

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

# CHƯƠNG 1

# NHẬP MÔN MATLAB

Vũ Văn Thiệu, Đinh Viết Sang, Nguyễn Khánh Phương

TÍNH TOÁN KHOA HỌC

# Nội dung

1. Giới thiệu chung về MATLAB
2. Làm việc với MATLAB
3. Lập trình với MATLAB
4. Các phép tính ma trận nâng cao
5. Đồ thị nâng cao
6. Vào ra dữ liệu

# Giới thiệu chung về MATLAB

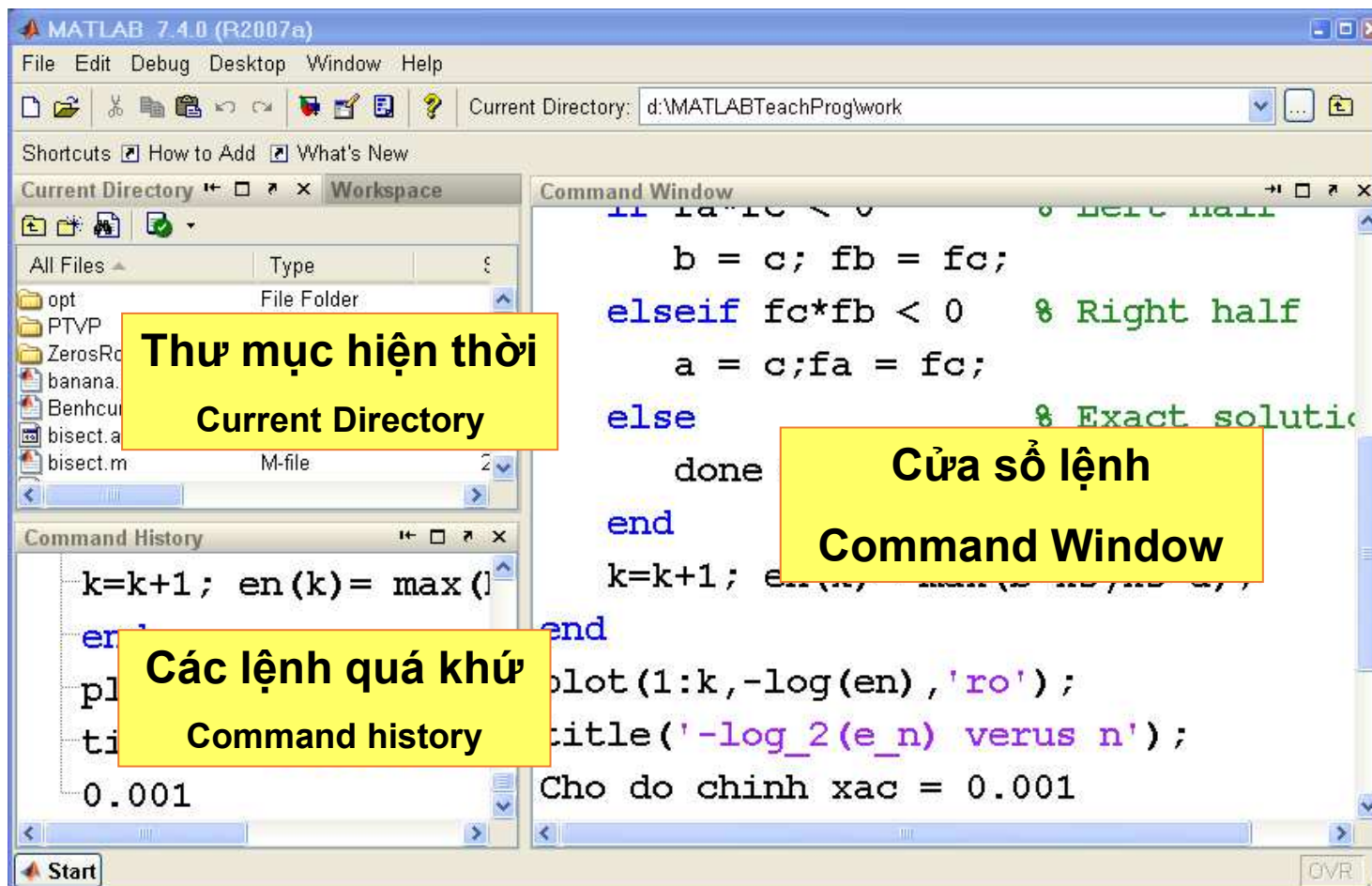
- MATLAB (Matrix Laboratory) là phần mềm của hãng MathWorks Inc.
- Đối tượng là các ma trận.
- MATLAB tích hợp các phương pháp tính toán, hiển thị và ngôn ngữ lập trình mạnh để cung cấp cho người sử dụng một môi trường làm việc thuận tiện để giải các vấn đề tính toán khoa học.
- Cấu trúc mở của MATLAB cho phép sử dụng MATLAB và các thành phần của nó để khảo sát dữ liệu, nghiên cứu các thuật toán và tạo các công cụ tiện ích của người sử dụng.

# Giới thiệu chung về MATLAB

- Matlab cũng đã tạo sẵn rất nhiều công cụ tiện ích như:
  - *Khai phá dữ liệu (Data acquisition)*
  - *Phân tích và khảo sát dữ liệu (Data analysis and exploration)*
  - *Hiển thị và xử lý ảnh (Visualization and image processing)*
  - *Dựng mẫu và Phát triển thuật toán (Algorithm prototyping and development)*
  - *Mô hình hóa và mô phỏng (Modeling and simulation)*
- MATLAB là công cụ được các nhà khoa học, kỹ sư sử dụng để phát triển các phần mềm giải các bài toán tính toán trong khoa học kỹ thuật.
- Bản thân MATLAB cũng cung cấp công cụ để giải nhiều bài toán của khoa học kỹ thuật.
- MATLAB được dùng trong nhiều trường đại học để hỗ trợ việc giảng dạy các giáo trình toán, đặc biệt là các giáo trình liên quan đến tính toán số như đại số tuyến tính ứng dụng, giải tích số, tính toán khoa học,

