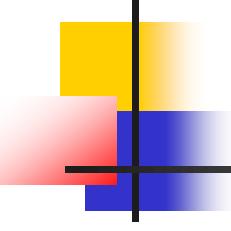


# Matlab编程及其应用

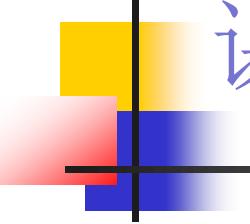
中国科技大学 信息学院  
陆伟

luwei@ustc.edu.cn



# 课程安排

- 地点: 3A111
- 时间: 周一 周三 周五 (七月份)
  - 1号 3号 5号
  - 8号 10号 12号
  - 15号
- 下午6、7、8节 14:00~16:30
- 考核方式: 平时作业 + 大作业



课程QQ群：

群聊号码：

**311730414**



2019MATLAB课程群

扫一扫二维码，加入群聊。

# Matlab软件下载

■ <http://ms.ustc.edu.cn/zbh.php>

中国科学技术大学软件正 版 软 件

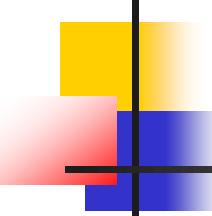
University of Science and Technology of China

■ 产品列表

| Windows | Office | ORIGIN | MATLAB    | 高斯      | 福昕PDF     | NOD32 |
|---------|--------|--------|-----------|---------|-----------|-------|
|         |        |        | Windows   |         | 安装密钥和激活文件 |       |
| 2018a   | 64位版   | 64位版   | 64位版      | 2018a   |           |       |
| 2017b   | 64位版   | 64位版   | 64位版      | 2017b   |           |       |
| 2017a   | 64位版   | 64位版   | 64位版      | 2017a   |           |       |
| 2016b   | 64位版   | 64位版   | 64位版      | 2016b   |           |       |
| 2016a   | 64位版   | 64位版   | 64位版      | 2016a   |           |       |
| 2015b   | 64位版   | 64位版   | 64位版 32位版 | 2015b   |           |       |
| 2015a   | 64位版   | 64位版   | 64位版 32位版 | 2015    |           |       |
| 安装说明    |        |        |           | 独立版激活密钥 |           |       |

■ 工具和文

- 自动更新服务
- 软件安装说明
- 升级 Windows 8.1
- 激活时出现0xC004F035错误解决
- Kms激活说明
- 应对windows和office 非标准安装脚本
- ISO to USB U盘制作工具
- Windows 7 U盘制作工具
- Windows 8 U盘制作工具
- Win7 USB/dvd tool download
- 启动时不能写入问题解决方法
- 一键制作Windows 8启动U盘教程



# 课程主要内容

- matlab基础：

- matlab简介、工作环境

- 基本数据类型与基本运算

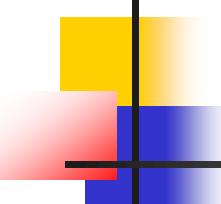
- 数组与矩阵

- 脚本与函数

- 字符串、单元数组、结构体、稀疏矩阵

- 数据可视化：

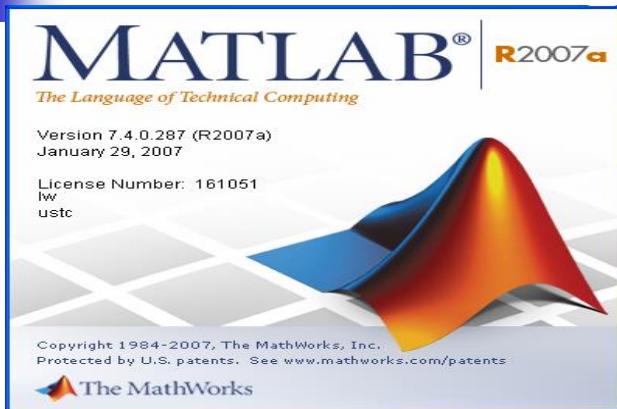
- 二维图形绘制、三维图形绘制



# 课程主要内容

- 数据分析与科学计算：
  - 多项式、插值拟合
  - 微分方程数值解法
  - 符号计算
- 图形句柄与GUI编写
- Simulink简介

# Matlab简介



**Matlab:**  
**Matrix Laboratory**  
矩阵 实验室

- 国际上应用最为广泛的科学与工程计算软件。
- 工程数学、控制、信号处理等许多专业课程的教学工具。
- 算法分析、系统仿真、数据可视化...
- 论文 ~ = matlab + word

# Matlab简介：历史



**Cleve Moler**



**Jack Little**

MathWorks公司创始人

**<http://blogs.mathworks.com/cleve/>**

“Experiments with MATLAB”

“Numerical Computing with MATLAB”

# Matlab简介：历史

**1984年， Matlab 1.0 （DOS版， 182K，  
20多个函数）**

**1992年， Matlab 4.0 （93年推出  
Windows版，加入 simulink）**

**1994年， Matlab 4.2（得到广泛重视和应用）**

**1999年， Matlab 5.3（真正实现32位运算）**

**2002年， Matlab 6.5（采用JIT加速器）**

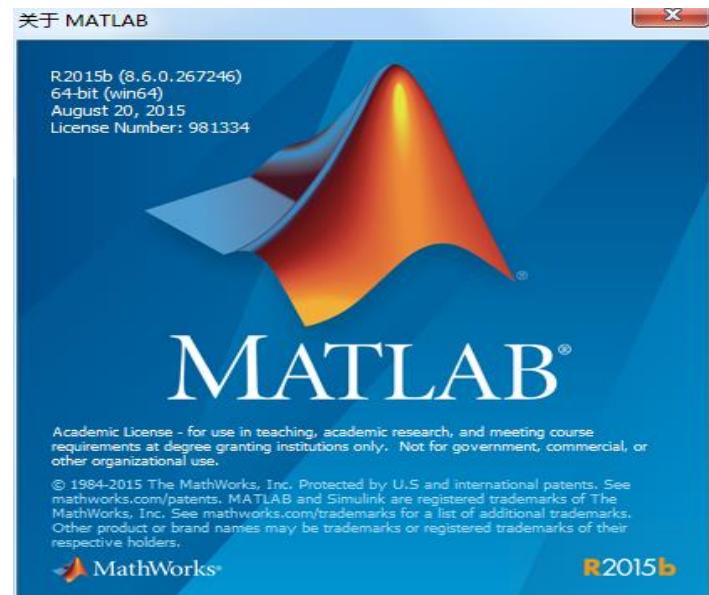
**2004年， Matlab 7.0 R14**

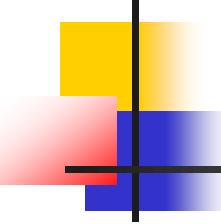
**2006年， Matlab 7.2 R2006a**

**2013年 Matlab 8.1 R2013a**

**2018年 R2018a**

.....



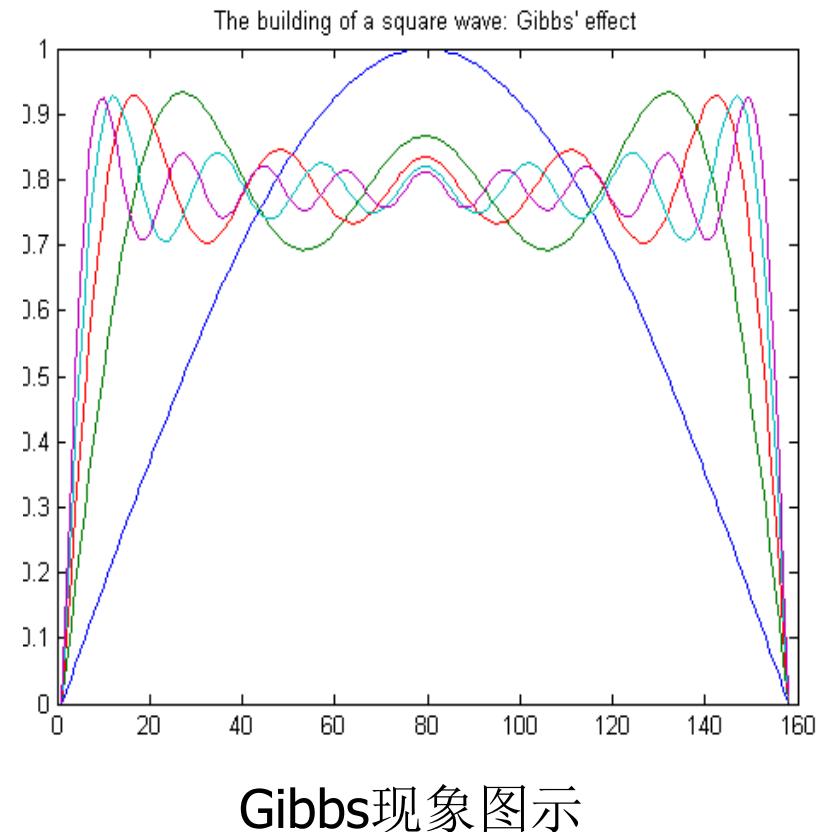


# Matlab简介：特点

- 强大的数值分析能力
- 丰富的可视化工具
- 简单易用的编程功能
- 图形化的系统建模
- 众多领域的工具箱
- ...
- 各种矩阵分解、SVD...
- 微分方程求解
- 统计
- 插值、拟合、优化
- .....
- 使得算法开发者跳出细节——砍树，不用从斧头做起。

# Matlab简介：特点

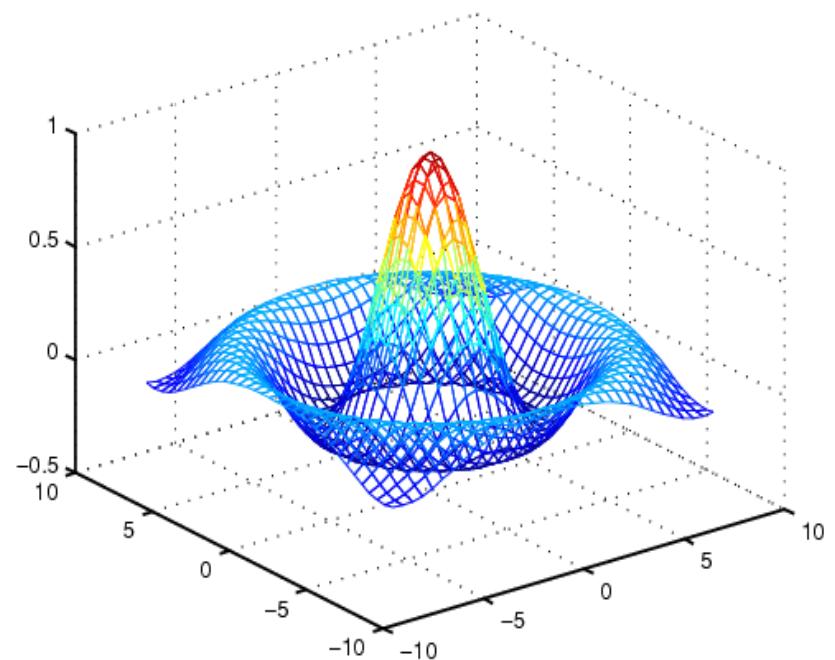
- 强大的数值分析能力
- 丰富的可视化工具
- 简单易用的编程功能
- 图形化的系统建模
- 众多领域的工具箱
- ...



# Matlab简介：特点

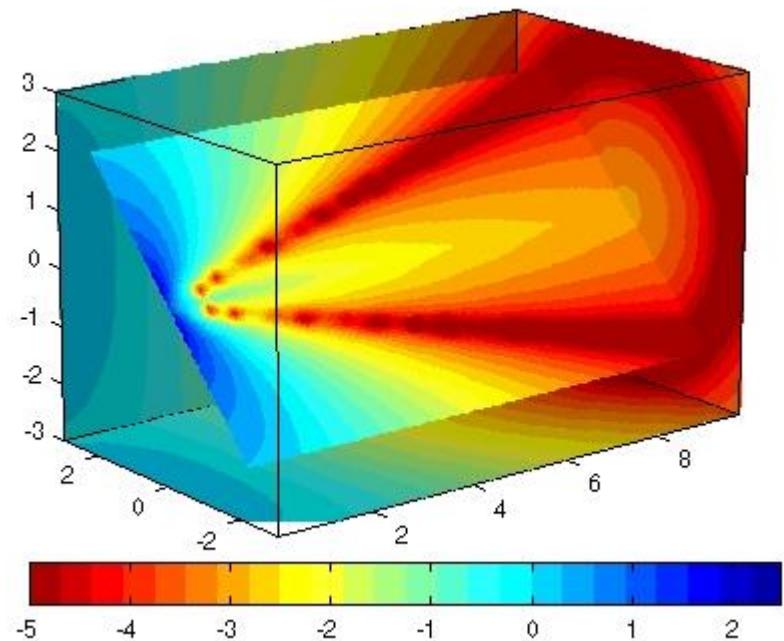
- 强大的数值分析能力
- 丰富的可视化工具
- 简单易用的编程功能
- 图形化的系统建模
- 众多领域的工具箱
- ...

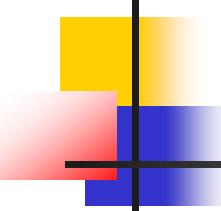
- 复杂数学公式图示



# Matlab简介：特点

- 强大的数值分析能力
  - 丰富的可视化工具
  - 简单易用的编程功能
  - 图形化的系统建模
  - 众多领域的工具箱
  - ...
- 发动机喷射气流的图形显示

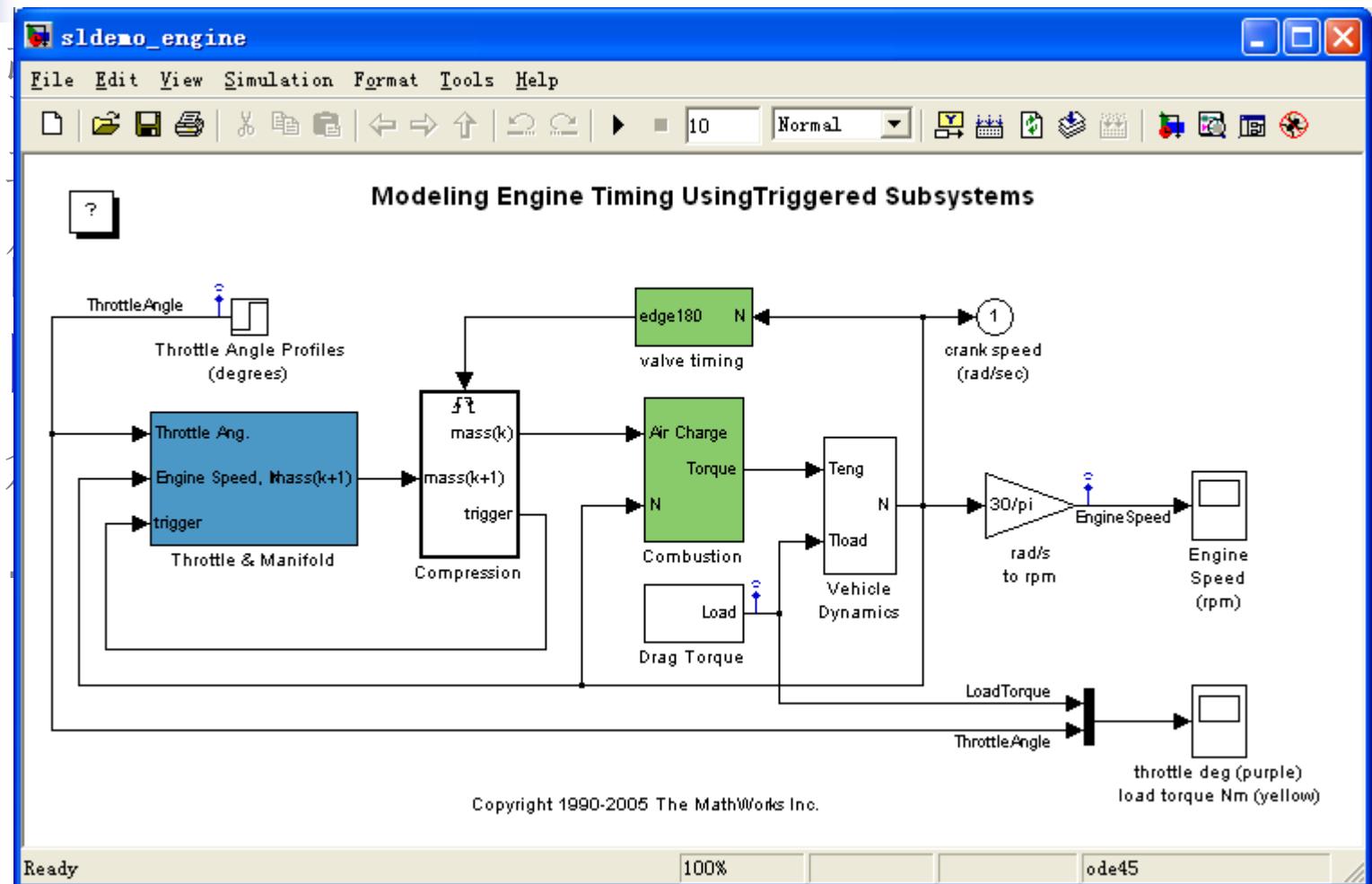


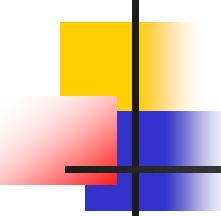


# Matlab简介：特点

- 强大的数值分析能力
- 丰富的可视化工具
- 简单易用的编程功能
- 图形化的系统建模
- 众多领域的工具箱
- ...
- 不用担心变量类型错误。
- 不用担心数组溢出。

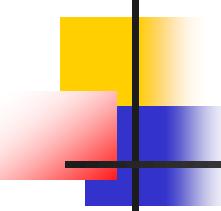
# Matlab简介：特点





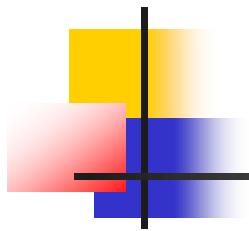
# Matlab简介：特点

- 强大的数值分析能力
  - 丰富的可视化工具
  - 简单易用的编程功能
  - 图形化的系统建模
  - 众多领域的工具箱
  - ...
- 
- 数学和优化
  - 统计和数据分析
  - 控制系统设计和分析
  - 信号处理
  - 通信
  - 图像处理
  - 测试和测量
  - 金融建模和分析
  - 数据库连接和报表
  - 分布式计算
  - ...



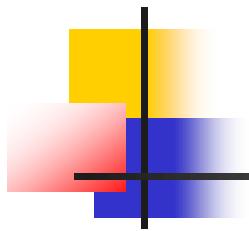
# Matlab简介：特点

- 强大的数值分析能力
- 丰富的可视化工具
- 简单易用的编程功能
- 图形化的系统建模
- 众多领域的工具箱
- ...
- 符号计算
- 与DSP、FPGA等硬件接口
- 与Excel、Word接口
- 与C、Java接口
- ...



# Matlab简介：网上资源

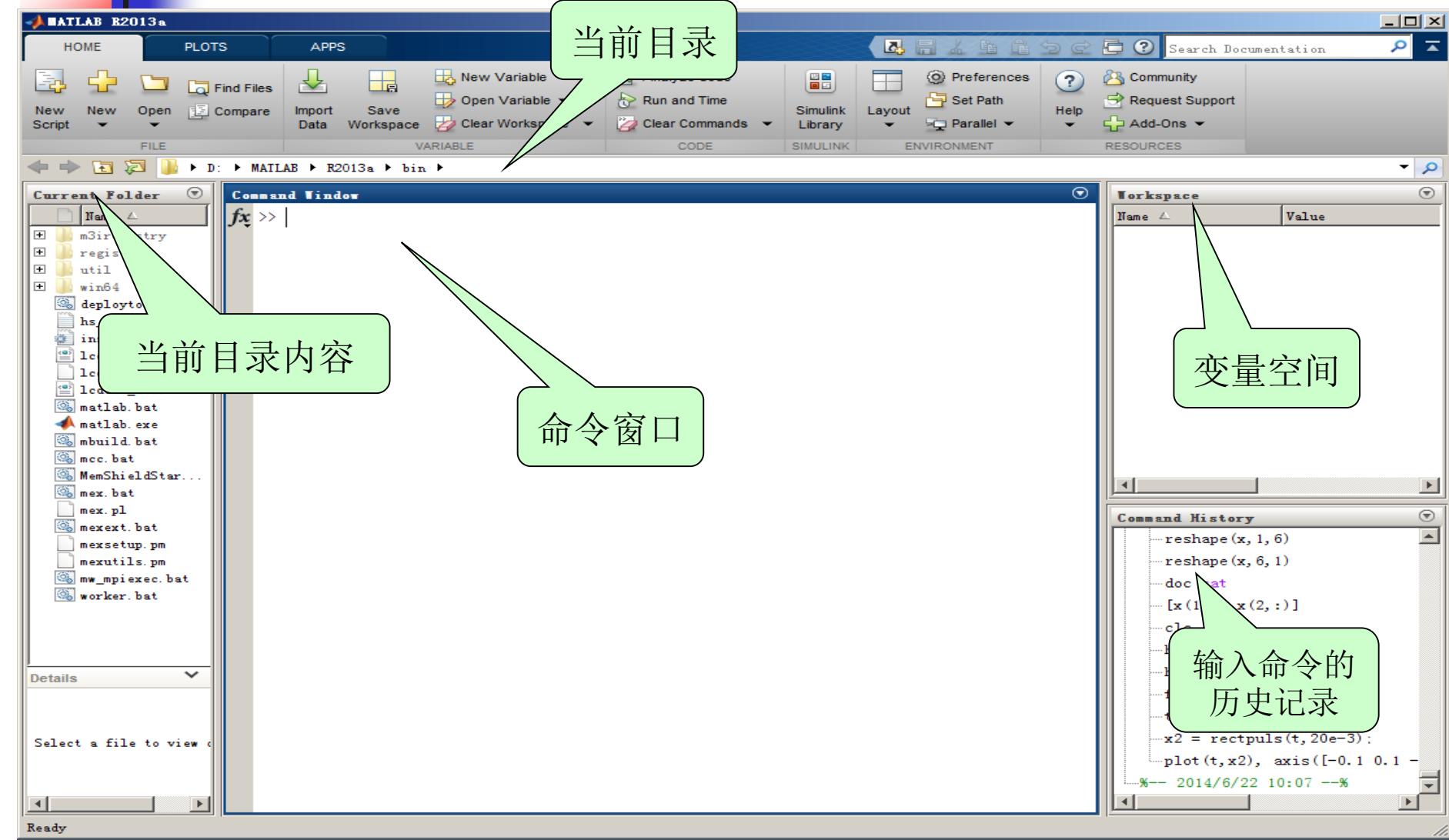
- [www.google.com](http://www.google.com)
- [www.mathworks.cn](http://www.mathworks.cn)



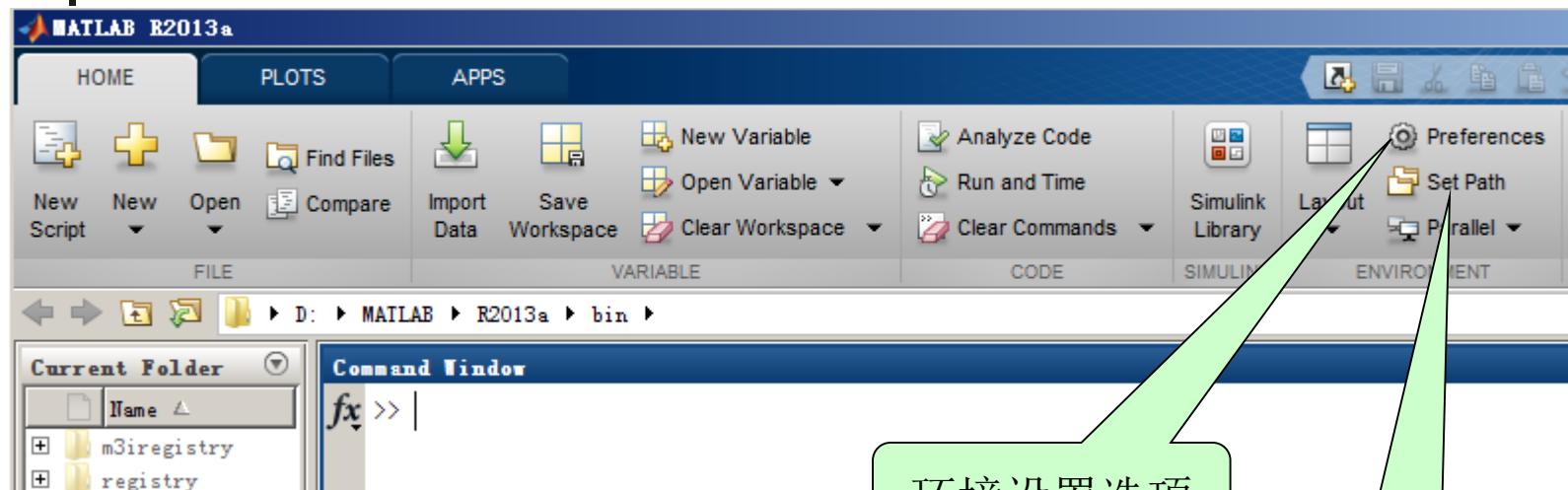
# Matlab简介

**Matlab再强大，也只是个工具！**

# Matlab简介:工作环境



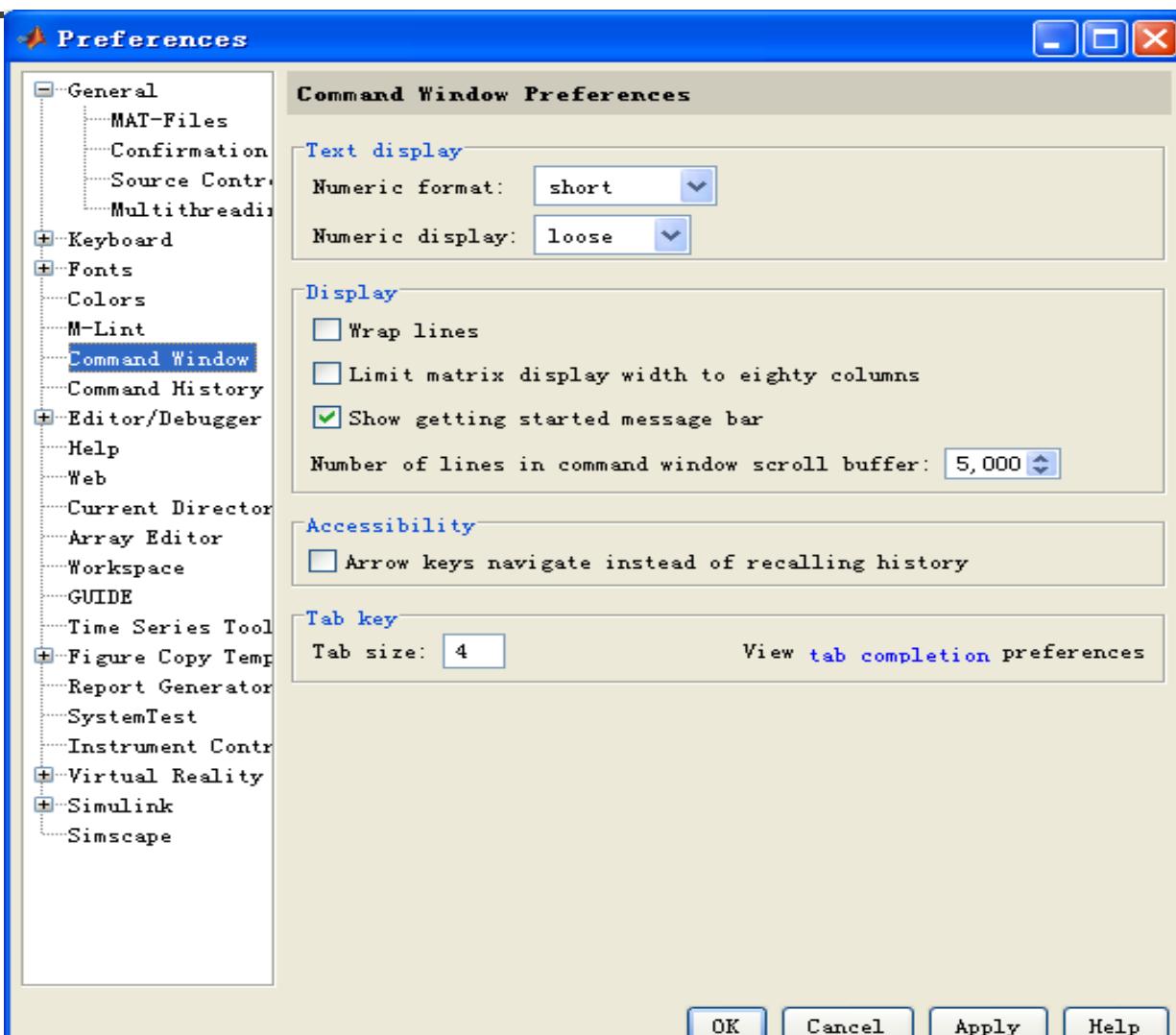
# Matlab : 工作环境



环境设置选项

搜索路径设置

# Matlab工作环境设置

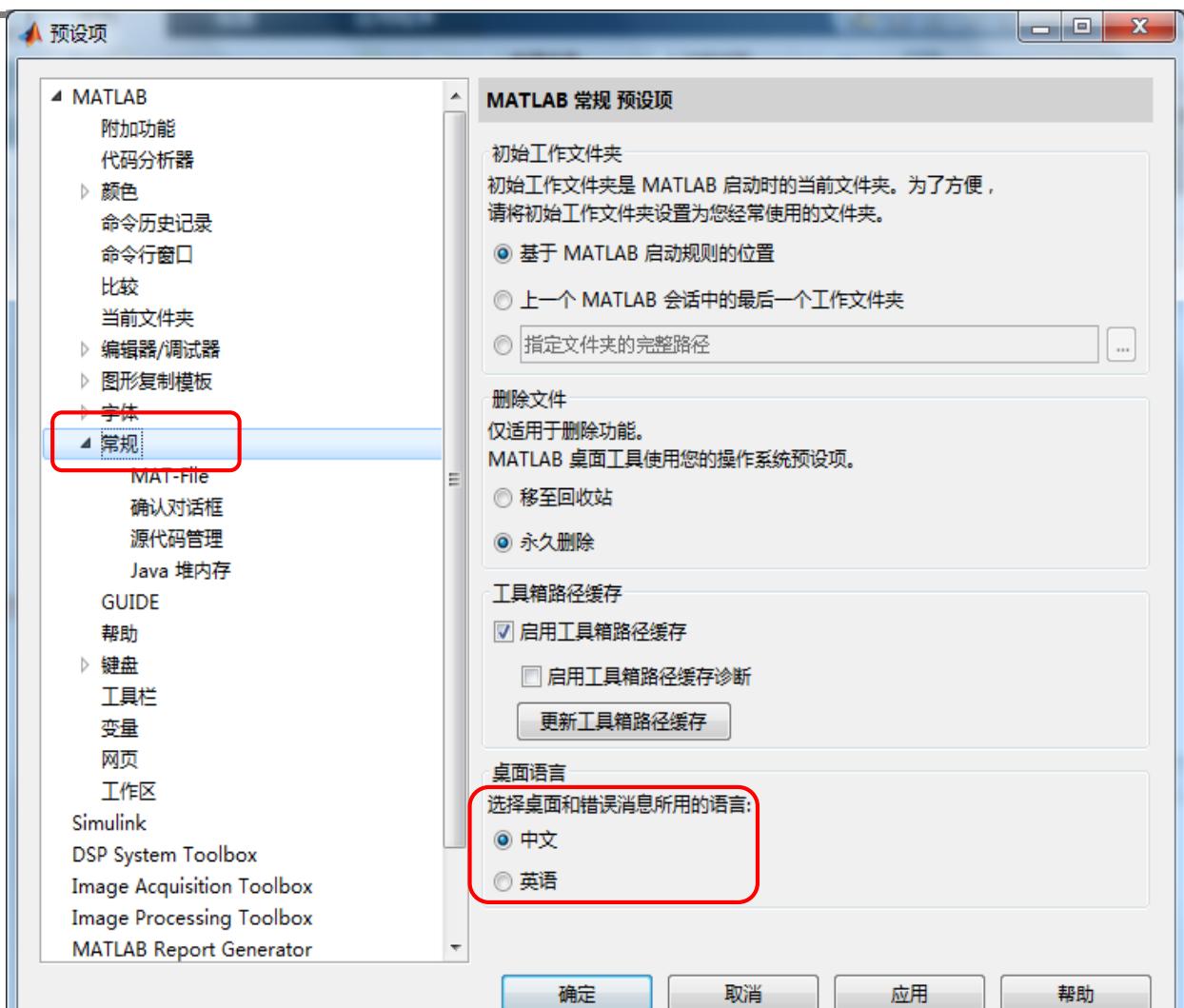


# 中文界面与英文界面转换

1、点击MATLAB菜单栏  
>预设

2、在预设界面中，  
> 常规  
> 选择桌面和错误信息  
的语言  
> 英文

3、确定后重启MATLAB



**Preferences**

**MATLAB**

- Add-Ons
- Code Analyzer
- Colors**
- Command History
- Command Window
- Comparison
- Current Folder
- Editor/Debugger**
- Figure Copy Template**
- Fonts**

**General**

- MAT-Files**
- Confirmation Dialogs
- Source Control
- Java Heap Memory
- GUIDE
- Help
- Keyboard**
- Toolbars
- Variables
- Web
- Workspace
- Simulink
- DSP System Toolbox
- Image Acquisition Toolbox
- Image Processing Toolbox
- MATLAB Report Generator
- Simulink 3D Animation**
- Simulink Control Design
- System Objects

**MATLAB General Preferences**

**Initial working folder**

The initial working folder is the current folder when MATLAB starts. For convenience, make the initial working folder a folder that you use frequently.

Location based on MATLAB startup rules

Last working folder from previous MATLAB session

Specify the full path to a folder  ...

**Deleting files**

Applies to the delete function only.  
MATLAB desktop tools use your operating system preference.

Move to the Recycle Bin

Delete permanently

**Toolbox path caching**

Enable toolbox path cache

Enable toolbox path cache diagnostics

**Update Toolbox Path Cache**

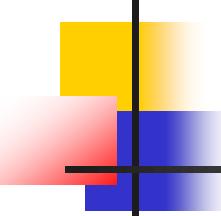
**Desktop language**

Select the language for the desktop and error messages:

Chinese

English

**OK** **Cancel** **Apply** **Help**



# Matlab简介：路径搜索

## MATLAB 的搜索顺序

当在命令窗口中或者一个 **M** 文件中输入一个元素名称时，**MATLAB** 按照下面的顺序搜索该元素的意义，以元素 **foo** 为例：

- 1) 查找**工作区**中是否存在名为 **foo** 的变量；
- 2) 在**当前路径**中查找是否存在名 **foo.m** 的文件；
- 3) 按照顺序查找**搜索路径**中是否存在该文件。如果存在多个名为 **foo.m** 的文件，则调用首先查到的文件。

# Matlab简介：路径搜索

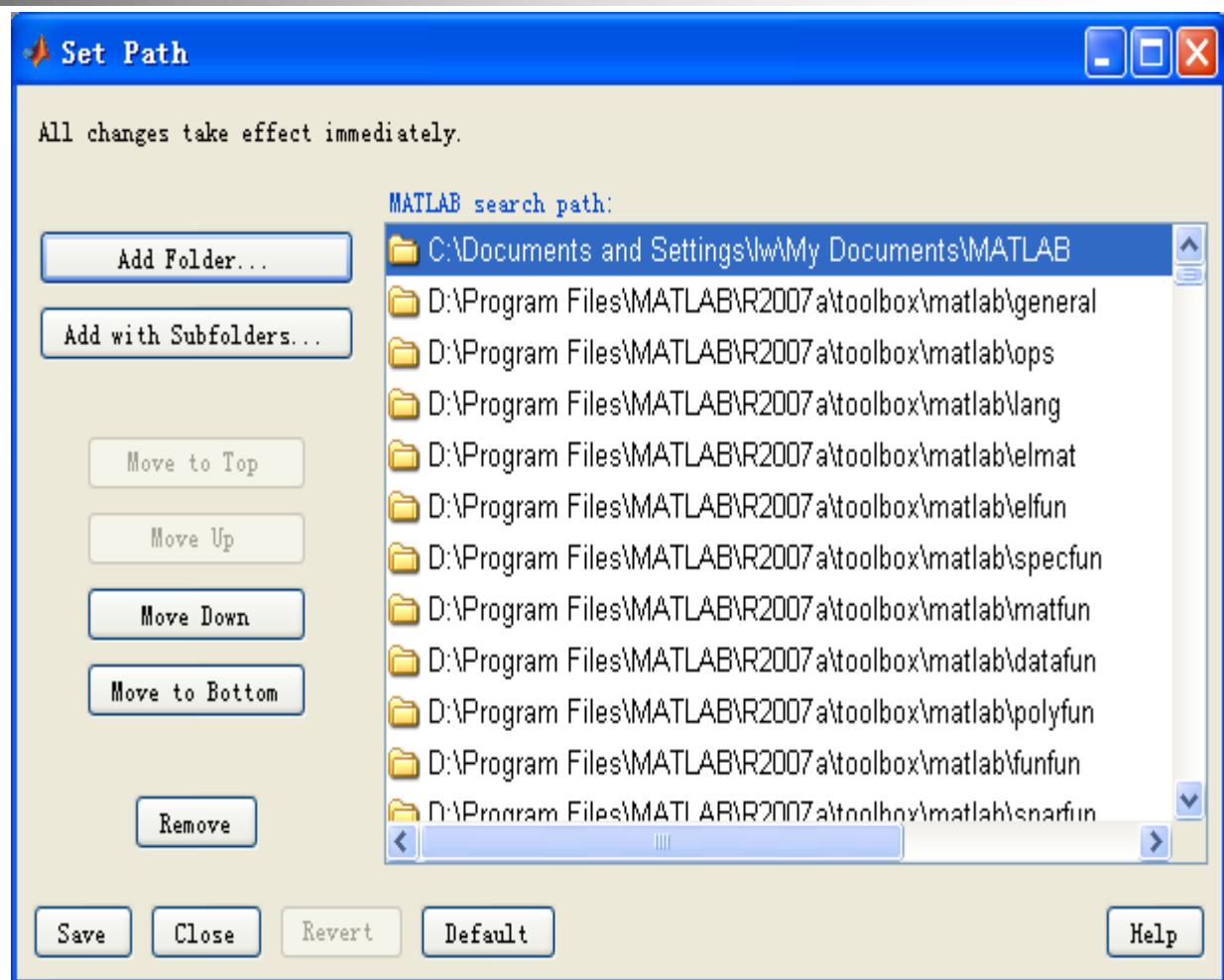
路径设置命令：

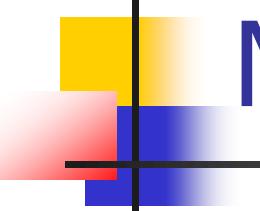
path  
path  
tool

addpath

rmpath

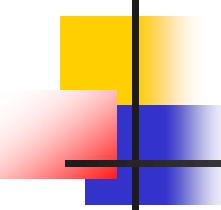
path





# Matlab基本操作

- 学习**MATLAB**的最简单方式：  
把**MATLAB**看作一个功能强大的  
“计算器”。
- 在命令窗口提示符处（>>）输入代码，  
**MATLAB**会即时返回操作结果。



# Matlab基本数学运算

- 两个数相加

```
>> 5.2+20  
ans = 25.2000
```

- 两个数相乘

```
>> 5*(-4)  
ans = -20
```

- 幂运算

```
>> 10^2  
ans = 100  
>> 5^(1/2)  
ans = 2.2361
```

**ans** 是Matlab内部变量，存储最近一次的运算结果。  
**(answer)**

# Matlab基本数学运算

- 两个数相除

```
>> 5 / 3 % 右除!
```

```
ans = 1.6667
```

```
>> 5 \ 3 % 左除
```

```
ans = 0.6000
```

- 运算优先级

```
>> (5^0.5-1)/2
```

```
ans = 0.6180
```

- 运算优先级与通常的一样。

# Matlab变量

## ■ 变量定义:

```
>> r = 4;
```

```
>> area = 2*pi*r^2
```

```
area = 100.5310
```

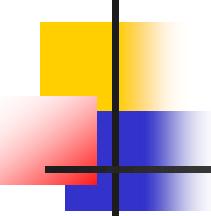
## 变量命名规则:

第一个字符必须是字母；

后面可以跟 字母、数字和下划线；如 r\_circle1, X314159

长度不超过63个字符；

变量名区分字母的大小写。 Cost、COST、cost都是不同的变量名



# Matlab变量

变量查询：

**who** 显示工作空间中的所有变量

**whos** 查看工作空间中变量的详细属性

```
>> who
```

```
Your variables are:  
area r
```

```
>> whos
```

| Name | Size | Bytes | Class  | Attributes |
|------|------|-------|--------|------------|
| area | 1x1  | 8     | double |            |
| r    | 1x1  | 8     | double |            |

工作空间中的变量清除指令：

**clear r** %清除变量r

**clear** %清除所有变量

# Matlab 变量

- Matlab 保留的**关键字**不能作为变量名：如for、while、if…

- 不要用Matlab中的一些**特殊变量名**：

**pi** 圆周率  $\pi$

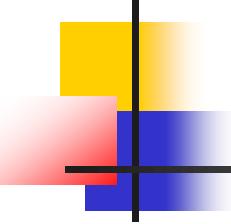
**inf/Inf** 无穷大      如 **5/0**

**nan/NaN Not-a-Number**, 一个不定值, 如 **0/0**

**eps** 浮点运算相对精度

**i/j** 虚部单位, 即  $\sqrt{-1}$

- 也不要用**内建函数名**做变量名, 如sin、log…

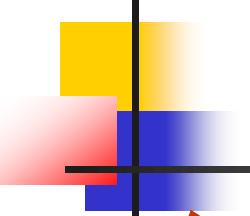


# Matlab变量

- 如果用函数名作变量名会怎样？

```
>> sin(pi/2) % sin是一个内建函数
ans =
    1
>> sin = 10 %如果把sin用作变量名
sin =
    10
>> sin(pi/2) % 出错 !
Subscript indices must either be real positive integers
or logicals.
```

若想恢复**sin**作为内建函数,怎么办?



# 基本数学函数

三角函数: `sin cos tan asin acos ...`

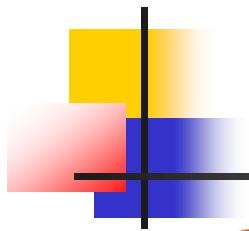
双曲函数: `sinh cosh tanh ...`

对数指数函数: `log`(自然对数) `log10 log2 exp`

取整函数: `fix floor ceil round`

随机数产生: `rand randn`

.....



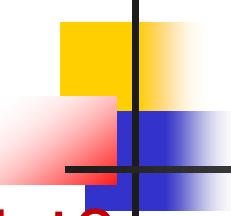
# 各种取整函数

**fix**      -> 0

**floor**    -> $-\infty$

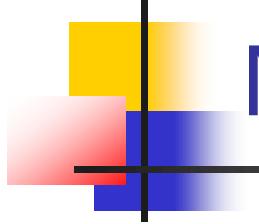
**ceil**      -> $+\infty$

**round**    ->最接近的整数



# Matlab数值类型

|              |               |                  |
|--------------|---------------|------------------|
| int8         | 8位有符号整型       | $-2^7 - 2^{7-1}$ |
| int16        | int32 int64   |                  |
| uint8        | 8位无符号整型       | $0 - 2^{8-1}$    |
| uint16       | uint32 uint64 |                  |
| single       | 单精度浮点数        |                  |
| double       | 双精度浮点数        | (默认)             |
| char         | 字符型、字符串       |                  |
| cell array   | 单元数组          |                  |
| struct array | 结构数组          |                  |



# Matlab数值类型

```
>> x1=int8(10.2)
```

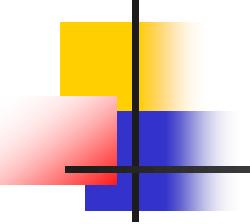
```
x1 =
```

```
10
```

```
>> x2 = int8(1000)
```

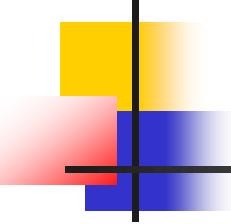
```
x2 =
```

```
127
```



# 复数定义与运算

```
a = 3+2i;  
b = 4-7*i;  
c = a+b;  
d = a*b;  
abs(a)          %模  
angle(a)        %角  
real(a)          %提取复数的实部  
imag(a)          %提取复数的虚部  
complex(3,2)    %构造一个复数， 实部为3， 虚部为2  
conj(a)          %a的共轭
```

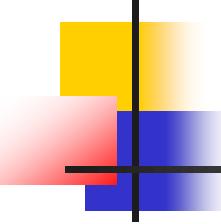


# 数组与矩阵

- matlab基本的数据组织形式是矩阵，
- matlab大部分函数可以直接对矩阵操作，
- 标量可以看做是 $1*1$ 的矩阵，
- 通常一维的称为数组或向量，
- 二维的称为矩阵或二维数组，
- 三维或三维以上的称为多维数组。

# 数组与矩阵

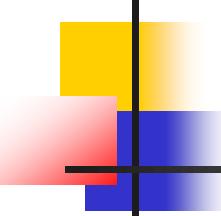
```
x1 = [1 2 3] %各个数之间是空格，生成行向量  
x2 = [1,2,3] %各个数之间是逗号，生成行向量  
x2 = [1; 2; 3] %各个数之间是分号，生成列向量  
>> A = [2 3 4  
           5   6  2] %生成2行3列矩阵  
  
A =  
  
     2         3         4  
     5         6         2  
  
A = [2 3 4 ; 5 6 2] %分号等同换行
```



# 数组与矩阵

- 可以用冒号生成间隔相等的数组：

```
>> a =[1:10] %默认间隔为1  
a = 1 2 3 4 5 6 7 8 9 10  
>> b =[ -2:3:8] %间隔为3  
b = -2 1 4 7  
>> c = [10:-2:0] %间隔为-2  
c = 10 8 6 4 2 0  
>> fs = 10;  
>> x = [0:1/fs:2*pi]
```



# 数组与矩阵

- 数组长度或矩阵行、列长度的查询：

**length**: 通常用来求数组长度

**size**: 可以得到矩阵行、列长度

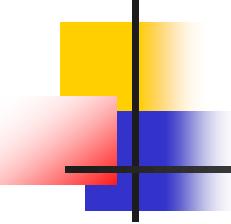
```
a =[1:10]; length(a); size(a);
```

```
b =[1 2 3;4 5 6];
```

```
L1 = size(b,1)
```

```
L2 = size(b,2)
```

```
[L3 L4] = size(b)
```



# 数组与矩阵

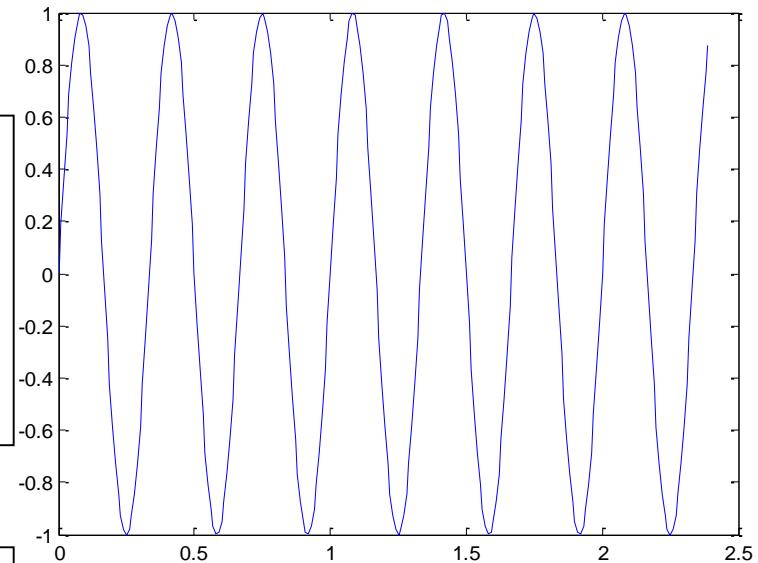
- matlab大部分函数可以直接对矩阵操作

```
>> x = [1 2 3]
>> x2 = [ 1 2 3; 4 5 6]
>> sin(x)
>> log(x)
>> exp(x)
>> 2*x.^2+3*x-2 % .^运算对每个元素操作
```

# 数组

例：产生一个3Hz正弦信号，采样频率为 100Hz

```
>> fs = 100;  
>> x = [0:1/fs:(240-1)/fs];  
>> y1=sin(2*pi*3*x);  
>> plot(x,y1)
```



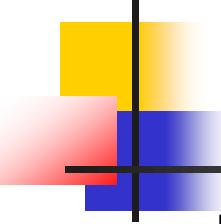
plot函数可以显示信号波形。

# 数组

实现如下函数：

$$x[n] = e^{jk\frac{2\pi}{N}n} + e^{-jk\frac{2\pi}{N}n} \quad n: 0 \sim N-1$$

```
>> N = 100;  
>> n = 0 : N-1;  
>> omega = 2*pi*n/N;  
>> x1 = exp(j*omega)+exp(-j*omega); plot(x1)  
>> x1 = exp(j*4*omega)+exp(-j*4*omega); plot(x1)  
>> x1 = (exp(j*4*omega)-exp(-j*4*omega))./(2*j);  
>> plot(x1)
```



# 随机信号产生

`rand` %0 – 1之间均匀分布的随机数

`rand(1)`

`x = rand(1:100000);`

`hist(x,30)` %直方图

%产生均值为0， 方差为1的高斯分布的随机数

`y = randn(1,100000);`

`hist(y,40)`

如何产生**-3**到**6**之间均匀分布的随机数?  
如何产生**1**到**100**之间均匀分布的整数?

# 随机信号产生

## ■ 设置（伪）随机数产生的方法与状态

```
rand(1,10)
```

```
rand(1,10)
```

```
rand('seed',11)
```

```
rand(1,10)
```

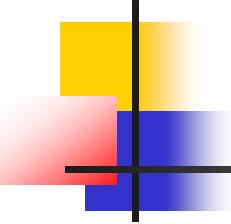
```
rand('seed',11)
```

```
rand(1,10)
```

```
rand('seed', 11)
```

随机数产生方法  
seed  
state  
twister

随机数发生器  
状态



# 随机信号产生

- 生成具有特定信噪比（SNR）的带噪信号

$$\text{SNR(dB)} = 10 \log_{10} \left( \frac{P_{\text{signal}}}{P_{\text{noise}}} \right) = 20 \log_{10} \left( \frac{A_{\text{signal}}}{A_{\text{noise}}} \right)$$

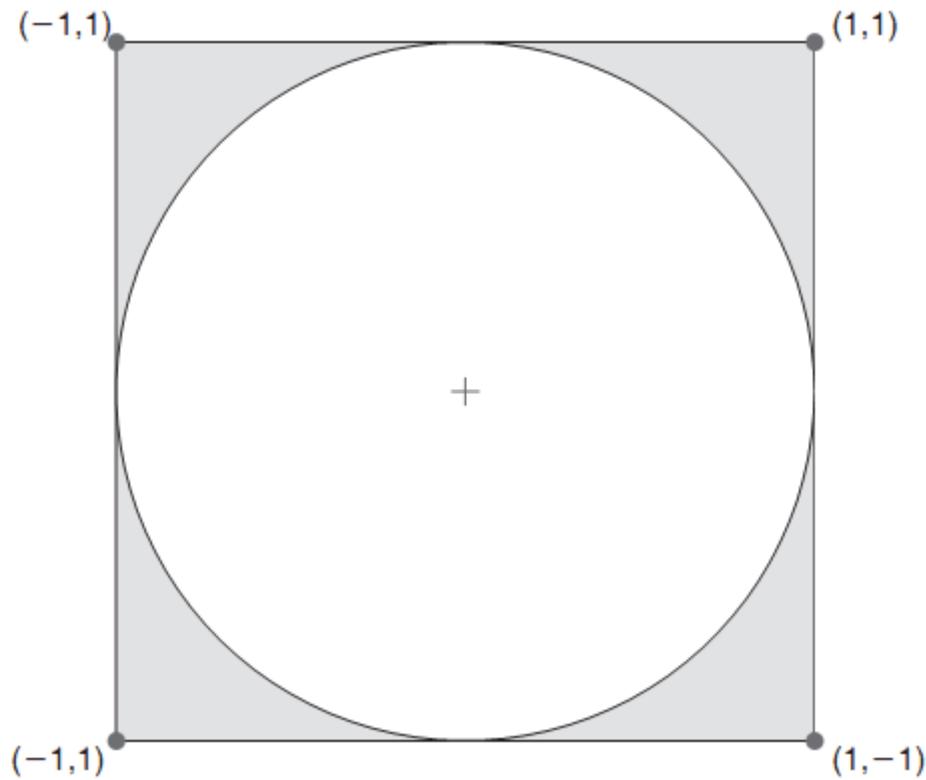
```
Pn = Ps/(10^(SNR/10))
```

```
Pn = sum((A*noise).^2)/length(noise)
```

```
A = (Pn*length(noise)/(sum(noise.^2)))^0.5;
```

# 随机信号产生

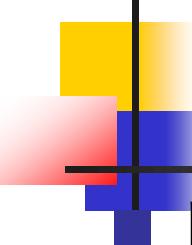
## ■ 蒙特卡罗方法求 $\pi$ 的近似值



$$x^2 + y^2 \leq 1$$

$$\frac{\text{hits}}{n} \approx \frac{\pi r^2}{(2r)^2}.$$

$$\pi \approx 4 \frac{\text{hits}}{n}.$$



# 关系运算

- 比较两个数之间的大小关系，若符合则结果为1，否则为0。
- 比较两个维数相同的矩阵，即比较两个矩阵对应的元素，比较结果仍然是一个矩阵

- < 小于
- <= 小于等于
- > 大于
- >= 大于等于
- == 等于
- ~= 不等于

```
A =[1:5];
B =[3 3 3 3 3];
c1 = A>B
c1 = 0 0 0 1 1
c2 = A<=2
c2 = 1 1 0 0 0
c3 = A == 4
c3 = 0 0 0 1 0
```

# 逻辑运算

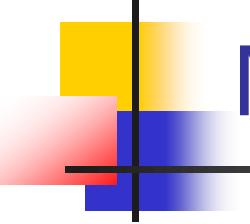
- **&** 与 两标量或两元素均非0则返回1，否则返回0
- **&&** 标量与
- **|** 或 两标量至少有一个是非0则返回1，否则返回0
- **||** 标量或
- **~** 非
- **xor(a,b)** 异或 两标量非0或均为0则返回0，否则返回1

# 逻辑运算

```
>> A = [0, 2, -1, 19, 0];
>> B = [4, 2, 0, 8, 0];
>> A&B      ans = 0 1 0 1 0
>> A|B      ans = 1 1 1 1 0
>> ~B       ans = 0 0 1 0 1
>> xor(A,B)  ans = 1 0 1 0 0
```

练习：

```
a = 5 ; b = 9;
c1 = (a<b)&&(b/a ==fix(b/a))
c2 = (a<b)|| (b/a ==fix(b/a))
```



# Matlab的输出显示

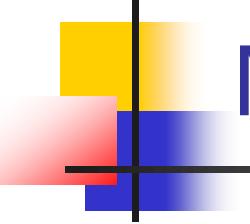
- 输出格式：

**Matlab** 以双精度执行所有的运算，但在命令行只显示五位。输出格式可以通过 **format** 命令指定。

**format** 只改变变量的输出格式， 但不会影响变量的值！

# 各种format格式

| 格式             | 解释                 | 例                      |
|----------------|--------------------|------------------------|
| format         | 短格式（缺省显示格式），同short | 3.1416                 |
| format short   | 短格式（缺省显示格式），只显示5位  | 3.1416                 |
| format long    | 长格式，双精度数15位，单精度数7位 | 3.1415926535897<br>9   |
| format short e | 短格式e方式（科学计数格式）     | 3.1416e+000            |
| format long e  | 长格式e方式             | 3.141592653589793e+000 |
| format short g | 短格式g方式             | 3.1416                 |
| format long g  | 长格式g方式             | 3.1415926535897<br>9   |
| format compact | 压缩格式               |                        |
| format loose   | 自由格式               |                        |



# Matlab的输出显示

■ 黄金分割比:  $\frac{\sqrt{5}-1}{2}$

```
>> x = (sqrt(5)-1)/2
```

```
x =
```

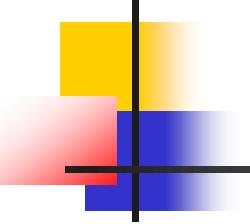
0.6180

```
>> format long
```

```
>> x
```

```
x =
```

0.618033988749895



# 变量存储与读取

- 将工作区中变量保存到文件
- 写入文件 **save**
- 从文件读 **load**
- 例：

```
>> x = 10*rand(3,4), y = 'hello';
```

```
>> save test.mat %将工作区中所有变量存入到当前  
%目录下名为test.mat的文件中。
```

```
>> clear %清除环境变量
```

```
>> load test.mat %读文件
```

# 变量存储与读取

- 将特定变量存入文件

**save test1.mat x**

文件名

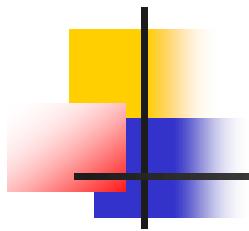
变量名

- 对同一文件再次存储时，原文件内容将被覆盖

**save test1.mat y**

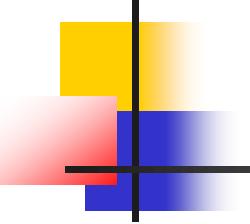
- 若不希望覆盖原文件内容，对文件追加存储

**save test1.mat -append y**



# Matlab帮助

- `help` 函数名
- `doc` 函数名
- `lookfor`
- `which`

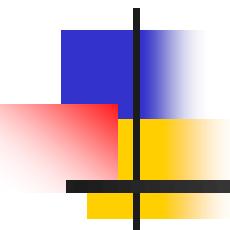


# 命令行常用指令

- clc
- clear
- more
- ctrl+c
- dir

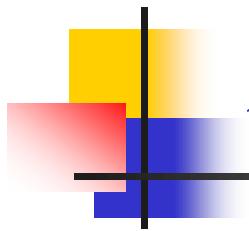
# Matlab编程与应用

## 第二讲



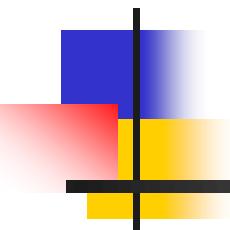
中国科技大学信息学院  
陆伟

[luwei@ustc.edu.cn](mailto:luwei@ustc.edu.cn)



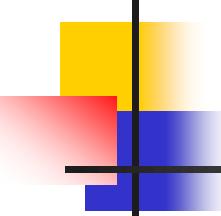
# 本讲内容：

- **Part1:** 数组与矩阵
- **Part2:** 脚本与函数
- **Part3:** 例子



Part1:

# Matlab数组与矩阵



# 数组与矩阵

## ■ matlab提供的一些矩阵生成函数：

**ones** %元素全为1的矩阵或数组

**zeros** %元素全为0的矩阵或数组

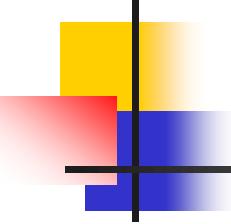
**eye** %对角线元素为1， 其他全为0的矩阵

**rand** %均匀分布的随机数

**randn** %高斯分布的随机数， 均值为0， 方差为1

**pascal** %由帕斯卡三角形得来的方阵

**magic** %行、列、对角线元素之和相等的方阵



# 数组与矩阵

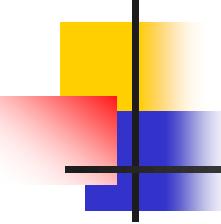
## ■ 数组寻址：

```
>> x=[1:10]+0.5)*10;  
>> x(4)  
>> x(3:8)  
>> x(0)      %数组下标从1开始!  
>> x(6:end)  
>> x(end-4:end)  
>> x(2,4,5)  %  
>> x([2,4,5])  
>> x([end:-1:1])  
>> x([1:2:end])
```

# 数组与矩阵

## ■ 矩阵寻址：

```
>> x =magic(5)  
x =  
    17     24      1      8     15  
    23      5      7     14     16  
     4      6     13     20     22  
    10     12     19     21      3  
    11     18     25      2      9  
>> x(3,2)  
>> x(2,:)  
>> x(2,2:5)  
>> x(:,1)  
>> x(:)  
>> x(:, :)
```



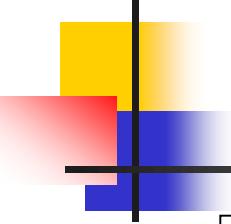
# 数组与矩阵

- 数组、矩阵寻址: 利用**find**函数

```
t = -10:0.1:10;  
>> x = sin(t);  
>> plot(t,x)
```

- 希望标出在绝对值大于0.7的x位置, 咋办?

```
ind = find(abs(x)>0.8);  
hold on;  
plot(t(ind),x(ind))
```



# 数组与矩阵

**reshape**

**repmat**

**cat**

**sort**

**max min mean find**

**flipud fliplr**

# 矩阵转置

■ 例：

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

```
A =
```

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |

```
A'
```

```
ans =
```

|   |   |
|---|---|
| 1 | 4 |
| 2 | 5 |
| 3 | 6 |

# 矩阵转置

例：

```
z = [1+2i  7-3i  3+4i;  6-2i  9i  4+7i]  
z =  
    1.0000 + 2.0000i    7.0000 - 3.0000i  
    3.0000 + 4.0000i  
    6.0000 - 2.0000i    0 + 9.0000i  
    4.0000 + 7.0000i  
z'  
ans =  
    1.0000 - 2.0000i    6.0000 + 2.0000i  
    7.0000 + 3.0000i    0 - 9.0000i  
    3.0000 - 4.0000i    4.0000 - 7.0000i
```

**A'** 是共轭转置，及**A<sup>H</sup>**！

# 矩阵转置

例：

```
Z = [1+2i  7-3i  3+4i;  6-2i  9i   4+7i]  
Z =  
    1.0000 + 2.0000i  7.0000 - 3.0000i  
    3.0000 + 4.0000i  
    6.0000 - 2.0000i  0 + 9.0000i  
    4.0000 + 7.0000i  
Z.'  
ans =  
    1.0000 + 2.0000i  6.0000 - 2.0000i  
    7.0000 - 3.0000i  0 + 9.0000i  
    3.0000 + 4.0000i  4.0000 + 7.0000i
```

A.' 得到  $\mathbf{A}^T$  !

