ĐẠI HỌC

BÁCH KHOA

25 YEARS ANNIVERSARY

SOICT

ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

CHAPTER 1

INTRODUCTION TO MATLAB

Vũ Văn Thiệu, Đinh Viết Sang, Nguyễn Khánh Phương

SCIENTIFIC COMPUTING

Content

1. Introduction about MATLAB
2. MATLAB working environment
3. MATLAB programming
4. Advanced matrix computation
5. Advanced graph
6. Polynimial
7. Advanced Data Input/Output

Introduction about MATLAB

- MATLAB (Matrix Laboratory) is a product of MathWorks Inc.
- MATLAB stands for MATrix LABoratory.
- MATLAB integrates powerful calculation methods, displays, and programming languages to provide users with a convenient working environment to solve many scientific computing problems.
- The open architecture of MATLAB allows the use of MATLAB and its components to survey data, study algorithms, and create user-friendly tools.

Ad/Disadvantages of MATLAB

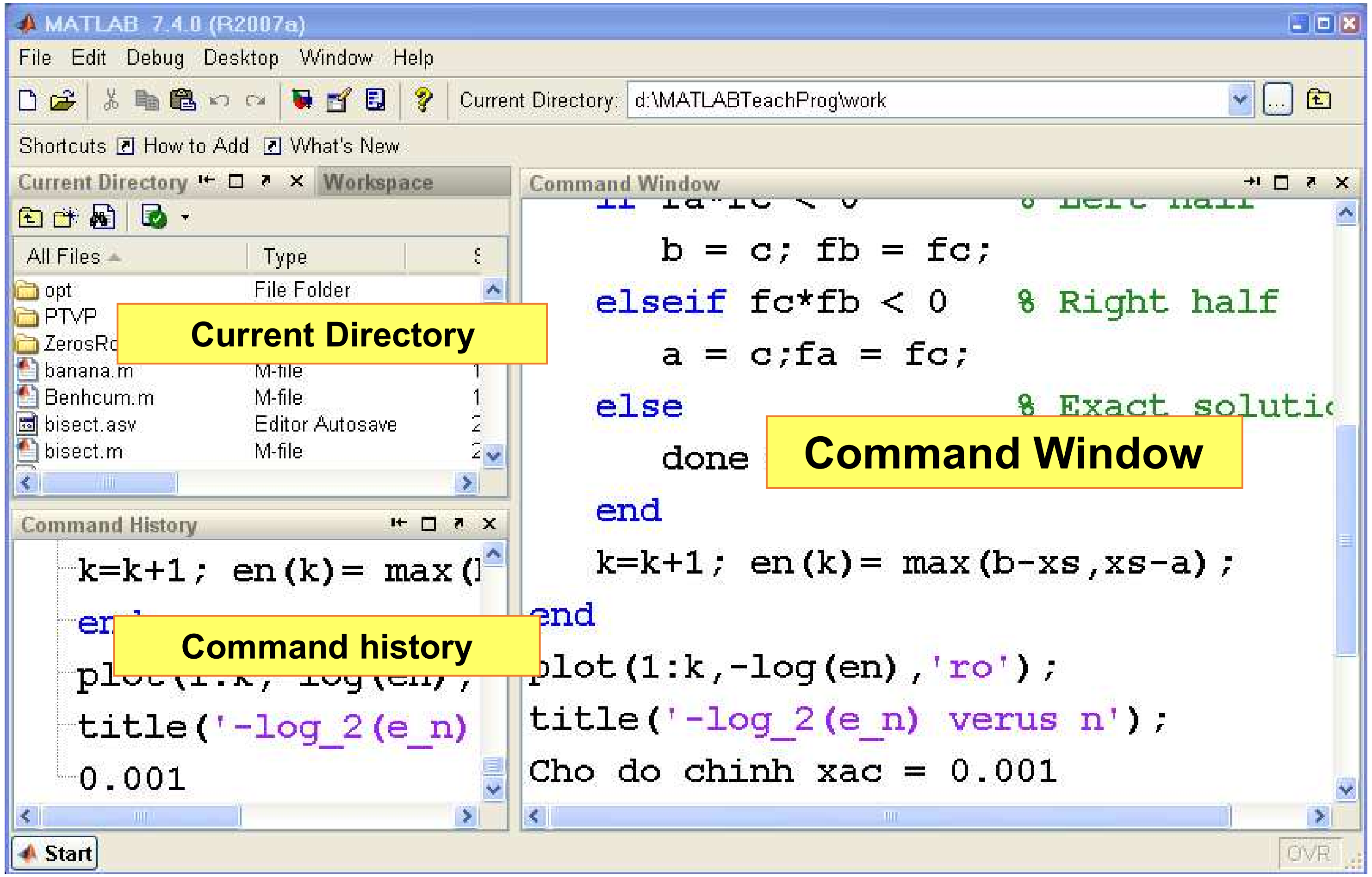
- Advantages
  - Ease of use
  - Powerful built-in routines and toolboxes (LOTS!!!)
  - Good visualization of results
  - Popularity in both academia and industry
- Disadvantages
  - Can be slow (MATLAB is an interpreted language)
  - Must be licensed (it’s not free :)

The use of MATLAB

- MATLAB has also created a lot of utility tools such as:
  - Data mining (Data acquisition)
  - Data analysis and exploration
  - Visualization and image processing
  - Algorithm prototyping and development
  - Modeling and simulation
- MATLAB is a tool used by scientists and engineers to develop software to solve computational problems in science and engineering.
- MATLAB itself also provides tools to solve many problems of science and technology.
- MATLAB is used in many universities to support the teaching of mathematics and scientific computing, especially those related to numerical computation such as applied linear algebra, numerical analysis, scientific computation,

