

THỰC HÀNH LABORATORY

Trường Đại Học Khoa Học Tự Nhiên, Tp HCM

Giao diện của Matlab

Các phép toán cơ bản

Sử dụng file .m và lập trình

Ma trận trong Matlab

Đồ thị 2D trong Matlab

Phép tính với biến symbolic

Ứng dụng MATLAB

We study the existence and unicity of invariant measure on \mathbb{R} of the Markov chain $(X_n)_{n \geq 0}$ defined by $X_0 = x$ and, for all $n \geq 1$

$$X_n := a_n X_{n-1} + b_n$$

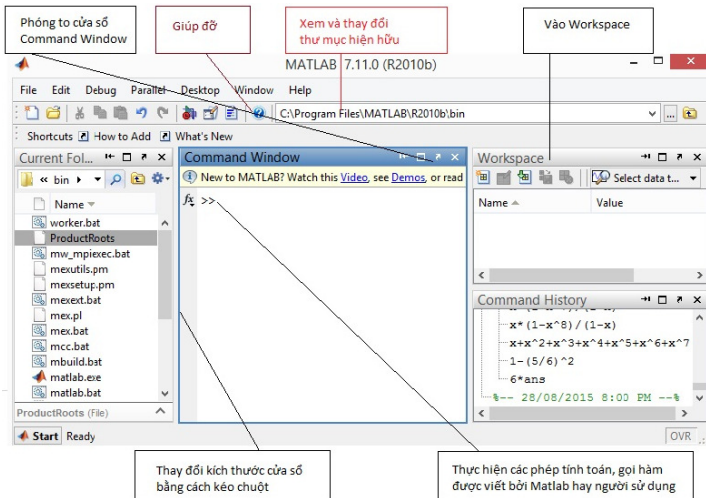
where the (a_n, b_n) , $n \geq 1$, are i.i.d. random variables with values in $\mathbb{R}^{*+} \times \mathbb{R}$. Both cases $\mathbb{E}(\log a_n) < 0$ and $\mathbb{E} \log a_n = 0$ will be considered; they lead to strongly different behaviours.

This Markov chain is a fundamental example of stochastic dynamical system and appears in several classical situations, as perpetuities, speculatives prices, and random walks on groups of matrices.

Giao diện của Matlab

1. Giao diện của Matlab

Khi chạy chương trình của Matlab, giao diện đầu tiên của chương trình xuất hiện với các thành phần cơ bản.



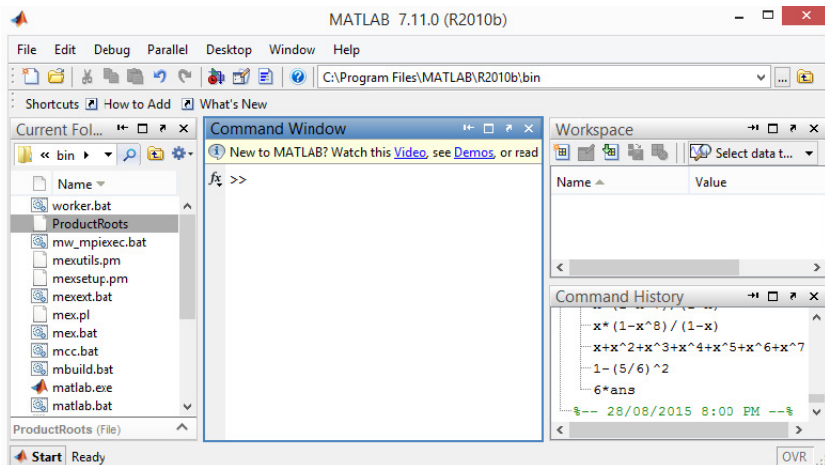
1. Giao diện của Matlab

a. Cửa sổ lệnh - command windows

Cửa sổ lệnh của Matlab cho phép người sử dụng các phép tính toán, gọi các lệnh, hàm hoặc gọi các chương trình. Trong quá trình sử dụng Matlab, ta có thể sử dụng một số hàm trợ giúp sau

- ▶ **help**: giúp đỡ, cho phép người dùng tra cứu bất cứ thông tin nào liên quan đến Matlab.
- ▶ **demo**: các file, các chương trình đã được Matlab tạo sẵn, được tập hợp theo các chuyên đề.
- ▶ **edit**: Mở chương trình soạn thảo hàm, chương trình.
- ▶ **ver**: Xem thông tin về phiên bản của Matlab và các thành phần của nó.
- ▶ **exit**: Thoát ra khỏi chương trình.

1. Giao diện của Matlab



1. Giao diện của Matlab

b. Cửa sổ không gian làm việc - work space

Cửa sổ không gian làm việc liệt kê tất cả các biến hiện đang sử dụng trong chương trình.

c. Cửa sổ thư mục hiện tại - current directory

Hiển thị thư mục hiện tại mà chương trình Matlab đang dẫn đến. Thư mục mặc định là

C:\Programs\MATLAB\R2010b\work

Chương trình cho phép thiết lập đường dẫn đến thư mục bất kì trên máy tính.

d. Cửa sổ lịch sử lệnh - command history

Ghi nhớ các lệnh đã thực hiện trên cửa sổ lệnh, có thể copy và dán lại các mệnh lệnh đã thực thi được lưu lại vào ngược lại cửa sổ lệnh.

Các phép toán cơ bản

2. Các phép toán cơ bản

Các phép toán cơ bản của Matlab được thực hiện trực tiếp trên cửa sổ lệnh command windows. Các phép toán cơ bản bao gồm:

Các phép toán số học, các phép toán lượng giác, các phép toán làm tròn, các phép toán so sánh, các phép toán về số phức.

2.1 Các phép toán số học

Để tính toán với các phép tính số học đơn giản, tại ngay đầu nhắc >> của cửa sổ lệnh Command Windows, chúng ta gõ vào trực tiếp:

2.1 Các phép toán số học

STT	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
1	+	Cộng	2+5	7
2	-	Trừ	1000-25	975
3	*	Nhân	10*10	100
4	/	Chia	100/5	20
5	^	Lũy thừa a^b	10^3	1000
6	<code>sqrt(x)</code>	Căn bậc 2	<code>sqrt(144)</code>	12
7	<code>exp(x)</code>	Hàm mũ (e^x)	<code>exp(1)</code>	2.7183
8	<code>log(x)</code>	Logarit tự nhiên ($\ln(x)$)	<code>log(exp(1))</code>	1
9	<code>log10(x)</code>	Logarit thập phân ($\log_{10}(x)$)	<code>log10(100)</code>	2

2.2 Các phép toán lượng giác

Khi sử dụng các phép toán lượng giác, chúng ta chú ý là Matlab hiểu các đối số của các hàm lượng giác là radian. Cũng như kết quả trả về của các hàm lượng giác ngược cũng là radian.

STT	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
1	sin	Sin	$\sin(30 \cdot \pi / 180)$	0.5000
2	cos	Cos	$\cos(0.5)$	0.8776
3	tan	Tang	$\tan(10 \cdot \pi / 180)$	0.1763
4	cot	Cotang	$\cot(45 \cdot \pi / 180)$	1
5	asin	arcsin	$\text{asin}(0.5) \cdot 180 / \pi$	30
6	acos	arccos	$\text{acos}(0.86) \cdot 180 / \pi$	30
7	atan	arctang	$\text{atan}(1) \cdot 180 / \pi$	45
8	acot	arccotang	$\text{acot}(1) \cdot 180 / \pi$	45

2.3 Các phép toán làm tròn và lấy phần dư

STT	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
1	fix	Làm tròn các thành phần thập phân về 0	fix(1.5680)	1
2	floor	Làm tròn về số nguyên gần nhất nhỏ hơn	floor(1.5680)	1
3	ceil	Làm tròn về số nguyên gần nhất lớn hơn	ceil(1.5680)	2
4	round	Làm tròn về số nguyên gần nhất	round(1.5680)	2
5	mod(x,y)	Tính phần dư phép chia, lấy theo y	mod(13,5)	3
6	rem(x,y)	Tính phần dư phép chia, lấy theo x	rem(13,2)	1
7	sign(x)	Lấy dấu của x	sign(-2)	-1

2.4 Các phép toán so sánh

Các phép toán so sánh sẽ so sánh giá trị của giá trị bên phải và bên trái của hàm so sánh, tùy theo từng trường hợp cụ thể mà giá trị trả về có thể là 1 hay 0.

STT	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
1	>	So sánh lớn hơn	$1 > 2$	0
2	<	So sánh nhỏ hơn	$1 < 2$	1
3	==	So sánh bằng	$1 == 2$	0
4	~=	So sánh không bằng	$1 ~= 2$	1
5	>=	So sánh lớn hơn hay bằng	$1 >= 2$	0
6	<=	So sánh nhỏ hơn hay bằng	$1 <= 2$	1

2.5 Các phép toán logic

Các phép toán logic sẽ so sánh giá trị của giá trị bên phải và bên trái của các hàm so sánh, tùy theo từng trường hợp cụ thể mà giá trị trả về có thể là 1 hay 0.

STT	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
1	&	Phép giao	$(1 > 2) \& (2 > 4)$	0
2	and	Phép giao	$\text{and}(1 > 3, 2 > 4)$	0
3		Phép hợp	$(1 > 3) (2 > 1)$	1
4	or	Phép hợp	$\text{or}(1 > 3, 2 > 1)$	1
5	~	Phép phủ định	$\sim(1 > 2)$	1
6	not	Phép phủ định	$\text{not}(1 > 2)$	1
7	xor	Phép Xor	$\text{xor}(1 < 3, 2 < 5)$	0

2.6 Phép toán gán

Toán tử gán được sử dụng trong Matlab dùng để thay thế cho giá trị hoặc một biểu thức tính toán.

Toán tử = trong Matlab được gọi là toán tử gán

Ví dụ: Cần tính biểu thức

$$A = \frac{51^6 + 76^7}{\ln(1076)e^{15}}$$

Khi đó, chúng ta sẽ thay giá trị tử số bằng biến B, giá trị mẫu số bằng biến C và giá trị A cần tính bằng B/C.

Cú pháp của toán tử gán:

Tên biến = giá trị hay biểu thức tính toán

Tên biến: Tối đa 31 kí tự, có phân biệt chữ hoa và chữ thường, có thể sử dụng các chữ cổ trong tên biến nhưng kí tự đầu tiên của tên biến phải là chữ.

