TP 01 : Prise en Main

Principe et objectifs de TP 01 :

Dans ce TP, l'environnement de Matlab est présenté. Un premier exemple introductif montre rapidement le principe de fonctionnement du logiciel. Nous présentons ensuite un ensemble de fonctions de base nécessaires pour débuter en Matlab. Objectif de montrer comment se pratique Matlab dans une première utilisation basique.

Exercice 01 (Quelques commandes Matlab)

Commencez par essayer les commandes suivantes :

% pour commencer dans un environnement propre

clear all % supprime toutes les variables de la mémoire

close all % ferme toutes les fenêtres graphiques

clc % nettoie la fenêtre de commande

format short g

- **clock** : affiche l'année, le mois, le jour, l'heure, les minutes et les secondes.
- **date** : Affiche la date.
- **ans** : quand on introduise des instructions anonymes (sans variables en sortie), le **Matlab** considère une variable **'ans'** par défaut pour enregistrer le résultat.
- **input** : permet de lire une valeur à partir du clavier (l'instruction habituelle lire) **Exemple :**x = **input** (‘taper un nombre : ‘)
- **disp** : permet d'afficher un tableau de valeurs numériques ou de caractères. L'autre façon d'afficher un tableau est de taper son nom. La commande **'disp'** se contente d'afficher le tableau sans écrire le nom de la variable, ce qui peut améliorer certaines présentations. On utilise fréquemment la commande **disp** avec un tableau qui est une chaîne de caractères pour afficher un message.
  Exemple :>> disp('la valeurs saisie est erronée').
- **clear** : permet de détruire une variable de l'espace de travail (si aucune n'est spécifiée, toutes les variables seront effacées).
- **who** : donne la liste des variables définies dans l'espace de travail actuel (essayer whos).
- **whos** : donne la liste des variables définies dans l'espace de travail avec plus de détails.
- **Help** : on utilise cette commande pour obtenir l'aide sur une méthode donnée.

Exercice 02

Ouvrez l'´editeur **Matlab ("File- New- M-file"**, ou cliquez sur la page blanche de la barre d'outil).

Créez le **script Matlab** suivant :

```
clc , clear all ;
% Ceci est un script Matlab,
% le signe "pourcent" permet de mettre des commentaires
% qui ne seront pas interprètes disp('Hello World !')
 %disp permet d'afficher ce que l'on veut a l'écran,
%les simples cotes ' indiquent
```

```
%qu'on veut afficher du texte.
disp('Une 1ere affectation pour a et b')
a = 5
b = 6
c = a;
a = b;
b = c;
disp('Les nouvelles valeurs de a et b')
a, b %la virgule permet de mettre sur une seule ligne
%plusieurs commandes
%elle joue le role de s´eparateur d'instructions,
%comme la touche entree.
```

Enregistrez le sous le nom voulu (sans accent et sans espace), puis appelez le dans l'interpréteur **Matlab**.

Pour déclarer une fonction sous Matlab, sur deux méthodes possibles :

1. **Commande fonction :** Son lexique est comme suit, en considérant ses entrées et sorties :

Function [out1, out2, ...] = fun (in1, in2, ...)

Exemple

```
function E=Estim (t)
 E=t.^(-0.5)*exp(-t/2)/sqrt(2*pi)
 end
 En deuxième lieu, on enregistre cette fonction dans un fichier
M-fille nomme, par exemple, Estim.m et pour l'utiliser, on fait
appel a cette fonction par son nom:

>> t=4
 t =
 4
>> Estim(t)
 E =
 0.0270
```

Commande inline : Son lexique est comme suit : INLINE (EXPR, ARG1, ARG2, ...), Ou EXPR représente la fonction à déclarer alors que ARG1, ARG2 représentent les variables ou les entrées.

Exemple

```
clc , clear all ;
f=inline ('sin(x)-log(x)-sqrt(x)')
%évaluation de la fonction f avec des valeurs compris entre 1
et 10
for x=1:10
fprintf( 'f(%f)=%f\n' ,x , f ( x ) )
end
```

Exercice 03

Considérons le nombre complexe suivant : z1= 1+2i et z2=3-5j.

1. Calculez : z1+z2, z1-z2, z1/z2, z1 x z2
2. Calculez z2 à la puissance 2.

Solution

```
clc , clear all ;
z1+z2=4.0000 -3.000i
z1-z2=-2.0000+7.000i
z1/z2=-0.2.059+0.3235i
z1*z2=13.000+1.0000i
z2^2=-16.0000-30.0000i
```

Exercice 04

Considérons le nombre complexe suivant :

z=[1+j 2-3j ; 4+2j 5-4j]

Donnez la partie réelle, la partie imaginaire, le conjugué et le module de z sous **Matlab** produit de z par son conjugué produit scalaire de z par son conjugué.

Solution

```
>> z=[1+j 2-3j ; 4+2j 5-4j]
z =
1.0000 + 1.0000i 2.0000 - 3.0000i
4.0000 + 2.0000i 5.0000 - 4.0000i
>> r=real(z) r = 1 2
4 5
m=imag(z) m = 1 -3
2 -4
>> conj(z)
ans = 1.0000 - 1.0000i 2.0000 + 3.0000i
4.0000 - 2.0000i 5.0000 + 4.0000i
>> abs(z) ans = 1.4142 3.6056
4.4721 6.4031
>> conjz=conj(z) conjz = 1.0000 - 1.0000i
2.0000 + 3.0000i
>> produitscalaire=conjz.*z
produitscalaire = 2 13
20 41
>> produit=z*conjz
= 4.0000 -16.0000i 21.0000 - 2.0000i
18 -28.0000i 43.0000 +16.0000i
```

Exercice 05

Donnez le résultat MATLAB pour chacune des commandes suivantes :

a=5 ; b=a+2 ; c=b-3;bc.

Solution

```
clc , clear all ;
a=5 ; b=a+2 ; c=b-3 ;
b*c
ans =
27
>> temp=27.48 ; poids=15.63 ; floor(temp), ceil(poids) ;
round(poids)
ans =
    27
ans =
    16
>> >> format rat
>> sin (pi/6)
ans =
       1/2
```

Exercice 06

Donnez des commandes **`function`** permettant d'évaluer les expressions suivantes :

1. `>> x=1; -x^2-(7/4)*x^5+x^6+1`
2. `>> x=exp(2); (x^3*sin(7*pi/4)^2)/cos(3*pi-1)`
3. `>> x=3i; -2*log(7*x+1)+sqrt(4*x^3-1)`

Exercice 07

Donner la suite de Commande function pour calculer les formules suivantes : $V = 4/3\pi R^3$
Où R = 4cm.

Solution

```
function v=volum(R)
v=4/3*pi*R^2
end
>> volum(4)
v =
    6501/97
ans =
    6501/97
```

Exercice 07

Soit l'équation $x^2 - 2x = 1$

1. Utiliser la fonction *fzero* pour trouver la première solution au voisinage de x1 = 1, puis utiliser *fzero* une nouvelle fois pour trouver la deuxième racine au voisinage de x2 = -3.
2. Utiliser la fonction *fsolve* pour trouver simultanément les deux racines sur l'intervalle [-3, 1].
3. Utiliser la fonction *solve* pour trouver les racines de cette équation.

Solution

1. Utilisation de *fsolve* pour trouver les racines de l'équation une par une :

```
>> f = @(x) x.^2 -2*x - 1;
>> t = -1:0.1:3;
>> plot(t,f(t)),grid
On voit que cette fonction s'annule à 2 reprises dans
l'intervalle [-1, 3] :
>> x1 = fzero(f,1)
x1 =
-0.4142
>> x2 = fzero(f,-3)
x2 =
-0.4142
2. Utilisation de fsolve pour trouver toutes les racines
simultanément :

>> fsolve(f,[-1,3])
ans =
-0.4142 2.4142
3. Utilisation de solve :

>> syms x
>> f = x^2 -2*x-1 ;
>> ezplot(f,[-2,3])
>> X = solve(f)
X =
2^(1/2) + 1
1 - 2^(1/2)
>> subs(X)
ans =
2.4142
-0.4142
```

TP 02 : Vecteurs et Matrices

Principe et objectifs de TP 02

Le but de ce TP est de maitriser la création de vecteurs et le calcul arithmétique sur les vecteurs. Avec Matlab il n'y a aucune différence entre variables entière, variables 'réelle' ou variables 'complexe', le système d'allocation dynamique s'en charge de l'opération.
Le calcul matriciel est le point fort de Matlab. La déclaration des matrices est très simple sous Matlab des matrices. Une matrice est un tableau à deux dimensions avec (n) lignes et (m) colonnes et contenant des éléments de même type. L'objectif de ce TP est de maitriser la manipulation des matrices avec le langage Matlab.

Exercice 01

Soit la série de nombres [11 18 13 15 6 12 17 16 16 1 13 22].

1. Entrer ces valeurs dans le vecteur x ;

x=[11 18 13 15 6 12 17 16 16 1 13 22].

2. Calculer la longueur N de ce vecteur ;

3. Calculer la somme S des éléments ;

4. Calculer la moyenne $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$

5. Calculer l'écart-type $\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$.

6. Calculer le vecteur dx = $\{x_{i+1} - x_i\}$ pour i = 1; 2; :::; N -1.

Solution

```
clc, clear all ;
format short g
N=length(x)
S=sum(x)
xbare=S/N % version pédestre
xbare=mean(x) % version rapide
segma=sqrt(sum((x-xbare).^2)/(N-1)) % version pédestre
segma=std(x) % version rapide
dx=x(2:end)-x(1:end-1) % version pédestre
dx=diff(x) % version rapide
```

Exercice 02

Soit les trois matrices A, B et C :$A = \begin{pmatrix} 2 & 3 \\ 8 & 2 \end{pmatrix}, B = \begin{pmatrix} 3 & -1 \\ 0 & 5 \end{pmatrix}, C = \begin{pmatrix} -4 & -3 \\ 0 & 1 \\ 1 & 1 \\ -3 & -6 \end{pmatrix}$.

1. Calculez les expressions suivantes :

- A*B-6
- C*B+1-zeros(4,2)
- C(end:-1:1,2).\22

2.Créez la matrice M qui contient les matrices A et B l'une sur l'autre pour définir la 1ère et la 2ème colonne, et la matrice C pour définir la 3ème et la 4ème colonne (M = [[A; B] C]).

$$M = \begin{pmatrix} 2 & 3 & -4 & -3 \\ 8 & 2 & 0 & 1 \\ 3 & -1 & 1 & 1 \\ 0 & 5 & -3 & -6 \end{pmatrix}$$

3. Donnez le résultat MATLAB pour chacune des commandes suivantes :

- M(3,2) = 3
- M([2,3],:) = [ ]
- tril(M,-1)+triu(M,2)

Solution

```
clc, clear all ;
A = [2 3 ; 8 2]
  %3. Création de la matrice M
A =
     2     3
     8     2
B = [3 -1; 0 5]
B =
     3    -1
5
C = [-4 -3; 0 1; 1 1; -3 -6]
C =
    -4    -3
     0     1
     1     1
    -3    -6
A*B-6
ans =
     0     7
    18    -4
C*B+1-zeros(4,2)
ans =
   -11   -10
     1     6
     4     5
    -8   -26
C(end:-1:1,2).\22
ans =
   -3.6667
   22.0000
   22.0000
   -7.3333
M = [[A; B] C]
M =
     2     3    -4    -3
     8     2     0     1
     3    -1     1     1
     0     5    -3    -6
M(3,2) = 3
M =
     2     3    -4    -3
     8     2     0     1
     3     3     1     1
```

```
     0     5    -3    -6
M([2,3],:) = [ ]
M =
     2     3    -4    -3
     0     5    -3    -6
A=tril(M,-2)+triu(M,3)
T =
     0     0    -4    -3
     0     0     0    -6
```

Exercice 03

Soient les vectrices colonnes et la matrice suivants :

$$\vec{V}_1 = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}, \vec{V}_2 = \begin{pmatrix} -2 \\ 1 \\ 6 \end{pmatrix}, \vec{V}_3 = \begin{pmatrix} -1 \\ 2 \\ -3 \end{pmatrix}, \; A = \begin{pmatrix} 1 & 2 & 4 \\ 0 & 5 & 4 \\ -6 & 2 & 8 \end{pmatrix}.$$

1. **Structures Matlab**
   - Entrer ces données sous Matlab.
   - Calculer $\vec{V}_1 + 4\vec{V}_2 - \frac{\vec{V}_3}{8}$.
   - Calculer le produit scalaire entre les vecteurs $\vec{V}_1 et \vec{V}_3$.
   - Calculer le produit A$\vec{V}_1$.

