



**Universidade Federal do Pará
Campus Universitário Castanhal
Faculdade de Matemática
Licenciatura em Matemática**

Trabalho de Conclusão de Curso

**Classes de Métodos Numéricos não
Convencionais para Determinação de
Raízes de Funções**

Henrique Pinheiro Galvão

Curuçá
2024

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

G182c Galvão, Henrique Pinheiro.
Classes de Métodos Numéricos não Convencionais para
Determinação de Raízes de Funções / Henrique Pinheiro Galvão. —
2024.
xii,66 f. : il. color.

Orientador(a): Prof. Dr. Valdelírio da Silva E Silva
Trabalho de Conclusão de Curso (Graduação) - Universidade
Federal do Pará, Campus Universitário de Castanhal, Faculdade de
Matemática, Castanhal, 2024.

1. Raízes de funções. 2. Métodos clássicos. 3. Métodos
numéricos não convencionais. I. Título.

CDD 518.2

Classes de Métodos Numéricos não Convencionais para Determinação de Raízes de Funções

Graduação

03/2019 – 02/2024

Submissão 29/01/2024

Defesa 07/02/2024

Versão Final 15/02/2024

Universidade Federal do Pará
Campus Universitário Castanhal
Faculdade de Matemática

Henrique Pinheiro Galvão

henriquehq27@gmail.com

Licenciando em Matemática

UFPA-Castanhal

Banca Examinadora:

Prof. Valdelírio da Silva e Silva

Orientador

Prof. Dr. Nildsen Fernando Lisboa da Silva

Membro da Faculdade

Prof. MSc. Romário Silva Duarte

Membro Externo

Dedicatória

Dedico este trabalho à minha família.

Agradecimentos

Primeiramente, agradeço a Deus por me iluminar durante essa caminhada acadêmica. Obrigado por me proporcionar coragem nesse percurso, pois nunca foram por méritos próprios.

Aos familiares, especialmente minha Mãe (Eduarda Campos), meu Pai (Maurício Pinheiro), meus Irmãos (Rafael Pinheiro), (Maurício Pinheiro Filho) e (Raone Pinheiro).

Ao meu orientador Valdelírio da Silva e Silva por toda contribuição, pois sem suas valiosas sugestões não seria possível esse trabalho.

Aos professores da Faculdade por toda contribuição ao longo desse percurso.

Aos amigos da turma pelo companheirismo acadêmico, em especial ao Deivid Tavares e Matheus Almeida.

Enfim, à todos aqueles que contribuíram de alguma forma para minha formação.

Epígrafe

“Há dois labirintos do espírito humano: Um respeita à composição do contínuo, o outro à natureza da liberdade; Ambos têm origem no mesmo infinito.”

Gottfried Wilhelm Von Leibniz

Resumo

O presente trabalho aborda alguns métodos numéricos não convencionais para determinar raízes de equações não lineares de uma variável, sendo eles: Illinois, Pegasus, Müller, Brent, Halley, Chebyshev e Newton–RM. A pesquisa ainda relaciona alguns aspectos que os métodos numéricos clássicos enfrentam na convergência do problema. O trabalho apresenta explicações relacionando as desvantagens dos métodos clássicos e como abordagens mais sofisticadas resolvem esses problemas, bem como as simulações geométricas comparativas. No decorrer desta pesquisa será apresentada a descrição dos métodos com breves abordagens históricas. Será feita a análise dos dados obtidos levando em consideração a quantidade de iterações e o erro relativo aproximado, bem como o tempo gasto.

Palavras-chaves: Raízes de Funções, Métodos Numéricos Clássicos, Métodos Numéricos não Convencionais.

Abstract

The present work addresses some unconventional numerical methods for determine roots of nonlinear equations of one variable, they are: Illinois, Pegasus, Müller, Brent, Halley, Chebyshev and Newton–RM. Research also lists some aspects that classical numerical methods face in the convergence of the problem. The work presents explanations relating the disadvantages of classical methods and how more sophisticated approaches solve these problems, as well as comparative geometric simulations. In the course of this research, a description of the methods will be presented with brief historical approaches. The data obtained will be analyzed taking into account the number of iterations and the approximate relative error, as well as the time spent.

Keywords: Roots of Functions, Classical Numerical Methods, Unconventional Numerical methods .

Lista de Figuras

1.1	Funções com características de Concavidade/Convexidade e Multiplicidade .	15
2.1	Simulação geométrica de divergência do método de Newton em pontos de inflexão	18
2.2	Simulação geométrica das iterações de Newton para raízes múltiplas	19
3.1	Simulações geométricas das iterações da Falsa posição e Illinois	22
3.2	Simulações geométricas das iterações da Falsa posição e Pegasus	23
3.3	Simulações geométricas das iterações da Falsa posição e Müller	26
3.4	Parábolas ajustadas aos mesmos pontos	28
3.5	Simulação geométrica do método de Brent	29
3.6	Simulações geométricas das iterações de Newton e Halley	31
3.7	Simulações geométricas de Halley em uma raiz próxima a pontos de inflexão	32
3.8	Simulações geométricas das iterações de Newton e Chebyshev	34
4.1	Gráfico da função $f(x) = \frac{1}{x} + \ln(x) - 100$	39
4.2	Aproximações e curvas do erro relativo dos métodos	40
4.3	Gráfico da função $f(x) = x^3 - x^2 - x - 1$	41
4.4	Aproximações e curvas do erro relativo dos métodos	42
4.5	Gráfico da função $f(x) = 5x^3 + x^2 - e^{1-2x} + \cos(x) + 20$	43
4.6	Aproximações e curvas do erro relativo dos métodos.	44
4.7	Gráfico da função $f(x) = x^3 - x^2 - x + 1$	45
4.8	Aproximações e curvas do erro relativo dos métodos	46

Lista de Tabelas

3.1	Família de fatores γ	24
4.1	Características dos métodos	37
4.2	Desempenho dos métodos em $f(x) = \frac{1}{x} + \ln(x) - 100$	39
4.3	Desempenho dos métodos em $f(x) = x^3 - x^2 - x - 1$	42
4.4	Desempenho dos métodos em $f(x) = 5x^3 + x^2 - e^{1-2x} + \cos(x) + 20$	44
4.5	Desempenho dos métodos em $f(x) = x^3 - x^2 - x + 1$	45
4.6	Desempenho dos métodos em f_1	47
4.7	Desempenho dos métodos em f_2	48
4.8	Desempenho dos métodos em f_3	48

Sumário

Introdução	11
1 Prioris	13
1.1 Método Gráfico	13
1.2 Métodos Iterativos	14
1.3 Funções com Anomalias	14
2 Alguns Métodos Clássicos	16
2.1 Método da Bissecção	16
2.2 Método de Newton (<i>Newton-Raphson</i>)	17
2.3 Método da Falsa Posição (<i>Regula Falsi</i>)	19
3 Métodos não Convencionais	21
3.1 Métodos Baseados em Interpolação Linear	21
3.2 Métodos Baseados em Interpolação Quadrática	25
3.3 Métodos com Derivadas de Segunda Ordem	30
4 Testes e Resultados	37
4.1 Desempenho Computacional	38
4.2 Métodos Clássicos × Métodos em Interpolação Linear Modificados	38
4.3 Métodos Clássicos × Métodos em Interpolação Quadrática	41
4.4 Métodos Clássicos × Métodos com Derivadas de Segunda Ordem	43
4.5 Comparação Geral entre os Métodos	46
Considerações Finais	50
Referências	51
Apêndice	53
A Códigos confeccionados para o desenvolvimento do trabalho	54

Introdução

Muitos fenômenos naturais conhecidos são modelados acerca de parâmetros não lineares (com funções não polinomiais), isto é, problemas que em sua estrutura apresentam tal complexidade de obter-se uma solução, não havendo métodos analíticos para encontrar tal resposta. Vários modelos matemáticos de áreas, como engenharia, biologia e economia ocorre a necessidade de determinar um termo que será a solução. Recorre-se aos métodos numéricos que, em suma, são processos iterativos com o objetivo de encontrar um número suficientemente próximo da solução exata.

Neste presente trabalho são propostos métodos não convencionais decorrente dos conhecidos na literatura e que não são apresentados em geral a nível de graduação. São eles: Método de Pegasus, Illinois, Muller, Brent, Chebyshev, Halley e Newton-RM. Esses métodos são variações ou modificações dos métodos mais conhecidos. Os métodos apresentados são baseados em três classes distintas: Interpolação linear, polinomial e derivadas de segunda ordem. A partir dessas novas classes de métodos é possível notar uma aceleração de convergência para aproximar zero de funções, pois os métodos tradicionais enfrentam alto custo computacional em funções que apresentam algum tipo de anomalia.

Os métodos de Illinois e Pegasus pertencem à classe de interpolação linear e são variações do método da Falsa Posição que contam com atualizações em seu processo iterativo com o objetivo de resolver problemas relacionados a retenção de extremos fixos presente no método clássico; os métodos de Müller e Brent, por sua vez, são algoritmos generalizados pertencentes à classe de interpolação quadrática, ou seja, o processo iterativo desses métodos são mediante parábolas; por fim, os métodos de Halley, Chebyshev e Newton-RM são métodos baseados em derivadas de segunda ordem que, em suma, apresentam modificações que aceleram a convergência para raiz em relação ao método de Newton tradicional.

O presente trabalho aborda métodos que são mais eficientes que os métodos tradicionais e que são pouco explorados ou utilizados no meio acadêmico elementar, ou seja, um único método não é capaz de solucionar todos os problemas, por esse motivo o assunto abordado é relevante, pois novos métodos são desenvolvidos para resolver as limitações dos métodos anteriores.

O trabalho está dividido em quatro capítulos, sendo o primeiro (capítulo 1) designado à

alguns breves conceitos importantes para o andamento do trabalho. O capítulo 2 apresenta de forma resumida os métodos clássicos e algumas vantagens e desvantagens. O terceiro capítulo (3) apresenta os métodos modificados que estão separados em seções, e que serão explanados breves introduções históricas e, posteriormente, o conceito geral dos métodos e algumas simulações geométricas entre os métodos estudados. No capítulo quarto (4) serão apresentados testes numéricos em algumas funções com características distintas, com o intuito de comparar os métodos estudados. Por fim, numa última seção (4.5) são feitas as considerações finais deste trabalho.

A determinação de raízes costuma ser associada instantaneamente à famosa “fórmula de Bhaskara” pelos alunos em geral, pois é conhecida como método fechado para encontrar zeros de funções quadráticas. Nesse caso específico, a fórmula analítica resolve problemas de natureza polinomial de grau 2, mas existem fenômenos não lineares que são modelados por funções ditas transcendentes (não polinomiais) e, que conseqüente, não existem sempre ferramentas analíticas para determinar tal solução. A existência de métodos para funções polinomiais até o 4° são conhecidos na literatura. Para solução polinomial de grau 3° utiliza-se o método de Cardano (sob condições especiais nos coeficientes); Nas funções até grau 4° tem-se o método de Ferrari ([NASCIMENTO et al., 2015](#)).

Considere o problema em determinar as raízes da equação

$$20e^x + \frac{5}{x}(e^x - 1) - 0,7 = 0.$$

Não existe fórmula analítica para determinar uma solução exata para essa equação. A determinação de zeros para equações não lineares está presente em diversas áreas de aplicação (projetos, engenharia, crescimento populacional, etc). Portanto há necessidade de se obter uma solução aproximada.

1.1 Método Gráfico

O método mais básico para estimar a solução de uma equação $f(x) = 0$ é construir um gráfico e observar onde essa função intercepta o eixo x . Esse ponto representa uma aproximação da raiz, mas será uma estimativa dita como grosseira, pois o critério será apenas visual.

A construção do gráfico pode ser feita utilizando softwares como `matlab`, `python` ou `GeoGebra`, mas técnicas gráficas são limitadas, pois fornecem resultados distantes do real, no entanto, a interpretação gráfica proporciona entender as características da função, e escolher o método numérico mais apropriado ([CHAPRA, 2013](#)).

1.2 Métodos Iterativos

Os métodos iterativos são algoritmos de repetição que servem para aproximar numericamente a solução de um problema, dessa forma, esse processo é denominado de iteração. O principal objetivo das iterações é encontrar a cada passo uma melhor aproximação para o problema. Segundo Souza (2015) existem critérios que devem ser definidos para cada problema a fim de encontrar uma solução aproximadamente razoável, mas que evite um número excessivo de iterações.

1.3 Funções com Anomalias

Os métodos iterativos tradicionais sob condições favoráveis funcionam perfeitamente, no entanto, quando a função apresenta complicações no intervalo em que a raiz se encontra, seu desempenho é afetado. Dessa forma, o número de iterações costuma ser elevado para métodos dessa classe.

Existem problemas intervalares que dificultam o processo iterativo dos métodos clássicos na busca pela raiz. Destacam-se os problemas de *convexidade/concavidade* e *multiplicidade* da raiz. Na figura 1.1 são exemplos de funções com raízes em intervalos com essas características.

Seja I um intervalo de $\mathbb{R}$ e $f : I \rightarrow \mathbb{R}$ uma função. Dizemos que f é convexa em I , ou seja, tem concavidade voltada para cima em I (figura 1.1-a), se

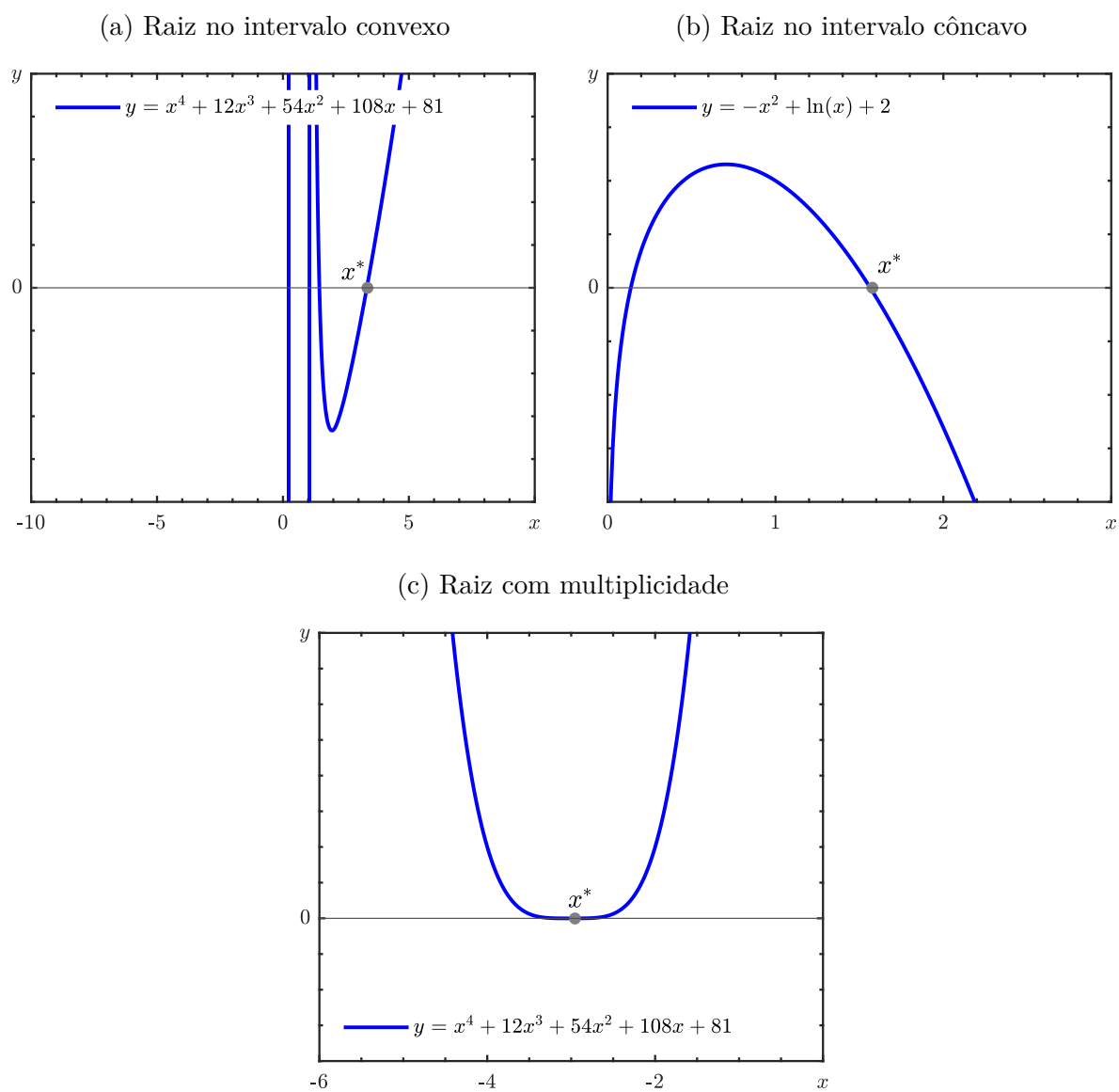
$$f(\lambda b(1 - \lambda)a) \leq \lambda f(b) + (1 - \lambda)f(a), \quad \text{tomando } \lambda \in [0, 1] \quad (1.1)$$

com $a, b \in I$. A função f é dita côncava no intervalo I , quando tem a concavidade voltada para baixo (figura 1.1-b), se

$$f(\lambda b(1 - \lambda)a) \geq \lambda f(b) + (1 - \lambda)f(a), \quad \text{tomando } \lambda \in [0, 1] \quad (1.2)$$

para cada $a, b \in I$.

