Chương 1: Matlab cơ bản

Viện Toán ứng dụng và Tin học, ĐHBK Hà Nội

Hà Nội, tháng 8 năm 2015

Nội dung

Matlab là gì

- MatLab (MATrix LABoratory) là một môi trường phần mềm (problem-solving environment - PSE) trong toán học tính toán.
- MatLab được phát triển vào cuối những năm 70 bởi Cleve Moler (Stanford) với mục đích giúp sinh viên thực hiện các tính toán số mà không cần phải học một ngôn ngữ lập trình bậc thấp, ví dụ Fortran.
- Được thiết kế bởi công ty MathWorks là một ngôn ngữ lập trình bậc cao chuyên sử dụng cho các tính toán kỹ thuật, đặc biệt là các bài toán có dạng ma trận hoặc vector. MatLab tích hợp các tính toán, đồ họa và lập trình trong một môi trường thân thiện, cho phép thể hiện các bài toán và nghiệm dưới dạng các ký hiệu toán học quen thuộc.
- MatLab là một hệ tương tác, có các thành phần dữ liệu cơ bản là một mảng mà không cần khai báo trước số chiều.
- MatLab đã trải qua nhiều năm phát triển với sự đóng góp của nhiều chuyên gia. Trong trường đại học, nó là một công cụ chuẩn cho các khóa học về toán, kỹ thuật và khoa học từ mở đầu đến nâng cao. Trong công nghiệp, MatLab là một công cụ hữu ích cho việc nghiên cứu, phát triển và phân tích các sản phẩm c lượng cao.

Matlab là gì

- MatLab (MATrix LABoratory) là một môi trường phần mềm (problem-solving environment - PSE) trong toán học tính toán.
- MatLab được phát triển vào cuối những năm 70 bởi Cleve Moler (Stanford) với mục đích giúp sinh viên thực hiện các tính toán số mà không cần phải học một ngôn ngữ lập trình bậc thấp, ví dụ Fortran.
- Được thiết kế bởi công ty MathWorks là một ngôn ngữ lập trình bậc cao chuyên sử dụng cho các tính toán kỹ thuật, đặc biệt là các bài toán có dạng ma trận hoặc vector. MatLab tích hợp các tính toán, đồ họa và lập trình trong một môi trường thân thiện, cho phép thể hiện các bài toán và nghiệm dưới dạng các ký hiệu toán học quen thuộc.
- MatLab là một hệ tương tác, có các thành phần dữ liệu cơ bản là một mảng mà không cần khai báo trước số chiều.
- MatLab đã trải qua nhiều năm phát triển với sự đóng góp của nhiều chuyên gia. Trong trường đại học, nó là một công cụ chuẩn cho các khóa học về toán, kỹ thuật và khoa học từ mở đầu đến nâng cao. Trong công nghiệp, MatLab là một công cụ hữu ích cho việc nghiên cứu, phát triển và phân tích các sản phẩm c lượng cao.

Matlab là gì

- MatLab (MATrix LABoratory) là một môi trường phần mềm (problem-solving environment - PSE) trong toán học tính toán.
- MatLab được phát triển vào cuối những năm 70 bởi Cleve Moler (Stanford) với mục đích giúp sinh viên thực hiện các tính toán số mà không cần phải học một ngôn ngữ lập trình bậc thấp, ví dụ Fortran.
- Được thiết kế bởi công ty MathWorks là một ngôn ngữ lập trình bậc cao chuyên sử dụng cho các tính toán kỹ thuật, đặc biệt là các bài toán có dạng ma trận hoặc vector. MatLab tích hợp các tính toán, đồ họa và lập trình trong một môi trường thân thiện, cho phép thể hiện các bài toán và nghiệm dưới dạng các ký hiệu toán học quen thuộc.
- MatLab là một hệ tương tác, có các thành phần dữ liệu cơ bản là một mảng mà không cần khai báo trước số chiều.
- MatLab đã trải qua nhiều năm phát triển với sự đóng góp của nhiều chuyên gia. Trong trường đại học, nó là một công cụ chuẩn cho các khóa học về toán, kỹ thuật và khoa học từ mở đầu đến nâng cao. Trong công nghiệp, MatLab là một công cụ hữu ích cho việc nghiên cứu, phát triển và phân tích các sản phẩm c lượng cao.

Matlab là gì

- MatLab (MATrix LABoratory) là một môi trường phần mềm (problem-solving environment - PSE) trong toán học tính toán.
- MatLab được phát triển vào cuối những năm 70 bởi Cleve Moler (Stanford) với mục đích giúp sinh viên thực hiện các tính toán số mà không cần phải học một ngôn ngữ lập trình bậc thấp, ví dụ Fortran.
- Được thiết kế bởi công ty MathWorks là một ngôn ngữ lập trình bậc cao chuyên sử dụng cho các tính toán kỹ thuật, đặc biệt là các bài toán có dạng ma trận hoặc vector. MatLab tích hợp các tính toán, đồ họa và lập trình trong một môi trường thân thiện, cho phép thể hiện các bài toán và nghiệm dưới dạng các ký hiệu toán học quen thuộc.
- MatLab là một hệ tương tác, có các thành phần dữ liệu cơ bản là một mảng mà không cần khai báo trước số chiều.
- MatLab đã trải qua nhiều năm phát triển với sự đóng góp của nhiều chuyên gia. Trong trường đại học, nó là một công cụ chuẩn cho các khóa học về toán, kỹ thuật và khoa học từ mở đầu đến nâng cao. Trong công nghiệp, MatLab là một công cụ hữu ích cho việc nghiên cứu, phát triển và phân tích các sản phẩm c lượng cao.

