Relatório: Método de Newton-Raphson com Visualização via Flask

Marcelo Roner
Universidade Federal De Goiás

21 de março de 2025

Sumário

1 Introdução

O *Método de Newton-Raphson* é um dos métodos mais populares para a busca de raízes de equações não lineares do tipo

$$f(x) = 0.$$

Ele possui convergência local e tipicamente de ordem 2 (isto é, converge quadraticamente), desde que algumas condições sejam satisfeitas. Além de exigir o cálculo de $f(x)$, o Método de Newton-Raphson também exige a derivada $f'(x)$.

Este relatório documenta uma implementação em Python usando o microframework `Flask` para fornecer uma interface web que recebe os parâmetros do método e a própria função, realiza o cálculo iterativamente, exibe o resultado textual das iterações e produz um *gráfico de convergência* do erro, em escala logarítmica.

2 Fundamentação Teórica

2.1 Forma Geral do Método de Newton-Raphson

Suponha que desejemos resolver

$$f(x) = 0,$$

onde f é contínua e *derivável* em uma vizinhança do zero procurado x^*. O Método de Newton-Raphson propõe a seguinte fórmula de iteração:

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

desde que $f'(x_n) \neq 0$. Geometricamente, a ideia é aproximar f por uma reta tangente na vizinhança de x_n e encontrar a raiz dessa reta para obter x_{n+1}.

2.2 Análise de Convergência

Suponha que $f'(x^*) \neq 0$ e que f seja duas vezes continuamente diferenciável (para simplificar a análise). Defina o erro

$$e_n = x_n - x^*.$$

Sabemos que $f(x^*) = 0$. Fazendo uma expansão de Taylor de f em torno de x^*, temos

$$f(x_n) = f'(x^*)\, e_n \; + \; \frac{f''(\xi_n)}{2}\,(e_n)^2,$$

para algum ξ_n entre x_n e x^*. Então,

$$x_{n+1} - x^* \;=\; e_{n+1} \;=\; \Big(x_n - \frac{f(x_n)}{f'(x_n)}\Big) - x^* \;=\; e_n - \frac{f'(x^*)\, e_n + \frac{f''(\xi_n)}{2}(e_n)^2}{f'(x_n)}.$$

Visto que $f'(x_n) = f'(x^*) + O(e_n)$, resulta

$$e_{n+1} \;=\; -\,\frac{f''(\xi_n)}{2\, f'(x^*)}\,(e_n)^2 + O\Big((e_n)^3\Big).$$

Em outras palavras,

$$|e_{n+1}| \;\approx\; C \cdot (|e_n|)^2,$$

para alguma constante C. Assim, a convergência do Método de Newton é local e *quadrática*, significando que, a cada iteração, o número de casas decimais corretas dobra aproximadamente (desde que estejamos suficientemente perto de x^* e $f'(x^*) \neq 0$).

3 Descrição do Código e Interface Web

A seguir, apresentamos o código-fonte em Python que implementa o `Flask` para criar uma interface web. O usuário insere:

- equation: A função $f(x)$ em formato Python (por exemplo, `x**2 - 2`).
- x_0: Chute inicial (valor real).
- tol: Tolerância para o critério de parada, isto é, se $|x_{n+1} - x_n| < \text{tol}$, paramos.
- max_iter: Número máximo de iterações permitidas.
- verbose: Se marcado, exibe log detalhado das iterações.

Além disso, o código gera um *gráfico de convergência* em escala logarítmica, mostrando o erro $|x_{n+1} - x_n|$ a cada iteração.