WORKING WITH MATLAB

MATLAB interface

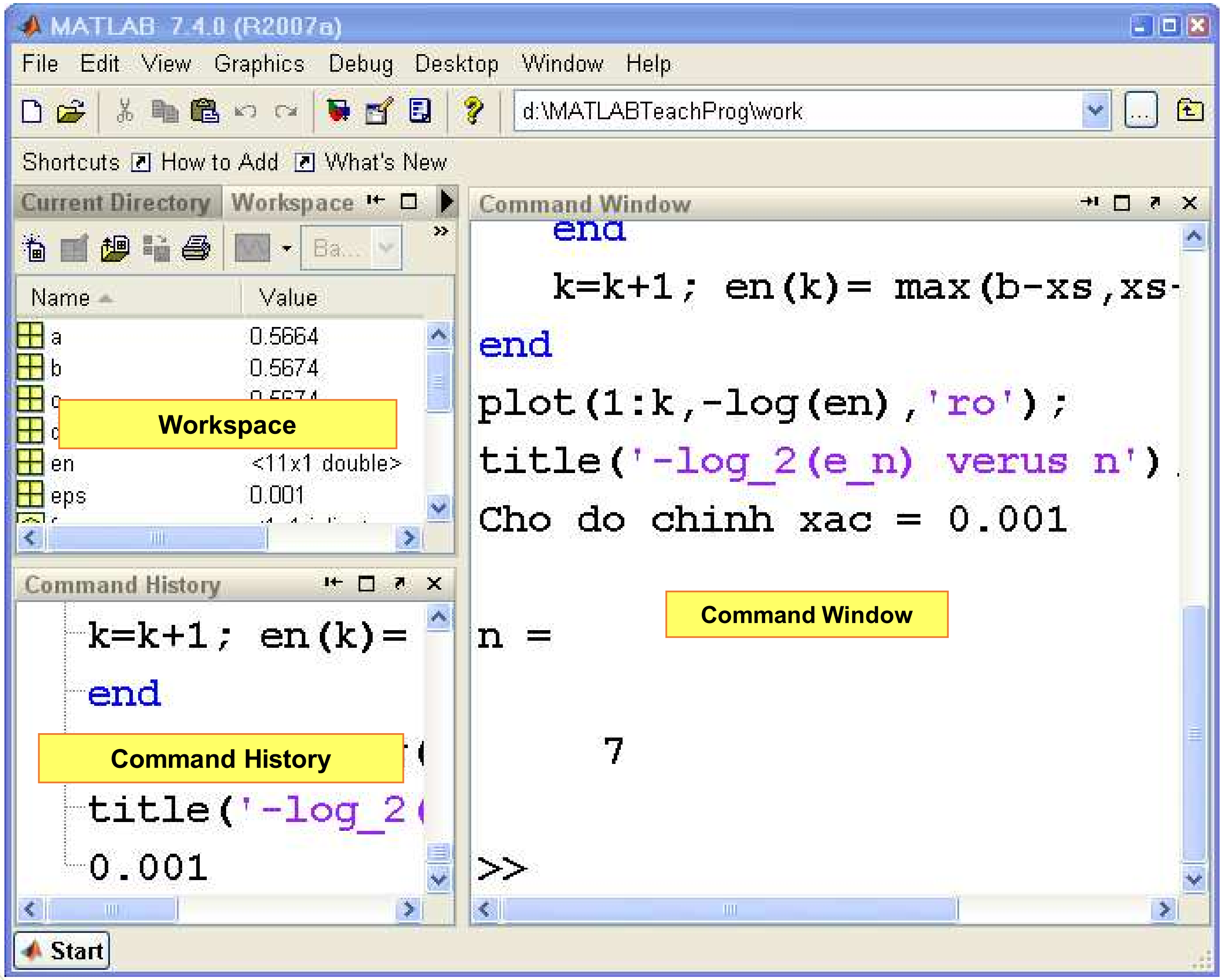


All Files	Type
opt	File Folder
PTVP	
ZerosRc	
banana.m	M-file
Benhcum.m	M-file
bisect.asv	Editor Autosave
bisect.m	M-file

```
    b = c; fb = fc;
  elseif fc*fb < 0    % Right half
    a = c;fa = fc;
  else                % Exact solutio
    done
  end
  k=k+1; en(k)= max(b-xs,xs-a);
end
plot(1:k,-log(en),'ro');
title('-log_2(e_n) verus n');
Cho do chinh xac = 0.001
```

```
k=k+1; en(k)= max(
plot(1:k,-log(en),
title('-log_2(e_n)
0.001
```

MATLAB interface



Name	Value
a	0.5664
b	0.5674
en	<11x1 double>
eps	0.001

```
    end
    k=k+1; en(k)= max(b-xs,xs-
end
plot(1:k,-log(en),'ro');
title('-log_2(e_n) verus n').
Cho do chinh xac = 0.001


n =


     7

>>
```

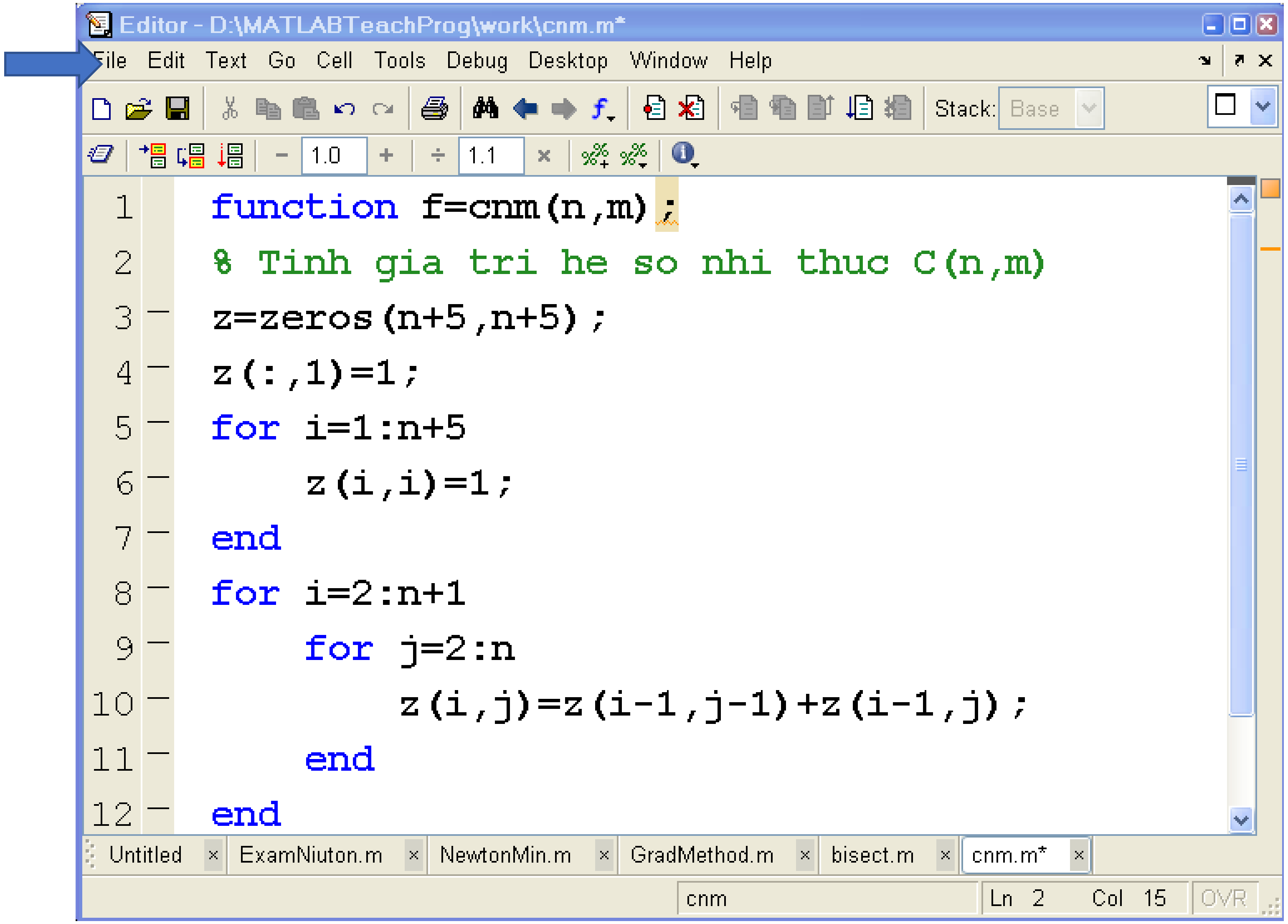
```
k=k+1; en(k)=
end

title('-log_2(
0.001
```

Program on MATLAB

- MATLAB can work as a super-calculater if we just need MATLAB to execute some commands by typing directly on the command window
- The program on MATLAB could be:
  - Scripts: The sequence of MATLAB commands written in a file is entered into the command window and is executed immediately
  - Functions: Program modules that take input and return results (e.g. sin function takes input x and returns sin(x))
- Programs can be created with any text editor (although MATLAB also provides its own program editor)

MATLAB Editor


Editor - D:\MATLABTeachProg\work\cnm.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
Untitled
ExamNiuton.m
NewtonMin.m
GradMethod.m
bisect.m
cnm.m*
cnm
Ln 2 Col 15 OVR

```
function f=cnm(n,m);
% Tinh gia tri he so nhi thuc C(n,m)
z=zeros(n+5,n+5);
z(:,1)=1;
for i=1:n+5
    z(i,i)=1;
end
for i=2:n+1
    for j=2:n
        z(i,j)=z(i-1,j-1)+z(i-1,j);
    end
end
```

Compiling in MATLAB

- MATLAB is an interpreted language.
  - Commands are typed directly in the command window and are executed instantly
  - Variables are allocated memory at the first time they are initialized
  - To re-execute a command, just retype the command
- All variables used in the command window are stored in the Workspace
  - New values can be assigned to variables if necessary
  - You can choose to delete some variables from the Workspace
  - Workspace can be stored in a data file
  - The data file extension is .mat (e.g. mydata.mat)
  - File is a binary file (.mat extension) can be loaded back into the Workspace

Commands, Statements, Variables

- Command
  - `save mydata` (save Workspace to `mydata.mat`)
  - `whos` (list all variables in Workspace)
- Assignment statement
  - A = width * length;
  - B = 267;
  - The assignment statement has only one variable name on the left-hand side of the assignment operator (=).
  - The right-hand side will be calculated based on the current values of the variables and the calculated result will be assigned to the variable on the left-hand side.
  - The value can be numeric or character
  - The type of the variable will be updated every time it is assigned a value
- Variable
  - Distinguish the first 31 characters; Distinguish between upper and lower case

Work in conversation mode

- When using the conversation mode, the user enters the command directly after the MATLAB prompt. When pressing the “Enter” button, the command will be executed.
- Example:

```
>> x = 1;
>> 4*atan(x)
```

- The result will be displayed on the screen as

```
ans = 3.1416
```

- Semicolon ";" at the end of the command is used to prevent MATLAB from displaying the result on the operation box.

