

第 1 章

MATLAB 基础知识

本章主要介绍 MATLAB 软件平台的基础知识。从一个有趣的魔方矩阵例子开始，介绍矩阵的基本操作及 MATLAB 的环境。对于 MATLAB 图形的绘制，重点描述二维绘图，对三维绘图和图像操作做了简单介绍。本章内容为后期进一步学习打下基础。

1.1 MATLAB 入门

MATLAB 入门的最好方法就是学会如何处理矩阵。在 MATLAB 中，矩阵是一个矩形的数字阵列。 1×1 矩阵的特别意义为标量，仅有一列或一行的矩阵则为向量。通常，最好将一切都看成矩阵。不同于其他编程语言，一次处理一个数据，MATLAB 能够迅速而方便地处理整个矩阵。

在本课程中，使用一个很好的示例矩阵，它出现在文艺复兴时期一位德国艺术家兼业余数学爱好者阿布列西特·杜勒的雕刻作品《忧郁者 I》中。这幅图布满了数学符号，如果仔细观察，你会发现右上角有一个矩阵（如图 1.1.1 所示）。这个矩阵就是众所周知的魔方阵，早在杜勒时代，人们就相信它具有魔力，而它的确证明具有值得研究的神奇特性。

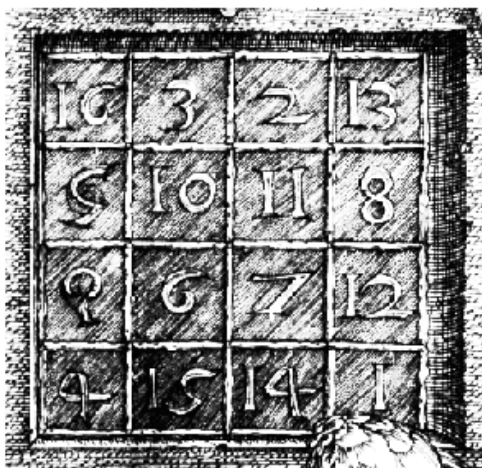


图 1.1.1 魔方阵

1.1.1 输入矩阵

可以用几种不同的方法将矩阵输入到 MATLAB 中：

1. 直接输入矩阵的元素列表。
2. 从外部数据文件加载矩阵。
3. 利用内置函数生成矩阵。
4. 在 M 文件中自定义函数创建矩阵。

需遵循的几个基本规则是：

1. 同一行内的元素用空格或逗号隔开。
2. 行与行之间，用分号“;”表示每行的结束。
3. 矩阵的全体元素使用方括号“[]”括住。

依据上述基本规则，要输入杜勒的矩阵，只需输入：

```
A=[16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

按回车键，MATLAB 显示：

$$A = \begin{bmatrix} 16 & 3 & 2 & 13 \\ 5 & 10 & 11 & 8 \\ 9 & 6 & 7 & 12 \\ 4 & 15 & 14 & 1 \end{bmatrix}$$

它有何神奇之处呢？

1.1.2 求和、转置和对角线

你可能已经发现，魔方矩阵的特性与不同的求和方式有关。计算任一行或任一列，或两主对角线上元素的和，总会得到相同的结果。下面使用 MATLAB 来验证这一结论。

尝试的第一条语句

```
sum(A)
```

MATLAB 显示

```
ans =  
    34    34    34    34
```

不指定输出变量时，MATLAB 默认用变量 `ans` (`answer` 的缩写) 来储存最近的计算结果。已经计算出的结果是一个行向量，它包含了矩阵 **A** 所有列的所有元素之和。很明显，每一列的和不可思议地相同，都是 34。

那么，每一行的和会怎样呢？

先将矩阵转置，计算转置矩阵每一列之和，然后将结果再转置。转置操作符用一撇或单引号 ‘`’` 表示。例如，输入

```
sum(A')'
```

```
ans =  
    34  
    34  
    34  
    34
```

此时得到的列向量为对矩阵 **A** 的每一行求和的结果。显然，各行的和仍为 34。

使用 `diag` 函数，很容易获得主对角线上所有元素之和。`diag` 函数的功能是提取矩阵对角线上的元素。

```
sum(diag(A))
```

```
ans =  
    34
```

另外一个对角线，即所谓的反对角线，`fliplr` 函数的功能是将矩阵左右翻转，将反对角线转换成主对角线。

```
sum(diag(fliplr(A)))
```

```
ans =  
    34
```

上述结果证明, 矩阵 **A** 确实是一个魔方矩阵。在此过程中, 列举了几个 MATLAB 矩阵运算。下面继续用此矩阵介绍 MATLAB 的其他功能。

1.1.3 下标

矩阵 **A** 的第 i 行、第 j 列的元素用 $\mathbf{A}(i, j)$ 表示。

那么, $\mathbf{A}(1,4)+\mathbf{A}(2,4)+\mathbf{A}(3,4)+\mathbf{A}(4,4)$ 的计算语句及其结果为

```
A(1,4)+A(2,4)+A(3,4)+A(4,4)
```

```
ans =  
34
```

除了上述双下标的元素表示方法, 矩阵索引也可以用一个单下标 $\mathbf{A}(k)$ 表示矩阵元素。这种方法常用来表示行向量和列向量的元素。但它也可以用于二维矩阵, 在这种情况下, 二维矩阵被视为由其原矩阵各列依次构成的一个长的列向量。因此, 对于魔方矩阵 **A**, $\mathbf{A}(8)$ 与 $\mathbf{A}(4,2)$ 是对矩阵 **A** 中 15 这一元素的两种表示方法。

1.1.4 冒号运算符

冒号 “:” 是 MATLAB 中最重要的运算符之一, 它有几种不同的表达形式。表达式 1:10 表示包含从 1 到 10 的整数行向量。