Matlab là gì

- MatLab (MATrix LABoratory) là một môi trường phần mềm (problem-solving environment - PSE) trong toán học tính toán.
- MatLab được phát triển vào cuối những năm 70 bởi Cleve Moler (Stanford) với mục đích giúp sinh viên thực hiện các tính toán số mà không cần phải học một ngôn ngữ lập trình bậc thấp, ví dụ Fortran.
- Được thiết kế bởi công ty MathWorks là một ngôn ngữ lập trình bậc cao chuyên sử dụng cho các tính toán kỹ thuật, đặc biệt là các bài toán có dạng ma trận hoặc vector. MatLab tích hợp các tính toán, đồ họa và lập trình trong một môi trường thân thiện, cho phép thể hiện các bài toán và nghiệm dưới dạng các ký hiệu toán học quen thuộc.
- MatLab là một hệ tương tác, có các thành phần dữ liệu cơ bản là một mảng mà không cần khai báo trước số chiều.
- MatLab đã trải qua nhiều năm phát triển với sự đóng góp của nhiều chuyên gia. Trong trường đại học, nó là một công cụ chuẩn cho các khóa học về toán, kỹ thuật và khoa học từ mở đầu đến nâng cao. Trong công nghiệp, MatLab là một công cụ hữu ích cho việc nghiên cứu, phát triển và phân tích các sản phẩm chất lượng cao.

Matlab là gì

MatLab ứng dụng trong

- Toán học và tính toán
- Phát triển các thuật toán
- Thu thập dữ liệu
- Mô hình hóa, mô phỏng
- Phân tích dữ liệu, thăm dò và trực quan hóa
- Đồ họa khoa học và kỹ thuật
- Phát triển các ứng dụng, xây dựng các giao diện người dùng

Matlab là gì

MatLab ứng dụng trong

- Toán học và tính toán
- Phát triển các thuật toán
- Thu thập dữ liệu
- Mô hình hóa, mô phỏng
- Phân tích dữ liệu, thăm dò và trực quan hóa
- Đồ họa khoa học và kỹ thuật
- Phát triển các ứng dụng, xây dựng các giao diện người dùng

Matlab là gì

MatLab ứng dụng trong

- Toán học và tính toán
- Phát triển các thuật toán
- Thu thập dữ liệu
- Mô hình hóa, mô phỏng
- Phân tích dữ liệu, thăm dò và trực quan hóa
- Đồ họa khoa học và kỹ thuật
- Phát triển các ứng dụng, xây dựng các giao diện người dùng

Matlab là gì

MatLab ứng dụng trong

- Toán học và tính toán
- Phát triển các thuật toán
- Thu thập dữ liệu
- Mô hình hóa, mô phỏng
- Phân tích dữ liệu, thăm dò và trực quan hóa
- Đồ họa khoa học và kỹ thuật
- Phát triển các ứng dụng, xây dựng các giao diện người dùng

Matlab là gì

MatLab ứng dụng trong

- Toán học và tính toán
- Phát triển các thuật toán
- Thu thập dữ liệu
- Mô hình hóa, mô phỏng
- Phân tích dữ liệu, thăm dò và trực quan hóa
- Đồ họa khoa học và kỹ thuật
- Phát triển các ứng dụng, xây dựng các giao diện người dùng

Matlab là gì

MatLab ứng dụng trong

- Toán học và tính toán
- Phát triển các thuật toán
- Thu thập dữ liệu
- Mô hình hóa, mô phỏng
- Phân tích dữ liệu, thăm dò và trực quan hóa
- Đồ họa khoa học và kỹ thuật
- Phát triển các ứng dụng, xây dựng các giao diện người dùng

Matlab là gì

MatLab ứng dụng trong

- Toán học và tính toán
- Phát triển các thuật toán
- Thu thập dữ liệu
- Mô hình hóa, mô phỏng
- Phân tích dữ liệu, thăm dò và trực quan hóa
- Đồ họa khoa học và kỹ thuật
- Phát triển các ứng dụng, xây dựng các giao diện người dùng

Matlab là gì

MatLab ứng dụng trong

- Toán học và tính toán
- Phát triển các thuật toán
- Thu thập dữ liệu
- Mô hình hóa, mô phỏng
- Phân tích dữ liệu, thăm dò và trực quan hóa
- Đồ họa khoa học và kỹ thuật
- Phát triển các ứng dụng, xây dựng các giao diện người dùng

Matlab là gì

Nét đặc trưng của MatLab là nó cung cấp một họ các Toolboxes, cho phép người dùng có thể học và áp dụng trong các kỹ thuật chuyên ngành. Toolboxes là tập hợp của các hàm ("M-files") cho phép mở rộng môi trường MatLab để giải một lớp các bài toán trong

- Xử lý tín hiệu (signal processing)
- Các hệ điều khiển (control systems)
- Mạng nơ-ron (neural networks)
- Logic mờ (fuzzy logic)
- Sóng nhỏ (wavelets)
- Mô phỏng (simulation)
- ...

Matlab là gì

Nét đặc trưng của MatLab là nó cung cấp một họ các Toolboxes, cho phép người dùng có thể học và áp dụng trong các kỹ thuật chuyên ngành. Toolboxes là tập hợp của các hàm ("M-files") cho phép mở rộng môi trường MatLab để giải một lớp các bài toán trong

- Xử lý tín hiệu (signal processing)
- Các hệ điều khiển (control systems)
- Mạng nơ-ron (neural networks)
- Logic mờ (fuzzy logic)
- Sóng nhỏ (wavelets)
- Mô phỏng (simulation)
- ...

Matlab là gì

Nét đặc trưng của MatLab là nó cung cấp một họ các Toolboxes, cho phép người dùng có thể học và áp dụng trong các kỹ thuật chuyên ngành. Toolboxes là tập hợp của các hàm ("M-files") cho phép mở rộng môi trường MatLab để giải một lớp các bài toán trong

- Xử lý tín hiệu (signal processing)
- Các hệ điều khiển (control systems)
- Mạng nơ-ron (neural networks)
- Logic mờ (fuzzy logic)
- Sóng nhỏ (wavelets)
- Mô phỏng (simulation)
- ...