Quanto as características gráficas de funções com raízes múltiplas (figura 1.1-c) é fácil perceber o “achatamento” no intervalo em que a raiz se encontra.

Figura 1.1: Funções com características de Concavidade/Convexidade e Multiplicidade

Fonte: Autoria própria

Alguns Métodos Clássicos

Neste capítulo explanaremos alguns métodos clássicos que são apresentados na literatura. Embora funcionem corretamente sob certas condições, existem desvantagens quando a função não é bem comportada ou apresenta alguns tipos de anomalias. Diante disso, serão apresentadas desvantagens significativas na obtenção de uma solução razoável para equações não lineares.

Todos os programas dos métodos numéricos apresentados aqui foram confeccionados por autoria própria e disponibilizados no apêndice A. Eles foram construídos na linguagem `matlab` sob o formato de `functions` (subprogramas), os quais podem ser facilmente executados também no `octave`. Além disso, todas as figuras de discussão dos métodos também foram criadas no `matlab` e sendo os códigos-fontes disponíveis para qualquer interessado mediante o contato com o autor deste trabalho.

2.1 Método da Bisseção

O método da bisseção é a primeira ferramenta apresentada para resolver problemas dessa natureza, pois apresenta um algoritmo simples para obtenção de uma raiz. O método baseia-se necessariamente em uma divisão intervalar, tendo a cada iteração um novo subintervalo utilizando o ponto médio

$$x_{k+1} = \frac{a_k + b_k}{2}. \quad (2.1)$$

Para que o método funcione corretamente, é necessário conhecer o intervalo em que a raiz se encontra; e, além da função ter que ser contínua nesse intervalo ela deve ter sinais opostos nos seus extremos. O método permite determinar o número máximo de iterações necessárias para calcular a raiz para uma certa tolerância a partir do intervalo em que $\bar{x}$ está (FILHO, 2018).

Teorema 1 (Teorema de Bolzano). *Seja $F(x)$ uma função contínua no intervalo $[a, b]$. Se $F(a) \cdot F(b) < 0$; então a função $F(x)$ tem pelo menos um zero no intervalo $[a, b]$.*

Uma vantagem significativa do método da bisseção refere-se a sua convergência garantida sob as condições do teorema (1), no entanto, uma desvantagem predominante é sua convergência lenta para solução. Isso ocorre, pois a cada iteração do método, o intervalo é reduzido pela metade (adotando-se intervalo de extremos com imagens de sinais contrários) e, conseqüentemente, gerando alto custo computacional que compromete seu desempenho na busca pela raiz.

A ordem de convergência linear pode ser entendida da seguinte maneira. Adotando-se um intervalo inicial, com amplitude $[b_k - a_k]$ que contém a raiz de $f(x) = 0$. Esse comprimento do intervalo é uma estimativa do erro, onde se define como: $\mathbf{e}_k = x_k - \bar{x}$, com $x_k = (a_k + b_k)/2$. Depois da primeira iteração, tanto o intervalo quanto o erro são reduzidos pela metade e, dessa forma, $|\mathbf{e}_1| = |\mathbf{e}_0|/2$, e assim por diante. Generalizando, temos:

$$|\mathbf{e}_{k+1}| = \frac{|\mathbf{e}_k|}{2}, \text{ fazendo } k = 0, 1, \dots$$

No geral, o método da bisseção é mais custoso do que métodos mais avançados, como o método de Newton, que geralmente converge mais rapidamente para a raiz da função sob condições ideais. O método da bisseção confeccionado para este trabalho apresenta-se em [A.1](#).

2.2 Método de Newton (*Newton-Raphson*)

O método de Newton é o mais clássico na busca de zeros de equações não lineares. Foi proposto por Isaac Newton em 1669 e depois por Joseph Raphson em 1690.

No método de Newton, dado um chute inicial $x_0 \in \mathbb{R}$, o método gera uma sequência $\{\mathbf{x}_k\}$ que converge para a raiz procurada. É conhecido também como método das tangentes, pois suas iterações representam retas que tangenciam o gráfico de f no ponto $[x_k, f(x_k)]$ ([FERREIRA et al., 2013](#)).

Segundo [Franco \(2006\)](#), o método constrói uma sequência que converge quadraticamente para a raiz $\bar{x}$ por meio de um modelo linear da função em dado ponto x_k , determinando uma nova aproximação x_{k+1} . [Ferreira et al. \(2013\)](#) explica que podemos obter o zero de f através do polinômio de Taylor de primeiro grau, isto é, Newton utiliza o truncamento linear de Taylor para aproximar suas raízes. Dessa forma, temos:

$$p(x) \approx f(\bar{x}) + f'(\bar{x})(x - \bar{x}), \tag{2.2}$$

onde o zero de $p(x)$ é:

$$x = \bar{x} - \frac{f(\bar{x})}{f'(\bar{x})}. \quad (2.3)$$

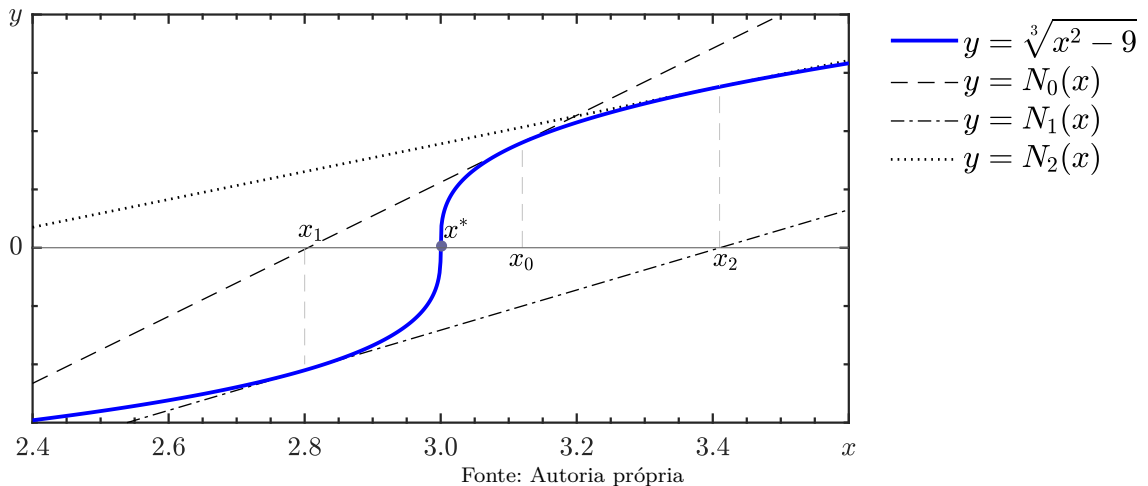
Definição 1. Dada uma função f real ou complexa, derivável com $f'(x_k) \neq 0$, define-se a função de Newton associada a $N_f(x)$:

$$N_f(x) = x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \dots \quad (2.4)$$

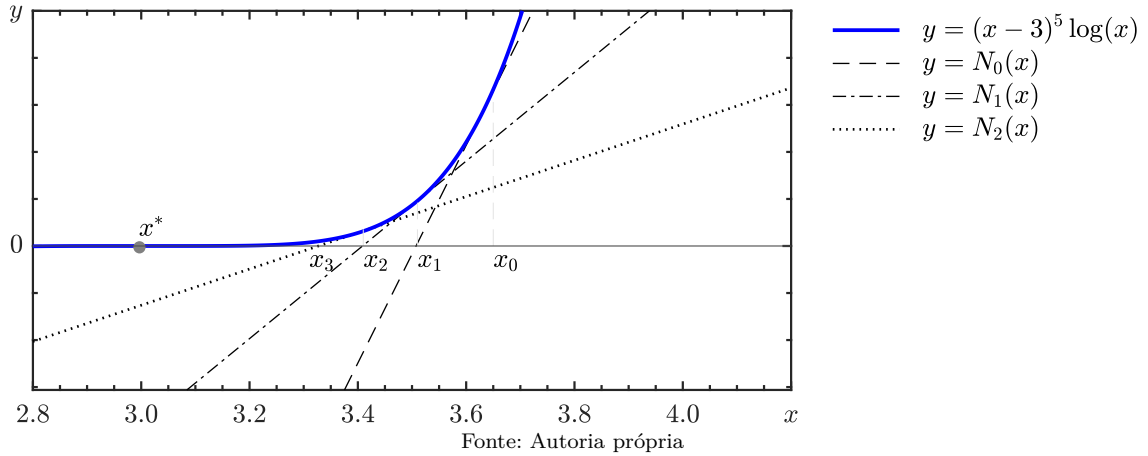
Assim, o método de Newton funciona partindo de um ponto inicial x_0 para a solução de f construindo uma sequência de aproximações, no entanto, se o ponto inicial é muito distante da solução o método diverge ou converge para outra raiz. Além disso, duas desvantagens significativas comprometem a eficiência do método, tanto no sentido de aproximação para a raiz, quanto na velocidade de sua convergência.

A primeira desvantagem está ilustrada na figura 2.1, que mostra as três primeiras aproximações para a função $f(x) = \sqrt[3]{x^2 - 9}$, explicitando que o método nunca convergirá para a raiz. Isso fica explícito, pois as retas $N_0(x)$, $N_1(x)$ e $N_2(x)$ que tangenciam f criam uma sequência de aproximações $(x_0, x_1, x_2, \dots, x_k)$ que se afastam da raiz. Isso mostra a deficiência do método em pontos próximos a regiões de inflexão, isto é, quando a $f''(x) = 0$ (CHAPRA; CANALE, 2011).

Figura 2.1: Simulação geométrica de divergência do método de Newton em pontos de inflexão



A segunda desvantagem se ilustra na figura 2.2. Apresentamos as primeiras iterações na busca por uma raiz múltipla, pois a função $f(x) = (x - 3)^5 \log(x)$ tem o 3 com multiplicidade 5.

Figura 2.2: Simulação geométrica das iterações de Newton para raízes múltiplas

É notável que as retas $N_0(x)$, $N_1(x)$ e $N_2(x)$ que tangenciam o gráfico de f , se aproximam de pontos onde a curva é quase constante. Em casos dessa natureza, o método de Newton tem sua velocidade de convergência reduzida, isto é, quando as aproximações tendem a valores próximo de zero o método passa a convergir com taxas lineares e, dessa forma, o método falha quando $f'(x) = 0$ visto que a definição (1) restringe essa característica!

É importante destacar que a escolha do x_0 implica no aumento ou diminuição na quantidade de iterações neste caso, isto é, um chute inicial mais próximo da raiz teria um comportamento gráfico diferente desta simulação.

O método de Newton foi programado por `function` e está registrado em [A.7](#)

2.3 Método da Falsa Posição (*Regula Falsi*)

A abordagem do método da falsa posição é similar ao método da secante, que por sua vez, resolve a deficiência do método de Newton referente a avaliação de $f'(x)$ em seu processo iterativo. A modificação ocorre mediante a substituição de $f'(x)$ pelo quociente das diferenças:

$$f'(x_k) \cong \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}. \quad (2.5)$$

Substituindo (2.5) em (2.4) obtemos:

$$x_{k+1} = x_k - \frac{f(x_k)}{\frac{f(x_k) - f(x_{k-1})}{(x_k - x_{k-1})}} \Rightarrow x_k - \frac{(x_k - x_{k-1})f(x_k)}{f(x_k) - f(x_{k-1})},$$

realizando algumas manipulações na expressão anterior, obtemos a fórmula iterativa do método da secante:

$$x_{k+1} = \frac{x_k f(x_{k-1}) - x_{k-1} f(x_k)}{f(x_{k-1}) - f(x_k)}, \quad k = 0, 1, \dots \quad (2.6)$$

Nesse sentido, o método da falsa posição é considerado uma modificação do método da secante, pois consiste em tomar duas aproximações iniciais x_{k-1} e x_k com a adição do **Teorema de Bolzano** (1) que sugere $f(x_{k-1})$ e $f(x_k)$ tenham sinais opostos, garantindo que a raiz esteja sempre isolada (FRANCO, 2006). Partindo de um intervalo $[x_{k-1}, x_k]$ que cumpram as condições do teorema (1), e utilizamos a fórmula iterativa do método da secante.

Segundo Palenzuela et al. (2019) teremos sempre a raiz procurada no intervalo desejado, pois assim como o método da bisseção, nossa convergência está assegurada pelo teorema (1), e assim podemos determinar o critério de parada mais adequado.

O método é intuitivamente simples comparado ao de Newton, mas a condição do teorema (1) garante convergência para raízes reais simples. No entanto, quando a função f apresenta anomalias de concavidade e convexidade no intervalo em que a raiz se encontra, sua velocidade de convergência diminui. Palenzuela et al. (2019, p. 21) explica que:

Portanto, mais uma vez temos um método simples que dificilmente requer quaisquer condições iniciais além das do Teorema de Bolzano. A velocidade de convergência é reduzida a um nível linear sempre que a função f no intervalo é côncava ou convexa porque uma das extremidades do intervalo permanece fixa ao longo do processo, torna-se um impasse, o que atrasa muito o método. (PALENZUELA et al., 2019, p. 21) (Tradução livre)

Teorema 2 (Teorema da convergência linear do método da falsa posição). *Seja f tal que $f(x_{k-1}) \cdot f(x_k) < 0$ com único zero $\bar{x}$ em $[x_{k-1}, x_k]$, se f for convexa ou concava em $[x_{k-1}, x_k]$ (ou seja $f'' > 0$ ou $f'' < 0$ respectivamente) então o método da falsa posição converge e é linear $\lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k} = M$, com $0 < M < 1$ obtida da fórmula $M = 1 - \frac{f'(\bar{x})}{f'(w)}$ com $w \in [\bar{x}, x_k]$, se f for convexa ou $w \in [x_{k-1}, \bar{x}]$, se f for concava.*

A retenção do ponto fixo nas iterações desse método em funções côncavas e convexas representa aumento de iterações! A aproximação para raiz procurada é gerada através de uma reta $g(x)$ que intercepta o eixo x . Então temos que $g(x) > f(x)$ ou $g(x) < f(x)$ no intervalo da raiz, uma relação de convexidade ou concavidade é tida, respectivamente.

Em suma, o método da falsa posição converge globalmente para raízes simples, mas a velocidade de convergência é reduzida drasticamente na presença de raízes múltiplas e anomalias referentes às característica da função no intervalo em que a raiz se encontra.

O método da falsa posição está codificado pela function A.2.

Métodos não Convencionais

Nesse capítulo abordaremos métodos modificados que contam com certas atualizações para resolver problemas como, o alto número de iterações em funções com características anômalas.

3.1 Métodos Baseados em Interpolação Linear

Essa classe de métodos utiliza polinômios lineares para aproximar raízes de equações, ou seja, em termos gráficos a representação dessa estimativa será uma reta que intercepta o eixo x . De acordo com [Filho \(2018\)](#) a velocidade de convergência para uma determinada raiz pode ser aumentada aproximando f por um polinômio de grau 1 em um intervalo apropriado.

3.1.1 Método de Illinois

O método de Illinois, o qual está codificado na [function A.3](#), é uma modificação do clássico método da falsa posição, proposto por [Dowell & Jarratt \(1971\)](#). Os autores acreditam que originalmente a ideia surgiu no centro de informática da Universidade de Illinois por volta de 1950, daí o nome proposto para o método.

A principal característica dessa modificação, consiste em solucionar o problema dos extremos fixos que ocorre no método tradicional da falsa posição. O procedimento para encontrar a melhor aproximação da raiz é similar ao método já apresentado no capítulo anterior, que é dado por [\(2.6\)](#) como:

$$x_{k+1} = \frac{x_k f(x_{k-1}) - x_{k-1} f(x_k)}{f(x_{k-1}) - f(x_k)}, \quad k = 0, 1, \dots$$

Existem duas situações que são parte do algoritmo de Illinois:

- (i) se $f(x_k) \cdot f(x_{k+1}) < 0$ então $[x_{k-1}, f(x_{k-1})]$ é substituído por $[x_k, f(x_k)]$;
- (ii) se $f(x_k) \cdot f(x_{k+1}) > 0$ então $[x_{k-1}, f(x_{k-1})]$ é substituído por $\left[x_{k-1}, \frac{f(x_{k-1})}{2}\right]$.

