附录 A MATLAB 基础

A.1 MATLAB 系统环境

MATLAB 是 Matrix Laboratory（矩阵实验室）的缩写，是由美国 Mathworks 公司推出的一种用于算法开发、数据可视化、数据分析以及数值计算的高级科学计算语言和交互式环境。MATLAB 集合了大量的计算算法，针对许多专门的领域都开发了功能强大的模块集和工具箱（单独提供的专用 MATLAB 函数集），解决了这些应用领域内特定类型的问题，使得它在许多科学领域中成为计算机辅助设计和分析、算法研究和应用开发的基本工具和首选平台，是当今最优秀的科技应用软件之一。

A.1.1 MATLAB 桌面工作环境

MATLAB 为用户提供了交互式的桌面工作环境，了解和熟悉这些工作环境是使用 MATLAB 的基础。启动 MATLAB 后进入如图 A-1 所示的用户界面，包括菜单栏、工具栏、【Start】按钮及若干个工作窗口。

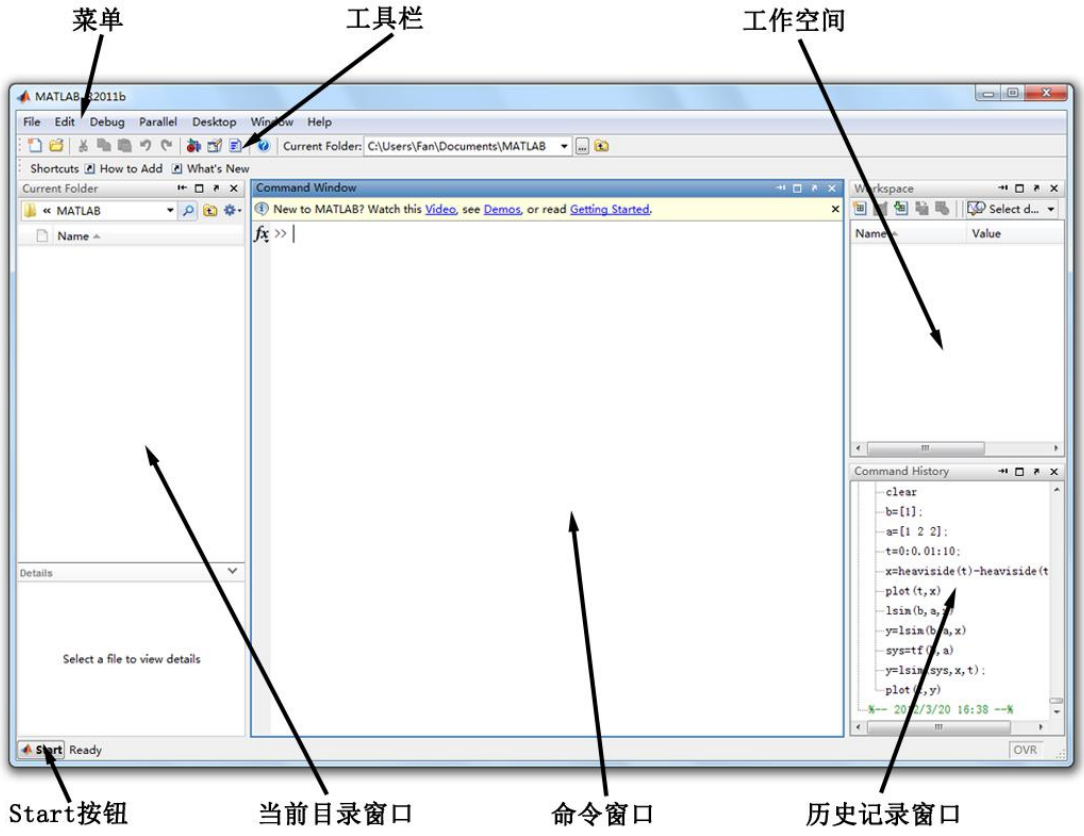

图 A-1 MATLAB 桌面环境

1. 【Start】按钮、菜单栏和工具栏

在用户界面的左下角有一个【Start】按钮，顶端是菜单栏和工具栏，选择其中的命令可以执行 MATLAB 的各种功能和工具。

2. 命令窗口（Command Window）

命令窗口是 MATLAB 的主要交互窗口，用于输入各种命令并显示命令的执行结果。例如我们用 MATLAB 实现一个简单的计算，在命令窗口中输入

```
>> x = 1;
>> y = 2;
>> z = x + y
```

输入每个语句后按回车键执行，最后显示结果为：

```
z =
     3
```

观察上述语句，执行第一句和第二句语句时，命令窗口中并没有输出结果，而执行第三句语句后，却在命令窗口输出了计算结果。这是因为，MATLAB 语句以分号“;”作为语句的结束，则执行该语句后不输出执行结果，如果没有以分号结束，则立即在命令窗口中显示该语句的运行结果。一般情况下， MATLAB 语句应以分号结束，因为在程序运行阶段，大量显示中间结果则会降低程序运行速度，但是在 MATLAB 程序调试阶段，显示某些语句的运行结果有助于程序的调试。

执行 clc 命令或通过【Edit】菜单→【Clear Command Window】，可以清除命令窗中的内容。

3. 工作空间（Workspace）

工作空间用于显示当前 MATLAB 工作内存中所有变量的名称、类型、数据结构、大小等信息。用户可以选中某个变量，单击鼠标右键可以选择对变量进行打开、保存、复制、删除等操作。此外，MATLAB 提供了一些管理和查看工作空间中变量的命令。

clear 命令

clear 命令用于删除工作空间中的变量，用法如下：

clear　变量名　清除工作空间中的指定变量，变量名可以是一个或多个。

clear　清除工作空间中的所有变量。

save 命令和 load 命令

使用 save 命令可以将工作空间中的指定变量或所有变量保存到 mat 文件中，相应的使用 load 命令从文件中将数据调入工作空间：

save　文件名　将工作空间中所有变量保存到名为“文件名.mat”的文件中。

save　文件名 变量名　将工作空间中变量名所列的变量保存到名为“文件名.mat”的文件中，变量名可以是一个或多个。

load 文件名　将保存在“文件名.mat”文件中的所有变量调入到工作空间。

who 命令和 whos 命令

who　查看工作空间中的变量名。

whos　查看工作空间中的变量的变量名、大小、类型和字节。

whos　变量名　查看指定变量的变量名、大小、类型和字节。

4. 历史记录窗口（Command History）

历史记录窗口记录了用户在命令窗口中运行过的所有命令，同时记录了命令运行的日期和时间。用户可以对已执行命令进行操作，选择某条命令单击鼠标右键，在展开的菜单中可以选择需要的操作，如剪切、复制、执行等。

5. 当前目录窗口（Current Directory）

当前目录窗口显示当前的工作目录及该目录下的所有文件，选中其中某个文件，单击鼠

标右键进行可以进行需要的操作，如打开、复制、删除、执行等。在当前目录窗口上方和工具栏右方各有一个当前目录设置区，用于设定当前目录。

A.1.2 MATLAB 帮助系统

MATLAB 为用户提供了丰富的帮助功能，用户可以轻松的获得帮助信息，并通过帮助系统学习和掌握 MATLAB。MATLAB 可以通过帮助命令（help 命令）、帮助窗口、Demo 演示等方式获得帮助。

1. 帮助命令（help 命令）

在 MATLAB 的命令窗口中直接输入 help 命令将会显示当前帮助系统中所包含的所有帮助项目。如果准确知道函数名称，则可以通过输入 help 加函数名称来获取该函数的帮助信息，例如用户如果对 sinc 函数的用法不了解，则可以通过执行 help sinc 命令来获取帮助信息。

```
>> help sinc
 SINC Sin(pi*x)/(pi*x) function.
    SINC(X) returns a matrix whose elements are the sinc of the elements
    of X, i.e.
          y = sin(pi*x)/(pi*x)      if x ~= 0
            = 1                           if x == 0
    where x is an element of the input matrix and y is the resultant
    output element.

    % Example of a sinc function for a linearly spaced vector:
    t = linspace(-5,5);
    y = sinc(t);
    plot(t,y);
    xlabel('Time (sec)');ylabel('Amplitude'); title('Sinc Function')

    See also square, sin, cos, chirp, diric, gauspuls, pulstran, rectpuls,
    and tripuls.

    Reference page in Help browser
       doc sinc
```

2. 帮助窗口

MATLAB 的帮助窗口能够提供友好的交互式帮助界面，供用户浏览帮助信息。通过【Help】菜单→【Products Help】可以进入帮助窗口，如图 A-2 所示。

帮助窗口左侧导航栏中列出了所有的帮助信息目录，只要单击相关内容，逐级查找就可以找到相应的帮助信息。此外，导航栏上方有一个搜索工具条，用户可以输入关键字查询感兴趣的帮助信息。

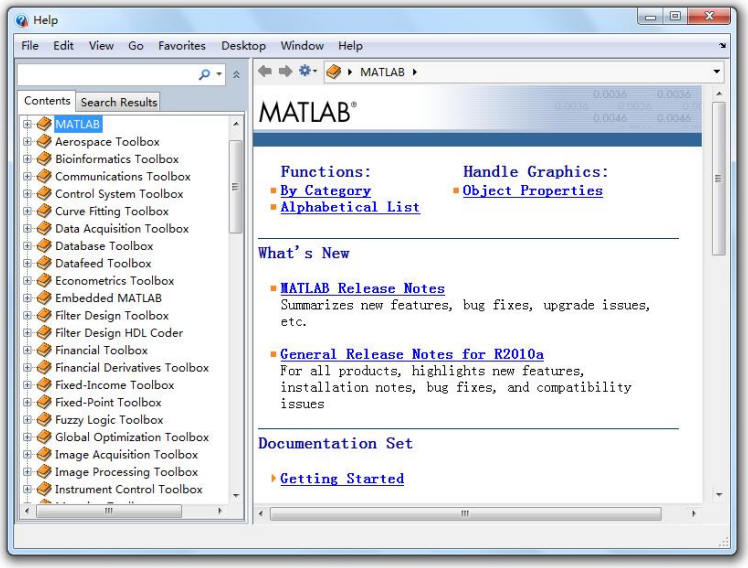

图 A-2 MATLAB 帮助窗口

3. Demo 演示

MATLAB 提供了一些很好的演示程序，这些程序由交互式界面引导，操作方便，示范作用强，是学习和掌握 MATLAB 的有效途径。Demo 演示程序可以从【Help】菜单→【Demos】或帮助窗口左侧导航栏的【Demos】选项卡启动。

