Trabalho I
Otimização - 2015.2
Universidade Federal do Rio de Janeiro

Felipe Matheus Fernandes Oliveira
felipemfo@poli.ufrj.br

Henrique Mattos
henriquemattos@poli.ufrj.br

Jean Américo Tomé
jamerico@poli.ufrj.br

Philipe Miranda de Moura
philipemoura@poli.ufrj.br

13 de janeiro de 2016

Conteúdo

1 Introdução **3**

2 Método de Fibonacci **3**

3 Método da Divisão Áurea **3**

4 Método da Interpolação polinomial **4**

5 Interface Gráfica (GUI) **5**

A Apêndice **7**
A.1 Método de Fibonacci 7
A.2 Método da Divisão Áurea 8
A.3 Método da Interpolação Polinomial 9

1 Introdução

Problemas de otimização, em geral, buscam encontrar o mínimo de uma função em um dado intervalo.

A maneira mais intuitiva de encontrar o mínimo é provinda do cálculo diferencial e consiste em derivar a função e igualar a zero. No entanto, essa forma pode não ser amigável, uma vez que pode ser muito trabalhoso encontrar a derivada. Além disso, igualar a derivada a zero pode não ter solução analítica. Por fim, esse método ainda tem a restrição de não trabalhar com funções não contínuas.

Essas características motivaram a obtenção de métodos outros métodos para obtenção de mínimos. Nesse trabalho, abordaremos o método de Fibonacci, o da Divisão Áurea e o da Interpolação Polinomial.

Para que esses métodos funcionem, convergindo para o mínimo global do intervalo, precisamos garantir que a função é unimodal, ou seja, possui apenas um ponto de mínimo no intervalo considerado. A função também precisa ser suave (de classe C^n com $n \geq 2$) no intervalo. Mesmo com essas características, ainda é possível que o método da Interpolação Polinomial não funcione, como será melhor visto na seção correspondente.

2 Método de Fibonacci

Para um intervalo inicial $I_k = \begin{bmatrix} a & b \end{bmatrix}$, em que a função f é unimodal, escolhemos dois pontos internos simétricos com relação ao centro. A posição exata desses pontos internos é determinada por um fator α_1, proveniente da sequência de Fibonacci.

$$\alpha_1 = \frac{f(k-1)}{f(k)} = \frac{2}{1+\sqrt{5}} \times \frac{1-p^k}{1-p^{k+1}} \qquad onde \qquad p = \frac{1-\sqrt{5}}{1+\sqrt{5}} \tag{1}$$

Se n for o número desejado de reduções de intervalo, associa-se a ele o índice de Fibonacci $k = n+1$, como visto na equação acima.

Após determinados os pontos internos, verificamos em qual intervalo de pontos o mínimo se encontra, atualizamos os novos extremos do intervalo e repetimos o método até atingir o número n, que é determinado pela tolerância desejada.

3 Método da Divisão Áurea

O método consiste em achar uma proporção tal que, dividindo o total ($\overline{AB}$) em 2 segmentos de tamanhos diferentes, a razão entre o total e o maior segmento (x) é igual a razão entre o maior e o menor segmento (d).

Através dessa razão estabelecemos as divisões dos intervalos das funções aos quais se deseja otimizar, adotamos o intervalo onde se encontra o ponto ótimo como novo segmento total e repetimos esse procedimento até satisfazer o erro desejado.

$$\frac{d}{x} = \frac{x}{d-x}$$

Da igualdade temos que a solução que interessa é:

$$x = \frac{d(\sqrt{5}-1)}{2} \approx 0,6180\, d$$

Há ainda nesse método uma grande relação com o método de Fibonacci , pois a razão entre dois números consecutivos de Fibonacci depende diretamente do fator k; e como podemos ver a seguir para números muito grandes esse fator tende a razão áurea.