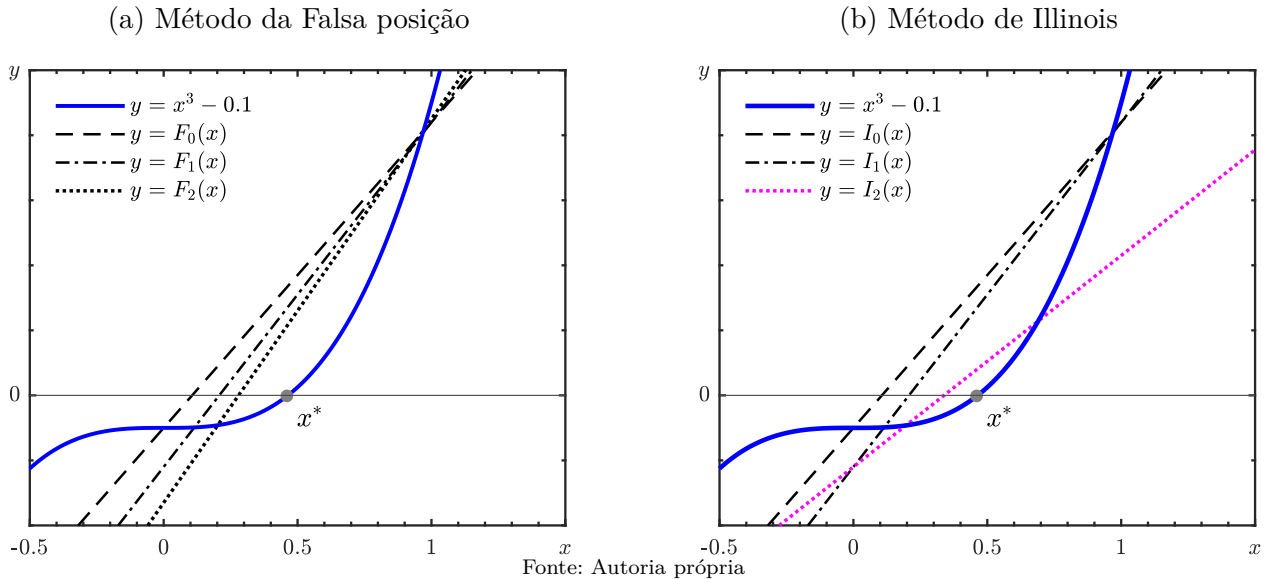
Assim, estamos relacionando a modificação quando ocorre a situação **(ii)**. Dessa forma, a fórmula iterativa é dada por:

$$x_{k+2} = \frac{x_{k+1} \frac{f(x_{k-1})}{2} - x_{k-1} f(x_{k+1})}{\frac{f(x_{k-1})}{2} - f(x_{k+1})}, \quad k = 0, 1, \dots \quad (3.1)$$

Segundo Palenzuela et al. (2019) esse cálculo modificado do novo ponto x_{k+2} , evita a retenção em um dos extremos que fica fixo no método da falsa posição.

Na figura 3.1 podemos observar que no método tradicional **(a)**, as retas iteradoras $F_0(x)$, $F_1(x)$ e $F_2(x)$ ficam fixas em um dos extremos do intervalo, enquanto no método de Illinois **(b)** a reta iteradora $I_2(x)$ (em magenta) tem seu ponto fixo reduzido e, consequentemente, deslocado em um ponto fora da curva. Isso ocorre porque o ponto $f(x_{k-1})$ é substituído pelo passo modificado $\frac{f(x_{k-1})}{2}$.

Figura 3.1: Simulações geométricas das iterações da Falsa posição e Illinois



O método da falsa posição converge lentamente quando a solução de f se encontra próximo de regiões de concavidade ou convexidade conforme o Teorema da convergência linear do método da falsa posição (2), pois o ponto mais distante da solução fica fixo e, em muitos casos, pode ser mais lento que o método da bisseção.

Diante disso, o método de Illinois reduz esse ponto fixo pela metade, e consequentemente define outro ponto de retenção para próxima iteração x_{k+3} .

3.1.2 Método de Pegasus

Os autores [Dowell & Jarratt \(1972\)](#) apresentam uma nova modificação que mostra convergência assintótica superior ao método da falsa posição, onde suas características de evitar as retenções em extremos fixos são as mesmas do método de Illinois.

Assim, observamos que o método de Pegasus (codificado na [function A.4](#)) é uma variante do método de Illinois, pois segue o mesmo princípio iterativo, no entanto, agora existe um valor γ que multiplica $f(x_{k-1})$ com objetivo de otimizar a velocidade de convergência.

$$x_{k+1} = \frac{x_k[\gamma f(x_{k-1})] - x_{k-1}f(x_k)}{[\gamma f(x_{k-1})] - f(x_k)}, \quad \gamma = \frac{f(x_k)}{f(x_{k+1}) + f(x_k)}, \quad 0 < \gamma < 1, \quad (3.2)$$

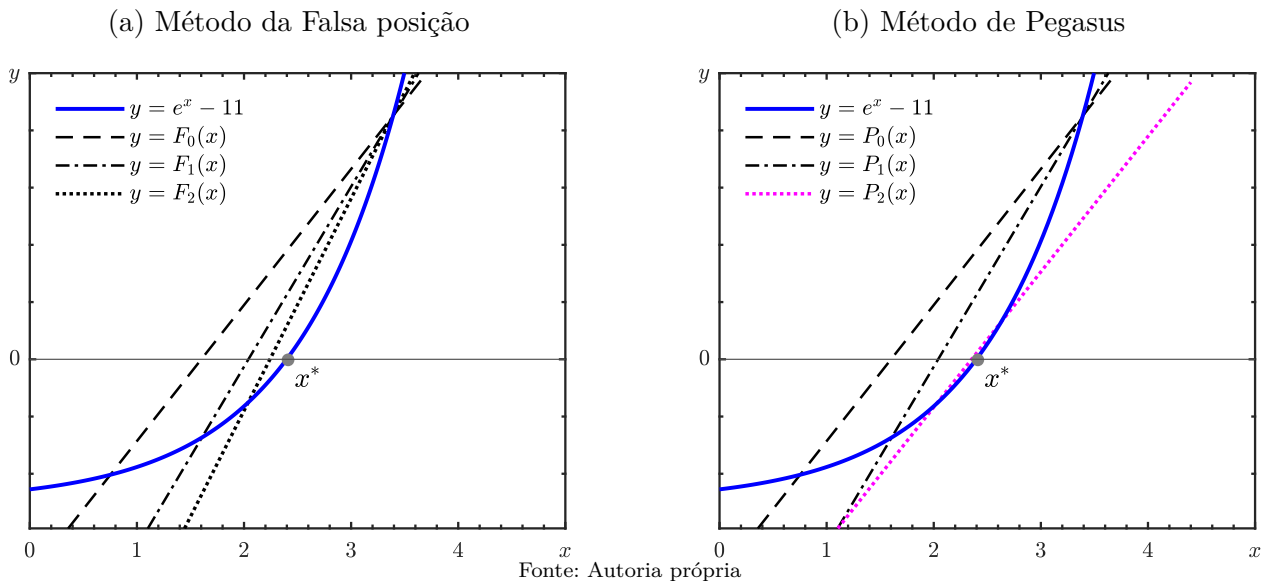
fazendo $k = 0, 1, \dots$

Diferente do método anterior, o valor de γ é adaptativo a cada passo do processo de busca pela raiz. Então segue a mesma situação no segundo passo do método.

- (i) se $f(x_k) \cdot f(x_{k+1}) < 0$ então $[x_{k-1}, f(x_{k-1})]$ é substituído por $[x_k, f(x_k)]$;
- (ii) se $f(x_k) \cdot f(x_{k+1}) > 0$ então $[x_{k-1}, f(x_{k-1})]$ é substituído por $\left[x_{k-1}, \frac{f(x_{k-1})f(x_k)}{f(x_{k+1}) + f(x_k)}\right]$.

Dessa forma, estamos reduzindo $f(x_{k-1})$ mediante a um fator γ para evitar retenções em um dos extremos fixos. Essa redução de $f(x_{k-1})$ implica em uma melhor aproximação para a raiz procurada e, conseqüentemente, diminuição em suas iterações.

Figura 3.2: Simulações geométricas das iterações da Falsa posição e Pegasus



Na figura 3.2 mostramos que geometricamente o método de Pegasus (a) tem a mesma

ideia do método de Illinois, ou seja, a reta $P_2(x)$ (em magenta) tem seu extremo fixo reduzido mediante ao fator γ .

Os fatores dos métodos de Illinois e Pegasus são apenas dois de uma família de parâmetros γ , isto é, existem outros valores que reduzem o ponto $f(x_{k-1})$ como a mesma eficiência ou até maior. Na tabela 3.1 são apresentados fatores com outra estrutura que podem ser substituídos nas **functions** A.3 ou A.4.

Tabela 3.1: Família de fatores γ

γ_n	FATOR
γ_1	$\frac{f(x_{k-1}) - f(x_k)}{2}$
γ_2	$\frac{f(x_{k-1}) - f(x_{k+1})}{2 + f(x_{k+1})/f(x_k)}$
γ_3	$\frac{f(x_{k-1}) - f(x_{k+1})}{[m + f(x_{k+1})/f(x_k)]^2}, 1 < m < 2$
γ_4	$\frac{f(x_{k-1}) + f(x_{k+1})}{[2 + f(x_{k+1})/f(x_k)]^2}$
γ_5	$\frac{f(x_{k-1})}{[1 + f(x_{k+1})/f(x_k)]^2}$
γ_6	$\frac{f(x_{k-1}) - f(x_{k+1})}{4}$
γ_7	$\frac{f(x_{k+1}) \cdot f(x_{k-1})}{f(x_k) + f(x_{k+1})}$

Fonte: Elaborada pelo autor mediante Galdino (2011)

Diante disso, fica evidênte a existência de métodos numéricos modificados que ainda são pouco conhecidos no meio acadêmico, pois o método da falsa posição tem outras abordagens como visto no método de Illinois e Pegasus, que teoricamente deixam o processo iterativo mais eficiente na busca pela solução de f . Existem outros métodos baseados em ponto fora da curva, os quais utilizam aspectos de diferenças divididas¹ em suas fórmulas iterativas.

¹São conhecidos como métodos de Anderson & Björck, que utilizam operadores de diferenças divididas em sua fórmula iterativa

3.2 Métodos Baseados em Interpolação Quadrática

A classe de métodos baseados em interpolação quadrática consiste em aproximação zeros de equações não lineares através de parábolas que interceptam o eixo x . [Filho \(2018\)](#) explica que aproximar as raízes de $f(x) = 0$ por um polinômio quadrático pode ser melhor do que um polinômio interpolador de grau linear. Serão apresentados dois métodos pertencentes a essa classe: método de Müller e o método de Brent.

3.2.1 Método de Müller

Esse método foi proposto pela primeira vez por David E. Müller em 1956. Basicamente o método consiste em aproximar raízes de equações através de polinômios quadráticos. Alguns autores associam à esse método como sendo uma extensão ou modificação do método da secante e falsa posição, pois o método de Müller necessita de três aproximações iniciais, que diferentemente dos anteriores, utilizam apenas duas aproximações ([BURDEN; FAIRES; BURDEN, 2016](#)).

Em [Obsieger \(2015\)](#) é descrito que o método utiliza os últimos três pontos para definir os coeficientes da parábola, e ainda considera sua intersecção com o eixo x como a próxima aproximação para raiz. Esses coeficientes determinam sua intersecção com o eixo x .

[Chapra & Canale \(2011, p. 144\)](#) explicam que:

O método consiste na determinação dos coeficientes da parábola que passa pelos três pontos. Esses coeficientes podem então ser substituídos na fórmula quadrática para se obter o ponto no qual a parábola intercepta o eixo x — isto é, a estimativa da raiz. O método se torna mais fácil escrevendo-se a equação da parábola em uma forma conveniente, $P(x) = a(x - x_k)^2 + b(x - x_k) + c$. ([CHAPRA; CANALE, 2011, p. 144](#))

Como citado acima, para determinar a aproximação, é utilizado a fórmula quadrática na equação da parábola na forma $P(x) = a(x - x_k)^2 + b(x - x_k) + c$. Os coeficientes² da equação acima podem ser encontrados seguindo as condições $P(x_k) = f(x_k)$, $P(x_{k-1}) = f(x_{k-1})$ e $P(x_{k-2}) = f(x_{k-2})$, assim temos que:

$$\begin{aligned} P(x_{k-2}) &= f(x_{k-2}) = a(x_{k-2} - x_{k-2})^2 + b(x_{k-2} - x_{k-2}) + c, \\ P(x_{k-1}) &= f(x_{k-1}) = a(x_{k-1} - x_{k-2})^2 + b(x_{k-1} - x_{k-2}) + c, \\ P(x_k) &= f(x_k) = a(x_k - x_{k-2})^2 + b(x_k - x_{k-2}) + c, \end{aligned}$$

²As constantes (coeficientes) deveriam está subscritos como a_k , b_k e c_k , pois dependem da iteração do método, mas eliminamos para simplificar a notação

observe que $f(x_{k-2}) = c$, assim definimos que:

$$a = \frac{(x_{k-1} - x_{k-2})[f(x_k) - f(x_{k-2})] - (x_k - x_{k-2})[f(x_{k-1}) - f(x_{k-2})]}{(x_k - x_{k-2})(x_{k-1} - x_{k-2})(x_k - x_{k-1})},$$

$$b = \frac{(x_{k-1} - x_{k-2})^2[f(x_{k-1}) - f(x_{k-2})] - (x_{k-1} - x_{k-2})^2[f(x_k) - f(x_{k-2})]}{(x_k - x_{k-2})(x_{k-1} - x_{k-2})(x_k - x_{k-1})},$$

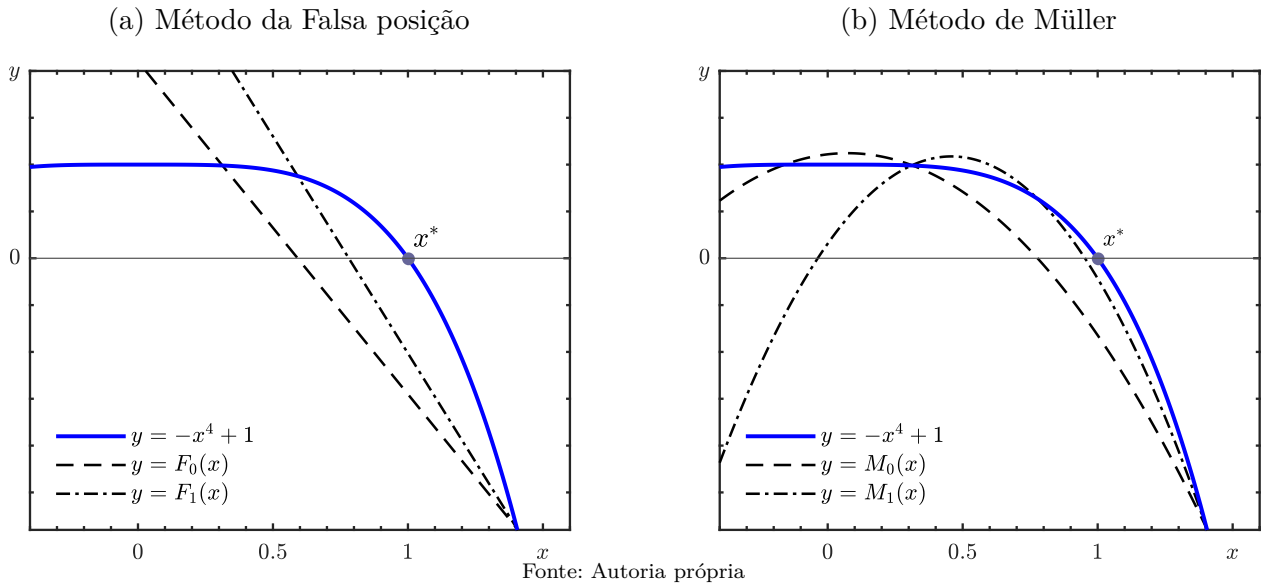
$$c = f(x_{k-2}).$$

Sendo assim, a nova aproximação para solução x_{k+1} será $P(x) = 0$ dada pela fórmula iterativa³ (3.3), no entanto, essa expressão está da forma que resolve problemas de estabilidade numérica⁴.

$$x_{k+1} = x_k - \frac{2c}{b \operatorname{sign}[b] \sqrt{b^2 - 4ac}}, \quad k = 0, 1, \dots \quad (3.3)$$

Podemos observar na figura 3.3 que geometricamente o método de Müller, representado pelas curvas iteradoras $M_0(x)$ e $M_1(x)$ já estão mais próximas da raiz em relação as retas $F_0(x)$ e $F_1(x)$ que descrevem o método da falsa posição, isto é, métodos baseados em interpolação quadrática geralmente tem uma melhor estimativa para a raiz comparados a métodos baseados em interpolação linear (é adotado o mesmo intervalo para os dois métodos).

Figura 3.3: Simulações geométricas das iterações da Falsa posição e Müller



Müller tem convergência superior ao método da secante, mas para raízes múltiplas, sua velocidade de convergir para raiz diminui. No entanto, umas das principais vantagens é a não

³Na expressão, $\operatorname{sign}[b]$ é a função que retorna o mesmo sinal do coeficiente b , ou seja $+$ ou $-$, e sendo portanto a operação entre b e $\sqrt{b^2 - 4ac}$

⁴Essa formulação é utilizada para minimizar o cancelamento na subtração

utilização de derivadas em sua fórmula iterativa, bem como a possibilidade de determinar zeros complexos.

A [function A.5](#) é o programa em `matlab` do método de Müller criado para este trabalho.

3.2.2 Método de Brent

Este método, codificado na [function A.6](#), pertence à classe de híbridos, pois combina três métodos: Interpolação linear, Interpolação Quadrática Inversa (IQI) e Bisseção. Foi apresentado por van Wijngaarden em 1963 e modificado por Theodorus Dekker em 1969 e, finalmente, refinado por Richard Peirce Brent em 1973 ([FILHO, 2018](#)). A robustez do método de Brent conta com a rapidez dos métodos baseados em interpolação linear, quadráticas e a garantia de convergência do método da bisseção.