从窗口左侧导航栏显示的 Demo 演示程序目录中可以选择要查询的内容，例如选择【ToolBoxes】→【Signal Processing Toolbox】→【Transforms】→【Chirp-z Transform】，右侧浏览器窗口中显示 Chirp-z 变换 Demo 演示程序相关信息（如图 A-3）。点击浏览器窗口左上角的“Open cztdemo.m in the Editor”链接，可以在 MATLAB 文件编辑器中查看该 Demo 演示程序的代码，而点击浏览器窗口右上角的“Run this demo”链接，可以启动 Demo 演示程序，如图 A-4 所示。

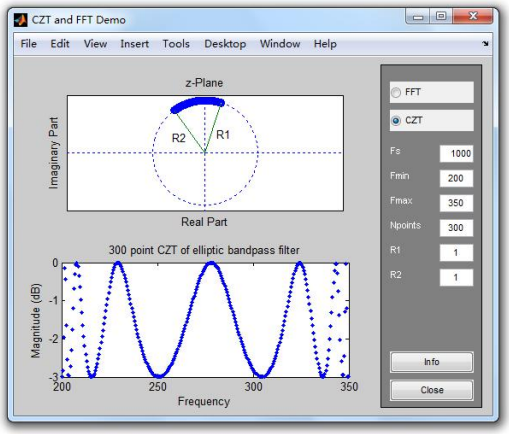

图 A-3　Demo 演示程序信息显示窗口

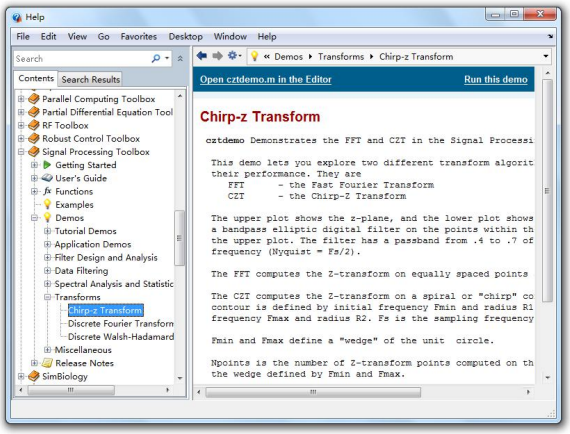

图 A-4 Demo 演示程序运行界面

A.2 MATLAB 的变量与数据类型

A.2.1 变量

MATLAB 的变量的命名需要遵循如下规则：

- ◆ 变量名区分大小写，例如 A 和 a 表示两个不同的变量。
- ◆ 变量名最多不超过 63 个字符（根据 MATLAB 版本的不同，这个数字会有所变化）。
- ◆ 变量名开头必须是英文字母，后面可以接英文字母、下划线、数字，但不能使用空格和标点符号。
- ◆ 不能使用系统函数名和系统保留字。

不同于其他传统编程语言，MATLAB 不要求事先对变量进行声明，也不需要定义变量类型，MATLAB 会根据赋予变量的值或对变量所进行的操作，自动生成变量并确定变量的数据类型和大小。在赋值过程中如果赋值变量已存在，MATLAB 将使用新值替代旧值，并以新值类型替代旧值类型。

MATLAB 赋值语句的基本结构为：

变量 = 数、表达式或函数；

这一过程把等号右边的数、表达式运算结果或函数执行结果赋给左边的变量。例如，在命令窗口中执行如下语句：

```
>> x=sqrt(5)-1
```

执行结果如下：

```
x =
    1.2361
```

这里，函数 sqrt 表示开方运算，在执行这个语句时，MATLAB 自动生成变量 x，并将“sqrt(5)-1”的计算结果赋值给 x。

此外，如果没有指定赋值变量，则结果将赋值给 MATLAB 默认的变量 ans，例如在命令窗口中直接输入表达式“sqrt(5) + 1”，执行结果如下：

```
>> sqrt(5) + 1
ans =
    3.2361
```

MATLAB 提供了一些预先定义好数值的变量，定义了编程和应用中经常用到的数据（如虚数单位、圆周率等），这些特殊的变量称为常量，表 A-1 列出了这些常用的特殊变量。

表 A-1 MATLAB 特殊变量表

变量名	基本意义
ans	默认变量名，MATLAB 将没有指定输出变量的计算结果保存到 ans 变量中
eps	浮点数的相对误差，如果某个量的绝对值小于 eps，可以认为这个量是 0
Inf 或 inf	无穷大，负无穷可以表示为-Inf
i 或 j	虚数单位，即 $\sqrt{-1}$
pi	圆周率
NaN 或 nan	非数值（Not a Numbeu），例如由 0/0、inf/inf 运算所得出的结果
realmax/realmin	最大/最小正实数
nargin/nargout	函数输入/输出变量数目
computer	计算机类型
version	MATLAB 版本字符串

A.2.2 数据类型

MATLAB 支持多种数据类型，包括数值、字符、逻辑、元胞、结构和函数等类型，所有类型的数据都以矩阵的形式保存。

数值类型包括整数型(有符号整数型和无符号整数型)和浮点型(单精度浮点型和双精度浮点型)。MATLAB 中数值的默认数据类型是双精度浮点型。对于复数，则由实部和虚部两个部分组成，用 i 或 j 表示需数单位，例如可以通过如下命令创建一个复数：

```
>> x = 1 + 2i
x =
    1.0000 + 2.0000i
```

字符类型用来表示字符和字符串，一般用在字符数组中，每个字符都有对应的 ASCII 数值，用一个 16 位数据表示。MATLAB 中字符串用单引号引用到程序中，例如创建一个字符串的代码及执行结果如下：

```
>> str = 'An example of String'
str =
     An example of String
```

逻辑类型是用 0 和 1 表示逻辑假和逻辑真。

元胞类型是 MATLAB 中比较特殊一种的数据类型，是元胞数组的基本单位，元胞可以是不同类型和大小的数据，这样可以将不同类型的数据集中在一个变量中。例如，创建一个 2×2 的元胞数组，其中 4 个元胞分别为标量、数值类型数组、字符串和元胞数组：

```
>> A = {1,ones(3);'string',cell(2,2)}
A =
     [        1]      [3x3 double]
     'string'        {2x2 cell    }
```

用{}可以访问元胞的值，例如 A{1,2}表示元胞数组 A 第 1 行第 2 列的元胞

```
>> A{1,2}
ans =
     1     1     1
     1     1     1
     1     1     1
```

结构类型也是 MATLAB 中比较特殊一种的数据类型，和元胞数组有许多类似之处，两者都可以在同一个变量中存放不同类型数据。结构采用“.”来访问数据，例如创建一个名为 family 结构来保存家庭信息，代码及执行结果如下：

```
>> family.name = 'my family';
>> family.number = 3;
>> family.people = 'father,mother,me'
family =
          name: 'my family'
        number: 3
        people: 'father,mother,me'
```

函数句柄用于间接调用一个函数的 MATLAB 值或数据类型，创建函数句柄后，可以通过句柄来实现函数功能。

A.3 矩阵

正如 MATLAB 一词的本意（Matrix Laboratory，矩阵实验室），MATLAB 的最大特色是强大的矩阵计算功能，矩阵是 MATLAB 进行数值计算最基本的运算单元。

A.3.1 矩阵的建立

1. 直接输入法

对于较小的简单的矩阵，可以通过直接输入矩阵的元素来创建矩阵，输入时矩阵的元素必须包含在方括号中，同一行中各元素之间以逗号或空格分开，不同行则以分号隔开。例如：

```
>> A = [1 2 3;4 5 6]
A =
     1     2     3
     4     5     6
```

矩阵中元素也可以用表达式代替，例如：

```
>> X=[-1.3,sqrt(4),(1+2+3)/5*4]
X =
   -1.3000    2.0000    4.8000
```

2. 由已知矩阵进行运算或拼接

可以由已知矩阵运算生成新的矩阵，例如对于上述 X 矩阵：

```
>> Y = 2 * X
Y =
   -2.6000    4.0000    9.6000
```

多个已知矩阵可以拼接成一个矩新的阵，例如：

```
>> Z = [X;Y]
Z =
   -1.3000    2.0000    4.8000
   -2.6000    4.0000    9.6000
```

3. 通过函数生成

MATLAB 提供了多个特殊矩阵的生成函数，下面列出一些常用的特殊矩阵函数。

1）单位矩阵

主对角线元素为 1，其他元素均为 0。

A = eye(n)　返回一个 n×n 阶的单位矩阵

A = eye(m,n)　返回一个 m×n 阶的单位矩阵

A = eye(size(B))　返回一个大小与 B 一样的单位矩阵

例如：

```
>> A = eye(3)
A =
     1     0     0
     0     1     0
     0     0     1
```

2）“0”矩阵

矩阵或数组所有元素为 0。

A = zeros(n)　返回一个 n×n 阶的 0 矩阵
A = zeros(m,n)　返回一个 m×n 阶的 0 矩阵
A = zeros(d1,d2,d3,…)　返回一个 d1×d2×d3×…阶的 0 矩阵
A = zeros(size(B))　返回一个大小与 B 一样的的 0 矩阵或数组
例如：

```
>> A = zeros(3,2)
A =
     0     0
     0     0
     0     0
```

3）“1”矩阵

矩阵或数组所有元素为 1。
A = ones(n)　返回一个 n×n 阶的 1 矩阵
A = ones(m,n)　返回一个 m×n 阶的 1 矩阵
A = ones(d1,d2,d3,…)　返回一个 d1×d2×d3×…阶的 1 矩阵
A = ones(size(B))　返回一个大小与 B 一样的的 1 矩阵或数组
例如：

```
>> A = ones(2,3)
A =
     1     1     1
     1     1     1
```

4）随机矩阵

其元素是随机产生的。
rand 函数，产生元素在[0，1]之间服从均匀分布的随机数数组或矩阵。
randn 函数，产生元素服从均值为 0，方差为 1 的正态分布的随机数数组或矩阵。
rand 函数与 randn 函数用法相同。
A = rand(n)　返回一个 n×n 阶的随机数矩阵
A = rand (m,n)　返回一个 m×n 阶的随机数矩阵
A = rand (d1,d2,d3,…)　返回一个 d1×d2×d3×…阶的随机数矩阵
A = rand (size(B))　返回一个大小与 B 一样的的随机数矩阵或数组
例如：

```
>> A = rand(3)
A =
    0.8147    0.6324    0.9575
    0.9058    0.0975    0.9649
    0.1270    0.2785    0.1576
```

5）线性间隔向量

产生线性增量的向量。
V = linspace(a,b)　产生一个在 a，b 间线性间隔的 100 点行向量。
V = linspace(a,b,n)　产生一个在 a，b 间线性间隔的 n 点行向量。
例如：

```
>> V = linspace(1,10,6)
V =
    1.0000    2.8000    4.6000    6.4000    8.2000   10.0000
```

6）对数间隔向量

产生对数增量的向量；对于产生频率向量特别有用。

V = logspace(a,b)　产生一个在间的 50 个对数间隔点的行向量。

V = logspace(a,b,n)　产生一个在间的 n 个对数间隔点的行向量。

V = logspace(a,pi)　产生一个在间的 50 个对数间隔点的行向量。

例如：

```
>> V = logspace(1,5,6)
V =
   1.0e+005 *
     0.0001    0.0006    0.0040    0.0251    0.1585    1.0000
```

A.3.2 冒号表达式

在 MATLAB 中冒号“:”是一个特殊的运算符，利用它可以产生行向量。冒号表达式的一般格式为 x = n1:n0:n2，表示产生一个从 n1 开始到 n2 结束，步长为 n0 的向量，n0 可以为负数。如果略去式中 n0 一项，此时默认步长为 1。例如：

```
>> x = 1:2:11
x =
     1     3     5     7     9    11
>> x=5:-1:1
x =
     5     4     3     2     1
>> x = 1:5
x =
     1     2     3     4     5
```

A.3.3 矩阵的拆分

1. 矩阵元素

MATLAB 允许对矩阵的单个元素进行赋值和操作，矩阵的元素可以通过元素的下标来访问，例如 A(a,b)表示矩阵 A 的第 a 行第 b 列元素，向量用一个下标，例如 x(a)表示向量 x 的第 a 个元素。这里需要特别注意的是，MATLAB 的下标从 1 开始，例如矩阵 A 的第 1 行第 1 列元素为 A (1,1)，而不是 A (0,0)。

2. 矩阵拆分

NATLAB 中可以利用冒号表达式对矩阵进行拆分获取子矩阵：

A(:,b)　表示取矩阵 A 第 b 列的元素；

A(a,:)　表示取矩阵 A 第 a 行的元素；

A(a:a+m,:)　表示取矩阵 A 第 a 行至第 a+m 行的元素；

A(:,b:b+n)　表示取矩阵 A 第 b 列至第 b+n 列的元素；

A(a:a+m,b:b+n)　表示取矩阵 A 第 a 行至第 a+m 行、第 b 列至第 b+n 列范围内的元素；

我们来看几个例子：

```
>> A = [1 2 3;4 5 6;7 8 9]
A =
     1     2     3
     4     5     6
```

```
     7     8     9
>>   B = A(1,:)
B =
     1     2     3
>> C = A(:,[1,3])
C =
     1     3
     4     6
     7     9
>> D = A(2:3,:)
D =
     4     5     6
     7     8     9
```

其中 A(1,:)中“1”表示矩阵 A 的第 1 行，“:”表示所有列。A(:,[1,3])中“:”表示矩阵 A 的所有行，“[1,3]”表示第 1 列和第 3 列；A(2:3,:)中“2:3”表示矩阵 A 的 2 到 3 行，“:”表示所有列。

A.3.4 矩阵的运算函数

MATLAB 提供了多种关于矩阵运算的函数，表 A-2 列出了一些常用的矩阵运算函数。

表 A-2 常用的矩阵运算函数

函数名	功能	函数名	功能
length	返回向量的长度	size	返回矩阵各维的大小
det	计算矩阵的行列式	fliplr	矩阵翻转
inv	矩阵求逆	svd	矩阵的奇异值分解
rank	计算矩阵的秩	max	矩阵元素求最大值
trace	计算矩阵的迹	min	矩阵元素求最小值
eig	矩阵的特征值和特征向量	sum	矩阵元素求和
poly	矩阵的特征多项式	mean	矩阵元素求平均值

A.4 MATLAB 数值运算

A.4.1 算术运算

MATLAB 提供了一系列运算符用于算术运算，如表 A-3 所示。

表 A-3 算术运算符

运算符	功能	运算符	功能
+	加法	-	减法
*	乘法	.*	点乘
/ 和 \	右除和左除	./ 和 .\	点除
^	乘方	.^	点乘方
'	转置		

1. 加减法

运算符“+”和“-”用于实现加法和减法运算，进行加法和减法的两个矩阵必须有相同的维数，除非其中一个是标量。标量可以任意维数的矩阵相加或相减，表示标量与矩阵的每一个元素相加或相减。例如：

```
>> A = [1 2 3;4 5 6];
>> B = [2 3 4;5 4 3];
>> A+B
ans =
     3     5     7
     9     9     9
>> A-B
ans =
    -1    -1    -1
    -1     1     3
>> A-1
ans =
     0     1     2
     3     4     5
>> 1+A
ans =
     2     3     4
     5     6     7
```

2. 乘法

“*”用于乘法运算，矩阵相乘与线性代数中的定义一致，设 A 为 m×n 矩阵，B 为 n×r 矩阵，则 A 与 B 的乘积 C 为 m×r 矩阵。若矩阵与标量相乘，表示标量与矩阵的每一个元素相乘。

```
>> A = [1 2 3;4 5 6];
>> B = [2 3 4;5 4 3;5 6 7];
>> C = A*B
C =
    27    29    31
    63    68    73
>> D = 3*A
D =
     3     6     9
    12    15    18
```

3. 除法

左除法（\）　A\B 表示 $A^{-1}*B$。

右除法（/）　A/B 表示 $A*B^{-1}$。

4. 乘方

乘方的运算符为“^”，对于矩阵的乘方运算可以表示为 A^x，要求 A 为方阵，x 为标量。对于标量 a，a^x 表示 a 的 x 次方。

5. 点运算

MATLAB 中专门设计了一种点运算，点运算符有“.*”、“./”、“.\”和“.^”，两个矩阵进行点运算表示它们的对应元素进行相关运算，要求进行运算的两个矩阵维数相同。例如对于矩阵 A 和 B：

```
>> A = [1 2 3;4 5 6];
>> B = [2 3 4;5 4 3];
>> A.*B
ans =
     2     6    12
    20    20    18
```

如果矩阵与标量进行点运算，则表示矩阵每一个元素分别与标量进行相应的运算，例如：

```
>> x=[2 3];
>> y=x.^2
y =
     4     9
```

6. 转置

实矩阵的转置用运算符“′”来实现。例如：

```
>> A = [1 2 3;4 5 6];
>> A′
ans =
1     4
2     5
3     6
```

对于复矩阵，运算符“′”表示复共轭转置。

A.4.2 关系运算和逻辑运算

关系运算符主要用于比较两个对象之间是否满足某种关系，若满足则比较结果为真，返回数值 1，若不满足则比较结果为假，返回数值 0。表 A-4 列出了 MATLAB 的关系运算符。

表 A-4 关系运算符

运算符	功能	运算符	功能
==	等于	<=	小于等于
<	小于	>=	大于等于
>	大于	~=	不等于

逻辑运算在计算机语言中也是普遍存在的，主要功能是判断参与比较的对象之间的某种逻辑关系，表 A-5 列出了 MATLAB 中的逻辑运算符及函数。

表 A-5 逻辑运算符和逻辑运算函数

	逻辑“与”	逻辑“或”	逻辑“非”	逻辑“异或”
逻辑运算符	C = A & B	C = A | B	C = ~A	
逻辑函数	C = and(A,B)	C = or(A,B)	C = not(A)	C = xor(A,B)
说　明	当 A、B 同时为真时，C 为真；否则 C 为假。	当 A、B 中至少有一个为真，C 为真；否则 C 为假。	若 A 为真，C 为假；否则 C 为真。	当 A，B 中只有一个为真，C 为真；否则 C 为假。

除了以上的逻辑运算符及函数以外，NATLAB 还提供了其他的一些逻辑函数，具体的函数和说明如表 A-6 所示。

表 A-6 常用逻辑函数

逻辑函数	说　　明
all	检查向量中的元素是否全为真，如果是则返回 1，否则返回 0；对于矩阵，则按列进行检查，返回元素为 0 或 1 的行向量。
any	检查向量中是否有非 0 元素，如果有则返回 1，否则返回 0；对于矩阵，则按列进行检查，返回元素为 0 或 1 的行向量。
isifinite	检查元素是否为有限值，如果是则返回 1，否则返回 0。
isnan	检查元素是否为 NAN，如果是则返回 1，否则返回 0。
isinf	检查元素是否为无限值，如果是则返回 1，否则返回 0。

A.4.3 复数及其运算

一个复数可以表示为 $x = a + jb$，其中 a 称为实部，b 称为虚部，j 表示虚数单位。

MATLAB 中用 i 和 j 表示虚数单位，可以用实部虚部的形式直接构造复数，例如：

```
>> x = 1 + 2*i
x =
    1.0000 + 2.0000i
```

将复数作为矩阵元素构造的矩阵称为复数矩阵，例如：

```
>> A = [1+i 1-i;1+2*i 1-2*i]
A =
    1.0000 + 1.0000i    1.0000 - 1.0000i
    1.0000 + 2.0000i    1.0000 - 2.0000i
```

MATLAB 提供了一些方便的函数用于复数的操作，包括实部虚部分解、求模、求相角等，具体的函数和说明如表 A-7 所示。

表 A-7 常用的复数操作函数

函数名	功能	函数名	功能
real	求复数或复数矩阵的实部	abs	求复数或复数矩阵的模
imag	求复数或复数矩阵的虚部	angle	求复数或复数矩阵的相角
conj	求复数或复数矩阵的共轭		

A.4.4 多项式运算

1. 多项式的 MATLAB 表示

在 MATLAB 中，多项式由一个行向量表示，行向量由多项式的系数按降幂的次序排列组成。例如多项式

$$A(s) = a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0$$

可以用向量 $A = [a_n, a_{n-1}, \cdots, a_1, a_0]$ 表示。

2. 多项式的运算

1）多项式加减法

如果表示两个多项式的向量长度相同，即两个多项式阶次相同，直接使用“+”、“-”运算符实现加减运算。如果两个多项式阶次不同，必须先将表示低阶多项式的向量补零，使之与高阶多项式具有相同的阶次，然后进行加减运算。

例如求多项式 s^2+2s+1 与 s^3-s^2+2s+3 的和可以表示为：

```
>> p1 = [0 1 2 1];
>> p2 = [1 -1 2 3];
>> p3 = p1 + p2
p3 =
     1     0     4     4
```

即运算结果为 s^3+4s+4 。

2）多项式相乘

函数 conv 用于求两个多项式的乘积多项式，调用格式为 C = conv(A,B)，其中 A 和 B 为表示多项式的行向量，C 返回乘积多项式的系数向量。

例如求多项式 s^2+3s+2 与 s^2+2s+1 的乘积可以表示为：

```
>> A = [1 3 2];
>> B = [1 2 1];
>> C = conv(A,B)
C =
     1     5     9     7     2
```

即运算结果为 $s^4+5s^3+9s^2+7s+2$ 。

3）多项式相除

函数 deconv 用于求两个多项式的相除运算，调用格式为[A,t] = deconv(C,B)，其中 C 为被除多项式，B 为除数多项式，A 为商多项式，t 为余数多项式。

