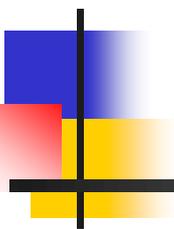


# Matlab编程与应用

## 第二讲



中国科技大学信息学院

陆伟

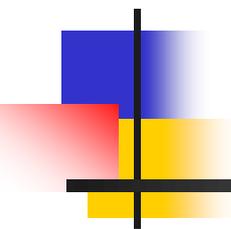
luwei@ustc.edu.cn



# 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

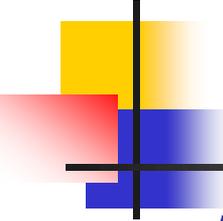


# 数组与矩阵

---

## ■ 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

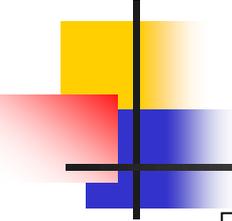
---

- 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:

$$z = [1+2i \quad 7-3i \quad 3+4i; \quad 6-2i \quad 9i \quad 4+7i]$$

$z =$

$$\begin{array}{cc} 1.0000 + 2.0000i & 7.0000 - 3.0000i \\ 3.0000 + 4.0000i & \\ 6.0000 - 2.0000i & 0 + 9.0000i \\ 4.0000 + 7.0000i & \end{array}$$

$z'$

$ans =$

$$\begin{array}{cc} 1.0000 - 2.0000i & 6.0000 + 2.0000i \\ 7.0000 + 3.0000i & 0 - 9.0000i \\ 3.0000 - 4.0000i & 4.0000 - 7.0000i \end{array}$$

**A'** 是共轭转置, 及 **A<sup>H</sup>** !

# 矩阵转置

例:

$$Z = [1+2i \quad 7-3i \quad 3+4i; \quad 6-2i \quad 9i \quad 4+7i]$$

$Z =$

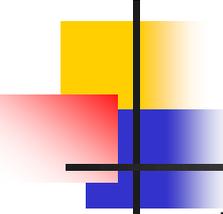
$$\begin{array}{cc} 1.0000 + 2.0000i & 7.0000 - 3.0000i \\ 3.0000 + 4.0000i & \\ 6.0000 - 2.0000i & 0 + 9.0000i \\ 4.0000 + 7.0000i & \end{array}$$

$Z.'$

$ans =$

$$\begin{array}{cc} 1.0000 + 2.0000i & 6.0000 - 2.0000i \\ 7.0000 - 3.0000i & 0 + 9.0000i \\ 3.0000 + 4.0000i & 4.0000 + 7.0000i \end{array}$$

**A.'** 得到  $A^T$  !



# 矩阵指数运算

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

```
X = A^2
```

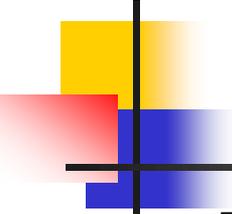
```
X =
```

```
     3     6    10
     6    14    25
    10    25    46
```

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

```
Y =
```

```
     1     1     1
     1     4     9
     1     9    36
```

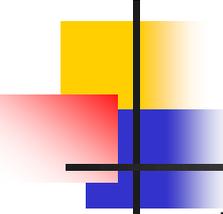


# 矩阵幂运算

**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

---

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



# 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

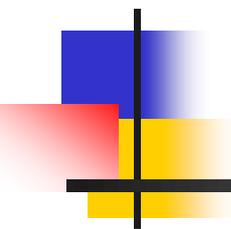
---

- 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:

---

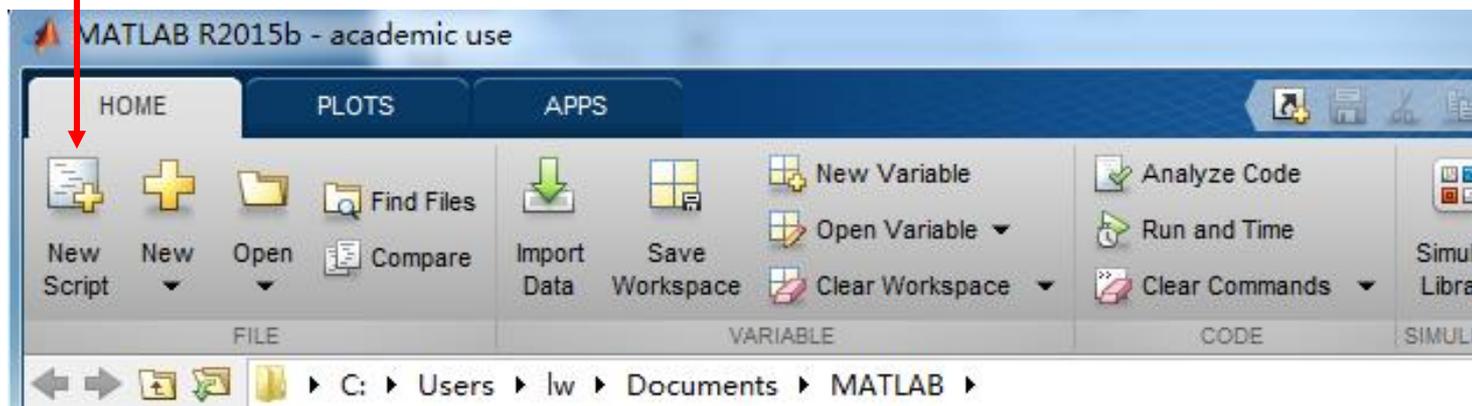
# 脚本与函数

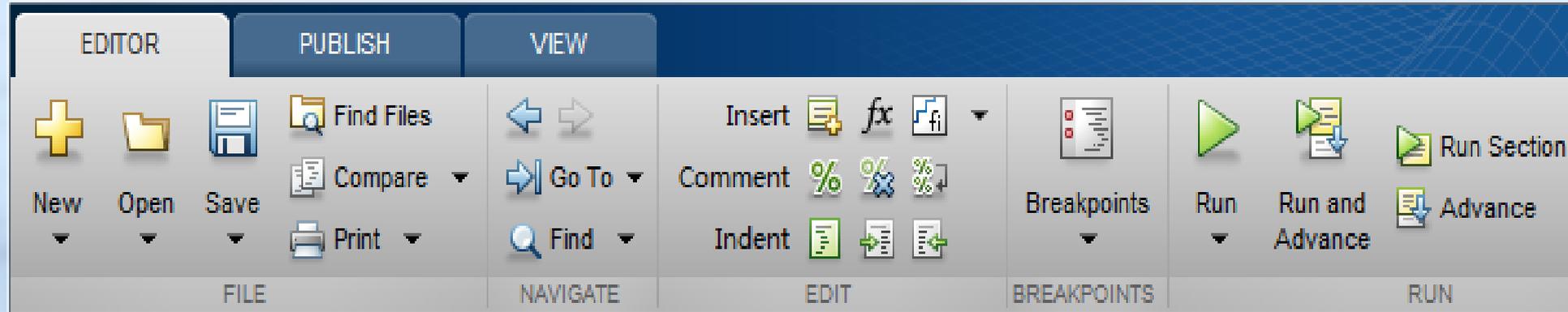
# matlab脚本文件

## ■ 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* +
1 %This is a demo script m file
2 %           2017.6.24
3
4 fs = 100; %采样频率
5 f0 = 2;
6 t = 0: 1/fs : 5;
7 x = exp(-t).*cos(2*pi*2*t); %信号
8 plot(t,x);
```

Help信息

注释

Breakpoint  
(调试断点)



# 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.

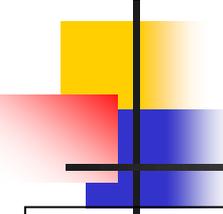


# input语句

---

- 例：求一元二次方程 $ax^2 + bx + c = 0$ 的根

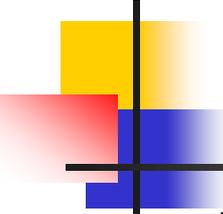
```
a=input('a=?');  
b=input('b=?');  
c=input('c=?');  
d=b*b-4*a*c;  
x1=(-b+sqrt(d))/(2*a)  
x2=(-b-sqrt(d))/(2*a)
```



# 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
```



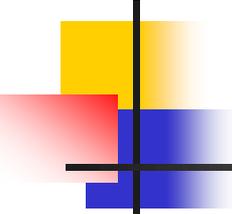
## 流程控制—for语句

```
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  $y = 1 + \frac{1}{3} + \frac{1}{5} + \dots + \frac{1}{2n-1}$ ，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
```

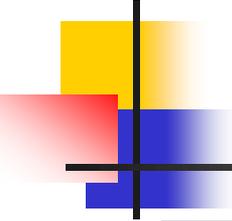
```
if 条件表达式  
    执行语句  
elseif  
    执行语句  
else  
    执行语句  
end
```



