2) Ecrire un programme qui simule 1000 lancers de dés et qui compte et affiche le nombre de 6 obtenus. Ce programme permet de simuler quelle loi discrète usuelle ?

Exercice 2 : On se propose de simuler informatiquement une variable aléatoire.

On considère le programme informatique suivant :

```
ini=floor(3*rand());  
if ini==2 then  
    y=floor(2*rand()+1)  
else y=3  
end  
disp(y)
```

On appelle Y le contenu de y après exécution du programme exo 2.

Donner la loi de Y , calculer et son espérance $E(Y)$.

Exercice 3 : Simulation de la loi géométrique de paramètre p .

1. En utilisant la fonction `rand()`.

a. Soit $p \in]0, 1[$. Soit On note $X \leftrightarrow U([0, 1])$. Calculer $P(X \leq p)$.

b. Ecrire un programme en Scilab qui permet de simuler une loi géométrique Y de paramètre p .

2. En utilisant la fonction `grand()`.

a. A l'aide d'une boucle `while` et de la commande `grand(1, 1, "bin", 1, p)` (qui permet de simuler une loi de Bernoulli), simuler une va qui suit une loi géométrique Y de paramètre p .

b. A l'aide du générateur `grand` simuler directement un échantillon $i.id$ de taille 100 de loi géométrique de paramètre $p = 0.2$ et tracer l'histogramme correspondant . (Le vecteur b définissant les classes varie de 0.5 à 20.5 avec un pas de 1.)

Fiche 1 : utilisation de la console .

Exercice 1 : Taper successivement les commandes suivantes et vérifier ce qui se passe :

$16/(2^3)$		$\%pi$
$2 + 3 * 4$		$sqrt(2)$
$2 + 3 * \%i$	puis	$SQRT(2)$
pi		$sqrt(2);$

Exercice 2 : Taper successivement les commandes suivantes et vérifier ce qui se passe :

$x = 5$		$x = x + 7$
$x = 7;$		$disp('x')$
$8x$		$y = \sin(x^3)$
$8 * x$	puis	$disp(y, 'valeur de y')$
$3 * 5$		$a = input('entrer un entier naturel non nul')$
$x + 12$		$\cos(1 + \log(a)^2)$
$disp(x)$		

Exercice 3 :

Caculer les valeurs suivantes : 1. $a = (2e^\pi - 5)^2$ 2. $b = \frac{\sqrt{3} - 2\sqrt[3]{8} + 3\ln 5}{6}$
3. $c = \sqrt{\left| \ln\left(\cos\left(\frac{\pi}{5}\right)\right) \right|}$ 4. $d = (5 + 2i)(3 - 4i)$

Exercice 4 :

On considère le script suivant :

```
x = input('Entrer x :' )  
y = input('Entrer y :' )  
z = x + y;  
x = x * z;  
y = x - z;  
disp(z,y,x)
```

On suppose qu'on entre les valeurs 3 et 7 au clavier. Quelles sont alors les valeurs affichées à l'issue de l'exécution de la commande " $disp(z,y,x)$ " ?

Exercice 5 :

Quelles sont les valeurs affichées à l'issue de l'exécution des instructions de la ligne suivante :

```
-- > x = 1;  
-- > y = %pi;  
-- > floor(y)  
-- > y = ans - x;  
-- > x = ans^y  
-- > y = 6 + x  
-- > y = ans - y
```

Exercice 6 : Calculer le prix TTC d'un article à partir de son prix HT, avec un taux de TVA à 19,6%.

Exercice 7 : Convertir un temps donné en secondes, en heures, minutes et secondes.

(On utilisera la fonction partie entière "floor")

Fiche n° 2 : calcul matriciel .

Exercice 1 : Taper successivement les commandes suivantes et vérifier ce qui se passe, en comprenant les messages d'erreur éventuels :

$x = [2,3,-1]$		$C = 1 : 4$
$y = x'$		$C * C$
$A = y * x$	puis	$C .* C$
$I = eye(3,3)$		$C * C'$
$C = A * I$		$exp(C)$
$C = A .* I$		

Exercice 2 : On considère les matrices suivantes : $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ et $B = \begin{pmatrix} 1 \\ -1 \\ -3 \end{pmatrix}$.

Taper successivement les commandes suivantes et vérifier ce qui se passe, en comprenant les messages d'erreur éventuels :

$A * B$		$2 * (A - 3)$
$A * A$		$inv(A)$
$A .* A$	puis	$B * B'$
$sin(A)$		$C = inv(B)$
$A ./ A$		$C * B$

Exercice 3 : On considère les matrices suivantes :

$$A = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 2 & 1 \\ 2 & 1 & -1 \end{pmatrix}, B = \begin{pmatrix} 0 & 1 \\ -2 & 1 \\ 1 & 0 \end{pmatrix}, C = \begin{pmatrix} -1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \text{ et } D = \begin{pmatrix} 1 & -2 \\ 1 & -2 \end{pmatrix}$$

Calculer les matrices suivantes si possible et vérifier vos calculs avec Scilab.

