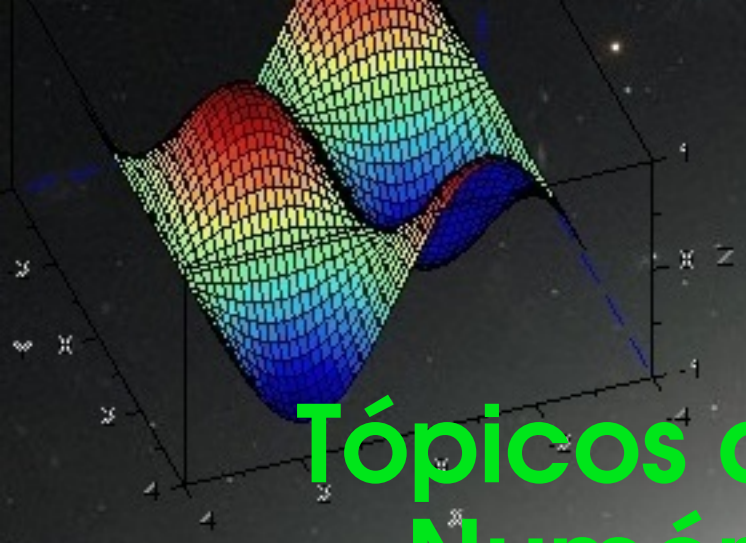
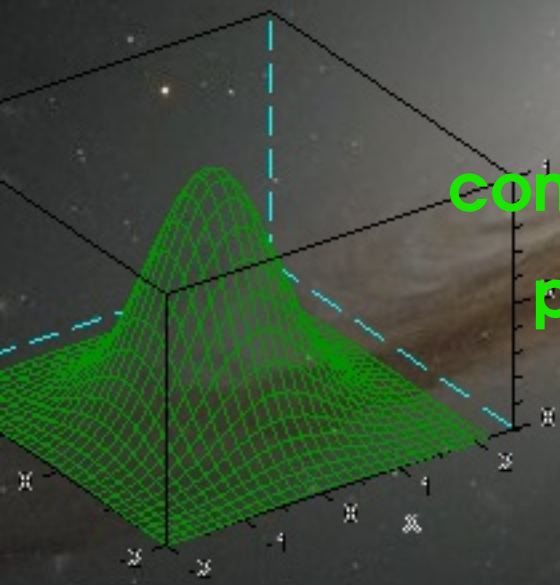
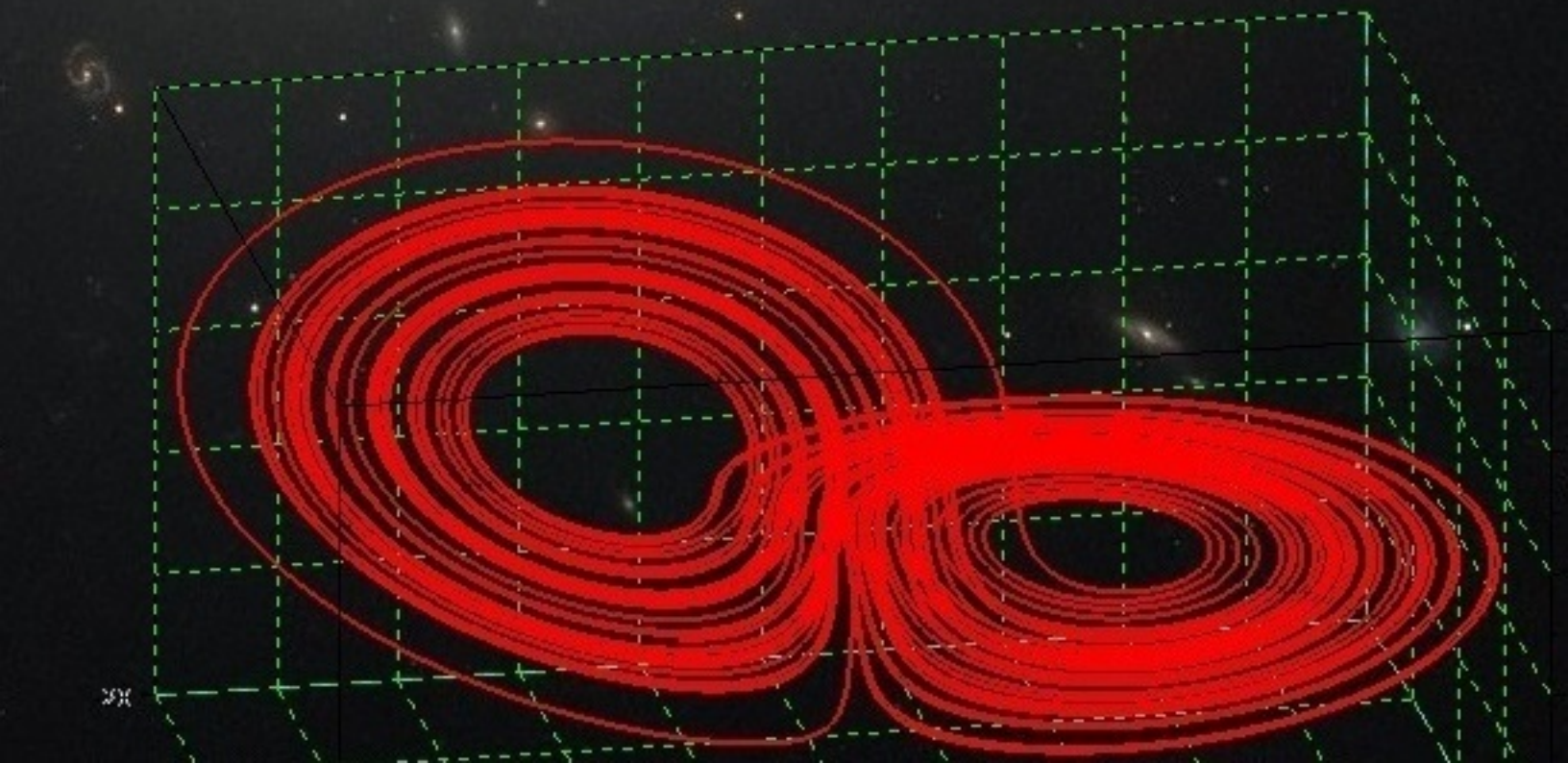


Tópicos de Métodos Numéricos com Scilab:

computação científica para engenheiros

Francisco J. A. de Aquino,
prof. IFCE - Campus Fortaleza





Tópicos de métodos numéricos com Scilab : computação científica para engenheiros

Impressão e Acabamento:
Pod Editora
Rua Imperatriz Leopoldina, 8/1110 – Pça Tiradentes
Centro – 20060-030 – Rio de Janeiro
Tel. 21 2236-0844 • atendimento@podeditora.com.br
www.podeditora.com.br

Revisão:
Letícia Rio Branco

Diagramação no software LATEX:
Francisco José Alves de Aquino

CIP-BRASIL. CATALOGAÇÃO-NA-FONTE
SINDICATO NACIONAL DOS EDITORES DE LIVROS, RJ

A669t

Aquino, Francisco José Alves de

Tópicos de métodos numéricos com scilab : computação científica para engenheiros / Francisco José Alves de Aquino. - 1. ed. - Rio de Janeiro : PoD, 2020.

232 p. : il. ; 30 cm.

Inclui bibliografia e índice

ISBN 978- 65-86147-82-7

1. Scilab (Programa de computador). 2. Cálculos numéricos - Programas de computador. I. Título.

20- 67881 CDD: 005.3
CDU: 004.42 07/12/2020

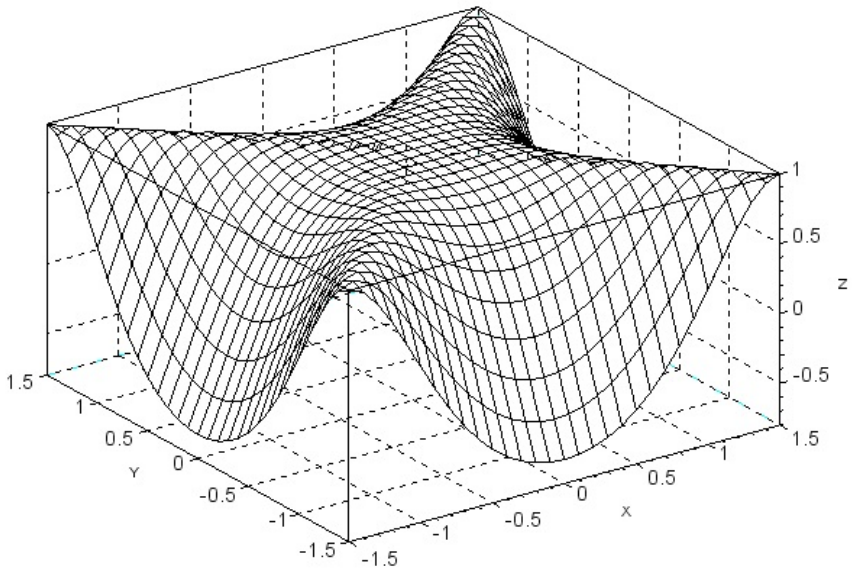
Leandra Felix da Cruz Candido – Bibliotecária – CRB-7/6135

Dedicatória e agradecimento

Dedico este trabalho a minha mãe e a minha esposa. São duas mulheres guerreiras que muito me ajudaram para o sucesso deste empreendimento, embora compreendam bem pouco de matemática, ou métodos numéricos.

_____

Agradeço aos meus professores pela minha formação, especialmente os da área de matemática, física e engenharia. Agradeço também o apoio recebido dos meus colegas do IFCE, em especial ao professor Glauber Cintra por ter lido parte do manuscrito. Sou grato também ao IFCE pelo apoio institucional e material que tornou possível a materialização deste livro. Não posso esquecer de agradecer aos diversos alunos que me ensinaram repetidas vezes que nem tudo o que é fácil para o professor é assim tão compreensível para eles.



Os nossos conhecimentos são a reunião do raciocínio e experiência de numerosas mentes.
Ralph Waldo Emerson (1803 - 1882), escritor, poeta, filósofo, ensaísta norte-americano.

Apresentação

Podemos definir a "Matemática" (do grego $\mu\alpha\theta\eta\mu\alpha$, máthema: ciência, conhecimento, aprendizagem, $\mu\alpha\theta\eta\mu\alpha\tau\iota\kappa\varsigma$, mathematikós: amante do conhecimento) como sendo o estudo de padrões nas estruturas de entidades abstratas e nas relações entre eles. Por essa definição, este não é exatamente um livro de matemática, mas um livro que usa conceitos matemáticos em aplicações e problemas práticos da ciência e engenharia, embora a gama de aplicações seja muito maior.

Este livro é resultado do meu interesse por métodos numéricos em geral e do uso de *software* livre como ferramenta para resolução de problemas dessa área. Resolvemos escolher um *software* e uma linguagem específica - o Scilab - que atendesse de forma mais completa possível as exigências dessa disciplina. Além disso, o Scilab ainda não é tão amplamente difundido entre os estudantes brasileiros. Tentamos cobrir, pelo menos parcialmente, esta lacuna.

O foco que apresentamos neste livro é de cunho mais prático e predominantemente voltado para engenheiros e cientistas que usam os métodos numéricos como uma ferramenta para resolução de problemas. Assim, evitamos uma abordagem mais detalhada de teoremas e as demonstrações matemáticas são resumidas ao mínimo necessário. Nas referências, para o leitor mais interessado nesses detalhes, temos vários autores que abordam esses tópicos com mais profundidade. Por outro lado, esperamos que o estudante que use esse livro para estudar já tenha concluído pelo menos o primeiro ano da faculdade e tenha conhecimento de Cálculo I e II, noções de programação e algoritmo e algum conhecimento de estatística.

Este Livro tem como capítulo inicial uma introdução ao Scilab. Não podemos esperar que todos os recursos existentes no Scilab sejam abordados, mas apresentamos o suficiente para que o leitor seja capaz de aprender sozinho como encontrar os recursos que sejam necessários posteriormente. Como exigência, o leitor já deve trazer algum conhecimento de lógica de programação em qualquer linguagem e noções de algoritmos e, claro, os fundamentos de cálculo diferencial e integral.

Nos capítulos 2 e 3, apresentamos noções de modelos, sistemas e de erros numéricos. Para alunos de cursos de engenharia que já fizeram a disciplina de sistemas lineares ou equivalente, esse segundo capítulo talvez possa ser visto como opcional. A representação numérica nos computadores e alguns tipos de erros estão descritos no terceiro capítulo.

No Capítulo 4, abordamos o problema de encontrar as raízes de uma função matemática. Apresentamos métodos diversos, entre eles o método da iteração linear, da bisseção, Newton-Raphson, entre outros. Para funções polinomiais, apresentamos uma função do Scilab, muito útil para o cálculo de suas raízes.

Já no Capítulo 5, discutimos os sistemas lineares e vários conceitos relacionados. Mostramos formas de resolver esses sistemas e concluímos o capítulo focando rapidamente em sistemas não-lineares.

Em seguida, estudamos os problemas de regressão e interpolação em dois capítulos independentes, mas profundamente relacionados. Sempre incluímos os recursos disponíveis no Scilab para solução desses problemas.

No Capítulo 8, apresentamos o problema da integração numérica usando fórmulas clássicas, tanto "fechadas" quanto "abertas". Estudamos também alguns problemas de integração imprópria, integração múltipla e o conceito de convolução.

Vemos alguns conceitos sobre derivação numérica no curto Capítulo 9. Já o Capítulo 10 traz o importante problema de solução numérica de equações diferenciais ordinárias. Novamente, apresentamos alguns dos recursos disponíveis no Scilab sobre esse tema.

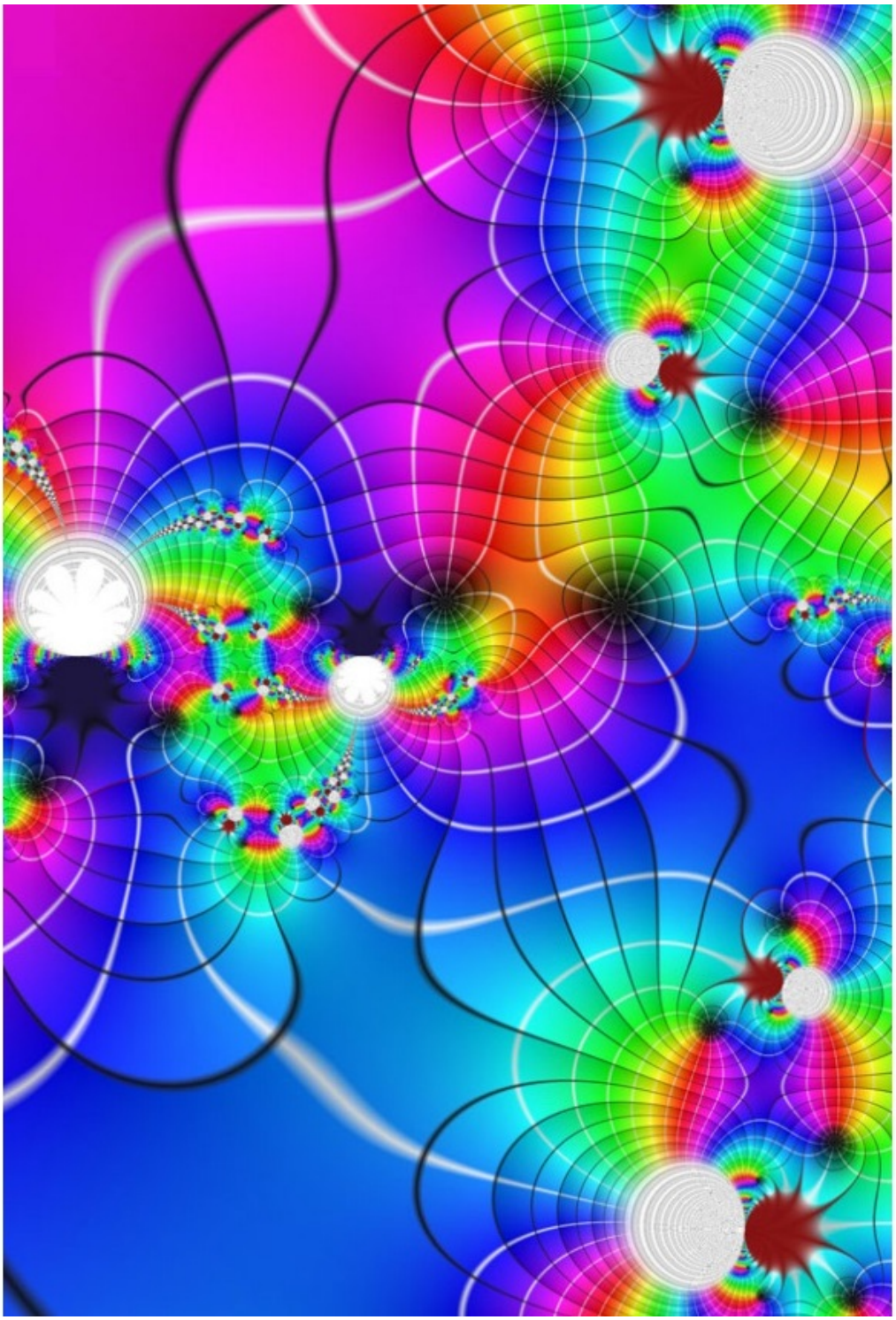
No Capítulo 11, incluímos algumas ideias sobre o conceito de números (pseudo) aleatórios e como gerá-los. Apresentamos o método de Monte Carlo e um exemplo para ilustrá-lo. Esse pode ser um tópico importante para quem trabalha com simulação computacional.

Em um curso de apenas um semestre, faltando tempo para cobrir todos esses capítulos, sugerimos que os capítulos 7, 9 e 11 não sejam incluídos. Por outro lado, a solução dos problemas ao final de cada capítulo é essencial ao aprendizado do conteúdo abordado. Os capítulos são mais ou menos independentes e a ordem em que serão abordados em sala de aula pode variar de acordo com o programa da disciplina ou o sentimento do professor e as contingências que sempre surgem ao longo dos semestres letivos. Não é obrigatório que todas as seções de um capítulo sejam discutidas.

Finalmente, optamos por usar uma linguagem não completamente formal. Usamos um tom um pouco mais coloquial em alguns momentos. Uma advertência: as casas decimais serão indicadas algumas vezes por vírgula (ex: "10,32" - quando estamos no texto), outras vezes por ponto (ex: "10.32" - quando estamos mostrando um código Scilab). Naturalmente, os problemas deste livro, em sua grande maioria, só podem ser resolvidos usando uma calculadora científica, ou um computador com o Scilab, ou algum *software* similar.

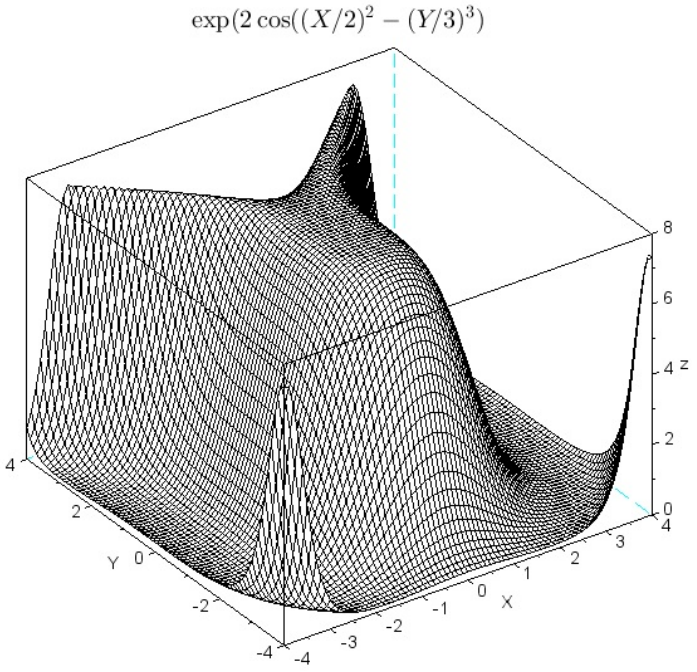
3,1415926535 8979323846 2643383279 5028841971 6939937510
5820974944 5923078164 0628620899 8628034825
3421170679 8214808651 3282306647
0938446095 5058223172
5359408128

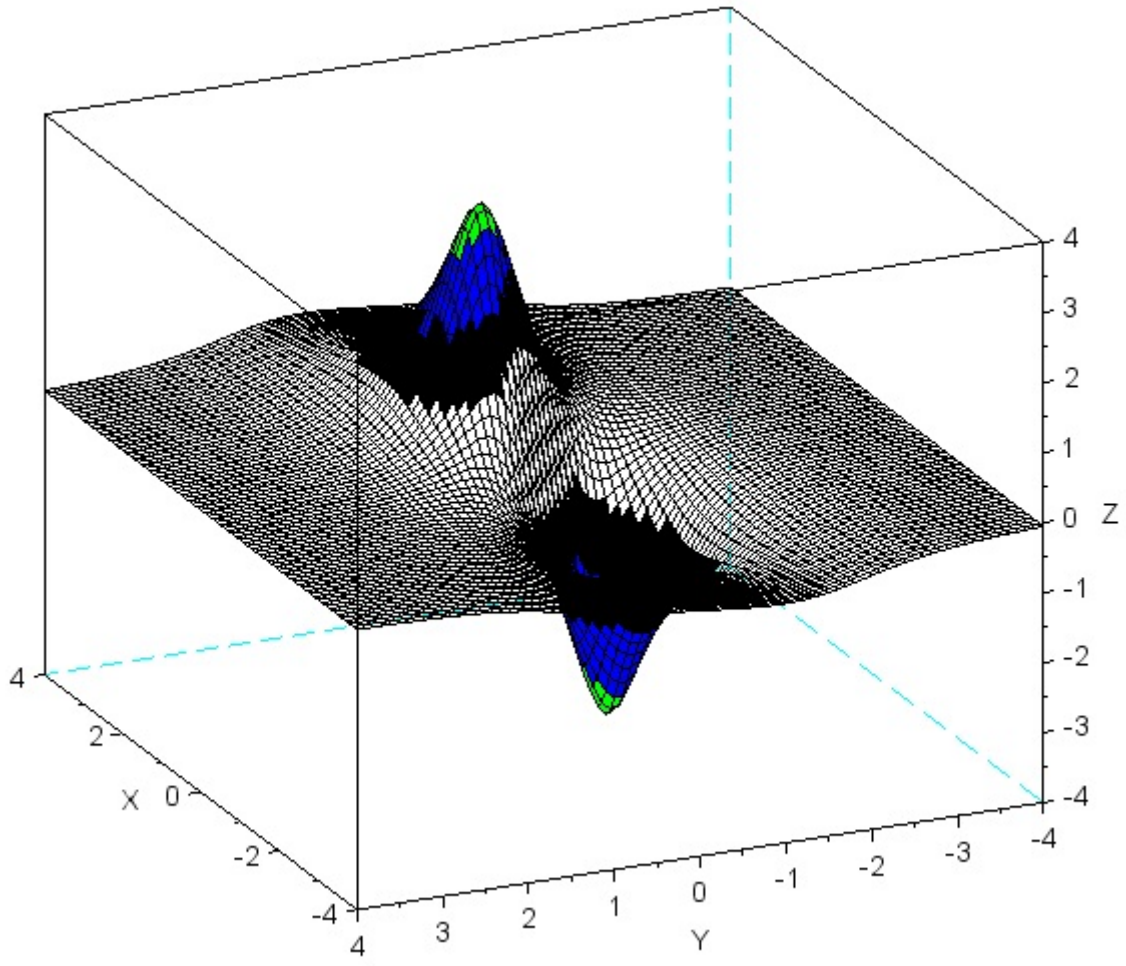
IFCE, Campus Fortaleza, dezembro de 2020.
Francisco José Alves de Aquino (*fcoalves_aq@ifce.edu.br*).



A matemática pode ser incrivelmente bela.

Sumário




4
3
2
1
0
-1
-2
-3
-4
Z
4
2
X 0
-2
-4
4
3
2
1
0
-1
-2
-3
-4
Y

1. Programação Scilab

A ciência é o que nós compreendemos suficientemente bem para explicar a um computador.
Donald Knuth (1938 - *) cientista da computação e professor emérito da Universidade de Stanford.

ESCOLHEMOS o *software* Scilab[1] (`http://www.scilab.org`) como ferramenta para demonstrar na prática os conceitos apresentados neste livro. O Scilab é um *software* livre que possui grande versatilidade para programação numérica, computação científica e geração de gráficos de forma muito semelhante ao Matlab. A linguagem Scilab também tem alguma semelhança com a linguagem *C*, mas não é necessário ter uma área para declaração de variáveis, por exemplo. O *script* Scilab, da mesma forma que o Matlab, é interpretado[2]. O Scilab ocupa muito menos espaço em disco que o Matlab.

Neste capítulo, apresentamos os conceitos básicos e os comandos necessários à implementação dos algoritmos numéricos. Para computação simbólica é melhor usar outros *softwares* como Maple, MathCad, Mathematica, wxMaxima e outros. Um ajuda *on-line* com todos os comandos Scilab pode ser visto no seguinte endereço: `https://help.scilab.org/docs/6.0.0/pt_BR/index.html`. Consideramos que o leitor tenha um domínio pelo menos elementar de algoritmos e o conhecimento em alguma linguagem de programação.

O Scilab *software* para computação numérica. Os códigos escritos no ambiente Scila são interpretados. Isso geralmente permite obter processos de desenvolvimento mais rápidos, pois o usuário acessa linguagem de alto nível, com um rico conjunto de recursos fornecidos pela biblioteca. Os usuários do Scilab podem desenvolver seus próprios módulos para que eles possam resolver seus problemas particulares. A linguagem Scilab permite dinamicamente compilar e ligar outras

[1] A pronúncia em sintaxe fonética internacional é “sailæb”.

[2] Aqui cabe uma observação: *todas* as linguagens de computador são *interpretadas* em algum nível. Por exemplo, um compilador *C* gera um arquivo executável “.exe” que será *interpretado* código por código pela CPU.

linguagens como Fortran e C, desta forma, bibliotecas externas podem ser usadas como se fossem uma parte dos recursos internos do Scilab. O Scilab também pode interagir com o *LabVIEW*, uma plataforma e ambiente de Linguagem de programação visual da *National Instruments*.

Do ponto de vista científico, o Scilab vem com muitos recursos. No início do desenvolvimento do Scilab, as características foram focadas na álgebra linear. Mas, rapidamente, o número de recursos foram estendidos para cobrir muitas áreas da computação científica. Por exemplo, a manipulação de números complexos é nativa da linguagem. Segue uma breve lista das suas capacidades:

- Álgebra linear, matrizes esparsas;
- Polinômios e funções racionais;
- Interpolação, aproximação;
- Otimização linear, quadrática e não linear;
- Resolução de Equação Diferencial Ordinária e de Equações Algébricas Diferenciais;
- Controle clássico e robusto, otimização linear da desigualdade da matriz;
- Processamento digital de sinais;
- Interfaces com *softwares* de computação simbólica (Maple, MuPAD);
- Estatística.

O Scilab fornece muitos recursos gráficos, incluindo um conjunto de funções de plotagem, que permitem criar gráficos 2D e 3D, bem como interfaces de usuário. Neste livro estamos usando a versão 6.0.0 do Scilab. Todos os *scripts* apresentados podem rodar em versões mais antigas. Ao digitar o comando "ver" (*version*) obtemos as informações de versão sobre Scilab instalado na máquina, como mostra a Tabela 1.1. Para qualquer dúvida sobre a sintaxe ou Scilab ou sobre o uso de algum comando, podemos usar o comando "help". Também podemos conseguir ajuda *on-line*, por exemplo no *link* `http://www.heikell.fi/downloads/scilabpdf.pdf`.

Tabela 1.1: Versão do Scilab usada neste livro

Software	valor
Scilab Version	6.0.0.1487071837
Operating System	Windows 8 6.2
Java version	1.8.0-40
Java runtime information	Java(TM) SE Runtime Environment (build 1.8.0)
Java Virtual Machine information	Java HotSpot(TM) 64 Bit Server VM (build 25.40)
Vendor specification	Oracle Corporation

Naturalmente, poderíamos ter escolhido alguma outra linguagem ou *software* para a implementação dos algoritmos deste livro. FORTRAN 99, C, Algol, Pascal, BASIC, Python, Maple, Derive, Mathematica, MathCAD, MuPAD são alguns exemplos de linguagens que poderíamos usar. A escolha do Scilab, como já exposto, não foi acidental, mas, até certo ponto, temos que admitir que foi arbitrária e a linguagem Python, por exemplo, seria também uma boa opção.

1.1 Instalação do Scilab

Instalar o Scilab é relativamente fácil. O primeiro passo é ir no endereço eletrônico `http://www.scilab.org/en/download/latest` e escolher qual versão e para qual sistema operacional queremos fazer a instalação, ver figuras 1.1, 1.2 e 1.3. Talvez seja necessário instalar também alguma máquina Java. Recomendamos instalar todos os módulos disponíveis.

Figura 1.1: Página *web* de instalação do Scilab.

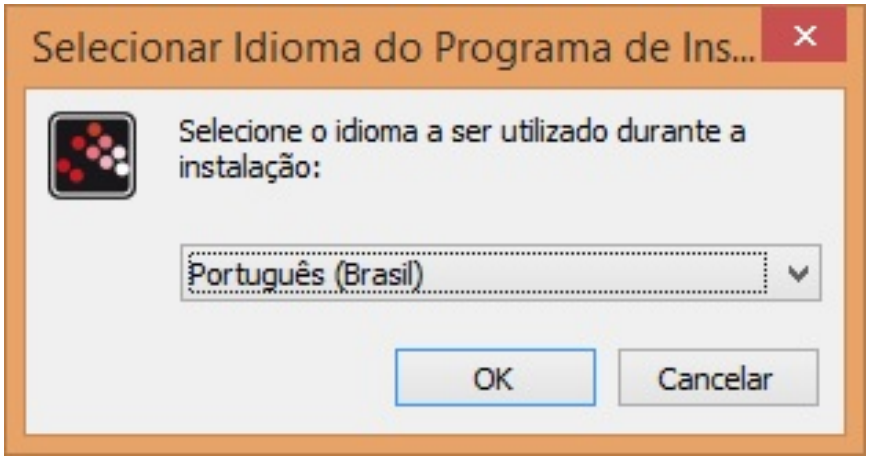


Figura 1.2: Seleção do idioma do Scilab.

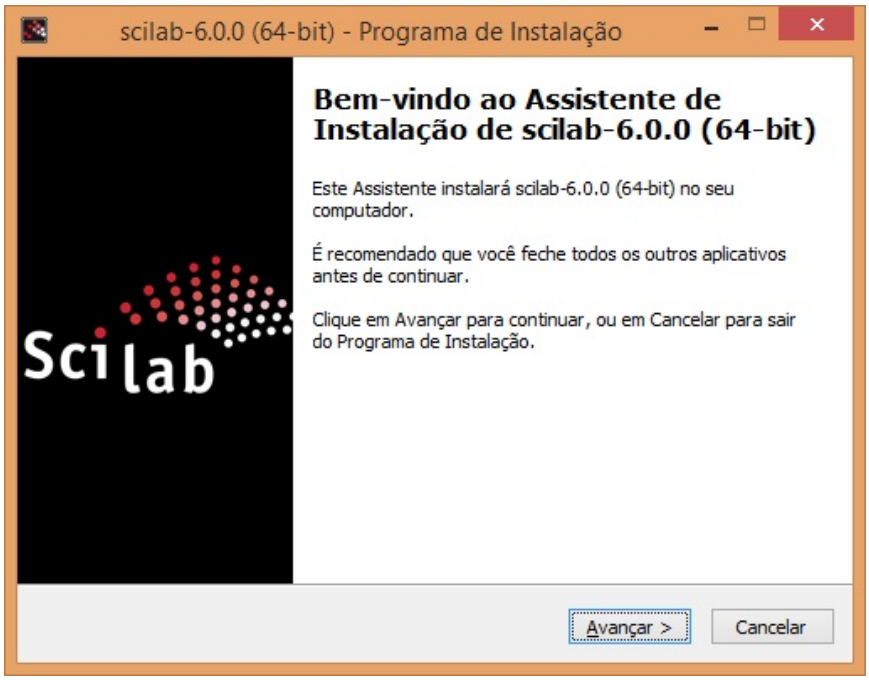


Figura 1.3: Assistente de instalação do Scilab.

1.2 Constantes pré-definidas

O Scilab tem uma relação de constantes pré-definidas que podem ser usadas nos programas. Essas constantes, algumas delas mostradas na Tabela 1.2, são precedidas pelo símbolo "%". Recomendamos não usar uma variável ou função com o mesmo nome de uma constante já definida. Um exemplo: para "converter" um valor real em complexo, basta multiplicá-lo por "*%i": x = 1 + sqrt(5)*%i. Isso gera o valor complexo $x = 1 + j\sqrt{5}$. Em geral, usamos j para representar a parte imaginária de um número complexo.

Tabela 1.2: Principais constantes do Scilab

constante	valor ou uso
%e	$2,7182818$ (neperiano)
%eps	$2,220.10^{-16}$
%i	$\sqrt{-1}$
%inf	"infinito"
%nan	"não é um número"
%pi	$3,1415927$ (π)
%s	usada como variável de polinômio
TMPDIR	diretório (*path*)

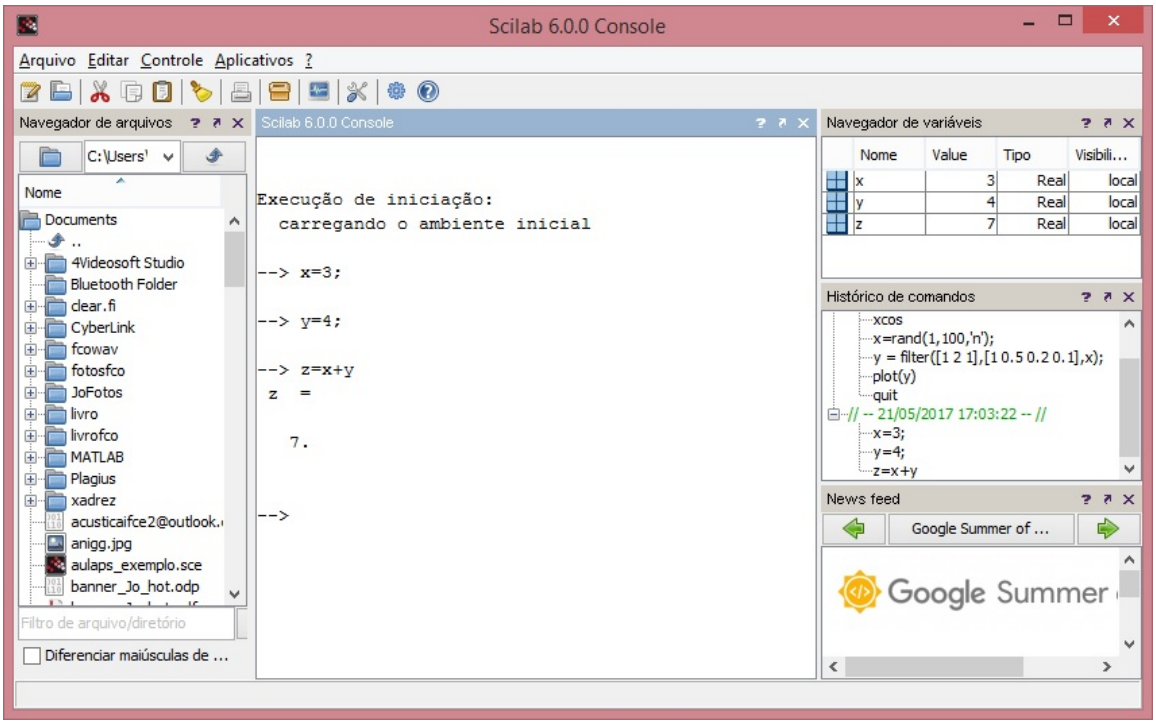


Figura 1.4: Janela inicial do Scilab

1.3 Operações básicas

O ambiente Scilab é mostrado na Figura 1.4. Podemos executar vários comandos nesta janela inicial. Podemos criar variáveis usando os caracteres ASCII de "a-z" e "A-Z", os dígitos "0-9" e caracteres especiais, mas devem começar com um caractere ASCII. Assim, "x9" é um nome válido para uma variável, mas "9x" não é. Os nomes das variáveis não podem ter mais que 24 caracteres. Minha sugestão: nomes com até 8 ou 9 caracteres. O comando de atribuição é realizado usando "=", exemplo: $x = 3$. Esse comando simples cria uma variável "x" com valor 3.

Para somar duas variáveis, usar o operador "+": "$z = x + y$". Podemos somar duas variáveis, vetores ou matrizes com o mesmo operador "+". Exemplo (uso direto do console):

```
–> x=3;
–> y=4;
–> z=x+y
 z =
 7.
```

Outros operadores para cálculo são: "-" (subtração), "*" (multiplicação), "/" (divisão), " ** " (potenciação), ver Tabela 1.3. Um exemplo: como fazer o cálculo da expressão $\ln(3,3) + \pi/\log(5,2) + 4,3^{2,5}$? No console podemos digitar:

```
–> log(3.3) + %pi/log10(5.2) + 4.3**2.5
 ans =
 43.923281
```

O termo “ans” significa “resposta”. A variável “ans” é criada automaticamente quando expressões não são atribuídas; “ans” contém a última expressão não atribuída. Existe diferença entre “log” e “log10”, o primeiro calcula o logaritmo natural, o segundo calcula o logaritmo decimal. Mais um exemplo simples: qual o valor de $sen(\pi)$? Usando o Scilab:

```
–> sin(%pi)
ans =
1.225D-16
```

O fato de que o valor calculado de $sen(\pi)$ não é exatamente igual a 0 é uma consequência do fato de que o Scilab armazena os números reais com números de ponto flutuante, ou seja, com precisão limitada. Mais um exemplo de limitação numérica com cálculos simples:

```
–>bb = 1 + 0.2 + 0.2 + 0.2 + 0.2 + 0.2 - 2
bb =
- 2.220D-16
```

O valor de “bb” apresenta um pequeno erro numérico devido a precisão finita. No capítulo 3, abordaremos esse tema com mais detalhes. Para finalizar esta seção, algumas regras que devem ser obedecidas na hora de criar uma variável:

- ter no máximo 24 caracteres, os caracteres da 25^o posição em diante são ignorados;
- começar com uma letra, seguido de outras letras ou números, alguns caracteres de pontuação são permitidos, como, por exemplo, _;
- letras maiúsculas e minúsculas geram variáveis diferentes, por exemplo: xyz ≠ XYZ.

Tabela 1.3: Operações matemáticas elementares

Operador	nome	exemplo
$=$	atribuição	$x = 2.786$, x recebe o valor 2.786
$+$	soma	$x = a + b$
$-$	subtração	$x = a - b$
$*$	produto	$x = a * b$
$/$	divisão direita	$x = a/b$ equivale a $x = ab^{-1}$
$\backslash$	divisão esquerda	$x = a\backslash b$ equivale a $x = a^{-1}b$
^	potência	$x = a\ \hat{}\ b$, equivale a $x = a^b$
$**$	potência	$x = a**b$, equivale a $x = a^b$

1.4 Funções e gráficos

O Scilab possui muitas funções prontas para uso. A Tabela 1.4 apresenta uma lista com muitas dessas funções. Um exemplo: qual o valor da expressão $tan(x) + x^{-2.5} + cosh(x/2)$ para $x = \pi/3$. Digitando no console do Scilab:

```
–> x=%pi/3; tan(x) + x**(-2.5) + cosh(x/2)
ans =
3.7633933
```

Existem muitos outros comandos, como, por exemplo, o curioso comando “calendar()”:

Tabela 1.4: Funções matemáticas e alguns comandos

acos	acosd	acosh	acoshm	acosm	acot	acotd	acoth
acsc	acscd	acsch	asec	asecd	asech	asin	asind
asinh	asinhm	asinm	atan	atand	atanh	atanhm	atanm
cos	cosd	cosh	coshm	cosm	cotd	cotg	coth
cothm	csc	cscd	csch	sec	secd	sech	sin
sinc	sind	sinh	sinhm	sinm	tan	tand	tanh
tanhm	tanm	exp	expm	log	log10	log1p	log2
logm	max	maxi	min	mini	modulo	pmodulo	sign
signm	sqrt	sqrtm	real	imag	ones	zeros	eye

```
-->calendar()
 ans  =
        ans(1)
 Jun 2017
       ans(2)
    Seg Ter Qua Qui Sex Sáb Dom
       ans(3)
    0.    0.    0.    1.    2.    3.    4.
    5.    6.    7.    8.    9.    10.   11.
    12.   13.   14.   15.   16.   17.   18.
    19.   20.   21.   22.   23.   24.   25.
    26.   27.   28.   29.   30.   0.    0.
    0.    0.    0.    0.    0.    0.    0.
```

Outros comandos relacionados com tempo e que podemos usar para calcular o tempo que um determinado código demorou para ser executado: “tic()”, “toc()” e “timer()”.

Até agora, usamos o Scilab basicamente como uma calculadora, digitando os comandos e realizando os cálculos diretamente no console. Quando o programa precisar de várias linhas de código e for necessário guardar para edição futura, é muito mais prático usar o editor de código do próprio Scilab - *SciNotes*, ver Figura 1.5.

1.5 Decisões e laços

Veremos agora os comandos básicos para tomar decisões (exemplo: ponto de parada de algoritmo) e para a execução de laços. O comando “if - then - else” é usado para executar alguma operação se a condição for verdadeira ou executar outra operação se a condição for falsa. Estrutura básica: *if* <expressão lógica> *then* <comandos> *else* <comandos> *end*. Exemplo:

```
–> if x>y then
> z = x + 2
> else
> z = y + 3
> end
```

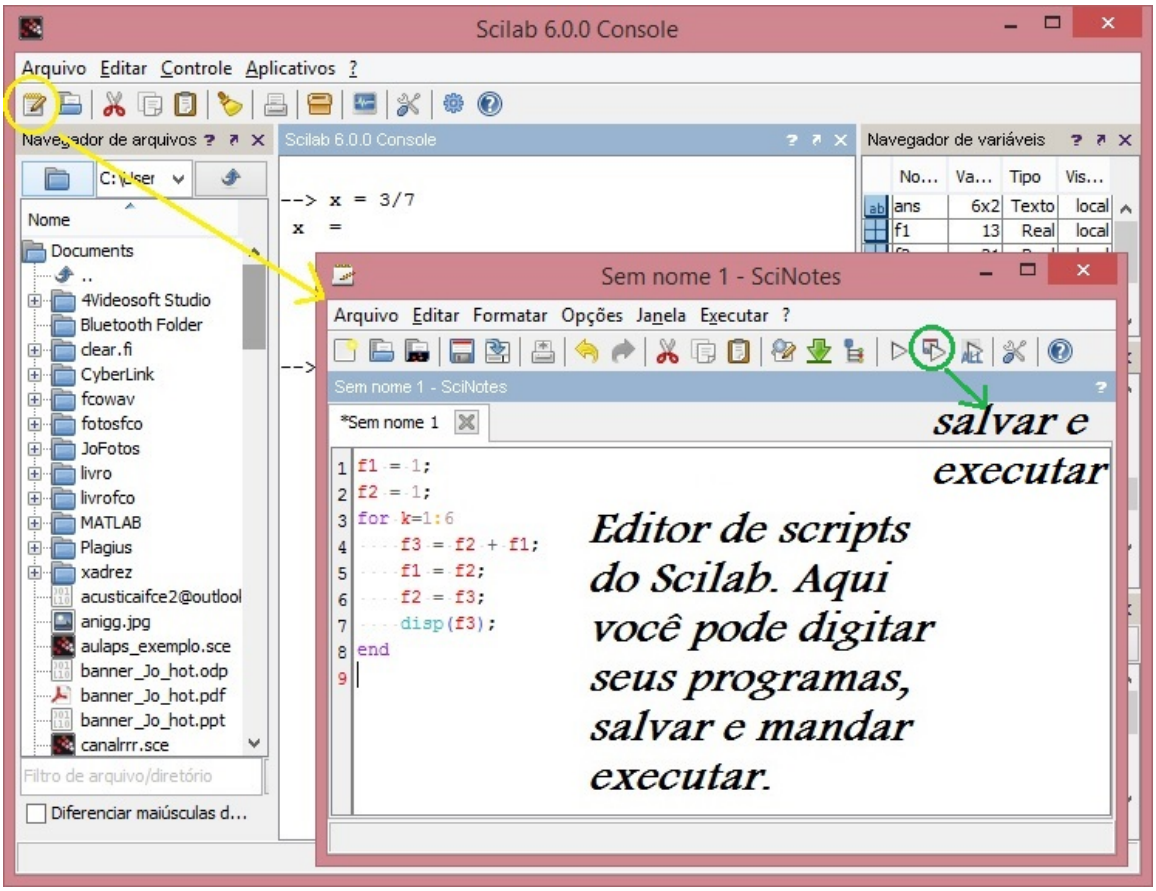


Figura 1.5: Editor de *scripts* do Scilab

O comando “while” (enquanto) realiza um teste lógico, se a condição for verdadeira um conjunto de instruções é executado, caso contrário, o laço termina. Formato: *while* <condição> <comandos> *end*. Exemplo:

```
a=1; k=1;
while a>%eps
    a=a/2;
    k=k+1;
end
```

O comando “for” realiza um laço de forma incondicional. Formato: *for* <intervalo> <comandos> *end*. Exemplo:

```
y = 0;
for k=1:5
   y = k**2;
   x = sqrt(y + 3);
```

```
end
```

No laço acima usando o comando "for", o intervalo é definido por "k=1:5" que faz com que a variável "k" varie de 1 a 5 com passo 1. Se fosse desejado que passo fosse $0,2$ (k assumindo os valores 1; 1,2; 1,4, ...), o laço seria reescrito como:

```
y = 0;
for k=1:0.2:5
   y = k**2;
   x = sqrt(y + 3);
end
```

Exemplo: Sequência de Fibonacci. A sequência de Fibonacci é definida como: $F_n = F_{n-1} + F_{n-2}$, com $F_0 = 1$ e $F_1 = 1$. Escrever um código Scilab que gere os primeiros termos da sequência de Fibonacci.

```
clc;
f1 = 1;
f2 = 1;
for k=1:6
    f3 = f2 + f1;
    f1 = f2;
    f2 = f3;
    disp(f3);
end
```

Que resulta na saída: 2, 3, 5, 8, 13, 21. O comando "disp()" (*display*) mostra o resultado (variável f3) no console, já "clc" limpa a tela do console. Uma forma mais compacta para gerar a mesma sequência:

```
v = [0, 1];
for k=1:10
   v = [v, v($)+v($-1)];
end;
```

Outros comandos úteis: *select* e *case*, *break*, *abort* e *pause*.

1.6 Vetores e matrizes

Suponha que seja desejado calcular o valor de $cos(x)$ com x variando de 0 a $\pi/2$ com um passo de $\pi/10$. Poderíamos escrever o *script*:

```
passo = %pi/10;
for x=0:passo:%pi/2
  disp([x, cos(x)]);
end
```

Uma forma mais eficiente seria simplesmente escrever:

```
passo = %pi/10;
x=0:passo:%pi/2;
c=cos(x);
```

No código acima, são criados dois *vetores*: "x" e "c", não é necessário o uso de um laço para o cálculo do cosseno de "x". Um vetor é um conjunto linha (elementos separados por espaço em branco ou vírgulas) ou coluna (elementos separados por ponto-e-vírgula ou "enter") de dados homogêneos (mesmo tipo). Exemplos de vetores criados manualmente no console:

```
–>v1 = [2, 4, -7 8, 3.3, -1.2, 0 7]; v2 = [-3.3; sqrt(2); sqrt(-2)];
–>v1, v2
 v1 =
 2. 4. - 7. 8. 3.3 - 1.2 0. 7.
 v2 =
- 3.3
1.4142136
1.4142136i
```

Podemos acessar qualquer elemento dentro de um vetor diretamente: "y = v1(2)", atribui à variável y o valor 4. O índice dos vetores começa no valor "1", não existe posição "0". A transposição converte um vetor linha em coluna, e vice-versa. Se o vetor for complexo, a transposição realiza também a função de conjugação complexa. Exemplo:

```
–>v3 = v2'
 v3 =
- 3.3 1.4142136 - 1.4142136i
```

Existem alguns comandos especiais para criar vetores e matrizes: *zeros*, *ones*, *eye*. Exemplo: "x = zeros(4,4)" gera uma matriz 4×4 de zeros:

```
–> x = zeros(4,4)
 x =
 0.    0.    0.    0.
 0.    0.    0.    0.
 0.    0.    0.    0.
 0.    0.    0.    0.
```

Já o comando "y = eye(3,3)" cria uma matriz identidade 3×3:

```
–> x = eye(3,3)
 x =
 1.    0.    0.
 0.    1.    0.
 0.    0.    1.
```

A soma, o produto e outras operações entre matrizes só podem ser realizadas se eles tiverem dimensões compatíveis, seguindo as regras usuais da matemática. A soma entre dois vetores é feita sinal "+", o produto é feito com ".*" (produto ponto a ponto) ou "*" produto entre vetores e matrizes. Se as dimensões não forem compatíveis, uma mensagem de erro do tipo "*Inconsistent row/column*

dimensions" será mostrada no console. O exemplo seguinte ilustra esses detalhes. Código:

```
A = [1, 2, 3, 4];
B = [4, 5, 0, 3];
M1 = [1, 2, 3; 4, 5, 6; 7, 8, 9];
M2 = eye(3,3);

C = A + B;
D = A .* B;
Ms = M1 + M2;
Mp = M1 * M2;
Mpp = M1 .* M2;
```

Os resultados são:
C = 5. 7. 3. 7.
D = 4. 10. 0. 12.
Ms =
2. 2. 3.
4. 6. 6.
7. 8. 10.
Mp =
1. 2. 3.
4. 5. 6.
7. 8. 9.
Mpp =
1. 0. 0.
0. 5. 0.
0. 0. 9.
Note que "Mp" e "Mpp" são matrizes diferentes: M1*M2 ≠ M1.*M2.

1.7 Sistemas lineares

A solução de um sistema linear de equações de N incógnitas e N equações é muito simples no Scilab. Seja um sistema na forma $Ax = B$, onde A é uma matriz $N \times N$, B um vetor coluna $N \times 1$. A solução é $x = A^{-1}B$. Em um script Scilab: "x = inv(A)*B". Vejamos um exemplo:

$$\begin{bmatrix} 3 & 2 & 2 & 1 \\ 1 & 3 & 0 & 3 \\ 1 & 1 & 5 & 4 \\ 3 & 3 & 2 & 2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 4 \\ 2 \end{bmatrix}$$

Em código Scilab:

```
A = [3 2 2 1; 1 3 0 3; 1 1 5 4; 3 3 2 2];
b = [3; 5; 4; 2];
x = inv(A)*b;
```

Resulta em:
x =
8.
-9.0909091
-5.4545455
8.0909091
Uma observação: o Scilab permite a criação de "hipermatrizes". Uma hipermatriz é um conjunto homogêneo de dados com mais de duas dimensões, usamos o comando "M=hypermat(dims [,v])" para essa tarefa.

1.8 Entrada, saída e gráficos

O comando "input" é usado para entrar com algum valor para uma variável, fornecendo ao usuário o *prompt* de texto esperando por uma entrada do teclado. Exemplo: "x = input("Número de iterações")". Para mostrar algum resultado na tela (console), podemos usar os comandos "disp()" e "mprintf()". Exemplo:

```
kk=22;
xx=7.8/4.3;
mprintf('Na iteração %i, o resultado e alpha=%f \n',kk,xx);
mprintf('%e \n', [1; 2; 3]);
mprintf('%d %d \n', [1, 2; 3, 4]);
```

Resposta:

```
Na iteração 22, o resultado e alpha=1.813953
1.000000e+00
2.000000e+00
3.000000e+00
1 2
3 4
```

No exemplo acima, os parâmetros "%d", "%e" e "" usados no comando "mprintf()" indicam a impressão de um valor inteiro, um número no formato científico e uma nova linha, respectivamente. O comando básico para fazer um gráfico é o "plot". Se digitarmos no console "plot", o Scilab irá gerar um exemplo de gráfico usando o comando "plot", o mesmo vale para "plot2d" e "plot3d". Podemos gerar um gráfico com várias curvas na mesma janela, o comando "plot" permite o controle da cor da curva e o tipo de traço gerado. Com o comando "legend", incluímos uma legenda (importante para gráficos com várias curvas) e com "xgrid", uma grade pontilhada no gráfico. O comando "close" fecha uma janela eventualmente aberta. Podemos ainda incluir um título, nomes para os eixos x e y. Exemplo (ver Figura 1.6):

```
close;
x = 0:0.01:6;
y = 2*exp(-x/2).*sin(2*x);
z = x.*(6-x).*cos(3*x)/4;
w = 2*x./(x.*x + 1);
```

```
plot(x,y,x,z,x,w); xgrid;
legend('2*exp(-x/2).*sin(2*x)','x.*(6-x).*cos(3*x)/4','2*x./(x.*x + 1)');
```

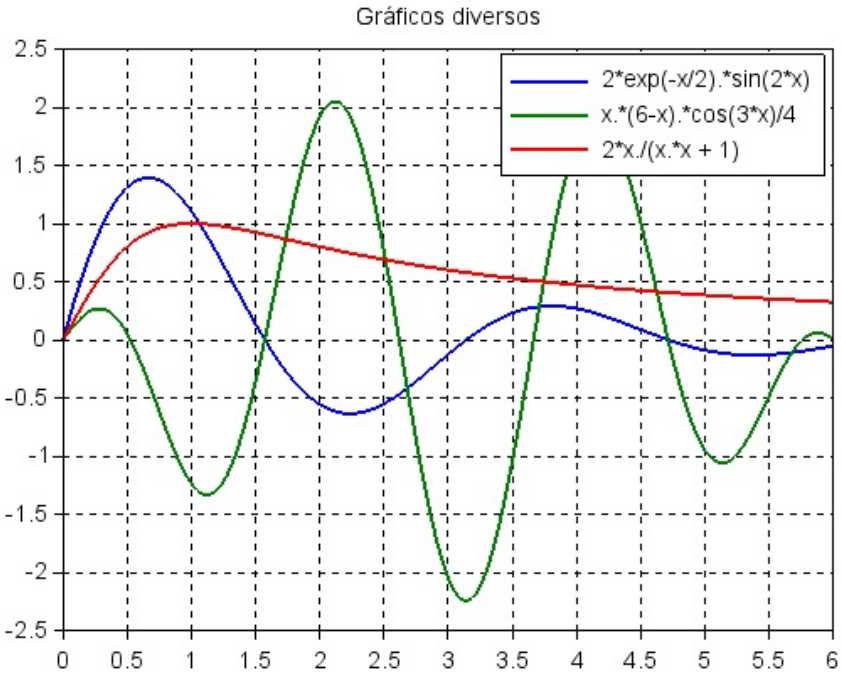


Figura 1.6: Alguns gráficos na mesma janela

Um segundo exemplo de gráfico (ver Figura 1.7):

```
close;
p4 = 16*%pi+0.1;
t = -p4:0.01:p4;
x1 = 1.1*cos(t/5);
y2 = exp(-abs(t)/12).*sin(t/3);
plot(x1,y2,'r',x1(1:40:$),y2(1:40:$),'m.');
title('Curva ...');
xlabel('Eixo x');
ylabel('Eixo y');
```

Neste segundo exemplo, o símbolo "$" indica o fim do vetor. Assim, sobre a curva suave formada pelo par "x1" e "y1", são colocados pontos de cor magenta ("m."). Obtemos esses pontos da própria curva, fazendo uma subamostragem de um a cada 40 pontos ("1:40:$"). O Scilab suporta alguns comandos LaTeXna hora de incluir um título ou legenda em um gráfico. Também podemos fazer gráficos 2D (ver Figura 1.8):

```
[X,Y]=meshgrid(-2:.1:2,-2:.1:2);
Z=X.^2+Y.^2;
close;
xtitle('$ z=x^2+y^2$');
mesh(X,Y,Z);
```

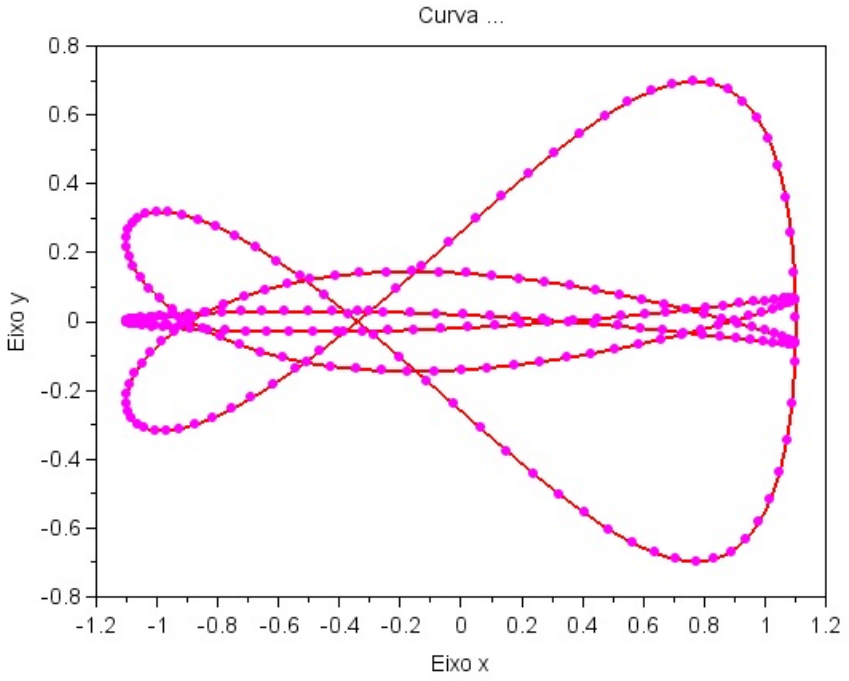


Figura 1.7: Exemplo de curva paramétrica

Com o comando "subplot" podemos criar vários gráficos em uma mesma janela. Vejamos um exemplo (Figura 1.9):

```
t = 0:0.01:16;
y1 = fix(t);
y2 = 1.2*cos(%pi*t/2);
y3 = sqrt(t);
y4 = exp(-t/4).*sin(%pi*t/3);
figure;
subplot(2,2,1); plot(t,y1); title('Função escada');
subplot(2,2,2); plot(t,y2); title('Função cosseno');
subplot(2,2,3); plot(t,y3); title(['Função ','$t^{1/2}$']);
subplot(2,2,4); plot(t,y4); title(['Função seno amortecido','$ e^{-t/4}\sin(\pi t/3)$']);
```

Algumas vezes estamos interessados em traçar o gráfico de alguma grandeza discreta no tempo como, por exemplo, a cotação do dólar em determinado período ou o valor do erro em cada iteração dentro de um laço. Para gerar o gráfico de barras, usamos o comando "bar". A Figura 1.10 mostra um exemplo de gráfico de barras. Na Figura 1.11 temos um exemplo do gráfico de de função em escala linear e em escala logarítmica, código para esse exemplo:

```
w=0.1:0.01:20;
s=w*%i;
hs = 5*s./(s.*s + 4*s + 4);
hsa = abs(hs);
subplot(1,2,1);
```

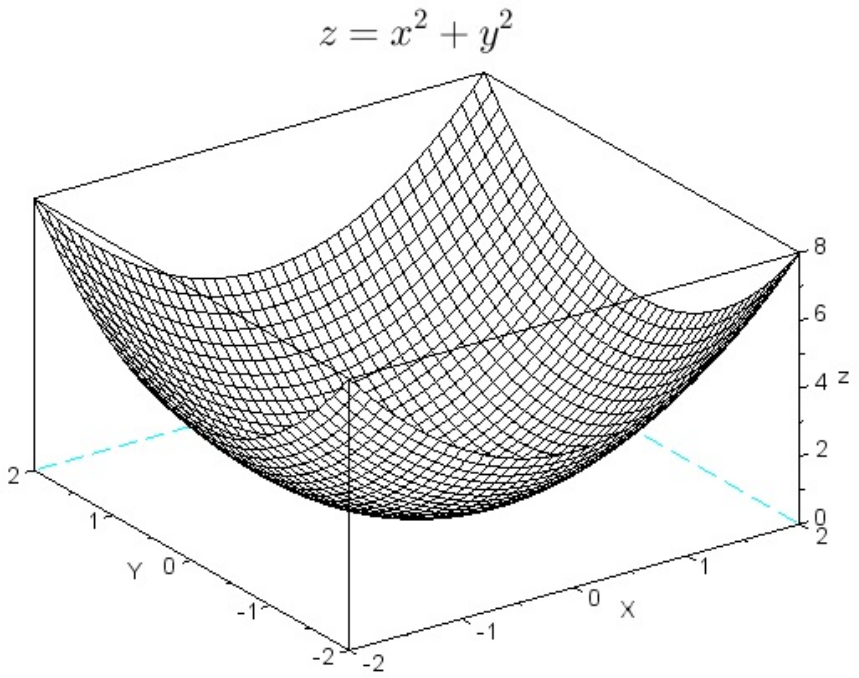


Figura 1.8: Exemplo de gráfico 2D

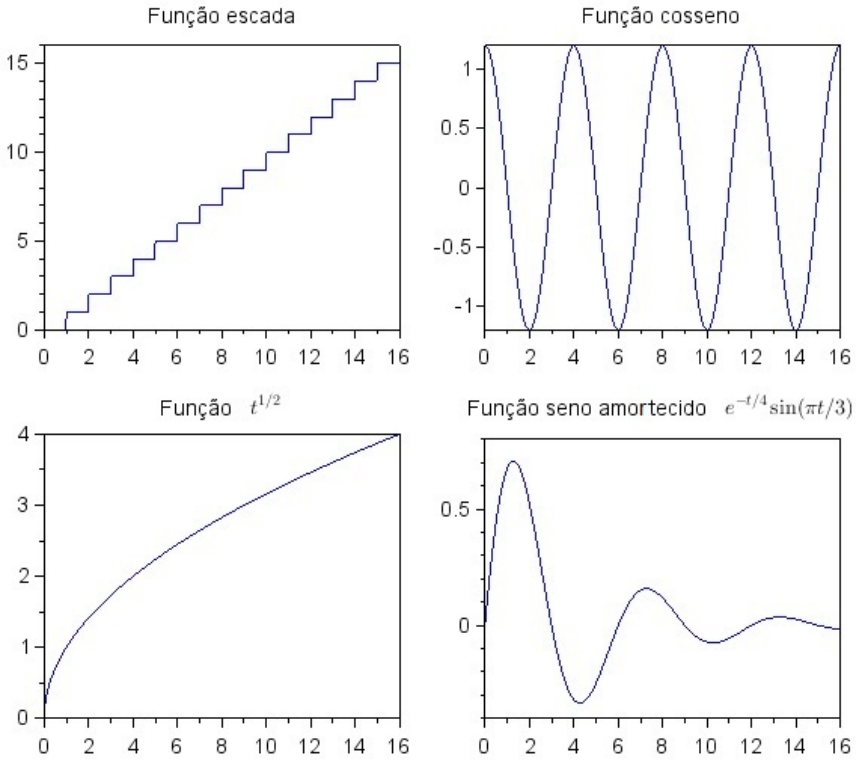


Figura 1.9: Uso do comando sublot para gerar vários gráficos em uma janela.

```
plot(w,hsa); xgrid(3); title('Gráfico em escala linear');
subplot(1,2,2);
plot(w,hsa); xgrid(3); title('Gráfico em escala LOG');
a=get("current_axes");
a.log_flags = "lln";  // eixos ‘‘x’’ e ‘‘y’’ em escala log
```

Já na Figura 1.12 temos outros exemplos de gráficos que podemos obter com os comandos "plot3d", "contour" e "param3d".

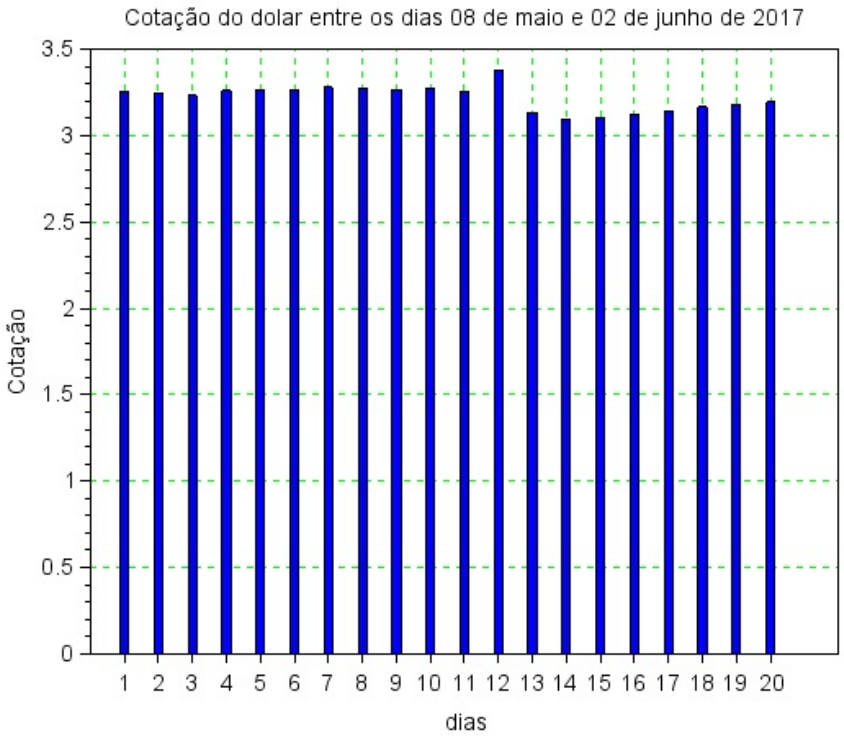


Figura 1.10: Gráfico de barras.

1.9 Criando novas funções

Frequentemente desejamos criar uma função nossa que será chamada no corpo do código uma ou mais vezes. A definição de uma função no Scilab é muito fácil e segue a seguinte estrutura:

```
function <argumentos_de_saída> = <nome_da_função><argumentos_de_entrada>
<declarações>
endfunction
```

Um exemplo de uso de função:

```
function [mmax, mmed, mmin] = fxmi(dados)  //esta função determina os valores
    mmax = max(dados);                     //máximo, médio e mínimo de um vetor
    mmed = mean(dados);
    mmin = min(dados);
```

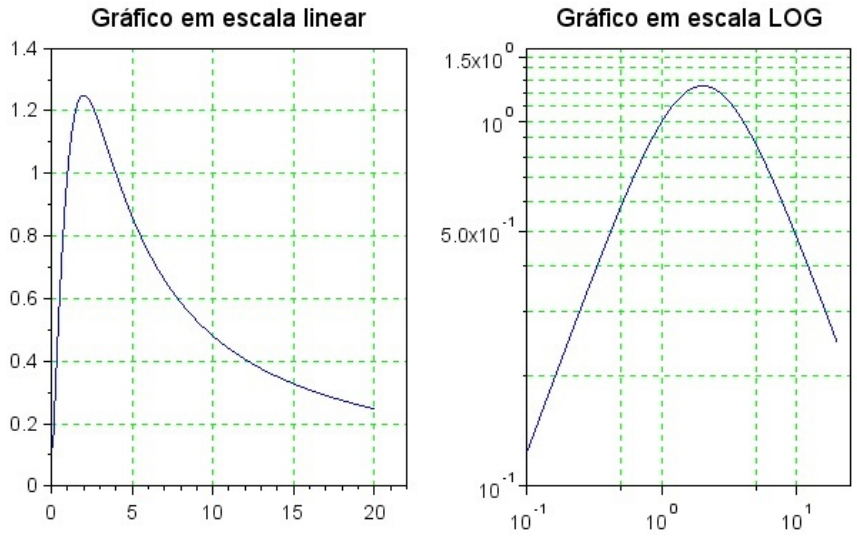


Figura 1.11: A mesma curva mas em escalas diferentes

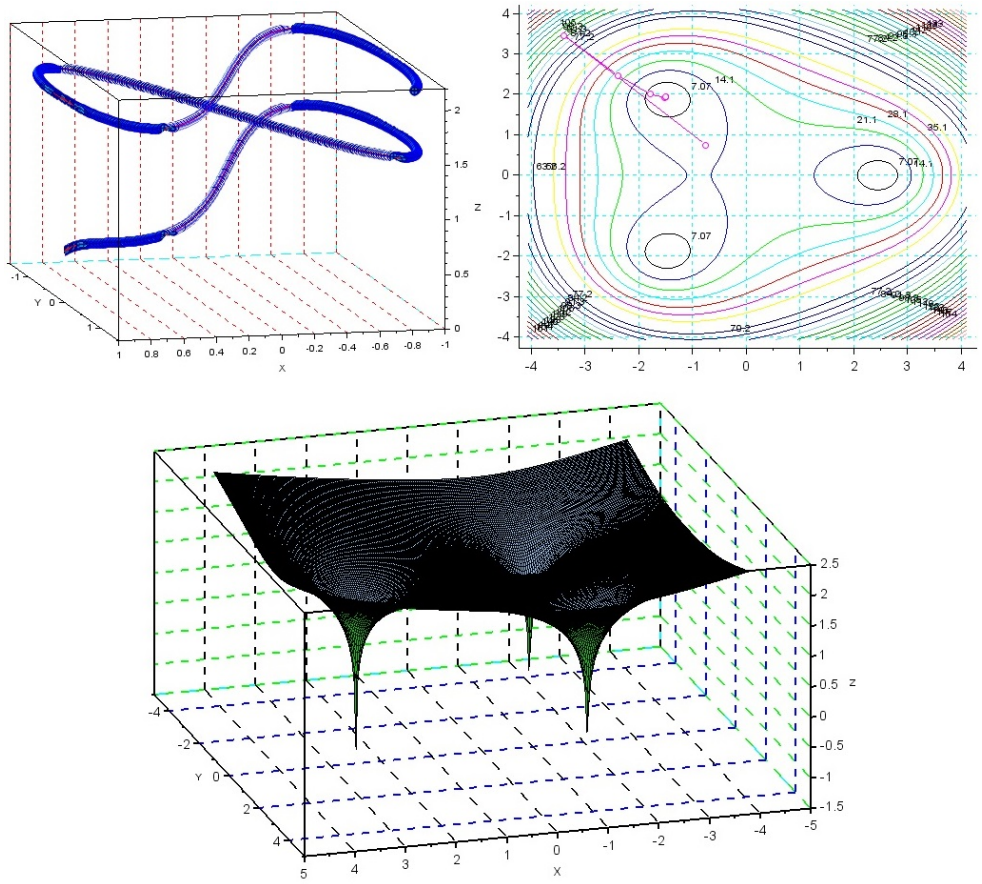


Figura 1.12: Gráficos diversos com “plot3d”, “contour” e “param3d”

```
endfunction

x1 = rand(1,100,'n');  // gerando número pseudo-aleatórios do tipo ‘‘normal’’ ('n')
x2 = rand(1,100,'n');
x3 = rand(1,100,'n');

[x1x, x1m, x1i] = fxmi(x1);
[x2x, x2m, x2i] = fxmi(x2);

disp([x1x, x1m, x1i]);   // mostrando os resultados
disp([x2x, x2m, x2i]);
disp(fxmi(x3));
```

A saída desse programa varia a cada vez que é executado, um exemplo:
1.7921657 - 0.1165078 - 2.4310499
2.0519496 - 0.0624232 - 3.248551
2.6464951

Notamos, no exemplo acima, que a função “disp(fxmi(x3))” só retorna um único valor o primeiro valor gerado pela função “fxmi”. Um segundo exemplo simples:

```
function ff = fpoli(x)  // Calculando o valor do polinômio
    ff = x^3 - 5*x^2 + 3*x - 6;
endfunction

a = fpoli(3.3);
b = fpoli(2.7);
c = fpoli(5.5);

disp([a, b, c]);
```

Tem como saída: “-14.613 -14.667 25.625”.

O fatorial de um número inteiro n pode ser calculado por: $n! = n(n-1)(n-2)... \times 2 \times 1$. A função a seguir implementa o cálculo do fatorial de um número inteiro.

```
function ft = ffat(x)  // função recursiva para cálculo de fatorial
    if x>1 then
        ft =x*ffat(x-1);
        else ft = 1;
    end;
endfunction

clc;
for k=1:6  // laço que mostra o fatorial de 1 a 6
    p = ffat(k);
    disp(p);
```

```
end;

k = input('Entre com um valor inteiro: ');
kf = ffat(k);
disp(kf);
```

Exemplo de saída:
1.
2.
6.
24.
120.
720.
Entre com um valor inteiro: 7
5040.

1.10 String, part, ascii

Algumas vezes, precisamos trabalhar com textos ou converter um valor numérico em um texto (*string*) ou, ainda, converter uma sequência de caracteres em valores numéricos inteiros. Podemos realizar essas operações com os comandos "string", "part" e "ascii", entre outros comandos. Um breve resumo de como esses comandos funcionam:

- string: converte um número (real ou inteiro) em um texto. Exemplo: txt = string(23.1238).
- part: seleciona um caractere de uma sequência de caracteres (texto). Exemplo: pp = 'abcdedfg'; n2 = part(pp,2). Neste caso, o valor de n2 é igual a 'b'.
- ascii: se a entrada for uma sequência de caracteres, ascii retorna a sequência numérica correspondente; se a entrada for uma sequência numérica de inteiros (entre 0 e 255) a saída é uma sequência de caracteres (texto).

Um exemplo:

```
x = rand(1,100,'n');  // vetor com 100 valores pseudo-aleatórios
xmax = max(x);  // valor máximo
xmin = min(x);  // valor mínimo
xmen = mean(x); // valor médio
valores = string([xmin, xmen, xmax]);
// valores: texto com os valores mínimo, médio e máximo
plot(x); title(['valores = ',valores]); // gráfico e título
```

1.11 Dois exemplos numéricos

Vejamos agora dois exemplos de problemas que podemos resolver numericamente usando alguns dos comandos estudados ao longo deste capítulo. Lembramos que as soluções apresentadas não objetivam a serem "ótimas" em qualquer sentido, mas servem apenas ilustrar o uso do Scilab. Os conceitos matemáticos envolvidos na solução desses problemas serão aprofundados nos próximos capítulos deste livro.

Problema 01. Seja a função $f(x) = 2\,\mathrm{sen}(x) - (x/2)^3$, para quais valores $f(x) = 0$? Solução: um raiz "trivial" de $f(x)$ é $x = 0$, a outra raiz está no intervalo $(2,3)$. Podemos obter essa outra raiz numericamente pelo seguinte procedimento numérico:

$A:\ x_0 = 2.5$ (inicialização)

$B:\ x_{k+1} = 2\sqrt[3]{2\,\mathrm{sen}(x_k)}$

$C:\ erro = x_{k+1} - x_k$

$D:\ k = k+1$

$E:$ se $erro > tol$ volte ao passo "B"

sendo tol uma tolerância, por exemplo: $tol = 10^{-3}$. Um código Scilab que realiza esse algoritmo é:

```
clc;        // limpa o console
x = 2.5;    // valor inicial
erro = 1;   // erro inicial
tol = 1e-3; // tolerância
k=0;        // contator
while erro > tol     // teste
    k=k+1;           // incrementa contator
    xk = 2*(2*sin(x))^(1/3);  // expressão de iteração
    disp([k, xk]);            // mostra resultados
    erro = abs(x-xk);         // calcula o erro
    x = xk;                   // atualiza o xk
end  // fim do laço
```

A saída deste programa é:

```
1.    2.1235116
2.    2.3879976
3.    2.2204861
4.    2.3355745
    ...
15.    2.2905194
16.    2.2914499
```

Logo, a raiz procurada é $x \cong 2,29145$.

Problema 02. Considere a função $f(t) = \sqrt{t}e^{-t/2} + e^{-t}\,\mathrm{sen}\,2t$, para $t \geq 0$. Determine, aproximadamente, seu ponto de máximo. Solução: não está sendo pedido a solução analítica (que é de obtenção difícil), logo, podemos apenas esboçar o gráfico usando o comando "plot". Devemos usar um passo pequeno para o "tempo". O código Scilab, neste caso, é muito simples:

```
t = 0:0.001:5;   // tempo, passo de 0.001
f = sqrt(t).*exp(-t/2) + exp(-t).*sin(2*t);  // função
plot(t,f);  // gráfico
[fmax, px] = max(f);  // encontrando o máximo da função
tmax = t(px);  // tempo de máximo
```

```
disp([tmax, fmax]);  // mostrando o ponto de máximo
plot(tmax,fmax,'ro');  // destacando o ponto de máximo no gráfico
xgrid; title('$f(t) = \sqrt{t}e^{-t/2} + e^{-t}\sin{2t}$'); // grid e título
xlabel('tempo'); ylabel('f(t)'); // eixos de tempo e amplitude da função
```

Podemos ver o resultado do código acima na Figura 1.13. O ponto de máximo é $t = 0,626$ e $f(t) = 1,0863495$.

