UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO DE CIÊNCIAS EXATAS
CURSO DE GRADUAÇÃO EM MATEMÁTICA

Nome Sobrenome

TÍTULO DA MONOGRAFIA

SEROPÉDICA
2025

Nome Sobrenome

TÍTULO DA MONOGRAFIA

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 “TÍTULO DA MONOGRAFIA”, apresentada e defendida por NOME SOBRENOME, matrícula 2022019000-0, foi aprovada pela Banca Examinadora com conceito “X”, recebendo o número 000.

Seropédica, 11 de junho 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

Capítulo 1

Pontos Flutuantes

Falar sobre sistemas numéricos e bases para representar números

Na matemática, para representar números, são usados os sistemas numéricos. Um sistema numérico é um conjunto de regras e símbolos que definem como escrevemos e entendemos esses números. A base de um sistema numérico é o que indica quantos símbolos diferentes são usados para representar os números e qual o peso de cada posição. Por exemplo, na base 10 (decimal), 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 da representação 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 aritiméticas dos processadores e em algumas linguagens de programação para endereçamento de memória.

Um número em uma base qualquer b pode ser representado na forma polinomial

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

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: $13_{10} = 1 \times 10^1 + 3 \times 10^0$

- Número na base binária: $1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

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

- Número na base decimal:

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

- Número na base binária:

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

$$3{,}5625_{10} = 11{,}1001_2$$

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 eficiente. Essa técnica é utilizada para representar e manipular números reais de forma prática e eficiente. Ela permite a representação de frações, além de números extremamente grandes ou pequenos, que não podem ser armazenados com precisão utilizando apenas inteiros ou ponto fixo.

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

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

em que:

- N é o número real;
- β 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), 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).

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... (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.

1.2 Representação de Números em Ponto Flutuante

Em qualquer máquina, apenas um subconjunto 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 aproximar a representação do valor real do número. Dois dos principais processos empregados para este fim são o truncamento e o arredondamento.

Truncamento

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 $i \geq n$ são descartados.

O erro introduzido por este processo, denominado *erro de truncamento*, é limitado superiormente por

$$|x - T(x)| < b^{-n}$$

Vejamos um exemplo.

Considere o número $x = 3,14156925_{10}$.

Sua expansão é: $x = 3 * 10^0 + 1 * 10^{-1} + 4 * 10^{-2} + 1 * 10^{-3} + 5 * 10^{-4} + 6 * 10^{-5} + 9 * 10^{-6} + 2 * 10^{-7} + 5 * 10^{-8}$.

Se quisermos truncar x com precisão de 5 casas decimais $p = 5$, descartamos todos os termos com potências menores que -5. O truncamento fica: $T(x) = 3 * 10^0 + 1 * 10^{-1} + 4 * 10^{-2} + 1 * 10^{-3} + 5 * 10^{-4} + 6 * 10^{-5}$ Portanto $T(x) = 3,14156$. O erro do truncamento é $E_T = x - T(x) \implies E_T = 3,14156925 - 3,14156 \implies E_T = 0,00000925$. A partir disto podemos verificar que

$$|E_T| < 10^{-5}$$

Arredondamento

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 $\frac{b}{2}$, 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:

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

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.

Menor e Maior valor num sistema de ponto flutuante

Considere uma máquina que opera no sistema:

$$\beta = 10; t = 5; e \in [-5, 5]$$

Os números serão representados da seguinte maneira:

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

O menor valor, em módulo, representado por este sistema é:

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

e o maior:

$$m = 0.99999 \times 10^5 = 99999$$

1.2.1 Casos Possíveis

Considere o conjunto dos números reais $\mathbb{R}$ e um subconjunto G definido como:

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

Onde:

- m é o menor valor positivo representável,
- M é o maior valor representável,
- Os números são representados na forma normalizada $x = \pm 0.d_1 d_2 d_3 \times 10^e$,
- O sistema possui três dígitos na mantissa, base $\beta = 10$, e expoente $e \in [-5, 5]$.

