Universidade do Estado do Rio de Janeiro
IPRJ

Relatório 1: Métodos numéricos da bisseção e de Newton-Raphson para cálculos de raízes reais de funções f(x) = 0

Gustavo de Souza Curty
Matheus Marendino
Matheus Iack

Nova Friburgo
2023

Sumário

Nova Friburgo
2023

1. Introdução

A busca por soluções numéricas de equações não lineares é uma tarefa fundamental em diversas áreas da matemática aplicada e ciências computacionais. Dentre os métodos mais utilizados para a aproximação de raízes de funções, utilizamos o Método da Bisseção e o Método de Newton-Raphson.

O Método da Bisseção é um método iterativo que se baseia no Teorema do Valor Intermediário, dividindo repetidamente um intervalo inicial em subintervalos menores até que a raiz seja encontrada com a precisão desejada. Este método é notável por sua simplicidade e convergência garantida, embora possa exigir um número maior de iterações em comparação com outros métodos.

Por outro lado, o Método de Newton-Raphson é um método de aproximação que utiliza derivadas para convergir rapidamente para a raiz. Este método explora a tangente à curva da função no ponto atual da iteração, proporcionando uma convergência mais rápida em comparação com o Método da Bisseção. No entanto, é importante notar que o sucesso do Método de Newton-Raphson depende da escolha adequada do ponto inicial e da continuidade da derivada da função.

2. Metodologia

2.1 Teorema de bolzano

O primeiro passo do algoritmo é utilizar o teorema de Bolzano para encontrar o intervalo em que a raiz da $f(x) = 0$ está.
Este teorema estabelece as condições necessárias para a existência de uma raiz em um intervalo fechado, fornecendo uma base sólida para a escolha do intervalo inicial nos métodos de aproximação.

Teorema 1 (Teorema de Bolzano): Seja f : R → R uma função contínua num intervalo fechado [a,b]. $Se\, f(a) * f(b) < 0$, então f tem pelo menos um zero (raiz) no intervalo aberto (a,b).

Teorema 2: Sob as hipóteses do teorema 1, se a derivada de f(x) existir e preservar o sinal no intervalo aberto (a,b), então f tem um único zero em (a,b). A condição de f(x) existir e preservar o sinal no intervalo aberto (a,b), significa:

$$f'(x) = \begin{cases} > 0, \forall\, x \in (a,b) \rightarrow f(x) \text{ crescente,} \\ < 0, \forall\, x \in (a,b) \rightarrow f(x) \text{ descrescente.} \end{cases}$$

2.2 Uso de gráfico para encontrar o intervalo que contém a raíz de f(x) = 0

A representação gráfica de funções desempenha um papel crucial na compreensão na visualização de conceitos matemáticos, particularmente na busca de raízes de equações. Ao visualizar o gráfico da função, é possível identificar intervalos iniciais que contenham raízes, baseando-se na mudança de sinal ou em padrões de comportamento da função.

No exemplo x^2 - sen(2x), temos o seguinte gráfico:

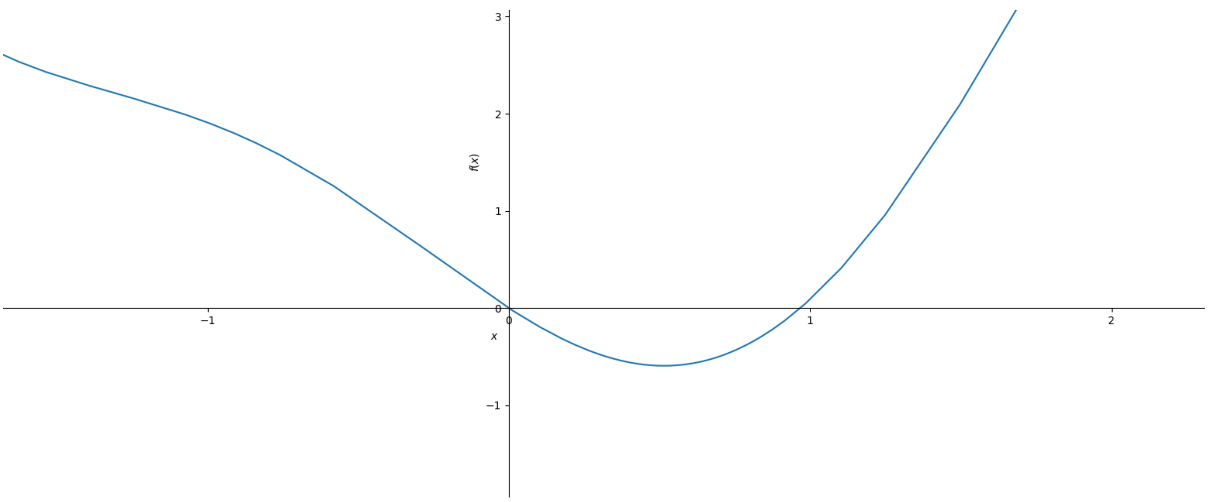


figura 1

Observa-se claramente a presença de uma raiz real em x=0 e outra raiz situada em algum ponto entre 0.1 e 1, conforme evidenciado pelo gráfico da figura 1. Com base nessa visualização, estabelecemos um intervalo inicial promissor , e a partir desse intervalo podemos seguir para o cálculo da outra raíz.

2.3 Método da Bisseção

O método da bisseção é uma técnica simples para encontrar uma raiz de uma função contínua $f(x)$ em um intervalo [a,b], onde $f(a)\ e\ f(b)$ têm sinais opostos (ou seja, a função muda de sinal no intervalo). A ideia fundamental é reduzir repetidamente o intervalo onde a raiz está localizada até que seja suficientemente pequeno.

O primeiro passo do método é a escolha do intervalo inicial. A escolha pode ser feito a partir de uma tabela que será construída com diversos valores de x , o objetivo dessa tabela é certificar-se de estar escolhendo dois intervalos que $f(a)\ e\ f(b)$ tenham sinais opostos ou seja $f(a)\ *\ f(b) < 0$
Uma outra maneira para definir A e B iniciais seria observando o gráfico da função f(x) = 0

Segundamente, para a primeira iteração, você calcula o ponto médio do intervalo [a,b] usando a fórmula $c = a + b/2$ .Este ponto médio é a primeira aproximação para a raiz.

O terceiro passo é localizar em qual subintervalo a raíz está localizada

Se $f(c)$ é suficientemente próximo de zero (conforme um critério de tolerância), então C é uma boa aproximação para a raiz, e você pode encerrar o método.

Se $f(a)\, .\, f(c) < 0$,isso significa que a raiz está no subintervalo [a,c] então $b = c$
Se $f(b)\, .\, f(c) < 0$,isso significa que a raiz está no subintervalo [b,c] então $a = c$

E por fim o critério de parada $|(b - a)/2| > \varepsilon$
A variável ε pode assumir, por exemplo, valores como 10^ -3 , 10^ -5 , 10^ -6 . Esse valor vai depender da natureza do problema a ser resolvido .

2.4 Método de Newton-Raphson

O Método de Newton-Raphson é um algoritmo iterativo para encontrar raízes de funções reais diferenciáveis. Este método utiliza a derivada da função para se aproximar rapidamente da raiz. Aqui estão os passos básicos do método:

O ponto inicial pode ser calculada pelo método da bisseção, ou seja,

$$x0 = (a + b)/2$$

A fórmula de Newton-Raphson para Xn é:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

E o critério de parada é:

$$|f(xn)| < \varepsilon$$

Dessa forma, é criado um método iterativo a fim de encontrar uma raiz real para a $f(x) = 0$.

3. Explicando o código

Para esse trabalho foi utilizado o Python .

