Universidade Federal do Rio de Janeiro
Programa de Engenharia Elétrica

Alunos: Bernardo Bouzan
Elly Fonseca Bouzan
Leonardo Santoro
Professor: Afonso Celso Del Nero
Data: 17/10/2010

Otimização – Aspectos Teóricos e Métodos Numéricos:

Trabalho 1

1 – Introdução

O relatório a seguir tem por objetivo apresentar brevemente o conceito utilizado nos métodos univariáveis da Seção Áurea, Fibonacci e de Interpolação Polinomial e, posteriormente, detalhar as construções de seus algoritmos a partir do software Matlab versão 7.8.0.

2 – Revisão Conceitual

2.1 – Método de Fibonacci

Dada uma função objetivo suave f(x), um intervalo inicial de busca I = [a,b] onde f(x) é unimodal e o número de iterações n que se deseja aplicar o algoritmo, deve-se obter um novo intervalo que contenha o mínimo da função objetivo no interior de [a,b].

2.1.1 – Algoritmo de Fibonacci

A - $p = \frac{1-\sqrt{5}}{1+\sqrt{5}}$; $\alpha = \frac{2}{1-\sqrt{5}} \frac{1-p^{n+1}}{1-p^{n+2}}$.

B - i = 1.

C - $x_1 = a$; $x_4 = b$; $L_{ini} = (b - a)$.

D - $x_2 = \alpha\, x_1 + (1 - \alpha)\, x_4$; $f_2 = f(x_2)$.

E - $x_3 = \alpha\, x_4 + (1 - \alpha)\, x_1$; $f_3 = f(x_3)$.

F - Se $f_2 < f_3$:

> $a = x_1$; $b = x_3$; $L_{fin} = (b - a)$.
>
> Se i = n $\longrightarrow$ I = [a,b] $\longrightarrow$ FIM
>
> $\alpha = (L_{ini} - L_{fin}) / L_{fin}$; i = i + 1.
>
> Retornar a C.

G - Se $f_2 \geq f_3$:

> $a = x_2$; $b = x_4$; $L_{fin} = (b - a)$.
>
> Se i = n $\longrightarrow$ I = [a,b] $\longrightarrow$ FIM
>
> $\alpha = (L_{ini} - L_{fin}) / L_{fin}$; i = i + 1.
> Retornar a C.

2.2 – Método da Razão Áurea

A idéia básica consiste em dada uma função objetivo suave f(x), um intervalo inicial de busca [a,b] onde f(x) é unimodal e uma tolerância ε, reduzir o intervalo fornecido de forma que o mesmo seja inferior a tolerância desejada.

2.2.1 – Algoritmo da Razão Áurea

A - x_1= a; x_4= b; L = (b - a); r = $\frac{(\sqrt{5}-1)}{2}$.

B - $x_2 = r\,x_1 + (1-r)\,x_4$.

C - $x_3 = (1-r)x_1 + r\,x_4$.

D - $f_2 = f(x_2)$; $f_3 = f(x_3)$.

E - Se $f_2 < f_3$:

a = x_1; b = x_3; L = (b – a).

Se L < ε ⟶ I = [a,b] ⟶ FIM

$x_3 = x_2$.

x_2 = r a + (1 – r) b ⟶ Retornar a D.

F - Se $f_2 \geq f_3$:

a = x_2; b = x_4; L = (b – a).

Se L < ε ⟶ I = [a,b] ⟶ FIM

$x_2 = x_3$.

x_3 = (1 – r) a + r b ⟶ Retornar a D.

2.3 – Método de Interpolação Polinomial

A idéia central do algoritmo é, na vizinhança de um ponto mínimo, aproximar a função objetivo f(x) por parábolas quadráticas. Portanto, dado um intervalo de busca inicial I = [a,b], deve-se escolher um ponto p pertencente à I e calcular a equação da parábola g(x) que passa por a, b e p. Posteriormente, será necessário calcular o ponto x_{min} que causa um mínimo em g(x) e descartar um ponto entre a, b e p utilizando como critério qual deles causa maior retorno em g(x). Deve-se então repetir todos os passos com o novo intervalo gerado até que a tolerância ε seja satisfeita.

2.3.1 – Algoritmo da Interpolação Quadrática

A - x_1= a; x_3= b.

B - $x_2 = r\,x_1 + (1 - r)\,x_3$.

C - $f_1 = f(x_1)$; $f_2 = f(x_2)$; $f_3 = f(x_3)$.

D - $x_{mín} = \dfrac{(x_2^2 - x_3^2)f_1 + (x_3^2 - x_1^2)f_2 + (x_1^2 - x_2^2)f_3}{2(x_2 - x_3)f_1 + (x_3 - x_1)f_2 + (x_1 - x_2)f_3}$;

$f_{mín} = f(x_{mín})$.

E - Se $\exists$ i tal $|x_{mín} - x_i| < \varepsilon \longrightarrow$ FIM.

F - Se $(x_3 - x_{mín})\,(x_{mín} - x_2) > 0$: k = 1;

Senão: k = 3.

G - Se $f_2 > f_{mín}$:

$x_k = x_2$; $f_k = f_2$; k = 2.

H – $x_{4-k} = x_{mín}$; $f_{4-k} = f_{mín} \longrightarrow$ Retornar a D.

2.4 – Método do Enquadramento Inicial

2.4.1 – O Critério dos 3 pontos

Sejam $x_1, x_2, x_3 \in \mathbb{R}$ tais que $x_1 < x_2 < x_3$ e uma função $f : \mathbb{R} \rightarrow \mathbb{R}$ suave. Se $f(x_1) \geq f(x_2) < f(x_3)$ então o intervalo $[x_1\ x_3]$ enquadra um mínimo.

2.4.2 – Método do Enquadramento Inicial

O objetivo do algoritmo é encontrar x_1, x_2 e x_3 que satisfaçam o critério dos 3 pontos e possam servir como intervalo enquadrante inicial. Dados inicialmente o ponto x_1 e um avanço Δ, calculamos $x_2 = x_1 + \Delta$. Se $f_2 \leq f_1$ isto significa que estamos andando no sentido correto e devemos continuar neste sentido até encontrar x_3 tal que $f_3 > f_2$. Quando o resultado da primeira verificação é $f_2 > f_1$ a função está crescendo e devemos inverter o sentido da busca (passo 4 abaixo).

A: $x_2 = x_1 + \Delta$

B: $f_1 = f(x_1)$ e $f_2 = f(x_2)$

C: se $f_2 \leq f_1$ -> Passo E (continua avançando)

D: a = x_1, b = x_2, x_1 = b, x_2 = a, $\Delta = -\Delta$ (inversão de sentido)

E: $x_3 = x_2 + \gamma\Delta$ e $f_3 = f(x_3)$ (continua avançando)

F: $f_3 > f_2$ -> Passo H (continua avançando)

G: $x_1 = x_2$, $x_2 = x_3$ -> Passo E (recomeça de x_2)

H: a = x_1, b = x_3

O fator de expansão pode ser dado por $\gamma = 1$ (todos os avaços iguais a Δ), ou usa-se $\gamma = 2$ ou $\gamma = 1{,}618$ (razão áurea) para acelerar o procedimento.

3 – Manual de Execução dos Algoritmos

Todos os códigos estão presentes para consulta no ANEXO 1.

Para inicializar um dos métodos, a função *optimize* deve ser executada a partir da linha de comando do Matlab, conforme mostrado na Figura 1.

