MATLAB 数据对象及运算

本章学习目标

(1) 掌握 MATLAB 数据的表示方法。
(2) 掌握在 MATLAB 中生成和引用变量的方法。
(3) 熟悉 MATLAB 算术运算规则。
(4) 熟悉 MATLAB 结构体、元胞数组和表的应用。

在 MATLAB 应用程序中引用的数据实体，如数、字符串、变量、文件等都称为数据对象。本章首先讲述 MATLAB 描述、表达数据的方法，然后介绍 MATLAB 变量的生成和引用方法，以及 MATLAB 的算术运算规则和实现算术运算的多种方法，最后介绍在 MATLAB 应用程序中如何使用结构体、元胞数组和表存储、处理半结构化和非结构化数据。

3.1 MATLAB 数据类型

数据是有类型的，不同类型数据的存储方式不同，值域不同，运算的规则和方法不同。MATLAB 有 17 个基础数据类型，包括数值(numeric)类、字符(char)和字符串(string)类、逻辑(logical)类、表(table 和 timetable)、结构体(struct)和元胞(cell)等，分别用于处理不同的数据实体。丰富的数据类型增强了 MATLAB 表达和处理数据的能力。

3.1.1 数值数据

数值数据是科学计算中最常见、应用最多的数据。不同的数值类型适用不同的应用场景。例如，采用 RGB 模式定义颜色时，元素的类型为 uint8；定义图像颜色的饱和度时，元素的类型为 double。

1. 数值数据类型

MATLAB 中的数值可以表示为有符号整型、无符号整型、单精度(single)和双精度(double)浮点型。若没有指定类型，MATLAB 将数值数据默认按双精度浮点类型存储和处理。

1) 整型

MATLAB 支持以 1 字节、2 字节、4 字节和 8 字节存储整型数据。以 uint8 类型为例，

该类型数据在内存中占用1字节,可描述的数为0～255。表3.1列出了MATLAB系统定义的整型和对应类型的值域。

表3.1 MATLAB的整型

类 型	值 域	类 型	值 域
uint8	$0\sim2^8-1$	int8	$-2^7\sim2^7-1$
uint16	$0\sim2^{16}-1$	int16	$-2^{15}\sim2^{15}-1$
uint32	$0\sim2^{32}-1$	int32	$-2^{31}\sim2^{31}-1$
uint64	$0\sim2^{64}-1$	int64	$-2^{63}\sim2^{63}-1$

要将数据以指定的整型方式存储于工作区,则调用与类型同名的转换函数。例如,将数12345指定用16位有符号整型存储,存于变量x,命令如下。

```
x = int16(12345);
```

使用类型转换函数将浮点数转换为整数时,MATLAB将舍入到最接近的整数。如果小数部分正好是0.5,则MATLAB会从两个同样临近的整数中选用绝对值更大的整数。例如:

```
>> x = int16([-1.5, -0.8, -0.23, 1.23, 1.5, 1.89])
x =
  1×6 int16 行向量
   -2   -1    0    1    2    2
```

此外,MATLAB还提供了按指定方式将浮点数转换为整数的函数。

- round函数:四舍五入为最近的小数或整数。
- fix函数:朝零方向四舍五入为最近的整数。
- floor函数:朝负无穷大方向四舍五入。
- ceil函数:朝正无穷大方向四舍五入。

例如,分别用以上函数将向量[－1.5,－0.8,－0.23,1.23,1.5,1.89]的各个元素转换为整型数,命令和输出如下。

```
>> x1 = round([-1.5, -0.8, -0.23, 1.23, 1.5, 1.89])
x1 =
    -2    -1     0     1     2     2
>> x2 = fix([-1.5, -0.8, -0.23, 1.23, 1.5, 1.89])
x2 =
    -1     0     0     1     1     1
>> x3 = floor([-1.5, -0.8, -0.23, 1.23, 1.5, 1.89])
x3 =
    -2    -1    -1     1     1     1
>> x4 = ceil([-1.5, -0.8, -0.23, 1.23, 1.5, 1.89])
x4 =
    -1     0     0     2     2     2
```

2）浮点型

浮点型用于存储和处理实数，在 MATLAB 中，用 single 和 double 描述。MATLAB 按照 IEEE 754 标准来构造浮点数，single 型数用 4 字节（32 位二进制）存储，double 型数用 8 字节（64 位二进制）存储。因此，实数以 double 型存储，比以 single 型存储精度高。MATLAB 默认以 double 型存储数值数据。

single 函数和 double 函数用于将其他类型数据转换为 single 型和 double 型。

3）复型

MATLAB 可以存储和处理复数，用 complex 描述。复数由实部和虚部构成，在 MATLAB 中，实部和虚部默认为 double 型，虚数单位用 i 或 j 表示，例如，5+6i、x + 1i * y。