4）多项式求根

多项式求根可以由函数 roots 实现，调用格式为 r = roots(A)，其中 A 为表示多项式的行向量，r 返回由列向量表示的多项式的根。例如，求多项式 s^2-3s+2 的根，可以执行如下命令：

```
>> A = [1 -3 2];
>> r = roots(A)
r =
     2
     1
```

函数 poly 可以用于由给定的根，创建多项式，调用格式为 A = poly(r)，其中 r 为行向量或列向量表示的多项式的根，A 返回多项式系数向量，例如：

```
>> r = [2;1];
>> A = poly(r)
A =
     1    -3     2
```

可见，roots 和 poly 互为逆运算。

5）多项式求值

函数 polyval 用来计算多项式的值，调用格式为 y = polyval(p,x)，此时返回由向量 p 表

示的多项式在 x 处的值，x 也可以为向量或矩阵，函数 polyval 把 x 的元素逐个代入多项式求值。例如求多项式 $2s^2+3s+4$ 当 $s=2$ 时的值可以通过如下命令实现：

```
>> p = [2 3 4];
>> y = polyval(p,2)
y =
      18
```

A.4.5 数学函数

常用的基本数学函数见表 A-8。

表 A-8 基本数学函数

函数名	功能	函数名	功能
abs	实数的绝对值和复数的模	exp	指数
acos	反余弦	fix	朝 0 方向取整
acosh	反双曲余弦	floor	朝负无穷方向取整
acot	反余切	gcd	最大公因子
acoth	反双曲余切	imag	取出复数的虚部
acsc	反余割	lcm	最小公倍数
acsch	反双曲余割	log	自然对数
angle	相角	log2	基为 2 的对数
asec	反正割	log10	常用对数
asech	反双曲正割	mod	求余
asin	反正弦	nchoosek	求矢量元素的全部的组合
asinh	反双曲正弦	real	复数的实部
atan	反正切	rem	除法的余数
atanh	反双曲正切	round	四舍五入取整
atan2	四象限反正切	sec	正割
ceil	朝正无穷方向取整	sech	双曲正割
conj	复共轭	sign	符号函数
cos	余弦	sin	正弦
cosh	双曲余弦	sinh	双曲正弦
cot	余切	sqrt	平方根
coth	双曲余切	tan	正切
csc	余割	tanh	双曲正切
csch	双曲余割		

A.5 MATLAB 符号运算

科学计算中，除了数值计算之外，计算式中带有变量或表达式的抽象运算也占了相当的

的比例，MATLAB 的符号数学工具箱提供了符号运算的功能来解决这一类问题，其运算对象为符号对象，例如符号常量、符号变量和数学表达式等。

这里，我们通过一个例子来初步了解符号运算。例如，对于函数 $f(x)=\sin x$，用 MATLAB 求解它的微分的解析表达式。这时我们用前面介绍的数值计算方法是无法实现的，而 MATLAB 的符号运算提供了一种新的符号对象数据类型，用来储存符号表达式。我们来看看这个例子如何利用符号运算解决：

```
>> f = sym('sin(x)');   %定义符号函数 f(x)
>> dif = diff(f)    %求 df(x)/dx
dif =
cos(x)
```

可以求得 $f(x)$ 的微分为 $\cos(x)$，上述运算的特点是严格按照代数、微积分的计算法则、公式进行计算，并尽给出解析表达式，它的运算是以推理解析的方式进行的。

A.5.1 符号对象

MATLAB 中 sym 命令用于创建符号对象，其调用格式为

符号对象名=sym(符号字符串)

符号字符串可以是常量、变量、函数或表达式。我们先通过一个实例来看看符号对象和数值的区别。例如，我们采用数值计算的方式来计算 $\sqrt{3}$：

```
>> sqrt(3)
ans =
    1.7321
```

然后我们来看采用符号对象的结果，采用 sym 命令创建符号对象，然后使用 sqrt 函数

```
>> x = sym(3)
x =
3
>> y = sqrt(x)
y =
3^(1/2)
```

可以看到得到的是一个符号表达式而不是数值，这个表达式是用字符串来存储的，要获得该符号表达式的数值结果，可以执行如下操作：

```
>> double(y)
ans =
    1.7321
```

下面的例子中采用 sym 命令分别创建了一个符号变量、一个表达式和一个符号方程：

```
>>    x = sym('x')
x =
x
>> f = sym('x^2+1')
f =
x^2 + 1
>> f=sym('x^2+2*x+1=0')
f =
x^2 + 2*x + 1 = 0
```

创建符号表达式以后可以对它进行各种计算，例如：

```
>> g= sym('x^2+1');
>> f = g^2
f =
(x^2 + 1)^2
```

syms 命令是建立符号变量的一种简洁方法，syms 命令的调用格式为

syms var1 var2 …

多个变量之间要用空格隔开，上述命令等同于：

```
var1 = sym(var1 );
var2 = sym(var2 );   …
```

例如：

```
>> syms x
```

等同于

```
>> x = sym('x');
```

多个变量时：

```
>> syms x a
```

等同于

```
>> x = sym('x') ;
>> a = sym('a') ;
```

使用已定义的符号变量可以组成符号表达式或符号方程，例如：

```
>> syms a b c x;
>> f = a*x^2+b*x+c
f =
a*x^2+b*x+c
```

A.5.2 符号的基本运算

1. 算术运算

在 MATLAB 中，符号运算的运算符和基本函数，与数值计算中的运算符和基本函数几乎完全相同，例如“+”、“-”、“*”、“/”、“^”等都可以直接使用,一些数学函数也可以直接使用。例如求符号表达式 ax^2+bx+c 和 x^2+x+1 的和：

```
>> syms a b c x;
>> f1 = sym('a*x^2+b*x+c')
f1 =
    a*x^2+b*x+c
>> f2 = sym('x^2+x+1')
f2 =
    x^2+x+1
>> f 3= f1 + f2
f 3=
c + x + b*x + a*x^2 + x^2 + 1
```

求 e^{x^2+x+1}：

```
>> f4=exp(f2)
```

f4 =

exp(x^2 + x + 1)

2. 符号微积分运算

符号表达式微分运算可以由命令 diff 来实现，积分运算可以由命令 int 来实现，调用格式分别如下：

diff(f)　符号表达式 f 对默认变量进行微分运算。

diff(f,x)　符号表达式 f 对指定变量 x 进行微分运算。

diff(f,n)　计算符号表达式 f 对默认变量或指定变量的 n 阶微分运算。

diff(f,x,n)　符号表达式 f 对指定变量 x 进行 n 阶微分运算。

int(f)　符号表达式 f 对默认变量的积分。

int (f,x)　符号表达式 f 对指定变量 x 的积分。

int(f,a,b)　符号表达式 f 对默认变量的积分，积分区间为[a,b]，a、b 为数值。

int(f,x,a,b)　符号表达式 f 对指定变量 x 的积分，，积分区间为[a,b]，a、b 为数值。

例如对符号表达式 ax^2+bx+c 求微分：

```
>> syms a b c x;
>> f = a*x^2+b*x+c
f =
a*x^2+b*x+c
>> g = diff(f,x)
g =
b + 2*a*x
```

对符号表达式 $\sin(s+3x)$ 求 $[\pi/2,\pi]$ 区间的积分：

```
>> syms x s;
>> f = sin(s+3*x)
f =
sin(s+3*x)
>> g = int(f,pi/2,pi)
g =
cos(s)/3 + sin(s)/3
```

3. 查询符号对象中的符号变量

MATLAB 提供了 findsym 函数用于查询符号对象中包含的符号变量，调用格式为

findsym(s,n)　返回符号对象 s 中的 n 个符号变量，如果不指定 n，则返回 s 中的全部符号变量。

例如：

```
>> syms a b c x;
>> f = a*x^2+b*x+c;
>> findsym(f)
ans =
a,b,c,x
```

可见，符号表达式 f 中使用了 4 个符号变量，分别为 a、b、c 和 x。

4. 提取有理式的分子和分母

如果符号表达式是一个有理式或可以展开为有理式，可利用 numden 函数来提取符号表达式的分子或分母，其调用格式为

[n,d] = numden(s)　分别提取符号表达式 s 的分子和分母，存放在 n 和 d 中。

例如：

```
>> syms x y              %创建符号变量
>> f=x/y+y/x;            %创建符号表达式
>> [n,d] = numden(f)     %提取有理式的分子和分母
n =
x^2 + y^2
d =
x*y
```

5. 因式分解与展开

factor(s)　对符号表达式 s 分解因式

例如：

```
>> f = sym(x^2+3*x+2)
f =
x^2 + 3*x + 2
>> f = factor(f)
f =
(x + 2)*(x + 1)
```

expand(s)　对符号表达式 s 进行展开

例如：

```
>> syms a x y;
>> f = sym('a*(x+y)')
f =
a*(x+y)
>> f = expand(f)
f =
a*x + a*y
```

collect(s)　对符号表达式 s 合并同类项

collect(s,v)　对符号表达式 s 按变量 v 合并同类项

例如：

```
>> syms x;
>> f = sym('(x-1)*(x-2)*(x-3)')
f =
(x-1)*(x-2)*(x-3)
>> f = collect(f)
f =
x^3 - 6*x^2 + 11*x – 6
```

6. 符号表达式化简

simplify(s)　应用函数规则对 s 进行化简

simple(s)　调用 MATLAB 的其他函数对表达式 s 进行综合化简，并显示化简过程。

例如：

```
>> syms x;
>> f = sym('sin(x)^2 + cos(x)^2')
f =
sin(x)^2+cos(x)^2
>> f = simplify(f)
f =
1
```

7. 变量替换

sub 命令用于符号表达式中进行变量替换，调用格式为

sub(f,x,y)　表示将符号表达式 f 中的 x 替换 y。

例如:

```
>> syms x s;
>> f = sym('x^2+1')
f =
      x^2+1
>> f = subs(f,x,s)
f =
      a^2 + 1
```

A.6 MATLAB 数据的图形可视化

科学计算中常常需要以图形的形式对数据进行显示和分析，MATLAB 在数据可视化方面提供了强大的功能，可以给出数据的二维、三维的图形表现，通过对图形线型、立面、色彩、渲染、光线、视角等的控制，可把数据的特征表现得淋漓尽致。

