Uso e aplicação em Java de Métodos Numéricos na Obtenção de Raízes de Equações

Usage and Application of Numerical Methods in Java for Finding Roots of Equations

Maria Eduarda dos Santos Pinheiro[1], Wellington José Corrêa[2]

RESUMO

O artigo em questão explora como esta combinação de métodos numéricos e programação em Java pode ser eficaz na resolução de problemas matemáticos complexos. O estudo começa destacando a importância da busca pelas raízes das equações em diversos campos, desde a engenharia até as ciências físicas e econômicas. Em seguida, centra-se na utilidade dos métodos numéricos para resolver estas equações, especialmente quando não existe uma solução analítica direta disponível. Java é apresentado como uma linguagem de programação versátil e amplamente utilizada que pode ser empregada para implementar esses métodos numéricos. A linguagem oferece recursos que facilitam a implementação de algoritmos de resolução de equações, tornando o processo mais eficiente. O artigo discute alguns dos métodos numéricos mais comuns usados para encontrar raízes de equações, como o método da bissecção, o método de Newton-Raphson, o método da secante e o método Regula Falsi. Também aborda considerações de precisão e eficiência computacional ao aplicar esses métodos. Por fim, para garantir resultados precisos e confiáveis, conclui-se a relevância da combinação de métodos numéricos e programação em Java na resolução de problemas práticos envolvendo raízes de equações.
PALAVRAS-CHAVE: raízes de equações; Java; métodos numéricos.

ABSTRACT

The article in question addresses explores how this combination of numerical methods and programming in Java can be effective in solving complex mathematical problems. The study begins by highlighting the importance of searching for the roots of equations in a variety of fields, from engineering to physical and economic sciences. It then focuses on the usefulness of numerical methods for solving these equations, especially when there is no direct analytical solution available. Java is touted as a versatile and widely used programming language that can be employed to implement these numerical methods. The language offers features that facilitate the implementation of equation-solving algorithms, making the process more efficient. The article discusses some of the most common numerical methods used to find roots of equations, such as the bisection method, the Newton-Raphson method, the secant method, and the Regula Falsi method. It also addresses considerations of accuracy and computational efficiency when applying these methods. Finally, to ensure accurate and reliable results it concludes the relevance of combining numerical methods and programming in Java in solving practical problems involving roots of equations.
KEYWORDS: root of equations; Java; numerical methods.

[1] Bolsista do CNPq. Universidade Tecnológica Federal do Paraná, Campo Mourão, Paraná, Brasil. E-mail: mariaesp10052005@gmail.com. ID Lattes: 8150213152124979.
[2] Docente no Departamento Acadêmico de Matemática. Universidade Tecnológica Federal do Paraná, Campo Mourão, Paraná, Brasil. E-mail: wcorrea@utfpr.edu.br. ID Lattes: 1045931096324971.

INTRODUÇÃO

A necessidade de encontrar valores de $x = \xi$ que satisfazem à equação $f(x) = 0$ aparece frequentemente em uma ampla variedade de problemas provenientes das Ciências e das Engenharias. Esses valores especiais são chamados de raízes da equação $f(x) = 0$ ou zeros da função $f(x)$.

Para equações algébricas de grau até quatro, suas raízes podem ser calculadas por meio de uma expressão, tal como $\frac{-b \pm \sqrt{\Delta}}{2\,a}$ para determinar as duas raízes de $f(x) = ax^2 + bx + c = 0$.

No entanto, para equações algébricas de grau superior a quatro e para a grande maioria das equações transcendentes, as raízes não podem ser calculadas analiticamente. Nesses casos, precisam ser usados métodos que encontrem uma solução aproximada para essas raízes.

O problema de calcular uma raiz pode ser dividido em duas fases:

(i) Isolamento da raiz, isto é, encontrar um intervalo [a,b] que contenha uma, e somente uma, raiz de $f(x) = 0$.

(ii) Refinamento da raiz, ou seja, a partir de um valor inicial $x_0 \in [a,b]$, gerar uma sequência $x_0,x_1,...,x_k,...$ que converge para uma raiz exata de ξ de $f(x) = 0$.