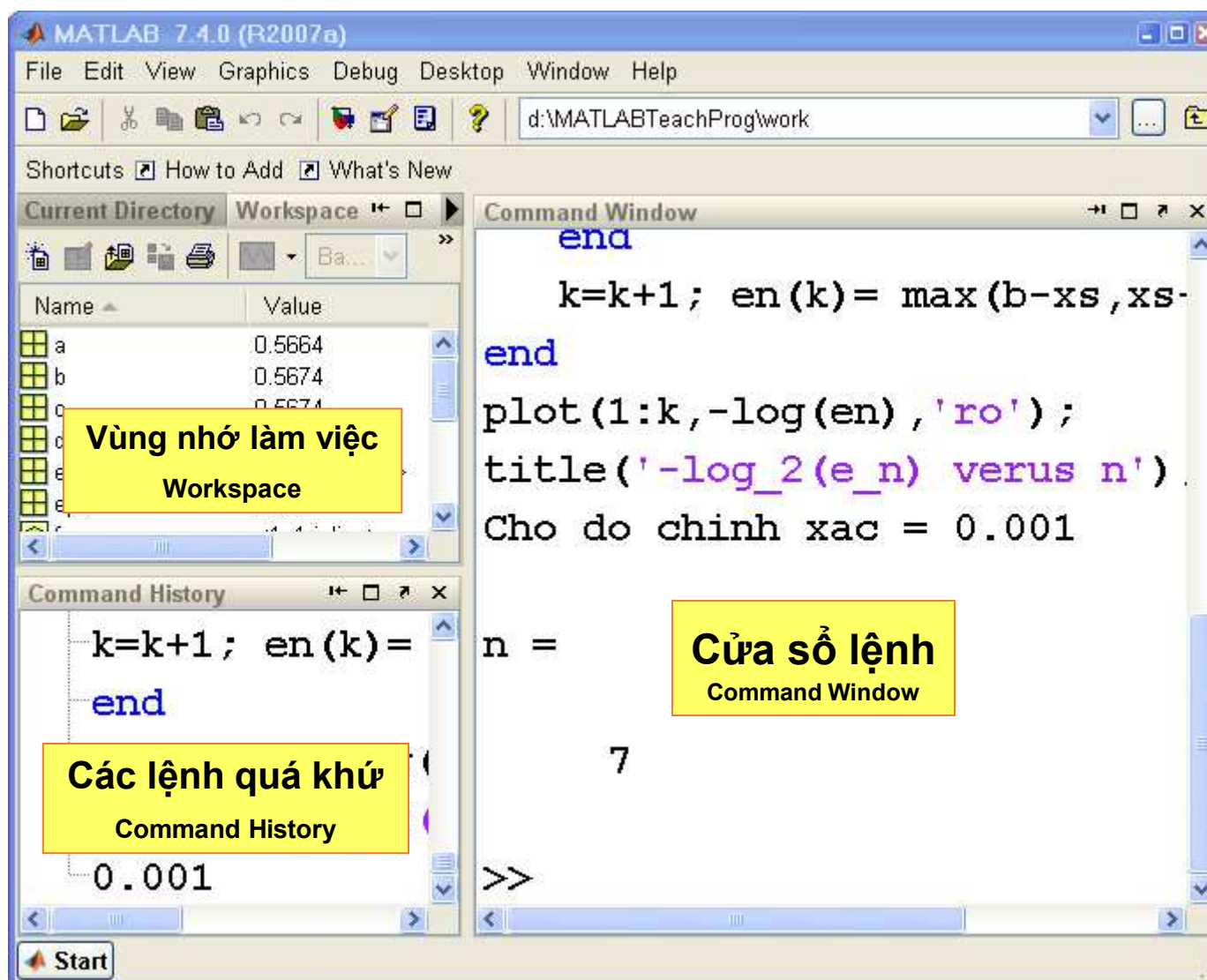
# LÀM VIỆC VỚI MATLAB

# Màn hình làm việc của Matlab





# Màn hình làm việc của Matlab



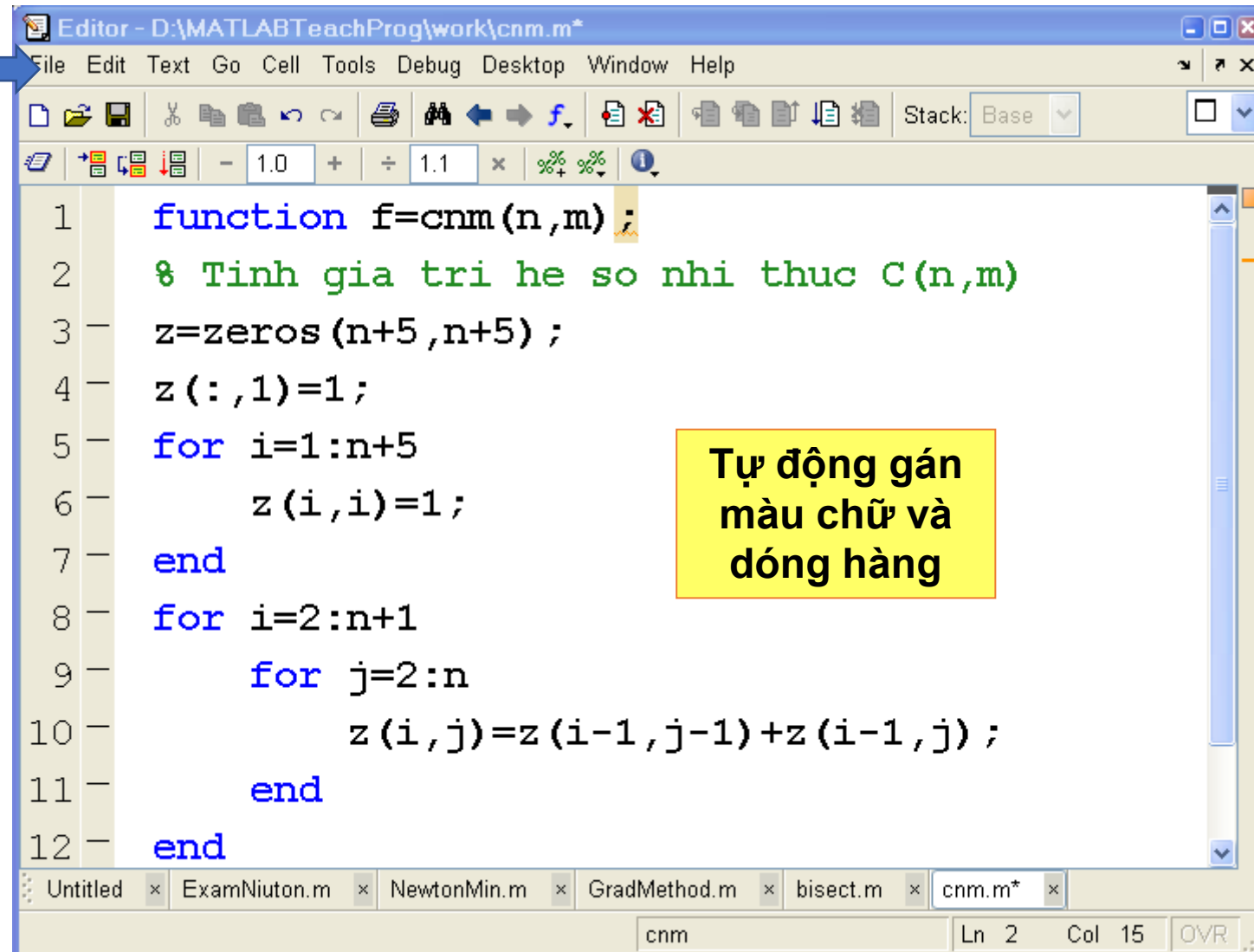


# Chương trình trên Matlab

- Matlab có thể làm việc như là một siêu máy tính cầm tay nếu chúng ta chỉ cần Matlab thực hiện một số lệnh bằng cách đánh trực tiếp trên cửa sổ lệnh...
- Chương trình được thực hiện bằng cách nào?
- Chương trình trong Matlab có thể là:
  - Kịch bản (Scripts), hoặc
  - Các hàm (Functions)
- **Scripts**: Dãy lệnh Matlab ghi trong một file được đưa vào cửa sổ lệnh và được thực hiện tức thì
- **Functions**: Các môđun chương trình tiếp nhận dữ liệu vào và trả lại kết quả (ví dụ hàm sin nhận đầu vào x và trả lại giá trị  $\sin(x)$ )
- Chương trình có thể được soạn thảo bằng bất cứ bộ soạn thảo văn bản nào (tuy nhiên Matlab cũng cung cấp bộ soạn thảo chương trình của riêng mình)

# Matlab Editor

Các chức  
năng



```
1 function f=cnm(n,m) ;
2 % Tinh gia tri he so nhi thuc C(n,m)
3 z=zeros(n+5,n+5) ;
4 z(:,1)=1;
5 for i=1:n+5
6     z(i,i)=1;
7 end
8 for i=2:n+1
9     for j=2:n
10         z(i,j)=z(i-1,j-1)+z(i-1,j) ;
11     end
12 end
```

Tự động gán  
màu chữ và  
đóng hàng

# Cơ cấu làm việc của Matlab

- Matlab là ngôn ngữ thông dịch (interpreted language)
  - Các câu lệnh được đánh trực tiếp trong **cửa sổ lệnh** và được thực hiện tức thì
  - Các biến được phân bổ bộ nhớ ngay lần đầu tiên chúng được khởi tạo
  - Muốn thực hiện lại một lệnh chỉ việc gõ lại lệnh đó
- Tất cả các biến được sử dụng trong cửa sổ lệnh được cất giữ vào **Vùng nhớ làm việc Base Workspace**
  - Có thể gán giá trị mới cho các biến nếu cần thiết
  - Có thể chọn để xóa bỏ một số biến khỏi vùng nhớ làm việc
  - Vùng nhớ làm việc có thể cất giữ vào một file dữ liệu
  - Phần mở rộng của file dữ liệu là **.mat** (ví dụ: **mydata.mat**)
  - File là file nhị phân
  - Các file dữ liệu (đuôi **.mat**) có thể nạp trở lại vào Vùng nhớ làm việc

# Câu lệnh, Chỉ thị & Biến

- Tại dấu nhắc của cửa sổ lệnh, người sử dụng có thể gõ:
  - **Lệnh (Command):**
    - `save mydata` (cất giữ vùng nhớ làm việc vào `mydata.mat`)
    - `whos` (hiển thị danh mục các biến trong vùng nhớ làm việc)
  - **Chỉ thị gán (Assignment Statement):**
    - `A = width * length;`
    - `B = 267;`
    - Câu lệnh gán chỉ có một tên biến ở vế trái của toán tử gán (=)
    - Vế phải sẽ được tính dựa vào giá trị hiện thời của các biến và kết quả tính được sẽ gán cho biến ở vế trái.
    - Giá trị có thể có dạng số hoặc dạng ký tự
    - **Kiểu của biến sẽ được cập nhật mỗi khi nó được gán giá trị**
- **Biến**
  - Phân biệt 31 ký tự đầu tiên; Phân biệt chữ hoa hay thường

# Làm việc trong chế độ hội thoại

- Khi sử dụng chế độ hội thoại, người sử dụng đánh trực tiếp câu lệnh vào sau dấu nhắc của MATLAB. Khi ấn nút “Enter”, dòng lệnh sẽ được thực hiện.
- Ví dụ,  
    `>> x = 1;`  
    `>> 4*atan(x)`
- Kết quả sẽ được đưa ra màn hình dưới dạng  
    `ans = 3.1416`
- Dấu chấm phẩy ";" ở cuối dòng lệnh được sử dụng để ngăn MATLAB không đưa kết quả của phép thao tác.

# Các từ khoá

- Matlab sử dụng một loạt các từ khoá (reserved words) mà để tránh xung đột, không nên sử dụng để đặt tên biến...

**for**

**end**

**if**

**while**

**function**

**return**

**elseif**

**case**

**otherwise**

**switch**

**continue**

**else**

**try**

**catch**

**global**

**persistent**

**break**

# Các tên biến và tên hàm chuẩn của MATLAB

Tên biến hằng	Mô tả
<code>ans</code>	Biến ngầm định chứa kết quả
<code>beep</code>	Phát tiếng kêu
<code>pi</code>	Hằng số pi
<code>eps</code>	Số 0 của Matlab
<code>inf</code>	infinity
<code>NaN</code>	not a number
<code>i</code> ( <i>hoặc</i> ) <code>j</code>	Đơn vị phức
<code>realmin, realmax</code>	Số thực dương nhỏ nhất và lớn nhất
<code>bitmax</code>	Số nguyên lớn nhất
<code>nargin, nargsout</code>	Số lượng biến vào/ra của lệnh gọi hàm
<code>varargin</code>	Số lượng biến trong lệnh gọi hàm
<code>varargout</code>	Số lượng biến đầu ra trong lệnh gọi hàm



# Các tên biến và tên hàm chuẩn của MATLAB

Tên hàm	Ý nghĩa
<b>abs (x)</b>	<b>Giá trị tuyệt đối</b>
<b>cos (x)</b>	<b>Cos</b>
<b>sin (x)</b>	<b>Sin</b>
<b>tan (x)</b>	<b>Tang</b>
<b>acos (x)</b>	<b>Arcos</b>
<b>asin(x)</b>	<b>Arsin</b>
<b>exp (x)</b>	<b>Hàm mũ của e</b>

# Các tên biến và tên hàm chuẩn của MATLAB

Tên hàm	Ý nghĩa
<b>imag (x)</b>	phần ảo của số phức
<b>log (x)</b>	<b>Logarithm cơ số e</b>
<b>log10 (x)</b>	Logarithm cơ số 10
<b>real (x)</b>	<b>phần thực của x</b>
<b>sign (x)</b>	Hàm dấu, trả lại dấu của đối số
<b>pow (x,y)</b>	<b>hàm mũ</b>
<b>sqrt (x)</b>	Căn bậc hai
<b>tan (x)</b>	<b>hàm tang</b>

# Khuôn dạng dữ liệu

- Mặc dù tất cả các tính toán số trong Matlab đều được thực hiện với độ chính xác kép (double precision), nhưng khuôn dạng của dữ liệu đưa ra có thể định dạng lại nhờ các lệnh định dạng của Matlab.
- Các biến ngầm định cũng như các biến của người sử dụng định nghĩa đều có thể đưa ra theo nhiều khuôn dạng khác nhau.
- Khuôn dạng được chọn nhờ sử dụng lệnh format:
  - **FORMAT SHORT** số dấu phẩy động có 4 chữ số sau dấu.
  - **FORMAT LONG** số dấu phẩy động có 14 chữ số.
  - **FORMAT SHORTE** số dấu phẩy động có 4 chữ số với số mũ.
  - **FORMAT LONGE** số dấu phẩy động có 15 chữ số với số mũ
  - **FORMAT RAT** biểu diễn đúng hoặc gần đúng dưới dạng phân số

# Khuôn dạng dữ liệu: Ví dụ

**>> 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**

# Khởi tạo biến vector và ma trận

- Một trong những điểm mạnh của MATLAB là nó cho phép làm việc với các ma trận và vector. Để sử dụng một biến ta cần khởi tạo nó. Có thể khởi tạo biến vector và ma trận theo nhiều cách.
- Đối với vector (hay ma trận chỉ có một dòng) ta có thể sử dụng các cách khởi tạo sau

```
>> 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];
```

# Khởi tạo biến vector và ma trận

- Một trong những cách khởi tạo vector thường dùng là sử dụng toán tử

first : increment : last

- Câu lệnh:

a= first : increment : last

khởi tạo vector a bắt đầu từ phần tử *first* và kết thúc tại phần tử *last* với độ dài bước là *increment*. Nếu không chỉ ra *increment*, thì giá trị ngầm định nó là bằng 1.

# Khởi tạo biến vector và ma trận

- Ví dụ:

1) Để khởi tạo vector  $a = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$  có thể thực hiện

```
>> a=1:10
```

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

2) Độ dài bước có thể là số âm

```
>> a=[100:-10:20]
```

```
a = 100 90 80 70 60 50 40 30 20
```

3) Các giá trị của thông số có thể xác định bởi biểu thức toán học

```
>> 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
```



# Khởi tạo biến vector và ma trận

- Để khởi tạo ma trận ta có thể làm như sau

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

```
>> B = [x; y; z];
```

```
>> C = [A; 10 11 12];
```

- Chú ý là các dòng của ma trận được phân tách nhau bởi dấu ";", trong khi đó các phần tử trên một dòng được phân tách bởi dấu cách (hoặc dấu phẩy).

# Địa chỉ của mảng

- Các phần tử của một mảng tổng quát (là vectơ hay ma trận) có thể địa chỉ hóa theo nhiều cách.
- Cách đơn giản nhất là chỉ ra phần tử bởi vị trí dòng và cột của nó trong mảng
- Các dòng và cột được đánh số bắt đầu từ 1.

# Địa chỉ của mảng

- Ví dụ

```
>> 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
```

# Địa chỉ của mảng

- Lấy một hàng

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

- Lấy một cột

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

- Lấy một ma trận con trong một ma trận lớn

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

- Đổi vị trí các hàng hoặc cột

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

# Địa chỉ của mảng

- Ghép 2 ma trận theo cột  
     $\gg C = [A \ B];$
- Ghép 2 ma trận theo hàng  
     $\gg C = [A; B];$
- Xóa một cột  
     $\gg A(:,2) = [];$

# Các phép toán với vector và ma trận

- Tính độ dài của véc tơ (đếm số phần tử)

**length**

- Tích thước của ma trận

**size**

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

# Các phép toán với vector và ma trận

- Cộng (và trừ) các ma trận cùng kích thước được thực hiện từng thành phần.

- Ví dụ

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

```
>> A+B
```

```
ans = 7 1 3  
      8 4 10
```

- Chú ý các ma trận phải có cùng kích thước:

```
>> C=[3 1; 6 4];
```

```
>> A+C
```

```
??? Error using ==> + Matrix dimensions  
must agree.
```



# Các phép toán với vector và ma trận

- Nhân với một số (chia cho một số khác không) cũng được thực hiện theo từng thành phần. Ví dụ:

```
>> A=[5 -1 2; 3 4 7];
```

```
>> 2*A
```

```
ans =
```

```
10 -2 4
```

```
6 8 14
```

- Phép toán \* trong tích trên là bắt buộc phải có:

```
>> 2A
```

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

# Các phép toán với vector và ma trận

- Cộng vector và nhân vector với một số được thực hiện tương tự.

```
>> v=[3; 5]; w=[-2; 7];
```

```
>> 10*v-5*w
```

```
ans =
```

```
40
```

```
15
```

# Các phép toán với vector và ma trận

- Phép nhân hai ma trận cũng có thể thực hiện được trên Matlab.
- Để nhân hai ma trận hay nhân ma trận với vector chỉ việc dùng toán tử  $*$  giống như phép nhân của các đại lượng vô hướng.
- Matlab nhận dạng kích thước của đầu vào và thực hiện phép nhân.
- Điều mà người sử dụng quan tâm là kích thước của các ma trận và vector phải phù hợp để có thể thực hiện được phép toán.

# Các phép toán với vector và ma trận

- Ví dụ:

```
>> 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
```

# Các phép toán với vector và ma trận

- Chuyển vị véc tơ

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

```
>> A'
```

```
ans =
```

```
4    5    6
```

```
5    4    3
```

# Các phép toán với vector và ma trận

- **Tích theo từng thành phần** của hai ma trận cùng kích thước A và B là ma trận  $A.*B$  với các phần tử là tích của các phần tử tương ứng của A và B.
- Ví dụ:

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

```
>> A.*B
```

```
ans = 3    4    3  
      -4   10   12
```

# Các phép toán với vector và ma trận

- **Phép chia và lũy thừa theo từng thành phần:**  
 $A./B$  và  $A.^B$  được định nghĩa tương tự. Lưu ý là các phép tính với các thành phần phải là có nghĩa, nếu không MATLAB sẽ báo lỗi.
- Ví dụ:

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

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



# Các phép toán với vector và ma trận

- **Phép cộng ma trận với vô hướng:** Giá trị của vô hướng được cộng vào từng thành phần của ma trận.
- **Ví dụ:**

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

```
>> A+5
```

```
ans =
```

6	7	8
7	8	9

# Các hàm véc tơ hóa

- Các hàm của Matlab đều được véc tơ hóa. Nghĩa là nếu input là một mảng thì output cũng là một mảng

- **Ví dụ:** Vẽ đồ thị hàm  $y=\sin(x)$  trên đoạn  $[0, 2*\pi]$

```
>> x = (0:.1:2*pi) ;
```

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

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

- **Ví dụ:**

```
x = (-5:.1:5) ;
```

```
>> y = x./(1+x.^2) ;
```

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

# Các hàm tạo ma trận đặc biệt

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

# Biến xâu trong MATLAB

- MATLAB cho phép sử dụng biến xâu ký tự: Sử dụng lệnh gán

**S = 'Any Characters'**

cho phép tạo mảng ký tự (xâu ký tự).