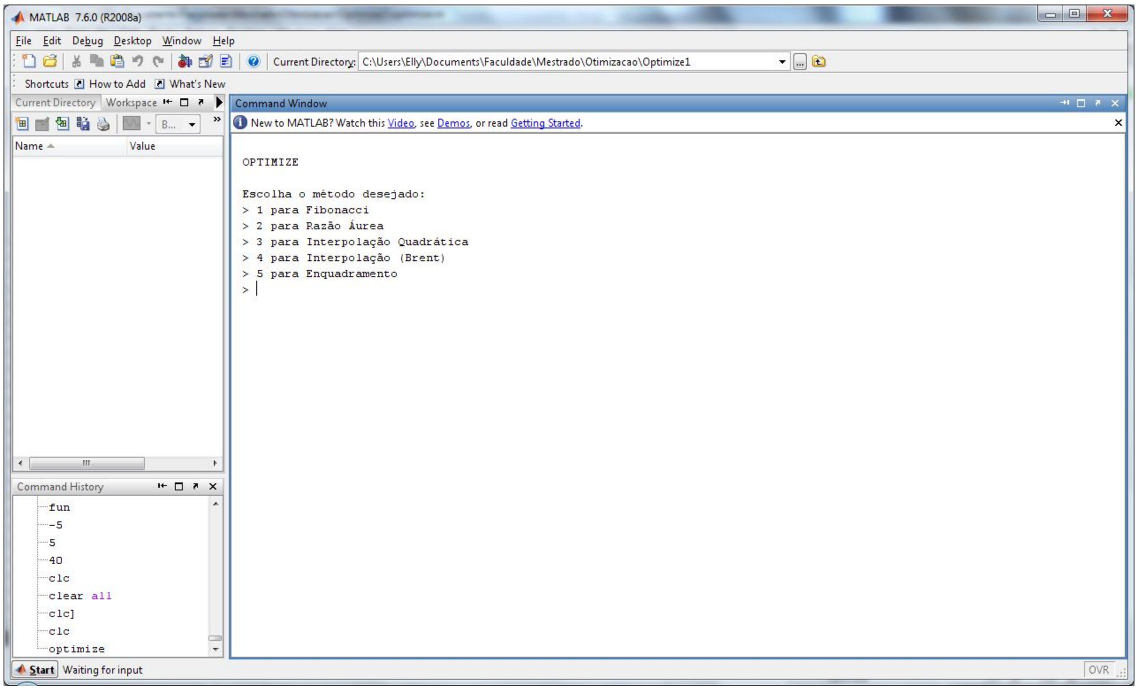

Figura 1 – Iniciando a execução da função Optimize

Nota-se que, uma vez iniciada a função, deve-se escolher qual método será utilizado para reduzir o intervalo de busca inicial, sendo 1 correspondente ao método de Fibonacci, 2 a Seção Áurea, 3 a Interpolação Quadrática, e 4 a Interpolação (Brent). O número 5 é o método do Enquadramento Inicial.

3.1 – Método de Fibonacci

Conforme descrito na seção 2.1, uma vez escolhido o método de Fibonacci, deve-se definir o intervalo de busca inicial, e também o número de iterações que o algoritmo realizará. O intervalo de busca inicial pode ser obtido através do método do Enquadramento Incial, explicado na seção 3.5 deste relatório. A Figura 2, apresentada a seguir, exemplifica o uso do método considerando $f(x) = (x-4)^3 + 4(x-4)^2 + 1$, intervalo de busca inicial $I_0 = [3,8674 ; 4,191]$, e número de iterações $n = 10$.

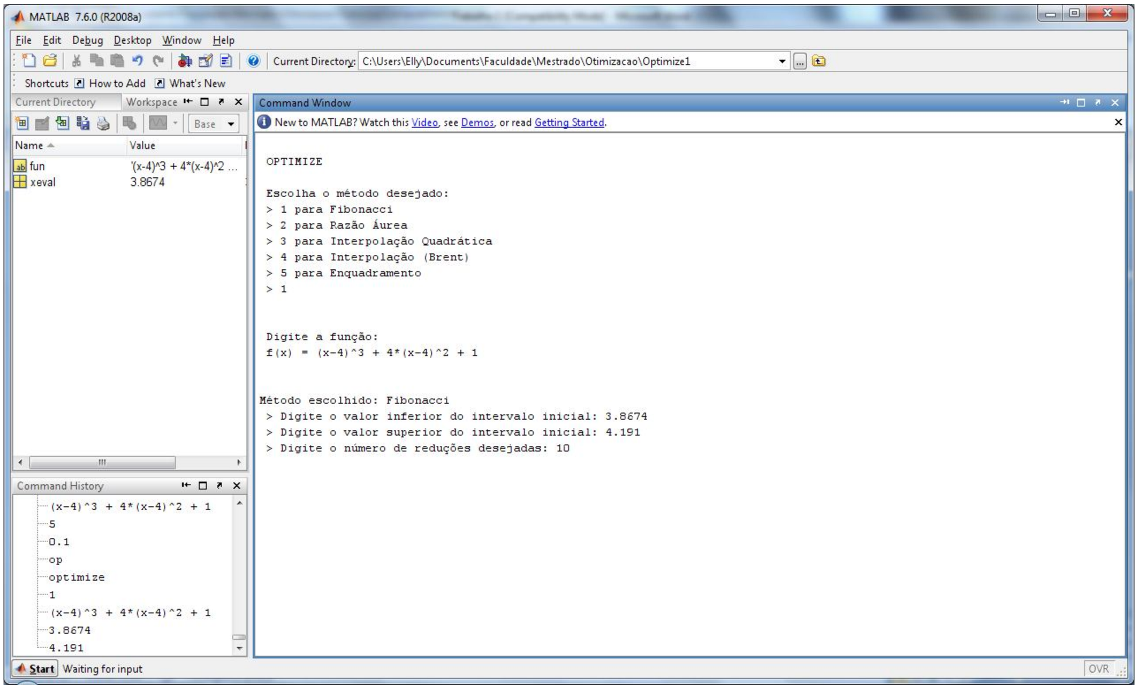

Figura 2 – Método de Fibonacci para I_0 = [3,8674 ; 4,191], n = 10

Observando-se a função objetivo proposta acima, pode-se calcular analiticamente e constatar que a mesma apresenta um ponto de mínimo em x = 4. Utilizando o Método de Fibonacci com apenas 10 iterações, pode-se afirmar que o mínimo procurado encontra-se no interior do intervalo I_f = [3,9979 ; 4,0021], conforme Figura 3. A Figura 4 exibe a evolução do enquadramento de busca a cada iteração.

