TP N° 1 Initiation à Matlab

Instructions et variables :

Une instruction est de la forme indiquée ci-contre

```
≫ variable = expression
```

Son exécution vient après le retour chariot (↵). Le signe « = » est utilisé pour affecter à la variable l'expression. Un exemple d'instruction est donné ci-dessous.

```
≫ A = [1   3; 5   6] (↵)
   A =
        1   3
        5   6
```

La matrice A est automatiquement affichée après l'exécution de l'instruction. Si l'instruction est suivie d'un point-virgule (;) le résultat ne s'affichera pas. Les opérations arithmétiques classiques peuvent être utilisées dans les expressions intervenant dans les instructions. Matlab peut aussi être utilisé en mode calculatrice, comme dans l'exemple ci-dessous.

```
≫ 15/3
ans = 5
```

Quand le nom de variable et « = » sont omis le résultat est affecté à la variable générique *ans*, Matlab possède la plupart des fonctions trigonométrique et mathématiques élémentaires. Les plus utiles sont résumées dans le tableau si dessous.

$\sin(x)$	Sinus des éléments de x	sqrt(x)	Racine carrée des éléments de x
cos(x)	Cosinus des éléments de x	imag(x)	Partie imaginaire de x
Asin(x)	Arcsinus des éléments de x	real(x)	Partie réelle de x
Acos(x)	Arccosinus des élements de x	conj(x)	Conjugé complexe de x
tan(x)	Tangeante des éléments de x	log(x)	Logarithme népérien des éléments de x
Atan(x)	Arctangente des éléments de x	log10(x)	Logarithme décimal des éléments de x
Abs(x)	Valeur absolue des éléments de x	exp(x)	Exponentielle des éléments de x

Les noms des variables commencent par des lettres et sont suivie par n'importe quel nombre de lettre et de nombres. N'utilisez par plus de 19 caractères pour nommer une variable, car Matlab ne prend en compte que les 19 premiers caractères. Une bonne pratique consiste à utiliser des noms de variables « parlants », permettant de reconnaitre rapidement ce que représente une variable (par exemple vit pour vitesse). Matlab fait aussi la différence entre majuscule et minuscule (les variables V et v ne sont par les mêmes). Matlab possède aussi des variables prédéfinie pi, Inf, Nan, i, elle représente $\pi, \infty,$

nombre indéfini (not a numbre), et le nombre imaginaire $\sqrt{-1}$. Ces variables prédéfinie peuvent être écrasées volontairement ou par accident. Les variables prédéfinies peuvent retrouver leur valeur par défaut en utilisant **clear** *nom de variable* (ex, clear i).

Les variables sont stockées dans le workspace, elles le sont automatiquement pour être utilisées par la suite dans la même session.

La fonction **who** donne la liste des variables dans le workspace, par exemple :

```
≫ z = 5 + 1
≫ M = [1   2]
≫ m = [1   0   3]
≫ A = [1   2; 3   5]
   A =
        1   2
        3   5
≫ who
your variable are :
A     M     m     z
```

La fonction **whos** donne la liste des variables dans le workspace, ainsi que des informations supplémentaires sur la dimension, le type de variable, et la mémoire allouée.

```
≫ whos
```

Name	Size	Elements	Bytes	Density	Complex
A	2 by 2	4	32	Full	No
M	1 by 2	2	16	Full	No
m	1 by 3	3	24	Full	No
z	1 by 1	1	16	Full	Yes

Grand total is 10 elements using 88 bytes

On peut supprimer une variable du workspace en utilisant la fonction **clear**, clear toute seul supprime toutes les variables, alors que clear suivi du nom des variables ne supprime que les variables dont les noms sont indiqués

Tous les calculs sous Matlab sont faits en double précision, cependant l'affichage sur l'écran peut se faire sous différents formats. Le format d'affichage peut être modifié à l'aide de la fonction **format**.

Exemple :

```
≫ pi
ans =
    3.1416
≫ format long ; pi
ans =
    3.14159265358979
≫ format short e; pi
ans =
    3.1416e + 000
≫ format long e; pi
ans =
    3.141592653589793e + 000
```

Matrices :

Matlab est contraction de *matrix laboratory*, c'est un programme interactif très performant pour le calcul matriciel. L'unité de base est la matrice, les vecteurs et les scalaires sont des cas spéciaux de matrices. Les opérations matricielles de base sont : l'addition, la soustraction, le produit, la transposition, la puissance et les opérations dites de tableau qui sont des opérations élément à élément. Il n'est pas nécessaire de définir la dimension ou le type d'une matrice sous Matlab, la mémoire est automatiquement allouée. Les opérations matricielles nécessitent évidemment la comptabilité des dimensions.

