UNIVERSITE DE GHARDAIA

DEPARTEMENT DES MATHEMATIQUES ET INFORMATIQUE

SUPPORT DE COURS

O.P.M

EXERCICES ET ENONCE DE TP

1EME ANNEE M.I

Suite du cours : OPM

Les Polynômes

Un Polynôme de la forme :

$P(x)=a_n x^n+a_{n-1}x^{n-1}+\ldots+a_2x^2+a_1x+a_0$

Est représenté en Matlab par un vecteur ligne **$[a_n, a_{n-1},\ldots,a_2,a_1,a_0]$** ou vecteur de colonne.

Pour évaluer un polynôme **P(a)** avec **a** une valeur fixe, on utilise la commande **Polyval(P,2)**

```
>> P = [1 2 1] ;

>> polyval(P, 0)

ans =

      1
```

Dérivation et Intégration d'un polynôme

- La commande **polyder(P)** retourne la dérivée d'un polynôme P.

- La commande **polyint(P)** retourne l'intégral d'un polynôme P.

Si on veut calculer l'intégral entre deux point x1 et x2, on utilise la commande **polyval** pour évaluer l'intégral sur ces deux points puis on applique la soustraction.

```
>> P = [1 2 1] ;

>> polyder(P)
ans =
       2      2
>> polyint(P)
ans =
      0.3333        1.0000       1.0000 0
>> polyval(Pint, 2) - polyval(Pint, 1)
ans =
       6.3333
```

Les statistiques descriptives

L'ensemble des fonctions suivantes prennent en arguments soit un vecteur, soit une matrice. Dans le cas d'une matrice chaque colonne est traitée comme une variable différente et le résultat des fonctions est donc un vecteur.

```
x = [1 5 74 ; 1 5 74 ; 3 6 10 ; 5 9 20 ; 7 8 99] ;
geomean(x)        % Moyenne géométrique
harmmean(x)       % Moyenne harmonique
mean(x)           % Moyenne arithmétique
median(x)         % Elément milieu du vecteur trie ´
mode(x)           % Valeur la plus fréquente
var(x)            % Variance (n-1)
range(x)          % Intervalle des valeurs
max(x)            % Valeur maximum
min(x)            % Valeur minimum
prod(x)           % Produit des ´éléments par colonnes
sum(x)            % Somme des ´éléments par colonnes
sort(x)           % Tri par colonnes
```

Affichage Graphique des Fonctions

Les possibilités graphiques de MATLAB sont innombrables.

Pour créer un graphe de fonction par exemple, il faut commencer par définir deux vecteurs:

x = 0:0.5:20;

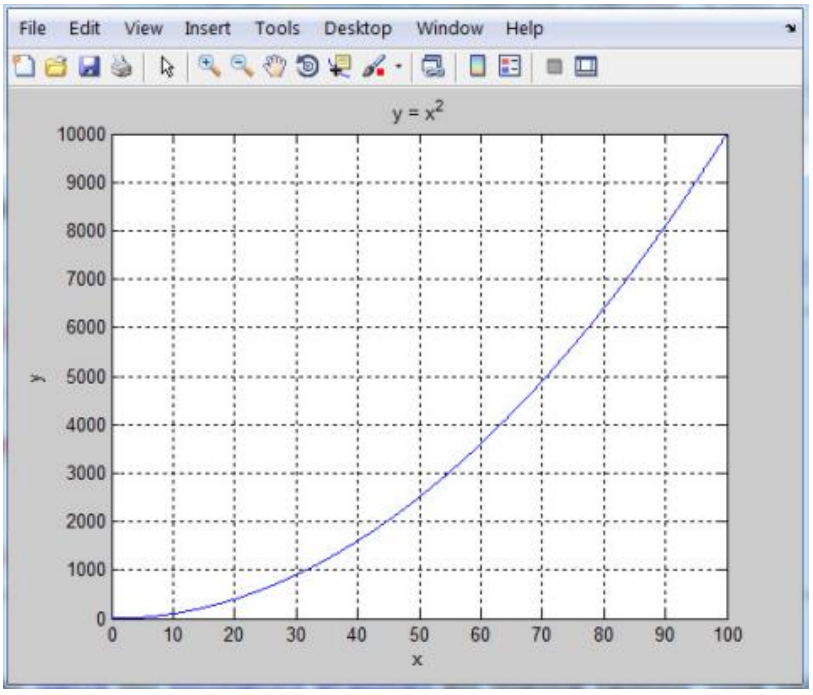
y = x^2;

ensuite on construit le graphe au moyen de la commande: **plot(x,y)**

On peut utiliser une grille (un quadrillage) ou ne pas le faire par les commandes: **grid** (ou **grid off**)