## 流程控制—if语句

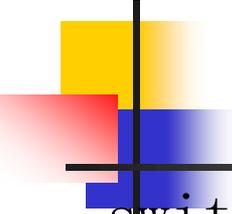
---

- 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—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。

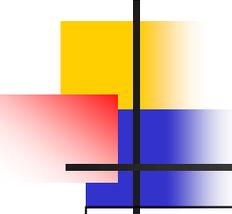


# 自定义函数

---

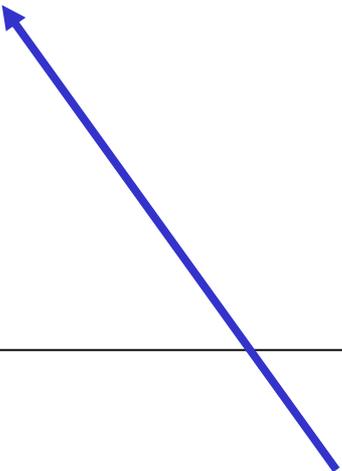
例： 建立一个函数文件计算  $\sin(x^2)$

```
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

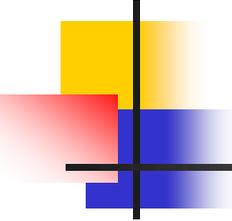
---

Script

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

function

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



# 自定义函数

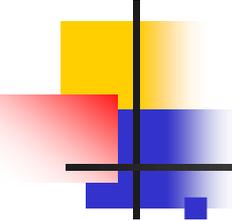
---

## ■ 练习:

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

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

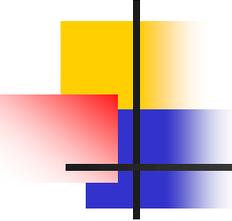
最终  $y = ?$



# 自定义函数

确定输入、输出变量的数目 `nargin, nargsout`

```
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 nargsout == 0  
    plot(x, y)  
else  
    x0 = x;  
    y0 = y;  
end
```

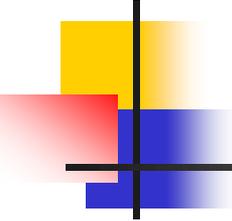


## 自定义函数

---

- 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。

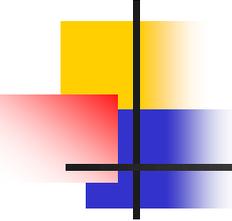
```
>> result = my1stfunc(3)
```



# 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

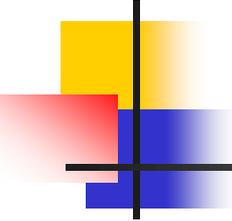


# 子函数

- 例：创建一个函数，输入两个数，输出两个数加、减后的结果

```
function [r1t_add,r1t_sub]=sfuncdemo(x,y)
%主函数
    r1t_add = add(x,y);
    r1t_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 (@)

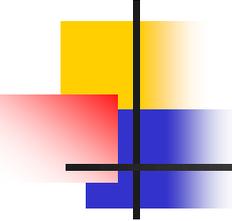
变量名=@(输入参数列表)运算表达式

```
例: >> sqr = @(x) x.^2; %创建
```

```
>> a = sqr(3) %调用
```

```
例: >> ln = @(x) log(x); %创建
```

```
>> a = ln(3) %调用
```



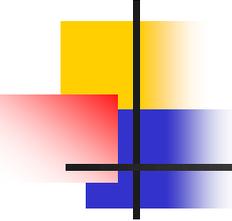
## 函数句柄 (@)

---

- 可以为matlab内建函数创建句柄。

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

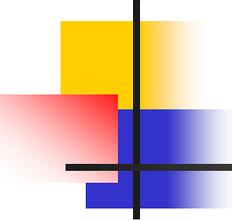
- 可以为用M文件创建的自定义函数创建句柄



## 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');
```

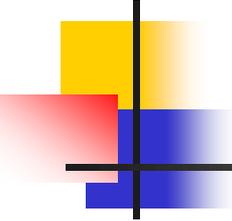
```
>>f(2,3)
```

```
ans = 5
```

```
>> Fofx=inline('x.^2*cos(a*x) -  
b', 'x', 'a', 'b');
```

```
>> g= Fofx([pi/3 pi/3.5],4,1)
```

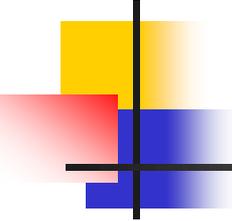
```
ans= -1.5483 -1.7259
```



# 函数调试

---

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



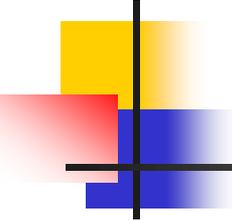
# 程序优化

---

- 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
```