# 矩阵指数运算

```
A = [1 1 1; 1 2 3; 1 3 6]
```

```
X = A^2
```

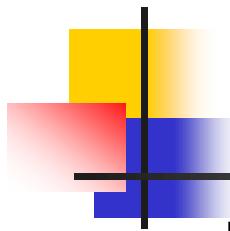
```
X =
```

|    |    |    |
|----|----|----|
| 3  | 6  | 10 |
| 6  | 14 | 25 |
| 10 | 25 | 46 |

```
Y = A.^2
```

```
Y =
```

|   |   |   |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 4 | 9 |



# 矩阵幂运算

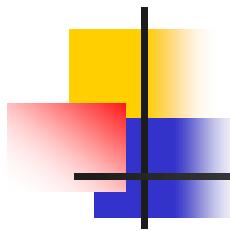
**B= expm (A) :**

$$B = e^A = I + \frac{1}{1!} A + \frac{1}{2!} A^2 + \frac{1}{3!} A^3 + \dots$$

**exp (A)** %逐个元素进行指数运算

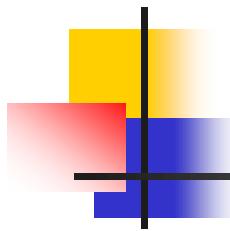
# 矩阵幂运算

```
A = [1 1 1;1 2 3;1 3 6]
X = expm(A)
X =
    1.0e+003 *
    0.1008      0.2407      0.4368
    0.2407      0.5867      1.0654
    0.4368      1.0654      1.9418
Y =exp (A) %逐个元素进行指数运算
Y =
    2.7183      2.7183      2.7183
    2.7183      7.3891     20.0855
    2.7183     20.0855   403.4288
```



# 矩阵的行列式、秩、迹

- 行列式:  $\det(\mathbf{A})$
- 秩: 矩阵线性无关的行数或列数;  
 $\text{rank}(\mathbf{A})$
- 迹: 矩阵的迹等于矩阵的对角线元素之和, 也等于矩阵的特征值之和;  
 $\text{trace}(\mathbf{A})$



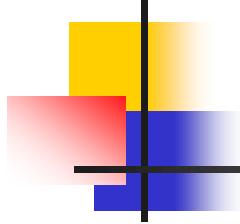
# 矩阵求逆

- **inv(A)**
- 矩阵A的逆矩阵表示为 $A^{-1}$ , 满足一下恒等式:

$$AA^{-1} = I$$

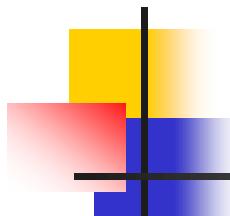
$$A^{-1}A = I$$

- 只有在A为方阵且满秩时,  $A^{-1}$ 才存在。



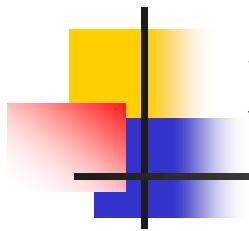
## 矩阵的特征值与特征向量

- $d = \text{eig}(A)$
- $[V, D] = \text{eig}(A)$



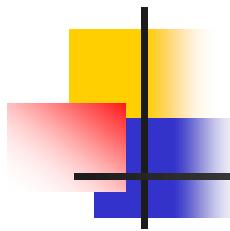
# 矩阵分解

- LU分解,  $A=LU$ , 利用高斯消元法, L为对角线为1的下三角矩阵, U为上三角矩阵。
- 奇异值分解 :
- QR分解:  $A=QR$ , Q为正交矩阵, R为上三角矩阵
- Cholesky分解: $A=R'R$ , A为正定矩阵 , R为上三角矩阵
- $[L,U] = \text{lu}(A)$
- $S = \text{svd}(A)$
- $[Q,R] = \text{qr}(A)$
- $\text{chol}(A)$



# 线性方程组求解

- $Ax = B \Rightarrow x = A \setminus B$  左除
- $Ax = B \Rightarrow x = A^{-1}B$



# 例：城市人口迁移问题

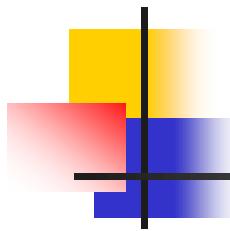
- 有甲乙丙丁四个城市间人口互相迁移，从甲->乙的人口数占甲当年总人口数量的18% ( $a_{21}$ )

$$P = \begin{bmatrix} 0.32 & 0.17 & 0.11 & 0.46 \\ 0.18 & 0.43 & 0.32 & 0.33 \\ 0.27 & 0.22 & 0.39 & 0.14 \\ 0.23 & 0.18 & 0.18 & 0.07 \end{bmatrix}$$

设开始时每个城市人口为：

甲 10000 乙 30000 丙 50000 丁 80000

问20年后各个城市人口数量为多少？



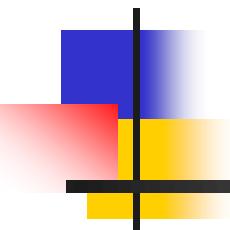
## 例：城市人口迁移问题

- 概率转移矩阵  $P = [a_{ij}]$ ;

$$a_{ij} \geq 0$$

$$\sum_{j=1}^N a_{ij} = 1$$

如  $P = \begin{bmatrix} 0.4 & 0.7 \\ 0.6 & 0.3 \end{bmatrix}$



Part2:

---

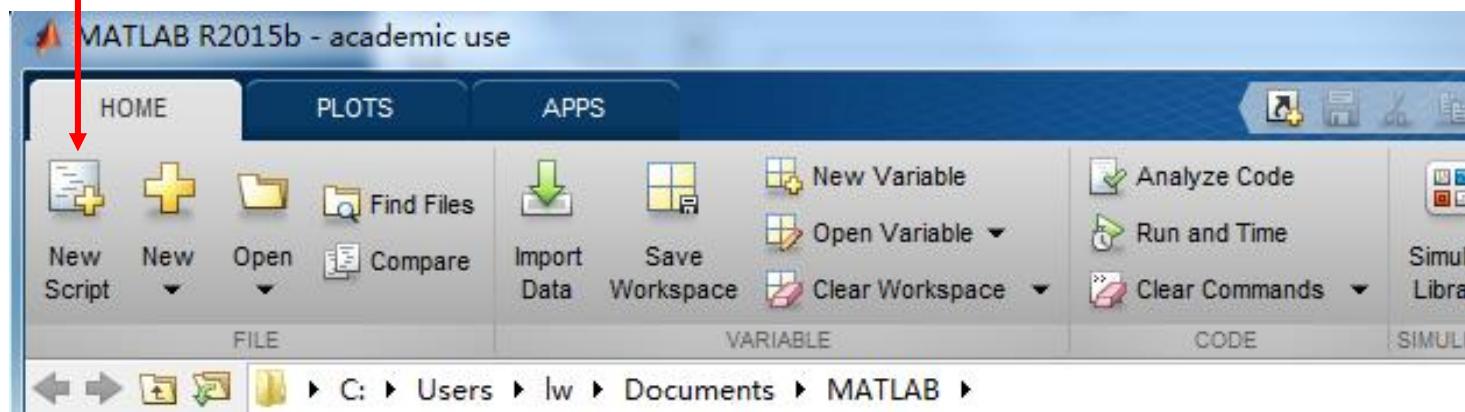
# 脚本与函数

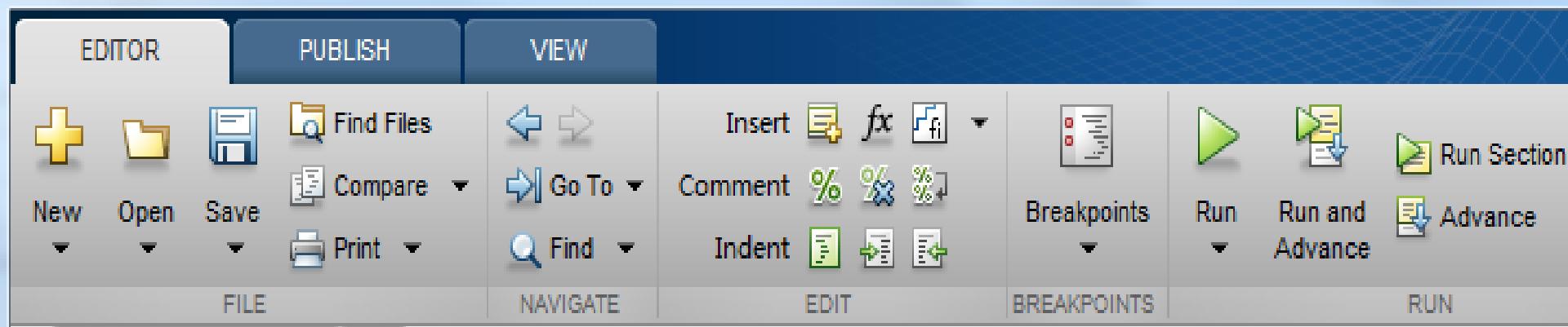
# matlab脚本文件

## ■ 脚本文件 (Script m file):

- ◆ 一串指令的集合；
- ◆ 执行结果与在命令窗口逐行输入执行结果一样；
- ◆ 没有输入输出参数。

新建脚本文件



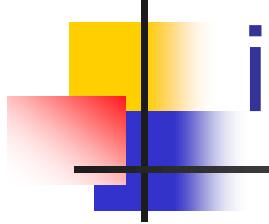


```
scriptdemo1.m* %This is a demo script m file
%
% 2017.6.24
fs = 100; %采样频率
f0 = 2;
t = 0: 1/fs : 5;
x = exp(-t).*cos(2*pi*2*t); %信号
plot(t,x);
```

Breakpoint  
(调试断点)

Help信息

注释



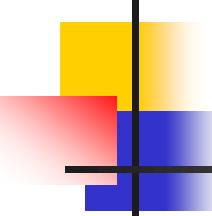
# input语句

- `user_entry = input('prompt')`

prompt是程序在命令窗口对用户的提示，用户输入内容被赋给变量`user_entry`.

- `user_entry = input('prompt', 's')`

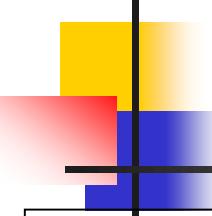
用户输入的内容作为字符串赋给变量  
`user_entry`.



# input语句

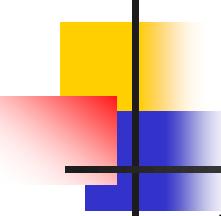
- 例：求一元二次方程 $ax^2 + bx + c = 0$ 的根

```
a=input('a=?');
b=input('b=?');
c=input('c=?');
d=b*b-4*a*c;
x1=(-b+sqrt(d))/(2*a)
x2=(-b-sqrt(d))/(2*a)
```



# input语句

```
reply=input('Do you want more? Y/N [Y]: ','s');
if isempty(reply)
    reply = 'Y';
end
if reply == 'Y'
    disp('You selected Yes');
else
    disp('You selected No');
end
```



# 流程控制—for语句

```
for 循环变量 = 表达式  
    循环体  
end
```

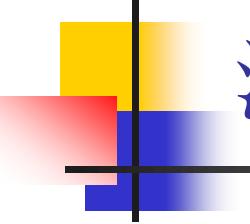
- 表达式一般为一个向量，循环变量被依次赋予向量中每个元素的值，并执行循环体。
- 表达式如  $m:s:n$ , 其中  $m, s, n$  都可以为整数，小数，负数。

如 `for k = -1:0.1:1;`  
`for k = 10:-1:0`

## 流程控制—for语句

例：已知  $y = 1 + \frac{1}{3} + \frac{1}{5} + \cdots + \frac{1}{2n-1}$ , 当  $n=100$  时，求  $y$  的值

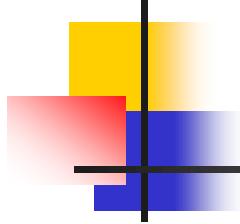
```
clear;
y=0; n=100;
for k=1:n
    y=y+1/(2*k-1);
end
```



# 流程控制—while语句

```
while 表达式  
    循环体  
end
```

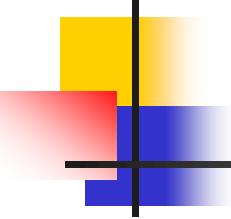
- 表达式一般由逻辑运算和关系运算等组成
- 只要表达式值不为零，即逻辑“真”，程序就继续循环；当表达式值为0就停止循环



## 流程控制—while语句

- 例：用while循环求1~100间整数的和。

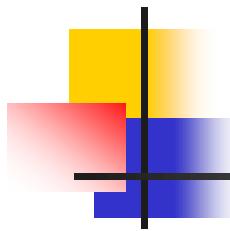
```
sum = 0;  
  
i = 1;  
  
while i<=100  
  
    sum = sum+i;  
  
    i = i +1;  
  
end
```



## 流程控制—while语句

- 例：用while循环求matlab的eps。

```
myeps = 1;  
  
while 1~= (1+myeps)  
  
myeps = myeps/2;  
  
end  
  
myeps = myeps*2
```



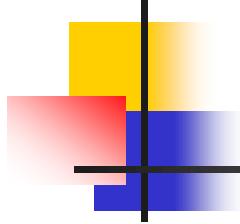
# 流程控制—if语句

## ■ 分支结构 if语句

```
if 条件表达式  
    执行语句  
end
```

```
if 条件表达式  
    执行语句  
else  
    执行语句  
end
```

```
if 条件表达式  
    执行语句  
elseif  
    执行语句  
else  
    执行语句  
end
```

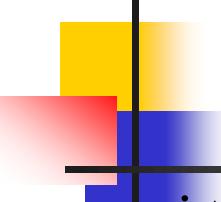


## 流程控制—if语句

- 例：输入一个字符，若为大写字母，则输出其对应的小写字母；若为小写字母，则输出其对应的大写字母；若为数字字符则输出其对应的数值，若为其他字符则原样输出。

# 流程控制—if语句

```
c=input('请输入一个字符','s');
if c>='A' & c<='Z'
    disp(setstr(abs(c)+abs('a')-
abs('A')));
elseif c>='a' & c<='z'
    disp(setstr(abs(c)-
abs('a')+abs('A'))));
elseif c>='0' & c<='9'
    disp(abs(c)-abs('0'));
else
    disp(c);
end
```



# 流程控制--switch语句

switch语句：根据表达式的取值不同，分别执行不同的语句

**switch** 表达式

**case** 表达式1

        执行语句1

**case** 表达式2

        执行语句2

.....

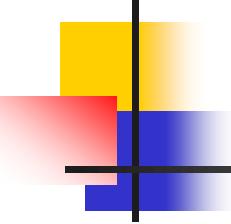
**case** 表达式m

        执行语句m

**otherwise**

        执行语句n

**end**



## 流程控制--switch语句

- 某商场对顾客所购买的商品实行打折销售，标准如下（商品价格用price来表示）：

|           |      |
|-----------|------|
| price<200 | 没有折扣 |
|-----------|------|

|               |      |
|---------------|------|
| 200≤price<500 | 3%折扣 |
|---------------|------|

|                |      |
|----------------|------|
| 500≤price<1000 | 5%折扣 |
|----------------|------|

|                 |      |
|-----------------|------|
| 1000≤price<2500 | 8%折扣 |
|-----------------|------|

|                 |       |
|-----------------|-------|
| 2500≤price<5000 | 10%折扣 |
|-----------------|-------|

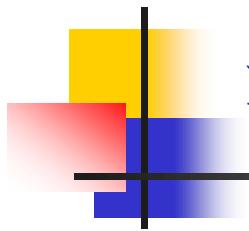
|            |       |
|------------|-------|
| 5000≤price | 14%折扣 |
|------------|-------|

输入所售商品的价格，求其实际销售价格。

# 流程控制--switch语句

```
price=input('请输入商品价格');
switch fix(price/100)
    case {0,1}                                %价格小于200
        rate=0;
    case {2,3,4}                                %价格200~500
        rate=3/100;
    case num2cell(5:9)                          %价格500~1000
        rate=5/100;
    case num2cell(10:24)                         %价格1000~2500
        rate=8/100;
    case num2cell(25:49)                         %价格2500~5000
        rate=10/100;
otherwise

```



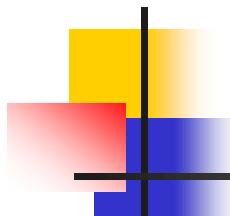
# 流程控制—continue、break语句

- **continue:** 跳过循环体中某些语句，继续下一个循环。
- **break:** 终止循环执行。执行脚本或函数中下一个语句。

# 流程控制—continue、break语句

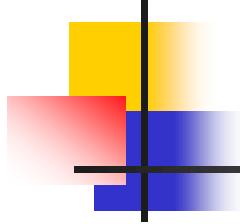
```
for k =1:5
    if k ==3
        continue
    end
    k
end
disp('The end of Loop')
■ 输出：1 2 4 5
```

```
for k =1:5
    if k ==3
        break
    end
    k
end
disp('The end of Loop')
■ 输出：1 2
```



# 脚本文件与函数文件

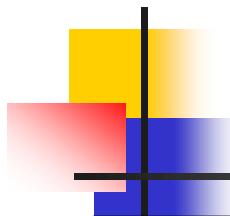
- 脚本文件 (**Script m file**) 实际上是一串指令的集合。执行结果与在命令窗口逐行输入执行结果完全一样。没有输入输出参数。
- 函数文件(**function m file**)一般有输入参数与输出参数。



## 自定义函数

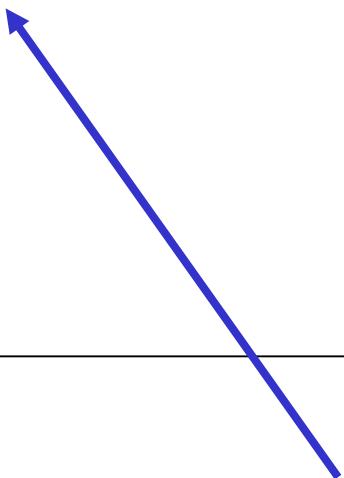
例：建立一个函数文件计算  $\sin(x^2)$

```
function y = my1stfunc(x)
    z = x.^2;
    y = sin(z);
```



## 自定义函数

```
function y = my1stfunc(x)
    z = x.^2;
    y = sin(z);
```



函数文件第一行 格式

# 自定义函数

```
function y = my1stfunc(x)
    z = x.^2;
    y = sin(z);
```

函数名

函数保存的文件名必须与函数名相同！即  
该函数必须被保存在**my1stfunc.m** 中

# 自定义函数

```
function y = my1stfunc(x)
    z = x.^2;
    y = sin(z);
```

输入变量

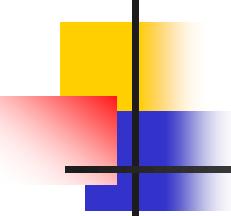
函数可以有多个输入变量，也可以没有输入变量

# 自定义函数

```
function y = my1stfunc(x)
    z = x.^2;
    y = sin(z);
```

输出变量

函数可以有多个输出变量，也可以没有输出变量



# 自定义函数

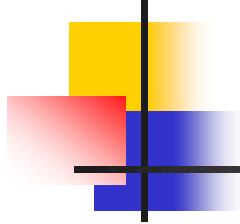
- 在函数中定义的变量为**局部变量**，存储在单独的内存工作区内，不被调用的程序所见。

Script

```
a = 1  
b = f(2)  
c = 3
```

function

```
function y = f(x)  
z = 2*x  
y = z+1
```



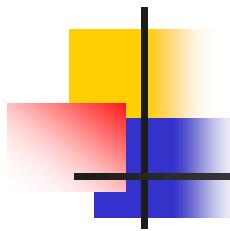
# 自定义函数

Script

- **$z = 1$**
- **$x = f(2)$**
- **$y = 3$**

function

```
function y = f(x)
z = 2*x
y = z+1
```



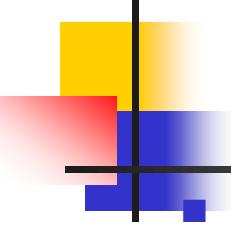
# 自定义函数

## ■ 练习：

```
x = 1;  
x = f(x+1);  
y = x+1
```

```
function y = f(x)  
x = x+1;  
y = x+1;
```

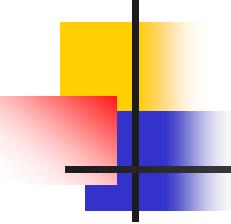
最终 y = ?



# 自定义函数

确定输入、输出变量的数目 **nargin, nargout**

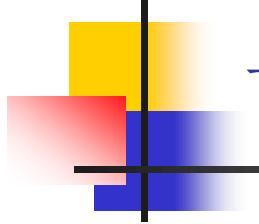
```
function [x0, y0] = myplot(x, y, npts,
angle, subdiv)
if nargin < 5, subdiv = 20; end
if nargin < 4, angle = 10; end
if nargin < 3, npts = 25; end
...
if nargout == 0
    plot(x, y)
else
    x0 = x;
    y0 = y;
end
```



# 自定义函数

- 函数可以在命令行被调用，也可以在别的函数文件或脚本文件中调用。
- 函数必须在当前目录下或者其所在目录位于**Matlab**的搜索路径中。

```
>> result = my1stfunc(3)
```



# 子函数

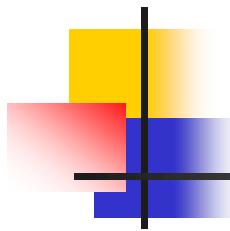
- 子函数：在一个函数文件中可以包含多个函数，与函数文件名相同的是主函数，其它为子函数。
- 子函数只能被函数文件内主函数或其它子函数调用

# 子函数

- 例：创建一个函数，输入两个数，输出两个数加、减后的结果

```
function [rlt_add, rlt_sub]=sfuncdemo(x,y)
%主函数
    rlt_add = add(x,y);
    rlt_sub = subtract(x,y);
function result = add(x,y)      %子函数
    result = x+y;
function output = subtract(x,y)    %子函数
    output = x-y;
```

- m文件**文件名必须和主函数名相同**，即  
`sfuncdemo.m`



# 函数句柄 (@)

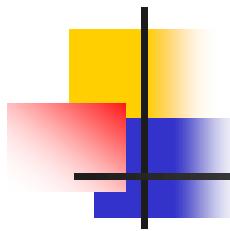
变量名=@(输入参数列表)运算表达式

例: >>**sqr** = @(x) x.^2; %创建

>> a = **sqr**(3) %调用

例: >>**ln** = @(x) log(x); %创建

>> a = **ln**(3) %调用

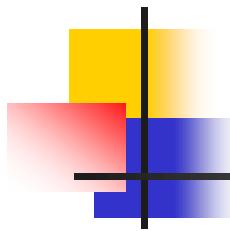


# 函数句柄 (@)

- 可以为matlab内建函数创建句柄。

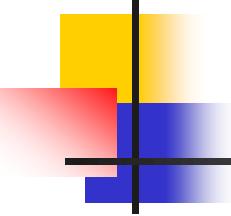
```
例: >>hd_sin = @sin; %创建  
>> a = hd_sin(pi)      %调用
```

- 可以为用M文件创建的自定义函数创建句柄



# 函数句柄 (@)

- 提高函数调用速度：matlab调用函数时每次都是要搜索所有的路径，如果一个函数在程序中需要经常用到，使用函数句柄，可以提高程序速度。
- 当matlab关闭或工作区被清空（clear），利用函数句柄创建的函数失效。

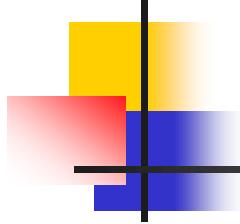


# 内联函数(inline)

- 变量名=inline('函数表达式', '变量名1', ... , '变量名n');

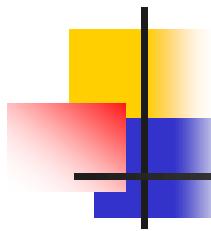
```
>> f=inline ('x+y' , 'x' , 'y');
>>f (2 , 3)
ans = 5

>> Fofx=inline ('x .^2*cos (a*x) -
b' , 'x' , 'a' , 'b' );
>> g= Fofx ([pi/3 pi/3.5] , 4 , 1)
ans= -1.5483 -1.7259
```



# 函数调试

- 在matlab的m文件编辑器中设置断点进行Debug。
- pause
- keyboard

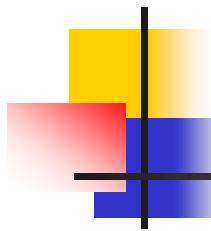


# 程序优化

- 使循环向量化

```
clear  
tic  
for t = 1:100000  
    y(t) = sin(t);  
end  
toc
```

```
clear  
tic  
t = 1:100000;  
y=sin(t);  
toc
```

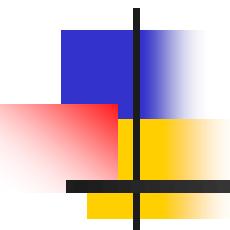


# 程序优化

- 为数组预先分配内存

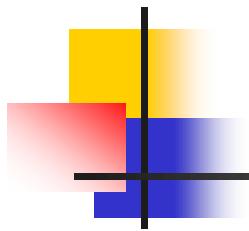
```
clear  
tic  
y = 0;  
for k = 2:1e8  
    y(k) = y(k-1)+1;  
end  
toc
```

```
clear  
tic  
y = zeros(1,1e8);  
for k = 2:1e8  
    y(k) = y(k-1)+1;  
end  
toc
```



## Part 3:

### 常用信号产生



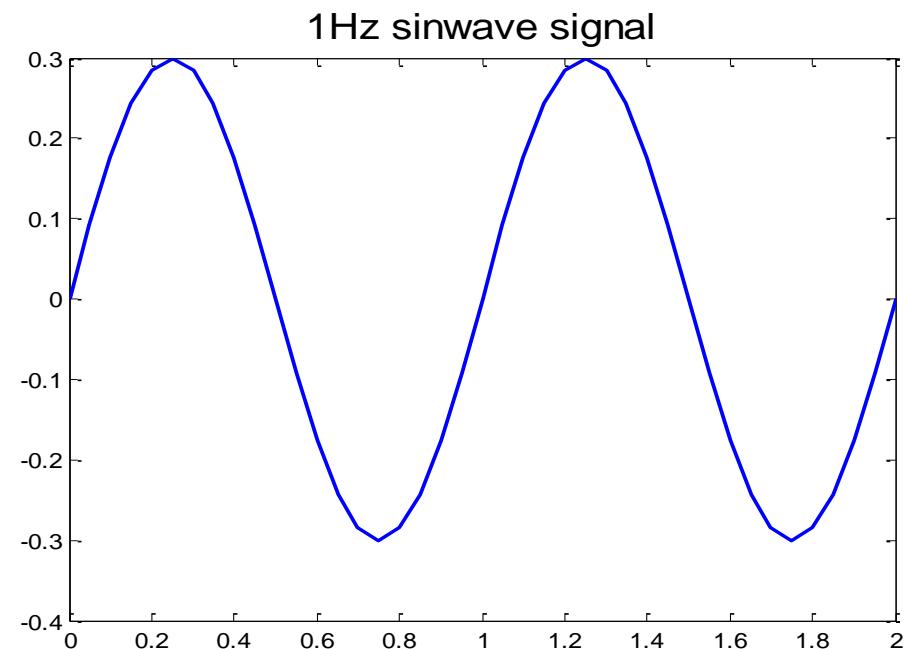
# 常用信号产生

# 正弦信号

- 产生频率为1Hz的正弦信号

$$x(t) = 0.3 \sin(2\pi f_0 t) \quad f_0 = 1\text{Hz}$$

```
f0 = 1;  
fs = 20;  
t = 0:1/fs:2;  
x =  
0.3*sin(2*pi*t);  
plot(t,x)
```

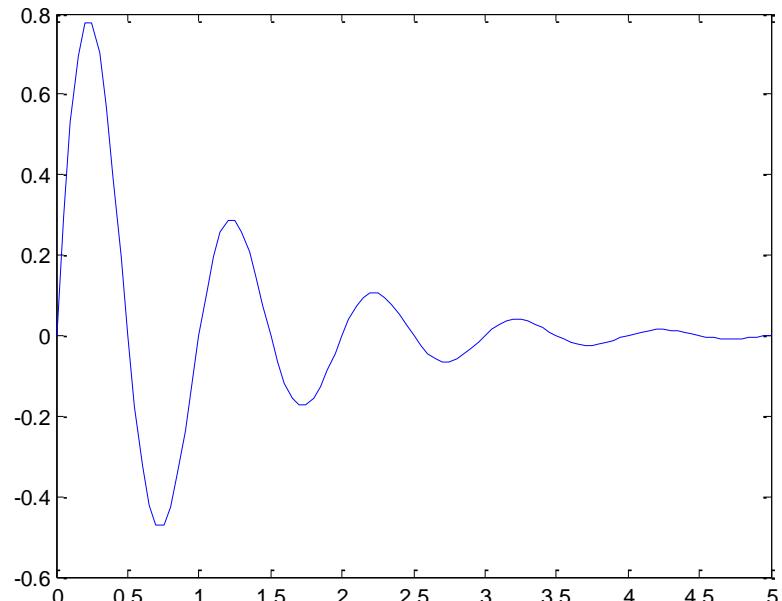


# 指数衰减的正弦信号

- 产生频率为1Hz的幅度呈指数衰减的正弦信号

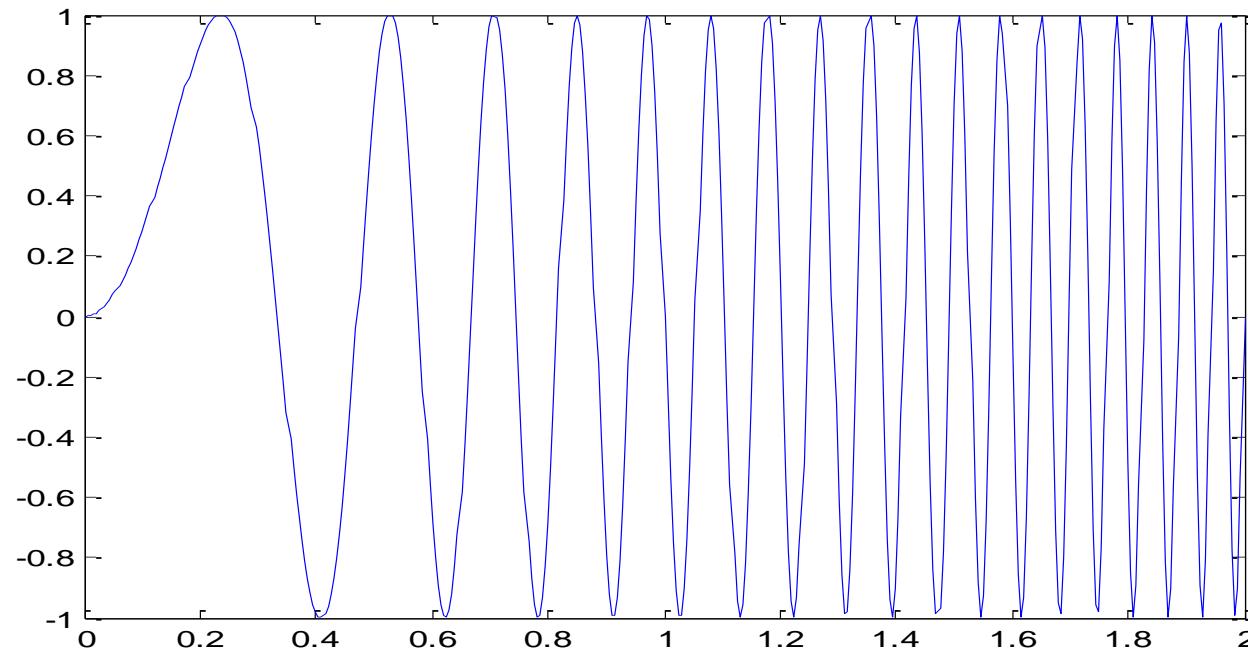
$$x(t) = e^{-t} \sin(2\pi f_0 t) \quad f_0 = 1\text{Hz}$$

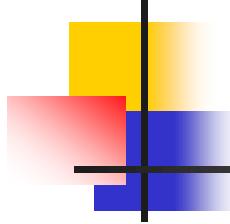
```
f0 = 1;  
fs = 20;  
t = 0:1/fs:5;  
x = exp(-t).*sin(2*pi*t);  
plot(t,x)
```



# 线性调频信号

- LFM(Linear Frequency Modulation)信号又称 chirp 信号，在雷达信号检测中用广泛应用。





# 线性调频信号

- 调频信号：瞬时频率是时间的函数

$$x(t) = \sin(2\pi f(t)t)$$

$$f(t) = f_0 + \beta t \quad \beta = (f_1 - f_0)t / (t_1 - 0)$$

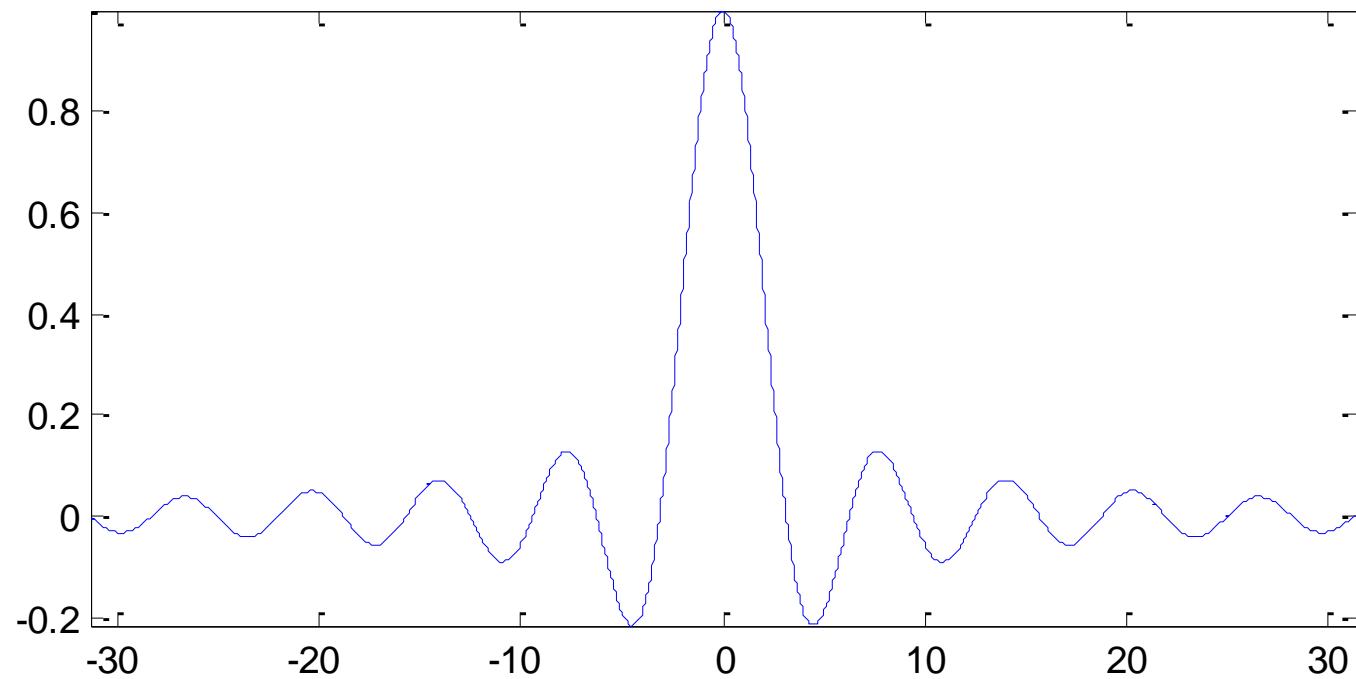
$f_0$  : 初始  $t=0$  时刻的瞬时频率;

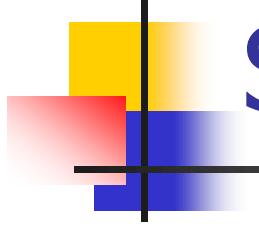
$f_1$  :  $t=t_1$  时刻的瞬时频率;

练习：试写出  $t=0 \sim 2s$ ,  $f(t) = 1 \sim 10Hz$  的线性调频信号

# Sinc信号

$$\frac{\sin(x)}{x}$$



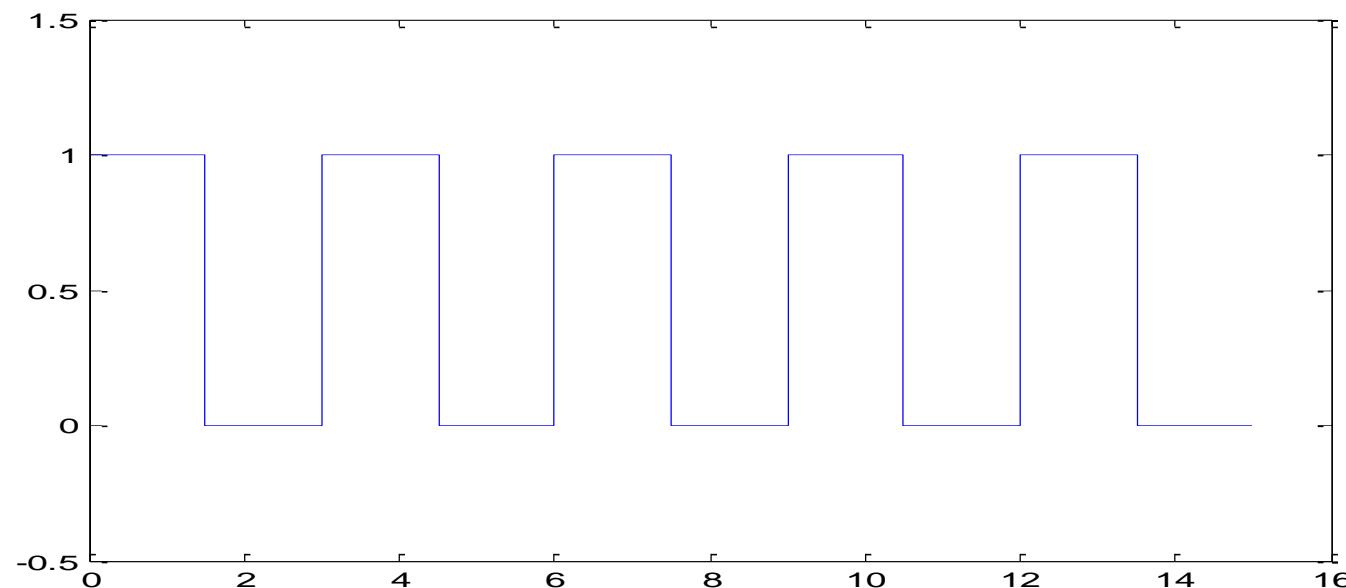


# Sinc信号

```
t = -(10*pi):0.01*pi:(10*pi);  
x = sin(t)./t;  
L = length(t);  
x((L+1)/2) = 1;  
plot(t,x)  
axis tight
```

# 连续周期信号产生

- 产生5个周期的方波信号，周期为3s, 占空比为50%，幅度为1。

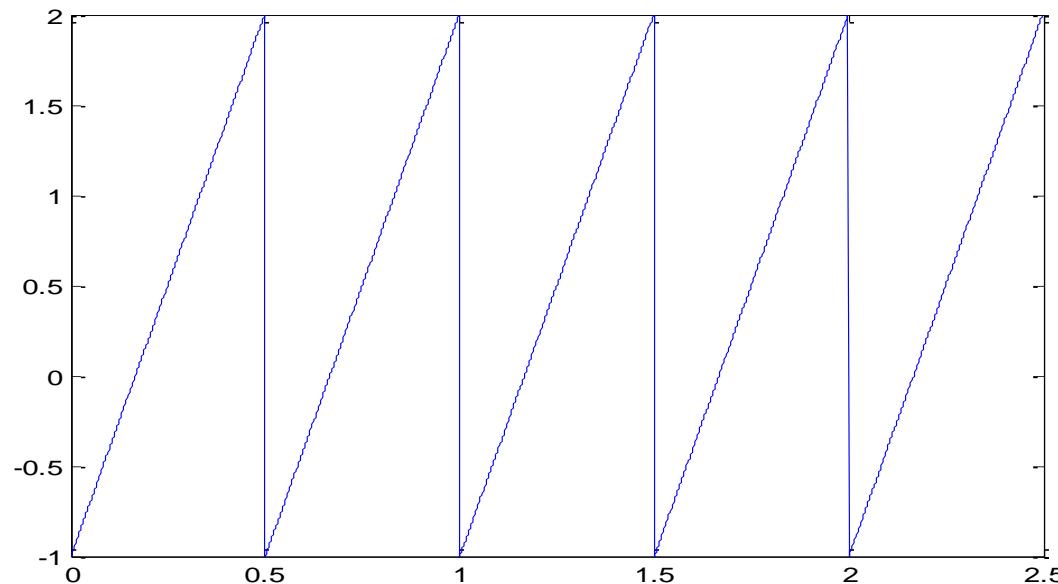


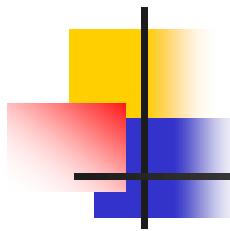
# 连续周期信号产生

```
N = 5          %5个周期
fs = 500;    %采样频率
T = 3;        %周期
t = 0:1/fs: T*N-1/fs;
%先产生一个周期
len1 = ones(1,floor(T*fs*0.5));
len0=zeros(1,floor(T*fs*(1-0.5)));
sig1 = [len1,len0];
%重复5次
sig = repmat(sig1,1,5);
plot(t,sig)
axis([0 16 -0.5 1.5])
```

# 连续周期信号产生

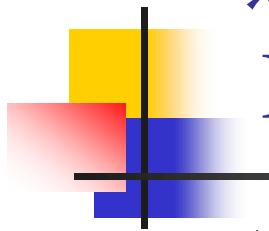
- 练习：产生5个周期的锯齿波信号，周期为0.5s, 幅度为从-1到+2。





# 随机信号的产生

- **rand** [0 1]之间均匀分布的随机信号
- **randn** 高斯分布的随机信号，均值为0，方差为1



# 连续时间周期信号的 Fourier 级数展开与合成

- 设实信号  $x(t)$  的周期为  $T$ ,  $\omega_0 = 2\pi f_0 = 2\pi/T$   
则  $x(t)$  可以展开为一组成谐波关系的正弦波的线性组合。

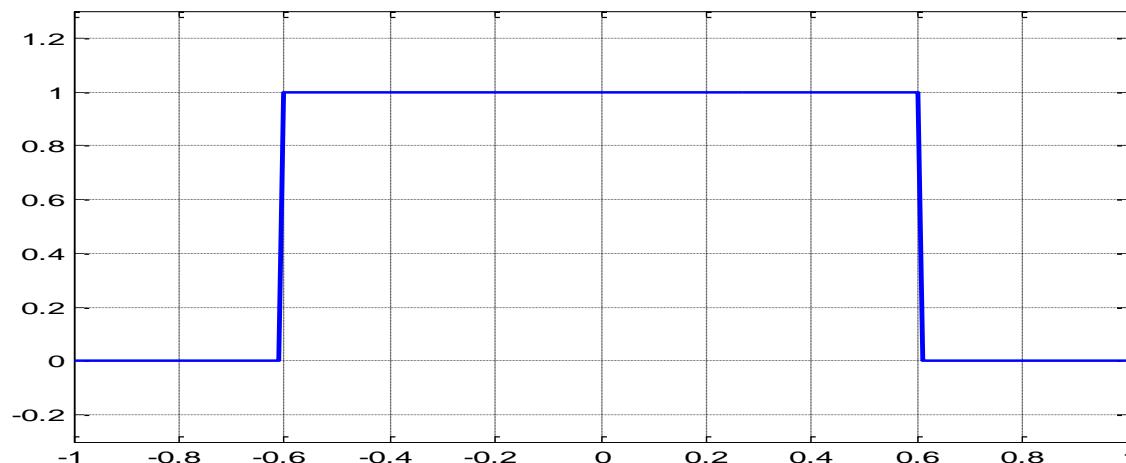
$$x(t) = \sum_{k=-\infty}^{+\infty} a_k e^{jk\omega_0 t} = a_0 + \sum_{k=1}^{+\infty} 2|a_k| \cos(k\omega_k t + \theta_k)$$

$$a_k = \frac{1}{T} \int_T x(t) e^{-jk\omega_k t} dt$$

# 连续时间周期信号的 Fourier级数展开与合成

- 设有周期 $T=2$ , 占空比60%的对称周期方波

$$a_k = \frac{\sin(0.6k\pi)}{k\pi} \quad x(t) = 1.2 + \sum_{k=1}^{+\infty} 2|a_k| \cos(k\pi t + \theta_k)$$



# 连续时间周期信号的 Fourier级数展开与合成

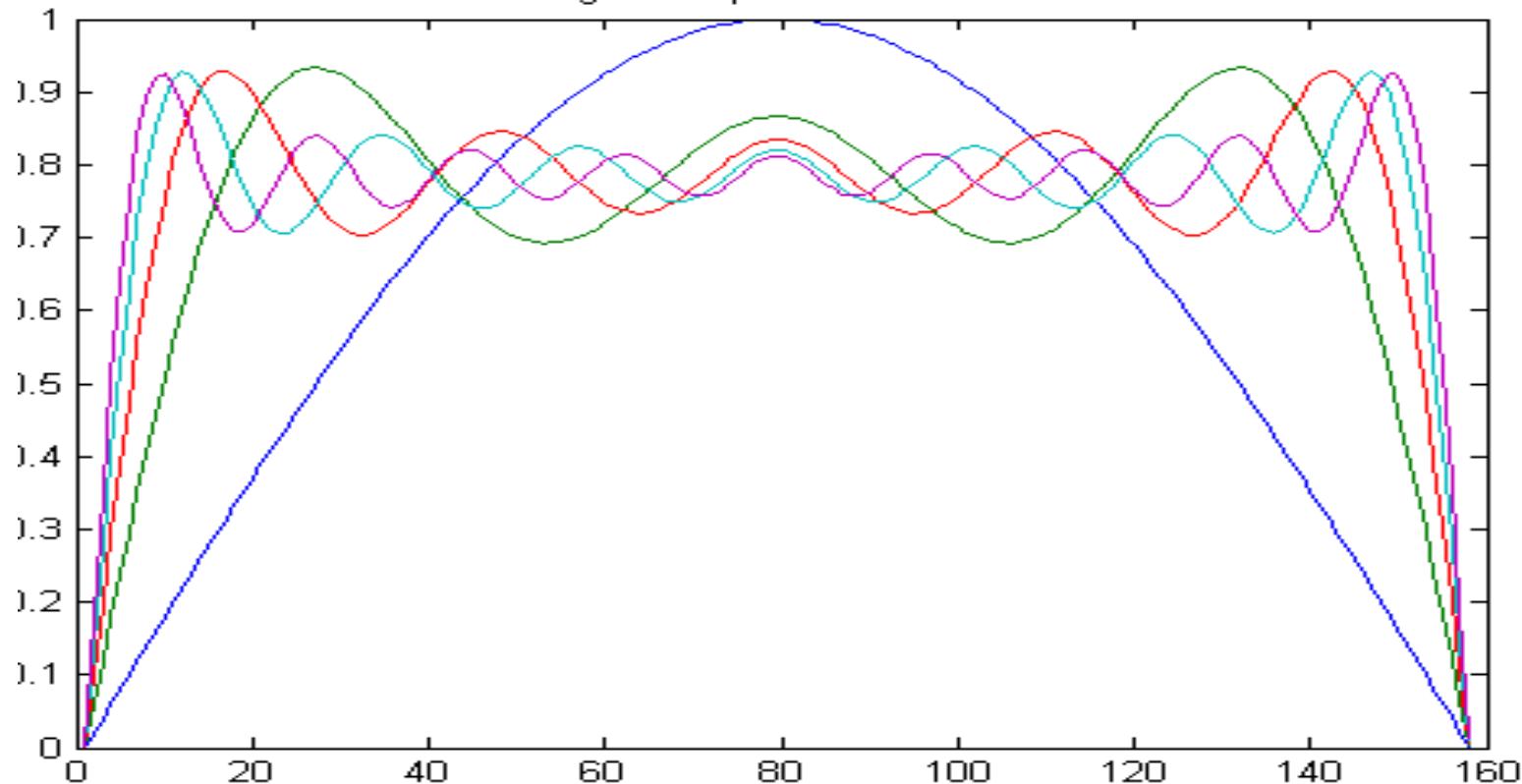
```
T = 2;
w0 = 2*pi/T;
F0 = 1/T;
fs = 100;
t1 = -1:1/fs:(-0.6-1/fs);
t2 = -0.6:1/fs:0.6;
t3 =(0.6+1/fs):1/fs:1;
x =[zeros(1,length(t1)),ones(1,length(t2)),...
zeros(1,length(t3))];
t = -1:1/fs:+1;
plot(t,x,'linewidth',2)
axis([-1 1,-0.3 1.3]),grid on,hold on
```

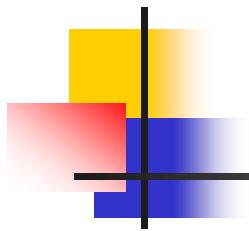
# 连续时间周期信号的 Fourier级数展开与合成

```
N = 40;
y=0.6;
yall=x;
for k = 1:N
a = sin(k*pi*0.6)/(k*pi);
absa=abs(a);
anglea = pi*(a < 0);
y = y + 2*absa*cos(k*pi*t+anglea);
plot(t,y),shg
pause(2)
end
```

# Gibbs现象

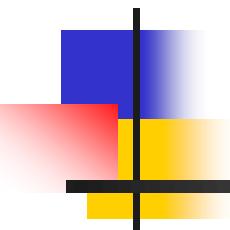
The building of a square wave: Gibbs' effect





# matlab的一些信号产生函数

- tripuls
- rectpuls
- gauspuls
- sawtooth
- pulstran
- chirp
- diric

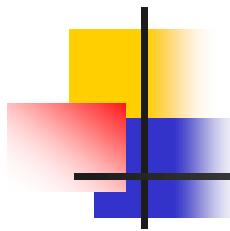


# Matlab编程与应用

## 第三讲

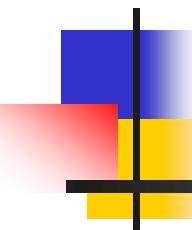
中国科技大学信息学院  
陆伟

[luwei@ustc.edu.cn](mailto:luwei@ustc.edu.cn)



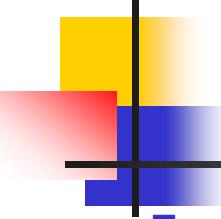
# 本讲内容

- part1: 二维绘图
- part2: 三维绘图
- part3: 动态显示



Part1:

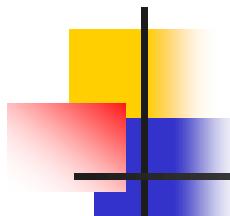
二维绘图



# 主要内容

## ■ 基本绘图函数： **plot** 函数

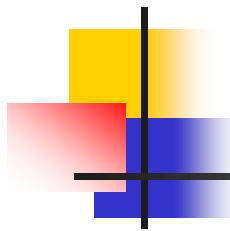
- 线型（Line Style）、点型（Marker）、颜色（Color）
- 二维绘图的辅助操作
  - 标注：图形名称、坐标轴名称、曲线标注、图例
  - 坐标轴控制
  - 图形保持（同一坐标轴绘制多个图形）
  - 窗口分割（同一窗口含有多个坐标轴）
  - 复数情况



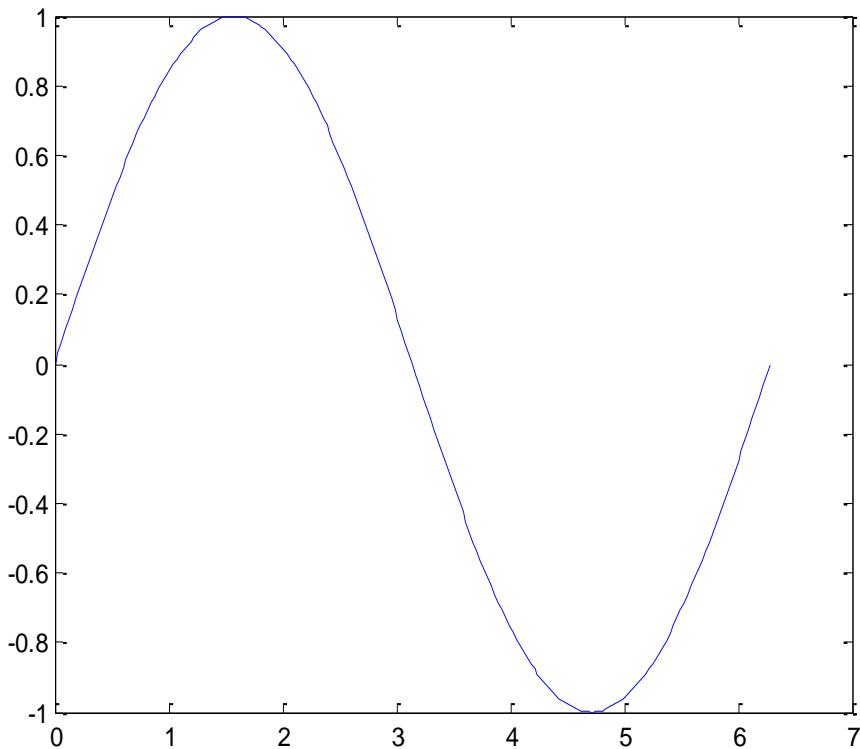
# 基本绘图函数-- plot

- **plot**: 最基本的绘图指令
- 对x坐标及对应的y坐标绘图
- 例:

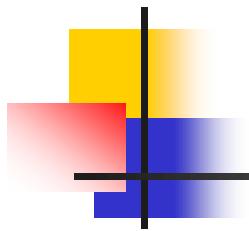
```
x = 0 : 0.01*pi : 2*pi ;  
y = sin(x) ;  
plot(x,y) ;
```



# plot基本绘图



- 要求x向量、y向量  
**长度必须相同！**
- 若只给定一个向量，则横坐标是索引值，  
如：`plot(y)`

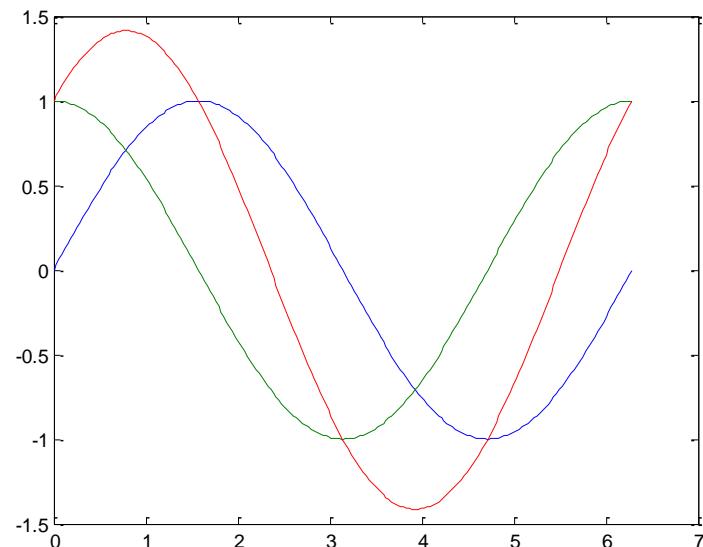


# plot基本绘图

- 一次画出多条曲线
- 例：

```
x = 0 : 0.01*pi : 2*pi ;  
y1 = sin(x);  
y2 = cos(x);  
y3 = sin(x)+cos(x);  
plot(x, y1, x, y2, x, y3);
```

# plot基本绘图



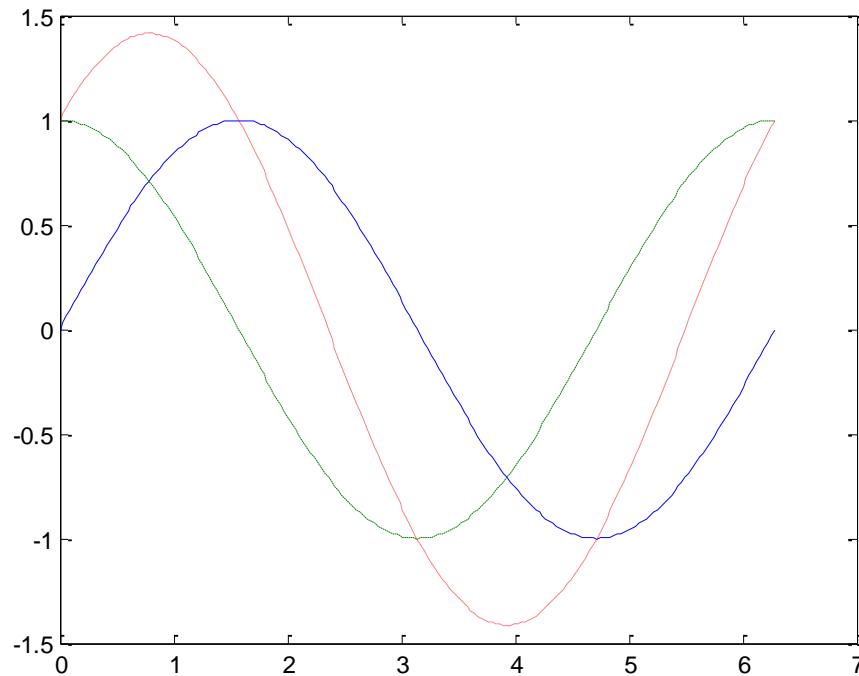
- 画多条曲线时，Matlab自动赋予每条曲线不同的颜色。
- 也可以把 $y_1$ 、 $y_2$ 、 $y_3$ 组成一个矩阵，此时会对矩阵的每个行向量作图。

```
z = [y1 ; y2 ;y3];  
plot(x,z);
```

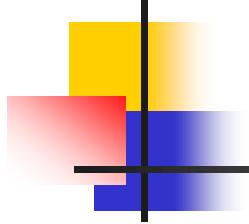
# plot基本绘图

## —设置曲线的线型 (Line Style)

例：



```
plot(x,y1,'-',x,y2,'--',x,y3,:');
```



# plot 基本绘图

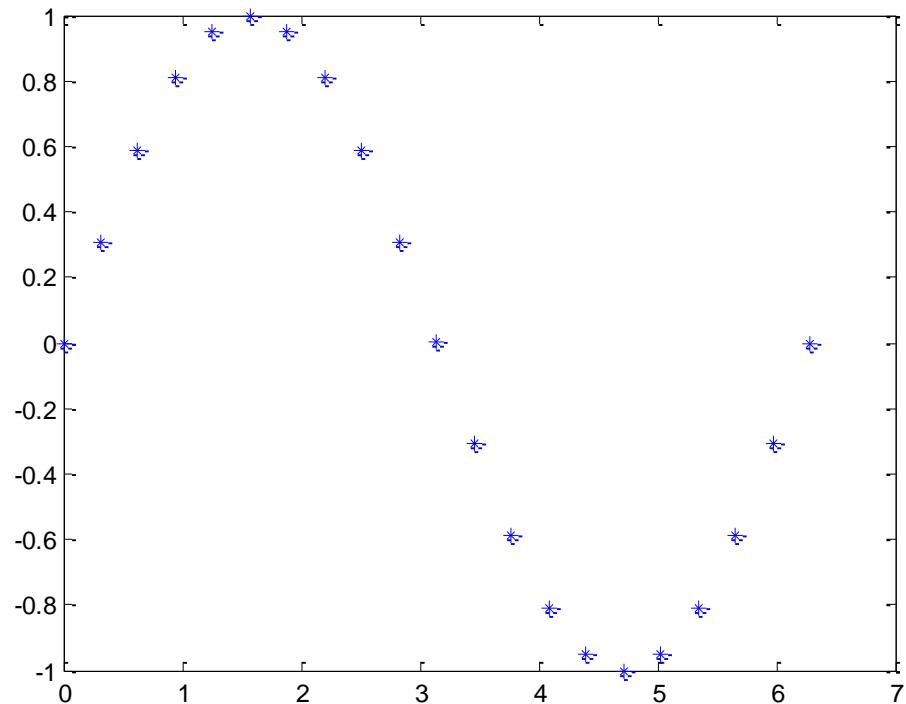
## — 设置曲线的线型 (Line Style)

| plot 函数的 Line Style | 说明      |
|---------------------|---------|
| -                   | 实线(默认值) |
| --                  | 虚线      |
| :                   | 点线      |
| -.                  | 点虚线     |

# plot基本绘图

## —设置曲线的点型 (Marker)

```
x = 0 : pi /10 : 2*pi;  
y = sin(x);  
plot(x,y,'*')
```

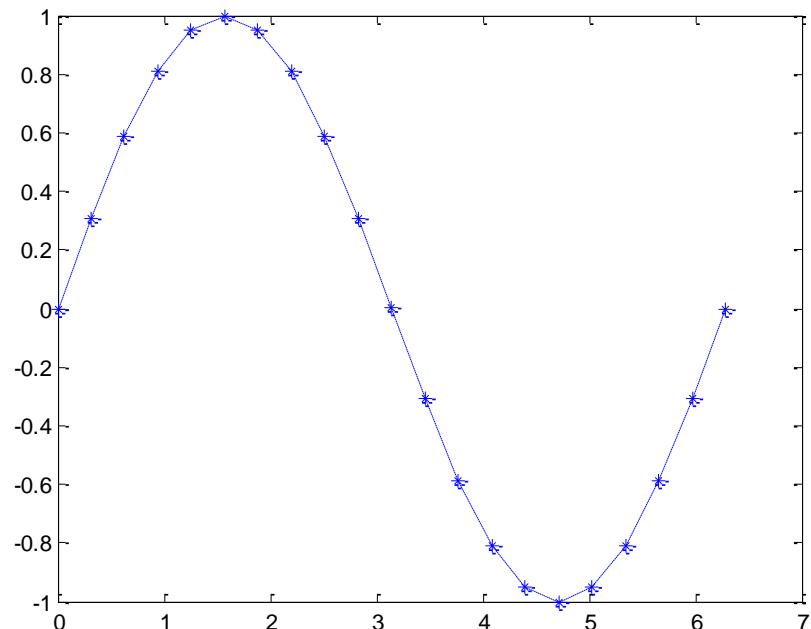


# plot基本绘图

## —设置曲线的点型 (Marker)

- 也可以同时设置画线与点型

```
x = 0 : pi /10 : 2*pi;  
y = sin(x);  
plot(x,y,'*-')
```



# plot 基本绘图

## — 设置曲线的点型 (Marker)

plot 函数的点型 (Marker)

说明

O

圆形

+

加号

X

叉号

\*

星号

.

