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 $\longrightarrow$ I = [a,b] $\longrightarrow$ FIM

$x_3 = x_2$.

$x_2 = r\,a + (1-r)\,b$ $\longrightarrow$ Retornar a D.

F - Se $f_2 \geq f_3$:

a = x_2; b = x_4; L = (b – a).

Se L < ε $\longrightarrow$ I = [a,b] $\longrightarrow$ FIM

$x_2 = x_3$.

$x_3 = (1-r)\,a + r\,b$ $\longrightarrow$ 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 - $x_{mín} = \frac{(x_2^2 - x_3^2)f_1 + (x_3^2 - x_1^2)f_2 + (x_1^2 - x_2^2)f_3}{2(x_2 - x_3)f_1 + (x_3 - x_1)f_2 + (x_1 - x_2)f_3}$;

$f_{mín} = f(x_{mín})$.

E - Se ∃ i tal $|x_{mín} - x_i| < ε$ ⟶ 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 – $x_{4-k} = x_{mín}$; $f_{4-k} = f_{mín}$ ⟶ Retornar a D.

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.

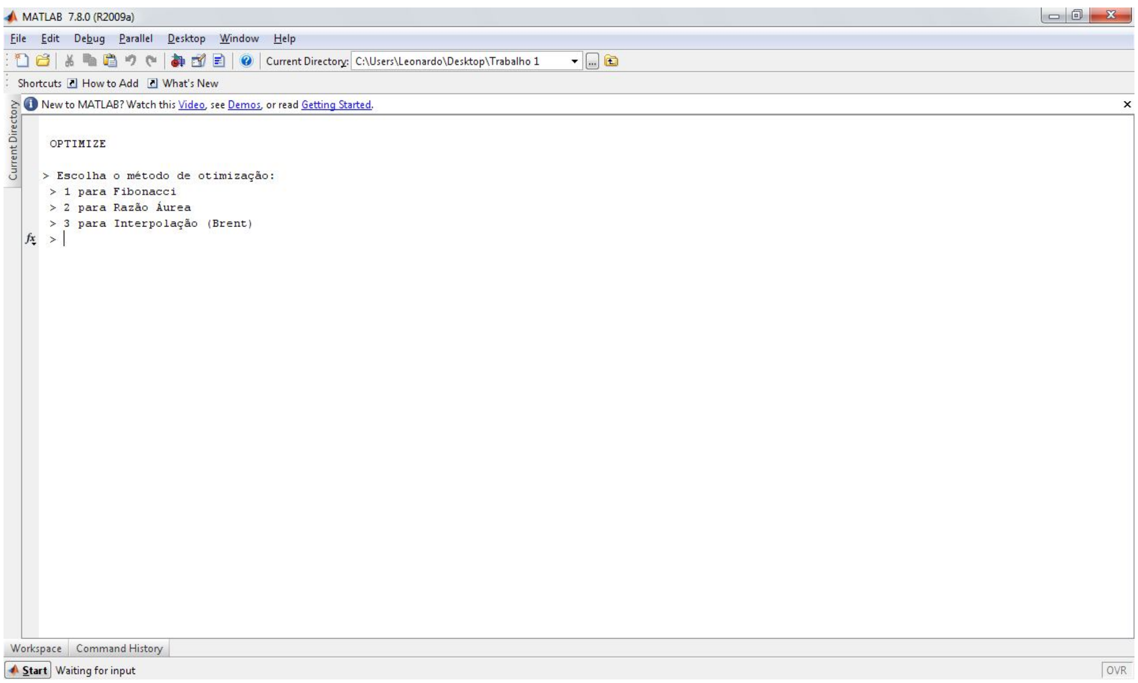

Figura 1 – Iniciando a execução da função Optimize

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 à Seção Áurea e 3 à Interpolação Polinomial.

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. A Figura 2, apresentada a seguir, exemplifica o uso do método considerando $f(x) = (x-4)^3 + 4(x-4)^2 + 1$ (definida em fun.m), intervalo de busca inicial $I_0 = [0, 10]$ e número de iterações $n = 10$.

