第 2 章

Matlab 基础知识

Matlab 语言以前是一种专门为进行矩阵计算所设计的语言，在以后的各个版本中逐步扩充其各种功能。现在 Matlab 不仅仅局限于矩阵计算领域，但其最基本、最重要的功能还是进行实数矩阵和复数矩阵的运算。在本章中，主要介绍 Matlab 语言及命令的基本知识，这是 Matlab 最基本和最重要的部分，对本章内容的深入理解和掌握是对其他各章进行理解和运用并对 Matlab 进行扩展的基础。用户在学习完本章的内容后，可以进行基本的数值计算，从而能够容易地解决许多学习和科研中遇到的计算问题。

学习目标

- 掌握一般运算符及操作符
- 掌握关系运算符
- 掌握逻辑运算及逻辑函数
- 掌握数值数据类型
- 了解 Matlab 的函数和特殊函数

2.1 一般运算符及操作符

一般运算符和操作符可以构成运算的最基本的操作指令，如加、减、乘、除和乘方等运算，这些运算指令几乎在所有计算机语言中都有，且大同小异。在 Matlab 中，几乎所有的操作都是以矩阵为基本运算单元的，这与其他计算机语言有很大不同，也是 Matlab 的重要特点，在以后的学习中应该充分理解和注意。

2.1.1 运算符

1. 矩阵的加减运算

其基本形式为 X±Y，X 和 Y 必须为同维的矩阵，此时各对应元素相加减。如果 X 与 Y 的维数不相同，则 Matlab 将给出错误信息，提示用户两个矩阵的维数不匹配。

例 2.1 矩阵加减

Matlab 中的矩阵加减运算：

```
>>X=(2  3; 4  5)
 X=
   2    3
   4    5
>>Y=(3  4; 4  3)
 Y=
   3    4
   4    3
X+Y=
   5    7
   8    8
X-Y=
   -1   -1
   0    2
```

2. 矩阵的乘法运算

X*Y 是两个矩阵 X 和 Y 的乘积，其中 X 和 Y 必须满足矩阵相乘的条件，即矩阵 X 的列数必须等于矩阵 Y 的行数。如果其中一个为 1×1 矩阵亦合法，此时便是将每一个矩阵的元素都分别与这个数值相乘。

例 2.2 矩阵乘法

Matlab 中的矩阵乘法运算：

```
>>X*Y
ans =
    18   17
    32   31
X*2
```

```
ans =
    4    6
    8   10
```

3. 矩阵的数组乘法

X.*Y 运算结果为两个矩阵的相应元素相乘，得到的结果与 X 和 Y 同维，此时 X 和 Y 也必须有相同的维数，除非其中一个为 1×1 矩阵，此时运算法则与 X*Y 相同。

例 2.3　数组乘法

Matlab 中的数组乘法运算：

```
>> X.*Y
ans =
    6   12
   16   15
>> 2.*X
ans =
    4    6
    8   10
```

4. 矩阵的乘方运算

(1)　x^Y 表示，如果 x 为数，而 Y 为方阵，结果由各特征值和特征向量计算得到。

(2)　X^y 表示，如果 X 是方阵、y 是一个大于 1 的整数，所得结果由 X 重复相乘 y 次得到；如果 y 不是整数，则将计算各特征值和特征向量的乘方。

(3)　如果 X 和 Y 都是矩阵，或 X 或 Y 不是方阵，则会显示错误信息。

例 2.4　矩阵乘方

Matlab 中的矩阵乘方运算：

```
>> X^2
ans =
   16   21
   28   37
>> X^1.5
ans =
  5.9125 - 0.1007i   7.7970 + 0.0573i
  10.3960 + 0.0764i  13.7095 - 0.0434i
>> 2^Y
ans =
  64.2500  63.7500
  63.7500  64.2500
```

5. 矩阵的数组乘方

X.^Y 的计算结果为 X 中元素对 Y 中对应元素求幂，形成的矩阵与原矩阵维数相等，这里 X 和 Y 必须维数相等，或其中一个为数，此时运算法则等同于 X^Y。

例 2.5　数组乘方

Matlab 中的数组乘方运算：

```
>> X.^Y
ans =
    8   81
  256  125
```

6. 矩阵的左除运算

A\B 称作矩阵 A 左除矩阵 B，其计算结果大致与 INV(A)B 相同，但其算法却是不相同的。如果 A 是 N×N 的方阵，而 B 是 N 维列向量，或是由若干 N 维列向量组成的矩阵，则 X=A\B 是方程 AX=B 的解，X 与 B 的大小相同，对于 X 和 B 的每个列向量，都有 AX(n)=B(n)，此解是由高斯消元法得到的。很显然，A\EYE(SIZE(A))=INV(A)EYE(SIZE(A))=INV(A)。如果 A 是 M×N 的矩阵(M≠N)，B 是 M 维列向量或由若干 M 维列向量组成的矩阵，则 X=A\B 是欠定或超定方程 AX=B 的最小二乘解。A 的有效秩 L 由旋转的 QR 分解得到，并至多在每列 L 个零元素上求解。

例 2.6　矩阵左除

Matlab 中的矩阵左除运算：

```
>>A =(1  2; 3  4)
 A=
    1    2
    3    4
>>B =(2  3; 3  2)
 B=
    2    3
    3    2
>> A\B
ans =
   -1.0000   -4.0000
    1.5000    3.5000
```

7. 矩阵的右除运算

B/A 称为矩阵 A 右除矩阵 B，其计算结果基本与 B*INV(A)相同，但其算法是不同的，可以由左除得到，即：B/A=(A'\B')'。它实际上是方程 XA=B 的解。

例 2.7　矩阵右除

Matlab 中的矩阵右除运算：

```
>> B/A
ans =
    0.5000   0.5000
   -3.0000   2.0000
>> (A'\B')'
ans =
    0.5000   0.5000
   -3.0000   2.0000
```

8. 矩阵的点除运算

B./A，如果 B 和 A 都是矩阵，且维数相同，则 B./A 就是 B 中的元素除以 A 中的对应元素，所得结果矩阵的大小与 B 和 A 都相同；如果 B 和 A 中有一个为数，则结果为此数与相应的矩阵中的每个元素做运算，结果矩阵与参加运算的矩阵大小相同。

例 2.8　矩阵点除

Matlab 中的矩阵点除运算：

```
>> B./A
ans =
   2.0000   1.5000
   1.0000   0.5000
>> B./2
ans =
   1.0000   1.5000
   1.5000   1.0000
```

