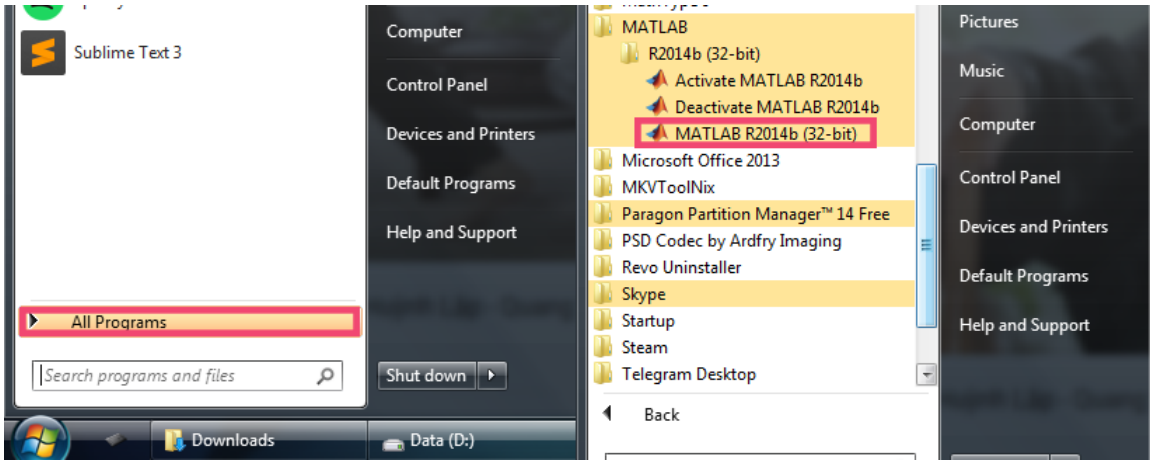
Mục lục

I. Cơ bản về MATLAB

1. Không gian làm việc

a. Giao diện chính

Có 3 cách khởi động chương trình Matlab từ máy tính chạy Windows:
- Khởi động từ biểu tượng ngoài màn hình.
- Mở trực tiếp tập tin có đuôi mở rộng là .m, ví dụ: Bai1.m hoặc Hamso.m
- Vào biểu tượng Start > All Program > MATLAB > Matlab 2014b.



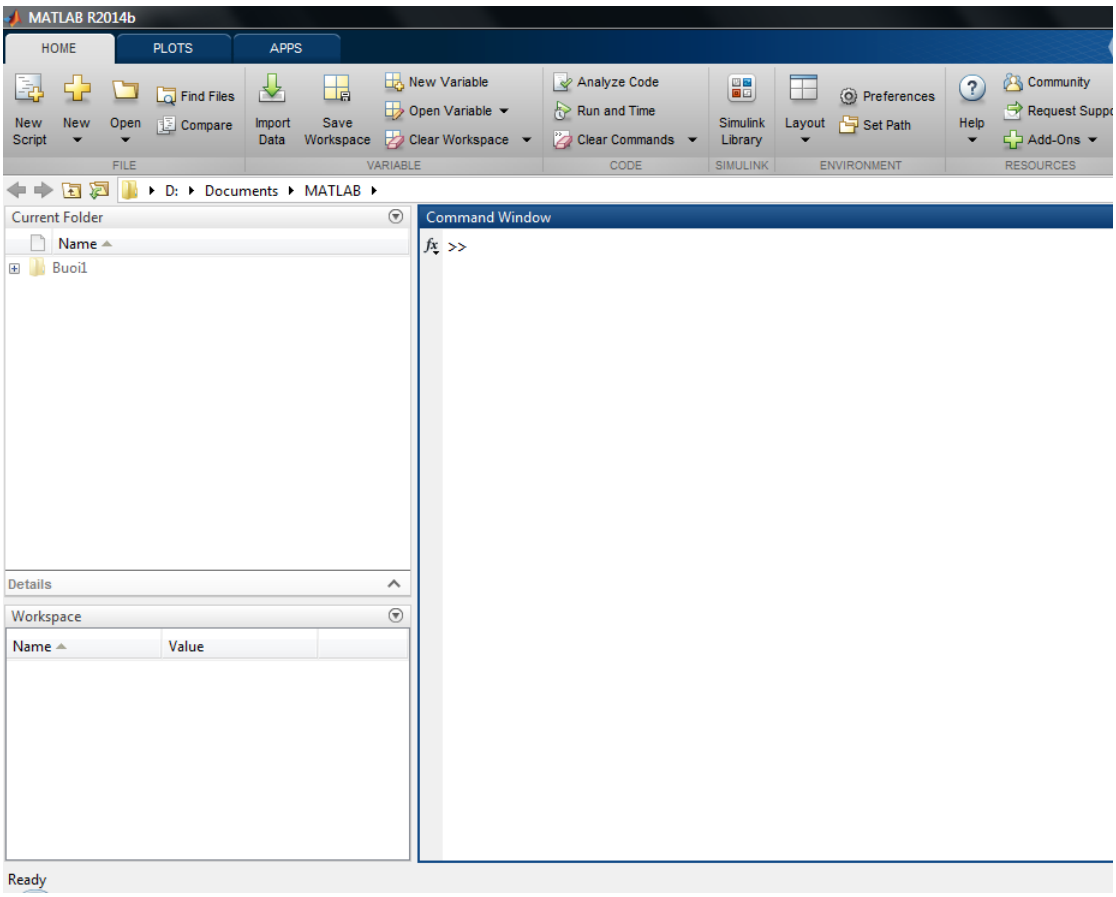
Hình 1: Thao tác mở chương trình từ Start Menu.

