

目录

MATLAB 中的高维数组	2
(1) 创建高维数组	3
(2) 引用高维数组	6
(3) 修改和删除高维数组	7
(4) 高维数组中与维度相关的三个函数	9
(5) 调用函数对高维数组进行计算	12
(6) 高维数组的算术运算和关系运算	14

讲解视频：可以在 bilibili 搜索“MATLAB 中的高维数组——入门知识讲解”。

在学习本节课程之前，**确保你已经具备了一定的 MATLAB 基础**，这将有助于你更好地理解 and 掌握高维数组的概念和操作。

如果你没有 MATLAB 基础的话，可以先看我的 MATLAB 入门课程：

<https://www.bilibili.com/video/BV1dN4y1Q7Kt>



MATLAB教程新手入门篇（数学建模清风主讲，适合零基...

2023年09月13日 21:15:14

当前字幕: 9

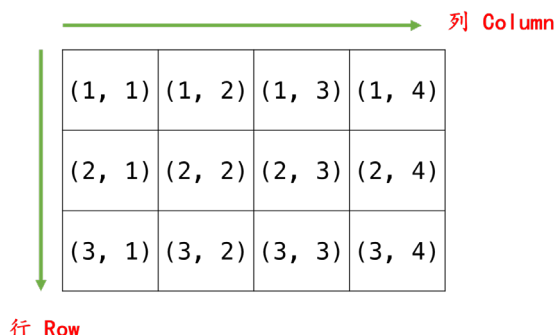
12.0万 4209 7121 1175 4238 1.1万 854

建议你先观看入门课程的第三章，然后再继续学习这个高维数组的课程，这样能够更好地建立起你的 MATLAB 知识体系。

MATLAB 中的高维数组——入门知识

MATLAB 中的高维数组（又称多维数组）是指具有两个以上维度的数组。

在矩阵中，两个维度由行和列表示：

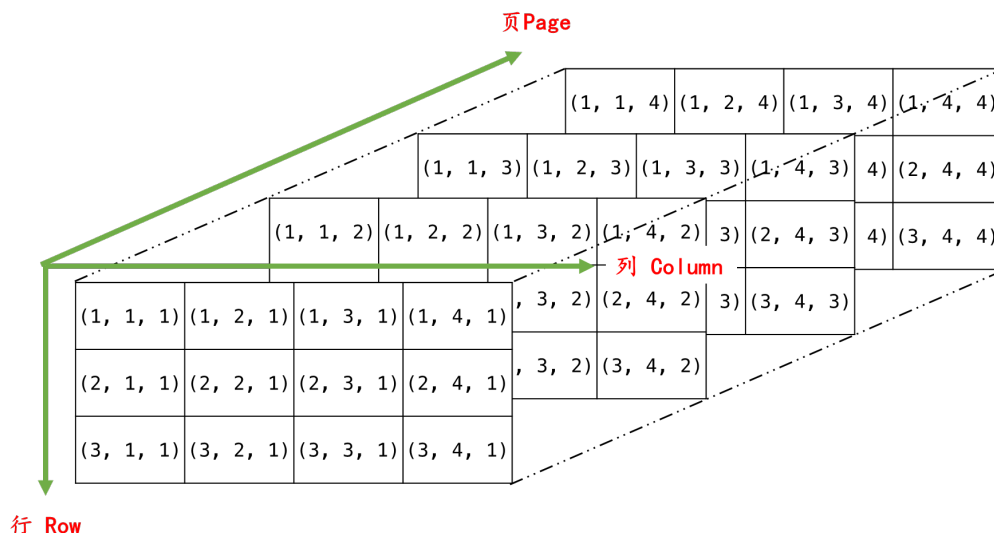


A 3x4 matrix is shown with row and column indices. A green arrow points down along the left side, labeled '行 Row' in red. Another green arrow points right along the top, labeled '列 Column' in red. The matrix elements are as follows:

(1, 1)	(1, 2)	(1, 3)	(1, 4)
(2, 1)	(2, 2)	(2, 3)	(2, 4)
(3, 1)	(3, 2)	(3, 3)	(3, 4)

矩阵中的每个元素由两个下标（即行索引和列索引）来定义。

高维数组是二维矩阵的扩展，并使用额外的下标进行索引。例如，三维数组使用三个下标。前两个维度就像一个矩阵，而第三个维度表示元素的页数或张数（有时也称为切片）：



A 3D array is shown with row, column, and page indices. A green arrow points down along the left side, labeled '行 Row' in red. Another green arrow points right along the top, labeled '列 Column' in red. A third green arrow points diagonally upwards from the bottom-left, labeled '页 Page' in red. The array is represented as a stack of 4 pages. The elements on each page are as follows:

