

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

\*\*\*\*\*

**BÀI GIẢNG**

**MATLAB VÀ ỨNG DỤNG**

**Chủ biên:   Phạm Văn Sự  
                  Ngô Đức Thiện  
                  Vũ Anh Đào**

**HÀ NỘI 2014**

# LỜI NÓI ĐẦU

MATLAB là từ viết tắt của "Matrix Laboratory" với khẩu hiệu "The language of technical computing", tức là một phần mềm tính toán kỹ thuật dựa trên các phép tính của ma trận. Ban đầu MATLAB được thiết kế bởi Cleve Moler vào những năm 1970 để sử dụng như một công cụ dạy học. Từ đó đến nay nó đã được phát triển thành một bộ phần mềm thương mại rất thành công. Phiên bản mới nhất hiện nay của MATLAB là một bộ phần mềm cho công việc tính toán trong các ngành kỹ thuật, trong khoa học và trong lĩnh vực toán học ứng dụng. MATLAB cho ta một ngôn ngữ lập trình mạnh, giao diện đồ họa đẹp, và một phạm vi rất rộng các kiến thức chuyên môn. MATLAB là một thương hiệu đã được thương mại hóa của tập đoàn MathWorks, Massachusetts, USA (hiện là nhà cung cấp hàng đầu thế giới cho các phần mềm tính toán kỹ thuật và thiết kế dựa trên mô hình).

Cuốn bài giảng này được viết dựa trên đề cương chương trình môn học "MATLAB VÀ ỨNG DỤNG" dành cho sinh viên chuyên ngành kỹ thuật điện tử. Các nội dung trong bài giảng là các cơ sở kiến thức cơ bản nhất giúp sinh viên tiếp cận cách sử dụng phần mềm MATLAB, bao gồm: cấu trúc các kiểu dữ liệu, các câu lệnh, các hàm cơ bản, cách thức lập trình tạo hàm, tạo giao diện đồ họa...và cuối cùng là một số công cụ nâng cao và các ví dụ ứng dụng cụ thể.

Do đây là phiên bản đầu tiên nên cuốn bài giảng này chắc chắn còn nhiều thiếu sót. Nhóm tác giả rất mong có được các ý kiến phản biện, đóng góp của đồng nghiệp và bạn đọc.

*Hà Nội, tháng 7 năm 2014*

Nhóm tác giả

# MỤC LỤC

LỜI NÓI ĐẦU .....	1
MỤC LỤC.....	2
CHƯƠNG 1. GIỚI THIỆU VỀ MATLAB.....	5
1.1. GIỚI THIỆU CHUNG.....	5
1.2. CÁC LỆNH CƠ BẢN TRONG MATLAB.....	9
1.2.1. Biến .....	9
1.2.2. Các hàm toán học thông thường.....	12
1.2.3. Một số lệnh điều khiển cơ bản .....	16
1.3. LẬP TRÌNH TRONG MATLAB.....	17
1.3.1. Các lệnh về M-file:.....	17
1.3.2. Cấu trúc điều khiển/rẽ nhánh.....	18
1.3.3. Làm việc với file dữ liệu .....	21
1.3.4. Hàm m-File.....	24
1.3.5. Các phép tính logic.....	26
1.4. ĐỒ HỌA TRONG MATLAB .....	28
1.4.1. Đồ họa trong hệ tọa độ phẳng 2D.....	28
1.4.2. Đồ thị trong không gian ba chiều (3D).....	36
1.5. SIMULINK.....	39
BÀI TẬP CHƯƠNG 1.....	45
CHƯƠNG 2. CẤU TRÚC DỮ LIỆU .....	47
2.1. GIỚI THIỆU CHUNG.....	47
2.2. MA TRẬN, VÉCTƠ VÀ ĐA THỨC .....	49
2.2.1. Véctơ .....	49
2.2.2. Ma trận .....	51
2.2.3. Các phép toán trên véctơ và ma trận .....	52
2.2.4. Đa thức .....	55
2.2.5. Mảng.....	59
2.3. MỘT SỐ KIỂU DỮ LIỆU KHÁC .....	62
2.3.1. Số thực.....	62
2.3.2. Số phức.....	62
2.3.3. Chuỗi .....	64

CHƯƠNG 3. MỘT SỐ CÔNG CỤ NÂNG CAO.....	66
3.1. GIỚI THIỆU CHUNG.....	66
3.2. TOOLBOX .....	66
3.2.1. Signal Processing Toolbox.....	66
3.2.2. Control System Toolbox .....	70
3.2.3. Symbolic Math Toolbox.....	79
3.3. GIAO DIỆN NGƯỜI DÙNG GUI .....	103
3.3.1. Giới thiệu về GUI.....	103
3.3.2. Cấu trúc và các đối tượng đồ họa .....	104
3.3.3. Tạo các uicontrols .....	108
3.3.4. Điều khiển với uicontrols .....	112
3.3.5. Tạo menu .....	120
CHƯƠNG 4. ỨNG DỤNG CỦA MATLAB.....	127
4.1. GIỚI THIỆU .....	127
4.2. ỨNG DỤNG MATLAB TRONG TOÁN HỌC.....	127
4.2.1. Giới thiệu chung.....	127
4.2.2. Một số bài toán toán học giải bằng MATLAB.....	127
4.3. ỨNG DỤNG MATLAB TRONG XỬ LÝ TÍN HIỆU SỐ.....	137
4.3.1. Giới thiệu chung.....	137
4.3.2. Một số bài toán xử lý tín hiệu số giải bằng MATLAB .....	137
4.4. ỨNG DỤNG MATLAB TRONG LÝ THUYẾT ĐIỀU KHIỂN TỰ ĐỘNG...	144
4.4.1. Giới thiệu chung.....	144
4.4.2. Một số bài toán điều khiển giải bằng MATLAB.....	144
4.5. ỨNG DỤNG MATLAB TRONG TRUYỀN THÔNG .....	154
4.5.1. Giới thiệu chung.....	154
4.5.2. Một số bài toán truyền thông giải bằng MATLAB .....	155
4.6. ỨNG DỤNG MATLAB TRONG KỸ THUẬT ĐIỆN TỬ .....	163
4.6.1. Giới thiệu chung.....	163
4.6.2. Một số bài toán Kỹ thuật điện tử giải bằng MATLAB .....	163
4.7. BÀI TẬP CHƯƠNG 4.....	167
4.7.1. Bài tập: Ứng dụng MATLAB trong giải toán .....	167
4.7.2. Bài tập: Ứng dụng MATLAB trong Xử lý tín hiệu số .....	168

4.7.3. Bài tập: Ứng dụng MATLAB trong Lý thuyết điều khiển.....	169
4.7.4. Bài tập: Ứng dụng MATLAB trong Truyền thông .....	171
4.7.5. Bài tập: Ứng dụng MATLAB trong Kỹ thuật điện tử.....	172
PHỤ LỤC.....	175
TÀI LIỆU THAM KHẢO.....	192

PDF

# CHƯƠNG 1. GIỚI THIỆU VỀ MATLAB

## 1.1. GIỚI THIỆU CHUNG

Hệ thống MATLAB (Matrix Laboratory) gồm có 5 phần chính:

- *Ngôn ngữ MATLAB*: Là một ngôn ngữ ma trận, mảng cấp cao với các câu lệnh, hàm, cấu trúc dữ liệu vào / ra, các tính năng lập trình hướng đối tượng. Nó cho phép lập trình các ứng dụng từ nhỏ đến lớn, từ đơn giản đến phức tạp. MATLAB là một ngôn ngữ bậc cao và môi trường tương tác cho phép bạn tiến hành các nhiệm vụ tính toán có cường độ lớn nhanh hơn với các ngôn ngữ lập trình như C, C++ và Fortran.

- *Môi trường làm việc của MATLAB*: Đây là một bộ các công cụ và phương tiện mà bạn sử dụng với tư cách là người dùng hoặc người lập trình MATLAB. Nó bao gồm các phương tiện cho việc quản lý các biến trong không gian làm việc Workspace cũng như xuất nhập khẩu dữ liệu. Nó cũng bao gồm các công cụ để phát triển, quản lý, gỡ rối và định hình M-file, ứng dụng của MATLAB.

- *Xử lý đồ họa*: Đây là một hệ thống đồ họa của MATLAB. Nó bao gồm các lệnh cao cấp cho trực quan hóa dữ liệu hai chiều và ba chiều, xử lý ảnh, ảnh động,... Nó cũng cung cấp các lệnh cấp thấp cho phép bạn tùy biến giao diện đồ họa cũng như đi xây dựng một giao diện đồ họa hoàn chỉnh cho ứng dụng MATLAB của mình.

- *Thư viện toán học MATLAB*: Đây là một tập hợp khổng lồ các thuật toán tính toán từ các hàm cơ bản như cộng, trừ, nhân, chia, sin, cos, tan, số học phức... tới các hàm phức tạp hơn như nghịch đảo, ma trận, tìm trị riêng của ma trận, phép biến đổi Fourier nhanh...

- *Giao diện chương trình ứng dụng MATLAB API* (Application Program Interface): Đây là một thư viện cho phép bạn viết các chương trình C và Fortran tương thích với MATLAB.

Simulink là một chương trình đi kèm với MATLAB, là một hệ thống tương tác với việc mô phỏng các hệ thống động học phi tuyến. Nó là một chương trình đồ họa sử dụng chuột để thao tác, cho phép mô hình hóa một hệ thống bằng cách vẽ một sơ đồ khối trên màn hình. Nó có thể làm việc với các hệ thống tuyến tính, phi tuyến, hệ thống liên tục theo thời gian, hệ thống gián đoạn theo thời gian, hệ thống đa biến...

Đối với các hệ thống phức tạp, phi tuyến, ngẫu nhiên, các tham số biến đổi theo thời gian, phương pháp giải tích truyền thống không thể cho ta lời giải chính xác được. Lúc này, phương pháp mô hình hóa và mô phỏng phát huy thế mạnh của mình và trong nhiều trường hợp nó là giải pháp duy nhất để nghiên cứu các hệ thống phức tạp.

Khả năng và những ứng dụng của MATLAB:

- Một trong những tính năng tuyệt vời nhất của MATLAB nhìn từ góc độ những nhà khoa học tính toán là thư viện dựng sẵn to lớn rất phong phú các chương trình tính toán và các công cụ hiển thị đồ họa.
- MATLAB cho phép người dùng tiến hành rất nhiều các nhiệm vụ thông thường liên quan tới việc giải quyết các vấn đề một cách số học. Nó cho phép chúng ta dành nhiều thời gian hơn cho việc suy nghĩ, khuyến khích chúng ta thí nghiệm.

- MATLAB ứng dụng những thuật toán rất tin cậy, vì vậy chúng ta có thể tin tưởng vào kết quả thu được.
- Các tính toán rất mạnh có thể được thực hiện chỉ với một hoặc hai câu lệnh.
- Có thể xây dựng riêng cho mình những hàm toán học cho những ứng dụng đặc biệt.
- MATLAB cung cấp giao diện đồ họa tuyệt hảo, các hình từ MATKAB có thể đem chèn vào LATEX và các tài liệu Word.



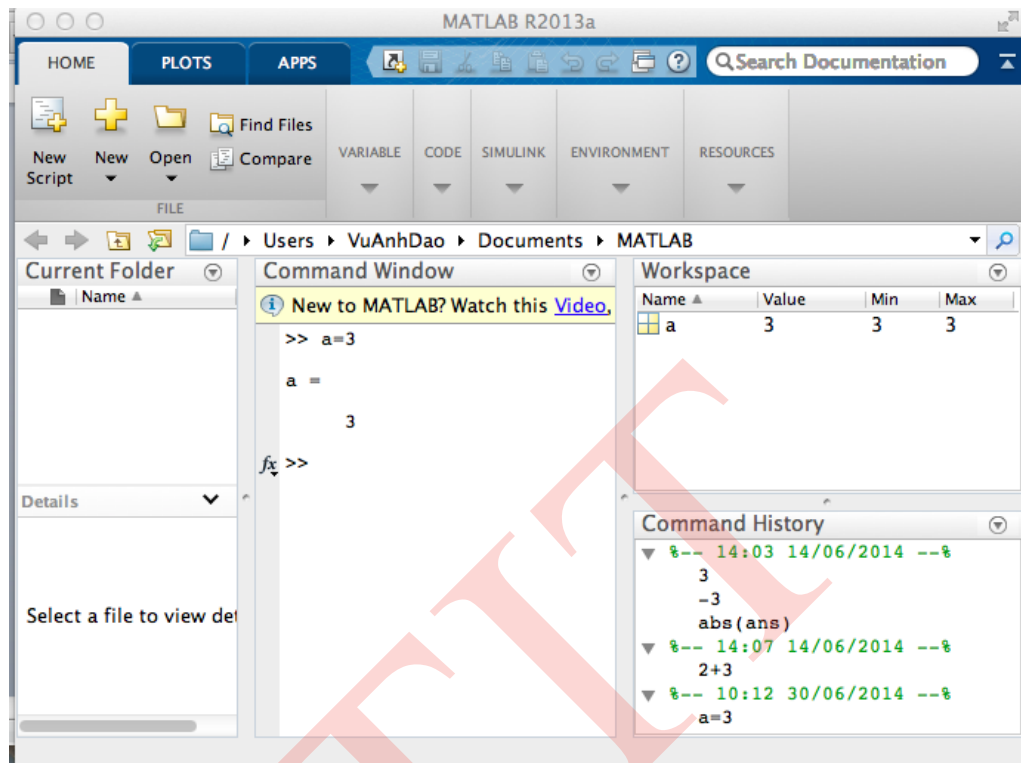
**Hình 1.1. Thông tin về phiên bản mới nhất của MATLAB**

Mỗi năm, công ty MathWorks cho update hai phiên bản. Phiên bản MATLAB như Hình 1.1 là phiên bản thứ hai được giới thiệu trong năm 2013.

- Về cơ bản, không gian làm việc của MATLAB gồm có các phần sau:

- + Cửa sổ lệnh (Command window)
- + Cửa sổ không gian làm việc (Workspace window)
- + Cửa sổ lịch sử lệnh (Command History window - lịch sử)
- + Cửa sổ biên tập mảng, vectơ, ma trận (Array editor window)
- + Cửa sổ địa chỉ thư mục hiện thời (Current directory window)
- + Cửa sổ trợ giúp (Help window)

\* **Cửa sổ lệnh:** Các diễn giải và câu của MATLAB được đánh giá khi bạn gõ vào 'Cửa sổ lệnh', và các kết quả tính toán cũng được thể hiện tại đây. Không giống như Fortran và các ngôn ngữ tính toán cần biên dịch khác, MATLAB là một môi trường tương tác, có nghĩa là khi đánh một câu lệnh thì MATLAB cố gắng thực thi nó ngay lập tức trước khi đòi hỏi một lệnh tiếp theo.



Hình 1.2. Giao diện của MATLAB

Cửa sổ lệnh của MATLAB có dấu nhắc (>>), tại đây có thể gõ các lệnh của MATLAB hoặc các biến

\* **Cửa sổ không gian làm việc (workspace):** Các biến và dữ liệu mà bạn nhập vào được MATLAB lưu trong một phần gọi là 'không gian làm việc'. Tất cả các biến, ngoại trừ những biến cục bộ thuộc về một M-file, sẽ được thể hiện trong không gian làm việc.

\* **Cửa sổ biên tập mảng (ma trận nói chung):** Khi đã có một mảng, ta có thể chỉnh sửa, biên tập lại nó bằng Array Editor. Công cụ này làm việc như một bảng tính (spreadsheet) cho ma trận.

\* **Cửa sổ địa chỉ thư mục hiện thời:** Thư mục hiện thời là nơi chương trình MATLAB sẽ tìm các M-file, và các file không gian làm việc (.mat files) mà bạn đã Load và Save.

\* **Cửa sổ trợ giúp:** Trợ giúp và các thông tin về MATLAB có thể được tìm thấy theo nhiều cách:

- + Từ dòng lệnh, bằng cách đánh help command.
- + Từ cửa sổ Help riêng biệt xuất phát ở Menu Help
- + Từ Helpdesk của MATLAB được lưu trữ trên đĩa hoặc CD-rom



+ Từ mạng Internet

Từ dòng lệnh, đơn giản nhất hãy đánh lệnh help và nhấn Enter

**Bảng 1.1. Các chủ đề trợ giúp**

Tên	Kết quả
matlab/general	Các lệnh với mục đích tổng quát.
matlab/ops	(operators) Các toán tử và các ký tự đặc biệt
matlab/lang	(language) Ngôn ngữ lập trình
matlab/elmat	(elementary) Ma trận căn bản
matlab/elfun	(elementary functions) Các hàm toán căn bản
matlab/specfun	(specialized functions) Các hàm toán đặc biệt

\* **Cửa sổ lịch sử lệnh:** Trong giao diện mặc định của MATLAB, cửa sổ “Command History” nằm ở góc phần tư bên dưới, phía trái. Các lệnh đã sử dụng trong các lần khởi động MATLAB gần đây đều được lưu lại. Mỗi lần khởi động MATLAB, toàn bộ các lệnh sử dụng trong lần đó sẽ được lưu lại thành một nhóm và có thể đóng/mở để xem.

\* **Ghi và phục hồi dữ liệu**

- Để nhớ các biến, MATLAB có thể ghi và gọi lại dữ liệu từ file trong máy tính. Mục Save Workplace as... trong bảng chọn File sẽ mở hộp hội thoại để ghi tất cả các biến hiện tại.

- Tương tự, mục Load Workplace trong bảng chọn File sẽ mở hộp hội thoại để gọi lại tất cả các biến mà ta đã ghi lại từ không gian làm việc trước.

Ghi chú: việc Load không làm mất các biến hiện có trong không gian làm việc hiện tại. Khi ta gọi lại các biến trùng tên với các biến trong không gian làm việc của MATLAB, nó sẽ thay đổi giá trị của các biến theo giá trị của các biến gọi ra từ file.

- Ngoài các bảng chọn, MATLAB còn cung cấp hai lệnh Save và Load, nó thực hiện một cách mềm dẻo hơn. Lệnh save cho phép ghi một hoặc nhiều hơn một biến tùy theo sự lựa chọn.

Ví dụ:

- Ghi dữ liệu: Để ghi các dữ liệu (các biến) vào tệp kiểu nhị phân với tên do ta đặt và đuôi tệp là MAT, ta có thể dùng lệnh sau:

```
>> Save Tên_tệp
```

hoặc chọn các mục trên thanh Menu: File/Save Workspace as

Ví dụ:

**save:** lưu tất cả các biến trong MATLAB theo kiểu nhị phân trong file matlab.mat

**save** dulieu: lưu tất cả các biến trong MATLAB theo kiểu nhị phân trong file matlab.mat

**save dulieuABCD – ASCII:** lưu các biến A, B, C, D theo dạng mã ASCII trong file dulieu.mat

```
>> save luu1
```

Các biến trong MATLAB được lưu trong tệp luu1.mat. Có thể ghi một số biến vào tệp theo lệnh sau:

```
>> save tên_tệp biến1 biến2...
```

Ví dụ:

```
>>save luu2 a b c
```

Các biến a,b,c được ghi trong tệp luu2.mat

- Phục hồi dữ liệu: Để phục hồi dữ liệu (các biến) trong các tệp đã ghi ta dùng lệnh:

```
>> Load tên_tệp
```

Ví dụ:

```
>>Load luu1
```

Lập trình theo nghĩa thông thường, là nhập vào máy những câu lệnh rõ ràng, theo một thứ tự nhất định sao cho khi máy thực hiện theo đúng thứ tự đó thì sẽ cho ta kết quả mong muốn. Khái niệm tương tự như vậy thường thấy trong các khóa học lập trình ngôn ngữ C, Pascal...

Khi khởi đầu với MatLab ta hãy hiểu theo nghĩa rộng hơn: lập trình còn có các bước biểu diễn bài toán dưới dạng các hàm và máy tính qua việc thực hiện các hàm này cho ta kết quả. Phương pháp này có mức độ trừu tượng cao hơn so với các câu lệnh chỉ dẫn đơn thuần.

## 1.2. CÁC LỆNH CƠ BẢN TRONG MATLAB

### 1.2.1. Biến

Những lệnh hoặc biến được lưu trong không gian làm việc của MATLAB và có thể được gọi lại khi cần. Dùng các mũi tên ( $\downarrow \uparrow$ ) để chọn các lệnh, có thể cắt, copy, dán và sửa chữa dòng lệnh. Ví dụ:

```
>> X=2
```

```
X =
```

```
2
```

```
>> A='XIN CHAO '
```

```
A =
```

```
XIN CHAO
```

Nếu không nhớ tên biến, ta có thể yêu cầu MATLAB cho danh sách các biến bằng cách đánh lệnh `who` từ dấu nhắc lệnh.

```
>> who
```

Your variables are:

A        B        a        ans        check    x        y        z

Lệnh `whos` sẽ cho biết chi tiết hơn các biến hiện có, giống như trong workspace.

Biến là một dãy kí tự được bắt đầu bằng chữ cái, có độ dài tối đa 31 kí tự, bao gồm các chữ cái, chữ số và dấu gạch dưới ( `_` ), có phân biệt chữ hoa và chữ thường.

Ví dụ: `x` ; `a12` ; `b_a`

Có thể gán giá trị cho tên biến bằng cách viết :

Tên\_biến = biểu thức

Ví dụ:

```
>> x=20;
```

```
>> a12=4;
```

```
>> b_a='ABCD';
```

**Bảng 1.2. Các biến đặc biệt trong MATLAB**

Các biến đặc biệt	Giá trị
ans	Tên biến mặc định dùng để trả về kết quả
pi	$\pi = 3.1415$
eps	Số dương nhỏ nhất, nếu cộng thêm 1 sẽ được số nhỏ nhất lớn hơn 1
inf	Để chỉ số vô cùng như kết quả của $1/0$
NAN hoặc nan	Dùng để chỉ số không xác định như kết quả của $0/0$
i hoặc j	Đơn vị ảo: $i = j = \sqrt{-1}$
nargin	Số các đối số đưa vào hàm được sử dụng
narout	Số các đối số hàm đưa ra
realmin	Số nhỏ nhất có thể được của số thực
realmax	Số lớn nhất có thể được của số thực

Các biến đặc biệt ở trên có sẵn giá trị, nếu ta thay đổi giá trị của nó thì giá trị ban đầu sẽ mất cho đến khi ta khởi động lại MATLAB thì nó mới trở lại giá trị ban đầu. Không nên thay đổi giá trị của các biến đặc biệt.

Ví dụ:

```
>> i
```

```
ans =
```

```

0.0000 + 1.0000i
>> j
ans =
0.0000 + 1.0000i
>> i*i
ans =
-1
>> pi
ans =
3.1416
>> eps
ans =
2.2204e-16
>> realmin
ans =
2.2251e-308
>> realmax
ans =
1.7977e+308

```

*\* Xoá các biến trong không gian làm việc:*

- Xoá tất cả các biến trong không gian làm việc: **clear**
- Xoá một biến: **clear tên\_biến**
- Xoá nhiều biến: **clear tên\_biến1 tên\_biến2**

```
>> clear a b c
```

- Dùng lệnh trên tất cả các biến bị xoá không khôi phục được, do vậy ta phải thận trọng khi dùng nó.

*\* Câu giải thích và ngắt dòng:* được viết sau dấu %

```
>> a=100 % a nhận giá trị 100
```

- Có thể viết nhiều lệnh trên một dòng, chúng được ngăn cách bởi dấu phẩy hoặc dấu chấm phẩy. Dấu phẩy là yêu cầu hiển thị kết quả trên màn hình, còn dấu chấm phẩy là không hiển thị kết quả trên màn hình.

Ví dụ:

```

>> A=2, B='abcde', x=456.32; y='mnopq'
A =
2
B =
abcde
y =

```

mnopq

- Dùng dấu ba chấm (...) viết sau phép toán để chỉ câu lệnh được tiếp tục ở hàng dưới. Không dùng dấu ba chấm cho các trường hợp khác, hay cho câu giải thích.

```
>>x=10,y=20
```

```
x =
```

```
10
```

```
y =
```

```
20
```

```
>>z=x+...
```

```
y
```

```
z =
```

```
30
```

Sau mỗi câu lệnh, nếu không muốn hiển thị kết quả của câu lệnh đó thì đánh dấu **chấm phẩy (;)**.

Ví dụ:

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

```
A =
```

```
1      2      3
```

```
4      5      6
```

```
2      5      7
```

```
>> B=[1 4;2 -4];
```

### 1.2.2. Các hàm toán học thông thường

Bảng 1.3. Các hàm lượng giác

Tên hàm	Chức năng
acos(x)	Hàm ngược của cosin
acosh(x)	Hàm ngược của hyperbol cosin
acot(x)	Nghịch đảo của hàm cotang
acoth(x)	Nghịch đảo của hàm hyperbol cotang
asin(x)	Hàm ngược của sine
asinh(x)	Hàm ngược của hyperbol sine
atan(x)	Hàm ngược của tangent
atan2(x,y)	Hàm ngược của tangent của phần thực của x và y
atanh(x)	Hàm ngược của hyperbol tangent
cos(x)	Hàm cosin của
cosh(x)	Hàm hyperbol cosin của x
cot(x)	Hàm cotang

$\coth(x)$	Hàm hyperbol cotang của x
max	Tìm giá trị lớn nhất
mean	Tìm giá trị trung bình
min	Tìm giá trị nhỏ nhất
$\sin(x)$	Hàm tính sine của x
$\sinh(x)$	Hàm tính hyperbol sine của x
Std	Tìm độ lệch quân phương
sum	Tính tổng
$\tan(x)$	Hàm tính tangent
$\tanh(x)$	Hàm tính hyperbol tangent

**Bảng 1.4. Các hàm lũy thừa**

Tên hàm	Ý nghĩa
$\exp(x)$	Hàm mũ
$\expm1(x)$	Tính $\exp(x)-1$
$\log(x)$	Hàm tính logarithm cơ số tự nhiên
$\log1p(x)$	Tính $\log(1+x)$
$\log10(x)$	Hàm tính logarithm cơ số 10
nthroot	Nghiệm thực bậc n của các số thực
reallog(x)	loga cơ số tự nhiên của số thực x
realsqrt(x)	Căn bậc hai của số $\geq 0$
$\sqrt{x}$	Tính căn bậc hai của x
$\gcd(x,y)$	Ước số chung lớn nhất của hai số nguyên x và y
$\text{lcm}(x,y)$	Bội số chung nhỏ nhất của hai số nguyên x và y

**Bảng 1.5. Các hàm liên quan đến số phức**

Tên hàm	Ý nghĩa
$\text{abs}(x)$	Tính giá trị tuyệt đối của x
$\text{angle}(x)$	Tính góc pha của số phức x
$\text{complex}(x)$	Tạo số phức x từ phần thực và phần ảo
$\text{conj}(x)$	Tìm số liên hợp phức của số phức x
$\text{imag}(x)$	Tìm phần ảo của số phức x
$\text{real}(x)$	Tìm thực của số phức x
$\text{isreal}(x)$	Hàm logic, trả về giá trị 'true' với mảng số thực
cplxpair	sắp xếp các số về các cặp liên hợp phức

**Bảng 1.6. Các hàm làm tròn và phần dư**

Tên hàm	Ý nghĩa
ceil(x)	Hàm làm tròn lên của số thực x
fix(x)	Hàm làm tròn giá trị của x về số nguyên gần nhất theo hướng số 0
floor(x)	Hàm làm tròn xuống của số thực x
mod	Lấy phần dư của phép chia
rem(x,y)	Phần dư của phép chia x/y
round(x)	Hàm làm tròn đến số nguyên gần nhất
sign(x)	Hàm dấu: trả về giá trị 1 nếu x dương và -1 nếu x âm

Các phép tính trên có mức độ ưu tiên như sau:

+ Các phép tính trong ngoặc đơn

+ Luỹ thừa ( $1+2^2 \rightarrow 1+4=5$ )

+ Phép nhân/chia (làm việc từ trái qua phải) ( $2*3/4=6/4=1.5$ )

+ Phép cộng/trừ (làm việc từ trái qua phải) ( $2-4+5=-2+5=3$ )

```
>> x=2+10/5+4^2-6*2
```

```
x =
```

```
8
```

```
>> x=abs(-3)/2
```

```
x =
```

```
1.5000
```

```
>> x=sqrt(6)/2
```

```
x =
```

```
1.2247
```

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

```
y =
```

```
0.9407
```

```
>> 2*atan(1)
```

```
ans =
```

```
1.5708
```

```
>> y=rem(13,3)
```

```
y =
```

```
1
```

```
>> gcd(12,26)
```

```
ans =
```

```
2
```

```
>> gcd(12,32)
```

```

ans =
     4
>> lcm(6,14)
ans =
    42
>> mod([1 2 3 5 7 8],3)
ans =
     1     2     0     2     1     2
>> x=[1 2 4 6 7 2 6 8];
>> sort(x)
ans =
     1     2     2     4     6     6     7     8

```

**Bảng 1.7. Các toán tử trong MATLAB**

Toán tử	Mô tả
+	Cộng
-	Trừ
*	Nhân đại số tuyến tính
.*	Nhân mảng cùng kích thước (nhân từng phần tử tương ứng với nhau)
/	Chia phải ( $B/A \sim B \cdot \text{inv}(A)$ , chính xác hơn $B/A = (A' \backslash B')$ )
\	Chia trái ( $A \backslash B \sim \text{inv}(A) * B$ )
./	Chia phải từng phần tử
.\	Chia trái từng phần tử
:	Toán tử hai chấm
.^	Luỹ thừa từng phần tử
'	Chuyển vị

Ví dụ:

```

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

```



```

1      2      4
2      5     -1
3      5      7
>> A+B
ans =
2      4      7
6     10      5
10    13     16
>> A.*B
ans =
1      4     12
8     25     -6
21    40     63
>> A*B
ans =
14     27     23
32     63     53
50     99     83

```

### 1.2.3. Một số lệnh điều khiển cơ bản

<code>clc</code>	Lệnh xóa cửa sổ lệnh (command window)
<code>pause</code>	Chờ sự đáp ứng từ phía người dùng
<code>=</code>	lệnh gán
<code>%</code>	câu lệnh sau dấu này được xem là dòng chú thích
<code>input</code>	lệnh lấy vào một giá trị.
<code>help</code>	lệnh yêu cầu sự giúp đỡ từ MATLAB
<code>save</code>	Lưu biến vào bộ nhớ
<code>load</code>	load biến từ file hay bộ nhớ
<code>break</code>	Thoát đột ngột khỏi vòng lặp WHILE hay FOR.
<code>continue</code>	Bỏ qua các lệnh hiện tại, tiếp tục thực hiện vòng lặp ở lần lặp tiếp theo.
<code>return</code>	lệnh trả về
<code>clf</code>	xóa hình hiện tại
<code>plot(signal)</code>	vẽ dạng sóng tín hiệu signal
<code>stairs(signal)</code>	vẽ tín hiệu signal theo dạng cầu thang.
<code>stem(signal)</code>	vẽ chuỗi dữ liệu rời rạc
<code>bar(signal)</code>	vẽ dữ liệu theo dạng cột

`mesh(A)`                    hiển thị đồ họa dạng 3D các giá trị ma trận

**\* Biểu diễn tín hiệu trên miền thời gian**

`n= [1:3]`    % Miền thời gian 1, 2, 3

`x=[1 2 3]`                % Tín hiệu rời rạc

`stem(n,x)`                % Biểu diễn tín hiệu x trên miền thời gian n

### 1.3. LẬP TRÌNH TRONG MATLAB

Trong MATLAB ta gõ lệnh vào từ cửa sổ lệnh, các lệnh sẽ được thực hiện ngay, nếu muốn thực hiện lại các lệnh ta lại phải gõ lại, như vậy không tiện, đặc biệt khi giải quyết các bài toán có mục tiêu cụ thể. Hơn nữa, các lệnh thực hiện trên cửa sổ lệnh sẽ biến mất khi đóng phần mềm hoặc bị mất điện đột ngột.

Để thuận lợi cho việc gõ các lệnh và lưu trữ lại các lệnh MATLAB cho phép mở file dạng văn bản để ghi các lệnh, file này gọi là Script file hay M\_file, phần mở rộng (đuôi file) là \*.m.

#### 1.3.1. Các lệnh về M-file:

- Mở tệp m-file mới: chọn các mục trong menu File/new/M-file.
- Mở tệp m-file cũ: chọn các mục trong menu File/Open sau đó chọn tên tệp m-file cần mở.
- Ghi tệp m-file: chọn các mục trong menu File/save sau đó gõ vào tên tệp m-file.
- Để chạy các lệnh của M-file, trên cửa sổ của script file ta chọn các mục Debug/Run, nhập dữ liệu từ cửa sổ lệnh (Command windows).

- MATLAB cung cấp một số hàm sử dụng trong m-file:

**Display(tên\_biến):**        hiển thị kết quả không có tên biến.

**Echo:** điều khiển cửa sổ lặp lại các lệnh của script file.

**Input:** sử dụng dấu nhắc để đưa dữ liệu vào.

**Keyboard:** trao điều khiển tạm thời cho bàn phím.

**Pause:** dừng lại cho đến khi người dùng nhấn một phím bất kỳ.

**Pause(n):** dừng lại  $n$  giây.

**Waitforbuttonpress:** dừng lại cho đến khi người dùng nhấn một phím bất kỳ.

Ví dụ: Viết chương trình nhập ba số  $a, b, c$  từ bàn phím. Kiểm tra xem ba số đó có tạo thành tam giác không? Nếu có thì tính chu vi và diện tích tam giác đó?

Chọn các mục File/new/M-file, tại cửa sổ Script file ta nhập vào các lệnh như sau:

```
a=input('nhap vao do dai canh a: ');
```

```
b=input('nhap vao do dai canh b: ');
```

```
c=input('nhap vao do dai canh c: ');
```

```

if (a>0) & (b>0) & (c>0) & (a+b>c) & (c+b>c) & (a+c>b)
cv=a+b+c;
disp ('chu vi tam giac la: ');
cv
p=cv/2;
disp ('dien tich tam giac la: ');
s=sqrt(p*(p-a)*(p-b)*(p-c))
else
disp ('3 so khong tao thanh tam giac ');
end;

```

Sau đó chọn File/save và gõ vào tên tệp là tamgiac.m. Chọn Debug/Run để chạy hàm, nhập dữ liệu cho ba cạnh của tam giác từ cửa sổ lệnh.

```

>> tamgiac
nhap vao do dai canh a : 3
nhap vao do dai canh b : 4
nhap vao do dai canh c : 5
chu vi tam giac la :
cv =
    12
dien tich tam giac la :
s =
     6

```

### 1.3.2. Cấu trúc điều khiển/rẽ nhánh

#### - Lệnh If:

```

IF expression
    statements
ELSEIF expression
    statements
ELSE
    statements
END

```

Ví dụ: Tìm nghiệm thực của phương trình bậc hai với các hệ số nhập từ bàn phím.

Chọn các mục File/new/M-file, tại cửa sổ Script file ta nhập vào các lệnh như sau:

```

a=input('a=');
b=input('b=');
c=input('c=');
delta=b*b-4*a*c

```

```

if delta<0
k='phuong trinh vo nghiem'
elseif delta==0
    x1=-b/(2*a)
    x2=x1
else
    x1=(-b+sqrt(delta))/(2*a)
    x2=(-b-sqrt(delta))/(2*a)
end

```

Sau đó chọn File/save và gõ vào tên tệp là nghiemptbac2.m. Chọn Debug/Run để chạy hàm, nhập dữ liệu cho các hệ số của phương trình bậc hai.

```

>> giaiptbac2
a=3
b=2
c=-5
delta =
    64
x1 =
    1
x2 =
   -1.6667

```

#### **- Lệnh switch:**

```

SWITCH switch_expr
CASE case_expr,
    statement,..., statement
CASE {case_expr1, case_expr2, case_expr3,...}
    statement,..., statement
OTHERWISE,
    statement,...,statement
END

```

Ví dụ: Hiện ra thời khoá biểu tương ứng với thứ (số) nhập từ bàn phím.

Chọn các mục File/new/M-file, tại cửa sổ Script file ta nhập vào các lệnh như sau:

```

t=input('nhap vao thu =');
switch t
case 2
    'Tin, Toan, Anh'

```

```

case 3
    ' Sinh, Su, Dia'
case 4
    ' Hoa, Ly, Anh'
case 5
    ' Toan, Tin, Ly'
case 6
    ' Van, Hoa, Dia'
case 7
    ' Toan, Su, Tin'
otherwise
    ' Nghi o nha'
end

```

Chọn File/save và gõ vào tên tệp là thoikhoabieu.m. Chọn Debug/Run để chạy hàm.

```

>> thoikhoabieu
nhap vao thu =5
ans =
    Toan, Tin, Ly

```

#### - Vòng lặp for:

```

FOR variable = expr
    statement,..., statement
END

```

Ví dụ: Viết chương trình tìm giai thừa của một số nhập từ bàn phím.

Chọn các mục File/new/M-file, tại cửa sổ Script file ta nhập vào các lệnh như sau:

```

n=input('nhap n tu keyboard:')
tg=1;
for i=1:n
    tg=tg*i;
end
tg

```

Chọn File/save và gõ vào tên tệp là tinhgiaithua.m. Chọn Debug/Run để chạy chương trình.

```

>> giaithua
nhap n tu keyboard:5
n =
     5
tg =
    120
- Vòng lặp while:

```

```

WHILE expression
    statements
END

```

Ví dụ: Tính  $q = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$

Chọn các mục File/new/M-file, tại cửa sổ Script file ta nhập vào các lệnh như sau:

```

n=input('nhap vao n :')
q=0;
i=1;
while i<=n
    q=q+1/i;
    i=i+1;
end
q

```

Chọn File/save và gõ vào tên tệp là tinhtongdayso.m. Chọn Debug/Run để chạy chương trình.

```

>> tinhtongdayso
nhap vao n :7
n =
    7
q =
    2.5929

```

### 1.3.3. Làm việc với file dữ liệu

#### \* Xuất và nhập văn bản

Lệnh **input** được sử dụng để yêu cầu người dùng nhập vào một số hay một chuỗi.

```

>> myname = input('Enter your name: ','s');
>> age = input('Enter your age: ');

```

Lưu ý: Ý nghĩa của hai lệnh trên, lệnh thứ nhất yêu cầu nhập vào một chuỗi và lệnh thứ hai yêu cầu nhập vào một số.

Để lưu trữ các biến trong workspace vào một file ta sử dụng lệnh `save tenfile`.

