

信号与系统实验指导书

哈尔滨工业大学 (深圳)

实验与创新实践教育中心

2020 年 5 月

实验一 MATLAB 基础

1.1 实验目的

- (1) 对 MATLAB 软件有一个基本的认识;
- (2) 理解矩阵（数组）概念及其各种运算和操作;
- (3) 掌握绘图函数;
- (4) 学会 M 文件的基本操作。

1.2 实验预习要求

- (1) 学会安装 MATLAB 软件;
- (2) 条件允许的情况下, 请在自己的电脑上安装 MATLAB 软件。

1.3 实验仪器

表 1-1 实验仪器与器件列表

名称	数量	型号（推荐）
电脑	1	CPU i5 以上
MATLAB 软件	1	2012 以上版本

1.4 实验原理

MATLAB 是由 MathWorks 公司开发的一套功能强大的数学软件, 也是当今科技界应用最广泛的计算机语言之一。它将数值计算、矩阵运算、图形图像处理、信号处理和仿真等诸多强大的功能集成在较易使用的交互式计算机环境之中, 为科学研究、工程应用提供了一种功能强、效率高的编程工具, 是其他许多语言所不能比拟的。

MATLAB 名字由 Matrix（矩阵）和 Laboratory（实验室）两个单词组合而成, 由名字就可以看出, 其最初设计和使用是针对于数学计算进行的, 后来随着软件不断发展、升级, 逐渐发展成一款在矩阵计算和数值分析方面首屈一指的商业数学软件。它可以进行矩阵运算、图形绘制、创建用户界面、仿真实验、连接其他编程语言等, 被广泛的应用于信号和图像处理、通信、控制系统设计、测试和测量、财务建模和分析以及计算生物学等众多不同的学科领域。特别是 MATLAB 所附带的几十种面向不同领域的工具箱（单独提供的专用 MATLAB 函数集）, 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。

在附录中, 介绍了 MATLAB 的基础知识以及程序设计的基本方法。

1.4.1 创建数据

我们知道在 MATLAB 中, 变量是以矩阵形式存在的。

- (1) 直接输入

直接输入数据，回车后即返回结果。在语句后面加上一个分号，再按回车键，会发现返回值被隐藏起来了。

注意：MATLAB 中引用一个变量无需先声明。

(2) 通过函数

1、有时我们需要定义一个连续变量的时候，比如时间变量向量，如果我们采用刚才输入的方法会有些麻烦，那么我们可以采用冒号的方法产生向量。



输入：`t=1: 10`。然后，再试一下输入 `tt=-5: 0.5: -1`。

从返回值中我们可以总结一些规律：一个冒号时，冒号前面是向量的起始值，冒号后面是结束值，默认步长是 1。两个冒号时，第 1 个冒号前面是向量的起始值，第二个冒号后面是结束值，两个冒号之间是步长。

如果我们想要创建 $n \times m$ 的矩阵，利用分号（`:`）来进行分行，例如，输入 `a=[1:5;6:10]`。

2、我们还可以通过调用函数的方法产生矩阵。

例如，分别输入 `zeros (2, 3)`、`ones (3, 2)`、`rand (3, 3)`、`linspace (1,5,10)` 观察输出结果，产生的分别是全 0、全 1、随机矩阵。前面三个函数括号里面逗号前后的数分别表示行数和列数，`linspace` 函数的意义可以通过软件自带的帮助功能来查看定义。

查看函数的定义和使用方法，有三种方法，一是在命令行窗口右键选中命令查看“关于所选内容的帮助”；二是搜索  里，输入函数名称直接搜索；三是打开  帮助对话框，在帮助对话框搜索查找函数。

注意：需要注意的是，如果同一个变量重复赋值，工作空间将保存最后一次的赋值。

(3) 导入数据文件

MATLAB 还可以自动导入其他文件中的数据，常用到文件类型有：`.xlsx`，`.txt`，`.csv` 等。

重要的数据可以通过保存工作区来进行保存，如果要清空所有工作区，可以使用 `clear` 命令。

注意：清空工作区后无法恢复，建议慎重使用。

1.4.2 基本运算函数

(1) 提取矩阵中的某些元素

首先我们随机定义一个 5×5 的矩阵 A。

输入依次输入 `A (6)`、`A (15)`、`A (19)`，查看返回值。

可以得到结论，矩阵在 MATLAB 中是以 1 维数组的形式存储的。首先是第 1 列，接着是第 2 列。接着是第 3 列……直到最后一列。

如果我们想要提取其中第二行，输入 `A(2,:)`，想要提取其中第二列，输入 `A(:,2)`。逗号前后分别是行、列序号。冒号前后分别表示从第几个到第几个，如果没有值表示所有值。

(2) 矩阵的基本运算

矩阵的基本运算包括：加、减、乘、除、幂运算、转置。这些运算符号和函数在附录中已给出，这里不再赘述。另外，可以通过 `clc` 命令，清空命令行窗口。

注意：如果我们在乘方运算符前面加上一个点则表示矩阵中的每个元素做平方运算。其他运算符也是一样的，在其前面加上一个点表示对矩阵元素的操作。

1.4.3 绘图函数

`plot` 是最基本的绘图函数，也是最常用的。

输入 `plot(x)` 时，默认绘图的横坐标为序号，纵坐标为 `x` 数组的值。输入 `plot(x, y)` 时，绘图以 `x` 的值为横坐标，`y` 的值为纵坐标。

绘图函数的用法在附录中，有详细介绍，这里不再赘述。

当想在同一幅 `figure` 中绘制多条曲线时，可以输入 `hold on` 命令，绘制完成后，输入 `hold off` 命令。绘图时，还可以通过改变绘图线条颜色或形状，添加图形标注与网格等，这些内容都在附录中有说明。

1.4.4 M 文件基本操作

在实际应用中，直接在 MATLAB 工作空间的命令窗口中输入简单的命令并不能够满足用户的所有需求，因此 MATLAB 提供了另一种强大的工作方式，即利用 M 文件编写工作方式。

实际运用中，我们可以通过 MATLAB 新建、编辑和保存 `.m` 文件，也可以将任意记事本文件重命名为 `.m` 文件，然后通过 MATLAB 软件打开该文档进行编辑和保存。

M 文件的基本操作和调试程序基本内容可以查看附录。

1.5 实验内容

要求：实验过分别写到每个实验内容下，需要时给出截图（不是整个屏幕的截图），最后给出实现这些内容的程序文件（所有内容放到一个 M 文件中，内容（4）不需要提交 M 文件）。

- (1) 创建一个任意 10×20 （10 行，20 列）的随机数组 `A`。`A1` 为数组 `A` 的子数组（sub-array），范围为第 3 行第 5 列到第 8 行第 12 列，求 `A1` 每一列的和、均值、方差。
- (2) 创建 `B`、`C` 为任意实数组成的 3×3 数组，分别进行以下计算：
 - $k \times B$, 其中 `k` 为任意自定义的实数
 - `B` 矩阵的 3 次方
 - `B` 的每个元素除以 `C` 对应的那个元素
 - `B+IjxC`
 - `B+IjxC` 的转置
- (3) 绘制一条 $\sin(x)$ 曲线，`x` 的范围在 0 到 4π 之间，要求绘图有名字、横纵坐标有图例，绘图有网格。
- (4) 将给定的 `.m` 文件（文件名为：Exp1），调试运行成功，并给出绘图结果。
- (5) 附加题：

计算第（1）题中数组 A 的第 4 行第 4 列到第 8 行第 8 列的子数组 A2 和第 5 行第 5 列到第 9 行第 9 列的子数组 A3 对应的每一列的相关系数（自己查公式），要求：当 A3 中有元素值小于 0.5 时，该值不参与计算；可以使用 if 和 for 语句来操作（语法自己查，与 C 语言类似），给出源代码和执行结果的截图。

1.6 实验思考题

- 1、linspace（1,5,10）表示的意义是什么？
- 2、怎么对矩阵的行进行计算（比如分别计算每一行的和、均值、方差）？
- 3、如何将数组进行排序（从大到小、从小到大）？

1.7 注意事项

- (1) 所有任务都必须独立完成，或查书，或百度谷歌，但是不允许复制粘贴（请自律），对每个任务都尽可能地详细回答，不能敷衍！！
- (2) 使用计算机和上网请遵守国家法律法规。