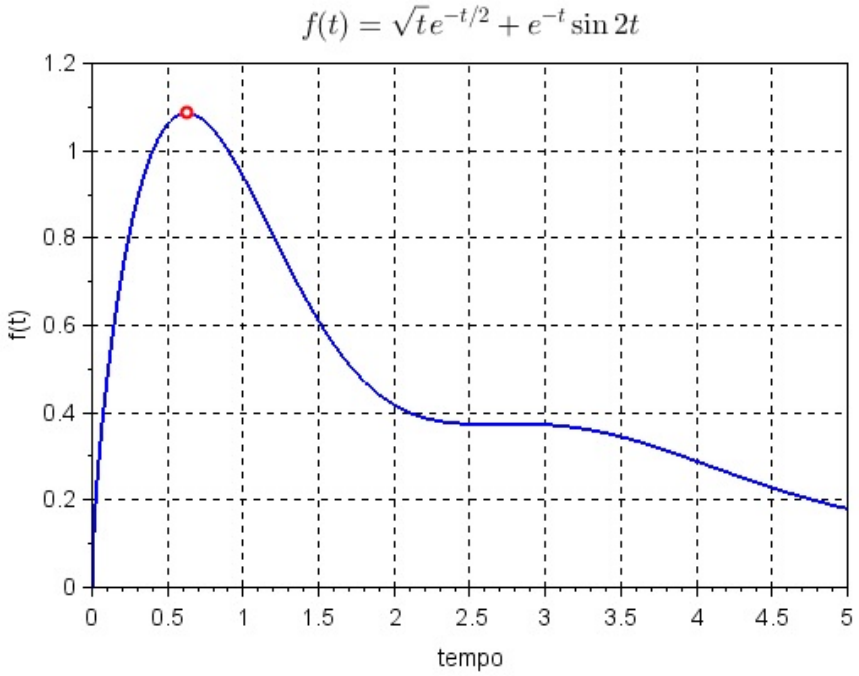


Figura 1.13: Exemplo 02: encontrando o máximo de uma função.

1.12 Usando GUI

Usamos o comando "uicontrol" para incluir botões, lista de itens, texto e outros objetos em uma janela. Podemos usar esses recursos para gerar uma janela com controles e mais amigável para a entrada de dados. Esses recursos não são "obrigatórios" para a solução da grande maioria dos problemas. Entretanto, quando queremos gerar uma janela com a opção de entrada de dados, pode ser interessante o uso desses controles. Cada "botão" ou "lista de itens" devem ser programados no código (posição, tamanho, localização na janela, etc), gerando um excesso de código que não executa um cálculo determinado. Iremos usar um exemplo para ilustrar o uso desses controles - ver Figura 1.14.

O código Scilab para gerar a janela indicada na Figura 1.14 é:

```
////////////////////////////////////
/// Gráfico de uma equação de 3o. grau.
/// Parâmetros de entrada: a, b, c, d,
```

```
/// da equação: f(x) = ax^3 + bx^2 + cx + d
/// Intervalo desejado: [x0, x1]
////////////////////////////////////

function calcular() // função calcular
// Pegando os coeficientes:
    popc = findobj("Tag", "a1");  // "apontando" o objeto "a1"
    a11 = evstr(get(popc, "string")); // convertendo texto em número
    popc = findobj("Tag", "a2");
    a12 = evstr(get(popc, "string"));
    popc = findobj("Tag", "a3");
    a13 = evstr(get(popc, "string"));
    popc = findobj("Tag", "a4");
    a14 = evstr(get(popc, "string"));

    popc = findobj("Tag", "x0");
    x00 = evstr(get(popc, "string"));
    popc = findobj("Tag", "x1");
    x11 = evstr(get(popc, "string"));

    pp = (x11-x00)/100;
    x = x00:pp:x11;  // gerando o intervalo

    f = ((a11.*x + a12).* x + a13).* x + a14;

    figure;   // gráfico da função com grid
    plot(x,f); xgrid;
endfunction

function ajuda()
    messagebox("Entre com o intervalo e com os coeficientes.",...
"modal", "info", ["Ok"]);
endfunction

close;  clc;   // fechando alguma janela aberta

f1 = figure();  // criando uma janela
f1.figure_name = " Gráfico de f(x) = ax^3 + bx^2 + cx + d";
altura = 180; largura = 350; // dimensões da janela
f1.position = [10 5 largura altura];
f1.info_message = ' Livro de métodos numéricos (2019).'

// Elementos de Texto estático:
text_00 = uicontrol(f1, "Position", [20 altura-60 70 20], ...
    "Style", "text", "FontSize", 11, "String", " Coeficientes:", ...
    "BackgroundColor", [1 0.5 0.5]);
```

```
text_01 = uicontrol(f1, "Position", [20 altura-30 70 20], ...
    "Style", "text", "FontSize", 11, "String", " Intervalo:", ...
    "BackgroundColor", [1 0.5 0.5]);

// Coeficientes do polinôminio
// Primeiro coeficiente:
a1 = uicontrol(f1, "Position", [110 altura-60 30 20], ...
    "Style", "edit", "FontSize", 11, "String", gettext("1"), ...
    "BackgroundColor", [1 1 1], "Tag", "a1");

// Segundo coeficiente:
a2 = uicontrol(f1, "Position", [145 altura-60 30 20], ...
    "Style", "edit", "FontSize", 11, "String", gettext("-2"), ...
    "BackgroundColor", [1 1 1], "Tag", "a2");

// Terceiro coeficiente:
a3 = uicontrol(f1, "Position", [180 altura-60 30 20], ...
    "Style", "edit", "FontSize", 11, "String", gettext("-3"), ...
    "BackgroundColor", [1 1 1], "Tag", "a3");

// Quarto coeficiente:
a4 = uicontrol(f1, "Position", [215 altura-60 30 20], ...
    "Style", "edit", "FontSize", 11, "String", gettext("4"), ...
    "BackgroundColor", [1 1 1], "Tag", "a4");

// Intervalo - x0:
x0 = uicontrol(f1, "Position", [110 altura-30 30 20], ...
    "Style", "edit", "FontSize", 11, "String", gettext("-3"), ...
    "BackgroundColor", [1 1 1], "Tag", "x0");

// Intervalo - x1:
x1 = uicontrol(f1, "Position", [145 altura-30 30 20], ...
    "Style", "edit", "FontSize", 11, "String", gettext("3"), ...
    "BackgroundColor", [1 1 1], "Tag", "x1");

// Calculando o gráfico da função!
button01 = uicontrol(f1, "Position", [largura/2-50 60 100 20], ...
    "Style", "pushbutton", "FontWeight", "bold", "FontSize", 12, ...
"String", "Calcular!", "Foregroundcolor",[0.1 0.1 1], "callback", "calcular();");

// Botão de ajuda:
but_ajuda = uicontrol(f1, "Position", [largura-110 20 100 20], ...
    "Style", "pushbutton", "FontWeight", "bold", "FontSize", 12, ...
"String", "Ajuda?", "Foregroundcolor",[0.1 0.8 0.2], "callback", "ajuda();");
```

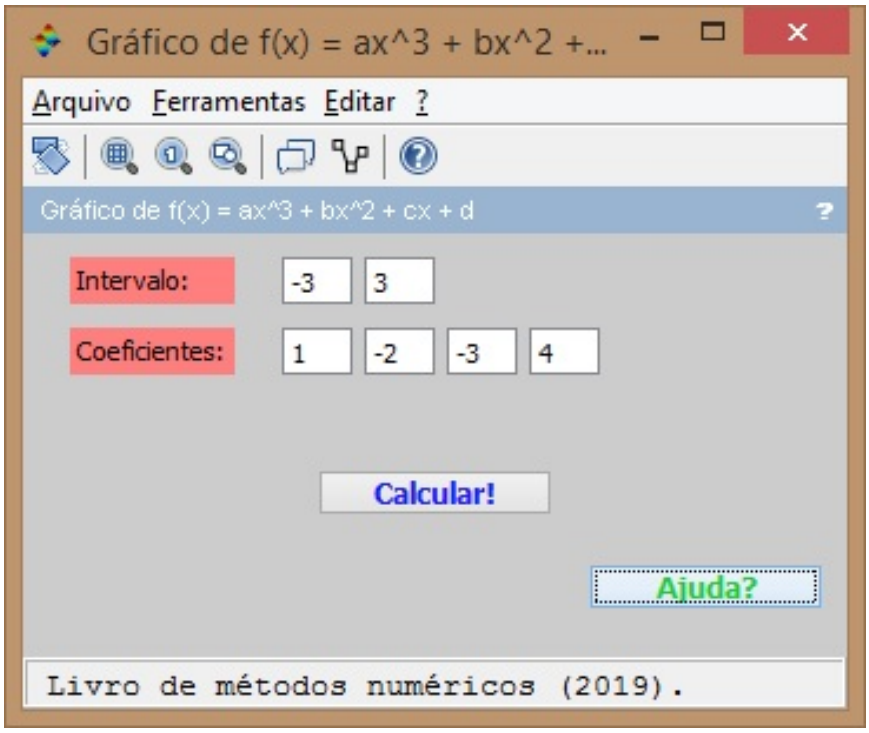


Figura 1.14: Janela com controles.

1.13 XCOS

O Scilab também possui um conjunto de blocos prontos para serem usados em uma interface gráfica. Esses blocos podem ser interligados para realizarem funções ou sistemas extremamente complexos. O XCOS é um *toolbox* que permite a simulação de sistemas dinâmicos tanto de natureza contínua (fontes de tensão senoidais, capacitores, massas, etc.), quanto discreta (filtros digitais, atrasos, *sample-and-hold*, etc). Possui também recursos de pós-processamento gráfico que permite ao usuário realizar a apresentação da resposta dinâmica do sistema. Trata-se de um ambiente de programação em comandos de alto nível onde a simulação se desenvolve através da utilização de blocos, contendo rotinas numéricas. Na Figura 1.15, temos uma visão geral desse ambiente[3]. Usaremos pouco esse recurso neste livro[4].

1.14 Algumas dicas

Ao longo deste livro, implementaremos diversos algoritmos. Esses programas devem ser legíveis e funcionais. Dificilmente os programas de computador podem ser testados para garantir 100% de confiabilidade. Quase sempre existe o perigo de erros ocultos e que, retrospectivamente, são *bugs*[5]

[3] Uma apostila sobre essa *toolbox*: `http://sites.poli.usp.br/d/pme3200/Tutorial_scicos.pdf` - acesso dia 02/09/2017

[4] Para leitores interessados, mantenho um blog e fiz algumas postagens mostrando o uso do XCOS: `http://alfanumericus.blogspot.com.br/search?q=xcos`

[5] O termo "*bug*", segundo uma história mais ou menos verossímil, foi cunhado em 1947, quando Grace Hopper (uma das primeiras mulheres programadoras de computadores) da Marinha dos EUA identificou um problema de computação sendo causado por uma traça (*bug*) em um relé (tubo ou conector, a história varia tem algumas variações). O "erro"

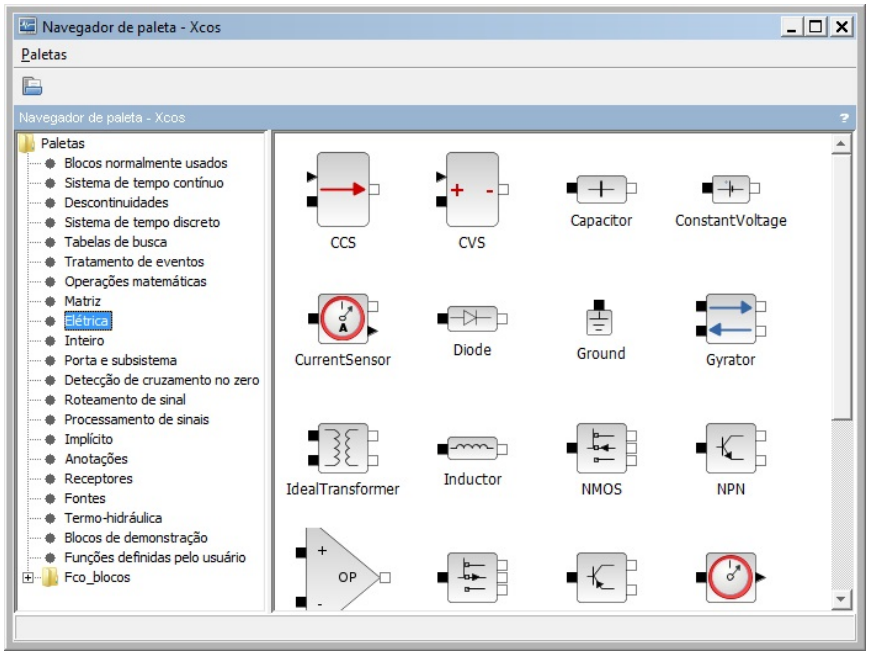


Figura 1.15: XCOS: interface gráfica do Scilab para modelagem de sistemas dinâmicos.

óbvios. Evitar armadilhas de programação comuns devem ser um objetivo mínimo e exige que os conheçamos. Tendo isso em mente, apresentamos algumas dicas:

- um programa é caracterizado por: a) sua estrutura e b) pelo que faz;
- dê às variáveis nomes claros e significativos;
- use apenas letras únicas para os eixos x, y e z, contadores de laços (j,k) e similares;
- divida o código em entidades lógicas com a ajuda de sub-rotinas ($function$);
- separar entidades estruturais por linhas vazias e comentários em destaque;
- mantenha o recuo de texto em laços, comandos de impressão, etc, para aumentar a clareza;
- seja pródigo com o uso de comentários (tente ler um código sem comentários que você mesmo escreveu a há dois meses e verá que comentar é vital);
- mantenha espaço entre os comandos e seus comentários de linha;
- um bom programa é curto e sem ambiguidade.

Além disso, temos que ter em mente que algoritmos matematicamente equivalentes não são, necessariamente, numericamente equivalentes. Finalmente, no Apêndice deste livro, temos alguns exemplos didáticos de como converter um algoritmo em um código Scilab.

1.15 Problemas

Questão 1. Escreva um código Scilab que gere uma tabela de cosseno e seno em função de um ângulo em graus no intervalo de 0 a 90 graus.

original era, portanto, um problema relacionado ao *hardware* e letal para o inseto.

Questão 2. Use a função "linspace" para gerar um vetor de 10 elementos iniciando com o valor 5 e terminando com o valor 15. O que faz a função "logspace"?

Questão 3. Usando o comando "plot", faça o gráfico da função $y(t) = e^{-t/2}\cos(t) + \sqrt{t}$, no intervalo $(0,6)$.

Questão 4. Faça o gráfico da função $f(t) = \cos(e^{-|t|/2})$ para $-5 \leq t \leq 5$.

Questão 5. A função "sinc" é definida por

$$\text{sinc}(t) = \frac{\text{sen}(\pi t)}{t}$$

faça o gráfico desta função para $-8 < t < 8$. Use alguma estratégia para evitar a divisão por zero.

Questão 6. Usando o comando "rand", podemos gerar uma matriz $N \times M$ com valores pseudo-aleatórios. Gere uma matriz 5×5 e calcule a soma dos elementos da sua diagonal principal. O comando "diag" pode ser útil.

Questão 7. Considere a questão anterior. Gere um vetor com os elementos da matriz, use o comando "matrix". Ordene esse vetor com o comando "gsort".

Questão 8. Encontre graficamente o máximo da função $y(t) = e^{-t/3}\sqrt{t}$ para $t > 0$.

Questão 9. Considere o seguinte algoritmo a seguir e escreva o *script* Scilab correspondente.

(a) entre com um número natural maior que dois;
(b) se for um número par, divida-o por dois e subtraia um; se for um número ímpar, multiplique por cinco e some um;
(c) escreva o resultado na tela;
(d) se resultado for maior que 1 e o número de iterações for menor que 999, volte ao passo "(b)", caso contrário, termine.

Questão 10. Calcule a soma da sequência $S_n = 1 - 1/3 + 1/5 - 1/7 + 1/11 - 1/13 + \ldots + \frac{(-1)^n}{2n+1}$. Verifique que $S_n \to \pi/4$ quando n cresce. Podemos calcular o valor de π usando várias outras expressões.

Questão 11. Execute o código a seguir. O que ele faz? Comente o código apropriadamente.

```
t=0:0.01:6;
y=exp(-t/2).*sqrt(t);
figure; plot(t,y);
[ym, pm] = max(y);
tm = t(pm);
plot(tm,ym,'ro');
xgrid;
```

Questão 12. Leia e interprete o código abaixo. O que ele faz?

```
[X,Y]=meshgrid(-3:.05:3,-3:.05:3);
Z=sin(3*exp(-(X.^2+Y.^2)));
xtitle('$ Z=\sin(3exp^{-(X.^2+Y.^2)}) $');
mesh(X,Y,Z); xgrid();
```

Questão 13. Crie uma função Scilab que receba 4 valores reais como parâmetros de entrada e gere como saídas o maior e a média desses valores de entrada.

Questão 14. A potência de um sinal senoidal $x(t) = A\cos(\omega t)$ é $P_x = A^2/2$. Gere um sinal senoidal e use o comando "variance" para estimar a sua potência. Verifique que o valor é realmente próximo à amplitude ao quadrado divido por dois.

Questão 15. Considere a função $f(x) = x^3 + 5 + e^{-x^2/2}$. Essa função apresenta um zero real, isto é, existe um valor ξ tal que $f(\xi) = 0$. Faça o gráfico desta função e estime o valor de ξ.

Questão 16. Escreva um código Scilab que calcule uma aproximação para o seguinte somatório:

$$S_n = \sum_{n=1}^{\infty} \frac{1}{n^2}$$

Questão 17. Um circuito elétrico formado por um resistor R, um capacitor C e um indutor L todos em paralelo (ver Figura 1.16). Nesse circuito, a impedância Z (Ω) é expressa por:

$$\frac{1}{Z} = \sqrt{\frac{1}{R} + \left(\omega C - \frac{1}{\omega L}\right)^2}$$

Esboce o gráfico $\omega \times Z$ para $R = 50\Omega$, $C = 1\mu F$, $L = 1mH$. Qual o valor máximo de Z?

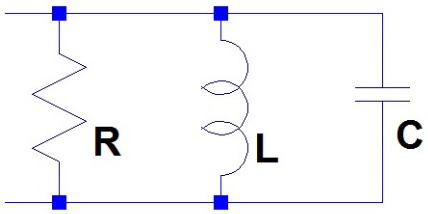


Figura 1.16: Circuito RLC paralelo - Questão 17.

Questão 18. Verifique se

$$\gamma_N = -\ln(N) + \sum_{n=1}^{N} \frac{1}{n}$$

converge para algum valor quando $N \to \infty$. Se convergir, qual é esse valor?

Questão 19. Podemos aproximar a função $\cos(x)$, quando $|x|$ é *pequeno* por

$$\cos(x) \cong 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

faça o gráfico de $\cos(x)$ e da aproximação acima no intervalo $-\pi < x < \pi$.

Questão 20. Um filtro digital discreto apresenta a seguinte equação diferença

$$y[k] - (1/3)y[k-1] + (1/5)y[k-2] = x[k] + 2x[k-1] + x[k-2]$$

Considere $y[-1] = y[-2] = 0$. Esboce o gráfico de barra $y[k]$ se o sinal de entrada é $x[k] = 2sen(\pi k/6)$, com $0 \leq k \leq 36$.

Questão 21. Um filtro digital discreto apresenta a seguinte equação diferença

$$y[k]-(1/3)y[k-1]+(1/5)y[k-2]=x[k]-\sqrt{3}x[k-1]+x[k-2]$$

Considere $y[-1]=y[-2]=0$. Esboce o gráfico de barra $y[k]$ se o sinal de entrada é $x[k]=2sen(\pi k/6)$, com $0\leq k\leq 36$. Houve alguma mudança significativa em relação à questão anterior?

Questão 22. Você pode encontrar facilmente a cotação do dólar nas últimas semanas nas páginas financeiras dos jornais eletrônicos ou em papel. Faça uma compilação desses dados e escreva um programa Scilab que calcule os valores máximo, médio e mínimo da cotação, além de fazer um gráfico dessa cotação.

Questão 23. A fórmula de Stirling é capaz de apresentar uma boa aproximação para o fatorial de um número N quando N é grande: $N!\cong\sqrt{2\pi}N^{N+1/2}e^{-N}$. Calcule e escreva uma tabela de três colunas mostrando o fatorial de 10 a 20, usando a fórmula de Stirling e o erro entre o valor exato e o valor aproximado. A função "gamma" pode ser útil.

Questão 24. Uma variável aleatória de Poisson modela fenômenos aleatórios contáveis em um intervalo de observação. Por exemplo, a taxa de bits errados em uma comunicação digital por um canal ruidoso é de λ erros por segundo. Em um certo intervalo de observação T, o número de solicitações segue uma distribuição de Poisson. Nesta distribuição $\alpha=\lambda T$ é o número médio de ocorrências do evento no intervalo de tempo considerado. A probabilidade de ocorrência do evento de Poisson é dada por:

$$p_x(x)=\frac{\alpha^x}{x!}e^{-\alpha}$$

com $x=0,1,2,...,12$ e $\alpha>0$. Faça um gráfico de barras para $p_x(x)$ quando $\alpha=3$. Qual o valor médio de $p_x(x)$?

Questão 25. Veja o código

```
close;
t=[0:0.05:6.5]';
z=exp(2.5*sin(1.5*t)*cos(t')-0.5);
plot3d(t,t,z);
```

observe que "t" é um vetor, mas "z" é uma matriz. Qual o formato do gráfico gerado?

Questão 26. Podemos expressar a função erro por:

$$erf(x)=e^{-x^2}\frac{2}{\sqrt{\pi}}\sum_{k=0}^{\infty}\frac{2^k x^{2k+1}}{1\cdot 3\cdot 5\cdots(2k+1)}$$

Essa função é implementada no Scilab como "erf()". Use a expressão acima para calcular $erf(1)$ (considere uma quantidade razoável de parcelas para o somatório) e compare com o valor dado pelo Scilab.

Questão 27. Existem muitas fórmulas para o cálculo do valor de π, entre elas:

$$\begin{aligned}\frac{\pi}{4}=&4\left(\frac{1}{5}-\frac{1}{3\times 5^3}+\frac{1}{5\times 5^5}-\cdots\right)\\&-\left(\frac{1}{239}-\frac{1}{3\times 239^3}+\frac{1}{5\times 239^5}-\cdots\right)\end{aligned}$$

Calcule o valor de π usando os 6 termos mostrados acima. Compare com o valor "exato" de π.

Questão 28. Um algoritmo para cálculo de π muito eficiente é expresso por (Press et al, 2011):

$$X_0 = \sqrt{2}$$
$$Pi_0 = 2 + \sqrt{2}$$
$$Y_0 = \sqrt[4]{2}$$

Laço:

$$X_{n+1} = \frac{1}{2}\left(\sqrt{X_n} + \frac{1}{\sqrt{X_n}}\right)$$
$$Pi_{n+1} = P_n\left(\frac{X_{n+1}+1}{Y_n+1}\right)$$
$$Y_{n+1} = \frac{Y_n\sqrt{X_{n+1}} + \frac{1}{\sqrt{X_{n+1}}}}{Y_n+1}$$

Implemente esse algoritmo em Scilab e verique que poucas iterações são necessárias para chegar na precisão numérica disponível.

Questão 29. A norma de Frobenius de uma matriz $N \times M$ é definida por

$$||M||_F = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}|a_{ij}|^2}$$

Escreva um código Scilab para calcular a norma de Frobenius de uma matriz 4×3 formada por números (pseudo) aleatórios com distribuição normal ou gaussiana.

Questão 30. Observe a curva elíptica: $y^2 = x^3 - 8x + 5$. Faça o gráfico de $x \times y$ para o intervalo $-5 < x < 5$. Note que y irá assumir valores complexos.

Questão 31. Considere as funções $f_1(x) = 2^x - 1$ e $f_2(x) = \sqrt{1+2x}$. Em qual ponto (x,y) as funções $f_1(x)$ e $f_2(x)$ se interceptam? Sugestão: faça os gráficos dessas funções.

Questão 32. As figuras ou curvas de Lissajous são gráficos obtidos por um sistema de equações paramétricas

$$x = A\sin(\omega_1 t + \phi_1)$$
$$y = B\sin(\omega_2 t + \phi_2)$$

que descreve um complexo movimento harmônico. Varie os parâmetros ω_1 e ω_2 e gere algumas dessas curvas. Observe que o tempo t precisa ter duração suficiente para gerar um ciclo completo.

Questão 33. Use o Scilab para mostrar que a série infinita

$$S = \sum_{k=0}^{\infty}\frac{1}{(2k+1)(2k+2)}$$

converge para $\ln(2)$.

Questão 34. Calcule o somatório abaixo:

$$S_n = \sum_{k=0}^{\infty}\frac{1}{(2n+1)^2}$$

Mostre que $S_n \to \frac{\pi^2}{8}$ quando $n \to \infty$.

Questão 35. Veja o código abaixo. Execute-o. O que ele faz? Qual a função do comando "waitbar"?

```
JanH=waitbar('Este é um exemplo.');
realtimeinit(0.3);
for j=0:0.1:1,
  realtime(3*j);
  waitbar(j,JanH);
end
close(JanH);
```

Questão 36. Considere uma sequência de Fibonacci. Mostre a relação F_{n+1}/F_n converge para $(\sqrt{5}+1)/2$ quando $n \to \infty$.

Questão 37. Graficamente, resolva de forma aproximada a equação

$$\frac{1}{5x}e^{-x^2} = 10^{-3}$$

Dica: faça o gráfico da função acima no intervalo de x variando de 1 a 4 com um passo *pequeno*.

Questão 38. Faça o gráfico da função racional

$$f(x) = \frac{1+x+x^2}{1-(5/6)x-(1/3)x^2+(1/6)x^3}$$

no intervalo $(-2,5)$. Use alguma estratégia para evitar divisão por zero.

Questão 39. Considere a função

$$f(x) = \cos\left(\pi\frac{1+x^2}{1-x^2}\right)$$

Esboce seu gráfico no intervalo $(-5,5)$. Essa função é periódica?

Questão 40. Analisando o gráfico da função

$$f(t) = 2te^{-t/3}$$

no intervalo $(0,6)$, encontre o seu valor máximo e o valor para o qual esse máximo ocorre.

Questão 41. Considere o seguinte algoritmo (do problema $2n+1$ ou conjectura de Collatz)[6]:

(a) escolha um número natural maior que zero;
(b) se for um número par, divida-o por dois; se for um número ímpar, multiplique por três e some um;
(c) se o número for maior que 1, volte ao passo (b), caso contrário, termine.

Qual a sequência numérica obtida quando escolhemos o número 27 como valor inicial? Escreva um código Scilab para resolver esse problema.

Questão 42. Os números primos são importantes na análise matemática. Faça um programa Scilab capaz de mostrar todos os números primos entre 3 e 10.000. Leia algo sobre o Crivo de Eratóstenes[7] antes de escrever o seu *script*. Existem quantos números primos entre 100 e 1.000?

[6] No meu *blog* você pode encontrar alguma informação adicional sobre esse problema ainda em *aberto* na Matemática. *Link*: http://alfanumericus.blogspot.com.br/2014/05/problema-3n1-ou-conjectura-de-collatz.html

[7] Eratóstenes de Cirene (Cirene, 276 a.C. - Alexandria, 194 a.C.) foi um matemático, gramático, poeta, geógrafo, bibliotecário e astrônomo da Grécia Antiga, mais conhecido por ter calculado a circunferência da Terra com um erro pequeno.

Questão 43. O conceito de massa relativística é importante quando estamos estudando corpos (principalmente partículas) viajando próximo à velocidade da luz. Sendo m_0 a massa de repouso de uma partícula. A massa, quando esta partícula se move, é expressa por:

$$m = \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}$$

onde v é a velocidade da partícula e c a velocidade da luz no vácuo. Faça um gráfico de $m \times v$.

Questão 44. Veja o código abaixo. O que ele faz? Quanto tempo demora para o Scilab gerar dois milhões de números "aleatórios" e, em seguida, colocá-los em ordem decrescente?

```
tic();
x=rand(1,2e6,'n');
y=gsort(x);
toc()
```

Questão 45. Faça o gráfico da função $f(x) = (x^2 - 4x + 3)^4 - 1/2$. Quais as raízes dessa função?

Questão 46. Execute o código abaixo. Observe a criação de um certo padrão no gráfico.

```
t = 0:0.01:100;
s = exp(-t/8).*sin(t);
c = exp(-t/10).*cos(2*t/3);
plot(s,c);
```

Questão 47. Usando o comando "roots", calcule todas as raízes da função polinomial

$$f(x) = x^5 - 6x^4 + 3x^2 + 9x - 8$$

Questão 48. Com o auxílio do comando "convol", calcule o seguinte produto polinomial: $(x^3 + 1{,}5x^2 - 3{,}3) \times (x^4 - 2{,}5x + 1{,}4)$.

Questão 49. Podemos calcular aproximadamente o valor de π^2 pela série indicada abaixo. Crie um código Scilab que efetue esse trabalho.

$$\frac{\pi^2}{8} \cong \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \cdots$$

Questão 50. Usando o comando "rand" para gerar 10.000 números pseudo-aleatórios com distribuição normal, determine os valores máximos, mínimos e o valor médio e a variância (comando "variance"). Rode o seu código diversas vezes, existe alguma variação de valores?

Questão 51. Refaça a questão anterior, mas inclua o comando "rand('seed',22)" antes de chamar o comando "rand". Rode o código algumas vezes. O que ocorre agora?

Questão 52. Considere a função $f(t) = 2te^{-t/2}\cos(3t)$, no intervalo $0 \leq t \leq 5$. Determine graficamente, de forma aproximada, quando a função atinge o valor máximo e em que instante de tempo isso ocorre.

Questão 53. Faça uma estimativa da área sob a curva da função $f(x) = \sqrt{1-x}$ com o eixo x no intervalo $0 \leq x \leq 1$.

Questão 54. Considere a afirmação: "dados dois números reais positivos, a média geométrica é sempre menor que a média aritmética". Essa afirmação está correta? Gere uma centena de pares de números e verifique se essa afirmação está correta.

Questão 55. Execute o seguinte código e observe o gráfico gerado:

```
close;
t=0:0.01:60;
c=cos(2*t);
s=exp(-t/5).*sin(4*t);
figure; plot(c,s);
```

Questão 56. Estime graficamente a solução de $15 - x^3 = e^{x/2} - 1$.

Questão 57. Execute o código abaixo:

```
x = 2;
df = 1;
k = 1;
while abs(df)>0.00001
    xe = x;
    x = (16 - exp(x/2))^(1/3);
    df = xe - x;
    k = k + 1;
    disp([k, x, df]);
end;
```

Mostre que o código acima consegue resolver a questão anterior.

Questão 58. Quais os números primos maiores que 1000 e menores que 2000? Dica: use o comando *primes*.

Questão 59. Quais os próximos números da Mega Sena? Faça um programa Scilab que gere seis números "aleatórios" entre 1 e 60. E boa sorte!

2. Modelagem matemática

Uma acumulação de fatos não faz uma ciência, tal como um conjunto de pedras não faz uma casa.
Jules Henri Poincaré (1854 - 1912), matemático, físico e filósofo da ciência.

Neste capítulo, apresentaremos alguns conceitos fundamentais de computação, modelagem matemática e como os problemas podem ser solucionados de forma numérica. Com o uso cada vez mais difundido dos computadores pessoais, os laboriosos cálculos que levavam dias ou semanas (ou uma vida inteira) para serem resolvidos, agora, são solucionados em questão de segundos. Esse poder computacional permite que problemas que antes eram insolúveis, agora tenham pelos menos uma solução numérica de alta precisão.

2.1 Modelo matemático

Podemos dizer que um **modelo matemático** é uma forma (quase sempre com algum grau de simplificação) de representar um cenário ou fenômeno da natureza, ou de um sistema construído pelo homem. Em geral, como os sistemas podem ser muito complexos, alguns detalhes menos importantes são desprezados. Logo, a solução de um modelo pode, em casos extremos, não ser compatível com a realidade. A construção de um modelo exige um equilíbrio entre detalhes relevantes e simplicidade geral. Os modelos são usados em muitas áreas da ciência e engenharia: biologia, ecologia, matemática aplicada, física, química, engenharia elétrica, mecânica ou civil, e muitas outras.

Dessa forma, um problema vindo de uma determinada área de interesse é convertido em uma equação matemática ou modelo algorítmico. Sendo um pouco mais formal, podemos dizer que um modelo matemático pode ser expresso por:

$$var\_dependente = f\,(var\_independentes, parâmetros, termos\ forçantes)$$

Uma vantagem de usarmos um modelo matemático é a facilidade em ajustar os parâmetros até que determinada resposta seja obtida. O custo de alteramos os parâmetros do modelo é muito menor

que ajustar um protótipo, além de ocorrer o risco de algum tipo de acidente. Por exemplo, quase sempre antes de montar e ligar um circuito eletrônico, o engenheiro eletrônico recorre a *software* de simulação, onde os componentes são modelados por equações matemáticas. Se o modelo em *software* funcionar, as chances do circuito real operar satisfatoriamente são muito grandes.

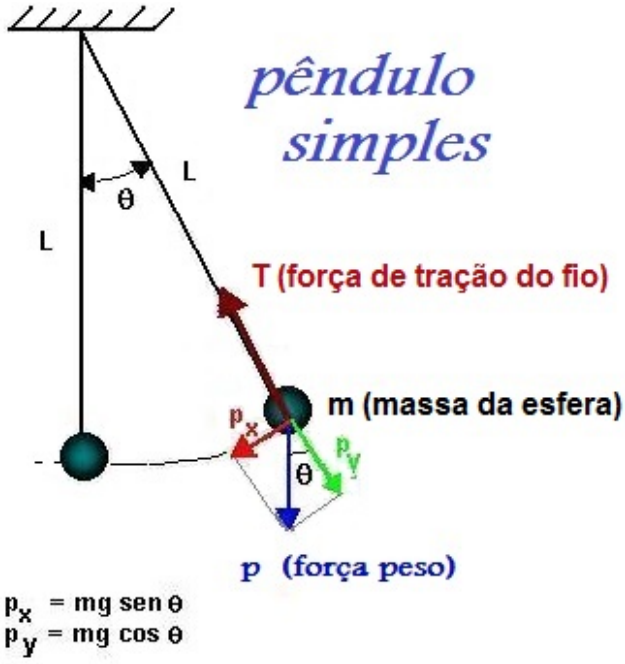


Figura 2.1: Pêndulo simples oscilante

Vejamos um exemplo mais concreto de um modelo matemático. Considere o modelo de um pêndulo simples:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L} sen(\theta) = 0 \tag{2.1}$$

A equação (2.1), obtida pela aplicação da segunda lei de Newton, descreve o ângulo de deslocamento $\theta(t)$ de um pêndulo de comprimento L sob a ação da força de gravidade g próximo à superfície da Terra, sem levar em conta o atrito com o ar, ver Figura 2.1.

A equação (2.1) é uma equação diferencial ordinária (EDO) de segunda ordem. Essa EDO, mesmo tendo aparência simples, é não linear (termo $sen(\theta)$) e não possui uma solução analítica fechada. Uma forma de contornar esse tipo de problema é com a linearização: uma simplificação no modelo que permite a existência de uma solução analítica. Nesse caso, notando que para pequenos ângulos ($\theta < 15^o$), isto é, quando o pêndulo está próximo da posição de equilíbrio, $sen(\theta) \cong \theta$. Logo, a equação (2.1) pode ser reescrita como:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0 \tag{2.2}$$

A equação (2.2), considerando que $\theta = \theta_{max}$ e $d\theta/dt = 0$ em $t = 0s$, possui a seguinte solução analítica:

$$\theta(t) = \theta_{max} \cos\left(\sqrt{\frac{g}{L}}t\right) \tag{2.3}$$

A linearização pode ser muito útil e aplicável em diversos casos práticos, entretanto, quando não for possível usá-la, é necessário recorrer a métodos numéricos para a solução dos problemas.

Vejamos um segundo problema. A corrente elétrica (i_D) em um diodo de junção de silício segue uma lei exponencial:

$$i_D = I_S(e^{\frac{v_D}{nV_T}} - 1) \tag{2.4}$$

sendo I_S a corrente de saturação (da ordem de nA), v_D a tensão direta aplicada, n um fator entre 1 e 2, $V_T \cong 26mV$ depende da temperatura ambiente. Para um circuito simples formado por uma fonte CC de tensão, um resistor e um diodo $1N4148$ de silício (com parâmetros: $I_S = 2,52nA$, $n = 1,752$) diretamente polarizado (ver Figura 2.2), segue que a corrente que flui no circuito pode ser equacionada como:

$$\frac{Vcc - v_D}{R} = I_S(e^{\frac{v_D}{nV_T}} - 1) \tag{2.5}$$

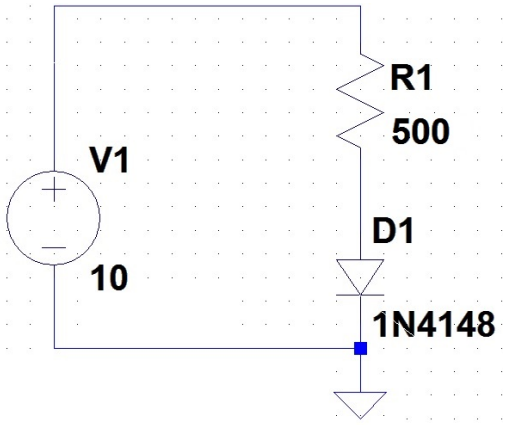


Figura 2.2: Circuito eletrônico com resistor e diodo

A equação (2.5) não pode ser resolvida de forma trivial, pois o cálculo de v_D envolve uma função exponencial e uma função linear. Esse circuito simples forma um sistema não linear. É necessária alguma estratégia numérica para a solução desse problema. Ou o uso de algum tipo de linearização que simplifique o problema. Uma simplificação muito usada é considerar que a tensão do diodo seja igual a 0,7 volt quando o diodo está diretamente polarizado, sendo a corrente limitada pelo resistor. Nesse caso,

$$\begin{aligned} i_D &\cong \frac{10 - 0,7}{500} \\ &\cong 18,6mA \end{aligned}$$

a solução *exata* do circuito (via simulador de circuitos eletrônicos que inclui ainda outros efeitos) leva aos valores *corretos* $i_D = 18,546mA$ e $v_D = 0,727volt$. A diferença entre o valor aproximado e o valor exato é, neste caso, realmente pequena e poderia ser ignorada em uma primeira aproximação.

2.2 Sobre Sistemas

Um conceito de sistema: entidade composta por partes interconectadas que recebe um ou mais sinais (discretos ou contínuos), processa-os de alguma forma e gera uma ou mais saídas. Os sinais de entrada podem ser grandezas físicas (tensão elétrica, força mecânica, ou de outro tipo) ou sequências de números que podem modificar o estado atual do sistema. Normalmente, podemos representar um sistema por uma relação matemática que relaciona entrada com saída ou por algum tipo de algoritmo. Na Figura 2.3 temos um exemplo de sistema mecânico (linear) e os principais parâmetros envolvidos, já na Figura 2.4 vemos um exemplo de modelo não linear, no caso um sistema amplificador com saturação (normalmente, essa é uma situação que devemos evitar em eletrônica).

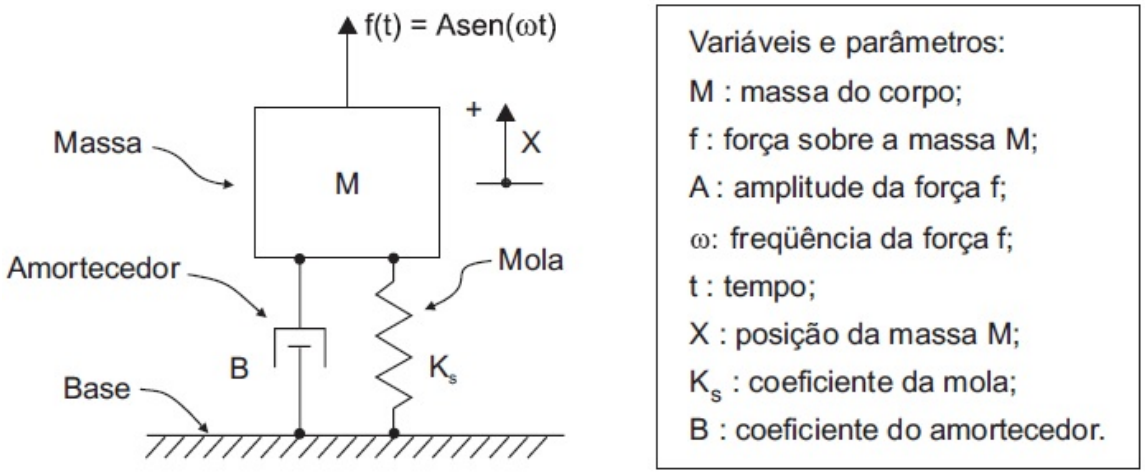


Figura 2.3: Exemplo de sistema mecânico, a entrada é a força $f(t)$. Adaptado de Felício (2010).

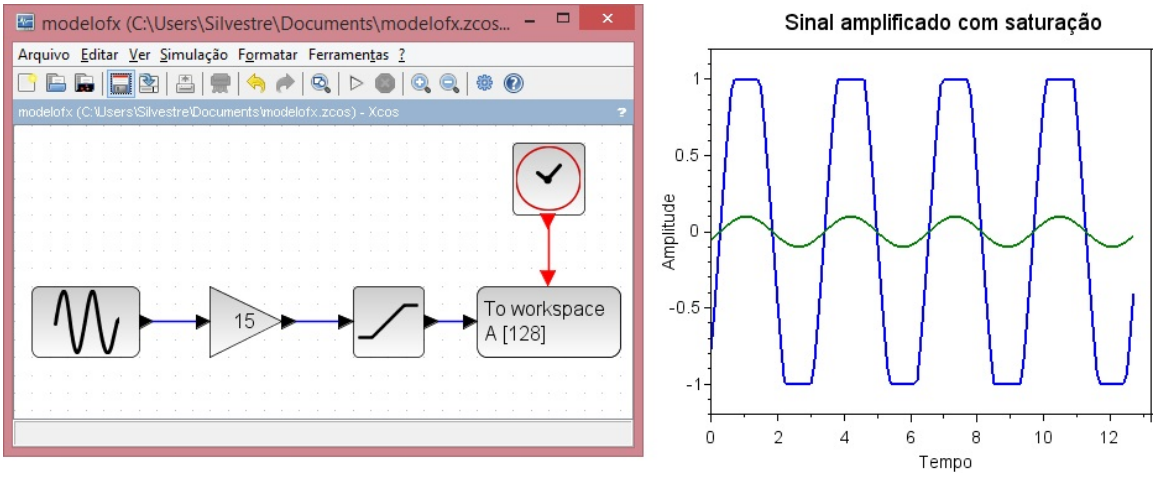


Figura 2.4: Exemplo de sistema não linear devido à saturação da saída. Uso do Xcos - ambiente de simulação do Scilab.

Existe uma grande variedade de sistemas. Os sistemas podem ser contínuos, isto é, suas entradas

e saídas são sinais contínuos no tempo ou discretos, ou seja, os sinais de entrada e saída são sequências de valores contáveis. Analisar um sistema significa encontrar uma relação entre a entrada e a saída, em geral, uma equação diferencial (sistemas contínuos) ou uma equação diferença (sistemas discretos). Sendo conhecida a entrada, é possível calcular a saída pela solução da equação diferencial ou diferença.

Os sistemas podem ser lineares[1] ou não lineares. Se o sistema for linear é possível, pelo menos teoricamente, encontrar analiticamente a solução. Para o caso de sistemas não lineares, não temos regras gerais de solução e, quase sempre, a solução é numérica.

2.3 Solução numérica

Quando não existe uma solução analítica ou ela é muito trabalhosa, podemos usar alguma outra estratégia para resolver a equação. Não existe uma solução analítica da equação (2.5), mas ela pode ser resolvida numericamente, por exemplo, com o código Scilab (ver Tabela 2.1) cuja convergência é rápida. Note que a solução obtida iterativamente é praticamente a mesma obtida pelo *software* de simulação de circuitos eletrônicos (LTspice IV, versão 4.23).

Vejamos agora um exemplo mais *puramente* matemático: qual a raiz da equação $f(x) = x^3 - 7x^2/2 + 3x - 5/4$? É fácil perceber que $f(2) < 0$ e que $f(3) > 0$. Logo, em algum ponto no intervalo $(2,3)$ a função $f(x)$ deve ter pelo menos um zero. Como calcular esse zero da função? Novamente, podemos usar um algoritmo para ir encontrando valores cada vez mais próximos da raiz:

$a)$ $x_{k+1} = \sqrt[3]{\dfrac{7x_k^2}{2} - 3x_k + \dfrac{5}{4}}$

$b)$ $erro = |x_{k+1} - x_k|$

$c)$ Se $erro > tolerância$ voltar ao passo $a)$

Ao rodar esse algoritmo com o valor inicial $x_0 = 2$, obtemos os valores listados na Tabela 2.2. Podemos observar que a convergência desta vez é muito lenta. Em um capítulo próximo, estudaremos a taxa de convergência desses algoritmos.

Um último exemplo de modelo que não pode ser resolvido analiticamente: o famoso *problema dos três corpos*[2]. Usando a lei de gravitação universal de Newton, podemos determinar o movimento dos planetas ao redor do Sol, mas, nesse cálculo, fazemos uma simplificação: consideramos as iterações entre os planetas não existe, somente a força entre o Sol e o planeta é considerada. Já no problema dos três corpos, estamos interessados em determinar quais são os comportamentos possíveis desses três corpos que interagem entre si através de uma força gravitacional newtoniana, sem considerar que algum deles tem massa desprezível em relação aos demais. Este problema é relativamente fácil de equacionar, porém, o fato é que é impossível de obtermos uma solução analítica. Devemos nos contentar em recorrer a uma solução numérica.

[1] Um sistema é linear se ele pode ser representado por uma ou um conjunto de equações diferenciais ordinárias lineares com coeficientes constantes.

[2] É o estudo do movimento de três corpos de massas arbitrárias, m_1, m_2 e m_3, em movimento por ação exclusiva da força de atração Newtoniana entre cada par de corpos

Tabela 2.1: *Script* Scilab e resultados

```
Is = 2.52e-9;
n = 1.752;
VT = 26.25e-3;
nVT = n*VT;
R = 500;
Vcc = 10;
vd = 0.8;
k1 = 1/nVT;
k2 = 1 + Vcc/(R*Is);
k3 = -1/(R*Is);
erro = 1;
while erro > 1e-5
                    vk = log(k2 + k3*vd)/k1;
                    erro = abs(vd - vk);
                    vd = vk;
                    id = Is*(exp(vd/(nVT)) - 1);
                    disp([vd, id*1000]);
end
```

Resultados:

v_D *volt*	i_D *mA*
0,7268077	18,400000
0,7271721	18,546385
0,7271703	18,545656
0,7271703	18,545659

2.4 Alguns conceitos e definições

Para finalizar este capítulo[3], vejamos alguns conceitos e definições. Os métodos numéricos são bem antigos, não é uma invenção recente. Eles surgiram já nas primeiras civilizações com a nascente economia e os registros contábeis. Os babilônios e egípcios, muitos séculos antes de Cristo, já usavam frações, tinham tabelas com os quadrados dos números menores que 60 e regras para encontrar as raízes de equações. Arquimedes de Siracusa (278-212, a.C.) conseguiu mostrar, usando o método da exaustão[4], que

$$3\frac{10}{71} < \pi < 3\frac{1}{7}$$

vários outros matemáticos (Pascal, Newton, Leibniz, Euler, Lagrange, Gauss, Simpson, Runge, Kutta, ...) também contribuíram com essa área.

[3]Como o objetivo principal deste livro é apresentar conceitos de métodos numéricos, não teremos uma seção de problemas sobre sistemas ou modelos muito extensa.

[4]O método da exaustão, quando aplicado à geometria, é um método para se encontrar a área de uma figura inscrevendo-se dentro dela uma sequência de polígonos cuja soma das áreas converge para a área da figura desejada.

Tabela 2.2: Calculando a raiz de $f(x) = x^3 - 7x^2/2 + 3x - 5/4$

k	erro	x_k
1	—-	2
2	0,0991682	2,0991682
3	0,0818861	2,1810543
4	0,0666853	2,2477396
–	—-	—-
47	0,0000013	2,4999957
48	0,0000010	2,4999967

Definição 2.4.1 Método numérico. Podemos dizer que é um conjunto de ferramentas ou métodos usados para se obter a solução de problemas matemáticos de forma aproximada. Esses métodos se aplicam principalmente a problemas que não apresentam uma solução exata, sendo necessária uma resolução numérica usando um computador (*hardware*) e um algoritmo (*software*), pois, quase sempre, as soluções são iterativas.

Já o conceito de algoritmo, tal como usamos hoje, vem do computador analítico de Charles Babbage (1791-1871) e da sua primeira programadora, Ada, condessa de Lovelace (1815-1852). Contudo, somente com a construção dos primeiros computadores eletromecânicos na década de 1940 é que os cientistas e engenheiros começaram a ter acesso a dispositivos de cálculo que eram capazes de rodar seus algoritmos numéricos[5].

Definição 2.4.2 Algoritmo. Podemos apresentar várias definições para algoritmo. Uma definição de trabalho é: uma lista de procedimentos bem definidos, na qual as instruções são executadas passo a passo a partir do começo da lista. O algoritmo deve conter todas as instruções necessárias para a solução de um problema. Um algoritmo pode ser facilmente visualizado através de um fluxograma.

Algoritmos em geral e numéricos em especial precisam ter as seguintes propriedades ou características:

- serem livres de erros lógicos;
- tratamento para erros operacionais (*overflow/underflow*);
- tempo de execução finito, isto é, deve apresentar algum critério de parada;
- independência da máquina usada;
- eficiência visando a economia de recursos (memória, tempo).

2.5 Problemas

Questão 1. A sua versão do Scilab tem o valor do π correto até qual casa decimal?

Questão 2. Você recebeu uma xícara de chá muito quente (temperatura de uns 90^oC), então, você precisou esperar 10 minutos para a temperatura chegar a um valor mais confortável (uns 40^oC). A temperatura ambiente é de $T_a = 20^oC$. A Lei de Newton do resfriamento é expressa por:

$$\frac{dT}{dt} = K(T - T_a)$$

[5] Recomendamos a leitura do livro de W. ISAACSON, Inovadores.

encontre a constante K para este exemplo.

Questão 3. O corpo humano libera cerca de 80W de potência quando em repouso. Considere uma sala fechada e selada (isolada) com dimensões $8m \times 6m\ times 3m$, com temperatura inicial de 20^oC e ocupada por 20 pessoas. Qual a variação de temperatura nesse ambiente nos primeiros 30 minutos? A relação entre o calor absorvido Q e a temperatura é expressa por: $Q = CM(T_2 - T_1)$, com M a massa do sistema, C a capacidade térmica e $T_2 - T_1$ a variação de temperatura. Cada pessoa ocupa cerca de $0,1m^3$ de volume. A densidade do ar é cerca $1,18kg/m^3$ e sua capacidade térmica é aproximadamente de $0,72kJ/KgK$.

Questão 4. Um sistema não linear é mostrado a seguir. Se a entrada for a tensão $v(t) = 3cos(10\pi t)$ volts, encontre a saída $y(t)$. O resistor $R = 330\Omega$, os diodos são de silício, por exemplo, o diodo de sinal 1N4148.

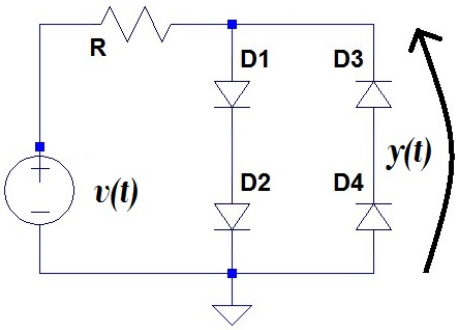


Figura 2.5: Sistema para questão 04.

Questão 5. Um homem possui R$ 1.000.000,00 em uma aplicação financeira que rende a taxa de 1,2% de juros ao mês sobre o valor aplicado. Ele pretende sacar o valor de R$ 20.000,00 ao mês, no dia seguinte aos juros aplicados. Gere um modelo financeiro para o cálculo do saldo dessa aplicação. Por quanto tempo ele poderá fazer esses saques?

Questão 6. No sistema abaixo, os parâmetros são: $R = 10\Omega$, $C = 0,1F$, $R1 = 20\Omega$ e $C1 = 0,1F$. A entrada é um pulso retangular $p(t) = 10$ volts, com duração de 1 segundo. Encontre a tensão de saída $vs(t)$. Considere que os capacitores não possuem carga inicial.

Questão 7. O crescimento populacional, em geral, é limitado for fatores tais como disponibilidade de alimentos e água, espaço físico, entre outros. Algumas vezes, a esse crescimento populacional pode ser expresso pela equação de crescimento logístico

$$\frac{dP}{dt} = kP(P_{max} - P)$$

onde k é uma constante, $P(t)$ a população, P_{max} a população máxima permitida. A partir dessa equação diferencial, podemos estimar a população pela seguinte equação diferença:

$$P(n+1) = P(n) + kP(n)(P_{max} - P(n))\Delta t$$

com $n \geq 0$ inteiro, $P(0)$ igual a um valor inicial. Escolha parâmetros que você considere razoáveis para rodar esse modelo discreto. Na Figura 2.7, temos alguns exemplos de curvas logísticas.

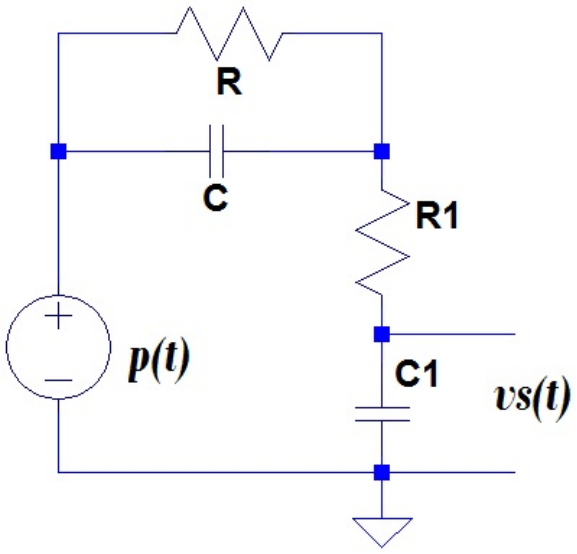


Figura 2.6: Sistema para questão 06.

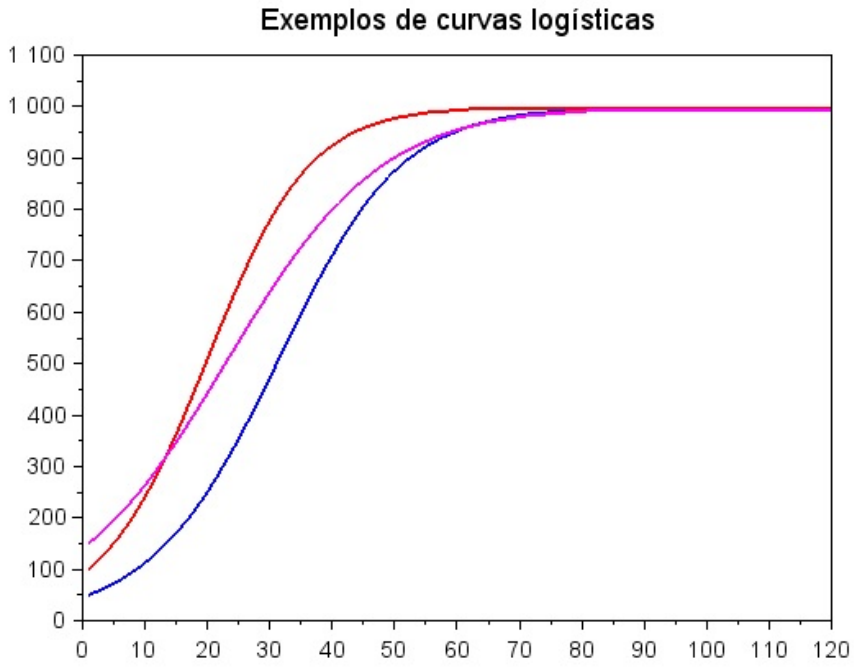


Figura 2.7: Exemplos de curvas logísticas com diferentes valores iniciais e de k, mas mesmo P_{max}.

Questão 8. Considere um projétil lançado com um ângulo θ entre 30^0 e 60^0 graus em relação à horizontal. Suponha que a velocidade inicial seja maior que 20 m/s e menor que 200 m/s. Elabore um modelo matemático para essa situação. A resistência do ar deve ser levada em conta (força de arrasto não desprezível) - proporcional ao quadrado da velocidade e contrária ao movimento. A Figura 2.8 ilustra o efeito da resistência do ar no lançamento de um projétil.

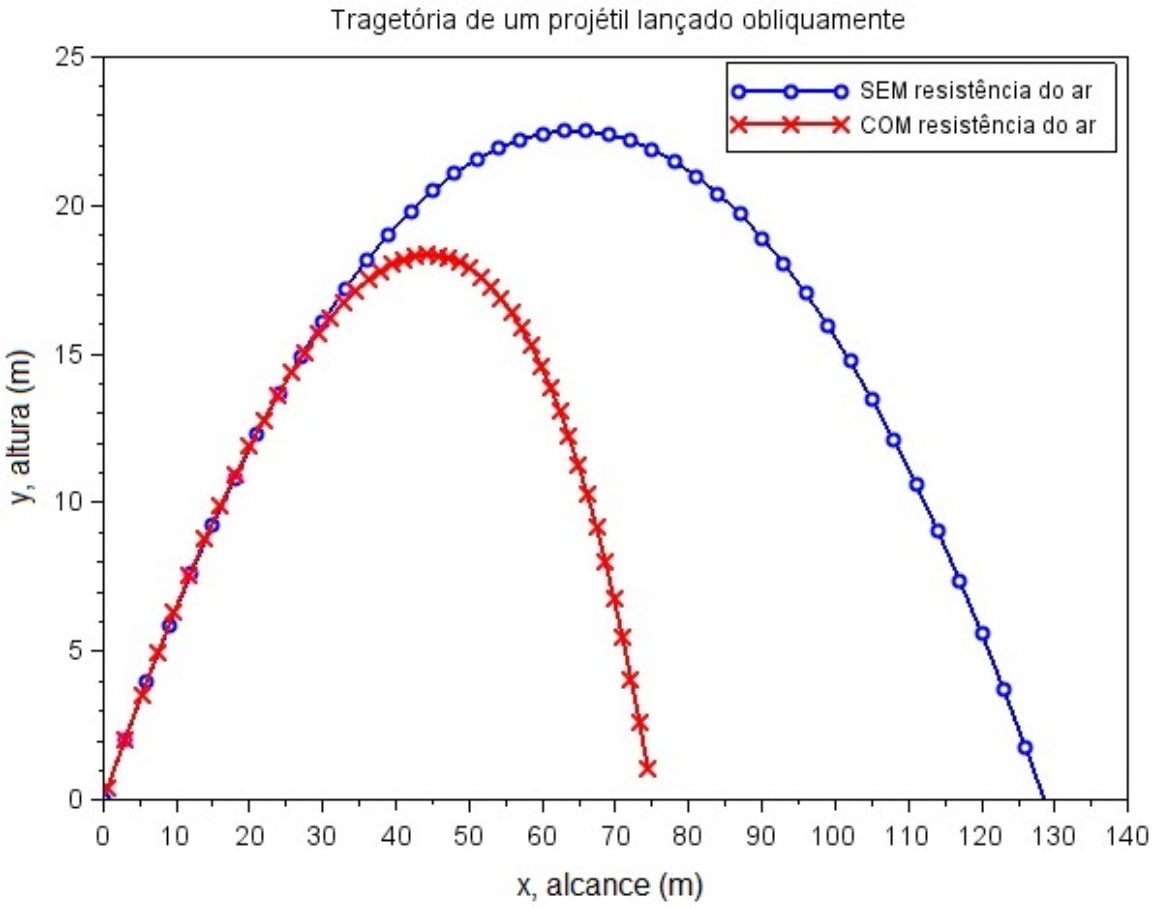
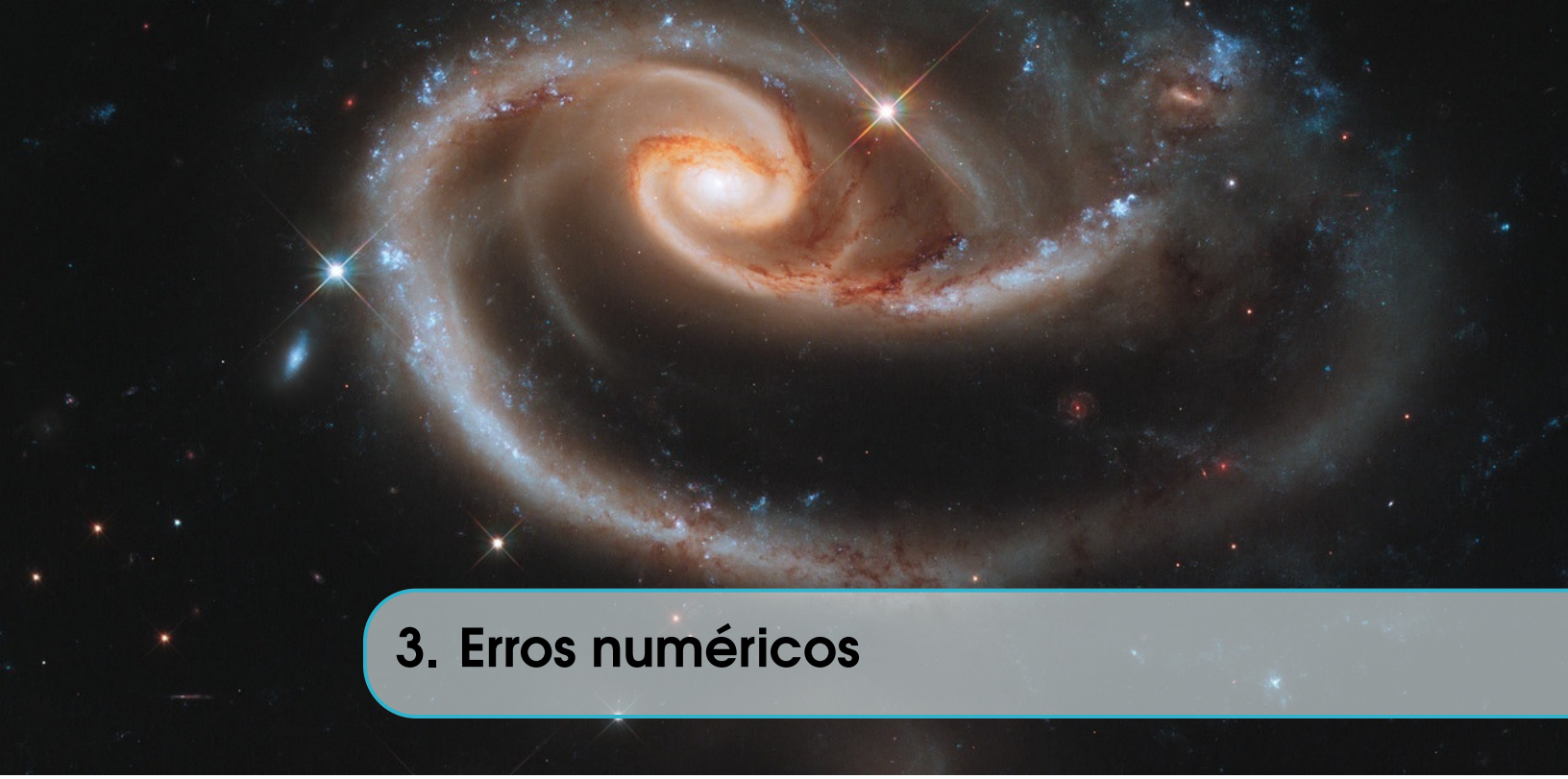


Figura 2.8: Trajetórias possíveis de um projétil.

Uma observação: a maior parte das questões deste capítulo exigem algum conhecimento de eletricidade, eletrônica ou física. Não desanime!



3. Erros numéricos

Testes de programas podem ser uma maneira muito eficaz para demonstrar a presença de erros, mas é irremediavelmente insuficiente para mostrar a sua ausência.
Edsger Wybe Dijkstra (1930 - 2002), cientista da computação holandês.

Neste capítulo veremos conceitos relacionados com erros numéricos e precisão nos cálculos de funções e como isso funciona no Scilab. Na prática, é *impossível* atingir a perfeição ou conseguirmos resultados numéricos exatos. Veremos como estimar os erros envolvidos e as limitações inerentes aos computadores. Quando falamos de erros numéricos, quase sempre eles são realmente pequenos, algo como 10^{-3} ou $0,15\%$. Mesmo assim, esses erros podem ter consequências desastrosas.

3.1 Computadores podem errar?

Usando uma calculadora comum, é provável que o resultado das operações $x = \sqrt{2}$ e, em seguida, $y = x^2$ não resulte exatamente em "2". Isso acontece porque os números são armazenados com precisão finita na memória da máquina. Devemos lembrar que entre 0 e 1 existem infinitos números. Não é possível representar *todos* esses números em uma memória finita. Esses pequenos erros podem se acumular depois de várias operações e resultar em uma resposta final realmente errada e bem distante do valor desejado. Vejamos um erro ainda mais evidente digitando diretamente no console do Scilab os seguintes comandos:

```
-->format(25)
 -->x=1/0.9
 x  =
     1.111111111111111604544
```

ora, sabemos que o valor de $1/0,9$ gera o valor $1,111111...$, mas esse não é o valor armazenado na variável "x" do exemplo acima.
Claro, além desses erros inerentes ao processo de cálculo usando computadores e calculadoras eletrônicas, falhas no algoritmo implementado, erros de digitação (trocar um "*" por ".*" ou a troca de "y2(1)" por "y1(2)"), a troca de um "OU" por um "E" em um laço condicional podem ser bem sutis e difíceis de localizar depois de horas de trabalho cansativo e levar a resultados totalmente errôneos. Nesse sentido, sim, os computadores atuais podem errar, "ajudados" em grande medida por alguém que deveria estar um pouco mais atento[1].

3.2 Inteiros no Scilab

Os números inteiros no Scilab podem ser ou não sinalizados, ocupar 1, 2 ou 4 bytes. Estes tipos de dados são especialmente úteis para armazenar objetos grandes como imagens, sinais longos ou quando estamos interessados em trabalhar somente com números inteiros. Podemos também converter números reais para inteiros:

- y=int8(X) - retorna números no intervalo [-128,127];
- y=uint8(X) - retorna números no intervalo [0,255];
- y=int16(X) - retorna números no intervalo [-32768,32767];
- y=uint16(X) - retorna números no intervalo [0, 65535];
- y=int32(X) - retorna números no intervalo [-2147483648,2147483647];
- y=uint32(X) - retorna números no intervalo [0, 4294967295];

Quando usamos esses inteiros, temos que ter algum cuidado. No Console do Scilab podemos, por exemplo, efetuar as seguintes operações indicadas na Tabela 3.1.

Tabela 3.1: Uso de int8() e uint8()

int8()	*uint8()*
–>y = int8(34)	–>y = uint8(34)
y =	y =
34	34
–> z=y*15	–> z=y*15
z =	z =
-2	254

Certamente, o valor "z = - 2" para "34*15" é uma surpresa à primeira vista; o mesmo podemos dizer de "34 * 15 = 254". Esse valores *estranhos* são consequências do formato dos números inteiros usados. Se o número está no formato "int8", então "127+1 = - 128"! Mais um exemplo desse tipo de problema: ao digitarmos "uint8(275.967)" no console, o resultado é "ans = 19". Em geral, não usamos número inteiros quando estamos trabalhando com métodos numéricos. O comando "double" (ex: Y = double(X)) converte os dados armazenados usando inteiros de 1, 2 ou 4 bytes em representação de ponto flutuante (chamaremos simplesmente *float*, por conveniência) de dupla precisão. Se a entrada já é um número de ponto flutuante de dupla precisão, nada é feito.

[1] "*Errar é humano, colocar a culpa no computador é mais humano ainda*" - autor desconhecido.

3.3 Aritmética computacional

Atualmente, o padrão IEEE (*Institute of Electrical and Electronics Engineers*) *Binary Floating Point Arithmetic Standart* 754-2008 (revisão do IEEE Std 754-1985) publicado em agosto de 2008 é usado para a realização de cálculos com números reais em um computador, seguido pela maioria dos fabricantes de *hardware*. Este padrão especifica formatos e métodos de mudança e aritmética para aritmética binária e decimal em ponto flutuante em ambientes de programação de computadores. O Scilab usa esse padrão.

O padrão IEEE 754-2008 especifica as condições de exceção e seu processamento padrão. A implementação de um sistema de ponto flutuante conforme este padrão pode ser realizada inteiramente em *software*, inteiramente em *hardware* ou em qualquer combinação de *software* e *hardware*. Para as operações especificadas na parte normativa deste padrão, os resultados numéricos e as exceções são determinados exclusivamente pelos valores dos dados de entrada, sequência de operações e formatos de destino, tudo sob o controle do usuário. Este padrão é um produto do Grupo de Trabalho de Ponto Flutuante e patrocinado pelo Comitê de Padrões de Microprocessadores da IEEE *Computer Society*.

Os números *floats* podem ser armazenados e manipulados em alguns formatos. Os formatos definidos têm 32, 64 ou 128 *bits*. Esses *bits* são distribuídos em três grupos: um *bit* de sinal, expoente ou característica (base 2) e mantissa. Assim, um *float* tem o formato $(-1)^{sinal} \times b^{expoente} \times m$, onde s é 0 ou 1, *expoente* é qualquer inteiro entre *emin* e *emax*; m é um número representado por uma sequência de bits 0 ou 1 na forma $d_0 d_1 d_2 ... d_{p-1}$. Esse formato também pode representar $-\infty$, $+\infty$, e os *não números qNaN* (*quiet not a number - silencioso*) e *sNaN* (sinalizado). O Scilab pode tratar exceções, por exemplo, uma divisão por zero, com o comando "ieee(mod)": ieee - ajusta o modo de exceção de ponto flutuante:

- ieee(0) - exceção de ponto flutuante produz um erro;
- ieee(1) - exceção de ponto flutuante produz um aviso;
- ieee(2) - exceção de ponto flutuante produz um Inf ou NaN ("infinito" ou "não é número").

As expressões $0 \times \infty$, $\infty - \infty$ e $0/0$ não tem sentido matemático. Elas são consideradas operações inválidas. A norma IEEE responde a essas operações com o resultado *NaN*.

Assim, devido a representação binária com uma quantidade finita de bits, teremos números máximos e mínimos que podem ser representados, como indicado na Tabela 3.3. Os dois códigos Scilab na Tabela 3.2 mostram que a ordem de grandeza dos números máximos e mínimos são 10^{-324} e 10^{308}. Dessa forma, a soma $1,2 \times 10^{308} + 1,3 \times 10^{308}$ resultará em "inf" (*overflow*). Já a divisão $4,94 \times 10^{-324}/2$ resulta em zero (*underflow*). Em geral, um *overflow* ocasiona algum tipo de erro e o programa pode parar, já no *underflow* o resultado é simplesmente igualado a zero.

3.4 Erros de arredondamento

Os erros de arredondamento são uma característica inerente do *hardware* do computador ou de qualquer calculadora eletrônica. Por exemplo, o número decimal "0,1" não tem uma representação binária perfeita:

$$\begin{aligned} 0,1 \cong & 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + \ldots \\ = & (0,0001100110011\cdots)_2 \end{aligned}$$

O erro cometido na representação de um valor *float* em sua representação binária é chamado de erro de arredondamento, mesmo que o valor tenha sido simplesmente truncado em certo número

Tabela 3.2: Números máximo e mínimo no Scilab

Mínimo	*Máximo*

x = 1e-20;
while x>0 do
  pmin = x;
  x = x/2;
end
disp(pmin);
	x = 1e20;
	while (isinf(x)==%f) do
	  pmax = x;
	  x = x*2;
	end
	disp(pmax)

4.94D-324	1.21D+308

Tabela 3.3: Formato IEEE de ponto flutuante (base 2)

Propriedade	*Single*	*Double*	*Extend*
total de bits	32	64	80
mantissa - bits	23	52	64
expoente - bits	8	11	15
expoente máximo	127	1023	16383
expoente mínimo	-126	-1022	-16382
maior número	$\approx 3,40 \times 10^{38}$	$\approx 1,80 \times 10^{308}$	$\approx 1,19 \times 10^{4932}$
menor número	$\approx 1,18 \times 10^{-38}$	$\approx 2,23 \times 10^{-308}$	$\approx 3,36 \times 10^{-4932}$
dígitos decimais	7	16	19

de bits. Quando efetuamos a subtração de dois valores próximos, podemos cometer um erro de arredondamento significativo. Quando somamos um número "grande" com um "pequeno", também forçamos um erro de arredondamento significativo. Exemplos - digitando diretamente no console do Scilab:

```
–>Ng = 1e9; Np = 1e-9; Ns = Ng + Np; Df = Ns - Ng
Df = 0.
–>p10000 = %pi*10000; p9999 = %pi*9999; df = p10000 - p9999
df = 3.1415927
```

Os erros de aproximação podem ser medidos de duas formas, descritas a seguir.

Definição 3.4.1 Erro Absoluto. Se $p*$ é uma aproximação do valor p, o erro absoluto é $\varepsilon_a = |p - p*|$

Definição 3.4.2 Erro Relativo. Se $p*$ é uma aproximação do valor p, o erro relativo é $\varepsilon_r = |p - p*|/|p|$, desde que $p \neq 0$.

Em geral, é preferível usar o erro relativo, já que este método leva em conta o valor do número que está sendo aproximado. Um exemplo simples: se $p = 4,56$ e $p* = 4,48$, então $\varepsilon_a = 0,08$ e $\varepsilon_r \cong 0,0175439$ ou $\varepsilon_r \cong 1,75\%$.

3.5 Erros de truncamento

Em teoria, o cálculo de algum valor, por exemplo, o cálculo numérico de uma integral, requer um número *infinito* de operações. Na prática, temos que nos limitar a um número finito de operações. A diferença entre o valor obtido no cálculo prático e a resposta verdadeira é o erro de truncamento. Notamos que, mesmo em um computador hipotético sem nenhum erro de arredondamento, os erros de truncamentos persistiriam. Como Press et al. (2011) gostam de enfatizar: "... é apenas um leve exagero dizer que minimização inteligente de erro de truncamento é praticamente todo o conteúdo da área de análise numérica!"

3.5.1 Série de Taylor

Uma função "suave" em torno de um ponto pode ser aproximada por uma função polinomial, essa afirmação é, em essência, o que diz o Teorema de Taylor[2]. Sendo mais formal:

Teorema 3.5.1 Seja $f(x)$ uma função contínua e com suas primeiras $n+1$ derivadas também contínuas no intervalo (a,x), então o valor da função em x é dado por (Série de Taylor):

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)(x-a)^2}{2!} + \frac{f'''(a)(x-a)^3}{3!} + \ldots + \frac{f^{(n)}(x-a)^n}{n!} + R_n$$

onde o resto R_n é dado por

$$R_n = \int_a^x \frac{(x-t)^n}{n!} f^{(n+1)}(t)dt$$

sendo t uma variável muda.

Para $a = 0$, a Série de Taylor é convertida na Série de Maclaurin:

$$f(x) = f(0) + f'(0)x + \frac{f''(0)x^2}{2!} + \frac{f'''(0)x^3}{3!} + \ldots + \frac{f^{(0)}x^n}{n!} + R_n$$

Por exemplo, a Série de Maclaurin para e^x:

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + \frac{x^n}{n!} + \cdots$$

Uma observação pertinente: nem toda série converge, mesmo quando seu termo geral $a_n \to 0$. Por exemplo, a chamada Série Harmônica

$$S_n = \sum_{n=1}^{\infty} \frac{1}{n}$$

não converge.

[2]Brook Taylor (Londres, 18 de agosto de 1685 Londres, 30 de novembro de 1731) foi um matemático britânico. Foi eleito Membro da Royal Society em 1712.

3.5.2 Estimativa do erro de truncamento

Podemos usar a Série de Taylor ou a de Maclaurin para estimar o erro de truncamento em um método numérico. Vejamos um exemplo considerando a função $f(x) = \cos(x)$. Estimar o valor de valor de $\cos(\pi/6)$ pela Série de Maclaurin e calcular o erro de truncamento.

Solução. Sabemos que $\cos(\pi/6) = \sqrt{3}/2 \cong 0,8660254037844386$. Podemos calcular:

$$\cos(x) = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

Sabemos que a aproximação de ordem zero é: $f(0) = 1$, que gera um erro de truncamento considerável:

$$\varepsilon_t = \left| \frac{1 - 0,8660254037844386}{0,8660254037844386} \right| \times 100\% = 15,47\%$$

A aproximação de primeira ordem, neste caso, não é melhor, pois $f'(0) = 0$, a de segunda ordem é:

$$\varepsilon_t = \left| \frac{1 - (\pi/6)^2/2 - 0,8660254037844386}{0,8660254037844386} \right| \times 100\% = 0,3583\%$$

que apresenta uma melhora significativa. Na Tabela 3.4 temos os resultados para mais termos adicionados. Para $n \geq 6$ a redução no erro de truncamento é praticamente insignificante.

Tabela 3.4: Erros de truncamento - aproximação da função $\cos(x)$ para $x = \pi/6$

Ordem n	$f(x)$	$\varepsilon_t(\%)$
0	1.0000000000	1.547005e+01
2	0.8629221611	3.583316e-01
4	0.8660538834	3.288545e-03
6	0.8660252641	1.612930e-05
8	0.8660254042	4.918027e-08
10	0.8660254038	1.022119e-10

Usando a Série de Taylor, podemos estimar a função $f(x+h)$ como:

$$f(x+h) \cong f(x) \qquad \text{estimativa de ordem zero}$$
$$f(x+h) \cong f(x) + f'(x)h \qquad \text{estimativa de ordem um}$$
$$f(x+h) \cong f(x) + f'(x)h + f''(x)\frac{h^2}{2} \qquad \text{estimativa de ordem dois}$$
$$f(x+h) \cong f(x) + f'(x)h + f''(x)\frac{h^2}{2} + f'''(x)\frac{h^3}{3!} \qquad \text{estimativa de ordem três}$$
$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2} + f'''(x)\frac{h^3}{3!} + \cdots + R_n \qquad \text{estimativa de ordem } n$$

Então, a estimativa de erro de truncamento é dada por:

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1} \tag{3.1}$$

onde ξ está entre x e $x+h$. De forma simplificada, podemos escrever: $R_n = O(h^{n+1})$, isto é, o erro de truncamento é da ordem de h^{n+1}. Novamente, quanto menor o valor de h, menor o erro de truncamento. O valor exato de ξ nem sempre é fácil de calcular, mas, algumas vezes, conseguimos estabelecer limitante superior para R_n. Por exemplo, para função $f(x) = cos(x)$ do exemplo acima ($x = 0$, $h = \pi/6$), podemos estimar R_6 por:

$$\begin{aligned} R_6 &= \frac{f^{(6+1)}(\xi)}{(6+1)!}(\pi/6)^{6+1} \\ &\leq \frac{(\pi/6)^7}{7!} \\ &\leq 2,14 \times 10^{-6} \end{aligned}$$

que é um valor compatível com os dados da Tabela 3.4.

