1 TP1

1.1 Quelques rappels

Commentaires : '//' pour le début, jusqu'à la fin de ligne (comme en C++)

1.1.1 Fonctions logiques

`False` et `True` sont représentés par 2 variables internes `%f` et `%t`, on peut les assigner à des noms de variables plus explicite, par exemple :

```
vrai=%t;
faux=%f;
```

On peut ensuite les utiliser dans des fonctions :

```
if ( var == vrai) then  <action si Vrai>
  else
    <action si Faux>
end;
```

Les fonctions booléennes (`and`, `or`, `&` ou `|`...) peuvent être utilisés.

1.1.2 Opérations vectorielles

Un vecteur dans Scilab se définit comme :

```
v1=[2, 1, 4]; // pas besoin de typer
```

On peut extraire la valeur d'un élément du vecteur, ici `v1(1) -> 2`.
Une matrice est définie comme un multi-vecteur :

```
m1=[2, 0, 3 ; 4, 5, 6];
```

1.1.3 Affichage d'une courbe

Pour afficher une courbe avec Scilab, on peut utiliser 2 procédés :
— directement par des tableaux de valeurs, par exemple

```
//définition des coordonnées x, de 0 à 20 par pas de 0,1
x=[0:0.1:20];

//Effacement de la fenêtre de sortie
clf();

//calcul de y0
y0=30*x;
//affichage, ligne continue noire
plot2d(x, y0, style=1);
```

— par définition d'une fonction :

```
//coefficients pour la fonction y=f(x)
A=100;
B=0.001;
m=0.01;

//définition d'une fonction y=f(x)
deff("[y]=f(x)","y=A*cos(x)*exp((-B*x)/(2*m))")

//intervalle de valeurs pour x
x=[0:0.1:20];

//affichage de la fonction f
fplot2d(x,f);
```

À faire :

1. dans une même fenêtre, afficher les courbes $y_0 = 30x$, $y_1 = x^2$ (opérateur `.^`) et $y_3 = A * cos(x) * e^{\frac{-Bx}{2m}}$ (opérateur `exp()`), pour $x \in [0, 20]$.
2. dans une nouvelle fenêtre, afficher la courbe polaire [1] $p = e^{cos(\theta)} - 2cos(4\theta) + sin^5(\theta/12)$ pour $\theta \in [0, 2\pi]$. Utilisez la documentation Scilab pour `polarplot()`. Attention à ne pas utiliser un pas trop élevé, si vous prenez sur l'angle θ vous n'aurez que 6 valeurs ($\pi = 3.14...$).

Notes : `clf()` permet d'effacer la fenêtre courante, `scf()` permet de créer une nouvelle fenêtre graphique.

1.2 À faire

1. appartenance d'un point à un plan dans l'espace $\mathbb{R}^3$: écrire une fonction qui renvoie un booléen selon l'appartenance d'un point P à un plan Π, tous deux passés en paramètre. Comme une équation du plan est : $a_0x + a_1y + a_2z + a_3 = 0$, on peut stocker les coefficients $a_0...a_3$ dans un vecteur.

```
//x y z
point=[2, 1, 0];

//a0 X +a1 Y + a2 Z + a3 = 0
plan=[1,  2,  3,  4];
```

Du coup on peux utiliser une relation vectorielle : l'appartenance d'un point à un plan revient à un produit scalaire ou à une multiplication du vecteur *plan* par le vecteur *point*. Sauf que :

— ces deux vecteurs ne sont pas de même dimension. On crée donc un vecteur à partir de point mais de dimension 4 :

```
point4 = [point, 1];
```

On utilisera ensuite les coordonnées homogènes mais c'est un bon début.

— la multiplication se fait entre un vecteur ligne et un vecteur colonne. Il faut transposer le deuxième (en utilisant la quote `'`) :

```
resultat = plan * point4';
```

Autant écrire une fonction qui prend un vecteur de dimension 3 pour un point et de dimension 4 pour un plan et qui retourne un booléen pour l'appartenance :

```
function b=appartenancePointPlan(Pt, Pl)
  ...
  ...
    if (resultat == 0) then
        b = vrai;
    else
        b = faux;
    end

endfunction;
```

2. appartenance d'un point à une droite dans l'espace $\mathbb{R}^2$: écrire une nouvelle fonction qui renvoie un booléen pour indiquer l'appartenance d'un point à une droite cartésienne, de la forme $a_0x + a_1y + a_2 = 0$. On peut utiliser `x = input("entrez la valeur : ")` pour lire une valeur dans la console, celle-ci peut être formatée.
3. utiliser l'exercice précédent pour tirer au hasard (`rand()`) des positions dans l'intervalle $x, y \in [0, 1]$ et vérifier l'appartenance de ces points à une droite. Vous pouvez faire une sortie graphique de la droite et des tirages successifs, par exemple pour les points :

1. Papillon de T. Fay, cf `http://www.mathcurve.com/courbes2d/ornementales/ornementales.shtml`

```
xpoly([],[],"marks"); //xpoly() comporte un champ .data qui est le vecteur des coordonnées des
    sommets du polygone
prand=gce();

pas=1;
cpt=0;

for i = 0:pas:1000
//il faut ici tirer un point(x,y) au hasard
    if (appartenancePointDroite(point, droite) == vrai) then
            cpt=cpt+1;
    end;
    prand.data=[prand.data; point];
end;
```

1.3 Liens

— WikiLivre Scilab `https://fr.wikibooks.org/wiki/Découvrir_Scilab`
— aide officielle (version 5.5.1) `https://help.scilab.org/docs/5.5.1/fr_FR/index.html`
— il existe aussi une page sur `developpez.com` `http://scilab.developpez.com/tutoriels/introduction-scilab/`
— une page sur le site officielle sur l'utilisation de scilab `http://www.scilab.org/community/education/maths/doc`