1.8 实验报告要求

- (1) 独立完成实验内容，诚实记录实验结果；
- (2) 实验思考题要写在实验报告中。
- (3) 实验体会、意见和建议写在实验结论之后。
- (4) 实验报告须包括：
 - 1、电子版的实验报告；
 - 2、程序源文件：*.m以上内容请按照以下顺序放到一个文件夹内，并将文件夹命名为：学号-姓名-实验*，如：180110888-张三-实验一。

附录 1 MATLAB 的基础知识

在 MATLAB 内部，任何数据类型，都是按照矩阵（数组）的形式进行存储和运算的。这里说的矩阵是广义的，它可以只有一个元素，也可以是一行或一列元素，还可能是最普通的二维数组，抑或高维空间的多维数组；其元素可以是任意数据类型，如数值型、逻辑型、字符串型或单元型等。

理解矩阵（数组）概念及其各种运算和操作，是学习 MATLAB 时的一个重要问题。

一、常用数据类型

数据类型是掌握任何一门编程语言都必须首先了解的内容。MATLAB 的数据类型主要有：逻辑、数值、字符串、矩阵、元胞、Java、函数句柄、稀疏以及结构等类型，其中数值型又有单精度型、双精度型以及整数型。向整数类型里面也分无符号型（uint8、uint16、uint32、uint64）和符号类型（int8、int16、int32、int64）两种。但是，需要牢记，在 MATLAB 中，所有数据不管是属于什么类型，都是以数组或矩阵的形式保存的。

下面介绍几种常用的数据类型。

1. 数值型

数值类型包括整数型（带符号和无符号）和浮点数（单精度和双精度）两种。在默认状态下，MATLAB 将所有的数都看做是双精度的浮点数；双精度浮点数以 64 为存储。所有的数值类型都支持基本的数组运算，除 int64 和 uint64 外所有的数值类型都可以应用于数学运算。

1) 整型

如图 1，在 MATLAB 中有 4 种符号类型和 4 种无符号整型。符号类型可以表示正数、负数和零，但是它表示的数值范围比无符号类型要小；无符号类型只能表示非负数。

```
>> x=325.499
x =
    325.4990
>> x=x+.001
x =
    325.5000
>> int16(x) %对x取整
ans =
     326
>> intmax('int8') %最大的整型数值
ans =
     127
>> intmin('int8') %最小的整型数值
ans =
    -128
```

图 1 整型

2) 浮点型

MATLAB 中用双精度或单精度来表示浮点类型的数据，默认为双精度，但用户可以用一个简单的转换函数把任何数据用单精度来表示。如图 2。

```

>> clear
>> x=23.456
x =
    23.4560
>> y=single(x) %用函数single创建单精度数据
y =
    23.4560
>> whos
  Name      Size      Bytes  Class  Attributes

  x         1x1         8  double
  y         1x1         4  single

```

图 2 浮点型

3) 复数

复数由两个独立部分组成：实部和虚部。虚部的基有单位为 $\sqrt{-1}$ ，在 MATLAB 中常常用字母 i 或 j 来表示，如图 3。

```

>> clear
>> x = rand(3) * 5 %复数的创建
x =
    1.9611    3.5302    0.2309
    3.2774    0.1592    0.4857
    0.8559    1.3846    4.1173
>> y = rand(3) * -8 %rand(a)函数表示随机产生a个0~1之间的实数
y =
   -5.5586   -0.2756   -6.1241
   -2.5368   -3.5100   -6.3616
   -7.6018   -3.0525   -1.4950
>> z = complex(x,y) % complex函数:c=complex(a,b), 则 c = a + bi
z =
    1.9611 - 5.5586i    3.5302 - 0.2756i    0.2309 - 6.1241i
    3.2774 - 2.5368i    0.1592 - 3.5100i    0.4857 - 6.3616i
    0.8559 - 7.6018i    1.3846 - 3.0525i    4.1173 - 1.4950i

```

图 3 复数

4) 无穷大和 NaN

在 MATLAB 中，用 inf, -inf 和 NaN（Not a Number）分别表示正无穷大、负无穷大和不确定值。

5) 数据类型的显示

最常见显示数据函数为 whos，它可以显示变量的类型、大小、占用空间以及属性值，在 MATLAB 中还提供了其他以下函数来检验数据的类型，如表 1 所示。

表 1 常见数据类型识别函数

命 令	操 作
whos x	显示 x 的数据类型属性
type_x = class(x)	获取 x 的数据类型并赋值给 type_x
isnumeric(x)	确定 x 是否为数值类型
isa(x,'integer'),isa(x,'uint64')	确定 x 是否为特别的数值类型(如任

isa(x,'float'), isa(x,'double') isa(x,'single')	一整型、无符号 64-bit 整型、任意浮点型、双精度型、单精度型等)
isreal(x)	确定 x 是否为实数或复数类型
isnan(x)	确定 x 是否为非数
isinf(x)	确定 x 是否为无穷数
isfinite(x)	确定 x 是否为有限数

2. 字符型

在 MATLAB 中，字符串指的是一个统一编码的字符排列。字符串用一个向量或字符来表示，字符串存储为字符数组，如图 4，每个元素占用一个 ASCII 字符，对与存储长度不一的字符串和包含多个串的数组最好使用元胞类型数组。

```
>> name = 'Xiao Ming' %创建1行9列的字符数组
name =
Xiao Ming
>> name2 = ['Xiao Ming'; 'Xiao Hong'] %创建二维字符数组
name2 =
Xiao Ming
Xiao Hong
>> whos
  Name      Size      Bytes Class  Attributes
  |-----|
  name      1x9          18 char
  name2     2x9          36 char
```

图 4 字符型

3. 逻辑类型

逻辑数组类型是用数字 0 和 1 分别表示逻辑假和逻辑真，逻辑类型的数据不一定是标量，MATLAB 也一样支持逻辑型数组，而且逻辑性的二维数组可能是稀疏的，如图 5。

```
>> x = magic(4) >= 9 %创建逻辑型数组
x =
     1     0     0     1
     0     1     1     0
     1     0     0     1
     0     1     1     0
```

图 5 逻辑类型

除以上几种数据类型外，常用的数据类型还包括元胞类型、结构类型、Java 类型、函数句柄类型等，这里就不逐一介绍了。

二、矩阵及其应用

MATLAB 语言是由最初专门用于矩阵运算的计算机语言发展而来的。MATLAB 语言最重要、最基本的功能就是进行实数或复数矩阵的运算，其所有的数值计算功能都是以矩阵(或数组)为基本单元来实现的，尤其是在 MATLAB 图形图像处理、信号处理和控制理论等方面涉及大量的矩阵运算。矩阵运算和数组运算在形式上有很多相似之处，但是在 MATLAB 中二者的运算性质是不同的，数组运算强调的是元素对元素的运算，而矩阵运算则采用线性代数的运算方式。读者不能把二者混为一谈，以免产生一些不可预期的错误。

1. 矩阵的创建

矩阵的创建方法和数组的创建方法类似，也可以通过直接输入、增量法、利用函数 linspace

或 `logspace` 等几种方式，当创建矩阵的数据比较多时，可以通过 MATLAB 提供的矩阵编辑器（Matrix Editor）来生成或者修改矩阵。

一般创建矩阵的方法同创建变量的方法是一样的，前面已经介绍过，这里介绍从 `.xls(.xlsx)` 文件导入矩阵的方法。在菜单栏点击 **Import Data**，在弹出的对话框中选择需要导入的数据文件，然后出现如图 6 的对话框。在 **Import** 对话框中，可以通过鼠标选择 **Sheet** 中的数据，也可以通过编辑菜单中的 **Range** 来规定需要导入的数据的范围，同时，可以选择以什么样的方式导入数据，并对没有选择的数进行怎么的操作（图中选择的操作是以不确定值代替）。最后，点击 **Import Selection** 按钮，则数据导入成功。导入成功后，可以通过编辑变量的方式，对矩阵进行修改名称、修改数据等操作，如图 7。

