École Polytechnique de Montréal
Département de génie électrique et génie informatique
Section télécommunications et micro-ondes

Initiation à MATLAB

Les travaux pratiques qui vous seront soumis au cours du trimestre sont planifiés en fonction de l'utilisation du logiciel MATLAB. C'est pourquoi nous vous proposons ici une séance d'initiation à ce logiciel. Ce fascicule comporte principalement :

- un texte d'introduction qui vous explique l'utilisation des instructions MATLAB les plus pertinentes pour le cours 3.463 ;
- une série d'exercices suggérés.

Nous vous conseillons fortement de lire le texte d'introduction avant la séance prévue au laboratoire. Vous aurez ainsi le temps d'effectuer les exercices suggérés durant la séance.

Notez que ce texte d'introduction ne présente MATLAB qu'en fonction de son utilisation pour les travaux pratiques du cours 3.463. Les possibilités et applications de ce logiciel sont beaucoup plus nombreuses. Pour une information plus complète sur MATLAB, consultez le manuel de référence disponible sous la rubrique « help » du logiciel.

Calendrier

Section	***Date***	***Heure***
1	21 janvier	9h30 à 12h20
2	28 janvier	9h30 à 12h20
3	27 janvier	8h30 à 11h20
4	3 février	8h30 à 11h20
5	22 janvier	13h45 à 16h35
6	29 janvier	13h45 à 16h35

Table des matières

1 Introduction à Matlab

MATLAB est un logiciel interactif permettant d'effectuer des calculs numériques complexes particulièrement utiles dans le domaine de l'ingénierie. Disponible sur de gros systèmes, il fut adapté pour l'ordinateur personnel muni d'un coprocesseur mathématique permettant une grande capacité de calcul.

Notez que MATLAB est un logiciel auto-documenté ; *des informations relatives à une instruction en particulier peuvent être obtenues à l'écran en composant « help », suivi du nom de l'instruction.*

1.1 Préliminaires

Le logiciel MATLAB est conçu en fonction de la manipulation de matrices et de vecteurs (un vecteur de longueur N est une matrice de dimension $1 \times N$ comportant une rangée et N colonnes).

On génère une matrice en écrivant ses éléments entre crochets, chaque rangée étant délimitée par un point-virgule. Par exemple, pour générer la matrice

$$\texttt{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

on compose :

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

On obtiendra la transposée et le déterminant de la matrice `A` en composant :

```
B = A.'
C = det(A)
```

Il est possible d'extraire un élément ou un groupe d'éléments d'une matrice ; par exemple :

```
D = A(2,1)
E = A([1,2],[2,3])
F = A(:,2)
G = A(2,:)
```

La matrice `D` contiendra l'élément situé à l'intersection de la 2$^{\text{e}}$ rangée et de la 1$^{\text{re}}$ colonne de `A`. La matrice `E` contiendra l'intersection des rangées 1 et 2 et des colonnes 2 et 3 de `A` :

$$\texttt{E} = \begin{pmatrix} 2 & 3 \\ 5 & 6 \end{pmatrix}$$

La matrice `F` de dimension 3×1 sera formée de la 2$^{\text{e}}$ colonne de `A`, tandis que la matrice `G` de dimension 1×3 sera formée de la 2$^{\text{e}}$ rangée de `A`. Que contiendra la matrice `H` après l'opération suivante :

```
H = A([3,1],:)
```

1.2 Génération de vecteurs

Forme : `A=AI:I:AF` où `AI` est la valeur initiale, `I` est l'incrément et `AF` est la valeur finale. Si `I` est omis, un incrément de 1 est pris par défaut.

Exemples :

`a = 1:.25:2` génère `a` = (1 1.25 1.5 1.75 2)

`b = (1:50)*3` génère `b` = (3 6 9 ... 147 150)

`c = 10:-1:5` génère `c` = (10 9 8 7 6 5)

Un vecteur peut être utilisé comme indice ; par exemple, en utilisant le vecteur `b` défini ci-dessus, l'instruction `d = b(30:-1:20)` génère le vecteur

$$\texttt{d} = (\underbrace{90}_{30^{\text{e}}\text{ elmt de b}} \quad 87 \quad 84 \quad \ldots \quad 63 \quad \underbrace{60}_{30^{\text{e}}\text{ elmt de b}})$$

Il est possible de générer des vecteurs dont la progression des éléments est logarithmique en utilisant l'instruction `logspace` : `e = logspace(K1,K2,N)` produit un vecteur `e` dont les N éléments varient logarithmiquement entre 10^{K1} et 10^{K2}. Lorsque le nombre d'éléments n'est pas spécifié, 50 est pris par défaut.