Inicialmente, a combinação contava apenas com o método da secante e bisseção, que foi proposto por Dekker para garantir a convergência rápida utilizando métodos baseados em aproximação linear. No entanto, o método da secante apresenta falhas de convergência se a função não for bem comportada no intervalo e, conseqüentemente, divergindo do intervalo escolhido. Brent sugere a modificação do método, implementando a interpolação quadrática inversa (IQI). O processo iterativo do método de Brent é descrito em [Moraes \(2021, p. 8\)](#):

Basicamente o método de Brent utilizará a interpolação quadrática inversa de modo a interpolar usando uma parábola, caso ela falhar usará o método da secante, de modo a convergir mais rápido para a raiz. Porém, caso esses métodos produzam um resultado onde a estimativa da raiz não esteja no intervalo desejável, o algoritmo passará a utilizar o método da bisseção, que possui maior garantia de convergência, gerando assim uma estimativa para a raiz aproximada estar dentro do intervalo que contém a raiz. ([MORAIS, 2021, p. 8](#))

Na interpolação inversa o polinômio interpolador de grau dois

$$P_2(x) \approx f^{-1}(y) = x \tag{3.4}$$

é construído a partir das coordenadas $[f(x_{k-2}), x_{k-2}]$, $[f(x_{k-1}), x_{k-1}]$ e $[f(x_k), x_k]$. Para obter uma aproximação de $f^{-1}(t)$, é necessário avaliar $P_2(t)$. Então, esse polinômio $P_2(y)$ pode ser um polinômio quadrático de Lagrange ([FILHO, 2018](#)).

O algoritmo de interpolação quadrática inversa tem a seguinte relação de recorrência:

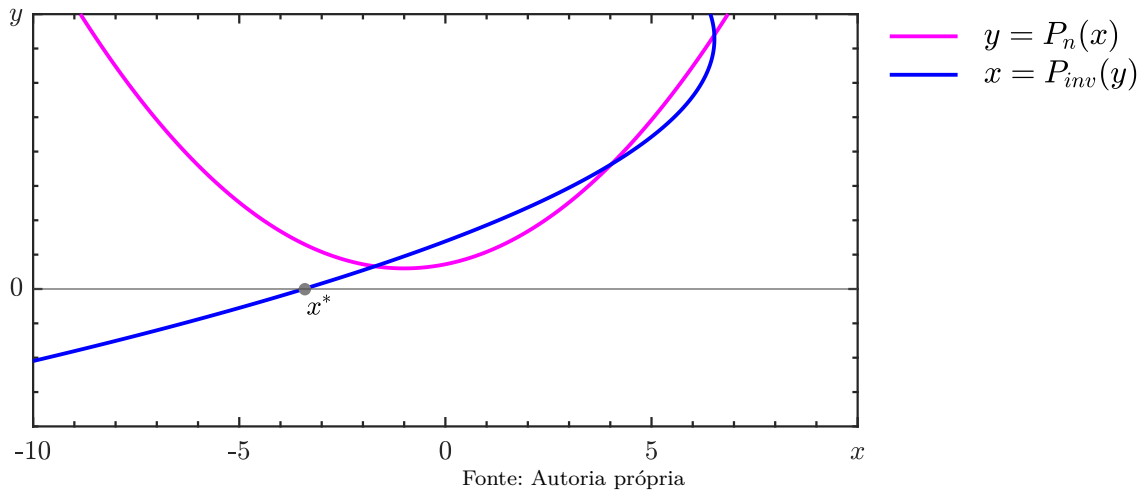
$$\begin{aligned} f^{-1}(y) = x_{k+1} = & x_{k-2} \frac{[y-f(x_{k-1})][y-f(x_k)]}{[f(x_{k-2})]-f(x_{k-1})][f(x_{k-2})-f(x_k)]} + x_{k-1} \frac{[y-f(x_{k-2})][y-f(x_k)]}{[f(x_{k-1})]-f(x_{k-2})][f(x_{k-1})-f(x_k)]} \\ & + x_k \frac{[y-f(x_{k-2})][y-f(x_{k-1})]}{[f(x_k)]-f(x_{k-2})][f(x_k)-f(x_{k-1})]} \end{aligned} \tag{3.5}$$

Filho (2018) explica que como $y = f(x) \Rightarrow x = f^{-1}(x)$, então haverá um ponto de abscissa em correspondência a $f^{-1}(0)$ tendo em vista uma aproximação da raiz de $f(x) = 0$, onde essa aproximação é dada por $P_2(0)$ em razão de (3.4).

Como acontece no método de Müller, podemos inferir aproximações mediante polinômios quadráticos, isto é, construir uma função quadrática que passe por três pontos distintos. A estimativa para raiz será melhor, pois, a utilização de uma curva normalmente gera aproximações mais satisfatórias que uma reta, entretanto, a abordagem por um polinômio quadrático pode falhar em alguns casos que a parábola não intercepta o eixo x , gerando zeros complexos (CHAPRA, 2013). Nesse sentido, a abordagem da interpolação quadrática inversa resolve o problema de falha na intersecção da parábola.

A figura 3.4 apresenta uma parábola normal $P_n(x)$ (em magenta) que não cruza o eixo horizontal, mas é facilmente resolvida pela implementação de uma parábola inversa $P_{inv}(y)$ (em azul) que agora intercepta o eixo x .

Figura 3.4: Parábolas ajustadas aos mesmos pontos



Por conseguinte, existe o cálculo de um ponto de aproximação (s) que é obtido primeiramente através da IQI e pelo método da secante, com $f(x_k) \neq f(x_{k-1}) \neq f(x_{k-2})$, caso esse processo seja insatisfatório, o algoritmo utilizará o método da bisseção (m).

Segundo Teixeira & Silva (2018) o método de Brent faz um teste inicial para o método da secante mediante duas comparações sob uma tolerância ϵ . Dessa forma, caso o método da bisseção seja utilizado e a condição $|\epsilon| < |x_k - x_{k-1}|$ seja satisfeita, o algoritmo utilizará interpolação (IQI ou secante), senão, o resultado do método da bisseção é avaliado na próxima iteração. Por outro lado, se na iteração ocorreu um passo de interpolação (IQI ou secante) e $|\epsilon| < |x_{k-1} - x_{k-2}|$ é satisfeito, utiliza-se novamente a interpolação, senão, o método da bisseção é usado novamente. Além disso, se na iteração anterior for utilizado o método da bisseção e $|s - x_k| < \frac{1}{2}|x_k - x_{k-1}|$ deve ser satisfeito, caso contrário, o método da bisseção

é realizado e seu resultado é aplicado na iteração posterior. Por fim, se a iteração anterior realizou a interpolação (IQI ou secante), então $|s - x_k| < \frac{1}{2}|x_{k-1} - x_{k-2}|$ deverá ser satisfeito.

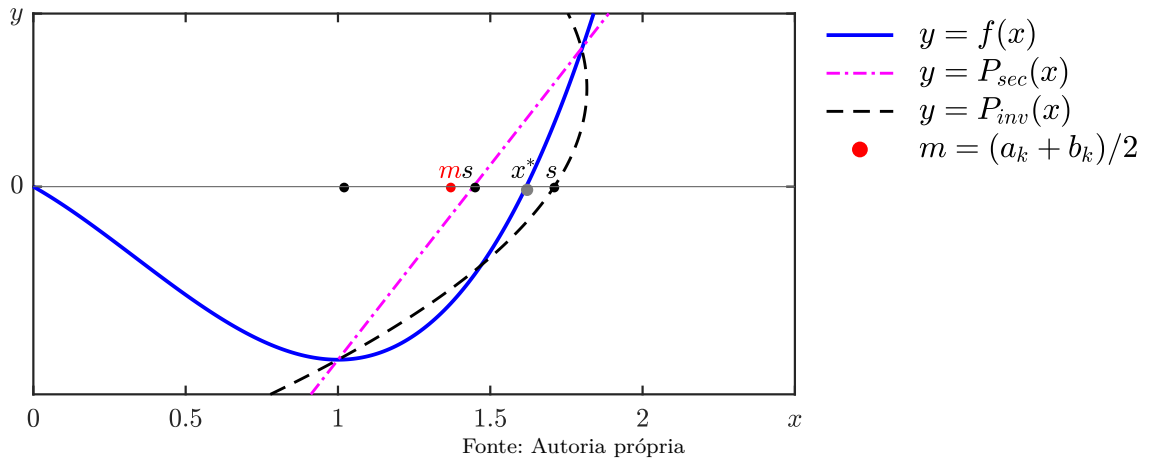
É descrito em [Teixeira & Silva \(2018, p. 5\)](#) as vantagens dessa modificação:

Esta modificação garante que a iteração k , o método da Bissecção será realizada em, no máximo $2 \log_2(|x_{k-1} - x_{k-2}|/\epsilon)$ iterações adicionais, a condição citada força o uso da interpolação reduzindo pela metade a cada duas iterações até que o tamanho do passo seja menor que ϵ . Brent provou que o seu método requer no máximo N^2 iterações, em que N indica o número de iterações para o método da bissecção. ([TEIXEIRA; SILVA, 2018, p. 5](#))

Nesse sentido, o processo de busca pela raiz se repete até que o critério de parada seja satisfeito.

A figura 3.5 representa o processo de isolamento da raiz, onde a reta $P_{sec}(x)$ (em magenta) representa uma iteração do método da secante, já a curva $P_{inv}(x)$ (em preto) é a interpolação quadrática inversa e ponto m (em vermelho), faz referência ao resultado do método da bissecção.

Figura 3.5: Simulação geométrica do método de Brent



De acordo com [Chapra \(2013 apud MOLER, 2004, p. 166\)](#) o método de Brent tem seu algoritmo presente na base da função `fzero` da biblioteca do `matlab`. Ademais, existem variações do algoritmo de Brent que utilizam no lugar do método da secante os métodos de Illinois e Pegasus, pois eles aceleram a convergência em intervalos de concavidade/convexidade conforme visto na (seção 3.1).

3.3 Métodos com Derivadas de Segunda Ordem

Os métodos dessa classe contam com derivadas de primeira e segunda ordem em sua fórmula iterativa. O método de Newton apresentado anteriormente utiliza a primeira derivada de f em sua iteração e, geometricamente, representa uma reta que tangencia o gráfico da função e que converge quadraticamente para uma raiz simples.

É descrito em [Eustaquio \(2013\)](#) que a utilização do polinômio de Taylor de segunda ordem pode gerar um modelo para f que interpola suas derivadas f' e f'' . Tal manipulação no polinômio cria uma classe de métodos de convergência cúbica, destacando-se nesse trabalho: método de Halley e de Chebyshev.

3.3.1 Método de Halley

O método de Halley⁵ foi proposto pelo astrônomo e matemático Edmond Halley em 1694, quase no mesmo período do surgimento do método de Newton. É conhecido por sua convergência cúbica e pelo fato de utilizar derivada de segunda ordem em sua fórmula iterativa.

$$x_{k+1} = x_k - \frac{2f(x_k)f'(x_k)}{2f'(x_k)^2 - f(x_k)f''(x_k)}, \quad k = 0, 1, \dots \quad (3.6)$$

De acordo com [Ferreira et al. \(2013\)](#) o método de Halley constrói uma sequência de aproximações para a solução de f , modelando a situação original. O método de Halley consiste em aproximar as raízes de f mediante a uma função h que geometricamente representa uma hipérbole tangente⁶ ao gráfico da função ([FERREIRA et al., 2013](#)).

[Ferreira & Mello \(2012, p. 276\)](#) explicam as características geométricas da seguinte forma:

Geometricamente, podemos determinar a fórmula iterativa do método de Halley tomando a cada passo a aproximação cujo gráfico é uma hipérbole tangente ao gráfico da função em torno do ponto atual. Esta hipérbole é o gráfico de uma função que interpola f e suas derivadas de ordem um e dois no ponto atual. Daí a denominação, também encontrada para o método de Halley, de método das hipérboles tangentes. ([FERREIRA; MELLO, 2012, p. 276](#))

Assim como o método de Newton utiliza um modelo de reta tangente ao gráfico de f para aproximar raízes, o método de Halley faz uso de um modelo baseado em hipérbole tangente

⁵Edmond Halley (1656 – 1742) foi um astrônomo e matemático britânico, célebre por ser o observador da órbita e decifrar o tempo do cometa Halley, em 1696

⁶O método de Halley também é conhecido como *método das hipérboles tangentes*

que é definida por:

$$h_k(x) = \frac{(x - x_k) + c}{a(x - x_k) + b}, \quad (3.7)$$

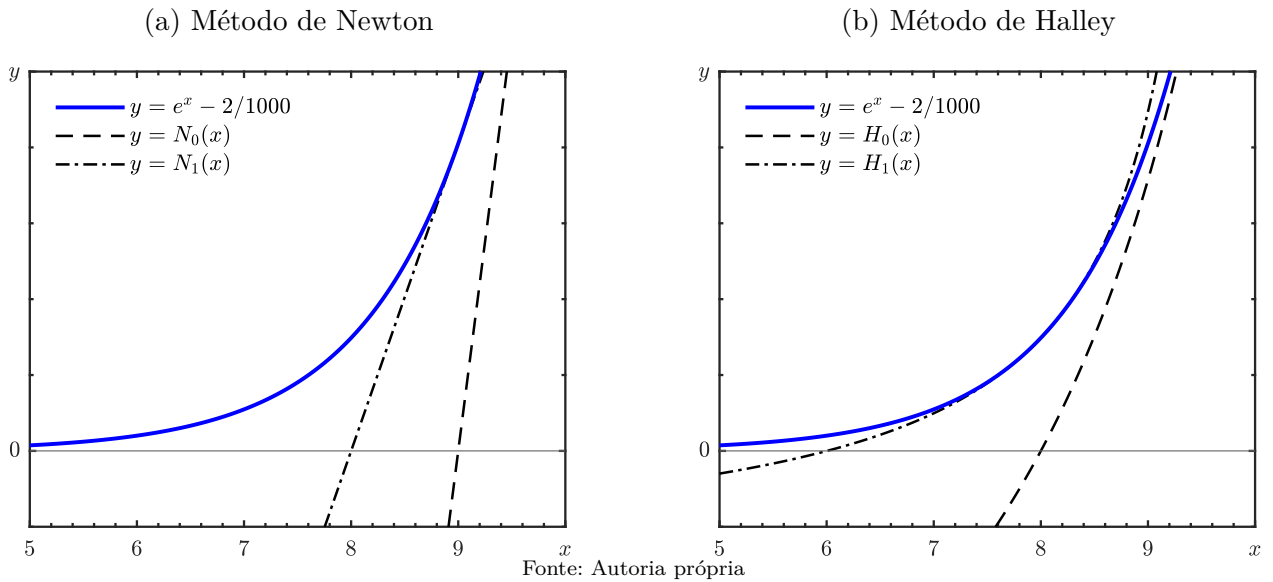
onde as constante a, b e c são dadas por:

$$\begin{aligned} a &= \frac{-2f''(x_k)}{2f'(x_k)^2 - f(x_k)f''(x_k)}, \\ b &= \frac{2f'(x_k)}{2f'(x_k)^2 - f(x_k)f''(x_k)}, \\ c &= \frac{2f(x_k)f'(x_k)}{2f'(x_k)^2 - f(x_k)f''(x_k)}. \end{aligned}$$

Para que o método de Halley esteja bem definido, os denominadores das constantes a, b e c devem ser não nulos. Além disso, [Ferreira et al. \(2013\)](#) explica que na fórmula iterativa (3.6), se $f''(x_k) = 0$ a fórmula iterativa torna-se igual ao método de Newton.

Na figura 3.6 apresentamos uma comparação geométrica de duas iterações no método de Newton e Halley na função $f(x) = e^x - 2/1000$, com $x_0 = 10$. Podemos perceber que as hipérboles $H_0(x)$ e $H_1(x)$ do método de Halley estão mais próximas da raiz em relação as retas tangentes $N_0(x)$ e $N_1(x)$ do método de Newton.

Figura 3.6: Simulações geométricas das iterações de Newton e Halley



Podemos obter o método de Halley através do polinômio de Taylor de ordem dois e aplicar um fator de correção de Newton. De acordo com [Eustaquio \(2013\)](#) um fator $(x - x_k)$ do termo quadrático do polinômio de Taylor será aproximado por $-\frac{f(x_k)}{f'(x_k)}$.

Seja p_k um polinômio de Taylor de segunda ordem, que é dado por:

$$p_k \approx f(x_k) + f'(x_k)(x - x_k) + \frac{f''(x_k)}{2}(x - x_k)^2. \quad (3.8)$$

Assim como o método de Newton pode ser obtido truncando esse polinômio na ordem de grau um, e consequentemente extrair sua raiz, podemos seguir da mesma forma e obter o zero de p_k (FERREIRA et al., 2013).

