第二章　数据结构及其运算

2.1　数据类型

MATLAB共有六种基本数据类型，即双精度型（double）、字符型（char）、稀疏型（sparse）、存储型（storage）、细胞型（cell）和结构型（struct），每一种类型可以构成一维、二维和多维的数组。

MATLAB计算都采用双精度，绝大部分函数都是对双精度矩阵和字符串操作的，其他几种数据类型用于特殊的场合。比如存储型可用于图像处理，稀疏型用于稀疏矩阵，细胞型和结构型一般用于编写大型软件。表2.1.1列出了这些数据类型的一些例子。

表2.1.1 数据类型举例

数据类型（class）	举　　例	解　　释
double	[1,2;3,4],5+6i	双精度数值类型，是最常用的类型。
char	‘Hello’	字符数组，每个字符占16位。
sparse	speye(5)	双精度稀疏矩阵，只存储矩阵中的非0元素
cell	{[1,2,3;4,5,6;7,8,9],‘hello’,eye(2)}	细胞数组，数组中的每个元素可为不同类型、不同维的数据。
struct	a.day=12; a.color=’red’ a.mat=magic(3)	结构数组相当于数据库的记录，把相关的数据列在一起，称为属性，不同属性的数据类型可以不同。
storage	unit8(magic(3))	8位型，为无符号整数，最大可表示255,不能进行数学运算。

数组（Array）是由一组数据排列成的长方阵列，可以是一维的行（或列），也可以是二维的矩阵，还可以是多维的。用户可以操作整个数组，也可以操作数组中的某个或者某些元素。

在MATLAB里，不需要用double、char等关键字来定义变量，会根据表达式的运算结果，自动确定变量的类型和大小。变量的数据类型可以用以下函数来查看：

isa（var，’type’）　　%变量var的数据类型名称如果是type，则返回1，否则返回0

class(var)　　%返回变量a的数据类型名称

whos var　　%查看变量var的详细情况

2.2　一维数组

2.1.1 一维数组的创建

1．逐个元素输入法

```
>> a=[1.0, 2+2.4i, 3*pi]                %逐个输入数组中的元素值
a =
        1.0000        2.0000 + 2.4000i      9.4248
```

2．冒号运算符法

```
>> b= 1:1:5                             %用冒号运算符循环产生数组元素
b =
     1     2     3     4     5
```

【说明】冒号运算符的格式是 startv : step : endv。startv是初值，即数组的第一个元素值；endv是终值，即数组的最后一个元素值；step称为步长，即数组元素每次增加的值；step为1时可以省略不写，step可以为负值，此时要求startv > endv。

3．线性分隔法

```
>> linspace(1,5,9)
ans =
Columns 1 through 8
  1.0000   1.5000   2.0000   2.5000   3.0000   3.5000   4.0000 4.5000
Column 9
  5.0000
```

【说明】

1. linspace函数的调用格式为：**x=linspace(a, b, n)。**
2. 数组的第一个元素值为a，最后一个元素值为b，数组中共有n个元素，这n个元素线性均匀分布于a和b之间，即数组元素依次为 $a+\frac{b-a}{n-1}*i,\quad i=0,1,\cdots,n-1$。

4．对数分隔法

```
>> logspace(0,3,4)
ans =
        1          10         100        1000
```

【说明】

1. logspace函数的调用格式为：**x=logspace(a, b, n)。**
2. 数组的第一个元素值为10^a，最后一个元素值为10^b，数组中共有n个元素，这n个元素的以10为底的对数值均匀分布与a和b之间，即数组元素依次为

$$10^{a+\frac{b-a}{n-1}*i},\quad i=0,1,\cdots,n\text{-}1$$

2.1.2 一维数组的访问

一维数组的访问遵循以下约定：

1. 用下标方式访问数组元素，下标要用一对圆括号( )引起来。
2. 下标代表的是元素在数组中的位置序号，从1开始，最大值为数组中元素的个数。
3. 下标可以是常量，也可以是变量。
4. 可以访问数组中的单个元素，也可以访问数组中的某些元素，即数组的子数组。

例2.2.1 数组的访问。

```
>> rand('state',0)                  %把均匀分布的伪随机发生器置为0状态
>> x=rand(1,5)                      %产生1×5的均匀分布随机数组
x =
    0.9501    0.2311    0.6068    0.4860    0.8913
>> x(1)                             %读数组的第1个元素
ans =
    0.9501
>> x([1,4])                         %返回数组x的第1个和第4个元素构成的子数组
ans =
    0.9501    0.4860
>> x(1:3)                           %返回数组的前3个元素构成的子数组
ans =
```

```
    0.9501    0.2311    0.6068
>> x(2:end)                %第2个元素与其后所有元素构成的子数组，end代表一维数组的最大下标
ans =
    0.2311    0.6068    0.4860    0.8913
>> x(1)=0                  %数组的第1个元素赋值0
x =
         0    0.2311    0.6068    0.4860    0.8913
>> x(2:4)=1                %数组的第2到第4个元素赋值1
x =
         0    1.0000    1.0000    1.0000    0.8913
>> x([1,2])=[2,3]          %数组的第1个元素和第2个元素分别赋值2和3
x =
    2.0000    3.0000    1.0000    1.0000    0.8913
>> x(end:-1:1)             %获得一维数组的倒序数组
ans =
    0.8913    1.0000    1.0000    3.0000    2.0000
```

