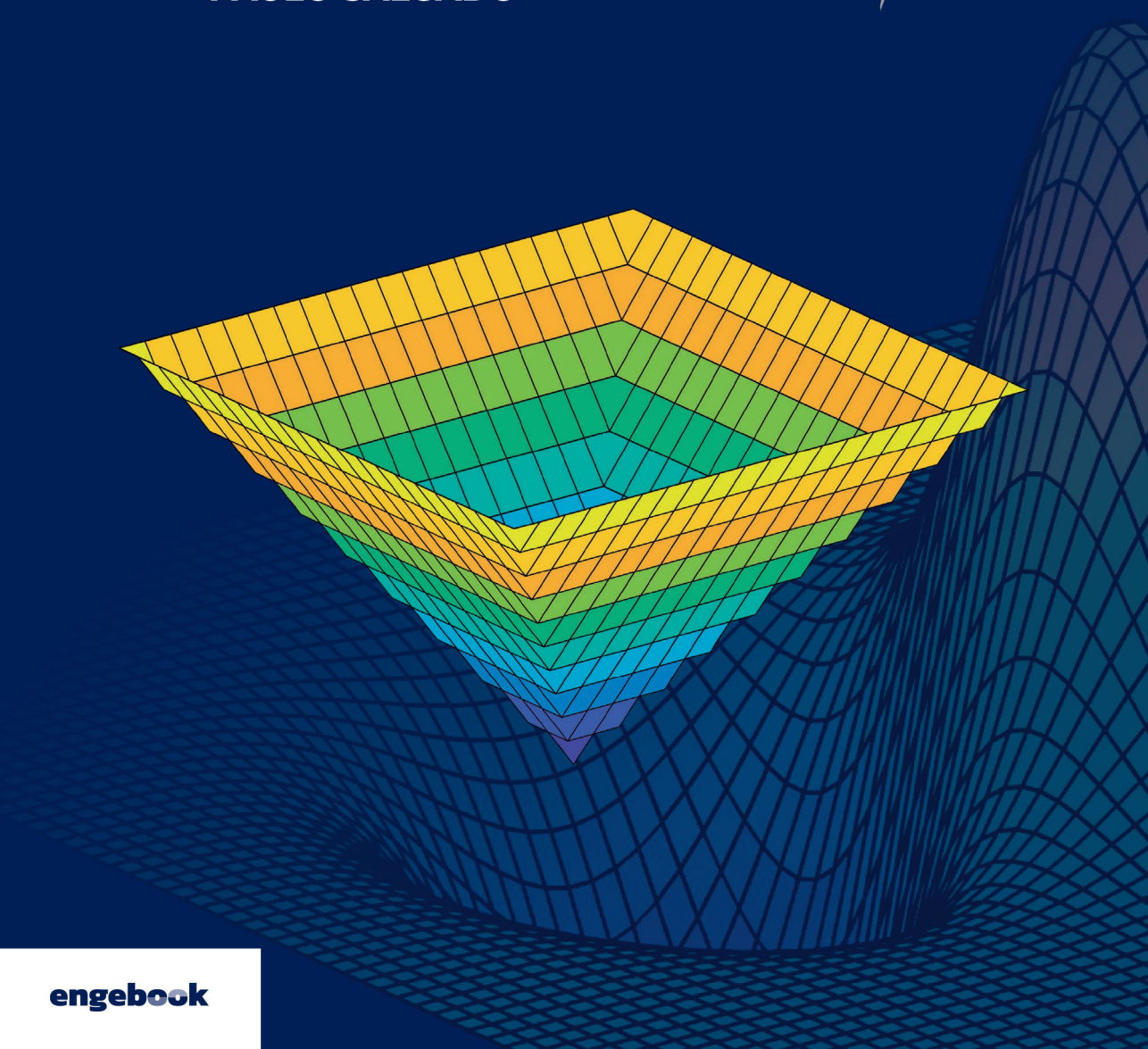
Introdução à Programação em

MATLAB

PAULO AFONSO
PAULO SALGADO

MATLAB® examples



engebook

Conteúdo

Prefácio

No mundo atual, onde se procura desenvolver, testar e validar soluções de engenharia de forma rápida, completa e integrada, mas também realizar análises complexas de dados, o engenheiro, o economista ou o investigador carecem, quase sempre, de uma linguagem de alto desempenho. Esta deve integrar facilidades de computação, visualização e programação, num ambiente fácil de usar, onde os problemas e as soluções são expressas em notação matemática comum. Neste quadro, torna-se mais fácil construir protótipos de modo mais eficiente e com um custo de desenvolvimento menor.

O Matlab™, desenvolvido pela MathWorks, é um laboratório matricial que permite realizar operações matemáticas matriciais e numéricas e que possui uma linguagem de programação proprietária. Esta plataforma permite ainda criar gráficos de funções de dados, implementar algoritmos, criar interfaces com o utilizador e permite a ligação com programas escritos em outras linguagens, incluindo C, C++, C#, Java, Fortran e Python. Um pacote adicional, o Simulink, adiciona a simulação gráfica multidomínio e desenho baseado num modelo para sistemas dinâmicos e embebidos. O uso comum do aplicativo Matlab envolve o uso da Janela de Comandos como um "Shell" matemático interativo ou a execução de arquivos de texto, contendo o código Matlab. Porém, o Matlab é construído para lidar com linguagem de script Matlab e para criar funções executáveis.

O GNU Octave também pode ser usado como ferramenta de cálculo numérico. Apresenta uma linguagem de programação de alto nível, como o Matlab, que facilita a resolução numérica de problemas lineares e não-lineares. É semelhante e é compatível com o Matlab que, por fazer parte do Projeto GNU, é um pacote de software livre sob os termos da Licença Pública Geral GNU. Existem outras alternativas gratuitas para o Matlab: o Scilab e o FreeMat. Tanto o Matlab como o Octave são sistemas interativos, cujo tipo de dados básico é o "array" ou matriz.

O Octave possui recursos comuns com o Matlab. Em ambos, o tipo de dados principal são as matrizes e têm suporte embebido para números complexos. Possuem funções matemáticas e bibliotecas incorporadas prontas a usar. Estas estão organizadas em "Toolboxes" que cobrem as necessidades de um grande leque de áreas de conhecimento, nas Engenharias (incluindo-se a Bioengenharia), Biologia, Economia e Matemática. Adicionalmente, aceitam funções criadas pelo utilizador.

Devido às capacidades referidas, estas plataformas são consideradas como um bom instrumento para rapidamente um aluno de Engenharia usar ferramentas computacionais de extrema capacidade e utilidade, mas também para dar início à aprendizagem de técnicas de programação. Acresce-se ainda o facto de as capacidades de programação se estenderem ao conceito de programação por objetos.

Dada a forte semelhança da estrutura principal destes dois programas, sempre que nos referirmos a um deles, assume-se a sua compatibilidade com o outro.

A estrutura deste livro foi orientada para que as aprendizagens e as competências sejam consolidadas através da realização de exemplos, mas também pela apresentação de problemas de estudo, sendo recorrente as respetivas sugestões de resolução.

Paulo Afonso e Paulo Salgado.

Capítulo 1

Sistemas Numéricos

1.1 Sistemas Numéricos

Os sistemas numéricos são usados para descrever uma quantidade ou representar uma certa informação, através do recurso a um número limitado de símbolos. Os sistemas mais convencionais são o sistema decimal e o binário, mas também o octal e o hexadecimal (Hex). No sistema decimal, usam-se dez símbolos, porventura pelo facto de o ser humano possuir dez dedos (dígitos) nas suas mãos e de frequentemente os usar para proceder à contagem. Desenvolvido na Índia por volta de 400 a.C., e depois adotado pelos Árabes, o sistema numérico hindu-arábico passou a ser o mais usado na Europa, a partir do século XIII. Este usa um símbolo para o zero, sendo os símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9, pela ordem exposta, usados para representar a quantidade crescente de zero a nove. Já o binário usa apenas os símbolos 0 e 1, enquanto que o octal usa os símbolos de 0 a 7, e o hexadecimal os símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, F. Em cada um destes sistemas, apesar de possuírem um conjunto limitado de símbolos, é possível representar qualquer quantidade. Por esse motivo as nossas calculadoras, auxiliares na operação de números em representação decimal, contêm dez teclas, identificadas com os símbolos 0 a 9. Por sua vez, os sistemas digitais eletrónicos, dos quais se incluem os microprocessadores, são apenas capazes de intrinsecamente lidar com representações digitais, traduzíveis por dois estados distintos de operação.

