Programmer avec Matlab

Pr. Bentbib

Faculté des Sciences et Techniques

Département de Mathématiques

Marrakech

Ces notes ne constituent pas un cours détaillé sur Matlab, mais plutôt un aperçu sur les notions principales qui servirons à utiliser Matlab d'une part comme outils aidant l'étudiant dans la résolution de problèmes de calcul scientifique et comme langage de programmation très adapté aux divers domaines scientifiques d'autre part.

L'étudiant pourra chercher lui-même les compléments d'information qui lui sont nécessaires avec l'outil d'aide de Matlab (le help).

Pourquoi Matlab ?

- **MATLAB**, son nom dérive de **MAT**rix **LAB**oratory car initialement fait pour les problèmes d'algèbre linéaire utilisant les matrices. Aujourd'hui c'est le logiciel le plus puissant pour faire du calcul scientifique.
- MATLAB a été élargie pour résoudre des problèmes dans divers domaines : Analyse de données, traitement de signale, optimisation et d'autres domaines du calcul scientifique.
- MATLAB est fort en graphisme et animation en **2-D** et **3-D**.

- MATLAB est un logiciel de calcul matriciel à syntaxe simple. Avec ses fonctions spécialisées, MATLAB peut être aussi considéré comme un langage de programmation adapté pour les problèmes scientifiques.

- MATLAB est un interpréteur: les instructions sont interprétées et exécutées ligne par ligne.

- MATLAB fonctionne en deux modes :

- 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 en langage MATLAB).

- Fenêtre Commande:

Dans cette fenêtre, l'usager donne les instructions et MATLAB retourne les résultats. On tape les instructions une ligne à la fois. Chaque ligne est exécutée immédiatement après la touche "Return".

Une ligne peut contenir plusieurs instructions séparées par des virgules (, ou ;). Des boucles FOR, WHILE, IF ... ELSE peuvent être sur plusieurs lignes.

- Fenêtres Graphique:

  MATLAB trace les graphiques dans ces fenêtre. Lorsque les fonctions graphiques sont appelées, la fenêtre Graphique s'ouvrira.

- Fichiers M:

  Ce sont des programmes en langage MATLAB (écrits par l'usager).

- Toolboxes: Ce sont des collections de fichiers M développés pour des domaines d'application spécifiques (Signal Processing Toolbox, System Identification Toolbox, Control System Toolbox, Analysis Toolbox , Robust Control Toolbox, Optimization Toolbox, etc.)

- FONCTION "HELP" :

  Pour obtenir de l'aide sur un sujet, une instruction ou une fonction, on tape help suivi par le sujet, l'instruction ou la fonction désirée.

- ESPACE DE TRAVAIL (Workspace)

  Les variables sont définies au fur et à mesure que l'on donne leurs noms et leurs valeurs numériques ou leurs expressions mathématiques. Les variables ainsi définies sont stockées dans l'espace de travail et peuvent être utilisées dans les calculs subséquents.

- INFORMATION SUR L'ESPACE DE TRAVAIL

  Pour obtenir une liste des variables dans l'espace de travail, on utilise les instructions suivantes:

1. who Affichage des variables dans l'espace de travail.
2. whos Affichage détaillé des variables dans l'espace de travail.

➢ ENREGISTRER DES VARIABLES DANS UN FICHIER

Pour enregistrer les variables de l'espace de travail dans un fichier, on utilise les instructions suivantes:

- save **Enregistrer toutes les variables dans un fichier matlab.m. Dans une session ultérieure, taper** load **pour ramener l'espace de travail enregistrée.**

- save **fichier1.m** x y z A X. **Enregistrer les variables x, y, z, A, X dans le fichier fichier1.m. Dans une session ultérieure, taper** load **fichier1 pour ramener les variables x, y, z, A, X dans l'espace de travail.**

➢ NOMBRES

Les nombres réels **peuvent être sous différents formats:**

5 1.0237 0.5245E-12 12.78e6 0.001234 -235.087

Les nombres complexes **peuvent être écrits sous forme cartésienne ou polaire: 0.5 + i*2.7 ou 1.25*exp(i*0.246)**

➢ FORMATS D'AFFICHAGE

Pour choisir le format d'affichage pour les nombres, on utilise l'instruction format:

format short 0.1234
format long 0.12345678901234
format short e 1.2341E+002
format long e 0.123456789012345E+002.

➢ OPÉRATIONS ARITHMÉTIQUES

+ Addition - Soustraction * Multiplication / Division à droite

\ Division à gauche ^Puissance

➢Quelques commandes de gestion de dossiers

- pwd ou cd :permet de connaître le dossier courant (Print Working Directory ou Current Directory)
- cd .. : remonte au dossier supérieur
- cd Rep : sélectionne le dossier inférieur nommé Rep
- ls (list) ou dir : donne la liste de tous les fichiers dans le dossier courant.
- what : donne la liste des fichiers .m dans le dossier courant.
- cd a: change de répertoir vers a:

Les MATRICES

- On définit une matrice A en donnant ses éléments:

Vecteurs et matrices

» help linspace LINSPACE Linearly spaced vector.

» LINSPACE(x1, x2, N) génère N points entre x1 et x2.

On peut définir un vecteur X en donnant la liste de ses éléments ou en donnant la suite qui forme le vecteur ou en utilisant une fonction qui génère un vecteur

- Le ";" à la fin de chaque instruction permet de ne pas afficher le résultat obtenu.

➢ EMPLOI DES INDICES

Les éléments d'un vecteur ou d'une matrice peuvent être adressés en utilisant les indices sous la forme suivante:

- t(10) **élément no. 10 du vecteur** t
- A(2,9) **élément se trouvant à ligne 2, colonne 9 de la matrice** A
- B(:,7) **la colonne 7 de la matrice** B
- C(3,:) **la ligne 3 de la matrice** B
- D(2:5,4:8) **extrait de la matrice** D **le bloc d'éléments dont la** ligne i **est comprise entre** 2 **et** 5 **et la** colonne j **comprise entre** 4 **et** 8

➢ OPÉRATIONS MATRICIELLES

Les opérations matricielles exécutées par MATLAB sont comme suit:

- B = A' **La matrice** B **est égale à la matrice** A transposée
- E = inv(A) **La matrice** E **est égale à la matrice** A inversée
- C = A + B **Addition**
- D = A - B **Soustraction**
- Z = X*Y **Multiplication**

- Z = X*Y Multiplication
- X = A\B Équivalent à inv(A)*B
- X = B/A Équivalent à B*inv(A)

OPÉRATION "ÉLÉMENT PAR ÉLÉMENT"

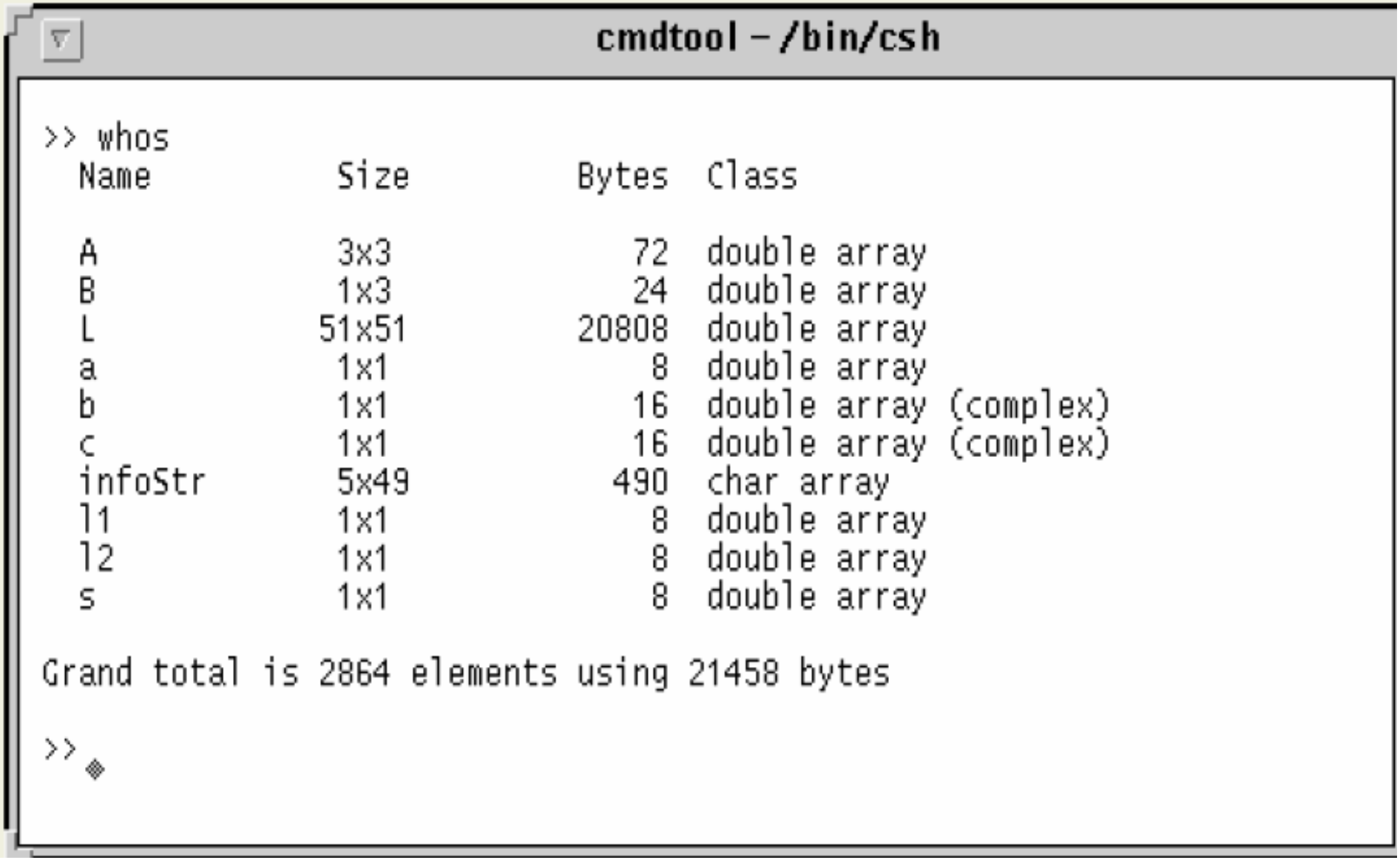
Les opérations "élément par élément" des vecteurs et des matrices sont effectuées en ajoutant un point (.) avant les opérations * / \ ^'