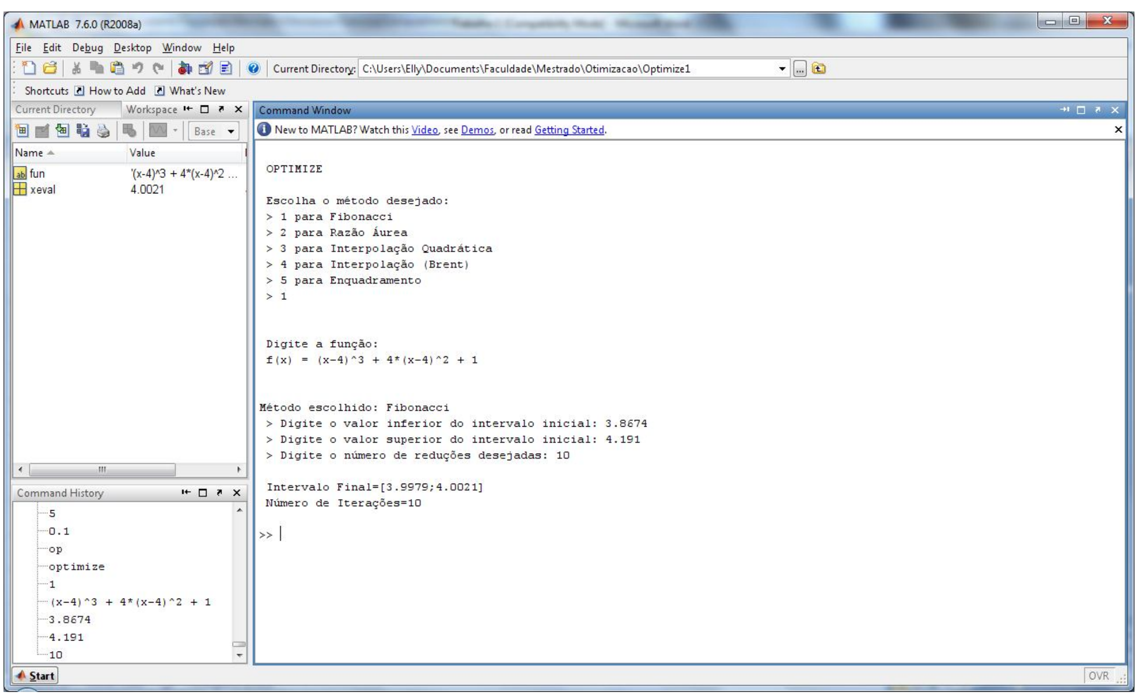

Figura 3 – Resultado do método de Fibonacci para n = 10.

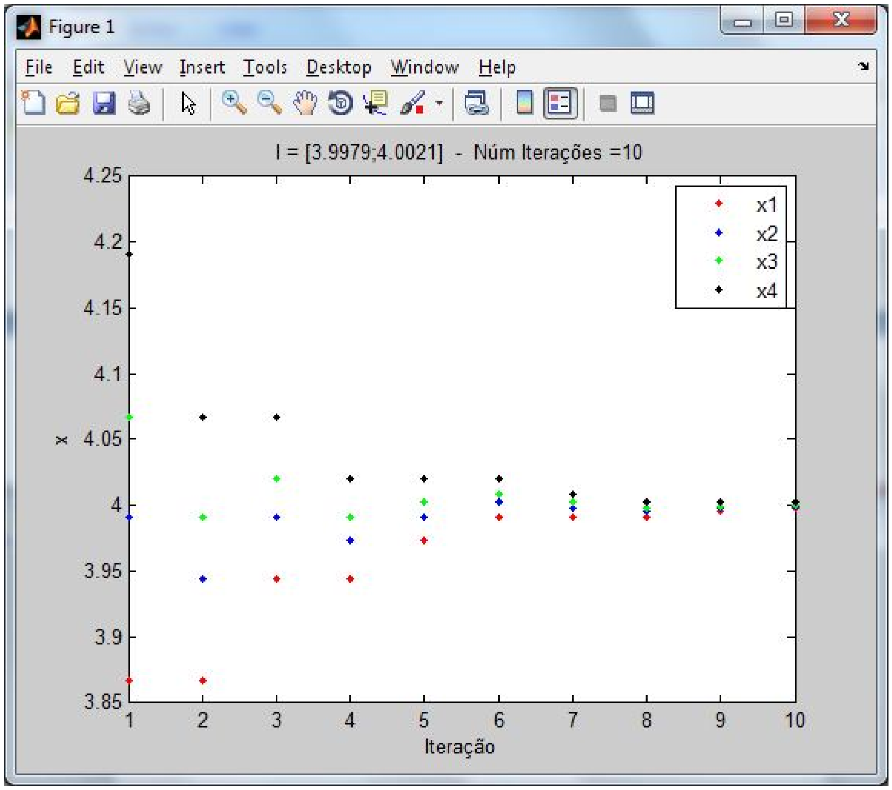

Figura 4 – Fibonacci: Evolução do enquadramento de busca para I_0 = [3,8674 ; 4,191] e n = 10.

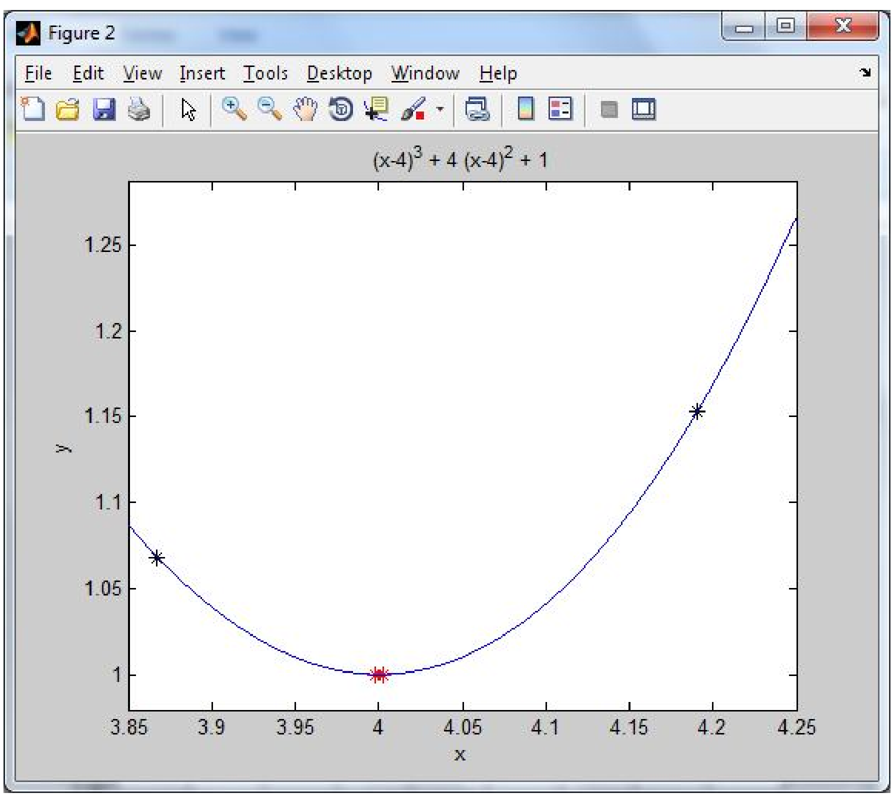

Figura 5 – Fibonacci: Resultado da minimização

3.2 – Método da Seção Áurea

De acordo com a seção 2.2, para execução do algoritmo da Seção Áurea, deve-se definir o intervalo de enquadramento inicial I_0 e também a tolerância desejada para o tamanho do intervalo desejado. Inicialmente será considerada novamente a função objetivo $f(x) = (x-4)^3 + 4(x-4)^2 + 1$, assim como o intervalo de busca inicial $I_0 = [3{,}8674 ; 4{,}191]$. Para que seja possível a comparação entre o desempenho alcançado com esse método e o de Fibonacci, será considerada a tolerância $\varepsilon = 4{,}0826 - 3{,}9506 = 0{,}132$ (resultado obtido anteriormente com 10 iterações). A Figuras 6 e 7 exibem os resultados para os valores de entrada mostrados acima.

