**MÉTODOS NUMÉRICOS PARA CALCULAR ZEROS DE FUNÇÕES**

**Evandro Pedro Alves de Mendonça**a**, Ricardo Sérgio Macêdo Tabosa Filho**a.

a *Núcleo de Tecnologia (NTI),* *Universidade Federal de Pernambuco (UFPE), Campus Acadêmico do Agreste (CAA), Rodovia BR-104, km 59, S/N, Nova Caruaru, CEP. 55.014-900, Caruaru-PE, Brasil,* [*http://www.ufpe.br/caa*](http://www.ufpe.br/caa)

**Palavras Chave:** raízes de funções, zeros de funções, métodos numéricos, convergência, gráficos, aproximações.

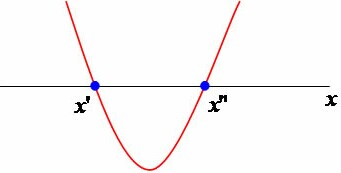
1. **INTRODUÇÃO**

Em várias áreas das ciências exatas encontra-se constantemente situações onde é necessário resolver um problema com a seguinte forma:

F(x) = 0 (1)

Algumas vezes, a resolução de um problema desse tipo pode ser feita de forma analítica, como é o caso das equações polinomiais de 1º e 2º grau. Porém, ao tratar-se de equações polinômiais de 3º grau em diante, ou quando trabalha-se com funções mais complexas, como seno, cosseno, a função logarítmica, entre outras, torna-se mais complicada a resolução do problema.

Graficamente, a resolução do problema representado na equação (1) acima é ponto onde a função f(x) intercepta o eixo das abcissas, ou eixo x. Esses pontos são as chamadas raízes da função. Isso é mostrado na imagem abaixo, onde x' e x'' são as raízes da função.



Como já dito, em alguns casos, o cálculo analítico das raízes de uma função é inviável. Uma saída para essa problema é a ultilização de métodos numéricos para a determinação das raízes. Inicialmente, procura-se dar uma estimativa para a raiz da função. Em seguida, é aplicado um método para aperfeiçoar essa estimativa inicial. É importante observar que, nos métodosnuméricos, nem sempre é possível fazer uma estimativa exata para a raiz, diferente do cálculo analítico. Muitas vezes apenas será possível realizar uma boa estimativa da raiz.

Os métodos de estimativa de raízes dividem-se em dois grupos, são eles: os métodos de confinamento, e os métodos abertos. Os métodos de confinamento consistem em ser obtido um intervalo inicial em que a raiz da função se encontra dentro daquele intervalo. Os métodos abertos consistem em dar-se uma solução inicial para a raiz, e em seguida serem feitas manipulações numéricas para o cálculo da raíz. No método aberto, essa estimativa inicial deve ser próxima da raiz exata.

Independente do método ultilizado, é necessário estabelecer uma tolâmcia no cálculo da raiz, visto que em alguns casos, o cálculo exato é praticamente impossível, além de que, o custo computacional é extremamente alto. Em termos simples, a tolerância seria o quanto a solução encontrada pode desviar-se da solução exata. Obviamente, quanto maior a tolência permitida, menor será o erro associado. Porém, uma boa escolha do método a ser utilizado leva a um menor custo computacional, e a um menor erro associado. Alguns métodos são listados a seguir:

* Bisseção - método de confinamento
* Regula falsi - método de confinamento
* Ponto fixo - método aberto
* Newton-Raphson - método aberto
* Secante - Método aberto

A escolha do método a ser utilizado depende do problema a ser resolvido e da precisão e exatidão necessárias. Além disso, existem casos específicos onde alguns desses métodos não funcionam, ou se distanciam muito da solução exata. Uma análise do procedimento usado por cada método leva a uma boa escolha de qual deve ser utilizado. O detalhamento de cada método é feito nos exercícios propostos deste trabalho.

1. **EXERCÍCIOS PROPOSTOS**

**1º questão:**

%variáveis

x = 0:0.5:10; %intervalos de 0,5 de comprimento

y = 5\*sin(x.^2)-exp(x/10); %f(x)

intervalos = []; %matriz de intervalos que contêm raízes

yintervalos = [];%matriz que contém a imagem dos intervalos

cont = 0; %contador

raizes = []; %vetor de raizes

imagensdasraizes = []; %vetor de imagens das raizes

xmed = []; %ponto médio de um intervalo

ymed = []; %imagem do ponto médio

%plotagem do gráfico da função

plot(x,y);

grid on;

