Univ. OEB
Dept. MI
2ème année maths
(Durée : 1 heure 30 min)

Examen Rattrapage d'OPM2

In [ ]:

Exercice 1. (5 pts)

- Convertir le nombre décimal x = 13.25 en virgule flottante suivant la norme IEEE-754 (32 bits)

In [ ]:

Exercice 2. (5 pts)

Le programme suivant calcule la racine carrée d'un nombre par la méthode de Newton.

```
a=input('donner un nombre positif:');
x=a/2;
precision = 6;
for i = 1:precision
x = (x + a / x) / 2;
end
disp(x)
```

1. Remplacer l'instruction **for** par l'instruction **while** en préservant la fonctionnalité du programme.
2. Modifier le programme pour qu'il soit applicable sur un vecteur et pas uniquement sur un nombre.
3. Transformez le programme en une fonction.

In [ ]:

Exercice 3. (5 pts)

Écrire une fonction antidiag(x,y,z) qui a pour arguments les trois vecteurs x = (x1, x2, . . . , x6), y = (y1, y2, . . . , y5), z = (z1, z2, . . . , z5) et qui retourne la matrice :

A =

```
0    0    0    0    z1   x1
0    0    0    z2   x2   y1
0    0    z3   x3   y2   0
0    z4   x4   y3   0    0
z5   x5   y4   0    0    0
x6   y5   0    0    0    0
```

In [ ]:

Exercice 4. (5 pts)

On veut représenter la fonction :
$f : t \dashrightarrow f(t) = (\exp(-t/10)\sin(t), \exp(-t/10)\cos(t), \exp(-t))$ sur $[-2\pi, 2\pi]$.

1. Créez la fonction f.
2. Donner la commande Matlab permettant de représenter la fonction f avec une variation de t de -2π jusqu'à 2π et un pas $2\pi/100$.

Bonne Chance

In [ ]:

Corrigé type

In [ ]:

Exercice 1. (5 pts)

- Convertir le nombre décimal x = 13.25 en virgule flottante suivant la norme IEEE-754 (32 bits)

In [ ]:

Solution :

In [ ]:

- Partie entière : 13 ==> 1101
- Partie décimale : 0.25 ==> 0.01
- (13.25)_10 = (1101,01)_2
- Normalisation IEEE-754 : => 1,10101 x 2^3

  (de la forme 1,xxxx où xxxx = mantisse)

Décomposition du nombre en ses divers éléments :

- Bit de signe : 0 (Nombre positif)
- Exposant sur 8 bits biaisé à 127 ==> 3 + 127 = 130 ==> 1000 0010
- Mantisse sur 23 bits : 1010 100 0000 0000 0000 0000
- Signe Exposant-biaisé Mantisse

  0 1000 0010 1010 1000 0000 0000 0000 000

In [2]:

```
ieee32_conv(13.25)
```

```
ans = 01000001010101000000000000000000
```

In [ ]:

Exercice 2. (5 pts)

Le programme suivant calcule la racine carrée d'un nombre par la méthode de Newton.

```
a=input('donner un nombre positif:');
x=a/2;
precision = 6;
for i = 1:precision
x = (x + a / x) / 2;
end
disp(x)
```

1. Remplacer l'instruction **for** par l'instruction **while** en préservant la fonctionnalité du programme.
2. Modifier le programme pour qu'il soit applicable sur un vecteur et pas uniquement sur un nombre.
3. Transformez le programme en une fonction.

In [ ]:

Solution :

In [22]:
```
%1.---
a=input('donner un nombre positif:');
x=a/2;
precision = 6;
i = 1;
while i <= precision
x = (x + a / x) / 2;
i = i+1;
end
disp(x)

%2.---
a=input('donner un vecteur nombres positifs:');
x=a/2;
precision = 6;
for i = 1:precision
x = (x + a ./ x) / 2; % opérations élément par élément
end
disp(x)

%3.---
function x=racine(a)
x=a/2;
precision = 6;
for i = 1:precision
x = (x + a ./ x) / 2;
end
end
```

```
donner un nombre positif:10
3.1623
donner un vecteur nombres positifs:1:5
   1.0000   1.4142   1.7321   2.0000   2.2361
```

In [8]:
```
Exercice 3. (5 pts)

Écrire une fonction antidiag(x,y,z) qui a pour arguments les trois vecteurs
x = (x1, x2, . . . , x6), y = (y1, y2, . . . , y5), z = (z1, z2, . . . , z5)
et qui retourne la matrice :

A =

    0    0    0    0    z1    x1
    0    0    0    z2   x2    y1
    0    0    z3   x3   y2    0
    0    z4   x4   y3   0     0
    z5   x5   y4   0    0     0
    x6   y5   0    0    0     0
```

In [ ]:
```
Solution :
```

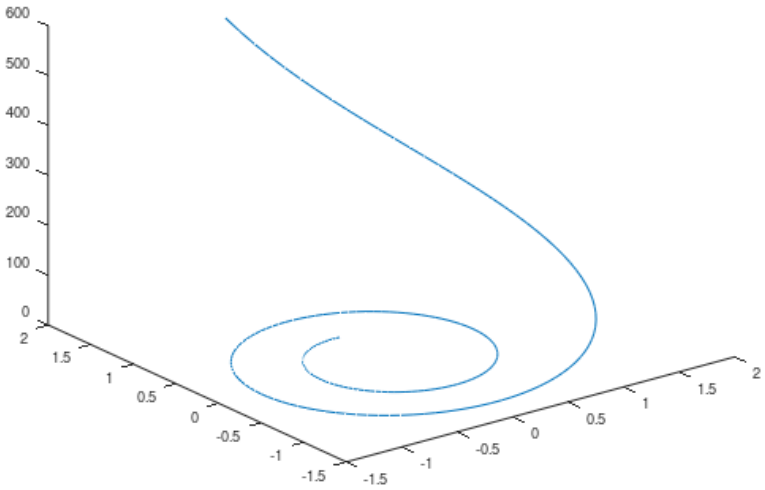
In [21]:
```
function A=antidiag(x,y,z)
mat1 = diag(x);
mat2 = diag(y,-1);
mat3 = diag(z,1);
B = mat1 + mat2 + mat3;
A = B(:,end:-1:1);
end
x=1:6; y=1:5;z=-5:-1;
A=antidiag(x,y,z)
```

```
A =

   0   0   0   0  -5   1
   0   0   0  -4   2   1
   0   0  -3   3   2   0
   0  -2   4   3   0   0
  -1   5   4   0   0   0
   6   5   0   0   0   0
```

In [ ]:
```
Exercice 4. (5 pts)
On veut représenter la fonction :
   f : t --> f(t) = (exp(-t/10) sin(t), exp(-t/10) cos(t), exp(-t)) sur [-2π, 2π].

1. Créez la fonction f.
2. Donner la commande Scilab permettant de représenter la fonction f
avec une variation de t de -2π jusqu'à 2π et un pas 2π/100.
```

In [ ]:
```
Solution :
```

In [18]:
```
function [x,y,z] = f(t)
x=exp(-t/10).*sin(t);
y=exp(-t/10).*cos(t);
z=exp(-t);
endfunction
t = -2*pi:2*pi/100:2*pi;
[x,y,z] = f(t);
plot3(x,y,z);
```



In [ ]: