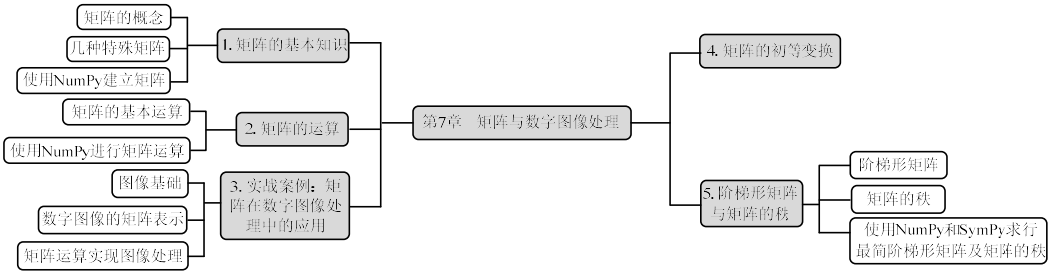
第 7 章　矩阵与数字图像处理

知识图谱：



学习目标：

（1）理解矩阵的概念，了解常见的几种特殊矩阵；
（2）掌握矩阵的加法、减法、数量乘法、乘法、转置和哈达马积；
（3）掌握使用 NumPy 建立矩阵并实现各种运算的方法；
（4）掌握矩阵初等变换的方法；
（5）理解矩阵秩的概念，掌握矩阵秩的求解方法；
（6）了解矩阵运算与数字图像处理的关系。

7.1　矩阵的基本知识

7.1.1　矩阵的概念

矩阵是线性代数中的一个重要概念和基本工具，是线性代数的主要研究对象之一，在系统理论、计算机科学、人工智能等领域都有广泛的应用。

【例 7-1】 某年某工厂 A、B、C 三种产品上半年的生产量如表 7-1 所示。

表 7-1　某工厂上半年生产量

（单位：万件）

产　品	月　份					
	1	2	3	4	5	6
A	1.8	2.3	1.9	2.3	2.9	2.4

续表

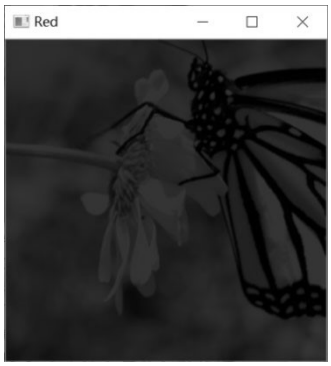
产　品	月　份					
	1	2	3	4	5	6
B	4.0	2.9	3.8	4.2	5.0	3.8
C	0.1	0.3	0.3	0.2	0.4	0.1

如果规定好产品和月份的顺序，那么产量可以用矩阵来描述：

$$\begin{pmatrix} 1.8 & 2.3 & 1.9 & 2.3 & 2.9 & 2.4 \\ 4.0 & 2.9 & 3.8 & 4.2 & 5.0 & 3.8 \\ 0.1 & 0.3 & 0.3 & 0.2 & 0.4 & 0.1 \end{pmatrix}。$$

【例 7-2】 计算机上可以实现颜色绚丽的图像。每像素点的颜色值都包含红色（R）、绿色（G）、蓝色（B）三个分量。所以计算机实现的每幅图像都对应三个颜色分量值，如图 7-1 所示。

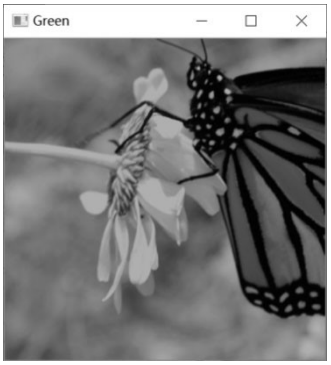
（a）原始图像



221	218	217	198	177	184	172	172	176	147	158	195	168	146	175
219	216	207	206	189	159	162	154	204	99	163	217	211	179	196
207	208	171	192	198	176	245	238	156	40	110	4	17	17	23
180	185	175	174	168	125	220	205	198	0	35	84	168	164	237
152	157	158	132	132	143	202	88	213	7	4	162	169	16	101
150	137	149	118	154	235	31	245	37	169	3	87	8	78	245
238	196	190	101	214	88	150	206	229	2	6	168	192	174	8
166	203	178	143	174	175	212	193	217	124	86	20	203	10	25
191	215	203	210	203	235	173	178	206	9	145	162	140	226	9
236	226	160	154	186	135	245	205	185	191	144	138	175	153	15
225	216	163	153	245	126	206	245	126	193	74	3	245	4	182
222	161	143	159	157	198	186	208	187	115	107	2	241	101	17
187	156	146	147	145	241	244	149	183	141	118	147	221	3	246
175	147	159	145	219	236	239	130	135	148	179	157	201	244	245
172	148	142	160	229	227	243	135	173	186	167	158	149	244	245
157	156	166	192	195	102	176	147	171	183	173	107	5	4	5
153	141	178	175	183	108	70	137	173	201	174	148	4	4	18
177	138	134	134	140	114	110	124	162	186	192	212	182	153	217
170	138	119	145	138	146	169	143	134	152	163	169	182	215	219
98	117	113	141	160	157	167	170	168	173	123	146	190	201	236

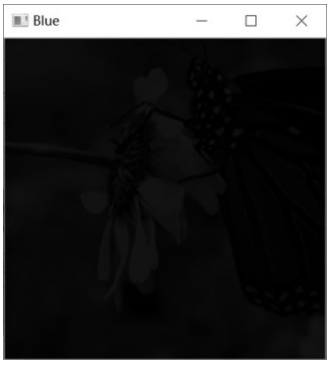
（b）红色分量及矩阵

图 7-1　原始图像与各颜色分量



199	196	198	173	148	157	161	153	153	134	130	159	162	162	201
197	197	183	184	167	139	158	136	174	87	132	183	212	200	180
189	180	152	162	179	166	246	237	131	39	89	4	17	5	6
161	159	158	147	157	134	223	186	177	0	29	29	33	39	71
127	138	139	122	134	141	175	86	194	7	5	105	35	8	90
120	120	125	133	147	210	27	245	33	170	4	28	5	53	95
245	201	166	114	200	117	161	206	232	1	7	109	90	105	4
142	188	161	187	231	203	191	189	221	131	88	8	112	5	11
170	197	198	196	199	241	176	175	199	11	101	110	115	112	0
227	209	166	159	190	117	245	206	178	193	98	103	90	102	5
215	197	171	164	245	115	202	245	115	194	50	4	155	4	105
206	175	164	162	160	192	170	207	179	110	61	3	133	63	16
162	163	163	153	157	241	244	128	170	137	116	104	127	4	145
158	161	158	152	221	238	240	103	120	117	215	105	116	126	146
170	157	150	163	230	226	243	103	146	169	216	100	89	128	146
149	158	175	171	166	122	164	118	154	165	183	100	5	5	4
155	150	178	167	182	130	123	157	161	192	166	131	4	5	19
180	151	153	150	153	137	131	141	181	190	170	182	168	155	223
161	167	141	165	148	155	182	172	154	161	175	186	183	197	223
107	143	138	161	165	141	182	218	194	180	169	182	205	196	236

（c）绿色分量及矩阵



189	184	194	160	124	106	89	95	85	94	87	119	98	83	99
183	188	171	170	148	97	85	79	120	86	81	145	165	99	93
163	160	138	133	148	120	251	247	86	49	77	6	34	4	14
130	126	130	110	113	102	233	142	135	2	32	2	0	1	1
87	107	81	95	110	109	128	100	153	15	7	33	1	2	90
85	103	69	92	109	175	12	255	35	178	7	2	7	62	0
212	155	126	100	169	54	48	199	239	7	11	39	41	35	3
95	133	121	43	55	77	57	173	228	153	102	1	44	2	2
116	151	176	184	175	214	52	157	184	17	50	41	91	46	0
210	175	134	134	167	46	255	223	149	189	37	67	39	35	4
187	159	124	121	255	41	193	255	71	195	30	8	27	12	39
170	126	100	112	111	158	136	214	157	60	16	10	32	44	20
106	113	99	105	93	250	252	90	139	83	87	47	32	9	20
94	108	109	101	230	245	243	64	73	89	152	29	35	28	10
117	113	103	126	234	233	253	71	93	123	128	23	50	28	16
104	105	114	122	147	67	135	83	101	119	119	94	12	10	9
105	99	123	107	134	78	0	67	103	133	101	79	7	9	26
112	91	80	97	106	66	61	58	98	110	104	140	113	95	199
102	77	57	99	97	90	101	87	84	75	91	104	123	153	227
54	69	54	92	106	85	112	133	91	92	64	86	152	135	245

