OUTILS MATHÉMATIQUES ET UTILISATION DE MATLAB

High School of Technology, Ibn Zohr University, Agadir, Morocco.

2019-2020

L'interface de Matlab (1): Command Window

C'est le terminal dans lequel on doit taper les commandes et sur lequel on verra l'affichage des résultats. Une ligne commence toujours par >>. Essayez la commande suivante:

```
>> 1+1
ans =
      2
```

et comparez à la commande :

```
>> 1+1;
```

Une commande utile lorsque l'on a un doute sur le type de variable que l'on vient de créer est whos :

```
>> whos('ans')
   Name      Size      Bytes    Class    Attributes
   Ans       1x1           8    double
```

L'interface de Matlab (2): Editor

- La plupart de votre travail sous Matlab va consister à créer ou modifier des fichier.m. Lorsque l'on réalise une tache sous Matlab, il est très souvent possible de le faire en utilisant uniquement la Command Window. Cependant lorsque cette tache devient plus complexe (plusieurs dizaines de ligne de code) on utilise la fenêtre Editor.
- Par exemple, on pourra réaliser une fonction *racineplus2(input)* qui à un paramètre d'entrée *input* va répondre $\sqrt{input + 2}$:

```
function [ output ] = racineplus2( input )
output=sqrt(input+2);
end
```

- On peut ensuite appeler cette fonction simplement dans la Command Window par :

```
>> racineplus2(7)
ans =
        3
```

L'interface de Matlab (3): "Hello World"

- Le script est le fichier .m le plus simple. Il s'agit simplement d'une liste de commandes. Pour commencer on fixe le USERPATH. On crée ensuite un fichier .m dans ce dossier et on nomme ce fichier hello.m:

```
str='Hello world';
str
```

- On commence de façon similaire à pour un script : on crée un fichier .m que l'on nomme hello2.m:

```
function [ str ] = hello2 ( prenom )
str = ['Hello ', prenom ];
end
```

- Remarque: utiliser la commande suivante:

```
>> newpath = 'C:\Users\Dell\Desktop\TP-GE';
userpath(newpath)
```

Outils de base: Types de variables

- Il existe cinq grands types de variables sous Matlab : les entiers, les réels, les complexes, les chaînes de caractères et le type logique. Définissons une variable de chaque type :

```
>> a = 1.3;   b = 3+i;   c = 'bonjour';
>> d1 = true(1==1);  d2 = logical(1);
>> e = int8(2);
```

On peut alors vérifier le type de ces différentes variable en utilisant la fonction whos :

```
>> whos

   Name        Size        Bytes    Class        Attributes
```

Arithmétique et opérations sur les scalaires

```
>> x = 1+1
x = 2
```

- On peut également travailler avec des variables définies par l'utilisateur :

```
>> x = 2; y = 1.5 ;
somme = x+y
difference = x-y
produit = x*y
division = x/y
```

```
>> x = 2; y = pi;
cos(y)
exp(x)
sqrt(x)
```

Vecteurs (1)

- La méthode la plus simples pour définir un vecteur est de donner sa description explicite à l'aide de la commande [ ], par exemple:

```
vec = [1   2   4   7   9   2.3]
```

- On peut également définir un vecteur colonne en utilisant le ;

```
col = [1 ; 2 ; 4 ; 7]
```

- On peut concaténer deux vecteurs :

```
vec1 = [1 3 5];
vec2 = [9 10 11];
vec = [vec1 vec2]
```

Vecteurs (2)

- On peut également prendre la transposée pour passer d'une ligne à une colonne ou réciproquement:

```
vec1 = [1 3 5];
vec = vec1'
```

- Il n'est pas nécessaires de définir la taille d'un vecteur (c'est automatique), par contre la commande length() permet de retourner cette quantité.

```
length(vec)
```

- Une autre méthode pour générer des vecteurs espacés linéairement consiste à utiliser [a, s, b]. On crée alors un vecteur entre a et b avec un espacements :

```
vec=[1:2:10]
```

Vecteurs (3)

- Le k-eme élément d'un vecteur vec peut être affiché grâce à la commande vec(k).

```
>> vec = linspace(1,10,10)
vec = 1 2 3 4 5 6 7 8 9 10
>> vec(4)
ans = 4
```

- On peut également utiliser des vecteurs d'indices pour extraire un sous-vecteur :

```
>> vec = linspace(1,89,9)
vec = 1 12 23 34 45 56 67 78 89
>> vec(3:6)
ans = 23 34 45 56
>> vec(4:2:8)
ans = 34 56 78
>> subvec=[1 3 5]; vec(subvec)
ans = 1 23 45
```