Exemple de définition de variables dans l'espace de travail

cmdtool – /bin/csh

```
>> a=1.23456

a =

    1.2346

>> b=-0.5+3.75i

b =

  -0.5000 + 3.7500i

>> c=a+b^2

c =

 -12.5779 - 3.7500i

>> A=[1.1 2.2 3.3;-3 5 -7;0 2 -4]

A =

    1.1000    2.2000    3.3000
   -3.0000    5.0000   -7.0000
         0    2.0000   -4.0000

>> B=[0.5 1.2 3.7]

B =

    0.5000    1.2000    3.7000
```

Exemple d'emploi de l'instruction whos



VECTEURS

On peut définir un vecteur x en donnant la liste de ses éléments:

```
>> x=[0.5 1.2 -3.75 5.82 -0.735]

x =

    0.5000    1.2000   -3.7500    5.8200   -0.7350
```

ou en donnant la suite qui forme le vecteur:

```
>> x=2:0.6:5

x =

    2.0000    2.6000    3.2000    3.8000    4.4000    5.0000
```

ou en utilisant une fonction qui génère un vecteur:

```
>> x=linspace(1,10,6)

x =

    1.0000    2.8000    4.6000    6.4000    8.2000   10.0000

>> y=logspace(1,3,7)

y =

   1.0e+03 *

    0.0100    0.0215    0.0464    0.1000    0.2154    0.4642    1.0
```

Au sujet de ";" à la fin de chaque ligne d'instruction, voir Remarque.

MATRICES

On définit une matrice A en donnant ses éléments:

```
>> A=[0.5 2.7 3.9;4.5 0.85 -1.23;-5.12 2.47 9.03]

A =

    0.5000    2.7000    3.9000
    4.5000    0.8500   -1.2300
   -5.1200    2.4700    9.0300
```

➢ Quelques fonctions prédéfinies

- ✓ **rand(n,m) : is an n-by-m matrix with random entries, chosen from a uniform distribution on the interval (0.0,1.0).**
- ✓ **eye(n) : is the n-by-n identity matrix.**
- ✓ **ones(n) : is an n-by-n matrix of ones.**

➢ Manipulation des matrices

- flipud(A) : flip up->down
- fliplr(A) : flip left->right
- rot90(A,2) :2 rotations de 90 degrés(sens trigo)
- diag(A): extrait le vecteur diagonale de A
- diag (diag(A)): la matrice diagonale de A
- triu(A) : extrait le triangle supérieur de A
- tril(A) : triangle inférieur de A

fonctions	signification
plot	representation graphique plane.
loglog	representation plane logarithmique.
semilogx **semilogy**	representation plane en semilog
polar	representation en coordonnées polaires
mesh	representation graphique en trois dimensions.
contour	contour plot.
title	titre de la figure.
xlabel	titre de l'axe des abscisses.
ylabel	titre de l'axe des ordonnées.

any	conditions logiques.
all	conditions logiques.
find	trouve les indices d'un tableau de valeurs logiques.
exist	cherche si les variables existent.
isnan	detecte si le résultat est un nombre.
finite	detecte si le résultat est infini.
isempty	detecte les matrices vides.
isstr	detecte les variables sous forme de lettres.
strcmp	comparaison des variables lettres.

- Opérateurs relationnels
  - == test d'égalité
  - ~= test de différence
  - >,<,>=,<= tests de comparaison

- Opérateurs logiques
  - && **ou** & : and (& pour un tableau)
  - || **ou** | : or (|| pour un tableau)
  - ~ not

Programmation avec Matlab

- Il est possible d'enregistrer une séquence d'instructions dans un fichier (appelé un M-file) et de les faire exécuter par MATLAB. Un tel fichier doit obligatoirement avoir une extension de la forme ".m". On distingue 2 types de M-file, les fichiers de scripts et les fichiers de fonctions.
- Un script est un ensemble d'instructions MATLAB qui joue le rôle de programme principal. Si le script est écrit dans le fichier de nom "nom.m" on l'exécute dans la fenêtre MATLAB en tapant "nom".

➢ Les fichiers de fonctions ont deux rôles:

- Ils permettent à l'utilisateur de définir des fonctions qui ne figurent pas parmi les fonctions incorporées de MATLAB et de les utiliser de la même manière que ces dernières (ces fonctions sont nommées fonctions utilisateur).

- Ils sont également un élément important dans la programmation d'applications où les fonctions jouent le rôle des fonctions et procédures des langages de programmation usuels..

➢ On définit la fonction fonc de la manière suivante:

function [v1, ..., vm] = fonc(s1, ..., sn)

% cette fonction calcule…

séquence d'instructions

- v1, ..., vm sont les variables de sortie de la fonction;
- s1, ..., sn sont les variables d'entrée de la fonction;
- séquence d'instructions est le corps de la fonction.
- Il est impératif que la fonction ayant pour nom fonc soit enregistrée dans un fichier de nom fonc.m sans quoi cette fonction ne sera pas visible par MATLAB

- Les lignes précédées du symbole % sont des lignes de commentaire. Les lignes de commentaire situées entre la ligne function ... et la 1-ere ligne d'instructions sont affichées si l'on demande de l'aide "help" sur la fonction.

➢Les Boucles:

- Les boucles permettent d'itérer les mêmes opérations plusieurs fois sans les réécrire dans le programme

✓*if …else …end* *Si … sinon …fin*

- La structure générale de **if** et **else** est la suivante:

```
>>  if expression logique 1 instructions A
    elseif expression logique 2 instructions B
    else expression logique 3 instructions C
    end
```

✓ *for* i=1:10 … *end* *De i=1 à 10*

for i=1:2:10 … *end* *Avec un pas de 2*

✓ *while* *true* … *end* *Tant que …*

```
while expression
    statements;
end
```

- Pour connaître la liste exhaustive, taper

  iskeyword

➢ Instruction **switch**

```
switch v
   case c1,
      séquence d'instructions 1
   case c2,
      séquence d'instructions 2
   otherwise
      séquence d'instructions
end
```

où v est une variable numérique ou une variable chaîne de caractères et les ci des constantes de même type que v.

✓ Exemple 1: v est une variable numérique

```
v=input('donner une valeur v=\n');
switch v
   case 1.1,
disp('cas numero 1')
   case 2.1,
disp('cas numero 2')
   otherwise
      disp('dernier cas')
end
```

✓ Exemple 2: v est une variable chaîne de caractères

```
v=input('donner une valeur v=\n');
switch v
   case {'oui','Oui','OUI','o','O'},
         disp('cas numero 1')
  case {'non','Non','NON','n','N'},
        disp('cas numero 2')
  otherwise
        disp('dernier cas')
end
```

✓Exemple de fonction

```
function y=temps(x)
% temps(X) affiche un message suivant le temps qu'il fait
% et retourne le paramètre d'entrée X changé de signe
  if length(x)>1 error('X doit être un scalaire'); end
  if x~=0
     disp('Hello, il fait beau')
  else
     disp('Espérons que demain sera meilleur!')
  end
y=-x;
```

➢Lecture et écriture de fichiers

✓**Entrée-sortie standard** : *disp, input*

```
N = input( 'Nombre de boucles  est N=\n'); % entrée interactive
disp(N)   % affiche la valeur de N
```

✓**Fichier de données** : *load, save*

save : ***écrit toutes les variables du workspace dans le fichier*** *matlab.m*

load : ***charge dans le workspace toutes les variables du fichier*** *matlab.m*

✓Fichier numérique : dlmread

- dlmread(‘nom_fichier’) lit les données numériques à partir de mon_fichier.