- Ví dụ:

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

```
msg =
```

```
You're right!
```

- Lưu ý: Để tạo dấu ' trong xâu ký tự cần đánh hai dấu '

# Biến chuỗi trong MATLAB

- Lệnh  $S = [S1 \ S2 \ \dots]$  ghép các chuỗi  $S1, S2, \dots$  thành một chuỗi mới  $S$ .
- Ví dụ:

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

- Lưu ý đến cách truy cập đến các thành phần trong chuỗi.

# Biến chuỗi trong MATLAB

- $S = \text{char}(X)$  tạo S là ký tự có mã ASCII là X.
- $X = \text{double}(S)$  chuyển ký tự thành số
- Ví dụ:

```
>> char([65 66 67])
```

```
ans =
```

```
ABC
```

```
>> X = double('ABC')
```

```
X =
```

```
65    66    67
```

# Biến chuỗi trong MATLAB

- Để khởi tạo biến mảng mà mỗi phần tử là một chuỗi, sử dụng:

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

# LẬP TRÌNH TRÊN MATLAB



# Câu lệnh trong Matlab

- Câu lệnh hay gộp nhất trong Matlab có dạng  
***variable = expression***

Câu lệnh này sẽ gán giá trị của biểu thức ***expression*** cho biến ***variable***.

Ví dụ:

```
>> x=2^10*pi
```

```
x =
```

```
3.216990877275948e+003
```

# Câu lệnh trong Matlab

- Câu lệnh của Matlab cũng còn có thể có dạng đơn giản như sau:

***expression***

trong trường hợp này giá trị của biểu thức sẽ được gán cho biến ngầm định có tên là ***ans***.

**Ví dụ:**

```
>> 2*pi*exp(-5)
```

```
ans =
```

```
0.04233576958521
```

# Câu lệnh trong Matlab

- Cuối cùng, một dạng nữa của câu lệnh trong Matlab là *variable*
  - Nếu như biến đã được gán giá trị trước đó thì nội dung của biến được đưa ra màn hình,
  - Nếu trái lại sẽ có thông báo rằng biến chưa được xác định.
- Người sử dụng có thể tận dụng điều này để kiểm tra xem một tên biến đã được dùng hay chưa.

# Câu lệnh trong Matlab

- Câu lệnh của Matlab sẽ được thực hiện ngay sau khi nhấn phím Enter. Nếu câu lệnh Matlab kết thúc bởi Enter, theo ngầm định, Matlab sẽ đưa kết quả thực hiện lên thiết bị ra chuẩn (ngầm định là màn hình).
- Muốn tránh việc đưa kết quả ra ngay trực tiếp sau câu lệnh, cần kết thúc câu lệnh bởi dấu ; sau đó mới đến Enter.
- Khi câu lệnh của Matlab quá dài có thể ngắt thành hai hoặc nhiều dòng sử dụng dấu nối dòng ... ở cuối mỗi dòng chứa câu lệnh.

# Câu lệnh trong Matlab

- **Ví dụ:**

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

- Để xóa tất cả biến trong Matlab, ta dùng lệnh

```
>> clear
```

- Để xóa một số biến cụ thể, ta dùng lệnh

```
>> clear var1 var 2 ...
```

trong đó *var1*, *var2*, .. Là các tên của cá biến cần xóa.

# Các phép toán quan hệ và logic

Phép toán quan hệ	Ý nghĩa
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
>	Lớn hơn
>=	Lớn hơn hoặc bằng
==	Bằng
~=	Không bằng
Phép toán logic	Ý nghĩa
&	AND (hội)
	OR (tuyển)
~	NOT (phủ định)
0	FALSE
Số khác không	TRUE

# Câu lệnh if

- Dạng tổng quát của câu lệnh if là

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

elseif và else là tùy chọn

- Nhóm lệnh đi ngay sau biểu thức (*expr*) đầu tiên có giá trị khác 0 sẽ được thực hiện.
- Nếu không có *expr* nào khác 0 thì nhóm lệnh sau else được thực hiện

# Câu lệnh if

- Ví dụ:

```
>> 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
```



# Câu lệnh if

- Các phép toán quan hệ:

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

- Ví dụ:

>> 5>3

ans =

1

>> 5<3

ans =

0

>> 5==3

ans =

0

# Câu lệnh vòng lặp for

- Vòng lặp for lặp lại các câu lệnh trong thân của nó đối với các giá trị của biến chạy được lấy từ một vector dòng cho trước.
- Ví dụ

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

# Câu lệnh vòng lặp for

- Chú ý đến việc sử dụng hàm nội trú **disp**: hàm này đưa ra màn hình nội dung của biến.
- Vòng lặp, cũng giống như câu lệnh if, phải kết thúc bởi **end**. Vòng lặp ở trên thường được viết dưới dạng như sau.

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

- Nhớ lại cách khởi tạo 1:4 cũng chính là [1, 2, 3, 4].

# Câu lệnh vòng lặp for

- Đoạn lệnh

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

sẽ tạo ra vector  $x = [1, 4, 9, 16]$ .

- Đoạn lệnh

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

sẽ tạo ra vector  $x = [16, 9, 4, 1]$ .

# Câu lệnh while

- Câu lệnh có dạng sau:

```
while  expr  
    Statements  
end
```

- Các câu lệnh trong thân của vòng lặp while sẽ được lặp lại chừng nào biểu thức *expr* còn là true (có giá trị khác 0).

# Câu lệnh while

- Ví dụ:

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

- Chú ý: Về mặt chính xác toán học thì  $1 + x > 1$  đối với mọi  $x > 0$ .

# Câu lệnh switch

- Câu lệnh **switch** cho phép thực hiện rẽ nhánh dựa trên giá trị biểu thức.
- Dạng tổng quát của câu lệnh **switch** là:

**switch** biethuc

**case** giatri

        cac cau lenh

**case** {giatri1, giatri2, giatri3, ...}

        cac cau lenh

    ...

**otherwise**

        cac cau lenh

**end**

## Ví dụ:

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

Kết quả: the weekend.



# Câu lệnh break

- Lệnh break dùng để ngắt việc thực hiện vòng lặp while hoặc for.
- Trong vòng lặp lồng nhau, break chỉ thoát khỏi vòng lặp trong cùng
- Nếu sử dụng break ngoài vòng lặp trong một kịch bản, nó sẽ chấm dứt thực hiện kịch bản
- Lệnh break trong cấu trúc IF hoặc Switch Case sẽ chấm dứt thực hiện câu lệnh đó

# Kịch bản (Script)

- Kịch bản là dãy các câu lệnh của Matlab được ghi lại trong một m – file, file đuôi .m.
- Khi đánh tên của file (không cần .m), dãy các lệnh này sẽ được thực hiện.
- Lưu ý: m – file phải được đặt tại một trong các thư mục mà Matlab sẽ tự động tìm kiếm m – file trong đó; danh mục các thư mục như thế có thể xem nhờ lệnh **path**.
- Một trong những thư mục mà Matlab luôn khảo sát chính là thư mục hiện thời. Thư mục này được hiển thị trong cửa sổ Current Directory.
- Người sử dụng có thể thay đổi thư mục hiện thời nhờ dùng các tiện ích trong cửa sổ này.
- **Câu hỏi:** thay đổi thư mục hiện thời như thế nào?