$$lim_{k\to\infty} \frac{f(k-1)}{f(k)} = \frac{\sqrt{5}-1}{2} = 0.618 = u$$

4 Método da Interpolação polinomial

Em intervalos pequenos, funções suaves podem ser aproximadas por funções polinomiais. Estas são chamadas de interpolações polinomiais das funções originais e podem ser de qualquer grau. Entretanto, quando, em um pequeno intervalo, existe um mínimo da função, esta se aproxima muito de uma parábola e, portanto, se torna mais adequado e desejável, para descobrir o valor deste mínimo, acharmos um polinômio quadrático que aproxime a função original. Este processo se dá, primeiramente, pela escolha de 3 pontos p, q e m, internos ao intervalo. Agora, queremos determinar uma parábola que passe pelos 3. Desta forma, conseguimos desenvolver o seguinte sistema de equações:

$$\begin{bmatrix} p^2 & p & 1 \\ q^2 & q & 1 \\ m^2 & m & 1 \end{bmatrix} \times \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} f_p \\ f_q \\ f_m \end{bmatrix} \tag{2}$$

Onde f_p, f_q e f_m são os valores da função para os respectivos pontos. A resolução do sistema nos dá os valores dos parâmetros da parábola. Agora, vamos usar este polinômio para encontrarmos o mínimo da função e conseguirmos otimizá-la. O primeiro passo é descobrirmos o mínimo, u, da parábola encontrada, $u = -\beta/(2\alpha)$.

Desta maneira, ficamos com 4 pontos do intervalo (p, q, m e u). O ponto u pode ser uma aproximação boa para o mínimo da função ou não. Para determinarmos isto, colocamos uma tolerância para a distância entre o mínimo encontrado e os outros pontos. Enquanto as distâncias forem maiores que um determinado valor, decidido pelo usuário, o método continua.

Vamos considerar (e isto é o que foi feito no método implementado neste trabalho) que, na escolha dos 3 primeiros pontos, p e m são os extremos do intervalo (no qual supõe-se conter um mínimo) e q é o ponto médio deste intervalo. Assim, continuando com o método (caso a tolerância citada não seja atendida), eliminamos o ponto extremo que tiver maior valor de função e ficamos, novamente com um intervalo e 3 pontos. Repetimos o processo de interpolação até a tolerância ser satisfeita.

Definido o funcionamento do método, vamos expor alguns problemas e limitações deste.

O primeiro problema ocorre caso o ponto de mínimo da parábola interpolada esteja fora do intervalo considerado. Neste caso, o método analisa qual seção do intervalo é maior (considerando os extremos e o ponto interno), divide esta seção pela razão áurea e pega o ponto encontrado mais perto do extremo considerado como o novo "mínimo". Caso as seções sejam iguais, ele utiliza a da esquerda.

O segundo problema ocorre caso, escolhido o primeiro intervalo, os pontos dos extremos e o ponto médio têm o mesmo valor de função (Exemplo: função seno de -π a π). Neste caso, o método não consegue resolver o primeiro sistema de equações, pois este será indeterminado.

Um terceiro problema pode ocorrer caso a função tenha concavidade positiva em qualquer ponto do intervalo considerado, pois, nesse caso, o ponto de "mínimo"encontrado pode ser, na verdade, um máximo, o que confundirá o método.