3.6 Erro total

O erro total gerado pelo truncamento e as operações de arrendondamento pode ser aproximado pela soma do erro de truncamento em um computador ideal mais o erro de arredondamento decorrente da quantidade de operações efetuadas.

A Figura 3.1 mostra a combinação do erro de truncamento e de arredondamento para o cálculo da derivada de uma função. A derivada de função pode ser aproximada por:

$$f'(x) \cong \frac{f(x+h) - f(x)}{h} + O(h) \qquad \text{(diferença progressiva)}$$

$$f'(x) \cong \frac{f(x+h) - f(x-h)}{2h} - O(h^2) \qquad \text{(diferença centrada)}$$

onde $O(h)$ e $O(h^2)$ são as ordens dos erros de truncamento. Para valores muito pequenos do passo h, o erro de arredondamento será significativo, pois teremos cancelamentos de termos da mesma ordem de grandeza. Já o erro de truncamento cresce com h (aproximação grosseira da derivada). Logo, devemos esperar que exista um valor ótimo do passo h que minimize o erro total. É exatamente isso que vemos na Figura 3.1 para função $f(x) = -\exp(-x/2) + \sin(2x)$ com $x = \pi/3$. Assim, o valor numérico mais preciso da derivada $f'(x)$ não é obtido quando o passo h é muito pequeno, mas quando ele assume um valor *ótimo*. Esse valor ótimo depende do tipo de *float* que estamos usando e da função (e suas derivadas) que está sendo avaliada. Para este exemplo, usando a diferença centrada, podemos estimar o passo ótimo por:

$$h^c_{otm} = \sqrt[3]{\frac{3\varepsilon}{M_c}} \tag{3.2}$$

onde ε é devido ao erro de arredondamento e M_c é o valor máximo de $f^{(3)}(\xi)$. Para derivada estimada pela diferença progressiva:

$$h^p_{otm} = \sqrt{\frac{4\varepsilon}{M_p}} \tag{3.3}$$

onde M_p é o valor máximo a de $f''(\xi)$. Nos dois casos, $\varepsilon \cong eps = 2,22 \times 10^{-16}$; $M_c \cong 8$ e $M_p \cong 4$. Logo, teremos: $h^c_{otm} \cong 4,4 \times 10^6$ e $h^p_{otm} \cong 1,5 \times 10^{-8}$. Estes valores estão compatíveis com as curvas da Figura 3.1. Na prática, entretanto, raramente é possível determinar o valor ótimo para h, pois teríamos que conhecer as derivadas da função.

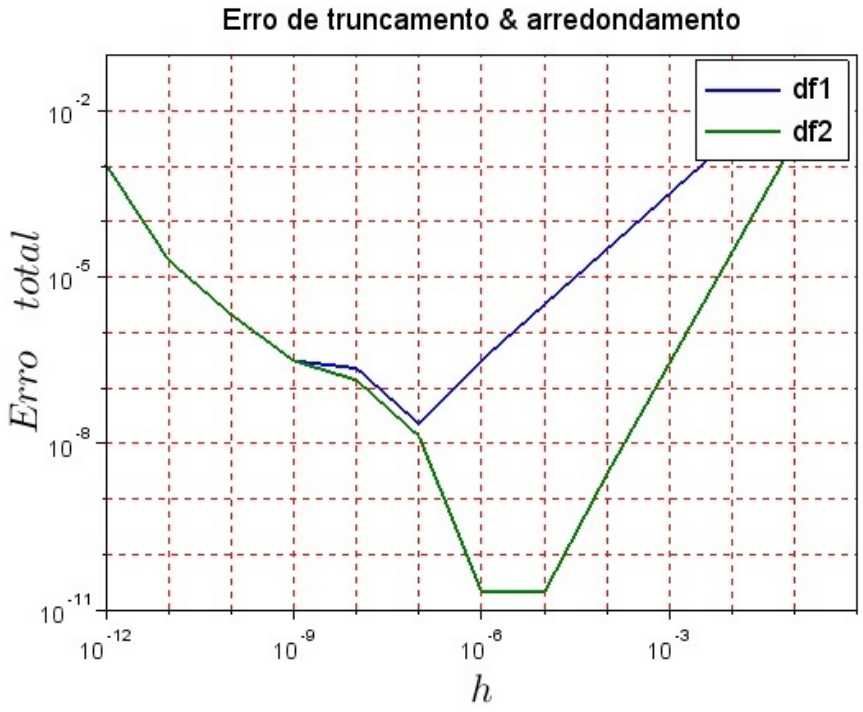


Figura 3.1: Erro total: truncamento "+" arredondamento
(df1: diferença progressiva; df2: diferença centrada)

3.7 Outros tipos de erros

Naturalmente, quando traduzimos um problema em algoritmo e, em seguida, implementamos esse algoritmo em uma linguagem, diversos erros podem ser incluídos, sejam eles de devido à simplificação excessiva do problema ou modelo, alguma falha ou instabilidade no algoritmo implementado, erros de digitação, etc. A própria ordem com que as operações são efetuadas pode levar a resultados nem sempre ligeiramente diferentes. Podemos ter também alguma incerteza nos dados de entrada ou inseri-los com algum erro, como, por exemplo, com a unidade errada (troca de graus por radianos é um exemplo de erro clássico). Essas incertezas e equívocos pode levar a uma solução totalmente errada, sem nenhuma conexão com a resposta desejada.

Consideremos a equação $x^2 + 2,15x + 2,14 = 0$, suas raízes são $x_1 = -1,2$ e $x_2 = -0,95$. Uma pequena "perturbação" no termo independente, por exemplo, $x^2 + 2,15x + 2,12 = 0$, leva às raízes $x_1 = -1,2637459$ e $x_2 = -0,8862541$. Um erro de $0,02$ levou a uma diferença "amplificada" de $0,0637459$, esse é um exemplo de problema de mal condicionamento. Algoritmos podem sofrer desse tipo de problema e gerarem resultados errôneos se os dados estiverem com algum erro inicial significativo.

Sempre que possível, esses erros devem ser minimizados ou, pelo menos identificados, usando, por exemplo, duas ou mais técnicas numéricas distintas para que os resultados obtidos sejam comparados. Algumas vezes, podemos usar a mesma técnica, mas com o número de iterações diferentes ou com o passo menor. Também podemos empregar estudos mais teóricos sobre a propagação de erros e análise usando alguma ferramenta, como a Série de Taylor.

3.8 Curiosidade: pequenos erros, grandes problemas

Uma fonte de muita dor de cabeça para engenheiros e cientistas é a conversão de unidades, por exemplo, do sistema métrico para o anglosaxão. Vejamos dois exemplos[3]:

- O Vesa, um navio sueco de guerra, considerado o mais poderoso do mundo, afundou em sua viagem inaugural em 1628 porque um lado era mais largo que o outro. Essa assimetria foi causada porque foram usadas réguas calibradas em unidades diferentes.
- Em 1983, um avião ficou levantou voo com apenas a metade do combustível necessário e teve que fazer um pouso de emergência. A conversão entre quilos e libras causou essa pane seca.

Não são só os erros "grosseiros" que podem manchar a reputação dos engenheiros. Algumas vezes um detalhe minúsculo pode causar uma grande dor de cabeça. Por exemplo, o famoso telescópio espacial Hubble (`http://hubblesite.org/the_telescope/`) apresentou um grave problema de "miopia" logo no início de suas operações. O tamanho do problema? Um erro de apenas 2,5 mícrons no espelho principal. O reparo desse erro custou milhões de dólares e uma arriscada viagem espacial com astronautas fazendo manobras no espaço sem gravidade.

3.9 Problemas

Questão 1. A divisão simples $x = 1/3$ usando o Scilab retém quantas casas decimais corretas?

Questão 2. O valor de π pode ser aproximado pela fração $355/113$. Quais os erros relativo e absoluto?

Questão 3. A medida do comprimento de uma caneta foi estimada em $18,30cm$. Se o valor correto é $18,25cm$, qual é o erro relativo e o erro absoluto?

Questão 4. O valor da resistência de um resistor ôhmico (que obedece a Lei de Ohm) comercial *sempre* tem alguma incerteza (tolerância). Essa incerteza é, em geral, expressa em termos percentuais. Assim, um resistor de 100Ω com uma tolerância de 10% pode ter qualquer valor entre 90Ω e 110Ω. Uma associação série de resistores é formada por 3 resistores de 120Ω (tolerância de 10%). Qual os valores máximos e mínimos esperados para essa associação de resistores?

Questão 5. Quantas casas decimais *corretas* tem a constante "%pi" no Scilab? O valor de π com 30 casas decimais corretas é: $\pi = 3,14159265358979323846264338327 9$.

Questão 6. O famoso matemático indiano Srinivasa Ramanujan descobriu a seguinte identidade:

$$\frac{1}{\pi} = \frac{\sqrt{8}}{9801} \sum_{n=0}^{N} \frac{(4n)!(1103+26390n)}{(n!396^n)^4}$$

com $N \to \infty$. Qual o erro absoluto cometido nesse cálculo para $N = 1,2,3$?

Questão 7. O logaritmo natural $1+x$ pode ser calculado pela série:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \ldots + (-1)^n \frac{x^n}{n} + \ldots$$

[3] Fonte aqui: `http://www.bbc.com/portuguese/noticias/2014/05/140530_erros_ciencia_engenharia_rb`. Acesso: 04 de janeiro de 2018.

Para qual faixa de valores de x a expressão acima consegue convergir? Qual o erro de truncamento cometido para $x = 0,5$ e o uso dos cinco primeiros termos da série?

Questão 8. Podemos aproximar

$$\frac{1}{1+x} \cong 1-x$$

quando o valor de $|x|$ é pequeno. Qual o valor máximo de x tal que o erro relativo nessa aproximação seja menor que 5%?

Questão 9. Podemos melhorar a aproximação da questão anterior por:

$$\frac{1}{1+x} \cong 1-x+x^2$$

para um o valor de $|x|$ pequeno. Qual o valor máximo de x tal que o erro relativo com esta nova aproximação seja menor que 5%?

Questão 10. Escreva e execute o seguinte código no Scilab:

```
N1 = 0.75e308;  N2 = 1.33e308;   // números grandes
Ns = N1 + N2; Nm = N1 - N2;
Np = N1 * N2; Nd = N1/N2;
disp([N1, N2, Ns, Nm, Np, Nd]);
```

Quais os resultados obtidos? Eles estão dentro do esperado? Comente.

Questão 11. Execute o seguinte código:

```
eps = 1; while (eps+1)>1; eps=eps/2; end; disp(eps);
```

O que "eps" significa? Pesquise sobre *machine epsilon*.

Questão 12. Considere a função $f(x) = (1+\cos(x))/(2-\cos(x))$. Estime a derivada de $f(x)$ em $x = \pi/4$ usando diferença progressiva e diferença centrada com $h = 10^{-6}$. Refaça os cálculos para $h = 10^{-8}$ e $h = 10^{-3}$. Calcule a derivada verdadeira de $f(x)$ em $x = \pi/4$ e compare os resultados.

Questão 13. Verifique que a soma

$$S = \sum_{k=1}^{1901} \frac{1}{1901}$$

quando efetuada usando o computador não é *exatamente* igual a 1. Por que existe essa diferença?

Questão 14. A função cosseno (com o ângulo em radianos) pode ser aproximada por:

$$\cos(\theta) = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + \ldots$$

Quantos termos dessa série são necessários para calcular o cosseno de $\pi/4$ com um erro de truncamento menor que $0,01\%$?

Questão 15. O π é um número irracional, entretanto podemos usar algumas aproximações muito boas. Por exemplo, $\pi \cong 355/113$ e $\pi \cong \sqrt[4]{97,5 - 1/11 + 1/8005667}$. Calcule os erros relativo e absoluto para essas duas aproximações.

4. Raízes de uma função

As equações são mais importantes para mim, porque a política é para o presente, mas uma equação é algo para a eternidade.
Albert Einstein (1879 - 1955), físico teórico alemão.

Neste capítulo, iremos apresentar diversas técnicas para o cálculo da raízes (zeros) de funções não lineares reais. O cálculo das raízes de polinômios de segundo grau ou de funções lineares é trivial, mas as raízes de funções exponenciais, logarítmicas e trigonométricas muitas vezes exige o uso de métodos numéricos. Esse tipo de cálculo surge naturalmente na resolução de diversos problemas de engenharia.

4.1 Inspeção gráfica

Se temos disponível um computador e um *software* (ou uma calculadora científica capaz de gerar gráficos) para traçar o gráfico de uma função qualquer, podemos analisar o gráfico e descobrir por simples inspeção onde estão localizados os zeros da função. Em geral, se a função não for periódica, os zeros estão "próximos" da origem ou se localizam no "infinito". Por exemplo, a função $f(t) = e^{-|t/3|}cos(\pi t)$ possui infinitos zeros, sendo que o primeiro zero para $t > 0$ se localiza em $t = 1/2$, como podemos ver na Figura 4.1. Já a função $f(t) = t^3 - t - 3$ possui um único zero real em $t \cong 1,6716999$, esse valor, com esse número de casas decimais, não é possível obter apenas analisando o gráfico da função.

Em geral, o valor *exato* de um zero de uma função pode não ser tão fácil de avaliar graficamente. Nestes casos, devemos tentar algum método iterativo que nos leve a esse valor. De qualquer forma, o gráfico da função é uma ferramenta útil para delimitar a região onde está localizado o zero ou os zeros da função estudada. Na verdade, se soubermos apenas que houve mudança no sinal da função, isto é, se $f(a) > 0$ e $f(b) < 0$, então já podemos concluir que existe pelo menos uma raiz entre no intervalo (a,b). Esse fato pode definido formalmente por meio de um teorema:

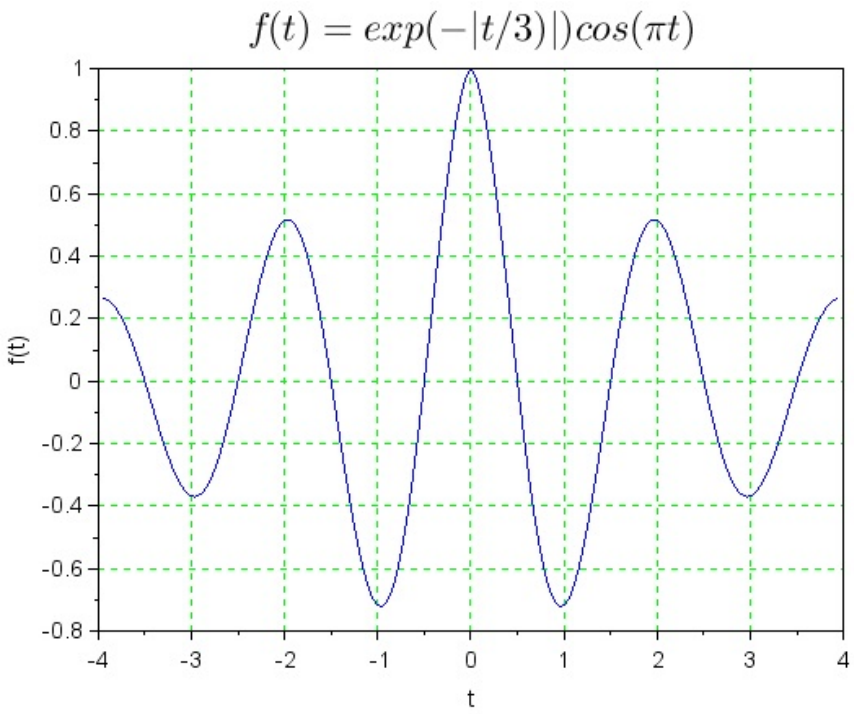


Figura 4.1: Exemplo de função com muitos zeros - inspeção gráfica.

Teorema 4.1.1 Seja $f(x)$ uma função contínua no intervalo (a,b), se $f(a)f(b) < 0$ então existe pelo menos um ponto $x = \xi$ entre a e b que é zero da função $f(x)$.

Considerando que o teorema 4.1.1 seja satisfeito, se $f'(x)$ existir e manter o sinal dentro no intervalo (a,b), então este intervalo contém um único zero de $f(x)$. Exemplo: vamos considerar novamente a função $f(t) = t^3 - t - 3$. É fácil perceber que: A mudança de sinal ocorre no intervalo

t :	-3	-2	-1	0	1	2	3
$f(t)$:	-	-	-	-	-	+	+

$(1,2)$, logo a raiz está nesse intervalo, como já era de nosso conhecimento. De forma geral, podemos descrever um algoritmo iterativo para cálculo de uma raiz de uma função como mostrado na Figura 4.2.

4.2 Método da iteração linear

Neste método, também chamado de método do ponto fixo, a função $f(x) = 0$ é convertida em fórmula de recorrência do tipo $x_{k+1} = g(x_k)$, sendo x_0 um chute inicial que pertence ao intervalo (a,b) onde está localizada a raiz de $f(x)$. A Figura 4.3 mostra a convergência desse método para a função $f(x) = 1 + 2x - x^3$. Forma de recorrência é $x_{k+1} = \sqrt[3]{1+2x}$ e valor inicial é $x_0 = 1$. Na interceptação das funções $y = x$ e $y = \sqrt[3]{1+2x}$ está o valor desejado, a raiz de $f(x)$.

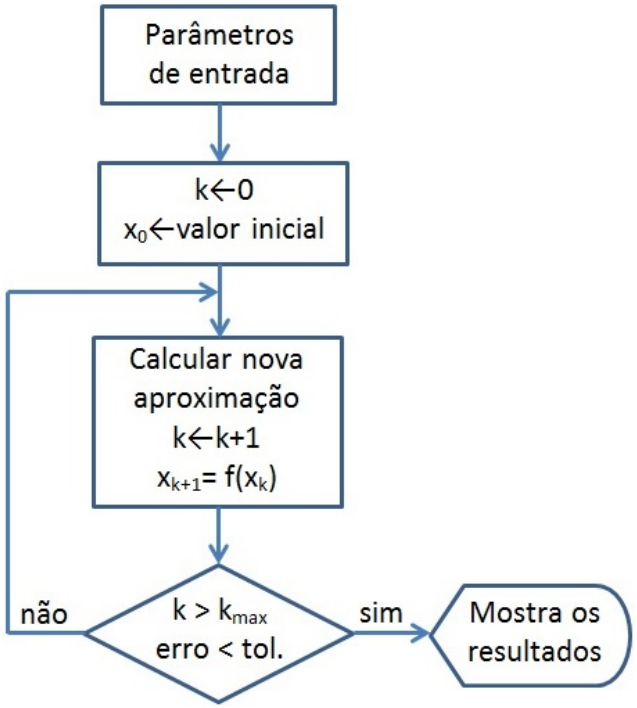


Figura 4.2: Diagrama de fluxo para cálculo de uma raiz de uma função

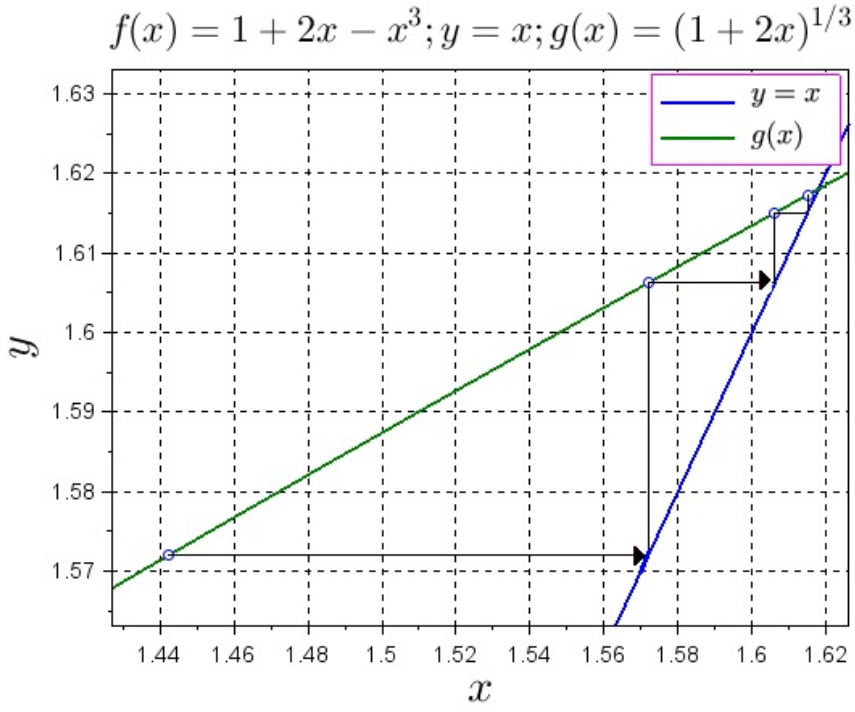


Figura 4.3: Exemplo de convergência do método do ponto fixo (iteração linear).

Veremos o funcionamento deste método usando novamente a função $f(t) = t^3 - t - 3$, então, podemos *escolher* como função de recorrência $t_{k+1} = \sqrt[3]{t_k + 3}$ e $t_0 = 1,5$. Assim, podemos escrever o seguinte código Scilab para o cálculo da raiz dessa função:

```
tk = 1.5; k = 0;
disp([k, tk]);
erro = 1;
while abs(erro) > 1e-9
    erro = tk;
    tk = (3 + tk)^(1/3);
    erro = erro - tk;
    k = k + 1;
    disp([k, tk]);
end
```

Obtemos o resultado indicado na Tabela 4.1. Podemos perceber pela Tabela 4.1 que o critério de parada poderia ter sido relaxado um pouco, pois a partir da $6^{a.}$ iteração o valor de t_k é praticamente o mesmo. Uma observação importante: o código Scilab acima tem um problema em potencial, se a função de iteração não convergir e o erro não diminuir, o programa pode entrar em um *loop* infinito. Para evitar esse risco, basta fazer, por exemplo: "while (abs(erro) > 1e-9)&(k < 25)", isto é, se o número de iterações passar de um certo valor, o laço é finalizado. De forma geral, um laço iterativo precisa ter um ou mais critérios de parada para evitar esses *loops* infinitos. Na Tabela 4.2, apresentamos os principais critérios de parada de um algoritmo iterativo para cálculo de raízes de uma função.

Tabela 4.1: Método de Iteração Linear, função: $t^3 - t - 3 = 0$,
função de iteração: $t_{k+1} = \sqrt[3]{t+3}$.

k	t_k
0	1,5
1	1,6509636
2	1,6692228
3	1,6714044
4	1,6716646
5	1,6716957
6	1,6716994
7	1,6716998
8	1,6716999
9	1,6716999
10	1,6716999

Tabela 4.2: Critérios de parada de um laço

número de iterações	tolerância da raiz	tolerância da função	erro relativo
$k > N_{max}$	$\lvert x_{k+1} - x_k \rvert < \varepsilon_x$	$\lvert f(x_k) \rvert < \varepsilon_f$	$\lvert x_{k+1} - x_k \rvert / x_{k+1} < \varepsilon_r$

Vejamos um outro exemplo. A função $f(t) = e^{-t/2} cos(2t/3) + 0,04$, ver Figura 4.4, possui dois

zeros para $t > 0$, um zero no intervalo $(2,3)$ e outro em $(5,7)$. Que função iterativa podemos usar? Podemos pensar em duas:

$$t_{k+1} = g_1(t_k) = -2\ln\left(-\frac{0,04}{cos(2t_k/3)}\right) \tag{4.1}$$

$$t_{k+1} = g_2(t_k) = \frac{3}{2}acos\left(-\frac{0,04}{exp(-t_k/2)}\right) \tag{4.2}$$

A primeira função, com $t_0 = 2$, consegue convergir rapidamente para a raiz do intervalo $(2,3)$, já a segunda função, com $t_0 = 5$, não converge para a solução, mas fica oscilando entre dois valores, como mostra a Tabela 4.3. Uma conclusão desse exemplo: o método da iteração linear nem sempre irá funcionar.

Tabela 4.3: Método de Iteração Linear, função: $e^{-t/2}\cos(2t/3) + 0,04$

k	t_k - $(g_1())$	t_k - $(g_2())$
0	2	5
1	2,5196145	6,4007597
2	2,5683860	4,7528451
3	2,5736606	6,4370241
4	2,5742390	4,6482501
5	2,5743025	6,4359227
6	2,5743095	4,6515261

Para finalizar, um teorema que mostra os critérios de convergência quando usamos a técnica do ponto fixo ou iteração linear:

Teorema 4.2.1 Sendo ξ uma raiz de $f(x) = 0$, isolada em um intervalo $I = (a,b)$ centrado em ξ e $g(x)$ uma função de iteração para $f(x) = 0$. Se

1. $g(x)$ e $g'(x)$ são contínuas em I;
2. $|g'(x)| < 1, \forall x \in I = (a,b)$, e
3. $x_0 \in I$;

então a sequência x_k gerada pelo processo iterativo $x_{k+1} = g(x_k)$ convergirá para ξ.

4.3 Método da bisseção

Nesta técnica, também conhecida como método de Bolzano, o intervalo (a,b), que contém uma raiz da função $f(x)$, é sucessivamente divido na metade até que $|a_k - b_k| < \varepsilon$ ou que a raiz seja eventualmente encontrada ou, ainda, que $|f(a_k)| < \varepsilon_f$ ou $|(f(b_k)| < \varepsilon_f$. Vejamos um exemplo. Consideremos novamente a função $f(t) = e^{-t/2}\cos(2t/3) + 0,04$. Para esta função não fomos capazes de determinar a sua raiz que está no intervalo $(5,7)$ usando o método do ponto fixo. O código Scilab a seguir é capaz de encontrar essa raiz, mas de forma relativamente lenta (ver Tabela 4.4). Escolhemos como principal critério de parada o tamanho do intervalo (a,b), se ele for menor que 10^{-4} o laço termina.

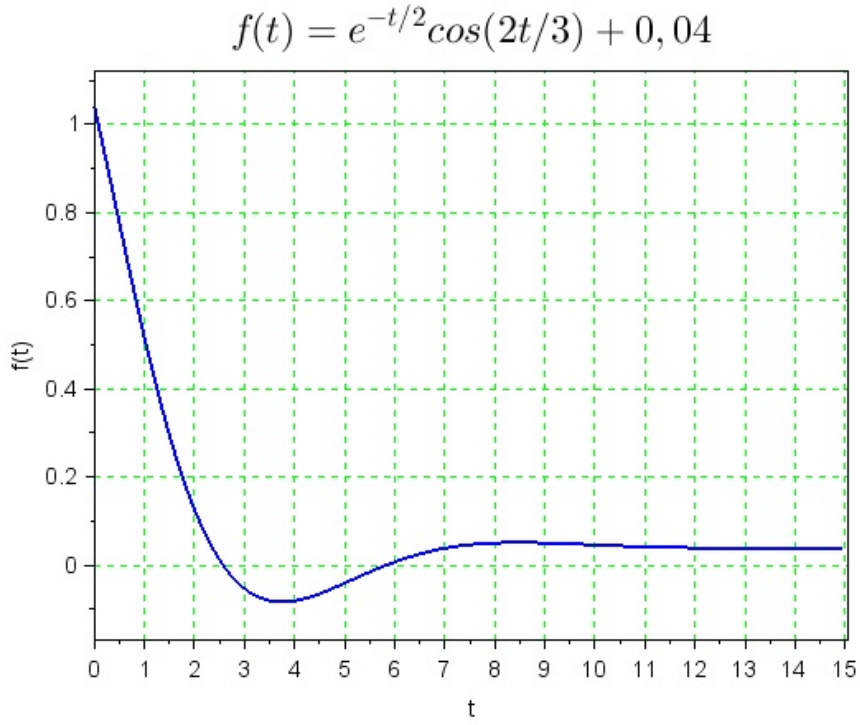


Figura 4.4: Gráfico da função $f(t) = e^{-t/2}\cos(2t/3) + 0,04$

```
a = 5;  //intervalo inicial
b = 7;
k = 0;  // contador
while (abs(a-b)>1e-4)&(k < 20)
    k = k + 1;
    c = (a+b)/2;  // novo ponto do intervalo
    ya = exp(-a/2).*cos(2*a/3) + 0.04;
    yb = exp(-b/2).*cos(2*b/3) + 0.04;
    yc = exp(-c/2).*cos(2*c/3) + 0.04;
    if ya*yc < 0 then b = c; else a = c; end; // novo intervalo
    disp([k, a, b, yc]);
end
```

O método da bisseção necessita do cálculo da função três vezes por iteração, logo, não é um método muito eficiente em termos computacionais. Podemos fazer uma estimativa no número de iterações que são necessárias para concluir esse método. Se o critério de parada for o tamanho do intervalo final com uma tolerância ε, então o número máximo de iterações será aproximadamente

$$N_{max} \cong \frac{\log(b-a) - \log(\varepsilon)}{\log(2)} \tag{4.3}$$

Este método pode ser usado para fazer um refinamento de uma solução gráfica inicial, mas não é adequado para, de forma geral, encontrar a raiz de uma função com grande precisão numérica.

Tabela 4.4: Método da bisseção, função: $e^{-t/2}\cos(2t/3)+0,04$

k	(a,b)	$f((a+b)/2)$
0	$(5,7)$	$---$
1	$(5,6)$	$0,0074570$
2	$(5,5;6)$	$-0,0153159$
3	$(5,75;6)$	$-0,0034481$
4	$(5,75;5,875)$	$0,0021398$
5	$(5,8125;5,875)$	$-0,0006219$
6	$(5,8125;5,84375)$	$0,0007672$
7	$(5,8125;5,828125)$	$0,0000747$
8	$(5,8203125;5,828125)$	$-0,0002731$
9	$(5,8242188;5,828125)$	$-0,0000991$
10	$(5,8261719;5,828125)$	$-0,0000122$
11	$(5,8261719;5,8271484)$	$0,0000313$
12	$(5,8261719;5,8266602)$	$0,0000095$
13	$(5,826416;5,8266602)$	$-0,0000013$
14	$(5,826416;5,8265381)$	$0,0000041$
15	$(5,826416;5,8264771)$	$0,0000014$

4.4 Método da falsa posição

Este método é semelhante ao da bisseção, mas, no lugar de simplesmente pegar a média do intervalo para definir o novo intervalo, o novo intervalo é definido por uma média ponderada, ver Figura 4.5. Isso pode acelerar o processo de convergência praticamente sem um custo computacional adicional. O novo intervalo é calculado por:

$$c = \frac{af(b)-bf(a)}{f(b)-f(a)}$$
$$a \leftarrow c \text{ se } f(a)f(c)<0$$
$$b \leftarrow c \text{ se } f(b)f(c)<0$$

O código a seguir mostra o funcionamento desse algoritmo, a Tabela 4.5, os resultados obtidos. Podemos notar uma melhora significativa na taxa de convergência em relação ao método da bisseção.

```
a = 5;  //intervalo inicial
b = 7;
c = 0;
k = 0;  // contador
yc = 1;
while (abs(yc)>1e-7)&(k < 25)
    k = k + 1;
    ya = exp(-a/2).*cos(2*a/3) + 0.04;
    yb = exp(-b/2).*cos(2*b/3) + 0.04;
    c = (a*yb - b*ya)/(yb - ya);
    yc = exp(-c/2).*cos(2*c/3) + 0.04;
    if ya*yc < 0 then b = c; else a = c; end; // novo intervalo
```

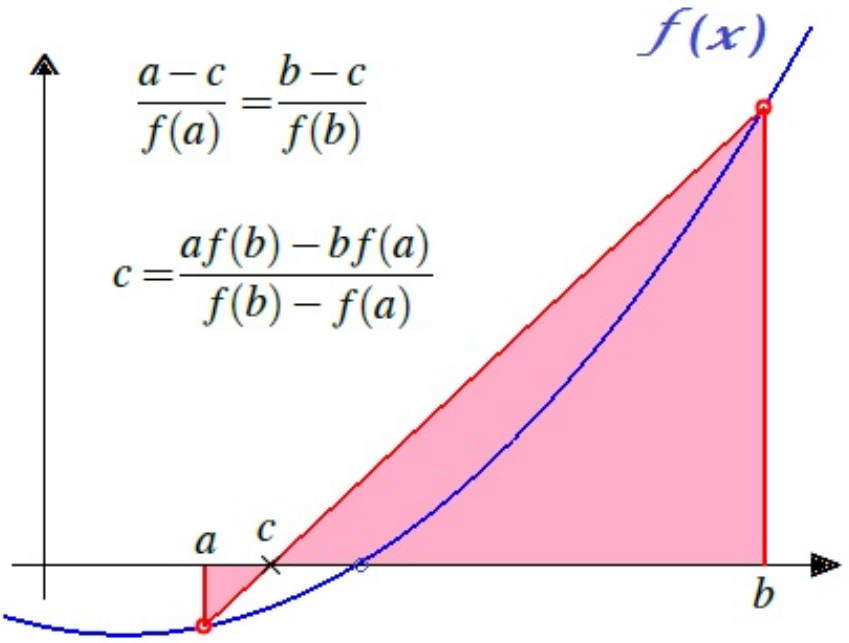


Figura 4.5: Cálculo da raiz aproximada usando o método da falsa posição

```
    disp([k, a, b, yc]);
end
```

Tabela 4.5: Método da falsa posição, função: $e^{-t/2}\cos(2t/3)+0,04$

k	(a,b)	$f(c)$
0	$(5,7)$	$---$
1	$(5;6,024759)$	$0,0084760$
2	$(5;5,8477013)$	$0,0009417$
3	$(5;5,8284763)$	$0,0000903$
4	$(5;5,8266371)$	$0,0000085$
5	$(5;5,8264638)$	$0,0000008$
6	$(5;5,8264475)$	$7,540D-08$

Neste exemplo, observamos que uma das extremidades ficou fixa. Isso ocorre porque a derivada segunda da função $f(t) = e^{-t/2}cos(2t/3)+0,04$ não muda de sinal no intervalo que contém a raiz. Esse fato pode, eventualmente, tornar a convergência mais lenta, como mostra o exemplo a seguir. Consideramos a função $f(x) = e^x - 2$, a raiz está no intervalo $(0,2)$. Abaixo, temos o código Scilab e na Tabela 4.6 apresentamos os resultados das iterações. Podemos constatar que a convergência é muito lenta nesse caso e são necessárias muitas iterações para o método convergir para próximo da raiz exata $\xi = \ln(2) = 0,6931472$.

```
function f = funcao(a)   //função
    f = exp(a) - 2;
endfunction
```

```
a = 0; b = 2;   //intervalo inicial
k = 0;  // contador
yc = 1;  //
while (abs(yc)>1e-6)&(k < 25)
    k = k + 1;
    ya = funcao(a);
    yb = funcao(b);
    c = (a*yb - b*ya)/(yb - ya);   // novo ponto
    yc = funcao(c);
    if ya*yc < 0 then b = c; else a = c; end; // novo intervalo
    disp([k, a, b, yc]);
end
```

Tabela 4.6: Método da falsa posição, função: $e^x - 2$

k	(a,b)	$f(c)$
1	(0,3130353 ; 2)	-0,6324302
2	(0,4902154 ; 2)	-0,3673322
3	(0,5865592 ; 2)	-0,2022081
4	(0,6376763 ; 2)	-0,1079209
5	(0,6644225 ; 2)	-0,0566321
6	(0,6783117 ; 2)	-0,0294519
...	...	...
20	(0,6931458 ; 2)	-0,0000027
21	(0,6931465 ; 2)	-0,0000014
22	(0,6931468 ; 2)	-0,0000007

Uma forma de evitar esse problema é usar o método de Pégaso.

4.5 Método de Pégaso

No método de Pégaso, o valor que está fixo é reduzido por um fator, isso faz com que a convergência seja acelerada. Outro método capaz de acelerar a convergência nestas situações é o método proposto por C. Ridders. Existem vários outros métodos mais avançados (Wijngaarden-Dekker-Brent, Muller, Ridders, Schröder) que são descritos, por exemplo, por Press et al. (2011) e Campos Filho (2013).

Podemos expressar o algoritmo de Pégaso por:

```
tol  // tolerância
kmax // número máximo de iterações
(a, b)  // intervalo inicial
k = 0   // contador
fa = funcao(a)  // função no ponto a
fb = funcao(b)  // função no ponto b
fx = fb
Enquanto (|(fx)|>tol) E (k < kmax)
```

```
    k = k + 1 // incrementando o contador
    x = b
    dx = -(b-a)*fx/(fb - fa)
    x = x + dx    // deslocando x em direção à raiz
    fx = funcao(x)
    Se (fx*yb)<0
    Então
        a = b
ya = yb
        Se não
    ya = ya*yb/(yb+fx)
    Fim Se
    b = x;
yb = fx
    Escreva(k, x, fx)
Fim enquanto
```

Rodando o algoritmo de Pégaso para a função $f(x) = e^x - 2$, obtemos os resultados indicados na Tabela 4.7. É evidente que esse método consegue convergir muito mais rápido que o algoritmo da falsa posição para este problema específico.

Tabela 4.7: Método de Pégaso, função: $e^x - 2$

k	c	$f(c)$
1	0,313035	-6,324302e-01
2	0,490215	-3,673322e-01
3	0,637075	-1,090582e-01
4	0,691369	-3,553371e-03
5	0,693193	9,170059e-05
6	0,693147	-8,155569e-08

4.6 Método de Newton-Raphson

O método de Newton-Raphson precisa de apenas um ponto inicial para iniciar a busca pela raiz de uma função. É um dos métodos mais usados para esse objetivo, pois tende a ser muito rápido. Podemos deduzir sua fórmula de iteração por uma análise geométrica, ver Figura 4.6. O cálculo da tangente da função no ponto inicial (x_0, y_0) leva, em geral, a um valor (x_1, y_1) mais próximo da raiz verdadeira da função. Assim, como algoritmo do método de Newton-Raphson, podemos escrever:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{4.4}$$

O teorema 4.6.1 garante formalmente as condições de convergência:

Teorema 4.6.1 Se $f(a)f(b) < 0$, e $f'(x)$ e $f''(x)$ forem não nulas e preservarem o sinal em (a,b), então, partindo-se da aproximação inicial $x_0 \in (a,b)$ tal que $f(x_0)f''(x_0) > 0$ é possível construir, pelo método de Newton-Raphson, uma sequência x_k que convirja para a raiz ξ de $f(x)$.

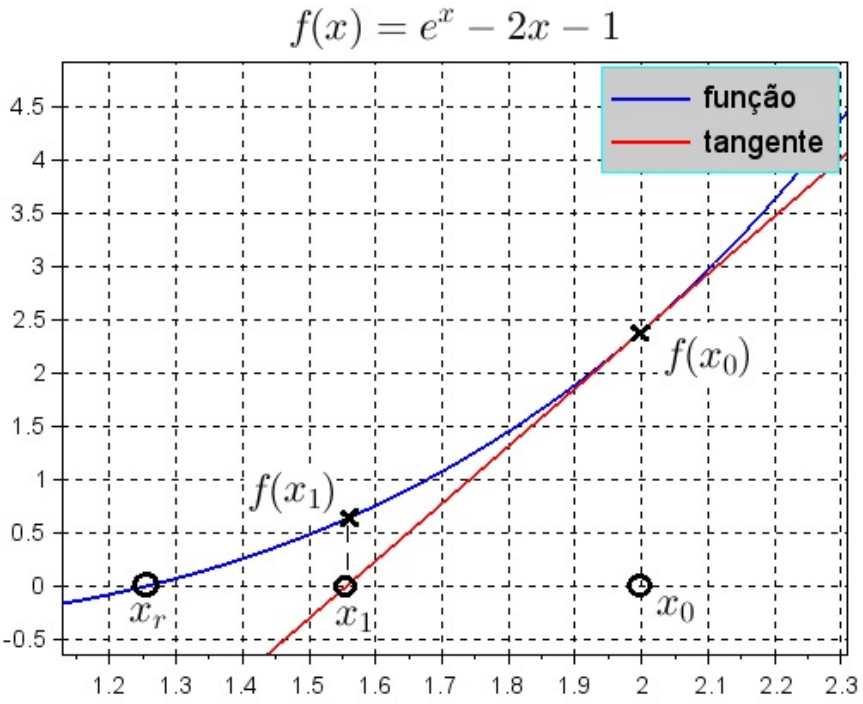


Figura 4.6: Método de Newton: descrição gráfica

Neste método, é necessário o conhecimento da derivada da função $f(x)$, o que nem sempre é uma tarefa fácil. Para função $f(x) = e^x - 2x - 1$, cuja derivada é $f'(x) = e^x - 2$, o método de Newton-Raphson fornece os valores indicados na Tabela 4.8 para $x_0 = 2$. Observamos que a convergência é rápida. Um possível código Scilab para este problema é:

```
function f = func2(a)   //função f(x)
    f = exp(a) - 2*a - 1;
endfunction

function g = func3(a)   //função f'(x)
    g = exp(a) - 2;
endfunction

// Inicialização do laço:
k = 0;
xk = 2;
erro = 1;
while (abs(erro)>1e-6)&(k<10)
    erro = func2(xk)/func3(xk);
    xk = xk - erro;
```

```
    k = k + 1;
    disp([k, xk, erro]);
end
```

Tabela 4.8: Método de Newton-Raphson, função: $e^x - 2x - 1$

k	x_k	$f(x_k)/f'(x_k)$
0	2	———
1	1,5566838	0,4433162
2	1,3271237	0,2295601
3	1,2616298	0,0654939
4	1,2564623	0,0051675
5	1,2564312	0,0000311
6	1,2564312	1,124D-09

Vejamos um segundo exemplo. A função $f(t) = e^{-t}cos(2t) - 1/4$ possui zero no intervalo $(0,1)$, logo $x_0 = 1/2$ é um bom chute inicial. O Método de Newton-Raphson fornece os valores mostrados na Tabela 4.9, como o valor inicial foi realmente bom, encontramos a raiz em poucas iterações. O

Tabela 4.9: Método de Newton-Raphson, função: $e^{-t}cos(2t) - 1/4$

k	x_k	$f(x_k)/f'(x_k)$
0	0,5	———
1	0.5576284	-0.0576284
2	0.5591112	-0.0014828
3	0.5591123	-0.0000011
4	0.5591123	-6.416D-13

método de Newton-Raphson, entretanto, pode apresentar uma baixa convergência em alguns casos ou mesmo divergir se a derivada de $f(x)$ for quase zero ou zero. Vamos considerar a função

$$f(x) = \frac{2e^{-2x} - 6}{2e^{-2x} + 6}$$

cujo gráfico podemos ver na Figura 4.7. Se fizermos uma escolha inapropriada para x_0, o algoritmo pode não convergir. Por exemplo, para $x_0 = -1$ a convergência ocorre rapidamente, mas para $x_0 = -1,7$ o algoritmo diverge, como podemos constatar na Tabela 4.10.

Tabela 4.10: Método de Newton-Raphson, função: $f(x) = \frac{2e^{-2x}-6}{2e^{-2x}+6}$

k	x_k	$f(x_k)/f'(x_k)$	k	x_k	$f(x_k)/f'(x_k)$
0	-1	———-	1	0.7719784	-2.4719784
1	-0.4857468	-0.5142532	2	-2.7225397	3.4945181
2	-0.5494775	0.0637307	3	16.575529	-19.298069
3	-0.5493061	-0.0001713	4	-1.872D+14	1.872D+14
4	-0.5493061	3.352D-12	5	Nan	Nan

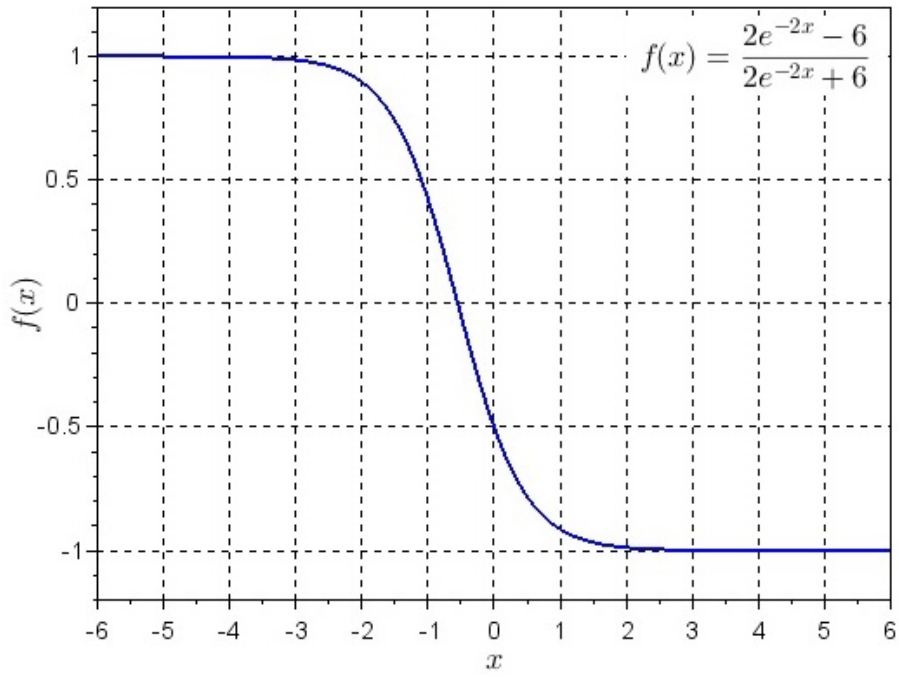


Figura 4.7: Função para a qual o método de Newton pode divergir.

Como último exemplo, veremos que é possível encontrar uma raiz complexa de um polinômio com método de Newton, desde que o chute inicial seja também complexo e próximo da raiz. Seja $p(x) = x^3 - 3x - 18$, com $x_0 = 0,4 + j/2$. Os valores obtidos são:

```
k = 1, xk = -5.001231 + 1.764978 j, |erro| = 4.771949e+00
k = 2, xk = -3.287471 + 1.272397 j, |erro| = 1.783146e+00
k = 3, xk = -2.015873 + 1.131441 j, |erro| = 1.279386e+00
k = 4, xk = -1.141179 + 1.695720 j, |erro| = 1.040914e+00
k = 5, xk = -1.587483 + 1.934558 j, |erro| = 5.061933e-01
k = 6, xk = -1.501400 + 1.933938 j, |erro| = 8.608522e-02
k = 7, xk = -1.499997 + 1.936492 j, |erro| = 2.914389e-03
k = 8, xk = -1.500000 + 1.936492 j, |erro| = 3.288545e-06
k = 9, xk = -1.500000 + 1.936492 j, |erro| = 4.188492e-12
```

observamos que a convergência é um pouco mais lenta que na busca por uma raiz real. O código Scilab para este exemplo é:

```
function f = fx(x)   //função
    f = (x-3).*(x.*x + 3*x + 6);
endfunction

function df = dfx(x)   //função
    df = (x-3).*(2*x + 3) + (x.*x + 3*x + 6);
endfunction
```

```
erro = 1; k = 1;
xk = -0.4+0.5*%i;
tol = 1e-6;
vxr = real(xk);
vxi = imag(xk);
while erro>tol
    erro = fx(xk)/dfx(xk);
    xk = xk - erro;
    vxr = [vxr, real(xk)]; vxi = [vxi, imag(xk)];
    erro = abs(erro);
    mprintf('k = %d, xk = %1.6f + %1.6f j, |erro| = %e \n',k,real(xk),imag(xk),erro);
    k = k + 1;
    if k>10 then erro = 0;
    end
end

close; plot(vxr, vxi,'m-*');
xlabel('$ Real $'); ylabel('$ Imaginário $');
title(['Caminho no plano complexo de ','$ x_k $']);
```

A Figura 4.8 mostra a trajetória de x_k no plano complexo.

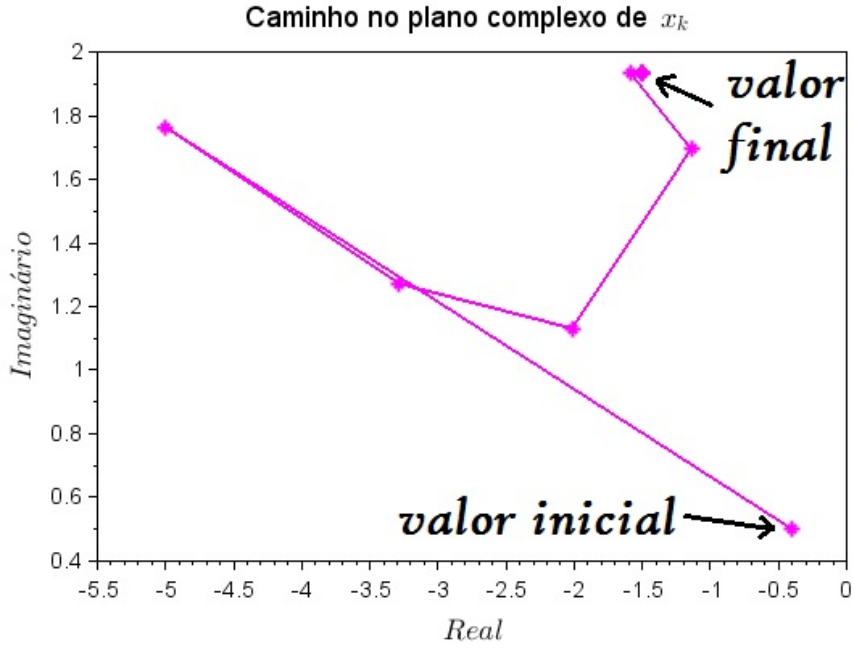


Figura 4.8: Trajetória de x_k no plano complexo

4.7 Método da secante

No método da secante, que é semelhante ao método da falsa posição e deriva do método de Newton, a derivada é aproximada por:

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

Assim, o método da secante tem como fórmula recursiva:

$$x_{k+1} = x_k - \frac{f(x_k)(x_{k-1} - x_k)}{f(x_{k-1}) - f(x_k)} \tag{4.5}$$

mas é mais prático usar o método da secante modificado. Dessa forma, a nova equação recursiva é

$$x_{k+1} = x_k - \frac{\delta f(x_k)}{f(x_k + \delta) - f(x_k)} \tag{4.6}$$

ou ainda,

$$x_{k+1} = x_k - \frac{2\delta f(x_k)}{f(x_k + \delta) - f(x_k - \delta)} \tag{4.7}$$

sendo δ um valor pequeno, por exemplo $\delta = 10^{-3}$. A expressão 4.7 pode produzir resultados levemente melhores que a equação 4.6. A convergência deste método é muito semelhante ao método de Newton, como podemos ver pela Tabela 4.11. Abaixo, temos o código Scilab:

```
function f = func2(a)   //função f(x)
    f = exp(a) - 2*a - 1;
endfunction

// Inicialização do laço:
k = 0;
xk = 2;  // chute inicial
erro = 1;
delta=0.001;
while (abs(erro)>1e-6)&(k<10)
    erro = delta*func2(xk)/(func2(xk+delta)-func2(xk));
    xk = xk - erro;
    k = k + 1;
    disp([k, xk, erro])
end
```

4.8 Uma comparação entre os métodos

Uma dúvida que naturalmente vem à tona é: qual o método usar? Na verdade, não existe um método de cálculo de raízes que seja infalível ou que sempre leve à raiz da função com um número razoável de iterações. Em geral, dos métodos estudados neste capítulo, o método de Newton-Raphson tende a ser o mais rápido, mas ele exige o conhecimento da derivada da função, o que nem sempre é fácil de

Tabela 4.11: Método da secante modificado, função: $e^x - 2x - 1$

k	x_k	$\frac{\delta f(x_k)}{f(x_k+\delta)-f(x_k)}$
0	2	———
1	1,5569876	0,4430124
2	1,3274428	0,2295448
3	1,2617444	0,0656984
4	1,2564698	0,0052746
5	1,2564313	0,0000385
6	1,2564312	4,641D-08

obter. Então, o método da secante modificado, no qual a derivada da função é aproximada (ver 4.6 e 4.7), pode ser usado eficientemente. Em seguida, o método da falsa posição tende a ser superior ao método da bisseção, mas, em casos "patológicos", o inverso pode ocorrer. Já o método da iteração linear (ou ponto fixo) depende da existência e da escolha de uma boa função de iteração $x_{k+1} = g(x_k)$. Todos esses métodos são beneficiados por uma análise gráfica preliminar.

Uma estratégia que pode funcionar muito bem é aplicar o método da bisseção por 3 ou 4 iterações para restringir o intervalo inicial e, em seguida, usar o método da secante modificado ou o método de Pégaso para encontrar a solução. Esse tipo de abordagem mista foi desenvolvido por Richard Brent, gerando o algoritmo de Brent (CHAPRA, 2013).

Vejamos um exemplo final comparando o desempenho de todos esses métodos. A função é $f(x) = x^2/2 - e^{x/2}$, ver gráfico na Figura 4.9. Essa função possui três zeros que estão nos intervalos $(-1,5;-0,5)$, $(2,5;3,5)$ e $(5,6)$. Iremos calcular apenas o zero localizado no intervalo $(2,5;3,5)$. Usaremos o seguinte critério de parada para todos os métodos: $|f(x_k)| < 5 \times 10^{-7}$ ou $k > 30$. O chute inicial será $x_0 = 2,5$ ou o intervalo $(2,5;3,5)$. Mostramos o código Scilab completo abaixo. Apresentamos os resultados das iterações nas tabelas 4.12 e 4.13.

```
clc; close; // limpando o console e fechando alguma janela

function f = func(a)   //função
    f = a.*a/2 - exp(a/2);
endfunction

function g = dfunc(a)   //derivada da função
    g = a -0.5*exp(a/2);
endfunction

tol = 5e-7; kmax = 30; // critérios de parada

x=-2:0.01:6;   // Gráfico da função
y=func(x);     // chamando a função
plot(x,y); xgrid;  // gráfico com grid;
title('$ \huge f(x) = x^2/2 - e^{x/2} $');
xlabel('$ \huge x $');  ylabel('$ \huge y $');
```

```
///////////////  Iteração linear - ponto fixo
disp('ponto fixo');
tk = 2.5; k = 0;  /////////// Linear
disp([k, tk, func(tk)]);
fx = 1;
while (abs(fx) > tol)&(k<kmax)
    tk = sqrt(2*exp(tk/2));
    fx = func(tk);
    k = k + 1;
    disp([k, tk, fx]);
end

///////////////  Método da bisseção
disp('bisseccao');
a = 2.5;  //intervalo inicial
b = 3.5;
k = 0;  // contador
yc = 1;
disp([k, a, func(a)]);
while (abs(yc)>tol)&(k < kmax)
    k = k + 1;
    c = (a+b)/2;  // novo ponto do intervalo
    ya = func(a);
    yb = func(b);
    yc = func(c);
    if ya*yc < 0 then b = c; else a = c; end; // novo intervalo
    disp([k, a, b, yc]);
end

///////////////  falsa posição
disp('falsa');
a = 2.5;  //intervalo inicial
b = 3.5;
k = 0;  // contador
yc = 1;
disp([k, a, func(a)]);
while (abs(yc)>tol)&(k < kmax)
    k = k + 1;
    ya = func(a);
    yb = func(b);
    c = (a*yb - b*ya)/(yb - ya);
    yc = func(c);
    if ya*yc < 0 then b = c; else a = c; end; // novo intervalo
    disp([k, a, b, yc]);
end
```

```
///////////////  Método de Pégaso:
a = 2.5;  //intervalo inicial
b = 3.5;
k = 0;  // contador
yc = 1;
ya = func(a);
yb = func(b);
fx = yb;
disp([k,b,fx]);
while (abs(fx)>tol)&(k < kmax)
    k = k + 1;
    x = b;
    dx = -(b-a)*fx/(yb - ya);
    x =x + dx;
    fx = func(x);
    if (fx*yb)<0 then a = b; ya = yb;
        else ya = ya*yb/(yb+fx);
    end
    b=x; yb = fx;
    disp([k,x,fx]);
end

///////////////  Inicialização do laço: Newton
disp('Newton');
k = 0;
xk = 2.5;
fx = 1;
disp([k, xk, func(xk)]);
while (abs(fx)>tol)&(k<kmax)
    erro = func(xk)/dfunc(xk);
    xk = xk - erro;
    k = k + 1;
    fx = func(xk);
    disp([k, xk, fx]);
end

///////////////  Inicialização do laço: Secante modificada
disp('Secante');
k = 0;
xk = 2.5;
fx = 1;
h=0.001;
disp([k, xk, func(xk)]);
while (abs(fx)>tol)&(k<kmax)
    dfnc = (func(xk+h)-func(xk-h))/(2*h);
```

```
    xk = xk - func(xk)/dfnc;
    k = k + 1;
    fx = func(xk);
    disp([k, xk, fx]);
end
```

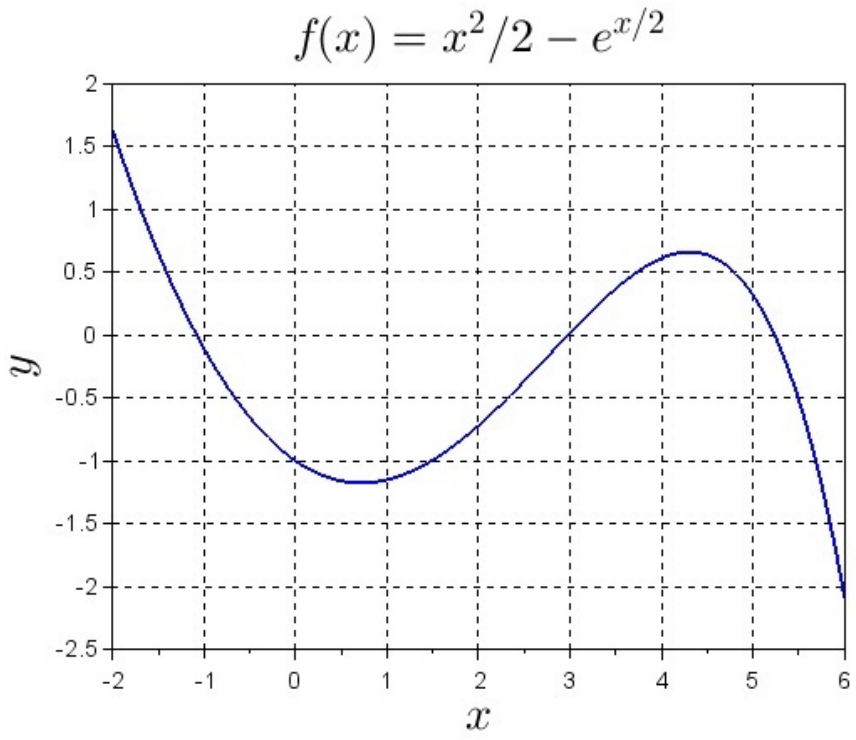


Figura 4.9: Função para comparação dos métodos.

Tabela 4.12: Comparação entre os métodos da falsa posição, Newton e secante modificada

	Falsa posição			Newton			Secante	
k	x_k	$f(x_k)$	k	x_k	$f(x_k)$	k	x_k	$f(x_k)$
0	2,5	-0,365343	0	2,5	-0,3653430	0	2,5	-0,3653430
1	2,9965651	0,0157026	1	2,9840079	0,0061554	1	2,9840079	0,0061554
2	2,9761020	0,0001355	2	2,9759194	-0,0000036	2	2,9759194	-0,0000036
3	2,9759255	0,0000010	3	2,9759241	-1,194D-12	3	2,9759241	-7,585D-13
4	2,9759241	7,632D-09	*	*******	*******	*	*******	*******

Para este problema final, os métodos de Newton, secante modificada, Pégaso e falsa posição, tiveram desempenhos semelhantes. Já os métodos da bisseção e iteração linear foram lentos, embora tenham chegado bem próximos da raiz desejada.

Tabela 4.13: Comparando os métodos de Pégaso, iteração linear e bisseção

	Pégaso			Iteração			bisseção	
k	x_k	$f(x_k)$	k	x_k	$f(x_k)$	k	x_k	$f(x_k)$
0	3,5	0.3703973	0	2,5	-0,3653430	0	2,5	-0,365343
1	2,996565	0.0157026	1	2,6420988	-0,2570088	1	3,0	0,0183109
2	2.975272	-0,0004975	2	2,7376456	-0,1833690	2	2,7500	-0,1738267
3	2.975925	7.4654e-07	3	2,8038262	-0,1322447	3	2,8750	-0,0773448
4	2.975924	3.4166e-11	4	2,8506018	-0,0961438	4	2,9375	-0,0293489
...	...	...	...	...	...	...	...	...
*	******	******	28	2,9758265	-0,0000744	17	2,9759293	0,0000039
*	******	******	29	2,9758515	-0,0000554	18	2,9759254	0,000001
*	******	******	30	2,9758701	-0,0000412	19	2,9759235	-0,0000005

4.9 Raízes de polinômios usando Scilab

Um caso especial de cálculo de raízes ocorre quando a função é polinomial. Não existem formas realmente práticas de calcular as raízes de polinômios de grau maior ou igual a três, a não ser em casos particulares. A necessidade de calcular as raízes de um polinômio surgem, por exemplo, quando estamos resolvendo uma equação diferencial ordinária de grau igual ou superior a dois.

Podemos calcular as raízes de polinômio de *qualquer* grau usando o comando "roots" do Scilab. Vejamos um exemplo básico inicialmente. Para o polinômio de segundo grau $f(x) = x^2 + 5x + 6$, as raízes são -2 e -3. Usando o comando "roots":

```
p = [1 5 6];   // coeficientes do polinômio
r = roots(p);  // uso do comando roots
disp(r);       // mostrando as raízes
```

Resulta em:
–> disp(r);
-3.
-2.
Que é a resposta esperada. O vetor "p" recebe os coeficientes de graus decrescentes do polinômio e o vetor "r" as raízes, sejam reais ou complexas. Quais as raízes de $f(x) = x^4 + 12x^3 + 52x^2 + 105x + 100$? Usando o comando "roots", obtemos:
-5.
-4.
-1.5 + 1.6583124i
-1.5 - 1.6583124i

O código Scilab é bastante compacto:

```
p = [1 12 52 105 100];
r = roots(p);
disp(r);
```

Vejamos mais um exemplo um pouco mais geral. Para o polinômio $f(x) = x^7 + 15x^6 - 20x^5 + 45x^4 - 62x^3 - 90x^2 - 120x + 150$ as raízes, no plano complexo, estão indicadas na Figura 4.10. São,

naturalmente, sete raízes, três raízes reais e mais quatro raízes complexas. Abaixo o código Scilab correspondente:

```
p = [1, 15, -20, 45,-62, -90, -120, 150];
r = roots(p);
rr = real(r); ri = imag(r);
figure; plot(rr,ri,'ro'); xgrid;
title('$\huge f(x) = x^7 + 15x^6 - 20x^5 + 45x^4 - 62x^3 - 90x^2 - 120x + 15$');
xlabel('$\huge Real $');
ylabel('$\huge Imaginário $');
```

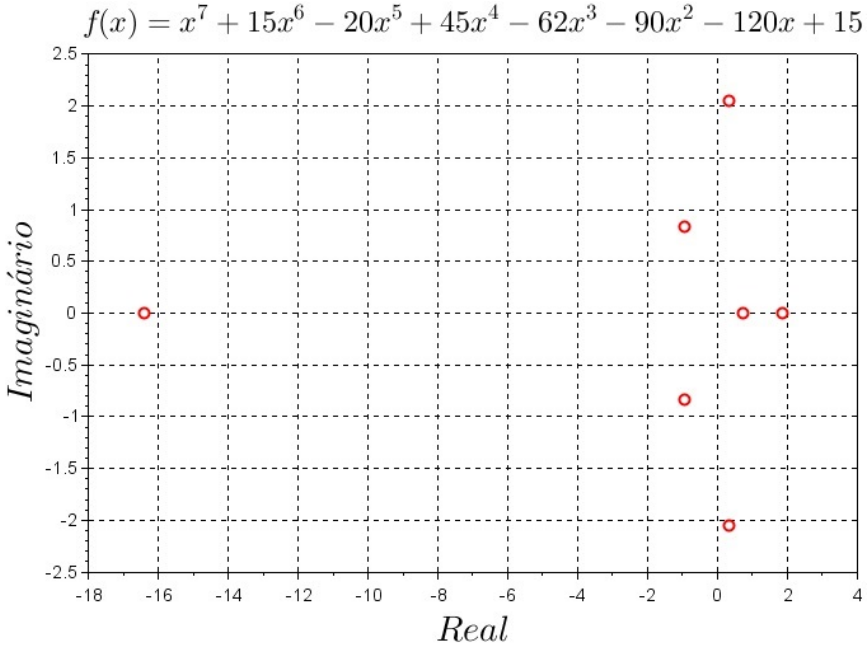


Figura 4.10: Raízes do polinômio $f(x)$ no plano complexo.

Contudo, nem tudo funciona tão bem assim. Polinômios de ordem mais alta e com raízes múltiplas podem levar a resultados numéricos distantes das raízes verdadeiras quando usamos o comando "roots". Por exemplo, vamos considerar o polinômio

$$x^{10} - 19x^9 + 138x^8 - 396x^7 - 378x^6 + 5670x^5 - 13608x^4 + 2916x^3 + 41553x^2 - 72171x + 39366 = 0$$

são raízes são todas reais: -2, -3 e $+3$ com multiplicidade 8. Quando usamos o comando "roots" obtemos:

```
 -2.
 -3.
  3.1252473
  3.0844113 + 0.0901234i
  3.0844113 - 0.0901234i
  2.9946546 + 0.1193926i
```

```
2.9946546 - 0.1193926i
2.9155662 + 0.0794103i
2.9155662 - 0.0794103i
2.8854884
```

apenas as raízes negativas foram calculadas corretamente, todas as raízes múltiplas apresentam algum erro, várias delas tendo, inclusive, uma parte imaginária residual. A Figura 4.11 mostra essas raízes no plano complexo. Percebemos que as raízes próximas a 3 formam um círculo.

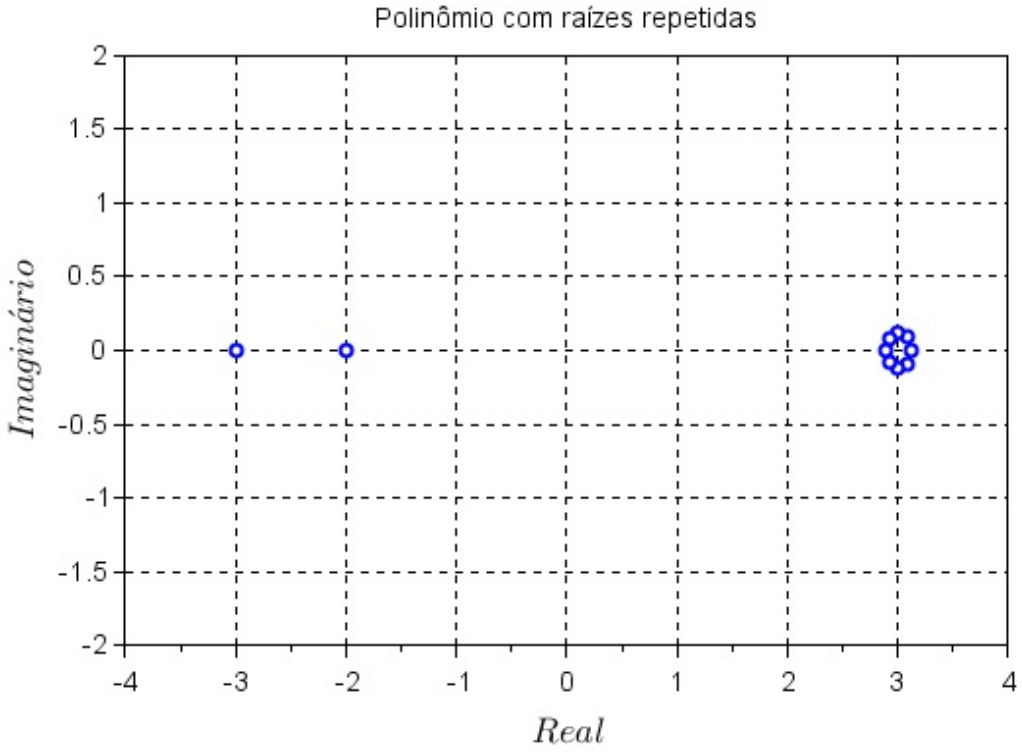


Figura 4.11: Raízes do polinômio $x^{10} - 19x^9 + 138x^8 - 396x^7 - 378x^6 + 5670x^5 - 13608x^4 + 2916x^3 + 41553x^2 - 72171x + 39366 = 0$ usando o comando *roots*.

4.10 Problemas

Questão 1. Esboce o gráfico da funções relacionadas abaixo e localize, aproximadamente, seus zeros:

(a) $f(x) = x^2 - 1 - e^{-x/2}$
(b) $f(x) = 2 - x\ln(x)$
(c) $f(x) = 2^x - 3$
(d) $f(x) = x^2 + 2sen(2x) - 1$

Questão 2. Usando o algoritmo do ponto fixo (iteração linear), encontre as raízes de $f(x) = e^{-x} + x^2 - 12$.

Questão 3. Usando o algoritmo do ponto fixo, localize a raiz de

$$f(x) = \frac{1}{4x}e^{-x^2} - 10^{-2}$$

no intervalo $(1,3)$.

Questão 4. Considere a equação

$$\frac{1}{k\sqrt{2\pi}}e^{-k^2/2} = 10^{-9}$$

Encontre o valor de k usando o método da iteração linear. Use o valor $k = 4$ como chute inicial.

Questão 5. Considere a função

$$f(x) = \frac{e^x - e^{-x}}{2}$$

Encontre o valor de x para o qual $f(x) = 6$ usando o método da iteração linear. Faça $x_0 = 1,5$ como valor inicial.

Questão 6. Considere a função $f(x) = x^2/2 - \text{sen}(x)$. Encontre o zero dessa função que está no intervalo $(1,2)$ usando método do ponto fixo.

Questão 7. Usando o método da bisseção, encontre o zero da função $f(x) = (x^8 + 5)(x + 1/2)^3$ que está no intervalo $(-1,0)$. Use como critério de parada $|b_k - a_k| < 10^{-4}$. Estime o número de iterações que serão necessárias.

Questão 8. Refaça as questões anteriores usando os métodos da falsa posição, Pégaso e secante modificada.

Questão 9. Podemos obter, aproximadamente, o valor de π ao aplicar o método de Newton-Raphson para encontra a raiz da função $f(x) = sen(x)$, usando como ponto de partida $x_0 = 2,5$. Verifique essa afirmação.

Questão 10. Podemos encontrar a raiz quadrada de um número positivo N aplicando a equação recursiva:

$$x_{k+1} = \frac{x_k + N/x_k}{2}$$

Mostre que essa forma de cálculo para $\sqrt{N}$, com $x_0 > 0$, conhecido como método babilônico, pode ser derivada do algoritmo de Newton-Raphson.

Questão 11. Com base na questão anterior, mostre que podemos calcular a raiz $\sqrt[p]{N}$ de um número positivo usando:

$$x_{k+1} = \frac{1}{p}\left((p-1)x_k + \frac{N}{x_k^{p-1}}\right)$$

Questão 12. A função $Q(z)$ é definida como

$$Q(z) = \int_z^\infty \frac{1}{\sqrt{2\pi}}e^{-y^2/2}dy$$

é muito usada para cálculo de taxa de erro em sistemas de comunicações digitais. Essa função pode ser aproximada por

$$Q(z) \cong \left(1 - \frac{0,7}{z^2}\right) \frac{1}{z\sqrt{2\pi}} e^{-z^2/2}$$

quando o valor de $z > 3/2$. Para qual valor de z a função $Q(z) = 10^{-3}$? Use os métodos da falsa posição, Pégaso e secante modificada para calcular o valor de z.

Questão 13. A resposta em frequência de um filtro do tipo passa-baixa é dada por:

$$H(s) = \frac{1}{s^2 + 10\sqrt{2}s + 100}$$

sendo $s = j\omega$, $0 \leq \omega \leq \infty$. Para qual valor de ω, o ganho $|H(s)|$ do filtro será igual a $1/10$? Escolha e use um dos métodos estudados para calcular ω.

Questão 14. Um circuito elétrico formado por um resistor $R = 20\Omega$, um capacitor $C = 1\mu F$ e um indutor $L = 1mH$, todos em paralelo. Nesse circuito, a impedância $Z(\Omega)$ é expressa por:

$$\frac{1}{Z} = \sqrt{\frac{1}{R} + \left(\omega C - \frac{1}{\omega L}\right)^2}$$

Para qual valor de ω a impedância é $Z = 25\Omega$?

Questão 15. Seja a função $f(t) = \text{sen}(2t)e^{-3t/4} + e^{-t}$. Encontre os dois zeros reais que estão no intervalo $(1,5; 3,5)$.

Questão 16. A equação elíptica $y^2 = x^3 - 9x + 4$ possui três zeros. Usando o método de sua preferência, encontre esses zeros.

Questão 17. Considere o polinômio $p(x) = x^5 + 2x^4 + 2x^3 - 23x^2 - 54x - 90$. Esboce seu gráfico e encontre sua raiz real usando qualquer um dos métodos estudados. Use o comando "roots" para calcular todas as raízes.

Questão 18. Refaça a questão anterior para $p(x) = x^7 + 4x^6 + 12x^5 - 7x^4 - 88x^3 - 336x^2 - 504x - 540$.

Questão 19. Calcule as raízes do polinômio $q(x) = x^8 + 20x^7 + 174x^6 + 860x^5 + 2641x^4 + 5160x^3 + 6264x^2 + 4320x + 1296$ com o comando "roots". As raízes são reais? Comente.

Questão 20. Tente usar o método de Newton-Raphson para encontrar uma das raízes da questão anterior.

Questão 21. Quais as raízes da função $f(x) = (x^2 - 4x + 3)^4 - 1/2$? Escolha um dos métodos apresentados neste capítulo. Compare o resultado obtido com o valor teórico.

5. Sistemas Lineares e não lineares

Os encantos dessa sublime ciência se revelam apenas àqueles que tem coragem de irem a fundo nela.
Carl Friedrich Gauss (1777 - 1855), matemático, astrônomo e físico alemão.

NA solução de alguns problemas nos deparamos com duas ou mais incógnitas relacionadas através de duas ou mais equações com coeficientes constantes. Assim, não temos uma equação para resolver, mas duas ou mais equações. Se essas equações tiverem a forma

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

então teremos um sistema de n equações lineares, onde a_{ij} são coeficientes constantes, x_i são variáveis e b_i são constantes, com $1 \leq i, j \leq n$. Por exemplo, como um passo intermediário na solução de um circuito elétrico, podemos encontrar um sistema de equações a partir das Leis de Kirchhoff[1]. Do circuito da Figura 5.1, obtemos:

$$\begin{cases} (R_1+R_8)I_1 & -R_1I_2 & +0I_3 & = V_1 \\ -R_1I_1 & +(R_1+R_2+R_4+R_6)I_2 & -R_2I_3 & = 0 \\ 0I_1 & -R_2I_2 & +(R_2+R_3+R_5+R_7)I_3 & = 0 \end{cases}$$

Outros problemas, como a solução de equações diferenciais, leva também a um sistemas de equações lineares. Problemas de Física que envolvem forças e momentos também podem levar a

[1] Gustav Robert Kirchhoff (Königsberg, 12 de março de 1824 - Berlim, 17 de outubro de 1887) foi um físico alemão. Suas contribuições científicas foram principalmente no campo dos circuitos elétricos, na espectroscopia, na emissão de radiação dos corpos negros e na teoria da elasticidade (modelo de placas de KirchhoffLove). Kirchhoff propôs o nome de "radiação do corpo negro", em 1862. É autor de duas leis fundamentais da teoria clássica dos circuitos elétricos e da emissão térmica.

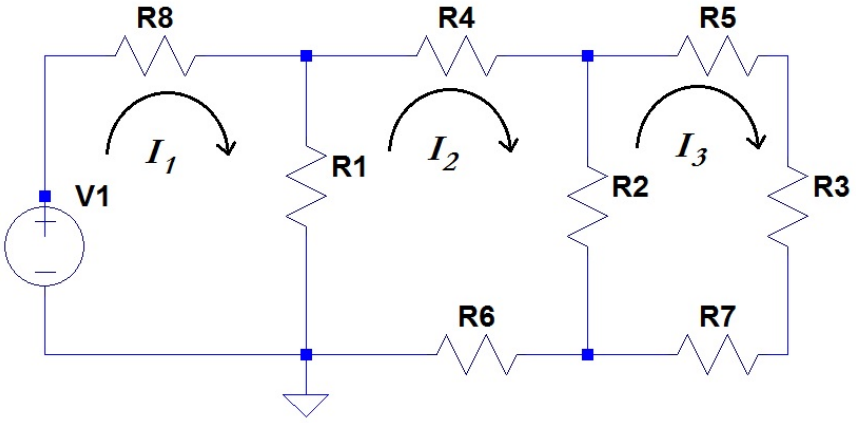


Figura 5.1: Circuito elétrico - Lei das Malhas

necessidade de resolver um sistema de equações. De forma mais compacta, podemos escrever um sistema de equações na forma:

$$\mathbf{Ax = b}$$

onde

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

sendo $\mathbf{A} \in \Re^{n \times n}$ a matriz dos coeficientes, $\mathbf{x}$ o vetor coluna das variáveis e $\mathbf{b}$ é um vetor coluna constante. Esse problema tem uma solução única se $Det(\mathbf{A}) \neq 0$: $\mathbf{x = A^{-1}b}$, pois $\mathbf{A^{-1}A = AA^{-1} = I}$ e $\mathbf{Ix = x}$. Para que o determinante de $\mathbf{A}$ não seja nulo, as equações precisam ser linearmente independentes.

