Chapter 6: Complex Data, Character Data, and Additional Plot Types

Yunong Zhang (张雨浓)

Email: zhynong@mail.sysu.edu.cn

Complex Data

- Commonly used in electrical engineering and mechanical systems
- $c=a+bi$

a: *real part*

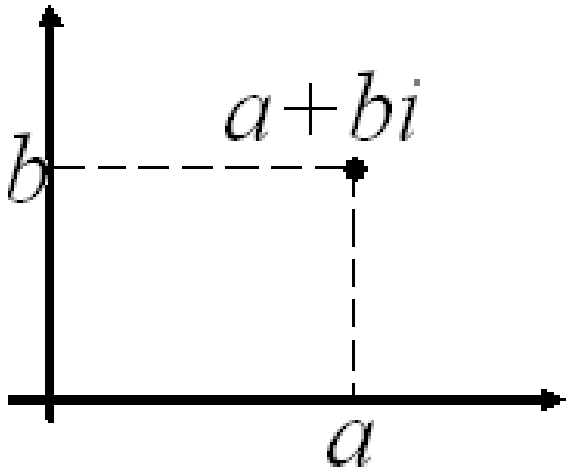
b: *imaginary part*

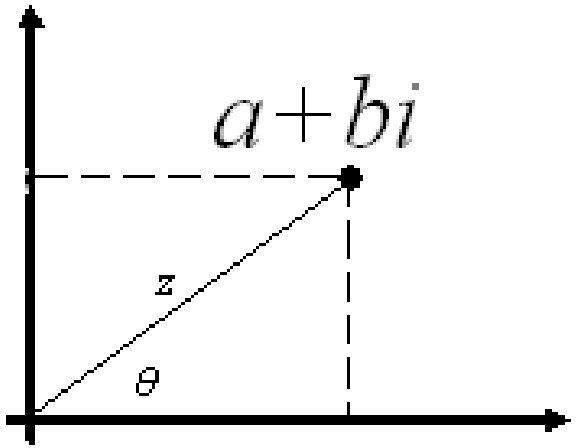
$$a = z\cos\theta$$

$$b = z\sin\theta$$

$$z = \sqrt{a^2 + b^2}$$

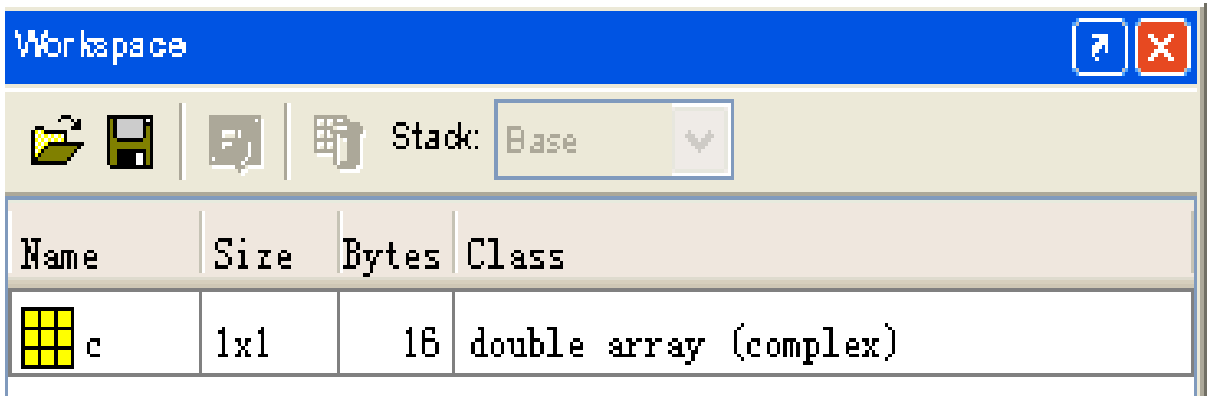
$$\theta = \tan^{-1}(b/a)$$





Complex Data (Cont.)

- c=4+3i, or
- c=4+3*i



>> help i

I Imaginary unit.

As the basic imaginary unit SQRT(-1), i and j are used to enter complex numbers. For example, the expressions 3+2i, 3+2*i, 3+2j, 3+2*j and 3+2*sqrt(-1) all have the same value.

Since both i and j are functions, they can be overridden and used as a variable. This permits you to use i or j as an index in FOR loops, etc.

See also J.

override重写、覆盖

Using complex numbers with relational operators

- c1=a1+b1*i; c2=a2+b2*i
- ==

  c1=c2 if and only if a1=a2 and b1=b2
- ~=

  c1~=c2 if a1~=a2 or b1~=b2
- >, <, >=, <=

  Only the real parts of the numbers are compared

Complex Functions

- **conj(c):** conj(c)=a-b*i
- **real(c):** Returns the real part of c
- **imag(c):** Returns the imaginary part of c
- **isreal(c):** Returns true (1) if no element of c has an imaginary component
- **abs(c):** Returns the magnitude of c
- **angle(c):** Returns the angle of c

Complex Functions

>> help exp

EXP Exponential.

EXP(X) is the exponential of the elements of X, e to the X.

For complex Z=X+i*Y, EXP(Z)=EXP(X)*(COS(Y)+i*SIN(Y)).

See also LOG, LOG10, EXPM, EXPINT.

Overloaded methods: help sym/exp.m

>> c=3+4i ➔ c = 3.0000 + 4.0000i

>> exp(c) ➔ ans = -13.1288 -15.2008i

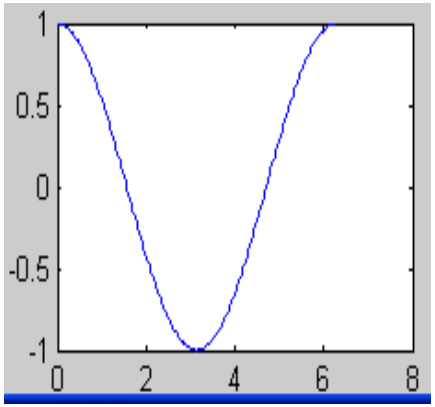
>> exp(2i) ➔ ans = -0.4161 + 0.9093i

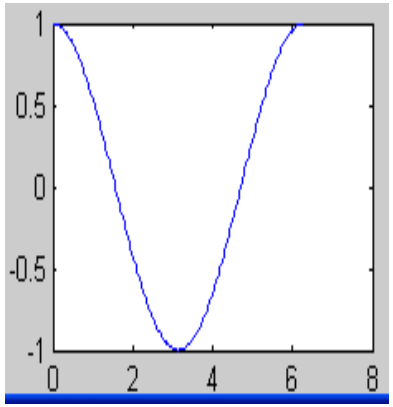
>> exp(-2i) ➔ ans = -0.4161 - 0.9093i

Plotting Complex Data

```
t: [0, 2*pi];
y(t)=cos(t);

t=0:pi/100:2*pi;
y=cos(t);
plot(t,y);
```

```
t: [0, 2*pi];
y(t)=cos(t)+ i*sin(t);

t=0:pi/100:2*pi;
y=cos(t)+i*sin(t);
plot(t,y)?
```