当所构造的 complex 型数的实部或虚部是非浮点型数时，须调用 complex 函数来生成 complex 型数，例如，生成一个虚部为 int8 型数的复数，命令如下。

```
>> complex(3, int8(4))
ans =
  int8
   3 +   4i
```

real 函数用于获取复型数的实部值，imag 函数用于获取复型数的虚部值。

2. 识别数据类型

为了提高运算的精度和效率，在 MATLAB 程序中，使用表 3.2 中的函数识别数据对象是否与指定类型一致。结果为 1 表示一致，为 0 表示不一致。

MATLAB 的 isa 函数用于判别数据对象是否为指定类型，isa 函数的调用格式为

```
isa(obj, ClassName)
```

其中，输入参数 obj 是要识别的数据对象，ClassName 是类型名。例如，判别数 1.23 是否为 double 类型，命令如下。

```
>> isa(1.23, 'double')
ans =
  logical
   1
```

表 3.2　判别数值数据类型的函数

函　数	说　明	示　例	
isinteger	判别是否为整型	>> isinteger(100) ans = logical 0	>> isinteger(int8(100)) ans = logical 1
isfloat	判别是否为浮点型	>> isfloat(int64(1.23e6)) ans = logical 0	>> isfloat(1.23e6) ans = logical 1

续表

函　数	说　明	示　例	
isnumerical	判别是否为数值型	>> isnumeric('1.23e6') ans = logical 0	>> isnumeric(1.23e6) ans = logical 1
isreal	判别是否为实数或复数	>> isreal(5+6i) ans = logical 0	>> isreal(5+6) ans = logical 1
isfinite	判别是否为有限值	>> isfinite(inf) ans = logical 0	>> isfinite(1.23e123) ans = logical 1
isinf	判别是否为无穷值	>> isinf(1.23e123) ans = logical 0	>> isinf(inf) ans = logical 1
isnan	判别是否为 NaN	>> isnan(0.1/0) ans = logical 0	>> isnan(0/0) ans = logical 1

3. 数据输入/输出格式

输入/输出数值数据时，可以采用日常记数法和科学记数法。例如，1.23456、－9.8765i、3.4＋5i 等是用日常记数法表示数，1.56789e2、1.234e－5 是采用科学记数法分别表示数 1.56789×10^{2}、1.234×10^{-5}。其中，字母 e 或 E 表示以 10 为底，字母前的数可以是实数、整数，不能是复数；字母后的数是 10 的幂，只能是整数。

在命令行窗口输出数据前，可以用 format 函数设置数据输出格式。format 函数的基本调用格式如下。

```
format style
```

或

```
format(style)
```

其中，输入参数 style 指定数据的输出格式，可以使用字符向量、字符串标量，常用格式字符串如表 3.3 所示，表中的示例数据是 double 类型。format 命令只影响数据的输出格式，不影响数据的存储精度。

在命令行窗口输出数据时，默认采用 losse 模式控制行距，本书为了节约排版空间，在运行示例前，执行 format compact 命令，将命令行窗口的输出模式设置为紧凑模式，不输出

空行。

表 3.3　常用输出格式

格式字符串	格　　式	输出示例 设 x=[4/3；1.2345e-6]；
short	十进制短格式(默认格式)，小数点后有 4 位有效数字	1.3333 0.0000
long	固定十进制长格式，小数点后有 15 位数字	1.333333333333333 0.000001234500000
rational	分式格式	4/3 1/810045
hex	十六进制格式	3ff5555555555555 3eb4b6231abfd271
compact	在命令行窗口输出时隐藏空行	
loose	在命令行窗口输出时有空行	

3.1.2　文本数据

网络中各种应用都会涉及文本和字符数据的处理。例如，QQ、微信、网页中传输的信息文本，传输过程中需要加密/解密、压缩/解压缩；在做社会实践调查时，常会从电子邮件、社交媒体内容和产品评论中提取文本和字符，进行排序、筛选、分词、聚类等分析和处理。在 MATLAB 中，使用字符(character)数组和字符串(string)数组来存储和处理文本数据。

1. 字符数组

MATLAB 字符数组用于存储字符序列，每个元素存储一个字符。存储单行字符序列的字符数组称为字符向量，只有 1 个元素的字符数组称为字符标量。

1) 生成字符向量

生成字符向量是通过单撇号界定字符序列来实现的，向量中的每个元素存储一个字符。例如：

```
>> ch1 = 'This is a book.';
```

若字符序列中含有单撇号，则该单撇号字符须用两个单撇号来表示。例如：

```
>> ch2 = 'It''s a book.'
ch2 =
    'It's a book.'
```

2) 生成字符数组

二维字符数组由若干字符向量构成，数组中的每行对应一个字符向量。例如：

```
>> ch = ['abcdef'; '123456'];
```

用字符向量生成字符数组时，各个字符向量的大小必须相同。如果各个字符向量长度

不等,可以使用 char 函数将长度不同的字符向量合成字符数组,例如:

```
>> language = char('Fortran','C++','MATLAB')
language =
  3×7 char 数组
    'Fortran'
    'C++    '
    'MATLAB '
```