Quando o sistema possui muitas equações, os métodos clássicos, como a regra de Cramer[2] para solução de sistemas apresentam um custo computacional, em termos de multiplicações e somas, muito elevado da ordem de $(n+1) \times (n-1) \times n!$. Por isso, são necessários métodos mais eficientes para resolver este tipo de problema.

5.1 Sistemas Lineares: alguns conceitos básicos e definições

Uma matriz é chamada de **matriz identidade I** quando todos os seus elementos são zeros, com exceção da diagonal principal cujos elementos são todos iguais a 1. O comando "eye" do Scilab cria uma matriz identidade. Se todos os elementos de uma matriz são zeros, então essa matriz é uma

[2]Gabriel Cramer (31 de julho de 1704 - 4 de janeiro de 1752) era um matemático suíço, nascido em Genebra. Ele era o filho do médico Jean Cramer e Anne Mallet Cramer. Cramer mostrou desde muito jovem mostrou ser uma promessa em matemática, concluiu o doutorado aos 18 anos. Cramer trabalhou em análise e determinantes. Ele se tornou professor de matemática em Geneva e escreveu trabalhos relacionados à física, também em geometria e história da matemática. Cramer é melhor conhecido pelo seu trabalho em determinantes (1750) mas também fez contribuições ao estudo das curvas algébricas (1750).

matriz nula. Podemos gerar um matriz nula com o comando "zeros(N,M)".

Uma matriz é chamada de "densa" se a maior parte de seus elementos não são nulos, caso contrário, ela é chamada de "esparsa". Quase sempre, problemas do mundo real gearam matrizes esparsas. Existem técnicas especiais eficientes para resolver este tipo problema (Duff et al., 1989).

Na transposição de matriz, os elementos das linhas e colunas trocam de posição, isto é, uma linha i se torna a coluna i e coluna j se torna a linha j. A transposição é indicada por $\mathbf{M}^T$. Uma matriz é simétrica se $\mathbf{A} = \mathbf{A}^T$, isto é, ela é simétrica em relação aos elementos da diagonal principal. Por exemplo:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 7 & 9 & 0 \\ 3 & 9 & 6 & 5 \\ 4 & 0 & 5 & 8 \end{bmatrix}, \mathbf{A}^T = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 7 & 9 & 0 \\ 3 & 9 & 6 & 5 \\ 4 & 0 & 5 & 8 \end{bmatrix}$$

O traço (ver comando "trace") de uma matriz (quadrada) é definido como sendo a soma dos elementos da diagonal principal:

$$\text{traço}(\mathbf{A}) = \sum_{i=1}^{N} a_{ii}$$

Dizemos que um sistema linear é **mal condicionado** se uma pequena variação nos coeficientes leva a uma solução muito diferente. Isso ocorre quando o sistema está próximo de ser singular. Por exemplo, o sistema

$$\begin{cases} 2,55x & +1,65y & = 8,4 \\ 4,87x & +3,15y & = 16,04 \end{cases}$$

Tem uma solução aproximada $x = 0,3076923$ e $y = 4,6153846$. Entretanto, a solução exata é muito diferente: $x = 2$ e $y = 2$. Logo, esse é um exemplo de sistema mal condicionado.

Definição 5.1.1 Norma de vetor. A norma de um vetor em $\Re^n$ é uma função, indicada por $\|\cdot\|$, de $\Re^n$ em $\Re$ com as seguintes propriedades:

1. $\|\mathbf{x}\| \geq 0$ para todo $\mathbf{x} \in \Re^n$;
2. $\|\mathbf{x}\| = 0$ se e somente se $\mathbf{x} = 0$;
3. $\|\alpha\mathbf{x}\| = |\alpha| \, \|\mathbf{x}\|$ para todo $\alpha \in \Re$ e $\mathbf{x} \in \Re^n$;
4. $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ para todo $\mathbf{x}, \mathbf{y} in \Re^n$.

As normais mais usuais para um vetor são as normas Euclidiana $\|\mathbf{x}\|_2$ e a norma "infinita" $\|\mathbf{x}\|_\infty$:

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2} \quad \text{e} \quad \|\mathbf{x}\|_\infty = \overset{max}{1 \leq i \leq n} |x_i|$$

Por exemplo, a norma Euclidiana do vetor $\mathbf{x} = [1;3;5]^T$ é 5,9160798; já sua norma infinita é 5. O comando "norm" do Scilab pode efetuar esse cálculo: norm(x,2). Podemos definir a distância ou diferença entre dois vetores como sendo $\|\mathbf{x}-\mathbf{y}\|_2$, por exemplo, a distância entre os vetores $\mathbf{x} = [1;3;5]^T$ e $\mathbf{y} = [0,9;3,1;4,8]^T$ é 0,244949. Em relação às normas de um vetor, também podemos afirmar:

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_2 \leq \sqrt{n} \|\mathbf{x}\|_\infty . \tag{5.1}$$

Definição 5.1.2 Norma de matriz. Uma norma de matriz $N \times N$ é uma função, indicada por $\|\cdot\|$, de valores reais com as seguintes propriedades:

1. $\|\mathbf{A}\| \geq 0$ para toda matriz $\mathbf{A}$ $N \times N$;
2. $\|\mathbf{x}\| = 0$ se e somente se $\mathbf{A}$ tiver todos os elementos iguais a 0;
3. $\|\alpha\mathbf{A}\| = |\alpha| \, \|\mathbf{x}\|$ para todo $\alpha \in \Re$;
4. $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$;
5. $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\|$.

Uma forma de definir a norma de uma matriz é tratá-la como como um vetor de comprimento N^2 aplicar as normas usuais de vetores. No Scilab, nós temos:

- norm(x) ou norm(x,2): é o maior valor singular de x (max(svd(x)));
- norm(x,1): a norma 1 de x, é a maior soma coluna a coluna : max(sum(abs(x),'r'));
- norm(x,'inf'), norm(x,%inf): maior soma linha a linha : max(sum(abs(x),'c'));
- norm(x,'fro'): norma de Frobenius, isto é, sqrt(sum(diag(x'*x))).

Para qualquer matriz $\mathbf{A}$ vale a relação:

$$\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F \leq N^{1/2} \|\mathbf{A}\|_2$$

.

Definição 5.1.3 Autovalores. Associado a uma matriz quadrada $N \times N$, temos o polinômio característico definido por

$$P(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I})$$

As raízes de $P(\lambda)$ são os autovalores da matriz $\mathbf{A}$.

Por exemplo, os autovalores da matriz

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

são as raízes do polinômio $(1-\lambda)(4-\lambda)-6=0$: $\lambda_1 = 5,3722813$ e $\lambda_2 = -0,3722813$.

Definição 5.1.4 Raio espectral. O raio espectral $\rho(\mathbf{A})$ de uma matriz $\mathbf{A}$ é definido por

$$\rho(\mathbf{A}) = max|\lambda|$$

em que λ é um autovalor de $\mathbf{A}$.

Por exemplo, o raio espectral da matriz

$$\mathbf{D} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

é igual a $\rho(\mathbf{D}) = 16,116844$, já sua norma $\|\mathbf{D}\|_2 = 16,848103$. A função "spec()" do Scilab calcula os autovalores de qualquer matriz quadrada. Existe uma relação próxima entre o raio espectral a e norma de uma matriz: $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$. Se a matriz for simétrica, então, teremos a igualdade: $\rho(\mathbf{A}) = \|\mathbf{A}\|_2$. Por exemplo, a matriz

$$\mathbf{G} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 0 \\ 3 & 0 & 9 \end{bmatrix}$$

é simétrica, seu raio espectral $\rho(\mathbf{G})$ e sua norma $\|\mathbf{G}\|_2$ são iguais a 10,084609.

Definição 5.1.5 Matriz convergente. Dizemos que uma matriz $\mathbf{A}_{N\times N}$ é convergente se

$$\lim_{k\to\infty} \mathbf{A}^k = \mathbf{0}$$

Se uma matriz $\mathbf{A}$ é convergente, então $\rho(\mathbf{A}) < 1$. Vejamos um exemplo numérico:

$$\mathbf{C} = \begin{bmatrix} 0,5 & 0,95 & 0 \\ -1 & -0,4 & 0,3 \\ -0,2 & 0,6 & 0,8 \end{bmatrix}, \quad \mathbf{C}^2 = \begin{bmatrix} -0,7 & 0,095 & 0,285 \\ -0,16 & 0,61 & 0,12 \\ -0,86 & 0,05 & 0,82 \end{bmatrix}, ...$$

$$\mathbf{C}^{20} = \begin{bmatrix} -0,0035 & -0,0044 & 0,0034 \\ 0,0039 & 0,0028 & -0,0003 \\ -0,0092 & -0,0029 & 0,0129 \end{bmatrix}$$

ou seja, $\lim_{k\to\infty} \mathbf{C}^k = \mathbf{0}$ e $\mathbf{C}$ é um exemplo de matriz convergente, os três autovalores desta matriz $\mathbf{C}$ são: $\lambda_i = (0,0513321 + 0,7520021j; 0,0513321 - 0,7520021j; 0,7973357)$. Esse conceito é importante quando trabalhamos com métodos iterativos para a solução de sistemas lineares. Quanto menor o valor de $|\lambda|_{max}$, mais rapidamente ocorre a convergência. Vejamos um segundo exemplo:

$$\mathbf{G} = \begin{bmatrix} -0,45 & 0,85 & 0,00 \\ 0,15 & 0,36 & -0,27 \\ -0,18 & 0,54 & -0,72 \end{bmatrix}, \quad \mathbf{G}^2 = \begin{bmatrix} 0,3315 & -0,0774 & -0,2322 \\ 0,0351 & 0,1128 & 0,0972 \\ 0,2916 & -0,3492 & 0,3726 \end{bmatrix}, ,.,$$

$$\mathbf{G}^{20} = \begin{bmatrix} 0,00009 & -0,00002 & -0,00021 \\ 0,00004 & -0,00006 & 0,00007 \\ 0,00025 & -0,00029 & 0,00012 \end{bmatrix}$$

A matriz $\mathbf{G}$ converge rapidamente para zero, seus autovalores são: $\lambda_i = (0,4222068; -0.6161034 + 0.2454846j; -0.6161034 - 0.2454846j)$.

5.2 Solução usando eliminação de Gauss

Se a matriz $\mathbf{A}$ do sistema $\mathbf{Ax} = \mathbf{b}$ for uma matriz triangular superior, com todos os elementos da diagonal principal diferentes de zero ($a_{kk} \neq 0$), isto é,

$$\begin{cases} a_{11}x_1 & +a_{12}x_2 & +a_{13}x_3 & +\cdots & +a_{1n}x_n & = b_1 \\ & a_{22}x_2 & +a_{23}x_3 & +\cdots & +a_{2n}x_n & = b_2 \\ & & a_{33}x_3 & \cdots & +a_{3n}x_n & = b_3 \\ & & & \ddots & \ddots & \vdots \\ & & & & a_{nn}x_n & = b_n \end{cases}$$

A solução é trivial: calculamos $x_n = b_n/a_{nn}$, em seguida

$$x_{n-1} = \frac{b_{n-1} - a_{n-1n}x_n}{a_{nn}}$$

para os próximos, seguimos com a substituição progressiva:

$$x_k = \frac{b_k - \sum_{j=k+1}^{n} a_{kj}x_j}{a_{kk}} \text{ para } k = n-2, ..., 1$$

Claro que nem todos os sistemas lineares práticos se apresentam na forma uma matriz triangular superior, mas, pelo menos em princípio, podemos trabalhar para transformar qualquer matriz em uma triangular superior usando uma técnica de eliminação progressiva. Assim, o método de eliminação de Gauss consiste em usar eliminação progressiva e, em seguida, substituição progressiva para obtermos a solução de um sistema de equações lineares, como ilustra a Figura 5.2.

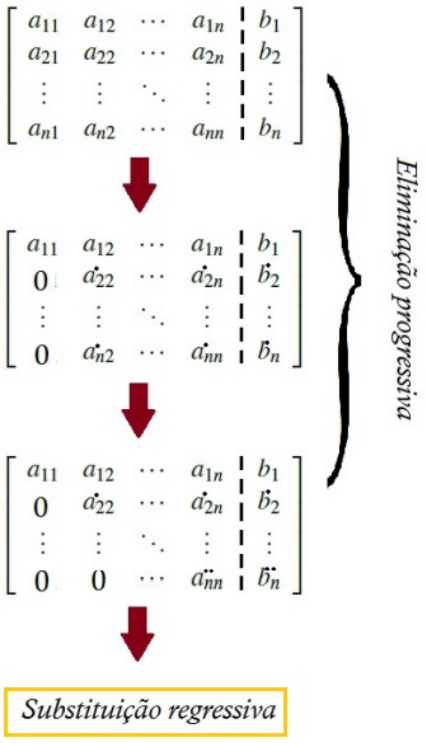


Figura 5.2: Processo de eliminação de Gauss

Na eliminação progressiva, procuramos "zerar" uma variável por vez de cada linha. Desta forma, na segunda linha eliminamos uma variável, na terceira linha duas variáveis são eliminadas e assim por diante, até obtermos uma sistema triangular superior de sistema de N equações. Podemos usar o seguinte algoritmo para isso:

Para k variando de 1 a $N-1$, faça
$d \leftarrow a_{k+1k}/a_{kk}$
$linha_{k+1} \leftarrow linha_{k+1} - d \times linha_k$, da coluna k até $N+1$

No algoritmo acima não está previso o caso de a_{kk} ser nulo ou próximo de zero. Nesses casos, é necessária uma troca de linhas. Vejamos um exemplo didático:

$$\begin{cases} 3x_1 & +2x_2 & +x_3 & = 7 \\ x_1 & +x_2 & +x_3 & = 4 \\ x_1 & +2x_2 & +2x_3 & = 7 \end{cases}$$

$$\begin{cases} 3x_1 & +2x_2 & +x_3 & = 7 \\ 0 & +1/3x_2 & +2/3x_3 & = 5/3 \\ 0 & +4/3x_2 & +5/3x_3 & = 14/3 \end{cases}$$

$$\begin{cases} 3x_1 & +2x_2 & +x_3 & = 7 \\ 0 & +1/3x_2 & +2/3x_3 & = 5/3 \\ 0 & +0 & -x_3 & = -2 \end{cases}$$

Logo, $x_3 = 2$ e

$$x_2 = \frac{5/3 - 4/3}{1/3} = 1$$

$$x_1 = \frac{7 - 2 - 2}{3} = 1$$

Código Scilab. O *script* a seguir consegue resolver um sistema de equações lineares $N \times N$ usando o método descrito acima. Neste código exemplo, o sistema é gerado aleatoriamente. Não existe a preocupação para evitar divisão por zero.

```
/// Sistema de equações lineares
/// solução usando o método de Gauss
clc;  // limpa a tela
N = 5; rand('seed',23); // Matriz 5 x 5
Ms = round(10*rand(N,N+1,'n')); // valores aleatórios
disp(Ms); // mostrando o sistema no console
for k=1:N-1  // Laço para gerar uma triangular superior
    for p=k+1:N
        a = Ms(p,k)/Ms(k,k);
       Ms(p,k:N+1) = Ms(p,k:N+1) - a*Ms(k,k:N+1);
   end
end
disp(Ms);  // mostrando a matriz no formato triangular sup.

x = zeros(N,1);  // inicializando o vetor resposta
x(N) = Ms(N,N+1)/Ms(N,N);  // cálculo de xN
for k=(N-1):-1:1  // substituição progressiva
    S = 0;
    for p=N:-1:(k+1)
        S = S + x(p)*Ms(k,p)
    end
    x(k) = (Ms(k,N+1)-S)/Ms(k,k);
end
disp(x);  // resposta final
```

O tempo para a solução de um sistema linear é, teoricamente, polinomial, pois o número de operações (somas, multiplicações e divisões) cresce com $O(N^3)$. Os gráficos da Figura 5.3 mostram o tempo necessário para resolver o sistema[3].

[3]Sistema Windows 8.1. Intel(R) Core(TM) i5-4200U CPU @ 1,60GHz, 6,00 GB de memória RAM

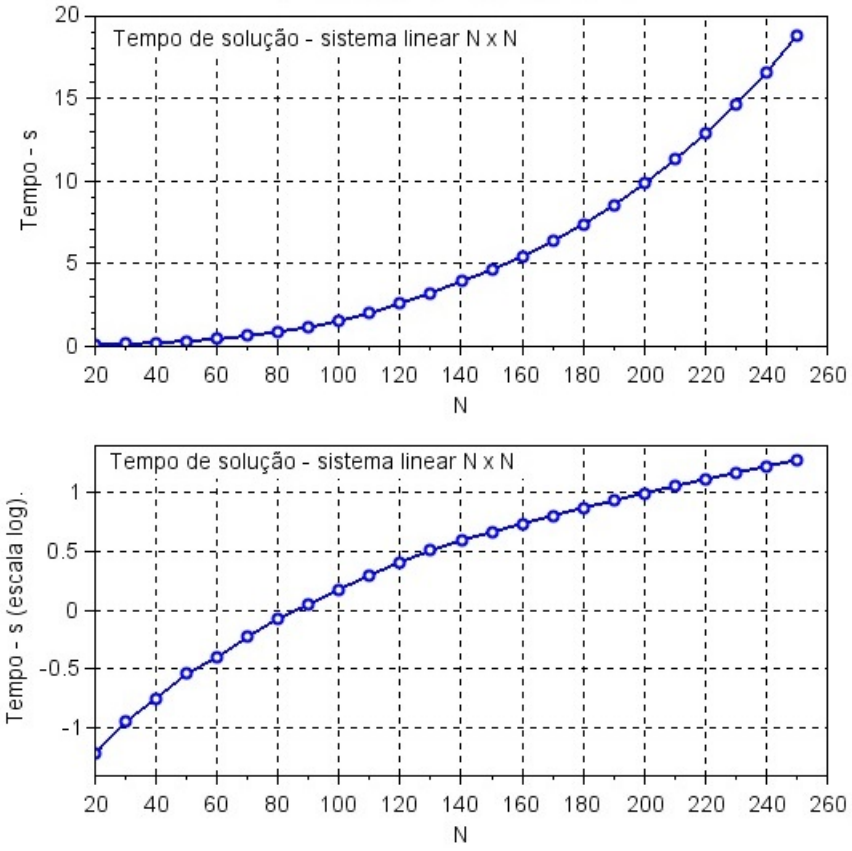


Figura 5.3: Tempo de solução de um sistema linear usando o *script* Scilab.

5.3 Solução iterativa

Em algumas situações, especialmente quando temos uma matriz esparsa, podemos optar por encontrar a solução de um sistema usando um método iterativo. Um exemplo desse tipo de método é o Gauss-Seidel[4]. Nos métodos iterativos, a partir de uma solução inicial x_0, calculamos sucessivamente os novos x_k, $k = 1, 2, ...$ valores até que algum critério de parada seja satisfeito. Antes de detalhar a teoria, veremos um exemplo simples:

$$\begin{cases} 4x_1 & +2x_2 & = 8 \\ x_1 & +3x_2 & = 7 \end{cases}$$

Usando $x_0^0 = 0$ e $x_1^0 = 0$ como valores iniciais:

$$x_1^1 = (8-0)/4 = 2$$
$$x_2^1 = (7-2)/3 = 5/3$$

Em seguida:

$$x_1^2 = (8-10/3)/4 = 7/6$$
$$x_2^2 = (7-7/6)/3 = 35/18$$

Com mais uma iteração:

$$x_1^3 = (8-35/9)/4 = 37/36$$
$$x_2^3 = (7-37/36)/3 = 215/108$$

Ou seja, usamos as fórmulas

$$x_1^k = (8-2x_2^{k-1})/4 \tag{5.2}$$
$$x_2^k = (7-x_1^k)/3 \tag{5.3}$$

para chegarmos nos resultados $x_1^3 \cong 1,0277778$ e $x_2^3 \cong 1.9907407$. Como mais algumas iterações chegaríamos ao resultado esperado $x_1 = 1$ e $x_2 = 2$ com um erro ε muito pequeno. Esse é o método Gauss-Seidel. Claro, nem todos os sistemas convergirão tão rapidamente. Devemos adotar algumas estratégias e cuidados para facilitar a convergência. Um ponto importante: a diagonal principal deve ser dominante para que ocorra a convergência. Por exemplo, em alguns casos, a ordem com que as equações são escritas podem influenciar muito na velocidade de convergência desse método iterativo. Uma vantagem do método iterativo é que os erros de arredondamento não são acumulados como no caso do método de eliminação, mas, em geral, não é possível a solução exata com um número finito de passos. Para sistemas de grande porte é, provavelmente, mais eficientes que os métodos diretos, principalmente com a utilização de computação de alto desempenho (vetorial e paralela).
Vejamos agora um exemplo no qual a convergência não ocorre (resultados na Tabela 5.1):

$$\begin{cases} 2x_1 & +3x_2 & +x_3 & = 6 \\ x_1 & +x_2 & +2x_3 & = 4 \\ x_1 & +2x_2 & +x_3 & = 4 \end{cases} \tag{5.4}$$

Tabela 5.1: Sistema 5.4 não convergente. Primeiras iterações.

k	x_1	x_2	x_3
0	0	0	0
1	3	1	-1
2	2	4	-6
3	0	16	-28
4	-7	67	-123
5	-36	286	-532
6	*	*	*

Por que a solução *simples* $\{1,1,1\}$ não foi obtida? Basicamente porque a diagonal principal não é dominante. Deve ficar claro que a convergência para a solução exata não é garantida para qualquer sistema. Veremos dois critérios que garantem a convergência, antes, porém, uma palavra sobre critério de parada.

5.3.1 Critério de parada

Todo método iterativo precisa ter algum critério de parada. Nem sempre a solução é encontrada, mas o processo tem que ser finalizado e alguma mensagem de erro gerada, ou o resultado desejado deve ser apresentado. Para o caso do método iterativo de Gauss-Seidel, podemos escolher como critério de parada o número máximo (n_{max}) de iterações (10, 20, 30 ou outro valor) ou que a diferença absoluta entre x_i^k e x_i^{k-1} seja menor que um certo ε escolhido previamente. Podemos também usar o erro relativo como critério de parada:

$$\varepsilon_r > \frac{x_i^k - x_i^{k-1}}{x_i^k},$$

desde que esse x_i não tenta a zero.

5.3.2 Critério das linhas

Ó critério mais simples para saber se podemos obter a solução de um sistema linear usando Gauss-Seidel é o das linhas. Basicamente, se os elementos da diagonal principal forem maiores que a soma dos módulos dos coeficientes da mesma linha, então a solução do sistema poderá ser atingida por um número finito de iterações com um certo erro ε. Matematicamente:

$$|a_{ii}| > \sum_{j=1, j\neq i}^{N} |a_{ij}| \tag{5.5}$$

Esse critério tem uma certa "folga", mesmo sistemas que não o satisfazem podem convergir. Sistemas que obedecem esse critério sempre poderão ser solucionados iterativamente usando Gauss-Seidel.

[4]Philipp Ludwig von Seidel (Zweibrücken, 23 de outubro de 1821 Munique, 13 de agosto de 1896) foi um matemático alemão. Seidel entrou na Universidade de Berlim em 1840 e estudou sob Dirichlet e Encke. Ele obteve seu doutorado em Munique em 1846 com a tese *Über die beyond Form der Spiegel em Teleskopen*. Ele lecionou sobre a teoria da probabilidade e também sobre o método dos mínimos quadrados.

Vejamos um exemplo. O sistema definido por

$$\begin{cases} 3x_1 & +2x_2 & +x_3 & = 6 \\ x_1 & +2x_2 & +x_3 & = 4 \\ x_1 & +2x_2 & +3x_3 & = 6 \end{cases} \tag{5.6}$$

consegue convergir, embora de forma relativamente lenta (convidamos o leitor a efetuar os cálculos). Já o sistema

$$\begin{cases} 4x_1 & +2x_2 & +x_3 & = 7 \\ x_1 & +3x_2 & +x_3 & = 5 \\ x_1 & +2x_2 & +5x_3 & = 8 \end{cases} \tag{5.7}$$

segue o critério das linhas e consegue convergir mais rapidamente (como o leitor poderá verificar facilmente).

5.3.3 Critério de Sassenfeld

O critério de Sassenfeld é um pouco mais difícil de calcular, mas também é um pouco mais rigoroso que o critério das linhas. Esse critério é estabelecido da seguinte forma:

$$\beta_1 = \frac{1}{|a_{11}|}\sum_{j=2}^{N}|a_{ij}|$$

$$\beta_i = \frac{1}{|a_{ii}|}\left[\sum_{j=1}^{i-1}|a_{ij}|\beta_j + \sum_{j=i+1}^{N}|a_{ij}|\right], i = 2,3,\ldots,N$$

$$M = max\{\beta_i\} < 1$$

Exemplo. Vejamos o sistema:

$$\begin{cases} 2x_1 & +x_2 & -0,5x_3 & +0,4x_4 & = 4,1 \\ 0,6x_1 & +3x_2 & -0,6x_3 & -0,3x_4 & = 3,6 \\ -0,7x_1 & -x_2 & +2x_3 & +0.5x_4 & = 5,3 \\ 0,4x_1 & +1,2x_2 & +0,8x_3 & +3x_4 & = 17,2 \end{cases} \tag{5.8}$$

Os valores de β_i são:

$$\beta_1 = \frac{1}{2}(1+0,5+0,4) = 0,95$$

$$\beta_2 = \frac{1}{3}(0,6\times 0,95+0,6+0,3) = 0,49$$

$$\beta_3 = \frac{1}{2}(0,7\times 0,95+1\times 0,49+0,5) = 0,8275$$

$$\beta_4 = \frac{1}{3}(0,4\times 0,95+1,2\times 0,49+0,8\times 0,8275) = 0,54333$$

Logo, $M = 0,95 < 1$ e esse sistema consegue convergir para solução usando o método iterativo de Gauss-Seidel. Note que esse sistema não obedece o critério das linhas. Os valores obtidos estão na Tabela 5.2. O código fonte *Scilab* para resolver esse sistema é mostrado em seguida. A Figura 5.4 mostra essa convergência graficamente.

Código Scilab:

Tabela 5.2: Sistema 5.8 segue o critério de Sassenfeld e converge em 9 iterações

k	x_1	x_2	x_3	x_4	ε
0	0	0	0	0	*
1	2,0500	0,7900	3,7625	4,1407	4,1407
2	1,7675	2,0131	3,2400	3,8284	1,2231
3	1,0878	2,0133	3,0803	3,9616	0,6797
4	1,0211	2,0080	3,0210	3,9884	0,0667
5	1,0036	2,0023	3,0053	3,9972	0,0175
6	1,0007	2,0006	3,0013	3,9993	0,0040
7	1,0001	2,0002	3,0003	3,9998	0,0010
8	1,0000	2,0000	3,0001	4,0000	0,0002
9	1,0000	2,0000	3,0000	4,0000	0,0001

```
/// Critério de Sassenfeld:
/// Entrada: matriz Mx quadrada
/// Saída: Bk : betas

function Bk = critLinha(Mx)
    Mx = abs(Mx);  // todos os valores são positivos
    tam = max(size(Mx));  // dimensão da matriz
    Bk = ones(1,tam);  // iniciando os betas
    Bk(1) = (sum(Mx(1,:))-Mx(1,1))/Mx(1,1);
    for k=2:tam
        Bk(k) = (sum(Mx(k,:).*Bk)-Mx(k,k)*Bk(k))/Mx(k,k);
    end
endfunction

clc; close; // Limpando o console e fechando janelas

M = [2 1 -0.5 0.4
     0.6 3 -0.6 -0.3
     -0.7 -1 2 0.5
     0.4 1.2 0.8 3]
b = M*[1;2;3;4]; // vetor
N = 4; // ordem da matriz

Cx = critLinha(M);
disp(Cx);

xk = 0*b;  // solução inicial

nmax = 20; // número máximo de paradas
erro = 1; // erro inicial
nk = 2;  // número da iteração
```

```
mvx = zeros(N,nmax); // guardando os valores de xk
mvx(:,1) = xk;
while erro>0.0001
    erro = xk;
    for n=1:N
        s = M(n,:)*xk;
        s = s - M(n,n)*xk(n);
        xk(n) = (b(n) - s)/M(n,n);
    end
    mvx(:,nk) = xk;
    erro = max(abs(erro - xk));
    printf('%d & %1.4f & %1.4f & %1.4f & %1.4f & %1.4f \n',...
    nk,xk(1),xk(2),xk(3),xk(4),erro);
    nk = nk + 1;
    if nk > nmax then
        erro =0;
    end; // número máximo de iterações
end
plot(1:nk-1,mvx(:,1:nk-1),'-o');
```

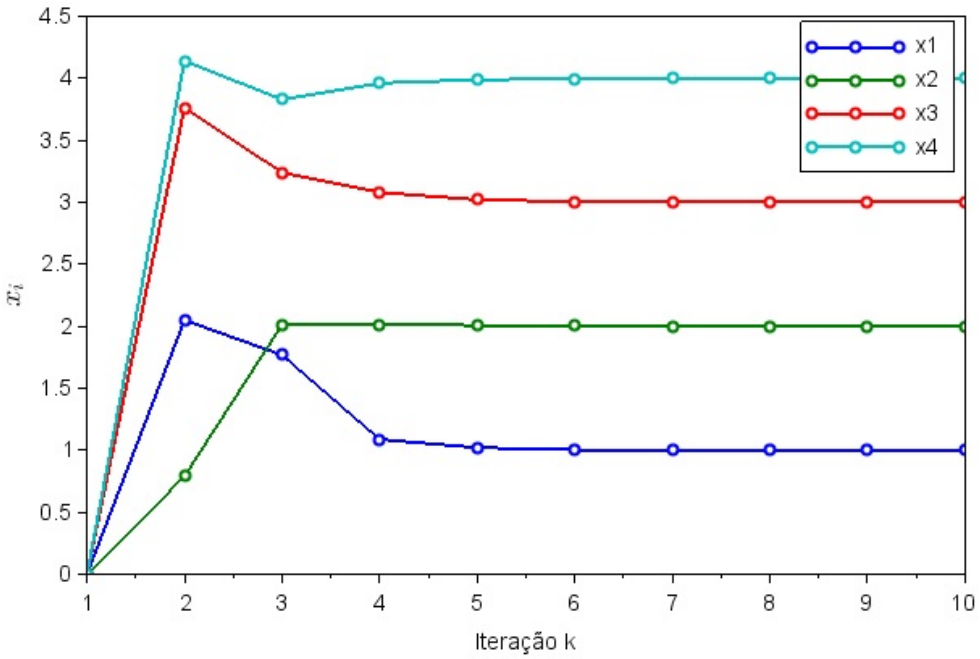


Figura 5.4: Convergência da solução do sistema exemplo

5.4 Forma matricial do método iterativo

Primeiramente, notamos que podemos escrever qualquer matriz quadrada **A** como: **A** = **D** - **L** - **U**, onde **D** é uma matriz diagonal, **L** é uma matriz triangular inferior (com a diagonal nula) e **U** é uma

matriz triangular superior (com a diagonal nula). Assim, um sistema na forma **Ax = b** pode ser escrito como (**D + L + U**)**x = b**. Então, podemos usar a seguinte forma iterativa:

$$\begin{aligned}(\mathbf{D}+\mathbf{L}+\mathbf{U})\mathbf{x} &= \mathbf{b}\\ (\mathbf{D}+\mathbf{L})\mathbf{x}^{k+1} &= -\mathbf{U}\mathbf{x}^{k}+\mathbf{b}\\ \mathbf{x}^{k+1} &= -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\mathbf{x}^{k}+(\mathbf{D}+\mathbf{L})^{-1}\mathbf{b}\\ \mathbf{x}^{k+1} &= \mathbf{T}_{gs}\mathbf{x}^{k}+\mathbf{c}_{gs}\end{aligned}$$

onde $\mathbf{T}_{gs} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}$ é a matriz de iteração e $\mathbf{c}_{gs} = (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b}$. Claro que essa é uma forma compacta de notação, muito útil para escrever um *script*, mas tem uma desvantagem: temos que calcular a inversa de uma matriz antes de começarmos os cálculos. Vejamos um exemplo simples usando o Scilab: resolver o sistema abaixo. A resposta, na forma de um gráfico, é apresentada na Figura 5.5.

$$\begin{cases} 4x_1 & +3x_2 & +2x_3 & +x_4 & = 15\\ 2x_1 & +4x_2 & +3x_3 & +x_4 & = 18\\ x_1 & +2x_2 & +3x_3 & +2x_4 & = 12\\ x_1 & +x_2 & +2x_3 & +3x_4 & = 6 \end{cases}$$

```
clc; close;

N = 4;  // Sistema 4 x 4:
M = [4 3 2 1
     2 4 3 1
     1 2 3 2
     1 1 2 3];
v = M*[1;2;3;-1];
N = 4;

x = inv(M)*v; // solução para comparação

xk = 0*v;

D = M.*eye(N,N);  // diagonal
L = tril(M) - D;  // matriz triang. inferior
U = triu(M) - D; // matriz tring. superior
Tg = -inv(D+L)*U;  // matriz de iteração
bg = inv(D+L)*v

nmax = 30;
erro = 1;
nk = 1;
mvx = zeros(N,nmax); mvx(:,1) = xk;
while erro>0.001
    erro = xk;
    xk = Tg*xk + bg;
```

```
    mvx(:,nk) = xk;
    erro = max(abs(erro - xk));
    nk = nk + 1;
    if nk > nmax then erro =0; end;
end
disp([x, xk]);  // mostrando e comparando os resultados
figure; plot(1:nk-1,mvx(:,1:nk-1),'-o');
```

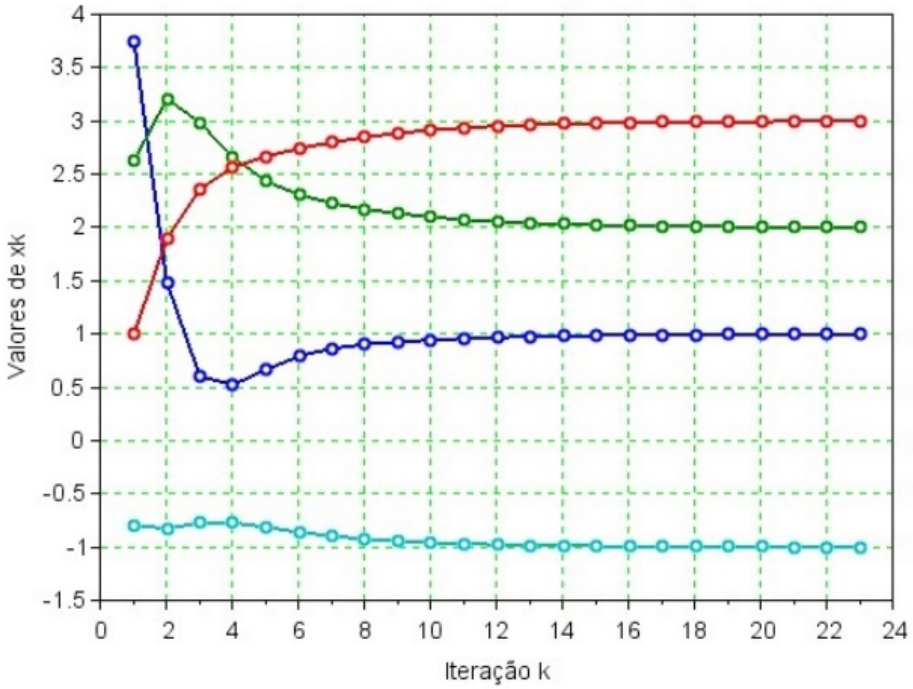


Figura 5.5: Convergência da solução - exemplo usando notação matricial.

5.5 Usando o Scilab para resolver sistemas lineares

Quando estamos interessados apenas em resolver um sistema de equações lineares usando os recursos do Scilab, temos várias ferramentas à disposição, dentre elas:

- det: calcula o determinante de uma matriz;
- diag: usado para incluir ou extrair a diagonal de uma matriz;
- rank: calcula o posto de uma matriz;
- inv: encontra a inversa de uma matriz;
- linsolve: é um solucionador de equações lineares;
- lusolve: é solucionador de sistemas lineares esparsos;
- norm: calcula a norma de uma matriz;
- sparse: definição de matriz esparsa a partir de uma matriz M;
- spec: retorna os autovalores de uma matriz;
- pinv: calcula a pseudo-inversa de uma matriz;

- trace: computa o traço de uma matriz;
- tril: extrai a parte triangular inferior de uma matriz (inclui a diagonal principal);
- triu: extrai a parte triangular superior de uma matriz (inclui a diagonal principal).

Vejamos um exemplo do uso de alguns desses comandos na solução de uma matriz esparsa 16×16. **Código Scilab:**

```
////////////// matriz esparsa
clc;

M = 4*eye(16,16);  // Cria uma matriz diagonal 16 x 16
// Elementos diferentes de zero:
M(1,2) = -1; M(1,5)=-1;
M(2,1) = -1; M(2,3)=-1; M(2,6)=-1;
M(3,2) = -1; M(3,4)=-1; M(3,7)=-1;
M(4,3) = -1; M(4,8)=-1;
M(5,1) = -1; M(5,6)=-1; M(5,9)=-1;
M(6,2) = -1; M(6,5)=-1; M(6,6)=-1; M(6,10)=-1;
M(7,3) = -1; M(7,6)=-1; M(7,8)=-1; M(7,11)=-1;
M(8,4) = -1; M(8,7)=-1; M(8,12)=-1;
M(9,5) = -1; M(9,10)=-1; M(9,13)=-1;
M(10,6) = -1; M(10,9)=-1; M(10,14)=-1; M(10,11)=-1;
M(12,8) = -1; M(12,11)=-1; M(12,16)=-1;
M(13,9) = -1; M(13,14)=-1;
M(14,10) = -1; M(14,13)=-1; M(14,15)=-1;
M(15,11) = -1; M(15,14)=-1; M(15,16)=-1;
M(16,12) = -1; M(16,15)=-1;

v = zeros(16,1); v(1)=1;

x1 = inv(M)*v;  // Solução pelo comando "inv"
x2 = M\v;   // Solução divisão com barra invertida

sp = sparse(M);  // Solução da matriz esparsa
x3=lusolve(sp,v);  // usando sparse e lusolve

x4 = linsolve(M,-v);  // solução usando linsolve

disp([x1, x2, x3, x4]); // mostrando os resultados
```

As soluções obtidas (x1, x2, x3 e x4) são essencialmente as mesmas:

```
    0.2752279    0.2752279    0.2752279    0.2752279
    0.0504558    0.0504558    0.0504558    0.0504558
    0.0078817    0.0078817    0.0078817    0.0078817
    0.0006964    0.0006964    0.0006964    0.0006964
    0.0504558    0.0504558    0.0504558    0.0504558
  - 0.0812864  - 0.0812864  - 0.0812864  - 0.0812864
```

```
  - 0.0196252  - 0.0196252  - 0.0196252  - 0.0196252
  - 0.0050962  - 0.0050962  - 0.0050962  - 0.0050962
    0.0078817    0.0078817    0.0078817    0.0078817
  - 0.0196252  - 0.0196252  - 0.0196252  - 0.0196252
    0.           0.           0.         - 4.524D-19
  - 0.0014561  - 0.0014561  - 0.0014561  - 0.0014561
    0.0006964    0.0006964    0.0006964    0.0006964
  - 0.0050962  - 0.0050962  - 0.0050962  - 0.0050962
  - 0.0014561  - 0.0014561  - 0.0014561  - 0.0014561
  - 0.0007280  - 0.0007280  - 0.0007280  - 0.0007280
```

5.6 Pseudo inversa

Em algumas situações podemos nos deparar com problemas nos quais o número de equações e de variáveis são diferentes. Nesses casos, a matriz **A** de **Ax = b** não é quadrada e podemos ter infinitas soluções. Uma solução possível pode ser obtida por:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

onde $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ é uma matriz conhecida como a pseudo-inversa ou a inversa generalizada de Moore-Penrose de **A**. Note que $\mathbf{A}^T\mathbf{A}$ gera uma matriz quadrada. Podemos usar o comando "pinv()" para esse cálculo. Vejamos um exemplo:

```
M = rand(2,3)
M  =
    0.2113249    0.0002211    0.6653811
    0.7560439    0.3303271    0.6283918
b = rand(2,1)
b  =
    0.8497452
    0.6857310
x = pinv(M)*b
 x  =
  - 0.0485449
  - 0.2719094
    1.2925887
```

Algumas vezes temos mais equações que incógnitas, como no exemplo a seguir (são 6 equações e somente 4 incógnitas):

```
A = [0.2 0.0 0.0 0.0
     0.8 0.2 0.0 0.0
     -0.1 0.8 0.2 0.0
     0.0 -0.1 0.8 0.2
     0.0 0.0 -0.1 0.8
     0.0 0.0 0.0 -0.1];
v = [0; 0; 1; 0; 0; 0];
w = pinv(A)*v
```

A solução no vetor **w** é: $[-0,2770464; 1,1754062; 0,1435250; 0,0174292]^T$.

5.7 Sistemas não lineares

Em alguma situações, nos deparamos com um sistema de equações não lineares. Por exemplo, quais são os pontos de interseção do círculo $x^2+y^2=2$ e da parábola $y-x^2=0$? O cálculo desses pontos envolve a solução do sistema de equações não lineares:

$$\begin{cases} x^2+y^2-2 &= 0 \\ -x^2+y &= 0 \end{cases}$$

Tendo isso em mente, iremos apresentar nesta seção uma forma de solução de um sistema de equações do tipo:

$$\begin{cases} f_1(x_1,x_2,\ldots x_n) &= 0 \\ f_2(x_1,x_2,\ldots x_n) &= 0 \\ f_3(x_1,x_2,\ldots x_n) &= 0 \\ \vdots & \vdots \\ f_n(x_1,x_2,\ldots x_n) &= 0 \end{cases}$$

usando os métodos de Newton/Secante vistos no capítulo anterior. Lembramos que o método de Newton pode ser resumido a

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

onde x_0 é um "chute" inicial, $f'(x_k)$ é derivada (ou uma estimativa da derivada) de f(x) em $x = x_k$ que podemos calcular por (h é um valor "pequeno"):

$$\frac{df(x)}{dx} \cong \frac{f(x+h)-f(x-h)}{2h}$$

Adaptando para o caso de um sistema de equações, sem incluir algumas simplificações possíveis, podemos escrever:

$$\begin{bmatrix} x_1^{k+1} \\ x_2^{k+1} \\ \vdots \\ x_n^{k+1} \end{bmatrix} = \begin{bmatrix} x_1^{k} \\ x_2^{k} \\ \vdots \\ x_n^{k} \end{bmatrix} - \begin{bmatrix} \frac{\partial f_1}{\partial x_1^k} & \frac{\partial f_1}{\partial x_2^k} & \cdots & \frac{\partial f_1}{\partial x_n^k} \\ \frac{\partial f_2}{\partial x_1^k} & \frac{\partial f_2}{\partial x_2^k} & \cdots & \frac{\partial f_2}{\partial x_n^k} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1^k} & \frac{\partial f_n}{\partial x_2^k} & \cdots & \frac{\partial f_n}{\partial x_n^k} \end{bmatrix}^{-1} \begin{bmatrix} f_1(x_1^k,\ldots,x_n^k) \\ f_2(x_1^k,\ldots,x_n^k) \\ \vdots \\ f_n(x_1^k,\ldots,x_n^k) \end{bmatrix} \tag{5.9}$$

ou de forma mais compacta:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{J}_i\mathbf{F}(\mathbf{x}^k) \tag{5.10}$$

onde

$$\mathbf{J}_i = \begin{bmatrix} \frac{\partial f_1}{\partial x_1^k} & \frac{\partial f_1}{\partial x_2^k} & \cdots & \frac{\partial f_1}{\partial x_n^k} \\ \frac{\partial f_2}{\partial x_1^k} & \frac{\partial f_2}{\partial x_2^k} & \cdots & \frac{\partial f_2}{\partial x_n^k} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1^k} & \frac{\partial f_n}{\partial x_2^k} & \cdots & \frac{\partial f_n}{\partial x_n^k} \end{bmatrix}^{-1}, \quad \mathbf{F}(\mathbf{x}) = \begin{bmatrix} f_1(x_1^k,\ldots,x_n^k) \\ f_2(x_1^k,\ldots,x_n^k) \\ \vdots \\ f_n(x_1^k,\ldots,x_n^k) \end{bmatrix}$$

A matriz $\mathbf{J}$ é conhecida como a matriz Jacobiana do sistema e $\mathbf{J}_i$ é a sua inversa. Vejamos um exemplo simples:

$$\begin{cases} \cos(\pi x) + \ln(y) - 2 &= 0 \\ x^3 + \sqrt{y} - 3 &= 0 \end{cases}$$

Quais os valores de x e y que satisfazem o sistema acima? Podemos ter como valores iniciais: $x_0 = 0,25$ e $y_0 = 0,75$. O código abaixo implementa a 5.9. Nesse *script*, incluímos um controle de parada: se o erro for menor que 10^{-4} ou se o número de iterações for maior que 10, o laço iterativo é encerrado. O Método de Newton tende a convergir rapidamente, mas o seu raio de convergência, isto é, a distância entre o chute inicial e a solução, é pequeno. Note que o jacobiano é calculado de forma aproximada e não analiticamente.

```
// função 1: entrada: "x1" e "x2":
function f1 = fx1(x1,x2)
    f1 = cos(%pi*x1) + log(x2) -2;
endfunction

// função 2: entrada: "x1" e "x2":
function f2 = fx2(x1,x2)
  f2 = x1*x1*x1 + sqrt(x2) -3;
endfunction

clc;  // limpando a tela

// valores iniciais e passo h:
x1 = 0.25; x2 = 0.75; h = 0.0001; h2 = 2*h;

// vetor inicial, erro inicial e contador do laço
x = [x1; x2]; er = 1; k=0;
while abs(er)>0.0001
      J = [(fx1(x1+h,x2)-fx1(x1-h,x2))/h2, (fx1(x1,x2+h)-fx1(x1,x2-h))/h2
      (fx2(x1+h,x2)-fx2(x1-h,x2))/h2, (fx2(x1,x2+h)-fx2(x1,x2-h))/h2];
      Ji = inv(J);  // calculando a inversa do jacobiano
      Fx = [fx1(x1,x2); fx2(x1,x2)];
      e = - Ji*Fx;  // cálculo do erro
      x = x + e;    // atualizado x
      x1 = x(1);    // atualizando x1
      x2 = x(2);    // atualizando x2
      disp(x);      // mostrando x
      er = max(abs(e));  // valor máximo do vetor erro
      k=k+1;  // atualizando o contador
      if k>10 then er = 0; end;  // muitas iterações: fim do laço
end;
disp([k, er, x1, x2]);  // mostrando os resultados
```

Os valores obtidos são:

```
1.4975422
4.0139379

0.665093
16.991348

0.5612562
6.451555

0.5341027
8.0206044

0.5303683
8.1269062

0.5303543
8.1271989
```

Logo, a solução é $x = 0,5303543$ e $y = 8,1271989$, obtida após 6 iterações. Note que a convergência não é garantida, mas, quando ela ocorre, o método tende a ter uma velocidade quadrática. Sistemas não lineares mais complexos tendem a precisar de mais iterações para convergência, além disso, se o chute inicial não perto o suficiente, a solução pode não ser alcançada. Por exemplo, o sistema abaixo

$$\begin{cases} \cos(\pi x_1/2) + \ln(x_2 x_3) + x_3^2 - 11 &= 0 \\ x_1 x_2 x_3 + \sqrt{x_1 x_3} + \cos(\pi x_3/2) - 8 &= 0 \\ x_1 x_3 + \exp(-x_1 x_2) - \cos(\pi x_2 x_3/6) - 4 &= 0 \end{cases}$$

consegue convergir para a solução $\mathbf{x} = [0,9657248; 2,1569066; 3,0124347]^T$, após 9 iterações, quando $\mathbf{x}_0 = [0,5; 0,5; 0,5]^T$. Já para $\mathbf{x}_0 = [0,25; 0,5; 0,5]^T$ não conseguimos encontrar a solução. Se o valor inicial for melhor, por exemplo, $\mathbf{x}_0 = [0,5; 0,5; 1,5]^T$, alcançamos a convergência com um menor número de iterações. Para obtermos esses resultados, fizemos uma adaptação no *script* apresentado anteriormente.

5.8 Problemas

Questão 1. Converta o sistema abaixo em um sistema triangular superior:

$$\begin{cases} 2x_1 + 2x_2 - x_3 = 6 \\ x_1 + 4x_2 - x_3 = 4 \\ x_1 - x_2 + 4x_3 = -1 \end{cases}$$

Questão 2. Encontre os valores para x_i do sistema abaixo usando o método de triangularização de Gauss

$$\begin{cases} 5x_1 + 2x_2 + x_3 = 9 \\ 2x_1 + x_2 - x_3 = 4 \\ x_1 - x_2 - 2x_3 = -1 \end{cases}$$

Questão 3. Converta o sistema abaixo em um sistema triangular superior e resolva-o:

$$\begin{cases} 7x_1 + 3x_2 - 2x_3 = 15 \\ x_1 + 4x_2 - 2x_3 = 6 \\ 2x_1 - 2x_2 + 6x_3 = -4 \end{cases}$$

Questão 4. Os coeficientes de um sistema de equações lineares é definido por: $a_{ij} = 2^{-|i-j|}$, com $1 \leq i, j \leq 5$, formando o sistema **Ax = b**, com **b** = $[2,1,1/2,1/4,1/8]^T$. Mostre que a solução é **x** = $[2,0,0,0,0]^T$.

Questão 5. A função $f(x) = ax^2 + bx + c$ passa pelos pontos $(-1,2)$, $(1,3)$ e $(2,5)$, encontre os valores de a, b e c usando o método direto.

Questão 6. A função $f(x) = ax^3 + bx^2 + cx + d$ passa pelos pontos $(-2,-5)$, $(1,3)$, $(2,2)$ e $(3,9)$, encontre os valores de a, b, c e d usando o método iterativo.

Questão 7. O sistema do problema 01 obedece aos critérios das linhas ou de Sassenfeld?

Questão 8. Considere a matriz simétrica abaixo:

$$\begin{bmatrix} 1,000 & 0,870 & 0,870 & 0,870 \\ 0,870 & 0,948 & 0,666 & 0,435 \\ 0,870 & 0,666 & 0,281 & -0,435 \\ 0,870 & 0,435 & -0,435 & -0,739 \end{bmatrix}$$

calcule a sua norma usando o comando "norm(x,2)" do Scilab e seus autovalores com a função "spec"; verifique que a norma desta matriz é exatamente igual ao maior autovalor.

Questão 9. Observe o circuito elétrico abaixo. Considere que todos os resistores são iguais $R = 1\Omega$ e que a fonte de tensão fornece V = 10 volts. Usando tanto o método direto quanto o método iterativo, calcule a corrente em cada uma das malhas do circuito.

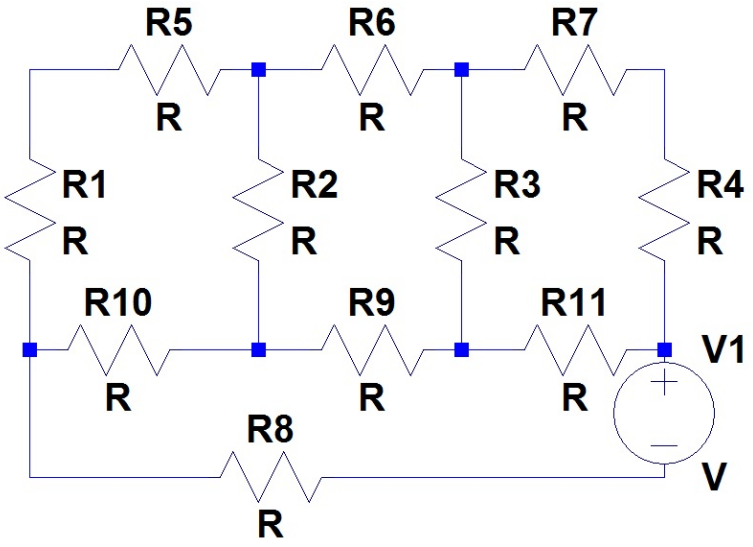


Figura 5.6: Circuito elétrico

Questão 10. Considere uma matriz quadrada $\mathbf{M}$, seja $\mathbf{M}_t = \mathbf{M}^T$, os autovalores de $\mathbf{M}$ e $\mathbf{M}_t$ são os mesmos?

Questão 11. Uma matriz $\mathbf{M}_{4\times4}$ tem seus elementos m_{ij} definidos por

$$m_{ij} = \frac{1}{(3/2)^i - (1/2)^j}$$

com $1 \leq i, j \leq 4$. Quais os autovalores de $\mathbf{M}$?

Questão 12. Considerando a matriz $\mathbf{M}_{4\times4}$ da questão anterior, ela é convergente? Verifique isso numericamente.

Questão 13. Ainda considerando a matriz $\mathbf{M}_{4\times4}$ das questões anteriores, mostre qua norma de Frobenius[5] é maior que $|\lambda_{max}|$.

Questão 14. Em nossas cozinhas, em geral, usamos o gás butano como combustível para aquecer e cozinhar os alimentos. A fórmula química do gás butano é C_4H_{10}. Podemos expressar a queima desse gás ocorre na presença do oxigênio atmosférico (O_2) e resulta em gás carbônico e água (H_2O). Faça o balanço dessa reação química:

$$xC_4H_{10} + yO_2 \rightarrow zCO_2 + wH_2O$$

Questão 15. Considere o sistema abaixo, verifique se podemos usar o procedimento iterativo para encontrar a solução. Encontre sua solução iterativamente.

$$\begin{bmatrix} 3,6 & -1,5 & 1 & 0,5 & 0,2 & 0,1 \\ 1,2 & 3,4 & -1 & 0,7 & -0,3 & -0,1 \\ 0,8 & 1,2 & -4 & 0,7 & 0,1 & 0,4 \\ -0,5 & 0,7 & 1,4 & 2,5 & 0,2 & 0,2 \\ -0,2 & 0,7 & 1,1 & 0,5 & -3,2 & 0,1 \\ 0,5 & -0,2 & 0,6 & 1,2 & 0,6 & -3,9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 3,5 \\ 2,1 \\ 3,3 \\ 1,9 \\ -2,3 \\ 1,7 \end{bmatrix}$$

Questão 16. O sistema a seguir não obedece o critério de Sassenfeld, mas podemos calcular a sua solução iterativamente. Quantas iterações são necessárias para que o sistema consiga convergir com um erro $|\varepsilon|$ máximo menor que 10^{-3}? Esse erro é calculado como $\varepsilon_k = min\{x_i^k - x_i^{k-1}\}$.

$$\begin{bmatrix} 2.50 & 2.00 & 1.00 & 1.00 & 1.00 & 1.00 & 0.50 \\ 1.00 & 3.00 & 1.00 & 1.00 & 1.00 & 0.50 & 0.25 \\ 1.00 & -1.20 & 2.50 & 1.00 & -2.00 & 1.00 & 0.10 \\ 1.00 & 0.50 & 1.50 & 2.20 & -1.00 & 0.50 & 0.20 \\ 0.50 & 0.50 & -1.20 & -1.00 & 2.10 & 1.00 & -0.20 \\ 0.50 & 1.50 & -1.00 & 1.00 & -1.00 & 2.00 & -0.10 \\ 0.25 & 0.50 & 1.50 & -1.00 & 1.00 & -1.00 & 2.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 24.250 \\ 21.875 \\ 6.050 \\ 13.200 \\ 10.500 \\ 11.550 \\ -0.250 \end{bmatrix}$$

Questão 17. O sistema abaixo está longe de ser do tipo ideal para resolvermos iterativamente. Verifique que o número total de iterações para que o erro absoluto ε seja menor que 5×10^{-3} é quase

[5] Ferdinand Georg Frobenius (26 de outubro de 1849 - 3 de agosto de 1917) foi um matemático alemão, mais conhecido por suas contribuições para a teoria das funções elípticas, as equações diferenciais, a teoria dos números e a teoria dos grupos. Ele também foi o primeiro a introduzir a noção de aproximações racionais de funções e apresentou a primeira prova completa para o teorema de Cayley-Hamilton.

90. O que você pode fazer para que esse sistema consiga convergir em um número muito menor de iterações? Dica: tente tornar a diagonal principal mais dominante. Você pode efetuar operações elementares antes de começar a resolver iterativamente.

$$\begin{bmatrix} 2.40 & 2.20 & 1.30 & 1.20 & 1.20 & 1.10 & 0.50 & 0.30 \\ -1.00 & 3.30 & -1.00 & 1.00 & 1.00 & 0.50 & 0.25 & 0.20 \\ -1.00 & -1.00 & 2.50 & -1.40 & -1.70 & -1.00 & 0.10 & 0.20 \\ 1.00 & 0.50 & 2.00 & 2.60 & 1.00 & 0.50 & 0.20 & 0.10 \\ 0.50 & 0.50 & -1.20 & -1.00 & 2.10 & 1.00 & -0.20 & 0.10 \\ 0.50 & 1.50 & -1.00 & 1.00 & -1.00 & 2.00 & -0.10 & -0.20 \\ 0.25 & -0.50 & 1.50 & -1.30 & 1.00 & -1.00 & 2.10 & -0.50 \\ -0.30 & 0.60 & 0.50 & 1.50 & -1.20 & 1.00 & -1.30 & -2.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 27.250 \\ 14.075 \\ -16.05 \\ 26.100 \\ 10.300 \\ 11.950 \\ -2.500 \\ 13.050 \end{bmatrix}$$

Questão 18. Usando o comando "rand" do Scilab, crie um sistema linear 100×100. Resolva esse sistema usando os recursos do Scilab e, também, escrevendo o seu próprio *script*. Compare os resultados. Houve alguma diferença numérica?

Questão 19. Para termos "certeza" que o método iterativo estudado levará à solução correta com um número finito de passos, para qualquer valor inicial de $\mathbf{x}^0$, o maior autovalor λ da matriz de iteração deve ter módulo menor que a unidade. Podemos calcular facilmente os autovalores de uma matriz usando o comando "spec". Verifique a convergência do sistema abaixo (calcule a matriz de iteração), em seguida, resolva-o iterativamente.

$$\begin{cases} 4x_1 + 3x_2 + 2x_3 = 12 \\ 2x_1 + 4x_2 + 3x_3 = 10 \\ x_1 + 2x_2 + 2x_3 = 8 \end{cases}$$

Questão 20. O circuito abaixo é "infinito" (ver Figura 5.7), isto é, as malhas formadas pelos resistores $R_k - R_{k+1}$ continuam indefinidamente. Estime o valor das correntes i_1, i_2, i_3 das primeiras malhas. A fonte de tensão CC fornece 10 volts, todos os resistores são iguais a $1k\Omega$. Verifique que o sistema de equações lineares formado é tridiagonal. Um sistema desse tipo pode ser resolvido de forma eficiente pelo método de Thomas[6]. Para escrever as equações que descrevem este circuito, se necessário, consulte um livro de circuitos elétricos.

Questão 21. Resolva o sistema abaixo tendo como chute inicial $\mathbf{x}_0 = [0,5; 2, 0]^T$. Quantas iterações são necessárias? Você pode adaptar o *script* da seção 5.7.

$$\begin{cases} \text{sen}(\pi x) + \ln(y) - 2 = 0 \\ 2x^3 + \sqrt{2y} - 3 = 0 \end{cases}$$

Questão 22. Encontre uma solução para o sistema não linear abaixo. Use como um chute inicial $\mathbf{x}_0 = [0,5; 1, 0]^T$.

$$\begin{cases} x^3 + 2xy^2 = 9 \\ \sqrt{x} - 2^y = 5 \end{cases}$$

Questão 23. Encontre uma solução para o sistema não linear abaixo. Use como um chute inicial $\mathbf{x}_0 = [0,5; 0, 5]^T$.

$$\begin{cases} x^2 + y^3 - 9 = 0 \\ \sqrt{x} - y + 2 = 0 \end{cases}$$

[6]Llewellyn Hilleth Thomas (Londres, 21 de outubro de 1903 - 20 de abril de 1992) foi um físico e matemático aplicado britânico. É mais conhecido por suas contribuições à física atômica. Ele criou um método eficiente de eliminação de Gauss para resolver sistemas lineares tridiagonais.

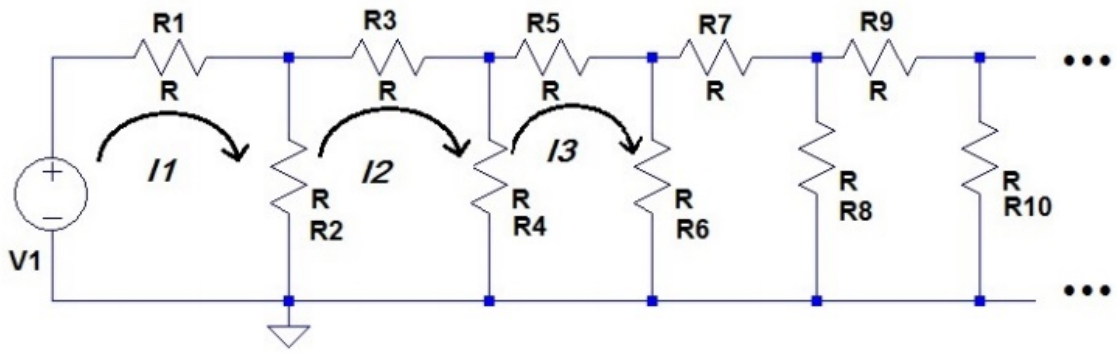


Figura 5.7: Circuito “infinito”.

Questão 24. Analise o sistema não linear abaixo. Tente um chute inicial e encontre a sua solução.

$$\begin{cases} x\sqrt{y}+e^{-xy}-2,2 &= 0 \\ \cos(x)+\operatorname{sen}(y)-0,4 &= 0 \end{cases}$$

Questão 25. Considere o sistema abaixo:

$$\begin{cases} \cos(\pi x_1/2)+\ln(x_2x_3)+x_3^2-11 &= 0 \\ x_1x_2x_3+\sqrt{x_1x_3}+\cos(\pi x_3/2)-8 &= 0 \\ x_1x_3+\exp(-x_1x_2)-\cos(\pi x_2x_3/6)-4 &= 0 \end{cases}$$

Mostre que com o valor inicial $\mathbf{x}_0=[0,5;0,5;1,5]^T$ alcançamos a solução, mas com $\mathbf{x}_0=[0,5;0,25;0,5]^T$ não. Por que isso acontece?

Questão 26. Mostre que para o sistema

$$\begin{cases} x^3+xy &= 3 \\ \sqrt{x}-y^2 &= 5 \end{cases}$$

Podemos usar as equações iterativas

$$x_{k+1}=\sqrt[3]{3-x_ky_k}$$
$$y_{k+1}=\sqrt{5-\sqrt{x_{k+1}}}$$

para encontrar a solução do sistema se o chute inicial for suficientemente próximo. Por exemplo, para $\mathbf{x}_0=[0,5;0,5]^T$ podemos obter a solução correta após algumas iterações. Verifique[7].

[7]Na postagem “Solução de sistemas de equações não lineares: método iterativo”, mostramos mais alguns detalhes dessa técnica, *link*: `http://alfanumericus.blogspot.com/2018/12/solucao-de-sistemas-de-equacoes-nao.html`

6. Regressão

Se o conhecimento pode criar problemas, não é através da ignorância que podemos solucioná-los.
Isaac Asimov (1919/1920 - 1992), bioquímico e escritor de ficção científica americano.

QUANDO temos um conjunto de pontos e queremos traçar uma reta ou uma curva $f(x)$ que, no intervalo (a,b), passe por esses pontos, estamos tratando de um problema de regressão linear (reta) ou não linear (curva). Às vezes, até conhecemos a função que gerou os pontos, mas queremos trabalhar com uma função mais simples que se aproxime da função original com uma certa margem de erro. Esse é também um problema de regressão.

6.1 Regressão linear

Na regressão linear, queremos encontrar a equação de uma reta $f(x) = ax + b$ que passe por um conjunto de pontos (x_i, y_i) dados. Em geral, esses pontos não são colineares e reta deverá passar *entre* os pontos e pelos pontos dados. A Figura 6.1 ilustra essa situação. Para calcular a função da reta, temos que minimizar alguma função erro. Podemos definir o erro como sendo a diferença entre y_i e $f(x_i)$: $e_i = y_i - ax_i + b$. Então, podemos definir a função de erro como sendo

$$E(a,b) = \sum_i e_i^2 \tag{6.1}$$

e agora devemos encontrar os valores de a e b que minimizem esse erro. O valor de e_i^2 é:

$$\begin{aligned} e_i^2 &= (y_i - ax_i - b)^2 \\ &= y_i^2 + a^2x_i^2 + b^2 - 2ay_ix_i + 2abx_i - 2by_i \end{aligned}$$

Para encontramos os valores ótimos de a e b devemos derivar $E(a,b)$:

$$\frac{\partial e_i^2}{\partial a} = ax_i^2 - y_i x_i + bx_i$$
$$\frac{\partial e_i^2}{\partial b} = b + ax_i - y_i$$

igualando a zero:

$$ax_i^2 + bx_i = y_i x_i$$
$$ax_i + b = y_i$$

aplicando o somatório:

$$\begin{bmatrix} \sum_i 1 & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} \sum_i y_i \\ \sum_i y_i x_i \end{bmatrix} \tag{6.2}$$

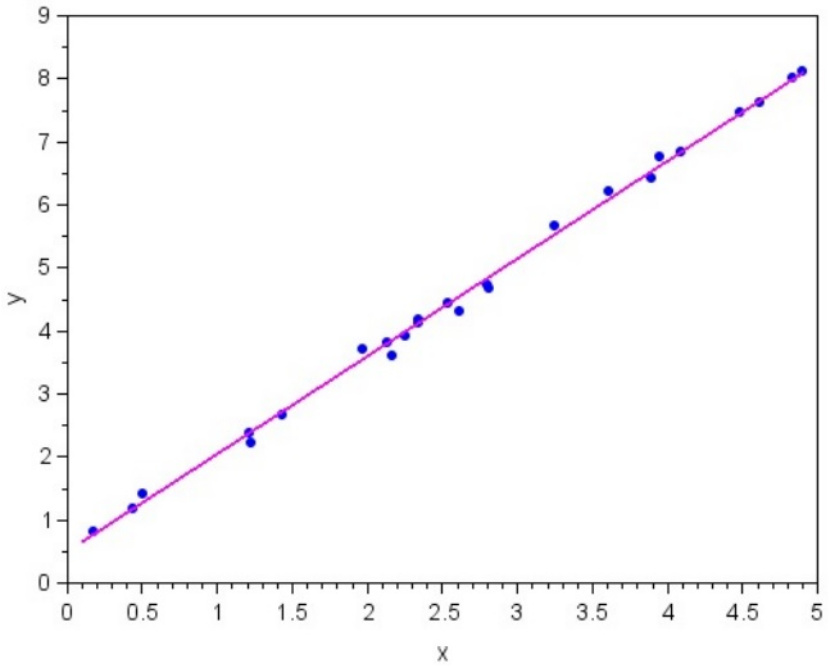


Figura 6.1: Pontos dados e uma reta que passa por eles

Logo, resolvendo o sistema indicado em (6.14) (*sistema normal*) obteremos a equação de uma reta que passa entre os pontos dados como menor erro quadrático. Como uma medida da qualidade do ajuste da reta aos pontos dados, podemos usar o seguinte índice de correlação:

$$r = \frac{S_y - S_r}{S_y} \tag{6.3}$$

onde

$$S_r = \sum_{i=1}^{N} (y_i - ax_i - b)^2 \quad \text{e} \quad S_y = \sum_{i=1}^{N} (y_i - \bar{y})^2$$

e

$$\bar{y} = \frac{1}{N}\sum_{i}^{N} y_i,$$

quanto mais próximo de "1" for o valor de r, melhor o ajuste da reta calculada aos pontos dados. Vejamos um exemplo usando o Scilab:

```
// Pontos (xi, yi) disponíveis:
x = [0.1; 0.15; 0.2; 0.6; 0.9; 1.1; 1.4; 1.5; 1.7; 2.1; 2.2; ...
2.3; 2.4; 2.7; 3.1; 3.2];
y = [1.24; 1.304; 1.566; 2.548; 2.86; 3.443; 3.875; 4.133; ...
4.374; 5.187; 5.475;  5.852;  5.721;  6.345;  7.208; 7.312];

// Cálculos:
M = [max(size(x)), sum(x);
     sum(x), sum(x.*x)];
v = [sum(y); sum(x.*y)];
ab = inv(M)*v;

// reta:
xr = 0.05:0.01:3.4;
yr = ab(1) + ab(2)*xr;
close; plot(x,y,'o',xr,yr,'m');
xlabel('xi'); ylabel('yi');
legend('pontos','f(x) - reta');

Sr = y - ab(1) - ab(2)*x; Sr = sum(Sr.*Sr);
yb = mean(y); Sy = y - yb; Sy = sum(Sy.*Sy);
r = sqrt((Sy - Sr)/Sy);  disp(r);
```

No código acima, o comando "max(size(x))" calcula o número de pontos existentes no vetor "x", já o comando "sum(x.*y)" equivale ao cálculo $\sum_i y_i x_i$. O valor de r calculado foi $r = 0,9983988$, o que mostra uma excelente concordância entre os pontos e a reta para este exemplo. A Figura 6.2 mostra o resultado do comando "plot".

Também podemos usar o comando "lsq" do Scilab para calcular a regressão linear ou regressão polinomial. Um exemplo do uso desse comando:

```
//Construindo os dados
x=(1:10)';  // vetor x coluna
y=3*x+2;  // reta

xr = x + 0.1*rand(x,'n'); // erro no eixo x
yr = y + 0.5*rand(y,'n'); //  erro no eixo y

plot(xr,yr,'m*');   // pontos no plano (X,Y)

//// calculando a melhor reta:
```

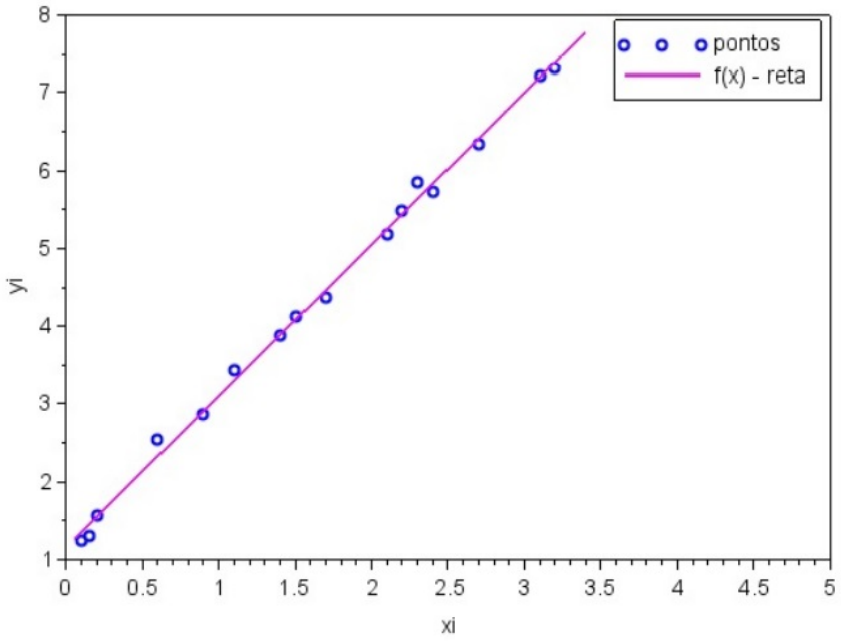


Figura 6.2: Exemplo Scilab de regressão linear

```
A = [xr, ones(xr)];
X = lsq(A,yr); // coeficientes ótimos da reta

xn = 1:0.1:10;  yn = X(1)*xn + X(2);
plot(xn,yn,'b');
```

6.2 Regressão polinomial

Em muitos casos, os pontos disponíveis não formam uma função linear, mas se aproximam mais de uma função quadrática ou outra curva não linear. Nesse caso, é provável que obtenhamos um melhor resultado calculando uma curva parabólica ou cúbica que passe pelos pontos experimentais. Então, nosso problema agora é determinar os parâmetros a, b e c da função $f(x) = ax^2 + bx + c$ que minimize o erro quadrático e_i^2. Assim, podemos escrever:

$$e_i = y_i - ax_i^2 - bx_i - c$$

Derivando o erro quadrático:

$$\frac{\partial e_i^2}{\partial a} = e_i x_i^2 = y_i x_i^2 - ax_i^4 - bx_i^3 - cx_i^2 \tag{6.4}$$

$$\frac{\partial e_i^2}{\partial b} = e_i x_i = y_i x_i - ax_i^3 - bx_i^2 - cx_i \tag{6.5}$$

$$\frac{\partial e_i^2}{\partial a} = e_i = y_i - ax_i^2 - bx_i - c \tag{6.6}$$

Aplicando o somatório como antes, obtemos o sistema normal de equações:

$$\begin{bmatrix} \sum_i 1 & \sum_i x_i & \sum_i x_i^2 \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i^3 \\ \sum_i x_i^2 & \sum_i x_i^3 & \sum_i x_i^4 \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix} = \begin{bmatrix} \sum_i y_i \\ \sum_i y_i x_i \\ \sum_i y_i x_i^2 \end{bmatrix} \tag{6.7}$$

Um exemplo usando o Scilab:

```
// Pontos (xi, yi) disponíveis:
x = [0.1; 0.15; 0.2; 0.6; 0.9; 1.1; 1.4; 1.5; 1.7; 2.1; 2.2; ...
2.3; 2.4; 2.7; 3.1; 3.2; 3.3; 3.4];
y = [1.048; 1.391; 1.32; 1.794; 2.155; 2.471; 2.705; 2.561; ...
2.605; 3.028; 3.01; 3.116; 3.163; 3.27; 3.35; 3.418; 3.50; 3.53]

// Cálculos:
M = [max(size(x)), sum(x), sum(x.*x)
     sum(x), sum(x.*x), sum(x.*x.*x)
     sum(x.*x), sum(x.*x.*x), sum(x.*x.*x.*x)]
v = [sum(y); sum(x.*y); sum(x.*y.*x)];
abc = inv(M)*v;

// parábola:
xr = 0.05:0.01:3.6;
yr = abc(1) + abc(2)*xr + abc(3)*xr.*xr;
plot(x,y,'o',xr,yr,'m');
xlabel('xi'); ylabel('yi');
legend('pontos','f(x) - 2o. grau');
```

Usando a mesma linha de raciocínio, podemos obter uma função cúbica $f(x) = ax^3 + bx^2 + cx + d$ que passa por um conjunto de pontos dados, resolvendo o seguinte sistema de equações:

$$\begin{bmatrix} \sum_i 1 & \sum_i x_i & \sum_i x_i^2 & \sum_i x_i^3 \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i^3 & \sum_i x_i^4 \\ \sum_i x_i^2 & \sum_i x_i^3 & \sum_i x_i^4 & \sum_i x_i^5 \\ \sum_i x_i^3 & \sum_i x_i^4 & \sum_i x_i^5 & \sum_i x_i^6 \end{bmatrix} \begin{bmatrix} d \\ c \\ b \\ a \end{bmatrix} = \begin{bmatrix} \sum_i y_i \\ \sum_i y_i x_i \\ \sum_i y_i x_i^2 \\ \sum_i y_i x_i^3 \end{bmatrix} \tag{6.8}$$

Vejamos, agora, mais um exemplo usando o Scilab. Iremos gerar pontos "aleatórios" entre 0 e 3 (aproximadamente) com algum "ruído":

```
rand('seed',16);  // inicializando o gerador de números ‘‘aleatórios’’
x = 3.5*rand(1,35,'u') ; x = gsort(x);
x = x - min(x) + 0.05; x = x(35:-1:1);  // vetor x ordenado
y = sqrt(sin(x)) + 0.02*rand(x,'n');  // vetor y
```

O código para calcular a reta, a parábola e a função cúbica que melhor se ajustam aos pontos está listado acima. A Figura 6.4 mostra a comparação entre as curvas obtidas.

```
// Cálculos:
M = [max(size(x)), sum(x);
```

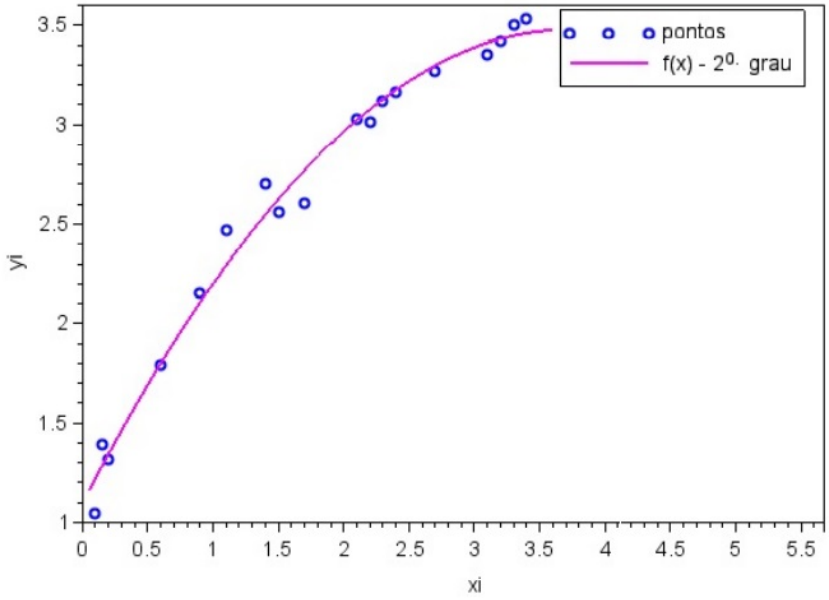


Figura 6.3: Pontos dados e uma parábola que passa por eles

```
     sum(x), sum(x.*x)];
v = [sum(y); sum(x.*y)];
ab = inv(M)*v;

// reta:
xr = x(1):0.01:x(35);
yr = ab(1) + ab(2)*xr;
close; plot(x,y,'o',xr,yr,'m');
xlabel('xi'); ylabel('yi');
yri = ab(1) + ab(2)*x;
e = y - yri; e2r = sum(e.*e);

// Cálculos:
M = [max(size(x)), sum(x), sum(x.*x)
     sum(x), sum(x.*x), sum(x.*x.*x)
     sum(x.*x), sum(x.*x.*x), sum(x.*x.*x.*x)]
v = [sum(y); sum(x.*y); sum(x.*y.*x)];
abc = inv(M)*v;

// parábola:
yr = abc(1) + abc(2)*xr + abc(3)*xr.*xr;
plot(xr,yr,'m');
yri = abc(1) + abc(2)*x + abc(3)*x.*x;
e = y - yri;  e2p = sum(e.*e);
```

```
// Cálculos:
M = [max(size(x)), sum(x), sum(x.*x), sum(x.*x.*x)
     sum(x), sum(x.*x), sum(x.*x.*x), sum(x.*x.*x.*x)
     sum(x.*x), sum(x.*x.*x), sum(x.*x.*x.*x), sum(x.*x.*x.*x.*x)
     sum(x.*x.*x), sum(x.*x.*x.*x), sum(x.*x.*x.*x.*x), sum(x.*x.*x.*x.*x.*x)]
v = [sum(y); sum(x.*y); sum(x.*y.*x); ; sum(x.*y.*x.*x)];
f = inv(M)*v;

// cúbica:
yr = f(1) + f(2)*xr + f(3)*xr.*xr + f(4)*xr.*xr.*xr; plot(xr,yr,'k');
yri = f(1) + f(2)*x + f(3)*x.*x + f(4)*x.*x.*x;
e = y - yri;  e2c = sum(e.*e);

legend('pontos','f(x) - reta','f(x) - 2o. grau','f(x) - 3o. grau');
title(['Soma dos erros (r, p, c): ',string([e2r, e2p, e2c])]);
```

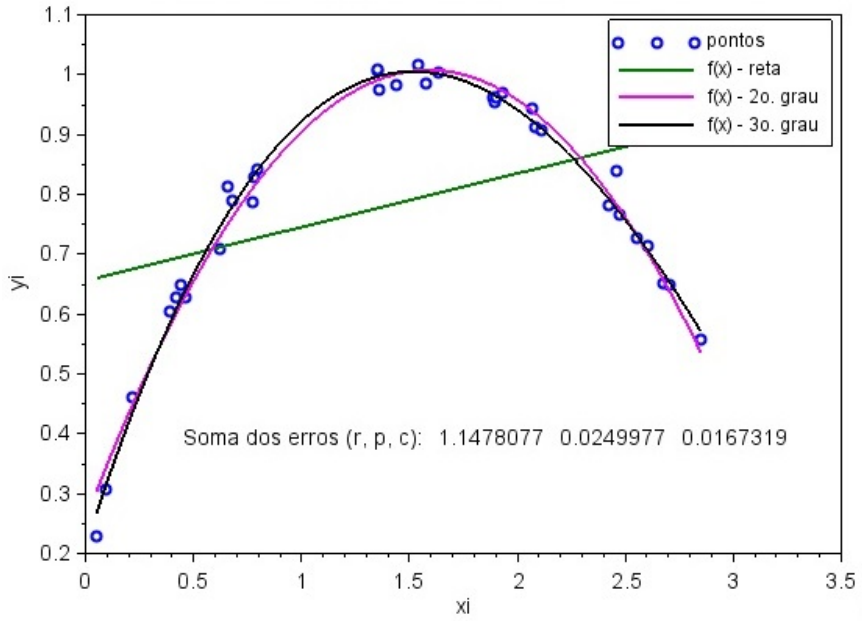


Figura 6.4: Comparação entre regressão linear, parabólica e cúbica. A aproximação por uma reta não é adequada. A curva cúbica apresenta um erro cerca de 33% menor que a parabólica.