- dlmwrite(‘nom_fichier’,M) ecrit les données numériques de M dans mon_fichier.

```
>> A = rand(6,5);   A = floor(A * 100);
>> dlmwrite('myfile.txt', A);
>> dlmread('myfile.txt');
```

Voir aussi *textread, textscan, importdata …*
Exemple de formats : '%d' pour un entier,
'%f' pour un réel, '%c‘ pour un caractère.

`%c`	Sequence of characters; number specified by field width
`%d`	Base 10 integers
`%e, %f, %g`	Floating-point numbers
`%i`	Defaults to signed base 10 integers. Data starting with `0` is read as base 8. Data starting with `0x` or `0X` is read as base 16.
`%o`	Signed octal integer returned as unsigned
`%s`	A series of non-white-space characters
`%u`	Unsigned decimal
`%x`	Signed hexadecimal integer returned as unsigned
`[...]`	Sequence of characters (`scanlist`)

✓**Fichier texte** : *fopen, fclose*

fopen : ouverture d'un fichier

fclose : fermeture d'un fichier

fscanf : lecture formatée

fprintf : écriture formatée

fid = fopen('test.txt','w'); ***ouvre un fichier ou le créé***

fprintf(fid,'%s\n','vecteur x'); ***écrit dans le fichier de nom*** *test.txt.*

fid est sa référence pour *matlab.*

✓**Fichier son** : *wavread, wavwrite*

➢ Exemple:

✓ Ecriture dans le fichier exemple.txt

```
x = linspace(0,1,10); u = [x; exp(x)];
fid = fopen('exemple.txt','wt');
fprintf(fid,'%6.2f  %12.8f\n',u);
fclose(fid);
```

✓ Lecture à partir du fichier exemple.txt

```
fid = fopen('exemple.txt', 'r');
v = fscanf(fid, '%f  %f', [2 inf])
fclose(fid)
```

Script : Résolution d'une équation de second degré ax^2+bx+c=0

```
a=input('donner le coefficient  a:  ');
 if a==0
disp('C est une equation de premier degres');
b  =  input('donner le coefficient  b::  ');
c  =  input('donner le coefficient  c::  ');
fprintf('%f x + %f =0\n', b, c);
if b==0 & c==0
   disp('C est l equation 0=0');
elseif  b==0 & c~=0
    disp('equation impossible');
else
   s=-c/b;
 fprintf('l equation admet comme racine s=%f \n ', s);
end
```

```
else
    b  =  input('donner le coefficient  b::  ');
    c  =  input('donner le coefficient  c::  ');
     disp('résolution d une equation de second degres');
     fprintf('%f x^2+ %f x + %f =0\n', a, b, c);
       D=b^2-4*a*c;
 if D>0
   s1=(-b+sqrt(D))/2*a; s2=(-b-sqrt(D))/2*a;
   disp('l equation admet deux solutions s1 et s2');
   fprintf('s1=%f \n s2=%f \n', s1, s2);
 elseif D==0
   s=-b/2*a;
   fprintf('l equation admet une racine d ouble s=%f \n ', s);
 else
   disp('l equation n admet pas de solutions reelles');
 end
end
```

`dlmread('NomDeFichier','delimiteur')`	lecture du fichier
`dlmwrite('NomDeFichier',M,'delimiteur')`	écriture de M dans le fichier
`textread('NomDeFichier','format')`	lecture du fichier
`fid=open('NomDeFichier')`	ouverture du fichier NomDeFichier
`[A,count]=fscanf(fid,'format')`	lecture du fichier ouvert par open
`fprintf(fid,'format',données)`	écriture des données avec un format
`close(fid)`	fermeture
`fprintf('format',données)`	écriture des données avec un format
`fprintf('format',données)`	écriture des données avec un format

- Autre exemple de fonction matlab:

```
function [A,rang] = matale(T,n,m)
% Construit une matrice A de n lignes et m colonnes ayant des
%éléments entiers générés de manière aléatoire entre 0 et T.
% Calcule le rang de la matrice si l'appel est effectué avec 2
%arguments de sortie. Si la matrice est carrée, le paramètre m
%peut être omis.
 if nargin == 2
      A = fix(T*rand(n));
 else
      A = fix(T*rand(m,n));
 end
 if nargout == 2    rang = rank(A);    end
```

➢Tracer une courbe à l'aide de MATLAB

- ✓La commande **plot** permet de tracer un ensemble de points M_{ij} de coordonnées (x_i , y_i).
- ✓La syntaxe est **plot(x,y).** Le vecteur x contient les valeurs x_i en abscisse et y les valeurs y_i en ordonnée.

- On peut spécifier **la couleur du trait**, **le symbole du point** et **le type de trait** par :

plot(x,y,'cst')

plot(x,y,'**cst**')

Les possibilités sont les suivantes:

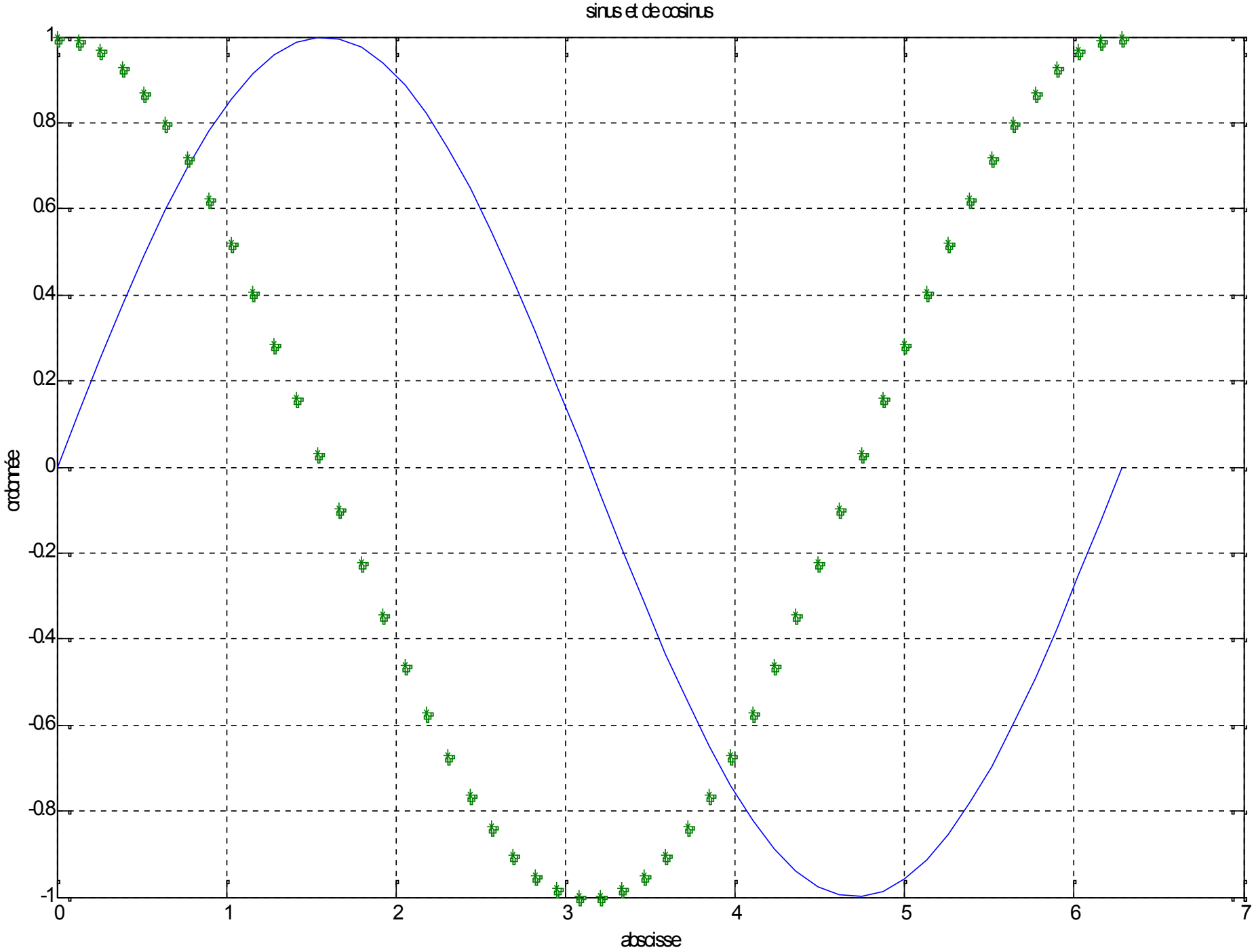
y	jaune	.	point	-	trait plein
m	magenta	o	cercle	:	pointillé court
c	cyan	x	marque x	--	pointillé long
r	rouge	+	plus	-.	pointillé mixte
g	vert	*	étoile		
b	bleu	s	carré		
w	blanc	d	losange		
k	noir	v	triangle (bas)		

Par défaut **c** = b, **s** = . et **t** = **-** : ce qui correspond à **un trait plein bleu** reliant les points entre eux

- On se propose de tracer les fonctions sinus et cosinus.

```
>> x = linspace(0,2*pi,20); y = sin(x); z = cos(x);

>> plot(x,y,x,z,'+'); title('sinus et de cosinus')

>> xlabel('abscisse'); ylabel('ordonnée'); grid;
```

On obtient la figure ci-dessous,


sinus et de cosinus
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
ordonnée
0
1
2
3
4
5
6
7
abscisse

➢Sauvegarder une figure

print permet de sauvegarder la figure d'une fenêtre graphique dans un fichier sous divers formats d'images.

