1. Resolva o problema *Aluffi-Pentini*,

$$\min_x f(x) \equiv 0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2,$$

considerando o valor inicial $(-1, 0.5)$,

a) usando o método quasi-Newton sem fornecer as primeiras derivadas da função objectivo.

b) usando o método quasi-Newton com procura unidimensional, fornecendo as primeiras derivadas da função objectivo.

c) usando o método de Newton com regiões de confiança, fornecendo as primeiras derivadas da função objectivo.

d) usando o método de Newton com regiões de confiança, fornecendo as primeiras e segundas derivadas da função objectivo.

Resolução:

```
function [f,g,h]=QN1(x)
f=0.25*x(1)^4-0.5*x(1)^2+0.1*x(1)+0.5*x(2)^2;
if nargout>1
    g=[x(1)^3-x(1)+0.1
        x(2)];
    if nargout>2
        h=[3*x(1)^2-1 0
            0 1];
    end
end
```

a)
```
>> x1=[-1 0.5];
>> [x,f,e,o]=fminunc('QN1',x1)
x =
   -1.0467   -0.0000
f =
   -0.3524
```

```
e =

     1

o =

       iterations: 5
        funcCount: 21
         stepsize: 1
    firstorderopt: 6.2997e-07
        algorithm: 'medium-scale: Quasi-Newton line search'
```

b)
```
>> op=optimset('GradObj','on','LargeScale','off');
>> x1=[-1 0.5];
>> [x,f,e,o]=fminunc('QN1',x1,op)
x =

   -1.0467   -0.0000

f =

   -0.3524

e =

     1

o =

       iterations: 5
        funcCount: 7
         stepsize: 1
    firstorderopt: 6.3007e-07
        algorithm: 'medium-scale: Quasi-Newton line search'
```

c)
```
>> op=optimset('GradObj','on');
>> x1=[-1 0.5];
>> [x,f,e,o]=fminunc('QN1',x1,op)
x =

   -1.0467         0

f =

   -0.3524

e =
```

```
     1

o =

         iterations: 3
          funcCount: 4
       cgiterations: 3
      firstorderopt: 7.0822e-10
          algorithm: 'large-scale: trust-region Newton'
```

d)
```
>> op=optimset('GradObj','on','Hessian','on');
>> x1=[-1 0.5];
>> [x,f,e,o]=fminunc('QN1',x1,op)

x =

   -1.0467         0

f =

   -0.3524

e =

     1

o =

         iterations: 3
          funcCount: 4
       cgiterations: 3
      firstorderopt: 7.0895e-10
          algorithm: 'large-scale: trust-region Newton'
```

2. No planeamento da produção de dois produtos, uma determinada companhia espera obter lucros iguais a P:

$$P(x_1, x_2) = \alpha_1(1 - e^{-\beta_1 x_1}) + \alpha_2(1 - e^{-\beta_2 x_2}) + \alpha_3(1 - e^{-\beta_3 x_1 x_2}) - x_1 - x_2,$$

em que x_1 é a quantia gasta para produzir e promover o produto 1, x_2 é a quantia gasta para produzir e promover o produto 2 e os α_i e β_i são constantes definidas. P, x_1 e x_2 estão em unidades de 10^5 euros. Calcule o lucro máximo para as seguintes condições:

$$\alpha_1 = 3,\ \alpha_2 = 4,\ \alpha_3 = 1,\ \beta_1 = 1.2,\ \beta_2 = 1.5,\ \text{ e } \beta_3 = 1.$$

(a) Resolva o problema usando o método quasi-Newton sem fornecer as primeiras derivadas da função objectivo. Considere a aproximação inicial $(1, 1)$.

(b) Resolva o problema usando o método quasi-Newton com procura unidimensional, fornecendo as primeiras derivadas da função objectivo. Considere a aproximação inicial da alínea anterior.

(c) Resolva novamente o problema mas seleccione agora o método de Newton com regiões de confiança.

Resolução:

```
function [f,g]=QN2(x)
a=[3 4 1];
b=[1.2 1.5 1];
P=a(1)*(1-exp(-b(1)*x(1)))+a(2)*(1-exp(-b(2)*x(2)))+a(3)*(1-exp(-b(3)*x(1)*x(2)))-x(1)-
f=-P;
if nargout>1
    g=-[a(1)*b(1)*exp(-b(1)*x(1))+a(3)*b(3)*x(2)*exp(-b(3)*x(1)*x(2))-1
        a(2)*b(2)*exp(-b(2)*x(2))+a(3)*b(3)*x(1)*exp(-b(3)*x(1)*x(2))-1];
end
```