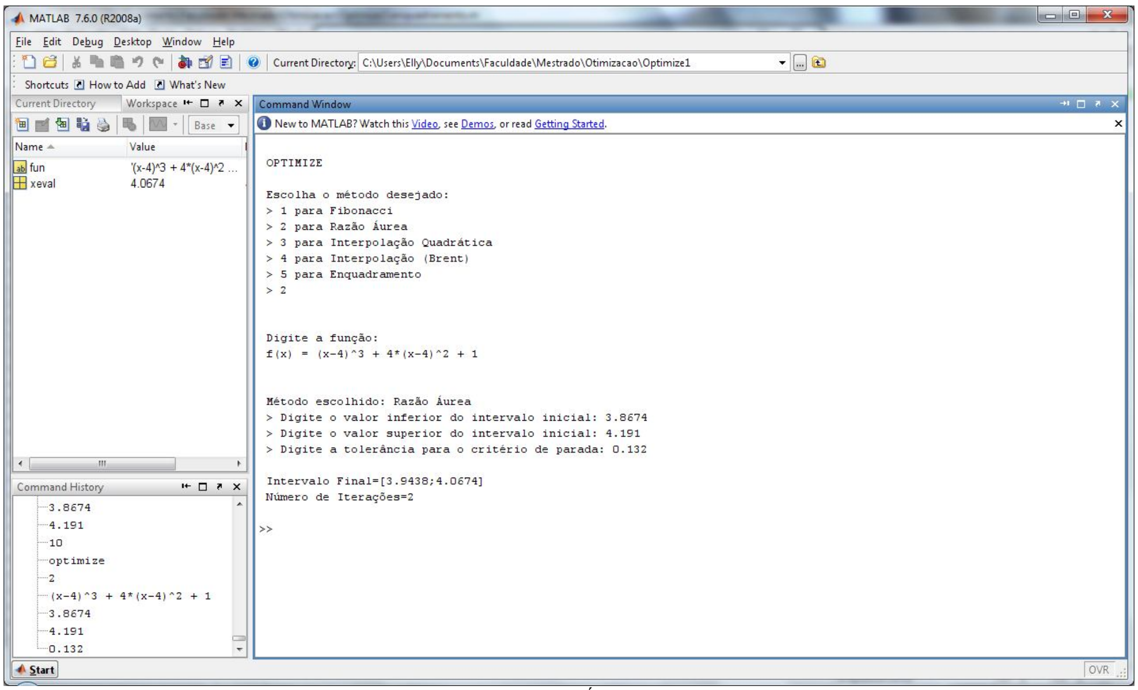

Figura 6 – Resultados para o Método da Razão Áurea para $I_0 = [3{,}8674 ; 4{,}191]$ e $\varepsilon = 0.132$.

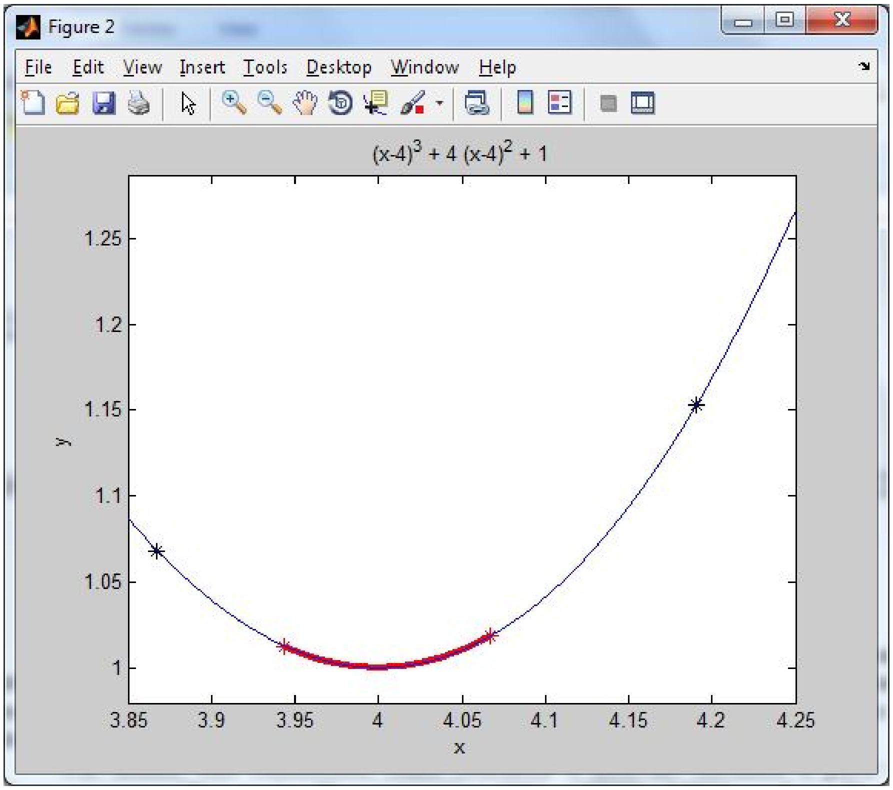

Figura 7 – Razão: Evolução do enquadramento de busca para I_0 = [3,8674 ; 4,191] e ε = 0.132.

Nota-se, a partir das Figuras 6 e 7, que para a mesma função objetivo usada anteriormente, o método da Seção Áurea realiza apenas 2 iterações, enquanto Fibonacci necessitou de 10. O intervalo final encontrado foi I_f = [3,9438 ; 4,0674], mostrando que de fato o mesmo contém o ponto de mínimo x = 4, calculado analiticamente.

3.3 – Método de Interpolação Polinomial

Conforme descrito na seção 2.3, para execução do método de Interpolação Polinomial, deve-se especificar, além da função objetivo, o intervalo de enquadramento inicial I_0 e o fator de tolerância ε. Novamente serão utilizados $f(x) = (x-4)^3 + 4(x-4)^2 + 1$ (definida em fun.m) e intervalo de busca inicial I_0 = [3,8674 ; 4,191]. Inicialmente, foi escolhida a tolerância ε = 0,01, como mostrado nas Figuras 8 e 9.

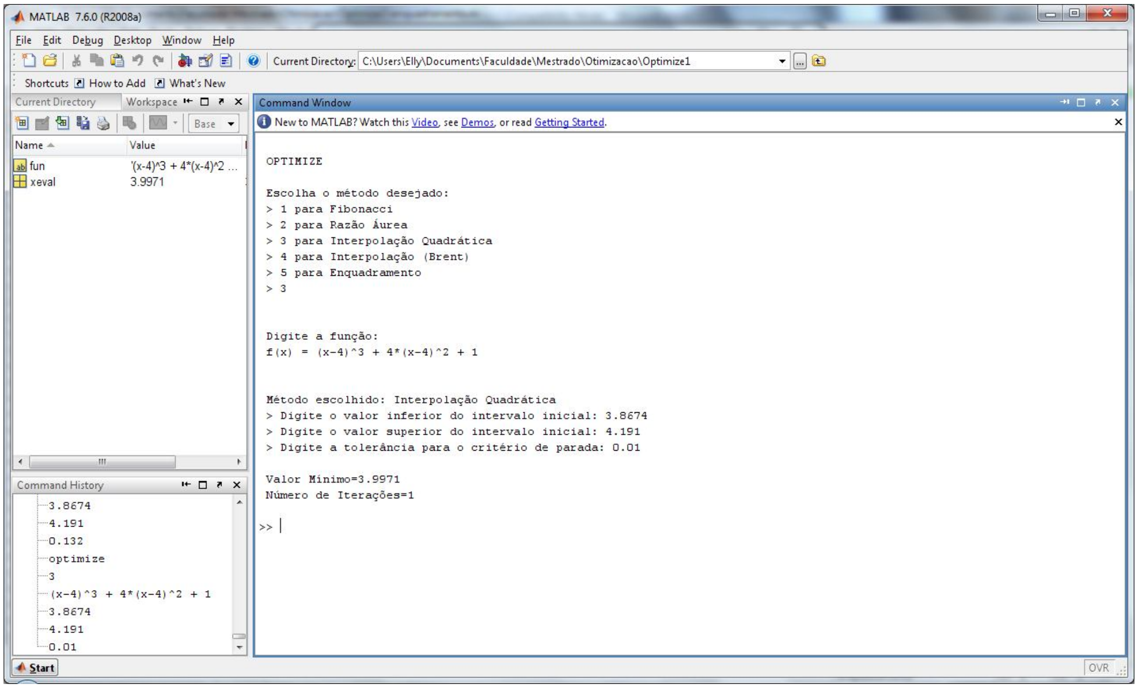

Figura 8 - Resultados para o Método da Interpolação Quadrática para I_0 = [3,8674 ; 4,191] e ε = 0.01.