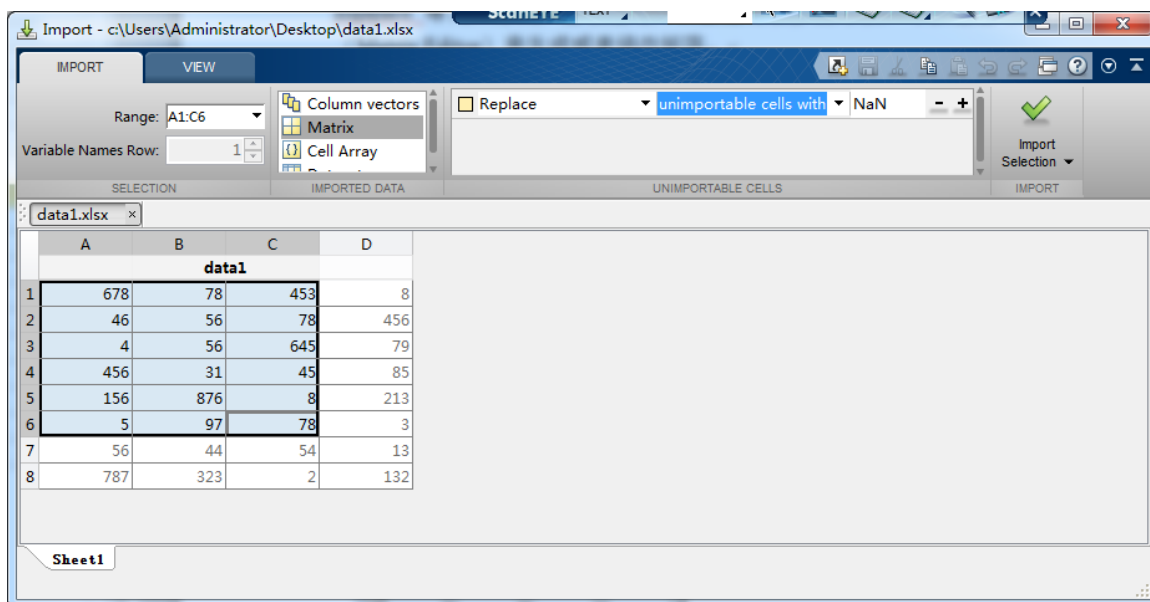


图 6 导入数据对话框

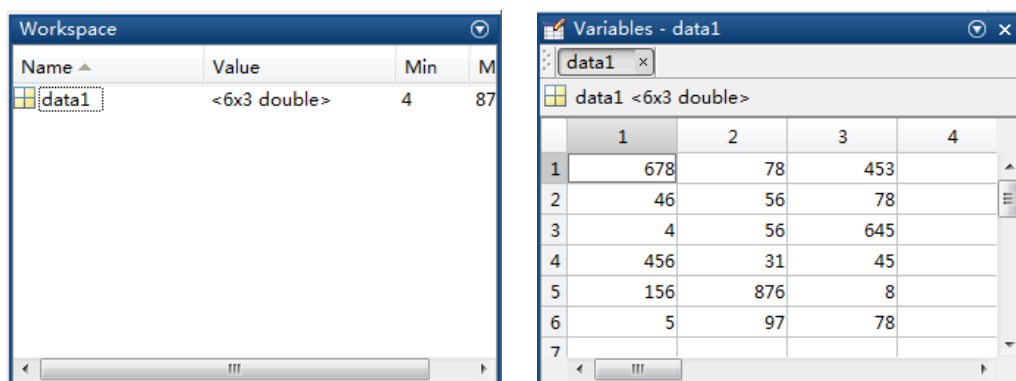


图 7 编辑矩阵

2. 矩阵的基本操作

1) 元素的提取

在矩阵操作中常常要获取矩阵中的某个特殊元素，在 MATLAB 中可以通过 `A(row,column)` 来提取单个元素，获取矩阵的部分元素可以采用冒运算符方法，具体如下。

`A(:)` 为 `A` 的所有元素。

`A(:, :)` 为二维矩阵 `A` 的所有元素。

`A(:, k)` 为 `A` 的第 `k` 列，`A(k, :)` 为 `A` 的第 `k` 行。

$A(k,m)$ 为 A 的第 k 个到第 m 个元素，默认是顺序是第 1 个是第 1 列第 1 行，第 1 列第 2 行，……，以此类推。

$A(:,k:m)$ 为 A 的第 k 列到第 m 列组成的子矩阵。

图图 8 为举例说明。

```
>> clear
>> a=[1,2,3]
a =
     1     2     3
>> A = vander(a) %范德蒙阵
A =
     1     1     1
     4     2     1
     9     3     1
>> A(3,1) %获得A第3行，第1列的元素
ans =
     9
>> A(5) %A的第5个元素的索引
ans =
     2
     .
>> A=hilb(3) %创建3维的希尔伯特矩阵
A =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
>> S = A(1,3) + A(2,3) + A(3,3)
S =
    0.7833
>> S = sum(A(1:3,3)) %提取矩阵多个元素，并计算和值
S =
    0.7833
>> S = sum(A(:)) %计算矩阵多有元素之和
S =
    3.7000
```

图 8 矩阵元素提取

2) 矩阵的旋转

矩阵的旋转操作在矩阵运算中也是经常用到的。MATLAB 提供的旋转操作主要有 3 个函数，分别是矩阵的左右旋转：flip1r(A)；矩阵的上下旋转：flipud(A)；矩阵逆时针旋转 90° ：rot90(A)、逆时针旋转 $k \times 90^\circ$ ：rot90(A,k)，如图 9。

```

>> clear
>> A = magic(3) %创建魔术矩阵A
A =
     8     1     6
     3     5     7
     4     9     2
>> B = fliplr(A)
B =
     6     1     8
     7     5     3
     2     9     4
>> C = flipud(A)
C =
     4     9     2
     3     5     7
     8     1     6
>> D = rot90(A,2) %逆时针旋转两个90°
D =
     2     9     4
     7     5     3
     6     1     8

```

图 9 矩阵的旋转

3) 矩阵的转置

矩阵的转置在控制理论等问题里面使用比较广泛，矩阵的转置与数组的转置操作是不相同的，在 MATLAB 中提供矩阵转置操作符为“'”，而数组转置操作符为“.'”，如图 10。

```

>> clear
>> A = [1 2;1i 2i]
A =
    1.0000 + 0.0000i    2.0000 + 0.0000i
    0.0000 + 1.0000i    0.0000 + 2.0000i
>> B = A' %矩阵转置
B =
    1.0000 + 0.0000i    0.0000 - 1.0000i
    2.0000 + 0.0000i    0.0000 - 2.0000i
>> C = A.' %数组转置
C =
    1.0000 + 0.0000i    0.0000 + 1.0000i
    2.0000 + 0.0000i    0.0000 + 2.0000i

```

图 10 矩阵的转置

4) 矩阵的缩放

矩阵的缩放包括矩阵的扩大和矩阵的缩小。

①矩阵的扩大。当将数据保存在矩阵的元素之外时，矩阵将会自动增大空间来保存这个新增的元素，如图 11。

```

>> clear
>> A = eye(3) %创建3维单位矩阵A
A =
    1    0    0
    0    1    0
    0    0    1
>> A(3,5) = 8
A =
    1    0    0    0    0
    0    1    0    0    0
    0    0    1    0    8

```

图 11 扩大

②矩阵的缩小。我们可以通过将行或列指定为空数组 **B** 从而删除矩阵中的行或列。但是不能从矩阵中删除单个的元素，如图 12。

```

>> clear
>> A = rand(3)
A =
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
>> A(1,3) = [] %从随机矩阵A中删除单个的元素，系统会报错
Subscripted assignment dimension mismatch.
>> A(:,2) = [] %删除一列
A =
    0.8147    0.2785
    0.9058    0.5469
    0.1270    0.9575

```

图 12 缩小

5) 获取矩阵的信息

在矩阵数值运算的时候常常要涉及矩阵的相关信息，MATLAB 提供了几个常用的函数来获取的矩阵信息，见表 2 矩阵常用命令。

表 2 矩阵常用命令

命 令	功 能
length(A)	获取矩阵 A 最长的维的长度
numel(A)	获取矩阵 A 的元素个数
size(A)	获取矩阵 A 的维数
ndims(A)	获取矩阵 A 维数的长度，即 ndims(A)=length(size(x))

3. 特殊矩阵

在矩阵论中介绍过许多特殊矩阵，这里介绍一些在 MATLAB 中常用到的特殊矩阵，它们在控制理论的问题中经常被用到，这些矩阵有的在本书之前的举例说明中出现过，这里再综合起来统一说明一下，见表 3 特殊矩阵。