Page	Row \ Column	1	2	3	4
1	1	(1, 1, 1)	(1, 2, 1)	(1, 3, 1)	(1, 4, 1)
1	2	(2, 1, 1)	(2, 2, 1)	(2, 3, 1)	(2, 4, 1)
1	3	(3, 1, 1)	(3, 2, 1)	(3, 3, 1)	(3, 4, 1)
2	1	(1, 1, 2)	(1, 2, 2)	(1, 3, 2)	(1, 4, 2)
2	2	(2, 1, 2)	(2, 2, 2)	(2, 3, 2)	(2, 4, 2)
2	3	(3, 1, 2)	(3, 2, 2)	(3, 3, 2)	(3, 4, 2)
3	1	(1, 1, 3)	(1, 2, 3)	(1, 3, 3)	(1, 4, 3)
3	2	(2, 1, 3)	(2, 2, 3)	(2, 3, 3)	(2, 4, 3)
3	3	(3, 1, 3)	(3, 2, 3)	(3, 3, 3)	(3, 4, 3)
4	1	(1, 1, 4)	(1, 2, 4)	(1, 3, 4)	(1, 4, 4)
4	2	(2, 1, 4)	(2, 2, 4)	(2, 3, 4)	(2, 4, 4)
4	3	(3, 1, 4)	(3, 2, 4)	(3, 3, 4)	(3, 4, 4)

那什么时候我们会用到高维数组呢？下面给大家列举一些常见的高维数组的应用领域：

图像处理：在图像处理中，高维数组常用于存储和操作图像数据。例如彩色图像通常以三维数组存储，其中两个维度表示图像的高度和宽度，第三个维度表示颜色通道（如 RGB）。

深度学习：深度学习领域中，高维数组（又称张量）用于表示和处理神经网络的输入、输出及其权重。例如，卷积神经网络（CNN）处理的图像数据就是以高维数组的形式输入的。

量子计算：在量子计算中，高维数组用于模拟和处理量子态。量子计算中的量子比特（qubit）状态可以用高维向量表示，而量子门操作可以用高维矩阵表示。

仿真优化：高维数组可以用于建立物理和工程系统的仿真模型，并对其进行优化。例如，利用计算流体力学模型对飞机、汽车等运动物体的流场进行仿真时，可以使用高维数组来存储和处理流场数据。通过对这些数据进行优化处理，我们可以设计出更高效的运动物体。

接下来，我们将为大家介绍高维数组的操作方法。值得一提的是，在日常实践中，三维以上的数组并不常见，其应用主要集中在某些特定领域，例如医学图像处理。因此，我们后续主要讲解三维数组。

（1）创建高维数组

高维数组的创建可以通过多种方式实现，下面我们介绍一些常用的方法。

方法一：使用函数创建，例如常见的 `zeros`、`ones`、`rand`、`randi` 函数。

<code>x1 = ones(3,4,2)</code>	<code>x1(:,:,1) =</code> 1 1 1 1 1 1 1 1 1 1 1 1 <code>x1(:,:,2) =</code> 1 1 1 1 1 1 1 1 1 1 1 1
<code>x2 = randi([1,10],3,4,2)</code>	<code>x2(:,:,1) =</code> 6 4 1 8 1 9 2 7 3 1 7 5 <code>x2(:,:,2) =</code> 6 2 4 1 3 7 7 10 8 2 8 8

为了方便理解，我们可以使用下面的图形表示 `x2`：

6	4	1	8
1	9	2	7
3	1	7	5

6	2	4	1
3	7	7	10
8	2	8	8

我们可以查看三维数组的大小：

<code>size(x2)</code> % 返回行、列和页数	3 4 2
<code>numel(x2)</code> % 元素的总数	24
<code>size(ones(5,6,10))</code> % 10 页的三维数组	5 6 10