2.6 Phép toán gán

Thực hiện phép gán giải quyết ví dụ trên như sau:

```
>> B = 51^6+76^7  
>> C = log(1076)*exp(15)  
>> A = B/C
```

Ví dụ:

- ▶ Phép gán: $x+2 = 20$ là sai vì bên trái toán tử gán không phải là một tên biến.
- ▶ Phép gán: $x = 5+y$ chỉ đúng nếu giá trị của biến y đã được xác định từ trước, nếu không Matlab sẽ báo sai.

```
>> x = 5;  
>> x = x+3
```

2.7 Dạng hiển thị số

```
>> b = 3/26;  
>> format long; b  
b =  
    0.115384615384615  
>> format short e; b  
b =  
    1.1538e-001  
>> format bank; b  
b =  
    0.12  
>> format short eng; b  
b =  
    115.3846e-003  
>> format hex; b  
b =  
    3fbd89d89d89d89e
```

```
>> format +; b  
b =  
    +  
>> format rat; b  
b =  
    3/26  
>> format short; b  
b =  
    0.1154  
>> format long eng; b  
b =  
    115.384615384615e-003
```

2.8 Các lệnh người dùng

- ▶ **clc**: Xóa (lau) cửa sổ lệnh. Mang tính chất hình thức, giá trị các biến vẫn tồn tại.
- ▶ **clear**: Giải phóng bộ nhớ biến ra khỏi bộ nhớ. Xóa workspace.
- ▶ **clear var1 var2**: Giải phóng các biến var1 và var2 ra khỏi bộ nhớ.
- ▶ **exist('name')**: Hỏi Matlab xem có tồn tại tập tin hay biến đã được thành lập có tên là name chưa.
- ▶ **quit**: Thoát khỏi khung chương trình Matlab.
- ▶ **who**: Liệt kê các biến hiện hành có trong bộ nhớ.
- ▶ **whos**: Liệt kê các biến hiện hành và kích thước của chúng trong bộ nhớ và chỉ rõ phần ảo của chúng nếu có.

2.8 Các lệnh người dùng

- ▶ **; dấu chấm phẩy**: Ở cuối dòng lệnh ngăn không cho Matlab hiển thị các kết quả ra cửa sổ lệnh.
- ▶ **... dấu ba chấm**: liên tục, khi dòng lệnh quá dài cần xuống dòng, dấu ba chấm ... ở cuối dòng lệnh báo cho Matlab biết còn tiếp tục ở dòng tiếp theo.
- ▶ **, dấu phẩy**: Ngăn cách các phần tử trong mảng.
- ▶ **: dấu hai chấm**: Được dùng để phát sinh một mảng có các phần tử cách đều nhau.
- ▶ **% dấu phân trăm**: Matlab xem như những gì sau dấu % là lời bình, lời chú thích cho lệnh. Thường dùng khi viết chương trình.

Sử dụng file .m và lập trình

3. Sử dụng file .m và lập trình

Ở chương 2, chúng ta đã sử dụng cửa sổ lệnh để thực hiện các phép tính toán bằng cách tính toán trực tiếp trên cửa sổ lệnh, nhưng để giải quyết các vấn đề phức tạp thì ta phải sử dụng các file được lập trình.

Các file được lập trình của Matlab được gọi là các M-file (các file này có đuôi là .m), do đó trong toàn bộ giáo trình các file được lập trình sẽ gọi là M-file.

3.1 Cách tạo M-file

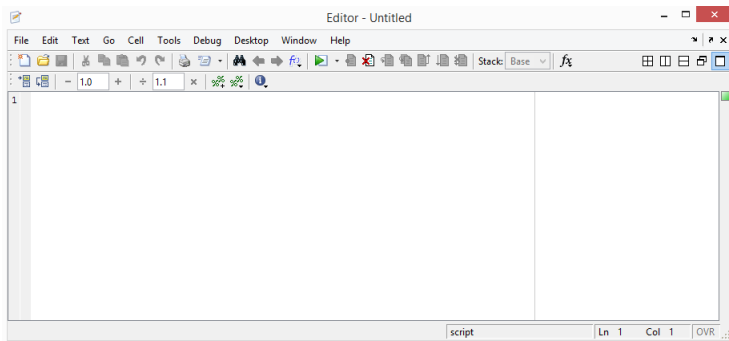
Có 2 cách để khởi động một chương trình biên soạn M-file.

1: Từ cửa sổ lệnh (command window), gõ edit.

2: Vào menu File, chọn New.

Khi đó, chương trình sẽ hiển thị một cửa sổ trắng để chúng ta soạn thảo.

3.1 Cách tạo M-file



Sau khi soạn thảo xong M-file, chúng ta nhấn: **Ctrl + S** hoặc **File/Save** để lưu file chương trình. Khi đặt tên file chương trình phải đúng theo quy định của Matlab. Cụ thể, tên file phải được bắt đầu bằng chữ, sau đó có thể sử dụng số, và được dùng dấu gạch ngang dưới để phân biệt, ví dụ tên file: baitap_21.m.

3.1 Cách tạo M-file

Để chạy chương trình, có thể sử dụng một trong 2 cách sau:

- 1: Trong môi trường soạn thảo M-file, chúng ta vào menu Debug/Run hoặc nhấn phím tắt F5.
- 2: Trong cửa sổ lệnh command window, chúng ta **nhập vào đúng tên M-file đã được lưu**, sau đó nhấn Enter..

3.1 Cách tạo M-file

Nếu chương trình **được lập trình đúng**, sau khi chạy chương trình, người sử dụng **chuyển ra cửa sổ lệnh để xem kết quả**.

Còn ngược lại, chương trình sẽ báo lỗi, Matlab phát ra **1 tiếng bip báo hiệu**, đồng thời chương trình sẽ tự **chuyển sang cửa sổ lệnh, thông báo cho người lập trình vị trí bị lỗi**.

3.2 Các hàm nhập và hàm xuất dữ liệu ra màn hình

a. Hàm nhập dữ liệu

Cú pháp:

```
x = input('prompt')
```

trong đó:

- ▶ **input:** Từ khóa của hàm nhập dữ liệu.
- ▶ **x:** Tên biến được gán giá trị nhập vào.
- ▶ **prompt:** Dòng text mà người sử dụng gõ vào.

Diễn đạt: Biến **x** sẽ có giá trị bằng giá trị mà người sử dụng nhập vào.

3.2 Các hàm nhập và hàm xuất dữ liệu ra màn hình

b. Hàm xuất dữ liệu ra màn hình

Cú pháp:

```
disp(x)  
disp('text')
```

trong đó:

- ▶ `disp`: Từ khóa của hàm xuất dữ liệu.
- ▶ `x`: Tên biến hoặc các giá trị số cần xuất ra màn hình.
- ▶ `text`: Dòng text mà người sử dụng cần xuất ra màn hình.

Ghi chú: Cú pháp `clear` hoặc `clear all` sẽ xóa toàn bộ các biến đang được sử dụng trong Matlab, giúp chương trình chạy đúng.

3.2 Các hàm nhập và hàm xuất dữ liệu ra màn hình

```
% chương trình tính diện tích hình chữ nhật
disp('tính diện tích hình chữ nhật');
a = input('nhập a = ');
b = input('nhập b = ');
S = a*b;
disp('diện tích hình chữ nhật');
disp(S)

% chương trình giải phương trình bậc hai
disp('CHƯƠNG TRÌNH GIẢI PHƯƠNG TRÌNH BẬC HAI');
disp('nhập vào các hệ số:');
a = input('nhập hệ số a = ');
b = input('nhập hệ số b = ');
c = input('nhập hệ số c = ');
delta = b^2-4*a*c;
disp('các nghiệm số: ');
x1 = (-b+sqrt(delta))/(2*a)
x2 = (-b-sqrt(delta))/(2*a)
```

3.3 Các hàm con

a. Hàm điều kiện

`if - elseif - else - end`

Dạng đơn giản:

`if - end (if - else - end)`

Cú pháp:

```
if expression
    statements
end
```

trong đó

- ▶ `if`, `end`: Từ khóa của hàm.
- ▶ `expression`: Điều kiện - biểu thức logic.
- ▶ `statements`: Lệnh hoặc nhóm lệnh cần thực thi.

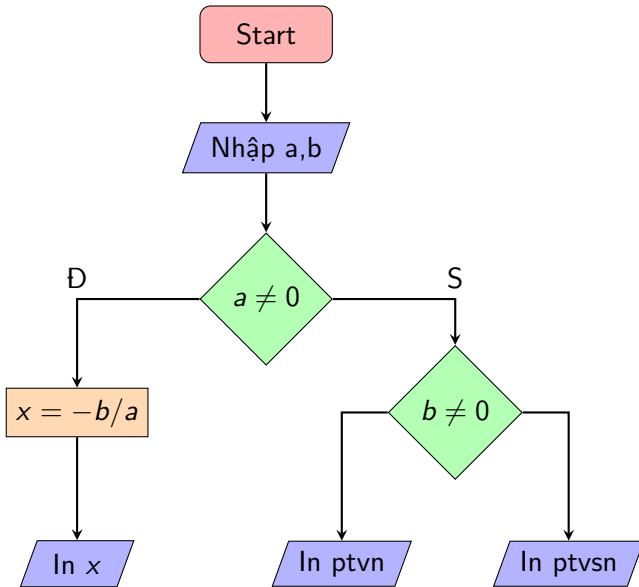
Diễn đạt: Nếu thỏa điều kiện - biểu thức logic thì sẽ thực hiện lệnh.

a. Hàm điều kiện

Ví dụ

```
a = input('nhap a = ');  
b = input('nhap b = ');  
  
if (a < b)  
    disp('a nho hon b');  
end  
  
if (a == b)  
    disp('a bang b');  
end  
  
if (a > b)  
    disp('a lon hon b');  
end
```

Viết chương trình giải phương trình bậc nhất $ax + b = 0$



Viết chương trình giải phương trình bậc nhất $ax + b = 0$

```
% Giai phuong trinh bac nhat
a = input('nhap a = ');
b = input('nhap b = ');
if (a ~= 0)
    x = -b/a;
    fprintf('nghiem so x = %9.5g\n',x);
else
    if (b ~= 0)
        disp('ptvn')
    else
        disp('ptvsn')
    end
end
```

```
% chương trình giải phương trình bậc hai
disp('giải phương trình bậc hai');
a = input('nhập hệ số a = ');
b = input('nhập hệ số b = ');
c = input('nhập hệ số c = ');
delta = b^2-4*a*c;
d = delta;
if (d < 0)
    disp('ptvn')
else
    if (d == 0)
        x = -b/(2*a);
        fprintf('ngheiem so x = %9.5g\n',x);
    else
        x1 = (-b-sqrt(d))/(2*a);
        x2 = (-b+sqrt(d))/(2*a);
        fprintf('ngheiem so x1 = %9.5g\n',x1);
        fprintf('ngheiem so x2 = %9.5g\n',x2);
    end
end
end
```

Cú pháp:

Dạng đầy đủ:

if - elseif - else - end

```
if expression 1
    statements 1
elseif expression 2
    statements 2
else
    statements 3
end
```

- ▶ if, elseif, else, end: Từ khóa của hàm
- ▶ expression 1, 2, 3: Điều kiện - biểu thức logic 1, 2, 3.
- ▶ statements 1, 2, 3: Lệnh hoặc nhóm lệnh thực thi 1, 2, 3.