%esse for captura os três primeiros intervalos que contêm raízes de f(x)

for i = 1:(length(x)-1)

if(y(i)\*y(i+1)<0)%condição para ter uma raíz entre x(i) e x(i+1)

intervalos = [intervalos;x(i) x(i+1)];%insere intervalo na matriz

cont = cont + 1;%incrementa contador

end

if(cont==3)%foram contados 3 intervalos com raíz

break %sai do laço de repetição

end

end

%a próxima linha calcula as imagens nos intervalos obtidos

yintervalos = 5\*sin(intervalos.^2)-exp(intervalos/10);

%método da bisseção para aproximação das raízes nos três intervalos obtidos

for i=1:3

%calculando o ponto médio

xmed = (intervalos(i,1)+intervalos(i,2))/2;

ymed = 5\*sin(xmed.^2)-exp(xmed/10);

cont = 0; %zerando contador de iterações

%calculando raíz levando em conta a tolerância 10^(-5) e um número

%máximo de iterações de 10000 (para o caso de ficar preso no laço)

while((abs(ymed) > 0.00001)&&(cont < 10000))

if (yintervalos(i,1)\*ymed<0) %raíz está entre f(a) e f(xmed)

%inserindo ponto médio no fim do intervalo

intervalos(i,2) = xmed;

yintervalos(i,2) = ymed;

else %raíz está entre f(xmed) e f(b)

%inserindo ponto médio no início do intervalo

intervalos(i,1) = xmed;

yintervalos(i,1) = ymed;

end

%calculando novo ponto médio

xmed = (intervalos(i,1)+intervalos(i,2))/2;

ymed = 5\*sin(xmed.^2)-exp(xmed/10);

cont = cont + 1; %incrementa contador de iterações

if(cont==10000) %número máximo de iterações atingido

fprintf('Raíz número %d atingiu o número máximo de iterações.\n',i);

end

end

raizes = [raizes;xmed]; %insere raíz encontrada no vetor de raízes

imagensdasraizes = [imagensdasraizes;ymed]; %o mesmo para a imagem

%informa o número de iterações do laço de repetição

fprintf('%d iterações para encontrar a raíz %d.\n',cont,i);

end

%Apresentação dos resultados

format LONGE;

fprintf('\n');

disp('Raízes:');

disp(raizes);

disp('Imagens:');

disp(imagensdasraizes);

**2º questão:**

%pontos do intervalo inicial

x1 = 0.5;

x2 = 3;

%calculando imagens iniciais

y1 = -4+8\*x1-5\*(x1.^2)+(x1.^3);

y2 = -4+8\*x2-5\*(x2.^2)+(x2.^3);

%calculando o ponto médio

xmed = (x1+x2)/2;

ymed = -4+8\*xmed-5\*(xmed.^2)+(xmed.^3);

%zerando contador de iterações

cont = 0;

%calculando raíz levando em conta a tolerância 10^(-5) e um número

%máximo de iterações de 10000 (para o caso de ficar preso no laço)

while((abs(ymed) > 1e-5)&&(cont < 10000))

if (y1\*ymed<0) %raíz está entre f(a) e f(xmed)

%inserindo ponto médio no fim do intervalo

x2 = xmed;

y2 = ymed;

else %raíz está entre f(xmed) e f(b)

%inserindo ponto médio no início do intervalo

x1 = xmed;

y1 = ymed;

end

%calculando novo ponto médio

xmed = (x1+x2)/2;

ymed = -4+8\*xmed-5\*(xmed.^2)+(xmed.^3);

cont = cont + 1; %incrementa contador de iterações

if(cont==10000) %número máximo de iterações atingido

disp('O método da bisseção atingiu o número máximo de iterações.');

end

end

%Apresentação dos resultados

%informa o número de iterações do laço de repetição

fprintf('%d iterações para encontrar a raíz.\n\n',cont);

%informa a imagem da raíz encontrada

fprintf('Imagem: %d\n\n',ymed);

%informa a raíz encontrada

fprintf('O método da bisseção converge para a raíz: %d.\n',xmed);

**3º questão:**

format LONGE;

tolerancia = 1e-10;

for x0 = -3:0.2:3 %soluções iniciais

xk = (2\*(x0^2)+2)/5; %função de aproximação

cont = 1; %contador de iterações

parar = 0; %variável booleana

while((parar==0)&&(cont<1000))

xk1 = (2\*(xk^2)+2)/5; %Calculando aproximação atual (depende da anterior)

if(abs(xk1-xk)/abs(xk1)<tolerancia) %critério de parada

parar = 1;

else %raíz ainda está fora da tolerância

cont = cont + 1;

xk = xk1; %atualizando valor da aproximação anterior

end

if(cont==1000)

disp('Número máximo de iterações atingido.');

end

end

%exibindo resultados

fprintf('Para x0 = %.2f,\n',x0);

fprintf('Raíz converge para: %f (%d iterações)\n\n',xk1,cont);

end

%O mesmo procedimento é realizado com a outra função de aproximação

%Ou seja, onde aparece "(2\*(x^2)+2)/5", foi utilizado "sqrt(((5\*x)/2)-1)".

