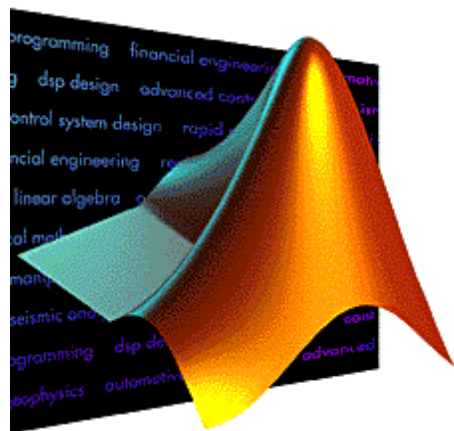

Matlab 程序设计与应用

第2章: Matlab数据及其运算

伍振海



Autumn, 2017 @swpu

第2章 Matlab数据及其运算



本章要点

- 2.1 MATLAB数据的特点
- 2.2 变量及其操作
- 2.3 MATLAB矩阵的表示
- 2.4 MATLAB数据的运算
- 2.5 字符串
- 2.6 结构数据和单元数据

变量类型

- **MATLAB** 是一种弱类型语言
 - 不需要初始化申明变量!
- **MATLAB** 支持多种变量类型，最常用的有
 - » 3.84
 - **64-bit double (default)**
 - » 'a'
 - **16-bit char**
- 最经常用的变量有：向量(**vector**)、矩阵(**matrix**)、双精度型(**double**)、字符型(**char**)
- 支持的类型还有：**complex, symbolic, 16-bit and 8 bit integers, etc.**

变量命名与赋值

- 要创建一个变量，只需要直接对其赋值：

- » `var1=3.14`

- » `myString='hello world'`

- 变量命名要求：

- **first character must be a LETTER**

- **after that, any combination of letters, numbers and _**

- **CASE SENSITIVE! (`var1` is different from `Var1`)**

变量命名与赋值

- 不要使用内建的变量名(表2.1), 如:
 - **i** and **j** can be used to indicate complex numbers
 - **pi** has the value 3.1415926...
 - **ans** stores the last unassigned value (like on a calculator)
 - **Inf** and **-Inf** are positive and negative infinity
 - **NaN** represents 'Not a Number'

例2-1 计算表达式 $\frac{5 + \cos 47^\circ}{1 + \sqrt{7} - 2i}$ 的值，并显示计算结果。

解： 在MATLAB命令窗口输入命令：

```
>> x=(5+cos(47*pi/180))/(1+sqrt(7)-2*i)
```

其中**pi**和**i**都是MATLAB预先定义的变量，分别代表代表圆周率 π 和虚数单位。

输出结果是：

x =

1.1980 + 0.6572i

变量管理

- 内存变量的显示: **who**和**whos**这两个命令用于显示在MATLAB工作空间中已经驻留的变量名清单。**who**命令只显示出驻留变量的名称, **whos**在给出变量名的同时, 还给出它们的大小、所占字节数及数据类型等信息。

» **who**

» **whos**

- 内存变量的删除: **clear**命令用于删除MATLAB工作空间中的变量。

» **clear var1**

» **clear all**

变量管理

图形界面管理方式----工作空间(**Work Sapce**):

MATLAB工作空间窗口专门用于内存变量的管理。在工作空间窗口中可以显示所有内存变量的属性。当选中某些变量后，再单击**Delete**按钮，就能删除这些变量。当选中某些变量后，再单击**Open**按钮，将进入变量编辑器。通过变量编辑器可以直接观察变量中的具体元素，也可修改变量中的具体元素。

变量管理

- 内存变量文件:

利用MAT文件可以把当前MATLAB工作空间中的一些有用变量长久地保留下来，扩展名是.mat。MAT文件的生成和装入由save和load命令来完成。

常用格式为:

save 文件名 [变量名表] [-append][-ascii]

load 文件名 [变量名表] [-ascii]

变量管理

其中，文件名可以带路径，但不需带扩展名`.mat`。当然，也可以指定任意扩展名。

变量名表中的变量个数不限，只要内存或文件中存在即可，变量名之间以空格分隔。当变量名表省略时，保存或装入全部变量。

`-ascii`选项使文件以ASCII格式处理，省略该选项时文件将以二进制格式处理。`save`命令中的`-append`选项控制将变量追加到MAT文件中。

数据输出格式

MATLAB用十进制数表示常数，有两种表示方法：

日常记数法：3.14159

科学记数法：1.78029e2, 1.78029E2

默认使用双精度数来表示和存储数据。

数据输出时用户可以用format命令设置或改变数据输出格式。format命令的格式为：

format 格式符

其中格式符决定数据的输出格式(表2.2)，如：

```
>> x=1/3
```

```
x =
```

```
0.3333
```

```
>> format long
```

```
>> x
```

```
x =
```

```
0.3333333333333333
```

MATLAB 中的数组

- 和所有其它语言一样，**数组**是**MATLAB**中非常重要的类型
- 两种数组
 - (1) 数字矩阵(双精度或复数)
 - (2) 结构数组与单元数据(更高级的数据结构)

MATLAB makes vectors easy!
That's its power!



行向量

- 行向量：中括号中的数据以空格或逗号分隔

```
» row = [1 2 5.4 -6.6]
```

```
» row = [1, 2, 5.4, -6.6];
```

- Command window:

```
>> row=[1 2 5.4 -6.6]
```

```
row =
```

```
1.0000    2.0000    5.4000   -6.6000
```

- Workspace:

Workspace			
Stack: Base			
Name	Size	Bytes	Class
row	1x4	32	double array

行向量

- 冒号表达式

冒号表达式可以产生一个行向量，一般格式是：

e1:e2:e3

其中e1为初始值，e2为步长，e3为终止值，缺省步长为1。

>> 1:0.5:3

1.0000 1.5000 2.0000 2.5000 3.0000

>> 1:1:3 %等价于1:3

1 2 3

提示：%为注释符，后面的内容不执行。

(类似于C语言中的//)

行向量

- **linspace**函数

在MATLAB中，还可以用**linspace**函数产生行向量。其调用格式为：

linspace(a,b,n)

其中**a**和**b**是生成向量的第一个和最后一个元素，**n**是元素总数。

显然，**linspace(a,b,n)**与**a:(b-a)/(n-1):b**等价。

>> 1:1:3 (等价于**1:3**)

