Relatório: Comparação de Duas Variações do Método do Ponto Fixo, incluindo o Método de Newton

Marcelo Roner
Universidade Federal de Goiás

21 de março de 2025

Sumário

1 Introdução

A busca por soluções de equações não lineares $f(x) = 0$ é essencial em diversas áreas, como engenharia, física, economia, entre outras. Dentre as variadas técnicas numéricas, o *Método do Ponto Fixo* ocupa posição de destaque pela simplicidade conceitual e por servir de porta de entrada para o estudo de métodos iterativos em geral.

A estratégia básica do Método do Ponto Fixo consiste em reformular a equação $f(x) = 0$ na forma

$$x = g(x),$$

de modo que a raiz procurada x^* seja um *ponto fixo* da função g, isto é, $g(x^*) = x^*$. Partindo de uma aproximação inicial x_0, gera-se uma sequência $\{x_n\}$ por

$$x_{n+1} = g(x_n),$$

esperando que $x_n \to x^*$ à medida que $n \to \infty$.

Neste relatório, comparamos **duas** variações do método de ponto fixo, a saber:

- **Método 1**: $g(x) = x - \lambda\, f(x)$, com λ como parâmetro de *relaxação*.
- **Método 2**: $g(x) = x - \dfrac{f(x)}{f'(x)}$, que, na prática, corresponde ao *Método de Newton* reescrito como um ponto fixo.

Ilustraremos a implementação computacional em Python de cada variação e, posteriormente, discutiremos os resultados obtidos ao aplicá-las à função $f(x) = e^{-x} - x$.

2 Fundamentação Teórica

Seja o problema de determinar x^* tal que $f(x^*) = 0$. Se escrevemos a equação como

$$x = g(x),$$

então x^* será um *ponto fixo* de g, ou seja, $g(x^*) = x^*$. Para construir uma sequência de aproximações $\{x_n\}$, definimos:

$$x_{n+1} = g(x_n).$$

2.1 Condição de Convergência Local

Uma condição clássica para a convergência local do Método do Ponto Fixo é o *Teorema do Ponto Fixo de Banach*, que, adaptado ao nosso contexto unidimensional, diz:

Teorema (Banach): Se existe um intervalo $[a, b]$ contendo x^* tal que $g(x)$ mapeia $[a, b]$ em si mesmo (i.e., $g : [a, b] \to [a, b]$) e

$$\max_{x \in [a,b]} \left| g'(x) \right| < 1,$$

então a sequência $x_{n+1} = g(x_n)$ converge para o ponto fixo x^* para qualquer $x_0 \in [a, b]$. Nessas condições, g é uma contração em $[a, b]$.

2.2 Análise de Erro

Seja $e_n = x_n - x^*$ o erro na n-ésima iteração. Supondo que g é continuamente diferenciável em x^*, podemos fazer a expansão de Taylor em torno de x^*:

$$x_{n+1} - x^* \;=\; g(x_n) - g(x^*) \;\approx\; g'(x^*)\,(x_n - x^*).$$

Portanto, em valor absoluto,

$$|e_{n+1}| \;\approx\; |g'(x^*)| \cdot |e_n|.$$

Se $\left|g'(x^*)\right| < 1$, o erro diminui de forma aproximadamente *geométrica*, isto é, $|e_{n+1}| \approx C\,|e_n|$ para algum $C < 1$. Assim, o método converge linearmente.

3 Descrição dos Dois Métodos

3.1 Método 1 (com Parâmetro de Relaxação)

A primeira variação, aqui chamada de **Método 1**, utiliza

$$x_{n+1} \;=\; g(x_n) \;=\; x_n - \lambda\, f(x_n). \tag{1}$$

Trata-se de uma forma simples de ponto fixo, cujo sucesso (ou seja, convergência) depende fortemente da escolha do λ. A justificativa para esta forma vem de tentar forçar uma *contração*:

$$g'(x) \;=\; 1 - \lambda\, f'(x).$$

Se conseguirmos garantir $\left|1 - \lambda\, f'(x)\right| < 1$ no intervalo considerado, então teremos convergência local. Muitas vezes, λ é chamada de *parâmetro de relaxação* ou *step size*.