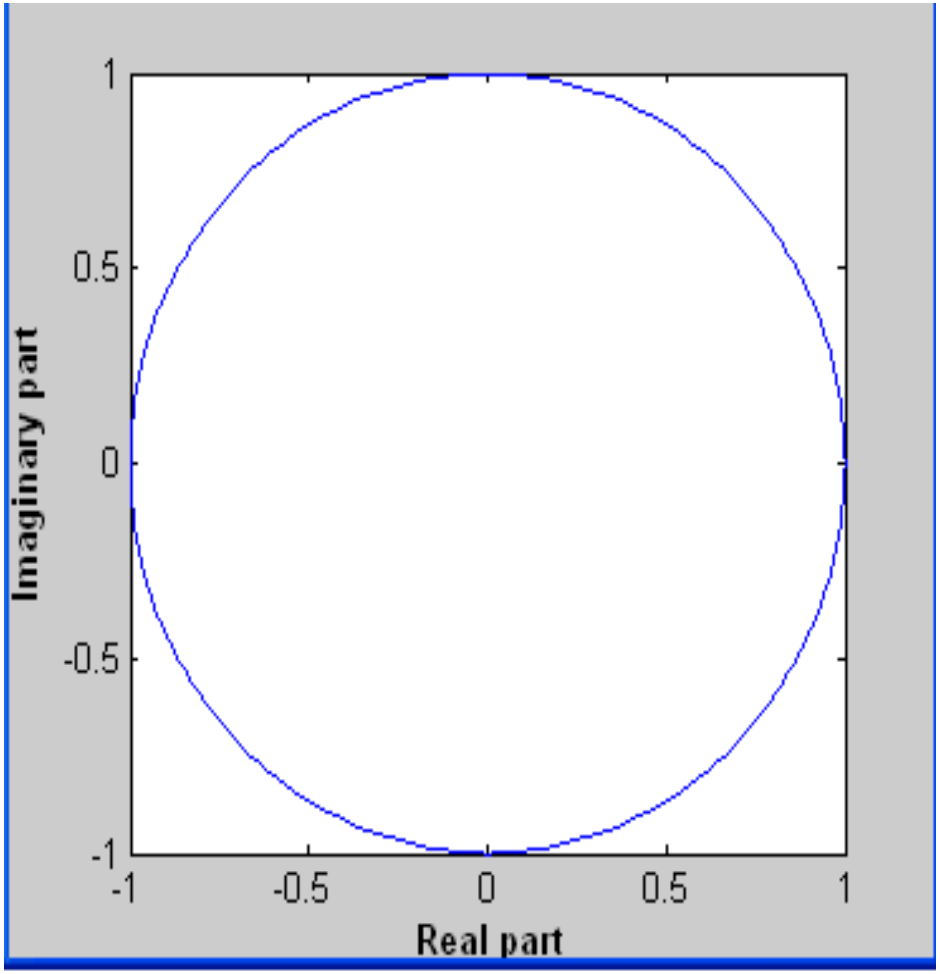




plot command:
Only the **real part** is plotted;
the imaginary part is ignored

Plotting Complex Data (Cont.)

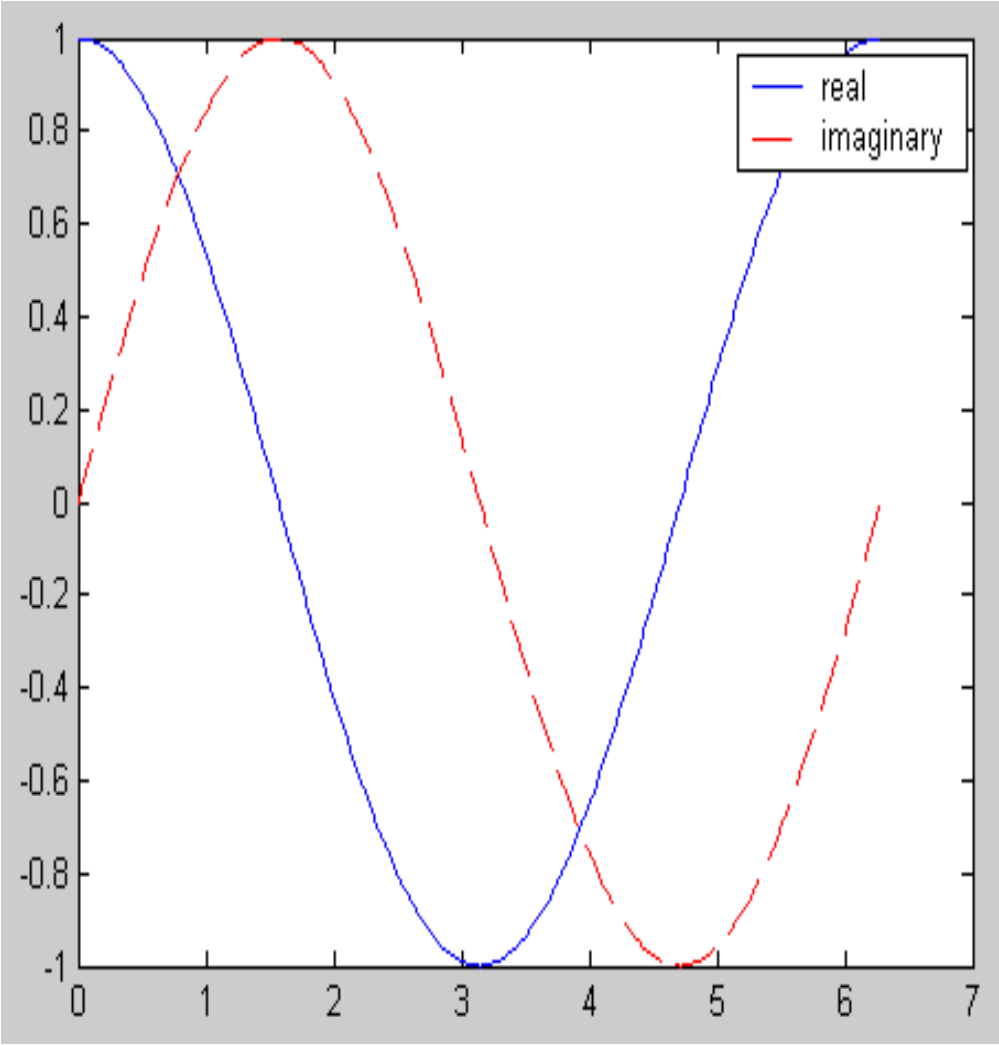
```
t: [0, 2*pi];
y(t)=cos(t)+ i*sin(t);

t=0:pi/100:2*pi;
y=cos(t)+i*sin(t);
plot(y);
xlabel('\bf Real part');
ylabel('\bf Imaginary part');
```



Plotting Complex Data (Cont.)