（d）蓝色分量及矩阵

图 7-1　原始图像与各颜色分量（续）

【例 7-3】 北京、巴黎、纽约之间的飞机航线距离如图 7-2 所示。

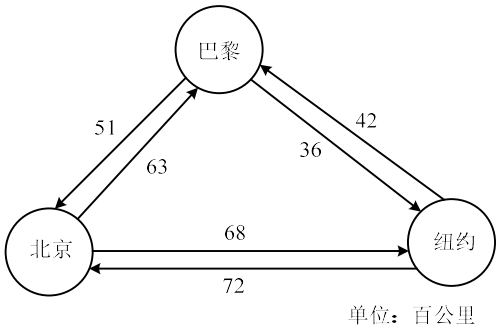


图 7-2　北京、巴黎、纽约之间的飞机航线距离

由于计算机难以直接处理类似图 7-2 所示图形一类的对象，这里将其转换为表格形式，如表 7-2 所示。

表 7-2　北京、巴黎、纽约之间的航线距离表　　（单位：百公里）

	北　京	巴　黎	纽　约
北京	0	63	68
巴黎	51	0	36
纽约	72	42	0

【例 7-4】 方程组

$$\begin{cases} k_1+3k_2-7k_3+2k_4=3, \\ 2k_1-k_2-k_3+3k_4=0, \\ -k_1+2k_2+5k_3-6k_4=1, \\ 4k_1+k_2-8k_3-3k_4=-6, \end{cases}$$

其中，系数连同最右侧的常数项按原来的位置可以构成一个表格

$$\begin{pmatrix} 1 & 3 & -7 & 2 & 3 \\ 2 & -1 & -1 & 3 & 0 \\ -1 & 2 & 5 & -6 & 1 \\ 4 & 1 & -8 & -3 & -6 \end{pmatrix}。$$

从前面的几个例子看到，生产统计、计算机图像、航线距离和线性方程组等问题中，都可以把性质相同的一些元素放在一起，表示为由数构成的表格形式。我们把这些由数构成的表格称为“矩阵”。

定义 7-1　由 $m\times n$ 个数 $a_{ij}(i=1,2,\cdots,m;\ j=1,2,\cdots,n)$ 排成的 m 行 n 列的矩形图表，称为一个 $m\times n$ 的**矩阵**，记作

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} 或 \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}。$$

其中，a_{ij} 是位于第 i 行、第 j 列的**元素**，i 称为**行脚标**，j 称为**列脚标**。

通常，矩阵用大写字母 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 表示，有时为了表明矩阵的行数和列数，可记作

$$\boldsymbol{A}_{m\times n} \quad 或 \quad (a_{ij})_{m\times n} \quad 或 \quad \boldsymbol{A}=(a_{ij})_{m\times n}。$$

所有元素都是零的矩阵称为**零矩阵**，记作 $\boldsymbol{O}_{m\times n}$ 或 $\boldsymbol{O}$。

当 $m=n$ 时，$\boldsymbol{A}_{n\times n}$ 称为 **n 阶方阵**，它由 n^2 个元素构成，成一个正方形数表：

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}。$$

其中，从左上角到右下角的对角线称为矩阵的**主对角线**，其上的元素是 $a_{11},a_{22},\cdots,a_{nn}$，称为**主对角线元素**。

当对矩阵 $\boldsymbol{A}=(a_{ij})_{m\times n}$ 中所有元素都取相反数时，得到的矩阵称为 $\boldsymbol{A}$ 的**负矩阵**，记作 $-\boldsymbol{A}$，即

$$-\boldsymbol{A}=\begin{bmatrix} -a_{11} & \cdots & -a_{1n} \\ \vdots & & \vdots \\ -a_{m1} & \cdots & -a_{mn} \end{bmatrix}。$$

向量也可以看作矩阵：

当 $m=1$ 时，$\boldsymbol{A}_{1\times n}=(a_{11},a_{12},\cdots,a_{1n})$ 是 n 维行向量，也称为**行矩阵**；

当 $n=1$ 时，$\boldsymbol{A}_{m\times 1}=\begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{pmatrix}$ 是 m 维列向量，也称为**列矩阵**。

7.1.2　几种特殊矩阵

1. 对角矩阵

如果 n 阶方阵 $\boldsymbol{A}$ 中，除主对角线外，其余元素都是 0，称为 n **阶对角矩阵**，即

$$\boldsymbol{A}=\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix} \text{ 或简写为 } \boldsymbol{A}=\begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix}。$$

注意，对角矩阵是方阵。

例如，$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$ 是一个 4 阶对角矩阵，也可写作 $\begin{pmatrix} 2 & & & \\ & -3 & & \\ & & 0 & \\ & & & 4 \end{pmatrix}$。

2. 数量矩阵

当 n 阶对角矩阵中主对角线上的元素相同时，称为**数量矩阵**，形式为

$$\begin{pmatrix} a & & & \\ & a & & \\ & & \ddots & \\ & & & a \end{pmatrix}。$$

3. 单位矩阵

n 阶数量矩阵中，如果主对角线上的元素都是 1，其余元素都是 0，称为 n 阶**单位矩阵**，记为 $\boldsymbol{E}_n$ 或 $\boldsymbol{E}$，即

$$\boldsymbol{E}=\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \text{ 或 } \boldsymbol{E}=\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}。$$

4. n 阶上三角矩阵与 n 阶下三角矩阵

如果 n 阶方阵中，主对角线以下的元素全部为 0，则称为 n **阶上三角矩阵**，结构为

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}。$$

如果 n 阶方阵中，主对角线以上的元素全部为零，则称为 n **阶下三角矩阵**，结构为

$$\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}。$$

上三角矩阵和下三角矩阵统称为**三角矩阵**。一个方阵既是上三角矩阵，又是下三角矩阵，当且仅当它是对角矩阵。

7.1.3　使用 NumPy 建立矩阵

1. 创建数组

可以调用 NumPy 的 array()函数创建一个向量或数组。

```
numpy.array(object, dtype = None, copy = True, order = None, subok =
 False, ndmin = 0)
```

其中：

- object 表示数组或嵌套的数列。
- dtype 表示数组元素的数据类型，可选。
- copy 表示对象是否需要复制，可选，默认为 true。
- order 表示创建数组的样式，C 为行方向，F 为列方向，A 为任意方向（默认）。
- subok 表示默认情况下，返回的数组被强制为基类数组。如果为 true，则返回子类。
- ndmin 表示指定生成数组的最小维度。

【例 7-5】 使用 NumPy 建立矩阵

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

并输出它的形状。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-5）

```
1. # 创建一个 2*3 数组
2. import numpy as np
3. a=np.array([[1,2,3],[4,5,6]])
4. print("a:\n",a)
5. print("a.shape:\n",a.shape)          # 通过 a.shape 查看 a 的形状
```

运行结果：

```
a:
 [[1 2 3]
 [4 5 6]]
a.shape:
 (2, 3)
```

【例 7-6】 使用 NumPy 创建了一个行向量和一个列向量。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-6）

```
1. # 使用最小维度创建矩阵
2. import numpy as np
3. b=np.array([[1,2,3]])                # 创建行向量
4. print(b)
5. print(b.shape)
6. c=np.array([[1],[2],[3]])            # 创建列向量
7. print(c)
8. print(c.shape)
```

运行结果：

```
[[1 2 3]]
```

```
(1, 3)

[[1]
 [2]
 [3]]

(3, 1)
```

注意，在创建行向量时，b=np.array([[1,2,3]])中必须有两个方括号。如果只有一个方括号，则表示一个一维数组（既不是行向量，也不是列向量），可以用 reshape 函数将其转换为指定维度的向量。

【例 7-7】 创建一个一维数组，然后分别将其转换为行向量和列向量。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-7）

```
1. # 创建一维数组并转换为行（列）向量
2. import numpy as np
3. x=np.array([1,2,3])          # 创建一维数组
4. print(x)
5. print(x.shape)
6. a=x.reshape((1,3))           # 创建行向量 a
7. print(a.shape)
8. b=x.reshape((3,1))           # 创建列向量 b
9. print(b.shape)
```

运行结果：

```
[1 2 3]
(3,)
(1, 3)
(3, 1)
```

使用 NumPy 创建矩阵时也可以指定最小维度。

【例 7-8】 指定最小维度创建矩阵。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-8）

```
1. # 使用最小维度创建矩阵
2. import numpy as np
3. a=np.array([1,2,3,4,5,6],ndmin=1)   # 创建最小维度为 1 的矩阵
4. print(a)
5. b=np.array([1,2,3,4,5,6],ndmin=2)   # 创建最小维度为 2 的矩阵
6. print(b)
```

运行结果：

```
a:  [1 2 3 4 5 6]
b:  [[1 2 3 4 5 6]]
```

创建矩阵时，可以使用参数 dtype 指定矩阵元素的数据类型。dtype 可以将矩阵元素的数据类型设置为 bool，int8，int32，uint8，uint32，float32，float64，complex 等。