Quando adicionamos um valor a mais a uma determinada quantia, esta é expressa por um determinado número, sendo a nova quantidade representada com recurso a dígitos imediatamente a seguir na lista de dígitos. Assim, se num cesto com 0 maçãs, houver a inclusão de uma maçã nesse mesmo cesto, o número de maçãs passa a representar-se pelo número 1, sendo este o dígito imediatamente a seguir. Com sucessivas adições de maçãs, ter-se-á: ...2, 3, 4, 5, 6, 7, 8, 9... Assim que é atingido o último símbolo da lista, este é substituído pelo primeiro símbolo da lista (0) e um novo dígito (1) é colocado à esquerda do dígito existente. Em suma, para representar um valor maior e imediatamente a seguir ao 9, esgotados que estão os dígitos da lista, escrevemos 10, que significa uma unidade de dezenas e zero unidades. O mesmo ocorrerá no incremento do número 99, estando esgotados os dígitos das

i	Operação
0	2018 | 2
1	-2018 | 1009 | 2
2	**0** -1008 | 504 | 2
3	**1** -504 | 252 | 2
4	**0** +252 | 126 | 2
5	**0** -126 | 63 | 2
6	**0** -62 | 31 | 2
7	**1** -30 | 15 | 2
8	**1** -14 | 7 | 2
9	**1** -6 | 3 | 2
10	**1** -2 | **1**
	1

Restos da divisão sucessiva

nº		10	9	8	7	6	5	4	3	2	1
resto	1	1	1	1	1	1	0	0	0	1	0

Tomando os restos das sucessivas divisões pela ordem referida, o número binário será: 11111100010_2. Assim 2018 (DEC) $= 11111100010_2$

Do mesmo modo, o número 2018 tem uma representação na base hexadecimal de 7E2.

Assim, 2018 $= 11111100010_2 = 7E2_{16}$.

Seja i o índice do número da divisão sucessiva, com $i = 0, 1, ...$ Para a i-ésima divisão realizada num número n_{i-1} (dividendo da última operação de divisão) com divisor b, tem-se como resultado o dividendo n_i e resto d_{i-1}, sendo que $n_{i-1} = n_i \times b + d_{i-1}$. Naturalmente a divisão só tem lugar se $n_{i-1} \geq b$. Caso contrário, ter-se-á $d_N = n_N$ e o processo de divisões recursivo é interrompido, onde $i = N + 1$ é o índice desta última operação. No início do processo, temos $n_0 = n$, o número a converter de base. Por iterações sucessivas, ter-se-á:

$$n_0 = d_0 + n_1 \times b$$
$$n_1 = d_1 + n_2 \times b \ \rightarrow \ n = d_0 + (d_1 + n_2 \times b) \times b$$
$$n_2 = d_2 + n_3 \times b \ \rightarrow \ n = d_0 + (d_1 + (d_2 + n_3 \times b) \times b) \times b$$
$$\vdots$$
$$n_{N-1} = d_{N-1} + n_N \times b \rightarrow n = d_0 + (d_1 + (d_2 + (... (d_{N-1} + n_N \times b)) \times b) \times b) \times b$$
$$n_N = d_N \rightarrow n = d_0 + (d_1 + (d_2 + (... (d_{N-1} + d_N \times b)) \times b) \times b) \times b$$

Deste modo, um valor n é representado na base b pelos dígitos $\left[d_N \quad \cdots \quad d_2 \quad d_1 \quad d_0\right]_b$, tendo em conta que:

$$n_b = \left[d_N \quad \cdots \quad d_2 \quad d_1 \quad d_0\right]_b = d_0 + b\,(d_1 + b\,(d_2 + b\,(d_3 + b \cdots (d_{N-1} + d_N b) ...)))$$

Capítulo 2

Valores e Funções Lógicas

2.1 Álgebra de Bool

Álgebra de Bool é o formalismo usado para o tratamento sistemático da lógica, a qual C. E. Shannon (1938) aplicou para caracterizar as propriedades de circuitos elétricos de comutação, cujo estado é representável por dois valores. Estes podem ser {V,F} (verdadeiro ou falso), {H,L} (high ou low), {1,0} (um ou zero) ou ainda "on" (ligado) ou "off" (desligado). Neste texto, usaremos indiferentemente cada um eles, com destaque para a notação {1,0}, a qual também é utilizada em eletrónica digital. Nestes casos, as variáveis lógicas podem ter apenas 2 valores (correspondentes a 2 estados) e o número de estados que uma função lógica pode assumir será igualmente finito. Deste modo, podemos descrever as funções Booleanas utilizando tabelas, também designadas por tabelas de verdade, sendo que nelas estão listadas todas as combinações de valores que as variáveis de entrada podem assumir e os correspondentes valores da função que representam as saídas.

Considere, a título de exemplo, um circuito lógico ou sistema com a saída resultante da combinação das entradas rotuladas como "a"e "b", como ilustra a figura 2.1. Existem "quatro"combinações de entrada possíveis de "LIGADO"e "DESLIGADO"para as duas entradas. No entanto, ao lidar com expressões booleanas e especialmente tabelas de verdade do portal lógico, não usamos geralmente "ON"(ligado) ou "OFF"(desligado), mas os valores de bit que representam um nível lógico "1"ou um nível lógico "0", respetivamente, podendo traduzir igualmente situações de "Verdadeiro"e "Falso". Estes assuntos são estudados na Lógica Matemática ou Lógica de Boole.

As quatro combinações possíveis de "a" e "b" para um portal lógico de 2 entradas são dadas como:

Combinação de entrada 1. OFF-OFF ou (0, 0)
Combinação de entrada 2. OFF-ON ou (0, 1)
Combinação de entrada 3. ON-OFF ou (1, 0)
Combinação de entrada 4. ON-ON ou (1, 1)

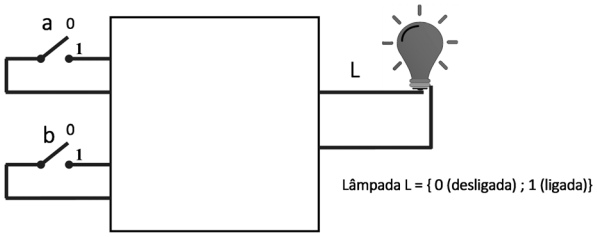


Figura 2.1: Sistema com duas entradas e uma saída.

Para um circuito lógico de 3 entradas, haverá 8 combinações possíveis. Para um número de 4 entradas, haverá 16 combinações, e assim por diante. Em suma, para um circuito lógico com n entradas, haverá 2^n combinações possíveis "ON"e "OFF".

Para o exemplo dado, a lâmpada só liga, se um dos interruptores de entrada estiver fechado, ao que corresponde a tabela lógica de verdade representada na tabela 2.1.

Tabela 2.1: Tabela de verdade do sistema representado na figura 2.1.

Entradas		Saída
a	**b**	**L**
0	0	0
0	1	1
1	0	1
1	1	0

Na álgebra Booleana, existem três operações ou funções básicas: NOT (complemento); OR (OU); AND (E). Qualquer função Booleana, por mais complexa que seja, pode ser representada pela combinação destas três operações básicas.

Em seguida, são apresentadas as tabelas de verdade para uma porta AND de 2 entradas, uma porta OR de 2 entradas e uma porta de entrada única (variável "a") NOT, onde as entradas são representadas pelas variáveis a e b, e a saída pela variável c. Os símbolos "&", "|", e "$\backsim$" são respetivamente os operadores lógicos AND (E), OR (OU), e NOT. Enquanto que a operação negação dá um resultado de valor oposto ao de entrada, a operação AND dará um valor 1, se, e apenas se, as duas entradas forem 1, enquanto que a operação OR dá um valor de saída 1, se apenas uma das entradas for 1.

Operador Negação ($\backsim$)

a	**c=a**
0	1
1	0

Operador "E" (&)

a	**b**	**c=a&b**
0	0	0
0	1	0
1	0	0
1	1	1

Operador "OU" (|)

a	**b**	**c=a|b**
0	0	0
0	1	1
1	0	1
1	1	1

Capítulo 3

Variáveis e Operações

3.1 Variáveis

Na matemática, uma variável é um símbolo ou letra que representa um valor abstrato. São frequentes o uso de letras, tais como "x", "y", "z" ou letras do alfabeto grego. Por vezes, o valor de uma variável é dependente do valor de outras variáveis. Um exemplo dessa dependência pode ser observado através da equação da reta $y = mx + b$, onde x é a variável independente e y a variável dependente, sendo m e b parâmetros constantes. Por exemplo, uma reta definida pelos parâmetros $m = 2$ e $b = 1$, um valor de $x = 1$ corresponde a uma variável y que assume o valor de 3. Já para $x = 2$, o valor de $y = 5$. Para um valor genérico de x, tem-se $y = 2x + 1$.

Em programação, uma variável está associada a um local de armazenamento (identificado por um endereço de memória) e está ligado a um determinado nome simbólico (um identificador), que contém uma quantidade conhecida ou desconhecida de informação designada como um valor. O nome da variável é a forma comum de referenciar o valor armazenado, além de se referir à própria variável, isto dependendo do contexto. Essa separação do nome e do conteúdo permite que seja usado independentemente da informação que ele representa. O identificador no código-fonte do computador pode ser vinculado a um valor durante o tempo de execução, e o valor da variável pode, portanto, mudar durante o curso da execução do programa. Porém, na maior parte das linguagens de programação, o tipo de informação armazenada por cada variável deve ser definida antes do seu uso. Carece, ainda, nessa fase de ser especificado o tipo de variável, podendo esta ser, entre outras, de "caractere", um número inteiro com ou sem sinal, número real ou em vírgula flutuante, apontador de memória etc.

diante.

Na maioria das vezes, a indexação em matrizes é feita usando dois índices – um para as linhas e outro para as colunas. Tomando como exemplo a matriz representada na figura 3.2, um elemento da linha i e coluna j pode ser acedido por A(i,j).