2. **Commandes Matlab**

Trouver les commandes Matlab permettant de :
- calculer $\|\vec{V}_1\|_2, \|\vec{V}_2\|_1, \|\vec{V}_3\|_\infty$.
- déterminer les dimensions de la matrice A, en extraire le nombre de colonnes ;
- calculer le déterminant et l'inverse de A.

3. **Résolution de systèmes linéaires**

Proposer deux méthodes permettant de résoudre le problème $A\vec{x} = \vec{V}_1$, et déterminer les commandes Matlab associées.

Solution

```
clc, clear all ;
format short g
v1 = [ 2 ; 3 ; 5 ]
v2 = [ -2 ; 1 ; 6 ]
v3 = [ -1 ; 2 ; -3 ]
A = [ 1 2 4 ; 0 5 4 ; -6 2 8]
v1+4*v2-v3/8
v1'*v2
A*v1
norm(v1,2)
norm(v2,1)
```

```
norm(v3,inf)
size(A)
size(A,2)
det(A)
inv (A)
x = inv(A)*v1
x = A\v1.
```

Exercice 04

Soit la matrice carrée A :

$$A = \begin{pmatrix} 10 & 10 & 0 \\ 0 & 10 & 10 \\ 10 & 19 & 10 \end{pmatrix}$$

1. Créer une matrice A.
2. Calculez le déterminant.
3. Trouver la matrice inverse de A.
4. Générer une vectrice colonne t qui va de 1 à 10 par pas de 0; 5.
5. Extraire la première ligne.
6. Extraire la deuxième colonne.
7. Extraire la diagonale.
8. Extraire le bloc contenant la deuxième et la troisième ligne avec la première et la deuxième colonne.

Solution

```
clc, clear all ;
A= [10 10 0; 0 10 10;10 19 10]
A =
10 10  0
 0   10  10
 10 19  10
>> det (A) % déterminant de A
ans =
100
>> inv (A)
ans =
0.9000  -1.0000   1.0000
 1.0000  1.0000  -1.0000
1.0000  -0.9000   1.0000
>> t = [1 : 0 . 5 : 1 0] % vecteur
t =
Columns 1 through 7
 1.0000 1.5000 2.0000 2.5000 3.0000 3.5000
4.0000
 Columns 8 through 14
4.5000 5.0000 5.5000 6.0000 6.5000 7.0000
```

```
7.5000
Columns 15 through 19
8.0000 8.5000 9.0000 9.5000 10.0000
>> A( 1 , : ) % premier l i g n e de A
ans =
10 10 0
>> A ( : 2) % deuxième colonne de A
ans =
10
10
19
>> diag (A) % di a g ona l e de A
ans =
 10
 10
 10
>> A (2 : 3 , 1 : 2 )
ans =
0 10
10 19
```

Exercice 05

Soient la matrice et les vectrices colonnes suivantes

$$\vec{V_1} = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}, \vec{b_2} = \begin{pmatrix} -2 \\ 1 \\ 6 \end{pmatrix}, \qquad A = \begin{pmatrix} 1/2 & 2/3 & 4/5 \\ 5/4 & 5/2 & 4/3 \\ -6/7 & 2/7 & 8/7 \end{pmatrix}.$$

On définit, pour n ≥1, la suite de vecteurs$\vec{v}_{n+1} = A\vec{v}_n + \vec{b}_2$.

1. Construire une fonction suite .m calculant les premiers termes de la suite$\vec{v}_n$. Cettefonction aura comme arguments d'entrée les données suivantes : la matrice A, le second membre$\vec{b}_2$, le terme initial$\vec{V}_1$, et le nombre de termes voulus nb_{it}.
2. Représenter graphiquement l'´evolution de chacune des composantes.

Qu'observe-t-on ?

Solution

Dans le fichier **sui .m**, créer la fonction :

```
Function u = sui(A, v1, b, nb_it)
v = zeros (3, nb_it+1);
v(:,1) = v1;
for k = 1:nb_it
v(:,k+1) = A*v(:,k)+b;
end
```

```
En sortie, le tableau u contient les itères
successifs(v_n)_{1≤n≤nb_it} .
Puis, taper en ligne :
A = [1/2 2/3 4/5 ; 5/4 5/2 4/3 ; -6/7 2/7 8/7]
b = [-2 ; 1 ; 6]
v1 = [2 ; 3 ; 5]
v = sui (A, v1, b, 30);
Pour étudier le comportement des itères, taper en ligne :
hold on
plot(v(1,:),'g')
plot(v(2,:),'r')
plot(v(3,:),'y')
hold off
```

Exercice 06

Écrire un script MATLAB qui permet de calculer les éléments de la matrice C, la somme de deux matrices A et B de dimensions 2*3 chacune.

Solution

```
clc, clear all ;
for i=1:2
for j=1:3
C(i,j)=A(i,j)+B(i,j);
end
end
C
```

Exercice 07

Écrire un programme Matlab qui déterminer :

1. le max et sa position de chaque linge
2. le min et sa position de chaque collons

Solution

```
clc , clear all
nl=3 ; nc=4 ;
a=[4 5 7 2 ; 3 9 8 -3 ; -2 0.5 11 2.7 ]
disp('==== le max et sa position pour chaque linge ===')
for i=1:nl
v(1:nc)=a(i,1:nc) ;
[max_l(i),loc_l(i)]=max(v);
end
max_l
loc_l
disp('==== le min et sa position pour chaque collons ===')
for j=1:nc
```

```
u(1:nl)=a(1:nl,j) ;
[min_c(j),loc_c(j)]=min(u);
end
min_c
loc_c
```

Exercice 08

Par la Méthodes d élimination de gauss résoudre le system des équations lainières Ax=b.

$$A=\begin{pmatrix}1 & -1 & 2\\ 3 & 1 & -1\\ -1 & 2 & 2\end{pmatrix},\ b=\begin{pmatrix}3\\ 6\\ 2\end{pmatrix}$$

```
clc,clear all
g=[1 -1 2 3;3 1 -1 6;-1 2 2 2 ];

[m,n]=size(g);
for j=1:m-1
        if g(j,j)~=0
    for i=j+1:m
        g(i,:)=g(i,:)-g(j,:)*(g(i,j)/g(j,j));
    end
        end
end;
b=g(:,n);
x=zeros(1,m);

for i=m:-1:1
    c=0;
    for j=1:m
        c=c+g(i,j)*x(j);
    end
    x(i)=(b(i)-c)/g(i,i);
end
disp ('Gauss elimination method:');
g
x'

Gauss jordan method:

h =

       1              0              0              1
       0              1              0             -2
       0              0              1             -3


ans =

       1
```

```
      -2
      -3
```

Exercice 09

Par la Méthodes d élimination de gauss jordanb résoudre le system des équations lainières Ax=b.

$$A = \begin{pmatrix} 3 & 2 & -1 \\ 4 & -3 & 2 \\ 1 & 1 & -3 \end{pmatrix},\ b = \begin{pmatrix} 2 \\ 4 \\ 8 \end{pmatrix}.$$

Solution

```
Clc,clear all
h=[3 2 -1 2;4 -3 2 4;1 1 -3 8];
[m,n]=size(h);
for j=1:m-1
        if h(j,j)~=0

        end

    for i=j+1:m
        h(i,:)=h(i,:)-h(j,:)*(h(i,j)/h(j,j));
    end
end


 for j=m:-1:2
     for i=j-1:-1:1
        h(i,:)=h(i,:)-h(j,:)*(h(i,j)/h(j,j));
    end
 end
for s=1:m

   h(s,:)=h(s,:)/h(s,s);
   x(s)=h(s,n);
end

disp ('Gauss jordan method:');
h

x'
```

Exercice 10

Par la Méthodes itérative de jacobi résoudre le system des équations lainières Ax=b.

$$A = \begin{pmatrix} 5 & 2 & -1 \\ 1 & 6 & -3 \\ 2 & 1 & 4 \end{pmatrix},\ b = \begin{pmatrix} 6 \\ 4 \\ 7 \end{pmatrix}.$$

Solution

```
clear all ;
close all ;
clc
x0=[0 0 0]';
a=[5 2 -1;1 6 -3;2 1 4]
b=[6 4 7]';
n =length(b);
err=1;
while err>10^-3
    for j=1:n
    x(j)=(b(j)-a(j,[1:j-1,j+1:n])*x0([1:j-1,j+1:n]))/a(j,j);
    end
   err=max(abs(x'-x0));
x0=x';
end
    r=x';
    disp(' méthode itérative jacobi 2')
r

a =

       5              2             -1
       1              6             -3
       2              1              4

 méthode itérative jacobi 2

r =

   78961/78960
    7089/7090
    4064/4063
```

Exercice 11

Par la méthode itérative de gauss saidle résoudre le system des équations lainières Ax=b.

Solution

```
%%%%methodes iteratives  de pour resouder un system des
éqautions lénieres%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%mihoubi/h%%%%%%%%%%
clear all ;close all ;clc
x0=[0 0 0]';
a=[10 -2 1; -2 10 -2;-2 -5 10];
b=[5 26 -7]';
d=diag(diag(a));
```

```
L=tril(a,-1);
U=triu(a,1);
x1=1;
xi=x0;
while max(abs(x1-xi))>10^-3
    x1=-inv(d+L)*U*x0+inv(d+L)*b;
    xi=x0;
    x0=x1;
end
p=x1;
disp(' méthode itérative gauss saidle')
p

méthode itérative gauss saidle

p =

   33452/33453
   83414/27805
   41734/41735
```

TP 03 : Éléments de programmation

Principe et objectifs de TP 03 :

Ce TP d'introduction porte sur l'utilisation des structures de programmation. On la déjà vu, les scripts et les fonctions sont deux outils incontournables de Matlab. Cependant pour faire des scripts complexes il est souvent nécessaire de faire appel aux structures de programmation comme les boucles et les réalisations conditionnelles.

Exercice 01 (Les scripts)

Pour pouvoir réutiliser les lignes de calcul, il est utile de les mettre dans un script. Un script est un fichier texte que Matlab pourra lire et effectuer.

1. Ouvrez l'éditeur de scripts de Matlab soit en cliquant sur la page blanche de la barre d'outils, soient allant dans le menu **"File-New-M-file".**

Créez le script suivant :

```
a = input('entrez a : ') % entrez a=
b = 6
b=b+a
a, b
```

Enregistrez le fichier et appelez le dans l'interpréteur.

2. Écrivez un programme qui demande deux valeurs a et b à l'utilisateur et qui les affiche, qui intervertit leurs contenus et qui les affiche à nouveau.

Solution

```
a = input('entrez a : ');
b = input('entrez b : ');
```

```
c = a;
a = b;
b = a;
disp(a);
disp(b);
```

Exercice 02

L'instruction **if** exécute un ensemble de commande si une condition est satisfaite.

