
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE MATEMÁTICA

Nome Sobrenome

# Métodos Numéricos

SEROPÉDICA
2025


Nome Sobrenome

# MÉTODOS NUMÉRICOS

Monografia apresentada à Banca Examinadora da Universidade Federal Rural do Rio de Janeiro, como parte dos requisitos para obtenção do título de Bacharel em Matemática sob orientação do Prof. Dr. Nome Sobrenome do Orientador

SEROPÉDICA

2025

**UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO**
**INSTITUTO DE CIÊNCIAS EXATAS**
**DEPARTAMENTO DE MATEMÁTICA**

**COORDENAÇÃO DO CURSO DE GRADUAÇÃO EM MATEMÁTICA**

A monografia "Métodos Numéricos", apresentada e defendida por NOME SOBRENOME, matrícula 2022019000-0, foi aprovada pela Banca Examinadora com conceito "X", recebendo o número 000.

Seropédica, 12 de setembro de 2025

BANCA EXAMINADORA:

Prof. Dr. Presidente da Banca
Orientador

Prof. Dr. Membro 1
Convidado 1

Prof. Dr. Membro 2
Convidado 1

# Agradecimentos

Aqui está um agradecimento.

# Resumo

Aqui está um resumo.

# Abstract

Here is an (optional) abstract.

# Sumário

**Introdução** **ii**

**1 Pontos Flutuantes** **1**

1.1 Aritmética de Ponto Flutuante . . . . . 2

1.1.1 Precisão Simples e Precisão Dupla . . . . . 3

1.1.2 Representação de Números em Sistemas de Ponto Flutuante . . . . 6

1.1.3 Representação Especial do Zero . . . . . 9

1.2 Erros e Limitações . . . . . 11

1.3 Perda de Significância em Somas . . . . . 11

1.3.1 Análise de Instabilidades e Casos Peculiares . . . . . 12

1.3.2 Discussão . . . . . 16

**2 Métodos Iterativos para Zeros de Função** **17**

2.1 Localização de Raízes . . . . . 17

2.2 Método do Ponto Fixo . . . . . 18

2.2.1 Ordem de convergência . . . . . 21

2.3 Método de Newton-Raphson . . . . . 21

# Introdução

# Capítulo 1

# Pontos Flutuantes

Na matemática, um sistema numérico é um conjunto de regras e símbolos utilizados para representar quantidades através do que chamamos de números. Existem dois tipos de sistemas: os posicionais e os aditivos.

O sistema aditivo é aquele em que os números são representados pelas somas dos valores dos símbolos, geralmente agrupados lado a lado em ordem decrescente, como, por exemplo, os sistemas romano e egípcio. Já o sistema posicional leva em conta não só os dígitos mas também a posição que eles ocupam no número. A quantidade de símbolos diferentes que são utilizados para representar os dígitos está ligada à **base** desse sistema, e cada posição do dígito no número refere-se a uma potência dessa base. Por exemplo, no sistema decimal (base 10), usamos os dígitos de 0 a 9. No sistema binário (base 2), usamos os dígitos 0 e 1. E já no sistema hexadecimal (base 16), usamos de 0 a 9 e as letras A a F (que representam 10 a 15).

Com essas diferentes formas de representar um número, a escolha do sistema depende do contexto e da aplicação. No uso cotidiano, a base decimal é a mais utilizada. Já as bases binária e hexadecimal, são amplamente utilizadas na ciência da computação em operações aritméticas dos processadores e em algumas linguagens de programação para endereçamento de memória.

Um número $N$ pode ser representado em uma base $b$ no seguinte formato

$$N = \pm \sum_{i=-k}^{n} d_i b^i, \tag{1.1}$$

em que $d_i$ são os dígitos na base $b$, $k$ é o número de casas decimais à direita do ponto, e $n+1+k$ é o número de dígitos significativos. Vejamos alguns exemplos.

**Exemplo 1.0.1.** Vamos escrever o número 13 nas bases 10 e 2.

- Número na base decimal: $13 = 1 \times 10^1 + 3 \times 10^0 = 13_{10}$
- Número na base binária: $13 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1101_2$

**Exemplo 1.0.2.** Agora vamos escrever o número 3,5625 nas bases 10 e 2.

- Número na base decimal:

$$3{,}5625 = 3 \times 10^0 + 5 \times 10^{-1} + 6 \times 10^{-2} + 2 \times 10^{-3} + 5 \times 10^{-4} = 3{,}5625_{10}$$

- Número na base binária:

$$3{,}5625 = 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 11{,}1001_2$$

## 1.1 Aritmética de Ponto Flutuante

A *aritmética de ponto flutuante* é o sistema adotado por computadores para que lidem com números reais utilizando uma notação compacta e eficaz. Essa técnica é utilizada para representar e manipular números reais de forma prática e eficiente. Ela permite representar números de grandezas diversas, que não podem ser armazenados com precisão, utilizando apenas números inteiros.

Um sistema de ponto flutuante $F$ pode ser definido como

$$F(\beta, t, L, U)$$

cuja representação normalizada de um número real N nesse sistema é dada por

$$N = \pm(0.d_1 d_2 ... d_t)_\beta \times \beta^e \tag{1.2}$$

em que

- $N$ é o número real;

- $\beta$ é a base que a máquina opera;
- $t$ é o número de dígitos na mantissa, tal que $0 \leq d_j \leq \beta - 1$, j = 1, ..., t, $d_1 \neq 0$;
- $L$ é o menor expoente inteiro;
- $U$ é o maior expoente inteiro;
- $e$ é o expoente inteiro no intervalo [$L$,$U$].

No padrão IEEE 754 (usado na maioria dos sistemas eletrônicos), um número de ponto flutuante é dividido em três partes:

- **Sinal (S)**: 1 bit indicando se o número é positivo ($S = 0$) ou negativo ($S = 1$),
- **Expoente (E)**: campo que representa o expoente com viés (bias),
- **Mantissa (M)**: parte fracionária significativa do número.