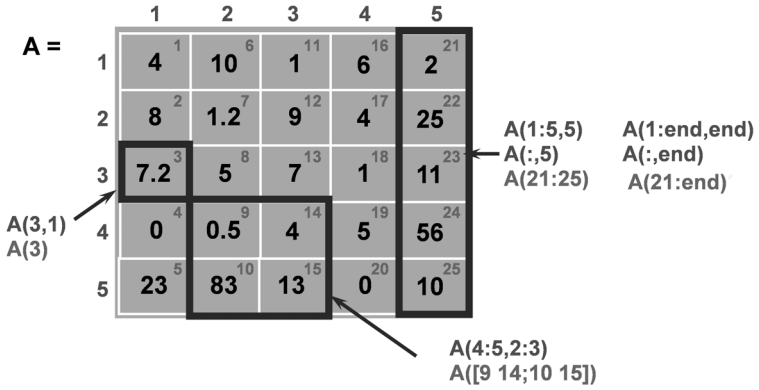


Figura 3.2: Representação da matriz A com dimensão 5x5.

O elemento 7.2 está na linha 3 e sobre a coluna 1. O seu valor pode ser acedido na matriz com A(3,1). O elemento 56 será A(4,5) e assim por diante.

Existe ainda uma outra forma de aceder aos elementos de uma matriz, embora menos usada. As células da matriz são numeradas sequencialmente, percorrendo por ordem crescente de coluna, e dentro desta, do primeiro ao último elemento. No exemplo da figura 3.2, esta numeração está representada no canto superior direito de cada célula com um número, de valores que vão de 1 (1.ª linha e 1.ª coluna) até ao valor de 25 (última linha, última coluna). Assim, o elemento 7.2 pode ser acedido por A(3) (em alternativa a A(3,1)), e o elemento de valor 11 por A(23) (de modo alternativo a A(3,5)).

As submatrizes podem ser copiadas se forem indexadas por vetores de índices. Por exemplo, uma matriz B, constituída pelos elementos da 1.ª e 3.ª linhas e da 2.ª, 4.ª e 5.ª colunas da matriz A:

```
>> B=A([1,3],[2,4,5])
```

```
B =
 10   6   2
  5   1  11
```

Capítulo 4

Representação Gráfica de Dados

O Matlab possui uma diversidade de recursos para exibir graficamente os dados contidos em vetores e matrizes. Existem séries de funções dedicadas à visualização de dados bidimensionais (2D) e tridimensionais (3D), processamento de imagens, animação e gráficos de apresentação. Para ilustrar, e assim compreender algumas das suas capacidades, recorreremos a exemplos demonstrativos.

4.1 Representação Gráfica 2D

A função "plot" cria um gráfico 2D que resulta da ligação de um conjunto de pontos, por meio de segmentos de reta que os une, na ordem em que as suas coordenadas são apresentadas nos vetores coordenadas x e y, respetivamente a abcissa e a ordenada.

Exemplo: Sejam x e y vetores das abcissas e ordenadas, respetivamente, de um conjunto de pontos, com os valores x=[0;2;5;8;10] e y=[-1;0;5;2;4]. O gráfico que une estes pontos é gerado pela execução dos seguintes comandos e está representado na figura 4.1,

```
>> x=[0;2;5;8;10]; y=[-1;0;5;2;4]; plot(x,y);
```

Se existirem mais do que um conjunto de curvas, genericamente n, no mesmo gráfico, estas poderão ser representadas, usando a função plot($x_1, y_1, x_2, y_2, \ldots, x_n, y_n$).

Para desenhar o gráfico de uma função $y = f(x)$, é necessário realizar os seguintes passos:

- Definir a gama de valores, desejavelmente contíguos, para a variável independente x, para os quais queremos desenhar a função.

- Definir a função: $y = f(x)$.

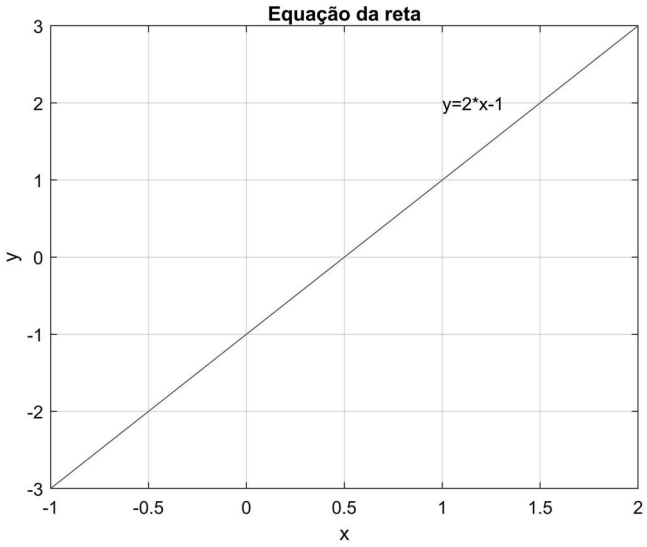


Figura 4.3: Adição de legendas ao gráfico da figura 4.2.

Exemplo: Representar graficamente a função quadrática $y = x^2 - x - 1$, para $-1 \leq x \leq 2$. A função está ilustrada na figura 4.4.

```
figure(2); % Cria uma nova figura número 2.  Se já existir, ativa-a.

% Criar um vetor coluna com 50 elementos de -1 a 2.
x=linspace(-1,2,50)';
% O vetor x é do tipo coluna uma vez ter sido aplicado a função
% transposta ao resultado da função linspace

y=x.^2-x-1;

plot(x,y);
xlabel('x'); % Rótulo no eixo dos xx
ylabel('y'); % Rótulo no eixo dos yy
title('Função quadrática'); % Título do gráfico
grid on; % habilitar a grelha

% Escrever a função na forma texto "y=x.^2-x-1"na posição de
% coordenadas(0,0.5)
text(0,0.5,'y=x.^2-x-1');
```

O mesmo gráfico pode conter várias curvas, conforme está exposto na figura 4.5. Existe a possibilidade de se definirem novos limites para os eixos (e.g. xlim – para o eixo "xx", ylim – para o eixo "yy"

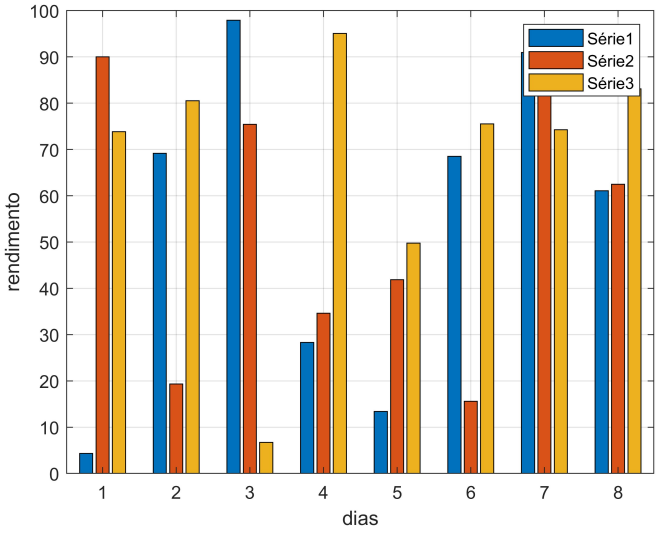


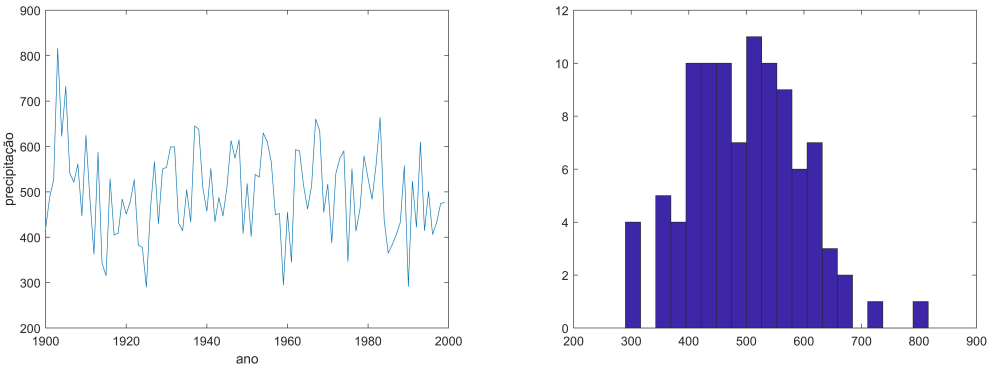
Figura 4.9: Representação de um gráfico de barras criado pela função "bar".