# Kịch bản: Ví dụ

- Ví dụ kịch bản: `plotsin.m`

**`x = 0:2*pi/N:2*pi;`**

**`y = sin(w*x);`**

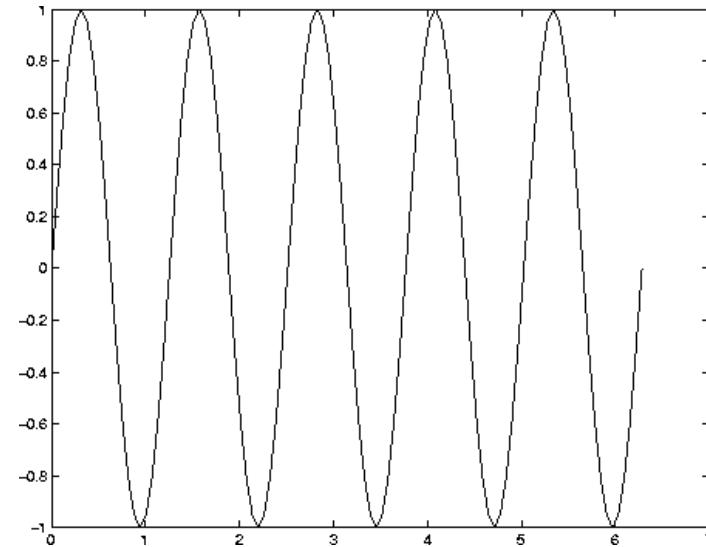
**`plot(x,y)`**

- Khi gõ lệnh

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

**`>> plotsin`**

sẽ ra đồ thị sau:



# Hàm (function)

- Cách định nghĩa hàm:

**function [out1,out2,...] = funcname(inp1, inp2,...)**

# Ví dụ

- Lập Hàm giải phương trình bậc 2

# CÁC PHÉP TÍNH MA TRẬN NÂNG CAO

# Các phép toán nâng cao

- Matlab cung cấp nhiều hàm liên quan đến xử lý ma trận như:
  - eig : tính trị riêng
  - lu : phân tích LU
  - chol : phân tích Cholesky
  - qr : phân tích qr
  - svd : tính single value
  - cond, condest, rcond: tính các số điều kiện
  - norm : tính chuẩn của ma trận

# Đồ họa nâng cao

- Vẽ đường cong 2D (2D curves),
- Vẽ mặt 3D (3D surfaces),
- Vẽ đường cong 3D (contour plots of 3D surfaces),

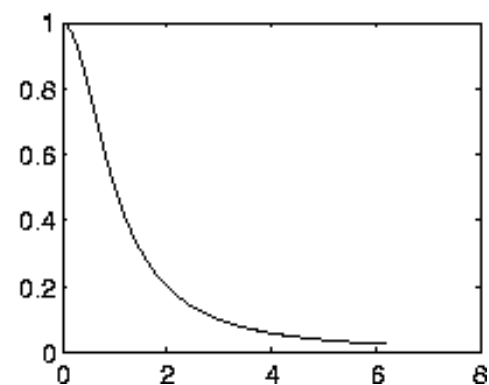
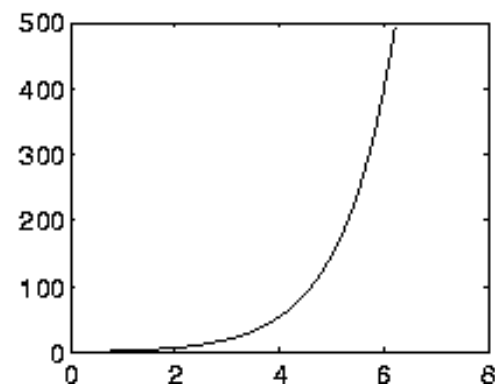
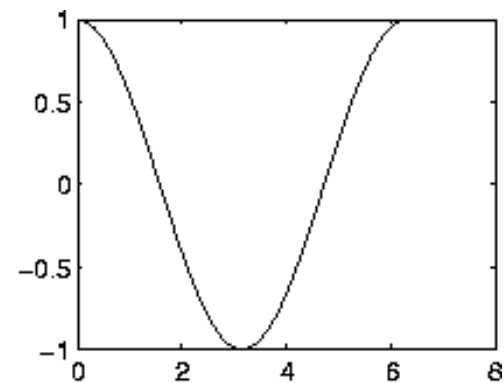
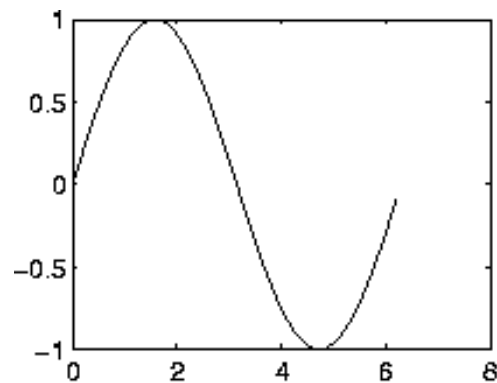


# Vẽ nhiều đồ thị trên cùng một cửa sổ

- Ví dụ:

```
>> 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))
```

# Đưa ra nhiều đồ thị



# Vẽ đường

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

# Vẽ đường

Màu nét vẽ	Ký tự đánh dấu	Nét vẽ
y yellow m magenta c cyan r red g green b blue w white k black	. (point) o (circle) x (x-mark) + (plus) * (star) s (square) d (diamond) v (triangular) ^ (triangular)	- solid : dotted -. dashdot -- dashed

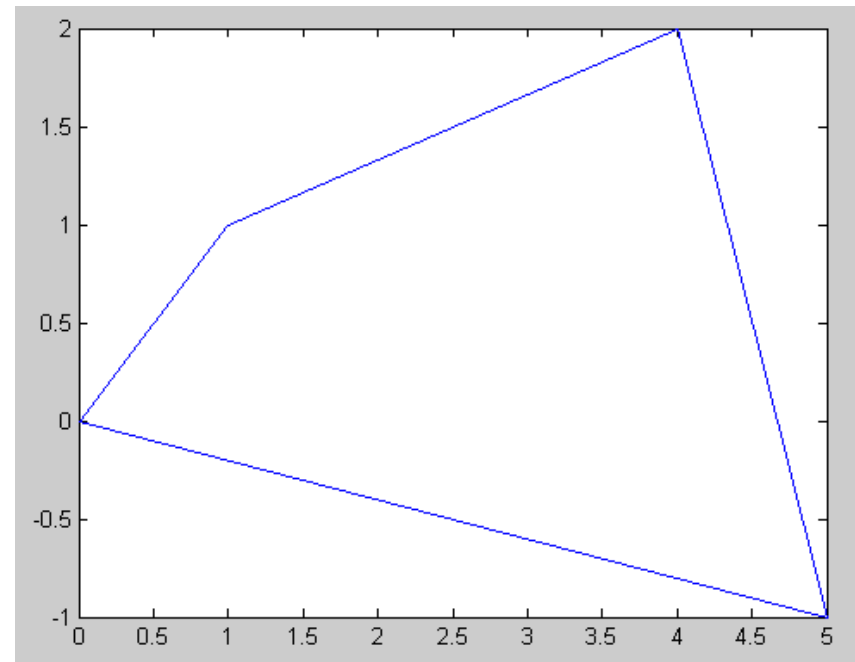
# Vẽ đường

- Ví dụ:

```
>> x=[0 1 4 5 0];
```

```
>> y=[0 1 2 -1 0];
```

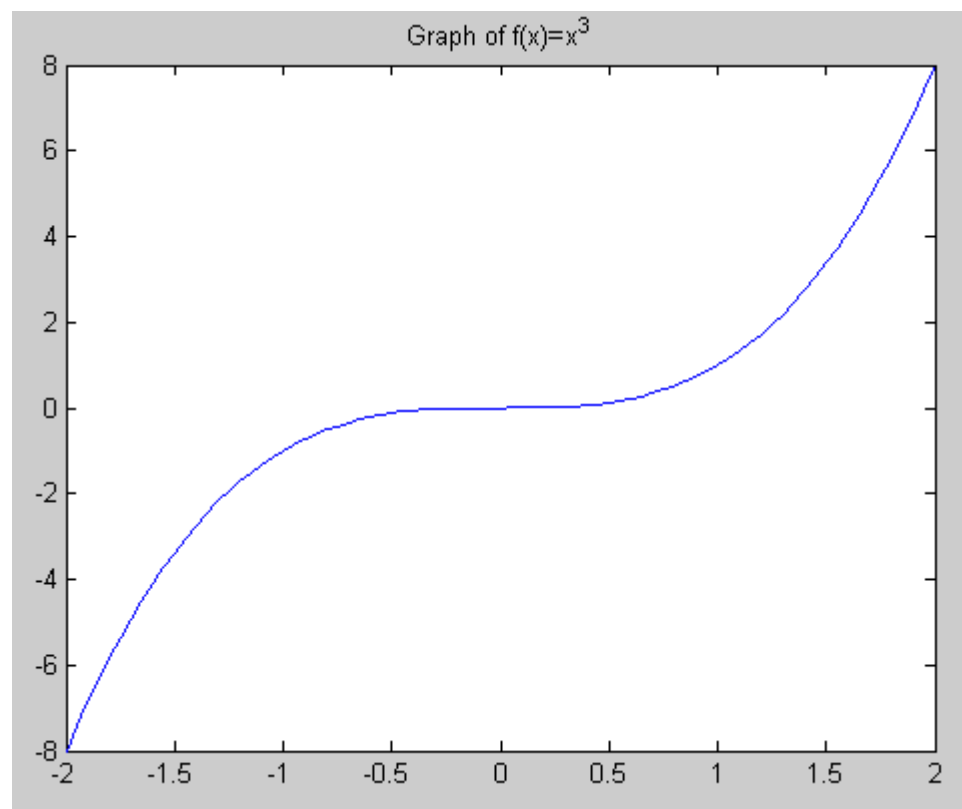
```
>> plot(x,y)
```



# Vẽ đường

- Ví dụ: Vẽ đồ thị hàm số  $y=x^3$  trên  $[-2,2]$ ,
  - `>> x=-2:.05:2;`
  - `>> y=x.^3;`
  - `>> plot(x,y)`
  - `>> title('Graph of f(x)=x^3')`