A fórmula completa de reconstrução do número é:

$$\text{Valor} = (-1)^S \times (1.M) \times 2^{E-\text{bias}}$$

onde:

- $S$ é o bit de sinal,
- $1.M$ indica que há um bit implícito "1"antes da mantissa nos números normalizados,
- bias é um valor constante que depende da precisão (por exemplo, 127 para 32 bits).

### 1.1.1 Precisão Simples e Precisão Dupla

Em sistemas computacionais, os números em ponto flutuante podem ser representados em diferentes níveis de precisão. Os dois mais comuns são:

- **Precisão Simples (32 bits)**
- **Precisão Dupla (64 bits)**

Esses formatos seguem o padrão `IEEE 754` de representação binária de números reais.

**Comparação entre os formatos**

| **Característica** | **Precisão Simples (32 bits)** | **Precisão Dupla (64 bits)** |
|---|---|---|
| Bits totais | 32 | 64 |
| Bit de sinal | 1 | 1 |
| Bits de expoente | 8 | 11 |
| Bits de mantissa | 23 | 52 |
| Bias | 127 | 1023 |
| Intervalo do expoente real | $-126$ a $+127$ | $-1022$ a $+1023$ |
| Precisão (dígitos decimais) | Aproximadamente 7 | Aproximadamente 16 |

**Exemplo: Representação em Precisão Simples**

Considere o número decimal $x = -12{,}25$. Sua representação em binário é:

$$x = -1100{,}01_2 = -1{,}10001 \times 2^3$$

Formato:

- Sinal: $s = 1$
- Mantissa (sem o bit oculto): 10001000000000000000000
- Expoente: $e = 3 + 127 = 130 = 10000010_2$

Portanto, o número seria representado, em binário de 32 bits, como:

$$\boxed{1\ 10000010\ 10001000000000000000000}$$

**Exemplo: Representação em Precisão Dupla**

Vamos representar o número decimal $x = 12{,}375$ em ponto flutuante com **precisão dupla (64 bits)**.

**1. Conversão para binário:**

$$12{,}375_{10} = 1100{,}011_2 = 1{,}100011 \times 2^3$$

**2. Identificação dos componentes:**

- **Sinal (s)**: Como o número é positivo, $s = 0$
- **Expoente real (e)**: 3
- **Bias**: Para precisão dupla, bias = 1023
- **Expoente com bias**: $e + \text{bias} = 3 + 1023 = 1026$
- **Expoente em binário (11 bits)**: $1026_{10} = 10000000010_2$
- **Mantissa (m)**: Os bits após o ponto da parte fracionária normalizada: $100011000000\ldots$ (completando até 52 bits)

**3. Representação final (64 bits):**

$$\boxed{0\ 10000000010\ 1000110000000000000000000000000000000000000000000000}$$

Essa é a representação de 12,375 em ponto flutuante com precisão dupla.

**Resumo:**

- **Bits de sinal**: 0
- **Bits do expoente**: 10000000010
- **Bits da mantissa**: 100011 seguidos de zeros até completar 52 bits

**Considerações**

A escolha entre precisão simples e dupla depende da aplicação:

- **Precisão Simples**: adequada para aplicações com memória limitada e que não exigem alta precisão.
- **Precisão Dupla**: usada em aplicações científicas, cálculos de engenharia, simulações e algoritmos numéricos mais sensíveis. Apesar do ganho de precisão, o uso de precisão dupla demanda mais memória e tempo de processamento.

### 1.1.2 Representação de Números em Sistemas de Ponto Flutuante

Em máquinas que operam em sistemas de ponto flutuante, apenas um subconjunto finito de $\mathbb{R}$ pode ser representado de maneira exata. Por isso, frequentemente, é necessário limitar a quantidade de dígitos significativos na representação de números a fim de adequá-los ao sistema que a máquina opera. Dois dos principais processos empregados para este fim são o **truncamento** e o **arredondamento**.

O truncamento consiste na supressão de todos os dígitos após uma determinada posição, sem qualquer ajuste adicional no último dígito mantido. Formalmente, dado um número real $x$, sua aproximação truncada com $n$ dígitos na base $b$ é expressa por

$$T(x) = \sum_{i=-k}^{n-1} d_i b^i$$

onde os dígitos $d_i$ com $i < -k$ são descartados.

O erro introduzido por este processo, dado por $E_T = x - T(x)$, é denominado *erro de truncamento*. Ele é limitado superiormente por

$$|x - T(x)| < b^{-k}. \tag{1.3}$$

**Exemplo 1.1.1.** Considere a aproximação com oito casas decimais de $\pi = 3,14159265$. Para truncar $\pi$ com precisão de 4 casas decimais, descartamos todos os termos da sexta casa em diante. Assim, o valor truncado fica $T(\pi) = 3,1415$. O erro do truncamento é, nesse caso, $E_T = 3,14159265 - 3,1415 = 0,00009265$. Diante disso, podemos observar que $|E_T| < 10^{-4}$.

O arredondamento, por outro lado, ajusta o último dígito mantido com base no valor do primeiro dígito descartado, buscando minimizar o erro absoluto da aproximação. No arredondamento simétrico (ou clássico), se o primeiro dígito descartado for maior ou igual a $\dfrac{b^{-n}}{2}$, incrementa-se o último dígito mantido em uma unidade; caso contrário, seu valor permanece inalterado.

