UNIVERSITE M. MAMMERI DE TIZI-OUZOU
FACULTE DU GENIE DE LA CONSTRUCTION
DEPARTEMENT DE *GENIE MECANIQUE*

TRAVAUX PRATIQUES

INFORMATIQUE 3

MATLAB

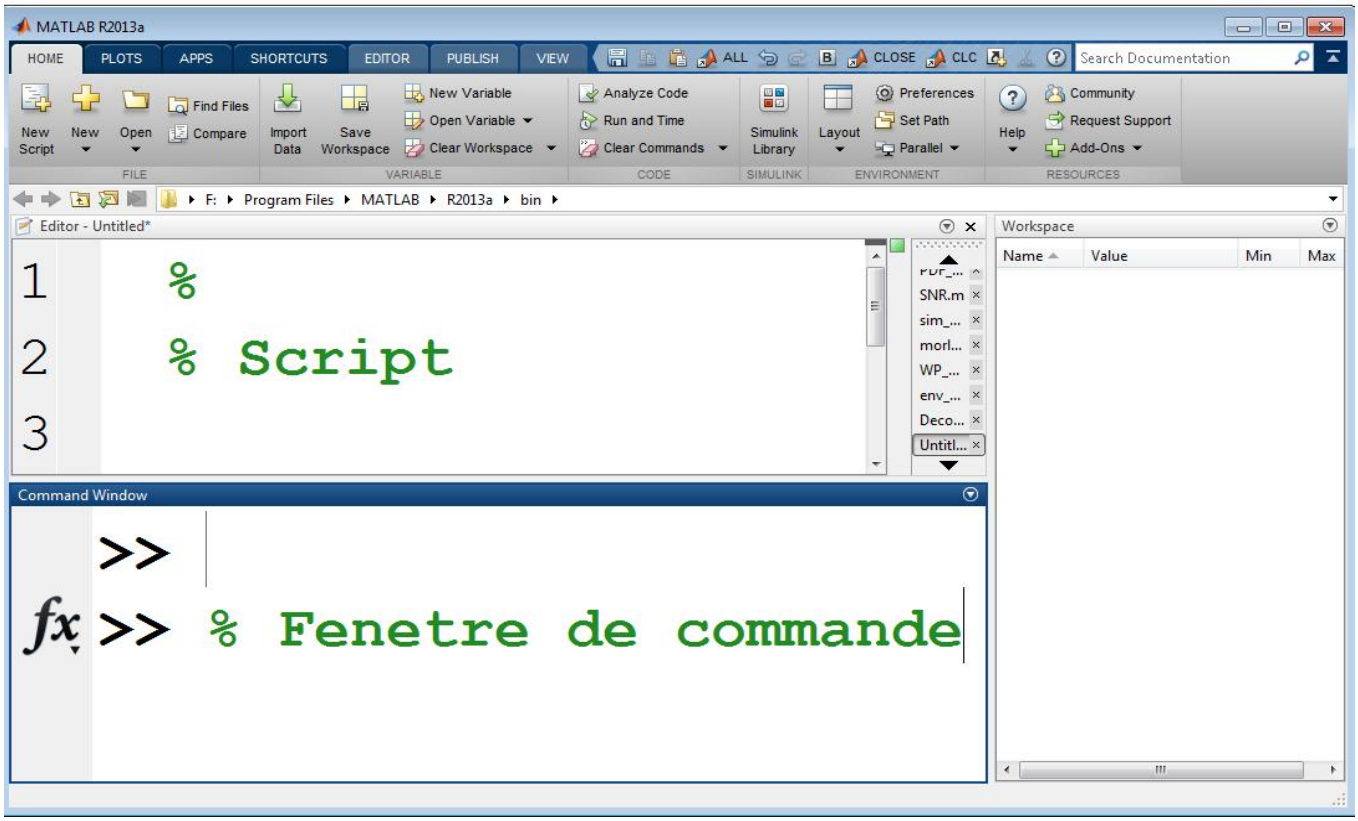
2018/2019

L2 Génie Mécanique - S3

M. BELAID Kamel

1- ENVIRONNEMENT MATLAB

Matlab (abréviation de "**Matrix Laboratory**"), est un environnement informatique conçu pour le calcul matriciel. L'élément de base est une matrice dont la dimension n'a pas à être fixée. Avec ses fonctions spécialisées, Matlab peut être considéré comme un langage de programmation adapté pour les problèmes scientifiques.



