MAP214 - Cálculo Numérico
Notas de Aula

Prof. Dr. Arnaldo Gammal

Compilado em 2019 por Caíke Crepaldi
(última atualização: 14 de agosto de 2020)

Sumário

Apresentação

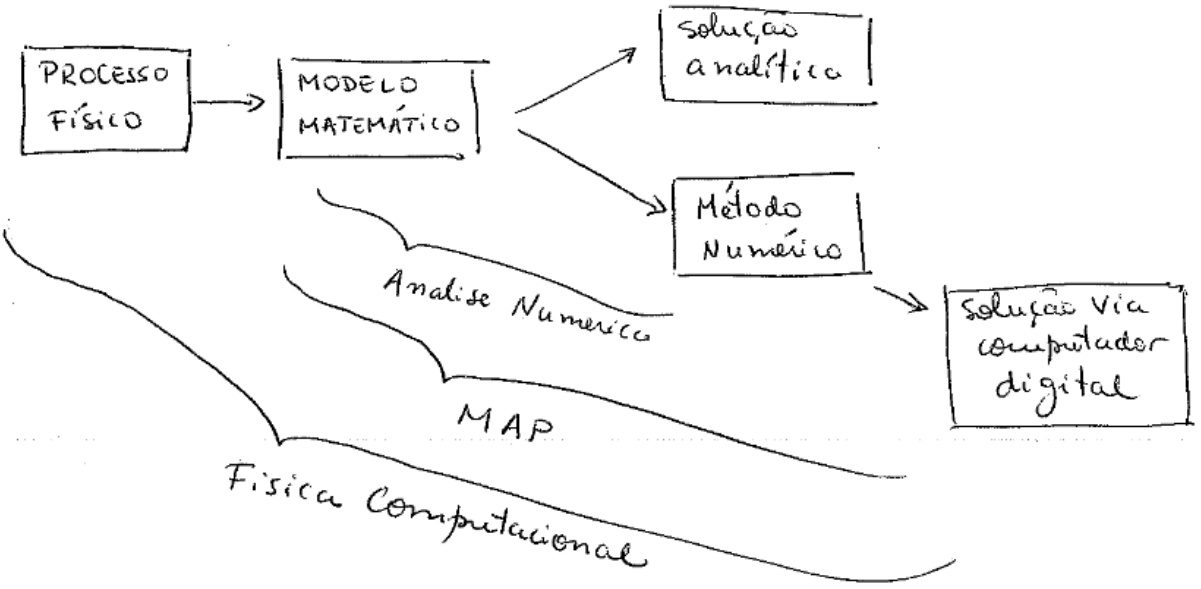
Esta apostila compila as notas de aula ministradas no Instituto de Física da Universidade de São Paulo, para a disciplina **MAP0214 - Cálculo Numérico**, oferecida para alunos do curso de bacharelado em Física. O curso foi apresentado pela primeira vez no IFUSP em 2008 e desde então vem tendo novos tópicos adicionados.

São Paulo, 2o. semestre de 2020.

Arnaldo Gammal

Introdução

Cálculo Numérico trata de métodos numéricos para a solução de problemas matemáticos.



Exemplos

- Processo Físico: Colisão de galáxias;
- Modelo Matemático: Equações diferenciais da Gravitação
- Método Numérico: Solução numérica das equações diferenciais
- Programação do método numérico

Método Numérico

- Utiliza apenas as operações aritméticas: $+, -, \times, \div$
- Permitem a solução de modelos matemáticos que não admitem solução analítica

Capítulo 1

Erros, Precisão Numérica e Ponto Flutuante

1.1 Onde aparecem os erros

1.1.1 Erro Inerente

São os erros na criação de um modelo matemático. Consideremos o caso da emissão de luz de uma lâmpada fluorescente (plasma):

- Conhecimento incompleto dos fenômenos: Excitações podem gerar moléculas incomuns.
- Natureza estatística: Colisões entre partículas do gás não são determinísticas.
- Incerteza nos parâmetros experimentais.
- Eliminação de efeitos de ordem superior.

1.1.2 Erros ao adotarmos um método numérico

- Truncamento de uma série, como é o caso da série usada para o cálculo de π:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} \ldots \quad \text{Gottfried Leibniz (1682)}$$

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{n=0}^{\infty} \frac{(4n)!(1103 + 26390n)}{(n!)^4 (396)^{4n}} \quad \text{Srinivasa Ramanujan } (\sim 1910)$$

- Aproximação de áreas por retângulos.

1.1.3 Erros computacionais

- Números reais são representados por um sistema de **ponto flutuante**.
- $\underbrace{\text{Truncamento} + \text{Arredondamento}}_{\text{ROUNDOFF}}$.

1.2 Erro absoluto e erro relativo

– Erro Absoluto, ε_x:

$$\varepsilon_x \equiv |x - \langle x \rangle|,$$

onde $\langle x \rangle$ é o valor verdadeiro de x.

– Erro relativo:

$$\varepsilon_r \equiv \frac{\varepsilon_x}{\langle x \rangle} = \frac{|x - \langle x \rangle|}{\langle x \rangle},$$

– Às vezes o erro absoluto é pequeno, mas o relativo é grande, *e.g.*[1]

$$\begin{aligned} \langle x \rangle &= 0.0006 \\ x &= 0.0005 \\ \varepsilon_x &= 0.0001 \\ \varepsilon_r &= 0.2 = 20\% \end{aligned}$$

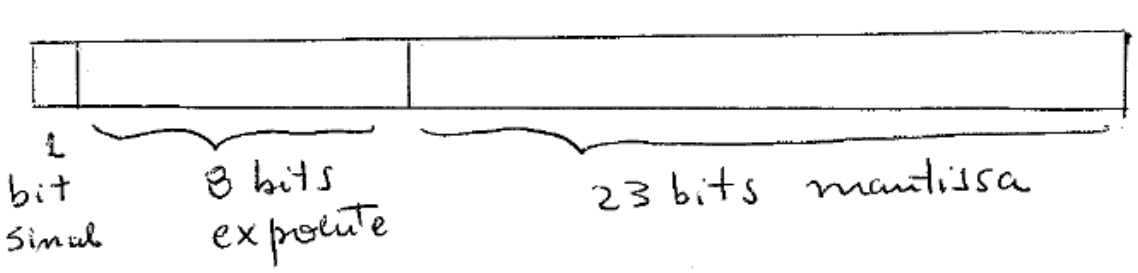
1.3 Definição de sistema de números de ponto flutuante

Um número real em sua representação de ponto flutuante (P.F.) é dado por

$$\pm \underbrace{.d_1 d_2 d_3 \ldots d_n}_{\text{mantissa}} \times \beta^e \ ,$$

onde β é a base, n é a precisão (ou seja, número de algarismos significativos na base β), e e é o expoente (com $L \leqslant e \leqslant U$).

Essa representação normalmente acontece com 32 bits.



Linguagens de programação de baixo nível normalmente trabalham com 2 níveis de precisão pré-definidos:

– Single Precision (precisão simples): representação em 32 bits ($\approx$ 7 dígitos decimais).

– Double Precision (precisão dupla): representação em 64 bits ($\approx$ 16 dígitos decimais).

Precisão	FORTRAN	C, C++	Sinal	Expoente	Valor do Exp.	Mantissa	Total	Casas decimais
Single	REAL	float	1 bit	8 bits	$(-126, +127)$	23 bits	32 bits	7
Double	REAL*8	double	1 bit	11 bits	$(-1022, +1023)$	52 bits	64 bits	16

[1]Representaremos os números separando a parte inteira da fracionária por ponto, para comparação direta com os resultados impressos pelos computador.

Dica

Entre na Wikipedia e procure por "IEEE 754-2019" ou "IEEE floating point arithmetic".

Valores mais distantes do zero:

- Single Precision: $\pm(1-2^{24}) \times 2^{+128} \approx \pm 3.40 \times 10^{+38}$.
- Double Precision: $\pm(1-(1/2)^{53}) \times 2^{+1024} \approx \pm 1.79 \times 10^{+308}$.

Valores mais próximos do zero:

- Single Precision: $\pm(1-2^{24}) \times 2^{-126} \approx \pm 1.17 \times 10^{-38}$.
- Double Precision: $\pm(1-(1/2)^{53}) \times 2^{-1022} \approx \pm 2.22 \times 10^{-308}$.

Exercício 1.1

Construa um programa que calcule n! de n=1 até o maior valor que possa ser representado. Faça isso em precisão simples e dupla.

Algoritmo 1.1: Fatorial.

```
1  begin programa
2    fator ← 1
3    do n=1 até 200
4      fator = n*fator
5      print n, fator
6    end do
7  end programa
```

Exercício 1.2

Imprimir com todos os decimais:

$$1/3 \ , \ \text{acos}(-1)$$

em precisão simples e dupla.

1.4 Operações básicas em P.F.

Sejam as operações aritméticas básicas em ponto flutuante denotadas como $\ominus$, $\oplus$, $\otimes$, $\oslash$. Em geral, temos que:

$$x \oplus y \neq x + y$$

$$x \otimes y \neq x \times y$$

Associatividade em geral não é válida, *e.g.*, faça em precisão simples (1/2+1/3)+1/9 e compare com 1/2+(1/3+1/9). A propriedade distributiva também não é, *i.e.*, $x \otimes (y \oplus z) \neq (x \otimes y) \oplus (x \otimes z)$.

Exercícios

1.1 Mostre que 0.1 na base 10 é representado por 0.0001100110011... na base 2.

1.2 Some 10000 vezes no computador o valor 0.0001 e imprima o resultado com sete decimais (precisão simples).

1.3 Calcule no computador o maior $n!$ até causar overflow (precisão simples e dupla). Imprima n e $n!$.

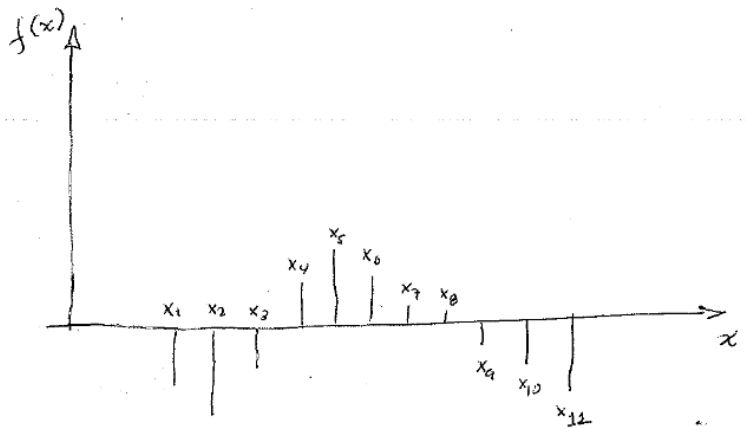
Capítulo 2

Zeros de Funções

2.1 Algoritmos

O problema de encontrar raízes de funções pode ser dividido em duas etapas

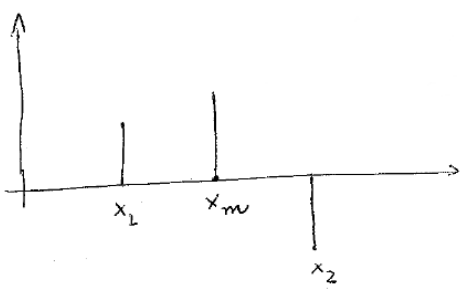
1. Determinar intervalos (ou regiões) onde exista uma única solução para $f(x) = 0$.
2. Sabendo os intervalos, determinar a solução ali contida por meio de um algoritmo.



Vamos abordar quatro métodos: o método de bisseção, o método de substituições sucessivas, o método de Newton-Raphson (ou apenas de Newton), e, por fim, o método das secantes.

2.1.1 Método de bisseção

Também conhecido em português como método da dicotomia, ou em inglês por *bisection* ou *half-interval method*.



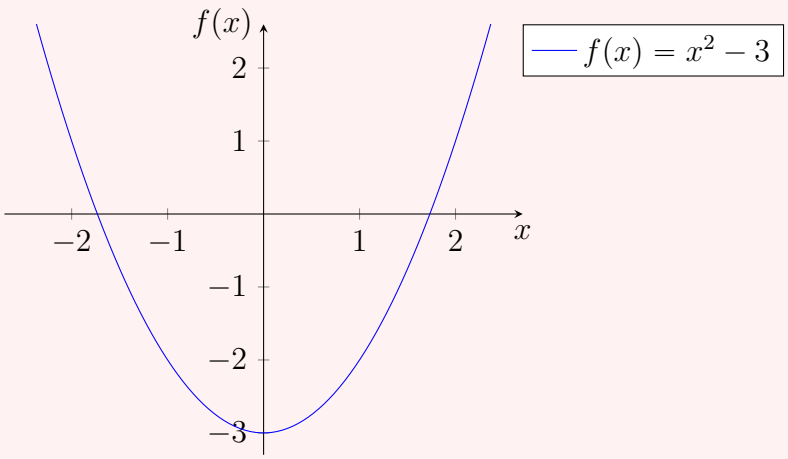
Algoritmo 2.1: Bisseção.

```
1  begin program
2      x1 ← a
3      x2 ← b
4      while |x2 − x1| > ε
5          xm = (x1 + x2)/2
6          if (sinal de f(xm) = sinal de f(x1))
7              x1 ← xm
8          else
9              x2 ← xm
10         end if
11     end while
12 end program
```

Exemplo

Encontre a solução de $x^2 - 3 = 0$ pelo método da bisseção.

Solução:



x_1	x_2	x_m	$f(x_1)$	$f(x_m)$	$e_n = \lvert x_2 - x_1 \rvert$
1	2	1.5	-3	-0.75	1
1.5	2	1.75	-0.75	0.0625	0.5
1.5	1.75	1.625	-0.75	-0.359	0.25
1.625	1.75	1.6875	-0.359	-0.152	0.125
1.6875	1.75	1.71875	-0.152	-0.0459	0.0625
1.71875	1.75	1.734375	-0.0459	$+8.05\times10^{-3}$	0.03125
1.71875	1.734375	1.7265625	-0.0459	-0.01898	2^{-6}
1.71875	1.734375	1.73046879	-0.01898	-5.47×10^{-3}	2^{-7}
1.73046879	1.734375	1.732421875	-5.47×10^{-3}	1.285×10^{-3}	2^{-8}

Note que

$$e_{n+1} = \frac{1}{2} e_n$$

Exercício 2.1

Refazer com a calculadora o exemplo anterior mas para $f(x) = x^2 - 5 = 0$.

De uma forma geral, um método é dito de convergência de ordem m se na vizinhança da solução temos

$$|e_{n+1}| = cte\,|e_n|^m \quad \text{ou} \quad \lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^m} = cte$$

No caso do método da bisseção, temos $cte = 1/2$ e $m = 1$, portanto ele é dito de **convergência linear**.