```
1 2 3 4 5 6 7 8 9 10
```

为了得到非单位的间距, 可以指定一个增量, 例如, 100:-7:50 是

```
100 93 86 79 72 65 58 51
```

0:pi/4:pi 是

```
0 0.7854 1.5708 2.3562 3.1416
```

下标表达式中使用冒号可以表示矩阵的一部分。 $\mathbf{A}(1:k,j)$ 指矩阵 **A** 的第 j 列前 k 个元素。 $\text{sum}(\mathbf{A}(1:4,4))$ 为计算 **A** 的第 4 列的和。但有一个更好的方式, 用冒号表示矩阵中某一行或列的所有元素, 用关键词 **end** 表示最后一行或列的所有元素。

因此, 用命令

```
sum(A(:,end))
```

```
ans =  
34
```

来计算 **A** 的最后一列元素之和。

为什么 4×4 魔方矩阵的和等于 34? 如果将 1 到 16 的整数之和等分为 4 份, 那么每一份的和为 $\text{sum}(1:16)/4$, 可得 34。

```
sum(1:16)/4
```

```
ans =  
34
```

1.1.5 魔方阵函数

实际上, MATLAB 提供了一个内嵌函数, 它可以创建任意大小的魔方矩阵。毫不奇怪, 这个函数就是 `magic`。

```
B = magic(4)
```

$$B = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

这个矩阵与杜勒雕刻版上的矩阵几乎一样, 具有所有相同的神奇性质。唯一的区别在于中间的两列调换了位置。为了将这个矩阵 **B** 转变成杜勒的矩阵 **A**, 调换 **B** 中间的两列

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

上式表示, 对于矩阵 **B** 的每一行, 将其列的顺序按照 “1, 3, 2, 4” 重新编排。

1.2 表达式

像大多数其他编程语言一样, MATLAB 提供了数学表达式。但与大多数编程语言不同, MATLAB 的表达式针对整个矩阵, 构成如下:

- (1) 变量
- (2) 数字
- (3) 运算符
- (4) 函数

1.2.1 变量

变量名由字母构成, 后面跟字母、数字或下画线。MATLAB 具有区分大小写的特性, 所以, “A” 和 “a” 是两个不同的变量。

1.2.2 数字

MATLAB 使用传统的十进制记数法, 带有一个可选的小数点和正负符号。科学记数法使用字母 `e` 表示以 10 为底的指数。虚数使用 `i` 或 `j` 作为后缀。

下面是一些合法数字的例子。

3	-99	0.000
9.6397238	1.60210e-20	6.02252e23
1i	-3.14159j	3e5i

1.2.3 运算符

使用常见算术符号的表达式和优先规则如表 1.1 所示。

表 1.1 使用常见算术符号的表达式

常用算术符号	含 义
+	加
-	减
*	乘
/	除
\	左除
^	幂
,	复共轭转置
()	指定计算顺序

1.2.4 函数

MATLAB 提供丰富的标准初等数学函数，包括 abs（绝对值）、sqrt（平方根）和 exp（指数函数）等。MATLAB 也提供很多高等数学函数，如 gamma（伽马）函数，这些函数中，大多数都接受复杂的参数。一些函数是 MATLAB 内置的，如 exp 和 cos。它们是 MATLAB 的部分核心构成，因此用起来比较高效，但内部的计算细节不能查询。

表 1.2 为一些特殊的函数提供了有用的常量值。

表 1.2 特殊函数的常量值

常 量	含 义
pi	3.14159265
i	虚数单位，-1
j	同 i
eps	浮点相对精度，2.2204e-16
real min	最小浮点数，2.2251e-308
real max	最大浮点数，1.7977e+308
Inf	无穷大
nan	不定数

我们已看到一些 MATLAB 表达式的例子，下面再举一些例子和运算结果。

```
rho = (1+sqrt(5))/2
rho = 1. 6180
a = abs(3+4i)
a=5
huge = exp(log( realmax) )
huge =1. 7977e+308
toobig = pi *huge
toobig = Inf
```

1.3 矩阵操作

下面介绍生成矩阵的其他方法。

1.3.1 生成矩阵

MATLAB 提供 4 个生成基本矩阵的函数（见表 1.3）。

表 1.3 4 个基本函数

函 数	说 明
zeros	全 0 矩阵
ones	全 1 矩阵
rang	均匀分布的随机矩阵
rangn	正态分布的随机矩阵

1.3.2 M 文件

M 文件是包含 MATLAB 代码的文本文件，可以用 M 文件创建自己的矩阵。在 MATLAB 命令行键入一些命令，即可创建包含该命令的文件，将其文件名保存为.m 格式。

例如，创建一个文件包含

```
A=[16, 2, 3, 13; 5, 11, 10, 8; 9, 7, 6, 12; 4, 14, 15, 1]
```

将该文件名保存为 magik.m。magik 文件的文本内容，即上文的示例魔方矩阵。

1.3.3 连接

连接是一个将小矩阵合并成大矩阵的过程。事实上，你创建的第一个矩阵，就是将它的单个元素连接在一起获得的。方括号[]是连接运算符。例如，以 4×4 魔方矩阵 **A** 为基础，构造 **B**

```
B=[A A+32; A+48 A+16]
```

连接 4 个子矩阵，得到 8×8 矩阵 **B**。

```
B =  
    16     3     2    13    48    35    34    45  
     5    10    11     8    37    42    43    40  
     9     6     7    12    41    38    39    44  
     4    15    14     1    36    47    46    33  
    64    51    50    61    32    19    18    29  
    53    58    59    56    21    26    27    24  
    57    54    55    60    25    22    23    28  
    52    63    62    49    20    31    30    17
```

这个矩阵仅是成为另一个魔方矩阵的过渡。它的元素是从 1~64 整数的重新排列。它的

列向量之和正是一个 8×8 魔方矩阵的值。