Seja $x$ um número real e $R(x)$ sua aproximação arredondada com $n$ dígitos na base $b$. O erro de arredondamento satisfaz:

$$|x - R(x)| \leq \frac{1}{2}b^{-n}$$

**Exemplo 1.1.2.** Ainda considerando a mesma aproximação de $\pi = 3,14159265$. Para arredondar $\pi$ com precisão de 4 casas decimais, vamos analisar o número da próxima casa em que queremos arredondar. Nesse caso, o número na quinta posição é 9, então vamos arredondar para cima.

$$R(\pi) = 3,1416$$

O erro do arredondamento é, nesse caso,

$$E_R = 3,14159265 - 3,1416 = -0,00000735.$$

Diante disso, podemos observar que $|E_R| \leq \frac{1}{2}10^{-4}$ .

Em geral, o erro máximo introduzido pelo arredondamento é metade daquele introduzido pelo truncamento, razão pela qual o arredondamento tende a produzir aproximações mais precisas.

Para compreendermos melhor as limitações na representação de números em um sistema de ponto flutuante, vamos explorar o exemplo a seguir. Suponha que uma máquina opere no sistema $F(10, 5, -5, 5)$. Nesse sistema, os números serão representados da seguinte maneira

$$\pm(0.d_1d_2...d_t) \times 10^e, 0 \leq |d_j| \leq 9, d_1 \neq 0, e \in [-5, 5]. \tag{1.4}$$

O menor valor, em módulo, representado nesse sistema é

$$m = 0.10000 \times 10^{-5} = 10^{-6},$$

enquanto que o maior é

$$M = 0.99999 \times 10^5 = 99999.$$

O subconjunto $G \subset \mathbb{R}$ definido como

$$G = \{x \in \mathbb{R} \mid m \leq |x| \leq M\}$$

$$G = \{x \in \mathbb{R} \mid m \leq |x| \leq M\}$$

é o conjunto dos números que são representáveis por esse sistema de ponto flutuante. Nesse conjunto:

- $m$ é o menor valor positivo representável;
- $M$ é o maior valor representável;
- Os números são representados na forma normalizada $\bar{x} = \pm 0.d_1d_2d_3d_4d_5 \times 10^e$.

Dado um número real $x$, as seguintes situações podem ocorrer:

**Caso 1: $x \in G$ (Número representável)**

Seja $x = 12237,76$.

Na forma normalizada temos $x = 0,1223776 \times 10^5$. Porém, esse número não pode ser representado precisamente no sistema $F$ e, portanto, precisamos aplicar uma das técnicas de aproximação. Utilizando o truncamento o resultado é $\bar{x} = 0{,}12237 \times 10^5$. Já com o arredondamento $\bar{x} = 0{,}12238 \times 10^5$.

O número está dentro da faixa de expoente permitida e é representável com perda controlada de precisão.

**Caso 2: $|x| < m$ (Underflow)**

Seja $x = 0{,}582 \times 10^{-6}$.

O expoente é –6, menor que o limite inferior $L = -5$. Portanto, o número não pode ser representado e ocorre **underflow**. Nesse caso, na maioria das vezes o valor é tratado como zero.

**Caso 3: $|x| > M$ (Overflow)**

Seja $x = 0{,}927 \times 10^6$

O expoente é +6, maior que $U = 5$. Portanto, ocorre **overflow** e o número não pode ser representado precisamente. Neste caso, o valor pode ser tratado como infinito ou como uma flag indicando o **overflow**.

### 1.1.3 Representação Especial do Zero

Na representação de ponto flutuante, um número real é geralmente expresso na forma normalizada:

$$N = \pm(d_1{,}d_2d_3 \ldots d_t)_\beta \times \beta^e,$$

com $d_1 \neq 0$, garantindo o aproveitamento máximo da precisão disponível e evitando representações redundantes. Contudo, o número zero não pode ser representado nesta forma, pois exigiria $d_1 = 0$, o que contraria a normalização.

Assim, o zero recebe uma *representação especial* denotada por

$$N = \pm(0{,}000 \ldots 0_t)_\beta \times \beta^{L-1},$$

em que $L$ é o menor expoente permitido no sistema. Este tratamento especial assegura que o zero seja manipulado de forma única e consistente dentro do sistema de ponto flutuante.

**Prevenção da perda de significância**

Em operações numéricas envolvendo números de ordens de grandeza muito distintas, pode ocorrer a *perda de significância*, isto é, quando os dígitos significativos de um número pequeno são eliminados na soma ou subtração com um número muito maior.

Considere um sistema de ponto flutuante $F(10, 7, -5, 5)$. Sejam $x = 0{,}000 \times 10^{-6}$ (ou seja, o número zero), $y = 0{,}276 \times 10^{-2}$.

Na operação de soma:

$$x + y = 0{,}276 \times 10^{-2},$$

como $x$ é exatamente zero, o resultado mantém integralmente os dígitos significativos de $y$.

Se o zero não tivesse uma representação especial e fosse tratado como um subnormal com expoente mínimo, o alinhamento das mantissas poderia comprometer a precisão de $y$, deslocando seus dígitos significativos e resultando em perda de informação.

Portanto, a representação especial do zero evita este problema, preservando a precisão e assegurando a estabilidade numérica das operações.

Vejamos a seguir uma comparação da técnicas de arredondamento e de truncamento.

Considere uma máquina decimal com 3 dígitos na mantissa e expoentes variando de –4 a 4:

| **Número Real** | **Arredondamento** | $E_R$ | **Truncamento** | $E_T$ |
|---|---|---|---|---|
| 5,678 | $0{,}568 \times 10^1$ | $0{,}2 \times 10^{-3}$ | $0{,}567 \times 10^1$ | $0{,}8 \times 10^{-3}$ |
| –192,73 | $-0{,}193 \times 10^3$ | $0{,}27 \times 10^1$ | $-0{,}192 \times 10^3$ | $0{,}73 \times 10^1$ |
| 3,14159 | $0{,}314 \times 10^1$ | $0{,}159 \times 10^{-2}$ | $0{,}314 \times 10^1$ | $0{,}159 \times 10^{-2}$ |
| 0,0000063 | Underflow | | | |
| 920000,0 | Overflow | | | |

## 1.2 Erros e Limitações

Erros em operações com pontos flutuantes podem se propagar e aumentar em cálculos mais complexos. Por exemplo, pequenos erros de arredondamento em etapas iniciais podem afetar significativamente o resultado final, especialmente em somas repetitivas ou subtrações de números muito próximos. Isso torna importante considerar a ordem das operações e o impacto da precisão em aplicações sensíveis.