Dessa forma, manipulando algebricamente (3.8), obtemos:

$$x - x_k = -\frac{f(x_k)}{f'(x_k) + \frac{1}{2}f''(x_k)(x - x_k)}. \quad (3.9)$$

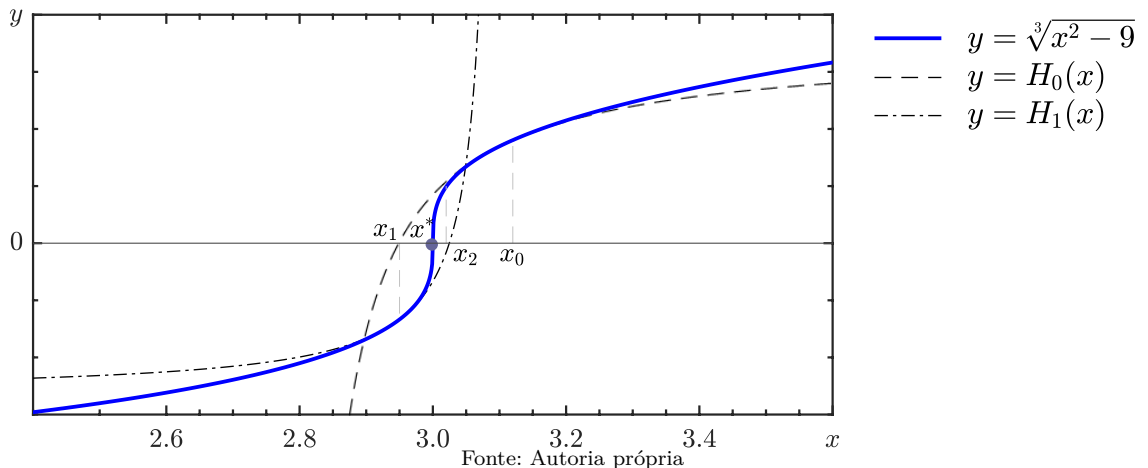
Aplicando a correção de Newton $(x - x_k) = -\frac{f(x_k)}{f'(x_k)}$ no lado direito da equação (3.9), temos:

$$x - x_k = -\frac{f(x_k)}{f'(x_k) - \frac{f''(x_k)f(x_k)}{2f'(x_k)}} \Rightarrow x = x_k - \frac{2f(x_k)f'(x_k)}{2f'(x_k)^2 - f''(x_k)f(x_k)},$$

dessa maneira, chegamos na fórmula iterativa do método de Halley. Ainda existe outra maneira de obter tal fórmula, pois podemos aplicar o passo de Newton ao produto $f \cdot g$, pois seus zeros contém soluções de f . Dessa forma, utilizamos $g = \frac{1}{\sqrt{f'}}$ e tem-se a mesma fórmula (FERREIRA; MELLO, 2012).

Na figura 3.7, observamos que o método de Halley representado pelas hipérboles tangentes $H_0(x)$ e $H_1(x)$ criam uma sequência $(x_0, x_1, x_2, \dots, x_k)$ que converge para raiz da função $f(x) = \sqrt[3]{x^2 - 9}$, tendo em vista que o método de Newton diverge próximos a pontos de inflexão conforme foi explicitado na figura 2.1.

Figura 3.7: Simulações geométricas de Halley em uma raiz próxima a pontos de inflexão



O método de Halley está codificado na [function A.8](#).

3.3.2 Método de Chebyshev

O método de Chebyshev é outro método de convergência cúbica que utiliza a segunda derivada em sua fórmula iterativa. Na literatura Russa ele é creditado a Chebyshev, mas [Traub \(1982\)](#) associa esse método a Euler.

Pertencente a classe de métodos Chebyshev-Halley, o método consiste em obter f^{-1} para calcular a raiz de $f^{-1}(0)$. Nesse sentido, f possui uma inversa $h = f^{-1}$ tendo o polinômio de Taylor de segunda ordem de h no ponto $y_k = f(x_k)$ dado por:

$$p_k(y) = g(y_k) + g'(y_k)(y - y_k) + \frac{g''(y_k)}{2}(y - y_k)^2, \quad (3.10)$$

e defini-se x_{k+1} como sendo $p_k(0)$, assim a equação (3.10) fica:

$$x_{k+1} = p_k(0) = g(y_k) - g'(y_k)y_k + \frac{g''(y_k)}{2}(y_k)^2. \quad (3.11)$$

No entanto, em casos específicos é complicado obter a inversa de uma função, pois nem sempre ela está disponível ([EUSTAQUIO, 2013](#)). A solução para definir o método de Chebyshev, trata-se de obter uma expressão para x_{k+1} em termos de f . Definimos esses termos como:

$$g(y) = x, \quad g'(y) = \frac{1}{f'(x)}, \quad \text{e} \quad g''(y) = -\frac{f''(x)}{f'(x)^3},$$

dessa maneira, temos o método de Chebyshev, substituindo os termos na equação (3.11):

$$x_{k+1} = x_k - \left[1 + \frac{f(x_k)f''(x_k)}{2f'(x_k)^2} \right] \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \dots \quad (3.12)$$

O modelo usado pelo método representa geometricamente uma parábola osculante⁷ que tangencia o gráfico de f . Conforme é descrito em [Esmaeili & Rezaei \(2012\)](#) a derivação em termos de uma parábola satisfaz as condições de tangência $m(x_k) = f'(x_k)$, $m'(x_k) = f'(x_k)$ e $m''(x_k) = f''(x_k)$. Portanto, a parábola é definida por:

$$x = am_k(x)^2 + bm_k(x) + c, \quad (3.13)$$

⁷Na geometria diferencial, uma curva osculante é uma curva plana de uma determinada família que possui a maior ordem possível de contato com outra curva

onde a, b e c são:

$$a = -\frac{f''(x_k)}{2f'(x_k)^3}, \quad b = \frac{f'(x_k)^2 + f(x_k)f''(x_k)}{f'(x_k)^3}, \quad c = x_k - \frac{f(x_k)}{f'(x_k)} \left[1 + \frac{f(x_k)f''(x_k)}{2f'(x_k)^2} \right].$$

E observe que a constante c é solução da equação (3.13).

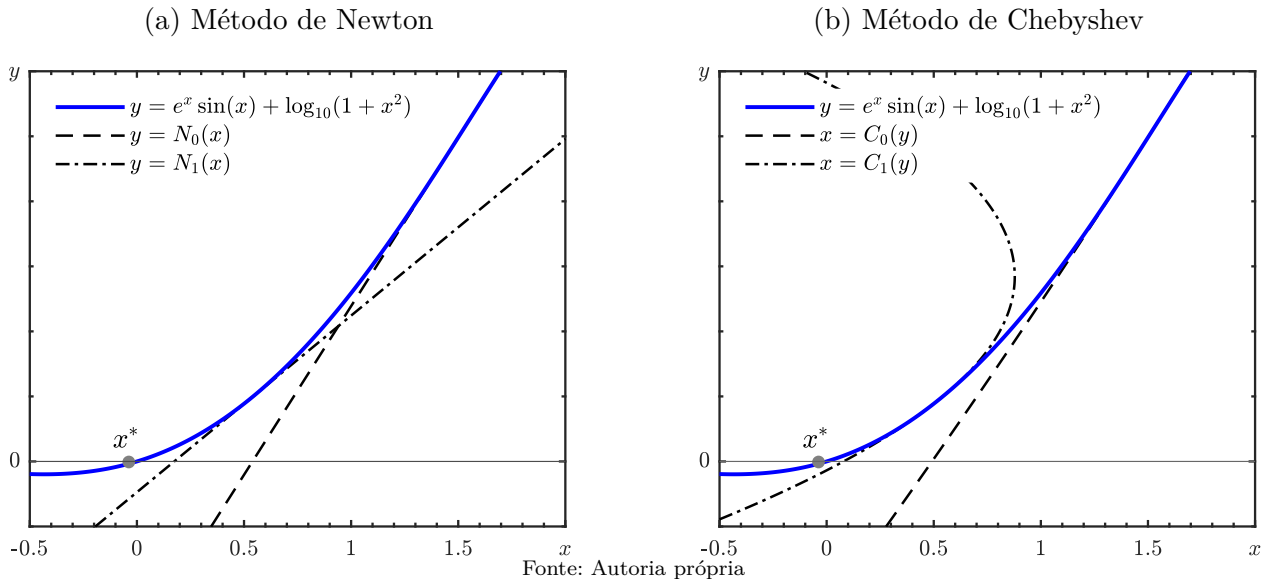
Além disso, historicamente podemos encontrar na literatura uma forma equivalente de escrever o método de Chebyshev, o qual é conhecido como método de Schröder⁸. A fórmula de Schröder é uma modificação do método de Newton para resolver problemas com raízes múltiplas quando sua multiplicidade é conhecida. O método de Schröder tem a seguinte fórmula de recorrência:

$$x_{k+1} = x_k - m \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, \dots \quad (3.14)$$

onde m é a multiplicidade da raiz.

Na figura 3.8 abaixo, comparamos as primeiras iterações de Newton e Chebyshev, explicitando a geometria de suas iterações na função $f(x) = e^x \sin(x) + \log_{10}(1 + x^2)$, partindo de um $x_0 = 1.5$. Destacando que $N_0(x)$ e $N_1(x)$ representam o modelo de reta tangente do método de Newton e $C_0(x)$ e $C_1(x)$ as parábolas inversas tangentes do método de Chebyshev.

Figura 3.8: Simulações geométricas das iterações de Newton e Chebyshev



O método de Chebyshev está no código em [function A.9](#).

⁸Friedrich Wilhelm Karl Ernst Schröder (1841-1902) foi um Matemático Alemão.

3.3.3 Método de Newton para raízes múltiplas

O método de Newton é muito robusto no cálculo de raízes simples, no entanto, quando a raiz procurada passa a ter presença de multiplicidade, o método de Newton tem seu desempenho prejudicado. De acordo com [Chapra & Canale \(2011\)](#), a presença de raízes múltiplas compromete o desempenho do método de Newton, pois a primeira derivada é nula na raiz e, consequentemente, gerando divisão por zero.

Além disso, é descrito em [Souza et al. \(2018b apud RALSTON; RABINOWITZ, 2001, p. 6\)](#) que:

[Ralston & Rabinowitz \(2001\)](#) observaram que nos casos em que a raiz possui multiplicidade o método de Newton-Raphson tem sua convergência comprometida, tornando-se apenas linear, ao invés de ser pelo menos quadrática, o que significa que o erro calculado será proporcional ao anterior, ao invés de ser proporcional ao seu quadrado. Para contornar isso, os mesmos resolveram definir uma nova função $u(x)$, que possui as mesmas raízes de $f(x)$. ([SOUZA et al., 2018b apud RALSTON; RABINOWITZ, 2001, p. 6](#))

Nesse sentido, definimos uma função auxiliar $u(x)$ com o objetivo de retirar $f'(x)$ do denominador no método de Newton:

$$u(x) = \frac{f(x)}{f'(x)}, \quad (3.15)$$

aplicando na equação de Newton (2.4), tem-se:

$$x_{k+1} = x_k - \frac{u(x_k)}{u'(x_k)}, \quad (3.16)$$

derivando a equação (3.15) através da regra do quociente, obtemos:

$$u'(x) = \frac{f'(x)f'(x) - f(x)f''(x)}{f'(x)^2} \quad (3.17)$$

Substituindo as equações (3.15) e (3.17) na equação (3.16), obtemos o método de Newton para raízes múltiplas.

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{f(x_k)^2 - f(x_k)f''(x_k)} \quad (3.18)$$

Essa modificação no método de Newton resolve o problema de indeterminação quando a primeira derivada se anula. Além disso, no método tradicional a convergência para zeros múltiplos é apenas linear, enquanto na modificação, a convergência quadrática é recuperada.

Uma fato interessante é mencionado em [Souza et al. \(2018a\)](#), pois para funções polinomiais que possuam uma única raiz, o método converge em apenas uma iteração. Desse modo, dada uma função $f(x) = (x - \bar{x})^m$, em que $\bar{x}$ e m são raiz e multiplicidade de f , respectivamente, aplicando na equação (3.18), temos:

$$\begin{aligned}
 x_{k+1} &= x_k - \frac{(x_k - \bar{x})^m [(x_k - \bar{x})^m]'}{[(x_k - \bar{x})^m]'^2 - (x_k - \bar{x})^m [(x_k - \bar{x})^m]''} \\
 x_{k+1} &= x_k - \frac{m(x_k - \bar{x})^m (x_k - \bar{x})^{m-1}}{[m(x_k - \bar{x})^{m-1}]^2 - (m^2 - m)(x_k - \bar{x})^m (x_k - \bar{x})^{m-1}} \\
 x_{k+1} &= x_k - \frac{m(x_k - \bar{x})^{2m-1}}{m^2(x_k - \bar{x})^{2m-2} - (m^2 - m)(x_k - \bar{x})^{2m-2}} \\
 x_{k+1} &= x_k - \frac{m(x_k - \bar{x})^{2m-1}}{m(x_k - \bar{x})^{2m-2}} \\
 x_{k+1} &= x_k - (x_k - \bar{x}) \\
 x_{k+1} &= \bar{x}
 \end{aligned}$$

A programação método de Newton para raízes múltiplas está na [function A.10](#).

Testes e Resultados

Este capítulo é destinado aos testes numéricos dos métodos mencionados. Apresentaremos comparações com os métodos tradicionais em funções com características específicas que afetam o custo computacional para encontrar a solução. Serão explicitados gráficos com as aproximações em função das iterações de ambos os métodos. A configuração da máquina para realizar o cálculos tem as seguintes características: Processador Intel Core i5 de 8ª Geração, 15 *Gigabytes* de memória RAM, SSD de 160 *Gigabytes* de armazenamento e Windows 10.

Os testes numéricos serão feitos em funções com aspectos mais complexos, ou seja, em funções com anomalias de *convexidade/concavidade* e *multiplicidade*. Algumas dessas funções serão retiradas de [Ganchovski & Traykov \(2023\)](#). Inicialmente, os testes serão feitos separadamente nos métodos baseados em interpolação linear, quadrática e com derivadas de segunda ordem, ou seja, a comparação será referente aos métodos clássicos apresentados anteriormente. A tabela 4.1 abaixo relaciona as características de convergência dos métodos clássicos e métodos não usuais.

Tabela 4.1: Características dos métodos

MÉTODO	TIPO	CONVERGÊNCIA	ORDEM
Bisseção (BSS)	Clássico	Linear	1
Falsa Posição (FP)	Clássico	Superlinear	1.618
Illinois (ILL)	Modificado	Superlinear	1.442
Pegasus (PGS)	Modificado	Superlinear	1.642
Brent (BRT)	Modificado (híbrido)	Superlinear	1.839
Müller (MLL)	Modificado	Superlinear	1.84
Newton (NT)	Clássico	Quadrática	2
NT raízes múltiplas (NT-RM)	Modificado	Quadrática	2
Halley (HLL)	Modificado	Cúbica	3
Chebyshev (CHV)	Modificado	Cúbica	3

Fonte: Elaborada pelo autor mediante [Ganchovski & Traykov \(2023\)](#)

4.1 Desempenho Computacional

Na análise do desempenho entre dois algoritmos é relevante avaliar os recursos computacionais disponíveis, existindo vários parâmetros de medidas nesse tipo de análise. As mais utilizadas estão relacionadas a eficiência temporal e eficiência espacial.

É válido ressaltar as configurações do computador o qual foram realizados os testes numéricos, pois o mesmo possui especificações muito boas, que de certo modo torna a avaliação de tempo computacional quase insignificante para as configurações de *hardware* da máquina, tal como a utilização do *Wise Memory Optimizer* que tem o objetivo de liberar/desfragmentar a memória, como também esvaziar a memória em espera. O aplicativo mencionado está disponível em (<https://www.wisecleaner.com/wise-memory-optimizer.html>).

4.1.1 Eficiência Temporal

De acordo com Souza (2015) a eficiência temporal leva-se em consideração as seguintes medidas: *tempo de resposta*, *taxa de transferência* e *complexidade computacional*. O *tempo de resposta* é uma medida que relaciona o tempo em que um determinado algoritmo reage a entrada de dados. A *taxa de transferência* consiste na avaliação de processamento dos dados de entrada e saída dentro de um período de tempo. Enquanto a *complexidade computacional* relaciona a quantidade de operações, seja lógicas ou aritméticas para resolver um problema.

4.1.2 Eficiência Espacial

A medida de eficiência espacial relaciona a quantidade de memória disponível que um algoritmo necessita para funcionar corretamente (SOUZA, 2015). Neste trabalho, as modificações de métodos não garantem que um problema será resolvido mais rápido que os métodos clássicos em alguns casos, pois algumas modificações incluem mais operações de lógica e aritmética.

Dessa forma, tanto a medida de *eficiência temporal* quanto a *eficiência espacial* são pertinentes, mesmo com a evolução dos hardwares, o tempo e quantidade de memória são imprescindíveis.

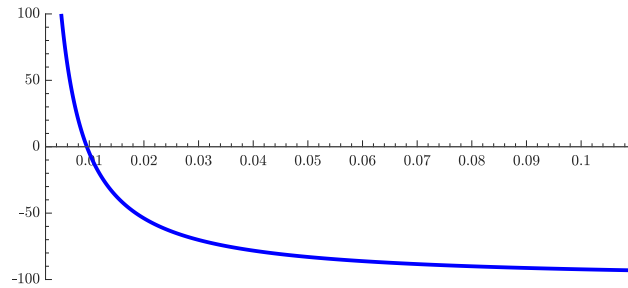
4.2 Comparação entre Métodos Clássicos e Métodos em Interpolação Linear Modificados

O método da Falsa Posição tem, como registrado na tabela 4.1, ordem de convergência 1.618, mas quando o método está em um intervalo com anomalias de concavidade/convexidade,

o método passa a convergir com taxas lineares (RALSTON; RABINOWITZ, 2001). Diante disso, a primeira comparação entre os métodos será na função abaixo, pois ela é um caso extremo sobre concavidade. Dessa forma, podemos observar o comportamento dos métodos clássicos com os métodos modificados da classe de interpolação linear.

$$f(x) = \frac{1}{x} + \ln(x) - 100. \quad (4.1)$$

Figura 4.1: Gráfico da função $f(x) = \frac{1}{x} + \ln(x) - 100$



Fonte: Autoria própria

Foi escolhido o intervalo $[0.002, 0.1]$ e um $x_0 = 0.002$ com um critério de parada de erro relativo $|x_{k+1} - x_k|/|x_{k+1}| < \epsilon$ e número limite de 100 iterações, com precisão de $\epsilon = 10^{-8}$.