5 Interface Gráfica (GUI)

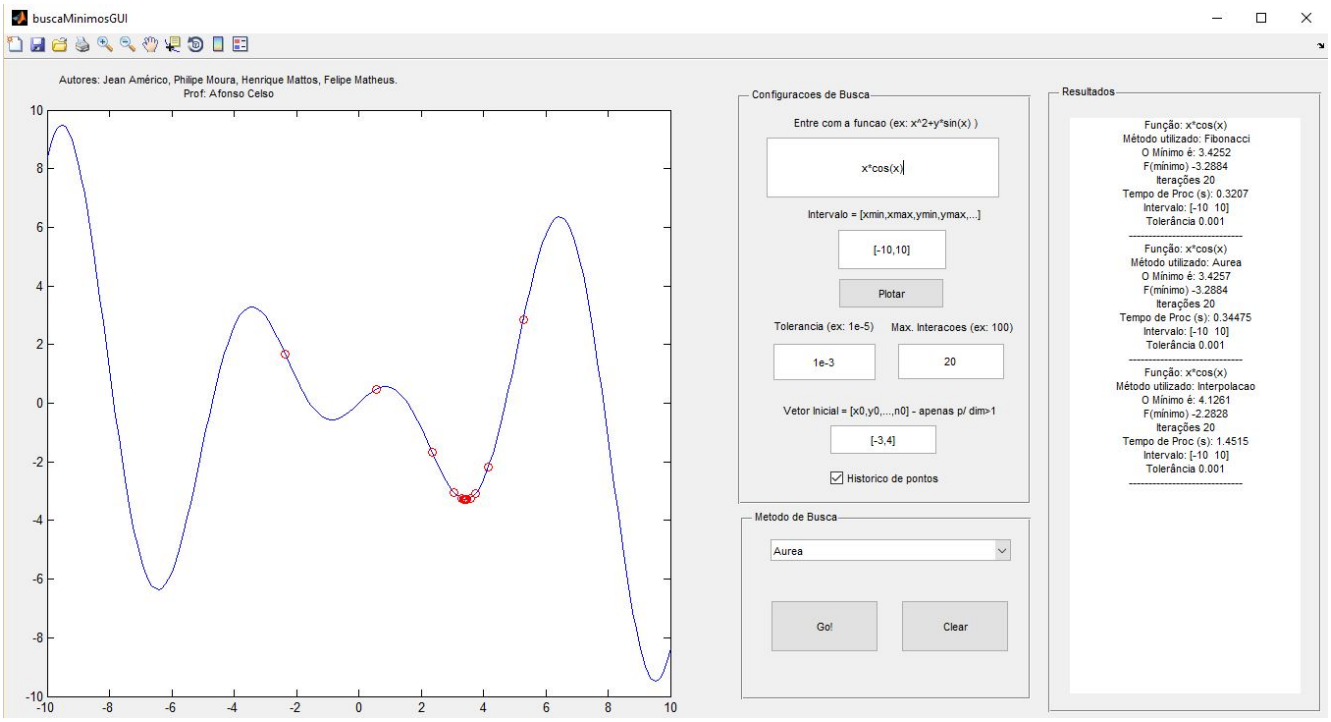

Figura 1: GUI desenvolvida pelo grupo

Além de funcional, um programa deve ser *user-friendly*. Pensando nisso, utilizamos a *Toolbox* GUIDE do MATLAB para desenvolver a interface gráfica do software e realizar a interação com o usuário final. De forma simples, fácil e intuitiva, o usuário pode entrar com os parâmetros da otimização:

- **Função:** uma expressão qualquer relacionando a variável 'x'. É permitindo utilizar funções pré-definidas do MATLAB, como sin(x), cos(x), exp(x), etc. Exemplo: `x^2, x^2+4*x^3+4, x^3*cos(-5*x)`
- **Intervalo:** Vetor intervalo da forma [a,b]. Onde 'a' representa o limite inferior e 'b' representa o limite superior. Para o caso de não haver argumentos o suficiente, o *plot* será padrão de [-10, 10]. Exemplo: [-4,4]
- **Botão Plotar:** Plota a função dentro do intervalo desejado. Esse mecanismo ajuda na escolha de um intervalo adequado antes de iniciar a busca.
- **Tolerância:** Valor numérico ou em forma de notação científica com a tolerância mínima. Exemplo: 0.0001 ou 1e-3
- **Máximo de Iterações:** Número máximo de iterações permitidas até o programa parar de procurar. Exemplo: 20, 100.
- **Vetor Inicial:** UTILIZADO APENAS NA SEGUNDA VERSÃO DO PROGRAMA, COM OTIMIZAÇÃO MULTIVARIÁVEL. IGNORAR.
- **Histórico de Pontos:** Se marcado, mostra no gráfico o conjunto de pontos encontrados em cada iteração.
- **Método de Busca:** Selecionar entre os três métodos possíveis para a otimização escalar: Fibonacci, Seção Aurea ou Interpolação Polinomial.