方法二：先创建二维矩阵，然后再进行扩展。

% 假如我们还是想得到上面的 x2，我们可以首先定义一个 3×4 的矩阵，作为三维数组中的第一页 x2 = [6 4 1 8;1 9 2 7;3 1 7 5]	<table><tr><td>6</td><td>4</td><td>1</td><td>8</td></tr><tr><td>1</td><td>9</td><td>2</td><td>7</td></tr><tr><td>3</td><td>1</td><td>7</td><td>5</td></tr></table>	6	4	1	8	1	9	2	7	3	1	7	5																				
6	4	1	8																														
1	9	2	7																														
3	1	7	5																														
% 现在添加第二页。要完成此操作，可将另一个 3×4 的矩阵赋给第三个维度中的索引值 2。 语法 x2(:,:,2) 在第一个和第二个维度中使用冒号，以在其中包含赋值表达式右侧的所有行和所有列。 x2(:,:,2) = [6 2 4 1;3 7 7 10;8 2 8 8]	<table><tr><td colspan="4">x2(:,:,1) =</td></tr><tr><td>6</td><td>4</td><td>1</td><td>8</td></tr><tr><td>1</td><td>9</td><td>2</td><td>7</td></tr><tr><td>3</td><td>1</td><td>7</td><td>5</td></tr><tr><td colspan="4">x2(:,:,2) =</td></tr><tr><td>6</td><td>2</td><td>4</td><td>1</td></tr><tr><td>3</td><td>7</td><td>7</td><td>10</td></tr><tr><td>8</td><td>2</td><td>8</td><td>8</td></tr></table>	x2(:,:,1) =				6	4	1	8	1	9	2	7	3	1	7	5	x2(:,:,2) =				6	2	4	1	3	7	7	10	8	2	8	8
x2(:,:,1) =																																	
6	4	1	8																														
1	9	2	7																														
3	1	7	5																														
x2(:,:,2) =																																	
6	2	4	1																														
3	7	7	10																														
8	2	8	8																														

再来看一个例子：

<pre>x3 = reshape(1:12,3,4); x3(1,1,3) = 666 % MATLAB 会自动扩展数组的大小以适应这个新值 % 其他位置的元素会自动用 0 填充</pre>	<pre>x3(:,:,1) = 1 4 7 10 2 5 8 11 3 6 9 12 x3(:,:,2) = 0 0 0 0 0 0 0 0 0 0 0 0 x3(:,:,3) = 666 0 0 0 0 0 0 0 0 0 0 0</pre>
--	--

因此，第二种方法的核心思想是对一个不存在的更高维度的索引位置进行赋值，此时 MATLAB 会自动扩展数组的维度。

方法三：使用 reshape 函数

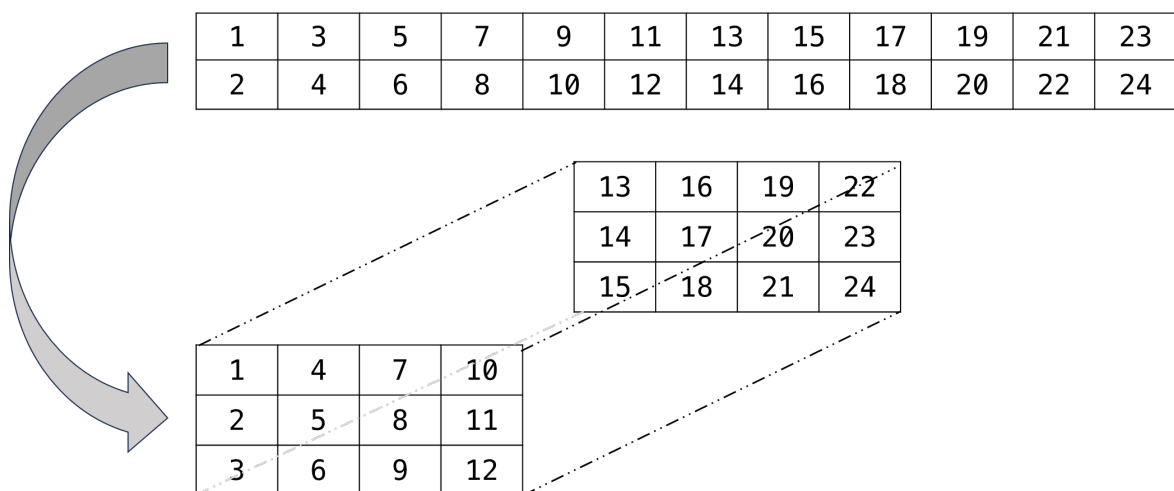
我们可以通过 reshape 函数将现有的数组重塑成指定的维度和大小。例如，如果我们有一个 2×12 的数组，我们可以将其重塑为一个 $4 \times 3 \times 2$ 的三维数组。

<pre>x = [6 3 9 1 7 7 6 8 7 4 8 10; 1 4 1 2 8 5 3 2 2 7 1 8]; x2 = reshape(x,3,4,2)</pre>	<pre>x2(:,:,1) = 6 4 1 8 1 9 2 7 3 1 7 5 x2(:,:,2) = 6 2 4 1 3 7 7 10 8 2 8 8</pre>
---	--