2.1.2 Método de substituições sucessivas

Também conhecido como *Método do Ponto Fixo*. Escrevemos a equação $f(x) = 0$ na forma

$$x = G(x) \Rightarrow \underbrace{x_{n+1} = G(x_n)}_{\text{função iterativa}}$$

Exemplo

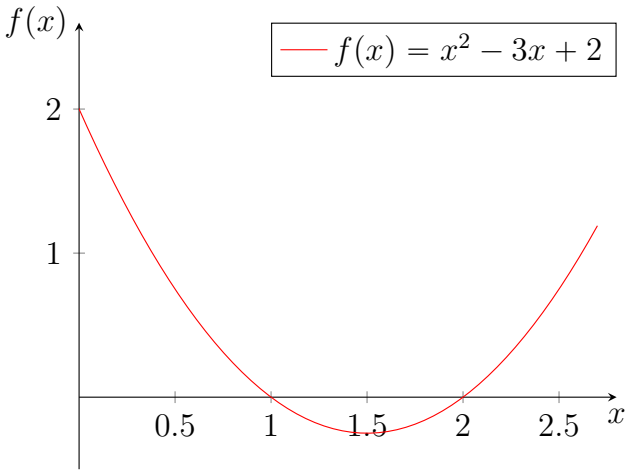
$x^2 - 3x + 2 = 0$ pode ser escrita na forma

$$x = (x^2 + 2)/3$$

$$x_{n+1} = (x_n^2 + 2)/3, \quad (n = 0, 1, 2, \dots)$$

Neste caso, tomando $x_0 = 0$, temos:

$x_1 = 0.666666667$
$x_2 = 0.887974394$
$\vdots$
$x_9 = 0.99151400 \quad \rightarrow \quad$ converge para $a = 1$



Estudo da convergência

$$e_n = x_n - a \tag{2.1}$$

$$G(x_n) - G(a) = G(a + e_n) - G(a) \tag{2.2}$$

Pela expansão em série de Taylor, temos que

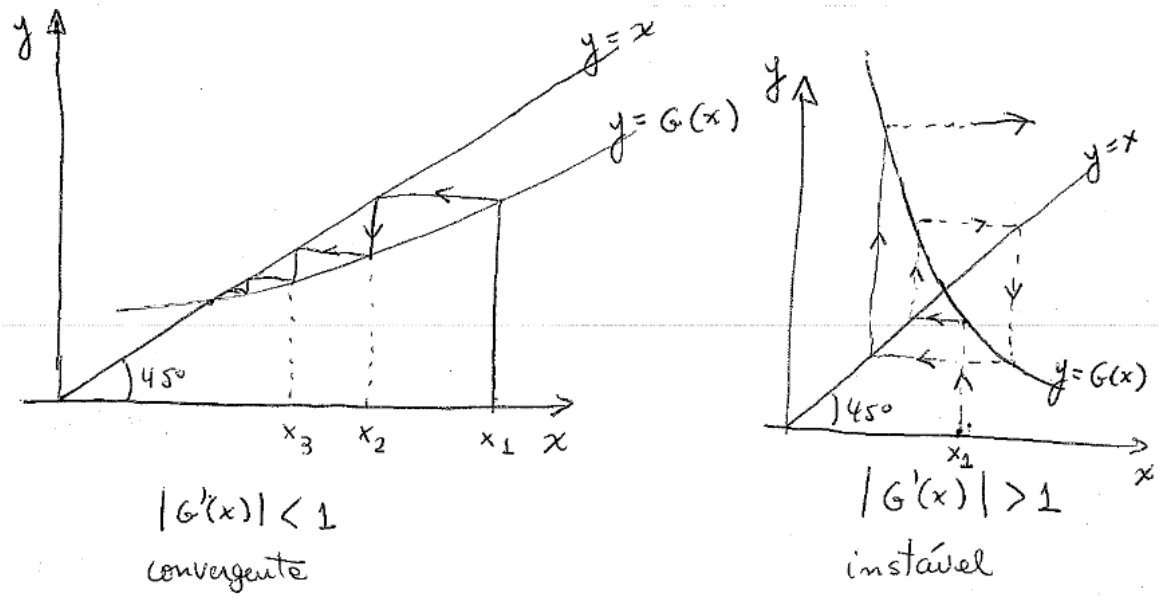
$$G(a + e_n) = G(a) + G'(\xi)e_n, \qquad a < \xi < a + e_n \tag{2.3}$$

Substituindo (2.3) em (2.2), temos

$$e_{n+1} = G'(\xi)e_n \ . \tag{2.4}$$

Se existe uma constante C tal que $|G'(x) \leqslant C|$ perto de $x = a$, então $|e_{n+1}| \leqslant C\ |e_n| \leqslant C^2\ |e_{n-1}| \leqslant \cdots \leqslant C^{n+1}\ |e_0|$ e e_{n+1} tende a zero. Como há uma relação linear entre e_n e e_{n+1}, a iteração converge linearmente para a raiz se $|G'(x)| < 1$ em algum intervalo contendo a solução e a estimativa inicial x_0.

Interpretação Geométrica

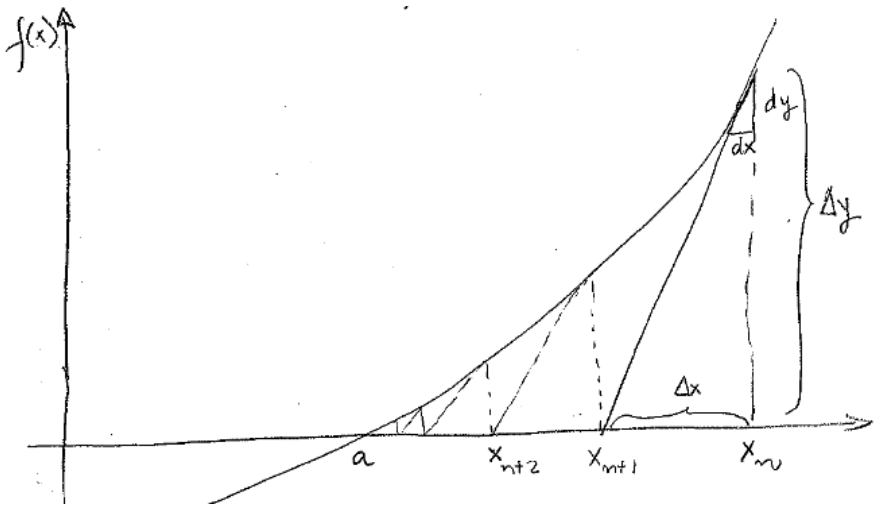


Nota

Funções iterativas podem levar a resultados não-convergentes. Em alguns casos, esses resultados são caóticos, como no clássico exemplo da equação logística:

$$x_{n+1} = r\ x_n(1 - x_n)$$

2.1.3 Método de Newton-Raphson



$$f'(x_n) = \frac{dy}{dx} = \frac{\Delta y}{\Delta x} = \frac{f(x_n)}{x_n - x_{n+1}} \Rightarrow x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2.5}$$

Podemos escrever na forma $x_{n+1} = G(x_n)$ onde

$$G(x) = x - \frac{f(x)}{f'(x)}. \tag{2.6}$$

De forma geral, podemos escrever o erro como

$$e_{n+1} = x_{n+1} - a = G(x_n) - G(a) = G(a + e_n) - G(a). \tag{2.7}$$

Expandindo em série de Taylor:

$$G(a + e_n) = G(a) + e_n\, G'(a) + e_n^2\, \frac{G''(\xi)}{2}, \qquad (a < \xi < a + e_n). \tag{2.8}$$

Derivando $G(x)$ em relação a x no ponto a

$$G'(a) = 1 - \left[\frac{f'(a)}{f'(a)} - \frac{f(a)f''(a)}{f'(a)f'(a)}\right] = 0 \tag{2.9}$$

e verificando que $G''(x)$ próximo a a vale aproximadamente

$$G''(a) = \frac{f''(a)}{f'(a)}\ . \tag{2.10}$$

Substituindo as equações (2.8), (2.9), e (2.10) em (2.7), temos, na vizinhança de a, que

$$e_{n+1} \approx \frac{f''(a)}{2f'(a)} e_n^2 \ \rightarrow \ \begin{array}{c}\text{o método converge na} \\ \text{vizinhança de } a\end{array} \tag{2.11}$$

Para a convergência, devemos ter

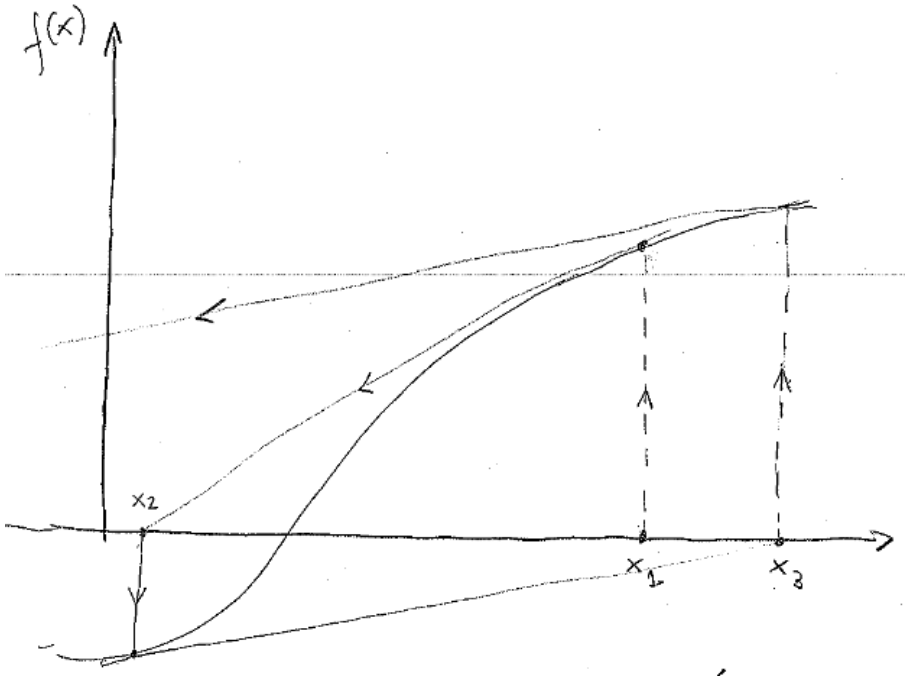
$$e_1 = \underbrace{\frac{f''(x)}{2f'(x)} e_0}_{<1 \text{ em módulo}}\ e_0$$

$$\therefore\ |e_0 f''(x)/f'(x)| < 2$$

Essa é a condição inicial de convergência $\forall x$ perto de a.

Teorema. *Se $f(a) = 0$, $f'(a) \neq 0$ e $f''(x)$ é contínua, então há um intervalo aberto I contendo a tal que se $x_1 \in I$ no método de N-R, então $x_n \to a$ quando $n \to \infty$.*

Com esse teorema em mente, observemos a seguinte figura:



Neste caso, a escolha de x_1 fez com que o método não convergisse ($x_1 \notin I$). Então, na prática, como podemos determinar x_1?

Dica

Faz-se previamente um gráfico de $f(x)$ ou usa-se um método seguro (e.g. bisseção) para encontrar x_1 próximo de a, e em seguida aplica-se N-R para chegar a solução final.

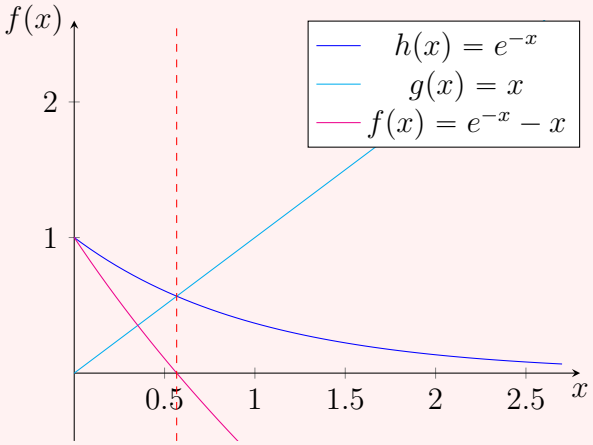
Exemplo

Encontre a solução de $e^{-x} = x$ usando o método de Newton-Raphson.

Solução:

$$f(x) = e^{-x} - x$$
$$f'(x) = -e^{-x} - 1$$
$$f'(x) = e^{-x}$$
$$|f(x)/f'(x)| < 1 \forall x$$

Desta forma, desde que $e_0 < 2$, devemos ter convergência quadrática.



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

Tomando $x_0 = 0.5$ temos

n	x_n	$f(x_n)$	$f'(x_n)$
0	0.5	0.106530659	-1.60653066
1	0.566311002	$1.304511685 \times 10^{-3}$	-1.5676155
2	0.567143165	1.96511×10^{-7}	-1.567413362
3	0.56714329	4.2×10^{-10}	

Esses resultados foram obtidos na calculadora. Eles devem ser ligeiramente diferentes no computador.

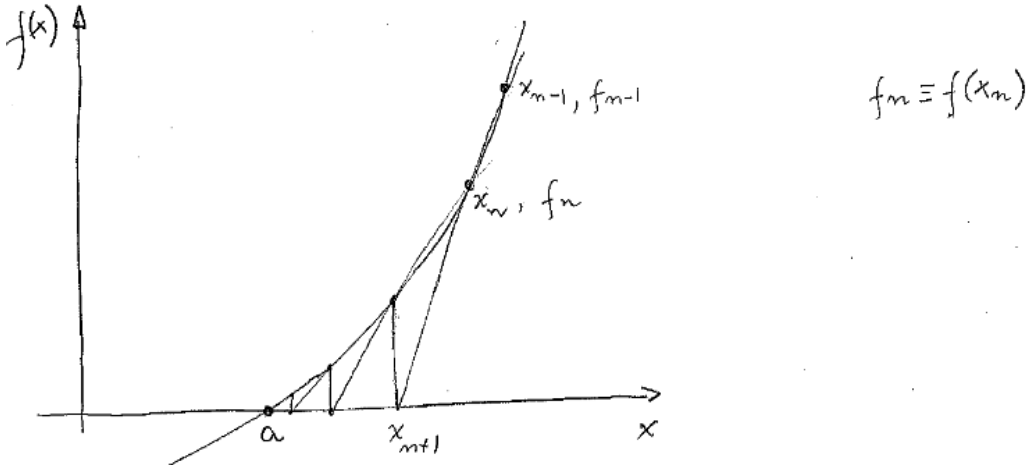
Dica

Um critério para parar o programa é

$$\underbrace{\left|\frac{x_{n+1}-x_n}{x_n}\right|}_{\text{erro relativo}} < \varepsilon,$$

onde ε é especificado.

2.1.4 Método das secantes



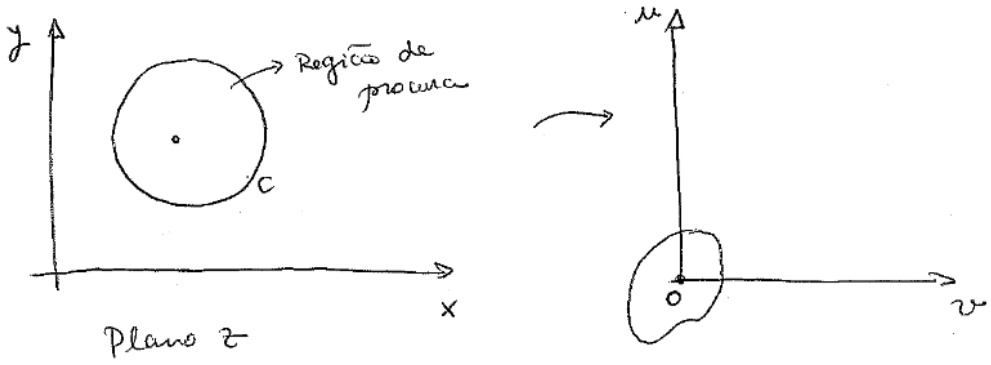
O método das secantes pode ser obtido a partir do método de N-R, substituindo $f'(x_n)$ por $\dfrac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}}$, dessa maneira temos

$$x_{n+1} = x_n - \frac{f(x_n)(x_n-x_{n-1})}{f(x_n)-f(x_{n-1})}\ . \tag{2.12}$$

Teorema. *Se $f(a)=0,\ f'(a)\neq 0,\ f''(a)\neq 0$ e f'' contínua, então:*

1. *Existe um intervalo aberto I contendo a tal que, se x_0 e x_1 são distintos em I, a sequência x_n do método das secantes converge para a raiz a quando $n\to\infty$.*
2. $\lim_{n\to\infty}\left|\dfrac{e_{n+1}}{e_n^m}\right| = cte \neq 0$, *onde* $m=\dfrac{1+\sqrt{5}}{2}\approx 1.618 \Rightarrow$ *entre linear e quadrático.*

2.2 Raízes complexas



Se ao variarmos z sobre C, $f(z)$ circundará a origem ao menos uma vez, podemos dizer que dentro de C existe uma raiz complexa. Métodos de N-R e secante podem ser aplicados respectivamente sem alterações nas formas

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)} \quad \text{e} \quad z_{n+1} = z_n - \frac{f(z_n)(z_n - z_{n-1})}{f(z_n) - f(z_{n-1})} \ . \tag{2.13}$$

São necessários $z_0,\ z_1$ próximos à raiz e podemos ter convergência, divergência ou circulação em torno da raiz.

Se uma raiz a for encontrada podemos encontrar outras raízes a partir da função

$$f_1(z) = \frac{f(z)}{z - a},$$

que é chamado de deflação. Raízes de polinômios também pode ser obtidas por N-R e secantes. Para cálculo de raízes, pode-se alternativamente usar o método de determinação de autovalores de um polinômio característico, ver Press *et al.* (1992). Alguns polinômios são patológicos e pequenas mudanças nos coeficientes podem no pior caso fazer com que raízes reais vão para o plano complexo como no caso Wilkinson, onde se busca encontrar as raízes do polinômio

$$(x-1)(x-2)...(x-19)(x-20) \ ,$$

ver detalhes Wilkinson (1963) e Acton (1970).

Dica

Pesquise na internet sobre Julia sets.

2.3 Sistemas de equações não-lineares

Seja o sistema de n equações não lineares com n incógnitas

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ f_2(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned} \tag{2.14}$$

que podem ser escritas na forma compacta $F(\mathbf{x}) = 0$ onde $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ e $F = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})\}$. Uma possível solução pode ser obtida generalizando o método de Newton. Para tanto vamos rederivar o método de Newton usando uma expansão em serie de Taylor torno de do valor inicial tomado $\mathbf{x}_0$

$$\mathrm{F}(\mathbf{x}) = \mathrm{F}(\mathbf{x}_0) + \mathrm{J}(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) + \ldots \tag{2.15}$$

onde J é a matriz Jacobiana, $\mathrm{J} \equiv \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \ldots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \ldots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \ldots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$. A aproximação seguinte $\mathbf{x}_1$ para a solução de $\mathrm{F}(\mathbf{x}) = 0$ é dada por $\mathrm{F}(\mathbf{x}_0) + \mathrm{J}_0(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) = 0$, $\mathrm{J}_0 \equiv \mathrm{J}(\mathbf{x}_0)$, ou

$$\mathrm{J}_0 \Delta\mathbf{x}_0 = -F(\mathbf{x}_0), \text{onde} \quad \Delta\mathbf{x}_0 \equiv \mathbf{x}_1 - \mathbf{x}_0 \tag{2.16}$$

que é um sistema linear que pode ser resolvido por exemplo pelo método eliminação de Gauss visto em detalhe no capítulo 3. Obtendo Δ_{x0}, $\mathbf{x}_1$ é calculado com $\mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}_0$ e assim para $\mathbf{x}_{k+1}$ temos

$$\mathrm{J}_k \Delta\mathbf{x}_k = -\mathrm{F}(\mathbf{x}_k) \quad \text{e} \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}_k \tag{2.17}$$

Note que no caso particular de uma única equação e uma única incógnita ($n = 1$), a equação (2.17) se reduz à (2.5).