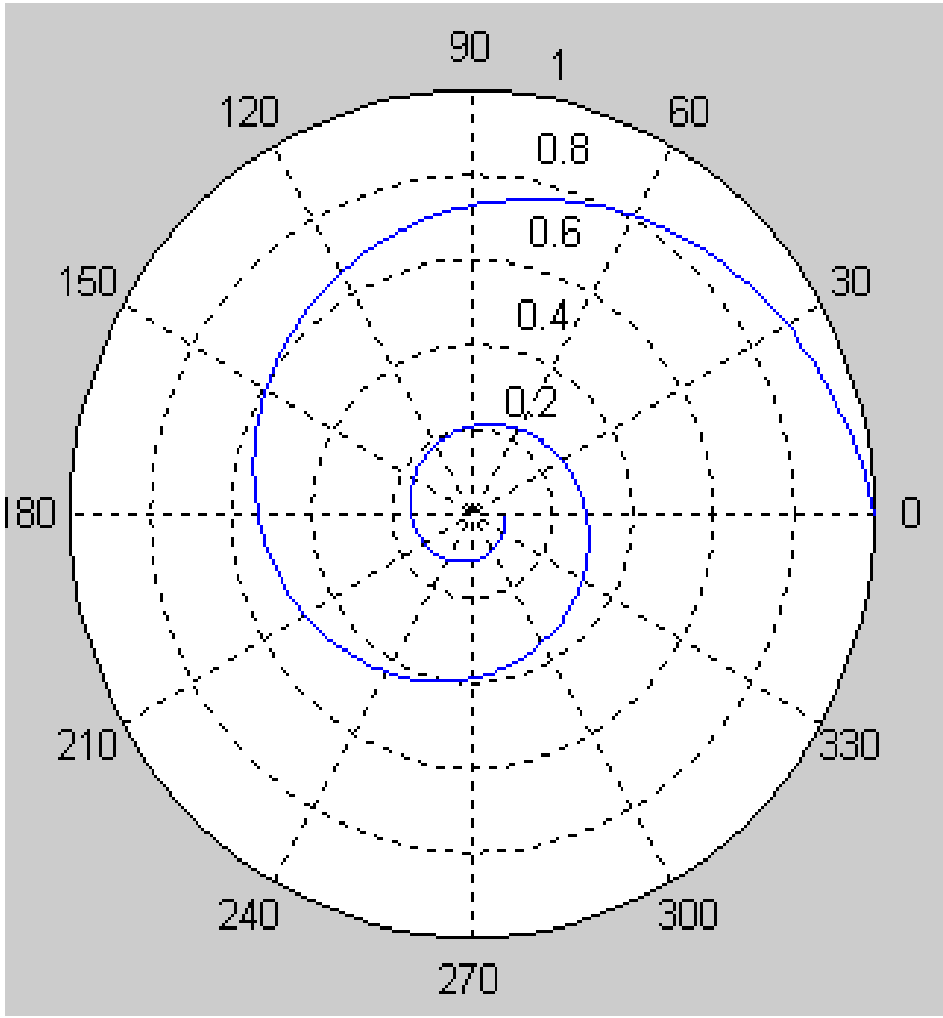
t: [0, 2*pi];
y(t)=cos(t)+ **i*sin(t);**

```
t=0:pi/100:2*pi;
y=cos(t)+i*sin(t);
plot(t,real(y),'b-');
hold on;
plot(t,imag(y),'r--');
legend('real','imaginary');
```



Plotting Complex Data (Cont.)

```
t=0:pi/100:4*pi;
y=exp(-0.2*t).*(cos(t)+i*sin(t));
polar(angle(y),abs(y));
```



String Functions

- str=‘Matlab programming’;
- str1=['Matlab programming', 'Stephen'];
- book=['Matlab programming’;
  'Stephen’];
  Illegal! Same length is required!
- book=char('Matlab programming’, 'Stephen’);

String Functions (Cont.)

```
>> str1=['Matlab programming', 'Stephen']

str1 =

Matlab programmingStephen

>> book=['Matlab programming'; 'Stephen']
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.

>> book=char('Matlab programming', 'Stephen')

book =

Matlab programming
Stephen
```

Workspace

Name	Size	Bytes	Class
book	2x18	72	char array
str1	1x25	50	char array

>> help char

CHAR Create character array (string).

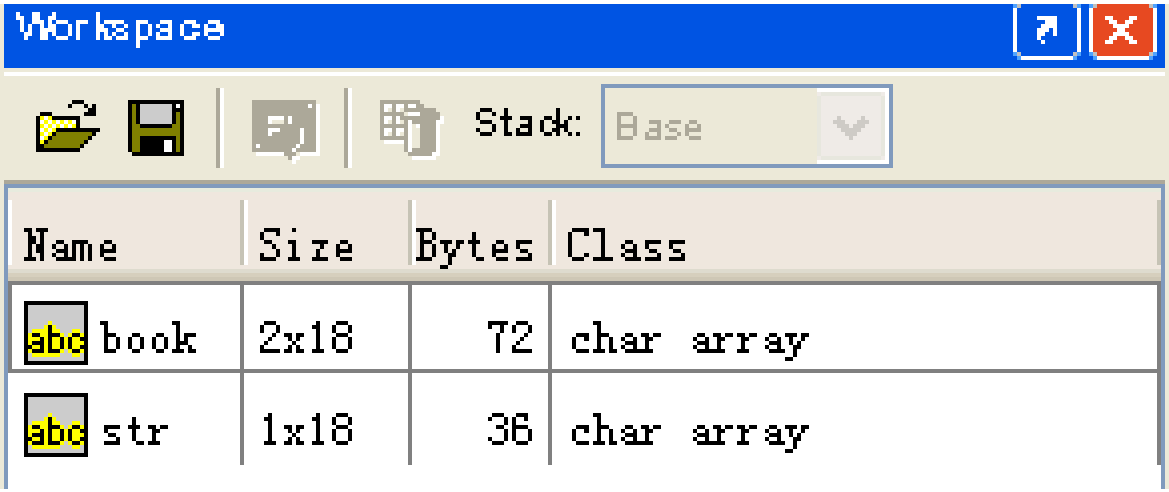
S = CHAR(X) converts the array X that contains positive integers representing character codes into a MATLAB character array (the first 127 codes are ASCII). The actual characters displayed depends on the character set encoding for a given font. The result for any elements of X outside the range from 0 to 65535 is not defined (and may vary from platform to platform). Use DOUBLE to convert a character array into its numeric codes.

S = CHAR(C), when C is a cell array of strings, places each element of C into the rows of the character array S. Use CELLSTR to convert back.

S = CHAR(T1,T2,T3,..) forms the character array S containing the text strings T1,T2,T3,... as rows. Automatically pads each string with blanks in order to form a valid matrix. Each text parameter, Ti, can itself be a character array. This allows the creation of arbitrarily large character arrays. Empty strings are significant.

String Functions (Cont.)

- Each character is stored in two bytes of memory



String Conversion Functions

- Convert data from the character type to the double type using the double function
- str='Matlab programming';
- x=double(str)

```
x =

    77    97   116   108    97    98    32   112   114
   111   103   114    97   109   109   105   110
   103
```

String Conversion Functions (Cont.)

- Convert data from the double type to the character type using the char function
- z=char(x)

z =

Matlab programming

Concatenating Strings

- strcat concatenates two or more strings horizontally, ignoring any trailing spaces
- strcat(‘string 1 ’, ‘string 2’);

- new_str=strcat('Matlab ','Programming')

new_str =

MatlabProgramming

Concatenating Strings (Cont.)

>> help strcat

STRCAT Concatenate strings.

T = STRCAT(S1,S2,S3,...) horizontally concatenates the corresponding rows of the character arrays S1, S2, S3 etc. All input arrays must have the same number of rows (or any can be a single string). When the inputs are all character arrays, the output is also a character array.

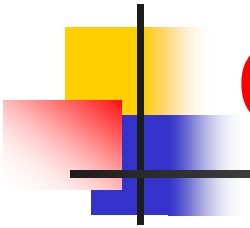
When any of the inputs is a cell array of strings, STRCAT returns a cell array of strings formed by concatenating the corresponding elements of S1, S2, etc. The inputs must all have the same size (or any can be a scalar). Any of the inputs can also be character arrays.

Trailing spaces in character array inputs are ignored and do not appear in the output. This is not true for inputs that are cell arrays of strings. Use the concatenation syntax [S1 S2 S3 ...] to preserve trailing spaces.

Concatenating Strings (Cont.)

- strvcat concatenates two or more strings vertically

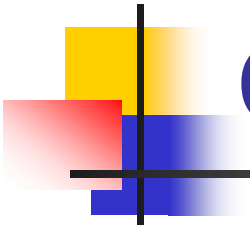
```
>>new_str=strvcat('Matlab','Programming')
new_str =

Matlab
Programming
```



Comparing Strings

- **strcmp**: Determines if two strings are identical
- **strcmpi**: Determines if two strings are identical ignoring case
- **strncmp**: Determines if the first n characters of two strings are identical
- **strncmpi**: Determines if the first n characters of two strings are identical ignoring case

Recall the JAVA-learning period…



Comparing Strings (Cont.)

str1=‘hello’

str2=‘Hello’

str3=‘help’

result=strcmp(str1,str2); result=0

result=strcmpi(str1,str2); result=1

result=strcmpi(str1,str3,2) result=1

Comparing Strings (Cont.)

str1='hello'

str2='Hello'

str3='help'

- == operator

```
>>result=(str1==str2)
result=
     0 1 1 1 1
```

Searching/Replacing Characters within a String

- Function findstr returns the starting position of all occurrences of the shorter of two strings within a longer string

- Function strrep finds all occurrences of one string within another one, and replaces them by a third string

请您思考、规划和做出函数/程序findvec，其returns the starting position of all occurrences of the shorter of two vectors（如[1,3,5]）within a longer vector（如[0,9,8,1,3,5,2,4,6,1,3,5,2,4,7,1,3,3,5,5,6,7,1,3,5]）.