在上面的例子中，我们有一个 2×12 的二维数组，它被重塑成一个 $3 \times 4 \times 2$ 的三维数组。在这个过程中，元素的顺序保持不变，但是它们在新数组中的位置发生了改变，以适应新的维度结构。这里的顺序实际上就是我们第三章介绍的线性索引的顺序，只不过这里拓展到了三维的情况：

首先，MATLAB 按列取出原始二维数组的元素，即首先是第一列的 6 和 1，然后是第二列的 3 和 4，依此类推。这些元素然后被放置到新的三维数组中，首先填满第一页（即第一个切片），然后继续填充第二页。因此，在新的三维数组中，第一页（ $x2(:,:,1)$ ）包含了原数组前半部分的元素，而第二页（ $x2(:,:,2)$ ）包含了后半部分的元素。

下面这张图可以更好的帮助大家理解转换关系：



方法四：使用 cat 函数

第三章我们介绍过 cat 函数的基本用法，它能在指定维度上拼接数组，例如：

cat(1,[2,3],[4,5]) % dim = 1 沿着行方向拼接	<table><tr><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td></tr></table>	2	3	4	5				
2	3								
4	5								
cat(2,[2,3],[4,5]) % dim = 2 沿着列方向拼接	<table><tr><td>2</td><td>3</td><td>4</td><td>5</td></tr></table>	2	3	4	5				
2	3	4	5						
cat(3,[2,3],[4,5]) % dim = 3 沿着页方向拼接	<table><tr><td colspan="2">ans(:, :, 1) =</td></tr><tr><td>2</td><td>3</td></tr><tr><td colspan="2">ans(:, :, 2) =</td></tr><tr><td>4</td><td>5</td></tr></table>	ans(:, :, 1) =		2	3	ans(:, :, 2) =		4	5
ans(:, :, 1) =									
2	3								
ans(:, :, 2) =									
4	5								

下面我们再来看一个实际的例子。在这个例子中，我们有一个元胞数组 C，其大小为 6 行 1 列。这个数组中的每个元胞内都包含一个 30×3 的矩阵，这些矩阵代表同一班级学生在不同年级的成绩。具体来说，每个矩阵的 30 行代表该班级 30 名学生，而 3 列分别代表这些学生在语文、数学和英语这三门科目上的成绩。这 6 个矩阵分别代表这些学生从一年级到六年级的成绩变化。例如，C{1} 包含学生在一年级时的成绩，C{2} 包含他们在二年级时的成绩，以此类推，直到 C{6}，它包含学生在六年级时的成绩。

首先我们先来随机生成这个元胞数组 C，这里我们借助 mat2cell 函数：

<code>C = mat2cell(randi(100,30*6,3),ones(1,6) * 30,3)</code>
6×1 cell 数组 {30×3 double} {30×3 double} {30×3 double} {30×3 double} {30×3 double} {30×3 double}

接下来，我们使用 cat 函数将这些年级的成绩数据在第三个维度上拼接起来：

<code>CC = cat(3,C{:});</code> <code>size(CC)</code>	30 3 6
---	------------------

下面我们来看拼接后的三维数组 CC。在数组 CC 中，每页（第三个维度）代表一个年级，每行代表一名学生，而各列则代表语文、数学和英语科目。这种数据结构使得我们可以轻松地进行跨年级的数据分析。

例如，我们可以使用下面的代码来计算每个学生在这六个年级中各科目的平均成绩：

mean(CC,3) % dim = 3, 即沿着页方向计算平均值	% 30 行 3 列		
	58.1667	47.5000	49.8333
	48.8333	68.5000	39.0000
	⋮		
	⋮		
	⋮		
	58.8333	59.8333	49.8333
	58.3333	55.3333	61.3333

我们还可以通过第三章介绍的其他的统计函数来进一步探索这些数据，比如使用 std 函数来计算标准差，了解学生成绩的波动情况；或者使用 max 和 min 函数来找出最高分和最低分。大家可以课后自己测试这些统计函数在三维数组中的用法。

当然，这种高维数组的应用不仅限于成绩分析，还可以扩展到其他领域，例如金融数据分析、科学实验数据处理等。在这些领域，高维数组能够提供一个结构化的方式来组织和分析大量复杂的数据集。