A.6.1 图形绘制基本函数

1. 二维图形绘制

1）plot 函数

plot 函数是 MATLAB 中最基本且应用最广泛的绘图函数，其基本格式如下：

plot(x)　当 x 为向量时，以该向量元素下标为横坐标，向量元素为纵坐标绘制连续曲线；当 x 为矩阵时，表示分别为每列元素绘制曲线。

plot(x,y)　当 x,y 为同维向量时，以 x 的元素为横坐标，y 的元素为纵坐标绘制连续曲线；当 x 为向量，y 为有一维与 x 等维的矩阵，绘制出多条不同色彩的曲线，曲线数目与 y 的另一维维数相同，这些曲线都以 x 的元素为横坐标；当 x 为矩阵,y 为与 x 的一维等维的向量，情况相似,只是曲线都以 y 的元素为纵坐标；当 x、y 为同维数矩阵，则以 x、y 的对应列为横、纵坐标分别绘制曲线。

plot(x,y,'s')　字符串 s 设定曲线的线型和色彩，表 A-9 列出了设定线型和颜色的字符串，利用表中这些选项可以把同一窗口中不同的曲线设置为不同的线型和颜色，这些字符串可以单独使用也可以组合使用，例如字符串'g'表示采用绿色绘制曲线，'--r'表示绘制红色的虚线，':yx'表示绘制黄色点线，同时用叉号标记数据点。当不指定该参数时则使用默认值，线型和色彩由 MATLAB 的默认设置来确定。

plot(x1,y1,'s1',x2,y2,'s2',…)　为每一组(x,y,'s')绘图，每一组数据的绘图规则与上述介绍相同。

表 A-9 颜色、线型与数据点标记符号设定字符串

颜色		线型		数据点标记符号			
字符串	说明	字符串	说明	字符串	说明	字符串	说明
'r'	红色	'-'	实线	'.'	点号	's'	小正方形
'g'	绿色	'--'	虚线	'*'	星号	'd'	菱形
'b'	蓝色	':'	点线	'o'	圆圈	'v'	下三角
'y'	黄色	'-.'	点划线	'x'	叉号	'^'	上三角
'm'	洋红			'+'	加号	'<'	左三角
'c'	青色					'>'	右三角
'w'	白色					'h'	六角形
'k'	黑色					'p'	五角形

以下为使用 plot 命令绘图的一个实例：

```
>> t=0:0.1:2*pi;
>> x=sin(t);
>> plot(t,x);
```

上述代码执行的结果如图 A-5 所示。

2）stem 函数

stem 命令用与绘制离散序列的杆状图形，常用的形式有：

stem(x)　以向量 x 的下标为横坐标绘制离散序列的杆状图形，图上数据点以圆圈标出。

stem(x,y)　以向量 x 为横坐标,绘制离散序列 y 的杆状图形。

stem(x,y,'filled')　用填充图的方式绘制离散序列的杆状图形。

stem(x,y,'s')　用选定的线型绘制离散数据序列图形，字符串 s 的含义与上述 plot 函数中的介绍相同。

例如：

```
>> n=0:10;
>> stem(n,exp(-n),'filled');
```

结果如图 A-6。

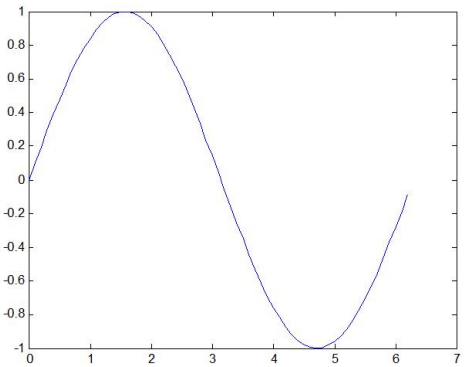

图 A-5 plot 函数绘图实例

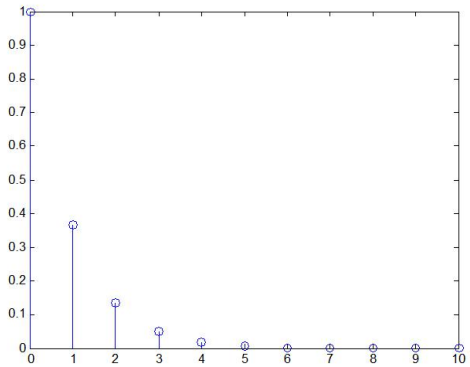

图 A-6 stem 函数绘图实例

3）ezplot 函数

ezplot 命令用于绘制二维符号函数的图形，主要有以下几种调用格式：

ezplot(f)　对于符号函数 f = f(x)，绘制 f = f(x)的图形，x 的默认范围为[-2*pi,2*pi]，对于符号函数 f = f(x,y)，绘制 f(x,y) = 0 的图形，x 和 y 的默认范围均为[-2*pi,2*pi]。

ezplot(f,[a,b])　绘制符号函数 f = f(x)的图形，x 的范围为[a,b]。

ezplot(f,[xmin,xmax,ymin,ymax])　绘制符号函数 f 的图形，x 的范围为[xmin,xmax]，y 的范围为[ymin,yman]。

我们来看一个简单的例子，

```
>> f = sym('cos(x)');
>> ezplot(f);
```

结果如图 A-7 所示。

4）fplot 函数

对于一元函数 y=f(x)，MATLAB 提供了一个专门的画图函数 fplot，可以方便地画出函数的图形，其常用的形式为：fplot(f,[xmin,xmax,ymin,ymax])，表示在[xmin,xmax,ymin,ymax]坐标轴范围内绘制由 f 表示的一元函数波形。例如：

```
>> fplot('sin(x)',[0,2*pi,-1.2,1.2]);
```

结果如图 A-8 所示。

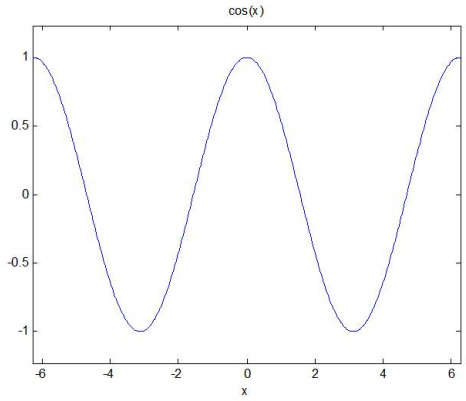

图 A-7 ezplot 函数绘图实例图

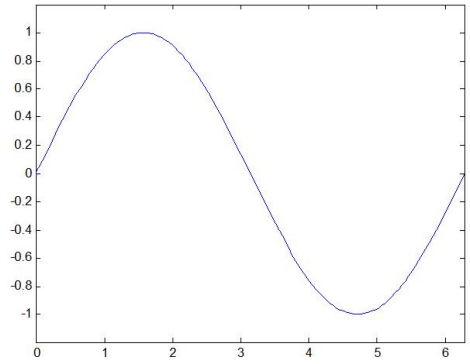

图 A-8 fplot 函数绘图实例

2. 三维图形绘制

1）plot3 函数

plot3 函数是三维绘图的基本函数，其使用格式有

```
plot3(x,y,z)
plot3(x,y,z,'s')
plot3(x1,y1,z1,'s',x2,y2,z2,'s',…)
```

当 x、y、z 为长度相同的向量时，plot3 函数将绘出一条以向量 x、y、z 为 x、y、z 轴坐标值的空间曲线。当 x、y、z 均为 m×n 的矩阵时，plot3 函数将绘出 n 条曲线，第 i 条空间曲线以分别以矩阵 x、y、z 的第 i 列为 x、y、z 轴坐标值。's'选项的含义与 plot 函数类似。

我们可以通过下面的例子来了解函数 plot3 的用法。

```
>> x=0:0.1:5;
>> y=sin(x);
>> z=x+y;
>> plot3(x,y,z);
```

结果如图 A-9 所示。

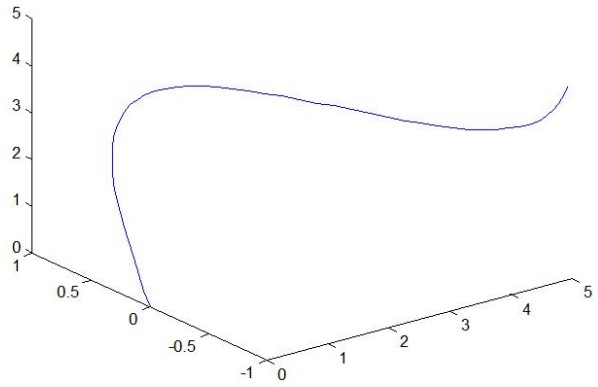

图 A-9 plot3 函数绘图实例

2）mesh 函数

mesh 函数与 plot3 函数的区别在于它绘出的不是单根曲线，而是以网格的形式绘出某一区间内的曲面。其使用格式如下

mesh(z)

mesh(x,y,z)

其中，x、y 必须均为向量，若 x、y 的长度分别为 m 和 n，则 z 必须是 m×n 的矩阵。若参数中不提供 x、y，则将（i,j）作为 z 矩阵元素的坐标值。

例如：

```
>> [x,y]=meshgrid(-12:0.5:12);
>> r=sqrt(x.^2+y.^2)+eps;
>> z=sin(r)./r;
>> mesh(z);
```

结果如图 A-10 所示。

3）surf 函数

三维表面函数 surf 与函数 mesh 的用法及其使用格式相同，不同之处在于绘得的图形是一个真正的曲面，而不是用网格来近似表达的。例如对于上一个例子中的 z，采样 surf 绘出图形如图 A-11 所示。

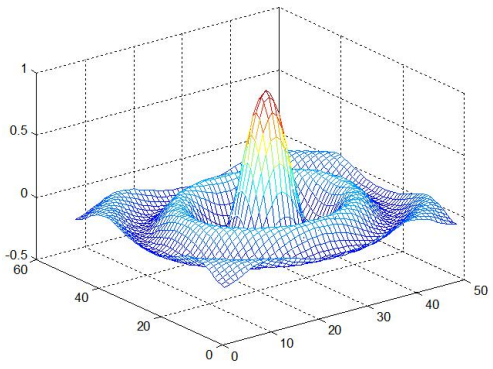

图 A-10 mesh 函数绘图实例

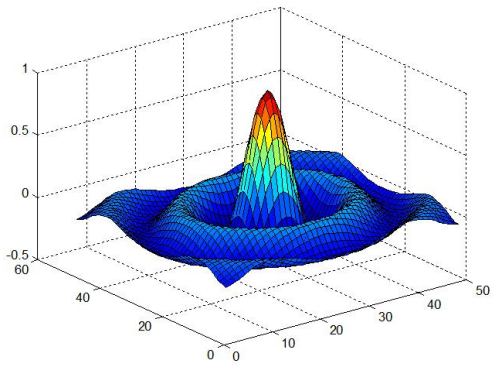

图 A-11 surf 函数绘图实例

A.6..2 图形控制函数

1. 图形标示函数

1）添加坐标轴标注

命令 xlabel、ylabel、zlabel 为 x、y、z 坐标轴添加标注。以函数 xlabel 为例，其调用格式为：

xlabel('string')

xlabel('string','property1',propertyvalue1,'property2',propertyvalue2,…)

其中'string'为要添加的标注文本，'property'指该文本的属性，propertyvalu 为相应的属性值。该命令把文本'string'按照指定的格式添加到 x 轴下方。可以设置的属性有很多，例如'FontWeigh' (字体粗细)、'FontName' (字体名称)、'FontSize' (字体大小)等。

