MATLAB - TP n°6

Exercice 1

Soit à résoudre l'équation

$x^2 + 2x = 1$

1. Utiliser la fonction *fzero* pour trouver la première solution au voisinage de $x_1 = 1$, puis utiliser *fzero* une nouvelle fois pour trouver la deuxième racine au voisinage de $x_2 = -3$.
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.

Voir corrigé plus bas

Exercice 2

Trouver les racines des équations suivantes :

1. $x^2 - \cos(x) = 0$
2. $\frac{x^2}{1+x} = 0$
3. $\frac{10}{x}(1 - e^x) = 0$
4. $x^4 - 8x^3 + 18x^2 - 16x + 5 = 0$

Exercice 3

Trouver le vecteur x qui minimise la fonction $f(x) = -x_1\, x_2\, x_3$, demarrant au point $x = [10;\ 10;\ 10]$, et sujet aux contraintes : $0 \leq x_1 + 2x_2 + 2x_3 \leq 72$.

Voir corrigé plus bas

Exercice 4

Soit le système d'équations suivant :

$$\begin{cases} x^2 + 2y = 10 \\ x + y = 4 \end{cases}$$

1. Résoudre ce système à l'aide de la fonction *fsolve* en démarrant :
   a. à partir du point (1, 3)
   b. à partir du point (-1, 3)
2. Après avoir déclaré des variables symboliques *x* et *y*, résoudre ce système en faisant appel à la fonction *solve*.
   Comparer les solutions obtenues en 1) et 2).

Voir corrigé plus bas

Exercice 5

Résoudre le système d'equations non lineares suivant :

$$\begin{cases} e^{-e^{-(x_1+x_2)}} - x_2(1 + {x_1}^2) = 0 \\ x_1 \cos(x_2) + x_2 \sin(x_1) - \frac{1}{2} = 0 \end{cases}$$

Voir corrigé plus bas

Exercice 6

Reprendre le programme non linéaire du cours :

$$\begin{cases} \min\ f(x) = -2x_1x_2 - 5{x_3}^2 \\ \text{avec} \\ 3x_1 + x_2 + 4x_3 \leq 30 \\ 5x_1 \quad + \ x_3 \leq 25 \end{cases}$$

puis invoquer cette fois-ci la fonction *fmincon* de la manière suivante :

```
>> [x, fx, ExitFlag, Output] = fmincon(func1, x0, A,b)
```

Exercice 7

Résoudre le programme quadratique suivant :

$$\begin{cases} \min g(x) = x_1^2 + x_1x_2 + 2x_2^2 + 2x_3^2 + 2x_2x_3 + 4x_1 + 6x_2 + 12x_3 \\ avec \\ x_1 + x_2 + x_3 \geq 6 \\ -x_1 - x_2 + 2x_3 \geq 2 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

en utilisant :

1. d'abord la fonction *fmincon* ;
2. puis en utilisant la fonction *quadprog* conçue spécialement pour les programmes quadratiques (voir le help de la fonction *quadprog)*.

Exercice 8

Soit le programme linéaire booléen (les valeurs possibles des x_i sont 0 ou 1 uniquement) ci-dessous. Retrouver les valeurs des variables x_1 … x_4 qui minimisent la fonction *f(x)* tel que x_1… x_4 vérifient les 4 contraintes suivantes :

$$\begin{cases} \min f(x) = -5x_1 - 2x_2 + 5x_3 - 7x_4 \\ avec \\ 2x_1 + 7x_2 + 3x_3 + x_4 \leq 5 \\ x_2 + \quad x_4 \leq 0 \\ x_1 \quad - x_3 \quad \leq 1 \\ x_1 \quad x_2 \quad - x_4 \leq 3 \\ et \\ x_1, x_2, x_3, x_4 \in \{0, 1\} \end{cases}$$

Indication : Pour résoudre ce programme linéaire booléen, rédiger et éxécuter le script suivant :

```
f = [-5; -2; 5; -7];
A = [2 7 3 1; 0 1 0 1; 1 0 -1 0; 1 1 0 -1];
b = [5; 0; 1; 3];
[x, fx, ExitFlag, Output] = bintprog(f,A,b)
```

Exercice 9

Soit le nuage de points suivant :

data = [-9 1; 11.6 5; 21.7 10; 32.4 20; 44.1 40; 51.7 60; 61.5 100; 75.9 200; 91.4 400; 108 760;];

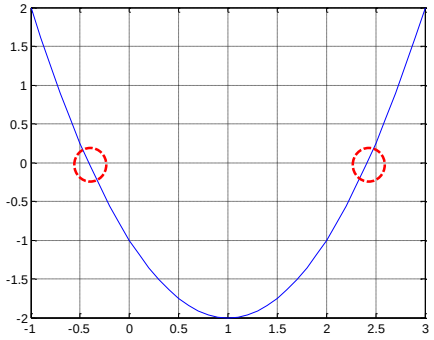
Retrouver le polynôme d'ordre 2 qui représente au mieux ce nuage à l'aide à l'aide de la fonction *polyfit*.