As **functions** de todos os testes serão executados um total de 50 vezes para calcular a média aritmética do tempo de processamento, a qual está codificada em A.11.

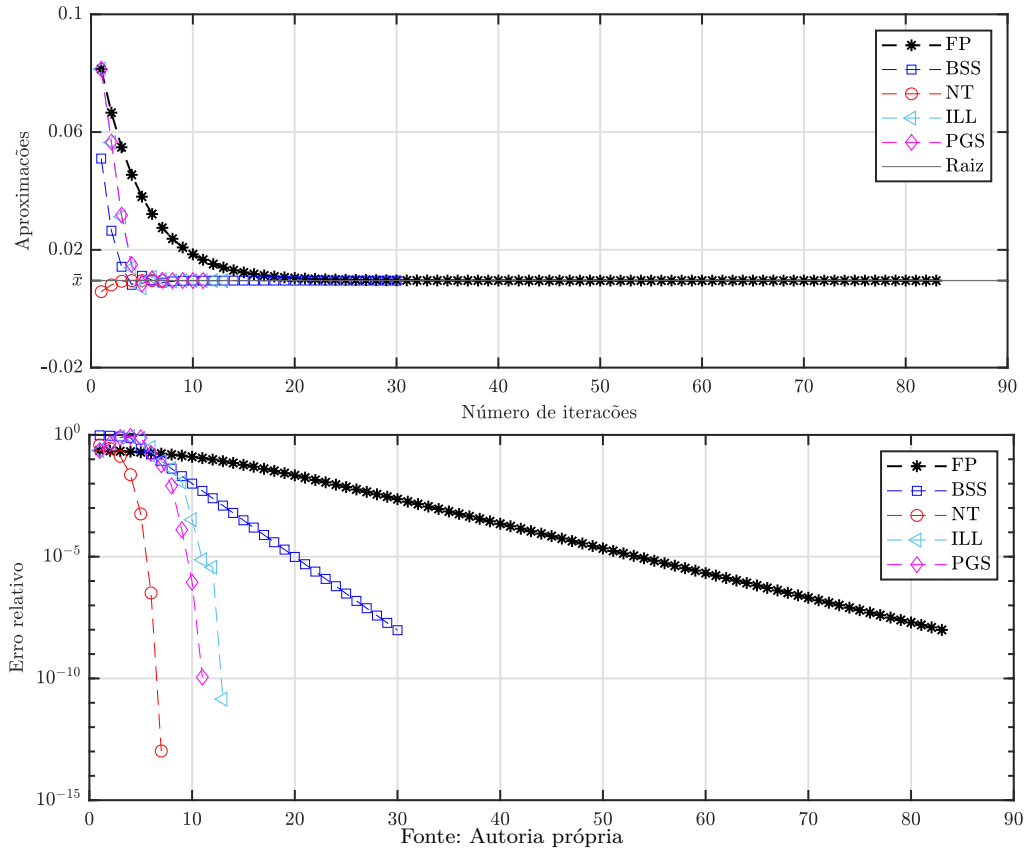
Os resultados obtidos estão na tabela 4.2 e na figura 4.2. Na tabela, além do erro relativo, são apresentados o tempo de processamento e o número de iterações. Já as imagens possuem as aproximações e as curvas dos erros relativos em função das iterações.

Tabela 4.2: Desempenho dos métodos em $f(x) = \frac{1}{x} + \ln(x) - 100$

MÉTODOS	ITERAÇÕES	ERRO RELATIVO	TEMPO (s)
Falsa posição	83	9.851×10^{-9}	9.684×10^{-5}
Bisseção	30	9.551×10^{-9}	4.116×10^{-5}
Newton	7	1.051×10^{-13}	2.504×10^{-5}
Illinois	13	1.405×10^{-11}	2.109×10^{-5}
Pegasus	11	1.122×10^{-10}	2.240×10^{-5}

Fonte: Autoria própria

De acordo com o resultados obtidos para a função (4.1), que apresenta anomalias de convexidade no intervalo da raiz, os métodos da falsa posição e bisseção tiveram um número de iterações consideravelmente alto em relação ao método de Newton e aos métodos modificados de Illinois e Pegasus. Isso mostra que o método da falsa posição bastante custoso nesse tipo

Figura 4.2: Aproximações e curvas do erro relativo dos métodos

de problema. O método de Newton é extremamente robusto na busca por raízes simples, foi o método com menos iterações e melhor erro relativo nesse problema. Comparando o método da falsa posição com suas variantes, o método de Illinois teve uma eficiência de 538,5% no número de iterações em relação ao método tradicional, enquanto o método de Pegasus teve eficiência de 654,5% em suas iterações. O método da bisseção apresentou iterações razoáveis para o problema comparado ao método da falsa posição. Isso demonstra a dificuldade do método em funções com característica de concavidade/convexidade, mas o método da falsa posição converge para solução mesmo precisando de mais iterações.

Observando-se os tempos de processamento o leitor sem experiência em aplicações poderia se questionar se a maioria não têm os objetivos alcançados satisfatoriamente! Isso é verdade quando se busca uma só raiz. No entanto, podemos dar um exemplo que foge disso! Em modelagens numéricas em engenharia é bastante comum aparecerem integrais impróprias (com um dos limites de integração infinito), e em muitos casos, as quadraturas Gauss-Lobato ou Gauss-Laguerre por exemplo não são satisfatórias. Para contornar essa ineficiência, tomam-se quadraturas limitadas entre as raízes dos integrandos, que num método iterativo realiza extrapolação para aceleração da convergência (uma boa referência disso é [Key \(2012\)](#)). Tam-

bém existem boas referências nesse artigo!). O número de subintervalos para as quadraturas não pode antecipadamente ser conhecido, e por isso um grande número de raízes deve ser encontrado antes das quadraturas, demandando um tempo agora computacional bem significativo, e cuja escolha do método que possui o menor para uma única raiz é crucial para uma otimização.

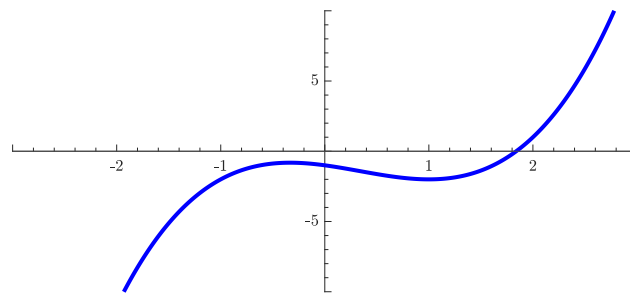
4.3 Comparação entre Métodos Clássicos e Métodos em Interpolação Quadrática

Agora vamos comparar os métodos clássicos em relação aos métodos baseados em interpolação quadrática, que são algoritmos que utilizam a estratégia de aproximar raízes por parábolas. O método de Müller é conhecido por sua convergência rápida e por determinar zeros complexos, enquanto o método de Brent é um algoritmo híbrido, que combina três abordagens distintas na busca pela raiz, e no qual podemos encontrar a base de seu algoritmo na rotina `fzero` do `matlab`.

O comparativo será analisado na função (4.2), pois as características intervalares onde a raiz se encontram são menos complexas que a função (4.1). Dessa forma, analisaremos o desempenho dos métodos baseados em (IQ) em relação aos métodos clássicos.

$$f(x) = x^3 - x^2 - x - 1 \quad (4.2)$$

Figura 4.3: Gráfico da função $f(x) = x^3 - x^2 - x - 1$



Fonte: Autoria própria

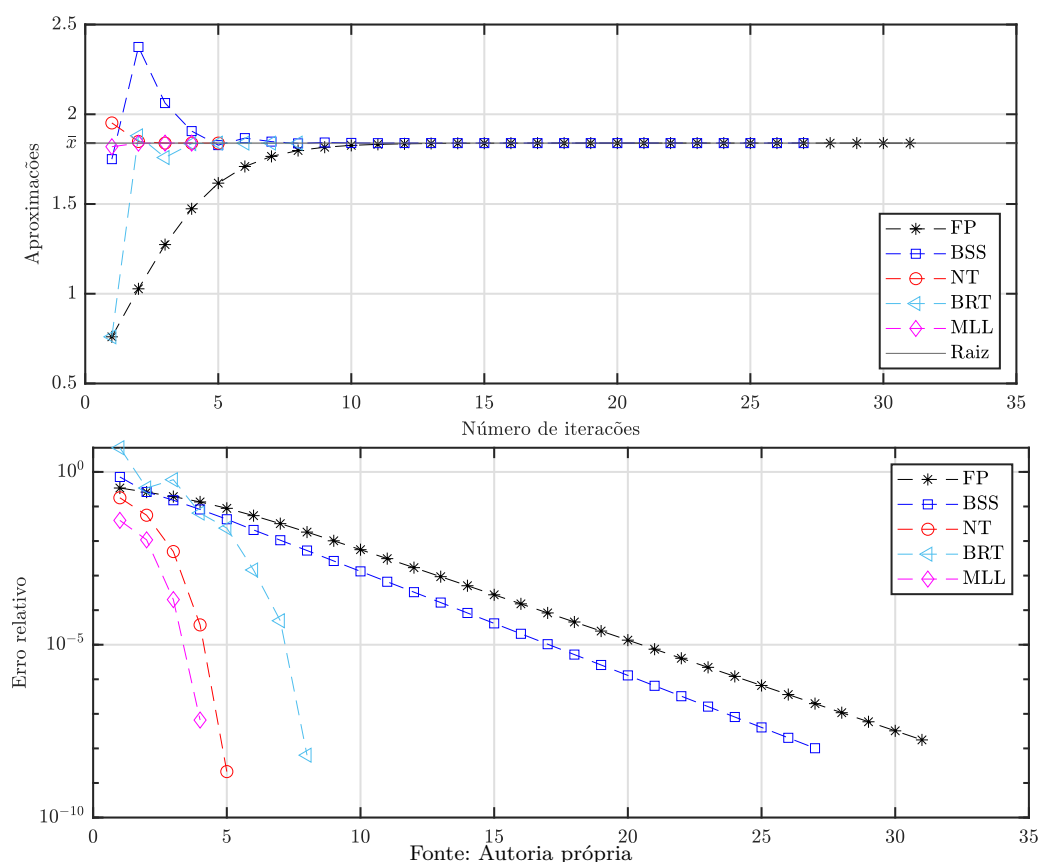
O intervalo escolhido foi $[0.5, 3]$, $x_0 = 3$, já o critério de parada é novamente $|x_{k+1} - x_k|/|x_{k+1}| < \epsilon$ sob a tolerância $\epsilon = 10^{-8}$ e o número limite de 100 iterações. Os resultados estão apresentados na tabela 4.3 e na figura 4.4.

Com base nos resultados obtidos é possível concluir que para funções onde a raiz encontra-se em um intervalo com menos complexidade, os métodos baseados em interpolação quadrática possuem uma eficiência superior aos métodos clássicos, mas isso depende também da

Tabela 4.3: Desempenho dos métodos em $f(x) = x^3 - x^2 - x - 1$

MÉTODOS	ITERAÇÕES	ERRO RELATIVO	TEMPO (s)
Falsa posição	31	1.762×10^{-8}	5.301×10^{-5}
Bissecção	27	2.025×10^{-8}	4.930×10^{-5}
Newton	5	2.134×10^{-9}	2.009×10^{-5}
Müller	4	6.610×10^{-8}	2.053×10^{-5}
Brent	8	3.649×10^{-9}	3.424×10^{-5}

Fonte: Autoria própria

Figura 4.4: Aproximações e curvas do erro relativo dos métodos

Fonte: Autoria própria

amplitude do intervalo escolhido, pois uma amplitude menos pode gerar outros resultados. Com exceção ao método de Newton que mesmo com chute inicial relativamente longe da raiz, convergiu em poucas iterações

O método da falsa posição convergiu para a raiz aproximada, no entanto, apresentou convergência lenta em relação aos outros métodos. O método da bissecção é confiável embora tenha um número relativamente alto de iterações. O método de Newton é o único dos algoritmos clássicos que mantém sua eficiência em comparação aos métodos não convencionais e

tem sua convergência para raiz em poucas iterações. Os métodos modificados são tão eficientes quanto o método de Newton e muito superiores aos métodos da falsa posição e bisseção por terem abordagens destinadas a contornar ineficiências de alguns métodos clássicos.

4.4 Comparação Entre Métodos Clássicos e Métodos com Derivadas de Segunda Ordem

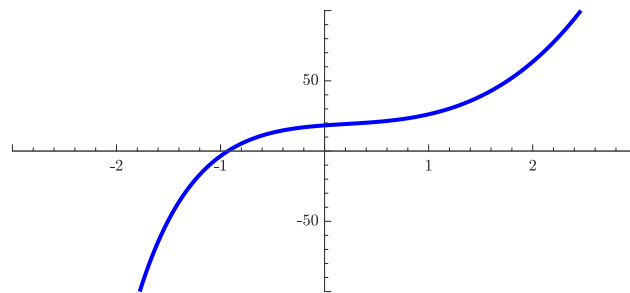
Os métodos baseados em derivadas de segunda ordem são variações do método de Newton que possuem convergência superior. Os métodos de Halley e Chebyshev tem suas explicações geométricas bem definidas, mas o método de Newton-RM não tem uma interpretação geométrica fundamental, pois esse algoritmo serve exclusivamente para manter a convergência quadrática em raízes múltiplas.

Os testes numéricos desta classe foram executados em duas funções com características distintas, pois queremos analisar o desempenho dos métodos em funções que tenham raízes simples e múltiplas. A primeira função possui raiz simples, e a segunda possui uma raiz com multiplicidade 2.

A primeira função, (4.3), tem sua raiz analisada no intervalado $[-2, 1]$ e aproximação inicial $x_0 = -2$. O teste foi com tolerância $\epsilon = 10^{-8}$ sobre o critério de parada do erro relativo $|x_{k+1} - x_k|/|x_{k+1}| < \epsilon$, e sob um limite de 100 iterações.

$$f(x) = 5x^3 + x^2 - e^{1-2x} + \cos(x) + 20. \quad (4.3)$$

Figura 4.5: Gráfico da função $f(x) = 5x^3 + x^2 - e^{1-2x} + \cos(x) + 20$



Fonte: Autoria própria

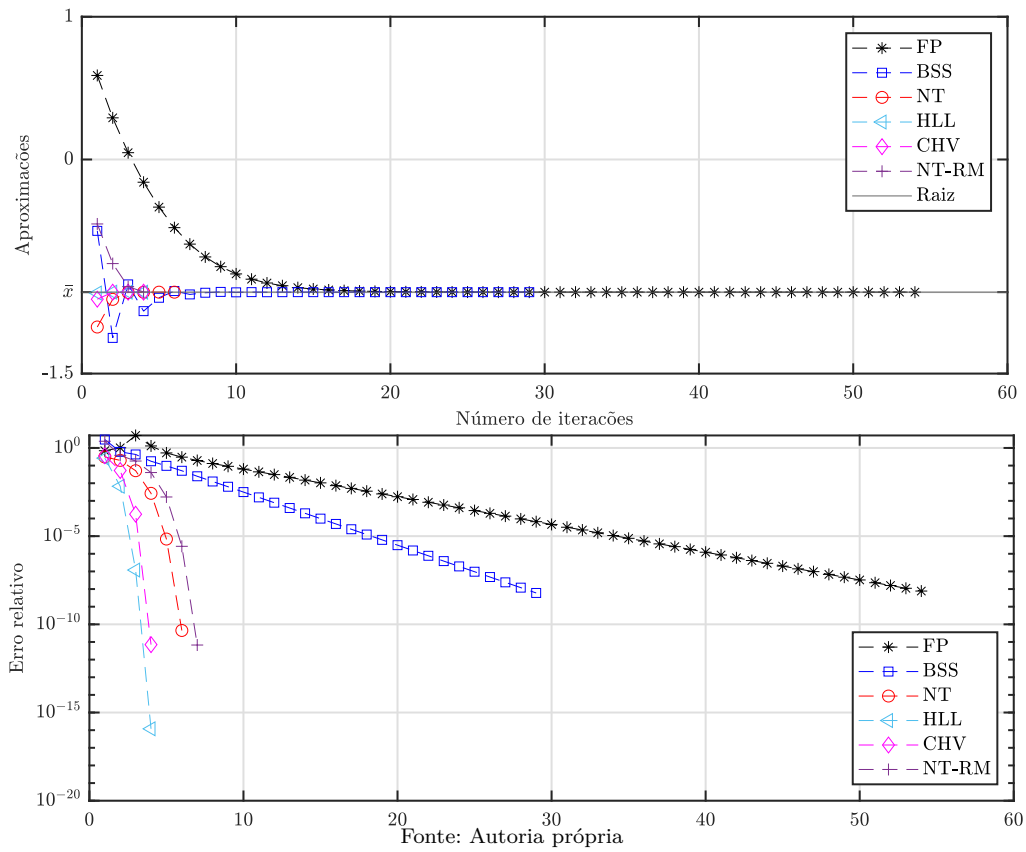
Os resultados numéricos para esse problema estão apresentados na tabela 4.4 e na figura 4.6.