点号

^

朝上三角形

V

朝下三角形

# plot 基本绘图

## — 设置曲线的点型 (Marker)

plot 函数的点型 (Marker)

说明

>

朝右三角形

<

朝左三角形

square

方形

diamond

菱形

pentagram

五角星形

hexagram

六角星形

None

无点型(默认值)

# plot基本绘图

## —设置曲线的颜色 (Color)

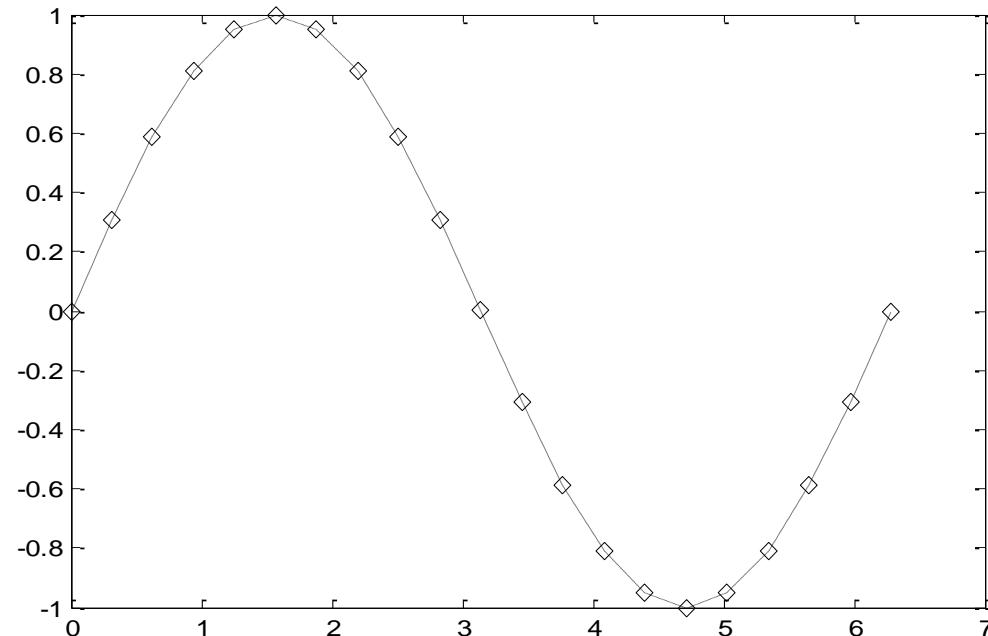
- 用黑色點線畫出正弦波
- 每一資料點畫上一個小菱形

```
x = 0 : pi /10 : 2*pi;  
y = sin(x);  
plot(x,y,'k:diamond')
```

%其中 “k”代表黑色，  
% “: ” 代表线型  
% “diamond”代表菱形的点

# plot基本绘图

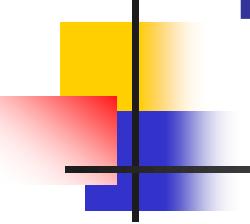
## —设置曲线的颜色 (Color)



# plot基本绘图

## —设置曲线的颜色 (Color)

| Plot函数的曲线颜色 | 曲线颜色         | RGB值    |
|-------------|--------------|---------|
| b           | 蓝色(Blue)     | (0,0,1) |
| c           | 青蓝色(Cyan)    | (0,1,1) |
| g           | 绿色(Green)    | (0,1,0) |
| k           | 黑色(Black)    | (0,0,0) |
| m           | 紫黑色(Magenta) | (1,0,1) |
| r           | 红色(Red)      | (1,0,0) |
| w           | 白色           | (1,1,1) |
| y           | 黄色(Yellow)   | (1,1,0) |



# plot基本绘图

## —加入说明文字

- 为增进图形的可读性，常常需要对图形或坐标轴加入说明。

| 指令     | 說明         |
|--------|------------|
| title  | 图形的标题      |
| xlabel | x 轴的說明     |
| ylabel | y 轴的說明     |
| legend | 标注图例       |
| text   | 图形中加入文字    |
| gtext  | 使用鼠标定位文字位置 |

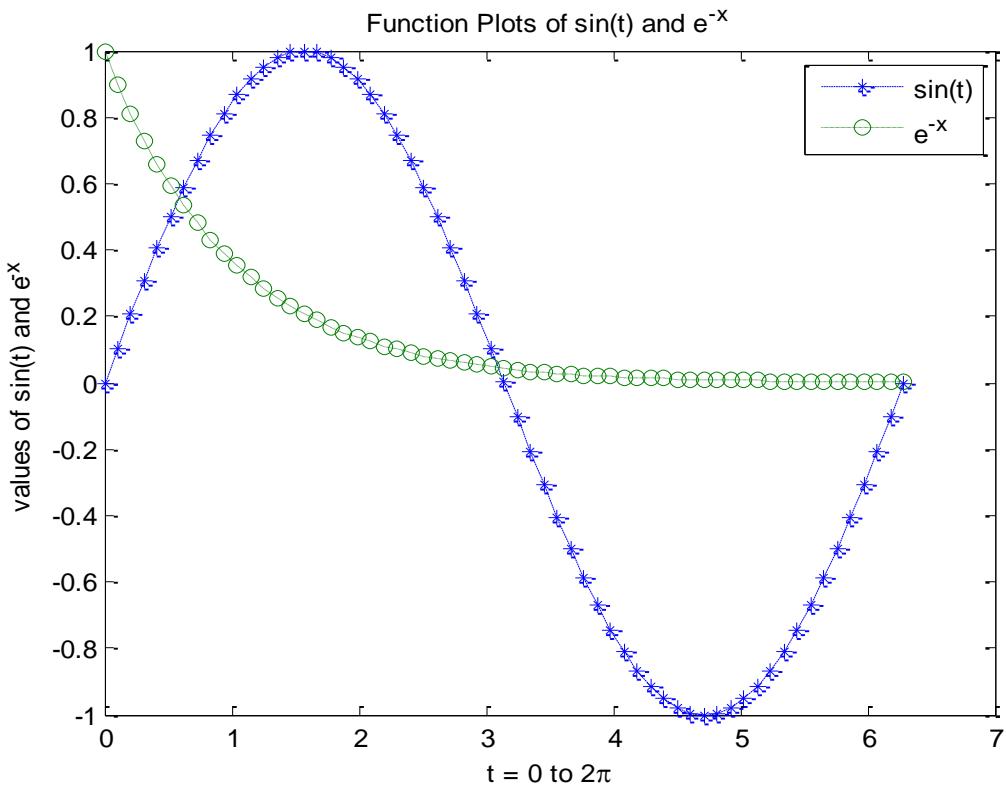
# plot基本绘图

## —加入说明文字

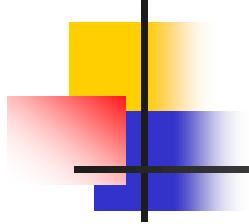
例

```
x = 0:pi/30:2*pi;  
  
y1 = sin(x);  
  
y2 = exp(-x);  
  
plot(x, y1, '--*', x, y2, ':o');  
  
xlabel('t = 0 to 2\pi');  
  
ylabel('values of sin(t) and e^{-x}');  
  
title('Function Plots of sin(t) and e^{-x}');  
  
legend('sin(t)', 'e^{-x}');
```

# plot基本绘图 —加入说明文字



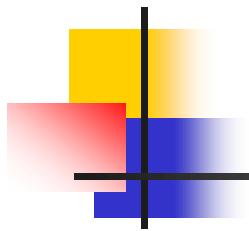
- **legend**指令：画出一小方块，给出每条曲线的说明
- “\”为特殊符号，用于产生上标、下标、希腊字母、数学符号等；遵循**LaTeX**数学模式
- 标题、坐标轴说明等的文字大小、位置也是可以修改的，见**Matlab Help**



## ■ 常用Latex用法

 $a^2 \quad a^{\wedge}\{2\}$  $a_2 \quad a\_ \{2\}$  $\infty \quad \backslash inf \ ty$  $\times \quad \backslash times$  $\oplus \times \quad \backslash oplus$  $\otimes \times \quad \backslash otimes$ 

|          |                     |
|----------|---------------------|
| $\alpha$ | <code>\alpha</code> |
| $\beta$  | <code>\beta</code>  |
| $\gamma$ | <code>\gamma</code> |
| $\pi$    | <code>\pi</code>    |
| $\tau$   | <code>\tall</code>  |
| $\Delta$ | <code>\Delta</code> |
| $\delta$ | <code>\delta</code> |
| $\Omega$ | <code>\Omega</code> |



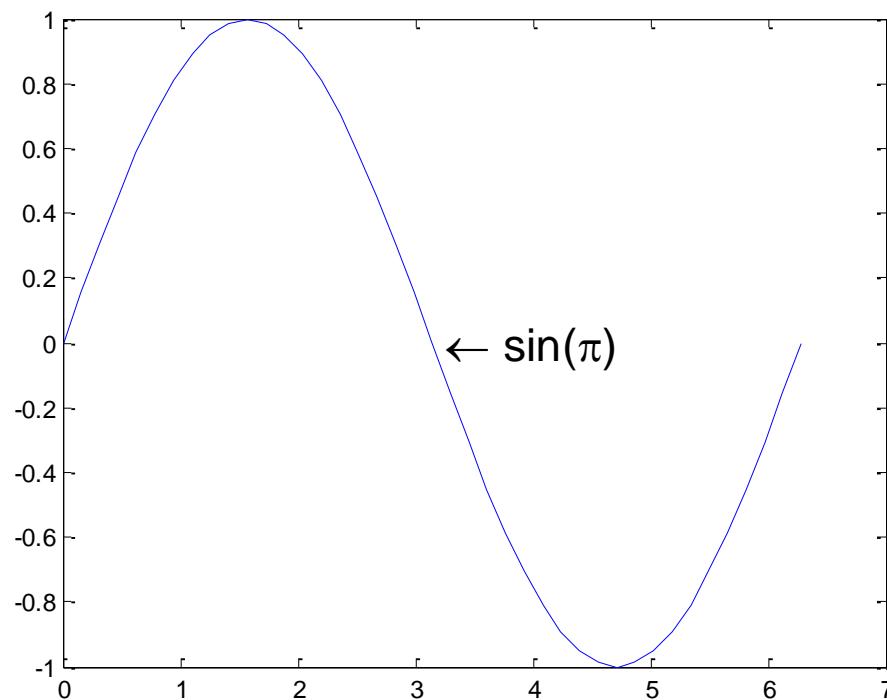
# plot基本绘图

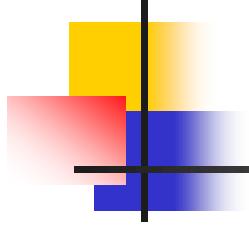
## —加入说明文字

- `text(x, y, 'string')`
  - x、y : 文字起始坐标位置
  - string : 文字内容
- 例

```
x = 0:pi/20:2*pi;
plot(x,sin(x))
text(pi,0,' \leftarrow
sin(\pi)', 'FontSize',18)
```

# plot基本绘图 —加入说明文字





# plot基本绘图

## —加入说明文字

- 有时不确定说明文字位置，可用鼠标确定文字位置。
- `gtext('string')`

用鼠标单击图形，在选中点处放入说明文字**string**

# plot基本绘图

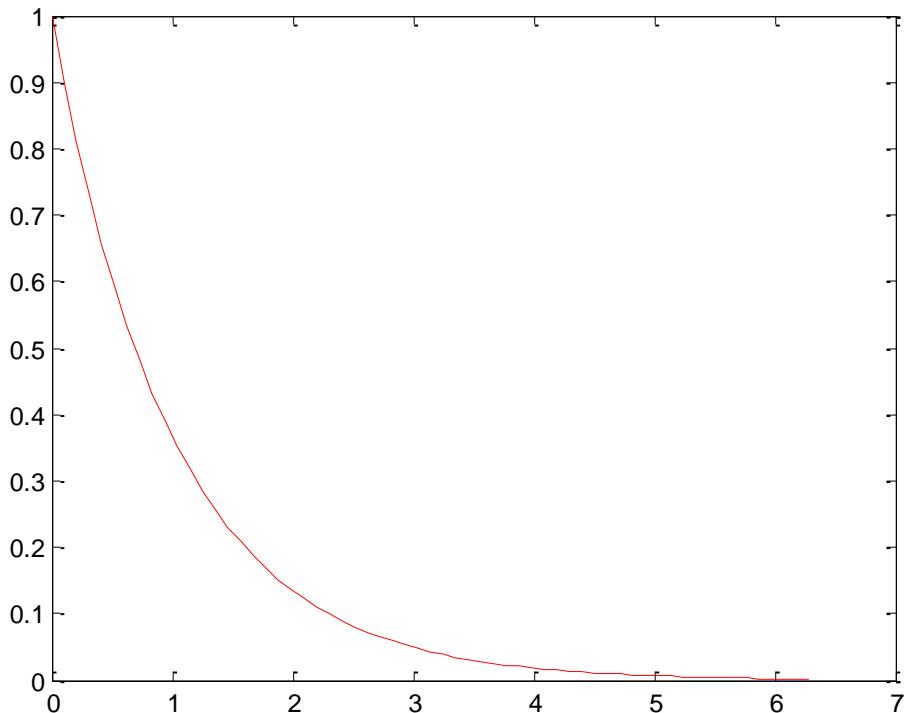
## —hold函数

- 有时希望在已经画了一条曲线的图上再添加一条曲线。

```
x = 0:pi/20:2*pi;  
plot(x,sin(x),'g');  
%再添加一条曲线  
plot(x,cos(x),'r');
```

- 会得到想要的结果吗？

# plot基本绘图 —hold函数

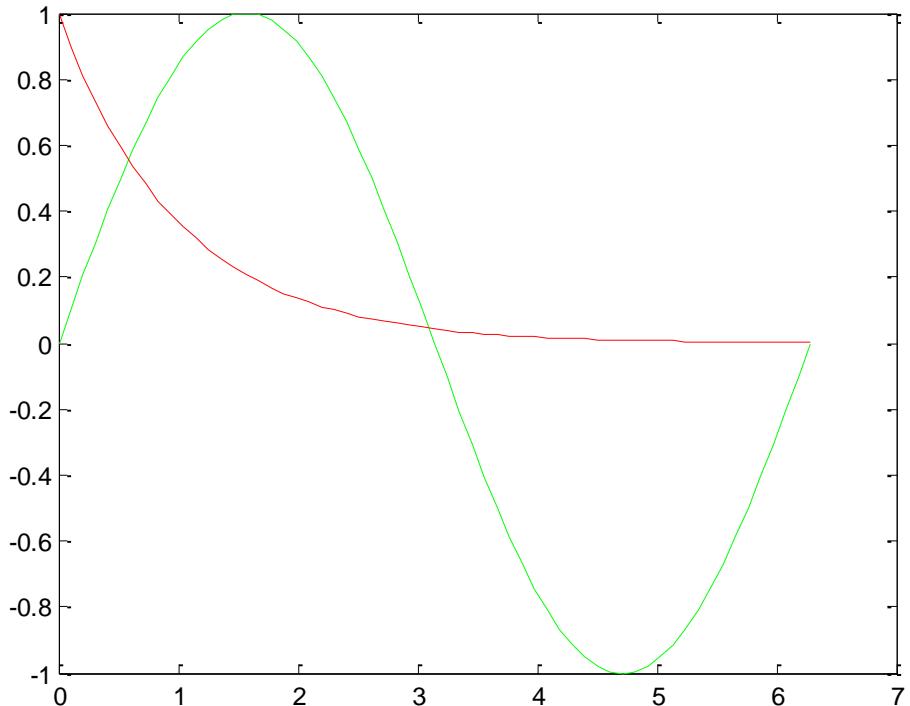


只画了后面一条  
曲线，第一条被  
擦除了。

# plot基本绘图

## —hold函数

```
x = 0:pi/20:2*pi;  
plot(x,sin(x),'g');  
  
hold on  
  
plot(x,cos(x),'r');  
  
hold off
```



# plot基本绘图

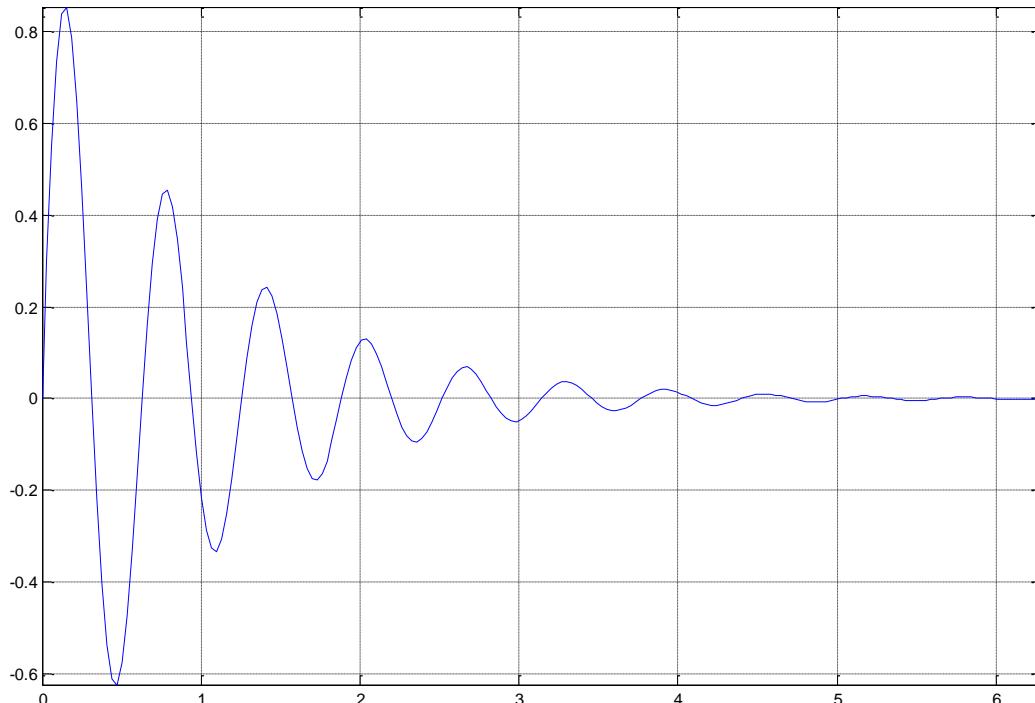
## —给图形加上网格

- 有时希望在图形上加上网格，以便更好地观察波形的变化。
- **grid**函数

```
x = 0:pi/100:2*pi;  
y = exp(-x).*sin(10*x);  
plot(x,y)  
grid on %给图形加上网格
```

# plot基本绘图

## —给图形加上网格



- 去掉网格：  
`grid off`
- 只用`grid` 则在  
`on`、`off`状态间切  
换
- `grid minor`

# plot基本绘图

## —双坐标轴

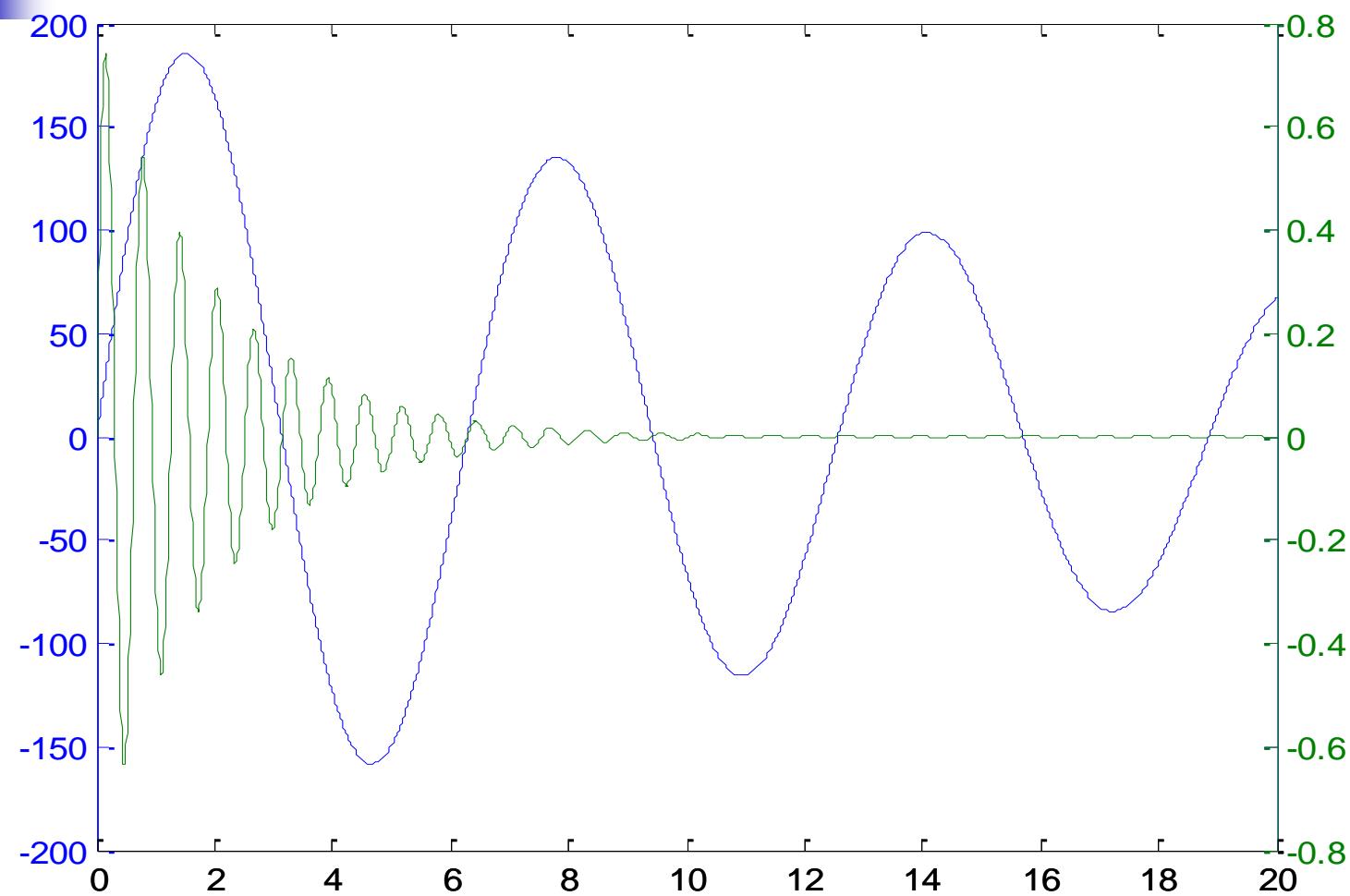
- **plotyy**函数

在同一窗口画幅度相差很大的两条曲线，采用不同的y轴刻度。

```
x = 0:0.01:20;  
y1 = 200*exp(-0.05*x).*sin(x);  
y2 = 0.8*exp(-0.5*x).*sin(10*x);  
plotyy(x,y1,x,y2,'plot');
```

# plot基本绘图

## —双坐标轴



# plot基本绘图

## —坐标轴控制

### ■ 坐标轴刻度与长宽比控制

- 默认坐标轴长宽比是窗口的长宽比
- 可用**axis**指令加以修改

**axis ([xmin xmax ymin ymax])**

**axis normal**      默认的长宽比

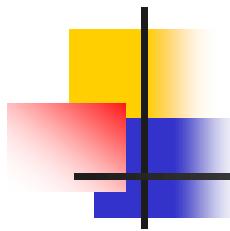
**axis square**      长宽比为1

**axis equal**      长宽比不变，但两轴刻度比例一致

**axis tight**      图轴紧贴图形

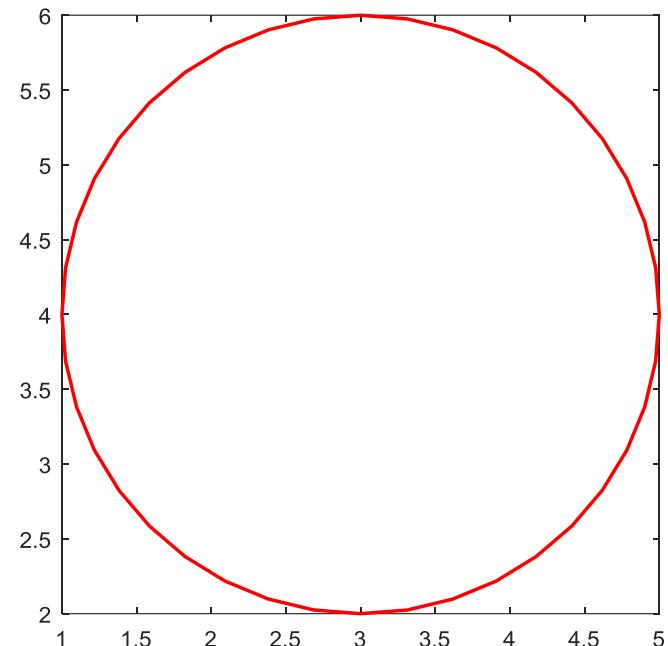
**axis off**

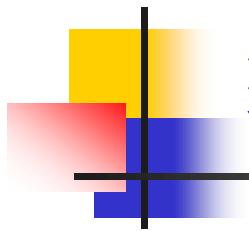
**axis on**



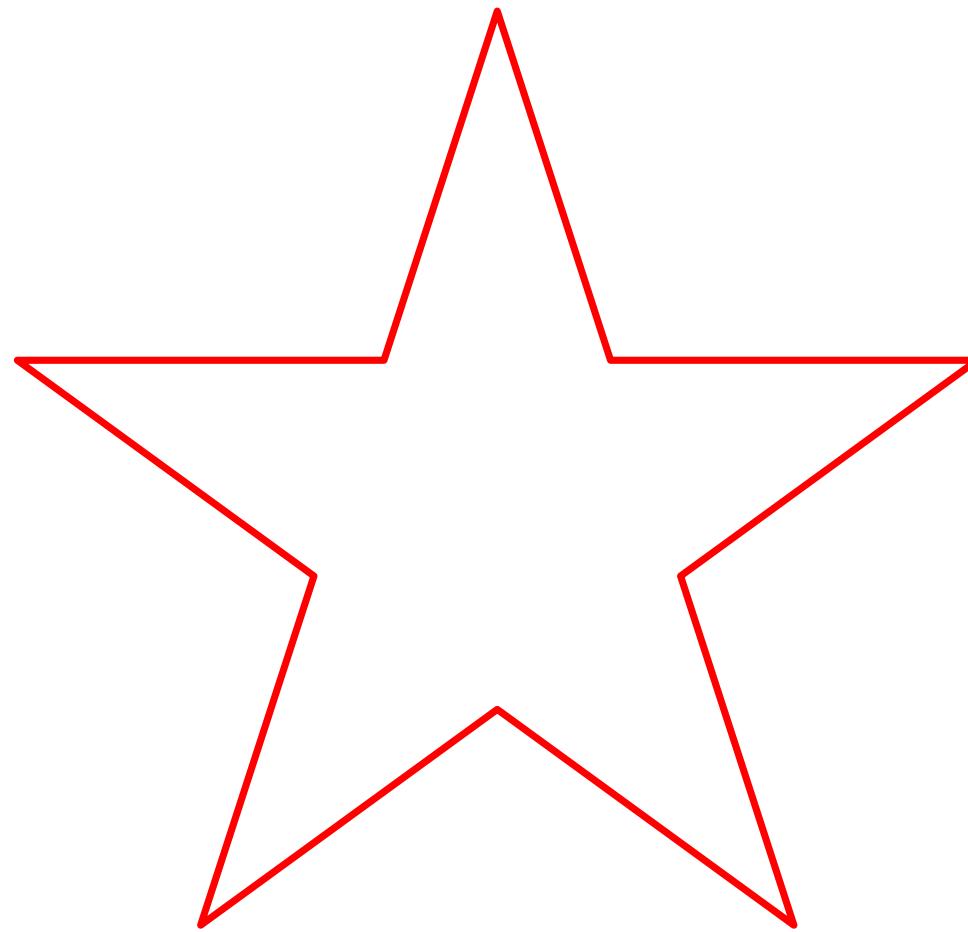
# 画一个圆

```
r = 2;  
theta = 0:pi/20:2*pi;  
x = r*cos(theta) +3;  
y = r*sin(theta) +4;  
plot(x,y,'r');  
axis equal
```





# 练习：画一个五角星



# plot基本绘图

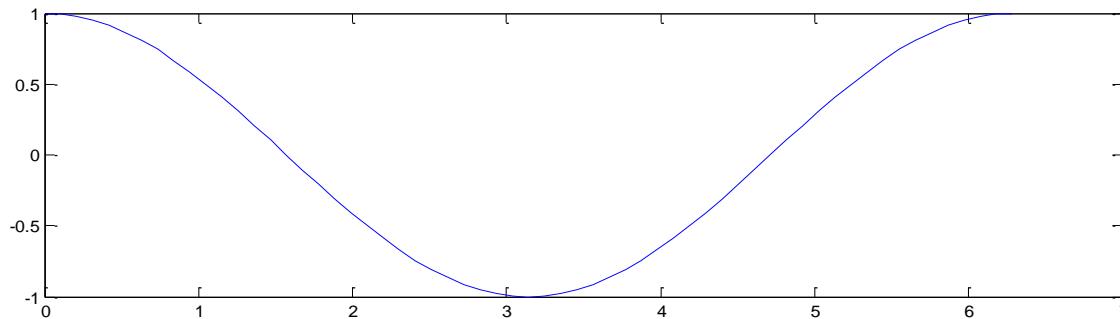
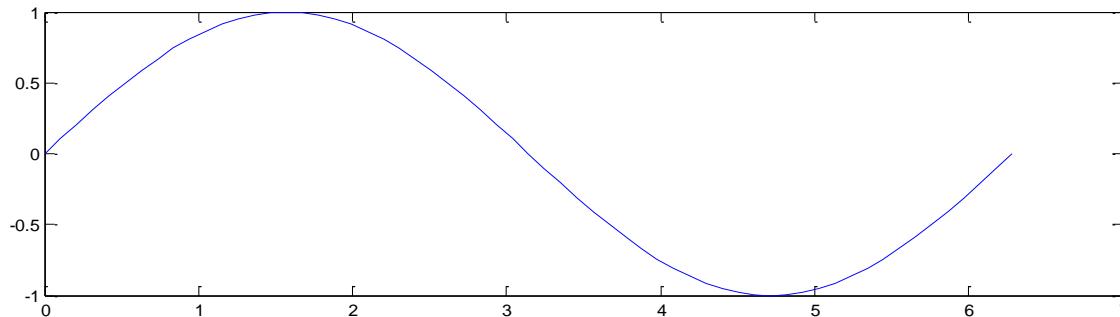
## —在一个窗口中画多个图形

- 若要画多条曲线，我们可以：
  - 在同一窗口中画多条曲线
  - 打开多个窗口，在每个窗口中画一条曲线
  - 还可以利用**subplot**函数，在同一窗口中开设多个子窗口，每个子窗口画一条曲线。

```
x=0:pi/30:2*pi;  
subplot(2,1,1); plot(x,sin(x))  
subplot(2,1,2); plot(x,cos(x))
```

# plot基本绘图

——在一个窗口中画多个图形



# plot基本绘图

## —在一个窗口中画多个图形

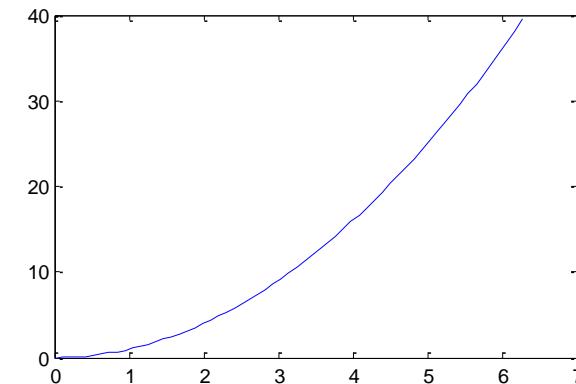
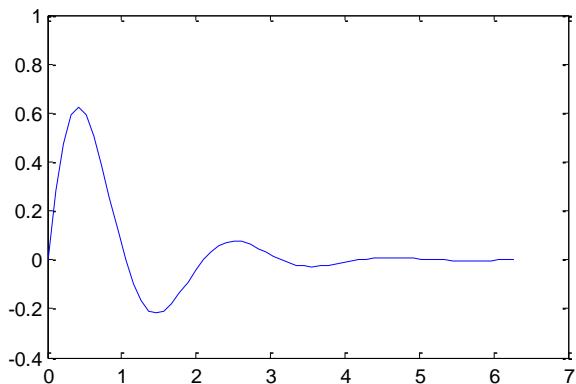
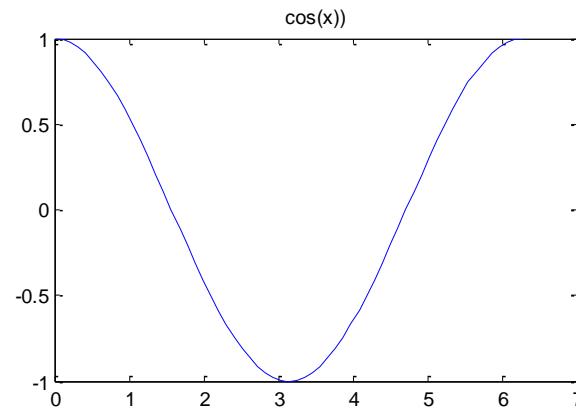
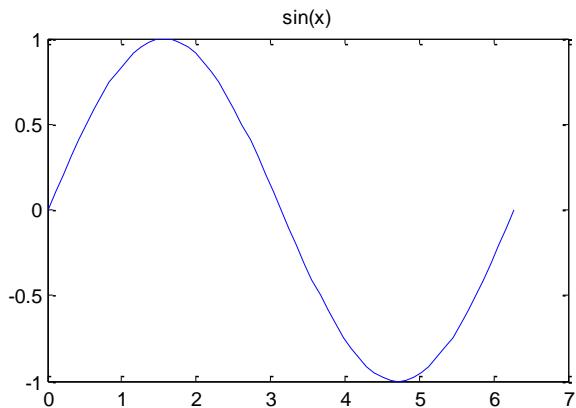
### ■ subplot(m,n,p)

- 将窗口分成  $m \times n$  个子窗口
- 下一个 plot 指令绘图于第  $p$  个子窗口
- $p$  的算法为由左至右，一列一列
- 也可写成： subplot(mnp)

```
x = 0:pi/30:2*pi
subplot(2,2,1) ;
plot(x,sin(x)) ; title('sin(x)')      %左上角
subplot(2,2,2) ;
plot(x,cos(x)) ; title('cos(x)')      %右上角
subplot(2,2,3) ; plot(x,exp(-x).*sin(3*x));
subplot(2,2,4) ; plot(x, x.^2);
```

# plot基本绘图

## —在一个窗口中画多个图形

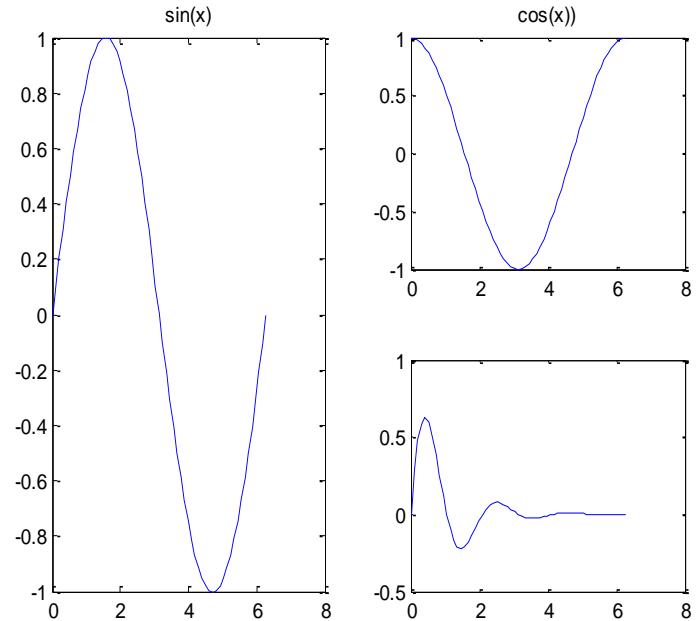


# plot基本绘图

## —在一个窗口中画多个图形

- 还可以组合子窗口

```
x = 0:pi/30:2*pi;
subplot(2,2,[1,3]) ;
plot(x,sin(x)) ; title('sin(x)')
subplot(2,2,2) ;
plot(x,cos(x)) ; title('cos(x)')
subplot(2,2,4);
plot(x,exp(-x).*sin(3*x));
```

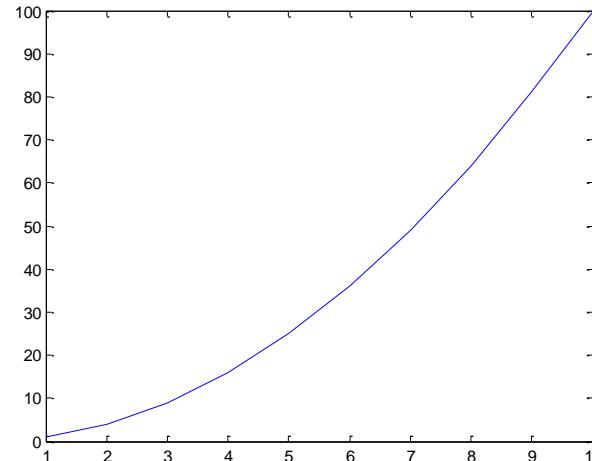


# plot基本绘图

## —当数组值为复数

- z是一个复数向量
- plot(z)将z 的实部和虚部当成x坐标和y坐标来画图,即  
**plot(real(z), imag(z))**

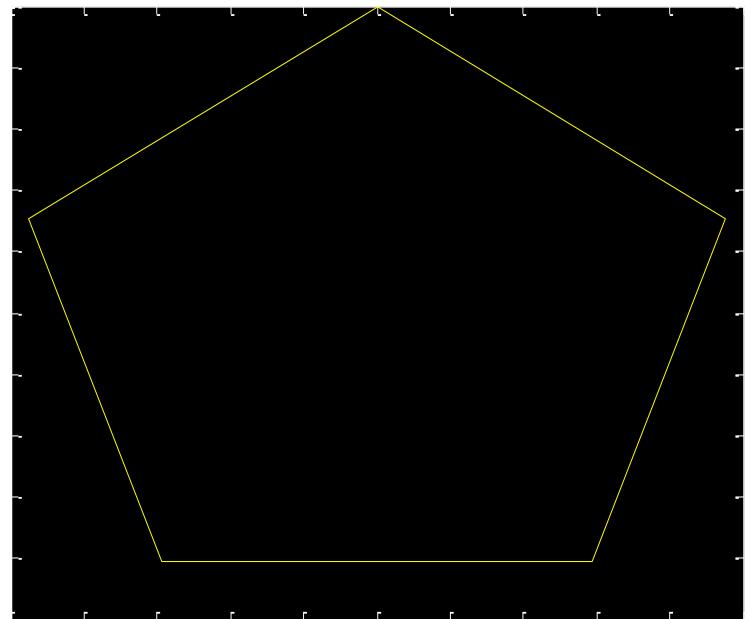
```
>> x = 1:10;  
>> y = x.^2;  
>> z =  
  
complex(x,y)  
>> plot(z)
```

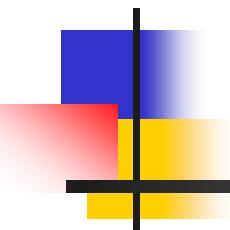


# plot基本绘图

## —当数组值为复数

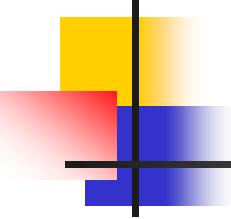
```
theta = pi/2 :  
2*pi/5 :(2*pi+pi/2);  
  
z = exp(j*theta);  
  
colordef black  
  
plot(z)  
  
axis equal tight
```





## Part2:

# 三维绘图



# 三维作图

- 三维曲线
- 空间曲面
- 等高线
- $v=f(x,y,z)$ 的可视化

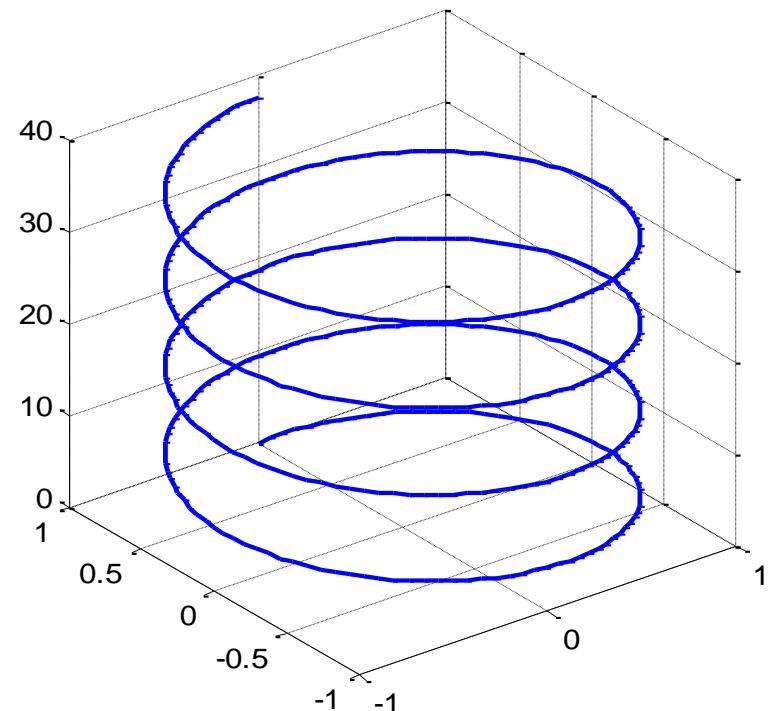
# 三维作图

- 绘制三维曲线： $x=x(t)$ ， $y=y(t)$ ， $z=z(t)$

```
plot3(x,y,z,)
```

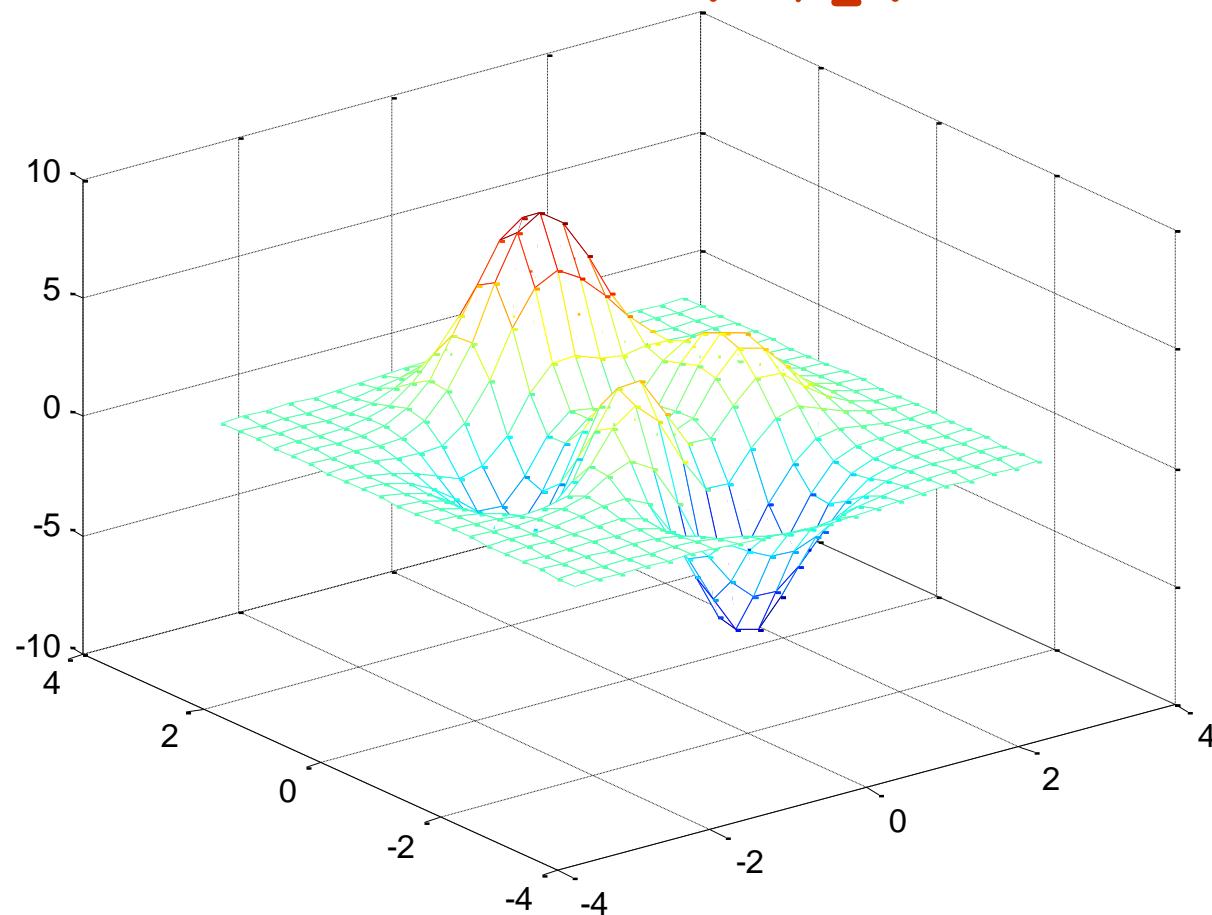
例：绘制三维螺旋线：

```
t = 0:pi/20:8*pi;  
x = sin(t);  
y = cos(t);  
z = 1.5*t;  
hp = plot3(x,y,z);  
set(hp,'linewidth',2,'color','b');  
grid on  
axis square
```



# 三维作图

■ 空间曲面:  $z = f(x, y)$



# 三维作图

## ■ **mesh (X, Y, Z)**

■ 网格生成函数: **meshgrid**

x, y 为给定的向量, X, Y 是网格划分后得到的网格矩阵

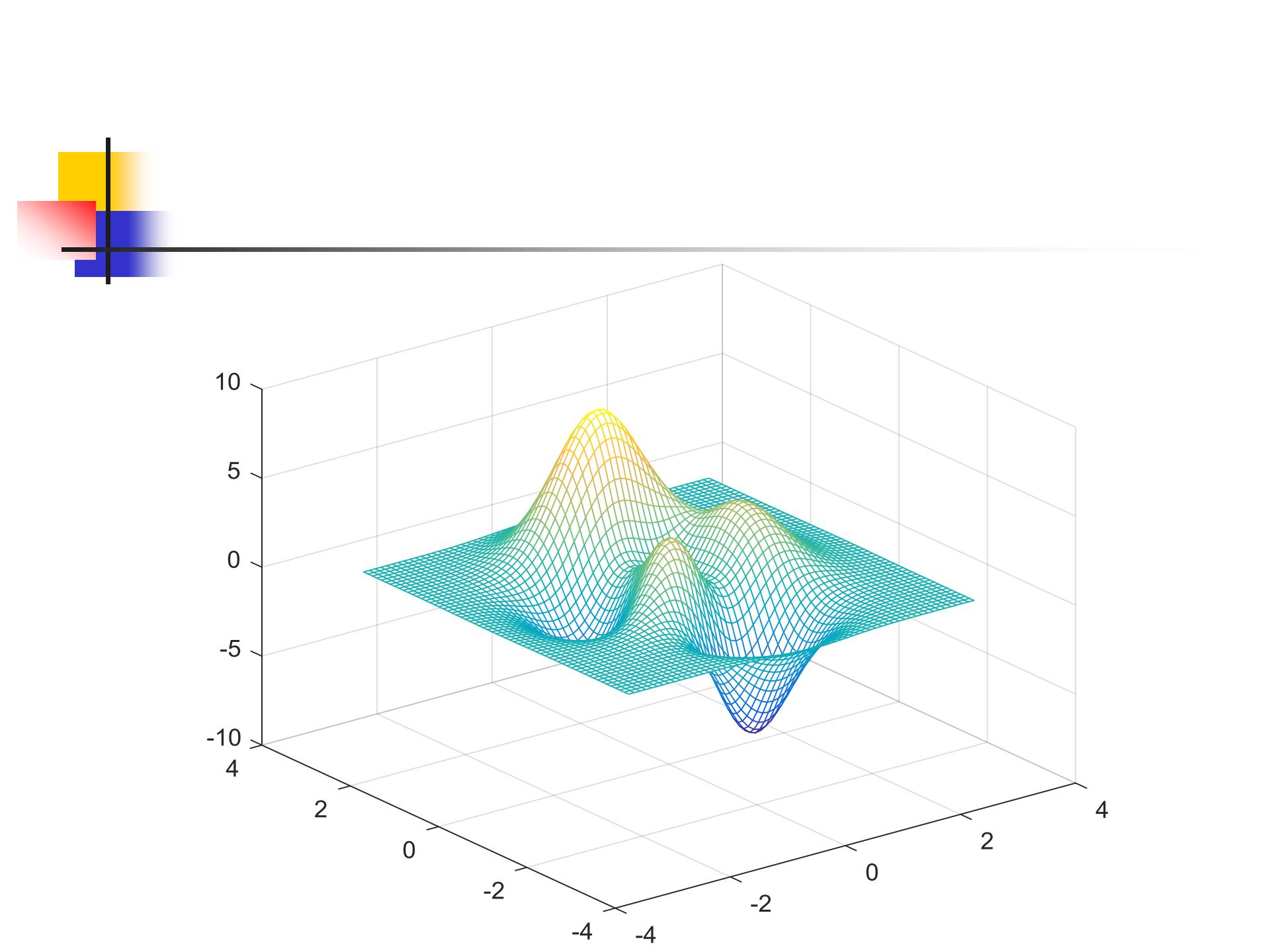
若 x = y, 则可简写为 [X, Y] = **meshgrid (x)**

例:

```
[X, Y] = meshgrid(-3 : 0.1 : 3);  
Z = peaks(X, Y);  
mesh(X, Y, Z)
```

>> peaks %matlab自带的测试函数

```
z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...  
- 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...  
- 1/3*exp(-(x+1).^2 - y.^2)
```



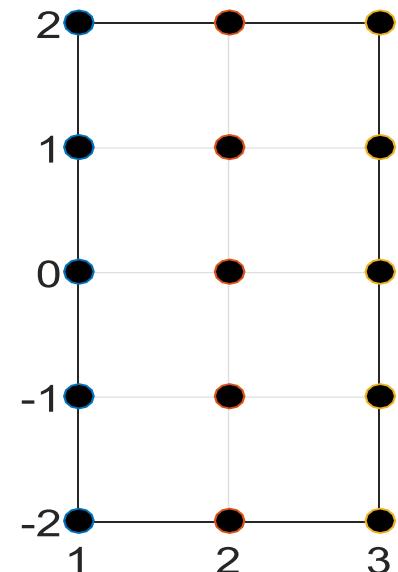
# meshgrid 函数

```
gx = [1 2 3];  
gy = [-2:2];  
[x,y ] =  
meshgrid(gx,gy)  
whos x y
```

|   |     |     |        |
|---|-----|-----|--------|
| x | 5x3 | 120 | double |
| y | 5x3 | 120 | double |

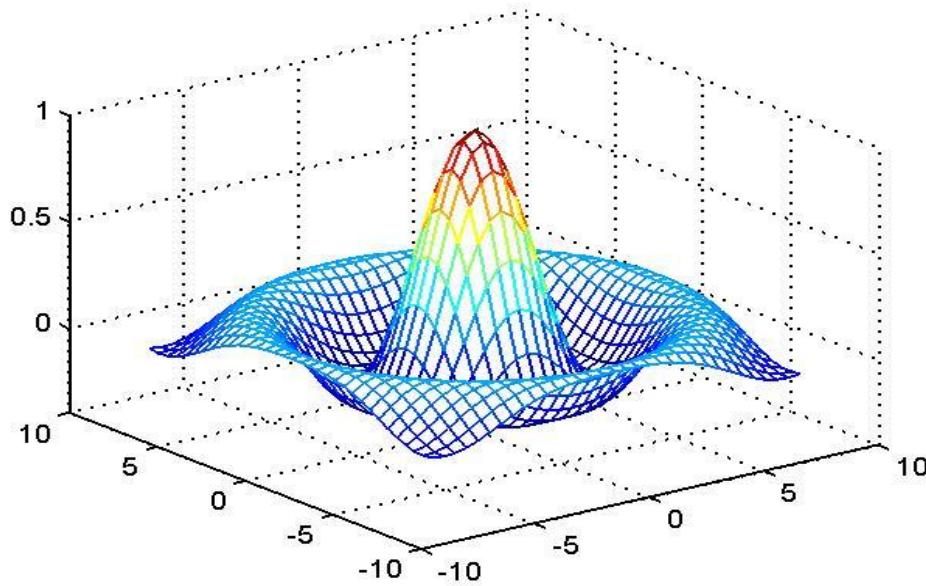
|     |   |   |   |
|-----|---|---|---|
| x = | 1 | 2 | 3 |
| 1   | 1 | 2 | 3 |
| 1   | 1 | 2 | 3 |
| 1   | 1 | 2 | 3 |
| 1   | 2 | 3 |   |

|     |    |    |    |
|-----|----|----|----|
| y = | -2 | -2 | -2 |
| -1  | -1 | -1 |    |
| 0   | 0  | 0  |    |
| 1   | 1  | 1  |    |
| 2   | 2  | 2  |    |

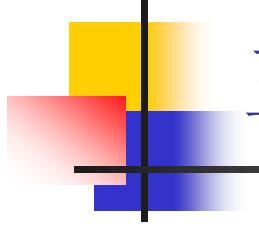


# 三维作图

## ■ 练习：“墨西哥帽子”

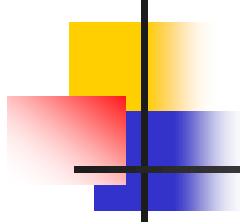


由函数  $z = \sin(r)/r$ , 其中  $r = \sqrt{x^2 + y^2}$  确定的曲面



# 三维作图

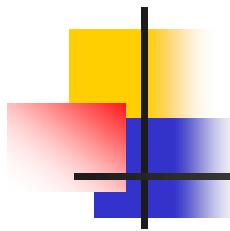
- 练习：画一个球体



## 三维作图

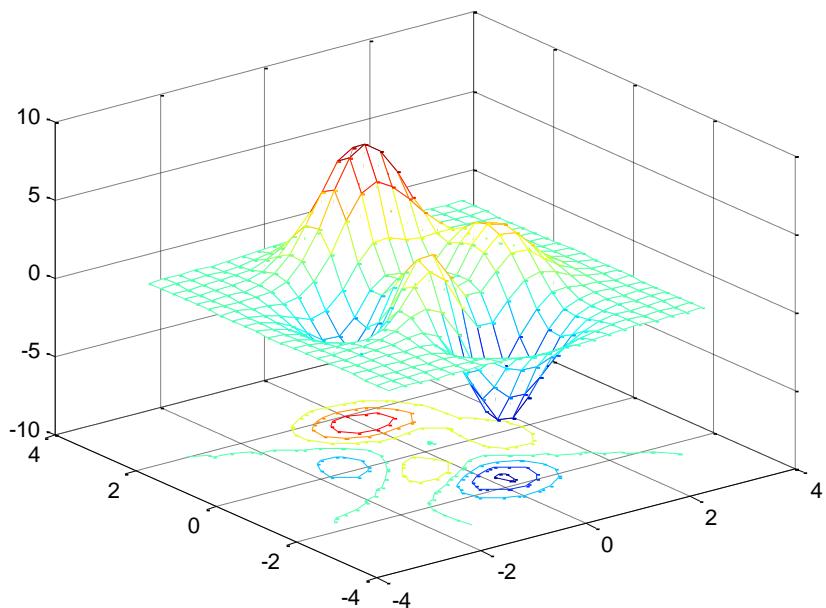
- **meshc**: 绘制有等高线的空间曲面
- **meshz**: 绘制含0平面的空间曲面

```
[X,Y] = meshgrid(-3:.4:3);  
Z = peaks(X,Y);  
meshc(X,Y,Z), title('meshc')  
figure  
meshz(X,Y,Z), title('meshz')
```

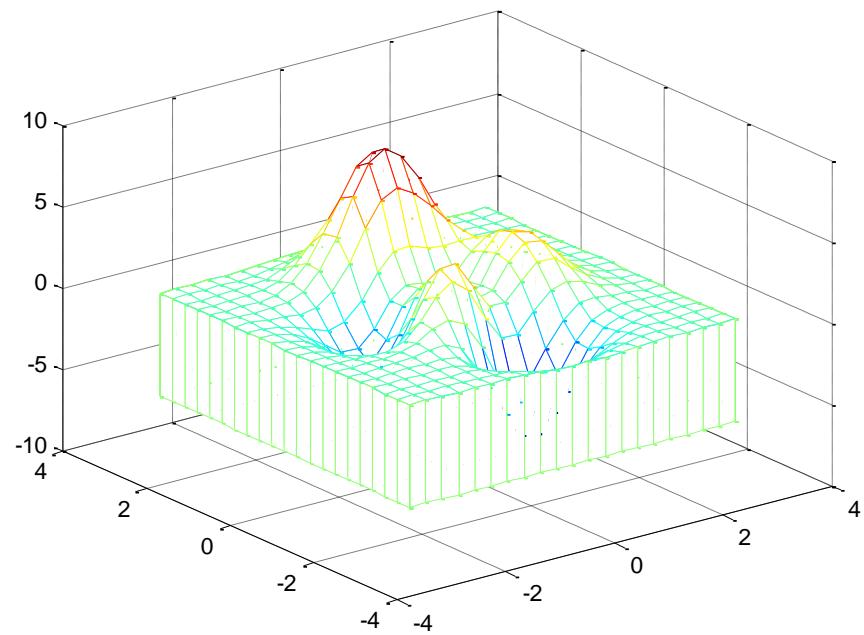


# 三维作图

meshc

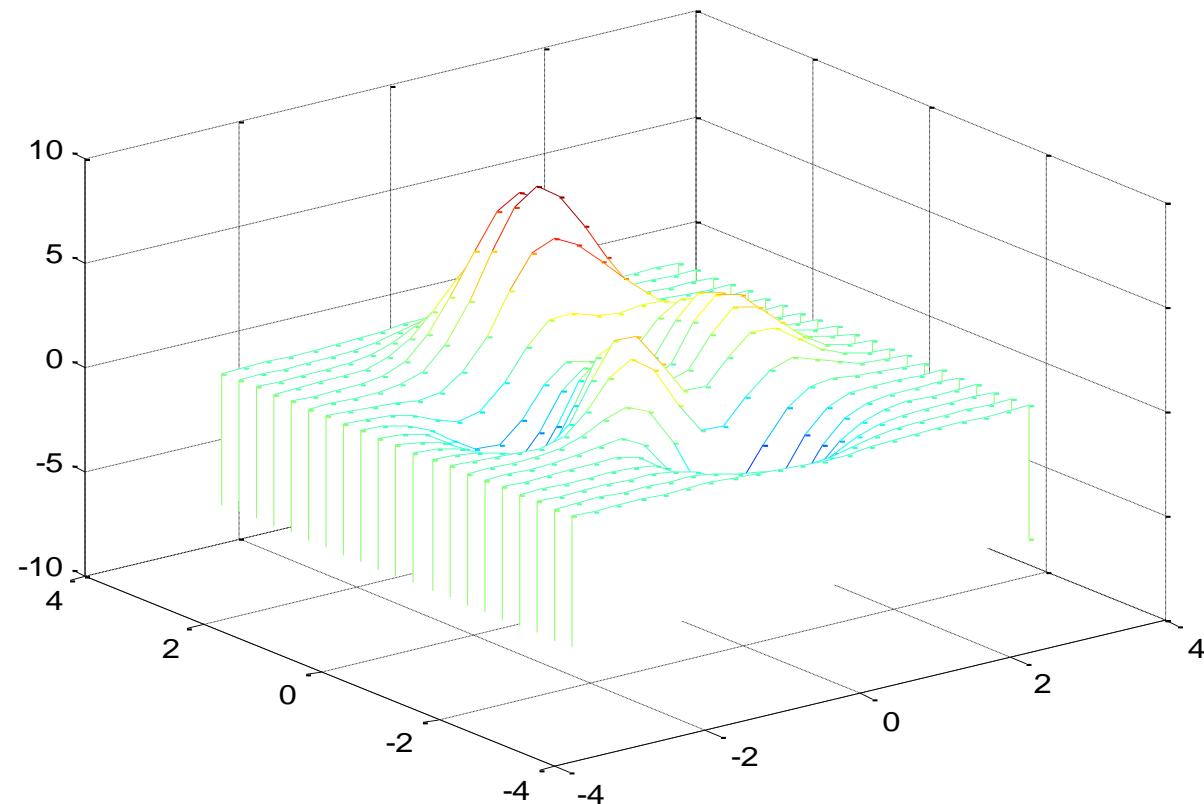


meshz



# 三维作图

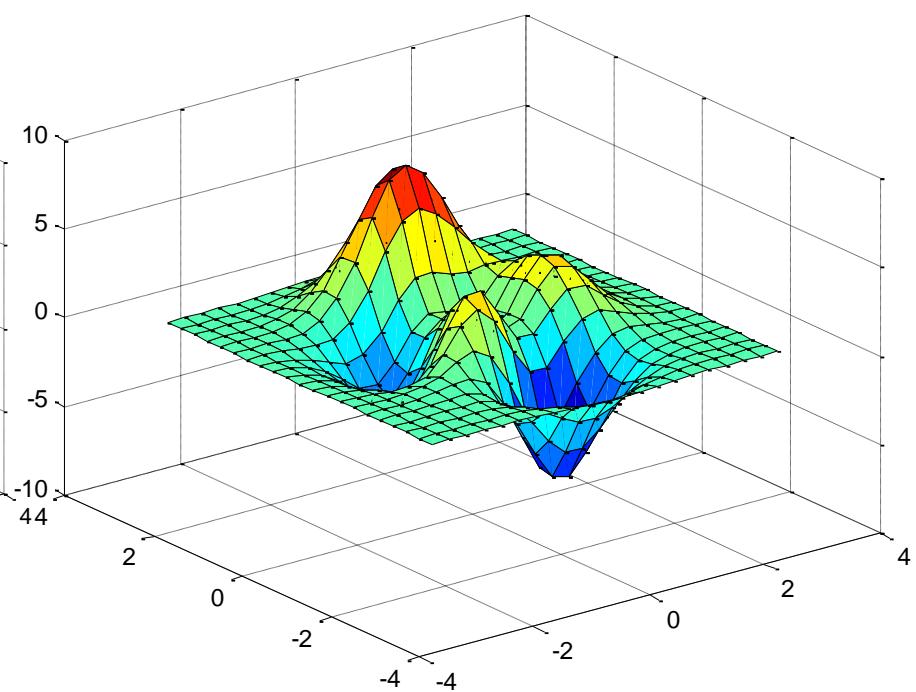
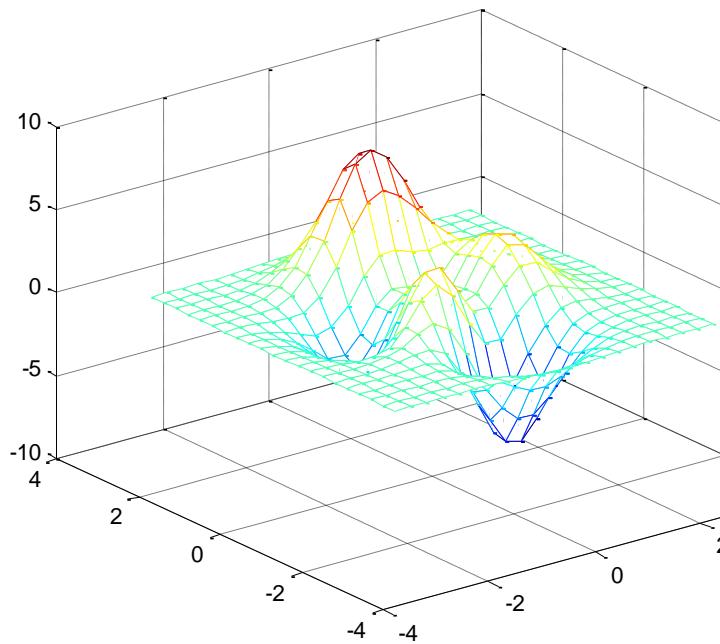
**waterfall** : 只绘制x轴方向的网格线



# 三维作图

- **surf(X, Y, Z)** 绘制由矩阵 X, Y, Z 所确定的曲面图，参数含义同 mesh。

**mesh** 绘制网格图， **surf** 绘制着色的三维表面图



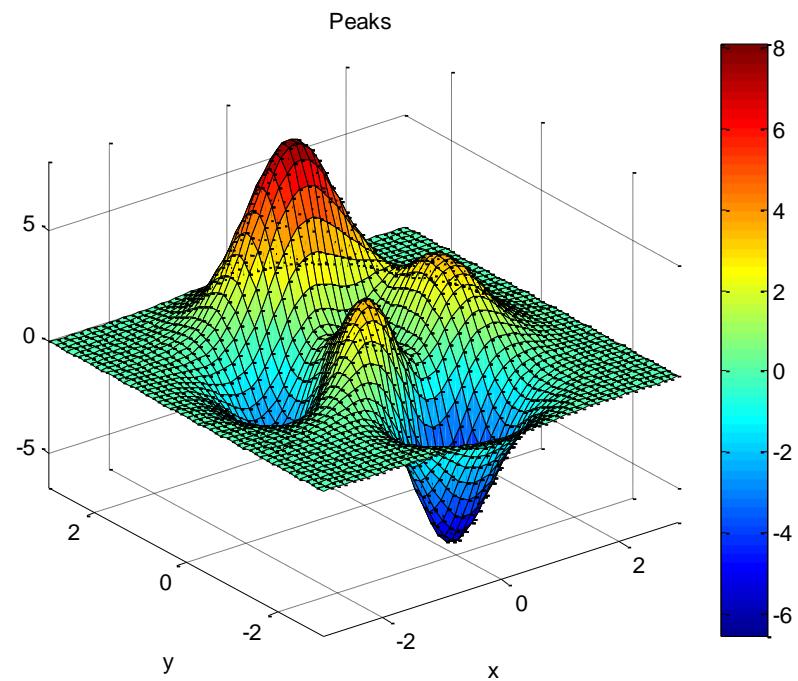
# 三维作图

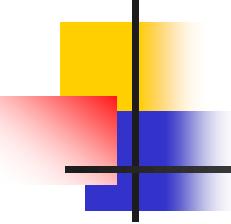
- 曲面颜色的控制:
- **colorbar**: 显示 MATLAB 如何以不同颜色代表曲面的高度。