9. 矩阵的 KRONECKER 张量积

K=KRON(A,B)返回 A 和 B 的张量积，它是一个大矩阵，取值为矩阵 A 和 B 的元素间所有的可能积。如果 A 是 m×n 矩阵，而 B 是 p×q 矩阵，则 KRON(A,B)是 mp×nq 的矩阵。例如，A 是 2×2 的矩阵，则有下式成立：

KRON(A，B)=[A(1,1)*B　　A(1,2)*B

　　A(2,1)*B　　A(2,2)*B]

如果 A 和 B 中有一个为稀疏矩阵，则只有非零元素会参与计算，所得的结果亦是稀疏矩阵。

例 2.9　矩阵张量积

Matlab 中的矩阵 KRONECKER 张量积运算：

```
>> kron(A,B)
ans =
    2    3    4    6
    3    2    6    4
    6    9    8   12
    9    6   12    8
```

2.1.2　操作符

1. 冒号“:”

此符号在矩阵的构造和运算中非常有用，它可以用来产生向量，用作矩阵的下标，以及部分地选择矩阵的元素，进行循环操作等，熟练掌握可以在矩阵的运算中受益匪浅。其基本用法有：

j:k　　等价于[j,j+1,…,k]

j:i:k　　等价于[j,j+I,j+2*I,…,k]

j:k 中，如果 j<k 则返回空值；j:i:k 中，如果 j>k 并且 i>0，或 j<k 并且 i<0，都会返回空值。

A(:,i)取 A 矩阵的第 i 列。

A(i,:)取 A 矩阵的第 i 行。

A(:,:)以 A 的所有元素构造二维矩阵，如果 A 是二维矩阵，则结果就等于 A。

A(j:k)等价于 A(j)，A(j+1)，…，A(k)。

A(:,:,k)三维矩阵 A 的第 3 页。

A(:)将所有 A 的元素作为一个列向量，如果此操作符在赋值语句的左边，则用右边矩阵的元素来填充矩阵 A，矩阵 A 的结构不变，但要求两边矩阵的元素个数相同，否则会出错。

例 2.10　使用冒号

Matlab 中冒号的用法：

```
>> a=rand(3,4)
a =
   0.8147   0.9134   0.2785   0.9649
   0.9058   0.6324   0.5469   0.1576
   0.1270   0.0975   0.9575   0.9706
>> b=rand(2,2,3)
b(:,:,1) =
   0.9572   0.8003
   0.4854   0.1419
b(:,:,2) =
   0.4218   0.7922
   0.9157   0.9595
b(:,:,3) =
   0.6557   0.8491
   0.0357   0.9340
>> b(:,:,3)
ans =
   0.6557   0.8491
   0.0357   0.9340
>> b(:,:)
ans =
   0.9572   0.8003   0.4218   0.7922   0.6557   0.8491
   0.4854   0.1419   0.9157   0.9595   0.0357   0.9340
>> a(:)=b
a =
   0.9572   0.1419   0.7922   0.0357
   0.4854   0.4218   0.9595   0.8491
   0.8003   0.9157   0.6557   0.9340
```

2. 百分号“%”

百分号在 M 文件和命令行中表注释，即在一行中百分号后面的语句都被忽略而不被执行。在 M 文件中，百分号后面的语句可以用 help 命令打印出来。

3. 连续号“…”

如果一条命令很长，一行容不下，可以用空格加 3 个或更多的点加在一行的末尾，表示此行未完，而在下一行继续。

4. 单引号“'”

表示矩阵的转置。

5. 分号“;”

分号用在“[]”内，表示矩阵中行的结尾；也可以用在每行命令的结尾，则命令不会回显。可以用在 M 文件中控制命令的显示，并压缩输出篇幅。

例 2.11　使用分号

Matlab 中分号的用法：

```
>> c=[1 2;3 4]
c =
    1    2
    3    4
```

2.2　数据格式显示

虽然在 Matlab 系统中数据的存储和计算都是以双精度进行的，但其显示格式却可以有不同的形式。在 Matlab 的命令行中，通常可以利用 format 命令来调整数据的不同显示格式。format 命令的格式和作用如下。

(1) format：默认值，数据显示格式与 short 格式相同。

(2) format short：短格式，只显示 5 位数值。对于小于 1000 且大于 0.0001 的小数，则整数部分会按数据的原格式显示，而小数只会显示到小数后面 4 位，如 123.45678 即显示 123.4568；对于大于 1000 或小于 0.0001 的小数，其显示格式与 format short e 相同，即小数部分只显示 5 位数，如输入 0.00001 便会显示 1.0000e-005，而输入 1000.123456 则会显示 1.0001e+003，而对于整数，绝对值小于 10^9 则按原样输出，而大于等于 10^9 的数则与 format short e 相同。可以通过实验来比较它与 format short e 之间的联系与区别。

(3) format long：长格式，显示 15 位数。即将所有的小数都用 e 格式输出，e 左边为 15 位数，如输入 1.2 则会输出 1.200000000000000e+000，这与 format long e 格式基本相同；而对于整数，绝对值小于 10^9 的数按原样输出，而大于等于 10^9 的数则按 e 格式输出，如输入 10000 则显示 10000，而输入-1000000000 则显示-1.000000000000000e+009。

(4) format short e：短格式 e 方式，对于任意小数采用 e 格式，只显示 5 位小数，如输入 1.2 则会显示 1.2000e+000；对于整数，其显示格式与用 format short 格式时相同。

(5) format long e：长格式 e 方式，显示 15 位小数。对于任意小数都会采用 e 格式显示，

如输入 1.2 则显示 1.200000000000000e+000；对于整数，其显示格式参照 format short。

(6) format short g：最优化短格式，最大显示 5 位数据，系统会根据数据的大小及形式采用比较好的显示数据的方式，如略去小数后面的零等。如输入 1.234e1 则显示 12.34，输入 1234.5678 则显示 1234.6，输入 1234567.8888 则显示 1.2346e+006，输入 0.02000 则显示 0.02 等，其显示方式看起来比较灵活和更加合理。对于整数的显示方式，可以参照 format short 格式。

(7) format long g：最优化长格式，最大显示 15 位数据。其显示规则大体与 format short g 相同，现举例加以说明。输入 1.2 则显示 1.2，输入 1.2345678900 则显示 1.23456789，输入 1.234567890123456789 则显示 1.23456789012346，输入 1234567890123456.1234567 则显示 1.23456789012346e+015.