Exemplo: Seja o sistema não linear dado por

$$\begin{cases} x_1^2 + x_2^2 + x_3^2 - 1 &= 0 \\ 2x_1^2 + x_2^2 - 4x_3 &= 0 \\ 3x_1^2 - 4x_2 + x_3^2 &= 0 \end{cases} \tag{2.18}$$

cuja solução com precisão até 4 algarismos significativos é $\mathbf{x} = \begin{bmatrix} 0.7852 \\ 0.4966 \\ 0.3699 \end{bmatrix}$. O jacobiano será então $\mathrm{J} = \begin{bmatrix} 2x_1 & 2x_2 & 2x_3 \\ 4x_1 & 2x_2 & -4 \\ 6x_1 & -4 & 2x_3 \end{bmatrix}$.

Iniciando com $\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$, fornece imediatamente $\mathrm{J}_0 = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -4 \\ 3 & -4 & 1 \end{bmatrix}$, $-\mathrm{F}(\mathbf{x}_0) = \begin{bmatrix} 0.25 \\ 1.25 \\ 1.00 \end{bmatrix}$.

Resolvendo o sistema $\mathrm{J}_0\Delta\mathbf{x}_0 = -\mathrm{F}(\mathbf{x}_0)$ obtemos $\Delta\mathbf{x}_0 = \begin{bmatrix} 0.375 \\ 0 \\ -0.125 \end{bmatrix}$

e $\mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 0.375 \\ 0 \\ -0.125 \end{bmatrix} = \begin{bmatrix} 0.875 \\ 0.500 \\ 0.375 \end{bmatrix}$, e assim sucessivamente.

2.4 Método de Broyden

O método de Newton para resolver o sistema de equações não lineares $\mathrm{F}(\mathbf{x}) = 0$ usa o jacobiano a cada iteração. mas o cálculo do jacobiano pode ser muito complicado e caro $\mathcal{O}n^3$.

O método de Broyden é uma generalização do método de secantes para múltiplas equações. No método de secantes a derivada é estimada como $f'(x_k) \approx \frac{f(x_k) - f(x_{k_1})}{x_k - x_{k-1}}$. O jacobiano poderia

ser estimado usando

$$\mathrm{J}_k(\mathbf{x}_k - \mathbf{x}_{k-1}) \simeq \mathrm{F}(\mathbf{x}_k) - \mathrm{F}(\mathbf{x}_{k-1}). \tag{2.19}$$

Mas essa equação não determina unicamente J_k e são necessárias restrições. Broyden sugeriu usar uma estimativa do jacobiano J_{k-1} e estimar J_k como solução da equação (2.19) que produza a mínima modificação de J_{k-1}, mínima no sentido da norma de Frobenius $\|\mathrm{J}_k - \mathrm{J}_{k-1}\|_F$, definida por $\|\mathrm{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2}$. Assim ele obteve a expressão

$$\mathrm{J}_k = \mathrm{J}_{k-1} + \frac{\Delta\mathrm{F}_k - \mathrm{J}_{k-1}\Delta\mathbf{x}_k}{\|\Delta\mathbf{x}_k\|^2}\Delta\mathbf{x}_k^T \ , \tag{2.20}$$

onde $\Delta\mathrm{F}_k \equiv \mathrm{F}(x_{k+1}) - \mathrm{F}(\mathbf{x}_k)$, e então prossegue-se a iteração com método de Newton

$$\mathrm{J}_k\Delta\mathbf{x}_k = -\mathrm{F}(\mathbf{x}_k) \quad \text{ou alternativamnte} \quad \mathbf{x}_{k+1} = \mathbf{x}_k - \mathrm{J}_k^{-1}\mathrm{F}(\mathbf{x}_k) \tag{2.21}$$

Broyden também sugeriu usar a fórmula de Sherman-Morrison[1] para atualizar o jacobiano invertido

$$\mathrm{J}_k^{-1} = \mathrm{J}_{k-1}^{-1} + \frac{\Delta\mathbf{x}_k - \mathrm{J}_{k-1}^{-1}\Delta\mathrm{F}_k}{\Delta\mathbf{x}_k^T\mathrm{J}_{k-1}^{-1}\Delta\mathrm{F}_k}\Delta\mathbf{x}_k^T\mathrm{J}_{k-1}^{-1} \tag{2.22}$$

Este método para estimativa de J é chamado de *good Broyden method*. Alternativamente pode-se usar uma atualização diferente para J_k^{-1} que minimize $\left\|\mathrm{J}_k^{-1} - \mathrm{J}_{k-1}^{-1}\right\|_F$ obtendo

$$\mathrm{J}_k^{-1} = \mathrm{J}_{k-1}^{-1} + \frac{\Delta\mathbf{x}_k - \mathrm{J}_{k-1}^{-1}\Delta\mathrm{F}_k}{\Delta\mathrm{F}_k^T\Delta\mathrm{F}_k}\Delta\mathrm{F}_k^T, \tag{2.23}$$

este chamado de *bad Broyden method*.

Vamos voltar ao nosso exemplo

$$\begin{cases} f_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 1 & = 0 \\ f_2(\mathbf{x}) = 2x_1^2 + x_2^2 - 4x_3 & = 0 \\ f_3(\mathbf{x}) = 3x_1^2 - 4x_2 + x_3^2 & = 0 \end{cases} \tag{2.24}$$

e novamente iniciar com $\mathbf{x}_0 = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$. Os elementos da matriz jacobiana são dados por $\mathrm{J}_{ij} = \partial f_i/\partial x_j$ e podem ser estimados na 1a iteração como

$$\mathrm{J}_{ij} = \frac{f_i(x_j + h_j) - f_i(x_j)}{h_j}, \quad \text{onde} \ \ h_j \ \ \text{tem um valor pequeno.} \tag{2.25}$$

Usando $h_j = 0.001$, $j = 1, \cdots, n$, temos uma 1a. estimativa de $\mathrm{J}_0 = \begin{bmatrix} 1.001 & 1.001 & 1.001 \\ 2.002 & 1.001 & -4 \\ 3.003 & -4 & 1.001 \end{bmatrix}$

e seguimos usando

$$\mathrm{J}_0\Delta\mathbf{x}_0 = \mathrm{F}(\mathbf{x}_0), \quad \mathrm{F}(\mathbf{x}_0) = \begin{bmatrix} 0.25 \\ 1.25 \\ 1 \end{bmatrix} \tag{2.26}$$

Resolvendo o sistema temos $\Delta\mathbf{x}_0 = \begin{bmatrix} 0.37469 \\ 0.00002 \\ -0.12496 \end{bmatrix} \rightarrow \mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}_0 = \begin{bmatrix} 0.87469 \\ 0.50002 \\ 0.37504 \end{bmatrix}$.

[1] Ver por exemplo em Press *et al.* (1992)

Agora podemos estimar J_1 usando a expressão (2.20),

$$\mathrm{J}_1 = \mathrm{J}_0 + \frac{\mathrm{F}(\mathbf{x}_1) - \mathrm{F}(\mathbf{x}_0) - \mathrm{J}_0\Delta\mathbf{x}_0}{\|\Delta\mathbf{x}_0\|^2}\Delta\mathbf{x}_0^T = \begin{bmatrix} 1.37509 & 1.00102 & -0.87624 \\ 2.67457 & 1.00104 & -4.22431 \\ 4.04965 & -3.9993 & -0.65193 \end{bmatrix} \tag{2.27}$$

e prosseguimos com $\mathrm{J}_1\Delta\mathbf{x}_1 = \mathrm{F}(\mathbf{x}_1)$, obtendo $\Delta\mathbf{x}_1 = \begin{bmatrix} -0.1935 \\ -0.00233 \\ -0.00349 \end{bmatrix}$

$\rightarrow \mathbf{x}_2 = \mathbf{x}_1 + \Delta\mathbf{x}_1 = \begin{bmatrix} 0.76534 \\ 0.49770 \\ 0.37154 \end{bmatrix}$ e usando novamente a expressão (2.20) obtemos J_2 e assim sucessivamente. Poderíamos de forma equivalente ter usado a expressão (2.22) mas logo no início precisaríamos ter obtido a matriz inversa do jacobiano.

Referências adicionais:
Press *et al.* (1992)
Burden and Faires (2003), cap. 10.

Exercícios

2.1 Calcule com uma calculadora não programável a raiz positiva de $x^2 - 5 = 0$ usando o método de bissecção, até o intervalo ser $\leqslant 0.001$.

2.2 A equação $x^2 - 3x + 2 = 0$ pode ser escrita como $x = G(x)$ de diversas formas diferentes para aplicação do método de substituições sucessivas. Determine analiticamente a região de convergência para as raízes $x = 1, 2$ e faça os gráficos de convergência $y = G(x)$ superposto à reta $y = x$ para os seguintes casos:

a) $x_{n+1} = (x_n^2 + 2)/3$
b) $x_{n+1} = \sqrt{(3x_n - 2)}$

2.3 Com uma calculadora não programável, encontre a raiz positiva de $\sin(x) = x/2$ usando os métodos de Newton-Raphson e secantes, com erro $\epsilon \leqslant 10^{-4}$.

Capítulo 3

Matrizes e Sistemas Lineares

3.1 Introdução

A solução de sistemas de equações lineares é um dos problemas mais frequentes na área científica e assuma a forma geral

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned} \tag{3.1}$$

Isto equivale a $A\mathbf{x} = \mathbf{b}$, ou

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} , \tag{3.2}$$

onde A é uma matriz $n \times n$, $\mathbf{b}$ é um vetor de dimensão n e $\mathbf{x}$ o vetor de dimensão n a ser determinado. O método mais conhecido de solução é a regra de Cramer dada por

$$x_i = \frac{1}{\det(A)} \begin{vmatrix} a_{11} & a_{12} & ... & a_{1,i-1} & b_1 & a_{1,i+1} & ... & a_{1n} \\ a_{11} & a_{12} & ... & a_{2,i-1} & b_2 & a_{2,i+1} & ... & a_{2n} \\ \vdots & & & & & & & \\ a_{n1} & a_{n2} & ... & a_{n,i-1} & b_n & a_{n,i+1} & ... & a_{nn} \end{vmatrix} \tag{3.3}$$

No entanto, se tentarmos resolver um sistema de 20 equações por Cramer teremos 21 determinantes de ordem 20. Cada determinante é decomposto em

$$a_{11}\text{det}_{19} + a_{12}\text{det}_{19} + ... + a_{1,20}\text{det}_{19}$$

e assim por diante, até chegarmos em determinantes de ordem 2 teremos $21 \times 20! \approx 10^{20}$ operações. Para um sistema $n \times n$ teremos $(n+1)!$ operações.

Vamos definir a unidade flop $\equiv$ uma operação de ponto flutuante. Uma calculadora científica simples é capaz de poucos flop/s (operações de ponto flutuante por segundo). Isso pode ser inferido pelo tempo de resposta de calcular 69! , ou rodar um pequeno programa em loop longo

que podemos estimar o número de operações. Para computadores o pico teórico de flops/s é dado por

$$\text{no. de núcleos} \times \text{frequência} \times \text{operações por ciclo} \tag{3.4}$$

Um computador pessoal (PC) é capaz de realizar $\approx$ 1000 Mflops/s $=10^9$ flop/s. Para realizar 10^{20} operações de ponto flutuante num PC precisamos de um tempo dado por

$$t = \frac{10^{20}}{10^9/s} = 10^{11}\text{s} \approx 100.000 \text{ anos} .$$

Consultando o site do TOP 500, que lista os mais poderosos supercomputadores do planeta, veremos que atualmente o topo da lista é ocupado pelo SUMMIT (Oak Ridge National Laboratory-Tennessee- EUA) que é capaz de realizar 148 petaflop/s $=148\times10^{15}$ flop/s. Nesse supercomputador, o tempo de execução seria

$$t = \frac{10^{20}}{15 \times 10^{16}} \text{ s} \approx 700 \text{ s} .$$

Só que o custo do SUMMIT é cerca de US$ 200 milhões ! Além disso se aumentássemos levemente para um sistema com 25 equações, já seria inviável e levaria 1000 anos... A Regra de Cramer não é eficiente e além disso introduz muitos erros de arredondamento. Optaremos por métodos mais diretos para a solução de sistemas de equações lineares.

Podemos ter matrizes cheias ou matrizes esparsas. Abaixo exemplos de matrizes esparsas.

Sistemas esparsos possuem soluções particulares e mais rápidas que os cheios. Estudaremos inicialmente métodos de solução de matrizes cheias.

3.2 Métodos diretos

3.2.1 Método de Eliminação de Gauss

Também conhecido como **Método de Escalonamento**. É o método mais simples para solução de um sistema de equações.

Descrição do método

Seja, por exemplo, o seguinte sistema de equações

$$\begin{array}{rcrcrcr} 2x & + & y & + & z & = & 7 \\ 4x & + & 4y & + & 3z & = & 21 \\ 6x & + & 7y & + & 4z & = & 32 \end{array} \tag{3.5}$$

Multiplicando a 1a. equação por (-2) e somando na 2a. e multiplicando a 1a. equação por (-3) e somando na 3a. temos

$$\begin{array}{rcrcrcr} 2x & + & y & + & z & = & 7 \\ & & 2y & + & z & = & 7 \\ & & 4y & + & z & = & 11 \end{array} \tag{3.6}$$

Multiplicando a 2a. equação por (-2) e somando na 3a temos

$$\begin{array}{rcrcrcr} 2x & + & y & + & z & = & 7 \\ & & 2y & + & z & = & 7 \\ & & & & -z & = & -3 \end{array} \tag{3.7}$$

Da 3a. equação temos $-z = -3 \rightarrow z = 3$. Substituindo o valor de x na 2a. equação temos $2y + 3 = 7 \rightarrow y = 2$ e finalmente substituindo os valores de z e y na 1a. equação temos

finalmente $2x + 2 + 3 = 7 \rightarrow x = 1$. Esta última sequência de operações é conhecida como substituição para trás (*"backward substitution"*).

Na forma matricial estamos resolvendo

$$\begin{bmatrix} 2 & 1 & 1 \\ 4 & 4 & 3 \\ 6 & 7 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ 21 \\ 32 \end{bmatrix} , \tag{3.8}$$

e transformamos o problema em

$$\begin{bmatrix} 2 & 1 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ 7 \\ -3 \end{bmatrix} , \tag{3.9}$$

onde agora do lado esquerdo temos uma *matriz triangular superior*.

Poderíamos ter trabalhado só com os números, sem escrever as incógnitas. Para tanto é conveniente escrever a chamada matriz aumentada dada por

$$\left[\begin{array}{ccc|c} 2 & 1 & 1 & 7 \\ 4 & 4 & 3 & 21 \\ 6 & 7 & 4 & 32 \end{array}\right] . \tag{3.10}$$

Como antes, multiplicamos a 1a. linha por (-2) e somamos na 2a e multiplicamos a 1a linha por (-3) e somamos na 3a, obtendo

$$\left[\begin{array}{ccc|c} 2 & 1 & 1 & 7 \\ 0 & 2 & 1 & 7 \\ 0 & 4 & 1 & 11 \end{array}\right] . \tag{3.11}$$

e finalmente multiplicamos a 2a. equação por (-2) e somamos na 3a.

$$\left[\begin{array}{ccc|c} 2 & 1 & 1 & 7 \\ 0 & 2 & 1 & 7 \\ 0 & 0 & -1 & -3 \end{array}\right] . \tag{3.12}$$

Assim, usando a matriz aumentada podemos realizar todas as operações no computador sem fazer referencia 'as variáveis x, y, z.

Generalizando, o sistema inicial após eliminação fica na forma da matriz triangular superior

$$\begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1n} \\ & a'_{22} & \cdots & a'_{2n} \\ & 0 & \ddots & \vdots \\ & & & a'_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_n \end{bmatrix} \tag{3.13}$$

e a solução final é feita por substituição para trás dada por

$$a'_{ii} x_i + \sum_{j=i+1}^{n} a'_{ij} x_j = b'_i , \tag{3.14}$$

ou seja,

$$x_i = \frac{1}{a'_{ii}} \left(b'_i - \sum_{j=i+1}^{n} a'_{ij} x_j \right) . \tag{3.15}$$

Resumindo, o método de Eliminação de Gauss é dividido em dois estágios, eliminação até a obter a matriz triangular superior e substituição para trás, até o resultado final.