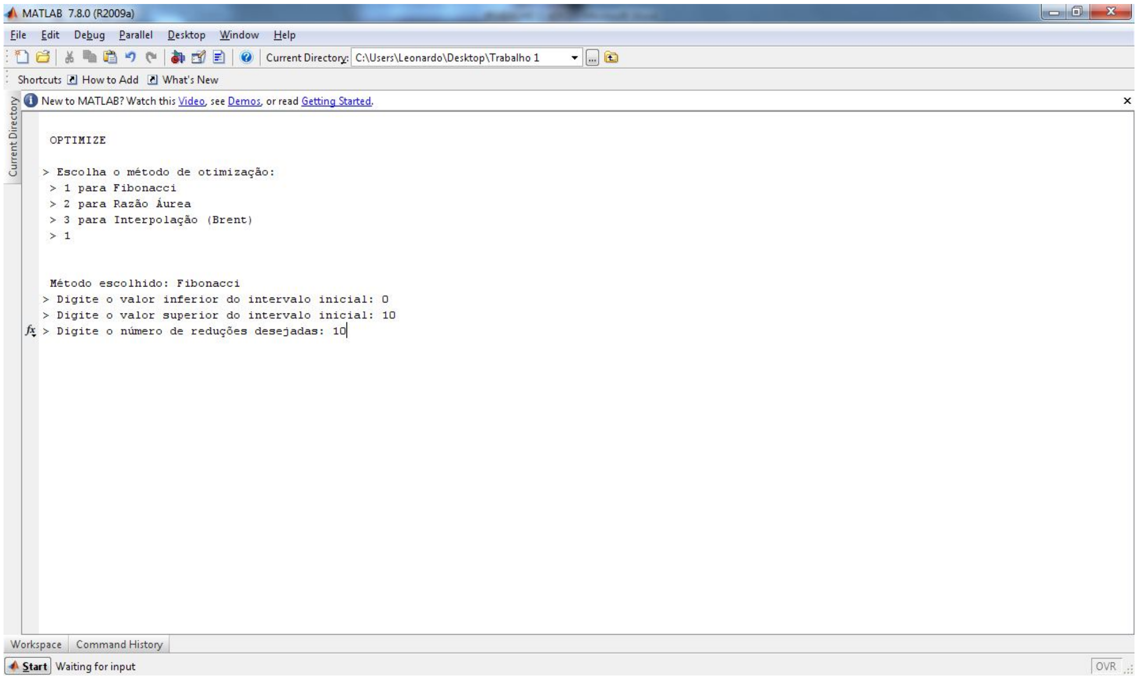

Figura 2 – Método de Fibonacci para $I_0 = [0,10]$, n = 10

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 $I_f = [3{,}9506,\ 4{,}0826]$, conforme Figura 3. A Figura 4 exibe a evolução do enquadramento de busca a cada iteração.

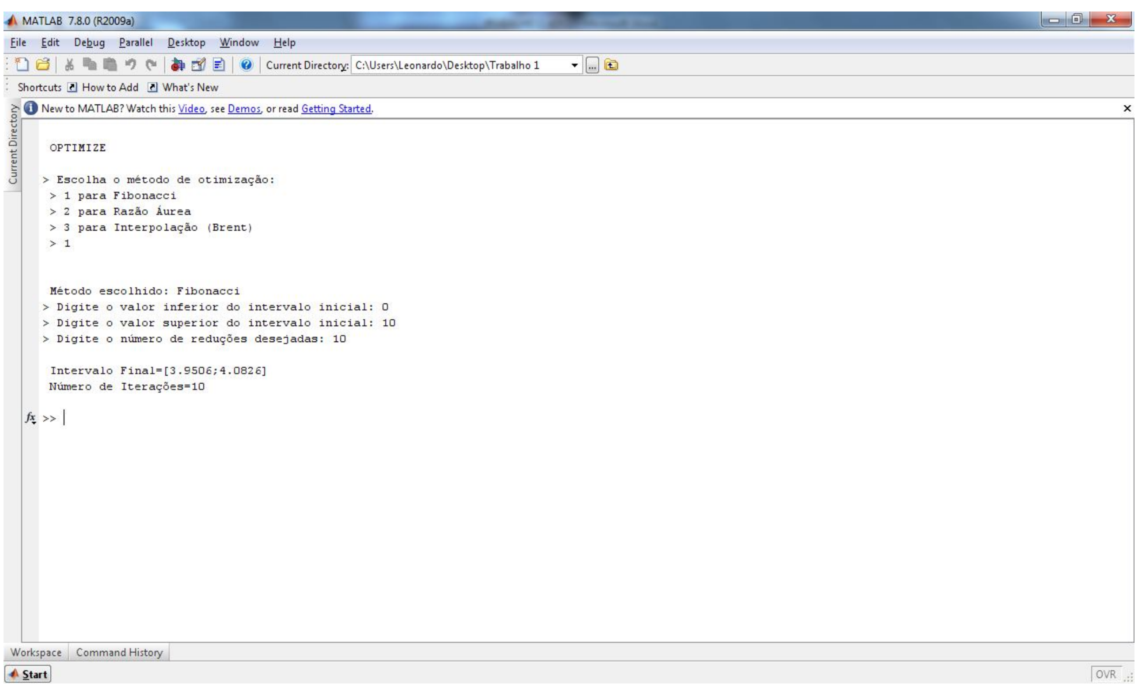

Figura 3 – Resultado do método de Fibonacci para n = 10.

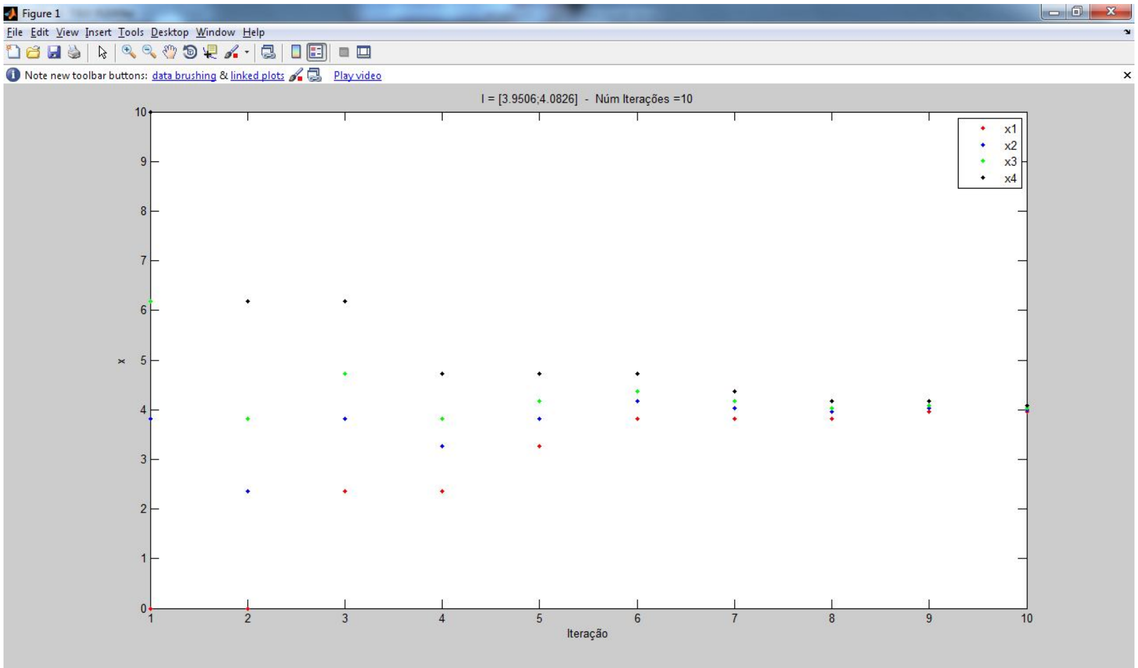

Figura 4 – Fibonacci: Evolução do enquadramento de busca para $I_0 = [0,10]$ e n = 10.