【例 7-9】 创建矩阵并指定数据类型。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-9）

```
1. # 创建矩阵并指定数据类型
2. import numpy as np
3. a = np.array([1,2,3,4,5,6],dtype=np.uint32)
4. b = np.array([[1,2],[3,4]],dtype=np.complex)
5. print("a: ",a)
6. print("b: ",b)
```

运行结果：

```
a:  [1 2 3 4 5 6]
b:  [[1.+0.j 2.+0.j]
     [3.+0.j 4.+0.j]]
```

2. 创建特殊的数组

对于一些常用的特殊矩阵，如单位矩阵、零矩阵等，可以调用内置的函数直接创建。

【例 7-10】 创建特殊矩阵。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-10）

```
1. # 创建特殊矩阵
2. import numpy as np
3. a1=np.zeros((2,3))          # 创建一个 2×3 的零矩阵
4. print(a1)
5. b=np.ones((3,4))            # 创建一个 3×4 的矩阵，所有元素都为 1
6. print(b)
7. c=np.eye(3)                 # 创建一个 3 阶的单位矩阵
8. print(c)
9. d=np.full((2,3),9)          # 创建一个 2×3 的常数矩阵，所有元素为 9
10. print(d)
```

运行结果：

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

[[9 9 9]
 [9 9 9]]
```

zeros_like()，ones_like()，full_like()等函数用于创建与参数数组的形状和类型相同的矩阵。

【例 7-11】 创建与指定矩阵形状相同的矩阵。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-11）

```
1. # 创建与指定矩阵形状相同的矩阵
2. import numpy as np
3.
4. a = np.array([[1,2,3],[4,5,6]])    # 创建一个 2×3 的矩阵 a
5. b=np.zeros_like(a)                 # 创建一个与矩阵 a 形状相同的零矩阵
6. print(b)
7. c=np.ones_like(a)                  # 创建一个与矩阵 a 形状相同的元素都为 1 的矩阵
8. print(c)
9. d=np.full_like(a,7)                # 创建一个与矩阵 a 形状相同的元素都为 7 的矩阵
10. print(d)
```

运行结果：

```
[[0 0 0]
 [0 0 0]]

[[1 1 1]
 [1 1 1]]

[[7 7 7]
 [7 7 7]]
```

Numpy 还提供了 arange()，linspace()和 logspace()等函数用于构造等差数列和等比数列。其中，arange()函数类似于内置函数 range()，通过指定起始值、终值和步长来创建表示等差数列的一维数组，注意所得到的结果中不包含终值。arange()语法格式为

```
1. np.arange([start, ]stop, [step, ]dtype=None)
```

其中：

- start 表示起始值，可忽略不写，默认从 0 开始。
- stop 表示结束值；生成的元素不包括结束值。
- step 表示步长，可忽略不写，默认为 1。
- dtype 表示设置显示元素的数据类型，默认为 None。

【例 7-12】 使用 arange()函数创建等差数列。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-12）

```
1. # 使用 arange()函数创建等差数列
2. print(np.arange(3) )    # 创建起始值为 0，终值为 3，默认步长为 1 的等差数列
3. print(np.arange(3.0) ) # 创建起始值为 0，终值为 3.0，默认步长为 1.0 的等差数列
4. print(np.arange(3,7) ) # 创建起始值为 3，终值为 7，默认步长为 1 的等差数列
5. print(np.arange(3,8,2))# 创建起始值为 3，终值为 8，步长为 2 的等差数列
```

运行结果：

```
[0 1 2]
[0. 1. 2.]
[3 4 5 6]
[3 5 7]
```

linspace()函数通过指定起始值、终值和元素个数来创建表示等差数列的一维数组，可以通过 endpoint 参数指定是否包含终值，默认值为 True，即包含终值。语法格式为

```
1. numpy.linspace(start,stop[, num=50[, endpoint=True[, retstep=False
[, dtype=None]]]])
```

其中：

- start 表示起始值。
- stop 表示终值。
- num 表示元素个数，默认为 50。
- endpoint 表示是否包含 stop 数值，默认为 True，包含 stop 值；否则为 False。
- retstep 表示返回值形式，默认为 False，返回等差数列组；若为 True，则返回结果(array(['samples', 'step']))。
- dtype 表示返回结果的数据类型，默认无，若无，则参考输入数据类型。

logspace()函数与 linspace()函数类似，但 logspace()函数所创建的数组是等比数列。语法格式为

```
1. numpy.logscale(start, stop, num, endpoint, base, dtype)
```

其中：

- start 表示起始值是 base ** start。
- stop 表示终止值是 base ** stop。
- num 表示范围内的数值数量，默认为 50。
- endpoint 如果为 True，表示终止值包含在输出数组中。
- base 表示对数空间的底数，默认为 10。
- dtype 表示输出数组的数据类型，如果没有提供，则取决于其他参数。

【例 7-13】 使用 linspace()和 logspace()函数创建等差数列和等比数列。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-13）

```
1. # 使用 linspace()和 logspace()函数创建等差数列和等比数列
2. a = np.linspace(10,20,9)   # 创建初始值为 10，终值为 20，包含 9 个点的等差数列，
                              # 包含终值 20
3. print(a)
4. b = np.linspace(10,20,5,endpoint=False)   # 创建初始值为 10，终值为 20，包含
                                               5 个点的等差数列，不包含终值 20
5. print(b)
6. c = np.linspace(10,20,5,retstep=True)     # 创建初始值为 10，终值为 20，包含
                                               5 个点的等差数列
7. print(c)                                  #返回数组和步长
8. d = np.logspace(0,4,5)   # 创建初始值为 10**0，终值为 10**4，默认底数为 10，
                              包含 5 个点的等比数列
9. print(d)
10. e = np.logspace(0,4,5,base=2)   # 创建初始值为 10**0，终值为 10**4，
                                      底数为 2，包含 5 个点的等比数列
11. print(e)
```

运行结果：

```
[10.   11.25 12.5  13.75 15.   16.25 17.5  18.75 20.  ]
[10. 12. 14. 16. 18.]
(array([10. , 12.5, 15. , 17.5, 20. ]), 2.5)
[1.e+00 1.e+01 1.e+02 1.e+03 1.e+04]
[ 1.  2.  4.  8.  16.]
```

3. 查看矩阵属性

建立矩阵后，可以使用 dtype、size 等命令查看矩阵的属性。

【例 7-14】 查看矩阵的属性。（代码：ch7 矩阵与数字图像处理\7.1 矩阵的概念\例 7-14）

```
1. # 查看矩阵的属性
2. import numpy as np
3. x=np.eye(5)
4. print("数组元素数据类型：",x.dtype)   # 打印数组元素数据类型
5. print("数组元素总数：",x.size)        # 打印数组尺寸，即数组元素总数
6. print("数组形状：",x.shape)           # 打印数组形状
7. print("数组的维度数目",x.ndim)        # 打印数组的维度数目
8. print("数组每个元素占用的内存大小",x.itemsize) # 数组每个元素占用的内存大小，
                                                以字节为单位
9. print("数组占用的总内存大小",x.nbytes) # 数组占用的总内存大小，以字节为单位
```

运行结果：