Opérations vectorielles (1)

```
>> vec = [1 3 5 6]; vec2 = [10 20 30 40];
>> vec+vec2
ans = 11   23   35   46
>> vec-vec2
ans =  -9  -17 -25   -34
>> vec.*vec2
ans = 10   60   150 2  40
>> vec2./vec
ans = 10.0000        6.6667        6.0000       6.6667
```

```
>> vec = linspace(1,10,10);
out=sqrt(vec);
>> out2=vec.^2
>> out3=cos(vec)
```

Opérations vectorielles (2)

Il existe aussi des commandes qui sont propres aux vecteurs:

- sum(x) : somme des éléments du vecteur x
- prod(x) : produit des éléments du vecteur x
- max(x) : plus grand élément du vecteur x
- min(x) : plus petit élément du vecteur x
- mean (x) : moyenne des éléments du vecteur x
- sort(x): ordonne les éléments du vecteur x par ordre croissant
- fliplr(x) : renverse l'ordre des éléments du vecteur x

TP 1

Donnez le code Matlab qui permet de :

1. Créez un vecteur colonne vec de 5 éléments linéairement espacés entre 2 et 3.
2. Ajoutez deux lignes à la fin de ce vecteur avec la valeur 0.
3. Ajoutez 1 au deuxième et sixième éléments de ce vecteur.
4. Créez un second vecteur vec2 colonne de même dimension que vec contenant les entiers pairs supérieurs ou égaux à 6.
5. Définir un vecteur sumvec comme la somme des deux vecteurs vec et vec2.
6. Définir un vecteur prodvec comme le produit termes à termes des deux vecteurs vec et vec2.
7. Quel est la somme des éléments de prodvec ?
8. Quel est la moyenne des éléments de sumvec ?
9. Quel est le plus grand éléments du vecteur $vec3=\frac{vec^2+\sqrt{vec2+1}}{vec.(vec2+1)}$?

Les matrices

- Une matrice va se définir de façon similaire à un vecteur avec la commande [ ]. On définit la matrice A :

$$A=\begin{pmatrix}1 & 2\\ 4 & 3\end{pmatrix}$$

```
>> A = [1 2 ; 4 3]
```

Cela est équivalent à

```
>> A = [1 2
4 3]
```

- Une matrice est composée de m lignes et n colonnes. Si on souhaite connaitre la valeur de m ou n, on utilise la commande size(A)

```
>> [m n] = size(A)
>> size(A)
>> size(A,1)
```

Les matrices

- Comme pour les vecteurs il existe des matrices prédéfinies:

- eye(n): la matrice identité (carrée de taille n)
- ones(m,n) : la matrice à m lignes et n colonnes dont tous les éléments valent 1
- zeros(m,n) : la matrice à m lignes et n colonnes dont tous les éléments valent 0
- rand(m,n) : une matrice à m lignes et n colonnes dont les éléments sont générées de manière aléatoire entre 0 et 1.
- magic(n) : une matrice magique de dimension n.

Les matrices

- Pour extraire un élément de la matrice on indique la ligne et la colonne de celui-ci:

```
>>A = [1 2 5 ; 4 3 6];
>>A(2,1)
```

Lorsque l'on souhaite extraire une colonne ou une ligne entière on utilise le symbole (:) comme on va le voir dans l'exemple suivant :

```
>> A(2,:)
>>A(:,1)
```

- Toutes les combinaisons sont alors possibles. On peut extraire 2 colonnes par exemple en faisant :

```
>> A(:,[1 2])
```

Les matrices

- Comme pour les vecteur il est possible d'obtenir la transposée d'une matrice avec la commande '.
- La commande diag permet d'extraire la diagonale d'une matrice.
- La même commande permet aussi de créer une matrice de diagonale fixée :

```
>> A=eye(3)
diag(A)'
ans =
1 1 1
>> v=[1:3];
>> diag(v)
ans =
                1 0 0
                0 2 0
                0 0 3
```

Les matrices Triangulaires

- On dispose également de la commande tril permet d'obtenir la partie triangulaire inférieure (l pour lower) d'une matrice. La commande triu permet d'obtenir la partie triangulaire supérieure (u pour upper) d'une matrice.

```
>> A = [ 2 1 1 ; -1 2 1 ; -1 -1 2]
>>triu(A)
>>triu(A)
```

- Comme pour la commande diag, les commandes triu et tril admettent un second paramètre k. On peut ainsi obtenir la partie triangulaire supérieure (ou inférieure) à partir de la k ediagonale. Ainsi,

```
>> A = [ 2 1 1 ; -1 2 1 ; -1 -1 2]
>>triu(A,1)
>>triu(A,-1)
```

TP2

- Construire la matrice T tri-diagonale à l'aide de la commande diag() utilisée 3 fois :

$$T = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}$$

1. Extraire de T les deux premières colonnes.
2. Extraire de T les éléments des colonnes et des lignes 2 à 4.
3. Créer une matrice T2 où la ligne 1 est échangée avec la ligne 3 puis la colonne 2 est remplacée par les valeurs de la colonne 4.