Número de operações

Assim como fizemos com a Regra de Cramer, podemos contar quantas operações são realizada no Método de Eliminação de Gauss. Se iniciarmos com a matriz aumentada

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right], \tag{3.16}$$

na eliminação da 1a. coluna teremos $n-1$ linhas e para cada uma delas teremos $n+1$ multiplicações[1] e n adições. Considerando todas as colunas a serem eliminadas, podemos construir uma tabela na forma

Eliminação	Multiplicações	Adições
1a. coluna	$(n-1)(n+1)$	$(n-1)n$
2a. coluna	$(n-2)n$	$(n-2)(n-1)$
$\vdots$	$\vdots$	$\vdots$
$(n-1)$a. coluna	$1 \cdot 3$	$1 \cdot 2$
TOTAL	$\sum_{i=1}^{n-1} i(i+2)$	$\sum_{i=1}^{n-1} i(i+1)$

Para multiplicações a soma resulta em[2]

$$\begin{aligned} \sum_{i=1}^{n-1} i(i+2) &= \sum_{i=1}^{n-1} i^2 + \sum_{i=1}^{n-1} 2i = \sum_{i=1}^{n} i^2 - n^2 + 2\left(\sum_{i=1}^{n} i - n\right) \\ &= \frac{n(n+1)(2n+1)}{6} - n^2 - 2\left(\frac{n(n+1)}{2} - n\right) = \frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}\ . \end{aligned} \tag{3.17}$$

De forma análoga, para adições teremos

$$\sum_{i=1}^{n-1} i(i+1) = \frac{n^3}{3} - \frac{n}{3}\ . \tag{3.18}$$

Já para a substituição para trás podemos também construir uma tabela de operações

Substituição	Multiplicações	Adições
1o.passo, linha n	1	0
2o.passo, linha $n-1$	2	1
$\vdots$	$\vdots$	$\vdots$
n o. passo coluna n	n	$n-1$
TOTAL	$n(n+1)/2$	$(n-1)n/2$

[1] Para descobrir o multiplicador da linha precisamos fazer uma divisão que, em linguagem binária, pode ter muito mais ciclos que uma operação de multiplicação. Por simplicidade contamos essa divisão como se fosse uma multiplicação.

[2] As fórmulas de $\sum_{i=1}^{n} i$ e $\sum_{i=1}^{n} i^2$ são bem conhecidas e existem generalizações para $\sum_{i=1}^{n} i^k$, k inteiro, chamadas de *fórmulas de Faulhaber*.

Finalmente, para a Eliminação de Gauss teremos um total de multiplicações dada pela soma dos resultados de (3.17) e da tabela acima, resultando em

$$\text{total de multiplicações} = \frac{n^3}{3} + n^2 - \frac{n}{3} \tag{3.19}$$

e analogamente, somando (3.18) com as adições da tabela acima teremos

$$\text{total de adições} = \frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}, \tag{3.20}$$

e para n grande temos entre multiplicações e adições $\approx 2n^3/3$ operações de ponto flutuante.

Voltando ao nosso problema da seção 3.1, a Eliminação de Gauss em um sistema com matriz 20×20 exigirá cerca de $2 \times 2^3 \times 10^3/3 \approx 10^3$flop e com um PC com 1000 Mflop/s será resolvido no tempo

$$t \approx \frac{10^3 \text{ flop}}{1000 \text{ Mflop/s}} \approx 10^{-6}\text{s}, \tag{3.21}$$

isso só considerando tempo de CPU, sem contar perda de eficiência no acesso à memória.

3.2.2 Pivotamento

Os elementos das digonais usados para eliminação são chamados de pivôs. Dependendo dos seus valores, podem ocorrer grandes erros nas soluções. Veremos esse fenômeno no seguinte exemplo

$$\begin{bmatrix} 10 & -7 & 0 \\ -3 & 2.099 & 6 \\ 5 & -1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 3.901 \\ 6 \end{bmatrix}, \tag{3.22}$$

cuja solução exata é $\mathbf{x} = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$.

Vamos considerar que no sistema de ponto flutuante tenha apenas 5 algarismos significativos. Multiplicando a 1a. equação por (0.3) e somando na 2a. e também multiplicando a 1a. equação por (-0.5) e somando na 3a. temos

$$\left[\begin{array}{ccc|c} 10 & -7 & 0 & 7 \\ 0 & -0.001 & 6 & 6.001 \\ 0 & 2.5 & 5 & 2.5 \end{array}\right] \tag{3.23}$$

Multiplicando a 2a. equação por $-2.5/(-0.001) = 2500$ e somando na 3a.

$$\left[\begin{array}{ccc|c} 10 & -7 & 0 & 7 \\ 0 & -0.001 & 6 & 6.001 \\ 0 & 0 & 15005 & 15004 \end{array}\right] \tag{3.24}$$

onde a última linha foi calculada na forma

$$\begin{array}{rr} 6.001 \times 2500 & = 15002.\not{5} \\ & \underline{+2.5} \\ & = 15004 \end{array} \tag{3.25}$$

Na substituição para trás teremos

$$\begin{aligned} x_3 &= \frac{15004}{15005} = 0.99993 \\ x_2 &= \frac{6.001 - 6 \times 0.99993}{-0.001} = \frac{6.001 - 5.99958}{-0.001} = -1.5 \text{ (erro muito grande)} \\ x_1 &= \frac{7 + 7 \times (-1.5)}{10} = \frac{7 - 10.5}{10} = -0.35 \end{aligned} \tag{3.26}$$

O pivô (-0.001) é muito pequeno em relação aos outros coeficientes o que resulta em um enorme multiplicador (2500) que gera erros em ponto flutuante. Estes erros por sua vez são ampliados devido à divisão pleo pivô. No método de Eliminação de Gauss utilizaremos o *pivotamento parcial* que consiste em trocar linhas de forma que tenhamos o maior valor absoluto do pivô. Isto garantirá multiplicadores $\leqslant 1$ em módulo como vemos a seguir:

$$\left[\begin{array}{ccc|c} 10 & -7 & 0 & 7 \\ 0 & -0.001 & 6 & 6.001 \\ 0 & 2.5 & 5 & 2.5 \end{array}\right] \rightarrow \text{pivotamento parcial} \rightarrow \left[\begin{array}{ccc|c} 10 & -7 & 0 & 7 \\ 0 & 2.5 & 5 & 2.5 \\ 0 & -0.001 & 6 & 6.001 \end{array}\right]$$

O multiplicador agora é $-0.001/(-2.5) = 0.0004$. Na 2a. equação teremos

$$0.0004 \times \left[\begin{array}{ccc|c} 0 & 2.5 & 5 & 2.5 \end{array}\right] \rightarrow \begin{array}{ccc|c} 0 & 0.001 & 0.002 & 0.001 \\ 0 & -0.001 & 6 & 6.001+ \\ \hline 0 & 0 & 6.002 & 6.002 \end{array}$$

e chegamos à matriz triangular superior

$$\left[\begin{array}{ccc|c} 10 & -7 & 0 & 7 \\ 0 & 2.5 & 5 & 2.5 \\ 0 & 0 & 6.002 & 6.002 \end{array}\right], \text{ donde obtemos } x_3 = 1,\ x_2 = -1,\ x_1 = 0.$$

3.2.3 Matriz Inversa por Gauss

Num dos livros da bibliografia há a citação "almost NEVER INVERT A MATRIX"pois essa inversão e posterior multiplicação causa muitos erros de ROUNDOFF. Mas de qualquer forma há ocasiões que necessitamos de fato da matriz inversa e podemos usar uma variante do método de eliminação de Gauss obtê-la.

O algoritmo inicia-se com uma matriz aumentada tendo do lado esquerdo a matriz A a ser invertida e do lado direito a matriz identidade. Fazendo operações de eliminação para frente, normalizando o pivô e eliminação para trás, transforma-se a matriz do lado esquerdo em matriz identidade e do lado direito forma-se a matriz inversa como ilustramos no exemplo seguir.

$$\left[\begin{array}{ccc|ccc} 2 & 1 & 1 & 1 & & \\ 4 & 4 & 3 & & 1 & \\ 6 & 7 & 4 & & & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 2 & 1 & 1 & 1 & & \\ 0 & 2 & 1 & -2 & 1 & \\ 0 & 4 & 1 & -3 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 2 & 1 & 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & -2 & 1 & 0 \\ 0 & 0 & -1 & 1 & -2 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 2 & 1 & 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -1 & 2 & -1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 2 & 1 & 0 & 2 & -2 & 1 \\ 0 & 2 & 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & -1 & 2 & -1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 2 & 1 & 0 & 2 & -2 & 1 \\ 0 & 1 & 0 & -1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 & -1 & 2 & -1 \end{array}\right]$$

$$\left[\begin{array}{ccc|ccc} 2 & 0 & 0 & 5/2 & -3/2 & 1/2 \\ 0 & 1 & 0 & -1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 & -1 & 2 & -1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 5/4 & -3/4 & 1/4 \\ 0 & 1 & 0 & -1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 & -1 & 2 & -1 \end{array}\right]$$

e o lado direito é a matriz inversa A^{-1}.

Exercício 3.1

Verifique que a matriz final obtida do lado direito é de fato a inversa da matriz inicial do lado esquerdo.

Exercício 3.2

Prove que o algoritmo conduz à inversa.

3.2.4 Diferentes vetores do lado direito

Em alguns problemas, os valores dos coeficientes da matriz não se alteram, mas precisamos calcular a solução para diferentes valores do vetor do lado direito. Assim teremos uma série de problemas na forma $A\mathbf{x_1} = \mathbf{b_1}$; $A\mathbf{x_2} = \mathbf{b_2} \cdots$. Ou seja

$$\left[\begin{array}{c|c} & \\ A & \mathbf{b_1} \\ & \end{array}\right] \; ; \; \left[\begin{array}{c|c} & \\ A & \mathbf{b_2} \\ & \end{array}\right] \cdots$$

Em vez de fazer a mesma eliminação diversas vezes podemos guardar a sequência de operações aplicadas na triangulação superior da matriz A para depois aplicar nos vetores $\mathbf{b_1}$, $\mathbf{b_2}$ $\cdots$

Exemplo

Seja a matriz

$$\begin{bmatrix} 2 & 6 & -2 \\ 1 & 3 & -4 \\ 3 & 6 & 9 \end{bmatrix} \quad \text{e} \quad \mathbf{p} = \begin{bmatrix} \\ \\ \end{bmatrix} \tag{3.27}$$

o vetor de pivotamento que contém o no. da linha que foi pivotada.

Vamos fazer o pivotamento da 3a. linha, e teremos

$$\begin{bmatrix} 2 & 6 & -2 \\ 1 & 3 & -4 \\ 3 & 6 & 9 \end{bmatrix} \quad ; \quad \begin{bmatrix} 3 \\ \\ \end{bmatrix} \tag{3.28}$$

Multiplicamos a 1a. linha por (-1/3) e somamos na 2a linha; multiplicamos a 1a linha por (-2/3) e somamos na 3a linha e obtemos

$$\left[\begin{array}{ccc} 3 & 6 & 9 \\ \hline -1/3 & 1 & -7 \\ -2/3 & 2 & -8 \end{array}\right] \quad ; \quad \begin{bmatrix} 3 \\ \\ \end{bmatrix} \tag{3.29}$$

onde agora nas posições que foram eliminadas guardamos os multiplicadores.

Na sequência pivotamos a 3a. linha,

$$\left[\begin{array}{ccc} 3 & 6 & 9 \\ \hline -1/3 & 2 & -8 \\ -2/3 & 1 & -7 \end{array}\right] \quad ; \quad \left[\begin{array}{c} 3 \\ 3 \\ \end{array}\right] \tag{3.30}$$

Multiplicando 2a. linha por $(-1/2)$ e somando na 3a.,

$$\left[\begin{array}{ccc} 3 & 6 & 9 \\ \hline -1/3 & 2 & -8 \\ -2/3 & -1/2 & -3 \end{array}\right] \quad ; \quad \left[\begin{array}{c} 3 \\ 3 \\ \end{array}\right] \tag{3.31}$$

Para uma aplicação, vamos agora fornecer o vetor $\mathbf{b} = \begin{bmatrix} 4 \\ -7 \\ 39 \end{bmatrix}$. No 1o. passo devemos trocar a 1a. linha com a linha p(1)=3 e obtemos $\begin{bmatrix} 39 \\ -7 \\ 4 \end{bmatrix}$. Multiplicamos a 1a. linha por $(-1/3)$ e somamos na 2a. E também multiplicamos a 1a. linha por $(-2/3)$ e somamos na 3a $\rightarrow \begin{bmatrix} 39 \\ -20 \\ -22 \end{bmatrix}$. Trocamos a 2a. linha com linha p(2)=3 $\rightarrow \begin{bmatrix} 39 \\ -22 \\ 20 \end{bmatrix}$. Ainda, multiplicamos 1a. linha por $(-1/2)$ e somamos na 3a $\rightarrow \begin{bmatrix} 39 \\ -22 \\ -9 \end{bmatrix}$. Ficamos então com o sistema

$$\begin{bmatrix} 3 & 6 & 9 \\ 0 & 2 & -8 \\ 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 39 \\ -22 \\ -9 \end{bmatrix} \tag{3.32}$$

e finalmente fazendo substituição para trás obtemos $x_3 = 3$, $x_2 = 1$, $x_1 = 2$.

Podemos escrever uma rotina para eliminação guardando as informações de pivotamento e triangulação na forma

Algoritmo 3.1: Eliminação.

```
1  begin program
2      A ← Matriz
3      for k = 1 até (n − 1)
4          /* Determine o índice do pivô l
5            Troque as linhas k e l com colunas indo de k até n */
6          p(k) ← l
7          for j = k + 1 até n
8              /* Determinação dos multiplicador abaixo do pivô (coluna k) para linha j
9                na linha j coluna k guarde os multiplicador (onde será eliminado o elemento),
10               adicione múltiplos da linha k àlinha j da coluna k + 1 até n */
11         end for
12     end for
13 end program
```

Exercício 3.3

Construa uma outra rotina que use as informações guardadas pela da rotina de eliminação e que dado um vetor $\mathbf{b}$ *determine a solução* $\mathbf{x}$*. Construa o programa numa linguagem desejada e verifique se é capaz de resolver o exemplo na equação (3.27).*

3.2.5 Decomposição LU

Suponha que possamo escrever a matriz A como o produto de uma matriz triangular inferior por uma matriz triangular superior na forma A=LU, i.e.,

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & \mathbf{0} & \\ l_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & \mathbf{0} & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix} . \tag{3.33}$$

Nesse caso podemos escrever o sistema na forma

$$\mathrm{A}\mathbf{x} = (\mathrm{LU})\mathbf{x} = \mathrm{L}(\mathrm{U}\mathbf{x}) = \mathbf{b}. \tag{3.34}$$

Se definirmos $\mathbf{y} \equiv \mathrm{U}\mathbf{x}$ podemos resolver $\mathrm{L}\mathbf{y} = \mathbf{b}$ e depois $\mathrm{U}\mathbf{x} = \mathbf{y}$. $\mathrm{L}\mathbf{y} = \mathbf{b}$ pode ser resolvido por substituição para frente

$$\begin{aligned} y_1 &= b_1 \\ y_i &= b_i - \sum_{j=1}^{l-1} l_{ij} y_j, \quad i = 2, 3, \cdots, n. \end{aligned} \tag{3.35}$$

e $\mathrm{U}\mathbf{x} = \mathbf{y}$ pode ser resolvido por substituição para trás

$$\begin{aligned} x_n &= \frac{y_{nn}}{u_{nn}} \\ x_i &= \frac{1}{u_{ii}} \left[y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right], \quad i = n-1, n-2, \cdots, 1. \end{aligned} \tag{3.36}$$

Algoritmos para realização da decomposição LU são mais elaborados quando incluem pivotamento e podem ser encontrados em Press et al. (1992).

Veremos o caso mais simples de **matriz tridiagonal**. Seja A uma matriz tridiagonal dada por

$$\mathrm{A} = \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \tag{3.37}$$

então a decomposição tem a forma simples

$$\mathrm{L} = \begin{bmatrix} 1 & & & & \\ l_2 & 1 & & & \\ & l_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & l_n & 1 \end{bmatrix}; \quad \mathrm{U} = \begin{bmatrix} u_1 & v_1 & & & \\ & u_2 & v_2 & & \\ & & \ddots & \ddots & \\ & & & u_{n-1} & v_{n-1} \\ & & & & u_n \end{bmatrix}. \tag{3.38}$$

Exercício: Se A é tridiagonal mostre que A pode ser LU decomposta sendo

$$\begin{aligned} & u_1 = b_1 \ , \quad v_i = c_i \ , \\ & l_j u_{j-1} = a_j \\ & l_j v_{j-1} + u_j = b_j \ , \ j = 2, 3, \cdots, n. \end{aligned} \tag{3.39}$$

Determinados L e U, procede-se como no caso geral, i.e., para encontrar a solução de A$\mathbf{x} = \mathbf{d}$ escrevemos LU$\mathbf{x} = \mathbf{d}$ e definindo $\mathbf{y} \equiv$ U$\mathbf{x}$ temos L$\mathbf{y} = \mathbf{d}$, U$\mathbf{x} = \mathbf{y}$ sendo estes dois últimos sistemas resolvidos por substituição para frente e para trás, respectivamente.