```
数组元素数据类型：float64
数组元素总数：25
数组形状：(5, 5)
数组的维度数目 2
数组每个元素占用的内存大小 8
数组占用的总内存大小 200
```

7.2　矩阵的运算

矩阵的运算包括加法、数乘、乘法等，只有对矩阵定义了一些有意义的计算后，才能使它成为进行理论研究和解决实际问题的有力工具。

7.2.1　矩阵的基本运算

1. 矩阵的加法

定义 7-2　设 $\boldsymbol{A}_{m\times n}=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$，$\boldsymbol{B}_{m\times n}=\begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & & \vdots \\ b_{m1} & \cdots & b_{mn} \end{pmatrix}$，则矩阵

$$\boldsymbol{C}_{m\times n}=\begin{pmatrix} a_{11}+b_{11} & \cdots & a_{1n}+b_{1n} \\ \vdots & & \vdots \\ a_{m1}+b_{m1} & \cdots & a_{mn}+b_{mn} \end{pmatrix} \tag{7-1}$$

称为 $\boldsymbol{A}_{m\times n}$和$\boldsymbol{B}_{m\times n}$ 的和，记作 $\boldsymbol{C}_{m\times n}=\boldsymbol{A}_{m\times n}+\boldsymbol{B}_{m\times n}$，如图 7-3 所示。

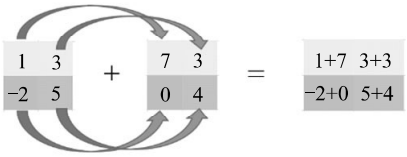


图 7-3　矩阵的加法

根据负矩阵的概念，可以定义矩阵的减法：

$$\boldsymbol{A}_{m\times n}-\boldsymbol{B}_{m\times n}=\boldsymbol{A}_{m\times n}+\left(-\boldsymbol{B}_{m\times n}\right)=\begin{pmatrix} a_{11}-b_{11} & \cdots & a_{1n}-b_{1n} \\ \vdots & & \vdots \\ a_{m1}-b_{m1} & \cdots & a_{mn}-b_{mn} \end{pmatrix} \tag{7-2}$$

【例 7-15】 设 $\boldsymbol{A}=\begin{pmatrix} -1 & 2 & 5 \\ 3 & -2 & 7 \end{pmatrix}$，$\boldsymbol{B}=\begin{pmatrix} 3 & 0 & -5 \\ 4 & 2 & 1 \end{pmatrix}$，求 $\boldsymbol{A}+\boldsymbol{B}$ 和 $\boldsymbol{A}-\boldsymbol{B}$。

解：$\boldsymbol{A}+\boldsymbol{B}=\begin{pmatrix} -1+3 & 2+0 & 5+(-5) \\ 3+4 & -2+2 & 7+1 \end{pmatrix}=\begin{pmatrix} 2 & 2 & 0 \\ 7 & 0 & 8 \end{pmatrix}$，

$\boldsymbol{A}-\boldsymbol{B}=\begin{pmatrix} -1-3 & 2-0 & 5-(-5) \\ 3-4 & -2-2 & 7-1 \end{pmatrix}=\begin{pmatrix} -4 & 2 & 10 \\ -1 & -4 & 6 \end{pmatrix}$。

矩阵的加减法即对应元素之间的加减，所以矩阵的加法有以下运算性质。

设 $\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$ 都是 $m\times n$ 矩阵，$\boldsymbol{O}$ 是 $m\times n$ 零矩阵，则

（1）**交换律**　$\boldsymbol{A}+\boldsymbol{B}=\boldsymbol{B}+\boldsymbol{A}$；　（7-3）

（2）**结合律**　$(\boldsymbol{A}+\boldsymbol{B})+\boldsymbol{C}=\boldsymbol{A}+(\boldsymbol{B}+\boldsymbol{C})$；　（7-4）

（3）$\boldsymbol{A}+\boldsymbol{O}=\boldsymbol{O}+\boldsymbol{A}=\boldsymbol{A}$。　（7-5）

2. 矩阵的数量乘法

定义 7-3　设矩阵 $\boldsymbol{A}_{m\times n}=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$，$k$ 为任意数，定义

$$k\boldsymbol{A}_{m\times n}=\begin{pmatrix} ka_{11} & \cdots & ka_{1n} \\ \vdots & & \vdots \\ ka_{m1} & \cdots & ka_{mn} \end{pmatrix} \tag{7-6}$$

称为 k 与 $\boldsymbol{A}_{m\times n}$ 的**数量乘积**，简称**数乘**，如图 7-4 所示。

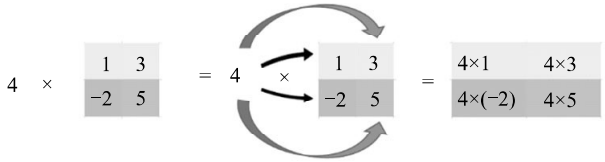


图 7-4　矩阵的数乘

【例 7-16】 设

$$\boldsymbol{A}=\begin{pmatrix}1&-3&2&0\\-4&2&-1&1\\3&5&-3&4\end{pmatrix},\quad \boldsymbol{B}=\begin{pmatrix}-1&-4&2&0\\0&3&-6&1\\-5&2&0&-4\end{pmatrix},$$

求矩阵 $\boldsymbol{X}$ 使得 $3\boldsymbol{A}-2\boldsymbol{X}=\boldsymbol{B}$。

解：

$$\boldsymbol{X}=\frac{1}{2}(3\boldsymbol{A}-\boldsymbol{B})=\frac{1}{2}\left(\begin{pmatrix}3&-9&6&0\\-12&6&-3&3\\9&15&-9&12\end{pmatrix}-\begin{pmatrix}-1&-4&2&0\\0&3&-6&1\\-5&2&0&-4\end{pmatrix}\right)$$

$$=\frac{1}{2}\begin{pmatrix}4&-5&4&0\\-12&3&3&2\\14&13&-9&16\end{pmatrix}$$

$$=\begin{pmatrix}2&-\frac{5}{2}&2&0\\-6&\frac{3}{2}&\frac{3}{2}&1\\7&\frac{13}{2}&\frac{-9}{2}&8\end{pmatrix}。$$

根据矩阵数乘的定义，可得出以下运算规律。

设 $\boldsymbol{A}$，$\boldsymbol{B}$ 是任意矩阵，k，l 是任意常数，则

（1）**结合律**　$(kl)\boldsymbol{A}=k(l\boldsymbol{A})=kl\boldsymbol{A}$；　（7-7）

（2）**分配律**　$(k+l)\boldsymbol{A}=k\boldsymbol{A}+l\boldsymbol{A}$，　$k(\boldsymbol{A}+\boldsymbol{B})=k\boldsymbol{A}+k\boldsymbol{B}$。　（7-8）

3. 矩阵的乘法

定义 7-4　设矩阵 $\boldsymbol{A}=(a_{ij})_{m\times s}$，$\boldsymbol{B}=(b_{ij})_{s\times n}$，则由元素

$$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\cdots+a_{is}b_{sj}\quad (i=1,2,\cdots,m;\ j=1,2,\cdots,n)$$

构成的 $m\times n$ 矩阵 $\boldsymbol{C}=(c_{ij})_{m\times n}$ 称为矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 的**乘积**，记作 $\boldsymbol{C}=\boldsymbol{AB}$。

【例 7-17】 已知矩阵 $\boldsymbol{A}=\begin{pmatrix}1 & 0 & -1\\ 2 & 1 & 3\end{pmatrix}$，　$\boldsymbol{B}=\begin{pmatrix}0 & 3 & 2 & 1\\ -2 & -1 & 0 & -1\\ 1 & 4 & -3 & 2\end{pmatrix}$，求 $\boldsymbol{AB}$。

解：

$$\boldsymbol{AB}=\begin{pmatrix}1 & 0 & -1\\ 2 & 1 & 3\end{pmatrix}\begin{pmatrix}0 & 3 & 2\\ -2 & -1 & 0\\ 1 & 4 & -3\end{pmatrix}$$

$$=\begin{pmatrix}1\times0+0\times(-2)+(-1)\times1 & 1\times3+0\times(-1)+(-1)\times4 & 1\times2+0\times0+(-1)\times(-3)\\ 2\times0+1\times(-2)+3\times1 & 2\times3+1\times(-1)+3\times4 & 2\times2+1\times0+3\times(-3)\end{pmatrix}$$

$$=\begin{pmatrix}-1 & -1 & 5\\ 1 & 17 & -5\end{pmatrix}。$$

根据矩阵乘法的定义可知，例 7-17 中 $\boldsymbol{BA}$ 是没有意义的。

关于矩阵的乘法，需要注意以下几点。

（1）矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{B}$ 可以相乘 $\Longleftrightarrow$ $\boldsymbol{A}$ 的列数= $\boldsymbol{B}$ 的行数；

（2）矩阵乘法不满足交换律，即一般情况下，$\boldsymbol{AB}\neq\boldsymbol{BA}$。

【例 7-18】 设

$$\boldsymbol{A}=\begin{pmatrix}a_1\\ a_2\\ \vdots\\ a_n\end{pmatrix},\quad \boldsymbol{B}=\begin{pmatrix}b_1, & b_2, & \cdots, & b_n\end{pmatrix},$$

分别求 $\boldsymbol{AB}$ 和 $\boldsymbol{BA}$。

解：$\boldsymbol{AB}=\begin{pmatrix}a_1\\ a_2\\ \vdots\\ a_n\end{pmatrix}\begin{pmatrix}b_1, & b_2, & \cdots, & b_n\end{pmatrix}=\begin{pmatrix}a_1b_1 & \cdots & a_1b_n\\ \vdots & & \vdots\\ a_nb_1 & \cdots & a_nb_n\end{pmatrix}$，

$$\boldsymbol{BA}=\begin{pmatrix}b_1, & b_2, & \cdots, & b_n\end{pmatrix}\begin{pmatrix}a_1\\ a_2\\ \vdots\\ a_n\end{pmatrix}=(b_1a_1+b_2a_2+\cdots+b_na_n)=b_1a_1+b_2a_2+\cdots+b_na_n。$$

所以，$\boldsymbol{AB}\neq\boldsymbol{BA}$。

（3）$\boldsymbol{AB}=\boldsymbol{O}$ $\not\Rightarrow$ $\boldsymbol{A}=\boldsymbol{O}$ 或者 $\boldsymbol{B}=\boldsymbol{O}$。

【例 7-19】 设 $\boldsymbol{A}=\begin{pmatrix}1 & 1\\ -1 & -1\end{pmatrix}$，　$\boldsymbol{B}=\begin{pmatrix}-1 & 1\\ 1 & -1\end{pmatrix}$，求 $\boldsymbol{AB}$。

解：$\boldsymbol{AB}=\begin{pmatrix}1 & 1\\ -1 & -1\end{pmatrix}\begin{pmatrix}-1 & 1\\ 1 & -1\end{pmatrix}=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}$。

例 7-19 中，$\boldsymbol{A}\neq\boldsymbol{O}$，$\boldsymbol{B}\neq\boldsymbol{O}$ 但是 $\boldsymbol{AB}=\boldsymbol{O}$。

（4）矩阵乘法不满足消去律，即