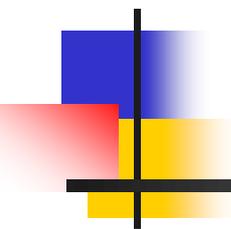
# 程序优化

---

- 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
```

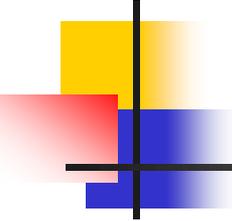
```
clear
tic
y = zeros(1,1e8);
for k = 2:1e8
    y(k) = y(k-1)+1;
end
toc
```



Part 3:

---

常用信号产生



# 常用信号产生

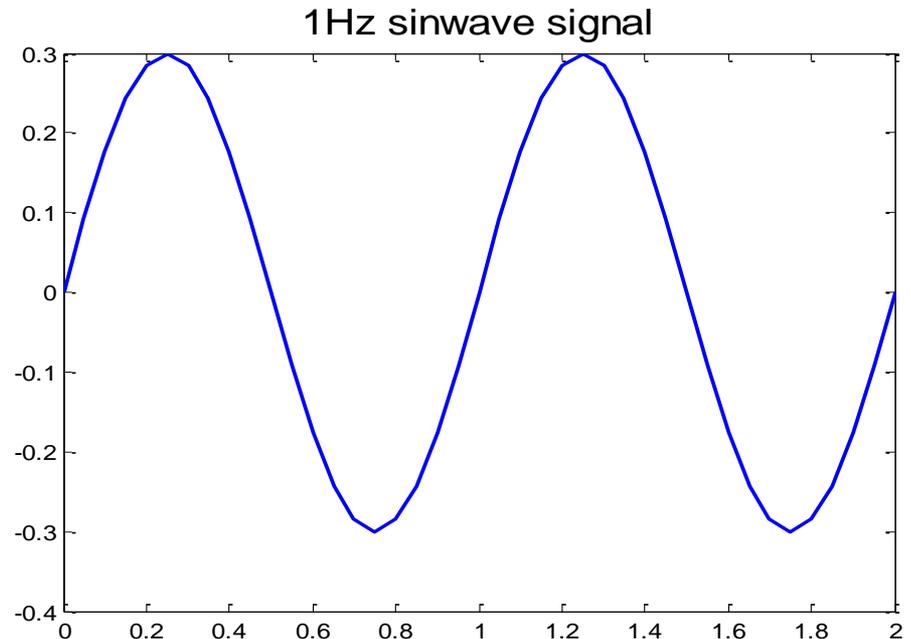
---

# 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)
```