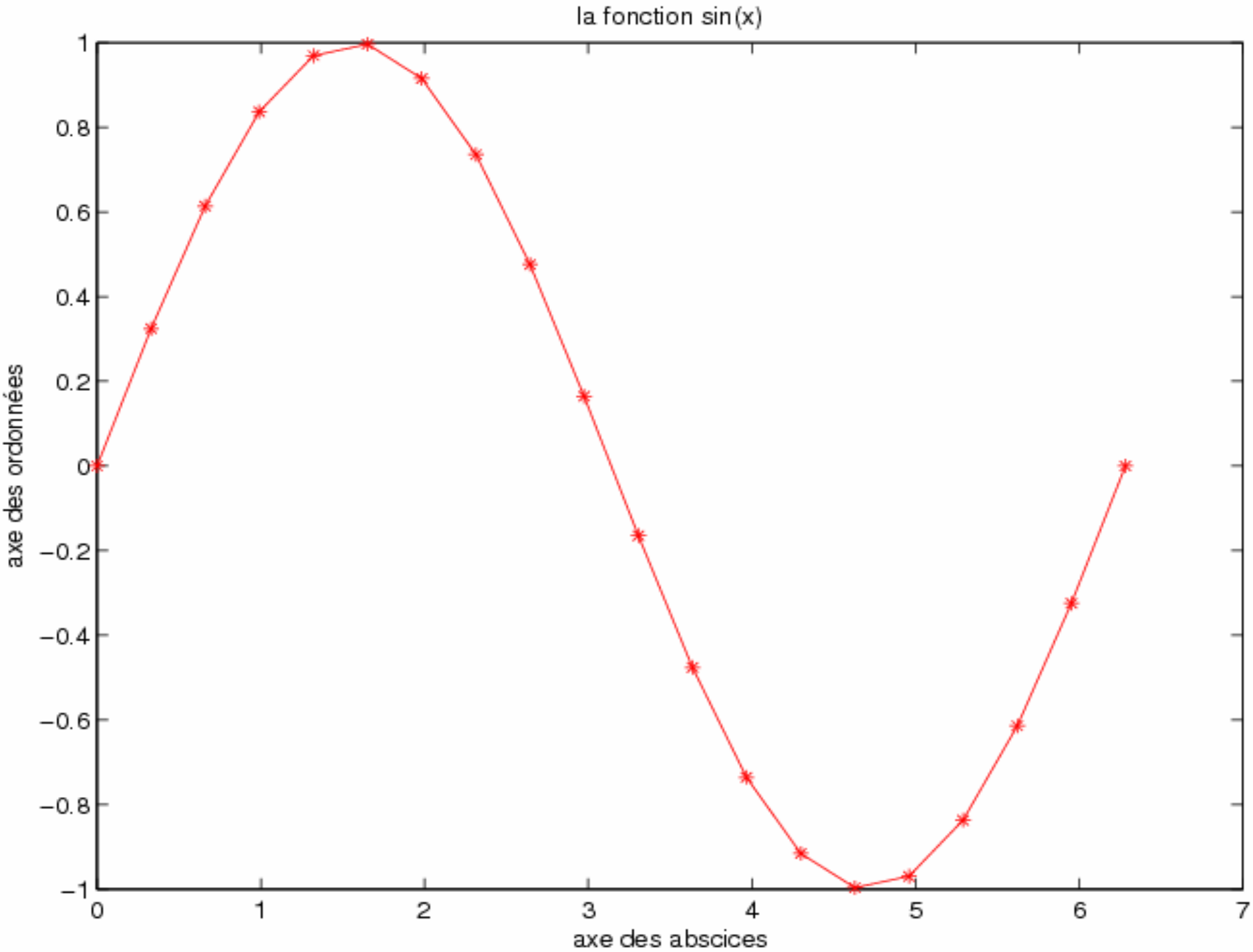
print -d<format> <nomfig>

<format> est le format de sauvegarde de la figure.

ps	:	PostScript noir et blanc
psc	:	PostScript couleur
eps	:	PostScript Encapsulé noir et blanc
epsc	:	PostScript Encapsulé couleur
jpeg	:	Format d'image JPEG
tiff	:	Format d'image TIFF

Exemple

```
u=linspace(0,2*pi,20); v=sin(u);
plot(u,v,'r*-');
xlabel('axe des abscices');
ylabel('axe des ordonnées');
title('la fonction sin(x)');
print -dpsc figure1.psc
```


la fonction sin(x)
axe des ordonnées
axe des abscices
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
1
2
3
4
5
6
7

➢Graphisme en 3-D

Pour la représentation d'une surface f(x, y), on a besoin de connaître les triplets de coordonnées (xi, yj, Zi,j), avec Zi,j=f(xi,yj), pour un certain nombre de points (xi,yj) où i=1, ..., n, j=1, ..., m.

On remarque que Z a la structure d'une matrice (Zi,j).

✓ meshgrid

[X,Y] = **meshgrid**(**x,y**)

Transforme le domaine spécifié par les vecteurs x et y en un maillage X et Y, pour évaluer des fonctions à deux variables.

X : est la matrice (**size(y),size(x)**) dont chaque ligne est le vecteur **x**.

Y : est la matrice (**size(y),size(x)**) dont chaque colonne est le vecteur **y**.

```
[X,Y] = meshgrid(1:3,5:8)
```