```
sum(B)
```

```
ans = 260 260 260 260 260 260 260 260
```

但是，它的行向量之和 $\text{sum}(B')$ ，并不总是相同的。为了使之成为真正 8×8 的魔方矩阵，还需要进一步的处理。

1.3.4 删除行和列

使用一对方括号，可删除一个矩阵的行和列。例如， $X=A$ ，然后，删除 X 的第二列，用 $X(:,2)=[]$ 表示。

```
X(:,2)=[]
```

结果是

```
X =  
    16     2    13  
     5    11     8  
     9     7    12  
     4    14     1
```

如果删除矩阵的单个元素，其结果不再是矩阵。所以，像这样的表达式 $X(1,2)=[]$ ，结果会报错。因此，用一个单下标删除一个元素或一系列元素，并把剩下的元素再形成一个行向量。

```
X(2:2:10)=[]
```

结果是

```
X =  
    16     9     2     7    13    12     1
```

1.4 命令窗口与常用函数

目前为止，已经使用了 MATLAB 命令行、键入命令和表达式，并且看到了命令窗口显示出的运行结果。

本节描述改变命令窗口外观的方法，并介绍几个常用函数。如果系统允许选择命令窗口的字体和字形，为了保证合适的间隔，建议采用固定宽度的字体，如 Fixedsys 或 Courier。

1.4.1 格式命令与长命令行

`format` 命令用于控制 MATLAB 显示的数据值格式。该命令只影响数据显示的方式，而不影响 MATLAB 对数据的计算和储存。

下面是向量 x 在不同格式和量级下的相应结果显示。

```

x=[4/3  1.2345e-6]
format short
1.3333  0.0000
format short e
1.3333e+000  1.2345e-006
format short g
1.3333  1.2345e-006
format long
1.333333333333333  0.000001234500000
format long e
1.333333333333333e+000 1.234500000000000e-006
format long g
1.333333333333333  1.2345e-006
format rat
4/3  1/810045
format hex
3ff5555555555555  3eb4b6231abfd271

```

如果矩阵中最大的元素比 10^3 大，或者最小的元素比 10^{-3} 小，那么 MATLAB 会为 short 型和 long 型数据采用一个常用的比例因子。

除了上述命令格式，利用 **format compact** 命令能够消除输出中出现的空白行，让你能在屏幕或窗口查看更多信息。

如果一条语句无法在一行输入完，则键入三个点“...”，然后按回车键，表示下一行继续该指令的输入。例如：

```
i=1+2+3+4+5+6+7+8+...+10+11;
```

1.4.2 隐藏输出

如果你简单输入一个语句，按回车键，MATLAB 会自动显示其运行结果。然而，如果该行命令是以“;”结尾的，MATLAB 仍会执行运算，但不显示任何输出。这一点对于生成大型矩阵时特别有用。

例如，`A=magic(100);`

（`magic` 函数是生成魔方矩阵的函数命令，后面会讲到。）

1.4.3 命令行编辑

键盘上的箭头键和控制键允许撤销、编辑或重新执行已输入的命令。

例如，假设错误地输入 `rho = (1 + sqrt(5)/2)`，即错误拼写了 `sqrt`，MATLAB 会显示 `Undefined function or variable 'sqrt'`（未定义的函数或变量 'sqrt'）。

这时，不需要重新输入整行命令，只需按 $\uparrow$ 键，拼写错误的命令会重新显示。用 $\leftarrow$ 键，移动光标并添加遗漏的 `r`。也可重复使用 $\uparrow$ 键召回之前的命令行，键入几个字符，然后用 $\uparrow$ 键，找到前面以这些字符开头的命令行。

表 1.4 是常用的命令行编辑指令。

表 1.4 命令行编辑指令

↑	Ctrl+p	召回上一行
↓	Ctrl+n	召回下一行
←	Ctrl+b	后退一个字符
→	Ctrl+f	前进一个字符
Ctrl+→	Ctrl+r	右移一个单词
Ctrl+←	Ctrl+l	左移一个单词
Home	Ctrl+a	移至行始
End	Ctrl+e	移至行末
Esc	Ctrl+u	清除该行
Del	Ctrl+d	删除光标所在字符
Backspace	Ctrl+h	删除光标之前字符
	Ctrl+k	删除至行末

1.4.4 常用基本函数

1. sum 函数

求和函数，表示数组元素的和。

例如，一维数组 $A=[1 \ 2 \ 3 \ 4 \ 5]$ ，对 A 求和，则 $S=\text{sum}(A)$ ，计算结果为 $S=15$ 。

又如，二维数组 $B=\begin{bmatrix} 1 & 4 \\ 6 & 11 \end{bmatrix}$ ，对 B 求和就是对 B 的列元素求和，则 $S=\text{sum}(A)$ ，计算

结果为 $S=[7 \ 15]$ 。

2. transpose 函数

转置函数，对数组进行转置。假定矩阵 $B=[1 \ 2 \ 3]$ ，对矩阵 B 使用转置函数，则

$C=\text{transpose}(B)$ ，可以得到 B 的转置矩阵 $C=\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ 。

3. diag 函数

该函数创建对角矩阵或得到矩阵的对角元素。假定矩阵 $E=[1 \ 2 \ 3]$ ，用 $D=\text{diag}(E)$ ，对 E 使用 diag 函数，生成矩阵 D 的主对角线元素为 E 矩阵的元素，其余元素为 0，得到

$D=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$ 。

又如，矩阵 $F=\begin{bmatrix} 2 & 3 & 3 \\ 3 & 2 & 3 \\ 3 & 1 & 3 \end{bmatrix}$ ，对 F 矩阵使用 diag 函数。输入 $G=\text{diag}(F)$ ，提取 F 矩阵

中的主对角线元素，从而构成了 G 矩阵的元素，即 $G=[2 \ 2 \ 3]$ 。

4. fliplr 函数

翻转函数，使数组的列从左到右翻转。假定矩阵 $A=[1 \ 2 \ 3 \ 4 \ 5]$ ，则使用

$H = \text{fliplr}(A)$ ，得到矩阵 A 的翻转矩阵 $H = [5 \ 4 \ 3 \ 2 \ 1]$ 。

又如，对矩阵 $F = \begin{bmatrix} 2 & 3 & 3 \\ 4 & 2 & 3 \\ 6 & 1 & 3 \end{bmatrix}$ ，使用 $H = \text{fliplr}(F)$ ，得到 F 的翻转矩阵 H ， $H = \begin{bmatrix} 3 & 3 & 2 \\ 3 & 2 & 4 \\ 3 & 1 & 6 \end{bmatrix}$ 。

5. magic 函数

该函数创建一个魔方矩阵，使其具有相同的行数和列数，并在每行每列及其对角线上的和都相等的矩阵，且矩阵中的每个元素互不相同。例如，使用 $I = \text{magic}(3)$ 这个命令函数可

以得到 $I = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$ 。

1.4.5 常用进阶函数

学习了常用的基本函数之后，再介绍几种常用的进阶函数。

1. conv 函数

卷积和多项式乘法函数，创建包含多项式和向量系数的 u 与 v 。例如， $u = [1 \ 0 \ 1]$ ， $v = [2 \ 7]$ ，使用函数 $w = \text{conv}(u, v)$ ，可以得到卷积后的向量 $w = [2 \ 7 \ 2 \ 7]$ 。

2. find 函数

该函数查找非零元素的索引和值的函数。假定一个矩阵 $X = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 4 \end{bmatrix}$ ，使用函数

$Y = \text{find}(X)$ 在矩阵 X 中找到非零元素为 $Y = [1 \ 5 \ 7 \ 8 \ 9]$ 。

又如，有矩阵 $X = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 7 & 5 \\ 4 & 9 & 2 \end{bmatrix}$ ，使用函数 $Y = \text{find}(X > 5)$ ，在矩阵 X 中找到元素值大于

5 的元素，将其单索引值输出到 Y 中， $Y = [1 \ 5 \ 6 \ 7]$ 。

3. step 函数

该函数用于绘制连续系统的阶跃响应图，生成一个随机的稳定传递函数模型

$G(s) = \frac{\text{num}(s)}{\text{den}(s)}$ ，写成具体 s 函数形式为

$$G(s) = \frac{10s + 25}{0.16s^3 + 1.96s^2 + 10s + 25}$$

阶跃响应图实现的程序如下所示。