3.1 Método da bisseção

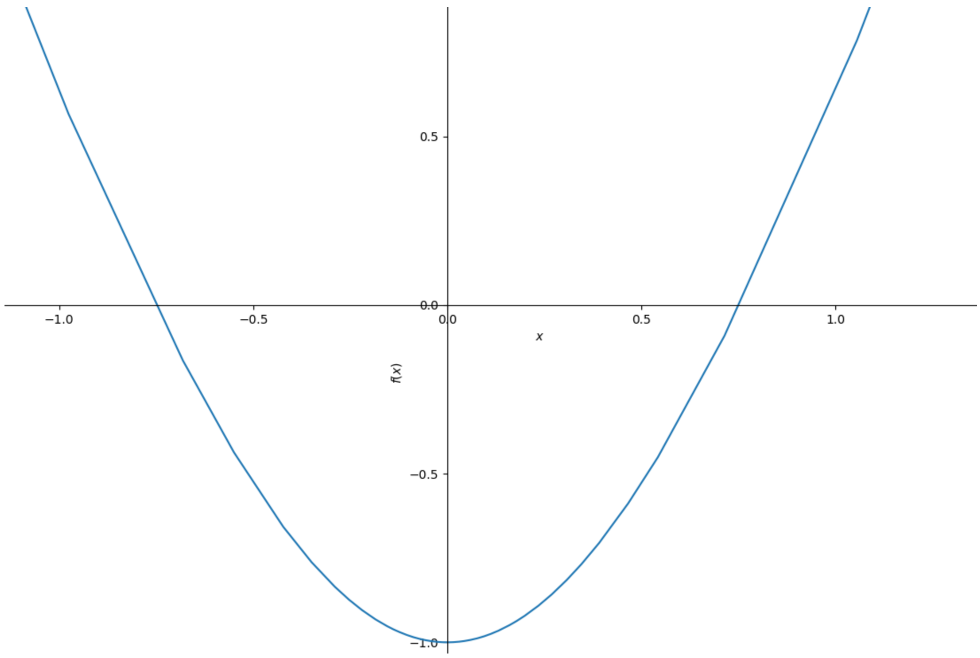
Questão: Calcule, utilizando qualquer método numérico uma aproximação de uma raiz real para as seguintes equações. Pare o processo quando o módulo de $f(xr) < \varepsilon$

a)$x^2 - e\hat{}(-x^2)$

Começamos o código importando as bibliotecas sympy para definir uma função com os símbolos e plotar o gráfico da função. Depois definimos a função f(x) como um função no python.

```
from sympy import symbols, sin,ln,cos ,exp,sqrt
from sympy.plotting import plot


# Define a variável simbólica
x = symbols('x')

# Plota o gráfico da função

f_x = x**2 - exp(-x**2)
p1 = plot(f_x, show=True)

#Define a função f(x)
def f(x):
    y = x**2 - exp(-x**2)
    return y
```

Ao rodar o código temos a seguinte imagem:



Percebe-se que existem duas raízes reais, uma entre o intervalo [0.5 ; 1.0] e outra entre [-0.5 ; -1.0]. E elas são simétricas .

Explicação do algoritmo:

```python
#Define o algoritimo da bisseção como função
def bissecao(a, b, E, max_iter):
    #Bolzano
    if f(a) * f(b) >= 0:
        print("Não há raiz neste intervalo.")
        return None
    #bisseção
    iteracao = 0
    while abs((b - a) / 2 )> E and iteracao < max_iter:
        xI = (a + b) / 2
        if f(xI) == 0:
            print("A raiz é:", xI)
            return xI
        else:
            if f(a) * f(xI) < 0:
                b = xI
            else:
                a = xI
        iteracao += 1

    if iteracao == max_iter:
        print("Número máximo de iterações atingido.")
        return None
    print("O numero de iteraçoes foi:",iteracao)
    print("A raiz é:", xI)
    return xI

print("Digite o intervalo inicial [a,b] e uma Tolerância(epsilon)")

a, b, E = float(input()), float(input()), float(input())
max_iter = 100


bissecao(a, b, E, max_iter)
```

1. O teorema de bolzano $f(a)\ *\ f(b)\ >=\ 0$,garante que não entraremos com um intervalo que não contenha nenhuma raíz.
2. Inicializamos um número de iterações igual a 0.
Depois criamos um loop while que vai rodar enquanto o módulo de (b-a)/2 for maior que Epsilon ou o número de iterações for menor que um número máximo de iterações;
3. A variável Xi recebe o valor de $(a\ +\ b)/2$.
Se Xi for igual a 0 significa que uma raíz foi encontrada, caso contrário o programa segue.
4. Em seguida temos f(a)*f(xi) < 0, se for verdadeiro, a variável b vai receber o valor de xI e o algoritmo segue. Caso contrário a variável a recebe o valor de xI.