Diễn đạt:

- Nếu thỏa điều kiện hoặc biểu thức logic 1 thì sẽ thực hiện lệnh 1.
- Nếu không thỏa điều kiện 1 và thỏa điều kiện 2 thì sẽ thực hiện lệnh 2.
- Nếu không thỏa cả điều kiện 1 và điều kiện 2 thì sẽ thực hiện lệnh 3.

Chú ý: Các điều kiện 1 và 2 không được trùng lặp nhau.

a. Hàm điều kiện

Ví dụ. Bài toán phân loại học sinh: Điểm 9 – 10 xếp loại giỏi, điểm 7 – 8 xếp loại khá, điểm 5 – 6 xếp loại trung bình, điểm 1, 2, 3, 4 xếp loại yếu. Nếu điểm vào không phải số nguyên nằm giữa 1 và 10 thì thông báo điểm không hợp lệ.

```
diem = input('nhap diem = :')
if (diem == 9)|(diem == 10)
    disp('loai gioi')
elseif (diem == 7)|(diem == 8)
    disp('loai kha')
elseif (diem == 5)|(diem == 6)
    disp('loai trung binh')
elseif (diem >= 1)|(diem <= 4)
    disp('loai yeu')
else
    disp('diem vao khong hop le')
end
```

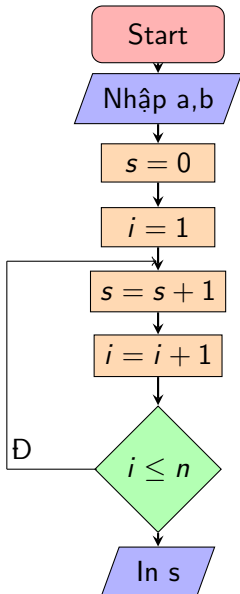
a. Hàm điều kiện

Quy tắc xử lý ưu tiên:

Trong một biểu thức Matlab vừa có toán tử số học, vừa có toán tử quan hệ và toán tử logic thì thứ tự xử lý trong Matlab như sau:

1. Các cặp dấu ngoặc được tính từ cặp trong cùng nhất.
2. Các toán tử số học và toán tử **NOT** (\sim) được tính từ trái qua phải.
3. Các toán tử quan hệ được tính từ trái qua phải.
4. Toán tử **AND**.
5. Toán tử **OR**.

b. Hàm vòng lặp for



Ví dụ cho vòng lặp for

$$S = 1 + 2 + 3 + \dots + n$$

$$S = 0 + 1 + 2 + \dots + n$$

```
clc
clear all
disp('tinh tong');
n = input('nhap so hang can tinh tong n = ');
s = 0; % gia tri ban dau cua tong s
for i = 1:n
    s = s+i;
end
fprintf('tong so s = %2.5g\n',s);
```

c. Hàm vòng lặp while

Được sử dụng khi số lần lặp không được biết trước. Quá trình lặp sẽ chấm dứt khi một điều kiện xác định nào đó được thỏa. Cấu trúc của vòng lặp while như sau:

Cú pháp:

```
while expression
    statements
end
```

trong đó:

- ▶ expression: Điều kiện của vòng lặp.
- ▶ statements: Lệnh hoặc nhóm lệnh thực thi của vòng lặp.

Diễn đạt:

Nếu thỏa điều kiện thì sẽ thực hiện lệnh của vòng lặp.

Hàm `while` được sử dụng khi chưa biết số lần lặp, trong khi hàm `for` được sử dụng khi đã biết rõ số lần lặp.

c. Hàm vòng lặp while

Ví dụ:

Tính $n! = 1.2.3 \dots n$.

GT = 1.1.2.3...n

```
clc
clear
disp('tinh giai thua');
n = input('nhap so hang can tinh giai thua n = ');
GT = 1;
i = 1;
while (i <= n)
    GT = GT*i;
    i = i+1;
end
fprintf('tong so s = %2.5g\n',GT);
```

d. Hàm vòng lặp switch-case

Cấu trúc switch-case cho phép chương trình có nhiều lựa chọn và thực hiện chỉ một trong những nhánh này, tùy thuộc vào giá trị của biểu thức đầu vào. Cấu trúc switch-case có dạng như sau:

Cú pháp:

switch	biểu thức đầu vào (vô hướng hoặc chuỗi kí tự)	
case	giá trị 1	nhóm lệnh 1
case	giá trị 2	nhóm lệnh 2
...		
case	giá trị n	nhóm lệnh n
otherwise		nhóm lệnh $n + 1$
end		

d. Hàm vòng lặp switch-case

Ví dụ: Trở lại bài toán phân loại học sinh, chương trình được viết với cấu trúc switch-case như sau

```
n = input('cho biet diem: ');
disp('phan loai: ');
switch n
    case {0,1,2,3,4}
        disp('loai yeu');
    case {5,6}
        disp('loai trung binh');
    case {7,8}
        disp('loai kha');
    case {9,10}
        disp('loai gioi');
    otherwise
        disp('khong hop le');
end
```

3.4 Hàm trong Matlab (function)

Hàm trong Matlab là một file.m có thể nhận tham số và trả về các giá trị.

Tên hàm phải trùng với tên file.m. Gọi lệnh bằng cách gõ tên hàm (tên file.m).

Cú pháp:

```
function tri_tra_ve = ten_ham(tham_so)  
...
```

Ví dụ:

```
% tbc.m  
function s = tbc(x)  
s = sum(x(:))/(length(x(:)));
```

Gọi lệnh:

```
>> t = tbc([2 3 4 5 6])
```

t sẽ bằng 4.

3.4 Hàm trong Matlab (function)

Script:

```
% UBC.m  
function [m,n] = UBC(a,b)  
m = gcd(a,b);  
n = lcm(a,b);
```

Gọi hàm:

```
>> [uoc,boi] = UBC(45,234)  
>> uoc  
    ans = 9  
>> boi  
    ans = 1170
```

3.4 Hàm trong Matlab (function)

Hàm phụ.

- ▶ (khác với hàm chính) chỉ nhằm hỗ trợ tính toán cho hàm chính trong script hàm.
- ▶ Hàm phụ nằm sau hàm chính.

Ví dụ:

```
% chinh.m  
function c = chinh(A)  
c = phu(A)-1  
  
function d = phu(B)  
d = min(B(:))
```

Gọi hàm:

```
>> chinh([23 2 34])  
ans = 1
```

Ma trận trong Matlab

4. Ma trận trong Matlab

- ▶ **Ma trận** là một mảng hình chữ nhật các con số.
- ▶ Ma trận gồm các **dòng (row)** và các **cột (column)**. Các dòng hay cột gọi chung là **vector**.
- ▶ Một con số trong Matlab là một ma trận 1x1.
- ▶ Thế mạnh của Matlab so với các ngôn ngữ lập trình khác là tính toán rất nhanh trên ma trận.

$$\begin{pmatrix} 16 & 3 & 2 & 13 \\ 5 & 10 & 11 & 8 \\ 9 & 6 & 7 & 12 \\ 4 & 15 & 14 & 1 \end{pmatrix}$$

Nhập ma trận

- ▶ Nhập trực tiếp danh sách các phần tử.
- ▶ Phát sinh ma trận bằng hàm sẵn có.
- ▶ Nhập từ file.
- ▶ Tạo ma trận bằng các file .m.

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

```
A =
```

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

- ▶ Dấu [và] mở đầu và kết thúc nhập ma trận.
- ▶ Dấu ; kết thúc một dòng.
- ▶ Các phần tử cách nhau bằng **khoảng trắng** hoặc dấu , .

Tổng các cột và chuyển vị của ma trận

A =

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

```
>> sum(A)
```

ans =

34 34 34 34

A' =

16	5	9	4
3	10	6	15
2	11	7	14
13	8	12	1

```
>> sum(A')
```

ans =

34 34 34 34

Đường chéo của ma trận

```
A =
```

```
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

```
>> diag(A)
```

```
ans =
```

```
    16
    10
     7
     1
```

Trích một phần tử của ma trận

- ▶ Phần tử $A_{i,j}$ được trích bằng biểu thức $A(i,j)$

```
A =
```

```
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

```
>> A(4,2)
```

```
ans =
```

```
    15
```

- ▶ $A(4,2)$ là phần tử ở dòng 4 cột 2, tức là phần tử 15

- ▶ Phép trích chỉ có một chỉ số sẽ theo thứ tự duyệt theo cột (xem ma trận là một vector cột dài)

```
>> A(2)
```

```
ans =
```

```
     5
```

- ▶ $A(2)$ là phần tử thứ 2 duyệt theo cột từ trái qua phải, từ trên xuống dưới.

Chỉ số vượt khỏi kích thước ma trận

A =

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

```
>> t = A(4,5)
```

Attempted to access A(4,5);
index out of bounds because
size(A)=[4,4].