## 1.3 Perda de Significância em Somas

A perda de significância (também conhecida como cancelamento catastrófico) ocorre de modo mais severo quando há grande diferença de ordem de grandeza entre os números envolvidos na operação. Por exemplo, considere:

$$x = 1{,}000 \times 10^4 \quad \text{e} \quad y = 0{,}276 \times 10^{-2}.$$

Para realizar a soma, ambos os operandos precisam ser expressos com o mesmo expoente:

$$x = 1{,}000000000 \times 10^4, \quad y = 0{,}000000276 \times 10^4.$$

A soma exata seria:

$$x + y = (1{,}000000000 + 0{,}000000276) \times 10^4 = 1{,}000000276 \times 10^4.$$

Contudo, devido à precisão limitada do sistema (7 dígitos significativos), a aritmética de ponto flutuante armazena apenas:

$$x + y \approx 1{,}0000002 \times 10^4.$$

Uma parte do termo $y$ é completamente desprezada, e a soma não resulta numericamente igual a $x + y$, evidenciando a perda catastrófica de significância.

Outro caso peculiar é a soma de um número muito grande com uma sequência de números pequenos. Dependendo da ordem em que as somas são realizadas, o número grande pode "mascarar"os pequenos, resultando em diferentes valores finais.

Por exemplo:

$$S = 10^8 + 10^{-1} + 10^{-2} + 10^{-3} + \ldots + 10^{-10}.$$

Se somarmos primeiro o número grande ($10^8$) e depois os números pequenos, muitos destes podem ser ignorados devido à falta de precisão da mantissa. Por outro lado, ao somar os números pequenos antes, o valor final será mais próximo do esperado.

Para ilustrar, suponha a seguinte ordem de cálculo:

- Caso 1: $S = 10^8 + (10^{-1} + 10^{-2} + \ldots + 10^{-10})$.
- Caso 2: $S = (10^{-1} + 10^{-2} + \ldots + 10^{-10}) + 10^8$.

No primeiro caso, muitos números pequenos são ignorados devido ao arredondamento. No segundo, o somatório dos números pequenos é calculado antes de adicionar o número grande, preservando mais informações significativas.

### 1.3.1 Análise de Instabilidades e Casos Peculiares

Na aritmética de ponto flutuante, certos casos resultam em erros devido à limitação da precisão e à maneira como os números são representados. A seguir, descrevemos dois exemplos clássicos que ilustram essas instabilidades.

#### Imprecisão de operações de Ponto flutuante

Considere o cálculo de $f(x) = x^{10} + 1 - x^{10}$ para $x \in [-60, 60]$. As partes envolvendo $x$ são variáveis e a parte "1"é o literal, ou seja, um valor fixo. Analiticamente, o resultado deveria ser exatamente 1. No entanto, em implementações numéricas, pequenas imprecisões na representação de $x^{10}$ podem levar a resultados instáveis, especialmente para valores de $|x|$ além de um limiar. Isso ocorre devido ao erro relacionado às operações de ponto flutuante, como mostrado na Figura 1.1.

Podemos observar que em um determinado limiar, a função para de se comportar como esperado $f(x) = 1$ e passa a assumir valores os de $f(x) = 2$ e $f(x) = 0$.

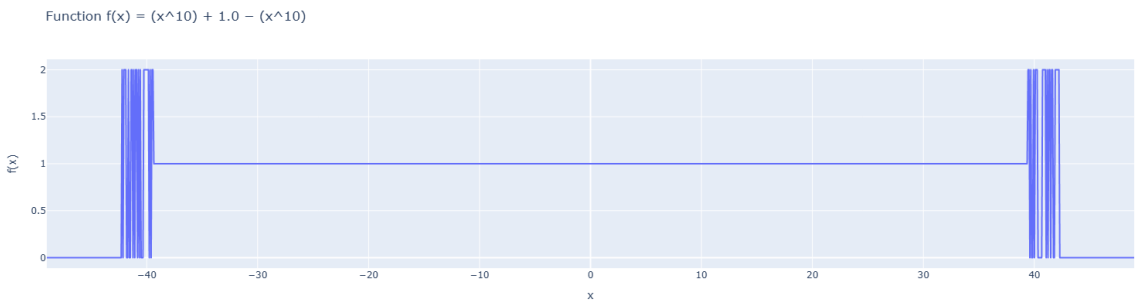


Figura 1.1: Comportamento da expressão $x^{10} + 1 - x^{10}$ no intervalo $[-40, 40]$.

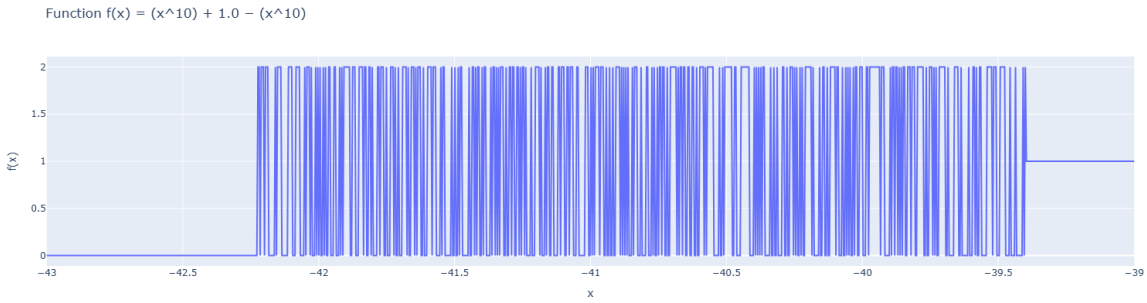


Figura 1.2: Comportamento da expressão $x^{10} + 1 - x^{10}$ no intervalo $[-43, -39]$.