La syntaxe générale est***if****(expression ou condition logique)*

instruction

end

If *(expression ou condition logique)*

instruction 1

else

instruction 2

end

Calculer la valeur de y

$$y = \begin{cases} \exp(x) + \sin(x) & ; x < 1 \\ x^2 - \ln(x) & ; x \geq 1 \end{cases}$$

Solution

```
x= - 3
if(x< 1)
y=exp(x)+sin(x)
else
y=x^2-log(x)
end
```

Exercice 03

On peut répéter des actions grâce aux boucles : la boucle for permet de changer la valeur d'une variable de manière régulière. La syntaxe pour la boucle for est la suivante :

```
for i=1: n
disp(i);
end
while(expression ou condition logique )
Instruction
end
```

Le code entre le for et le end est exécuté n fois : une première fois avec la variable i à 1, une deuxième fois avec la variable i à 2, etc jusqu'à n.

1. Écrivez un programme qui demande deux entiers *a* et *b* et qui affiche le résultat de la somme suivante$\sum_{K=1}^{b} K^a$.

Solution

```
a = input('entrez a : ');
```

```
b = input('entrez b : ');
S=0;
for k=1:b
S=S+k^a;
end
S
```

Exercice 04

En utilisant la méthode de Dichotomie calculé la valeur approchée de la racine de f(x) = 2x-cos(x) et eps=0.001 .

```
clc,clear all
a=0; fa=2*a-cos(a);
b=pi/6; fb=2*b-cos(b);
eps=0.001;
if fa*fb<0
 while (b-a)>eps
 x=(a+b)/2;
 fx=2*x-cos(x);
 if fa*fx<0
 b=x;
 else
 end
  if fa*fx>0
 a=x;
  end

 if fa*fb==0
     x;
 end;
 end
end
 x
>>
x =
    0.4505
```

Exercice 05

En utilisant la boucle while ou la boucle for, écrire un programme en Matlab qui:

1. Calculer la somme :$\sum_{K=1}^{N} K = 1 + 2 + 3 + \cdots . N$.

2.Calculer la factorielle de N$N!= 1*2*3*....*N$

Solution

```
clc; clear all
n=5; s=0 ; fact=1 ; k=1 ;
while (k<=n )
```

```
s=s+k;
fact=fact*k ;
k=k+1;
end
s
fact
clc; clear all
n=5 ; s=0 ; fact=1 ;
for k=1 :1 :n
s=s+k ;
fact=fact*k ;
end
s
fact
```

Exercice 06

Le jeu *dichotomie* est un jeu qui se joue à deux joueurs (et qui n'est pas très amusant en fait). Le premier joueur choisit un nombre entre 1 et 100, et le second joueur essaie de deviner ce nombre. A chaque essai du second joueur, le premier indique si le nombre est plus grand ou plus petit que le nombre à deviner (ou égal, et dans ce cas le jeu est terminé).

1. Écrivez un script qui permet de jouer à *dichotomie* contre l'ordinateur (le script choisit un nombre, et le joueur essaie de le deviner).

Indications :

- Utilisez la fonction randi(n) qui renvoie un nombre entier tiré aléatoirement entre 1 et *n* pour choisir le nombre ;
- Utilisez l'instruction a = input('Entrez un nombre : ') qui demande à l'utilisateur d'entrer un nombre et enregistre la valeur dans la variable a.

Solution

```
k=randi(100);
a=k-1;
while a~=k
a=input('Entrez un nombre :');
if a>k
disp('Le nombre est plus grand');
elseif a<k
disp('Le nombre est plus petit');
end
end
disp('Vous avez gagné');
```

Exercice 07

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*.

A l'aide des fonctions *tic* et *toc*, calculer les temps d'exécution de chacune de ces méthodes. Comparez dans un même graphe ces temps.

Solution

Voici un script fact.m possible pour répondre à cet exercice :

```
% Calcul du factoriel par 3 méthodes
N = 0 : 2 :12;
n = size(N,2);
RES = zeros(3,n); % matrice des résultats
TE = zeros(3,n); % matrice des temps d'exécution
% 1) A l'aide de la fonction built-in "factorial" de Matlab
for k = 1: n
tic
RES(1,k) = factorial(N(k));
TE(1,k) = toc;
end % 2) A l'aide de la boucle for
for k = 1:n
tic
RES(2,k) = 1;
for i = 1 : N(k),
RES(2,k) = RES(2,k) * i;
end
TE(2,k) = toc;
end % 3) A l'aide de la boucle while
for k=1:n
tic
RES(3,k) = 1;
i = 2;
while i <= N(k)
RES(3,k) = RES(3,k) * i;
i = i + 1;
end
TE(3,k) = toc;
end
format long
disp 'Resultats obtenus par les 3 méthodes : ' RES
disp ' Comparison des temps d''execution :' TE plot(N, TE')
title('Comparison des temps d''execution')
xlabel('valeur de N')
ylabel('Temps d''execution en seconde')
legend('factorial','for','while')
grid
```

Exercise 08

Nous avons déjà vu comment utiliser une boucle for pour répéter plusieurs instructions. Cependant il fallait connaître à l'avance le nombre de fois que l'on voulait répéter la boucle. Le mot-clé while sert à exécuter une suite d'instructions jusqu'à ce qu'une ce qu'une condition soit vérifiée.

1. Exécutez le script suivant :

```
x = 1;
while x < 1000
x
x = 2*x;
end
```

Que fait-il ? Avez-vous compris la signification du mot-clé while ?

Réponse : Ce programme calcule la plus petite puissance de deux supérieure à mille.

2. Écrivez une fonction cube(n) qui renvoie le plus grand cube inférieur ou égal à *n*.

Solution

```
function r=cube(n)
i=0;
while i^3<=n
i=i+1;
end
r=(i-1)^3;
```

3. Écrivez une fonction somme(n) qui renvoie le plus grand entier *k* vérifiant

$$\sum_{i=1}^{K} i^2 \leq n$$

Solution

```
function r=somme(n)
i=0;
S=0;
while S<=n
i=i+1;
S=S+i^2;
end
r=i-1;
```

Exercice 09

Créons un programme qui trouve les racines d'une équation de second degré désigné par : ax^2+bx+c=0.

Solution

```
% Programme de résolution de l'équation a*x^2+b*x+c=0
 a = input ('Entrez la valeur de a : '); % lire a
 b = input ('Entrez la valeur de b : ');   % lire b
```

```
 c = input ('Entrez la valeur de c : ');   % lire c
delta = b^2+4*a*c ; % Calculer delta
if delta<0
  disp('Pas de solution')   % Pas de solution
elseif delta==0
  disp('Solution double : ') % Solution double
 x=-b/(2*a)
else
 disp('Deux solutions distinctes: ') % Deux solutions
 x1=(-b+sqrt(delta))/(2*a)
 x2=(-b-sqrt(delta))/(2*a)
end
```

Si nous voulons exécuter le programme, il suffit de taper le nom du programme :

```
>> equatioddegre
Entrez la valeur de a : 9
Entrez la valeur de b : -2
Entrez la valeur de c : 4
Deux solutions distinctes:

x1 =
    0.7870
x2 =
   -0.5648
```

Exercice 10

Exponentielle réelle négative Programmer en Matlab une fonction qui prend en entrée un réel t et un entier N et qui retourne la somme partielle

$$\sum_{K=0}^{N} x^K / k!$$

Tester votre fonction avec t = 16 et diverses valeurs de N.

Solution

```
function [A] = exp(t,N)
A=1;
 for k =1:N
 A=A+(t.^k)/ factorial (k);
end
 end
```

Exercice 11

Par la méthode de gauss jordan pour résoudre un system des équations lainières hx=b si

$$H = \begin{matrix} 1 & 1 & 1 & -1 \\ 2 & -1 & -1 & 1, \\ 1 & 3 & -2 & 4 \end{matrix} \quad b = \begin{matrix} 4 \\ -1 \\ -14 \end{matrix}$$

-5 1 3 2 -4

Solution

```
clc,clear ,all
h=[1 1 1 -1 4;2 -1 -1 1 -1;1 3 -2 4 -14;-5 1 3 2 -4]
[m,n]=size(h);
for j=1:m-1
        if h(j,j)==0
            t=h(1,:);h(1,:)=h(j+1,:);
            h(j+1,:)=t;
        end

    for i=j+1:m
        h(i,:)=h(i,:)-h(j,:)*(h(i,j)/h(j,j));
    end
end
 for j=m:-1:2
     for i=j-1:-1:1
        h(i,:)=h(i,:)-h(j,:)*(h(i,j)/h(j,j));
    end
end

for s=1:m
   h(s,:)=h(s,:)/h(s,s);
   x(s)=h(s,n);
end

disp ('Gauss  jordan method:');
h
x'
Gauss  jordan method:
h =

    1.0000         0         0    0.0000    1.0000
         0    1.0000         0    0.0000   -1.0000
         0         0    1.0000   -0.0000    2.0000
         0         0         0    1.0000   -2.0000
ans =

    1.0000
   -1.0000
    2.0000
   -2.0000
```

Exercice 12

En utilisant la méthode de Newton calculer la valeur approchée de la racine de f(x) = x^2-17 avec eps=10^-5 .

```
clc,clear ,all
```

```
 a=4;
 b=4.5;
 eps=0.1;
 x=4.5;
N=0;
 err=1;
 f= inline('(x^2)-17');
df=inline('2*x');
 fprintf('N | x | f(x) \n')
 if f(a)*f(b)<0
 while err>eps
 N=N+1;
 xt=x-(f(x)/df(x)); %xn+1=xn_f(xn)/f'(xn)
 err=abs(x-xt);
 x=xt;
 fprintf('%i|%8.5x| %8.5f \n',N,x,f(x))
 end
 end
('x soloution approchee')
 X
N | x | f(x)
1|4.13889e+000|  0.13040
2|4.12314e+000|  0.00025

ans =
x soloution approchee
x =
    1105/268
```

TP04 : Polynômes

Principe et objectifs de TP 04 :

Dans cette série, nous travaillons sur le calcul polynomial qui est à la base de nombreux domaines scientifiques, notamment le traitement du signal numérique et analogique, le contrôle de processus, l'approximation de fonctions et l'interpolation de courbes.

Exercice 01

En utilisant les commandes Matlab : définie le polynôme $P(x) = 3x^4 - 2x^2 + x$:

1. Calculer P(1), P'(3.5) et P''(0). (polyval et polyder)
2. Définie le vecteur V qui contient 100 valeurs compris entre 0 et 2 (linspace).
3. Évaluer le polynôme P(x) sur les points de V.
4. Tracer la courbe du polynôme P dans l'intervalle [1,3]. (plot)
5. Soit le polynôme$S(x) = x^4 - x^3 + 1$, calculer la somme de S et P.

Solution

```
clc ; clear all
% Partie polynome
% definir le polynome p(x)
% p(x)=3x^4_2x^2+x.
 p=[3 0 _2 1 0 ]
 % caluler p(2)
 p2=polyval (p , 2 )
 % calculer p'(3)
 dp=polyder (p)
 dp3=polyval (dp , 3 )
% calculer p"(0)
d2p=polyder (dp)
 d2p0=polyval ( d2p , 0 )
% definir le vecteur v
 v=l inspace ( 0 , 2 , 1 0 0 )
 % evaluer p(x) en v
pv=polyval (p , v )
 % tracer pv sur [0 2]
plot (v , pv )
 % definir s(x)=x^4_x^3+1
s=[1 _1 0 0 1 ]
 % la somme de s et p
som=s+p
```

Exercice 02

Soit le polynôme $P(x) = 2x^4 - 3x^3 + 13x - 25$

1. Donner la représentation de ce polynôme dans MATLAB
2. Donner le résultat de polyval(P,4) et de polyval(polyder(P),4) en justifiant les calculs

Donner la commande MATLAB qui permet de trouver les racines de P(x)

Solution

```
cla ,clear all
% d e c l a r a t i o n du v e c t e u r r e p r e s e n t a t i v e
 % du polynome P( x )
 P=[2 -3 0 13 -25]
 polyval(P, 4 ) % c a l c u l P(4)
 polyval(polyder (P ) , 4 ) % c a l c u l P'(4)
 roots (P)
```

Exercice 03

Soit la fonction $f(x) = 3x - (1/2)e^x$ écrire un script matlab permettant de trouver la racine de cette équation sur l'intervalle [0 1] par la méthode du point fixe, en utilisant l'lalgorithme .