Matlab là gì

Nét đặc trưng của MatLab là nó cung cấp một họ các Toolboxes, cho phép người dùng có thể học và áp dụng trong các kỹ thuật chuyên ngành. Toolboxes là tập hợp của các hàm ("M-files") cho phép mở rộng môi trường MatLab để giải một lớp các bài toán trong

- Xử lý tín hiệu (signal processing)
- Các hệ điều khiển (control systems)
- Mạng nơ-ron (neural networks)
- Logic mờ (fuzzy logic)
- Sóng nhỏ (wavelets)
- Mô phỏng (simulation)
- ...

Matlab là gì

Nét đặc trưng của MatLab là nó cung cấp một họ các Toolboxes, cho phép người dùng có thể học và áp dụng trong các kỹ thuật chuyên ngành. Toolboxes là tập hợp của các hàm ("M-files") cho phép mở rộng môi trường MatLab để giải một lớp các bài toán trong

- Xử lý tín hiệu (signal processing)
- Các hệ điều khiển (control systems)
- Mạng nơ-ron (neural networks)
- Logic mờ (fuzzy logic)
- Sóng nhỏ (wavelets)
- Mô phỏng (simulation)
- ...

Matlab là gì

Nét đặc trưng của MatLab là nó cung cấp một họ các Toolboxes, cho phép người dùng có thể học và áp dụng trong các kỹ thuật chuyên ngành. Toolboxes là tập hợp của các hàm ("M-files") cho phép mở rộng môi trường MatLab để giải một lớp các bài toán trong

- Xử lý tín hiệu (signal processing)
- Các hệ điều khiển (control systems)
- Mạng nơ-ron (neural networks)
- Logic mờ (fuzzy logic)
- Sóng nhỏ (wavelets)
- Mô phỏng (simulation)
- ...

Matlab là gì

Nét đặc trưng của MatLab là nó cung cấp một họ các Toolboxes, cho phép người dùng có thể học và áp dụng trong các kỹ thuật chuyên ngành. Toolboxes là tập hợp của các hàm ("M-files") cho phép mở rộng môi trường MatLab để giải một lớp các bài toán trong

- Xử lý tín hiệu (signal processing)
- Các hệ điều khiển (control systems)
- Mạng nơ-ron (neural networks)
- Logic mờ (fuzzy logic)
- Sóng nhỏ (wavelets)
- Mô phỏng (simulation)
- ...

Matlab là gì

Nét đặc trưng của MatLab là nó cung cấp một họ các Toolboxes, cho phép người dùng có thể học và áp dụng trong các kỹ thuật chuyên ngành. Toolboxes là tập hợp của các hàm ("M-files") cho phép mở rộng môi trường MatLab để giải một lớp các bài toán trong

- Xử lý tín hiệu (signal processing)
- Các hệ điều khiển (control systems)
- Mạng nơ-ron (neural networks)
- Logic mờ (fuzzy logic)
- Sóng nhỏ (wavelets)
- Mô phỏng (simulation)
- ...

Nội dung

Biến (Variables) (1)

- MatLab không yêu cầu phải khai báo biến cũng như số chiều. Trong MatLab , một biến được khai báo và khởi tạo thông qua lệnh gán, ví dụ:

```
>> num = 98
num =
98
>> pi = 3.1415926535897931
pi =
3.1416
>> msg = 'Hello World'
msg =
Hello World
```

- Tên biến bao gồm các ký tự chữ, số và ký hiệu gạch dưới (_). Tên biến phải bắt đầu bằng ký tự chữ và có độ dài tùy thích. Tuy nhiên, MatLab chỉ sử dụng N ký tự đầu tiên được tính bằng lệnh

```
>> N = namelengthmax
N =
63
```

Biến (Variables)(2)

- Ví dụ các kiểu tên biến hợp lệ: `arg1`, `no_name`, `vars`, `Vars`. Khi tên biến không hợp lệ, sẽ xuất hiện dòng thông báo lỗi:

```
>> 4rum = 'Forum'
??? 4rum = 'Forum'
|
Error: Unexpected matlab expression.
```

- Ta có thể kiểm tra tính hợp lệ của tên biến bằng lệnh `isvarname`

```
>> isvarname('4u')
ans =
     0
```

- MatLab phân biệt chữ hoa và chữ thường. Do đó `A` và a là các biến khác nhau
- Khi MatLab gặp một tên biến mới, nó tự động tạo ra và lưu trong bộ nhớ. Nếu biến đó đã tồn tại, MatLab sẽ thay đổi giá trị và nếu cần, cấp phát bộ nhớ mới.

Số (Numbers)

- MatLab sử dụng ký hiệu thập phân theo qui ước với số chữ số tùy chọn và các dấu +,- cho các số. Ký hiệu khoa học sử dụng chữ cái e cho lũy thừa của 10. Số phức sử dụng các chữ i hoặc j cho đơn vị ảo. Một số ví dụ

3	-99	0.0001
9.6397238	1.60210e-20	6.02252e23
1i	-3.14159j	3e5i

- Tất cả các số được lưu trữ bên trong bằng cách sử dụng `long format` theo chuẩn dấu chấm động `IEEE` (Institute of Electrical and Electronics Engineers). Các số dưới dạng dấu chấm động có độ chính xác hữu hạn với 16 chữ số thập phân có nghĩa và nằm trong khoảng $\left(10^{-308}, 10^{+308}\right)$.

Các toán tử (Operators)

Các biểu thức MatLab sử dụng các toán tử quen thuộc theo thứ tự ưu tiên (từ dưới lên trên).