Vamos manipular essa expressão e ver como ela se comporta em diferentes situações.

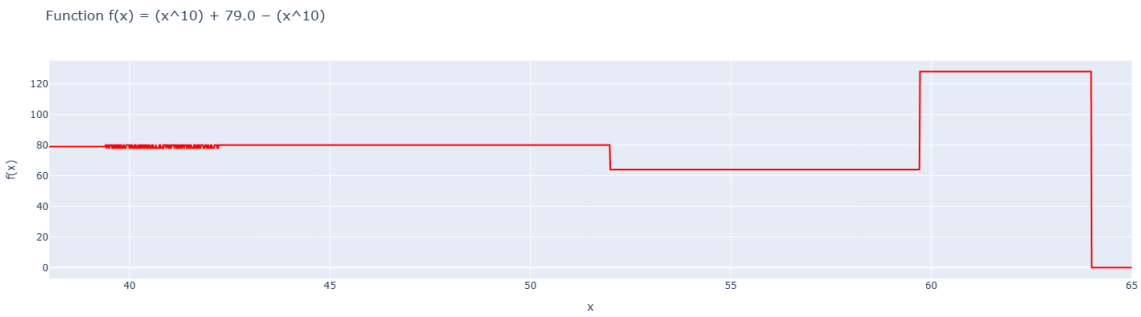


Figura 1.3: Comportamento da expressão $x^{10} + 79 - x^{10}$ no intervalo $[40, 65]$.

Na figura 1.3. é possível observar que a função assume mais de dois valores inesperados para $f(x)$, nesse caso, o conjunto imagem no intervalo $[40, 65]$ é $Im(f) = \{0{,}64{,}78{,}79{,}80{,}128\}$. Em geral, para literais múltiplos de 3, a função assume 3 ou mais valores inesperados.

Explicação: blablabla

Uma dúvida comum é se a precisão desses sistemas afeta nessa expressão. Vamos comparar então um sistema de float32 (Cerca de 7-8 bits dígitos decimais de precisão) a um sistema float64 (Cerca de 15-16 dígitos decimais de precisão)

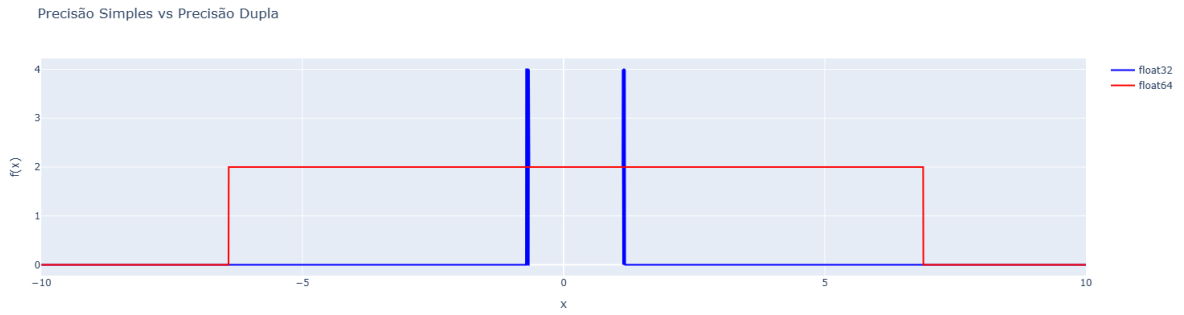


Figura 1.4: Comportamento da expressão $x^{10} + 2 - x^{10}$ no intervalo $[10, 10]$.

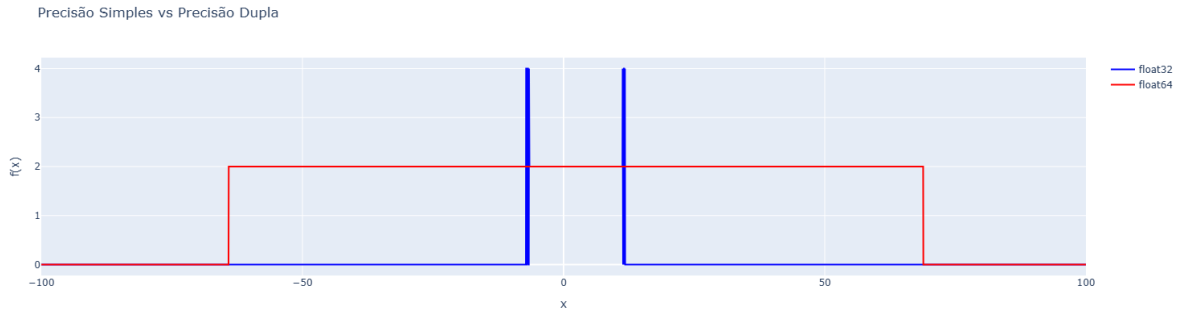


Figura 1.5: Comportamento da expressão $x^{10} + 2 - x^{10}$ no intervalo $[100, 100]$.

Podemos observar em 1.4 e 1.5 que a função mantém o mesmo comportamento em intervalos diferentes. Explicação: ?????