Com apenas 22 iterações, o método de Fibonacci praticamente converge para x = 4, conforme Figuras 5 e 6.

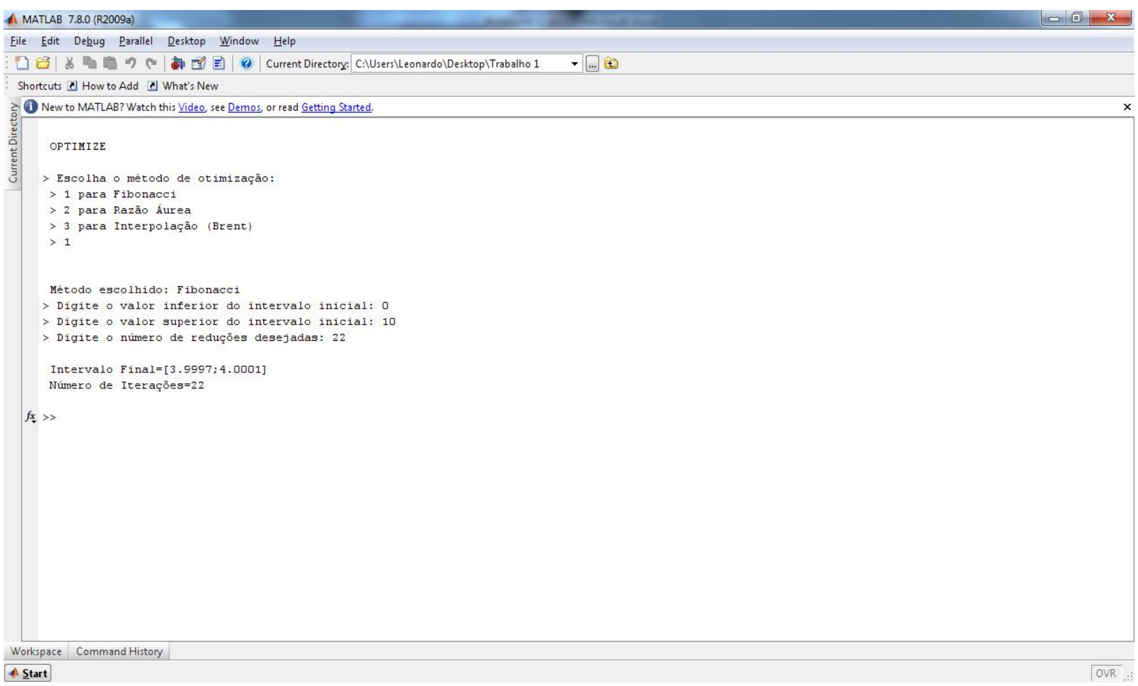

Figura 5 – Resultado do método de Fibonacci para n = 22.

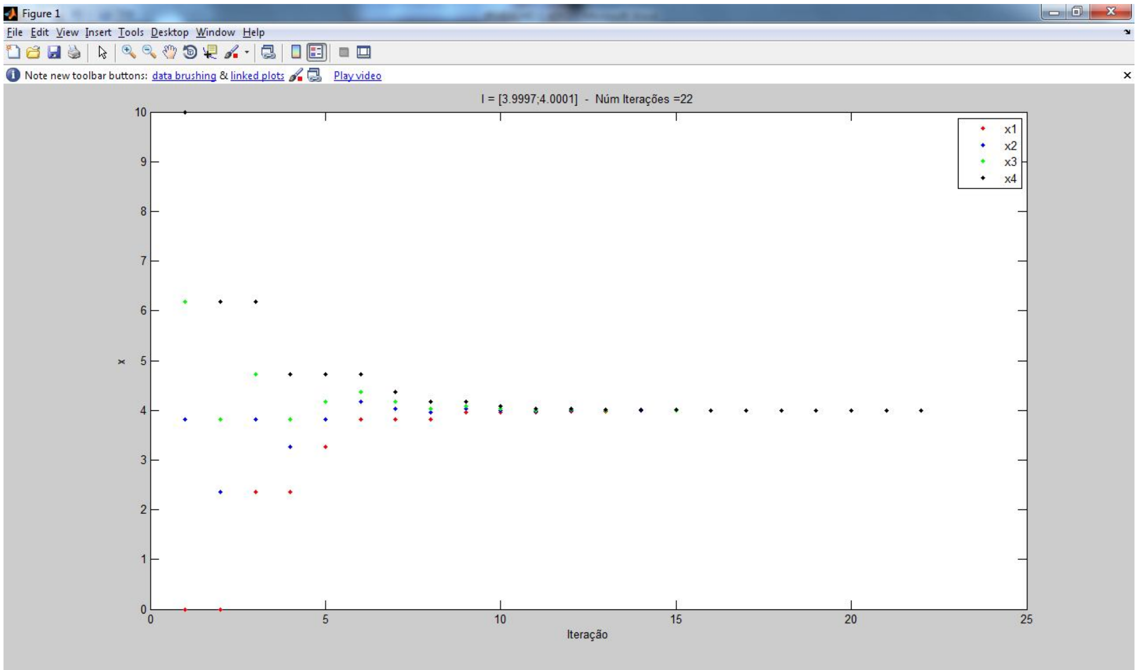

Figura 6 – Fibonacci: Evolução do enquadramento de busca para I_0 = [0,10] e n = 22.

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 = [0, 10]. Para que seja possível a comparação entre o desempenho alcançado com esse método e o de Fibonacci, será considerada a tolerância ε = 4,0826 - 3,9506 = 0,132 (resultado obtido anteriormente com 10 iterações). A Figuras 7 e 8 exibem os resultados para os valores de entrada mostrados acima.