Để lấy lại dữ liệu đã ghi vào file ta sử dụng lệnh `load tenfile`.

Ví dụ: Tạo một ma trận A, và một đoạn ký tự B, sau đó lưu file này với tên là `dulieu`

```

>>A=[1 2 3;4 5 6;7 8 9];
>>B = 'Hanoi Vietnam';
>>Save dulieu

```

Khi cần sử dụng dữ liệu này, ta gõ lệnh:

```

>>load dulieu

```

Có hai hàm được sử dụng để xuất văn bản là **disp** và **fprintf**. Hàm **disp** chỉ thể hiện giá trị của một đối số là một ma trận số hay ma trận chuỗi. Ví dụ:

```
>> disp('This is a statement.') % xuất ra một chuỗi.  
This is a statement.  
>> disp(rand(3)) % xuất ra một ma trận.  
0.2221 0.0129 0.8519  
0.4885 0.0538 0.5039  
0.2290 0.3949 0.4239
```

#### + **Hàm FPRINTF**

Hàm **fprintf** (tương tự trong ngôn ngữ lập trình C) được sử dụng để ghi dữ liệu vào file hay xuất ra màn hình định dạng chính xác của đối số.

Cú pháp: COUNT = FPRINTF(FID,FORMAT,A,...) định dạng dữ liệu phần thực của ma trận A, theo định dạng cụ thể của FORMAT (dạng string) cấu trúc định dạng FORRMAT theo ngôn ngữ lập trình C, các định dạng bao gồm dấu %, và ghi dữ liệu vào file có nhận dạng là FID. Đối số COUNT là số byte ghi vào thành công.

Ví dụ:

```
>> x = 2;  
>> fprintf('Square root of %g is %8.6f.\n', x, sqrt(x));  
Square root of 2 is 1.414214.  
>> str = 'beginning';  
>> fprintf('Every %s is difficult.\n',str);  
Every beginning is difficult.
```

Hàm **fprintf** có khả năng biến đổi, định dạng và ghi đối số của nó vào một file hay thể hiện chúng trên màn hình theo một định dạng đặc biệt. Cú pháp sau đây được sử dụng để xuất ra màn hình:

```
fprintf (format, a, ...)
```

Chuỗi **format** chứa các ký tự thông thường sẽ được đưa đến ngõ ra và các chỉ định biến đổi, mỗi chỉ định biến đổi sẽ chuyển đổi một đối số của hàm **fprintf** thành chuỗi ký tự có định dạng tương ứng và sau đó in ra màn hình. a là các đối số của hàm **fprintf**. Mỗi chỉ định biến đổi bắt đầu bằng ký tự ‘%’ và kết thúc bởi một ký tự chuyển đổi. Giữa ‘%’ và ký tự chuyển đổi có thể là:

- dấu '-': khi có ký tự này đối số sẽ được in ra với định dạng canh lề trái, trong trường hợp ngược lại, mặc định là canh lề phải.
- số xác định chiều dài tối thiểu: là phạm vi tối thiểu mà đối số sẽ được in ra, trong trường hợp chiều dài của đối số lớn hơn chiều dài tối thiểu thì phạm vi in ra sẽ bằng với chiều dài của đối số.
- số xác định số các chữ số thập phân
- dấu ‘.’: là dấu dùng để phân biệt hai định dạng trên.

d	Biến đổi đối số sang kiểu thập phân
u	biến đổi đối số sang kiểu số thập phân không dấu
c	biến đổi đối số thành một ký tự
s	biến đổi đối số thành một chuỗi
e	biến đổi số <b>single</b> hay <b>double</b> thành số thập phân có dạng <b>[±]m.nnnnnnE[±]xx.</b>
f	biến đổi số <b>single</b> hay <b>double</b> thành số thập phân có dạng <b>[±]mmm.nnnnnn.</b>
g	định dạng giống trường hợp e và f nhưng được viết ở dạng ngắn hơn. Các số zero không có ý nghĩa sẽ không được in ra

Các định dạng \n, \r, \t, \b lần lượt được sử dụng để tạo ra một dòng mới, xuống dòng, tab và tab ngược (ngược với phím tab). Để in ra ký tự '\' ta sử dụng '\\', tương tự, để in ra ký tự '%' ta sử dụng '%%'. Phân tích ví dụ sau đây:

```
>> fprintf('look at %20.6e!\n', 1000*sqrt(2))
look at 1.414214e+3!
>> fprintf('look at %-20.6f!', 1000*sqrt(2))
look at 1414.213562 !
```

Trong cả hai trường hợp, khoảng tối thiểu để in số là 20 và số chữ số thập phân là 6.

Trong trường hợp đầu, giá trị 1000\*sqrt(2) được định dạng canh lề phải và trong trường hợp thứ hai, bởi vì có dấu '-' nên giá trị 1000\*sqrt(2) được định dạng canh lề trái. Sự khác nhau của hai kết quả còn do ý nghĩa của các ký tự biến đổi e và f.

#### + **Hàm FOPEN**

FID = FOPEN(FILENAME) mở một file có tên FILENAME. Trong đó: FILENAME có thể là gồm cả đường dẫn của file.

FID là một số nguyên gọi là nhận dạng của file. Ta sử dụng FID làm đối số đầu tiên để tác động vào/ ra với file trong các hàm khác. Nếu không mở được file thì FID = -1.

FID = FOPEN(FILENAME, PERMISSION) sẽ mở file tên FILENAME hoạt động theo chế độ PERMISSION. Đối số PERMISSION có thể là:

- 'r'    đọc
- 'w'    ghi (có thể tạo nếu cần thiết)
- 'a'    thêm (có thể tạo nếu cần thiết)
- 'r+'   đọc và ghi (không tạo)
- 'w+'   cắt ngắn hoặc tạo để đọc và ghi
- 'a+'   đọc và thêm (có thể tạo nếu cần thiết)

#### + **Hàm FCLOSE**



ST = FCLOSE(FID) đóng file có nhận dạng FID. Hàm FCLOSE sẽ cho kết quả là 0 nếu đóng file thành công, và bằng – 1 nếu không thành công.

Dưới đây là một ví dụ tạo một file ghi dữ liệu vào file và đóng file:

```
>>x = 0:.1:1; y = [x; exp(x)]; % Tạo ma trận y=exp(x).
>>fid = fopen('exp.txt','w'); % Mở (tạo mới) file: exp.txt
>>fprintf(fid,'%6.2f %12.8f\n',y); % Ghi ma trận y vào file
>>fclose(fid); % Đóng file
>>tg=load('exp.txt'); % lấy dữ liệu từ file đưa vào biến tg
```

### 1.3.4. Hàm m-File

MATLAB đã xây dựng rất nhiều hàm để người sử dụng chỉ việc gọi ra và dùng (như một số ví dụ ở trên). Tuy nhiên, MATLAB cũng cung cấp cấu trúc để người sử dụng có thể xây dựng các hàm của mình dưới dạng M- file, khi đó hàm lập trình được sẽ sử dụng như một hàm có sẵn trong thư viện của MATLAB. Cấu trúc một hàm M-file như sau:

```
Function Tên_biến= Tên_hàm(các tham số vào)
% Các câu chú thích
khởi lệnh
Tên_biến = biểu thức
[return]
```

Các quy định và các tính chất với hàm M-file:

- Tên\_hàm và tên M-file phải giống nhau.
- Trong thân hàm có lệnh gán giá trị của biểu thức cho tên biến.
- Trong hàm có thể chứa các hàm khác nhưng các hàm con trong hàm đó chỉ được gọi chỉ được gọi trong chính nó.
- Mỗi hàm có một không gian làm việc riêng tách biệt so với môi trường MATLAB. Các biến được tạo ra trong hàm chỉ nằm trong không gian làm việc của hàm đó và được giải phóng khi hàm kết thúc.
- Các dòng chú thích sẽ được hiện ra khi dùng lệnh help.
- Các tham số vào và ra khi một hàm được gọi chỉ có tác dụng bên trong hàm đó. Biến nargin chứa các tham số đưa vào, narginout chứa các giá trị đưa ra.
- Nếu các hàm muốn dùng chung các biến thì các biến đó phải khai báo là biến toàn cục : Global tên\_biến
- Trong hàm có thể gặp dòng lệnh Return, cho phép kết thúc một hàm mà không cần phải thực hiện hết các lệnh của hàm đó.
- Hàm error của MATLAB cho hiển thị chuỗi kí tự trên cửa sổ lệnh và dùng thực hiện hàm. Hàm này thường được dùng để cảnh báo việc sử dụng hàm không đúng. Ví dụ hàm phải đưa vào ba tham số, nếu không đủ tham số thì sẽ có lỗi :

```
if nargin <3
    error('Phải đưa vào du 3 tham so')
end
```

- Lời gọi hàm: Trong các hàm khác hoặc tại cửa sổ lệnh sử dụng hàm phải có lời gọi hàm. Lời gọi hàm viết như sau :

Tên\_biến=Tên\_hàm(các tham số thực sự)

Các tham số thực sự chứa giá trị và tương ứng với các tham số trong hàm.

Ví dụ : Xây dựng hàm tính diện tích của tam giác biết 3 cạnh a,b,c.

Chọn các mục File/new/M-file, tại cửa sổ Script file ta nhập vào các lệnh như sau:

```
function dttg(a,b,c)
% Ham tinh dien tich tam giac biet 3 canh a,b,c
% Nhap vao gia tri 3 canh tam giac (dieu kien cac canh duong va
% 1 canh phai nho hon tong 2 canh)
```

```
if nargin <3
error('Phai dua vao du 3 tham so')
end
disp('Chu vi tam giac:')
p=(a+b+c)/2
disp('Dien tich tam giac:')
y=sqrt(p*(p-a)*(p-b)*(p-c))
```

Chọn File/save và gõ vào tên tệp là tinhtongdayso.m. Chọn Debug/Run để chạy chương trình.

```
>> dttg(2,3,4)
```

```
Chu vi tam giac:
```

```
p =
```

```
4.5000
```

```
Dien tich tam giac:
```

```
y =
```

```
2.9047
```

```
ans =
```

```
2.9047
```

```
>> help dttg
```

```
Ham tinh dien tich tam giac biet 3 canh a,b,c
```

```
Nhap vao gia tri 3 canh tam giac (dieu kien cac canh duong va %
1 canh phai nho hon tong 2 canh)
```

**\* Một số lệnh quản lý file:**

**cd:** hiển thị thư mục hiện thời

**cd tên\_thư\_mục:** thay đổi thư mục đưa ra.

**dir:** danh sách các file trong thư mục hiện thời.

**edit tên\_tệp\_m\_file:** mở tệp để soạn thảo.

**delete tên\_tệp\_m\_file:** xóa tệp.

**path:** hiển thị hoặc sửa đường dẫn.

**type tên\_tệp\_m\_file:** hiển thị nội dung M-file trong cửa sổ lệnh

**what:** hiển danh sách các m-file và MAT-file

### 1.3.5. Các phép tính logic

*\* Các toán tử quan hệ*

**Bảng 1.8. Một số toán tử quan hệ**

Toán tử quan hệ	Ý nghĩa
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
>	Lớn hơn
>=	Lớn hơn hoặc bằng
==	Bằng
~=	Khác

Kết quả của toán tử quan hệ cho giá trị 1 (đúng) hoặc 0 (sai).

Ví dụ:

```
>> a=3;
```

```
>> b=5;
```

```
>> a==b
```

```
ans =
```

```
0
```

```
>> a>b
```

```
ans =
```

```
0
```

```
>> a~=b
```

```
ans =
```

```
1
```

- Các toán tử quan hệ thực hiện so sánh từng phần tử giữa hai mảng. Nó cho kết quả là một mảng logic có cùng kích cỡ, với các phần tử của mảng là 1 nếu quan hệ đó là đúng, và phần tử của mảng là 0 nếu sai.

- Các toán tử <, >, <=, và >= chỉ sử dụng phần thực của các toán hạng cho phép so sánh. Các toán tử == và ~= kiểm tra cả phần thực và phần ảo.

Ví dụ:

```
>> x=3*eye(3)
```

```
x =
```

```
3    0    0
```

```

0      3      0
0      0      3
>> y=[1 2 3;4 5 6;-1 2 -3]
y =
     1     2     3
     4     5     6
    -1     2    -3
>> x<y
ans =
     0     1     1
     1     1     1
     0     1     0

```

### \* Các toán tử logic

**Bảng 1.9. Một số toán tử quan hệ**

Toán tử logic	Ý nghĩa
&	AND
	OR
~	NOT

Kết quả của toán tử logic cho giá trị 1 (đúng) hoặc 0 (sai).

Ví dụ:

```

>> a=6;
>> b=-3;
>> (a>0) & (b>0)
ans =
     0
>> (a>0) & (b<0)
ans =
     1

```

Ký hiệu: ~= nghĩa là không bằng.

~(x>5) nghĩa là x không lớn hơn 5, nếu x=4 thì phép kiểm tra này sẽ cho kết quả là 1.

- Một ứng dụng khác nữa của các phép kiểm tra logic là ta có thể 'xóa' (hay bỏ qua) một số phần tử của ma trận:

Ví dụ:

Nhập một vectơ x và hằng số a. Giữ lại các phần tử của x có giá trị lớn hơn a?

```

>> x=[1 2 -3 0 4 2 7 3]
x =
     1     2    -3     0     4     2     7     3

```

```
>> a=3;
>> check=x>3
check =
     0     0     0     0     1     0     1     0
>> x=x.*check
x =
     0     0     0     0     4     0     7     0
```

#### \* Thời gian

- **clock**: Hàm trả về ngày và giờ hiện tại
- **date**: Hàm trả về ngày hiện tại.
- **weekday**: Hàm trả về thứ trong tuần.
- **eomday(năm,tháng)**: Hàm trả về ngày cuối tháng.
- **calendar(năm,tháng)**: Hàm trả về lịch tháng

```
>> eomday(2014,7)
ans =
    31
>> clock
ans =
    1.0e+03 *
    2.0140    0.0070    0.0080    0.0140    0.0260    0.0350
>> calendar(2014,7)

          Jul 2014
S      M      Tu      W      Th      F      S
0       0       1       2       3       4       5
6       7       8       9      10      11      12
13      14      15      16      17      18      19
20      21      22      23      24      25      26
27      28      29      30      31       0       0
0       0       0       0       0       0       0
```

## 1.4. ĐỒ HỌA TRONG MATLAB

### 1.4.1. Đồ họa trong hệ tọa độ phẳng 2D

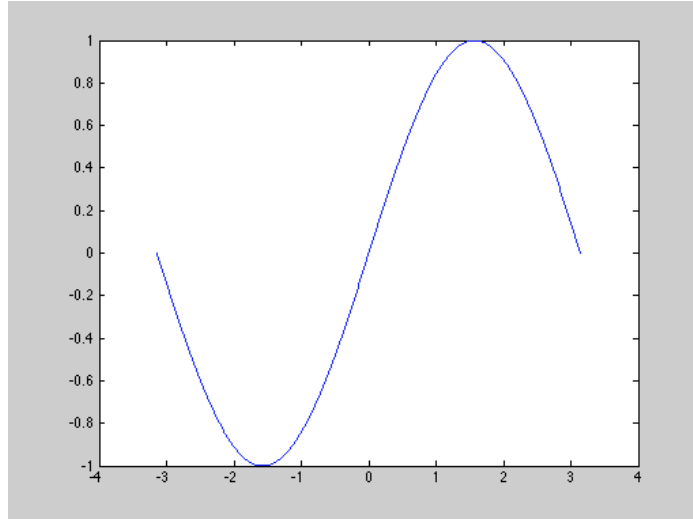
#### \* Plot(x,y)

Lệnh này dùng vẽ đồ thị của một mảng dữ liệu trên hệ trục thích hợp. Trong đó x là mảng dữ liệu cho trục x, y là mảng dữ liệu cho trục y, nối các điểm (x,y) bằng đường thẳng.

Ví dụ:

```
>> x=[-pi:0.01:pi];
>> y=sin(x);
```

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

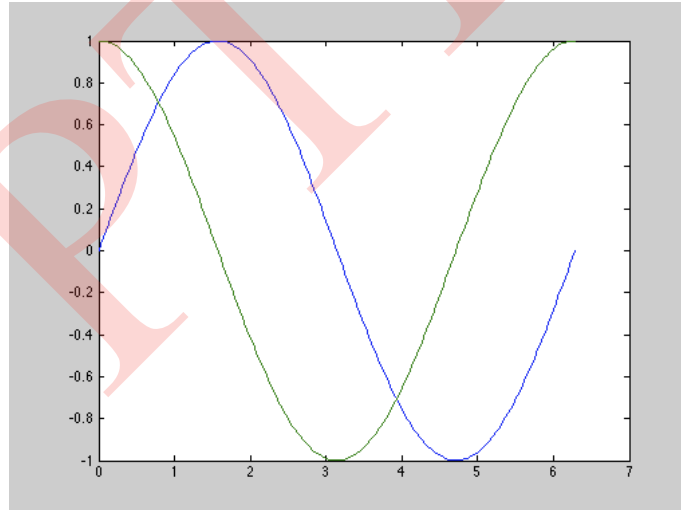


Hình 1.3. Đồ thị hình sin

### **Plot(x,y,x,z)**

Lệnh này vẽ hai hàm  $y$  và  $z$  theo  $x$  trên cùng một hệ trục. Ví dụ:

```
>> t=linspace(0,2*pi,200);  
>> y1=sin(t);  
>> y2=cos(t);  
>> plot(t,y1,t,y2)
```



Hình 1.4. Đồ thị hình sine và cosin trên cùng một figure

### **\* Kiểu đường, dấu, màu**

Nếu khi vẽ không khai báo, MATLAB sẽ để màu mặc định là blue, kiểu đường là solid và không có các điểm đánh dấu trên đồ thị. Tuy nhiên, ta cũng có thể khai báo màu, kiểu đường bằng cách đưa thêm vào lệnh plot các đối số thứ 3 sau mỗi cặp dữ liệu của mảng. Các đối số tùy chọn là một xâu kí tự, có thể chứa một hoặc nhiều kí tự theo bảng dưới đây.

**Bảng 1.10. Bảng quy ước màu sắc**

Mã	Màu	Mã	Màu
<b>r</b>	đỏ (red)	<b>m</b>	đỏ tím (magenta)
<b>g</b>	xanh lá (green)	<b>y</b>	vàng (yellow)
<b>b</b>	Xanh da trời (blue)	<b>k</b>	đen (black)
<b>c</b>	xanh xám (cyan)	<b>w</b>	trắng (white)

**Bảng 1.11. Các dạng điểm đánh dấu**

Mã	Kiểu đánh dấu	Mã	Kiểu đánh dấu
<b>+</b>	dấu cộng	<b>.</b>	điểm
<b>o</b>	vòng tròn	<b>x</b>	chữ thập
<b>*</b>	dấu sao	<b>s</b>	hình vuông
<b>d</b>	hạt kim cương	<b>v</b>	tam giác hướng xuống
<b>^</b>	tam giác hướng lên	<b>&lt;</b>	tam giác sang trái
<b>&gt;</b>	tam giác sang phải	<b>h</b>	lục giác
<b>p</b>	ngũ giác		

**Bảng 1.12. các dạng đường thẳng**

Mã	Kiểu đường	Mã	Kiểu đường
<b>-</b>	đường liền	<b>:</b>	đường chấm chấm
<b>--</b>	đường đứt nét	<b>-. </b>	đường gạch chấm

Cú pháp: **Plot(x,y, 'các biểu tượng màu - dấu - nét')**

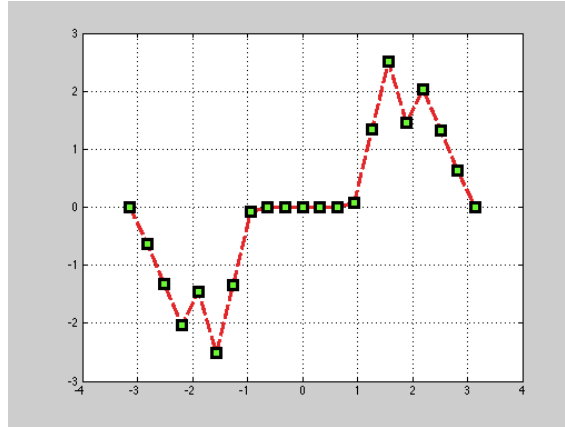
Ví dụ: Vẽ đồ thị thoả mãn yêu cầu sau:

Chương trình vẽ đường cong  $y = f(x)$  có các đặc tả:

- + đường vẽ là đường đứt nét (--).
- + khối đánh dấu hình vuông (s)
- + đường vẽ màu đỏ (r)
- + đường vẽ rộng 2 point
- + các cạnh của khối đánh màu đen
- + khối đánh dấu màu green
- + kích thước khối đánh dấu 10 point

```
x=[-pi:pi/10:pi];
y=tan(sin(x))-sin(tan(x));
plot(x, y, '--rs', 'LineWidth', 2, 'MarkerEdgeColor', 'k',...
'MarkerFaceColor', 'g', 'MarkerSize', 10)
```

grid

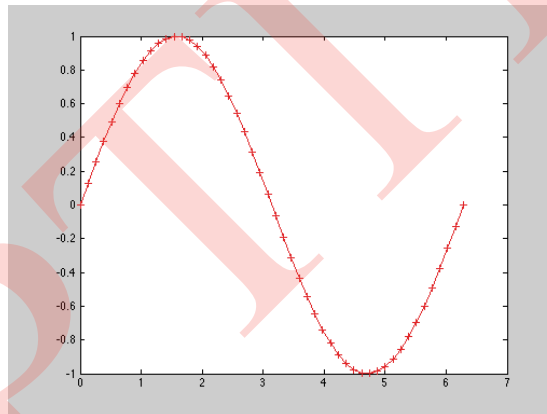


Hình 1.5. Đồ thị  $y=\tan(\sin(x))-\sin(\tan(x))$

Ví dụ:

```
>> t=linspace(0,2*pi,50);  
>> y1=sin(t);  
>> plot(t,y1,'r+-')
```

Đồ thị sẽ là đường màu đỏ, dấu +, nét liền



Hình 1.6. Đồ thị  $y=\sin(x)$

\* **Kiểu hiển thị đồ thị:** Lệnh **colordef** cho phép lựa chọn kiểu hiển thị. Giá trị mặc định của **colordef** là **white**, kiểu này sử dụng trục tọa độ, màu nền là màu trắng, tiêu đề trục màu đen nên hình vẽ màu xám sáng. Nếu thích nền màu đen ta có thể đổi lại bằng lệnh **colordef black**. Kiểu này sẽ cho nền trục tọa độ đen, nền hình vẽ màu tối xám và tiêu đề trục màu trắng.

\* **Trục, nhãn, lời chú giải, lưới, hộp chứa trục:**

- Trục: Việc phân chia thang bậc của trục thường được MATLAB tự động làm, tuy nhiên người sử dụng cũng có thể chia lại bằng lệnh:

```
axis([x_min,x_max,y_min,y_max]): Chia trục 2D (2 chiều)  
axis([x_min,x_max,y_min,y_max,z_min,z_max]): Chia trục 3D  
axis('auto'): Trao quyền chia trục lại cho MatLab
```

- Nhãn:



**xlabel('string'):** Điền nhãn trục x.

**ylabel('string'):** Điền nhãn trục y.

**zlabel('string'):** Điền nhãn trục z.

**title('string'):** Điền tiêu đề của đồ thị

**text (x\_value, y\_value, 'string'):** Điền đoạn văn bản vào đồ thị tại toạ độ (x\_value, y\_value)

**gtext('string'):** Thêm xâu kí tự vào vị trí kích chuột

- Lời chú giải

Legend (string\_1, string\_2,..., [position])

- Lưới:

grid on: hiện lưới

grid off: không hiện lưới (mặc định)

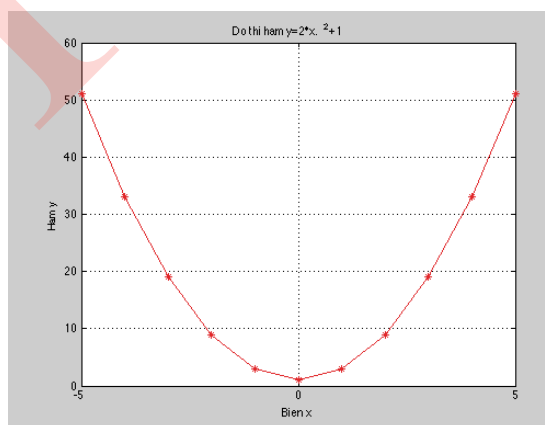
- Hộp chứa đồ thị

box off : tắt đi

box on: Khôi phục lại (mặc định)

Ví dụ:

```
x=[-5:5];  
y=2*x.^2+1;  
plot(x,y,'r*-')  
xlabel('Bien x')  
ylabel('Ham y')  
grid  
title('Do thi ham y=2*x.^2+1')
```



**Hình 1.7. Đồ thị hàm số  $y = 2x^2 + 1$**

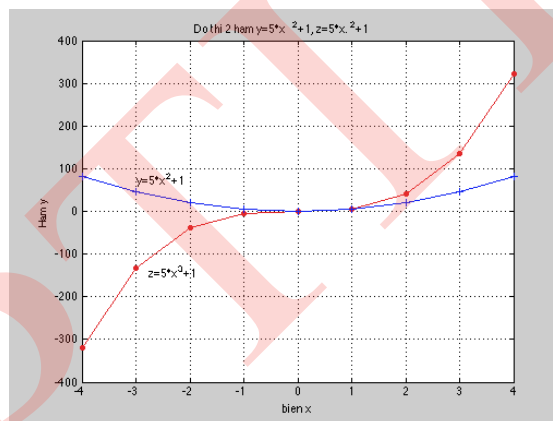
\* Chú ý: Nếu lệnh plot được gọi mà chỉ có một đối số (ví dụ: plot(y)) thì hàm plot sẽ đưa ra kết quả khác nhau phụ thuộc vào dữ liệu chứa trong y.

+ Nếu  $y$  là số phức thì  $\text{plot}(y)$  tương đương với  $\text{plot}(\text{real}(y))$  và  $\text{plot}(\text{imag}(y))$ , trong tất cả các trường hợp khác thì phần ảo của  $y$  thường được bỏ qua.

+ Nếu  $y$  là số thực thì  $\text{plot}(y)$  tương ứng với  $\text{plot}(1:\text{length}(y),y)$ .

Ví dụ:

```
x=[-4:4];  
y=5*x.^3+1;  
plot(x,y)  
box off  
z=5*x.^2+1;  
plot(x,y,'r.-',x,z,'b+-')  
gtext('y=5*x^2+1')  
gtext('z=5*x^3+1')  
title('Do thi 2 ham y=5*x^2+1, z=5*x.^2+1')  
xlabel('bien x')  
ylabel('Ham y')  
grid
```

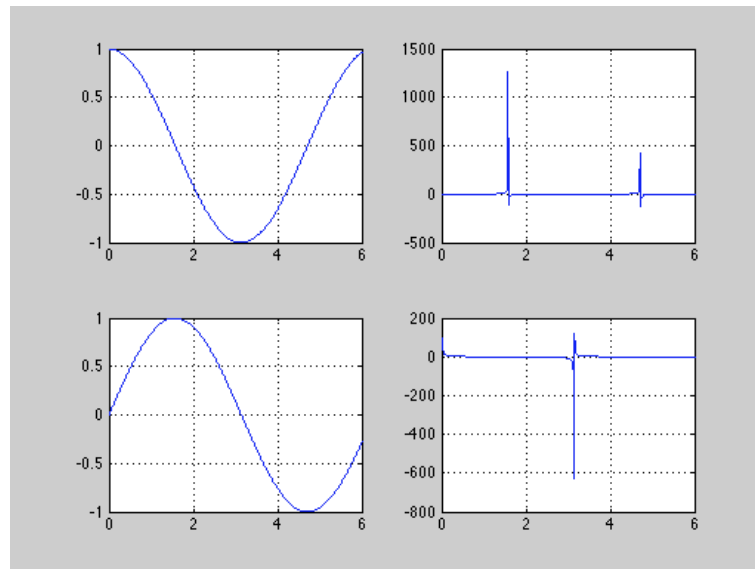


Hình 1.8. Hai đồ thị trên cùng một figure

#### \* Một đồ thị chứa nhiều hệ trục

**subplot(m,n,p)** trong đó cửa sổ hiện chia thành ma trận  $m \times n$  khoảng để vẽ đồ thị,  $p$  là cửa sổ hoạt động. Các cửa sổ được đánh số từ trái qua phải, từ trên xuống dưới.

```
x=[0:0.01:6];  
y=cos(x);  
z=sin(x)./cos(x);  
t=sin(x);  
p=cos(x)./sin(x);  
subplot(2,2,1), plot(x,y), grid  
subplot(2,2,2), plot(x,z), grid  
subplot(2,2,3), plot(x,t), grid  
subplot(2,2,4), plot(x,p), grid
```



Hình 1.9. Một đồ thị chứa nhiều hệ trục

### \* Hệ trục tạo độ

MATLAB cung cấp công cụ kiểm soát hình dáng và thang chia của hia trục tọa độ bằng lệnh **axis**. Các thông tin đầy đủ về lệnh này có thể xem bằng lệnh **help axis**. Các trường hợp cơ bản của lệnh này được trình bày theo bảng dưới đây:

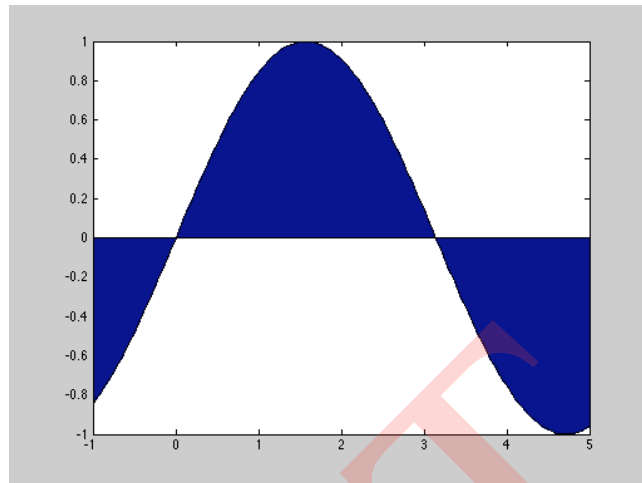
Bảng 1.13. Các trường hợp của lệnh axis

Lệnh	Chức năng
<code>axis([xmin xmax ymin ymax])</code>	Thiết lập các giá trị min, max của 2 trục theo giá trị trong véc tơ hàng.
<code>v=axis</code>	v là véc tơ có chứa thang chia cho đồ thị theo dạng [xmin xmax ymin ymax]
<code>axis auto</code>	Thang chia mặc định
<code>axis('auto')</code>	xmin=min(x), xmax=max(x),...
<code>axis manual</code>	Giới hạn thang chia như thang chia hiện tại
<code>axis xy</code>	Sử dụng hệ tọa độ decac (mặc định), gốc ở góc trái bên dưới.
<code>axis ij</code>	Sử dụng hệ tọa độ ma trận, gốc ở góc trên trái.
<code>axis square</code>	Thiết lập đồ thị hiện tại là hình vuông, mặc định là hình chữ nhật.
<code>axis equal</code>	Thiết lập thang giống nhau cho cả hai hệ trục
<code>axis tightequare</code>	Tương tự như axis equal nhưng hộp đồ thị vừa đủ đối với dữ liệu.
<code>axis normal</code>	Tắt các chế độ khác, ở chế độ bình thường
<code>axis off</code>	Tắt bỏ các chế độ nền, trục, nhãn,...
<code>axis on</code>	Ngược với axis off

**\* Một số dạng đồ thị khác**

**Area(x,y): dạng diện tích**

```
>> x=[-1:0.05:5];
>> y=sin(x);
>> area(x,y)
```

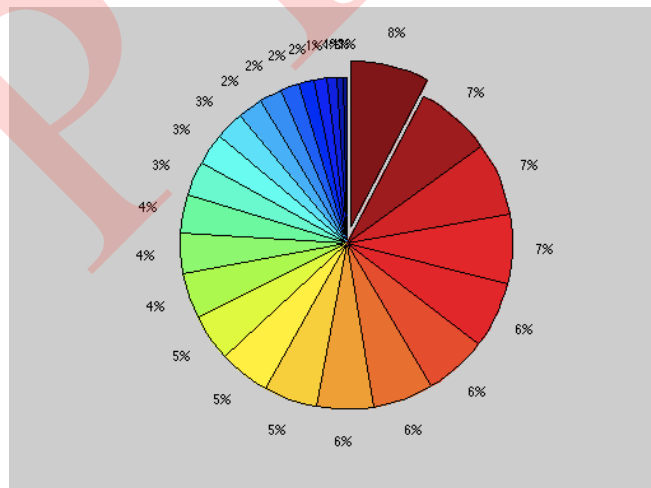


**Hình 1.10. Đồ thị dạng area**

**pie(a,b): dạng bánh tròn**

*a* là véc tơ giá trị, *b* là véc tơ logic tùy chọn.

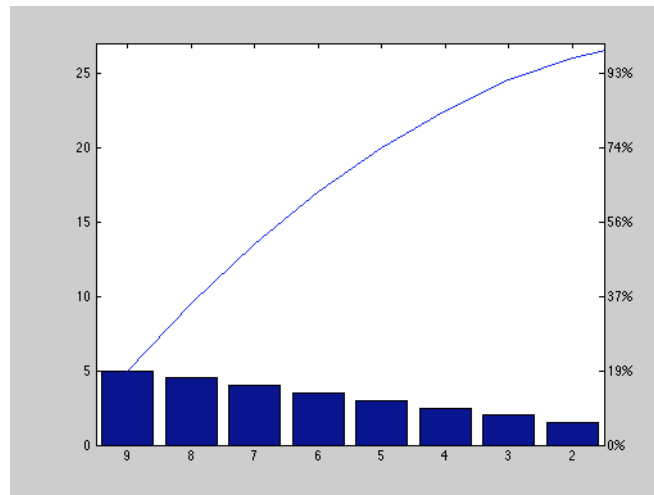
```
>> pie(x,x==max(x))
>> box off
>> pie(x,x==max(x))
```



**Hình 1.11. Đồ thị dạng pie**

**pareto(x): dạng cột**

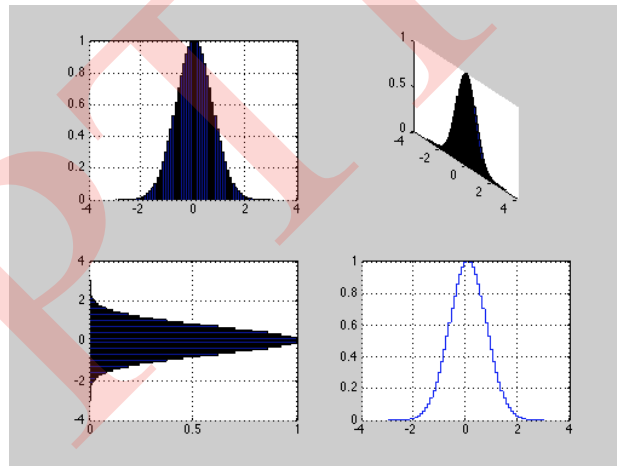
```
>> x=[1:0.5:5];
>> pareto(x)
```



Hình 1.12. Đồ thị dạng pareto

Ví dụ:

```
>> x=[-3:0.1:3];
>> y=exp(-x.^2);
>> subplot(221),bar(x,y),grid
>> subplot(222),bar3(x,y),grid
>> subplot(223),barh(x,y),grid
>> subplot(224),stairs(x,y),grid
```



Hình 1.13. Một số dạng đồ thị đặc biệt

\* Các lệnh khác:

- clf: xóa nội dung của figure hiện tại.
- delete figure (number): Xóa figure.
- close (number)/ close all: Đóng figure

#### 1.4.2. Đồ thị trong không gian ba chiều (3D)

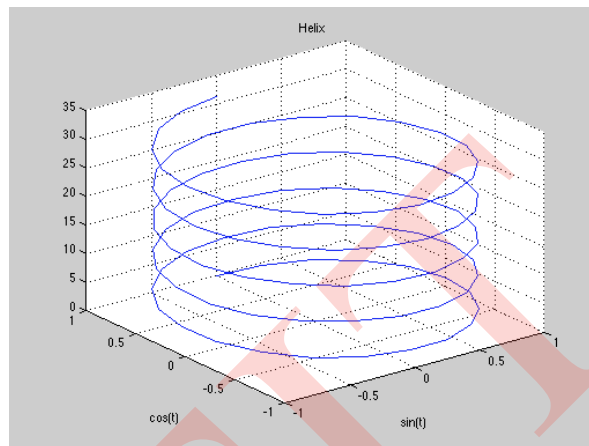
\* Lệnh plot từ trong không gian 2 chiều có thể mở rộng cho không gian 3 chiều bằng lệnh plot3 như sau:

```
plot3 (x1, y1, z1, S1, x2, y2, z2, S2,...)
```

Trong đó  $x_n$ ,  $y_n$  và  $z_n$  là các vector hoặc ma trận, và  $s_n$  là xâu kí tự tùy chọn dùng cho việc khai báo màu và kiểu đường.

Ví dụ:

```
>> t = linspace (0,10*pi);  
>> plot3( sin(t), cos(t), t)  
>> title('Helix')  
>> xlabel('sin(t)')  
>> ylabel('cos(t)')  
>> grid
```



Hình 1.14. Đồ thị 3D

#### \* Đồ thị lưới:

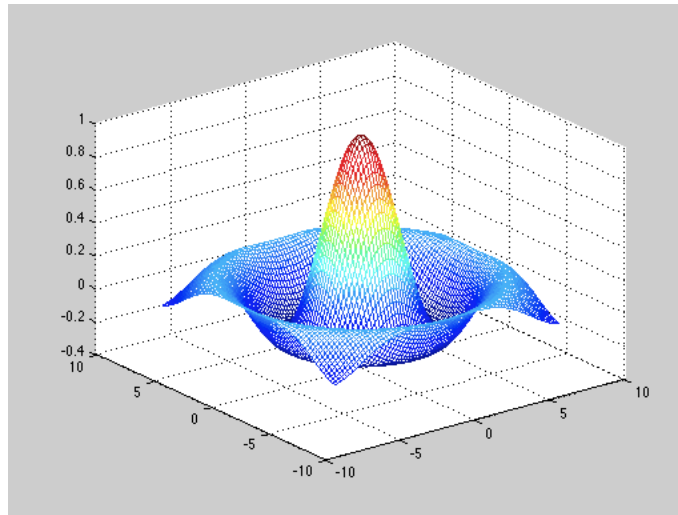
MATLAB định nghĩa bề mặt lưới bằng các điểm theo hướng trục  $z$  ở trên đường kẻ ô hình vuông trên mặt phẳng  $xy$ . Nó tạo lên một mẫu đồ thị bằng cách ghép các điểm gần kề với các đường thẳng. Kết quả là nó trông như một mạng lưới đánh cá với các mặt lưới là các điểm dữ liệu. Đồ thị loại này thường dùng để quan sát những ma trận lớn hoặc vẽ những hàm có 2 biến.