```
num=[10 25];
den=[0.16 1.96 10 25];
t=0:0.02:5;
step(num,den,t);
```

绘制出连续系统的阶跃响应如图 1.4.1 所示。

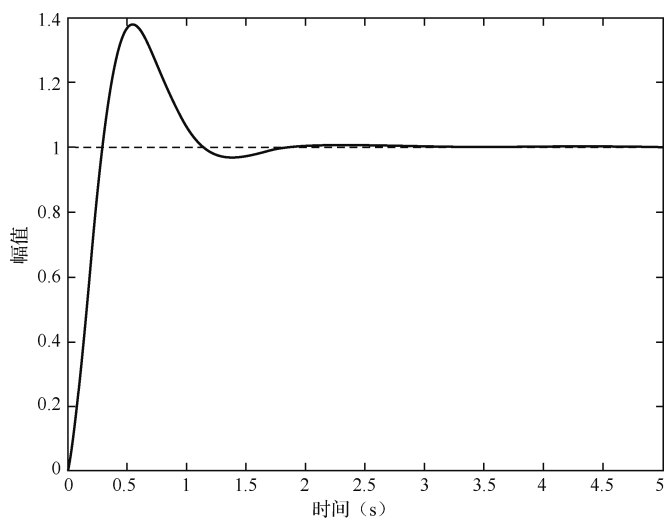


图 1.4.1 连续系统的阶跃响应

4. impulse 函数

该函数用于绘制连续系统的脉冲响应图，生成一个随机的稳定传递函数模型

$G(s) = \frac{\text{num}(s)}{\text{den}(s)}$ ，写成具体 s 函数形式为

$$G(s) = \frac{20}{s^4 + 8s^3 + 36s^2 + 40s + 20}$$

实现的程序如下所示。