1.3 Boucles

Forme :

```
for I=x
 .
 .
 .
end
```

où `x` est un vecteur comportant les valeurs que doit prendre la variable I à chaque itération.

Exemple :

```
a=[ ]
for I=1:3
  a=[a 0 0 1 1]
end
```

Après cette boucle, on aura `a` = (0 0 1 1 0 0 1 1 0 0 1 1). Notez de quelle façon le vecteur `a` est construit à l'intérieur de la boucle : la ligne `a=[a 0 0 1 1]` signifie que l'on place dans `a` ce qu'il y avait déjà, suivi des éléments 0 0 1 1. La ligne `a=[ ]` nous assure que le vecteur `a` est bien vide avant l'exécution de la boucle.

1.4 Instructions conditionnelles

Forme :

```
if xxxxxx   condition (voir les informations données à la section 4)
 .
 .          instructions effectuées si
 .          la condition est respectée
 .
else
 .
 .          instructions effectuées si la
 .          condition n'est pas respectée
 .
end
```

(`else` : facultatif)

Exemple :

```
a=[0 -1 4 -6 9 -7]
b=[ ]
c=[ ]
```

```
for I=1:length(a)
  if a(I)<0
    b=[b a(I)]
  else
    c=[c a(I)]
  end
end
```

Après cette boucle on aura `b` $= (-1 \ -6 \ -7)$ et `c` $= (0\ 4\ 9)$. Remarquez ici de quelle façon on vérifie une condition sur un seul élément de `a` à l'intérieur de la boucle.

1.5 Opérations sur les matrices et vecteurs

Notez que des opérations mathématiques peuvent être effectuées sur des matrices ou vecteurs entiers à l'aide d'une seule instruction, sans boucle `for ... end`. Par exemple, soit le vecteur `t` généré par l'instruction `t = 2*pi*[0:100]/100`. On veut évaluer le sinus et le cosinus de chaque élément de `t` : les instructions `r = sin(t)` et `s = cos(t)` nous donnent les vecteurs `r` et `s` contenant les résultats désirés. La somme de `r` et `s`, leur produit point à point et le carré des éléments de `r`, sont obtenus respectivement par les instructions

```
a = r + s
b = r .* s
c = r.^2
```

Le dernier résultat peut également être obtenu par l'instruction

```
c = r .* r
```

Remarques

1. Tous les résultats des calculs effectués sont affichés à l'écran ; en ajoutant un point-virgule à la fin d'une instruction, on annule l'affichage automatique du résultat ;
2. pour faire apparaître une valeur à l'écran, il suffit de composer son nom seul.

1.6 Affichage graphique

Forme : `plot(x,y)` où `x` et `y` sont des vecteurs de même longueur. Cette commande trace à l'écran `y` en fonction de `x`. Par défaut, les points seront reliés entre eux ; on peut toutefois choisir d'afficher uniquement les points avec l'instruction `plot(x,t,'o')`. Les autres caractères possibles sont `'x'`, `'.'`, `'*'` et `'+'`.

Si `x` est omis, les éléments de `y` sont tracés en fonction de leur rang dans le vecteur.

Plusieurs courbes peuvent être affichées sur le même graphique ; par exemple : `plot(x1,y1,x2,y2,x3,y3)`

Pour une échelle semi logarithmique ou logarithmique, on utilise les instructions `semilogx(x,y)`, `semilogy(x, y)` ou `loglog(x,y)`.

Pour l'affichage d'un titre et l'identification des axes, on utilise respectivement les instructions `title`, `xlabel` et `ylabel`. L'instruction `grid` trace des lignes de référence sur le graphique.