表 3 特殊矩阵

矩阵名	函数名	矩阵特点	举 例	
			函数输入	结果
零矩阵	zeros()	元素全部为 0	zeros(3)	0 0 0 0 0 0 0 0 0
全 1 矩阵	ones()	元素全部为 1	ones(3)	1 1 1 1 1 1 1 1 1
单位矩阵	eye()	对角线为 1, 其余为 0, 通常用 I 表示	eye(3)	1 0 0 0 1 0 0 0 1
对角矩阵	diag()	对角线上为任意数, 其他元素为 0	a=[1,2,3] diag(a)	1 0 0 0 2 0 0 0 3
上三角阵	triu()	对角线的下面部分全为 0	a=[1 2 3;4 5 6; 7 8 9] triu(a)	1 2 3 0 5 6 0 0 9
下三角阵	tril()	对角线的上面部分全为 0	a=[1 2 3;4 5 6; 7 8 9] tril(a)	1 0 0 4 5 0 7 8 9
随机矩阵	rand()	由(0,1)区间内的随机数组成	rand(3)	0.9648 0.9571 0.1418 0.1576 0.4853 0.4217 0.9705 0.8002 0.9157
魔术矩阵	magic()	每行、每列以及对角线上的元素之和相等	magic(3)	8 1 6 3 5 7 4 9 2
范德蒙阵	vander()	呈现范德蒙式特点, 由矩阵元素次方组成	a=[1,2,3] vander(a)	1 1 1 4 2 1 9 3 1
伴随矩阵	company()	一个矩阵的伴随矩阵	A=[1 2 8 7] compan(A)	-2 -8 -7 1 0 0 0 1 0
Hilbert 阵	hilb()	又称病态矩阵, i 行 j 列的元素均为 $1/(i+j-1)$	hilb(3)	1 0.5000 0.3333 0.5000 0.3333 0.2500 0.3333 0.2500 0.2000

Hadamard 阵	hadamard()	元素均由 1 或-1 组成，且满足 $H' * N = N * I$ 。	hadamard(4)	$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$
---------------	------------	---	-------------	--

下面举例说明常用特殊矩阵在实际中的应用，如图 13 蒙特卡洛评估方法，和图 14 运用魔术矩阵绘制三维图形。

```
>> clear
>> close all
>> rand('seed',123456)
>> NumberInside = 0;
>> PiEstimate = zeros(500,1); %通过zeros函数创建一个500行，1列的零矩阵，并赋值给PiEstimate
>> for k = 1: 500 %for循环，k从1循环到500，k每取一个值，进行以下操作，直至k超出范围
x = -1 + 2*rand(100,1);
y = -1 + 2*rand(100,1);
NumberInside = NumberInside + sum(x.^2 + y.^2 <= 1); %x.^2表示x数组的每个值取2次方
PiEstimate(k) = (NumberInside / (k*100)) * 4; %计算估值
end %本次循环结束，当k值不在1~500范围内时，整个循环结束
>> plot(PiEstimate) %表示将估值曲线画出来
>> title(sprintf('Monte Carlo Estimate of Pi = %3.5f', PiEstimate(500))); %为曲线图添加图例
>> xlabel('Hundreds of Trials')
```

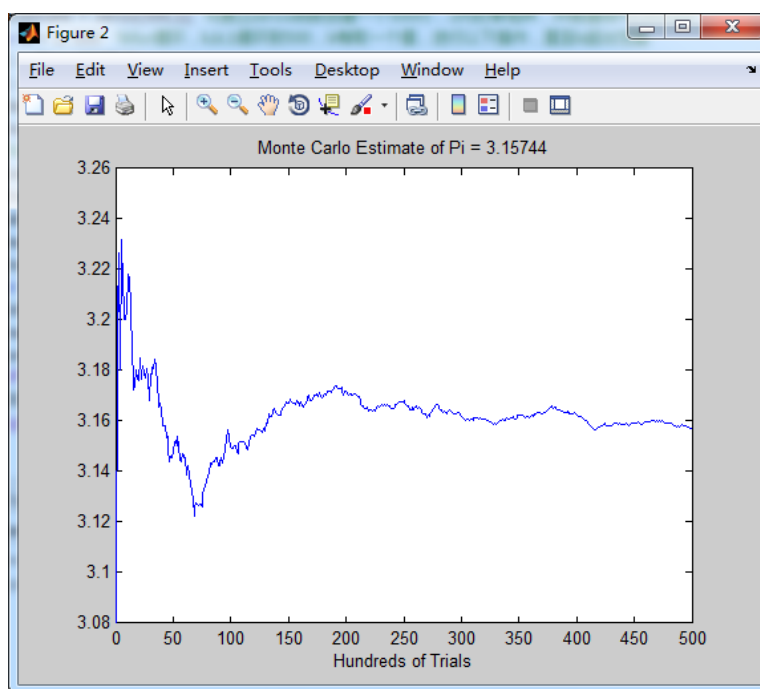


图 13 蒙特卡洛评估方法

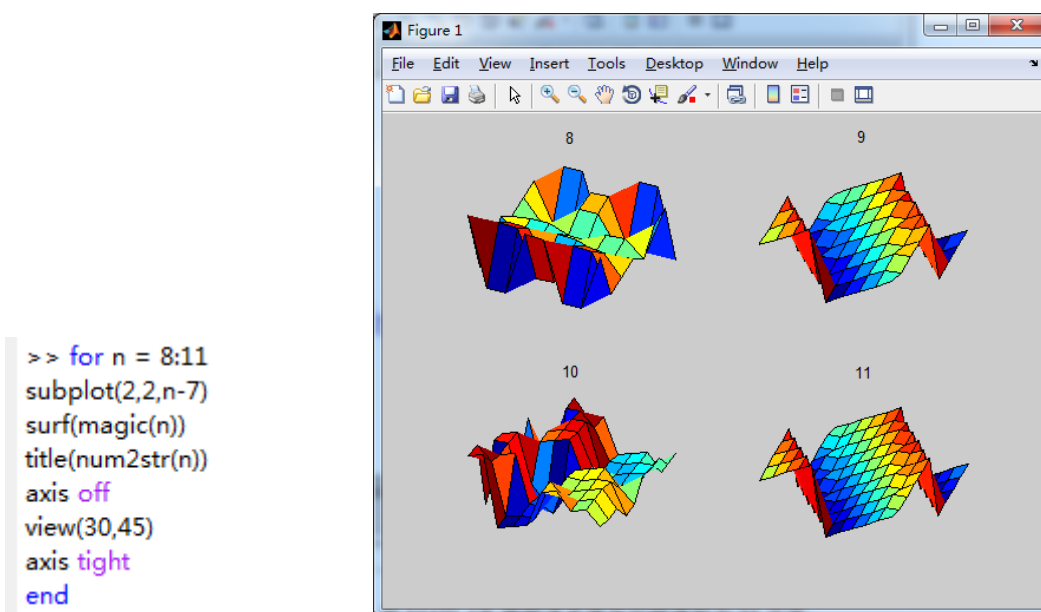


图 14 运用魔术矩阵绘制三维图形

4. 矩阵的基本数值运算

矩阵的基本运算主要包括矩阵和常数的运算、矩阵与矩阵的运算、矩阵的幂运算、指数运算、对数运算、开方运算、矩阵的逆运算，以及矩阵的行列式运算。

1) 矩阵的加法（减法）运算

矩阵的加法（减法）运算包括矩阵与常数之间和矩阵与矩阵之间的加法(减法)运算。前者是指矩阵中各元素与常数之间的运算，后者是指矩阵中各元素之间的加法(减法)运算，需要注意的是/在进行矩阵的加法(减法)运算时它们的维数必须相同，如图 15 矩阵的加法（减法）运算。

```

>> A = rand(3)
A =
    0.6126    0.4134    0.5953
    0.0709    0.6867    0.8137
    0.6878    0.9414    0.5012
>> B = A+10
B =
   10.6126   10.4134   10.5953
   10.0709   10.6867   10.8137
   10.6878   10.9414   10.5012
>> C = magic(3)
C =
     8     1     6
     3     5     7
     4     9     2
>> D = B - C
D =
     2.6126     9.4134     4.5953
     7.0709     5.6867     3.8137
     6.6878     1.9414     8.5012