Assim, a solução de um sistema A$\mathbf{x} = \mathbf{d}$, sendo A triadiagonal, fica dada simplesmente por três laços

$$\begin{cases} u_1 = b_1 \\ l_j = a_j/u_{j-1} - l_i z_{i-1}, \ j = 2, 3, \cdots, n \\ u_j = b_j - l_j c_{j-1} \end{cases} \qquad \text{Decomposição LU.} \tag{3.40}$$

$$\begin{cases} z_1 = d_1 \\ z_i = d_i - l_i z_{i-1}, \ i = 2, 3, \cdots, n \end{cases} \qquad \text{Substituição para frente.} \tag{3.41}$$

$$\begin{cases} x_n = z_n/u_n \\ x_k = z_k - c_k x_{k+1}/u_k, k = n-1, n-2, \cdots, 1 \end{cases} \qquad \text{Substituição para trás.} \tag{3.42}$$

Os dois primeiros laços podem ser juntados em um só. O total de operações é $\mathcal{O}(n)$.

Exercício 3.4

Determine esse número de operações, contando divisões como multiplicações e subtrações como adições.

Podemos construir para matrizes cheias uma tabela comparativa do número de multiplicações (ou adições) de diversos métodos com respeito a diferentes problemas como vemos a seguir.

Número de multiplicações

Problema/Método	Elimin. de Gauss	Gauss-Jordan	LU
Solução de Sistemas	$\frac{1}{3}n^3$	$\frac{1}{2}n^3$	$\frac{1}{3}n^3$
m lados direitos	$\frac{1}{3}n^3 + \frac{1}{2}mn^2$	$\frac{1}{2}n^3 + mn^2$	$\frac{1}{3}n^3 + \frac{1}{2}mn^2$
Inversão de Matriz	n^3	n^3	n^3

3.2.6 Refinamento

Quando o sistema é resolvido, após a sequência de operações podem se acumular erros de roundoff, reduzindo a precisão da solução. Esse problema pode ser minimizado através do

refinamento, que descrevermos a seguir. Seja o sistema

$$\mathrm{A}\mathbf{x} = \mathbf{b} \ . \tag{3.43}$$

Resolvendo por Eliminação de Gauss obtemos um vetor solução $\mathbf{x}_0$. Vamos chamar de erro a diferença entre o valor verdadeiro e o valor obtido, $\mathbf{e}_0 \equiv \mathbf{x} - \mathbf{x}_0$. Substituindo nos sistema (3.43) temos

$$\mathrm{A}(\mathbf{x}_0 + \mathbf{e}_0) = \mathbf{b} \rightarrow \mathrm{A}\mathbf{e}_0 = \underbrace{\mathbf{b} - \mathrm{A}\mathbf{x}_0}_{\equiv \mathbf{r}(\text{resíduo})} \ , \tag{3.44}$$

onde chamamos a diferença de $\mathbf{b} - \mathrm{A}\mathbf{x}_0$ de **resíduo**. Resolvendo sistema (3.44) determinamos $\mathbf{e}_0$ e então $\mathbf{x} = \mathbf{x}_0 + \mathbf{e}_0$ deveria ser a solução correta mas ainda pode conter erros e podemos chamá-la de $\mathbf{x}_1 \equiv \mathbf{x}_0 + \mathbf{e}_0$. Define-se a seguir $\mathbf{e}_1 \equiv \mathbf{x} - \mathbf{x_1}$ e substitui-se em (3.43) e resolve-se $\mathrm{A}\mathbf{e}_1 = \mathbf{b} - \mathrm{A}\mathbf{x}_1$, obtendo-se $\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{e}_1$ e assim por diante, até convergir.

3.2.7 Sistemas mal-condicionados

Sistemas mal-condicionados são os quais pequenas modificações nos coeficientes acarretam grandes modificações no resultado. Em inglês são chamados de "ill-conditioned systems". Vamos ver através de um exemplo como isso ocorre. Seja o sistema

$$\begin{cases} x + y = 1 \\ 99x + 100y = 99.5 \end{cases} \tag{3.45}$$

cuja solução exata é $x = 0.5$, $y = 0.5$. Agora considere o sistema

$$\begin{cases} x + y = 1 \\ 99.4x + 99.9y = 99.2 \end{cases} \tag{3.46}$$

cuja solução única e exata é $x = 1.4$, $y = -0.4$, que é muito diferente da solução do sistema anterior.

Desenhando as retas correspondentes às equações podemos ver porque isso acontece. As retas são quase paralelas, i.e., quase linearmente dependentes, e pequenas mudanças na inclinação de uma das retas podem acarretar grandes mudanças no pontos de encontro, que correspondem às soluções.

Num sistema com n equações, pensando nos vetores linha (ou vetores coluna) $\mathbf{a}_1, \mathbf{a}_2 \cdots, \mathbf{a}_n$, se eles forem quase linearmente dependentes então o sistema será mal-condicionado. Uma maneira de "medir"o condicionamento de uma matriz é calcular seu determinante. Mas o valor do determinante depende da norma de cada um dos vetores.

Assim, normalizamos os vetores para depois para depois de calcular o determinante. Isto produzirá um valor entre 0 e 1. Se o módulo do determinante for próximo de zero então o sistema é quase singular e o sistema é mal-condicionado.

Na forma matricial, o exemplo fica

$$\begin{bmatrix} 1 & 1 \\ 99 & 100 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 99.5 \end{bmatrix} \tag{3.47}$$

Normalizando o vetores linha da matriz temos

$$\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 99/140.716 & 100/140.716 \end{bmatrix} \tag{3.48}$$

e o módulo do determinante fornece

$$\left| \frac{1}{\sqrt{2}} \frac{100}{140.716} - \frac{1}{\sqrt{2}} \frac{99}{140.716} \right| \approx 5 \times 10^{-3} \tag{3.49}$$

que é próximo de zero, ou seja, vetores quase paralelos. Há outras medidas de condicionamento de uma matriz assim como há fórmulas que relacionam o erro cometido no método de Gauss com essas medidas e com no. de algarismos significativos. Veja por exemplo Blum (1972) e Press. et al. (1992) – Singular Value Decomposition (SVD).

Dica

Pesquise **Matrizes de Hilbert**.

3.3 Métodos Iterativos

Também são chamados de métodos indiretos. O método de eliminação de Gauss se torna ineficiente quando o no. de equações atinge $n \gtrsim 100$, pois o no. de operações de ponto flutuante é $\mathcal{O}(n^3)$. Mais detalhes, ver Blum (1972) p. 131.

Nos métodos iterativos arbitra-se um vetor inicial $\mathbf{x}^{(0)}$ e calcula-se um vetor $\mathbf{x}^{(1)}$ como função de $\mathbf{x}^{(0)}$ e assim sucessivamente. Podemos escrever isso na forma

$$\mathbf{x}^{(k+1)} = g(\mathbf{x}^{(k)}) \ , \tag{3.50}$$

onde k é a k-ésima iteração, e iterar até obter uma precisão desejada que seja uma diferença suficientemente pequena entre $\mathbf{x}^{(k+1)}$ e $\mathbf{x}^{(k)}$.

3.3.1 Método de Jacobi

Dado o sistema linear $\mathrm{A}\mathbf{x} = \mathbf{b}$, explicitamente dado por

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ...a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ...a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + ...a_{nn}x_n &= b_n \ , \end{aligned} \tag{3.51}$$

podemos reescrevê-lo na forma

$$\begin{aligned} x_1 &= \frac{1}{a_{11}} \left(b_1 - a_{12}x_2 - a_{13}x_3 ... - a_{1n}x_n \right) \\ x_2 &= \frac{1}{a_{22}} \left(b_2 - a_{21}x_1 - a_{23}x_3 - ... - a_{2n}x_n \right) \\ &\vdots \\ x_n &= \frac{1}{a_{nn}} \left(b_n - a_{n1}x_1 - a_{n2}x_2 - ... - a_{n,n-1}x_{n-1} \right) \ . \end{aligned} \tag{3.52}$$

Escolhemos arbitrariamente um vetor inicial $\mathbf{x}^{(0)}$ e substituímos do lado direito das equações acima obtendo um novo vetor $\mathbf{x}^{(1)}$ e assim sucessivamente teremos as relações

$$\begin{aligned} x_1^{k+1} &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^k - a_{13}x_3^k ... - a_{1n}x_n^k\right) \\ x_2^{k+1} &= \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^k - a_{23}x_3^k - ... - a_{2n}x_n^k\right) \\ &\vdots \\ x_n^{k+1} &= \frac{1}{a_{nn}}\left(b_n - a_{n1}x_1^k - a_{n2}x_2^k - ... - a_{n,n-1}x_{n-1}^k\right) \ , \end{aligned} \tag{3.53}$$

ou de forma mais compacta

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^k\right) \ . \tag{3.54}$$

Exemplo: considere o seguinte sistema de equações

$$\begin{array}{rcrcrcr} 4x_1 & + & 2x_2 & + & x_3 & = & 11 \\ -x_1 & + & 2x_2 & & & = & 3 \\ 2x_1 & + & x_2 & + & 4x_3 & = & 16 \end{array} \tag{3.55}$$

cuja solução é $\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$. Reescrevemos as equações como

$$\begin{aligned} x_1 &= \frac{1}{4}\left(11 - 2x_2 - x_3\right) \\ x_2 &= \frac{1}{2}\left(3 + x_1\right) \\ x_3 &= \frac{1}{4}\left(16 - 2x_1 - x_2\right) \ , \end{aligned} \tag{3.56}$$

o que sugere o método iterativo

$$\begin{aligned} x_1^{k+1} &= \frac{1}{4}\left(11 - 2x_2^k - x_3^k\right) \\ x_2^{k+1} &= \frac{1}{2}\left(3 + x_1^k\right) \\ x_3^{k+1} &= \frac{1}{4}\left(16 - 2x_1^k - x_2^k\right) \ . \end{aligned} \tag{3.57}$$

Começando com um vetor arbitrário $\mathbf{x}^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ temos

$$\begin{aligned} x_1^{(1)} &= \frac{1}{4}\left(11 - 2\cdot 1 - 1\right) = 2 \\ x_2^{(1)} &= \frac{1}{2}\left(3 + 1\right) = 2 \\ x_3^{(1)} &= \frac{1}{4}\left(16 - 2\cdot 1 - 1\right) = \frac{13}{4} \ . \end{aligned} \tag{3.58}$$

Substituindo $\mathbf{x}^{(1)} = \begin{bmatrix} 2 \\ 2 \\ 13/4 \end{bmatrix}$ do lado direito do sistema (3.57) obtemos

$$\begin{aligned} x_1^{(2)} &= \frac{1}{4}\left(11 - 2\cdot 2 - \frac{13}{4}\right) = \frac{15}{16} \\ x_2^{(2)} &= \frac{1}{2}\left(3 + 2\right) = \frac{5}{2} \\ x_3^{(2)} &= \frac{1}{4}\left(16 - 2\cdot 2 - 2\right) = \frac{5}{2}\ . \end{aligned} \tag{3.59}$$

Colocando os resultados numa tabela obtemos

k	x_1^k	x_2^k	x_3^k	$\max\{\lvert x_i^k - x_i^{k-1}\rvert\}_{i=1,\cdots,n}$
0	1	1	1	—
1	2	2	$^{13}/_{4}$	$^{9}/_{4}$
2	$^{15}/_{16}$	$^{5}/_{2}$	$^{5}/_{2}$	$^{3}/_{4}$
3	$^{7}/_{8}$	$^{63}/_{32}$	$^{93}/_{32}$	$^{17}/_{32}$
4	$^{133}/_{128}$	$^{31}/_{16}$	$^{393}/_{128}$	$^{21}/_{128}$
5	$^{519}/_{512}$	$^{517}/_{256}$	$^{767}/_{256}$	$^{21}/_{256}$

e para $k = 5$ já estamos com uma diferença menor que 1% da solução exata.

Vamos agora analisar a convergência do método de Jacobi. Podemos escrever a matriz A como

$$\mathrm{A} = \mathbb{L} + \mathrm{D} + \mathbb{U}\ , \tag{3.60}$$

onde $\mathbb{L}$ é uma matrix triangular inferior (sem diagonal), $\mathbb{U}$ é uma matrix triangular superior (sem diagonal) e D é uma matriz diagonal. O sistema pode ser escrito como

$$\begin{aligned} \mathrm{A}\mathbf{x} &= (\mathbb{L} + \mathrm{D} + \mathbb{U})\mathbf{x} = \mathbf{b} \\ \mathrm{D}\mathbf{x} &= -(\mathbb{L} + \mathbb{U})\mathbf{x} + \mathbf{b} \\ \mathbf{x} &= \mathrm{D}^{-1}[-(\mathbb{L} + \mathbb{U})\mathbf{x} + \mathbf{b}] \\ \mathbf{x} &= \mathbb{J}\mathbf{x} + \mathbf{c} \end{aligned} \tag{3.61}$$

onde $\mathbb{J} \equiv -\mathrm{D}^{-1}(\mathbb{L} + \mathbb{U})$ e $\mathbf{c} \equiv \mathrm{D}^{-1}\mathbf{b}$. Explicitamente $\mathbb{J} = -\begin{bmatrix} 0 & \frac{a_{12}}{a_{11}} & \frac{a_{13}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \frac{a_{23}}{a_{22}} & \cdots & \frac{a_{2n}}{a_{22}} \\ \frac{a_{31}}{a_{33}} & \frac{a_{32}}{a_{33}} & 0 & \cdots & \frac{a_{3n}}{a_{33}} \\ \vdots & \vdots & \ddots & 0 & \vdots \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \cdots & \frac{a_{n,n-1}}{a_{nn}} & 0 \end{bmatrix}$

Aplicando o método iterativo de Jacobi temos

$$\mathbf{x}^{(k+1)} = \mathbb{J}\mathbf{x}^{(k)} + \mathbf{c} \tag{3.62}$$

Partindo de $\mathbf{x}^{(0)}$ e iterando sucessivamente temos

$$\mathbf{x}^{(k+1)} = \mathbb{J}^k\mathbf{x}^{(0)} + \left[\mathbb{1} + \mathbb{J} + \mathbb{J}^2 + \cdots \mathbb{J}^{k-1}\right]\mathbf{c} \tag{3.63}$$

Para que convirja requer que

$$\lim_{k\to\infty} \mathbb{J}^k = [0] \tag{3.64}$$

o que implica $\lim_{k\to\infty}[\mathbb{1} + \mathbb{J} + \mathbb{J}^2 + \cdots \mathbb{J}^{k-1}] = (\mathbb{1} - \mathbb{J})^{-1}$. Assim, quando (3.64) é satisfeita, $\mathbf{x} = \lim_{k\to\infty}$ existe e $\mathbf{x} = \mathbf{0} + (\mathbb{1} - \mathbb{J})^{-1}\mathbf{c}$, i.e., $(\mathbb{1} - \mathbb{J})\mathbf{x} = \mathbf{c}$ ou $\mathbf{x} = \mathbb{J}\mathbf{x} + \mathbf{c}$. Mas a condição (3.64) é válida se e somente se todos os autovalores da matriz $\mathbb{J}$ forem em módulo < 1. Seja $\rho_s \equiv \max\{|\lambda_1|, |\lambda_2|, \cdots |\lambda_n|\}$, onde ρ_s é chamado de raio de espectral e λ_i são os autovalores da matriz $\mathbb{J}$. Então para atingir precisão p após k iterações devemos ter

$$\rho_s^k \approx 10^{-p} \;\rightarrow\; k \approx \frac{p \ln 10}{\ln \rho_s} \tag{3.65}$$

Assim se ρ_s estiver próximo de 1 a convergência será muito lenta. Existem métodos de aceleração, ver por exemplo Quinney (1987), Press et al., seção 19.5 (1992). Determinar os autovalores da matriz $\mathbb{J}$ requererá outro algoritmo em geral. Na prática, muitas vezes é mais fácil testar numericamente a convergência.

Uma condição mais simples, porém apenas suficiente, que assegura a convergência do método é que os sistema possua diagonal principal estritamente dominante, ou seja,

$$|a_{ii}| > \sum_{\substack{j \\ j\neq i}} |a_{ij}| \;, \tag{3.66}$$

que é chamado de **Critério das Linhas**. Exercício: Mostre que o sistema

$$\begin{bmatrix} 4 & 2 & 1 \\ -1 & 2 & 0 \\ 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 3 \\ 16 \end{bmatrix} \tag{3.67}$$

satisfaz o Critério das Linhas.

Note que um sistema que não satisfaz o Critério das Linhas pode convergir. Alterando a ordem das linhas ou colunas pode tornar um sistema convergente em divergente e vice-versa.

