Sujet TP2 OL2

Objectifs pédagogiques : Découvrir les instructions et commandes structurées de Scilab.
Utiliser les commandes de xcas pour donner la valeur exacte d'intégrales.

1. Les instructions et commandes structurées de Scilab

1.1 L'instruction for

Syntaxe :
```
for variable=deb:pas:fin
    instructions
end
```

Remarque : le **pas** peut-être négatif et vaut 1 lorsqu'il n'est pas précisé.

➢ **Exercice1 :** taper les instructions ci-dessous dans **un fichier script,**

```
1 clear
2 close
3 x=0:0.1:20                    //création du vecteur x
4 for i=1:length(x)             //début de la boucle for
5     y(i)=x(i)+cos(x(i))       //création des éléments du vecteur y pas à pas
6 end                           //fin de la boucle for»
7 plot2d(x,y,2)                 //tracé les points (x(i),y(i)) reliés
```

a. Donner l'équation de la courbe tracée par scilab.
b. Donner l'intervalle des abscisses et le nombre de points reliés.
c. A la place des lignes 4 à 6, y mettre cette ligne : **y=x+cos(x)**
d. Qu'observez-vous ?
e. Déterminer la somme des éléments du vecteur **x**, en utilisant une boucle for puis en utilisant la fonction **sum** de scilab (taper **Help sum** dans la console de scilab)

☞ **remarque :** dans beaucoup de cas on peut se passer de la boucle for en scilab, sachant que les fonctions prédéfinies sous Scilab acceptent en entrée un vecteur-tableau.

1.2. Les opérateurs logiques

Les opérateurs logiques (&, |, ~) et relationnels (<, >, <=, >=, ==, ~=) peuvent être nécessaire pour programmer voici quelques exemple de syntaxe sous Scilab.

➢ **Découverte :** taper les instructions suivantes dans la console et bien comprendre les résultats obtenus.

```
--> x=2;y=0
-->(y>=-4) & (x<=3)      //ET logique
--> x | y                 //OU logique
--> ~x                   //NON logique
--> y==4                  //égalité
--> y~=5                  //différence
```

☞ *On rappelle que* ***toute valeur non nulle est considérée en logique comme vraie.***

1.3. L'instruction while

Syntaxe :
```
while condition
  instructions
end
```

➢ **Exercice2 :** taper les instructions ci-dessous dans **un fichier script.**
Mettre une copie d'écran d'une exécution dans votre compte-rendu. Puis expliquer l'action de ce programme.

```
clear
x=input('Donner un nombre positif supérieur à 1 et inférier à un million, x = ')
Max=10^6;
n=1;
while (x^n <= Max)
n=n+1;
end
disp(n)
disp(Max)
disp('La valeur de '+string(x)+'^'+string(n)+' est '+string(x^(n)))
```

a. Expliquer l'action de ce programme

b. Adapter ce programme pour qu'à un réel ***x*** compris strictement entre -1 et 1, qui sera donné en entrée, on puisse donner en sortie le nombre entier minimal ***n*** pour que x^n soit proche de 0 à epsilon près (on prendra *epsilon* $=10^{-6}$).
Mettez dans votre compte-rendu un test d'exécution pour x=0.5 et x=-0.3

☞ *Indication* : on pourra utiliser le test $|x^n| \leq \varepsilon$ qui est équivalent à $-\varepsilon \leq x^n \leq \varepsilon$

1.4. L'instruction if

Syntaxe :
```
if condition then
     instruction I1
else
     instruction I2
end
```

Remarque : Des tests successifs sont possibles en utilisant **`elseif`**

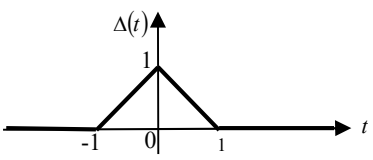
➢ **Exercice3 :** Taper les instructions ci-dessous dans **un fichier script**.

```
clear
close
x=-3:0.01:3
for i=1:length(x)
    if (x(i)>=0)
        y(i)=1
    else
        y(i)=0
    end          //fin du if
end              //fin du for
plot2d(x,y,5)
```

a. Mettre une copie d'écran de l'exécution dans votre compte-rendu puis expliquer l'action de ce programme.

b. Simplifier le programme (qui ne fera alors que 2 lignes) en utilisant la fonction `bool2s`. Taper `help bool2s` dans la console pour avoir l'aide en ligne.

c. Adapter le programme (avec `bool2s`) pour tracer le graphe du signal suivant sur $[-3;3]$:



2. Exercices de synthèse à faire avec l'aide de scilab

➢ **Exercice4 : Suite de Fibonacci**

Les termes de la suite de Fibonacci (u_n) vérifient l'équation récurrente suivante :

$$u_{n+2} = u_{n+1} + u_n \quad \text{avec pour démarrer } u_0 = 1 \text{ et } u_1 = 1$$

a. Déterminer les 15 premiers termes de la suite de Fibonacci (que l'on pourra stocker dans un tableau U).

☞ *Attention* : *sous scilab, un vecteur-tableau ne commence qu'à l'indice 1 donc* U(1)=u_0

b. Déterminer les 15 premières valeurs des quotients de 2 termes consécutifs $\frac{u_{n+1}}{u_n}$ de la suite.

Représenter ces éléments sur un graphe. Qu'observe-t-on ?

☞ *Indication* : dans la fonction `plot2d` option `style=[-1]` permet de tracer un nuage de point.

c. On peut montrer que le quotient de 2 termes consécutifs de cette suite de Fibonacci tend vers le « **nombre d'or** » qui est défini par $\varphi = \frac{1+\sqrt{5}}{2}$. On a donc $\lim_{n\to+\infty}\left(\frac{u_{n+1}}{u_n}\right) = \varphi$.

Écrire un script qui permet de déterminer la première valeur $n_{\min}$ à partir de laquelle pour $n \geq n_{\min}$, on a $\frac{u_{n+1}}{u_n}$ proche du nombre d'or à 10^{-3} près.

➢ **Exercice5 :** Créer une matrice (un tableau) à 20 lignes et 20 colonnes telle que les éléments sur la diagonale soient égaux à 1, ceux au-dessus de la diagonale égaux à 2 et ceux au-dessous de la diagonale égaux à 3.

➢ **Exercice6 :** Le taux du Livret Jeune est de 1,25% nets d'impôts et de prélèvements sociaux. En déposant 1000 euros sur un Livret Jeune, combien d'années faut-il pour augmenter ce capital de 100 euros ?

3. Calculs exacts de primitive ou d'intégrales avec xcas

Pour calculer une intégrale, un logiciel de calcul formel comme Xcas, recherche une primitive ici avec la fonction `int` (ou `integrate` ou `integrer`) puis l'évalue entre les bornes, afin d'obtenir une valeur exacte.

- `int(expression)` : donne une primitive de `expression` par rapport à x ;
- `int(expression,x,a,b)` : donne la valeur exacte de l'intégrale $\int_a^b \text{expression}\ dx$;

➢ **Exercice7 :** en vous aidant de xcas, déterminer les primitives de l'exercice 1 du TD3 de Ma2.