Solution

```
clear; clc;
```

```
a=0;
b=1;
eps=0.00001;
fa=3*a-0.2*exp(a);
fb=3*b-0.2*exp(b);
if fa*fb<0
x0=a;
x1=0.2*exp(x0)/3;
while(abs(x1-x0)>eps)
x0=x1;
x1=0.2*exp(x0)/3;
end;
fprintf('la racine vaut: %f',x1);
else
fprintf('pas de racines dans cette intervalle');
end
```

Exercice 04

Dans cette exercice, on a choisi quelque problème mathématique résolu directement en langage Matlab sansfaire discrétisation numérique comme :

1. équation non linéaire analytiquement :$f(x) = x^2 + 5x + 6$
2. calcule la dérive : $diff(f,x) = \frac{df}{dx} = d/dx(1/(x^2 + 2x + 2))$
3. calcule $\int f(x)dx = \int(1/(x^2 + 2x + 2))\, dx$
4. résoudre équation différentielle du1^{ere} ordre :$\frac{dy}{dx} = F(x,y) = (x^2 - 1).y$
5. résoudre équation différentielle du 2^{eme} ordre$\frac{d^2y}{dx^2} - 5\frac{dy}{dx} + 6y = x^2$

Solution

```
clc ;clear all
disp('== resoudre equation non lineaire analytiquement ==')
syms x
xc=eval(solve(x.^2+5*x+6))
disp('========= la deriver d un fonction ==========')
syms x ;
derive=diff('1/(x^2+2*x+2)',x)
disp('========= integral ============')
syms x ;
intf=int('1/(x^2+2*x+2)',x)
disp('== solution analytiquement eq diff 1 ere ordre =')
dsolve('D1y=(x^2-1)*y','x')
disp('=== solution analytiquement eq diff 2 eme ordre ==')
dsolve('D2y-5*D1y+6*y=x^2','x')
```

Exercice 05

Soit le polynôme P $P(x) = 2x^3 - 5x^2 + 7x - 20$

1. Représenter le polynôme P sous Matlab

2. Tracez la courbe de P sur l'intervalle [-9 9]. Ajoutez le titre 'Courbe Représentative de p(x)'.
3. Donner la commande permettant d'évaluer P avec la valeur x= 6
4. Donner la commande permettant de calculer les racines de P(x).

Solution

```
clc,clear all
p=[2 -5 7 -20]
 x=[-9:0.001:9];
 f=2*x.^3-5*x.^2+7*x+20;
 plot(x,f)
 title('Courbe Representative de P(x)')
 polyval(p,5)
  roots(p)
```

Exercice 06

Soit à résoudre l'équation

$x^2 - 2x = 1$

1. Utiliser la fonction *fzero* pour trouver la première solution au voisinage de x1 = 1, puis utiliser *fzero* une nouvelle fois pour trouver la deuxième racine au voisinage de x2 = -3.

2. Utiliser la fonction *fsolve* pour trouver simultanément les deux racines sur l'intervalle [-3, 1].

3. Utiliser la fonction *solve* pour trouver les racines de cette équation.

Solution

1. Utilisation de *fsolve* pour trouver les racines de l'équation une par une :

```
>> f = @(x) x.^2 -2*x - 1;
>> t = -1:0.1:3;
>> plot(t,f(t)),grid
On voit que cette fonction s'annule à 2 reprises dans
l'intervalle [-1, 3] :
>> x1 = fzero(f,1)
x1 =
-0.4142
>> x2 = fzero(f,-3)
x2 =
-0.4142
2. Utilisation de fsolve pour trouver toutes les racines
simultanément :

>> fsolve(f,[-1,3])
ans =
-0.4142 2.4142
3. Utilisation de solve :

>> syms x
```

```
>> f = x^2 -2*x-1 ;
>> ezplot(f,[-2,3])
>> X = solve(f)
X =
2^(1/2) + 1
1 - 2^(1/2)
>> subs(X)
ans =
2.4142
-0.4142
```

TP05 : Graphisme en Matlab

Principe et objectifs de TP 05 :

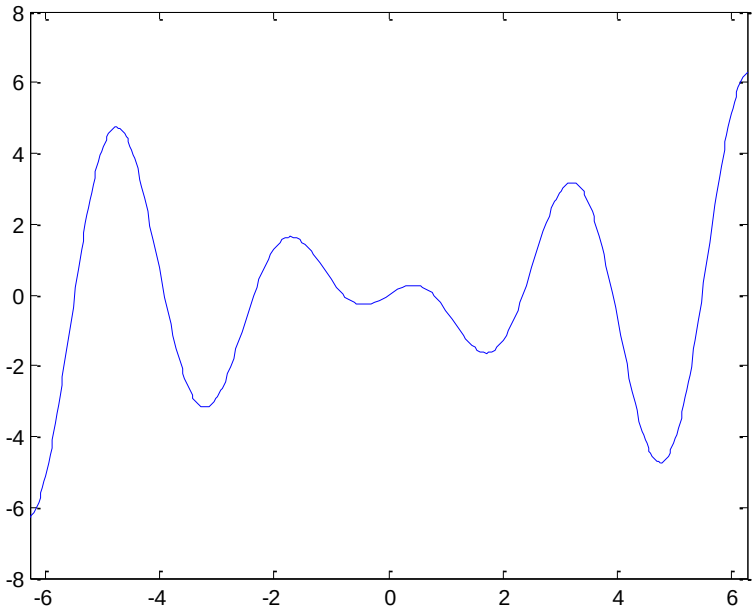
Les bibliothèques de Matlab proposent un très grand nombre de fonctionnalités pour la manipulation d'objets graphiques. Nous ne présentons ici que quelques principes de base utiles pour la visualisation des courbes. Si nous nous concentrons particulièrement sur la représentation graphique à 2 dimensions, il est possible d'aller bien plus loin : graphismes 3D (courbes, maillages, surfaces…).

Exercice 01

tracer le graphe de la fonction f(x)= xcos(2x) dans [-2pi 2pi]

Solution

```
>> fplot('x*cos(2*x)',[-2*pi 2*pi])
```



Pour tracer le graphe de la fonction h(x), on peut définir la fonction utilisateur h dans le fichier h.m de la manière suivante :

function y=h(x)

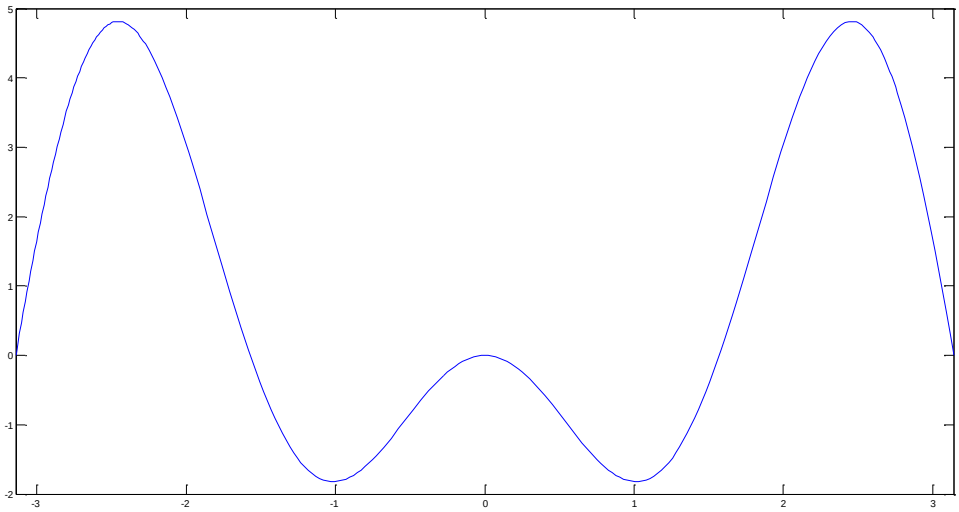
```
y=….;
>>fplot('h', [-x x]).
```

L'autre façon de procéder est d'exécuter l'instruction : `fplot('h', [-x x]).`

Exemple

tracer le graphe de la fonction `2*x*sin(-2*x) dans [-pi pi]`

```
function y=h11(x)
y='2*x*sin(-2*x)';
ans =
2*x*sin(-2*x)
>> fplot('2*x*sin(-2*x)', [-pi pi])
```



Exercice 02

Écrire le programme Matlab qui permet de tracer sur le même graphe les fonctions F1, F2 et F3 sur l'intervalle [-1 1].

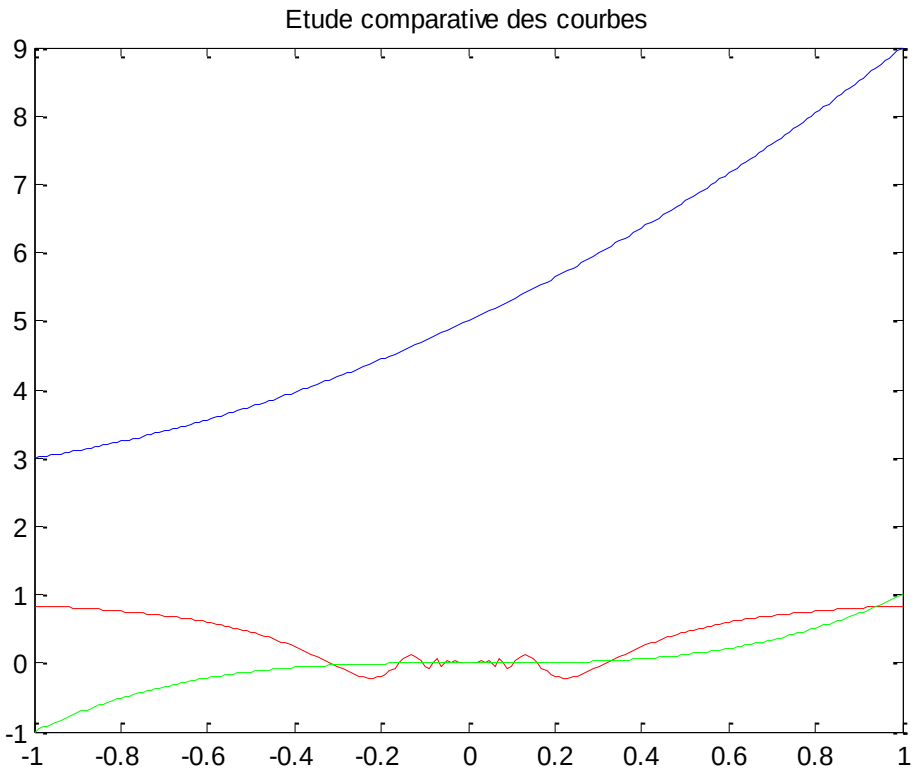
`f1=x._sin(1./x), f2=x.^3, f3=x.^2+3_x+5.`

Avec comme titre 'Étude comparative des courbes'