- ▶ Việc truy xuất phần tử vi phạm kích thước ma trận
- ▶ Nằm bên phải phép gán

```
>> X = A;
```

```
>> X(4,5) = 17
```

X =

16	3	2	13	0
5	10	11	8	0
9	6	7	12	0
4	15	14	1	17

- ▶ Mở rộng ma trận
- ▶ Nằm bên trái phép gán

Dấu hai chấm ':'

- ▶ Dấu hai chấm ':' là một trong những phép tính quan trọng của MATLAB.
- ▶ Ví dụ: Tạo một vector dòng gồm các số nguyên từ 1 đến 10

```
>> 1:10
```

```
ans =
```

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

- ▶ Để tạo bước tăng giảm khác 1

```
>> 100:-7:50
```

```
ans =
```

```
100 93 86 79 72 65 58 51
```

```
>> 0:pi/4:pi
```

```
ans =
```

```
0 0.7854 1.5708 2.3562 3.1416
```

Dấu hai chấm ':'

- ▶ $A(1:k, j)$ trích k số đầu tiên ở cột thứ j của ma trận A

```
A =
```

```
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

```
>> A(1:4,4)
```

```
ans =
```

```
    13
     8
    12
     1
```

```
>> sum( A(1:4,4))
```

```
ans =
```

```
    34
```

- ▶ `sum(A(1:4,4))` tính tổng 4 số đầu tiên của cột thứ 4 của ma trận A
- ▶ Dấu hai chấm ':' đứng một mình sẽ chỉ toàn bộ phần tử của dòng hoặc cột
- ▶ Từ khóa "end" chỉ chỉ số cuối cùng của dòng hoặc cột

```
>> A(:,end)
```

```
ans =
```

```
    13
     8
    12
     1
```

Trích nhiều phần tử

- Sử dụng dấu "[,]" để liệt kê vị trí cần trích

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

```
A =
```

```
     2     4     3
```

```
     8     6     7
```

```
>> x = [9 4 2 1]
```

```
x =
```

```
     9     4     2     1
```

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

```
ans =
```

```
     6
```

```
     4
```

```
>> x([2,4])
```

```
ans =
```

```
     4
```

```
     1
```

- Có thể sử dụng dấu ":" để trích dãy các phần tử

```
>> A(2,1:3)
```

```
ans =
```

```
     8     6     7
```

```
>> x(3:-1:1)
```

```
ans =
```

```
     2     4     9
```


Tạo ma trận bằng file.m và lệnh load

- ▶ File.m là một file văn bản ghi các dòng lệnh MATLAB.
- ▶ Có thể soạn thảo bằng MATLAB Editor hoặc bất kỳ trình soạn thảo văn bản nào.
- ▶ Lưu file có đuôi .m
- ▶ Gõ tên file để thực thi nội dung các dòng lệnh trong file.
- ▶ Tạo một file có nội dung như sau:

```
A = [ 16.0 3.0 2.0 13.0  
      5.0 10.0 11.0 8.0  
      9.0 6.0 7.0 12.0  
      4.0 15.0 14.0 1.0 ];
```

- ▶ Lưu với tên magik.m. Dòng lệnh

```
>> magik
```

```
>> A
```

```
A =
```

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

sẽ đọc file và tạo biến A là ma trận như trên.

Ghép hai ma trận

Cho các ma trận sau:

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

A =

1	2	3
4	5	6

```
>> B = [10 12;11 13]
```

B =

10	12
11	13

```
>> C = [7 8 9;9 7 8;8 9 7]
```

C =

7	8	9
9	7	8
8	9	7

► Thêm cột

```
>> D = [A B]
```

D =

1	2	3	10	12
4	5	6	11	13

► Thêm dòng

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

E =

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

Xóa dòng xóa cột

```
A =
```

```
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

```
>> X = A;
```

```
>> X(:,2) = []
```

```
X =
```

```
    16     2    13
     5    11     8
     9     7    12
     4    14     1
```

- ▶ Không được xóa một phần tử kiểu như $X(1,2) = []$

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

```
Subscripted assignment dimension mismatch.
```

- ▶ Dùng chỉ số với dấu ':' để xóa một hay nhiều phần tử

```
>> X(2:2:10) = []
```

```
X =
```

```
    16     9     2     7    13    12     1
```

Các hàm đặc biệt cho ma trận

► zeros

- zeros(n)
- zeros(m,n)
- zeros([m n])
- zeros(size(A))

► ones

- ones(n)
- ones(m,n)
- ones([m n])
- ones(size(A))

► eye

- eye(n)
- eye(m,n)
- eye(size(A))

- pascal
- magic
- numel(A)
- length(A)
- rand(m,n)
- diag(v,k), diag(v)
- tril, triu
- linspace(a,b),
linspace(a,b,n)
- logspace(a,b,n)

Các hàm đặc biệt cho ma trận

Stt	Tên hàm	Ý nghĩa	Ví dụ	Kết Quả
1	<code>zeros(a,b)</code>	tạo ma trận $a \times b$ các phần tử = 0	<code>zeros(2,3)</code>	0 0 0 0 0 0
2	<code>ones(a,b)</code>	tạo ma trận $a \times b$ các phần tử = 1	<code>ones(2,3)</code>	1 1 1 1 1 1
3	<code>eye(a,b)</code>	tạo ma trận $a \times b$ các phần tử đường chéo = 1	<code>eye(3,3)</code>	1 0 0 0 1 0 0 0 1
4	<code>repmat (a,b)</code>	tạo ma trận $b \times b$ các phần tử có giá trị = a	<code>repmat (2,3)</code>	2 2 2 2 2 2 2 2 2

Các hàm đặc biệt cho ma trận

Stt	Tên hàm	Ý nghĩa	Ví dụ	Kết Quả
5	rand(a,b)	tạo ma trận $a \times b$ các phần tử ngẫu nhiên	rand(2,2)	0.2785 0.9575 0.5469 0.9649
6	randn (a,b)	tạo ma trận $a \times b$ các phần tử ngẫu nhiên phân bố đều	randn (2,2)	-1.3499 0.7254 3.0349 -0.0631

Các hàm đặc biệt cho ma trận

Stt	Tên hàm	Ý nghĩa	Ví dụ	Kết Quả
7	linspace (a,b,n)	tạo ma trận hàng b phần tử phân bố đều từ a đến n	linspace (1,3,3)	1 2 3
8	logspace (a,b,n)	tạo ma trận hàng n phần tử phân bố đều từ 10^a đến 10^b	logspace (1,5,3)	10 1000 100000

Các hàm đặc biệt cho ma trận

```
>> D = eye(3)
```

```
D =
```

```
    1    0    0
    0    1    0
    0    0    1
```

```
>> eye(3,2)
```

```
ans =
```

```
    1    0
    0    1
    0    0
```

```
>> pascal(3)
```

```
ans =
```

```
    1    1    1
    1    2    3
    1    3    6
```

```
>> magic(3)
```

```
ans =
```

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


Các hàm đặc biệt cho ma trận

```
>> A = zeros(3)
```

```
A =
```

```
    0    0    0
    0    0    0
    0    0    0
```

```
>> B = zeros(2,3)
```

```
B =
```

```
    0    0    0
    0    0    0
```

```
>> size(A)
```

```
ans =
```

```
    3    3
```

```
>> zeros(size(B))
```

```
ans =
```

```
    0    0    0
    0    0    0
```

Các hàm đặc biệt cho ma trận

```
>> B = zeros(2,3)
```

```
B =
```

```
    0    0    0
    0    0    0
```

```
>> numel(B)
```

```
ans =
```

```
6
```

```
>> length(B)
```

```
ans =
```

```
3
```

```
>> rand(3,2)
```

```
ans =
```

```
0.8003    0.9157
0.1419    0.7922
0.4218    0.9595
```

```
>> C = ones(3)
```

```
C =
```

```
1    1    1
1    1    1
1    1    1
```

Các phép toán trong ma trận và vector

Phép tính	Phép tính
$+, -$	Cộng hoặc trừ hai ma trận cùng kích thước
$A * B$	Nhân hai ma trận A và B
A/B	Chia hai ma trận (chia phải) A và B
$A \setminus B$	Chia trái hai ma trận B và A
$A .* B$	Nhân từng phần tử của hai ma trận A và B
$A ./ B$	Chia từng phần tử của hai ma trận A và B
$A . \setminus B$	Chia từng phần tử của hai ma trận B và A
\wedge	Mũ cho từng phần tử của mảng

Các phép toán trong ma trận và vector

Stt	Dòng lệnh	Ý nghĩa	Kết quả	
1	A=[1 2;3 4]	Tạo ma trận	1 2 3 4	
2	B=[1,2;3,4]	Tạo ma trận	1 2 3 4	
3	2*A	Nhân một số với ma trận	2 4 6 8	
4	B/2	Chia ma trận cho một số	0.5000 1.0000 1.5000 2.0000	

Khi đem ma trận nhân hoặc chia cho một số, ta sẽ được một ma trận cùng kích thước, và các phần tử sẽ có giá trị bằng phần tử tương ứng của ma trận cũ nhân hoặc chia cho số đó.

Các phép toán trong ma trận và vector

Stt	Dòng lệnh	Ý nghĩa	Kết quả
5	A+B	Cộng hai ma trận	2 4 6 8
6	A-B	Trừ hai ma trận	0 0 0 0

- 2 ma trận đem cộng hoặc trừ phải cùng kích thước

- Kết quả là một ma trận có cùng kích thước, các phần tử là tổng hoặc hiệu của 2 phần tử tương ứng

7	A*B	Nhân hai ma trận	7 10 15 22
8	A\B	Chia hai ma trận	1 0 0 1

- Chỉ áp dụng giải phương trình $AX = B$, với A là ma trận vuông, B là ma trận cột có cùng kích thước với A

Các phép toán trong ma trận và vector

Stt	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
1	size	Kích thước ma trận	Z	4 4
2	ndims	số chiều	ndims(Z)	2
3	length	chiều dài	length(Z)	4
4	numel	số phần tử ma trận	numel(Z)	16
5	max	vector hàng, chứa các phần tử lớn nhất theo từng cột	max(Z)	4 6 8 10

Các phép toán trong ma trận và vector

Stt	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
6	min	vector hàng, chứa các phần tử nhỏ nhất theo từng cột	min(Z)	1 3 5 7
7	sum	vector hàng, chứa tổng các phần tử theo từng cột	sum(Z)	10 18 26 34
8	sort	Sắp xếp theo thứ tự tăng dần trong từng cột	sum(Z)	1 3 5 7 2 4 6 8 3 5 7 9 4 6 8 10

Các phép toán trong ma trận và vector

$$A = \begin{bmatrix} 11 & 2 & 6 \\ 4 & 17 & 26 \\ 17 & 8 & 49 \end{bmatrix}$$

Stt	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
1	det	Tính định thức ma trận	det(A)	5825
2	,	Chuyển vị ma trận	A'	<div> <div>11</div> <div>4</div> <div>17</div> <div>2</div> <div>17</div> <div>8</div> <div>6</div> <div>26</div> <div>49</div> </div>

Các phép toán trong ma trận và vector

Stt	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
3	tril	Tạo ma trận tam giác dưới	tril(A)	$\begin{bmatrix} 11 & 0 & 0 \\ 2 & 17 & 0 \\ 6 & 26 & 49 \end{bmatrix}$
4	triu	Tạo ma trận tam giác trên	triu(A)	$\begin{bmatrix} 11 & 4 & 17 \\ 0 & 17 & 8 \\ 0 & 0 & 49 \end{bmatrix}$
5	inv	Ngược đảo ma trận	inv(A)	$\begin{bmatrix} 0.1073 & -0.0086 & -0.0086 \\ 0.0422 & 0.0750 & -0.0450 \\ -0.0441 & -0.0093 & 0.0307 \end{bmatrix}$

Các phép toán trong ma trận và vector

- ▶ **Lệnh phân tích ma trận thành thừa số cholesky**
 - `chol(A)`
- ▶ Ma trận A sẽ được phân tích thành tích của hai ma trận

$$A = R'.R$$

Với R là ma trận tam giác trên.