3.3.2 Método de Gauss-Seidel

É muito semelhante ao Método de Jacobi, só que para calcular x_i^{k+1} aproveita-se os valores já calculados $x_1^{k+1}, x_2^{k+2}, \cdots, x_{i-1}^{k+1}$. As equações tomam a seguinte forma

$$\begin{aligned}
x_1^{k+1} &= \frac{1}{a_{11}}\left(b_1 - a_{12}x_2^k - a_{13}x_3^k - \ldots - a_{1n}x_n^k\right) \\
x_2^{k+1} &= \frac{1}{a_{22}}\left(b_2 - a_{21}x_1^{k+1} - a_{23}x_3^k - \ldots - a_{2n}x_n^k\right) \\
x_3^{k+1} &= \frac{1}{a_{33}}\left(b_3 - a_{31}x_1^{k+1} - a_{32}x_3^{k+1} - a_{23}x_3^k - \ldots - a_{3n}x_n^k\right) \\
&\vdots \\
x_n^{k+1} &= \frac{1}{a_{nn}}\left(b_n - a_{n1}x_1^{k+1} - a_{n2}x_2^{k+1} - \ldots - a_{n,n-1}x_{n-1}^{k+1}\right) \;,
\end{aligned} \tag{3.68}$$

em forma mais compacta pode ser escrito

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k\right) \tag{3.69}$$

Num programa de computador pode ser simplesmente escrito como um único vetor,

$$x_i \leftarrow \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j\right) \;. \tag{3.70}$$

Exemplo: Considere novamente o sistema de equações dado por (3.55). Como vimos ele pode ser reescrito na forma

$$\begin{aligned} x_1 &= \frac{1}{4}\left(11 - 2x_2 - x_3\right) \\ x_2 &= \frac{1}{2}\left(3 + x_1\right) \\ x_3 &= \frac{1}{4}\left(16 - 2x_1 - x_2\right) \ , \end{aligned} \tag{3.71}$$

e vamos escolher novamente o vetor $\mathbf{x}^{(0)} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$. Aplicando o método de Gauss-Seidel teremos

$$\begin{aligned} x_1^{(1)} &= \frac{1}{4}\left(11 - 2 \cdot 1 - 1\right) = 2 \\ x_2^{(1)} &= \frac{1}{2}\left(3 + 2\right) = \frac{5}{2} \\ x_3^{(1)} &= \frac{1}{4}\left(16 - 2 \cdot 2 - \frac{5}{2}\right) = \frac{19}{8} \ , \quad \text{ou seja,} \end{aligned} \tag{3.72}$$

$\mathbf{x}^{(1)} = \begin{bmatrix} 2 \\ 5/2 \\ 19/8 \end{bmatrix}$ e sucessivamente teremos $\mathbf{x}^{(2)} = \begin{bmatrix} 29/32 \\ 125/64 \\ 783/256 \end{bmatrix}$; $\mathbf{x}^{(3)} = \begin{bmatrix} 1033/1024 \\ 4095/2048 \\ 24541/8192 \end{bmatrix} \approx \begin{bmatrix} 1.0087 \\ 1.9995 \\ 2.9957 \end{bmatrix}$.

Note que neste exemplo a taxa de convergência é muito maior que no método de Jacobi.

Convergência

Vamos estudar agora a convergência do Método de Gauss-Seidel. Novamente podemos escrever A como

$$\mathrm{A} = \mathbb{L} + \mathrm{D} + \mathbb{U} \ , \tag{3.73}$$

e o sistema pode ser escrito como

$$\begin{aligned} \mathrm{A}\mathbf{x} &= (\mathbb{L} + \mathrm{D} + \mathbb{U})\mathbf{x} = \mathbf{b} \\ \mathrm{D}\mathbf{x} &= -(\mathbb{L} + \mathbb{U})\mathbf{x} + \mathbf{b} \\ \mathbf{x} &= \mathrm{D}^{-1}[-(\mathbb{L} + \mathbb{U})\mathbf{x} + \mathbf{b}] \end{aligned} \tag{3.74}$$

Aplicando o método de Gauss-Seidel temos

$$\mathbf{x}^{(k+1)} = \mathrm{D}^{-1}[-(\mathbb{L}\mathbf{x}^{(k+1)} + \mathbb{U}\mathbf{x}^{(k)}) + \mathbf{b}] \ , \tag{3.75}$$

multiplicando tudo por D e rearranjando

$$(\mathrm{D} + \mathbb{L})\mathbf{x}^{(k+1)} = -\mathbb{U}\mathbf{x}^{(k)} + \mathbf{b} \ , \tag{3.76}$$

$$\mathbf{x}^{(k+1)} = -(\mathrm{D} + \mathbb{L})^{-1}\mathbb{U}\mathbf{x}^{(k)} + (\mathrm{D} + \mathbb{L})^{-1}\mathbf{b} \ , \tag{3.77}$$

ou

$$\mathbf{x}^{(k+1)} = \mathbb{G}\mathbf{x}^{(k)} + \mathbf{c} \ , \tag{3.78}$$

onde $\mathbb{G} \equiv -(\mathrm{D} + \mathbb{L})^{-1}\mathbb{U}$ e $\mathbf{c} \equiv (\mathrm{D} + \mathbb{L})^{-1}\mathbf{b}$.

A expressão (3.78) está na mesma forma que a equação (3.62) estudada no método de Jacobi, portanto uma condição necessária e suficiente para a convergência é que os autovalores de $\mathbb{G}$ devem ser todos em módulo menores que 1.

Ou seja, $\det[\mathbb{G} - \lambda\mathbb{1}] = 0$, $|\lambda_i| < 1$, todo i. A convergência (ou não) tanto no método de Jacobi como no método de Gauss-Seidel independem do vetor inicial escolhido.

O critério das linhas também pode ser aplicado ao método e Gauss-Seidel, também como condição apenas suficiente.

Para o método de Gauss-Seidel exite outro critério menos restritivo que o critério das linhas, chamado de critério de Sassenfeld. O critério de Sassenfeld afirma que se definirmos $\beta_i's$ de forma que

$$\begin{aligned} \beta_1 &= \frac{|a_{12}| + |a_{13}| + ... + |a_{1n}|}{|a_{11}|} \,, \\ \beta_i &= \frac{\sum_{j=1}^{i-1} \beta_j |a_{ij}| + \sum_{j=i+1}^{n} a_{ij}}{|a_{ii}|} \,, \end{aligned} \tag{3.79}$$

então o método de Gauss-Seidel converge se $\beta_i < 1$, para todo $i = 1, \cdots, n$. A prova pode ser encontrada em Asano & Colli (2009).

Exercício: Seja o sistema $\mathrm{A}\mathbf{x} = \mathbf{b}$ com $\mathrm{A} = \begin{bmatrix} -1 & -5 & 4 \\ -6 & -2 & 8 \\ 3 & -1 & 1 \end{bmatrix}$, Verifique se a matriz A satisfaz o Critério das Linhas ou Sassenfled por alguma permutação de linhas.

Solução: permutando linhas, buscando colocar em módulo os maiores termos de cada linha na diagonal obtemos

$\begin{bmatrix} 3 & -1 & 1 \\ -1 & -5 & 4 \\ -6 & -2 & 8 \end{bmatrix} \rightarrow$ não satisfaz critério das linhas e portanto não sabemos se os métodos de Jacobi ou Gauss-Seidel convergem por esse critério. Para aplicar o critério de Sassenfeld calculamos

$$\begin{aligned} \beta_1 &= \frac{|-1| + |1|}{3} = \frac{2}{3} \,, \\ \beta_2 &= \frac{\frac{2}{3}|-1| + 4}{5} = \frac{14}{15} \,, \\ \beta_3 &= \frac{\frac{2}{3}|-6| + \frac{14}{15}|-2|}{8} = \frac{11}{15} \,. \end{aligned} \tag{3.80}$$

Como $\beta_1, \beta_2, \beta_3 < 1$ então satisfaz o critério de Sassenfeld e portanto o método de Gauss-Seidel converge.

Existem outros métodos iterativos como o **Método do Gradiente Conjugado**. Eles foram desenvolvidos para matrizes simétricas. Para matrizes esparsas têm convergência muito superior ao métodos de Jacobi e Gauss-Seidel. Devem ser empregados em sistemas gigantes. Para matrizes assimétricas foi desenvolvido o gradiente bi-conjugado que não é estável e uma outra versão estabilizada denominada gradiente bi-conjugado estabilizado (BI-CGSTAB) foi desenvolvida por Van der Vorst (1992).

Exercícios

3.5 Considere o sistema abaixo na forma $\mathbf{Ax} = \mathbf{b}$ dado por

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & -2 \\ 1 & 3 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 14 \\ 4 \\ 4 \end{bmatrix}$$

a) Resolva pelo método de Eliminação de Gauss. Use o pivotamento parcial se necessário.
b) A partir da matriz triangular superior calcule o determinante da matriz **A**.
c) Inverta a matriz **A** pelo método de Eliminação de Gauss.

3.6 Seja o sistema $\mathbf{Ax} = \mathbf{b}$ na forma

$$\begin{bmatrix} -1 & 4 & -3 \\ -5 & -3 & 8 \\ -6 & 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ -3 \\ 3 \end{bmatrix}$$

a) A matriz **A** satisfaz o Critério das Linhas para alguma permutação de linhas?
b) A matriz **A** tem alguma permutação que satisfaz o Critério de Sassenfeld?
c) Escreva as equações de recorrência do método de Gauss-Seidel e calcule uma iteração a partir de $\mathbf{x}^{(0)} = (1, 0, 0)$.

Capítulo 4

Interpolação de Funções

4.1 Introdução

Suponhamos que o conjunto de pontos com duas coordenadas x e y, conhecidos por um processo qualquer

$$(x_0, y_0),\ (x_1, y_1),\ ,(x_2, y_2),\ \ldots,\ (x_n, y_n)\ ,$$

$$x_0 < x_1 < x_2 < \cdots < x_n\ .$$

O problema de interpolação é achar para um determinado valor $x \neq x_0,\ x_1,\ \ldots,\ x_n$ um valor razoável para y.

Isto é obtido determinando-se uma função $y = f(x)$ a partir das coordenadas conhecidas. Se o conjunto de coordenadas obtidas tem uma alta precisão, é razoável exigir que $y_i = f(x_i)$, $i = 0, 1, \ldots, n$. Caso contrário não é justificável esta exigência e podemos ter $y_i \neq f(x_i)$ que poderá até corrigir valores obtidos imprecisamente.

4.2 Funções utilizadas para interpolação

São inúmeros os tipos de funções utilizadas para interpolação e devemos ter um certo critério para escolher uma delas, levando em consideração o seu grau de suavidade no intervalo considerado e a sua simplicidade.

Forma geral pra $f(x)$

$$f(x) = a_0 f_0(x) + a_1 f_1(x) + \cdots + a_n f_n(x)\ ,$$

onde, de modo geral, $f_i(x)\,(i = 0, 1, \ldots, n)$ representa uma classe de funções. As classes mais usadas são:

(a) Monômios: $f_i(x) = x^i$

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = \sum_{i=0}^{n} a_i x^i$$

(b) Funções de Fourier: $f_i(x) = a_i \cos(ix) + b_i \sin(ix)$

$$\begin{aligned} f(x) = a_0 + & a_1 \cos(x) + a_2 \cos(2x) + \cdots + a_n \cos(nx) + \\ & b_1 \sin(x) + b_2 \sin(2x) + \cdots + b_n \sin(nx) \end{aligned}$$

$$f(x) = a_0 + \sum_{i=1}^{n} (a_i \cos(ix) + b_i \sin(ix))$$

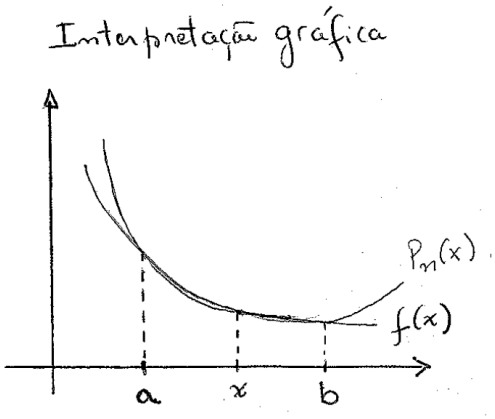
(c) Exponenciais: $f_i(x) = e^{b_i x}$

$$f(x) = a_0 e^{b_0 x} + a_1 e^{b_1 x} + \cdots + a_n e^{b_n x} = \sum_{i=0}^{n} a_i e^{b_i x}$$

Dentre essas, a mais usada é a dos Monômios (ou Polinômios) e citamos as seguintes vantagens:

- Sua teoria é simples e bem desenvolvida;
- São fáceis de serem calculados;
- As somas, produtos e diferenças são polinômios;
- Se $P_n(x)$ é um polinômio, $P_n(x+a)$ e $P_n(ax)$ também são;
- As outras classes de funções podem ser aproximadas por polinômios;
- Teorema de aproximação de Weierstrass:

Teorema. *Se $f(x)$ é contínua em $[a,b]$ então $\forall \epsilon > 0$, existe um $P_n(x)$ de grau n, $n = g(\epsilon)$, tal que $|f(x) - P_n(x)| < \epsilon$, $(a \leqslant x \leqslant b)$.*



4.3 Interpolação Polinomial

Polinômio: $P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$
Incógnitas: $a_0, a_1, a_2, \ldots, a_n$

(a) Polinômio Interpolante
Utilizado quando os dados são bastante precisos

$$P_n(x_i) = y_i, \quad i = 0, 1, \ldots, n$$

Teorema. *Existe um e só um polinômio de grau n ou menos que assume valores específicos para $n+1$ valores de x.*

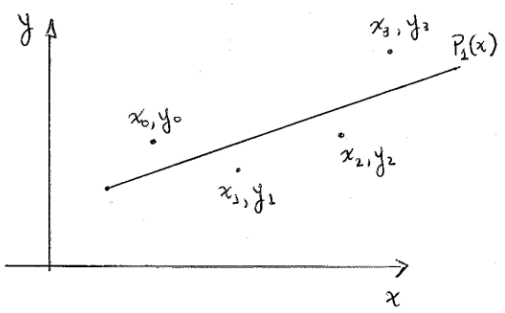
(b) Polinômio dos Mínimos Quadrados
Quando os y_i não são precisos, não se deve exigir que $P_n(x_i) = y_i$. Para simplificação, se $m+1$ é o número de pontos, calculamos um $P_n(x)$ tal que $m >> n$.
Critério dos Mínimos Quadrados: a soma dos quadrados das diferenças entre y_i e $P_n(x_i)$

deve ser mínima.
Estabelece-se um valor para n, sendo $(m+1)$ pontos tal que

$$E = \sum_{i=0}^{m} [P_n(x_i) - y_i]^2 \text{ é mínimo}$$

Se $m = n$, $E = 0$.

Exemplo: $n = 1$
$m = 3$



4.4 Cálculo de Polinômios

Seja $P_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$, onde são conhecidos $a_0, a_1, a_2, \ldots, a_n$. Dado x podemos calcular $P_n(x)$.

Algoritmo 4.1: Polinômio.

```
1  begin program
2      p ← 0
3      for i = 1 até n
4          p ← p + a_i × x^i
5      end for
6  end program
```

Teríamos $\frac{n(n+1)}{2}$ multiplicações e n adições (PF). A regra de Horner transforma $P_n(x)$ em

$$P_n(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)\ldots))$$

Algoritmo 4.2: Polinômio (Horner).

```
1  begin program
2      p ← a(n)
3      for i = n − 1 até 0
4          p ← p × x + a_i
5      end for
6  end program
```

4.5 Polinômio Interpolante

4.5.1 Sistema de equações para os coeficientes

Seja o sistema

$$\begin{aligned}
a_0 + a_1x_0 + a_2x_0^2 + \cdots + a_nx_0^n &= f(x_0)\\
a_0 + a_1x_1 + a_2x_1^2 + \cdots + a_nx_1^n &= f(x_1)\\
a_0 + a_1x_2 + a_2x_2^2 + \cdots + a_nx_2^n &= f(x_2)\ ,\\
\vdots \qquad\qquad\qquad\qquad\qquad &\quad \vdots\\
a_0 + a_1x_n + a_2x_n^2 + \cdots + a_nx_n^n &= f(x_n)
\end{aligned}$$

onde $a_0, a_1, a_2, \ldots, a_n$ são incógnitas.

A maneira de se determinar as incógnitas pode ser resolver por Gauss o sistema abaixo:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ 1 & x_2 & x_2^2 & \ldots & x_2^n \\ \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{4.1}$$

Mas a existência de x e x^n numa mesma linha tende a tornar o sistema desbalanceado.

Exemplo: Se $x_i = 0.1$ e $n = 10 \Rightarrow x_i^{10} = 10^{-10}$.

Existem outros métodos de se determinar o polinômio interpolante de modo mais simples que a resolução do sistema.

4.5.2 Polinômio de Lagrange

$$P_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i) \ \text{ onde } \ L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

$$P_n(x) = L_0(x)f(x_0) + L_1(x)f(x_1) + L_2(x)f(x_2) + \cdots + L_n(x)f(x_n)$$

$$\begin{aligned}
P_n(x) = {} & \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)} f(x_0)\\
& + \frac{(x-x_0)(x-x_2)\ldots(x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots(x_1-x_n)} f(x_1)\\
& \vdots\\
& + \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} f(x_i)\\
& \vdots\\
& + \frac{(x-x_0)(x-x_1)\ldots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\ldots(x_n-x_{n-1})} f(x_n),\ f(x_i) = y_i
\end{aligned}$$

Note que $L_k(x_i) = \delta_{ik} = \begin{cases} 1,\ i = k \\ 0,\ i \neq k \end{cases}$.