On peut aussi ajouter un titre au graphe, ajouter des étiquettes aux axes par la commande : **title('graphe de sinus'), xlabel('x'), ylabel('y')**

```
>> x = linspace(0, 100, 1000);                  % créer une liste de 1000 éléments
>> y = x.^2;                                    % obtenir leurs carrées
>> plot(x,y)                                    % afficher le graphe des carrées
>> grid                                         % afficher la grille
>> title('y = x^2'), xlabel('x'), ylabel('y')   % afficher le titre et les étiquettes
```



La commande **axis([xmin xmax ymin ymax])** permet d'introduire la valeur maximale et minimale pour chaque axe.

La commande **axis auto** permet de choisir automatiquement les valeurs limites de chaque axe.

```
>> axis([1 5 1 5])          % échelle unique pour x et y
>> axis([1 5 1 10])         % échelle différent pour x et y
>> axis auto                % échelle automatique pour x et y
```

-La commande **plot(x, y, LineSpec)**, affiche un graphe avec la spécification donnée par **LineSpec** qui est formée au plus de trois valeurs: **style de ligne, type de marquer,** et **la couleur**.

Par exemple :

plot(x, y, '-+b') : affiche un graphe bleu de la forme -+-+-+…

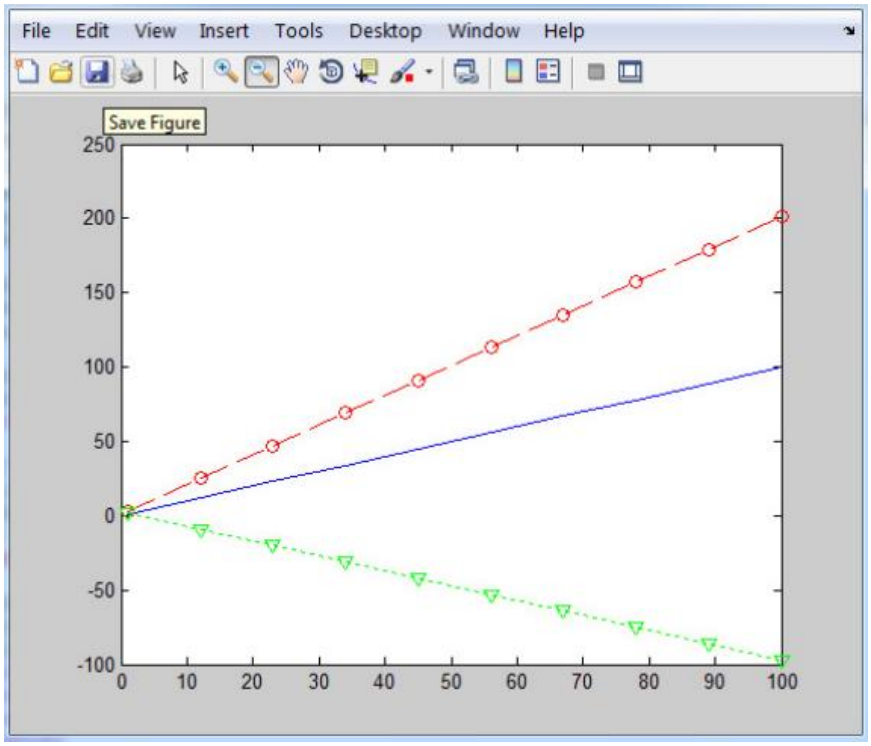
-Noter bien que la spécification **'type de marqueur'** est utilisée pour les points choisies (vecteurs x et y), par contre la spécification **'style de ligne'** couvre toute la courbe.

