TP I **INITIATION au logiciel MATLAB**

Ce TP a pour but d'apprendre à utiliser le logiciel Matlab afin de pouvoir développer des applications simples en traitement du signal.

***Créer d'abord un répertoire portant votre nom* sous la directorie work pour y développer vos programmes :** >>mkdir « votre nom » **puis** >>cd « votre nom »**.**

I- Manipulation des variables

On distingue les variables scalaires et les variables vectorielles (matricielles en général).

1- Variables scalaires

Dans un premier temps on génère trois variables scalaires a, b et c de la manière suivante :

```
>> a=2 ;
>> b=3;
>> c=4 ;
```

On peut consulter la valeur d'une variable en entrant son nom :

>> a

La réponse serait :

```
a=
     2
```

Cela signifie que les valeurs des variables sont mémorisées automatiquement avec leurs noms.

2- Taille des variables dans la mémoire

La taille d'un scalaire de type « double » est **8 Bytes = 8 Octets = 64 Bits**

3- Commandes de base :

- **who** et **whos** :

  Affiche la *taille mémoire* et *type*s de toutes les variables utilisées.

- **cd** :

  Affiche le répertoire (directorie) où vous opérez en ce moment.

- **what, dir** :

  Affiche la liste les *noms* des fichiers contenus dans le répertoire actuel.

- **help *nom_fonction*** :

  Donne un descriptif de la fonction et ses arguments d'entrée sortie.

4- Opérations sur les variables

```
>>d=a+b+c
>>e=a+b*c
>>f=(a+b)*c
```

```
>>g=(a/b)*c
>>h=a^2
```

5- Les matrices et les variables vectorielles

Matlab est optimisé pour l'usage matriciel : Eviter les formulations non matricielles.

- **[a b c]** est un vecteur ligne.
- **[a ;b ;c ;]** est un vecteur colonne.
- **V'** est le transposé du vecteur **V**
- **u=1:5** est le vecteur [1 2 3 4 5] (de même que [1 :5] et (1 :5).
- **t=0 :2 :15** est le vecteur [0 2 4 6 8 10 12 14]
- **sin(t)** est le vecteur [sin(0) sin(2) …sin(14)]
- zeros(1,N) est le vecteur ligne nul à N éléments.
- Ones(1,N) est le vecteur ligne à N éléments égaux à 1.

6- Opérations sur les vecteurs

- Il faut respecter les dimensions des vecteurs et matrices.

Génération automatique d'un vecteur

V=début:pas:fin;

- On définit une valeur de début : *début*
- On définit une valeur de fin : *fin*
- On définit un pas de progression linéaire ou logarithmique (incrémentation) : *pas ;*
- si le pas n'est pas spécifié, il est égal à 1 automatiquement.

```
>>v3=1:10
>>v4=1:-0.5:-1
>>debut=0;fin=256;pas=8;
>>v5=debut:pas:fin

>>debut=0;fin=2*pi;pas=0.1;      où pi désigne le nombre 3.14…
>>v5=debut:pas:fin
```

Une liste des fonctions les plus courantes est disponible dans l'aide en ligne en tapant la commande (elfun désigne elementary functions : sin, cos, log, exp…)

>>**help elfun**

7- Les matrices

Saisir la matrice 3x3 suivante :

a=[1 2 3 ;4 5 6 ;7 8 9]

Noter que deux lignes sont séparées par un point virgule.

Si vous entrez la commande **a(:)** vous obtenez le vecteur *colonne* **[1 4 7 2 5 8 3 6 9]'**.

La fonction **eig** donne les valeurs propres de la matrice a.

zeros(N) est *la matrice* nulle NxN, **eye(N)** est la matrice identité NxN.

II- ROGRAMMATION

1-LES SCRIPTS

Plutôt que de taper les commandes au clavier les unes après les autres pour effectuer une tâche, ce qui vous oblige à refaire la même chose à chaque utilisation de cette tâche, il est préférable de grouper les commandes dans un fichier à extension **.m**, ainsi tous les programmes auront pour nom ***name.m***. Il suffit alors de taper ***name*** pour que la tâche s'exécute.

Exemple 1 :

Créez un fichier qui s'appelle essai1.m, qui génère un signal sinusoïdal x(t) de n points, puis visualisez le à l'aide de la commande **plot**(x).

- Etape 1 : édition du fichier par la commande : edit essai1.m
- Etape 2 : taper les commandes suivantes dans la fenêtre d'édition :

%ce fichier génère et affiche une sinusoïde

t=0:0.1:2*pi; % t est le vecteur temps avec un pas d'échantillonnage 0.1
x=sin(t); % le signal x est un vecteur de même taille que le vecteur t

plot(t,x) ;grid on % dessine à l'écran x en fonction de t

Remarques :

1- Le texte débutant par % est un commentaire de votre choix, il est ignoré par le logiciel. Vous pouvez supprimer ces commentaires, mais ils sont utiles lorsqu'on a plusieurs programmes.

2- Chaque instruction doit être suivie d'un point virgule.

- Etape 3 : Sauvegardez le fichier dans le répertoire en cours ;
- Etape 4 : Exécutez le programme en entrant la commande suivante :

>>**essai1**

2- Utilitaires graphiques

- L'instruction **title('titre de la courbe ou la figure')** ajoute un titre à la figure visualisée.
- **xlabel('titre des abscisses')** affiche un titre horizontal suivant x.
- **ylabel('titre des ordonnés')** affiche un titre vertical suivant y.
- **grid on** et **grid off** quadrille ou non le graphique.