Cette fenêtre assez simple d'utilisation comporte un menu très standard. En dessous du menu, on distingue trois fenêtres distinctes. En haut à gauche, il s'agit de l'éditeur. Juste en dessous, c'est l'invite de commandes Matlab dans laquelle on va taper les commandes et lancer les programmes (**Command Window**). La partie droite, on a l'historique des commandes exécutées (**Workspace**).

Matlab est un langage interprété. Il n'est pas nécessaire de compiler un programme avant de l'exécuter et toute commande tapée dans la fenêtre de commande est immédiatement exécutée. Il existe deux modes de fonctionnement :

- **mode interactif** : Matlab exécute les instructions au fur et à mesure qu'elles sont données par l'usager.
- **mode exécutif** : Matlab exécute ligne par ligne un fichier 'm' (programme ne langage Matlab).

L'interface utilisateur de MATLAB affiche les fenêtres suivantes :

- **Fenêtre de commande (Command Window)** : dans cette fenêtre, l'usager donne les instructions et Matlab retourne les résultats.

">>" symbole/prompt apparaissant à gauche et indiquant que l'interpréteur est prêt à recevoir une commande.

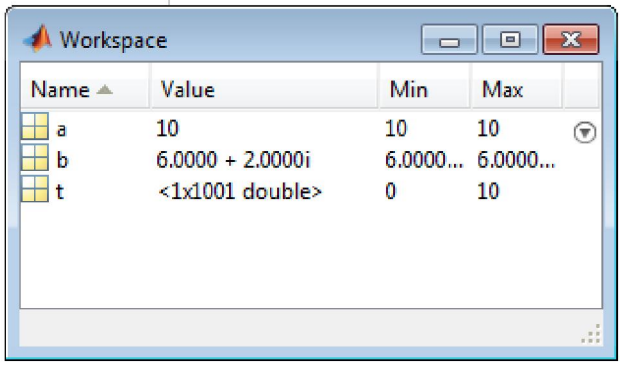
```
>>  commande
résultat (Affichage du résultat)
>>  interpréteur disponible

>>  commande ;  %  Le point-virgule provoque l'absence de l'affichage du résultat
>>  interpréteur disponible
```

- **Fichiers .m** (Scripts) : ce sont les programmes écrits par l'usager en langage Matlab. Ce sont des fichiers texte dont l'extension est .m . Ils sont exécutés par Matlab en tapant simplement leur nom et contrairement aux fonctions (ou en appuyant sur la touche F5 sur le clavier).

Ces fichiers contiennent une suite d'instructions. Ces instructions pourraient tout aussi bien être exécutées directement dans l'invite de commande Matlab. On peut donc y effectuer des opérations d'entrées/sorties, des calculs, exécuter des commandes et des fonctions, . . .

- **Espace de travail (Workspace)** : La zone de travail de MATLAB comprend l'ensemble des variables accumulées pendant une session de MATLAB et enregistrées dans la mémoire (figure ci-contre).



- **Help** : permet d'obtenir de l'aide sur une commande (appuyer sur la touche F1 pour y accéder).