这 4 个字符串的长度分别为 7、3、6,用 char 函数生成字符数组时,每行长度都按照字符向量的最大长度 7 进行了扩充,扩充的方法是在原字符串后添加空格。

2. 字符串数组

MATLAB 字符串数组是文本片段的集合,字符串数组中的每个元素存储一个字符串,字符串长度可以不同。只有 1 个元素的字符串数组称为字符串标量。

1) 生成字符串

生成字符串是通过双引号括起字符序列来实现的。例如:

```
>> str1 = "Hello, world";
```

如果字符序列包含双引号,则在定义中使用两个双引号。例如:

```
>> str = "They said, ""Welcome!"" and waved."
str =
"They said, "Welcome!" and waved."
```

2) 生成字符串数组

字符串数组适合存储多段文本。字符串数组中的每个元素各存储一段文本,各段文本的长度可以不同。例如:

```
A = ["a","bb","ccc"; "dddd","eeeeee","This is a test."]
A =
  2×3 string array
    "a"       "bb"        "ccc"
    "dddd"    "eeeeee"    "This is a test."
```

3) 连接字符串

在 MATLAB 中,可以通过加号运算符连接两个字符串。例如:

```
>> name = "Andy";
>> str = "Hello," + name
str =
    "Hello,Andy"
```

这种运算也可以应用于两个大小相等的字符串数组,将两个数组中对应位置的字符串两两相连。例如:

```
>> s1 = ["Red" "Blue" "Green"];
>> s2 = ["Truck" "Sky" "Tree"];
>> s = s1 + s2
s =
  1×3 string 数组
    "RedTruck"    "BlueSky"    "GreenTree"
```

3. 文本数据与其他类型数据的转换

在做文本分析时，常常需要将文字数据转换为数值，或者在给图表添加标注时，需要将数值转换为文字后输出。表 3.4 列出了 MATLAB 文本数据与其他类型数据的转换函数。

表 3.4　文本数据与其他类型数据的转换函数

函　数	功　能	示　例
char	将数值数组转换为字符数组，字符数组的每个元素为数值数组中对应元素的 ASCII 字符	>> C = char([65 66; 67 68]) C = 2×2 char 数组 'AB' 'CD'
string	将数值数组转换为字符串数组	>> S = string([65 66; 67 68]) S = 2×2 string 数组 "65"　"66" "67"　"68"
num2str	按指定精度将数值数组转换为字符数组	>> S1=num2str([12345, 789; 6.568, 0.0523], 3) S1 = 2×18 char 数组 '1.23e+04　789' '　6.57　0.0523'
mat2str	按指定精度将数值矩阵转换为字符向量	>> S2=mat2str([12345, 789; 6.568, 0.0523], 3) S2 = '[1.23e+04 789;6.57 0.0523]'
str2num	将字符串或字符向量转换为数值数组	>> C = str2num('[3.85,2.91; 7.74,8.99]') C = 3.8500　2.9100 7.7400　8.9900
feval	将字符向量转换为函数，代入参数进行计算	>> feval('mod', 10, 3) ans = 1
eval	将字符串转换为数值数组	>> C = eval("[3.8/2,2.91; 7.74,8.99]") C = 1.9000　2.9100 7.7400　8.9900

注：65、66、67、68 分别是字母 A、B、C、D 的 ASCII 码。

3.2 MATLAB变量

MATLAB变量用于存储数值和字符、符号对象、图形对象等，变量的大小、类型根据所存储的数据自动确定，也可以调用MATLAB函数（如zeros、uint8、reshape函数）指定和改变变量的大小和类型。

3.2.1 建立变量

1. 赋值

在MATLAB中，通过赋值命令来建立变量。MATLAB赋值命令的基本格式如下。

```
变量 = 表达式;
```

在执行赋值命令时，先对赋值号右端表达式进行计算，再将结果赋给左边的变量。赋值号左边变量的类型、大小根据右端表达式的计算结果自动确定。当仅有表达式时，将表达式的值赋给预定义变量ans。

如果赋值命令后没有分号，命令行窗口会输出变量的值。如果赋值命令后有分号，则仅执行赋值操作，命令行窗口不输出变量的值。例如：

```
>> x1 = 2.58; x2=int8(-16.3), x3=single(16.3)
x2 =
  int8
   -16
x3 =
  single
   16.3000
```

执行以上命令，因为x1=2.58后有分号，不输出x1的值；int8函数、single函数分别将其输入参数转换为int8类型、single类型，生成的变量x2、x3不是默认的double型，因此输出变量x2、x3的类型和值。

2. 赋句柄

在MATLAB中，还可以将函数句柄赋给变量，此时变量的类型为function_handle。构造函数句柄的方法如下。

```
f = @myfunction;
```