1 2 3

>> linspace(1,3,3)

1 2 3

列向量

- 列向量：中括号中的数据以分号分开

» `column = [4;2;7;4]`



```
>> column=[4;2;7;4]
```

- **Command window:**

```
column =
```

- **Workspace:**

```
4  
2  
7  
4
```

Workspace			
 Stack: Base			
Name	Size	Bytes	Class
 column	4x1	32	double array

列向量

- 行向量转置得到列向量
- 转置运算符: **'**

>> col=row'

如:

>> x=1:0.5:3 %生成行向量

>> y=row' %转置成列向量

向量大小与长度

- 可以通过如下方式区分行向量与列向量：
 - 在**WorkSpace**中查看
 - 在**command window**中显示变量

➤ 使用**size**函数

```
>> size(row)
```

```
ans =
```

```
1      4
```

```
>> size(column)
```

```
ans =
```

```
4      1
```

向量大小与长度

➤ 获得一个向量的长度: **length**函数

```
>> length(row)
```

```
ans =
```

```
4
```

```
>> length(column)
```

```
ans =
```

```
4
```

➤ 获得一个向量元素个数: **numel**函数

```
>> numel(row)
```

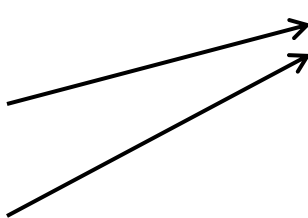
矩阵的建立

建立矩阵的方法与建立向量的方法类似

1、使用直接输入法

» **a= [1 2;3 4] ;**

» **a= [1,2;3,4] ;**

$$a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$


也可用回车符代替分号：

**a=[1 2
3 4]**

矩阵的建立

2、利用**M**文件建立矩阵

对于比较大且比较复杂的矩阵，可以为它专门建立一个**M**文件。

例2-2 利用M文件建立MYMAT矩阵。

(1) 启动有关编辑程序或MATLAB文本编辑器，并输入待建矩阵：

(2) 把输入的内容以纯文本方式存盘(设文件名为 `mymatrix.m`)。

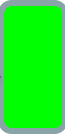
(3) 在MATLAB命令窗口中输入 `mymatrix`，即运行该M文件，就会自动建立一个名为MYMAT的矩阵，可供以后使用。

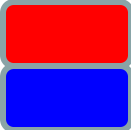
矩阵的建立

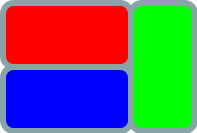
3、使用向量或矩阵组合（维数问题）

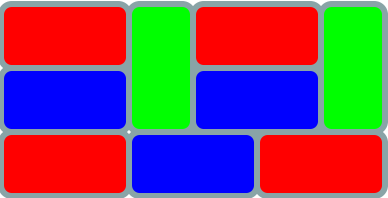
» `a = [1 2];` → 

» `b = [3 4];` → 

» `c = [5;6];` → 

» `d = [a;b];` → 

» `e = [d c];` → 

» `f = [[e e];[a b a]];` → 

» `str = ['Hello, I am ' 'John'];`

➤ **Strings are character vectors**

矩阵的大小

➤使用**size**函数

```
>> A=rand(3,5);
```

```
>> size(A)      %ans=3    5
```

➤**length**函数获得矩阵的行列数中的最大值

```
>> length(A)    %ans=5
```

➤获得矩阵的元素个数

```
M1: >> Nom=numel(A)      %Nom =15
```

```
M2: >> [r,c]=size(A)     %r=3,  c=5
```

```
>> Nom=r*c              %Nom =15
```

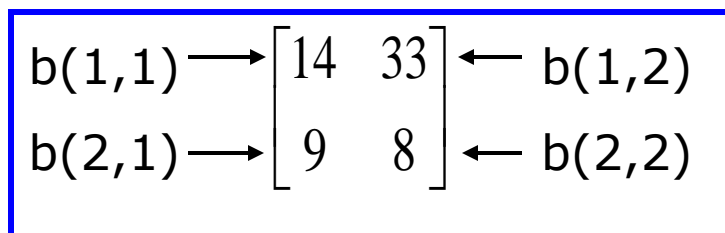
➤获得矩阵的维度

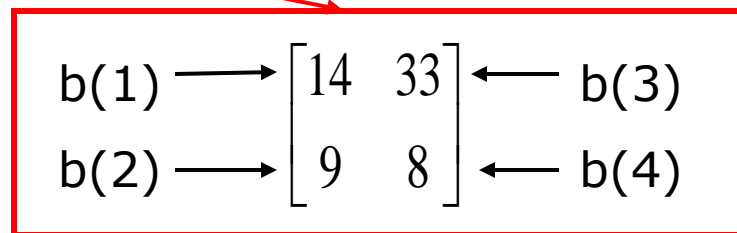
```
M1: >> Nd=ndims(A)       %Nd =2
```

```
M2: >> Nd=length(size(A)) %Nd =2
```

矩阵元素的引用

- 通过 **下标** 或 **序号** 来引用矩阵的元素


$$\begin{array}{lcl} b(1,1) \longrightarrow & \begin{bmatrix} 14 & 33 \end{bmatrix} & \longleftarrow b(1,2) \\ b(2,1) \longrightarrow & \begin{bmatrix} 9 & 8 \end{bmatrix} & \longleftarrow b(2,2) \end{array}$$


$$\begin{array}{lcl} b(1) \longrightarrow & \begin{bmatrix} 14 & 33 \end{bmatrix} & \longleftarrow b(3) \\ b(2) \longrightarrow & \begin{bmatrix} 9 & 8 \end{bmatrix} & \longleftarrow b(4) \end{array}$$

显然，序号(**Index**)与下标(**Subscript**)是一一对应的，以 $m \times n$ 矩阵A为例，矩阵元素A(i,j)的序号为 $(j-1)*m+i$ 。其相互转换关系也可利用**sub2ind**和**ind2sub**函数求得。

矩阵的拆分

1. 利用冒号表达式获得子矩阵

① $A(i,j)$: 取A矩阵第i行、第j列的元素。

$A([i,j],[m,n,p])$: 取第i,j行, m,n,p列的元素

$A(:,j)$: 取A矩阵的第j列全部元素;