%Abaixo, plotagem dos gráficos para discussão

x = -10:0.1:10;

y = (2\*(x.^2)+2)/5; %primeira função de aproximação

y1= -10:0.1:10; %função identidade

y2= sqrt(((5\*x)/2)-1); %segunda função de aproximação

y = [y;y1;y2];

plot(x,y)

grid on

**4º questão:**

%Código para a letra a)

tic

x = input('Digite o valor inicial: '); %chute inicial para o cálculo da raiz

y = ((x-2)^2) - log(x); %Calcula o valor da função no ponto/chute inicial

y1 = 2\*(x-2) - 1/x; %Calcula a taxa de variação da função, ou seja, a derivada, no ponto inicial.

raiz = x - y/y1; %Calcula o ponto em que a reta tangente cruza o eixo x

i = 0; %Contador para número de iterações

while(abs(raiz - x)>1/100000) %condição para que existam 3 dígitos significativos

x = raiz; %X é atualizado para o valor onde a reta tangente cruza o eixo x

y = ((x-2)^2) - log(x); % A função é atualizada com o novo ponto x

y1=2\*(x-2) - 1/x; %A derivada é atualizada com o novo ponto de x

raiz = x - y/y1;

% É encontrado o ponto em que a nova reta tangente ao valor da função

% em relação ao ponto x cruza o eixo x.

i = i+1;

end

% Ao terminar o laço, raiz vem com o valor dentro da tolerância aceitável

fprintf('Raiz: %f - %d interações\n\n', raiz, i);

%Plotagem do gráfico da função

x = 0:0.5:6;

y = f(x);

plot(x,y);

grid on

toc

%Nas letras b) e c) dessa mesma questão, o código é o mesmo. O que muda é a função

%y, fornecida no enunciado, e sua derivada y1.

**5º questão:**

%Código para a letra a)

tic

a = x1; %Ponto inicial do intervalo

b = x2; %Ponto final do intervalo

ya = ((a-2)^2) - log(a); % Valor da função no ponto a

yb = ((b-2)^2) - log(b); % Valor da função no ponto b

inc = ya - yb/a-b; % Inclinação da reta secante

raiz = a - ya/inc; % Estimativa da raiz/ Equação da reta secante quando y = 0

i = 0; % contador para número de interações

while(abs(raiz-a)>1/100000)

a = raiz; % atualização do intervalo

ya = ((a-2)^2) - log(a); %cálculo de um ponto em que a secante passa

yb = ((b-2)^2) - log(b); %cálculo do outro ponto em que a secante passa

inc = ya - yb/a-b; % cálculo da inclinação da secante

raiz = a - ya/inc; % raiz aproximada

i = i+1; %incremento do contador

end

%Apresentação dos resultados

fprintf('No intervalo de 1 até 2:\n');

fprintf('Raiz : %f - %d iterações\n\n', raiz, i);

%Plotagem do gráfico da função

x = 0:0.5:6;

y = ((x-2).^2) - log(x);

plot(x,y);

grid on

toc

%Nas letras b) e c) dessa mesma questão, o código é o mesmo. O que muda são as funções

%ya e yb, que são fornecidas no enunciado.

**6º questão:**

**A)**

tic

x = []; %chute inicial para o cálculo da raiz

y = log(x^2 + 1) - (exp(1)^(0.4\*x))\*(cos(pi\*x));

%Calcula o valor da função no ponto/chute inicial

y1 = ((2\*x)/((x^2)+1)) - (0.4\*(exp(1)^0.4\*x)\*cos(pi\*x)) + ((exp(1)^(0.4\*x))\*sin(pi\*x)\*pi);

%Calcula a taxa de variação da função, ou seja, a derivada, no ponto inicial.

raiz = x - y/y1; %Calcula o ponto em que a reta tangente cruza o eixo x

i = 0; %Contador para número de iterações

while(abs(raiz - x)>1/100000) %condição para que existam 3 dígitos significativos

x = raiz; %X é atualizado para o valor onde a reta tangente cruza o eixo x

y = log(x^2 + 1) - (exp(1)^(0.4\*x))\*(cos(pi\*x)); % A função é atualizada com o no ponto x

y1 = ((2\*x)/((x^2)+1)) - (0.4\*(exp(1)^0.4\*x)\*cos(pi\*x)) + ((exp(1)^(0.4\*x))\*sin(pi\*x)\*pi);