a)
```
>> x1=[1 1];
>> [x,f,e,o]=fminunc('QN2',x1)
x =
```

```
     1.2905    1.3623

f =

   -4.0189

e =

     1

o =

       iterations: 6
        funcCount: 21
         stepsize: 1
    firstorderopt: 9.6993e-07
        algorithm: 'medium-scale: Quasi-Newton line search'
```

b)
```
>> op=optimset('GradObj','on','LargeScale','off');
>> x1=[1 1];
>> [x,f,e,o]=fminunc('QN2',x1,op)
x =

     1.2905    1.3623

f =

   -4.0189

e =

     1

o =

       iterations: 6
        funcCount: 7
         stepsize: 1
    firstorderopt: 9.8930e-07
        algorithm: 'medium-scale: Quasi-Newton line search'
```

c)
```
>> op=optimset('GradObj','on');
>> x1=[1 1];
>> [x,f,e,o]=fminunc('QN2',x1,op)
x =

     1.2905    1.3623
```

```
f =

   -4.0189

e =

     1

o =

          iterations: 4
           funcCount: 5
        cgiterations: 4
       firstorderopt: 4.2341e-11
           algorithm: 'large-scale: trust-region Newton'
```

O lucro máximo é 4.0189×10^5 €.

4. Resolva o problema *Epistatic Michalewicz*

$$\min_x f(x) \equiv -\sum_{i=1}^{n} \sin(y_i) \left(\sin \left(\frac{i y_i^2}{\pi} \right) \right)^{2m}$$

$$y_i = \begin{cases} x_i \cos(\theta) - x_{i+1} \sin(\theta), & i = 1, 3, 5, \ldots, < n \\ x_i \sin(\theta) + x_{i+1} \cos(\theta), & i = 2, 4, 6, \ldots, < n \\ x_i & i = n \end{cases}$$

pelo método quasi-Newton (sem fornecer derivadas) para $n = 5$ e para $n = 10$. Considere $\theta = \frac{\pi}{6}$, $m = 10$ e o valor inicial $x^{(1)} = \begin{cases} 2, & i = 1, 3, 5, \ldots, \leq n \\ 1, & i = 2, 4, 6, \ldots, \leq n \end{cases}$.

Resolução:

```
function f=QN4(x,t,m)

n=length(x);

i=1:2:n-1;

y(i)=x(i)*cos(t)-x(i+1)*sin(t);

i=2:2:n-1;

y(i)=x(i)*sin(t)+x(i+1)*cos(t);

i=n;

y(i)=x(i);

i=1:n;

f=-sum(sin(y).*(sin(i.*y.^2/pi)).^(2*m));


>> n=5;

>> i=1:2:n;

>> x1(i)=2;

>> i=2:2:n;

>> x1(i)=1;

>> t=pi/6;

>> m=10;

>> [x,f,e,o]=fminunc('QN4',x1,[],t,m)

x =
```

```
    2.0000    1.0000    2.0458    0.9735    2.0000

f =

   -0.9591

e =

     2

o =

       iterations: 4
        funcCount: 42
         stepsize: 1
    firstorderopt: 5.4240e-06
        algorithm: 'medium-scale: Quasi-Newton line search'

>> n=10;
>> i=1:2:n;
>> x1(i)=2;
>> i=2:2:n;
>> x1(i)=1;
>> op=optimset('MaxFunEvals',2000);
>> [x,f,e,o]=fminunc('QN4',x1,op)
x =
1.9992 1.5170 1.4994 1.2041 1.9426 0.7170 2.0909 1.0861 1.6472 0.9215
f =

   3.7634e-07

e =

     1

o =

       iterations: 111
        funcCount: 1265
         stepsize: 1
    firstorderopt: 8.7931e-06
        algorithm: 'medium-scale: Quasi-Newton line search'
```

3. Considere o seguinte problema não diferenciável

$$\min_{x \in \mathbb{R}^n} f(x) \equiv n \left(\max_{1 \leq i \leq n} x_i \right) - \sum_{i=1}^{n} x_i.$$

Para $n = 2$ e a partir da aproximação inicial $x_i = i - (\frac{n}{2} + 0.5)$, $i = 1, \ldots, n$, calcule a solução.

Repita a resolução considerando agora $n = 5$ e TolX= 10^{-20}. Resolva ainda acrescentando a opção MaxFunEvals=10000. Acrescente ainda a opção MaxIter=10000. Comente os resultados.