Solution

```
clc;
 clear;
 % Cree les vecteur x
 x=[_1:0.01:1]
 % Calculer F1(x),F2(x) et F3(x)
 f1=x._sin(1./x) % n'oubliez pas le point avant la division
 f2=x.^3
f3=x.^2+3_x+5
 % Representer les trois courbes
plot(x,f1,'r')
 hold on
 plot(x,f2,'g')
 plot(x,f3)
```

```
title('Etude comparative des courbes')
 hold of
```

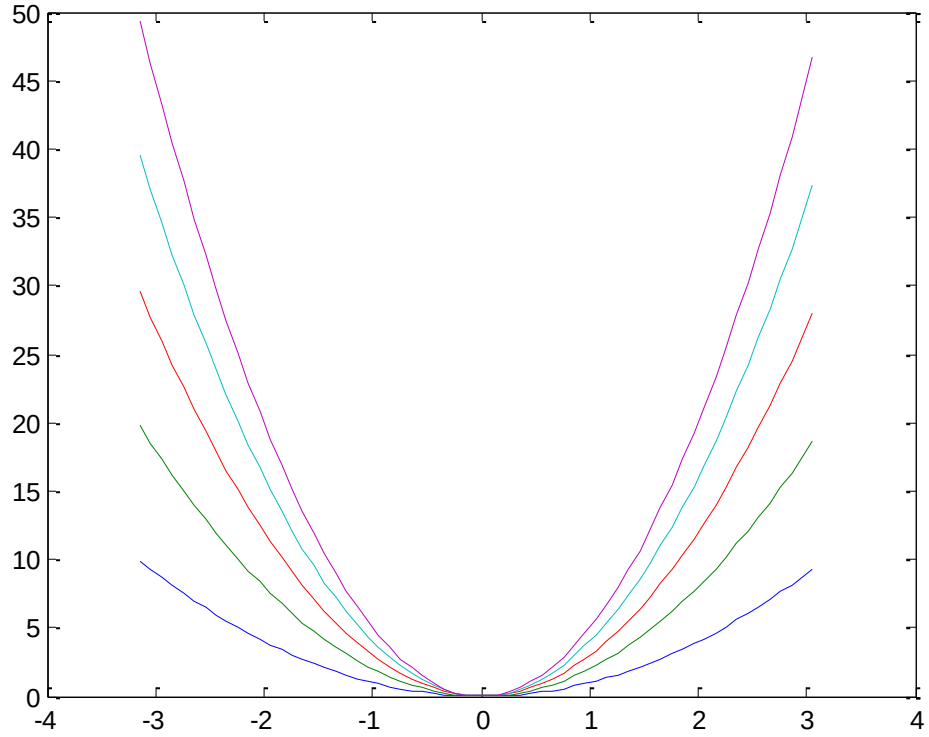


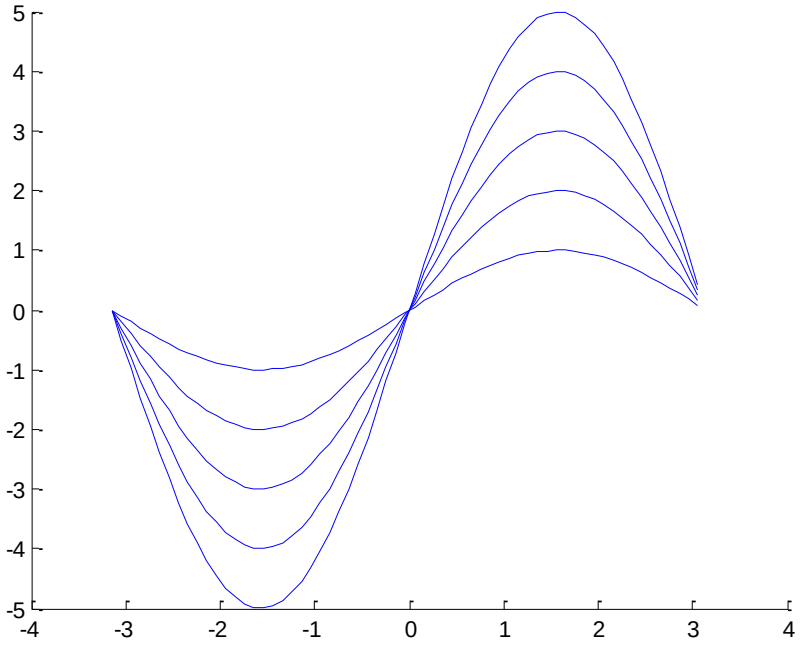
Exercice 03

Écrire un programme qui trace plusieurs courbes un seul variablement utilisant la commande Holde on.

Solution

```
Clc,clear all
x=[-pi:0.1:pi] ;
y1= x.^2 ; y2= 2*x.^2 ; y3= 3*x.^2 ;
y4= 4*x.^2 ; y5= 5*x.^2 ;
figure ; plot(x,y1,x,y2,x,y3,x,y4,x,y5);
figure; hold on;
for i=1:5
y=i*sin(x) ;
fig4=plot(x,y);
end
```





Exercice 04

Écrire un programme qui trace plusieurs courbes par la command **subplot(m,n,i)** si f1=cos(2*x), f2=sin(3x), f3=tan(x+1), f4=acos(x^2) f5=asin(x^2), f6=atan(x^2-1)

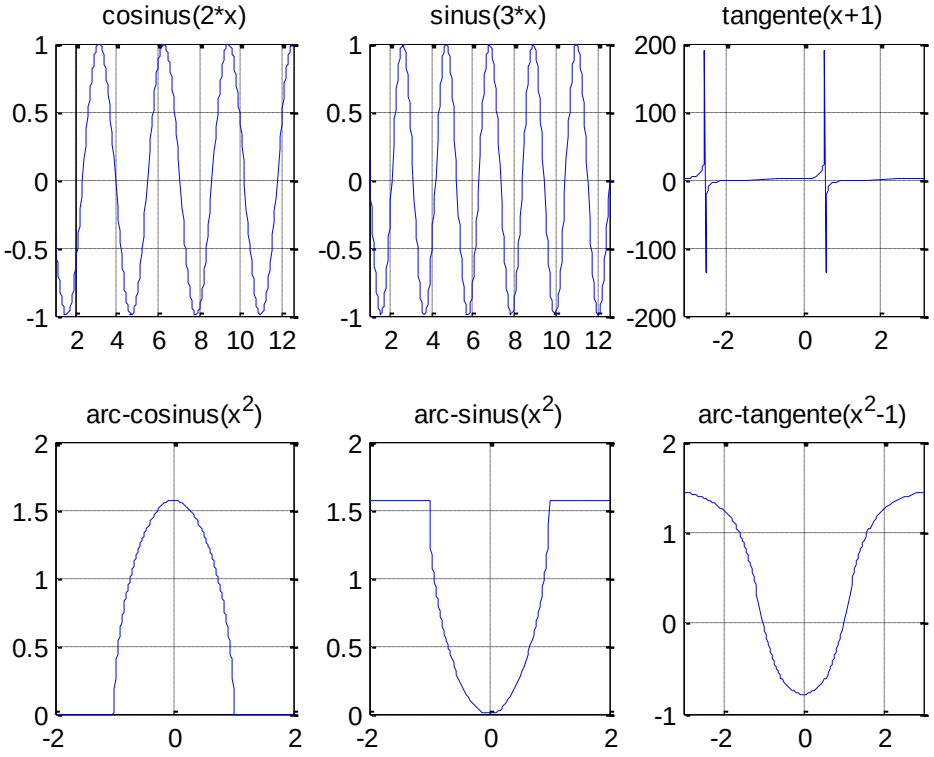
La syntaxe est **subplot(m,n,i)** où

- **m** est le nombre de sous-fenêtres verticalement ;
- **n** est le nombre de sous-fenêtres horizontalement ;
- **i** sert à spécifier dans quelle sous-fenêtre doit s'effectuer l'affichage.

Les fenêtres sont numérotées de gauche à droite et de haut en bas.

Solution

```
Clc, clear all
figure
subplot(2,3,1), fplot('cos(2*x)',[1 4*pi]),
title('cosinus(2*x)'), grid
subplot(2,3,2), fplot('sin(3*x)',[1 4*pi]),
title('sinus(3*x)'), grid
subplot(2,3,3), fplot('tan(x+1)',[-pi pi]),
title('tangente(x+1)'), grid
subplot(2,3,4), fplot('acos(x^2)',[-2 2]), title('arc-
cosinus(x^2)'), grid
subplot(2,3,5), fplot('asin(x^2)',[-2 2]), title('arc-
sinus(x^2)'), grid
subplot(2,3,6), fplot('atan(x^2-1)',[-sqrt(9) sqrt(9)]),
title('arc-tangente(x^2-1)'), grid
```



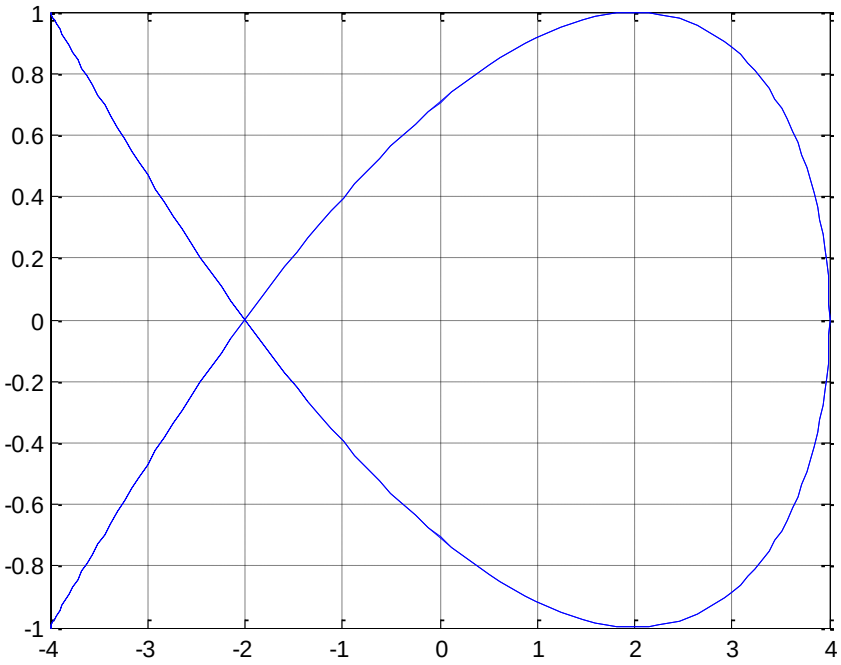
Exercice 05

Écrire un programme qui trace le courbe d'équations paramétriques d'une ellipse :

x = 4cos(2t),y = sin(3t).