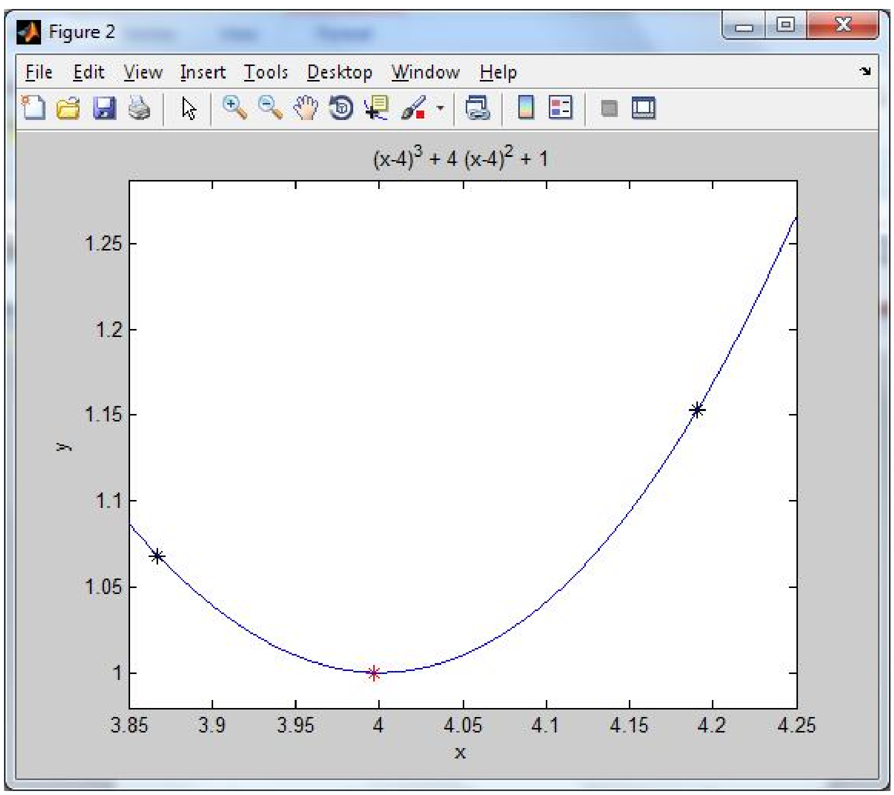

Figura 9 - Interpolação Quadrática: Evolução do enquadramento de busca para I_0 = [3,8674 ; 4,191] e ε = 0.01.

Note que apesar de a tolerância inserida ao algoritmo ser ε = 0,01, o método é interrompido quando existe x_i tal que $|x_{mín} - x_i| < ε$, sendo $x_{mín}$ correspondente ao ponto de mínimo da parábola interpolada g(x). Assim, nos casos em que $x_{mín}$ calculado para g(x) não é idêntico ao ponto de mínimo calculado analiticamente para a função objetivo f(x), o ponto encontrado ao fim do método de interpolação quadrática não necessariamente estará na tolerância especificada inicialmente pelo usuário.

Por fim, pode-se observar que com apenas 1 iteração, o algoritmo acima se aproximou do mínimo analítico possuindo um erro menor que 0,1 %.

Repetindo-se a simulação, porém com tolerância ε = 0,001, chega-se aos resultados das Figuras 10 e 11.

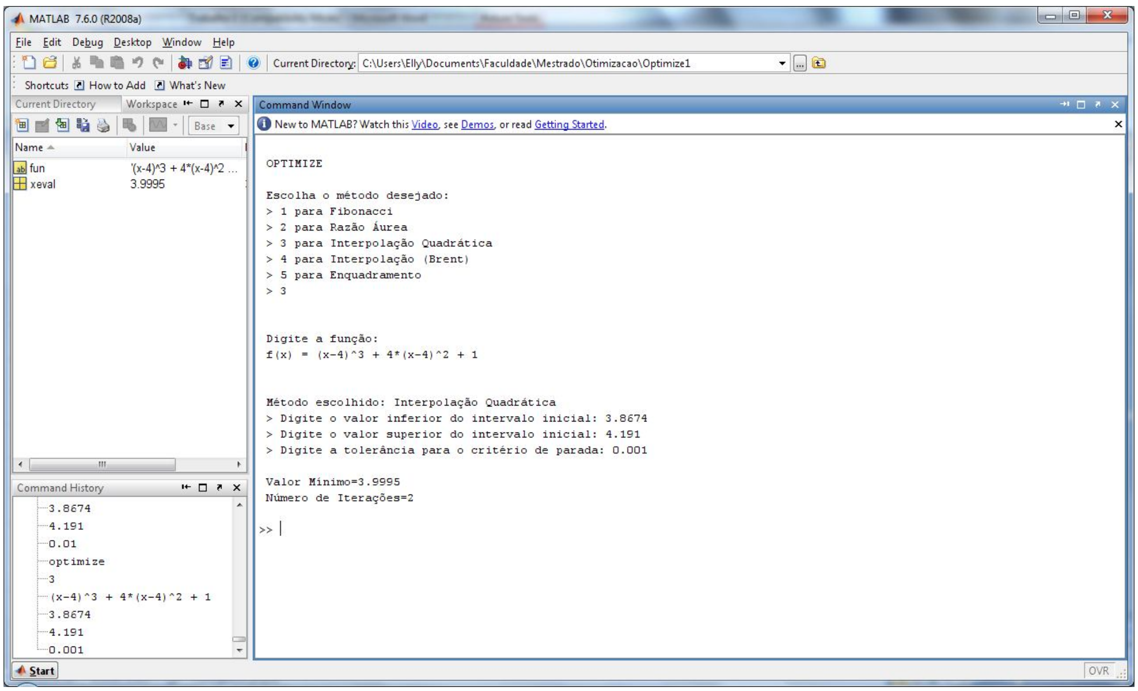

Figura 10 - Resultados para o Método da Interpolação Quadrática para I_0 = [3,8674 ; 4,191] e ε = 0.001