2.3 二维数组

2.3.1 二维数组的建立

1．逐个输入数组元素值

如果数组内元素数量少，可以直接从键盘逐个输入元素的值，需要遵循的规则是：

(1) 整个数组必须用中括号“[ ]”括起来；

(2) 数组的行与行之间用分号“；”分隔，或者用回车符分隔；

(3) 每行之间的元素必须用逗号“，”或者空格分隔；

(4) 分隔符必须是英文字符，即在英文状态下输入分号、括号、方括号、逗号等。

例2.3.1 二维数组的直接输入。

```
>> x=[1,2,3; 4,5,6; 2*pi,3+i,0]
x =
   1.0000             2.0000             3.0000
   4.0000             5.0000             6.0000
   6.2832             3.0000 + 1.0000i        0
```

2．利用M文件

如果数组元素很多，或者元素值要经常改变，我们可以采用M文件来输入和保存数组。

例2.3.2 用M文件实现对数组x的输入和保存。

（1）在当前目录下，用程序编辑器建立一个名为MyData.m的文件

（2）在编辑器中输入如下内容

```
x=[1,2,3,4,5,6,7,8,9,10;
   11,12,13,14,15,16,17,18,19,20;
   21,22,23,24,25,26,27,28,29,30];
```

(3)保存MyData.m文件

(4)在命令窗口键入MyData，就可以在内存中建立数组x并读入数组元素的值。

2.3.2 二维数组的访问

二维数组的访问遵循以下约定：

1. 用下标方式访问数组元素，下标要用一对圆形括号()引起来。
2. 可以用双下标方式访问数组元素，格式为(r,c)，其中r为二维数组的行下标，c为二维数组的列下标，下标之间用逗号分隔。
3. 可以用单下标方式访问二维数组，二维数组的单下标是按照列优先规则排序的，即二维数组被看作是从第一列开始从左到右依次将各列首末位连接而成的一维数组，单下标表示元素在这个一维数组中的位置。
4. 单下标和双下标具有对应关系，其值可以通过ind2sub( )和sub2ind( )函数进行转换。
5. 可以访问二维数组的某个元素及其子数组，可以对元素和子数组赋值。

例2.3.2 二维数组的访问。

```
>> A=zeros(2,4)                 %生成2行4列的全0数组
A =
     0     0     0     0
     0     0     0     0
>> A(2,4)=4                     %第2行第4列元素赋值4
A =
     0     0     0     0
     0     0     0     4
>> ind=sub2ind(size(A),2,4)     %将A的第2行第4列的双下标值转换为单下标值
ind =
     8
>> A(8)=16                      %将单下标为8的元素赋值为16，相当于A(2,4)=16
A =
     0     0     0     0
     0     0     0    16
>> A(1:4)=1:4                   %将单下标为1到4的元素分别赋值为1，2，3，4
A =
     1     3     0     0
     2     4     0    16
>> A([1,3,5,7])=5               %将单下标为1，3，5，7的元素都赋值为5
A =
     5     5     5     5
     2     4     0    16
>> A(:,2)=0                     %将第2列所有元素赋值为0
A =
     5     0     5     5
     2     0     0    16
>> A(:,:)=0                     %所有元素赋值为0，双下标方式
A =
     0     0     0     0
     0     0     0     0
>> A(:,[2,4])=ones(2,2)         %第2列和第4列赋值为相应2×2全1数组
A =
```

```
     0     1     0     1
     0     1     0     1
>> A(:,1)=[]                    %通过给第1列元素赋值空数组实现删除数组的第1列
A =
     1     0     1
     1     0     1
>> A(1:3)=[]                    %单下标为1到3的元素，此时数组变为一维数组
A =
     0     1     1
```

【说明】有关空数组

1. 某一维长度为0的数组称为空数组。
2. 空数组用[]表示，表示数组中没有元素，但可以表示计算结果为“空”。
3. 仅仅能用isempty( )函数正确判断数组是否为空。
4. 可以通过给数组元素赋值空数组来缩小数组的大小。
5. 尽量不要用空数组参与逻辑运算和关系运算。

例2.3.3 空数组的用法。

```
>> a=[];                        %定义了空数组变量a
a =
     []
>> b=ones(3,0)                  %定义了二维3×0的空数组变量b
b =
   Empty matrix: 3-by-0
>> isempty(b)                   %判断数组b是否为空数组
ans =
     1
>> x=[1,2,3; 4,5,6; 7,8,9];     %定义了3×3数组x
>> x(3,:)=[]                    %删除数组x的第3行
x =
     1     2     3
     4     5     6
```

2.4 高维数组

2.4.1 高维数组的创建

可以采用下列方法创建高维数组：

1. 直接通过全下标方式进行元素赋值。
2. 用低维数组合成高维数组。
3. 用数组生成函数（ones/zeros/rand等）生成高维数组。
4. 用数组操作函数（repmat/reshape等）构造高维数组。

例2.4.1 高维数组的创建

```
>> A(2,2,3)=10                  %创建三维（2×2×3）数组
A(:,:,1) =
     0     0
     0     0
```

```
A(:,:,2) =
     0     0
     0     0
A(:,:,3) =
     0     0
     0    10
>> A1=ones(2,3)                %用ones创建2×3的全1数组
A1 =
     1     1     1
     1     1     1
>> A2=eye(2,3)                 %用eye创建2×3的单位数组
A2 =
     1     0     0
     0     1     0
>> A3=rand(2,3)                %用rand创建2×3的随机数组
A3 =
    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621
>> B(:,:,1)=A1; B(:,:,2)=A2; B(:,:,3)=A3;   %用3个2×3数组合成2×3×3数组B
>> B
B(:,:,1) =
     1     1     1
     1     1     1
B(:,:,2) =
     1     0     0
     0     1     0
B(:,:,3) =
    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621
```