Para verificar que este é o polinômio interpolante basta substituir x_i

$$\begin{aligned}
P(x_i) &= L_0(x_i)\, y_0 + \cdots + L_{i-1}(x_i)\, y_{i-1} + L_i(x_i)\, y_i + L_{i+1}(x_i)\, y_{i+1} + \ldots\\
&= \cancel{0 \cdot y_0} + \cdots + \cancel{0 \cdot y_{i-1}} + 1 \cdot y_i + \cancel{0 \cdot y_{i+1}} + \cdots + \cancel{0 \cdot y_n}
\end{aligned}$$

Exemplo

Calcule o polinômio interpolante para os pontos $(-1,-11)$, $(1,3)$, $(2,7)$, $(3,17)$.

Solução:

$$\begin{aligned}P_3(x) = &\frac{(x-1)(x-2)(x-3)}{(-1-1)(-1-2)(-1-3)}(-11)\\ &+\frac{(x+1)\qquad\quad(x-2)(x-3)}{(1+1)\qquad\quad(1-2)(1-3)}(3)\\ &+\frac{(x+1)(x-1)\qquad\quad(x-3)}{(2+1)(2-1)\qquad\quad(2-3)}(7)\\ &+\frac{(x+1)(x-1)(x-2)}{(3+1)(3-1)(3-2)}(17)\end{aligned}$$

$$\begin{aligned}P_3(x) =&\frac{11}{24}(x^3-6x^2+11x-6)+\frac{18}{24}(x^3-4x^2+x+6)\\ &\frac{-56}{24}(x^3-3x^2-x+3)+\frac{51}{24}(x^3-2x^2-x+2)\end{aligned}$$

$$P_3(x) = x^3-3x^2+6x-1$$

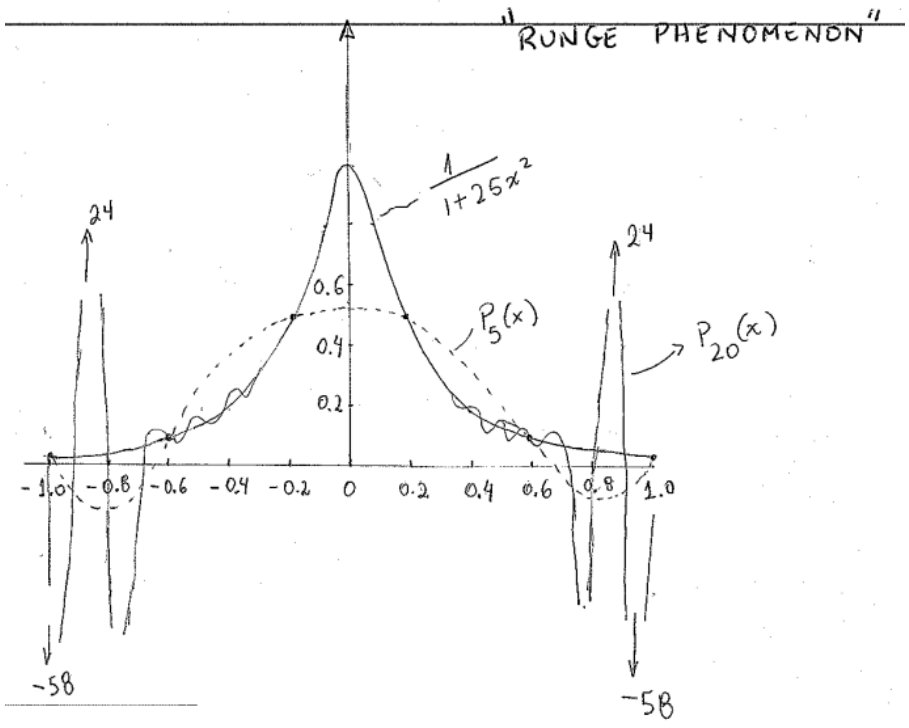
Observações

- O polinômio é o mesmo que aquele calculado por um sistema de equações pois ele é único.
- Na prática, as fórmulas de Lagrange não são usadas para determinar os coeficientes do polinômio interpolante, devido à complexidade dos cálculos e do algoritmo necessário.
- Estas fórmulas são usadas diretamente para interpolar $y = P_n(x)$.
- Com dois "loops", um para cálculo de $\prod$ e outro para o cálculo de $\sum$, obtém-se o valor de $y = P_n(x)$ interpolado.

4.5.3 Falhas do polinômio interpolante

Consideremos como função geradora dos pontos base a função de Runge $y = \dfrac{1}{1+25x^2}$.

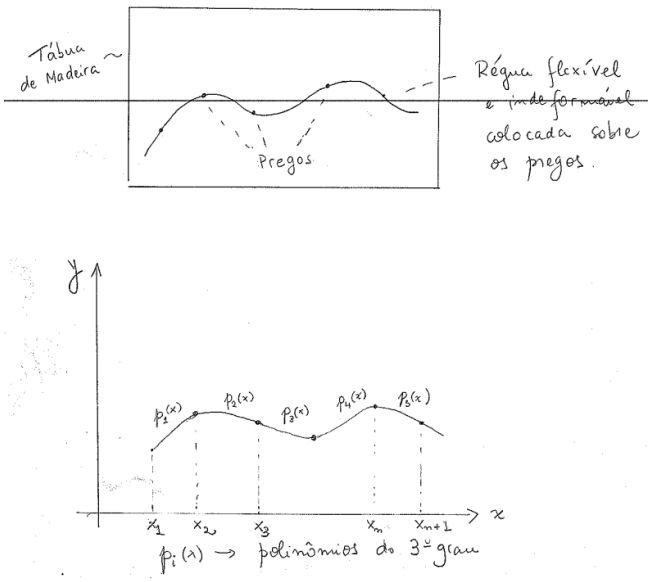
Vamos construir o gráfico da função de Runge e dos polinômios interpolantes $P_5(x)$ e $P_{20}(x)$ no intervalo $[-1,1]$:



Vemos que a divergência é maior quanto maior for o grau do polinômio. Temos uma curva não suave (com picos e vales). A solução é garantir a suavidade da curva, estabelecendo condições para as derivadas da função interpolante.

4.6 Interpolação por Spline

Este método de interpolação foi criado com a finalidade de garantir a suavidade da função interpolante. O nome "spline" vem do método utilizado por desenhistas para traçarem curvas suaves, especialmente para construção de navios.



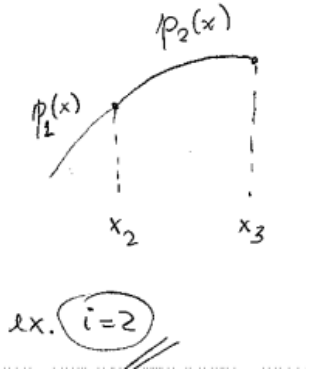
Consiste em atribuir um polinômio de 3º grau para cada par de pontos base consecutivos. Dessa forma, para $n+1$ pontos base teremos n polinômios.

Os polinômios do 3º grau são utilizados devido à uma teoria de deformações elásticas de barras flexíveis (sua forma se aproxima da forma de energia mínima armazenada).

Para que a variação do raio de curvatura seja contínua nos pontos base, em cada ponto a 2ª derivada do polinômio anterior é igual à do polinômio posterior. O mesmo se dá com a 1ª derivada.

Dessa maneira, temos:

# de equações	
$n-1$	$p_i''(x_i) = p_{i-1}''(x_i)$
$n-1$	$p_i'(x_i) = p_{i-1}'(x_i)$
n	$p_i(x_i) = y_i$
n	$p_i(x_{i+1}) = y_{i+1}$
$4n-2$ equações	



4.6.1 Dedução das fórmulas de "Spline"

Basicamente é necessário determinar 4 coeficientes para cada um dos n polinômios de 3º grau. São necessárias $4n$ equações. Os valores dos polinômios e suas derivadas nos $n+1$ pontos fornecem $4n-2$ equações pois não podemos calcular $p_1'(x_1)$ e $p_1''(x_1)$.

Para contornar esse problema, assumiremos que no início e no final da régua usada em spline, a inclinação é constante. Logo a 1ª derivada é constante e a 2ª derivada é nula

$$p_1''(x_1) = 0 \quad \text{e} \quad p_n''(x_{n+1}) = 0$$

e temos $4n$ equações.

Para facilitar a notação vamos definir

$$h_i \equiv x_{i+1} - x_i \ , \tag{4.2}$$
$$\phi_i \equiv p_{i-1}''(x_i) = p_i''(x_i) \ . \tag{4.3}$$

Sabemos que cada $p_i''(x)$ é um segmento de reta:

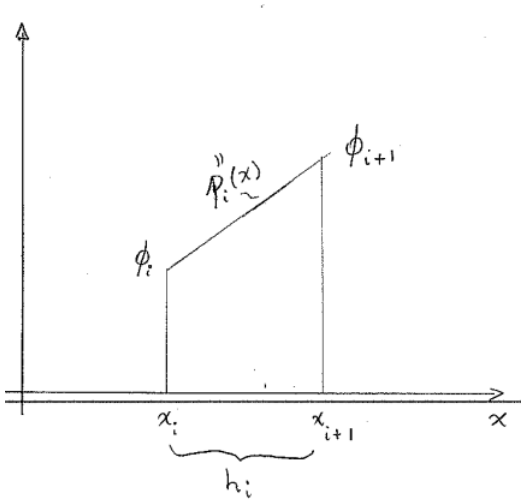


Figura 4.1: 2a. derivada de polinômio interpolante corresponde à segmento de reta

$$p_i''(x) = \phi_i + \frac{(\phi_{i+1} - \phi_i)(x - x_i)}{h_i} = \frac{\phi_i(x_{i+1} - x_i)}{h_i} + \frac{(\phi_{i+1} - \phi_i)(x - x_i)}{h_i} \ . \tag{4.4}$$

Rearranjando:

$$p_i''(x) = \frac{(x_{i+1} - x)}{h_i}\phi_i + \frac{(x - x_i)}{h_i}\phi_{i+1} \ . \tag{4.5}$$

Integrando $p_i''(x)$ duas vezes e impondo as condições

$$p_i(x_i) = y_i \quad \text{e} \quad p_i(x_{i+1}) = y_{i+1} \ , \tag{4.6}$$

teremos

$$p_i'(x) = -\frac{1}{2}\frac{(x_{i+1} - x)^2}{h_i}\phi_i + \frac{1}{2}\frac{(x - x_i)^2}{h_i}\phi_{i+1} + C_1 \ , \tag{4.7}$$

$$p_i(x) = \left(-\frac{1}{2}\right)\left(-\frac{1}{3}\right)\frac{(x_{i+1} - x)^3}{h_i}\phi_i + \left(\frac{1}{2}\right)\left(\frac{1}{3}\right)\frac{(x - x_i)^3}{h_i}\phi_{i+1} + \underbrace{C_1 x + C_2}_{C_1'(x_{i+1}-x)+C_2'(x-x_i)} \ , \tag{4.8}$$

Usando (4.6), temos

$$y_i = \frac{1}{6}\frac{(x_{i+1} - x_i)^3}{h_i}\phi_i + C_1'(x_{i+1} - x_i) \rightarrow C_1' = \frac{y_i}{h_i} - \frac{h_i\phi_i}{6} \ , \tag{4.9}$$

$$y_{i+1} = \frac{1}{6}\frac{(x_{i+1} - x_i)^3}{h_i}\phi_{i+1} + C_2'(x_{i+1} - x_i) \rightarrow C_2' = \frac{y_{i+1}}{h_i} - \frac{h_i\phi_{i+1}}{6} \ , \tag{4.10}$$

$$\begin{aligned} \Rightarrow p_i(x) = & \frac{\phi_i}{6h_i}(x_{i+1} - x)^3 + \frac{\phi_{i+1}}{6h_i}(x - x_i) \\ & + \left(\frac{y_i}{h_i} - \frac{h_i\phi_i}{6}\right)(x_{i+1} - x) + \left(\frac{y_{i+1}}{h_i} - \frac{h_i\phi_{i+1}}{6}\right)(x - x_i) \ , \end{aligned} \tag{4.11}$$

onde $\phi_1 \ldots \phi_n$ são incógnitas. Derivando $p_i(x)$ e $p_{i-1}(x)$ temos

$$p_i'(x_i) = p_{i-1}'(x_i) \ , \tag{4.12}$$

temos

$$\begin{aligned} p_i'(x_i) &= \frac{\phi_i}{6h_i}(-3)(x_{i+1}-x_i)^2 + \left(\frac{y_i}{h_i} - \frac{h_i\phi_i}{6}\right)(-1) + \left(\frac{y_{i+1}}{h_i} - \frac{h_i\phi_{i+1}}{6}\right) \\ &= -\frac{1}{2}h_i\phi_i - \left(\frac{y_i}{h_i} - \frac{h_i\phi_i}{6}\right) + \frac{y_{i+1}}{h_i} + \frac{h_i\phi_{i+1}}{6} \ , \end{aligned} \tag{4.13}$$

$$\begin{aligned} p_{i-1}'(x_i) =& \frac{\phi_{i-1}}{6h_{i-1}}(x_i - x)^3 + \frac{\phi_i}{6h_{i-1}}(x - x_{i-1})^3 \\ &+ \left(\frac{y_{i-1}}{h_{i-1}} - \frac{h_{i-1}\phi_{i-1}}{6}\right)(x_i - x) + \left(\frac{y_i}{h_{i-1}} - \frac{h_{i-1}\phi_i}{6}\right)(x - x_{i-1}) \ , \end{aligned} \tag{4.14}$$

$$p_{i-1}'(x_i) = \frac{1}{2}h_i\phi_i + \left(\frac{y_{i-1}}{h_{i-1}} - \frac{h_{i-1}\phi_{i-1}}{6}\right)(-1) + \frac{y_{i-1}}{h_{i-1}} + \frac{h_{i-1}\phi_i}{6} \tag{4.15}$$

Usanod a condição $p_i'(x_i) = p_{i-1}'(x_i)$, igualamos (4.14) e (4.15), multiplicando tudo por (-6) e reordenando os termos chegamos a

$$h_{i-1}\phi_{i-1} + 2\left(h_{i-1} + h_i\right)\phi_i + h_i\phi_{i+1} = 6\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right) \tag{4.16}$$

, que são $n-1$ equações para $n+1$ incógnitas, mas que fica determinado assumindo $\phi_1 = 0$ e $\phi_{n+1} = 0$. O sistema pode ser colocando na forma de matriz tridiagonal

$$\begin{bmatrix} 2(h_1+h_2) & h_2 & & & & & \\ h_2 & 2(h_2+h_3) & h_3 & & & & \\ & h_3 & 2(h_3+h_4) & & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} & \\ & & & & h_{n-1} & 2(h_{n-1}+h_n) \end{bmatrix} \begin{bmatrix} \phi_2 \\ \phi_3 \\ \phi_4 \\ \vdots \\ \phi_{n-1} \\ \phi_n \end{bmatrix} = \begin{bmatrix} e_2 - e_1 \\ e_3 - e_2 \\ e_4 - e_3 \\ \vdots \\ e_{n-1} - e_{n-2} \\ c_n - e_{n-1} \end{bmatrix} \tag{4.17}$$

onde $e_i = 6\left(\frac{y_{i+1}-y_i}{h_i}\right)$. O sistema tridiagonal pode ser resolvido por uma rotina ou numa implementação direta usando a decomposição LU vista no capítulo 3 como a seguir

$$u_2 = 2(h_1 + h_2) \tag{4.18}$$

$$u_i = 2(h_{i-1} + h_i) - \frac{h_i^2}{u_{i-1}} \qquad (i = 3, \ldots, n) \tag{4.19}$$

$$z_2 = e_2 - e_1 \tag{4.20}$$

$$z_i = 2(e_i + e_{i-1}) - \frac{h_{i-1}}{u_{i-1}}z_{i-1} \qquad (i = 3, \ldots, n) \tag{4.21}$$

Os ϕ_i's sào calculados por

$$\phi_n = z_n/u_n \tag{4.22}$$

$$\phi_i = (z_i - h_i\phi_{i+1})/u_i \qquad (i = n-1, \ldots, 2) \tag{4.23}$$

e os polinômios são finalmente obtidos

$$p_i(x) = y_i + \alpha_i(x - x_i) = \beta_i(x - x_i)^2 + \gamma_i(x - x_i)^3 \tag{4.24}$$

onde

$$\alpha_i = p_i'(x_i) = \frac{y_{i+1} - y_i}{h_i} - \frac{\phi_{i+1}h_i}{6} - \frac{\phi h_i}{3} \tag{4.25}$$

$$\beta_i = \frac{p_i''(x_i)}{2} = \frac{\phi_i}{2} \tag{4.26}$$

$$\gamma_i = \frac{p_i'''(x_i)}{6} = \frac{\phi_{i+1} - \phi i}{6h_i} \qquad (i = 2, \ldots, n-1), \tag{4.27}$$

sendo que na última equação usamos que $p_i'''(x_i)$ corresponde à inclinação da reta da fig.4.1 naquele ponto.