```

图 15 矩阵的加法（减法）运算

2) 矩阵的乘法运算

矩阵的乘法运算不同于数组的乘法运算，它们的运算符号也不相同，前者为“*”，而后者

为“.*”。两矩阵在做乘法运算时内维必须一致，除非其中一个为标量；而数组运算的乘法运算要求两个数组必须大小相同，除非其中一个为标量，如图 16 矩阵的乘法运算。

```
>> A = pascal(3);
>> B = magic(3);
>> m = 3; n = 3;
>> for i = 1:m
for j = 1:n
C(i,j) = A(i,:) * B(:,j);
end
end
>> C
C =
    15    15    15
    26    38    26
    41    70    39
>> X = A * B
X =
    15    15    15
    26    38    26
    41    70    39
>> Y = B * A
Y =
    15    28    47
    15    34    60
    15    28    43
>> Z = B .* A
Z =
     8     1     6
     3    10    21
     4    27    12
>> ZZ = A .* B
ZZ =
     8     1     6
     3    10    21
     4    27    12
```

图 16 矩阵的乘法运算

3) 矩阵的除法运算

在 MATLAB 中，矩阵的除法运算有两种：左除和右除，运算符分别为“\”和“/”，二者的概念完全不同。矩阵的除法运算一般可以用来求解方程组的解，但应注意矩阵的左除用来求解线性方程组 $A \cdot X = b$ ，其中 $X = A \backslash b$ ；矩阵的右除用来求解线性方程组 $X \cdot A = b$ ，其中 $X = A / b$ ，如图 17 矩阵的除法运算。


```

>> clear
>> A = magic(3);
>> B = [1 3 4;2 1 2;2 1 1];
>> C1 = A\B %矩阵的左除，等价于inv(A)*B
C1 =
   -0.0139    0.3611    0.3639
    0.1944   -0.0556   -0.0944
    0.1528    0.0278    0.1972
>> C2 = A/B %矩阵的右除，等价于A*inv(B)
C2 =
   -1.2000    6.2000   -1.6000
    1.4000    0.6000    0.2000
    2.8000   -9.8000   10.4000

```

图 17 矩阵的除法运算

4) 矩阵的幂运算

矩阵的幂运算表达式为“ A^B ”，其中 A 可以是矩阵或者标量，且 B 不能为矩阵，如图 18 矩阵的幂运算。

```

>> A = rand(3)
A =
    0.2075    0.6227    0.3547
    0.8892    0.7432    0.5753
    0.2717    0.5842    0.3619
>> A^-2
ans =
  1.0e+04 *
    0.1978    0.0323   -0.2513
    0.4049    0.0677   -0.5168
   -0.8222   -0.1366    1.0482

```

图 18 矩阵的幂运算

图中，显示结果时，MATLAB 给出了结果显示的有效位数，表达为 $1.0e+04^*$ ，表示的是保留 4 为有效数字。

5) 矩阵的指数运算、对数运算和开方运算

矩阵的指数运算、对数运算和开方运算不是对矩阵中的单个元素的运算，而是对矩阵整体的运算。矩阵的这 3 种运算分别通过 MATLAB 提供的函数 `expm`、`logm`、`sqrtm` 来实现，这与对应的数组的 3 种运算函数 `exp`、`log`、`sqrt` 是不同的，如图 19 矩阵的指数运算、对数运算和开方运算。

```

>> a = [1 2 3]; %创建数组a
>> A = vander(a)
A =
     1     1     1
     4     2     1
     9     3     1
>> B = sqrtm(A) %矩阵的开方运算
B =
 0.8943 + 0.7487i  0.3536 + 0.0000i  0.2912 - 0.2496i
 1.3518 - 0.2496i  1.0607 + 0.0000i  0.3744 + 0.0832i
 2.6828 - 1.9965i  1.0607 - 0.0000i  0.8735 + 0.6655i
>> C = sqrt(A) %数组的开方运算
C =
 1.0000  1.0000  1.0000
 2.0000  1.4142  1.0000
 3.0000  1.7321  1.0000

```

图 19 矩阵的指数运算、对数运算和开方运算

6) 矩阵的逆运算

矩阵的逆运算即求矩阵的逆矩阵，它是矩阵运算中非常重要的运算之一。在线性代数里面求解逆矩阵是一件非常复杂的事情，在 MATLAB 中，提供的函数 `inv` 能够简便地解决问题，如图 20 矩阵的逆运算。

```

>> clear
>> A = [4 8 9; 7 6 3; 1 5 2];
>> inv(A)
ans =
 -0.0186  0.1801 -0.1863
 -0.0683 -0.0062  0.3168
 0.1801 -0.0745 -0.1988

```

图 20 矩阵的逆运算

7) 矩阵的行列式运算

在 MATLAB 中，直接使用函数 `det` 可求得矩阵的行列式大小，如图 21 矩阵的行列式运算。

```

>> A = [4 8 9; 7 6 3; 1 5 2];
>> det(A)
ans =
 161.0000

```

图 21 矩阵的行列式运算

8) 矩阵基本运算应用

矩阵在实际应用中的应用非常广泛，下面通过一个例子来说明。利用矩阵运算求解下面的线性方程组。

$$\begin{cases} 2x_1 + 5x_2 + 4x_3 = 7 \\ -x_1 + 3x_2 = 10 \\ x_1 - 2x_2 + 6x_3 = 3 \end{cases}$$

首先，把线性方程组变换成矩阵除法的运算形式： $A \cdot X = B$ ，有： $A = [2 \ 5 \ 4; -1 \ 3 \ 0; 1 \ -2 \ 6]$ ，

$B=[7;10;3]$ 。然后通过矩阵的左除运算即可求解方程组，如图 22。

```
>> A = [2 5 4;-1 3 0;1 -2 6]
A =
     2     5     4
    -1     3     0
     1    -2     6
>> B = [7;10;3]
B =
     7
    10
     3
>> X = A\B
X =
   -4.6774
    1.7742
    1.8710
```

图 22 矩阵运算例子

5. 矩阵的特征参数运算

特征值和特征向量的求解和运算问题是线性代数中一个重要的课题。它们在工程应用和科学实践中应用非常广泛，下面将介绍常见的矩阵特征参数的运算及其应用。

1) 特征值运算

在线性代数里面，求矩阵的特征值也是一件比较麻烦的事情。在 MATLAB 中，矩阵的特征值可以通过函数 `eig` 和 `eigs` 来得到，如图 23。

```
>> A = magic(5)
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
>> E = eig(A)
E =
   65.0000
  -21.2768
  -13.1263
   21.2768
   13.1263
>> [V,D] = eig(A) %V,D分别为矩阵A的特征向量和特征值组成的矩阵
V =
   -0.4472    0.0976   -0.6330    0.6780   -0.2619
   -0.4472    0.3525    0.5895    0.3223   -0.1732
   -0.4472    0.5501   -0.3915   -0.5501    0.3915
   -0.4472   -0.3223    0.1732   -0.3525   -0.5895
   -0.4472   -0.6780    0.2619   -0.0976    0.6330
D =
  65.0000     0     0     0     0
     0  -21.2768     0     0     0
     0     0  -13.1263     0     0
     0     0     0  21.2768     0
     0     0     0     0  13.1263
```

图 23 特征值运算

2) 秩运算

矩阵的秩在线性代数中运用也十分广泛，MATLAB 提供求矩阵的秩的函数为 rank，如图 24。

```
>> A = rand(5)
A =
    0.8147    0.0975    0.1576    0.1419    0.6557
    0.9058    0.2785    0.9706    0.4218    0.0357
    0.1270    0.5469    0.9572    0.9157    0.8491
    0.9134    0.9575    0.4854    0.7922    0.9340
    0.6324    0.9649    0.8003    0.9595    0.6787
>> r = rank(A)
r =
     5
```

图 24 秩运算

3) 逆运算及伪逆运算

矩阵的逆，有些资料和书籍上也称为矩阵的迹，指主对角线上所有元素的和。在 MATLAB 中，提供函数 trace 来进行逆运算，伪逆运算的函数为 pinv，如图 25。

```
>> clear
>> A = [1 6 8;4 0 9;3 7 5];
>> tra = trace(A)
tra =
     6
>> p = pinv(A)
p =
   -0.3103    0.1281    0.2660
    0.0345   -0.0936    0.1133
    0.1379    0.0542   -0.1182
```