2. Crie um vetor coluna t de dimensão 100, com valores igualmente espaçados entre 0 e 1. Represente graficamente as seguintes funções trigonométricas, sobrepondo-as numa mesma figura com linhas de cor diferentes.
   a) $\sin(2\pi t)$
   (Sugestão: >> t=linspace(0,1,100); y=sin(2*pi*t); plot(t,y); xlabel('Tempo,t'); ylabel('y=sin(2πt)');)
   b) $\cos(2\pi t)$
   c) $\sin(2\pi(t-0.25))$
   d) $0.5\sin(2\pi t)+0.2\cos(2\pi t)$
   e) $sin(\pi t)+0.5\sin(2\pi t)-0.1\sin(3\pi t)+0.2\sin(4\pi t)$

3. Represente graficamente as seguintes funções no espaço $\mathbb{R}^2$, para o intervalo de valores, $x \in [-1,1]$, através da escolha de um número suficiente de pontos.
   a) $f(x)=2x-1$
   (Sugestão: >> x=(-1:1/100:1)'; y=2*x-1; plot(x,y); ylabel('y = 2 × x -1');xlabel('x');)
   b) $f(x)=2x^2-x+1$
   c) $f(x)=e^{-2x^2}$
   d) $f(x)=e^{-x}\cos(10x)$

4. Comece por criar um vetor coluna a com valores de ângulos entre 0 e 2π, espaçados de um valor angular correspondente a $5°$. Em seguida, desenhe graficamente uma circunferência de raio 2 e centro na origem. As equações paramétricas da circunferência são:

$$\begin{cases} x = r\,cos(a) \\ y = r\,sin(a) \end{cases}$$



(a) (b)

Figura 4.10: Representação de 100 pontos aleatórios e histograma.

4.3 Gráficos de Superfície

Os gráficos tridimensionais exprimem basicamente uma relação entre duas variáveis independentes, que designaremos por x e y. Uma outra variável, z, contém os valores dessa relação. A forma normal de exprimir essa relação é através de uma função $z = f(x, y)$. A sua representação gráfica pode ser realizada através de pontos discretos, linhas ou superfícies. Para os dois primeiros tipos pode-se usar a função "plot3".

A função "plot3" estende os gráficos de linhas à terceira dimensão, como mostra o exemplo do desenho da função de equações paramétricas $x = \cos(z) \cdot \cos(50r)$; $y = \cos(z) \cdot \sin(50r)$ para $0 \leq z \leq \frac{\pi}{2}$, representado na figura 4.11. O gráfico foi criado pela execução do seguinte código:

```
>> z=(0:pi/1000:pi/2)';
>> x=cos(z).*cos(50*z);
>> y=cos(z).*sin(50*z);
>> plot3(x,y,z);
>> box on;
>> grid on;
```

Para criar gráficos de superfície a 3D, são necessários os seguintes passos:

1. Criar vetores que estabelecem o domínio da superfície e da sua resolução (se os valores forem iguais, é suficiente definir apenas um deles). Por exemplo, definir o domínio de x, no intervalo [-2,2], e y, no intervalo [-4,4]: x=-2:0.2:2 e y=-4:0.5:4.

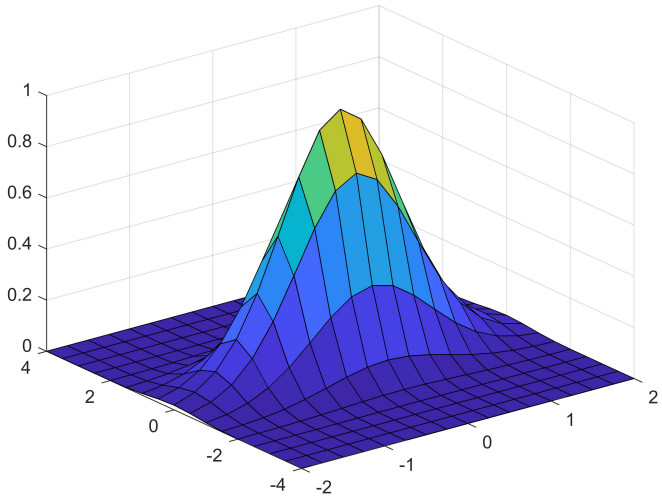


Figura 4.12: Representação de um gráfico em 3D, utilizando a função surf(X, Y, Z).

Exemplo: Desenhar a superfície da função $f(x, y) = (2x - y).e^{-(x^2+y^2)}$ no domínio [-3,3]x[-3,3]. O resultado está representado na figura 4.13, que é o resultado da execução do código seguinte.

```
figure(1);
[x,y] = meshgrid(-3:0.1:3); % Grelha de pontos do domínio
f=(2*x - y).* exp(-x.^2 - y.^2); % Função matemática da superfície
surf(x,y,f); % Desenha superfície
xlabel('x');
ylabel('y');

zlabel('f({x,y})=({2x-y}) \cdot {e^{-({{x^2}+{y^2}})}}');
% As funções "labels"interpretam TEX, tal como sucede neste exemplo.
```

Sugestão: Substitua a função "surf" pela "mesh". Compare as diferenças do resultado.

Uma superfície de uma função de duas variáveis pode ser representada num gráfico 2D por curvas, ao longo das quais a função tem um valor constante, que também podem ser designadas por linhas de nível. Estas criam mapas de contorno, unindo pontos de igual elevação. A função "contour" exibe linhas de nível de uma matriz Z. Essas linhas podem ser rotuladas, usando a função "clabel", que também pode ser obtida pela ativação da propriedade do gráfico 'LevelStep'. O exemplo seguinte cria um gráfico de contornos na função usada no exemplo anterior.

No exemplo que se segue, far-se-á a representação da função f do exemplo anterior, com um número

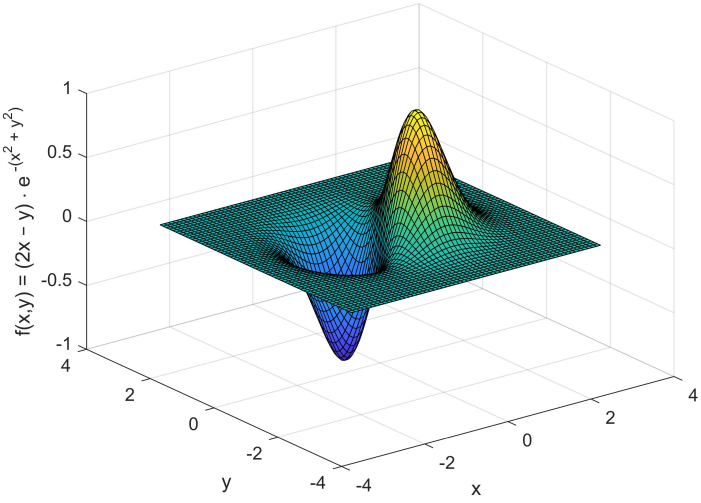


Figura 4.13: Representação da função $f(x,y) = (2x - y).e^{-(x^2+y^2)}$ no domínio [-3,3]x[-3,3].

de 15 linhas. Adicionalmente, a amplitude do nível associado a cada linha é representada. Para esse efeito, na geração do objeto gráfico é guardado, na variável h, o seu "handle", para que mais tarde se possam aceder às suas características. Na sua criação, foram assumidos valores por omissão e que podem ser alterados mais tarde. Por exemplo, o acesso à característica 'LevelStep' é realizado, usando a função "get". Por sua vez, a característica "ShowText" é ativada, usando o comando "set", que resultará na visualização do seu valor sobre cada curva de nível.

Exemplo: Desenhar numa nova figura as linhas de nível da função $f(x,y) = (2x - y).e^{-(x^2+y^2)}$ usada no exemplo anterior. O resultado está patente na figura 4.14.

```
figure(2);
[C,h] = contour(x,y,f,15); % A função "contour"desenha linhas de nível
% (neste exemplo em número de 15)

LS=get(h,'LevelStep');
% Obtém do objeto gráfico de "handle"h o valor da propriedade
% "LevelStep"

set(h,'ShowText','on'); % Faz indicar o valor do nível de cada linha
```

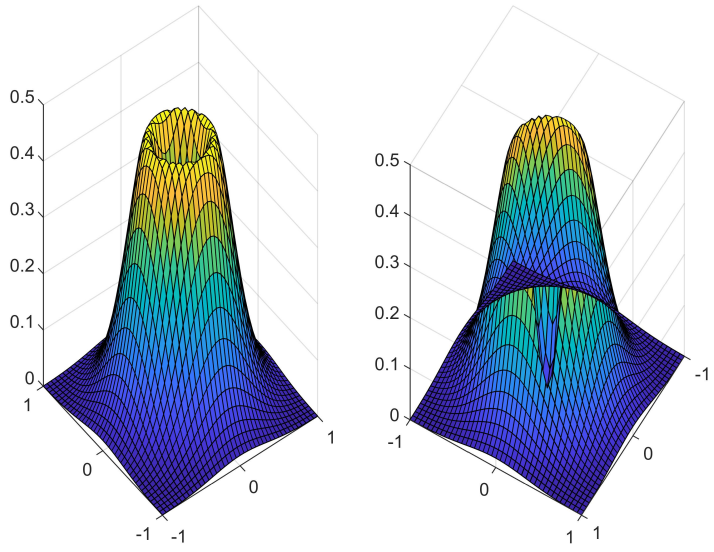


Figura 4.15: Representação gráfica do exercício 5.

4. Repita a questão anterior, usando a função $f(x) = e^{-4x^2}$.

5. Desenhe a superfície da função $f(x,y) = e^{-4\left(x^2+y^2\right)}e^{-16\left(x^2+y^2\right)}$ no domínio $x \in [-1,1]$ e $y \in [-1,1]$. O resultado está representado na figura 4.15.