例:

```
peaks;
```

```
colorbar;
```

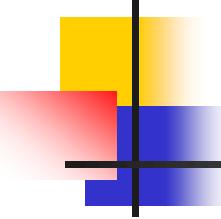




# 三维作图

- 曲面颜色的控制：

| 顏色              | Red (紅色) | Green (綠色) | Blue (藍色) |
|-----------------|----------|------------|-----------|
| black (黑)       | 0        | 0          | 0         |
| white (白)       | 1        | 1          | 1         |
| red (紅)         | 1        | 0          | 0         |
| green (綠)       | 0        | 1          | 0         |
| blue (藍)        | 0        | 0          | 1         |
| yellow (黃)      | 1        | 1          | 0         |
| magenta (錳紫)    | 1        | 0          | 1         |
| cyan (青藍)       | 0        | 1          | 1         |
| gray (灰)        | 0.5      | 0.5        | 0.5       |
| dark red (暗紅)   | 0.5      | 0          | 0         |
| copper (銅色)     | 1        | 0.62       | 0.4       |
| aquamarine (碧綠) | 0.49     | 1          | 0.83      |



# 三维作图

- 颜色映射图 **colormap**

```
cm = colormap;  
size(cm)  
ans =  
    64      3
```

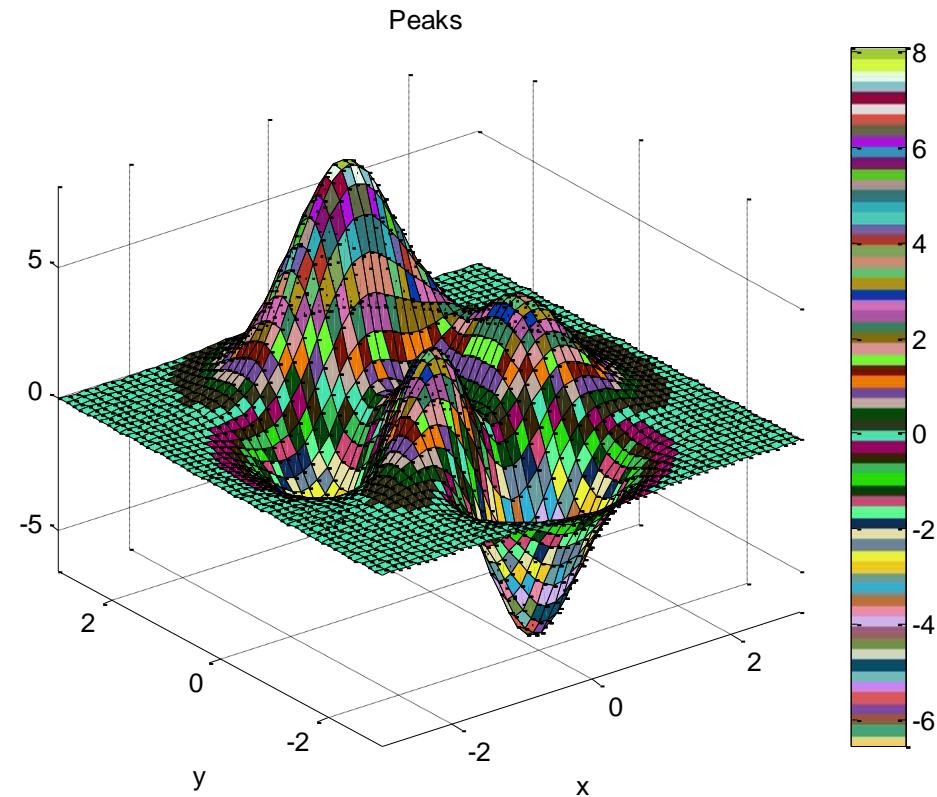
- cm是64\*3的矩阵，每行代表RGB的成分，即一种颜色。
- matlab在画图时，把第一行的颜色对应曲面的最高点，最后一行颜色对应曲面最低点。其余高度颜色依照线性内插法给出。

# 三维作图

- 改变颜色映射图，可得到不同颜色的曲面。

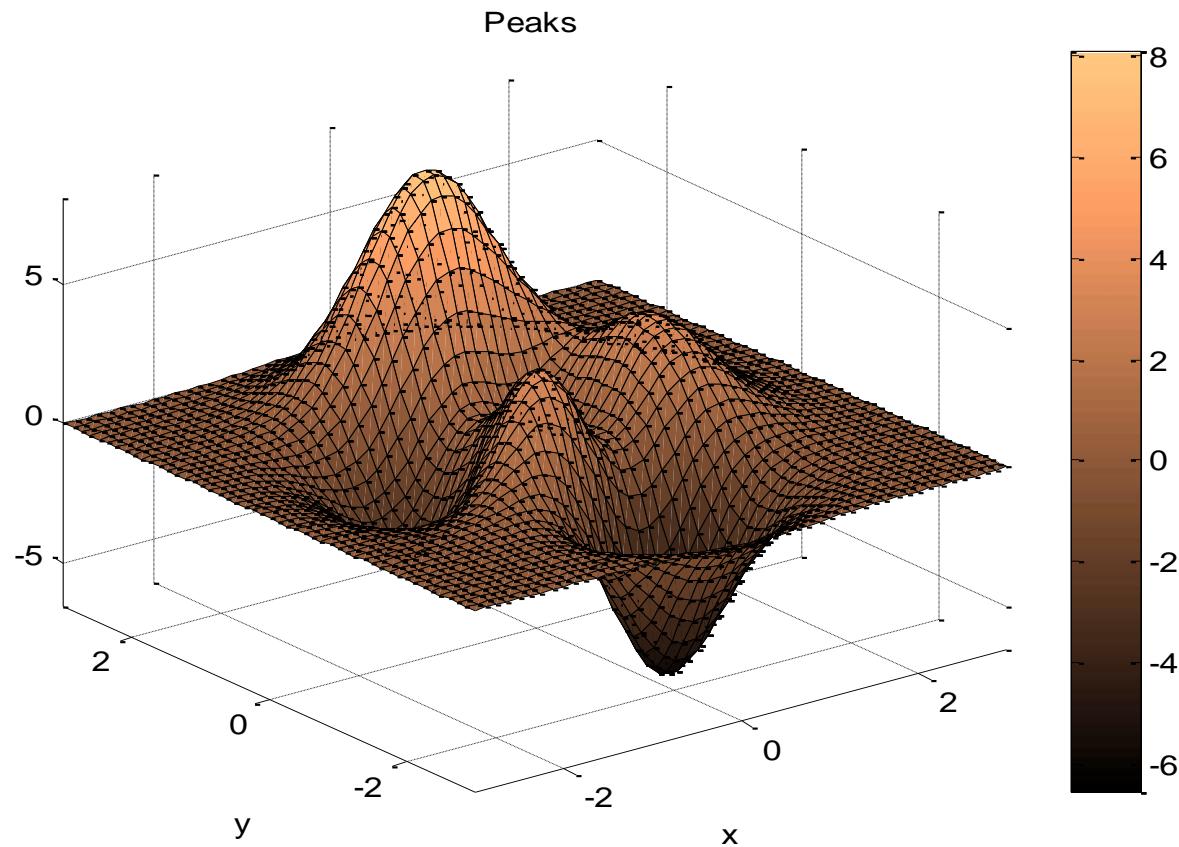
例：

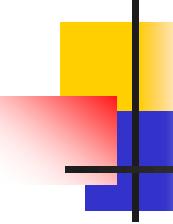
```
peaks;  
colormap(rand(64,3));  
colorbar;
```



# 三维作图

■ `colormap ('copper')`

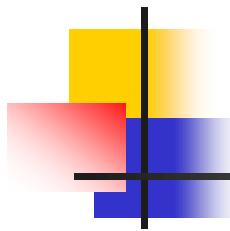




# 三维作图

## ■ matlab预设的颜色映射图

| 指令              | 說明              |
|-----------------|-----------------|
| colormap hsv    | HSV 的顏色對應表（預設值） |
| colormap hot    | 代表“熱”的顏色對應表     |
| colormap cool   | 代表“冷”的顏色對應表     |
| colormap summer | 代表“夏天”的顏色對應表    |
| colormap gray   | 代表“灰階”的顏色對應表    |
| colormap copper | 代表“銅色”的顏色對應表    |
| colormap autumn | 代表“秋天”的顏色對應表    |
| colormap winter | 代表“冬天”的顏色對應表    |
| colormap spring | 代表“春天”的顏色對應表    |
| colormap bone   | 代表“X光片”的顏色對應表   |
| colormap pink   | 代表“粉紅”的顏色對應表    |
| colormap flag   | 代表“旗幟”的顏色對應表    |



# 三维作图

- 以曲面梯度大小设定颜色:

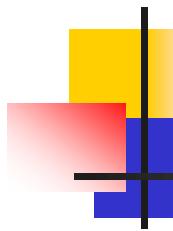
```
[x,y,z] = peaks;  
surf(x,y,z,gradient(z));
```

- 以曲面曲率大小设定颜色:

```
[x,y,z] = peaks;  
surf(x,y,z,del2(z));
```

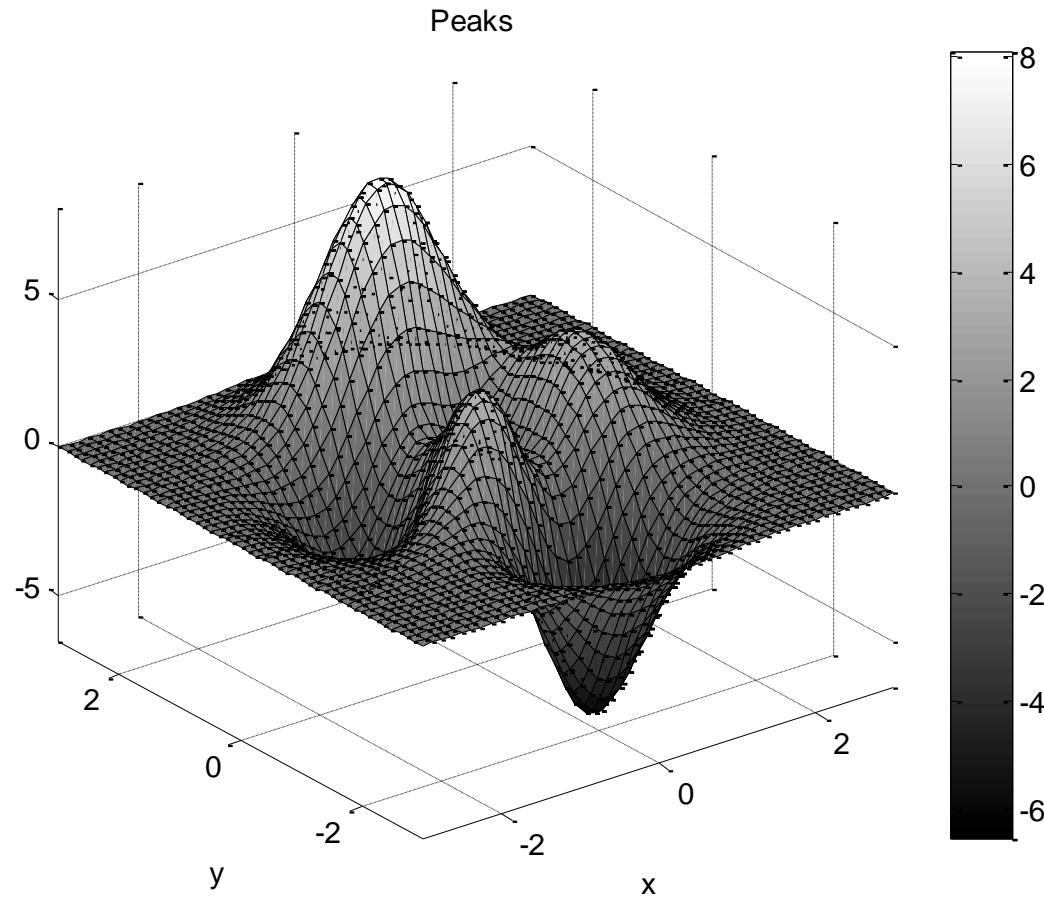
- 使得表面颜色连续变化

```
surf(peaks)  
axis tight  
shading interp
```



# 三维作图

■ colormap gray



# 三维作图

- 改变图形的观察视角：
- 同一物体，从不同角度看，图形也不同，比如一个椭球体。

```
view(az,el);
```

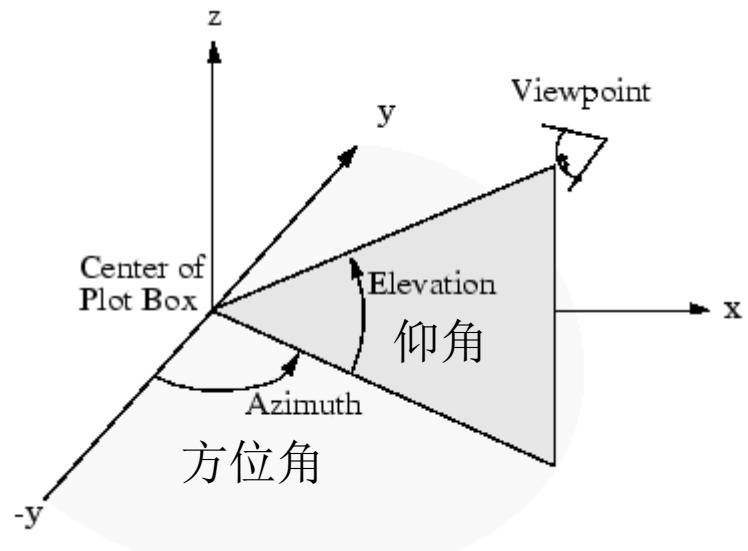
az:方位角

el: 仰角

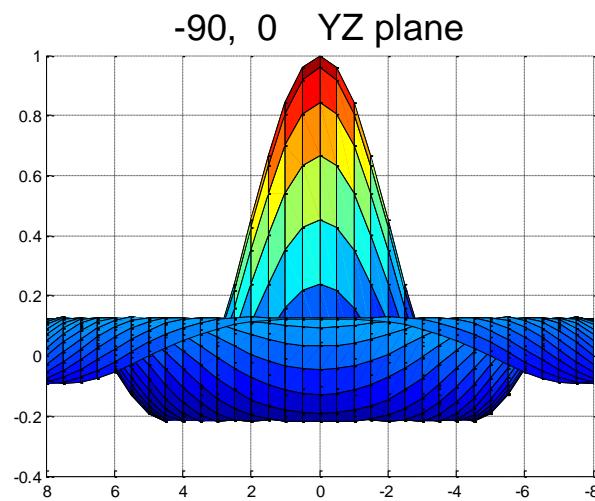
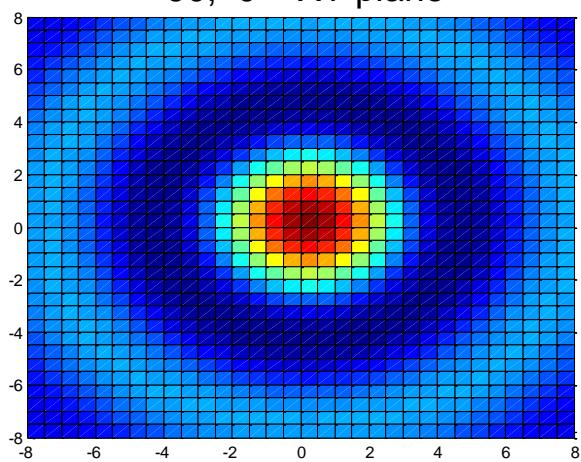
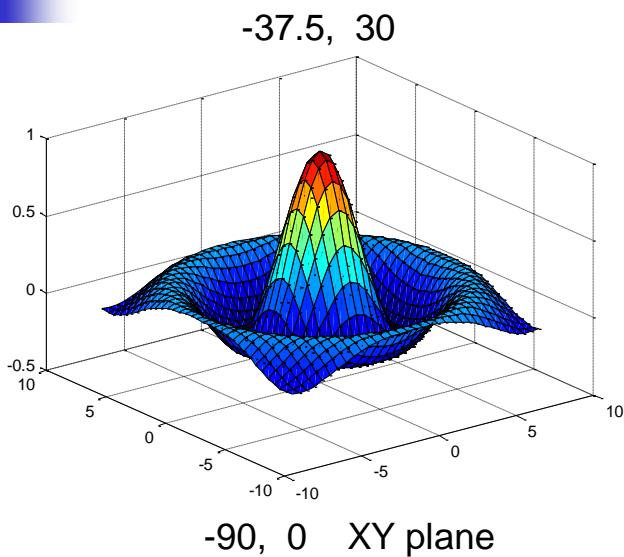
```
view(x,y,z)
```

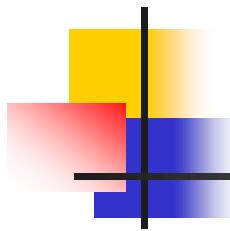
```
view(2); az=0, el=90
```

```
view(3); az=-37.5,el=30
```



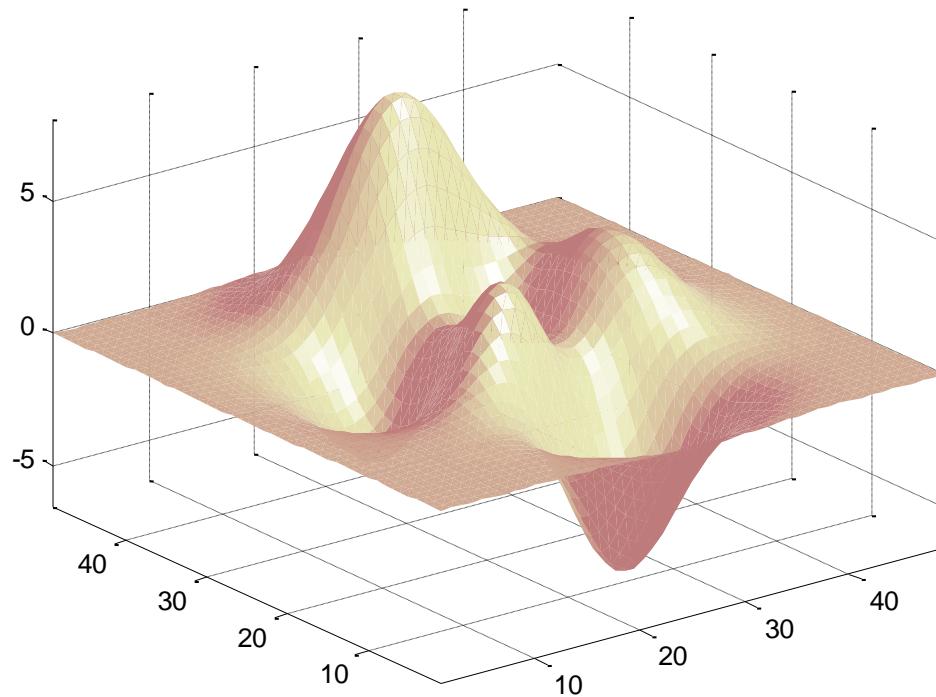
# 三维作图

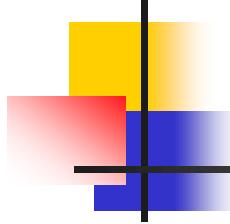




# 三维作图

- 试试这个效果：





## 三维作图

- 绘制等高线图：

**contour**: 平面上的等高线

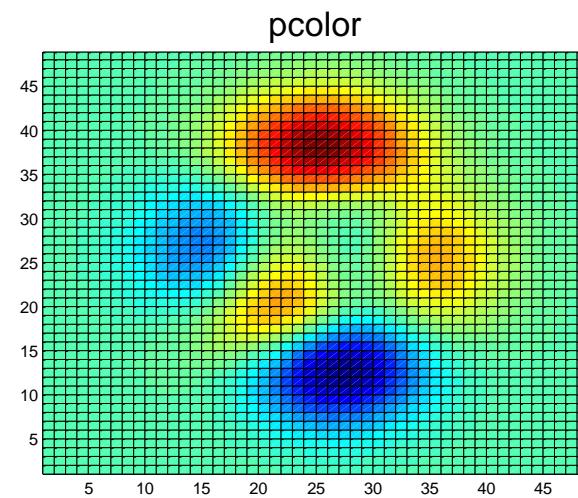
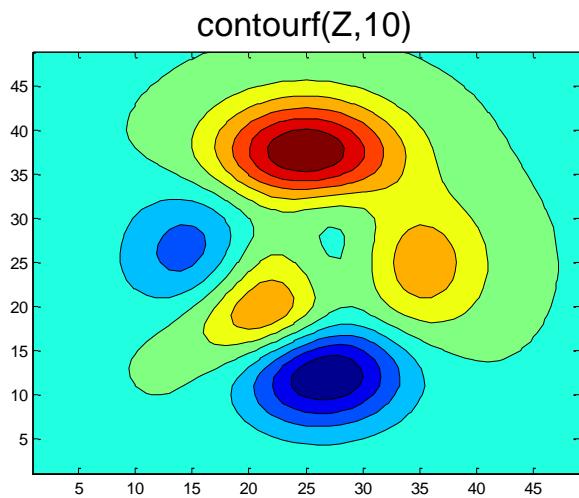
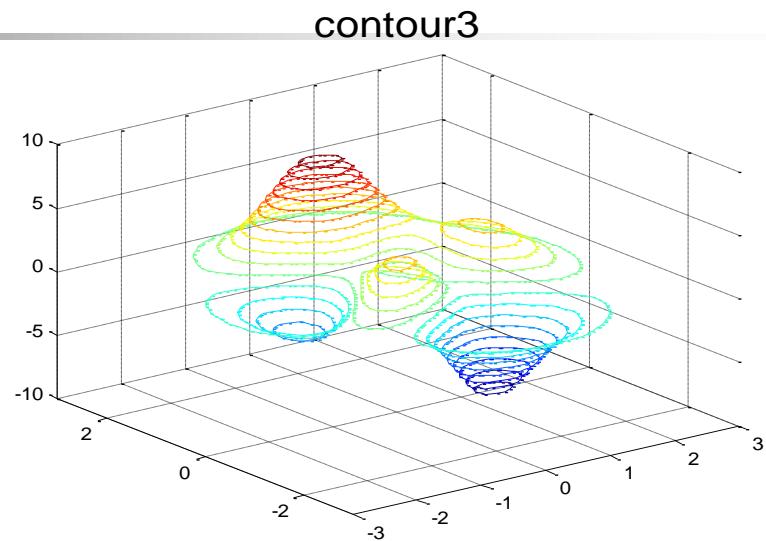
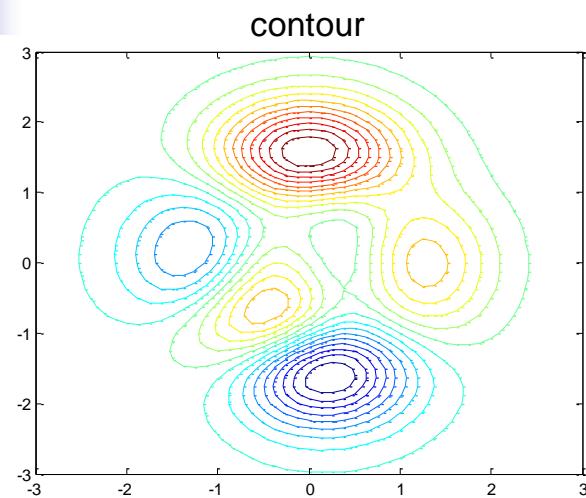
**contour3**: 空间中的等高线

**pcolor** : 在二维平面中以颜色表示曲面的高度

**contourf** :

**clabel**: 标注等高线高度

# 三维作图

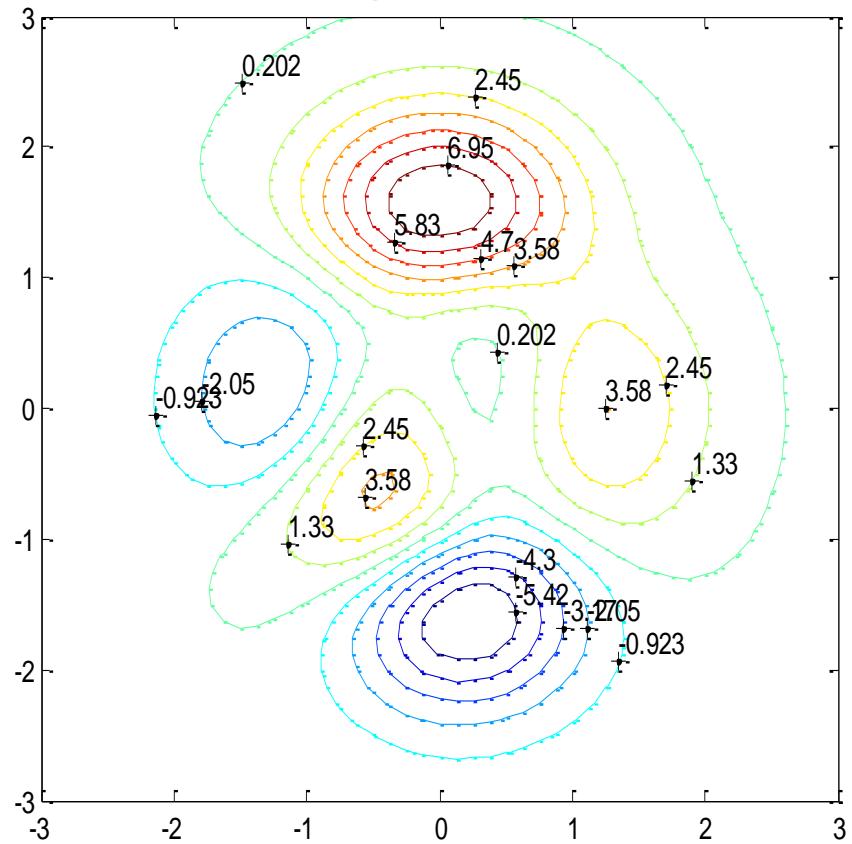


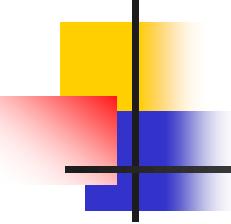
# 三维作图

- 给等高线加高度标注

```
[X,Y,Z] = peaks;  
C = contour(X,Y,Z,12);  
clabel(C)
```

Contour plot with labels





## 三维作图

- $v = f(x, y, z)$  的可视化：

比如：空间各点的温度（标量场）；

海洋中各处水的流速与方向（矢量场）。

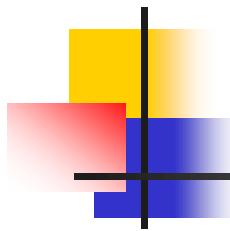
- 常用方法：对三维体切片，在截面处用不同颜色表征在该点的函数值大小。

```
slice(X, Y, Z, V, sx, sy, sz)
```

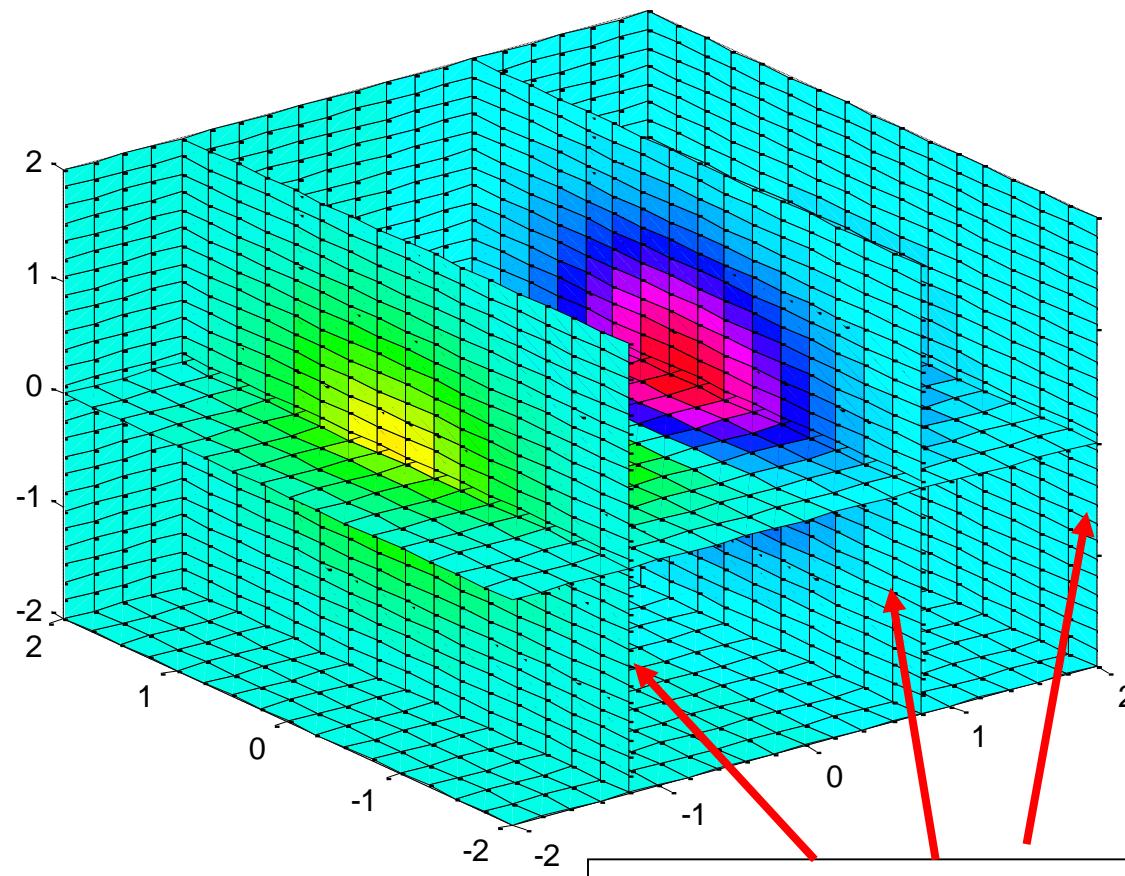
# 三维作图

$$\mathbf{v} = xe^{(-x^2 - y^2 - z^2)}$$

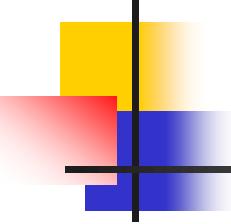
```
[x,y,z] = meshgrid(-2:.2:2,-2:.25:2, ...
-2:.16:2); % 产生三维网格
v = x.*exp(-x.^2-y.^2-z.^2); % 待绘制的函数
xslice = [-1.2,.8,2]; % x=-1.2平面...
yslice = 2;
zslice = [-2,0];
slice(x,y,z,v,xslice,yslice,zslice)
colormap hsv
```



# 三维作图

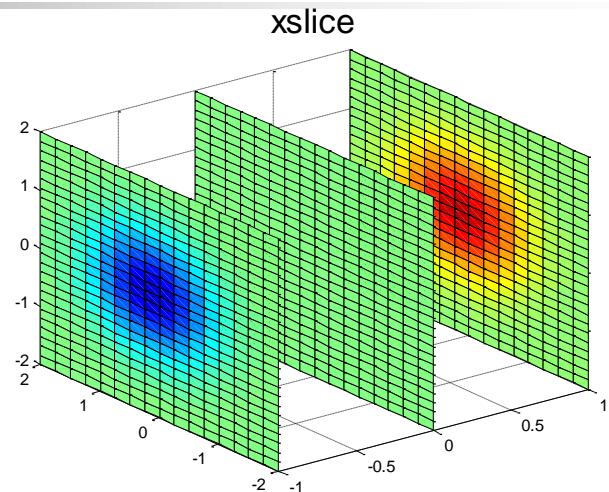


`xslice=[-1.2 ,0.8 2]`



# 三维作图

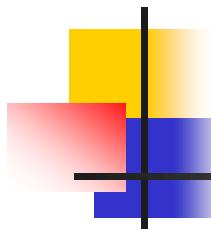
```
slice(x,y,z,v,...  
[-1 0 1],[],[])
```



```
slice(x,y,z,v,...  
[],[-1 0 1], [])
```



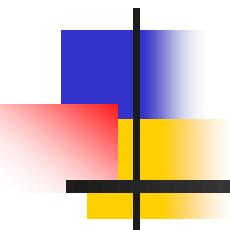
?



# 练习

- 用 **ezmesh** 和 **ezsurf** 分别绘制一个圆环面，并将它们放在一个图形界面内，观察它们的三维内插不同之处。

$$\begin{cases} x = (R + r \cos u) \cos v \\ y = (R + r \cos u) \sin v \\ z = r \sin u \end{cases}$$

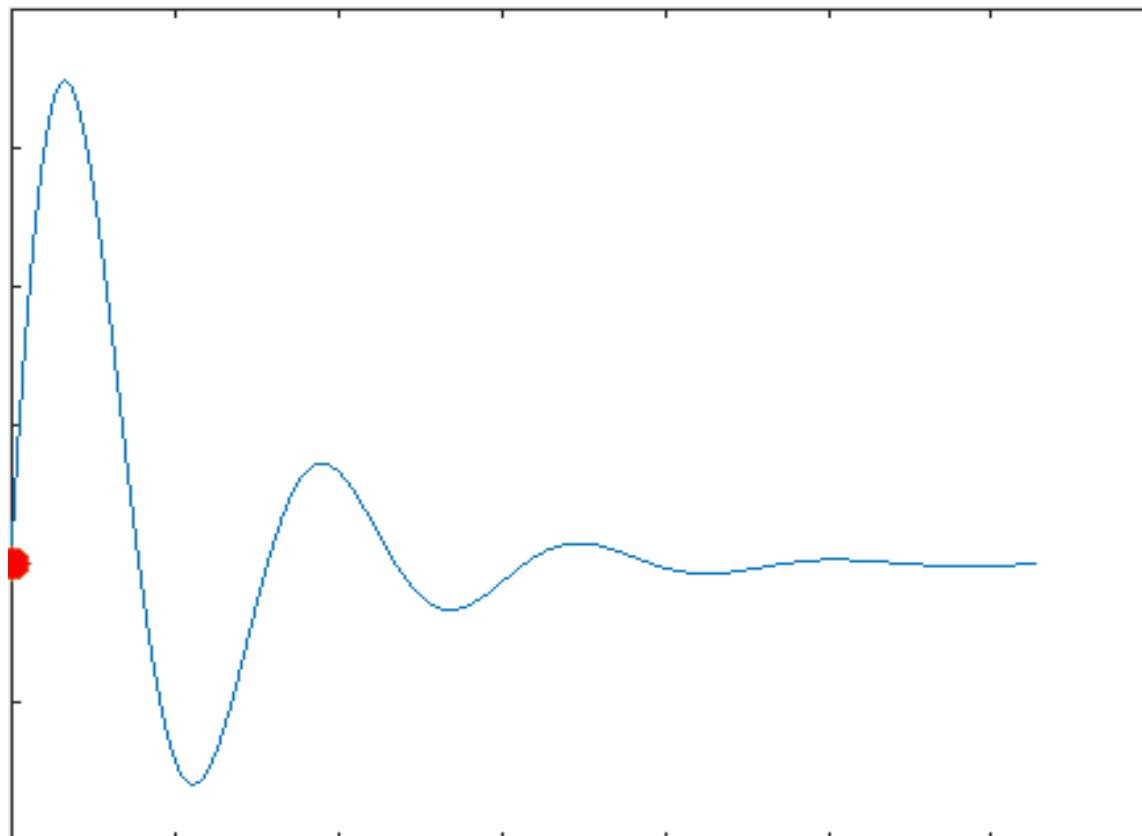


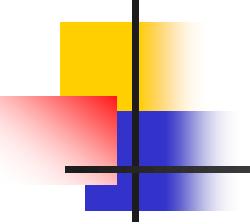
## Part3:

---

# 动态演示

- 一个点在曲线上移动：

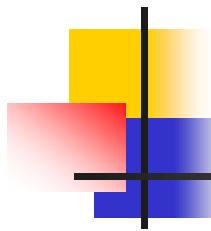




## ■ 一个点在曲线上移动：

```
t=0:0.05:4*pi;  
N = length(t);  
y = sin(2*t).*exp(-t/2);  
h=plot(t , y);  
hold on
```

```
hp = plot(t(1),y(1),'marker','o',  
          'markersize',10,  
          'markerfacecolor','r');  
for k=2:N  
    set(hp,'xdata',t(k),'ydata',y(k));  
    %drawnow  
    pause(0.05)  
end
```



# 将动画存为GIF文件

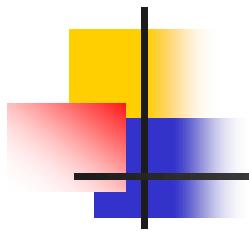
## ■ imwrite 函数

```
F = getframe;
im = frame2im(F);
[I,map] = rgb2ind(im,256); %Gif只能用256色
%写入 GIF89a 格式文件
if k == 1;
    imwrite(I,map,'test.gif','GIF', 'Loopcount',inf,'DelayTime',0.1);
else
    imwrite(I,map, 'test.gif' , 'GIF' , 'WriteMode' , 'append' ,
'DelayTime' ,0.1) ;
end
```

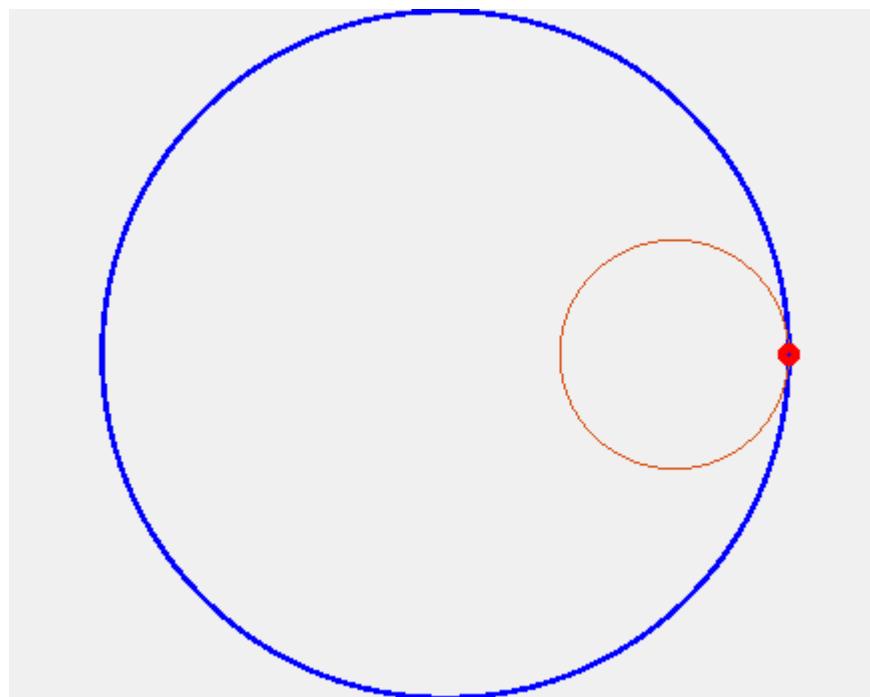
# 内摆线Hypocycloid

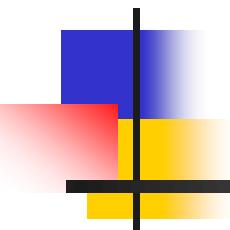
- 定义：一个动圆紧贴另一个圆的内部滚动时，动圆上一定点P的运动轨迹。

$$\begin{cases} x = (R - r)\cos\theta + r\cos\left(\frac{R-r}{r}\theta\right) \\ y = (R - r)\sin\theta - r\sin\left(\frac{R-r}{r}\theta\right) \end{cases}$$



# 内摆线



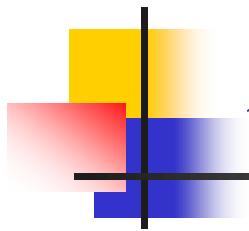


# Matlab编程与应用

## 第四讲

中国科技大学信息学院  
陆伟

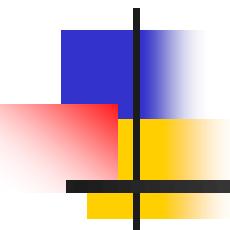
[luwei@ustc.edu.cn](mailto:luwei@ustc.edu.cn)



# 本讲内容

---

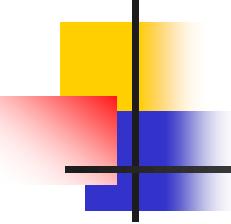
- part1: 图像句柄
- part2: 字符串
- part3: 单元数组与结构体
- part4: 稀疏矩阵



# Part1:

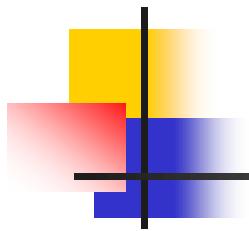
---

## 图像句柄

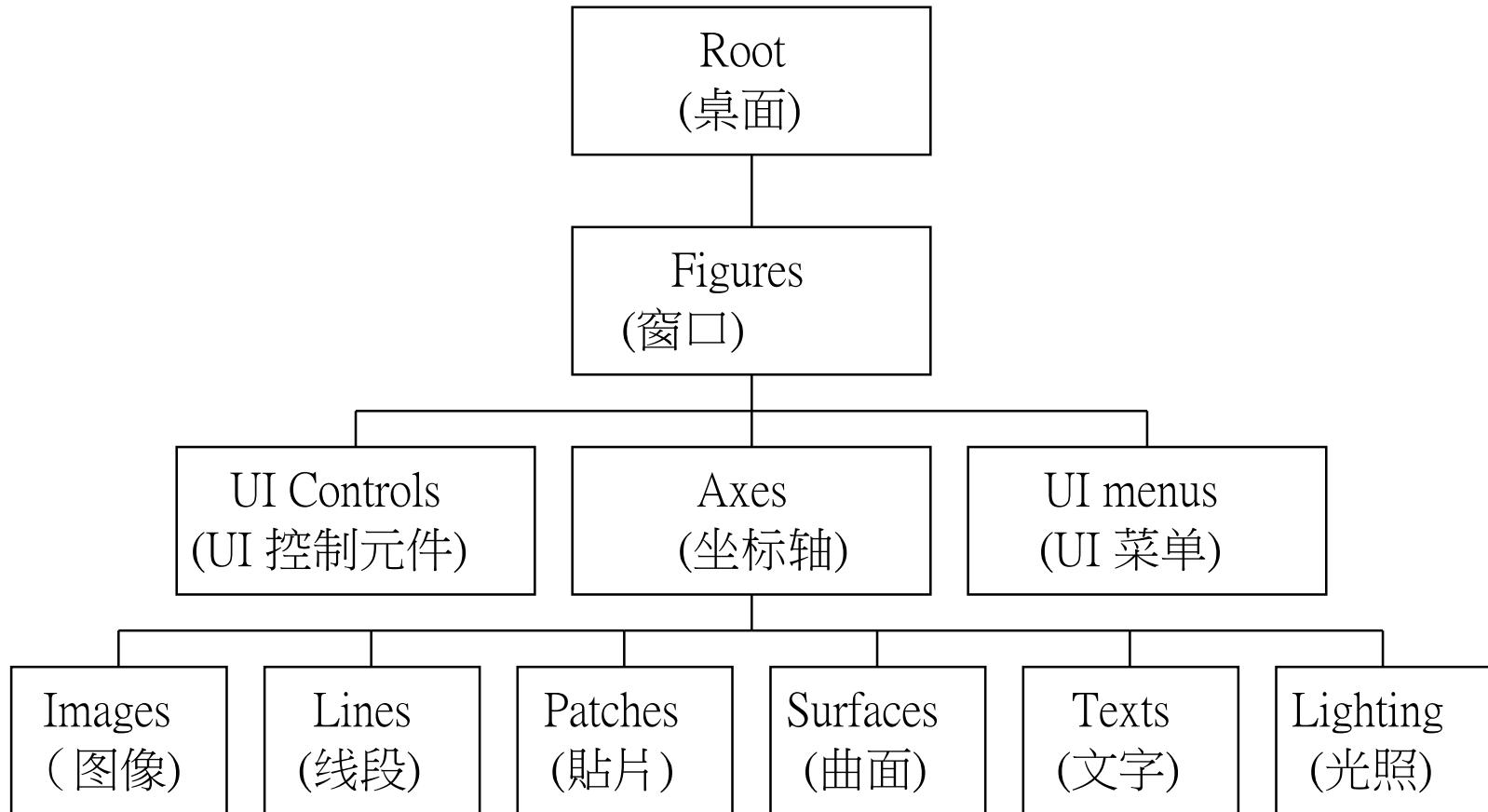


# 图形对象

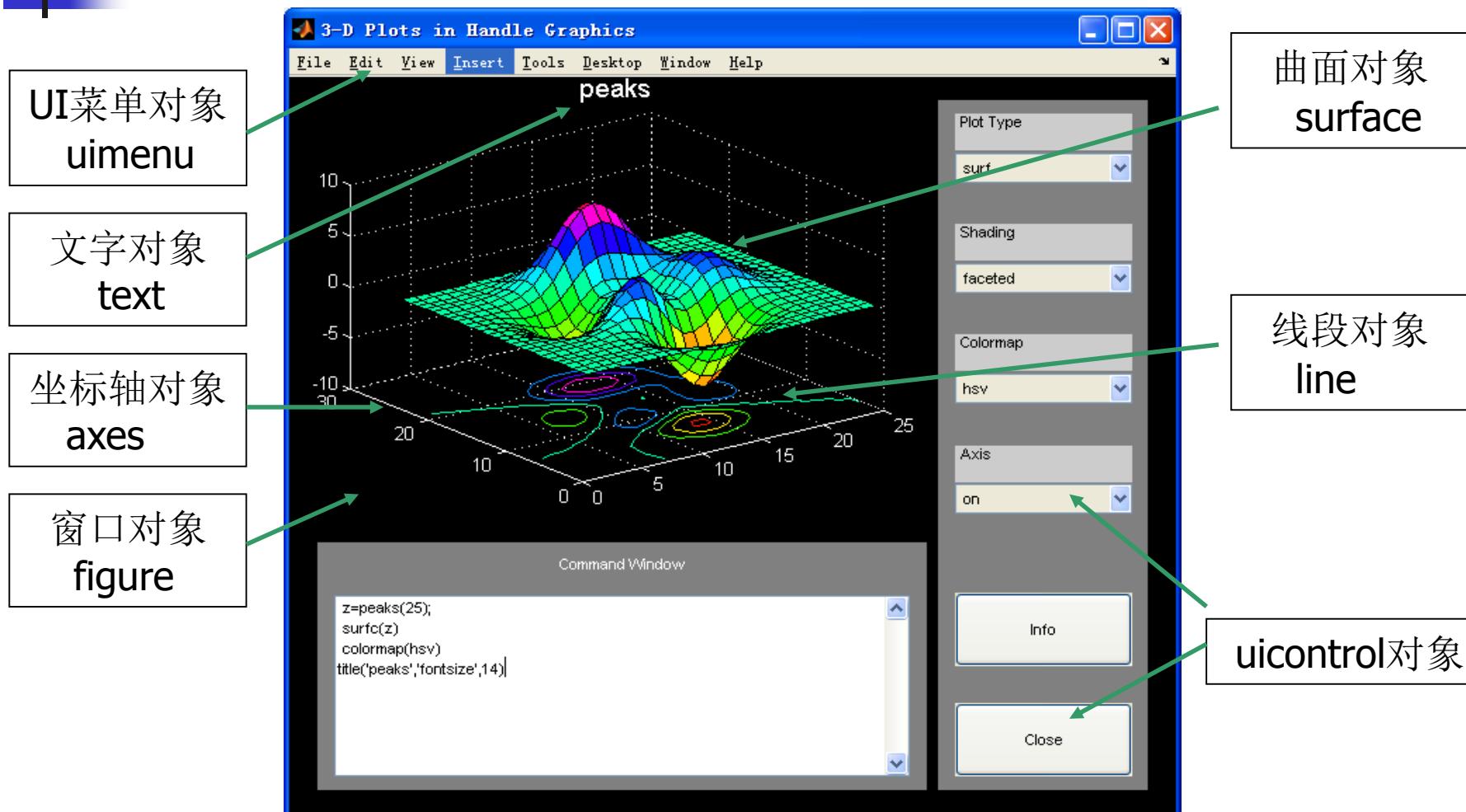
- 每个构成图形的基本单位都被视为一个对象(**Object**)，例如：  
    曲线、曲面、坐标轴、文字...
- 每个对象都分配有一个独一无二的句柄(**handle**)，就像每个人都有一  
    个独一无二的身份证号码
- 根据对象的句柄，就可以修改图形对象的所有属性

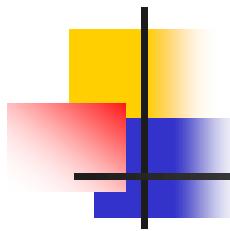


# 图形对象的层次结构



# 图形对象的层次结构





# 图形对象的属性编辑器

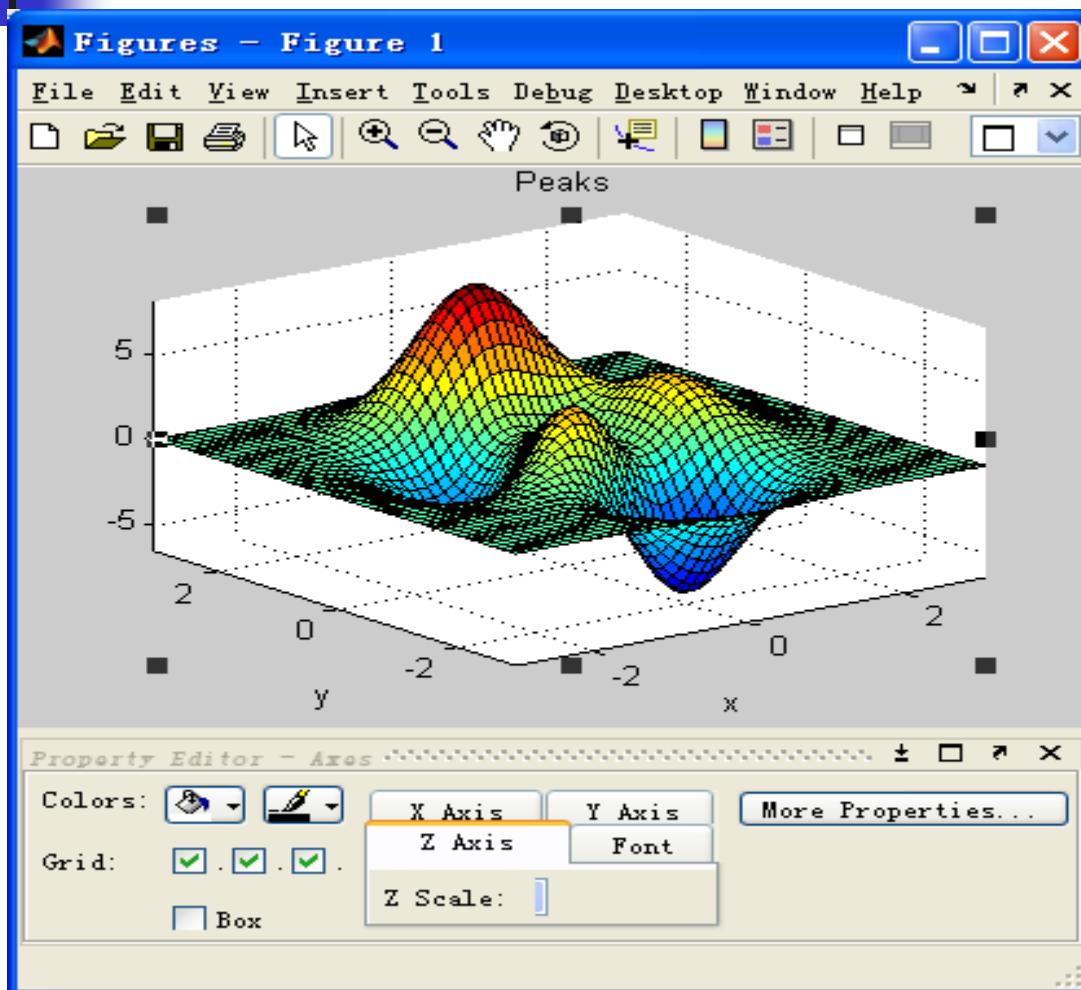
- 先画图，再利用**propedit**开启属性编辑器

```
>>peaks; %画出peaks 3D图
```

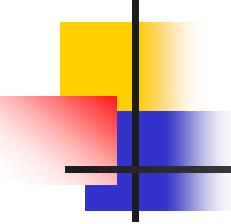
```
>>propedit; %开启属性编辑器
```

- 也可以点选工具行上面的图示  开启图形编辑器

# 图形对象编辑器



- 鼠标左键点击希望编辑的对象（如 figure、axes、text、进行修改。



# 图形对象的属性编辑

- 常常需要在命令行或m文件中对图形对象的属性进行修改

**set**：设定某图形对象的某个属性值

**get**: 获得某个图形对象的现有属性值

**findobj**: 在句柄图形的层次结构中，找到想要进行编辑的图形对象。

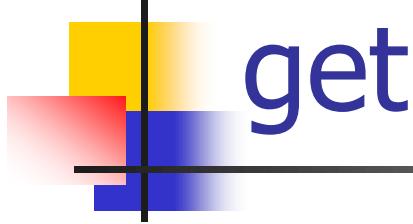
**gcf**:

**gca**:

**gco**:

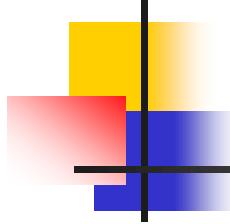
# set

```
t = 0 : 0.1 :4*pi;  
y = exp(-t/4).*sin(t);  
h = plot(t,y);  
set(h,'linewidth', 3);  
set(h,'Marker','o');  
set(h,'markersize',20);
```



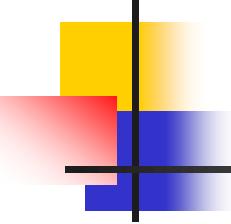
# get

```
>>get(h,' linewidth' );  
  
>>get(h,' color' );  
  
>>get(h)
```



# findobj

```
>>plot(rand(10,2));  
  
>>h = findobj(0,'type','line');  
  
>>set(h,'linewidth',3);
```



# axes与axis的区别

- axes是创建坐标轴， axis是设定其范围。

```
clear all;  
x=0:10*pi;  
y=sin(x);
```

% 创建一个坐标系。起点是左边占到显示窗口的十分之一处，  
%下边占到十分之二处， 宽占十分之三， 高占十分之四。

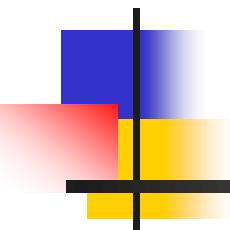
```
axes('position',[0.1 0.2 0.3 0.4]);
```

```
plot(x,y); %画图。
```

% 设置x的坐标范围是0到 $2\pi$ ， y的范围是-0.5到0.5。看横纵坐标

```
axis([0 2*pi -0.5 0.5]);
```

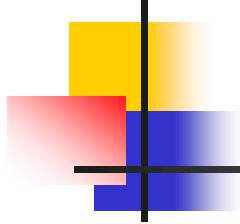
```
%axes;
```



## Part2:

---

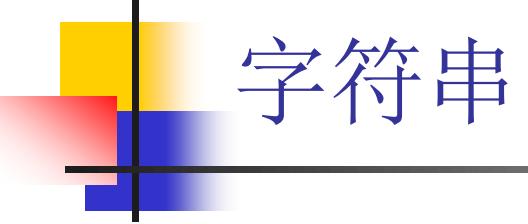
# Matlab字符串



# 主要内容

---

- 字符串生成
- 与数值之间转化
- 查找
- 匹配
- 连接
- 比较



# 字符串

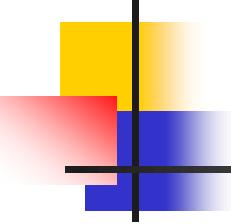
- 字符串是ASCII数值型数组，每个字符用两个字节表示，只是显示为字符形式。

- 应用：

给用户提示：‘please enter the r:’

显示结果：‘The distance is 8 mm.’