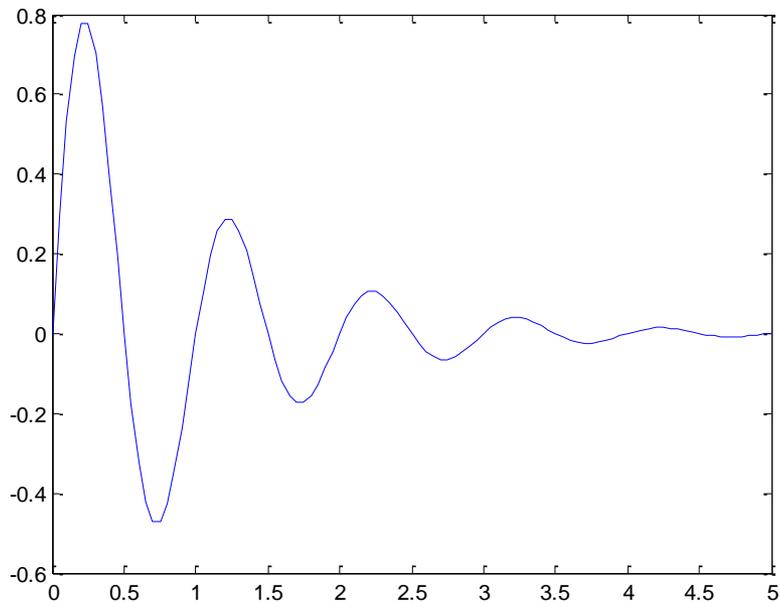


# 指数衰减的正弦信号

- 产生频率为1Hz的幅度呈指数衰减的正弦信号

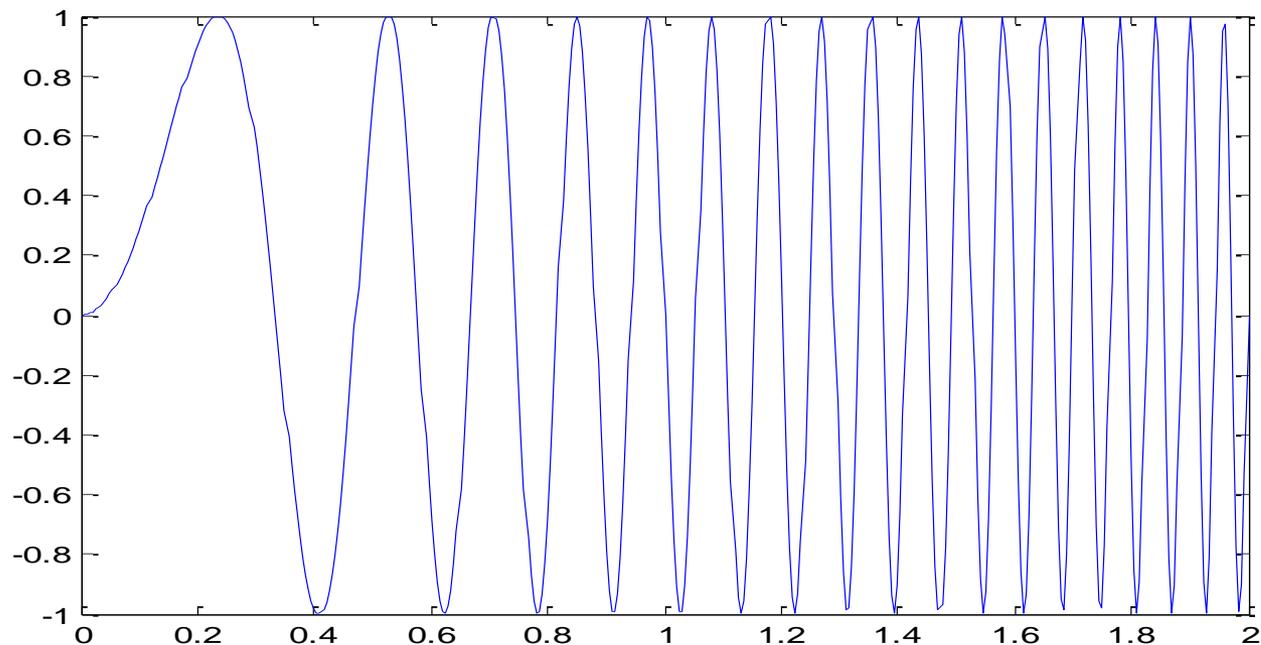
$$x(t) = e^{-t} \sin(2\pi f_0 t) \quad f_0 = 1\text{Hz}$$

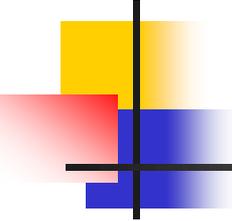
```
f0 = 1;  
fs = 20;  
t = 0:1/fs:5;  
x = exp(-t) .* sin(2*pi*t);  
plot(t,x)
```



