Aproximacao de raiz de newton raphson

O método de Newton–Raphson é uma técnica iterativa para aproximar raízes de uma função, ou seja, resolver:

$f(x)=0$

Ideia intuitiva:

Você começa com um chute inicial x_o.

Em cada passo, você usa a reta tangente da função nesse ponto para chegar a uma aproximação melhor da raiz.

Fórmula do método:

A iteração é dada por:

$x_{n+1} = x_n - f(x_n) / f'(x_n)$

Onde:

$f(x)$ é a função

$f'(x)$ é a derivada

x_n é a aproximação atual

x_{n+1} é a próxima aproximação

Passo a passo:

Escolha um valor inicial

1) x_0
2) Calcule $f(x_0)$ e $f'(x_0)$
3) Aplique a fórmula
4) Repita até:
    a) o erro ser pequeno, ou
    b) atingir o número máximo de iterações

Implementação genérica em Zig:

```
const std = @import("std");

pub fn newtonRaphson(
    f: fn (f64) f64,
    df: fn (f64) f64,
    x0: f64,
    eps: f64,
    max_iter: usize,
) !f64 {
    var x = x0;
    var i: usize = 0;

    while (i < max_iter) : (i += 1) {
        const fx = f(x);
        const dfx = df(x);
        if (@abs(dfx) < eps) {
            return error.DerivadaZero;
        }
        const x_next = x - fx / dfx;
        if (@abs(x_next - x) < eps) {
            return x_next;
        }
        x = x_next;
    }
    return error.NaoConvergiu;
}

fn f(x: f64) f64 {
    return x * x - 2.0;
}

fn df(x: f64) f64 {
    return 2.0 * x;
}


pub fn main() !void {
    const root = try newtonRaphson(
        f,
        df,
        1.0,     // chute inicial
        1e-12,   // tolerância
        100      // max iterações
    );

    std.debug.print("Raiz ≈ {d}\n", .{root});
}
```

Erros customizados:

Zig cria erros automaticamente a partir do error.X:

- DerivadaZero
- NaoConvergiu

Observações importantes em Zig:

✔ Funções como parâmetros

- Zig aceita fn (f64) f64 diretamente
- Não há overhead de closures

✔ Segurança numérica

- Sempre cheque df(x) ≈ 0
- Use f64 para melhor convergência

✔ Performance

- Convergência quadrática
- Pouquíssimas iterações