   Sugestão de resolução para as alíneas c) e d):

```
[X,Y]=meshgrid(-1:0.05:1); % Grelha de pontos no domínio
% Função a desenhar
Z=exp(-4*(X.^2+Y.^2))-exp(-16*(X.^2+Y.^2));
figure(1);
% Vista normal
subplot(1,2,1);
surf(X,Y,Z);
% Vista da superfície numa perspetiva de Azimute=-31 e Elevação=-36
subplot(1,2,2);
surf(X,Y,Z);
view(-31,-36);
```

6. Represente graficamente as seguintes funções em gráficos no espaço $\mathbb{R}^3$, para $x \in [-1,1]$ e $y \in [-1,1]$:

   a) $f(x,y) = 2x + y - 1$

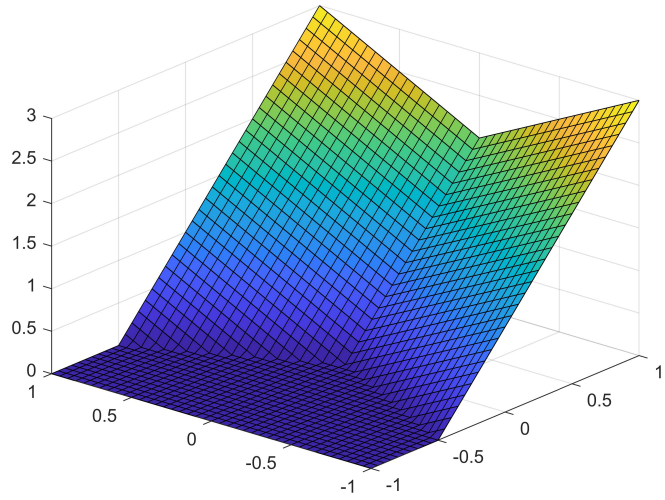


Figura 4.16: Resultado do exercício 6, c).

b) $f(x,y) = \min(0, 2x+y)$

c) $f(x,y) = \min(0, \min(2x-y, 2x+y))$. O resultado está representado na figura 4.16.

d) $f(x,y) = 2x^2 - y + 0.1 \cdot x \cdot y + 1$

e) $f(x,y) = |x-y| + |x+y|$

f) $f(x,y) = e^{-4x^2} \cdot e^{-2y^2}$

g) $f(x,y) = e^{-x}\cos(10y)$

h) $f(x,y) = -10 + \begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} + \begin{bmatrix}x & y\end{bmatrix}\begin{bmatrix}1 & 0.5\\-0.5 & 2\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}$. O resultado pode ser observado na figura 4.18.

i) $f(x,y) = \frac{1}{1+e^{-4(x-y+1)}}$

j) $f(x,y) = \frac{e^{4(x-y+1)} - e^{-4(x-y+1)}}{e^{4(x-y+1)} + e^{-4(x-y+1)}}$

Resolução do exercício 6, e):

```
% Criar uma grelha de pontos para as coordenadas x e y.
[X,Y]=meshgrid(-1:0.1:1); % X e Y são matrizes de dimensão 21x21
Z=abs(X-Y)+abs(X+Y);
surf(X,Y,Z);
xlabel('x');
ylabel('y');
zlabel('z');
```

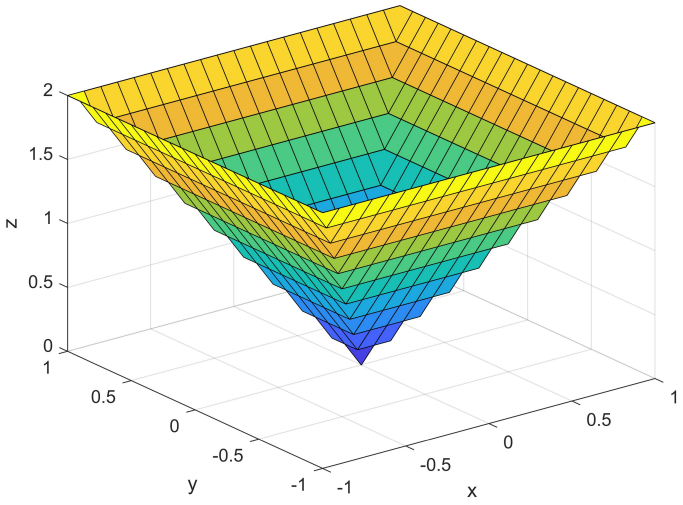


Figura 4.17: Resultado do exercício 6, e).

Resolução do exercício 6, h):

```
% Criar uma grelha de pontos para as coordenadas x e y.
[X,Y]=meshgrid(-1:0.1:1);
% X e Y - matrizes de dimensão 21x21 (nx x ny)
[nx,ny]=size(X);
% Copiar os elementos da Matriz X (21x21) para o vetor x (1 x 441)
x=reshape(X,1,nx*ny);
% Copiar os elementos da Matriz Y (21x21) para o vetor y (1 x 441)
y=reshape(Y,1,nx*ny);
% Criar a matriz xy=[x y]
xy=[x;y];
% Calcular a função para todos os valores (colunas) de xy
b=[1 -1]; A=[1 0.5; -0.5 2];
f=-10+b*xy+sum(xy.*(A*xy));
Z=reshape(f,nx,ny);
surf(X,Y,Z);
xlabel('x'); ylabel('y'); zlabel('z');
```

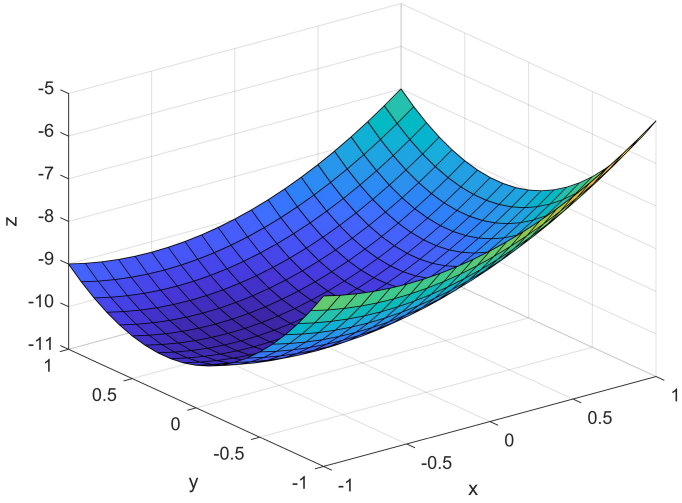


Figura 4.18: Resultado do exercício 6, h).

7. Para um intervalo de valores $x \in [-1, 1]$ e $y \in [-1, 1]$, represente graficamente as seguintes funções:

   a) $f(x, y) = \sin(2\pi x) \cdot \cos(2\pi y)$

   b) $f(x, y) = \sin\left(2\pi\left(x^2 + y^2\right)\right)/\left(2\pi\left(x^2 + y^2\right)\right)$

   c) $f(x, y) = e^{-x^2-y^2}$

   d) $f(x, y) = e^{-x^2-y^2+4xy}$

8. Desenhar um cubo de aresta unitária. O resultado pode ser observado na figura 4.19. (Resolução: coordenadas das arestas: X=[0 1 1 0 0; 0 1 1 0 0]; Y=[0 0 1 1 0; 0 0 1 1 0]; Z=[0 0 0 0 0; 1 1 1 1 1]; surf(X, Y, Z); view(-35,45)).

9. A função surf realiza a junção de segmentos definidos pelos seus pontos extremos. Todos os objetos gráficos seguintes foram obtidos, executando o comando surf(X, Y, Z). Indique para cada caso as matrizes X, Y e Z.

   a) **Solução:** (O resultado está representado na figura 4.20.)

```
% Vértices dos segmentos (das arestas das superfícies planas)
X=[-1 -1 1 1 -1;...
-1 -1 1 1 -1]
Y=[-1 1 1 -1 -1;...
-1 1 1 -1 -1]
Z=[-1 -1 -1 -1 -1;...
1 1 1 1 1]
surf(X,Y,Z);
alpha(.5); % Fator de transparência das superfícies.
```

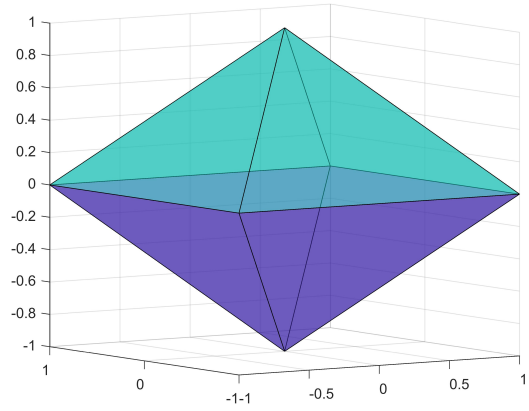


Figura 4.23: Resultado do exercício 9, d).

e) **Complete os valores da Matriz** Z. (Solução: O objeto resultante está representado na figura 4.24.)

```
% Vértices...
X=[0 0 0 0 0;...
-1 -1 1 1 -1;...
-.5 -.5 .5 .5 -.5;...
-.5 -.5 .5 .5 -.5];
Y=[0 0 0 0 0;...
-1 1 1 -1 -1;...
-.5 .5 .5 -.5 -.5;...
-.5 .5 .5 -.5 -.5];
Z=[??  ??  ??  ??  ??;...
??  ??  ??  ??  ??;...
??  ??  ??  ??  ??;...
??  ??  ??  ??  ??];
surf(X,Y,Z);
alpha(.5);
view(-36,16);
```