X =

1	2	3
1	2	3
1	2	3
1	2	3

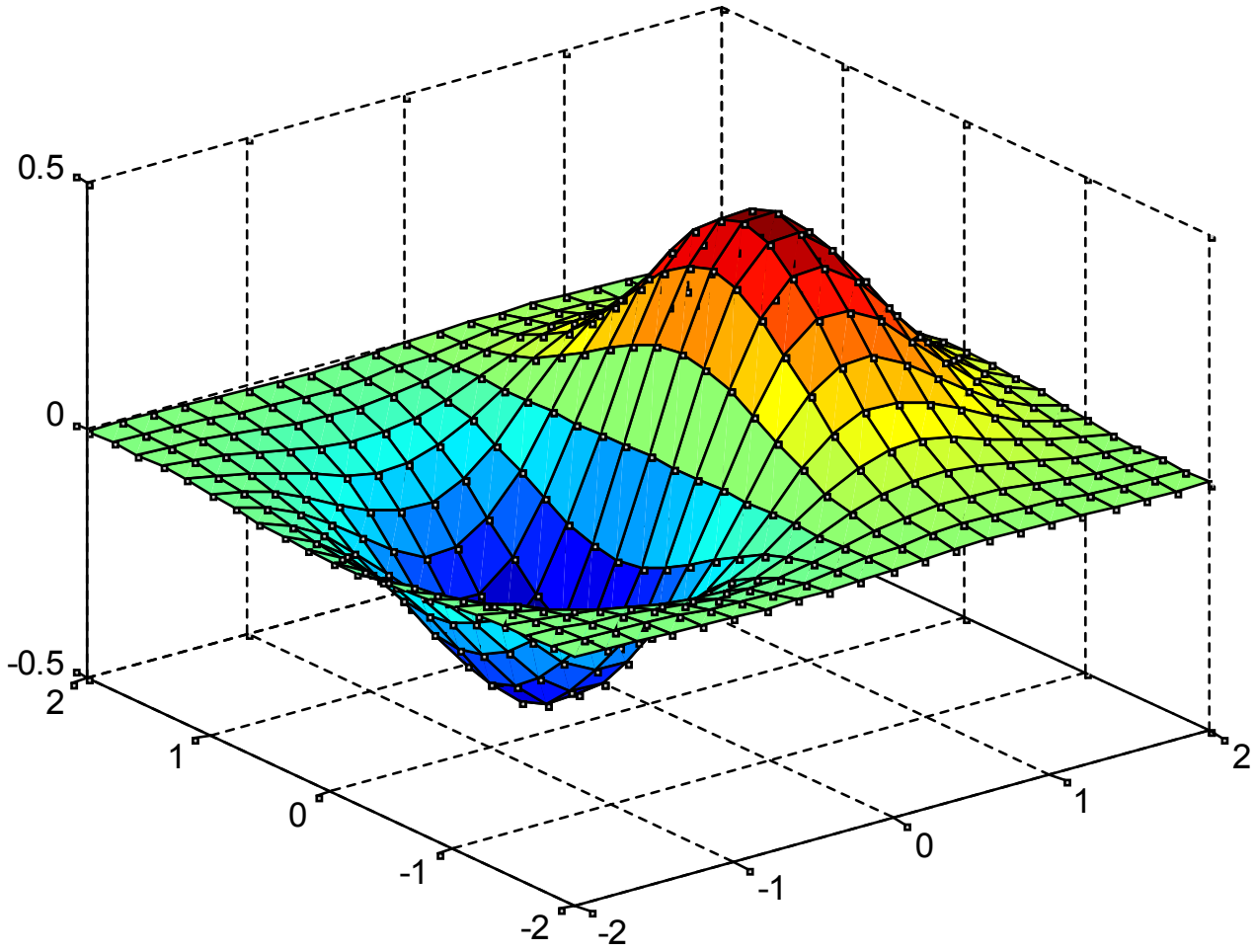
Y =

5	5	5
6	6	6
7	7	7
8	8	8

[X,Y] = meshgrid(-2:.2:2, -2:.2:2);

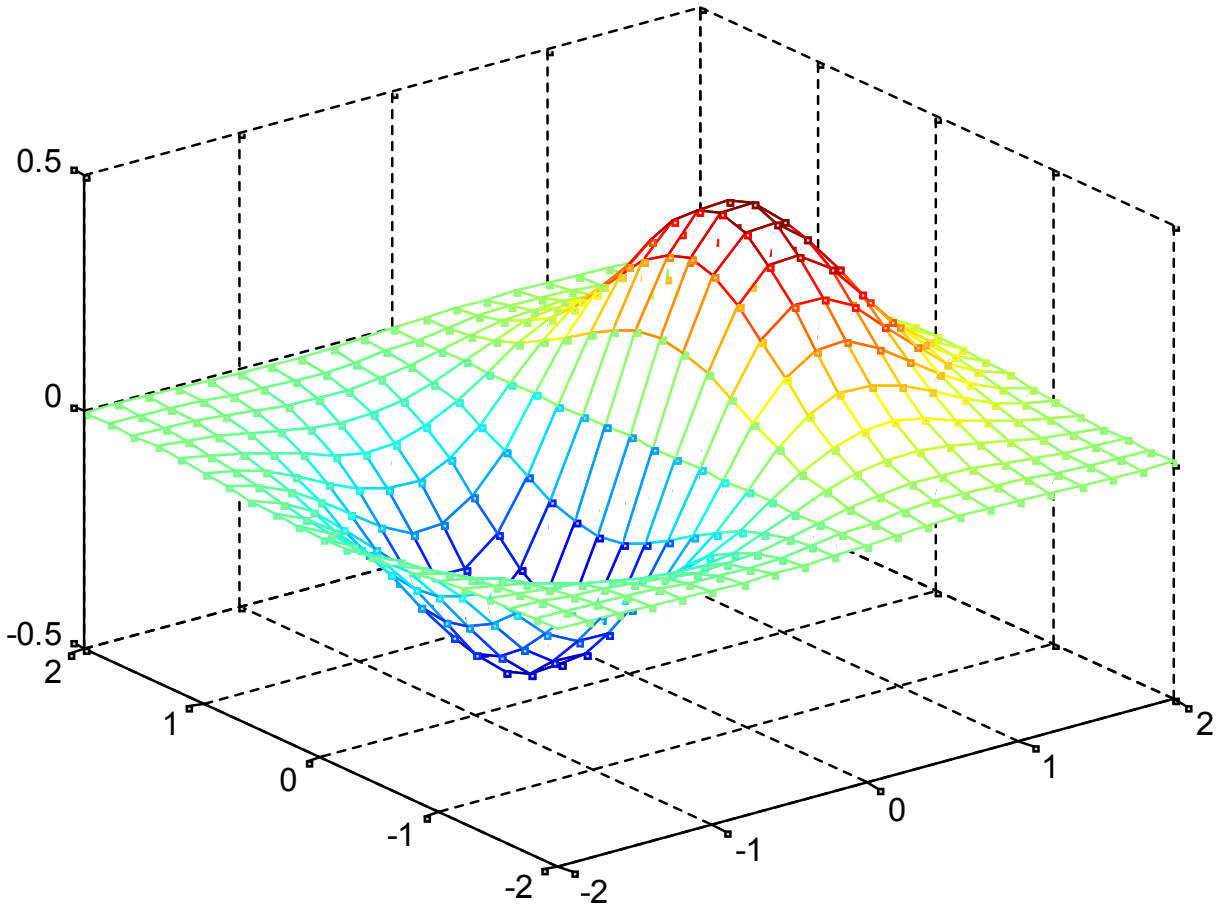
Z = X .* exp(-X.^2 - Y.^2); surf(X,Y,Z)

surf : Crée une surface en trois dimensions



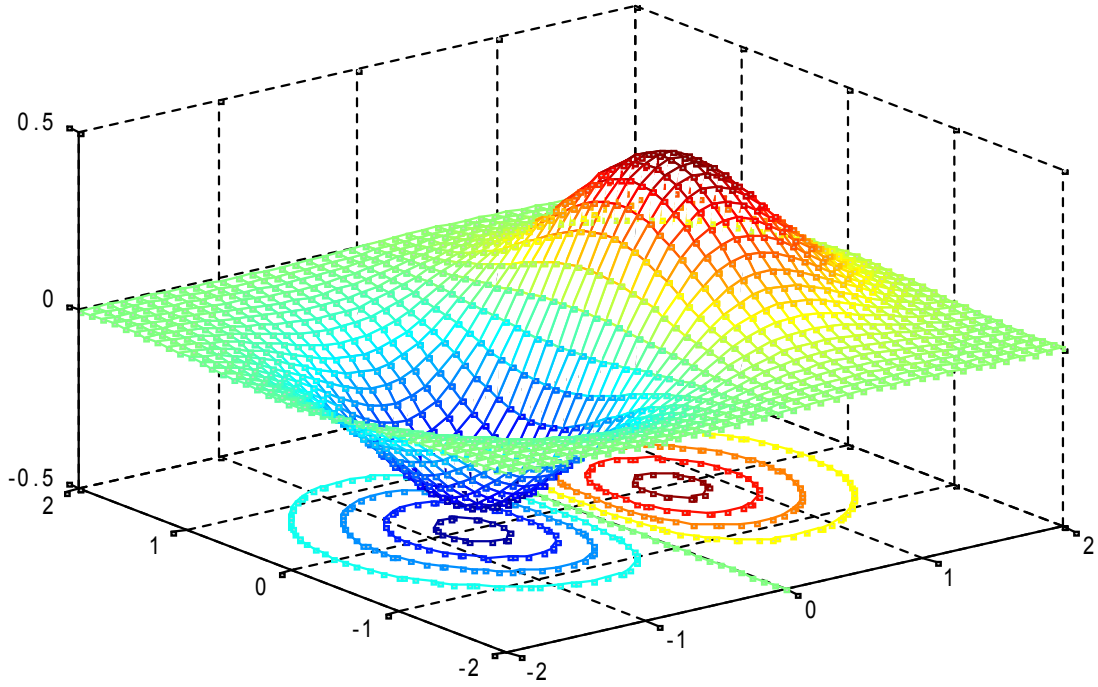
mesh(X,Y,Z) : Trace la maille ou grillage en 3-D

(mesh = maillage)