Giao diện chính của chương trình sau khi khởi động sẽ tương tự như hình bên dưới. Có 3 khu vực làm việc chính. Khung lớn nhất được bao viền màu xanh dương là Command Window. Khung Current Folder nằm ở phần trên, bên trái của Command Window. Phía dưới khung Current Folder là Khung Workspace.

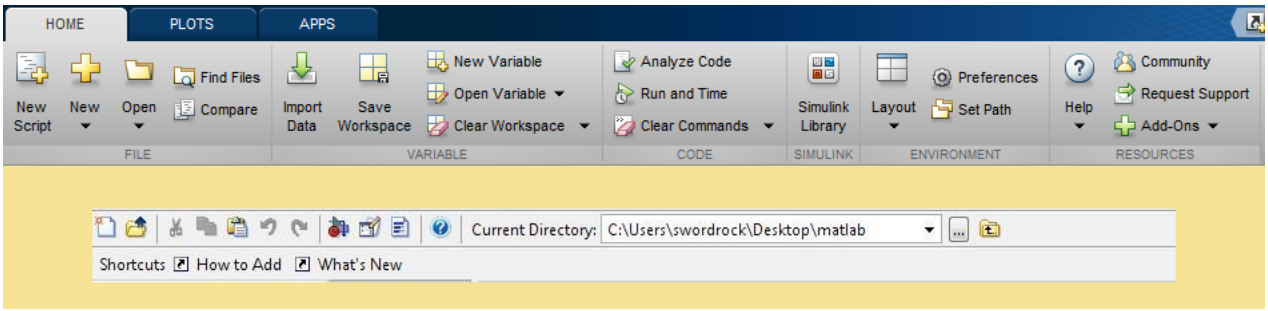
Trong đó:

- **Command Window:** Khu vực người dùng nhập các câu lệnh điều khiển và tính toán. Mỗi câu lệnh sẽ được thực hiện khi người dùng nhấn Enter.
- **Current Folder:** Khu vực truy cập các tập tin m-file, m-hàm... được lưu trữ trên ổ đĩa máy tính.
- **Workspace:** Liệt kê các biến và dữ liệu đang được xử lý tạm thời. Ngoài ra người dùng có thể thao tác thêm bớt chỉnh sửa các giá trị của biến và tham số trực tiếp trên Workspace mà không cần dùng đến câu lệnh.

Ngoài ba khu vực chính trên, kể từ phiên bản MATLAB 2012a, thanh công cụ phía trên cùng truyền thống đã được thay thế bằng các thẻ lệnh Ribbon với biểu tượng cụ thể và rõ ràng hơn.