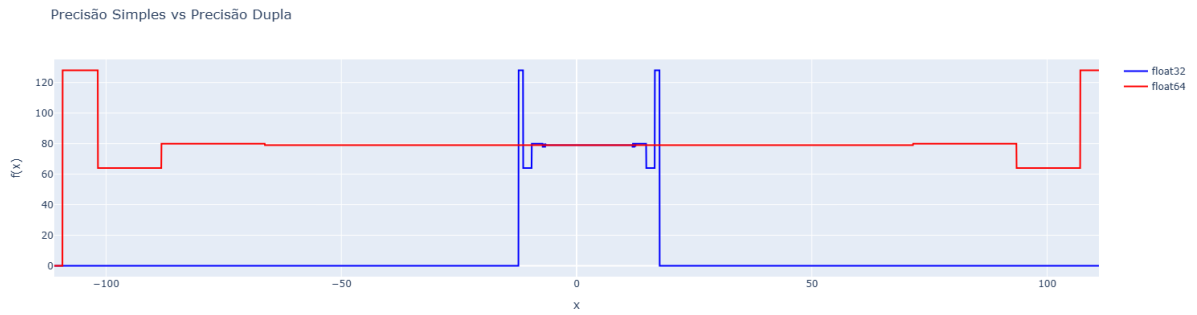


Figura 1.6: Comportamento da expressão $x^{10} + 79 - x^{10}$ no intervalo $[100, 100]$.

Em 1.6 a Função se comporta diferente nos 2 extremos. Explicação: ?????

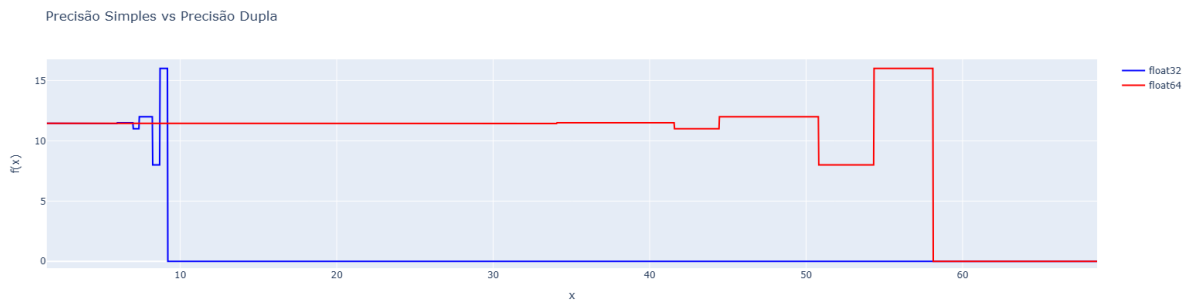


Figura 1.7: Comportamento da expressão $x^{10} + 11.444342323244 - x^{10}$ no intervalo $[0, 60]$.

Em geral, como mostrado nas figuras 1.4, 1.5, 1.6 e 1.7 que a instabilidade ocorre onde o $|x|$ da precisão simples é menor do que na precisão dupla.

Outro caso interessante é quando analisamos a função $f(x) = (x \times x \times x \times x \times x \times x \times x \times x \times x \times x) - x^{10}$. Vejamos o gráfico dessa função.

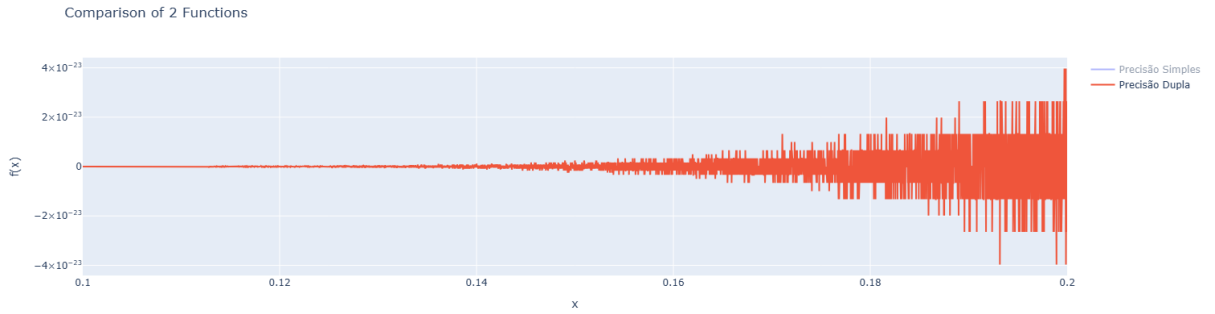


Figura 1.8: Comportamento da expressão $f(x) = (x \times x \times x \times x \times x \times x \times x \times x \times x \times x) - x^{10}$, com precisão dupla, no intervalo $[0{,}1, 0{,}2]$.

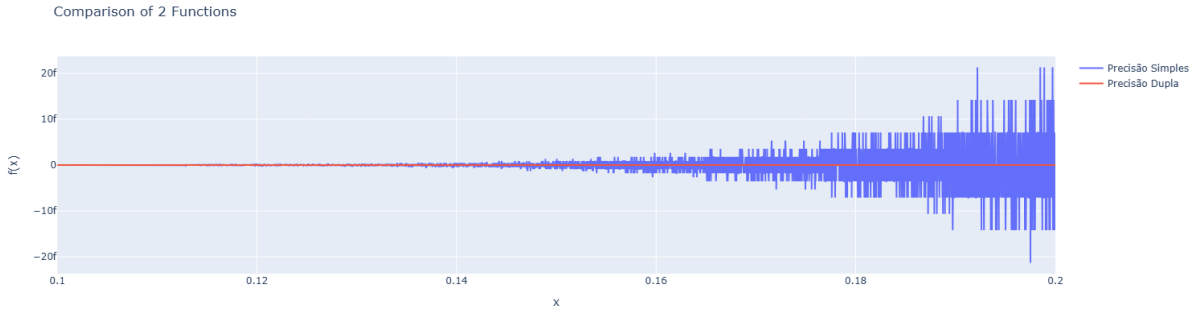


Figura 1.9: Comportamento da expressão $f(x) = (x \times x \times x \times x \times x \times x \times x \times x \times x \times x) - x^{10}$, com precisão simples, no intervalo $[0{,}1, 0{,}2]$.