Phép toán	**Ý nghĩa**
+	Cộng
-	Trừ
*	Nhân
/	Chia
\	Chia trái
^	Lũy thừa
'	Chuyển vị, chuyển vị liên hợp phức
( )	Xác định thứ tự ưu tiên của các phép toán

Các hàm (Functions) (1)

- MatLab cung cấp một số lượng rất phong phú các hàm toán học sơ cấp (elementary mathematical functions), ví dụ `abs, sqrt, exp, sin`.
- Phép lấy căn bậc hai hay logarit của số âm sẽ tự động cho một giá trị phức thích hợp.
- MatLab cũng đồng thời cung cấp rất nhiều các hàm toán học nâng cao (advanced mathematical functions), bao gồm các hàm Bessel và gamma. Hầu hết các hàm này chấp nhận đối số phức.
- Để hiển thị danh sách các hàm toán học sơ cấp, nhập vào

```
>> help elfun
```

Tương tự đối với danh sách các hàm toán học nâng cao và hàm xử lý ma trận

```
>> help specfun
>> help elmat
```

Các hàm (Functions) (2)

- Một số các hàm như `sqrt, sin` được cài đặt sẵn (built-in functions). Các hàm này là một phần của nhân MATLAB nên chúng rất hiệu quả, nhưng ta không biết được các tính toán chi tiết. Các hàm khác, ví dụ `bessel` được lập trình dưới dạng m-files.
- Có một số sự khác biệt giữa các hàm được cài đặt sẵn và các hàm khác. Ví dụ, với các hàm built-in, ta không thể xem mã còn đối với các hàm khác ta có thể xem mã và thậm chí sửa đổi nếu muốn. Kiểm chứng điều này bằng các lệnh

```
>> type sqrt
>> type bessel
```

- Nhiều hàm đặc biệt cho ta giá trị của các hằng số hữu ích

`pi`	$3.14159265\ldots$
`i,j`	Đơn vị ảo, $\sqrt{-1}$
`eps`	Sai số tương đối dạng dấu chấm động, $\varepsilon = 2.2204e-016$
`realmin`	Số thực nhỏ nhất, $2.2251e-308$
`realmax`	Số thực lớn nhất, $1.7977e+308$
`Inf`	∞
`NaN`	Not-a-number

Các hàm (Functions) (3)

- `inf` được tạo bởi phép chia một số khác 0 cho 0, hoặc việc tính giá trị của một biểu thức toán học đúng đắn mà bị tràn bộ nhớ, tức là vượt quá `realmax`.
- `NaN` được tạo ra khi cố gắng tính giá trị của một biểu thức dạng $\frac{0}{0}$ hoặc $\infty - \infty$.
- Ta có thể gán cho các hằng giá trị mới, ví dụ

  ```
  >> eps = 1.e-6
  ```

  và sử dụng giá trị này cho một dãy tính toán. Giá trị ban đầu sẽ được phục hồi với

  ```
  >> clear eps
  ```

Nội dung

Vector

Vector là một ma trận có một hàng hoặc một cột. Để khởi tạo vector hàng chứa các giá trị rời rạc, các phần tử trong vector phải nằm trong cặp ngoặc vuông [] và được ngăn cách bởi dấu phẩy "," hoặc khoảng trắng ␣.

```
>> arr1 = [1 2 3]
arr1 =
     1     2     3
>> arr2 = [0,-5]
arr2 =
     0    -5
>> arr3 = [arr1 arr2]
arr3 =
     1     2     3     0    -5
```

Vector

Để khởi tạo vector hàng chứa các giá trị liên tiếp hoặc cách nhau một giá trị nhất định (bước nhảy), MatLab sử dụng toán tử ":", đồng thời giá trị đầu và cuối của vector không cần thiết phải đặt trong dấu ngoặc vuông [].

```
>> arr1 = 1:5
arr1 =
     1     2     3     4     5
>> arr2 = [1:0.5:2]
arr2 =
    1.0000    1.5000    2.0000
>> arr3 = 10:-1:6
arr3 =
    10     9     8     7     6
```

Để tạo vector rỗng (không chứa phần tử nào) ta khai báo như sau:

```
>> emp_arr = []
emp_arr =
     []
```

Vector

Ngược lại, để tạo vector cột, ta cần chuyển vị vector hàng bằng cách dùng toán tử "'" hoặc dùng dấu ";" để ngăn cách các phần tử

```
>> col_arr=[1:3]'
col_arr =
     1
     2
     3
>> col_arr=[1;2;3]
col_arr =
     1
     2
     3
```

Vector

Hàm `linspace`

Cú pháp

```
y = linspace(a,b)
y = linspace(a,b,n)
```

Mô tả

- Hàm `linspace` tạo ra một vector với khoảng cách tuyến tính. Nó tương tự như toán tử hai chấm ":", nhưng xác định trước số điểm chia n.
- `y = linspace(a,b)` tạo ra một vector hàng y với 100 điểm cách đều bao gồm cả a và b.
- `y = linspace(a,b,n)` tạo ra một vector hàng y với n điểm cách đều bao gồm cả a và b. Với $n < 2$, hàm `linspace` trả về b.

Vector

Hàm `logspace`

Cú pháp

```
y = logspace(a,b)
y = logspace(a,b,n)
y = logspace(a,pi)
```

Mô tả

- Hàm `linspace` tạo ra một vector với khoảng cách logarit.
- `y = logspace(a,b)` tạo ra một vector hàng y gồm 50 điểm trong khoảng 10^a và 10^b.
- `y = logspace(a,b,n)` tạo ra một vector hàng y với n điểm trong khoảng 10^a và 10^b. Nếu $n < 2$, trả về 10^b.
- `y = logspace(a,pi)` tạo ra các điểm giữa 10^a and π, thường sử dụng trong xử lý tín hiệu số.

