Modelos Numéricos e Otimização não Linear

Mini-Projeto Matlab 2

Grupo 20

- Celso André Carvalho Rodrigues – A83655
- Luís Manuel Pereira - A77667
- Ricardo Miguel Santos Gomes - A93785
- Rúben de Castro Rodrigues - A80960

Descrição do Problema

Após uma pesquisa de um problema que se enquadrasse no objetivo do projeto, problemas de otimização não linear sem restrições, encontramos um tema bastante interessante, função Ackley. Esta é uma função usada para testes de otimização de algoritmos e vamos trabalhar com a sua versão bidimensional (n=2), que é a versão mais usual.

$$f(\mathbf{x}) = -a \exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)\right) + a + \exp(1)$$

Usamos para as variáveis os seguintes valores, a = 20, b = 0.2 e c = 2π, sendo estes os valores recomendados do problema.

Usamos o site **https://www.sfu.ca/~ssurjano/ackley.html** para tirar as informações acima descritas e para nos informarmos acerca do problema.

Objetivo e condições de aplicabilidade

Com este projeto procuramos enriquecer o nosso conhecimento à cerca das duas rotinas usadas no MATLAB, fminunc e fminsearch.

Para utilização da rotina fminunc tivemos de verificar se a função em causa é diferenciável, porque a rotina em questão apenas funciona com funções diferenciáveis. Contudo, analisando a nossa função objetivo é fácil de verificar que esta é diferenciável, podendo ser usadas ambas as rotinas.

Testes computacionais

Com o objetivo de realizar as rotinas no MATLAB, criamos a seguinte função do problema.

```
function y = ackley(x)

% Ackley function.

n = 2;
a = 20; b = 0.2; c = 2*pi;
s1 = 0; s2 = 0;
for i=1:n;
    s1 = s1+x(i)^2;
    s2 = s2+cos(c*x(i));
end
y = -a*exp(-b*sqrt(1/n*s1))-exp(1/n*s2)+a+exp(1);

end
```

Rotina fminunc

```
xmin =

     0     0


fmin =

   8.8818e-16


exitflag =

     5


output =

  struct with fields:

       iterations: 2
        funcCount: 60
         stepsize: 1.4142
     lssteplength: 0.6107
    firstorderopt: 2.8284
        algorithm: 'quasi-newton'
          message: '↵Local minimum possible.
```

Para executar a rotina fminunc, utilizamos o comando [xmin,fmin,exitflag,output]=fminunc('ackley',[-1 1],options), em que options = optimset('Display','iter','MaxFunEvals',500,'TolX',1.0000e-8,'TolFun',1.000000e-8), e com uma aproximação inicial de –1 e 1, cujos resultados estão apresentados na imagem acima.

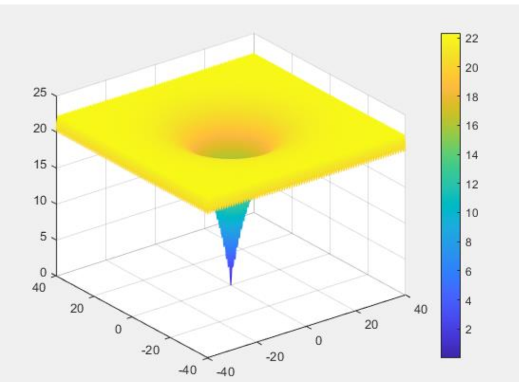
Analisando os resultados, verificamos que x1=0 e x2 =0, sendo o valor da função objetivo 8.8818e-16. Dado que exitflag=5, >0, percebemos que a função convergiu e encontrou um “Local minimum possible”, tendo sido realizadas 2 iterações e o valor da função calculado 60 vezes.

Rotina fminsearch

```
xmin =

   1.0e-08 *

   -0.1496    0.2143


fmin =

   7.3921e-09


exitflag =

     1


output =

  struct with fields:

    iterations: 48
     funcCount: 87
     algorithm: 'Nelder-Mead simplex direct search'
       message: 'Optimization terminated:↵ the curr
```

Para executar a rotina fminsearch, utilizamos o comando [xmin,fmin,exitflag,output] = = fminsearch('ackley',[-0.001 0.001],options), em que options = = optimset('Display','iter','MaxFunEvals',500,'TolX',1.0000e-8,'TolFun',1.000000e-8), e com uma aproximação inicial de -0.001 e 0.001, cujos resultados estão apresentados na imagem acima. Analisando os resultados, verificamos que x1= -0.1496e-08 e x2 = 0.2143e-08, sendo o valor da função objetivo 7.3921e-09. Dado que exitflag=1, >0, percebemos que a função convergiu e terminou, “Optimization terminated”, tendo sido realizadas 48 iterações e o valor da função calculado 87 vezes.

Conclusão



Através dos resultados obtidos, podemos concluir que a nossa função objetivo converge para um mínimo global, cujo ponto é (0,0), embora tenha bastantes mínimos locais. Para confirmar os resultados, criamos também um gráfico para fácil verificação do mesmo. Finalizando o trabalho consideramos que o maior desafio foi encontrar um problema que enquadrasse o projeto e ao mesmo tempo desafiasse o nosso conhecimento.