Hình 2: Giao diện chính của chương trình



Hình 3: Thẻ Home (hình trên) ở giao diện mới và thanh công cụ truyền thống (hình dưới)

b. Các thao tác cơ bản

MATLAB thực hiện các phép cộng, trừ, nhân, chia như một chiếc máy tính bình thường. Xét các ví dụ đơn giản sau:

Ví dụ 1: Tính 3 + 2 + 6; 4 x 6 x 25 + 32 - 9 : 3.

```
>> 3 + 2 + 6 =
ans =
        12
>> 4 * 6 * 25 + 32 - 9 / 3 =
ans =
        629
```

Lưu ý: MATLAB không quan tâm đến khoảng trắng giữa các dấu và luôn ưu tiên phép nhân rồi mới đến phép cộng. Kí hiệu `ans` là viết tắt của từ "answer" có nghĩa là kết quả của phép tính.

Ví dụ 2: Tính $\sin\frac{\pi}{2} + \cos\frac{\pi}{2}$

```
>> sin(\pi/2)+cos(\pi/2)=
ans =
        1
```

2. Biến, hằng số, hàm cơ bản

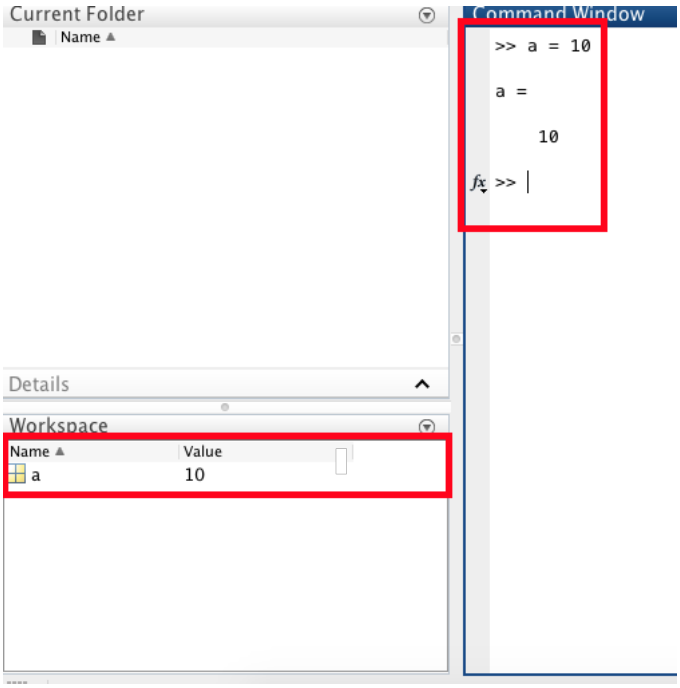
Trong MATLAB, ta có thể gián giá trị cho biến với tên gọi bất kỳ, điều này giúp cho việc tính toán rõ ràng vả dễ dàng hơn. Đặt biệt với những bài tính toán với các biến số, ta dễ dàng thay đổi giá trị của biến số để có được các kết quả phù hợp với yêu cầu tính toán.
Yêu cầu khi đặt tên biến: Tên biến phải bắt đầu bằng chữ cái, các ký tự sau ký tự đầu tiên có thể là số, chữ hoặc ký tự "_". Tên biến có độ dài tối đa là 31 ký tự, trong tên biến không được dùng dấu chấm câu. Ngoài ra, MATLAB phân biệt ký tự hoa sẽ khác với ký tự thường. Tức là biến "var1" sẽ khác với biến "Var1".
Một số biến đã được định nghĩa sẵn (một số còn được gọi là hằng số):

- Ký tự i và j được dùng cho đơn vị phức.
- Biến eps dùng để chỉ số nhỏ nhất, số này khi cộng với 1 ta được số nhỏ nhất lớn hơn 1.
- Biến pi dùng để chỉ số pi (π), với giá trị là 3.1415926...
- Biến ans dùng để lưu kết quả vừa tính toán trước đó.
- Biến Inf và -Inf (lưu ý ký tự I đứng đầu được viết hoa) dùng để biểu diễn dương và âm vô cực.
- Ký tự NaN thể hiện cho "Not a Number", không phải là một số.