Resolução:

```
function f=NM3(x)
n=length(x);
f=n*max(x)-sum(x);

>> n=2;
>> i=1:n;
>> x1(i)=i-(n/2+0.5);
>> [x,f,e,o]=fminsearch('NM3',x1)
x =
   -0.7164   -0.7164
f =
   6.4289e-12
e =
     1
o =
    iterations: 66
     funcCount: 119
     algorithm: 'Nelder-Mead simplex direct search'

>> n=5;
>> i=1:n;
```

```
>> x1(i)=i-(n/2+0.5);
>> op=optimset('TolX',1e-20);
>> [x,f,e,o]=fminsearch('NM3',x1,op)
Exiting: Maximum number of function evaluations has been exceeded
         - increase MaxFunEvals option.
         Current function value: 3.853342
x =
   -3.8423    0.0069    0.0083    0.0088    0.0087
f =
    3.8533
e =
     0
o =
    iterations: 586
     funcCount: 1000
     algorithm: 'Nelder-Mead simplex direct search'

>> op=optimset('TolX',1e-20,'MaxFunEvals',10000);
>> [x,f,e,o]=fminsearch('NM3',x1,op)
Exiting: Maximum number of iterations has been exceeded
         - increase MaxIter option.
         Current function value: 0.338017
x =
    0.1272    0.0083    0.0184    0.0170    0.1272
f =
    0.3380
e =
     0
o =
    iterations: 1000
     funcCount: 1730
     algorithm: 'Nelder-Mead simplex direct search'
```

```
>> op=optimset('TolX',1e-20,'MaxFunEvals',10000,'MaxIter',10000);
>> [x,f,e,o]=fminsearch('NM3',x1,op)
x =
    0.1272    0.0083    0.0184    0.0170    0.1272
f =
    0.3380
e =
     1
o =
     iterations: 1175
     funcCount: 2046
     algorithm: 'Nelder-Mead simplex direct search'
```

O número máximo de cálculos de função e de iterações que estão por defeito no MATLAB não são suficientes para este problema convergir (`exitflag=0`), sendo, por isso, necessário aumentá-los usando as opções `MaxFunEvals` e `MaxIter`, tal como os avisos do MATLAB sugerem.

1. Resolva o problema em MATLAB

$$\begin{array}{ll} \text{minimizar} & 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3 \\ \text{sujeito a} & 8x_1 + 14x_2 + 7x_3 - 56 = 0 \\ & x_1^2 + x_2^2 + x_3^2 - 25 = 0 \\ & 0 \leq x_i, i = 1, 2, 3 \end{array}$$

com $x^1 = (2, 2, 2)^T$.

Resolução:

```
function f=SQP1(x)
f=1000-x(1)^2-2*x(2)^2-x(3)^2-x(1)*x(2)-x(1)*x(3);

function [c,ceq]=SQP1_r(x)
c=[];
ceq=x(1)^2+x(2)^2+x(3)^2-25;

>> x0=[2 2 2];
>> A=[];
>> b=[];
>> Aeq=[8 14 7];
>> beq=56;
>> lb=[0 0 0];
>> ub=[];
>> [x,f,exitf,outp]=fmincon('SQP1',x0,A,b,Aeq,beq,lb,ub,'SQP1_r')
x =
    3.5121    0.2170    3.5522
f =
  961.7152
exitf =
     4
outp =
         iterations: 8
```

```
      funcCount: 33
   lssteplength: 1
       stepsize: 1.0889e-06
      algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
  firstorderopt: 1.2042e-06
constrviolation: 5.5195e-08
```

2. Resolva o seguinte problema em MATLAB

$$\begin{aligned} \text{minimizar} \quad & (x_1 + 3x_2 + x_3)^2 + 4\,(x_1 - x_2)^2 \\ \text{sujeito a} \quad & 6x_2 + 4x_3 - x_1^3 - 3 \geq 0 \\ & 1 - x_1 - x_2 - x_3 = 0 \\ & 0 \leq x_i, i = 1, 2, 3 \end{aligned}$$

com $x^1 = (0.1, 0.7, 0.2)^T$.

Resolução:

```
function f=SQP2(x)
f=(x(1)+3*x(2)+x(3))^2+4*(x(1)-x(2))^2;

function [c,ceq]=SQP2_r(x)
c=-6*x(2)-4*x(3)+x(1)^3+3;
ceq=[];

>> x0=[0.1 0.7 0.2];
>> A=[];
>> b=[];
>> Aeq=[1 1 1];
>> beq=1;
>> lb=[0 0 0];
>> ub=[];
>> [x,f,exitf,outp]=fmincon('SQP2',x0,A,b,Aeq,beq,lb,ub,'SQP2_r')
x =
    0.0000   -0.0000    1.0000
f =
    1.0000
exitf =
     1
outp =
         iterations: 3
```

```
       funcCount: 12
    lssteplength: 1
        stepsize: 1.2397e-16
       algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'
   firstorderopt: 7.8200e-17
 constrviolation: 1.1102e-16
```