(8) format hex：十六进制格式。显示二进制双精度数的 16 进制形式，如输入 1.2 则显示 3ff3333333333333，输入-2.3456 则显示 c002c3c9eecbfb16。

(9) format bank：货币银行格式。保留小数点后两位小数，对于整数亦是如此。但并不是简单的四舍五入法，如输入 0.355 则显示 0.36，而输入 0.345 则显示 0.34，对于任意整数其后都显示两位小数，如输入 1 则显示 1.00。

(10) format rat：有理格式。对于整数则照常显示，而对于小数则用两个整数相除的方式显示。如输入 0.5 及 1/2 则都会显示 1/2，输入 6.33333 则显示 19/3。注意：用有理小数表示分数是有一定精度要求的，如输入 6.3333 则只会显示 63333/10000 而不会显示 19/3。

(11) format +：紧密格式。用“+”、“-”或空格表示数据为正、负或零。如输入 1-2 则会显示-号，输入 3 则会显示+号，输入 0 则会输出一个空格。

(12) format compact：紧凑格式，即在输入的命令及回显结果之间不加空行。

(13) format loose：疏松格式，即在输入命令及回显结果之间都加空行。

例 2.12　使用 format 命令

Matlab 中 fromat 命令的用法：

```
>> format compact
>> a=[2 3;4 5]
a =
     2     3
     4     5
>> format loose
>> a=[2 3;4 5]

a =

     2     3
     4     5
```

注意　不论采用什么样的显示格式，数据在内存中的格式是不会变的，即不影响数据的存储。对于 a=19/3，当用 format short 格式显示时是 6.3333，而改用 format rat 显示时仍是 19/3，不会因为上面已经用 short 格式显示过而丢失精度。

2.3　关系运算符

关系运算符主要用来对数与矩阵、矩阵与矩阵进行比较，并返回反映二者之间大小关系的由数 0 和 1 组成的矩阵。基本的关系运算符主要有：>、<、<=、>=、==、~=这 6 个。下面分别介绍它们的用法。

1. 大于：“>”

A>B，如果 A 矩阵中的元素大于 B 矩阵中相应位置的元素，则在输出矩阵的此位置上输出 1，反之则输出 0；如果其中之一为数值，则会将这个数与另一对象的每一个元素进行比较。

函数 gt(A,B)亦是判断 A 是否大于 B，A 和 B 可以是矩阵、数值或任意其他的对象。

2. 小于：“<”

A<B，如果 A 矩阵中的元素小于 B 矩阵中相应位置的元素，则在输出矩阵的此位置上输出 1，反之则输出 0；如果其中之一为数值，则会将这个数与另一对象的每一个元素进行比较。

函数 lt(A,B)亦是判断 A 是否小于 B，A 和 B 可以是矩阵、数值或任意其他的对象。

3. 大于等于：“>=”

A>=B，如果矩阵 A 中的元素大于或等于矩阵 B 中相应位置的元素，则在输出矩阵的此位置上输出 1，反之则输出 0；如果其中之一为数值，则将这个数与另一对象的每一个元素进行比较。

函数 ge(A,B)亦是判断 A 是否大于或等于 B，A 和 B 可以是矩阵、数值或任意其他的对象。

4. 小于等于：“<=”

A<=B，如果矩阵 A 中的元素小于或等于矩阵 B 中相应位置的元素，则在输出矩阵的此位置上输出 1，反之则输出 0；如果其中之一为数值，则将这个数与另一对象的每一个元素进行比较。

函数 le(A,B)亦是判断 A 是否小于或等于 B，A 和 B 可以是矩阵、数值或者任意其他的对象。

5. 等于：“==”

A==B，如果 A 和 B 都为矩阵，则 A 和 B 必须具有相同的维数，运算时将 A 中的元素和 B 中的对应元素进行比较，如果两者相等，则在输出矩阵的对应位置输出 1，反之输出 0。如果 A 和 B 有一个为数，则将这个数与另一个矩阵的所有元素进行比较。无论何种情况，

返回结果都是同参与运算的矩阵有相同维数的由 0 和 1 组成的矩阵。其余关系运算中对 A 和 B 的要求和返回结果的维数所满足的条件亦是如此。

函数 eq(A,B)是对两个对象进行比较，看是否相等。其中，A 和 B 可以是矩阵和数值，但也可以是其他的对象，如 figure 对象。

6. 不等于：“~=”

A~=B，与 A==B 相反，如果 A 和 B 中相同位置上的对应元素不相等，则在返回矩阵的相应位置上输出 1，反之输出 0。由此可见，所得矩阵应该与 A==B 得到的矩阵中 0 和 1 的位置相互颠倒。

同样，函数 ne(A,B)对两个对象 A 和 B 进行比较，也可以用于对两个矩阵进行比较。

例 2.13　关系运算符示例

Matlab 中关系运算符的用法：

```
>> a=[1 2;3 4]
a =
     1           2
     3           4
>> a>1
ans =
    0    1
    1    1
>> a<3
ans =
    1    1
    0    0
>> a>=2
ans =
    0    1
    1    1
>> a<=2
ans =
    1    1
    0    0
>> eq(a,b)
ans =
    1    1
    0    0
>> a==1       %找出 a 中等于 1 的元素
ans =
    1    0
    0    0
>> a~=b
ans =
    0    0
    1    1
```

2.4　逻辑运算及逻辑函数

逻辑运算和逻辑函数在计算机语言中是普遍存在的，在 Matlab 中包含与、或、非、异或 4 种基本的逻辑运算。逻辑表达式和逻辑函数的值应该为一个逻辑量“真”或“假”。Matlab 系统在给出逻辑运算的结果时，以数值“1”代表逻辑“真”，以“0”代表“假”，但在判断一个量是否为“真”时，以 0 代表“假”，以任意的非零值代表“真”。Matlab 的逻辑运算也是以矩阵为基本运算单元的。

2.4.1　逻辑运算

符号“&”、“|”、“~”、“xor”分别代表逻辑运算中的与、或、非、异或，它们作用于数组元素。0 的逻辑量为“假”，而任意非零数的逻辑量为“真”。逻辑运算的运算法则有：