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

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

Exemplo: Seja $x = 237{,}762$. Na forma normalizada: $x = 0{,}237762 \times 10^3$

Com 3 dgitos de preciso:

- **Truncamento:** $x \approx 0{,}237 \times 10^2$
- **Arredondamento:** $x \approx 0{,}238 \times 10^2$

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

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

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

O expoente é –6, menor que o limite inferior $e_{\min} = -5$. Portanto, o número não pode ser representado e ocorre **underflow** (valor é tratado como zero).

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

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

O expoente é +6, maior que $e_{\max} = 5$. Portanto, ocorre **overflow** e o número não pode ser representado corretamente.

Representação do Zero

O valor zero não pode ser normalizado e tem representação especial denotada por

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

Representação especial do zero e prevenção da perda de significância

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

$$N = \pm(d_1{,}d_2 d_3 \ldots d_t)_\beta \times \beta^e,$$

onde $d_1 \neq 0$, 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*:

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

onde 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*, 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 com precisão limitada a 7 dígitos significativos (padrão `float` IEEE 754). Sejam:

$$x = 0{,}000 \times 10^4 \quad \text{(ou seja, o número zero)}, \qquad y = 0{,}276 \times 10^{-2}.$$

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.

Observação sobre perda de significância catastrófica

A perda de significância 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.

Exemplo Prático: Representações

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

Número Real	**Arredondamento**	**Truncamento**
5,678	$0{,}568 \times 10^1$	$0{,}567 \times 10^1$
–192,73	$-0{,}193 \times 10^3$	$-0{,}192 \times 10^3$
3,14159	$0{,}314 \times 10^1$	$0{,}314 \times 10^1$
0,0000063	Underflow (não representável)	
920000,0	Overflow (não representável)	

1.3 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.

1.3.1 Erro Absoluto e Relativo

Em cálculos numéricos, frequentemente lidamos com aproximações devido a arredondamentos e truncamentos. Para avaliar a precisão dessas aproximações, utilizamos as métricas de erro absoluto e erro relativo.

O erro absoluto mede a diferença entre o valor real x_r e o valor aproximado x_a, ou seja, a quantidade exata de erro na aproximação. Ele é definido como:

$$\text{EA} = |x_a - x_r|$$

Quanto menor for o erro absoluto, mais próxima a aproximação está do valor real. No entanto, essa métrica não fornece informações diretas sobre a importância do erro em relação ao tamanho do número em questão.

O erro relativo mede o erro absoluto em relação ao valor real, ou seja, indica proporcionalmente o quão distante a aproximação está. Ele é definido como:

$$\text{ER} = \frac{|x_a - x_r|}{|x_r|}$$

Essa métrica é útil quando lidamos com valores de grandezas muito diferentes. Por exemplo, um erro absoluto de 0.1 pode ser insignificante se estivermos tratando de números na ordem de milhares, mas pode ser relevante se estivermos lidando com valores pequenos, como 0.2.

Exemplo de Cálculo:

Exemplo 1.3.1. Considere que temos um valor real $x_r = 10.5$ e uma aproximação $x_a = 10.3$. Podemos calcular os erros conforme segue:

- **Erro absoluto:**
$$|10.3 - 10.5| = 0.2$$

- **Erro relativo:**
$$\frac{0.2}{10.5} \approx 0.019$$

Neste caso, o erro absoluto é 0.2, o que indica que nossa aproximação difere do valor real por essa quantidade. O erro relativo, sendo aproximadamente 0.019 (ou 1.9%), nos mostra que essa diferença representa menos de 2% do valor real, indicando que a aproximação é razoavelmente precisa.

O erro relativo se torna crucial quando lidamos com números muito grandes ou muito pequenos. Por exemplo:

- Se tivermos um valor real de 10^6 e um erro absoluto de 1, isso representa um erro relativo extremamente pequeno (10^{-6}), indicando que a aproximação é muito boa.

- Se tivermos um valor real de 0.0005 e um erro absoluto de 0.0001, o erro relativo será $\frac{0.0001}{0.0005} = 0.2$, ou seja, 20%, um valor bastante alto, indicando uma aproximação ruim.

Portanto, o erro relativo nos ajuda a interpretar melhor a qualidade de uma aproximação, independentemente da escala dos números envolvidos.