Para a função (4.3) todos métodos convergiram para a raiz analisada. Os métodos de Halley e Chebyshev tiveram melhor eficiência tanto no número de iterações quanto no erro relativo, se destacando Halley. O método de Newton-RM apresentou números de iterações

Tabela 4.4: Desempenho dos métodos em $f(x) = 5x^3 + x^2 - e^{1-2x} + \cos(x) + 20$

MÉTODOS	ITERAÇÕES	ERRO RELATIVO	TEMPO (s)
Falsa posição	54	7.709×10^{-9}	8.705×10^{-5}
Bisseção	29	6.011×10^{-9}	6.557×10^{-5}
Newton	6	4.483×10^{-11}	2.143×10^{-5}
Halley	4	1.194×10^{-16}	3.247×10^{-5}
Chebyshev	4	7.069×10^{-12}	3.900×10^{-5}
Newton raízes múltiplas	7	6.712×10^{-12}	3.840×10^{-5}

Fonte: Autoria própria

Figura 4.6: Aproximações e curvas do erro relativo dos métodos.

Fonte: Autoria própria

semelhantes à Newton, e teve o erro relativo superior, mesmo que o propósito do método seja resolver problemas com raízes múltiplas, manteve convergência semelhante a Newton para raízes simples. Os métodos da falsa posição e bisseção foram menos eficientes tanto no número de iterações quando no erro relativo.

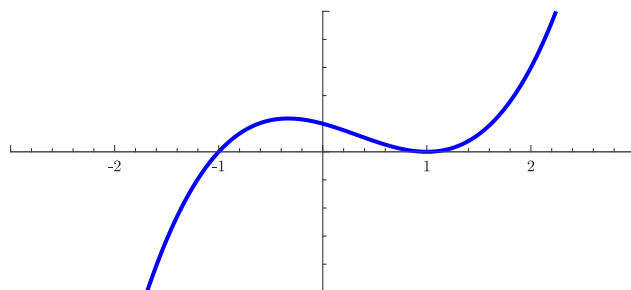
A segunda função será analisada no intervalo $[1, 4]$ com os mesmos parâmetros do teste anterior. No entanto, a função (4.4) tem uma raiz com multiplicidade par, isto significa

graficamente que a função não cruza o eixo x no intervalo em que a raiz se encontra. Dessa forma, métodos intervalares que utilizam o **Teorema de Bolzano** (1) em seu algoritmo não funcionam, pois por hipótese os sinais de $f(a)$ e $f(b)$ devem ser opostos.

Será definido um chute inicial $x_0 = 4$. A figura 4.7 abaixo mostra o gráfico da função estudada.

$$f(x) = x^3 - x^2 - x + 1. \quad (4.4)$$

Figura 4.7: Gráfico da função $f(x) = x^3 - x^2 - x + 1$



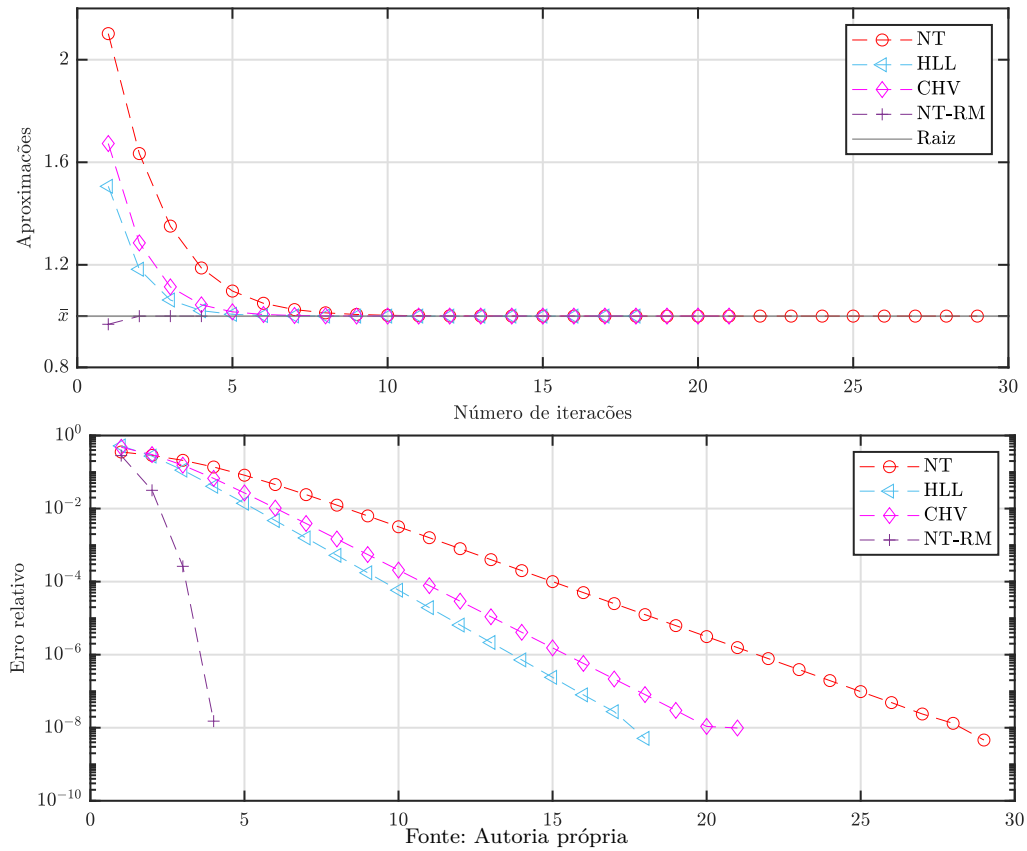
Fonte: Autoria própria

Os resultados obtidos para este teste estão explanados na tabela 4.5 e na figura 4.8, ressaltando que nesses teste os resultados para os métodos intervalares não estão presentes pelos motivos característicos de uma função com raiz com multiplicidade par e, consequentemente, sua implicação nos algoritmos da bisseção e falsa Posição.

Tabela 4.5: Desempenho dos métodos em $f(x) = x^3 - x^2 - x + 1$

MÉTODOS	ITERAÇÕES	ERRO RELATIVO	TEMPO (s)
Falsa posição	-	-	-
Bisseção	-	-	-
Newton	29	4.6070×10^{-9}	7.628×10^{-5}
Halley	18	5.1128×10^{-9}	7.126×10^{-5}
Chebyshev	21	9.8347×10^{-9}	7.410×10^{-5}
Newton raízes múltiplas	4	1.5152×10^{-8}	2.438×10^{-5}

Fonte: Autoria própria

Figura 4.8: Aproximações e curvas do erro relativo dos métodos

Na função (4.4) ocorre o problema de determinar uma raiz múltipla, e somente os métodos baseados em derivadas convergem para a solução. No entanto, o método de Newton na presença multiplicidade passa a convergir com taxas lineares. Os métodos de Halley e Chebyshev são mais eficientes que Newton no número de iterações e com erros relativos próximos. O método de Newton-RM mostra-se muito robusto e eficiente nesse tipo de situação, em relação ao método clássico. Ele é 625% mais eficiente em quantidade de iterações. Comparado a Halley e Chebyshev, o método de Newton-RM foi 375% e 425% mais eficiente em número de iterações, respectivamente. Quanto ao seu erro relativo, Newton-RM apresentou pouca discrepância em relação aos demais.

4.5 Comparação Geral entre os Métodos

Nesta seção apresentaremos o comparativo em três funções com características diferentes. No entanto, utilizar um conjunto de funções para comparar o desempenho é longe do ideal, isto é, a dependência na escolha da função e suas características, bem como a escolha dos valores iniciais e também a amplitude do intervalo ao qual a raiz se encontra pode caracterizar esco-

lhas que demandem melhor resultados num método que em outro, pois os diversos métodos numéricos em cada exemplo tem uma característica distinta; e consequentemente um único método não é capaz resolver inúmeros problemas sozinho. Entretanto, é interessante analisar o desempenho geral dos métodos estudados nos capítulos anteriores, pois a deficiência de um método pode ser facilmente resolvida por outro algoritmo.

Para todos os métodos foram usados os mesmos parâmetros dos teste anteriores: máximo de 100 iterações, tolerância $\epsilon = 10^{-8}$ sob critério de parada $|x_{k+1} - x_k|/|x_{k+1}|$. Ademais, o tempo de processamento também está presente nesta análise.

Para os métodos que necessitam de uma estimativa inicial x_0 , ela foi escolhida baseado-se em uma das extremidades dos intervalos para os métodos que utilizam o **teorema de Bolzano** (1).

As funções escolhidas para serem analisadas estão listadas a seguir com os seus devidos intervalos.

- $f_1(x) = 2x^3 + 4x^3 + 3x^2 - 10x - 15$, $\bar{x} \in [0, 3]$, $x_0 = 3$, raiz em intervalo convexo.
- $f_2(x) = -x^2 + \ln(x) + 2$, $\bar{x} \in [1, 3]$, $x_0 = 3$, raiz em intervalo côncavo.
- $f_3(x) = x^4 - 11x^3 + 36x^2 - 16x - 64$, $\bar{x} \in [3.8, 4.5]$, $x_0 = 4.5$, raiz com multiplicidade.

Os resultados obtidos para as funções f_1 , f_2 e f_3 estão apresentados nas tabelas 4.6, 4.7 e 4.8 a seguir.

Tabela 4.6: Desempenho dos métodos em f_1

MÉTODO	ITERAÇÕES	ERRO RELATIVO	TEMPO (s)
Falsa posição	34	1.146×10^{-8}	6.290×10^{-5}
Bisseção	27	1.422×10^{-8}	5.674×10^{-5}
Newton	5	8.102×10^{-9}	2.097×10^{-5}
Illinois	10	1.120×10^{-9}	2.805×10^{-5}
Pegasus	9	3.432×10^{-11}	2.729×10^{-5}
Müller	4	4.695×10^{-7}	2.245×10^{-5}
Brent	8	9.480×10^{-8}	3.627×10^{-5}
Halley	3	1.472×10^{-8}	2.023×10^{-5}
Chebyshev	4	1.413×10^{-16}	2.454×10^{-5}
Newton raízes múltiplas	6	1.695×10^{-15}	2.802×10^{-5}

Fonte: Autoria própria

Tabela 4.7: Desempenho dos métodos em f_2

MÉTODO	ITERAÇÕES	ERRO RELATIVO	TEMPO (s)
Falsa posição	20	6.639×10^{-9}	3.783×10^{-5}
Bisseção	27	9.525×10^{-9}	5.211×10^{-5}
Newton	5	1.812×10^{-13}	2.063×10^{-5}
Illinois	8	1.442×10^{-11}	3.152×10^{-5}
Pegasus	7	4.504×10^{-10}	2.808×10^{-5}
Müller	4	6.447×10^{-8}	2.273×10^{-5}
Brent	7	3.743×10^{-7}	3.078×10^{-5}
Halley	3	4.079×10^{-11}	2.773×10^{-5}
Chebyshev	3	3.935×10^{-9}	2.868×10^{-5}
Newton raízes múltiplas	5	1.353×10^{-9}	2.308×10^{-5}

Fonte: Autoria própria

Tabela 4.8: Desempenho dos métodos em f_3

MÉTODO	ITERAÇÕES	ERRO RELATIVO	TEMPO (s)
Falsa posição	100	4.952×10^{-4}	1.920×10^{-4}
Bisseção	26	2.608×10^{-8}	5.687×10^{-5}
Newton	67	3.612×10^{-6}	1.163×10^{-4}
Illinois	39	5.138×10^{-8}	7.923×10^{-5}
Pegasus	59	3.576×10^{-7}	9.137×10^{-5}
Müller	54	2.289×10^{-8}	1.548×10^{-4}
Brent	43	6.982×10^{-6}	8.840×10^{-5}
Halley	24	1.773×10^{-6}	8.394×10^{-5}
Chebyshev	50	7.683×10^{-7}	1.647×10^{-4}
Newton raízes múltiplas	5	4.226×10^{-8}	2.860×10^{-5}

Fonte: Autoria própria

Os dados obtidos nas tabelas 4.6 e 4.7 demonstram o mesmo direcionamento dos testes preliminares, pois os resultados dos métodos clássicos são similares. O método de Newton já é muito consolidado para determinar raízes, e convergiu rapidamente para solução. Por outro lado, o método da bisseção apesar de contar com a vantagem de garantia de convergência, ainda é muito custoso na busca pela raiz. O algoritmo da falsa posição é muito ineficiente quando está diante de funções com anomalias.

Os métodos baseados em interpolação linear modificados são excelentes alternativas, por contar com um fator de redução de extremos fixos em seu processo iterativo, fica evidente

nos resultados a diminuição de suas iterações, e até mesmo no tempo de processamento.

Já os *métodos baseados em interpolação quadrática* são mais eficientes que os métodos clássicos (com exceção de Newton, em alguns casos), mesmo que a *complexidade computacional* de Müller e Brent sejam maior, o tempo de processamento foi pequeno pelo fato de convergirem em poucas iterações.

Nos *métodos com derivadas de segunda ordem* o número de iterações em relação aos métodos clássicos são extremamente baixos (com exceção comparando-se a Newton). Nos resultados numéricos de [Ferreira et al. \(2013\)](#) o método de Halley (isso inclui o método de Chebyshev) que tem convergência cúbica para raízes simples é mais preciso que o método de Newton, pois obtém-se uma aproximação com erro relativo menor.

Na tabela 4.8 estão apresentados os resultados dos métodos para a função f_3 , que pode ser considerado um problema patológico para a maioria dos métodos que determinam uma aproximação para raízes reais simples. A função analisada $f_3(x) = x^4 - 11x^3 + 36x^2 - 16x - 64 = (x - 4)^3(x + 1)$ tem o 4 com multiplicidade 3.

Os métodos clássicos tiveram elevado número de iterações e aumento do tempo de processamento, se destacando o método da falsa posição, que atingiu o limite de iterações estabelecido, isso demonstra a deficiência na busca por zeros múltiplos. Todos os métodos não convencionais tiveram iterações e tempo de execução aumentados, com exceção ao método de Newton para raízes múltiplas, que tem objetivo de recuperar a convergência quadrática para esse tipo de situação. Nesse sentido, quando o problema é determinar uma raiz múltipla, o método de Newton – RM é o mais indicado para se determinar uma raiz com multiplicidade.

Ainda é válido ressaltar que os métodos de Illinois, Pegasus e Müller também podem ser modificados para raízes múltiplas, pois podemos utilizar a função (3.15) para recuperar sua convergência ([Filho \(2018\)](#) apresenta como essas modificações pode ser implementadas).

Considerações Finais

O uso de métodos numéricos é imprescindível para resolver problemas não lineares, pois aparecem nas mais diversas áreas do conhecimento. A pesquisa possibilitou analisar amplamente a utilização dos métodos estudados e, dessa forma, compreender sua aplicação em diversos tipos de funções que possuem características distintas. Assim tem-se métodos com diferentes abordagens que proporcionam mais eficiência em determinado tipo de situação comparado aos métodos já conhecidos.

Com análise dos dados obtidos podemos observar que as modificações presentes nos métodos não convencionais funcionam perfeitamente, e podem ser facilmente implementados em qualquer linguagem de programação. Vale ressaltar a importância do ambiente de programação `matlab`, que possibilitou a codificação dos métodos e a plotagem de todas as figuras presentes neste trabalho, além de resultados satisfatórios.

Nossa análise baseou-se na escolha de funções com características intervalares mais complexas para comparar a eficiência dos métodos estudados. De modo geral, os métodos de Illinois e Pegasus são substitutos diretos dos métodos da Falsa Posição, e até mesmo do método das Secantes. Os métodos de Müller e Brent são uma ótima escolha quando o método de Newton necessita do cálculo analítico de funções mais complexas. Os métodos de Halley, Chebyshev e Newton–RM são modificações que contam com derivadas segunda em seu processo iterativo. Halley e Chebyshev por exemplo são mais eficientes que o método tradicional de Newton, pois conseguem convergir mais rapidamente para solução. Por fim, o método de Newton para raízes múltiplas é robusto tanto na determinação de raízes simples quando múltiplas.

Neste trabalho foram abordados três classes distintas para determinar raízes reais simples/múltiplas, mas existem outros métodos até mais eficientes que os estudados aqui. Assim é pertinente destacar a investigação de outros algoritmos de determinação, tanto de raízes reais quando raízes complexas.