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 một ma trận  $X$  mà các hàng của nó là bản sao của vector  $x$  và ma trận  $Y$  có các cột của nó là bản sao của vector  $y$ . 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.

Ví dụ:

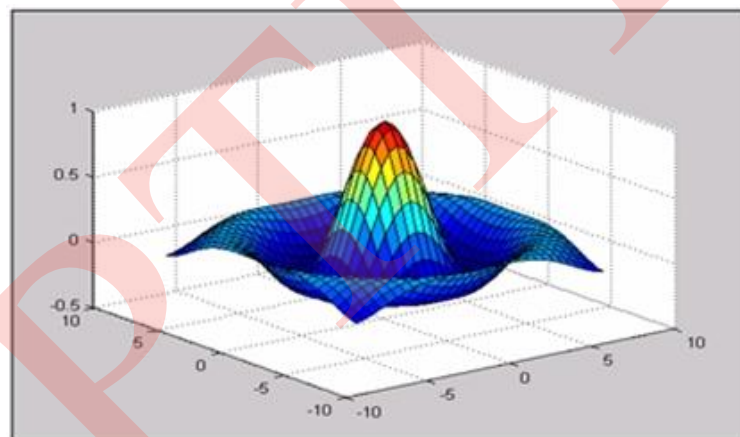
```
>> x = -7.5: 0.2:7.5;  
>> y = x;  
>> [X, Y] = meshgrid(x, y);  
>> R = sqrt(X.^2 + Y.^2)+eps;  
>> Z = sin(R)./R;  
>> mesh (X, Y, Z)
```



**Hình 1.15. Đồ thị dạng MESH**

Ma trận R chứa bán kính của mỗi điểm trong [X, Y], nó là khoảng cách từ mỗi điểm đến tâm ma trận. Cộng thêm **eps** để không xảy ra phép chia cho 0. Ma trận Z chứa sin của bán kính chia cho bán kính mỗi điểm trong sơ đồ.

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



**Hình 1.16. Đồ thị dạng SURF**

### \* Thao tác với đồ thị

MATLAB cho phép khai báo góc để từ đó có thể quan sát được đồ thị trong không gian 3 chiều.

- Hàm `view(azimuth, elevation)` thiết lập góc xem bằng việc khai báo azimuth và elevation. "Elevation" mô tả vị trí người quan sát, xem như là góc đo bằng độ trên hệ trục x-y. "Azimuth" mô tả góc trong hệ trục nơi người quan sát đứng.

"Azimuth" được đo bằng độ từ phần âm trục y. Phần âm trục y có thể quay theo chiều kim đồng hồ một góc -37,50 từ phía người quan sát, "Elevation" là góc mà tại đó mắt người quan sát thấy được mặt phẳng x-y. Sử dụng hàm `view` cho phép người quan sát có thể quan sát hình vẽ từ các góc độ khác nhau

Ví dụ: `view(-50,0)`

- Lệnh "view" có một dạng khác rất tiện ích khi sử dụng là view([X Y Z]) cho phép bạn quan sát trên một vectơ chứa hệ trục tọa độ decac. Khoảng cách từ vị trí bạn quan sát đến gốc tọa độ không bị ảnh hưởng.

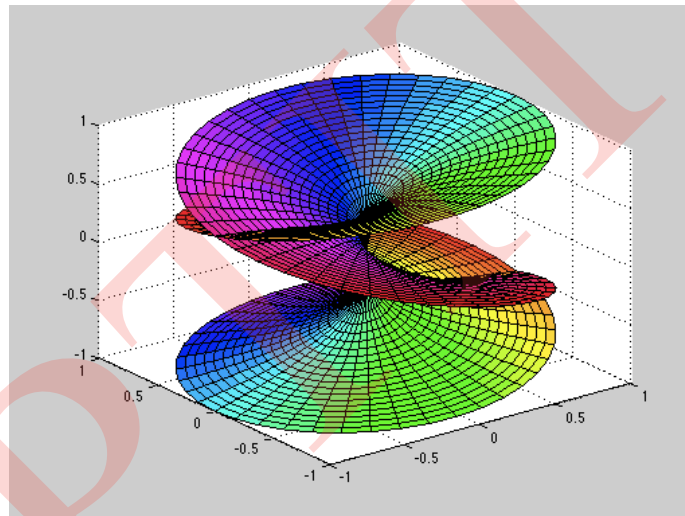
Ví dụ: view([0 10 0]), view([0 -1 0]) và view ([0 0]) cho kết quả giống nhau.

Một công cụ khác cũng rất hữu dụng với việc quan sát đồ thị không gian 3 chiều là hàm **rotate3d**. Các thông số Azimuth và elevation có thể được tác động bởi chuột, lệnh rotate3d on cho phép chuột can thiệp, còn rotate3d off không cho phép.

Lệnh Hidden dấu các nét khuất. Khi ta vẽ đồ thị, một số phần của nó bị che khuất bởi các phần khác nên chỉ có thể nhìn thấy phần ở trong tầm nhìn. Nếu chuyển đến hidden off, ta có thể thấy phần khuất đó qua mạng lưới. Mặc định là hidden on.

Ví dụ:

```
>> colormap(hsv(64))
>> z = cplxgrid(30);
>> cplxroot(3)
```



Hình 1.17. Đồ thị hàm  $y = \sqrt[3]{z}$  trong không gian số phức

## 1.5. SIMULINK

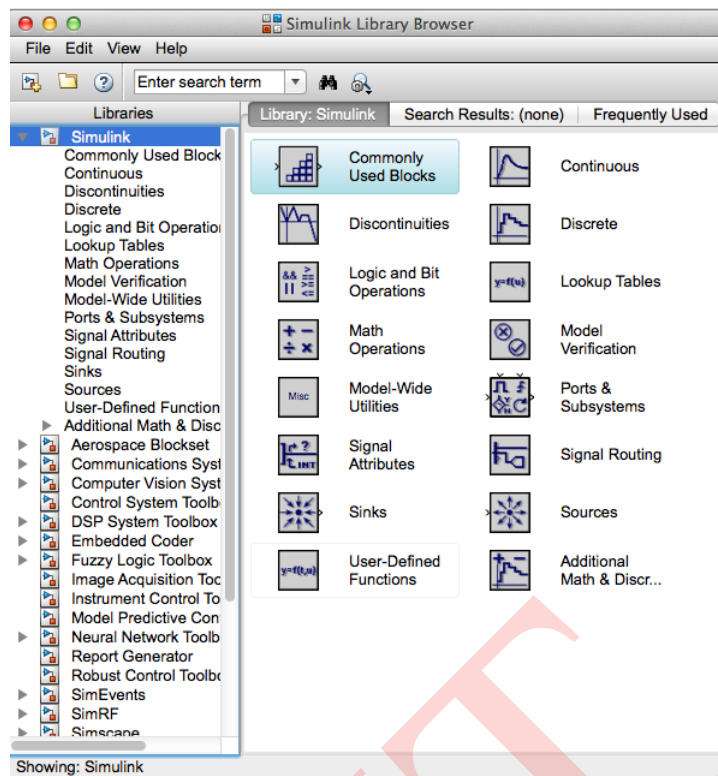
SIMULINK là một công cụ rất mạnh của MATLAB để xây dựng các mô hình một cách trực quan và dễ hiểu. Để mô tả hay xây dựng hệ thống ta chỉ cần liên kết các khối có sẵn trong thư viện của SIMULINK lại với nhau. Sau đó, tiến hành mô phỏng hệ thống để xem xét ảnh hưởng của bộ điều khiển đến đáp ứng quá độ của hệ thống và đánh giá chất lượng hệ thống.

**\* Các bước tiến hành để xây dựng một ứng dụng mới trong SIMULINK:**

+ Sau khi khởi động MATLAB, gõ lệnh simulink hoặc nhấn vào nút simulink trên thanh công cụ thì cửa sổ SIMULINK hiện ra (như ở Hình 1.8)

+ Trong cửa sổ SIMULINK, vào menu File / New để mở cửa sổ cho một ứng dụng mới.





**Hình 1.18. Cửa sổ Simulink**

Kích chuột vào các thư viện cần thiết để chọn khối cần tìm. Kích chuột trái vào khối này, sau đó kéo và thả vào cửa sổ ứng dụng vừa mới tạo ra. Nháy chuột kép vào khối này để cài đặt và thay đổi các thông số.

+ Có thể nhân số lượng các khối bằng cách dùng chức năng Copy và Paste. Kích chuột trái nối các ngõ vào / ra của các khối để hình thành sơ đồ hệ thống.

+ Có thể dời một hoặc nhiều khối từ vị trí này đến vị trí khác bằng cách nhấp chuột để chọn các khối đó và kéo đến vị trí mới. Dùng phím Delete để xóa các phần không cần thiết hay bị sai khi chọn.

+ Có thể viết chú thích trong cửa sổ ứng dụng bằng cách double click vào một vị trí trống và gõ câu chú thích vào. Vào menu Format / Font để thay đổi kiểu chữ.

+ Như vậy, mô hình hệ thống đã xây dựng xong. Bây giờ tiến hành mô phỏng hệ thống bằng cách vào menu Simulation / Simulation Parameters để cài đặt các thông số mô phỏng.

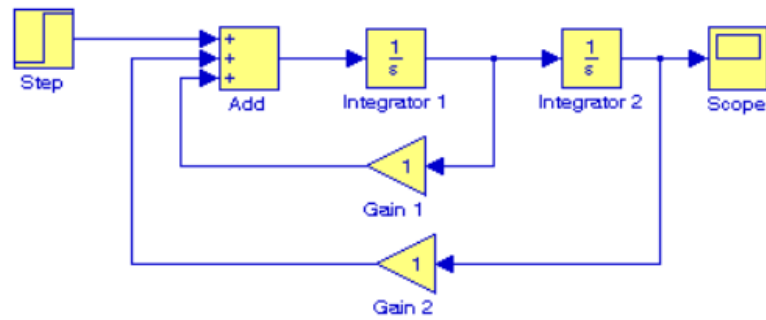
**\* Cửa sổ Simulation Parameters hiện ra như sau:**

+ Start time: thời điểm bắt đầu mô phỏng. Mặc định chọn bằng 0.

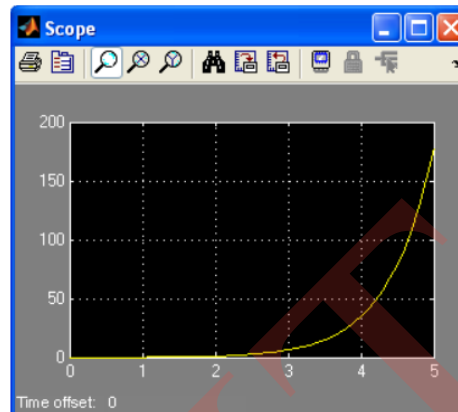
+ Stop time: thời điểm kết thúc mô phỏng. Giá trị này chọn theo đặc tính của hệ thống. Nếu hệ thống có thời hằng lớn thì giá trị Stop time cũng phải lớn để quan sát hết thời gian quá độ của hệ thống.

+ Các thông số còn lại chọn mặc định.

+ Chạy mô phỏng bằng cách vào menu Simulation / Start. Khi thời gian mô phỏng bằng giá trị Stop time thì quá trình mô phỏng dừng lại. Trong quá trình mô phỏng, nếu ta muốn dừng nửa chừng thì vào menu Simulation / Stop.



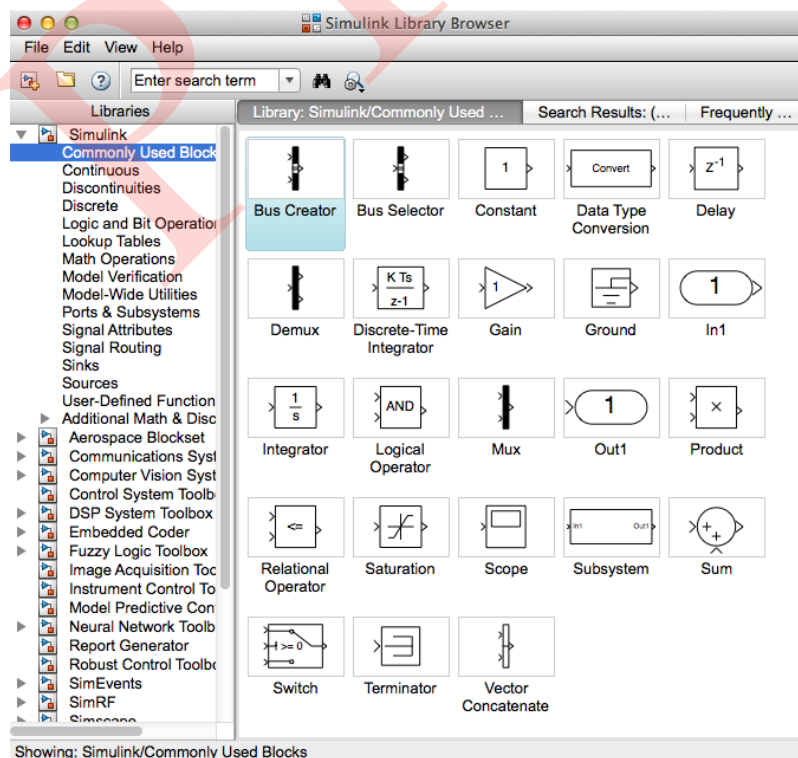
Hình 1.19. Mô phỏng hệ thống trên Simulink



Hình 1.20. Đáp ứng của hệ thống trong hình 1.19

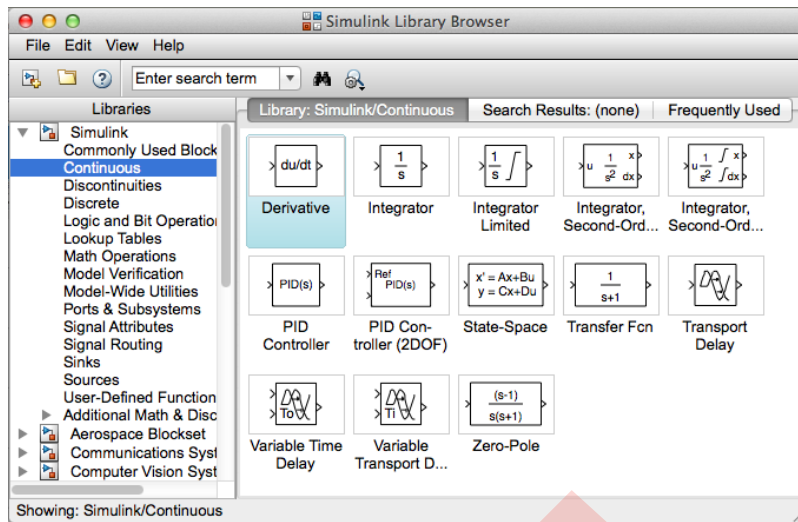
#### \* Một số thư viện thường dùng trong Simulink

- Thư viện Commonly Used Block: Là thư viện chứa các khối hay được sử dụng để xây dựng các hệ thống trên Simulink. Các khối trong thư viện này có thể nằm trong các thư viện khác (ví dụ khối Gain nằm trong thư viện Math Operations).



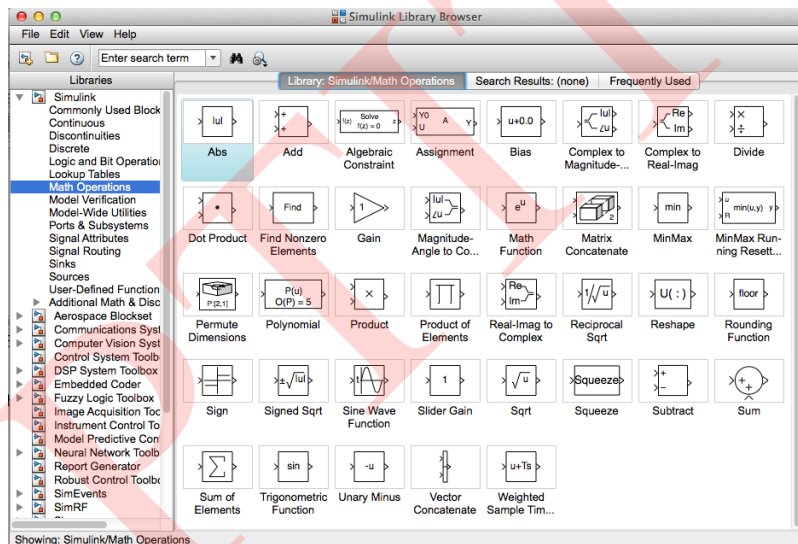
Hình 1.21. Thư viện Commonly Used Block

- Thư viện Continuous: Là thư viện chứa các khối để xây dựng hệ thống liên tục. Ngược với nó là các khối nằm trong thư viện Discontinuities dùng để xây dựng hệ thống rời rạc.



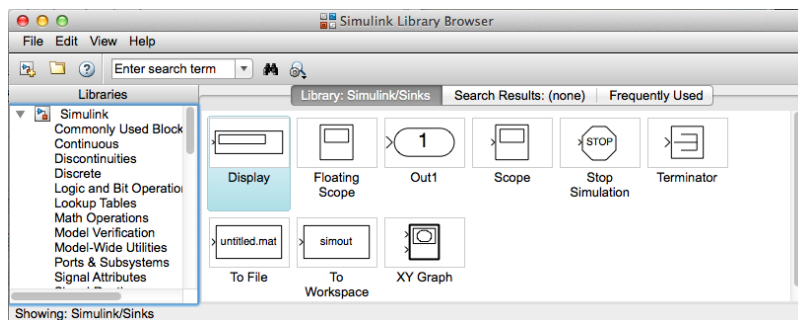
Hình 1.22. Thư viện Commonly Used Block

- Thư viện Math Operations: Là thư viện chứa các khối biểu diễn các hàm toán học.



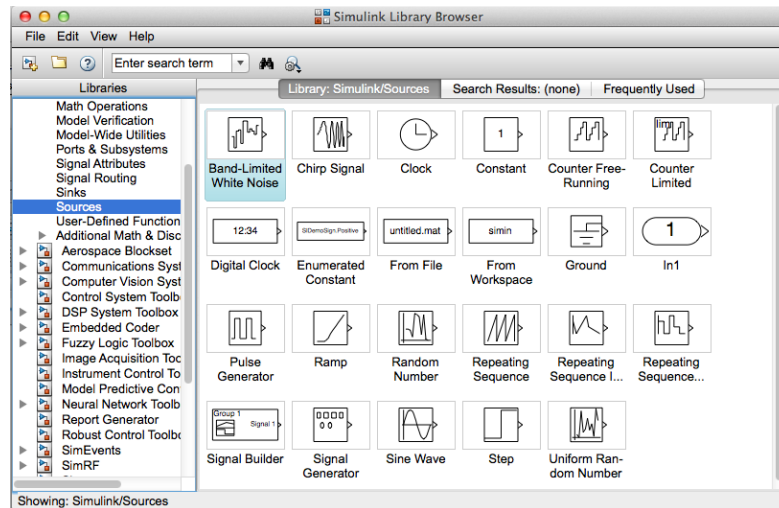
Hình 1.23. Thư viện Math Operations

- Thư viện sinks: Là thư viện chứa khối biểu diễn đầu ra của hệ thống.



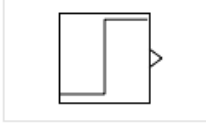
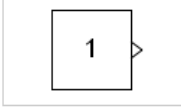
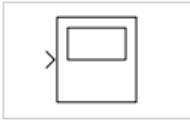

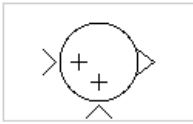
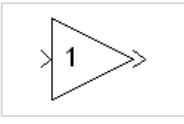
Hình 1.24. Thư viện Sinks

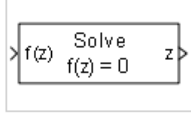
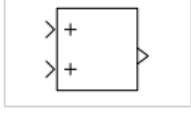
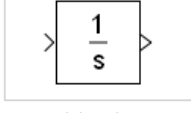
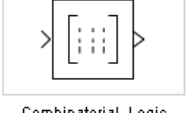
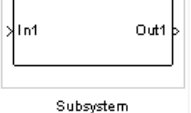

- Thư viện Source: Là thư viện chứa các khối đầu vào của hệ thống.



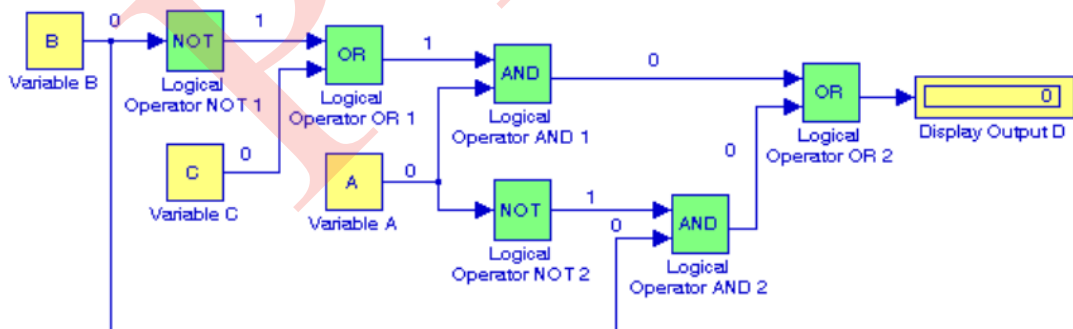
Hình 1.25. Thư viện Source

Ví dụ minh họa chức năng một số khối (vị trí, chức năng, cách cài đặt tham số):

 <p>Step</p>	<p>Khối Step (ở thư viện Simulink \ Sources) có chức năng xuất ra tín hiệu hàm bậc thang. Double click vào khối này để cài đặt các thông số:</p> <ul style="list-style-type: none"> <li>• Step time : khoảng thời gian đầu ra chuyển sang mức Final value kể từ lúc bắt đầu mô phỏng. Cài đặt giá trị này bằng 0.</li> <li>• Initial value : Giá trị ban đầu. Cài đặt bằng 0.</li> <li>• Final value : Giá trị lúc sau. Cài đặt theo giá trị ta muốn tác động tới hệ thống. Nếu là hàm bậc thang đơn vị thì giá trị này bằng 1.</li> <li>• Sample time : thời gian lấy mẫu. Cài đặt bằng 0.</li> </ul>
 <p>Constant</p>	<p>Khối Constant (ở thư viện Simulink \ Sources) có chức năng xuất ra một giá trị không đổi (hằng số).</p>
 <p>Scope</p>	<p>Khối Scope (ở thư viện Simulink \ Sinks) là cửa sổ xem các tín hiệu theo thời gian, tỉ lệ xích của các trục được điều chỉnh tự động để quan sát tín hiệu một cách đầy đủ.</p>
 <p>Display</p>	<p>Khối Scope (ở thư viện Simulink \ Sinks) là cửa sổ xem các tín hiệu dưới dạng số.</p>
 <p>Sum</p>	<p>Khối Sum (ở thư viện Simulink \ Math Operations) là bộ tổng (cộng hay trừ) các tín hiệu, thường dùng để lấy hiệu số của tín hiệu đặt với tín hiệu phản hồi. Double click để thay đổi dấu của bộ tổng.</p>
 <p>Gain</p>	<p>Khối Gain (ở thư viện Simulink \ Math Operations) là bộ tỉ lệ. Tín hiệu sau khi qua khối này sẽ được nhân với giá trị Gain. Double click để thay đổi giá trị độ lợi Gain.</p>

 Algebraic Constraint	Khối Algebraic Constraint (ở thư viện Simulink \ Math Operations) có chức năng giải phương trình.
 Add	Khối Add (ở thư viện Simulink \ Math Operations) có chức năng cộng/trừ các tín hiệu vào.
 Integrator	Khối Integrator (ở thư viện Simulink \ Continuous) có tác dụng lấy tích phân của tín hiệu đầu vào.
 Combinatorial Logic	Khối Combinatorial Logic (ở thư viện Simulink \ Logic and Bit Operations) có tác dụng tạo ra bảng trạng thái.
 Subsystem	Khối Subsystem (ở thư viện Simulink \ Commonly Used Blocks) gom các khối lại tạo thành một hệ thống con, nhằm thu gọn hệ thống.
 Logical Operator	Khối Logical Operator (ở thư viện Simulink \ Logic and Bit Operations) có tác dụng tạo ra cổng logic (AND, OR, NAND, NOR...).

Ví dụ: Xây dựng hệ logic như Hình 1.26, kiểm tra kết quả trong bảng trạng thái và nhận xét?



Hình 1.26. Hệ logic

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

## BÀI TẬP CHƯƠNG 1

**Bài 1-1.** Tạo một vectơ gồm các phần tử chẵn nằm trong khoảng từ 12 đến 35.

**Bài 1-2.** Cho vectơ  $x$ :

- Cộng thêm 13 vào mỗi phần tử của  $x$
- Trừ 2 của các phần tử có chỉ số lẻ
- Tính căn bậc hai của các phần tử có chỉ số chẵn.

**Bài 1-3.** Câu hỏi 1.1: Điền kết quả vào dấu chấm hỏi (?):

```
if n > 1
    m = n+1
else
    m = n - 1
end
```

a)  $n = 7$   $m = ?$

b)  $n = 0$   $m = ?$

c)  $n = -10$   $m = ?$

d)  $n = 1$   $m = ?$

e)  $n = -1$   $m = ?$

**Bài 1-4.** Viết Script file để tạo hàm có đặc điểm sau:

$t(y) = 200$	với $y < 10.000$
$= 200 + 0.1 (y - 10,000)$	với $10.000 \leq y < 20.000$
$= 1,200 + 0.15 (y - 20,000)$	với $20.000 \leq y < 50.000$
$= 5,700 + 0.25 (y - 50,000)$	với $y \geq 50,000$

**Bài 1-5.** Viết script file:

- Nhập hai số  $a, b$  từ bàn phím
- Tính tổng, hiệu hai số và in kết quả ra màn hình

**Bài 1-6.** Viết function file:

- Nhập một số từ bàn phím
- Kiểm tra xem số đó có phải là số nguyên tố không?

**Bài 1-7.** Cho véc tơ  $x = [3 \ 1 \ 5 \ 7 \ 9 \ 2 \ 6]$ . Giải thích ý nghĩa của từng lệnh và kết quả của chúng?

- a)  $x(1:5)$
- b)  $x(1:end)$
- c)  $x(1:end-1)$
- d)  $x(6:-2:1)$
- e)  $x([1 \ 6 \ 2 \ 1 \ 1])$

**Bài 1-8.** Viết function file:

- a) Nhập hai số  $a, b$  từ bàn phím
- b) Kiểm tra và in ra màn hình số lớn

**Bài 1-9.** Viết kết quả của các dòng lệnh sau:

- a) `subs (a+b,a,4)`
- b) `subs(cos(a)+sin(b),{a,b},{sym(1,2)}`
- c) `expand((x-2)*(x-4))`
- d) `factor(x^3-y^3)`
- e) `linspace(1,5,8)`

**Bài 1-10.** Cho mảng  $A = [2 \ 7 \ 9 \ 7 ; 3 \ 1 \ 5 \ 6 ; 8 \ 1 \ 2 \ 5]$ .

Giải thích kết quả của các lệnh sau:

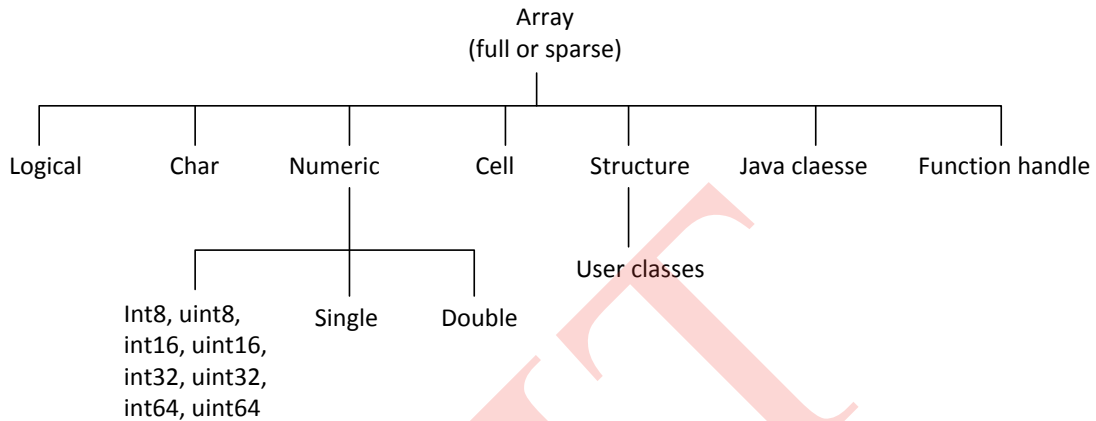
- a)  $A(:, [1 \ 4])$
- b) `reshape(A,2,6)`
- c)  $[A \ A(end,:)]$
- d)  $A(1:3,:)$
- e)  $[A ; A(1:2,:)]$

## CHƯƠNG 2. CẤU TRÚC DỮ LIỆU

### 2.1. GIỚI THIỆU CHUNG

MATLAB sử dụng 15 kiểu (loại) dữ liệu chính. Mỗi một kiểu dữ liệu này đều ở dạng của một ma trận hoặc mảng. Các mảng hoặc ma trận này có kích cỡ tối thiểu là  $0 \times 0$  và có thể phát triển tới mảng n-chiều với kích cỡ tùy ý.

Ngoài ra còn có các kiểu dữ liệu do người dùng định nghĩa, kiểu hướng đối tượng, và kiểu dữ liệu liên quan tới Java.



Hình 2.1. Các kiểu dữ liệu của MATLAB

\* Định dạng kết quả:

Sử dụng lệnh **'format'** cùng các định dạng. Lệnh này chỉ làm thay đổi cách mà kết quả được hiển thị trên màn hình, không làm thay đổi độ chính xác của số hoặc phép tính. Hầu hết các phép tính số học của MATLAB được thực hiện với độ chính xác Double, nghĩa là độ chính xác 16 chữ số sau dấu phẩy thập phân.

Để thực hiện lệnh, từ dấu nhắc của cửa sổ lệnh đánh một trong các lệnh sau:

format short : dấu phẩy thập phân cố định, 5 chữ số

format long : dấu phẩy cố định, 15 chữ số

format short e : ký hiệu khoa học, 5 chữ số

format long e : ký hiệu khoa học, 15 chữ số

format short g : dấu phẩy cố định hoặc động, 5 chữ số

format long g : dấu phẩy cố định hoặc dấu phẩy động, 15 chữ số

format hex : format dạng Hex a (hệ 16)

format '+' : dương (+), âm (-), và ký tự trắng (blank) ứng với 0

format bank : Dollars và cents

format rat : tỷ lệ xấp xỉ integer

Thông thường, 'format short' là dạng mặc định. Khi được gọi lên, một dạng format sẽ có hiệu lực tới khi nó được thay đổi.



Đại số tuyến tính là trái tim và là phần hồn của MATLAB. Trong thực tế thì ban đầu MATLAB là từ viết tắt của “matrix laboratory”. Vì vậy hơn bất kỳ ngôn ngữ nào khác, MATLAB khuyến khích tận dụng mọi khả năng của các mảng, vectơ và ma trận.

**\* Một vài thuật ngữ:**

- **Mảng:** là một tập hợp các số, được gọi là các ‘phần tử’ hay các ‘đầu số’, được biết đến với một hoặc nhiều chỉ số chạy suốt các tập hợp chỉ số. Trong MATLAB, các tập hợp chỉ số luôn là chuỗi số nguyên tố bắt đầu bằng 1.

- **Số chiều** của một mảng là số các chỉ số cần thiết để định nghĩa một phần tử trong mảng. Chẳng hạn mảng 2 chiều sẽ cần 2 chỉ số  $i$  và  $j$  để đặc trưng cho một phần tử của mảng.

- **Kích thước** của mảng là một danh sách các kích thước của các tập hợp chỉ số, ví dụ:

```
>> r = [1 2 3; -1 -2 -7];  
>> size(r)  
ans =  
2 3
```

Nghĩa là kích thước của mảng  $r$  sẽ là  $2 \times 3$  (2 hàng, 3 cột).

- **Ma trận** là một mảng hai chiều (kích thước  $m \times n$  với các quy luật đặt biệt cho phép cộng, nhân và các tính toán khác. Nó đặc trưng cho một sự biến đổi tuyến tính về toán học. Hai chiều của ma trận là hàng và cột ( $m$  hàng và  $n$  cột).

- **Vectơ** là một ma trận mà một chiều chỉ có chỉ số = 1. Cụ thể, một vectơ hàng là một ma trận chỉ có một hàng (kích thước  $1 \times n$ ), còn một vectơ cột là một ma trận chỉ có một cột (kích thước  $m \times 1$ ).

- Mặc dù khái niệm **mảng** tổng quát hơn và ít tính chất toán học hơn một **ma trận**, nhưng hai thuật ngữ này vẫn thường được dùng lẫn lộn với nhau. Hơn nữa, MATLAB đôi khi không có một sự phân biệt chính thức nào, thậm chí là giữa một đại lượng vô hướng và một ma trận kích thước  $1 \times 1$ .

- Các lệnh có thể được sắp xếp theo sự phân biệt giữa mảng/ma trận, nhưng MATLAB thường cho phép ta sử dụng chúng lẫn lộn một cách thoải mái. Ý tưởng ở đây (và bất cứ chỗ nào khác) là MATLAB muốn giữ cho ngôn ngữ của mình đơn giản và tự nhiên, để ta có thể tự mình tránh khỏi các rắc rối.

- Các phần tử đơn lẻ trong ma trận có thể được tiếp cận và sửa đổi bằng cách sử dụng chỉ số phần tử (subscripting). Trong MATLAB, phần tử thứ  $i$  của vectơ  $V$  được biểu diễn bằng ký hiệu  $V(i)$ , chỉ số được viết trong ngoặc đơn. Ví dụ:

```
>> V = [10 20 30]  
V =  
10 20 30  
>> V(2)  
ans =  
20  
>> V(2)=50
```

```
V =  
10 50 30
```

## 2.2. MA TRẬN, VÉCTOR VÀ ĐA THỨC

### 2.2.1. Véc tơ

#### 2.2.1.1. Véc tơ hàng

Véc tơ hàng là chuỗi các số được phân cách bởi *dấu phẩy* hoặc *khoảng trống*. Số lượng các phần tử được gọi là ‘chiều dài’ của véc tơ. Cú pháp cơ bản để nhập 1 véc tơ là một chuỗi các giá trị được bao trong cặp ngoặc vuông [ ]. Ví dụ:

```
>> v = [ 1 3 sqrt(5) ]  
v =  
1.0000 3.0000 2.2361  
>> length(v)  
ans =  
3
```

hoặc cách khai báo khác cho kết quả tương tự, sử dụng các *dấu phẩy* (,)

```
>> v = [1, 3, sqrt(5)]  
v =  
1.0000 3.0000 2.2361
```

Trong ví dụ đầu tiên, các khoảng trống được dùng để phân cách các phần tử của véc tơ. Khoảng trống (space) rất quan trọng trong khi khai báo véc tơ, điều này có thể minh họa bằng sự khác biệt nhỏ giữa hai dòng lệnh dưới đây:

```
>> v2 = [3+4 5]  
v2 =  
7 5  
>> v3 = [3 +4 5]  
v3 =  
3 4 5
```

Như đã đề cập ở trên, chúng ta có thể xem hoặc thay đổi giá trị của những phần tử riêng biệt của véc tơ.

```
>> A = [1, -2, 3];  
A =  
1 -2 3  
>> A(2) = 4  
A =  
1 4 3
```

### 2.2.1.2. Vector cột

Véc tơ cột có cấu tạo tương tự như véc tơ hàng. Khi định nghĩa véc tơ cột, các phần tử được phân cách nhau bởi **dấu chấm phẩy ‘;’** hoặc bởi một dòng mới. Ví dụ:

```
>> c = [ 1; 3; sqrt(5) ]
```

```
c =
```

```
1.0000
```

```
3.0000
```

```
2.2361
```

```
>> c2 = [3 ↵
```

```
4 ↵
```

```
5]↵
```

```
c2 =
```

```
3
```

```
4
```

```
5
```

```
>> c3 = 2*c2 - 3*c2
```

```
c3 =
```

```
-7.0000
```

```
-6.0000
```

```
-10.5279
```

Ví dụ trên cho thấy các véc tơ cột có thể được cộng hoặc trừ với nhau nếu chúng có cùng chiều dài.

### 2.2.1.3. Toán tử hai chấm ( : )

- Toán tử (:) dùng để tạo ra véc tơ hàng một cách nhanh chóng:

```
>> x = 1:4
```

```
x =
```

```
1 2 3 4
```

```
>> y = 3:7
```

```
y =
```

```
3 4 5 6 7
```

```
>> z = 1:-1
```

```
z =
```

```
Empty matrix: 1-by-0
```

Một cách tổng quát thì cú pháp  $a : b : c$  sẽ tạo ra một véc tơ với các phần tử bắt đầu từ giá trị của  $a$ , tăng dần với bước tăng bằng giá trị của  $b$ , cho tới khi đạt tới giá trị của  $c$  (sẽ không tạo ra một giá trị vượt quá  $c$ ). Điều này giải thích vì sao  $1:-1$  tạo ra một véc tơ rỗng  $[]$ .

```
>> 0.32:0.1:0.6
```

```
ans =
```

```
0.3200 0.4200 0.5200
>> -1.4:-0.3:-2
ans =
-1.4000 -1.7000 -2.0000
```

- Toán tử ':' còn được dùng để trích xuất một phần của véc-tơ. Ví dụ

```
>> r = [1:2:6, -1:-2:-7]
r =
1 3 5 -1 -3 -5 -7
```

để trích ra các phần tử từ thứ 3 đến thứ 6 ta có thể dùng lệnh:

```
>> r(3:6)
ans =
5 -1 -3 -5
```

hoặc để trích các phần tử theo một quy luật, chẳng hạn:

```
>> r(1:2:7)
ans =
1 5 -3 -7
```

Ngoài ra, dấu hai chấm (:) còn được dùng trong các vòng lặp **for**, ví dụ:

```
>> A=[];
>> for j = 1:2:10
A= [A j];
end
A=
1      3      5      7      9
```

### 2.2.2. Ma trận

Ma trận là dạng cấu trúc dữ liệu cơ bản của MATLAB. Từ dấu nhắc của cửa sổ nhập lệnh, đánh lệnh **help elmat** hoặc **help matfun** để có một danh sách các lệnh và hàm làm việc với ma trận trong MATLAB.