2.4.2 多维数组的访问

对于高维数组的访问，有下列约定：

1. 可以通过全下标方式访问。对于三维数组来说，第一维下标称为“行下标”，第二维下标称为“列下标”，第三维下标一般称为“页下标。”
2. 可以通过单下标方式访问。高维数组的单下标是按照后维优先的次序排列的，对于三维数组来说，先排列“页”，页内先排列“列”，列内再排列“行”，即第1行第1列第1页的元素单下标为1，然后先变化行下标，再变化列下标，最后变化页下标。
3. 数组的维数通过ndims函数获取。
4. 数组的尺寸通过size函数获取
5. 数组的所有维中的最大长度通过length函数获取。

例2.4.2 高维数组的访问。

```
>> A=ones(2,4,3);          %创建2×4×3的三维数组
>> size(A)                 %获取数组A的大小
ans =
     2     4     3
```

```
>> length(A)              %获取数组A的最长维数的长度
ans =
     4
>> ndims(A)               %获取数组A的维数
ans =
     3
>> A(:)=1:2*4*3           %用单下标方式给数组A的所有元素赋值
A(:,:,1) =
     1     3     5     7
     2     4     6     8
A(:,:,2) =
     9    11    13    15
    10    12    14    16
A(:,:,3) =
    17    19    21    23
    18    20    22    24
>> A(1:8)                 %用单下标方式访问数组A的前8个元素
ans =
     1     2     3     4     5     6     7     8
```

2.5 数组操作

2.5.1 标准数组的生成

数学中定义了很多标准数组或者矩阵，如全1数组、全零数组、对角阵等，在MATLAB中有相应的函数用来生成这些标准数组。

1. ones

【功能】：生成全1数组，即数组中的元素都为1。

【调用格式】

Y=ones(n)	生成n×n的全1矩阵
Y=ones(m1,m2,…,mk)	生成m1×m2×…×mk的全1数组
Y=ones(size(A))	生成和数组A同样尺寸的全1数组

2. zeros

【功能】：生成全0数组，即数组中的元素都为0。

【调用格式】：同ones函数

3. rand

【功能】：生成均匀分布随机数组。

【调用格式】

Y=rand('state',v)	设置随机发生器的初始状态为v，其他同ones函数

4. randn

【功能】：产生正态分布随机数组。

【调用格式】：同rand函数

5. magic

【功能】：产生魔方矩阵,不适用于高维数组。

【调用格式】

M=magic(n)　　产生n×n的魔方矩阵

6. eye

【功能】：产生单位矩阵，即主对角线元素都为1而其他元素都为0的二维数组

【调用格式】

Y=eye(n)

Y=eye(n,m)

Y=eys(size(A))

Y=eye(m,n,classname)

【说明】：classname是字符串，表示元素的数据类型名称，可取'double', 'single', 'int8', 'uint8', 'int16', 'uint16', 'int32', 'uint32', 'int64', 'uint64'。

7. diag

【功能】：产生对角阵，即矩阵的某个对角线元素不全为0，其他元素为0。

【调用格式】

X = diag(v,k)　　生成length(v)+k阶方阵，并在第k条对角线放置元素v

X = diag(v)　　生成length(v)阶方阵，并在主对角线放置元素v

v = diag(X,k)　　返回方阵X的第k条对角线元素构成的列向量

v = diag(X)　　返回方阵X的主对角线元素构成的列向量

【说明】：v为行向量，表示对角线元素；k为对角线位置，k=0表示主对角线，k>0表示在主对角线上方的第k条对角线，k<0表示在主对角线下方的第(-k)条对角线。

例2.5.1 生成特殊矩阵。

```
>> ones(2);                %生成2×2的全1矩阵
>> zeros(2,3);             %生成2×3的全0矩阵
>> A=eye(2,3);             %生成2×3的单位矩阵
>> rand(size(A));          %生成和A相同大小的随机矩阵
>> diag([1,2,3],2);        %生成5×5矩阵，在主对角线上方第2条对角线放置1，2，3
>> B=[1,2,3;4,5,6;7,8,9];
>> diag(B)                 %获取矩阵B的主对角线元素
ans =
     1
     5
     9
>> diag(B,-1)              %获取矩阵B主对角线下方的第1条对角线的元素
ans =
     4
     8
```

2.5.2 数组操作

对数组的操作包括数组的扩展、收缩、重排、元素交换和子数组访问等。数组操作可以通过两种方式实现，一种是通过MATLAB提供的运算符（逗号，分号，括号等）来实现，另外一种是使用MATLAB提供的数组操作函数。下面介绍常用的数组操作函数。

1. cat

【功能】：把大小相同的若干数组，沿着指定维的方向，串接成新数组。

【调用格式】

C = cat(dim, A, B)

C = cat(dim, A1, A2, A3, A4, ...)

【说明】：A，B，A1，A2等为被串接数组，要求这些数组同维；dim表示串接方向，1表示行，2表示列，3表示页，以此类推。

2. fliplr

【功能】：沿着垂直中线，左右（Left-Right）对称交换数组元素（不超过2维）

【调用格式】

```
B = fliplr(A)
```

3. flipud

【功能】：沿着水平中线，上下（Up-Down）对称交换数组元素（不超过2维）

【调用格式】

```
B = flipud(A)
```

4. rot90

【功能】：逆时针旋转二维数组。

【调用格式】

```
B = rot90(A)          逆时针旋转矩阵90度
B = rot90(A,k)        逆时针旋转矩阵90*k度
```

5. repmat

【功能】：按指定维上的数目，分块铺放指定数组。

【调用格式】

```
B = repmat(A,m,n)             沿着第1维铺放m个A，第2维铺放n个A
B = repmat(A,[m n])
B = repmat(A,[m n p...])
```

6. reshape

【功能】：在总元素不变的前提下，重新安排数组各个维的长度，形成新数组。

【调用格式】

```
B = reshape(A,m,n)
B = reshape(A,m,n,p,...)
B = reshape(A,[m n p ...])
B = reshape(A,...,[],...)
```