Searching/Replacing Characters within a String (Cont.)

- Function strrep finds all occurrences of one string within another one, and replaces them by a third string
- strrep(str,searched_str,replaced_str)

- book='Matlab programming';
  result=strrep(book,'ab','cd')
  'Matlcd programming'

Uppercase and Lowercase Conversion

- upper(), lower()
- book='Matlab programming‘

```
>> result=upper(book)

result =

MATLAB PROGRAMMING
```

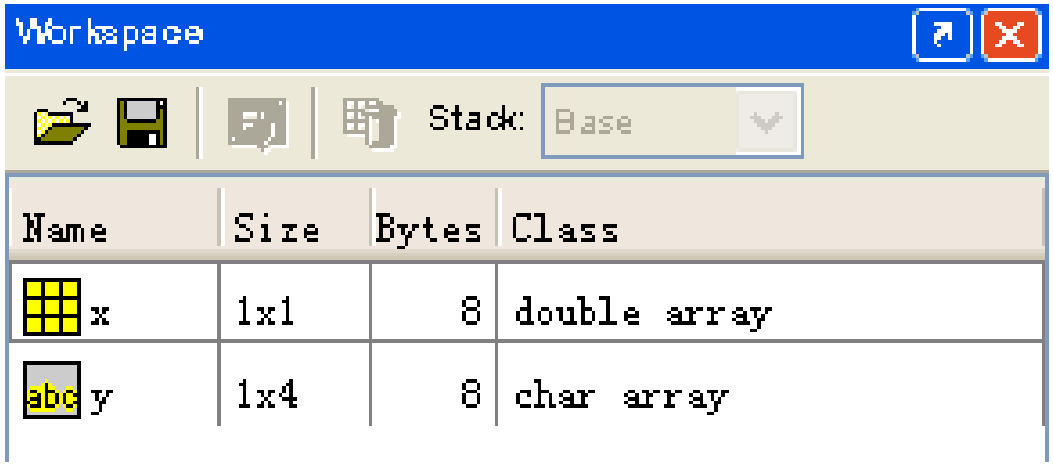
Uppercase and Lowercase Conversion (Cont.)

```
>> result=lower(book)

result =

matlab programming
```

Numeric-to-String Conversions

- int2str() converts an integer to a string array

```
x=5618;
>> y=int2str(x)
y =

5618
```



眼见相同，其实不同（类型不同）

Numeric-to-String Conversions (Cont.)

- T=num2str (X) converts the matrix X into a string representation T with about 4 digits and an exponent if required.

```
>> p=num2str(pi)

p =

3.1416
```

Numeric-to-String Conversions (Cont.)

- T =num2str(X,N) converts the matrix X into a string representation with a maximum N digits of precision.

```
>> p=num2str(pi,10)
p =

3.141592654
```

不会就自己查，还不会就同学讨论，再不会就问老师...

>> help num2str

NUM2STR Convert number to string.

T = NUM2STR(X) converts the matrix X into a string representation T with about 4 digits and an exponent if required. This is useful for labeling plots with the TITLE, XLABEL, YLABEL, and TEXT commands.

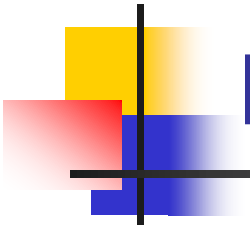
T = NUM2STR(X,N) converts the matrix X into a string representation with a maximum N digits of precision. The default number of digits is based on the magnitude of the elements of X.

T = NUM2STR(X,FORMAT) uses the format string FORMAT (see SPRINTF for details).

Example: num2str(randn(2,2),3) produces the string matrix

'-0.433 0.125'

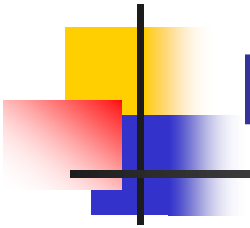
' -1.67 0.288'



Numeric-to-String Conversions (Cont.)

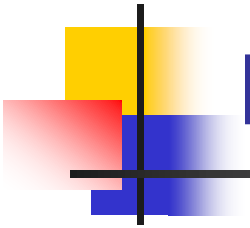
? cf. decimal point, decimal digits

- dec2bin: Convert decimal integer to binary string
- bin2dec: Convert binary string to decimal integer
- dec2hex: Convert decimal integer to hexadecimal string
- hex2dec: Convert hexadecimal string to decimal integer



Numeric-to-String Conversions (Cont.)

```
>> x=5618;
>> result=dec2bin(x)

result =

1010111110010
```



Numeric-to-String Conversions (Cont.)

```
>> result=dec2hex(x)

result =

15F2
```

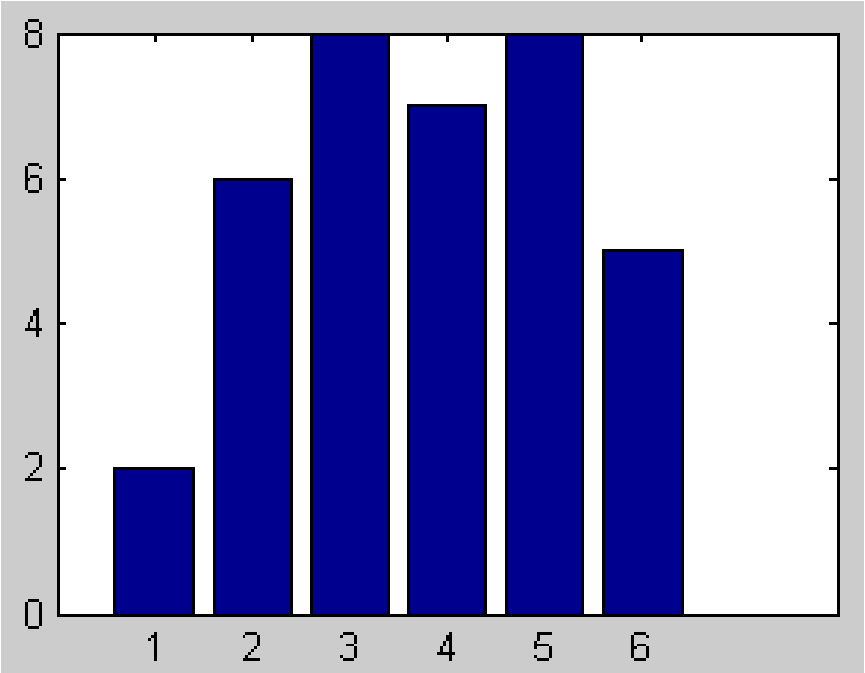
My simple testing

```
>> dec2bin(0.88)              ans = 0
>> dec2bin(-1)                ans = /
>> dec2bin(-10)               ans = 0//0
>> dec2bin(-100)              ans = 00///00
>> dec2bin(100)               ans = 1100100
>> dec2hex(1.011)
??? Error using ==> reshape
To RESHAPE the number of elements must not change.
Error in ==> C:\MATLAB6p1\toolbox\matlab\strfun\...
On line 41  ==> h = reshape(h,n,length(d))';
>> dec2hex(0.011)
??? Error using ==> reshape
```

Additional two-dimensional plots

- bar(x,y): draws y as vertical bars at the positions of x. The vector x must be monotonically increasing or decreasing.

```
>> x=[1 2 3 4 5 6];
>> y=[2 6 8 7 8 5];
>> bar(x,y);
```



Additional Two-dimensional plots (Cont.)