Vector

Chỉ số

Giá trị của một phần tử tại một vị trí bất kỳ trong vector được truy xuất thông qua chỉ số. Trong MatLab , chỉ số **luôn bắt đầu từ 1** và có thể là một giá trị đơn hoặc một mảng

- Trích phần tử thứ i: `X(i)`
- Trích nhiều phần tử: `X([danh sách các vị trí])`

```
>> arr = 10:-1:0
arr =
    10     9     8     7     6     5     4     3     2     1     0
>> arr(5)
ans =
     6
>> arr([7,8,11])
ans =
     4     3     0
```

- Để xóa một phần tử trong vector, ta gán phần tử đó với vector rỗng:

```
>> arr([2 5]) = []
arr =
    10     8     7     5     4     3     2     1     0
```

Vector

Vector và biểu thức logic

Biểu thức logic cho phép truy xuất một cách linh hoạt đến các thành phần của một vector hay ma trận. Ví dụ

```
>> x = [-1 0 2 3 5 6 7 4 9];
>> x>0
ans =
     0     0     1     1     1     1     1     1     1
>> x(x>0)
ans
     2     3     5     6     7     4     9
>> x(x>2 & x<=5)
ans =
     3     5     4
>> x>2
ans =
      0     0     0     1     1     1     1     1     1
```

Vector

Các hàm logic: `all`, `any` và `find`

- `any`: Kiểm tra xem có tồn tại một phần tử của vector thỏa mãn điều kiện nào đó không. Nếu có thì trả về 1, ngược lại là 0. Ví dụ

```
>> x=[-1 2 3];
>> any(x>0)
ans =
     1
```

- `all`: Kiểm tra xem tất cả các phần tử của vector thỏa mãn điều kiện nào đó không. Ví dụ

```
>> all(x<0)
ans =
     0
```

- `find`: trả về các chỉ số của một vector thỏa mãn một điều kiện nào đó. Ví dụ

```
>> A = [1 2 4;4 5 6]
>> find(isprime(A))% xuất ra các vị trí có giá trị là một số nguyên tố
ans =
     3
     4
```

Vector

Các phép toán cơ bản trên vector

```
a.*b; % nhân từng từ
a.^b % trả về vector dạng (a_1^{b_1},...,a_n^{b_n})
a.^n; % lũy thừa từng từ
a.\b; % chia trái
a./b; % chia phải
a & b; % không nhầm lẫn với &&
a | b; % không nhầm lẫn với ||
~a; % phủ định
sort(a); sort(a,'descend'); % Sắp xếp mảng a theo thứ tự tăng, giảm dần
arrayfun(@fn,a); % tính giá trị hàm fn tại từng thành phần của a
                 % (không mấy khi dùng)
isequal(a,b); % Đúng nếu a==b
ismember(a,b);  % Đúng khi mọi phần tử của a đều là phần tử của b
intersect(a,b); % Các phần tử chung của a và b (phép giao 2 tập hợp)
union(a,b); % Tất cả các phần tử thuộc a hoặc b (phép hợp 2 tập hợp)
setdiff(a,b); %  Các phần tử thuộc a mà không thuộc b (hiệu 2 tập hợp)
setxor(a,b); % Các phần tử không thuộc phần chung của a và b
```

Nội dung

Đa thức trong MatLab

- $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$
- n là bậc của đa thức
- Ví dụ

$$\begin{aligned} f(x) &= 2x^2 - 4x + 10 \text{ bậc } 2 \\ f(x) &= 6 \text{ bậc } 0 \end{aligned}$$

- Trong MatLab , đa thức được biểu diễn bởi một vector hàng có các thành phần chính là các hệ số. Vector trên phải chứa tất cả các hệ số, kể cả hệ số bậc 0.
- Ví dụ

$8x + 5$	`p=[8 5]`
$6x^2 - 150$	`h=[6 0 -150]`

Giá trị của đa thức

- MatLab có thể tính giá trị của một đa thức tại điểm x sử dụng hàm

  `polyval(p,x)`

  trong đó

  - `p` là vector biểu diễn đa thức
  - `x` là một số, biến hoặc biểu thức

- Ví dụ

```
>> p=[5 6 -7 3];
>> x=2;
>> y=polyval(p,x)
y =
    53
```

Nghiệm của đa thức

- Nhắc lại rằng nghiệm của đa thức là các giá trị của biến sao cho giá trị của đa thức tại đó bằng 0
- MatLab có thể tìm các nghiệm của một đa thức bằng lệnh

```
r=roots(p)
```

trong đó

  - `p` là vector biểu diễn đa thức
  - `r` vector cột chứa các nghiệm của đa thức

- Ví dụ

```
>> p=[1 -2 -3];
>> r=roots(p)
r =
     3.0000
    -1.0000
```

Tìm đa thức khi biết trước các nghiệm

- Cho trước các nghiệm của một đa thức, MatLab có thể tính các hệ số của đa thức đó bằng lệnh

  `p=poly(r)`

  trong đó

  - `r` là vector hàng hoặc cột chứa các nghiệm của đa thức
  - `p` là vector hàng chứa các hệ số

- Ví dụ

```
>> r=[-3;2];
>> p=poly(r)
p =
     1     1    -6
% f(x)=x^2+x-6
```

Cộng đa thức

- Để cộng, trừ hai vector trong MatLab các vector hệ số cần phải cùng kích cỡ, do đó vector có độ dài ngắn hơn phải thêm các phần tử 0
- Ví dụ

```
% f1(x)=3x^6+15x^5-10x^3-3x^2+15x-40
% f2(x)=3x^3-2x-6
>> p1=[3 15 0 -10 -3 15 -40];
>> p2=[0 0 0 3 0 -2 -6];
>> p=p1+p2
p =
     3    15     0    -7    -3    13   -46
% f(x)=3x^6+15x^5-7x^3-3x^2+13x-46
```

Nhân đa thức

- Cú pháp

  `c=conv(a,b)`

  trong đó

  - a và b là các vector hệ số của các đa thức
  - c là vector hệ số của tích

- Ví dụ

```
>> a=[2 1 -3];
>> b=[1 1];
>> c=conv(a,b)
c =
     2     3    -2    -3
```

Chia đa thức

- Cú pháp

  `[q,r]=deconv(u,v)`

  trong đó

  - `u` vector hệ số của các đa thức bị chia
  - `v` vector hệ số của các đa thức chia
  - `q` là vector hệ số của thương
  - `r` là vector hệ số của phần dư

- Ví dụ

```
>> u=[1 -9 -10];
>> v=[1 1];
>> [q,r]=deconv(u,v)
q =
     1   -10
r =
     0     0     0
```