Symbole	Style de ligne
-	Ligne solide (par défaut)
--	La ligne non continue
:	La ligne pointillée
-.	La ligne tiret-point

Symbole	Type de Marqueur
o	Cercle
+	Signe plus
*	Astérisque
.	Point
x	Croix
s	Carré
d	Diamant
^	Triangle vers le haut
v	Triangle vers le bas
>	Triangle pointant à droite
<	Triangle pointant à gauche
p	Pentagramme
h	Hexagramme

Symbole	Couleur
y	Jaune
m	Magenta
c	Cyan
r	Rouge
g	Vert
b	Bleu
w	Blanc
k	Noir

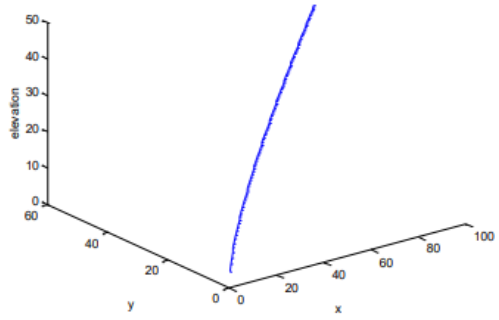
```
>> x = linspace(1,100, 10);
>> y3 = -x+3;
>> plot(x, y1, '-b', x, y2, '--or',x, y3, ':vg')
% y1 en bleu et ligne solide
% y2 en rouge et ligne non-continue, marquer o
% y3 en vert et ligne en pointillé
```

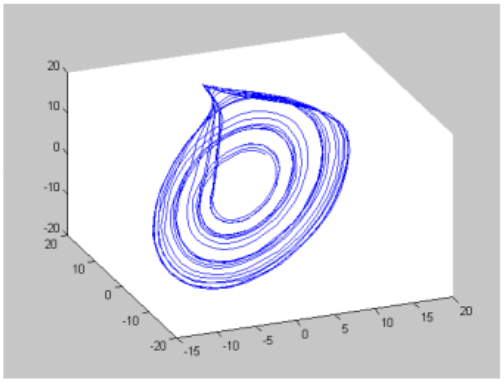


Le graphisme 3D

L'instruction **plot3**, permet de tracer une variables **z** en fonction de deux variables x et y (vous pouvez trouver de l'aide **plot3** pour la syntaxe).

```
>> x=1:100;
>> y=0.5:0.5:50;
>> z=2*x.^0.5+3*y.^0.5;
>> plot3(x,y,z),xlabel('x'),ylabel('y'),zlabel('elevation'),title('Exemple')
```





Affichage des images

(Veuillez utilser l'aide de matlab pour voir la syntaxe)

L'instruction **image** :

-Pour lire l'image :

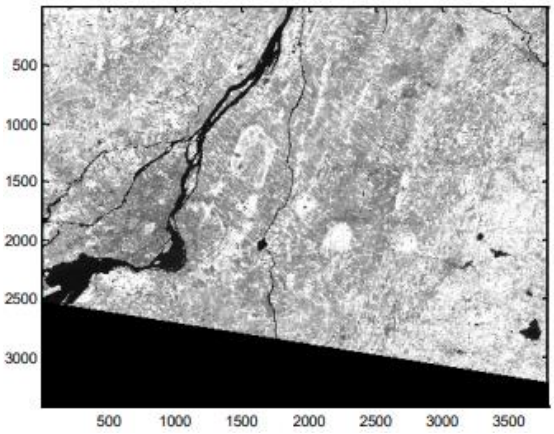
fid=fopen('b4juin.raw','r');

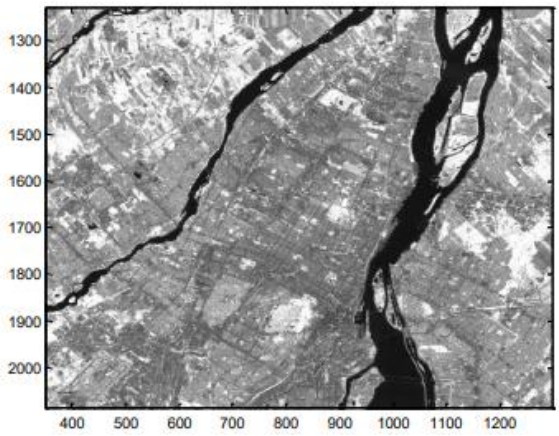
IM = fread(fid,[3786,3424],'uint8') ;

fclose(fid);

-Pour l'afficher :

Image(IM) utilise sa palette de couleurs par défaut Ou
colormap(gray);image(IM) pour affichage en niveaux de gris





Les Fichiers

Dès que l'on manipule un nombre conséquent de données (, on est amené à utiliser des fichiers de données pour stocker ces données afin de les réutiliser dans MATLAB ou d'autres logiciels.