（2）引用高维数组

这部分我们来介绍高维数组的引用。和二维数组（即矩阵）类似，我们既可以使用各维度的索引来引用数组的元素，又可以使用线性索引来引用数组的元素。

下面我们以三维数组 x 为例，来讲解相关的例子。

$x = [6 \ 4 \ 1 \ 8; 1 \ 9 \ 2 \ 7; 3 \ 1 \ 7 \ 5];$ $x(:, :, 2) = [6 \ 2 \ 4 \ 1; 3 \ 7 \ 7 \ 10; 8 \ 2 \ 8 \ 8]$	
$x(:, :, 1) =$ 6 4 1 8 1 9 2 7 3 1 7 5 $x(:, :, 2) =$ 6 2 4 1 3 7 7 10 8 2 8 8	
% 引用第二行、第四列、第二页的元素 $x(2, 4, 2)$	
10	
% 引用第二行、第四列的元素（故意不指定页索引） $x(2, 4)$ % MATLAB 会默认返回第一页（第一个切片）中对应位置的元素。	
7	
% 引用二三两行、一四两列、一二页的元素 $x(2:3, [1, 4], 1:2)$	
$ans(:, :, 1) =$ 1 7 3 5 $ans(:, :, 2) =$ 3 10 8 8	
% 引用每一行、倒数第二列、最后一页的元素 $x(:, end-1, end)$	
4 7 8	

% 将 x 中的所有元素按照线性索引的顺序重构成一个列向量（包含 24 个元素）
 $x(:)$ % 前 12 个元素来自第一页，后 12 个来自第二页（使用不同的颜色帮助大家更好的观察）

6
1
3
4
9
1
1
2
7
8
7
5
6
3
8
2
7
2
4
7
8
1
10
8

6	4	1	8
1	9	2	7
3	1	7	5

6	2	4	1
3	7	7	10
8	2	8	8

% 线性索引为 1、11 和 21 号位置的元素
 $x([1,11,21])$

6 7 8

6	4	1	8
1	9	2	7
3	1	7	5

6	2	4	1
3	7	7	10
8	2	8	8

（3）修改和删除高维数组

我们可以利用等号赋值的方法对高维数组中引用位置的元素进行修改，等号右侧数据的大小要和引用位置的数据大小兼容。

还是以上面的 x 数组为例，我们先将 x 赋值给 y ，然后对 y 进行修改：

$y = x;$
 $y(:,2,1) = [1 \ 3 \ 8]$ % 等号左侧是一个列向量，等号右侧是一个相同元素个数的行向量
 % 只需要大小兼容就能进行修改

左侧是修改前的示意图（下同）：

6	4	1	8
1	9	2	7
3	1	7	5

6	2	4	1
3	7	7	10
8	2	8	8

右侧是修改后的示意图（下同）：

6	1	1	8
1	3	2	7
3	8	7	5

6	2	4	1
3	7	7	10
8	2	8	8

```
y = x;
y(:,2,:) = cat(3,[1 3 8],[0 1 5]) % 左侧大小是 3×1×2, 右侧大小是 1×3×2
```

				6	2	4	1
				3	7	7	10
				8	2	8	8
6	4	1	8				
1	9	2	7				
3	1	7	5				

				6	0	4	1
				3	1	7	10
				8	5	8	8
6	1	1	8				
1	3	2	7				
3	8	7	5				

```
y = x;
y(:,2,:) = 8 % 右侧是一个常数
```

				6	2	4	1
				3	7	7	10
				8	2	8	8
6	4	1	8				
1	9	2	7				
3	1	7	5				

				6	8	4	1
				3	8	7	10
				8	8	8	8
6	8	1	8				
1	8	2	7				
3	8	7	5				

% 注意：修改数组元素时，MATLAB 不支持下面这种兼容模式：

```
y = x;
y(:,2,:) = [1 3 8]
```

报错：无法执行赋值，因为左侧的大小为 3×1×2，右侧的大小为 1×3。

% 将右侧改成列向量也不行

```
y(:,2,:) = [1 3 8]'
```

报错：无法执行赋值，因为左侧的大小为 3×1×2，右侧的大小为 3×1。

% 为什么我们的潜意识中会觉得 MATLAB 会输出下面的结果：

				6	2	4	1
				3	7	7	10
				8	2	8	8
6	4	1	8				
1	9	2	7				
3	1	7	5				

				6	1	4	1
				3	3	7	10
				8	8	8	8
6	1	1	8				
1	3	2	7				
3	8	7	5				

% 这是因为我们和算术运算中的五种兼容模式混淆了！

% 下面我们回顾一下矩阵的修改：

```
x = [1 2 3;
     4 5 6;
     7 8 9];
x(:, [1,2]) = [6;8;10]
```

报错：无法执行赋值，因为左侧的大小为 3×2，右侧的大小为 3×1。

% 但是在算术运算中是支持这种兼容模式的：

```
x(:, [1,2]) + [6;8;10]
```

```
7      8
12     13
17     18
```