Reserved words

- MATLAB uses a series of keywords that, to avoid conflicts, should not be used to name variables…

```
for
end
if
while
function
return
elsif
case
otherwise
```

```
switch
continue
else
try
catch
global
persistent
break
```

Predefined/Built-in variables and constants

Variables and Constants	Description
`ans`	Default variable containing the result
`beep`	Beep voice
`pi`	Constant pi
`eps`	Zero
`inf`	Infinity
`NaN`	Not a number
`i  or  j`	Complex number
`realmin, realmax`	Min and Max real number
`bitmax`	Max integer number
`nargin, nargout`	Number of In/Output arguments of a function

Predefined/Built-in functions

Func	Description
abs (*x*)	Absolute
cos (*x*)	Cos
sin (*x*)	Sin
tan (*x*)	Tan
acos (*x*)	Arcos
asin(x)	Arsin
exp (*x*)	Exponent of e

Predefined/Built-in functions

Func	Description
imag (*x*)	Imaginary part of complex number
log (*x*)	Logarithm (e)
log10 (*x*)	Logarithm (10)
real (*x*)	Real part of complex number
sign (*x*)	Sign of x
sqrt (x)	Squart
Pow(x,y)	x power y

Data type

- Although all numerical calculations in MATLAB are performed with double precision, the format of the output data can be reformatted using MATLAB 's format commands.
- The format is selected using the format command:

FORMAT SHORT	**real number with 4 digits after dot sign**
FORMAT LONG	**real number with 14 digits after dot sign**
FORMAT SHORTE	**floating-point number**
FORMAT LONGE f	**floating-point number**
FORMAT RAT	**fraction number**

Data type

```
>> pi

ans = 3.1416

>> format long, pi

ans = 3.14159265358979

>> format long e, pi

ans = 3.141592653589793e+000
```

```
>> format short e, pi

ans = 3.1416e+000

>> format rat, pi

ans = 355/113
```

Initialize vector and matrix variables

- One of the strengths of MATLAB is that it allows working with matrices and vectors in a very convenient way.
- To use a variable, we need to initialize it. Vector and matrix variables can be initialized in many ways.
- For vectors (or matrices with only one row) we can use the following initialization methods:

```
>> a = [1 2 3 4 5 6 7 8 9 10];
>> b = [1:10];
>> c = [1:0.5:5.5];
>> d = sin(a);
>> e = [5 d 6];
```

Initialize vector and matrix variables

- One of the most common ways to initialize vectors is to use the operator ':'

  first : increment : last

- Command:

  a= first : increment : last

initialize the vector *a* starting at the first element and ending at the last element with a step length of increment. If increment is not specified, its default value is 1

Initialize vector and matrix variables

- **E.g.:**

1) To initialize vector *a* = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

```
>> a=1:10
a =    1    2    3    4    5    6    7    8    9   10
```

2) Step size can be negative

```
>> a=[100:-10:20]
a =  100   90   80   70   60   50   40   30   20
```

3) The values of the parameter can be determined by the mathematical expression

```
>> c=0:pi/6:2*pi
c =  Columns 1 through 6
0    0.5236    1.0472    1.5708    2.0944    2.6180
Columns 7 through 12
3.1416    3.6652    4.1888    4.7124    5.2360    5.7596
Column 13     6.2832
```

Initialize vector and matrix variables

- To initialize matrix

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [x; y; z];
>> C = [A; 10 11 12];
```

- Note that the rows of the matrix are separated by a semi-colon ";", while the elements on a line are separated by spaces (or commas ‘,’).

Array address

- The elements of a general array (either a vector or a matrix) are addressable in many ways.
- The simplest way is to indicate the element by its row and column position in the array.
- Rows and columns are numbered starting from 1.

Array address

- **E.g.:**

```
>> x = [10 5 3 7 -2 8];
>> x(5)
ans =
-2
>> A = [3 4 9; 2 5 1; 7 4 2]
A =
3 4 9
2 5 1
7 4 1
>> A(1,3)
ans =
9
```

Array address

- Take a row

```
>> A(3,:)
```

- Take a column

```
>> A(:,2)
```

- Take a sub-matrix

```
>> Asub = A(i:j,k:l);
```

- Exchange column

```
>> B = A(:,[3 1 2]);
```

Array address

- Merge two matrices by column

```
>> C = [A B];
```

- Merge two matrices by row

```
>> C = [A; B];
```

- Delete a column

```
>> A(:,2) = [];
```

Operations with vectors and matrices

- Length of vector

  length

- Size of matrix

  size

```
>> [rows cols] = size(A);
```

Operations with vectors and matrices

- Addition (and subtraction) of matrices of the same size is performed component-by-component.

```
>> A=[5 -1 2; 3 4 7]; B=[2 2 1; 5 0 3];
>> A+B
 ans = 7 1 3
       8 4 10
```

- Note that the matrices must have the same size:

```
•   >> C=[3 1; 6 4];
    >> A+C
  ??? Error using ==> + Matrix dimensions
   must agree.
```

Operations with vectors and matrices

- Multiplication matrix by a number (dividing by a non-zero number) is also done by components.

```
>> A=[5 -1 2; 3 4 7];
>> 2*A
 ans =
            10 -2  4
             6  8 14
```

- Note that, '*' sign must be used:

```
>> 2A
??? 2 | Missing operator, comma, or
 semi-colon.
```

Operations with vectors and matrices

- Adding vectors and multiplying vectors by a number are done similarly.

```
>> v=[3; 5]; w=[-2; 7];
>> 10*v-5*w
ans =
40
15
```

Operations with vectors and matrices

- Multiplication of two matrices is also possible in MATLAB.
- To multiply two matrices or multiply a matrix by a vector simply use the ‘*’ operator just like the multiplication of scalar quantities.
- MATLAB recognizes the size of the input and does the multiplication.
- What the user cares about is that the sizes of the matrices and vectors must match for the math to work.

Operations with vectors and matrices

- E.g.:

```
>> x = [1 2 3];
>> A = [4 5 6; 5 4 3];
>> b = A*x
??? Error using ==> *
Inner matrix dimensions must agree.
>> y = [1; 2; 3];
>> b = A*y
b =
32
22
```

Operations with vectors and matrices

- Transpose of a vector

```
>> A = [ 4 5; 5 4; 6 3]
>> A'
ans =
    4     5     6
    5     4     3
```

Operations with vectors and matrices

- Matrix multiplication (multiple by elements) of two matrices of the same size A and B is a matrix A.*B with elements being the product of the corresponding elements of A and B.

```
>> A=[ 1 2 3; 4 5 6]; B=[3 2 1;-1 2 2];
>> A.*B
ans = 3   4   3
     -4  10  12
```

Operations with vectors and matrices

- Division and exponentiation by components: A./B and A.^B are defined similarly. Note that calculations with components must be meaningful, otherwise MATLAB will give an error.

```
>> A./B
 ans = 1/3  1   3
        -4 5/2 3
>> A.^B
ans =  1    4   3
      1/4 25 36
```

Operations with vectors and matrices

- Addition of a matrix with a scalar: The value of a scalar is added to each element of the matrix.

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

Vectorization functions

- MATLAB functions are vectorized. That is, if the input is an array, the output is also an array
- Example: Plot the function y=sin(x) on [0, 2*pi]

```
>> x = (0:.1:2*pi);
>> y = sin(x);
>> plot(x,y)
```

- E.g.:

```
x = (-5:.1:5);
>> y = x./(1+x.^2);
>> plot(x,y)
```

Special matrix functions

- zeros(m,n);
- ones(m,n);
- eye(n);
- diag(v);

Strings

- MATLAB allows the use of character string variable
- The assignment command

```
S = 'Any Characters'
```

create character array (string).

- E.g.:

```
>> msg = 'You''re right!'
      msg =
                You're right!
```

Strings

- The statement S = [S1 S2 ...] concatenates the strings S1, S2,... into a new string S.