%A derivada é atualizada com o novo ponto de x

raiz = x - y/y1;

% É encontrado o ponto em que a nova reta tangente ao valor da função

% em relação ao ponto x cruza o eixo x.

i = i+1;

end

% Ao terminar o laço, raiz vem com o valor dentro da tolerância aceitável

fprintf('Raiz: %f - %d interações\n\n', raiz, i);

%Plotagem do gráfico da função

x = -3:0.25:0;

y = log(x.^2 + 1) - (exp(1).^(0.4.\*x)).\*(cos(pi.\*x));

plot(x,y);

grid on

toc

**B)**

tic

% ultilização da função fezero

y = fzero('log(x^2 + 1) - (exp(1)^(0.4\*x))\*(cos(pi\*x))', -1);

%Apresentação do resultado

fprintf('\n\nA raiz é: %f\n', y);

toc

1. **DISCUSSÃO DOS RESULTADOS**

**1º questão:** Usando o método da bisseção, foi possível encontrar as três primeiras raízes positivas da função em um tempo de aproximadamente 0.010959 segundos. As raízes encontradas, com precisão de 10-5 foram:

1º raiz é: 0.459305 - 14 iterações

2º raiz é: 1.703574 - 18 iterações

3º raiz é: 2.558212 - 14 iterações

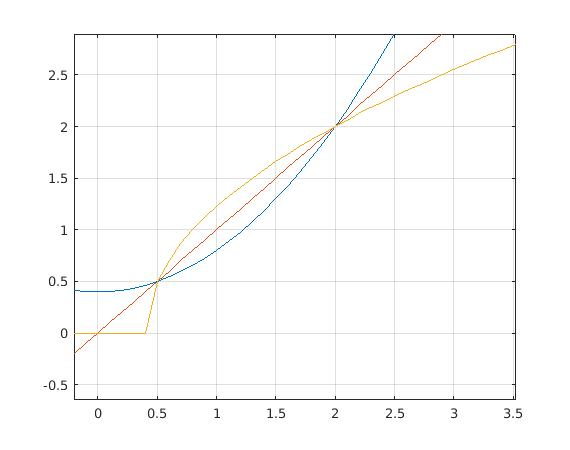
**2º questão:**

**A)** Usando o método da bisseção para a função -4 +8x – 5x2-x3, no intervalo de 0,5 até 3, as iterações convergem para a raiz:

0.999992 - 15 interações Tempo de execução: 0.155697 segundos.

**B)** Não é possível ultilizar o método da bisseção na raiz x=2. Isso se dá porque nesse ponto a função não cruza o eixo x, apenas o toca. No método da bisseção, quando em ambos os extremos do intervalo a função é positiva, ou é negativa, o método interpreta que ali não há raízes. Portanto, neste caso, o método da bisseção falha.

**3º questão:** De acordo com o gráfico abaixo, percebe-se que a cada iteração, usando a primeira função (linha azul), a sequência gerada por ela converge para a raíz "0,5", enquanto que a segunda função (linha amarela), converge para a raíz "2". Isso se deve ao fato de que a raíz de cada iteração é calculada com base numa imagem calculada anteriormente com a mesma função, a partir de um ponto dado inicialmente, desde que ese ponto esteja no intervalo de convergência. A cada iteração, a nova imagem se aproxima mais da raíz da função. Graficamente, é possível perceber que a cada iteração, a projeção horizontal da imagem calculada na função identidade (linha vermelha) revela uma nova imagem a partir de uma projeção vertical nesse ponto, e, assim, as projeções posteriores vão se aproximando cada vez mais da interseção da função de aproximação com a função identidade. Essa interseção é a raíz que se deseja calcular. Para a primeira função, as projeções tendem à interseção da esquerda do gráfico e, para a segunda função, tendem à da direita, isto é, "0,5" e "2", respectivamente. Portanto, para encontrar a raíz "0,5", faz-se uso da função 1: [(2\*(x0^2)+2)/5]; já para a raíz "2", faz-se uso da função 2: [sqrt(((5\*x)/2)-1)].



Tempo de execução para encontrar a raíz com todas as soluções iniciais fornecidas:  
Função 1: 0.734s  
Função 2: 0.794s

**4º questão:** Em todos os casos de aplicação do método de Newton-Raphson, quando a derivada de primeira e segunda ordem nos intervalos com raízes são contínuas, então a ordem de convergência do método é igual a 2. Isso se aplica a todos os problemas em que foi usado o método Newton-Raphson, e estes estão descritos abaixo.