Para concluirmos esta seção, veremos um exemplo do uso do comando "lsq" para fazer regressão polinomial:

```
//Construindo os dados
x=(1:10)';  // vetor x coluna
y=-0.35*x.*x+3*x+2;  // reta
xr = x + 0.1*rand(x,'n'); // erro no eixo x
yr = y + 0.5*rand(y,'n'); //  erro no eixo y
plot(xr,yr,'m*');  // pontos
```

```
//// calculando a melhor reta:
A = [ones(xr),xr,xr.*xr];
X = lsq(A,yr); // coeficientes ótimos da parábola
xn = 1:0.1:10;
yn = X(3)*xn.*xn+X(2)*xn + X(1);
plot(xn,yn,'b');  // parábola que passa pelos pontos
```

6.3 Regressão por outras funções

Algumas curvas podem ser mais adequadas que as polinomiais para passar pelos pontos dados. Existem muitas funções possíveis que podem ser testadas para isso, por exemplo (ver Figura 6.5):

$$f(x) = a_0 + a_1 e^{kx} \quad \text{(função exponencial)} \tag{6.9}$$

$$f(x) = \frac{b_0 + b_1 e^{k_1 x}}{a_0 + a_1 e^{k_2 x}} \quad \text{(função logística)} \tag{6.10}$$

$$f(x) = a_0 + a_1 \ln(x) \quad \text{(função logarítmica)} \tag{6.11}$$

$$f(x) = \frac{b_0 + b_1 x + b_2 x^2 + \cdots}{a_0 + a_1 x + a_2 x^2 + \cdots} \quad \text{(função racional)} \tag{6.12}$$

$$f(x) = a_0 + a_1 \cos(w_0 t) + a_2 \cos(w_1 t) + \ldots \quad \text{(função senoidal)} \tag{6.13}$$

Saber qual dos tipos de funções acima se encaixa melhor nos dados disponíveis requer alguma experiência. Trataremos do uso de funções senoidais para a próxima seção. Vejamos um exemplo simples de como usar os mínimos quadrados para encontrar uma função que melhor se ajusta aos dados.

Considere os pontos indicados na Figura 6.6. Qual a melhor função para aproximar esses pontos? Talvez, não seja evidente, mas uma função racional é a melhor opção. "Escolhida" uma função do tipo racional, resta saber quantos coeficientes ela terá. Levando em conta que os pontos apresentam apenas um ponto de máximo e depois caem lentamente, provavelmente uma função racional do tipo

$$f(x) = \frac{P(x)}{Q(x)} = \frac{b_0 + b_1 x + b_2 x^2}{1 + a_1 x + a_2 x^2}$$

seja suficiente para o ajuste dos pontos (note que fizemos $a_0 = 1$). Logo, podemos escrever:

$$\frac{b_0 + b_1 x_i + b_2 x_i^2}{1 + a_1 x_i + a_2 x_i^2} = y_i$$

$$b_0 + b_1 x_i + b_2 x_i^2 - a_1 x_i y_i - a_2 x_i^2 y_i = y_i$$

Assim, podemos formar um sistema de equações na forma:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & -x_0 y_0 & -x_0^2 y_0 \\ 1 & x_1 & x_1^2 & -x_1 y_1 & -x_1^2 y_1 \\ 1 & x_2 & x_2^2 & -x_2 y_2 & -x_2^2 y_2 \\ 1 & x_3 & x_3^2 & -x_3 y_3 & -x_3^2 y_3 \\ 1 & x_4 & x_4^2 & -x_4 y_4 & -x_4^2 y_4 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{6.14}$$

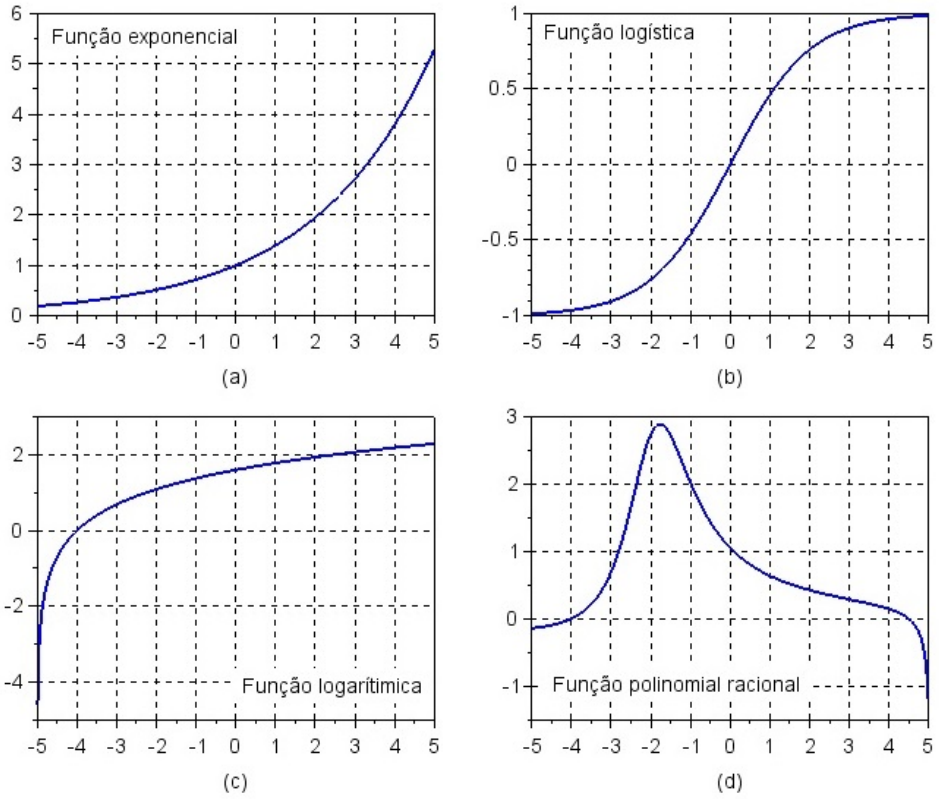


Figura 6.5: Exemplos de funções não polinomiais, existem diversas outras.

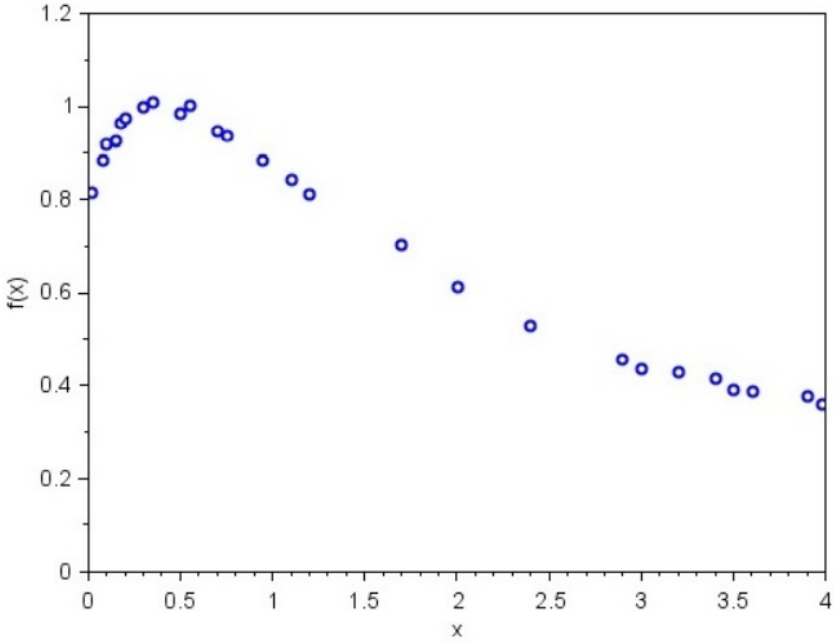


Figura 6.6: Pontos de uma função não linear

para o cálculo dos coeficientes da função racional $f(x)$. Essa estratégia, entretanto, é sensível ao "ruído" nos dados (dados, por exemplo, oriundos de medições experimentais). Ainda fica a questão de quais pontos escolher, já que serão necessários apenas 5 dos 26 pontos que temos. Nesse caso, podemos escolher 5 pontos que estejam mais ou menos igualmente espaçados, incluindo os extremos. No código Scilab a seguir, exemplificamos essa técnica. Apresentamos o resultado obtido na Figura 6.7.

```
clc; close; // pontos ‘‘experimentais’’:
xr = [0.02, 0.08, 0.1, 0.15, 0.18, 0.2, 0.3, 0.35, 0.5, 0.55, 0.7, ...
0.75,0.95, 1.1, 1.2, 1.7, 2, 2.4, 2.9, 3, 3.2, 3.4, 3.5, 3.6, 3.9, 3.98];
yr = (0.80 + 2*xr)./(1 + xr + 1.2*xr.*xr); // função ‘‘desconhecida’’
rand('seed',15);
yr = yr + 0.01*rand(xr,'n');  // adição de ruído
plot(xr,yr,'o');

M = zeros(5,5);
vy = zeros(5,1);
xk = xr(2:5:$); // pontos espaçados
yk = yr(2:5:$);
for k=1:5
    x = xk(k); x2 = x*x; y = yk(k);
    M(k,:) = [1, x, x2, -x*y, -x2*y];
    vy(k) = y;
end
c = inv(M)*vy;  // cálculo dos coeficientes
disp(c);

/// Gráfico da nova função obtida:
x = 0:0.01:4;
y = (c(1) + c(2)*x + c(3)*x.*x)./(1 + c(4)*x + c(5)*x.*x);
plot(x,y,'k'); legend('pontos','função com racional');
xlabel('x'); ylabel('f(x)');
```

6.4 Regressão senoidal

Quando os sinais de interesse são periódicos, eles podem ser aproximados por funções senoidais. Iremos usar o comando "fft" para auxiliar nos cálculos. Essa abordagem é mais geral do que simplesmente encontrar uma função senoidal $f(t) = A_0 + A_1\cos(\omega t + \phi)$ que se aproxime dos pontos dados ou de uma função periódica não conhecida. Com o comando "fft", podemos "descobrir" as componentes de diferentes frequências que formam o sinal periódico. Antes, porém, vamos verificar dois fatos:

1. a somas de senos e cossenos cujas frequências sejam múltiplos da mesma frequência fundamental geram um sinal periódico;
2. (quase todos) sinais periódicos não senoidais podem ser representados por uma soma de senos e cossenos mais um valor médio (Série de Fourier[1]);

[1] Jean-Baptiste Joseph Fourier (Auxerre, 21 de março de 1768 - Paris, 16 de maio de 1830) foi um matemático e

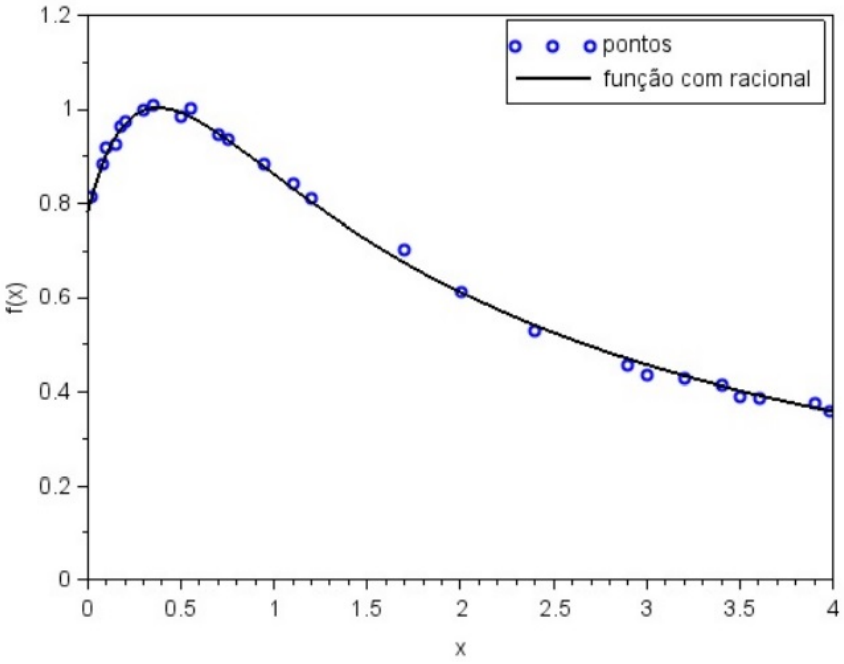


Figura 6.7: Regressão usando uma razão de polinômios.

através de dois exemplos bem didáticos. Seja a soma de duas senoides:

$$f(t) = 3\sin(2\pi t) + 2\cos(4\pi t) + \cos(6\pi t),$$

essa soma gera um sinal periódico como mostra a Figura 6.8(a). Já um sinal periódico do tipo onda quadrada $q(t)$ pode ser decomposto em uma soma de cossenos (Figura 6.8(b)):

$$q(t) = \cos(2\pi t) - (1/3)\cos(6\pi t) + (1/5)\cos(10\pi t) - (1/7)\cos(14\pi t) + (1/9)\cos(18\pi t) - ...$$

A Série Trigonométrica de Fourier tem a forma

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left[a_n \cdot \cos\left(\frac{2n\pi t}{T}\right) + b_n \cdot \text{sen}\left(\frac{2n\pi t}{T}\right)\right] \tag{6.15}$$

onde T é o período fundamental de $f(t)$ e

$$a_0 = \frac{1}{T}\int_c^{c+T} f(t)dt, \tag{6.16}$$

$$a_n = \frac{2}{T}\int_c^{c+T} f(t)\cos\left(\frac{2n\pi t}{T}\right)dt,\ n \geq 1 \tag{6.17}$$

$$b_n = \frac{2}{T}\int_c^{c+T} f(t)\,\text{sen}\left(\frac{2n\pi t}{T}\right)dt,\ n \geq 1 \tag{6.18}$$

c é uma constante qualquer que usamos para simplificar o cálculo das integrais. Aqui, cabe uma observação: se a função $f(t)$ for uma função par, então os termos b_n são nulos; se a função for ímpar, então a_n são zeros.

físico francês, celebrado por iniciar a investigação sobre a decomposição de funções periódicas em séries trigonométricas convergentes, chamadas séries de Fourier, e a sua aplicação aos problemas da condução do calor.

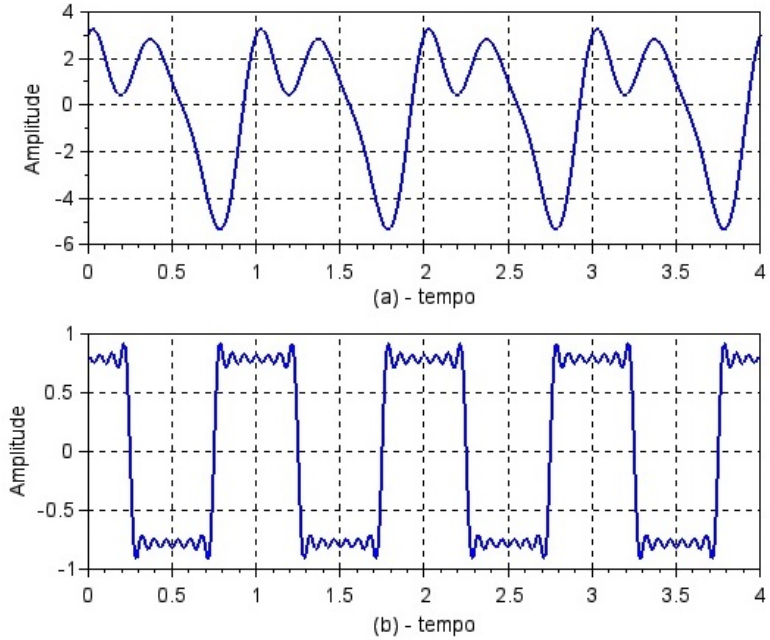


Figura 6.8: Sinais periódicos: (a) soma de senos gera um sinal periódico; (b) onda quadrada como a soma de senoides (Série de Fourier).

Exemplo. Vejamos um exemplo completo do uso da função "fft" para o cálculo dos coeficientes a_n e b_n da série de Fourier tendo como base o gráfico mostrado pela Figura 6.9. A ideia básica é calcularmos o espectro do sinal periódico com o comando "fft" e, em seguida, "montar" um sinal com as componentes de maior amplitude, pois valores de baixa amplitude no espectro podem ser apenas ruído. Com essa estratégia, se o sinal tiver vários períodos, acabamos por filtrar algum ruído existente na função. Podemos ver que o período do sinal é de 1 Hz e que temos 10 períodos completos. Na Figura 6.10, observamos o espectro de amplitude do sinal tanto em escala linear quanto em escala logarítmica. Com base no espectro calculado, a função obtida é:

$$\begin{aligned} f(t) \cong\ & 1.0926367\cos(2\pi t) + 0.2942306\,\mathrm{sen}(4\pi t) - 0.3166524\cos(6\pi t) \\ & - 0.0450103\,\mathrm{sen}(8\pi t) + 0.1255820\cos(10\pi t) + 0.0230651\,\mathrm{sen}(12\pi t) \\ & + 0.0779507\cos(8\pi t) - 0.0593305\cos(14\pi t) + 0.0373689\cos(18\pi t) + \cdots \end{aligned}$$

que podemos ver na Figura 6.11. Mesmo que o sinal original esteja contaminado com ruído de forma significativa, o resultado obtido será muito semelhante, como podemos ver na Figura 6.12.

O código Scilab completo para este exemplo é:

```
xdel(winsid());  // fechando todas as janelas

t=0:0.01:(10-0.01);  // tempo
p2=2*%pi;  // valor de 2 pi
// Sinal:
s = cos(p2*t) + 0.5*sin(2*p2*t) - 0.3*cos(3*p2*t) +...
 0.1*cos(4*p2*t + 0.5) + 0.05*cos(5*p2*t);
s = sqrt(abs(s)).*sign(s);
```

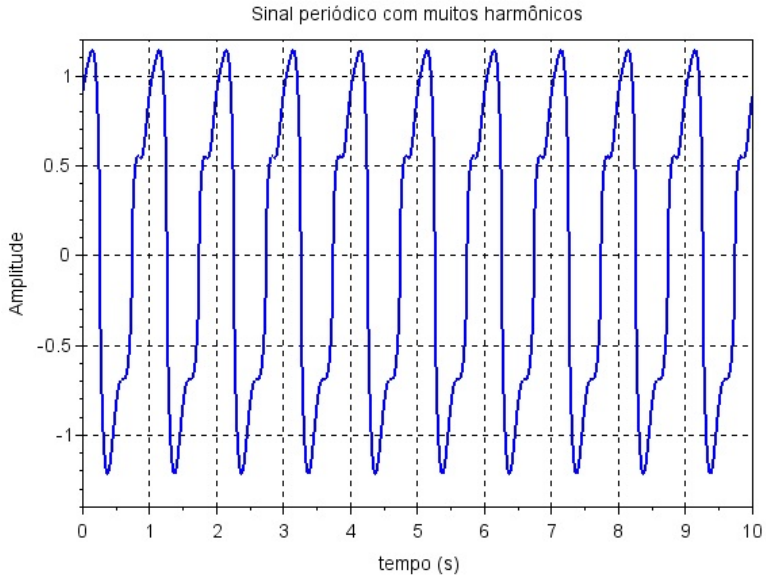


Figura 6.9: Função periódica rica em harmônicos.

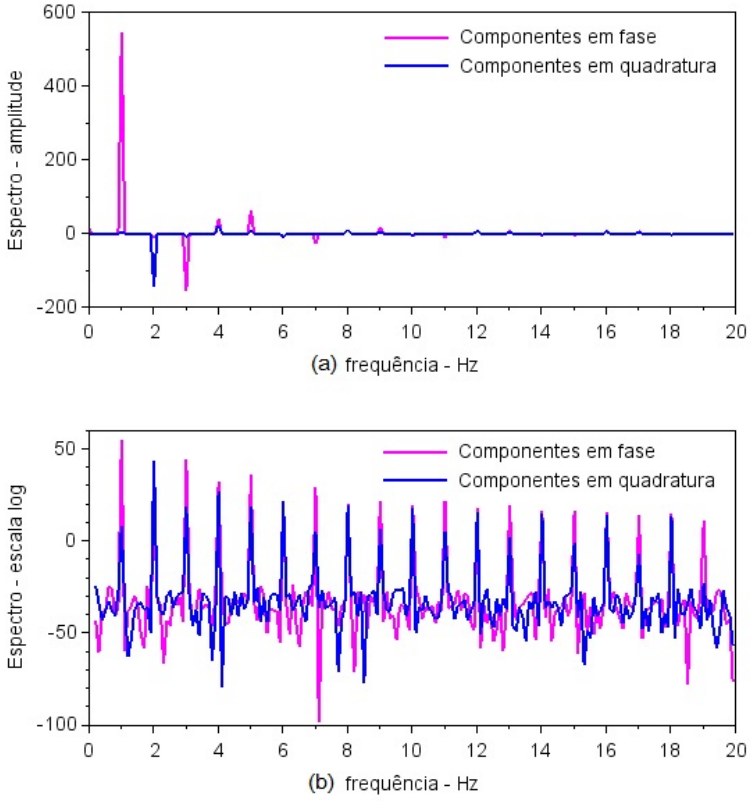


Figura 6.10: Espectro da função periódica: (a) escala linear; (b) escala logarítmica.

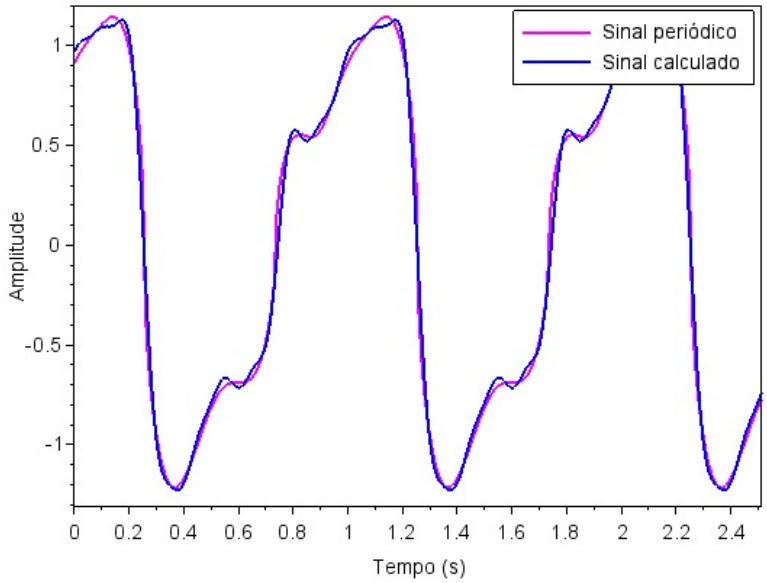


Figura 6.11: Comparando o sinal periódico com o sinal estimado.

```
sp = s;
s = s + 0.001*rand(s,'n'); // um pouco de ruído
// Gráfico:
plot(t,s); xgrid; title('Sinal periódico com muitos harmônicos');
xlabel('tempo (s)'); ylabel('Amplitude');

// Calculando o espectro:
f = 1:max(size(s)); f = f - 1;
f = f/max(f); f = f*100;  // eixo de frequência
sf = fft(s); sfr = real(sf); sfc = imag(sf); // uso da fft
figure; subplot(2,1,1);
plot(f(1:200), sfr(1:200),'m');
plot(f(1:200), sfc(1:200),'b');
legend('Componentes em fase','Componentes em quadratura');
xlabel('frequência - Hz'); ylabel('Espectro - amplitude');
subplot(2,1,2); e=1e-16; // evitando logarítmo do zero
plot(f(3:200), 20*log10(abs(sfr(3:200))+e),'m',...
f(3:200), 20*log10(abs(sfc(3:200))+e),'b');
xlabel('frequência - Hz'); ylabel('Espectro - escala log');
legend('Componentes em fase','Componentes em quadratura');

//Encontrando os picos no espectro
tt = 1:max(size(s));
tt=tt-1; tt=tt*0.01;
p = zeros(1,5);
fp = p;
```

```
ss = 0*tt;
tam2 = round(max(size(sfr))/2);
sfr = sfr(1:tam2); sfr(1) = 0;
sfc = sfc(1:tam2); sfc(1) = 0;
m = mean(s); picoc = 0 picos = 0; // média e conta picos
kf = 500; //fator de escala - 10 períodos e dt = 0,01;
for k=1:6 // os maiores picos
    [a,b] = max(abs(sfr));
    fq = round(f(b));
    if abs(sfr(b))>picoc then picoc = abs(sfr(b)); end;
    if abs(sfr(b))>(picoc/50) then
        ss = ss + sfr(b)*cos(fq*p2*tt)/kf;
        disp([fq,sfr(b)/kf,0]);
    end;
    sfr(b) = 0;
    [a,b] = max(abs(sfc));
    fq = round(f(b));
    if abs(sfc(b))>picos then picos = abs(sfc(b)); end;
    if abs(sfc(b))>(picoc/50) then
        ss = ss - sfc(b)*sin(fq*p2*tt)/kf;
        disp([fq,-sfc(b)/kf,1]);
    end;
    sfc(b) = 0;
end
ss = m + ss;  // harmônicas mais valor médio.
figure; plot(tt,sp,'m',tt,ss,'b');
legend('Sinal periódico','Sinal calculado');
```

6.5 Problemas

Questão 1. Considere os pontos no plano XY: $(1,1)$, $(2,3)$, $(3,4)$. Encontre a melhor reta que se ajuste a esses pontos.

Questão 2. Refaça a questão anterior usando uma função quadrática.

Questão 3. Aproxime a função $\cos(\pi t)$ no intervalo $(-1/2, 1/2)$ por uma função quadrática.

Questão 4. Refaça a questão anterior usando uma função cúbica. Houve uma melhora significativa no ajuste? Justifique.

Questão 5. Aproxime a função $\text{sen}(\pi t)$ no intervalo $(0,1)$ por uma função quadrática.

Questão 6. Refaça a questão anterior usando uma função cúbica. Houve uma melhora significativa no ajuste? Justifique.

Questão 7. Aproxime a função $f(x) = \sqrt{1+x}$, no intervalo $(1,5)$, por uma reta.

Questão 8. Refaça a questão anterior, mas, agora, usando uma função quadrática para aproximação. O ajuste ficou melhor? Explique.

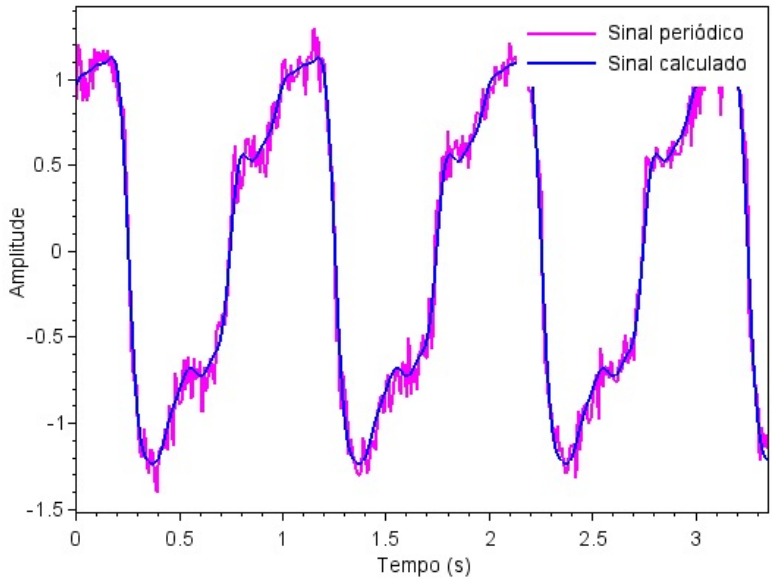


Figura 6.12: Comparando o sinal periódico com ruído e o sinal estimado.

Questão 9. Considere os seguintes pontos mostrados abaixo.
a) encontre uma função quadrática que melhor se ajuste a esses pontos.
b) encontre uma função cúbica que melhor se ajuste a esses pontos.
c) encontre uma função racional na forma

$$f(x) = \frac{b_0 + b_1 x + b_2 x^2}{1 + a_1 x + a_2 x^2}$$

que melhor se ajuste a esses pontos.

x_i	0,50	0,90	1,10	1,40	2,10	2,40
y_i	1,00	1,85	2,20	1,90	1,80	1,40

Questão 10. Refaça as questões anteriores usando o comando "lsq" do Scilab.

Questão 11. Considere a função $f(x) = \sqrt{2x/(1+x)}$ no intervalo aberto $(0, +\infty)$. Encontre uma função racional que se aproxime desta função. Estime a quantidade necessária de coeficientes.

Questão 12. Aproxime a função $f(x) = \ln(1+x)$ no intervalo $(0, 10)$ por uma função cúbica.

Questão 13. Um sinal periódico passa pelos seguintes pontos indicados abaixo. Encontre uma função trigonométrica (senos e cossenos) que melhor se ajuste a esses pontos.

t_i	0,00	0,50	1,00	1,50	2,00	2,50	3,00	3,50	4,00
$f(t_i)$	0,00	1,00	2,00	2,50	0,00	1,00	2,00	2,50	0,00

Questão 14. Quando uma estrela explode em supernova, ela apresenta um "súbito" aumento de brilho que depois cai de forma mais lenta, ao longo de centenas de dias. A Figura 6.13 mostra,

aproximadamente, a curva de luz típica de uma supernova do tipo Ia[2] (na Figura 6.14 vemos um exemplo do surgimento de uma supernova na galáxia espiral Cata-Cento ou M101 na constelação de Ursa Maior em 2011). Encontre uma função que se ajuste aos pontos dados.

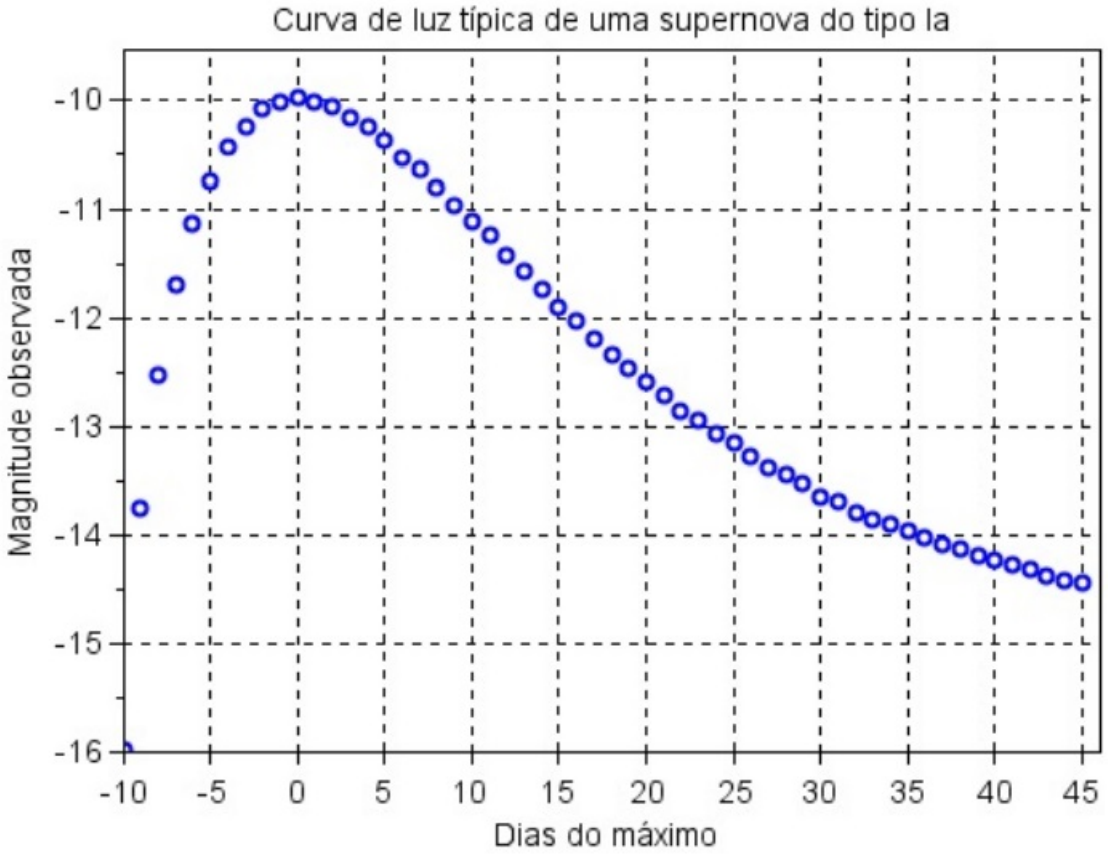


Figura 6.13: Curva de luz típica de estrela que explode com supernova do tipo Ia

Questão 15. A função $f(t) = |15\cos(120\pi t)|$ é um cosseno "retificado". Encontre a série de Fourier $s(t)$ correspondente. Com as primeiras componentes harmônicas calculadas, esboce o gráfico de $s(t)$ e compare com $f(t)$.

Questão 16. A função $f(t) = \text{sen}(2\pi t) + |\text{sen}(2\pi t)|$ descreve uma "retificação de meia onda". Encontre uma função $g(t)$ que seja a soma de senos e cossenos (mais um valor médio) que consiga se aproximar de $f(t)$.

Questão 17. Verifique que a soma de cossenos

$$y(t) = \cos(t) - (1/3)\cos(3t) + (1/5)\cos(5t) - (1/7)\cos(7t) + (1/9)\cos(9t) - \cdots$$

se aproxima de uma função do tipo onda quadrada.

[2] Ver artigo de Shlomo Dado1 e Arnon Dar: Analytical Expressions For Light-Curves Of Ordinary And Superluminous Supernovae Type Ia, junho/2015.

Figura 6.14: Galáxia espiral M101: (a) antes da supernova 2011fe, (b) quando a supernova 2011fe estava próxima do seu brilho máximo.

7. Interpolação

A matemática é o alfabeto com qual Deus escreveu o universo. Galileu Galilei (1564 - 1642), matemático e físico italiano.

O problema de interpolação é muito próximo ao de regressão ou ajuste de curvas. Afinal, se calculamos uma curva que melhor se ajusta a um certo conjunto de pontos em um intervalo conhecido (a,b), podemos usar essa curva calcular qualquer novo ponto dentro do intervalo considerado. Entretanto, se desejamos apenas calcular um ponto dentro do intervalo, podemos usar procedimentos mais simples e menos custosos computacionalmente. São esses procedimentos que estudaremos neste capítulo.

7.1 Interpolação linear

O primeiro caso de interpolação que iremos tratar é o caso linear. Para exemplificar, vamos considerar os pontos (x_0,y_0) e (x_1,y_1) no plano XY. Qual o valor provável de (x_k,y_k) com $y_0 < y_k < y_1$ e $x_0 < x_k < x_1$ (ver Figura 7.1)? Analisando a Figura 7.1, podemos escrever (fazendo semelhança de triângulos):

$$\frac{x_k - x_0}{y_k - y_0} = \frac{x_1 - x_0}{y_1 - y_0}$$
$$y_k - y_0 = \frac{(x_k - x_0)(y_1 - y_0)}{x_1 - x_0}$$

chegamos então a

$$y_k = y_0 + \frac{(x_k - x_0)(y_1 - y_0)}{x_1 - x_0} \tag{7.1}$$

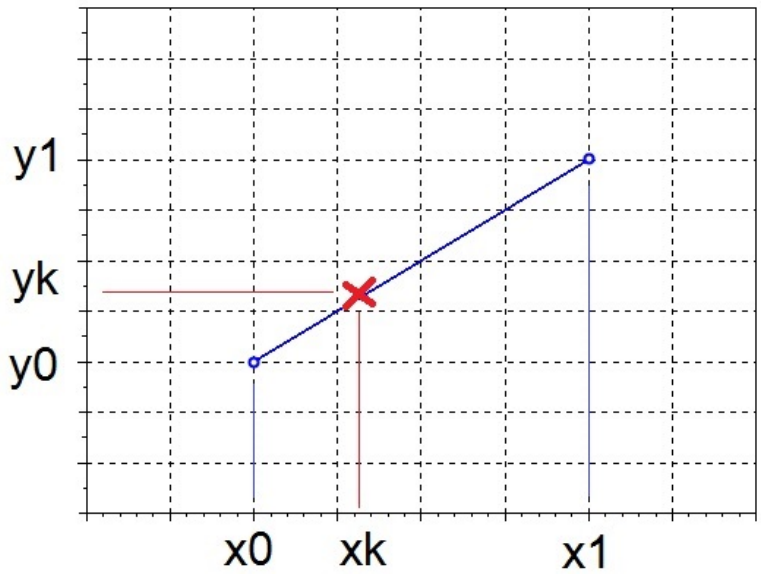


Figura 7.1: Interpolação linear

Vejamos um exemplo. Se temos os pontos $(0,200; 0,8973387)$ e $(0,700; 1,7884354)$, qual o valor y para $x = 0,4$? Cálculos:

$$y_k = y_1 + \frac{(x_k - x_0)(y_1 - y_0)}{x_1 - x_0}$$
$$y_k = 0,8973387 + \frac{(0,400 - 0,200)(1,7884354 - 0,8973387)}{0,700 - 0,200}$$
$$y_k = 1,2537773$$

Sabendo que a função exata é $y(x) = 2\sin(x) + 0,5$, o valor de y_k deveria ser $y_k = 1,2788367$ que corresponde a um erro relativo de aproximadamente $1,96\%$. Com mais pontos, esse erro tende a ser menor.

7.2 Interpolação de Newton

Considerando uma tabela de N pontos (x_k, y_k), $0 \leq k \leq N$, o polinômio interpolador de Newton de grau N é da forma:

$$P_N(x) = y_0 + \Delta_1^0(x - x_0) + \Delta_2^0(x - x_0)(x - x_1) + \Delta_3^0(x - x_0)(x - x_1)(x - x_2) + \cdots , \tag{7.2}$$

Ou de forma mais compacta:

$$P_N(x) = f(x_0) + \sum_{n=1}^{N} \Delta_n^0 \prod_{j=0}^{n-1}(x - x_j) , \tag{7.3}$$

onde os "deltas" são calculados por:

$$\Delta_1^n = \frac{f(x_{n+1}) - f(y_n)}{x_{n+1} - x_n}\ , 0 \leq n \leq N-1$$
$$\Delta_2^n = \frac{\Delta_1^{n+1} - \Delta_1^n}{x_{n+2} - x_n}\ , 0 \leq n \leq N-2$$
$$\Delta_3^n = \frac{\Delta_2^{n+1} - \Delta_2^n}{x_{n+3} - x_n}\ , 0 \leq n \leq N-3$$
$$\vdots \quad \vdots$$
$$\Delta_L^n = \frac{\Delta_{L-1}^{n+1} - \Delta_{L-1}^n}{x_{n+L} - x_n}\ , 0 \leq n \leq N-L$$

Vejamos um exemplo numérico. Qual o valor de $f(x = 0,70)$ se temos os pontos da Tabela 7.1? O polinômio interpolador de Newton é:

$$P_N(x) \cong 0,70 + 1,968(x-0,1) - 0,2628(x-0,1)(x-0,25) - 0,303(x-0,1)(x-0,25)(x-0,45)$$

logo, o valor de $P_N(0,70)$ é 1,7888151. O valor "exato" é 1,7884354 (erro de apenas $-0,02\%$), sabendo que a função é $f(x) = 2\sin(x) + 0,5$. Se os pontos forem originados de um polinômio de grau N, precisamos de $N+1$ pontos para que o erro seja virtualmente nulo. Se incluirmos mais um ponto na Tabela 7.1, os cálculos anteriores não precisam ser refeitos, apenas acrescentamos os novos cálculos.

Tabela 7.1: Tabela de diferenças para formar o polinômio interpolador de Newton

n	x_n	$f(x_n)$	Δ_1^n	Δ_2^n	Δ_3^n
0	0,10	0,6996668	1,9676072	-0,2628328	-0,3029816
1	0,25	0,9948079	1,8756157	-0,4900690	*
2	0,45	1,3699311	1,5815744	*	*
3	0,85	2,0025608	*	*	*

Exemplo - Código Scilab. Podemos usar o comando "diff" para auxiliar nos cálculos dos "deltas" do interpolador de Newton. Para os pontos indicados na Tabela 7.2, queremos encontrar o valor de $f(x = 1,70)$. Apresentamos o código em seguida.

Tabela 7.2: Tabela de diferenças para formar o polinômio interpolador de Newton

n	x_n	y_n
0	0.05	1.0488088
1	0.75	1.5811388
2	1.25	1.8708287
3	2.20	2.3237900
4	3.30	2.7568098

```
// Pontos (x,y):
x = [0.05; 0.75; 1.25; 2.20; 3.30];
y = [1.0488088; 1.5811388; 1.8708287; 2.32379; 2.7568098];

// Calculando as diferenças:
dx = diff(x);
dy = diff(y)./dx;
dx2 = x(3:$)-x(1:$-2);
dy2 = diff(dy)./dx2;
dx3 = x(4:$)-x(1:$-3);
dy3 = diff(dy2)./dx3;
dx4 = x(5:$)-x(1:$-4);
dy4 = diff(dy3)./dx4;

// Preparado para mostrar os dados:
dx = [dx; 0];
dx2 = [dx2; 0; 0];
dx3 = [dx3; 0; 0];
dy = [dy; 0];
dy2 = [dy2; 0; 0];
dy3 = [dy3; 0; 0; 0];
dy4 = [dy4; 0; 0; 0; 0];

disp([x, y, dy, dy2, dy3, dy4]);

// Calculando o valor de f(x = 1,70):
xk = 1.70;
px = y(1) + dy(1)*(xk-x(1)) + dy2(1)*(xk-x(1))*(xk-x(2)) +...
 dy3(1)*(xk-x(1))*(xk-x(2))*(xk-x(3)) + ...
 dy4(1)*(xk-x(1))*(xk-x(2))*(xk-x(3))*(xk-x(4));
disp(px);
```

O valor de $f(1,70)$ é de 2,0960984. Sabendo que esses pontos são da função $f(x) = \sqrt{2x+1}$, o erro relativo é de apenas $0,0724\%$.

7.3 Interpolação de Lagrange

Um interpolador de Lagrange[1] para uma tabela com $N+1$ pontos tem a forma

$$l(x) = \sum_{n=0}^{N} L_n(x) f(x_n) \tag{7.4}$$

[1] Joseph Louis Lagrange (Turim, 25 de janeiro de 1736 - Paris, 10 de abril de 1813) foi um matemático italiano, mas viveu grande parte de sua vida na Alemanha e na França. Sua obra prima foi o tratado *Méchanique Analytique* - Mecânica Analítica.

com

$$L_n(x) = \prod_{i=0, i\neq n}^{N} \frac{(x - x_i)}{(x_n - x_i)}$$

Por exemplo, vamos considerar a Tabela 7.3. Qual o valor de $l(x = 1,70)$?

Tabela 7.3: Tabela de pontos para formar um polinômio interpolador de Lagrange

n	x_n	y_n
0	1,00	1,3722203
1	1,50	4,1916283
2	1,90	4,0904257
3	2,20	1,8465579

Neste exemplo, o polinômio interpolador de Lagrange tem a forma:

$$l(x) = y_0 \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} + y_1 \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} + y_2 \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} + y_3 \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$$

substituindo x por $x = 1,70$ e outros valores, obtemos:

$$\begin{aligned} l(x) &= 1,3722203 \frac{(1,70-1,50)(1,70-1,90)(1,70-2,20)}{(1,0-1,50)(1,0-1,90)(1,0-2,20)} \\ &+ 4,1916283 \frac{(1,70-1,0)(1,70-1,50)(1,70-1,90)}{(1,50-1,0)(1,50-1,90)(1,50-2,20)} \\ &+ 4,0904257 \frac{(1,70-1,0)(1,70-1,50)(1,70-2,90)}{(1,90-1,0)(1,90-1,50)(1,90-2,20)} \\ &+ 1,8465579 \frac{(1,70-1,0)(1,70-1,50)(1,70-1,90)}{(2,20-1,0)(2,20-1,50)(2,20-1,90)} \\ &\cong 4,4910199 \end{aligned}$$

Sabendo que a função é $f(x) = -2\cos(2x)\exp(x/2)$, o erro relativo é apenas de $0,73\%$. Vejamos agora um *script* Scilab para resolver um segundo exemplo. Considere a Tabela 7.4. O código Scilab é:

```
/// Pontos:
d = [0.   0.25    0.3894004
     1.   1.      0.7357589
     2.   1.75    0.6082088
     3.   2.5     0.4104250
     4.   3.25    0.2520324
     5.   4.      0.1465251
     6.   4.75    0.0821911];
x = d(:,2);  // x: 2a. coluna
```

Tabela 7.4: Tabela de pontos para um polinômio interpolador de Lagrange

n	x_n	y_n
0	0,25	0,3894004
1	1,00	0,7357589
2	1,75	0,6082088
3	2,50	0,4104250
4	3,25	0,2520324
5	4,00	0,1465251
6	4,75	0,0821911

```
y = d(:,3);  // y: 3a. coluna
P = ones(x);  // Produtório - inicialização
N = max(size(x));  // Número de pontos
xi = 2.10;   // valor a ser interpolado
for k=1:N
    for n=1:N
        if (abs(n-k)>0)
            then
            P(k) = P(k)*(xi-x(n))/(x(k)-x(n));
        end;
    end
end
px = sum(P.*y);  // ponto calculado
```

O valor caluclado é $p(x = 2,10) \cong 0,5140166$.

7.4 Interpolação com Scilab

Existem vários comandos no Scilab que implementam alguma forma de interpolação. Alguns desses comandos estão listados a seguir:

- interp: função de avaliação de spline cúbico;
- interp1: função de interpolação 1d;
- interp2d: função de avaliação spline bicúbica (2d);
- interp3d: função de avaliação spline 3d;
- interpln: interpolação linear;
- smooth: suavização por funções splines;
- splin: interpolação por spline cúbico;
- splin2d: interpolação por spline bicúbico em grides 2d;
- splin3d: interpolação spline em grides 3d.

A seguir, apresentamos um exemplo de código Scilab usando o comando "interp1" (ver Figura 7.2). Neste exemplo, calculamos os valores da função $y = x^2e^{-x} + x^2/100$ no intervalo $(0,5)$, mas com apenas 15 pontos. Com esses 15 pontos, recalculamos por interpolação outros 35 pontos usando o comando "interp1". Código:

```
x=linspace(0,5,15);   // "x": variando de 0 a 5, 15 pontos
```

```
y=x.*x.*exp(-x) + (x.*x/100);  // valor da função y
xx=linspace(0,5,50);  // "xx": valores a serem interpolados
yy1=interp1(x,y,xx,'linear');  // interpolação linear
yy2=interp1(x,y,xx,'spline'); // interpolação do tipo spline
plot(xx,[yy1;yy2],x,y,'*'); // gráfico
xtitle('Interpolação de uma função);
legend(['linear','spline','pontos'],a=2); // legenda.
```

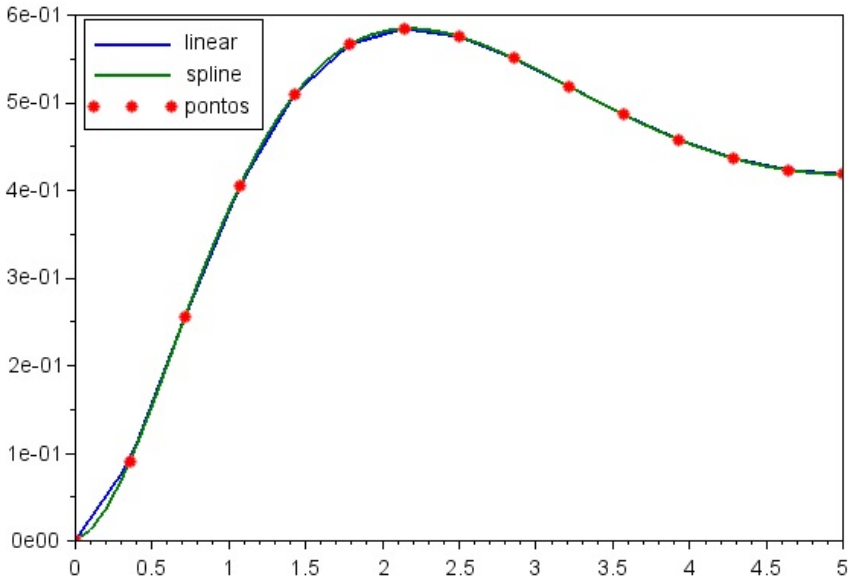


Figura 7.2: Exemplo de interpolação usando comandos do Scilab.

7.5 Extrapolação

Podemos definir extrapolação como o processo de estimar um valor fora do intervalo de pontos conhecidos. Como, em geral, não podemos ter certeza se existe uma tendência de crescimento ou decrescimento, podemos cometer grandes erros neste tipo de estimativa. A extrapolação não deixa de ser um passo no escuro. Vejamos um exemplo ilustrativo. Considere a Tabela 7.5. Qual o valor esperado para $x_5 = 0,5$?

Tabela 7.5: Tabela de pontos para extrapolação

n	x_n	y_n
0	-2,0	0,0183156
1	-1,5	0,1053992
2	-1,0	0,3678794
3	-0,5	0,7788008
4	0,0	1,0000000

Usando Lagrange, Newton e "spline", obtemos os pontos extrapolados

$$x_{Lagrange} = 0,6065307$$
$$x_{Newton} = 0,3821056$$
$$x_{spline} = 0,6155116$$
$$x_{exato} = 0,7788008$$

indicados na Figura 7.3. Percebemos que os três métodos são insatisfatórios, sendo que o de Newton apresentou o pior resultado.

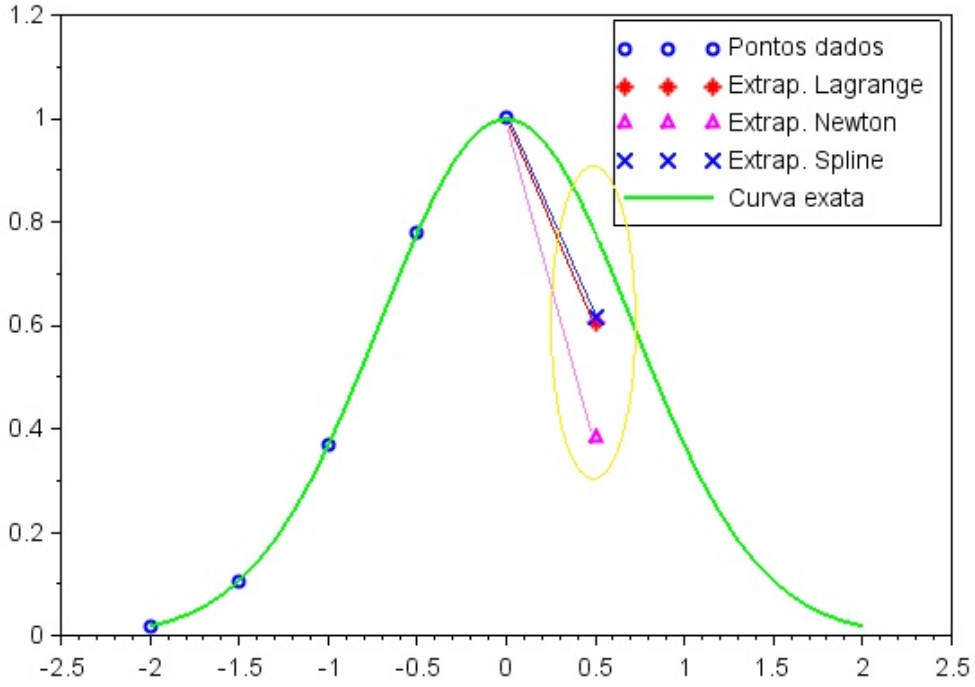


Figura 7.3: Extrapolação de uma curva suave.

7.6 Problemas

Questão 1. Quais as semelhanças e diferenças entre regressão e interpolação?

Questão 2. Usando interpolação linear, encontre o valor de $f(x = 1,5)$ se são conhecidos os pontos $(1,2)$ e $(3,5)$.

Questão 3. Considere os pontos no plano XY: $(1,1)$, $(2,3)$, $(3,4)$ e $(4,2)$ estejam ligados por uma função "suave". Encontre uma estimativa para $f(x = 1,5)$ usando a fórmula de Newton.

Questão 4. Refaça a questão anterior usando um polinômio de Lagrange.

Questão 5. Considere a tabela a seguir. Calcule o valor aproximado de $f(x = 2,2)$ usando Newton e Lagrange.

Questão 6. Refaça a questão anterior usando o comando "interp1" do Scilab.

Tabela 7.6: Exercício - item 5

n	x_n	y_n
0	0,25	0,401
1	1,00	0,750
2	1,75	0,610
3	2,50	0,400

Tabela 7.7: Exercício 7, uso do comando "interp1".

n	x_n	y_n
0	-1,0	1,6666667
1	-0,5	1,5833333
2	0,0	2,0000000
3	0,5	1,5833333
4	1,0	1,6666667
5	1,5	2,6136364
6	2,0	4,2222222
7	2,5	6,3981481
8	3,0	9,1052632

Questão 7. Analise a tabela abaixo Calcule a função no intervalo $-1,0 \leq x \leq 3$ com um passo $h = 0,1$.

Questão 8. Suponha que você disponha de uma tabela da função seno, em graus, no intervalo $0 \leq \theta \leq 90^0$, com um passo de 10^0. Calcule, aproximadamente, o valor de $y = \text{sen}(35^0)$.

Questão 9. Um foguete experimental de pequeno porte foi lançado e os dados de telemetria de sua altura em relação ao solo são apresentados na tabela a seguir. Determine a altura máxima que ele deve ter atingido. Um paraquedas é acionado pouco depois dele atingir a altura máxima, por isso a sua queda é bem lenta.

Tabela 7.8: Exercício 9.

n	$tempo_n$	$altura_n$
0	0,0	0,00
1	0,5	15,30
2	1,0	33,20
3	1,5	62,30
4	2,0	69,80
5	2,5	69,70
6	3,0	67,10
7	3,5	65,60
8	4,0	64,80

Questão 10. A população brasileira tem crescido de forma quase exponencial desde os primeiros censos. Atualmente, segue crescendo, mas em ritmo bem menor. A Tabela 7.9 indica o tamanho da

população brasileira ao longo das últimas décadas. Faça uma estimativa da população no ano de 2012: $a)$ usando interpolação linear; $b)$ usando Lagrange e os dados de 1980 a 2016; $c)$ usando o "splin" do Scilab e todos os dados da tabela; $d)$ compare e comente os resultados.

Tabela 7.9: Exercício 10.

1900	17.438.434
1920	30.635.605
1940	41.236.315
1950	51.944.397
1960	70.992.343
1970	94.508.583
1980	121.150.573
1991	146.917.459
2000	169.590.693
2010	190.755.799
2016	206.756.201

Questão 11. Considerando os dados da questão anterior, faça uma estimativa da população no ano de 2030:

(a) Lagrange - usando todos os dados;
(b) Newton - usando todos os dados;
(c) Spline - usando todos os dados;
(d) Lagrange - usando os dados de 1970 em diante;
(e) Newton - usando os dados de 1970 em diante;
(f) Spline - usando os dados de 1970 em diante;
(g) compare e comente os resultados obtidos nos itens anteriores.

Questão 12. Considere o cálculo da função $f(x) = 3xe^{-x/2}$ no pontos $x = [0;1;2;3;4]$. Calcule o valor de $f(x = 5)$ usando o conceito de extrapolação. Use os métodos de Lagrange, Newton e Spline. Compare esses resultados com o valor analítico exato.

Questão 13. Encontre um função do tipo $f(x) = a + bc^x$ (a, b, $c > 0$ são constantes) que passa pelos pontos indicados abaixo. Estime os valores de $f(x = 1,30)$ e $f(x = 3,54)$.

x : 0,0000	0,50000	1,0000	1,5000	2,0000	2,5000	3,0000
y : 0,0000	0,64575	1,5000	2,6301	4,1250	6,1026	8,7188

8. Integração numérica

O conhecimento que temos das coisas é pequeno, na verdade , quando comparado com a imensidão daquilo em que ainda somos ignorantes.
Laplace, Pierre Simon (1749-1827) foi um matemático, astrônomo e físico francês.

CALCULAR a área sob uma curva é uma motivação básica do uso de integrais. Quando a curva é de uma função que possui uma antiderivada simples de avaliar, não precisamos de um método numérico para calcular a integral dessa função. Entretanto, se dispomos apenas de pontos de uma curva não conhecida, ou quando a antiderivada não pode ser calculada analiticamente, então a solução é usar alguma técnica numérica para o cálculo da integral. Por exemplo, a função e^{x^2} não possui uma antiderivada. Uma possibilidade é substituir a função que está sendo integrada por algum polinômio, cuja a integração é fácil de calcular.

8.1 Conceito de integral

Suponha que a velocidade $v(t)$ de um carro não é constante, mas apresenta uma variação ao longo do tempo. O cálculo da distância $s(t)$ percorrida por esse carro ao longo do tempo, digamos que do instante t_1 a t_2, pode ser calculada pela integração dessa velocidade entre esses instantes:

$$s(t) = \int_{t_1}^{t_2} v(t)dt$$

A integração de uma função $f(x)$ de um ponto a até um ponto b é definida[1] por:

$$\begin{aligned} I &= \int_a^b f(x)dx \\ &= F(a) - F(b) \end{aligned}$$

[1] Desde $f(x)$ que seja limitada dentro do intervalo (a,b), entre outras condições.

onde $dF(x)/dx = f(x)$. Podemos ilustrar essa operação pela Figura 8.1.

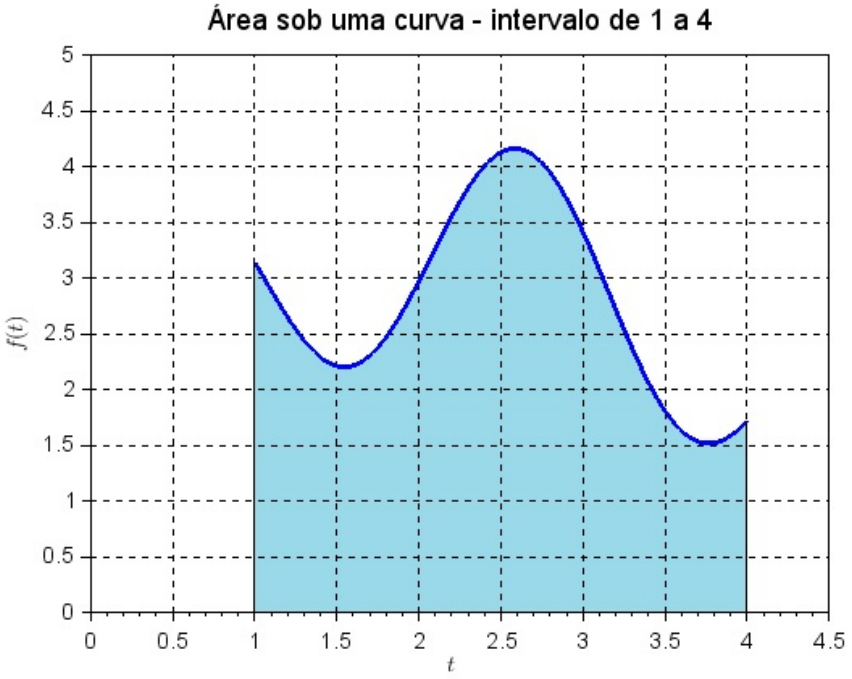


Figura 8.1: Integral como cálculo de uma área

Por exemplo,

$$\int_0^2 x^3 dx = 4$$

pois $F(x) = x^4/4$ é a antiderivada de x^3 e $F(2)$ - $F(0)$ = 4. Algumas funções, por exemplo $f(x) = \sqrt{1+x^2}$, são mais difíceis de calcular, outras como, por exemplo

$$f(x) = \sqrt{\frac{1-x^2/2}{1-x^2}}$$

é uma função elíptica e não possui uma antiderivada calculável analiticamente.

Podemos calcular a média de vários valores discretos (por exemplo, a média de notas de uma prova de métodos numéricos) y_i por:

$$m_y = \frac{\sum_{i=1}^{N} y_i}{N} \tag{8.1}$$

Já o valor médio de uma função contínua $y(t)$ no intervalo (a,b), necessário quando se deseja calcular, por exemplo, o valor médio de um sinal contínuo, pode ser calculada por:

$$m_y = \frac{\int_a^b y(t)dt}{b-a} \tag{8.2}$$

As equações (8.1) e (8.3) indicam, mesmo que de forma apenas intuitiva, a aproximação da integral com o somatório. De fato, se dividirmos uma função contínua em vários sub intervalos, a soma da área de cada um desses subintervalos levará ao valor da integral.

8.2 Fórmulas de Newton-Cotes

As fórmulas de Newton-Cotes[2] surgem da ideia de calcular a integral da função $f(x)$ tendo como base o valor da função $f(x)$ em número finito de pontos equidistantes, os pares $(x_0, f_0), (x_1, f_1), ..., (x_n, f_n)$. Com esses pontos, calculamos um polinômio $p(x)$ de grau n que passa por esses pontos e calculamos a integral de $p(x)$ no intervalo desejado, o valor dessa integral é uma aproximação da integral de $f(x)$:

$$I_p = \int_a^b p(x)dx \cong \int_a^b f(x)dx \tag{8.3}$$

Nos métodos Newton-Cotes "fechados" temos $x_0 = a$ e $x_n = b$ e $h = x_{i+1} - x_i$, $h > 0$ é passo ou a distância entre dois pontos vizinhos.

8.2.1 Regra do trapézio

A função $f(x)$ é aproximada por uma função linear $p_1(x)$. Assim, a integral de $f(x)$ é aproximada pela área de um trapézio, como ilustra a Figura 8.2:

$$\int_a^b f(x)dx \cong \int_{a=x_0}^{b=x_n} p_1(x)dx = \frac{h}{2}(f(a) + f(b)) \tag{8.4}$$

Essa fórmula fornece o valor exato da integral quando a função $f(x)$ é um polinômio de grau um. Para obtermos um menor erro, podemos dividir o intervalo e aplicar a mesma regra várias vezes, como ilustra a Figura 8.3.

Vejamos como funciona a regra do trapézio repetida ou estendida. Quando dividimos o intervalo em subintervalos igualmente espaçados, aplicamos a regra do trapézio para cada subintervalo, o que resulta em

$$I_t = \frac{h}{2}(f(x_0) + f(x_1)) + \frac{h}{2}(f(x_1) + f(x_2)) + \cdots + \frac{h}{2}(f(x_{n-1}) + f(x_n)) \tag{8.5}$$

$$= \frac{h}{2}(f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)) \tag{8.6}$$

$$= \frac{h}{2}\sum_{i=0}^{n} c_i f(x_i) \tag{8.7}$$

com $c_0 = c_n = 1$ e $c_i = 2$, para $i = 1, 2, ..., n-1$. Exemplo: calcule a integral

$$\int_1^5 \frac{1}{x+1}dx$$

[2] Isaac Newton dispensa apresentações. Roger Cotes (Burbage, 10 de julho de 1682 Cambridge, 5 de junho de 1716) foi um matemático inglês. Membro da Royal Society, conhecido por trabalhar conjuntamente com Isaac Newton, revisando a segunda edição de seu famoso livro, *Philosophiae Naturalis Principia Mathematica*. Também inventou as fórmulas da integração numérica conhecidas como fórmulas de Newton-Cotes e foi o primeiro a introduzir o que hoje é conhecido como fórmula de Euler, na forma $\log_e(\cos x + i \sin x) = ix$. Foi o primeiro Professor Plumiano de Astronomia e Filosofia Experimental da Universidade de Cambridge, de 1707 até sua morte.

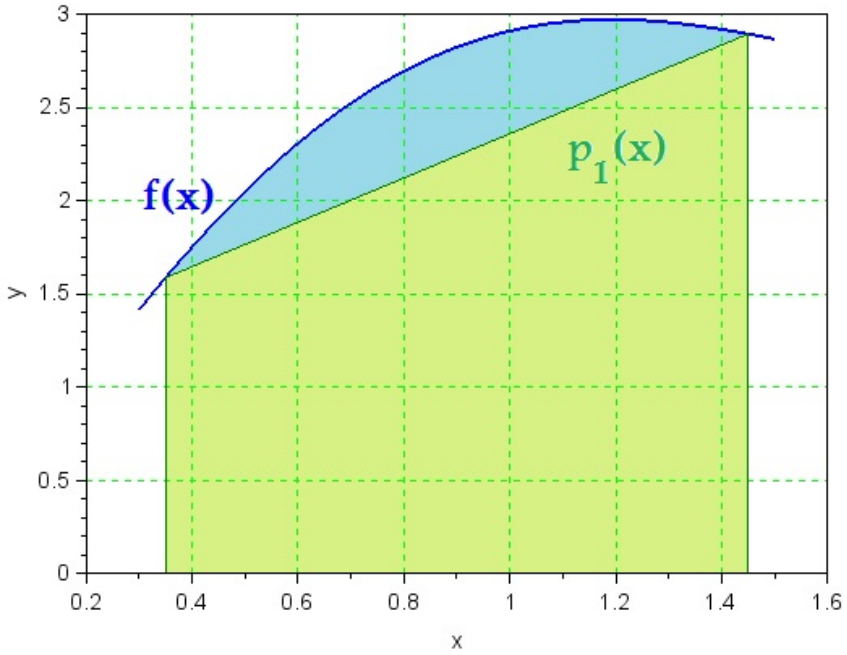


Figura 8.2: Aproximação da área da integral por um trapézio

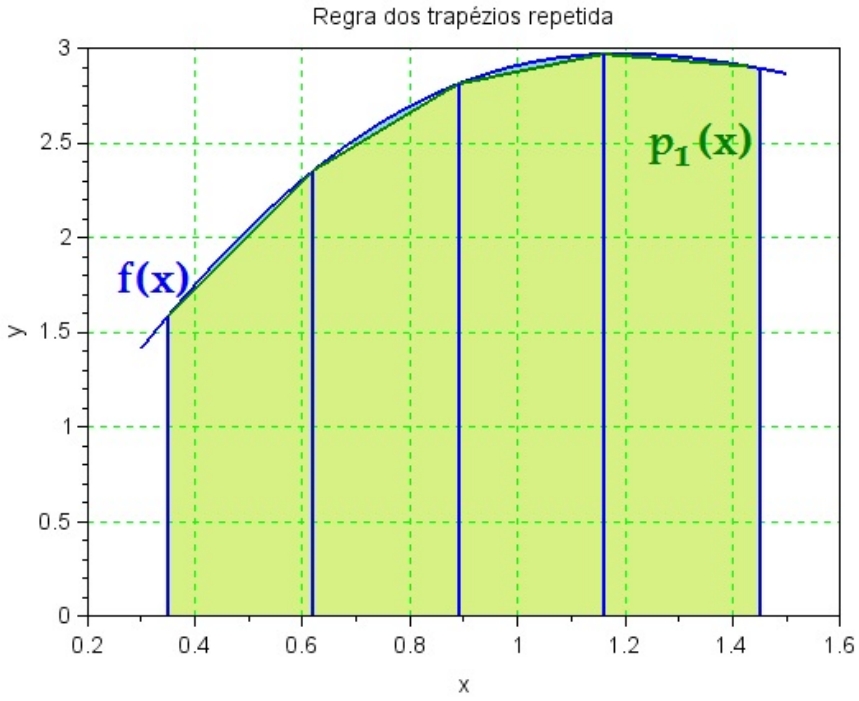


Figura 8.3: Regra dos trapézios repetida

considerando os pontos $x_i = (1,2,3,4,5)$, $h = 1$. Os valores da função nos pontos x_i são 1/2; 1/3; 1/4; 1/5; 1/6. Logo

$$I_t \cong \frac{1}{2}\left(\frac{1}{2} + 2\frac{1}{3} + 2\frac{1}{4} + 2\frac{1}{5} + \frac{1}{6}\right) = \frac{67}{60}$$

Neste exemplo, o erro é $\varepsilon = 67/60 - (\ln(6) - \ln(2)) = 0,0180544$. Naturalmente, quanto mais dividirmos o intervalo, mais cálculos serão efetuados e menor será o erro no calculo da integral. Para esta função específica $(f(x) = 1/(x+1))$, obtemos os resultados mostrados na Tabela 8.1. Observamos que o erro ε diminui com o h^2, na verdade, é possível mostrar que

$$|\varepsilon| \leq \frac{b-a}{12} h^2 L_{f2}$$

onde $L_{f2} = max(|f''(x)|)$, com $x \in [a,b]$. O código Scilab para encontrar os valores mostrados na Tabela 8.1 é:

```
dx = [1; 0.1; 0.01; 0.001];
itt = log(3); A = 1;
for kk=1:4
   x=1:dx(kk):5;
   fx=A./(x+1);
   it = inttrap(x,fx);
   erro = it - itt;
   printf('dx  *  it  *  itt *  erro \n %1.3f * %1.6f * %1.6f * %e \n',...
 dx(kk),it, itt, erro);
end;
```

a função "inttrap" do Scilab calcula a integral usando a regra do trapézio.

Tabela 8.1: Regra do trapézio repetida, função: $1/(x+1)$, para $1 \leq x \leq 5$.

h	I_t	ε - erro
1	1,116667	1.805438e-02
0,1	1,098797	1.851338e-04
0,01	1,098614	1.851847e-06
0,001	1,098612	1.851852e-08

Um segundo exemplo: calcular a integral da função $f(x) = x + \cos(2x)$ no intervalo $0 \leq x \leq 1$ com passo $h = 0,1$. Obtemos os dados indicados na Tabela 8.2. Logo,

$$\int_0^1 (x + \cos(2x))dx \cong \frac{0,1}{2}(1,000 + 2 \times 1,080 + \ldots + 2 \times 0,673 + 0,584) = 0,9531322.$$

O valor exato dessa integral é 0,9546487; o erro numérico é $\varepsilon = -0,0015165$.

8.2.2 Primeira Regra de Simpson

Consideraremos agora um polinômio $p(x)$ de grau dois que passa pelos pontos $x_0 = a$, $x_1 = x0 + h$ e $x_2 = x0 + 2h$ para fazer a estimativa da integral de $f(x)$. Usando a fórmula de Lagrange, podemos

Tabela 8.2: Regra do trapézio repetida, função: $x+\cos(2x)$, para $0 \leq x \leq 1$.

x	$f(x)$
0,000	1,000000
0,100	1,080067
0,200	1,121061
0,300	1,125336
0,400	1,096707
0,500	1,040302
0,600	0,962358
0,700	0,869967
0,800	0,770800
0,900	0,672798
1,000	0,583853

escrever esse polinômio por:

$$p_2(x) = \frac{(x-x_1)(x-x_2)}{(-h)(-2h)}f(x_0) + \frac{(x-x_0)(x-x_2)}{(h)(-h)}f(x_1) + \frac{(x-x_0)(x-x_1)}{(2h)(h)}f(x_2)$$

Dessa forma,

$$\begin{aligned}\int_{x_0}^{x_2} f(x)dx &\cong \int_{x_0}^{x_2} p_2(x)dx \\ &= \frac{f(x_0)}{2h^2}\int_{x_0}^{x_2}(x-x_1)(x-x_2)dx - \frac{f(x_1)}{h^2}\int_{x_0}^{x_2}(x-x_0)(x-x_2)dx \\ &+ \frac{f(x_2)}{2h^2}\int_{x_0}^{x_2}(x-x_0)(x-x_1)dx \\ &= \frac{h}{3}\left(f(x_0)+4f(x_1)+f(x_2)\right)\end{aligned}$$

que é a primeira regra de Simpson[3] ou regra $1/3$ de Simpson, que ilustramos na Figura 8.4. Um exemplo simples: seja a função $f(x) = x^2 - 2x + 1$, integrada no intervalo $x_0 = 1$ a $x_2 = 3$, com passo $h = 1$. O valor dessa integral usando a primeira regra de Simpson é:

$$\begin{aligned}I_s &= \frac{1}{3}\left(f(1)+4f(2)+f(3)\right) \\ &= \frac{1}{3}(0+4+4) \\ &= \frac{8}{3}\end{aligned}$$

[3]Thomas Simpson nasceu em 20 de agosto de 1710 em Market Bosworth, Leicestershire, Inglaterra. Morreu em 14 maio de 1761 em Market Bosworth. Thomas recebeu pouca educação formal. Ele frequentou a escola no Market Bosworth por um tempo, mas seu primeiro trabalho foi como um tecelão. Ele foi autodidata em matemática. Ele trabalhou com probabilidade, mas ficou mais conhecido pelos seus trabalhos sobre interpolação e integração numérica.

que é o *valor* exato desta integral. Esse fato não é surpresa, pois para funções polinômiais de até grau três o resultado deve ser realmente correto. Vejamos um segundo exemplo (para $h = 1$):

$$\begin{aligned} I &= \int_0^3 2 - x^2\sqrt{x} + e^x dx \\ &\cong \frac{1}{3}(3,7182818 + 4 \times 3,7322018 + 6,4970797) \\ &\cong 8,3813896 \end{aligned}$$

O valor exato é dado por:

$$\begin{aligned} I &= \int_0^3 2 - x^2\sqrt{x} + e^x dx \\ &= \left[2x - \frac{2}{7}x^{7/2} + e^x\right]_1^3 \\ &= 8,2914346 \end{aligned}$$

o que corresponde a um erro absoluto $|\varepsilon| = 0,0899550$. Da mesma forma que na regra do trapézio, podemos dividir o intervalo (a,b) em subintervalos, tornando o passo h menor e reduzindo o erro absoluto da integração. Assim, a fórmula para cálculo da integral usando a primeira regra de Simpson fica:

$$I_{sr} = \frac{h}{3}\left(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right) \quad (8.8)$$

A equação 8.8 precisa ter $n \geq 5$, sendo n um valor ímpar. Vamos repetir o exemplo anterior usando um passo $h = 1/4$. Os dados estão na Tabela 8.3. Neste caso, $I_{sr} = 8,2918257$ e o erro $|\varepsilon| = 0,0003911$ é cerca de 230 vezes menor, o que é compatível com a expressão do erro:

$$|\varepsilon_{sr}| \leq \frac{b-a}{180}h^4 L_{f4} \quad (8.9)$$

com $L_{f4} = max(|f^{iv}(x)|)$ e $x \in [a,b]$.