Exemple : En utilisant les vecteurs `t`, `r`, `s` et `a` définis dans l'exemple du paragraphe 1.5, on peut tracer les courbes à l'aide des instructions suivantes :

```
plot(t,r,'o',t,s,'x',t,a)
grid
```

```
title('Somme de deux sinusoïdes')
xlabel('Temps  (secondes)')
ylabel('Amplitude  (volts)')
```

1.7 Multiplication de matrices

Forme : `A=B*C` où `B` et `C` sont les matrices à multiplier. Après l'opération, le résultat sera dans `A`.

Attention : il ne faut pas confondre les instructions `B*C` et `B.*C`. La première est la multiplication matricielle de `B` et `C` dont les dimensions sont $M \times N$ et $N \times P$ respectivement. La deuxième est la multiplication élément par élément des matrices `B` et `C` dont les dimensions doivent être identiques.

On peut utiliser la multiplication matricielle pour effectuer des sommations de la forme :

$$\mathtt{s} = \sum_{n=0}^{N-1} a_n b_n$$

Soit `a` et `b` les vecteurs de dimension N dont les composantes respectives sont a_n et b_n. La somme `s` peut être évaluée par la multiplication matricielle suivante :

$$\mathtt{s} = \mathtt{a} \cdot \mathtt{b}^T$$

qui s'effectue à l'aide de l'instruction

```
s = a * b.'
```

Notez que la transposition (opérateur MATLAB `.'`) est indispensable pour que le produit matriciel représente en fait le produit scalaire des vecteurs `a` et `b` (qu'aurait-on obtenu avec l'instruction `s= a.' * b`?). Cette manière d'effectuer les sommations est *beaucoup plus efficace* que l'utilisation d'une boucle `for ... end` qui, en langage MATLAB, est notoirement lente.

1.8 Nombres complexes

MATLAB offre la possibilité de travailler avec des nombres complexes ; il s'agit simplement de définir les variables en conséquence.

Exemples :

```
z = sqrt(-1)
z1 = 5+2*z
z2 = exp(-z*pi/4)
```

Les variables MATLAB `i` et `j` sont initialement définies comme $(-1)^{1/2}$; cette définition est évidemment annulée lorsque ces variables sont redéfinies autrement.

1.9 Réponse en fréquence d'un filtre

L'instruction `bode` permet d'obtenir la réponse en fréquence d'un filtre dont on connaît la fonction de transfert.

Forme : `[M,P]=bode(N,D,W)` où `N` est le vecteur représentant le polynôme du numérateur de la fonction de transfert du filtre, `D` est le vecteur représentant le dénominateur, et `W` est un vecteur de fréquences (en radians/seconde) selon lesquelles la réponse est évaluée.

Après l'opération, `M` est un vecteur contenant le module de la réponse en fonction des fréquences contenues dans `W` et `P` est un vecteur contenant le déphasage (en degrés) en fonction de `W`.

On peut obtenir le module en dB en effectuant : `C = 20*log10(M)`. `C` est construit en effectuant l'opération $20\log_{10}(\cdot)$ sur chaque élément de `M`.

Exemple : pour obtenir la réponse en fréquence d'un filtre dont la fonction de transfert est $H(s) = s^3/(s^3+2s^2+2s+1)$ selon une centaine de points entre les fréquences 0 et 3 rad/s, on utilisera les instructions suivantes :

```
N = [1 0 0 0];          ← numérateur : 1s^3 + 0s^2 + 0s + 0
D = [1 2 2 1];          ← dénominateur: 1s^3 + 2s^2 + 2s + 1
W = 3*(0:100)/100;
[M,P] = bode(N,D,W);
```

Les résultats peuvent être ensuite affichés graphiquement à l'aide des instructions vues au paragraphe 1.6. Dans le cas où seul le résultat graphique nous intéresse, on peut omettre le membre de gauche de la dernière ligne et écrire uniquement `bode(N,D,W)` ce qui permet l'affichage direct du résultat.

1.10 Réponse temporelle d'un filtre

Il est possible d'évaluer la réponse temporelle d'un filtre à l'impulsion, à l'échelon et à une entrée quelconque. Pour la réponse à l'impulsion, on utilise l'instruction `A=impulse(N,D,T)`, où `N` et `D` sont les vecteurs représentant respectivement le numérateur et le dénominateur de la fonction de transfert du filtre, et où `T` est le vecteur des instants (régulièrement espacés) auxquels sera calculée la réponse du système. Après la simulation, cette réponse est contenue dans le vecteur `A`.