Biến thông thường: (hay còn được gọi là biến vô hướng) là biến đucợ người dùng định nghĩa bằng cách gián một giá trị cụ thể nào đó trực tiếp thông qua câu lệnh trong Command Windows.
Ví dụ 3: Tạo biến tên a chứa giá trị là số 10. Ta sẽ nhập lệnh vào Command Windows như sau:

```
>> a = 10
a =
     10
```

Ngay lập tức biến tên a sẽ được liệt kê trong khu vực Workspace. Người dùng có thể truy cập hoặc điều chỉnh giá trị biến ngay trong cửa sổ Workspace.



Hình 4: Tạo biến ở khu vực Command Window và quản lý biến tại khu vực Workspace.

Ví dụ 4: Tạo biến c dựa vào biến a có sẵn biết rằng c được tính qua biểu thức $c = a^2 + a^3$. Ta tiếp tục nhập lệnh sau vào khu vực Command Window từ ví dụ trước đó:

```
>> c = a^2 + a^3
c =
     1100
```

Qua ví dụ 2 ta có thể tạo biến dựa vào giá trị của một biến khác cho trước. Ngoài ra, ta có thể ẩn đi kết quả tính toán của câu lệnh bằng cách thêm dấu ";" vào cuối câu lệnh. Ví dụ sau sẽ giúp ta dễ hình dung:

```
>> b = 6 + c;
>> b = 6 + c
b =
     1106
```

3. Quản lý tập tin, M-file

a. Tạo thư mục lưu trữ

Trong MATLAB, ta nên tập thói quen dùng các thư mục được sắp xếp gọn gàng để phân loại dữ liệu đang xử lý. Những thao tác cơ bản gồm:

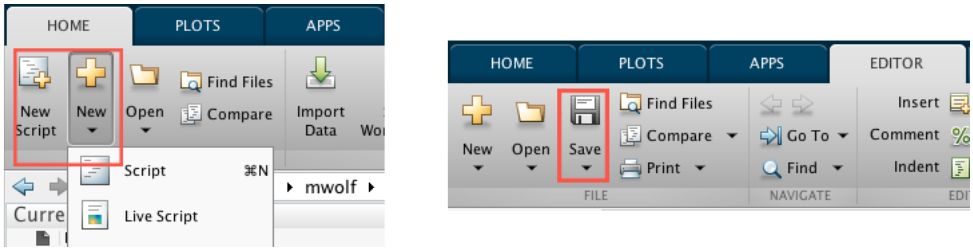
- Để tạo thư mục mới, chọn nút "Browse for folder" ở gần khu vực Current Folder. Sau đó ta tạo thư mục mới trong cửa sổ vừa hiện ra, đặt tên cho thư mục (lưu ý rằng tên thư mục không được phép chứa ký tự khoảng trắng).
- Chọn thư mục vừa tạo và nhấn nút "Open" để mở.
- Lúc này, thư mục hiện hành chính là thư mục bạn vừa mới tạo.
- Không chỉ cho phép người dùng tạo thư mục thông qua cửa sổ "Browse for folder", người dùng còn có thể xoá, sửa hay di chuyển thư mục trong cửa sổ đó.

Nâng cao: Với những người dùng thường xuyên lưu trữ chương trình hoặc các tập tin m-file ở các thư mục nằm nhiều nơi trên ổ đĩa, người dùng có thể thêm đường dẫn thư mục đó vào cửa sổ **Set Path (trong phần Enviroment)** để có thể chạy lệnh tính toán mà không cần quan tâm vị trí lưu trữ thư mục chứa các tập tin đó.

b. Scripts (Đoạn mã lệnh)