# Đồ thị hàm số $y = x^3$



# Vẽ đường cong tham số

- Ví dụ: Vẽ đường cong tham số:

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

```
>> t=0:.1:4*pi;
```

```
>> x=2*t.*cos(t)./(t+1);
```

```
>> y=2*t.*sin(t)./(t+1);
```

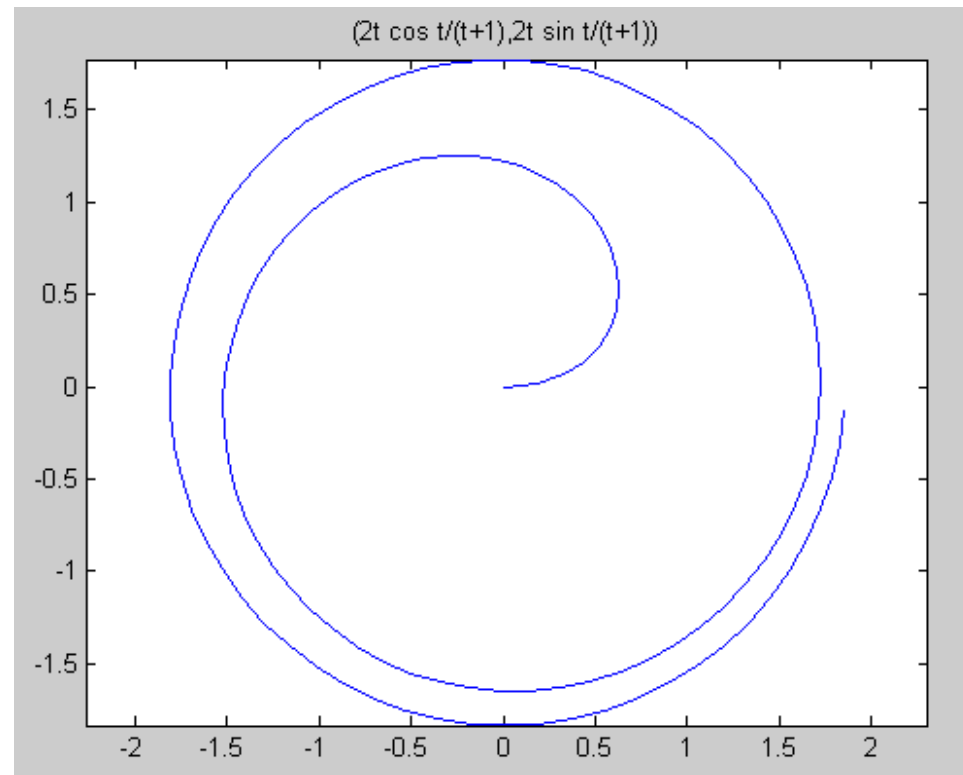
```
>> plot(x,y);
```

```
>> title('(2tcos/(t+1),2tsint/(t+1))')
```



# Vẽ đường cong tham số

- Cân bằng trục  
    >> **axis equal**



# Vẽ đường

- Vẽ nhiều đường trên một khung hình:

- Dùng 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  $(x-1)^2+(y-1)^2=1$ ')
```

# Vẽ đường cong 3D

- **plot3**
- **Ví dụ:**

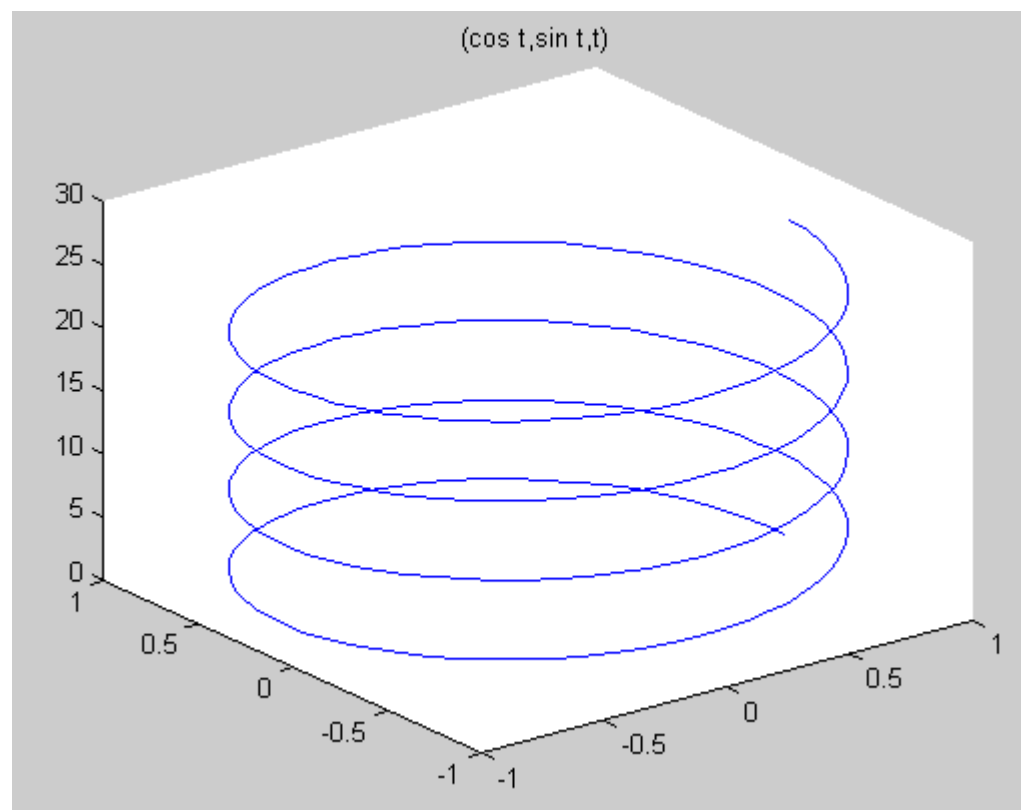
$\mathbf{r}(t) = (\cos(t), \sin(t), t)$  với  $t \in [0, 8\pi]$

```
>> t=0:.1:8*pi;
```

```
>> plot3(cos(t),sin(t),t)
```

```
>> title('(cos t,sin t,t)')
```

# Vẽ đường cong 3D



# Vẽ mặt

- Vẽ đồ thị hàm số  $f(x,y)$  trên hình:

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

```
>> x=0:4;
```

```
>> y=0:.5:3;
```

```
>> [X,Y]=meshgrid(x,y)
```

# Vẽ mặt

- Ví dụ: vẽ đồ thị hàm số sau:

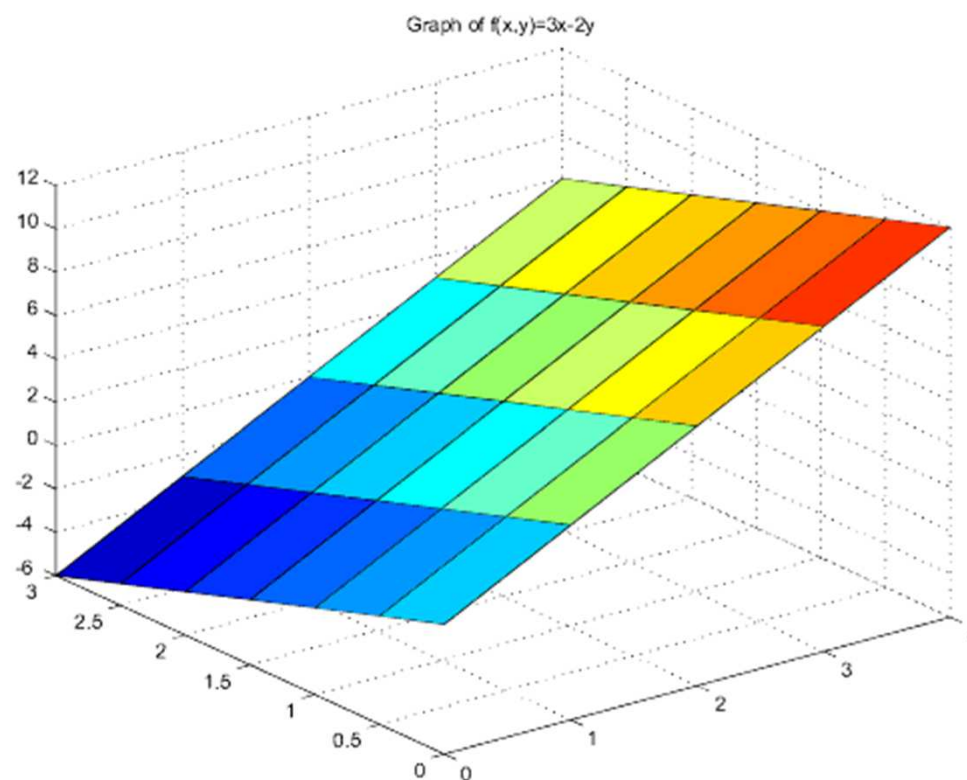
$$f(x,y)=3x - 2y.$$

```
>> Z=3*X-2*Y
```

```
>> surf(X,Y,Z)
```

```
>> title('Graph of f(x,y)=3x-2y')
```

# Vẽ mặt



# Vẽ mặt

- Ví dụ: Vẽ đồ thị hàm số

$f(x,y)=x^2y - 2y$  trên miền  $[-2,2] \times [-1,1]$ .