Pour la réponse à l'échelon, on utilise l'instruction `B=step(N,D,T)`.

Pour obtenir la réponse d'un filtre à une entrée quelconque déterminée par l'usager, on utilise l'instruction `C=lsim(N,D,E,T)`, où `N` et `D` sont les vecteurs représentant respectivement le numérateur et le dénominateur de la fonction de transfert du filtre ; `T` est le vecteur de temps, et `E` est un vecteur représentant l'amplitude du signal d'entrée en fonction du temps `T`. Après la simulation, `C` est un vecteur contenant la réponse du système au signal d'entrée représenté par `E` en fonction du temps `T`.

Lorsqu'elles sont appelées seules (sans membre de gauche), ces instructions retournent le résultat sous forme graphique.

1.11 Macro-instructions

Bien que chaque instruction puisse être entrée directement au clavier, il est souvent utile de grouper un ensemble d'instructions à exécuter qui sera identifié par un seul nom et qui pourra être conservé et édité à volonté. Ceci est possible en plaçant les instructions dans un *fichier dont le nom se termine par* `.m`.

Exemple : on crée un fichier nommé `graphe.m` comprenant les lignes suivantes :

```
% COMMANDE GRAPHIQUE
plot(X,Y)
grid
% AFFICHAGE DU TITRE
title('RÉSULTATS')
% IDENTIFICATION DES AXES
```

```
xlabel('TEMPS(SEC)')
ylabel('AMPLITUDE')
```

Chaque fois que `graphe` est appelé du clavier ou d'un autre fichier, la courbe de `Y` en fonction de `X` est tracée à l'écran avec affichage du titre et identification des axes (on suppose ici que les vecteurs `X` et `Y` ont été préalablement définis avant l'appel de `graphe`).

Il est possible également de créer ses propres fonctions MATLAB auxquelles on doit transmettre un ou plusieurs paramètres.

Exemple : on crée un fichier nommé `sa.m` comprenant les instructions suivantes :

```
function Y=sa(X)
if X==0
  Y=1;
else
  Y=sin(x)/x;
end
```

Chaque fois que l'on demande `sa(v)`, on obtient l'évaluation de sin(v)/v. Avant l'exécution de la fonction, la valeur du paramètre transmis est placée dans `X` ; le résultat retourné est la valeur de `Y` après que toutes les instructions aient été exécutées.

Il est souligner que *les variables d'une fonction de type* `function` *sont indépendantes des autres variables, ce qui n'est pas le cas lorsque le fichier ne débute pas par l'instruction* `function`.

Une fonction peut retourner plusieurs résultats.

Exemple :

```
function [X,Y,Z]=CALCUL(A,B,C,D)
X=A+B+C+D
Y=A*B*C*D
Z=A*B+C*D
```

Les fichiers `.m` peuvent être créés à l'aide de l'éditeur de texte de votre choix.

2 Procédure initiale

Le logiciel MATLAB, ainsi qu'un éditeur sont disponibles sur le réseau des micro-ordinateurs du local A-408. *En dehors des heures de travaux pratiques,* MATLAB *est disponible au C-539.4.*

Une fois MATLAB chargé on peut accéder à l'éditeur selon une procédure qui sera expliquée au laboratoire.

Ne pas oublier pas que les noms des fichiers doivent se terminer par `.m` pour que MATLAB les reconnaisse.

Remarque : Lorsqu'un fichier `function` a été modifié à l'aide de l'éditeur pendant une séance MATLAB, n'oubliez pas d'effacer l'ancienne version de la mémoire vive de l'ordinateur en composant `clear`, suivi du nom de la fonction ; vous êtes ainsi assurés que MATLAB ira lire la nouvelle version sur le disque.

3 Exercices suggérés

Note : habituez-vous dès maintenant aux manipulations de fichiers à l'aide de l'éditeur.

Exercice 1

1. Générez la matrice $\texttt{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$.
2. Générez la transposée de `A`.
3. Trouvez le déterminant de `A`.