Cú pháp của việc định nghĩa và khởi tạo ma trận như sau: *các khoảng trống* (hoặc *dấu phẩy*) phân cách các phần tử trong một hàng, và các *dấu chấm phẩy* là ký hiệu cho biết bắt đầu một hàng mới sau đó.

Ví dụ:

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

MATLAB sẽ đưa ra kết quả:

```
A =
2 -1 0 0
1 1 2 3
-1 4 0 5
```

Và biến A bây giờ chứa một ma trận 3x4.

Các phần tử đơn lẻ của một ma trận có thể được tiếp cận và chỉnh sửa theo cùng một cách như với các vectơ, đó là cung cấp các *chỉ số hàng và cột*. Ví dụ lệnh:

$A(3,2) = 0$  sẽ thay thế giá trị phần tử ở hàng 3, cột 2 của ma trận A thành 0.

\* Một số phép tính với ma trận

`zeros(n,m)` : tạo ma trận kích thước  $n \times m$  với tất cả các phần tử đều bằng 0.

`ones(n,m)` : tạo ma trận kích thước  $n \times m$  với tất cả các phần tử bằng 1.

`eye(n)` : tạo ma trận đơn vị kích thước  $n \times n$ .

`rand(n,m)` : ma trận (n.m) các phần tử từ 0 đến 1

`diag(V,k)` : nếu V là một vectơ thì sẽ tạo ma trận đường chéo

$A'$  : Phép chuyển vị của ma trận A

`sum`: Tính tổng các phần tử trên từng cột của ma trận  $m \times n$  thành ma trận  $1 \times n$ .

`diag`: Lấy các phần tử đường chéo của ma trận

`det`: tính định thức ma trận

`rank`: tính hạng của ma trận

`inv`: tính ma trận nghịch đảo

Cộng trừ 2 ma trận:  $A(n.m) \pm B(n.m) = C(n.m)$

Nhân 2 ma trận:  $A(n.m) * B(m.k) = C(n.k)$

Nhân mảng:  $C = A .* B$  %  $(C(i,j) = A(i,j) * B(i,j))$

Chia trái mảng:  $C = A ./ B$  %  $(C(i,j) = B(i,j) / A(i,j))$

Chia phải mảng:  $C = A ./ B$  %  $(C(i,j) = A(i,j) / B(i,j))$

Chia trái ma trận:  $C = A \setminus B = \text{inv}(A) * B$  % (pt:  $AX = B$ )

Chia phải ma trận:  $C = A ./ B = B * \text{inv}(A)$  % (pt:  $XA = B$ )

Lũy thừa ma trận:  $A ^ P$

Để khởi tạo một ma trận vuông đặc biệt, bạn có thể sử dụng dạng ngắn **zeros(n)**, lệnh này đã ngầm định rằng số hàng và số cột của ma trận là bằng nhau.

## 2.2.3. Các phép toán trên vectơ và ma trận

### 2.2.3.1. Các phép toán số học

Chúng ta có thể tiến hành một số phép toán số học nhất định (cộng, trừ) với các vectơ hoặc ma trận cùng kích thước. MATLAB sẽ báo lỗi khi ta thực hiện các phép toán này với các ma trận có kích thước (chiều dài) khác nhau. Ví dụ:

```
>>v1 = [1 2 3]
```

```
>>v2 = [ 4 5 6]
```

```
>> v1+v2
```

```
ans =  
5 7 9
```

Một véc tơ cũng có thể nhân được với một đại lượng vô hướng (một số), thao tác được MATLAB tiến hành với từng phần tử. Ví dụ:

```
>>v1*2  
ans =  
2 4 6
```

Để tiến hành các tính toán cùng loại (tính toán với từng phần tử): nhân, chia và lũy thừa, MATLAB đưa ra các toán tử ".\*", "./" và ".^". Ví dụ:

```
>> a = [1 2 3; 2 4 1; 2 7 2];  
>> b = [1 3 1; 2 7 1; 1 2 3];
```

Nhân hai ma trận  $a$  và  $b$  theo cách thông thường:

```
>> a*b  
ans =  
8 23 12  
11 36 9  
18 59 15
```

Nhân từng phần tử của ma trận  $a$  với các phần tử tương ứng của ma trận  $b$  (sử dụng dấu chấm):

```
>> a.*b  
ans =  
1 6 3  
4 28 1  
2 14 6
```

Tương tự như thế đối với các phép toán chia (./) và lũy thừa (.^). Ví dụ:

```
>>c = [1 3; 5 2];  
>>c^2 % tương đương c*c  
ans =  
16 9  
15 19  
>>c.^2 % Bình phương các phần tử của c.  
ans =  
1 9  
25 4
```

Tất cả các hàm số học dựng sẵn của MATLAB được thiết kế để hoạt động với các véc tơ (và ma trận), vì vậy chúng ta có thể xây dựng các diễn giải đại số hoạt động với từng phần tử của véc tơ.

Ví dụ: đoạn mã lệnh dưới đây tính toán giá trị biểu thức:  $s = x^2 + \sqrt{xy} - 3y$  theo từng phần tử. Tính với mỗi một phần tử trong vectơ  $x$  và  $y$ :

```
>> x = [1 2 3]; y = [4 5 6];
>> s = x.^2 + sqrt(x.*y) - 3*y
s =
    -9.0000    -7.8377    -4.7574
```

Lưu ý các phép tính của các đại lượng vô hướng trên các vectơ khác như thế nào với cách làm việc phần tử với phần tử, ví dụ:  $3*y$  rõ ràng là nhân số vô hướng với vectơ, trong khi  $x*y$  thì khác, vì vậy ở đây ta cần phải sử dụng  $x.*y$

Chú ý: Các phép cộng và trừ phần tử với phần tử lẽ ra cũng phải sử dụng  $."+$  và  $.-$ , tuy nhiên trong ví dụ này thì không cần thiết.

### 2.2.3.2. Ghép các vectơ và ma trận

Có thể tạo ra một ma trận từ những ma trận có trước nếu như kích thước của chúng tương thích với nhau, ví dụ:

```
>> x = [1 2 3], y = [3 4];
>> z = [2*x, -y]
z =
     2     4     6    -3    -4
>> sort(z)
ans =
    -4    -3     2     4     6
```

Lưu ý rằng câu lệnh cuối cùng (sort) để sắp xếp các phần tử của vectơ theo chiều tăng dần. Ta cũng có thể sử dụng các lệnh *cat*, *vertcat*, *horzcat* để ghép nối các vectơ.

```
>> a = [1 2 3; 2 4 1];
>> b = [1 3 1; 2 7 1];
>> c = [a b]
c =
     1     2     3     1     3     1
     2     4     1     2     7     1
>> c = [a; b]
c =
     1     2     3
     2     4     1
     1     3     1
     2     7     1
```

### 2.2.3.3. Các lệnh cho thông tin về ma trận (vectơ):

`size` : kích thước theo mỗi chiều

length : kích thước của chiều dài nhất (đặc biệt là cho véctor)  
 ndims : số chiều  
 find : các chỉ số của các phần tử khác 0

#### 2.2.3.4. Chuyển vị ma trận:

Ta có thể chuyển đổi một véctor hàng thành một véctor cột (và ngược lại) bằng một quá trình gọi là ‘chuyển vị’ – ký hiệu bằng ký tự " ' " (dấu nháy đơn). Hãy xem các ví dụ sau:

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

```
a =
```

```
1     2     3
2     4     1
```

```
>> a'
```

```
ans =
```

```
1     2
2     4
3     1
```

```
>> b = [1 2 3]
```

```
b =
```

```
1     2     3
```

```
>>b'
```

```
ans =
```

```
1
2
3
```

```
>> c=[a' b']
```

```
c =
```

```
1     2     1
2     4     2
3     1     3
```

#### 2.2.4. Đa thức

##### 2.2.4.1. Biểu diễn các đa thức

MATLAB biểu diễn các đa thức bằng các véctor chứa hệ số sắp xếp theo thứ tự giảm của số mũ. Ví dụ, xét phương trình sau:

$$p(x) = x^3 - 2x - 5$$

Để nhập đa thức này vào MATLAB, sử dụng:

```
>>p = [1 0 -2 -5];
```



#### 2.2.4.2. Tính toán đa thức

Hàm `polyval` tính toán một đa thức với một giá trị cụ thể. Để tính  $p$  với  $s=5$ , sử dụng

```
>> polyval(p, 5)
ans =
    110
```

Hàm này cũng tính giá trị của đa thức với trường hợp ma trận. Trong trường hợp này  $p(x) = x^3 - 2x - 5$  sẽ trở thành  $p(X) = X^3 - 2X - 5I$ , với  $X$  là một ma trận vuông và  $I$  là ma trận đơn vị. Ví dụ, tạo ma trận vuông  $X$  và tính giá trị của đa thức  $p$  với  $X$ .

```
>> X = [2 4 5; -1 0 3; 7 1 5];
>> Y = polyvalm(p, X)
Y =
    377    179    439
    111     81    136
    490    253    639
```

#### 2.2.4.3. Tính nghiệm của đa thức

Hàm `roots` sẽ tính toán các nghiệm của một đa thức:

```
>> r = roots(p)
r =
    2.0946
   -1.0473 + 1.1359i
   -1.0473 - 1.1359i
```

Theo quy ước, MATLAB lưu trữ các nghiệm theo các vectơ cột. Hàm `poly` sẽ trả lại các hệ số của đa thức:

```
>> p2 = poly(r)
p2 =
     1    8.8818e-16    -2    -5
```

Hai hàm `poly` và `roots` là ngược nhau.

#### 2.2.4.4. Tính đạo hàm

Hàm `polyder` tính đạo hàm của một đa thức. Để tính đạo hàm của đa thức:

$p = [1 \ 0 \ -2 \ -5]$  ta thực hiện như sau:

```
>> q = polyder(p)
q =
     3     0    -2
```

Hàm `polyder` cũng tính đạo hàm của tích hoặc thương của hai đa thức. Ví dụ, tạo hai đa thức  $a$  và  $b$ :

```
>> a = [1 3 5];
```

```
>>b = [2 4 6];
```

Tính đạo hàm của tích  $a*b$  bằng hàm `polyder` với một lựa chọn đầu ra:

```
>>c = polyder(a,b)
```

```
c =
```

```
8      30      56      38
```

Tính đạo hàm của thương  $a/b$  bằng hàm `polyder` với hai đầu ra:

```
>>[q,d] = polyder(a,b)
```

```
q =
```

```
-2      -8      -2
```

```
d =
```

```
4      16      40      48      36
```

$q/d$  là kết quả của phép toán.

#### 2.2.4.5. Phân tích phân số thành các thành phần

Hàm **residue** tìm các thành phần tách phân số của thương hai đa thức. Điều này rất hữu ích để ứng dụng trong các hệ thống mô tả hàm truyền đạt. Ví dụ  $b$  và  $a$ , nếu không có nghiệm kép,

$$\frac{b(s)}{a(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \dots + \frac{r_n}{s-p_n} + k_s$$

Trong đó  $r$  là một vectơ cột,  $p$  là vectơ cột chứa các điểm cực, và  $k$  là vectơ hàng chứa số hạng. Xét hàm truyền đạt sau:

$$\frac{-4s+8}{s^2+6s+8}$$

```
>>b = [-4 8];
```

```
>>a = [1 6 8];
```

```
>>[r,p,k] = residue(b,a)
```

```
r =
```

```
-12
```

```
8
```

```
p =
```

```
-4
```

```
-2
```

```
k =
```

```
[]
```

Nếu cho 3 đầu vào ( $r$ ,  $p$ , và  $k$ ), thì hàm `residue` lại chuyển lại dạng đa thức:

```
>>[b2,a2] = residue(r,p,k)
```

```
b2 =
```

```
-4
```

```
8
```

```
a2 =
```

#### 2.2.4.6. Các đa thức riêng

Hàm **poly** cũng tính các hệ số của đa thức riêng của một ma trận:

```
>>A = [1.2 3 -0.9; 5 1.75 6; 9 0 1];
>>poly(A)
ans =
    1.0000    -3.9500    -1.8500   -163.2750
```

Nghiệm của đa thức này, tính bằng hàm **roots**, là nghiệm riêng, hay giá trị riêng, của ma trận A. (sử dụng hàm **eig** để tính trực tiếp các giá trị riêng của ma trận).

#### 2.2.4.7. Đa thức xấp xỉ

Hàm **polyfit** tìm các hệ số của một đa thức xấp xỉ với một tập các dữ liệu theo phương pháp bình phương tối thiểu:

```
>>p = polyfit(x,y,n)
```

Trong đó **x** và **y** là các vectơ chứa dữ liệu **x** và **y** cần xấp xỉ, và **n** là bậc của đa thức cần tìm. Ví dụ, xét tập dữ liệu.

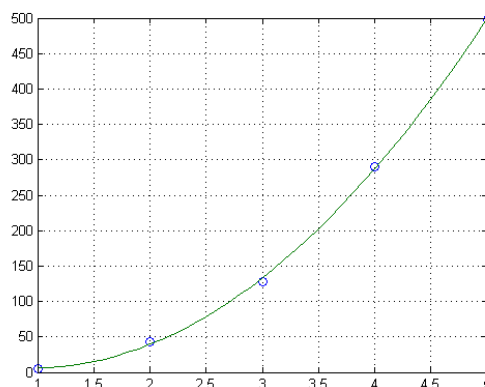
```
>>x = [1 2 3 4 5];
>>y = [5.5 43.1 128 290.7 498.4];
```

Đa thức bậc ba xấp xỉ với dữ liệu là:

```
>>p = polyfit(x,y,3)
p =
   -0.1917   31.5821  -60.3262   35.3400
```

Tính toán các giá trị của đa thức xấp xỉ và vẽ các giá trị đó so sánh với giá trị thật:

```
>>x2 = 1:.1:5;
>>y2 = polyval(p,x2);
>>plot(x,y,'o',x2,y2);
>>grid on
```



Hình 2.2. So sánh sai khác giữa đa thức xấp xỉ và các giá trị thật

## 2.2.5. Mảng

### 2.2.5.1. Mảng một chiều và vectơ

Như đã đề cập ở trên khái niệm **mảng** tổng quát hơn và ít tính chất toán học hơn một **ma trận**, nhưng hai thuật ngữ này vẫn thường được dùng lẫn với nhau.

Các mảng *1 chiều* chính là các *vectơ*.

Trong MATLAB các kiểu dữ liệu mảng có các tính chất:

- Số chiều của mảng (ta có thể lập những mảng  $n$  chiều)
- Mỗi phần tử của mảng này có cùng kiểu (nguyên, thực, kí tự,...)
- Mảng là một cấu trúc tiết kiệm không gian nhớ và thuận lợi cho tính toán.
- Những ma trận lớn có nhiều phần tử 0 (ma trận thưa) có thể được lập thành bởi lệnh `sparse`. Ví dụ:

```
>>a = zeros(100); a(1,3) = 10; a(21,5) = pi; b = sparse(a);
```

Các cách làm việc với mảng một chiều về cơ bản cũng giống như với vectơ đã trình bày ở mục 2.2.1. Ví dụ như:

+ Muốn tính số phần tử trong mảng một chiều  $u$  dùng lệnh: `length(u)`

+ Muốn truy xuất phần tử thứ  $n$  trong mảng: `ten_mang(n)`

+ Cách tạo mảng:

\* `Ten_mang = [m1,m2,m3,...]` % Mảng kiểu dòng

\* `Ten_mang = [m1;m2;m3;...]` % Mảng kiểu cột

\* `Ten_mang = [m1 m2 m3...]` % Vectơ dòng

\* `Ten_mang = [a:Δ:b]` % Mảng kiểu cách đều

Ví dụ:

```
>>x=[0:2:9]
```

```
x =
```

```
0      2      4      6      8
```

\* **Hàm `linspace(x1,x2,n)`**

Tạo một vectơ dòng có các phần tử cách đều, với  $x_1$  là cận dưới,  $x_2$  là cận trên và  $n$  là số điểm của dãy (kể cả  $x_1$  và  $x_2$ ). Nếu  $n$  bị bỏ qua, giá trị mặc định của nó là 100.

Ví dụ:

```
>>z=linspace(2,8,7)
```

```
z =
```

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

\* **Hàm `logspace(a,b,n)`**

Sẽ tạo một vectơ có  $n$  phần tử cách quãng theo thang logarit, phần tử đầu là  $10^a$  và cuối là  $10^b$ , các phần tử giữa có dạng  $10^x$  trong đó  $x$  là 1 điểm cách đều giữa  $a$  và  $b$ . Có  $(n-2)$  điểm như vậy. Nếu  $n$  bị bỏ qua trong câu lệnh thì số mặc định của  $n$  là 50.

Ví dụ:

```
>>u = logspace(-1,1,4)
u =
    0.1000    0.4642    2.1544   10.0000
```

### 2.2.5.2. Mảng nhiều chiều

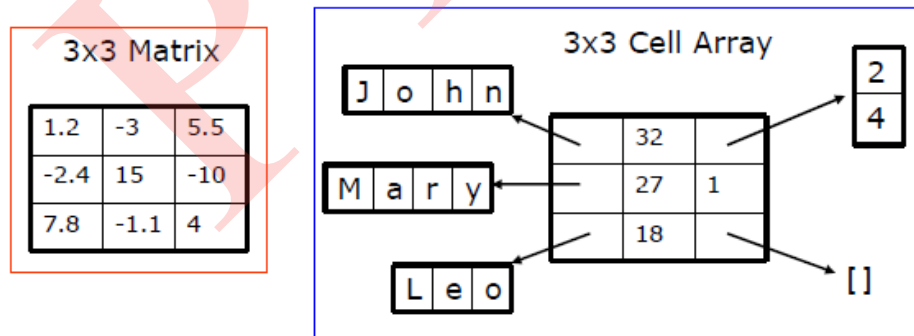
MATLAB có thể làm việc với mảng nhiều chiều ( $n$  chiều) nhằm tiết kiệm không gian bộ nhớ cũng như thuận lợi cho việc tính toán. Các *mảng 2 chiều* chính là các *ma trận*, kích thước của ma trận gồm một bộ số  $m \times n$  với  $m$  là số hàng và  $n$  là số cột của ma trận. Các phép toán đối với ma trận đã được trình bày trong mục 2.2.2.

Ở một số trường hợp, khi cần có các cấu trúc dữ liệu phức tạp hơn, ta có thể sử dụng một số cấu trúc dữ liệu khác như sau:

- **cell array**: giống với kiểu mảng thường, nhưng các phần tử trong mảng *không cần phải có chung kiểu*.
- **structs**: cấu trúc dữ liệu trừu tượng gồm nhiều kiểu dữ liệu hợp lại. Thường dùng trong lập trình hướng đối tượng.

#### Cell

- Cell giống như một ma trận, nhưng mỗi phần tử của ma trận này có thể chứa một loại dữ liệu riêng (có thể là một ma trận khác).
- Ví dụ: Một cell có thể chứa tên người, tuổi, tuổi của những đứa con của họ. Trong khi đó nếu sử dụng matrix, ta cần 3 biến ma trận, mỗi ma trận có 3 phần tử để lưu những dữ liệu trên, như Hình 2.3.



Hình 2.3. Cấu trúc dữ liệu Cell

- Cách khai báo và sử dụng một cell:
  - + Để tạo một cell, dùng từ khóa **cell** và chỉ rõ kích thước của cell.

```
>> a = cell(3, 10) % a sẽ là một cell với 3 hàng, 10 cột
```
  - + Hoặc có thể khai báo bằng cách liệt kê các phần tử của cell trong dấu ngoặc { }

```
>> c={'hello world',[1 5 6 2],rand(3,2)};
% c là một cell có 1 hàng, 3 cột
```
  - + Mỗi phần tử của cell có thể là mọi loại dữ liệu.

+ Để truy cập đến từng phần tử của cell, dùng dấu { }

```
>> a{1,1}=[1 3 4 -10];  
>> a{2,1}='hello world 2';  
>> a{1,2}=c{3};
```

### Struct:

- **struct** cho phép bạn và lưu một tập các biến liên quan (trong các field) và giữ chúng trong cùng 1 biến (struct là kiểu dữ liệu giống trong C)
- Để khởi tạo một struct:

```
>> s=struct([]); % size(s) là 1x1.
```

Việc khởi tạo các biến trong struct là không bắt buộc, tuy nhiên với các struct lớn bạn nên khởi tạo trước khi sử dụng.

- Để thêm một trường (**field**) trong struct:

```
>> s.name = 'Jack Bauer';  
>> s.scores = [95 98 67];  
>> s.year = 'G3';
```

- field có thể là matrix, cell, hoặc là một struct khác

- Để xem thêm về struct, gõ **doc struct**

- Ví dụ khởi tạo và sử dụng struct:

```
>> ppl=struct('name',{'John','Mary','Leo'},'age',{32,27,18},'childAge',{[2;4],1,[]});
```

% ppl là một mảng gồm 3 struct, mỗi struct có 3 field dữ liệu

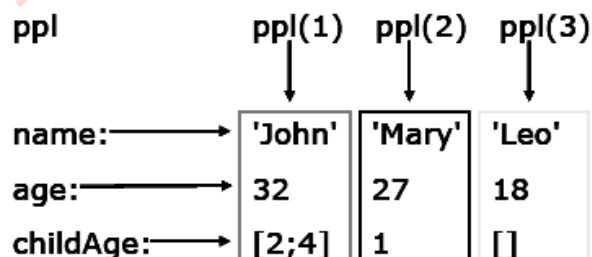
% là name, age và childAge

% chú ý rằng các cell ở khởi tạo trên phải có cùng kích thước

```
>> person = ppl(2) % biến person giờ là 1 struct
```

```
>> person.name % trả về Mary
```

```
>> ppl(1).age % trả về 32
```



- Truy cập vào struct

- với struct, để truy cập vào dữ liệu cần truy cập thông qua tên field

```
>> stu = s.name;
```

```
>> scor = s.scores;
```

- Để truy cập một mảng  $n \times 1$  struct, dùng chỉ số

```
>> person = ppl(2); % person là 1 struct với các field name,
% age và Childage
>> personName=ppl(2).name;
>> a=[ppl.age];
```

## 2.3. MỘT SỐ KIỂU DỮ LIỆU KHÁC

### 2.3.1. Số thực

Trong MATLAB, tất cả các số thực được lưu với độ chính xác **double**, không giống các ngôn ngữ lập trình khác như C hay Fortran khi chỉ có một loại riêng biệt float hay real\*8 cho các số thực với độ chính xác single.

Độ chính xác "double" hay "double precision" là định dạng nhị phân chiếm 64 bit (8 byte) và phần định trị của nó có độ chính xác 53 bit (tương đương với khoảng 16 chữ số thập phân).

Dạng mô tả số thực theo dấu phẩy động cực kỳ hiệu quả cho việc nhập các số rất lớn hoặc rất bé. Ta dùng ký hiệu 'e' cho dấu phẩy động, chẳng hạn -1.23456e-7 là biểu diễn của  $-1.23456 \times 10^{-7}$ ; và 8.76e+12 là số  $8.76 \times 10^{12}$ .

Ví dụ:

```
>> 1.23e-2
ans =
    0.0123
```

### 2.3.2. Số phức

Một trong các ưu thế của MATLAB là làm việc với số phức. Số phức của MATLAB được định nghĩa theo nhiều cách.

\* Cách 1 : Chèn ký tự *i* hoặc *j* vào phần ảo.

```
>> c1=1+3i
c1 =
1.0000 + 3.0000i
>> c1=2-4j
c1 =
2.0000 - 4.0000i
```

\* Cách 2 : Dùng căn bậc hai của một số âm.

```
>> c2=3+sqrt(-1)
c2 =
3.0000 + 1.0000i
>> c2=4-sqrt(-4)
c2 =
4.0000 - 2.0000i
```

\* Cách 3 : Dùng biểu thức \*i hoặc \*j

```
>> c3=2-sin(1)*j
c3 =
2.0000 - 0.8415i
>> c3=3+cos(1)*i
c3 =
3.0000 + 0.5403i
>> c3=4+2*i
c3 =
4.0000 + 2.0000i
>> c3=4+(6/3)*i
c3 =
4.0000 + 2.0000i
```

\* Các phép toán đối với số phức đều thao tác tương tự như số thực.

```
>> a=2+3i
a =
2.0000 + 3.0000i
>> b=1+4i
b =
1.0000 + 4.0000i
>> a+b
ans =
3.0000 + 7.0000i
>> a-b
ans =
1.0000 - 1.0000i
>> a/b
ans =
0.8235 - 0.2941i
>> a*b
ans =
-10.0000 +11.0000
```

\* Có thể biểu diễn số phức ở tọa độ cực (độ lớn và góc)

$$M\angle\theta = Me^{i\theta} = a + ib$$

Ở trên số phức biểu diễn bằng độ lớn M và góc  $\theta$ , quan hệ giữa các đại lượng này và phần thực, phần ảo được biểu diễn dưới dạng đại số:

M: dùng hàm **abs** để tính độ lớn,  $M = \text{abs}(\text{so phuc})$ .

$\theta = \arctan \frac{b}{a}$  : dùng hàm **angle** tính góc,  $\theta = \text{angle}(\text{so phuc})$ .



$$a = M \cos \theta; b = M \sin \theta$$

Ví dụ:

```
>> c=1+2i
c =
1.0000 + 2.0000i
>> M=abs(c)
M =
3.1623
>> goc=angle(c) % góc θ tính bằng radian
goc =
1.1071 % radian
>> goc_do = goc*180/pi % chuyển góc θ sang độ
goc_do =
63.4349 % góc tính bằng độ
```

**Chú ý,** nếu một vectơ hoặc ma trận chứa dữ liệu có giá trị phức, thì phép chuyển vị sẽ tạo giá liên hợp phức. Để lấy chuyển vị một vectơ/ ma trận chứa giá trị phức mà không muốn lấy liên hợp ta dùng ký hiệu (.''). Ví dụ như sau:

```
>> x=[1-i 3+i 2+3*i 4-2*i]; %1X4
>> x'
ans =
1.0000 + 1.0000i
3.0000 - 1.0000i
2.0000 - 3.0000i
4.0000 + 2.0000i
>> x=x.'; %4X1
x =
1.0000 - 1.0000i
3.0000 + 1.0000i
2.0000 + 3.0000i
4.0000 - 2.0000i
```

### 2.3.3. Chuỗi

Chuỗi là một mảng tập hợp của các ký tự trong bảng mã ASCII. Chuỗi được viết trong hai dấu nháy đơn ('...'), một dấu đầu và một dấu cuối của chuỗi.

Ví dụ: t = 'ABCD' ; q = 'Ha noi'

\* Hàm **double** cho mã ASCII của chuỗi:

```
>> double(t)
ans =
65 66 67 68
>> double(q)
```

```
ans =  
72 97 32 110 111 105
```

\* Hàm **char** chuyển lại thành chuỗi:

```
>> char(q)  
ans =  
Ha noi
```

Việc truy xuất các ký tự trong chuỗi cũng theo cách thức của vectơ, tức là theo chỉ số nằm trong dấu ngoặc đơn. Ví dụ:

```
>>t(2)  
ans =  
B  
>>t(2) = 'e'  
t =  
AeCD
```

## CHƯƠNG 3. MỘT SỐ CÔNG CỤ NÂNG CAO

### 3.1. GIỚI THIỆU CHUNG

MATLAB cung cấp sẵn khá nhiều các công cụ hữu ích (gọi là các Toolbox), hầu hết các lĩnh vực kỹ thuật có trong MATLAB đều có các toolbox riêng. Trong chương này sẽ đề cập đến 3 toolbox điển hình liên quan đến các tính toán toán học, xử lý tín hiệu và hệ thống điều khiển tự động. Phần cuối chương là một số kỹ thuật thực hiện giao diện người dùng GUI.

### 3.2. TOOLBOX

#### 3.2.1. Signal Processing Toolbox

Toolbox xử lý tín hiệu cung cấp các thuật toán theo chuẩn công nghiệp và các ứng dụng cho xử lý tín hiệu tương tự và xử lý tín hiệu số (DSP). Ta có thể sử dụng toolbox để mô phỏng các tín hiệu trong miền thời gian và miền tần số, tính toán các phép biến đổi Fourier (FFT) cho các phép phân tích phổ, thiết kế các bộ lọc FIR, IIR, và thực hiện các kỹ thuật như phép chập, điều chế, tái lấy mẫu và các kỹ thuật xử lý tín hiệu khác. Các thuật toán trong toolbox có thể được dùng làm cơ sở để phát triển các thuật toán cho người sử dụng vào việc xử lý âm thanh và tiếng nói hoặc thông tin vô tuyến băng tần gốc.

Các tính năng chính của Signal Processing Toolbox

- Các mô hình hệ thống tuyến tính và hệ thống tín hiệu.
- Biến đổi tín hiệu hiệu, bao gồm: biến đổi Fourier nhanh (FFT), biến đổi Fourier rời rạc (DFT).
- Chế độ quá độ, chế độ xung, và các hàm đánh giá mức trạng thái cho cá tín hiệu hai mức.
- Các hàm đo lường tín hiệu thống kê và cửa sổ tín hiệu.
- Các thuật toán đánh giá mật độ phổ công suất.
- Các phương pháp phân tích, thiết kế và xây dựng các bộ lọc số FIR và IIR.
- Các phương pháp thiết kế bộ lọc tương tự, như bộ lọc Butterworth, Chebyshev và Bessel.
- Mô hình thông số chuỗi thời gian và dự đoán tuyến tính.

##### 3.2.1.1. Tạo tín hiệu

###### a) Vectơ thời gian

Để tạo một tín hiệu tương tự trong MATLAB, cần có một vectơ thời gian để xác định các thời điểm lấy mẫu tín hiệu. Tín hiệu tạo ra sẽ là một vectơ mà mỗi phần tử của nó chính là giá trị của mẫu tín hiệu được lấy ở thời điểm xác định bởi phần tử tương ứng của vectơ thời gian.

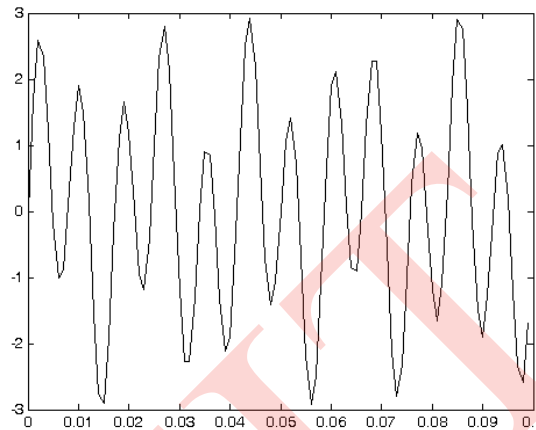
Ví dụ, để tạo tín hiệu  $y = \sin(100\pi t) + 2\sin(240\pi t)$ , ta lấy mẫu tín hiệu tại các thời điểm cách nhau 0,001s và ta có vectơ thời gian  $t$  và vectơ biểu diễn tín hiệu  $y$  như sau:

```
>> t = (0:0.001:1)';
>> y = sin(2*pi*50*t) + 2*sin(2*pi*120*t);
```

Vectơ  $t$  có 1001 phần tử từ 0 đến 1 với các bước cách nhau 0,001, và  $y$  cũng sẽ có 1001 phần tử tương ứng.

Dùng hàm **plot** để vẽ 100 mẫu đầu tiên của  $y$  như trong Hình 3.1.

```
>> plot(t(1:100), y(1:100))
```



**Hình 3.1. Tạo vectơ thời gian và vectơ tín hiệu**

Với phương pháp trên, ta có thể khởi tạo bất kỳ tín hiệu nào ta muốn, chỉ cần xác định biểu thức thời gian của nó. Sau đây là một số tín hiệu đặc biệt.

**b) Các hàm xung đơn vị, hàm nhảy bậc và tín hiệu dốc**

```
>> t = (0:0.001:1)'; % vectơ thời gian
>> y = [1; zeros(99,1)]; % hàm xung đơn vị
>> y = ones(100,1); % hàm nhảy bậc (với điều kiện đầu là 0)
>> y = t; % hàm dốc
```

**c) Các hàm tạo tín hiệu tuần hoàn**

```
>> fs = 10000; % tần số lấy mẫu
>> t = 0:1/fs:1.5; % vectơ thời gian
>> x = sin(2*pi*50*t); % Tín hiệu sin
>> y = square(2*pi*50*t); % Tạo xung vuông
>> z = sawtooth(2*pi*50*t); % Tín hiệu răng cưa
```

Hàm `sawtooth(t,with)` tạo tín hiệu răng cưa hoặc tam giác có các đỉnh  $\pm 1$ , chu kỳ  $2\pi$ , tham số tùy chọn `with` là tỷ lệ thời gian lên trên tổng chu kỳ. Nếu `with = 0.5` thì ta sẽ có xung tam giác.

Hàm `square(t,duty)` tạo xung vuông có các mức là  $\pm 1$ , chu kỳ  $2\pi$ , tham số tùy chọn `duty` là hệ số lấp đầy (duty cycle).

#### d) Các hàm tạo tín hiệu không tuần hoàn

Hàm `gauspuls(t, fc, bw)` tạo một xung Gaussian tần số RF có biên độ bằng 1, tần số trung tâm **fc** (hertz) và băng thông **bw**.

Hàm `chirp` tạo tín hiệu tần số quét, có nhiều phương pháp quét khác nhau: tuyến tính, bậc hai hay logarithm.

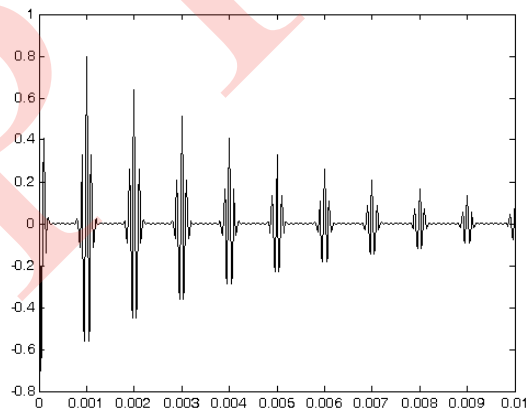
#### e) Hàm *pulstran*

Hàm `pulstran` tạo một chuỗi các xung có cùng dạng với một xung gốc.

Ví dụ. Tạo một chuỗi xung là sự lặp lại của các xung Gauss sau những khoảng thời gian bằng nhau. Các thông số cụ thể như sau: tốc độ lấy mẫu của chuỗi xung là 50 kHz, chiều dài chuỗi xung là 10ms, tốc độ lặp lại của chuỗi xung Gauss là 1 kHz, suy hao theo hàm mũ cơ số 0.8. Xung Gauss có tần số trung tâm là 10 kHz, băng thông 50%.

```
>> T = 0:1/50E3:10E-3; % véc tơ thời gian 0 - 10ms)
>> D = [0:1/1E3:10E-3;0.8.^(0:10)]'; % Cột 1 của D xác định các
      % thời điểm lặp, cột 2 của D xác định biên độ tương ứng
      % của xung Gauss (bị suy hao).
>> Y = pulstran(T,D,'gauspuls',10E3,0.5); % Gọi hàm pulstran,
      % hai thông số cuối là tần số trung tâm và tỷ lệ băng
      % thông của xung Gauss
>> plot(T,Y) % vẽ tín hiệu
```

Kết quả như sau:



Hình 3.2. Tạo chuỗi xung

#### f) Hàm *sinc(x)*

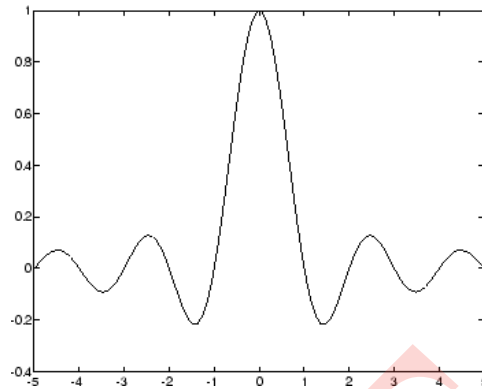
Hàm `sinc` là biến đổi Fourier ngược của xung chữ nhật có chiều rộng bằng  $2\pi$  và chiều cao bằng 1:

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x} = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega x} d\omega \quad (3.1)$$

$x$  có thể là một véc tơ hay một ma trận. Khi  $x = 0$  thì  $\text{sinc}(x) = 1$ .

Để vẽ hàm sinc với một vectơ không gian tuyến tính có giá trị từ -5 đến 5, sử dụng các lệnh sau đây:

```
>> x = linspace(-5,5);
>> y = sinc(x);
>> plot(x,y)
```



Hình 3.3. Hàm sinc

#### g) Hàm Dirichlet

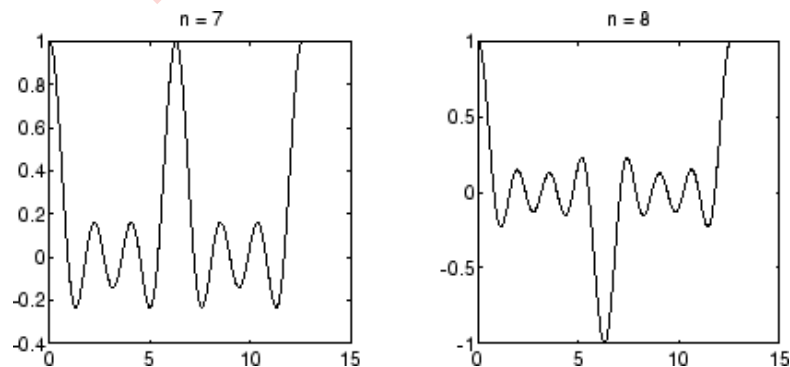
Hàm Dirichlet còn gọi là hàm sinc tuần hoàn hay hàm sinc chồng lấn (aliased sinc), với một vectơ  $x$  đầu vào, hàm Dirichlet  $D(x)$  được định nghĩa như sau:

$$D(x) = \begin{cases} \frac{\sin(Nx/2)}{N \sin(x/2)}, & x \neq 2\pi k, \quad k = 0, \pm 1, \pm 2, \pm 3 \dots \\ (-1)^{k(N-1)}, & x = 2\pi k, \quad k = 0, \pm 1, \pm 2, \pm 3 \dots \end{cases} \quad (3.2)$$

Trong đó,  $N$  là giá trị nguyên dương. Với  $N$  lẻ, hàm Dirichlet có chu kỳ  $2\pi$ ; với  $N$  chẵn, chu kỳ của nó là  $4\pi$ . Trong MATLAB để dùng hàm Dirichlet ta gọi hàm: `diric`.