$A(i,:)$: A矩阵第i行全部元素;

$A(:)$: 取所有元素, 且返回为一维列向量

矩阵的拆分

```
>> A=[1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15; 16 17 18  
19 20]
```

```
>> A(2,3) %取第2行第3列    >> A(:,2) %取第2列
```

```
ans =
```

```
8
```

```
>> A([2,3],[3,4])
```

```
%取第2,3行第3,4列
```

```
ans =
```

```
8    9
```

```
13   14
```

```
>> A(2,:) %取第2行
```

```
ans =
```

```
6    7    8    9   10
```

```
ans =
```

```
2
```

```
7
```

```
12
```

```
17
```

```
>> A(:) %取所有元素
```

```
1
```

```
2
```

```
....
```

矩阵的拆分

② **A(i:m,:)**表示取A矩阵第i~m行的全部元素；(m>i)

A(:,k:m)表示取A矩阵第k~m列的全部元素 (n>k)

A(i:m,k:n)表示取A矩阵第i~m行内，并在第k~n列中的所有元素。

```
>> A(2:3, 4:5) %第2、3行，4，5列元素
```

```
ans =
```

```
9    10
```

```
14   15
```

```
>> A(:,5)=[1;2;3;4] %为第5列重新赋值
```

此外，还可利用一般向量和**end**运算符来表示矩阵下标，从而获得子矩阵。**end**表示某一维的末尾元素下标。

矩阵的拆分

2. 利用空矩阵删除矩阵的元素: $X=[]$

注意: $X=[]$ 与`clear X`不同

$X=[]$: 将矩阵清空, 维数置为0, X 变量仍在

`clear X`: 从工作空间中删除 X 变量。

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

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

```
>> A(:,[3:5])=[]
```

```
A = 1 2 6 7 8 9
```

矩阵的拆分

3. 重新排列矩阵(reshape)

reshape(A,m,n): 将矩阵A重新排成 $m \times n$ 的二维矩阵。

注意：在MATLAB中，**矩阵元素按列存储**，即：

首先存储矩阵的第1列元素，

然后存储第2列元素，.....，

一直到矩阵的最后一列元素。

reshape函数只是改变原矩阵的行数和列数，即改变其逻辑结构，但并不改变原矩阵元素个数及其存储结构。

矩阵的拆分

如:

```
>> B=A(1:4,1:4) %取第1: 4行, 1: 4列
```

```
>> reshape(B,2,8) %重新排列为2x8矩阵
```

ans =

1 11 2 12 3 13 4 14

6 16 7 17 8 18 9 19

2.4.1 算术运算

1. 基本算术运算

- 基本运算符：加,减,乘,除,乘方(+,-,*,/,^)
 - » $7/45$
 - » $(1+i) * (2+i)$
 - » $1 / 0$
 - » $0 / 0$
 - » 4^2
 - » $(3+4*j)^2$
- 使用括号进行复杂的运算
 - » $((2+3)*3)^{0.1}$
- 括号外的运算符不可省略
 - » $3(1+0.7)$ %gives an error

2.4.1 算术运算

(1) 矩阵加减运算

若A和B矩阵的维数相同: A和B对应元素相加减。

$$\begin{array}{r} \begin{bmatrix} 12 & 3 & 32 & -11 \end{bmatrix} \\ + \begin{bmatrix} 2 & 11 & -30 & 32 \end{bmatrix} \\ \hline = \begin{bmatrix} 14 & 14 & 2 & 21 \end{bmatrix} \end{array} \quad \begin{bmatrix} 12 \\ 1 \\ -10 \\ 0 \end{bmatrix} - \begin{bmatrix} 3 \\ -1 \\ 13 \\ 33 \end{bmatrix} = \begin{bmatrix} 9 \\ 2 \\ -23 \\ -33 \end{bmatrix}$$

一个标量可以和其它任意维数矩阵相加减。

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

```
A =
```

```
1  2
3  4
```

```
>> B=A+1
```

```
B =
```

```
2  3
4  5
```

```
>> B-A
```

```
ans =
```

```
1  1
1  1
```


2.4.1 算术运算

若A与B的维数不同：出错。

- 如：行向量与列向量相加会出错

» `c = row + column`

- 如果元素个数相等，可以转置后再相加

» `c = row' + column`

» `r = row + column'`

- 对向量可以使用函数求和或求积

» `s=sum(row) ;`

» `p=prod(row) ;`

2.4.1 算术运算

(2). 矩阵乘法

矩阵 $A(m \times n)$ 和 $B(n \times p)$, 则 $C=A*B=C(m \times p)$ 。

➤ 要求: A 的列数= B 的行数

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots \\ a_{2,1} & a_{2,2} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} B = \begin{bmatrix} b_{1,1} & b_{1,2} & \dots \\ b_{2,1} & b_{2,2} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

则

$$AB = \begin{bmatrix} a_{1,1} [b_{1,1} & b_{1,2} & \dots] + a_{1,2} [b_{2,1} & b_{2,2} & \dots] + \dots \\ a_{2,1} [b_{1,1} & b_{1,2} & \dots] + a_{2,2} [b_{2,1} & b_{2,2} & \dots] + \dots \\ \vdots \end{bmatrix}$$

2.4.1 算术运算

(2). 矩阵乘法

矩阵 $A(m \times n)$ 和 $B(n \times p)$, 则 $C=A*B=C(m \times p)$ 。

➤ 要求: A 的列数= B 的行数

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} * \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = 11$$
$$1 \times 3 * 3 \times 1 = 1 \times 1$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Must be square to do powers

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 9 \\ 6 & 12 & 18 \\ 9 & 18 & 27 \end{bmatrix}$$

$$3 \times 3 * 3 \times 3 = 3 \times 3$$

2.4.1 算术运算

```
>> A=[1 2 3; 4 5 6];  
>> B=[1,2; 3 4; 5 6];
```

```
>> C=A*B
```

C =

22	28
49	64

```
>> D= B*A
```

D =

9	12	15
19	26	33
29	40	51

2.4.1 算术运算

(3). 矩阵除法

在MATLAB中，有两种矩阵除法运算：

- **\:** 左除， $A \setminus B \sim \text{inv}(A) * B$ ，即 $A \setminus B$ 等效于A的逆左乘B矩阵
- **/:** 右除， $B / A \sim B * \text{inv}(A)$ ，即 B / A 等效于A矩阵的逆右乘B矩阵