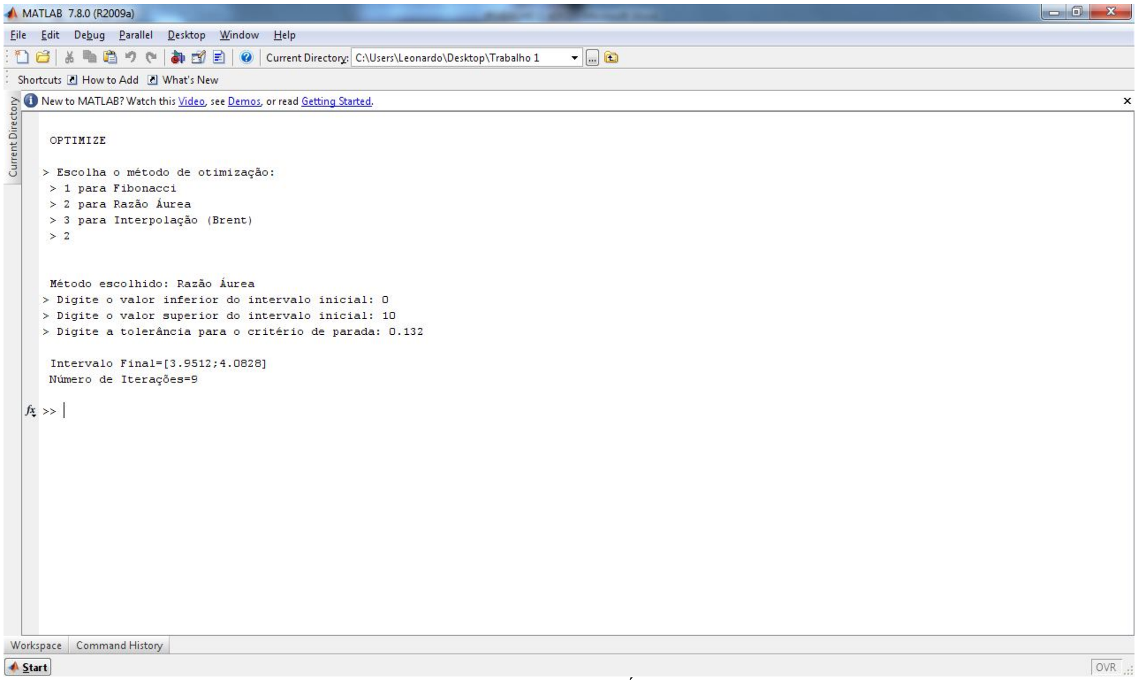

Figura 7 – Resultados para o Método da Razão Áurea para $I_0 = [0,10]$ e $\varepsilon = 0.132$.

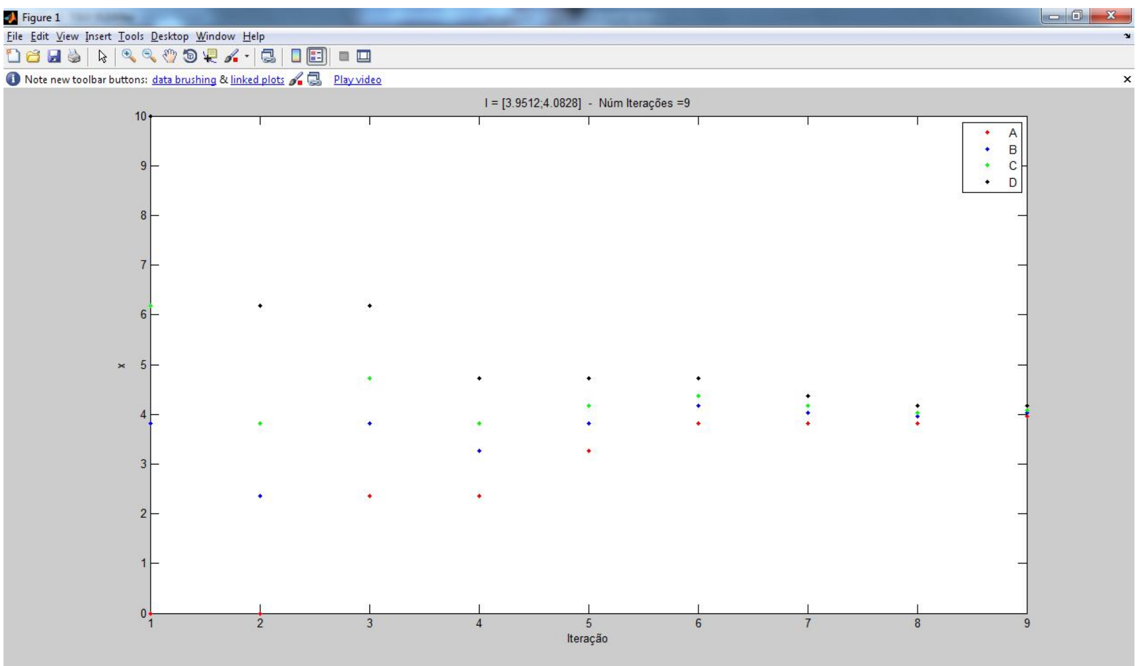

Figura 8 – Razão: Evolução do enquadramento de busca para $I_0 = [0,10]$ e $\varepsilon = 0.132$.

Nota-se, a partir das Figuras 7 e 8, que para a mesma função objetivo usada anteriormente, o método da Seção Áurea realiza apenas 9 iterações, enquanto Fibonacci necessitou de 10. O intervalo final encontrado foi $I_f = [3{,}9512,\ 4{,}0828]$, mostrando que de fato o mesmo contém o ponto de mínimo $x = 4$, calculado analiticamente.