▶ Ví dụ:

```
A =  
    11     2     6  
     4    17    26  
    17     8    49  
  
>> chol(A)  
  
ans =  
    3.3166    0.6030    1.8091  
         0    4.0788    6.1070  
         0         0    2.9037
```

Các phép toán trong ma trận và vector

- ▶ **Lệnh phân tích ma trận thành thừa số LU**

$$\text{[L,U]} = \text{lu}(A)$$

- ▶ Ma trận A sẽ được phân tích thành tích của hai ma trận

$$A = L.U$$

Với L là ma trận tam giác dưới, U là ma trận tam giác trên.

Các phép toán trong ma trận và vector

► Ví dụ:

A =

11	2	6
4	17	26
17	8	49

>> [L,U] = lu(A)

L =

0.6471	-0.2101	1.0000
0.2353	1.0000	0
1.0000	0	0

U =

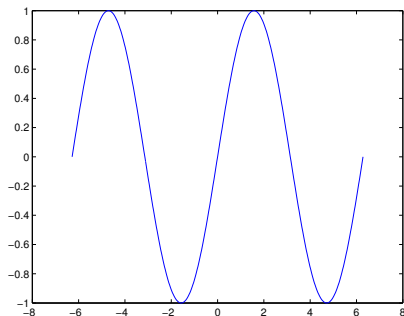
17.0000	8.0000	49.0000
0	15.1176	14.4706
0	0	-22.6654

Đồ thị 2D trong Matlab

5. Đồ thị 2D trong Matlab

- ▶ Hàm `plot(x,y)` vẽ các điểm x , y tương ứng lên mặt phẳng.
- ▶ Ví dụ

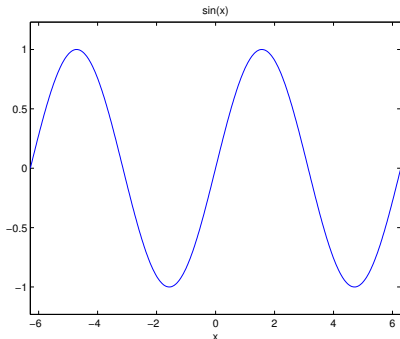
```
>> x = -2*pi:pi/100:2*pi;  
>> plot(x,sin(x))
```



5. Đồ thị 2D trong Matlab

- **Chú ý:** Đối với hàm sơ cấp có thể dùng lệnh `ezplot('Hàm số cần vẽ')`. Lệnh sau vẽ đồ thị hàm $\sin(x)$

```
>> ezplot('sin(x)')
```



Vẽ nhiều đồ thị

Để vẽ nhiều đồ thị ta sử dụng hàm `plot` với cú pháp sau

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

Matlab sẽ vẽ một đường x theo t và một đường y theo t .
Hoặc chúng ta có thể sử dụng lệnh `hold`

```
>> hold
```

Lệnh `hold` sẽ lưu giữ đồ thị hiện hữu, khi chúng ta thực hiện lệnh vẽ tiếp theo thì đồ thị mới sẽ được thêm vào đồ thị cũ.

Vẽ nhiều đồ thị

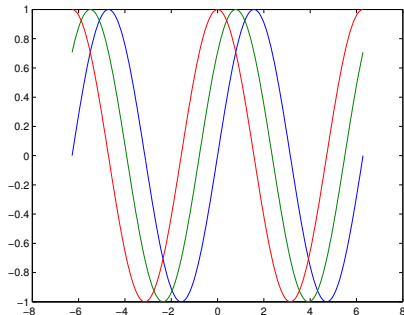
- ▶ Dùng dạng
`plot(x, y1, x, y2, ...)`

- ▶ Ví dụ

```
>> x = -2*pi:pi/100:2*pi;  
>> y1 = sin(x);  
>> y2 = sin(x + pi/4);  
>> y3 = sin(x + pi/2);  
>> plot(x, y1, x, y2,  
x, y3)
```

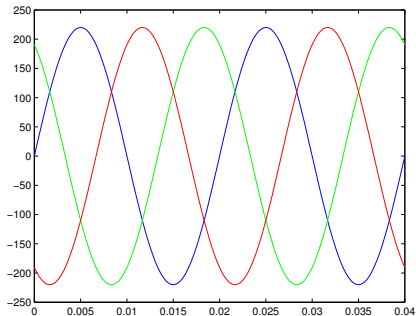
- ▶ Cũng có thể dùng
`plot(x, y)` trong đó

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



Vẽ nhiều đồ thị

```
f = 50;  
T = 1/f;  
t = 0:T/100:2*T;  
va = 220*sin(2*pi*f*t);  
plot(t,va,'blue');  
hold on;  
vb = 220*sin(2*pi*f*t +  
+ 120*pi/180);  
plot(t,vb,'green');  
hold on;  
vc = 220*sin(2*pi*f*t +  
- 120*pi/180);  
plot(t,vc,'red');
```



Các thuộc tính nét vẽ

Cú pháp

```
>>plot(X1,Y1,LineSpec,...)
```

Hàm plot có sử dụng các thuộc tính nét vẽ LineSpec cho phép thiết lập các thuộc tính của đồ thị. Các thuộc tính là:

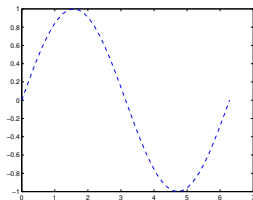
- ▶ Line style : định dạng kiểu nét, ví dụ nét chấm, nét đứt, ...
- ▶ Line style : định dạng độ rộng nét vẽ, đơn vị là point
- ▶ Color : định dạng màu của nét vẽ, ví dụ màu xanh, màu đỏ,...
- ▶ Marker type: định dạng điểm vẽ.

Thuộc tính**Loại nét vẽ**

- : nét liền
- - : nét đứt
- : : nét chấm
- . : nét chấm gạch

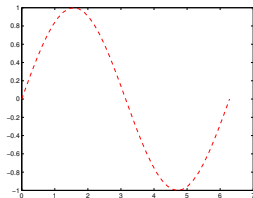
Lập trình trong Matlab

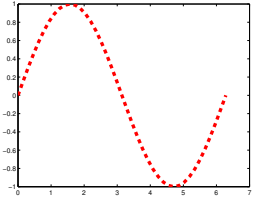
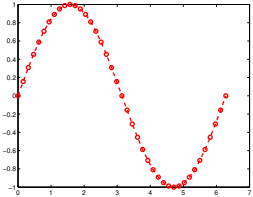
```
t = 0:pi/20:2*pi;
plot(t,sin(t),'--')
```

Đồ thị**Màu nét vẽ**

- r : đỏ
- y : vàng
- w : trắng
- b : xanh nước
- biển
- k : đen

```
t = 0:pi/20:2*pi;
plot(t,sin(t),'-- r')
```



Thuộc tính	Lập trình trong Matlab	Đồ thị
<p>Độ lớn nét vẽ</p> <p>Tính theo đơn vị là point</p>	<pre>t = 0:pi/20:2*pi; plot(t,sin(t),'-- r', 'LineWidth',5)</pre>	
<p>Marker</p> <p>+ : dấu cộng</p> <p>o : vòng tròn</p> <p>^ : dấu mũ</p> <p>* : hoa thị</p> <p>x : chữ x</p>	<pre>t = 0:pi/20:2*pi; plot(t,sin(t),'-- r o', 'LineWidth',2)</pre>	

Lệnh cho trục tọa độ

▶ Lệnh axis

```
>> axis([xmin xmax ymin ymax])
```

▶ Tùy chỉnh các kiểu trục tọa độ

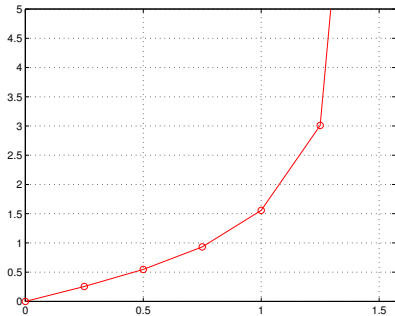
- ▶ axis on/off/auto
- ▶ axis normal/square/equal/tight
- ▶ axis ij/xy
- ▶ grid on/off

▶ Xác định giới hạn của trục Ox và Oy

```
>> xlim([xmin xmax])  
>> ylim([ymin ymax])
```

Ví dụ

```
>> x = 0:0.25:pi/2;  
>> y = tan(x);  
  
>> plot(x,y,'-ro');  
  
>> axis([0 pi/2 0 5])  
>> grid on
```



Chú thích đồ thị

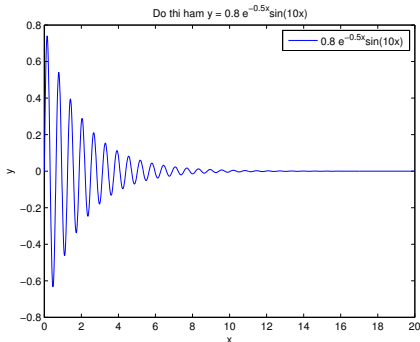


xlabel, ylabel, title, legend



Ví dụ

```
>> x = 0:0.01:20;  
>> y = 0.8*exp(-0.5*x)  
.*sin(10*x);  
>> plot(x,y);  
  
>> xlabel('x');  
>> ylabel('y');  
>> legend('0.8 e-0.5xsin(10x)');  
>> title('Do thi ham  
y = 0.8 e-0.5xsin(10x)');
```



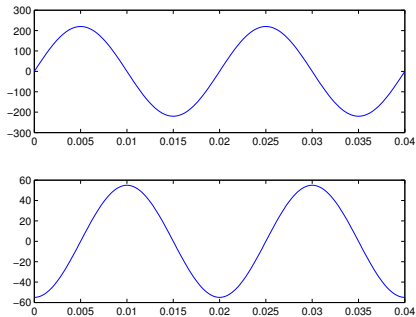
Hàm subplot

```
>> subplot(p,q,i)
```

Hàm subplot sẽ chia số thành một bảng các ô nhỏ, với p là số dòng của các ô và q là số cột của các ô. Đồ thị sẽ được vẽ ở ô thứ i .

