Ano Letivo 2020/2021

Mini-projeto Matlab 2

Grupo 19


a)

b)

c)

d)

a) Ana Catarina Canelas (a93872) b) Ana Filipa Pereira (a89589)
c) Carolina Santejo (a89500) d) Raquel Costa (a89464)

Introdução

Neste projeto, o grupo escolheu um problema de otimização referente ao lucro obtido através da venda de barris de aço no Canadá e nos Estados Unidos da América. Estes barris possuem custos de produção e preços de mercado (lucro) que diminuem com o aumento das vendas.

Esta escolha, teve em conta que queríamos tratar de um problema de otimização não linear e que não contivesse restrições. Em relação à fonte, o grupo adaptou aos requisitos deste projeto, as fórmulas de um problema presente no enunciado de um teste da cadeira MA 132 lecionada na Universidade estadual da Carolina do Norte.

Fonte: https://ma132.wordpress.ncsu.edu/lesson-4-maximizing-profit-in-two-markets/

Formulação Matemática

Sendo x as unidades vendidas no Canadá e y as vendidas nos EUA, e o lucro a diferença entre o proveito e o custo, temos o seguinte:

$$custo(x,y) = 10000 + 3.5 \times (x+y)$$

$$\text{proveito}(x,y) = (\alpha - 0.07x - 0.02y)x + (\beta - 0.01x - 0.1y)y$$

$$lucro(x,y) = (\alpha - 0.07x - 0.02y)x + (\beta - 0.01x - 0.1y)y - (10000 + 3.5 \times (x+y))$$

- Na função custo, 10000 é um custo fixo e 3.5 é o custo de produção de cada barril.
- Na função lucro, os valores decimais são as reduções que os preços de mercado sofrem com a venda de cada barril.
- Alfa e beta são parâmetros que representam o preço dos barris no mercado do Canadá e dos EUA, respetivamente.

MatLab

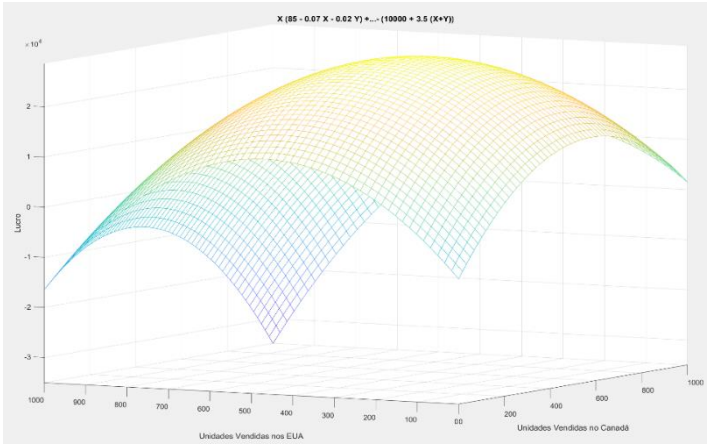

Figura 1- Gráfico do lucro

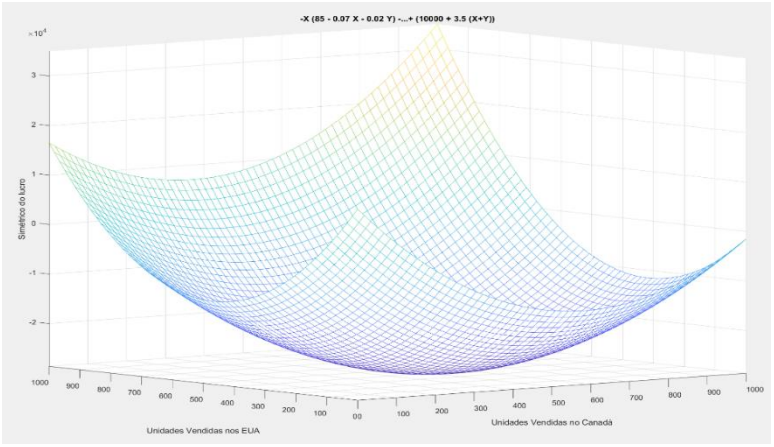

Figura 2- Gráfico da função simétrica do lucro

Comando utilizado para o desenho da função do lucro e do seu simétrico, respetivamente (α=85 e β=97):

```
ezmesh('X*(85 - 0.07*X - 0.02*Y) + Y*(97 - 0.01*X - 0.1*Y) - (10000 + 3.5*(X+Y))',[0,1000],[0,1000])
```

```
ezmesh('-X*(85 - 0.07*X - 0.02*Y) - Y*(97 - 0.01*X - 0.1*Y) + (10000 + 3.5*(X+Y))',[0,1000],[0,1000])
```