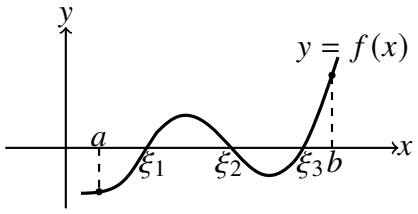
Para obter o item (i), isto é, isolar uma raiz, deve-se recorrer ao seguinte resultado:

Teorema 0.1. *Se uma função contínua $f(x)$ assume valores de sinais opostos nos pontos extremos do intervalo [a,b], isto é, $f(a) \cdot f(b) < 0$, então, o intervalo conterá, no mínimo, uma raiz da equação $f(x) = 0$, em outras palavras, haverá, no mínimo, um número $\xi \in (a,b)$ tal que $f(\xi) = 0$.*

Demonstração. Veja (GUIDORIZZI, 2000), página 511. □

Segundo (DORNELLES FILHO, 2016), embora uma prova rigorosa do teorema possa ser encontrada em (GUIDORIZZI, 2000), graficamente o seu entendimento é imediato: sem perda de generalidade, pode-se assumir que se o gráfico de uma função está abaixo do eixo horizontal em $x = a$ e segue continuamente até acima do eixo horizontal em $x = b$, então, em algum ponto entre a e b, o gráfico passou sobre o eixo horizontal. A Figura 1 ilustra tal situação:

Figura 1 – Quando $f(a) \cdot f(b) < 0$ tem-se pelo menos uma raiz da equação $f(x) = 0$.



Antes de serem abordados os métodos para obter a sequência $\{x_0, x_1, \ldots, x_k, \ldots, \xi\}$ que convirja para a raiz exata ξ de $f(x) = 0$, é necessário definir um critério de parada, ou seja, quando se deve interromper a geração desta sequência. Na prática, a sequência é interrompida quando seus valores satisfazerem a pelo menos um dos critérios:

1. $|x_k - x_{k+1}| \leq \varepsilon$
2. $\frac{|x_k - x_{k+1}|}{|x_{k+1}|} \leq \varepsilon$
3. $|f(x_k)| \leq \varepsilon$

MATERIAIS E MÉTODOS

Como mencionado anteriormente, os métodos numéricos usados neste trabalho foram implementados na linguagem Java. O Java é uma linguagem de programação de alto nível que foi desenvolvida pela *Sun Microsystems* (agora parte da *Oracle Corporation*) na década de 1990. Ela se destaca por várias características e é amplamente utilizada em uma variedade de aplicações. O desenvolvimento das lógicas dos métodos numéricos foi realizado no ambiente de programação *NetBeans*, escolhido por sua praticidade e eficiência.

Como mencionado anteriormente, o intuito é empregar os seguintes métodos numéricos para determinar raízes de uma equação:

- Método da Bissecção

  O método da bissecção é um dos métodos numéricos mais simples e eficazes para encontrar as raízes de uma equação em um intervalo específico. No que segue, recorre-se aos seguintes passos:

  1. **Escolha Inicial**: Inicialmente, você precisa escolher um intervalo [a, b] onde a raiz está contida. Certifique-se de que $f(a)$ e $f(b)$ tenham sinais opostos, indicando que há uma raiz no intervalo.
  2. **Iterações**: O método da bissecção divide repetidamente o intervalo ao meio, calculando o valor de f no ponto médio $c = \frac{a+b}{2}$.
  3. **Critério da Parada**: Verifique se o critério da parada foi satisfeito. Se isso acontecer, o método é concluído, e c é uma boa aproximação da raiz.
  4. **Atualização do Intervalo**: Se $f(c)$ não atender ao critério da parada, atualize o intervalo [a, b] de acordo com o sinal de $f(c)$. Se $f(a)$ e $f(c)$ tiverem sinais opostos, defina $b = c$; caso contrário, defina $a = c$.
  5. **Repetição**: Repita os passos 2 a 4 até que o critério da parada seja satisfeito.