- 在逻辑数组中，0 代表逻辑“假”，1 代表逻辑“真”。
- 如果两个标量 a 和 b 参加运算，则各逻辑运算的运算规则如表 2-1 所示。
- 如果两个维数相同的矩阵参与运算，则将 A 和 B 相同位置上的元素按标量逻辑运算的规则进行计算，结果返回与矩阵 A 和 B 同样大小的矩阵，其元素由同位置上的 A 和 B 的元素进行逻辑运算的结果所决定。
- 如果标量 a 和矩阵 A 参加运算，则将 a 和 A 中的所有元素进行逻辑运算，返回结果是由 0 和 1 组成的与 A 具有同样维数的矩阵。
- 在逻辑操作符、关系操作符和计算操作符三者中，逻辑操作符的优先级最小，但是逻辑“非”的优先级最高。
- 在逻辑“与”、“或”、“非”三者中，“与”的优先级高于“或”的优先级，而都低于“非”的优先级。
- 通过增加“( )”可以改变各操作符之间的优先级。

表 2-1　逻辑运算规则

输入		“与”	“或”	“异或”	“非”
A	B	a&b	a|b	xor(a,b)	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

例 2.14　使用逻辑运算符

Matlab 中逻辑运算符的用法：

```
>> a=[1 2;3 4]
a =
    1    2
    3    4
>> b=[1 0;0 1]
b =
    1    0
    0    1
>> a&b          %求逻辑“与”
ans =
    1    0
    0    1
>> a|b           %求逻辑“或”
ans =
    1    1
    1    1
>> ~b            %求逻辑“非”
ans =
    0    1
    1    0
>> xor(a,b)       %求逻辑“异或”
ans =
    0    1
    1    0
>> ~a-1>=1
ans =
    0    0
    0    0
>> a-1>=1&0
ans =
    0    0
    0    0
>> ~(a-1>=1)
ans =
    1    0
    0    0
>> ~a-1
ans =
   -1   -1
   -1   -1
>> 1|0&0
ans =
    1
>> 0&0|1
ans =
    1
>> and(a,b)
ans =
    1    0
    0    1
```

在 M 文件中，可以用 and(A,B)、or(A,B)、not(A)分别进行“ 与”、“ 或”、“ 非”的操作。亦可在命令行中进行直接输入，其结果与用“ &”、“ |”、“ ~”运算所得结果相同。

2.4.2　逻辑函数

Matlab 从 3.0 版本开始提供逻辑函数，这些函数在交互运算及进行矩阵的变化中非常有用，可以很方便地查找或替换矩阵中满足一定条件的部分或所有元素，在使用过程中应认真体会每个函数的具体用法，才能在实际应用过程中灵活运用。

1. all：判断是否所有元素为非零数

如果要获得矩阵或向量中非零元素的位置或个数，可以利用 all 函数。此函数对于向量 a(行向量或列向量)，如果向量中的每个元素都是非零数，all(a)返回逻辑“真”，即“1”，如果至少一个元素为零，则返回逻辑“非”，即“0”。对于矩阵 A，all(A)是作用于列向量上的，即如果矩阵 A 的某列所有元素都是非零数，则返回结果的当前列为逻辑“真”，即“1”；如果至少有一个为零，则返回结果的当前列为逻辑“假”，即“0”。显然，返回结果为与矩阵 A 具有同列维的行向量。如果 A 是多维矩阵，则 all(A)将是第 1 个不是单维的维作为向量，进行运算，运算规则同向量运算。all(A,dim)将指定的第 dim 维作为向量进行运算。

例 2.15　判断矩阵元素

判断矩阵 A=[0 1 2;3 4 5]的所有元素是否都大于或等于 1。

```
>> all(all(A))
ans =
    0
>> A>=1
ans =
    0    1    1
    1    1    1
>> all(A>=1)
ans =
    0    1    1
```

而对此向量再做一次 all 运算，便可以得到最终的结果 0。

从以上分析不难看出，如果 A=[6 1 2;3 4 5]，则 all(all(A>=1))=1。

例 2.16　使用 all 函数

举例说明 all( )函数运算的规则。

```
>> A=[0 1 2;3 4 5]
A =
    0    1    2
    3    4    5
>> all(A)
ans =
    0    1    1
>> all(A,2)
ans =
    0
    1
```

```
>> c=rand(1,2,3)
c(:,:,1) =
   0.8147   0.9058
c(:,:,2) =
   0.1270   0.9134
c(:,:,3) =
   0.6324   0.0975
>> all(c)
ans(:,:,1) =
    1
ans(:,:,2) =
    1
ans(:,:,3) =
    1
```

由上例可以看出，三维列向量 c 进行 all()的逻辑运算时，是将第二维作为向量进行计算的，而不是将第一维作为一个向量。

在 M 函数中，可以充分利用 all()函数的功能，对向量或矩阵进行判断，并根据返回结果做出相应的反应。

例如下面的一段程序：

```
if all(all(A>=1))
   %存放将要执行的代码
end
```

2. any：判断是否有一个向量元素为非零数

在矩阵处理中，有时候会需要判断矩阵中的元素是否有零或非零值，如在对矩阵进行数组除时，就需要判断作除数的矩阵是否有零元素，any 函数可以实现这一功能。此函数也有三种调用格式：any(a)、any(A)、any(A,dim)。对于向量 a(行向量或列向量)，如果向量中至少有一个元素为非零数，any(a)返回逻辑“真”(即“1”)，而如果所有元素都为零，则返回逻辑“非”(即“0”)。对于矩阵 A，any(A)与 all(A)一样，也是作用于列向量上的，即如果矩阵 A 的某列中存在某个元素为非零数，则返回结果的当前列为逻辑“真”，即为“1”，如果所有元素都为零，则返回结果的当前列为逻辑“假”，即“0”。如果 A 是多维矩阵，则 any(A)将第 1 个不是单维的维作为向量，进行运算，运算规则同向量运算。any(A,dim)将指定的第 dim 维作为向量进行计算。

例 2.17　使用 any 函数

举例说明 any( )函数运算的规则。

```
>> A=[0 1 2;0 3 4]
A =
    0    1    2
    0    3    4
>> any(A)
ans =
    0    1    1
```

```
>> any(A,2)
ans =
    1
    1
```

any()函数的其余规则与 all()基本相同，在此不再赘述。

3. exist：查看变量或函数是否存在

在 Matlab 程序设计中，有时候需要知道变量是否已经被定义过，即是否存在于当前内存中，有时候还需要详细了解变量的类型，这时 exist 函数就显得非常有用。a=exist(‘A’)返回变量或函数的状态或类型，a 的值及 A 对应的状态或类型分别为：

a 的值	**A 的状态或类型**
0	如果对象 A 不存在或没在 Matlab 的搜索路径下
1	如果 A 是工作空间中的一个变量
2	如果 A 是一个 M 文件或是一个在 Matlab 搜索路径下的未知类型的文件
3	如果 A 是一个 Matlab 搜索路径下的 MEX 文件
4	如果 A 是一个 Matlab 搜索路径下的已编译的 Simulink 函数(MDL 文件)
5	如果 A 是 Matlab 的内置函数
6	如果 A 是一个 Matlab 搜索路径下的 P 文件
7	如果 A 是一个路径，不一定是 Matlab 的搜索路径