Exercice 2

1. Générez le vecteur `a` = (0 1 2 3 4 5 6)
2. À partir du vecteur `a`, générez le vecteur `b` = (0 1 4 9 16 25 36) en une seule opération, *sans utiliser de boucle* `for ... end`.
3. Calculez *sans boucle* la somme

$$\texttt{s} = \sum_{n=0}^{6} b_n e^{-a_n/10}$$

où a_n et b_n sont les éléments de `a` et `b`.

Exercice 3

1. Soit le vecteur `a` défini par

$$\texttt{a} = (3+4i \quad 5+9i \quad -3-4i \quad -5-9i \quad 3-4i \quad 5-9i \quad -3+4i \quad -5+9i).$$

À l'aide d'une boucle `for ... end` et d'une instruction conditionnelle `if ... end`, générez le vecteur `b` contenant les éléments de `a` dont la partie imaginaire est positive.

2. Générez maintenant le vecteur `b` *sans utiliser de boucle* `for ... end`. Vous pourrez faire appel à l'instruction `find`.
3. Générez le vecteur `m` contenant les modules des éléments du vecteur `a`.
4. Générez le vecteur `p` contenant les phases des éléments du vecteur `a`.

Exercice 4

Soit $H(s)$ la fonction de transfert d'un filtre définie par :

$$H(s) = \frac{1}{s^4 + 2,61s^3 + 3,41s^2 + 2,61s + 1}\,.$$

1. À l'aide de l'instruction `roots`, trouvez les pôles de $H(s)$.
2. La fonction `plot(z)` où `z` est un vecteur complexe permet de tracer la partie imaginaire de `z` en fonction de sa partie réelle. En utilisant cette propriété, représentez graphiquement les pôles de $H(s)$ dans le plan complexe, sans omettre d'identifier les axes.

Remarques

Les échelles sont choisies automatiquement. En cas de besoin, il est toutefois possible d'imposer une échelle d'affichage en employant l'instruction `axis([xmin xmax ymin ymax])` après une commande `plot`. On peut annuler cette échelle en composant `axis('auto')`.

Pour imprimer un graphique affiché à l'écran, utilisez l'option `PRINT` sous la rubrique `FILE`.

Exercice 5

En une seule instruction et sans utiliser de boucle, générez les vecteurs :

1. `a` = (0 5 10 15 ... 100)
2. `b` = (0.1 ... 10), dont la progression des éléments est logarithmique.

Exercice 6

1. À l'aide de la fonction MATLAB `[M,P]=bode(NUM,DEN,W)`, trouvez le diagramme de Bode du filtre dont la fonction de transfert est :

$$H(s) = \frac{1}{s^4 + 0,58s^3 + 1,17s^2 + 0,40s + 0,18}\,.$$

   Comme vecteur de fréquences, on utilisera le vecteur `b` obtenu à l'exercice précédent.
2. Transformez `M` afin d'avoir le module en dB.
3. Affichez graphiquement le module en fonction de la fréquence en utilisant l'instruction `semilogx(W,M)` et les commandes d'affichage du titre et d'identification des axes.

Exercice 7

Soit le filtre représenté par la fonction

$$H(s) = \frac{1}{s^3 + 2s^2 + 2s + 1}$$

Obtenez graphiquement sa réponse temporelle à la fonction échelon. Le vecteur des instants où la réponse est calculée pourra être spécifié par l'instruction `T=0:4E-3:0.2`, mais cette définition pourra être modifiée si vous ne la jugez pas adéquate.

Exercice 8

Composez une macro-instruction qui, à partir d'un vecteur d'entrée représentant un polynôme, sélectionne les racines situées à gauche de l'axe imaginaire.

Exercice 9

Sans utiliser de boucle, modifiez la fonction `sa` du paragraphe 1.11 de façon à ce que le paramètre d'entrée `X` puisse être un vecteur plutôt qu'une valeur scalaire.

4 Aide en ligne de Matlab

Une aide en ligne est disponible avec le logiciel MATLAB. Il suffit de se reporter à la rubrique *Table of Contents* du menu *Help* de la fenêtre de commande MATLAB. On a ainsi accès à une liste structurée de l'ensemble des instructions du langage, puis, par des liens hypertextes, à une description de chaque instruction. Pour une description succincte d'une instruction dont on connaît le nom, on peut aussi taper

`help nom_de_l'instruction`

dans la fenêtre de commande.