$$\boldsymbol{AB}=\boldsymbol{AC}\text{ 或 }\boldsymbol{BA}=\boldsymbol{CA}\text{，且 }\boldsymbol{A}\neq\boldsymbol{O}\ \not\Rightarrow\ \boldsymbol{B}=\boldsymbol{C}\text{。}$$

根据矩阵乘法的定义，可以得到以下性质（证明略）。

（1）结合律：$(\boldsymbol{AB})\boldsymbol{C}=\boldsymbol{A}(\boldsymbol{BC})$；　（7-9）

（2）分配律：$\boldsymbol{A}(\boldsymbol{B}+\boldsymbol{C})=\boldsymbol{AB}+\boldsymbol{AC}$，$(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{C}=\boldsymbol{AC}+\boldsymbol{BC}$；　（7-10）

（3）对任意常数 k，有 $k(\boldsymbol{AB})=(k\boldsymbol{A})\boldsymbol{B}=\boldsymbol{A}(k\boldsymbol{B})$；　（7-11）

（4）$\boldsymbol{E}_m$，$\boldsymbol{E}_n$ 为单位矩阵，对任意矩阵 $\boldsymbol{A}_{m\times n}$ 有

$$\boldsymbol{E}_m\boldsymbol{A}_{m\times n}=\boldsymbol{A}_{m\times n}\boldsymbol{E}_n=\boldsymbol{A}_{m\times n}\text{，}\qquad(7\text{-}12)$$

特别地，若 $\boldsymbol{A}$ 为方阵，则 $\boldsymbol{E}_n\boldsymbol{A}_{n\times n}=\boldsymbol{A}_{n\times n}\boldsymbol{E}_n=\boldsymbol{A}_{n\times n}$。　（7-13）

与数字的幂类似，也可以定义方阵的幂。

定义 7-5　设 $\boldsymbol{A}$ 为 n 阶方阵，定义：

$$\boldsymbol{A}^k=\underbrace{\boldsymbol{A}\boldsymbol{A}\cdots\boldsymbol{A}}_{k\text{个}\boldsymbol{A}\text{相乘}}\qquad(7\text{-}14)$$

称为方阵 $\boldsymbol{A}$ 的 k 次幂，并且规定 $\boldsymbol{A}^0=\boldsymbol{E}$。

由于矩阵的乘法满足结合律，于是

$$\boldsymbol{A}^k\boldsymbol{A}^l=\boldsymbol{A}^{k+l},\ \left(\boldsymbol{A}^k\right)^l=\boldsymbol{A}^{kl}\qquad(k,l\text{为非负整数})\text{。}\qquad(7\text{-}15)$$

由于矩阵的乘法不满足交换律，所以

$$(\boldsymbol{AB})^k\neq\boldsymbol{A}^k\boldsymbol{B}^k\text{。}\qquad(7\text{-}16)$$

4. 矩阵的转置

定义 7-6　设矩阵

$$\boldsymbol{A}=\begin{pmatrix}a_{11} & \cdots & a_{1n}\\ \vdots & & \vdots\\ a_{m1} & \cdots & a_{mn}\end{pmatrix},$$

把矩阵的行与列互换得到的 $n\times m$ 矩阵，称为 $\boldsymbol{A}$ 的**转置矩阵**（图 7-5），记作 $\boldsymbol{A}^{\mathrm{T}}$，即

$$\boldsymbol{A}^{\mathrm{T}}=\begin{pmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & & \vdots \\ a_{1n} & \cdots & a_{mn} \end{pmatrix}。\tag{7-17}$$

例如，$\boldsymbol{A}=\begin{pmatrix} 1 & 0 & -1 \\ 2 & 1 & 3 \end{pmatrix}$，则 $\boldsymbol{A}^{\mathrm{T}}=\begin{pmatrix} 1 & 2 \\ 0 & 1 \\ -1 & 3 \end{pmatrix}$；再如图 7-5 中所示的矩阵转置。

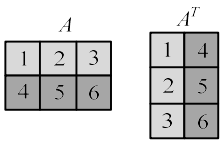


图 7-5　矩阵的转置

显然，$\boldsymbol{A}$ 与 $\boldsymbol{A}^{\mathrm{T}}$ 互为转置矩阵。同样，行、列向量互为转置向量。由于列向量的书写不方便，所以经常把列向量写作行向量的转置：

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}=(a_1,a_2,\cdots,a_n)^{\mathrm{T}}。\tag{7-18}$$

矩阵转置的运算规律：

（1）$\left(\boldsymbol{A}^{\mathrm{T}}\right)^{\mathrm{T}}=\boldsymbol{A}$；　　（7-19）

（2）$\left(\boldsymbol{A}+\boldsymbol{B}\right)^{\mathrm{T}}=\boldsymbol{A}^{\mathrm{T}}+\boldsymbol{B}^{\mathrm{T}}$；　　（7-20）

（3）$\left(k\boldsymbol{A}\right)^{\mathrm{T}}=k\boldsymbol{A}^{\mathrm{T}}$，$k$ 为实数；　　（7-21）

（4）$\left(\boldsymbol{AB}\right)^{\mathrm{T}}=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}$，$(\boldsymbol{A}_1\boldsymbol{A}_2\cdots\boldsymbol{A}_k)^{\mathrm{T}}=\boldsymbol{A}_k^{\mathrm{T}}\boldsymbol{A}_{k-1}^{\mathrm{T}}\cdots\boldsymbol{A}_1^{\mathrm{T}}$。　　（7-22）

【例 7-20】 设 $\boldsymbol{A}=\begin{pmatrix} -1 & 3 & 1 \\ -2 & 0 & 1 \\ 1 & 2 & 0 \end{pmatrix}$，$\boldsymbol{B}=\begin{pmatrix} -1 & 1 & -2 \\ -1 & -2 & 0 \\ 0 & 1 & 1 \end{pmatrix}$，求 $(\boldsymbol{AB})^{\mathrm{T}}$，$\boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}$，$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}$。

解：　$\boldsymbol{AB}=\begin{pmatrix} -2 & -6 & 3 \\ 2 & -1 & 5 \\ -3 & -3 & -2 \end{pmatrix}$，所以 $\left(\boldsymbol{AB}\right)^{\mathrm{T}}=\begin{pmatrix} -2 & 2 & -3 \\ -6 & -1 & -3 \\ 3 & 5 & -2 \end{pmatrix}$；

$$\boldsymbol{A}^{\mathrm{T}}=\begin{pmatrix} -1 & -2 & 1 \\ 3 & 0 & 2 \\ 1 & 1 & 0 \end{pmatrix},\ \boldsymbol{B}^{\mathrm{T}}=\begin{pmatrix} -1 & -1 & 0 \\ 1 & -2 & 1 \\ -2 & 0 & 1 \end{pmatrix},$$

所以

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}=\begin{pmatrix}-3 & 5 & -1\\ -7 & -3 & 2\\ 0 & -3 & 1\end{pmatrix},\quad \boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}=\begin{pmatrix}-2 & 2 & -3\\ -6 & -1 & -3\\ 3 & 5 & -2\end{pmatrix}。$$

由例 7-20 知，$(\boldsymbol{AB})^{\mathrm{T}}=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\neq\boldsymbol{A}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}}$。　　（7-23）

5. 矩阵的哈达马积

普通意义上的矩阵乘积比较复杂，在某些应用中还需要用到同阶矩阵的对应元素的乘积，称之为矩阵的“哈达马积”(Hadamard product)。

定义 7-7　设两个 $m\times n$ 矩阵 $\boldsymbol{A}=(a_{ij})_{m\times n}$，$\boldsymbol{B}=(b_{ij})_{m\times n}$，称矩阵 $\boldsymbol{C}=(a_{ij}\times b_{ij})_{m\times n}$ 为 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的**哈达马积**，记作 $\boldsymbol{C}=\boldsymbol{A}*\boldsymbol{B}$。

【例 7-21】 设 $\boldsymbol{A}=\begin{pmatrix}3 & 1 & -2\\ 0 & -1 & 4\end{pmatrix}$，$\boldsymbol{B}=\begin{pmatrix}1 & 0 & -5\\ 3 & -4 & 2\end{pmatrix}$，求 $\boldsymbol{A}*\boldsymbol{B}$。

解：$\boldsymbol{A}*\boldsymbol{B}=\begin{pmatrix}3\times1 & 1\times0 & -2\times(-5)\\ 0\times3 & -1\times(-4) & 4\times2\end{pmatrix}=\begin{pmatrix}3 & 0 & 10\\ 0 & 4 & 8\end{pmatrix}$。

根据哈达马积的定义，可以得出以下运算规律。

（1）交换律：$\boldsymbol{A}*\boldsymbol{B}=\boldsymbol{B}*\boldsymbol{A}$；　　（7-24）

（2）结合律：$(\boldsymbol{A}*\boldsymbol{B})*\boldsymbol{C}=\boldsymbol{A}*(\boldsymbol{B}*\boldsymbol{C})$；　　（7-25）

（3）分配率：$\boldsymbol{A}*(\boldsymbol{B}+\boldsymbol{C})=\boldsymbol{A}*\boldsymbol{B}+\boldsymbol{A}*\boldsymbol{C}$；　$(\boldsymbol{A}+\boldsymbol{B})*\boldsymbol{C}=\boldsymbol{A}*\boldsymbol{C}+\boldsymbol{B}*\boldsymbol{C}$。　　（7-26）

注意：

（1）计算两个矩阵的哈达马积，两矩阵必须同型；

（2）$\boldsymbol{A}*\boldsymbol{B}=\boldsymbol{0}\not\Longrightarrow\boldsymbol{A}=\boldsymbol{0}$ 或者 $\boldsymbol{B}=\boldsymbol{0}$，例如，$\boldsymbol{A}=\begin{pmatrix}0 & 0\\ -2 & 1\end{pmatrix}\neq\boldsymbol{0}$，$\boldsymbol{B}=\begin{pmatrix}-1 & 1\\ 0 & 0\end{pmatrix}\neq\boldsymbol{0}$，但是 $\boldsymbol{A}*\boldsymbol{B}=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}$。

7.2.2　使用 NumPy 进行矩阵运算

使用 Numpy 可以方便地实现矩阵的各种运算。

1. 矩阵运算

NumPy 中的常见矩阵运算如表 7-3 所示。

表 7-3　NumPy 中的常见矩阵运算

运　　算	等 价 函 数	含　　义
A+B	np.add()	矩阵加法
A−B	np.subtract(A,B)	矩阵减法
A@B	np.dot(A,B) A.dot(B)	矩阵乘法
A.T	A.transpose()	矩阵转置
2*A	np.dot(2,A) A.dot(2)	数与矩阵乘法

【例 7-22】矩阵的基本运算。（代码：ch7 矩阵与数字图像处理\7.2 矩阵的运算\例 7-22）