Solution

```
t = 0 : pi/200 : 2*pi
x = 4*cos(2*t)
y = sin(3*t)
plot ( x , y )
grid on
```

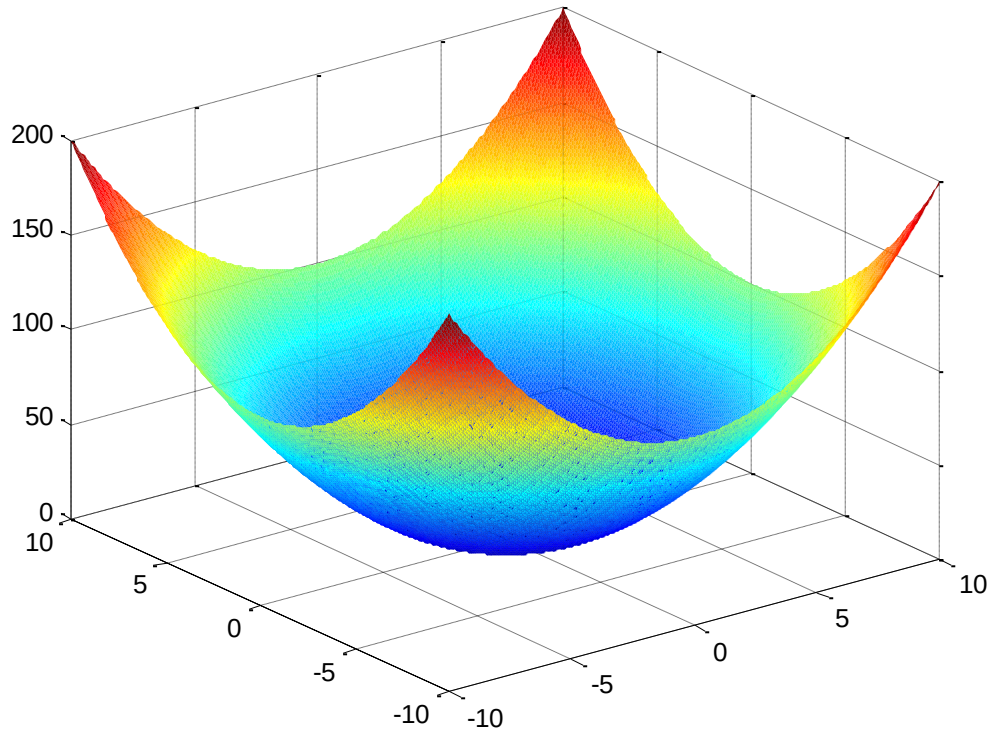


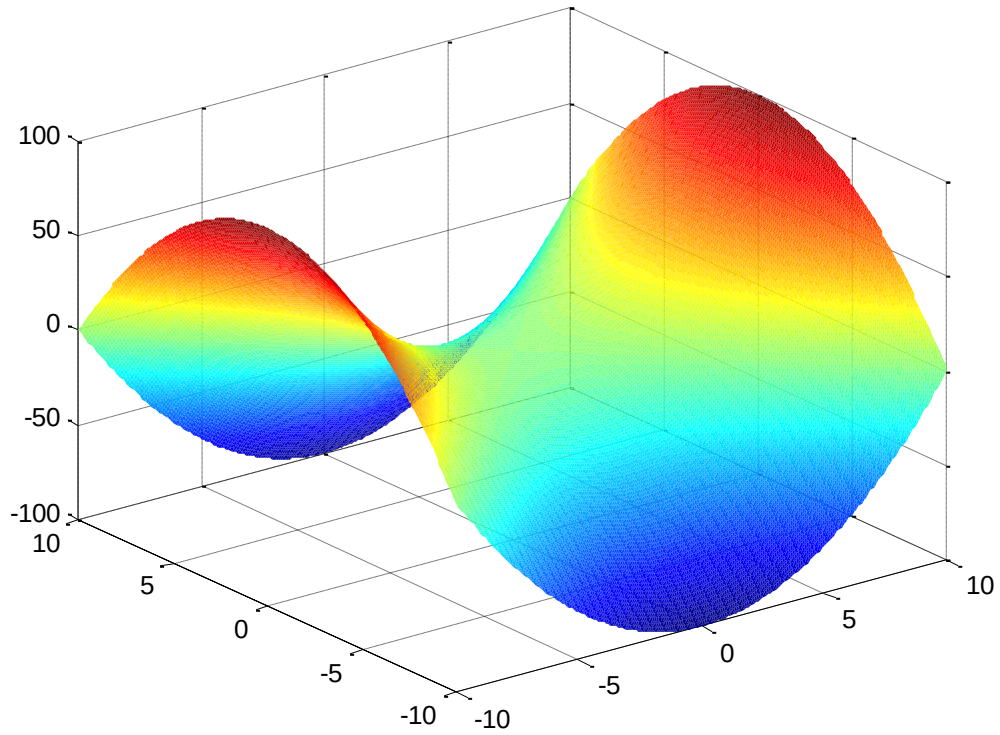
Exercice 06

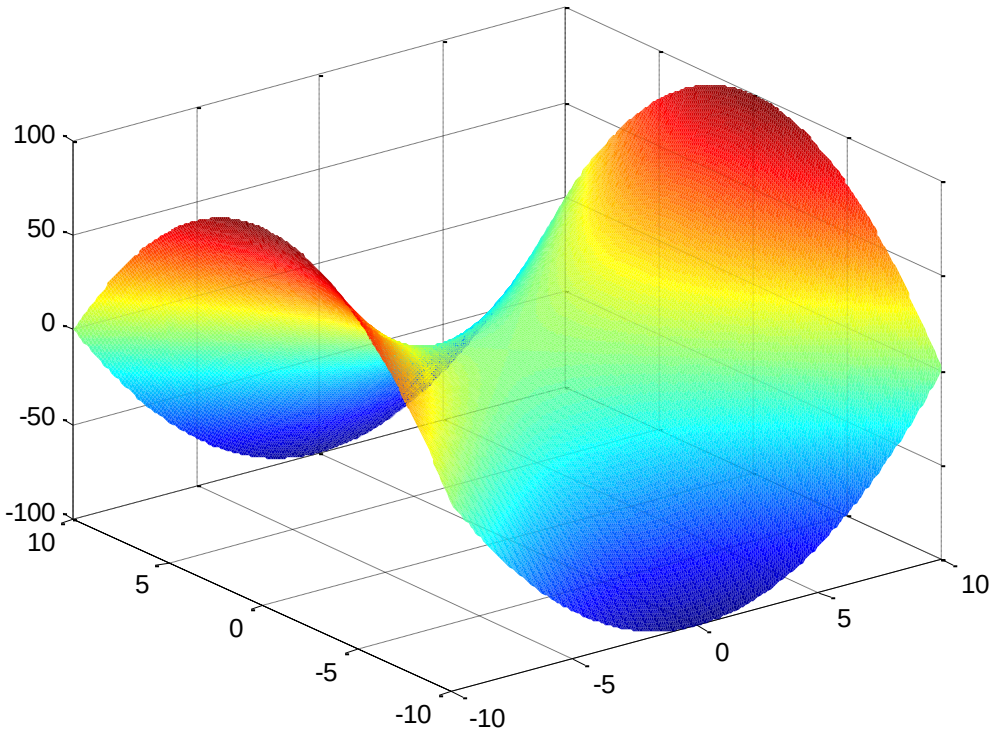
Écrire un programme qui trace les courbes à deux variables .z = f(x, y) = $x^2 \mp y^2$.

Solution

```
clc ; clear all ; close all ;
[x,y]=meshgrid(-10:0.1:10,-10:0.1:10);
%==============================
z = x.^2 + y.^2 ;
figure ; surf(x,y,z,'lineStyle','none');
%==============================
z = x .^2 -y .^2 ;
figure ; surf(x,y,z,'lineStyle','none');
%==============================
r = sqrt(x.^2 + y.^2) ;
z = sin(r)./r;
figure ; surf(x,y,z,'lineStyle','none');
```


200
150
100
50
0
10
5
0
-5
-10
-10
-5
0
5
10


100
50
0
-50
-100
10
5
0
-5
-10
-10
-5
0
5
10

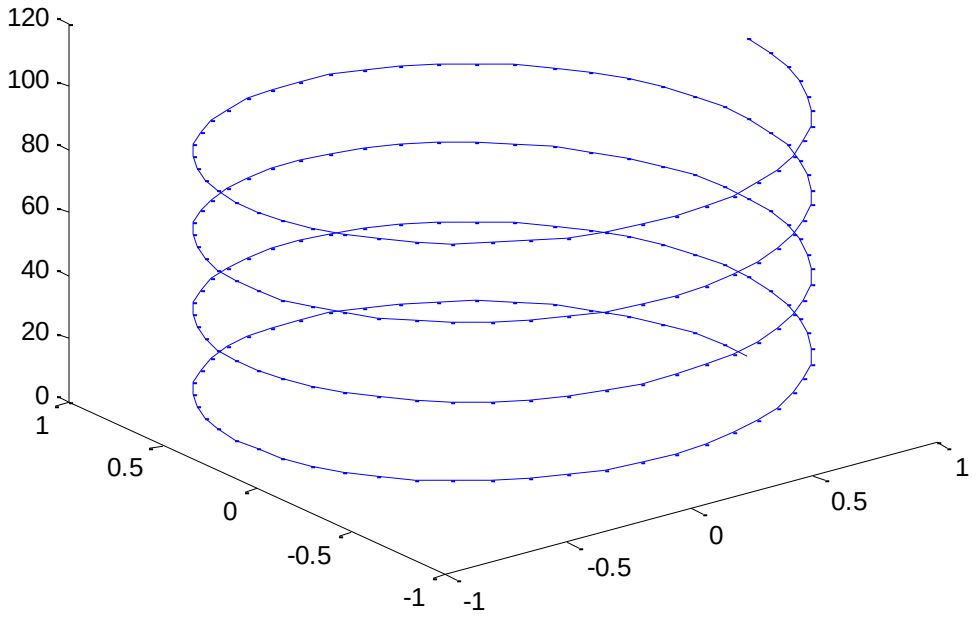


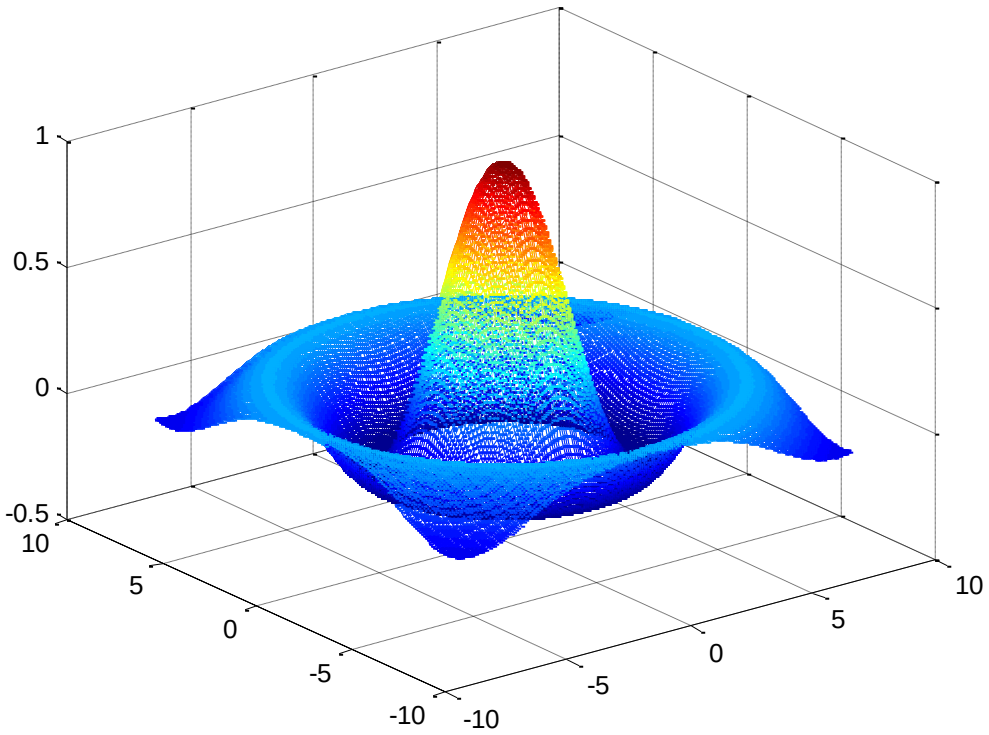
Exercice 07

1. Dessiner la courbe 3D suivante :x = cos(t); y = sin(t); z = 4t;
2. Dessiner la surface de la fonction à deux variables : $Z = \frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$