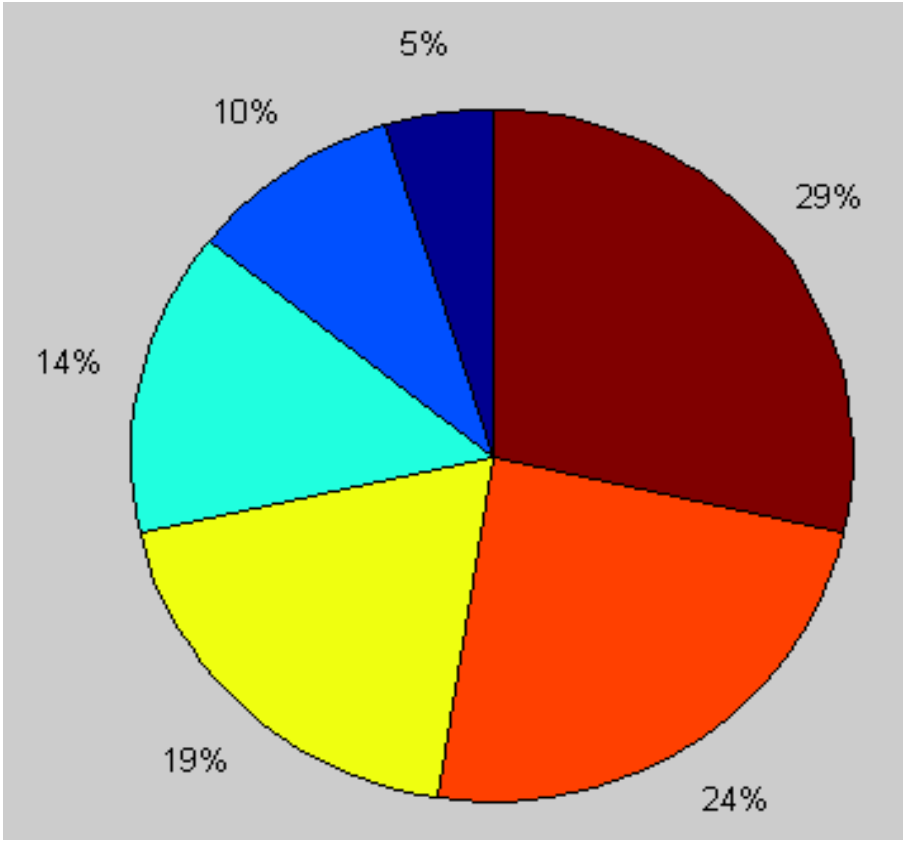
- bar(x,y,width) specifies the width of the bars. Values of width > 1, produce overlapped bars. The default value is width=0.8.

```
>> x=[1 2 1 4 7 5];
>> y=[2 6 8 7 8 5];
>> bar(x,y)
??? Error using ==> set
Values must be monotonically increasing.
```

Additional Two-dimensional plots (Cont.)

- pie(x): draws a pie plot of the data in the vector x. The values in x are normalized via x/sum(x) to determine the area of each slice of pie.

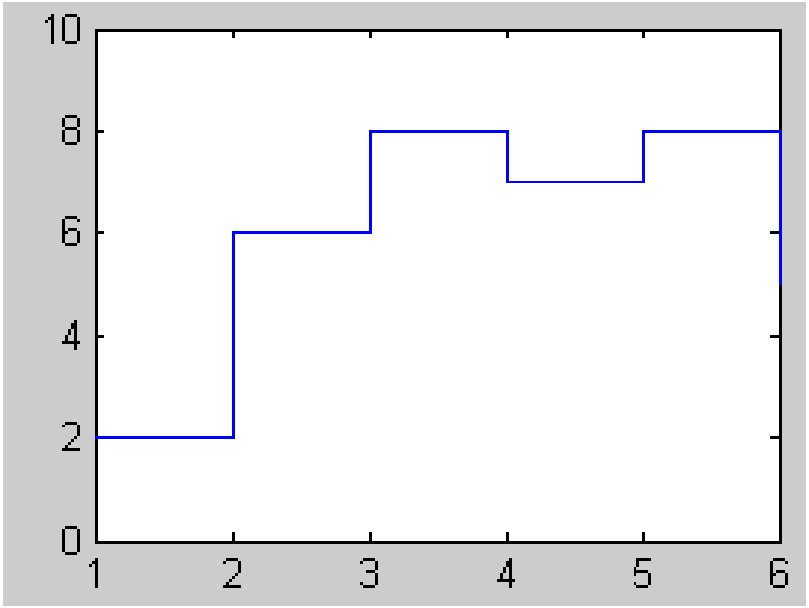
```
>> x=[1 2 3 4 5 6];
>> pie(x);
```



Additional Two-dimensional plots (Cont.)

- stairs(x,y): draws a stairstep graph of the elements in vector y at the locations specified in x. The x-values must be in ascending order and evenly spaced

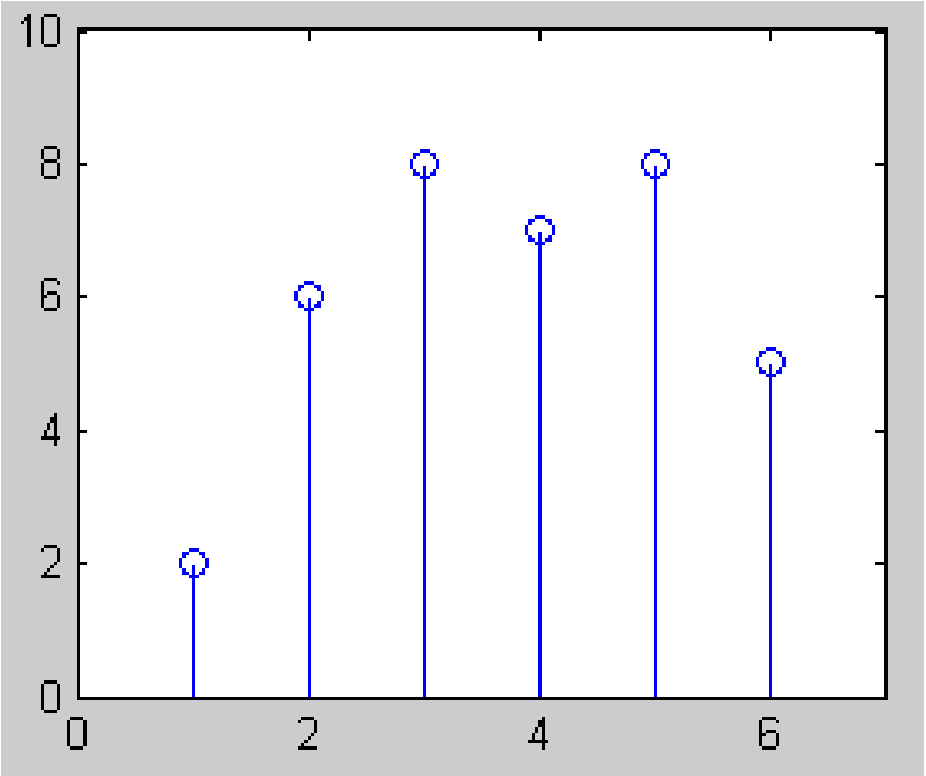
```
>> x=[1 2 3 4 5 6];
>>y=[2 6 8 7 8 5];
>>stairs(x,y);
```



Additional Two-dimensional plots (Cont.)