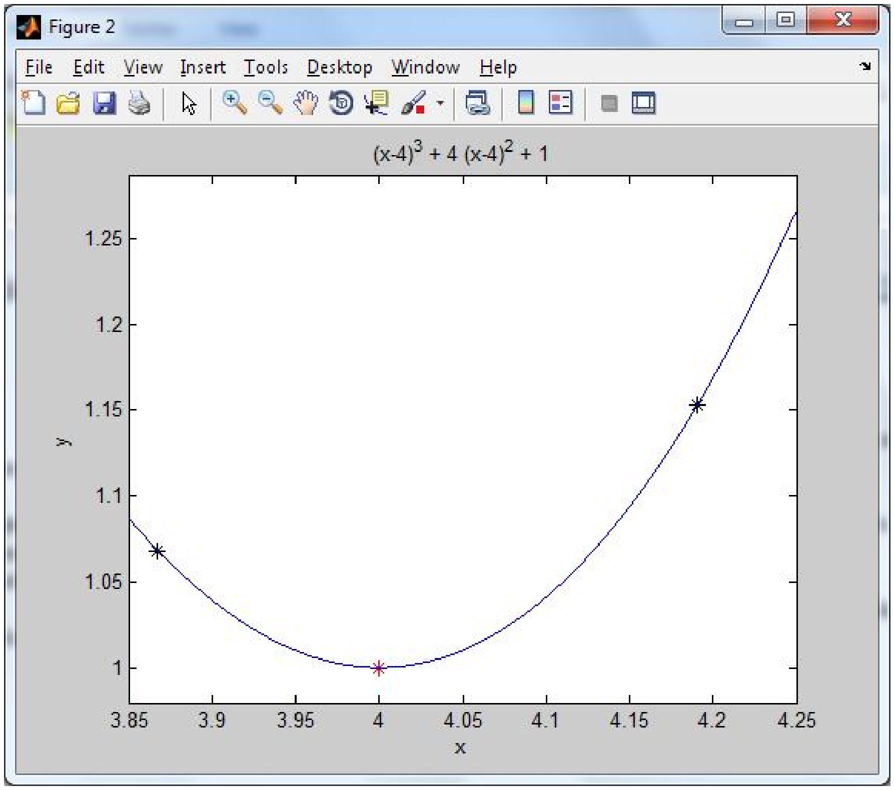

Figura 11 - Interpolação Quadrática: Evolução do enquadramento de busca para $I_0 = [3{,}8674 ; 4{,}191]$ e $\varepsilon = 0.001$.

Dessa vez, o algoritmo encontra o ponto de mínimo calculado praticamente sem erros, porém o número de iterações aumentou para i = 2.

3.4 – Algoritmo de Brent

Por último, foi implementado o algoritmo de Brent, que possui idéia semelhante a apresentada anteriormente, conforme ANEXO I. Novamente, a tolerância especificada pelo usuário não necessariamente será atendida pelo ponto de mínimo encontrado através do método.

Utilizando $I_0 = [3{,}8674 ; 4{,}191]$ e $\varepsilon = 0.001$, os resultados apresentados nas Figuras 12 e 13 são alcançados.

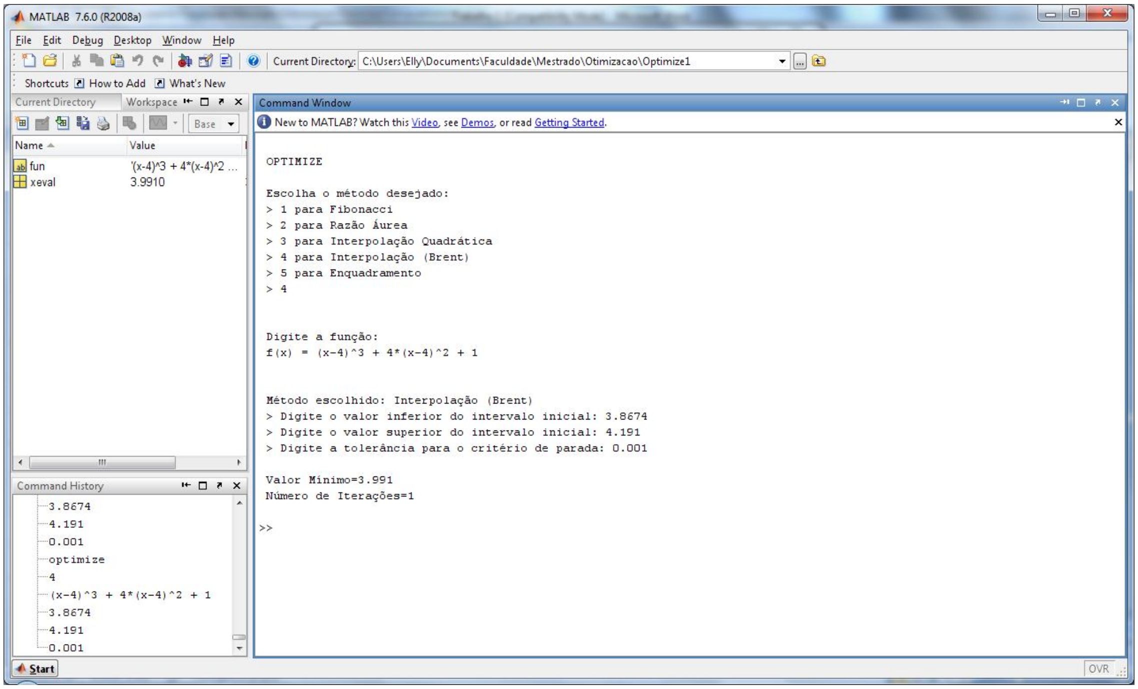

Figura 12 - Resultados para o Método de Brent para I_0 = [3,8674 ; 4,191] e ε = 0.001

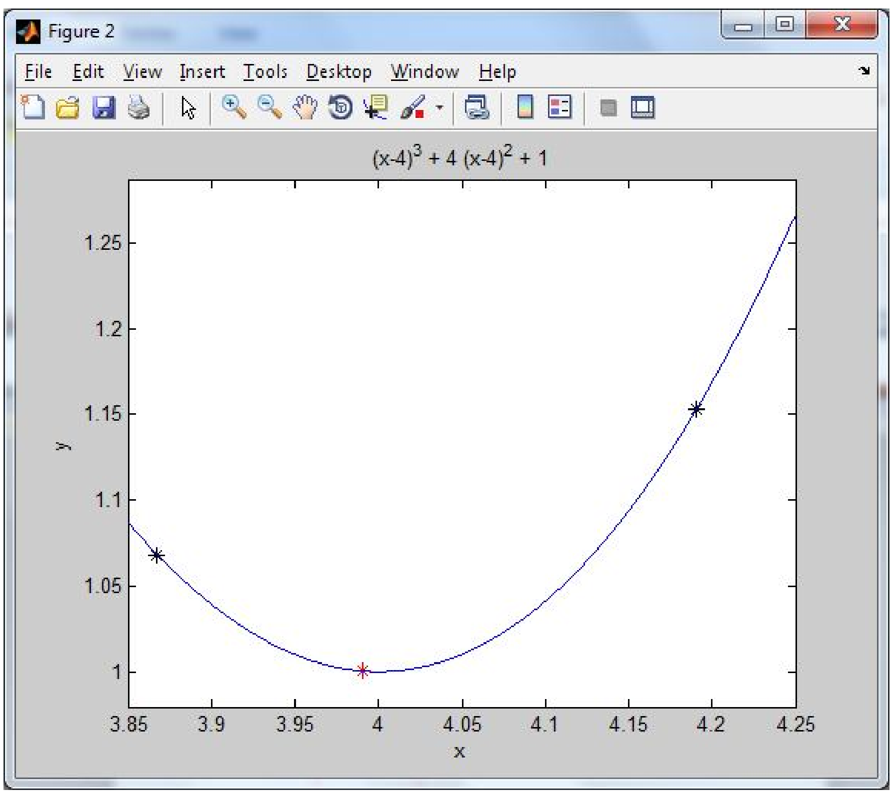

Figura 13 - Brent: Evolução do enquadramento de busca para I_0 = [3,8674 ; 4,191] e ε = 0.001.

Com apenas uma iteração, o método de Brent alcançou o resultado com 0,2% de erro.

3.5 – Algoritmo do Enquadramento Inicial

O Algoritmo do Enquandramento Inicial deve ser executado antes de qualquer um dos métodos de busca linear. Dado um ponto inicial e um delta de avanço fixo, ele retorna um o intervalo inicial de enquadramento da busca linear. A figura 14 mostra o programa optimize executado no modo Enquadramento Inicial para a função $f(x) = (x-4)^3 + 4(x-4)^2 + 1$. O ponto inicial usado foi 5, e o avaço inicial igual a 0,1.