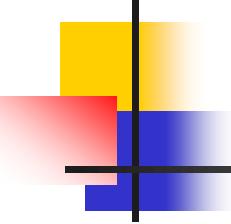
图形注释：



# 字符串

- 字符串：用单引号括起来的一个或多个字符。

```
>> str1 = 'Hello world!'
str1 =
Hello world!
>> size(str1)
ans =
     1      12
>> whos
  Name    Size    Bytes    Class         Attributes
  str1    1x12        24    char
```

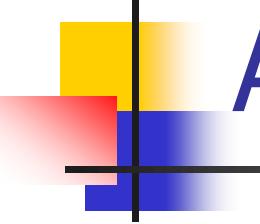


# 字符串

- 字符串显示为对应的ASCII码值：

```
>> double(str1)
ans =
    Columns 1 through 10
        72 101 108 108 111 32 119 111 114 108
    Columns 11 through 12
        100      33
```

```
>> double('abc 0123')
ans =
    97  98  99  32  32  48  49  50  51
```



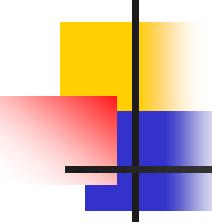
# ASCII码

- ASCII : American Standard Code for Information Interchange  
美国信息交换标准代码
- 基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语，统一规定了上述常用符号用哪些二进制数来表示。
- 汉字编码 GB2312标准 双字节

ASCII 字符代码表 一

| 高四位  |    | ASCII非打印控制字符 |               |        |       |    |      |         |        |     |    | ASCII 打印字符 |    |      |    |      |    |      |    |      |    |            |  |
|------|----|--------------|---------------|--------|-------|----|------|---------|--------|-----|----|------------|----|------|----|------|----|------|----|------|----|------------|--|
|      |    | 0000         |               |        |       |    | 0001 |         |        |     |    | 0010       |    | 0011 |    | 0100 |    | 0101 |    | 0110 |    | 0111       |  |
|      |    | 0            |               |        |       |    | 1    |         |        |     |    | 2          |    | 3    |    | 4    |    | 5    |    | 6    |    | 7          |  |
| +逆制  | 字符 | ctrl         | 代码            | 字符解释   | +逆制   | 字符 | ctrl | 代码      | 字符解释   | +逆制 | 字符 | +逆制        | 字符 | +逆制  | 字符 | +逆制  | 字符 | +逆制  | 字符 | +逆制  | 字符 | ctrl       |  |
| 0000 | 0  | 0            | BLANK<br>NULL | ^@ NUL | 空     | 16 | ►    | ^P DLE  | 数据链路转意 | 32  |    | 48         | 0  | 64   | @  | 80   | P  | 96   | '  | 112  | p  |            |  |
| 0001 | 1  | 1            | ☺             | ^A SOH | 头标开始  | 17 | ◀    | ^Q DC1  | 设备控制 1 | 33  | !  | 49         | 1  | 65   | A  | 81   | Q  | 97   | a  | 113  | q  |            |  |
| 0010 | 2  | 2            | ☻             | ^B STX | 正文开始  | 18 | ↕    | ^R DC2  | 设备控制 2 | 34  | "  | 50         | 2  | 66   | B  | 82   | R  | 98   | b  | 114  | r  |            |  |
| 0011 | 3  | 3            | ♥             | ^C ETX | 正文结束  | 19 | !!   | ^S DC3  | 设备控制 3 | 35  | #  | 51         | 3  | 67   | C  | 83   | S  | 99   | c  | 115  | s  |            |  |
| 0100 | 4  | 4            | ◆             | ^D EOT | 传输结束  | 20 | ¶    | ^T DC4  | 设备控制 4 | 36  | \$ | 52         | 4  | 68   | D  | 84   | T  | 100  | d  | 116  | t  |            |  |
| 0101 | 5  | 5            | ♣             | ^E ENQ | 查询    | 21 | ƒ    | ^U NAK  | 反确认    | 37  | %  | 53         | 5  | 69   | E  | 85   | U  | 101  | e  | 117  | u  |            |  |
| 0110 | 6  | 6            | ♠             | ^F ACK | 确认    | 22 | ■    | ^V SYN  | 同步空闲   | 38  | &  | 54         | 6  | 70   | F  | 86   | V  | 102  | f  | 118  | v  |            |  |
| 0111 | 7  | 7            | ●             | ^G BEL | 震铃    | 23 | ↕    | ^W ETB  | 传输块结束  | 39  | '  | 55         | 7  | 71   | G  | 87   | W  | 103  | g  | 119  | w  |            |  |
| 1000 | 8  | 8            | ▣             | ^H BS  | 退格    | 24 | ↑    | ^X CAN  | 取消     | 40  | (  | 56         | 8  | 72   | H  | 88   | X  | 104  | h  | 120  | x  |            |  |
| 1001 | 9  | 9            | ○             | ^I TAB | 水平制表符 | 25 | ↓    | ^Y EM   | 媒体结束   | 41  | )  | 57         | 9  | 73   | I  | 89   | Y  | 105  | i  | 121  | y  |            |  |
| 1010 | A  | 10           | ◎             | ^J LF  | 换行/新行 | 26 | →    | ^Z SUB  | 替换     | 42  | *  | 58         | :  | 74   | J  | 90   | Z  | 106  | j  | 122  | z  |            |  |
| 1011 | B  | 11           | ♂             | ^K VT  | 竖直制表符 | 27 | ←    | ^[_ ESC | 转意     | 43  | +  | 59         | ;  | 75   | K  | 91   | [  | 107  | k  | 123  | {  |            |  |
| 1100 | C  | 12           | ♀             | ^L FF  | 换页/新页 | 28 | ∟    | ^` FS   | 文件分隔符  | 44  | ,  | 60         | <  | 76   | L  | 92   | \  | 108  | l  | 124  |    |            |  |
| 1101 | D  | 13           | ♪             | ^M CR  | 回车    | 29 | ↔    | ^] GS   | 组分隔符   | 45  | -  | 61         | =  | 77   | M  | 93   | ]  | 109  | m  | 125  | }  |            |  |
| 1110 | E  | 14           | ♫             | ^N SO  | 移出    | 30 | ▲    | ^_ RS   | 记录分隔符  | 46  | .  | 62         | >  | 78   | N  | 94   | ^  | 110  | n  | 126  | ~  |            |  |
| 1111 | F  | 15           | ⌚             | ^O SI  | 移入    | 31 | ▼    | ^- US   | 单元分隔符  | 47  | /  | 63         | ?  | 79   | O  | 95   | _  | 111  | o  | 127  | △  | Back space |  |

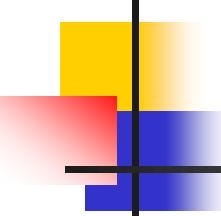
注：表中的ASCII字符可以用：ALT + “小键盘上的数字键” 输入



# 字符串

## ■ ASCII码中数值转换为字符串： **char**

```
>> x = 'a' %变量x是字符串，含有一个字符a  
x =  
a  
>> y=double(x) %将字符a转换为ASCII码  
y =  
    97  
>> z = char(y) %将ASCII码转换为字符  
z =  
a
```



# 字符串

- ASCII码中数值转换为字符串： **char**

```
>> x = [];
>> for k = 1:10
    x = [x, 'a'+k-1];
end
>> x
x =
    97  98  99  100  101  102  103  104  105  106
>> char(x)
ans =
abcdefghijklmnopqrstuvwxyz
```

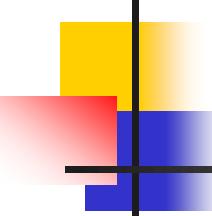
# 字符串

- 双精度数转换成字符串： num2str

```
>> x=31.2;
>> y = num2str(x)
y = 31.2
>> whos
  Name      Size      Bytes  Class       Attributes
  x            1x1          8  double
  y            1x4          8  char
```

**x=39; a = num2str(x); b = char(x);**

问： a是否等于b？

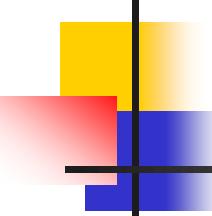


# 字符串

- 字符串转换成双精度数： **str2num**

```
>> str = '36' ;
>> x=str2num(str) %就是36
x = 36
>> y = double(str) %变成对应的ASCII码
y = 51      54
>> whos
```

| Name | Size | Bytes | Class  |
|------|------|-------|--------|
| str  | 1x2  | 4     | char   |
| x    | 1x1  | 8     | double |
| y    | 1x2  | 16    | double |



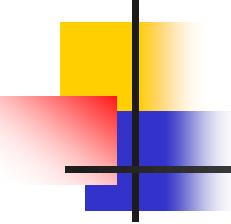
# 字符串

- 字符串显示: **disp**   **sprintf**

```
disp('Now ,do the second step...')
```

%用于提示程序进程

```
for k = 1:10
...
    disp(['Loop ' int2str(k) ' is done'])
%
end
```



# 字符串

- `sprintf(formatSpec,A1,...,An)`

```
str1 = sprintf('The value of pi is %.2f',pi)
```

```
str2 = sprintf('Some numbers:%5d,%2d',33,2^3)
```

`%d` 整数

`%f` 浮点数

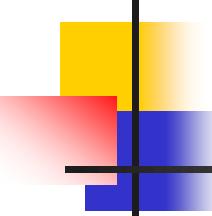
`%c` 单个字符

`%s` 字符串

`%5d` 整数

`%6.2f` 浮点数共6位，小数2位

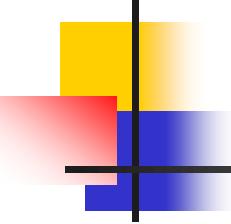
`%.3f` 指定浮点数小数位数为3



# 字符串

- 字符串数组: char

```
>> kemu = char('math', 'english', 'DSP')
kemu =
math
english
DSP      %字符串长度不等, matlab会填充空格使其相等
>> kemu(1)
ans = m
>> kemu(1, :)
ans = math
>> a = kemu(1, :)
a = math
>> a = deblank(a) %清除空格
a =  math
```



# 字符串

---

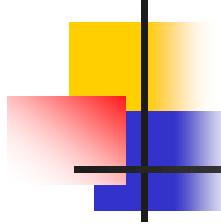
- 字符串操作

**strcat**: 连接两个或多个字符串

**strcmp**: 比较两个字符串是否完全一致

**strstr**: 在一个字符串中查找特定子字符串

**strrep**: 用一个子字符串代替特定字符串



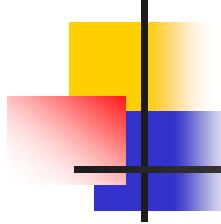
# 字符串

```
a = 'hello '
b = 'goodbye'
strcat(a, b)
ans =
hellogoodbye
[a b]
ans =
hello goodbye
```

# 字符串

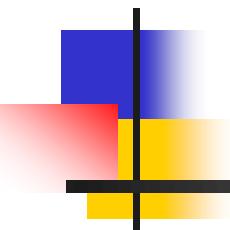
```
strcmp('Yes', 'No')
ans = 0
strcmp('Yes', 'Yes')
ans = 1
```

```
S = 'Find the starting indices of the ...
      pattern string';
strfind(S, 'in')
ans = 2 15 19 45
strfind(S, 'In')
ans = []
```



# 字符串

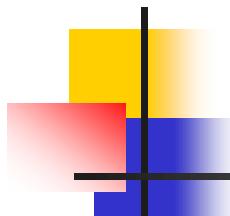
```
s1 = 'This is a good example.';  
str = strrep(s1, 'good', 'great')  
str =  
This is a great example.
```



## Part3:

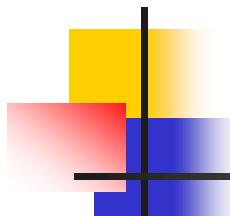
---

单元数组 (cell array) 与  
结构体 (structures)



# 单元数组 (cell array)

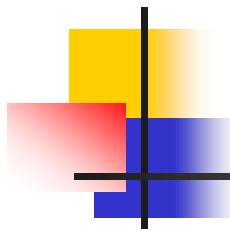
- 将不同类型的、相关的数据集成在一个变量中。
- 该变量称为单元数组，其中的每个元素称为单元 (cell)。
- cell可以是任何数据类型：字符串、双精度数组、其他单元数组。不同的cell可以包含不同的数据类型。



# 单元数组 (cell array)

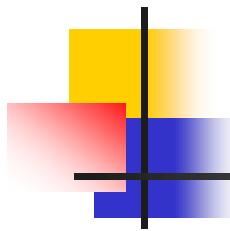
## 3-by-3 Cell Array

|  |                                     |                        |   |   |   |   |   |   |   |                                 |                        |
|--|-------------------------------------|------------------------|---|---|---|---|---|---|---|---------------------------------|------------------------|
| <b>cell 1,1</b><br><table border="1"><tr><td>1</td><td>4</td><td>3</td></tr><tr><td>0</td><td>5</td><td>8</td></tr><tr><td>7</td><td>2</td><td>9</td></tr></table> | 1                                   | 4                      | 3 | 0 | 5 | 8 | 7 | 2 | 9 | <b>cell 1,2</b><br>'Anne Smith' | <b>cell 1,3</b><br>[ ] |
| 1  | 4                                   | 3                      |   |   |   |   |   |   |   |                                 |                        |
| 0  | 5                                   | 8                      |   |   |   |   |   |   |   |                                 |                        |
| 7  | 2                                   | 9                      |   |   |   |   |   |   |   |                                 |                        |
| <b>cell 2,1</b><br>3+7i  | <b>cell 2,2</b><br>[-3.14 ... 3.14] | <b>cell 2,3</b><br>[ ] |   |   |   |   |   |   |   |                                 |                        |
| <b>cell 3,1</b><br>[ ]   | <b>cell 3,2</b><br>[ ]              | <b>cell 3,3</b><br>5   |   |   |   |   |   |   |   |                                 |                        |



# 单元数组 (cell array)

- 用途：使得大量的相关数据的处理与引用变得简单而方便。  
比如：一段处理的语音信号，除了语音数据外，还希望记录相关信息，如说话人；性别；采样率；录制时间；
- GUI、Simulink程序中数据的传递



# 创建单元数组

- 创建一个 $2 \times 2$ 的单元数组：

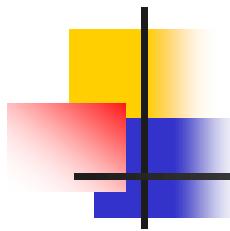
方法一：大括号在右边：

$A(1,1) = \{ [1 \ 2 \ 3 ; 4 \ 5 \ 6] \};$

$A(1,2) = \{3+4i\};$

$A(2,1) = \{'hello world!' \};$

$A(2,2) = \{1:10\};$



# 创建单元数组

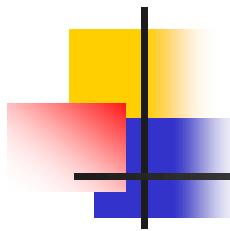
方法二：大括号在左边：

`A{1,1} = [1 2 3 ; 4 5 6];`

`A{1,2} = 3+4i;`

`A{2,1} = 'hello world!';`

`A{2,2} = 1:10 ;`

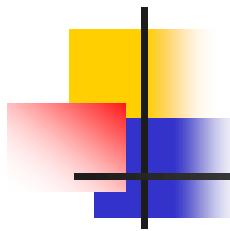


## 创建单元数组

方法三：直接赋值

B = { [1 2], '张三', , 2+3i ,5  };

C = { [1:10], 'USTC'; 4-2j,2 };



# 创建单元数组

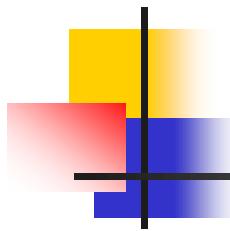
- 方法四：首先生成一个空单元数组，再添加数据。

```
C = cell(2,3)
```

```
C(1,1) = 'this is wrong' ;
```

```
C(1,1) = {'this is right'} ;
```

```
C{2,2} = 'this works too';
```

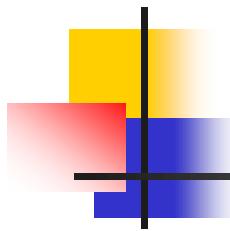


# 单元数组的内容显示

```
>> A
```

```
A =
```

|                |                    |
|----------------|--------------------|
| [2x3 double]   | [3.0000 + 4.0000i] |
| 'hello,world!' | [1x10 double]      |



# 单元数组的内容显示(celldisp)

```
>> celldisp(A)
```

```
A{1,1} =
```

```
    1   2   3  
    4   5   6
```

```
A{2,1} =
```

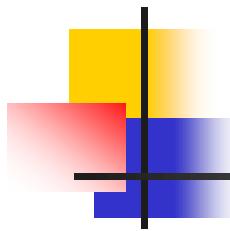
```
    hello,world!
```

```
A{1,2} =
```

```
    3.0000 + 4.0000i
```

```
A{2,2} =
```

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



# 单元数组的内容显示

```
>> A{:}
```

```
ans =
```

```
1 2 3  
4 5 6
```

```
ans =
```

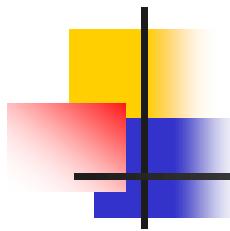
```
hello,world!
```

```
ans =
```

```
3.0000 + 4.0000i
```

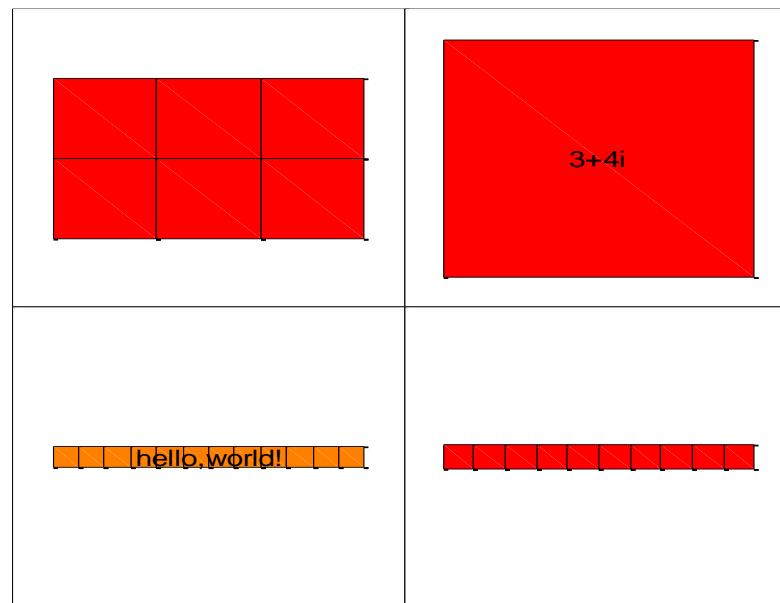
```
ans =
```

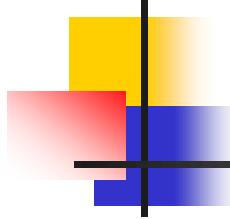
```
1 2 3 4 5 6 7 8 9 10
```



# 单元数组的内容显示

```
>> cellplot(A)
```





# 单元内容获取

```
>> x = A(2,2)
```

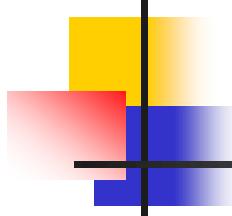
```
x =
```

```
[1x10 double]
```

```
>> x=A{2,2}
```

```
x =
```

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



# 单元内容删除

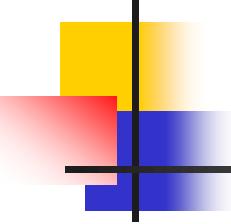
```
>> A(2,1)=[]
```

??? A null assignment can have only one non-colon index.

```
>> A{2,1} =[]
```

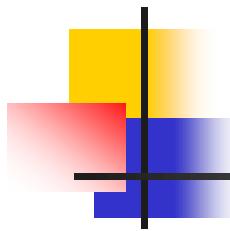
A =

```
[2x3 double]  [3.0000 + 4.0000i]  
      []        [1x10 double]
```



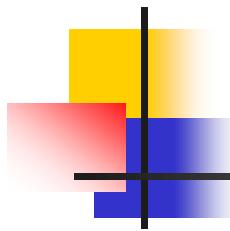
# 结构体(structures)

- 类似与单元数组，将不同类型的数据集成在一个变量中；
- 结构体中每个元素称为字段(field)；
- 每个字段(field)可以是任何数据类型：字符串、双精度数组、其他单元数组。不同字段可以包含不同的数据类型。



# 创建结构体

- 创建一个包含学生个人资料的结构体 `student`, 可能的字段有: `name`、`id`、`scores`等。
- `student.name = '小明' ;`  
`student.id = 'PB1234567';`  
`student.scores = [98,92,90];`

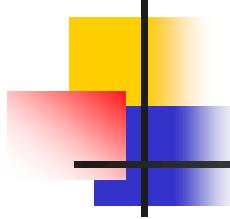


# 创建结构体

```
student =  
    name: '小明'  
    id: 'PB1234567'  
    scores: [98 92 90]
```

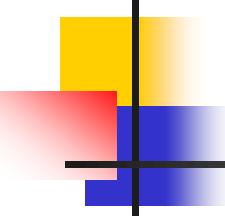
要加入第二个学生的资料，可以：

```
student(2).name = '小刚' ;  
student(2).id = 'PB2345678';  
student(2).scores = [75,100,86];
```



# 创建结构体

```
student =  
1x2 struct array with fields:  
    name  
    id  
    scores
```



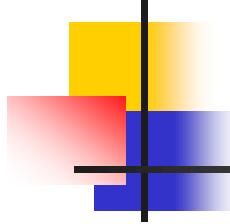
# 结构体

---

- 创建结构体变量circle:

```
>> circle.radius = 2.5;  
>> circle.center = [0,1];  
>> circle.linestyle = '--';  
>> circle.color = 'red'
```

```
circle =  
    radius: 2.5000  
    center: [0 1]  
    linestyle: '--'  
    color: 'red'
```



# 结构体

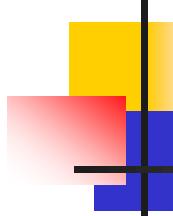
```
>> size(circle)
```

```
ans =
```

```
1 1
```

```
>> whos
```

| Name   | Size | Bytes | Class  | Attributes |
|--------|------|-------|--------|------------|
| circle | 1x1  | 530   | struct |            |

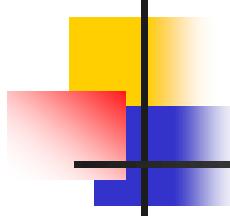


# 结构体

希望再加入一个圆：

```
>> circle(2).radius = 3.4;  
>> circle(2).color = 'green';  
>> circle(2).linestyle = ':';  
>> circle(2).center = [2.3 -1.2]
```

```
circle =  
1x2 struct array with fields:  
    radius  
    center  
    linestyle  
    color
```



# 字段内容的获取

```
>> rad2 = circle(2).radius
```

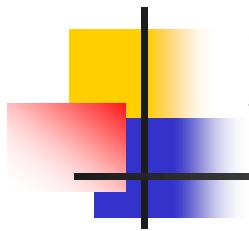
```
rad2 =
```

```
3.4000
```

```
>> area1 = pi*circle(1).radius^2
```

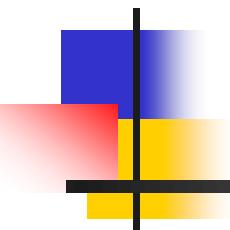
```
area1 =
```

```
19.6350
```



# 获取结构体的字段信息

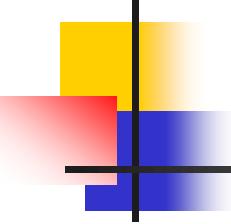
- `s(1,1).name = 'alice';`
- `s(1,1).ID = 0;`
- `s(2,1).name = 'lw';`
- `s(2,1).ID = 1;`
  
- `names = fieldnames(s)`



## Part3:

---

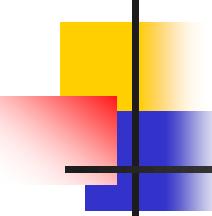
# 稀疏矩阵



# 稀疏矩阵

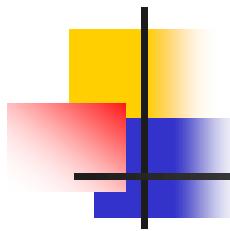
- 稀疏矩阵：指大多数元素为0，只有少数非零元素的矩阵。

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 7 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



# 稀疏矩阵

- 实际应用中常常会遇到：  
描述电路拓扑结构的矩阵、电网、交通。
- 若按普通方式(**Full Matrix**)存储稀疏矩阵，占用大量内存空间和运算时间。
- 稀疏矩阵的存储方式：  
只存储其中的非0元素，并记录相应的行、列位置。



# 稀疏矩阵的创建

- 将全矩阵(Full Matrix)转换为稀疏矩阵.

```
S=sparse(A)
```

例:

```
A=[0 0 5 0; 3 0 3 0 ; 0 0 0 1; 0 4 3 0]
```

```
S=sparse(A)
```

```
whos
```

# 稀疏矩阵的创建

**A =** 0 0 5 0

3 0 3 0

0 0 0 1

0 4 3 0

**S =**(2,1) 3

(4,2) 4

(1,3) 5

(2,3) 3

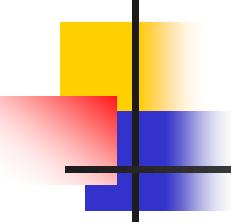
(4,3) 3

(3,4) 1

**Name Size Bytes Class Attributes**

**A** 4x4 128 double

**S** 4x4 92 double sparse



# 稀疏矩阵的创建

## ■ 直接创建稀疏矩阵：

```
S=sparse(i,j,s,m,n)
```

i 和 j 分别是矩阵非零元素的行和列指标向量，

s 是非零元素值向量，

m, n 分别是矩阵的行数和列数。

i、j、s是长度相同的向量。

s(k) 的二维下标即是 i(k) 与 j(k) .

# 稀疏矩阵的创建

```
>> S = sparse([1 3 2], [1 2 4], [2 4 1], 3, 4)
```

```
S = (1,1)    2  
      (3,2)    4  
      (2,4)    1
```

- 也可以在sparse指令中加入第六个参数：

```
S=sparse(i,j,s,m,n,nzmax)
```

- 最后一个参数nzmax告诉matlab该稀疏矩阵最多有多少个非零元素，便于预先分配内存。

# 稀疏矩阵的创建

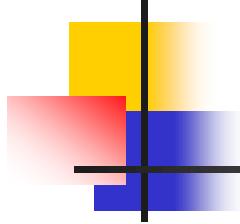
## 从文件中创建稀疏矩阵

利用**load**和**spconvert**函数可以从包含一系列下标和非零元素的文本文件中输入稀疏矩阵。

例：设文本文件 **T.txt** 中有三列内容，第一列是一些行下标，第二列是列下标，第三列是非零元素值。

|   |   |   |
|---|---|---|
| 1 | 3 | 5 |
| 2 | 1 | 3 |
| 2 | 3 | 3 |
| 3 | 4 | 1 |
| 4 | 2 | 4 |
| 4 | 3 | 3 |

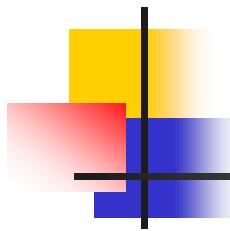
```
load T.txt  
S=spconvert(T)
```



# 稀疏矩阵

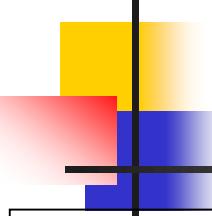
- 稀疏矩阵转换为全矩阵：

**A=full(S)**



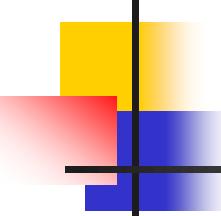
# 稀疏矩阵信息查看

- **nnz (S)** : 返回S中非零元素的个数
- **nonzeros (S)** : 返回列向量，含所有非零元素；
- **nzmax (S)** :
- **spy (S)** : 图形化形式表达稀疏矩阵；
- **[i, j, s]=find (S)**



# 稀疏矩阵运算

```
>> S = sparse([1 3 2], [1 2 4], [2 4 1], 3, 4);  
>> x = sin(S);  
>> x = size(S);  
>> A = full(S);  
>> B = A.*S;  
>> C = A + S;
```

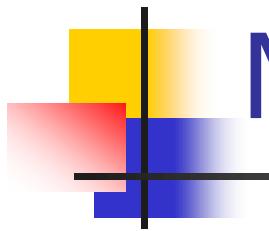


# 稀疏矩阵运算

- MATLAB提供了许多可以应用于稀疏矩阵处理的函数

## Linear algebra.

|          |                                      |
|----------|--------------------------------------|
| eigs     | - A few eigenvalues, using ARPACK.   |
| svds     | - A few singular values, using eigs. |
| ilu      | - Incomplete LU factorization.       |
| luinc    | - Incomplete LU factorization.       |
| cholinc  | - Incomplete Cholesky factorization. |
| normest  | - Estimate the matrix 2-norm.        |
| condeest | - 1-norm condition number estimate.  |
| sprank   | - Structural rank.                   |



# Matlab音频信号相关函数

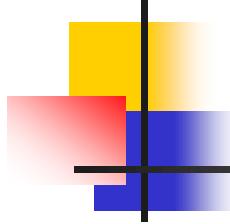
**audioread** %从文件中读取音频信号

**audiowrite** %将音频数据写入文件

**sound** %播放声音

**soundsc** %播放声音

**audiorecorder** %录音



# 录音

## ■ audiorecorder(Fs,nBits,nChannels)

```
recObj = audiorecorder; % 默认: 单声道 , 采样率fs  
= 8000bps , 量化精度nbits = 8bit
```

```
recordblocking( recObj, 5) %录音5秒钟
```

```
play(recObj); %播放声音
```

```
y = getaudiodata(recObj);
```

```
plot(y);
```

# Matlab音频信号相关函数 (老版matlab)

**wavread** %从wav文件中读取语音信号

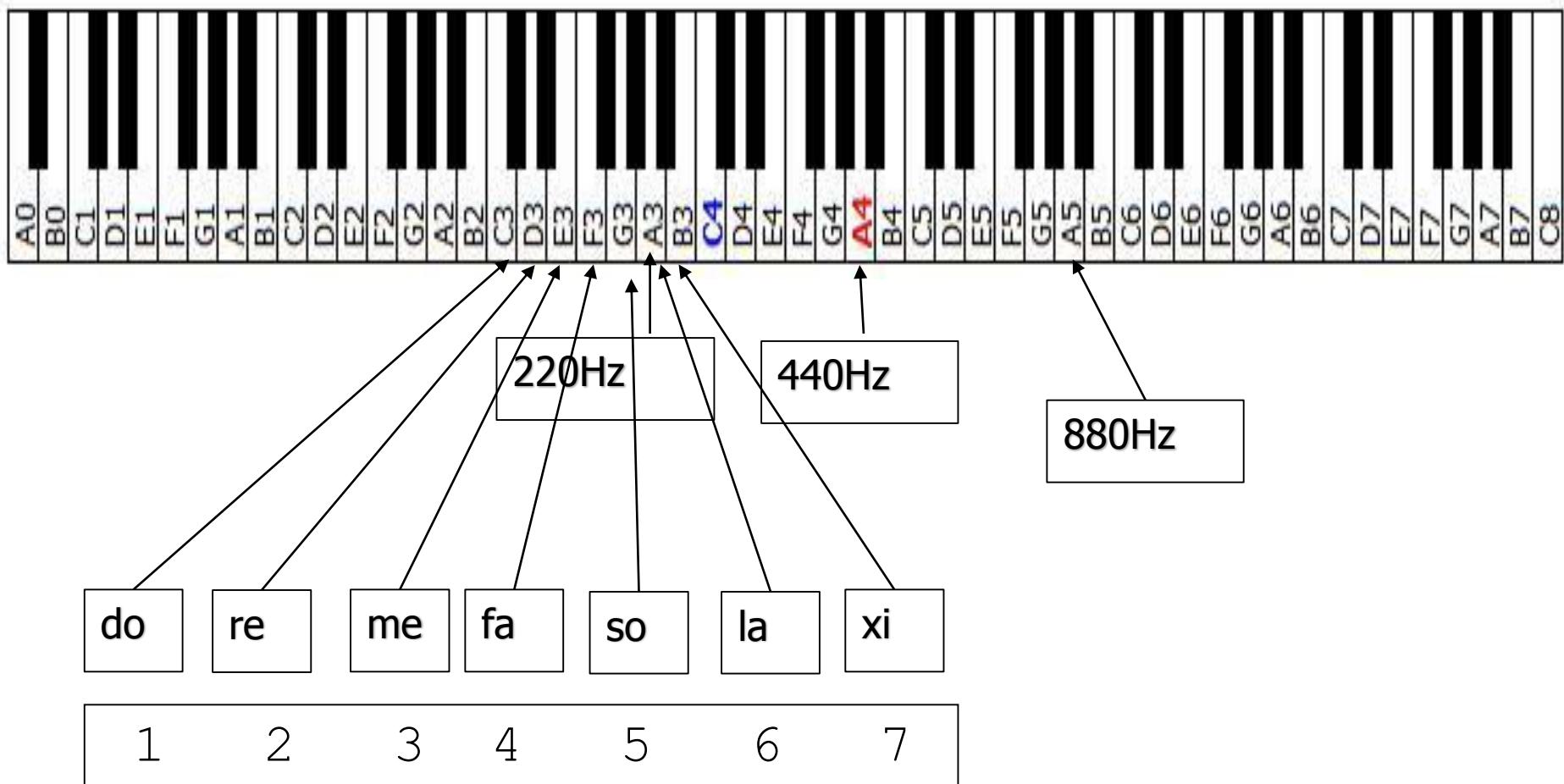
**wavrecord** %录音

**wavwrite** %将音频数据写入wav文件

**sound** %播放声音

**soundsc** %播放声音

# 音阶合成



# 音阶合成

## ■ 十二平均律

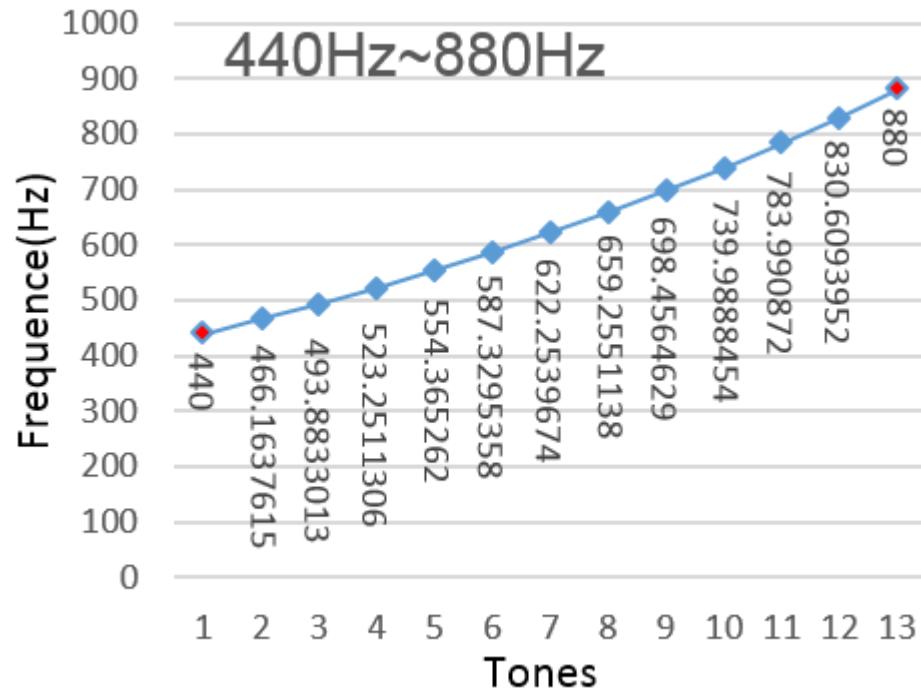
频率 (Hz) :

110-220-440-880

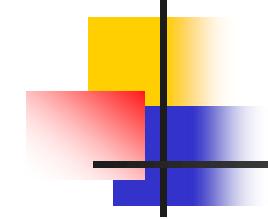
两个相邻的频率为一个八度

比如在440Hz-880Hz之间，  
按等比关系分成12个单音，  
每个单音的频率为：

$$440 \times 2^{\frac{k}{12}} \quad k = 0, 1, 2 \dots 12$$



图片来源： [《程序控》博客 --](http://www.cnblogs.com/devymex/)  
<http://www.cnblogs.com/devymex/>



```
Fs = 44100; T = 1/Fs; t = 0:T:0.4;
fA=440; x = [0:12];
f = fA*2.^x/12; flow = fA/2*2.^x/12; fhigh = fA*2*2.^x/12;
note = @(n) sin(2*pi*f(n)*t);
notelow = @(n) sin(2*pi*flow(n)*t);
notehigh = @(n) sin(2*pi*fhigh(n)*t) ;
notec = @(n) 0.6*notelow(n)+note(n)+0.6*notehigh(n);
m123 = [note(1),note(3),note(5)];
m123_c = [notec(1),notec(3),notec(5)];
mall_c =
[notec(1),notec(3),notec(5),notec(6),notec(8),notec(10),notec(12)];
sound(m123_c,Fs)
```

# 音阶合成



## 两只老虎

(大众乐谱网制谱)

法国古歌曲调  
佚名填词

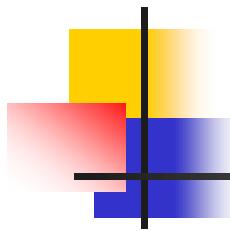
小快板



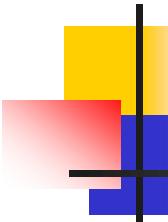
1=G 1 2 3 1 1 2 3 1 3 4 5 — 3 4 5 —  
两 只 老 虎， 两 只 老 虎， 跑 得 快， 跑 得 快。



5 6 5 4 3 1 5 6 5 4 3 1 1 5 · 1 — 1 5 · 1 —  
一 只 没 有 耳 朵， 一 只 没 有 尾 巴， 真 奇 怪！ 真 奇 怪！



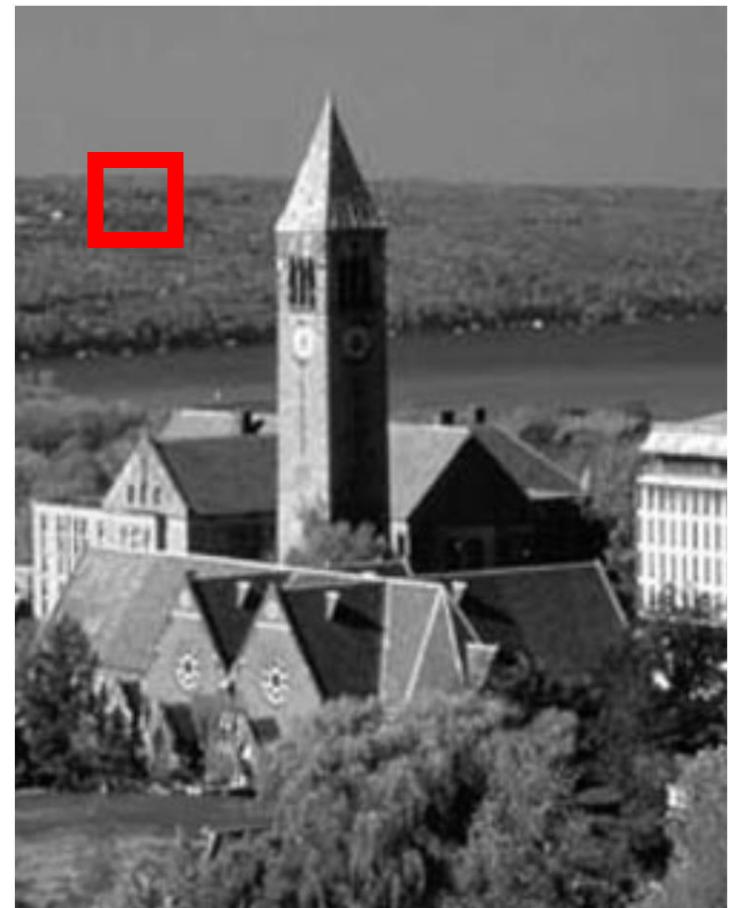
# 图像信号

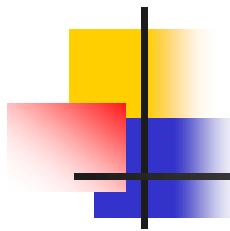


# 灰度图像

- 灰度图像  $\Leftrightarrow$  二维矩阵A  
 $0 \leq A(i,j) \leq 255$   
(黑) (白)

|     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|
| 49  | 55  | 58  | 59  | 57  | 53  |
| 60  | 67  | 71  | 72  | 72  | 70  |
| 102 | 108 | 111 | 111 | 112 | 112 |
| 157 | 167 | 169 | 167 | 165 | 164 |
| 196 | 205 | 208 | 207 | 205 | 205 |
| 199 | 208 | 212 | 214 | 213 | 216 |
| 190 | 192 | 193 | 195 | 195 | 197 |
| 174 | 169 | 165 | 163 | 162 | 161 |

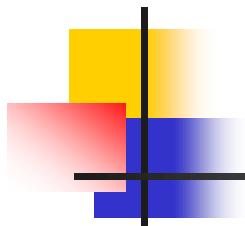




# 彩色图像(BMP格式)

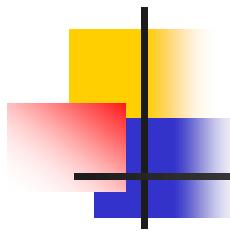
- 彩色图像 ( BMP格式)  $\leftrightarrow$  三维数组

|                              |            |
|------------------------------|------------|
| $0 \leq A(i, j, 1) \leq 255$ | 红色 (Red)   |
| $0 \leq A(i, j, 2) \leq 255$ | 绿色 (Green) |
| $0 \leq A(i, j, 3) \leq 255$ | 蓝色 (Blue)  |



Cornell University Law School  
Photograph by Cornell University Photography





## 其他图像格式

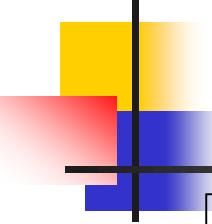
- JPEG (Joint Photographic Experts Group)
- GIF (Graphics Interchange Format)

基本目的：数据压缩

# 图像翻转

## ■ 图像左右翻转





# 图像翻转

```
A = imread('LawSchool.jpg');

[nr,nc,np] = size(A);

for r = 1:nr

    for c = 1:nc

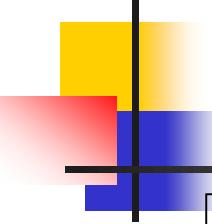
        for p = 1:np

            B(r,c,p) = A(r,nc -c+1);

        end

    end

end
```



# 图像翻转

```
A = imread('LawSchool.jpg');

[nr,nc,np] = size(A);

for r = 1:nr

    for c = 1:nc

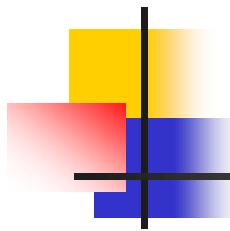
        for p = 1:np

            B(r,c,p) = A(r,nc -c+1);

        end

    end

end
```



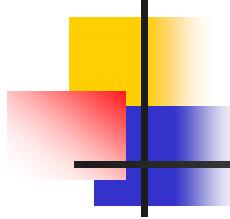
# 图像翻转

```
>> A = imread('LawSchool.jpg');
>> B(:,:,1) = fliplr(A(:,:,1));
>> B(:,:,2) = fliplr(A(:,:,2));
>> B(:,:,3) = fliplr(A(:,:,3));
>> imshow(B)
```

# 彩色图像->灰度图像



Cornell University Law School  
Photograph by Cornell University Photography

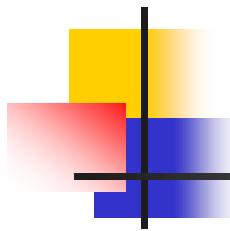


# 彩色图像->灰度图像

```
A =imread('lawschool.jpg');  
gA = (A(:,:,1)+A(:,:,2)+A(:,:,3))/3;  
imshow(gA);
```

# 彩色图像->灰度图像





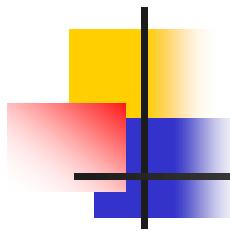
# 彩色图像->灰度图像

```
A =imread('lawschool.jpg');  
A = double(A);  
gA = (A(:,:,1)+A(:,:,2)+A(:,:,3))/3;  
imshow(uint8(gA));
```

# 彩色图像->灰度图像



Cornell University Law School  
Photograph by Cornell University Photography



# 彩色图像->灰度图像

- 由于人眼对红绿蓝三色敏感程度不同，常乘以不同比例因子。

```
A =imread('lawschool.jpg') ;
A = double(A) ;
gA = 0.3*A(:,:,1)+0.59*A(:,:,2)+0.11*A(:,:,3) ;
imshow(uint8(gA)) ;
```

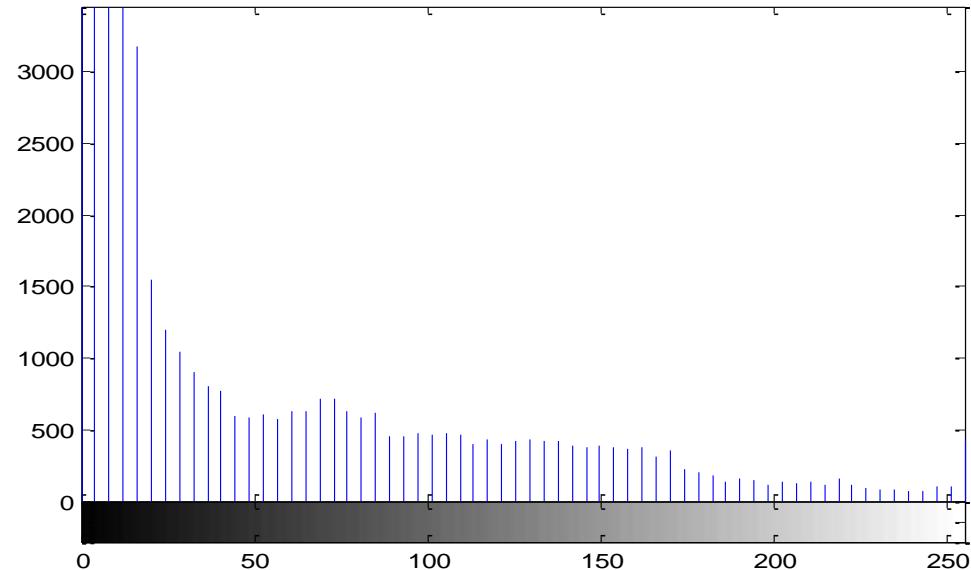
# 彩色图像->灰度图像

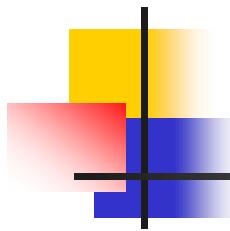


Cornell University Law School  
Photograph by Cornell University Photography

# 图像直方图

- 对于一张灰度图，该图的直方图就是占各个灰度值的像素点的个数的统计；
- 直方图是图像的一种统计特性

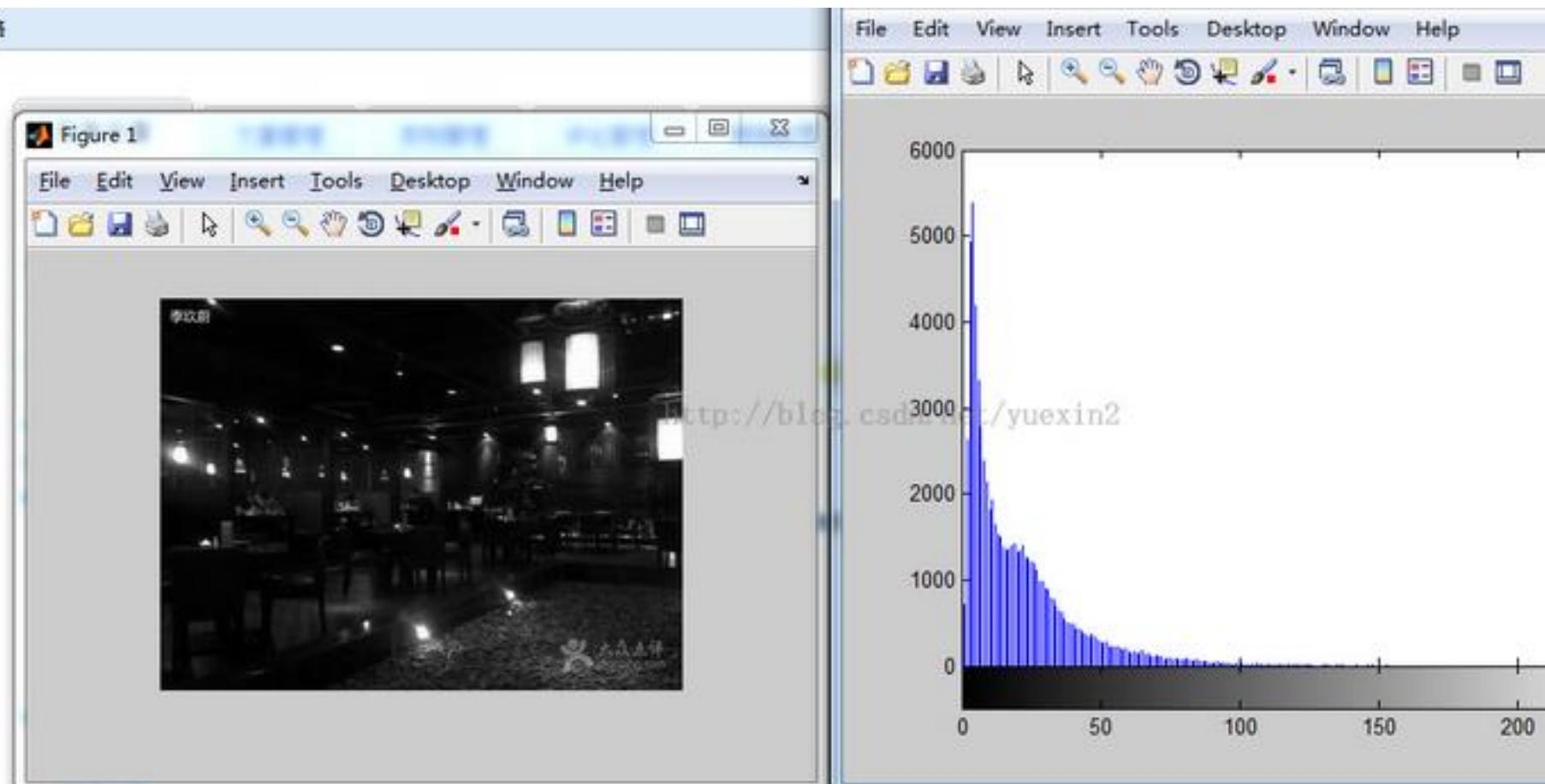




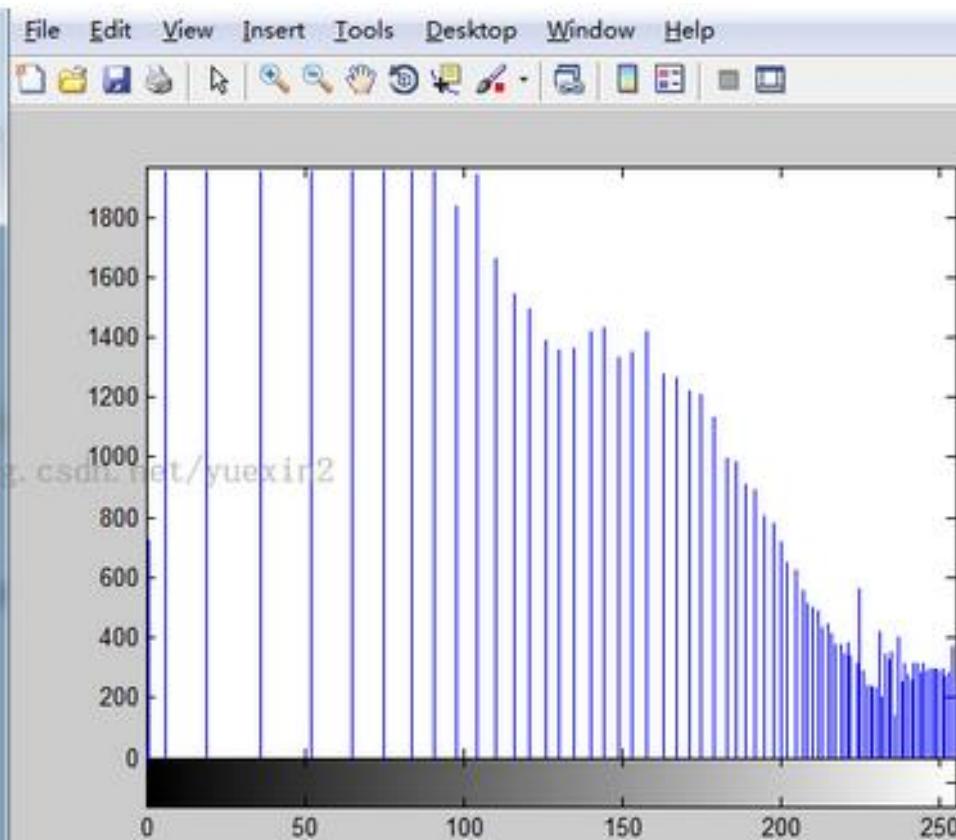
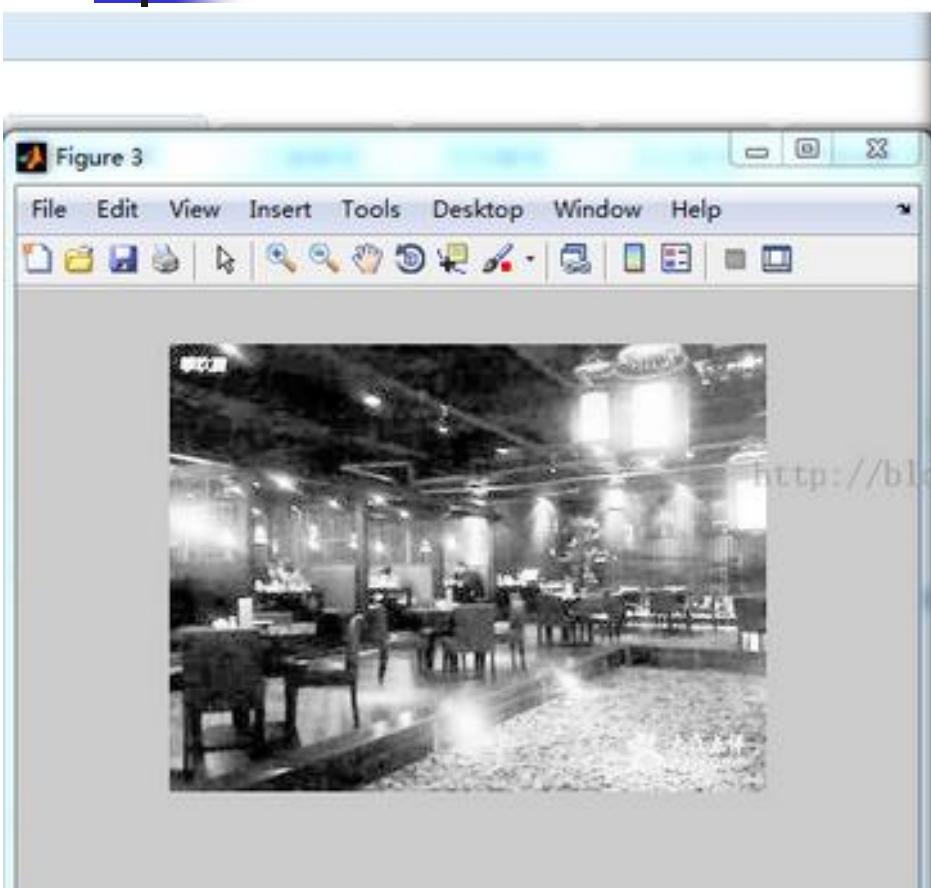
# 直方图均衡

- 直方图均衡化是通过拉伸像素强度分布范围来增强图像对比度的一种方法
- 均衡化指的是把一个分布 (给定的直方图) 映射 到另一个分布 (一个更宽更统一的强度值分布), 所以强度值分布会在整个范围内展开

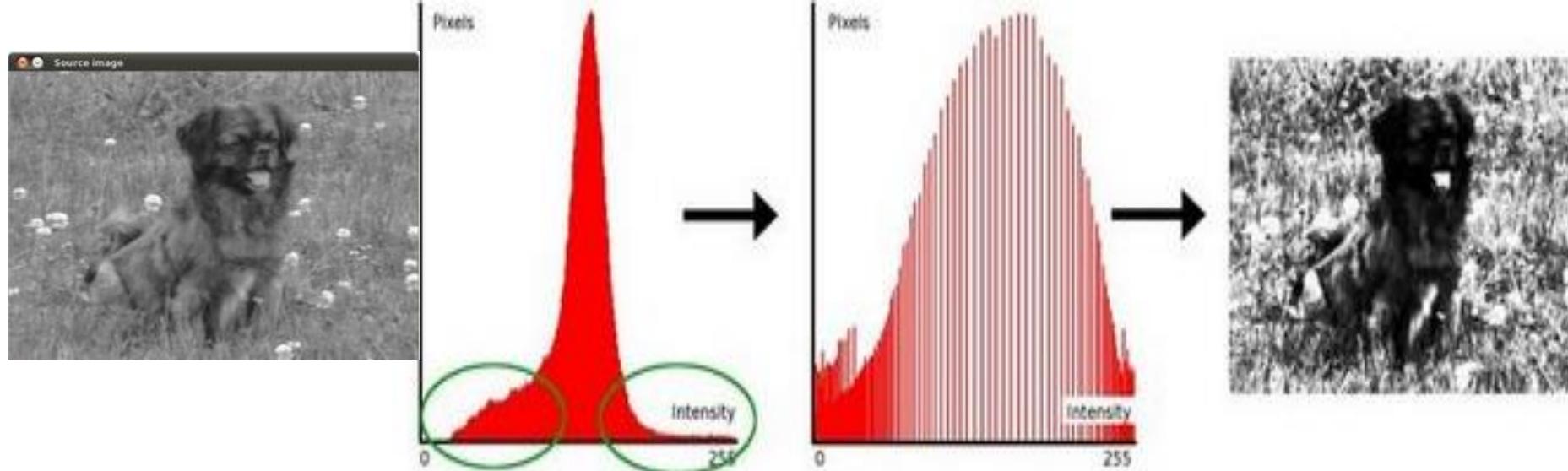
# 直方图均衡



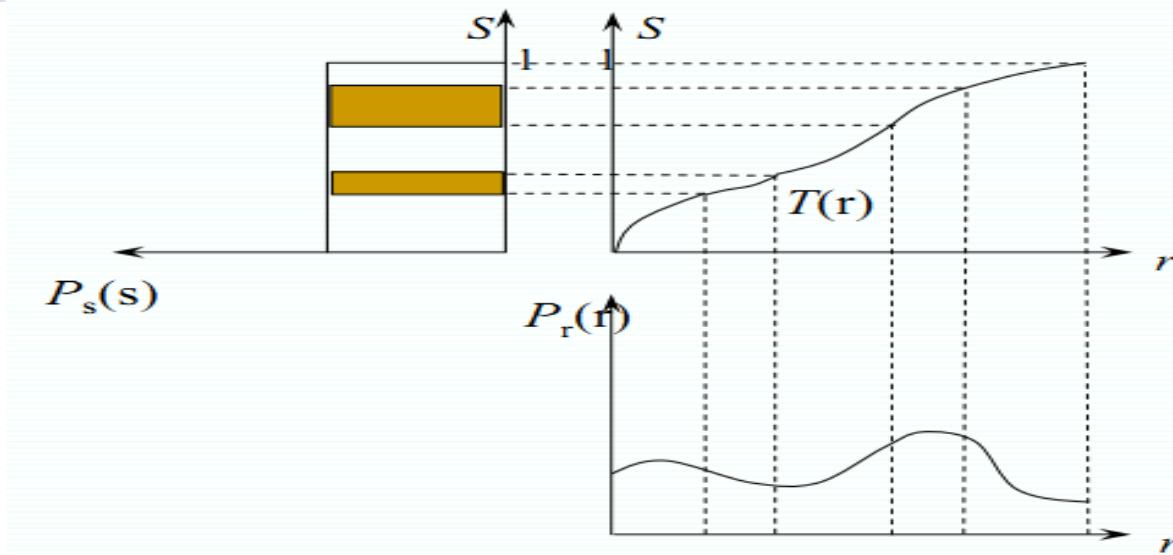
# 直方图均衡



# 直方图均衡

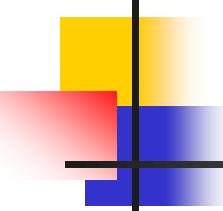


# 直方图均衡



对于  $s = T(r)$  假定：

- (1) 在  $0 \leq r \leq 1$  区间内， $T(r)$  为单调递增函数，且满足  $0 \leq T(r) \leq 1$
- (2) 变换  $r = T^{-1}(S)$  存在， $0 \leq S \leq 1$ ，也满足类似(1)的条件， $0 \leq r \leq 1$  有  $0 \leq T(r) \leq 1$



# 直方图均衡

对于连续的函数， $P_r(r)$ 和 $P_s(s)$ 分别是灰度r和s的概率密度函数，可知：

$$P_s(s) = P_r(r)dr/ds$$

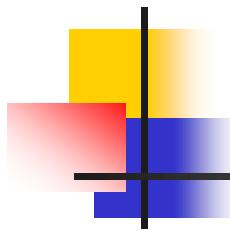
直方图均衡化的目的是保证每个灰度级的概率密度相等，即是一个常数：  
$$P_s(s) = 1/L$$

L是均衡化后灰度的变化范围，在这里归一化为1，即：

$$P_s(s)=1 \Rightarrow ds = P_r(r)dr \Rightarrow s = \int P_r(r)dr$$

此式表明，当变换函数为原图像密度函数的分布函数时，能达到直方图均衡化目的。

对于离散的情况有： $s_k = \sum_{j=0}^k n_j/n$  \*

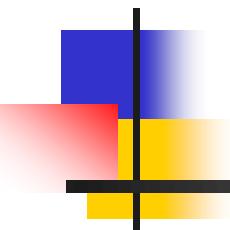


# 直方图均衡

- $h(r_k) = n_k$
- 灰度级为 $r_k$ 发生的概率估计值

$$p(r_k) = n_k/n \quad k = 0, 1, \dots, 256$$

- 希望新的灰度级分布:  $p_s(s) = 1/256$
- 映射函数  $s = T(r)$

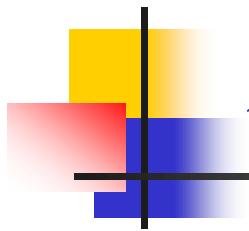


# Matlab编程与应用

## 第五讲

中国科技大学信息学院  
陆伟

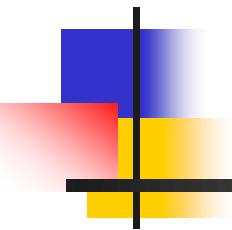
[luwei@ustc.edu.cn](mailto:luwei@ustc.edu.cn)



# 本讲内容

---

- part1: 基于GUIDE的GUI设计
- Part2: GUI的编程实现



# part1

---

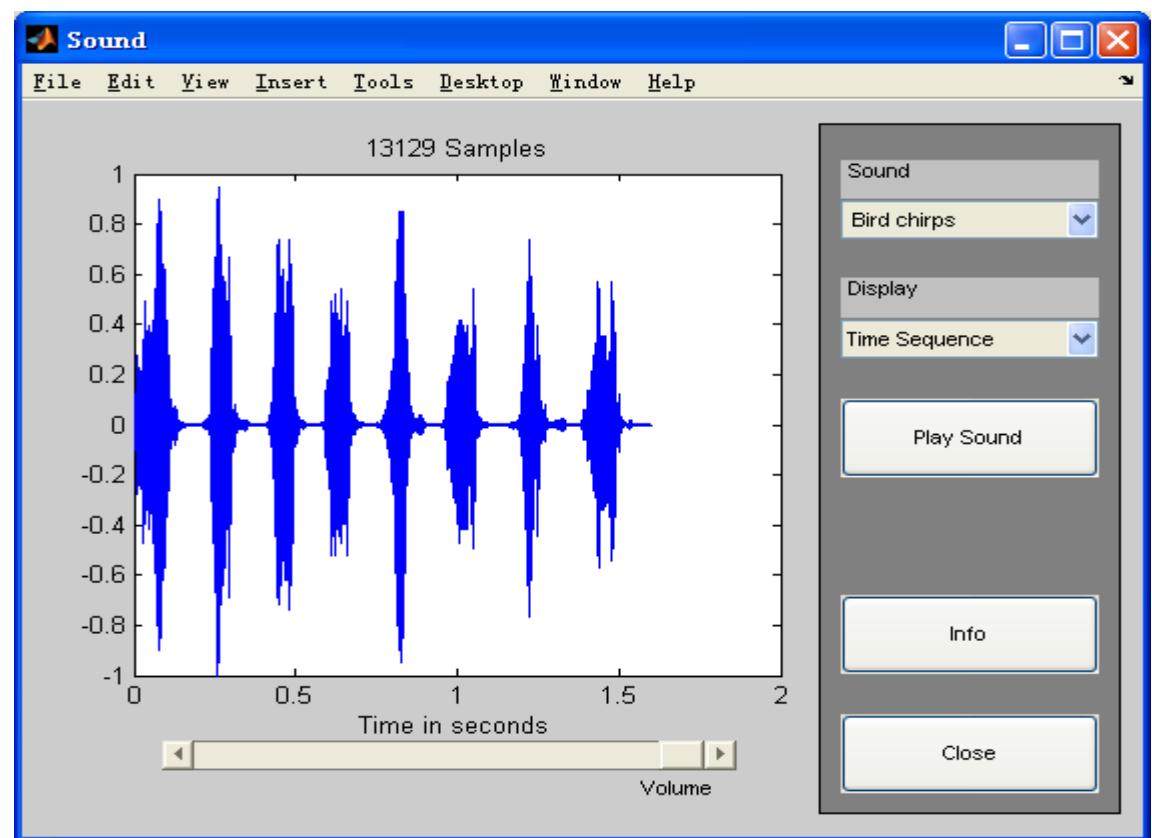
## GUI程序设计(一) ——GUIDE实现

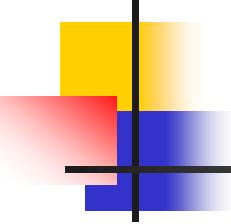
# GUI介绍

## ■ GUI: (Graphical User Interfaces) 图形用户界面

例: matlab的一个demo

xpsound

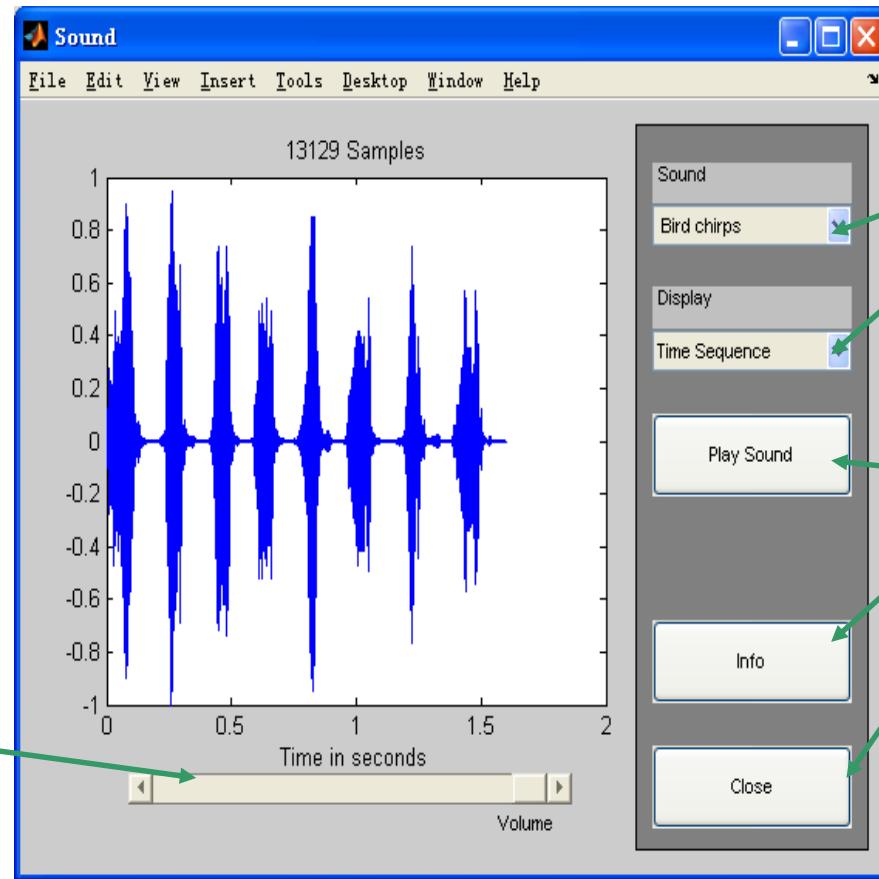




# GUI介绍

- 主要利用各种控件 (**uicontrols**)与用户交互： 按钮、 滑动条、 列表框...
- 每个控件也是图形对象， 也拥有属性与回调函数
- 改变控件的属性可以调整控件的外观。利用回调函数可以对用户的操作做出相应的响应。 (事件驱动)

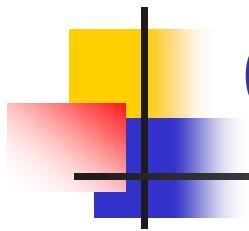
# GUI介绍



滑动条  
slider

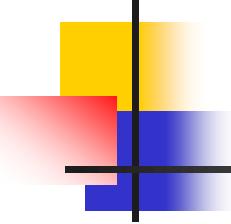
弹出式菜单  
pop-up menu

按钮  
push button



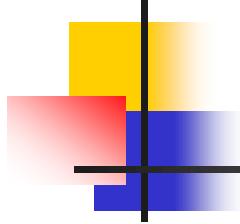
# GUI介绍

- Matlab中创建GUI的三种方式：
  1. 编程实现
  2. 利用GUIDE工具
  3. 利用App Designer 工具



# GUIDE

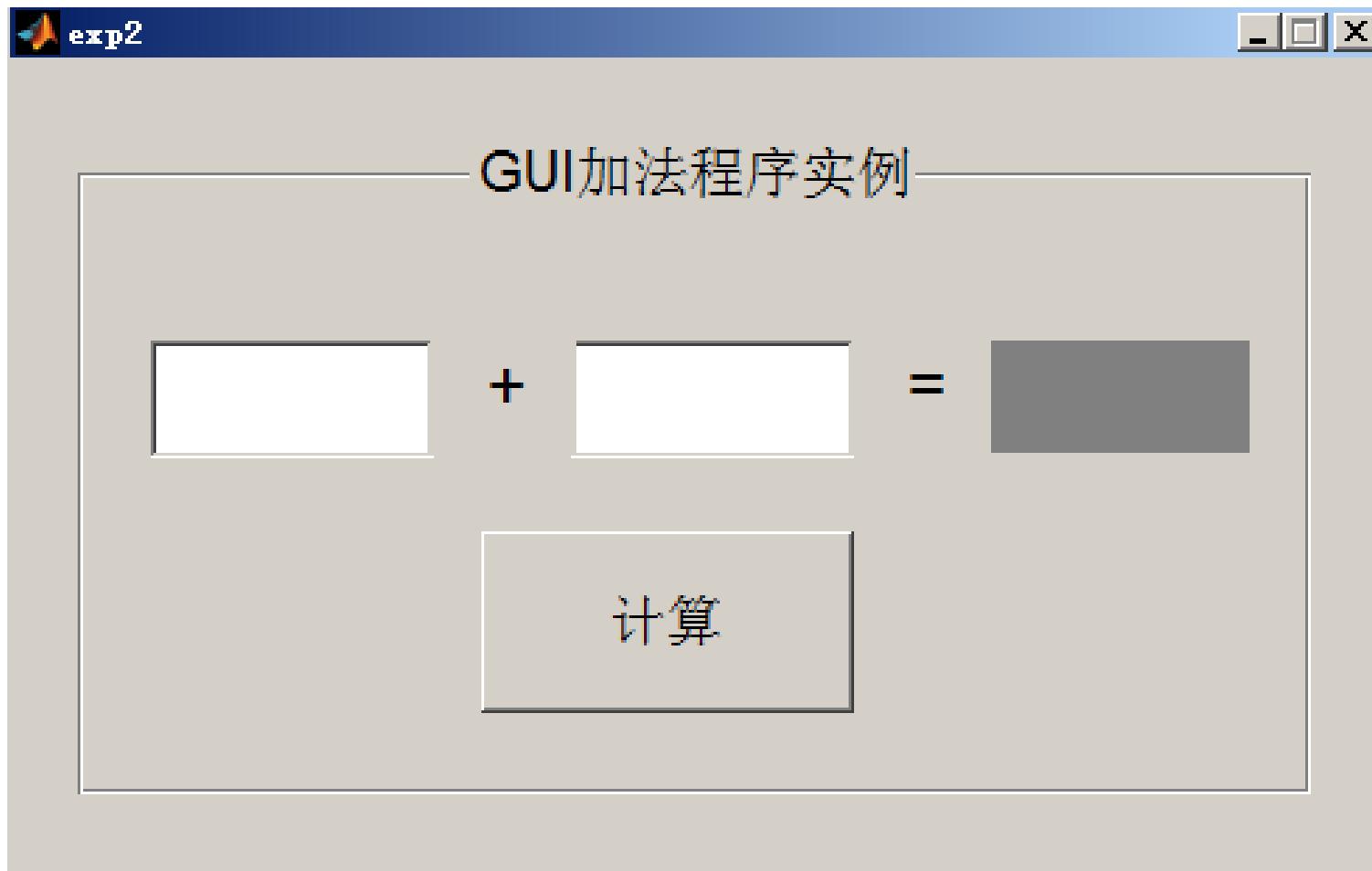
- **Graphic User Interface Designer Environment**
- **GUIDE**生成两个文件：
  - .fig文件**：保存**GUI**的**layout**,就是展现给用户的界面，界面含有**axes,button,listbox**等**控件**。
  - .m文件**：包含对各种事件的响应，即**回调函数**。