- stem(x,y): plots the data sequence y as stems from the x axis terminated with circles for the data value

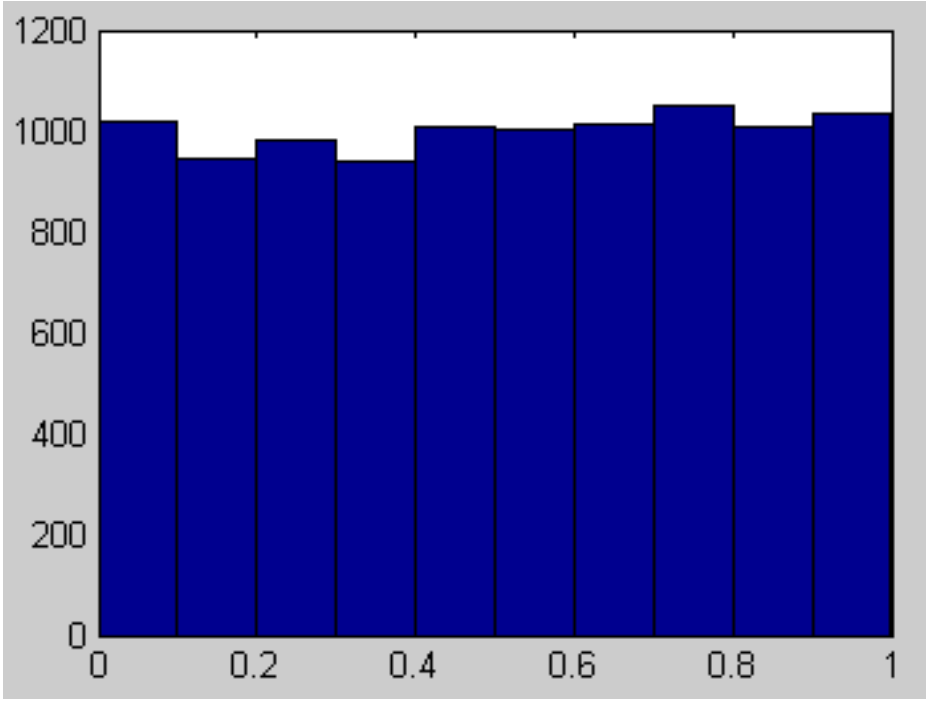
```
>> x=[1 2 3 4 5 6];
>>y=[2 6 8 7 8 5];
>>stem(x,y);
```



Additional Two-dimensional plots (Cont.)

- hist(y): produces a histogram bar plot of the results with 10 equally spaced bins

```
>> y=rand(10000,1);
>> hist(y);
```



- hist(y,nbins)

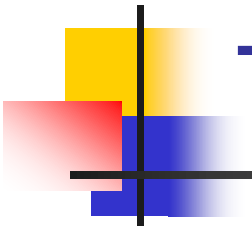
dust bin 垃圾箱

Three-dimensional Plots

- z=f(x,y);
- plot3(x,y,z);

$$z(x,y) = e^{-0.5\left[x^2+0.5(x-y)^2\right]}$$

$$x \in [-4,4], y \in [-4,4]$$

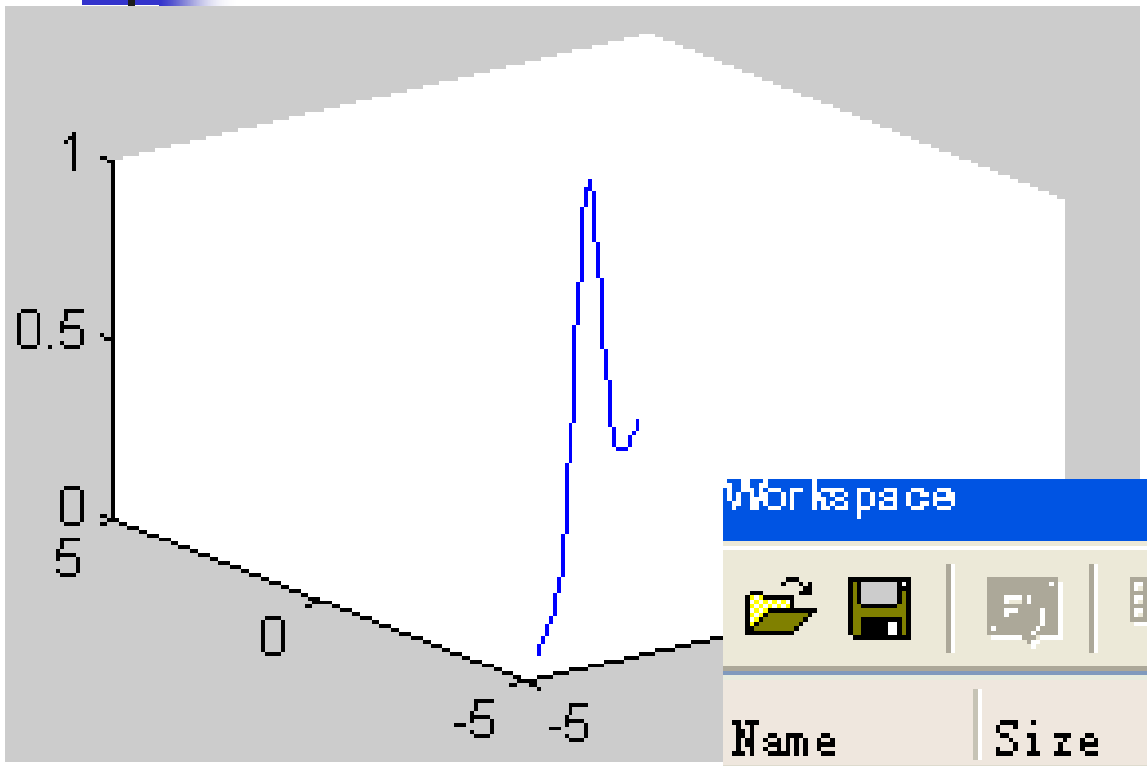


Three-dimensional Plots (Cont.)

p280.m

```
x=-4:0.2:4;
y=-4:0.2:4;
z=exp(-0.5*(x.*x+0.5*(x-y).^2));
plot3(x,y,z);
```

Three-dimensional Plots (Cont.)




```
[x1,x2,...,x41];
[y1,y2,...,y41];
[z1,z2,...,z41];
```

Workspace

Stack: Base

Name	Size	Bytes	Class
x	1x41	328	double array
y	1x41	328	double array
z	1x41	328	double array

Three-dimensional Plots (Cont.)