【说明】：A是待重新安排的数组；m，n，p等是新数组各个维的长度；[]表示自动计算某个维的长度而无需用户指定。

7. tril

【功能】：提取矩阵的下三角元素，生成下三角阵。

【调用格式】

```
L = tril(X)
L = tril(X,k)
```

【说明】：X为待提取的矩阵；k为三角阵的分界线位置，含义同diag函数。

8. triu

【功能】：提取矩阵的上三角元素，生成上三角阵。

【调用格式】

```
L = triu(X)
L = triu(X,k)
```

【说明】：X为待提取的矩阵；k为三角阵的分界线位置，含义同diag函数。

例2.5.2 用MATLAB运算符实现数组操作。

```
>> A=[1,2;3,4]; B=[5,6;7,8];
>> C=[A,B]                    %矩阵A和B横向组合
C =
```

```
     1     2     5     6
     3     4     7     8
>> D=[A;B]                    %矩阵A和B纵向组合
D =
     1     2
     3     4
     5     6
     7     8
>> E=[A,B+100;[0,0,0,0]]     %矩阵组合
E =
     1     2   105   106
     3     4   107   108
     0     0     0     0
>> A(:,3)=[5,6]'              %通过赋值来扩展矩阵，矩阵第3列赋值5，6
A =
     1     2     5
     3     4     6
>> AA=A(:,[1:2,2,1])         %重复使用矩阵元素，扩展矩阵
AA =
     1     2     2     1
     3     4     4     3
```

例2.5.3 用数组操作函数实现数组操作。

```
>> A=1:12;B=reshape(A,2,6) %将A重新构造成2×6的数组
B =
     1     3     5     7     9    11
     2     4     6     8    10    12
>> A=[1 2; 3 4]; B=[0 0;0 0];
>> cat(2,A,B)                %将矩阵A和B按第2维串接（列方向）
ans =
     1     2     0     0
     3     4     0     0
>> fliplr(A)                 %左右交换矩阵的元素
ans =
     2     1
     4     3
>> flipud(A)                 %上下交换矩阵的元素
ans =
     3     4
     1     2
>> s=[1,2,3,4,5];fliplr(s) %对于行向量来说，左右交换元素相当与向量元素逆序
ans =  5     4     3     2     1
>> flipud(s)                 %对与行向量来说，上下交换元素意味着维持原向量
ans =  1     2     3     4     5
>> rot90(A)                  %逆时针旋转A矩阵90度
ans =
     2     4
```

```
     1     3
>> C=repmat(A,2,2)          %将数组A沿着第1维方向铺2块，第2维方向铺2块
C =
     1     2     1     2
     3     4     3     4
     1     2     1     2
     3     4     3     4
>> tril(C)                  %以主对角线为分界线，生成C的下三角矩阵
ans =
     1     0     0     0
     3     4     0     0
     1     2     1     0
     3     4     3     4
```

2.6 数组运算与矩阵运算

2.6.1 数组运算

MATLAB 定义了数组运算，数组运算是指对数组中的每个元素进行相同的运算。数组运算可以通过 MATLAB 提供的运算符和数组运算函数实现。

1. 用数组运算符进行数组运算

表 2.6.1 常用数组运算符表

数组运算	功能	说明
A+B	数组加法运算	
A-B	数组减法运算	
A.*B	数组相乘	A 和 B 相同位置元素的乘积作为结果数组的元素
A./B	数组相除	A 和 B 相同位置元素相除作为结果数组的元素
A.\B	与 A./B 相同	
A.^p	数组各元素求 p 次幂	
A#B	A、B 数组对应元素间进行关系运算	#代表关系运算符
A@B	A、B 数组对应元素间进行逻辑运算	@代表逻辑运算符
A.'	数组转置	非共轭转置
s◎A	标量 s 与数组 A 运算	s 与 A 的每个元素进行运算，◎代表某个运算符

2. 数组运算函数

表 2.6.2 常用数组运算函数表

数组函数	运算函数	函数说明
三角函数	sin，cos，asin，asinh，asec，sect，tan，atan 等	三角函数
指数对数函数	exp	指数函数
	log	自然对数函数
	log10	以 10 为底的对数函数
	log2	以 2 为底的对数函数

	pow2	2 的幂函数
	sqrt	平方根函数
复数函数	abs	绝对值，模
	angle	相角（弧度）
	imag	复数的虚部
	real	复数的实部
	conj	复数的共轭
取整函数	ceil	向+∞方向取整
	fix	向 0 方向取整
	floor	向-∞方向取整
	round	向最近的整数取整
符号函数	sign	操作数为正返回 1，为负返回-1，为零返回 0

2.6.2 矩阵运算

矩阵和二维数组在数据结构上是完全相同的，但是矩阵是一种数学变换或者数学算子，矩阵的运算在数学上有严格的运算规则定义，和数组运算是不同的。矩阵运算可以通过 MATLAB 运算符实现，也提供矩阵函数来支持矩阵运算。

表 2.6.3 常用矩阵运算表

矩阵运算	功能	说明
A+B	矩阵加法	
A-B	矩阵减法	
A*B	矩阵乘法	要满足维数要求，一般不符合交换律
A/B	矩阵右除	右除是先计算矩阵的逆再相乘，要快一点
A\B	矩阵左除	不需要计算逆矩阵直接进行除运算，可避免被除矩阵的奇异性所带来的麻烦
A^p	矩阵乘方	
s*A	标量 s 与矩阵 A 相乘	标量 s 分别与 A 的每个元素相乘
expm(A)	矩阵的指数函数	
logm(A)	矩阵的对数函数	
sqrtm(A)	矩阵的平方根函数	
inv(A)	矩阵的逆矩阵	