Exemple d'opérations matricielles :

```
≫ A = [1   3; 5   9] ; B = [4   −7; 10   0] ;
≫ A + B
ans =
     5   −4
    15    9
≫ b = [1; 5];
≫ A ∗ b
ans =
    16
    50
≫ A′
ans =
     1   5
     3   9
```

Les opérations de type tableau nécessitent de faire précéder l'étoile d'un point (.*).

Exemple d'opération de tableau ci-contre :

```
>> A = [1; 2; 3]; B = [−6; 7; 10];
>> A.* B
ans =
     −6
     14
     30
>> A.^2
ans =
      1
      4
      9
```

On peut générer un vecteur à l'aide de Matlab en spécifiant la valeur initiale, la valeur finale et l'incrément : $x = [x_i : dx : x_f]$

Exemple : supposons que l'on veuille tracer la fonction $y = x\cos(x)$ en fonction de x, pour $x = 0{,}0.1, 0.2, \ldots, 1.0$. La premiére étape est de générer une table de données (x, y) ceci est illustré ci-contre :

```
>> x = [0: 0.1: 1]'; y = x.* cos(x) ;
>> [x  y]
ans =
         0    0
    0.1000    0.0995
    0.2000    0.1960
    0.3000    0.2866
    0.4000    0.3684
    0.5000    0.4388
    0.6000    0.4952
```

Fonctions :

Matlab présente un certain nombre de fonctions prédéfinies. Par exemple, si x est un vecteur, on peut trouver dans une variable y la somme de ses éléments à l'aide de la fonction **sum**

```
>> x = [1   2   3]
>> y = sum(x)
y =
   6
```

La fonction **fzero** est utilisée pour déterminer le zéro d'une fonction étant donné un point initial.

```
>> fzero('sin', 3)
ans =
    3.1416
```

Matlab donne la possibilité de définir d'autres fonctions. Pour ce fait, il faut tout d'abord créer un fichier qu'on appelera par exemple 'fact' et qui permettra de déterminer la factoriel d'un nombre donné. Les instructions du fichier sont comme suite :

```
%fichier fact.m
function m = fact(n)
if n == 0
        m = 1; % parceque 0! = 1
else
    m = 1;
    for i = 1: n
         m = m * i;
    end;
end
```

Le % est utilisé pour écrire un commentaire qui n'est pas pris en considération lors de l'exécution du fichier.
Dans l'espace de travail on a par exemple

```
>> fact(5)
ans =
     120
```

Graphisme :
Le graphisme joue un rôle important dans l'analyse et la synthèse des systèmes asservis. La résolution d'un problème de commande nécessite souvent l'utilisation d'une multitude de types de données. Matlab permet la visualisation du tracé de Bode, du lieu des racines, du tracé de Nyquist etc…

L'objectif de cette partie est de se familiariser avec les possibilités graphiques de Matlab. Matlab utilise une fenêtre pour visualiser les graphes. Le tracé standard (x,y) est réalisé à l'aide de la fonction **plot**. Les données générées ci-dessus peuvent être représentées à l'aide de **plot(x,y)** :

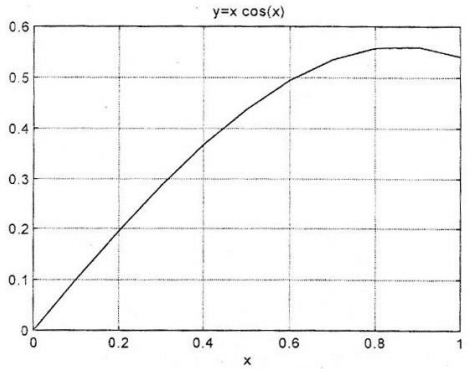


Figure 1.

En cliquant n fois sur une courbe la fonction **ginput(n)** donne les coordonnées des points sélectionnés.

Fonctions plot disponibles :

plot(x,y)	Trace le vecteur u en fonction du vecteur x
semilogx(x,y)	Trace le vecteur y en fonction du vecteur x l'axe x est logarithmique, et l'axe y est linéaire
semilogy(x,y)	Trace le vecteur y en fonction du vecteur x l'axe x est linéaire et l'axe y logarithmique
lolog(x,y)	Trace le vecteur y en fonction du vecteur x les deux axes étant logarithmiques

Il existe des possibilités pour améliorer la qualité du graphe, tels que mettre les légendes, mettre une grille…, fonctions disponibles pour améliorer un graphe :

title ('texte')	Place la chaîne de caractères 'texte' en haut du graphe
xlabel('texte')	Place la chaîne de caractères 'texte' sur l'axe de x
ylabel('texte')	Place la chaîne de caractères 'texte' sur l'axe de y
text(p1, p2, 'texte', 'sc')	Place la chaîne de caractère 'texte' au point de coordonnées (p1,p2) où (0,0) est l'extrémité gauche et (1,1) l'extrémité droite de l'écran.
subplot	Subdivise la fenêtre graphique
Grid	Place la grille sur le dessin courant