下面再来介绍高维数组的删除操作，我们需要将等号右侧改为空向量 []。

```
y = x;
y(:,2,:) = [] % 删除每一页的第二列数据
```

6	4	1	8
1	9	2	7
3	1	7	5

6	2	4	1
3	7	7	10
8	2	8	8

6	1	8
1	2	7
3	7	5

6	4	1
3	7	10
8	8	8

```
y = x;
y(:,2,1) = [] % 不能够只删除第一页中的第二列数据
```

报错：空赋值只能具有一个非冒号索引。

```
y = x;
y(:, :, 1) = [] % 删除第一页，此时 MATLAB 会返回一个矩阵
```

6	4	1	8
1	9	2	7
3	1	7	5

6	2	4	1
3	7	7	10
8	2	8	8

6	2	4	1
3	7	7	10
8	2	8	8

```
y = x;
% 删除线性索引是 2 或者 3 的倍数对应位置的元素
ind = unique([2:2:numel(y), 3:3:numel(y)]);
% ind = [2 3 4 6 8 9 10 12 14 15 16 18 20 21 22 24]
y(ind) = [] % 使用线性索引删除时，会返回一个行向量
```

6	4	1	8
1	9	2	7
3	1	7	5

6	2	4	1
3	7	7	10
8	2	8	8

6	9	1	7	6	7	4	10
---	---	---	---	---	---	---	----

（4）高维数组中与维度相关的三个函数

处理高维数组的数据时，我们常常需要对高维数组的维度进行调整。下表给出了高维数组中常用的三个函数，这三个函数都与高维数组的维度相关。

函数名称	主要功能
ndims	ndims 函数用于返回数组的维度数量。ndims(A) 可以返回数组 A 的维度数量，其结果与 length(size(A)) 相同。
squeeze	在高维数组中，有时会存在一些长度为 1 的维度，这些维度在某些情况下可能是多余的。squeeze 函数可以删除这些长度为 1 的维度，从而简化数组的结构，使其更加紧凑。
permute	数组的维度顺序有时会影响数据处理的方式和效率。permute 函数允许我们重新排列数组的维度，这种置换操作在多维数据分析中尤其有用。

ndims 函数和 squeeze 的使用方法非常简单，我们直接来看例子：

<code>x = ones(2,3,4);</code> <code>ndims(x)</code> <code>% length(size(x))</code>	3
<code>y1 = ones(2,1,2)</code>	<code>y1(:, :, 1) =</code> 1 1 <code>y1(:, :, 2) =</code> 1 1
<code>squeeze(y1)</code>	1 1 1 1
<code>y2 = zeros(1,3,2)</code>	<code>y2(:, :, 1) =</code> 0 0 0 <code>y2(:, :, 2) =</code> 0 0 0
<code>squeeze(y2)</code>	0 0 0 0 0 0

下面我们重点介绍 permute 函数的使用方法：

`B = permute(A, dimorder)` 按照向量 `dimorder` 指定的顺序重新排列数组的维度。例如，`permute(A, [2 1])` 交换矩阵 `A` 的行和列维度。通常，输出数组的第 `i` 个维度是输入数组的维度 `dimorder(i)`。

上面这段话来自 MATLAB 的官方文档，我们来看下面的例子：

<code>x = [1, 2, 3;</code> 4, 5, 6]; <code>permute(x, [2,1])</code> <code>% 上面代码的效果等价于对 x 进行了转置</code>	1 4 2 5 3 6
<code>permute(x, [1,2])</code> <code>% 和 x 完全相同</code>	1 2 3 4 5 6
<code>% 下面我们来看三维数组的例子：</code> <code>y = cat(3, [1,2,3;4,5,6], [7,8,9;10,11,12])</code>	<code>y(:, :, 1) =</code> 1 2 3 4 5 6 <code>y(:, :, 2) =</code> 7 8 9 10 11 12
<code>permute(y, [2,1,3])</code> <code>% 对每一页中的矩阵分别转置</code>	<code>ans(:, :, 1) =</code> 1 4 2 5 3 6 <code>ans(:, :, 2) =</code> 7 10 8 11 9 12
<code>y'</code> <code>% 高维数组不能使用转置运算符</code> <code>% 要转置我们可以使用 permute 函数</code> <code>% 另外，MATLAB 在 2020b 版本推出了两个新函数：</code> <code>% pagetranspose 函数 pagematranspose 函数</code> <code>% 它们可以按页进行转置，前者等价于对每一页使用：</code> <code>X(:, :, i) .'</code> ，后者等价于 <code>X(:, :, i)'</code> 。它们的区别在	报错：TRANSPOSE 不支持 N 维数组。