如果对象 A 存在于 Matlab 的搜索路径下，但并不是 Matlab 可以识别的非 M 文件，即非 MDL 文件、P 文件、MEX 文件时，exist(‘A’)或 exist(‘A.ext’)将返回 2。

exist(‘A’)中，A 可以是当前路径下的子目录或相对路径。

例如，Matlab 的当前路径为 D:\Program Files\Matlab，则输入 exist(‘bin’)便会得到：

```
>> exist('bin')
ans =
    7
```

当然也可以检验计算机上的任意一条路径，但必须要输入全部路径。

例如下面一段程序内容：

```
>> exist('D:\Program Files\Matlab')
ans =
    7
```

findall.p 是 toolbox\matlab 目录下的一个 p 文件，输入：

```
>> exist('findall')
ans =
    6
>> exist('sin')
ans =
    5
```

flag=exist('A',kind)，如果 Matlab 找到指定类型 kind 下的对象 A 的话，此语句返回逻辑“真”，即 flag 值为 1，否则返回逻辑“假”，flag 值为 0。

其中，kind 参数的取值可以说明如下：

- exist('A','var')　　仅检查工作空间中的变量
- exist('A','builtin')　　仅检查 Matlab 的内置函数
- exist('A','file')　　检查 Matlab 搜索路径下的文件和路径
- exist('A','dir')　　仅检查路径

例如下面一段程序内容：

```
>> exist('a','var')          %在工作空间的变量中查找变量 a
ans =
    0                       %表明没有找到名为“a”的变量
>> exist('A','var')          %在工作空间的变量中查找变量 A
ans =
    1                       %表明找到名为“A”的变量
>> exist('A','builtin') )    %在内置函数中查找函数 A
ans =
    0                       %表明没有找到名为“A”的函数
```

4. find：找出向量或矩阵中非零元素的位置标识

在许多情况下，都需要对矩阵中符号某一特定条件的元素的位置进行定位，如将某一矩阵中为零的元素设为 1 等。如果这个矩阵的元素非常多，手工修改就非常麻烦，而灵活运用 find 函数和各种逻辑及关系运算可以实现绝大多数条件的元素定位。find 函数的基本用法有 k=find(A)，[i,j]=find(A)，[i,j,v]=find(A)，这是个很有用的逻辑函数，在对数组元素进行查找、替换和修改变化等操作中占有非常重要的地位，熟练运用可以方便而灵活地对数组进行操作。

k=find(A) 此函数返回由矩阵 A 的所有非零元素的位置标识组成的向量。如果没有非零元素则会返回空值。

例 2.18　使用 find 函数

举例说明 find( )函数的运算规则。

```
>> a=[0 1;2 3;0 4]
a =
    0    1
    2    3
    0    4
>> find(a)       %查找 a 中的非零元素
ans =            %结果显示元素的标识是按列进行的，即从 1 开始，
    2            %数完第 1 列再数第 2 列，依次数下去
    4
    5
    6
>> b(:,:,1)=[0 0;1 2]
b =
    0    0
```

```
    1    2
>> b(:,:,2)=[3 4;5 6]
b(:,:,1) =
    0    0
    1    2
b(:,:,2) =
    3    4
    5    6
>> find(b)
ans =
    2
    4
    5
    6
    7
    8
```

从以上不难看出，多维数组的元素标识是从低维到高维依次进位的，如对于一个三维数组 b，它的元素 b(1,1,1)，b(2,1,1)，b(1,2,1)，b(2,2,1)，b(1,1,2)，b(2,1,2)，b(1,2,2)，b(2,2,2)的标识依次为 1，2，3，4，5，6，7，8。

[i,j]=find(A) 此函数返回矩阵 A 的非零元素的行和列的标识，其中 i 代表行标，j 代表列标。此函数经常用在稀疏矩阵中。在多维矩阵中通常将第一维用 i 表示，将其余各维作为第二维，用 j 表示。如对于上面的三维矩阵 b，有：

```
>> [i,j]=find(b)
i =
    2
    2
    1
    2
    1
    2
j =
    1
    2
    3
    3
    4
    4
```

[i,j,v]=find(A) 此函数返回矩阵 A 的非零元素的行和列的标识，其中 i 代表行标，j 代表列标，同时，将相应的非零元素的值放于列向量 v 中，即 i 和 j 的值与[i,j]=find(A)取值相同，只是增加了非零元素的值这一项，例如下面一段程序：

```
>> [i,j,v]=find(b)
i =
    2
    2
    1
    2
    1
    2
```

```
j =
    1
    2
    3
    3
    4
    4
v =
    1
    2
    3
    5
    4
    6
>> a=[-1 -2;-3 -4]
a =
   -1   -2
   -3   -4
>> find(a<-3)       %找出 a 中小于-3 的元素的位置
ans =
    4
>> find(a<-2)       %找出 a 中小于-2 的元素的位置
ans =
    2
    4
>> find(a==-1)       %找出 a 中等于-1 的元素的位置
ans =
    1
>> a(find(a==-4))=-5       %找出 a 中等于-4 的元素换为-5
a =
   -1   -2
   -3   -5
```

可以实现部分矩阵的替换。

例 2.19　矩阵替换

利用 find( )函数实现部分矩阵的替换。

```
>> b=[3 4;5 6]
b =
    3    4
    5    6
>> a(find(a==-3))=b(find(a==-3))     %将矩阵 a 中等于-3 的元素换成
a =                                  %矩阵 b 中相应位置上的元素
   -1   -2
    5   -5                           %原来的-3 被替换为 5
>> a(find(a==-5))=[]                 %将矩阵 a 中等于-5 的元素删除
a =
   -1    5   -2
```

在矩阵计算中通常可以采用这种方法进行矩阵删除，即将矩阵的某个或某行元素直接赋值为零。

如果要删除矩阵 b 中的第 2 行，可以这样实现：

```
>> b(2,:)=[]
b =
    3    4
```

上面“()”中的“:”为操作符，代表第 2 行的所有元素。

5. isfinite：确认矩阵元素是否为有限值

isfinite(A) 如果矩阵 A 中的元素为有限值，则此函数在返回矩阵的相应位置上输出 1，否则输出 0。有限值为具有确定值的数，而 NaN，+Inf，-Inf 等都被视为无限值。函数 isinf(A) 判断矩阵 A 的元素是否为无限值，用法与 isfinite 相同。