2）添加图形标题

命令 title 给图形添加标题，调用格式和 xlabel 类似：

title('string')

title('string','property1',propertyvalue1, 'property2',propertyvalue2,…)

title 命令把文本'string'添加到图形上方。

3）添加文本

使用命令 text 可以在图形窗口的任何位置加入文本字符串。该函数调用格式为

text(x,y, 'string')

这里 x、y 用于指定加入字符串的位置，'string'是需要添加的字符串。

4）添加图例

在 MATLAB 中，为了便于对图形的观察和分析，用户可以使用 legend 命令为图形添加图例。legend 函数函数的一般调用格式为：

legend('stringl', 'string2', 'string3',…)

只要指定标注字符串，该函数就会按顺序把字符串添加到相应的曲线线型符号之后。

图例在缺省情况下自动置于图形的右上角，并自动把线型和标注字符串按顺序排列到一起。MATLAB 还允许很方便地对图例进行调整：用鼠标左键点住图例拖动即可移动图例到需要的位置；用鼠标左键双击图例中的某个字符串就可以对该字符串进行编辑。

在 legend 命令中还可以加入一个参数对图例的位置作出设置，它的使用格式如下

legend('string1', 'string2', 'string3',…,'Location',LOC)

表 A-10 列出了上式中 LOC 的取值及含义。

表 A-10 legend 命令 LOC 参数格式

字符串	说明	字符串	说明
'North'	坐标系上方	'SouthOutside'	坐标系底部外侧
'South'	坐标系底部	'EastOutside'	坐标系右方外侧
'East'	坐标系右方	'WestOutside'	坐标系左方外侧
'West'	坐标系左方	'NorthEastOutside'	坐标系右上角外侧
'NorthEast'	坐标系右上角	'NorthWestOutside'	坐标系左上角外侧
'NorthWest'	坐标系左上角	'SouthEastOutside'	坐标系右下角外侧
'SouthEast'	坐标系右下角	'SouthWestOutside'	坐标系左下角外侧
'SouthWest'	坐标系左下角	'Best'	坐标系中最佳位置
'NorthOutside'	坐标系上方外侧	'BestOutside'	坐标系外最佳位置

5）特征字符串

图形标示函数中，我们需要提供要标示的文本作为参数，然而有一些特殊字符，我们无法从键盘直接输入，例如π、∞、Ψ等。这些字符可以由特征字符串来表示。表 A-11 列出了一些常用的特征字符串。

表 A-11 常用特征字符串

特征字符串	字符	特征字符串	字符	特征字符串	字符
\alpha	α	\upsilon	υ	\rho	ρ
\beta	β	\phi	Φ	\sigma	σ
\gamma	γ	\chi	χ	\varsigma	ς
\delta	δ	\psi	ψ	\tau	τ
\epsilon	ε	\omega	ω	\equiv	≡
\zeta	ζ	\Gamma	Γ	\cap	∩
\eta	η	\Delta	Δ	\supset	⊃
\theta	θ	\Theta	Θ	\int	∫
\vartheta	ϑ	\Lambda	Λ	\forall	∀
\iota	ι	\Xi	Ξ	\exists	∃
\kappa	κ	\Pi	Π	\infty	∞
\lambda	λ	\Sigma	Σ	\approx	≈
\mu	μ	\Upsilon	ϒ	\subseteq	⊆
\nu	ν	\Phi	Φ	\in	∈
\xi	ξ	\Psi	Ψ	\oplus	⊕
\pi	π	\Omega	Ω	\cup	∪

下面给出了图形标注应用的一个例子：

```
%Eg_A_1.m
x=0:0.1:2*pi;
plot(x,sin(x),':',x,cos(x));
axis tight;
xlabel('x(0~2\pi)');
ylabel('y');
title('sin(x) and cos(x)','Fontsize',13,'FontWeight','Bold','FontName','宋体');
text(pi,0,'\leftarrowsin(\pi)=0') ;
text(0.1,-0.8,['Date：',date]);
text(0.1,-0.95,['MATLAB Version:',version]);
legend('Sin Wave','Cos Wave','Location','NorthEast');
```

运行结果如图 A-12 所示。

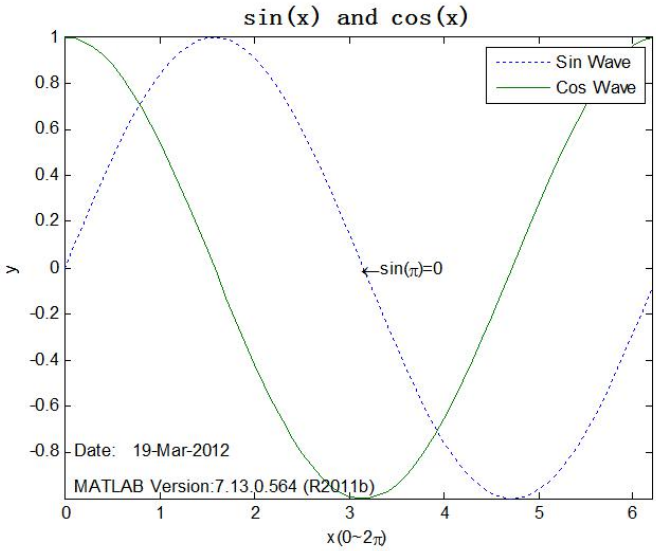

图 A-12 图形标示函数应用实例

2. 坐标轴控制函数

1）axis 函数

MATLAB 的绘图命令可以根据要绘制曲线数据的范围自动选择合适的坐标系，使得曲线尽可能清晰地显示出来，所以一般情况下用户不必自己选择绘图坐标。但对有些图形，如果用户觉得自动选择的坐标不合适，则可以用手动的方式调整坐标系。手动调整坐标系的工作可以由 axis 命令来完成，这里介绍该命令的部分功能：

axis([xmin xmax ymin ymax]) 设置各坐标轴的最大和最小值，其中 x 轴的范围为 xmin ~ xmax，y 轴的范围为 ymin ~ ymax。

axis tight 把坐标轴的范围定为所绘制数据的范围，即坐标轴中没有多余的部分。

axis off 关闭所用坐标轴上的标记、格栅和单位标记。

axis on 显示坐标轴上的标记、单位和格栅。

axis fill 将坐标轴取值范围分别设置为绘图所用数据在相应方向上的最大值和最小值。

axis auto 坐标轴范围采用缺省设置。

2）grid 函数

grid 函数用于打开或关闭坐标系中的网格线，具体使用格式如下：

grid on 对当前坐标加上网格线。

grid off 撤销网格线。

3）hold 函数

hold on 保持当前的图形所有的坐标特性以在已存在的图形上再添加其他图形。

hold off 删除前面的图形返回到缺省状态，在绘制新的图形以前重设所有坐标特性。

我们通过一个实例来说明上述函数的使用方法：

```
>> t=0:0.1:4*pi;
>> plot(t,sin(t));
>> hold on;
>> plot(t,cos(t),'--');
>> grid on;
>> axis([0 4*pi -1.2 1.2]);
```

运行结果如图 A-13，在这个例子里使用 hold 函数将两条曲线绘制在同一个坐标系中，

同时将 x 轴的范围设定为 0~4π，将 y 轴的范围设定为-1.2~1.2，并打开坐标系的网格线。

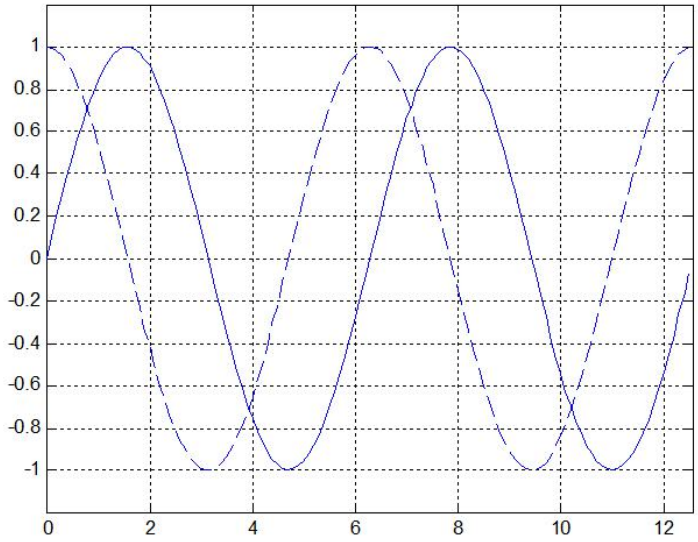

图 A-13 坐标轴控制函数应用实例

3. 窗口控制函数

1）图形窗口函数

figure 创建一个图形窗口并返回它的句柄。

figure(H) 使 H 成为当前窗口，位于最上端并处于可视状态。如果窗口 H 不存在，则将创建一个句柄为 H 的图形窗口。

2）图形窗口分割函数

函数 subplot 的调用格式为：subplot(m,n,i) 或 subplot(mni)，它的含意是把图形窗口分割为 m 行 n 列子窗口，然后选定第 i 个子窗口为当前窗口。例如命令 subplot(2,2,3)或 subplot(223)指的是把图形窗口分为 2 行 2 列共四个子窗口，选择第 3 个（即第 2 行第 1 列）子窗口为当前子窗口进行操作。这里通过一个实例来说明：

```
>> t=0:0.1:2*pi;
>> subplot(221);
>> plot(t,sin(t));
>> title('sin(t)');
>> subplot(222);
>> plot(t,cos(t));
>> title('cos(t)');
>> subplot(223);
>> plot(t,t+2);
>> title('t+2');
>> subplot(224);
>> plot(t,2*t);
>> title('2*t');
```