```
# 矩阵的运算
1. import numpy as np
2. A=np.array([[1,2],[3,4]])
3. print(A)
4. B=np.array([[2,1],[0,5]])
5. print(B)
6. print(A+B)
7. print(np.dot(A,B))
8. print(A.T)
9. print(2*A)
```

运行结果：

```
[[1 2]
 [3 4]]

[[2 1]
 [0 5]]

[[3 3]
 [3 9]]

[[ 2 11]
 [ 6 23]]

[[1 3]
 [2 4]]

[[2 4]
 [6 8]]
```

2. 对应元素之间的运算

除了数学中定义的矩阵运算外，为了便于使用，NumPy 中还定义了矩阵对应元素的运算，如表 7-4 所示。

表 7-4　NumPy 中矩阵对应元素的运算

运　　算	含　　义
X*Y 或 np.multiply(x,y)	对应元素相乘（哈达马积）
X/Y 或 np.true_divede(x,y)	对应元素相除
X//Y 或 np.floor_divide(x,y)	对应元素相除后，返回值取整
X**Y 或 np.power(x,y)	对应元素进行幂运算
X**2	对矩阵 x 的每个元素进行幂运算
X%Y 或 np.remainder(x,y)	对应元素相除后取余数

【例 7-23】 矩阵对应元素的运算。（代码：ch7 矩阵与数字图像处理\7.2 矩阵的运算\例 7-23）

```
# 矩阵对应元素的运算
1. import numpy as np
2. X=np.arraY([[1,2,3],[4,5,6]])
3. print(X)
4. print('-'*20)
5. Y=np.array([[2,1,3],[2,5,3]])
6. print(Y)
7. print('-'*20)
8. print(X+Y)
9. print('-'*20)
10. print(X*Y)
11. print('-'*20)
12. print(X/Y)
13. print('-'*20)
14. print(X//Y)
15. print('-'*20)
16. print(X**Y)
17. print('-'*20)
18. print(X%Y)
```

运行结果：