- Método de Newton-Raphson

  O método de Newton-Raphson é um método iterativo amplamente utilizado para encontrar raízes de equações. Ele é particularmente eficaz quando se busca uma raiz próxima a um ponto de partida inicial razoável. A seguir, usa-se os seguintes passos:

  1. **Escolha Inicial**: Escolha um intervalo $[a,b]$ de modo que o produto $f(a) \cdot f(b)$ seja negativo. Em seguida, de acordo com (CAMPOS FILHO, 2018), considere um valor inicial $x_0 \in [a,b]$ tal que $f(x_0) \cdot f''(x_0) > 0$. Com tal escolha de x_0, é possível construir uma sequência $\{x_0, x_1, \ldots, x_k, \ldots\}$ que converge para a raiz $\xi \in [a,b]$.
  2. **Iterações**: A partir de x_0 escolhido adequadamente acima, itere usando a fórmula iterativa:

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

onde $f'(x_n)$ é a derivada de f em relação a x no ponto x_n.

3. **Critério da parada**: Continue as iterações até que um dos critérios da parada apresentados acima seja atendido.
4. **Resultado**: O valor de x após o critério da parada ser contemplado, é uma boa aproximação para a raiz da equação $f(x) = 0$.

- Método da Secante

  O método da secante é um método numérico para encontrar raízes de equações que não requer o cálculo direto da derivada da função. Este método é especialmente útil quando não é prático ou fácil calcular a derivada analítica da função. A seguir, emprega-se os seguintes passos:

  1. **Escolha Inicial**: Inicie com dois pontos iniciais $x_0 = a$ e $x_1 = b$ de modo que $f(a) \cdot f(b) < 0$.
  2. **Iterações**: Para cada iteração subsequente, calcule o próximo ponto x_{n+1} usando a fórmula iterativa:
  $$x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \tag{2}$$
  3. **Critério da parada**: Continue as iterações até que um dos critérios da parada apresentados acima seja atendido.
  4. **Resultado**: O valor de x após a convergência é uma boa aproximação para a raiz da equação $f(x) = 0$.

- Método Regula Falsi

  O método Regula Falsi possui os mesmos passos do método da secante com diferença que antes de avaliar o critério da parada, tal método retém o ponto no qual o valor da função tem sinal oposto no ponto mais recente, assegurando desta forma, que a raiz continue isolada entre dois pontos.

Para mais detalhes sobre os métodos usados como por exemplo, convergência, vantagens e desvantagens, recomenda-se a leitura das seguintes referências (BURDEN, R. L.; FAIRES; BURDEN, A. M., 2016), (CAMPOS FILHO, 2018), (DORNELLES FILHO, 2016), dentre outros. As Figuras 2 e 3 nos mostram as implementações dos métodos da bisssecção e Newton-Raphson, respectivamente:

Figura 2 – Método da Bissecção

```
import ...2 lines
public class main {
    public static void main(String[] args) {
        Function<Double, Double> fa = x -> x * x - 4;
        double a = 0.1;
        double b = 1.0;
        double tolerancia = 1e-3;
        int maxInteracoes = 20;
        double root = Bissecao(fa, a, b, tolerancia, maxInteracoes);
        System.out.println("A raiz é: " + formatNumber(root));
    }
    public static double Bissecao(Function<Double, Double> fa, double a, double b, double tolerancia, int maxInteracoes) {
        int interacao = 0;
        System.out.println("Iteração\t\tValor de a\t\tValor de b\t\tValor de x\t\tValor de f(a)\t\tValor de f(x)\t\tValor absoluto da dife
        while (interacao < maxInteracoes) {
            double x = (a + b) / 2.0;
            double fValueA = fa.apply(a);
            double fValueX = fa.apply(x);
            System.out.printf("%d\t\t\t%.6f\t\t\t%.6f\t\t\t%.6f\t\t\t%.6f\t\t\t%.6f\n",interacao,a,b,x,fValueA,fValueX,Math.abs(b - a));
            if (Math.abs(b - a) < tolerancia || Math.abs(fValueX) < tolerancia) {
                return x;
            }
            if (fValueA * fValueX < 0) {
                b = x;
            } else {
                a = x;
            }
            interacao++;
        }
        throw new RuntimeException("O método não convergiu para uma raiz após " + maxInteracoes + " iterações.");
```

Fonte: Autoria própria (2023).