其中，符号@是函数句柄的定义符，myfunction可以是MATLAB内置函数（如sqrt），也可以是自定义函数。通过函数句柄调用原函数，与直接调用函数的结果一致。例如：

```
>> f1 = @sqrt;
>> f1(9)     %等效于 sqrt(9)
ans =
     3
```

在 MATLAB 程序中，通常将单行表达式用匿名函数形式来定义，然后将匿名函数句柄赋给变量。匿名函数的定义方法如下。

```
h = @(arglist)anonymous_function;
```

其中，arglist 是匿名函数的自变量，如果有多个自变量，变量之间用逗号分隔；anonymous_function 是含有自变量的表达式。例如：

```
>> h1 = @(x)3*x+x*x;
>> h2 = @(x, y)sin(x)+cos(y);
>> h3 = @(a,b,c)a*b+c;
```

定义匿名函数后，可以用存储函数句柄的变量作为函数名，代入参数，计算对应表达式的值。例如：

```
>> h3(2, 10, 6)     %计算 2*10+6
ans =
    26
```

3.2.2 MATLAB 矩阵

线性代数中，把由 $m \times n$ 个数排列成的数表称为大小为 $m \times n$ 的矩阵。若矩阵的行数 m 等于列数 n，称为方阵；若矩阵只有一列（即 n 为 1），称为列向量；若矩阵只有一行（即 m 为 1），称为行向量。

在 MATLAB 中，数据均以数组的形式进行存储和处理，矩阵是二维数组，标量视为 1×1 数组，有 n 个元素的行向量视为 $1 \times n$ 数组，有 m 个元素的列向量视为 $m \times 1$ 数组。

1. 构造矩阵和向量

1）构造矩阵

MATLAB 使用一对方括号“[ ]”（称为数组构造符）构造矩阵，同一行的各元素之间用空格或逗号分隔，行与行之间用分号分隔。例如：

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

执行以上命令，变量 A 存储了一个大小为 3×3 的数值矩阵。

MATLAB 数值矩阵的元素也可以是复数。例如，建立复数矩阵：

```
>> B = [1, 2+7i, 5*sqrt(-2); 3, 2.5i, 3.5+6i]
B =
   1.0000 + 0.0000i   2.0000 + 7.0000i   0.0000 + 7.0711i
   3.0000 + 0.0000i   0.0000 + 2.5000i   3.5000 + 6.0000i
```

MATLAB还提供了repelem、repmat函数来构造有大量相同元素的矩阵。

(1) repelem函数。

用于在构造数组时重复使用给定向量/矩阵中的元素,基本调用格式如下。

```
u = repelem(v, n)
```

输入参数v通常是标量或向量,指定构造矩阵的元素;输入参数n指定重复使用v中各元素的次数。例如:

```
>> a = [1 2 3 4];
>> t = repelem(a,3)
t =
    1    1    1    2    2    2    3    3    3    4    4    4
```

参数n也可以是向量,此时,变量n应与变量v的长度相同,变量n的元素逐一指定变量v中对应元素的重复次数。例如:

```
>> a = [1 2 3 4];
>> t = repelem(a,[3,2,2,1])
t =
    1    1    1    2    2    3    3    4
```

repelem函数的另一种调用格式如下。

```
B = repelem(A, r1, r2, … , rN)
```

输入参数A是数组,每个元素依次按参数r1, r2, …, rN进行重复,r1, r2, …, rN依次对应各个维度。例如:

```
>> a = [1 2 3 4];
>> t = repelem(a,3,2)
t =
    1    1    2    2    3    3    4    4
    1    1    2    2    3    3    4    4
    1    1    2    2    3    3    4    4
```

(2) repmat函数。

用于在构造数组时重复使用给定的数组,有以下两种调用格式。

```
B = repmat(A, n)
B = repmat(A, r1, r2, … , rN)
```

第1种格式中的输入参数n若是标量,变量B中存储A的n×n个副本;若参数n是一个二元行向量[p,q],则B中存储A的p×q个副本。第2种格式中的参数r1,r2,…,rN指定从各个维度重复使用数组A的次数,B中存储A的r1 × r2 × … × rN个副本。例如:

```
>> A = [100 200 300];
>> B = repmat(A,2)
B =
   100   200   300   100   200   300
   100   200   300   100   200   300
>> B = repmat(A,2,3)                                    %与 repmat(A,[2,3])等效
B =
   100   200   300   100   200   300   100   200   300
   100   200   300   100   200   300   100   200   300
```

2）构造等间距行向量

在 MATLAB 中，构造线性等间距的行向量通常采用冒号表达式。冒号表达式格式如下。

```
a : b : c
```

其中，a 为初始值，b 为步长，c 为终止值。冒号表达式可产生一个由 a 开始到 c 结束，以 b 为步长递增/递减的行向量。例如：

```
>> t=0:2:10
t =
     0     2     4     6     8    10
```