```
num=20;
den=[1 8 36 40 20];
t=1:0.1:10;
impz(num,den,t);
```

绘制出连续系统的脉冲响应如图 1.4.2 所示。

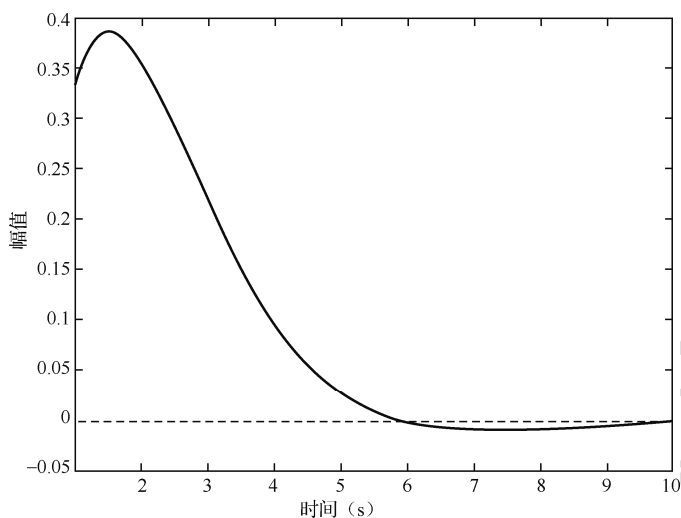


图 1.4.2 连续系统的脉冲响应

掌握以上几种常用的函数，将对后面课程的学习打下良好基础。

1.5 绘图

MATLAB 提供大量的工具用来图形化地显示向量和矩阵，同时也可以注释和打印这些图表。本节将介绍一些重要的图形化函数，及一些典型实例。

1.5.1 创建一个图

根据输入参数的不同，`plot` 函数有不同的形式。如果 y 是一个向量，则 `plot(y)` 输出一个分段的线性图形，来显示 y 的各个元素与其对应下标的关系。如果指定两个向量作为参数，则 `plot(x,y)` 表示 y 是 x 的函数而做出的图。

例如，绘制正弦函数从 0 到 2π 值的图，如图 1.5.1 所示，其中，图 (a) 是 `plot(y)` 绘制出来的，图 (b) 是 `plot(t,y)` 绘制出来的。两幅图的纵坐标是一样的，区别在于横坐标。前者是与 y 值相对应的下标，后者则是与 y 值相对应的 t 值。

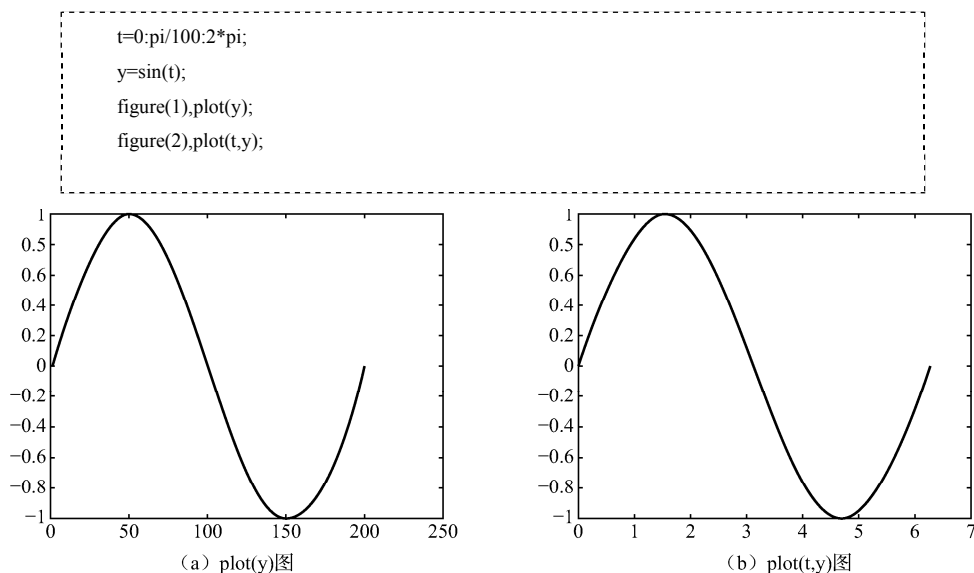


图 1.5.1 正弦函数

通过一个 `plot` 函数，可对多个 x - y 组合绘制多重图。MATLAB 会自动循环使用预定义的颜色列表来区分每组数据（用户也可自定义）。

例如，下面程序是绘制三个 t 的相应正弦函数的图像（见图 1.5.2），每条曲线都用不同的颜色加以区分。

```
t=0:pi/100:2*pi;  
y=sin(t);  
y1=sin(t-0.25);  
y2=sin(t-0.5);  
plot(t,y,'r-',t,y1,'g--',t,y2,'b-.')
```

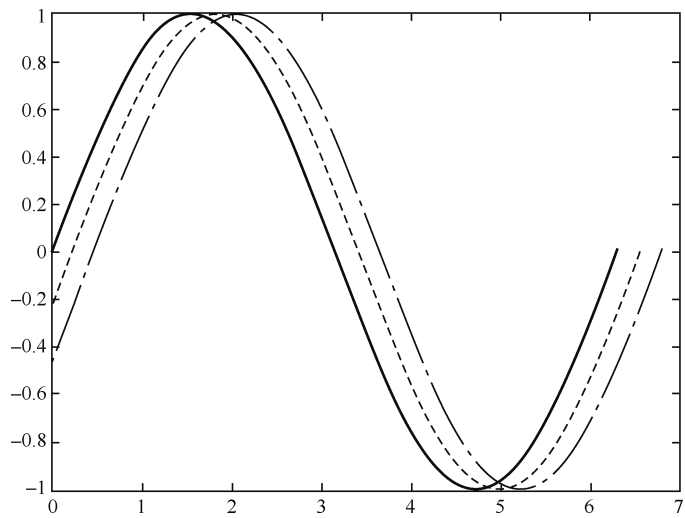


图 1.5.2 三条不同颜色的正弦曲线

曲线的颜色、线型和标记（如加号或圆圈），均可以自定义，即