2.4.1 算术运算

条件:

- ✓ A 矩阵是非奇异方阵, 则 $A \setminus B$ 和 B/A 运算可以实现。
- ✓ 左除式要求参与运算的矩阵具有相同的行数
- ✓ 右除式要求参与运算的矩阵具有相同的列数
- ✓ 对于矩阵运算, 一般 $A \setminus B \neq B/A$ 。
- ✓ 对于含有标量的运算, 两种除法运算的结果相同, 如
 $3/4 = 4 \setminus 3 = 0.75$ 。

$a = [10.5, 25]$, 则 $a/5 = 5 \setminus a = [2.1000 \ 5.0000]$ 。

2.4.1 算术运算

>> a=[1 2 3; 4 2 6; 7 4 9]

a =

1 2 3

4 2 6

7 4 9

>> b=[4 3 2; 7 5 1; 12 7 92]

b =

4 3 2

7 5 1

12 7 92

>> c1=a\b %左除

c1 =

0.5000 -0.5000 44.5000

1.0000 0.0000 46.0000

0.5000 1.1667 -44.8333

>> c2=b/a %右除

c2 =

-0.1667 -3.3333 2.5000

-0.8333 -7.6667 5.5000

12.8333 63.6667 -36.5000

2.4.1 算术运算

(4). 矩阵的乘方

矩阵的乘方： A^x ，要求A为方阵，x为标量。

$$A^2 = A * A$$

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

A =

1 2

3 4

```
>> A^2
```

ans =

7 10

15 22

2.4.1 算术运算

2. 点运算

矩阵对应元素之间进行基本运算，包括：

$.^*$, $./$, $.^$

➤ 要求两矩阵的维数相同。

» `a=[1,2,3];b=[4;2;1];`

» `a.*b, a./b, a.^b` → all errors

» `a.*b', a./b', a.^(b')` → all valid

➤ 矩阵的乘除法 ≠ 矩阵的点乘除，即 $A*B \neq A.*B$,
 $A/B \neq A./B$

2.4.1 算术运算

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} .* \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = \text{ERROR}$$

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} .* \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 3 \end{bmatrix}$$

$$3 \times 1 .* 3 \times 1 = 3 \times 1$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} .^{\wedge} 2 = \begin{bmatrix} 1^2 & 2^2 \\ 3^2 & 4^2 \end{bmatrix}$$

Can be any dimension

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix} .* \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

$$3 \times 3 .* 3 \times 3 = 3 \times 3$$

2.4.1 算术运算

例：作出函数 $y = x \sin(x)$ 在 $[-2\pi, 2\pi]$ 区间内的图形。

提示：二维曲线作图函数 `plot` 用法：

`plot(x,y)` : 以 x 向量为横坐标, y 向量为纵坐标作图。

➤ 数据点越多, 曲线越光滑。

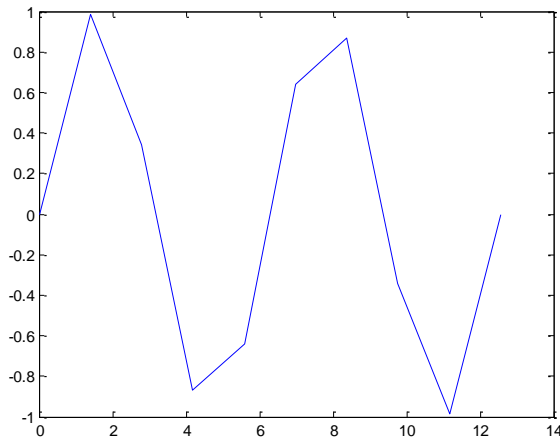
```
>> x=linspace(0,4*pi,10);
```

```
>> plot(x,sin(x));
```

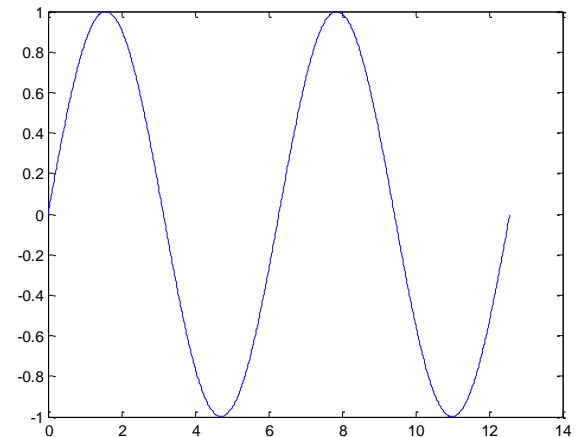
➤ x,y 向量大小必须相等, 否则出错。

```
>> plot([1 2], [1 2 3]) %error!
```

10 x values:



1000 x values:



2.4.1 算术运算

例：作出函数 $y = x \sin(x)$ 在 $[-2\pi, 2\pi]$ 区间内的图形。

提示：二维曲线作图函数 **plot** 用法：

plot(x,y) : 以 x 向量为横坐标, y 向量为纵坐标作图。
(x,y 向量大小必须相等)。

1、生成坐标点数据

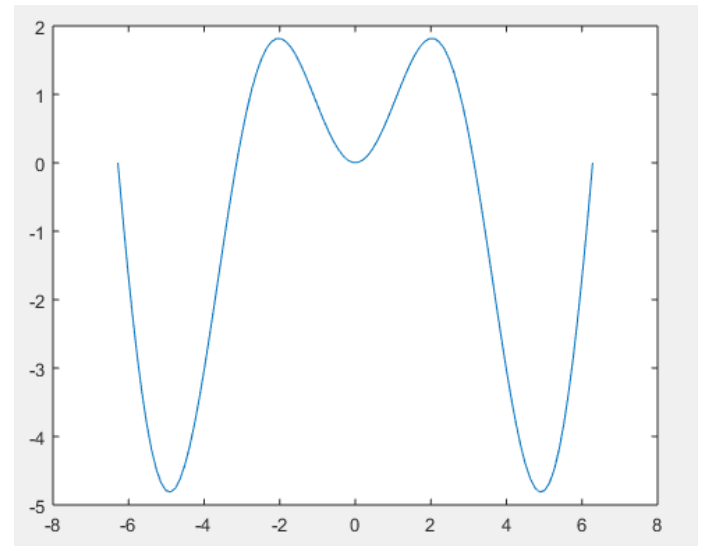
```
>> x=linspace(-2*pi,2*pi,100);
```

```
>> y=x*sin(x);    % error
```

```
>> y=x.*sin(x);
```