```
>> [X,Y]=meshgrid(-2:.1:2,-1:.1:1);
```

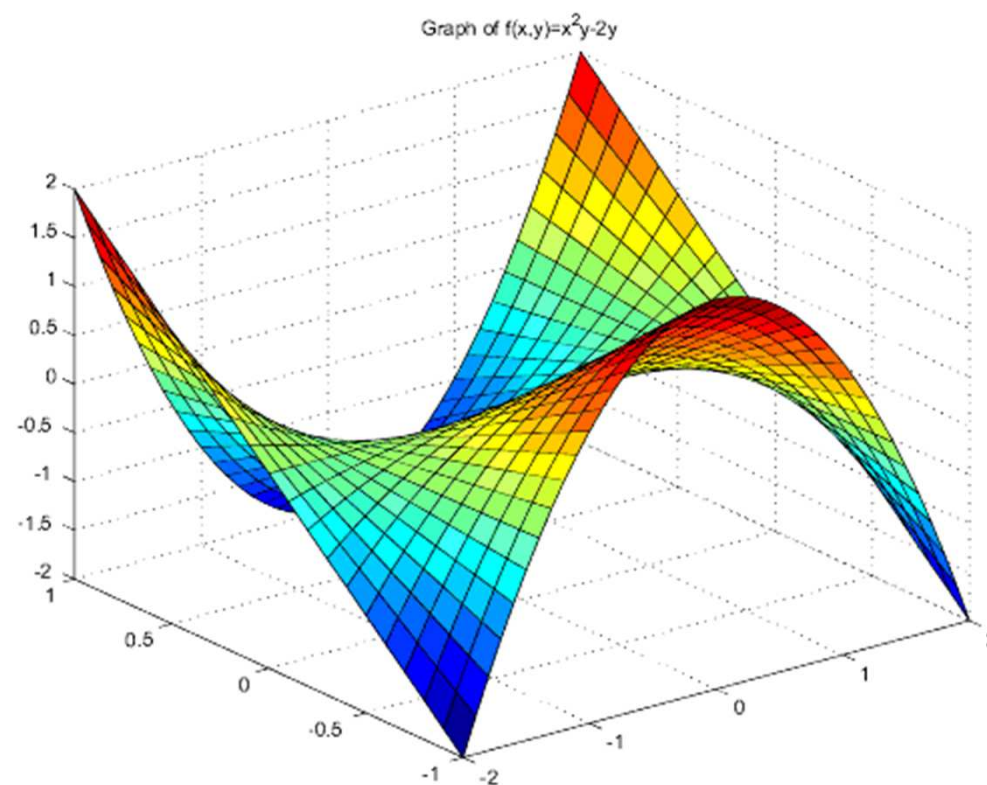
```
>> Z=(X.^2).*Y-2*Y;
```

```
>> surf(X,Y,Z)
```

```
>> title('Graph of f(x,y)=x^2y-2y')
```



# Vẽ mặt



# Đa thức trong Matlab

- Đa thức bậc  $n$

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

- Biểu diễn bằng vector hàng  $n+1$  phần tử
  - Sắp xếp hệ số từ bậc cao nhất đến bậc 0
  - Ví dụ  $p = 3x^4 + x^2 - 3x + 5$ 
    - `>>p=[3 0 1 -3 5]`
    - `p= 3 0 1 -3 5`

# Các phép tính với đa thức

HÀM	Ý NGHĨA
<code>conv(p1,p2)</code>	Nhân hai đa thức
<code>[k,d]=deconv (p1,p2)</code>	Chia hai đa thức ( k= kết quả; d =phần dư)
<code>k=polyder(p)</code>	Tìm đạo hàm của đa thức p
<code>k=polyder(p,q)</code>	Tìm đạo hàm của đa thức tích ( $p \cdot q$ )
<code>[n,d]=polyder(num,den)</code>	Tìm đạo hàm (dạng $n/d$ ) của phân thức ( $num/den$ )
<code>roots(p)</code>	Tìm nghiệm đa thức p
<code>p=poly(r)</code>	Lập đa thức p từ vector r chứa các nghiệm.
<code>polyval(p,x)</code>	Tính giá trị của đa thức tại x (x có thể là mảng)
<code>[r,p,k]= residue(num,den)</code>	Tìm các thành phần tối giản của phân thức
<code>[num,den]=residue(r,p,k)</code>	Chuyển các thành phần tối giản thành 1 phân thức
<code>printsys(num,den,'s')</code>	in phân thức có dạng tỉ số 2 đa thức theo s
<code>[z,p,k]=tf2zp(num,den)</code>	Tìm các zero z, cực p, độ lợi k của phân thức

# Các phép tính với đa thức

- **Ví dụ 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
```

- **Ví dụ 2:**

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

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

$$k(x)=x^2 +4$$

$$r=[1 \ 2]$$

$$p=[-2 \ 2]$$

# Các phép tính với đa thức

**Ví dụ 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
```

Do đó :

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

# Vào-Ra dữ liệu

- **Vào dữ liệu từ bàn phím**
  - Các lệnh liên quan đến nhập dữ liệu từ bàn phím: input, keyboard, menu và pause.
- **Lệnh input:** Lệnh này đưa ra thông báo nhắc người sử dụng nhập dữ liệu và nhận dữ liệu đánh từ bàn phím. Lệnh có dạng

**R = input(*string*)**

trong đó R là tên biến, *string* là xâu ký tự chứa thông báo nhắc người sử dụng biết về dữ liệu cần nạp. Dữ liệu có thể là biểu thức bất kỳ. Nếu người sử dụng ấn Enter ngay thì R sẽ là ma trận rỗng.

# Vào dữ liệu từ bàn phím

- Ví dụ:

```
>> m=input('Nhap so dong cua ma tran:
      ')
```

Nhap so dong cua ma tran: 3

m =

3

```
>> n = input('Hay nhap so cot cua ma
      tran n =')
```

Hay nhap so cot cua ma tran n =5

n =

5

# Vào dữ liệu từ bàn phím

- Ví dụ

```
>> a =input('Hay nhap cac phan tu cua ma tran a  
theo khuan dang:\n [Cacphan tu dong 1 ;\n    ...  
\n Cac phan tu dong m] : \n\n')
```

Hay nhap cac phan tu cua ma tran a theo khuan dang:

[Cacphan tu dong 1 ;

...

Cac phan tu dong m] :

[1 2 3 4 5;

6 7 8 9 10;

11 12 13 14 15]



# Vào-Ra dữ liệu

- **Lệnh keyboard**

- Khi lệnh này được đặt trong m – file thì nó ngắt việc thực hiện chương trình.
- Màn hình xuất hiện dấu nhắc >>K.
- Người sử dụng có thể xem hoặc biến đổi giá trị các biến bằng các lệnh của Matlab.
- Chấm dứt chế độ dùng **Enter**.
- Tiện lợi cho việc gỡ rối (debug) chương trình.

# Vào-Ra dữ liệu

- **Lệnh menu:** Cho phép tạo bảng lựa chọn. Lệnh có dạng sau.

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

- Lệnh sẽ hiển thị tiêu đề của bảng chọn chứa trong chuỗi HEADER, tiếp đến là dãy các lựa chọn trong các chuỗi: ITEM1, ..., ITEMn.
- Chỉ số của lựa chọn sẽ được trả lại cho biến CHOICE.
- Trong màn hình đồ họa Matlab sẽ hiển thị bảng lựa chọn trong một cửa sổ MENU với các phím ấn chọn.

# Vào-Ra dữ liệu

- Ví dụ:

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

- Trên màn hình đồ họa hiển thị bảng lựa chọn:



- Người sử dụng chọn bằng việc click chuột vào lựa chọn cần thiết.

# Vào-Ra dữ liệu

- Cách sử dụng khác của menu

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

trong đó ITEMLIST là mảng chuỗi.

- Ví dụ:

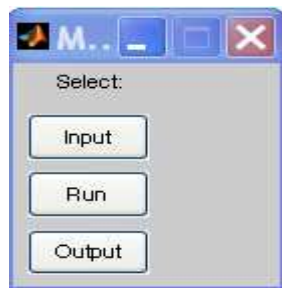
```
>> ItemList={'Input','Run','Output'};
```