Opérations et fonctions portant sur les matrices

1. det(A) : renvoie le déterminant de la matrice carrée A
2. eig(A) : renvoie les valeurs propres (eigenvalues) de la matrice carrée A
3. inv(A) : renvoie l'inverse de la matrice carrée A.
4. norm(A) : renvoie la norme 2 de la matrice A
5. norm(A,2) : même chose que norm(A).
6. norm(A,1) : norme 1 de la matrice A (max sur j)
7. norm(A,inf) : norme infini de la matrice A (max sur i)

TP3

- Soit la matrice suivante:

$$A=\begin{pmatrix}1 & 1 & 1\\ 0 & 2 & 3\\ 0 & 1 & 1\end{pmatrix}$$

1. Calculer le déterminant de la matrice A.
2. Calculer l'inverse de la matrice A.
3. Calculer les valeurs propres de A.
4. Calculer la norme 2 de A.
5. Calculer la norme 1 de A.
6. Calculer la norme infinie de A.

Opérations matricielles

- Sous Matlab, on calcule le produit matriciel en utilisant simplement le signe A*B:

```
>> A = [4 2 ; 0 1]; B = [1 2 3; 5 4 6];
>>M=A*B
```

- Pour calculer l'inverse d'une matrice:

```
>> A = [4 2 ; 0 1];
>>M = inv(A)
```

- La division se définit à partir de l'inverse :

$$\frac{A}{B} = AB^{-1}$$

```
= [4 2 ; 0 1]; B = [1 2 ; 5 4];
>> M = A/B
```

Résoudre un système linéaire

- Le système que nous cherchons à résoudre est le suivant:

$$\begin{cases} 3x + 5y + z = 1 \\ 7x - 2y + 4z = -3 \\ -6x + 3y + 2z = 3 \end{cases}$$

```
>> A = [3 5 1 ; 7 -2 4 ; -6 3 2]; b = [1 -3 3]';
>> X=inv(A)*b
```

La seconde méthode est plus élégante.

```
>> A = [3 5 1 ; 7 -2 4 ; -6 3 2]; b = [1 -3 3]';
X = A\b
```

Graphique d'une fonction

- La fonction de base pour tracer un graphique avec MATLAB est la commande plot qui prend comme arguments une série de points donnés sous la forme de 2 vecteurs, qu'elle relie de segments de droites.

```
>> x=0:0.1:2*pi;                                % l'ordonnée
>> plot(x,sin(x),'b-o',x,cos(x),'m--+'); % le graphe du sinus et du cosinus
>> axis([0 2*pi -1.1 1.1]);                    % définition des axes
>> title('Le titre du graphique');
>> xlabel('L''axe des x');
>> ylabel('L''axe des y');
>> legend('sinus','cosinus');
```

```
>> clf reset                    % on réinitialise l'environnement graphique
>> hold on
>> plot(x,sin(x),'b-o');        % un premier graphique
>> plot(x,cos(x),'m--+');       % on superpose un deuxième graphique
>> hold off
```

Graphique de plusieurs fonctions

On peut aussi comparer des résultats sous forme graphique à l'aide de la commande subplot :

```
>> subplot(2,2,1); plot(x,sin(x),'b-o'); axis([0 2*pi -1.1 1.1]);
>> subplot(2,2,2); plot(x,cos(x),'m-+'); axis([0 2*pi -1.1 1.1]);
>> subplot(2,2,3:4); plot(x,sin(x),'b-o',x,cos(x),'m--+');
>> axis([0 2*pi -1.1 1.1]);
```

La fonction fplot facilite le tracé de graphes de fonctions, en automatisant le choix des points où les fonctions sont évaluées :

```
>> fplot('[sin(x),cos(x)]',[0 2*pi],'b-+')
```

La commande fplot permet de tracer le graphe d'une fonction sur un intervalle donné. >> fplot('[sin(x)/x , cos(x)/x]', [-5, 5, -1, 1])

TP4

- Tracer le graphe de la fonction

  - f(x) = x sin(x) entre −2*pi et 2*pi.

- Tracer dans la même figure les fonctions suivantes:
  - g(x)=exp(x) entre -1 et 1
  - h(x)=ln(x) entre 1/e et e
  - k(x)=x.

y	m	c	r	g	b	k
jaune	magenta	cyan	rouge	vert	bleu	noir

Les graphiques tridimensionnels

```
>> e = exp(1);
>> figure
>> hold on
>> fplot('exp',[-1 1])
>> fplot('log',[1/e e])
>> plot([-1:0.01:e],[-1:0.01:e])
>> grid
>> hold off
```

Finalement, les graphiques tridimensionnels, de type paramétrique, sont tracés à l'aide d'une généralisation de la commande plot :

```
>> t=linspace(-5,5,1000);
>> x=(1+t.^2).*sin(20*t);
 >> y=(1+t.^2).*cos(20*t);
>> z=t;
>> plot3(x,y,z);
>> grid on;
```