Por último, será considerado como tolerância o intervalo calculado a partir de Fibonacci para $n = 22$ iterações. Nesse caso, $\varepsilon = 4{,}0001 - 3{,}9997 = 0{,}0004$, conforme Figuras 9 e 10.

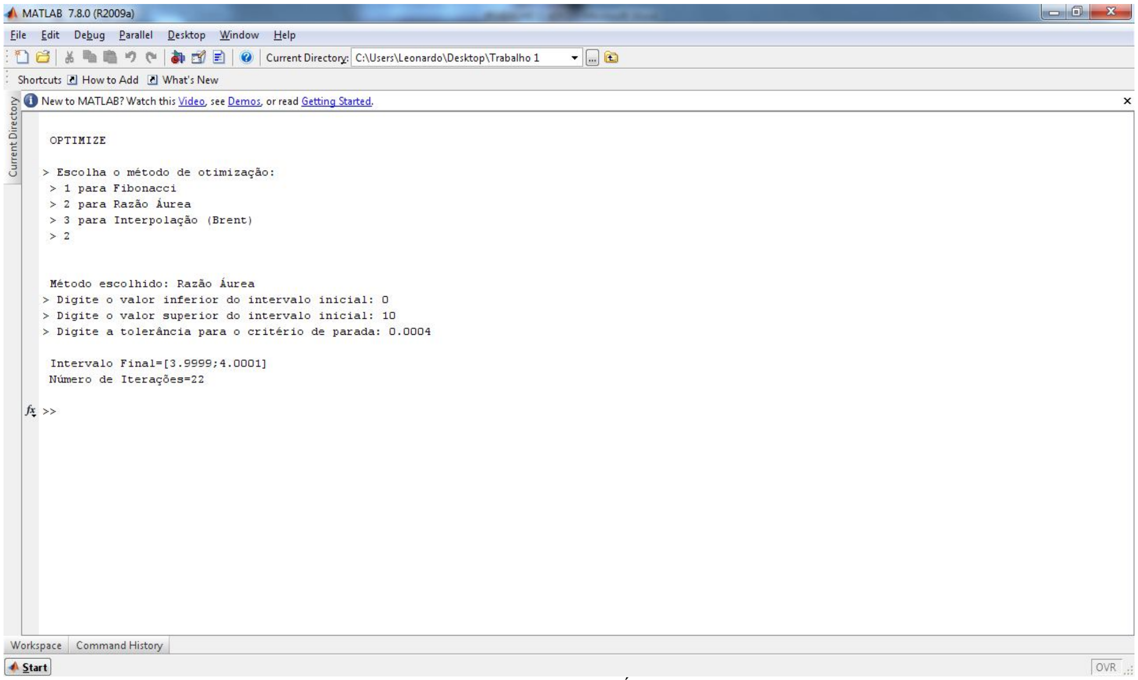

Figura 9 – Resultados para o Método da Razão Áurea para $I_0 = [0,10]$ e $\varepsilon = 0.0004$.

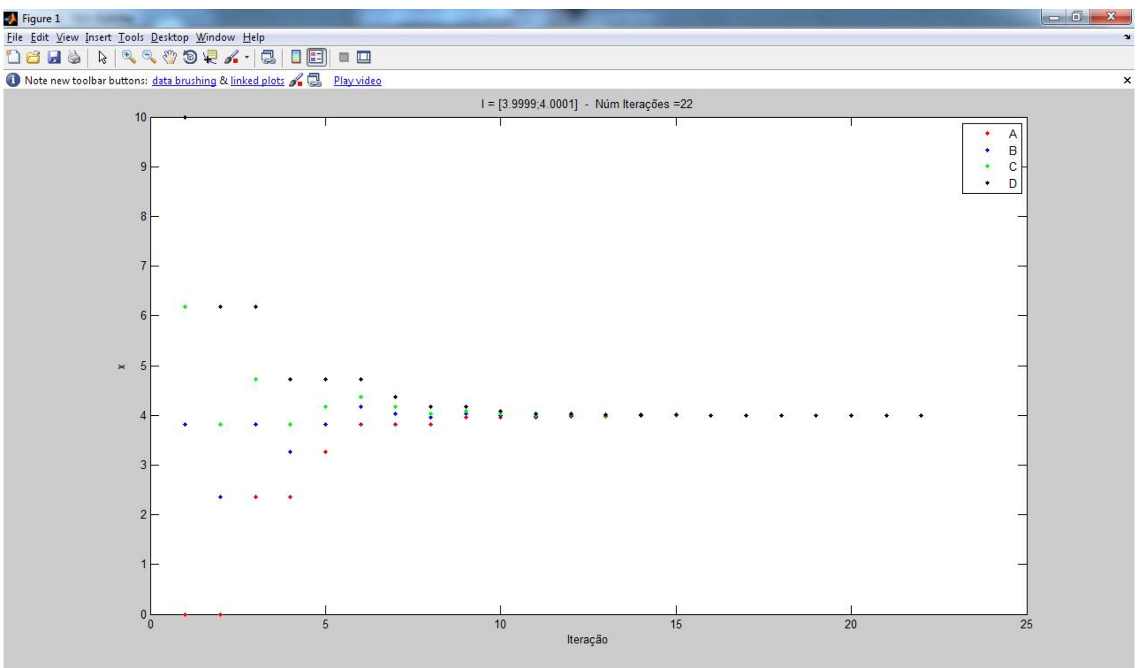

Figura 10 – RazãoÁurea: Evolução do enquadramento de busca para $I_0 = [0,10]$ e $\varepsilon = 0.0004$.

O resultado mostra que o número de iterações necessárias para atingir a tolerância especificada foi o mesmo obtido pelo algoritmo de Fibonacci.

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 $f(x) = (x\text{-}4)^3 + 4(x\text{-}4)^2 + 1$ (definida em fun.m) e intervalo de busca inicial $I_0 = [0, 10]$. Inicialmente, foi escolhida a tolerância $\varepsilon = 0{,}01$, como mostrado nas Figuras 11 e 12.