3.2 Método 2 (Método de Newton Formulado como Ponto Fixo)

A segunda variação, denominada aqui de **Método 2**, faz uso do *Método de Newton*. Observamos que o passo iterativo de Newton clássico é

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

Reescrevendo:

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

vemos que a forma se encaixa perfeitamente na ideia de ponto fixo. Nesse caso,

$$g'(x) \;=\; 1 - \frac{f'(x)\, f'(x) - f(x)\, f''(x)}{\bigl(f'(x)\bigr)^2} \;=\; \frac{f(x)\, f''(x)}{\bigl(f'(x)\bigr)^2}.$$

Se $f'(x^*) \neq 0$ e $f''(x)$ é limitado, então para x próximo de x^* tem-se

$$|g'(x^*)| \;=\; \left|\frac{f(x^*)\, f''(x^*)}{\bigl(f'(x^*)\bigr)^2}\right|,$$

mas $f(x^*) = 0$. Assim, $\left|g'(x^*)\right| = 0 < 1$. Essa condição sugere que, quando não há singularidade ($f'(x^*) \neq 0$), o Método de Newton converge localmente e, de fato, costuma apresentar *convergência quadrática*.

4 Implementação Computacional

A seguir, exibimos um código em Python que implementa ambos os métodos e gera gráficos de comparação. A função considerada para teste é

$$f(x) = e^{-x} - x,$$

no intervalo $(0, 2)$. Comentamos brevemente os passos principais:

- *verificar_convergencia*: checa se $\max_{x\in[a,b]} |g'(x)| < 1$ e se $g(x)$ se mantém dentro de $[a, b]$.
- *ponto_fixo*: implementa genericamente a iteração $x_{n+1} = g(x_n)$.
- *aplicar_ponto_fixo_metodo1*: aplica o Método 1 varrendo diversos valores de λ.
- *aplicar_ponto_fixo_metodo2*: aplica o Método 2 (Newton) e verifica convergência local.

```python
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt

# Definição de x como símbolo
x = sp.symbols('x', real=True)
# Função de teste
f_expr = sp.exp(-x) - x
# Intervalo de interesse
intervalo = (0, 2)
tol = 1e-6
max_iter = 100

# Derivada de f para o Método 2
f_prime = sp.diff(f_expr, x)

# Conjunto de valores de lambda para o Método 1
lambdas = np.linspace(-2, 2, 40)
lambdas = lambdas[lambdas != 0]  # exclui lambda = 0

def verificar_convergencia(g, x, intervalo):
    """
    Verifica aproximadamente se |g'(x)| < 1 em [a,b]
    e se g(x) mapeia [a,b] em [a,b].
    """
    a, b = intervalo
    g_prime = sp.diff(g, x)
    g_prime_func = sp.lambdify(x, sp.Abs(g_prime), modules=['numpy'])
    x_vals = np.linspace(a+1e-6, b-1e-6, 1000)
    g_prime_vals = g_prime_func(x_vals)

    if np.max(g_prime_vals) < 1:
        # Verifica se g([a,b]) subset [a,b]
        g_func = sp.lambdify(x, g, modules=['numpy'])
        g_vals = g_func(x_vals)
        if np.all(g_vals >= a) and np.all(g_vals <= b):
            return True
    return False
```

```python

def ponto_fixo(g_func, x0, tol=1e-6, max_iter=100):
    """
    Itera x_{n+1} = g_func(x_n) até convergir ou
    atingir max_iter.
    """
    x_n = x0
    iteracoes = [x_n]
    for _ in range(max_iter):
        x_next = g_func(x_n)
        iteracoes.append(x_next)
        if abs(x_next - x_n) < tol:
            return x_next, iteracoes, True
        x_n = x_next
    return x_n, iteracoes, False

def aplicar_ponto_fixo_metodo1(f_expr, intervalo, x0=None, tol=1e-6, max_iter=100):
    """
    Aplica x_{n+1} = x_n - lambda * f(x_n) para vários lambdas.
    Retorna lista de dicionários com resultados.
    """
    x = sp.symbols('x', real=True)
    resultados = []
    a, b = intervalo
    if x0 is None:
        x0 = 0.5*(a+b)

    for lam in lambdas:
        g_expr = x - lam*f_expr
        if verificar_convergencia(g_expr, x, intervalo):
            g_func = sp.lambdify(x, g_expr, modules=['numpy'])
            raiz, iter_list, conv = ponto_fixo(g_func, x0, tol=tol,
            ↪  max_iter=max_iter)
            num_iter = len(iter_list) - 1
            resultados.append({
                'lambda': lam,
                'raiz': raiz,
                'iteracoes': iter_list,
                'num_iter': num_iter,
                'convergiu': conv
            })
        else:
            resultados.append({
                'lambda': lam,
                'raiz': None,
                'iteracoes': None,
                'num_iter': None,
                'convergiu': False
            })
    return resultados

def aplicar_ponto_fixo_metodo2(f_expr, intervalo, x0=None, tol=1e-6, max_iter=100):
    """
    Aplica x_{n+1} = x_n - f(x_n)/f'(x_n) (Método de Newton).
    Verifica convergência local.
    """
    x = sp.symbols('x', real=True)
    # g(x) = x - f(x)/f'(x)
```

```
    g_expr = x - f_expr/f_prime
    if x0 is None:
        a, b = intervalo
        x0 = 0.5*(a+b)

    if verificar_convergencia(g_expr, x, intervalo):
        g_func = sp.lambdify(x, g_expr, modules=['numpy'])
        raiz, iter_list, conv = ponto_fixo(g_func, x0, tol=tol, max_iter=max_iter)
        num_iter = len(iter_list) - 1
        return {
            'raiz': raiz,
            'iteracoes': iter_list,
            'num_iter': num_iter,
            'convergiu': conv
        }
    else:
        return None

# -------------------------------------------------------
# Execução e Gráficos
# -------------------------------------------------------
res_metodo1 = aplicar_ponto_fixo_metodo1(f_expr, intervalo)
res_metodo2 = aplicar_ponto_fixo_metodo2(f_expr, intervalo)

# Coleta de dados para plot de método 1
lambdas_validos = []
num_iters_m1 = []
for res in res_metodo1:
    if res['convergiu']:
        lambdas_validos.append(res['lambda'])
        num_iters_m1.append(res['num_iter'])

# Plot do número de iterações do Método 1 vs lambda
plt.figure(figsize=(10,5))
plt.plot(lambdas_validos, num_iters_m1, 'o-', label='Método 1')
if res_metodo2:
    plt.axhline(y=res_metodo2['num_iter'], color='r', linestyle='--', label='Método
    ↪  2')
plt.xlabel('Lambda')
plt.ylabel('Número de Iterações')
plt.title('Número de Iterações até Convergência')
plt.legend()
plt.grid(True)
plt.show()

# Análise mais detalhada com melhor lambda do Método 1
if res_metodo2:
    # Melhor resultado do Método 1 (menor número de iterações)
    best_res_m1 = min(
        [r for r in res_metodo1 if r['convergiu']],
        key=lambda r: r['num_iter']
    )
    it_m1 = best_res_m1['iteracoes']
    # Erro absoluto sucessivo
    err_m1 = [abs(it_m1[i+1] - it_m1[i]) for i in range(len(it_m1)-1)]

    it_m2 = res_metodo2['iteracoes']
    err_m2 = [abs(it_m2[i+1] - it_m2[i]) for i in range(len(it_m2)-1)]
```

```python

    fig, axes = plt.subplots(1, 2, figsize=(12, 5))

    # (a) Evolução das aproximações
    axes[0].plot(range(len(it_m1)), it_m1, marker='o', linestyle='-',
                 label=f"Método 1 (={best_res_m1['lambda']:.2f})")
    axes[0].plot(range(len(it_m2)), it_m2, marker='s', linestyle='-',
                 label='Método 2')
    axes[0].set_title('Evolução das Aproximações')
    axes[0].set_xlabel('Iteração')
    axes[0].set_ylabel('Aproximação de x')
    axes[0].grid(True)
    axes[0].legend()

    # (b) Erro absoluto (escala log)
    axes[1].semilogy(range(1, len(err_m1)+1), err_m1, marker='o', linestyle='-',
                     label=f"Método 1 (={best_res_m1['lambda']:.2f})")
    axes[1].semilogy(range(1, len(err_m2)+1), err_m2, marker='s', linestyle='-',
                     label='Método 2')
    axes[1].set_title('Erro Absoluto por Iteração (Escala Log)')
    axes[1].set_xlabel('Iteração')
    axes[1].set_ylabel('Erro')
    axes[1].grid(True, which='both', ls='--')
    axes[1].legend()
    plt.tight_layout()
    plt.show()
```

5 Demonstração Teórica e Resultados

Nesta seção, fazemos uma análise mais detalhada acerca da convergência de cada método (**Método 1** e **Método 2**). Em seguida, apresentamos e discutimos os resultados para o caso numérico considerado.