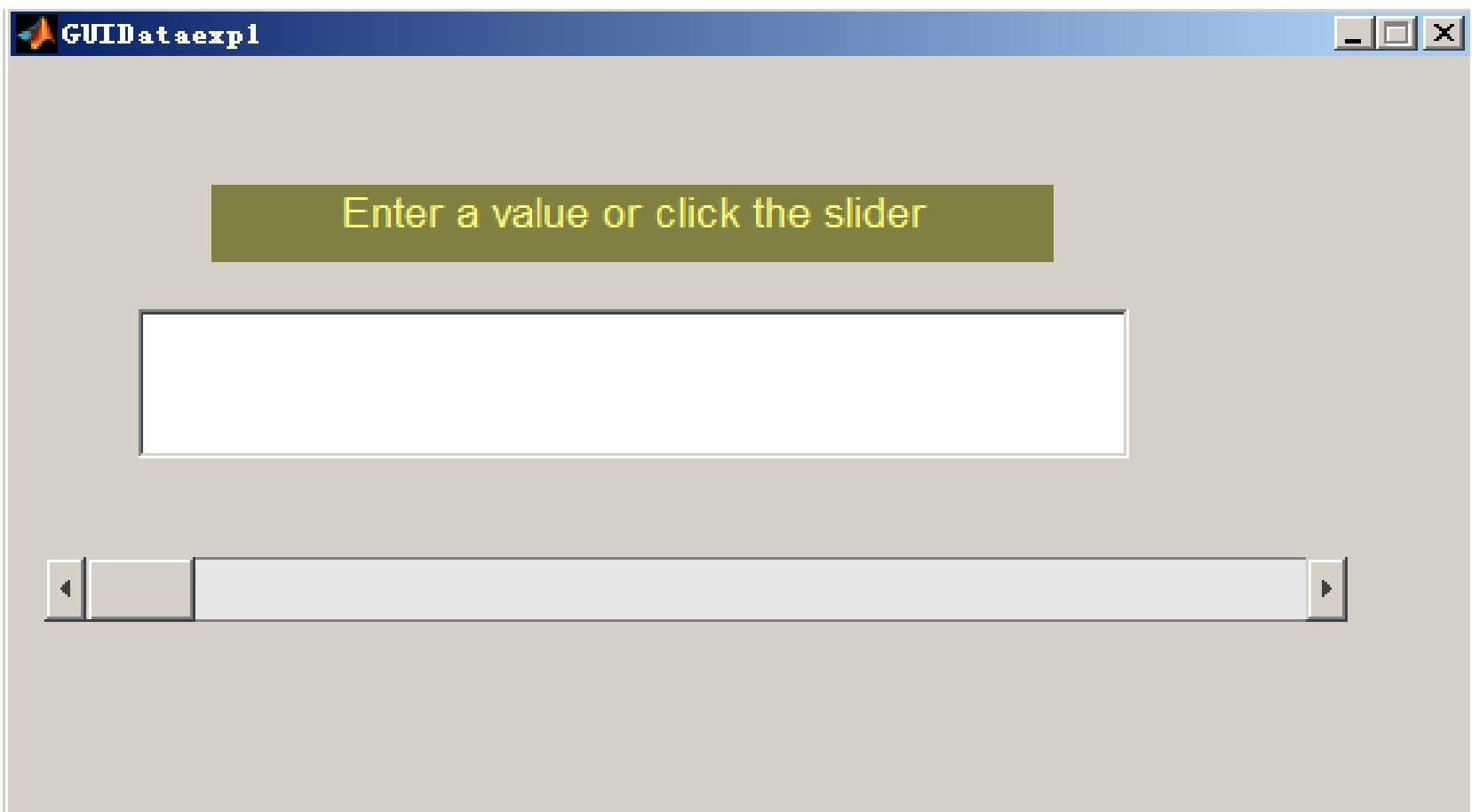
# GUIDE

- 主要设计过程
  - 1. 构思整个GUI的布局与设计任务；
  - 2. 利用**GUIDE**进行界面设计；
  - 3. 设置界面上各个控件的属性；
  - 4. 编写回调函数代码。

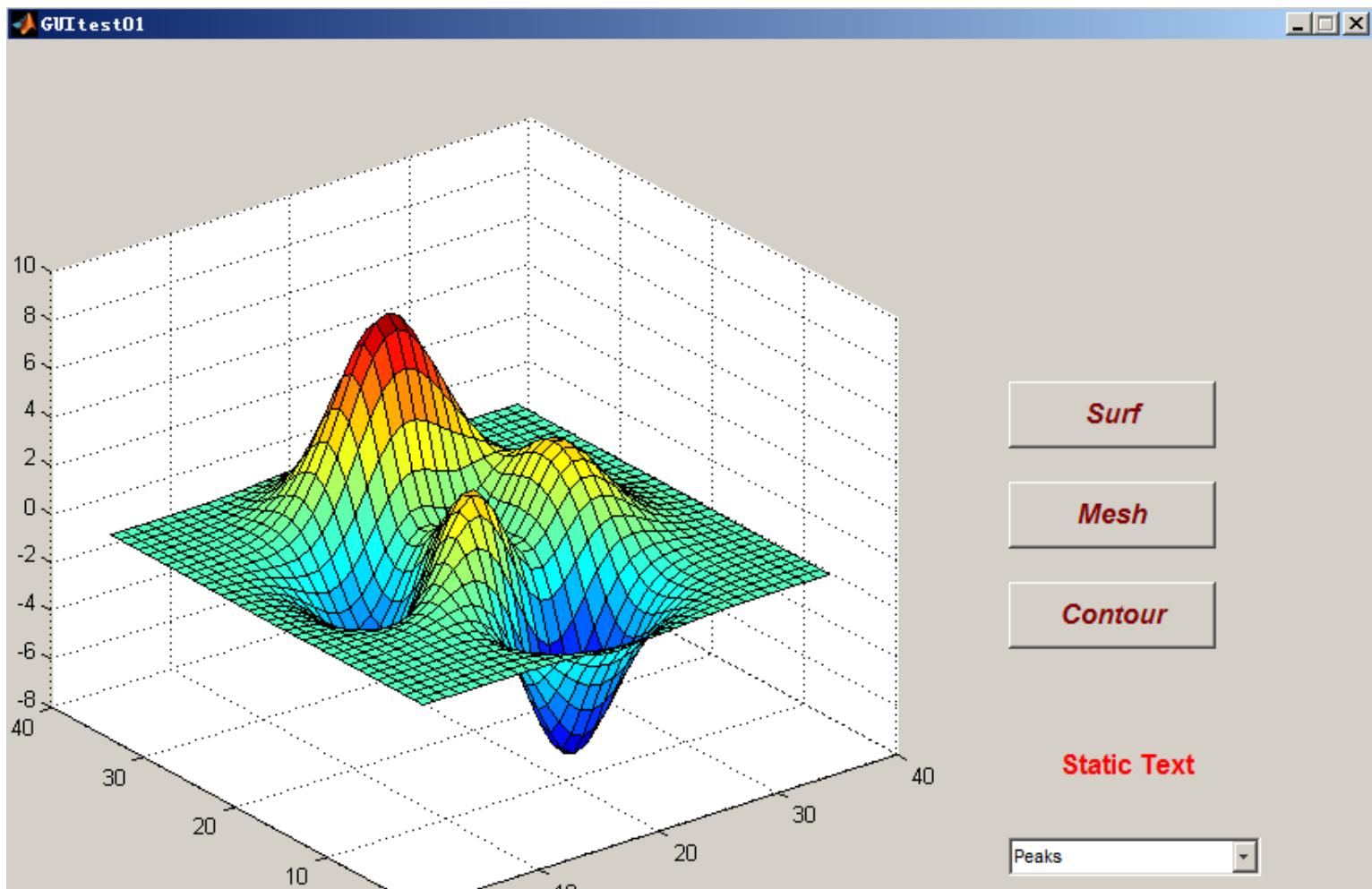
# GUIDE例1



# GUIDE例2

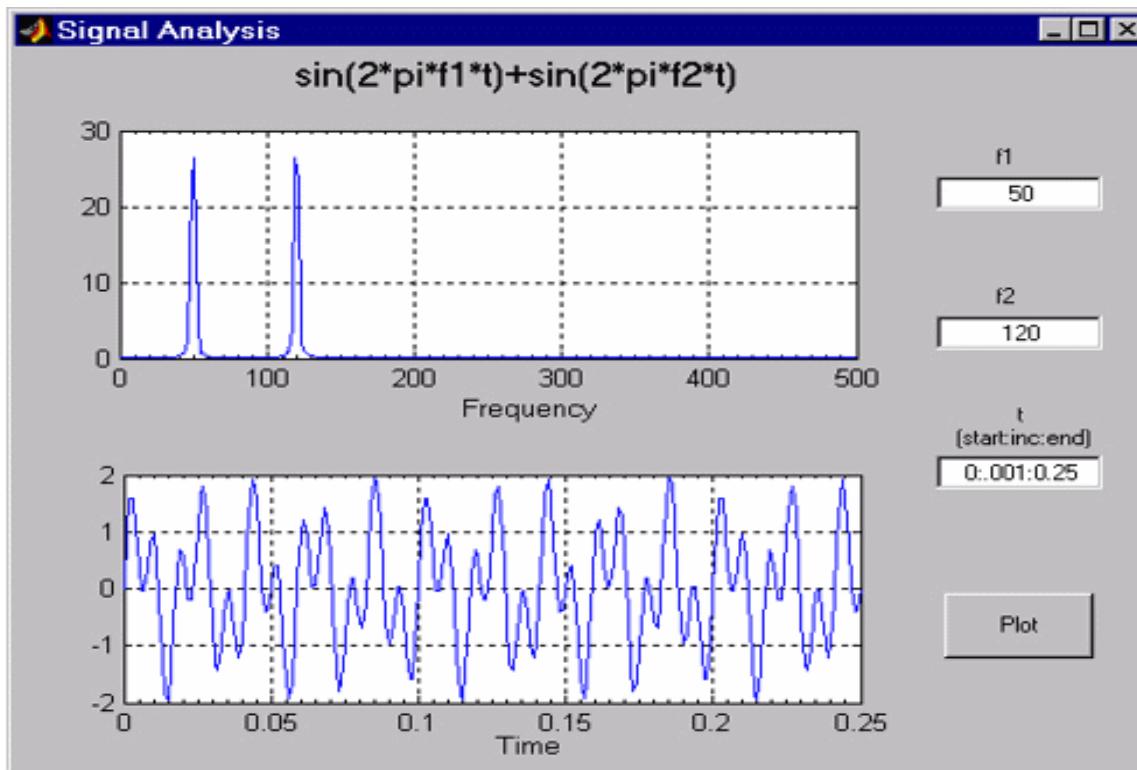


# GUIDE例3



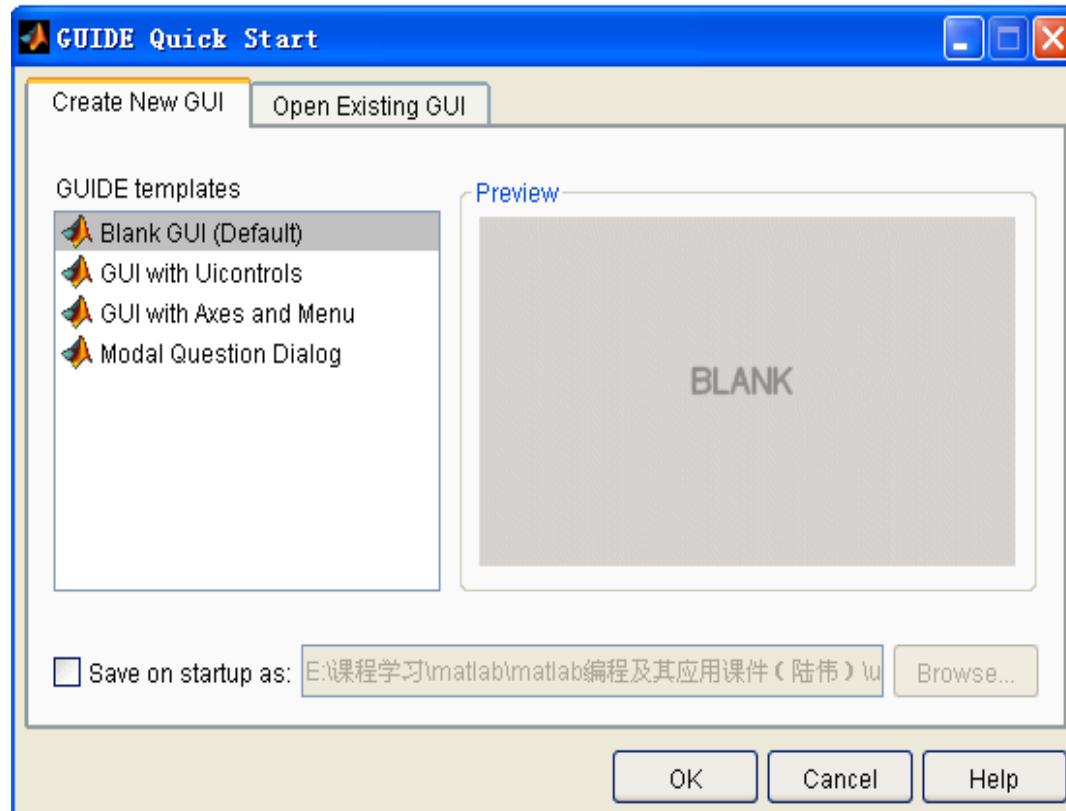
# GUIDE例4

例：设计一个GUI，生成一个由两个不同频率正弦波相加的信号，显示其时域波形与频谱，频率值和时间可由用户输入。



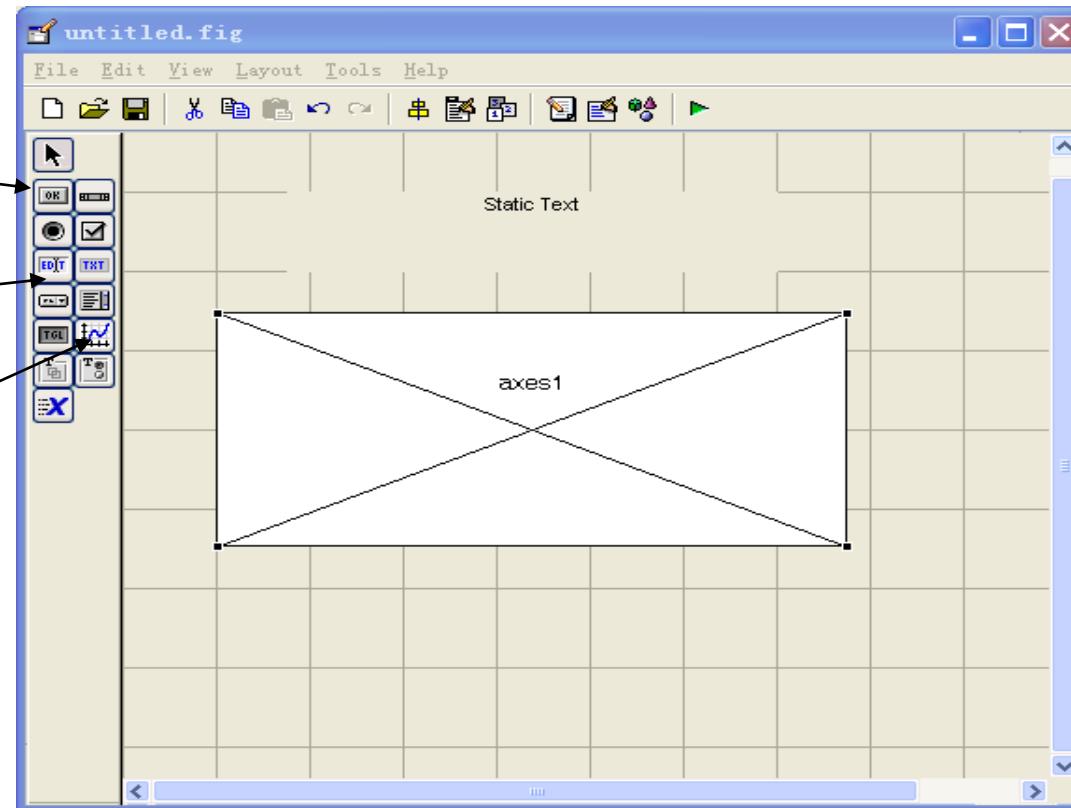
# GUIDE

1. 新建一个空白的**GUI**并保存。



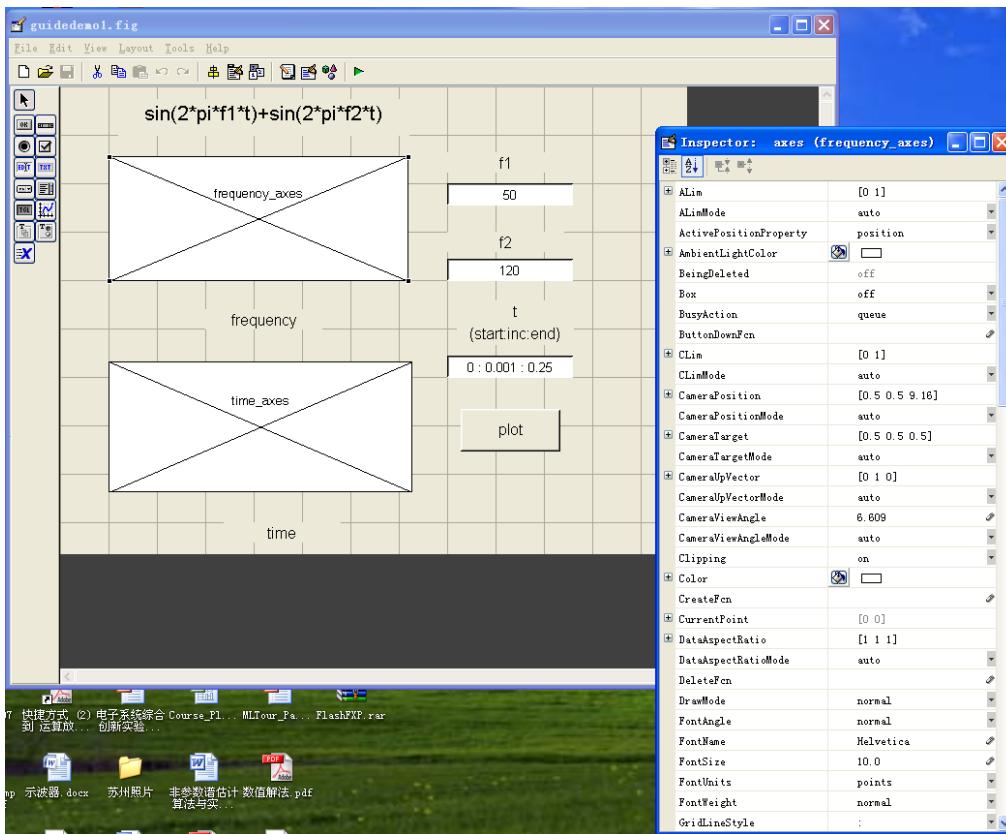
# GUIDE

2. 按照预先设计，在界面合适位置放置控件。



# GUIDE

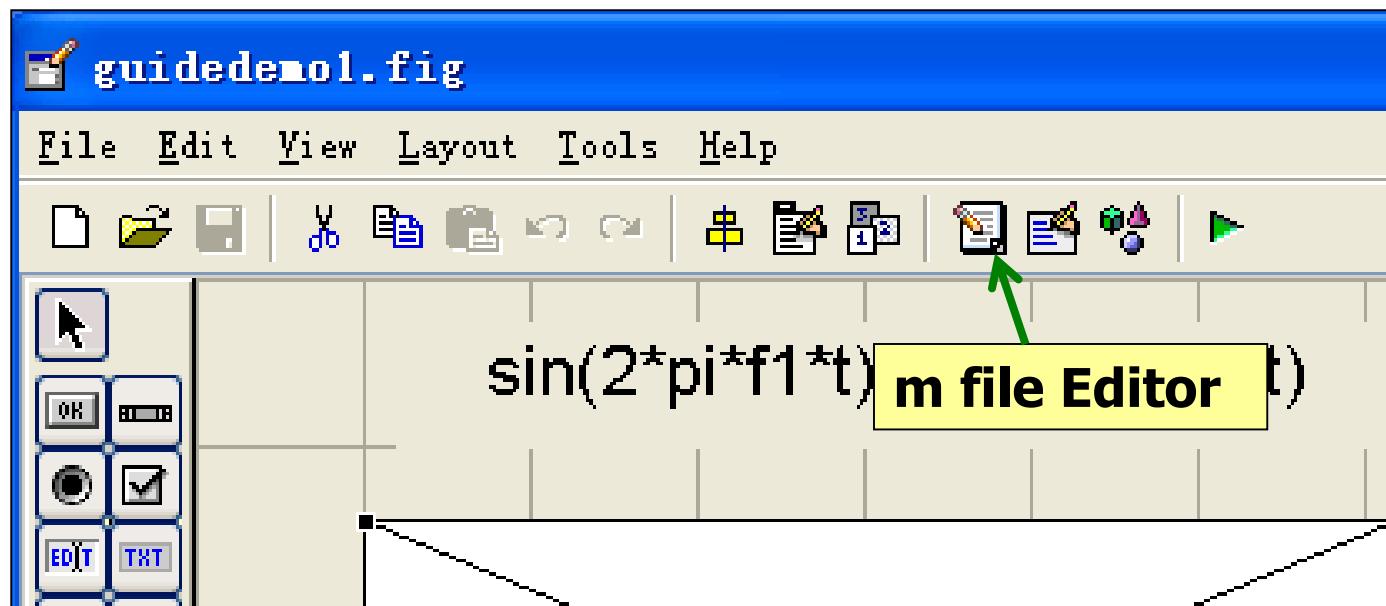
## 3. 双击某控件，设置控件的一些属性。

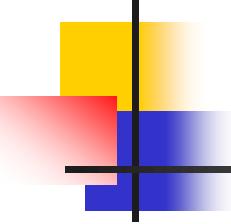


- 要恰当定义每个控件的‘tag’属性，便于标识，如‘time\_axes’，‘freq\_axes’。
- 每个控件‘tag’属性不能相同。因为m文件的回调函数利用该属性找到相应的控件。

# GUIDE

## 4. 打开对应的.m文件编写响应的回调函数





# GUIDE

- m文件中有很多子函数，大部分函数不需要添加代码，但也**不能随意删除**！
- 回调函数**三个参数的含义**：

```
% --- Executes on button press in pushbutton1.  
function pushbutton1_Callback(hObject, eventdata, handles)  
% hObject    handle to pushbutton1 (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)  
% Get user input from GUI
```

# GUIDE

function p  
handles)  
% hObject  
(hObject即

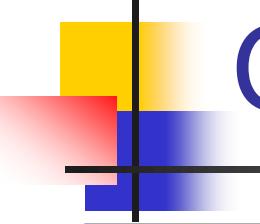
% eventdata  
% handles  
件的句柄，  
字段值为记

The screenshot shows the MATLAB Editor window with the following code:

```
Editor - E:\课程学习\matlab\matlab编程及其应用课件(陆伟)\图形句柄与GUI编程\... File Edit Text Go Cell Tools Debug Desktop Window Help pushbutton1... pushbutton1_Callback(hObject, eventdata, handles)
```

The code defines a function handle for pushbutton1\_Callback. The handles structure contains the following fields and values:

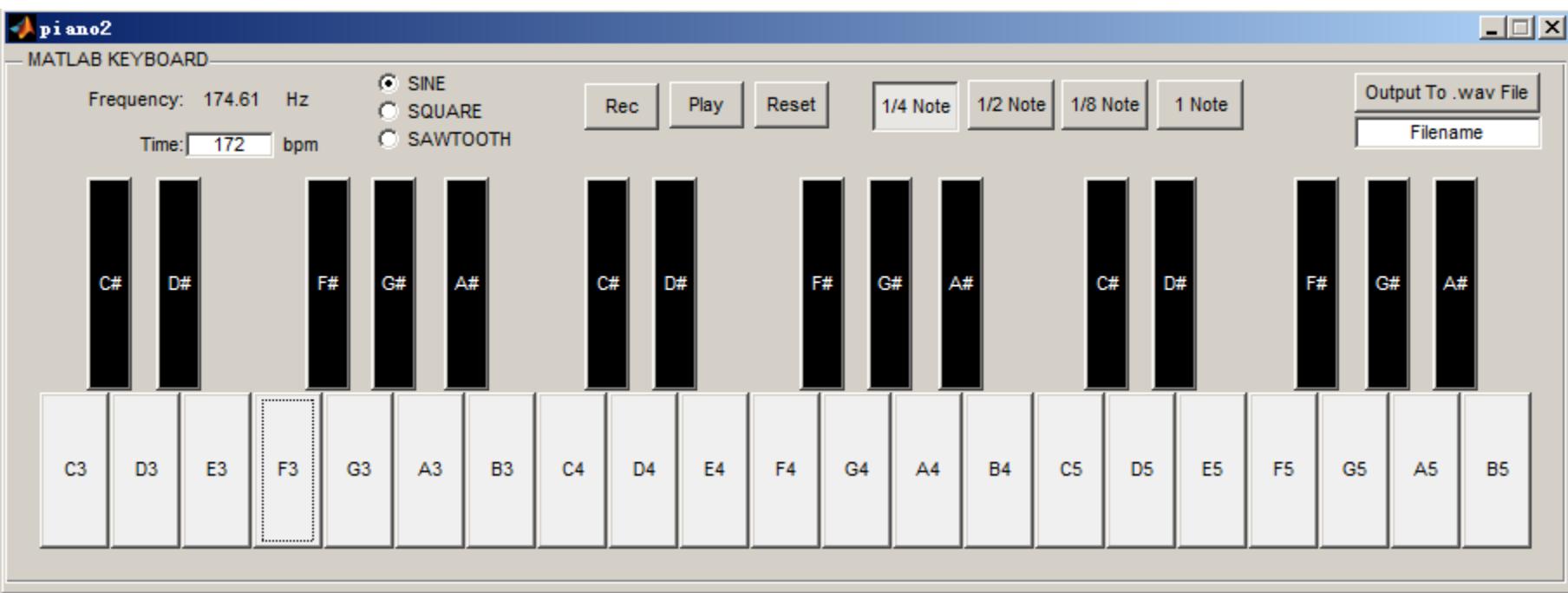
|                       |          |
|-----------------------|----------|
| handles: 1x1 struct = |          |
| figure1:              | 186.0018 |
| text6:                | 23.0024  |
| text5:                | 22.0024  |
| pushbutton1:          | 21.0024  |
| t_input:              | 20.0027  |
| text4:                | 19.0106  |
| f2_input:             | 18.0106  |
| text3:                | 17.0029  |
| f1_input:             | 16.0029  |
| text2:                | 15.0031  |
| text1:                | 197.0016 |
| time_axes:            | 192.0016 |
| frequency_axes:       | 187.0018 |
| output:               | 186.0018 |

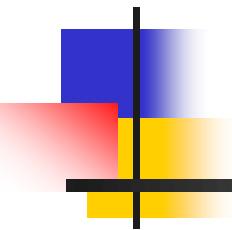


# GUIDE

```
function pushbutton1_Callback(hObject, eventdata, handles)
f1 = str2double(get(handles.f1_input,'String'));
% handles.f1_input: tag属性为f1_input的编辑框的句柄
% get(handles.f1_input,'String')取出该编辑框中
    string%属性的字符串
% str2double(...), 将该字符串转化为double.
f2 = str2double(get(handles.f2_input,'String'));
t = eval(get(handles.t_input,'String'));
% get(handles.t_input,'String')取出的字符串为
0:0.001:0.25, 无法用 str2double函数转换为双精度数
数组, 用eval函数实现
.....
```

# GUIDE例5

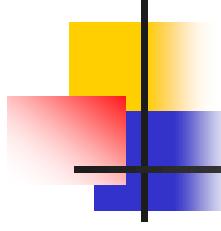




# part2

---

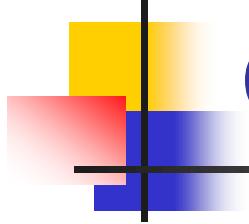
## GUI程序设计(二) ——编程实现



# GUI编程实现

## ■ 例：

```
%创建一个按钮
hp1 = uicontrol('style','pushbutton');
%修改按钮属性
set(hp1,'position',[100,100,200,200])
set(hp1,'string','按一下');
set(hp1,'fontsize',24)
%定义回调函数
cmd = 'disp('''你按了我一下''')';
set(hp1,'callback',cmd)
```



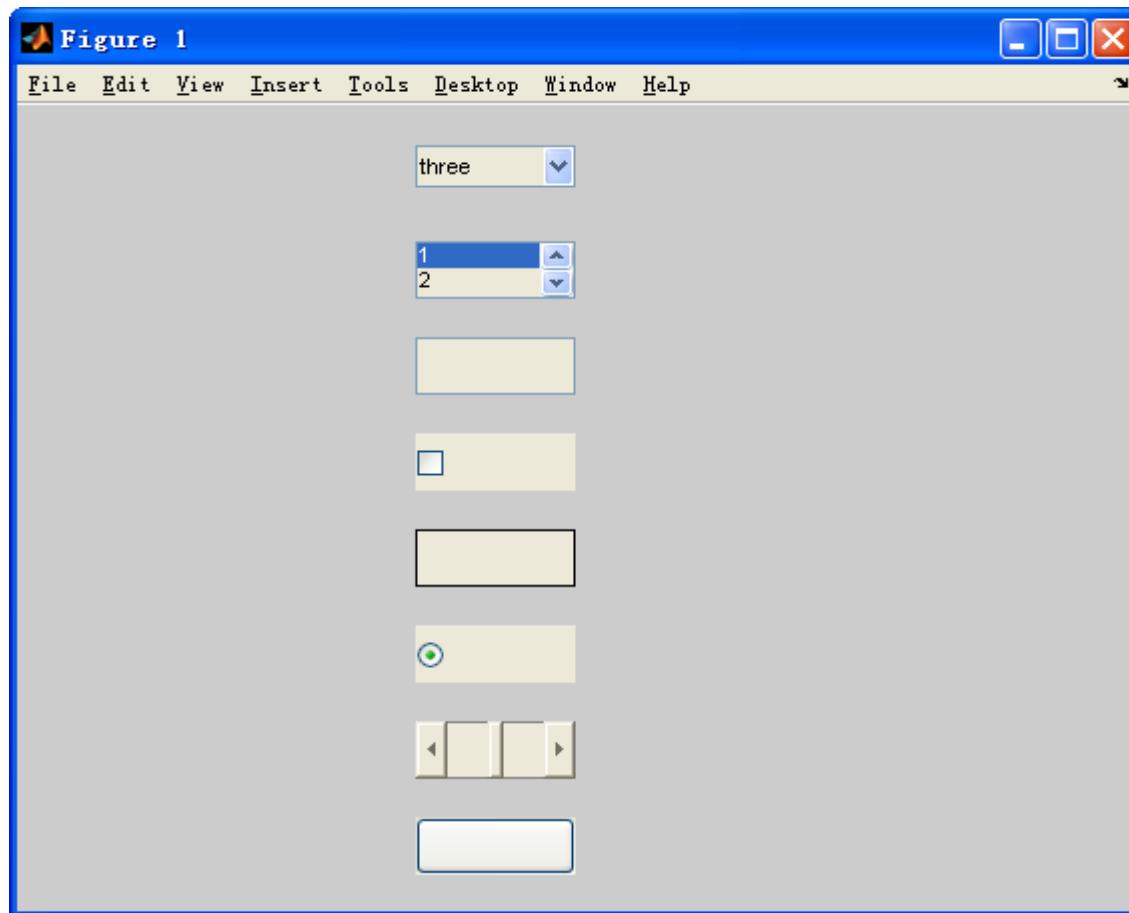
# GUI编程实现

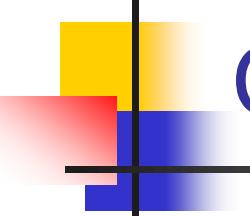
- 添加一个关闭按钮

```
hp2 = uicontrol('style','pushbutton');
set(hp1,'position',[320,100,200,200])
set(hp1,'string','关闭');
set(hp1,'fontsize',24)
set(hp1,'callback','close')
```

# GUI编程实现

## ■ matlab的一些控件

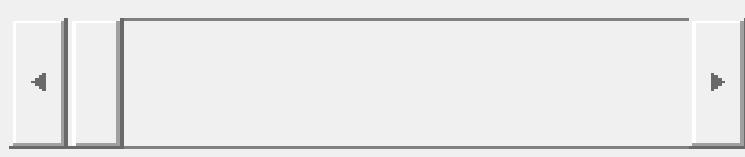




# GUI编程实现

```
close all  
uicontrol('style','push','position',[200 20 80 30]);  
uicontrol('style','slide','position',[200 70 80 30]);  
uicontrol('style','radio','position',[200 120 80 30]);  
uicontrol('style','frame','position',[200 170 80 30]);  
uicontrol('style','check','position',[200 220 80 30]);  
uicontrol('style','edit','position',[200 270 80 30]);  
uicontrol('style','list','position',[200 320 80  
30],'string', '1|2|3|4');  
uicontrol('style','popup','position',[200 370 80  
30],'string','one|two|three');
```

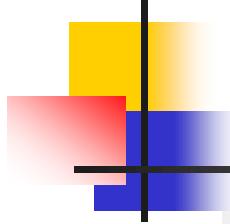
# Slider控件



```
hp1 =  
uicontrol('style','slider');  
  
set(hp1,'position',[100,10  
0,200,50])  
  
set(hp1,'max',100,'min',0,'  
value',25);
```

- **slider** 控件的一些属性：

|            |              |
|------------|--------------|
| Max        | 1.0          |
| Min        | 0.0          |
| SliderStep | [1x2 double] |
| Style      | slider       |
| Value      | 0.45         |



# Popupmenu控件



- ```
h1 = uicontrol('style','popup', ...
    'position',[200 370 180 60],...
    'string','one|two|three');
```

```
str1 = get(h1,'str')
```

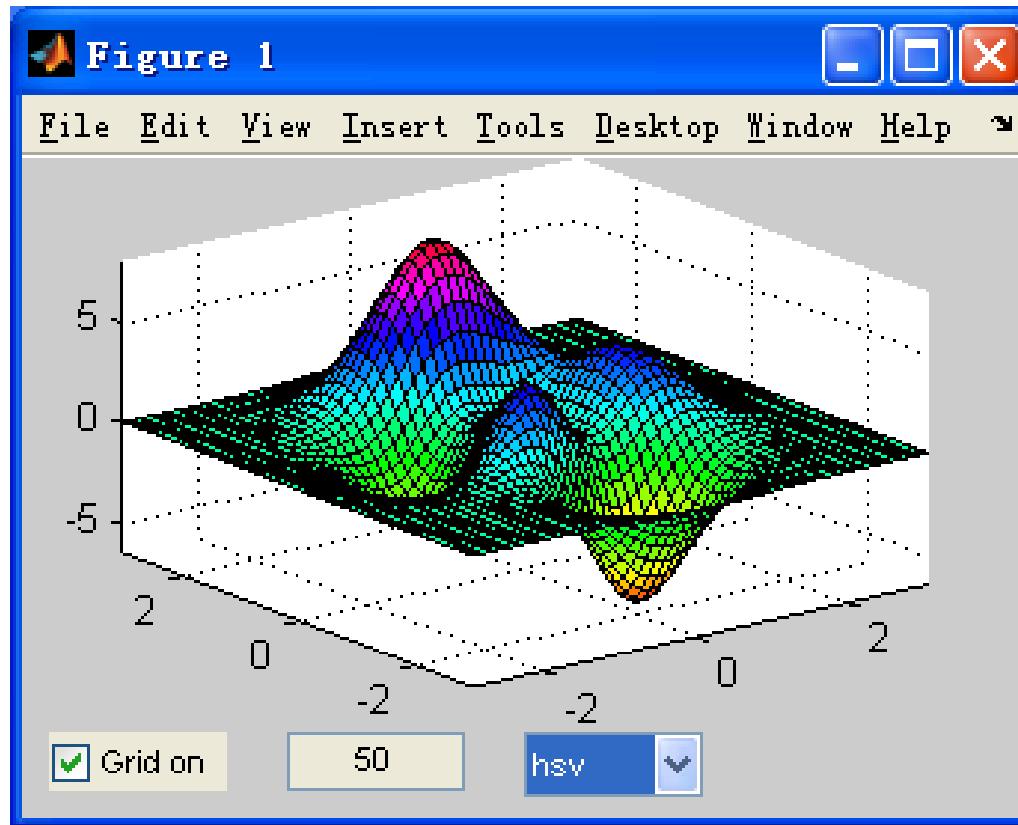
```
ind = get(h1,'value')
```

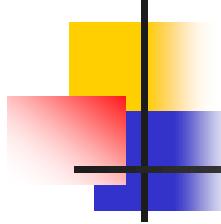
  

```
str1(ind,:)
```

# GUI编程实现

- 例： ui01.m





# GUI编程实现

## ui01.m 代码

```
figure('position', [30 30 300 200]);
axes('position', [0.1 0.2 0.8 0.8]);
pointNum = 20;
[xx, yy, zz] = peaks(pointNum);
surf(xx, yy, zz);
axis tight
```

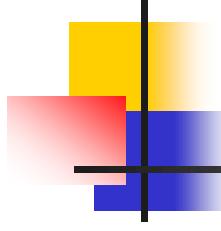
...后面还有

# GUI编程实现

## ■ ui01.m 代码

```
h1 = uicontrol('style', 'checkbox', 'string', 'Grid  
on/off', ...  
    'position', [10, 10, 80, 20], 'value', 1);  
h2 = uicontrol('style', 'edit', 'string',  
int2str(pointNum), ...  
    'position', [100, 10, 60, 20]);  
h3 = uicontrol('style', 'popupmenu', ...  
    'string', 'hsv|hot|cool', ...  
    'position', [180, 10, 60, 20]);  
  
set(h1, 'callback', 'grid');  
set(h2, 'callback', 'cb2');  
set(h3, 'callback', 'cb3');
```

...后面还有

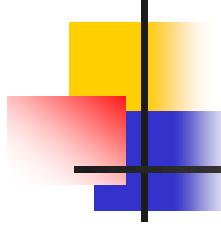


# GUI编程实现

- ui01.m 中文本编辑框控件的回调函数 cb2.m

```
pointNum = round(str2num(get(h2, 'string'))));
if pointNum <= 1 | pointNum > 100,
    pointNum = 10;
    set(h2, 'string', int2str(pointNum));
end
[xx, yy, zz] = peaks(pointNum);
surf(xx, yy, zz);
axis tight;

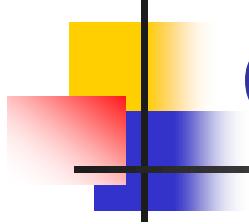
if get(h1, 'value')==1,
    grid on;
else
    grid off;
end
```



# GUI编程实现

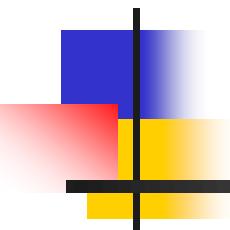
- ui01.m中弹出式菜单的回调函数cb3.m

```
switch get(h3, 'value')
    case 1
        colormap(hsv);
    case 2
        colormap(hot);
    case 3
        colormap(cool);
    otherwise
        disp('Unknown option');
end
```



# GUI编程实现

- **ui01.m**的潜在问题：
  1. 需要三个m文件，管理不方便
  2. 使用的变量都在**matlab**基本工作空间中，容易造成变量冲突与覆盖。
- 改进方法：使用**Switch case**结构的程序设计方法  
例：**ui02.m**

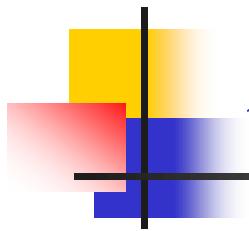


# Matlab编程与应用

## 第六讲

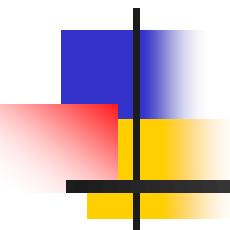
中国科技大学信息学院  
陆伟

[luwei@ustc.edu.cn](mailto:luwei@ustc.edu.cn)



# 本讲内容

- part1: Matlab多项式运算
- part2: 微分方程数值解法
- part3: 符号计算



## Part1:

---

# Matlab多项式运算

# Matlab中多项式表示方法：

n 阶多项式用长度为 n+1 的向量表示，  
缺少的幂次项系数为 0 !

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Matlab 中表示为向量：  $[a_n, a_{n-1}, \dots, a_1, a_0]$

例： $2x^3 - x^2 + 3 \longrightarrow [2 \ -1 \ 0 \ 3]$



系数中的0不能省！

■ 多项式显示： $\text{poly2sym}(p)$

或者  $\text{poly2str}(p, 'x')$

# 多项式加减运算

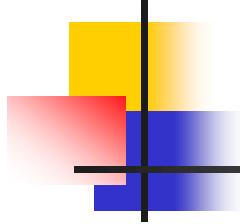
多项式加减就是其所对应的系数向量的加减运算

- 对于次数相同的多项式，可以直接对其系数向量进行加减运算；
- 如果两个多项式次数不同，则应该把低次多项式中系数不足的高次项用 0 补足，然后进行加减运算。

例：  $p_1 = 2x^3 - x^2 + 3 \longrightarrow [2, -1, 0, 3]$

$p_2 = 2x + 1 \longrightarrow [0, 0, 2, 1]$

$p_1 + p_2 = 2x^3 - x^2 + 2x + 4 \longleftarrow [2, -1, 2, 4]$



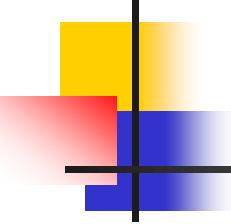
# 多项式加减运算

- 练习：编一个自动补零的多项式加减函数

$p = \text{mypolyadd}(a, b)$

输入：  $a, b$  为待相加多项式对应的向量（可以是列向量）

输出： 相加后多项式对应的向量。



## 多项式乘法、除法运算

□ 多项式乘法运算:  **$k = \text{conv}(p, q)$**

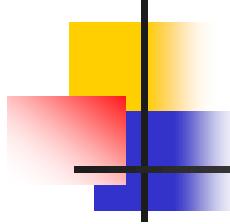
例: 计算多项式  $2x^3 - x^2 + 3$  和  $2x + 1$  的乘积

```
>> p=[2,-1,0,3];
>> q=[2,1];
>> k=conv(p,q);
```

□ 多项式除法运算:  **$[k, r] = \text{deconv}(p, q)$**

其中  **$k$**  返回的是多项式  **$p$**  除以  **$q$**  的商,  **$r$**  是余式。

$$[k, r] = \text{deconv}(p, q) \iff p = \text{conv}(q, k) + r$$



# 多项式求导

## □ polyder

**k=polyder(p)** : 多项式 p 的导数;

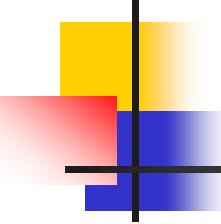
**k=polyder(p,q)** : p\*q 的导数;

**[k,d]=polyder(p,q)** : p/q 的导数, k 是分子, d 是分母

例: 已知  $p(x) = 2x^3 - x^2 + 3, \quad q(x) = 2x + 1,$

求  $p', (p \cdot q)', (p/q)'$

```
>> k1=polyder([2,-1,0,3]);  
>> k2=polyder([2,-1,0,3],[2,1]);  
>> [k2,d]=polyder([2,-1,0,3],[2,1]);
```



# 多项式积分

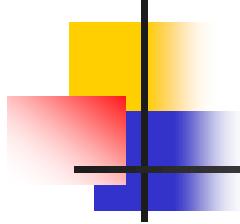
## □ polyint

**k=polyint(p,a)** : 多项式 p 的积分，积分后常数项为a;

**k=polyint(p)** :多项式 p 的积分，积分后常数项为0;

例：先对  $p(x) = 2x^3 - x^2 + 3$  求导，再积分。

```
>> p = polyder([2,-1,0,3]);  
>> k1=polyint(p);  
>> k2=polyder(p,3);
```



## 多项式求根

### □ **r = roots(p)**

例：求多项式  $p(x) = 2x^3 - x^2 + 3$  的根

```
>> p = [2 -1 0 3];  
>> r = roots(p)
```

### □ 已知多项式的根，构建对应的多项式

```
>> p1 = poly(r);
```

# 多项式求值

□ 计算多项式在给定点的值

◆ 代数多项式求值

**y = polyval (p, x) :** 计算多项式 p 在 x 点的值

**注：若 x 是向量或矩阵，则采用数组运算（点运算）！**

**例：**已知  $p(x) = 2x^3 - x^2 + 3$ ， 分别取  $x=2$  和一个  $2 \times 2$  矩阵，  
求  $p(x)$  在  $x$  处的值

```
>> p=[2, -1, 0, 3];  
>> x=2; y=polyval(p, x)  
>> x=[-1, 2; -2, 1]; y=polyval(p, x)
```

# 矩阵多项式求值

```
Y=polyvalm(p,X) % X 必须是方阵
```

例：已知  $p(x) = 2x^3 - x^2 + 3$ , 则

```
polyvalm(p,A)=2*A*A*A - A*A + 3*eye(size(A))  
polyval(P,A)=2*A.*A.*A - A.*A + 3*ones(size(A))
```

```
>> p=[2,-1,0,3];  
>> x=[-1, 2;-2,1];polyval(p,x)  
>> polyvalm(p,x)
```

# 部分分式展开

**[r,p,k] = residue(b,a)**

$$\frac{b(s)}{a(s)} = \frac{b_1 s^m + b_2 s^{m-1} + b_3 s^{m-2} + \dots + b_{m+1}}{a_1 s^n + a_2 s^{n-1} + a_3 s^{n-2} + \dots + a_{n+1}}$$

$$\frac{b(s)}{a(s)} = \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} + \dots + \frac{r_n}{s - p_n} + k(s)$$

# 部分分式展开

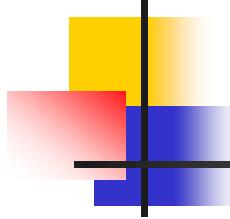
例：

$$\frac{b(s)}{a(s)} = \frac{5s^3 + 3s^2 - 2s + 7}{-4s^3 + 8s + 3}$$

**b** = [ 5 3 -2 7]

**a** = [-4 0 8 3]

[**r**, **p**, **k**] = residue(**b**, **a**)

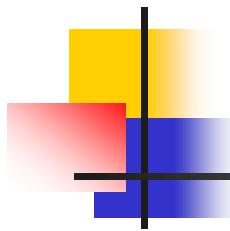


# 部分分式展开

例：

$$H(s) = \frac{s^2}{s + 1}$$

```
b = [ 1 0 0]
a = [ 1 1]
[r, p, k] = residue(b, a)
```



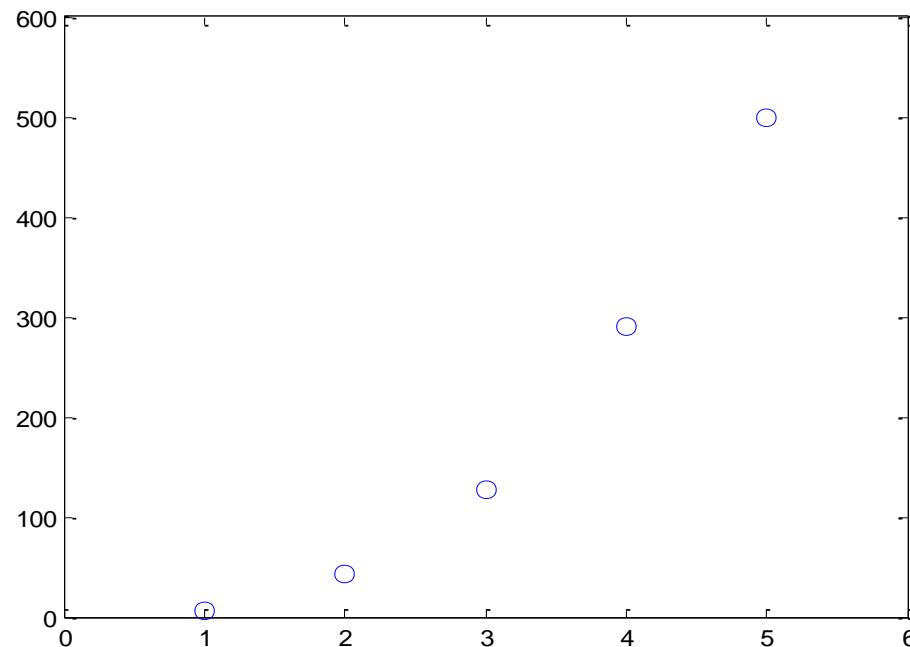
# 部分分式展开

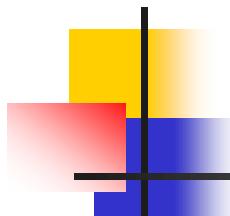
- 若已知展开后的系数，想要还原原来的多项式，仍然用**residue**函数

$[b, a] = \text{residue}(r, p, k)$

# 多项式数据拟合

- 曲线拟合：用数学函数 $y=f(x)$ 来表达一组数据的变化规律。
- 若 $f(x)$ 为多项式，也称为多项式拟合。





# 多项式数据拟合

- 假定我们要用一阶多项式拟合:  $f(x) = a_1x+a_0$
- 问题: 如何选择多项式系数 $a_1, a_2$  ?
- 准则: 均方误差最小;
- 方法: 最小二乘法。

**p = polyfit(x,y,n)**

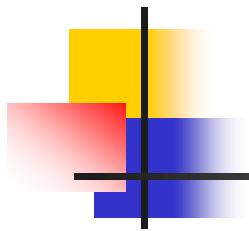
x,y:已知数据点的横纵坐标,

n : 事先确定的多项式阶次

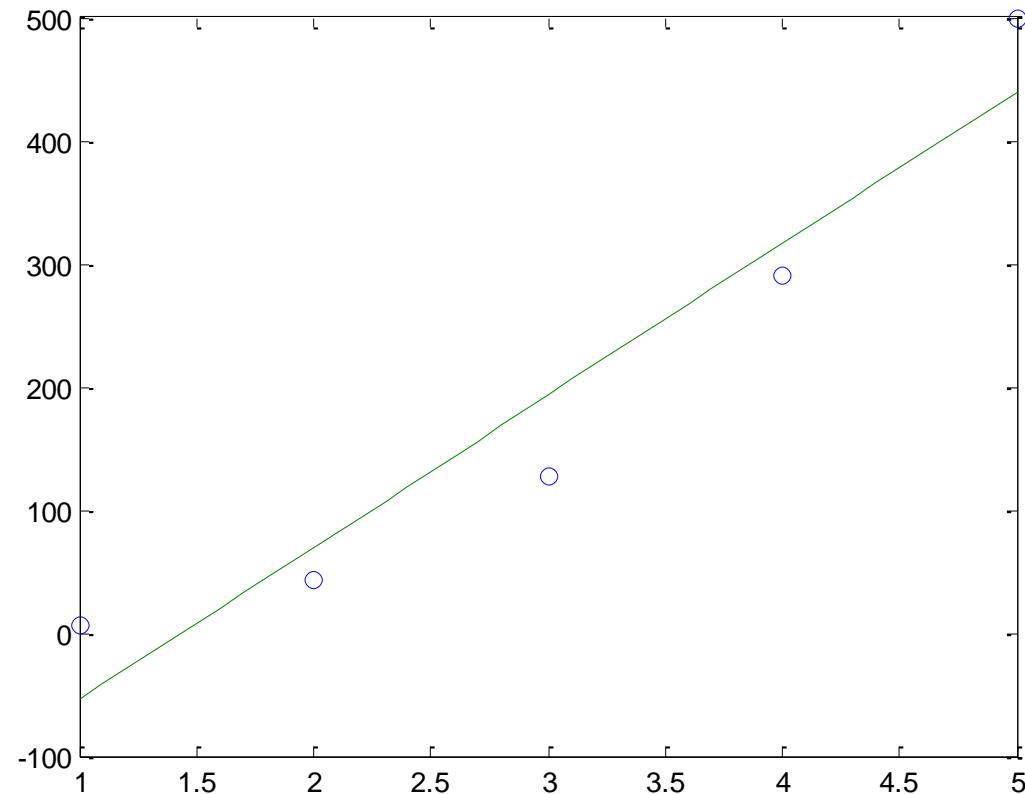
p: 返回的多项式系数

# 多项式数据拟合

```
x = [1 2 3 4 5];  
y = [5.5 43.1 128 290.7 498.4];  
p = polyfit(x,y,1);  
x2 = 1:0.1:5;  
y2 = polyval(p,x2);  
plot(x,y,'o',x2,y2);  
grid on  
yy = polyval(p,x);  
err = sum((y-yy).^2);
```

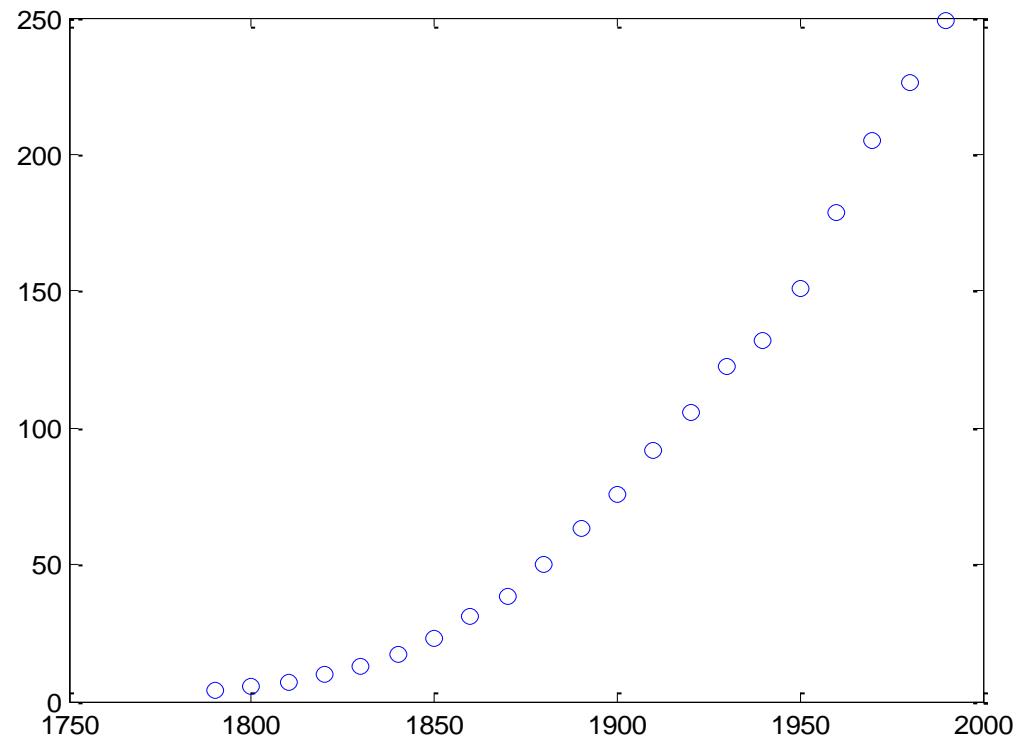


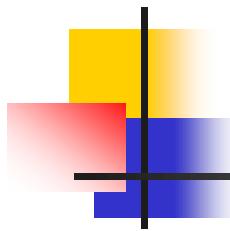
# 多项式数据拟合



# 多项式数据拟合

- 美国人口在1790年至1990年（10年一期）的人口数量统计数据。（census.mat）





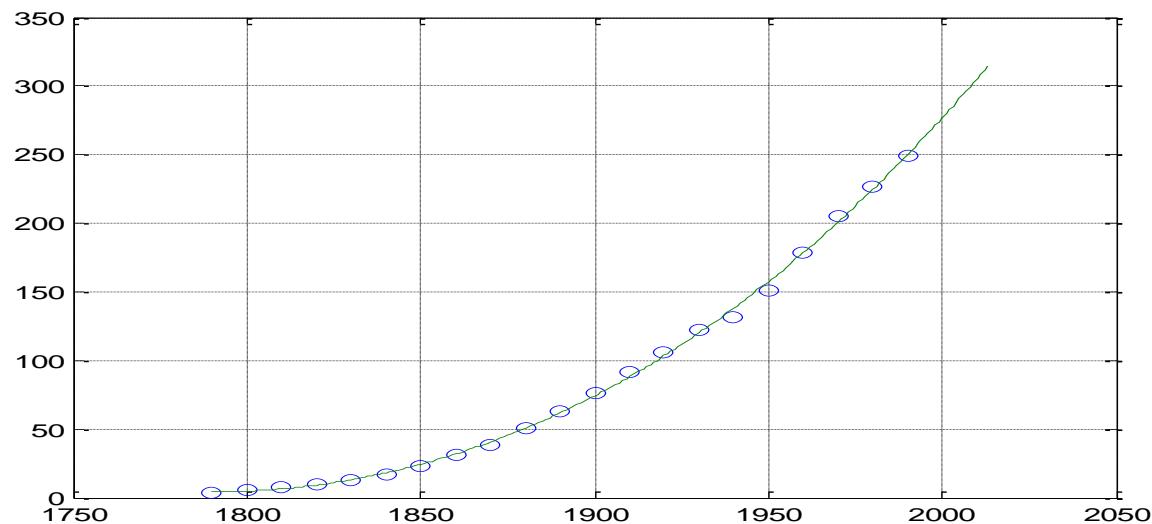
# 多项式数据拟合

- 希望预测美国在2013年的总人口

```
load census.mat  
p3 = polyfit(cdate,pop,3);  
cdate2 = 1790:2013;  
pop2 = polyval(p3,cdate2);  
plot(cdate,pop,'o',cdate2,pop2);  
pop2013 = polyval(p3,2013)  
pop2013 =
```

314.2938

# 多项式数据拟合



>> pop2(end)

ans = 3.263193232696340e+02

The screenshot shows a web browser window with three tabs open:

- Google search results for "美国人口数量 2017"
- A tab titled "人口時鐘 - 美國目前的人口" (Population Clock - Current Population of the United States)
- A tab titled "2017年美国人口数量, 人口统计" (2017 U.S. Population, Demographics)

The main content area displays the following information:

This website uses cookies to ensure you get the best experience on our website [More info](#) [Got it!](#)

**countryometers** 广告 Google 1. 人口統計 2. 美國 3. 印度人口 聯繫我們

**美國人口**

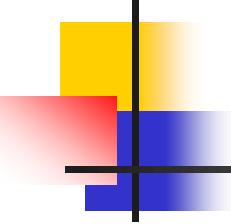
Google 已關閉廣告  
不再顯示這則廣告  
為什麼會顯示這則廣告 ? ▶

**美國人口時鐘**

11-07-2017 20:57:07

**326 726 812** 目前的人口

**161 296 353** 目前的男性人口 (49.4%)

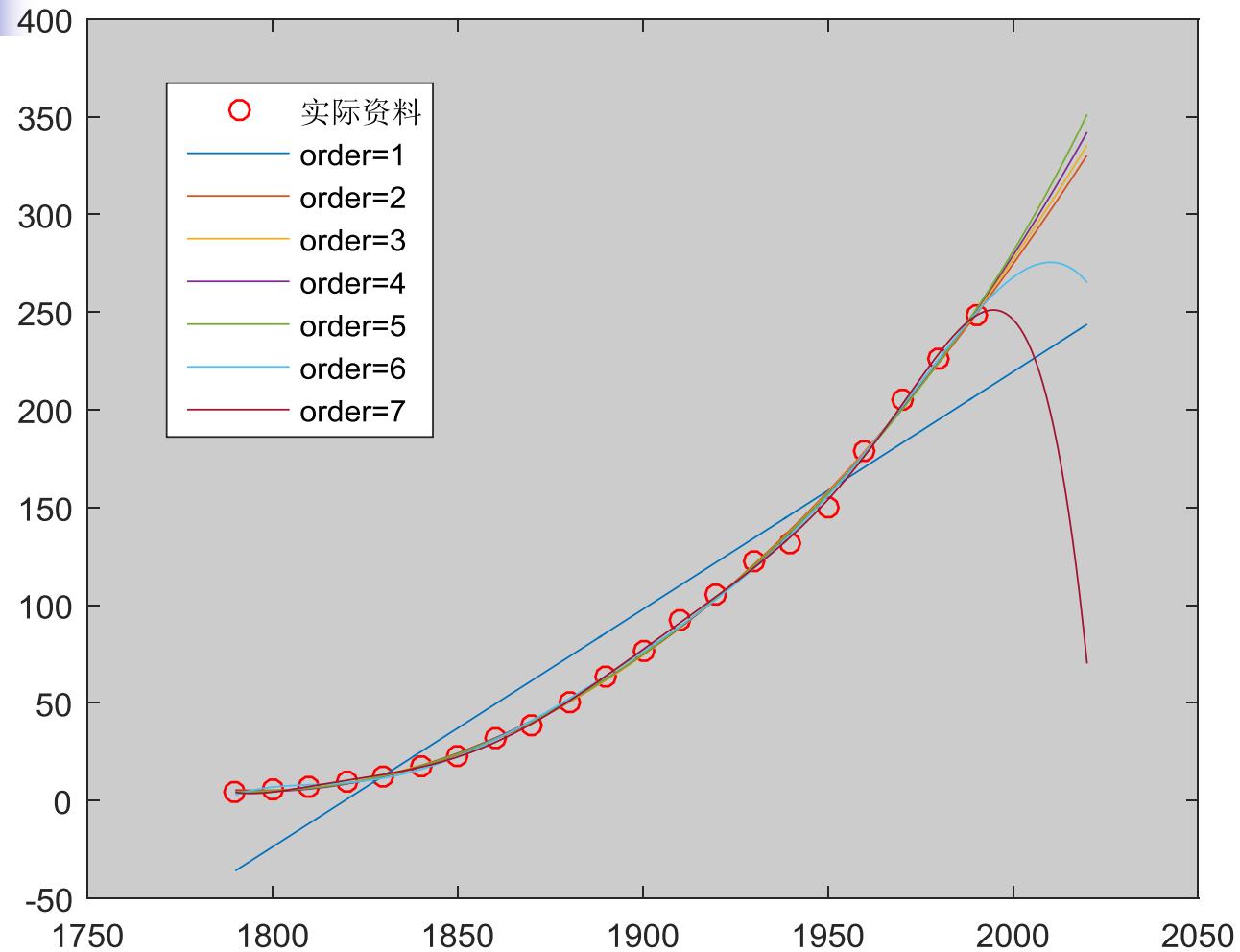


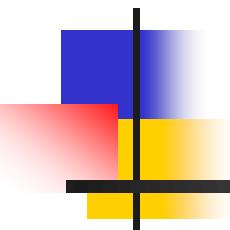
# 多项式数据拟合

- 选择不同阶次多项式拟合，预测结果差异很大。

```
load census.mat
cdate2 = min(cdate) : (max(cdate)+30) ;
curve = zeros(7, length(cdate2)) ;
for i = 1:7
curve(i,:) = polyval(polyfit(cdate,pop,i),cdate2) ;
end
plot(cdate, pop, 'o'), hold on,
plot(cdate2, curve);
legend('实际资料', 'order=1', 'order=2', 'order=3',
'order=4', 'order=5', 'order=6', 'order=7');
```