Écriture de fichiers de données (.mat)

Il est possible de sauvegarder toutes ou certaines variables existantes dans le **workspace**, sous forme d'un fichier **.mat** (spécifique à MATLAB), qui conserve la structure des données MATLAB : pour cela, on utilisera la fonction **save**.

1-Pour sauvegarder toutes les variables définies dans le workspace, On utilise la commande suivante :

save(nom_Fichier_mat) ;

```
save('DATAS.mat') ;   % sauvegrade de tout le workspace
```

nom_Fichier_mat : est une chaîne de caractères correspondant au nom du fichier mat que l'on souhaite créer.

2-Pour sauvegarder certaines variables uniquement, il suffit de préciser leur nom, sous forme de chaînes de caractères :

save(nom_Fichier_mat,nom_Variable_1,…,nom_Variable_n) ;

```
save('./Docs/DATAS_xy.mat','x','y') ;   % sauvegarde des variables x et y
```

nom_Fichier_mat : est une chaîne de caractère correspondant au nom du fichier mat que l'on souhaite créer, et nom_Variable_1,
... nom_Variable_n sont les chaînes de caractères correspondant aux noms des variables spécifiques à enregistrer.

Créer un fichier CSV

Un fichier CSV (*Comma-separated values*) est un format de fichier stockant des données tabulaires sous forme de valeurs séparées par des virgules.
L'extension du fichier est généralement (.csv)

Ce format n'est pas spécifique à MATLAB. Il est adapté aux données numériques. Si l'on souhaite récupérer des données pour les traiter avec un logiciel externe, il est utile de savoir créer un fichier CSV : pour cela, on utilise la fonction suivante :

csvwrite(nom_Fichier_csv, tableau_Donnees) ;

```
>> M = [3 6 9 12 15; 5 10 15 20 25; ...
  7 14 21 28 35; 11 22 33 44 55];
>> csvwrite('Data.csv',M) ;
```

nom_Fichier_csv : est une chaîne de caractères correspondant au nom du fichier csv que l'on souhaite créer, et **tableau_Donnees** est le tableau à sauvegarder.

La fonction **type** permet d'afficher le contenu de ce fichier ('Data.csv').

```
>> type('Data.csv') ;

3,6,9,12,15
5,10,15,20,25
7,14,21,28,35
11,22,33,44,55
```

Note : Le fichier **csv** peut porter n'importe quelle extension (voire aucune).

Souvent, on choisit comme extension **csv**, **dat** ou **txt**. Ce choix peut être guidé par l'usage que l'on souhaite en faire : pour des raisons pratiques, certains logiciels imposent l'extension du fichier.

Créer un fichier texte, sans contraintes

Le format `csv` (qu'on vient de voir) est un format de fichier simple et courant, lisible comme tout fichier texte, mais il ne permet pas de stocker n'importe quelles données (seulement des valeurs numériques), et les valeurs sont stockées avec cinq décimales au plus.

Si l'on veut s'affranchir de ces contraintes, tout en conservant le côté pratique des fichiers texte, il faut créer son propre fichier.