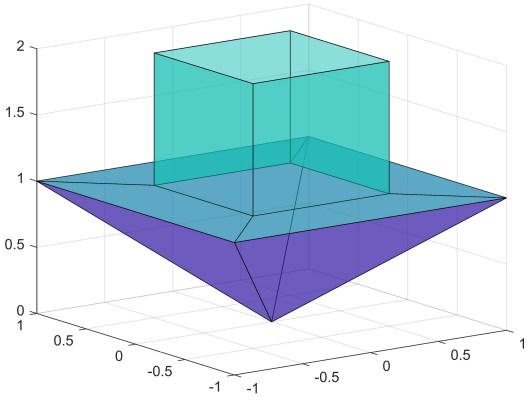
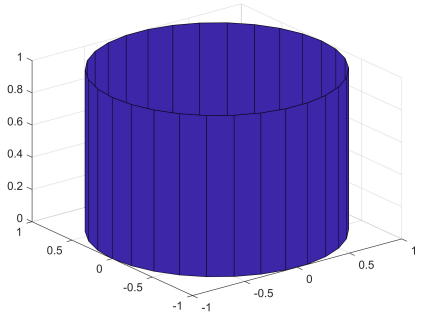
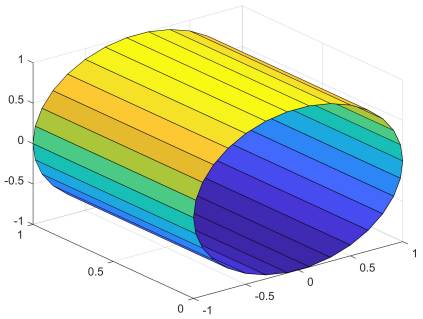


Figura 4.24: Resultado do exercício 9, e).

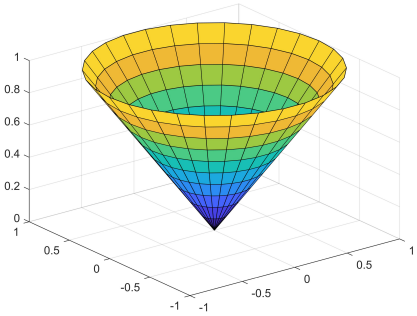
10. Construa, recorrendo às funções do Matlab, os sólidos geométricos presentes nos gráficos da figura 4.25:



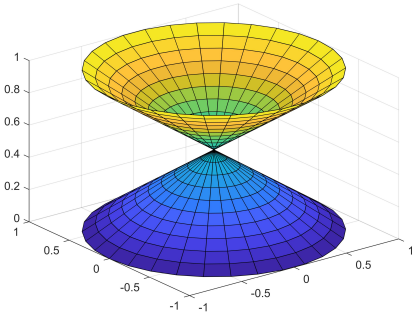
a) Sugestão:>>cylinder(1,30)



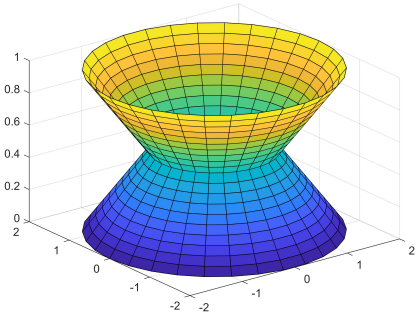
b) Sugestão:>>[X,Y,Z]=cylinder(1,30);surf(X,Z,Y)



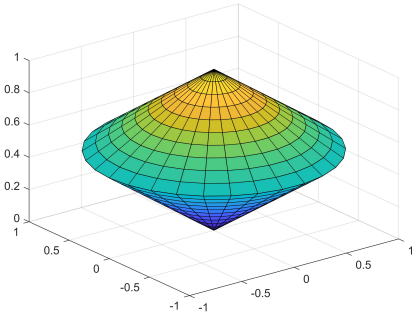
c)



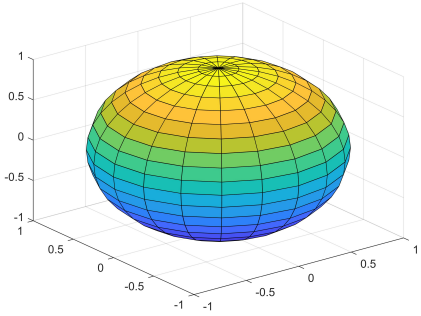
d)



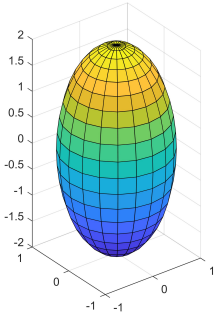
e)



f)

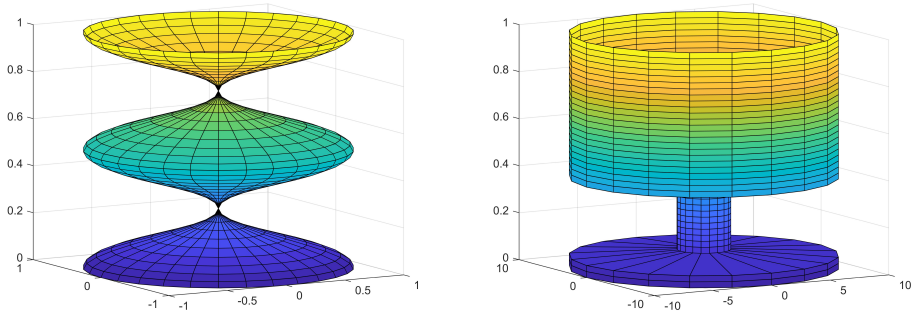


g)



h)

11. Construa um objeto gráfico semelhante ao sólido representado na figura 4.26.



i) j)

Figura 4.25: Representação dos sólidos geométricos relativos ao exercício 10.

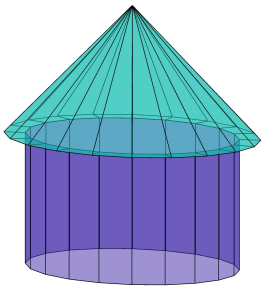


Figura 4.26: Representação da figura geométrica relativa ao exercício 11.

12. O código seguinte produz o sólido geométrico representado na figura 4.27. Que modificações deve fazer no código para obter a figura 4.28?

```
[X,Y,~]=sphere(21);
s=sin(0:2*pi/21:2*pi);
Z=repmat(s',1,22);
surf(X,Z,Y);
alpha(0.5);
xlabel('x');
ylabel('y');
zlabel('z');
```

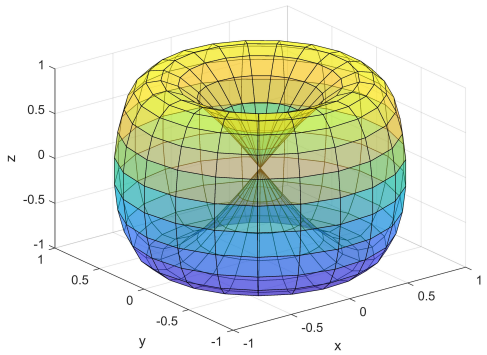


Figura 4.27: Representação do código correspondente ao exercício 12.

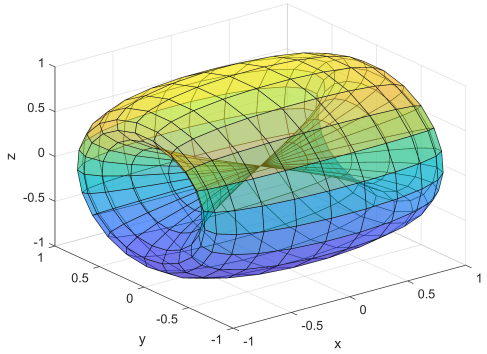


Figura 4.28: Gráfico alusivo ao pedido de alteração do código apresentado no exercício 12.

Capítulo 5

Derivação e Integração Numérica

5.1 Derivação Numérica

A derivada de uma função $y = f(x)$, no ponto do domínio x, é uma função, designada por f', que exprime a taxa de variação da variável de saída y em relação à variável de entrada x. Define-se pela razão da variação ínfima de y com a variação diferencial de x: $f'(x) = \frac{dy}{dx}$, isto é:

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Na salvaguarda da função derivada $f'(x)$ existir na vizinhança de x, para uma quantidade de h extremamente pequena, a razão:

$$\frac{\Delta y}{\Delta x} = \frac{f(x+h) - f(x)}{h}$$

é um valor aproximado da função derivada em x. Esse valor só será igual ao do valor da derivada no limite de h ser zero. Apesar desse facto, esta última expressão é usada para determinar numericamente a derivada de uma função, se o erro cometido pela referida aproximação for confortavelmente pequeno para a exatidão exigida.