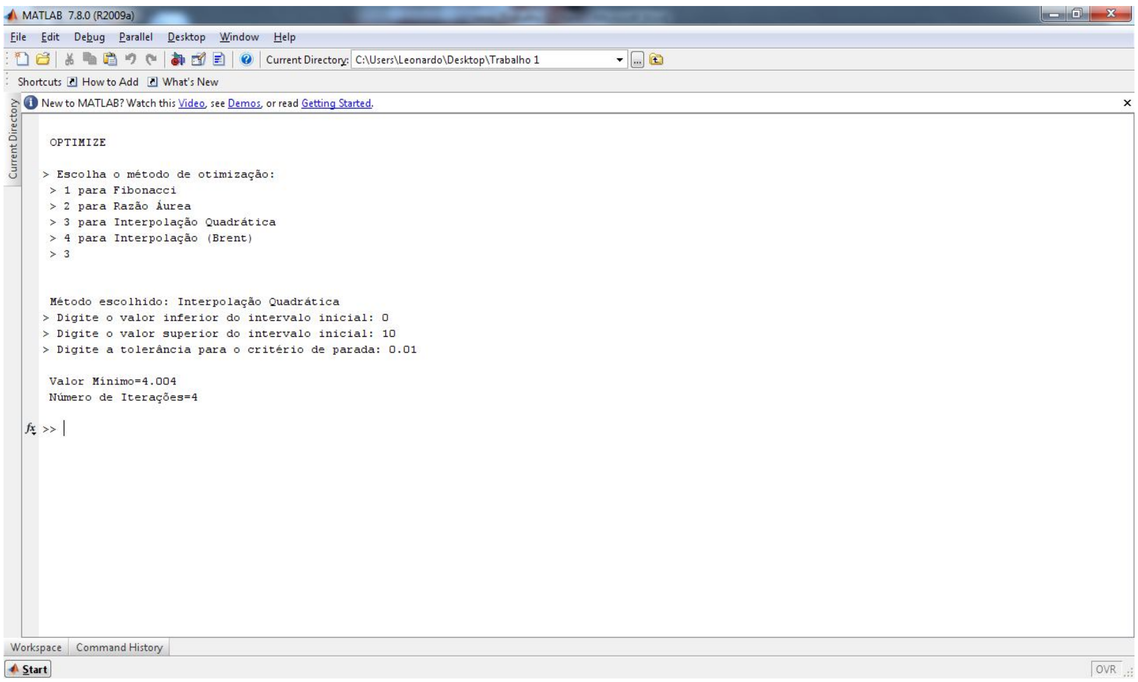

Figura 11 - Resultados para o Método da Interpolação Quadrática para $I_0 = [0,10]$ e $\varepsilon = 0.01$.

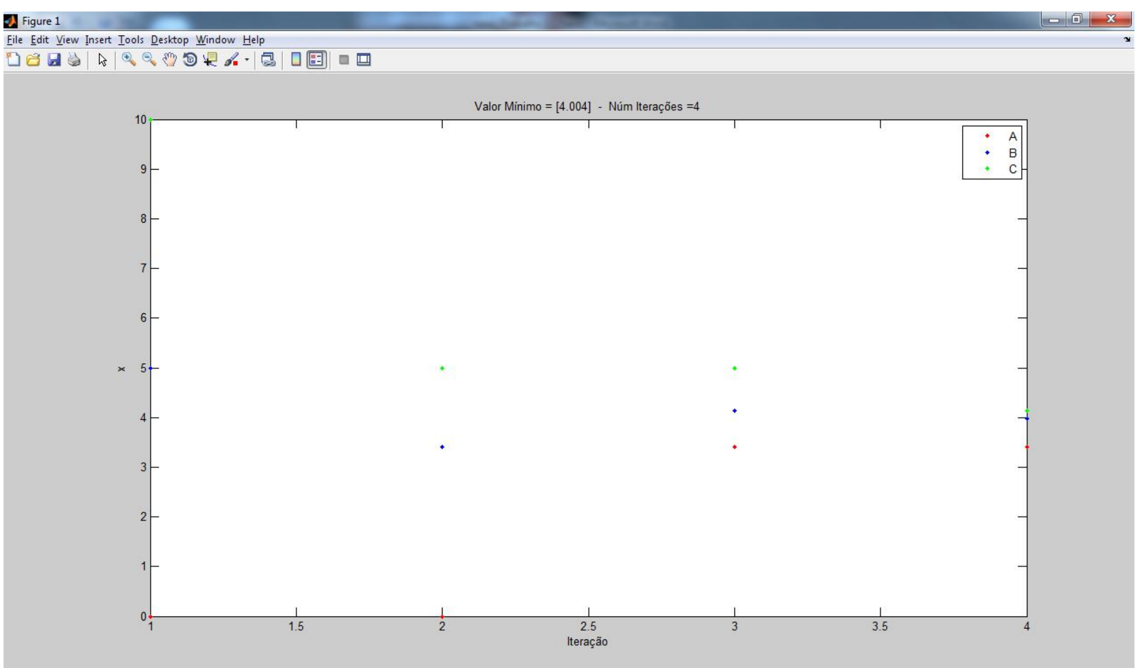

Figura 12 - Interpolação Quadrática: Evolução do enquadramento de busca para $I_0 = [0,10]$ e $\varepsilon = 0.01$.

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 4 iterações, o algoritmo acima se aproximou do mínimo analítico possuindo apenas 0,1 % de erro.

Repetindo-se a simulação, porém com tolerância ε = 0,001, chega-se aos resultados das Figuras 13 e 14.