运行结果如图 A-14 所示。

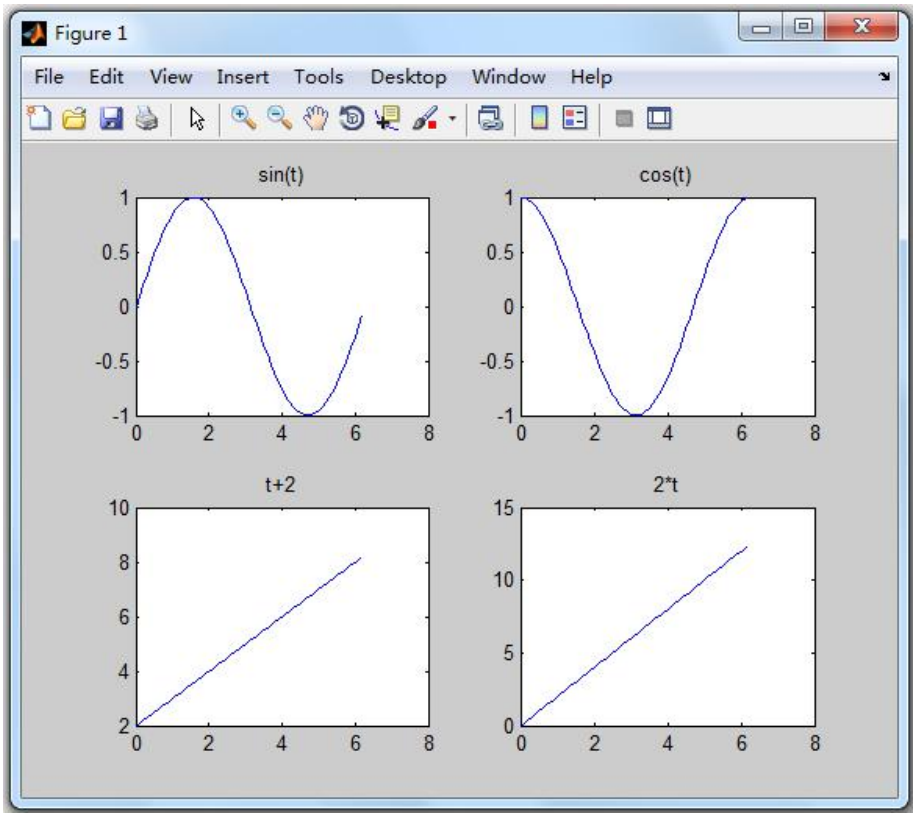

图 A-14 subplot 函数应用实例

A.7 MATLAB 程序设计基础

MATLAB 除了在命令窗口逐句输入命令执行外，作为一种高级应用软件，MATLAB 还提供了自己的编程语言和编程环境。

A.7.1 MATLAB 文件

MATLAB 的命令窗口为用户使用 MATLAB 解决问题带来了很大的方便，对于很多简单的问题，用户可以通过在命令窗口输入一组命令来解决。然而当要解决的问题所需的命令较多且命令结构复杂，或需要经常修改参数进行算法调试的时候，直接在命令窗口输入命令的方法就显得繁琐和笨拙了。这时候我们通常把解决问题所需的命令写成一个由多行语句组成的文本文件中，让 MATLAB 来执行这个文件，这个文本文件以扩展名“.m”结尾，我们称之为 MATLAB 文件。MATLAB 文件分为两种，一种称为脚本文件，另一种称为函数文件。

1. 文件编辑器（Editor）

在 MATLAB 环境中 MATLAB 文件的编写需要通过 MATLAB 文件编辑器来进行。默认情况下 MATLAB 文件编辑器不会随 MATLAB 的启动而开启，需要编写 MATLAB 文件时，可以通过【File】菜单→【New】→【M-File】启动 MATLAB 文件编辑器，用户界面如图 A-15 所示。MATLAB 文件编辑器中不仅可以编辑 MATLAB 文件，还可以对 MATLAB 文件进行调试，MATLAB 文件编辑器自动用不同颜色区别程序的不同内容，容易检查发现错误，使用非常方便。

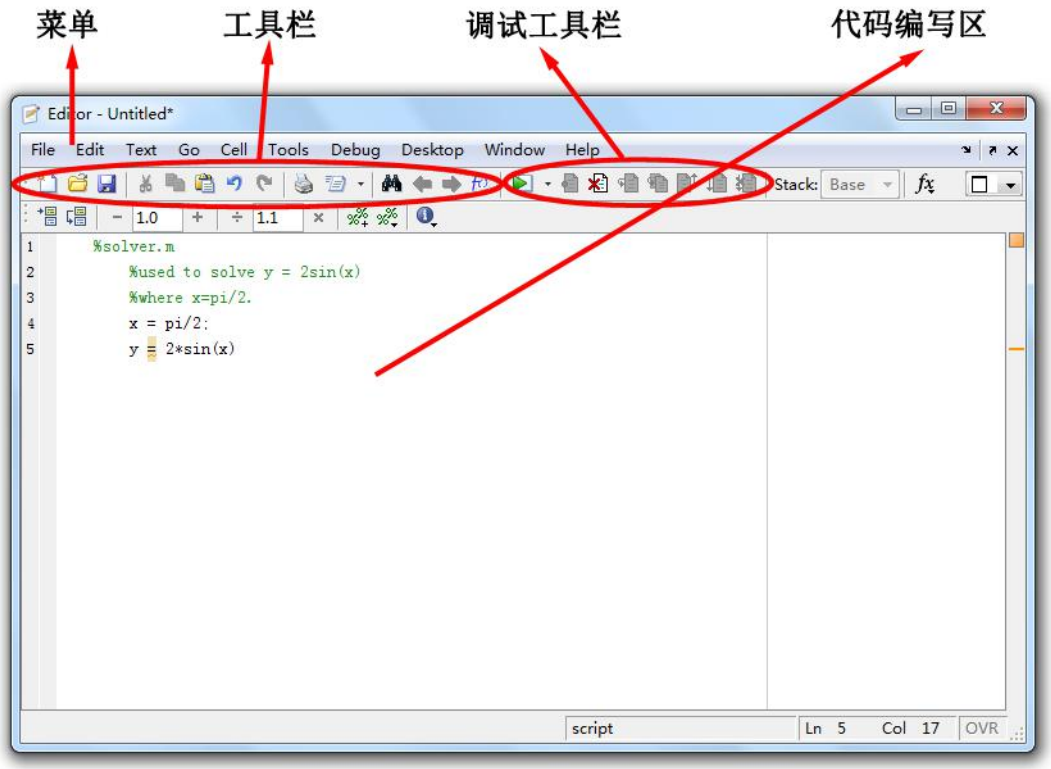

图 A-15 MATLAB 文件编辑器

2. 脚本文件

脚本文件是按照 MATLAB 语言语法写成的 MATLAB 命令的集合，将解决问题所需的一系列 MATLAB 命令语句按顺序写在一个 MATLAB 文件中，脚本文件不接受输入和输出参数，是最简单的 MATLAB 文件。脚本文件与 MATLAB 工作空间共享变量空间，脚本文件运行结束后，产生的所有变量都会保存在 MATLAB 工作空间中，可以对这些变量继续进行操作和运算。

下面是一个求解已知表达式的脚本文件。

```
%solver.m
%used to solve y = 2sin(x)
%where x=pi/2.
x = pi/2;
y = 2*sin(x)
```

每一行中%后面部分为注释内容。我们在 MATLAB 文件编辑器中输入上述语句，然后将其保存为文件名为 solver.m 的文件，这样就成功创建了一个脚本文件。脚本文件有两种执行方式：

- 在命令窗口中输入脚本文件文件名，执行该文件。
- 通过 MATLAB 文件编辑器上方【Debug】菜单→【Save File and Run】或工具栏中的运行按钮执行。

例如，我们在命令窗口中只要输入文件名 solver 并回车，即可执行该文件：

```
>> solver
y =
     2
```

3. 函数文件

另一种 MATLAB 文件我们称之为函数文件，用于扩充 MATLAB 函数库时使用，可以接受参数输入和输出。函数文件具有独立的内部变量空间，函数文件运算中产生的变量都保存在函数本身的临时变量空间中，不会与 MATLAB 工作空间中的变量相互覆盖。函数文件一般处理输入参数传递的数据，并把处理结果作为函数输出参数返回给 MATLAB 工作空间中的指定接收变量。

函数文件在文件开头以关键字 function 声明，格式为：

function [返回值 1,返回值 2,…] = 函数名(参数 1,参数 2,…)

…函数体语句…

例如，编写一个 MATLAB 函数用于求解 $x^2 + y^2$，在 MATLAB 文件编辑器中编写如下代码：

```
function z = my_fun(x,y)
% z = my_fun(x,y)
% Used to solve z = x^2 + y^2
z = x^2 + y^2;      %solve, and return the result
```

保存函数文件时，文件名与函数名相同，例如上述代码应保存为文件“my_fun.m”。在命令窗口中输入命令调用该函数：

```
>> a = 2;
>> b = 3;
>> c = my_fun(a,b)
c =
      13
```

在函数文件中到第一个非注释行为为止的注释行是帮助文本，使用 help 命令查询 my_fun 函数的帮助信息时返回这些文本。需要帮助时返回该文本。例如使用 help 命令查询 my_fun 函数的帮助信息：

```
>> help my_fun
z = my_fun(x,y)
Used to solve z = x^2 + y^2
```

需要注意的是，MATLAB 文件命名时文件名必须以英文字母开头，后面可以接字母、数字、下划线，中间不能有空格，且应避免与系统函数名和系统保留字相同。

4. MATLAB 函数的类型

MATLAB 的函数可以分为主函数和子函数、嵌套函数、私有函数、匿名函数等。

1）主函数和子函数

MATLAB 允许一个函数文件包含多个函数，其中第一个出现的函数称为主函数，其他函数称为子函数，保存时文件名与主函数名相同。主函数可以在该文件之外调用(如在命令窗口或其他 MATLAB 文件中)，子函数只能在主函数和该 MATLAB 文件的其他子函数中调用。例如在 MATLAB 文件编辑器中编写如下代码：

```
function [avg,med] = my_stats(x,y)
%avg 返回 x 和 y 的平均值，med 返回 x 和 y 较大的值
avg = average(x,y);
med = median(x,y);


function z = average(x,y);
```

```
%求平均值
z = (x + y)/2;

function z = median(x,y)
%求 x 和 y 较大的值
if x > y
    z = x;
else
    z = y;
end
```

将以上代码保存为文件“my_stats.m”，在命令窗口中输入如下命令调用该函数：

```
>> [a,m] = my_stats(2,4)
a =
     3
m =
     4
```

2）嵌套函数

在一个 MATLAB 函数内部可以定义一个或多个函数，这种在其他函数内部定义的函数成为嵌套函数。嵌套可以多层发生，即一个函数内部可以嵌套多个函数，而这些嵌套函数内部可以继续嵌套函数。

嵌套函数的格式如下：

```
function x = fun1(p1,p2,…)
……
……
    function y = fun2(r1,r2,…)
    ……
    end
……
……
end
```

使用嵌套函数时，父函数和嵌套函数都需要以 end 表示函数结束。我们来看嵌套函数的一个实例：

```
function x = my_nest(p1,p2)
p3 = my_nest1(p1);
    function y = my_nest1(r)
    y = 3 * r;
    end
x = p2 + p3;
end
```

保存该函数文件为“my_nest.m”，在命令窗口调用该函数：

```
>> x = my_nest(1,2)
x =
     5
```

在多层嵌套函数中，嵌套函数的互相调用与嵌套的层次密切相关，例如：

```
function A(x,y)       %外层函数 A
B(x,y);
D(x)
    function B(x,y)       %A 的嵌套函数(第一层嵌套)
    C(x);
    D(y);
        function C(x)       %B 的嵌套函数(第二层嵌套)
        D(x);
        end
    end
    function D(x)       %A 的嵌套函数(第一层嵌套)
    E(x);
        function E(x)       %D 的嵌套函数(第二层嵌套)
        …
        end
    end
end
```