8.2.3 Segunda Regra de Simpson

Na segunda regra de Simpson, o intervalo de integração contém 4 pontos (x_0, x_1, x_2, x_3), com $x_0 = a$, $x_3 = b$, $x_1 = x_0 + h$, $x_2 = x_1 + h$, e o valor da integral é dado por (a dedução é semelhante a primeira regra, mas é mais trabalhosa):

$$\int_a^b f(x)dx \cong \frac{3h}{8}\left(f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)\right) \quad (8.10)$$

A expressão 8.10 fornece valores corretos para polinômios de até grau três, nesse ponto não é melhor que a primeira regra de Simpson. Tomemos como exemplo inicial a função já estudada $f(x) = 2 - x^2\sqrt{x} + e^x$. Para integrar no intervalo $(1,3)$, o passo h deverá ser $h = 2/3$, ver Tabela 8.4. O erro $|\varepsilon| = 0,0405007$ que é cerca da metade do obtido pela primeira regra com o uso de apenas mais um ponto dentro do intervalo.

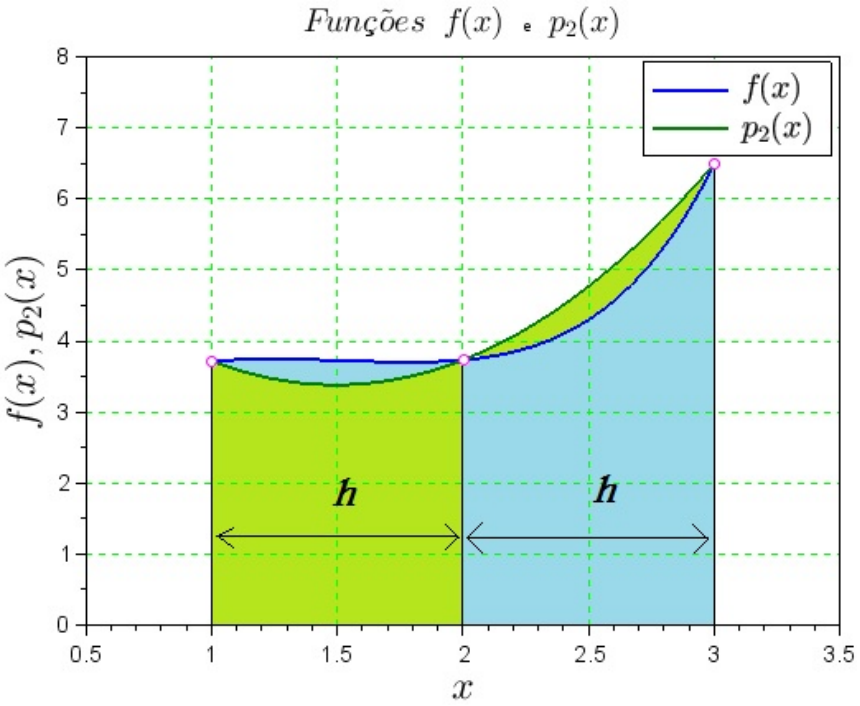


Figura 8.4: Regra $1/3$ de Simpson

Tabela 8.3: Primeira regra de Simpson repetida,
função: $2-x^2\sqrt{x}+e^x$, para $1 \leq x \leq 3$, $h = 1/4$.

x_i	$f(x_i)$	c_i	$c_i f(x_i)$
1,00	3,718282	1	3,718282
1,25	3,743415	4	14,973659
1,50	3,726013	2	7,452026
1,75	3,703296	4	14,813184
2,00	3,732202	2	7,464404
2,25	3,893986	4	15,575943
2,50	4,300376	2	8,600753
2,75	5,101644	4	20,406578
3,00	6,497080	1	6,497080
	$(h/3)\sum c_i f(x_i)$	=	8,2918257

Tabela 8.4: Segunda regra de Simpson,
função: $2-x^2\sqrt{x}+e^x$, para $1 \leq x \leq 3$, $h = 2/3$.

x_i	$f(x_i)$	c_i	$c_i f(x_i)$
1,00	3,718282	1	3,718282
1,67	3,708394	3	11,125183
2,33	3,995732	3	11,987197
3,00	6,497080	1	6,497080
	$(h/3)\sum c_i f(x_i)$	=	8,3319353

Mais uma vez, se dividirmos o intervalo em subintervalos, obteremos melhores resultados. Assim,

$$\int_a^b f(x)dx \cong \frac{3h}{8}\left(f(x_0)+3f(x_1)+3f(x_2)+2f(x_3)+3f(x_4)+3f(x_5)+2f(x_6)+\ldots+f(x_n)\right) \tag{8.11}$$

neste caso, n é impar, $n \geq 7$, e podemos expressá-lo por: $n = 4+3k$, $k = 1,2,3,\ldots$. Aplicando a segunda regra de Simpson repetida para a função $2-x^2\sqrt{x}+e^x$, com passo $h = 1/3$, obtemos os dados da Tabela 8.5. Observamos que o erro $|\varepsilon| = 0,0027287$, cerca de 15 vezes menor e está próximo do valor esperado ($erro \approx (1/2)^4 = 1/16$).

Tabela 8.5: Segunda regra de Simpson repetida, função: $2-x^2\sqrt{x}+e^x$, para $1 \leq x \leq 3$, $h = 2/3$.

x_i	$f(x_i)$	c_i	$c_i f(x_i)$
1,000	3,718282	1	3,718282
1,333	3,740867	3	11,222601
1,667	3,708394	3	11,125183
2,000	3,732202	2	7,464404
2,333	3,995732	3	11,987197
2,667	4,779520	3	14,338561
3,000	6,497080	1	6,497080
	$(3h/8)\sum c_i f(x_i)$	=	8,2941633

8.2.4 Regra de Boole

A regra de Boole[4] para integração usa 5 pontos e pode ser expressa por:

$$\int_a^b f(x)dx \cong \frac{2h}{45}\left(7f(x_0)+32f(x_1)+12f(x_2)+32f(x_3)+7f(x_4)\right) \tag{8.12}$$

A expressão 8.12 fornece valores corretos para polinômios de até grau cinco, bem superior às regras de Simpson. Mais uma vez, usemos como exemplo a função $f(x) = 2-x^2\sqrt{x}+e^x$. Para integrar no intervalo $(1,3)$, o passo h deverá ser $h = 1/2$, ver Tabela 8.6. O erro $|\varepsilon| = 0,0005302$ é inferior ao obtido pela segunda regra de Simpson.

Naturalmente, podemos dividir o intervalo (a,b) em mais pontos e criar uma regra de Boole estendida.

8.2.5 Outras regras

Quando os valores de uma função $f(x)$ estão tabulados nos pontos x_i igualmente espaçados por $h = x_(i+1) - x_i$, então $f_1 = f(x_1)$, $f_2 = f(x_2)$, ..., $f_7 = f(x_7)$. Então, a **regra de Hardy** que se

[4]George Boole nasceu em 2 de novembro de 1815 em Lincoln, Lincolnshire, Inglaterra. Morreu em 8 de dezembro de 1864 em Ballintemple, County Cork, Irlanda. Boole foi um matemático e filósofo britânico, criador da álgebra booleana, fundamental para o desenvolvimento da computação moderna. Algumas vezes, a regra de Boole é chamada erroneamente de regra de Bode devido um erro tipográfico (Abramowitz e Stegun 1972, p. 886).

Tabela 8.6: Regra de Boole,
função: $2-x^2\sqrt{x}+e^x$, para $1 \leq x \leq 3$, $h = 1/2$.

x_i	$f(x_i)$	c_i	$c_i f(x_i)$
1,000	3,718282	7	26,027973
1,500	3,726013	32	119,232420
2,000	3,732202	12	44,786422
2,500	4,300376	32	137,612041
3,000	6,497080	7	45,479558
	$(2h/45)\sum c_i f(x_i)$	=	8,2914346

aproxima da integral de $f(x)$ é dada pela fórmula de Newton-Cotes:

$$\int_{x_1}^{x_7} f(x)dx = \frac{h}{100}(28f_1 + 162f_2 + 220f_4 + 162f_6 + 28f_7) \tag{8.13}$$

A **regra de Weddle** é expressa por:

$$\int_{x_1}^{x_7} f(x)dx = \frac{3h}{10}(f_1 + 5f_2 + f_3 + 6f_4 + f_5 + 5f_6 + f_7) \tag{8.14}$$

A **regra de Shovelton** é dada pela fórmula de Newton-Cotes:

$$\int_{x_1}^{x_{11}} f(x)dx = \frac{5h}{126}(8(f_1 + f_{11}) + 35(f_2 + f_4 + f_8 + f_{10}) + 15(f_3 + f_5 + f_7 + f_9) + 36f_6) \tag{8.15}$$

Para n pontos igualmente espaçados, podemos usar a **regra de Durand**:

$$\int_{x_1}^{x_n} f(x)dx = h\left(\frac{2}{5}f_1 + \frac{11}{10}f_2 + f_3 + \ldots + f_{n-2} + \frac{11}{10}f_{n-1} + \frac{2}{5}f_n\right) \tag{8.16}$$

8.2.6 Fórmulas de Newton-Cotes Abertas

As fórmulas que vimos até agora são *fórmulas fechadas*, isto é, incluem os pontos extremos do intervalo. Em alguns casos, pode ser importante não incluir esses pontos extremos. Nessas situações, podemos usar as fórmulas abertas. Algumas delas (com o erro de truncamento incluído):

$$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{2}(f_1 + f_2) + \frac{3h^3}{4}f''(\xi) \tag{8.17}$$

$$\int_{x_0}^{x_4} f(x)dx = \frac{4h}{3}(2f_1 - f_2 + 2f_3) + \frac{28h^5}{90}f^{(4)}(\xi) \tag{8.18}$$

$$\int_{x_0}^{x_5} f(x)dx = \frac{5h}{24}(11f_1 + f_2 + f_3 + 11f_4) + \frac{95h^5}{144}f^{(4)}(\xi) \tag{8.19}$$

$$\int_{x_0}^{x_6} f(x)dx = \frac{6h}{20}(11f_1 - 14f_2 + 26f_3 - 14f_4 + 11f_5) + \frac{41h^7}{140}f^{(6)}(\xi) \tag{8.20}$$

Vejamos um exemplo: calcular a integral da função $f(x) = -(x-1)(x-3)$ no intervalo $(1,4)$ com passo $h = 0,5$. Código Scilab:

```
f=zeros(1,5)
for k=1:0.5:3
    x=k;
    f(k*2-1) = -((x-1)*(x-3));
end
p = [0, 2, -1, 2, 0];
fp = f.*p;
sfp = sum(fp)*4*0.5/3
disp(sfp);
```

O valor calculado é $I = 1,3333333$, que é exato, pois $f^{(4)}(\xi) = 0$

8.2.7 Uma comparação entre os métodos de Newton-Cotes

Para finalizar o estudo dos métodos de Newton-Cotes, realizaremos uma exemplo comparativo usando quatro funções distintas (ver Figura 8.5). O objetivo é ver qual método apresenta menor erro no cálculo da integral para um dado conjunto de pontos. Para comparação ser mais justa, usaremos a mesma função $f(x)$ e um total de 13 pontos para cada método. Todos os métodos permitem o uso desse número de pontos nas suas formas estendidas. Mostramos o código Scilab a seguir:

```
function fx = fnc(x, kf)
    select kf
        case 1 then fx = 2 - 4*x.*x.*x + exp(2*x);
        case 2 then fx = 4*exp(-x).*sin(2*x);
        case 3 then fx = (x).^(1/3) + exp(x/2);
        case 4 then fx = 3 + sin(2*x) - x.*x;
     end;
endfunction

clc;
n=0:12; // 13 pontos
xh= n/8;  // x varia de 0 a 1.5, h = 1/8
p1 = [1 2 2 2 2 2 2 2 2 2 2 2 1];  // regra dos trapézios
p2 = [1 4 2 4 2 4 2 4 2 4 2 4 1];  // 1o. regra de Simpson
p3 = [1 3 3 2 3 3 2 3 3 2 3 3 1];  // 2o. regra de Simpson
p4 = [7 32 12 32 14 32 12 32 14 32 12 32 7];  // Regra de Bode;
h = 1/8; // passo

for kf=1:4
    fh = fnc(xh,kf); // chamada da função
    select kf
       case 1 then It = 2*xh(13) - (xh(13)^4) + exp(2*xh(13))/2...
    -(2*xh(1) - (xh(1)^4) + exp(2*xh(1))/2);
       case 2 then It = 4*exp(-xh(13))*(-2*cos(2*xh(13)) + ...
sin(2*xh(13)))/5 + 8/5;
       case 3 then It = (3/5)*(xh(13))^(5/3) + 2*exp(xh(13)/2) - 2;
       case 4 then It = 3*xh(13) -cos(2*xh(13))/2 - xh(13)^3/3 + 1/2;
```

```
    end
    ittrap = sum(p1.*fh.*h/2); //disp([ittrap, It]);
    itsimp1 = sum(p2.*fh.*h/3);
    itsimp2 = sum(p3.*fh.*h*3/8);
    itbode = sum(p4.*fh.*h*2/45);
    mprintf('%1.6f & %1.6f  & %1.6f & %1.6f \n',...
    ittrap, itsimp1, itsimp2, itbode);
    mprintf('%1.6f & %1.6f  & %1.6f & %1.6f \n',...
    ittrap-It, itsimp1-It, itsimp2-It, itbode-It);
end;
```

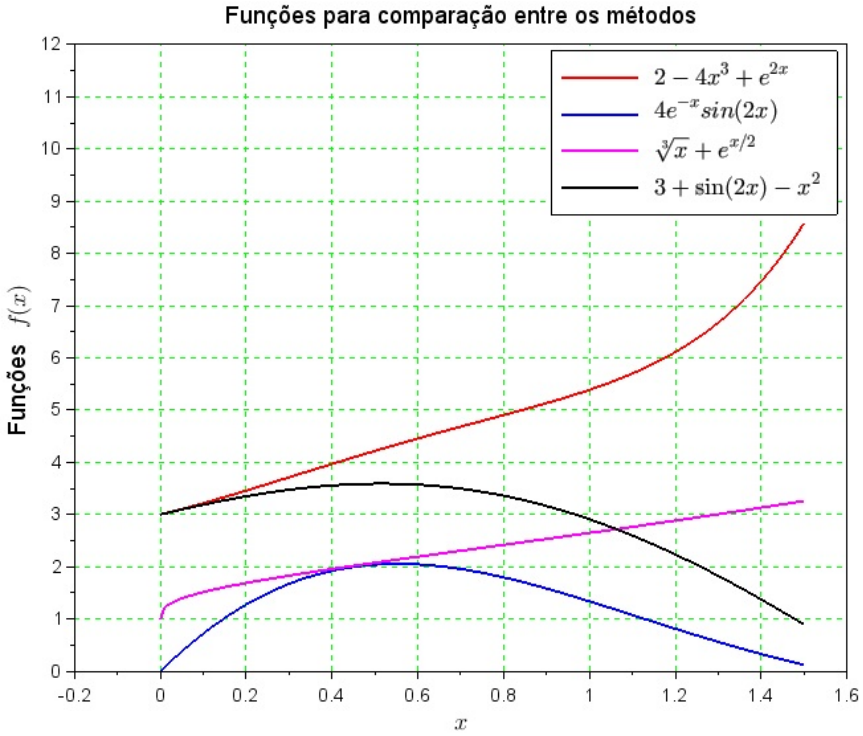


Figura 8.5: Funções para comparação entre os métodos.

Na Tabela 8.7, apresentamos os resultados obtidos. Notamos que os métodos apresentam um desempenho semelhante, com uma ligeira vantagem para a regra de Boole e a regra dos trapézios sempre apresentando o pior desempenho.

8.3 Integração de Romberg

Está técnica foi desenvolvida por Werner Romberg,[5] que publicou o método que leva seu nome em 1955. Basicamente, essa técnica usa a regra dos trapézios sucessivamente e obtém um resultado de

[5]Werner Romberg (19092003) foi criado em Berlim. Entrou na Universidade de Heidelberg em 1928, onde estudou matemática e ciências físicas. Também estudou em Munique. Seus interesses incluíram o uso de computadores para computação numérica.

Tabela 8.7: Comparação entre os métodos de integração para as funções $f_1(x) = 2 - 4x^3 + e^{2x}$, $f_2(x) = 4e^{-x}sen(2x)$, $f_2(x) = \sqrt[3]{x} + e^{x/2}$ e $f_3(x) = 3 + sen(2x) - x^2$.

	Trap.	Simpson 1	Simpson 2	Boole
$\int f_1(x)dx$:	7,494762	7,480474	7,480728	7,480273
ε_1:	0,014494	0,000206	0,000459	0,000005
$\int f_2(x)dx$:	1,915360	1,928260	1,928277	1,928247
ε_2:	-0,063266	-0,050366	-0,050349	-0,050379
$\int f_3(x)dx$:	3,505528	3,513252	3,511679	3,514118
ε_3:	0,092194	0,099918	0,098345	0,100784
$\int f_4(x)dx$:	4,360902	4,370018	4,370046	4,369996
ε_4:	-0,009094	0,000022	0,000049	-0,000001

grande acurácia pela combinação dos resultados anteriores.
A aplicação da regra dos trapézios estendida pode ser representada por:

$$I_1 = I_t(h_1) + E(h_1)$$

onde I é o valor exato da integral, $h_1 = (b-a)/N$, $N \geq 1$ é um número inteiro e $E(h) \cong -(b-a)h^2 f''(\xi)/12$ é o erro estimado da integração numérica. Quanto menor o valor do passo h, menor o erro. Para um passo $h_2 = h_1/2$:

$$\begin{aligned} I_2 =& I_t(h_2) + E(h_2) \\ =& I_t(h_2) - (b-a)(h_2)^2 f''(\xi)/12 \\ =& I_t(h_2) - (1/4)(b-a)(h_1)^2 f''(\xi)/12 \end{aligned}$$

Observado essa última expressão, podemos, aproximadamente, cancelar o erro:

$$\begin{aligned} 3I \cong& 4I_2 - I_1 \\ \cong& 4I_t(h_2) - (b-a)(h_1)^2 f''(\xi)/12 - I_t(h_1) + (b-a)(h_1)^2 f''(\xi)/12 \\ \cong& 4I_t(h_2) - I_t(h_1) \end{aligned}$$

Chegando ao resultado desejado:

$$I \cong \frac{4I_t(h_2) - I_t(h_1)}{3} \tag{8.21}$$

Em resumo: tendo uma estimativa I_1 da integral usando a regra dos trapézios com o passo h, fazemos uma nova estimativa I_2 com passo $h/2$ e a integral desejada tem valor aproximado por $I = (4I_2 - I_1)/3$. Vejamos um exemplo.
Consideremos a função $f(x) = x^4 + 1$, no intervalo $(1,2)$, com passo $h = 1$. A integral de $f(x)$ pela regra dos trapézios é $I_1 = \frac{1}{2}(2+17) = 9,5$. Com passo $h = 1/2$: $I_2 = \frac{1}{4}(2+2,0625+17) = 7,78125$. Então, o valor mais acurado é:

$$\begin{aligned} I \cong& \frac{4 \times 7,78125 - 9,5}{3} \\ \cong& 7,2083333 \end{aligned}$$

que é um valor mais próximo do valor exato $7,2$. Usando unicamente a regra dos trapézios, o passo teria que ser $h = 0,0588235$ para um erro semelhante.
Podemos usar o mesmo raciocínio para primeira regra de Simpson ($h_2 = h_1/2$, I_s é integral usando a primeira regra de Simpson):

$$I \cong \frac{16I_s(h_2) - I_s(h_1)}{15} \tag{8.22}$$

Podemos generalizar o uso da equação 8.21 para gerar o algoritmo de Romberg aplicando sucessivamente a expressão:

$$I_{j,k} \cong \frac{4^{k-1}I_{j+1,k-1} - I_{j,k-1}}{4^{k-1} - 1} \tag{8.23}$$

onde $I_{j,k}$ é a integral melhorada, $I_{j+1,k-1}$ e $I_{j,k-1}$ são as versões menos acuradas. A integração usando o algoritmo de Romberg exige o conhecimento da função $f(x)$, pois dados tabulados dificilmente teriam os pontos necessários para os refinamentos. Vejamos um exemplo. Consideremos a função $f(x) = x^4 - 2x^2 + 3$, integrada no intervalo $(1,3)$, $h = 2$ como passo inicial. O valor exato dessa integral é 37,066667. Calculamos o valor aproximado da integral com $h = 1, 1/2, 1/4, 1/8$, em seguida usamos a expressão 8.23. Os resultados obtidos estão na Tabela 8.8. Observamos que o erro final é zero.

Tabela 8.8: Algoritmo de Romberg, função: $x^4 - 2x^2 + 3$.

$h(i)$	$I(h_i)$	$I(L,1)$	$I(L,2)$	$I(L,3)$
2,00	68,000000	37,333333	37,066667	37,066667
1,00	45,000000	37,083333	37,066667	0,000000
0,50	39,062500	37,067708	0,000000	0,000000
0,25	37,566406	0,000000	0,000000	0,000000

$$I(L+1,k) = \frac{4^k I(L+1,k-1) - I(L,k-1)}{4^k - 1}$$

Note que $(4 \times 45 - 68)/3 = 37,3333$; $(4 \times 39,0625 - 45,000)/3 = 37,083333$; e assim por diante. Um código Scilab que pode gerar os dados da Tabela 8.8 é:

```
function f = fx(x)   //função
    f = x.^4 - 2*x.^2 + 3;
endfunction

function g = fxi(x)   //integral da função
    g = x.^5/5 - 2*x.^3/3 + 3*x;
endfunction

MR = zeros(4,4);  //Matriz de Romberg (MR)
h = 2;     // passo inicial
for k=1:4
```

```
    x = 1:h:3;
    y = fx(x);
    MR(k,1) = inttrap(x,y);  //Primeira coluna da MR
    h = h/2;
end;

for C=2:4   // Algoritmo de Romberg - sem critério de parada
    k4 = 4^(C-1);
    for L=1:(5-C)
        MR(L,C) = (k4*MR(L+1,C-1) - MR(L,C-1))/(k4-1);
    end
end

for L=1:4   // Mostrando os resultados
    mprintf('%1.6f & %1.6f & %1.6f & %1.6f \n',MR(L,1:4));
end

valor_teorico = fxi(3) - fxi(1);  // valor teórico da integral
erro = MR(1,4) - valor_teorico;  // erro
mprintf('Erro = %e,  *** %1.6f',erro,valor_teorico);
```

Vejamos um segundo exemplo. A função $f(x) = 4 + 4\cos(x) - 8e^{-4x}$, o valor da sua integral no intervalo $(0,2)$ é igual a $9,6378606$. Aplicando o algoritmo descrito acima, com passo inicial $h = 2$, obtemos os dados da Tabela 8.9. Observamos que o erro já não é desprezível, mas $\varepsilon \cong -1,6383 \times 10^{-3}$. Ainda assim, esse resultado é muito superior à simples aplicação da regra dos trapézios com um passo $h = 0,125$.

Tabela 8.9: Algoritmo de Romberg, função: $4 + 4\cos(x) - 8e^{-4x}$.

$h(i)$	$I(h_i)$	$I(L,1)$	$I(L,2)$	$I(L,3)$	$I(L,4)$
2,000	2,332729	8,797155	9,569108	9,636222	9,637850
1,000	7,181049	9,520861	9,635174	9,637844	0,000000
0,500	8,935908	9,628029	9,637802	0,000000	0,000000
0,250	9,454999	9,637191	0,000000	0,000000	0,000000
0,125	9,591643	0,000000	0,000000	0,000000	0,000000

$$I(L+1,k) = \frac{4^k I(L+1,k-1) - I(L,k-1)}{4^k - 1}$$

8.4 Quadratura de Gauss

Observe a Figura 8.6. É fácil percebermos que um deslocamento os pontos do intervalo de integração pode levar a uma melhor estimativa da integral da função. Essa é a estratégia usada pels técnicas de integração de Gauss-Legendre[6]. A quadratura de Gauss exige o conhecimento da função $f(x)$ que

[6]Johann Carl Friedrich **Gauss** nasceu em Braunschweig, 30 de abril de 1777 e faleceu em Göttingen, 23 de fevereiro de 1855. Filho de pais humildes, foi um matemático, astrônomo e físico alemão que contribuiu muito em diversas áreas

está sendo integrada.

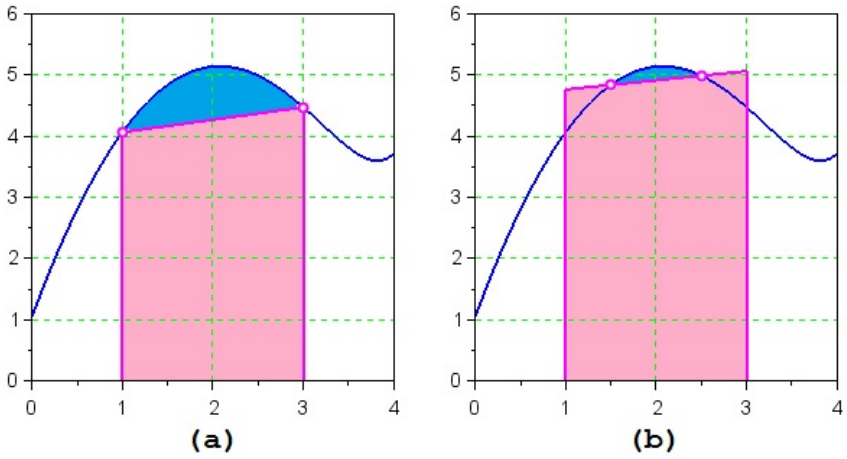


Figura 8.6: Cálculo da integral de uma função $f(x)$ usando regra do trapézio (a) pontos extremos; (b) estimativa melhorada pelo deslocamento dos pontos.

Considere uma função $f(x)$ continua no intervalo $(-1,1)$. Podemos calcular uma estimativa da integral I usando dois pontos por:

$$I \cong w_0 f(x_0) + w_1 f(x_1) \tag{8.24}$$

onde w_0 e w_1 são coeficientes ainda desconhecidos e $-1 < x_0 < 0$, $0 < x_1 < 1$ são os pontos de interesse dentro do intervalo $(-1,1)$. Dessa forma, nós temos 4 incógnitas e precisamos de 4 equações. Podemos obter w_i e x_i pela solução do seguinte sistema de equações:

$$w_0 + w_1 = \int_{-1}^{1} 1dx = 2 \qquad w_0 x_0 + w_1 x_1 = \int_{-1}^{1} x dx = 0$$
$$w_0 x_0^2 + w_1 x_1^2 = \int_{-1}^{1} x^2 dx = \frac{2}{3} \qquad w_0 x_0^3 + w_1 x_1^3 = \int_{-1}^{1} x^3 dx = 0$$

cuja solução é $w_0 = 1$, $w_1 = 1$, $x_0 = -1/\sqrt{3}$ $x_1 = 1/\sqrt{3}$. Logo, a fórmula de Gauss-Legendre para dois pontos é

$$I \cong f\left(\frac{-\sqrt{3}}{3}\right) + f\left(\frac{+\sqrt{3}}{3}\right) \tag{8.25}$$

Vejamos um exemplo. Considere a função $f(x) = cos(x)$, o valor exato da integral de $f(x)$ de -1 a 1 é $I = 1,682942$. Usando a regra do trapézio, encontramos o valor $I_t = 1,5403023$. Já com a

da ciência, dentre elas a teoria dos números, estatística, análise matemática, geometria diferencial, geodésia, geofísica, eletroestática, astronomia e óptica. Adrien-Marie **Legendre** nasceu em Paris em 18 de setembro de 1752, em uma família rica. Foi-lhe dada uma educação de qualidade superior no Collège Mazarin em Paris, elaborando sua tese em física e matemática em 1770. De 1775 a 1780, lecionou na École Militaire, em Paris, e a partir de 1795 na École Normale, e foi associado ao Bureau des Longitudes. Fez importantes contribuições à estatística, teoria dos números, álgebra abstrata e análise matemática. Foi eleito membro da Royal Society em 1789. Legendre morreu em Paris em 9 de janeiro de 1833.

quadratura de Gauss o valor calculado é

$$\begin{aligned} I_G =& f\left(\frac{-\sqrt{3}}{3}\right) + f\left(\frac{+\sqrt{3}}{3}\right) \\ =& \cos\left(\frac{-\sqrt{3}}{3}\right) + \cos\left(\frac{-\sqrt{3}}{3}\right) \\ =& 1,6758237 \end{aligned}$$

que apresenta um erro bem menor quando comparado com o erro a regra do trapézio simples, na verdade, para este caso, o erro é até menor que usando a primeira regra de Simpson.

Em geral, o intervalo de interesse não é $(-1,1)$, mas (a,b). Esse inconveniente pode ser contornado por uma traca de variável: $x = \alpha z + \beta$, tal que $x = a \to z = -1$ e $x = b \to z = +1$:

$$\alpha = \frac{b-a}{2} \text{ e } \beta = \frac{a+b}{2}$$

com $dx = \alpha dz$. Por exemplo:

$$\int_1^3 f(x)dx = \alpha \int_{-1}^{+1} f(\alpha z + \beta)dz$$

De forma semelhante, podemos obter fórmulas para a quadratura de Gauss-Legendre para mais pontos. Apresentamos na Tabela 8.10 os pesos w_i e os pontos x_i usados na quadratura de Gauss-Legendre.

Tabela 8.10: Quadratura de Gauss-Legendre: pesos w_i e argumentos x_i para função $f(x)$

Pontos	Pesos w_i	Argumentos x_i	Erro de truncamento
2	$w_0 = 1$	$x_0 = -\sqrt{3}/3$	$\cong f^{(4)}(\xi)$
	$w_1 = 1$	$x_1 = +\sqrt{3}/3$	
3	$w_0 = 5/9$	$x_0 = -\sqrt{3/5}$	$\cong f^{(6)}(\xi)$
	$w_1 = 8/9$	$x_1 = 0$	
	$w_2 = 5/9$	$x_2 = +\sqrt{3/5}$	
4	$w_0 = (18-\sqrt{30})/36$	$x_0 = -\sqrt{525+70\sqrt{30}}/35$	$\cong f^{(8)}(\xi)$
	$w_1 = (18+\sqrt{30})/36$	$x_1 = -\sqrt{525-70\sqrt{30}}/35$	
	$w_2 = (18+\sqrt{30})/36$	$x_2 = +\sqrt{525+70\sqrt{30}}/35$	
	$w_3 = (18-\sqrt{30})/36$	$x_3 = +\sqrt{525-70\sqrt{30}}/35$	
5	$w_0 = \frac{1}{900}(322-13\sqrt{70})$	$x_1 = -\frac{1}{21}\sqrt{245+14\sqrt{70}}$	$\cong f^{(10)}(\xi)$
	$w_1 = \frac{1}{900}(322+13\sqrt{70})$	$x_2 = -\frac{1}{21}\sqrt{245-14\sqrt{70}}$	
	$w_2 = \frac{128}{225}$	$x_3 = 0$	
	$w_3 = \frac{1}{900}(322+13\sqrt{70})$	$x_4 = +\frac{1}{21}\sqrt{245-14\sqrt{70}}$	
	$w_4 = \frac{1}{900}(322-13\sqrt{70})$	$x_5 = +\frac{1}{21}\sqrt{245+14\sqrt{70}}$	

Vejamos agora um exemplo mais completo do uso quadratura de Gauss-Legendre: calcular o valor da integral da função $f(t) = t^2\cos(t/2)$ no intervalo $(0,\pi)$ usando 4 pontos.

Solução. Devemos inicial fazer uma troca de variável: $t = \alpha x + \beta$:

$$\alpha = \frac{\pi}{2} \qquad \beta = \frac{\pi}{2}$$
$$t = \frac{\pi}{2}(x+1) \qquad dt = \frac{\pi}{2}dx$$

Assim,

$$\begin{aligned} I &= \int_0^{\pi} t^2 \cos(t/2) dt \\ &= \frac{\pi}{2}\int_{-1}^{1} \frac{\pi^2}{4}(x+1)^2 \cos\left(\frac{\pi}{4}(x+1)\right) dx \\ &= \frac{\pi^3}{8}\int_{-1}^{1} (x+1)^2 \cos\left(\frac{\pi}{4}(x+1)\right) dx \end{aligned}$$

Finalmente, usando os pontos indicados na Tabela 8.10:

$$\begin{aligned} I_G =& \frac{\pi^3}{8}\left(w_0(x_0+1)^2\cos\left(\frac{\pi}{4}(x_0+1)\right) + w_1(x_1+1)^2\cos\left(\frac{\pi}{4}(x_1+1)\right)\right) + \\ & \frac{\pi^3}{8}\left(w_2(x_2+1)^2\cos\left(\frac{\pi}{4}(x_2+1)\right) + w_3(x_3+1)^2\cos\left(\frac{\pi}{4}(x_3+1)\right)\right) \\ =& 3,7392165 \end{aligned}$$

Esse valor está correto até a quinta casa decimal, pois o valor exato é $I = 2\pi^2 - 16 = 3,7392088$.

8.5 Integração imprópria

Ocorrem integrais impróprias quando a função $f(x)$ é ilimitada, isto é, $f(x) \to \pm\infty$ dentro do intervalo de interesse ou quando um dos limites de integração é infinito. Alguns exemplos:

$$\int_{-1}^{2} \frac{1}{x} dx \qquad \int_0^{\infty} \frac{1}{x+1} dx$$
$$\int_{-\infty}^{+\infty} \frac{e^{-x^2}}{2} dx \qquad \int_0^3 \frac{e^x}{x} dx$$

Nestes casos, devemos usar alguma estratégia para evitar a singularidade ou os limites infinitos. Por exemplo,

$$\begin{aligned} I_{imp} &= \int_0^{\infty} f(x)dx \\ &= \int_0^a f(x)dx + \int_a^{\infty} f(x)dx \\ &\cong \int_0^a f(x)dx \end{aligned}$$

tal que para algum valor de a o erro $|\int_a^{\infty} f(x)dx| \leq \varepsilon$. Mostraremos alguns exemplos de como resolver numericamente essas integrais. Outra possibilidade é fazer uma troca de variáveis. Por

exemplo, fazendo a troca $t = x/(1+x)$ o intervalo $(0,+\infty)$ é convertido em $(0,1)$. Vejamos um exemplo:

$$I_{imp} = \int_0^{\infty} \frac{1}{1+x^2} dx$$

$$\text{fazendo } t = \frac{x}{1+x},\ dx = \frac{1}{1-t^2} dt$$

$$I_t = \int_0^1 \frac{1}{1-2t+2t^2} dt$$

dividindo o intervalo $(0,1)$ com um passo $h = 0,1$; obtemos os valores indicados na Tabela 8.11. O valor aproximado da integral usando a primeira regra de Simpson é $I_{imp} = 1,570795$, o erro é $|I_{imp} - \pi/2| = 9,389 \times 10^{-9}$, um erro bem pequeno.

Tabela 8.11: Integral da função $f(t) = \frac{1}{1-2t+2t^2}$.

t	$f(t)$
0,000	1,000000
0,100	1,219512
0,200	1,470588
0,300	1,724138
0,400	1,923077
0,500	2,000000
0,600	1,923077
0,700	1,724138
0,800	1,470588
0,900	1,219512
1,000	1,000000
$\int \cong$	1,570795

Já a integral

$$I_{imp} = \int_0^1 \frac{\sqrt{1+x^2}}{\sqrt{1-x^2}} dx$$

é imprópria porque $f(x) \to \infty$ em $x = 1$. Neste caso, podemos fazer a seguinte troca de variável: $x = \sin(\omega)$, $dx = \cos(\omega) d\omega$, e o novo intervalo de integração é $(0, \pi/2)$. Assim,

$$I_\omega = \int_0^{\pi/2} \sqrt{1+\sin^2(\omega)} d\omega$$

que podemos calcular facilmente por qualquer método numérico.

8.6 Integração múltipla

Em algumas situações, estamos interessados em calcular integrais do tipo

$$I_D = \int_{x_1}^{x_2} \int_{y_1}^{y_2} f(x,y) dy dx$$

Nestes casos, uma estratégia possível é dividir a região retangular R definida por pelos pontos (x_1, y_1) e (x_2, y_2) em vários pontos $(x_i, y_j) \in R$ e fazer uma soma ponderada dos valores de $f(x_i, y_j)$ de forma similar ao uso, por exemplo, da Regra de Simpson, para a obtenção do volume definido pela função $f(x,y)$. A área de integração pode ser dividida em passos h_x e h_y diferentes. Vejamos um exemplo:

$$I_D = \int_0^1 \int_0^1 xydydx$$

Podemos usar um passo $h_x = 1/4$ e um passo $h_y = 1/3$. Assim, calculamos a função nos pontos $(0,0)$, $(0,1/3)$, $(0,2/3)$, $(0,1)$, $(1/4,0)$, etc, como indicamos na Tabela 8.12.

Tabela 8.12: Pontos da função $f(x,y) = xy$.

x/y	0	1/4	2/4	3/4	1
0	0.0	0.0	0.0	0.0	0.0
1/3	0.0	1/12	2/12	3/12	1/3
2/3	0.0	2/12	4/12	6/12	2/3
1	0.0	1/4	2/4	3/4	1

Podemos, então, calcular volume como

$$\begin{aligned}
I_D &\cong h_x h_y \sum_{n=0}^{n=5} \sum_{m=0}^{m=4} f(x_n, y_m) \\
&= h_x h_y (f(0,0) + f(0,1/3) + f(0,2/3) + f(0,1) + f(1/4,0) + \cdots + f(1,1)) \\
&= \frac{1}{3}\frac{1}{4}(0+0+\cdots+1/12+2/12+3/12+1/3+2/12+\cdots+1) \\
&= 5/12.
\end{aligned}$$

Esse valor de I_D não é muito satisfatório, pois o valor *exato* desta integral é $1/4$. Onde está o erro? Nesse cálculo não fizemos o uso de "pesos" adequados. Se usarmos a primeira Regra de Simpson repetida para o eixo X e a segunda regra de Simpson para o eixo Y (ver Tabela 8.13). Dessa forma, o cálculo dessa integral é

$$\begin{aligned}
I_D &\cong \frac{3h_x}{8}\frac{h_y}{3} \sum_{n=0}^{n=4} \sum_{m=0}^{m=3} p(n,m) f(x_n, y_m) \\
&= \frac{3h_x}{8}\frac{h_y}{3}(f(0,0) + 3 \times f(0,1/3) + 3 \times f(0,2/3) + f(0,1) + \cdots + f(1,1)) \\
&= \frac{3}{16}\frac{1}{12}(0+0+\cdots+1/12+2/12+3/12+1/3+2/12+\cdots+1) \\
&= \frac{1}{4}.
\end{aligned}$$

Que é o valor exato. Note que os pesos usados são: $p(n,m) = c_X(n)c_Y(m)$, com $c_X = [1,4,2,4,1]$ e $c_Y = [1,3,3,1]$.

Vejamos um exemplo mais desafiador: qual o volume gerado pela função

$$f(x,y) = \sqrt{1 + \text{sen}(x + 3y/2)}$$

Tabela 8.13: Pesos para o cálculo de I_D.

x/y	0	1/4	2/4	3/4	1
0	1	4	2	4	1
1/3	3	12	6	12	3
2/3	3	12	6	12	3
1	1	4	2	4	1

para $0 \leq x \leq 2$ e $0 \leq y \leq 2$, como mostra Figura 8.7. Se usarmos $h_x = h_y = 1/4$ e a primeira regra de Simpson repetida, obtemos $I_D \cong 4,4114672$. O código Scilab a seguir mostra todos os cálculos necessários.

```
hx=1/4; hy=1/4;  // passos
a=1; b=1;
F = zeros(9,9);
px = 2*ones(1,9);
px(2:2:8)=4;
px(1)=1; px(9)=1;
py = px; P = F;
for x=0:hx:2
    for y=0:hy:2
        F(a,b) = sqrt(1+sin(x+3*y/2)); //exp(-x)*exp(-y);
        P(a,b) = py(a)*px(b);
        a=a+1;
    end
    a=1;
    b=b+1;
end
FP = F.*P;
I2 = hx*hy*(1/3)*(1/3)*sum(sum(FP)); ///(sum(sum(P)));
disp(I2);
```

8.7 Integração numérica com Scilab

Podemos ainda usar os recursos disponíveis no Scilab para integração numérica. Os comandos básicos para integração numérica são:

- int2d integral definida 2d por quadratura e cubatura;
- int3d integral definida 3d pelo método da quadratura e cubatura;
- intc integral de Cauchy;
- integrate integração pela quadratura;
- intg integral definida;
- intl integral de Cauchy;
- intsplin integração de dados experimentais por interpolação por *spline*;
- inttrap integração de dados experimentais por interpolação trapezoidal.

Vejamos um exemplo do uso dos comandos "inttrap", "intsplin", "intg" e "integrate" para o cálculo da integral da função $f(x) = e^{-x/2} + \sin(x) + (1/2)\cos(3x)$ (ver Figura 8.8) no intervalo

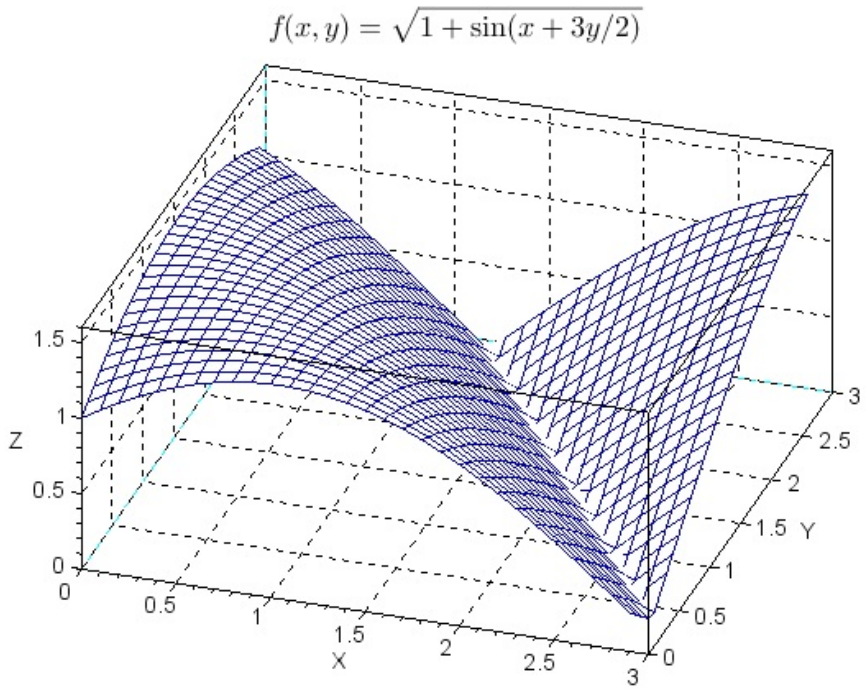


Figura 8.7: Função exemplo para integração dupla.

$(0;2,75)$:

$$I_{exemplo} = \int_0^{2,75} e^{-x/2} + \sin(x) + (1/2)\cos(3x)dx$$

cujo valor teórico é $I_{exemplo} = -2e^{-2,75/2} - \cos(2,75) + 0,5\sin(3 \times 2,75)/3 + 3 \cong 3,5723905554$.
Código:

```
function y=fx(x)
    y=exp(-x/2) + sin(x) + 0.5*cos(3*x);
endfunction

x = 0:0.25:2.75;
f = fx(x);
xg = 0:0.01:2.75; fg = fx(xg);
close; plot(x,f,'-o',xg,fg);

It = -2*exp(-x($)/2) - cos(x($)) + 0.5*sin(3*x($))/3 + 3;
fim = max(size(x));
Itt = inttrap(x,f); er1 = abs(It - Itt)*100/It;
Its = intsplin(x,f); er2 = abs(It - Its)*100/It;
Itg =intg(x(1),x(fim),fx); er3 = abs(It - Itg)*100/It;
Itq =integrate('exp(-x/2) + sin(x) + 0.5*cos(3*x)','x',x(1),x($));
er4 = abs(It - Itq)*100/It;
mprintf('It=%1.6f, Itt=%1.6f, Its=%1.6f, Itg=%1.6f, Itq=%1.6f\n',...
```

```
It,Itt,Its,Itg,Itq);
mprintf('*****, e-Itt=%1.6f, e-Its=%1.6f, e-Itg=%1.6f, e-Itq=%1.6f\n',...
er1,er2,er3,er4);
```

Resultados (valores das integrais numéricas e erros relativos):

```
Itt=3.557027,   Its=3.572949,   Itg=3.572391,   Itq=3.572391
e-Itt=0.430076, e-Its=0.015621, e-Itg=0.000000, e-Itq=0.000000
```

Como esperado, a integral usando a regra dos trapézios apresentou um erro relativamente grande, usando interpolação por *spline* o erro foi menor, mas, usando quadratura e integração definida, os resultados foram ainda melhores.

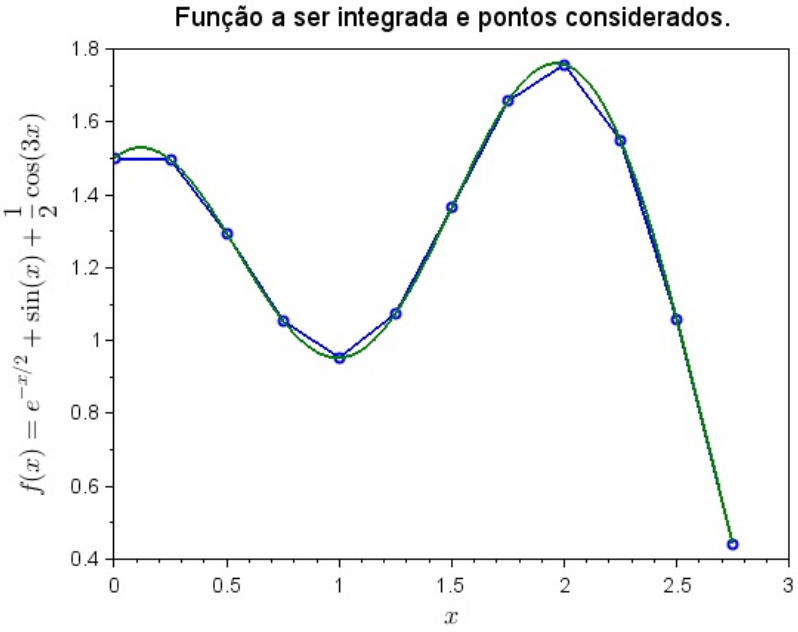


Figura 8.8: Exemplo para integração usando os recursos do Scilab

8.8 Convolução numérica com Scilab

A operação de convolução (substantivo feminino: ato ou efeito de enrolar-se para dentro.) é definida como

$$y(t) = x(t) * h(t) \tag{8.26}$$

$$= \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau \tag{8.27}$$

Em palavras, podemos dizer que a convolução, em matemática e na área de processamento de sinal, é um operador linear que, a partir de duas funções dadas, resulta numa terceira função que mede a área subentendida pela superposição delas em função do deslocamento existente entre elas. Se essa definição não ficou clara, não se preocupe, apenas siga o que diz a expressão matemática.

Em geral, usamos o símbolo $*$ para indicar a convolução entre dois sinais, sejam eles discretos ou contínios. Existe também a convolução bidimensional usada, por exemplo, no processamento de imagens, mas isso foge ao escopo deste livro. No Scilab podemos, naturalmente, efetuar somente convoluções numéricas e de sinais discretos, mas, com algum cuidado, que podem se aproximar bastante dos valores analíticos. Do "help" do Scilab tiramos:

- Chamada: [y] = convol(h,x), onde os argumentos são:
- h: um vetor "menor", primeira entrada.
- x: um segundo vetor de entrada (vetor mais longo).
- y: um vetor com o resultado da convolução.

Vejamos um exemplo simples: seja $x(t) = e^{-t}$ para $t >= 0$ e $h(t) = e^{-t}$ para $t \geq 0$, $y(t) = e^{-t} * e^{-t}$, resulta em $y(t) = te^{-t}$, com $t \geq 0$. Para realizar no Scilab este cálculo e mostrar graficamente, devemos escolher um passo dt relativamente pequeno, digamos, $dt = 0,005$. Neste caso, obtemos o resultado mostrado na Figura 8.9. Mostramos o código Scilab a seguir.

```
close; clc;
dt = 0.005; t=0:dt:5; // passo e tempo discreto
h = exp(-t);  x = h; // funções h(t) e x(t)
y = dt*convol(x,h); // convolução, note o "dt*"
y = [0, y(1:max(size(x))-1)]; // ajuste
yt = t.*exp(-t); // solução teórica
plot(t,y,t,yt); xgrid; // gráfico
xlabel('tempo'); ylabel('y(t)');
legend('Convolução numérica','Covolução teórica');
```

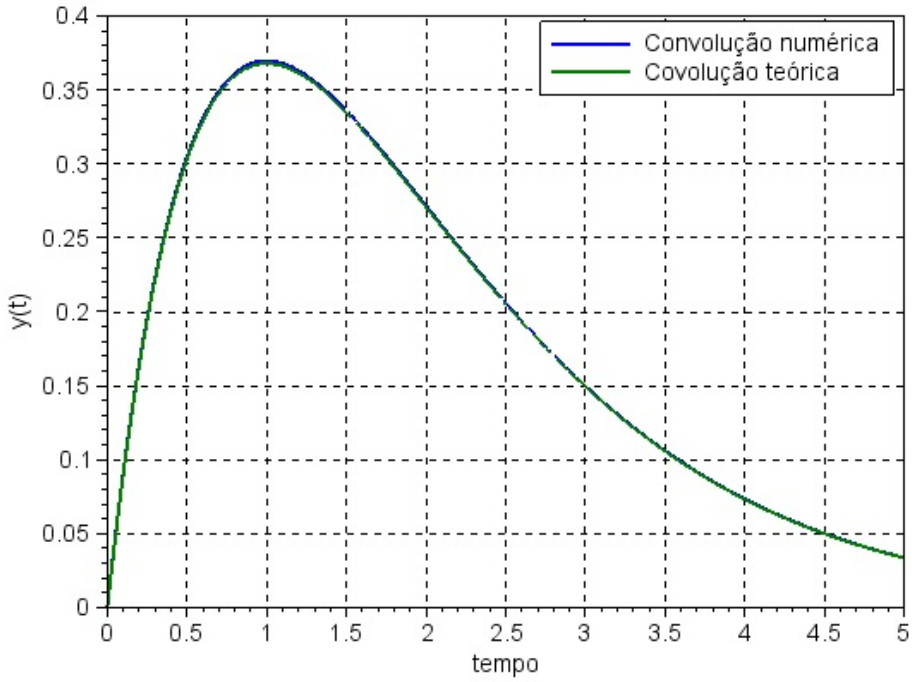


Figura 8.9: Convolução de e^{-t} com e^{-t}

8.9 Problemas

Questão 1. Calcule as seguintes integrais usando a regra dos trapézios, estipule um passo adequado para cada caso.

$$a.\int_0^1 x^2+x dx \qquad b.\int_0^1 cos(x)dx \qquad c.\int_{-1}^1 e^{-x^2}dx$$
$$d.\int_0^1 \frac{1}{1+x}dx \qquad e.\int_0^1 \sqrt{1-x}dx \qquad f.\int_0^1 \frac{x}{2-x}dx$$

Questão 2. Refaça a primeira questão usando a primeira de Simpson.

Questão 3. Refaça a primeira questão usando a segunda de Simpson.

Questão 4. Refaça a primeira questão usando a regra de Boole.

Questão 5. Calcule a integral

$$\int_0^\pi (1+\text{sen}(2x))dx$$

a. analiticamente; b. pela regra dos Trapézios repetida ($h=\pi/4$); c. pela primeira regra de Simpson repetida ($h=\pi/4$). Compare os erros.

Questão 6. A integral tem valor igual a $\pi/4$, usando a segunda regra de Simpson, escolha um passo h de tal forma que o erro absoluto do cálculo numérico da integral seja menor que $0,01$.

$$\int_0^1 \sqrt{1-x^2}dx$$

Questão 7. Uma função $f(x)$ apresenta os seguintes valores:

$x: 1,000$	$1,250$	$1,500$	$1,75$	$2,000$
$f(x): 1,841$	$2,069$	$2,275$	$2,41$	$2,425$

Calcule a integral de $f(x)$ usando a regra dos trapézios composta e a regra de Boole.

Questão 8. Use o algoritmo de Romberg para calcular a seguinte integral:

$$\int_0^4 e^{-x}\cos(x/4)dx$$

use os passos $h=1,00$; $h=0,50$ e $h=0,25$.

Questão 9. Usando o método de sua escolha, calcule a integral

$$\int_0^3 \frac{\cos(x)}{x+1}dx$$

Questão 10. Refaça a questão anterior usando a técnica de quadratura de Gauss.

Questão 11. Usando fórmula aberta 8.20, calcule a integral

$$\int_{-3}^3 \text{sen}(3e^{-x^2})dx$$

Questão 12. Calcule as integrais impróprias

$$a. \int_0^\infty \frac{x}{x^3+1}dx \qquad b. \int_0^\infty \frac{\cos(x)}{x^3+1}dx$$
$$c. \int_0^\infty \frac{\sqrt{x}}{x^2+1}dx \qquad d. \int_1^\infty \frac{\ln(x)}{x^2+1}dx$$
$$e. \int_0^\infty \frac{\sin x}{x^2+1}dx \qquad f. \int_{-\infty}^\infty \frac{\cos(x)}{x^2+1}dx$$

Questão 13. Verifique o valor da integral imprópria:

$$\int_0^\infty \frac{x}{e^x-1}dx = \frac{\pi^2}{6}$$

Questão 14. Calcule as integrais duplas a seguir:

$$I_D = \int_0^3 \int_0^3 \sqrt{x^2+y^2}dydx \qquad I_D = \int_0^3 \int_0^3 \sqrt{x^2+y^2}dydx$$
$$I_D = \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \cos((x+y)/2)dydx \qquad I_D = \int_1^3 \int_1^3 \ln(x+y)dydx$$

Questão 15. Considere a função $f(x,y) = e^{-(x^2+y^2)}$, faça uma estimativa do volume total gerado por essa função e o plano XY. Use $h_x = h_y = 0,2$, defina a área no plano XY. O gráfico desta função é mostrado na Figura 8.10.

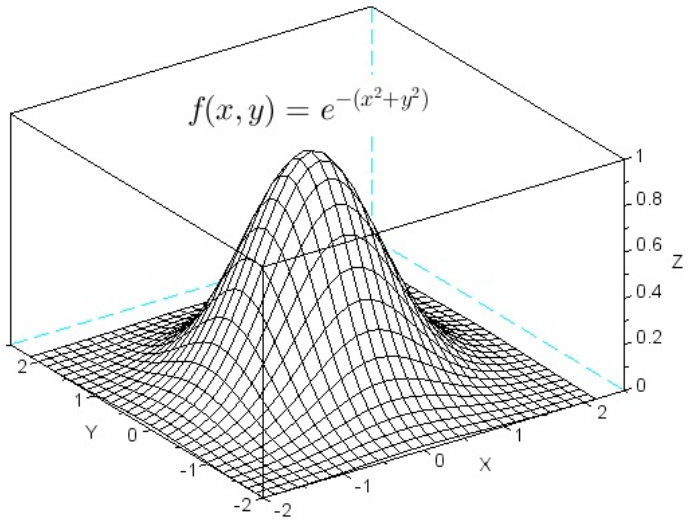


Figura 8.10: Volume sob a função $f(x,y) = e^{-(x^2+y^2)}$.

Questão 16. Refaça a questão anterior para função $f(x,y) = \text{sen}(3e^{-(x^2+y^2)})$.

Questão 17. Calcule o gráfico da convolução entre o pulso $h(t)$, definido como sendo $h(t) = 1$ para $0 \leq t \leq 2$ e $h(t) = 0$ para outros valores de t, e o sinal $x(t) = \cos(2t)$, para $t \geq 0$.

9. Derivação numérica

A experiência não nos ensina as essências das coisas.
Baruch de Espinoza (1632 - 1677), filósofo do século XVII.

A Derivada de uma função nos indica a taxa de variação dessa função em relação a uma variável. Um exemplo simples: a velocidade de um carro pode ser medida em m/s, isto é, a velocidade é a medida da taxa de variação da distância pelo tempo decorrido. Muitos problemas de física e engenharia podem ser expressos por meio de taxas de variação. Por exemplo, em circuitos elétricos, a corrente em um capacitor é proporcional à variação temporal da tensão:

$$i_C = C\frac{dV_C}{dt}. \tag{9.1}$$

Podemos definir a derivada de uma função como

$$\frac{df(t)}{dt} = \lim_{\Delta t \to 0} \frac{f(t+\Delta t) - f(t)}{\Delta t} \tag{9.2}$$

Quando a função não é conhecida ou diferenciação analítica é impossível, o cálculo deve ser feito numericamente. Podemos realizar esse cálculo com um conjunto e pontos discretos. Esses pontos podem ser oriundos de uma função $f(t)$ conhecida ou de uma tabela de valores. Podemos ainda, através dos pontos da tabela, obter uma curva analítica (função de aproximação) que passe pelos pontos e que seja diferenciada com facilidade, como ilustra a Figura 9.1. Neste breve capítulo, aprenderemos algumas técnicas de cálculo numérico da derivada de uma função ou de um conjunto de pontos tabelados.

9.1 Aproximação por diferenças finitas

A derivada $f'(t)$ de uma função $f(t)$, no ponto $t = t_0$, é definida como sendo:

$$idf\frac{df(t)}{dt}|_{t=t_0} = f'(t_0) = \lim_{t \to t_0} \frac{f(t) - f(t_0)}{t - t_0} \tag{9.3}$$

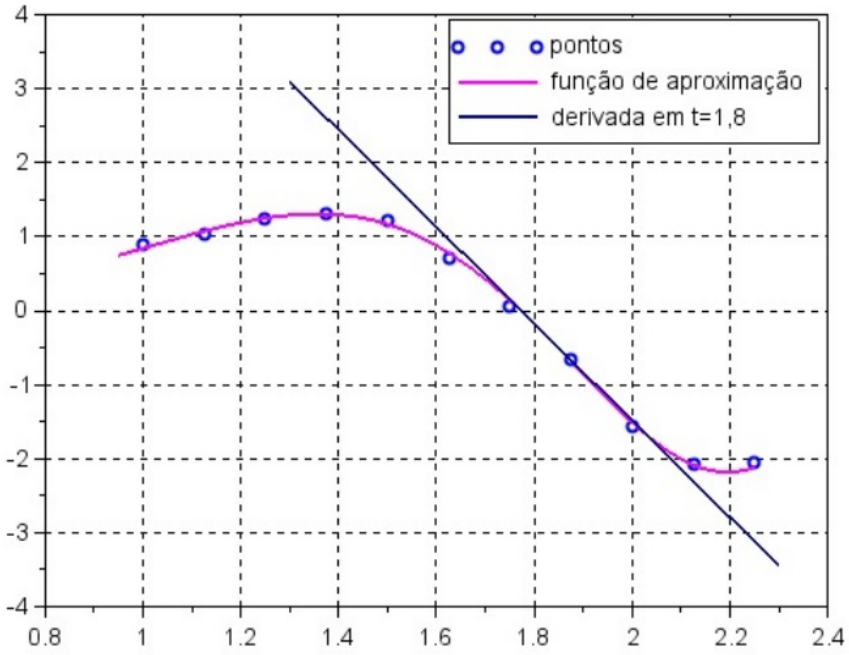


Figura 9.1: Pontos disponíveis, função de aproximação que passa pelos pontos, derivada numérica usando a função.

Esse derivada pode ser aproximada por diferentes valores de t. A medida que t se aproxima de t_0, a derivada se aproxima da reta tangente à função $f(t)$ no ponto $t = t_0$. Podemos usar, basicamente, três formas para o cálculo numérico dessa derivada:

- diferença progressiva:

$$\frac{df(t)}{dt}|_{t=t_0} \cong \frac{f(t_0+h)-f(t_0)}{h}$$

- diferença regressiva:

$$\frac{df(t)}{dt}|_{t=t_0} \cong \frac{f(t_0)-f(t_0-h)}{h}$$

- diferença central:

$$\frac{df(t)}{dt}|_{t=t_0} \cong \frac{f(t_0+h)-f(t_0-h)}{2h}$$

com h sendo um valor "pequeno". Vejamos um exemplo numérico para ilustrar esses conceitos. Vamos considerar o cálculo da derivda a função $f(t) = t^2 + \cos(\pi t)$ no ponto $t = 1$, sendo conhecidos os pontos $t = 0,75$ e $t = 1,25$.

Solução. O valor da função nos pontos considerados são: $f(0,75) = -0,1446068$, $f(1) = 0$ e $f(1,25) = 0,8553932$. A derivada analítica é $f'(t) = 2t - \pi\,\text{sen}(\pi t)$, em $t = 1$, $f'(t = 1) = 2$. Os valores numéricos são:

- diferença progressiva:

$$\frac{df(t)}{dt}|_{t=1} \cong \frac{f(1,25)-f(1)}{1,25-1} = \frac{0,8553932}{0,25} = 3,4215729$$

- diferença regressiva:

$$\frac{df(t)}{dt}|_{t=1} \cong \frac{f(1)-f(0,75)}{1-0,75} = \frac{0,1446068}{0,25} = 0,5784271$$

- diferença central:

$$\frac{df(t)}{dt}|_{t=1} \cong \frac{f(1,25)-f(0,75)}{1,25-0,75} = \frac{0,8553932+0,1446068}{0,5} = 2$$

Percebemos que o erro cometido no cálculo da derivada é menor (neste exemplo específico é zero!) para a diferença central. Na Figura 9.2, temos um exemplo do cálculo da derivada da função $f(t) = \sin(2t)e^{-t}$ usando diferenças progressiva, regressiva e central.

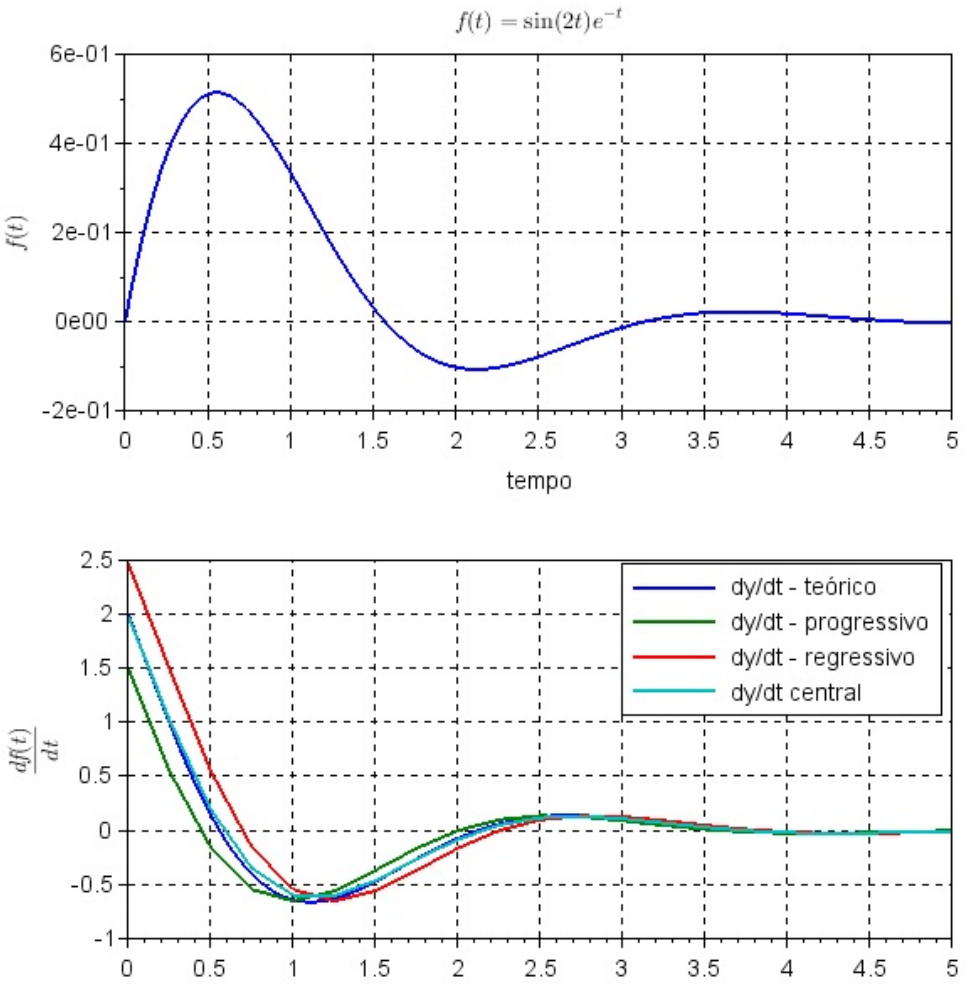


Figura 9.2: Exemplo de cálculo de derivadas, passo $\Delta t = 0,25$.

Naturalmente, quanto menor o valor do passo h, melhor a aproximação da derivada. Porém, existe um limite para o valor mínimo desse passo a partir do qual o erro aumenta, pois os erros de arredondamento começam a contribuir de forma significativa para o erro total. Vejamos um exemplo

simples: calcular a derivada progressiva da função $f(x) = \cos(x)$ em $x = 2,0$, com o passo iniciando em $0,1$ e sendo reduzido progressivamente. Os dados obtidos estão na Tabela 9.1. Neste exemplo, percebemos que para $h < 10^{-8}$ o erro no cálculo da derivada começa a aumentar.

Tabela 9.1: Pontos não igualmente espaçados.

h	f'_n	$erro$
1,000e-01	-8,8699e-01	2,2305e-02
1,000e-02	-9,0720e-01	2,0959e-03
1,000e-03	-9,0909e-01	2,0822e-04
1,000e-04	-9,0928e-01	2,0809e-05
1,000e-05	-9,0930e-01	2,0807e-06
1,000e-06	-9,0930e-01	2,0797e-07
1,000e-07	-9,0930e-01	2,2231e-08
1,000e-08	-9,0930e-01	6,1331e-09
1,000e-09	-9,0930e-01	4,3827e-08
1,000e-10	-9,0930e-01	3,4475e-07
1,000e-11	-9,0930e-01	2,9859e-06
1,000e-12	-9,0933e-01	3,0741e-05

9.2 Derivada com alta acurácia

Podemos usar a Série de Taylor para melhorar a precisão do cálculo da derivada de uma função. Podemos expressar a Série de Taylor progressiva por:

$$f(t_i+h) = f(t_i) + f'(t_i)h + f''(t_i)\frac{h^2}{2!} + f'''(t_i)\frac{h^3}{3!} + \cdots \tag{9.4}$$

logo, podemos escrever a derivada primeira como

$$f'(t_i) = \frac{f(t_i+h) - f(t_i)}{h} - f''(t_i)\frac{h}{2} + O(h^2) \tag{9.5}$$

Podemos estimar $f''(t_i)$ por

$$f''(t_i) = \frac{f(t_i+2h) - 2f(t_i+h) + f(t_i)}{h^2} + O(h),$$

finalmente,

$$f'(t_i) = \frac{-f(t_i+2h) + 4f(t_i+h) - 3f(t_i)}{2h} + O(h^2). \tag{9.6}$$

Usando a Série de Taylor regressiva:

$$f(t_i-h) = f(t_i) - f'(t_i)h + f''(t_i)\frac{h^2}{2!} - f'''(t_i)\frac{h^3}{3!} + \cdots \tag{9.7}$$

Combinando as equações (9.4) e (9.7), obtemos:

$$f'(t_i) = \frac{f(t_i+h) - f(t_i-h)}{2h} + O(h^2). \tag{9.8}$$

De forma geral, com mais pontos podemos realizar uma melhor estimativa da derivada de uma função ou o cálculo de derivadas de ordem superior. Dessa forma, temos as seguintes fórmulas para as derivadas centradas (todas com erro da ordem de h^4):

- primeira derivada:

$$f'(t_i) = \frac{-f(t_i+2h) + 8f(t_i+h) - 8f(t_i-h) + f(t_i-2h)}{12h}$$

- segunda derivada:

$$f''(t_i) = \frac{-f(t_i+2h) + 16f(t_i+h) - 30f(t_i) + 16f(t_i-h) - f(t_i-2h)}{12h^2}$$

- terceira derivada:

$$f'''(t_i) = \frac{-f(t_i+3h) + 8f(t_i+2h) - 13f(t_i+h) + 13f(t_i-h) - 8f(t_i-2h) + f(t_i-3h)}{8h^3}$$

- quarta derivada:

$$f''''(t_i) = \frac{-f(t_i+4h) + 12f(t_i+2h) - 39f(t_i+h) + 56f(t_i) - 39f(t_i-h) + 12f(t_i-2h) - f(t_i-3h)}{6h^4}$$

Tendo o conhecimento da função $f(t)$, podemos ainda incluir a Extrapolação de Richardson para melhorar a acurácia da derivada. Basicamente, fazemos

$$D = \frac{4D(h_2) - D(h_1)}{3}, \tag{9.9}$$

sendo $h_2 = h_1/2$. Vejamos um exemplo numérico do uso da Extrapolação de Richardson. Seja a função $f(t) = \sqrt{t}\sin(\pi t)$, qual o valor da derivada desta função em $t = 2$? Solução usado o Scilab:

```
clc;
h1 = 0.5;
x = 0:h1:5;
y = sqrt(x).*sin(%pi*x);
d1 = (y(5+1) - y(5-1))/(2*h1); disp([x(5), y(5)]);

h2 = 0.25;
x = 0:h2:5;
y = sqrt(x).*sin(%pi*x);
d2 = (y(9+1) - y(9-1))/(2*h2); disp([x(9), y(9)]);

d = (4*d2-d1)/3;   // Extrapolação de Richardosn
dtt = sqrt(2)*%pi;  // valor teórico
disp([d1, d2, d, dtt]);

e1 = 100*abs(dtt-d1)/dtt; e2 = 100*abs(dtt-d2)/dtt;
er = 100*abs(dtt-d)/dtt;
disp('Erros (%):');
disp([e1, e2, d]);
```

Obtemos como resultados:

```
    2.8058837    3.992149    4.3875708    4.4428829
 Erros (%):
    36.845428    10.145077   1.2449602
```

Observamos que o erro diminui com o valor do passo e que a Extrapolação de Richardson melhora significativamente a estimativa da derivada.

9.3 Dados desigualmente espaçados

Até agora, todas as derivadas foram calculadas a partir de dados igualmente espaçados. Em algumas situações, podemos ter acesso apenas a dados tabelados que podem não estar igualmente espaçados. Uma solução é, a partir dos pontos conhecidos, calcularmos um polinômio interpolador que passa por estes pontos e, de posse desse polinômio, calcularmos as derivadas desejadas.

Por exemplo, vamos considerar os pontos tabelados desigualmente espaçados

```
  x      y
 1,3   -0,9224213
 1,8   -0,7885967
 2,1    0,4478082
 2,6    1,5335326
```

sendo desejado calcular a derivada em $f(x = 2,0)$. Podemos calcular um polinômio $p(x) = ax^3 + bx^2 + cx + d$ que passa por esses pontos usando o seguinte *script* Scilab:

```
x = [1.3, 1.8, 2.1, 2.6];
y  = [ - 0.9224213,  - 0.7885967,    0.4478082,    1.5335326];
M = [x(1)^3 x(1)^2 x(1) 1
        x(2)^3 x(2)^2 x(2) 1
        x(3)^3 x(3)^2 x(3) 1
        x(4)^3 x(4)^2 x(4) 1];
v = y';
cf = inv(M)*v
```

Obtemos os seguintes coeficientes e polinômio $p(x)$:

```
 cf  =
  - 5.5803859
    33.835132
  - 64.051855
    37.423725
```

Logo, $p(x) = -5,5803859x^3 + 33,835132x^2 - 64,051855x + 37,423725$. Assim, a derivada em $x = 2$ é, aproximadamente, igual a

$$\frac{dp}{dx}|_{x=2} = -5,5803859 \times 3 \times 4 + 33,835132 \times 2 \times 2 - 64,051855 \cong 4,3240429$$

Também podemos, já que temos o polinômio $p(x)$, fazer x variar em torno de 2 com um passo h pequeno e usarmos uma das expressões já conhecidas para o cálculo da derivada de $p(x)$.

9.4 Derivadas usando o Scilab

Basicamente, as derivadas numéricas com o Scilab são calculadas com o comando “diff”. O exemplo a seguir ilustra o seu uso (ver Figura 9.3). Notamos que ocorre um erro devido o uso de um passo h relativamente grande. O comando “diff” implementa a derivada por diferença progressiva simples: y=diff(x) calcula a função de diferença y=x(2:10)-x(1:9), para um vetor x com 10 elementos.

```
//diferenciação aproximada
h=0.1;
t=0:h:10;
y=sin(t);
t2 = 0:h:(10-h);
dy=diff(sin(t))/h; //diferenciação aproximada da função seno
dyt = cos(t);  //diferenciação teórica
plot(t,y,t2,dy,t,dyt);
title('Exemplo do uso do comando diff');
legend('f(t) = sin(t)','df = diff(f)/h','dft = cos(t)');
xgrid(3); xlabel('tempo');
```

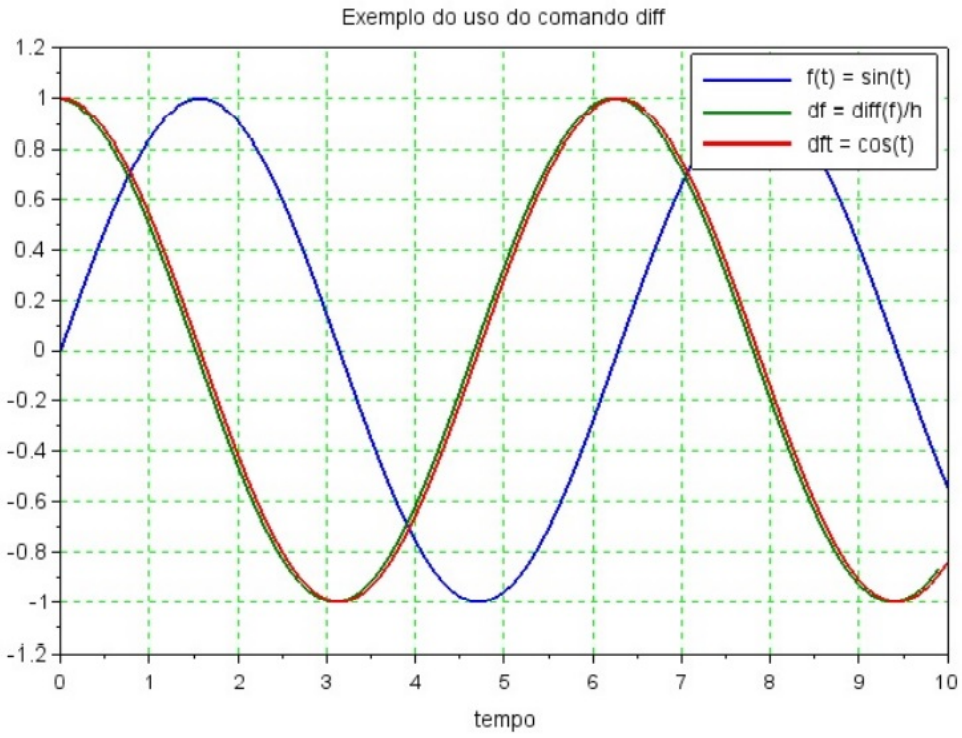


Figura 9.3: Uso comando “diff”.

9.5 Derivadas parciais

Algumas funções são dependentes de duas ou mais variáveis, como, por exemplo,

$$f(x,y) = x^2 + y^2$$

Podemos calcular a derivada dessas funções para cada uma dessas variáveis, esse cálculo é chamado de derivada parcial. Para função acima, a derivada parcial em relação a x é

$$\frac{\partial f(x,y)}{\partial x} = 2x$$

De forma semelhante ao cálculo numérico das derivadas simples, podemos calcular derivadas parciais pela variação da função em relação a uma das variáveis. Por exemplo:

$$\frac{\partial f(x,y)}{\partial x} \cong \frac{f(x+\Delta x,y) - f(x-\Delta x,y)}{2\Delta x} \tag{9.10}$$

$$\frac{\partial f(x,y)}{\partial y} \cong \frac{f(x,y+\Delta y) - f(x,y-\Delta y)}{2\Delta y} \tag{9.11}$$

Podemos calcular as derivadas parciais de ordem superior, isto é, derivadas de segunda ordem ou maior, pela derivação seguida em relação à variável de interesse. Por exemplo:

$$\frac{\partial^2 f(x,y)}{\partial x \partial y} = \frac{\partial}{\partial x}\left(\frac{\partial f(x,y)}{\partial x}\right)$$

Numericamente, teremos:

$$\frac{\partial^2 f(x,y)}{\partial x \partial y} \cong \frac{f(x+\Delta x,y+\Delta y) - f(x+\Delta x,y-\Delta y) - f(x-\Delta x,y+\Delta y) + +f(x-\Delta x,y-\Delta y)}{4\Delta x \Delta y} \tag{9.12}$$

9.6 Problemas

Questão 1. Calcule a derivada das funções

$a.\ g(x) = x^3 + e^{x/2}$ $\quad$ $b.\ g(x) = x^{-3} + e^{-x/2}$

$c.\ g(x) = \cos(x)e^{x/2}$ $\quad$ $d.\ g(x) = \ln(x+1)$

$e.\ g(x) = \sin(x)e^{-x/2}$ $\quad$ $f.\ g(x) = \dfrac{x+1}{x^2-1}$

$g.\ g(x) = \cos(2x)e^{-x}$ $\quad$ $h.\ g(x) = \dfrac{x^2-1}{x^2+1}$

$i.\ g(x) = e^{-x} - \sqrt{x}$ $\quad$ $j.\ g(x) = \dfrac{e^x}{x^2+x+1}$

$k.\ g(x) = \sqrt{x}e^{-x/3}$ $\quad$ $l.\ g(x) = \ln(x^2+3x+2)$

$m.\ g(x) = x\sqrt{x+1}$ $\quad$ $n.\ g(x) = \dfrac{x}{\sqrt{x+2}}$

$o.\ g(x) = x^2\sqrt{x+2}$ $\quad$ $p.\ g(x) = \dfrac{x+2}{\sqrt{x+4}}$

em $x_0 = 1$ usando diferença progressiva, diferença regressiva e diferença centrada com um $h = 0,1$. Qual apresenta o menor erro absoluto?