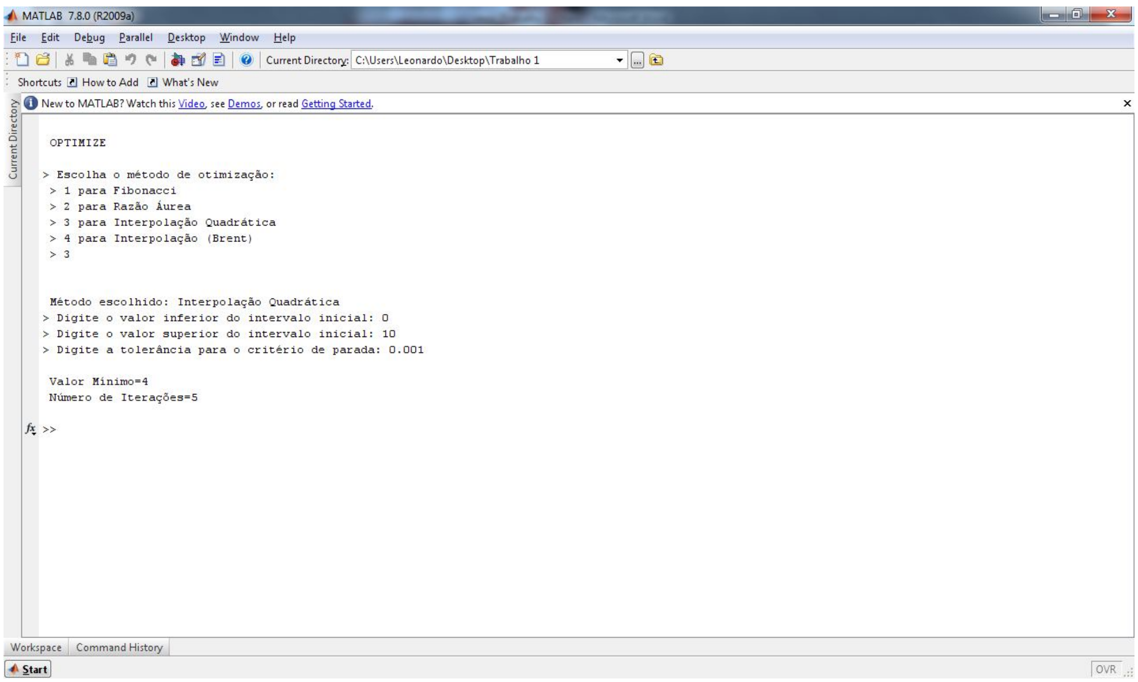

Figura 13 - Resultados para o Método da Interpolação Quadrática para $I_0 = [0,10]$ e ε = 0.001

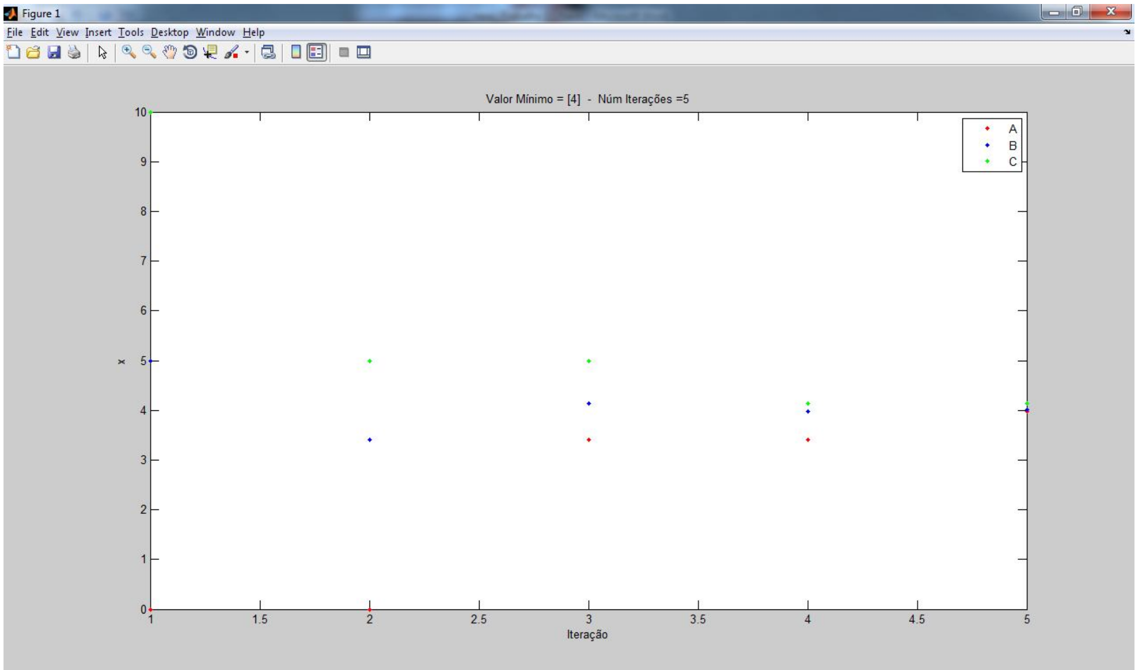

Figura 14 - Interpolação Quadrática: Evolução do enquadramento de busca para $I_0 = [0,10]$ e $\varepsilon = 0.001$.

Dessa vez, o algoritmo encontra o ponto de mínimo calculado analiticamente sem erros, porém o número de iterações permaneceu como i = 4.

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 $I_0 = [0,10]$ e $\varepsilon = 0.001$, os resultados apresentados nas Figuras 15 e 16 são alcançados.