>> ***help*** *:* permet d'obtenir l'aide de l'aide et donne une liste thématique ;
>> ***help*** *nom de fonction* : donne la définition de la fonction désignée et des exemples d'utilisation ;
>> ***lookfor*** *sujet* : donne une liste des rubriques de l'aide en ligne en relation avec le sujet indiqué
La bonne utilisation de l'aide en ligne est fondamentale pour travailler correctement avec Matlab.
Si l'on souhaite obtenir de l'aide sur certaines fonctions dont on connait le nom (par exemple inv), on écrit :

```
help inv
inv    Matrix inverse.
inv(X) is the inverse of the square matrix X.
    A warning message is printed if X is badly scaled or
nearly singular.
```

Commentaires : Le symbole **%** introduit un commentaire, celui-ci n'est pas évalué par l'interpréteur.

2- DONNEES

Les **constantes spéciales** prédéfinies dans Matlab :

NaN : Not-A-Number
inf : l'infini
ans : la dernière réponse en date
i ou **j** : le nombre imaginaire
pi : le nombre π

Pour i et j, il est fortement déconseillé de les utiliser (en général, on les utilise comme indices de boucles usuels), même si on ne manipule pas des complexes.

Les **opérateurs logiques** : Voici la liste des opérateurs logiques servant entre autres à écrire les conditions :

==	égal à
~=	différent de
<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieur ou égal à
&	et
|	ou
~	non

Tableau 1

2.1- Vecteurs

Pour définir un vecteur dont les valeurs sont uniformément reparties, il existe deux méthodes :
Vec=début : pas : fin ;
vec=**linspace**(debut,fin,nombre de valeurs) ;

Pour définir un vecteur ou une matrice élément par élément, il faut procéder comme suit :
– Les crochets **[]** servent à encadrer la matrice ;
– un **espace** ou une virgule « **,** » sépare deux éléments d'une même ligne ;
– un point-virgule « **;** » sépare les lignes entre elles.
NB. Matlab distingue les majuscules et les minuscules ;

- ***Vecteur ligne***

```
v=[1 2 3 4 5] ;                 % vecteur v= 1 2 3 4 5
v=linspace(0,5,10) ;            % vecteur entre 0 et 5 qui aura 10 valeurs
v=[1:5];                        % résultat identique à la ligne précédente (incrément de 1 par défaut)
v= [1:1:5];                     % idem incrément spécifié
```

- ***Vecteur transposé***

```
v=[1 2 3 4 5] ';                % vecteur 5*1
v=[1:5]';
v= [1:1:5]';
w=v';
```

- *la recherche d'éléments dans unvecteur*

```
v1(i) ;                         % ième élément ATTENTION le premier est à i=1
v(2 : 5)                        % donne les éléments du vecteur v entre 2 et 5
v<2                             % donne les éléments de v qui sont inférieurs à 2 (donne 1 0 0 0 0)
v>=5                            % donne les éléments de v qui sont supérieurs ou égaux à 5 (donne 0 0 0 0 1)
find(v==2)                      % donne l'emplacement des éléments du vecteur v qui sont égaux à 2 (donne 2)
```

(NB. **un seul =** : affectation ; **deux ==** : égalité)

```
length(v)                       %dimension du vecteur (donne 5)
mean(v)                         %moyenne du vecteur (donne 3)
```

2.2- Matrices

```
A1=[1 2 3; 4 5 6; 10 11 12];    %matrice rectangle 3*3
A2=[7 8 9
4 5 6
7 8 9];
A3=A2'                          % transposition
A4=A2*5                         % multiplication des éléments de la matrice A2 par un scalaire (ici 5)
size(A);                        %dimension de la matrice (donne 3x3)
inv(A);                         %inverse de la matrice
```

- *La recherche d'éléments dans un vecteur*

A1(i,j); % élément ligne i colonne j ; Ex. A1(3 , 2)= 11

A1(:,n); % nième colonne , exemple A1(: , 3) =

3
6
1

A1(p,:); % pième ligne , exemple A1(1 , :) =

1	2	3

A1(i:j,:); % sous matrice des lignes i à j , Ex. A1(2 :3 , :) =

4	5	6
10	11	12

A1(i:j,k:l); % sous matrice des lignes i à j colonnes k à l ,Ex. A1(1 :3 , 2:3) =