1.4 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.

1.4.1 Instabilidade em Operações Sequenciais

Considere o cálculo de $x^{10} + 1 - x^{10}$ para $x \in [-60, 60]$. 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 extremos de x. Isso ocorre devido ao erro de arredondamento acumulado durante as operações, como mostrado na Figura 2.

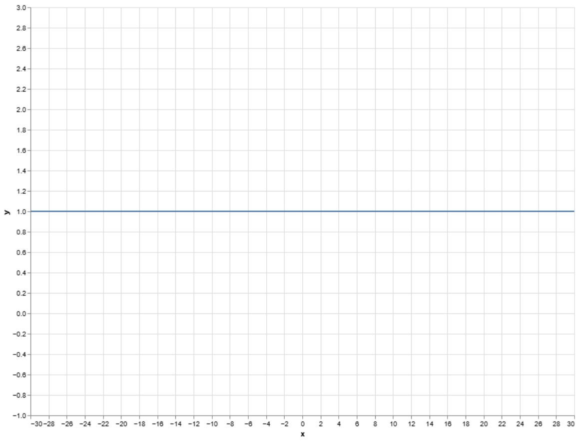

Figura 1.1: Comportamento estável no cálculo de $x^{10} + 1 - x^{10}$ no intervalo $[-30, 30]$.

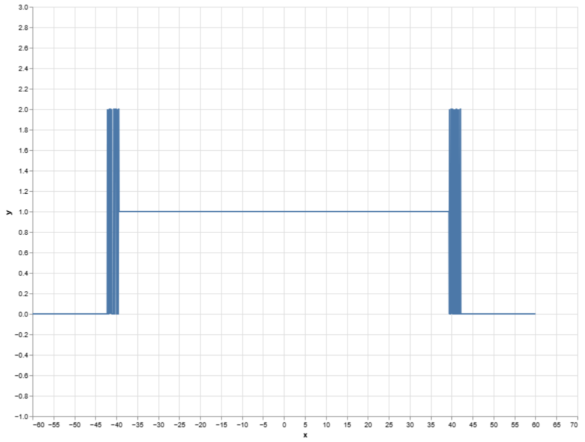

Figura 1.2: Comportamento instável no cálculo de $x^{10} + 1 - x^{10}$ no intervalo $[-60, 60]$.

1.4.2 Perda de Significância em Somas

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.

1.4.3 Discussão

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

Capítulo 2

Métodos Iterativos para Zeros de Função

2.1 Localização de Raízes

Aqui, depois, fazer uma breve introdução sobre métodos para cálculo de raízes de funções, falando dos passos iniciais de determinação de intervalos que contenham raízes e em seguida do processo de refinamento.

2.1.1 f(a)f(b) ¡ 0

2.1.2 g(x) = h(x)

Dada uma função, como $f(x) = x^3 - 9x + 3$, é possível encontrar suas raízes $f(\xi) = 0$ analisando onde duas funções - g(x) e h(x) - construídas a partir delas se interceptam, ou seja, g(x) = h(x), $x^3 = 9x - 3$. Contar uma história do tipo ”nem sempre é possível determinar raízes de funções de forma anlítica e por isso recorremos a métodos numéricos e bla bla bla”.

2.2 Método do Ponto Fixo

Comece aqui a descrever o método do ponto fixo. Diga em linguagem escrita qual é a essência do MPF. Achei interessante que tu que criou o título dessa seção, indicando que vc farejou a linha mental entre o parágrafo anterior e essa parte após esse comentário aqui em cinza

Pode-se sistematizar essa prática construindo-se uma função de iteração na qual se isola x em um lado da igualdade. Usando o mesmo exemplo podemos obter, por exemplo:

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

e) $\varphi_5(x) = x^3 - 8x + 3$

Todas essas funções tem um ponto fixo $\varphi(\xi) = \xi$ na raíz de f, $f(\xi) = 0$, o que significa que as interseções entre $\varphi(x)$ e a função identidade, são raízes ξ da função original f(x). Enunciar de forma mais precisa essa equivalência, não está legal aqui. Agora está legal? Faça isso ser uma proposição. Faça isso depois de definir a fórmula geral de uma função de iteração $\varphi(x) = x + A(x)f(x)$ a proposição vem depois, junto com a fórmula geral, isto por 2 motivos que são, essencialmente, o mesmo: 1. Redundância; 2. Esse fato é uma exposição intuitiva que decorre do que já foi apresentado antes, a formalização sistemática com a prova vem depois como redundância.