```
>> name = ['Michel' ' Paul ' ' Heath ']
   name =
          Michel Paul  Heath
>> name(1)
   ans =
          M
>> name(1)+name(9)
   ans =
         174
```

- Note how to access the elements in the string.

Strings

- S = char(X) converts numbers (ASCII code) to the corresponding characters.
- X = double(S) converts characters to numbers (ASCII code)

```
>> char([65 66 67])
    ans =
             ABC
>> X = double('ABC')
   X =
      65     66     67
```

Strings

- To initialize an array variable where each element is a string, use:

```
  >> S = {'Hello' 'Yes' 'No' 'Goodbye'}
     S =
      'Hello'     'Yes'     'No'
'Goodbye'
  >> S(4)
     ans =
         'Goodbye'
```

PROGRAMMING

Statement

- Assignment statement

    variable = expression

    assign value of ***expression*** to ***variable***.

```
>> x=2^10*pi
x =
   3.216990877275948e+003
```

Statement

- Statement in MATLAB can be:

  expression

  in this case the value of the expression will be assigned to the default variable ***ans.***

```
>> 2*pi*exp(-5)
ans =
   0.04233576958521
```

Statement

- Statement in MATLAB can be:

  variable

  - If the **variable** has been assigned a value before, the content of the variable is displayed on the screen
  - Otherwise, there will be a message that the variable has not been defined.

- The user can take advantage of this to check if a variable name has been used.

Statement

- MATLAB command will be executed immediately after pressing Enter key. If the MATLAB statement ends with Enter, by default, MATLAB will output the execution result to stdout (the screen by default).
- To avoid putting the result directly after the statement, you need to end the statement with a ‘;’ then press Enter.

Statement

- When a MATLAB statement is too long, it can be broken into two or more lines using a hyphen ... at the end of each line containing the statement.
- E.g:

```
>> avariablewithlongname = 100 + (32-17.33)*5 ...
                             + 2^3 - log(10)/log(2);
```

Statement

- To delete all variables in MATLAB, we use the command

  >> clear

- To delete specific variables, we use the command

  ***>> clear** var1 var 2 ...*

Relational operations

Relational operations	Meaning
<	Less than
<=	Less than or equal
>	Bigger than
>=	Bigger than or equal
==	Equal
~=	Not Equal

Logical operations

Logical operations	Meaning
&	AND
|	OR
~	NOT
0	FALSE
<>0	TRUE

IF statement

```
if expr1
   statements1
elseif expr2
   statements2
 . . .
else
   statements
end
```

elseif and else are optional

- The group of commands following the first expression with a non-zero (TRUE) value will be executed immediately.
- If there is no expression with a non-zero (TRUE) value, the group of statements after else is executed

IF statement

```
 >> t = rand(1);
>> if t > 0.75
       s = 0;
   elseif t < 0.25
       s = 1;
   else
       s = 1-2*(t-0.25);
   end
>> s
s =
      0
>> t
t =
    0.7622
```

IF statement

- Logical operations:

<, >, <=, >=, ==, ~=

- E.g.

```
>> 5>3
ans =
    1
>> 5<3
ans =
    0
>> 5==3
ans =
    0
```

FOR statement

- The FOR loop iterates the statements in its body for the values of the index variable taken from a given vector.

```
>> for i=[1,2,3,4]
        disp(i^2)
    end
      1
      4
      9
     16
```

FOR statement

- The FOR loop, like the IF statement, must be ended with **end**. The above loop is usually written as follows.

```
>> for i=1:4
      disp(i^2)
   end
    1
    4
    9
   16
```

- Note: the 1:4 initialization is also [1, 2, 3, 4].

FOR statement

- Both scripts

```
n=4; x = []; for i=1:n, x = [x, i^2], end
```

  Or:

```
x=[];
for i= 1:n
    x = [x, i^2];
end
```

  will create vector x = [1, 4, 9, 16].

- Script

```
n=4;x=[]; for i=n:-1:1, x = [x, i^2],end
```

  will create vector x = [16, 9, 4, 1].

WHILE statement

```
while   expr
   statements
end
```

- The statements in the body of the WHILE loop will be repeated as long as the expr expression is true (with a non-zero value).

WHILE statement

```
>> x=1;
>> while 1+x > 1
      x = x/2;
   end
>> x
x =
   1.1102e-16
```

SWITCH statement

- The SWITCH statement allows branching based on the expression value.

```
switch expression
    case value
        statements
    case {value 1, value 2, value 3,...}
        statements
    ...
    otherwise
        statements
end
```

SWITCH statement

```
>> date= ‘Sunday’;
>> switch lower(date)
      case {‘Sunday’,‘saturday’}
             disp(‘the weekend.')
      otherwise
             disp(‘the workday.')
      end

Result: the weekend.
```

BREAK

- The **break** statement is used to interrupt the execution of a **while** or **for** loop
- In a nested loop, **break** only exits the innermost loop
- If **break** is used outside the loop in a script, it will terminate the script execution
- The **break** statement in the IF or SWITCH structure will terminate the execution of that statement

Script

- The script is a sequence of MATLAB commands recorded in a .m file.
- When typing the name of the file (without .m), this sequence of commands will be executed.
- Note: .m file must be located in one of the directories that MATLAB will automatically search for; A list of such directories can be viewed with the **path** command.
- One of the directories that MATLAB always looks at is the current directory. This directory is displayed in the Current Directory window.
- The user can change the current directory using the utilities in this window.
- Question: how to change current directory?

Script

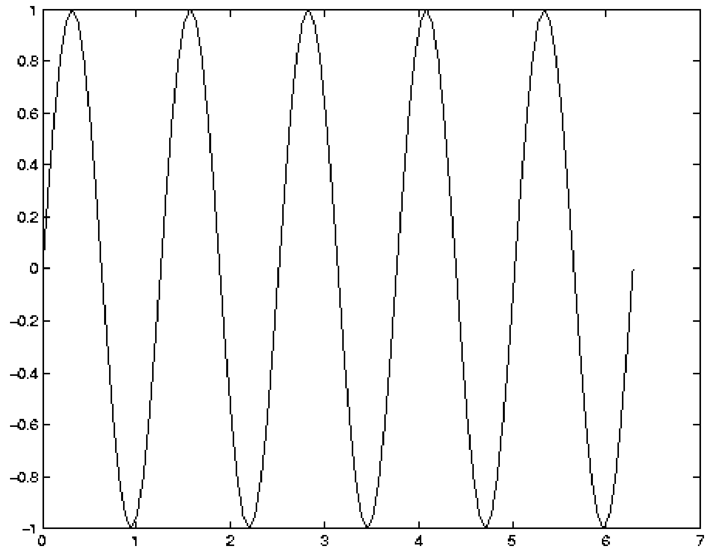
- Script: plotsin.m

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

- Type

```
>> N=100;w=5;
>> plotsin
```

will draw graph:



Function

- Format:

```
function [out1,out2,...] = funcname(inp1, inp2,...)
      statements
end
```

Exercise

- Write a MATLAB function to solve a second order equation:

  $a*x^2 + b*x + c = 0$

ADVANCED MATRIX COMPUTATIONS

List of advanced matrix computations

• eig	: Eigenvalue
• lu	: LU analysis
• chol	: Cholesky analysis
• qr	: QR analysis
• svd	: Calculate single value
• cond, condest, rcond	
	: Calculate conditional numbers
• norm	: Normality of the matrix

ADVANCED GRAPH

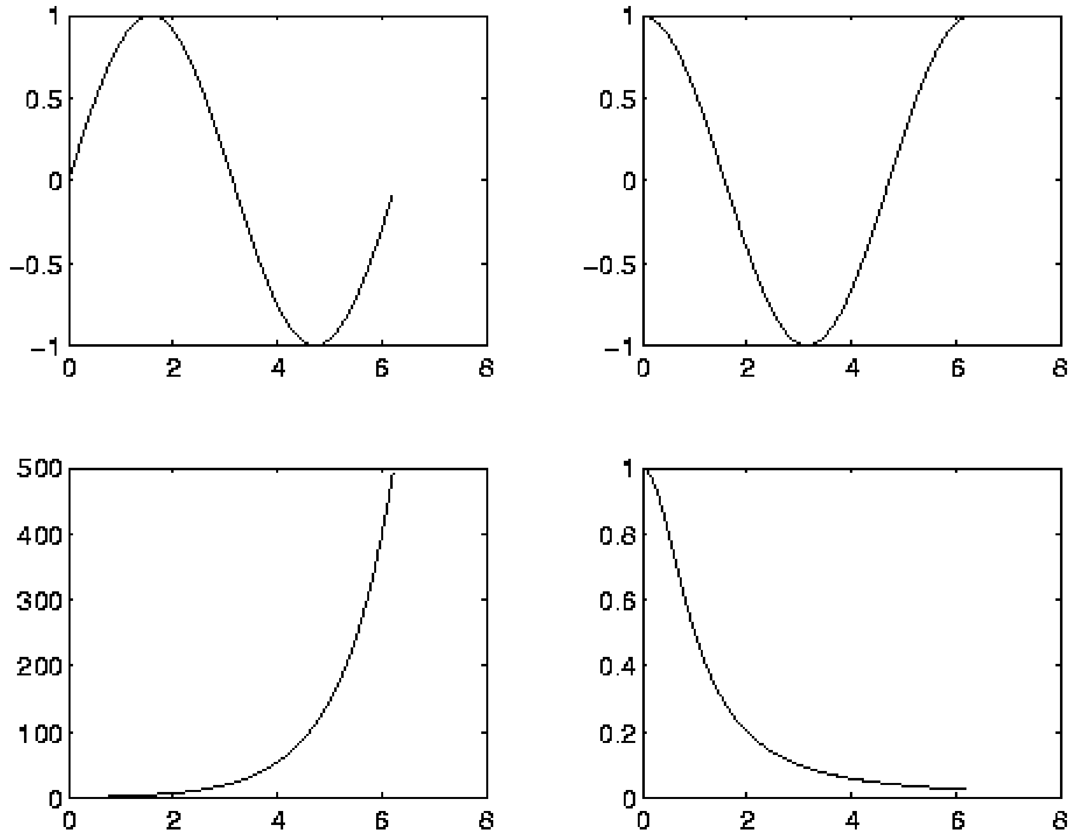
Plot Functions

- Plot
- Plot 2D curves
- Plot 3D surfaces
- Contour plots of 3D surfaces

Draw multiple graphs on the same window