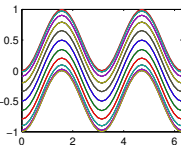
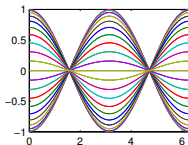
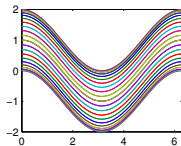
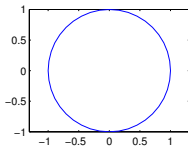
```
f = 50;  
T = 1/f;  
t = 0:T/100:2*T;  
v = 220*sin(2*pi*f*t);  
i = 55*sin(2*pi*f*t  
- 90*pi/180);
```

```
subplot(2,1,1);  
plot(t,v);  
subplot(2,1,2);  
plot(t,i);
```



Ví dụ

```
t = 0:pi/20:2*pi; [x,y] = meshgrid(t);  
subplot(2,2,1)  
plot(sin(t),cos(t))  
axis equal  
subplot(2,2,2)  
z = sin(x)+cos(y);  
plot(t,z)  
axis([0 2*pi -2 2])  
subplot(2,2,3)  
z = sin(x).*cos(y);  
plot(t,z)  
axis([0 2*pi -1 1])  
subplot(2,2,4)  
z = (sin(x).^2)-(cos(y).^2);  
plot(t,z)  
axis([0 2*pi -1 1])
```



Hàm vẽ các đồ thị khác

a. Hàm bar: vẽ đồ thị thanh đứng

```
>> bar(x,y)
```

Hàm bar sẽ vẽ các thanh đứng có độ cao là y_i tương ứng tại vị trí x_i

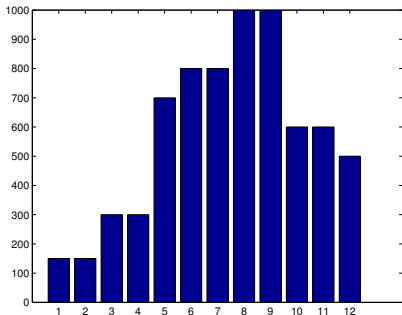
Ví dụ: Vẽ đồ thị phụ tải của một xí nghiệp theo số liệu sau:

Tháng	1	2	3	4	5	6	7	8
Công suất - kw	150	150	300	300	700	800	800	1000

Tháng	9	10	11	12
Công suất - kw	1000	600	600	500

Ví dụ

```
>> x = [1 2 3 4 5 6 7 8 9  
10 11 12];  
>> y = [150 150 300 300 700  
800 800 1000 1000 600 600  
500];  
  
>> bar(x,y)
```



Hàm vẽ các đồ thị khác

b. Hàm pie: vẽ đồ thị dạng hình quạt

```
>> pie(x)
```

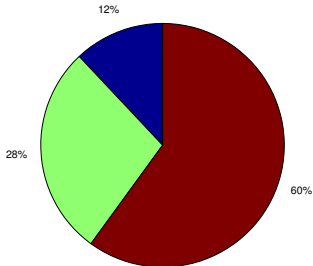
Hàm pie sẽ vẽ đồ thị dạng hình quạt với diện tích tỉ lệ với phần trăm.

Ví dụ: Cho xí nghiệp có điện năng tiêu thụ hàng tháng được tính toán theo giờ cao điểm, thấp điểm, trung bình như sau

Điện năng	Cao điểm	Thấp điểm	Trung bình
Điện năng - kw	15000	35000	75000

Ví dụ

```
>> x = [15000 35000 75000];  
>> pie(x)
```



Về các mặt

Bước đầu tiên là đưa ra đồ thị lưới của hàm hai biến $z = f(x, y)$, tương ứng với ma trận X và Y chứa các hàng và các cột lặp đi lặp lại, MATLAB cung cấp hàm `meshgrid` cho mục đích này:

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

Tạo ma trận X mà hàng của nó là bản sao của vectơ x , và ma trận Y có các cột là bản sao của vectơ y . Cặp ma trận này sau đó được sử dụng để ước lượng hàm hai biến sử dụng đặc tính toán học về mảng của MATLAB

Vẽ các mặt

Để vẽ bề mặt ta sử dụng các hàm:

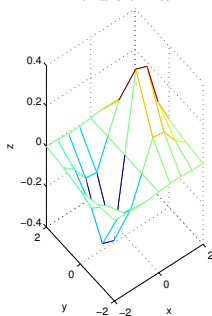
- ▶ `mesh(X,Y,Z)`: nối các điểm với nhau trong một lưới chữ nhật.
- ▶ `meshc(X,Y,Z)`: vẽ các đường *contour* bên dưới đồ thị.
- ▶ `meshz(X,Y,Z)`: vẽ các đường thẳng đứng viền quanh đồ thị.
- ▶ `waterfall(X,Y,Z)`: vẽ mặt với hiệu ứng như thác đổ.

Ví dụ: Vẽ mặt xác định bởi phương trình

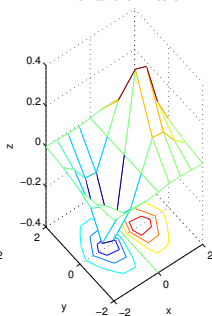
$$z(x,y) = xe^{-x^2-y^2}$$


```
x = -2:0.5:2;  
y = -2:1:2;  
[X,Y] = meshgrid(x,y);  
Z = X.*exp(-X.^2 - Y.^2);  
subplot(1,2,1);  
mesh(X,Y,Z)  
xlabel('x');  
ylabel('y');  
zlabel('z');  
title('Ve mat voi lenh mesh')  
subplot(1,2,2);  
meshc(X,Y,Z)  
xlabel('x');  
ylabel('y');  
zlabel('z');  
title('Ve mat voi lenh meshc');
```

Ve mat voi lenh mesh

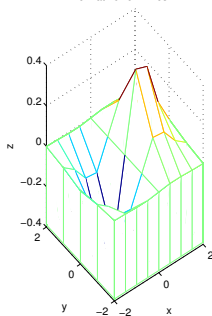


Ve mat voi lenh meshc

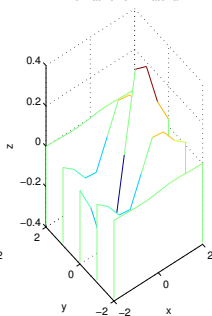


```
x = -2:0.5:2;  
y = -2:1:2;  
[X,Y] = meshgrid(x,y);  
Z = X.*exp(-X.^2 - Y.^2);  
subplot(1,2,1);  
meshz(X,Y,Z)  
xlabel('x');  
ylabel('y');  
zlabel('z');  
title('Ve mat voi lenh meshz');  
subplot(1,2,2);  
waterfall(X,Y,Z)  
xlabel('x');  
ylabel('y');  
zlabel('z');  
title('Ve mat voi lenh waterfall');
```

Ve mat voi lenh meshz



Ve mat voi lenh waterfall

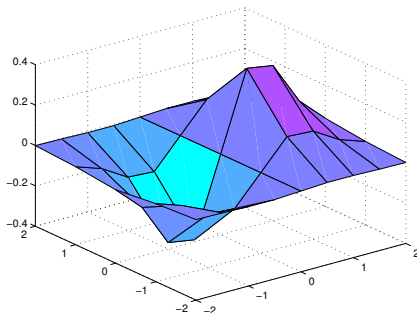


Vẽ các mặt được tô bóng từ một ma trận bằng hàm surf, surfc

Ví dụ: Vẽ mặt xác định bởi phương trình

$$z(x, y) = xe^{-x^2 - y^2}$$

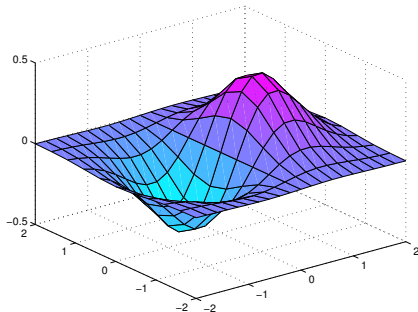
```
x = -2:0.5:2;  
y = -2:1:2;  
[X,Y] = meshgrid(x,y);  
Z = X.*exp(-X.^2 - Y.^2);  
  
surf(X,Y,Z);  
colormap('cool');
```



Ví dụ

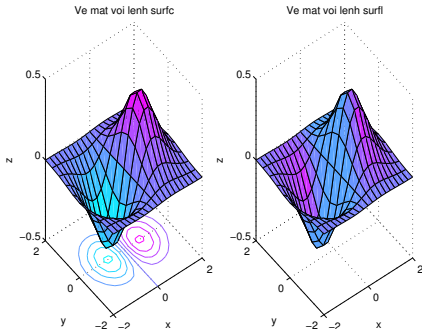
Ta có thể tạo nhiều lưới hơn để mượt hơn.

```
x = -2:0.2:2;  
y = -2:0.4:2;  
[X,Y] = meshgrid(x,y);  
Z = X.*exp(-X.^2 - Y.^2);  
  
surf(X,Y,Z);  
colormap('cool');
```



- ▶ Lệnh `surf(X,Y,Z)`: vẽ các đường *contour* bên dưới đồ thị
- ▶ Lệnh `surf1(X,Y,Z,s)`: vẽ mặt có bóng sáng. Đối số s xác định hướng của nguồn sáng trên bề mặt vẽ. s là một vectơ tùy chọn trong hệ tọa độ decac hay trong tọa độ cầu. Nếu không khai báo giá trị mặc định của s là 45 theo chiều kim đồng hồ từ vị trí người quan sát

```
x = -2:0.2:2;  
y = -2:0.4:2;  
[X,Y] = meshgrid(x,y);  
Z = X.*exp(-X.^2 - Y.^2);  
subplot(1,2,1);  
surfc(X,Y,Z)  
xlabel('x');  
ylabel('y');  
zlabel('z');  
colormap(cool);  
title('Ve mat voi lenh surfc');  
subplot(1,2,2);  
surfl(X,Y,Z)  
xlabel('x');  
ylabel('y');  
zlabel('z');  
title('Ve mat voi lenh surfl');  
colormap(cool)
```



Phép tính với biến symbolic

Khai báo

► Khai báo biến

```
>> syms a b c x
```

Hoặc

```
>> a = sym('a')  
>> b = sym('b')  
>> c = sym('c')  
>> x = sym('x')
```