Để vẽ hàm Dirichlet trong dải từ 0 đến  $4\pi$  với  $N = 7$  và  $N = 8$ , sử dụng các lệnh sau:

```
>> x = linspace(0,4*pi,300);
>> plot(x,diric(x,7)); axis tight;
>> plot(x,diric(x,8)); axis tight;
```



Hình 3.4. Hàm Dirichlet

Một số hàm khác của Signal Processing Toolbox có trong phụ lục của tài liệu này.

### 3.2.2. Control System Toolbox

Control System Toolbox cung cấp các thuật toán chuẩn công nghiệp và các ứng dụng để phân tích mang tính hệ thống, thiết kế, và điều chỉnh các hệ thống điều khiển tuyến tính. Ta có thể định nghĩa hệ thống bằng một hàm truyền đạt, không gian trạng thái, điểm không – điểm cực – độ lợi, hoặc mô hình đáp ứng tần số. Các ứng dụng và các hàm, ví dụ như hàm vẽ đáp ứng nhảy bậc và đồ thị Bode, cho phép chúng ta hình dung được hoạt động của hệ thống trong miền thời gian và miền tần số. Ta có thể điều chỉnh các tham số bù sử dụng bộ điều khiển tự động PID, phương pháp quỹ tích gốc, thiết kế LQR/LQG và các kỹ thuật tương tác và điều khiển khác. Ta có thể kiểm tra thiết kế bằng cách thay đổi thời gian quá độ, độ quá điều chỉnh lượng dự trữ pha và độ lợi và các yêu cầu khác.

Các chức năng chính của Control System Toolbox

- Hàm truyền đạt, không gian trạng thái, điểm cực – điểm không – độ lợi, và các mô hình đáp ứng tần số của các hệ thống tuyến tính
- Các kết nối nối tiếp, song song, hồi tiếp và các kết nối sơ đồ khối tổng quát của các mô hình tuyến tính.
- Đáp ứng nhảy bậc, vẽ tiêu chuẩn Nyquist, và các công cụ phân tích tính ổn định và thông số hoạt động trong miền thời gian và miền tần số
- Quỹ tích gốc, đồ thị Bode, LQR, LQG, và các kỹ thuật thiết kế hệ thống điều khiển cổ điển và theo không gian trạng thái
- Tự động điều chỉnh các bộ điều khiển PID
- Chuyển đổi mô hình biểu diễn, rời rạc hóa mô hình liên tục theo thời gian, và xấp xỉ bậc thấp cho các hệ thống bậc cao.
- Các thuật toán LAPACK và SLICOT để tối ưu độ chính xác và hoạt động.

#### 3.2.2.1. Định nghĩa một hệ thống tuyến tính

##### a) Định nghĩa bằng hàm truyền đạt

+ Hệ thống một đầu vào – một đầu ra (SISO-Single Input Single Output)

Câu lệnh: `sys = tf(num, den, T)`

- *Num*: vectơ chứa các hệ số của đa thức ở tử số, bậc từ cao đến thấp theo toán tử Laplace (hệ liên tục) hoặc theo toán tử  $z$  (hệ rời rạc).
- *Den*: vectơ chứa các hệ số của đa thức ở mẫu số, bậc từ cao xuống thấp.
- *T*: Chu kỳ lấy mẫu, chỉ dùng cho hệ rời rạc (tính bằng s)

**Ví dụ:**

Định nghĩa một hàm trong MATLAB: 
$$F(p) = 3 \frac{p+2}{p^2+2p+4}$$

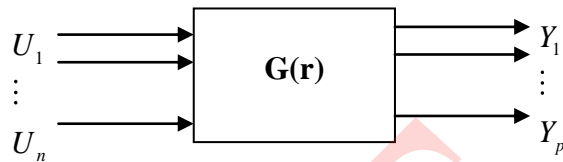
```
>> num = 3*[1 2];
```

```
>> den = [1 2 4];
>> sys1 = tf(num,den)
```

Ví dụ với hệ thống:  $F(z) = 2,1 \frac{z-0,6}{z^2-0,56z+0,4}$

```
>> num = 2.1*[1 -0.6];
>> den = [1 -0.56 4];
>> T = 0.5;
>> sys2 = tf(num,den,T)
```

+ **Hệ thống nhiều đầu vào – nhiều đầu ra (MIMO)**



$$G(r) = \begin{bmatrix} G_{11}(r) & G_{12}(r) & \dots & G_{1n}(r) \\ G_{21}(r) & G_{22}(r) & \dots & G_{2n}(r) \\ \dots & \dots & \dots & \dots \\ G_{p1}(r) & G_{p2}(r) & \dots & G_{pn}(r) \end{bmatrix}$$

Câu lệnh:

```
>> G11 = tf(num11,den11,T);
>> G12 = tf(num12,den12,T);
.....
>> G1n = tf(num2n,den2n,T);
>> G21 = tf(num21,den21,T);
>> G22 = tf(num22,den22,T);
.....
>> G2n = tf(num2n,den2n,T);
>> Gp1 = tf(nump1,denp1,T);
>> Gp2 = tf(nump2,denp2,T);
.....
>> Gpn = tf(numpn,denpn,T);
>> sys = [G11,G12,...,G1n; G22,...,G2n; ...; Gp1,Gp2,...,Gpn];
```

**b) Định nghĩa bằng điểm không – điểm cực**

+ **Hệ thống một đầu vào – một đầu ra (SISO)**

Câu lệnh: `sys = zpk(Z,P,K,T)`

- Z, P là các vectơ hàng chứa danh sách các điểm không và điểm cực của hệ thống.
- K là hệ số khuếch đại.



Chú ý: nếu hệ thống không có điểm không (cực) thì ta đặt là [].

**Ví dụ:**

Cho hệ thống có hàm truyền đạt:  $F(p) = \frac{p+2}{p(p+5)}$

Câu lệnh:

```
>>Z=-2; % điểm không (nghiệm của tử số)
>>P=[0 -5]; % điểm cực (nghiệm của mẫu số)
>>K=1;
>>sys=zpk(Z,P,K)
```

#### + Hệ thống nhiều đầu vào – nhiều đầu ra (MIMO)

Câu lệnh:

```
>> G11 = zpk(Z11,P11,T);
>> G12 = zpk(Z12,P12,T);
.....
>> G1n = zpk(Z1n,P1n,T);
>> G21 = zpk(Z21,P21,T);
>> G22 = zpk(Z22,P22,T);
.....
>> G2n = zpk(Z2n,P2n,T);
.....
>> Gp1 = zpk(Zp1,Pp1,T);
>> Gp2 = zpk(Zp2,Pp2,T);
.....
>> Gpn = zpk(Z2n,P2n,T);
>> sys = [G11,G12,...,G1n; G22,...,G2n; ...; Gp1,Gp2,...,Gpn];
```

#### c) Phương trình trạng thái

Câu lệnh: `sys = ss(A,B,C,D,T)`

- A, B, C, D: là các ma trận trạng thái định nghĩa hệ thống.
- T: chu kỳ lấy mẫu

#### d) Chuyển đổi giữa các dạng biểu diễn

- Chuyển đổi phương trình trạng thái sang hàm truyền đạt:  
`[num,den] = ss2tf(A,B,C,D)`
- Chuyển đổi từ dạng điểm không/ điểm cực sang hàm truyền đạt:  
`[num,den] = zpk2tf(Z,P,K)`
- Chuyển đổi từ hàm truyền đạt sang phương trình trạng thái  
`[A,B,C,D] = tf2ss(num,den)`

### e) Chuyển đổi giữa hệ liên tục và rời rạc

#### Số hóa một hệ thống liên tục

Câu lệnh: `sys_dis = c2d(sys, T, method)`

- `sys, sys_dis` - hệ thống liên tục và hệ thống rời rạc.
- `T` - thời gian lấy mẫu
- `method` - phương pháp lấy mẫu: 'zoh' lấy mẫu bậc 0, 'foh' lấy mẫu bậc 1, 'tustin' phương pháp Tustin...

Ví dụ: chuyển một khâu liên tục có hàm truyền đạt  $G(p) = \frac{2}{0,5p+1}$  sang khâu rời rạc

bằng phương pháp giữ mẫu bậc 0, chu kỳ lấy mẫu  $T = 0.01s$ :

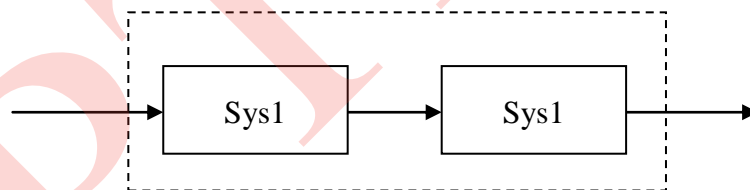
```
>>num=2;  
>>den = [0.5 1];  
>>sysc=tf(num,den);  
>>sysd=c2d(sysc,0.01,'zoh');
```

#### Hệ thống liên tục tương đương của một hệ thống rời rạc

Câu lệnh: `sys=d2c(sys_dis, method);`

### 3.2.2.2. Biến đổi sơ đồ tương đương

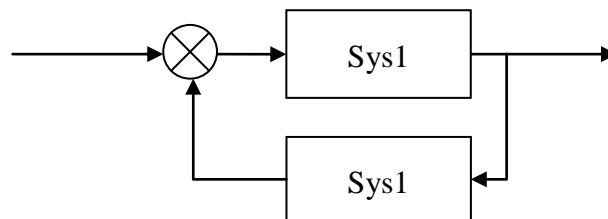
#### a) Mắc nối tiếp



Câu lệnh: `sys=series(sys1, sys2);`

#### b) Mắc phản hồi

Câu lệnh: `sys=feedback(sys1, sys2, sign);`



**sign** = +1 nếu phản hồi dương và **sign** = -1 (hoặc không có sign) nếu phản hồi âm

### 3.2.2.3. Phân tích hệ thống

#### a) Trong miền thời gian

##### Hàm quá độ

Câu lệnh: `step(sys)`

Vẽ hàm quá độ của hệ thống tuyến tính `sys`. Khoảng thời gian vẽ và bước thời gian do MATLAB tự chọn.

Một số trường hợp khác:

- `step(sys, t_end)` : vẽ hàm quá độ từ thời điểm  $t=0$  đến thời điểm  $t_{end}$ .
- `step(sys, T)` : vẽ hàm quá độ trong khoảng thời gian  $T$ .  $T$  được định nghĩa như sau:  $T = T_i : dt : T_f$ . Đối với hệ thống liên tục,  $dt$  là bước vẽ, đối với hệ thống rời rạc,  $dt = T_s$  là chu kỳ lấy mẫu.
- `step(sys1, sys2, sys3, ...)` ; vẽ hàm  $h(t)$  cho nhiều hệ thống đồng thời.
- `[y, t]=step(sys)` ; tính hàm đáp ứng  $h(t)$  và lưu vào các biến  $y$  và  $t$  tương ứng.

##### Hàm trọng lượng

Câu lệnh: `impulse(sys)` ;

#### b) Trong miền tần số

##### Đồ thị Bode

Câu lệnh: `bode(sys)` ;

Vẽ đặc tính tần số Bode của hệ thống tuyến tính `sys`. Dải tần vẽ do Matlab tự chọn.

Một số trường hợp khác:

- `bode(sys, {w_start, w_end})` ; vẽ đồ thị Bode từ tần số  $w_{start}$  đến tần số  $w_{end}$ .
- `bode(sys, w)` ; vẽ đồ thị Bode theo véc tơ tần số  $w$ . Véc tơ tần số  $w$  được định nghĩa bằng hàm *logspace*. Ví dụ: `w=logspace(-2, 2, 100)` định nghĩa véc tơ  $w$  gồm 100 điểm, từ tần số  $10^{-2}$  đến  $10^2$ .
- `bode(sys1, sys2, sys3, ...)` ; vẽ đồ thị Bode của nhiều hệ thống đồng thời.
- `[mag, phi, w]=bode(sys, ...)` ; lưu tất cả các điểm tính toán của đồ thị Bode vào véc tơ  $mag$ ,  $phi$  ứng với tần số  $w$ .
- Chú ý: Đối với hệ thống rời rạc, dải tần số để vẽ phải thỏa mãn định lý Shannon.

##### Đặc tính Nyquist

Câu lệnh: `nyquist(sys)` ;

```
nyquist(sys,{w_start,w_end});
nyquist(sys,w);
nyquist(sys1,sys2,sys3,...,w);
[real,ima,w] =nyquist(sys,...);
```

### **Đặc tính Nichols**

Câu lệnh: `nichols(sys);`

```
nichols(sys,{w_start,w_end});
nichols(sys,w);
nichols(sys1, sys2, sys3,...,w);
[mag,phi,w] =nichols(sys,...);
```

Tính toán  $|G(\omega)|$ ,  $\arg[G(\omega)]$  và vẽ trong mặt phẳng Black.

Ví dụ: Vẽ các đặc tính tần số của hệ thống sau:

$$G(p) = \frac{\omega_0^2}{p^2 + 2\xi\omega_0 p + \omega_0^2} \quad \text{với } \omega_0 = 1 \text{ rad/s và } \xi = 0.5$$

```
>>w0=1;
>>xi=0.5;
>>num=w0^2;
>>den=[1 2*xi*w0 w0^2];
>>G=tf(num,den);
>>w=logspace(-2,2,100);
>>bode(G,w); % Vẽ đồ thị Bode trong dải tần số w.
>>nichols(G); % vẽ đặc tính Nichols trong dải tần số tự chọn
                % của MATLAB;
>>nyquist(G); % Vẽ đặc tính Nyquist
```

### **c) Một số hàm để phân tích**

#### ***margin***

`Margin(sys);` vẽ đồ thị Bode của hệ thống một đầu vào /một đầu ra (SISO) và chỉ ra độ dự trữ biên độ, độ dự trữ pha tại các tần số tương ứng.

`delta_L,delta_phi,w_L,w_phi]= margin(sys);` tính toán và lưu độ dự trữ biên độ vào biến `delta_L` tại tần số `w_L`, lưu độ dự trữ pha vào biến `delta_phi` tại tần số `w_phi`.

#### ***pole***

`Vec-pol=pole(sys);` tính các điểm cực của hệ thống và lưu vào biến `vec_pol`.

#### ***tzero***

`vec_zer=tzero(sys);` tính các điểm không của hệ thống và lưu vào biến `vec_zer`.

#### **pzmap**

`[vec_pol,vec_zer]=pzmap(sys);` tính các điểm cực và điểm không của hệ thống và lưu vào các biến tương ứng.

`pzmap(sys)` tính các điểm cực và điểm không và biểu diễn trên mặt phẳng phức.

#### **dcgain**

`G0=dcgain(sys);` tính hệ số khuếch đại tĩnh của hệ thống và lưu vào `G0`.

### **d) Một số hàm đặc biệt trong không gian trạng thái**

#### **Hàm ctrl**

Câu lệnh: `C_com=ctrl(A,B);`

`C_com=ctrl(sys);`

Tính ma trận "điều khiển được" **C** của hệ thống. Ma trận **C** được định nghĩa như sau:

$$C=[A \ AB \ A^2B \ \dots \ A^{n-1}B] \text{ với } A \in \mathbb{R}^{n \times n}$$

#### **Hàm obsv**

Câu lệnh: `O_obs=obsv(A,C);`

`O_obs=obsv(sys);`

Tính ma trận "quan sát được" **O** của hệ thống. Ma trận **O** được định nghĩa như sau:

$$O=[C \ CA \ CA^2 \ \dots \ CA^{n-1}]$$

#### **Hàm ctrbf**

Câu lệnh: `[Ab,Bb,Cb,T,k]=ctrbf(A,B,C)`

Chuyển về dạng chuẩn "điều khiển được" của một hệ thống biểu diễn dưới dạng phương trình trạng thái.

Trong đó:  $A_b=TAT^{-1}$ ,  $B_b=TB$ ,  $C_b=CT^{-1}$ ,  $T$  là ma trận chuyển đổi.

#### **Hàm obsvf**

Câu lệnh: `[Ab,Bb,Cb,T,k]=obsvf(A,B,C)`

Chuyển về dạng chuẩn "quan sát được" của một hệ thống biểu diễn dưới dạng phương trình trạng thái. Trong đó:  $A_b=TAT^{-1}$ ,  $B_b=TB$ ,  $C_b=CT^{-1}$ ,  $T$  là ma trận chuyển đổi.

### **3.2.2.4. Ví dụ tổng hợp**

Cho một hệ thống kín phản hồi -1, trong đó hàm truyền đạt của hệ hở là:

$$G(p) = \frac{K}{p(1+\tau t)} * \frac{\omega_0^2}{p^2 + 2\xi\omega_0 p + \omega_0^2} \text{ với } K=1, \tau=10s; \omega_0=1\text{rad/s}; \xi=0,5$$

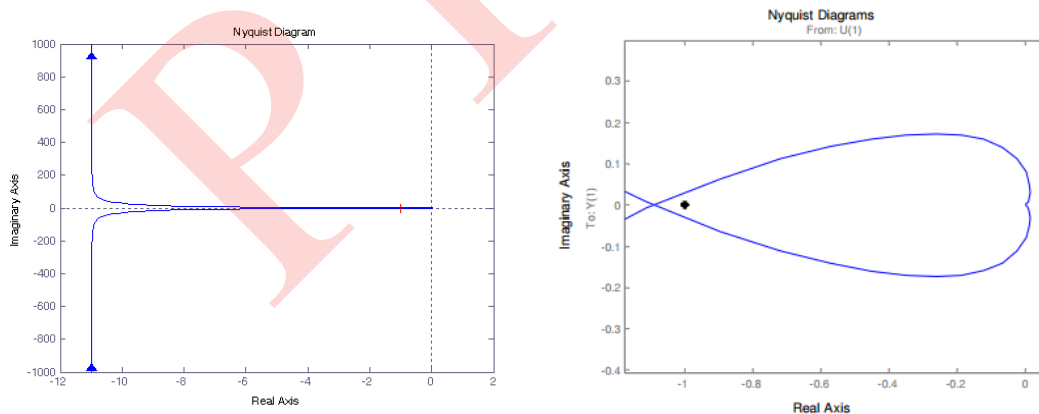
- 1) Vẽ đặc tính tần số Nyquist. Chứng tỏ rằng hệ kín không ổn định.
- 2) Vẽ đáp ứng quá độ của hệ thống kín.
- 3) Để hệ thống ổn định, người ta hiệu chỉnh hệ số khuếch đại  $K = 0,111$ . Xác định tần số cắt, độ dự trữ biên độ và độ dự trữ pha của hệ thống trong trường hợp này.
- 4) Xác định các thông số quá độ (thời gian quá độ lớn nhất  $T_{\max}$ , độ quá điều chỉnh cực đại  $\sigma_{\max}$ ) của hệ thống đã hiệu chỉnh.

Thực hiện:

**Câu 1:**

```
>>K=1;to=10;w0=1;xi=0.5;
>>num1=K;den1=[to 1 0] ;
>>num2=w0^2;den2=[1 2*xi*w0 w0^2] ;
>>G=tf(num1,den1)*tf(num2,den2)
Transfer function:
          1
-----
10 s^4 + 11 s^3 + 11 s^2 + s
>>w=logspace(-3,2,100) ; % tạo véc tơ tần số để vẽ các đặc tính
                           % tần số.
>>nyquist(G,w);
```

Đặc tính được biểu diễn như Hình 3.5 dưới đây.



**Hình 3.5. Đặc tính tần số Nyquist của hệ hở**

Để xét tính ổn định của hệ kín dùng tiêu chuẩn Nyquist, trước tiên ta xét tính ổn định của hệ thống hở. Nghiệm của phương trình đặc tính của hệ hở được xác định:

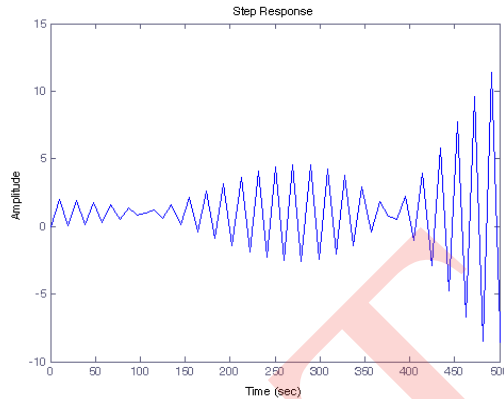
```
>> pole(G)
ans =
      0
-0.5000 + 0.8660i
-0.5000 - 0.8660i
-0.1000
```

Hệ có 1 nghiệm bằng 0 nên ở biên giới ổn định.

Quan sát đặc tính tần số Nyquist của hệ thống hở trên hình 3.2 (phần phóng to bên phải), ta thấy đặc tính Nyquist bao điểm  $(-1, j0)$ , và do hệ hở ở biên giới ổn định nên theo tiêu chuẩn Nyquist, **hệ thống kín sẽ ổn định.**

**Câu 2:**

```
>>G_loop=feedback(G,1,-1); % hàm truyền hệ kín
>>step(G_loop);
```

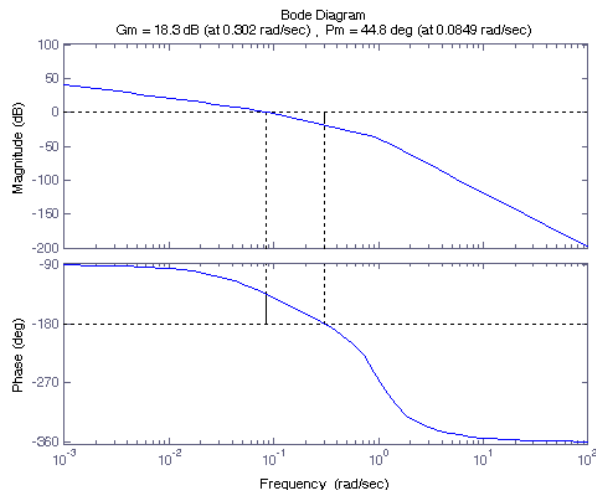


**Hình 3.6. Đáp ứng quá độ của hệ thống kín**

**Câu 3**

```
>>K=0.111; num1=K; % Thay đổi hệ số khuếch đại K
>>GK=tf(num1,den1)*tf(num2,den2)
Transfer function:
0.111
-----
10 s^4 + 11 s^3 + 11 s^2 + s
>>margin(GK)
```

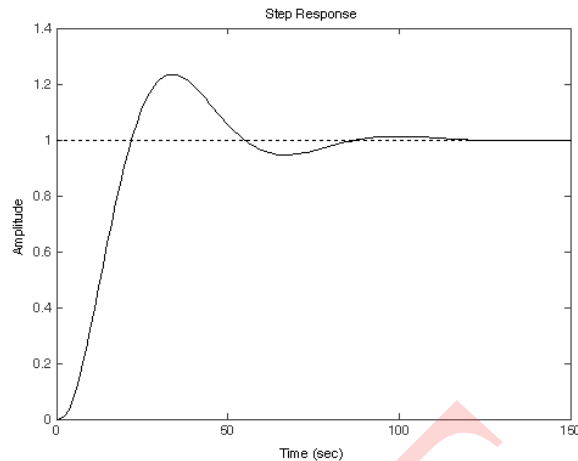
Đồ thị Bode của hệ hở đã hiệu chỉnh biểu diễn trên hình 3.3. Từ đồ thị này, ta có thể xác định được:  $\Delta G = 18.34\text{dB}$ ;  $\Delta\varphi = 44.78^\circ$ ;  $\omega_c = 0.085\text{rad/s}$



**Hình 3.7. Đồ thị Bode của hệ thống hở đã hiệu chỉnh**

**Câu 4:**

```
>>GK_loop=feedback(GK,1,-1);  
>>step(GK_loop);
```



**Hình 3.8. Đáp ứng quá độ của hệ thống kín đã hiệu chỉnh**

Sử dụng con trỏ chuột và kích vào các điểm cần tìm trên đồ thị, ta xác định được:

$$\sigma_{\max} = 23\%; T_{\max} = 70.7s.$$

### 3.2.3. Symbolic Math Toolbox

Symbolic Math Toolbox cung cấp các hàm để giải và mô tả các ký hiệu toán học, và đánh giá độ chính xác của các biểu thức số học. Ta có thể thực hiện các phép giải tích như vi phân, tích phân, rút gọn, biến đổi và giải phương trình. Ta cũng có thể tạo các code cho MATLAB, Simulink... từ các mô tả bằng chữ.

Symbolic Math Toolbox bao gồm ngôn ngữ MuPAD, nó được tối ưu để hoạt động trong mô tả bằng chữ. MuPAD là một hệ thống đại số máy tính, đầu tiên được phát triển bởi nhóm nghiên cứu của đại học Paderborn (Đức). Đến 2008 được Mathworks mua lại và code MuPAD nằm trong Symbolic Math Toolbox.

#### *Các tính năng chính của Symbolic Math Toolbox*

- Tích hợp ký hiệu, vi phân, biến đổi và đại số tuyến tính
- Giải phương trình đại số và phương trình vi phân thuần nhất.
- Rút gọn và các thao tác trên biến
- Tạo code từ các mô tả ký hiệu cho MATLAB, Simulink, Simscape, C, Fortran, MathML và Tex.
- Các phép toán với độ chính xác tùy chọn theo yêu cầu.

Một số thư viện của Symbolic Math Toolbox như bảng dưới đây.



**Bảng 3.1. Một số thư viện của Symbolic Math Toolbox**

<b>Tiện ích</b>	<b>Nội dung</b>
Calculus	Đạo hàm, tích phân, giới hạn, tổng và chuỗi Taylor
Linear Algebra	Nghịch đảo, định thức, giá trị riêng, phân tích và dạng chính tắc của ma trận
Simplification	Giải bằng chữ và số các phương trình đại số và vi phân
Variable – Precision Arithmetic (VPA)	Các phép toán (số học) với độ chính xác tùy chọn theo yêu cầu.
Transform	Biến đổi Laplace, Fourier và Z
Special Mathematical Function	Các hàm toán học đặc biệt của các ứng dụng toán học kinh điển

### 3.2.3.1. Các đối tượng tượng trưng

#### a) Giới thiệu chung

Các đối tượng tượng trưng (chữ, ký hiệu - symbolic) là dạng dữ liệu đặc biệt trong MATLAB sử dụng trong Symbolic Math Toolbox. Nó cho phép thực hiện các thao tác toán học trong không gian làm việc của MATLAB mà không cần tính giá trị số. Ta có thể sử dụng các đối tượng tượng trưng để thực hiện nhiều tính toán và phân tích khác nhau:

- Vi phân, bao gồm cả vi phân riêng phần.
- Tích phân xác định và tích phân bất định.
- Giới hạn, bao gồm cả giới hạn một phía.
- Tổng, bao gồm chuỗi Taylor.
- Các phép tính ma trận
- Giải các phương trình đại số và phương trình vi phân.
- Các phép toán với độ chính xác tùy chọn theo yêu cầu.
- Biến đổi tích phân.

Các đối tượng tượng trưng là các biến chữ tượng trưng, số tượng trưng, biểu tượng, ma trận tượng trưng và các hàm tượng trưng.

#### b) Các biến tượng trưng

Để khai báo các biến  $x$  và  $y$  là đối tượng tượng trưng (biến chữ) sử dụng câu lệnh *syms*

```
>>syms x y
```

Ta có thể thao tác trên các biến chữ theo quy tắc toán học. Ví dụ:

```
>> x+x+y
```

```
ans
```

$$2*x+y$$

### c) Số tượng trưng

Để tạo số tượng trưng (biến số), sử dụng lệnh **sym**

```
>> a=sym('2')
```

Nếu tạo các số 15 hoặc nhỏ hơn thập phân có thể bỏ qua dấu (')

```
>>a=sym(2)
```

Ví dụ sau cho thấy sự khác nhau giữa dữ liệu kiểu **double** và dữ liệu biến số tương ứng.

```
>>sqrt(2)
```

Cho kết quả là

```
ans =  
1.4142
```

Trong khi đó, nếu ta tính căn bậc hai của số tượng trưng 2:

```
>>a=sqrt(sym(2))
```

Kết quả sẽ là:

```
a =  
2^(1/2)
```

Để có kết quả của biến số ta dùng lệnh **double**

```
>>double(a)  
ans =  
1.4142
```

Ta cũng có thể tạo dạng thập phân bằng các biến số

```
>>sym(2)/sym(5)  
ans =  
2/5
```

Hoặc đơn giản hơn là:

```
>>sym(2/5)  
ans =  
2/5
```

MATLAB thực hiện các phép số học theo dạng biến phân số khác với cách nó tính theo dạng phân số chuẩn. Theo mặc định, MATLAB lưu trữ tất cả các giá trị số theo kiểu dữ liệu thập phân dạng double. Ví dụ:

```
>>2/5+1/3  
ans =  
0.7333
```

Nếu ta cộng các biến số dạng thập phân, MATLAB sẽ tìm mẫu số chung và cộng các phân số với nhau theo cách cộng phân số thông thường:

```
>>sym(2/5)+sym(1/3)
```

```
ans =  
11/15
```

### 3.2.3.2. Tạo các biến và các mô tả symbolic

#### a) Tạo các biến symbolic

Lệnh **sym** tạo các biến và các mô tả tượng trưng. Ví dụ:

```
>>x=sym('x');  
>>a=sym('alpha');
```

Sẽ tạo một biến  $x$  với giá trị gán vào là  $x$ , và tạo một biến  $a$  với giá trị gán là  $\alpha$ . Một cách khác để tạo các biến tượng trưng là sử dụng lệnh **syms**.

```
>>syms x  
>>a=sym('alpha');
```

Ta có thể sử dụng lệnh **sym** hoặc **syms** để tạo các biến tượng trưng.

Với lệnh **syms**:

- Không cần dùng dấu ngoặc và dấu nháy đơn: `syms x`
- Có thể tạo nhiều biến với một câu lệnh.
- Thuận lợi để tạo các biến riêng lẻ hoặc đa biến.

Với lệnh **sym**:

- Cần có dấu ngoặc và nháy đơn: `x=sym('x')`. Khi tạo các biến số nhỏ hơn 10 có thể bỏ dấu nháy đơn: `f=sym(5)`
- Mỗi lệnh chỉ tạo một biến.
- Thuận tiện để tạo các biến số hoặc các mô tả tượng trưng.
- Thuận tiện để tạo các biến trong các hàm và script.

#### b) Tạo các mô tả

Giả sử muốn tạo một biến chữ để mô tả tỷ số:

$$\varphi = \frac{1+\sqrt{5}}{2}$$

Câu lệnh:

```
>>phi=sym('(1+sqrt(5))/2');
```

Bây giờ ta có thể thực hiện các phép tính với biến `phi`. Ví dụ:

```
>>f=phi^2-phi-1
```

Kết quả:

```
f =  
(1/2+1/2*5^(1/2))^2-3/2-1/2*5^(1/2)
```

Giả sử ta muốn giải phương trình bậc 2  $f = ax^2 + bx + c$ . Nhập các dòng lệnh sau:

```
>>f = sym('a*x^2 + b*x + c');
```

Điều này sẽ gán mô tả  $ax^2 + bx + c$  vào biến  $f$ . Tuy nhiên, trong trường hợp này, Symbolic Math Toolbox sẽ không tạo các biến tương ứng với các số hạng  $a, b, c$  và  $x$ . Để thực hiện phép toán trong mô tả của  $f$ , ta phải tạo các biến. Cách tốt nhất là nhập theo các lệnh sau:

```
>>a= sym('a');
```

```
>>b= sym('b');
```

```
>>c= sym('c');
```

```
>>x= sym('x');
```

Hoặc đơn giản là:

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

Sau đó nhập:

```
>>f= a*x^2 + b*x + c
```

Chú ý: Để tạo các mô tả cố định, phải dùng lệnh **sym**, không dùng lệnh **syms**. Ví dụ, để tạo một mô tả với giá trị cố định là 5, nhập lệnh `f=sym(5)`. Lệnh `f=5` không định nghĩa  $f$  là một mô tả tương trưng.

Ngoài các lệnh **sym** và **syms**, cũng có một số lệnh khá hiệu quả trong việc tạo các mô tả hàm biến chữ, ví dụ hàm: **inline**

Cú pháp: **INLINE(EXPR)** sẽ tạo một đối tượng hàm từ các diễn tả trong đối số **EXPR**. Các đầu vào được tự động xác định bởi **EXPR** làm tên các biến. Nếu không có biến tồn tại thì tên biến 'x' sẽ được dùng.

Cú pháp **INLINE(EXPR, ARG1, ARG2, ...)** sẽ tạo một hàm mà các đầu vào được định bằng các đối số **ARG1, ARG2, ...**

Ví dụ:

```
>>g = inline('t^2') % tạo hàm với biến số t.
```

```
g =
```

```
Inline function:
```

```
g(t) = t^2
```

```
>> g(2) % Thay biến số t = 2 và tính kết quả
```

```
ans =
```

```
4
```

Tạo hàm với hai biến định rõ

```
>>g = inline('sin(2*pi*f + theta)', 'f', 'theta')
```

### c) Tạo hàm symbolic

Có thể dùng hai lệnh **sym** và **syms** để tạo các hàm. Ví dụ, ta có thể tạo hàm  $f(x, y)$  trong đó  $x, y$  là các biến. Các đơn giản nhất là dùng lệnh **syms**:

```
>>syms f(x,y)
```

Tương tự ta có thể dùng lệnh `sym` để tạo hàm. Chú ý lệnh ***sym*** chỉ tạo hàm, nó không tạo các biến cho các đối số. Ta phải tạo các biến trước khi tạo hàm:

```
>>syms x y
>>f(x,y)=sym('f(x,y)');
```

Thay vì tạo một hàm tùy ý, nếu muốn tạo một hàm toán học cụ thể ta sử dụng 2 bước. Đầu tiên tạo các biến chữ tương ứng các đối số của hàm:

```
>>syms x y
```

Sau đó gán các mô tả toán học cho hàm. Trong trường hợp này, thao tác gán sẽ tạo hàm mới:

```
>>f(x,y)=x^3*y^3
f(x,y) =
x^3*y^3
```

Sau khi đã tạo được hàm, ta có thể tính vi phân, tích phân hoặc rút gọn nó, thay thế giá trị và thực hiện các phép toán học. Ví dụ, tính đạo hàm bậc 2 của  $f(x,y)$  với biến  $y$ . Kết quả  $d^2fy$  là một hàm:

```
>>d2fy=diff(f,y,2)
d2fy(x,y) =
6*x^3*y
```

Bây giờ tính  $f(x,y)$  với  $x=y+1$

```
>>f(y+1,y)
ans =
y^3*(y + 1)^3
```

#### ***d) Tạo các đối tượng symbolic với tên giống nhau***

Nếu ta đặt một biến giống với mô tả tượng trưng đã có thì áp dụng lệnh ***syms*** cho biến, MATLAB sẽ xóa bỏ các mô tả trước đó của biến. Ví dụ:

```
>>syms a b
>>f=a+b
```

Kết quả là

```
f =
a + b
```

Sau đó ta lại nhập lệnh:

```
>>syms f
f
```

thì MATLAB sẽ xóa bỏ giá trị  $a+b$  khỏi  $f$ .

```
f=
f
```

Ta có thể dùng lệnh **syms** để xóa các biến đã gán trước đó. Tuy nhiên, lệnh **syms** không xóa với các trường hợp biến là số: phức, thực và dương.

#### ***e) Tạo ma trận các biến chữ***

##### ***+ Sử dụng các đối tượng đã có***

Ma trận vòng có tính chất là mỗi hàng chứa tất cả các phần tử ở hàng trước bằng cách hoán vị vòng đi một vị trí. Ví dụ, tạo ma trận vòng với các phần tử  $a, b$  và  $c$ , dùng các lệnh sau:

```
>>syms a b c
>>A=[a b c;c a b; b c a]
A =
[ a, b, c]
[ c, a, b]
[ b, c, a]
```

Do ma trận A là ma trận vòng, tổng các phần tử trong mỗi hàng hoặc mỗi cột đều cho kết quả giống nhau. Tìm tổng của tất cả các phần tử tại hàng đầu tiên:

```
>>sum(A(1,:))
ans =
a + b + c
```

Để kiểm tra liệu tổng của các phần tử trong hàng đầu tiên có bằng tổng các phần tử của cột thứ hai hay không, sử dụng lệnh **logical**:

```
>>logical (sum(a1,')==sum(A(:,2)))
Kết quả bằng nhau:
ans=
1
```

##### ***+ Tạo các phần tử cùng với ma trận***

Hàm **sym** cũng cho phép ta định nghĩa ma trận hoặc các véc tơ tượng trưng mà không cần định nghĩa các phần tử của nó. Trong trường hợp này, hàm **sym** sẽ tạo các phần tử của một ma trận tượng trưng cùng với ma trận. Nó sẽ mô tả tất cả các phần tử theo cùng một dạng: chỉ số hàng và cột. Sử dụng tham số đầu tiên của hàm **sym** để làm cơ sở xác định tên của các phần tử. Ta có thể sử dụng bất kỳ tên hợp lệ nào. Để kiểm tra xem tên có hợp lệ không, sử dụng lệnh **isvarname**. Theo mặc định, hàm **sym** sẽ phân tách chỉ số hàng và cột bằng nét gạch ngang. Ví dụ, để tạo một ma trận A với kích thước 2x4 với các phần tử  $A_{1,1}, \dots, A_{2,4}$

```
>>A=sym('A',[2 4])
A =
[ A1_1, A1_2, A1_3, A1_4]
[ A2_1, A2_2, A2_3, A2_4]
```

Để điều khiển tên các phần tử trong ma trận, sử dụng cú pháp **%d** ở đối số đầu tiên:

```
>>A=sym('A%d%d',[2 4])
```

```
A =
[ A11, A12, A13, A14]
[ A21, A22, A23, A24]
```

#### + **Tạo ma trận các biến số**

Có cách hiệu quả khi sử dụng lệnh **sym** để chuyển đổi một ma trận từ số sang dạng biến.  
Câu lệnh:

```
>>A=hilb(3)
```

Tạo ma trận Hilbert 3x3

```
A =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
```

Sử dụng lệnh **sym** cho A

```
>>A=sym(A)
```

Ta có ma trận Hilbert theo dạng biến

```
>> A=sym(A)
A =
[ 1, 1/2, 1/3]
[ 1/2, 1/3, 1/4]
[ 1/3, 1/4, 1/5]
```

#### + **Tìm các biến chữ trong mô tả, trong hàm và trong ma trận**

Để tìm các biến trong các mô tả, trong hàm hay trong ma trận, sử dụng lệnh **symvar**. Ví dụ, tìm tất cả các biến trong mô tả  $f$  và  $g$ :

```
>>syms a b n t x
>>f=x^n;
>>g=sin(a*t+b);
>>symvar(f)
ans=
[n, x]
```

Hàm **symvar** sắp xếp các biến theo thứ tự alphabet. Tương tự, ta có thể tìm các biến trong  $g$  như sau:

```
>>symvar(g)
ans =
[ a, b, t]
```

Hàm **symvar** cũng cho ra  $n$  biến đầu tiên trong các mô tả, trong ma trận hay hàm. Để xác định số biến cần tìm, sử dụng tham số thứ hai của hàm **symvar**, Ví dụ, tìm hai biến đầu tiên trong  $g$ :

```
>>symvar(g,2)
```

```
ans =  
[ t, b]
```

Chú ý rằng hai biến đầu tiên trong trường hợp này là  $a$  và  $b$ . Khi ta gọi hàm *symvar* với 2 tham số, nó sẽ sắp xếp các biến gần với  $x$ .

```
>>syms x y w z  
>>f(w,z)=x*w + y*z;  
>>symvar(f)  
ans =  
[ w, x, y, z]
```

Khi ta gọi hàm *symvar* với 2 tham số, nó trả kết quả các đầu vào của hàm trước các biến khác:

```
>>symvar(f,2)  
ans =  
[ w, z]
```

#### + *Tìm biến mặc định*

Nếu ta không định rõ một biến độc lập khi thực hiện thay thế, vi phân hay tích phân, MATLAB sử dụng biến mặc định. Biến mặc định thông thường là một chữ cái gần chữ  $x$  nhất, hoặc đầu vào đầu tiên của hàm. Để tìm biến mặc định, sử dụng lệnh *symvar(f,1)*.