O valor de aproximação da derivada também poder ser obtido para o valor médio de um intervalo, tal como ilustrado na figura 5.1, recorrendo à seguinte expressão:

$$\frac{\Delta y}{\Delta x} = \frac{f(x+h/2\,) - f(x-h/2\,)}{h}$$

Seja a função trigonométrica $f(x) = \sin(x)$, com função derivada $f'(x) = \cos(x)$ que será usada para confrontar os valores obtidos pelo método de derivação numérico apresentado. Tomemos h =0.01

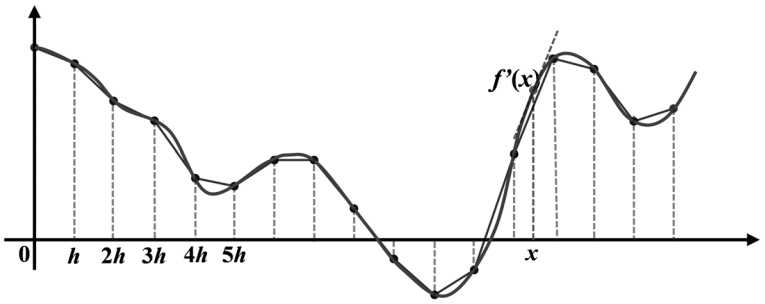


Figura 5.1: O valor da derivada pode ser aproximado ao valor médio num intervalo.

e 200 valores distintos de x entre 0 e 2π. Analise o seguinte código Matlab e, em seguida, execute-o na janela de comandos.

```
% Vetor com 200 ângulos, igualmente espaçados entre 0 e 2*PI
x=linspace(0,2*pi,200);

% Incremento
h=0.1;

% Vetor derivada da função "sin", para os valores do vetor x
dy_dx=cos(x);

% Vetor da aproximação numérica da derivada da função "sin",
% para os valores do vetor x
dyNdx=(sin(x+h)-sin(x))/h;

% Apresentar na figura a função derivada e a sua aproximação
figure(1);
plot(x,dy_dx,'b',x,dyNdx,'--m');
grid on;
xlabel('x');
ylabel('df/dx');
title('sin''(x)=cos(x)');
legend('Derivada','Aprox.  Num.');
```

A figura 5.2 mostra o resultado da aplicação do método numérico (linha a tracejado), confrontado com o resultado teórico (linha contínua).

O erro da aproximação numérica da derivada é calculado pela diferença entre o valor da função derivada e o valor dado pelo método numérico. O código seguinte realiza duas experiências para dois valores de h, 0.1 e 0.01. Os resultados estão representados na figura 5.3.

Capítulo 6

Decisão Condicional

Em geral, num programa, as instruções são executadas sequencialmente. A primeira instrução é executada em primeiro lugar, seguida da segunda e assim por diante. Porém, as linguagens de programação fornecem várias alternativas para controlar a execução das instruções. Entre estas, as estruturas de controle permitem definir caminhos de execução mais elaborados, condicionando a execução de determinados blocos de código à satisfação de critérios ou condições lógicas. O mais conhecido, e certamente o mais utilizado, é a decisão condicional **if...else...end**.

6.1 Decisão "if ... end"

A decisão condicional if ... end obedece à seguinte estrutura:

if *expressão*

% corpo do **if...end**

...
end

A instrução **if** avalia a expressão de teste que lhe segue. A avaliação é realizada de acordo com os operadores relacionais e lógicos anteriormente estudados. Se a expressão de teste é avaliada como verdadeira (diferente de zero), as instruções dentro do corpo de **if...end** são executadas. Se a expressão de teste é avaliada como falsa (igual a zero), as instruções dentro do corpo de **if...end** são ignoradas. Este modo de operação comporta-se como o definido pelo fluxograma da figura 6.1.

Em termos operacionais, a execução do bloco de código definido entre a declaração **if** e a instrução **end** fica condicionada à veracidade da expressão lógica avaliada. Se a condição não for verdadeira, o programa salta para o comando de execução imediatamente após o **end**.

6.2 Decisão "if...else...end"

Uma declaração condicional pode ser seguida por uma declaração opcional **"else"** (senão) que será executada em alternativa, ou seja, caso a expressão Booleana do **"if"** seja falsa. Assim, se a expressão Booleana for verdadeira, o bloco associado ao **"if"** é executado, caso contrário, será o bloco **"else"** a ser processado. A estrutura da declaração **if...else...end** é por conseguinte:

if *expressão*

% Código dentro do corpo do **if**

...
else

% Código dentro do corpo do **else**

...
end
A estrutura de execução do teste **if...else...end** está ilustrada no fluxograma da figura 6.2.

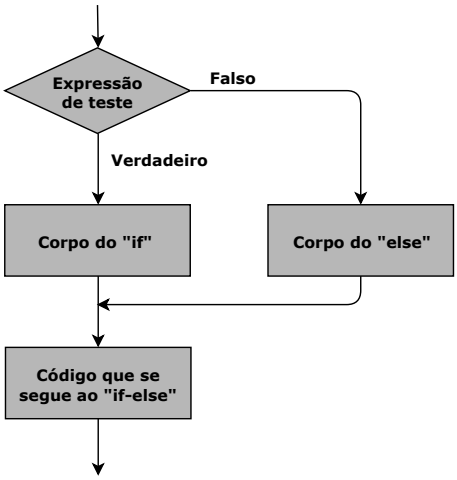


Figura 6.2: Fluxograma da estrutura de decisão "if...else...end"

Exemplo: A função absoluto $f(x) = |x|$:

```
if x>=0
      y=x;
else
      y=-x;
end
```

Capítulo 7

Ciclos

7.1 Introdução

Existem situações frequentes em que é preciso executar um bloco de código várias vezes.

Uma instrução de "loop"ou de ciclo permite executar uma instrução ou grupo de instruções repetidamente, enquanto um determinado conjunto de condições lógicas se mantiverem válidas. Estas são averiguadas em cada ciclo.

7.2 Ciclo "while...end"

Seja a estrutura de código do ciclo while...end:

while *expressão*
    % Código dentro do corpo do while
    ...
end

Se a expressão de teste for verdadeira (diferente de zero), o código dentro do corpo de ciclo **while** é executado. O processo continua até que a expressão de teste seja falsa. Apenas quando a expressão de teste é falsa, é que o ciclo **while** é interrompido, passando a execução para as linhas seguintes ao ciclo, tal como se ilustra no diagrama da figura 7.1.

Acresce ainda que, no Matlab, uma expressão é tida como verdadeira, quando o seu resultado é um valor não-vazio e contenha somente elementos (lógicos ou números reais) diferentes de zero. De outro modo, a expressão é falsa.

i	*j*	*y*=							
1	2		2	5	1	0	4	3	
1	2		5	2*	1	0	4	3	↔
1	3		1	2	5*	0	4	3	↔
1	4		0	2	5	1*	4	3	↔
1	5		0	2	5	1	4*	3	↓
1	6		0	2	5	1	4	3*	↓
2	3		0	2	5*	1	4	3	↓
2	4		0	1	5	2*	4	3	↔
2	5		0	1	5	2	4*	3	↓
2	6		0	1	5	2	4	3*	↓
3	4		0	1	2	5*	4	3	↔
3	5		0	1	2	5	4*	3	↓
3	6		0	1	2	5	4	3*	↓
4	5		0	1	2	4	5*	3	↔
4	6		0	1	2	3	5	4*	↔
5	6		0	1	2	3	4	5*	↔

Figura 7.4: Quadro que mostra a sequência dos valores do vetor y para os ciclos completos de i e j.

$4 > 1$).
Procede-se à troca das suas posições (↔).
j=2: (1 **4** 3 2 5) -> (1 3 **4** 2 5); dado ser $4 > 3$, trocar posições (↔).
j=3: (1 3 **4** 2 5) –> (1 3 2 **4** 5); dado ser $4 > 2$, trocar posições (↔).
j=4: (1 3 2 **4** 5) –> (1 3 2 **4** 5); nesta fase os dois últimos números já estão ordenados ($5 > 4$). O algoritmo não executa a troca de posições (↓). Note-se que, no final desta 1ª passagem por todos os elementos do vetor, os elementos 1 e 5 estão na sua posição final.

2.ª passagem (i=2):

j=1: (1 3 2 4 5) –> (1 3 2 4 5); não há lugar a troca (↓).
j=2: (1 3 2 4 5) –> (1 2 3 4 5); dado 3>2, trocar posições (↔).
j=3: (1 2 3 4 5) –> (1 2 3 4 5); não há lugar a troca (↓).

3.ª passagem (i=3):
j=1: (1 2 3 4 5) –> (1 2 3 4 5); não há lugar a troca (↓).
j=2: (1 2 3 4 5) –> (1 2 3 4 5); não há lugar a troca (↓).

Capítulo 8

Scripts e Funções

8.1 Introdução

O Matlab é um programa interativo para computação numérica e visualização de dados. Este permite que o utilizador digite comandos na consola ou janela de comandos, onde é exibido o "prompt" assinalado pelos símbolos >>. Um interpretador nativo interpreta e executa esses comandos, dando como resposta variáveis que farão parte da memória WorkSpace, onde poderão ser de novo usadas por outros comandos ou funções executadas posteriormente. Pode ainda resultar dessa execução a criação de objetos gráficos, ficheiros de dados ou mensagens diretamente escritas na janela de comandos. Foi deste modo que, nos capítulos anteriores, foram apresentados exemplos compostos por comandos ou pelas suas combinações mais ou menos complexas, na forma de uma sequência de linhas de comandos. O resultado de todos esses exemplos pode ser obtido pela digitação dessa(s) linha(s) de comando(s), na ordem apresentada.