Visto que a função escolhida é diferenciável é possível utilizar tanto a rotina *fminunc* como a *fminsearch*. No entanto, estas rotinas são usadas para calcular o mínimo de uma função, o que no nosso problema não se aplica, pois, o objetivo é determinar o lucro máximo possível de se obter. Assim, teremos de determinar o mínimo global da função simétrica, de modo a descobrir o máximo da função.

Testes Computacionais - *fminunc*

Pela análise do gráfico da figura 2 podemos ver, pela forma da função, que existe um mínimo global (será um ponto no "vale" do gráfico). Usemos, os seguintes pontos iniciais: P1 (300,400), P2 (600,500) e P3 (-100,300). Iremos começar por usar a rotina ***fminunc*** que implementa o algoritmo *Quasi-Newton* (que tem convergência superlinear devido ao uso de derivadas no seu cálculo).

```
xmin =

  497.9705  392.8044


fmin =

  -2.8656e+04
```

Figura 3-Resultado Fminunc

Para qualquer um dos três pontos iniciais, o minimizante e o mínimo calculado, pela rotina são os obtidos na figura 3, respetivamente.

Após a **análise dos resultados fornecidos**, concluímos que, o valor do mínimo global da função simétrica do lucro é -228656 que corresponde ao minimizante x = 497.9705 e y = 392.8044. No contexto da função original que queríamos maximizar, o valor de x e y mantêm o seu valor, mas o lucro máximo é o simétrico do valor obtido, logo é igual a 228656 unidades monetárias. Este lucro obtém-se com a venda de 497.9705 barris de aço para o Canadá e 392.8044 para os EUA. Visto que não é possível dividir um barril em porções mais pequenas, uma alternativa seria que x fossem 498 unidades e y fossem 393 unidades.

Iremos agora **alterar os valores dos parâmetros α e β** de forma a verificar o comportamento da função lucro com a variação dos preços de mercado dos barris.

```
xmin =

  373.2473  541.5130


fmin =

  -3.5139e+04
```

Figura 4- Resultado com α = 72 e β = 123

```
xmin =

  418.6347  329.7048


fmin =

  -1.7279e+04
```

Figura 5 -Resultado com α = 72 e β = 82

- Pela figura 4 conclui-se que diminuir o preço de mercado dos barris no Canadá e aumentar nos EUA, leva a um aumento do lucro, sendo este de 35139 unidades monetárias.
- Pela figura 5 conclui-se que diminuir o preço de mercado dos barris no Canadá e anos EUA, leva a uma diminuição do lucro, sendo este de 17279 unidades monetárias.

Script Utilizado:

```
%Parâmetros
c = [85,97];


%Desenho
%x = ezmesh('-X*(85 - 0.07*X - 0.02*Y) - Y*(97 - 0.01*X - 0.1*Y) + (10000 + 3.5*(X+Y))',[0,1000],[0,1000])


%Aplicação da Rotina
[xmin,fmin,options,exitflag,output]=fminunc('tp2funcl',[300,400],[],c);
```

Function Utilizada:

```
function [f] = tp2funcl(x,c)
P = x(1)*(c(1) - 0.07*x(1) - 0.02*x(2)) + x(2)*(c(2) - 0.01*x(1) - 0.1*x(2)) - (10000 + 3.5*(x(1)+x(2)));
f = -P;
end
```