NaN 称为不确定值，通常由 0/0、Inf± Inf、Inf/Inf 及 NaN 与其他任何数进行运算得到；Inf 称为无穷大数，可以由任意非零实数除以零得到，而复数除以零会得到 NaN 数。

例 2.20　使用 isfinite 函数

isfinite( )函数的用法。

```
>> c=[1 2;3 4]
c =
    1    2
    3    4
>> isfinite(c)
ans =
    1    1
    1    1
>> d=[1 +inf;nan -inf]
d =
    1  Inf
  NaN  -Inf
>> isfinite(d)
ans =
    1    0
    0    0
>> isinf(d)
ans =
    0    1
    0    1
```

6. isempty：确认矩阵是否为空矩阵

不要把空矩阵、零矩阵及矩阵不存在等三个概念混淆，空矩阵说明矩阵存在，但是矩阵没有元素；零矩阵是指矩阵的所有元素都为零；矩阵不存在是指当前的工作空间中没有定义此矩阵变量。Isempty(A)可以判断一个存在的矩阵变量 A 是否为空矩阵，如果 A 矩阵为空矩阵则返回逻辑“真”，否则返回逻辑“假”，一个零矩阵至少有一维是零，如 0×0，0×5，0×3×3 等。零矩阵没有任何元素，可以用函数 size(A)来判断，如果其中有一维为

零，则 A 就是零矩阵。

例 2.21　使用 isempty 函数

isempty( )函数的用法。

```
>> a=[]
a =
    []
>> size(a)
ans =
    0    0
>> b=rand(3,3,3);
>> b(:,:,:)=[]
b =
  Empty array: 0-by-3-by-3
>> size(b)
ans =
    0    3    3
```

b 矩阵是一个 0× 3× 3 维矩阵，因而 a 是个零矩阵。

```
>> isempty(b)
ans =
    1
```

7. isequal：判断几个对象是否相等

isequal(A,B,C,…)　如果要判断的所有对象 A,B,C,…具有相同的类型、大小和内容，对于矩阵来说，就是所有矩阵的维数相同，而且矩阵元素的数值相同，如果满足这样的条件，此函数返回逻辑“真”，反之，只要有一个对象与其他对象不相同，就会返回逻辑“假”。

例 2.22　使用 isequal 函数

isequal()函数的用法。

```
>> a=[1 2]
a =
    1    2
>> b=[1 2]
b =
    1    2
>> c='hello world'
c =
hello world
>> d=[1;2]
d =
    1
    2
>> isequal(a,d)
ans =
    0
>> isequal(a,b)
```

```
ans =
    1
>> isequal(a,b,c)
ans =
    0
```

8. isnumeric：判断对象是否是数据

isnumeric(A)　如果 A 是数据矩阵，如稀疏矩、双精度矩阵、复数矩阵等，此函数返回逻辑“真”，反之，如果 A 是字符串、结构体矩阵等，则返回逻辑“假”。

例 2.23　使用 isnumeric( )函数

isnumeric( )函数的用法。

```
>> isnumeric(a)
ans =
    1
>> isnumeric(c)
ans =
    0
>> e=[1+2i 3+4i]
e =
  1.0000 + 2.0000i  3.0000 + 4.0000i
>> isnumeric(e)
ans =
    1
```

还有一些逻辑函数，也比较常用，为了保持完整性，将其部分列出，以供参考。

- issparse：判断是否为稀疏矩阵。
- ischar：判断是否为字符串。
- islogical：判断一个矩阵是否为逻辑类型。
- isfield：判断对象是否为某个结构体矩阵的域。
- isstruct：判断是否为结构体类型。
- ishandle：判断是否为图像句柄。

2.5　数值数据类型

数字作为 Matlab 中数学运算的最基本对象。在 Matlab 中，主要包括各种有无符号的整数型数据、双精度型和单精度型数据。本节主要介绍这些数值数据类型。

2.5.1　整数

在 Matlab 中，支持如表 2-2 所示的各种整数型数据类型，主要包括 8 位、16 位、32 位和 64 位的有符号和无符号类型的整数数据类型。由于在 Matlab 中默认的数据类型是双精度型的数据，因此，在定义整型数据变量时，需要指定变量的数据类型。

表 2-2　Matlab 中的整数类型

数据类型	说　明
uint8	8 位无符号整数，数值范围为 0~255(0~2^8-1)
int8	8 位有符号整数，数值范围为-128~127(-2^7~2^7-1)
uint16	16 位无符号整数，数值范围为 0~65535(0~2^{16}-1)
int16	16 位有符号整数，数值范围为-32768~32767(-2^{15}~2^{15}-1)
uint32	32 位无符号整数，数值范围为 0~4294967295(0~2^{32}-1)
int32	32 位有符号整数，数值范围为-2147483648~2147483647(-2^{31}~2^{31}-1)
uint64	64 位无符号整数，数值范围为 0~18446744073709551615(0~2^{64}-1)
int64	64 位无符号整数，数值范围为 -9223372036854775808~9223372036854775807(-2^{63}~2^{63}-1)

需要说明的是，表中定义的整数数据类型不同，但是这些类型的数据具有相同的性质。每种类型的数据都可以通过函数 intmax 和 intmin 来查询此种数据类型的上下限。

例 2.24　查询数据类型上下限

利用 intmax( )和 intmin( )函数查询数据类型的上下限。

```
>> a=int16(50)
a =
    50
>> intmin('int16')
ans =
 -32768
>> intmax('int16')
ans =
  32767
>> class(a)
ans =
int16
>> b=65
b =
   65
>> class(b)
ans =
double
```

以上代码中，显示出 Matlab 中的整型数据定义方法及其默认的数据类型，同时通过 intmin 和 intmax 来获取整型数据的上下限。此外，class 函数可以获取所定义变量的数据类型，例如下面的程序：

```
>> x1=int8(1:9)
x1 =
   1   2   3   4   5   6   7   8   9
>> x2=int8(randperm(9))
x2 =
   6   3   7   8   5   1   2   4   9
>> x1+x2
```

```
ans =
   7   5  10  12  10   7   9  12  18
>> x1-x2
ans =
  -5  -1  -4  -4   0   5   5   4   0
>> x1.*x2
ans =
   6   6  21  32  25   6  14  32  81
>> x1./x2
ans =
   0   1   0   1   1   6   4   2   1
```