Scripts (còn được gọi là đoạn mã lệnh) là một tập tin chứa các câu lệnh chạy trong mội trường MATLAB. Các câu lệnh được lưu trữ trong tập tin này sẽ được thực thi theo trình tự.
Scripts được viết trong cửa sổ MATLAB Editor tích hợp sẵn trong MATLAB và được lưu trên ổ đĩa dưới dạng các tập tin có đuôi mở rộng là .m, dung lượng thường không quá 100MB. Ngoài ra, ta hoàn toàn có thể dùng một công cụ soạn thảo cơ bản như **Notepad** mặc định của Windows để tạo ra một tập tin đuôi .m nhưng yêu cầu các câu lệnh trong đó phải đúng và thực thi được trong môi trường MATLAB.
Các cách tạo một tập tin MATLAB (đuôi .m):

- Từ Command Windows gõ câu lệnh với cú pháp `edit tenfile.m`, trong đó ta thay thế `tenfile.m` bằng tên tập tin muốn tạo. Sau đó chọn OK trong cửa sổ xác nhận tạo tập tin tenfile.m.
- Chọn vào biểu tượng **New Script** trong khu vực **File** hoặc chọn File > New > Script đều được. Sau khi cửa sổ soạn thảo đã mở, ta chọn biểu tượng **Save** trong khu vực File và tiến hành đặt tên cho tập tin cùng với việc chọn khu vực lưu trữ.



Hình 5: Khu vực tạo Script mới (hình bên trái) và khu vực lưu Script (hình bên phải).

- Ngoài hai cách trên, ta còn có thể dùng tổ hợp phím ctrl + N (đối với Windows) hoặc ⌘ + N (đối với MacOS) để mở cửa số MATLAB Editor và tạo tập tin mới.

Lưu ý: Nội dung ban đầu của script thường là những câu lệnh, nhưng nếu trong script có yêu cầu input (nhập dữ liệu vào) và có lệnh output (xuất dữ liệu ra) thì nó sẽ trở thành một m-hàm.

c. Ghi chú (Comment)

Trong lúc soạn thảo các câu lệnh ở cửa sổ MATLAB Editor, người dùng có thể thêm những dòng chú thích vào dòng nào đó, qua đó giúp người dùng có thể đánh dấu, ghi nhớ hoặc tra cứu lại ý nghĩa của đoạn mã mà họ sử dụng. Các phần chú thích đó còn được gọi là những **Comment** và có hai cách thực hiện tạo một **Comment**:

- Đối với nội dung ghi chú ngắn, có thể chỉ 1 dòng thì chỉ cần thêm ký tự "%" vào trước câu ghi chú đó là được. Ví dụ: `%Day la cau ghi chu.` Thường những câu ghi chú sẽ được quy định chữ màu xanh lá để giúp người dùng dễ dàng phân biệt.
- Đối với những phần ghi chú có nhiều dòng, nội dung dài. Người dùng sẽ dùng cú pháp như dưới dây:

```
%{
Noi dung ghi chu, dong 1.
Noi dung ghi chu, dong 2.
...
Noi dung ghi chu, dong n.
%}
```

II. Các phép toán và thao tác với mảng

1. Các phép toán với mảng

a. Mảng đơn

Mảng đơn trong MATLAB có cấu trúc tương tự như các ngôn ngữ lập trình khác. Với mảng ta có thể lưu một dãy giá trị bất kỳ và thực hiện tính toán, sắp xếp các giá trị đó.
Ví dụ 5: Tạo một mảng số nguyên từ 1 đến 10 và đặt tên là mảng A.
Ta sẽ tạo một mảng A gồm các phần tử 1, 2, 3, ..., 10 bằng cú pháp sau:

```
>> A=[1,2,3,4,5,6,7,8,9,10]
A =
     1     2     3     4     5     6     7     8     9    10
```

Ngoài cách liệt kê như trên, ta có thể tạo mảng A bằng cú pháp ngắn gọn hơn. Chẳng hạn:

```
>> A=[1:10]
A =
     1     2     3     4     5     6     7     8     9    10
```