冒号表达式中如果省略 b，则步长默认为 1。例如，t = 0：5 与 t = 0：1：5 等价。

生成线性等间距的行向量也可使用 colon 函数，colon(a, c)等效于冒号表达式 a：c，colon(a, b, c)等效于冒号表达式 a：b：c。

linspace 函数也用于生成线性等间距的行向量，其调用格式如下。

```
linspace(a, b, n)
```

其中，输入参数 a 和 b 指定向量的第 1 个和最后 1 个元素，参数 n 指定向量元素个数。当 n 省略时，默认生成 100 个元素。显然，linspace(a, b, n)与 a:(b−a)/(n−1):b 等价。例如：

```
>> x1 = linspace(0,6,4)
x1 =
     0     2     4     6
```

如果输入参数 b＜a，则生成的向量是线性递减序列。例如：

```
>> x2 = linspace(0,-8,6)
x2 =
    0   -1.6000   -3.2000   -4.8000   -6.4000   -8.0000
```

如果要在行向量的末尾添加元素，则通过矩阵构造符和矩阵元素分隔符来实现。例如：

```
>> x12 = [x1, 100, 20]
x12 =
     0     2     4     6   100    20
```

MATLAB还提供了logspace函数，用于生成对数等间距的行向量，其调用格式为

```
logspace(a, b, n)
```

函数的用法与linspace函数相同。例如：

```
>> x3 = logspace(0,4,5)
x3 =
          1          10         100        1000       10000
```

冒号表达式、linspace函数也可用于生成datetime和duration类型值的序列，方法与生成线性等间距的数值向量方法相同。caldays函数用于设置间隔天数。例如：

```
>> tt = t1:caldays(2):t2
tt =
  1×3 datetime 数组
   2022-11-01 08:00:00   2022-11-03 08:00:00   2022-11-05 08:00:00
```

3）合并矩阵

在MATLAB中，可以使用数组构造符和合并函数来合并矩阵。

（1）数组构造符。

可以用逗号或空格连接两个行数相同的矩阵，用分号连接两个列数相同的矩阵，从而拼接成大矩阵。例如：

```
>> A = [1,2,3,4; 5,6,7,8];
>> B = [10; 20];
>> X = [A, B]
X =
     1     2     3     4    10
     5     6     7     8    20
>> C = [100,101,102,103];
>> Y = [A; C]
Y =
     1     2     3     4
     5     6     7     8
   100   101   102   103
```

（2）合并矩阵的函数。

MATLAB的cat、horzcat、vertcat等函数，用于合并矩阵。cat函数的调用格式如下。

```
C = cat(dim, A1, A2, …, An)
```

其中，输入参数dim指定合并的维度，A1，A2，…，An是要进行合并的矩阵，这些矩阵应大小兼容。例如：

```
>> A = ones(2,3); B=zeros(2,4);
>> X1 = cat(1, A, B)
错误使用 cat
要串联的数组的维度不一致
```

矩阵 A、B 的第 1 维度长度（即行数）相同，第 2 维度长度（即列数）不同，因此，不能沿第 1 维度合并，只能沿第 2 维度进行合并。执行以下命令，合并操作能够完成。

```
>> X1 = cat(2, A, B)
X1 =
     1     1     1     0     0     0     0
     1     1     1     0     0     0     0
```

若两个矩阵的第 1 维度长度相同，可以调用 horzcat 函数进行水平方向的串联；若两个矩阵的第 2 维度长度相同，则可以调用 vertcat 函数进行垂直方向的串联。

2. 构造特殊矩阵

在数据可视化分析中，经常需要用到一些特殊形式的矩阵，如零矩阵、幺矩阵、单位矩阵等。MATLAB 提供了构造这些特殊形式矩阵的函数。

1）零/幺矩阵和单位矩阵

为了提高程序的运行性能，在使用大矩阵时，通常先利用 zeros、ones 函数构造相应大小的零矩阵或幺矩阵，从而预备工作空间，而不是在程序运行时逐步扩充变量的工作空间。

- zeros 函数：生成零矩阵，即元素值全为 0 的矩阵。
- ones 函数：生成幺矩阵，即元素值全为 1 的矩阵。
- true 函数：生成元素值全为逻辑 1 的矩阵。
- false 函数：生成元素值全为逻辑 0 的矩阵。
- eye 函数：生成单位矩阵，即主对角线上元素值为 1、其他位置元素值为 0 的矩阵。

这几个函数的调用格式相似，下面以生成零矩阵的 zeros 函数为例进行说明。zeros 函数的基本调用格式如下。

```
zeros(m,n, classname)
```

调用该函数，将生成大小为 m×n 的零矩阵。若 n 省略，则生成大小为 m×m 的零矩阵；若 m 和 n 都省略，则生成一个值为 0 的标量。其中，参数 classname 用类型字符串描述（如'double'、'single'、'int8'等），指定矩阵元素的类型，省略时，元素默认为 double 类型。