► Khai báo biến phức

```
>> syms x y real;  
>> z = x + i*y
```

Hoặc

```
>> x = sym('x','real');  
>> y = sym('y','real');  
>> z = x + i*y
```


Khai báo biểu thức

- ▶ Khai báo biểu thức:

$$f = 2x + b$$

```
>> syms x b
```

```
>> f = 2*x + b
```

Hoặc

```
>> f = sym('2*x + b')
```

- ▶ Lưu ý:

```
>> g = sym('5')
```

Khác

```
>> g = 5
```

Tìm biến hình thức

- ▶ Lệnh `findsym(f)`: Tìm biến hình thức trong biểu thức
- ▶ Lệnh `findsym(f,1)`: Tìm biến hình thức mặc định trong biểu thức
- ▶ **Ví dụ:**

```
>> syms a b n t x z
>> f = x^n
>> g = sin(a*t + b)
```

```
>> findsym(f)
ans =
n,x
>> findsym(g)
ans =
a,b,t
```

```
>> syms a b n t x z
>> f = x^n
>> g = sin(a*t + b)
```

```
>> findsym(g,1)
ans =
t
>> findsym(f,1)
ans =
x
```

Hiển thị biểu hình học dưới dạng số học

```
>> t = 0.1;
>> sym(t,'f')
ans =
3602879701896397/36028797018963968
>> sym(t,'r')
ans =
1/10
>> sym(t,'e')
ans =
eps/40 + 1/10
>> sym(t,'d')
ans =
0.100000000000000000555111512312578
>> digits(7)
>> sym(t,'d')
ans =
0.1
```

Các phép toán cơ bản

Stt	Lệnh trong MATLAB	Ý nghĩa	Kết quả
1	<code>x = sym('x','real')</code>	Tạo biến x là Tạo biến số thực	
2	<code>x = sym('x','positive')</code>	Tạo biến x là số thực dương	
3	<code>syms x y;</code>	Định nghĩa 2 biến x, y	
4	<code>syms</code>	Liệt kê các biến mà chương trình quản lý	'x' 'y'
5	<code>A = x + 1</code> <code>B = y^2 - 1</code>	Tạo hai biến symbolic mới A và B	

Các phép toán cơ bản

Stt	Lệnh trong MATLAB	Ý nghĩa	Kết quả
6	A+B	Cộng 2 biến A và B	$y^2 + x$
7	A-B	Trừ 2 biến A và B	$- y^2 + x + 2$
8	A*B	Nhân 2 biến A và B	$(y^2 - 1)*(x + 1)$
9	A/B	Biến A chia biến B	$(x + 1)/(y^2 - 1)$
10	A^B	A lũy thừa B	$(x + 1)^(y^2 - 1)$

Các phép tính đạo hàm và tích phân

Stt	Lệnh trong MATLAB	Ý nghĩa	Kết quả
11	<code>diff(B)</code>	Tính đạo hàm biểu thức B	$2*y$
12	<code>diff(B,2)</code>	Tính đạo hàm biểu thức B bậc 2	2
13	<code>diff(A*B,y)</code>	Tính đạo hàm biểu thức A*B b theo biến y	$2*y*(x + 1)$
14	<code>diff(A*B,y,2)</code>	Tính đạo hàm bậc 2 biểu thức A*B b theo biến y	$2*x + 2$

Các phép tính đạo hàm và tích phân

Stt	Lệnh trong MATLAB	Ý nghĩa	Kết quả
15	<code>int(A)</code>	Tính tích phân biểu thức A	$(x*(x + 2))/2$
16	<code>int(A,0,5)</code>	Tính tích phân biểu thức A từ 0 đến 5	$35/2$
17	<code>int(A*B,x)</code>	Tính tích phân biểu thức A*B theo biến x	$((y^2 - 1)*(x + 1)^2)/2$

Đạo hàm

► Lệnh:

```
diff(Y)
```

Với Y là hàm số hoặc biểu hình thức cần lấy đạo hàm.

► Ví dụ:

```
>> syms x;  
>> f = sin(5*x);  
>> diff(f)  
ans =  
5*cos(5*x)  
  
>> g = exp(x)*cos(x)  
>> diff(g)  
ans =  
exp(x)*cos(x)  
- exp(x)*sin(x)
```

```
>> c = sym('5');  
  
>> diff(c)  
ans =  
0  
  
>> diff(5)  
ans =  
[]
```

Vì 5 không phải biểu hình thức.

Đạo hàm cấp 2

► Lệnh:

```
diff(Y,2)
```

Hoặc

```
diff(diff(Y))
```

Với Y là hàm số hoặc biểu thức cần lấy đạo hàm.

► Ví dụ:

```
>> syms x
>> g = exp(x)*cos(x)
g =
exp(x)*cos(x)

>> diff(g,2)
ans =
-2*exp(x)*sin(x)
```

```
>> syms x
>> g = exp(x)*cos(x)
g =
exp(x)*cos(x)

>> diff(diff(g))
ans =
-2*exp(x)*sin(x)
```

Đạo hàm đa biến

Gọi $f = f(x,y)$ thì

► Đạo hàm theo x

```
diff(f,x)
```

► Đạo hàm cấp 2 theo x

```
diff(f,x,2)
```

► Nếu x là biến mặc định của f thì

```
diff(f,2)
```

Tương đương với

```
diff(f,x,2)
```

► Đạo hàm theo y

```
diff(f,y)
```

► Đạo hàm cấp 2 theo y

```
diff(f,y,2)
```

Ví dụ

```
>> syms s t
>> f = sin(s*t);

>> diff(f,t)
ans =
s*cos(s*t)

>> diff(f,s)
ans =
t*cos(s*t)
```

```
>> findsym(f,1)
ans =
t
>> diff(f,t,2)
ans =
-s^2*sin(s*t)

>> diff(f,2)
ans =
-s^2*sin(s*t)
```

Ta thấy kết quả từ lệnh `findsym`
`t` là biến mặc định nên
`diff(f,t,2) = diff(f,2)`

Đạo hàm với ma trận

```
>> syms a x
>> A = [cos(a*x) sin(a*x); -sin(a*x) cos(a*x) ]

A =
[ cos(a*x), sin(a*x)]
[ -sin(a*x), cos(a*x)]

>> diff(A)

ans =
[ -a*sin(a*x),  a*cos(a*x)]
[ -a*cos(a*x), -a*sin(a*x)]
```

Tích Phân

- ▶ Lệnh tìm nguyên hàm của hàm $f = f(x)$

```
int(f,x)
```

Hoặc

```
int(f)
```

- ▶ Lệnh tính tích phân của hàm f từ a đến b

```
int(f,x,a,b)
```

Hoặc

```
int(f,a,b)
```

Ví dụ

```
>> syms x n a b t
```

```
>> f = x^n;
```

```
>> int(f,x)
```

```
ans =
```

```
piecewise([n == -1, log(x)],  
[n ~= -1, x^(n + 1)/(n + 1)])
```

```
>> int(f)
```

```
ans =
```

```
piecewise([n == -1, log(x)],  
[n ~= -1, x^(n + 1)/(n + 1)])
```

```
>> g = cos(a*t + b)
```

```
>> int(g)
```

```
ans =
```

```
sin(b + a*t)/a
```

```
>> h = sin(2*x);
```

```
>> int(h,0,pi/2)
```

```
ans =
```

```
1
```

```
>> u = exp(-x^2);
```

```
>> int(u,0,inf)
```

```
ans =
```

```
pi^(1/2)/2
```

Giới hạn

► $\lim_{x \rightarrow 0} f(x)$

```
limit(f)
```

► $\lim_{x \rightarrow a} f(x)$

```
limit(f,x,a)
```

Hoặc

```
limit(f,x)
```

► $\lim_{x \rightarrow a^-} f(x)$

```
limit(f,x,a,'left')
```

► $\lim_{x \rightarrow a^+} f(x)$

```
limit(f,x,a,'right')
```

Ví dụ

```
>> syms h n x
>> limit( ( cos(x+h)-cos(x) )/h, h, 0 )
ans =
-sin(x)
>> limit( (1+x/n)^n, n, inf )
ans =
exp(x)
>> limit( x/abs(x), x, 0, 'left' )
ans =
-1
>> limit( x/abs(x), x, 0, 'right' )
ans =
1
>> limit( x/abs(x), x, 0 )
ans =
NaN
```


Tổng chuỗi

Tính:

$$S_1 = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots$$

$$S_2 = 1 + x + x^2 + \dots$$

```
>> syms x k
```

```
>> S1 = symsum(1/k^2, 1, inf )
```

```
S1 =
```

```
pi^2/6
```

```
>> S2 = symsum(x^k, k, 0, inf)
```

```
S2 =
```

```
piecewise([1 <= x, Inf], [abs(x) < 1, -1/(x - 1)])
```

Các phép tính toán thu gọn

Stt	Lệnh trong MATLAB	Ý nghĩa	Kết quả
1	<code>collect((x+1)^2 + (x+y)^2, y)</code>	Gom các lũy thừa cùng bậc theo biến y	$y^2 + (2*x)*y + (x + 1)^2 + x^2$
2	<code>expand((x+y+1)^2)</code>	Khai triển các biểu thức	$x^2 + 2*x*y + 2*x + y^2 + 2*y + 1$
3	<code>factor(x^3-y^3)</code>	Phân tích thành thừa số	$(x - y)*(x^2 + x*y + y^2)$
4	<code>simplify((x^3 - y^3)/(x^2 - y^2))</code>	Thu gọn biểu thức	$x + y^2/(x + y)$

collect

- ▶ `collect(f)` - $f = f(x)$
- ▶ `collect(f,y)` - $f = f(x,y,\dots)$
 - ▶ Đơn giản hàm f bằng các nhóm các biến x có cùng số mũ.
 - ▶ Trường hợp f có nhiều biến `collect(f,y)` sẽ chỉ định gom nhóm theo biến y
 - ▶ `collect(f)` gom nhóm theo biến mặc định được chỉ ra trong `findsym(f)`.