上面的例子通过两种方式定义了两个 int8 类型的整型向量，这两个向量的元素之间进行加减乘除运算。其中，randperm(9)函数随机生成 1~9 之间的随机向量。在进行乘除运算时，在向量后加“.”表示两个向量之间进行元素间的乘除运算。需要注意的是，在进行除法运算时，Matlab 首先将向量中的整数元素作为双精度类型的数据进行运算，然后根据四舍五入的原则得到整型数据相除的结果。

在 Matlab 命令窗口中输入：

```
>> x1
x1 =
   1   2   3   4   5   6   7   8   9
>> class(x1)
ans =
int8
>> x2=cast(x1,'uint16')
x2 =
    1    2    3    4    5    6    7    8    9
>> x1+x2
Error using  +
Integers can only be combined with integers of the same class, or scalar doubles.
>> x2+5
ans =
    6    7    8    9   10   11   12   13   14
>> class(ans)
ans =
uint16
```

从以上可以看出，不同类型的整型数据之间不能进行数学运算。但是，Matlab 支持双精度标量和整型数据之间的数学运算，原因在于 Matlab 将双精度类型的标量数据转化成整型数据再进行计算，例如下面的程序：

```
>> x1=int8(randperm(9))
x1 =
   7   2   8   5   6   9   4   1   3
>> x1=x1+110
x1 =
  117  112  118  115  116  119  114  111  113
>> x2=cast(x1,'uint8')
x2 =
  117  112  118  115  116  119  114  111  113
>> x2-115
```

```
ans =
   2   0   3   0   1   4   0  0   0
```

在 Matlab 的整型数据中，每种类型的整型数据都存在一定的数值范围，因此数学运算过程中会产生结果溢出问题。当运算过程中产生溢出问题时，Matlab 采用饱和处理问题的方式处理，即将计算结果设定为溢出方向的上下限数值。在进行混合数据计算时，Matlab 仅支持双精度标量和一个整型数据之间进行计算。由于对整型数据之间的运算关系，Matlab 只支持同种类型的整型数据之间进行计算，因此，除 64 位的整型数据之外，整型数据的存储比双精度数据的存储速度要快得多。

2.5.2 浮点数

双精度类型(double)的数据是 Matlab 的默认数据类型。有时为节省存储空间，Matlab 也支持单精度类型(single)的数据。单精度和双精度类型的取值范围可以选择用函数 realmin、realmax 来得到。单精度类型浮点数的精度可以通过函数 eps 来得到。

此外，需要注意的是对于单精度的数据变量，创建方法和整型数据的创建方法相同。而对于单精度数据和双精度数据之间的混合运算，处理结果为单精度的数据结果。

例 2.25 浮点数运算举例

浮点数运算举例。

```
>> realmin('single')
ans =
  1.1755e-038
>> realmax('single')
ans =
  3.4028e+038
>> realmin('double')
ans =
  2.2251e-308
>> realmax('double')
ans =
  1.7977e+308
>> eps
ans =
  2.2204e-016
>> x1=single(1:9)
x1 =
    1    2    3    4    5    6   7    8    9
>> x2=ones(1,4,'single')
x2 =
    1    1    1    1
>> class(x1)
ans =
single
>> class(x2)
ans =
single
>> x3=rand(1,9)
```

```
x3 =
   0.7922  0.9595  0.6557  0.0357  0.8491  0.9340  0.6787  0.7577 0.7431
>> class(x3)
ans =
double
>> x4=x1+x3
x4 =
  Columns 1 through 3
    1.7922   2.9595   3.6557
  Columns 4 through 6
    4.0357   5.8491   6.9340
  Columns 7 through 9
    7.6787   8.7577   9.7431
>> class(x4)
ans =
single
```

2.5.3　整型浮点数间的操作函数

Matlab 中提供了大量的数据操作函数，可以进行不同数据类型变量的创建或数据之间的转化处理。表 2-3 所示为其中的一部分数据操作函数，主要实现整型变量或数据与双精度类型变量或数据之间的转换操作。

表 2-3　Matlab 中常见的数值数据函数

函数名称	描　述
double	创建或转化为双精度数据
single	创建或转化为单精度数据
int8,int16,int32,int64	创建或转化为有符号的整型数据
uint8,uint16,uint32,uint64	创建或转化为无符号的整型数据
isnumeric	数据类型判断函数，如果为整型数据或浮点数，那么函数返回 true
isinteger	整型数据判断函数，如果为整型数据，则返回 true
isfloat	浮点数类型判断函数，如果为单精度或双精度数据，则返回 true
isa(x,‘type’)	判断 x 是否为指定的 type 类型数据，若是则返回 true
cast(x,‘type’)	将 x 的数据类型转化为 type 类型的数据
intmax(‘type’)	返回整型数据的最大数值
intmin(‘type’)	返回整型数据的最小数值
realmax(‘type’)	返回浮点数的最大数值
realmin(‘type’)	返回浮点数的最小数值
eps(‘type’)	返回 type 类型数据的 eps 数值(浮点数值，即精度)
eps(‘x’)	x 的 eps 数值

表中，type 为 numeric、integer、float 及其他数据类型。

2.5.4 复数

Matlab 的一个比较强大的功能是能够直接在复数域上进行运算，而不用进行任何特殊的操作。而有些编程语言。在定义复数的时候，需要进行特殊的处理。在 Matlab 中，复数的书写方法和运算表达形式与数学中复数的书写方法和运算表达形式相同，复数单位可以通过 i 或 j 来表达。

在 Matlab 中，可以采用提供的命令来进行复数的极坐标形式和直角坐标形式之间的转化。通过欧拉恒等式，可以将复数的极坐标形式和直角坐标形式联系起来，即可以表示为 $M\angle\theta = Me^{i\theta} = a + bi = z$。在 Matlab 中，利用系统所提供的内置转换命令，可以很方便地得到复数的一些基本数值。例如：

real(z)　　计算复数的实部 z=rcosθ

imag(z)　　计算复数的虚部 z=rsinθ

abs(z)　　计算复数的模 $\sqrt{a^2+b^2}$

angle(z)　　以弧度为单位给出复数的幅角 $\arctan\frac{b}{a}$

例 2.26　复数的基本运算

复数基本运算举例。

```
>> z1=2+3i        %复数一般表达式
z1 =
   2.0000 + 3.0000i
>> z2=3*exp(i*pi/4)     %极坐标形式表达复数
z2 =
   2.1213 + 2.1213i
>> z3=complex(1,2)     %通过函数定义复数
z3 =
   1.0000 + 2.0000i
>> z4=sqrt(-4)         %运算得到复数
z4 =
   0 + 2.0000i
>> z5=6+sin(1)i
 z5=6+sin(1)i
Error: Unexpected Matlab expression.
>> z5=6+sin(1)*i         %符号函数定义
z5 =
   6.0000 + 0.8415i
>> z=z1*z2/z3
z =
   3.8184 + 2.9698i
>> z=3+4i
z =
   3.0000 + 4.0000i
>> real_z=real(z)      %实部
real_z =
    3
>> imag_z=imag(z)     %虚部
```

```
imag_z =
    4
>> mag_z=abs(z)      %模
mag_z =
    5
>> angle_z=angle(z)     %幅角(弧度)
angle_z =
   0.9273
>> angle_z=angle(z)*180/pi      %幅角(角度)
angle_z =
  53.1301
```