Solution

```
t = 0 : 8*pi/200 : 8*pi;

x = cos(t); y = sin(t); z = 4*t;

figure ; plot3(x,y,z)
x = -8 : 0.5 : 8; y = x ;

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

R = sqrt(X.^2 + Y.^2) + eps ; % ajouter eps pour éviter R=0

Z = sin(R) ./ R ;

mesh(X, Y, Z) % Matlab accepte aussi mesh(x,y,Z)
```


120
100
80
60
40
20
0
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

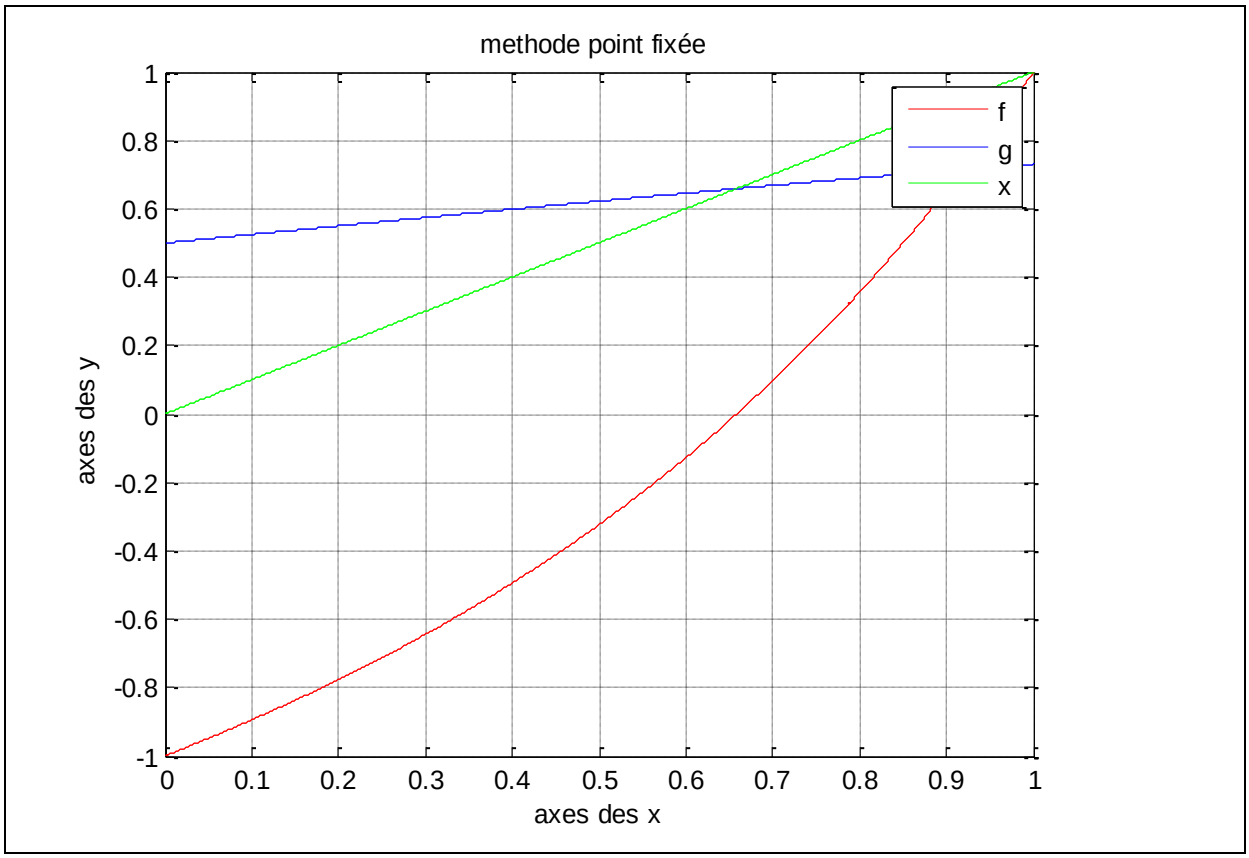

1
0.5
0
-0.5
10
5
0
-5
-10
-10
-5
0
5
10

Exercice 08

En utilisant la méthodes de point fixée calculé la valeur approchée de la racine de f(x) = x(exp(x)+1)-exp(x) et tol=10^-5 et Nbrit=30(nombre maximal d'itérations), x0=1(valeurs initiales)x=exp(x)./(exp(x)+1).

Solution

```
Clc,clear all
x0=1;N=0;eps=10^-4;Nb=20;
 while N<Nb
N=N+1;
x1=exp(x0)/(exp(x0)+1);
x0=x1;
if abs(x1-x0)<eps
soll=x1;
end
 end
 'soloution approchee f(x) '
soll
x=0:0.0005:1;
f=x.*(exp(x)+1)-exp(x);
g=exp(x)./(exp(x)+1);
plot(x,f,'r')
hold on
plot(x,g,'b')
plot(x,x,'g')
grid on
legend( 'f','g' , 'x' )
 title ( 'methode point fixée ')
 xlabel ( 'axes des x')
 ylabel ( 'axes des y ')
 hold on

ans =

soloution approchee f(x)


soll =

     663/1006
```



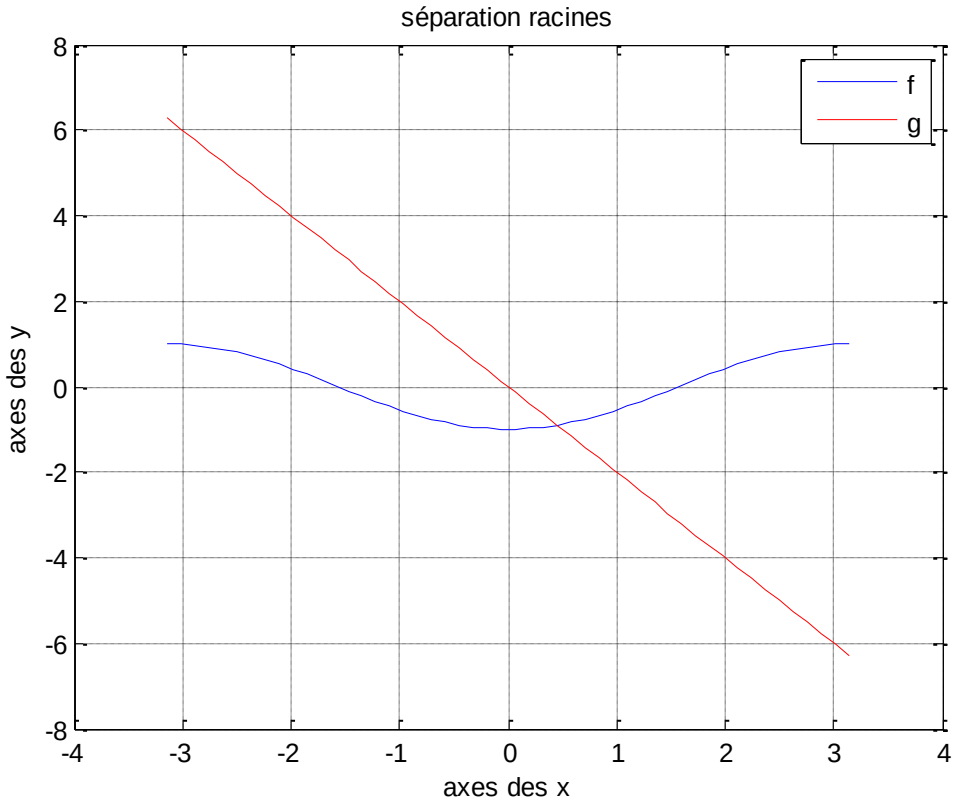
Exercice 09

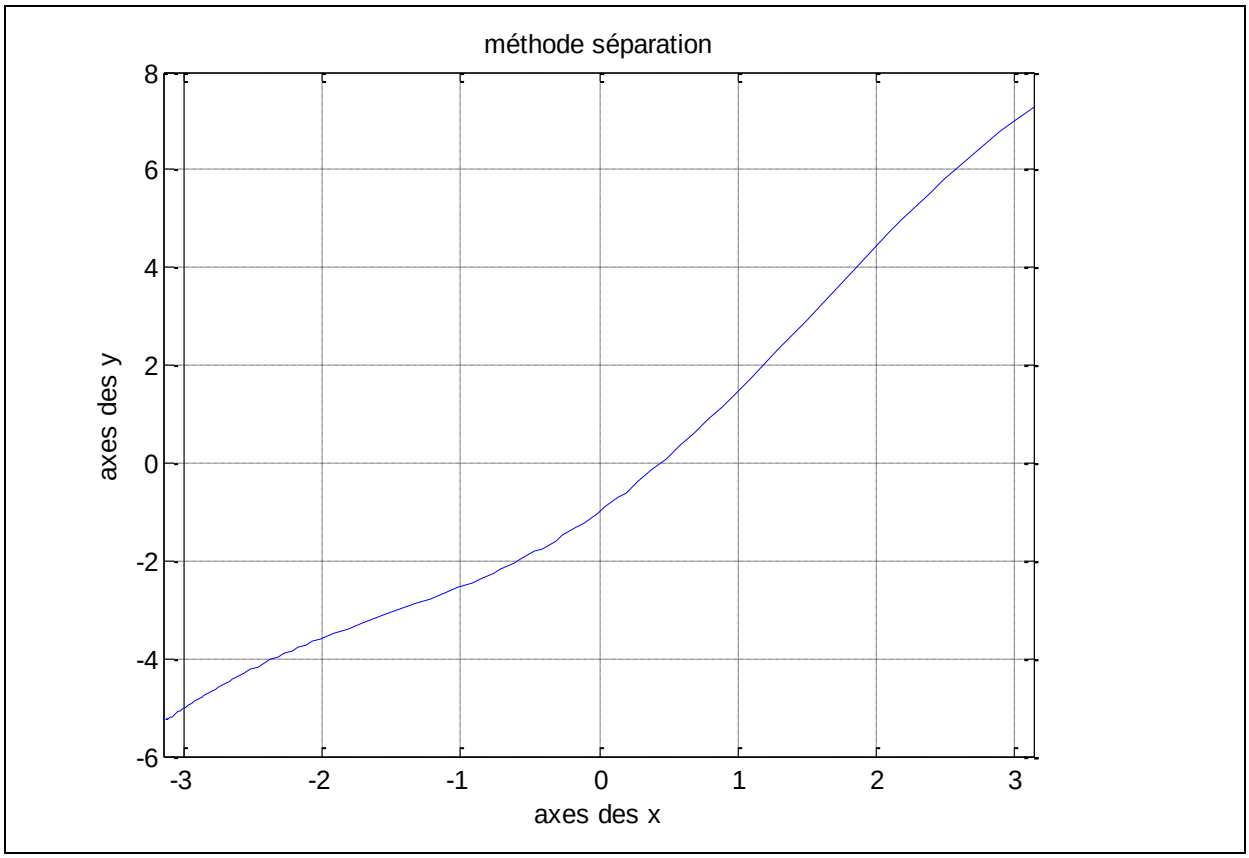
En utilisant les fonctionnalités graphiques de MATLAB, localiser la racine positive de l'équation f(x)=2*sin(x)-x.