2	3
5	6
11	12

find(A1==i) % donne l'emplacement des éléments de la matrice A1 qui sont égaux à i , Ex. find(A1==2)=4

% le comptage se fait selon les colonnes

- *Matrices particulières*

ones(i,j) % matrice de uns de dimension i par j . Ex : ones(2,3)
zeros(i,j) % matrice de zéros de dimension i par j. Ex : zeros(5,1)
eye(i) % matrice identité de dimension i. Ex : eye(4)

- *Multiplication et division des matrices*

A1*A2 % multiplication matricielle
A1.*A2 % multiplication élément par élément

L'opérateur * désigne le produit de deux matrices (comme vous l'avez vu en maths), tandis que .* désigne le résultat du produit terme à terme de deux matrices de mêmes dimensions : la case (i; j) du résultat est le produit des cases (i; j) de chacune des deux matrices de départ.

A1/A2 % division matricielle
A1.*A2 % division élément par élément
(NB. Ceci s'applique aussi pour les vecteurs)

symbole	définition
[]	définition matricielle et concaténation
;	séparateur de colonne
()	extraction/insertion d'un élément
'	transposition
+	addition
-	soustraction
*	produit matriciel
\	division à gauche
/	division à droite
^	puissance
.*	produit élément par élément
.\	division à gauche élément par élément
./	division à droite élément par élément
.^	puissance élément par élément

Tableau 2

- *Suppression des données :* **clear**

>> **clear** % supprime toutes les variables
>> **clearA** % supprime la variable A

- *Sauvegarde des données :* **save, load**

save **:** permet de sauver tout ou partie de l'espace de travail sous forme de fichiers binaires appelés "fichiers .**mat**". Ces commandes permettent d'intervenir directement sur l'environnement de travail.

>> **save** *nom_fichier A B C* : sauve les variables A B et C dans le fichier nom_fichier

>> **save** : enregistre la totalité de l'espace de travail dans le fichier *matlab.mat*;

>> **save** *nom de fichier* : l'espace de travail est enregistré dans le fichier *nom de fichier* ;

>> **save** *nom de variable . . . nom de variable* : enregistre les variables indiquées (et les objets qui leurs sont associés) dans un fichier .mat qui porte le nom de la première variable ;

>> **save** *nom de fichier nom de variable . . . nom de variable* : enregistre les variables dans le fichier dont le nom a été indiqué.

load : permet d'ajouter le contenu d'un fichier **.mat** à l'espace de travail actuel ;

>> **load** *nom_fichier* : récupération du fichier *nom_fichier.*

3- GRAPHISME

Tout tracé avec Matlab, s'effectue dans une fenêtre graphique que l'on crée par la commande figure ou quand on exécute une commande de dessin (plot …). On peut créer autant de fenêtres graphiques que l'on veut celles-ci étant numérotées de 1 à N au fur et à mesure de leur création. La fenêtre graphique par défaut et la dernière qui a été créée par figure ou la dernière activée par une commande de dessin ou sélectionnée avec la souris.

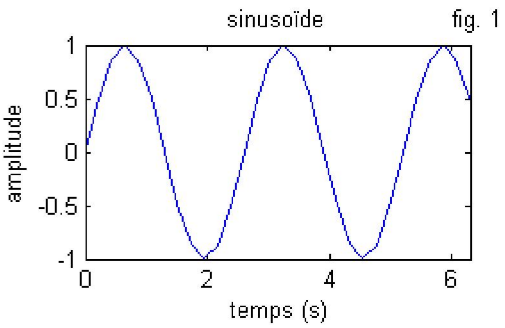
figure % crée une fenêtre graphique qui devient la figure par défaut,
figure(n) **%** crée une fenêtre graphique numéro n qui devient la fenêtre active