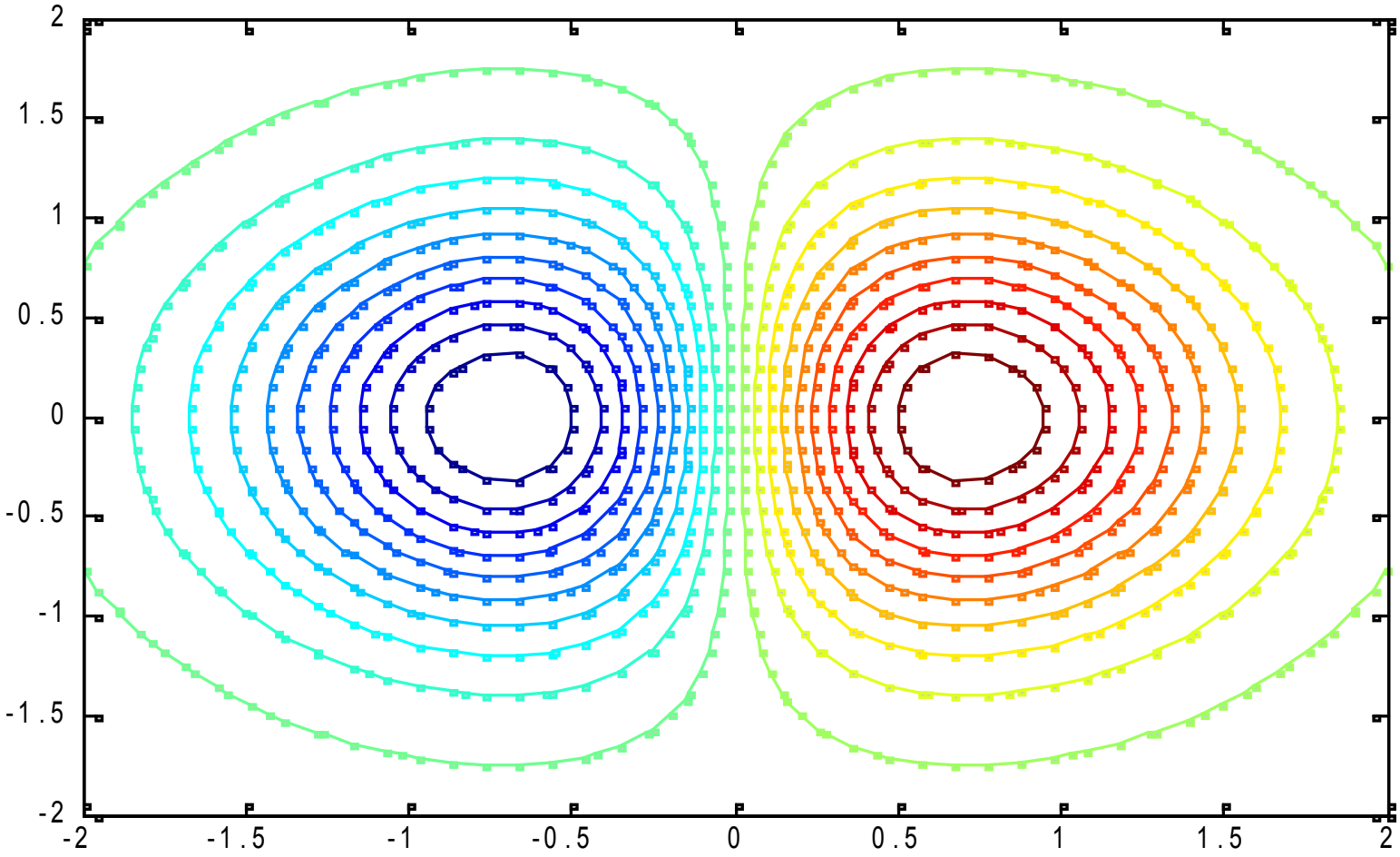
On peut utiliser la fonction linspace pour définir les vecteurs x et y.

```
x= linspace(-2, 2, 40); y = linspace(-2, 2, 40);
[X,Y] = meshgrid(x, y);  Z = X .* exp(-X.^2 - Y.^2);
meshc(X, Y, Z)  % grillage 3D avec les contours sur le plan de base
```



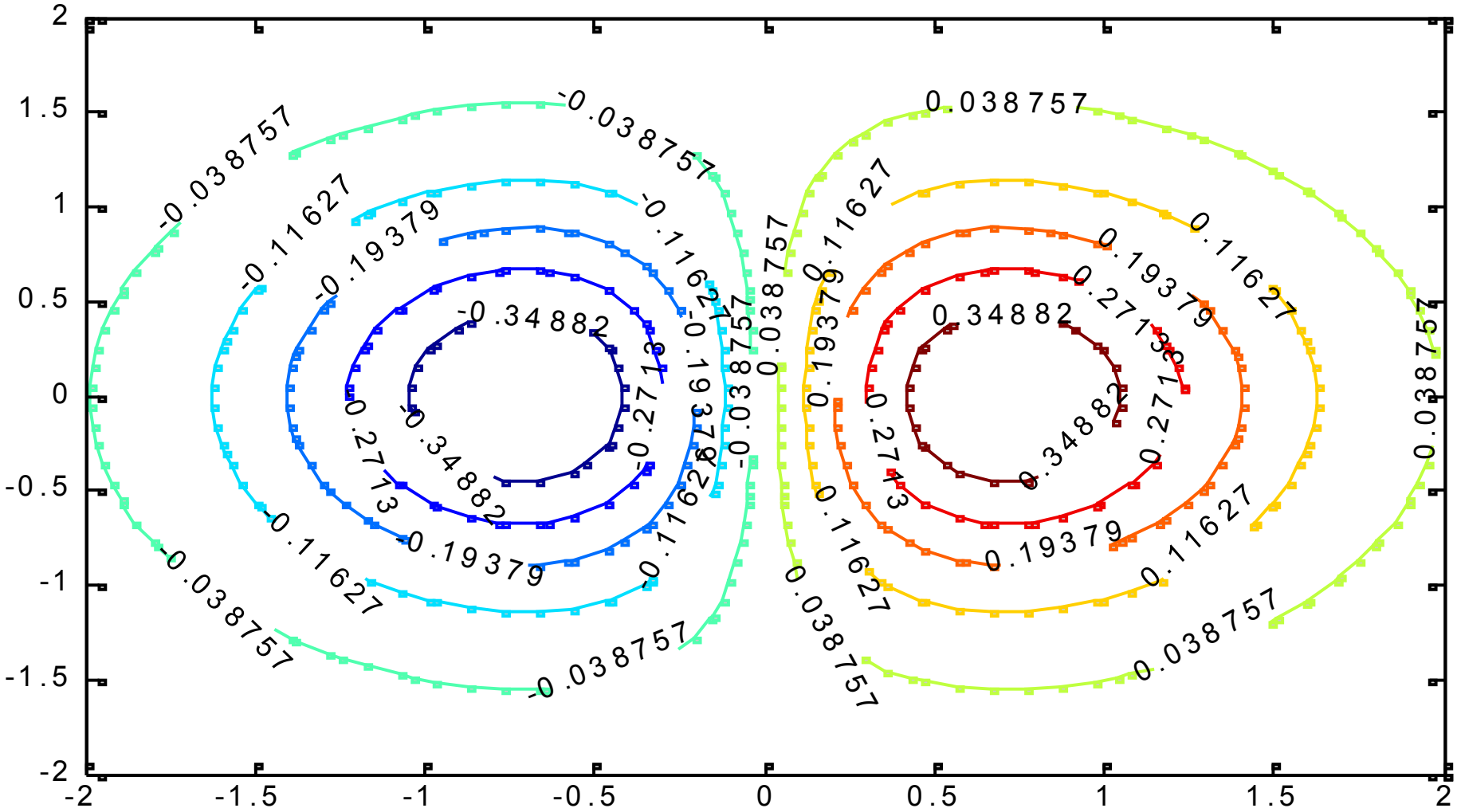
On peut tracer les courbes de niveau par : contour

contour(X,Y,Z,20) % 20 lignes de niveau



On peut donner le niveau des courbes par :

[c,h]=contour(X,Y,Z,10); clabel(c,h);



