

Métodos Numéricos e Otimização não Linear

Miniprojeto 1 – Grupo 9 – Versão A

Alexandra Candeias (a89521)
Meriem Khammassi (a85829)
Patrícia Pereira (a89578)
Simão Brito (a89482)

1. Introdução

No âmbito da UC de Métodos Numéricos e Otimização não Linear, foi-nos proposta a realização de um mini projeto que consiste em utilizar a função `fsolve` do MATLAB de modo a resolver um problema de equações ou sistema de equações aplicado de uma determinada área.

2. Descrição do problema:

O problema por nós selecionado e que se aplica à Engenharia Industrial foi:

“Numa start-up de montagem de computadores foi pedido para determinar o número mínimo de computadores que a loja tem de vender para conseguir obter lucro.

A equação que nos dá esse valor (n), depois de considerar o total de custo e o total de vendas é:

$$F(n) = 40n^{1.5} - 875n + 35000 = 0$$

Pretendemos calcular as raízes da equação de modo a encontrar o número mínimo de computadores que são necessários vender para atingir o lucro.

(Fonte: https://nm.mathforcollege.com/strippedfiles/mws/ind/03nle/mws_ind_nle_txt_newton_examples.pdf)

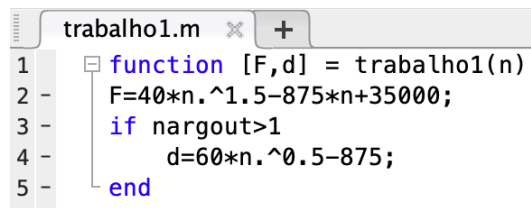
3. Resolução do problema:

Utilizamos para aproximação inicial o valor $x_0 = 50.0$ e os valores de $\varepsilon_1 = \varepsilon_2 = 1.0 \times 10^{-3}$.

Pretende-se resolver $F(n) = 0$.

O primeiro passo é calcular a derivada de f : $f'(n) = d = 60n^{0.5} - 875$

Ficheiro m:



```

1 function [F,d] = trabalho1(n)
2 F=40*n.^1.5-875*n+35000;
3 if nargin>1
4     d=60*n.^0.5-875;
5 end

```

Figura 1-Função a analisar

3.1. $x_0 = 50.0$

Opções iniciais:

```
>> x0=[50]
```

```
>> options=optimset('Jacobian','on','TolX',1.0e-3,'TolFun',1.0e-3);
```

Comando usado:

```
>> [nsol,fsol,exitflag,output]=fsolve('trabalho1',x0,options)
```

Resultado:

```

nsol =
    62.6917

fsol =
    1.1642e-10

exitflag =
     1

output =
    struct with fields:
        iterations: 6

```

Apesar de obtermos uma solução possível e uma *exitflag* positiva, resolvemos testar o mesmo programa e mesmos comandos com valores iniciais maiores elevados, uma vez que a otimização é pequena.

3.2. $x_0 = 60.0$

```

nsol =
    62.6917

fsol =
     0

exitflag =
    1

output =
    struct with fields:
        iterations: 4

```

Testamos também para os valores iniciais de 60, 63, 70 e ainda valores muito dispersos destes obtendo sempre resultados semelhantes, variando na sua maioria apenas o f , ou seja o lucro.

Resolvemos assim começar a variar os valores da TolX e TolFun, para o valor inicial 50.

3.3. $\varepsilon_1 = \varepsilon_2 = 5.0 \times 10^{-3}$

Opções:

```
>> x0=[50]

>> options=optimset('Jacobian','on','TolX',5e-3,'TolFun',5e-3);

nsol =

    62.6917

fsol =

    1.1642e-10

exitflag =

     1

output =

    struct with fields:
        iterations: 6
```

Continuamos a obter uma *exitflag* de 1, para $\varepsilon_1 = \varepsilon_2 = 5.0 \times 10^{-3}$ e para $\varepsilon_1 = \varepsilon_2 = 1.0 \times 10^{-2}$

3.4. $\varepsilon_1 = \varepsilon_2 = 5.0 \times 10^{-2}$

```
>> options=optimset('Jacobian','on','TolX',5e-2,'TolFun',5e-2);

nsol =

    62.6917

fsol =

    0.0031

exitflag =

     2

output =

    struct with fields:
        iterations: 5
```

A otimização termina com sucesso, sendo a *exitflag* 2, ou seja, a mudança em *nsol* é menor que a *OPTIONS.TolFun*. Resolvemos ainda testar, com estes valores de ε , outros valores iniciais, porém não obtemos melhor solução.

4. Conclusão

A solução ótima e adequada ao nosso problema será $x_0=50$ obtendo um valor de $n=62.6917$. Pelo que para obtermos lucro a loja terá de vender no mínimo 63 computadores.

Na realização deste mini-projeto deparamo-nos com certas dificuldades. A maior foi interpretar os dados que obtemos uma vez que alterando os valores iniciais os resultados não variavam muito. Isto obrigou-nos a ter de pensar noutras formas de conseguirmos obter uma melhor otimização, acabando por olhar os valores dos parâmetros dentro do *optimset*.

Deste modo pensamos ter ultrapassado o problema, e consideramos ter realizado o propósito deste projeto.