3.1- Graphes 2D

• ***plot***

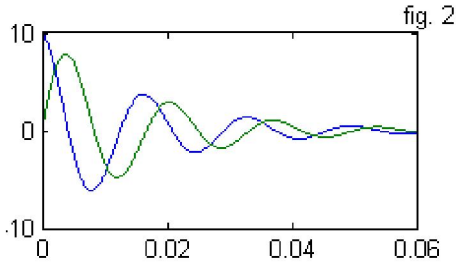
- ***Exemple 1 (fig. 1)***

```
t = linspace(0,2*pi,30);
x = sin(2*pi*5*t);
plot(t,x);        % dessin de x en fonction de t, interpolation linéaire
                  % entre les points.
title('sinusoïde');                 % titre de la figure
xlabel('temps (s)');                % commentaire sur l'axe x
ylabel('amplitude');                % idem axe y
```



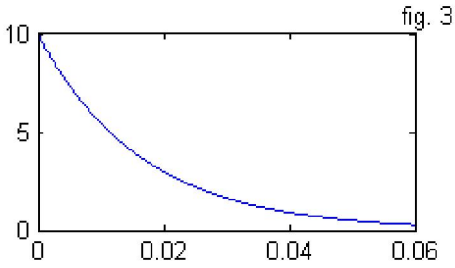
- ***Exemple 2 (fig. 2)***

```
t=0 :0.01e-3 :0.06 ;
y=10*exp(-60*t).*cos(120*pi*t) ;   % on définit la première fonction :
                                   %y(t) = 10 e^{-60t} cos (120 π t)
z=10*exp(-60*t).*sin(120*pi*t) ;   % on définit la deuxième fonction :
                                   %z(t) = 10 e^{-60t} sin (120 π t)
plot(t,y,t,z)   % permet de dessiner deux courbes dans une même figure
ou bien
plot(t,y), hold on   % désactivation par hold off
```



Un plot provoque l'effacement du dessin précédent (par défaut), on peut superposer des dessins en mettant le commutateur hold on.

```
hold off      % on quitte le hold on pour dessiner la prochaine courbe à part
plot(t,z),    % le numéro par défaut de la figure est 1
a=10*exp(-60*t) ;      % a(t) = 10e^{-60t}   (fig. 3)
figure(2), plot(t,a)   % on dessine la fonction a(t) dans une figure de
                       % numéro 2
```

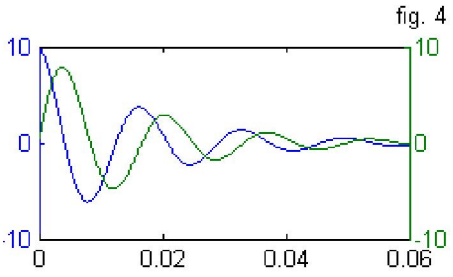


Les axes sont définis automatiquement ; on peut choisir les bornes des coordonnées du graphe à l'aide de la fonction **axis** :

```
axis([xmin,xmax,ymin,ymax]);
```

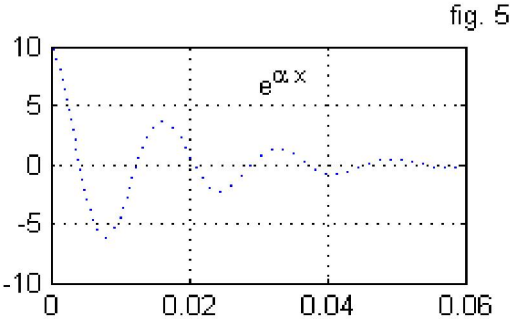
dans le cas où deux courbes sont représentées dans la même figure dont leurs échelles sont très différentes on utilise la fonction plotyy

```
plotyy(t,y,t,z)   % permet de représenter l'échelle d'une courbe à gauche et
                  %l'échelle de l'autre à droite (fig. 4).
```