- **Botão Clear:** Limpa a caixa de textos Resultados à direita.
- **Botão Go!:** Inicia a busca.

A Apêndice

A.1 Método de Fibonacci

```
function [min, val, numI] = Fibo_Ph(expr, tol, limites, grafbool, lim_it)
    switch nargin
        case 3
            lim_it = -1;
            grafbool = 0;
        case 4
            lim_it = -1;
    end
    x=symvar(expr);
    if grafbool==1
        k=limites(1):0.1:limites(2);
        funcao=subs(expr,x,k);
        plot(k,funcao)
        hold on
    end

    raiz5 = (5)^(1/2);
    temp = (limites(2)-limites(1))/tol;
    Reducoes = 1;
    while ((raiz5/5)*(((1+raiz5)/2)^(Reducoes+1)-((1-raiz5)/2)^(Reducoes+1))) < temp %esse while determina
        Reducoes = Reducoes + 1;
    end
    if lim_it > Reducoes + 1
        q = Reducoes;
    end
    q = Reducoes + 1; %q is the new k. "k" foi reservada pelo jean
    p=(1-raiz5)/(1+raiz5);
    alfa = ((2/(1+raiz5))*((1-(p)^(q))/(1-(p)^(q+1))));
    numI=1;
    x4=limites(2);
    x1=limites(1);
    while x4-x1 > tol
        x1=limites(1);
        x4=limites(2);
        Linicial=limites(2)-limites(1);
        x2=alfa*x1+(1-alfa)*x4;
        f2=subs(expr,x,x2);
        x3=alfa*x4+(1-alfa)*x1;
        f3=subs(expr,x,x3);
        if f2<f3
            limites(1)=x1;
            limites(2)=x3;
            min=limites(2);
            Lfinal=limites(2)-limites(1);
            if numI==lim_it
                break;
            end
            alfa = (Linicial - Lfinal)/Lfinal;
            numI = numI + 1;
        else
            limites(1)=x2;
            limites(2)=x4;
            min=limites(1);
            Lfinal=limites(2)-limites(1);
            if numI==lim_it
                break;
            end
            alfa = (Linicial - Lfinal)/Lfinal;
            numI = numI + 1;
        end
        val=subs(expr,x,min);
        if grafbool ==1
```

```
            plot(min,val,'r-o');
    end
end
```

A.2 Método da Divisão Áurea

```
function [min,val, numI] = aurea4(expr,tol,limites,grafbool,lim_it)
%expr = string da equacao, tol = tolerancia, [li,lf] = vetor com lim_its
%inferior e superior, lim_it = lim_it de interacoes, grafbool = booleano
%que diz se quer ou nao o grafico
    switch nargin
        case 3
            lim_it = -1;
            grafbool = 0;
        case 4
            lim_it = -1;
    end
    %Setando defaults para as variaveis
    numI = 0;
    x = symvar(expr);
    ra = 0.61803398875;
    x1 = limites(1);
    x4 = limites(2);
    %verificar se deseja plotar
    if grafbool == 1
        k = limites(1):0.1:limites(2);
        funcao = subs(expr,x,k);
        plot(k,funcao)
        hold on
    end
    %inicio do algoritmo de razao aurea
    numI = 0;
    min = x1;
    val = subs(expr,x,min);
    x3 = x1+ra*(x4-x1);
    x2 = x1+(1-ra)*(x4-x1);
    f2 = double(subs(expr,x,x2));
    f3 = double(subs(expr,x,x3));
    while abs(x1-x4) > tol;
        numI = numI + 1;
        if numI == lim_it
            break;
        end
        if f2 > f3
            x1 = x2;
            x2 = x3;
            x3 = x4-(1-ra)*(x4-x1);
            f2 = double(subs(expr,x,x2));
            f3 = double(subs(expr,x,x3));
            min = x2;
            val = double(subs(expr,x,min));
        else
            x4 = x3;
            x3 = x2;
            x2 = x1+((1-ra)*(x4-x1));
            f2 = double(subs(expr,x,x2));
            f3 = double(subs(expr,x,x3));
            min = x3;
            val = double(subs(expr,x,min));
        end;
        if grafbool==1
            plot(min,val,'r-o');
        end
    end;
end
```