【说明】

1. 点运算。MATLAB 运算符提供了点运算功能。在常用的算数运算符前面加上一个“.”则代表运算是按照数组运算规则进行运算，否则是按照矩阵运算规则进行运算的。

2. MATLAB 中有些运算函数的名字是某个函数名字后加了一个字母 m，通常情况下，这两个函数的运算功能是相同的，只是加了 m 的函数按照矩阵运算规则运算，另外一个函数按照数组运算规则运算。

例 2.6.1 矩阵运算和数组运算。

```
>> A=[1,4;9,16];
>> Aq=sqrt(A)                %数组的平方根
Aq =
     1     2
     3     4
>> Aqm=sqrtm(A)              %矩阵的平方根
Aqm =
```

```
    0.4662 + 0.9359i   0.8860 - 0.2189i
    1.9935 - 0.4924i   3.7888 + 0.1152i
>> Aq*Aq                        %矩阵乘法
ans =
     7    10
    15    22
>> Aq.*Aq                       %数组乘法
ans =
     1     4
     9    16
>> B=eye(2);
>> exp(B)                       %数组指数
ans =
    2.7183    1.0000
    1.0000    2.7183
>> expm(A)                      %矩阵指数
ans =
  1.0e+007 *
    0.7988    1.5181
    3.4158    6.4918
```

2.6.3 特殊的运算结果

1.正无穷大 Inf

有些数值计算结果是趋向于无穷大（Infinity），MATLAB 中用 Inf 表示正无穷大。下面的表达式会产生无穷大的运算结果。

1/0, 1.e1000, 2^2000, exp(1000), 2^100/realmin

2.非数 NaN

有些计算结果会产生非数（Not a Number），MATLAB 中用 NaN 来表示非数。下面的表达式会产生非数的计算结果。

0/0, 0*inf,inf/inf

【说明】MATLAB 给出了 Inf 和 NaN 的目的是为了避免因为运算结果出现无穷大或者不可预知的非数情况而造成程序执行的中断。

例 2.6.2 无穷大和非数。

```
>> A=[0 1 5;2 NaN inf]; B=[0 0 15;2 5 inf];
>> C=A./B
Warning: Divide by zero.
C =
       NaN       Inf    0.3333
    1.0000       NaN       NaN
```

2.7 多项式

在MATLAB里多项式用其系数行向量表示，如多项式P（x）= $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$。表示为P=[$a_n$,a_{n-1}, …,a_1, a_0]。

2.7.1 多项式的创建

1. 直接输入法

例如： P=[1,2,3,4]

2. poly(A)

若 A 为方阵，则创建方阵 A 的特征多项式。

3. poly(a)

如果向量a=[b_n b_{n-1}…b_1 b_0]，则创建 (x-b_0) (x-b_1) … (x-b_{n-1})(x- b_n)生成的多项式的系数向量，即创建全部根为b_n , b_{n-1}, … , b_1 , b_0 对应的多项式。

2.7.2 多项式的运算

数学上对多项式的运算有严格的定义，多项式运算包括多项式相乘、多项式相除、多项式相加减、多项式求导、多项式求值、多项式的部分分式展开和多项式求根等。

1． 多项式加减：+ -

【说明】：使用 MATLAB 的运算符 + - 实现多项式的加减法。

2． 多项式相乘: p=conv(p1,p2)

【说明】：p 为多项式 p1 和多项式 p2 的乘积。

3． 多项式相除：[q,r]=deconv(p1,p2)

【说明】：多项式 p1 除以多项式 p2，商多项式为 q，余多项式为 r。

4． 多项式求导

dp=polyder(p)	多项式 p 的导数多项式为 dp
dp=polyder(p1,p2)	多项式 p1 和多项式 p2 乘积的导数多项式为 dp
[num,den]=polyder(p1,p2)	有理分式(p1/p2)的求导后的有理分式为(num/den)

5． 多项式求值

pA=polyval(p,A)	按数组运算规则求多项式 p 在自变量为 A 时的值
MA=ployvalm(p,A)	按矩阵运算规则求多项式 p 在自变量为 A 时的值

【说明】：当自变量A为矩阵时，多项式中的常数项a_0被当作a_0*eye(n)处理

6． 多项式求根：r=roots(p)

【说明】：r 为多项式 p 的根组成的列向量

7． 部分分式分解：[r,p,k]=residue(num,den)

【说明】：有理分式(num/den)部分分式分解后，r 为留数，p 为极点，k 为直项。

8． 多项式拟合：p=polyfit(x,y,n)

【说明】：由给定数据 x 和 y 拟合出 n 阶多项式 p，用来逼近 y=f(x)曲线。

9． 矩阵的特征多项式：p=poly(A)

【说明】：多项式 p 为矩阵 A 的特征多项式。

例 2.7.1 多项式的创建和运算。

```
>> p1=[1,1];                %定义多项式 p1(x)=x+1
>> PS1=poly2str(p1,'x')     %用习惯的方式显示多项式
PS1 =
    x + 1
>> p2=[1,1,1];              %定义多项式 p2(x)=x^2+x+1
>> PS2=poly2str(p2,'x');
>> [q,r]=deconv(p2,p1)      %求 p2/p1
q =
```

```
     1     0
r =
     0     0     1
>> disp(['商：',poly2str(q,'x')]),  disp(['余式：',poly2str(r,'x')])
商：    x
余式：    1
>> p3=conv(p1,p2)          %计算 p3=p1*p2
P3=
     1     2     2     1
>> r=roots(p3)             %求多项式 p4 的根 r
r =
  -1.0000
  -0.5000 + 0.8660i
  -0.5000 - 0.8660i
>> poly(r)                 %求根为 r 的对应多项式，结果应为 p4
ans =
    1.0000    2.0000    2.0000    1.0000
>> polyder(p3);            %求 p3 的导数多项式
ans =
     3     4     2
>> den=conv([1,1],conv([1,1],[1,2]))      %定义分母多项式为(x+1)(x+1)(x+2)
>> num=[1,1,1];                           %定义分子多项式为(x^2+x+1)
>> [n,d]=polyder(num,den);                %求（num/den）的导数多项式（n/d）
n =
    -1    -2    -2    -4    -3
d =
     1     8    26    44    41    20     4
>> [r,p,k]=residue(num,den)                %求(num/den)的部分分式分解
r =                                       %结果：3/(x+2) - 2/(x+1) + 1/(x+1)^2
     3
    -2
     1
p =
   -2.0000
   -1.0000
   -1.0000
k =
     []
```