- *Les couleurs et les types de lignes*

```
plot(t,y,' :')    % ligne pointillée (fig. 5)
plot(t,y,'b :')   % ligne pointillée et de couleur bleue
plot(t,y,'o')     % idem pas d'interpolation, chaque point marqué
                  %par o (voir tableau 3)
grid              % affiche une grille (fig. 5)
```



valeur	correspondance	valeur	correspondance
-	ligne continue	r	rouge
--	ligne discontinue	g	vert
:	ligne pointillée	c	cyan
+	signes plus	m	magenta
o	cercles	y	jaune
*	astérisques	k	noir
.	points	w	blanc
x	croix		
s	carrés		
d	diamant		
p	pentagramme		
^	triangles vers le haut		
v	triangles vers le bas		
<	triangles vers la gauche		
>	triangles vers la droite		

Tableau 3

- *Ajout du texte sur les courbes*

text(x,y,'texte') % place *texte* à la position x y dans la fenêtre
Ex. text(0.02,7,'e^{\alpha x}') % place $e^{\alpha x}$ à la position x=2
% et y=2 dans la fenêtre ***(fig. 5)***
gtext('texte') % place *texte* à la position définie avec la souris

Une fenêtre graphique peut être subdivisée en plusieurs tracés,
subplot(n,p,q) % subdivision en n*q dessin et
% sélectionne du qième
Exemple ***(fig. 6)***:
x=[-2 :0.1 :2] ; y=exp((-x.^2)/2) ;
subplot(2,2,1), plot(x,y) % courbe continue
subplot(2,2,2), **stem**(x,y) % diagramme en bâtons
subplot(2,2,3), **bar**(x,y) % diagramme à barres
subplot(2,2,4), **stairs**(x,y) % diagramme en escaliers

fig. 6

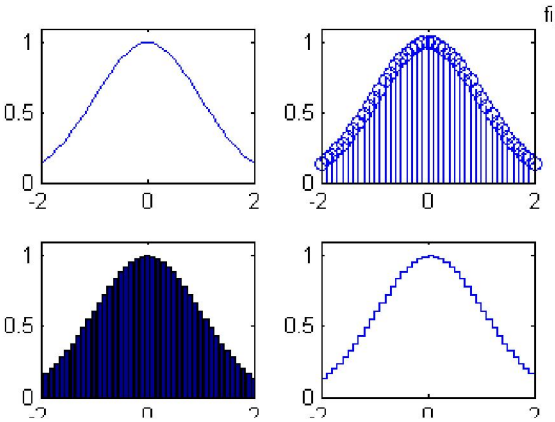
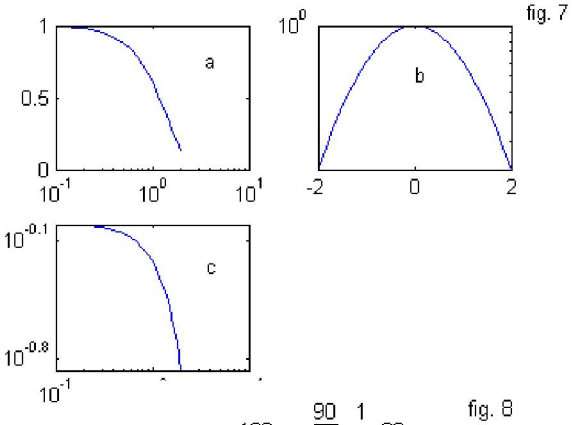