遍历思想

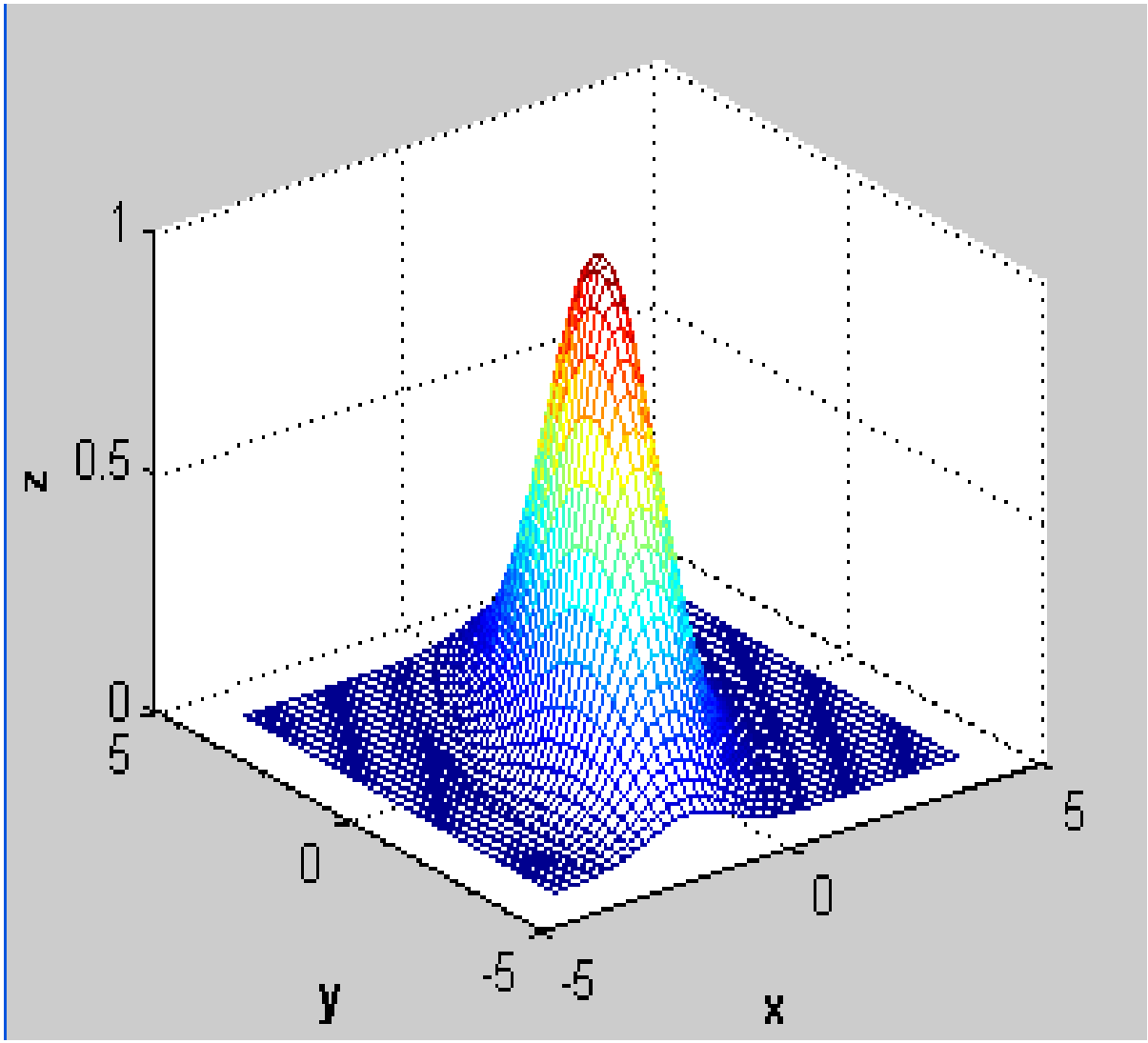
```
y=y1
x=[x1,x2,...,xn];
z=[z11,z12,...,z1n];

y=y2
x=[x1,x2,...,xn];
z=[z21,z22,...,z2n];
...
y=ym
x=[x1,x2,...,xn];
z=[zm1,zm2,...,zmn];
```

```
X=[x1,x2,...,xn
    x1,x2,...,xn
    ...
    x1,x2,...,xn];

Y=[y1,y1,...,y1
    y2,y2,...,y2
     ...
   ym,ym,...,ym];
Z=[z11,z12,...,z1n
    z21,z22,...,z2n
    ...
    zm1,zm2,...,zmn];
```

有点“绕脑”

Three-dimensional Plots (Cont.)

mesh(X,Y,Z);
surf(X,Y,Z);
contour(X,Y,Z);



Mesh 网状

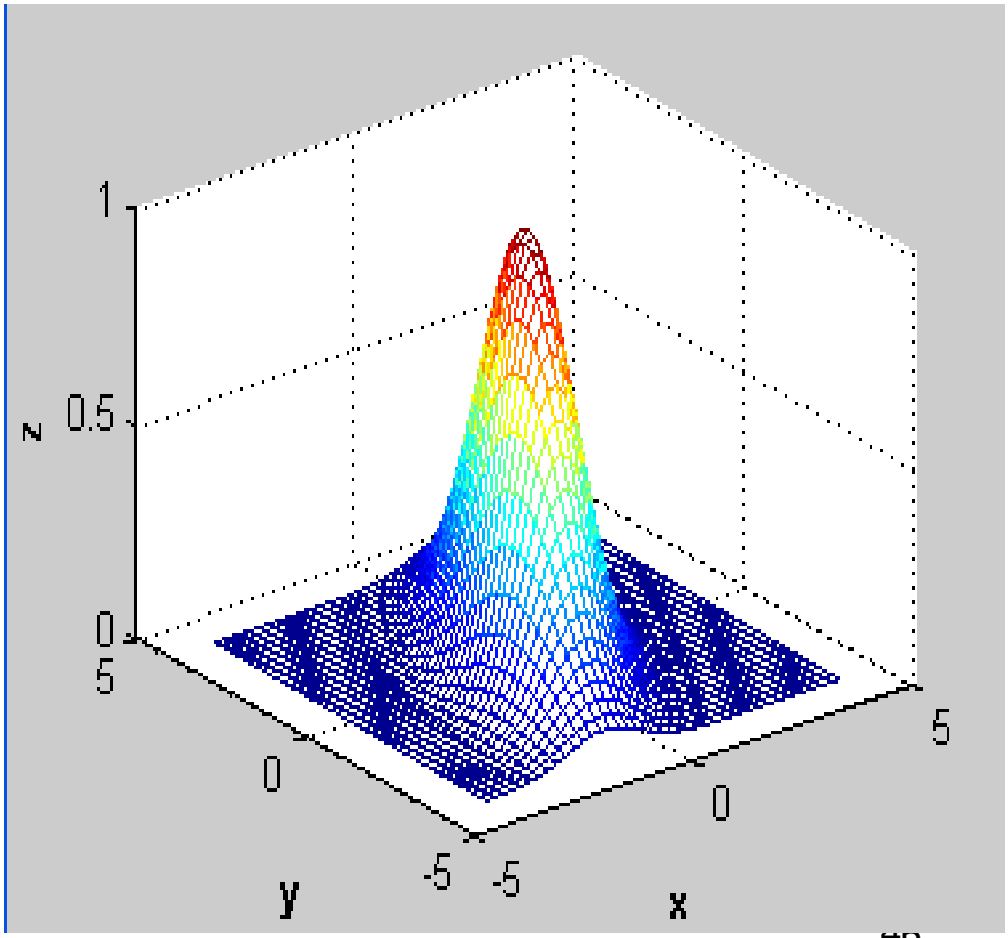
Surface 表面

Contour 等高线

Three-dimensional Plots (Cont.)