Voir corrigé plus bas

Corrigé Exercice 1

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] :

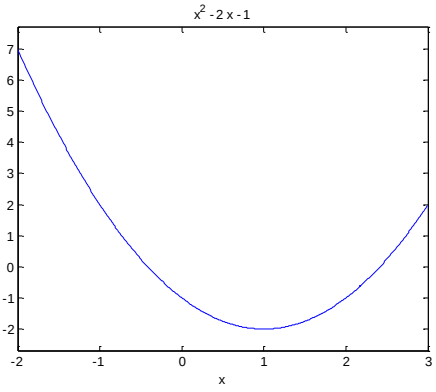
```
>> x1 = fzero(f,1)
x1 =
    -0.4142
>> x2 = fzero(f,-3)
x2 =
    -0.4142
```

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
```



Corrigé Exercice 3

Premièrement, écrire un fichier M-file qui renvoie un scalaire *f* qui est une évaluation de *f(x)* au point *x* :

```
function f = func2(x)
f = - x(1) * x(2) * x(3);
```

Puis réécrire les contraintes sous la forme :

$$\begin{cases} x_1 + 2x_2 + 2x_3 \leq 72 \\ -x_1 - 2x_2 - 2x_3 \leq 0 \end{cases}$$

Puisque toutes les contraintes sont linéaires, formuler les par l'écriture matricielle :

$$Ax \leq b$$

où :

$$A = \begin{bmatrix} 1 & 2 & 2 \\ -1 & -2 & -2 \end{bmatrix}, \quad \text{et} \quad b = \begin{bmatrix} 72 \\ 0 \end{bmatrix}$$

Puis, créer la matrice *A* et les vecteurs *b* et x_0, et invoquer la routine d'optimisation avec contrainte *fmincon* :

```
>> A = [1 2 2; -1 -2 -2 ] ;
>> b = [72 ; 0 ] ;
```

```
>> x0 = [10;  10;  10] ;
>> [x,fx, ExitFlag, Output] = fmincon(@func2, x0, A, b)
```

Ici, on demande à la fonction *fmincon* de retourner :

- *x* : la solution optimale trouvée ;
- *fx* : la valeur de f au point x ;
- *ExitFlag* : indique comment *fmincon* a terminé (voir le help pour les différentes valeurs de *ExitFlag*) ;
- *Output* : donne des détails supplémentaires sur l'exécution de *fmincon* (nombre d'itérations, nombre de fois où la fonction *f(x)* a été évaluée, l'algorithme utilisé, etc.)

Apres 12 itérations, le résultat renvoyé par Matlab est :

```
x =
   24.0000
   12.0000
   12.0000
fx =
  -3.4560e+03
ExitFlag =
     5
Output =
         iterations: 12
          funcCount: 53
       lssteplength: 1
           stepsize: 4.6551e-05
          algorithm: 'medium-scale: SQP, Quasi-Newton, line-
                      search'
      firstorderopt: 4.7596e-04
    constrviolation: 0
            message: [1x772 char]
```

Corrigé Exercice 4

1. Créer le fichier M-file func4.m suivant :

```
function f = func4(t)
f(1) = t(1)^2 + 2*t(2) - 10;
f(2) = t(1) + t(2) - 4;
```

   a. Saisir les commandes suivantes à la ligne de commande

```
>> S01 = [1 3];
>> [S1,FS1, ExitFlag] = fsolve(@func4,S1)
S1 =
   -0.7321    4.7321
FS1 =
   1.0e-11 *
    0.6699   -0.0008
ExitFlag =
     1
```

   b. Saisir les commandes suivantes à la ligne de commande :

```
>> S02 = [ -1 3 ];
>> [S2,FS2, ExitFlag] = fsolve(@func4,S02)
S2 =
```

```
   -0.7321    4.7321
FS2 =
   1.0e-11 *
    0.6699   -0.0008
ExitFlag =
     1
```

2. Saisir les commandes suivantes à la ligne de commande :

```
>> syms x y
>> f1 = x^2 + 2*y -10 ;
>> f2 = x + y -4 ;
>> [X,Y] = solve(f1,f2)
X =
 3^(1/2) + 1
 1 - 3^(1/2)

Y =
 3 - 3^(1/2)
 3^(1/2) + 3

>> subs(X)
ans =
    2.7321
   -0.7321

>> subs(Y)
ans =
    1.2679
    4.7321
```

Corrigé Exercice 5

Créer d'abord le fichier script correspondant :

```
function F = func(x)
F(1) = exp(-exp(-(x(1)+x(2)))) - x(2)*(1+x(1)^2);
F(2) = x(1)*cos(x(2)) + x(2)*sin(x(1)) - 0.5;
```

Puis choisir une solution de départ x_0, et appeler la fonction *fsolve*.

Corrigé Exercice 9

```
>> data = [ -9 1; 11.6 5; 21.7 10; 32.4 20; 44.1 40; 51.7 60;
61.5 100; 75.9 200; 91.4 400; 108 760 ]
>> x = data(:,1);
>> y = data(:,2);
>> p = polyfit(x,y,2)
>> yv = polyval(p,x)
>> clf % effacer la fenêtre graphique
>> plot(x,y,'r*')
>> hold on
>> plot(x,yv,'b-')
```