例如，建立一个 2×3 的零矩阵，命令如下。

```
>> T1 = zeros(2,3);                          %或 T1=zeros([2,3]);
```

建立一个 3×3 的零矩阵，且矩阵中的元素为整型，命令如下。

```
>> T2 = zeros(3,'int16');
```

zeros 函数还有其他用法。

- zeros(m, 'like', p)生成的零矩阵元素的类型与变量 p 一致。
- zeros(sz1,sz2,…,szN)生成大小为 sz1×sz2×… ×szN 的全 0 数组,参数 sz1、sz2、…、szN 依次指定各维的长度。
- zeros(sz)生成一个由向量 sz 中元素依次指定各维大小的全 0 数组。

例如,建立一个 2×3 的零矩阵,且矩阵元素和变量 x 类型相同,命令如下。

```
>> x = 1+2i;
>> T3 = zeros(2,3,'like',x);
```

x 存储的是复数,执行命令后,T3 中的元素都是复数。

2) 随机数

虚拟游戏中的发牌、网络应用的数据加密,都需要使用随机数。在统计分析和控制过程中,在蒙特卡罗类型的数值模拟中,在具有非确定性行为的人工智能算法中,或在遗传算法中模拟神经网络,常常也需要随机数。在计算机程序中,使用的是采用某个算法得到的伪随机数。

MATLAB 提供了以下 4 个构造随机数矩阵的函数。

- rand(m, n)函数:返回一组值在(0,1)区间均匀分布的随机数,构造大小为 m×n 的矩阵。n 省略时,生成 m×m 矩阵;m 和 n 都省略时,则生成一个随机标量。
- randn(m, n)函数:返回一组均值为 0、方差为 1 的标准正态分布随机数,构造大小为 m×n 的矩阵。
- randi(imax, m, n)函数:返回一组值在[1, imax]区间均匀分布的伪随机整数,构造大小为 m×n 的矩阵。
- randperm(imax, k)函数:将[1, imax]区间的整数随机排列成一个数列,提取该数列前 k 个元素构造一个向量。同一个向量中元素的值不会重复。k 省略时,默认 k 的值是 imax。

这些函数都是从预先建立的伪随机数列中依次取出多个数,按函数的输入参数指定的方式构造矩阵。因此,在不同计算机、不同程序中以同样的方式调用同一个函数,得到的随机矩阵是相同的。若需要生成不同的随机数列,则在调用以上函数之前调用 rng 函数。rng 函数的调用格式为

```
rng(seed, generator)
```

其中,输入参数 seed 指定生成随机数的种子,可取值是 0(默认)、正整数、'shuffle'(指定用当前时间作为生成随机数的种子)、结构体;参数 generator 指定生成随机数的算法,可取值是'twister'(即梅森旋转算法,默认值)、'simdTwister'、'combRecursive'、'multFibonacci'、'philox'、'threefry'。

(1) 调用 rand 函数将得到一组在(0, 1)区间均匀分布的随机数 x。若想生成一组在(a,b)区间上均匀分布的随机数 y,可以用 $y_i=a+(b-a)x_i$ 计算得到。例如,生成在区间(10,30)内均匀分布的随机数,构造大小为 3×4 的矩阵,命令如下。

```
>> rng(0)
>> a = 10; b = 30;
>> A1 = a + (b-a) * rand(3,4)
A1 =
   26.2945   28.2675   15.5700   29.2978
   28.1158   22.6472   20.9376   13.1523
   12.5397   11.9508   29.1501   29.4119
```

由于每调用一次随机函数，MATLAB 都是从预定义的伪随机数列中依次取数，即第 n 次调用，会在第 $n-1$ 次调用取出的数后取数。命令 rng(0)将使得取数指针回退到随机数列的第 1 个位置。

(2) 调用 randn 函数将得到一组均值为 0、方差为 1 的标准正态分布随机数 x。若想生成一组均值为 μ、方差为 σ^2 的随机数 y，可用 $y_i=\mu+\sigma x_i$ 计算得到。例如，生成均值为0.6、方差为 0.1 的正态分布随机数，生成 4 阶矩阵，命令如下。

```
>> A2 = 0.6 + sqrt(0.1) * randn(4);
```

(3) 调用 randi 函数将得到一组在[1, imax]区间随机分布的整数。若想生成一组在[a,b]区间的随机数 y，可以用 $y_i=(a-1)+(b-a+1)x_i$ 计算得到。例如，生成在区间[10,30]内随机分布的整数，构造大小为 1×6 的矩阵，命令如下。

```
>> A3 = 9 + randi(21, 1, 6) ;
```

第 1 个参数 21 是 30−10+1，即区间[10,30]有 21 个整数。调用 randi 函数得到的随机数列中，可能出现值相同的元素。

(4) 调用 randperm 函数将得到一组在[1, imax]区间随机排列的整数，且每一元素值不同，用这组数构造向量。例如，生成在区间[10, 30]的随机整数，构造有 6 个元素的向量，命令如下。

```
>> A4 = 9 + randperm(21, 6);
```