```
>> HEADER='Select:';
```

```
>> CHOICE = MENU(HEADER, ItemList)
```

```
CHOICE =
```

```
1
```



# Đưa ra màn hình

- **Đưa ra màn hình nội dung mảng**
  - Gõ tên biến không có ; cuối lệnh

- **Ví dụ:**

```
>> x = [1 2 3] ;
```

```
>> y=[2; 3; 4] ;
```

```
>> x*y
```

```
ans =
```

```
20
```

# Đưa ra màn hình

- **Lệnh disp:**

- Dùng để hiển thị nội dung của biến ra màn hình và không hiển thị tên biến
- Làm việc dù có; hay không
- Lệnh có dạng

**disp(X)**

với X là biến hoặc biểu thức cần hiển thị.

- **Ví dụ:**

```
>> 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
```

# Đưa ra màn hình

- **Ví dụ:** Lập bảng tính giá trị hàm sin

```
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))
```

# Đưa ra màn hình

- Kết quả thực hiện có thể có dạng:

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



# Đưa ra màn hình

- **Lệnh fprintf**
- Lệnh có dạng

**fprintf(dialog\_format, danh\_sach\_bien)**

trong đó **dialog\_format** là biến (hằng) chuỗi ký tự thông báo và các ký tự định khuôn dạng dữ liệu ra.

# Đưa ra màn hình

- Ví dụ:

```
>> 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
```

# Vào ra với file văn bản

- Matlab cung cấp các lệnh cho phép làm việc với các file
- Ở đây chỉ hạn chế trình bày các thao tác với file văn bản
- Các lệnh chính liên quan đến làm việc với file văn bản của Matlab là:
  - fopen, fclose
  - frewind, fread, fwrite
  - fscanf, fprintf..

# Vào ra với file văn bản

- **Lệnh fopen.** Dùng để mở file

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

- Lệnh này mở file văn bản có tên cho bởi FILENAME (hằng xâu ký tự hoặc biến xâu)
- Chế độ làm việc được chỉ ra bởi PERMISSION.
- PERMISSION có thể là:
  - 'rt' mở file để đọc
  - 'wt' mở file để ghi (nếu chưa có file sẽ tạo ra file mới có tên FILENAME)
  - 'at' mở file nối đuôi (tạo file mới có tên FILENAME nếu chưa có)
  - 'rt+' mở file để đọc và ghi (không tạo file mới)
- Biến nhận dạng *fileID* nhận giá trị nguyên và ta sẽ sử dụng nó để thâm nhập vào file.

# Vào ra với file văn bản

- Trong trường hợp muốn kiểm tra lỗi mở file có thể sử dụng lệnh

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

nếu gặp lỗi mở file biến MESSAGE sẽ chứa thông báo lỗi.

# Vào ra với file văn bản

- Ví dụ:

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

# Vào ra với file văn bản

- **Lệnh đóng file fclose:** Đóng file làm việc.

**ST = fclose(FID)**

- Lệnh này sẽ đóng file ứng với biến nhận dạng FID
- fclose trả lại biến ST giá trị 0 nếu nó hoàn thành việc đóng file và -1 nếu gặp lỗi.
- Lệnh ST=fclose('all') đóng tất cả các file đang mở ngoại trừ 0, 1, 2.

- **Lệnh frewind.**

**frewind(FID)**

đặt con trỏ file của FID vào đầu file.

# Vào ra với file văn bản

- **Lệnh fscanf:** Đọc dữ liệu vào từ file

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

- Lệnh này đọc dữ liệu từ file tương ứng với FID
- Chuyển đổi dữ liệu về khuôn dạng được xác định bởi biến xâu FORMAT
- Gán giá trị cho mảng A
- COUNT là biến ra tùy chọn dùng để chứa số lượng phần tử đọc được.
- SIZE là biến tùy chọn chỉ giới hạn số phần tử được đọc vào từ file, nếu vắng mặt thì toàn bộ file được xét. Các giá trị có thể có của biến là:
  - N : đọc không quá N phần tử từ file vào vector cột
  - Inf : đọc không quá kết thúc file
  - [M, N] : đọc không quá M\*N phần tử và đưa vào ma trận kích thước không quá MxN theo từng cột, N có thể là inf nhưng M thì phải là hữu hạn.



# Vào ra với file văn bản

- Nếu ma trận A là kết quả của việc chuyển khuôn dạng ký tự và biến SIZE không có dạng [M, N] thì vector dòng sẽ được trả lại.
- FORMAT là biến chứa các ký tự chuyển đổi khuôn dạng của ngôn ngữ C.
- Các ký tự định khuôn dạng bao gồm: %, các ký hiệu thay thế, độ dài trường, các ký tự chuyển đổi khuôn dạng: d, i, o, u, x, e, f, g, s, c và [...] (liệt kê tập hợp).
- Nếu %s được sử dụng thì khi đọc một phần tử có thể dẫn đến phải sử dụng một loạt các thành phần của ma trận, mỗi thành phần giữ một ký tự.
- Hãy sử dụng %c để đọc ký tự trắng (space) và khuôn dạng %s bỏ qua các ký tự trắng.

# Vào ra với file văn bản

- Nếu chỉ thị định dạng gồm cả số lẫn ký tự thì ma trận kết quả sẽ là ma trận số và mỗi ký tự sẽ được chuyển thành số bằng giá trị mã ASCII của nó.
- fscanf khác với lệnh này trong C ở chỗ nó là lệnh vector hóa trả lại đối số là ma trận.
- Biến xâu định dạng sẽ được sử dụng lặp lại cho đến khi gặp kết thúc file hoặc đọc đủ số lượng phần tử chỉ ra bởi SIZE.

# Vào ra với file văn bản

- Ví dụ:

- Lệnh

$S = \text{fscanf}(\text{fid}, '%s')$

đọc (và trả lại) một xâu.

- Lệnh

$A = \text{fscanf}(\text{fid}, '%5d')$

đọc các số nguyên có 5 chữ số thập phân.

# Vào ra với file văn bản

- **Ví dụ:** Giả sử có file văn bản với tên “kq” chứa xâu “Day la ket qua dua ra”. Khi đó ta có thể đọc dữ liệu vào như sau:

```
>> fid=fopen('kq','rt')
fid =
     3
>> x = fscanf(fid,'%s')
x =
Daylaketquaduara
>> frewind(fid)
>> x = fscanf(fid,'%s%c')
x =
Day la ket qua dua ra
```

# Vào ra với file văn bản

- **Ví dụ:** Giả sử file văn bản Matrix.txt chứa hai dòng

1 2 3 4 5  
6 7 8 9 10

- Hãy theo dõi kết quả làm việc của các lệnh sau để thấy tác động của lệnh fscanf

```
>> fopen('matrix.txt','rt')
```

```
ans =
```

```
4
```

```
>> A=fscanf(4,'%i',[2,5])
```

```
A =
```

1	3	5	7	9
2	4	6	8	10

# Vào ra với file văn bản

```
>> 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
```

# Vào ra với file văn bản

- **Lệnh fprintf** : ghi dữ liệu theo khuôn dạng ra file.

**COUNT = FPRINTF(FID, FORMAT, A, . . .)**

- Lệnh này sẽ định dạng các thành phần của ma trận A (và các biến tiếp theo trong danh sách) theo khuôn dạng được xác định bởi biến chuỗi format, và ghi ra file tương ứng với biến nhận dạng FID.
- Biến:
  - COUNT đếm số byte dữ liệu được ghi ra.
  - FID là số nguyên nhận dạng tên file thu được từ lệnh fopen. Có thể dùng số 1 nếu sử dụng thiết bị ra chuẩn (màn hình). Nếu lệnh này không có FID thì mặc định đưa kết quả ra màn hình.
  - FORMAT là biến chuỗi chứa các ký tự mô tả định dạng dữ liệu giống như trong ngôn ngữ C
  - Các ký tự \n, \r, \t, \b, \f có thể dùng để tạo linefeed, carriage return, tab, backspace, và formfeed character tương ứng.
  - Sử dụng \\ để tạo dấu \ và %% để tạo ký tự %.

# Vào ra với file văn bản

- Ví dụ:

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

Tạo ra file văn bản có tên 'exp.txt' chứa bảng giá trị của hàm mũ.

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