Ví dụ

```
>> syms x t
```

```
>> f = x^3 - 6*x^2 + 11*x - 6;
```

```
>> g = (x - 1)*(x - 2)*(x - 3);
```

```
>> h = -6 + ( 11 + (-6 + x)*x )*x;
```

```
>> pretty(f)
```

$$x^3 - 6x^2 + 11x - 6$$

```
>> pretty(g)
```

$$(x - 1)(x - 2)(x - 3)$$

```
>> pretty(h)
```

$$x(x(x - 6) + 11) - 6$$

Ví dụ

```
>> collect(f)
ans =
x^3 - 6*x^2 + 11*x - 6
>> collect(g)
ans =
x^3 - 6*x^2 + 11*x - 6
>> collect(h)
ans =
x^3 - 6*x^2 + 11*x - 6
```

```
>> f = (1 + x)*t + x*t;
>> collect(f)
ans =
(2*t)*x + t
>> collect(f,t)
ans =
(2*x + 1)*t
```

expand

- ▶ `expand(f)` - Khai triển biểu thức f
- ▶ **Ví dụ:**

```
>> syms x y a b
>> f = a*(x+y);
>> expand(f)
ans =
a*x + a*y
>> g = (x-1)*(x-2)*(x-3);
>> expand(g)
ans =
x^3 - 6*x^2 + 11*x - 6
```

```
>> h = exp(a+b);
>> expand(h)
ans =
exp(a)*exp(b)
>> k = cos(3*x);
>> expand(k)
ans =
4*cos(x)^3 - 3*cos(x)
```

factor

- ▶ `factor(f)` - phân tích thành thừa số
- ▶ **Ví dụ:**

```
>> syms x
>> f = x^3 - 6*x^2 + 11*x - 6;
>> g = x^3 - 6*x^2 + 11*x - 5;
>> h = x^6 + 1;
>> factor(f)
ans =
(x - 3)*(x - 1)*(x - 2)
>> factor(g)
ans =
x^3 - 6*x^2 + 11*x - 5
>> factor(h)
ans =
(x^2 + 1)*(x^4 - x^2 + 1)
```

simplify

- ▶ `simplify(f)`- đơn giản biểu thức f .
- ▶ Ví dụ:

```
>> syms x
>> f = x*(x*(x - 6) + 11)-6;
>> simplify(f)
ans =
(x - 1)*(x - 2)*(x - 3)
>> g = (1 - x^2)/(1 - x);
>> simplify(g)
ans =
x + 1
```

```
>> syms x y positive
>> simplify(log(x*y))
ans =
log(x*y)
>> h = cos(x)^2 + sin(x)^2;
>> simplify(h)
ans =
1
```


simple

- ▶ `simple(f)`- rút gọn biểu thức f , kết hợp các phép toán của `simplify`, `collect`, `factor`.
- ▶ Ví dụ:

```
>> syms a
>> f = (1/a^3 + 6/a^2
+ 12/a + 8)^(1/3);
>> simplify(f)
ans =
((2*a + 1)^3/a^3)^(1/3)
>> simple(f)
ans =
((2*a + 1)^3/a^3)^(1/3)
```

```
>> syms x y positive
>> h = log(x*y);
>> simplify(h)
ans =
log(x*y)
>> simple(f)
ans =
log(x*y)
```

- ▶ `poly2sym(a,x)`- tạo một đa thức theo biến `x` với các hệ số được lấy lần lượt từ mảng `a`
- ▶ **Ví dụ:**

```
>> syms x
>> a = [1 4 -7 -10];
>> p = poly2sym(a,x)
p =
x^3 + 4*x^2 - 7*x - 10
```

- ▶ `s = sym2poly(p)`- trích các hệ số của đa thức `p` chứa vào mảng `s`.
- ▶ **Ví dụ:**

```
>> syms x
>> p = 4*x^3 - 2*x^2 + 5*x -16;
>> s = sym2poly(p)
s =
    4    -2     5   -16
```

Ứng dụng Matlab

Tính toán trong đại số tuyến tính

► Khai báo ma trận

```
>> syms a b c d t
```

```
>> A = [a b; c d]
```

```
A =
```

```
[ a, b]
```

```
[ c, d]
```

```
>> B = [cos(t) sin(t);
```

```
        -sin(t) cos(t)]
```

```
B =
```

```
[ cos(t), sin(t)]
```

```
[ -sin(t), cos(t)]
```

```
>> C = [t 1 0; 1 t 1; 0 1 t]
```

```
C =
```

```
[ t, 1, 0]
```

```
[ 1, t, 1]
```

```
[ 0, 1, t]
```

```
>> D = round(rand(3,3))
```

```
D =
```

```
      1      1      0
```

```
      1      1      1
```

```
      0      0      1
```

```
>> D = sym(D)
```

```
D =
```

```
[ 1, 1, 0]
```

```
[ 1, 1, 1]
```

```
[ 0, 0, 1]
```

Tính toán trong đại số tuyến tính

Các phép toán với hai ma trận A và B

- ▶ Phép cộng: $A + B$
- ▶ Phép trừ: $A - B$
- ▶ Phép nhân: $A*B$, $A \setminus B$ ($A*inv(B)$), $A \setminus B$ ($inv(A)*B$)
- ▶ Lũy thừa: A^n
- ▶ Phép chuyển vị: $A.'$

Tính toán trong đại số tuyến tính

Các hàm xử lý ma trận:

- ▶ `inv(A)` : Tìm ma trận nghịch đảo của ma trận A
- ▶ `det(A)` : Tính định thức của ma trận A
- ▶ `rank(A)` : Tìm hạng của ma trận A
- ▶ `diag(A)` : Trích đường chéo của ma trận A
- ▶ `tril(A)` : Tạo ma trận tam giác dưới từ ma trận A
- ▶ `triu(A)` : Tạo ma trận tam giác trên từ ma trận A

Ví dụ

```
>> c = floor(10*rand(4))
```

```
c =
```

```
    9    4    9    0
```

```
    1    8    7    8
```

```
    9    1    9    9
```

```
    9    4    6    6
```

```
>> D = sym(c)
```

```
D =
```

```
[ 9, 4, 9, 0]
```

```
[ 1, 8, 7, 8]
```

```
[ 9, 1, 9, 9]
```

```
[ 9, 4, 6, 6]
```

Ví dụ

```
>> A = inv(D)
A =
    -5/576, -5/64, -7/144, 17/96]
    1/128, 9/128, -5/32, 9/64]
    67/576, 3/64, 17/144, -23/96]
   -125/1152, 3/128, 17/288, 3/64]
>> inv(A)*A
ans =
    1, 0, 0, 0]
    0, 1, 0, 0]
    0, 0, 1, 0]
    0, 0, 0, 1]
>> det(A)
ans =
1/3456
```


Ví dụ

```

>> b = ones(1,4)
b =
     1     1     1     1
>> x = b/A
x =
[ 28, 17, 31, 23]
>> x*A
ans =
[ 1, 1, 1, 1]

```

Ví dụ

```
>> A^3
ans =
[ 359105/254803968,      2899/1048576,      266299/63700992,
 -370301/42467328]
[ -465101/56623104,      54425/18874368,      32257/14155776,
 55307/28311552]
[ -148135/254803968, -123815/28311552,      -208061/63700992,
 368459/42467328]
[ 2256281/509607936,      35993/56623104, -335357/127401984,
 -43687/28311552]
```

Giải phương trình đại số

- ▶ `solve(f)`- Giải phương trình $f(x) = 0$
- ▶ **Ví dụ:**

```
>> syms a b c x
>> f = a*x^2 + b*x + c;
>> solve(f)

ans =
-(b + (b^2 - 4*a*c)^(1/2))/(2*a)
-(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

Giải phương trình đại số

- ▶ `solve(f)` - Giải phương trình theo biến mặc định được chỉ ra trong hàm `findsym(f,1)`, ở ví dụ trên

```
>> findsym(f,1)
ans =
x
```

- ▶ `solve(f,a)` - Giải phương trình theo biến chỉ định là `a` (tương tự cho `b`, `c`).

- ▶ **Ví dụ:**

```
>> syms a b c x
>> f = a*x^2 + b*x + c;
>> solve(f,b)
ans =
-(a*x^2 + c)/x
```

Giải phương trình đại số

- ▶ `solve('f(x) = g(x)')` - Giải phương trình $f(x) = g(x)$.
Lưu ý phải đặt trong dấu nháy.
- ▶ **Ví dụ:**

```
>> syms x
>> s = solve('cos(2*x) + sin(x) = 1')

s =

      0
    pi/6
(5*pi)/6
```

Giải phương trình đại số

- ▶ `solve('f(x)', 'g(x)', h(x), ...)`- Giải hệ nhiều phương trình.
- ▶ **Ví dụ:** Giải hệ:

$$\begin{cases} x^2 y^2 = 0 \\ x - \frac{y}{2} = \alpha \end{cases}$$

```
>> syms x y alpha
>> [x, y] = solve('x^2*y^2 = 0','x - y/2 = alpha')
x =
    alpha
     0

y =
     0
-2*alpha
```

Ví dụ

Giải hệ

$$\begin{cases} u^2 + v^2 &= a^2 \\ u + v &= 1 \\ a^2 - 2 * a &= 3 \end{cases}$$

```
>> syms a u v  
>> S = solve('u^2 + v^2 =  
a^2', 'u + v = 1',  
'a^2 - 2*a = 3')
```

S =

```
  a: [4x1 sym]  
  u: [4x1 sym]  
  v: [4x1 sym]
```

```
>> S.a  
ans =  
    -1  
    -1  
     3  
     3
```

Giải phương trình đạo hàm riêng

- ▶ dsolve- Hàm giải phương trình - hệ phương trình đạo hàm riêng.
- ▶ **Ví dụ 1:** Giải

$$\frac{dy}{dt} = 1 + y^2, y(0) = 1$$

```
>> y = dsolve('Dy = 1 + y^2', 'y(0) = 1')
```

```
y =  
tan(pi/4 + t)
```


Giải phương trình đạo hàm riêng

► Ví dụ 2: Giải

$$\frac{d^2y}{dx^2} = \cos(2x) - y, y(0) = 1, \frac{d}{dx}y(0) = 0$$

```
>> y = dsolve('D2y = cos(2*x) - y', 'y(0) = 1',  
    'Dy(0) = 0', 'x');
```

```
>> simplify(y)
```

```
ans =
```

```
1 - (8*(cos(x)/2 - 1/2)^2)/3
```

Giải phương trình đạo hàm riêng

► Ví dụ 3: Giải

$$\begin{cases} \frac{d^3}{dx^3} = u \\ u(0) = 1; u'(0) = -1; u''(0) = \pi \end{cases}$$

```
>> dsolve('D3u = u', 'u(0) = 1', 'Du(0) = -1',  
'D2u(0) = pi', 'x')
```

```
ans =
```

```
(pi*exp(x))/3 - exp(-x/2)*cos((3^(1/2)*x)/2)*(pi/3 -  
(3^(1/2)*exp(-x/2)*sin((3^(1/2)*x)/2)*(pi + 1))/3
```

Giải phương trình đạo hàm riêng

► Ví dụ 4: Giải

$$\begin{cases} \frac{df}{dt} = 3f(t) + 4g(t), f(0) = 0 \\ \frac{dg}{dt} = -4f(t) + 3g(t), g(0) = 1 \end{cases}$$

```
>> [f,g] = dsolve('Df = 3*f + 4*g','Dg = -4*f + 3*g',  
'f(0) = 0', 'g(0) = 1')
```

```
f =  
sin(4*t)*exp(3*t)
```

```
g =  
cos(4*t)*exp(3*t)
```