# 多项式数据拟合

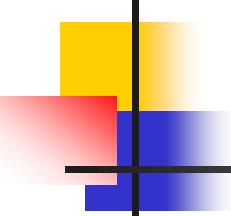




## Part2:

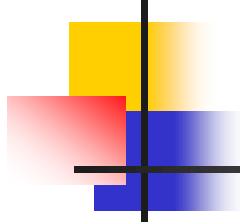
---

# 微分方程数值解法



# 微分方程数值解法

- Matlab中包含一套完整的微分方程数值解法的算法程序
  - ODE: 常微分方程（初值）
  - BVP: 边界值问题的常微分方程
  - DDE: 延迟微分方程
  - IDE: 隐微分方程
  - PDE: 偏微分方程
- 其符号工具箱(Symbolic Math Toolbox)提供了求微分方程解析解的算法程序



# 微分方程数值解法

- 本节讲解Matlab中初值问题常微分方程的数值解法。

ODE: ordinary differential equations

Initial Value Problems for ODEs

# 微分方程数值解法

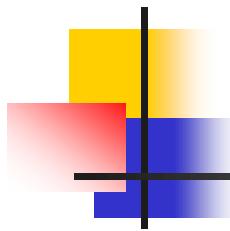
- 微分方程数值解法介绍--- Euler 折线法
- 考虑一维经典初值问题

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0, \quad x \in [a, b]$$

方程

初值

求值区间



# 微分方程数值解法

◆ 基本思想：用差商代替微商

根据 Talyor 公式， $y(x)$  在点  $x_k$  处有

$$y(x) = y(x_k) + (x - x_k)y'(x_k) + O(\Delta x^2)$$

$$y(x_{k+1}) = y(x_k) + hy'(x_k) + O(h^2)$$

$$h = x_{k+1} - x_k$$

$$\left. \frac{dy}{dx} \right|_{x_k} = \frac{y(x_{k+1}) - y(x_k)}{h} + O(h) \approx \frac{y(x_{k+1}) - y(x_k)}{h}$$

# 微分方程数值解法

□ 具体步骤：分割求解区间，差商代替微商，解代数方程

◆ 分割求解区间

等距剖分： $a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$

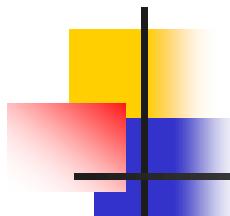
步长： $h = x_{k+1} - x_k = (b - a) / n, k = 0, 1, 2, \dots, n - 1$

◆ 差商代替微商

$$\left. \frac{dy}{dx} \right|_{x_k} \approx \frac{y(x_{k+1}) - y(x_k)}{h} \rightarrow y(x_{k+1}) \approx y(x_k) + h y'(x_k)$$

得方程组：

$$\begin{cases} y_0 = y(x_0) \\ y_{k+1} = y_k + h f(x_k, y_k) \\ x_{k+1} = x_k + h \end{cases}$$



# 微分方程数值解法

例：用 Euler 法解初值问题

$$\begin{cases} \frac{dy}{dx} = y + \frac{2x}{y^2} & x \in [0, 2] \\ y(0) = 1 \end{cases}$$

解：取步长  $h = (2 - 0)/n = 2/n$ ，得差分方程

$$\begin{cases} x_0 = 0, \quad y_0 = 1 \\ y_{k+1} = y_k + h f(x_k, y_k) = y_k + h(y_k + 2x_k / y_k) \\ x_{k+1} = x_k + h \end{cases}$$

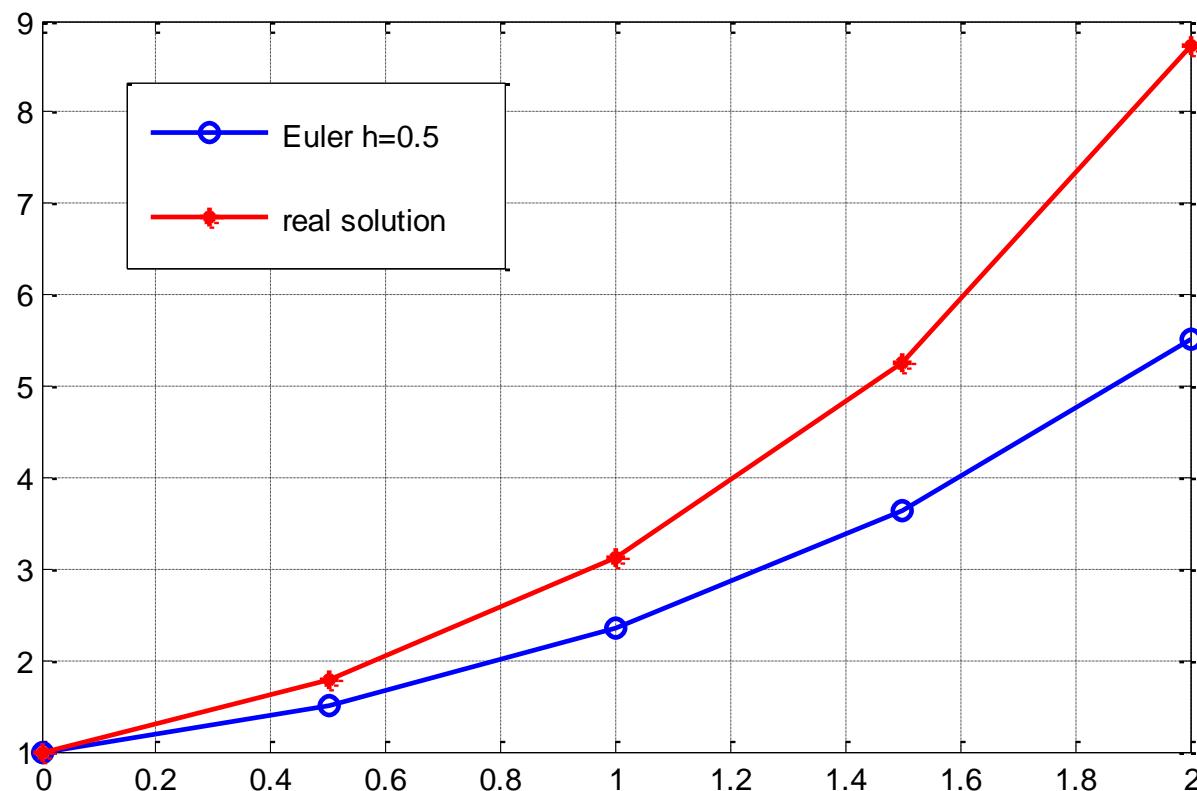
练习：编写**matlab**程序，用**Euler**法解上述微分方程

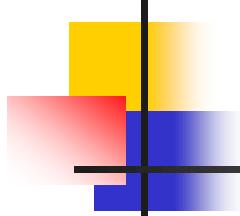
# 微分方程数值解法

```
a = 0; %[a b]为积分区间  
b = 2;  
n = 5; %中间点数  
h = (b-a) / (n-1); %步长  
x = linspace(a,b,n);  
y = zeros(size(x));  
y(1) = 1;  
for k = 1:n-1  
    y(k+1) = y(k) +h * (y(k)+x(k) / (y(k).^2));  
end  
plot(x,y, 'o-');
```

# 微分方程数值解法

**解析解:**  $y = \left( \frac{5}{3} e^{3x} - 2x - \frac{2}{3} \right)^{1/3}$



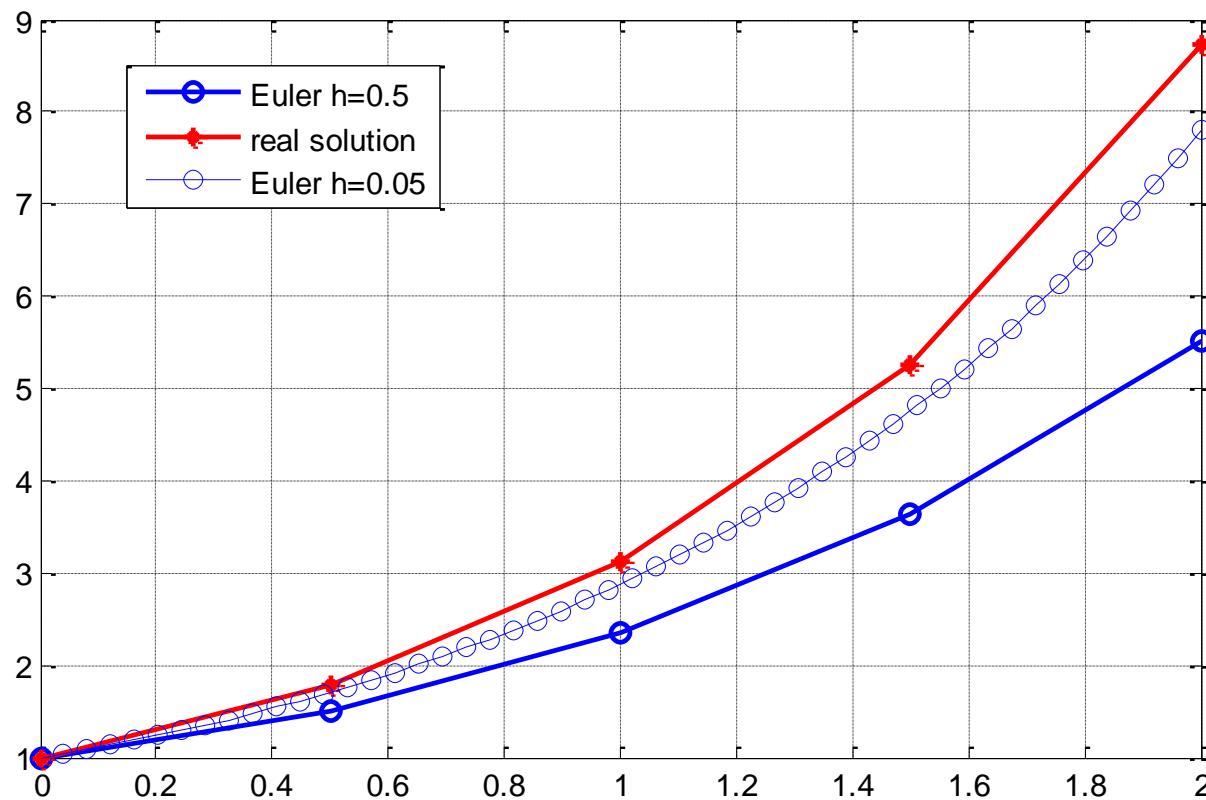


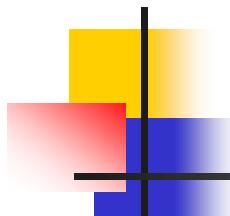
□ 为了减小误差，可采用以下方法：

- ◆ 让步长  $h$  取得更小一些；
- ◆ 改用具有较高精度的数值方法：  
**Runge-Kutta (龙格-库塔) 方法**

# 微分方程数值解法

- 步长减小，精度会提高。





# 微分方程数值解法

## □ 四阶R-K方法

$$\begin{cases} y_0 = y(x_0), & x_{k+1} = x_k + h \\ y_{k+1} = y_k + h(L_1 + 2L_2 + 2L_3 + L_4)/6 \end{cases}$$

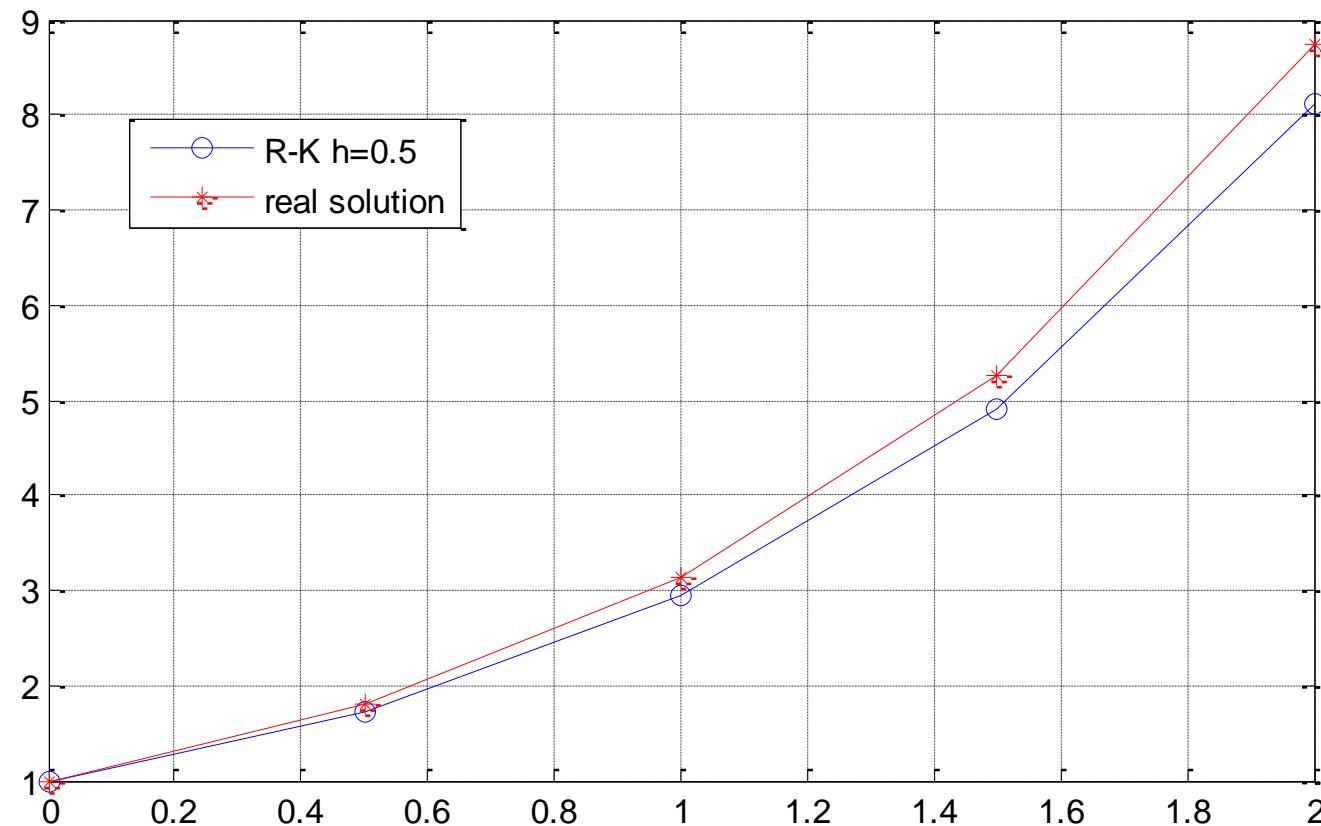
其中

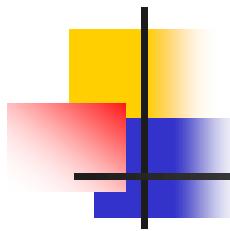
$$\begin{cases} L_1 = f(x_k, y_k) \\ L_2 = f(x_k + h/2, y_k + hL_1/2) \\ L_3 = f(x_k + h/2, y_k + hL_2/2) \\ L_4 = f(x_k + h, y_k + hL_3) \end{cases}$$

# 微分方程数值解法

```
a = 0; b = 2; n = 50; h = (b-a)/(n-1);
x = linspace(a,b,n); y = zeros(size(x));
y(1) = 1;
f = @(u,v) u+v/u^2;
for k = 1:n-1
    L1 = f(y(k),x(k));
    L2 = f(y(k)+h*L1/2, x(k)+h/2);
    L3 = f(y(k)+h*L2/2, x(k)+h/2);
    L4 = f(y(k)+h*L3, x(k)+h );
    y(k+1) = y(k) +h *(L1+2*L2+2*L3+L4)/6;
end
```

# 微分方程数值解法





# 微分方程数值解法

## □ 用 Matlab自带函数 解初值问题

◆ 求解析解: `dsolve`

◆ 求数值解:

`ode45`、`ode23`、

`ode113`、`ode23t`、`ode15s`、

`ode23s`、`ode23tb`

# 微分方程数值解法

```
[T, Y] = solver(odefun, tspan, y0)
```

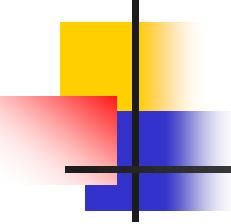
其中  $y_0$  为初值条件，  $tspan$  为求解区间； Matlab 在数值求解时 **自动对求解区间进行分割**，  $T$  (向量) 中返回的是分割点的值(自变量)，  $Y$  (向量) 中返回的是解函数在这些分割点上的函数值。

*solver* 为 Matlab 的 **ODE 求解器** (可以是  $ode45$ 、 $ode23$ 、 $ode113$ 、 $ode15s$ 、 $ode23s$ 、 $ode23t$ 、 $ode23tb$ )

没有一种算法可以有效地解决所有的 ODE 问题，因此 MATLAB 提供了 **多种 ODE 求解器**，对于不同的 ODE，可以调用不同的求解器。

# 微分方程数值解法

| 求解器                  | ODE类型 | 特点                                            | 说明                                  |
|----------------------|-------|-----------------------------------------------|-------------------------------------|
| <code>ode45</code>   | 非刚性   | 单步法; 4, 5 阶 R-K 方法;<br>累计截断误差为 $(\Delta x)^3$ | 大部分场合的首选方法                          |
| <code>ode23</code>   | 非刚性   | 单步法; 2, 3 阶 R-K 方法;<br>累计截断误差为 $(\Delta x)^3$ | 使用于精度较低的情形                          |
| <code>ode113</code>  | 非刚性   | 多步法; Adams 算法; 高低精度均可到 $10^{-3} \sim 10^{-6}$ | 计算时间比 <code>ode45</code> 短          |
| <code>ode23t</code>  | 适度刚性  | 采用梯形算法                                        | 适度刚性情形                              |
| <code>ode15s</code>  | 刚性    | 多步法; Gear's 反向数值微分; 精度中等                      | 若 <code>ode45</code> 失效时, 可尝试使用     |
| <code>ode23s</code>  | 刚性    | 单步法; 2 阶 Rosebrock 算法; 低精度                    | 当精度较低时, 计算时间比 <code>ode15s</code> 短 |
| <code>ode23tb</code> | 刚性    | 梯形算法; 低精度                                     | 当精度较低时, 计算时间比 <code>ode15s</code> 短 |



# 微分方程数值解法

例：求初值问题  $\begin{cases} \frac{dy}{dx} = -2y + 2x^2 + 2x \\ y(0) = 1 \end{cases}$  的数值解，求解范围为 [0,0.5]

```
%fun=inline(' -2*y+2*x^2+2*x' , 'x' , 'y') ;
fun = @(x,y) -2*y+2*x^2+2*x;
[x,y]=ode23(fun,[0,0.5],1);
```

注：也可以在 `tspan` 中指定对求解区间的分割，如：

```
[x,y]=ode23(fun,[0:0.1:0.5],1); %此时 x=[0:0.1:0.5]
```

# 微分方程数值解法

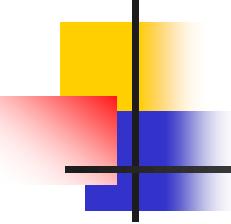
如果需求解的问题是高阶常微分方程，则需将其化为一阶常微分方程组，此时需用函数文件来定义该常微分方程组。

例：求解 Ver der Pol 初值问题

$$\begin{cases} \frac{d^2y}{dt^2} - \mu(1-y^2)\frac{dy}{dt} + y = 0 \\ y(0) = 1, \quad y'(0) = 0, \quad \mu = 7 \end{cases}$$

令  $x_1 = y, x_2 = \frac{dy}{dt}$ ，则原方程可化为

$$\begin{cases} dx_1/dt = x_2 \\ dx_2/dt = \mu(1-x_1^2)x_2 - x_1 \\ x_1(0) = 1, \quad x_2(0) = 0, \quad \mu = 7 \end{cases}$$



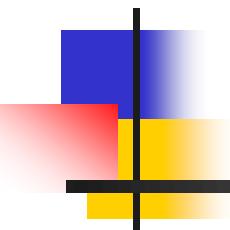
# 微分方程数值解法

- 先编写函数文件 `verderpol.m`

```
function xprime=verderpol(t,x)
global mu;
xprime=[x(2) ; mu*(1-x(1)^2)*x(2) - x(1)];
```

- 再编写脚本文件 `vdpl.m`, 在命令窗口直接运行该文件。

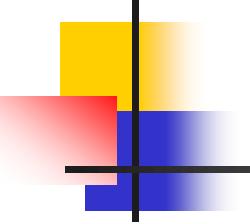
```
clear;
global mu;mu=7;
y0=[1;0];
[t,x]=ode45('verderpol',[0,40],y0);
plot(t,x(:,1),'r-');
```



## Part3:

---

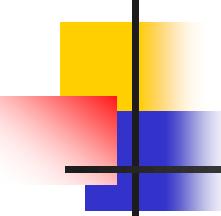
# 符号计算



# 内容

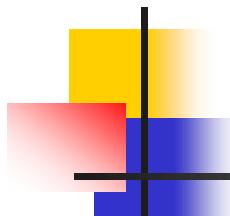
---

- 符号计算与数值计算
- 符号计算基础
- 积分
- 求导
- 代数方程求解
- 微分方程求解
- 傅立叶变换
- 拉普拉斯变换



# 符号计算基础

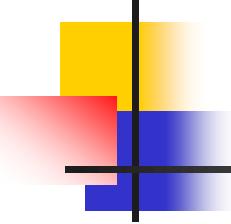
- 符号数学工具箱(Symbolic Math Toolbox)
- 对函数表达式进行微分、积分等运算，求解代数方程、微分方程。结果为相应解析解。
- 先定义符号对象（符号变量、符号表达式），再调用相应符号函数进行推理，得到解析解。



# 符号对象的建立

- 符号变量定义： **sym** 函数， **syms** 函数
- $a = \text{sym}('a');$
- **syms c x t alpha ;** %注意变量间不能有逗号！

注意：**pi,i,j** 不能作为符号变量！



# 符号对象的建立

- 符号表达式建立：
  - 利用**sym**函数直接建立

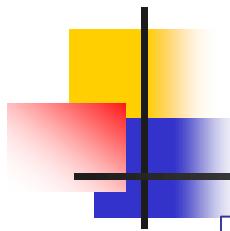
(不推荐，以后版本可能不支持)

例： $x^a e^{-x}$

```
f=sym('x^a*exp(-x)'); %注意单引号！
```

- 使用已定义符号变量组成

```
>> syms x,a  
>> f = x^a*exp(-x)*sin(2*pi*x)  
>> pretty(f)
```



# 变量替换

- **subs(f,x,a)**

将符号表达式**f**中的符号变量**x**替换为**a**

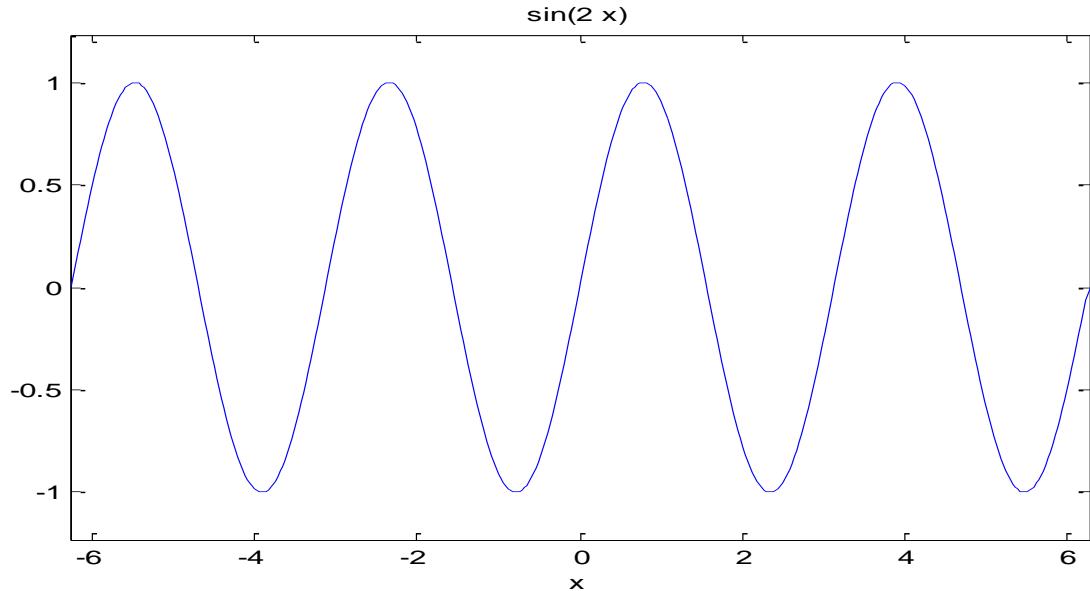
**a**可以为 数值/符号变量/符号表达式

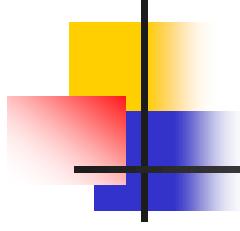
```
syms a b c d;  
A = [ a b ;c d];  
x=det(A);  
x1 = subs(x,a,3);  
x2 = subs(x,[ b c d] ,[1 7 4]);  
x3 = double(x2); %把符号变量变成双精度数
```

# 函数画图

## ■ ezplot(f)

```
>> syms x  
>> ezplot(sin(2*x)) %默认横坐标范围[-2pi,2pi]
```



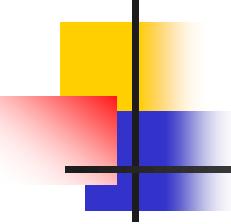


# 函数画图

- 画由参数方程 $x(t), y(t)$ 决定的曲线：

**ezplot(x,y)**

```
syms t  
x = t*sin(5*t);  
y = t*cos(5*t);  
ezplot(x,y)
```

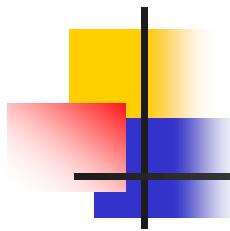


# 函数求导 (Differentiation)

- **diff(f)** 对函数f求导

```
>> syms x  
>> f = sin(x^2)  
>> df = diff(f)
```

- ```
>> syms x t  
>> diff(x*t^2))  
>> diff(x*t^2,t)
```

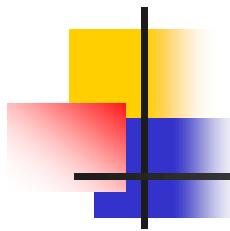


# 函数求导 (Differentiation)

- **diff(f)** 对函数f求导

- $\frac{df(x)}{dx}$ ,  $f(x) = \sin(5x)$

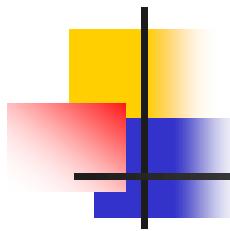
```
>> syms x  
>> f = sin(5*x);  
>> diff_f = diff(f)
```



# 函数求导 (Differentiation)

- $\frac{d^2f(t)}{dt^2}$ ,  $f(t) = \sin(5t)$

```
>> syms t  
>> f = sin(5*t);  
>> diff_f = diff(f,2) %对f求二阶导数
```



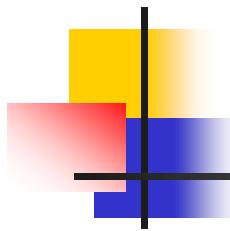
# 函数求导 (Differentiation)

- $\frac{df(x)}{dx}$ ,  $f(x) = e^{-ax}\sin(bx)$

```
>> syms a b x  
>> f = exp(-a*x)*sin(b*x);  
>> diff_f = diff(f) %对变量x求导
```

```
>> subs(f,a,2) %将变量a替换为2  
>> subs(f,[a b],[2 3])
```

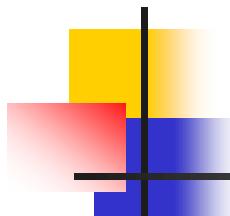
```
>> diff_f2 = diff(f,a) % 对变量a求导
```



# 找出符号函数中的符号变量

- **symvar** 函数

```
syms x y a b  
f(a, b) = a*x^2/(sin(3*y - b));  
symvar(f)
```

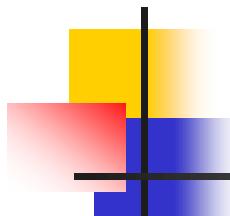


# 不定积分(indefinite integral)

- **int(expr)** 对函数表达式expr进行不定积分

$$\int (1 + t) dt$$

```
>> syms t  
>> int_f=int(1+t)  
int_f =  
(t*(t + 2))/2
```



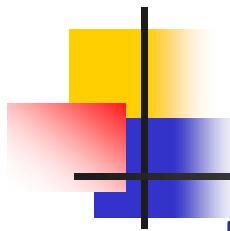
# 不定积分(indefinite integral)

- **int(expr,v)** 对函数表达式expr进行不定积分,指定积分变量为v

$$\int \frac{x}{1+z^2} dx$$

$$\int \frac{x}{1+z^2} dz$$

```
>> syms x z  
>> f = x/(1+z^2)  
>> int_f1 = int(f) %默认积分变量为x  
>> int_f2 = int(f,z)%指定积分变量为z
```

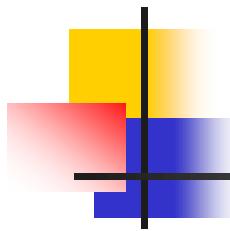


# 定积分(definite integral)

- **int(expr,a,b)** 对函数表达式expr在区间[a, b]求定积分

$$\int_0^6 x dx$$

```
>> syms x  
>> f = x;  
>> int_f = int(f,0,6)
```



# 定积分(definite integral)

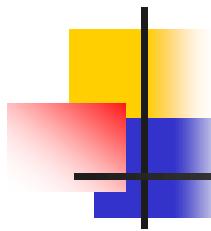
$$\int_{-6}^6 x^2 \cos(x) dx$$

```
>> syms x
>> f = x^2*cos(x);
>> int_f = int(f,-6,6)
int_f =
24*cos(6) + 68*sin(6)
>> double(int_f)      %转换为双精度数
ans = 4.043833002081825
```

求曲线  $e^x$  关于 x 轴旋转得到的旋转体在  $1 \leq x \leq 2$  内的体积

$$\int_a^b \pi[f(x)]^2 dx$$

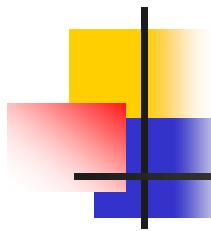
```
syms x a b
f = exp(-x)
intf = int(pi*f^2 , a ,b )
V = subs(intf,[a b],[1 2])
v = double(V);
```



# 数值积分

`q = quad(fun,a,b)`    fun:被积函数;  
                              a,b:积分区间  
  
(不推荐)

- `>>g = @(x) exp(sin(x));`
- `>>quad(g,0,10)`
- `>>quad('exp(sin(t))',0,10);`

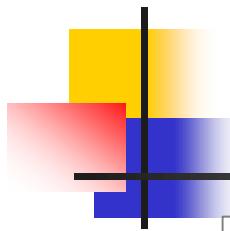


# 数值积分

```
q = integral(fun,xmin,xmax) fun:被积函数;  
xmin,xmax :积分区间
```

例：求积分 $f(x) = e^{-x^2}(\ln x)^2$  [0, +∞]

- >>f = @(x) exp(-x.^2).\*log(x).^2;
- >>integral(f,0,+inf)



## 二重数值积分

```
q = integral2(fun,xmin,xmax,ymin,ymax)
```

例：求积分  $f(x) = \frac{1}{\sqrt{x+y}(1+x+y)}$

积分区间 :  $0 \leq x \leq 1, 0 \leq y \leq 1 - x$

- $f = @(x,y) 1./(\sqrt{x+y}.*(1+x+y));$
- $y_{\text{max}} = @(x) 1-x;$
- $\text{integral2}(f,0,1,0,y_{\text{max}});$

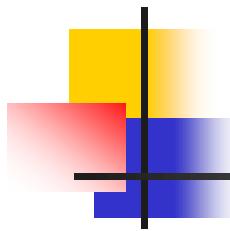
# 定积分(definite integral)

## ■ 计算卷积:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t - \tau)d\tau$$

设  $x(t) = h(t) = e^{-t^2/2}$

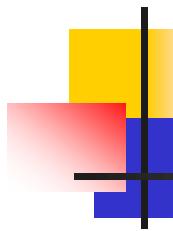
```
syms t tau
y = int(exp(-tau^2/2)*exp(-(t-tau)^2/2), -inf,inf)
y = 2^(1/2)*pi^(1/2)*exp(-tau^2/2)          %不对!
y = int(exp(-tau^2/2)*exp(-(t-tau)^2/2),tau,-inf,inf)
y = pi^(1/2)*exp(-t^2/4)                      %OK!
```



# 定积分(definite integral)

计算卷积:  $e^{-t^2/2} * e^{-t^2/2}$

- >> syms t tau
- >> x = @(t)exp(-t^2/2)
- >> f =x(tau)\*x(t-tau)
- >> y = int(f,tau,-inf,inf)



# 定积分(definite integral)

$$x(t) = \cos(\omega_0 t)$$
$$h(t) = e^{-6t} u(t)$$

```
>> syms t tau w0 real
>> f = cos(w0*(t-tau))*exp(-6*(tau))*heaviside(tau)
>> int(f,tau,-inf,inf)

ans =
(6*cos(t*w0) + w0*sin(t*w0))/(w0^2 + 36)
```

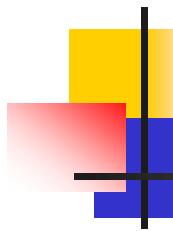
- **heaviside(x)** 阶跃函数

$$u(t) = \begin{cases} 1 & t > 0 \\ 0 & t < 0 \end{cases}$$

- **dirac(x)** 狄拉克函数

$$\delta(t) = 0 \quad t \neq 0$$

$$\int_{-\infty}^{+\infty} \delta(t) dt = 1$$



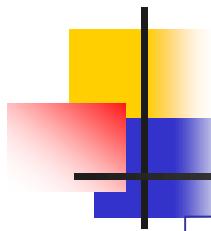
# 多重不定积分

$$\iiint xy^2z^5 dx dy dz$$

```
>> syms x y z  
>> int(int(int(x*y^2*z^5,x),y),z)
```

ans =

$$(x^2 y^3 z^6)/36$$



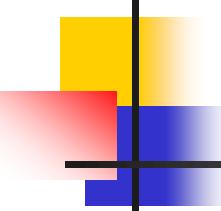
# 多重定积分

$$\int_1^2 \int_2^4 x^2 y dx dy$$

```
>> syms x y  
>> int(int(x^2*y,x,2,4),y,1,2)
```

ans =

28

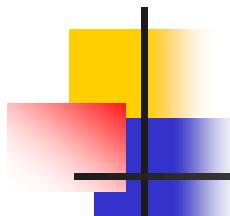


# 代数方程求解

- **solve(eqn)** 求解代数方程eqn

$$x^2 + 3x + 2 = 0$$

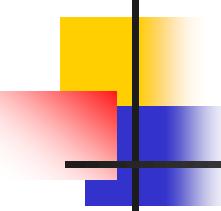
```
>> S = solve('x^2+3*x+2') %默认为 ...=0  
>> S(1) %S是一个数组  
ans = -2  
>> S1 = solve('x^2+3*x+2=0') %结果一样
```



# 代数方程求解

$$x^2 + 3x + 2 = 0$$

```
>> syms x  
>> solve(x^2+3*x+2) %表达式不加单引号  
>> solve(x^2+3*x+2=0) %出错  
>> solve(x^2+3*x+2==0) %正确  
>> solve(3*x == -x^2-2) %可以
```



# 代数方程求解

- **solve(eqn, var)** 指定变量var为未知数

$$ax^2 + bx + c = 0$$

```
>> syms a b c x  
>> solve(a*x^2+b*x+c == 0) %默认变量为x  
>> solve(a*x^2+b*x+c == 0, c) %指定变量为c
```

# 代数方程求解

$$x^3 + 1 = 0$$

```
>> syms x %默认x为复数
```

```
>> solve(x^3+1 == 0)
```

```
>> pretty(ans)
```

```
>> syms x real %规定x为实数
```

```
>> solve(x^3+1 == 0) %得到一个实根
```

```
>> syms x positive %规定x为正实数
```

```
>> solve(x^3+1 == 0) %答案为空
```

# 代数方程求解

求解代数方程组：

**solve(eqn1,eqn2,...,eqnN,var1,var2,...,varN')**

$$2x - 3y + 4z = 5$$

$$y + 4z + x = 10$$

$$-2z + 3x + 4y = 0$$

```
>> syms x y z
```

```
>> eq1 = '2*x-3*y+4*z=5'
```

```
>> eq2='y+4*z+x=10'
```

```
>> eq3=' -2*z+3*x+4*y=0'
```

```
>> [x,y,z] = solve(eq1,eq2,eq3,x,y,z)
```

# 常微分方程求解

## ■ $S = \text{dsolve}(\text{eqn})$

例:  $\frac{dy}{dt} = ty$

```
>> syms y(t) % 定义一个符号函数  
>> dsolve(diff(y)==t*y)  
ans =  
C2*exp(t^2/2) % C2 是待定系数, 需要初始条件确定
```

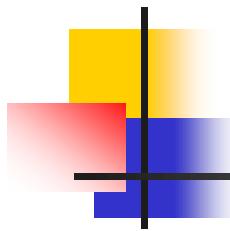
```
>>y(t) = dsolve(diff(y) == t*y, y(0) == 2)  
y(t) =  
2*exp(t^2/2)
```

# 常微分方程求解

## ■ $S = \text{dsolve}(\text{eqn})$

例:  $\frac{d^2y}{dx^2} = \cos(2 * x) - y$

```
syms y(x)
Dy = diff(y);
y(x) = dsolve(diff(y, 2) == cos(2*x) - y, y(0) == 1, Dy(0) == 0);
y(x) = simplify(y)
y(x) =
1 - (8*sin(x/2)^4)/3
```



# 傅立叶变换

## ■ Fourier正变换

$$F(jw) = \int_{-\infty}^{+\infty} f(t)e^{-jwt} dt$$

`fourier(f,trans_var,eval_point)`

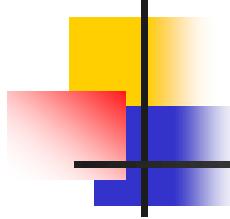
`trans_var`时域变量, `eval_point`频域变量w

## ■ Fourier逆变换:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(jw)e^{jwt} dw$$

`ifourier(F,trans_var,eval_point)`

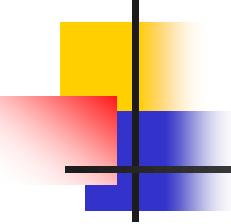
`trans_var`频域变量w, `eval_point`时域变量



# 傅立叶变换

求高斯函数  $f(x) = e^{-x^2}$  的fourier变换

```
>>syms x w  
>>f = exp(-x^2);  
>>fourier(f)  
ans =  
pi^(1/2)*exp(-w^2/4)
```

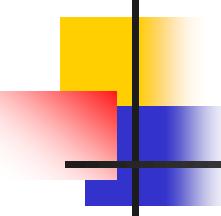


# 傅立叶变换

求方波信号的fourier变换

```
f = heaviside(t+2)- heaviside(t-2);  
ezplot(f,[-3,3]) %画出时域波形
```

```
ft = fourier(f)  
pretty(ft)  
ft2 = simplify(ft)  
figure  
ezplot(ft)
```



# 拉普拉斯变换

## ■ Laplace 正变换

$$F(s) = \int_0^{+\infty} f(t)e^{-st} dt$$

`laplace(f,trans_var,eval_point)`

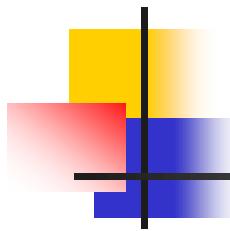
`trans_var`默认为t, `eval_point`默认为s

## ■ Laplace 逆变换:

$$f(t) = \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} F(s)e^{st} ds$$

`ilaplace(F,trans_var,eval_point)`

`trans_var`默认为s, `eval_point`默认为t



# 拉普拉斯变换

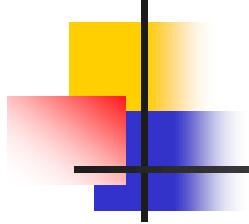
```
>> syms s t  
>> laplace(exp(-t))
```

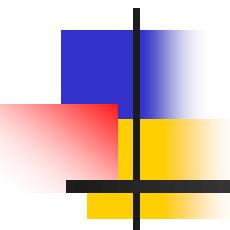
```
ans =
```

```
1/(s + 1)  
>> ilaplace(F)
```

```
ans =
```

```
exp(-t)
```

- 
- 注：本讲内容大部分引自华东师范大学课程《数学实验》

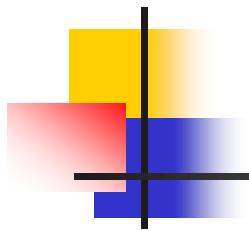


# Matlab编程及其应用

## 第七讲

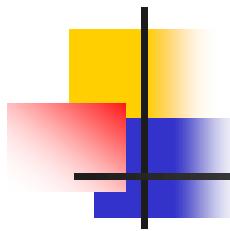
中国科技大学信息学院  
陆伟

[luwei@ustc.edu.cn](mailto:luwei@ustc.edu.cn)



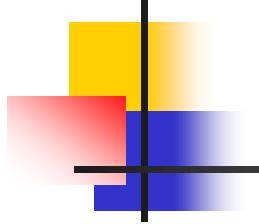
# Simulink基础

- 基本操作：启动->建立新mdl文件->导入模块->连接->仿真设置->仿真
- 常用模块库简介
- 模块基本操作
- 连线基本操作
- 例



# Simulink概述

- 模型化图形输入，对动态系统建模与仿真。
- Simulink 中的“Simu”一词表示可用于计算机仿真，而“Link”一词表示它能进行系统连接，即把一系列模块连接起来，构成复杂的系统模型。

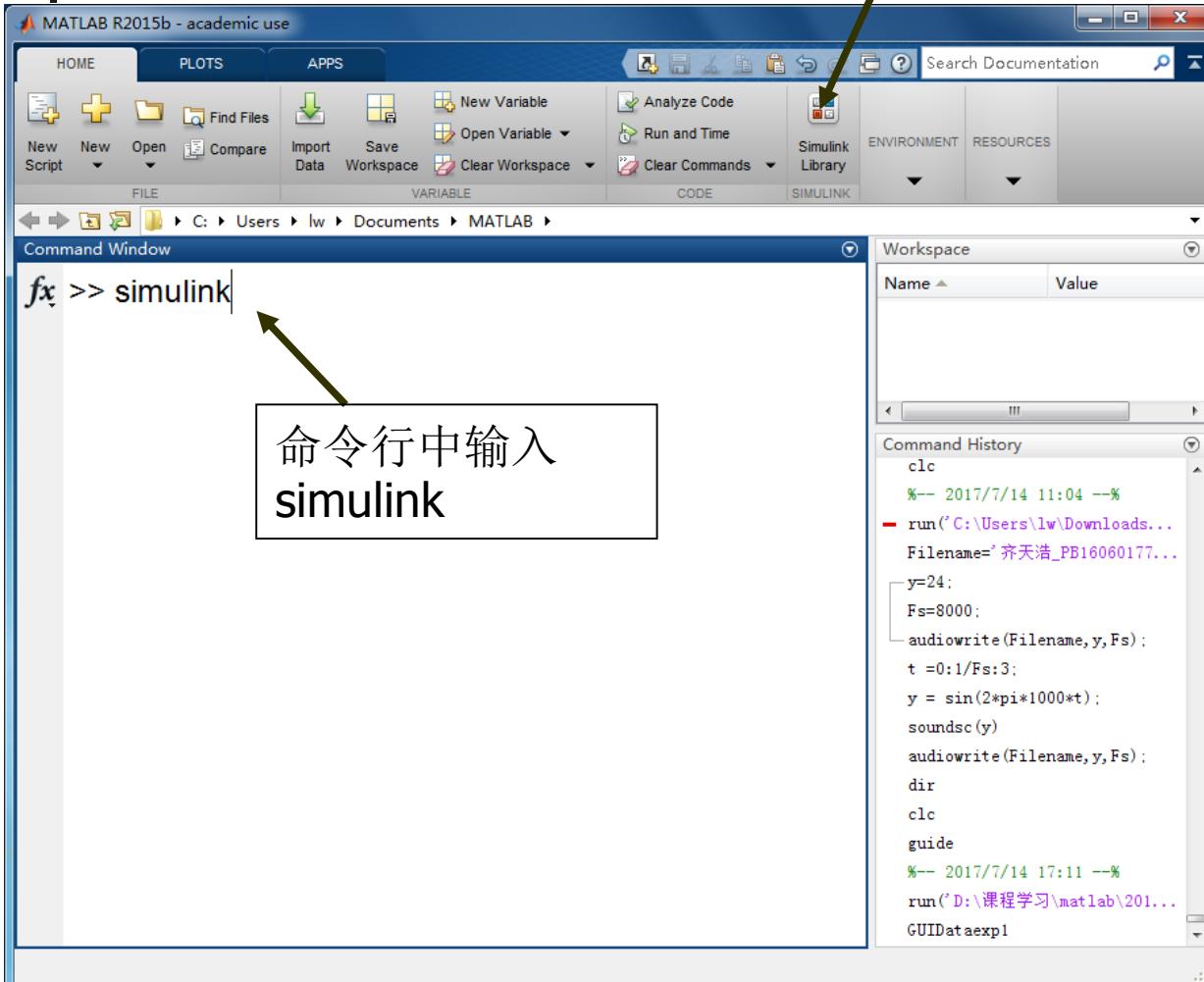


## ■ Simulink仿真模型的一般结构

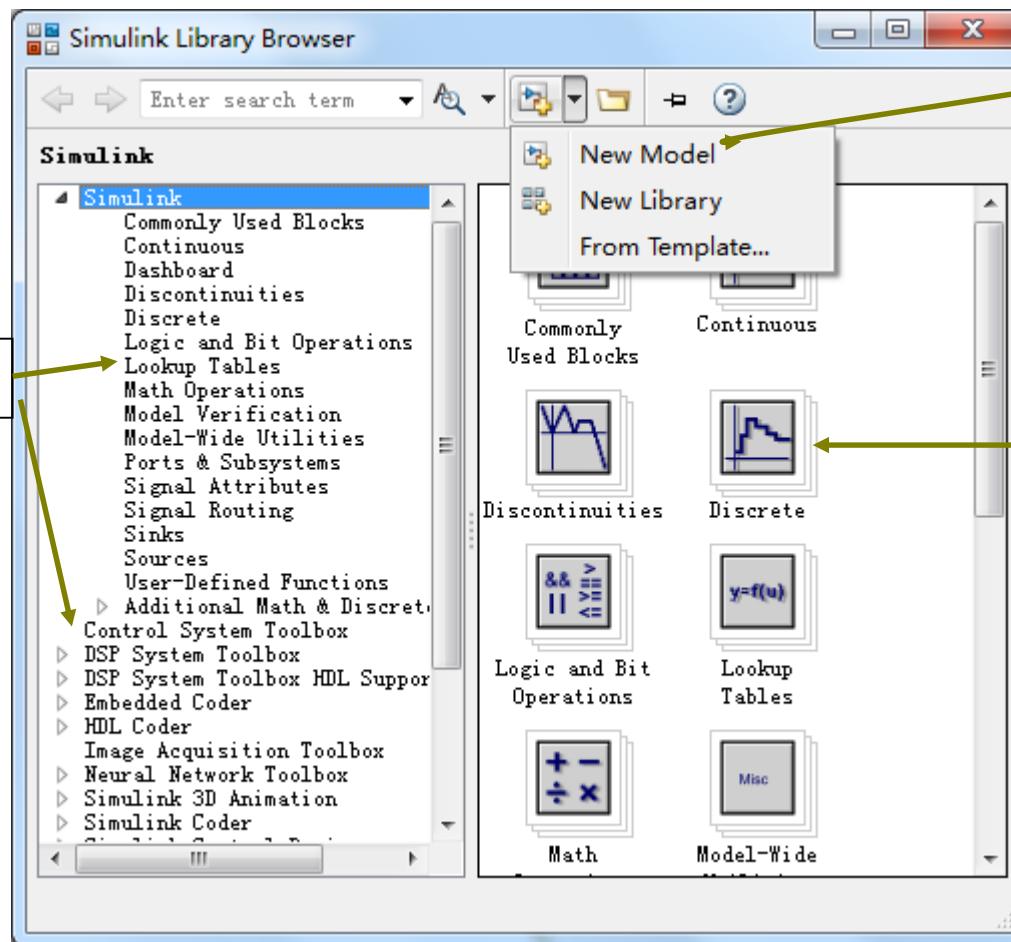


# 启动simulink

点击simulink图标



# Simulink模块库



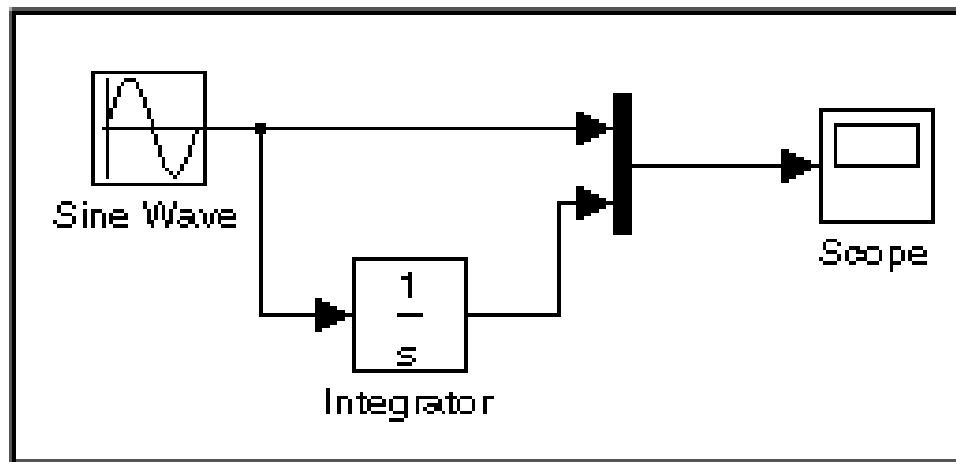
新建一个Simlink  
仿真文件

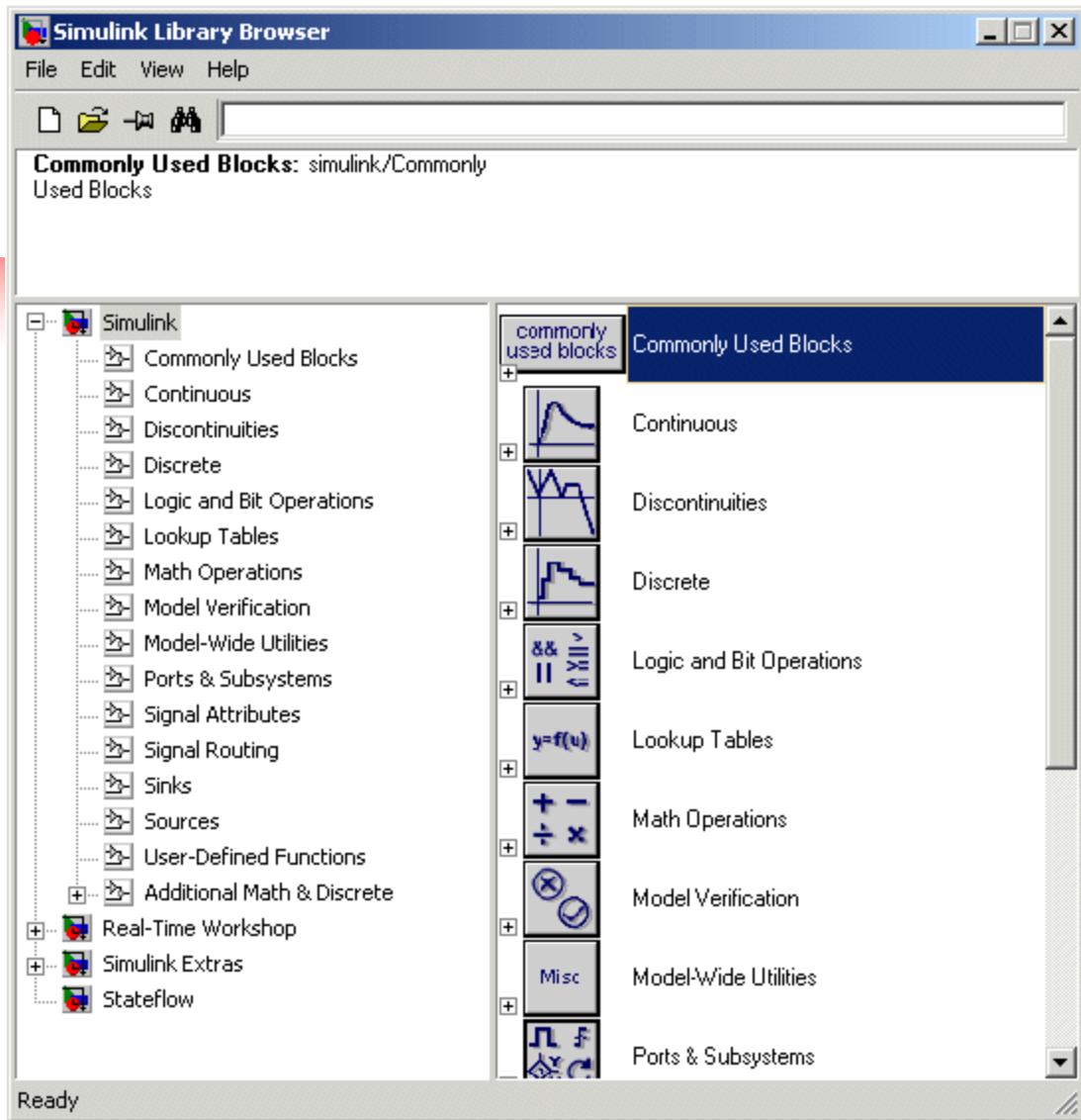
Simlink模块库

Simlink模块

# Simulink基本操作

- 复制模块：确定模型中包含哪些模块，然后在**库浏览器**中找到所需要的模块，将需要的模块从模块库中复制到模型中。



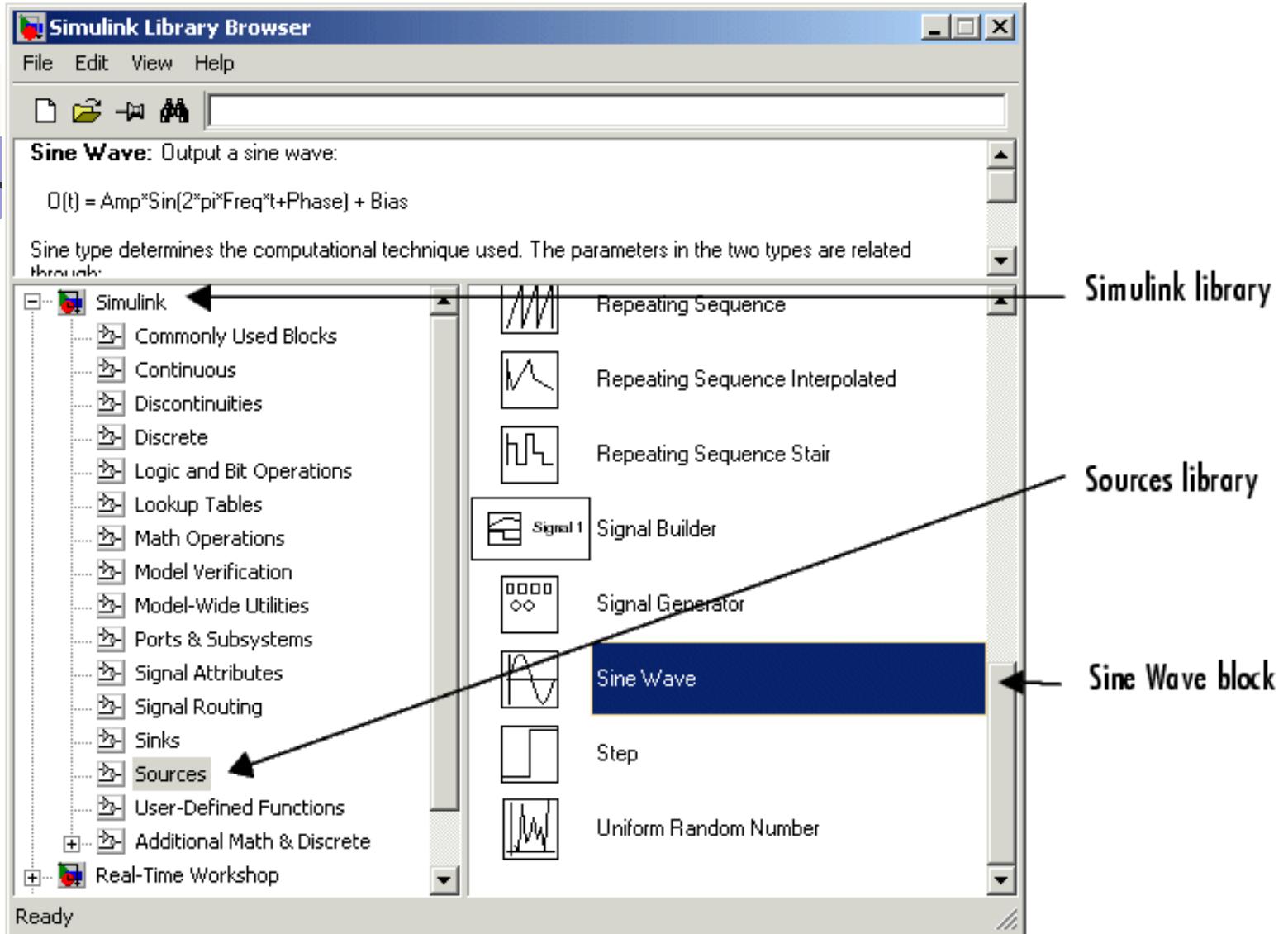


四个模块涉及的模块库分别为：

- **Sources** (信号源模块库)
- **Continuous** (连续模块库)
- **Sinks** (信号输出模块库)
- **Signal Routing** (信号路由模块库)

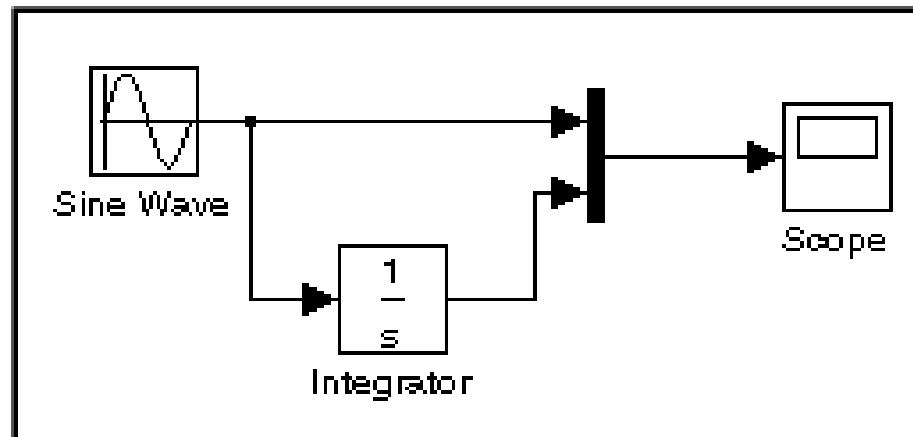
此例中，模型包括四个模块：

- 正弦波模块 (**sine Wave**)、积分模块(**Integrator**)、示波器模块 (**Scope**)、组合模块 (**Mux**)



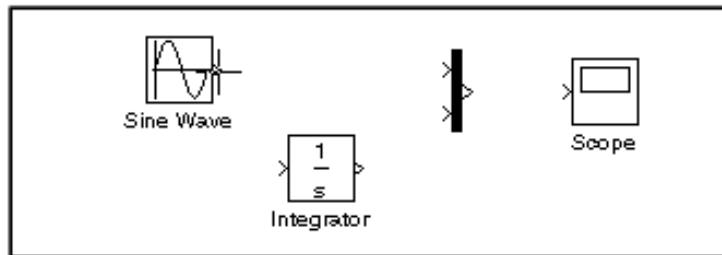
## 增加信号线

- 把一个模块的输出与另一个模块的输入连接起来
- 在一条已有的信号线上引出另一条信号线，这两条线将传送相同信号给各自对象。



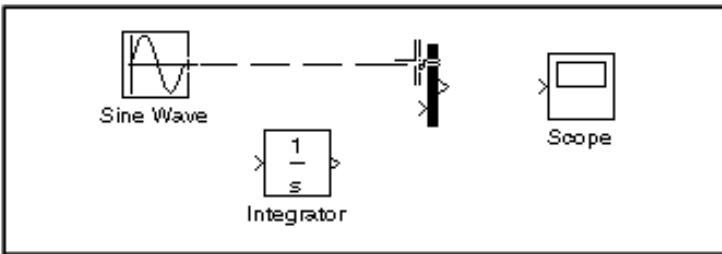
## ■ 把一个模块的输出与另一个模块的输入连接起来

Now it's time to connect the blocks. Connect the Sine Wave block to the top input port of the Mux block. Position the pointer over the output port on the right side of the Sine Wave block. Notice that the cursor shape changes to crosshairs.

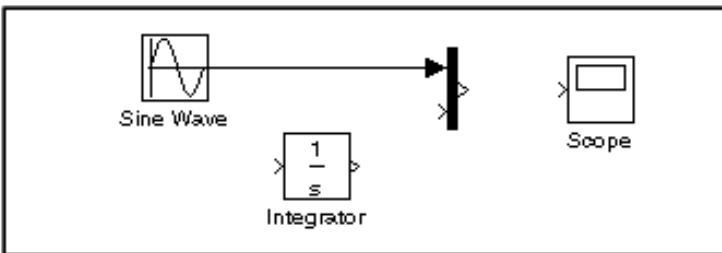


Hold down the mouse button and move the cursor to the top input port of the Mux block.

Notice that the line is dashed while the mouse button is down and that the cursor shape changes to double-lined crosshairs as it approaches the Mux block.



Now release the mouse button. The blocks are connected. You can also connect the line to the block by releasing the mouse button while the pointer is over the block. If you do, the line is connected to the input port closest to the cursor's position.