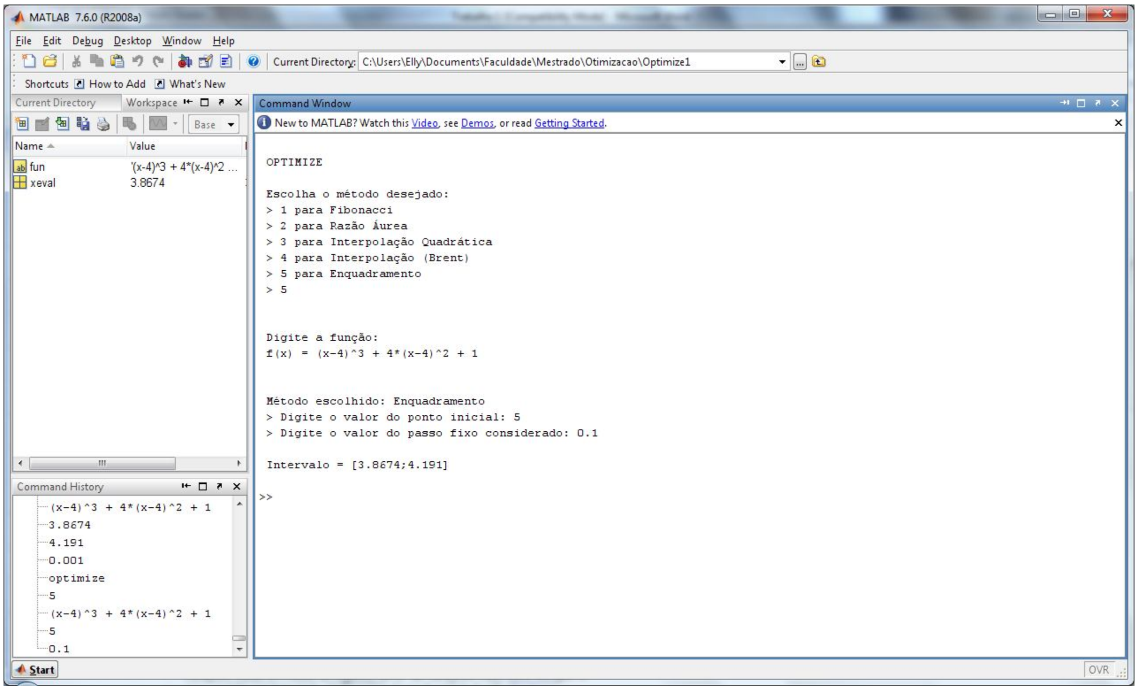

Figura 14 - Enquadramento: Resultado do m'etodo para I = 5 e Δ = 0,1.

Pela figura 14, observa-se que o intervalo de enquadramento encontrado foi [3,8674 ; 4,191], que engloba o mínimo analítico 4. Este valor foi usado como intervalo inicial para todos os algotimos de busca linear deste relatório.

ANEXO I

- fun.m

```
function f = fun(xeval)

assignin('base','xeval',xeval);
f = evalin('base','subs(fun,xeval);');
```

- optimize.m

```
function optimize

clear all;
clc;
format long;

fprintf(1,'\n OPTIMIZE\n\n');

% ------------ Entradas
eval_fun='fun';

method = input(' Escolha o método desejado: \n > 1 para Fibonacci \n > 2
para Razão Áurea \n > 3 para Interpolação Quadrática \n > 4 para
Interpolação (Brent) \n > 5 para Enquadramento \n > ');

fun = input('\n\n Digite a função: \n f(x) = ','s');
assignin('base','fun',fun);

if method == 1

    fprintf(1,'\n\nMétodo escolhido: Fibonacci \n');

    I1(1) = input(' > Digite o valor inferior do intervalo inicial: ');
    I1(2) = input(' > Digite o valor superior do intervalo inicial: ');
    nit = input(' > Digite o número de reduções desejadas: ');

    [I,cont_it] = fibonacci(eval_fun,I1,nit);

    fprintf(1,strcat('\n Intervalo
Final=[',num2str(I(1)),';',num2str(I(2)),']'));
    fprintf(1,strcat('\n Número de Iterações=',num2str(cont_it),'\n\n'));

    grafico(eval_fun,I1,I);
end

if method == 2

    fprintf(1,'\n\n Método escolhido: Razão Áurea \n');

    I1(1) = input(' > Digite o valor inferior do intervalo inicial: ');
    I1(2) = input(' > Digite o valor superior do intervalo inicial: ');
    eps = input(' > Digite a tolerância para o critério de parada: ');
```

```
    [I,cont_it] = aurea(eval_fun,I1,eps);

    fprintf(1,strcat('\n Intervalo
Final=[',num2str(I(1)),';',num2str(I(2)),']'));
    fprintf(1,strcat('\n Número de Iterações=',num2str(cont_it),'\n\n'));

    grafico(eval_fun,I1,I);
end

if method == 3

    fprintf(1,'\n\n Método escolhido: Interpolação Quadrática \n');

    I1(1) = input(' > Digite o valor inferior do intervalo inicial: ');
    I1(2) = input(' > Digite o valor superior do intervalo inicial: ');
    eps = input(' > Digite a tolerância para o critério de parada: ');

    [I,cont_it] = interpolacao(eval_fun,I1,eps);

    fprintf(1,strcat('\n Valor Mínimo= ',num2str(I)));
    fprintf(1,strcat('\n Número de Iterações=',num2str(cont_it),'\n\n'));

    grafico(eval_fun,I1,I);
end

if method == 4

    fprintf(1,'\n\n Método escolhido: Interpolação (Brent) \n');

    I1(1) = input(' > Digite o valor inferior do intervalo inicial: ');
    I1(2) = input(' > Digite o valor superior do intervalo inicial: ');
    eps = input(' > Digite a tolerância para o critério de parada: ');

    [I,cont_it] = brent(eval_fun,I1,eps);

    fprintf(1,strcat('\n Valor Mínimo= ',num2str(I)));
    fprintf(1,strcat('\n Número de Iterações=',num2str(cont_it),'\n\n'));

    grafico(eval_fun,I1,I);
end

if method == 5

    fprintf(1,'\n\n Método escolhido: Enquadramento \n');

    po = input(' > Digite o valor do ponto inicial: ');
    passo = input(' > Digite o valor do passo fixo considerado: ');

    [I] = enquadramento(eval_fun,po,passo);

    fprintf(1,strcat('\n Intervalo =
[',num2str(I(1)),';',num2str(I(2)),']', '\n\n'));

end
```

- **enquadramento.m**

```
function  [I] = enquadramento(fun,po,passo)

exp = 1.618;

continua = 1;

x(1) = po;

f(1) = feval(fun,x(1));

x(2) = x(1) + passo;

f(2) = feval(fun,x(2));

if f(1) < f(2)

        a = x(1);
        fa = f(1);
        b = x(2);
        fb = f(2);
        x(1) = b;
        f(1) = fb;
        x(2) = a;
        f(2) = fa;
        passo = - passo;

end

while continua

    x(3) = x(2) + exp*passo;

    f(3) = feval(fun,x(3));

    if f(2) >= f(3)

        x(1) = x(2);
        f(1) = f(2);
        x(2) = x(3);
        f(2) = f(3);

    else

        continua = 0;

    end
end

if x(1) < x(3)
    a = x(1);
    b = x(3);
else
    a = x(3);
    b = x(1);
end
```

```
I = [a b]';
```