3.1 Código em Python (app.py)

A função principal `newton_method_symbolic` recebe a equação em formato string, interpreta-a com `sympy`, e itera. Observamos que:

- Se a derivada $f'(x_n)$ ficar muito próxima de zero, interrompemos para evitar divisão por zero.
- Armazenamos cada aproximação x_n em `approximations`.
- No final, construímos a lista de erros sucessivos $|x_{n+1} - x_n|$ para plotar.

```python
from flask import Flask, render_template, request
import sympy as sp
import numpy as np
import matplotlib
matplotlib.use('Agg')  # permite uso do matplotlib sem ambiente gráfico
import matplotlib.pyplot as plt
import io, base64

app = Flask(__name__)

def newton_method_symbolic(equation_str, x0, tol=1e-6, max_iter=100, verbose=False):
    x = sp.symbols('x')
    try:
        f_sym = sp.sympify(equation_str)
    except Exception as e:
        return None, False, 0, f"Erro na interpretação da equação: {e}", []
    df_sym = sp.diff(f_sym, x)
    f = sp.lambdify(x, f_sym, 'numpy')
    df = sp.lambdify(x, df_sym, 'numpy')

    log = []
    approximations = []  # guardar cada x_i
    x_current = x0

    for i in range(1, max_iter + 1):
        try:
```

```
            f_val = f(x_current)
            df_val = df(x_current)
        except Exception as e:
            return x_current, False, i, f"Erro ao avaliar a função: {e}",
            ↪ approximations

        approximations.append(x_current)

        if abs(df_val) < 1e-12:
            log.append(f"Iteração {i}: Derivada muito próxima de zero em x =
            ↪ {x_current:.6f}.")
            return x_current, False, i, "\n".join(log), approximations

        x_next = x_current - f_val / df_val

        if verbose:
            log.append(
                f"Iteração {i}: x = {x_current:.6f}, "
                f"f(x) = {f_val:.6f}, f'(x) = {df_val:.6f}, "
                f"Próximo x = {x_next:.6f}"
            )

        # Critério de parada
        if abs(x_next - x_current) < tol:
            approximations.append(x_next)
            return x_next, True, i, "\n".join(log), approximations

        x_current = x_next

    # Se saiu do loop sem convergir
    approximations.append(x_current)
    return x_current, False, max_iter, "\n".join(log), approximations


@app.route('/', methods=['GET', 'POST'])
def index():
    result = ''
    log_text = ''
    plot_url = ''

    if request.method == 'POST':
        equation = request.form.get('equation')
        x0 = float(request.form.get('x0'))
        tol = float(request.form.get('tol'))
        max_iter = int(request.form.get('max_iter'))
        verbose = 'verbose' in request.form

        root, converged, iters, log_text, approximations = newton_method_symbolic(
            equation, x0, tol, max_iter, verbose
        )

        if converged:
            result = f"Convergência alcançada: raiz = {root:.6f} em {iters}
            ↪ iterações."
        else:
            result = f"Não convergiu em {iters} iterações. Última aproximação:
            ↪ {root:.6f}"

```

```
        # Gera o gráfico de erro (|x_{n+1} - x_n|) em escala log
        if len(approximations) > 1:
            errors = []
            for i in range(len(approximations) - 1):
                errors.append(abs(approximations[i+1] - approximations[i]))

            fig, ax = plt.subplots(figsize=(6,4))
            ax.set_title("Evolução do Erro a cada Iteração (Método de Newton)")
            ax.set_xlabel("Iteração")
            ax.set_ylabel("Erro (|x_{n+1} - x_n|)")
            ax.set_yscale("log")
            ax.grid(True, which='both', ls='--')

            ax.plot(range(1, len(errors) + 1), errors, marker='o', linestyle='-')

            buf = io.BytesIO()
            plt.tight_layout()
            plt.savefig(buf, format='png')
            buf.seek(0)
            # Codifica em base64 para embutir no HTML
            plot_url = base64.b64encode(buf.getvalue()).decode('utf-8')
            plt.close(fig)

    return render_template(
        'index.html',
        result=result,
        log=log_text,
        plot_url=plot_url
    )


if __name__ == '__main__':
    app.run(debug=True)
```

3.2 Template HTML (index.html)

O HTML abaixo exibe os campos de entrada (equação, chute inicial etc.) e mostra:

- O resultado resumido da execução (convergência ou não).
- Um texto com todo o *log* das iterações, caso o usuário marque *Exibir detalhes*.
- Se o gráfico de erro foi gerado, ele é apresentado dentro de uma `<div>` de classe `plot-container`.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Método de Newton-Raphson</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            padding: 20px;
        }
```

```
        .container {
            max-width: 600px;
            margin: auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h1 {
            text-align: center;
            color: #333;
        }
        form input, form button, form label {
            width: 100%;
            padding: 10px;
            margin-top: 10px;
        }
        textarea {
            width: 100%;
            padding: 10px;
            height: 150px;
            margin-top: 10px;
        }
        button {
            background-color: #28a745;
            color: white;
            border: none;
            cursor: pointer;
        }
        button:hover {
            background-color: #218838;
        }
        .result, .log {
            margin-top: 15px;
            padding: 10px;
            background-color: #e9ecef;
            border-radius: 5px;
        }
        /* Container para o gráfico */
        .plot-container {
            margin-top: 15px;
            background-color: #fff;
            border-radius: 5px;
            padding: 10px;
            text-align: center;
        }
        /* Ajustes para a imagem do gráfico */
        img {
            max-width: 100%;
            height: auto;
            display: block;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Método de Newton-Raphson</h1>
```

```
        <form method="POST">
            <label for="equation">Equação f(x):</label>
            <input type="text" id="equation" name="equation" placeholder="Ex: x**2 -
            ↪  2" required>

            <label for="x0">Chute Inicial (x0):</label>
            <input type="number" step="any" id="x0" name="x0" placeholder="Ex: 1.0"
            ↪  required>

            <label for="tol">Tolerância:</label>
            <input type="number" step="any" id="tol" name="tol" placeholder="Ex:
            ↪  1e-6" required>

            <label for="max_iter">Máximo de Iterações:</label>
            <input type="number" id="max_iter" name="max_iter" placeholder="Ex: 100"
            ↪  required>

            <label for="verbose">
                <input type="checkbox" id="verbose" name="verbose"> Exibir detalhes
                ↪  das iterações
            </label>

            <button type="submit">Calcular</button>
        </form>

        {% if result %}
        <div class="result">
            <strong>Resultado:</strong>
            <p>{{ result }}</p>
        </div>
        {% endif %}

        {% if log %}
        <div class="log">
            <strong>Detalhes das Iterações:</strong>
            <textarea readonly>{{ log }}</textarea>
        </div>
        {% endif %}

        {% if plot_url %}
        <div class="plot-container">
            <h3>Gráfico de Convergência</h3>
            <img src="{{ plot_url }}" alt="Gráfico de Erro">
        </div>
        {% endif %}
    </div>
</body>
</html>
```

4 Conclusão

Apresentamos uma implementação do *Método de Newton-Raphson* usando `Flask` para a interação via navegador web. O usuário informa a função de interesse em formato simbólico (`x**2 - 2, por exemplo`), bem como o chute inicial, a tolerância e o número máximo de iterações. O método então itera conforme a derivada fornecida simbolicamente

por `Sympy`, e produz um gráfico para ilustrar a queda do erro a cada iteração.

O Método de Newton-Raphson destaca-se pela *convergência rápida (quadrática)* em comparação a outros métodos de raiz, desde que $f'(x^*) \neq 0$ e que o chute inicial esteja em uma região adequada. Em problemas em que a derivada é difícil de calcular ou se anula nas proximidades da raiz, outras estratégias podem ser necessárias (como o Método da Secante ou variações híbridas).

5 Referências

Referências

[1] R. L. Burden, J. D. Faires, *Análise Numérica*, 9ª ed. Cengage Learning, 2011.

[2] M. A. G. Ruggiero, V. L. Lopes, *Cálculo Numérico: Aspectos Teóricos e Computacionais*, Makron Books, 1997.

[3] `Sympy` documentation: `https://docs.sympy.org/`

[4] `Flask` documentation: `https://flask.palletsprojects.com/`