```
>> t = (0:.1:2*pi)';
>> subplot(2,2,1)
>> plot(t,sin(t))
>> subplot(2,2,2)
>> plot(t,cos(t))
>> subplot(2,2,3)
>> plot(t,exp(t))
>> subplot(2,2,4)
>> plot(t,1./(1+t.^2))
```

Draw multiple graphs on the same window



Plot

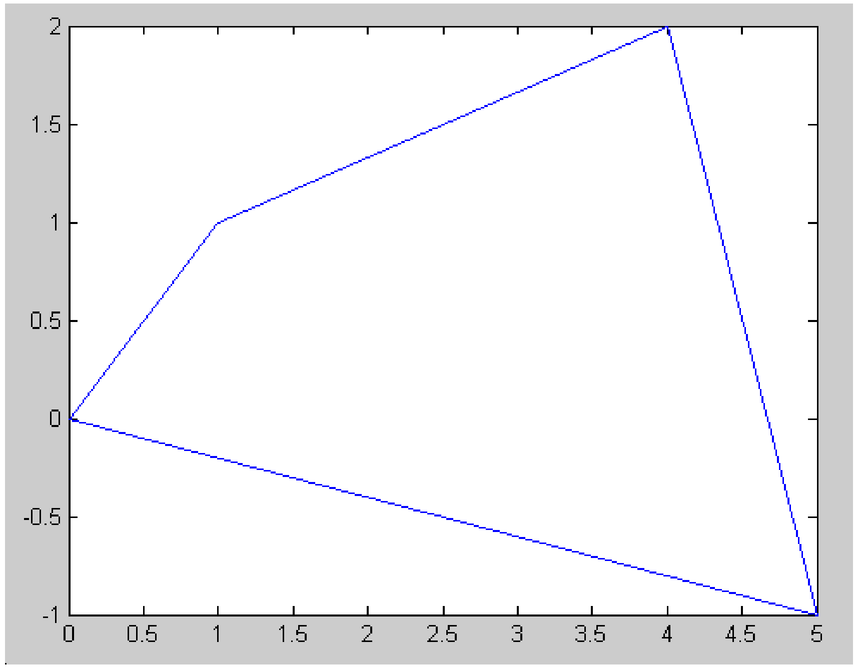
- **plot(x,y)**
- **plot(x1, y1, linestyle1,...,xn,yn,linestylen)**

Plot

Color	Marker	Line
y yellow	. (point)	- solid
m magenta	o (circle)	: dotted
c cyan	x (x-mark)	-. dashdot
r red	+ (plus)	-- dashed
g green	* (star)	
b blue	s (square)	
w white	d (diamond)	
k black	v (triangular)	
	^ (triangular)	

Plot

```
>> x=[0 1 4 5 0];
>> y=[0 1 2 -1 0];
>> plot(x,y)
```

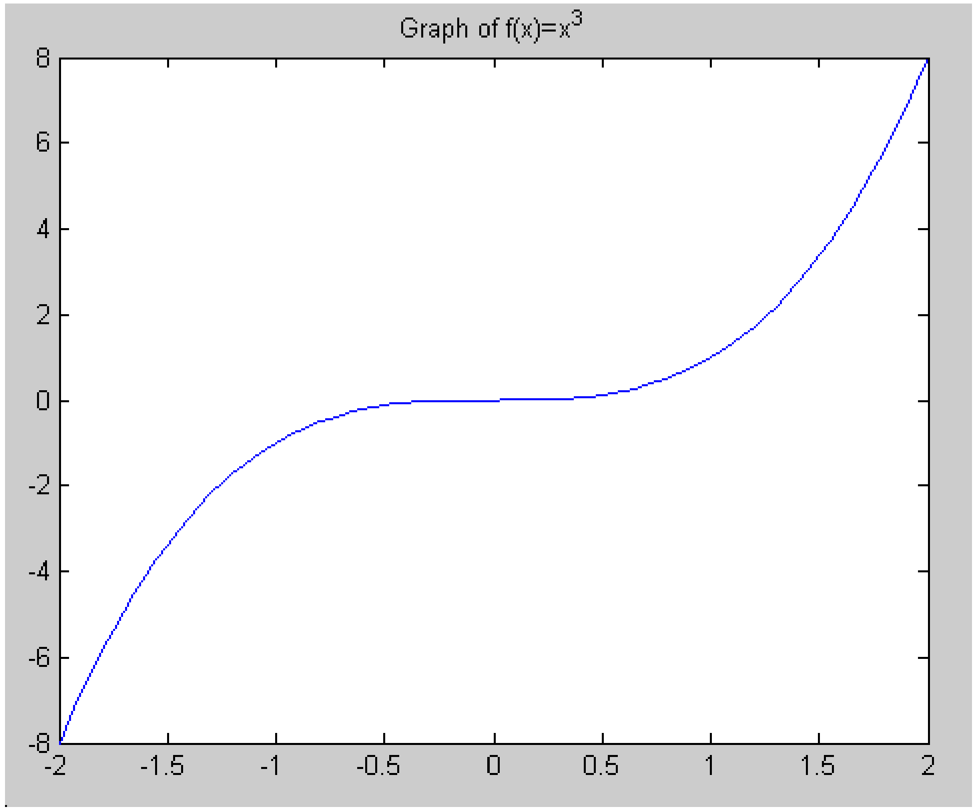


Plot

- Plot the function $y=x^3$ on [-2,2]

  - **>> x=-2:.05:2;**
  - **>> y=x.^3;**
  - **>> plot(x,y)**
  - **>> title('Graph of f(x)=x^3')**

Result



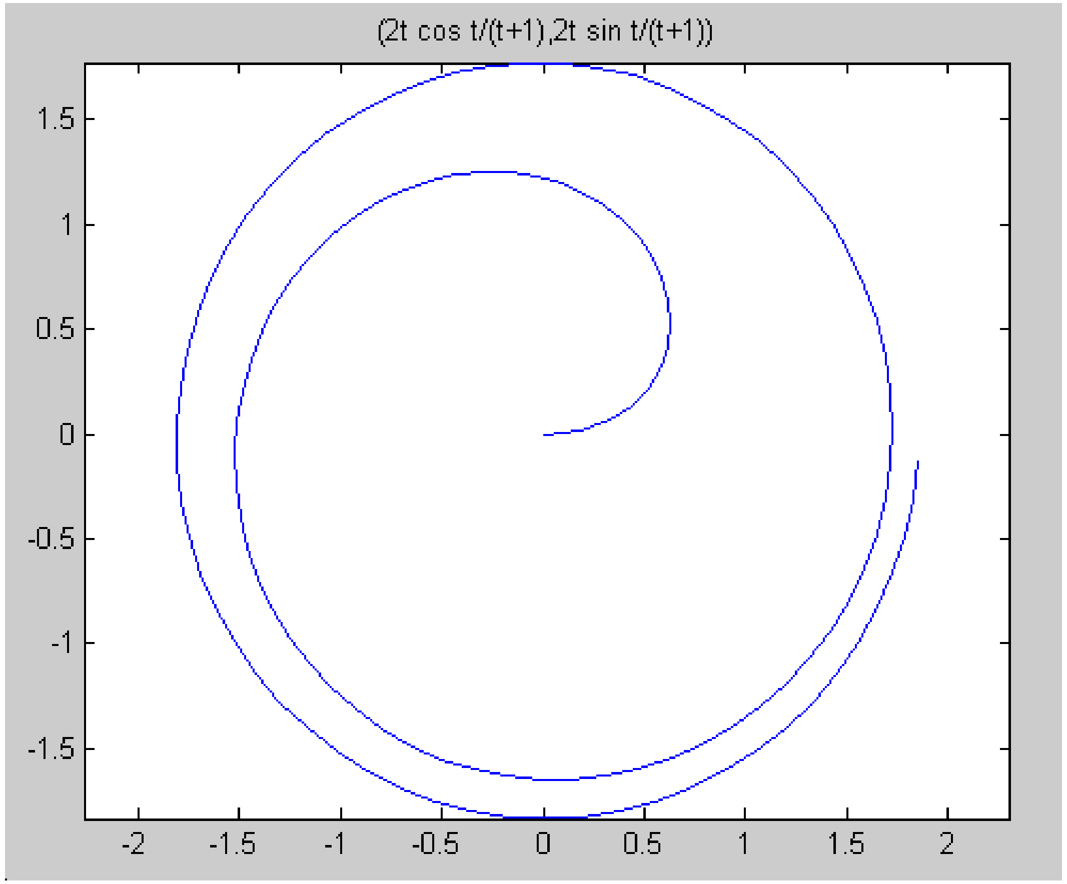
Plot the parametric curve

$\mathbf{r}(t) = (2t\cos t/(t+1), 2t\sin t/(t+1))$ víi $t \in [0,4\pi]$