3. 矩阵结构变换

在进行数据可视化时，常常需要将数据所组成的矩阵结构做一些变换后，再来分析、评估矩阵的特性。MATLAB 提供了多种变换矩阵结构的函数。

1) 提取矩阵对角线

diag 函数用于提取矩阵对角线元素，构造 1 个向量。其基本调用格式如下。

```
v = diag(A, k);
```

提取矩阵 A 的第 k 条对角线上的元素，生成 1 个列向量。参数 k 为对角线编号，k 省略时，默认为 0。例如：

```
>> A = [1,2,3; 11,12,13; 110,120,130];
>> d = diag(A)
d =
     1
    12
   130
```

由左上至右下的对角线称为主对角线(即0号对角线),由右上至左下的对角线称为副对角线。与主对角线平行,往上对角线编号依次为1、2、3、…,往下对角线编号依次为－1、－2、－3、…。例如,提取矩阵A主对角线两侧对角线的元素,命令如下。

```
>> d1 = diag(A,1)
d1 =
     2
    13
>> d2 = diag(A,-1)
d2 =
    11
   120
```

2) 构造对角矩阵

主对角线上的元素是非零值,其余元素都是0的矩阵,称为对角矩阵。对角线上的元素值为1的对角矩阵称为单位矩阵。diag函数也可用于构造对角矩阵,其调用格式如下。

```
D = diag(v, k)
```

其中,输入参数v是向量,参数k指定将向量v放置在第k号对角线。k省略时,默认k为0,即将向量v放置在主对角线。例如:

```
>> diag(10:2:14, -1)
ans =
     0     0     0     0
    10     0     0     0
     0    12     0     0
     0     0    14     0
```

用diag函数来构造5阶单位矩阵,命令如下。

```
>> E = diag([1, 1, 1, 1, 1]);                    %与 eye(5) 等效
```

3) 构造三角矩阵

三角矩阵分为上三角矩阵和下三角矩阵。上三角矩阵是位于指定对角线以下的元素值为0的矩阵,下三角矩阵是位于指定对角线以上的元素值为0的矩阵。

triu函数用于构造上三角矩阵。triu函数的基本调用格式为

```
U = triu(A, k)
```

其中，输入参数 k 指定从编号为 k 的对角线往上进行截取，当 k 省略时，默认 k 为 0。生成的矩阵 U 与 A 具有相同的行数和列数。例如，提取矩阵 A 的上三角元素，生成上三角矩阵 B，命令如下。

```
>> A = randi(99,5,5);
>> B = triu(A)
B =
    81    10    16    15    65
     0    28    97    42     4
     0     0    95    91    85
     0     0     0    79    93
     0     0     0     0    68
```

结果显示：triu 函数在构造新矩阵时，提取源矩阵的上三角部分，用其构造新矩阵的上三角部分，而新矩阵的下三角部分元素值全为 0。

在 MATLAB 中，构造下三角矩阵的函数是 tril，其用法与 triu 函数相同。

4）转置矩阵

所谓转置，即把源矩阵的第 1 行变成目标矩阵第 1 列，源矩阵的第 2 行变成目标矩阵第 2 列，以此类推。大小为 $m \times n$ 的矩阵经过转置运算后，形成大小为 $n \times m$ 的矩阵。MATLAB 中，转置运算使用运算符“.'”或“'”，或调用转置运算函数 transpose。例如：

```
>> A = randi(9,2,3)
A =
     3     6     2
     7     2     5
>> B = A.'                                %或 B = transpose(A)
B =
     3     7
     6     2
     2     5
```

复共轭转置运算是针对含复数元素的矩阵，除了将矩阵转置，还对原矩阵中的复数元素的虚部求反。复共轭转置运算使用运算符“'”，或调用函数 ctranspose。例如：

```
>> A = [3,4-1i,2+2i; 7i,1+1i,6-1i]
A =
   3.0000 + 0.0000i   4.0000 - 1.0000i   2.0000 + 2.0000i
   0.0000 + 7.0000i   1.0000 + 1.0000i   6.0000 - 1.0000i
>> B1 = A'                               %或 B1 = ctranspose(A)
B1 =
   3.0000 + 0.0000i   0.0000 - 7.0000i
   4.0000 + 1.0000i   1.0000 - 1.0000i
   2.0000 - 2.0000i   6.0000 + 1.0000i
```

矩阵 B1 中的元素与矩阵 A 的对应元素虚部符号相反。如果仅对复数矩阵进行转置，命令如下。

```
>> B2=A.';                                          %或 B2=transpose(A)
```

5）旋转矩阵

在MATLAB中，rot90函数用于以90°为单位对矩阵A进行旋转。函数的调用格式如下。

```
rot90(A, k)
```

其中，输入参数k指定90°的倍数（必须为整数），当k省略时，k默认为1。旋转矩阵时，以矩阵左上角为支点，若k为正整数，则将矩阵A按逆时针方向进行旋转；若k为负整数，则将矩阵A按顺时针方向进行旋转。