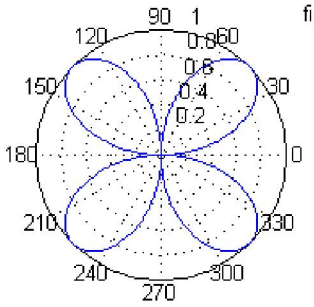
fig. 7



- *Axes logarithmiques*

semilogx(x,y) % x en logarithme et y linéaire ***(fig. 7a)***
semilogy(x,y) % x linéaire et y en logarithme ***(fig. 7b)***
loglog(x,y) % x en logarithme et y en logarithme ***(fig. 7c)***

fig. 8

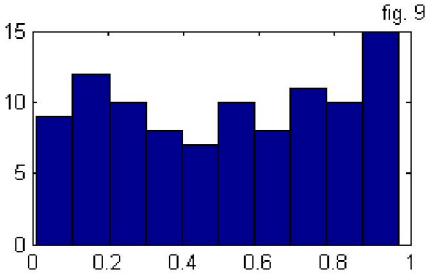


- *Autres graphiques*

polar % représentation en coordonnées polaires ***(fig. 8)***
t=0:0.01 :2*pi ;
polar(t,abs(sin(2*t)))

hist % histogramme ***(fig. 9)***
>>x=-2.9 :0.01 :2.9 ;
y=rand(100,1); find(y<=0.1) ;
hist(y)

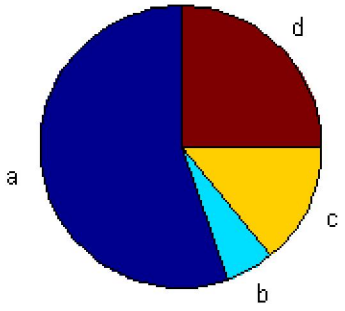
fig. 9



pie % diagramme circulaire ***(fig. 10)***

```
x=[100 10 25 45];
etiquettes={'a','b','c','d'};
pie(x,etiquettes)
pie3(x,etiquettes)
```

fig. 10



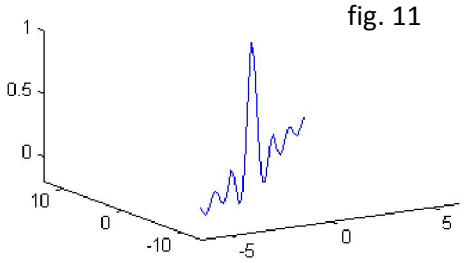
3.2 Graphes 3D

- *Courbes 3D*

• ***plot3***

```
x=-2*pi :pi/10 :2*pi ;
y=2*x ;
z=sinc(x-y)
plot3(x,y,z)  % dessine z en fonction de x et y (fig. 11)
```

fig. 11



- *Surfaces*

```
[x,y,z]=peaks(30) ;
```

mesh(x,y,z) % surface sous forme de grillage ***(fig. 12)***

surf(x,y,z) % les mailles du grillage en couleur ***(fig. 13)***

contour(x,y,z) % donne la projection de la surface sur le plan % oxy ***(fig. 14)***

fig. 12

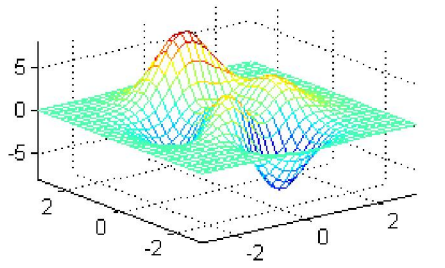


fig. 13

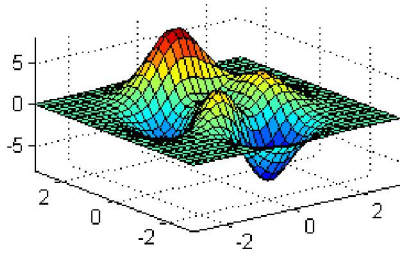
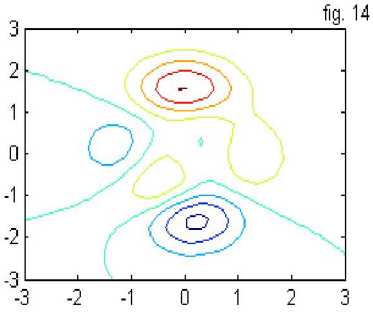


fig. 14



4- BOUCLES ET INSTRUCTIONS DE CONTRÔLE

4.1- Les boucles

Il en existe de deux types en Matlab : les boucles **for** et les boucles **while**. Les boucles for s´écrivent de la façon suivante :

```
for  i=deb :pas :fin
            . . .      (instructions à exécuter)
           . . .
end
```

Quand le pas (qui est un entier relatif) n'est pas précisé, il est fixé à la valeur 1 par défaut.