Referências

- BURDEN, R. L.; FAIRES, J. D.; BURDEN, A. M. *Análise numérica*. 3. ed. São Paulo, SP: Cengage Learning, 2016. ISBN 9788522106011. [25](#)
- CHAPRA, S. C. *Métodos Numéricos Aplicados com MATLAB® para Engenheiros e Cientistas*. 3. ed. Porto Alegre, RS: AMGH Editora, 2013. [13](#), [28](#), [29](#)
- CHAPRA, S. C.; CANALE, R. P. *Métodos Numéricos para Engenharia*. 5. ed. São Paulo, SP: McGraw Hill Brasil, 2011. [18](#), [25](#), [35](#)
- DOWELL, M.; JARRATT, P. A modified regula falsi method for computing the root of an equation. 1971. [21](#)
- DOWELL, M.; JARRATT, P. The “pegasus” method for computing the root of an equation. *BIT Numerical Mathematics*, Springer, v. 12, p. 503–508, 1972. [23](#)
- ESMAEILI, H.; REZAEI, A. A uniparametric family of modifications for Chebyshev’s method. *Lecturas matemáticas*, Sociedad Colombiana de Matemáticas, v. 33, n. 2, p. 95–106, 2012. [33](#)
- EUSTAQUIO, R. G. *Classe de métodos Chebyshev–Halley inexata livre de tensores com convergência cúbica para resolução de sistemas não lineares e um estudo sobre o raio de convergência*. Tese (Doutorado), 2013. [30](#), [31](#), [33](#)
- FERREIRA, D. G.; MELLO, M. P. Derivadas de ordem superior-método de Halley. 2012. [30](#), [32](#)
- FERREIRA, D. G. et al. *Métodos de otimização de terceira ordem*. Dissertação (Mestrado) — Universidade Estadual de Campinas (UNICAMP), 2013. [17](#), [30](#), [31](#), [32](#), [49](#)
- FILHO, F. F. *Algoritmos numéricos: uma abordagem moderna de cálculo numérico*. São Paulo, SP: LTC, 2018. [16](#), [21](#), [25](#), [27](#), [28](#), [49](#)
- FRANCO, N. B. *Cálculo numérico*. São Paulo, SP: Pearson, 2006. [17](#), [20](#)
- GALDINO, S. A family of regula falsi root-finding methods. Recife, PE, v. 1, p. 514–517, 2011. [24](#)
- GANCHOVSKI, N.; TRAYKOV, A. Modified Anderson-Bjork’s method for solving non-linear equations in structural mechanics. v. 1276, n. 1, p. 012010, 2023. [37](#)
- KEY, K. Is the fast hankel transform faster than quadrature?. *Geophysics*, v. 77, n. 3, p. F21–F30, May-Jun 2012. N/a. [40](#)

- MOLER, C. B. *Numerical computing with MATLAB*. Philadelphia, EUA: SIAM, 2004. 29
- MORAIS, W. L. d. Análise comparativa de desempenho do método Van Wijngaarden-Dekker-Brent (VWDB) em relação a outros métodos numéricos já conhecidos para obtenção de zero de funções. Universidade Federal Rural do Semi-Árido, 2021. 27
- NASCIMENTO, D. A. d. et al. Métodos para encontrar raízes exatas e aproximadas de funções polinomiais até o 4º grau. Universidade Federal da Paraíba, 2015. 13
- OBSIEGER, B. Numerical methods ii-roots and equation systems. Boris Obsieger, 2015. 25
- PALENZUELA, P. J. F. et al. Algoritmos tipo Illinois para la resolución numérica de ecuaciones no lineales y su implementación en matlab. 2019. 20, 22
- RALSTON, A.; RABINOWITZ, P. A first course in numerical analysis. Courier Corporation, 2001. 35, 39
- SOUZA, E. A. de. Métodos iterativos para problemas nao lineares. v. 12, 2015. 14, 38
- SOUZA, Q. E. d. O. et al. Comparativo entre os métodos de Newton-Raphson e das raízes múltiplas no cálculo de raízes de funções. Universidade Federal Rural do Semi-Árido, 2018. 36
- SOUZA, Q. E. de O. et al. Estudo comparativo entre os métodos de newton e das raízes múltiplas. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 6, n. 2, 2018. 35
- TEIXEIRA, T.; SILVA, I. Método do formigueiro para encontrar os zeros de funções reais. *Revista Eletrônica de Iniciação Científica em Computação*, v. 16, n. 4, 2018. 28, 29
- TRAUB, J. F. *Iterative methods for the solution of equations*. New York, EUA: American Mathematical Soc., 1982. v. 312. 33

Apêndice



Códigos confeccionados para o desenvolvimento do trabalho

Código A.1: Método da Bissecção

```
1 % UNIVERSIDADE FEDERAL DO PARA
2 % Autor: Henrique Pinheiro Galvao
3 % Titulo: Metodo da Bissecao
4 % Data: 01/11/2023
5
6 function [x, iter ,aprox] = Bissecao(f,a,b,tol)
7 format long
8 fa = f(a); fb = f(b);
9 iter = 0;
10 aprox = [];
11 absx = 1e3;
12 errel = 1e8 * tol * absx;
13 %-----
14 if fa * fb > 0
15     error('Mesmo sinal')
16 end
17 %-----
18 while errel > tol * max([1,absx])
19     x = (a + b)/2;
20     absx = abs(x);
21     errel = min([abs(x-a)./abs(x),abs(x-b)./abs(x)]);
22     if f(a) * f(x) < 0
23         b = x;
```

```

24         else
25             a = x;
26         end
27 %-----
28 aux = iter;
29 iter = aux + 1;
30 aprox(iter) = x;
31         if iter > 100
32             break
33         end
34 end

```

Código A.2: Método da Falsa Posição

```

1 % UNIVERSIDADE FEDERAL DO PARA
2 % Autor: Henrique Pinheiro Galvao
3 % Titulo: Metodo da Falsa Posicao
4 % Data: 01/11/2023
5
6 function [x, iter, aprox] = Falsaposicao(f,a,b,tol)
7 format long
8 fa = f(a); fb = f(b); iter = 0; aprox = [];
9 absx = 1e3; errel = 1e8 * tol * absx;
10 %-----
11 if fa * fb > 0
12     error('Mesmo sinal')
13 end
14 %-----
15 while errel > tol * max([1, absx])
16     x = b-fb*(b-a)/(fb-fa);
17     fx = f(x);
18     absx = abs(x);
19     errel = min([abs(x-a)./abs(x), abs(x-b)./abs(x)]);
20 %-----
21 if isnan(fx)

```

```

22     fx = fb;
23     x = b;
24     break
25 end
26 %-----
27 if isnan(x)
28     fx = fb; x = b;
29     break
30 end
31 %-----
32 if sign(fx)~=sign(fb)
33     a = x; fa = fx;
34     else
35     b = x; fb = fx;
36 end
37 iter = iter + 1;  aprox(iter) = x;
38 if iter > 100
39     break
40 end
41 end

```

Código A.3: Método de Illinois

```

1 % UNIVERSIDADE FEDERAL DO PARA
2 % Autor: Henrique Pinheiro Galvao
3 % Titulo: Metodo de Illinois
4 % Data: 01/11/2023
5
6 function [x, iter, aprox] = Illinois(f,a,b,tol)
7 fa = f(a); fb = f(b); iter = 0; aprox = [];
8 absx = 1e3; errel = 1e8 * tol * absx;
9 %-----
10 if fa * fb > 0
11     error('Mesmo sinal')
12 end

```

```

13 %-----
14 while errel > tol * max([1,absx])
15     x = b-fb*(b-a)/(fb-fa);
16     fx = f(x); absx = abs(x);
17     errel = min([abs(x-a)./abs(x),abs(x-b)./abs(x)]);
18 %-----
19 if isnan(fx)
20     fx = fb; x = b;
21     break
22 end
23 %-----
24 if isnan(x)
25     fx = fb; x=b;
26     break
27 end
28 %-----
29 if sign(fx)~= sign(fb)
30     a = b; fa = fb;
31     else
32     fa = fa/2; % Passo Modificado
33 end
34 %-----
35 iter = iter + 1; b = x; fb = fx;
36 aprox(iter) = x;
37 if iter > 100
38     break
39 end
40 end
41 end

```

Código A.4: Método de Pegasus

```

1 % UNIVERSIDADE FEDERAL DO PARA
2 % Autor: Henrique Pinheiro Galvao
3 % Titulo: Metodo de Pegasus

```

```

4  % Data: 01/11/2023
5
6  function [x, iter, aprox] = Pegasus(f, a, b, tol)
7  fa = f(a); fb = f(b); iter = 0; aprox = [];
8  absx = 1e3; errel = 1e8 * tol * absx;
9  %-----
10 if fa * fb > 0
11     error('Mesmo sinal')
12 end
13 %-----
14 while errel > tol * max([1, absx])
15     x = b - fb * (b - a) / (fb - fa);
16     fx = f(x);
17     absx = abs(x);
18     errel = min([abs(x - a) ./ abs(x), abs(x - b) ./ abs(x)]);
19 %-----
20 if isnan(fx)
21     fx = fb; x = b;
22     break
23 end
24 %-----
25 if isnan(x)
26     fx = fb; x = b;
27 break
28 end
29 %-----
30 if sign(fx) ~= sign(fb)
31     a = b; fa = fb;
32     else
33     fa = (fa .* fb) / (fb + fx); % Passo Modificado
34 end
35 %-----
36 b = x; fb = fx; iter = iter + 1; aprox(iter) = x;
37 if iter > 100
38     break
39 end

```

```

40 end
41 end

```

Código A.5: Método de Müller

```

1  % UNIVERSIDADE FEDERAL DO PARA
2  % Autor: Henrique Pinheiro Galvao
3  % Titulo: Metodo de Muller
4  % Data: 01/11/2023
5
6  function [x, iter, aprox] = Muller(f,a,b,tol)
7  format long
8  iter = 0; aprox = [];
9  absx = 1e3; errel = 1e8 * tol * absx;
10 fa = f(a); fb = f(b);           %Avalia a funcao em (a) e (b)
11 x = (a + b)/2; fx = f(x);       %Obtencao do terceiro ponto via
    Bissecacao
12 while errel > tol * max([1,absx])
13     %Computar os coeficientes da parabola
14     ha = x - a; hb = b - x;
15     da = (fa - fx)/ha; db = (fb - fx)/hb;
16     AA = (da + db)/(ha + hb);
17     BB = (ha * db - hb * da)/(ha + hb);
18     CC = fx;
19     % Computa o denominador do metodo;
20     h = sqrt(BB^2 - 4 * AA * CC);
21     if abs(BB + h) > abs(BB - h)
22         raiz = - 2 * CC/(BB + h);
23     else
24         raiz = - 2 * CC/(BB - h);
25     end
26     t = x; ft = fx;
27     x = x + raiz; fx = f(x);
28     absx = abs(x);
29     if abs(raiz) <= tol && abs(fx) <= iter

```

```

30         break
31     end
32     %Atualizacao das variaveis
33     a = b; fa = fb;
34     b = t; fb = ft;
35     errel = abs(t - x)./abs(t);
36     %-----
37     iter = iter + 1; aprox(iter) = x;
38     if iter > 100
39         break
40     end
41 end
42 end

```

Código A.6: Método de Brent

```

1  % UNIVERSIDADE FEDERAL DO PARA
2  % Autor: Henrique Pinheiro Galvao
3  % Titulo: Metodo de Brent
4  % Data: 01/11/2023
5
6  function [x, iter, aprox] = Brent(f,a,b,tol)
7  format long
8  iter = 0; aprox = [];
9  absx = 1e3; errel = 1e8 * tol * absx;
10 fa = f(a); fb = f(b); %Avalia a funcao em (a) e (b)
11 if fa * fb > 0 % Verifica o sinal do intervalo
12     error('Funcao tem o mesmo sinal nesse intervalo!!')
13 end
14 %-----
15 x = b; fx = f(x); fx = fb;
16 while errel > tol * max([1,absx])
17     %Alteracao a, b e c para que b seja a melhor aproximacao da
18     raiz
19     if fb * fx > 0

```

```

19         x = a; fx = fa;
20         d = b - a; e = d;
21     end
22 %-----
23     if abs(fx) < abs(fb)
24         a = b; b = x; x = a;
25         fa = fb; fb = fx; fx = fa;
26     end
27 %-----
28     toler = 2*eps*abs(b)+0.5*tol;
29     xm = 0.5*(x - b);
30     if abs(xm) <= toler && fb == 0
31         break
32     end
33     % Escolha entre Interpolacao e Bissecao!
34     if abs(e) >= toler && abs(fa) > abs(fb)
35         s = fb/fa;
36         if a == x      %Metodo das Secantes!
37             p = 2*xm*s; q = 1-s;
38         else %interpolacao Quadratica Inversa!
39             q = fa/fx; r = fb/fx;
40             p = s*(2*xm*q*(q-r)-(b-a)*(r-1));
41             q = (q-1)*(r-1)*(s-1);
42         end
43 %-----
44         if p > 0
45             q = -q;
46         else
47             p = abs(p);
48         end
49 %-----
50         if 2*p < min(3*xm*q - abs(toler*q),abs(e*q))
51             e = d; d = p/q;
52         else %Uso da bissecao devido a falha na interpolacao!
53             d = xm; e = d;
54         end

```

```

55     %Uso da bissecao por causa da reducao lenta dos limites!
56     else
57         d = xm; e = d;
58     end
59     %-----
60     a = b;
61     fa = fb; %Calcula a nova estimativa;
62     if abs(d) > toler
63         b = b + d;
64     else
65         b = b + sign(xm)*toler;
66     end
67     errel = abs(b-x)/abs(b);
68     %-----
69     fb = f(b);
70     iter = iter + 1;
71     aprox(iter) = b;
72     if iter > 100
73         break
74     end
75 end
76 end

```

Código A.7: Método de Newton

```

1 % UNIVERSIDADE FEDERAL DO PARA
2 % Autor: Henrique Pinheiro Galvao
3 % Titulo: Metodo de Newton
4 % Data: 01/11/2023
5
6 function [x, iter, aprox] = Newton(f,df,x0,tol)
7 format long
8 iter = 1; %Compensa uma iteracao feita fora do loop!
9 aprox = [];
10 x = x0 - f(x0)/df(x0);

```

```

11 absx = abs(x); errel = abs(x-x0)/absx;
12 %-----
13 while errel > tol * max([1,absx])
14     xold = x;
15     x = x - f(x) / df(x);
16     absx = abs(x);
17     errel = abs(x-xold)/abs(x);
18 %-----
19 iter = iter + 1;
20 aprox(iter) = x;
21 if iter > 100
22     break
23 end
24 end
25 end

```

Código A.8: Método de Halley

```

1 % UNIVERSIDADE FEDERAL DO PARA
2 % Autor: Henrique Pinheiro Galvao
3 % Titulo: Metodo de Halley
4 % Data: 01/11/2023
5
6 function [x, iter, aprox] = Halley(f,df,df2,x0,tol)
7 format long
8 iter = 1; %Compensa uma iteracao feita fora do loop!
9 aprox = [];
10 x = x0 - (2 * f(x0) * df(x0)) / (2 * df(x0)^2 - f(x0) * df2(x0));
11 absx = abs(x); errel = abs(x-x0)/absx;
12 %-----
13 while errel > tol * max([1,absx])
14     xold = x;
15     x = x - (2 * f(x) * df(x)) / (2 * df(x)^2 - f(x) * df2(x));
16     absx = abs(x);
17     errel = abs(x-xold)/abs(x);

```

```

18 %-----
19 iter = iter + 1;
20 aprox(iter) = x;
21 if iter > 100
22     break
23 end
24 end
25 end

```

Código A.9: Método de Chebyshev

```

1 % UNIVERSIDADE FEDERAL DO PARA
2 % Autor: Henrique Pinheiro Galvao
3 % Titulo: Metodo de Chebyshev
4 % Data: 01/11/2023
5
6 function [x, iter, aprox] = Halley(f, df, df2, x0, tol)
7 format long
8 iter = 1; %Compensa uma iteracao feita fora do loop!
9 aprox = [];
10 x = x0 - (f(x0)/df(x0))*(1 + (f(x0)*df2(x0) / (2 * df(x0)^2)));
11 absx = abs(x); errel = abs(x-x0)/absx;
12 %-----
13 while errel > tol * max([1, absx])
14     xold = x;
15     x = x - (f(x)/df(x))*(1 + (f(x)*df2(x) / (2 * df(x)^2)));
16     absx = abs(x);
17     errel = abs(x-xold)/abs(x);
18 %-----
19 iter = iter + 1;
20 aprox(iter) = x;
21 if iter > 100
22     break
23 end
24 end

```

25 `end`

Código A.10: Método Newton–RM

```

1  % UNIVERSIDADE FEDERAL DO PARA
2  % Autor: Henrique Pinheiro Galvao
3  % Titulo: Metodo de Newton–RM
4  % Data: 01/11/2023
5
6  function [x, iter, aprox] = NewtonRM(f, df, df2, x0, tol)
7  format long
8  iter = 1; %Compensa uma iteracao feita fora do loop!
9  aprox = [];
10 x = x0 - (f(x0)*df(x0))/(df(x0)^2 - f(x0)*df2(x0));
11 absx = abs(x); errel = abs(x-x0)/absx;
12 %-----
13 while errel > tol * max([1, absx])
14     xold = x;
15     x = x - (f(x)*df(x))/(df(x)^2 - f(x)*df2(x));
16     absx = abs(x);
17     errel = abs(x-xold)/abs(x);
18 %-----
19 iter = iter + 1;
20 aprox(iter) = x;
21     if iter > 100
22 break
23 end
24 end
25 end

```

Código A.11: Média aritmética dos tempos

```
1  clc ;
2  tempTotal = 0;  %Tempo inicial!
3  numTeste = 50;  %Numero de testes!
4  for i = 1:numTeste
5      tic ;
6      [x, iter ,aprox] = Metodo(f,a,b,tol);
7      T = toc ;
8      tempTotal = tempTotal + T; %Somatorio dos tempos!
9  end
10
11 tempMedio = tempTotal / numTeste; %Media aritmetica dos tempos!
12 disp([ 'Tempo medio de execucao:',num2str(tempMedio), ' segundos' ]);
```