# 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

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

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

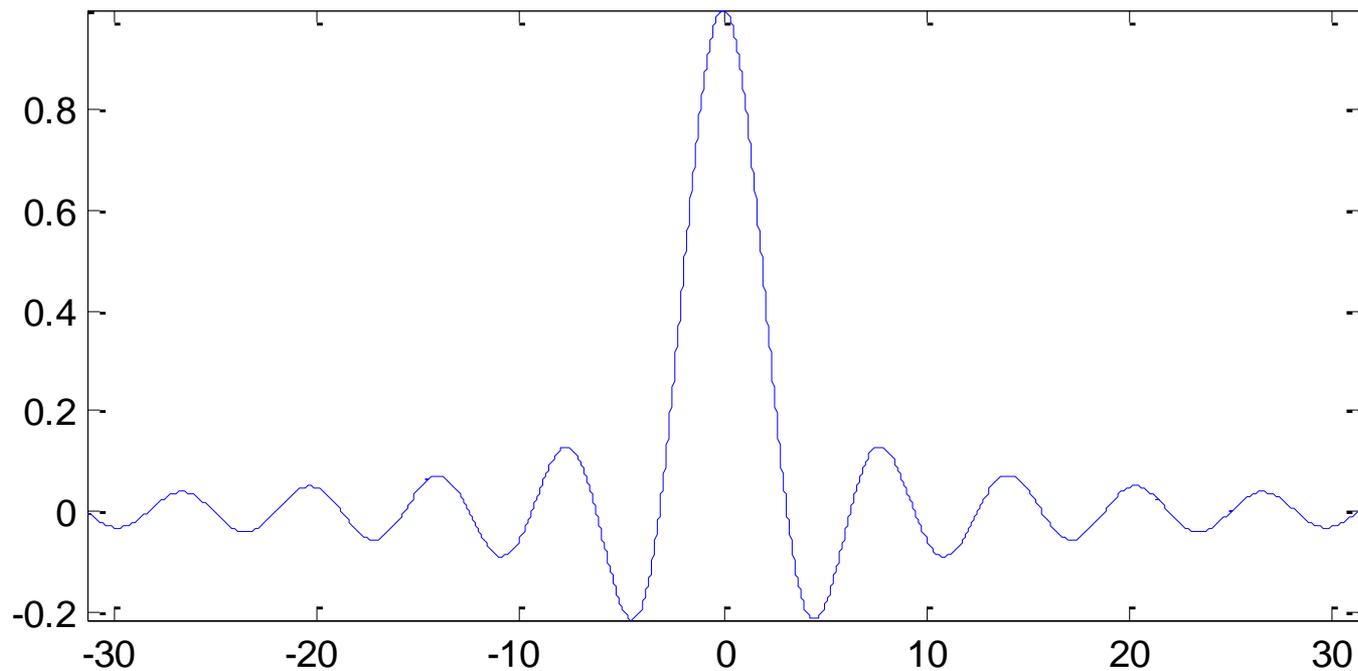
$f_0$  : 初始  $t=0$  时刻的瞬时频率;

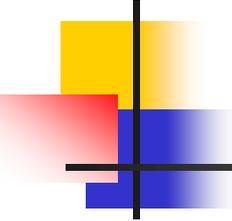
$f_1$  :  $t = t_1$  时刻的瞬时频率;