2、作图

```
>> plot(x,y)
```



2.4.1 算术运算

3. 常用数学函数(表2.3)

➤ 记住常用的函数:

sin, cos, tan, cot: 正弦、余弦、正切、余切

sqrt: 开平方, $\text{sqrt}(x)=x^{0.5}$

abs: 取绝对值

round: (四舍五入)

log, log10: 自然对数、常用对数

$$\log_a b = \frac{\log_n b}{\log_n a}$$

exp: 自然指数函数, $\text{exp}(x)=e^x$ (注意: matlab 中未定义e的值, **不能直接使用语法: e^x**)

pow2: 2的幂, $\text{pow2}(3)=2^3$

sign: 符号函数 $\text{sign}(x)=1, -1, 0$ ($x>0, x<0, x=0$)

2.4.1 算术运算

- 凡涉及角度的函数，均应以弧度为单位

sin, cos, tan, cot, asin, acos, atan.....

- 所有函数都可以作用在标量或矩阵上，作用在矩阵上时是**将函数作用在矩阵的每个元素上**

x=1:0.2:2;y=sin(x)

>> A=[1 2;3 4]

A =

1 2

3 4

>> sin(A)

ans =

0.8415 0.9093

0.1411 -0.7568

2.4.1 算术运算

取整函数的区别:

➤ **fix(x)** : 向零方向取整

$$\text{fix}(-2.6)=-2, \quad \text{fix}(2.6)=2$$

➤ **floor (x)** : 不大于x的最大整数（向负无穷大方向取整）

$$\text{floor}(-2.6)=-3, \quad \text{floor}(2.6)=2$$

➤ **ceil(x)**: 不小于x的最小整数（向正无穷大方向取整）

$$\text{ceil}(-2.6)=-2, \quad \text{ceil}(2.6)=3$$

➤ **round(x)**: 四舍五入

$$\text{round}(-2.6)=-3, \quad \text{round}(2.6)=3$$

$$\text{round}(-2.4)=-2, \quad \text{round}(2.4)=2$$

2.4.1 算术运算

求余(模)函数rem与mod的区别:

- $\text{rem}(x,y)=x-y*\text{fix}(x/y)$ (除后余数)
- $\text{mod}(x,y)=x-y*\text{floor}(x/y)$ (有符号数的除后余数)

当x、y符号相同(同正或同负)时, $\text{rem}=\text{mod}$

当x、y符号不同时, $\text{rem}(x,y)$ 与x同号, $\text{mod}(x,y)$ 与y同号

2.4.1 算术运算

求余(模)函数rem与mod的区别:

- $\text{mod}(10,3)=1$, $\text{mod}(-10,-3)=-1$

- $\text{rem}(10,3)=1$, $\text{rem}(-10,-3)=-1$

- $\text{mod}(10,-3)=-2$, $\text{mod}(-10,3)=2$

- $\text{rem}(10,-3)=1$, $\text{rem}(-10,3)=-1$

$x/y=10/-3=-3.3333$, $\text{floor}(-3.3333)=-4$, $10-(-3)*(-4)=-2$

$\text{fix}(-3.3333)=-3$, $10-(-3)*(-3)=1$

2.4.2 关系运算

MATLAB提供了6种关系运算符：

< 小于

<= 小于或等于

> 大于

>= 大于或等于

== 等于

~= 不等于

2.4.2 关系运算

关系运算符的运算法则为：

(1) 当两个比较量是标量时，直接比较两数的大小。若关系成立，关系表达式结果为1，否则为0。

1: True, 成立

0: False, 不成立

```
>> 1>2
```

```
ans =
```

```
0
```

```
>> 1<=2
```

```
ans =
```

```
1
```

2.4.2 关系运算

(2) 当参与比较的量是两个维数相同的矩阵时，比较是对两矩阵相同位置的元素按标量关系运算规则逐个进行，并给出元素比较结果。最终的关系运算的结果是一个维数与原矩阵相同的矩阵，它的元素由0或1组成。

```
>> A=[4 2; 5 9];
```

```
>> B=[3 8; 7 6];
```

```
>> C=A>B
```

```
C =
```

```
1    0
```

```
0    1
```

2.4.2 关系运算

(3) 当参与比较的一个是标量，而另一个是矩阵时，则把标量与矩阵的每一个元素按标量关系运算规则逐个比较，并给出元素比较结果。最终的关系运算的结果是一个维数与原矩阵相同的矩阵，它的元素由0或1组成。

```
>> A=[4 2; 5 9];
```

```
>> b=5;
```

```
>> A>b
```

```
ans =
```

```
0    0
```

```
0    1
```

2.4.2 关系运算

例：判断方阵A的元素是否能被3整除

A=[1 2 3 4; 5 6 7 8 ; 9 10 11 12 ; 13 14 15 16];

B=rem(A,3) %矩阵A的每个元素除以3的余数矩阵

P=B==0 %或 P=mod(A,3)==0,或P=rem(A,3)==0

0被扩展为与A同维数的零矩阵，以便与A矩阵元素进行一一对应比较，P是A矩阵与0矩阵进行等于(==)比较的结果矩阵。

A =

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

B =

1	2	0	1
2	0	1	2
0	1	2	0
1	2	0	1

P =

0	0	1	0
0	1	0	0
1	0	0	1
0	0	1	0

2.4.3 逻辑运算

MATLAB提供了3种逻辑运算符：

&与 **|** 或 **~**非

逻辑运算的运算法则（与c语言相同）：

- 非零元素为真，用1表示，零元素为假，用0表示
- **a&b**: a、b全为非零时，运算结果为1，否则为0
- **a|b**: a、b中只要有一个非零，运算结果为1；同时为零，则结果为0
- **~a**: 当a是零时，运算结果为1；当a非零时，运算结果为0。