Todas essas funções podem ser adequadas a forma geral da função de iteração que é

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

em que $A(\xi) \neq 0$. Podemos adequar, por exemplo, φ_5 da função $f(x) = x^3 - 9x + 3$ a essa forma através da seguinte manipulação algébrica, começando por somar 0 (somando e subtraindo x):

$$\varphi_5(x) = x^3 - 8x + 3$$
$$x - x + \varphi_5(x) = x - x + x^3 - 8x + 3$$
$$0 + \varphi_5(x) = x + x^3 - 9x + 3$$
$$\varphi_5(x) = x + f(x)$$
$$\varphi_5(x) = x + (1)f(x)$$

em que A(x) = 1. Sendo A(x) uma função constante, podemos perceber que ela nunca será nula.

Escreva algo do tipo ”vamos demonstrar o resultado da Proposição XXX”. Use um ambiente do tipo proof ou prova. Qual prova?

Queremos provar que $f(\xi) = 0 \iff \varphi(\xi) = \xi$. Para isso, usaremos a forma geral da função de iteração $\varphi(x) = x + A(x)f(x)$.

Provando primeiramente que a raiz da função é ponto fixo da função de iteração $f(\xi) = 0 \Rightarrow \varphi(\xi) = \xi$

$$
\begin{aligned}
\varphi(\xi) &= \xi + A(\xi)f(\xi),\ f(\xi) = 0 \\
\varphi(\xi) &= \xi + A(\xi) \cdot 0 \\
\varphi(\xi) &= \xi + 0 \\
\varphi(\xi) &= \xi
\end{aligned}
$$

Com isso, concluí-se que a função de iteração calculada na raíz da função f é de fato igual a ξ.

Interessa agora provar que se a função de iteração for calculada no ponto fixo dela, obtemos a raíz da função, $\varphi(\xi) = \xi \Rightarrow f(\xi) = 0$.

$$
\begin{aligned}
\varphi(\xi) &= \xi + A(\xi)f(\xi),\ f(\xi) = 0 \\
\xi &= \xi + A(\xi)f(\xi) \\
0 &= A(\xi)f(\xi),\ A(\xi) \neq 0 \\
f(\xi) &= 0
\end{aligned}
$$

ou seja, o ponto fixo ξ da função de iteração φ é raíz da função original f, q.e.d. (*quod erat demonstrandum*, qual estava-se a demonstrar).

2.3 Prova da convergência

Dada uma função $\varphi(x)$ ela é chamada de função de iteração, pois $\varphi(x_k) = x_{k+1}$ e através dessas iterações podemos buscar o ponto fixo dela na qual $x_k = \xi$. Provaremos a seguir que dada uma raíz ξ de f(x) isolada num intervalo I centrado na mesma, a sequência recursiva $\varphi(x_k) = x_{k+1}$ converge para ξ tendo: a função contínua e derivável, e sua derivada também contínua, no intervalo (a,b); o módulo (M) de sua derivada limitado e inferior a unidade no intervalo ($|\varphi'(x)| \leq M < 1 \ \forall x \in (a, b)$); e o primeiro valor da iteração no intervalo ($x_0 \in I$).

$$\begin{aligned} x_{k+1} &= \varphi(x_k) \\ x_{k+1} - \xi &= \varphi(x_k) - \xi, \text{ como } \xi \text{ é ponto fixo de } \varphi \\ x_{k+1} - \xi &= \varphi(x_k) - \varphi(\xi) \end{aligned} \tag{1}$$

pelo Teorema do Valor Médio (TVM) temos que

$$\frac{\varphi(x_k) - \varphi(\xi)}{x_k - \xi} = \varphi'(c_k) \tag{2.2}$$

então

$$\begin{aligned} x_{k+1} - \xi &= (x_k - \xi)\ \varphi'(c_k) \\ |x_{k+1} - \xi| &= |(x_k - \xi)\ \varphi'(c_k)| \\ |x_{k+1} - \xi| &= |x_k - \xi|\ |\varphi'(c_k)|, \text{ e como } |\varphi'(x)| < 1 \\ |x_k - \xi|\ |\varphi'(c_k)| &< |x_k - \xi| \\ |x_{k+1} - \xi| &< |x_k - \xi| \end{aligned}$$