5.1 Prova de Convergência para o Método 1

Suponha que desejamos resolver $f(x) = 0$. Definimos

$$x_{n+1} = g(x_n) = x_n \; - \; \lambda\, f(x_n),$$

e analisamos a derivada de g:

$$g'(x) = 1 - \lambda f'(x).$$

Se existir um intervalo $[a, b]$ tal que:

1. $\forall x \in [a, b], \;\; g(x) \in [a, b]$, i.e. o método não *sai* do intervalo,
2. $\max_{x\in[a,b]} |g'(x)| < 1$,

então o Teorema do Ponto Fixo de Banach garante a convergência de x_n para o ponto fixo x^* (que satisfaz $g(x^*) = x^*$, ou seja, $f(x^*) = 0$).

Ordem de convergência

Para estimar a ordem, consideremos o erro $e_n = x_n - x^*$. Fazendo expansão de Taylor de f em torno de x^* e lembrando que $f(x^*) = 0$, temos

$$f(x_n) = f'(x^*)\, e_n + O(e_n^2).$$

Logo,

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

Portanto,

$$e_{n+1} = (1 - \lambda f'(x^*))\, e_n + O(e_n^2).$$

Isto mostra que a convergência do Método 1, quando converge, é *linear* (ordem 1), com taxa definida por $\Big|1 - \lambda f'(x^*)\Big|$.

5.2 Prova de Convergência para o Método 2 (Newton)

Para o **Método 2**, temos:

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

Seja x^* tal que $f(x^*) = 0$ e $f'(x^*) \neq 0$. Analisando a função

$$g(x) = x - \frac{f(x)}{f'(x)},$$

observa-se que

$$g'(x) = 1 - \frac{f'(x)\, f'(x) - f(x) f''(x)}{\big(f'(x)\big)^2} = \frac{f(x)\, f''(x)}{\big(f'(x)\big)^2}.$$

Dado que $f(x^*) = 0$, imediatamente obtemos $g'(x^*) = 0$. Isso sugere $\Big|g'(x^*)\Big| = 0 < 1$, de modo que o Método de Newton *localmente* converge. Mais ainda, essa derivada nula implica *convergência de ordem 2* (quadrática), pois a cada iteração o erro se reduz, em termos de magnitude, aproximadamente ao quadrado do erro anterior.

Convergência Quadrática

Supondo que $f'(x^*) \neq 0$ e f seja duas vezes continuamente diferenciável, podemos escrever a expansão de Taylor em torno de x^*:

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

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

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

Como $f'(x_n) = f'(x^*) + O(e_n)$, substituímos na fração e obtemos

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

Fazendo a simplificação, verifica-se que

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

Em termos de ordem de convergência, resulta

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

para algum $C > 0$. Esse fato caracteriza a *convergência quadrática* do Método de Newton.

5.3 Resultados Numéricos e Discussões

Para exemplificar, tomamos $f(x) = e^{-x} - x$ no intervalo $(0, 2)$. A raiz aproximada está próxima de 0.567143. No **Método 1**, variamos λ entre -2 e 2, excluindo $\lambda = 0$. A Figura 1 exibe como o número de iterações até convergir muda em função de λ.

Observa-se que:

- Alguns valores de λ não satisfazem $\max |g'(x)| < 1$ no intervalo. Nesses casos, o método não converge (deixamos *num_iter* = *None*).
- Há um intervalo de valores de λ para os quais o método é estável e converge, mas com número de iterações relativamente grande se λ for muito pequeno (pois o passo fica curto) ou muito grande (pois a função deixa de ser contrativa em boa parte do intervalo).