```
[[1 2 3]
 [4 5 6]]
--------------------
[[2 1 3]
```

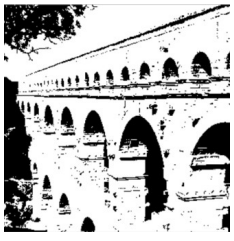
```
 [2 5 3]]
--------------------
[[ 3  3  6]
 [ 6 10  9]]
--------------------
[[ 2  2  9]
 [ 8 25 18]]
-------------------
[[0.5 2.  1. ]
 [2.  1.  2. ]]
--------------------
[[0 2 1]
 [2 1 2]]
--------------------
[[   1    2   27]
 [  16 3125  216]]
--------------------
[[1 0 0]
 [0 0 0]]
```

7.3　实战案例：矩阵在数字图像处理中的应用

7.3.1　图像基础

心理学研究表明，人类获取的信息 83%来自视觉。20 世纪 50 年代以来，基于计算机的数字图像处理技术发展迅速，在安防、医学、产品检测等众多领域得到了广泛应用。那么，什么是数字图像呢？

数字图像，又称“数码图像”或“数位图像”，是以二维数字组形式表示的二维图像。根据数字图像在计算机中表示方法的不同，分为二值图像、灰度图像和 RGB 图像等，如图 7-6 所示。


（a）二值图像


（b）灰度图像

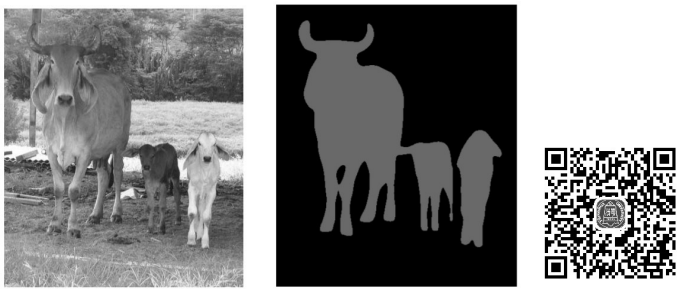

（c）彩色图像

图 7-6　各种数字图像

数字图像处理（Digital Image Processing，DIP）又称“计算机图像处理”，它是指将图像信号转换成数字信号并利用计算机对其进行处理的过程，如图 7-7～图 7-9 所示。

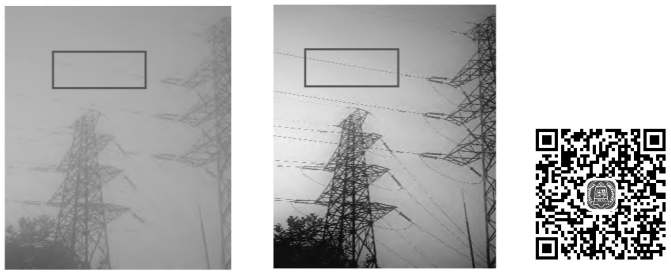
（a）原始图像　　（b）提取图像轮廓

图 7-7　图像轮廓检测



（a）原始图像　　（b）分割后的图像

图 7-8　图像分割①



（a）原始图像（有雾气）　　（b）增强后的图像

图 7-9　图像增强②

① Zhao H , Shi J , Qi X , et al. Pyramid scence parsing network[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2017.

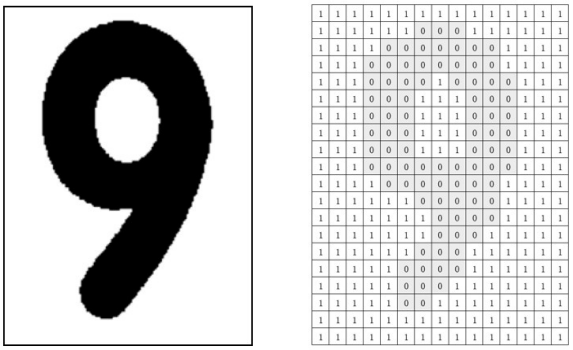
② 谢有庆，何涛，邱捷. 基于分数阶微分的电力系统有雾图像增强研究[J/OL]. 广东电力，2020，27（9）1-9[2020-10-12]. http://kns.cnki.net/kcms/detail/44.1420.TM. 20201009. 0934.036.html.

7.3.2　数字图像的矩阵表示

在计算机中，一个二维图像通常用一个 $M \times N$ 的矩阵表示，其中 M 表示图像的行数，N 表示图像的列数。矩阵的每个元素对应图像的 1 像素，元素的值表示像素的灰度值或者颜色值。在不同的图像类型中，元素的取值范围也不同。

1. 二值图像

在二值图像中，像素的值取 0 或者 1，其中 0 表示黑色，1 表示白色。二值图像一般用来描述文字或者简单图形，占用空间少，但是颜色区分度小，只能描述轮廓，如图 7-10 所示。



（a）原始二值图像　　（b）二值图像的矩阵

图 7-10　数字“9”的二值图像及对应矩阵

2. 灰度图像

灰度图像的像素值取值范围为 0～255，其中 0 表示黑色，255 表示白色，1～254 表示不同的深浅灰色。相比二值图像，灰度图像增加了颜色的种类，但是缺乏色彩，如图 7-11 所示。

3. RGB 图像

也称为“真彩色图像”，一幅 RGB 图像由 3 个大小相同的二维数组构成，分别代表各像素中 R（红色）、G（绿色）、B（蓝色）的亮度值。每种颜色的取值范围为 $[0,255]$，如图 7-12 和图 7-13 所示。

（a）原始灰度图像

106	166	113	130	124	135	150	202	212	107	159	159	159	71	89
104	165	112	128	130	144	182	196	199	227	163	117	62	71	154
115	172	115	117	121	158	179	181	181	192	186	195	68	83	161
110	172	115	153	118	153	165	162	162	183	195	158	70	153	161
117	171	114	108	123	139	102	108	169	191	127	169	77	162	161
115	176	114	126	125	99	79	111	168	197	97	75	117	163	161
115	175	118	133	140	84	94	191	179	196	62	79	148	163	149
107	179	119	175	86	92	169	85	124	99	67	68	161	158	145
93	180	154	97	81	66	66	177	158	172	75	63	136	147	190
84	177	126	88	74	164	98	161	136	162	85	112	134	137	200
87	180	132	95	96	61	109	154	158	143	91	149	152	175	201
82	180	81	135	140	67	78	147	160	67	92	154	151	202	203
69	176	72	138	121	62	75	136	154	161	98	126	129	200	168
109	177	72	71	125	61	70	150	156	168	119	150	136	204	119
172	175	70	73	144	126	116	147	154	178	209	143	129	160	105
176	178	64	73	131	66	131	154	150	168	209	96	150	87	103
160	181	78	87	89	69	139	151	155	160	200	107	126	114	118

（b）灰度图像的矩阵

图 7-11　灰度图像及对应矩阵

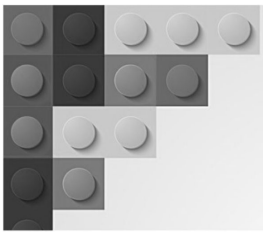


图 7-12　彩色图像

11	15	8	195	196	198	253	254	252	252	252	251	252	254	252
11	28	8	195	218	195	245	241	253	252	240	252	244	237	253
9	28	15	191	210	151	237	237	221	240	237	248	237	237	251
12	28	15	198	207	183	253	241	240	253	236	252	253	221	252
8	12	12	203	201	202	221	220	218	39	38	42	239	238	243
0	32	1	196	222	195	244	238	243	11	27	13	229	231	230
0	33	11	199	213	183	245	244	243	24	25	10	230	231	231
1	33	9	197	204	161	226	241	217	24	18	13	231	232	232
0	7	5	201	176	196	239	207	236	15	15	8	232	232	233
0	0	2	253	253	253	253	254	247	229	231	231	232	233	234
1	33	3	249	242	254	243	240	253	230	231	232	233	234	235
0	32	8	244	235	196	241	236	226	231	232	233	234	235	236
0	26	12	252	240	225	253	244	239	231	233	234	235	237	238
122	126	127	251	251	255	232	234	232	232	233	234	236	236	239
195	220	193	240	243	242	226	230	231	233	234	235	237	238	239
192	214	165	222	244	232	229	232	232	233	235	237	237	239	240
197	208	148	237	244	194	230	231	233	234	236	238	238	240	241
197	178	191	240	198	240	230	232	233	235	236	237	239	240	242
197	194	198	226	229	229	231	232	234	235	236	239	240	241	241
195	216	195	224	228	229	231	232	234	236	237	239	240	242	243

（a）红色分量矩阵

143	141	145	22	23	21	205	206	206	206	206	205	206	204	207
141	157	145	24	27	25	208	201	205	221	200	203	215	198	204
139	153	101	19	23	22	208	192	184	203	191	202	201	190	206
142	149	131	21	16	24	205	189	199	205	184	203	205	163	206
115	119	120	30	30	30	104	110	104	130	131	125	228	230	227
103	146	102	20	29	23	100	118	99	141	161	143	231	231	232
98	141	92	22	26	24	128	112	103	156	154	142	231	232	232
101	127	88	22	21	24	97	104	98	155	150	141	232	233	233
104	106	106	21	24	23	101	96	104	141	129	143	233	233	234
103	101	102	205	204	205	205	205	208	230	232	232	233	234	235
102	148	103	200	202	204	206	200	205	231	232	233	234	235	236
98	135	76	206	196	161	208	194	185	232	233	234	235	236	237
104	122	100	205	186	190	205	192	198	232	234	235	236	237	238
61	59	58	155	153	152	212	212	213	233	234	235	237	237	239
23	35	23	100	130	100	230	231	232	234	235	236	237	238	239
21	27	21	106	114	100	230	233	233	234	236	237	238	239	240
22	22	22	100	101	89	231	232	234	235	237	238	238	240	241
22	23	24	100	100	102	231	233	234	236	237	238	239	240	242
22	23	20	226	229	230	232	233	235	236	237	239	240	241	241
23	29	22	228	229	230	232	233	235	237	238	239	240	242	243

（b）绿色分量矩阵

8	5	5	6	6	5	7	8	9	9	7	8	9	9	2
7	12	5	4	9	2	6	17	11	16	17	12	15	15	5
3	8	4	0	6	3	6	13	7	11	17	12	17	16	15
8	17	8	3	8	3	11	25	13	9	22	13	11	29	13
157	153	149	13	12	20	0	0	0	0	0	0	182	181	176
195	209	192	6	12	3	3	32	2	5	14	9	230	231	231
191	206	157	4	9	3	35	37	6	8	9	5	233	234	234
195	199	145	3	3	5	5	36	8	13	5	4	234	235	235
192	187	194	6	1	6	0	4	3	6	5	4	235	235	236
195	191	190	11	13	11	11	14	9	232	234	234	235	236	237
194	208	191	10	16	9	4	14	11	233	234	235	236	237	238
190	201	123	9	17	7	9	12	5	234	235	236	237	238	239
195	199	172	11	15	10	11	8	12	234	236	237	238	239	238
95	102	103	16	18	19	117	111	110	235	236	237	239	239	241
1	17	8	2	38	0	231	233	234	236	237	238	239	240	241
1	10	10	7	36	15	232	235	235	236	238	239	240	241	242
1	7	10	4	41	8	233	234	236	237	239	240	240	242	243
1	0	8	2	1	1	233	235	236	238	239	240	241	242	244
1	3	8	228	229	234	234	235	237	238	239	239	242	241	243
1	12	6	229	231	232	234	235	237	239	240	241	242	244	243

（c）蓝色分量矩阵

图 7-13　真彩色图像的颜色矩阵

7.3.3　矩阵运算实现图像处理

由于数字图像用矩阵表示，对图像的各种处理就表现为对图像矩阵进行各种运算。

1. 矩阵加法实现图像叠加效果

将两个矩阵相加，即对应点像素的值相加，可以实现两幅图像的叠加（若对应点像素值的和超过 255，则以 255 为除数进行取余运算）。

【例 7-24】 两个矩阵相加实现图像叠加效果，如图 7-14 所示。（代码：ch7 矩阵与数字图像处理\7.3 实战案例：矩阵在数字图像处理中的应用\例 7-24）

```
1. import cv2
2.
3. img1 = cv2.imread(r'figures/eagle.png')    # 读取图片 1
4. cv2.imshow('img1',img1)                    # 显示图片 1
5. img2 = cv2.imread(r'figures/frag.png')     # 读取图片 2
```