Questão 2. Refaça a questão anterior, mas usando as fórmulas de maior acurácia de derivadas centradas com 4 pontos.

Questão 3. Calcule a derivada da função $f(x) = x^3 - 3x - 1$ e gere a função $g(x)$ no intervalo $(0,2)$ com um passo $h = 0,05$ usando o comando "diff".

Questão 4. Refaça a questão anterior usando diferença centrada. Qual solução apresenta menor erro?

Questão 5. Considere a função $f(t) = \cos(t^2)$, calcule sua derivada em $t = 2$ usando diferença progressiva, diferença regressiva e diferença centrada com um $h = 0,01$.

Questão 6. Encontre, aproximadamente, o valor máximo da função $g(t) = te^{-t}\cos(2t)$. Lembre-se que o valor máximo (ou mínimo) da função ocorre quando a derivada é zero.

Questão 7. Sabemos que a derivada de uma função exponencial é a própria função exponencial: $f(x) = e^{kx} \Leftrightarrow f'(x) = ke^{kx}$. Calcule numericamente as derivadas primeira, segunda e terceira da função $f(x) = e^x$, em $x_0 = 0$, com $h = 0,001$. Compare com os resultados teóricos.

Questão 8. Para a função

$$f(x) = \frac{e^x + 1}{e^x - 1}$$

qual o valor ideal de h para o cálculo da derivada usando diferença progressiva simples? E se for usada derivada centrada com 4 pontos?

Questão 9. Considere a função $f(x) = e^{-x}\,\mathrm{sen}(2x)$ para $x \geq 0$ (ver Figura 9.4. Qual a derivada em $x = 1$? Escreva um código Scilab para desenhar a função $f'(x)$ usando o comando "diff".

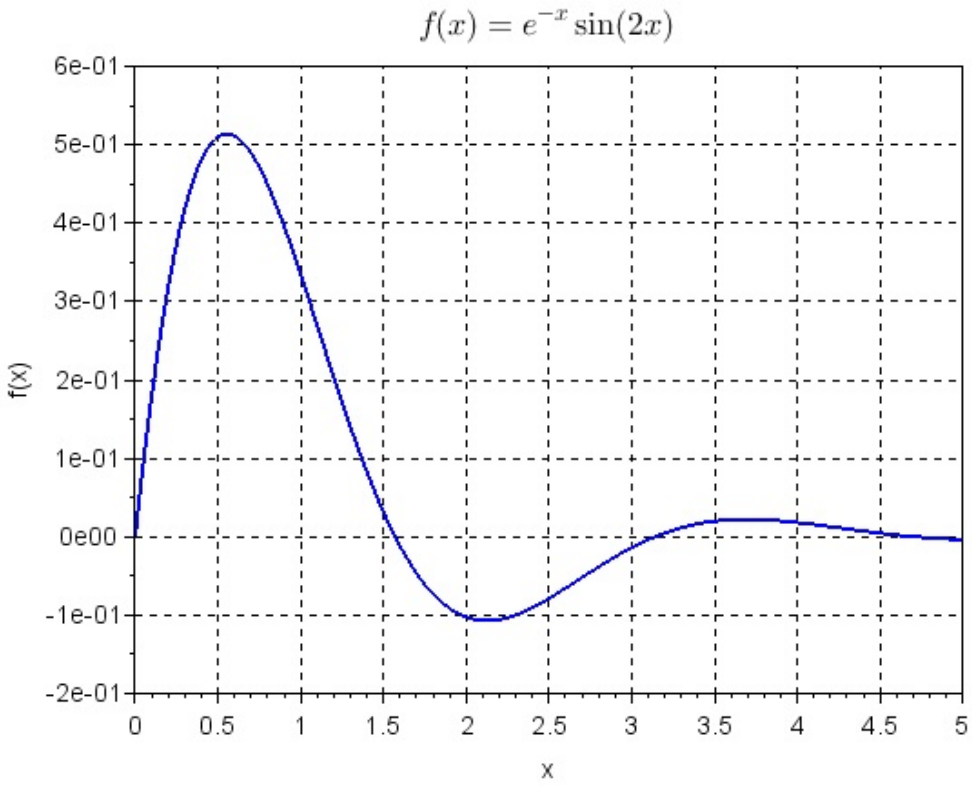


Figura 9.4: Função $f(x) = e^{-x}\,\mathrm{sen}(2x)$

Questão 10. Dados os pontos tabelados abaixo, encontre a derivada da função em $x = 2,00$.

Tabela 9.2: Pontos não igualmente espaçados.

x_i	$f(x_i)$
0,50	1,207
0,70	1,537
1,00	2,000
1,20	2,295
1,70	3,004
2,10	3,549
2,30	3,817
2,40	3,949

Questão 11. Dados os pontos tabelados abaixo, encontre a derivada da função em $x = 1,10$.

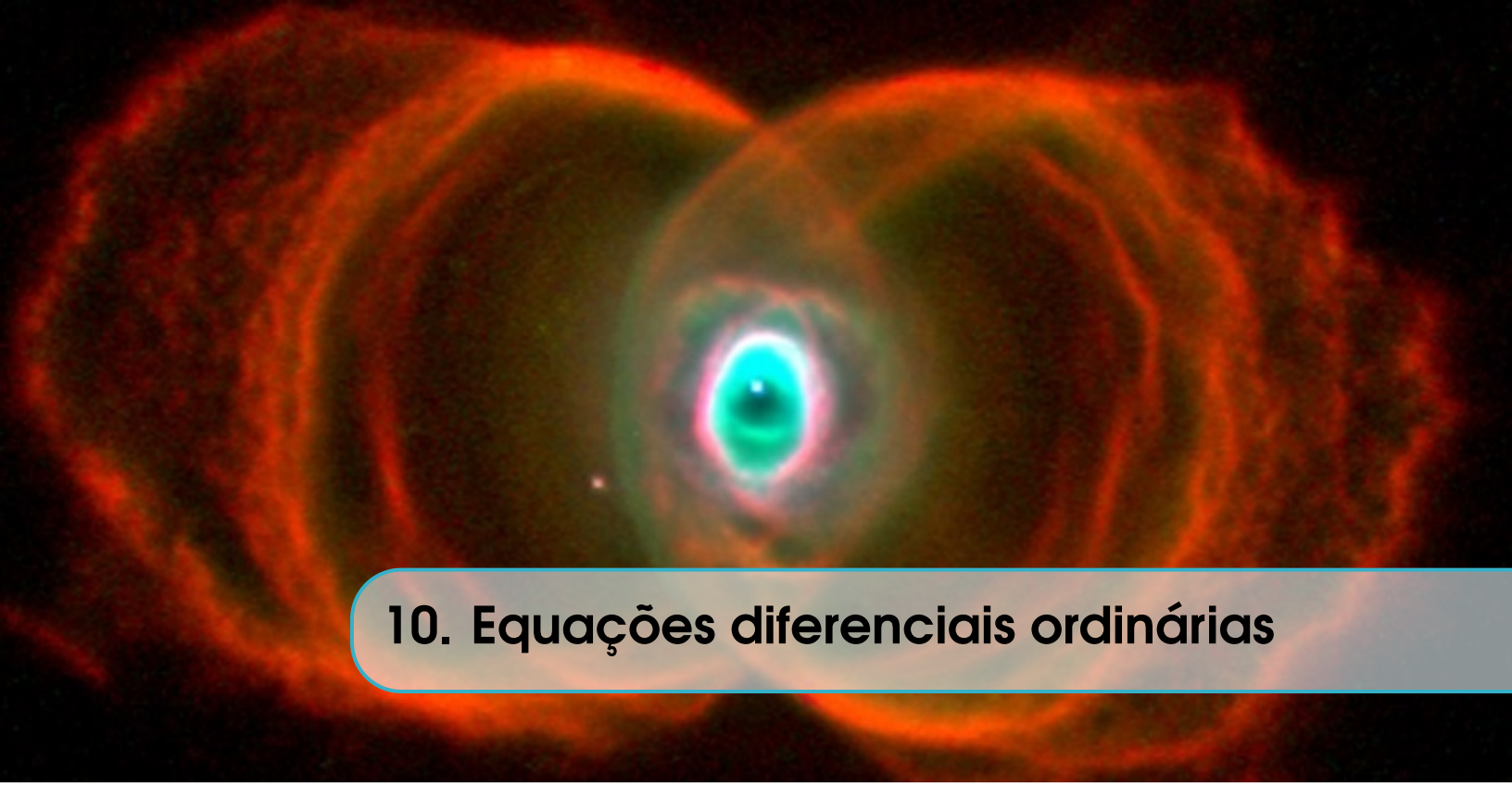
Tabela 9.3: Pontos não igualmente espaçados.

x_i	$f(x_i)$
0,0	0,0000000 0,2
0,3188288	
0,5	0,5103780
0,9	0,3959369
1,0	0,3345118
1,2	0,2034456
1,4	0,0826071
1,5	0,0314881

Questão 12. Considere a função $f(x,y) = x^3 + 3xy + y^2$. Calcule $\partial f/\partial x$ e $\partial^2 f/(\partial x \partial y)$ analiticamente e numericamente com $x = 3$, $y = 2$ e $\Delta x = \Delta y = 0,001$. Compare os resultados.

Questão 13. Considere a função $f(x,y) = x^3 + 3x^2y + 4xy^2 + y^3$. Calcule $\partial f/\partial x$ e $\partial^2 f/(\partial x \partial y)$ analiticamente e numericamente com $x = 3$, $y = 2$ e $\Delta x = \Delta y = 0,001$. Compare os resultados.

Questão 14. Refaça a questão anterior para $f(x,y) = \cos(x)e^{-y^2}$ com $x = 1$, $y = 1$ e $\Delta x = \Delta y = 0,001$.



10. Equações diferenciais ordinárias

Não existem métodos fáceis para resolver problemas difíceis.
René Descartes (1596 - 1650), filósofo, físico e matemático francês.

MUITOS problemas de engenharia, economia e das ciências em geral nos levam ter que resolver uma equação diferencial ordinária (EDO) ou uma equação diferencial parcial (EDP). Uma equação diferencial é dita ordinária se a função incógnita desconhecida depende de uma única variável independente, em geral, o tempo. Já em uma EDP, uma função incógnita depende de duas ou mais variáveis independentes, aparecendo então derivadas parciais. Essas equações, especialmente as que são não lineares, dificilmente possuem uma solução analítica, precisamos nos contentar com soluções numéricas aproximadas. Neste capítulo, o foco principal será na solução numérica de EDOs, começando com os métodos mais simples.

10.1 Método de Euler

Vamos considerar inicialmente a equação

$$\frac{dy}{dt} = 1 - e^{-t} \tag{10.1}$$

Se usarmos a aproximação $dy/dt = (y(t+\Delta t) - y(t))/\Delta t$, então, podemos reescrever (10.1) como:

$$y(t+\Delta t) = y(t) + \Delta t(1 - e^{-t}) \tag{10.2}$$

Podemos resolver numericamente a expressão (10.2) desde que tenhamos um ponto inicial, por exemplo, $y(0) = 1$. Uma EDO de primeira ordem precisa de uma condição inicial. Usando um passo

$\Delta t = 0,1$, podemos escrever o seguinte algoritmo:

(1) $t_0 = 0$
(2) $\Delta t = 0,1$
(3) $y_0 = 1$
(4) $y_{k+1} = y_k + \Delta t(1 - e^{-t_k})$
(5) $t_{k+1} = t_k + \Delta t$
(6)Se $t_{k+1} < t_{max}$ voltar ao passo (4)

Em essência, no algoritmo acima estamos usando o o Método de Euler. Para o exemplo acima, a solução analítica é $y(t) = t + e^{-t}$. Executando o algoritmo e calculando os valores teóricos, obtemos a Tabela 10.1. Percebemos um erro crescente nos valores numéricos fornecidos. Podemos reduzir esse erro diminuindo o valor do passo a custa de um maior esforço computacional. Por outro lado, se o valor do passo for excessivamente grande, o algoritmo acima pode se tornar instável.

Tabela 10.1: Solução da equação $\frac{dy}{dt} = 1 - e^{-t}$ usando o Método de Euler

t	$y_n(t)$	$y_t(t)$	$erro_r$ (%)
0,000	1,000000	1,000000	0,00
0,100	1,009516	1,004837	0,47
0,200	1,027643	1,018731	0,87
0,300	1,053561	1,040818	1,22
0,400	1,086529	1,070320	1,51
0,500	1,125876	1,106531	1,75
0,600	1,170995	1,148812	1,93
0,700	1,221337	1,196585	2,07
0,800	1,276404	1,249329	2,17
0,900	1,335747	1,306570	2,23
1,000	1,398959	1,367879	2,27

Sendo um pouco mais geral, se a equação for da forma

$$\frac{dy}{dt} = f(t, y(t)) \tag{10.3}$$

com $y(t_0)$ conhecido e passo h, então a solução numérica pelo Método de Euler é

$$y(t_{k+1}) = y(t_k) + hf(t_k, y(t_k)) \tag{10.4}$$
$$t_{k+1} = t_k + h \tag{10.5}$$

O erro de truncamento é da ordem de[1]

$$\varepsilon_t = \frac{h^2}{2} y''(\xi)$$

[1]Usaremos sempre que conveniente a notação y' para denotar a derivada dy/dt e y'' para indicar derivada a segunda d^2y/dt^2. Da mesma forma, usaremos Δt e h para indicar um passo no tempo ou na variável independente.

onde $\xi \in (t_k, t_{k+1})$. Vejamos um segundo exemplo: $y' = -y - 2\cos(2t)e^{-t}$, com a condição inicial $y(0) = 1$, $t_{max} = 1s$ e passo $h = 0,1$. A solução analítica é $y(t) = sen(2t)e^{-t}$. Código Scilab (resultados na Tabela 10.2):

```
clc;
h = 0.1; yn = 1; yt = 1; t = 0;
mprintf('Tempo, yn, yt \n');
mprintf('%1.3f & %f & %f \n',t,yn,yt);
for k=1:10
    t = t + h;
    yn = yn - h*(yn + 2*sin(2*t)*exp(-t));
    yt = cos(2*t)*exp(-t);
    mprintf('%1.3f & %f & %f  \n',t,yn,yt);
end
```

Tabela 10.2: Solução da equação $y' = -y - 2\cos(2t)e^{-t}$ usando o Método de Euler

t	$y_n(t)$	$y_t(t)$
0,000	1,000000	1,000000
0,100	0,900000	0,886801
0,200	0,774047	0,754101
0,300	0,632877	0,611424
0,400	0,485930	0,467016
0,500	0,341165	0,327710
0,600	0,204973	0,198866
0,700	0,082173	0,084403
0,800	-0,023916	-0,013120
0,900	-0,111352	-0,092373
1,000	-0,179404	-0,153092

Observamos mais uma vez que o erro se agrava a cada iteração, mas diminui no final pois $y(t \to \infty) = 0$, como podemos ver na Figura 10.1.

10.2 Método de Euler melhorado ou de Heun

Podemos melhorar consideravelmente o Método de Euler se fizermos duas estimativas a cada iteração. Assim, o algoritmo básico é:

$$K_1 = y(t_k) + hf(t_k, y(t_k)) \tag{10.6}$$

$$y(t_{k+1}) = y(t_k) + \frac{h}{2}\left(f(t_k, y(t_k)) + f(t_k + h, K_1)\right) \tag{10.7}$$

$$t_{k+1} = t_k + h \tag{10.8}$$

Podemos dizer que K_1 prediz uma estimativa e que essa estimativa é corrigida em seguida. Aplicando o Método de Euler melhorado ou método de Heun para a equação $y' = -y - 2\cos(2t)e^{-t}$, obtemos os valores da Tabela 10.3

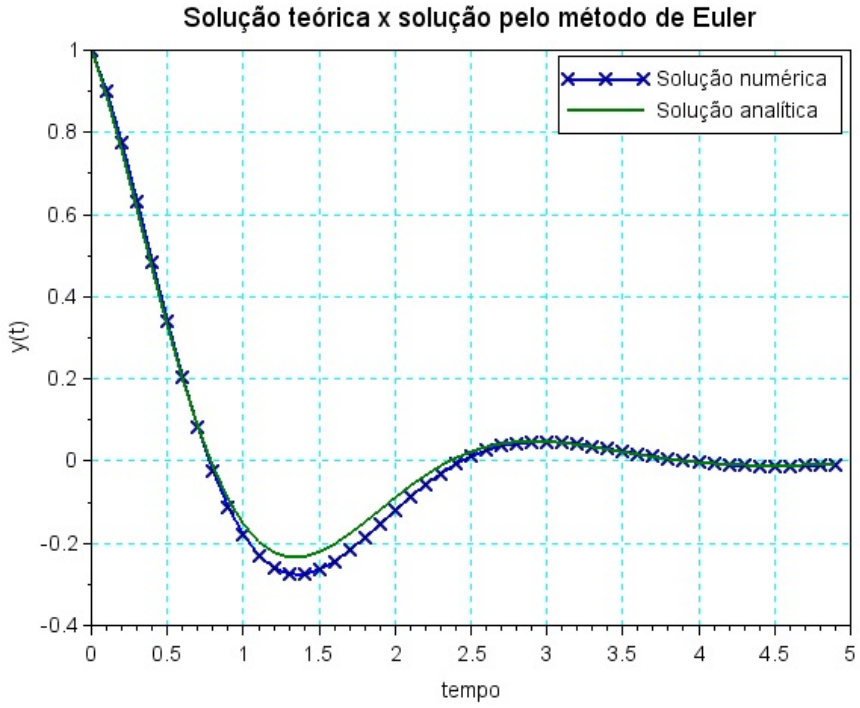


Figura 10.1: Uso do método de Euler - comparação entre solução numérica e analítica.

Tabela 10.3: Solução da equação $y' = -y - 2\cos(2t)e^{-t}$
usando o Método de Euler Melhorado

t	$y_{n2}(t)$	$y_t(t)$
0,000	1,000000	1,000000
0,100	0,887024	0,886801
0,200	0,754695	0,754101
0,300	0,612474	0,611424
0,400	0,468557	0,467016
0,500	0,329729	0,327710
0,600	0,201319	0,198866
0,700	0,087222	0,084403
0,800	-0,010021	-0,013120
0,900	-0,089085	-0,092373
1,000	-0,149707	-0,153092

Os resultados são bem melhores que os obtidos com o Método de Euler simples, ver também a Figura 10.2. O Código Scilab é:

```
////////// Euler Melhorado:
// y' = -y - 2cos(2t)e^{-t}
// y(0) = 1 e passo h = 0,1
function r = fty(tt,yy)
    r = -yy -2*sin(2*tt).*exp(-tt);
endfunction

h = 0.1; yn = 1; yt = 1; t = 0; yn2 = 1;
mprintf('Tempo, yn, yt \n');
mprintf('%1.3f & %f & %f \n',t,yn,yt);

for k=1:10
    k1 = yn2 + h*fty(t,yn2);
    yn2 = yn2 + h*(fty(t,yn2) + fty(t+h,k1))/2;
    t = t + h;
    yt = cos(2*t)*exp(-t);
    mprintf('%1.3f & %f & %f \n',t,yn2,yt);
end
```

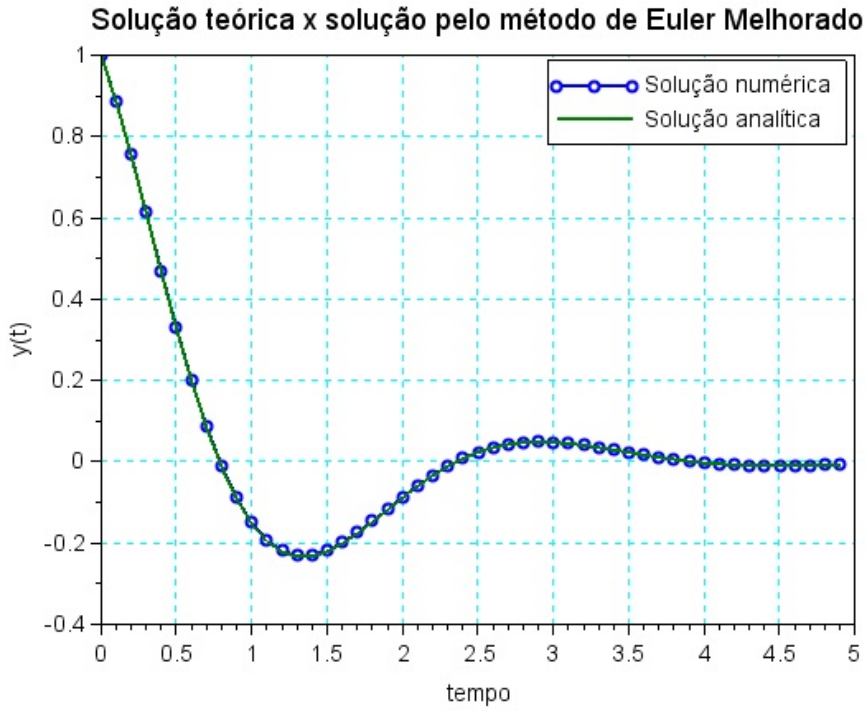


Figura 10.2: Uso do método de Euler melhorado (Heun) - comparação entre solução numérica e analítica.

10.3 Métodos de Runge-Kutta

Lembramos que a série de Taylor pode ser escrita como:

$$y(t_k+h) = y(t_k) + hy'(t_k) + \frac{h^2}{2}y''(t_k) + \frac{h^3}{3!}y'''(t_k) + \dots$$

então, podemos escrever a solução da EDO $y' = f(t,y)$ como sendo

$$y(t_{k+1}) = y(t_k) + hf(t_k,y_k) + \frac{h^2}{2}f'(t_k,y_k) \tag{10.9}$$

se a derivada $f'(t,y)$ for fácil de calcular. Esse método pode resultar em uma solução comparável ao do método de Heun. Os métodos de Runge-Kutta[2] (RK) tentam obter a da série de Taylor sem que sejam calculadas as derivadas de ordem superior.

10.3.1 Runge-Kutta de terceira ordem

Podemos descrever o método RK-3 por:

$$\begin{aligned}
K_1 &= hf(t_k,y_k)\\
K_2 &= hf(t_k+h/2, y_k+K_1/2)\\
K_3 &= hf(t_k+h, y_k+K_2)\\
y_{k+1} &= y_k + \frac{1}{4}(K_1+2K_2+K_3)\\
t_{k+1} &= t_k+h
\end{aligned}$$

Este método apresenta um erro de truncamento bem inferior aos métodos de segunda ordem.

10.3.2 Runge-Kutta de quarta ordem

Este é método RK mais utilizado, pois apresenta uma excelente relação custo-benefício. Podemos descrever o método RK-4 por:

$$\begin{aligned}
K_1 &= hf(t_k,y_k)\\
K_2 &= hf(t_k+h/2, y_k+K_1/2)\\
K_3 &= hf(t_k+h/2, y_k+K_2/2)\\
K_4 &= hf(t_k+h, y_k+K_3)\\
y_{k+1} &= y_k + \frac{1}{6}(K_1+2K_2+2K_3+K_4)\\
t_{k+1} &= t_k+h
\end{aligned}$$

Este método apresenta um erro de truncamento inferior ao RK-3.

[2]Carl David Tolmé Runge - Nascido em 30 de agosto de 1856 em Bremen, Alemanha. Morreu: 3 de janeiro de 1927 em Göttingen, Alemanha. Professor de matemática aplicada na Universidade de Göttingen. Também tem trabalhos importantes na Física na área de linhas espectrais de emissão de elementos afetados por um campo magnético.
Martin Wilhelm Kutta - Nascido: 3 de novembro de 1867 em Pitschen, Alta Silésia (agora Byczyna, Polônia). Morreu: 25 de dezembro de 1944 em Fürstenfeldbruck, na Alemanha. Além da matemática, Kutta tem contribuição importante na área de aerodinâmica.

10.3.3 Comparação entre os métodos RK

Para comparar os métodos vistos até agora, vamos considerar a EDO

$$\frac{dy}{dt} = \frac{2y}{t} - y$$

com a condição inicial $t_0 = 0,1s$, $y_0 = 0,009048$, cuja solução é $y(t) = t^2 e^{-t}$ para $t > 0,1s$. Os resultados estão na Tabela 10.4. Na Figura 10.3, temos uma visão geral da solução analítica e as soluções apresentadas pelos métodos numéricos. Sem dúvida, o método RK-4 apresentou melhor desempenho.

Tabela 10.4: Solução da equação $y' = 2y/t - y$: comparação dos métodos Runge-Kutta

t	$y_t(t)$	$Euler$	$RK2$	$RK3$	$RK4$
0,100	0,009048	0,009048	0,009048	0,009048	0,009048
0,200	0,032749	0,026240	0,029452	0,031159	0,032405
0,300	0,066674	0,049857	0,058561	0,062878	0,065917
0,400	0,107251	0,078109	0,093503	0,100898	0,106021
0,500	0,151633	0,109352	0,131839	0,142536	0,149889
0,600	0,197572	0,142158	0,171611	0,185668	0,195299
0,700	0,243327	0,175328	0,211285	0,228649	0,240526
0,800	0,287571	0,207889	0,249694	0,270224	0,284261
0,900	0,329321	0,239072	0,285969	0,309466	0,325531
1,000	0,367879	0,268292	0,319491	0,345713	0,363645

Chegamos aos resultados da Tabela 10.4 com o Código Scilab que segue:

```
clc; close;
//// y' = f(t,y)
function f = fty(t,y)
    f = 2*y/t - y;
endfunction
dt=0.1;
tt=dt:dt:8;
yt = tt.*tt.*exp(-tt);yt = tt.*tt.*exp(-tt);
t = dt; y1 = yt(1); y2 = y1;  y3 = y2; y4 = y2; dt = 0.1;
yk1 = zeros(1,11) ; yk2 = yk1; yk3 = yk1; yk4 = yk1;

for k=1:max(size(tt))
    yk1(k) = y1;
    yk2(k) = y2;
    yk3(k) = y3;
    yk4(k) = y4;

    y1 = y1 + dt*fty(t,y1); // Euler

    k1 = fty(t,y2);        // RK-2
```

```
    k2 = fty(t+dt,y2+k1*dt);
    y2 = y2 + dt*(k1+k2)/2;

    k1 = dt*fty(t,y3);    // RK-3
    k2 = dt*fty(t+dt/2,y3+k1/2);
    k3 = dt*fty(t+dt,y3+k2);
    y3 = y3 + (k1+2*k2 + k3)/4;

    k1 = dt*fty(t,y4);    // RK-4
    k2 = dt*fty(t+dt/2,y4+k1/2);
    k3 = dt*fty(t+dt/2,y4+k2/2);
    k4 = dt*fty(t+dt,y4+k3);
    y4 = y4 + (k1+2*k2 + 2*k3 + k4)/6;

     t = t + dt;   // atualizando o tempo
end;
for k=1:max(size(tt))
    mprintf('%1.3f & %f & %f & %f & %f & %f \n',...
tt(k),yt(k),yk1(k),yk2(k),yk3(k),yk4(k));
end;
close; plot(tt,yt,tt,yk1,tt,yk2,tt,yk3,tt,yk4);
legend('Curva teórica','Euler','RK-2','RK-3','RK-4');
xgrid(7); xlabel('$tempo$'); ylabel('$y(t)$');
```

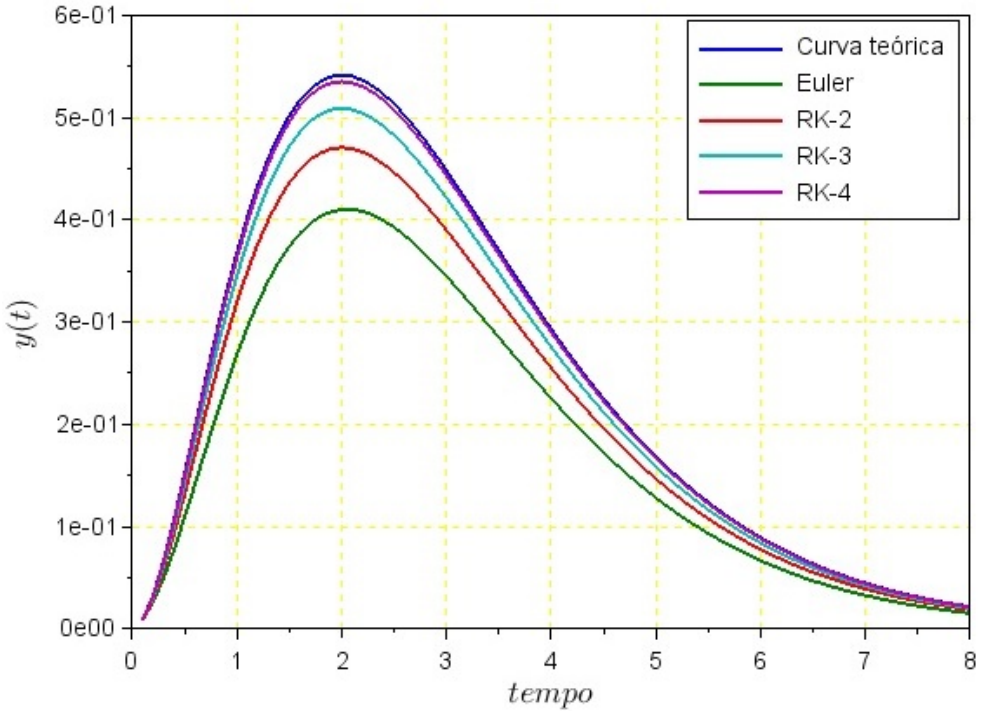


Figura 10.3: Comparação entre os métodos Runge-Kutta, equação: $y' = 2y/t - y$.

Uma observação final sobre os métodos RK: existem métodos RK explícitos de 5^a ordem ou mais alta, mas eles são pouco práticos para o uso rotineiro. Para problemas que requerem alta precisão, esses métodos de alta ordem podem ser necessários. Por exemplo, se queremos calcular a órbita dos planetas do Sistema Solar daqui um milhão de anos, teremos que usar métodos de altíssima acurácia. Para saber mais sobre esse tópico ver, por exemplo, Butcher (1964), Luther (1968), Hairer (1978), Darmand e Prince (1980), Shampine (1986). No Apêndice, apresentamos dois métodos RK de ordem 5.

10.4 EDOs de 2a. ordem

Até agora, vimos a solução de uma EDO de primeira ordem, mas muitos problemas exigem a solução de uma EDO de segunda ordem ou de ordem superior. Nestes casos, são necessárias duas (EDO de 2a. ordem) ou mais condições iniciais. Podemos usar a seguinte estratégia: "quebrar" a EDO em EDOs de primeira ordem. Assim, por exemplo, podemos escrever

$$\frac{d^2y}{dt^2} + g(t,y)\frac{dy}{dt} = f(t,y)$$

como

$$\frac{dy}{dt} = z(t,y)$$
$$\frac{dz}{dt} = -g(t,y)z(t,y) + f(t,y)$$

e, em seguida, aplicar um dos métodos estudados. Vejamos um exemplo mais detalhado. Considere a equação diferencial

$$\frac{d^2y}{dt^2} + 5\frac{dy}{dt} = -6y + \cos(3t)$$

com as condições iniciais $y(0) = 0$ e $y'(0) = 1$, $g(t,y) = 5$ e $f(t,y) = -6y + \cos(3t)$. Usando um passo $h = 0,05$, podemos equacionar (usando o método de Euler melhorado) como:

$$\begin{aligned}
z' &= -5z - 6y + \cos(3t)\\
y' &= z, \text{ logo:}\\
K_1 &= hz_k\\
L_1 &= h(-5z_k - 6y_k + \cos(3t_k))\\
K_2 &= h(z_k + K_1)\\
L_2 &= h(-5(z_k + L_1) - 6(y_k + K_1) + \cos(3(t_k + h)))\\
z_{k+1} &= z_k + h(L_1 + L_2)/2\\
y_{k+1} &= y_k + (K_1 + K_2)/2\\
t_{k+1} &= t_k + h
\end{aligned}$$

Note que as expressões de cálculo de z_k e y_k estão entrelaçadas, isso proporciona um menor erro numérico. A solução teórica dessa EDO é

$$y(t) = \frac{1}{78}(5\,\mathrm{sen}(3t) - \cos(3t)) + \frac{11}{13}e^{-2t} - \frac{5}{6}5e^{-3t}$$

Na Figura 10.4, temos uma comparação entre a solução numérica e a solução analítica.

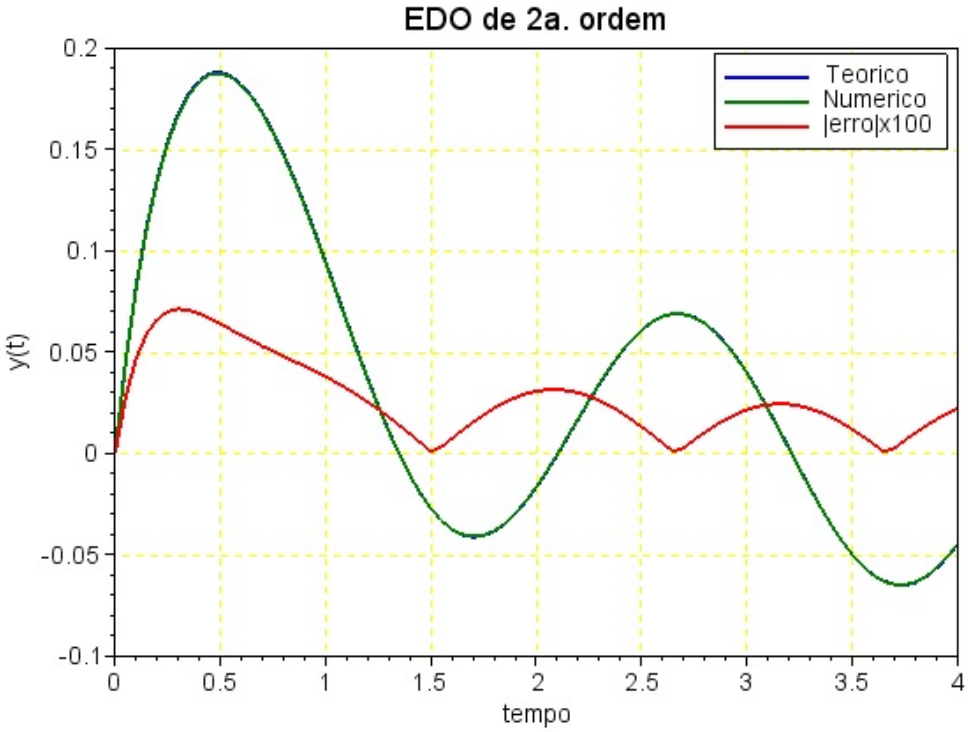


Figura 10.4: Solução numérica de uma EDO de 2a. ordem

10.5 Métodos de predição/correção

Até agora, estudamos métodos de passo simples para resolver uma EDO. Existem, entretanto, técnicas que usam várias estimativas da função e formas de corrigir os valores de y_k. O Método de Adams-Bashforth/Adams-Moulton de passo 3 é um exemplo de método preditor-corretor, ele pode ser expresso por (BURDEN & FAIRES, 2008):

$$
\begin{aligned}
f_1 &= f(t_{k-3}, y_{k-3}) \\
f_2 &= f(t_{k-2}, y_{k-2}) \\
f_3 &= f(t_{k-1}, y_{k-1}) \\
y_k^* &= y_{k-1} + h\left(23f_3 - 16f_2 + 5f_1\right)/12 \text{ (predição)} \\
f_4 &= f(t_k, y_k^*) \\
y_k &= y_{k-1} + h\left(9f_4 + 19f_3 - 5f_2 + f_1\right)/24 \text{ (valor corrigido)}
\end{aligned}
$$

E o de passo 4:

$$
\begin{aligned}
f_0 &= f(t_{k-4}, y_{k-4}) \\
f_1 &= f(t_{k-3}, y_{k-3}) \\
f_2 &= f(t_{k-2}, y_{k-2}) \\
f_3 &= f(t_{k-1}, y_{k-1}) \\
y_k^* &= y_{k-1} + h\left(55f_3 - 59f_2 + 37f_1 - 9f_0\right)/24 \text{ (predição)} \\
f_4 &= f(t_k, y_k^*) \\
y_k &= y_{k-1} + h\left(251f_4 + 646f_3 - 264f_2 + 106f_1 - 19f_0\right)/720 \text{ (valor corrigido)}
\end{aligned}
$$

Um problema desse método é a exigência de vários valores iniciais. Para uma EDO de primeira ordem, nós, normalmente, temos apenas o conhecimento de y_0, os valores de y_1, y_2 e y_3 devem ser, em geral, calculados por outro método, normalmente o RK-4. Vejamos um exemplo: dada a EDO $y' = y/2 + 3t^{1/2}e^{t/2}$ com $y(0) = 0$ e um passo $h = 0,20$, calcular $y(t)$ para $t \leq 1,8$. Os valores $y(0,2)$, $y(0,4)$ e $y(0,6)$ foram calculados pelo método RK-4. O código Scilab para o algoritmo de Adams-Bashforth/Adams-Moulton de passo 4 está indicado abaixo. Na Tabela 10.5, temos os resultados numéricos.

```
function f=ff(t,y)
    f = y/2 + 3*exp(t/2).*sqrt(t);
endfunction

h = 0.2; vy = 0 + zeros(1,10);
ya = vy; yt = ya;
t = 0; vt = 0*vy; ym = ya; y = vy(1);
//// Valores iniciais:
for k=2:10
    k1 = h*ff(t,y);
    k2 = h*ff(t+h/2,y+k1/2);
    k3 = h*ff(t+h/2,y+k2/2);
    k4 = h*ff(t+h,y+k3);
    y = y + (k1+2*k2+2*k3+k4)/6;
    vy(k) = y;
    t = t + h;
    yt(k) = 2*exp(t/2)*(t^(3/2));
    ya(k) = y;  vt(k) = t;
    disp([t,y]);
end

for k=5:10   //Adams:
    y0 = ff(vt(k-4),ya(k-4));
    y1 = ff(vt(k-3),ya(k-3));
    y2 = ff(vt(k-2),ya(k-2));
    y3 = ff(vt(k-1),ya(k-1));
    ya(k) = ya(k-1) + h*(55*y3-59*y2+37*y1-9*y0)/24;
    y4 = ff(vt(k),ya(k));
```

```
    ya(k) = ya(k-1) + h*(251*y4+646*y3-264*y2+106*y1-19*y0)/720;
end;
disp('Teórico, RK-4, Adams');
disp([yt', vy', ya']);
```

Tabela 10.5: Solução da equação $y' = y/2 + 3t^{1/2}e^{t/2}$: comparação dos métodos Runge-Kutta e Adams

t	$y(t)$	$y(t) - RK4$	$y(t) - Adams$
0,00	0,000000	0,000000	0,000000
0,20	0,197699	0,189216	0,189216
0,40	0,617986	0,608582	0,608582
0,60	1,254715	1,244315	1,244315
0,80	2,134926	2,123428	2,127574
1,00	3,297443	3,284733	3,289605
1,20	4,790475	4,776426	4,781795
1,40	6,671572	6,656042	6,661996
1,60	9,008356	8,991189	8,997769
1,80	11,879654	11,860677	11,867943

10.6 Um comentário sobre o passo h

A solução numérica de uma EDO requer o uso de um passo de integração h. Se esse passo for muito grande, os erros podem ser intoleráveis ou levar a uma instabilidade da solução; se o passo for muito pequeno, o custo computacional pode ser muito elevado e tempo gasto também. Além disso, uma redução do passo não leva necessariamente a um erro final menor. Logo, a escolha desse passo é uma tarefa não trivial. Existem métodos para fazer um ajuste automático do passo (solução de passo variável) que conseguem uma boa aproximação numérica dentro de um tempo de cálculo razoável.

Evidenciaremos, através de um exemplo simples, que a escolha adequada do passo pode reduzir o erro entre a solução numérica e a solução analítica a um valor mínimo. Isso ocorre porque o erro é composto por duas parcelas: erros de truncamento (devido à fórmula usada na solução da equação diferencial) e erros de arredondamento (devido o número finito de *bits* do computador). Quanto menor o valor do passo, menor o erro de truncamento, mas os erros de arredondamento vão se acumulando. Assim, existe um ponto (passo ótimo) onde essa soma é mínima (ver Figura 10.5). O passo ótimo também depende do método usado para solucionar a EDO.

Analisemos a EDO $y' = 2te^{-t} - y$. Com $y(0) = 0$ a solução analítica é $y(t) = t^2e^{-t}$. A solução analítica será usada para calcular os erros das soluções numéricas. Para solução numérica e cálculo dos erros, usamos os métodos RK-2 e RK-4. O valor para que o erro seja mínimo é diferente para RK-2 ($h_{oRK2} \approx 5 \times 10^{-6}$) e para o RK-4 ($h_{oRK4} \approx 10^{-3}$), como podemos ver na Figura 10.6. Assim, mesmo efetuando mais cálculos por iteração, o RK-4 pode ser mais rápido, pois pode usar um passo muito maior para a mesma precisão.

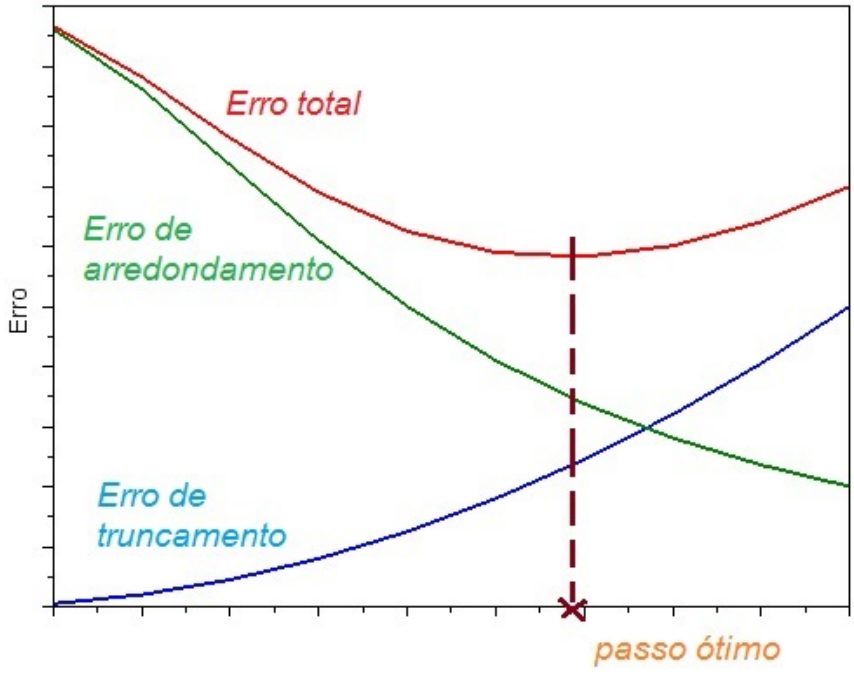


Figura 10.5: Escolha do passo h: buscando o valor ótimo.

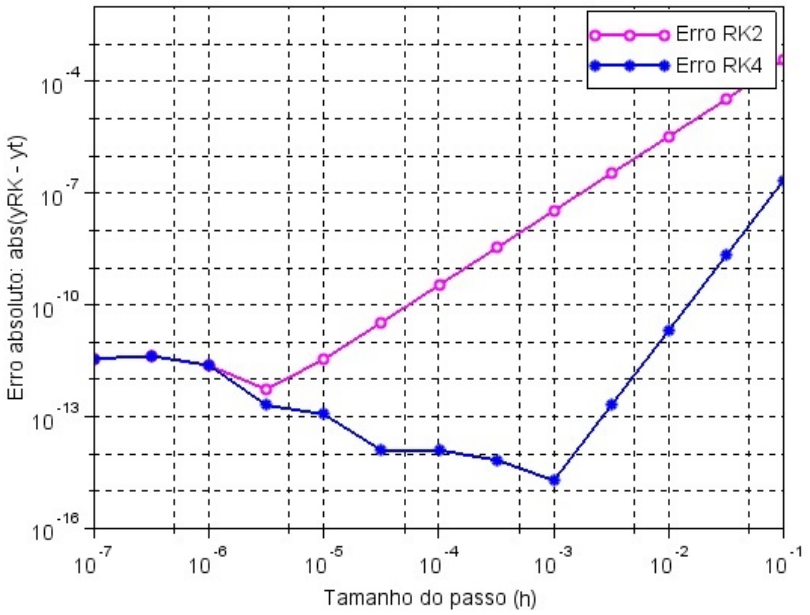


Figura 10.6: Escolha do passo h: buscando o valor ótimo.

10.7 Solução usando o comando "ode"

O comando "ode" é um solucionador de equações diferenciais ordinárias. A sua chamada mais básica é: "y=ode(y0,t0,t,f)", onde y0 é a condição inicial em t0, t é um vetor de reais (tempos nos quais a solução é computada) e f é uma função externa (função, lista ou *string*). Também podemos incluir qual método (adams, stiff, rk, rkf,) de solução será usado para o cálculo de "y", se este parâmetro não for passado, então, o solucionador *lsoda* do pacote **ODEPACK** é chamado por padrão. Ele escolhe automaticamente entre o método preditor-corretor não-rígido de Adams e a Fórmula de Diferenciação Retroativa (FDR) rígida. Ele utiliza o método não rígido, inicialmente, e monitora os dados para decidir qual método utilizar.

Vejamos um exemplo. Consideremos a EDO $y' = -y - 4e^{-t}\sin(4t)$, com $y(0) = 1$, cuja solução é $y(t) = e^{-t}\cos(4t)$. Após a solução usando o comando "ode", ver *script* Scilab abaixo, obtemos a Figura 10.7. Observamos que o erro é muito pequeno, da ordem de 10^{-11}. Neste caso, claramente, poderíamos ter aumentando o passo de $0,05$ para algo maior.

```
function f=ff(t,y)
    f = -y - 4*exp(-t)*sin(4*t);
endfunction

t=0:0.05:4;  y=exp(-t).*cos(4*t); // solução teórica

y0=1; t0=0;
yo=ode('rk',y0,t0,t,ff);  // chamada da função ''ode''
erro2b = (y-yo);
ge = round(log10(0.1/max(erro2b)));
erro2b = erro2b*(10^ge);
plot(t,y,t,yo,'r',t,erro2b,'g');
title('Solução usando o comando ode do Scilab');
xlabel('tempo'); ylabel('y(t)');
legend('Teórico','Numérico','|erro|x10^'+string(ge)); xgrid(0);
```

10.8 Aplicação a circuitos elétricos

Em circuitos elétricos que têm um ou mais elementos reativos (capacitores ou indutores), o cálculo de correntes e tensões necessariamente nos leva a uma equação diferencial ordinária, que relaciona a variável de entrada com a variável de saída. O primeiro problema é encontrar essa EDO usando alguma técnica de circuitos (Lei das Malhas ou Lei dos Nós, por exemplo). Vencida essa etapa, teremos a EDO desejada que poderemos solucionar de forma analítica ou numérica.

Vejamos o exemplo a seguir (ver Figura 10.8[3]). Iremos considerar que o capacitor e o indutor estão sem energia para $t < 0$ e que a chave ideal fecha em $t = 0s$. Queremos calcular a tensão de

[3]Exemplo tirado do meu *blog*. Outros exemplos de simulação também estão disponíveis. *Link*: `http://alfanumericus.blogspot.com.br/2013/03/calculando-e-simulado-um-circuito.html`

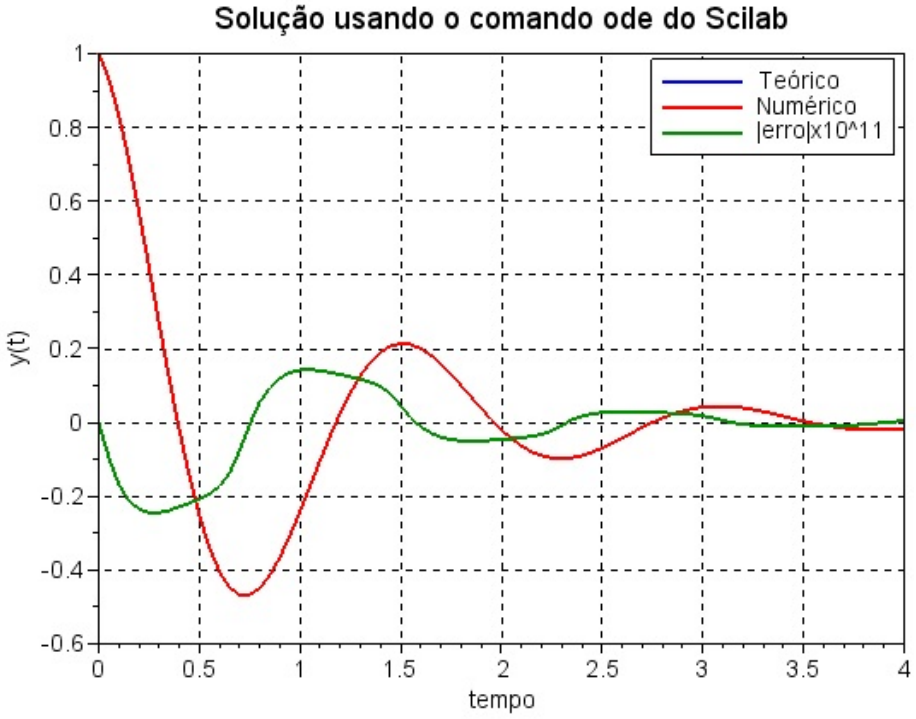


Figura 10.7: Solução de $y' = -y - 4e^{-t}\sin(4t)$ via comando "ode".

saída $y(t)$ sobre o indutor e capacitor que estão em paralelo. Sabemos da análise de circuitos que

$$i_R(t) = \frac{y(t)-10}{R} \text{ corrente elétrica no resistor}$$
$$i_C(t) = C\frac{dy}{dt} \text{ corrente no capacitor}$$
$$i_L(t) = \frac{1}{L}\int_{-\infty}^{t} y(\tau)d\tau \text{ corrente no indutor}$$

Sabemos também que $i_R + i_C + i_L = 0$. Logo,

$$\frac{y(t)-10}{R} + C\frac{dy}{dt} + \frac{1}{L}\int_{-\infty}^{t} y(\tau)d\tau = 0$$

Derivando a equação acima e substituindo os valores de C, L e R obtemos:

$$\frac{d^2y}{dt^2} + \frac{1}{2}\frac{dy}{dt} + y(t) = 0$$

com as condições iniciais $y(0) = 0$ e $y'(0) = 5$, pois

$$\frac{dy}{dt}|_{t=0} = -\frac{y(0)-10}{2} - \int_{-\infty}^{0} y(\tau)d\tau$$
$$= \frac{10}{2} - 0 = 5$$

A solução analítica desta EDO é

$$y(t) = \frac{4\sqrt{15}}{3} e^{-t/4} \sin\left(\frac{\sqrt{15}}{4} t\right) \text{ volt}$$

Usando o método de Euler melhorado, conseguimos a solução numérica mostrada na Figura 10.9. O erro relativo máximo é da ordem de $0,2\%$, o que mostra uma boa concordância entre o valor teórico e o numérico. Para uma precisão ainda maior, poderíamos usar RK-3 ou RK-4. O código Scilab é semelhante aos já mostrados.

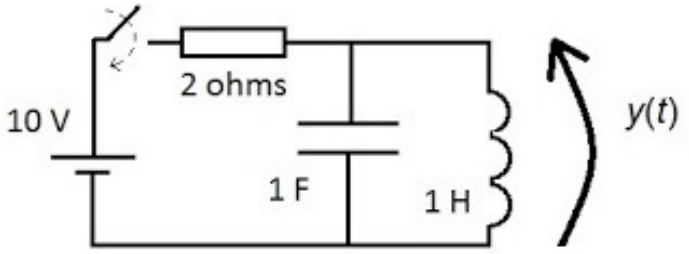


Figura 10.8: Circuito elétrico: EDO de 2a. ordem.

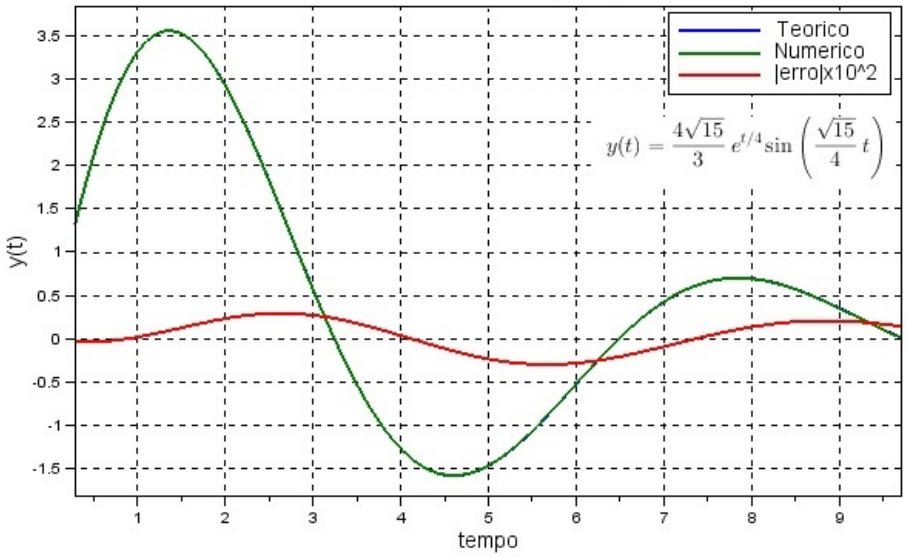


Figura 10.9: Solução da EDO do circuito elétrico (RK-2).

10.9 Problemas

Questão 1. Usando o método de Euler, encontre $y(t)$ para $0 \leq t \leq 1s$. Use o passo $h = 0,05$:

(a) $y' = 2y$, $y(0) = 1$.
(b) $y' = 2cos(2t)$, com $y(0) = 0$.
(c) $y' = t^{-1/2}$, $y(0) = 1$.
(d) $y' = e^{-3t} - 3y$, $y(0) = 0$.

(e) $y' = -2ty$, $y(0) = 1$.
(f) $y' = exp(-t)$, $y(0) = 0$.
(g) $y' + 2y = \cos(3t)$, $y(0) = 0$.
(h) $y' = e^{-y}$, com $y(0) = 1$.
(i) $y' = t/y + e^{-t/2}$, com $y(0) = 1$.
(j) $y^2 y' = \cos(t) + 0,15$, com $y(0) = 1$.

Questão 2. Refaça os itens da questão anterior usando o método de Euler melhorado.

Questão 3. Para a EDO $y' = -2ty$, $y(0) = 1$, use o método de Taylor até a terceira ordem, isto é, use a expressão

$$y_{k+1} = y_k + hy' + \frac{h^2}{2}y'' + \frac{h^3}{6}y'''$$

para calcular $y(t)$. Mostre que esse resultado tem maior acurácia do que se fosse usado o método de Euler melhorado.

Questão 4. Refaça a primeira questão usando os métodos RK-3 e RK-4.

Questão 5. Refaça a primeira questão usando o método de Adams-Bashforth/Adams-Moulton de passo 4.

Questão 6. Refaça a primeira questão usando o comando "ode" do Scilab.

Questão 7. Podemos modelar a queda de um corpo de massa m sob a ação da força gravitacional g (suposta constante) em um ambiente com a presença da resistência do ar por:

$$m\frac{dv}{dt} = mg - kv^2$$

onde k é uma constante. Suponha que $m = 1,0$ kg, $g = 9,81$ m/s^2 e $k = 0,1$. Encontre a velocidade em função do tempo usando so métodos de Euler, RK-2, RK-3 e RK-4. Escolha um passo h adequado.

Questão 8. A convivência entre presas e predadores não é fácil. Os predadores dependem de suas presas para a sobrevivência. Já as presas, sem esse "controle" externo, poderiam aumentar muito em número e exaurir os recursos naturais, levando toda a população à extinção. É um equilíbrio dinâmico delicado. Esse equilíbrio, para o caso de um predador (ex: lobos) e uma presa (ex: coelhos) específicos, foi matematicamente estabelecido pelos pioneiros na biomatemática A. Lotka e V. Volterra, no início do século XX. O sistema de equações diferenciais Lotka-Volterra pode ser expresso por:

$$\frac{dx}{dt} = x(\alpha - \beta y)$$
$$\frac{dy}{dt} = y(\delta x - \gamma y)$$

Onde x é a população de predadores e y é a população de presas. Pequenas variações nos parâmetros α, β, δ e γ podem levar a resultados instáveis e à extinção dos dois grupos. A solução desse sistema de equações (quando não ocorre a extinção) apresenta um comportamento oscilatório periódico simples. Vale lembrar que esse é um modelo simplificado, pois não inclui fatores externos como, por exemplo, o clima, limitação de recursos naturais, competição dentro da própria população, etc. Simule o sistema acima para uma população de lobos inicialmente com 35 indivíduos, mil coelhos e os parâmetros $\alpha = 0,1$; $\beta = 0,00015$; $\delta = 0,005$; $\gamma = 0,15$.

Questão 9. O sistema de equações diferenciais não lineares acopladas abaixo

$$\frac{dx}{dt} = s(y-x)$$
$$\frac{dy}{dt} = rx - y - xz$$
$$\frac{dz}{dt} = -bz + xy$$

modelam a intensidade de movimento do fluido atmosférico x às variações de temperatura y e z nas direções horizontal e vertical, respectivamente. Resolva esse sistema para os parâmetros $h = 0,01$, $s = 10$, $b = 8/3$ e $r = 28$, com $x(0) = 10$, $y(0) = 20$, $z(0) = 30$. Faça t variar de 0 a 10 e aprecie as belas curvas que surgem.

Questão 10. Considere a seguinte equação diferencial:

$$\frac{d^2y}{dt^2} + 15\frac{dy}{dt} + 70y(t) = 5\cos(10t)$$

com $y(0) = 0$ e $y'(0) = 5$. Determine $y(t)$ para $0 < t < 3$. Apresente o resultado na forma de um gráfico $t \times y$. Escolha um passo h adequado para esse problema.

Questão 11. Considere a seguinte EDO de 2^a ordem:

$$\frac{d^2y}{dt^2} + 10\frac{dy}{dt} + 40y(t) = 10 + 5\,\mathrm{sen}(5t)$$

com $y(0) = 0$ e $y'(0) = 1$. Determine $y(t)$ para $0 < t < 5$. Apresente o resultado na forma de um gráfico $t \times y$. Escolha um passo h adequado para esse problema.

11. Gerando números pseudo aleatórios

Até mesmo o acaso não é impenetrável; tem suas próprias regras.
Novalis (Georg P. Friedrich von Hardenberg, 1772 - 1801), escritor alemão.

EXISTEM muitas aplicações (ex: criptografia) ou simulações (ex: taxa de erro em sistemas de comunicação digital) nas quais são requeridos sequências de números aleatórios. Na verdade, não existe uma forma segura de gerar números *aleatórios* por *software*, mas podemos gerar sequências pseudoaleatórias que apresentem boas características estatísticas. Existem equipamentos que conseguem gerar números aleatórios baseados em fenômenos físicos, mas eles são caros. Neste capítulo daremos ênfase aos geradores de números pseudoaleatórios do Scilab. Antes de prosseguirmos, uma definição de trabalho para gerador de números "aleatórios": algoritmo capaz de gerar uma sequência de números "independentes" uns dos outros de longo período e que apresentam alguma distribuição estatística, normalmente, distribuição uniforme.

11.1 Um gerador simples

Como podemos gerar números pseudoaleatórios? Ter um bom gerador de números pseudoaleatórios não é uma tarefa trivial. Uma forma básica de conseguirmos é com o uso da seguinte fórmula (gerador congruente linear (GCL), apresentado por D. H. Lehmer[1] em 1949):

$$x_{n+1} = (Ax_n + B) \; mod \; M \tag{11.1}$$

onde M é um número inteiro grande, B é normalmente um valor pequeno ou zero e A um fator multiplicador da ordem de 10^5. O valor inicial x_0 é a "semente". Para cada semente, é, idealmente, gerada uma sequência completamente nova de números. Eventualmente, a sequência irá se repetir

[1] Derrick Henry "Dick" Lehmer (Berkeley (Califórnia), 23 de fevereiro de 1905 Berkeley (Califórnia), 22 de maio de 1991) foi um matemático estadunidense. Tinha interesse em teoria dos números e chegou a trabalhar com o Último Teorema de Fermat, também se interessou pela área de Computação. Recebeu o prêmio Gibbs Lecture em 1965.

após uma certa quantidade valores gerados, como ilustra a Figura 11.1. Para aprofundar esse tema sugerimos o artigo de Volchan (2004) e o capítulo sobre números pseudoaleatórios de PRESS et al. (2011). Existem testes sofisticados para saber se uma sequência é mais ou menos "aleatória" (RUKHIN, A., et al., 2010), mas isso foge ao escopo deste livro. Este é um problema que aflige muitos engenheiros, inclusive Dilbert[2].

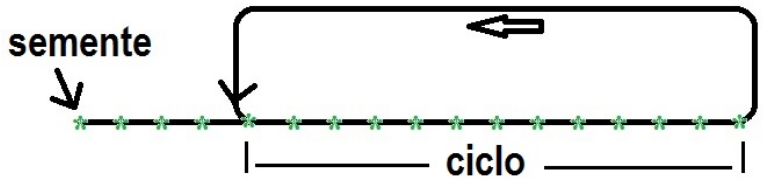


Figura 11.1: Sequência pseudoaleatória.

Da equação 11.1, com os parâmetros $M = 2147483647$, $A = 39373$, $B = 3$ e $x_0 = 345$, podemos escrever o seguinte código Scilab e obtemos a seguinte sequência mostrada na Figura 11.2. A divisão $xs = xs/M$ é para normalizar os números gerados entre 0 e 1.

```
M = 2147483647;
A = 39373;
B = 3;
xs = 345 + zeros(1,100);  // inicializando o vetor xs
for k=2:100
    xs(k) = (A * xs(k-1) + B);
    xs(k) = modulo(xs(k),M);
end
xs = xs/M;   // valores no intervalo (0,1)
disp([mean(xs), max(xs), min(xs), variance(xs)]);
close; plot(xs); title('Exemplo de Sequência Pseudoaleatória');
```

11.2 Distribuição uniforme com Scilab

O comando básico para gerar sequências pseudoaleatórias é o "rand". Esse comando, dependendo dos parâmetros adicionais que são usados, pode gerar uma sequência ou uma matriz de números pseudoaleatórios com distribuição uniforme ou normal, ou iniciar a semente com um valor. A sua sintaxe para gerar uma sequência uniforme é: "rand(1,N1,'u');". Um exemplo simples:

```
/////// Gerando Números P-aleatórios
rand('seed',345);  // semente inicial
x = rand(1,10000,'u');  // gerando 10000 números com distribuição uniforme
xm = mean(x); // média dos valores gerados
```

[2]Um gerador não muito confiável de números aleatórios: `https://boardgamegeek.com/thread/1652674/scott-adams-funny`.

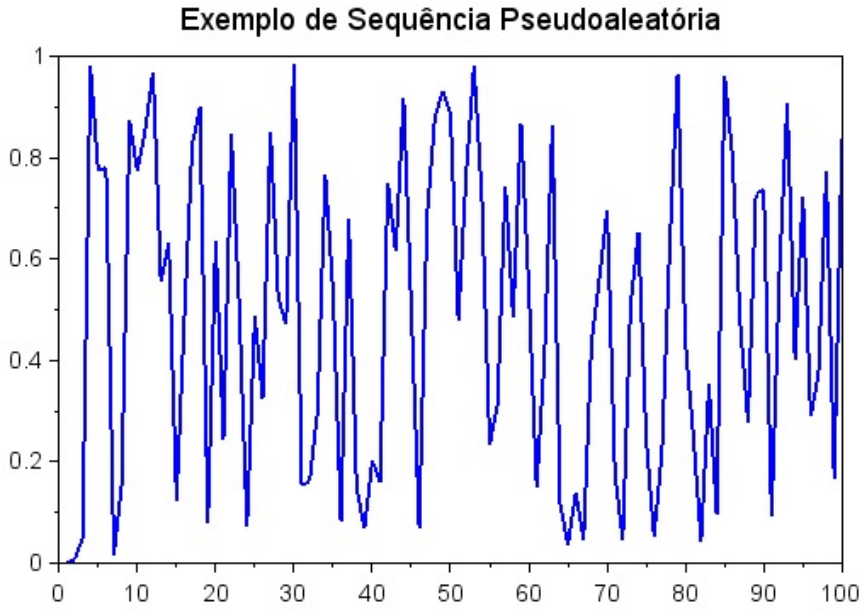


Figura 11.2: Sequência gerada pelo código Scilab.

```
xv = variance(x); // variância de x
xmax = max(x); // valor máximo de x
xmin = min(x); // valor mínimo de x
histplot(10,x); title('Histograma de x');
mprintf('Min: %f, Med: %f, Max: %f, Var: %f \n',xmin,xm,xmax,xv);
```

Os valores ideais para mínimo, média, máximo e variância são: 0, $1/2$, 1 e $1/12$, os obtidos são:

```
Min: 0.000002, Med: 0.500554, Max: 0.999872, Var: 0.082618.
```

que estão bem próximos dos valores ideais. A Figura 11.3 mostra o histograma da sequência gerada.

11.3 Distribuição gaussiana

Em muitos casos, vários fatores diferentes e independentes contribuem para gerar um sinal contínuo ou discreto. Esse sinal pode ser modelado como uma função aleatória com distribuição normal. O ruído térmico é um exemplo de sinal que distribuição normal, sendo a média nula. No Scilab, podemos gerar um sinal com distribuição normal ou gaussiana usando o comando "rand" já apresentado: "rand(N1, N2, 'n');" gera uma matriz $N_1 \times N_2$ de números pseudoaleatórios com distribuição normal e média nula. Vejamos um exemplo:

```
/////// Gerando Números P-aleatórios
rand('seed',345);  // semente inicial
x = rand(1,10000,'u');  // gerando 10000 números com distribuição normal
xm = mean(x); // valor médio dos valores gerados
xv = variance(x); // variância de x
xmax = max(x); // valor máximo de x
```

```
xmin = min(x); // valor mínimo de x
histplot(15x); title('Histograma de x');
mprintf('Min: %f, Med: %f, Max: %f, Var: %f \n',xmin,xm,xmax,xv);
```

Os valores ideais para mínimo, média, máximo e variância são: $-\infty$, 0, $+\infty$ e 1, os obtidos são:

```
Min: -4.175325, Med: 0.022787, Max: 3.545110, Var: 1.020335
```

a média e a variância estão bem próximos dos valores ideais, mas, naturalmente, os valores máximos e mínimos obtidos pela simulação *nunca* poderiam chegar "perto" de ∞. A Figura 11.4 mostra o histograma da sequência gerada. A Figura 11.5 mostra o efeito de um sinal aleatório adicionado a um sinal senoidal.

Comparando o valores gerados com o "rand" com a função de densidade teórica (ver Figura 11.6)

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

```
x = rand(1,1e5, 'n');     // gerando 10^5 valores
x = (x-mean(x))/sqrt(variance(x)); // normalizando
histplot(31,x);  // histograma
z=-4:0.01:4;
y=exp(-z.*z/2)/sqrt(2*%pi);
plot(z,y,'m');  // gráfico teórico sobre o histograma
title('Comparação entre curva teórica e simulação');
```

11.4 Outras distribuições

Algumas situações exigem o uso de sequências pseudoaleatórias com comportamento estatístico que não seja uniforme ou normal. A velocidade do vento em regiões litorâneas tende, por exemplo, a seguir uma distribuição de Weibull. O Scilab possui um extensa chamada da função "grand":

- Y = grand(m, n, "bet", A, B) - distribuição beta;
- Y = grand(m, n, "bin", N, p) - distribuição binomial;
- Y = grand(m, n, "nbn", N, p) - distribuição binomial negativa;
- Y = grand(m, n, "chi", Df) - distribuição chi-quadrado;
- Y = grand(m, n, "nch", Df, Xnon) - distribuição chi-quadrado não central;
- Y = grand(m, n, "exp", Av) - distribuição exponencial;
- Y = grand(m, n, "gam", shape, rate) - distribuição gama;
- Y = grand(m, n, "nor", Av, Sd) - distribuição normal;
- Y = grand(m, n, "geom", p) - distribuição geométrica;
- Y = grand(m, n, "poi", mu) - distribuição de Poisson;
- Y = grand(m, n, "unf", Low, High) - distribuição uniforme, o valor "High" nunca aparece;

Onde "m" e "n" são as dimensões de uma matriz $m \times n$. Notamos que a distribuição Weibull não está na lista acima. Uma outra distribuição que não está na relação acima é a de Rayleigh[3]. A

[3]Lord Rayleigh (John William Strutt). Nascido em 12 de novembro de 1842, Langford Grove, Maldon, Essex, Reino Unido. Morreu: 30 de junho de 1919, Reino Unido. Prêmio Nobel de Física de 1904, estava no Royal Institution of Great Britain, Londres, Reino Unido. Motivação do prêmio: "por suas investigações sobre as densidades dos gases mais importantes e pela descoberta de argônio em relação a esses estudos".

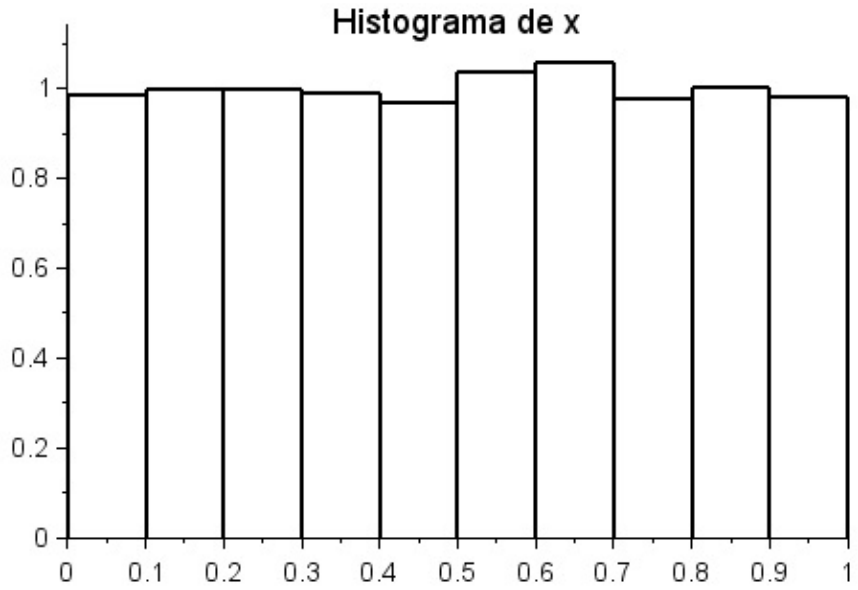


Figura 11.3: Distribuição uniforme com Scilab - números pseudoaleatórios.

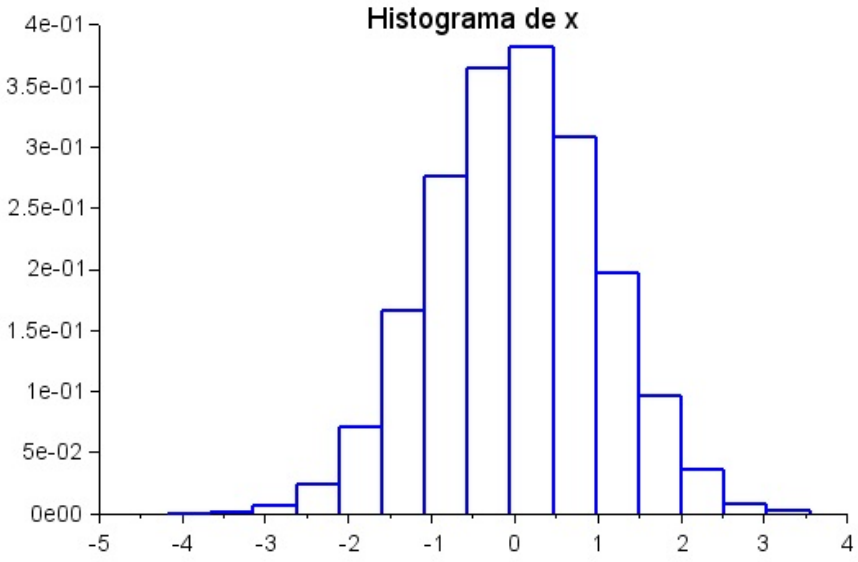


Figura 11.4: Distribuição normal com Scilab - números pseudoaleatórios.

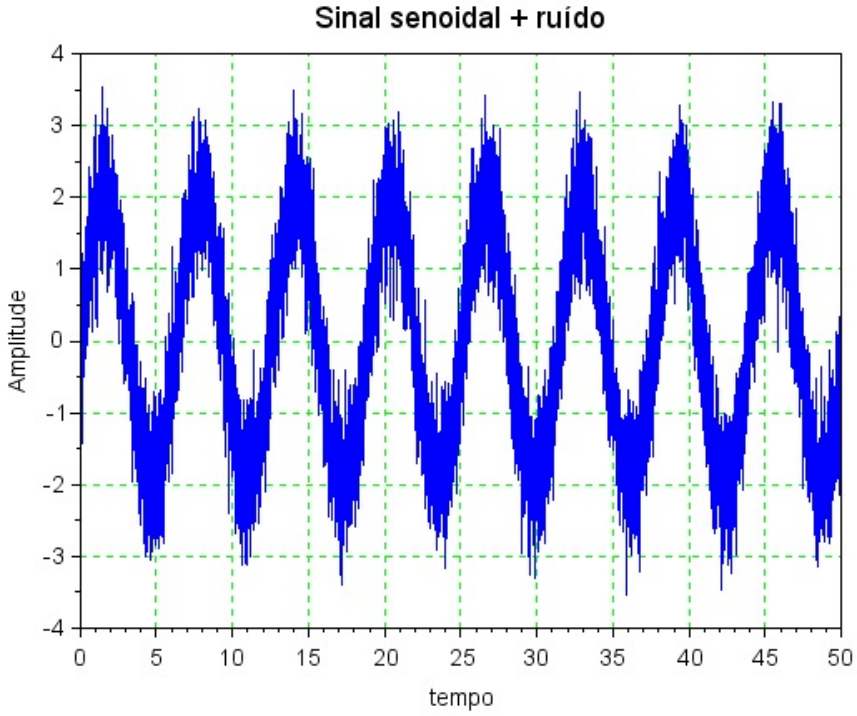


Figura 11.5: Sinal senoidal com sinal aleatório adicionado.

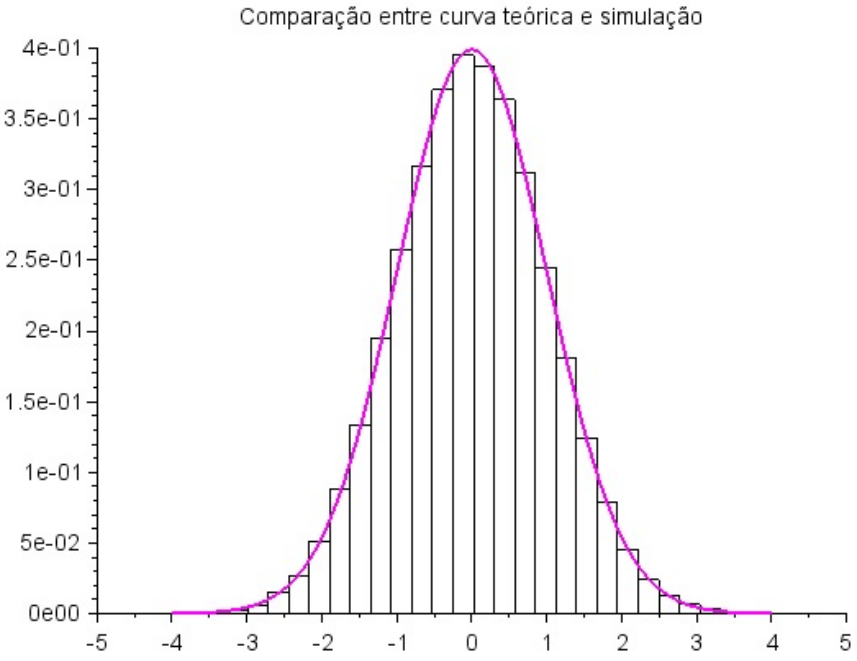


Figura 11.6: Comparação entre curva teórica e histograma simulado.

distribuição de Rayleigh pode ser facilmente obtida a partir de duas distribuições normais:

$$R = \sqrt{X^2 + Y^2} \tag{11.2}$$

sendo X e Y distribuições normais, independentes e de médias nulas. O código Scilab a seguir mostra a construção de uma distribuição de Rayleigh (ver Figura 11.7):

```
/////// Gerando Números P-aleatórios com distribuição Rayleigh
x =grand(1,10000,'nor',0,1); x2=x.*x;
y =grand(1,10000,'nor',0,1); y2=y.*y;
R = sqrt(x2 + y2);
close; histplot(20,R);
title('Distribuição de Rayleigh');
```

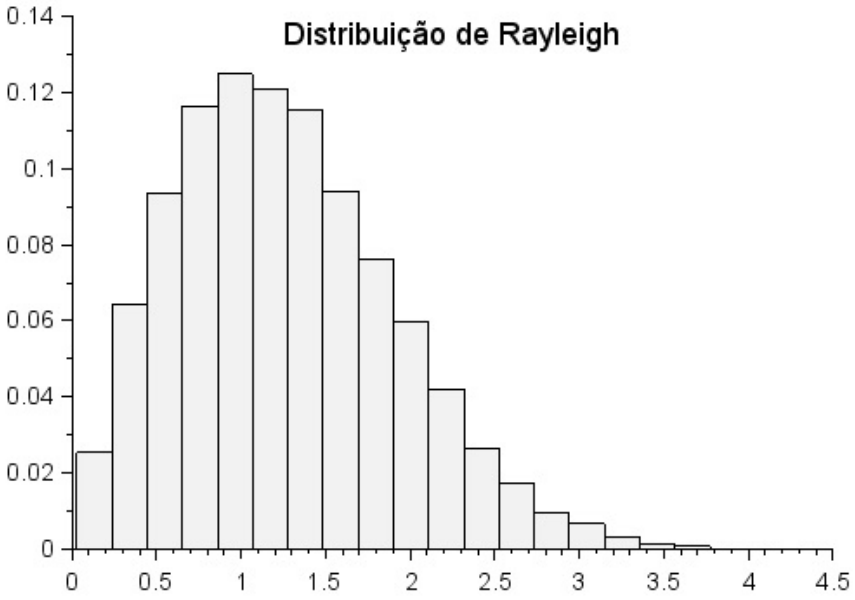


Figura 11.7: Distribuição Rayleigh.