图 25 逆运算及伪逆运算

4) 正交化运算

在矩阵论中，判断一个矩阵是否是正交矩阵的充分必要条件是矩阵的列向量都是单位向量，且两两正交。在 MATLAB 中，通过函数 orth 来求得矩阵的正交矩阵，如图 26。

```
>> A = [25 18 9;30 2 15;8 40 6];
>> B = orth(A)
B =
   -0.5874   -0.1954   -0.7854
   -0.4690   -0.7086    0.5271
   -0.6595    0.6780    0.3246
>> I = B*B' %验证B是否为正交矩阵
I =
    1.0000     0    0.0000
         0    1.0000   -0.0000
    0.0000   -0.0000    1.0000
```

图 26 正交化运算

5) 其他常用运算

矩阵的特征值还包括条件数、范数、奇异值等，这些运算在实际线性代数中都不太好求，在 MATLAB 中，都给出了对应的函数获取特征值，这里就不逐一举例说明，具体函数如表 4。

表 4 矩阵运算常用命令

函 数	功 能
-----	-----

cond()	计算矩阵的条件数，默认为 2-范式
condest()	计算 1-范式矩阵的条件数
recond()	计算矩阵的逆条件数
norm(A,'*')	计算矩阵的范数，*为类型，inf 表示无穷范数，fro 表示 Frobenius 范数，2 表示 2-范数
normest()	计算矩阵的范数，默认为 2-范数值
svd()	计算矩阵的奇异值
svds()	计算矩阵的奇异值（向量）

6. 矩阵的分解运算

MATLAB 有强大的数学处理能力，主要是因为它提供大量的矩阵运算函数，这些函数能够帮助用户非常轻松地解决数学计算中那些求解过程复杂的难题。这里将要介绍一些在数值分析中占据重要位置的分解运算。矩阵的分解运算是指将给定的矩阵分解成特殊矩阵的乘积的过程。一般的矩阵分解运算主要有：三角（LU）分解、正交（QR）分解、Chollesky（CHOL）分解、特征值（EIG）分解和奇异值（SVD）分解。下面就不一一阐述了。

三、常用运算

数学运算中除了前面的基本数值计算以外，还支持其他许多的科学运算方式。主要有符号表达式及其基本运算，包括符号精度的控制、符号矩阵和代数运算、符号微积分、积分变换和方程求解等，另外还有关系和逻辑运算、多项式运算等，它们可以广泛的应用于数学、物理、工程力学等各个学科的科研和工程中。

1. 符号运算

与数值运算的不同，数值运算中必须先对变量赋值，然后才能参与运算，而符号运算无须事先对独立变量赋值，运算结果以标准的符号形式表达。符号运算可以获得任意精度的解。

符号表达式一定要用单引号（'）括起来 MATLAB 才能识别，单引号内因内容可以是符号表达式，也可以是符号方程，如 $f1='a*x^2+b*x+c'$ 为二次三项式， $f2='a*x^2+b*x+c=0'$ 为方程。符号表达式或符号方程可以赋给符号变量，以方便调用，也可以不赋给符号变量直接参与运算。

MATLAB 中用 `sym(str)` 定义符号变量。MATLAB 支持的符号计算几乎和数值计算完全相同，关系和逻辑运算符同样也可针对符号变量进行。符号表达式到数值变量的转换可以用 `subs(f,x,y)` 实现，即用 `y` 替换表达式 `f` 中的 `x`，如果 `y` 仍是符号变量，则 `subs` 实现符号替换。而符号矩阵可以通过命令 `sym()` 创建，如图 27。

```
>> A = sym('[a, 2*b; 3*a, 0]')
A =
[ a, 2*b]
[ 3*a, 0]
>> clear
>> A = [1/3, 2.3; 1/0.7, 2/5];
>> sym(A)
ans =
[ 1/3, 23/10]
[ 10/7, 2/5]
```

图 27 创建符号矩阵

“clear”函数之前的部分，完成了一个符号矩阵的创建工作。符号矩阵的每一行的两端都有方括号，这是与 MATLAB 数值矩阵的一个重要区别。

另一方面，也将数值矩阵转化符号矩阵函数调用格式为 `sym(A)`，`A` 为数值矩阵，具体用法如图 27 “clear” 函数之后的部分。

数值运算中，所有矩阵运算操作指令都比较直观、简单，而符号运算就不同了，所有涉及符号运算的操作都由专用函数来进行，如表 5 所示为常用的符号运算函数。

表 5 常用的符号运算函数

函 数	功 能	函 数	功 能
<code>symadd()</code>	符号矩阵的加	<code>charploy()</code>	特征多项式
<code>symsub()</code>	符号矩阵的减	<code>determ()</code>	符号矩阵行列式的值
<code>symmul()</code>	符号矩阵的乘	<code>eigensys()</code>	特征值和特征向量
<code>symdiv()</code>	符号矩阵的除	<code>inverse()</code>	逆矩阵
<code>sympow()</code>	符号矩阵的幂运算	<code>transpose()</code>	矩阵的转置
<code>symop()</code>	符号矩阵的综合运算	<code>jordan()</code>	约旦标准型
<code>symsize()</code>	求符号矩阵的维数	<code>simple()</code>	符号矩阵的简化

2. 关系和逻辑运算

除了拥有强大的矩阵数学运算功能和前面介绍的符号运算外，**MATLAB** 同样拥有功能强大的关系运算和逻辑运算。在执行关系及逻辑运算时，**MATLAB** 将输入的不为 0 的数值都视为真（`true`），而为 0 的数值则视为否（`false`）。同时，判断为真者以 1 表示，而判断为否者以 0 表示，其中各个运算元须用在两个大小相同的数组或是矩阵中的比较。

1) 关系操作符

主要有 6 个关系操作符，见表 6。

表 6 常用关系操作符

关系操作符	说 明	对应的函数
<code>==</code>	等于	<code>eq(A,B)</code>
<code>~=</code>	不等于	<code>ne(A,B)</code>
<code><</code>	小于	<code>lt(A,B)</code>
<code>></code>	大于	<code>gt(A,B)</code>
<code><=</code>	小于等于	<code>le(A,B)</code>
<code>>=</code>	大于等于	<code>ge(A,B)</code>

2) 逻辑运算符

MATLAB 的逻辑运算符与其他语言一样主要有 3 种：逻辑与、逻辑或和逻辑非，其特点和关系运算符相似，相应的符号和函数见表****。

表 7 常用逻辑运算符

逻辑操作符	说 明	对应的函数
<code>&</code>	逻辑与	<code>and(A,B)</code>
<code> </code>	逻辑或	<code>or(A,B)</code>
<code>~</code>	逻辑非	<code>nor(A,B)</code>

3. 多项式运算

多项式运算是数学中最基本的运算之一，在许多学科里面都有着非常广泛的应用。MATLAB 提供了许多多项式运算函数，如多项式的求值、求根、多项式的微积分运算、曲线拟合、插值以及部分分式展开等，常用的一些函数如表 8 所示。

表 8 常用多项式运算函数

函 数	功 能
conv()	多项式相乘、卷积
deconv()	多项式相除、反卷积
poly()	用多项式的根求多项式系数
polyder()	多项式求导
polyfit()	多项式拟合
polyval()	代数多项式求值
polyvalm()	矩阵多项式求值
residue()	部分分式展开（残差运算）
roots()	多项式求根

利用 MATLAB 中的函数对多项式进行是非常简单的一件事情，这为工程应用和科学计算节约了大量的时间，如求多项式 $f(x)=5x^3+6x^2+3x+9$ 微分，如图 28，得到微分后的多项式为 $g(x)=15x^2+12x+3$ 。再对 $g(x)=15x^2+12x+3$ 求积分，得到的结果是 $f(x)=5x^3+6x^2+3x+C$ ，其中 C 为常数。

```
>> p = [5 6 3 9];
>> m = polyder(p)
m =
    15    12     3
>> s = length(m):-1:1
s =
     3     2     1
>> p = [m./s,0]
p =
     5     6     3     0
```

图 28 微分例子

4. 运算符的优先级

在 MATLAB 中各种运算符的优先级依次降低，如表 9。

表 9 优先级表

各种运算符的优先级依次降低
‘（矩阵转置）、^（矩阵幂）、.^（数组转置）、.^（数组幂）
~（逻辑非）
（乘）、/（左除）、\（右除）、.（点乘）、./（左点乘）、.\（右点除）
+（加）、-（减）