- a) BA b) CB c) AB d) $BC - 2A^2$
 e) $B \times D$ f) $C \times B - D$ g) $D \times B$ h) $(D - I_2) \times C$.

Exercice 4 : Calculer les matrices suivantes si possible et vérifier vos calculs avec Scilab.

$$1. \begin{pmatrix} 1 & 0 & 2 \\ 3 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 0 & -3 \\ -1 & 1 \end{pmatrix} \quad 2. \begin{pmatrix} 2 \\ 4 \\ -3 \end{pmatrix} \times {}^t \begin{pmatrix} 2 \\ 4 \\ -3 \end{pmatrix} \quad 3. \begin{pmatrix} 5 & -1 & 2 \\ 0 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Exercice 5 : On note A , B et C les matrices suivantes :

$$A = \begin{pmatrix} 1 & 3 & 2 \\ -5 & 3 & 1 \\ -10 & 0 & 3 \\ 1 & 0 & -2 \end{pmatrix}, B = \begin{pmatrix} 1 & -2 & 5 \\ 6 & 1 & -1 \end{pmatrix}, C = \begin{pmatrix} 10 & -5 \\ 3 & 1 \end{pmatrix}.$$

Dans les questions qui suivent certaines opérations ne sont pas définies. On essaiera cependant de les exécuter avec Scilab de manière à comprendre les messages d'erreurs.

1. Calculer les matrices AB, BA et $A^t B$.
2. Calculer la matrice $D = I_2 - B^t B$.
3. Calculer les inverses des matrices A, B, C et D . Vérifier les résultats obtenus.

Exercice 6 : On donne : $A = \begin{pmatrix} 2 & 3 & -2 \\ 0 & 1 & 5 \end{pmatrix}$, $B = \begin{pmatrix} 2 & 1 & 2 \\ -4 & -1 & 1 \\ 3 & 0 & -3 \end{pmatrix}$, $C = \begin{pmatrix} -1 & -1 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 2 & 1 & 2 & 3 \end{pmatrix}$ et

$$D = \begin{pmatrix} 1 & 4 & -1 \\ -1 & 3 & 0 \end{pmatrix}.$$

1. Calculer $2A - 3D$ et vérifier votre calcul avec Scilab.
2. Vérifier par le calcul puis à l'aide de Scilab que $(A + D)C = AC + DC$.
3. Vérifier par le calcul puis à l'aide de Scilab que ${}^t(A \times C) = {}^t C \times {}^t A$.
4. Vérifier avec Scilab la propriété d'associativité du produit matriciel.

Exercice 7 : On pose : $A = \begin{pmatrix} 1 & -1 & 7 \\ -4 & 2 & 11 \\ 8 & 0 & 3 \end{pmatrix}$ et $B = \begin{pmatrix} 3 & -2 & -1 \\ 7 & 8 & 6 \\ 5 & 1 & 3 \end{pmatrix}$.

Que font les instructions suivantes ?

$3 * A$; $A .* B$; $A.^B$; $\cos(A)$; $\exp(B)$; $A(2,:) * B(:,3)$; $A(2,3) = 6$

Exercice 8 : Soient $A = \begin{pmatrix} 1 & 3 & 2 \\ 1 & 3 & 1 \\ -1 & 0 & 0 \\ 1 & 0 & -2 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & 1 & -1 & 1 \\ 0 & 1 & 0 & 3 \\ 1 & 2 & 1 & 2 \end{pmatrix}$.

1. On note u la deuxième colonne de A et v la dernière ligne de B , déterminer la matrice uv .
2. On note C la matrice obtenue à partir de la matrice A en remplaçant u par $-u$. Expliquer comment construire C à partir de A par des instructions *Scilab*.
3. Quelles sont les valeurs affichées à l'issue de l'exécution des instructions de la ligne suivante :
 $x = A(3,1); y = B(2,4);$
 $x * y$
 $A(2,:) * B(:,2)$
 $A(:,3).^2$

Exercice 9 : On considère la matrice $A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

1. Calculer A^2 et vérifier vos calculs avec Scilab.
2. Déterminer deux réels a et b tels que $A^2 - aA - bI = 0$. Vérifier vos solutions à l'aide de Scilab.
3. En déduire que A est inversible et déterminer A^{-1} . Vérifier votre résultat à l'aide de Scilab.