**A)** Usando o método de Newton-Raphson na função (x-2)2-ln(x), nos intervalos de 1 a 2, e de *e* a 4, consirando 1 e 4 como pontos iniciais, obtem-se as seguintes raízes, respectivamente:

Raiz: 1.412391 - 3 iterações tempo: 1.147785 segundos

Raiz: 3.057104 - 4 iterações tempo: 1.219878 segundos

**B)**Usando o método de Newton-Raphson na função *e*x-3x2, nos intervalos de 0 a 1, e de 3 a 5, considerando 0 e 3 como pontos iniciais, obtem-se as seguintes raízes, respectivamente:

Raiz: -0.458962 - 5 iterações tempo: 1.837445 segundos

Raiz: 3.733079 - 8 iterações tempo: 1.204661 segundos

**C)** Usando o método de Newton-Raphson na função *e*x+2-x +2cos(x) - 6, no intervalo de 1 a 2, considerando 1 como ponto inicial, obtem-se a seguinte raíz:

Raiz: 1.829384 - 10 iterações Tempo: 0.997889 segundos

**5º questão:** Como as funções da questão 4 são as mesmas da questão 5, os gráficos também são os mesmos. Foi observado que o método da secante necessita de um número maior de iterações em comparação com o método de Newton. Em compensação, o método de Newton precisa de mais tempo para chegar a à raiz desejada quando comparado com o método da secante.

**A)** Usando o método da secante na função (x-2)2-ln(x), nos intervalos de 1 a 2, e de *e* a 4, obtem-se as seguintes raízes, respectivamente:

Raiz : 1.412390 - 11 iterações

Raiz : 3.057107 - 15 iterações Tempo: 0.061873 segundos

**B)** Usando o método da Secante na função *e*x-3x2, nos intervalos de 0 a 1, e de 3 a 5, obtem-se as seguintes raízes, respectivamente:

Raiz : 0.910008 - 8 iterações

Raiz : 3.733083 - 55 iterações Tempo: 0.069742 segundos

**C)** Usando o método da Secante na função *e*x+2-x +2cos(x) - 6, no intervalo de 1 a 2, obtem-se a seguinte raíz:

Raiz : 1.829385 - 5 iterações Tempo: 0.061056 segundos

**6º questão:**

**A)** Dentre os métodos citados nos problemas anteriores, sabe-se que os dois métodos que convergem para o resultado mais rápido são o método da secante e o método de Newton. Porém, como abordado nos exercícios 4 e 5, o método de Newton precisa de menos iterações para alcançar a tolerância desejada, mesmo levando mais tempo. Isso significa um custo computacional menor, logo, é mais viável usar o método de Newton. Porém, é importante levar em conta que a derivada da função já foi fornecida no programa, caso contrário, o método de newton seria mais desvantajoso. Usando o método de Newton para a função ln(x2+1) - e0.4xcos(pi\*x), considerando o ponto inicial igual a –1,2,5, obtem-se as seguintes raízes negativas, respectivamente:

Raiz: -0.434143 - 5 iterações Tempo: 0.099332 segundos

Raiz: 0.450657 - 3 iterações Tempo: 3.017734 segundos

Raiz: 1.744738 - 5 iterações Tempo: 2.500891 segundos

**B)** A função fzero do MATLAB usa o método da bisseção e é ultilizada para funções não lineares. Aplicando-a para a função ln(x2+1) - e0.4xcos(pi\*x), obtem-se as seguintes raízes, com chute inicial igual a –1,1,2, respectivamente:

Raiz : -0.434143 Tempo: 0.013530 segundos

Raiz: 0.450657 Tempo: 0.023412 segundos

Raiz: 1.744738 Tempo: 0.007416 segundos

**C)** Apesar de obter as mesmas raízes, a função já implementada no MATLAB é indiscutivelmente mais eficiente, com tempo de execução muito menor que o que foi implementado nessa questão.

1. **CONCLUSÃO**

Após implementação dos códigos propostos e análise dos resultados obtidos, conclui-se que há uma variedade considerável de métodos numéricos disponíveis para se chegar ao mesmo resultado: calcular raízes de funções. Porém, para cada caso, haverá sempre um método melhor que outro, e as situações variam muito, pois isso depende do tipo de função que está sendo usada. É importante estar ciente de questões como tempo de processamento, número de iterações, convergência e precisão. Todos esses parâmetros podem ser ajustados conforme especificações do projetos com o qual se está trabalhando.