A instabilidade na precisão simples (Figura 1.9) ocorre em torno de $2 \times 10^{-23}$, enquanto na precisão dupla (Figura 1.8) está em torno de $10^{-15}$ (10 femto). Observa-se,

portanto, que a diferença de instabilidade é extremamente menor na precisão dupla em comparação com a simples.

Uma outra situação é a diferença de como o computador interpreta algumas funções se forem reescritas de maneiras diferentes. Seja $p(x) = (x-1)^6$ e $q(x) = x^6 - 6x^5 + 15x^4 - 20x^3 + 15x^2 - 6x + 1$, analiticamente essas funções são idênticas, porém existem problemas de cancelamento catastrófico na hora de analisarmos ambas em um sistema de ponto flutuante.

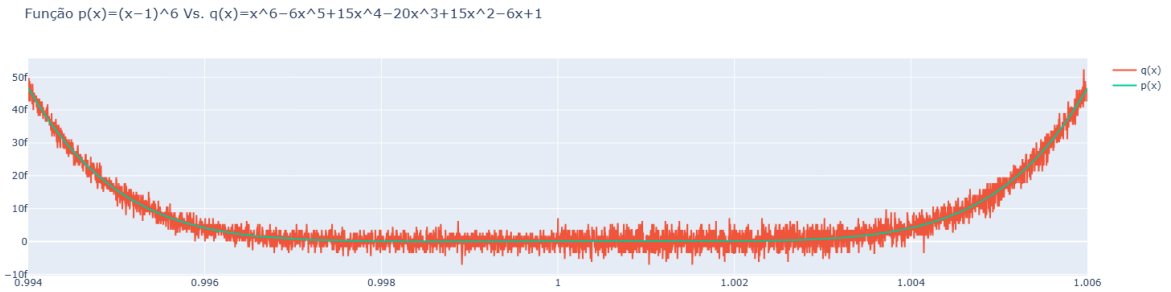


Figura 1.10: Comparação entre as funções $p(x) = (x-1)^6$ e $q(x) = x^6 - 6x^5 + 15x^4 - 20x^3 + 15x^2 - 6x + 1$.

Explicação: yada yada yada

### 1.3.2 Discussão

Esses exemplos destacam a importância da ordem das operações e da análise cuidadosa ao trabalhar com algoritmos numéricos. Técnicas como a reordenação de cálculos e o uso de formatos de precisão estendida podem ajudar a minimizar esses erros em contextos críticos.

# Capítulo 2

# Métodos Iterativos para Zeros de Função

Em muitas aplicações, as soluções buscadas se resumem a encontrar os zeros (ou raízes) de uma função. Entretanto, nem sempre é possível fazê-lo analiticamente, devido a natureza das componentes envolvidas na função, por exemplo, funções polinomiais a partir do $3^{\underline{o}}$ grau, somas de funções trigonométricas e logarítmicas, entre outras. Nesse ínterim, recorremos então a maneiras de obter valores aproximados para tais raízes.

Uma classe de métodos utilizados para aproximar raízes de funções são os **métodos iterativos**. A essência desses métodos está em, partindo de um chute inicial e de uma função apropriada $\varphi$, obter uma sequência $x_k$ onde cada termo é obtido do anterior recursivamente como $x_{k+1} = \varphi(x_k)$. Essa sequência, sob certas hipóteses, converge para a raiz $\xi$ da função.

Ao longo do capítulo, reservaremos o símbolo $\xi$ para representar raízes de funções.

## 2.1 Localização de Raízes

Mais adiante no capítulo, para garantir a convergência da sequência iterativa, é necessário que o primeiro termo esteja suficientemente próximo da raiz e, desse modo, faz-se necessário restringir as funções a intervalos que contenham raízes. Quando as funções envolvidas são contínuas, o resultado a seguir garante a existência de raízes em um intervalo [a,b] desde que as imagens dos extremos tenham sinais opostos.

**Proposição 2.1.1** (Teorema de Bolzano)**.** *Seja $f(x)$ uma função contínua no intervalo $[a, b]$. Se $f(a)f(b) < 0$, então há pelo menos uma raiz $\xi \in (a, b)$.*

Adicionarei um exemplo com gráfico(s).

Outra forma de localizar raízes de uma dada função $f(x)$ é escrevê-la como a diferença entre as funções $g(x) - h(x)$, pois se $f(\xi) = 0$ temos que $g(\xi) - h(\xi) = 0$ ou, equivalentemente, $g(\xi) = h(\xi)$. Graficamente, $\xi$ é a abscissa do ponto de interseção entre as funções g(x) e h(x).

, como $f(x) = x^3 - 9x + 3$, por exemplo, $x^3 = 9x - 3$.

idem

## 2.2 Método do Ponto Fixo

O método do ponto fixo *é um método iterativo* que transforma o problema de buscar as raízes de uma função $f(x)$ no problema de encontrar os pontos fixos de uma outra função $\varphi(x)$, denominada de **função de iteração de ponto fixo**. A partir dessa função de iteração uma sequência é construída recursivamente começando em um valor inicial $x_0$ que convergirá para a raiz $\xi$ de $f(x)$ desde que sejam observadas certas condições sob a função $\varphi(x)$ e o dado inicial $x_0$.