La fonction **fprintf** permet d'écrire une chaîne de caractères formatée dans un fichier donné, selon la syntaxe suivante :

Note : Les étudiants sont invités à revoir les Formats de données utilisé par le logiciel Matlab à l'adresse suivante :

https://fr.mathworks.com/help/matlab/matlab_prog/formatting-strings.html

```
pointeur_vers_fichier = fopen('Nom_du_fichier.txt','w') ;
fprintf(pointeur_vers_fichier, Chaine_de_Caracteres_avec_Plusieurs_Formats,
        variable_a_formater_1, ..., variable_a_formater_N) ;
fclose(pointeur_vers_fichier) ;
```

```
% création des données
x=linspace(0,pi,100);
```

```
y=cos(x).*sin(x);
Datas=[x;y];
%
% ouverture du fichier en écriture
fileID=fopen('MonFichier.txt','w');
%
% écriture de x et y selon le format choisi
fprintf(fileID,'x : %2.8f \t f(x) = %2.6f \n',Datas);
%
% fermeture du fichier
fclose(fileID);
```

Pour visualiser le résultat, on utilise la fonction **type** :

```
>> type('MonFichier.txt')

x : 0.00000000   f(x) = 0.000000
x : 0.03173326   f(x) = 0.031712
x : 0.06346652   f(x) = 0.063296
x : 0.09519978   f(x) = 0.094626
x : 0.12693304   f(x) = 0.125574
x : 0.15866630   f(x) = 0.156017
x : 0.19039955   f(x) = 0.185831
x : 0.22213281   f(x) = 0.214897
  ...
  ...
x : 3.04639288   f(x) = -0.094626
x : 3.07812614   f(x) = -0.063296
x : 3.10985939   f(x) = -0.031712
x : 3.14159265   f(x) = -0.000000
>>
```

Lecture de fichiers de données

Nous avons vu comment écrire dans des fichiers mat ou csv, nous allons voir maintenant comment lire dans ce type de fichiers pour récupérer des données existantes.

La fonction load

Pour récupérer les données contenues dans un fichier **mat**, il faut utiliser la fonction **load**, avec une syntaxe équivalente à celle de la fonction **save** :

-Si l'on veut lire toutes les variables d'un fichier mat pour les rendre disponibles dans le **workspace**, on utilise la syntaxe suivante :

load(nom_Fichier_mat) ;

```
>> load('DATAS.mat') ;
```

nom_Fichier_mat : est une chaîne de caractères correspondant au nom du fichier mat que l'on souhaite lire.

-Si l'on veut récupérer certaines variables uniquement, il suffit de préciser leurs noms, sous forme de chaînes de caractères :

load(nom_Fichier_mat,nom_Variable_1,...,nom_Variable_n) ;

```
load('./Docs/DATAS_xy.mat','x','y') ;
```

nom_Fichier_mat : est une chaîne de caractères correspondant au nom du fichier mat que l'on souhaite lire, et **nom_Variable_1, ..., nom_Variable_n** sont les chaînes de caractères correspondant aux noms des variables spécifiques à récupérer.

Note : La fonction **load** sait lire d'autres formats que **les fichiers mat** : si le fichier à ouvrir est d'un autre format, **load** fera automatiquement appel à d'autres fonctions (par exemple **csvread** ou **importdata**) afin d'employer la méthode la plus adaptée au format reconnu...

La fonction csvread

La fonction **csvread** permet la lecture seulement les données numériques contenues dans un fichier de type CSV ;

```
DATAS=csvread('Data.csv');
```

Il est possible de ne récupérer qu'une partie des données, en spécifiant l'indice de ligne et de colonne de la première donnée lue :

```
%lecture à partir de la ligne n°1599 et de la première colonne
DATAS=csvread('Data.csv',1599,0);
```

La fonction importdata

Un fichier de données peut comporter des lignes de commentaires, ou avoir un autre séparateur que la virgule. Dans ces cas-là, la fonction **csvread** est inadaptée.

Il est préférable d'utiliser la fonction **importdata**.

```
RD = importdata('Data.csv',';',2);
```

```
datas = RD.data;
titre = RD.textdata(1);
labels = RD.colheaders;
```

Exercices de TP

Création de m-files

Les m-files permettent d'enregistrer les scripts sous forme de fichiers-texte et servent en particulier à définir de nouvelles fonctions (une grande partie des fonctions prédéfinies de MATLAB sont stockées sous forme de m-files).

Les m-files peuvent être crées par n'importe quel éditeur. Dans les versions récentes de MATLAB, il existe un petit éditeur intégré que l'on peut appeler à partir du menu file ou à partir de la barre de menu de la fenêtre de commande.

Exemple :

Dans la fenêtre de l'éditeur tapez les lignes suivantes :

```
% script - essai . m

a = .5;

b = pi;

c = a * b
```

Sauvez le fichier dans le répertoire de travail sous le nom de essai.m.