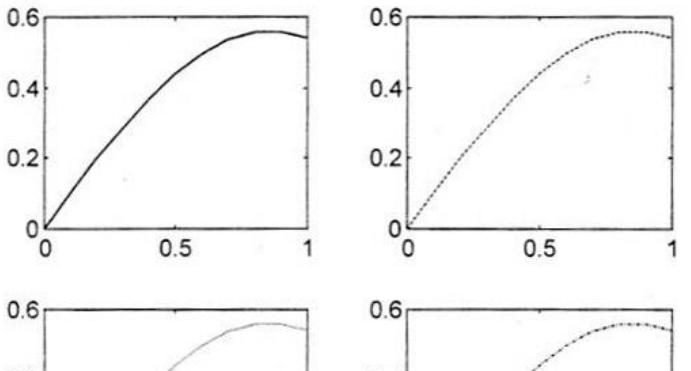
Le graphe de la figure 1 a été obtenu à l'aide des instructions suivantes :

```
≫ x = [0: 0.1: 1]'; y = x.* cos(x) ;
≫ plot(x, y)
≫ xlabel('x')
≫ ylabel('y')
≫ title('y = x cos(x)')
≫ grid
```

On peut aussi fixer le type de trait du dessin à l'aide de **plot(x,y, 'type de trait')**, ceux disponible sont : '-' ligne continue, '--' ligne discontinue, ' :' ligne pointillé,'-.' ligne discontinue pointillée
La commande **subplot(mnp)** permet de subdiviser la fenêtre graphique en m x n fenêtres, l'entier p indique où la fenêtres sont numérotées.

Par exemple en reprenant l'exemple de la fonction $y = x\cos(x)$ (représenté 4 fois) on obtient la figure 2 à l'aide des instructions suivante :

```
≫ x = [0: .1: 1]; y = x.* cos(x) ;
≫subplot(221),plot(x,y,' -' )
≫subplot(222),plot(x,y,'--')
≫subplot(223),plot(x,y,':')
≫subplot(224),plot(x,y,'-.')
```



Supposant que l'on veuille tracer la fonction $y(t) = \sin(at)$ où a est une variable. A l'aide d'un éditeur de texte nous écrivons un fichier qu'on appellera **plotdata.m**. on introduit la valeur de a au prompt de Matlab, ainsi a se trouvera dans le workspace, le fichier utilisera la valeur la plus récente de a.

```
% plotdata. m
% ceci est un fichier pour tracer la fonction y = sin(a * t)
%
% la valeur de a doit exister dans le workspace avant
% d'appeler le fichier
%
t = [0: 0.01: 1];
y = sin(a * t) ;
plot (t, y)
xlabel('temps[sec]')
ylabel(y(t) = sin(a * t)')
grid
```

Simulink

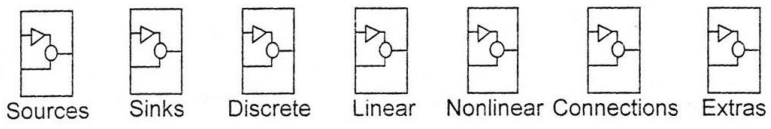
Simulink est un programme pour la simulation temporelle des systèmes dynamiques. Il se concrétise en deux étapes : la première est une étape d'édition ; et la deuxième est une étape d'analyse du modèle.

Pour faciliter l'utilisation de ce logiciel, les concepteurs ont choisi d'utiliser des icônes pour l'implémentation des différentes fonctions. Le modèle ainsi réalisé est visuellement compréhensible. Il y a deux moyens d'analyser les systèmes. Le premier consiste à simuler le modèle en fonction de ses différents paramètres dans la fenêtre d'édition, la sortie équivalente est alors, un graphe, une courbe équivalente à celle d'un oscilloscope. La deuxième analyse serait une approche mathématique du système, en récupérant les équations d'état du modèle défini sous Simulink vers Matlab.

- **Démarrage d'une session Simulink**

Pour ce fait, dans la fenêtre Matlab derrière le prompt, taper simulink. Le programme Simulink va alors s'installer en ouvrant une fenêtre intitulée Simulink, composée de plusieurs icônes

❖ **Identification des boîtes de Simulink**



Source permet d'apporter sur le schéma toutes sortes d'entrées temporelles que pourrait subir un système.

Sinks permet l'ouverture d'une fenêtre contenant les icônes de visualisation simple et d'enregistrement des signaux du processus.

Discret ouvre une fenêtre correspondant aux différentes équations des transformées en Z. elle utilise donc des variables discrètes.