Với giá trị đầu và giá trị cuối, ta hoàn toàn có thể tạo một mảng chứa rất nhiều số mà không phải nhập vào một câu lệnh dài dòng. Mặt định MATLAB sẽ quy ước mỗi phần tử tiếp theo sẽ tăng lên một đơn vị nhưng ta có thể thay đổi giá trị tăng lên theo ý muốn. Ví dụ ta cho giá trị tăng lên giữa các đơn vị từ 1 đến 10 sẽ là 0.5 như sau:

```
>> A=[1:0.5:10]
A =
Columns 1 through 12
```

```
    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000    4.5000
       5.0000    5.5000    6.0000    6.5000
  Columns 13 through 19
    7.0000    7.5000    8.0000    8.5000    9.0000    9.5000   10.0000
```

Ví dụ 6: Tạo một mảng chứa các số nguyên lẻ từ 31 đến 51. Ta thực hiện như sau:

```
>> B=[31:2:51]
B =
   31    33    35    37    39    41    43    45    47    49    51
```

Tóm lại: Để tạo mảng, ta đặt các phần tử của mảng vào giữa hai dấu ngoặc vuông "[...]", giữa hai phần tử của mảng có thể là dấu cách hoặc dấu phẩy ",". Nếu mảng cần tạo có dạng chuỗi số liên tục, ta dùng dấu hai chấm ":" để ngăn cách phần tử đầu với phần tử cuối. Ta có thể thay đổi chênh lệch giữa các phần tử bằng cách chèn thêm độ chênh lệch vào giữa phần tử đầu và phần tử cuối, lưu ý luôn có dấu hai chấm ":" ngăn cách giữa các phần tử.

b. Địa chỉ của mảng

Các thành phần trong mảng luôn có số thứ tự, phần tử đầu tiên có số thứ tự là 1 và tăng dần cho đến phần tử cuối cùng trong mảng.
Để truy cập đến phần tử thứ `i` trong mảng `A` chẳng hạn, ta chỉ cần sử dụng cú pháp `A(i)` là MATLAB sẽ trả về giá trị phần tử đó trong mảng A ứng với thứ tự i.
Ví dụ 7: Cho mảng A gồm các số từ 1 đến 10. Thực hiện các truy cập sau:

- Khởi tạo mảng A.

```
>> A=[1:10]
A =
   1    2    3    4    5    6    7    8    9   10
```

- Xuất phần tử thứ 4 của mảng A.

```
>> A(4)
ans =
   4
```

- Xuất các phần tử từ thứ tự 2 đến 6 trong mảng A.

```
>> A(2:6)
ans =
   2    3    4    5    6
```

- Xuất các phần tử từ thứ tự 3 đến phần tử cuối cùng trong mảng A.

```
>> A(3:end)
ans =
   3    4    5    6    7    8    9   10
```

- Xuất các phần tử từ thứ tự 7 trở về trước trong mảng A.

```
>> A(7:-1:1)
```

```
ans =
   7     6     5     4     3     2     1
```

- Xuất các phần tử từ thứ tự 2 đến thứ tự 9, nhưng vị trí phần tử sau lớn hơn phần tử trước 2 đơn vị trong mảng A.

```
>> A(2:2:9)
ans =
   2     4     6     8
```

- Tạo mảng B mới gồm các phần tử thứ tự 3, 5, 9 của mảng A.

```
>> B=A([5,3,9])
B =
   5     3     9
```

c. Cấu trúc của mảng

Như phần trước, ngoài cách tạo một mảng tự động bằng phần tử đầu, phần tử cuối và độ chênh lệch, người dùng còn có thể tạo mảng bằng lệnh `linspace`. Cú pháp của hàm này như sau:

`linspace(giá trị phần tử đầu, giá trị phần tử cuối, số các phần tử)`

Điểm khác biệt của phương pháp này là ta không cần biết độ chênh lệch giữa các phần tử, chỉ với số lượng các phần tử cần có trong mảng là đủ.
Ví dụ 8: Tạo một mảng A có 10 phần tử có giá trị từ 0 đến π. Cú pháp như sau:

```
>> A=linspace(0,pi,10)
A =
        0    0.3491    0.6981    1.0472    1.3963    1.7453    2.0944    2.4435    2.7925
             3.1416
```