Exercícios

4.1 a) Escreva a expressão do polinômio interpolante $y = P_3(x)$ usando o método de Lagrange para os pontos $(-1, -11), (1, 5), (2, 13), (3, -13)$.

b) Construa um programa em C (ou C++, FORTRAN) que calcule o valor de y dado $x = 1.5$ usando o polinômio interpolante do item a). Bastam dois laços, um interno para o produtório e outro externo para o somatório.

Capítulo 5

Integração Numérica

5.1 Introdução

A integração também é chamada de Quadratura. O objetivo aqui é calcular a integral

$$I = \int_a^b f(x)dx \tag{5.1}$$

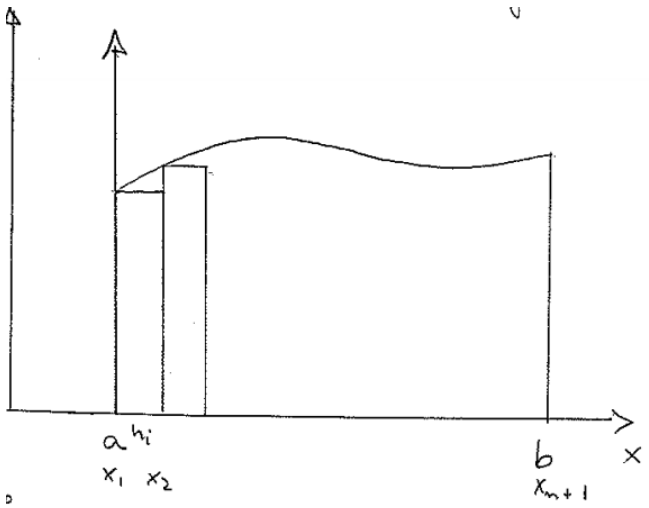
numericamente. Isto pode ser necessário se a integral for muito complicada ou impossível de ser feita analiticamente, como por exemplo $\int_0^1 e^{-x^2}dx$. Há vários métodos de integração numérica dos quais estudaremos: Integração por Retângulos, Regra do Ponto Médio, Regra do Trapézio, Regra de Simpson, Regra de Spline, Romberg, Quadraturas adaptativas, Quadratura de Gauss e Integração por Monte-Carlo.

5.2 Integração por Retângulos

É o método mais simples e de mais fácil implementação, porém é pouco acurado. Seja o intervalo $[a, b]$ no eixo x. A integral sob a curva delimitada pela função $f(x)$ é dada por

$$I = \int_a^b f(x)dx \sim \sum_{i=1}^{n} h_i f(x_i) = \sum_{i=1}^{n} R_i \tag{5.2}$$

onde $x_1 = a, x_{n+1} = b, h_i \equiv x_{i+1} - x_i$ e $R_i \equiv h_i f(x_i)$ é a área do i-ésimo retângulo.



Expandindo $f(x)$ em série de Taylor em torno do ponto x_i temos

$$f(x) = f(x_i) + (x - x_i)f'(x_i) + \dots \tag{5.3}$$

e a área do i-ésimo intervalo será

$$\begin{aligned} I_i = \int_{x_i}^{x_{i+1}} f(x)dx &= \int_{x_i}^{x_{i+1}} f(x_i)dx + \int_{x_i}^{x_{i+1}} (x - x_i)f'(x_i) + \dots \\ &= f(x_i)x\Big|_{x_i}^{x_{i+1}} + \frac{1}{2}(x - x_i)^2 f'(x_i)\Big|_{x_i}^{x_{i+1}} + \dots \\ &= f(x_i)h_i + \frac{1}{2}h_i^2 f'(x_i) + \dots \end{aligned} \tag{5.4}$$

Assim o erro na integral em um intervalo é da ordem de $\mathcal{O}(h^2)$, ou $I_i - R_i = \mathcal{O}(h^2)$. Se considerarmos a soma de N intervalos, $N = (b-a)/h$, o erro total será a soma de dos erros de todos intervalos, i.e., $\sim Nh^2$, ou seja, o erro total é $\mathcal{O}(h)$.

5.3 Regra do Ponto Médio ("Midpoint Rule")

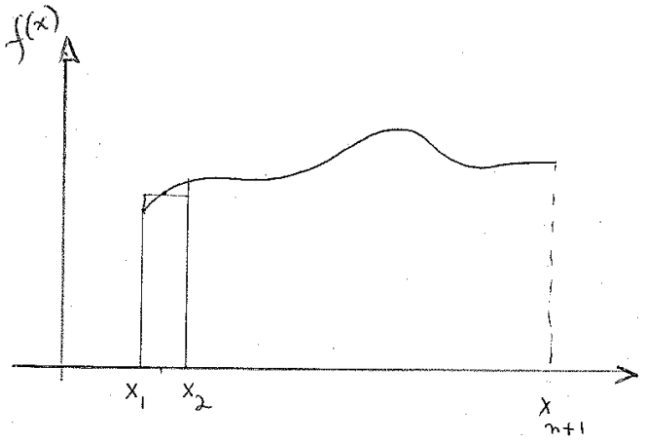
Também se utiliza de retângulos para uma aproximação da integral mas usa o ponto médio de cada intervalo considerado.

Pela regra do ponto médio, o valor da integral em um subintervalo pode ser aproximado na forma

$$I_i = \int_{x_i}^{x_{i+1}} f(x)dx \sim h_i f(\bar{x}_i) = M_i, \quad \text{onde} \quad \bar{x}_i = \frac{x_i + x_{i+1}}{2} \tag{5.5}$$

Assim, o valor da total da integral será

$$I = \int_{x_1}^{x_{n+1}} f(x)dx \sim \sum_{i=1}^{n} h_i f(\bar{x}_i) = M \tag{5.6}$$



Para avaliarmos o erro, vamos expandir em série de Taylor em torno de $\bar{x}_i$, obtendo

$$f(x) = f(\bar{x}_i) + (x - \bar{x}_i)f'(\bar{x}_i) + \frac{1}{2}(x - \bar{x}_i)^2 f''(\bar{x}_i) + \frac{1}{3!}(x - \bar{x}_i)^3 + \dots \tag{5.7}$$

Substituindo (5.7) em (5.6) temos

$$I_i = \int_{x_i}^{x_{i+1}} f(x)dx = h_i f(\bar{x}_i) + \frac{1}{24}h_i^3 f''(\bar{x}_i) + \dots \tag{5.8}$$

então o erro obtido será

$$I_i - M_i = \frac{1}{24} h_i^3 f''(\bar{x}_i) + \dots \tag{5.9}$$

e portanto, após somar N subintervalos, o erro total será $\mathcal{O}(h^2)$.

5.4 Regra do Trapézio

Os pontos são unidos por retas formando trapézios conforme a figura fig.5.1.

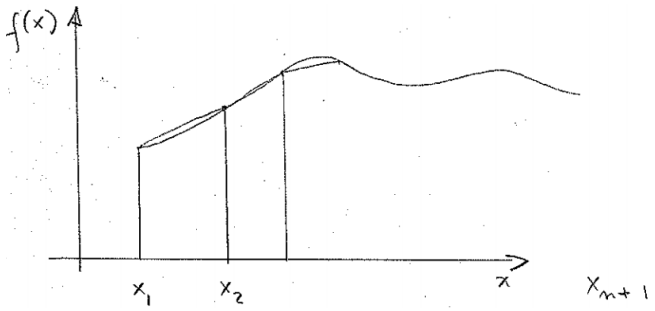


Figura 5.1

A área será dada por

$$I_i \sim \frac{\text{base+base}}{2} \times \text{altura} = h_i \frac{f(x_i) + f(x_{i+1})}{2} = h_i T_i \tag{5.10}$$

$$I \sim T = \sum_{i=1}^{n} h_i T_i = h_i \frac{f(x_i) + f(x_{i+1})}{2} \tag{5.11}$$

$$T = \frac{h}{2}[f_1 + 2f_2 + 2f_3 + \cdots 2f_n + f_{n+1}] = \frac{h}{2}[f_1 + f_{n+1} + 2\sum_{i=2}^{n} f_n] \tag{5.12}$$

onde $f_i \equiv f(x_i)$.

Cálculo do erro na Regra do Trapézio:

Pelo teorema Fundamental do Cálculo, existe uma função F tal que $F'(x) = f(x)$. Para pontos equidistantes, o erro na Regra do Trapézio será dado por

$$I = T_i = \int_{x_i}^{x_{i+1}} F'(x)dx - \frac{h}{2}\{f(x_i) + f(x_{i+1})\} \tag{5.13}$$

$$= F(x_i + h) - F(x_i) - \frac{h}{2}\{f(x_i) + f(x_i + h)\} \tag{5.14}$$

Expandindo o primeiro e o último termo em série de Taylor

$$I = \quad F(x_i) + hf(x_i) + \frac{h^2}{2} f'(x_i) + \frac{h^3}{6} f''(x_i) + \cdots \tag{5.15}$$

$$- F(x_i) \tag{5.16}$$

$$- \frac{h}{2} \tag{5.17}$$

$$- \frac{h}{2} f(x_i) - \frac{h^2}{2} f'(x_i) - \frac{h^3}{4} f'''(x_i) + \cdots \tag{5.18}$$

$$= -\frac{h^3}{12} f''(x_i) + \mathcal{O}(h^4) \tag{5.19}$$

que é o erro em um subintervalo. Se somarmos n subintervalos onde $n = \dfrac{b-a}{h}$, teremos um erro total $\mathcal{O}(h^2)$.

Exercício: Estime $\int_0^1 e^{-x^2} dx$ usando a Regra do Trapézio tomando $n = 2, 4, 6, 8, 16$. (O valor da integral é $I = 0.746824133$ com precisão até 9 decimais)

Solução:

$n = 2, h = 0.25,$

$$T = \frac{0.5}{2}[f(0) + 2f(0.5) + f(1)] = 0.73137021$$

$n = 4, h = 0.25,$

$$T = \frac{0.25}{2}[f(0) + 2f(0.25) + 2f(0.5) + 2f(0.75) + f(1.0)] = 0.7429840$$

$n = 8, h = 0.125,$

$$T = \frac{0.125}{2}[f(0) + 2f(0.125) + \cdots + 2f(0.875) + f(1.0)] = 0.7458655$$

$n = 16, h = 0.0625,$

$$T = 0.7465846$$

5.5 Regra de Simpson

$$S = \sum_{j=1}^{n/2} S_j = \sum_{j=1}^{n/2} \int_{x_i}^{x_{i+2}} p_j(x) dx \tag{5.20}$$

onde $i = 2j - 1$, $p_j(x)$ é um polinômio do 2º grau passando pelos pontos (x_i, f_i), (x_{i+1}, f_{i+1}) e (x_{i+2}, f_{i+2}), $f_i = f(x_i)$.

Como não estamos interessados no polinômio em si mas na sua integral definida, podemos mover a área em questão para a origem conforme a figura. Agora teremos que encontrar a área sob um polinômio do 2o grau que passam pelos pontos $(-h, f_i), (0, f_{i+1})$ e (h, f_{i+2}) e podemos usar a interpolação de Lagrange vista no capítulo anterior, como segue. O polinômio interpolante de Lagrange de 2o grau é dado por

$$\begin{aligned} p(x) = & \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} f_1 \\ & + \frac{(x - x_1)\qquad (x - x_3)}{(x_2 - x_1)\qquad (x_2 - x_3)} f_2 \\ & + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} f_3 \end{aligned} \tag{5.21}$$

Substituindo os pontos temos

$$\begin{aligned} p(x) = & \frac{(x - 0)(x - h)}{(-h - 0)(-h - h)} f_i \\ & + \frac{(x - (-h))\qquad (x - h)}{(0 - (-h))\qquad (0 - h)} f_{i+1} \\ & + \frac{(x - (-h))(x - 0)}{(h - (-h))(h - 0)} f_{i+2} \end{aligned} \tag{5.22}$$

ou

$$p(x) = \frac{1}{2h^2}\{x(x-h)f_i - 2(x+h)(x-h)f_{i+1} + x(x+h)f_{i+2}\} \tag{5.23}$$

Integrando ambos os membros de $-h$ até $+h$

$$\int_{-h}^{+h} p(x) = \frac{1}{2h^2}\int_{-h}^{+h}\{x(x-h)f_i - 2(x+h)(x-h)f_{i+1} + x(x+h)f_{i+2}\}dx \tag{5.24}$$

As integrais de cada um dos termos valem

$$\int_{-h}^{+h} x(x-h)dx = \frac{2}{3}h^3, \quad \int_{-h}^{+h} x(x-h)dx = -\frac{4}{3}h^3, \quad \int_{-h}^{+h} x(x+h)dx = \frac{2}{3}h^3 \tag{5.25}$$

Substituindo os resultados (5.25) em (5.24) temos

$$\int_{-h}^{+h} p(x) = \frac{h}{3}(f_i + 4f_{i+1} + f_{i+2}), \quad f_i \equiv f(x_i). \tag{5.26}$$

Somando a integral de todos os polinômios temos finalmente

$$\begin{aligned} S &= \frac{h}{3}[(f_1 + 4f_2 + f_3) + (f_3 + 4f_4 + f_5) + \cdots + (f_{n-1} + 4f_n + f_{n+1})] \\ &= \frac{h}{3}[(f_1 + 4f_2 + 2f_3 + 4f_4 + 2f_5 + 4f_6 + \cdots + 2f_{n-1} + 4f_n + f_{n+1}] \\ &= \frac{h}{3}[f_1 + f_{n+1} + 4\sum_{\substack{i=2 \\ \text{pares}}}^{n} f_i + 2\sum_{\substack{i=3 \\ \text{ímpares}}}^{n-1} f_i] \ . \end{aligned} \tag{5.27}$$

5.6 Considerações finais sobre projetos de pesquisa

- Ao usar "pacotes"sempre devemos desconfiar deles. Devemos lembrar que muitas vezes eles não foram suficientemente testados especialmente em problemas novos, objeto da pesquisa. Os pacotes também em geral não permitem um controle mais apurado dos parâmetros de convergência, nem a impressão valores intermediários. Na impossibilidade de construir o próprio código deve-se se usar outros pacotes que realizam o mesmo cálculo, de preferência com algoritmos diferentes, para confrontar os resultados.

- Muitas vezes acredita-se que devemos usar o método mais acurado para o nosso problema. Isso nem sempre é produtivo e pode levar a atrasos no projeto. Por exemplo, para resolver uma simples EDO basta o método de Euler que pode ser implementado quase instantaneamente. Já o RK é muito mais acurado mas pode-se cometer diversos erros de digitação e tem uma construção mais demorada. Devemos fazer avaliações prévias não só de quanto tempo o programa demorará para fornecer o resultado mas também do tempo de implementação. Em muitos casos obtemos resultados mais rápidos e corretos com uma implementação simples aguardando um tempo maior de execução do programa.

- Projetos complexos devem ser divididos em pequenos subprojetos cujas rotinas devem ser testadas em separado. Frequentemente o pesquisador inexperiente tenta rapidamente construir todo o programa sem um pragmático teste de partes do programa, com impressão de tabelas de resultados intermediários. Deve-se construir um programa preliminar que em geral temos soluções analíticas ou respostas conhecidas numéricas. Só então poderemos inserir novos ingredientes ao novo modelo em questão.

- Em geral o problema que estamos resolvendo não tem solução analítica. Como validar o cálculo? Há várias técnicas, entre elas, a verificação leis de conservação do modelo e situações limites, onde parâmetros podem se tornar muito grandes ou muito pequenos.

Referências Bibliográficas

F. S. Acton. *Numerical Methods that Work*. Harper & Row, 1970.

C. H. Asano e E. Colli. *Cálculo Numérico e Aplicações*. `http://www.ime.usp.br/~asano/LivroNumerico/LivroNumerico.pdf`, 2009.

K. E. Atkinson. *An Introduction to Numerical Analysis, 2nd ed.* John Wiley & Sons, 1989.

E. K. Blum. *Numerical Analysis and Computation: Theory and Practice*. Addison Wesley, 1972.

C. G. Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Math. Comput.*, 19(92):577–593, 1965. doi: 10.1090/s0025-5718-1965-0198670-6.

R. L. Burden e J. D. Faires. *Análise Numérica*. Thomson, 2003.

B. Carnahan, H. A. Luther, and J. O. Wilkes. *Applied Numerical Methods*. John Wiley & Sons, New York, 1969.

F. C. Mokarzel. *Cálculo Numérico-Apostila de Notas de Aulas, CMP-20/83*. ITA, São José dos Campos, 1983.

Tao Pang. *An Introduction to Computational Physics, 2nd ed.* Cambridge, New York, 1989.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in FORTRAN (or C, C++)*. Cambridge Univ. Press, Cambridge, 1992.

D. Quinney. *An Introduction to the Numerical Solution of Differential Equations*. Research Studies Press and John Wiley & Sons, revised ed., 1987.

R. D. Skeel and J.B. Keiper. *Elementary Numerical Computing with Mathematica*. MacGraw Hill, 1993.

P. A. Stark. *Introduction to Numerical Methods*. MacMillan, 1970.

H. A. Van der Vorst. BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 13:631–644, 1992.

J. H. Wilkinson. *Rounding errors in algebraic processes*. Englewood Cliffs-Prentice Hall, 1963.