Testes Computacionais - *fminsearch*

Iremos agora abordar os resultados obtidos com a rotina ***fminsearch*** do Matlab, esta utiliza o algoritmo de *Nelder-Mead*, que tem baixa velocidade de convergência, ao contrário do *Quasi-Newton*. Além disso, iremos também alterar um dos **parâmetros de otimização** de modo a obter um output mais detalhado da rotina executada. Primeiramente, usamos o mesmo ponto inicial P1(300,400) que usamos na rotina *fminunc*, e tal como podemos observar na Figura 4, foram obtidos os mesmos valores tanto para o minimizante (*xmin*) como para o mínimo (*fmin*). Também é possível observar que a *exitflag* devolveu o valor 1, isto quer dizer que convergiu dentro dos critérios por defeito, e fez 51 iterações e calculou a função 98 vezes, usando o algoritmo de *Nelder-Mead*. Caso o valor obtido na exitflag fosse 0 então significaria que tinha saído do processo iterativo, isto é, não verificou as condições do Critério de Paragem nas iterações realizadas.

```
xmin =

  497.9704  392.8045


fmin =

  -2.8656e+04


exitflag =

     1


output =

  struct with fields:

    iterations: 51
     funcCount: 98
     algorithm: 'Nelder-Mead simplex direct search'
       message: 'Optimization terminated:↵ the current x satisfies
```

Figura 4 - Output fminsearch

Iteration	Func-count	min f(x)	Procedure
0	1	-25950	
1	3	-26346.8	initial simplex
2	5	-26558.1	expand
3	7	-27287	expand
4	9	-27651.6	expand
5	11	-28521.7	expand
6	12	-28521.7	reflect
7	14	-28521.7	contract outside
8	16	-28643.1	reflect
9	18	-28643.1	contract outside
10	19	-28643.1	reflect
11	21	-28643.1	contract inside
12	23	-28646.9	contract inside
13	25	-28652.9	contract inside
14	27	-28655.3	contract inside
15	29	-28655.3	contract inside
16	30	-28655.3	reflect
17	32	-28655.5	contract inside
18	34	-28655.8	contract outside
19	36	-28655.8	contract inside
20	38	-28655.8	contract inside
21	40	-28655.9	contract inside
22	42	-28655.9	contract inside
23	44	-28655.9	contract inside
24	46	-28655.9	contract inside
25	48	-28655.9	contract inside
26	50	-28655.9	contract inside
27	52	-28655.9	contract outside
28	54	-28655.9	contract inside
29	56	-28655.9	contract inside
30	58	-28655.9	contract outside
31	60	-28655.9	contract inside
32	62	-28655.9	contract outside
33	64	-28655.9	contract inside
34	66	-28655.9	contract inside
35	68	-28655.9	contract inside
41	79	-28655.9	contract inside
42	81	-28655.9	contract inside
43	83	-28655.9	contract inside
44	85	-28655.9	contract inside
45	87	-28655.9	contract inside
46	89	-28655.9	contract inside
47	90	-28655.9	reflect
48	92	-28655.9	contract inside
49	94	-28655.9	contract inside
50	96	-28655.9	contract outside
51	98	-28655.9	contract inside

```
Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-04
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-04
```

Figura 6 - Output MatLab

De modo a obter o processo detalhado em cada iteração realizada. O grupo utilizou o comando o "optimset", tal como é possível observar na Figura 5. Deste modo, tal como sabemos em cada iteração são definidos pontos auxiliares que são candidatos a vértices de um novo *simplex*, e estes poderão ser aceites ou rejeitados. Os pontos são construídos com base em várias operações: refletir, expandir e contrair (para o interior/exterior). Observando o output devolvido com a alteração das opções/parâmetros de otimização, conseguimos identificar qual o tipo de ação que o algoritmo tomou em cada iteração.

Script utilizado:

```
%Parâmetros
c = [85,97];

options=optimset('display','iter');

%Aplicação da Rotina
[xmin,fmin,exitflag,output]=fminsearch('tp2func1',[300,400],options,c)
```

Figura 5 - Script fminsearch

Conclusão

Este trabalho permitiu não só consolidar o conhecimento adquirido na cadeira relativo a otimização não linear, bem como a familiarização com as rotinas *fminunc* e *fminsearch* do *Matlab*. Além disto, a pesquisa por um problema que possuísse um contexto real, demonstrou a importância da otimização, no nosso dia-a-dia e nas mais diversas áreas.