```
>> t=0:.1:4*pi;
>> x=2*t.*cos(t)./(t+1);
>> y=2*t.*sin(t)./(t+1);
>> plot(x,y);
>> title('(2tcost/(t+1),2tsint/(t+1))')
```

Plot the parametric curve

```
>> axis equal
```



Plot

- Plot multiple curves on the same window:
  - **Use: hold on**

```
>> t=0:pi/20:2*pi;
>> plot(2*cos(t),2*sin(t))
>> hold on
>> plot(1+cos(t),1+sin(t))
>> axis equal
>> title('The circles x^2+y^2=4 and (x-1)^2
  +(y-1)^2=1')
```

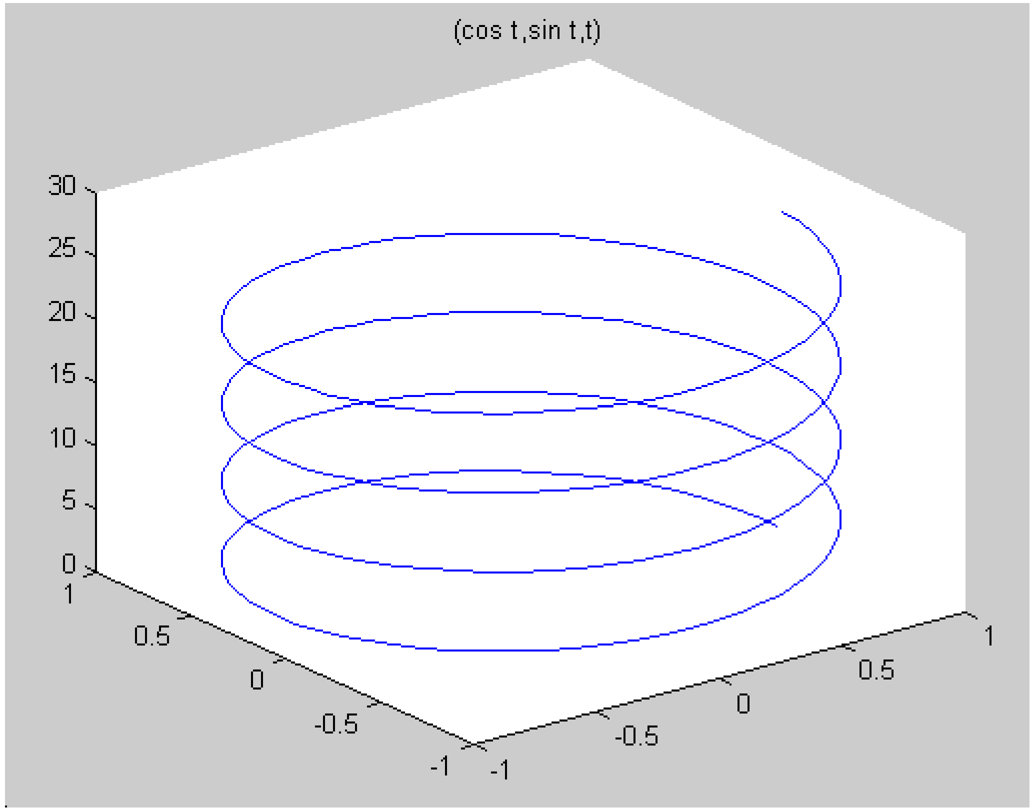
Draw 3D curves

plot3

r(t)=(cos(t),sin(t),t) víi $t \in [0,8\pi]$

```
>> t=0:.1:8*pi;
>> plot3(cos(t),sin(t),t)
>> title('(cos t,sin t,t)')
```

Draw 3D curves



Draw surface

- Draw graph of function $f(x,y)$ on:

$R = [a,b]\times[c,d] = \{(x,y) \mid a\leq x \leq b;\ c \leq y \leq d\},$

```
>> x=0:4;
>> y=0:.5:3;
>> [X,Y]= meshgrid(x,y))
```

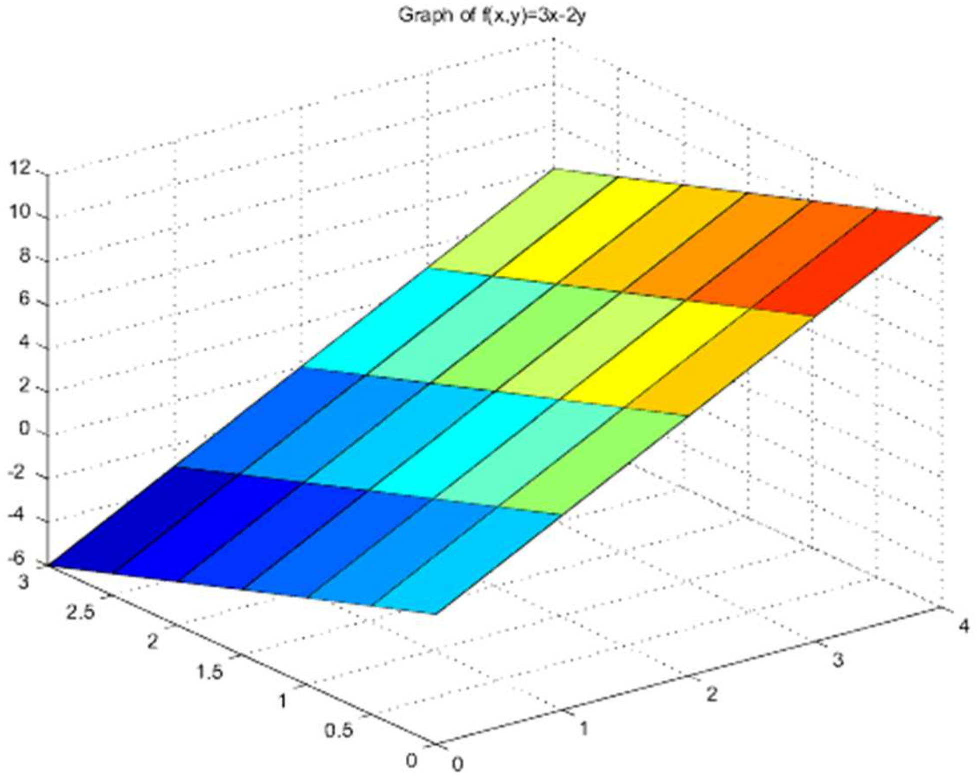
Draw surface

- E.g:

$$f(x,y)=3x - 2y.$$