Figura 3 – Método de Newton-Raphson

```
package main;
import ...
public class main {
    public static void main(String[] args) {
        Function<Double, Double> fa = x -> x * x - 4;
        Function<Double, Double> fb = x -> 2 * x;
        double x0 = 1.2;
        double tolerancia = 1e-3;
        int maxInteracoes = 20;
        double root = NR(fa, fb, x0, tolerancia, maxInteracoes);
        System.out.println("A raiz é: " + root);
    }
    public static double NR(Function<Double, Double> fa, Function<Double, Double> fb, double x0, double tolerancia, int maxInteracoes) {
        double x = x0;
        int interacao = 0;
        System.out.println("Iteração\t\tValor de x\t\tValor de f(x)\t\tValor da derivada\t\tDelta x");
        while (interacao < maxInteracoes) {
            double fValue = fa.apply(x);
            double fPrimeValue = fb.apply(x);
            double deltaX = -fValue / fPrimeValue;
            x += deltaX;
            System.out.printf("%d\t\t\t%.6f\t\t\t%.6f\t\t\t%.6f\n", interacao, x, fValue, fPrimeValue, deltaX);
            if (Math.abs(deltaX) < tolerancia) {
                return x;
            }
            interacao++;
        }
        throw new RuntimeException("O método não convergiu para uma raiz após " + maxInteracoes + " iterações.");
    }
}
```

Fonte: Autoria própria (2023).

RESULTADOS E DISCUSSÕES

Os algoritmos utilizados em nosso estudo, forneceram como resultados apresentados, as saídas das implementações em JAVA dos métodos estudados. As Tabelas 1 e 2 apresentam as saídas dos códigos implementados apresentados nas Figuras 2 e 3, respectivamente para $f(x) = x^2 - 4 = 0$ no intervalo $[1,2]$. Como pode-se ver, adotando quatro casas decimais, o método de Newton-Raphson possui uma convergência mais rápida que o método da bissecção com valor aproximado da raiz $x = 2$. com erro absoluto nulo e três iterações, enquanto que o método da bissecção teve valor aproximado

da raiz de $f(x)$ como sendo $x = 1{,}996$ e erro absoluto 0,007 e sete iterações.

Tabela 1 – Método da Bissecção

Iteração	Valor de a	Valor de b	Valor de x	Valor de $f(a)$	Valor de $f(x)$	Erro Absoluto
0	1	2	1,500	-3,000	-1,750	–
1	1,500	2	1,750	-1,750	-0,937	0,500
2	1,750	2	1,875	-0,937	-0,484	0,250
3	1,875	2	1,937	-0,484	-0,246	0,125
4	1,937	2	1,968	-0,246	-0,124	0,062
5	1,968	2	1,984	-0,124	-0,062	0,031
6	1,984	2	1,992	-0,062	-0,031	0,015
7	1,992	2	1,996	-0,031	-0,015	0,007

Fonte: Autoria própria (2023).

Tabela 2 – Método de Newton-Raphson

Iteração	Valor de x	Valor de $f(x)$	Valor da Derivada	Erro absoluto
0	2,266	-2,560	2,40	1,06
1	2,015	1,137	4,53	0,250
2	2,000	0,062	4,03	0,015
3	2,000	0,0002	4,00	0,000

Fonte: Autoria própria (2023).

CONCLUSÃO

O Java se mostrou uma importante ferramenta na obtenção da raiz de uma equação. Além disso, tal ferramenta permite que os educadores ensinem conceitos matemáticos e científicos complexos de uma maneira mais acessível e visual.

Disponibilidade de Código

A implementação computacional desenvolvida está disponível no *GitHub*, por meio do link: https://github.com/mariaesp1005/metodosJava.git.

Conflito de interesse

Não há conflito de interesse.

REFERÊNCIAS

BURDEN, Richard L; FAIRES, J Douglas; BURDEN, Annette M. **Análise numérica**. [S.l.]: Cengage Learning, 2016.

CAMPOS FILHO, Frederico Ferreira. **Algoritmos numéricos: uma abordagem moderna de cálculo numérico**. [S.l.]: Editora LTC, 2018.

DORNELLES FILHO, Adalberto Ayjara. **Fundamentos de cálculo numérico**. [S.l.]: Bookman Editora, 2016.

GUIDORIZZI, Hamilton Luiz. **Um curso de cálculo, vol. 1 .** [S.l.]: Grupo Gen-LTC, 2000.