O Método da Bissecção

Alberson da Silva Miranda[a,*]

[a] *Instituto Federal do Espírito Santo, Licenciatura em Matemática, Av. Vitória, Vitória, 29.040-780*

Resumo

O método da bissecção é um método numérico baseado no Teorema de Bolzano-Weierstrass para encontrar raízes de equações. Este trabalho tem como objetivo oferecer uma implementação em Python do método da bissecção.

Palavras-chave: álgebra, teorema do valor intermediário

1. INTRODUÇÃO

Para encontrar soluções de equações não lineares $f : f(x) = 0$, talvez o método mais simples a ser implementado seja o *médodo da bissecção*. Bissectar significa dividir (*sectar*) em duas partes (*bi*). Esse método é um corolário do Teorema do Valor Intermediário de Bolzano-Weierstrass.

Teorema 1.1 (Teorema do Valor Intermediário)**.** *Se f for uma função contínua em um intervalo fechado $[a, b]$ e k um número qualquer entre $f(a)$ e $f(b)$, inclusive, então existe no mínimo um número x no intervalo $[a, b]$, tal que $f(x) = k$ (Anton et al., 2012).*

E uma consequência direta desse teorema é sua aplicação para encontrar raízes de equações não lineares na forma do seguinte corolário:

Corolário 1.1. *Se f for uma função contínua em $[a, b]$, e $f(a)$ e $f(b)$ forem diferentes de zero com sinais opostos, então existe, no mínimo, uma solução para a equação $f(x) = 0$ no intervalo (a, b).*

Demonstração

Como $f(a)$ e $f(b)$ têm sinais opostos, 0 está entre $f(a)$ e $f(b)$. Dessa forma, por Teorema 1.1, existe no mínimo um número x no intervalo $[a, b]$, tal que $f(x) = 0$. Contudo, $f(a)$ e $f(b)$ são diferentes de zero, logo x deve estar situado entre (a, b), o que completa a prova (Anton et al., 2012).

Este trabalho tem como objetivo oferecer uma implementação em Python do método da bissecção para encontrar raízes de equações não lineares.

2. METODOLOGIA

Para implementarmos o método da bissecção, precisamos de uma função $f : [a, b] \to \mathbb{R}$ contínua em um intervalo tal que $f(a)$ e $f(b)$ tenham sinais opostos e, consequentemente, $f(a)f(b) < 0$. O método da bissecção consiste em dividir o intervalo $[a, b]$ ao meio, obtendo os subintervalos $[a, m]$ e $[m, b]$, onde $m = \frac{a+b}{2}$ é o ponto médio do intervalo, e considerar como novo intervalo aquele que contém a raiz da equação $f(x) = 0$,

[*] Corresponding author

Email address: `alberson.miranda@estudante.ifes.edu.br` (Alberson da Silva Miranda)

ou seja, o intervalo em que f tem sinais opostos nos extremos. Esse processo é repetido interativamente até se alcançar a precisão desejada (Andrade, 2020).

A partir disso, podemos encontrar uma solução para a equação $f(x) = 0$ no intervalo (a, b). Tome, por exemplo, a função