```
plot(x, y, 'color style marker')
```

color_style_marker 是字符串（由单引号括起来），由颜色、线型和标记构成，详见表 1.5。

表 1.5 曲线设置参数表

曲线颜色		曲线线型		数据点形	
选 项	意 义	选 项	意 义	选 项	意 义
b	蓝色（blue）	—	实线（默认）	.	实点
c	青色（cyan）	:	点线	+	十字形
g	绿色（green）	—·	点断线	O	圆圈
k	黑色（black）	---	虚线	*	星号
m	红紫色（magenta）			×	叉号
r	红色（red）			s	正方形
w	白色（white）			d	菱形
y	黄色（yellow）			h	六角形
				p	五角形
				∨	下三角
				^	上三角
				<	左三角
				>	右三角

1.5.2 图形窗口

如果屏幕上无图形窗口，plot 函数自动打开一个新的图形窗口；若已有图形窗口存在，

plot 函数默认使用该窗口绘图。若要打开一个新的图形窗口，并使它成为当前图形窗口，则需要键入 figure 指令。

为了使已存在的图形窗口成为当前图形窗口，需要键入 figure(n)，n 为图形窗口的标题栏的序号。随后的图形命令结果均显示于该窗口（见图 1.5.3）。

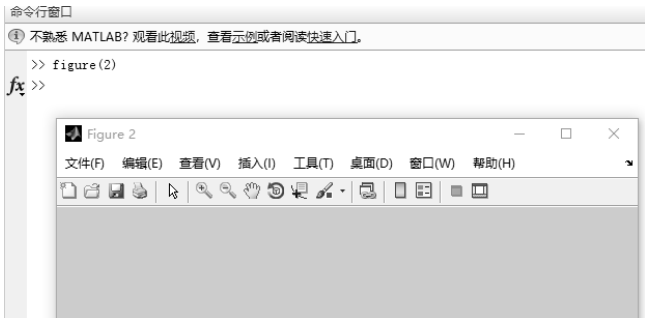


图 1.5.3 figure(2)

1.5.3 添加图形至现有的图

Hold 命令可在已存在的图形窗口中添加图形。如果键入了 hold on 命令，MATLAB 不会移走现有图形，而是把新数据添加到当前图块，必要时会改变坐标轴的标尺。

1.5.4 绘制子图

subplot 函数可在同一窗口显示多幅图形或将其打印到同一张纸上。

输入 subplot(m,n,p)，将图形窗口分成 $m \times n$ 个子区域，并选择第 p 个区域作为当前图形。例如，在 figure 窗口的 4 个不同的子区域中分别作图，如图 1.5.4 所示。

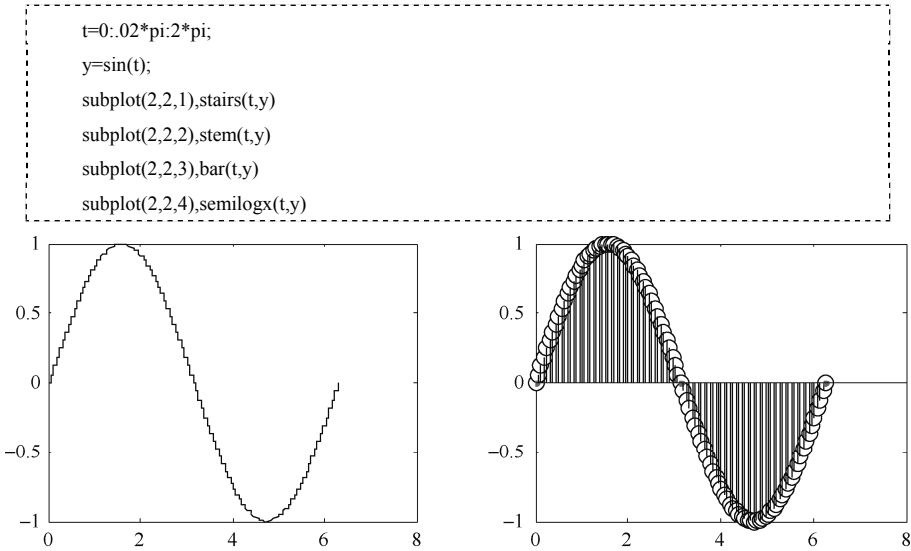


图 1.5.4 4 个子图

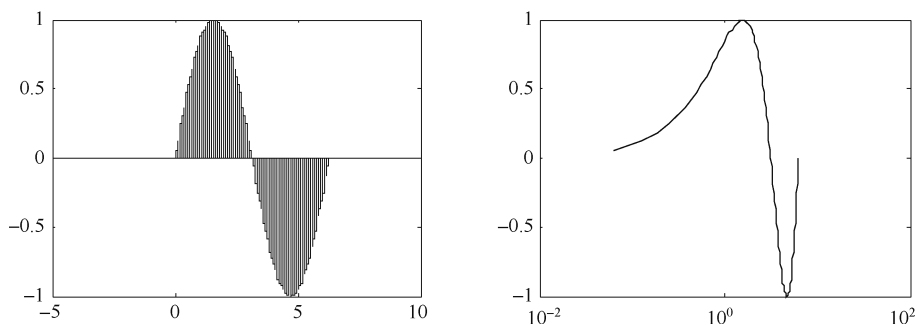


图 1.5.4 4 个子图 (续)

1.5.5 隐函数绘图

`ezplot(fun)` 在默认区间 $-2\pi < x < 2\pi$ 上绘制表达式 $\text{fun}(x)$ 的图像。

`fun` 是一个处理 M 文件函数句柄、一个匿名函数或一个字符串。

`ezplot(fun[min,max])`: 指在区间 $\min < x < \max$ 和 $\min < y < \max$ 上绘制 $\text{fun}(x,y)=0$ 的图像, 如图 1.5.5 所示。