```
>> Z=3*X-2*Y
>> surf(X,Y,Z)
>> title('Graph of f(x,y)=3x-2y')
```

Draw surface



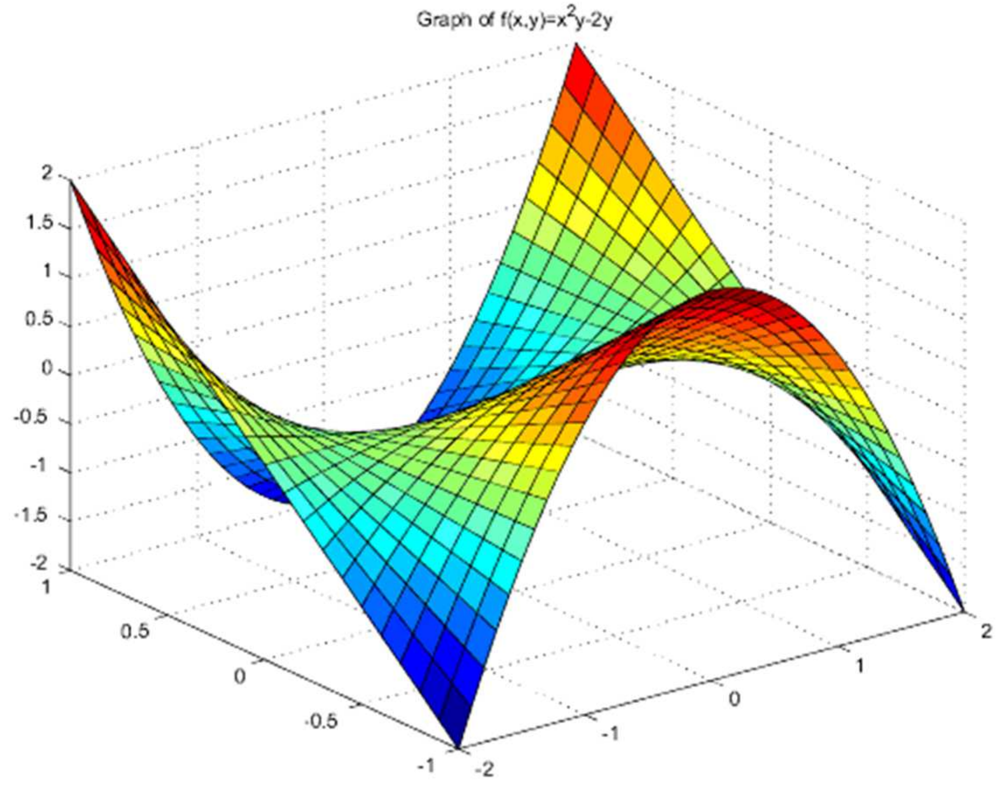
Draw surface

- E.g:

  $f(x,y)=x^2y - 2y$ **on [-2,2] × [-1,1].**

```
>> [X,Y]=meshgrid(-2:.1:2,-1:.1:1);
>> Z=(X.^2).*Y-2*Y;
>> surf(X,Y,Z)
>> title('Graph of f(x,y)=x^2y-2y')
```

Draw surface



POLYNOMIAL

Polynomial representation

- Polynomial of degree n

  $P = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

- Represented by a row vector of n+1 elements
  - Sort coefficients from highest order to 0
  - E.g: $p=3x^4 + x^2 - 3x+5$
    - `>>p=[3 0 1 -3 5]`
    - `p= 3 0 1 -3 5`

Functions with polynomials

conv	Convolution and polynomial multiplication
deconv	Deconvolution and polynomial division
poly	Polynomial with specified roots
polyder	Polynomial derivative
polyeig	Polynomial eigenvalue problem
polyfit	Polynomial curve fitting
polyint	Analytic polynomial integration
polyval	Polynomial evaluation
polyvalm	Matrix polynomial evaluation
residue	Convert between partial fraction expansion and polynomial coefficients
roots	Polynomial roots

Calculations with polynomials

- **E.g. 1:**

```
 >> r=[1 3];
>> p=poly(r)
p = 1 – 4  3

>>polyval(p,[1 3 5])
ans =
                0     0     8

>> printsys(r,p,'x')
 num/den =
       x + 3
   -------------
   x^2 - 4 x + 3
```

- **E.g. 2:**

$$\frac{x^4+3x-14}{x^2-4}=\frac{(x^4-16+3x+2)}{x^2-4}$$

$$=x^2+4+\frac{1}{x+2}+\frac{2}{x-2}$$

k(x)=x^2 +4

r=[1 2]

p=[-2 2]

Calculations with polynomials

E.g. 3:

$$G(s) = \frac{4s^2 + 16s + 12}{s^4 + 12s^3 + 44s^2 + 48s}$$

```
>> num=[4 16 12]
>> den=[1 12 44 48 0]
>> [z,p,k]=tf2zp(num,den)
   z =
       -3
       -1
   p =
            0
      -6.0000
      -4.0000
      -2.0000
   k =
        4
```

$$G(s) = \frac{K(s - z_1)(s - z_2)}{(s - p_1)(s - p_2)(s - p_3)} = \frac{4(s+3)(s+1)}{(s+6)(s+4)(s+2)}$$

ADVANCE INPUT/OUTPUT

Input/Output

- Enter data from the keyboard
  - Commands related to inputting data from the keyboard: ***input, keyboard, menu*** and ***pause***.
- Command ***input***:
  - This command gives a message prompting the user to enter data and receive input from the keyboard.
- The command has the form:

```
R = input(string)
```

where **R** is the variable name, **string** is the string containing the message prompting the user to know about the data to be loaded. Data can be any expression. If the user presses Enter immediately, **R** will be an empty matrix.

Enter data from the keyboard

- **E.g.:**

```
>> m=input('Enter the number of rows of
  the matrix: ')
Enter the number of rows of the matrix:
  3
m =
       3
>> n = input('Enter the number of
  columns of the matrix n =')
Enter the number of columns of the
  matrix n =5
n =
       5
```

Enter data from the keyboard

```
>> a =input('Enter the elements of the matrix in
  the format:\n [Row elements 1 ;\n    ...    \n Row
  elements m ;] : \n\n')
Enter the elements of the matrix in the format:
 [Row elements 1 ;
    ...
 Row elements m] :
[1 2 3 4 5;
6 7 8 9 10;
11 12 13 14 15]
```

Input/Output

- Command ***keyboard***
  - When this command is placed in the m-file it interrupts the execution of the program.
  - The screen appears prompt >>K.
  - User can view or change the values of variables using MATLAB commands.
  - Terminate the mode using **Enter**.
  - Convenient for debugging programs.

Input/Output

- Command ***menu***:
  - Creating selection table. The command has the form:

```
CHOICE = menu(HEADER,ITEM1,..., ITEMn)
```

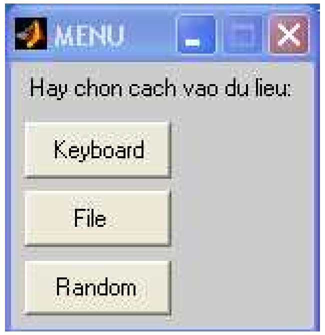
  - The command will display the title of the menu contained in the string HEADER, followed by the range of choices in the strings: ITEM1, ..., ITEMn.
  - The index of the selection will be returned to the CHOICE variable.
- In the graphical display MATLAB will display the menu in a MENU window with selection keys.

Input/Output

- **E.g.:**

```
>> CHOICE = menu('Hay chon cach vao du
 lieu: ','Keyboard', 'File   ','Random ` )
```

- Result:



- The user selects by clicking on the required option.

Input/Output

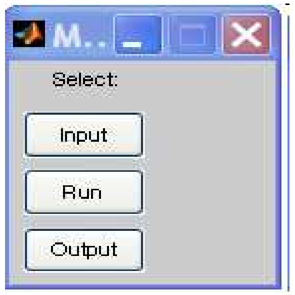
- We can use ***menu*** command

  ```
  CHOICE = menu(HEADER, ITEMLIST)
  ```

  in which ITEMLIST is a string.

- E.g.:

  ```
  >> ItemList={'Input','Run','Output'};
  >> HEADER='Select:';
  >> CHOICE = MENU(HEADER, ItemList)
  CHOICE =
       1
  ```



Print to screen

- Print an array to the screen
  - Type the array variable name without the ';' at the end
- E.g.:

```
>> x = [1 2 3];
>> y=[2; 3; 4];
>> x*y
ans =
    20
```

Print to screen

- Command ***disp***:
  - Used to display the contents of the variable to the screen and not display the variable name
  - Work despite having ';' or not
  - The command has the form:

    disp(X)

    where X is the variable or expression to be display.