2.8 关系运算、逻辑运算和运算符

2.8.1 逻辑值

在程序流程的控制和解决问题的分析判断中，需要对某些命题的真假给出答案，因此 MATLAB 定义了逻辑值，包括“逻辑真”和“逻辑假”。对于逻辑值，MATLAB 有如下约定：

1. 在关系表达式和逻辑表达式的输入中，任何非 0 数为“逻辑真”，只有 0 为“逻辑假”。

2. 关系表达式和逻辑表达式的计算结果是一个由 0 和 1 构成的“逻辑数组”（Logical Array’），逻辑数组中“1”表示真，“0”表示假。

3. 逻辑数组属于“数值数组”的子类，它可以作为数值数组参与数值计算，也可以用于数组寻访等特殊场合。比如：用逻辑矩阵作为数组下标，可以提取数组中逻辑矩阵真值位置处的元素。

4. 关系运算符和逻辑运算符遵循数组运算规则。

2.8.2 关系运算符

MATLAB 提供的关系运算符如下：

<	小于	<=	小于等于
>	大于	>=	大于等于
= =	等于	!=	不等于

2.8.3 逻辑运算符

MATLAB 提供了 3 种逻辑操作，他们分别是数组逻辑操作、位逻辑操作和先决逻辑操作。

1. 数组逻辑操作

   & 逻辑与　　| 逻辑或　　~ 逻辑非　　xor 逻辑异或

2. 位逻辑操作函数（操作数必须是非负整形标量或者数组）

   bitand 位与　　bitor 位或　　bitcmp 位非　　bitnor 位异或

3. 先决逻辑运算符（要求操作数为标量）

   && 先决与，如果第一个操作数为假，则不判断其他操作数，直接给出结论“假”

   || 先决或，如果第一个操作数为真，则不判断其他操作数，直接给出结论“真”

例2.8.1 逻辑运算符和关系运算符的使用。

```
>> A=rand(3);                %生成 3×3 的随机数组 A
>> LA=abs(A)>0.5             %判断 A 的绝对值大于 0.5，得到一个逻辑矩阵
LA =
     0     0     1
     0     1     1
     1     1     0
>> A(LA)                     %用逻辑矩阵作为数组下标，提取数组中逻辑矩阵真值位置处的元素
ans =
    0.8214
    0.6154
    0.7919
    0.9218
    0.7382
>> (A>0.2)&(A<0.8)           %判断 A>0.2 且 A<0.8
ans =
     1     1     0
     0     1     1
     0     1     0
```

2.8.4 逻辑函数

MATLAB提供了许多逻辑函数，这些逻辑函数的运行结果是逻辑矩阵。下面介绍常用的逻辑函数，这些逻辑函数从函数名上就可以看出函数的功能。

1. any(v)

向量v中有非0元素，结果为1，否则为0。对矩阵的运算结果是行向量。

2. all(v)

向量v中都是非0元素，结果为1。对矩阵的运算结果是行向量。

3. isequal(A,B)

判断A、B是否相等，数组运算

4. ismember(A,B)

A的元素是B中的元素，则A相应位置的结果为1，否则为0。

5. 判断特殊数据的逻辑函数

isempty isfinite isinf isletter isnan isprime isreal isspace

6. 判断数据类型的逻辑函数

iscell ischar iscellstr isfield isglobal ishandle islogical
isnumeric isobject issparse isstruct

例2.8.3 逻辑函数的应用。

```
>> A=[0 1 5;2 NaN inf];
>> B=[0 0 15;2 5 inf];
>> ismember(A,B)          %判断A中的元素是否是B中的元素
ans =
     1     0     1
     1     0     1
>> isnan(A)               %判断A中的元素是否为非数
ans =
     0     0     0
     0     1     0
>> isinf(A)               %判断A中的元素是否为无穷
ans =
     0     0     0
     0     0     1
>> isfinite(A)            %判断A中的元素是否是“数”，不是nan或者inf
ans =
     1     1     1
     1     0     0
>> C=[0 1 2;0 0 3;0 4 5];
>> all(C)                 %判断C的列向量中的元素是否都是非0值
ans =
     0     0     1
>> any(C)                 %判断C的列向量中的元素是否有非0值
ans =
     0     1     1
```

例 2.8.4 将 3 阶魔方阵中大于 5 的元素改为 0。

```
>> A=magic(3);                %生成 3 阶魔法阵
```

```
>> LA=A>5                          %求取魔法阵 A>5 的逻辑矩阵
LA =
     1     0     1
     0     0     1
     0     1     0
>> pos=find(A>5)                   %找到(A>5)的逻辑矩阵的真值下标
pos =
     1
     6
     7
     8
>> A(LA)=0;                        %用逻辑数组作为矩阵下标实现赋值
>> A(pos)=0                        %用单下标方式实现赋值
A =
     0     1     0
     3     5     0
     4     0     2
```

2.8.5 运算符及其优先级

MATLAB 的运算符可分为三类：算术运算符、关系运算符和逻辑运算符。除去个别运算符外，一般可认为算术运算符的优先级最高，其次是关系运算符，再其次是逻辑运算符。表 2.8.1 是 MATLAB 运算符优先级的约定。