Ví dụ:

```
>>syms s t  
>>f = s + t;  
>>symvar(f,1)  
ans =  
t  
  
>>syms sx tx  
>>f= sx + tx;  
>>symvar(f,1)  
ans =  
tx
```

### 3.2.3.3. Một số phép tính toán

#### a) *Đạo hàm*

Tạo biến và hàm:

```
>>syms a x  
>>f = sin(a*x)
```

Khi đó đạo hàm của  $f(x)$  theo  $x$  sẽ như sau:

```
>> df=diff(f)  
df =
```



$$\cos(a*x)*a$$

Để tính đạo hàm bậc 2 của hàm  $f$  theo  $x$  và  $a$  ta làm như sau:

```
>> diff(f,2)
ans =
-sin(a*x)*a^2
>> diff(f,a,2)
ans =
-sin(a*x)*x^2
```

Hàm diff có thể dùng với đối số ma trận. Trong trường hợp này đạo hàm được thực hiện trên từng phần tử. Ví dụ:

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

Kết quả:

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

Lệnh:

```
>>dy=diff(A)
```

Cho kết quả:

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

**Bảng 3.2. Hàm tính đạo hàm**

Hàm toán học	Lệnh MATLAB
$f = x^n$ $f' = nx^{n-1}$	$f=x^n$ diff(f) hoặc diff(f,x)
$g = \sin(at+b)$ $g' = a \cos(at+b)$	$g=\sin(a*t+b)$ diff(g) hoặc diff(g,t)
$f = \exp(ax+b)$	syms a b x $f=\exp(a*x+b)$
$\frac{df}{dx}$	diff(x) hoặc diff(f,x)
$\frac{df}{da}$	diff(f,a)
$\frac{d^2 f}{da^2}$	diff(f,a,2)

## b) Giới hạn

Đạo hàm của một hàm là giới hạn sau đây nếu nó tồn tại:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

Symbolic Math Toolbox cho phép giới hạn của một hàm một cách trực tiếp.

Câu lệnh:

```
>>syms h n x
>>dc=limit((cos(x+h)-cos(x))/h,h,0)
```

Cho kết quả:

```
dc =
-sin(x)
```

Và:

```
>>limit((1+x/n)^n,n,inf)
```

Cho kết quả:

```
ans=
exp(x)
```

minh họa 2 trong số các giới hạn quan trọng của toán học: đạo hàm (trong trường hợp hàm cos) và hàm mũ. Trong khi nhiều giới hạn  $\lim_{x \rightarrow a} f(x)$  là hai phía (nghĩa là kết quả như nhau cho dù  $x$  tiến tới bên phải hay bên trái  $a$ ) lại có những giới hạn phải và trái khác nhau. Do đó 3 giới hạn:

```
>>limit(1/x,x,0) % tương đương, limit(1/x
```

Cho kết quả :

```
ans=
NaN
```

Lệnh:

```
>>limit(1/x,x,0,'left')
```

Cho:

```
ans=
-inf
```

Lệnh

```
>>limit(1/x,x,0,'right')
```

Cho:

```
ans=
inf
```

Như vậy `limit(f)` tương đương với `limit(f,x,0)`. Một số lệnh giới hạn cho trong bảng dưới:

**Bảng 3.3. Hàm tính giới hạn**

Hàm toán học	Lệnh MATLAB
$\lim_{x \rightarrow 0} f(x)$	<code>limit(f)</code>
$\lim_{x \rightarrow a} f(x)$	<code>limit(f,x,a)</code> hoặc <code>limit(f,a)</code>
$\lim_{x \rightarrow -a} f(x)$	<code>limit(f,x,a,'left')</code>
$\lim_{x \rightarrow +a} f(x)$	<code>limit(f,x,a,'right')</code>

**c) Tích phân****+ Các vấn đề chung:**

Nếu  $f$  là một hàm thì  $\text{inf}(f)$  sẽ tìm một biểu thức khác  $F$  sao cho  $\text{diff}(F) = f$ . Như vậy  $\text{inf}(f)$  cho ta tích phân bất định của  $f$ . Tương tự như đạo hàm  $\text{inf}(f,v)$  lấy tích phân theo biến độc lập  $v$ . Ta có bảng dưới:

**Bảng 3.4. Hàm tính tích phân**

Hàm toán học	Lệnh MATLAB
$\int x^n dx = \frac{x^{n+1}}{n+1}$	<code>int(x^n)</code> hay <code>int(x^n,x)</code>
$\int_0^{\pi/2} \sin(2x) dx = 1$	<code>int(sin(2*x),0,pi/2)</code> hay <code>int(sin(2*x),x,0,pi/2)</code>
$g = \cos(at+b)$ $\int g(t) dt = \frac{1}{a} \sin(at+b)$	<code>g=cos(a*t+b)</code> <code>int(g)</code> hay <code>int(g,t)</code>

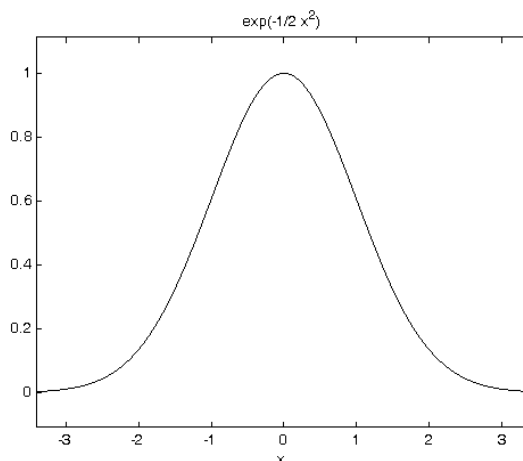
Khi MATLAB không tìm được tích phân nó viết lại lệnh đã nhập vào.

**+ Tích phân với hằng số thực**

Một trong các vấn đề khi tính tích phân là giá trị của các thông số. Ta xét hàm  $e^{-(kx)^2}$ . Hàm này rõ ràng có giá trị dương với mọi  $k$  và  $x$  và có dạng hình chuông. Giá trị của hàm tiến đến 0 khi  $x \rightarrow \pm\infty$  với mọi số thực  $k$ . Ta lấy ví dụ  $k = 1/\sqrt{2}$  và vẽ đồ thị của hàm bằng các lệnh sau:

```
>>syms x
>>k=sym(1/sqrt(2));
>>f=exp(-(k*x)^2);
>>ezplot(f)
```

Kết quả như hình sau:



Hình 3.9. Đồ thị hàm  $\exp(-1/2 x^2)$

Tuy nhiên nhân tính toán Maple không coi  $k^2$  và  $x^2$  là những số dương mà chỉ là các biến hình thức, không có thuộc tính toán học.

Chú ý: Maple là một gói phần mềm toán học thương mại phục vụ cho nhiều mục đích. Nó phát triển lần đầu tiên vào năm 1980 bởi Nhóm Tính toán Hình thức tại Đại học Waterloo ở Waterloo, Ontario, Canada. Người dùng có thể nhập biểu thức toán học theo các ký hiệu toán học truyền thống. Có thể dễ dàng tạo ra những giao diện người dùng tùy chọn. Maple hỗ trợ cho cả tính toán số và tính toán hình thức, cũng như hiển thị. Phiên bản hiện tại là Maple 13 được phát hành vào tháng 5 năm 2009.

Do vậy khi tính  $\int_{-\infty}^{\infty} e^{-(kx)^2} dx$  bằng các lệnh:

```
>>syms x k
>>f=exp(-(k*x)^2);
>>int(f,x,-inf,inf) % tương đương, int(f,-inf,inf)
```

Kết quả sẽ là:

```
Definite integration: Can't determine if the integral is
convergent.
```

```
Need to know the sign of --> k^2
```

```
Will now try indefinite integration and then take limits.
```

```
Warning: Explicit integral could not be found.
```

```
ans =
```

```
int(exp(-k^2*x^2),x= -inf..inf)
```

+ **Các biến thực theo sym:**

Chú ý là Maple không thể xác định dấu của  $k^2$ . Ta phải xử lý bằng cách làm cho  $k$  trở thành số thực bằng lệnh **sym**. Hàm sym có một tùy chọn **real** cho phép ta khai báo  $k$  là biến thực. Và do đó tích phân trên hoàn toàn tính được trong toolbox nhờ các lệnh sau:

```
>> syms k real % Chắc chắn x đã được khai báo
>>int(f,x,-inf,inf)
```

Kết quả là:

```
ans =  
signum(k)/k*pi^(1/2)
```

Chú ý là  $k$  bây giờ là biến chữ trong vùng làm việc của MATLAB và là biến thực trong vùng làm việc của Maple. Khi nhập lệnh:

```
>>clear k
```

Ta chỉ xóa được  $k$  trong vùng làm việc của MATLAB, muốn  $k$  không còn là số thực trong Maple ta phải dùng lệnh:

```
>>syms k unreal
```

Ta có bảng một số lệnh sau:

**Bảng 3.5. Một số hàm mũ và tích phân**

Hàm toán học	Lệnh MATLAB
$f(x) = e^{-kx}$	<code>syms k x</code> <code>f=exp(-k*x)</code>
$\int f(x)dx$	<code>inf(f)</code> hay <code>int(f,x)</code>
$\int f(k)dk$	<code>inf(f,k)</code>
$\int_0^1 f(x)dx$	<code>int(f,0,1)</code> hay <code>int(f,x,0,1)</code>
$g(x) = e^{-(kx)^2}$	<code>syms k x real</code> <code>g=exp(-(k*x)^2)</code>
$\int_{-\infty}^{\infty} g(x)dx$	<code>int(g,-inf,inf)</code> hay <code>int(g,x,-inf,inf)</code>

#### d) Tính tổng

Ta có thể tính tổng biểu thức chữ khi chúng tồn tại bằng cách dùng lệnh ***symcum***.

Ví dụ chuỗi:  $1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots$  cho tổng là  $\frac{\pi^2}{6}$ , còn chuỗi  $1 + x^2 + x^3 + \dots$  có tổng là  $\frac{1}{1-x}$ .

Các tổng trên được tính như sau:

```
>>syms x k  
>>s1 = symsum(1/k^2,1,inf)  
>>s2 = symsum(x^k,k,0,inf)  
s1 =  
1/6*pi^2  
s2 =  
-1/(x-1)
```

### e) Chuỗi Taylor

Cho hàm  $f(x)$  thì:

```
>>T=taylor(f,8)
```

Cho kết quả:

T =

$1/9+2/81*x^2+5/1458*x^4+49/131220*x^6$

Là khai triển Taylor của  $f(x)$  lân cận  $x=0$  (khai triển MacLaurin) có chứa 8 số hạng khác 0.

Câu lệnh:

```
>>syms x
```

```
>>g = exp(x*sin(x))
```

```
>>t = taylor(g,12,2)
```

tạo ra khai triển Taylor của  $f(x)$  tại  $x=2$  và chứa đến 12 số hạng khác 0. Ta vẽ các hàm này lên cùng một đồ thị để thấy được khả năng xấp xỉ chuỗi Taylor với hàm thực  $g$  ở trên như các câu lệnh sau:

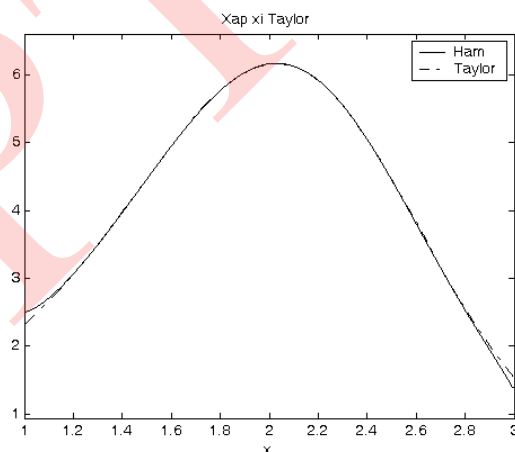
```
>>xd = 1:0.05:3; yd = subs(g,x,xd);
```

```
>>ezplot(t, [1,3]); hold on;
```

```
>>plot(xd, yd, 'r-.')
```

```
>>title('Xap xi Taylor ');
```

```
>>legend('Ham','Taylor')
```



Hình 3.10. Xấp xỉ chuỗi Taylor với hàm thực

Tiếp đó ta dùng lệnh:

```
>>pretty(t)
```

để in các kết quả dưới dạng biểu thức toán học dễ đọc.

### 3.2.3.4. Rút gọn và thay số

#### a) Rút gọn biểu thức

Ta xét 3 biểu thức khác nhau sau đây:

```
>>syms x
>>f = x^3-6*x^2+11*x-6
>>g = (x-1)*(x-2)*(x-3)
>>h = x*(x*(x-6)+11)-6
```

Thực hiện các lệnh:

```
>>pretty(f), pretty(g), pretty(h)
```

ta nhận được:

```
f =
x^3-6*x^2+11*x-6
g =
(x-1)*(x-2)*(x-3)
```

Cả 3 biểu thức là các dạng biểu diễn toán học khác nhau của cùng một hàm toán học, đó là đa thức bậc 3 theo  $x$ . Mỗi một dạng thích hợp với một dạng tính toán. Dạng thứ nhất  $f$  là dạng chung nhất thường được dùng biểu diễn đa thức. Nó đơn giản là một tổ hợp tuyến tính của các số mũ của  $x$ . Dạng thứ 2, hàm  $g$ , là dạng phân tích thành thừa số. Nó biểu diễn nghiệm của đa thức. Tuy nhiên không phải đa thức nào cũng có nghiệm, nghĩa là có thể phân tích được thành thừa số. Dạng thứ 3 là dạng Horner của đa thức. Nó rất tiện dụng dùng để tính trị số của đa thức tại mọi giá trị nào đó của  $x$ .

Symbolic Math Toolbox cung cấp một số hàm dùng để biến đổi các biểu thức đại số và lượng giác thành các biểu thức đơn giản hơn. Đó là: **collect**, **expand**, **horner**, **factor**, **simplify** và **simple**.

#### + **collect**

Câu lệnh:

```
>>collect(f)
```

xem  $f$  như một đa thức gồm các biến  $x$  và gộp tất cả các hệ số cùng bậc của  $x$ . Đối số thứ 2 chỉ rõ biến gộp nếu có nhiều biến trong biểu thức. Một số ví dụ như sau:

$f$	<code>collect(f)</code>
$(x-1)(x-2)(x-3)$	$x^3-6x^2+11x-6$
$x*(x*(x-6)+11)-6$	$x^3-6x^2+11x-6$
$(1+x)*t + x*t$	$2*x*t+t$

#### + **expand**

Câu lệnh:

```
>>expand(f)
```

khai triển biểu thức. Ví dụ như sau:

$f$	$\text{expand}(f)$
$a * (x+y)$	$a*x+a*y$
$(x-1) * (x-2) * (x-3)$	$x^3-6*x^2+11*x-6$
$x * (x * (x-6) + 11) - 6$	$x^3-6*x^2+11*x-6$
$\exp(a+b)$	$\exp(a) + \exp(b)$
$\cos(x+y)$	$\cos(x) * \cos(y) - \sin(x) * \sin(y)$
$\cos(3 * \arccos(x))$	$4*x^3-3*x$

+ **horner**

Câu lệnh:

```
>>horner(f)
```

Biến đổi một đa thức thành dạng Horner hay dạng lồng nhau. Ví dụ như sau:

$f$	$\text{horner}(f)$
$x^3-6*x^2+11*x-6$	$-6+(11+(-6+x)*x)*x$
$1.1+2.2*x+3.3*x^2$	$11/10+(11/5+33/10*x)*x$

+ **factor**

Nếu  $f$  là đa thức hệ số hữu tỷ thì câu lệnh:

```
>>factor(f)
```

Biểu diễn  $f$  là tích của các đa thức có bậc thấp hơn với hệ số hữu tỷ. Ví dụ:

$f$	$\text{factor}(f)$
$x^3-6*x^2+11*x-6$	$(x-1) * (x-2) * (x-3)$
$x^3-6*x^2+11*x-5$	$x^3-6*x^2+11*x-5$
$x^6+1$	$(x^2+1) * (x^4-x^2+1)$

Đây là một ví dụ khác về phân tích đa thức  $x^n+1$  thành thừa số:

```
>>syms x;
>>n = 1:9;
>>x = x(ones(size(n)));
>>p = x.^n + 1;
>>f = factor(p);
>>[p; f].'
```

Trả về ma trận với các đa thức ở cột thứ nhất và các thừa số ở cột thứ 2:

```
[ x+1,      x+1]
[ x^2+1,    x^2+1]
```



```
[ x^3+1, (x+1)*(x^2-x+1) ]
[x^4+1, x^4+1 ]
[x^5+1, (x+1)*(x^4-x^3+x^2-x+1)]
[x^6+1, (x^2+1)*(x^4-x^2+1) ]
[x^7+1, (x+1)*(1-x+x^2-x^3+x^4-x^5+x^6) ]
[x^8+1, x^8+1 ]
[x^9+1, (x+1)*(x^2-x+1)*(x^6-x^3+1) ]
```

Hàm **factor** có thể phân tích các đối tượng chữ có chứa số nguyên thành thừa số. Ví dụ:

```
>>one = '1'
>>for n = 1:11
N(n,:) = sym(one(1,ones(1,n)));
end
>>[N factor(N)]
```

cho kết quả:

```
[          1,          1]
[         11,        (11)]
[        111,       (3)*(37)]
[       1111,      (11)*(101)]
[      11111,     (41)*(271)]
[     111111, (3)*(7)*(11)*(13)*(37)]
[    1111111,  (239)*(4649)]
[   11111111, (11)*(73)*(101)*(137)]
[  111111111, (3)^2*(37)*(333667)]
[ 1111111111, (11)*(41)*(271)*(9091)]
[11111111111, (513239)*(21649)]
```

+ **simplify**

Hàm **simplify** là một hàm mạnh dùng để rút gọn các biểu thức. Ví dụ như sau:

f	simplify(f)
$x*(x*(x-6)+11)-6$	$x^3-6*x^2+11*x-6$
$(1-x^2)/(1-x)$	$x + 1$
<code>syms x y positive</code> $\log(x*y)$	$\log(x) + \log(y)$
$\exp(x) * \exp(y)$	$\exp(x+y)$
$\cos(x)^2 + \sin(x)^2$	1

+ **simple**

Hàm **simple** đưa ra dạng ngắn nhất có thể có của một biểu thức. Hàm này có nhiều dạng, mỗi dạng trả về kết quả khác nhau:

Dạng **simple(f)** hiển thị dạng ngắn nhất. Ví dụ:

```
>>syms x
>>simple(cos(x)^2 + sin(x)^2)
ans =
1
```

Trong một số trường hợp, áp dụng dạng **simple** 2 lần để nhận được hiệu quả rút gọn cao hơn. Ví dụ:

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

cho ta kết quả:

$1/a+2$

Trong khi lệnh:

```
>>syms a
>>simple(f)
```

cho ta:

$(2*a+1)/a$

Hàm **simple** đặc biệt hiệu quả trên các biểu thức lượng giác, ví dụ:

f	simple(f)
$\cos(x)^2 + \sin(x)^2$	1
$2*\cos(x)^2 - \sin(x)^2$	$3*\cos(x)^2 - 1$
$\cos(x)^2 - \sin(x)^2$	$\cos(2*x)$
$\cos(x) + (-\sin(x)^2)^{(1/2)}$	$\cos(x) + i*\sin(x)$
$\cos(x) + i*\sin(x)$	$\exp(i*x)$
$\cos(3*\arccos(x))$	$4*x^3 - 3*x$

#### b) Thay số:

Có hai hàm dùng để thay số là **subexpr** và **subs**

+ **subexpr**:

```
>>syms a x
>>s = solve(x^3+a*x+1)
```

giải phương trình:  $x^3 + ax + 1 = 0$  theo  $x$ .

kết quả:

```
[ 1/6*(-108+12*(12*a^3+81)^(1/2))^(1/3)
 -2*a/(-108+12*(12*a^3+81)^(1/2))^(1/3)]
[ -1/12*(-108+12*(12*a^3+81)^(1/2))^(1/3)
 +a/(-108+12*(12*a^3+81)^(1/2))^(1/3)]
```

```

+1/2*i*3^(1/2)*(1/6*(-108+12*(12*a^3+81)^(1/2))^(1/3))
+2*a/(-108+12*(12*a^3+81)^(1/2))^(1/3)) ]
[ -1/12*(-108+12*(12*a^3+81)^(1/2))^(1/3)
+a/(-108+12*(12*a^3+81)^(1/2))^(1/3)
-1/2*i*3^(1/2)*(1/6*(-108+12*(12*a^3+81)^(1/2))^(1/3))
+2*a/(-108+12*(12*a^3+81)^(1/2))^(1/3)) ]

```

+ **subs**:

Tìm giá trị riêng và vectơ riêng của ma trận A:

```

>>syms a b c
>>A = [a b c; b c a; c a b];
>>[v,E] = eig(A)
v =
[ 1, -(a+(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)-b)/(a-c),
-(a-(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)-b)/(a-c) ]
[ 1, -(b-c-(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2))/(a-c),
-(b-c+(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2))/(a-c) ]
[ 1, 1, 1]
E =
[ b+a+c, 0, 0]
[ 0, (b^2-b*a-c*b-c*a+a^2+c^2)^(1/2), 0]
[ 0, 0, -(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2) ]

```

Giả sử ta muốn thay biểu thức khá dài:

$$(b^2-b*a-c*b-c*a+a^2+c^2)^{(1/2)}$$

Trong  $v$  và  $E$ . Trước hết ta dùng hàm **subexpr**

```
>>v = subexpr(v,'S')
```

cho ta kết quả:

```

S =
(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)
v =
[ -(a+S-b)/(a-c), -(a-S-b)/(a-c), 1]
[ -(b-c-S)/(a-c), -(b-c+S)/(a-c), 1]
[ 1, 1, 1]

```

Sau đó thay S vào E:

```

>>E = subs(E,S,'S')
E =
[ S, 0, 0]
[ 0, -S, 0]
[ 0, 0, b+c+a]

```

Bây giờ muốn tính  $v$  khi  $a = 10$ . Ta dùng lệnh sau:

```
>>subs(v,a,10)
```

sẽ thay các biến  $a$  trong  $v$  bằng số 10:

```
[ -(10+S-b)/(10-c), -(10-S-b)/(10-c), 1]
[ -(b-c-S)/(10-c), -(b-c+S)/(10-c), 1]
[ 1, 1, 1]
```

Chú ý là các biểu thức có  $S$  không bị ảnh hưởng gì cả, nghĩa là biến  $a$  trong  $S$  không được thay bằng 10. Hàm **subs** là hàm hữu ích để thay thế nhiều giá trị của nhiều biến trong cùng một biểu thức. Xét  $S$ , giả sử ngoài việc thay  $a = 10$  ta cũng muốn thay  $b=2$  và  $c=10$  vào biểu thức. Cách đơn giản nhất là đặt các giá trị  $a,b,c$ , trong workspace của MATLAB. Sau đó lệnh **subs** sẽ tính kết quả:

```
>>a = 10; b = 2; c = 10;
>>subs(S)
ans=
8
```

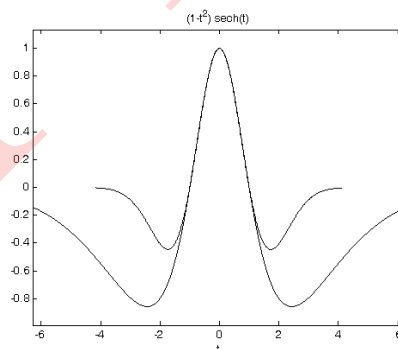
Lệnh **subs** có thể kết hợp với lệnh **double** để tính trị số của một biểu thức. Giả sử ta có:

```
>>syms t
>>M = (1-t^2)*exp(-1/2*t^2);
>>P = (1-t^2)*sech(t);
```

Và muốn xem trên đồ thị  $P$  và  $M$  khác nhau thế nào ta dùng các lệnh:

```
>>ezplot(M);
>>hold on;
>>ezplot(P)
```

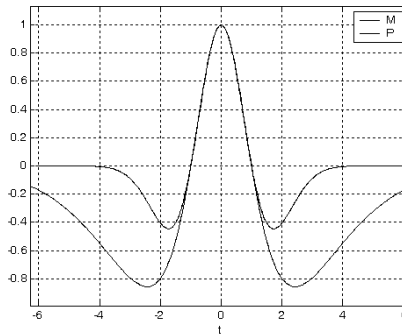
và đồ thị như hình vẽ dưới.



Tuy nhiên ta vẫn khó hình dung được sự sai khác giữa hai đường cong. Vì vậy tốt hơn chúng ta kết hợp **subs**, **double** lại.

```
>>T = -6:0.05:6;
>>MT = double(subs(M,t,T));
>>PT = double(subs(P,t,T));
>>plot(T,MT,'b',T,PT,'r-.')
>>title(' ')
>>legend('M','P')
```

```
>>xlabel('t');
>>grid
```



### 3.2.3.5. Giải phương trình

#### a) Giải các phương trình đại số

Nếu S là một biểu thức dạng chữ thì:

```
>>solve(S)
```

sẽ tìm giá trị của biến chữ trong S để  $S=0$ . Ví dụ

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

```
>>S = a*x^2 + b*x + c;
```

```
>>solve(S)
```

cho ta:

```
ans =
```

```
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
```

```
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

Đây là véc tơ chữ mà các phần tử của nó là 2 nghiệm của phương trình.

Nếu ta muốn tìm nghiệm với một biến, ta phải chỉ rõ biến trong tùy chọn đầu vào. Ví dụ ta muốn giải S theo b thì ta viết:

```
>>b = solve(S,b)
```

và nhận được kết quả:

```
b =
```

```
-(a*x^2+c)/x
```

Chú ý ví dụ trên giả thiết phương trình có dạng  $f(x)=0$ . Nếu ta muốn giải phương trình có dạng  $f(x)=q(x)$  ta phải sử dụng chuỗi. Đặc biệt là lệnh:

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

cho 4 nghiệm:

```
s =
```

```
[ 0]
```

```
[ pi]
```

```
[ 1/6*pi]
```

```
[ 5/6*pi]
```

Phương trình  $x^3 - 2x^2 = x - 1$  giúp ta hiểu cách giải phương trình.

```
>>s = solve('x^3-2*x^2 = x-1')
```

Ta tính giá trị số của nghiệm:

```
>>double(s)
```

```
ans =
```

```
2.24697960371747 + 0.000000000000000i
```

```
-0.80193773580484 + 0.000000000000000i
```

```
0.55495813208737 - 0.000000000000000i
```

Ta thấy các nghiệm của phương trình là số thực, điều này không đúng. Để biết chính xác nghiệm ta dùng lệnh:

```
vpa(s, 10)
```

tạo ra:

```
ans =
```

```
[ 2.246979604+.1e-9*i] [ -.8019377357+.3e-9*i]
```

```
[ .5549581323-.5e-9*i]
```

Điều này nghĩa là phần ảo của s rất nhỏ nhưng khác 0. Xét ví dụ khác:

```
>>syms x
```

```
>>s = solve('tan(x)+sin(x)=2');
```

Kết quả là một vectơ 4×1. Như trên ta dùng lệnh double:

```
>>X = double(s)
```

```
X =
```

```
0.8863
```

```
-1.8979
```

```
2.0766 - 1.5151i
```

```
2.0766 + 1.5151i
```

### ***b) Hệ phương trình đại số***

Xét hệ phương trình:

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

Và ta cần tìm  $x$  và  $y$ . Trước hết tạo các đối tượng cần thiết:

```
>>syms x y alpha
```

Có nhiều cách để biểu diễn nghiệm. Một trong các cách đó là viết:

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

Và kết quả là:

```
x =
```

```

[      0]
[      0]
[ alpha]
[ alpha]
y =
[ -2*alpha]
[ -2*alpha]
[      0]
[      0]

```

Sau đó viết vectơ nghiệm:

```
>>v = [x, y]
```

cho ta:

```

v =
[      0, -2*alpha]
[      0, -2*alpha]
[  alpha,      0]
[  alpha,      0]

```

Cách gán các nghiệm như trên chỉ thích hợp với hệ có ít phương trình. Với hệ có nhiều phương trình, hàm ***solve*** tạo ra một cấu trúc mà các trường của nó là các nghiệm. Khảo sát hệ phương trình sau:

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

Dùng lệnh:

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

Cho kết quả:

```

S =
a: [2x1 sym]
u: [2x1 sym]
v: [2x1 sym]

```

Các nghiệm là các trường của S. Đó là:

```
>>S.a
```

Tạo ra:

```

ans =
[-1]
[ 3]

```

Tương tự tìm được nghiệm  $u$  và  $v$ .

Cấu trúc S bây giờ có thể được xử lý bằng trường và chỉ số để truy cập đến các phần riêng biệt của nghiệm. Ví dụ nếu ta muốn kiểm tra nghiệm thứ 2, ta có thể dùng lệnh sau:

```
>>s2 = [S.a(2), S.u(2), S.v(2)]
```

để trích thành phần thứ 2 của mỗi trường:

```
s2 =  
[3, 5, -4]
```

Lệnh sau:

```
>>M = [S.a, S.u, S.v]
```

Tạo ra ma trận nghiệm M:

```
M =  
[-1, 1, 0]  
[ 3, 5, -4]
```

Mà mỗi hàng là một nghiệm của hệ.

Nếu hệ phương trình là tuyến tính, ta có thể dùng ma trận để giải:

```
>>clear u v x y  
>>syms u v x y  
>>S = solve('x+2*y=u', '4*x+5*y=v');
```

và:

```
>>sol = [S.x;S.y]  
>>A = [1 2; 4 5];  
>>b = [u; v];  
>>z = A\b
```

cho:

```
sol =  
[ -5/3*u+2/3*v]  
[ 4/3*u-1/3*v]  
z =  
[ -5/3*u+2/3*v]  
[ 4/3*u-1/3*v]
```

### 3.3. GIAO DIỆN NGƯỜI DÙNG GUI

#### 3.3.1. Giới thiệu về GUI

Graphical User Interface (GUI) là giao diện bằng hình ảnh của chương trình, nó có thể bao gồm một hay nhiều cửa sổ đồ họa chứa các điều khiển (hay các *thành phần*) cho phép người sử dụng thực hiện các nhiệm vụ tương tác. Người dùng GUI không cần tạo các script hoặc gõ lệnh tại cửa sổ lệnh để hoàn thành tác vụ. Không giống như code của chương trình, khi dùng GUI không cần nắm cụ thể cách mà tác vụ được thực hiện.



Các thành phần GUI có thể bao gồm: menu, toolbar, push buttons, radio buttons, list boxes, and sliders. Các GUI được tạo ra bằng các tool của MATLAB, có thể thực hiện bất kỳ tính toán nào, đọc và ghi file dữ liệu, liên hệ với các GUI khác, và hiển thị dữ liệu theo bảng hoặc đồ thị.

Khi thiết kế GUI nên theo một cách dễ hiểu và thân thiện để người sử dụng có thể hiểu và dự đoán được kết quả của một tác động.

### 3.3.2. Cấu trúc và các đối tượng đồ họa

GUI cung cấp cho người sử dụng một môi trường làm việc thân thiện để họ tập trung vào các ứng dụng của chương trình hơn là đi tìm hiểu cách thức làm việc của chương trình. Tuy nhiên, tạo ra GUI là công việc khó khăn đối với người lập trình bởi vì chương trình phải được xử lý với các click chuột cho bất kỳ thành phần nào của GUI và vào bất kỳ thời điểm nào. Trong MATLAB, để tạo ra một GUI lưu ý ba yêu cầu chính sau đây:

- **Component** (các thành phần): mỗi đối tượng trong GUI (nút nhấn, nhãn, hộp soạn thảo, ...) là một thành phần. Các thành phần được phân loại thành: công cụ điều khiển (nút nhấn, hộp soạn thảo, thanh trượt, ...), các thành phần tĩnh (khung hình, chuỗi ký tự, ...), menu và axes (là các hệ trục dùng để hiển thị đồ họa). Các công cụ điều khiển và các thành phần tĩnh được tạo ra bởi hàm `uicontrol`, menu được tạo ra bởi các hàm `uimenu` và `uicontextmenu`, axes được tạo ra bởi hàm `axes`.

- **Figure**: các thành phần của GUI phải được sắp xếp vào trong một figure, là một cửa sổ được hiển thị trên màn hình máy vi tính. Trong các chương trước, một figure được tự động tạo ra khi vẽ đồ thị. Lệnh `figure` tạo ra một figure được sử dụng để chứa các thành phần của GUI.

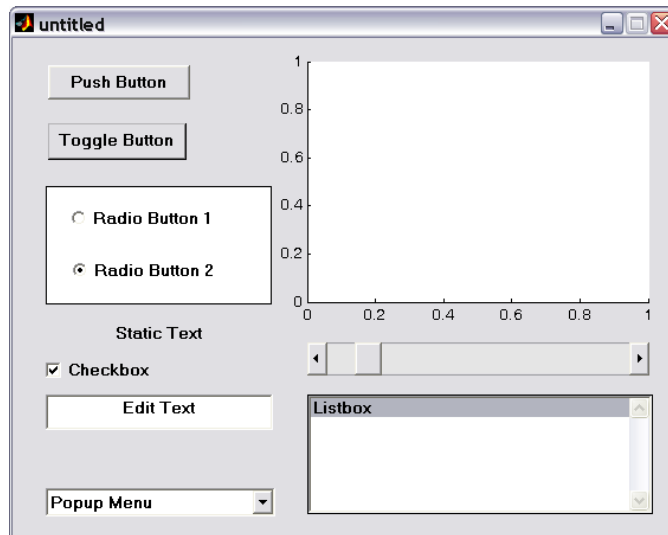
- **Callback**: cuối cùng, khi người sử dụng tác động vào chương trình bằng cách nhấn chuột hay gõ bàn phím thì chương trình phải đáp ứng lại mỗi sự kiện này. Ví dụ, trong trường hợp người sử dụng tác động vào một nút nhấn thì MATLAB sẽ thực thi một hàm tương ứng với nút nhấn đó. Mỗi thành phần của GUI phải *callback* một hàm tương ứng của nó.

Các thành phần cơ bản của GUI được tóm tắt trong bảng sau và được thể hiện trong Hình 3.11. Chúng ta sẽ tìm hiểu những thành phần này thông qua các ví dụ và sau đó sử dụng chúng để tạo ra các GUI.

**Bảng 3.6. Mô tả một số thành phần của GUI**

Công cụ	Tạo bởi hàm	Mô tả
<b>Các công cụ điều khiển</b>		
Pushbutton	<code>uicontrol</code>	Là một nút nhấn. Nó sẽ gọi hàm khi nhấn vào nó.
Toggle button	<code>uicontrol</code>	Là nút nhấn có hai trạng thái là “on” và “off”. Khi có tác động nó sẽ gọi hàm tương ứng và thay đổi trạng thái từ “on” sang “off” hoặc ngược lại.

Radio button	uicontrol	Cũng là một nút nhấn có hai trạng thái được thể hiện bởi một vòng tròn nhỏ, trạng thái “on” tương ứng với trường hợp có dấu chấm giữa vòng tròn và ngược lại là trạng thái “off”. Trong một nhóm Radio button ta chỉ có thể chọn được một thành phần. Khi có tác động vào mỗi thành phần sẽ có một hàm được gọi.
Check box	uicontrol	Cũng là một nút nhấn có hai trạng thái được thể hiện bởi một hình vuông nhỏ, trạng thái “on” tương ứng với trường hợp có đánh dấu giữa hình vuông và ngược lại là trạng thái “off”. Khi có tác động nó sẽ gọi hàm tương ứng và thay đổi trạng thái từ “on” sang “off” hoặc ngược lại.
List box	uicontrol	Là một danh sách các chuỗi. Người sử dụng có thể chọn một chuỗi bằng cách click hoặc double click vào nó. Chương trình sẽ gọi một hàm khi có một chuỗi được chọn.
Popup menus	uicontrol	Là công cụ cho phép chúng ta chọn một chuỗi trong một nhóm các chuỗi. Danh sách tất cả các chuỗi sẽ được hiển thị khi có click chuột. Khi không có click chuột công cụ chỉ thể hiện chuỗi hiện tại được chọn.
Slider	uicontrol	Là công cụ cho phép điều chỉnh một cách liên tục giá trị trong một thanh trượt. Mỗi khi giá trị của thanh trượt thay đổi sẽ có hàm được gọi.
<b><i>Các thành phần tĩnh</i></b>		
Frame	uicontrol	Được sử dụng để tạo ra một khung hình chữ nhật. Frame còn được sử dụng để nhóm các công cụ điều khiển lại với nhau. Frame không có khả năng gọi hàm.
Text field	uicontrol	Được sử dụng để tạo ra một nhãn bao gồm các ký tự. Text field không có khả năng gọi hàm.
<b><i>Menu và trục đồ thị</i></b>		
Menu items	uicontrol	Được sử dụng để tạo ra menu trên thanh công cụ. Chương trình sẽ gọi hàm khi một đối tượng trong menu được chọn.
Context menus	uicontextmenu	Được sử dụng để tạo ra menu xuất hiện khi right-click vào một hình trong giao diện.
Axes	axes	Được sử dụng để tạo một hệ trục đồ thị. Axes không có khả năng gọi hàm.



**Hình 3.11. Một ví dụ thiết kế giao diện đồ họa**

Mỗi đối tượng của GUI đều có một số thuộc tính mà người sử dụng có thể thay đổi tùy ý. Các thuộc tính có một số điểm khác nhau tùy vào từng đối tượng (figures, axes, uicontrol,...).

Đặc tính và cách ứng dụng của tất cả các thuộc tính này đều được đề cập đến một cách chi tiết trong phần Help của MATLAB. Bảng 3.7 liệt kê một số đặc tính quan trọng nhất của figure và uicontrol.

Giá trị của các thuộc tính này có thể thay đổi bằng Property Inspector hoặc thay đổi trực tiếp bằng cách sử dụng lệnh *get* hay *set*. Trong trường hợp thiết kế một giao diện ta nên sử dụng "Property Inspector" để thay đổi thuộc tính của các đối tượng còn trong phạm vi của một chương trình thì chúng ta phải sử dụng lệnh *get* hay *set*.

**Bảng 3.7. Một số thuộc tính cơ bản của figure**

Tên thuộc tính	Mô tả	Giá trị
Position	Vị trí và kích thước của figure	<i>Value</i> : là một véc tơ gồm 4 thành phần [left, bottom, width, height] <i>Default</i> : phụ thuộc vào trạng thái hiện tại
Units	Đơn vị của các thành phần trong <i>Position</i> .	<i>Values</i> : inches, centimeters, normalized, points, pixels, characters <i>Default</i> : pixels
Color	Màu nền của figure.	<i>Values</i> : <i>Default</i> : phụ thuộc vào màu hiện tại của hệ thống.
MenuBar	Tắt hay hiện thanh công cụ	<i>Values</i> : none, figure <i>Default</i> : figure
Name	Tiêu đề của figure	<i>Values</i> : chuỗi <i>Default</i> : " (chuỗi rỗng)

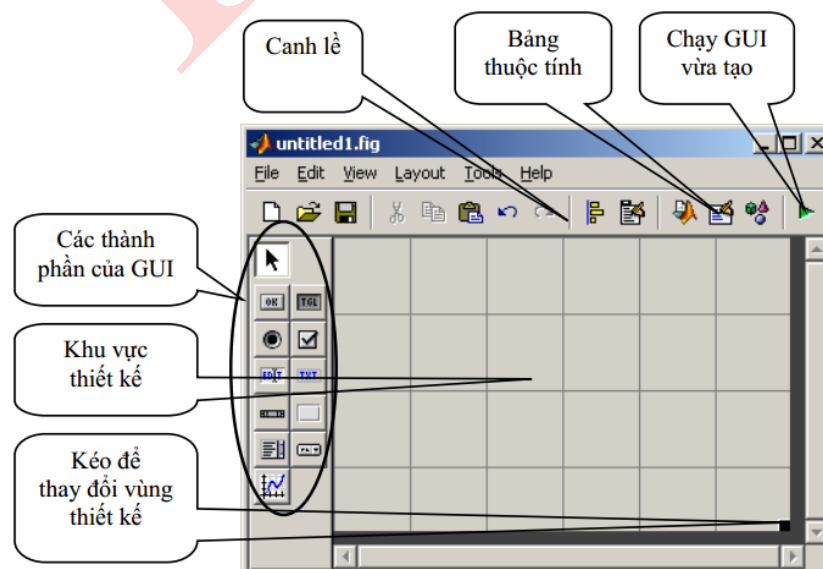
NumberTitle	Hiển thị "Figure No. n", với $n$ là số thứ tự figure	<i>Values:</i> on, off <i>Default:</i> on
Resize	Cửa sổ hình có thể thay đổi kích thước được hay không.	<i>Values:</i> on, off <i>Default:</i> on
Pointer	Chọn kiểu hình ảnh dấu nhắc chuột	<i>Values:</i> crosshair, arrow, watch, topl, topr, botl, botr, circle, cross, fleur, left, right, top, bottom, fullcrosshair, ibeam, custom <i>Default:</i> arrow
BackgroundColor	Màu nền của đối tượng	<i>Value:</i> <i>Default:</i> phụ thuộc vào hệ thống
ForegroundColor	Màu chữ	<i>Value:</i> <i>Default:</i> [0 0 0]
String	Nhãn của Uicontrol	<i>Value:</i> string
Enable	Enable hay disable uicontrol	<i>Value:</i> on, inactive, off <i>Default:</i> on
Style	Kiểu của đối tượng	<i>Value:</i> pushbutton, togglebutton, radiobutton, checkbox, edit, text, slider, frame, listbox, popupmenu <i>Default:</i> pushbutton
Position	Kích thước và vị trí của đối tượng	<i>Value:</i> ma trận 1x4 <i>Default:</i> [20 20 60 20]
Units	Đơn vị trong vectơ vị trí	<i>Value:</i> pixels, normalized, inches, centimeters, points, characters <i>Default:</i> pixels
FontAngle	Trạng thái của ký tự	<i>Value:</i> normal, italic, oblique <i>Default:</i> normal
FontName	Font chữ	<i>Value:</i> chuỗi <i>Default:</i> phụ thuộc vào hệ thống
FontSize	Kích thước chữ	<i>Value:</i> kích thước trong FontUnits <i>Default:</i> phụ thuộc vào hệ thống
FontWeight	Độ đậm nhạt của ký tự	<i>Value:</i> light, normal, demi, bold <i>Default:</i> normal
HorizontalAlignment	Canh lề	<i>Value:</i> left, center, right <i>Default:</i> phụ thuộc vào đối tượng uicontrol
Callback	Hoạt động điều khiển	<i>Value:</i> string

Max	Giá trị lớn nhất	<i>Value:</i> vô hướng <i>Default:</i> phụ thuộc đối tượng
Min	Giá trị nhỏ nhất	<i>Value:</i> vô hướng <i>Default:</i> phụ thuộc đối tượng
Value	Giá trị hiện tại của đối tượng	<i>Value:</i> vô hướng hay vectơ <i>Default:</i> phụ thuộc đối tượng

### 3.3.3. Tạo các uicontrols

Trong MATLAB, công cụ guide được sử dụng để tạo ra các GUI, công cụ này cho phép bố trí, lựa chọn và sắp xếp các thành phần. Các thuộc tính của mỗi thành phần: tên, màu sắc, kích cỡ, font chữ, ... đều có thể được thay đổi. Công cụ guide được thực thi bằng cách chọn File – New – GUI, khi được gọi guide sẽ tạo ra một Layout Editor, như Hình 3.12. Vùng màu xám có những đường kẻ là vùng làm việc, trong vùng này chúng ta có thể sắp xếp các thành phần để tạo nên giao diện. Ở bên trái vùng làm việc là các thành phần có sẵn trong GUI. Chúng ta có thể tạo ra bất kỳ thành phần nào bằng cách click vào biểu tượng của nó, sau đó kéo vào thả vào vùng làm việc. Bên trên vùng làm việc là thanh công cụ bao gồm các công cụ thường sử dụng. Sau đây là các bước cơ bản để tạo ra một giao diện trong MATLAB:

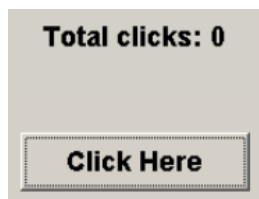
- Xác định các thành phần của giao diện và hàm bị tác động bởi mỗi thành phần. Phác họa trên giấy vị trí của các thành phần.
- Mở công cụ tạo GUI (File – New – GUI), sắp xếp các thành phần vào vùng làm việc. Nếu cần thiết, thay đổi kích thước của vùng làm việc cũng như của mỗi thành phần.
- Thiết lập thuộc tính của mỗi thành phần chẳng hạn như: tên, màu sắc, hiển thị,...
- Lưu giao diện vừa tạo. Khi lưu một giao diện MATLAB sẽ tạo ra hai file có cùng tên nhưng khác phần mở rộng. File có phần mở rộng **'fig'** chứa nội dung của giao diện, trong khi file có phần mở rộng **'m'** chứa những đoạn mã liên quan đến giao diện.
- Viết hàm để thực thi lệnh gọi của mỗi thành phần trong giao diện.



Hình 3.12. Cửa sổ tạo giao diện GUI

Để hiểu rõ hơn hãy xét ví dụ sau đây. Trong ví dụ này, giao diện đơn giản chỉ gồm một nút nhấn và một chuỗi ký tự. Mỗi khi nhấn vào nút nhấn dòng ký tự sẽ được cập nhật và hiển thị số lần nhấn vào nút nhấn.

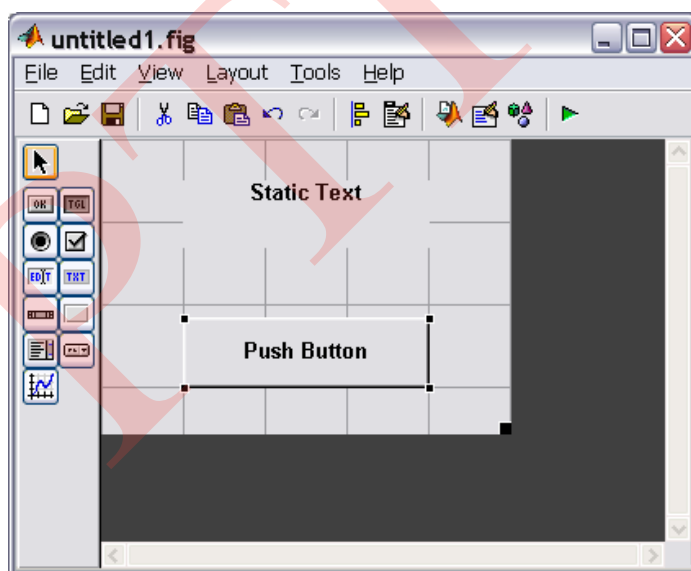
**Bước 1:** Trong ví dụ này, giao diện chỉ bao gồm một nút nhấn và một dòng ký tự. Hàm được gọi từ nút nhấn sẽ cập nhật chuỗi ký tự và thể hiện số lần nhấn vào nút nhấn. Phác họa giao diện như Hình 3.13.



Hình 3.13. Phác họa của một giao diện đơn giản

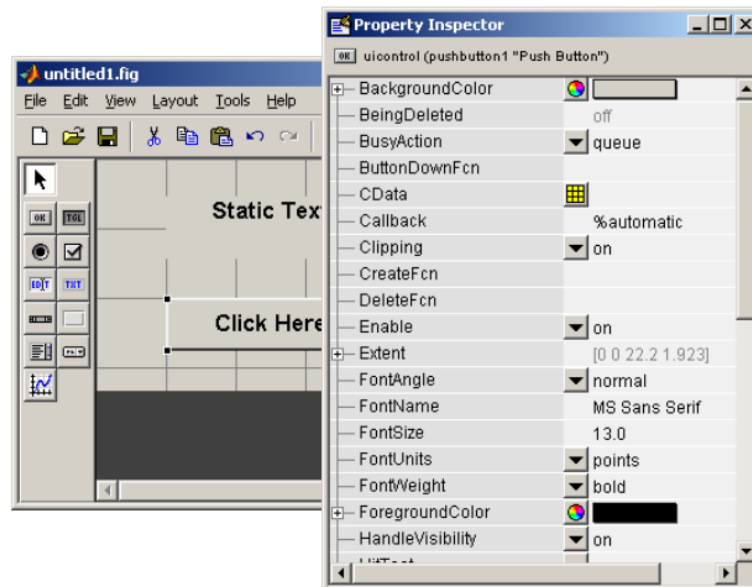
**Bước 2:** Mở chương trình guide để bắt đầu bố trí các thành phần vào cửa sổ làm việc. Chương trình guide sẽ mở ra một cửa sổ như Hình 3.12.

Đầu tiên chúng ta phải thay đổi kích thước của vùng làm việc, đây chính là kích thước của toàn bộ giao diện. Kích thước của vùng làm việc được thay đổi bằng cách kéo hình vuông nhỏ ở góc dưới bên phải của cửa sổ làm việc. Sau đó chúng ta tạo ra một nút nhấn bằng cách nhấn vào biểu tượng nút nhấn (push button), kéo và thả vào vùng làm việc. Làm tương tự để tạo ra một chuỗi ký tự (static text). Sau khi thực hiện bước này ta có kết quả như sau:



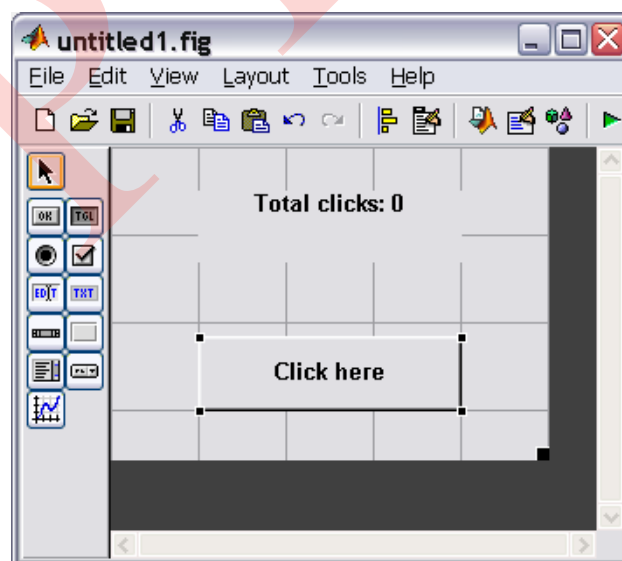
Hình 3.14. Cửa sổ thể hiện nội dung của giao diện

**Bước 3:** Để thiết lập thuộc tính cho nút nhấn, đầu tiên ta phải chọn nút nhấn cần thiết lập các thuộc tính, sau đó chọn mục “Property Inspector” trong thanh công cụ. Ta cũng có thể tiến hành như sau, right-click vào đối tượng và chọn mục “Inspect Properties”, hoặc kích đúp vào đối tượng. Cửa sổ “Property Inspector” được thể hiện như Hình 3.15. Cửa sổ này liệt kê tất cả các thuộc tính của nút nhấn và cho phép chúng ta thay đổi giá trị của các thuộc tính này.



Hình 3.15. Cửa sổ thể hiện các thuộc tính của nút nhấn.

Với nút nhấn chúng ta có thể thay đổi nhiều thuộc tính chẳng hạn như màu sắc, kích cỡ, font chữ,.... Tuy nhiên chúng ta cần phải xác lập hai thuộc tính **String property**, là dòng ký tự xuất hiện trên nút nhấn và thuộc tính thứ hai cần xác lập là **Tag property**, là tên của nút nhấn. Trong trường hợp này String property được thiết lập là 'Click Here' và Tag property được thiết lập là 'MyFirstButton'. Đối với đối tượng chuỗi ký tự, chúng ta cũng thiết lập hai thuộc tính: **String property** là chuỗi ký tự xuất hiện trên giao diện và Tag property là tên của đối tượng chuỗi. Tên của đối tượng chuỗi là yêu cầu cần thiết trong quá trình gọi hàm để cập nhật nội dung của chuỗi. Trong trường hợp này String property được thiết lập là chuỗi '**Total clicks: 0**' và Tag property được thiết lập là '**MyFirstText**'. Sau khi thực hiện các bước ở trên ta có Hình 3.16.

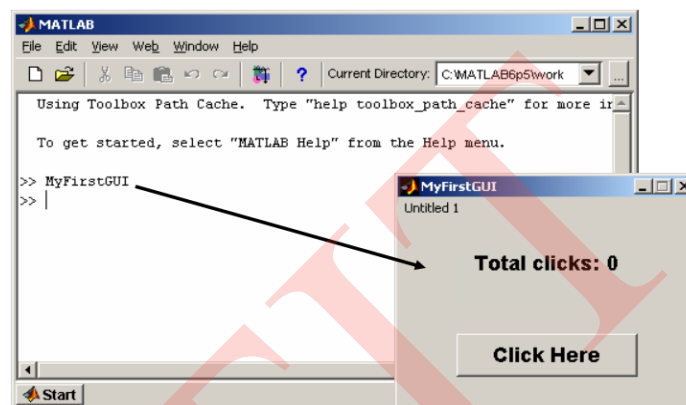


Hình 3.16. Vùng thiết kế sau khi cài đặt các thuộc tính

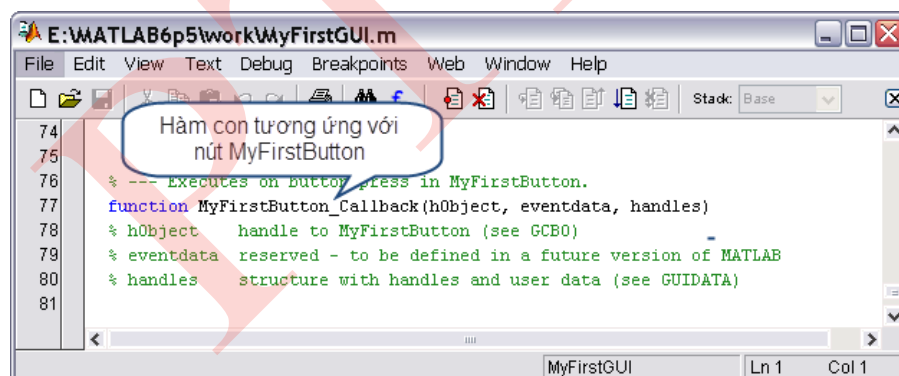
**Bước 4:** Lưu giao diện vừa tạo với tên MyFirstGUI. Sau khi lưu chương trình sẽ tạo ra hai file MyFirstGUI.fig và MyFirstGUI.m. Đến đây chúng ta đã tạo xong giao diện nhưng chưa hoàn thành bài tập như ý tưởng ban đầu. Chúng ta có thể bắt đầu chương trình bằng cách gõ lệnh MyFirstGUI trong cửa sổ lệnh, kết quả như Hình 3.17. Chương trình sẽ không có hoạt



động gì khi ta nhấn vào nút nhấn bởi vì ta chưa lập trình cho những hàm được gọi. Một phần của file MyFirstGUI.m được trình bày trong Hình 3.18. File này chứa hàm MyFirstGUI và một số hàm con tương ứng với tác động của mỗi thành phần trong giao diện. Nếu gọi hàm MyFirstGUI trong trường hợp không có đối số ngõ vào thì nội dung của file MyFirstGUI.fig được thể hiện. Trong trường hợp có đối số ngõ vào khi gọi hàm MyFirstGUI.fig thì đối số đầu tiên là tên của hàm con và các đối số còn lại sẽ được đưa đến hàm con đó. Mỗi hàm con được gọi sẽ tương ứng với một đối tượng trong giao diện. Khi click chuột vào một đối tượng thì MATLAB sẽ gọi hàm con tương ứng với đối tượng đó. Tên của hàm được gọi sẽ là giá trị **Tag property** của đối tượng tương ứng. Tên của hàm chính là tên của đối tượng cộng với chuỗi ký tự ‘\_Callback’. Như vậy hàm con tương ứng với nút nhấn MyFirstButton sẽ là MyFirstButton\_Callback. File .m của chương trình sẽ tạo ra các hàm con tương ứng với tất cả các đối tượng.



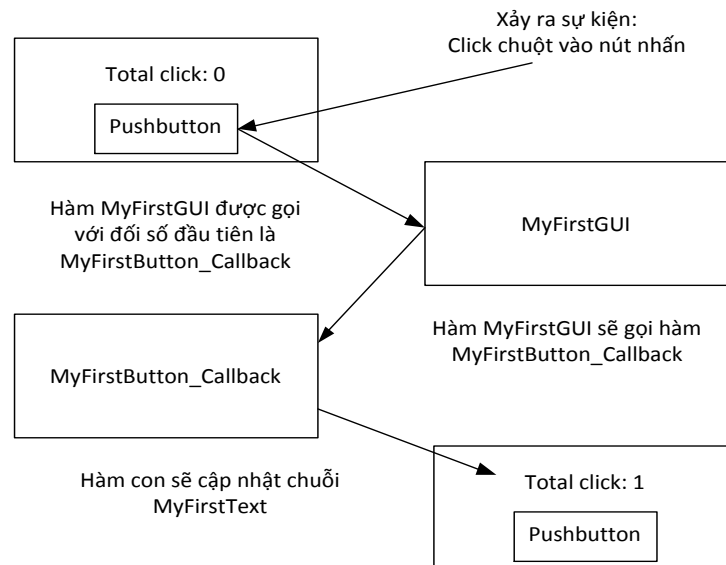
Hình 3.17. Gõ MyFirstGUI trong cửa sổ lệnh để bắt đầu chương trình



Hình 3.18. Một phần nội dung của file .m

**Bước 5:** Trong bước này, chúng ta sẽ lập trình cho hàm con tương ứng với nút nhấn. Trong hàm này chúng ta sẽ sử dụng một biến **persistent** (là biến mà giá trị của nó sẽ được nhớ giữa những lần gọi hàm) để đếm số click chuột vào nút nhấn. Khi có click chuột vào nút nhấn, MATLAB sẽ gọi hàm MyFirstGUI với đối số đầu tiên là MyFirstButton\_callback. Sau đó hàm MyFirstGUI sẽ gọi hàm con MyFirstButton\_callback, quá trình này được thể hiện trong Hình 3.19. Trong hàm con, giá trị biến đếm sẽ tăng lên khi có click chuột, một chuỗi mới chứa giá trị biến đếm sẽ tạo ra và được lưu vào trong thuộc tính String property của đối tượng MyFirstText.





**Hình 3.19. Quá trình gọi hàm và cập nhật giá trị đếm**

Nội dung của hàm con được lập trình như sau:

```
function MyFirstButton_Callback(hObject, eventdata, handles)
% Khai báo và thiết lập giá trị ban đầu cho biến đếm count
persistent count
if isempty(count)
count = 0;
end
% Cập nhật giá trị biến đếm
count = count + 1;
% Tạo một chuỗi mới
str = sprintf('Total clicks: %d', count);
% Cập nhật giá trị của đối tượng chuỗi
set(handles.MyFirstText, 'String', str);
```

Lưu ý rằng hàm con khai báo biến count có dạng biến “nhớ” và gán giá trị ban đầu cho nó là 0. Kết quả của giao diện sau khi click chuột bốn lần vào nút nhấn như Hình 3.20.



**Hình 3.20. Kết quả sau khi nhấn nút bốn lần**

### 3.3.4. Điều khiển với uicontrols

Phần này miêu tả cách tạo lập và sử dụng các thành phần đối tượng trong GUI:

- Text Fields

- Edit Boxes
- Frames
- Pushbuttons
- Toggle Buttons
- Checkboxes
- Radio Buttons
- Popup Menus
- List Boxes
- Slide

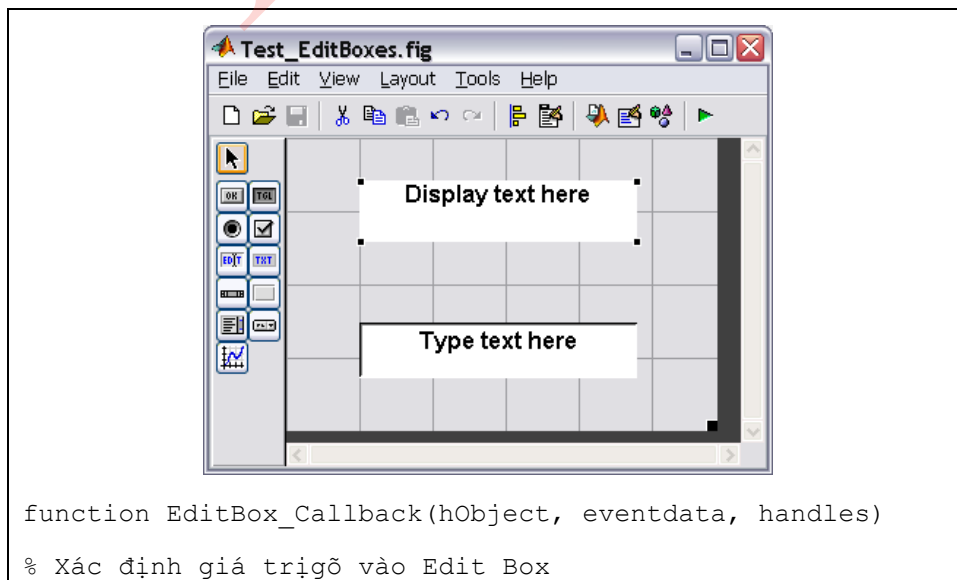
#### a) Text Fields

Text Field là một đối tượng được sử dụng để thể hiện chuỗi ký tự trên màn hình. Chúng ta có thể xác định thuộc tính canh lề của chuỗi ký tự. Mặc định, chuỗi ký tự sẽ được canh ở giữa. Text Field được tạo ra bằng cách sử dụng hàm **uicontrol** với thuộc tính **style** là 'text' hay cũng có thể tạo ra bởi công cụ **text** trong Layout Editor. Text Field không có khả năng gọi hàm nhưng giá trị của nó có thể được cập nhật trong một hàm bằng cách thay đổi thuộc tính String của Text Field.

#### b) Edit Boxes

Edit Box là một đối tượng cho phép người sử dụng nhập vào một chuỗi ký tự. Sau khi nhập xong chuỗi ký tự và nhấn Enter Edit Box sẽ gọi hàm tương ứng của nó. Đối tượng này được tạo ra bằng cách sử dụng hàm **uicontrol** với thuộc tính **style** là 'edit' hay cũng có thể tạo ra bởi công cụ **edit box** trong Layout Editor.

Hình 3.21 thể hiện một giao diện đơn giản gồm một Edit Box có tên 'EditBox' và một Text Field có tên là 'TextBox'. Khi người sử dụng nhập vào khung Edit Box thì nó tự động gọi hàm `EditBox_Callback`. Chương trình này có nhiệm vụ xác định chuỗi ký tự được nhập vào, sau đó gán chuỗi này cho Text Field và hiển thị lên màn hình giao diện. Hình 3.22 là giao diện của chương trình khi bắt đầu và sau khi nhập 'PTIT' vào Edit Box.



```

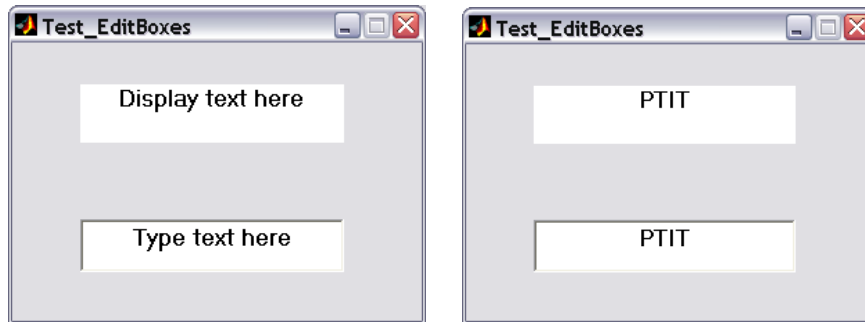
str=get(handles.EditBox,'string');

%Gán giá trị vào Text Field

set(handles.Text,'string',str);

```

Hình 3.21. Ví dụ về Edit Box và nội dung của chương trình con được gọi



Hình 3.22. Giao diện của chương trình trước và sau khi gõ 'PTIT' vào Edit Box

### c) Frames

Frame là một đối tượng được sử dụng để vẽ hình chữ nhật, nó còn được sử dụng để nhóm các đối tượng có liên quan lại với nhau. Ví dụ một Frame được sử dụng để nhóm các Radio Button trong Hình 3.11 lại với nhau. Một Frame được tạo bằng cách sử dụng hàm **uicontrol** với đối tượng **style** là **'frame'** hay cũng có thể tạo ra bởi công cụ frame trong Layout Editor.

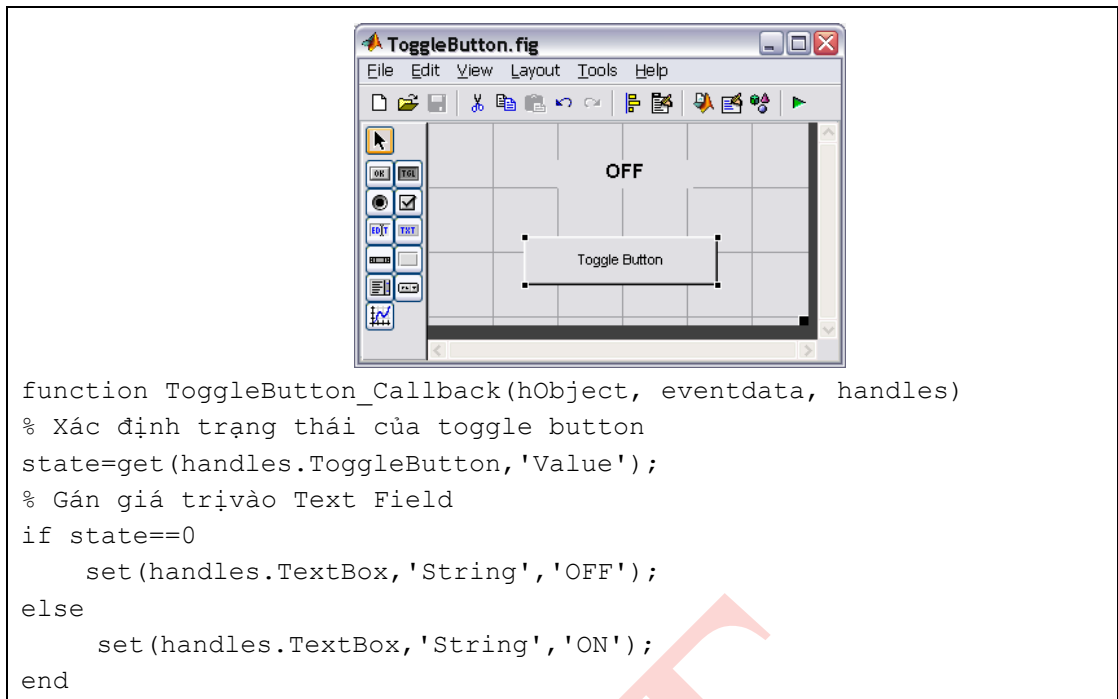
### d) Pushbuttons

PushButton là một đối tượng cho phép người sử dụng kích hoạt một tác vụ bằng cách click chuột vào nó. Đối tượng này được tạo ra bởi hàm **uicontrol** với thuộc tính **style** là **'pushbutton'** hay cũng có thể tạo ra bởi công cụ pushbutton trong Layout Editor. Một ví dụ về hoạt động của PushButton được miêu tả bởi hàm MyFirstGUI trong Hình 3.20.

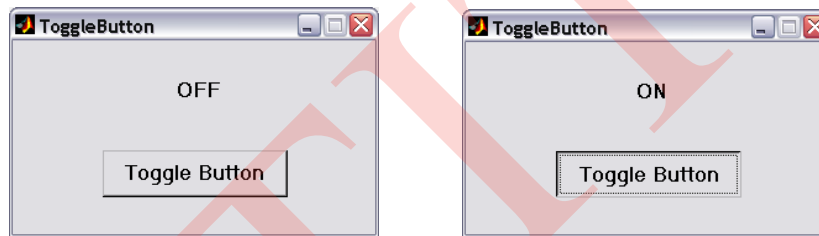
### e) Toggle Buttons

Toggle Button là một kiểu nút nhấn có hai trạng thái on và off tương ứng với trường hợp bị nhấn xuống hay không bị nhấn xuống. Mỗi khi có click chuột vào nó, Toggle Button sẽ thay đổi trạng thái và gọi hàm tương ứng của nó. Thuộc tính Value của Toggle Button được xác định bởi giá trị lớn nhất (thường là 1) khi nó ở trạng thái on và được xác định bởi giá trị nhỏ nhất (thường là 0) khi nó ở trạng thái off. Kiểu nút nhấn này được tạo ra bằng hàm **uicontrol** với thuộc tính **style** là **'toggle button'** hay cũng có thể tạo ra bởi công cụ toggle button trong Layout Editor.

Hình 3.23 trình bày một giao diện đơn giản bao gồm một Toggle Button tên 'ToggleButton' và một Text Field tên 'TextBox'. Khi người sử dụng click vào Toggle Button nó sẽ tự động gọi hàm ToggleButton\_Callback. Hàm này xác định trạng thái của nút nhấn từ thuộc tính 'Value', sau đó chương trình sẽ thể hiện trạng thái của nút nhấn bởi Text Field. Hình 3.24 thể hiện giao diện trước và sau khi có click chuột vào Toggle Button.



Hình 3.23. Ví dụ về Toggle Button và nội dung của chương trình con được gọi

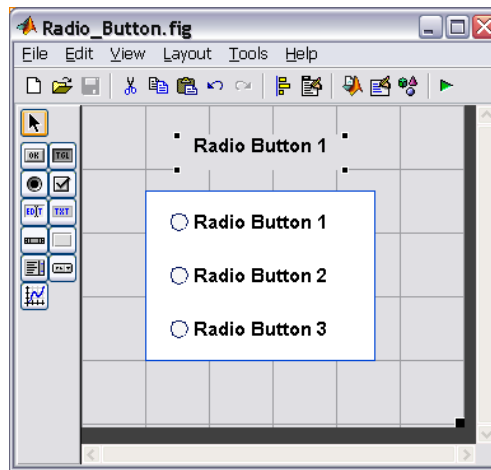


Hình 3.24. Hai trạng thái ON và OFF của Toggle Button

#### f) Checkboxes và Radio Buttons

Về cơ bản Checkbox và Radio Button là tương tự với Toggle Button ngoại trừ hình dạng khác nhau của chúng. Giống như Toggle Button, Checkbox và Radio Button cũng có hai trạng thái ON và OFF. Trạng thái của chúng sẽ thay đổi mỗi khi có click chuột và một hàm tương ứng sẽ được gọi. Thuộc tính Value của Checkbox và Radio Button được xác định bởi giá trị lớn nhất (thường là 1) khi nó ở trạng thái on và được xác định bởi giá trị nhỏ nhất (thường là 0) khi nó ở trạng thái off. Hình dạng của Checkbox và Radio Button được mô tả trong Hình 3.11.

Hai đối tượng này có thể được tạo ra từ **Layout Editor** hay sử dụng hàm uicontrol với thuộc tính **style** tương ứng là **'checkbox'** và **'radiobutton'**. Thông thường CheckBox được sử dụng trong các lựa chọn on/off và một nhóm Radio Button được sử dụng để xác định một trong số các lựa chọn. Hình 3.25 thể hiện một ví dụ về cách sử dụng Radio Button và hàm được gọi tương ứng. Trong ví dụ này, chương trình sẽ tạo ra ba Radio Button có nhãn là 'RadioButton 1', 'RadioButton 2' và 'RadioButton 3'. Khi người sử dụng click vào một Radio Button thì hàm con tương ứng sẽ được thực thi, hàm này sẽ thể hiện đối tượng được lựa chọn, mở Radio Button hiện tại và tắt tất cả các Radio Button khác. Lưu ý rằng một Frame được sử dụng để nhóm các Radio Button này lại với nhau. Hình 3.26 thể hiện giao diện khi đối tượng thứ hai được chọn.