例如，将矩阵A按顺时针方向旋转90°，命令如下。

```
>> A = rand(3,2)
A =
    0.5060    0.9593
    0.6991    0.5472
    0.8909    0.1386
>> B = rot90(A,-1)
B =
    0.8909    0.6991    0.5060
    0.1386    0.5472    0.9593
```

若要将矩阵A按逆时针方向旋转90°，则命令如下。

```
>> C=rot90(A);
```

6）翻转矩阵

矩阵的翻转分为左右翻转和上下翻转。fliplr函数用于对矩阵A实施水平方向的翻转，其调用格式如下。

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

对矩阵实施左右翻转是将原矩阵的第1列和最后1列调换，第2列和倒数第2列调换，以此类推。例如：

```
>> A = randi(99,2,5)
A =
    78    13    47    34    79
    93    57     2    17    31
>> B = fliplr(A)
B =
    79    34    47    13    78
    31    17     2    57    93
```

flipud 函数用于对矩阵 A 实施垂直方向的翻转，用法与 flipud 函数相似。矩阵的上下翻转是将原矩阵的第 1 行与最后 1 行调换，第 2 行与倒数第 2 行调换，以此类推。

MATLAB 还提供了 flip 函数翻转数组，其调用格式如下。

```
B = flip(A, dim)
```

其中，参数 dim 指定翻转的维度，若 A 是矩阵，则当 dim 为 1（默认值）时，沿垂直方向翻转；dim 为 2 时，沿水平方向翻转。dim 省略时，默认在大小不等于 1 的首个维度上进行翻转。

3.2.3 MATLAB 数组

向量、矩阵是二维数组，多维数组是指具有两个以上维度的数组。例如，将彩色图像数据导入工作区，生成的就是三维数组。

1. 构造数组

多维数组是二维矩阵的扩展。构造多维数组，可以先构造二维矩阵，再进行扩展。

1）构造三维数组

三维数组使用三个下标，可以看成连续排列的多个矩阵，前两个维度对应于一个矩阵的行、列，第三个维度对应于矩阵的排列次序。

在 MATLAB 中，通常通过给存储矩阵的变量增加第三维坐标的方式，将矩阵扩展为三维数组。例如，先定义一个大小为 3×3 的矩阵，存储于变量 A，然后将 A 扩展为 3×3×2 的三维数组，命令如下。

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(:, :, 2) = [10 11 12; 13 14 15; 16 17 18];
```

第 2 条命令中的 A(:, :, 2)的第 1 个冒号表示所有行，第 2 个冒号表示所有列，第 3 个参数指定赋值号右端的这个矩阵存储为变量 A 的第 2 个矩阵，此时 A 的大小变为 3×3×2，前面构造的矩阵此时自动转变为变量 A 的第 1 个矩阵。

2）构造特殊数组

前面介绍的构造特殊矩阵的 zeros、ones、rand 等函数都可以扩展应用到多维数组。例如，生成大小为 3×2×3 的随机数组，命令如下。

```
>> X = rand([3,2,3]);
```

3）重构数组

构造数组后，可以利用 reshape 函数改变数组结构。reshape 函数的调用格式如下。

```
B = reshape(A, sz)
```

其中，输入参数 sz 是一个向量，向量中的元素依次指定各个维度的大小。例如，将有 6 个元素的向量重构为大小为 2×3 的矩阵，命令如下。

```
>> A = reshape(1:6, [2, 3])
A =
     1     3     5
     2     4     6
```

reshape 函数还有另一种调用格式：

```
B = reshape(A, sz1, … , szN)
```

其中，参数 sz1～szN 都是标量，用于依次指定各个维度的大小。例如，将有 12 个元素的行向量重构为大小为 2×3×2 的数组，命令如下。

```
>> A3=reshape(1:12, 2, 3, 2);
```

2. 获取数组属性

将数据导入工作区后，需要了解数据的规模、数据类型等属性，为数据的进一步处理提供依据。MATLAB 提供了 size、length、class 等函数获取数组的大小、类型等属性。

1）查询数据规模

size 函数用于获取数组的大小。size 函数的调用格式如下。

```
szdim = size(A, dim)
```

其中，输入参数 dim 指定维度，dim 省略时，默认返回所有维度的长度。例如，获取前面建立的矩阵 A3 的大小：

```
>> size(A3)
ans =
     2     3     2
>> size(A3, 2)
ans =
     3
```

length 函数用于获取数组最长维度元素的个数。length 函数的调用格式如下。

```
L = length(A)
```

若输入参数 A 是向量，则返回向量中元素的个数。例如：

```
>> length(1:2:10)
ans =
    5
```

若 length 函数的输入参数是数组，则返回数组中最长维度元素的个数。例如：

```
>> length(A3)
```