2.4.3 逻辑运算

- 若参与逻辑运算的是两个同维矩阵，那么运算将对矩阵相同位置上的元素按标量规则逐个进行。最终运算结果是一个与原矩阵同维的矩阵，其元素由1或0组成。
- 若参与逻辑运算的一个是标量，一个是矩阵，那么运算将在标量与矩阵中的每个元素之间按标量规则逐个进行。最终运算结果是一个与矩阵同维的矩阵，其元素由1或0组成。
- 逻辑非是单目运算符，也服从矩阵运算规则。

2.4.3 逻辑运算

```
>> A=[ 2 -3; 0 0.2]
```

```
A =
```

```
    2.0000   -3.0000
```

```
    0    0.2000
```

```
>> A&B
```

```
ans =
```

```
    0    1
```

```
    0    1
```

```
>> A|B
```

```
ans =
```

```
    1    1
```

```
    0    1
```

```
>> B=[0, 5; 0 2]
```

```
B =
```

```
    0    5
```

```
    0    2
```

```
>> ~A
```

```
ans =
```

```
    0    0
```

```
    1    0
```

2.4.3 逻辑运算

运算优先级:

- 算术运算符>关系运算符>逻辑运算符
- 若有括号，则括号具有最高优先级

```
>> x=3&-1>0.5
```

```
x = 0
```

```
>> x=(3&-1)>0.5
```

```
x = 1
```

```
>>x=(3&-1+1)>0.5
```

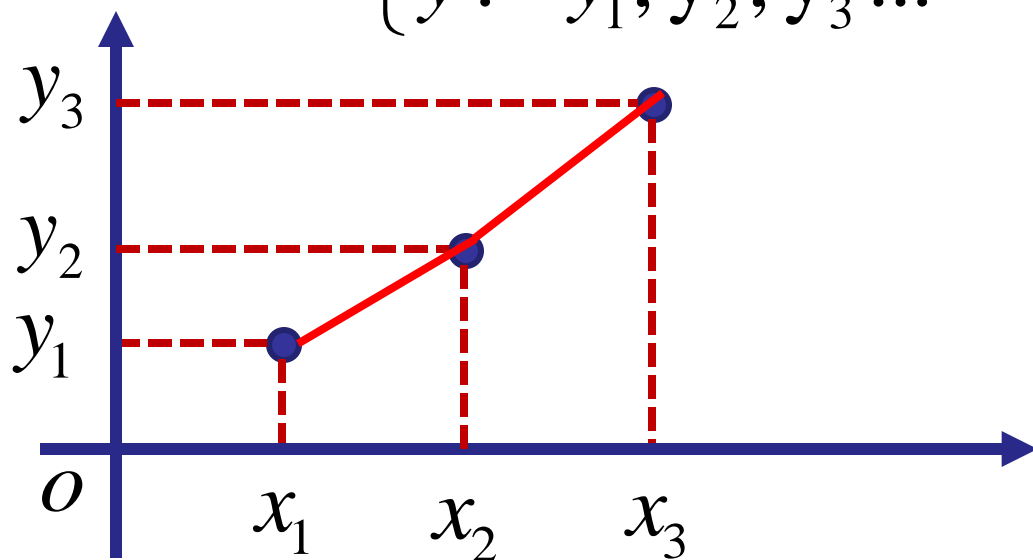
```
x=?
```

2.4.3 逻辑运算

例：用逻辑运算符表达分段函数并描绘图形：

$$y = \begin{cases} \sin(x) & (-\pi \leq x < 0) \\ \sqrt{x} & (0 \leq x \leq \pi) \end{cases}$$

分析：描点，连接。 $\begin{cases} x: & x_1, x_2, x_3 \dots \\ y: & y_1, y_2, y_3 \dots \end{cases}$



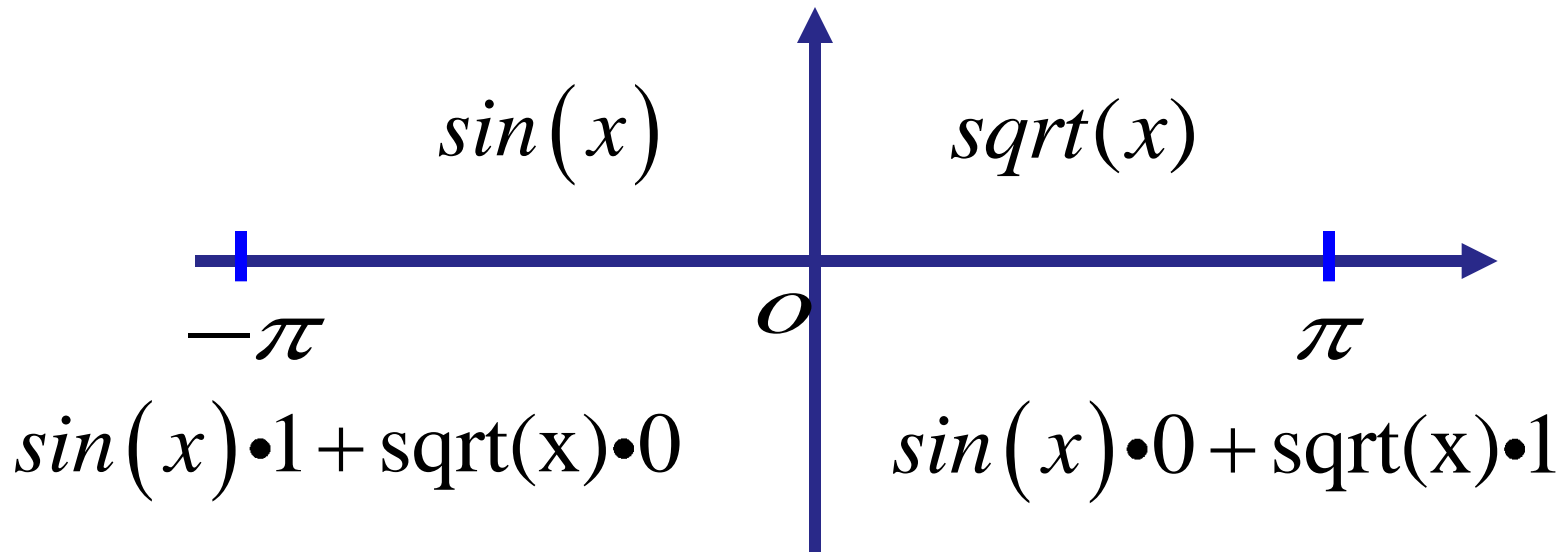
2.4.3 逻辑运算

(1): 生成 x 坐标的离散的坐标数据点:

```
>> x=linspace(-pi,pi,100)
```

(2): 计算对应的 y 坐标:

一般的顺序结构语言（如C语言），使用循环+判断
Matlab中有没有什么更简捷的办法？

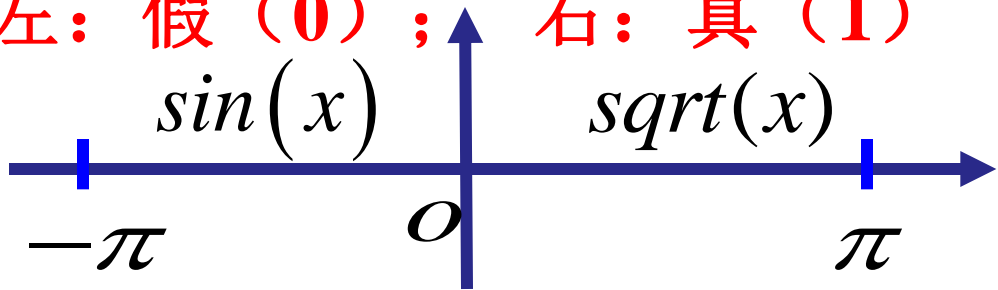


2.4.3 逻辑运算

把分段函数的分段条件范围写出：

$-\pi \leq x < 0$: 左：真（1）； 右：假（0）

$0 \leq x \leq \pi$: 左：假（0）； 右：真（1）



$x \geq -\pi \& x < 0$ 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0

$x \geq 0 \& x \leq \pi$ 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

$(x \geq -\pi \& x < 0) * \sin(x)$ $\sin(x) \dots \sin(x)$ 0 0 0 0 0 0 0 0

$(x \geq 0 \& x \leq \pi) * \sqrt{x}$ 0 0 0 0 0 0 0 0 $\sqrt{x} \dots \sqrt{x}$

}
+

$(x \geq -\pi \& x < 0) .* \sin(x) + (x \geq 0 \& x \leq \pi) .* \sqrt{x}$

$\sin(x) \dots \sin(x) \sqrt{x} \dots \sqrt{x}$

2.4.3 逻辑运算

(3): 计算做图:

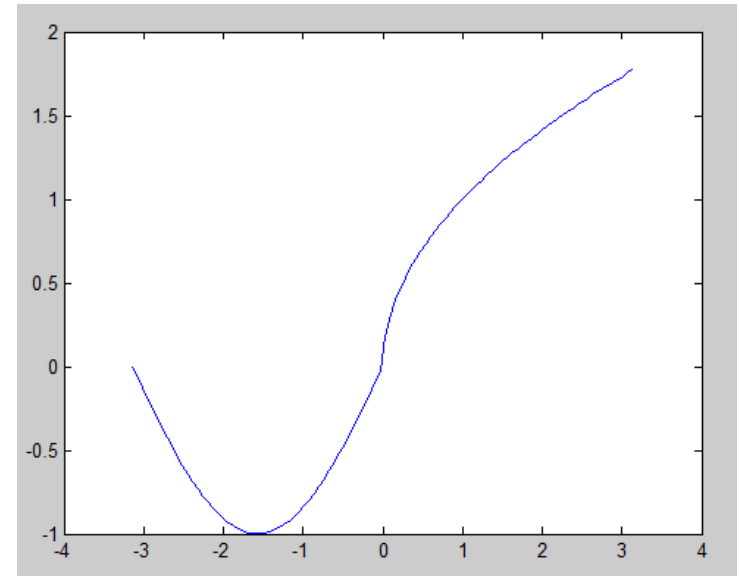
```
>> plot(x,y)
```

完整的程序:

```
>> x=linspace(-pi,pi,100)
```

```
>> y=(x>=-pi&x<0).*sin(x)...  
    +(x>=0&x<=pi).*sqrt(x);
```

```
>> plot(x,y);
```



总结：分段函数值的计算：

$$y = f1.*cond1 + f2.*cond2$$

2.4.3 逻辑运算

常用关系与逻辑运算函数(表2.4):

➤ **all(A)**: A的所有元素非零, 是=1, 否=0

➤ **any(A)**: A的任何一个元素非零, 是=1, 否=0

对向量:

```
>>A=[0,0,0]
```

```
>> all(A)    %0
```

```
>> all(A')   %0
```

对矩阵: 以列为单位

```
>> B=[0,1,3; 0,9,0; 0,2,5]
```

```
>> all(B)
```

```
ans =
```

```
0    1    0
```

重要的向量化运算函数：find

- **find** 是Matlab中一个非常重要的函数
 - 返回非零元素的位置，并以列向量形式返回
 - 可以避免循环，简化计算→**向量化运算！**
- 用法： **find(A)**: 找出A中非零元素的位置

Exa:

```
>> A=[0,3,5;2,0,9];  
>> x=find(A)
```

```
>> A=[0,3,5;2,0,9]  
A =  
    0    3    5  
    2    0    9
```

```
>> x=find(A)
```

```
x =
```

```
    2  
    3  
    5  
    6
```


重要的向量化运算函数: **find**

- **find**的常用方式: **index=find(cond)**

例:生成一个随机矩阵, 找出(0.4,0.8)内元素位置及其值

» **x=rand(3,4);**

» **inds = find(x>0.4 & x<0.8);**

%等效于A=x>0.4 & x<0.8; inds=find(A)

%inds 包含介于0.4和0.8之间的x的序号

» **val = x(inds);**

Example: Avoiding Loops

- `x= sin(linspace(0,10*pi,100))`,
求出为x中为正的元素个数?