表 2.8.1　运算符的优先级

优先级别	运算符	功能	运算符	功能	运算符	功能	运算符	功能	运算符	功能	运算符	功能
1	()	括号	.	成员符								
2	.’	转置	’	共轭转置	.^	数组幂	^	矩阵幂				
3	+	代数正	-	代数负	~	逻辑非						
4	.*	数组乘	.\	数组除	./	数组除	*	矩阵乘	\	矩阵左除	/	矩阵右除
5	+	加	-	减								
6	:	冒号										
7	<	小于	>	大于	==	等于	>=	不小于	<=	不大于	!=	不等于
8	&	逻辑与										
9	|	逻辑非										
10	&&	先决与										
11	||	先决非										
12	=	赋值										

【说明】如果书写表达式的时候，无法弄清某些运算符的优先级，建议使用括号来规定运算次序。

2.9 字符串数组

2.9.1 字符与字符串

MATLAB 有强大的字符处理能力，字符串处理主要用于数据的可视化显示、宏操作、符号计算和文件操作等。MATLAB 中，对于字符和字符串有如下描述：

1. 字符是字符串中的一个元素，一个英文字符或者一个汉字都占用一个字符位。
2. 字符在内存中是用其 ASCII 码存储的，通常一个字符的 ASCII 码为 2 个 Byte。
3. 字符或者字符串必须放在“单引号”对中。
4. 如果字符串中出现“单引号”字符，则用 2 个“单引号”表示。
5. 字符串可以看作是一维字符数组。
6. 一维字符串数组可以看作是二维字符数组。

例 2.9.1 字符串的定义、输入和创建。

```
>> ch='s';                    %定义一个字符变量 ch
>> str='字符串测试';          %中文字符串定义
>> size(str)                  %中文字符串的大小，一个中文字符也占一个字符位
ans =
     1     5
>> str1='This''s a book';     %用 2 个单引号在字符串中表示“单引号”字符
>> str2=['one';'two'];        %用分号“;”对同样大小的字符串进行垂直连接
>> s=char('2.18','摄氏度');   %用 char 函数串接字符串
>> size(s)      %新串为 2×4 字符数组，第二个串为 3 个汉字，也分配了 4 个字符位
ans =
     2     4
```

2.9.2 字符串数组的访问

一维字符串数组相当于二维字符数组，因此其访问方式和二维数组的访问方式相同，可以采用单下标和双下标方式访问。后如无特别说明，本书中的“字符串数组”指的是一维字符串数组。

例 2.9.2 字符串的访问。

```
>> s1='28.50';
>> s2='公斤';
>> s=char(s1,s2);
>> s(2);                      %采用单下标方式访问字符数组的第二个元素
>> s(3)='0'                   %字符数组的第 3 个元素（第 1 行第 2 列）赋值字符“0”
s =
20.50
公斤
```

2.9.3 字符串转换函数

MATLAB 提供了大量的字符串转换函数，实现各种数据类型和字符串之间的相互转换。这些字符串转换函数包括：

abs　　把字符串转换为 ASCII 码

base2dec　bin2dec　dec2base　dec2hex　hex2dec　hex2num

int2str　mat2str　num2str　str2num str2mat

char　double　强制转换函数

fprintf　sprintf　sscanf　字符串的格式化输入输出函数

【说明】

1. dec 表示十进制整数；num 表示浮点数；int 表示整数；mat 表示数值矩阵；
2. base 表示任意进制串；bin 表示 2 进制串；hex 表示 16 进制串；str 表示字符串；
3. 2 表示 To（转换到），如 bin2dec 的意思是“把二进制串转换到十进制整数”。

2.9.4 字符串操作函数

用户可以使用 MATLAB 提供的字符串操作函数对字符串进行各种操作，这些操作包括：

1. 创建
   blanks　char　mat2str deblank
2. 连接
   strcat　strvcat
3. 搜索与替换
   findstr　strmatch　strrep　strtok
4. 执行宏字符串
   eval　feval
5. 逻辑判断
   ischar　isletter　isspace
6. 大小写转换
   lower　upper
7. 字符串比较
   strcmp　strncmp
8. 对齐方式
   strjust

以上字符串操作函数的详细使用方法，读者可以参阅 MATLAB 的超文本帮助。

2.10 细胞数组

2.10.1 细胞数组的数据结构

程序设计中会遇到这样的问题：为了便于处理数据和简化程序代码，希望将不同类型的数据放置在某种数据结构中，并且通过每个数据在这种数据结构中的位置索引来访问该数据。数值数组(Numberic Array)中只能放置相同类型的数据，不能解决这样的问题。

MATLAB 定义了细胞数组（Cell Array），其与数值数组的比较如下：

1. 相同点
   数值数组和细胞数组都是数组，都可以用下标（单下标、全下标）方式访问。
2. 不同点
   数值数组的元素是相同类型的数据，细胞数组的元素可以是不同类型的数据。

2.10.2 细胞数组的访问

细胞数组也是数组，因此用户可以用访问数组的方法来访问细胞数组，即通过单下标或全下标的方式访问细胞数组中的某个元素或子数组。对于细胞数组，有两个重要概念：

1. 细胞

细胞（Cell）是细胞数组中的元素，可以把细胞当作是一种新的数据类型。细胞的访问是通过圆括号（）和下标实现的，就和访问数组元素一样。

2. 细胞的内容

细胞的内容是指细胞内放置的数据，这些数据可以是任何 MATLAB 数据类型。访问细胞的内容是通过花括号{}和下标实现的。细胞的内容可以当作是普通 MATLAB 变量，使用方法和普通变量没有区别。细胞数组可以嵌套使用，即细胞的内容可以是细胞数组。