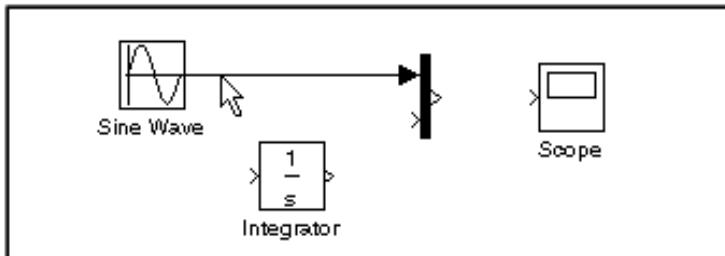
(1) 把鼠标指针移到第一个模块端口的任意位置，光标将变成十字形。

(2) 按下鼠标，拖动鼠标指针定位到第二个模块输入端口的位置。

(3) 释放鼠标，**simulink**用一个带箭头的实线信号线代替端口的符号，用来表示信号的流向。

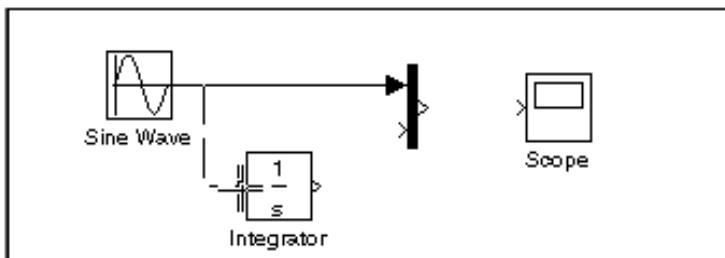
## ■ 在一条已有的信号线上引出另一条信号线

1. First, position the pointer *on the line* between the Sine Wave and the Mux block.



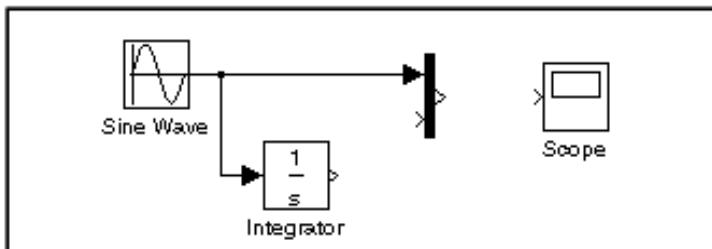
(1) 把鼠标指针移到这根信号线上的某个位置，这个位置就是引出新信号线的起始位置。

2. Press and hold down the **Ctrl** key (or click the right mouse button). Press the mouse button, then drag the pointer to the Integrator block's input port or over the Integrator block itself.

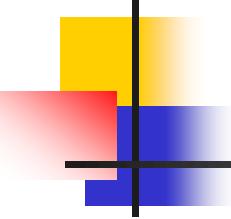


(2) 在按下**Ctrl**键的同时，按下鼠标，拖动鼠标到目标端口。

3. Release the mouse button. Simulink draws a line between the starting point and the Integrator block's input port.



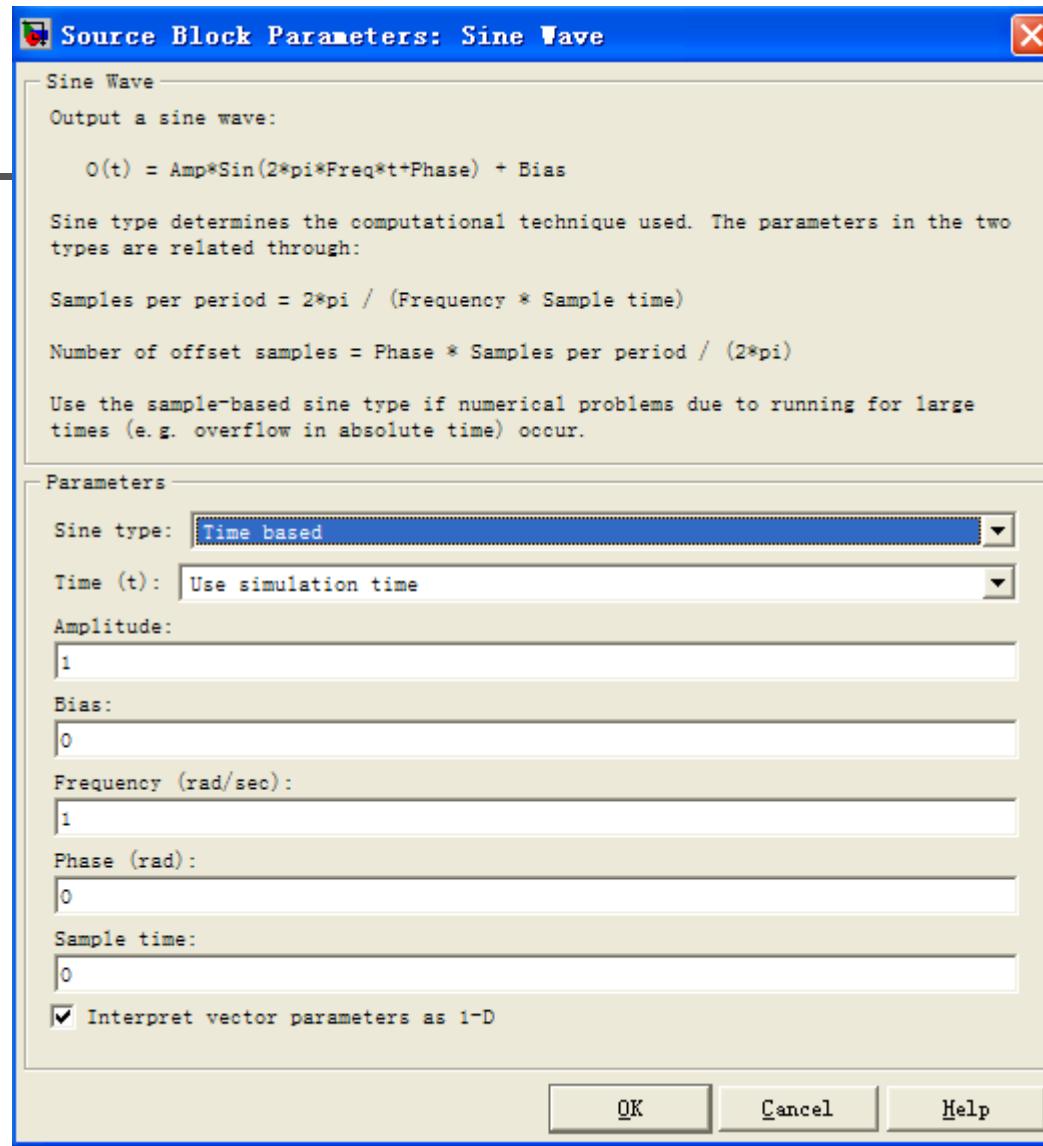
(2) 释放鼠标按钮和**Ctrl**键，那么**Simulink**就在起始位置和目标端口之间创建了一条新信号线。



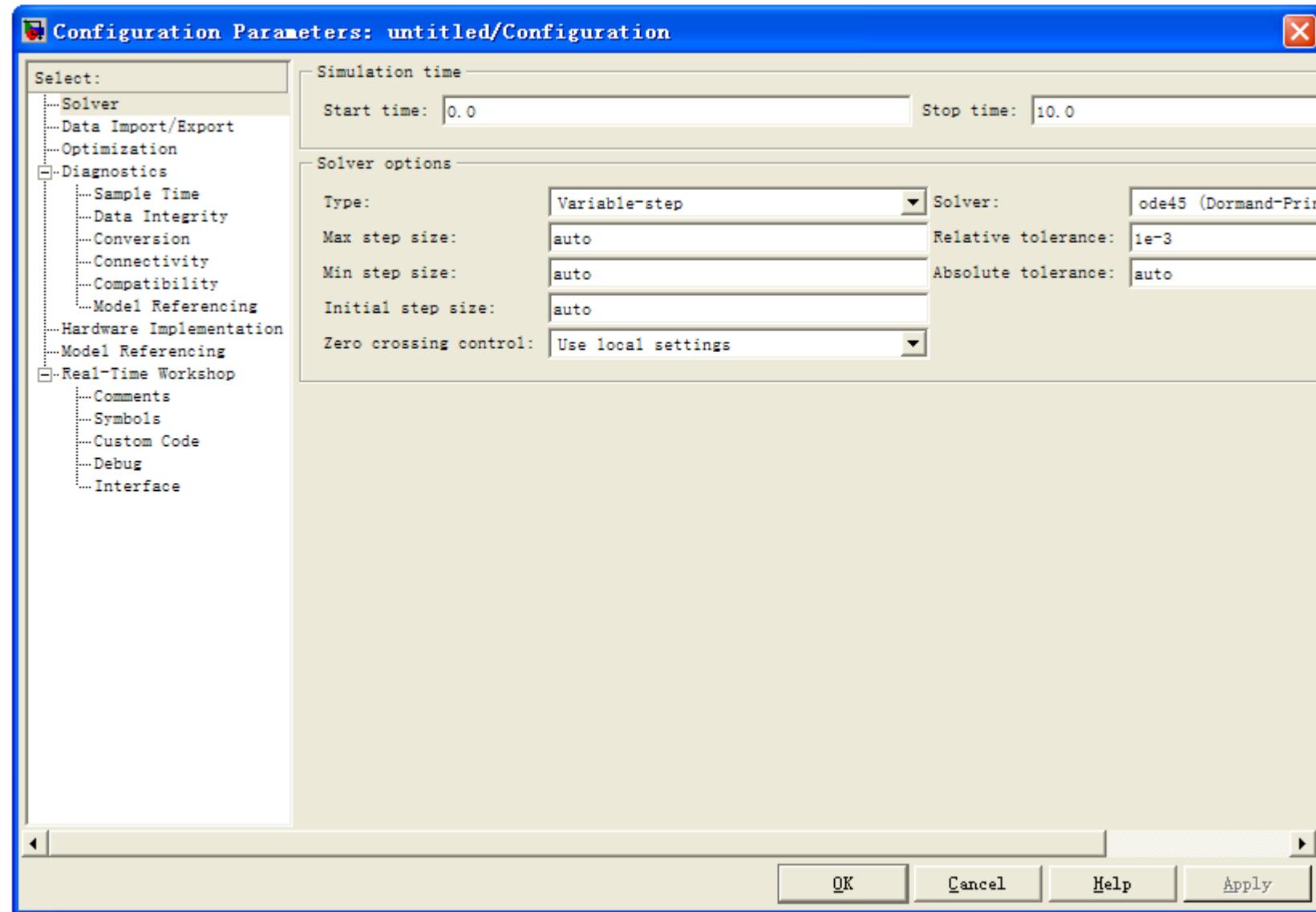
## 确定模型参数

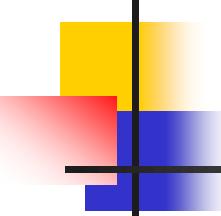
- 在模型窗口选中一个模块，用鼠标双击该模块，**Simulink**就打开模块基本属性对话框。
- 在模型窗口选择【**Simulation: Configuration parameters**】菜单，可进行仿真参数设置。

# ■ Sine wave 模块属性设置对话框



# ■ 仿真参数设置





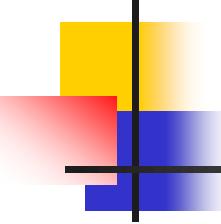
# 求解器(Solver)的设置

## (1) Simulation time (仿真时间设置)

- 修改仿真的开始和结束时间。

## (2) Solvers options (求解器选项设置)

- Solver:** Simulink模型仿真一般需要采用微分方程或微分方程组的数值解法，用户可以根据仿真模型的特点，选择最合适的求解方法；
- Type:** 选择可变步长或固定步长；
- 在可变步长中，有**Max step size**, **Min step size**, **Initial step size**
- Zero-crossing control** (零点穿越控制)
- Relative tolerance**, **Absolute tolerance** (容许误差控制)



# 仿真

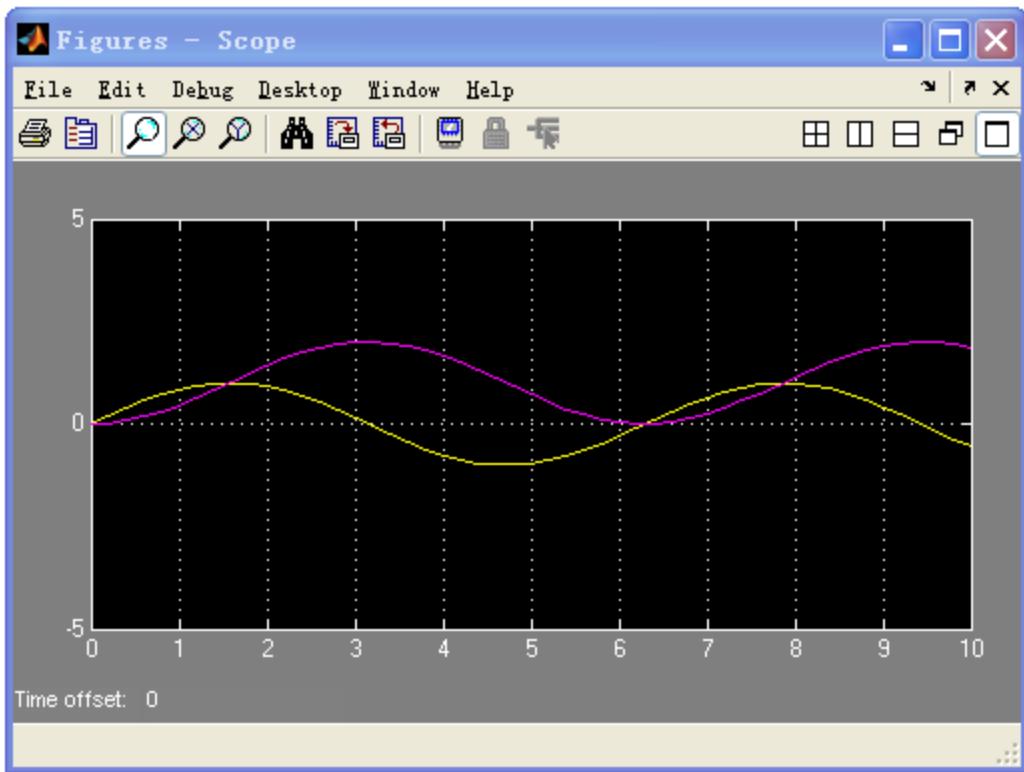
- 仿真过程的启动

在模型窗口选择 【Simulation】 → 【Start】

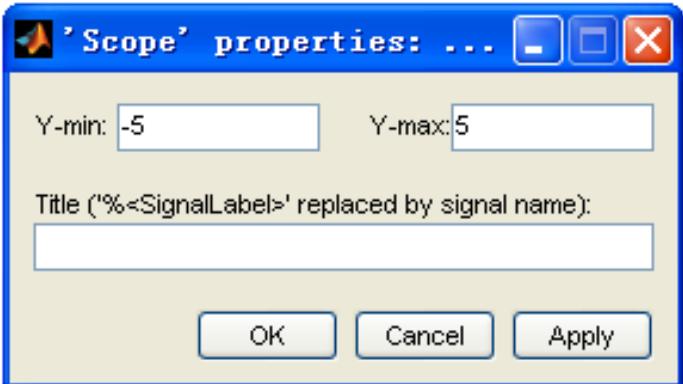
- 仿真过程的诊断

如果仿真过程出现错误，仿真一般会自动停止，并弹出一个诊断对话框显示错误的相关信息。

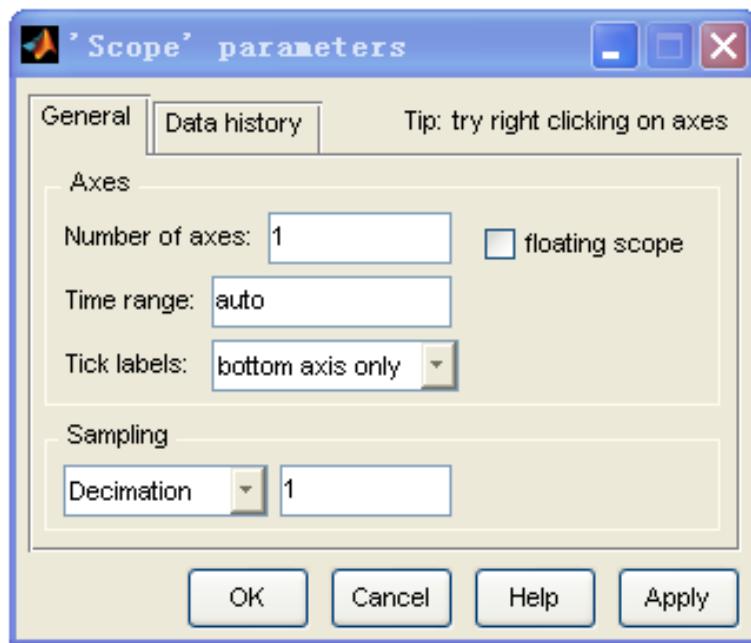
- **Message:** 错误类型，如模块错误或警告；
- **Source:** 发生错误的模块名称；
- **Fullpath:** 导致错误的对象的完整路径；
- **Summary:** 错误的简单说明；
- **Reported by :** 报告错误的组件。

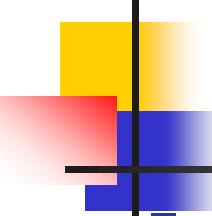


在示波器窗口单击鼠标右键，弹出：



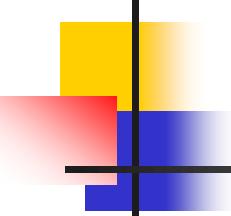
在示波器窗口单击工具按钮 ，弹出 **Scope** 模块的参数设置窗口：





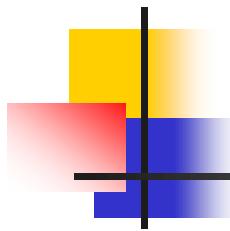
## 模块基本操作：

- 模块的选定 选多个模块时，按下Shift键，依次选定所需的模块，或者按住鼠标左键，拉虚线框
- 模块的移动 按住鼠标左键
- 改变模块的方向 单击**【Format】** → **【Flip Block】** 菜单项，可将模块左右镜像翻转，单击**【Format】** → **【Rotate Block】** 菜单项，可将模块旋转90°
- 复制模块



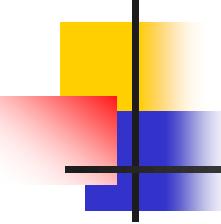
## 模块基本操作：

- 改变大小：选中模块，对模块出现的4个黑色标记进行拖曳即可。
- 模块命名：先用鼠标在需要更改的名称上单击一下，然后直接更改即可。名称在功能模块上的位置也可以变换180度，可以用**Format**菜单中的**Flip Name**来实现，也可以直接通过鼠标进行拖曳。**Hide Name**可以隐藏模块名称。



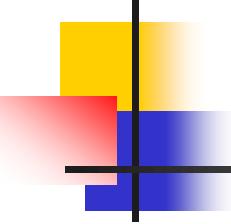
## 模块基本操作：

- 颜色设定： **Format**菜单中的**Foreground Color**可以改变模块的前景颜色，**Background Color**可以改变模块的背景颜色； 而模型窗口的颜色可以通过**Screen Color**来改变。



# 信号线操作

- **改变粗细：**线所以有粗细是因为线引出的信号可以是标量信号或向量信号，当选中**Format**菜单下的**Wide Vector Lines**时，线的粗细会根据线所引出的信号是标量还是向量而改变，如果信号为标量则为细线，若为向量则为粗线。选中**Vector Line Widths**则可以显示出向量引出线的宽度，即向量信号由多少个单一信号合成
- **设定标签：**只要在线上双击鼠标，即可输入该线的说明标签。也可以通过选中线，然后打开**Edit**菜单下的**Signal Properties**进行设定，其中**signal name**属性的作用是标明信号的名称，设置这个名称反映在模型上的直接效果就是与该信号有关的端口相连的所有直线附近都会出现写有信号名称的标签。

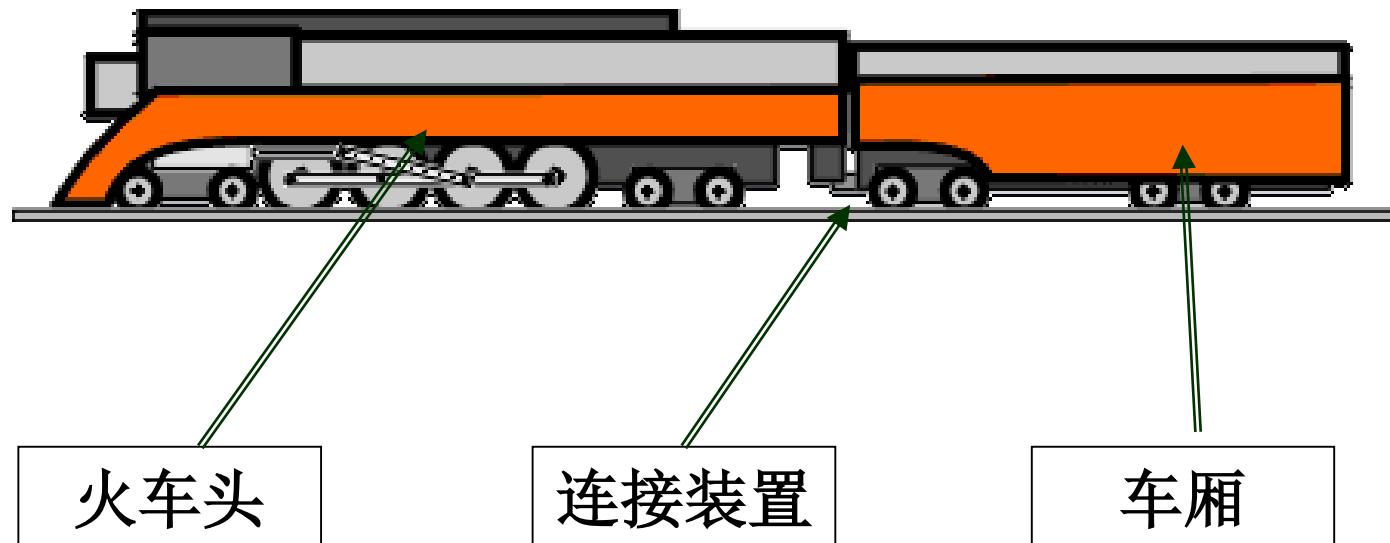


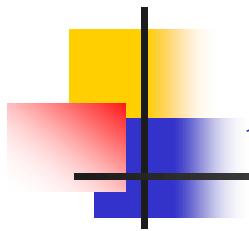
# 信号线操作

- 线的折弯：按住**Shift**键，再用鼠标在要折弯的线处单击一下，就会出现圆圈，表示折点，利用折点就可以改变线的形状。
- 线的分支：按住鼠标右键，在需要分支的地方拉出即可以。或者按住**Ctrl**键，并在要建立分支的地方用鼠标拉出即可。

# 火车系统仿真

## ■ 物理系统：



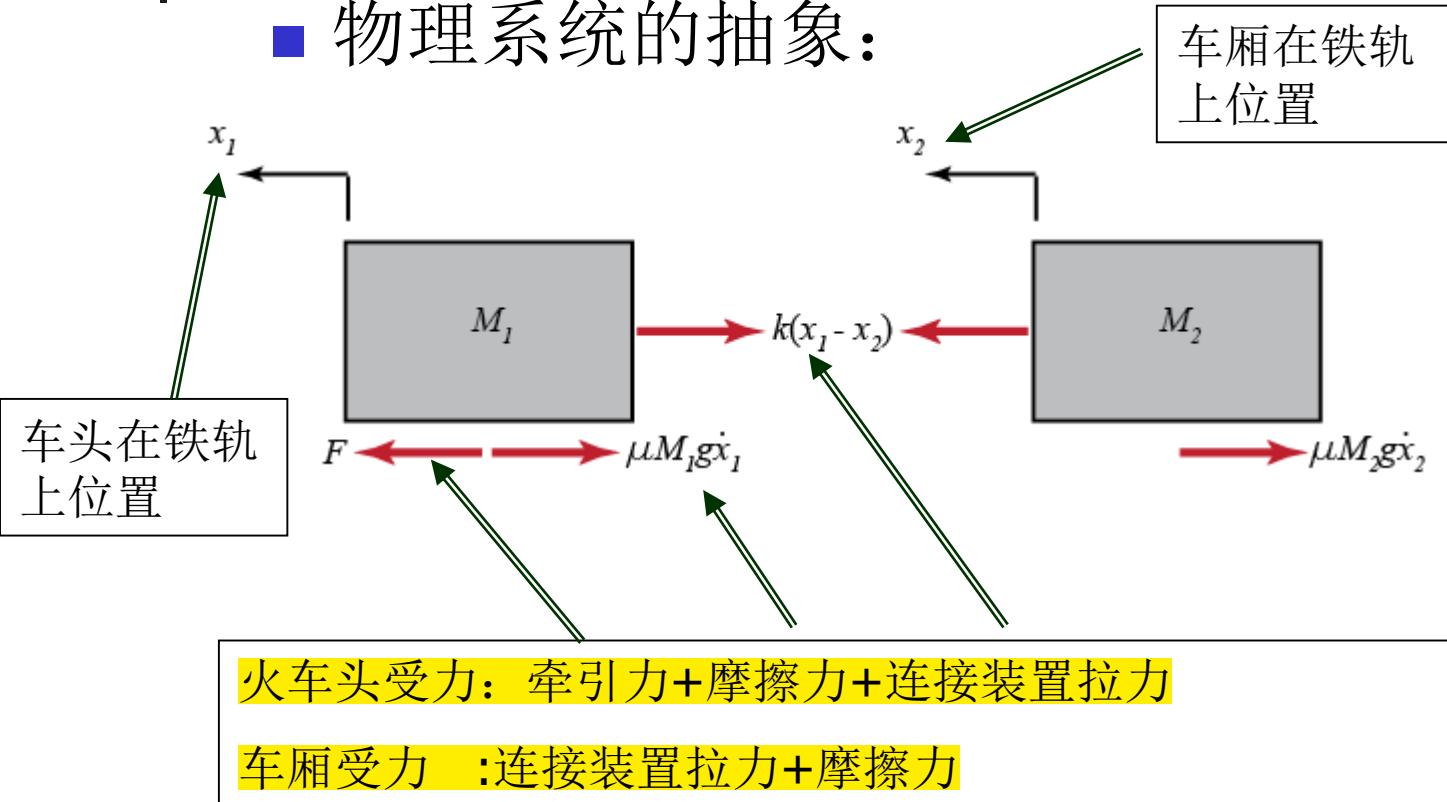


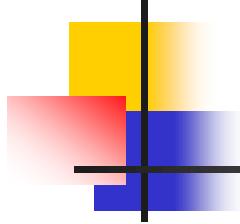
# 火车系统仿真

- 仿真目标：车速与火车头牵引力 $F$ 的关系
- 设火车头质量为 $M_1$ , 车厢质量 $M_2$ ;
- 连接装置设为弹簧，弹性系数为 $k$
- 希望 $F$ 改变时，火车速度能尽快改变并与 $F$ 成正比。

# 火车系统仿真

## ■ 物理系统的抽象：





# 火车系统仿真

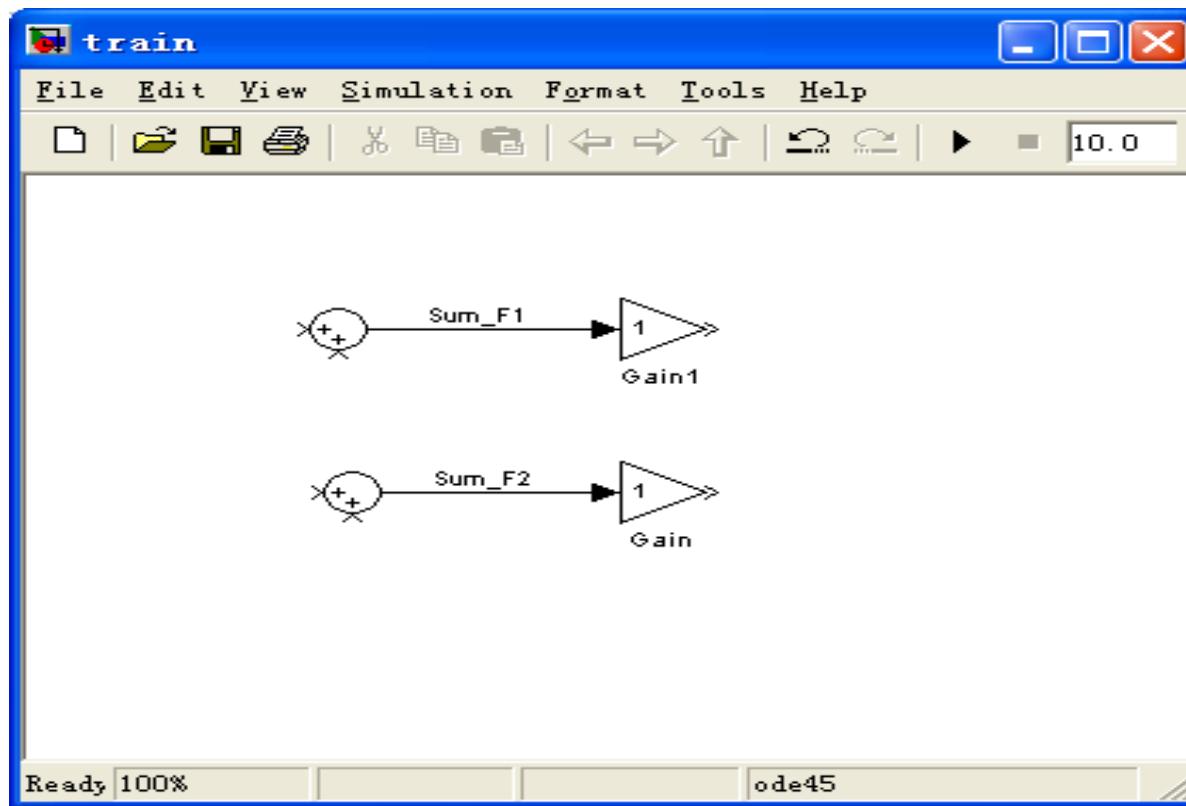
- 物理系统的数学模型：  
牛顿第二定律： **F = ma**

$$\Sigma F_1 = F - k(x_1 - x_2) - \mu M_1 g \dot{x}_1 = M_1 \ddot{x}_1$$

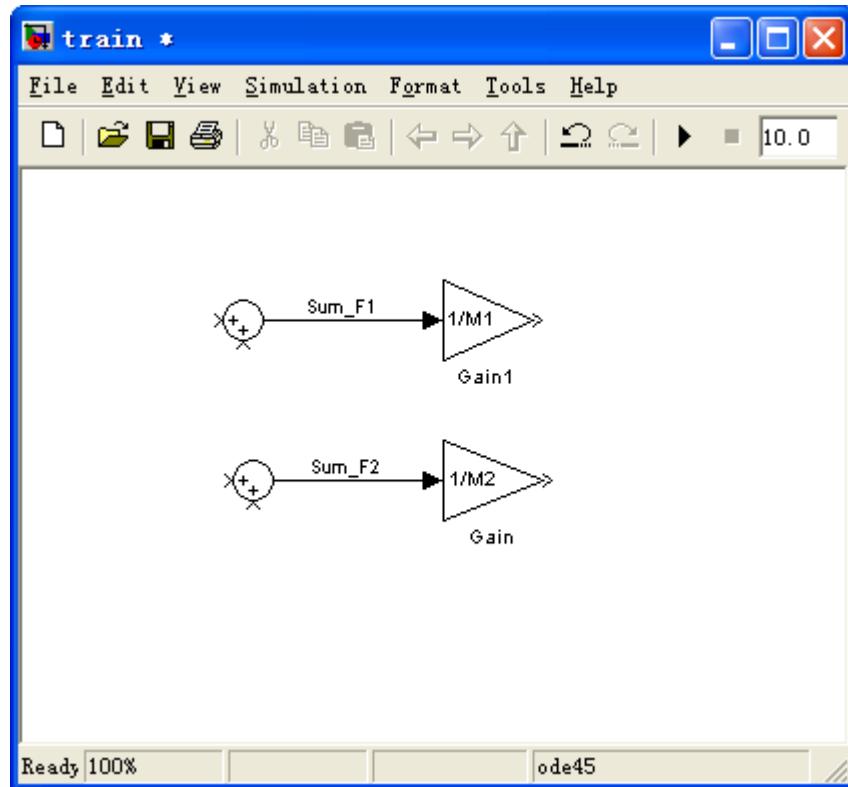
$$\Sigma F_2 = k(x_1 - x_2) - \mu M_2 g \dot{x}_2 = M_2 \ddot{x}_2$$

# 火车系统仿真

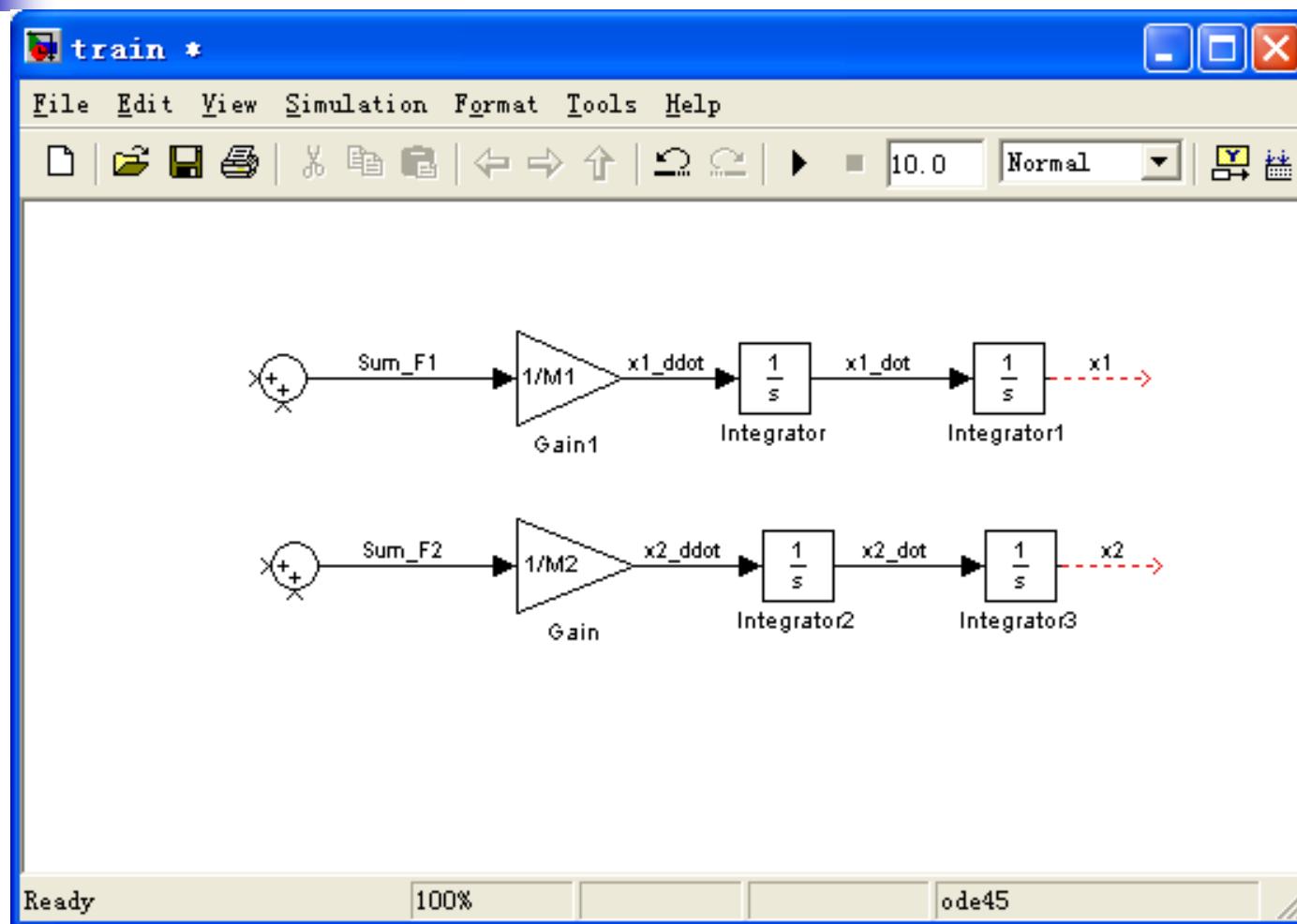
## ■ 构建simulink模型



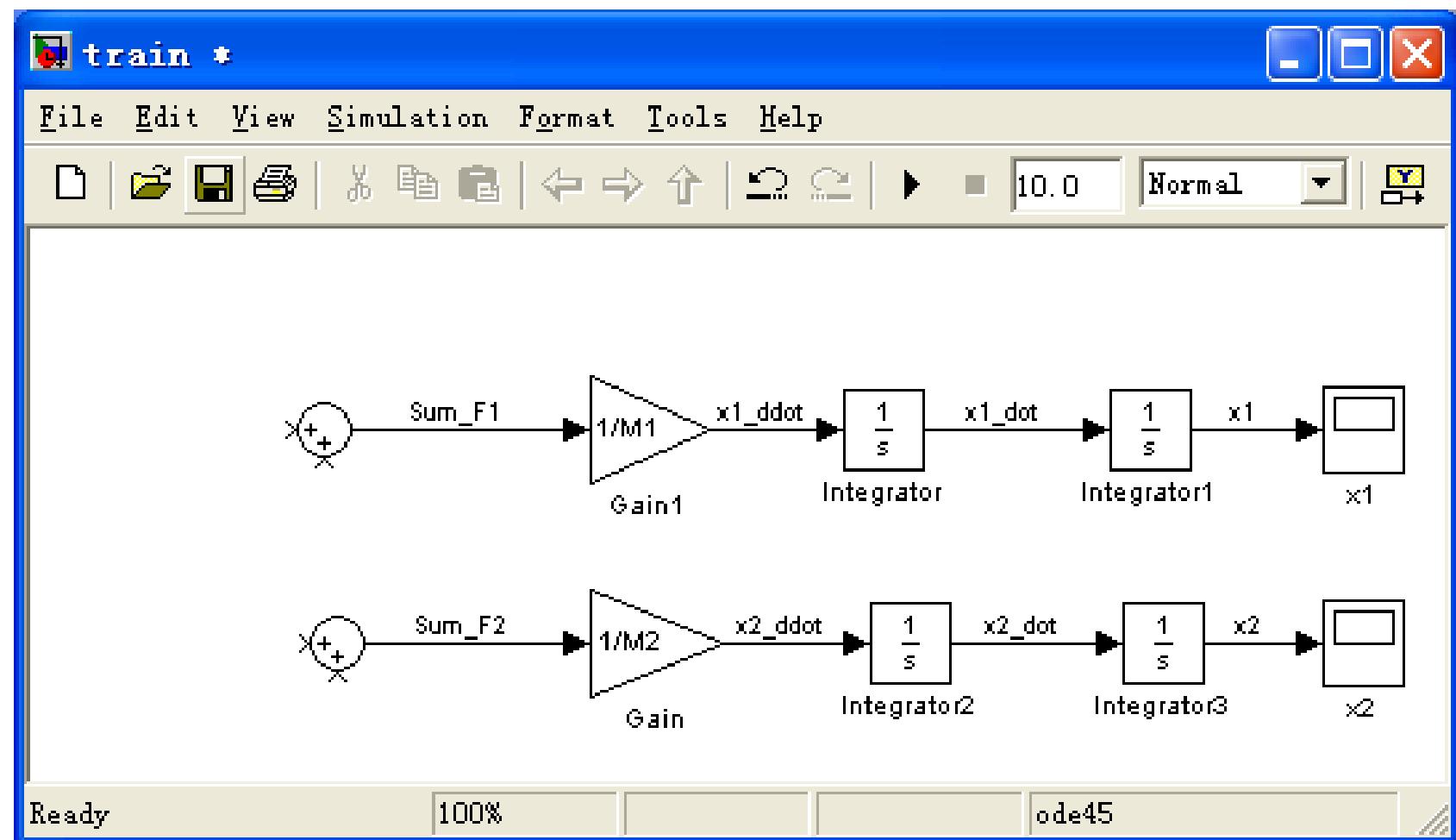
# 火车系统仿真



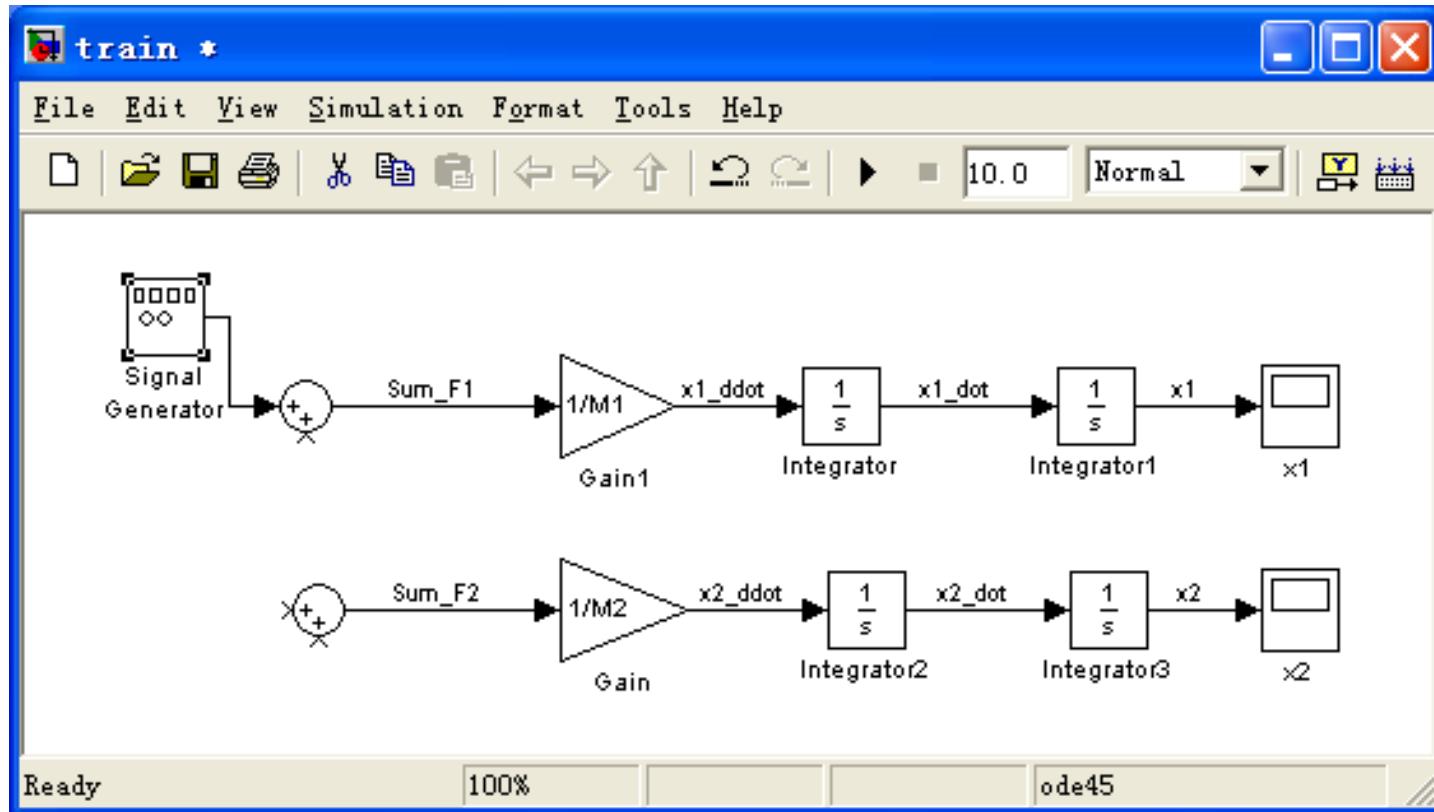
# 火车系统仿真



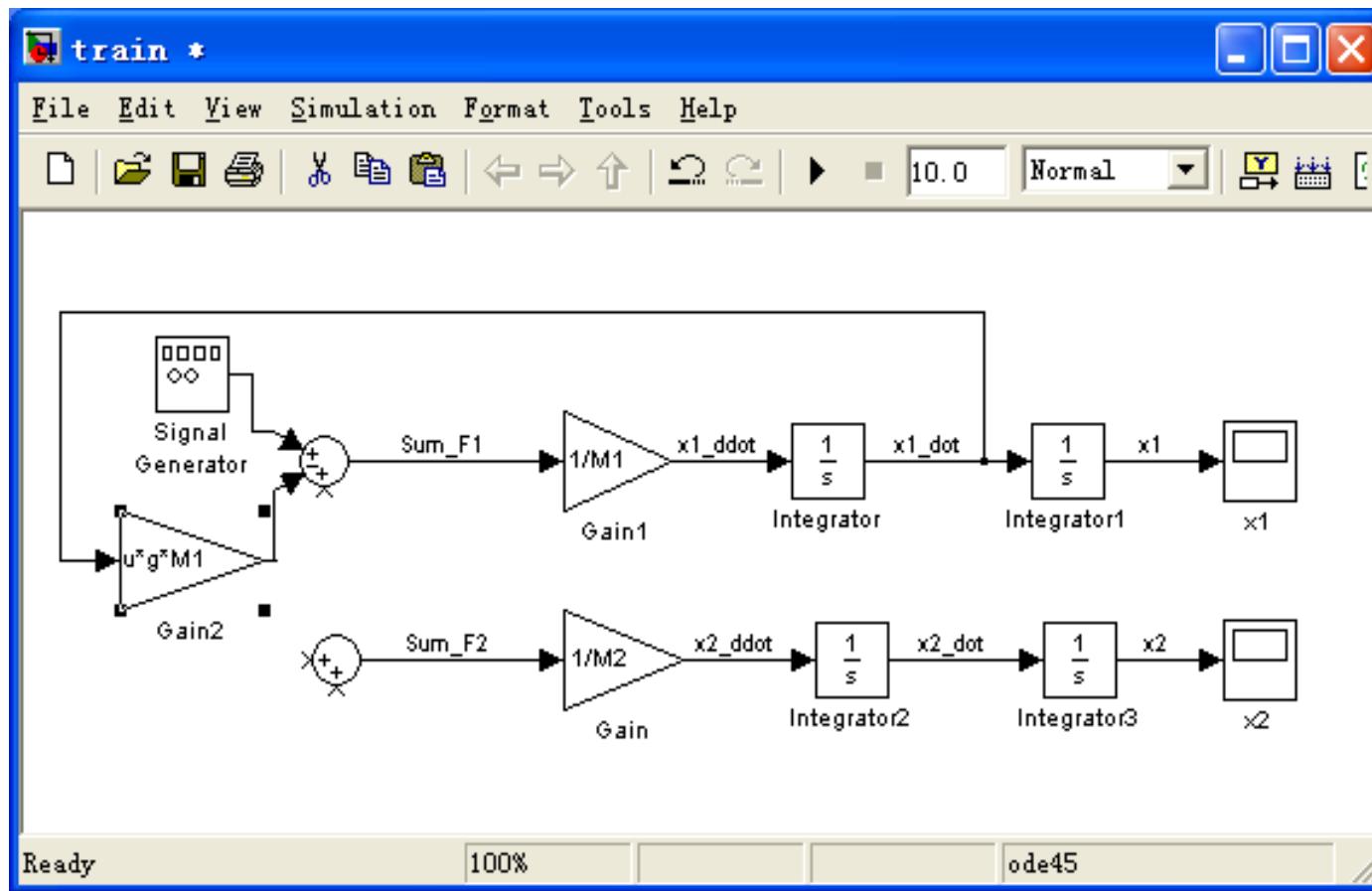
# 火车系统仿真



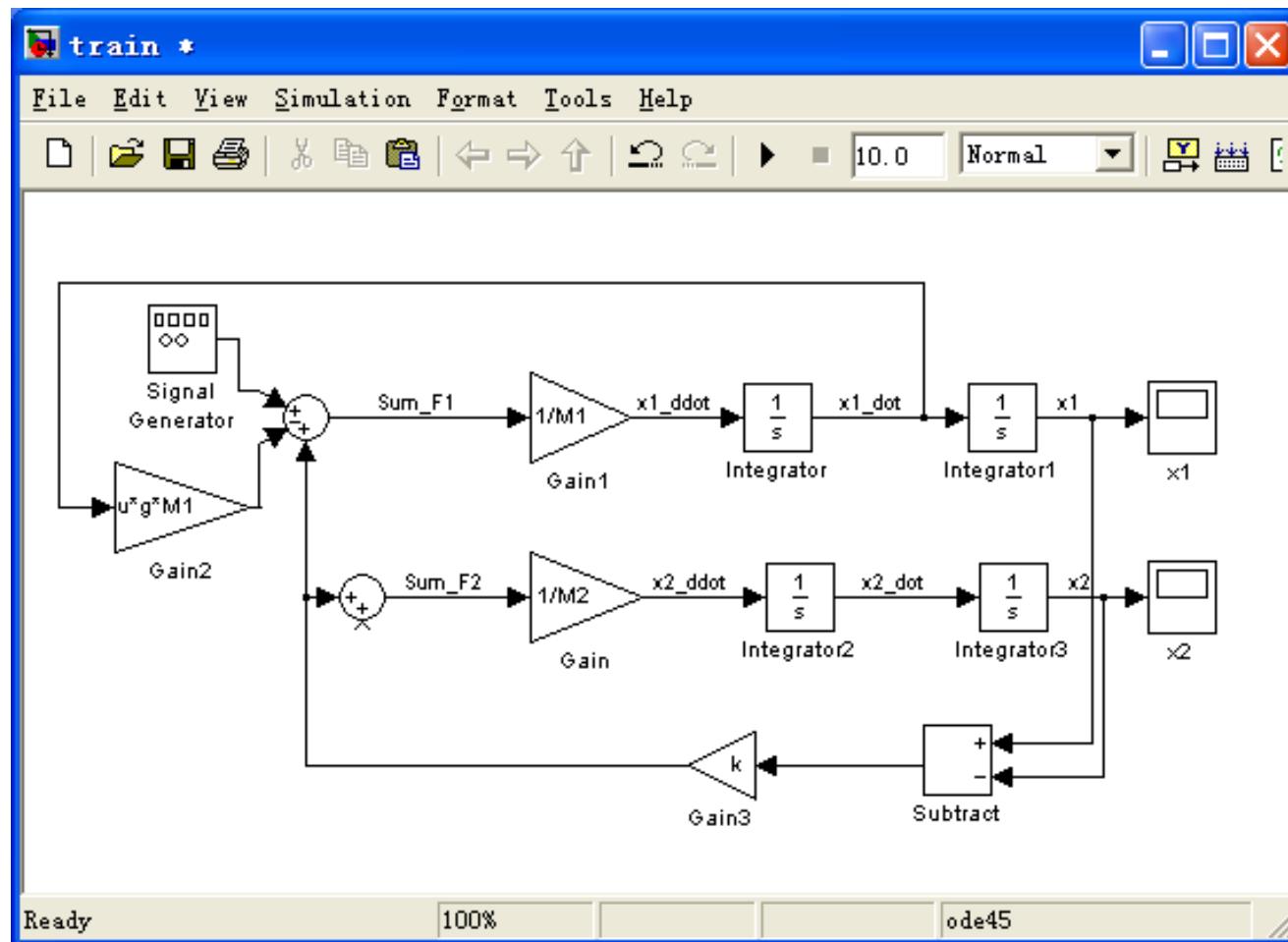
# 火车系统仿真



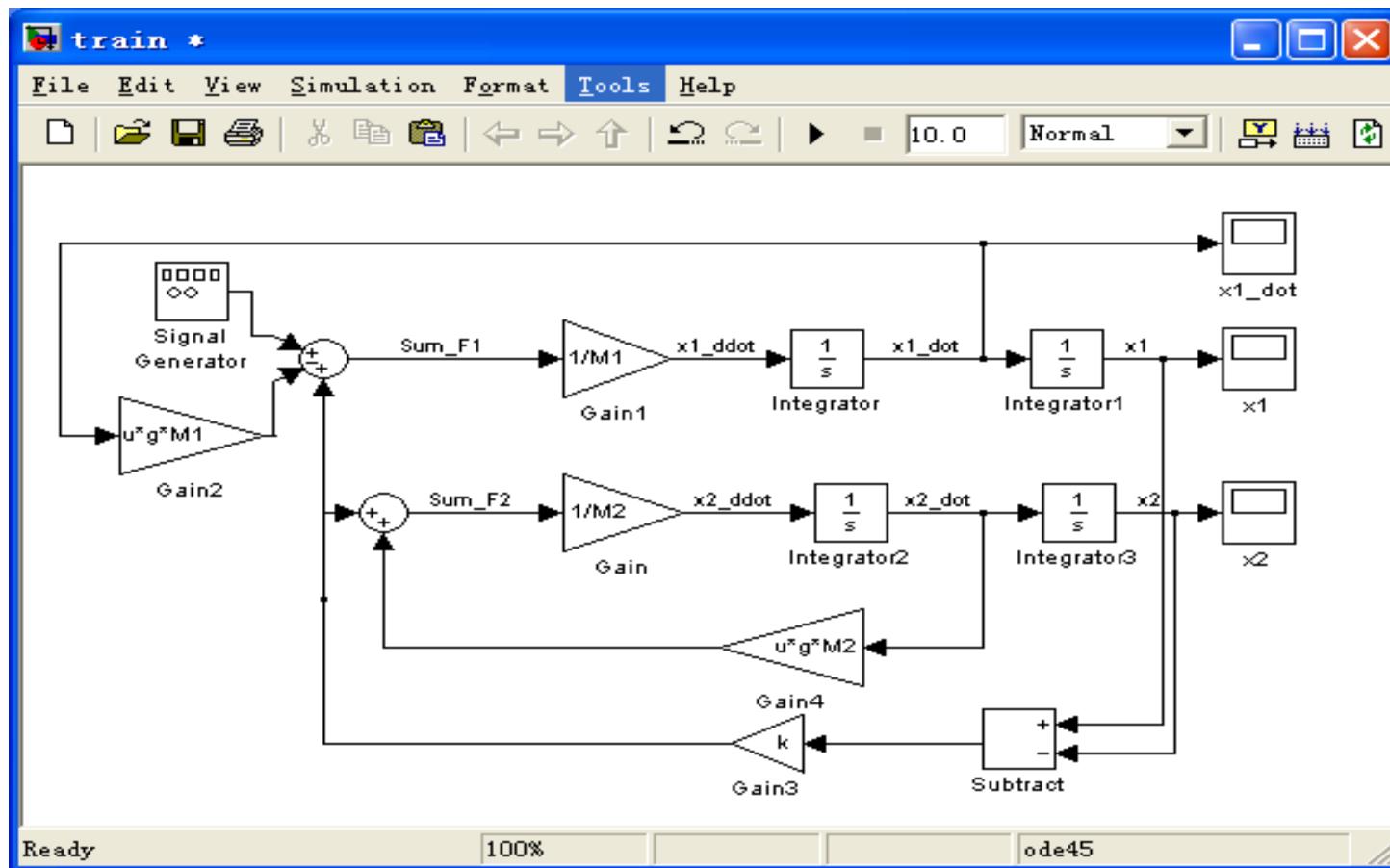
# 火车系统仿真

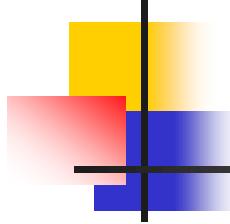


# 火车系统仿真



# 火车系统仿真





# 火车系统仿真

- 在命令行窗口输入相关参数

```
M1=1;
```

```
M2=0.5;
```

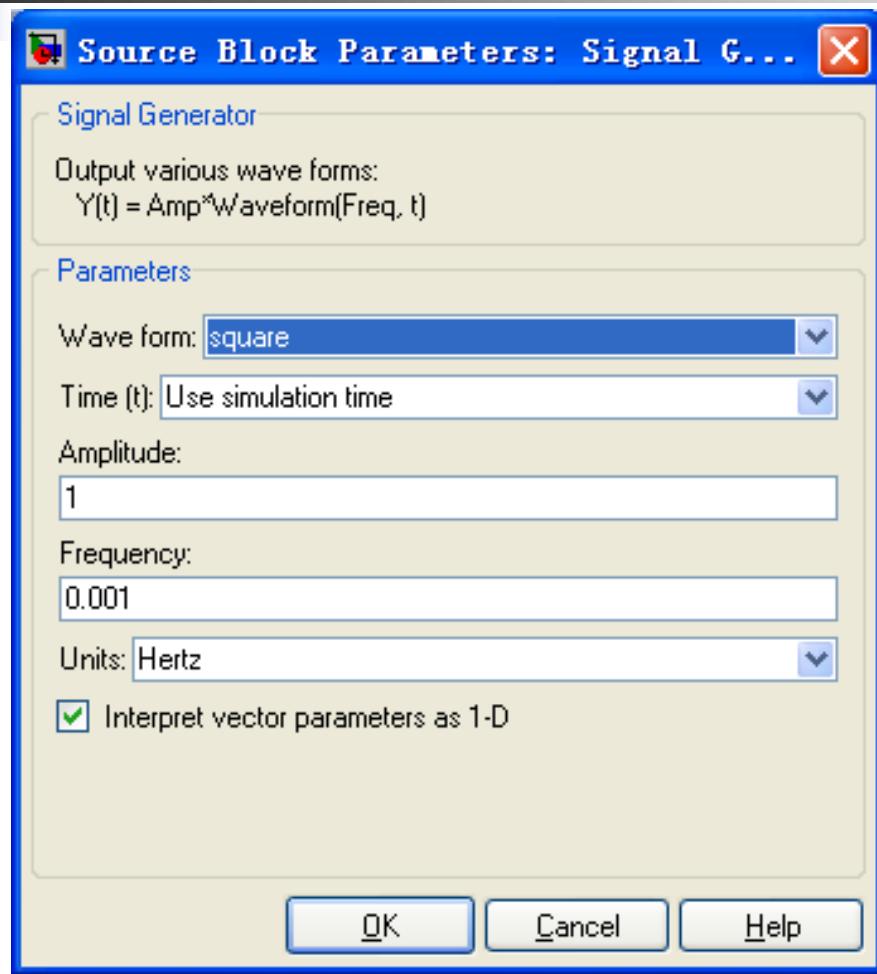
```
k=1;
```

```
F = 1;
```

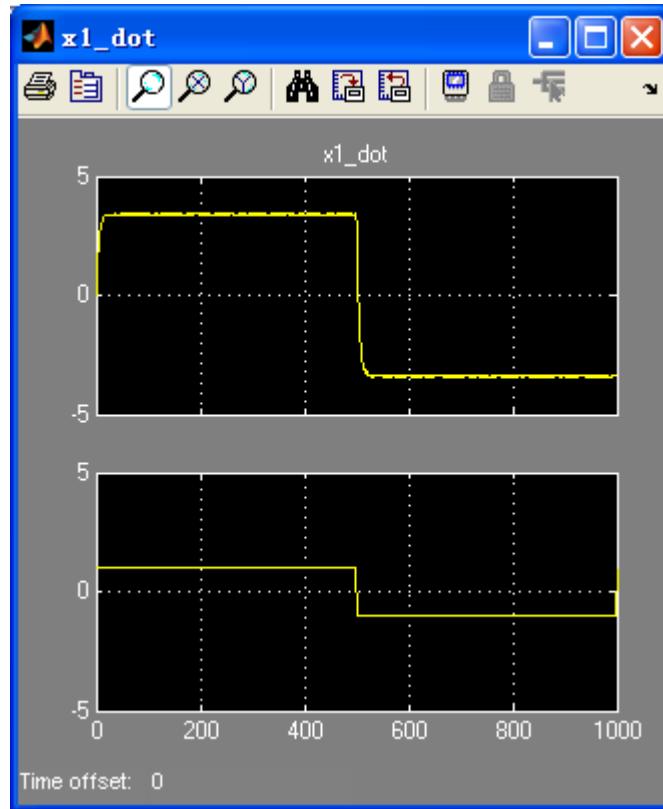
```
mu = 0.02;
```

```
g = 9.8;
```

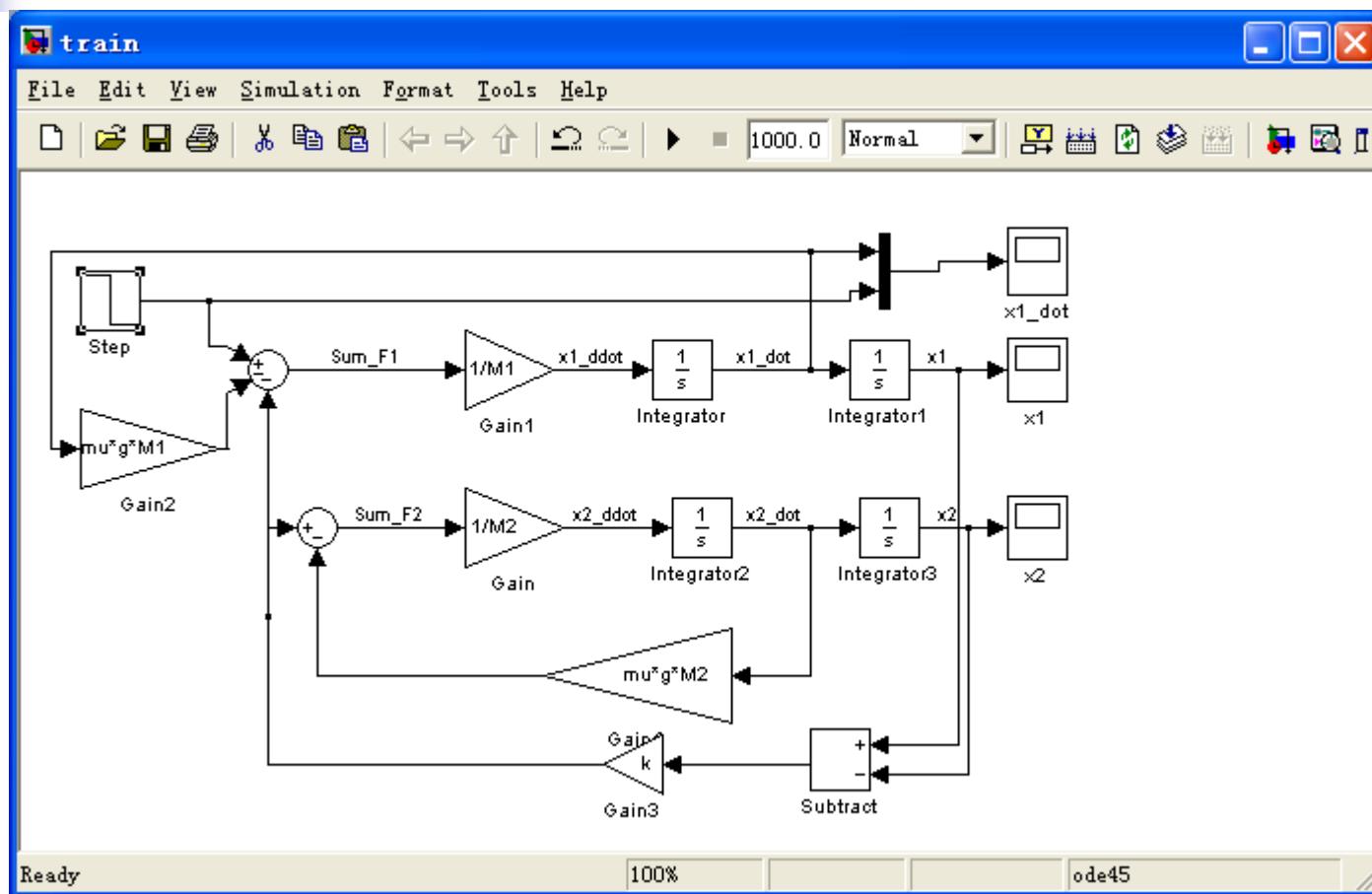
# 火车系统仿真

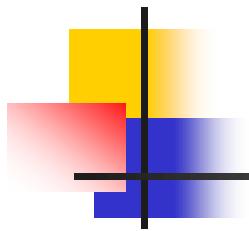


# 火车系统仿真



# 火车系统仿真





谢 谢 !