Ta có thể ghép nhiều mảng lại với nhau để thành một mảng lớn hơn. Cách làm này sẽ giúp tiết kiệm thời gian với những mảng có nhiều giá trị tuân theo nhiều quy luật khác nhau.
Ví dụ 9: Tạo mảng A chứa 5 phần tử đầu là các số lẻ từ 1 đến 10, 5 phần tử sau là các số chẵn từ 10 đến 20. Cú pháp như sau:

```
>> B=[1:2:10]
B =
   1     3     5     7     9
>> C=[10:2:20]
C =
   10    12    14    16    18    20
>> A=[B,C]
A =
   1     3     5     7     9    10    12    14    16    18    20
```

d. Vectơ hàng và vectơ cột

Ở phần trước, các giá trị trong mảng được xếp thành dãy số nằm theo một hàng ngang, do đó người ta còn gọi nó là một vectơ hàng. Ngoài các sắp xếp đó, MATLAB cũng cho phép người dùng tạo cấu trúc mảng nhưng các giá trị được xếp theo một cột thẳng đứng, còn được gọi là vectơ cột. Các thao tác tính toán đối với vectơ cột đều được áp dụng tương tự như với vectơ hàng.

Ví dụ 10: Tạo một vectơ cột a có giá trị sau a = $(1, 2, 3, 4)$. Cú pháp thực hiện:

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

Điểm khác biệt trong lúc tạo một vectơ cột là các phần tử ngăn cách nhau bởi dấu chấm phẩy ";". Ngoài ra ta hoàn toàn có thể dùng toán tử chuyển vị trong MATLAB (sử dụng dấu nháy đơn (') để thực hiện) sẽ cho ra kết quả tương tự. Ví dụ dưới đây sẽ minh hoạ rõ hơn:

Ví dụ 11: Tạo vectơ a có giá trị a = $(2, 4, 1, 3)$, tiếp tục tạo vectơ b là chuyển vị của vectơ a và tạo vectơ c là chuyển vị của vectơ b.

```
>> a=[2;4;1;3]
a =
   2
   4
   1
   3
>> b=a'
b =
   2    4    1    3
>> c=b'
c =
   2
   4
   1
   3
```

e. Mảng có phần tử là 0 hoặc 1

f. Thao tác đối với mảng

g. Tìm kiếm mảng con

h. So sánh mảng

i. Kích cỡ của mảng

j. Mảng nhiều chiều

2. Các thao tác với mảng

a. Tạo phương trình tuyến tính

b. Các hàm ma trận

c. Các ma trận đặc biệt

III. Vòng lặp điều khiển

1. Vòng lặp for

2. Vòng lặp while

3. Cấu trúc if-else-end

4. Cấu trúc switch-case

IV. Đồ hoạ trong hệ toạ độ phẳng và không gian ba chiều

1. Đồ hoạ trong hệ toạ độ phẳng

a. Sử dụng lệnh plot

b. Kiểu đường, dấu và màu

c. Kiểu đồ thị

d. Đồ thị lưới, hộp chức trục, nhãn và lời chú giải

e. Kiến tạo hệ trục toạ độ

f. In hình

g. Thao tác với đồ thị

"Một số kiến thức về phần mềm MATLAB"

Chương 1: Cơ bản về MATLAB

1.1 Không gian làm việc

1.2 Biến, câu giải thích, chấm câu

1.3 Các hằng số, các phép toán cơ bản, các hàm toán học.

1.4 Quản lý tập, M-file, M-hàm

(C1 → C4)

Chương 2: Các thao tác và phép toán với mảng

(C6 và C7)

Chương 3: Vòng lặp điều khiển (C11)

Chương 4: Đồ thị trong mặt phẳng và trong không gian

(Bổ sung nhiều ví dụ và bài tập)

```
1   // Hello.java
2   for (i:1);
3   import javax.swing.JApplet;
4   import java.awt.Graphics;
5
6   public class Hello extends JApplet {
7      public void paintComponent(Graphics g) {
8         g.drawString("Hello, world!", 65, 95);
9      }
10  }
```