$$f(x) = -2x^6 - 1.5x^4 + 10x + 2 \tag{1}$$

para determinarmos sua raiz. Por ser um método numérico, precisamos definir um ϵ para o erro relativo. Para isso, vamos definir $\epsilon = 0.00001$, ou seja, queremos nos aproximar da raiz de $f(x)$ com 10^{-5} de precisão.

```python
# definindo a função
def f(x):
  f_x = -2*x**6 - 1.5*x**4 + 10*x + 2
  return(f_x)

# definindo epsilon
epsilon = 0.00001

# definindo intervalo
x_a = 1
x_b = 2

# se f(a) e f(b) têm sinais opostos
if f(x_a) * f(x_b) < 0:
  # calcula erro relativo
  erro = abs((x_b - x_a) / x_b)
  # enquanto erro é maior que epsilon
  while erro > epsilon:
    # calcula bissecção
    x_0 = (x_a + x_b) / 2
    # encontra em qual lado da bissecção está a raiz
    if f(x_a) * f(x_0) < 0:
      # se raiz está no intervalo [a, bissecção], substitui b
      x_b = x_0
    else:
      # se raiz está no intervalo [bissecção, b], substitui a
      x_a = x_0
    # calcula novo erro
    erro = abs((x_b - x_a) / x_b)
  # printa valor de x
  print("O valor de x é: " + str(round(x_0, 6)))
# caso f(a) e f(b) não tenham sinais opostos
else:
  print("não há raiz no intervalo")

# verifica se imagem é 0
print("O valor da imagem deve ser zero: " + str(round(f(x_0), 6)))
```

```
O valor de x é: 1.321281
O valor da imagem deve ser zero: -0.000311
```

No exemplo acima, encontramos $x = 1.321$, o que nos rendeu uma imagem de $f(x) = -0.000311$, ou seja, $f(x)$ é muito próximo de zero. Para visualizarmos melhor o resultado, podemos plotar o gráfico de $f(x)$ e verificar se o valor encontrado é realmente uma raiz.

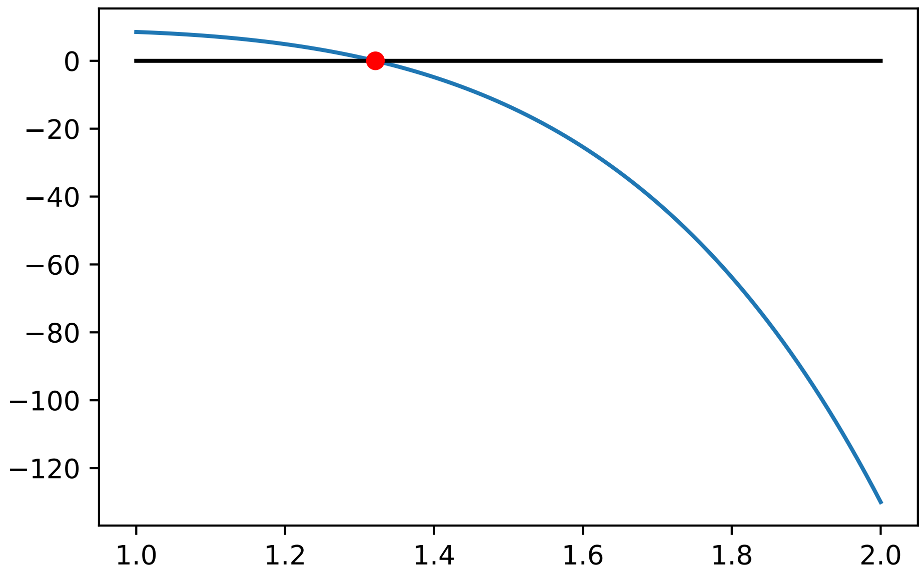
```python
# importando biblioteca
import matplotlib.pyplot as plt
import numpy as np

# definindo função
def f(x):
  f_x = -2*x**6 - 1.5*x**4 + 10*x + 2
  return(f_x)

# definindo intervalo
x = np.linspace(1, 2, 100)

# setando valor de x
x_0 = 1.321281

# plotando gráfico
plt.plot(x, f(x))
# plotando eixo x
plt.plot(x, 0*x, 'k')
# plotando ponto de interseção
plt.plot(x_0, f(x_0), 'ro')
# exibindo gráfico
plt.show()
```



REFERÊNCIAS

Andrade, D., 2020. Método da bissecção. URL: https://metodosnumericos.com.br/bisseccao.pdf.

Anton, H., Bivens, I., Davis, S., 2012. Calculus: early transcendentals. 10th ed ed., John Wiley & Sons, Hoboken, NJ. OCLC: 776774492.