11.5 Método de Monte Carlo

O Método de Monte Carlo (que faz referência aos famosos cassinos de Monte Carlo, um dos distritos de Mônaco) emprega números aleatórios em sucessivas simulações para obter um resultado que não é possível (ou é muito difícil) de obtermos analiticamente. Por exemplo, podemos tentar calcular o tempo médio que um carro leva para ir de um ponto A até um B dentro de uma cidade considerando o tempo parado em semáforos, eventuais acidentes ou obstáculos imprevisíveis. Em uma situação dessas, seriam necessárias várias simulações e algumas condições adicionais para ter uma estimativa realista dos tempos mínimo e máximo que esse carro levaria no percurso. Em várias outras situações podem ser úteis o uso de valores aleatórios, como, por exemplo, o cálculo da área de alguma área irregular, ver Figura 11.8. Alguns autores até consideram que os fatores aleatórios são determinantes

na nossa vida, a cada instante decisões tomadas por nós ou por outros podem afetar de forma dramática nossas existências, como indica Mlodinow em seu livro **O Andar do Bêbado**: Como o Acaso Determina Nossas Vidas.

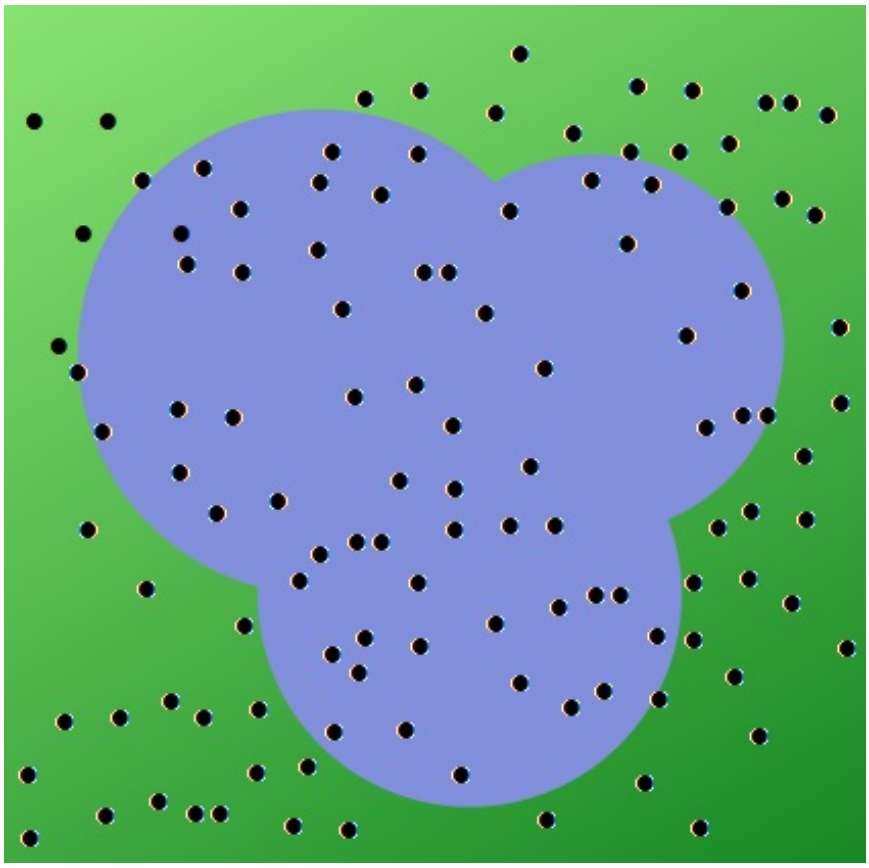


Figura 11.8: Calculando a área de um lago com o uso de números aleatórios.

Exemplo - Método Monte Carlo. Vejamos um exemplo da área de comunicação de dados digitais para demonstrar o uso do Método de Monte Carlo. Uma fonte binária gera uma sequência de *bits* 0s e 1s. Esses bits trafegam por um canal ruidoso de tal forma que o receptor tem que decidir sobre esse sinal contaminado. Se o sinal recebido estiver perto de 0, então, é considerado que o bit transmitido foi zero, caso esteja mais próximo de 1, o receptor decidirá por 1. Qual a taxa de erro quando a variância do ruído é de 0,05? Iremos considerar que o ruído tem característica de sequência aleatória normal de média nula.

Solução. Para que seja estatisticamente significante, devemos gerar muitos valores. Provavelmente, 10^5 é suficiente com folga. Podemos usar o comando "rand" para gerar tanto os valores binários (0, 1) quando o ruído normal. Para gerar 0 e 1:

```
x = grand(1,N,'nor',0,1); // números positivos e negativos
bits = (sign(x) + 1)/2;  // positivo --> 1; negativo --> 0
```

O receptor tem que decidir pelo bit 1 se o sinal recebido for maior que 0,5; caso contrário será considerado bit 0. O código completo é:

```
N=10^5;
x = grand(1,N,'nor',0,1);
bits = (sign(x) + 1)/2;
ruido = grand(1,N,'nor',0,sqrt(0.1));
br = bits + ruido;
bt = 0*br;
for k=1:N
    if br(k)>0.5 then bt(k)=1; end;
end
difs = abs(round(br - bits));
T_erros = max(size(find(difs>0)));
taxa = T_erros/N;
mprintf('Taxa de erros = %f \n',taxa);
```

Executando o código acima três vezes, obtivemos os seguintes resultados:

```
Taxa de erros = 0.025230
Taxa de erros = 0.024590
Taxa de erros = 0.024080
```

Logo, a taxa de erro esperada para esta situação é cerca de $2,45\%$. Em geral, quando fatores não totalmente compreendidos ou mensurados são entradas de um sistema, o Método de Monte Carlo é uma estratégia adequada. Por exemplo, o estudo dos ventos, direção e intensidade, são fundamentais para a decisão de instalação de um parque eólico.

11.6 Verificando se a sequência é "aleatória"

Concluímos este capítulo com um comentário sobre a "aleatoriedade" de uma sequência gerada por um algoritmo. Sabemos que essas sequências não são aleatórias, mas, sob certos critérios, elas podem se passar razoavelmente como randômicas. Na Figura 11.9, temos a comparação entre dois geradores.

O código que usamos para gerar a Figura 11.9 é:

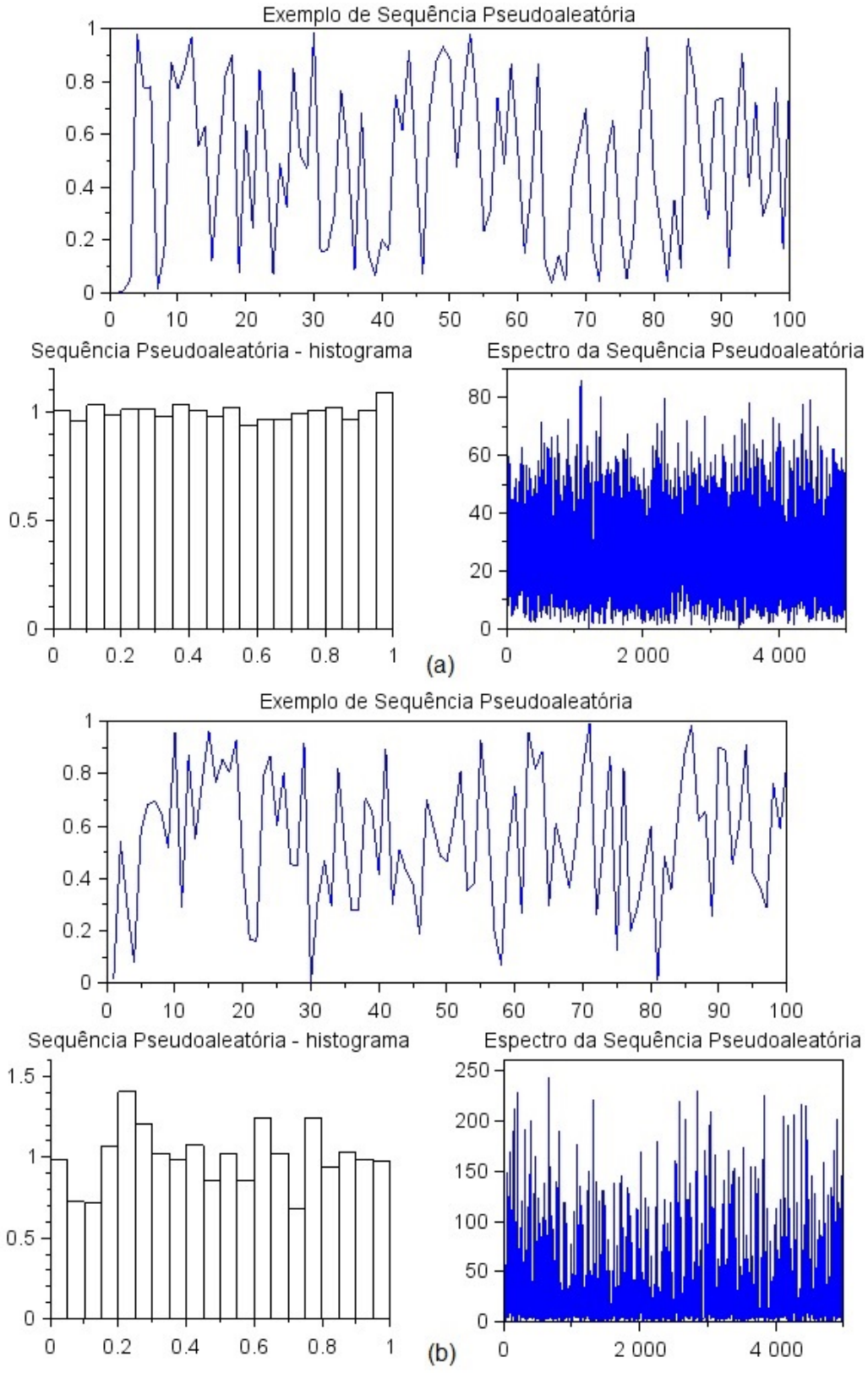


Figura 11.9: Dois geradores de números aleatórios: (a) um bom gerador; (b) um mau gerador.

```
// Gerador ‘‘bom’’
M = 2147483647;
A = 39373;
B = 3;
N = 1e4;
xs = 345 + zeros(1,N);
for k=2:N
    xs(k) = (A * xs(k-1) + B);
    xs(k) = modulo(xs(k),M);
end
xb = xs/M;
disp([mean(xb),max(xb),min(xb),...
variance(xb)]);
xsf = abs(fft(xb));
close;
subplot(2,1,1); plot(xb(1:100));
title('Exemplo de Sequência ...
Pseudoaleatória');
subplot(2,2,3); histplot(20,xb);
title('Seq. Pseudoaleatória - ...
histograma');
subplot(2,2,4); plot(xsf(2:N/2));
title('Espectro da Sequência ...
Pseudoaleatória');
```

```
// Gerador ‘‘ruim’’
M = 1877;
A = 297;
B = 3;
xs = 35 + zeros(1,N);
for k=2:N
    xs(k) = (A * xs(k-1) + B);
    xs(k) = modulo(xs(k),M);
end
xr = xs/M;
disp([mean(xr),max(xr),min(xr),...
variance(xr)]);
xsf = abs(fft(xr));
figure;
subplot(2,1,1); plot(xr(1:100));
title('Exemplo de Sequência ...
Pseudoaleatória');
subplot(2,2,3); histplot(20,xr);
title('Seq. Pseudoaleatória - ...
histograma');
subplot(2,2,4); plot(xsf(2:N/2));
title('Espectro da Sequência ...
Pseudoaleatória');
```

Esses dois geradores deveriam gerar idealmente sequências uniformemente distribuídas. A primeira sequência consegue ser mais próxima da distribuição uniforme, já a segunda está longe de ser "aleatória", pois apresenta um ciclo de repetição muito curto e piores valores estatísticos. O cálculo do espectro (fft) das sequências geradas também indicam que a segunda sequência é menos "branca", o que nos mostra que ela é pior. Um teste de qui-quadrado (χ^2) também revela que a primeira sequência é melhor distribuída. Podemos realizar esse teste com a seguinte sequência de comandos (aproveitando o código acima):

```
q1 = histplot(20,xb);  // gerador ‘‘bom’’
q2 = histplot(20,xr);  // gerador ‘‘ruim’’

dq1 = q1 - 1;
dq1 = sum(dq1.*dq1);

dq2 = q2 - 1;
dq2 = sum(dq2.*dq2);

disp([dq1, dq2]);
```

O resultado mostrado é:

```
0.0218248    0.6403845
```

ou seja, a primeira sequência gera valores mais próximos da distribuição ideal (distribuição uniforme).

11.7 Problemas

Questão 1. Um estudante implementou um gerador congruente linear com os seguintes parâmetros: $A = 33$, $B = 1$, $M = 3997$, com a semente $x_0 = 15$. Esse é um bom gerador de números "aleatórios"? Os números começam a se repetir para qual tamanho de sequência?

Questão 2. Bons geradores de números "aleatórios" que rodam com a fórmula de congruência linear usam os seguintes valores:

A	M	B	Referência/autor
16807	2147483647	0	Lewis, Goodman e Miller
39373	2147483647	0	L'Ecuyer
742938285	2147483647	0	Fishman e Moore
950706376	2147483647	0	Fishman e Moore
1226874159	2147483647	0	Fishman e Moore
40692	2147483399	0	L'Ecuyer
40014	2147483563	0	L'Ecuyer

Verifique que, com estes parâmetros, a sequência precisa ser muito longa para começar a se repetir, idealmente o tamanho da sequência é igual a M.

Questão 3. Refaça o exemplo do Método Carlo (sequência de bits contaminados por ruído) apresentado para um ruído com variância igual $0,1$ e $0,5$, mantendo o mesmo cenário geral. O que ocorre com a taxa de erro?

Questão 4. Suponha que a população brasileira segue uma distribuição normal com média igual a 100 e desvio padrão igual 16 no quesito inteligência (o famoso teste de **QI** - quociente de inteligência). Qual a probabilidade de encontrar ao acaso uma pessoa com **QI**$>$ 150? Quantos brasileiros devem ter **QI** acima de 180? Segundo o IBGE (Instituto Brasileiro de Geografia e Estatística), a população brasileira é aproximadamente de 211,7 milhões de brasileiros (em 2020).

Questão 5. Um atirador de elite alveja um alvo fixo. Depois de muitos tiros, alguns quase acertam o alvo central e outros ficam em volta do centro a distâncias variáveis. Como poderíamos modelar essas distâncias ao centro do alvo?

Questão 6. Às três horas da manhã, um homem bêbado tenta caminhar em linha reta para frente com o objetivo de atravessar uma rua muito larga, mas tudo o que ele consegue é dar um ou dois passos para frente seguido de um passo para o lado direito (50% de probabilidade) ou esquerdo (50% de probabilidade). Suponha que ele consegue dar um passo de um metro a cada segundo (a frente ou para o lado). A probabilidade dele conseguir dar dois passos à frente é de 20%, 40% de dar apenas um passo à frente é 40% para um dos lados. Modele esse "sistema" e calcule, em média, quanto tempo ele precisaria para atravessar uma rua de 20 m de largura. Não se preocupe com os carros: é feriado e não existem carros circulando nesse horário.

Questão 7. Usando um gerador de números aleatórios uniformemente distribuídos, estime a área de um círculo de raio unitário. Dica: a Figura 11.8 pode lhe ajudar pensar neste problema.

Questão 8. Usando o comando "rand", com distribuição normal, gere 10^6 números "aleatórios". Quais as estatísticas básicas dessa sequência? Qual o maior valor e o menor valores gerados?

Questão 9. O resistor é um componente usual nos circuitos elétricos e eletrônicos. Na especificação de um resistor, sempre existe a tolerância: o valor dele pode variar dentro de uma faixa de valores

em torno de um valor nominal. Se essa variação puder ser modelada como um valor aleatório com distribuição uniforme de média nula, quantos resistores de 100Ω precisamos ter para "garantir" que pelo menos um deles será "exatamente" (erro menor que 1Ω) de 100Ω?. Considere que você tem disponível resistores com tolerância de 10%, isto é, os valores podem variar de 90Ω a 110Ω.

Questão 10. Dois resistores (ver questão anterior) de $100\Omega \pm 10\%$ são usados em paralelo gerando um resistor equivalente. Qual a probabilidade dessa associação de resistores gerar um resistor equivalente maior que 57Ω? Lembrar que

$$R_{eq} = \frac{R_1 R_2}{R_1 + R_2}$$

Questão 11. Refaça a questão anterior, mas usando três resistores de $100\Omega \pm 10\%$ em paralelo para gerar um resistor equivalente. Qual a probabilidade dessa associação de resistores gerar um resistor equivalente menor que 30Ω? Lembrar que

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$

Questão 12. Considere uma associação série de cinco resistores de mesmo valor nominal $(1k\Omega)$ e mesma tolerância (10%). Qual a probabilidade de que o resultado dessa associação seja maior que $5,4k\Omega$?

Questão 13. Usando o comando "rand" (ou "grand") gere quatro sequências de 10000 valores "aleatórios" cada uma, todas com distribuição uniforme. Ao somar essas quatro sequências para gerar uma nova sequência, qual o formato do histograma (comando "histplot") desta sequência? Essa nova sequência se aproxima de uma sequência com distribuição normal?

Questão 14. Considere uma transmissão de 10^6 dados binários (0 ou 1). A probabilidade de geração de zeros e uns são iguais a 50%. Qual a probabilidade de chegar mais de 500.500 bits 0 no receptor? Considere que os bits são independentes e que não existe ruído.

Questão 15. Em uma comunicação digital, são transmitidos 80.000 *bits* por um canal ruidoso em grupos de 8 *bits*. A probabilidade que um *bit* seja erroneamente interpretado errado no receptor é de 1%. Usando o Scilab para modelar esse cenário, qual a probabilidade de que ocorram dois erros em um grupo de 8 *bits*?

Questão 16. Quais os próximos números da Mega Sena? A resposta dessa pergunta pode valer mais de um milhão de reais. Faça um programa Scilab que gere seis números "aleatórios" entre 1 e 60 com distribuição uniforme. E boa sorte!

Questão 17. A resposta em frequência de um filtro do tipo passa-faixa é dada por

$$G(\omega) = \frac{j\omega}{(j\omega)^2 + jQ\omega + \omega_0^2}$$

com $j = \sqrt{-1}$, $Q = 20 \pm 5\%$, $w_0^2 = 100 \pm 10\%$ e $0 < \omega < 50$ rad/s. Usando o método de Monte Carlo, verifique em qual faixa de frequência ocorre o ganho máximo.

Questão 18. Simule o lançamento simultâneo de 5 dados não viciados. A variável desejada é a soma dos resultados. A distribuição dessa soma se aproxima de uma distribuição gaussiana? Se sim, qual a média e o desvio padrão?

Questão 19. A sequência de números naturais que observamos nas casas decimais de π (3,1415926535 8979323846 2643383279 ...) pode ser usada como um gerador de números aleatórios inteiros na faixa de 0 a 9? Discuta esse problema.

Questão 20. Considere o lançamento de uma moeda honesta. Qual a probabilidade de aparecerem 5 caras em sequência? E 4 caras e uma coroa? Faça um programa o Scilab para simular esse experimento.

Questão 21. Considere uma transmissão de dados binários, isto é, símbolos $\pm\pm 1$ são transmitidos por um transmissor para um receptor através de um canal de comunicação. Se esses dados são contaminados por um ruído gaussiano de média nula e variância $\sigma^2 = 0,25$, qual a probabilidade de erro no receptor?

12. Apêndice - constantes e miscelânea

Arquimedes será lembrado enquanto Ésquilo foi esquecido, porque os idiomas morrem mas as ideias matemáticas permanecem. "Imortalidade" pode ser uma ideia tola, mas provavelmente um matemático tem a melhor chance que pode existir de obtê-la. G. H. Hardy (Cranleigh, 7 de fevereiro de 1877 Cambridge, 1 de dezembro de 1947) foi um matemático inglês.

APRESENTAMOS neste apêndice algumas informações, fórmulas e complementos que podem ser úteis tanto para este livro quanto para outros temas de matemática, física e engenharia.

12.1 Constantes matemáticas e físicas

Durante séculos, houve um contínuo interesse em calcular mais e mais casas decimais para algumas constantes matemática, como, por exemplo, o π. Hoje, temos programas que consegue usar algoritmos avançados que podem calcular o valor do π como milhões de casas decimais. Alguns matemáticos devotaram muito tempo de suas vidas nessa tarefa, hoje algo que pode ser visto como "desnecessário". O π é, provavelmente, a constante matemática irracional mais bem conhecida. Seus primeiros valores são (calculados com o programa **SuperPi** - ver Figura 12.1)[1]:

$\pi \cong 3,$
1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 5923078164
0628620899 8628034825 3421170679 8214808651 3282306647 0938446095 5058223172
5359408128 4811174502 8410270193 8521105559 6446229489 5493038196 4428810975
6659334461 2847564823 3786783165 2712019091 4564856692 3460348610 4543266482
1339360726 0249141273 7245870066 0631558817 4881520920 9628292540 9171536 ...

[1] Um *link* para baixar este programa: `https://www.techpowerup.com/download/super-pi/`

Já o número de Euler (base dos logaritmos naturais) é

$e \cong 2,$
718281828 459045235 360287471 352662497 757247093 699959574 966967627 724076630
353547594 571382093 699959574 966967627 724076630 353547594 571382178 525166427
427466391 932003059 921817413 596629178 525166427 427466391 932003059 921817 ...

Algumas expressões para o cálculo de e:

$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n$$

$$e = \lim_{h\to 0} (1+h)^{\frac{1}{h}}$$

$$e = \lim_{n\to\infty} \frac{n}{\sqrt[n]{n!}}$$

$$e = \lim_{n\to\infty} \left[\frac{(n+1)^{n+1}}{n^n} - \frac{n^n}{(n-1)^{n-1}}\right]$$

Outras constantes matemáticas e físicas importantes estão listadas na Tabela 12.1.

Tabela 12.1: Algumas constantes físicas e matemáticas

Símb.	Valor	Nome
γ	0,57721 56649 01532 86060 65120 90082 40243	constante de Euler-Mascheroni
ϕ	1,61803 39887 49894 84820 45868 34365 63811	Número de Ouro
$\sqrt{2}$	1,41421 35623 73095 04880 16887 24209 69807	raiz quadrada de 2
$\sqrt{3}$	1,73205 08075 68772 935...	raiz quadrada de 3
$\log_{10} 2$	0,30102 99956 63981 19521 37389	logaritmo de 2 na base 10
$\ln 2$	0,69314 71805 59945 30941 7232...	logaritmo natural de 2
e^{π}	23,14069 26327 79269 006...	exponencial de π
π^e	22,45915 77183 61045 473...	π elevado a e
c_0	299.792.458 m/s	velocidade da luz no vácuo
q_e	-1,60217653 $\times 10^{?19}$ C	carga do elétron
m_e	9,10938215 $\times 10^{-31}$ kg	massa do elétron
m_p	1,672621777(74)$times 10^{-27}$ kg	massa do próton
m_n	1,674928 $times 10^{-27}$ kg	massa do nêutron
G	6,67408 $\times 10^{-11}$ $m^3 kg^{-1} s^{-2}$	constante Gravitacional
N_a	6,022140857 $\times 10^{23}$ mol^{-1}	constante de Avogrado
h	6,6260693 $\times 10^{-34}$ Joule-s	constante de Planck
ell_P	$\sqrt{\frac{G}{c^3}} \approx 1.616\,229(38) \times 10^{-35}$ m	comprimento de Planck
k	1,3806505 $\times 10^{-23}$ Joule/kelvin	constante de Boltzmann
K_o	8,9874 $\times 10^9$ Nm^2C^{-2}	Constante de Coulomb

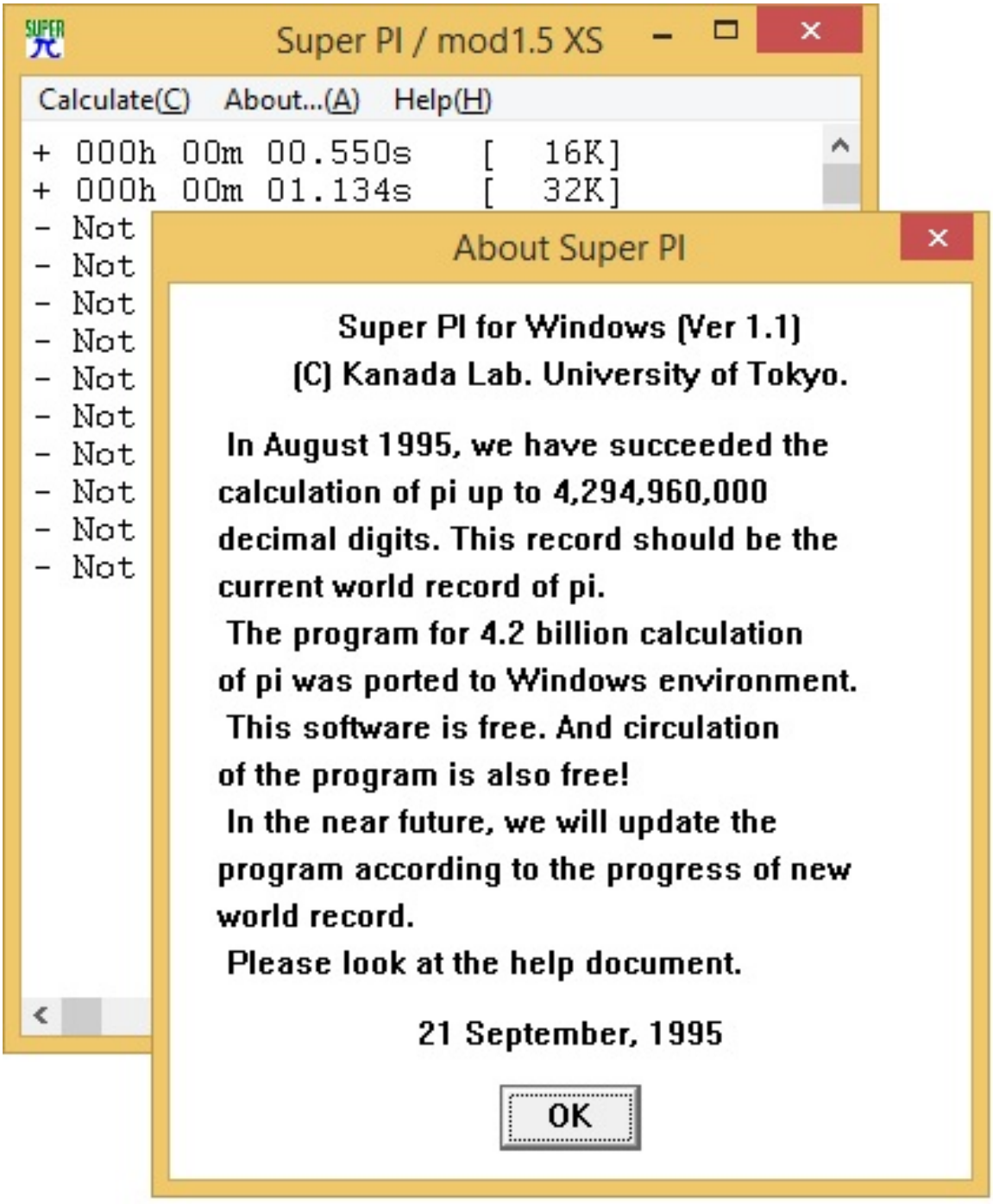


Figura 12.1: *Software* livre que calcula o valor de π com *muitas* casas decimais.

12.2 Somatórios

$$\sum_{n=1}^{N} 1 = N, \quad \sum_{n=1}^{N} n = \frac{N(N+1)}{2}$$
$$\sum_{n=1}^{N} n^2 = \frac{N(N+1)(2N+1)}{6}$$
$$\sum_{n=1}^{N} n^3 = \left(\frac{N(N+1)}{2}\right)^2$$
$$\sum_{n=1}^{N} n^4 = \frac{N(N+1)(2N+1)(3N^2+3N-1)}{30}$$
$$\sum_{n=1}^{N} n^5 = \frac{N^2(N+1)^2(2N^2+2N-1)}{12}$$

Outros somatórios:

$$\sum_{n=0}^{N} a^n = \frac{a^{N+1}-1}{a-1}, a \neq 1$$
$$\sum_{n=0}^{\infty} a^n = \frac{1}{1-a}, |a| < 1$$
$$\sum_{n=0}^{N-1} na^n = \frac{a - Na^N + (N-1)a^{N+1}}{(1-a)^2}, a \neq 1$$
$$\sum_{p=1}^{N} \frac{1}{p(p+1)} = \frac{N}{N+1}$$
$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$
$$\sum_{p=1}^{N} \frac{1}{p^2+3p+2} = \frac{N}{2(N+2)}$$
$$\sum_{n=0}^{\infty} \frac{x^n}{n!} = e^x$$
$$\sum_{k=1}^{\infty} \frac{k}{2(k-1)!} = e$$
$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4}$$
$$\sum_{n=1}^{\infty} \frac{(\theta)^n(-1)^{n+1}}{(2n-1)!} = \text{sen}(\theta)$$

12.3 Algumas integrais definidas

Talvez a integral imprópria mais conhecida seja

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

algumas outras são:

$$\int_0^{\infty} \frac{dx}{(x+1)\sqrt{x}} = \pi$$
$$\int_0^{\infty} \frac{\operatorname{sen}(x)}{x} dx = \frac{\pi}{2}$$
$$\int_0^{\infty} \frac{dx}{x^2+1} = \frac{\pi}{2}$$
$$\int_0^{\infty} \frac{x}{e^x-1} dx = \frac{\pi^2}{6}$$
$$\int_0^{\infty} \frac{x^3}{e^x-1} dx = \frac{\pi^4}{15}$$

12.4 Alfabeto grego

Neste livro, bem como em *todos* os livros de matemática e física, usamos o alfabeto grego para representar símbolos, variáveis e incógnitas. Na Tabela 12.2, listamos as letras gregas (maiúsculas e minúsculas) e a pronúncia aproximada em português.

Tabela 12.2: Algumas constantes físicas e matemáticas

Alfabeto grego					
Maiúscula	Minúscula	Nome	Maiúscula	Minúscula	Nome
A	α	alfa	O	o	omicron
B	β	beta	N	ν	ni
X	χ	qui	Π	π	pi
Δ	δ	delta	Θ	θ	teta
E	ε	epsilon	P	ρ	rô
Φ	ϕ, φ	fi	Σ	σ, ς	sigma
Γ	γ	gama	T	τ	tau
H	η	eta	Υ	υ	úpsilon
I	ι	iota	Ω	ω	ômega
K	κ	capa	Ξ	ξ	csi
Λ	λ	lambda	Ψ	ψ	psi
M	μ	mi	Z	ζ	zeta

12.5 Números primos

Os números primos apresentam um fascínio e um desafio perene para os matemáticos. Eles estão na base de muitos teoremas interessantes e conjecturas que se mostram muito difíceis de provar. Os números primos estão no cerne de algumas formas sofisticadas de criptografia e são usados para dar segurança em operações bancárias via *internet*. Os primeiros números primos são (ver o comando *primes*):

2	3	5	7	11	13	17	19	23	29
31	37	41	43	47	53	59	61	67	71
73	79	83	89	97	101	103	107	109	113
127	131	137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223	227	229
233	239	241	251	257	263	269	271	277	281
283	293	307	311	313	317	331	337	347	349
353	359	367	373	379	383	389	397	401	409
419	421	431	433	439	443	449	457	461	463
467	479	487	491	499	503	509	521	523	541
547	557	563	569	571	577	587	593	599	601
607	613	617	619	631	641	643	647	653	659
661	673	677	683	691	701	709	719	727	733
739	743	751	757	761	769	773	787	797	809
811	821	823	827	829	839	853	857	859	863
877	881	883	887	907	911	919	929	937	941
947	953	967	971	977	983	991	997	1009	1013
1019	1021	1031	1033	1039	1049	1051	1061	1063	1069
1087	1091	1093	1097	1103	1109	1117	1123	1129	1151
1153	1163	1171	1181	1187	1193	1201	1213	1217	1223
1229	1231	1237	1249	1259	1277	1279	1283	1289	1291
1297	1301	1303	1307	1319	1321	1327	1361	1367	1373
1381	1399	1409	1423	1427	1429	1433	1439	1447	1451
1453	1459	1471	1481	1483	1487	1489	1493	1499	1511
1523	1531	1543	1549	1553	1559	1567	1571	1579	1583
1597	1601	1607	1609	1613	1619	1621	1627	1637	1657
1663	1667	1669	1693	1697	1699	1709	1721	1723	1733
1741	1747	1753	1759	1777	1783	1787	1789	1801	1811
1823	1831	1847	1861	1867	1871	1873	1877	1879	1889
1901	1907	1913	1931	1933	1949	1951	1973	1979	1987
1993	1997	1999	2003	2011	2017	2027	2029	2039	2053
2063	2069	2081	2083	2087	2089	2099	2111	2113	2129
2131	2137	2141	2143	2153	2161	2179	2203	2207	2213
2221	2237	2239	2243	2251	2267	2269	2273	2281	2287
2293	2297	2309	2311	2333	2339	2341	2347	2351	2357
2371	2377	2381	2383	2389	2393	2399	2411	2417	2423
2437	2441	2447	2459	2467	2473	2477	2503	2521	2531
2539	2543	2549	2551	2557	2579	2591	2593	2609	2617

Fonte: `http://primes.utm.edu/lists/small/1000.txt`, acesso: 02/01/2018.

Existe um teorema que relaciona a quantidade de números primos e quantidade de números inteiros (ver Figura 12.2):

Teorema 12.5.1 Teorema dos números primos. Seja $\Pi(n)$ a função de contagem de números primos, que retorna o número de números primos entre 1 e n. Então, vale o limite:

$$\lim_{n\to\infty} \frac{\Pi(n)}{n/\ln n} = 1$$

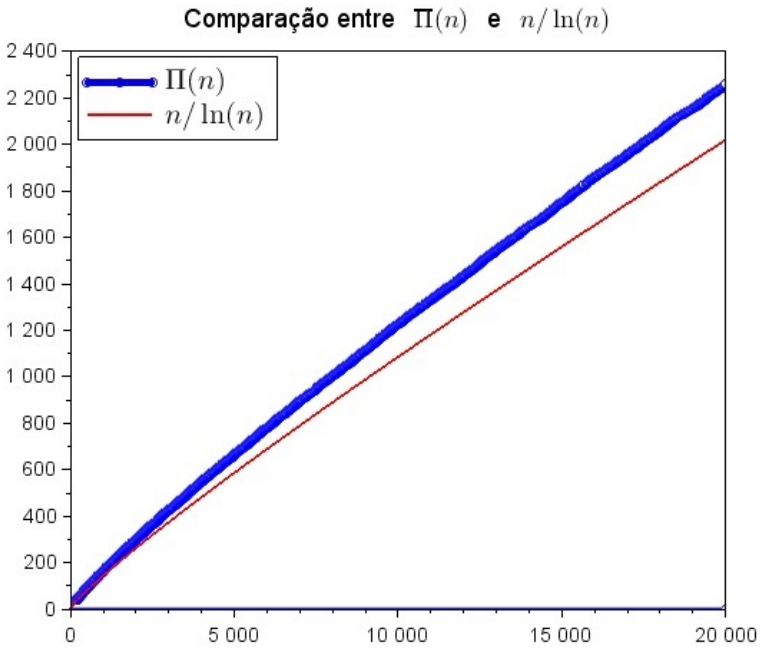


Figura 12.2: Número de números primos entre todos os inteiros. Somente para valores muito grandes ($> 10^{23}$) é que o limite indicado pelo Teorema 12.5.1 realmente se aproxima de 1.

12.6 Alguns *Links* interessantes

- Associação Brasileira de Estatística (ABE) - `http://www.redeabe.org.br/site/`
- Association for Computing Machinery (ACM) - `https://www.acm.org/`
- American Mathematical Society (AMS) - `http://www.ams.org/home/page`
- London Mathematical Society (LMS) - `https://www.lms.ac.uk/`
- Soc. Brasileira de Matemática Aplicada e Computacional - `http://www.sbmac.org.br/`
- Soc. Bras. de Educação Matemática (SBEM) - `http://www.sbembrasil.org.br/sbembrasil/`
- MacTutor History of Mathematics archive - `http://www-history.mcs.st-and.ac.uk/`
- Instituto Nacional de Matemática Pura e Aplicada (IMPA) - `https://impa.br/`
- Departamento de Matemática da UFC - `http://www.mat.ufc.br/`
- *Software* Maxima (computação simbólica) - `https://sourceforge.net/projects/maxima/`
- Wolfram Alpha - `https://www.wolframalpha.com/`

12.7 Convertendo um problema ou algoritmo em código Scilab

Dado um problema de método numérico, em geral, é fácil encontrar um algoritmo que consiga resolvê-lo. E, de posse do algoritmo, é relativamente direto converter em um código Scilab (ou em outra linguagem científica).

Resolvendo uma equação de segundo grau. Vejamos um exemplo simples inicialmente: dados "a", "b" e "c" da equação do segundo grau $ax^2+bx+c=0$, encontrar as suas raízes usando a fórmula de Báskara. Podemos escrever o seguinte algoritmo:

```
Mostre('Entre com a, b e c de uma equação do 2o. grau:')
Entre com 'a'
Entre com 'b'
Entre com 'c'
Calcule: delta = b*b - 4*a*c
Calcule: rdelta = raiz(delta)
Calcule: x1 = (-b + rdelta)/(2*a)
Calcule: x2 = (-b - rdelta)/(2*a)
Mostre('As raízes são', x1, x2)
```

Um possível código Scilab é:

```
clc;
disp('Entre com a, b e c de uma equação do 2o. grau:');
a = input('Entre com a: ');
b = input('Entre com b: ');
c = input('Entre com c: ');
delta = b*b - 4*a*c;
rdelta = sqrt(delta);
x1 = (-b+rdelta)/(2*a);
x2 = (-b-rdelta)/(2*a);
disp([x1, x2],'Raízes:');
```

Para este exemplo simples, temos praticamente uma tradução literal de cada linha do algoritmo por uma linha de código Scilab. Entretanto, usando os recursos do Scilab (comando "roots"), podemos ter um programa mais compacto (e menos fácil de entender):

```
clc;
disp('Entre com a, b e c de uma equação do 2o. grau:');
a = input('Entre com a: ');
b = input('Entre com b: ');
c = input('Entre com c: ');
x = roots([a,b,c]);
disp([x1, x2],'Raízes:');
```

Um exemplo de execução deste código:

```
 Entre com a, b e c de uma equação do 2o. grau:
Entre com a: 2
Entre com b: 3
```

```
Entre com c: 4

 Raízes:

  - 1.1666667 + 0.7993053i  - 1.1666667 - 0.7993053i
```

Nos códigos acima não temos uma crítica em relação ao valor de "a". Se "a" for nulo teremos uma divisão por zero e um erro. Uma possível solução deste problema em potencial é testar o valor de "a" e só prosseguir quando "a" for diferente de zero. O novo código incluindo esse teste é:

```
clc;
disp('Entre com a, b e c de uma equação do 2o. grau:');
a = input('Entre com a: ');
while a==0
    a = input('Entre com a: ');
end
b = input('Entre com b: ');
c = input('Entre com c: ');
x = roots([a,b,c]);
disp([x1, x2],'Raízes:');
```

Encontrar uma função cúbica. Vejamos um segundo exemplo de problema convertido em código Scilab. Considere a função

$$f(x) = \frac{\cos(x)}{e^{-x}+1}$$

Desejamos encontrar uma função polinomial de terceiro grau $p(x) = ax^3 + bx^2 + cx + d$ que se aproxima da função $f(x)$ no intervalo $0 \leq x \leq 3$. Para resolver esse problema, podemos calcular a função $f(x)$ nos pontos indicados na Tabela 12.3.

Tabela 12.3: Tabela para função $f(x)$.

x	$f(x)$
0,0	0,5
1,0	0,3949926
2,0	- 0,3665409
3,0	- 0,9430412

Logo, podemos equacionar o seguinte sistema:

$$\begin{aligned} d &= 0,50 \\ a+b+c+d &= 0,3949926 \\ 8a+4b+2c+d &= -0,3665409 \\ 27a+9b+3c+d &= -0,9430412 \end{aligned}$$

Para resolver o sistema acima, podemos usar o comando "inv". O código Scilab, incluindo os gráficos das funções $f(x)$ e $p(x)$, é:

```
close; clc;  // fechando alguma janela aberta e limpando o console

// função f(x) no intervalo [0, 3]:
x = 0:0.01:3;
f = cos(x)./(exp(-x)+ 1);
plot(x,f);  // gráfico de f(x)

x = [0, 1, 2, 3];   // pontos de interesse
f = cos(x)./(exp(-x)+ 1);  // valor da função nos pontos indicados
plot(x,f,'ro');  // destacando os pontos de interesse

M = [0 0 0 1;1 1 1 1; 8 4 2 1; 27 9 3 1]; // Matriz com os valores de x
d = f';
c = inv(M)*d // Solução do sistema.

// Verificando:
x = 0:0.01:3;
p = c(1)*x.*x.*x + c(2)*x.*x + c(3)*x + c(4);  // função p(x)
plot(x,p,'m');  // sobrepondo os gráficos
legend('f(x)','pontos','p(x)');
```

Os coeficientes calculados da função cúbica $p(x)$ são: $a = 0,1402599$; $b = -0,7490428$; $c = 0,5037755$ e $d = 0,50$. Na Figura 12.3, temos os gráficos de $f(x)$ e $p(x)$.

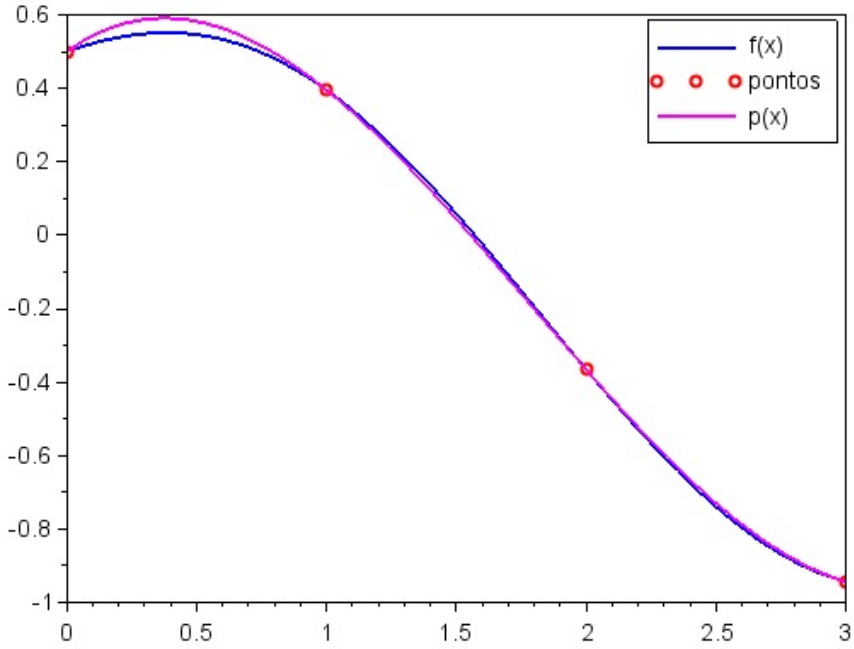


Figura 12.3: Exemplo de interpolação por função cúbica.

12.8 Métodos RK-5

No capítulo sobre solução de equações diferenciais, estudamos até o RK-4. Apresentamos, agora, um exemplo de código Scilab que implementa dois métodos de Runge-Kutta de quinta ordem. O código abaixo soluciona a equação

$$y' = -\frac{1}{2}y - 2e^{-x/2}\,\mathrm{sen}(2x)$$

cuja solução analítica é $y = e^{-x/2}\cos(2x)$. Como valor inicial nós temos $x_0 = 0$ e $y_0 = 1$, o passo é $h = 1/5$ e calculamos até o valor $x_f = 10$. Comparamos os resultados das soluções usando RK-4 e RK-5 nas figuras 12.4, 12.5 e 12.6. O RK-5 é cerca de $\times 1000$ ou $\times 100$ (RK-Butcher) mais preciso, para este exemplo, que o RK-4. Código Scilab:

```
// função:
function fy = fi(x)
    fy = exp(-x/2).*cos(2*x);
endfunction
// equação diferencial:
function fx = ff(x,y)
    fx = -0.5*y -2*exp(-x/2).*sin(2*x);
endfunction
clc;

/// Constantes:
a21 = 1/5; a31 = 3/40; a32 = 9/40;  a41 = 44/45;
a42 = -56/15; a43 = 32/9; a51 = 19372/6561; a52 = -25360/2187;
a53 = 64448/6561; a54 = -212/729; a61 = 9017/3168; a62 = -355/33;
a63 = 46732/5247; a64 = 49/176; a65 = -5103/18656; a71 = 35/384;
a73 = 500/1113; a74 = 125/192; a75 = -2187/6784; a76 = 11/84;
c2 = 1/5; c3 = 3/10; c4 = 4/5;  c5 = 8/9; c6 = 1; c7 = 1;

// Parâmetros:
m = 50; b0 = 10; a0 = 0; y0 = 1;
h = (b0 - a0)/m; xt = a0; yt = y0; yt4 = yt;  y6=yt4; yf = y6; y5=yf;
vy5 = ones(1,m); vy4 = vy5; vx = 0*vy4; v5=vy5;

for k = 1:m-1
    x = xt; y = yt; k1 = h*ff(x, y); // avaliar ff(x, y)
    x = xt + c2*h; y = yt + a21*k1; k2 = h*ff(x, y);
    x = xt + c3*h; y = yt + a31*k1 + a32*k2; k3 = h*ff(x, y);
    x = xt + c4*h; y = yt + a41*k1 + a42*k2 + a43*k3;
k4 = h*ff(x, y);
    x = xt + c5*h; y = yt + a51*k1 + a52*k2 + a53*k3 + a54*k4;
k5 = h*ff(x, y);
    x = xt + c6*h; y = yt + a61*k1 + a62*k2 + a63*k3 + a64*k4 + a65*k5;
k6 = h*ff(x, y);
    x = xt + c7*h; y = yt + a71*k1 + a73*k3 + a74*k4 + a75*k5 + a76*k6;
```

```
    yt = yt + a71*k1 + a73*k3 + a74*k4 + a75*k5 + a76*k6;

    vy5(k+1) = yt;  // RK-5

    k1 = ff(xt, yt4); // avaliar ff(x, y)
    k2 = ff(xt+h/2, yt4+k1*h/2);
    k3 = ff(xt+h/2, yt4+h*k2/2);
    k4 = ff(xt+h, yt4+h*k3);
    yt4 = yt4 + h*(k1 + 2*(k2 + k3) + k4)/6;
    vy4(k+1) = yt4;  // RK-4

    k1 = ff(xt, y5); // avaliar ff(x, y)
    k2 = ff(xt+h/4, y5+k1*h/4);
    k3 = ff(xt+h/4, y5+h*(k1+k2)/8);
    k4 = ff(xt+h/2, y5+h*(k3-k2/2));
    k5 = ff(xt+3*h/4, y5+h*(3*k1+9*k4)/16);
    k6 = ff(xt+h, y5+h*(-3*k1+2*k2+12*k3-12*k4+8*k5)/7);
    y5 = y5 + h*(7*(k1+k6) + 32*(k3 + k5) + 12*k4)/90;
    v5(k+1) = y5;  // RK-Butcher

    xt = xt + h; // atualizando x
    vx(k+1) = xt;
end;

// Solução teórica:
yttt = fi(vx);
// Gráficos:
close; close; close; plot(vx,yttt,vx,vy4,vx,vy5,vx,v5,'k');
legend('Solução teórica','Sol. RK-4','Sol. RK-5','RK-Butcher');
xlabel('tempo'); ylabel('y(t)');

// Erros absolutos:
ep = 1e-20;  // evitar o cálculo de log de zero
e4 = abs(yttt-vy4); e4l = log10(ep + e4);
e5 = abs(yttt-vy5); e5l = log10(ep + e5);
e8 = abs(yttt-v5); e8l = log10(ep + e8);

figure; plot(vx,e4l,vx,e5l,vx,e8l);
legend('Erro RK-4','Erro RK-5','RK-Butcher');
xlabel('tempo'); ylabel('Erro absoluto');

figure; plot(vx,e4,vx,e5*1000,vx,e8*100);
legend('Erro RK-4','Erro RK-5 x 1000','RK-Butcher x 100');
xlabel('tempo'); ylabel('Erro absoluto');
```

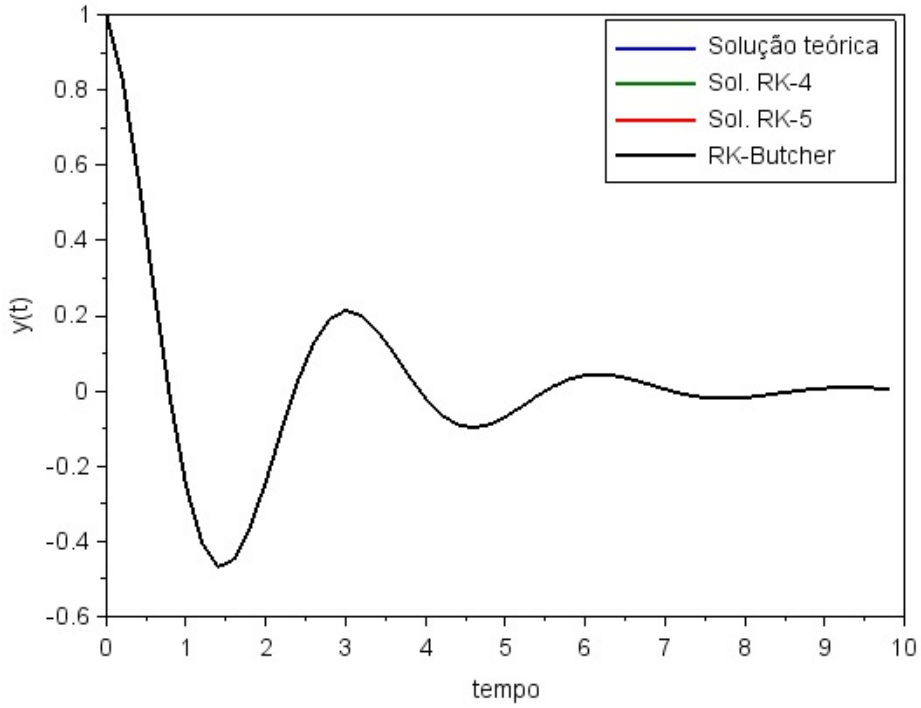


Figura 12.4: Soluções teóricas e numéricas (RK-4, RK-5 e RK-Butcher).

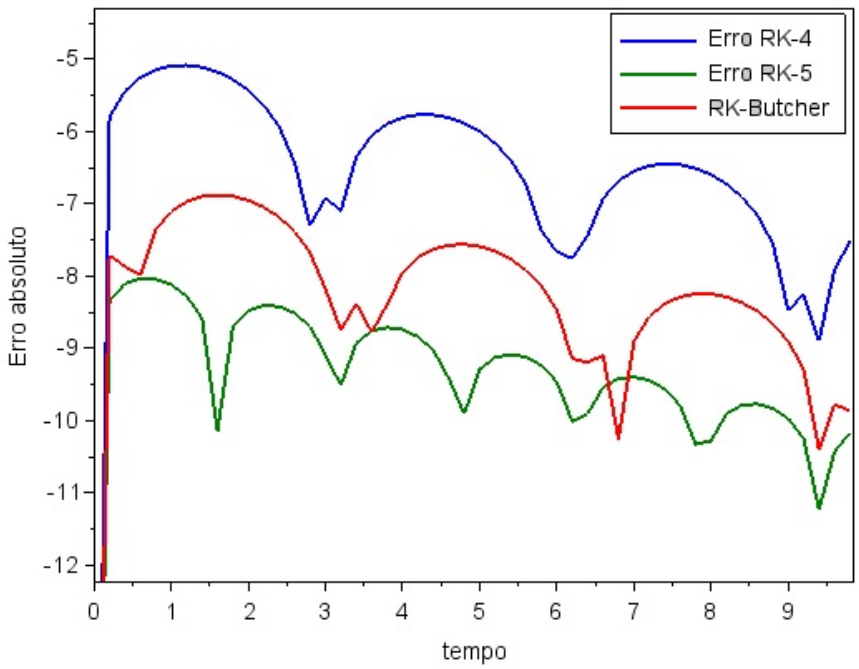


Figura 12.5: Erros absolutos do RK-4, RK-5 e do RK-Butcher em escala logarítmica.

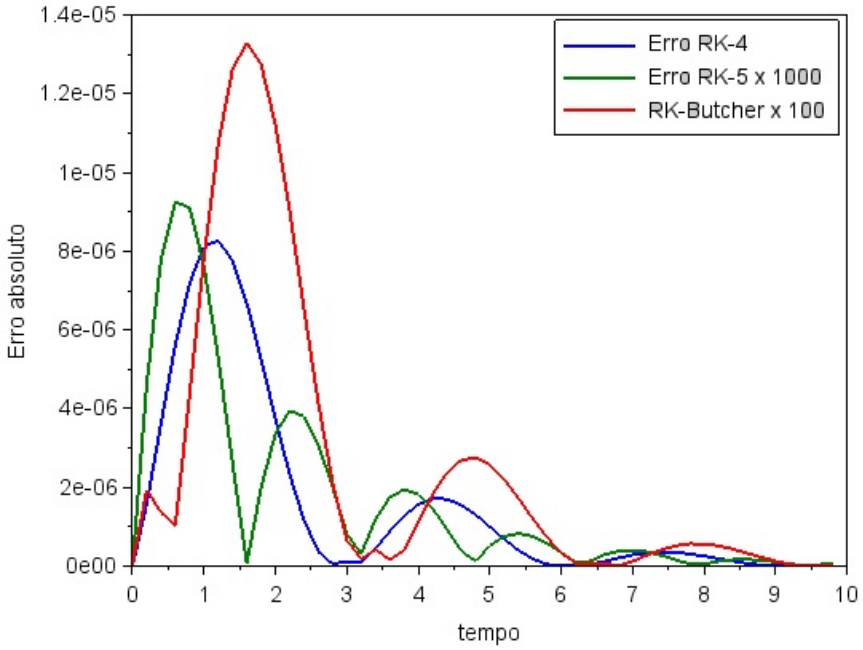


Figura 12.6: Erros absolutos do RK-4, RK-5 e do RK-Butcher em escala linear.

12.9 Tabelas

Tabela 12.4: Tabela de logaritmo decimal.

x	$\log_{10}(x)$	x	$\log_{10}(x)$	x	$\log_{10}$
1	0,000000	11	1,041393	30	1,477121
2	0,301030	12	1,079181	40	1,602060
3	0,477121	13	1,113943	50	1,698970
4	0,602060	14	1,146128	60	1,778151
5	0,698970	15	1,176091	70	1,845098
6	0,778151	16	1,204120	80	1,903090
7	0,845098	17	1,230449	90	1,954243
8	0,903090	18	1,255273	100	2,000000
9	0,954243	19	1,278754	110	2,041393
10	1,000000	20	1,301030	120	2,079181

Dois exemplos de uso da Tabela 12.4:

$$\begin{aligned}\log_{10}(56) &= \log_{10}(14\times 4)\\ &= \log_{10}(14)+\log_{10}(4)\\ &= 1,146128+0,602060\\ &= 1,748188\end{aligned}$$

$$\begin{aligned}\log_{10}(220) &= \log_{10}(2\times 11\times 10)\\ &= \log_{10}(2)+\log_{10}(11)+\log_{10}(10)\\ &= 0,301030+1,041393+1,000\\ &= 2,342423\end{aligned}$$

Tabela 12.5: Tabela de seno, cosseno e tangente para alguns ângulos.

θ (graus)	sen(θ)	cos(θ)	tan(θ)
0,0	0,000000	1,000000	0,000000
2,5	0,043619	0,999048	0,043661
5,0	0,087156	0,996195	0,087489
7,5	0,130526	0,991445	0,131652
10,0	0,173648	0,984808	0,176327
12,5	0,216440	0,976296	0,221695
15,0	0,258819	0,965926	0,267949
17,5	0,300706	0,953717	0,315299
20,0	0,342020	0,939693	0,363970
22,5	0,382683	0,923880	0,414214
25,0	0,422618	0,906308	0,466308
27,5	0,461749	0,887011	0,520567
30,0	0,500000	0,866025	0,577350
32,5	0,537300	0,843391	0,637070
35,0	0,573576	0,819152	0,700208
37,5	0,608761	0,793353	0,767327
40,0	0,642788	0,766044	0,839100
42,5	0,675590	0,737277	0,916331
45,0	0,707107	0,707107	1,000000
47,5	0,737277	0,675590	1,091309
50,0	0,766044	0,642788	1,191754
52,5	0,793353	0,608761	1,303225
55,0	0,819152	0,573576	1,428148
57,5	0,843391	0,537300	1,569686
60,0	0,866025	0,500000	1,732051
62,5	0,887011	0,461749	1,920982
65,0	0,906308	0,422618	2,144507
67,5	0,923880	0,382683	2,414214
70,0	0,939693	0,342020	2,747477
72,5	0,953717	0,300706	3,171595
75,0	0,965926	0,258819	3,732051
77,5	0,976296	0,216440	4,510709
80,0	0,984808	0,173648	5,671282
82,5	0,991445	0,130526	7,595754
85,0	0,996195	0,087156	11,430052
87,5	0,999048	0,043619	22,903766
90,0	1,000000	0,000000	∞

Tabela 12.6: Tabela das funções $Q(x)$ e $erfc(x)$.

x	$Q(x)$	$erfc(x)$
0,0	5,000000e-01	1,000000e+00
0,2	4,207403e-01	7,772974e-01
0,4	3,445783e-01	5,716076e-01
0,6	2,742531e-01	3,961439e-01
0,8	2,118554e-01	2,578990e-01
1,0	1,586553e-01	1,572992e-01
1,2	1,150697e-01	8,968602e-02
1,4	8,075666e-02	4,771488e-02
1,6	5,479929e-02	2,365162e-02
1,8	3,593032e-02	1,090950e-02
2,0	2,275013e-02	4,677735e-03
2,2	1,390345e-02	1,862846e-03
2,4	8,197536e-03	6,885139e-04
2,6	4,661188e-03	2,360344e-04
2,8	2,555130e-03	7,501319e-05
3,0	1,349898e-03	2,209050e-05
3,2	6,871379e-04	6,025761e-06
3,4	3,369293e-04	1,521993e-06
3,6	1,591086e-04	3,558630e-07
3,8	7,234804e-05	7,700393e-08
4,0	3,167124e-05	1,541726e-08
4,2	1,334575e-05	2,855494e-09
4,4	5,412544e-06	4,891710e-10
4,6	2,112455e-06	7,749600e-11
4,8	7,933282e-07	1,135214e-11
5,0	2,866516e-07	1,537460e-12
5,2	9,964426e-08	1,924906e-13
5,4	3,332045e-08	2,227679e-14
5,6	1,071759e-08	2,382836e-15
5,8	3,315746e-09	2,355589e-16
6,0	9,865876e-10	2,151974e-17
6,2	2,823158e-10	1,816676e-18
6,4	7,768848e-11	1,417080e-19
6,6	2,055789e-11	1,021325e-20
6,8	5,230958e-12	6,800861e-22
7,0	1,279813e-12	4,183826e-23

Tabela 12.7: Tabela de logaritmo natural.

x	$\ln(x)$	x	$\ln(x)$	x	$\ln(x)$
1	0,000000	36	3,583519	71	4,262680
2	0,693147	37	3,610918	72	4,276666
3	1,098612	38	3,637586	73	4,290459
4	1,386294	39	3,663562	74	4,304065
5	1,609438	40	3,688879	75	4,317488
6	1,791759	41	3,713572	76	4,330733
7	1,945910	42	3,737670	77	4,343805
8	2,079442	43	3,761200	78	4,356709
9	2,197225	44	3,784190	79	4,369448
10	2,302585	45	3,806662	80	4,382027
11	2,397895	46	3,828641	81	4,394449
12	2,484907	47	3,850148	82	4,406719
13	2,564949	48	3,871201	83	4,418841
14	2,639057	49	3,891820	84	4,430817
15	2,708050	50	3,912023	85	4,442651
16	2,772589	51	3,931826	86	4,454347
17	2,833213	52	3,951244	87	4,465908
18	2,890372	53	3,970292	88	4,477337
19	2,944439	54	3,988984	89	4,488636
20	2,995732	55	4,007333	90	4,499810
21	3,044522	56	4,025352	91	4,510860
22	3,091042	57	4,043051	92	4,521789
23	3,135494	58	4,060443	93	4,532599
24	3,178054	59	4,077537	94	4,543295
25	3,218876	60	4,094345	95	4,553877
26	3,258097	61	4,110874	96	4,564348
27	3,295837	62	4,127134	97	4,574711
28	3,332205	63	4,143135	98	4,584967
29	3,367296	64	4,158883	99	4,595120
30	3,401197	65	4,174387	100	4,605170
31	3,433987	66	4,189655	101	4,615121
32	3,465736	67	4,204693	102	4,624973
33	3,496508	68	4,219508	103	4,634729
34	3,526361	69	4,234107	104	4,644391
35	3,555348	70	4,248495	105	4,653960

A função Gama (representada por Γ) pode ser definida por:

$$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx$$

Para n inteiro positivo a função $\Gamma(n)$ apresenta a seguinte propriedade:

$$\Gamma(n+1) = n! \text{ ou } \Gamma(n) = (n-1)!$$

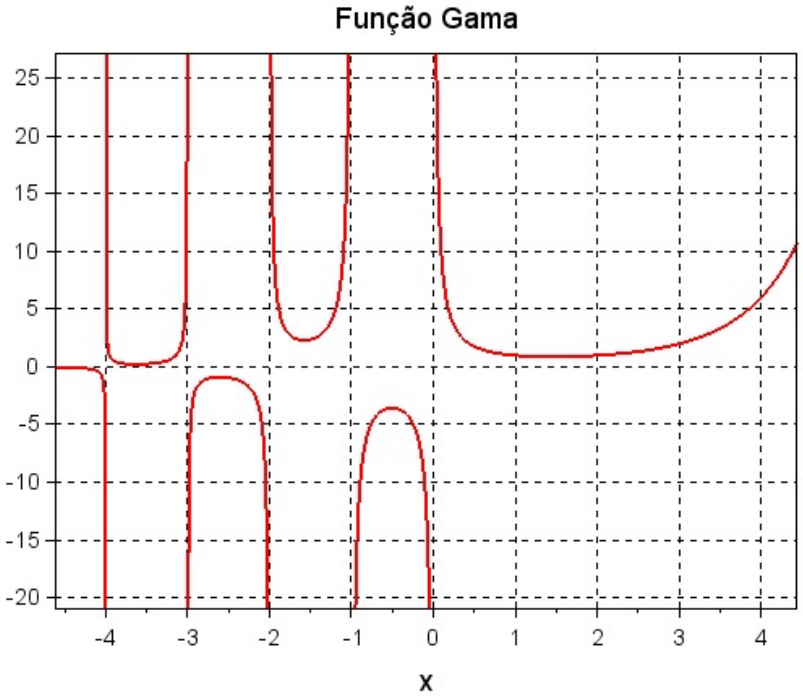


Figura 12.7: Função $\Gamma(x)$.

Tabela 12.8: Pequena tabela fatorial.

n	$n!$	n	$n!$
0	1	7	5040
1	1	8	40320
2	2	9	362880
3	6	10	3628800
4	24	11	39916800
5	120	12	479001599
6	720	13	1932053503

Uma aproximação para o fatorial de $n!$:

$$n! \cong \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \frac{1}{12n} + \frac{1}{288n^2} - \frac{139}{51840n^3} - \frac{571}{2488320n^4} + \cdots\right)$$

13. Referências

Se eu vi mais longe, foi por estar sobre ombros de gigantes.
Sir Isaac Newton (25 de dezembro 1642, 31 de março de 1727).

ABRAMOWITZ, M., STEGUN, I. A. (Eds.). **Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables**, 9th printing. New York: Dover, 1972. Também disponível em `http://people.math.sfu.ca/~cbm/aands/intro.htm`, data de acesso: 07 de outubro de 2017.

BOYCE, William E.; DIPRIMA, Richard C. **Equações diferenciais elementares e problemas de valores de contorno**. 7.ed. Rio de Janeiro, RJ: LTC, 2002. 416 p. ISBN 85-216-1312-1.

BRUDEN, Richard L., FAIRES, J. Douglas. Análise Numérica. Tradução da 8ª edição norte-americana. São Paulo: CENGAGE Learning, 2008. 721 p. ISBN 9788522106011

BURIAN, Reinaldo; LIMA, Antonio Carlos de; HETEM JUNIOR, Annibal. Cálculo numérico. Rio de Janeiro: LTC, 2013. 153 p. (Fundamentos de Informática). ISBN 9788521615620.

BUTHCER, J. C. (1964). On Runge-Kutta processes of high order. Journal of the Australian Mathematical Society, 4(2), 179-194. doi:10.1017/S1446788700023387. Acesso: 04/01/2018.

CAMPOS FILHO, F. F. **Algoritmos Numéricos**. 2. ed. Rio de Janeiro: LTC, 2013.

CHAPRA, Steven C.; ALÍPIO, Rafael Silva. **Métodos numéricos aplicados com matlab para engenheiros e cientistas**. 3. ed. Porto Alegre, RS: AMGH, 2013. 655 p., il. ISBN 9788580551761.

CHAPRA, Steven C.; CANALE, Raymond P. **Métodos numéricos para engenharia**. 5. ed. São Paulo, SP: McGraw-Hill, 2014. 809 p. ISBN 9788586804878.

DENNIS JR., J. E.; SCHNABEL, Robert B. **Numerical Methods for Unconstrained Optimization and Nonlinear Equations**. SIAM's Classics in Applied Mathematics. ISBN: 978-0-89871-364-0.

DORMAND, J.R., PRINCE, P.J. **A family of embedded Runge-Kutta formulae**. Journal of Computational and Applied Mathematics, vol. 6, Issue 1, 1980, pages 19-26, ISSN 0377-0427. Disponível em `https://doi.org/10.1016/0771-050X(80)90013-3`. Acesso 19 de janeiro de 2018.

DUFF, I. S; ERISMAN, A. M; REID, J. K. **Direct Methods for Sparse Matrices**. Clarendon Press, New York, NY, USA , 1989. ISBN: 0-19-853421-3

FELÍCIO, Luiz Carlos. **Modelagem da dinâmica de sistemas e estudo da resposta**. Segunda Edição - São Carlos: RiMa, 2010. 568 p. ISBN - 978-85-7656-169-9

FERNANDES, Edite M. da G. P. **Computação Numérica**. Universidade do Minho, 2. ed. 1997. Disponível em: <https://repositorium.sdum.uminho.pt/bitstream/1822/5828/1/livro.pdf>. Acesso: 04 de junho de 2017. ISBN 9729694419.

FRANCO, Neide Maria Bertoldi. **Cálculo Numérico**. [S.l.]: Pearson. 520 p. Disponível em: <http://ifce.bv3.digitalpages.com.br/users/publications/9788576050872>. Acesso: 30 de maio de 2017. ISBN 9788576050872.

HAIRER, E. **A Runge-Kutta method of order 10**. J. Inst. Math. Applics. 21 (1978) pp. 47-59.

HUMES, Ana F. P. de Castro et al. **Noções de cálculo numérico**. São Paulo: McGraw-Hill; 1984. 201 p.

IEEE Standard for Floating-Point Arithmetic. Sponsor Microprocessor Standards Committee of the IEEE Computer Society. IEEE-SA Standards Board. The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA. ISBN 978-0-7381-5752-8

ISAACSON, Eugene; KELLER, Herbert Bishop. **Analysis of numerical methods**. New York, USA: Dover Publications, 1996. 541 p. ISBN 0-486-68029-0.

ISAACSON, W. **Os Inovadores**: Uma biografia da revolução digital. Companhia das Letras, 2014. 568p. ISBN: 9788535925029.

GILAT, Amos. **MATLAB com aplicações em engenharia**. 2. ed. Porto Alegre: Bookman, 2006. 359 p. ISBN 9788540701861.

GILAT, Amos; SUBRAMANIAM, Vish. **Métodos numéricos para engenheiros e cientistas:** uma introdução com aplicações usando o MATLAB. Porto Alegre, RS: Bookman, 2008. 479p.

KLEE, H. **Simulation of Dynamic Systems with MATLAB and Simulink**. CR PRESS, Boca Raton, 2007. ISBN: 9781420044188.

LATHI, B. P. **Sinais e Sistemas Lineares**. 2.ed. Porto Alegre: Bookman, 2007. 856 p. ISBN: 8560031138.

LUTHER, H. A. An explicit sixth-order Runge-Kutta formula. Journal: Math. Comp. 22 (1968), 434-436. DOI: https://doi.org/10.1090/S0025-5718-68-99876-1.

MAIA, Miriam Lourenço et al. **Cálculo numérico com aplicações**. 2.ed. São Paulo, SP: Harbra, c1987. 367 p. ISBN 85-294-0089-5.

PRESS, William H. et al. **Métodos numéricos aplicados**: rotinas em C++. 3. ed. Porto Alegre, RS: Bookman, 2011. 1261 p. ISBN 9788577808861.

RUKHIN, A., et al. **A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications**. National Institute of Standards and Technology Special Publication 800-22, 131 pages (April 2010).

RUGGIERO, Márcia A. Gomes; LOPES, Vera L. da Rocha. **Cálculo numérico**: aspectos teóricos e computacionais. 2. ed. São Paulo, SP: Pearson Makron Books, 1996. 406 p. ISBN 978-85-346-0204-4.

SANTOS, Vitoriano Ruas de Barros. **Curso de cálculo numérico**. Rio de Janeiro, RJ: Livro Técnico, 1972. 256 p.

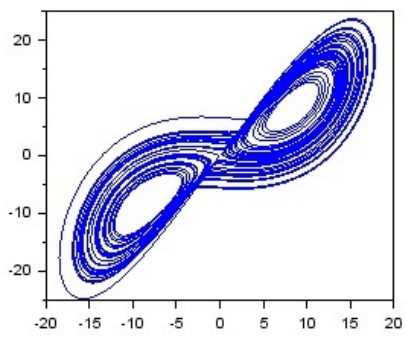
SHAMPINE, Lawrence F. **Some Practical Runge-Kutta Formulas**. Mathematics of Computation, American Mathematical Society, 46 (173): 135150, 1986. JSTOR 2008219.

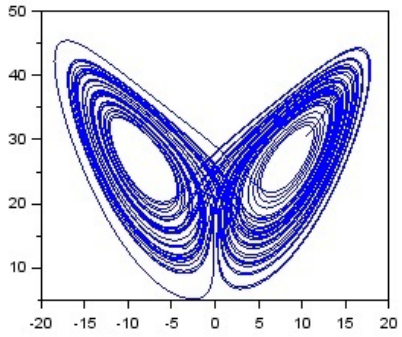
SPERANDIO, D.; MENDES, J. T.; SILVA, L. H. M. **Cálculo numérico**, 2ª edição. [S.l.]: Pearson. 360 p. Disponível em: <http://ifce.bv3.digitalpages.com.br/users/publications/9788543006536>. Acesso em: 12 set. 2017. ISBN 9788543006536.

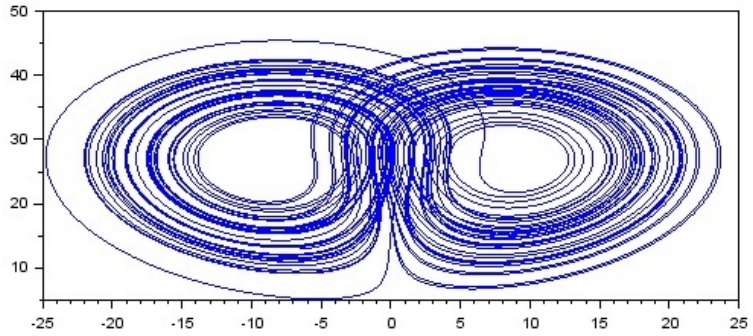
SPERANDIO, Décio. **Cálculo numérico**: características matemáticas e computacionais dos métodos numéricos. São Paulo, SP: Pearson Prentice Hall, 2006. 354p. ISBN 85-87918-74-5.

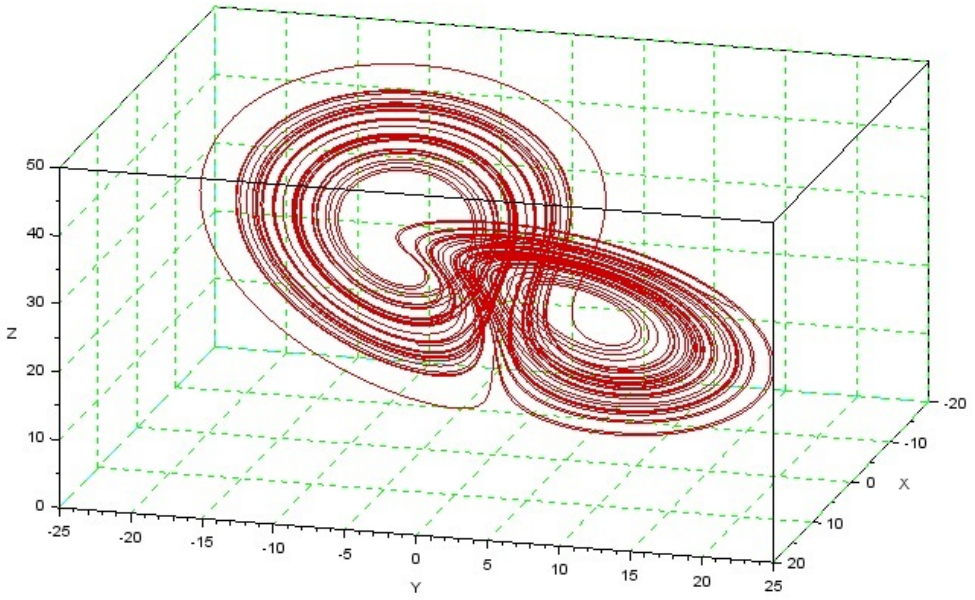
VOLCHAN, Sérgio B. **What Is a Random Sequence?** The American Mathematical Monthly, Vol. 109, 2002, pp. 4663.

ZILL, Dennis G.; CULLEN, Michael R. **Equações diferenciais**, v.2. 3. ed. São Paulo, SP: Pearson Makron Books, 2012. v.2. ISBN 9788534611411.


20
10
0
-10
-20
-20 -15 -10 -5 0 5 10 15 20


50
40
30
20
10
-20 -15 -10 -5 0 5 10 15 20


50
40
30
20
10
-25 -20 -15 -10 -5 0 5 10 15 20 25


Z
50
40
30
20
10
0
-25 -20 -15 -10 -5 0 5 10 15 20 25
Y
-20 -10 0 10 20
X

14. Sobre o Autor - currículo

TENHO formação técnica em telecomunicações (1988, antiga ETFCE - atual IFCE), graduação pela Universidade Federal do Ceará (1992), mestrado (1998, Lab. Linse) e doutorado (2008, Lab. GPqCom) pela Universidade Federal de Santa Catarina, todos em Engenharia Elétrica. Sou professor do Instituto Federal de Educação, Ciência e Tecnologia do Ceará desde 1994. Fui professor nos cursos de Eng. Eletrônica e Eng. de Telecomunicações da Universidade de Fortaleza (Unifor) de 2000 a 2003. Tenho experiência nas áreas de pesquisa e ensino em Engenharia com ênfase em filtros lineares, eletrônica, métodos numéricos e processamento digital de sinais. Ministro ou já ministrei aulas de métodos numéricos, sistemas lineares, eletricidade, eletrônica analógica, metodologia científica, monografia, sistema de comunicação, sistemas lineares, processos estocásticos e equações diferenciais. Também tenho experiência no ensino técnico de nível médio, além de orientação de iniciação científica e de mestrado. Na área administrativa já atuei como coordenador de curso de graduação (Eng. de Computação) como Chefe de Departamento do Departamento de Telemática (jan/2011 a fev/2013) do IFCE - Campus Fortaleza e coordenador do PPGET (Programa de Pós-Graduação em Engenharia de Telecomunicações). Atualmente, também sou professor nos programas de mestrado PPGET e PROFEPT (Programa de Pós-Graduação em Educação Profissional e Tecnológica). Mantenho um *blog* (`http://alfanumericus.blogspot.com.br/`) sobre temas diversos, incluindo simulação computacional e dicas para estudantes de graduação e mestrado, desde 2009. Sou professor titular do IFCE desde outubro de 2015 e tenho cerca de 10 orientações de mestrado concluídas (2020).

Composto e Impresso no Brasil
Impressão Sob Demanda

21 2236-0844
www.podeditora.com.br
atendimento@podeditora.com.br

2020