Đạo hàm của đa thức

- MatLab có thể tính đạo hàm của đa thức bởi lệnh

```
k=polyder(p)
```

  - p là vector hệ số của đa thức
  - k là vector hệ số của đạo hàm

- Ví dụ

```
>> p=[3 -2 4];
>> k=polyder(p)
k =
     6    -2
```

Nguyên hàm của đa thức

- MatLab có thể tính nguyên hàm của đa thức bởi lệnh

```
g=polyint(h,k)
```

  - `h` là vector hệ số của đa thức
  - `g` là vector hệ số của nguyên hàm
  - `k` là hằng số tích phân, mặc định là 0

- Ví dụ

```
>> h=[6 0 0];
>> g=polyint(h)
g =
     2     0     0     0
```

Nội dung

Nhập ma trận (1)

Khi nhập ma trận trong môi trường dòng lệnh ta phải tuân theo các qui định sau:

- Ngăn cách các phần tử của ma trận bởi dấu "," hay khoảng trắng ␣
- Dùng dấu ";" để kết thúc một hàng
- Bao các phần tử của ma trận bởi cặp dấu [].

Ví dụ nhập một ma trận

```
>> 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
```

Nhập ma trận (2)

Bây giờ ta nhập lệnh

```
>> sum(A)
ans =
    34    34    34    34
```

nghĩa là nó lấy tổng các cột vì MatLab được viết để làm việc với cột. Muốn lấy tổng của của các hàng ta cần chuyển vị ma trận

```
>> A'
ans =
    16     5     9     4
     3    10     6    15
     2    11     7    14
    13     8    12     1
```

Chú ý

Ma trận a = [] là ma trận rỗng.

Nhập ma trận (3)

Chỉ số

- Phần tử ở hàng `i`, cột `j` của ma trận (cỡ $m \times n$) `A` là `A(i,j)`.
- Ta cũng có thể tham chiếu tới phần tử của mảng nhờ một chỉ số, ví dụ $A(k)$ với $k = i + (j-1)m$ (duyệt theo cột, từ trên xuống dưới, từ trái qua phải). Để chuyển từ chỉ số ma trận sang chỉ số mảng 1 chiều dùng lệnh

```
>> k=sub2ind(size(A),i,j)
```

Ngược lại, để chuyển từ chỉ số mảng 1 chiều sang chỉ số ma trận, dùng hàm `ind2sub`

```
>> [i,j]=ind2sub(size(A),k)
```

- Trong MatLab , chỉ số cuối cùng của hàng hay cột của ma trận hoặc vector có thể thay thế bởi `end`. Ví dụ:

```
>> x=[1 2 3; 4 5 6];
>> y=x(1:end,1:end-1)
y =
     1     2
     4     5
```

Nhập ma trận (4)

Kích thước

Để xác định kích thước của một ma trận ta dùng lệnh `length` (trả về kích thước lớn nhất) hay lệnh `size` (số hàng và cột). Ví dụ:

```
>> c = [1 2 3 4; 5 6 7 8];
>> length(c)
ans =
     4
>> [m, n] = size(c)
m =
     2
n =
     4
```

Nhập ma trận (5)

Các lệnh tính kích thước của ma trận được liệt kê dưới bảng sau:

`whos`	Hiển thị các biến trong không gian làm việc cùng kích cỡ tương ứng
`s = size(A)`	Trả về là vector hàng s, s(1)-số hàng và s(2)-số cột
`[r,c] = size(A)`	Trả về hai số `r,c` ứng với số hàng và số cột
`r = size(A,1)`	Trả về số hàng của `A`
`c = size(A,2)`	Trả về số cột của `A`
`n = length(A)`	Trả về `max(size(A))` khi `A` khác `[]`

Nhập ma trận (6)

Toán tử

Toán tử ":" là một toán tử rất quan trọng của MATLAB , nó xuất hiện ở nhiều dạng khác nhau.

Ví dụ

Biểu thức

```
>> 1:10
```

cho kết quả là một vector hàng chứa 10 số nguyên liên tiếp từ 1 đến 10

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

Nhập ma trận (7)

```
>> 100:-5:50
```

tạo một dãy số từ 100 đến 50, mỗi lần giảm 5

```
ans =
   100    95    90    85    80    75    70    65    60    55    50
```

```
>> 0: pi/4: pi
ans =
         0    0.7854    1.5708    2.3562    3.1416
```

Các biểu thức chỉ số tham chiếu đến một phần của ma trận. Viết `A(1:k,j)` là tham chiếu đến k phần tử đầu tiên của cột j của ma trận. Ngoài ra toán tử ":" tham chiếu tới **tất cả các phần tử** trong một hàng hay một cột.

```
>> A(:,3)
ans =
     2
    11
     7
    14
```

Nhập ma trận (8)

và

```
>> A(3,:)
ans =
     9     6     7    12
```

Viết `B = A(:, [1 3 2 4])` sẽ tạo ra ma trận B bằng cách đổi thứ tự các cột từ `[1 2 3 4]` thành `[1 3 2 4]`

```
>> B=A(:,[1 3 2 4])
B =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

Tạo các ma trận từ các hàm có sẵn(1)

MatLab cung cấp các hàm để tạo các ma trận cơ bản

`zeros`	Tất cả các phần tử bằng 0
`ones`	Tất cả các phần tử bằng 1
`rand`	Các phần tử có phân bố đều trên $[0, 1]$
`randn`	Các phần tử có phân bố chuẩn trên $[0, 1]$
`magic(n)`	Tạo ra ma trận cấp n gồm các số nguyên từ 1 đến n^2 với tổng các hàng bằng tổng các cột $n \geq 3$.
`pascal(n)`	Tạo ra ma trận xác định dương mà các phần tử lấy từ tam giác Pascal.
`eye(n)`	Tạo ma trận đơn vị cấp n

Tạo các ma trận từ các hàm có sẵn (2)

Sau đây là một số ví dụ:

```
>> Z=zeros(2,4)
Z =
     0     0     0     0
     0     0     0     0
>> F=5*ones(3)
F =
     5     5     5
     5     5     5
     5     5     5
>> R=randn(4)
R =
    0.5377    0.3188    3.5784    0.7254
    1.8339   -1.3077    2.7694   -0.0631
   -2.2588   -0.4336   -1.3499    0.7147
    0.8622    0.3426    3.0349   -0.2050
```

Hàm `load`

Hàm `load` đọc một file văn bản chứa các dữ liệu số. File văn bản phải được tổ chức như là một bảng chữ nhật của các số, cách nhau bởi các khoảng trắng ␣, mỗi hàng trên một dòng và số phần tử trên mỗi hàng là như nhau. Ví dụ, ta tạo file matrix.dat có nội dung sau:

```
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
```

Khi đó, lệnh

```
>> load matrix.dat
```

sẽ đọc file và tạo ra một biến matrix chứa các phần tử như trên.

M-files

Ta có thể tạo ra các ma trận bằng cách sử dụng các file văn bản chứa mã MatLab (M-files). Sử dụng trình soạn thảo Matlab Editor hoặc một trình soạn thảo bất kỳ tạo ra một file chứa các lệnh giống như dùng trong môi trường dòng lệnh MatLab , sau đó lưu file này dưới dạng ".m". Ví dụ, tạo ra một file chứa 5 dòng 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 file trên dưới tên `matrix.m`. Khi đó lệnh

```
>> matrix
```

sẽ đọc file và tạo ra một biến A có các phần tử như trên.

Ghép nối (Concatenation)

Ta có thể ghép nối các ma trận nhỏ để tạo thành các ma trận lớn hơn. Ví dụ

```
 >> A=ones(3)
A =
     1     1     1
     1     1     1
     1     1     1
>> B=[A A+3; A+4 A+6]
B =
     1     1     1     4     4     4
     1     1     1     4     4     4
     1     1     1     4     4     4
     5     5     5     7     7     7
     5     5     5     7     7     7
     5     5     5     7     7     7
```

Xóa hàng và cột

Ta có thể xóa hàng và cột của ma trận bằng cách gán cho chúng giá trị []. Ví dụ

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

Để xóa cột thứ 2 của X:

```
>> X(:,2)=[]
X =
     1     3
     4     6
     7     9
```

Một số lệnh xử lý ma trận (1)

Cộng	`X= A + B`
Trừ	`X= A - B`
Nhân	`X= A * B`
	`A.*B` nhân các phần tử tương ứng với nhau
Chia	`X = A/B`, khi đó `X*A = B`
	`X = A\B`, khi đó `A*X = B`
	`X=A./B` chia các phần tử tương ứng cho nhau.
Lũy thừa	`X = A^2`
	`X = A.^`: lũy thừa từng từ
Chuyển vị	`X=A'`
(liên hợp đối với ma trận phức)	
	`A.'`: chuyển vị (không liên hợp)
Nghịch đảo	`X = inv(A)`
Định thức	`d = det(A)`
Hệ đại số tuyến tính `Ax = b`	Nghiệm `x = A\b`

Một số lệnh xử lý ma trận (2) (Đọc thêm help)

Phân tích Cholesky	`R=chol(A)`
Phân tích LU	`[L,U]= lu(A)`
Phân tích QR	`[Q,R] = qr(A)`
Giá trị riêng, vector riêng	`eig(A), [d,r] = eig(A)`
Quay ma trận	`B = rot90(A)`
Đảo ma trận từ trái sang phải	`C=fliplr(A)`
Đảo ma trận từ trên xuống dưới	`D=flipud(A)`
Định dạng lại ma trận A với số hàng mới m và số cột mới n	`reshape(A,m,n)`
Lấy các phần tử trên đường chéo chính và lưu thành một vector	`diag(A)`
Chọn đường chéo tùy theo giá trị của k	`diag(A,k)` $k = 0$ chọn đường chéo chính $k > 0$ chọn đường chéo thứ k trên đường chéo chính $k < 0$ chọn đường chéo thứ k dưới dưới chéo chính

Một số lệnh xử lý ma trận (3)

Tạo ma trận có đường chéo chính là vector `v`	`A = diag(v)`
Tạo ma trận cùng cỡ với `a`, có các phần tử trên đường chéo chính và phía trên đường chéo chính, các phần tử khác bằng 0	`b = triu(a)`
Tạo ma trận cùng cỡ với `a`, có các phần tử trên đường chéo chính và phía dưới đường chéo chính, các phần tử khác bằng 0	`b = tril(a)`

Nội dung

Cấu trúc

- Là một cách tổ chức các dữ liệu liên quan
- Ví dụ, tạo một cấu trúc `s` với các trường `x,y` và name

```
>> s.y=1;
>> s.x=[1 1];
>> s.name='abc';
>> s
s =
       y: 1
       x: [1 1]
    name: 'abc'
```

hoặc đơn giản hơn với từ khóa `struct`:

```
>> s2=struct('y',1,'x',[1 1],'name','abc')
```

- Sử dụng các trường như là các biến bình thường

Cấu trúc

- Liệt kê danh sách các trường

```
f=fieldnames(s);
```

- Tham chiếu động tới các trường (dynamic field reference):

```
s.x;        % tham chiếu tĩnh (static field reference) tới s.x
s.('x')     % tham chiếu động tới s.x
```

- Vòng lặp trên các trường

```
f=fields(s); % tương đương với f=fieldnames(s)
for i=1:length(s)
    doSomething(s.(f{i}));
end
% hoặc
for f=fields(s) %
    doSomething(s.(char(f)));
end
% gọn nhất
structfun(@doSomething,s);
```

Cấu trúc

- Ta có thể tạo mảng của các cấu trúc, ví dụ

```
for i=1:10
    s(i).y=rand();
    s(i).x=[i:i+2];
    s(i).name=sprintf('name %d',i);
end
```

- Biến đổi mảng cấu trúc → mảng thông thường