5. O programa vai seguir enquanto não encontrar uma raiz que satisfaça o critério de parada .
6. Por fim, imprime o número de iterações e o valor da raiz numérica

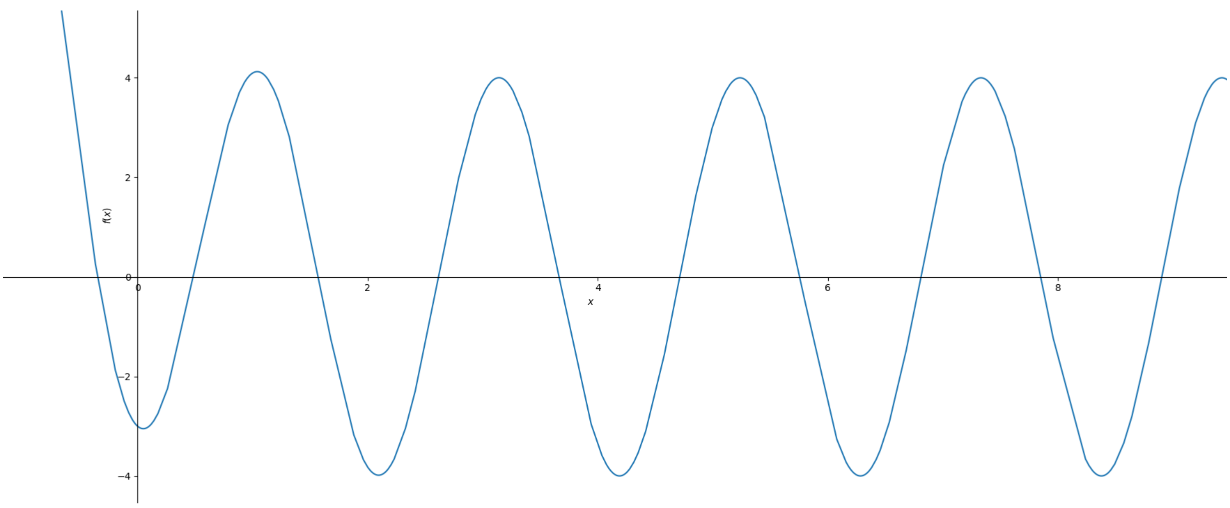
3.2 Método de Newton-Raphson

Questão: $e\hat{}(-\ 2x)\ \ -\ \ 4\ *\ cos(3x)$

E mantivemos o mesmo raciocínio usado acima. Importamos as bibliotecas necessárias, definimos as funções f(x) e além disso definimos uma função que irá calcular a derivada de f(x) numericamente.

```python
from sympy import symbols, ln, exp, cos,sin
from sympy.plotting import plot

x = symbols('x')

# Plota o gráfico da função
f_x = exp(-2*x) - 4*cos(3*x)
p1 = plot(f_x, show=True)

# Define a função f(x)
def f(x):
    fx = exp(-2*x) - 4*cos(3*x)
    return fx

# Define a função para calcular a derivada numericamente
def fLx(x):
    h = 0.0000001
    derivada = ((f(x + h) - f(x)) / h)
    return derivada
```

Ao rodar o código temos a imagem do gráfico da função e podemos ter a ideia de um intervalo inicial x0.



Explicando o algoritmo:

```python
# Implementação do Método de Newton-Raphson
def newtonRaphson(fx, fLx, x, iter_max, tol):
    i = 0
    while i <= iter_max:
        x = x - fx(x) / fLx(x)
        i += 1

        if abs(fx(x)) < tol:
            print("O numero de iteraçoes foi:",i)
            return 'A raiz aproximada é', x
    return 'O método falhou após', i, 'iterações'

# Chamar a função Newton-Raphson
resultado = newtonRaphson(f, fLx, 3.5 , 100, 1e-8)
print(resultado)
```

Nesse algoritmo, definimos uma função Newton-Raphson que entra com os seguintes parâmetros:
A função f(x), a derivada numérica fLx , um x0 inicial que pode ser calculado utilizando o método da bisseção $(a + b/2)$, um número máximo de iterações e a tolerância Epsilon .

1- Iniciamos um i inicial igual a 0.
2- Criamos um loop While, enquanto i for menor ou igual ao número máximo de iterações o algoritmo vai seguir.
3- entramos com a fórmula do método $x = x - (f(x)/f'(x))$
A variável x vai receber um novo valor a cada iteração.
4- Se o módulo da f(x) for menor que a Tolerância epsilon o programa imprime quantas iterações foram necessárias e qual é o valor da raiz aproximada
5- Caso o algoritmo não encontre nenhuma raiz o programa se encerra e imprime a mensagem que o método falhou.

6- Ao chamar a função é necessário entrar com um x inicial(Nesse caso escolhemos 3 e 4 como um intervalo [a;b]), um número máximo de iterações e o Epsilon.