Em alternativa, estas linhas, na sequência pretendida, podem fazer parte de um ficheiro de texto designado por "script", que pode ser evocado posteriormente e executado. Tal garante um grau de flexibilidade e extensibilidade de programação, que ainda pode ser construída na forma de funções que aceitem entrada e saídas de valores. Ambos os ficheiros do tipo "script" ou de funções possuem a extensão ".m", e, como ficheiro de texto, podem ser abertos e editados por qualquer programa de edição de texto.

No primeiro caso, designam-se por "m-files script" e, no segundo, de "m-files function". Um outro género de ficheiro é usado para guardar dados numa estrutura própria, nomeadamente os arquivos com extensão ".mat"para binários ou, para forma de texto, com extensão ".txt". São manipulados pelos comandos "save" e "load".

Do ponto de vista descrito, o Matlab incorpora e interpreta uma linguagem de programação imperativa. Foi assim, desde as primeiras versões, passando a incluir nas versões mais recentes a programação orientada a objetos, cujo estudo não fará parte deste livro. O paradigma de programação imperativa,

25. Seja a função $d = f(x)$. Implemente uma rotina que dê o integral da função no intervalo $[a, b]$ pela aproximação da soma de n ...
    a) ... Retângulos;
    b) ... Trapézios.

26. Implemente a resolução do sistema de equações $Ax = b$ pelo método de Eliminação de Gauss.

27. Da definição da função de exponenciação, tem-se:

$$x^n = \begin{cases} 1 & , se\ n = 0 \\ x \times x^{n-1} & , outros\ casos \end{cases}$$

    Implemente uma função recursiva para o cálculo da exponenciação de um número inteiro.

28. O algoritmo de bissecção permite encontrar as raízes de uma função contínua $y = f(x)$ num intervalo $[a, b]$, desde que os sinais de $f(a)$ e $f(b)$ sejam opostos (isto é, $f(a) \times f(b) < 0$). Deste modo, haverá pelo menos um zero no intervalo considerado, isto é, $\exists x \in [a, b] : f(x) = 0$, como se exemplifica na figura 8.1.

    O método divide o intervalo no seu ponto médio $c = (a + b)/2$. Este valor irá substituir um dos extremos do intervalo anterior, garantindo que a função $f(x)$ continue a ter sinal contrário nos extremos deste novo intervalo, e que aí continue a existir um zero da função ($f(x) = 0$), à medida que vai estreitando a largura do intervalo. Assim, se $f(a) f(c) < 0$, o novo intervalo será $[a, c]$, caso contrário, será $[c, b]$. O processo repete-se até que c aproxime a raiz com a precisão desejada e a largura do intervalo seja inferior a uma quantidade suficientemente pequena, $|b - a| < \varepsilon$.

    a) Implemente o algoritmo, recorrendo a um processo cíclico;
    b) Recorra a um método recursivo para implementar o algoritmo numa função.

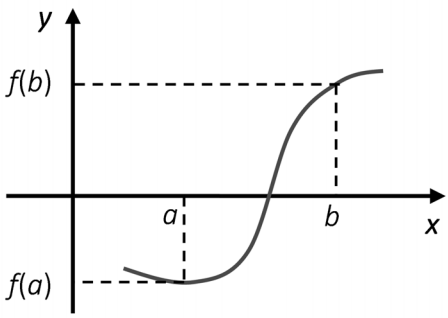


Figura 8.1: Representação do algoritmo da bissecção relativo ao exercício n.º 28.

Capítulo 9

Input / Output (I/O)

9.1 Importar e Exportar Dados

Importar e exportar dados, da e para a WorkSpace do Matlab, melhora significativamente a aplicabilidade e potencia as suas capacidades de utilização. O Matlab dispõe de funções personalizadas para importar e exportar arquivos específicos, nomeadamente arquivos MAT, dados de texto (txt), folhas de cálculo (por ex. EXCEL), dados científicos, imagens, áudio e vídeo, dados XML (eXtensible Markup Language), entre outros, acrescentando-se ainda a capacidade de recolha de dados em páginas WEB. Far-se-á, de seguida, a apresentação de alguns exemplos usados neste domínio.

Uma forma simples de exportar os dados de uma variável e dar a conhecer o seu valor, consiste em evocá-la na linha de comandos, no script ou função, sem que a linha termine com um ponto-e-vírgula. Por exemplo, se na WorkSpace existir a variável x (com o valor escalar 1) e o vetor linha v (com elementos inteiros de 1 a 5), a sequência seguinte de execução revela os seus valores na "janela de comandos".

```
>> x
x =
       1
```

Outro modo de mostrar o valor de uma variável X (sem apresentar o nome da variável) é com a função disp(X) (display).

```
>> x=1:5;
>>disp(x)
       1  2  3  4  5
```

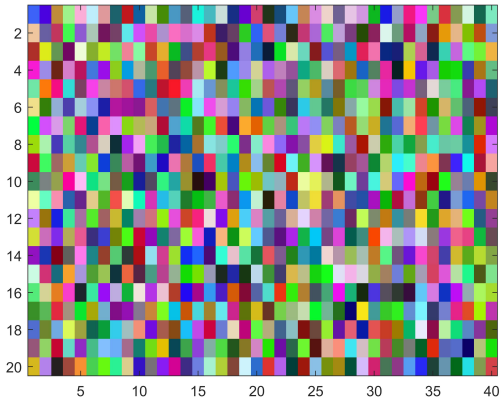


Figura 9.1: Representação da imagem gerada pela função A=uint8(rand(20,40,3).*255);.

ficheiro 'noiseimg.jpg'. O resultado pode ser consultado graficamente na figura 9.1.

```
>> A=uint8(rand(20,40,3).*255);
>> image(A);
>> imwrite(A,'noiseimg','jpg');
```

A tabela 9.3 resume as principais funções de leitura/escrita de imagens.

Tabela 9.3: Principais funções de leitura/escrita de imagens

Funções	Tarefa
imfinfo	Informação acerca do ficheiro de imagem.
imread	Lê a imagem contida num ficheiro.
imwrite	Grava uma imagem num ficheiro de imagem.
im2java	Converte uma imagem numa imagem Java.

De modo equivalente, existem funções para ler e escrever dados áudio em ficheiros, nomeadamente as funções **"audioinfo"**, **"audioread"** e **"audiowrite"**. Os sinais de áudio mono ou stereo, os dois mais comuns, têm respetivamente dimensão vetorial ou matricial. Os dados do som stereo estão em 2 colunas, uma por canal, de valores normalizados no intervalo [-1,1]. Um outro parâmetro importante é a frequência de amostragem dos sinais de áudio, "Fs", que indica a taxa de reprodução ou leitura temporal das amostras.

Introdução à Programação em MATLAB

PAULO AFONSO
PAULO SALGADO

MATLAB® examples

Sobre a obra

O MATLAB™ é uma poderosa ferramenta de cálculo informática que é utilizada em diversos domínios do conhecimento. Este livro pretende simplificar e dinamizar a utilização deste programa em duas vertentes: a primeira como ferramenta de cálculo e a segunda como elemento de introdução à programação.

A aprendizagem é feita através de uma abordagem simples às funcionalidades e regras do MATLAB™, recorrendo a inúmeros exemplos demonstrativos, e identificando os pontos chave que permitem dotar os leitores de um conjunto de regras que permitem utilizar esta plataforma nas tarefas mais comuns.

O livro foi pensado para ser usado em unidades letivas do ensino superior e universitário, em especial nos estudantes dos primeiros anos e profissionais com ou sem formação superior, em áreas tão dispersas como a economia, gestão, biologia, matemática e engenharia. A quantidade de exercícios resolvidos e propostos diferencia esta obra entre as que exploram esta temática.

Sobre os autores

Paulo Afonso é licenciado em Engenharia Eletrotécnica e de Computadores pela Faculdade de Engenharia da Universidade do Porto (FEUP), e doutorado em Engenharia Química pela Faculdade de Ciências e Tecnologia da Universidade de Coimbra (FCTUC). É professor Coordenador na Escola Superior de Tecnologia e Gestão de Águeda (ESTGA) pertencente à Universidade de Aveiro, sendo regente das unidades curriculares de Instrumentação Industrial, Eletrónica Industrial, e Microcontroladores e Microprocessadores. É Investigador no Instituto de Telecomunicações.

Paulo Salgado é licenciado e mestre em Engenharia Eletrotécnica e de Computadores pela Faculdade de Engenharia da Universidade do Porto (FEUP), e doutorado em Engenharia Eletrotécnica e de Computadores pela Universidade de Trás-os-Montes e Alto Douro (UTAD). É Professor Associado na Escola de Ciências e Tecnologia do Departamento de Engenharias da UTAD, sendo regente das unidades curriculares de Robótica, Sistemas Inteligentes e Programação. É Investigador nas áreas da Inteligência Artificial, Robótica e Instrumentação Eletrónica.

Também disponível em formato *e-book*

ISBN: 978-989-892-744-6

www.**engebook**.pt

engebook