Exécution d'un m-file

Pour exécuter le script contenu dans un m-file et Il suffit de taper le nom de ce m file dans la fenêtre de commande suivi de < entrer >

```
>> essai
```

Exercices de révision

(On a déjà vu ce genre d'exercices pendant les séances de TP)

Exercice 1

Faire un programme Matlab qui calcule la somme suivante

$$\sum_{i=3}^{n} i$$

Solution:

```
n=input('donner la valeur de n');
s=0;
for i=3:n
s=s+i;
end
disp('la somme s est: '),s
%disp(['la somme s est: ',num2str(s)]
```

Exercice 2

Faire un programme sous Matlab qui calcule la somme suivante :

S=1+2/2! +3/3!+... on arrête le calcule quand S>2.5

Solution:

```
clear all
s=1;i=1,f=1;
while s<=2.5
i=i+1
f=f*i;
```

```
s=s+i/f
end
```

Exercice 3

Faire un programme sous Matlab qui résout le problème suivant :

1. y = x si x<0 2. y=x2 si x≥0

Solution

```
clear all
x=input('introduire la valeur de x ');
if x<0
y=x;
else
y=x^2;
end
disp('la valeur de y est: '),y
```

Notes : Les exercices suivants sont sans solutions

L'étudiant doit veillez à les résoudre

Exercise 4

Programmer les fonctions suivantes sous Matlab et indiquer ce qu'elles réalisent

```
function r = ps1(U,V)

r = U * V' ;
```

```
function r = ps2(U,V)

n = size(U,2);

r = 0. ;

for i = 1:n,

 r = r + U(i) * V(i);

end
```

```
function Y=insere(X,a)

[m,n] = size(X);

Y=a*ones(1,2*n);

Y(2 : 1+m , n+1 : end) = X;
```

Exercice 5

Ecrire un script Matlab où l'on définit le vecteur N = 1 :2 :12, puis on calcule le factoriel (le factoriel de l'entier k est 1*2…*k) de chacun des éléments de N à l'aide de trois méthodes :

1. en utilisant la fonction built-in factorial de Matlab,

2. en utilisant la boucle for,

3. en utilisant la boucle while.

Exercice 6

Dans un premier script, data1.m, saisir la matrice A et le vecteur x suivants :

A = [1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16] et x = [−5 −10 −15]

Dans un autre script, trait1.m, appeler le script data1.m pour charger les matrices A et X, puis saisir les instructions suivantes :

D1 = diag(A)

D2 = diag(diag(A))

DX = diag(x)

D3 = diag(A,2)

D4 = diag(diag(A,2))

D5 = diag(diag(A,2),2)

Exercice 7

- Définir le vecteur x = [0 pi/10 2pi/10 …. 2pi],

- Calculer les vecteurs y1 = sin(x) et y2 = cos(x) correspondants au vecteur x,

- Tracer la fonction sinus avec plot(x, y1),

- Mettre un quadrillage de fond par la fonction grid on (inverse grid off),

- Tracer sur le même graphique la fonction y2 = cos x (fonction hold on, inverse hold off),

- Taper figure pour ouvrir une nouvelle fenêtre sans fermer la première, puis tracer y = exp(cos (x)).

Exercice 8

Reproduire les commandes suivantes en ligne de commande et expliquer le résultat obtenu.

x=linspace (0 , 2* pi , 1 0 0) y=sin (x) z=cos (x) plot (x , y) plot (x , z) plot (x , y , x , z) clf hold on plot (x , y)	plot (x , z) clf hold on plot (x , z) plot (x , y , ' r ') clf plot (x , y , ' b−− ' , x , z , ' * ')

Exercice 9

1-Ecrivez un script "angles.m" stockant dans un fichier "angles.txt", pour 10 valeurs d'un angle alpha variant de 0 à 180°, l'angle, son sinus et son cosinus, sous la forme d'un tableau à 3 colonnes

2-Ecrivez un script "produit.m" permettant de lire le fichier contenant le tableau, calculant le produit sinus*cosinus, affichant celui-ci à l'écran, et traçant la courbe correspondante