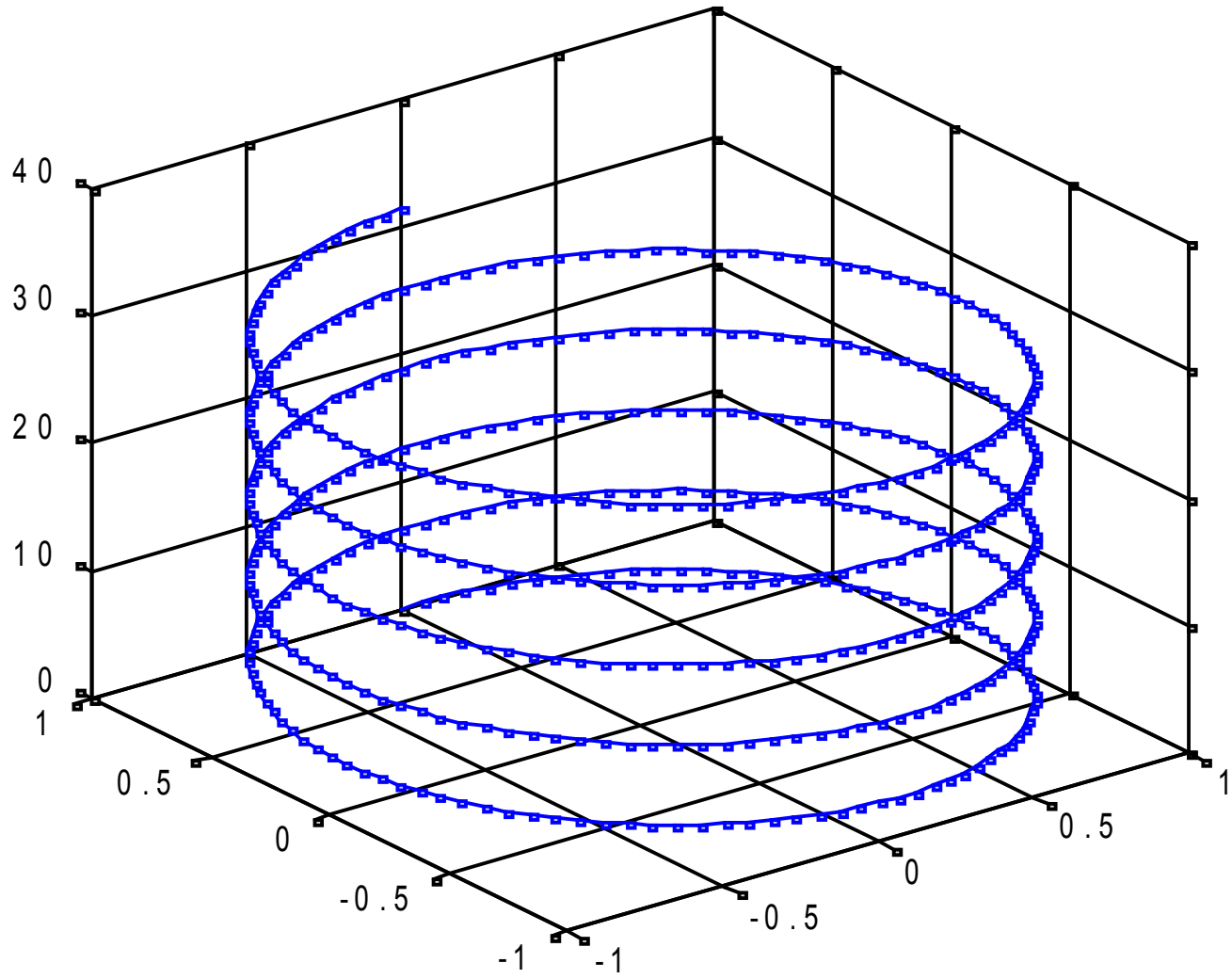
Si x est un vecteur de longueur n, y est un vecteur de longueur m et Z une matrice (m,n)

mesh(X,Y,Z) trace le maillage représenté par les points (x(j), y(i), Z(i,j)).

✓plot3

C'est l'équivalent de plot en 3 dimension (3-D). Si x, y, et z sont trois vecteurs de même longueur plot3(x,y,z) trace la ligne dans l'espace 3-D qui passe par les points de coordonnées les éléments x, **y,** et **z**.

```
z = 0:pi/50:10*pi; plot3(sin(z),cos(z),z) axis square;
grid on
```



➢ Animations par Matlab

✓ Exemple

```
t= linspace(1, 3, 40);
for i = 1:40
  [X,Y] = meshgrid(-2:.1:2, -2:.1:2);
  Z = t(i)*X .* exp(-X.^2 - Y.^2); surf(X,Y,Z);
  I(i) = getframe;
end
movie(I,10)   % Anime l'image 10 fois
```

➢Généralités

✓Trouver les racines d'un polynôme p(x).

MATLAB représente un polynôme comme un vecteur ligne. Par exemple le polynôme,

$$p = x^4 - 12x^3 + 25x + 116$$

s'écrit p = [1 -12 0 25 116]

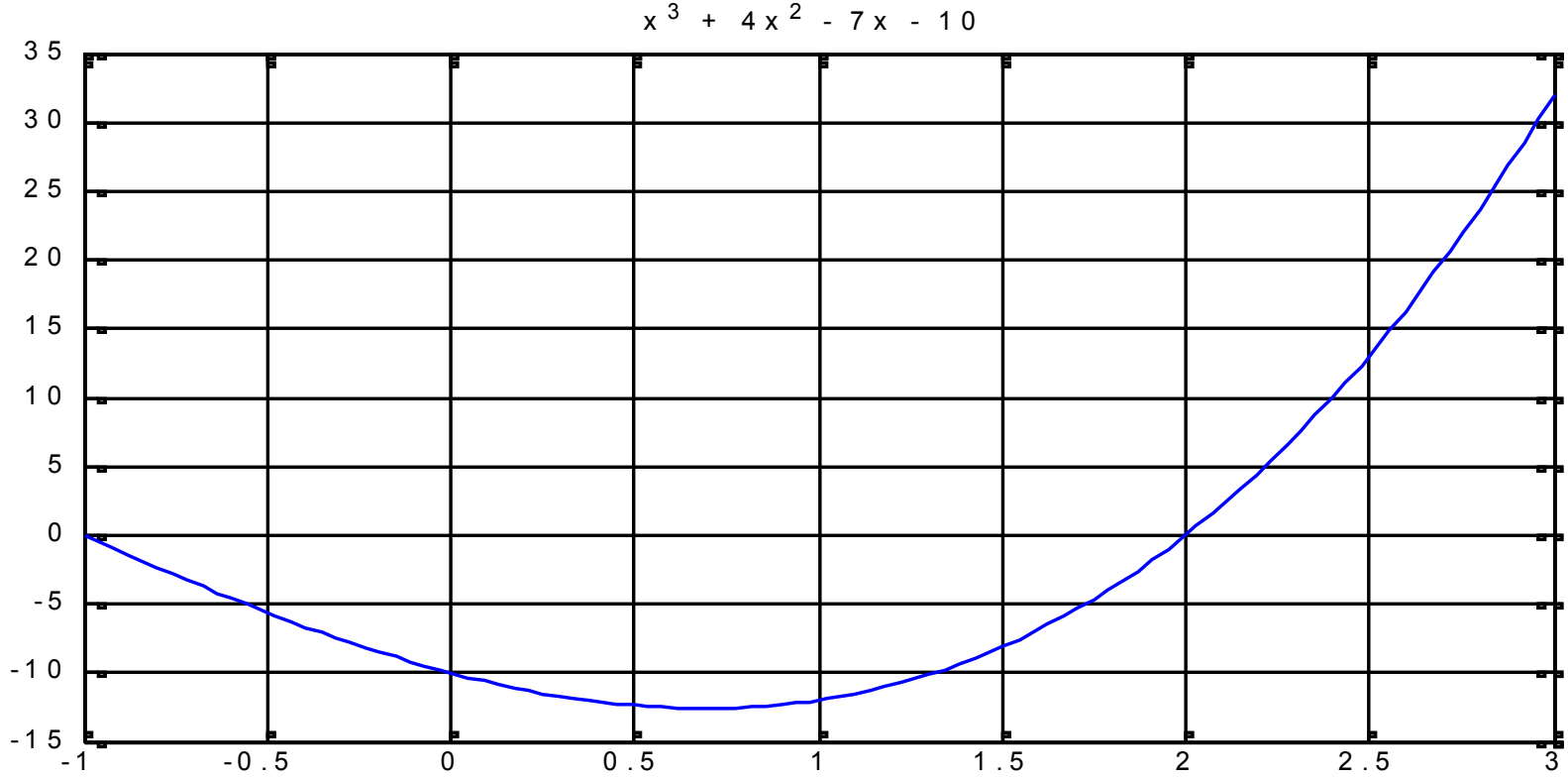
Pour obtenir les racines de ce polynôme, on utilise la commande roots.

```
r = roots(p);
```

➢ Tracé de la courbe

du polynôme p(x)= x^3 + 4x^2 - 7x - 10

```
>>  p = [1 4 -7 -10];  x = linspace(-1,3,100);
>> y = polyval(p,x);
>> plot(x,v); title('x^3 + 4x^2 - 7x - 10'); grid;
```



✓**Multiplication** de deux polynômes.

On applique la fonction MATLAB conv

>> a = [1 2 3 4]; b = [1 4 9 16];

>> c = conv(a,b)

✓**Division** de deux polynômes a par b.

