Exercício 2 (fsolve)

m-file (sem derivadas):

```
function [f] = poluente( x )
f(1)=70*exp(x(1))+20*exp(x(2))-27.5702;
f(2)=70*exp(2*x(1))+20*exp(2*x(2))-17.6567;
end
```

Na janela de comandos:

```
>> x0=[-1,-0.1];
>> options=optimset('TolX',1.0e-10,'Tolfun',1.0e-8,'Display','iter');
>> [xsol,fsol,exitflag,output]=fsolve('poluente',x0,options)
Iteration   Func-count     f(x)          …
    0          3             332.
    1          6             113.51
    2          9             1.96736
    3         12                         0.00241358
    4         15                       8.68711e-009
    5         18             7.46056e-020
    6         21             0

xsol =  -2.000001586403948  -0.100000174536568           % solução do problema
fsol =     0     0
exitflag =     1
output =
      iterations: 6
       funcCount: 21
```

----

m-file (com derivada):

```
function [ f,j] = poluente( x )
f(1)=70*exp(x(1))+20*exp(x(2))-27.5702;
f(2)=70*exp(2*x(1))+20*exp(2*x(2))-17.6567;
if nargout>1
j=[70*exp(x(1)) 20*exp(x(2));140*exp(2*x(1)) 40*exp(2*x(2))];
end
end
```

Na janela de comandos:

```
>> x0=[-1,-0.1];
>> options=optimset('TolX',1.0e-10,'Tolfun',1.0e-8,'Jacobian','on');
>> [xsol,fsol,exitflag,output]=fsolve('poluente',x0,options)
xsol =  -2.000001586403947  -0.100000174536568

fsol =  1.0e-014 *
  0.355271367880050                  0
exitflag =     1
output =
      iterations: 6
       funcCount: 7
```

Comandos a experimentar:

```
>>help fsolve
```

Ver o doc fsolve (que aparece no comando anterior): ver as hipóteses da exitflag

```
>>optimset
```

Reparar nas seguintes opções que podem ser alteradas; experimentar alterar:

```
  Display: [ off | iter | iter-detailed | notify | notify-detailed | final | final-detailed ]
  MaxFunEvals: [ positive scalar ]
  MaxIter: [ positive scalar ]
   TolFun: [ positive scalar ]
  TolX: [ positive scalar ]
  Jacobian: [ on | {off} ]
```

```
>>optimset fsolve
```

O comando anterior mostra os valores de optimset para a função fsolve