```
ezplot('x*y*sin(x^2+y^2)+(x+y)^2*exp(-(x+y))')
```

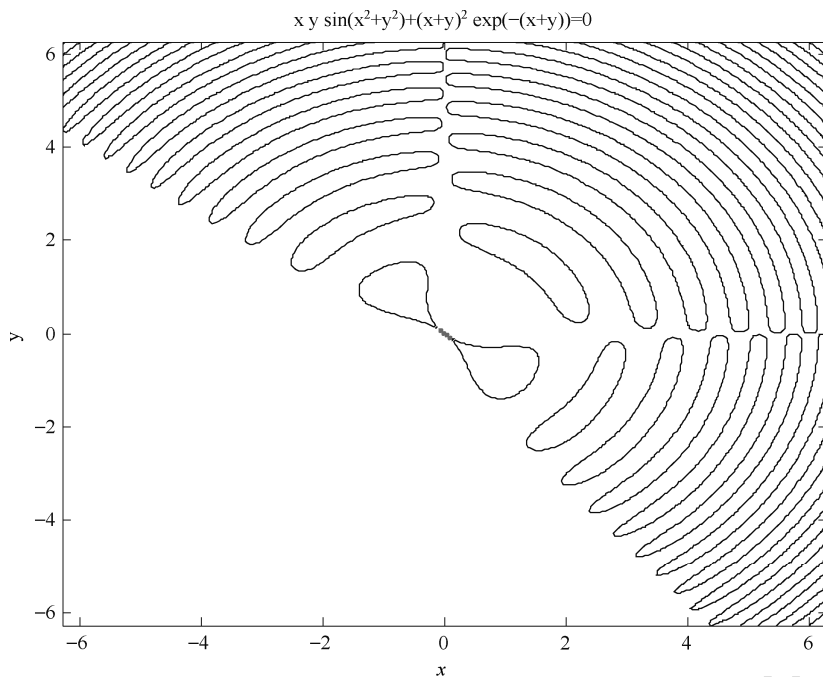


图 1.5.5 隐函数绘图

1.5.6 轴的控制

`axis` 函数包含多种选项, 可自定义图像的缩放比例、方位和纵横比。

通常, MATLAB 可以根据数值的最大值和最小值来选择一个合适的坐标轴范围。axis

函数可以通过自定义坐标轴范围来覆盖默认设置。例如，

`axis([ xmin xmax ymin ymax])`

`axis` 函数也接受一系列的轴控制关键词。

- `axis auto` 返回坐标轴的默认刻度，即自动模式
- `grid off` 关闭网格线
- `grid on` 显示网格线

1.5.7 轴标记和标题

用 `xlabel`、`ylabel` 和 `zlabel` 命令添加 x 、 y 和 z 轴标签。可用 `title` 函数在图形窗口的顶部添加标题，用 `text` 函数在图形内任意位置添加文本。`tex` 标记的子集则产生希腊字母、数学符号和替代字体。详见表 1.6。

表 1.6 图形标注所用特殊字符表

命 令	所代表字符	命 令	所代表字符
<code>\alpha</code>	α	<code>geq</code>	$\geq$
<code>\beta</code>	β	<code>neq</code>	$\neq$
<code>\gamma</code>	γ	<code>\equiv</code>	$\equiv$
<code>\delta</code>	δ	<code>\approx</code>	$\approx$
<code>\omega</code>	ω	<code>\leq</code>	$\leq$
<code>\zeta</code>	ζ	<code>\leftarrow</code>	$\leftarrow$
<code>\eta</code>	η	<code>\uparrow</code>	$\uparrow$
<code>\lambda</code>	λ	<code>\downarrow</code>	$\downarrow$
<code>\xi</code>	ξ	<code>\rightarrow</code>	$\rightarrow$
<code>\pi</code>	π		

例子详见下面程序及图 1.5.6。

```
t=-pi:pi/100:pi;
y=sin(t);
plot(t,y)
axis([-pi pi -1 1])
xlabel('\pi\leq t\leq\pi');
ylabel('sin(t)');
title('Graph of the sine function')
```

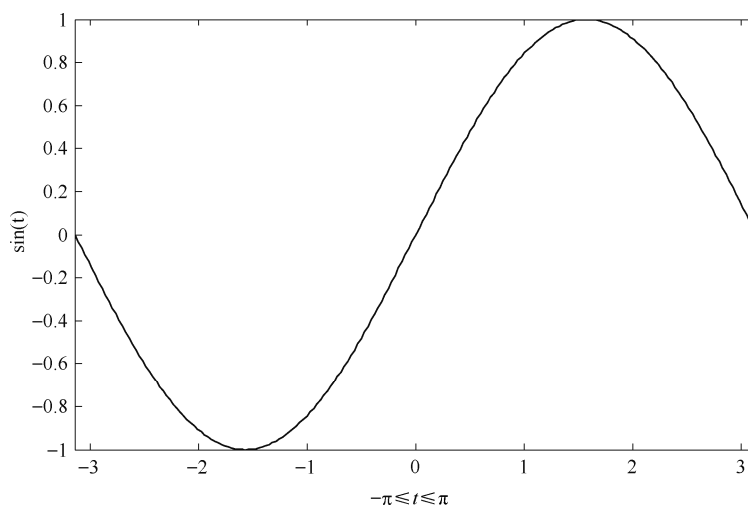


图 1.5.6 标记的正弦曲线图

1.5.8 plot3 和 mesh

plot3 函数用于显示一组数据点的三维图。例子详见下面程序及图 1.5.7。