```
>> B = [1 2 3; 4 5 6; 7 8 9]
>> disp(B^2);
    30     36     42
    66     81     96
   102    126    150
>> disp(B.^2);
     1      4      9
    16     25     36
    49     64     81
```

Print to screen

- E.g.:

```
clc
n = input('Give number of point n = ');
x = linspace(0,1,n);
y = sin(2*pi*x);
disp(' ')
disp('!   k     !   x(k) !  sin(x(k))  !')
disp('----------------------------------')
for k=1:n   degrees = (k-1)*360/(n-1);
disp(sprintf('!  %2.0f     !   %3.0f  !   %6.3f
  !',k,degrees,y(k)));
end
disp( ' ');
disp('x(k) is given in degrees.')
disp(sprintf('One Degree = %5.3e Radians',pi/180))
```

Print to screen

- Result

```
Give number of point n = 10
!   k     !   x(k) !  sin(x(k))  !
---------------------------------
!   1     !     0  !    0.000    !
!   2     !    40  !    0.643    !
!   3     !    80  !    0.985    !
!   4     !   120  !    0.866    !
!   5     !   160  !    0.342    !
!   6     !   200  !   -0.342    !
!   7     !   240  !   -0.866    !
!   8     !   280  !   -0.985    !
!   9     !   320  !   -0.643    !
!  10     !   360  !   -0.000    !
x(k) is given in degrees.
One Degree = 1.745e-002 Radians
```

Print to screen

- Command ***fprintf***

```
fprintf(dialog_format, list_of_variables)
```

where **dialog_format** is a variable (constant) of the message string and the output format characters.

Print to screen

- E.g.:

```
>> A = rand(3,3)
A =
    0.1389    0.6038    0.0153
    0.2028    0.2722    0.7468
    0.1987    0.1988    0.4451
>> fprintf('Length of matrix A: %i%i',length(A))
Length of matrix A: 3
>> fprintf('Square of A:\n
 %i%i%i\n%i%i%i\n%i%i%i\n',A^2)
Square of A:
 1.447541e-0012.317550e-0011.563636e-001
2.512430e-0013.449860e-0012.625931e-001
4.598234e-0015.387547e-0013.496177e-001
```

Input/Output with text files

- MATLAB provides commands that allow working with files
- The main commands related to working with MATLAB text files are:
  - fopen, fclose
  - frewind, fread, fwrite
  - fscanf, fprintf

Input/Output with text files

- ***fopen:*** Open a file

```
fileID = fopen(FILENAME, PERMISSION)
```

  - This command opens a text file named by FILENAME (character string constant or string variable)
  - The working mode is indicated by PERMISSION
  - PERMISSION can be:
    - 'rt' opens file for reading
    - 'wt' opens file for writing (if there is no file, a new file named FILENAME will be created)
    - 'at' opens file for extending (create a new file named FILENAME if not already there)
    - 'rt+' opens file for reading and writing (does not create new files)
  - The fileID identifier takes an integer value and we will use it to access the file.

Input/Output with text files

- If you want to check if the file opening is error, you can use the command

  ```
  [FID, MESSAGE] = fopen(FILENAME,PERMISSION)
  ```

  if there is an error, opening the MESSAGE variable file will contain an error message.

Input/Output with text files

- E.g.:

```
>> [fileID, MESSAGE] = fopen('ETU.txt','rt');
>> fileID
fileID =
     -1
>> MESSAGE
MESSAGE =
No such file or directory
```

Input/Output with text files

- ***fclose*: close the file.**

```
ST = fclose(FID)
```

  - This command will close the file corresponding to the identifier variable FID
  - ***fclose*** returns the ST variable the value 0 if it completes closing the file and -1 if it encounters an error.
  - The ST=***fclose***('all') command closes all open files except 0, 1, 2.
- ***frewind*: put the pointer of the FID at the beginning of the file**

```
frewind(FID)
```

Input/Output with text files

- ***fscanf***: Read data from file

```
[A, COUNT] = fscanf(FID, FORMAT, SIZE)
```

  - This command reads data from the file corresponding to FID
  - Converts data to the format specified by the FORMAT string variable
  - Assign value to array A
  - COUNT is an optional output variable used to contain the number of readable elements.
  - SIZE is an optional variable that only limits the number of elements to be read from the file, if absent, the entire file is considered. Possible values of the variable are:
    - N : reads no more than N elements from the file into the column vector
    - Inf : reads until the end of the file
    - [M, N] : reads no more than M*N elements and enters a matrix of size no more than MxN in columns, N can be Inf but M must be finite.

Input/Output with text files

- If the matrix A is the result of a character format conversion and the SIZE variable is not of the form [M, N] then the line vector is returned.
- FORMAT is a variable containing the format conversion characters of the C language.
- Formatting characters include: %, substitution symbols, field length, format conversion characters: d, i, o, u, x, e, f, g, s, c and […] (list).
- If %s is used, reading an element can result in the use of a series of elements of the matrix, each holding a character.
- Use %c to read spaces and the %s format to ignore spaces.

Input/Output with text files

- If the format variable contains both numbers and characters, the resulting matrix will be a numeric matrix and each character will be converted to a number by its ASCII code value.
- ***fscanf*** differs from this command in C is that it is a vectorization command that returns a matrix argument.
- The format string variable will be used repeatedly until the end of the file is reached or the number of elements specified by SIZE is read.

Input/Output with text files

- Command

```
S = fscanf(fid,'%s')
```

reads and returns a string.

- Command

```
A = fscanf(fid,'%5d')
```

reads integers with 5 digits

Input/Output with text files

- E.g.: Suppose there is a text file with the name "kq" containing the string “This Is Output". Then we can read the input as follows:

```
>> fid=fopen('kq','rt')
fid =
     3
>> x = fscanf(fid,'%s')
x =
ThisIsOutput
>> frewind(fid)
>> x = fscanf(fid,'%s%c')
x =
This Is Output
```

Input/Output with text files

- E.g.: Suppose the text file Matrix.txt contains two lines:

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

```
>> fopen('matrix.txt','rt')
ans =
         4
>> A=fscanf(4,'%i',[2,5])
A =
         1        3        5        7         9
         2        4        6        8        10
```

Input/Output with text files

```
>> frewind(4);
>> B = fscanf(4,'%i',[5,2])
B =
     1     6
     2     7
     3     8
     4     9
     5    10
>> frewind(4);
>> C = fscanf(4,'%i',6)
C =
     1
     2
     3
     4
     5
     6
```

Input/Output with text files

- ***fprint:*** write data to file.

  COUNT = FPRINTF(FID,FORMAT,A,...)
- This command will format the elements of matrix A (and subsequent variables in the list) in the format specified by the string variable format, and write to the file corresponding to the identifier variable FID
  - COUNT counts the number of bytes of data written.
  - FID is an integer that identifies the filename obtained from the ***fopen*** command. Number 1 can be used if standard output (monitor) is used. If this command does not have a FID, the default output will be displayed on the screen.
  - FORMAT is a string variable that contains characters that describe the data format as in the C language
  - The characters \n, \r, \t, \b, \f can be used to create linefeed, carriage return, tab, backspace, and formfeed characters, respectively.
  - Use \\ to create a \ and %% to create the % character.

Input/Output with text files

- E.g.:

```
x = 0:.1:1; y = [x; exp(x)];
  fid = fopen('exp.txt','w');
  fprintf(fid,'%6.2f  %12.8f\r',y);
  fclose(fid);
```

Create a text file named 'exp.txt' containing the table of values of the exponential function.

```
 0.00    1.00000000
0.10    1.10517092
0.20    1.22140276
0.30    1.34985881
0.40    1.49182470
0.50    1.64872127
 . . .
```