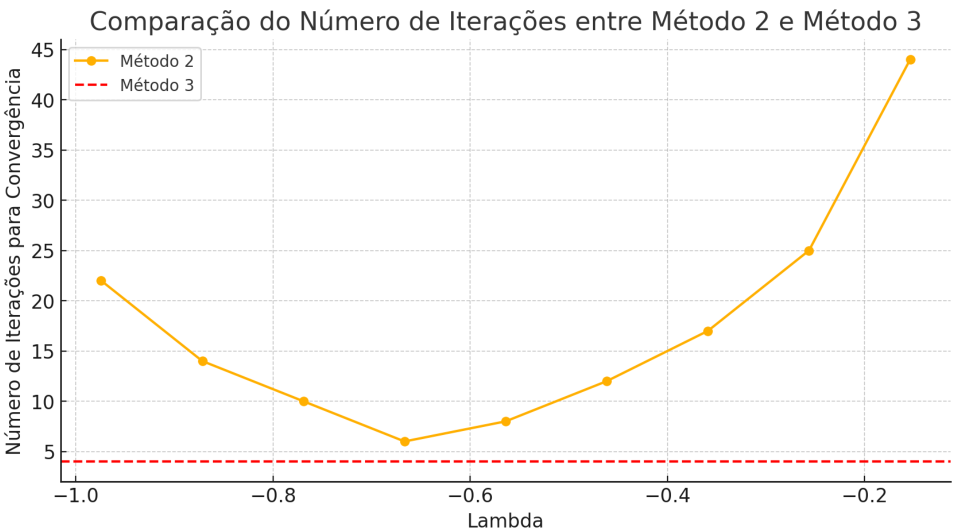
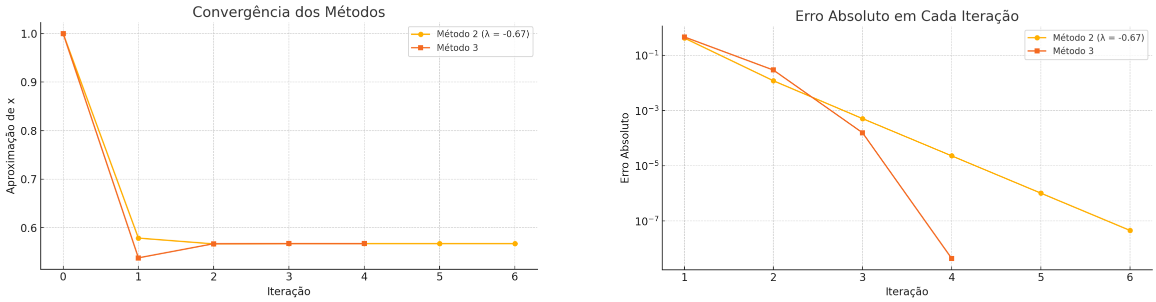


Figura 1: Número de iterações para convergência em função de λ (Método 1). A linha tracejada vermelha é o número de iterações do Método 2, que é independente de λ.

A Figura 2 exibe, no painel (a), a evolução das aproximações x_n para o melhor caso do Método 1 (i.e., o λ que necessitou menos iterações) em comparação com o Método 2 (Newton). O painel (b) mostra o erro absoluto sucessivo em escala logarítmica, evidenciando como o Método 2 *rapidamente* faz o erro cair.

Conclusão Numérica: Os testes reforçam a análise teórica de que o Método 1 possui convergência linear e sensível à escolha de λ. Já o Método 2 (Newton) apresenta convergência mais rápida (de segunda ordem) e não depende de parâmetro extra, mas exige o cálculo de derivada e pode falhar caso $f'(x^*) = 0$.



(a) Evolução das aproximações. (b) Erro absoluto (escala log).

Figura 2: Comparação entre o melhor caso do Método 1 e o Método 2 (Newton).

6 Conclusões Finais

Neste relatório, analisamos duas variações de métodos de ponto fixo para resolver $f(x) = 0$, com *ênfase na fundamentação teórica e comparação numérica*:

1. **Método 1**: $x_{n+1} = x_n - \lambda\, f(x_n)$, que depende de λ para garantir convergência local e costuma apresentar convergência *linear*.

2. **Método 2**: $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$ (Newton), apresentando convergência *quadrática* quando $f'(x^*) \neq 0$.

A análise teórica (Seção 5) embasa os experimentos numéricos. Pudemos constatar que o Método 1 convergirá bem apenas se λ for escolhido de maneira apropriada de modo que $\left|1 - \lambda f'(x)\right| < 1$. Já o Método 2 independe de λ e, sob hipóteses regulares sobre f, converge mais rapidamente.

No entanto, o Método 2 exige o cálculo de derivada e, em casos onde $f'(x^*) \approx 0$, pode apresentar instabilidade ou falha de convergência. Em aplicações práticas, muitas vezes usa-se uma etapa de *busca de passo* (*line search*) ou regularizações para contornar problemas de divergência ou aproximações ruins.

7 Referências

Referências

[1] J. Kiusalaas. *Numerical Methods in Engineering with Python*. Cambridge University Press, 2005.

[2] N. B. Franco. *Cálculo Numérico*. Pearson Prentice Hall, 2007.

[3] S. Lynch. *Dynamical Systems with Applications Using Python*. Springer, 2018.

[4] M. A. G. Ruggiero, V. L. Lopes. *Cálculo Numérico: aspectos teóricos e computacionais.* Makron Books, 1997.