A.3 Método da Interpolação Polinomial

```
function [ Y1, Y2, Y3 ] = intpol_henrique( X1, X2, X3, X4, X5 )
v = X3(1):0.1:X3(2);
x = symvar(X1);
f = subs(X1, x, v);
plot(v, f);
hold on;
x1=X3(1);
x3=X3(2);
x2=(x1+x3)/2;
ra = (sqrt(5)-1)/2;
e = 1e-8;
fx1 = subs(X1, x, x1);
if X5
    plot (x1, fx1, 'go');
end
fx2 = subs(X1, x, x2);
if X5
    plot (x2, fx2, 'yo');
end
fx3 = subs(X1, x, x3);
if X5
    plot (x3, fx3, 'bo');
end
A = [double(x1^2) double(x1) double(1); double(x2^2) double(x2) double(1); double(x3^2) double(x3) double(1
B = [double(fx1); double(fx2); double(fx3)];
C = linsolve (A, B);
a = C(1,1);
b = C(2,1);
xmin = (-b)/(2*a);
if or(xmin<x1, xmin>x3)
    if x2-x1==x3-x2
        xmin = ra*x2 + (1 - ra)*x1;
    elseif x2-x1>x3-x2
        xmin = ra*x2 + (1 - ra)*x1;
    else
        xmin = ra*x3 + (1 - ra)*x2;
    end
end
fxmin = subs(X1, x, xmin);
if X5
    plot (xmin, fxmin, 'ko');
end
k = 1
while and(and(and(abs(xmin-x1)>X2, abs(xmin-x2)>X2), abs(xmin-x3)>X2), k~=X4)
    m = max(max(fx1, fx2), fx3);
    if and(x1<xmin, xmin<x2)
        if m==fx1
            x1 = xmin;
        elseif m==fx3
            x3 = x2;
            x2 = xmin;
        end
    else
        if m==fx1
            x1 = x2;
            x2 = xmin;
        elseif m==fx3
            x3 = xmin;
        end
    end
    fx1 = subs(X1, x, x1);
    fx2 = subs(X1, x, x2);
    fx3 = subs(X1, x, x3);
    if and(and(abs(fx3-fx2)<e, abs(fx3-fx1)<e), abs(fx2-fx1)<e)
        break;
```

```
    end
    if X5
        plot (x1, fx1, 'go');
        plot (x2, fx2, 'yo');
        plot (x3, fx3, 'bo');
    end
    A = [double(x1^2) double(x1) double(1); double(x2^2) double(x2) double(1); double(x3^2) double(x3) doub
    B = [double(fx1); double(fx2); double(fx3)];
    C = linsolve (A, B);
    a = C(1,1);
    b = C(2,1);
    xmin = (-b)/(2*a);
    if or(xmin<x1, xmin>x3)
        if x2-x1==x3-x2
            xmin = ra*x2 + (1 - ra)*x1;
        elseif x2-x1>x3-x2
            xmin = ra*x2 + (1 - ra)*x1;
        else
            xmin = ra*x3 + (1 - ra)*x2;
        end
    end
    fxmin = subs(X1, x, xmin);
    if X5
        plot (xmin, fxmin, 'ko');
    end
    k = k+1
end
plot (xmin, fxmin, 'ro');
Y1 = double (xmin);
Y2 = double (fxmin);
Y3 = double (k);
hold off
end
```