使用 **loop and if/else**

```
count=0;
for n=1:length(x)
    if x(n)>0
        count=count+1;
    end
end
```

更高效的方法

```
count=length(find(x>0));
```

length(x)	Loop time	Find time
100	0.01	0
10,000	0.1	0
100,000	0.22	0
1,000,000	1.5	0.04

- Avoid loops!
- Built-in functions will make it faster to write and execute

2.5 字符串

➤ 字符串用成对的单引号表示

```
>> str='hello world'
```

➤ 字符串：本质为一个行向量

（每一个字符对应一个向量的一个元素）

➤ 多行时：相当于一个字符矩阵

（要求每一行的字符数必须相等）

```
>> str=['abcde';'12345']
```

```
str =  
abcde  
12345
```

2.5 字符串

➤ 字符串存储形式: **ASCII码**

➤ 常用字符串函数

B=abs(A): 取得A矩阵中的字符串的ASCII码数值

C=char(B): 将B矩阵ASCII值转换为字符串

```
>> A=['abcde';'12345']
```

```
>> B=abs(A)
```

```
>> C=char(B)
```

A =
abcde
12345

B =					
97	98	99	100	101	
49	50	51	52	53	

C =
abcde
12345

2.5 字符串

num2str: 将数字转换为字符串

str2num: 将字符串转换为数字

```
>> x=10;
```

```
>> cx=num2str(x) ;
```

```
>> y=str2num(cx) ;
```

工作区		
名称 ▲	值	类
abc cx	'10'	char
x	10	double
y	10	double

strcat(str1,str2,...,strn) =[str1,str2,...,strn]

连接字符串str1,str2,...,strn

```
>> str2=' zhang'
```

```
>> str2=' san'
```

```
>> sname=strcat(str1,str2)
```

```
>> sname=[str1,str2]
```

2.6 结构数据(structs)与单元数据(cell array)

- 目前学习了：二维矩阵→更维度矩阵
- 有些情况下，需要更复杂的数据结构来处理一些问题，包括：
 - 单元数据(**Cell array**): 类似于数组，但是每一个元素都可以是不同类型的数据。
 - 结构数据(**Structs**): 可以把不同的属性的数据类型纳入到一个变量名下

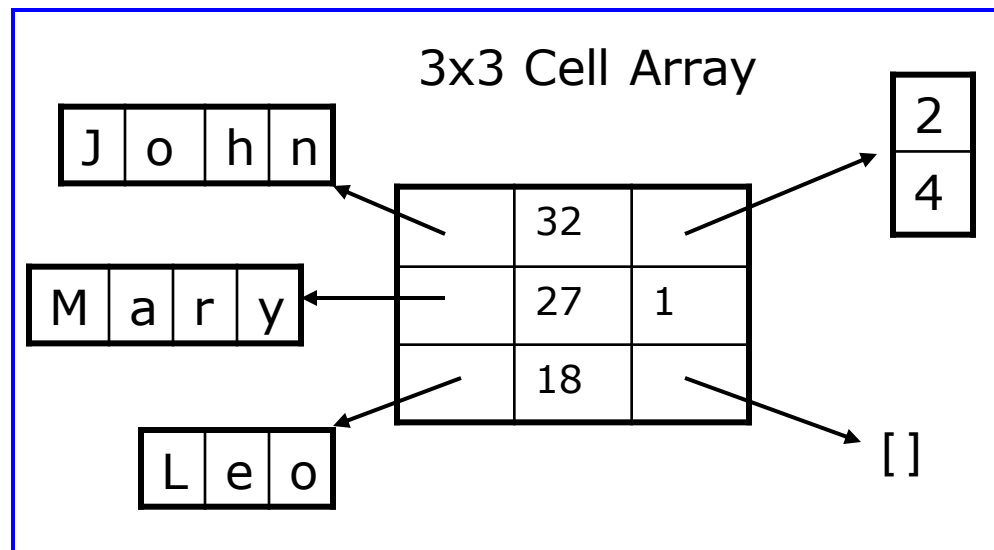
——**Matlab**中的面向对象编程

单元数据: Cells

- 单元数据类似于矩阵，但是每个元素都可以包含任意的数据类型（甚至包括矩阵）

3x3 Matrix

1.2	-3	5.5
-2.4	15	-10
7.8	-1.1	4



- 如：一个单元数据可以包括人的姓名、年龄、以及他的亲人的年龄等。
- 如果用矩阵来完成上面的数据存储，至少需要三个变量。

Cells: initialization

- 单元数据初始化:
 - » `a=cell(3,10);`
 - 建立一个**3x10**的单元数据结构
- 也可以直接赋值来建立一个单元数据, 数据使用大括号 `{}`
 - » `c={'hello world',[1 5 6 2],rand(3,2)};`
 - 建立一个**1x3**的单元数据
- 单元数据的每一个元素都可以是任意类型的数据
- 要引用单元数据一个元素, 必须使用大括号 `{}`
 - » `a{1,1}=[1 3 4 -10];`
 - » `a{2,1}='hello world 2';`
 - » `a{1,2}=c{3};`

结构数据：Structs

- 结构数据可以把具有相关性的变量存储到一个变量中
 - 类似于**C++**中的面向对象
- 初始化一个空的结构数据：
 - » `s=struct([]);`
 - `size(s) → 0x0`
 - 可初始化，也可直接赋值创建，但建议先初始化
- 添加结构成员：
 - » `s.name = 'Jack Bauer';`
 - » `s.scores = [95 98 67];`
 - » `s.year = 'G3';`
 - 成员可以是任意数据类型：矩阵、单元、甚至结构
 - 具有相关性的变量可以放在一起

结构矩阵: Struct Matrix

- 可以直接赋值初始化结构矩阵

```
» ppl=struct('name',{ 'John' , 'Mary' , 'Leo' } ,  
...  
'age' , {32 , 27 , 18} , 'childAge' , { [2;4] , 1 , [] }  
);
```

➤ **size(s2)=1x3**

➤ 每个元素必须具有相同数量的成员数

```
» person=ppl(2);
```

➤ **person** 包含: **name, age, childAge**

```
» person.name
```

➤ **returns 'Mary'**

```
» ppl(1).age
```

➤ **returns 32**

	ppl	ppl(1)	ppl(2)	ppl(3)
		↓	↓	↓
name:	→	'John'	'Mary'	'Leo'
age:	→	32	27	18
childAge:	→	[2;4]	1	[]

结构数据的引用

- 对 **1x1** 结构数据
 - » `stu=s.name;`
 - » `scor=s.scores;`
- 对 **nx1 struct** 数据（甚至矩阵），使用序号（或下标）
 - » `person=ppl(2);`
 - **person** 包含成员：**name,age, childAge**
 - » `personName=ppl(2).name;`
 - **personName** is 'Mary'
 - » `a=[ppl.age];`
 - 返回结构 **ppl** 的所有成员的年龄，并存储在 **1x3** 向量中。