- fibonacci.m

```
function [I,cont_it] = fibonacci(fun,I1,nit)

% ------------ Fibonacci        A---------B-----C---------D
k = nit+1;
p = (1-sqrt(5))/(1+sqrt(5));
alpha = (2/(1+sqrt(5)))*(1-p^k)/(1-p^(k+1));

a = I1(1);
b = I1(2);

for cont_it = 1 : nit

    x1 = a;
    x4 = b;
    Lini = a - b;
    x2 = alpha*x1 + (1-alpha)*x4;
    x3 = alpha*x4 + (1-alpha)*x1;

    [f1]=feval(fun,x1);
    [f4]=feval(fun,x4);
    [f2]=feval(fun,x2);
    [f3]=feval(fun,x3);

    plot(cont_it,x1,'r.');
    hold on
    plot(cont_it,x2,'b.');
    plot(cont_it,x3,'g.');
    plot(cont_it,x4,'k.');

    if  f2 < f3

        a = x1;
        b = x3;
        Lfin = a - b;

        alpha = (Lini - Lfin)/(Lfin);

    else
        a = x2;
        b = x4;
        Lfin = a - b;
    end

end

I = [x1 x4]';

legend('x1','x2','x3','x4');
title(strcat('I = [',num2str(I(1)),';',num2str(I(2)),']  -  Núm Iterações =
',num2str(cont_it)));
xlabel('Iteração');
ylabel('x');
```

- **aurea.m**

```
function [I,cont_it] = aurea(fun,I1,eps)

% ------------ Razao Áurea        A---------B-----C---------D
xa = I1(1);
xd = I1(2);
xc = (((xd)-xa)*((-1+sqrt(5))/2)+xa);
xb = xd-(xc-xa);

[fa]=feval(fun,xa);
[fd]=feval(fun,xd);
[fc]=feval(fun,xc);
[fb]=feval(fun,xb);

intervalo = xd - xa;

cont_it = 0;

while (intervalo > eps)

    cont_it = cont_it + 1;

    plot(cont_it,xa,'r.');
    hold on
    plot(cont_it,xb,'b.');
    plot(cont_it,xc,'g.');
    plot(cont_it,xd,'k.');

    if  fb > fc
        xa = xb;
        fa = fb;
        xb = xc;
        fb = fc;
        % D nao muda
        xc = xa + xd - xb;
        [fc]=feval(fun,xc);
    else
        xd = xc;
        fd = fc;
        xc = xb;
        fc = fb;
        % A nao muda
        xb = xd + xa - xc;
        [fb]=feval(fun,xb);
    end

    intervalo = norm(xd - xa);
end

I = [xa xd]';

legend('A','B','C','D');
title(strcat('I = [',num2str(I(1)),';',num2str(I(2)),']  -  Núm Iterações =
',num2str(cont_it)));
xlabel('Iteração');
```

```
ylabel('x');
```

- interpolacao.m

```
function [I,cont_it] = interpolacao(fun,I1,eps)

x(1) = I1(1);
x(3) = I1(2);
x(2) = 0.5*(x(1) + x(3));

[f(1)]=feval(fun,x(1));
[f(3)]=feval(fun,x(3));
[f(2)]=feval(fun,x(2));

cont_it = 0;

xmin = 0.5*((((x(2)^2 - x(3)^2)*f(1)) + ((x(3)^2 - x(1)^2)*f(2)) + ((x(1)^2
- x(2)^2)*f(3)))/(((x(2) - x(3))*f(1)) + ((x(3) - x(1))*f(2)) + ((x(1) -
x(2))*f(3))));
[fmin]=feval(fun,xmin);

while ((abs(xmin - x(1)) > eps) &&  (abs(xmin - x(2)) > eps) && (abs(xmin -
x(3)) > eps))

    cont_it = cont_it + 1;

    plot(cont_it,x(1),'r.');
    hold on
    plot(cont_it,x(2),'b.');
    plot(cont_it,x(3),'g.');

    if  (x(3) - xmin)*(xmin - x(2)) > 0
        k = 1;
    else
        k = 3;
    end

    if (fmin < f(2))
       x(k) = x(2);
       f(k) = f(2);
       k = 2;
    end

    x(4-k) = xmin;
    f(4-k) = fmin;
    [f(1)]=feval(fun,x(1));
    [f(3)]=feval(fun,x(3));
    [f(2)]=feval(fun,x(2));
xmin = 0.5*((((x(2)^2 - x(3)^2)*f(1)) + ((x(3)^2 - x(1)^2)*f(2)) + ((x(1)^2
- x(2)^2)*f(3)))/(((x(2) - x(3))*f(1)) + ((x(3) - x(1))*f(2)) + ((x(1) -
x(2))*f(3))));
    [fmin]=feval(fun,xmin);

end

X = [ abs(xmin - x(1)); abs(xmin - x(2)); abs(xmin - x(3))];
```

```
xminimo = min(X, [], 1) + xmin;

I = xminimo;

legend('A','B','C');
title(strcat('Valor Mínimo = [',num2str(I),']  -  Núm Iterações =
',num2str(cont_it)));
xlabel('Iteração');
ylabel('x');
```

- brent.m

```
function [I,cont_it] = brent(fun,I1,eps)

% ------------ Brent
a = I1(1);
b = I1(2);
c = 0.5*(3.0 - sqrt(5.0));

x = a + c*(b - a);
v = x;
w = x;
d = 0;
e = d;

[fa]=feval(fun,a);
[fb]=feval(fun,b);
[fx]=feval(fun,x);
[fv]=feval(fun,v);
[fw]=feval(fun,w);

cont_it = 0;

while (true)

    cont_it = cont_it + 1;

    m = 0.5*(a + b);

    tol = sqrt(eps)*abs(x) + eps;
    t2 = 2.0*tol;
    if (abs(x - m) <= t2 - 0.5*(b - a))
        break;
    else
        p = 0;
        q = 0;
        r = 0;

        if (abs(e) > tol)
            r = (x - w)*(fx - fv);
            q = (x - v)*(fx - fw);
            p = (x - v)*q - (x - w)*r;
            q = 2.0*(q - r);
            if (q > 0.0)
                p = -p;
            else
                q = -q;
            end
            r = e;
```

```
    e = d;
end

if (abs(p) < abs(0.5*q*r) && p < q*(a - x) && p < q*(b - x))
    d = p/q;
    u = x + d;
    if (u - a < t2 || b - u < t2)
        if (x < m)
            d = tol;
        else
            d = -tol;
        end
    end
else
    if (x < m)
        e = b;
    else
        e = a;
    end
    e = e - x;
    d = c*e;
end

if (abs(d) >= tol)
    u = x + d;
else
    if (d > 0.0)
        u = x + tol;
    else
        u = x - tol;
    end
end
[fu]=feval(fun,u);
if (fu <= fx)
    if (u < x)
        b = x;
    else
        a = x;
    end
    v = w;
    fv = fw;
    w = x;
    fw = fx;
    x = u;
    fx = fu;
else
    if (u < x)
        a = u;
    else
        b = u;
    end
    if (fu <= fw || w == x)
        v = w;
        fv = fw;
        w = u;
        fw = fu;
    else if (fu <= fv || v == x || v == w)
            v = u;
            fv = fu;
        end
    end
```

```
        end
    end
end

I = [x]';
```

- **grafico.m**

```
function grafico(eval_fun,I1,I)

figure

plot(min(I1),feval(eval_fun , min(I1)),'k*','MarkerSize',8);

hold on;

plot(max(I1),feval(eval_fun , max(I1)),'k*','MarkerSize',8);

plot(min(I),feval(eval_fun , min(I)),'r*','MarkerSize',8);

plot(max(I),feval(eval_fun , max(I)),'r*','MarkerSize',8);

scale = axis;

if length(I) == 2
    ezplot(evalin('base','fun'),[min(I) max(I)]);
    h=get(gca,'children');
    set(h(1),'Color','red');
    set(h(1),'LineWidth',3);
end

ezplot(evalin('base','fun'),scale(1:2));

xlabel('x');
ylabel('y');
```