No limite, $x_k = \xi$, como pode-se verificar pela seguinte sequência de cálculos. Começando por retomar (1) obter-se-á que $|x_1 - \xi| \leq M\ |x_0 - \xi|$

$$|x_1 - \xi| = |\varphi(x_0) - \xi|$$

$$\text{Pelo TVM, } \varphi(x_0) - \xi = (x_0 - \xi)\ \varphi'(c_0), \text{ com } c_0 \in (x_0, \xi)$$

$$|x_1 - \xi| = |x_0 - \xi|\ |\varphi'(c_0)|$$

$$\text{Como } |\varphi'(x)| \leq M,\ |\varphi'(c_0)|\ |x_0 - \xi| \leq M\ |x_0 - \xi|$$

$$|x_1 - \xi| \leq M\ |x_0 - \xi|$$

O próximo passo é obter o semelhante com x_2 e x_1, então generalizar isso para x_k, concluindo que $|x_k - \xi| \leq M^k\ |x_0 - \xi|$

$$|x_2 - \xi| = |\varphi(x_1) - \xi|$$

$$|\varphi(x_1) - \xi| = |x_1 - \xi|\ |\varphi'(c_1)|, \text{ com } c_1 \in (x_1, \xi)$$

$$|x_2 - \xi| = |x_1 - \xi|\ |\varphi'(c_1)|$$

$$|\varphi'(x)| \leq M, \text{ então } |\varphi'(c_1)|\ |x_1 - \xi| \leq M\ |x_1 - \xi|$$

$$|x_2 - \xi| \leq M\ |x_1 - \xi|$$

$$|x_1 - \xi| \leq M\ |x_0 - \xi|,\ |x_2 - \xi| \leq M^2\ |x_0 - \xi|$$

$$\vdots$$

$$|x_k - \xi| = |\varphi(x_{k-1}) - \xi|$$

$$|\varphi(x_{k-1}) - \xi| = |x_{k-1} - \xi|\ |\varphi'(c_{k-1})|, \text{ com } c_{k-1} \in (x_{k-1}, \xi)$$

$$|x_k - \xi| = |x_{k-1} - \xi|\ |\varphi'(c_{k-1})|$$

$$|\varphi'(x)| \leq M, \text{ então } |\varphi'(c_{k-1})|\ |x_{k-1} - \xi| \leq M\ |x_{k-1} - \xi|$$

$$|x_k - \xi| \leq M\ |x_{k-1} - \xi|$$

$$|x_{k-1} - \xi| \leq M\ |x_{k-2} - \xi|,\ |x_k - \xi| \leq M^2\ |x_{k-2} - \xi|$$

$$|x_k - \xi| \leq\ ...\ \leq M^k\ |x_0 - \xi|,\ |x_k - \xi| \leq M^k\ |x_0 - \xi|$$

Enfim aplica-se o limite para obter que x_k converge para a raiz ξ de f, (tendo $0 < M < 1$)

$$\lim_{k\to\infty} |x_k - \xi| \leq M^k \; |x_0 - \xi|$$

$$\lim_{k\to\infty} |x_k - \xi| \leq 0 \; |x_0 - \xi|$$

$$\lim_{k\to\infty} |x_k - \xi| = 0$$

logo,

$$\lim_{k\to\infty} x_k = \xi \tag{2.3}$$

2.4 Método de Newton-Raphson

Referências Bibliográficas

[1] RUGGIERO, M. A. G., DA ROCHA LOPES, V. L., 1998, *Cálculo numérico: aspectos teóricos e computacionais.* Sao Paulo, Pearson/Makron.

[1]