例 2.27　利用 Matlab 求解复数的根

利用 Matlab 求解复数的根，并将这些根用图形表达出来(图 2-1)。

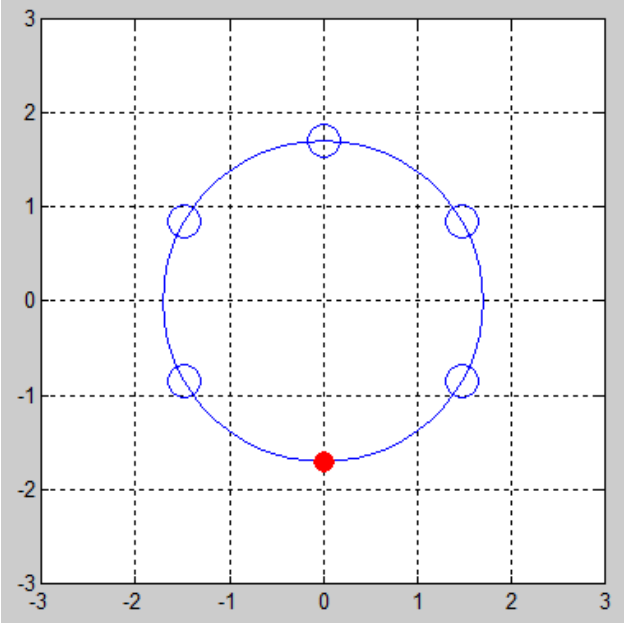

图 2-1　求解复数的根

```
>> res=(-15)^(1/3)      %直接求解复数的根
res =
   1.2331 + 2.1358i
p=[1,0,0,0,0,0,25];       %构造多项式求解所有根
>> r=roots(p)
r =
  -1.4809 + 0.8550i
  -1.4809 - 0.8550i
  -0.0000 + 1.7100i
  -0.0000 - 1.7100i
   1.4809 + 0.8550i
   1.4809 - 0.8550i
>> t=0:pi/30:2*pi;
>> x=1.71*sin(t);
>> y=1.71*cos(t);
```

```
>> plot(x,y,'b'),grid on
>> plot(x,y,'b'),grid on
>> hold on
>> plot(r(4),'.','MarkerSize',30,'color','r')
>> plot(r([1,2,3,5,6]),'o','MarkerSize',15,'color','b')
>> axis([-3,3,-3,3]),axis square
>> hold off
```

为得到-25 的 6 次方根，通过构造多项式函数得到了所有根。最后，通过使用 Matlab 提供的绘图函数表示了得到的 5 个根。

2.6 函数和特殊函数简明介绍

Matlab 主要进行数学计算，因而各种数学函数在以后的计算中都是必不可少的，这些数学函数和大多数的数学函数的书写形式相同。需要注意的是，在利用这些数学函数求解时，角度都用弧度来表示，表 2-4 至表 2-6 所示为常用的一些数学函数。

表 2-4 三角函数

sin	正弦函数	sinh	双曲正弦函数
asin	反正弦函数	asinh	反双曲正弦函数
cos	余弦函数	cosh	双曲余弦函数
acos	反余弦函数	acosh	反双曲余弦函数
tan	正切函数	tanh	双曲正切函数
atan	反正切函数	atanh	反双曲正切函数
sec	正割函数	sech	双曲正割函数
asec	反正割函数	asech	反双曲正割函数
cot	余切函数	costh	双曲余切函数
acot	反余切函数	acoth	反双曲余切函数

表 2-5 其他常用计算函数

fix	朝零方向取整	roud	四舍五入到最近的整数
floor	朝负无穷方向取整	rem	求两整数相除的余数
ceil	朝正无穷方向取整	exp	求指数函数
log	自然对数	log10	求以 10 为底的函数
sqrt	求数值的平方根	abs	求绝对值
conj	求复数的共轭	imag	求复数的虚部
real	求复数的实部		

表 2-6　常用矩阵函数

sqrtm	求矩阵的平方根	expm	求矩阵的指数值
funm	求按矩阵计算的函数值	logm	求矩阵的对数值

2.7　课后练习

1. 在 Matlab 中如何建立矩阵 $\begin{bmatrix}5 & 7 & 3\\4 & 9 & 1\end{bmatrix}$，并将其赋予变量 a?
2. 有几种建立矩阵的方法？各有什么优点？
3. 在进行算术运算时，数组运算和矩阵运算各有什么要求？
4. 数组运算和矩阵运算的运算符有什么区别？
5. 计算矩阵 $\begin{bmatrix}5 & 3 & 5\\3 & 7 & 4\\7 & 9 & 8\end{bmatrix}$ 与 $\begin{bmatrix}2 & 4 & 2\\6 & 7 & 9\\8 & 3 & 6\end{bmatrix}$ 之和。
6. 求 $x=\begin{bmatrix}4+8i & 3+5i & 2-7i & 1+4i & 7-5i\\3+2i & 7-6i & 9+4i & 3-9i & 4+4i\end{bmatrix}$ 的共轭转置。
7. 计算 $a=\begin{bmatrix}6 & 9 & 3\\2 & 7 & 5\end{bmatrix}$ 与 $b=\begin{bmatrix}2 & 4 & 1\\4 & 6 & 8\end{bmatrix}$ 的数组乘积。
8. “左除”与“右除”有什么区别？
9. 对于 $AX=B$，如果 $A=\begin{bmatrix}4 & 9 & 2\\7 & 6 & 4\\3 & 5 & 7\end{bmatrix}$，$B=\begin{bmatrix}37\\26\\28\end{bmatrix}$，求解 X。