例 2.10.1 细胞数组。

```
>> str='This is a string';
>> s=2*pi;
>> v=[1,2,3,4];
>> M=reshape(1:8,2,[]);
>> A(1,1)={str};          %用{str}将字符串数组 str 构造成细胞，然后赋值给第 1 行第 1 列的细胞
>> A(1,2)={s};
>> A(2,1)={v+1};
>> A(2,2)={M}
A =
    'This is a string'    [    6.2832]
          [1x4 double]    [2x4 double]
>> A{1,1}(1,end)          %访问第 1 行第 1 列细胞内容的第 1 行最后一列
ans =
g
>> A{1,2}=pi;             %改变第 1 行第 2 列细胞的内容
>> A{2,3}={str};          %将第 2 行第 3 列细胞的内容再赋值为一个 1×1 细胞数组，即细胞的嵌套
>> A{2,3}(1)              %获取第 2 行第 3 列细胞内容的第 1 个分量数据，是一个细胞
ans =
    'This is a string'
>> [v1,v2,v3]=deal(A{[1,3,5]})  %一次读取多个细胞内容，分别赋值给多个变量
v1 =
This is a string
v2 =
    3.1416
v3 =
     []
v =
     1     2     3     4
v1 =
>> A(3,:)=[];             %删除细胞数组的第 3 行，方法和数组操作相同
```

2.11 结构体数组

2.11.1 结构体数组的数据结构

结构体是若干个数据的集合，这些数据可以是不同的MATLAB数据类型，且结构体中的每个数据都被分派了一个名字，称为“域名”或“成员名”。结构体数组是由多个结构体数据构成的数组，其基本元素为结构体数据。

结构体类型是用户根据需要，用基本数据类型构造出来的新数据类型。结构体能够更准确直观地描述客观事物，能够把和某个事物相关的属性数据构造到一个数据结构中，并给这个事物的各个属性命名，可以用直观的名字去访问这个事物的属性。合理的构造和使用结构体，能够使程序编码获得简化，大大提高程序的可读性和可维护性。

2.11.2 结构体数据的访问

结构体的访问约定如下：

1. 访问结构体数据的某个成员，要使用成员（域）运算符“.”,其格式为：
   结构体数据.成员名
2. 访问结构体数组中的元素，即结构体数据，其方法和访问数组元素相同。
3. 结构体数组可以是一维、二维和多维的。
4. 结构体的成员可以作为普通MATLAB变量使用。
5. 结构体可以嵌套使用，即结构体的成员可以是其他的结构体。
6. 结构体的成员可以是细胞数组。

2.11.3 结构体数组的创建

1.通过直接给结构体的成员赋值来创建

例2.11.1 用直接输入结构体成员内容的方法创建结构体数据

```
>> a.Num=154;                  %直接输入成员的值来创建单结构体变量a，即1×1结构体数组
>> a.Name='小王';
>> a.Age=25;
>> a.Birthday.Year=1970;   %创建Birthday成员为另外的结构体数据
>> a.Birthday.Month=1;
>> a.Birthday.Dat=10;
>> a.Tel='010-88888888';
>> a.Tel=char(a.Tel,'13901088888');
>> a
a =
          Num: 154
         Name: '小王'
          Age: 25
     Birthday: [1x1 struct]
          Tel: [2x12 char]
>> a.Birthday
```

```
ans =
     Year: 1970
    Month: 1
      Dat: 10
```

2.利用构造函数创建结构体数组

可以用函数 struct 建立结构体数组，其用法为：

s=struct('field1',values1, 'field2',values2,---)

【说明】

1. field1，field2为成员名，必须为字符串
2. values1,values2为成员的值。如果建立的不是单结构体变量（1×1结构体数组），要求它们是具有相同维数的细胞数组。
3. 任何情况下，空数组“[]”作为成员值可以创建新的空成员。
4. struct函数无法创建具有嵌套格式的结构体数据，被嵌套的结构体成员可以通过直接输入法创建。

例2.11.2 用struct函数创建结构体数据。

```
%用struct函数构造单结构体变量b
>> b=struct('Num',154,'Name','小王','Age',25,'Birthday',[],'Tel',char('010', '139'));
>> d=struct('Year',1970,'Month',1,'Day',10);    %构造单结构体变量d，表示生日
>> b.Birthday=d;%通过直接赋值的方法创建结构体变量b的成员Birthday为生日类型的结构体类型
>> b.Birthday
ans =
     Year: 1970
    Month: 1
      Day: 10
>> t=cell(2,3); %用细胞数组作为成员值，建立并初始化（2×3）结构体数组
>> Person1=struct('Num',t, 'Name',t, 'Age',t, 'Birthday',t, 'Tel',t)
Person1 =
2x3 struct array with fields:
    Num
    Name
    Age
    Birthday
    Tel
```

例2.11.3 结构体数组的创建和访问。

```
%创建一个具有3个成员的1×2的结构体数组s
>> s = struct('type',{'big','little'},'color','red','x',{3 1});
>> s(1);              %显示结构体数组s的第1个元素
>> s(1).type(3)       %显示结构体数组s的第1个元素的type成员的第3个元素
ans =
g
%给结构体数组s的第一个元素增加一个名为data的成员，其数据类型为细胞数组
>> s(1).data={'Memo',sqrt(2),[1,2;3,4]};
%显示结构体数组s的第1个元素的data成员的第3个细胞的内容的第2行和第2列
>> s(1).data{3}(2,2)
ans =   4
```

小结

数据结构是程序设计的重要基础，使用合理的数据结构去描述问题，能够缩短程序代码、简化程序结构、便于程序维护。本章主要对MATLAB常用的数据结构的定义和使用方法进行描述，同时介绍了MATLAB的运算符、关系操作和逻辑操作，结合丰富的例题和详细注释，非常容易掌握，并达到不断提高的效果。