Il existe une autre manière d'écrire une boucle for dont l'indice de boucle parcourt les éléments d'un vecteur v :

```
for i=v
   . . .
   . . .
end
```

- ***Exemple 1***

```
for n=1 :5
for m=1 :4
A(n,m)=n^2+m^2;
end
end
```

```
>>A =
2   5   10   17
 5   8   13   20
10   13   18   25
17   20   25   32
26   29   34   41
```

- ***Exemple 2***

```
n=0.1
fori=1:5
for m=1 :4
A(i,m)=n^2+m^2;
end
   n=n+0.1;
end
```

```
>> A =
    1.0100   4.0100   9.0100   16.0100
    1.0400   4.0400   9.0400   16.0400
    1.0900   4.0900   9.0900   16.0900
    1.1600   4.1600   9.1600   16.1600
    1.2500   4.2500   9.2500   16.2500
```

Dans ce deuxième exemple, la variable n prend des valeurs décimales (n=0.1,0.2,0.3,0.4,0.5) donc l'écriture A(n,m)=n^2+m^2 n'est juste car les indices de la matrice A ne sont pas repérables par des nombres décimaux (il faut utiliser des entiers), c'est pour ça qu'un conteur i est créé pour la variable n et à chaque fois on ajoute le pas 0.1 à l'ancienne valeur de n.

- ***Exemple 3***

```
v=[-1 3 0 4] ; x=1 ;
for k=v
    x=x+k;
end
```

Les boucles **while** s'écrivent de la façon suivante :

```
while  condition
            . . . (instructions à exécuter)
            . . .
end
```

- ***Exemple 1***

```
x=1
while x<14
      x=x+5;
end
```

ce programme exécute la boucle tant que la condition x<14 est toujours vérifiée.

- ***Exemple 2***

```
n=0
whilen<10
      n=input('donner un nombre supérieur à 10 : )
end
```

ce programme s'exécute jusqu'à ce que le nombre n soit >= 10.
La fonction **input** permet de renvoyer le texte écrit entre les parenthèses (donner un nombre supérieur à 10) dans command window (CW). Le nombre tapé sera affecté à la variable n.

4.2- Les instructions de contrôle

Pour effectuer un test, on peut utiliser la combinaison classique **if** :

if expression 1
. . . (Commandes à exécuter si expression 1 est vraie)
elseif expression 1
. . . (Commandes à exécuter si expression 2 est vraie)
else
. . . (Commandes à exécuter si aucune expression 2n'est vraie)
End

- ***Exemple 1***

```
n=input('donner un nombre positif :  ')
ifrem(n,3)==0       % rem donne le reste d'une division
disp('ce nombre est divisible par 3')       % disp : permet d'afficher ce qu'il y a entre parenthèse dans CW
elseif rem(n,5)==0
disp ('ce nombre est divisible par 5')
else
disp ('ce nombre n''est pas divisible par 3 ou par 5')
end
```

- ***Exemple 2***

```
n=input(' donner un nombre n :  ');
if (n==2)
disp('   vrai')
else
disp('   faux')
end
```

- ***Exemple 3***

```
num=input(' donner un nombre num = ');
n=3 ;
if num==1
A=ones(n)
elseif num==2
A=zeros(n)
elseif num==3|num==4        % | : ou logique
A=rand(n)                   % rand : donne des valeurs aléatoires entre 0 et 1
else
error('numéro d''exemple non prévu ')      % error : permet d'afficher comme message d'erreur le texte : numéro
                                            % d''exemple non prévu
end
```