```
for i=1:length(s)
    X(:,i)=s(i).x;
end
```

hoặc nhanh hơn

```
X=[s.x];
```

Nội dung

Mảng tế bào

- Mảng tế bào có thể chứa các kiểu dữ liệu bất kỳ

```
>> a=cell(3,2);
>> a{1,1}=1;
>> a{3,1}='hello';
>> a{2,2}=randn(100,100);
```

- Hữu dụng cho việc xử lý các xâu và tránh được việc dùng `squeeze()`
- Sử dụng mảng tế bào với các kiểu dữ liệu khác có thể gây rắc rối
  - chỉ số với dấu `()` cho ta các thành phần của mảng tế bào mà bản thân chúng là các tế bào
  - chỉ số với dấu `{}` chuyển các thành phần của mảng tế bào sang dạng dữ liệu cơ bản, trả về dạng danh sách cách nhau bởi dấu phẩy "," nếu có nhiều hơn một phần tử.

Mảng tế bào

Ví dụ

```
>> a={[1 2],3}
a =
    [1x2 double]    [3]
>> y=a{1}
y =
     1     2
>> ycell=a(1)
ycell =
    [1x2 double]
>> x=y+1
x =
     2     3
>> xcell=ycell+1
??? Undefined function or method 'plus' for input arguments of type
'cell'.
>> onetwothree=[a{1:2}]
onetwothree =
     1     2     3
```

Nội dung

Vẽ đồ thị 2-D

Lệnh cơ bản:

```
plot(x,f(x))
```

Trong đó, x là vector chứa miền xác định của hàm có biểu thức là $f(x)$.

Ví dụ 1

Vẽ đồ thị hàm số $y = \sin(x)$ *với* x *biến thiên trong khoảng* $[0, 2\pi]$*:*

```
x = 0:pi/100: 2*pi;
y = sin(x);
plot(x, y);
```

Vẽ đồ thị 2-D

Chú thích thêm cho đồ thị

`text(x, y, '...')`	Đặt chú thích lên đồ thị tại tọa độ (x,y)
`gtext('...')`	Đặt chú thích lên đồ thị, vị trí được xác định bởi click chuột
`title('...')`	Tiêu đề của đồ thị.
`legend('...','...',...)`	Thêm chú giải cho đồ thị.
`xlabel('...')`	Ghi nhãn cho trục Ox
`ylabel('...')`	Ghi nhãn cho trục Oy
`\bf`	Font in đậm
`\it`	Font in nghiêng
`\rm`	Font chữ thường
`hold on/off`	Bật/tắt chế độ cho phép vẽ nhiều đồ thị trong cùng một hệ trục tọa độ

Vẽ đồ thị 2-D

Tùy chỉnh nét vẽ, dấu và màu sắc

Lệnh tổng quát

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

Trong đó

- Các màu sắc: `'c'`-cyan, `'m'`-tím (magenta), `'y'`-vàng (yellow), `'r'`-đỏ (red), `'g'`-xanh lá cây (green), `'b'`-xanh nước biển (blue), `'w'`-trắng (white) và `'k'`-đen (black).
- Nét vẽ: `'-'`: nét liền, `'--'`: nét đứt, `':'`: chấm chấm, `'-.'`: gạch chấm.
- Dấu: `'+'`, `'o'`, `'*'` và `'x'`; `'s'`: □, 'd': ♦ , `'^'`: ▲, `'v'`: ▼, `'>'`: ▶, `'<'`: ◀, `'p'`: ★, `'h'`: *ngôi sao 6 cạnh*.

Vẽ đồ thị 2-D

Tùy chỉnh màu sắc và độ rộng của nét vẽ

`LineWidth`	Độ rộng của nét vẽ, tính bằng pt
`MarkerEdgecolor`	Màu sắc của đường viền dấu
`MarkerFacecolor`	Màu bên trong dấu
`Markersize`	Độ lớn của dấu, tính bằng pt

Ví dụ 2

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'-rs','LineWidth',2,'MarkerEdgecolor','k', ...
'MarkerFacecolor','g', 'Markersize',10)
```

Vẽ đồ thị 2-D

Xác định tọa độ, tùy chỉnh các kiểu tọa độ

```
axis([xmin xmax ymin ymax])
xlim([xmin xmax])
ylim([ymin ymax])
axis on/off/auto
axis normal/square/equal/tight
axis ij/xy
grid on/off
```

Vẽ nhiều đồ thị trong cùng một cửa sổ

```
>> subplot(m, n, p):
```

tạo ra một ma trận m hàng, n cột chứa $m \times n$ đồ thị , p là vị trí của từng đồ thị, thứ tự từ trên xuống dưới.

Vẽ đồ thị 3-D

Lệnh cơ bản

```
>> plot3(x, y, z)
```

Trong plot3, ta cần xác định các vector (x, y, z). Để vẽ mặt $z = f(x, y)$, sử dụng lệnh

```
>> meshgrid(x,y)
```

Ví dụ 3

```
>> t = 0:0.02*pi:25*pi;
>> x = sin(t); y = cos(t);
>> z = t;
>> plot3(x,y,z);
```

Ví dụ 4

Vẽ mặt $z(x, y) = x^2 y e^{-x^2-y^2}$ với $-4 \leq x \leq 4; \quad -4 \leq y \leq 4$

```
[x,y]=meshgrid([-4:0.1:4]);
z=x.*x.*y.*exp(-x.^ 2-y.^ 2);
plot3(x,y,z)
```

Vẽ đồ thị 3-D

Một số lệnh khác (đọc help!)

- `contour / contourf / contour3`
- `mesh / meshc / meshz`
- `surf / surfc`
- `waterfall`
- `bar3 / bar3h`
- `pie3 / fill3`
- `comet3 / scatter3 / stem3`

In và xuất đồ thị

- Dùng lệnh

```
>> print -dtiff -r200 mygraph.tiff print -deps2 mygraph.eps
```

- Sử dụng Plotting Tools