:、<、>、<=、>=、~=
&（逻辑与）
（逻辑或）
&&（先决与）
（先决或）

四、MATLAB 绘图

MATLAB 不仅具有强大的数值运算功能，还具有强大的二维和三维绘图功能，尤其擅长于各种科学计算运算结果的可视化。计算的可视化可以将杂乱的数据通过图形来表示，从中观测出其内在的关系。MATLAB 的图形命令格式简单，可以使用不同线型、色彩、数据点标记和标记等来修饰图形。

1. 二维绘图

在 MATLAB 中，绘制曲线的基本函数有很多，表 10 列出了常用的基本二维绘图函数。

表 10 二维绘图常用函数

函 数	功 能
plot()	二维绘图
plotyy()	在同一坐标轴中进行双 Y 轴绘图
semilogx()	X 轴以对数为刻度的二维绘图
semilogy()	Y 轴以对数为刻度的二维绘图
loglog()	X 轴和 Y 轴都以对数为刻度的二维绘图
polar()	绘制极坐标图
contour()	绘制等高线图

其中最常用的是 **plot 函数**，使用该函数可以非常简单地绘制出一个任意的二维图形，其他许多特殊绘图函数都是以它为基础而形成的。plot 函数主要用于绘制线性坐标平面图形，对于不同的输入参数，其用不同的形式可实现不同的绘制效果。plot 函数常用的形式有以下几种：**plot(x)**默认自变量，当 x 是实向量时，以该向量元素的下标为横坐标，元素值为纵坐标画出一条连续曲线，相当于绘制折线图；**plot(x,y)**绘制单条曲线，分别用于存储 x 坐标和 y 坐标数据，x, y 是相同类型的等长向量，表示线条上抽样点的两个坐标值；**plot(x1,y1,x2,y2,...)**绘制多条曲线，含义同 **plot(x,y)**；**plot(x,y,s)**函数中附加的 s 参数为定义线型及曲线其他属性（如颜色、宽度等）的字符串，可以控制对绘制的曲线进行修饰和控制。

2. 三维绘图

在 MATLAB 中，用户经常用到的绘制三维图形的命令有 **plot3** 函数、网格函数及着色函数。

plot3 函数也是绘制三维图形的基本命令，与二位图形的 plot 函数类似，是 plot 函数的三维扩展。**plot3** 函数语法与 plot 函数也类似，如 **plot3(x,y,z)**即表示绘制 x, y, z 向量的三维图形，**plot3(x,y,z,s)**中 s 有关取值与 plot 函数一致。

3. 统计绘图

在 MATLAB 中，二维统计分析图形很多，常见的有条形图、阶梯图、杆图和填充图等，常见的函数见表 11。

表 11 统计图

函 数	功 能
bar()	绘制条形图（竖直）
barh()	绘制水平条形图
errorbar()	绘制误差图
hist()	绘制柱形图
stem()	绘制火柴杆图
stairs()	绘制阶梯图
pie()	绘制扇形图

4. 图形修饰与控制

1) 绘图控制参数

绘图控制参数是指绘图函数中，用于控制图形属性，如颜色、线型和点型的参数，如 `plot(x,y,'s')` 中的 's'。MATLAB 提供了一些绘图选项，用于确定所绘曲线的线型、颜色和数据点标记符号，它们可以组合使用。例如，'b:.' 表示蓝色点划线，'y:d' 表示黄色虚线并用菱形符标记数据点。当选项省略时，MATLAB 规定，线型一律用实线，颜色为蓝色。常用绘图控制参数见表 12。

表 12 绘图参数

线 型	基本点标记	颜色
- 实线	. 点	y 黄色
: 虚线	o 小圆圈	m 棕色
-. 点划线	× 叉子符	c 青色
-- 间断线	+ 加号	r 红色
	* 星号	g 绿色
	s 方格	b 蓝色
	d 菱形	w 白色
		k 黑色

2) 图形标注与坐标控制

为了美化图形，还常用如下图形的修饰与控制函数，见表 13。

表 13 标注与坐标

函 数	功 能
title()	给图形加标题
xlabel()	给 X 轴加标注
ylabel()	给 Y 轴加标注
text()	在图形指定的任意位置加标注
gtext ()	利用鼠标将标注加到图形任意位置
grid on/off	打开/关闭坐标网格线
legend()	添加图例
axis()	控制坐标轴刻度（默认为 1）

附录 2 MATLAB 程序设计

通过前面的介绍已经知道，MATLAB 语言提供了各种功能强大的函数库，为了调用这些函数实现用户所需的计算和仿真功能，需要进行程序设计、调试和运行。

一、M 文件

在实际应用中，直接在 MATLAB 工作空间的命令窗口中输入简单的命令并不能够满足用户的所有需求，因此 MATLAB 提供了另一种强大的工作方式，即利用 M 文件编写工作方式。

M 文件因其扩展名为 .m 而得名，它是一个标准的文本文件，因此可以在任何文本编辑器中进行编辑、存储、修改和读取。M 文件的语法类似于一般的高级语言，是一种程序化的编程语言，但又比一般的高级语言简单、直观，且程序易调试、交互性强。MATLAB 在初始运行 M 文件时会将其代码装入内存，再次运行该文件时会直接从内存中取出代码运行，因此会大大加快程序的运行速度。

1. M 文件分类

M 文件分为两种，一种是脚本文件(Scripts File)，另外一种是一种函数文件(Function File)。

MATLAB 脚本文件类似于 DOS 系统中 .bat 批处理文件，即脚本文件通常为一连串的 MATLAB 指令，可以将烦琐的计算或操作放在一个 M 文件里面，每当调用这一连串指令时，只需输入 M 文件名即可，从而简化了操作。MATLAB 脚本文件通常无输入参数和返回参数，利用的数据和产生的中间结果都保存在 MATLAB 的基本工作空间。

MATLAB 函数文件与脚本文件不同，可以接受输入变量，并可返回结果。M 函数文件通常在扩充 MATLAB 函数库中使用，并且可以接受参数，也可以返回参数，而且不像 C 语言一个函数只能返回一个值，MATLAB 函数可以返回任意多个值，其利用的数据和产生的中间结果都保存在函数本身独立的工作空间中。

M 函数文件的实现对于用户来说是透明的。M 函数文件运行时，会创建此函数的函数工作空间(Function Workplace)，运算中产生的变量都存放在这个工作空间中，与其他函数空间和 MATLAB 基本工作空间相独立。因此大型的程序宜用函数文件，便于封装与调试。

2. M 文件编辑

新建脚本文件可以单击按钮【NewScript】或【New】-【Script】，MATLAB 会自动弹出 Editor 窗口，在该窗口上可以对新建的 M 文件进行编辑、调试。类似的，新建一个函数文件可以在菜单栏选择【New】-【Function】，MATLAB 自动弹出的 Editor 窗口，该窗口与脚本文件编辑窗口是一致的。由于 .m 文件可以通过记事本等文本工具编辑和查看，反之，普通的文本文件也可以通过 MATLAB 打开和编辑，只需要将后缀名改成 .m 即可。M 文件的编辑窗口如图 29，一个窗口允许同时打开多个文件，新建的脚本文件是空的，函数文件里面已经有几行代码。