下面我们结合生活中的实际例子，进一步讲解 `permute` 函数的用法。

假设我们有一个三维数组 `x`，它记录了 4 名学生在三门科目上的成绩，涵盖了两个不同的学期。具体来说，`x(:, :, 1)` 包含了第一个学期的成绩，而 `x(:, :, 2)` 包含了第二个学期的成绩。每一行代表一名学生，每一列代表一门科目。

```
x = [6 4 3; 1 9 2; 3 1 7; 5 6 7];
```

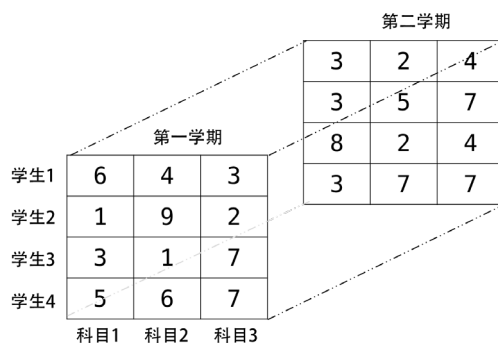
```
x(:, :, 2) = [3 2 4; 3 5 7; 8 2 4; 3 7 7]; % x 的大小是 4×3×2
```

```
x(:, :, 1) =
```

```
6 4 3
1 9 2
3 1 7
5 6 7
```

```
x(:, :, 2) =
```

```
3 2 4
3 5 7
8 2 4
3 7 7
```



现在，如果我们想按学期、科目和学生的顺序重新组织这些数据，我们可以使用 `permute` 函数来实现。`permute` 函数能够按照指定的顺序重新排列数组的维度。在我们的案例中，我们希望将原始数组中的学期维度（第三维）放到最前面，然后是科目（第二维）和学生（第一维）。这可以通过下面的命令实现：

```
y1 = permute(x, [3, 2, 1]) % y1 的大小是 2×3×4
```

```
y1(:, :, 1) =
```

```
6 4 3
3 2 4
```

```
y1(:, :, 2) =
```

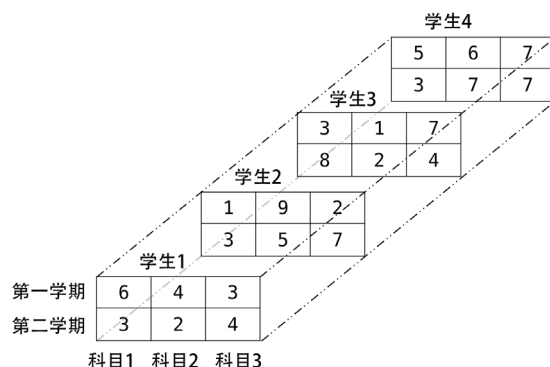
```
1 9 2
3 5 7
```

```
y1(:, :, 3) =
```

```
3 1 7
8 2 4
```

```
y1(:, :, 4) =
```

```
5 6 7
3 7 7
```



类似的，假设我们想按学期、学生和科目的顺序重新排列数据，可以这样做：

```
y2 = permute(x, [3, 1, 2]) % y1 的大小是 2×4×3
```

```
y2(:, :, 1) =
```

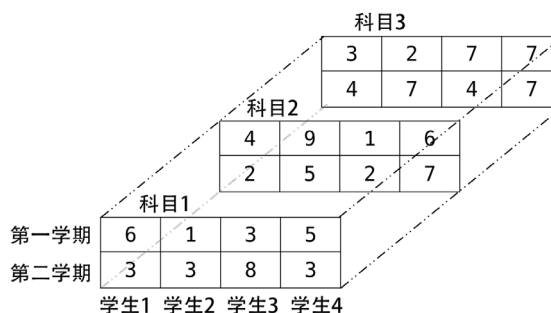
```
6 1 3 5
3 3 8 3
```

```
y2(:, :, 2) =
```

```
4 9 1 6
2 5 2 7
```

```
y2(:, :, 3) =
```

```
3 2 7 7
4 7 4 7
```



通过重新排列数组的维度，我们可以根据不同的需求，灵活地查看和分析数据。

（5）调用函数对高维数组进行计算

在第三章中，我们介绍过许多统计函数，例如 `sum` 求和函数、`mean` 求均值函数、`min` 求最小值函数等。这些函数也能用于三维数组的计算。下面以 `sum` 函数为例，给大家演示它在使用高维数组中的使用方法。

```
x = [6 4 1 8;1 9 2 7;3 1 7 5];
x(:,:,2) = [6 2 4 1;3 7 7 10;8 2 8 8]
```

`x(:,:,1) =`

```
6     4     1     8
1     9     2     7
3     1     7     5
```

`x(:,:,2) =`

```
6     2     4     1
3     7     7    10
8     2     8     8
```