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

# Sinc信号

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





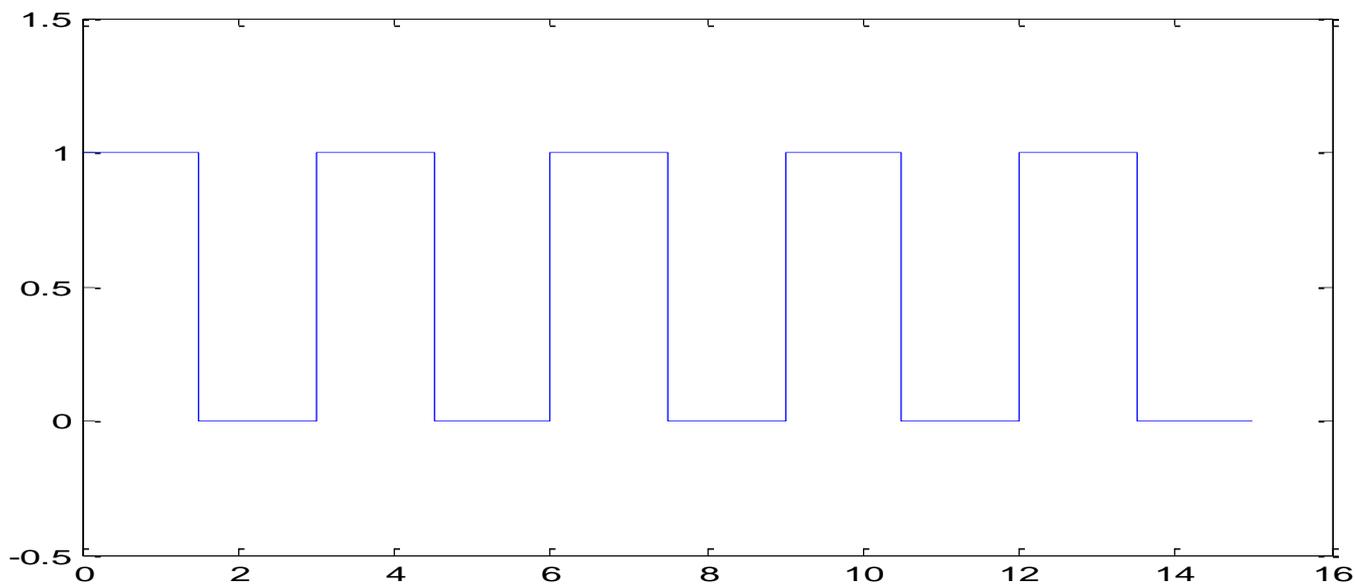
# Sinc信号

---

```
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。

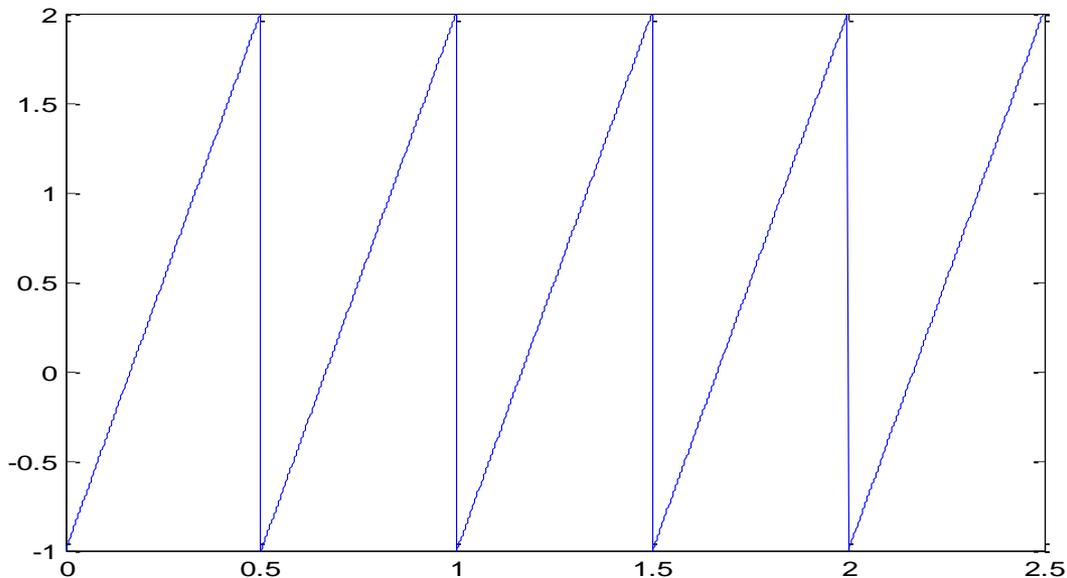


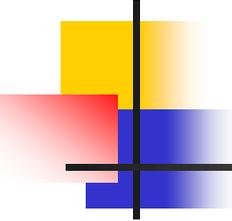
# 连续周期信号产生

```
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 F o u r i e r 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。

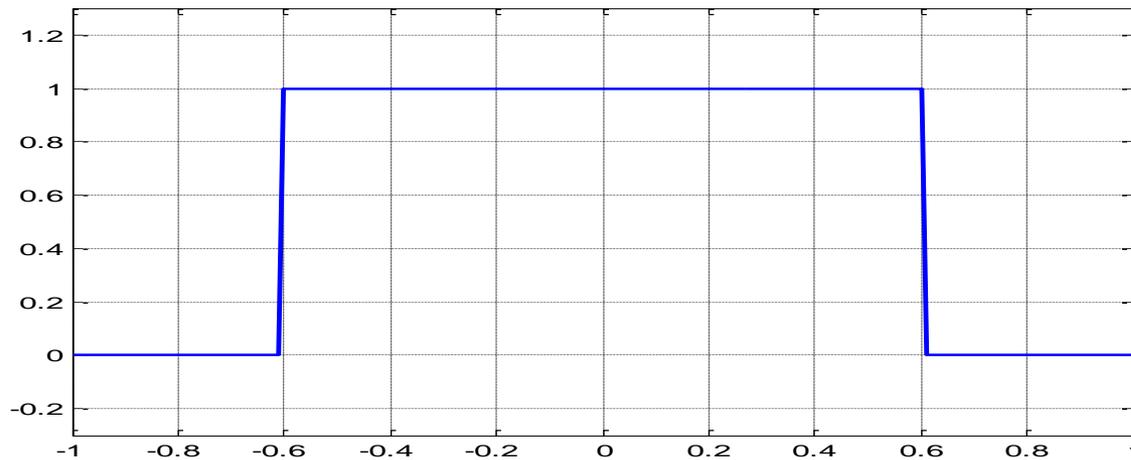
$$x(t) = \sum_{k=-\infty}^{+\infty} a_k e^{jk\omega_0 t} = a_0 + \sum_{k=1}^{+\infty} 2|a_k| \cos(k\omega_k t + \theta_k)$$