```
6. cv2.imshow('img2',img2)      # 显示图片 2
7. result=(img1+img2)%255       # 将两张图片矩阵相加，结果变换在[0,255]的范围内
8. cv2.imshow('img1+img2',result) # 显示矩阵相加后的图片
9. cv2.waitKey(0)               # 设置 waitKey(0) , 代表按任意键可继续
10. cv2.destroyAllWindows()     # 关闭窗口
```

运行结果：

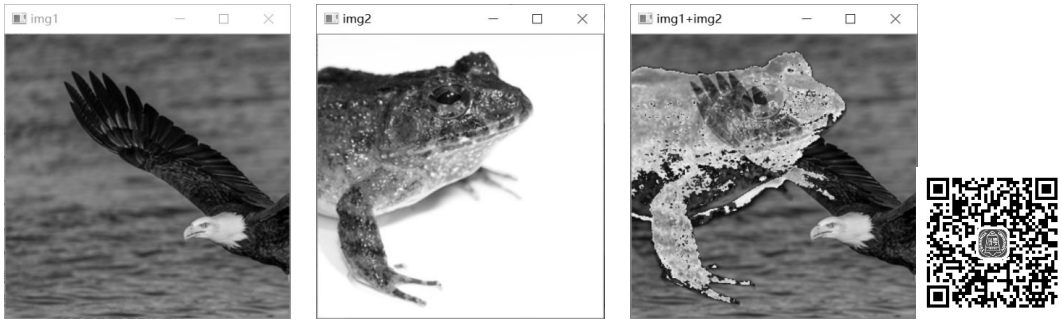


图 7-14　矩阵加法实现图像叠加效果

2. 矩阵减法实现底片效果与图像分离效果

将矩阵的每个对应点像素值用 255 去减，则会实现底片的效果。而用混合图像减去背景，还可以实现图像分离效果。

【例 7-25】 矩阵减法实现底片效果与图像分离效果，如图 7-15 和图 7-16 所示。（代码：ch7 矩阵与数字图像处理\7.3 实战案例：矩阵在数字图像处理中的应用\例 7-25）

```
1. import cv2
2.
3. img = cv2.imread(r'figures/eagle.png')     # 读取图片
4. cv2.imshow('original image',img)           # 显示图片
5. cv2.imshow('255-img',255-img)              # 显示 255 减去图像矩阵后的效果
6.
7. img1 = cv2.imread('figures/img1.png')      # 读取图像
8. img2 = cv2.imread('figures/img2.png')      # 读取图像
9. cv2.imshow('img1',img1)                    # 显示图片
10. cv2.imshow('img2',img2)                   # 显示图片
11. cv2.imshow("img1-img2",img1-img2) # 显示两个图像矩阵相减后对应的图像
12. cv2.waitKey(0)                          # 设置 waitKey(0) , 代表按任意键可继续
13. cv2.destroyAllWindows()                   # 关闭窗口
```

运行结果：

图 7-15　矩阵减法实现底片效果

（a）coin and tyre　　（b）coin　　（c）tyre

图 7-16　矩阵减法实现图像分离效果

3. 数与矩阵相乘改变图像亮度

将矩阵乘以一个常数可以增大像素值，从而提高图像的亮度。这一效果也可以通过对图像的每像素值加上同一个常数实现。

【例 7-26】 数与矩阵相乘改变图像亮度，如图 7-17 所示。（代码：ch7 矩阵与数字图像处理\7.3 实战案例：矩阵在数字图像处理中的应用\例 7-26）

```
1. import cv2
2.
3. img = cv2.imread('figures/dark.png') # 读取图像
4. cv2.imshow('original image',img)     # 显示原图像
5. cv2.imshow('result',img*2)           # 将原图像的矩阵乘以常数 2
6. cv2.waitKey(0)                       # 设置 waitKey(0) , 代表按任意键可继续
7. cv2.destroyAllWindows()              # 关闭窗口
```

运行结果：

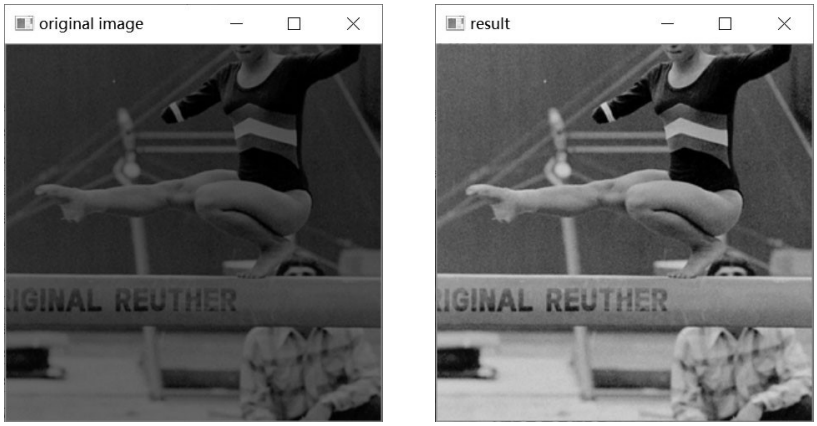


图 7-17　数与矩阵相乘改变图像亮度

4. 图像翻转

矩阵的转置可以实现矩阵中行与列互换，对应图像也具有同样的效果。

【例 7-27】 利用矩阵转置运算实现图像翻转，如图 7-18 所示。（代码：ch7 矩阵与数字图像处理\7.3 实战案例：矩阵在数字图像处理中的应用\例 7-27）

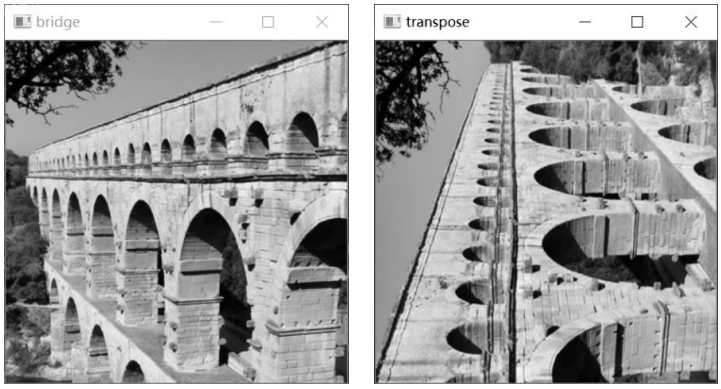


图 7-18　矩阵转置实现图片翻转效果

```
1. # 导入 opencv 模块
2. import cv2
3. # 以灰度图像的形式打开图片文件
4. img = cv2.imread('figures/bridge.jpg',cv2.IMREAD_GRAYSCALE)
5. cv2.imshow('original image',img)          # 显示原图像
```

```
print(img.shape)                                    # 显示原图像的规格
cv2.imshow(r'transposed image',img.T)               # 显示转置后的图像
cv2.waitKey(0)                                  # 设置 waitKey(0) , 代表按任意键可继续
cv2.destroyAllWindows()                             # 关闭窗口
```

运行结果：

```
(400, 400)
```

7.4　矩阵的初等变换

在利用矩阵解决问题的过程中，例如求解线性方程组，需要对矩阵做一些变换，从而得到最终结论。归结起来，对矩阵的变换有三种类型，称为矩阵的初等变换。

定义 7-8　对一个矩阵 $\boldsymbol{A}=(a_{ij})_{m\times n}$ 实施以下三种类型的变换，称为矩阵的**初等行（列）变换**，统称为矩阵的**初等变换**：

（1）互换 $\boldsymbol{A}$ 中两行（列）；

（2）用一个非零常数 k 乘 $\boldsymbol{A}$ 的某一行（列）；

（3）用一个常数 k 乘 $\boldsymbol{A}$ 的某一行（列），加到另一行（列）上。

由于经过初等变换后两个矩阵不相等，因而用箭头“$\rightarrow$”进行连接，而不能用“$=$”。

下面以初等行变换为例进行解释。

设 $\boldsymbol{A}=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}=\begin{pmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \vdots \\ \boldsymbol{\alpha}_m \end{pmatrix}$，其中 $\boldsymbol{\alpha}_i=(a_{i1},a_{i2},\cdots,a_{in}),\ i=1,2,\cdots,m$。

（1）互换 $\boldsymbol{A}$ 中的 i,j 两行：

$$\boldsymbol{A}=\begin{pmatrix} \vdots \\ \boldsymbol{\alpha}_i \\ \vdots \\ \boldsymbol{\alpha}_j \\ \vdots \end{pmatrix}\rightarrow \boldsymbol{B}=\begin{pmatrix} \vdots \\ \boldsymbol{\alpha}_j \\ \vdots \\ \boldsymbol{\alpha}_i \\ \vdots \end{pmatrix};$$

（2）用一个非零常数 k 乘 $\boldsymbol{A}$ 的第 i 行：

$$\boldsymbol{A}=\begin{pmatrix} \vdots \\ \boldsymbol{\alpha}_i \\ \vdots \end{pmatrix}\rightarrow \boldsymbol{B}=\begin{pmatrix} \vdots \\ k\boldsymbol{\alpha}_i \\ \vdots \end{pmatrix},\ k\neq 0;$$