```
t=0:.02*pi:2*pi;
x=t.^3.*sin(3*t).*exp(-t);
y=t.^3.*cos(3*t).*exp(-t);
z=t.^2;
plot3(x,y,z);grid on
stem3(x,y,z);hold on;
plot3(x,y,z);grid on
```

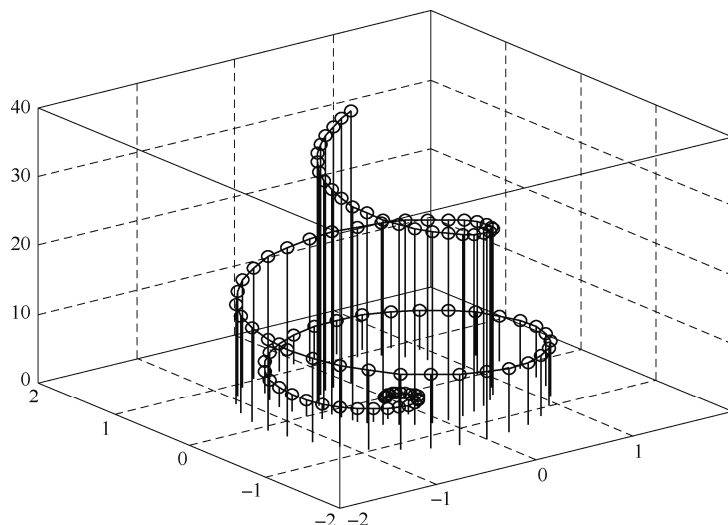


图 1.5.7 plot3 函数绘图

要显示两个变量的函数 $z = f(x,y)$ 图像，分别在函数的定义域，生成各自含有重复的行与

列的矩阵 $\mathbf{X}$ 和 $\mathbf{Y}$ 。然后用矩阵 $\mathbf{X}$ 和 $\mathbf{Y}$ 来计算和绘制函数。函数 `meshgrid` 会将两个向量 $\mathbf{x}$, $\mathbf{y}$ 指定的定义域转换成矩阵 $\mathbf{X}$ 和 $\mathbf{Y}$ ，以用于计算双变量函数。 $\mathbf{X}$ 的行和 $\mathbf{Y}$ 的列分别是 $\mathbf{x}$ 和 $\mathbf{y}$ 向量的副本。

例子详见下面程序及图 1.5.8。

```
[x,y]=meshgrid(-3:0.1:3,-2:0.1:2);  
z=(x.^2-2*x).*exp(-x.^2-y.^2-x.*y);  
mesh(x,y,z)
```

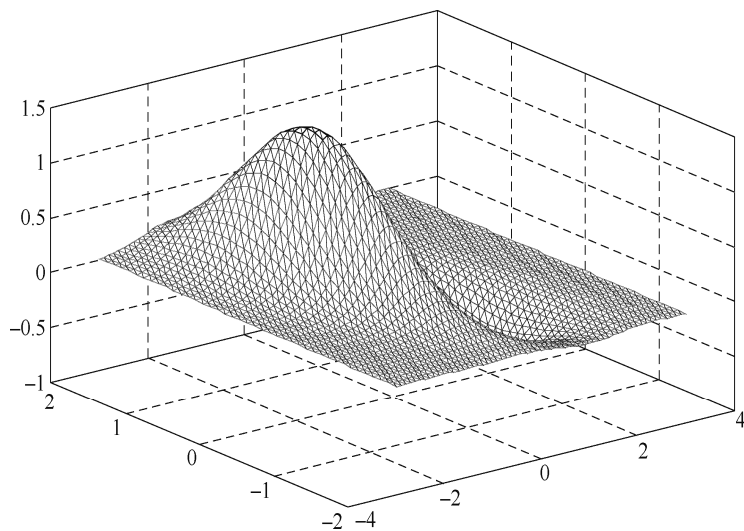


图 1.5.8 mesh 函数绘图

1.5.9 图像

从图形文件读取图像并显示图像。例子详见下面程序及图 1.5.9。

```
I=imread('d:\My Documents\My Pictures\5.jpg');  
imshow(I);  
whos I
```



图 1.5.9 I 图

whos I

Name	Size	Bytes	Class	Attributes
I	839×1024×3	2577408	uint8	

读取彩色图像的 RGB 三个通道并显示，例子详见下面程序及图 1.5.10。

```
I=imread('C:\Users\andy\Pictures\I.jpg');
Rim=I(:,:,1);
Gim=I(:,:,2);
Bim=I(:,:,3);
subplot(2,2,1);
imshow(I),title('I');
subplot(2,2,2);
imshow(Rim),title('Rim')
subplot(2,2,3);
imshow(Gim),title('Gim')
subplot(2,2,4);
imshow(Bim), title('Bim')
```

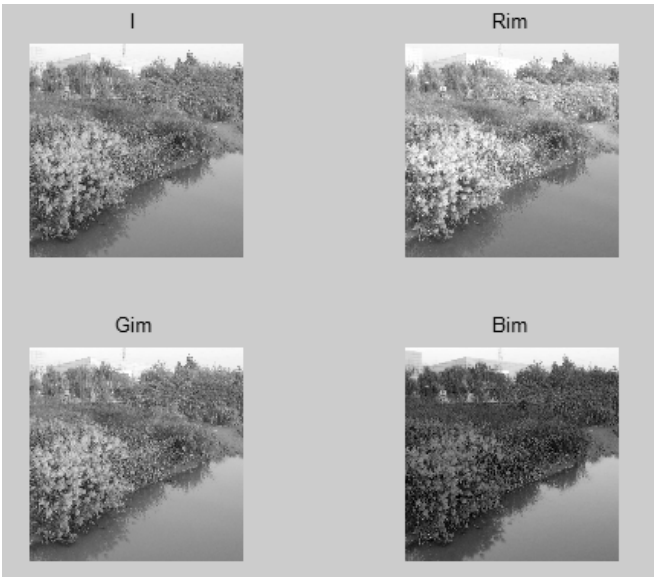


图 1.5.10 RGB 三个通道图像

本章小结

- 1. 本章开始以一个魔方矩阵为例，通过对魔方矩阵的验证，介绍了矩阵的相关知识：矩阵输入、矩阵运算、矩阵操作及 M 文件等。
- 2. 介绍 MATLAB 环境及其常用函数。
- 3. 介绍 MATLAB 二维和三维绘图，以及图像的操作。