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

# 连续时间周期信号的 F o u r i e r 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)
```

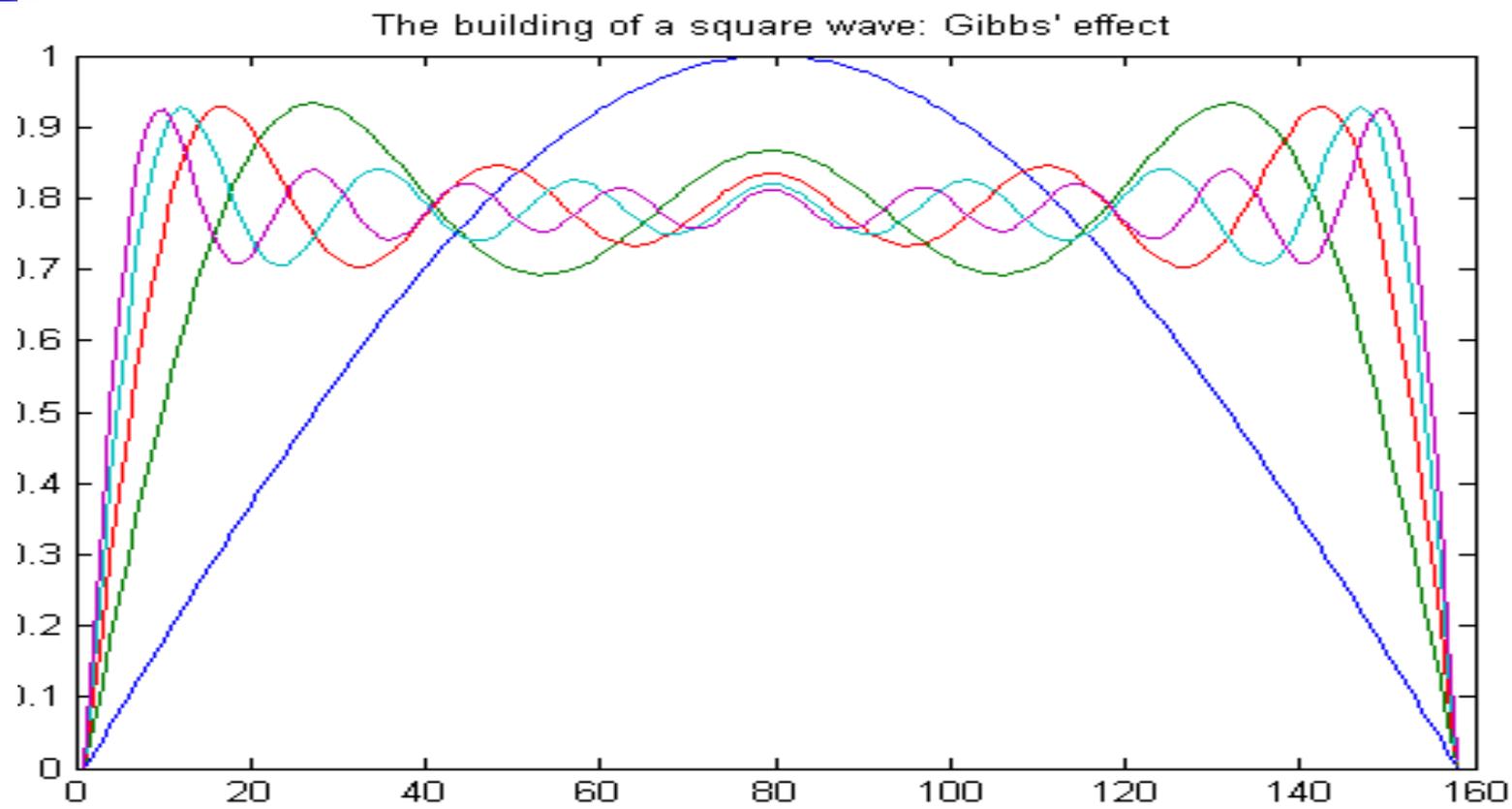
```
axis([-1 1,-0.3 1.3]),grid on,hold on
```

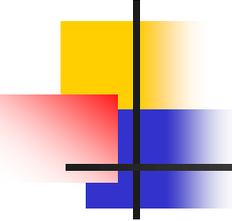
# 连续时间周期信号的

# 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





# matlab的一些信号产生函数

---

- tripuls
- rectpuls
- gauuspuls
- sawtooth
- pulstran
- chirp
- diric