一个函数可以调用自己的下一层嵌套函数，但不能调用更深层次的嵌套函数(A 可以调用 B 和 D，但不可以调用 C 和 E)；嵌套函数可以调用同一个父函数下与自己相同层次的其他嵌套函数(B 和 D 可以互相调用)；嵌套函数可以调用与其父函数相同层次的其他嵌套函数，但不能调用与其父函数相同层次的嵌套函数的更深层次嵌套函数(C 可以调用 D，但不可以调用 E)。

3）私有函数

私有函数是具有限制性访问权限的函数，这些函数在编写上与普通函数一样，但是需要保存在以“private”命名的目录下。私有函数只能被 private 目录的父目录下的 MATLAB 文件访问，在其父目录之外没有访问权限。

4）匿名函数

匿名函数是 MATLAB 函数的一种简单形式，通常只由一句很简单的声明语句组成，不要求有 MATLAB 文件。可以在 MATLAB 命令窗口或 MATLAB 文件中定义和调用它。创建匿名函数的语法如下：

```
fhandle = @(参数列表)表达式
```

“@”符号是 MATLAB 中创建函数句柄的操作符，由参数列表和表达式确定的函数创建句柄，并返回给 fhandle，之后可以通过 fhandle 来调用函数。

我们以求解函数为例来说明匿名函数的使用：

```
>> my_solver = @(x,y)x^2 + y^2;
>> z = my_solver(1,2)
z =
     5
```

A.7.2 MATLAB 程序流程控制

同其他计算机编程语言一样，MATLAB 也提供了多种经典的程序结构控制语句，如循环结构（for 循环、while 循环）和条件分支结构（if else 条件分支结构、switch case 条件分支结构），下面分别进行介绍。

1. for 循环结构

for 循环语句的一般格式为：

```
for var=array
    statements
end
```

这里的：

var：　　循环控制变量。

array：　　循环控制向量。

statements：　要重复执行的程序代码。

end：　　循环结束的关键字（要与 for 成对使用）。

例如下面的 for 循环将执行 6 次：

```
%Eg_A_2.m
for i = 1:6
    x(i) = i^2;
end
disp(x);
```

执行上述 MATLAB 文件，结果如下：

```
>> Eg_A_2
     1     4     9    16    25    36
```

for 循环可以根据需要嵌套多次。例如下面的程序为矩阵 A 赋值：

```
%Eg_A_3.m
for i = 1:5
    for j = 1:5
        A(i,j) = i + j;
    end
end
disp(A)
```

执行上述 MATLAB 文件，结果如下：

```
>> Eg_A_3
     2     3     4     5     6
     3     4     5     6     7
     4     5     6     7     8
     5     6     7     8     9
     6     7     8     9    10
```

2. while 循环结构

while 循环语句的一般格式为:

```
while expression
    statements
end
```

当表达式为真时，执行 while 和 end 语句之间的程序代码。与 for 循环不同的是，while 循环不能指定循环的次数，每次执行语句后再判断表达式的值是否为真，如果是继续执行 while 和 end 语句之间的程序代码，如果不是则跳出循环体。

例如，使用 while 循环实现 $n=5$ 时 $s=1+2+4+\cdots 2^n$ 的求和计算代码如下

```
%Eg_A_4.m
s=0;
n=0;
while n<=5
    s=s+2^n;
    n=n+1;
end
disp(['s=' num2str(s)]);
```

执行上述 MATLAB 文件，结果如下：

```
>> Eg_A_4
s=63
```

3. if-else 条件分支结构

if 语句的基本格式为：

```
if logical_expression
    statements
end
```

如果 if 语句所跟逻辑表达式为真，执行 if 和 end 之间的程序代码，如果该逻辑表达式为假，跳过 if 和 end 之间的所有程序代码，继续执行下面的代码。

if 语句常见的格式还有:

```
if expression
    statements_1
else
    statements_2
end
```

或者

```
if expression
    statements_1
elseif expression
    statements_2
elseif
    …
else
    statements_n
end
```

上述语句首先判断 if 语句后的表达式，当为真时，执行 statements_1,否则跳到 else 语句或 elseif 语句，判断其后的表达式，若为真，执行相应的命令，若为假，继续条转到下一条，直到 if 语句结束。

例如，生成一个长度为 10 的序列，前 5 各元素为 1，后 5 各元素为-1，可以使用 if-else 语句实现如下：

```
%Eg_A_5.m
for i=1:10
```

```
    if i<=5
        x(i)=1;
    else
        x(i)=-1;
    end
end
disp(x);
```

执行上述 MATLAB 文件，结果如下：

```
>> Eg_A_5
     1     1     1     1     1    -1    -1    -1    -1    -1
```

4. switch case 条件分支结构

switch-case 语句的一般格式为:

```
switch  expression
    case value1
        statements_1
    case value2
        statements_2
    …
    otherwise
        statements_n
end
```

当 expression 与 case 语句后的任何一个分支相匹配，则相应执行该 case 语句后的程序代码，如果与所有的分支都不匹配，则执行 otherwise 后的程序代码。

例如下面的代码可以实现根据 n 的不同取值，生成不同的矩阵：

```
switch n
    case 1
        A=ones(3,3);
    case 2
        A=zeros(3,3);
    otherwise
        A=rand(3,3);
end
```

上述程序表示：当 n=1 时，生成一个 3×3 的“0”矩阵；当 n=2 时，生成一个 3×3 的“1”矩阵；当 n 为其他值时，生成一个 3×3 的随机矩阵。

5. 其他程序流程控制指令

1）pause 命令

用于暂停程序的运行，调用格式如下：

pause 暂停程序运行，按任意键后继续

pause(n) 暂停程序运行，等待 n 秒后继续

2）break 命令

用于实现 for 循环或 while 循环的终止，当执行到 break 语句时跳出循环结构，不必等

到循环自然结束。

3）continue 命令

用于 for 循环或 while 循环结构中，表示忽略此次循环的其他语句，直接进入下一各循环。

4）input 命令

用于程序执行过程中通过键盘交互式输入数据，调用格式如下：

a = input(str)　将通过键盘输入的内容赋值给变量 a，str 表示显示的提示信息

a = input(str,'s')　将通过键盘输入的内容作为字符串赋值给变量 a

例如，执行如下命令

```
>> a = input('请输入参数 a 的值：')
```

命令行窗口中会显示提示信息：

请输入参数 a 的值：

此时输入内容，例如输入数字“10”，则运行结果为

```
a =
     10
```

5）disp 命令

用于输出数据到命令窗口，调用格式为

disp(x)　表示将变量 x 的内容显示到命令行窗口中。

A.7.3 MATLAB 程序调试

MATLAB 提供了程序调试功能，可以对 MATLAB 文件进行调试。MATLAB 程序出错主要为以下两类：

- 语法错误，通常为函数名拼写错误，括号遗漏等。
- 算法错误，逻辑上出错，而语法上是正确的，这类错误不易查找。

MATLAB 程序出错时，命令窗口中会输出错误提示信息，指出错误发生在哪一行。对于语法错误，可以根据错误提示信息很容易的发现并更正错误，而算法错误则需要依靠一些其他的调试方法或工具实现。

MATLAB 程序调试有直接调试和工具调试两种方法。

直接调试法指在 MATLAB 文件中添加一些输出，以便分析和发现可能的错误。例如，将某些语句结尾的分号去掉或在适当位置添加输出某些关键变量值的语句，使程序输出一些中间计算结果。

工具调试指利用 MATLAB 的调试器（Debugger）进行调试，主要通过 MATLAB 文件编辑器【Debug】菜单下的子项进行。

【Debug】菜单下的常用子项及功能如下：

- 【Open M-File when Debugging】　用于调试时打开 MATLAB 文件
- 【Step】　用于单步调试程序
- 【Step In】　用于单步调试进入子程序
- 【Step Out】　用于单步调试从子程序跳出
- 【Save File and Run】　保存文件并运行
- 【Go Until Cursor】　运行到当前 MATLAB 文件光标所在行
- 【Set/Clear Breakpoint】　设置或清除断点
- 【Set/Modify Conditional Breakpoint】　设置或修改条件断点
- 【Enable/Disabled Breakpoint】　设置断点为有效或无效
- 【Clear Breakpoints in All Files】　清除所有 MATLAB 文件中的断点

◆ 【Exit Debug Model】 退出调试模式

上述子项有对应的快捷工具按钮，位于 MATLAB 文件编辑器的上方的工具栏中，如图 A-16 所示。

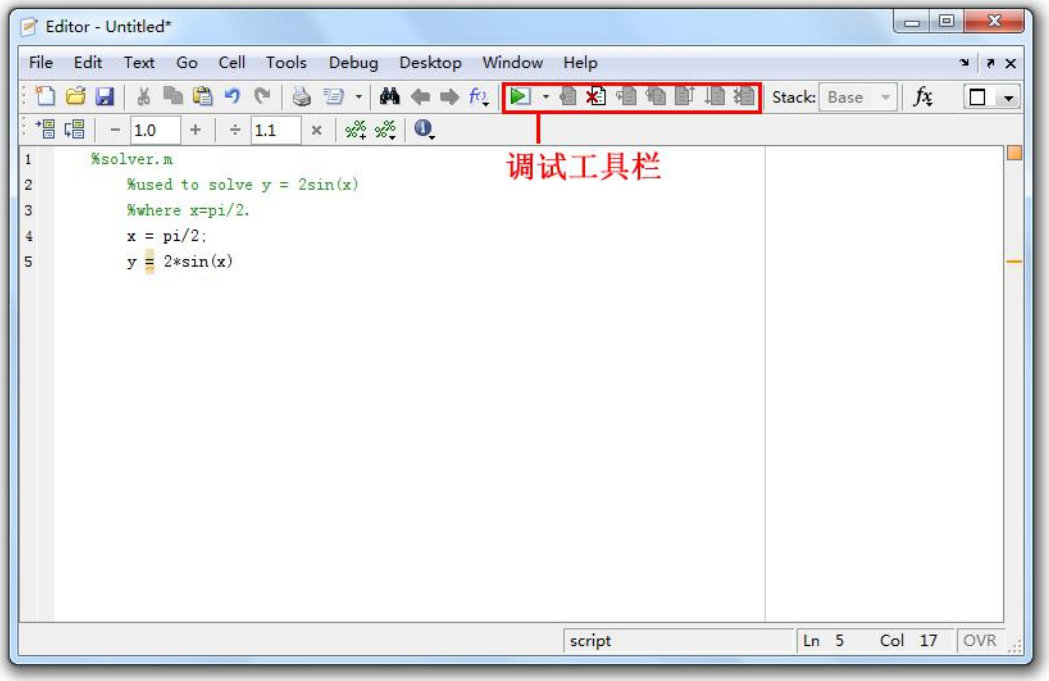

图 A-16 调试工具栏