Exemple 2 :

Générez,à l'aide d'un programme **prog1.m**, deux périodes des deux signaux x(t)=**cos(t)** et y(t)=**sin(t)** et visualisez les deux signaux sur une même figure, l'une en rouge et l'autre en bleu à l'aide de la fonction **plot(t,x,'b',t,y,'r').** Mettre une légende.

Exemple 3 :

Visualisez sur une *même figure*,à l'aide d'un programme **prog2.m**, les quatre signaux **cos(t), sin(t),log10(t)** et **exp(t)** en utilisant la fonction **subplot(.,.,.)** qui divise l'écran en quatre sous figures : (2,2,1),(2,2,2),(2,2,3) et (2,2,4).

>>**subplot(2,2,1),plot(t,x);**

Saisie d'une donnée au clavier :

Pour saisir une variable x à partir du clavier, on utilise l'instruction :

x=input('x=') ;

Affichage d'un message à l'écran :

Pour afficher à l'écran un message personnel suivi d'un retour à la ligne :

fprintf('*message personnel* **\n') ;**

Affichage de la valeur d'une variable :

Pour afficher à l'écran la valeur d'une variable x :

fprintf('x=**'%d) ;**

III- LES FONCTIONS

Si plusieurs de vos programmes personnels utilisent en commun une liste d'instructions, il est préférable de regrouper ces instructions sous forme d'un programme indépendant. A chaque besoin on appelle le dit programme par son nom : c'est une fonction.

Une fonction possède des paramètres d'entrée et des paramètres de sortie, dont la syntaxe de déclaration est la suivante :

function [sortie1,sortie2,…]=nom_fonction(entrée1,entrée2,…)

Le programme matlab correspondant à la fonction doit porter le même nom que la fonction : **nom_fonction.m**

Exemple 1 :

Calcul de la moyenne arithmétique m d'un vecteur v de dimension n :

$$m=\frac{1}{n}\sum_{i=1}^{n} v(i)$$

La fonction **moyenne.m** comportera les instructions suivantes :

```
function resultat=moyenne(v,n);        %déclaration de la fonction

r=0;                                   %initialisation de la moyenne
for
   k=1:n      r=r+v(k);
end
   resultat=r/n;
```

On note la présence de la boucle itérative :

for indice=debut:pas:fin

instruction **;**

end

Qui est équivalente à :

indice=debut ;

(**B**) instruction;

debut=debut+pas;

si debut≤fin aller à **B**;

Mise en œuvre de la fonction

>>moyenne(1:100,100)

Comparer avec la fonction matlab **mean.**

Exemple 2 :

Créez une fonction **puissance.m** qui donne la valeur efficace **s** et la puissance moyenne **p** d'un signal.

x étant un vecteur de composantes x(i) , i=1..n, la valeur efficace de x est :

$$s=\sqrt{p}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}x^2(i)}$$

Testez cette fonction sur un signal sinusoïdal d'amplitude A et de période 1.

Comparer avec la fonction matlab **std.**

Exemple 3 : Diagramme de bode.

1- Créez une fonction **spectre(f,H)** qui affiche à l'écran le spectre d'amplitude **20log|H(f)|** et le spectre de phase **Arg(H(f))** en fonction de **f** en Hertz avec indication du titre de chaque courbe ; Les paramètres d'entrée sont le vecteur fréquence **f=0:f$_{max}$** et le vecteur complexe **H**.

Fonctions conseillées :**abs**(x),**angle**(x) et **log10**(x).

2- Utilisez la fonction **spectre** dans un programme indépendant **myfilter.m** pour tracer les diagrammes de Bode des filtres passe bas et passe haut dont les fonctions de transferts sont définies par :

$$H_1(f)=\frac{1}{1+j.\frac{f}{f_c}}\ ,\ H_2(f)=\frac{j.\frac{f}{f_c}}{1+j.\frac{f}{f_c}}$$

f$_c$ étant la fréquence de coupure demandée par le programme et fournie par l'utilisateur à l'exécution.

Exemple 4 **: vitesse de convergence d'une série**

Considérons la série en N suivante (qui converge vers exponentiel de x) :

$$s(x,N)=\sum_{k=0}^{N}\frac{x^k}{k!}$$

Réaliser :

- Une fonction **factoriel(n)** qui calcule le factoriel d'un entier naturel.
- Une fonction **somme(x,n)** qui calcule s(x,n).
- Un programme **prog3.m** qui calcule la valeur de l'exponentiel de 2 avec une précision (incertitude relative) supérieure à 95% (incertitude inférieur à 5%).

PARTIE THEORIQUE

1. En utilisant $e^{j.n\alpha}=\left[e^{j.\alpha}\right]^n$, Trouver les expressions de cos(nα) et sin(nα) pour n=2 et n=3.
2. De même retrouver les expressions de cos(α+β) et sin(α+β).
3. Calculer la somme $\sum_{n=1}^{\infty}s(n)$, avec s(n)=$\frac{1}{2^n}+j\frac{1}{3^n}$
4. Même question pour s(n)=$(\frac{j}{3})^n$
5. Trouver les complexes z tels que : $\bar{z}^N=z^{-N}$
6. trouver les racines cubiques de 1.
7. Calculer le produit infini $\prod_{n=1}^{\infty}e^{j\pi/2^n}$
8. Calculer $I_n=\int_0^{\pi/2}\sin^n(x)dx$ et $J_n=\int_0^{\pi/2}\cos^n(x)dx$

Le compte rendu de la séance doit être rédigé sur la feuille qui vous est fournie et doit contenir <u>en plus de la partie théorique</u> :

<u>Un listing des programmes</u> :

prog1.m

prog2.m

puissance.m

spectre(f,H)

myfilter.m

factoriel(n)

somme(x,n)

prog3.m