On utilise la commande deconv

>> [q,r] = deconv(a,b)

q : représente le quotient de la division

r : représente le reste de la division

➢ DÉRIVÉES NUMÉRIQUES

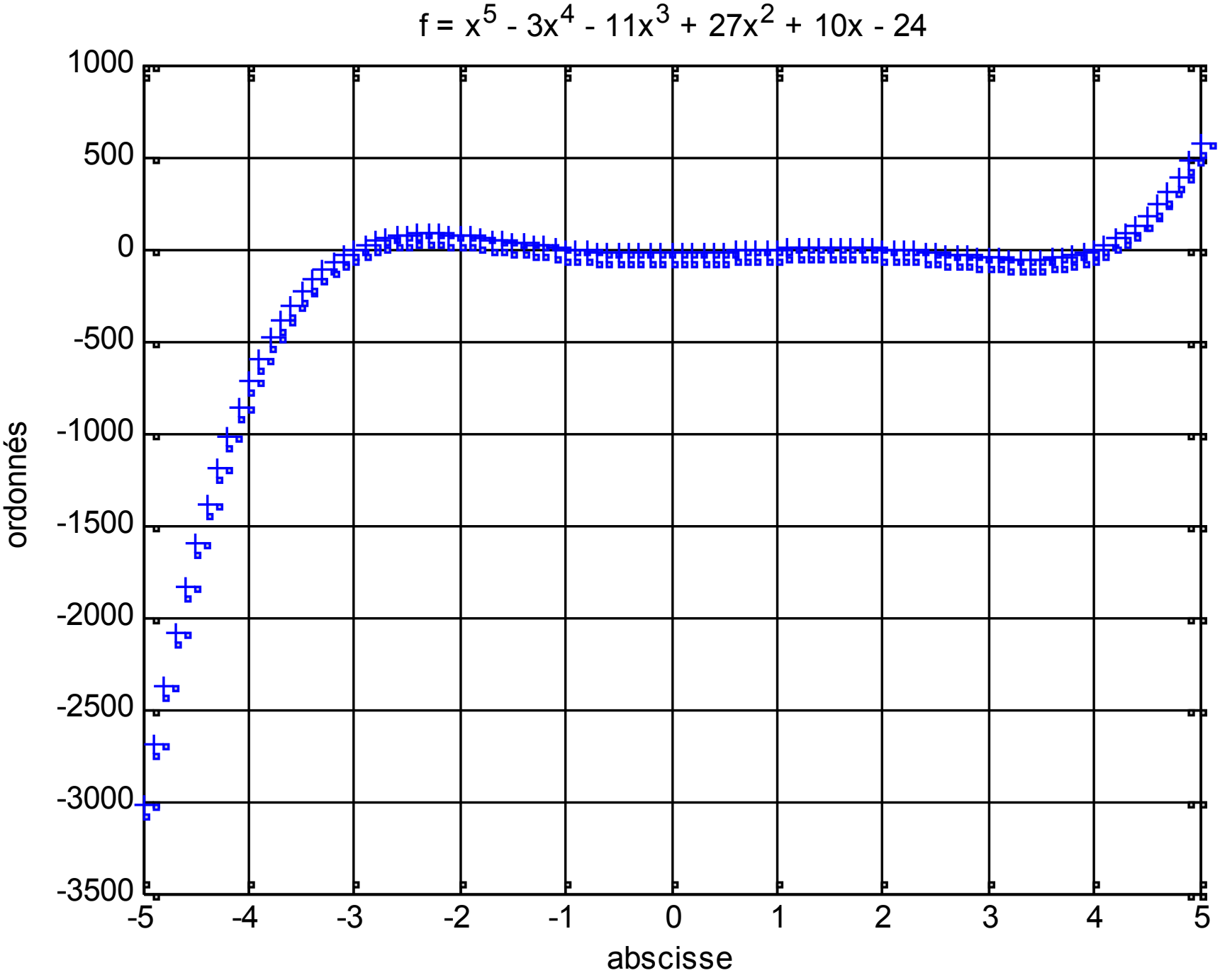
Considérons la fonction suivante

$$f(x) = x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24$$

1) Représentons cette fonction entre -5 et 5.
2) Calculer sa dérivée par rapport à x.
3) Représenter graphiquement cette dérivée.
4) Déterminer les valeurs de x pour lesquelles la fonction f(x) admet des maxima et des minima locaux.

1) Représentons cette fonction entre -5 et 5.

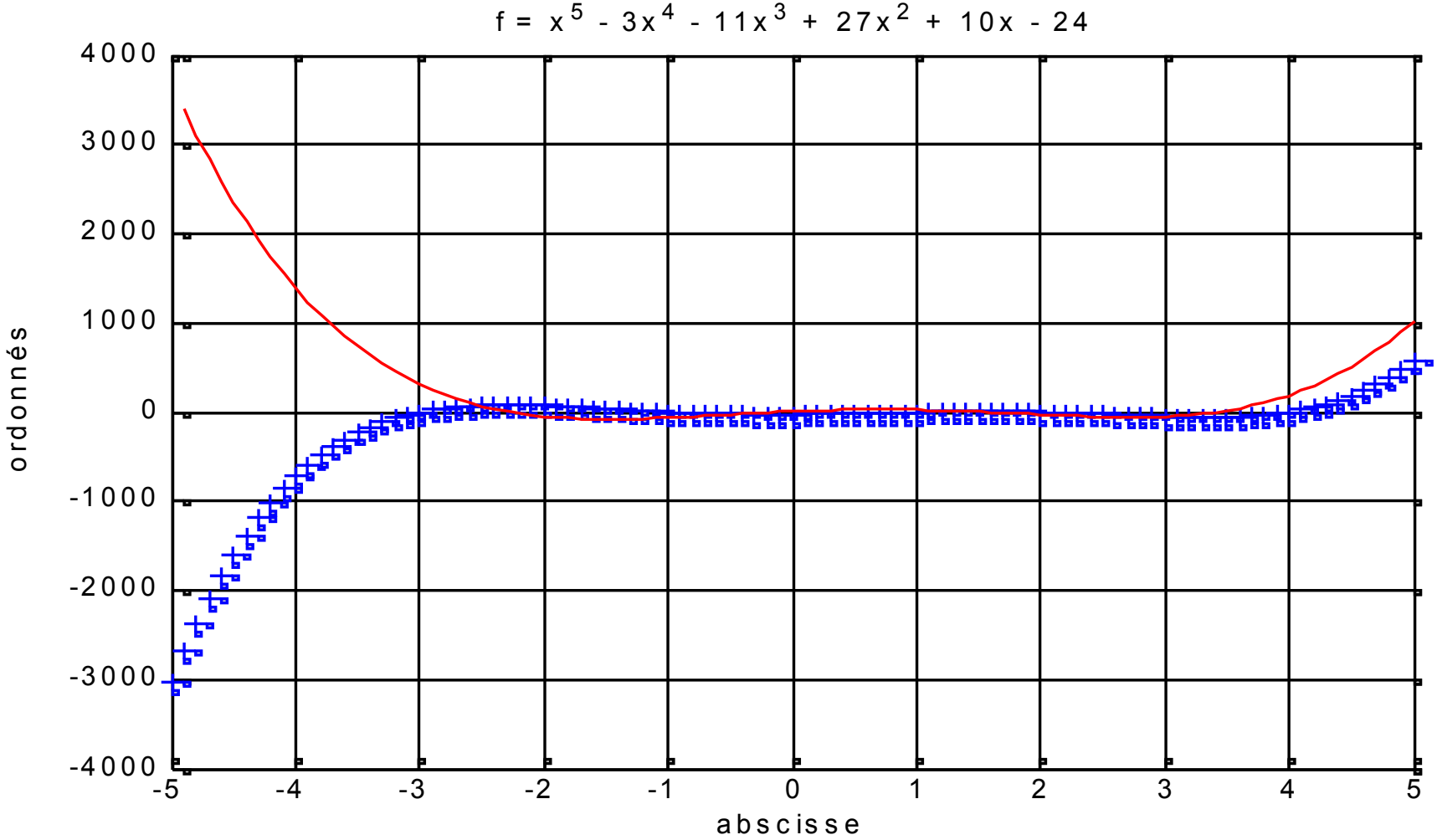
```
>> x = -5:0.1:5;
>> f = x.^5 - 3*x.^4 - 11*x.^3 + 27*x.^2 + 10*x -24;
>> hold on; plot(x,f,'+');
>> title(' f = x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24')
>> xlabel('abscisse')
>> ylabel('ordonnés')
>> grid
```


f = x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24
1000
500
0
-500
-1000
-1500
-2000
-2500
-3000
-3500
ordonnés
-5
-4
-3
-2
-1
0
1
2
3
4
5
abscisse

2,3) Calculer sa dérivée par rapport à x et la tracer.

>> df = diff(f)./diff(x); xd = x(2:length(x));

>> plot(xd,df);



4) Déterminer les valeurs de x pour lesquelles la fonction f(x) admet des maxima et des minima locaux.

```
>> produit = df(1:length(df)-1).*df(2:length(df));
>> local = xd(find(produit<0));

local =

   -2.3000   -0.2000    1.5000    3.4000
```

➢Construction de fonctions

- inline(‘expression’, ‘x1’, ‘x2’, ...)

  construit une fonction de variables x1, x2, …

```
>> g = inline('sin(2*pi*x + y)', 'x', 'y')

g =

     Inline function:
     g(x,y) = sin(2*pi*x + y)
```

➢Quelques méthodes préprogrammés

Instruction	Description
`fzero(f,a)`	recherche des zéros d'une fonction `f` autour de `a`
`quad(f,a,b)`	calcul de l'intégrale d'une fonction `f` entre `a` et `b`
`spline(xx,yy)`	calcul de la spline cubique passant par les points (xx,yy)
`fft(a)`	transformation de Fourier rapide du vecteur `a`
`ode23(f,t,y0)`	résolution de l'équation `y'=f(t,x), y(0)=y0`