O primeiro passo é gerar funções de iteração $\varphi$ para $f(x)$, o que pode ser feito isolando $x$ na equação $f(x) = 0$. Por exemplo, manipulando a função $x^3 - 9x + 3$ da seguinte forma

$$x^3 - 8x + 3 = x$$

obtemos a função de iteração $\varphi(x) = x^3 - 8x + 3$. Com a mesma lógica, outras possíveis funções de iteração para $f$ são

a) $\varphi_1(x) = \dfrac{x^3}{9} + \dfrac{1}{3}$

b) $\varphi_2(x) = \sqrt[3]{9x - 3}$

c) $\varphi_3(x) = \dfrac{9}{x} - \dfrac{3}{x^2}$

d) $\varphi_4(x) = \sqrt{9 - \dfrac{3}{x}}$

e) $\varphi_5(x) = -\sqrt{9 - \dfrac{3}{x}}$

f) $\varphi_6(x) = x^3 - 8x + 3$

A forma geral da função de iteração é

$$\varphi(x) = x + A(x)f(x) \tag{2.1}$$

com $A(\xi) \neq 0$. Por exemplo, a $\varphi_1 = \frac{x^3}{9} + \frac{1}{3}$ na forma geral ficaria

$$\varphi_1(x) = x + \frac{1}{9}f(x)$$

em que $A(x) = \frac{1}{9}$. Nesse caso, pode-se observar que A$(\xi) \neq 0$.

O resultado a seguir relaciona a raiz de uma função com pontos fixos de uma função de iteração associada a essa função.

**Proposição 2.2.1.** *Seja $\xi$ uma raiz de uma função $f(x)$ e seja $\varphi(x)$ uma função de iteração de associada a $f(x)$. Então, $f(\xi) = 0$ se, e somente se, $\varphi(\xi) = \xi$.*

*Demonstração.* ($\Rightarrow$) Pela forma geral da função de iteração temos que $\varphi(\xi) = \xi + A(\xi)f(\xi)$. Uma vez que $f(\xi) = 0$, então $\varphi(\xi) = \xi$.

($\Leftarrow$) Começando novamente pela forma geral da função de iteração, temos que $\varphi(\xi) = \xi + A(\xi)f(\xi)$. Como $\varphi(\xi) = \xi$, concluímos que $A(\xi)f(\xi) = 0$. Tendo como hipótese que $A(\xi) \neq 0$, então $f(\xi) = 0$. □

Aqui, antes de ir para o resultado principal, dar exemplos de funções de iteração que fazem a sequencia convergir e divergir. Inserir gráficos assim como no livro da Vera. ok!

Sob condições a respeito da função de iteração, sua derivada e o dado inicial, a convergência da sequência iterativa é garantida, como pode-se observar a seguir.

**Teorema 2.2.1.** *Seja $\xi$ uma raiz de f(x), isolada num intervalo I centrado nessa raiz. Considere uma função de iteração $\varphi(x)$ associada a f(x). Sob as seguintes hipóteses:*

*i) $\varphi(x)$ e $\varphi'(x)$ são contínuas em I,*

*ii) $|\varphi'(x)| \leq M < 1$ em I,*

*iii) $x_0 \in I$,*

*a sequência $x_{k+1} = \varphi(x_k)$ converge para a raiz $\xi$.*

*Demonstração.* Como $x_{k+1} = \varphi(x_k)$, subtraindo $\xi$ de ambos os lados da igualdade e usando o fato de que $\varphi(\xi) = \xi$ temos

$$x_{k+1} - \xi = \varphi(x_k) - \varphi(\xi). \tag{2.2}$$

Pelo Teorema do Valor Médio (TVM) podemos escrever

$$\varphi(x_k) - \varphi(\xi) = \varphi'(c_k)(x_k - \xi) \tag{2.3}$$

com $c_k$ entre $x_k$ e $\xi$. Então, substituindo (2.3) em (2.2), temos

$$\begin{aligned} |x_{k+1} - \xi| &= |(x_k - \xi)\ \varphi'(c_k)| \\ &= |x_k - \xi|\ |\varphi'(c_k)| \\ &< |x_k - \xi| \end{aligned} \tag{2.4}$$

uma vez que $|\varphi'(x)| < 1$. Como $x_0 \in I$, podemos concluir que $x_k \in I$ para todo $k$ já que, por (2.4), $|x_k - \xi| < |x_0 - \xi|$.

Na sequência, provaremos que $x_k$ converge para a raiz $\xi$. Vamos começar mostrando que

$$|x_1 - \xi| \leq M\ |x_0 - \xi|. \tag{2.5}$$

Observe que, como $x_1 = \varphi(x_0)$, temos que $x_1 - \xi = \varphi(x_0) - \varphi(\xi)$. Pelo Teorema do Valor Médio temos que $\varphi(x_0) - \varphi(\xi) = (x_0 - \xi)\ \varphi'(c_0)$, para algum $c_0$ entre $x_0$ e $\xi$. Uma vez que $|\varphi'(x)| \leq M$ no intervalo $I$, a seguinte desigualdade é válida

$$\begin{aligned} |x_1 - \xi| &= |x_0 - \xi|\ |\varphi'(c_0)| \\ &\leq M\ |x_0 - \xi| \end{aligned}$$

e provamos a desigualdade (2.5). De modo similar prova-se que $|x_2 - \xi| \leq M\ |x_1 - \xi|$ que, combinado com (2.5), implica que $|x_2 - \xi| \leq M^2\ |x_0 - \xi|$. Repetindo o processo k vezes pode-se concluir que

$$|x_k - \xi| \leq M^k\ |x_0 - \xi|. \tag{2.6}$$

Como $0 < M < 1$, se k tende a infinito, $M^k$ tende a 0 e, portanto, $M^k\ |x_0 - \xi|$ também tende a 0. Assim, provamos que

$$\lim_{k\to\infty} x_k = \xi, \tag{2.7}$$

ou seja, $x_k$ converge para a raiz. □

### 2.2.1 Ordem de convergência

$$\varphi'(x) = \frac{x^2}{3}$$

$|\varphi'(x)| < 1$ para $x \in (-\sqrt{3}, \sqrt{3})$

## 2.3 Método de Newton-Raphson

[? ]