				6	2	4	1
				3	7	7	10
				8	2	8	8
6	4	1	8				
1	9	2	7				
3	1	7	5				

`sum(x)` % 等价于 `sum(x,1)` 即 `dim = 1`，沿着第一个维度计算

`ans(:,:,1) =`

```
10    14    10    20
```

`ans(:,:,2) =`

```
17    11    19    19
```

					17	11	19	19
				6	2	4	1	
				3	7	7	10	
				8	2	8	8	
6	4	1	8					
1	9	2	7					
3	1	7	5					
10	14	10	20					

`sum(x,2)` % `dim = 2`，沿着第二个维度计算

`ans(:,:,1) =`

```
19
```

```
19
```

```
16
```

`ans(:,:,2) =`

```
13
```

```
27
```

```
26
```

				6	2	4	1	13
				3	7	7	10	27
				8	2	8	8	26
19	6	4	1	8				
19	1	9	2	7				
16	3	1	7	5				

`sum(x,3)` % `dim = 3`，沿着第三个维度计算

```
12     6     5     9
```

```
4     16     9    17
```

```
11     3    15    13
```

相同颜色的位置元素相加

				6	2	4	1
				3	7	7	10
				8	2	8	8
6	4	1	8				
1	9	2	7				
3	1	7	5				

上面代码中，我们指定了不同的维度对 x 进行求和。那么，我们应该如何求出 x 中所有元素的和呢？

<code>sum(x(:))</code> % 计算三维数组 x 中所有元素的和 % 也可以使用三次 <code>sum</code> 函数计算： <code>sum(sum(sum(x)))</code>	
120	<div>计算所有元素的和</div>

从 MATLAB2018b 版本开始，以 `sum` 函数为代表的众多统计函数支持对多个维度执行运算：我们可以将 `dim` 参数指定为一个由运算维度组成的向量，或指定 "all" 选项对所有数组的维度运算。因此，2018b 版本之后，我们也可以使用 `sum(x,"all")` 函数计算 x 中所有元素的和。

<code>sum(x,[1,2])</code> % 对第一维度和第二维度上的所有元素求和(即对 x 的每个页中的所有元素求和) % 只能使用 MATLAB2018b 以及之后的版本才能运行，下同	
<code>ans(:,:,1) =</code> 54 <code>ans(:,:,2) =</code> 66	
<code>sum(x,[1,3])</code> % 对第一维度和第三维度上的所有元素求和	
27 25 29 39	
<code>sum(x,[2,3])</code> % 对第二维度和第三维度上的所有元素求和	
32 46 42	

以上就是 `sum` 函数用于高维数组的例子，大家也可以自己测试其他的统计函数。

（6）高维数组的算术运算和关系运算

在 MATLAB 中，高维数组不仅支持基本的数据操作，如创建、引用和修改，它们同样能够进行算术和关系运算。

下面我们来看几个例子：

<pre>x = [6 4 1 8;1 9 2 7;3 1 7 5]; x(:,:,2) = [6 2 4 1;3 7 7 10;8 2 8 8]; x</pre>	<pre>x(:,:,1) = 6 4 1 8 1 9 2 7 3 1 7 5 x(:,:,2) = 6 2 4 1 3 7 7 10 8 2 8 8</pre>
<pre>x + [1 2 3 4]</pre>	<pre>ans(:,:,1) = 7 6 4 12 2 11 5 11 4 3 10 9 ans(:,:,2) = 7 4 7 5 4 9 10 14 9 4 11 12</pre>
<pre>x .* [1;2;3]</pre>	<pre>ans(:,:,1) = 6 4 1 8 2 18 4 14 9 3 21 15 ans(:,:,2) = 6 2 4 1 6 14 14 20 24 6 24 24</pre>
<pre>c = [1 1 1 1; 2 2 2 2; 3 3 3 3]; x .^ c</pre>	<pre>ans(:,:,1) = 6 4 1 8 1 81 4 49 27 1 343 125 ans(:,:,2) = 6 2 4 1 9 49 49 100 512 8 512 512</pre>
<pre>x >= 5</pre>	<pre>ans(:,:,1) = 1 0 0 1 0 1 0 1 0 0 1 1 ans(:,:,2) = 1 0 0 0 0 1 1 1 1 0 1 1</pre>
<pre>(x >= 3) & (x <= 6)</pre>	<pre>ans(:,:,1) = 1 1 0 0 0 0 0 0 1 0 0 1 ans(:,:,2) = 1 0 1 0 1 0 0 0 0 0 0 0</pre>

从上面的例子可以看出，高维数组的计算非常灵活，支持多种兼容模式，这为高维数据的计算和分析提供了方便。