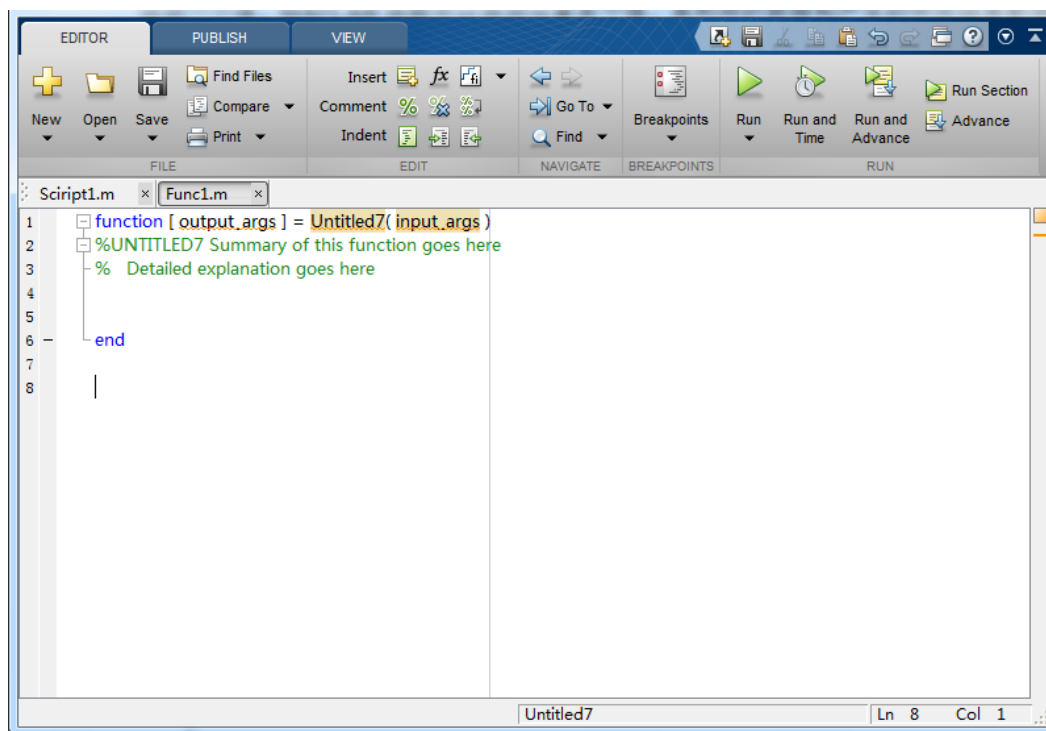


图 29 M 文件编辑

图 29 中，output_args 和 input_args 代表函数的输出和输入，在编辑函数文件时，将参数输入其中，一般在函数文件头一行注释说明函数的作用。函数的内容，在以下的空白处进行编辑，图 30 是一个编辑好的简单的函数文件。

```

1  function [a,b,c] = Func1(x,y,z)
2  %Func1 Summary of this function goes here
3  % Detailed explanation goes here
4  a = x+y+z;
5  b = a/3;
6  c = sqrt((x-b)*(x-b) + (y-b)*(y-b) + (z-b)*(z-b));
7  end
8
9

```

图 30 M 文件开头

函数文件必须满足一些标准，主要有以下几点：

- 1) 函数文件名和出现在文件的第一行的函数名一般应相同。实际上，MATLAB 忽略了第一行的函数名，并且根据实际存储的 M 文件名来执行函数。
- 2) 函数的文件名最多可以使用的字符数有一定限制，其最大值由操作系统决定。
- 3) 函数名必须以一个字母开头。开头之后，可以是任意字母、数字和下划线的组合。这个命名规则与变量的命名规则相同。
- 4) 一个函数文件的第一行被称为“函数声明行”，且函数式 M 文件必须包括 **function** 这个关键词。其后就是这个函数最常用的方式调用的语法。在第一行声明的输入和输出变量是这个函数的局部变量。输入变量包含传递给这个函数的数据，输出变量包含从这个函数输出的变量。
- 5) 在函数声明行之后的第一个连续的注释行的集合是这个函数的帮助文本。第一个注释

行被称作 H1 行。这一行也是 `lookfor` 命令搜索的靠。H1 打通常包括大写的函数名以及这个函数功能的一个简要描述。在第一行之后的注释行描述了可能的调用语法所使用的算法，而且可能会有简单的示例。

6) 在第一个连续注释行集合之后的所有语句构成了函数体。一个函数的函数体包含了对输入参数进行运算并将运算结果赋值给输出参数的 **MATLAB** 语句。

7) 函数文件可以包含对脚本文件的调用。当遇到一个脚本文件时，这个脚本文件就在这个函数的工作区执行，而不是在 **MATLAB** 的工作区执行。

8) 一个函数文件中可以出现多个函数。这些函数被称作子函数或者是局部函数。子函数以一个标准的函数声明语句开始，并且遵循所有的函数创建规则。子函数可以被这个 **M** 文件中的子函数调用，也可以被这个 **M** 数文件中的其他函数调用。

二、程序调试

在编制程序的过程中，不可避免地会遇到一些未知的错误，尤其是对初学者来说更是如此。此外，**MATLAB** 系统还提供了一个帮助用户提高 **M** 文件的执行速度的工具。在分析一个 **M** 文件的执行时，**MATLAB** 系统能够为用户标识出代码的哪一行花费的运行时间最长，这就为程序优化提供了便利。下面介绍程序调试常用的工具和方法。

1. 错误和警告

一般来说，应用程序的错误有两类，一类是语法错误，另一类是运行时的错误。

语法错误包括词法或文法的错误，例如函数名的拼写错、表达式书写错等。最常见的原因有两种，一是函数输入参数的类型错误，二是矩阵运算过程中阶数不匹配。对于语法错误，**MATLAB** 会立即标记出这些错误，并返回所遇到的错误类型及该错误所在 **M** 文件中的行数，利用这些信息可以很方便地查找相关的错误位置和类型。通过 **MATLAB** 帮助等，查阅对应的正确的数据类型或函数要求，可以方便的修改掉这些错误。

运行错误是指程序的运行结果有错误，出现溢出或者是死循环等异常现象，这类错误也称为程序逻辑错误。**MATLAB** 系统有时能够将这些错误语句标识出来，但一般情况下，这些错误都很难被发现。

除此之外，**MATLAB** 程序有时还会出现警告 (**warning**)，这些警告有时并不影响程序的正常运行和正确结果的输出。在调试程序时，合理的调试顺序是先处理错误，并且按照错误的顺序逐一进行处理，最后处理警告。如果警告的事项不影响程序的正确运行和运行速度，可以选择性的忽略。

2. 设置断点

在调试程序和改正错误时，断点是十分必要的一种方法，尤其在处理程序错误时，断点是查找错误必不可少的。

首先选择程序运行时需要中断的地方，一般选择条件、循环语句开始或结束的地方，或者是一个变量计算完成需要检验正确与否的地方。然后在菜单命令选择【**breakpoints**】-【**Set/Clear**】，或使用快捷键 **F12**。设置后，在相应的程序行的初始处会出现一个红点，如图 31。清除断点的方式与设置断点一致，即再按一次【**breakpoints**】-【**Set/Clear**】或 **F12**。通过命令【**breakpoints**】-【**Set Condition**】还可以设置条件断点，使用时，在对话框输入条件表达式即可。

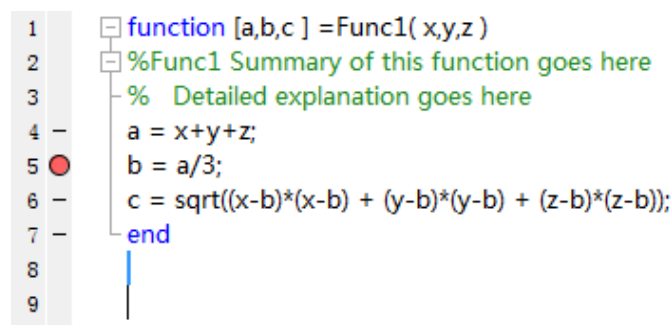


图 31 断点

断点设置之后，选择【Run】或按 F5 键运行程序，程序运行到断点处将自动停止，并且在工作区会自动显示已经用到的数据项的值，这时，就可以检验这些值是否正确。然后，选择【Continue】或者按 F5 键，程序将继续运行至下一个断点处。

3. 简单程序的调试方法

调试函数 M 文件有多种方法，对于简单问题，用下述的一种或者几种方法来解决是常直观的。

1) 将函数中被选定的行的分号去掉，这样运算的中间结果就可以在控制窗口汇总显示出来。

2) 在 M 文件中选定的位置置入 `keyboard` 命令，以便将临时控制权交给键盘，这样做以后，函数工作区就可以进行查询，并且可以根据需要改变变量的值。在键盘提示符下输入 `return` 命令就可以恢复函数执行，也就是说，在 `k>>` 下输入 `return` 就可以了。

3) 通过在 M 文件开头的函数定义语句之前插入 `%`，把函数 M 文件变成脚本 M 文件。在以脚本 M 文件执行的时候，其工作空间就是 MATLAB 基本工作空间，这样在出现错误的时候可以查询这个工作空间了。

4) 在适当的位置利用命令显示变量值，或者设置断点，查看和判断运行到该位置时输出值是否正确。

5) 领用 `echo on` 和 `echo off` 显示执行的指令行，判断程序流是否正确。

程序调试还有许多其他的工具和方法，这里就不逐一进行介绍了，随着对 MATLAB 的进一步使用，读者可以查阅相关资料深入研究。