Linear est équivalente à la précédente mais pour les systèmes continus.

Nonlinéaire contient toutes sortes de fonctions telles que multiplicateur, fonction Matlab, Hystérésis, relais, retard pur,…

Connections permet d'apporter au schéma des entrées-sorties et des multiplexeurs-démultiplexeur.

Extracts contient des librairies supplémentaires et plus spécialisées de filtres, de systèmes linéaires et non linéaire, des méthodes d'identification, des générateurs…

❖ **Edition du schéma à simuler**

Pour pouvoir débuter la session d'édition, il faut ouvrir une nouvelle fenêtre pour implanter le système à simuler. On ouvre alors le menu déroulant de « File » et on clique sur « New ». une nouvelle feuille d'édition s'ouvre. Il faudra préalablement, pour éviter de la perdre, lui donner un titre. Cela se fait dans le menu « Fichier » avec l'option « Enregistrer » ou « Enregistrer sous… ». la feuille d'édition sera alors nommée.

Pour l'édition d'un schéma donné, il faut insérer les icônes lui correspondant. Pour cela on ouvre les fenêtres des icônes désirées avec un double clic, on sélectionne l'icône en cliquant dessus, puis on la déplace sur la feuille d'édition en gardant le bouton gauche de la souris enfoncé jusqu'à l'endroit désiré. Un double clic sur l'icône permet d'ouvrir la fenêtre de ses paramètres pour pouvoir en changer les valeurs.

Une fois que toutes les icônes sont placées, on effectue la mise en place des fils, connexions, entre les différentes boîtes. Pour cela, il suffit de cliquer sur l'entrée ou la sortie d'une boîte, puis sans relâcher le bouton de la souris, allez sur l'extrémité d'une autre boîte.

Pour l'analyse temporelle des systèmes, il existe des icônes correspondant à des échelons, des rampes et des oscilloscopes. Avant de lancer la simulation il faut régler les différents paramètres mises à la disposition de l'utilisateur dans le menu « Simulation » puis « Paramètres ». Ils se composent du temps de départ et de fin de la visualisation sur les Scopes, du pas mini et maxi de la méthode d'intégration du système et de la tolérance d'erreur permise dans les calculs de la méthodes. Une fois cela bien établi, on lance la simulation par « Simulation » puis « Start ».

❖ **Visualisation des courbes sur la figure de Matlab**

Généralement, les figures ne sont pas bien visibles sur les scopes et aussi la lecture des coordonnées de certains points ne se fait pas avec une grande précision. Pour ce fait on procède à la visualisation des courbes obtenues par simulation sur des figures par le biais d'instruction de Matlab.

Dans la feuille d'édition d'un schéma donné, on remplace le scope par le bloc « To File » de la librairie « Sinks ». on modifie par la suite des paramêtres.

- Donner un nom de la forme Nom_fiche.mat au fichier contenant les points de simulation ;
- Mettre un nom à la place de la variable *ans* et ce sera la matrice dont chaque ligne représente une composante (temps, amplitude,…), soit par exemple x.

Il faut revenir par la suite à Matlab dans lequel on crée un fichier .m contenant les instructions suivantes :

```
% Charger le fichier des données
load Nom_fiche.mat

% le temps est la 1ère ligne de la matrice x
t = x(1,:)

% l'amplitude est la 2ème ligne de la matrice x
y = x(2,:)

%faire le tracé apparaissant sur le scope
plot(t,y)
```

Travail demandé

Un système masse-ressort est décrit par l'équation différentielle suivante :

$$M\ddot{y} + f\dot{y} + ky = r(t)$$

Ce modèle est linéaire invariant, il n'est valable que pour de petits déplacement. M est la masse, f est le coefficient de frottement et k est la raideur du ressort. Ce système pourrait représenter un amortisseur pour une voiture. L'objectif pourrait être la synthèse d'un système de commande pour rendre la conduite plus souple quand le véhicule travers une route non uniforme.

La réponse du système pour une condition initiale donnée est exprimée par :

$$y(t) = \frac{y(0)}{\sqrt{1-\xi^2}} e^{-\xi\omega_n t}\sin(\omega_n\sqrt{1-\xi^2}t + \arcsin(\xi))$$

La réponse transitoire est sous amortie si $\xi < 1$, suramortie pour $\xi > 1$ et critique pour $\xi = 1$

Ecrire un fichier Matlab permettant de représenter la réponse du système masse-ressort pour les cas suivant :

$$- y(t) = 0.15 m_t \omega_n = \sqrt{2}\, rd/s, \xi = \frac{3}{2\sqrt{2}}$$

$$- \; y(0) = 0.15 m_t \omega_n = \sqrt{2}\, rd/s, \xi = \frac{1}{2\sqrt{2}}$$