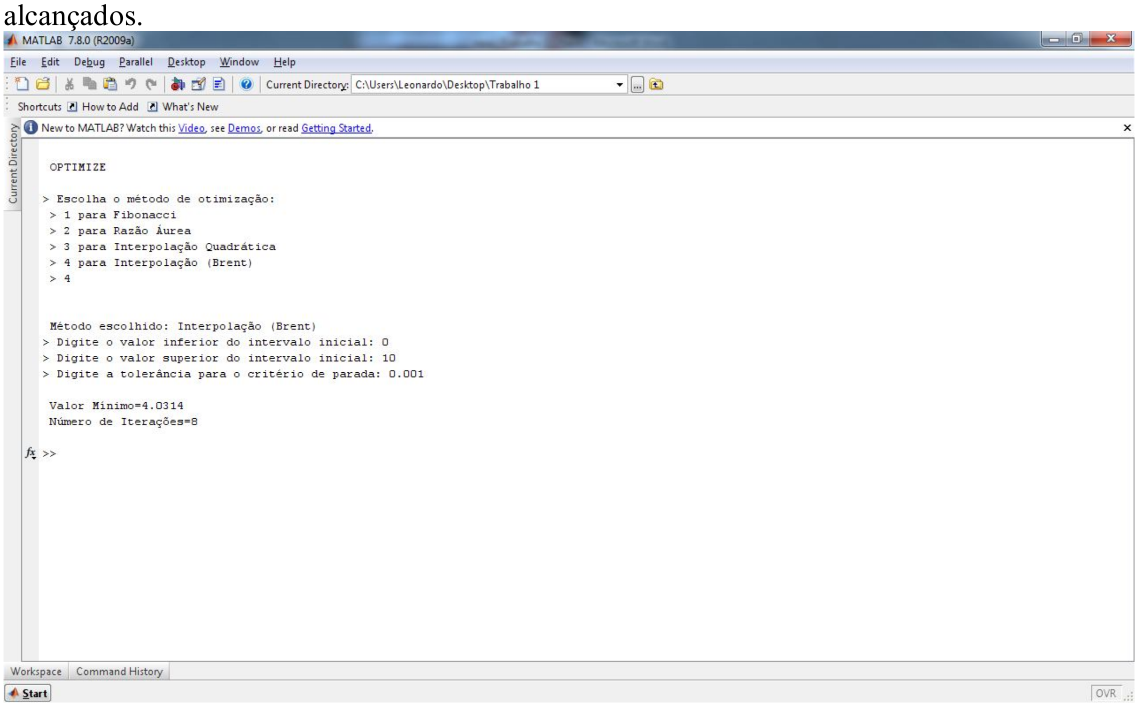

Figura 15 - Resultados para o Método de Brent para $I_0 = [0,10]$ e $\varepsilon = 0.001$

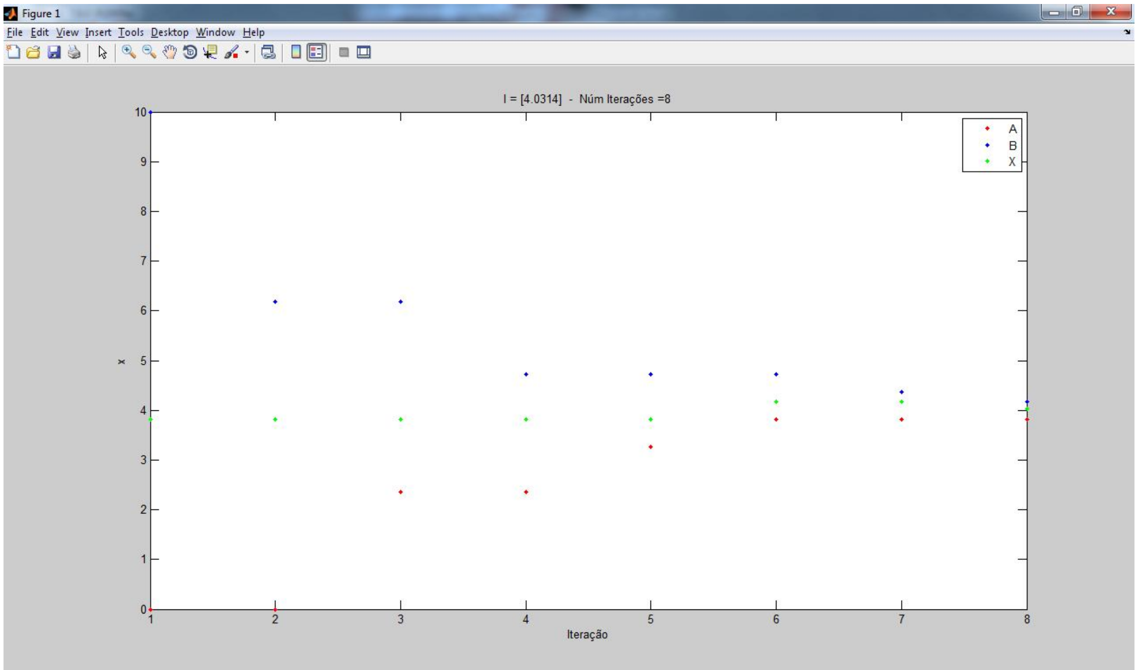

Figura 16 - Brent: Evolução do enquadramento de busca para I_0 = [0,10] e ε = 0.001.

Conforme resultados obtidos, concluiu-se que o método de Brent apresentou desempenho inferior ao obtido pelo da Interpolação Quadrática, uma vez eu foram necessárias 8 iterações para o algoritmo atender ao critério de parada especificado, e o erro obtido no cálculo foi de 0.785 %.

ANEXO I

- fun.m

```
function [f]=fun(x)

f = (x-4)^3 + 4*(x-4)^2 + 1;  %define a função objetivo f(x)
```

- optimize.m

```
function optimize

clear all;
clc;
format long;

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

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

method = input('> Escolha o método de otimização: \n > 1 para Fibonacci \n
> 2 para Razão Áurea \n > 3 para Interpolação (Brent) \n > ');

if method == 1

    fprintf(1,'\n\n Método 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(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'));

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(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'));
end

if method == 3

    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(fun,I1,eps);

    fprintf(1,strcat('\n Intervalo Final=',num2str(I)));
    fprintf(1,strcat('\n Número de Iterações=',num2str(cont_it),'\n\n'));
end
```

- 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]';
```