Solution

```
%%% Methode 1:
clear ;          %effacer les variables en cours
clc ;            %effacer l'ecran
f=inline('-cos(x)+2*x');
figure(1)          % %donner un numero a la figure
fplot(f,[-pi, pi])
X=fzero(@(x)(-exp(x)+x.^2+2),0);
title('méthode séparation')
xlabel('axes des x')       %ajouter une etiquete a l'axe des x
ylabel('axes des y')       %ajouter une etiquete a l'axe des x
grid on                 %geler la figure pour ajouter un %autre
graphe
                                                              %%%
% Methode 2:
x=linspace(-pi,pi,50);
f=-cos(x);
g=-2*x;
figure(2)
plot(x,f)
```

```
hold on
plot(x,g,'r')    %representer f en fonction de x %en utilisant
la couleur rouge
title('séparation racines')
grid on
legend('f','g')    %etiquete des deux courbe
xlabel('axes des x')
ylabel('axes des y')
hold on    %fait apparaitre la grille
```





TP6 : Calcul symbolique

Exercice 01

Résolution numérique avec MATLAB de l'équation de la thermocline (océan pacifique)

$$\frac{dh(t)}{dt} = -\tanh\left[\kappa\, h(t-\tau)\right] + b\,\cos(2\pi\, t),\ t \geq 0,$$
$$h(t) = 1 \qquad \text{for } t \in [-\tau,\ 0), \quad \phi(t) \in X.$$

k=100; b=1; $\tau = 0.01$

avec t variant de 0 à 10

Solution :

```
tau = 0.01;
dt = 0.01;
k = 100;
b = 1;
t = [-tau:dt:10];
h(1) = 1;
h(2) = 1;
```

```
for i = 2:length(t)-1
   h(i+1) = dt*(-tanh(k*h(i-1))+b*cos(2*pi*t(i)))+h(i);
end
plot(t, h)
```

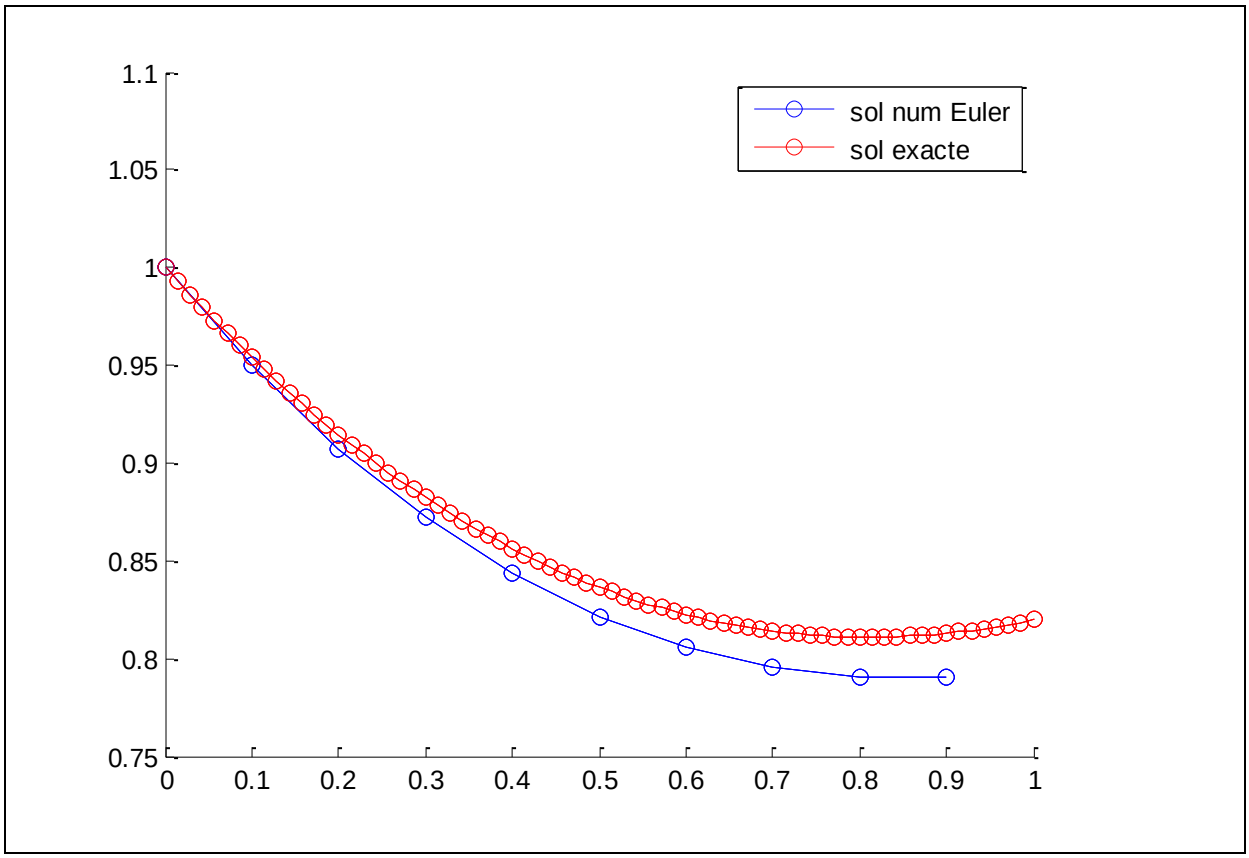
Exercice 02

Résoudre numériquement, par le biais des méthodes d'Euler, l'équation différentielle du premier ordre et compares le résulta par la solution exacte y suivante :

$$\begin{cases} y' = \dfrac{(t-y)}{2} \\ y(0) = 1 \end{cases}, y(t) = 3e^{-t/2} + t - 2.$$

Solution

```
clear all ;
clc;
 dydt=inline('(t-y)/2','t','y');

 a=0;b=1;n=10;h=(b-a)/n;t=a:h:b;
 epsilon=0.0001;u(1)=1+epsilon;

 for i=1:n-1
u(i+1)=u(i)+h*(dydt(t(i),u(i)));
end
 hold on

plot(t(1:end-1),u,'o-b')

clear all ;
clc;
a=0;b=1;n=70;h=(b-a)/n;t=a:h:b;
dydt=inline('(t-y)/2','t','y');%EQUATION?DIFFERENTIELLE
yt=inline('3*exp(-t/2)-2+t','t');%SA?SOLUTION?EXACTE
plot(t,yt(t),'o-r')

legend(' sol num Euler',' sol exacte')
```



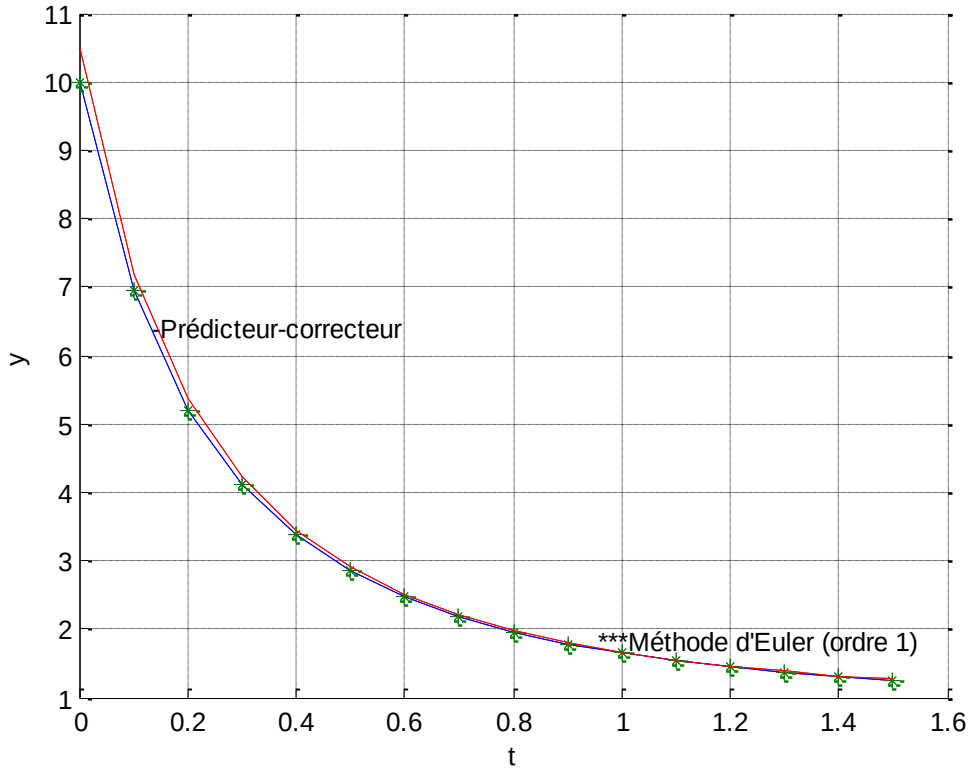
Exercice 03

Résolution d'une équation différentielle du 1er ordre par deux méthodes (Méthode du prédicateur-correcteur, Méthode d'Euler à l'ordre 1)

Solution:

```
Clear all;
clc;
hold off;
% Méthode d'Euler à l'ordre 1
y1(1)=10;t(1)=0;h=0.1;n=1;
while t(n)<1.5
n=n+1;
t(n)=t(n-1)+h;
y1(n)=y1(n-1)+h*(-y1(n-1).^1.5+1);
end
% Méthode du prédicteur-correcteur
ym(1)=10.5;t(1)=0;h=0.1;n=1;e=1e-4;n1=10000;
while t(n)<1.5
n=n+1;
t(n)=t(n-1)+h;
ym(n)=ym(n-1)+h*(-ym(n-1).^1.5+1);
% Itération succéssives et substitution
```

```
for k=1:n1
ymb=ym(n)+0.5*h*(-(ym(n)).^1.5-(ym(n-1)).^1.5+2);
if abs(ym(n)-ymb)<=e;
break;
printf('Erreur de (%d)=%f\n',k,abs(ym(n)-ymb));
end
if k==n1
fprintf('Pas de convergence pour t=%f\n',t(n));
end
end
end
fprintf('t=%f\t y1=%f\t ym=%f\n',t,y1,ym);
grid on;
hold on;
plot(t,y1,t,y1,'*');
plot(t,ym,'-r');
xlabel('t');ylabel('y');
gtext('***Méthode d''Euler (ordre 1)');
gtext('-Prédicteur-correcteur');
```



Exercice 04

Résolution d'une équation différentielle du 1er ordre par méthodes Euler et Runge-Kutta 4 et comparaison avec la solution exacte

Solution :

```
% Résolution d'une équation différentielle ordinaire  *
% du 1er ordre par  méthodes Euler etRunge-Kutta 4 et
coparaison avec
%la soloution exacte  *
clear all ;
clc;
dydt=inline('(t-y)/2','t','y');
 a=0;b=3;n=80;h=(b-a)/n;t=a:h:b;
epsilon=0.0001;u(1)=1+epsilon;
for i=1:n-1
u1(i)=u(i);u2(i)=u(i)+h/2*dydt(t(i),u1(i));
u3(i)=u(i)+h/2*dydt(t(i)+h/2,u2(i));u4(i)=u(i)+h*dydt(t(i)+h/2
,u3(i));
u(i+1)=u(i)+h/6*(dydt(t(i),u1(i))+2*dydt(t(i)+h/2,...
u2(i))+2*dydt(t(i)+h/2,u3(i))+dydt(t(i+1),u4(i)));
end
hold on
plot(t(1:end-1),u,'o-g')
xlabel('Temps (en s)');
ylabel('Vitesse (en m/s)');
clear all ;
clc;
 dydt=inline('(t-y)/2','t','y');
 a=0;b=3;n=80;h=(b-a)/n;t=a:h:b;
 epsilon=0.0001;u(1)=1+epsilon;
 for i=1:n-1
u(i+1)=u(i)+h*(dydt(t(i),u(i)));
end
 hold on
plot(t(1:end-1),u,'o-b')
xlabel('t');
ylabel('y(t)');
clear all ;
clc;
a=0;b=3;n=80;h=(b-a)/n;t=a:h:b;
dydt=inline('(t-y)/2','t','y');%EQUATION?DIFFERENTIELLE
yt=inline('3*exp(-t/2)-2+t','t');%SA?SOLUTION?EXACTE
plot(t,yt(t),'r')
```

```
legend('Runge-Kutta','Euler','exacte')
```