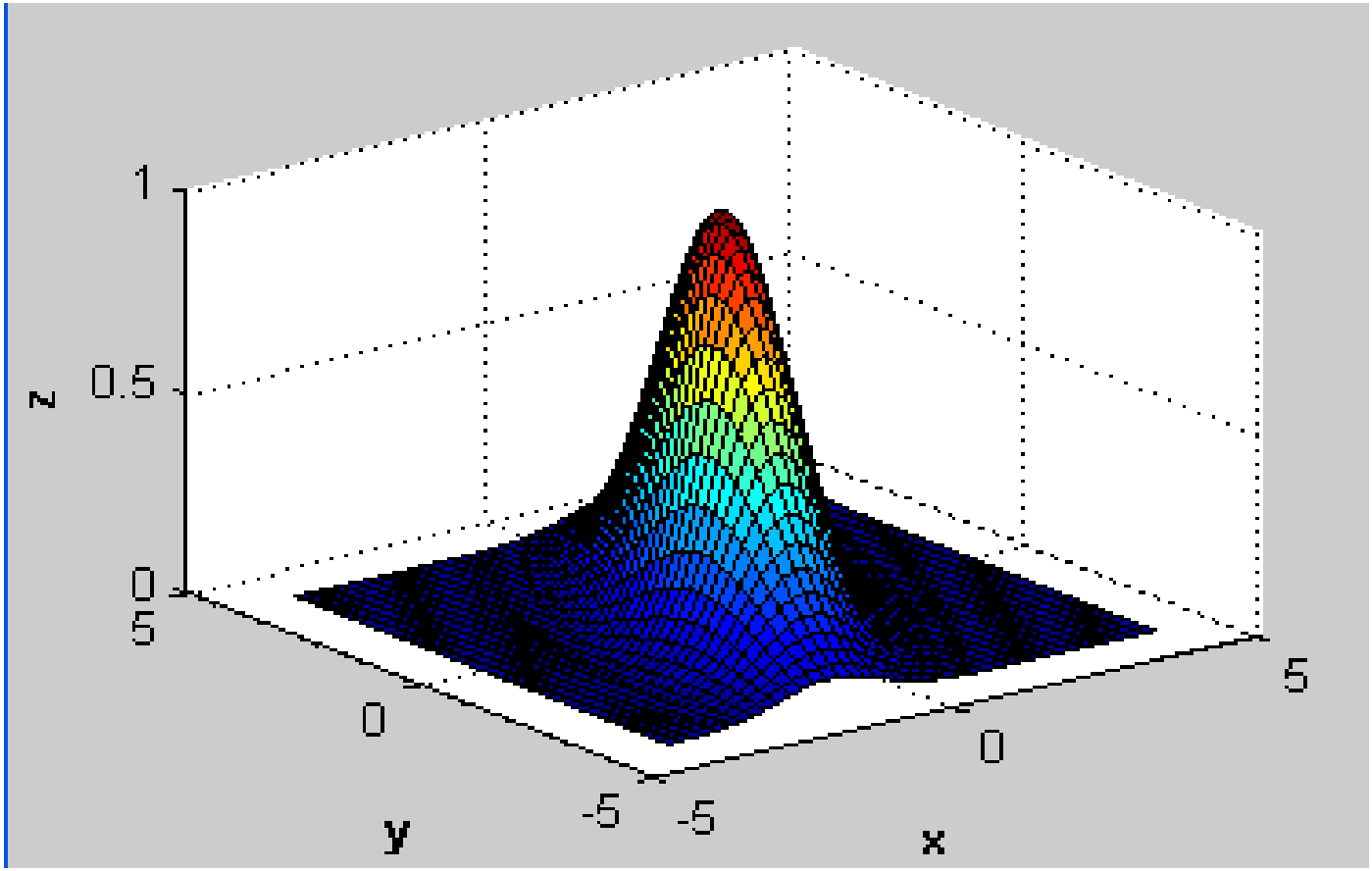
```
[x y]=meshgrid(-4:0.2:4,-4:0.2:4);
z=exp(-0.5*(x.*x+0.5*(x-y).^2));
mesh(x,y,z);
xlabel('\bfx');
ylabel('\bfy');
zlabel('\bfz');
```

```
>> [x y]=meshgrid(1:3,4:6);
>> x
 x =

     1     2     3
     1     2     3
     1     2     3

>> y
 y =

     4     4     4
     5     5     5
     6     6     6
```



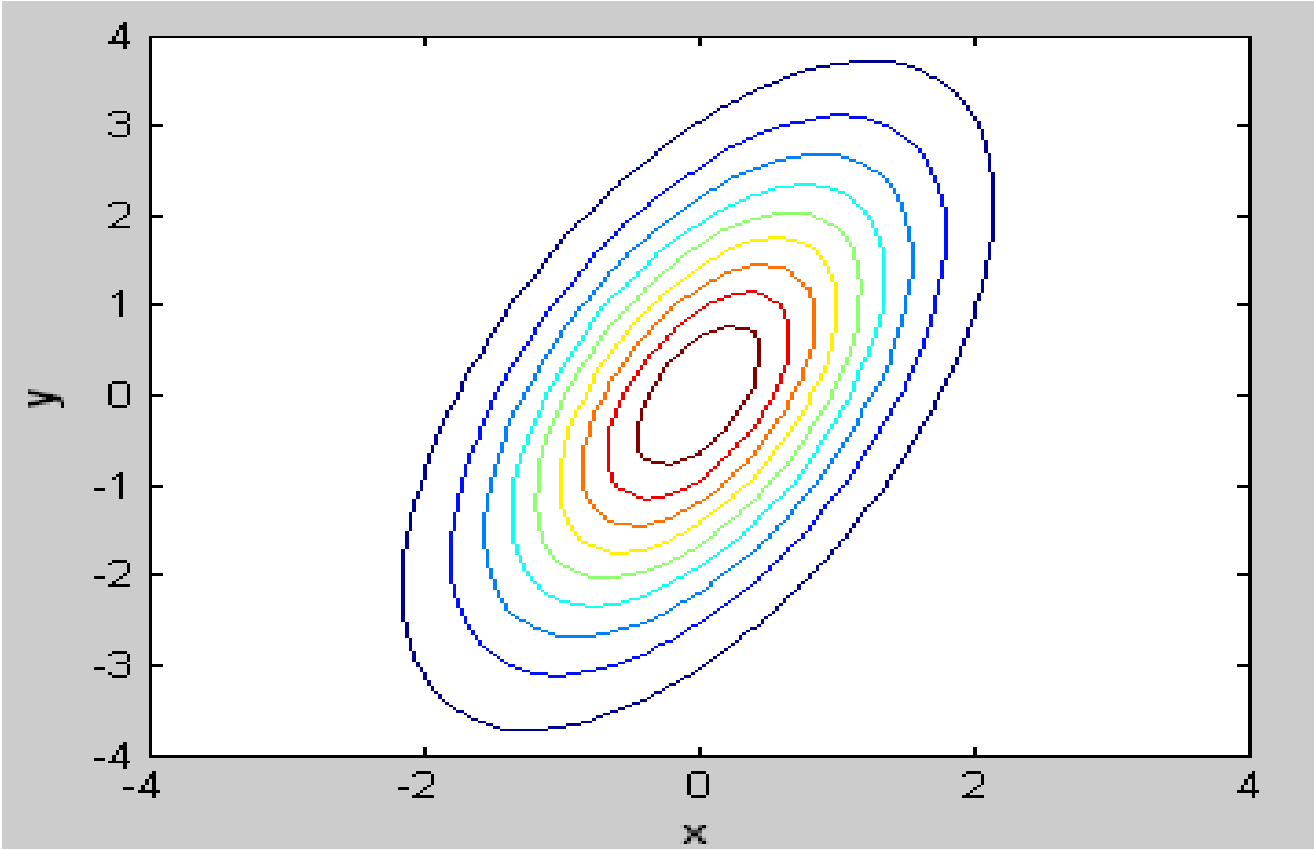
Three-dimensional Plots (Cont.)

```
[x y]=meshgrid(-4:0.2:4,-4:0.2:4);
z=exp(-0.5*(x.*x+0.5*(x-y).^2));
surf(x,y,z);
xlabel('\bfx');
ylabel('\bfy');
zlabel('\bfz');
```



Three-dimensional Plots (Cont.)

```
[x y]=meshgrid(-4:0.2:4,-4:0.2:4);
z=exp(-0.5*(x.*x+0.5*(x-y).^2));
contour(x,y,z);
xlabel('\bfx');
ylabel('\bfy');
```



Sincere Thanks!

- Using this group of PPTs, please read

- [1] Yunong Zhang, Weimu Ma, Xiao-Dong Li, Hong-Zhou Tan, Ke Chen, MATLAB Simulink modeling and simulation of LVI-based primal-dual neural network for solving linear and quadratic programs, Neurocomputing 72 (2009) 1679-1687

- [2] Yunong Zhang, Chenfu Yi, Weimu Ma, Simulation and verification of Zhang neural network for online time-varying matrix inversion, Simulation Modelling Practice and Theory 17 (2009) 1603-1617