```
function radiobutton1_Callback(hObject, eventdata, handles)
set(handles.TextBox,'String','Radio Button 1');
set(handles.radiobutton1,'Value',1);
set(handles.radiobutton2,'Value',0);
set(handles.radiobutton3,'Value',0);

function radiobutton2_Callback(hObject, eventdata, handles)
set(handles.TextBox,'String','Radio Button 2');
set(handles.radiobutton1,'Value',0);
set(handles.radiobutton2,'Value',1);
set(handles.radiobutton3,'Value',0);

function radiobutton3_Callback(hObject, eventdata, handles)
set(handles.TextBox,'String','Radio Button 3');
set(handles.radiobutton1,'Value',0);
set(handles.radiobutton2,'Value',0);
set(handles.radiobutton3,'Value',1);
```

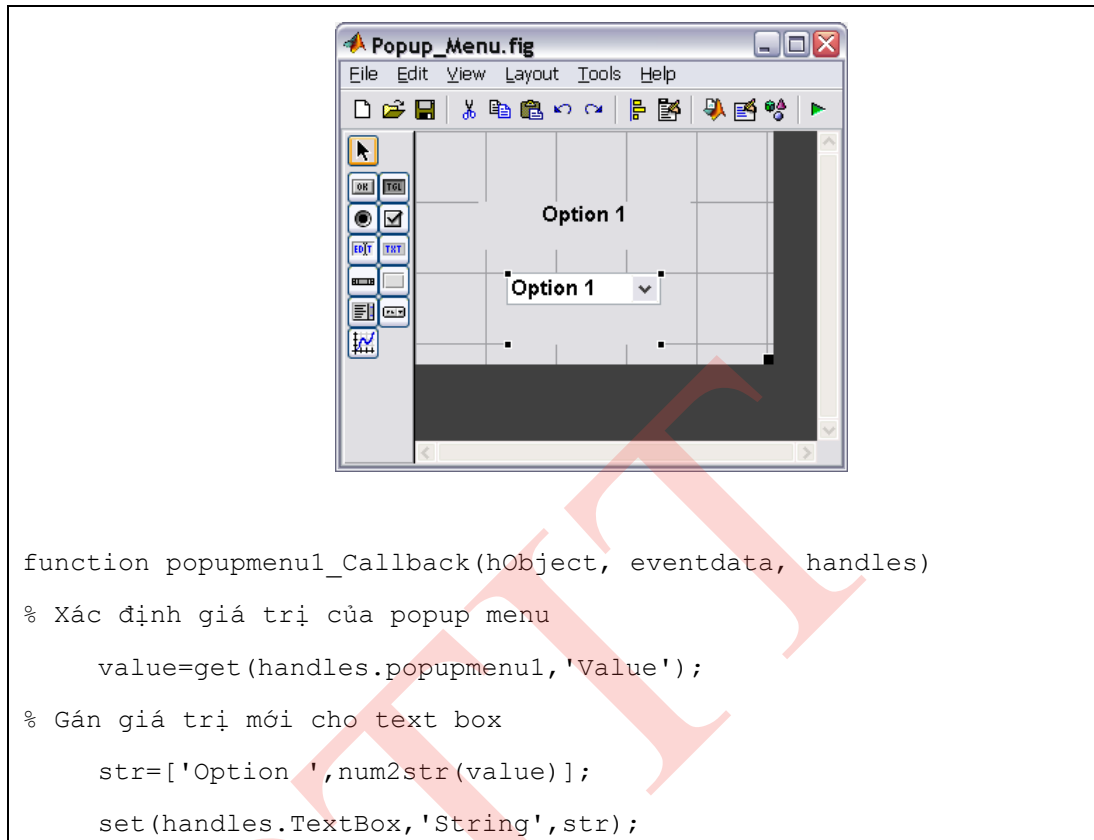
**Hình 3.25. Ví dụ về Radio Button và nội dung của chương trình con được gọi**



**Hình 3.26. Giao diện của chương trình Radio\_Button**

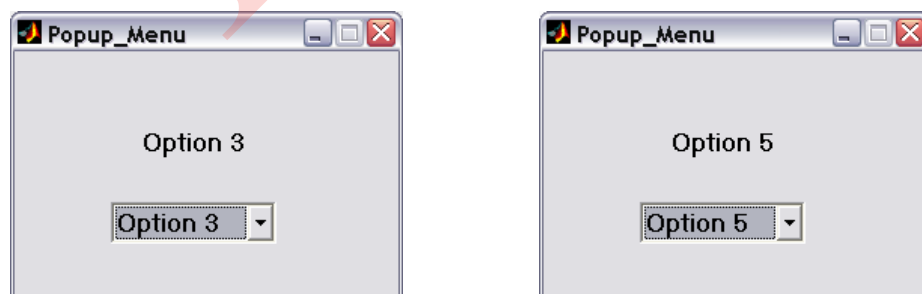
### g) Popup Menus

Popup Menu cho phép người sử dụng chọn một giá trị trong số các lựa chọn. *Danh sách các lựa chọn được xác định bởi một mảng của các chuỗi.* Giá trị của thuộc tính 'Value' thể hiện kết quả lựa chọn hiện tại. Một Popup Menu có thể được tạo ra bởi công cụ **popup menu** trong Layout Editor.



Hình 3.27. Ví dụ về Popup Menu và nội dung của chương trình con được gọi

Hình 3.27 thể hiện một ví dụ của popup menu và nội dung của hàm được gọi tương ứng. Trong ví dụ này, một popup menu được tạo ra với năm trường hợp được gắn nhãn 'Option 1', 'Option 2', ...., hàm này sẽ xác định giá trị được chọn trong đối số 'Value' và hiển thị giá trị được chọn. Hình 3.28 là trường hợp Option 3 và Option 5 được chọn.



Hình 3.28. Giao diện của chương trình Popup\_Menu

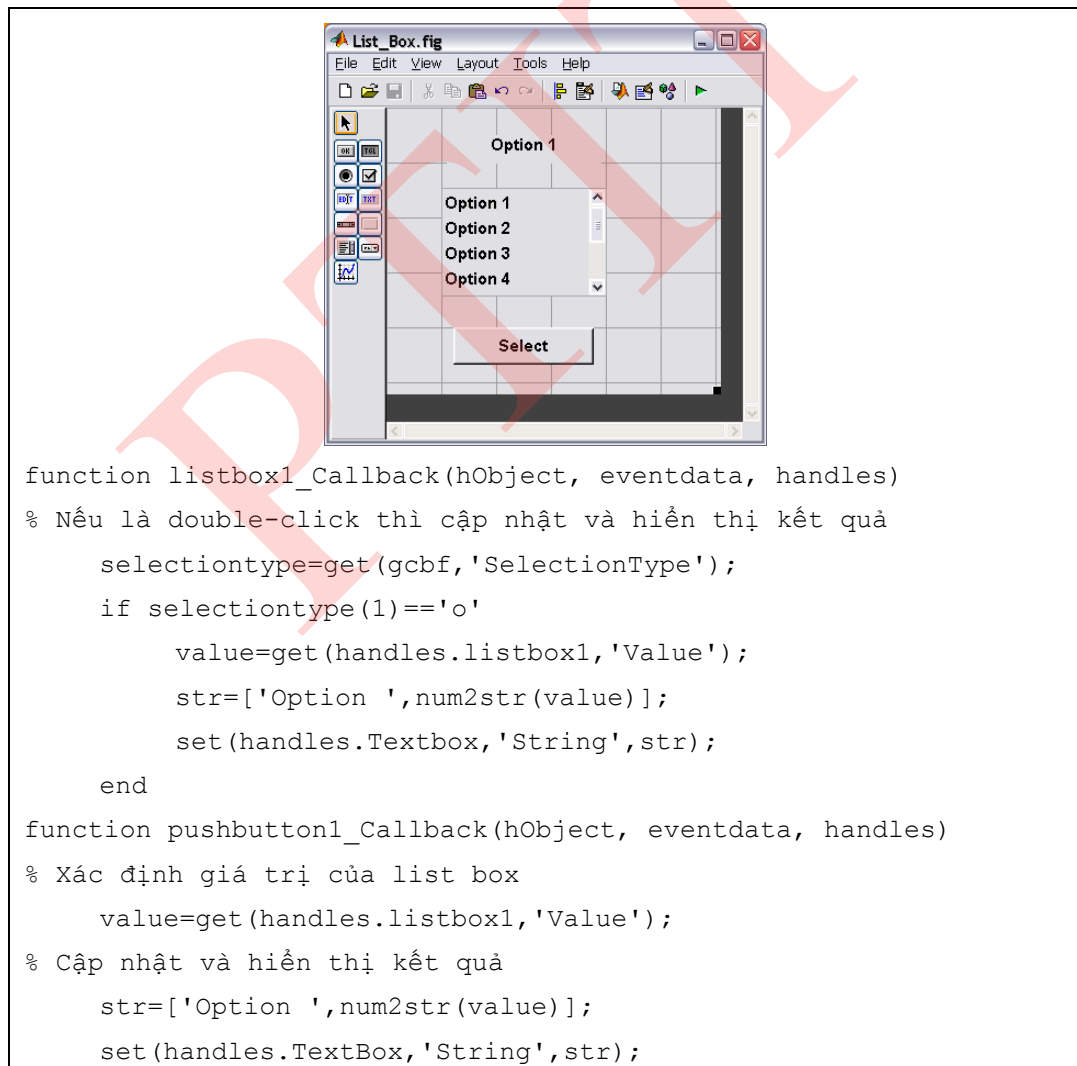
### h) List Boxes

List Box tạo ra một đối tượng có nhiều dòng ký tự và cho phép người sử dụng chọn một hay nhiều dòng trong số các dòng này. Nếu số dòng ký tự của đối tượng lớn hơn phạm vi của List Box thì một thanh cuộn sẽ được tạo ra cho phép người sử dụng di chuyển lên xuống trong

phạm vi của List Box. Những dòng ký tự mà người sử dụng có thể lựa chọn được xác định bởi một mảng các chuỗi và thuộc tính 'Value' sẽ xác định dòng nào được chọn. Một List Box được tạo ra bởi hàm **uicontrol** với thuộc tính **style** là 'listbox' hoặc được tạo ra từ công cụ listbox trong Layout Editor.

Trong trường hợp sử dụng một single-click để chọn một thành phần trong List Box thì chương trình sẽ không gây ra bất kỳ phản ứng nào cho đến khi có một tác động bên ngoài xảy ra, chẳng hạn như một nhấn nút. Tuy nhiên trong trường hợp sử dụng double-click thì có thể gây ra một tác động ngay lập tức. Có thể sử dụng thuộc tính SelectionType của figure để phân biệt single-click hay double-click. Giá trị của thuộc tính SelectionType sẽ là 'normal' tương ứng với single-click và trong trường hợp double-click thì giá trị này sẽ là 'open'. Điều này cũng đúng trong trường hợp chọn nhiều thành phần cùng một lúc, là trường hợp khoảng cách giữa giá trị của hai thuộc tính min và max lớn hơn 1. Còn trong các trường hợp khác thì chỉ có duy nhất một thành phần được chọn.

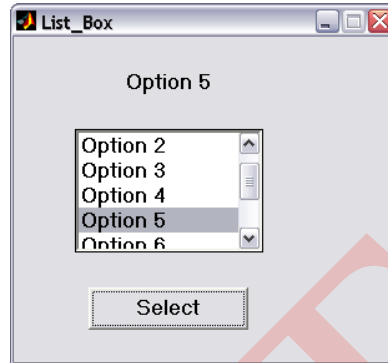
Hình 3.29 thể hiện một ví dụ bao gồm một list box có tám thành phần 'Option 1', 'Option 2', ..., 'Option 8', một nút nhấn để thực thi chương trình và một text field để hiển thị kết quả. Các hàm được gọi tương ứng với nút nhấn và list box như ở dưới hình.



**Hình 3.29. Ví dụ về List Box và nội dung của chương trình con được gọi**

- Nếu nút nhấn được chọn thì hàm *pushbutton1\_callback* sẽ xác định giá trị của List Box và thể hiện kết quả thông qua Text Field.
- Nếu list box được chọn, đầu tiên hàm *listbox1\_callback* sẽ xác định trạng thái click chuột là single hay double. Nếu là single-click thì chương trình sẽ chương trình không làm gì cả, còn trong trường hợp là double-click thì hàm *listbox1\_callback* sẽ xác định giá trị của List Box và thể hiện kết quả thông qua Text Field.

Giao diện của ví dụ này được thể hiện trong Hình 3.30.

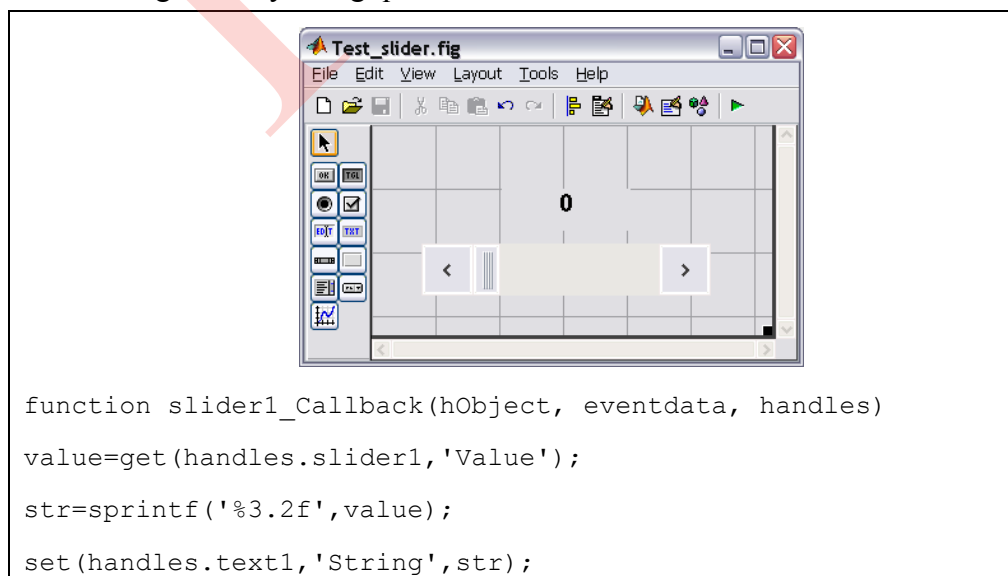


Hình 3.30. Giao diện của chương trình List\_Box

#### i) Sliders

Slider là một đối tượng cho phép người sử dụng lựa chọn giá trị trong một khoảng xác định bằng cách dùng chuột di chuyển thanh trượt. Thuộc tính 'Value' sẽ xác định giá trị của Slider trong khoảng [min max]. Đối tượng này có thể được tạo ra bởi hàm **uicontrol** với thuộc tính **style** là 'slider' hoặc được tạo ra bằng công cụ slider trong Layout Editor.

Hình 3.32 thể hiện một ví dụ đơn giản chỉ chứa một slider và một text field. Thiết lập giá trị của thuộc tính min là zero và giá trị của thuộc tính max là mười. Khi kéo thanh trượt, chương trình sẽ gọi hàm *slider1\_callback*, hàm này sẽ xác định giá trị của slider từ thuộc tính 'value' và thể hiện giá trị này thông qua text field.



Hình 3.31. Ví dụ về Slider và nội dung của chương trình con được gọi





Hình 3.32. Giao diện của chương trình Test\_slider

### 3.3.5. Tạo menu


Các hàm:

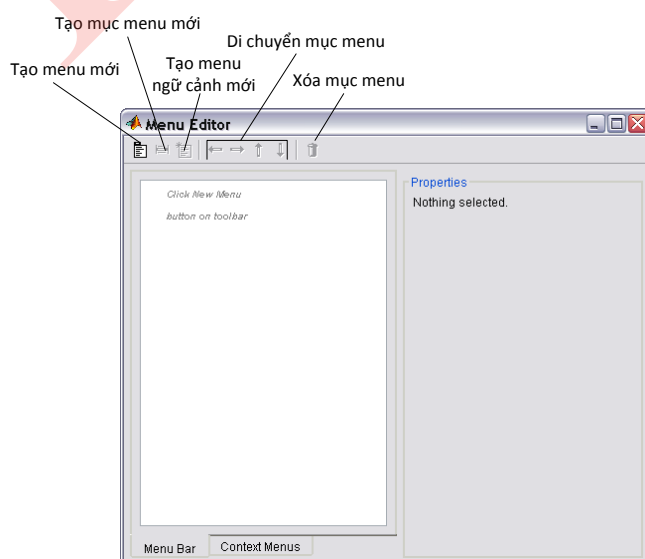
uimenu	Tạo Menu và các mục trong menu trên cửa sổ giao diện
uicontextmenu	Tạo menu ngữ cảnh
uitoolbar	Tạo Toolbar trên cửa sổ
uipushtool	Tạo nút ấn trên toolbar
uitoggletool	Tạo nút bật/tắt (toggle button) trên toolbar

Thuộc tính

Uimenu Properties	Mô tả thuộc tính của menu
Uicontextmenu Properties	Mô tả thuộc tính menu ngữ cảnh
Uitoolbar Properties	Mô tả thuộc tính của toolbar
Uipushtool Properties	Mô tả thuộc tính của nút ấn (push)
Uitoggletool Properties	Mô tả thuộc tính nút bật/tắt

#### 3.3.5.1. Tạo menu cho giao diện

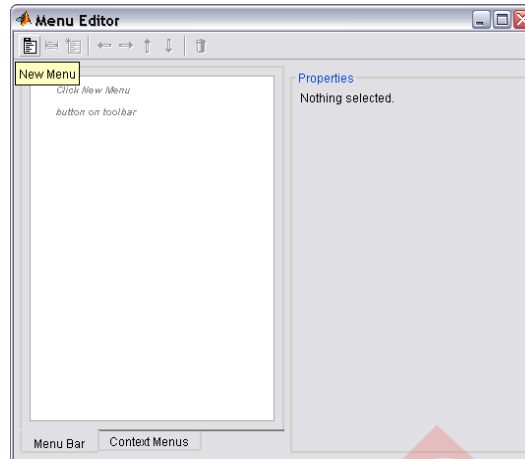
Ta có thể dùng GUIDE để tạo menu cho GUI với các menu dạng kéo xuống (pull-down) hoặc menu ngữ cảnh gắn với một đối tượng. Ta có thể tạo cả 2 loại menu bằng cách dùng Menu Editor. Gọi Menu Editor từ mục **Tools** hoặc click vào **Menu Editor** button .



Hình 3.33. Hội thoại tạo menu cho giao diện

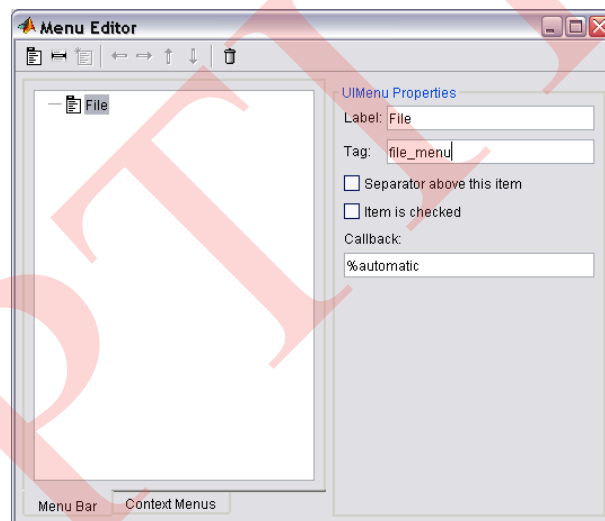
### a) Tạo một Menu

- Click vào nút New Menu trên thanh công cụ. Một menu có tên là, Untitled 1, xuất hiện bên trái của hộp hội thoại.



Hình 3.34. Tạo menu mới

- Click vào tên của menu title để hiển thị các lựa chọn thuộc tính của menu bên ô bên phải màn hình.



Hình 3.35. Thuộc tính của menu

- Điền vào các trường **Label** và **Tag** cho menu. Ví dụ, đặt **Label** là File và **Tag** là file\_menu. Click ra ngoài để các thay đổi có hiệu lực..

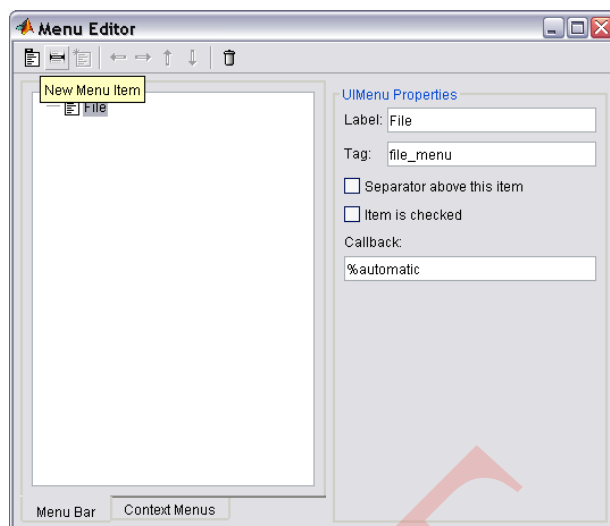
**Label** là chuỗi ký tự làm nhãn cho menu.

**Tag** chuỗi ký tự làm định dạng cho đối tượng menu.

### b) Thêm các mục cho một menu

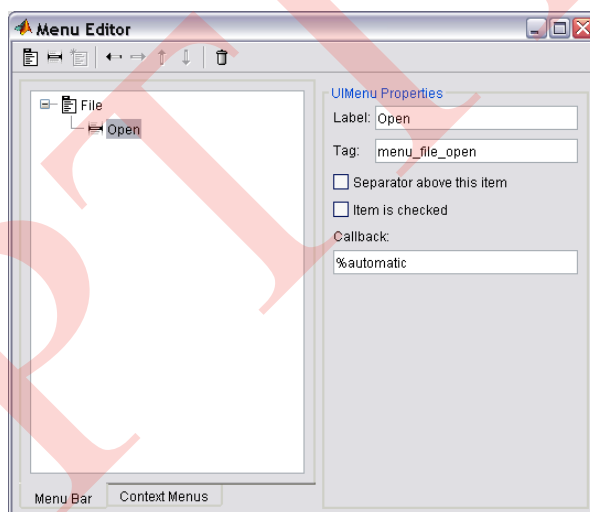
Sử dụng nút **New Menu Item** để tạo các mục cho menu, đây là nội dung sẽ hiển thị khi menu trải xuống

- Thêm một mục menu **Open** dưới menu File, bằng cách chọn File sau đó click vào nút **New Menu Item** trên thanh công cụ. Một mục menu tạm thời được gán nhãn là, Untitled, sẽ xuất hiện.



Hình 3.36. Tạo các mục cho menu

- Điền các trường **Label** và **Tag**. Ví dụ, đặt **Label** là Open và **Tag** là menu\_file\_open. Click ra ngoài để các thay đổi có hiệu lực.



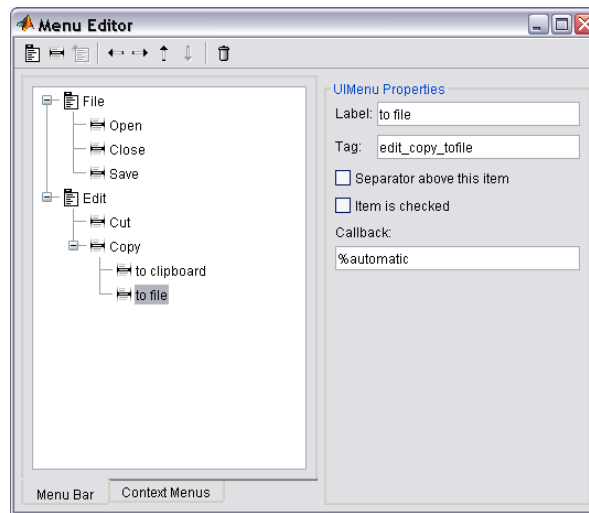
Hình 3.37. Điền tên và tag cho các mục của menu

### c) Additional Drop-Down Menus

Để tạo thêm các mục menu, sử dụng nút **New Menu** giống như cách tạo menu File. Ví dụ, hình dưới mô tả menu thả Edit.

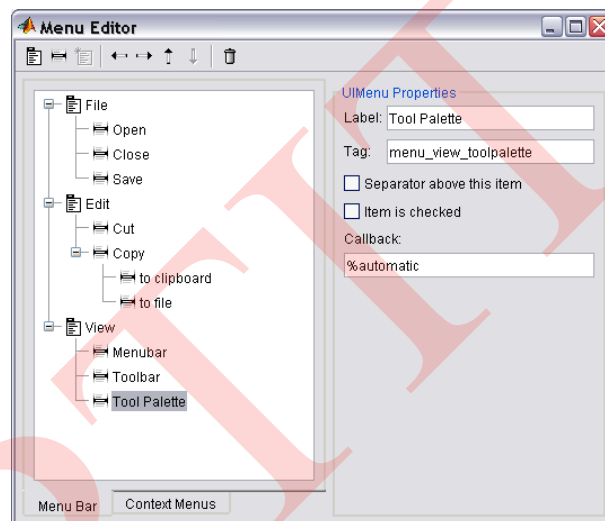
### d) Menu dây chuyền

Để tạo menu dây chuyền, chọn mục menu sẽ thực hiện menu dây chuyền, sau đó click vào nút **New Menu Item**. Như ví dụ dưới đây mục, Copy là menu dây chuyền.



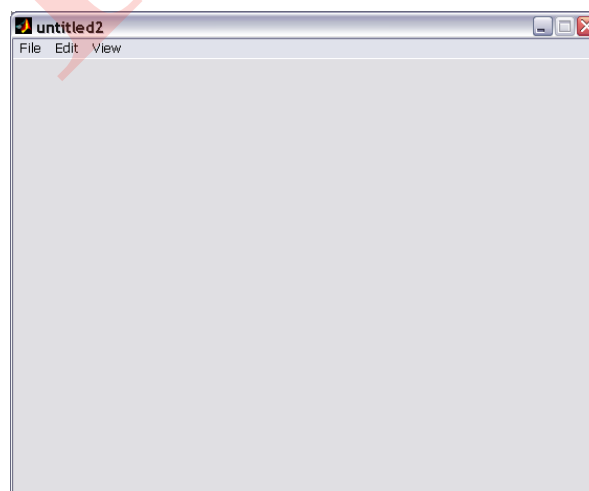
**Hình 3.38. Tạo menu dây chuyền**

Dưới đây là ví dụ gồm 3 mục menu được thiết lập cho menu Menu bar.



**Hình 3.39. Một ví dụ về menu gồm 3 mục**

Khi ta chạy GUI, tên của menu xuất hiện trên thanh menu.



**Hình 3.40. Giao diện menu khi thực thi**

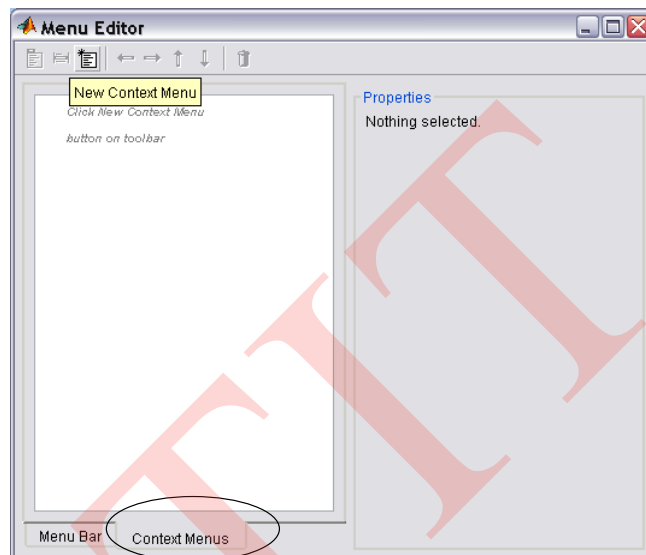
### 3.3.5.2. Menu ngữ cảnh

Một menu ngữ cảnh sẽ xuất hiện khi người dùng kích chuột phải vào một đối tượng được định nghĩa trước. Menu Editor cho phép ta định nghĩa menu ngữ cảnh và liên kết chúng với các đối tượng. Thủ tục gồm 3 bước:

#### a) Tạo menu mẹ

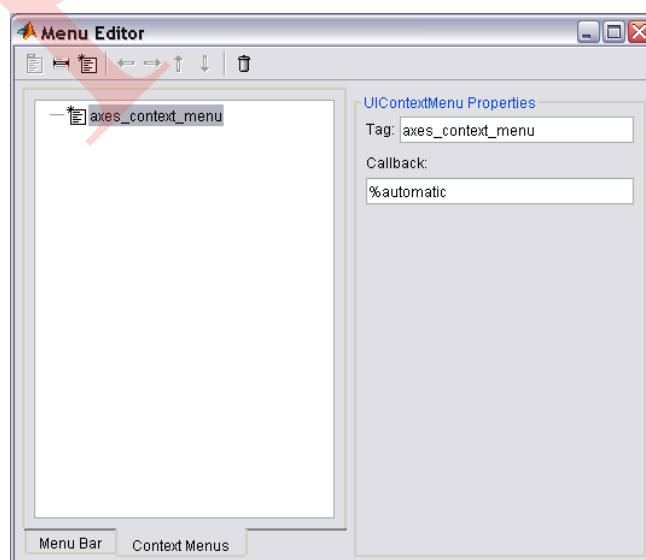
Tất cả các mục trong menu ngữ cảnh đều là con của một menu mà không hiển thị trên thanh menu. Để định nghĩa menu mẹ ta làm như sau:

- Chọn Menu Editor → tab **Context Menus** và chọn nút **New Context Menu** trên thanh công cụ.



Hình 3.41. Tạo mới menu ngữ cảnh

- Chọn menu, và trong trường **Tag** nhập vào tên tag của menu ngữ cảnh, như trong ví dụ này là `axes_context_menu`.

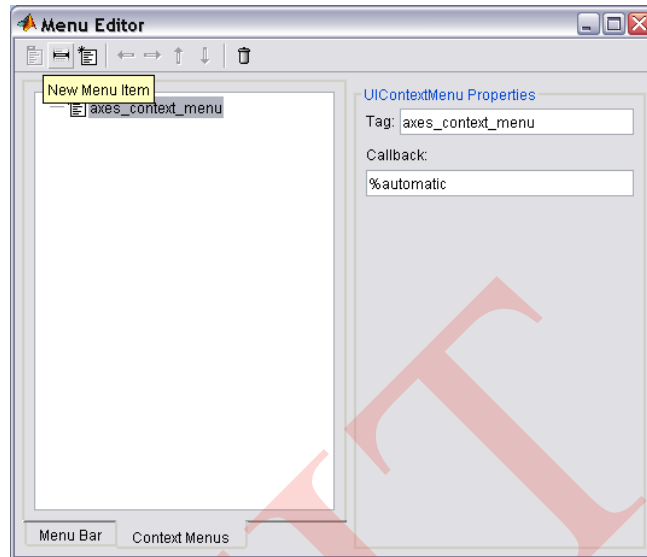


Hình 3.42. Nhập tên và tag của menu ngữ cảnh

### ***b) Thêm các mục vào menu ngữ cảnh***

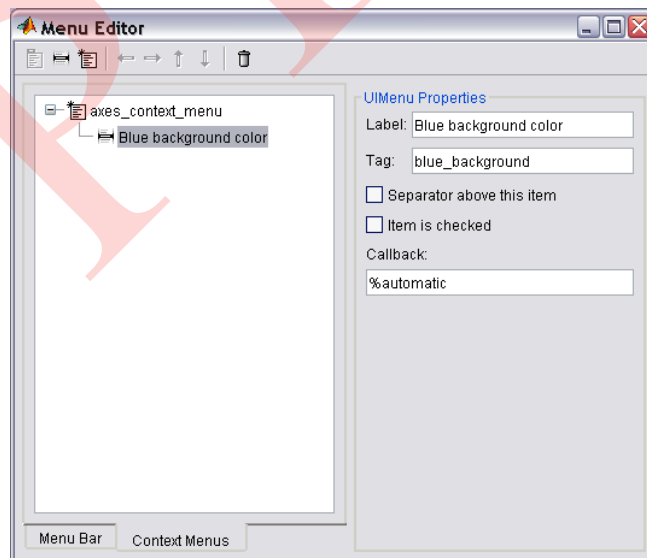
Sử dụng nút **New Menu Item** để tạo các mục cho menu sẽ xuất hiện trong menu ngữ cảnh.

- Thêm mục **Blue background color** cho menu bằng cách chọn `axes_context_menu` và click vào **New Menu Item**. Một menu tạm thời được gán nhãn, `Untitled`, xuất hiện.



**Hình 3.43. Thêm mục cho menu ngữ cảnh**

- Điền vào các trường **Label** và **Tag** cho mục menu mới. Ví dụ, đặt **Label** thành **Blue background color** và đặt **Tag** thành **blue\_background**. Kích ra ngoài màn hình để có hiệu lực.



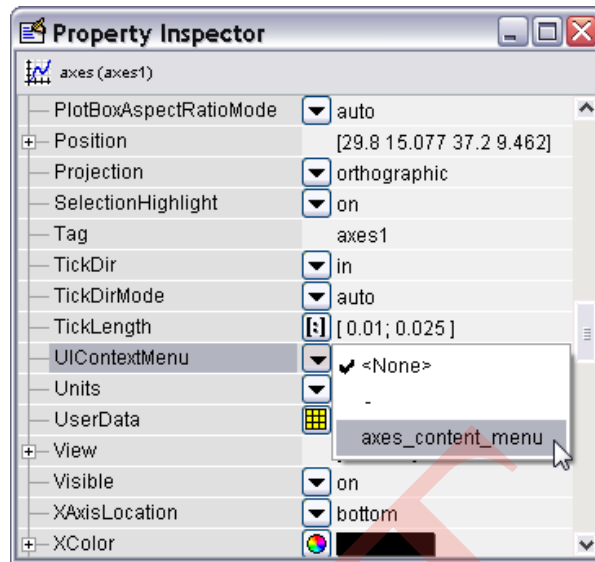
**Hình 3.44. Thay đổi các trường của menu ngữ cảnh**

### ***c) Liên kết menu ngữ cảnh với một đối tượng***

- Trong cửa sổ Editor, chọn một đối tượng để định nghĩa menu ngữ cảnh.

- Sử dụng Property Inspector để đặt thuộc tính `UIContextMenu` của đối tượng này thành tên của menu ngữ cảnh mong muốn.

Hình sau mô tả thuộc tính `UIContextMenu` cho đối tượng `axes` với Tag là `axes1`.



Hình 3.45. Liên kết menu ngữ cảnh với một đối tượng

Trong code nguồn GUI, hoàn thiện các hàm **callback** cho các mỗi mục của menu ngữ cảnh. Mỗi một callback sẽ được chạy khi người dùng chọn một mục của menu ngữ cảnh.

### 3.3.5.3. Danh sách một số hàm

<code>axes</code>	Tạo một hệ trục tọa độ
<code>figure</code>	Tạo một figure (cửa sổ giao diện) mới
<code>get</code>	Truy xuất giá trị của các thuộc tính của một đối tượng giao diện
<code>guide</code>	Mở chương trình thiết kế giao diện GUI
<code>set</code>	Thay đổi giá trị của các thuộc tính của một đối tượng giao diện
<code>uicontextmenu</code>	Tạo một đối tượng context menu
<code>uicontrol</code>	Tạo một đối tượng điều khiển trong giao diện với người sử dụng
<code>uimenu</code>	Tạo một đối tượng menu

## CHƯƠNG 4. ỨNG DỤNG CỦA MATLAB

### 4.1. GIỚI THIỆU

MATLAB được coi như một công cụ hỗ trợ đắc lực cho bài giảng của rất nhiều môn trong nhiều lĩnh vực. Khi giới thiệu MATLAB và yêu cầu thực hiện bài tập tính toán với sự hỗ trợ của MATLAB, hầu hết các sinh viên khối ngành kỹ thuật nói chung, sinh viên ngành Kỹ thuật Điện tử - Viễn thông nói riêng đều đặt một câu hỏi: Tại sao sử dụng MATLAB? Có rất nhiều câu trả lời cho câu hỏi này.

Trước hết, MATLAB được sử dụng một cách rộng rãi trong nhiều chương trình đào tạo khối ngành kỹ thuật bởi vì MATLAB là một ngôn ngữ lập trình đa dụng mức cao. Việc triển khai một thuật toán với sự hỗ trợ của MATLAB rất nhanh chóng và thuận tiện. Với sự hỗ trợ của MATLAB, công việc kiểm nghiệm các kiến thức trong học tập và các kết quả nghiên cứu không những nhanh mà còn hiệu quả.

Mặt khác, ngôn ngữ lập trình MATLAB về cơ bản gần với những ngôn ngữ lập trình phổ biến khác nên việc tiếp cận MATLAB không mấy khó khăn. Hơn thế nữa, ngôn ngữ lập trình MATLAB cùng với các bộ công cụ được phát triển bởi Mathworks cũng như của các hãng khác cung cấp cho người sử dụng một loạt các hàm thư viện phong phú với những thuật toán cơ bản được thực hiện một cách tối ưu.

Với sự hỗ trợ đó, việc thực hiện giải các bài toán phức tạp trong MATLAB đôi khi chỉ cần một vài dòng lệnh.

Trong phần này, chúng ta sẽ làm quen với việc sử dụng MATLAB thực hiện giải một số bài toán trong một số lĩnh vực phổ biến. Do sự phong phú của các hàm thư viện cũng như các bộ công cụ của MATLAB, cũng như hạn chế về thời gian và không gian trình bày của bài giảng này, chúng ta sẽ không thực hiện liệt kê đầy đủ mà chỉ thông qua một số ví dụ cụ thể. Để biết đầy đủ và chi tiết, người học có thể tham khảo các tài liệu hướng dẫn của Mathworks [4]-[11]

### 4.2. ỨNG DỤNG MATLAB TRONG TOÁN HỌC

#### 4.2.1. Giới thiệu chung

Vốn dĩ là một công cụ hỗ trợ tính toán, đặc biệt là tính toán với ma trận nên không có gì là quá khi cho rằng MATLAB là công cụ hỗ trợ đắc lực cho việc giải quyết các bài toán toán học. Trong phần này chúng ta sẽ xem xét việc áp dụng các hàm thư viện của MATLAB để thực hiện việc giải các bài toán toán học cơ bản.

Các kiến thức cơ bản về Đại số tuyến tính và Giải tích, người học có thể tham khảo thêm trong các tài liệu bài giảng [12]-[17].

#### 4.2.2. Một số bài toán toán học giải bằng MATLAB

##### 4.2.2.1. Giải bài toán Đại số

Để làm việc với đa thức, MATLAB cung cấp một số hàm thư viện cho phép định nghĩa, biểu diễn, tìm nghiệm, tính giá trị đa thức.



Hàm	Ý nghĩa	Ví dụ
poly()	Xây dựng đa thức từ nghiệm, từ hệ số, hoặc đa thức đặc trưng của ma trận	poly([1 2 3])
roots()	Tìm nghiệm của đa thức	roots([1 2 1])
polyval()	Tính giá trị của một đa thức	polyval(p)
polyder()	Tính đạo hàm của đa thức	polyder(p)
poly2str()	Biểu diễn đa thức thành chuỗi để hiển thị	poly2str(p,'x')
conv()	Thực hiện phép nhân (chập) hai đa thức	conv(p1,p2)
deconv()	Thực hiện chia hai đa thức, cho kết quả phần nguyên và phần dư	[d,r]=deconv(p1,p2)

*Xem thêm tài liệu hướng dẫn sử dụng MATLAB*

**Ví dụ 4-1:** Cho đa thức  $f(x) = x^4 - 10x^3 + 35x^2 - 50x + 24$ . Tìm nghiệm của đa thức.

Khi thực hiện giải bài toán tìm nghiệm đa thức, chúng ta biết rằng với các đa thức bậc  $\geq 3$  thì không tồn tại công thức giải tích tường minh để tính nghiệm. Với bài toán trong ví dụ này đầu tiên chúng ta sẽ thực hiện nhẩm nghiệm, sau đó sử dụng kết quả nghiệm nhẩm được để giảm bậc đa thức để đưa về các đa thức bậc thấp hơn và có thể thực hiện giải theo công thức đã biết. Tuy nhiên, dễ thấy rằng việc nhẩm nghiệm không hề đơn giản nếu nghiệm đó là các giá trị không nguyên hoặc có giá trị không đủ nhỏ cho phép nhẩm dễ dàng.

Việc thực hiện giải bài toán trên sẽ trở nên đơn giản nếu chúng ta sử dụng sự hỗ trợ của MATLAB.

```
% Liệt kê các hệ số của đa thức
p=[1 -10 35 -50 24];
% Tính các nghiệm của đa thức bằng hàm roots()
root_p=roots(p)
```

**Chú ý:** Việc giải phương trình cũng có thể thực hiện bằng hàm thư viện *solve()*

```
>>syms x;
>>solve(x^4-10*x^3+35*x^2-50*x+24==0)
```

**Ví dụ 4-2:** Xây dựng một đa thức biết đa thức có các nghiệm là 1, 2, 3, và 4.

Về mặt lý thuyết, với các nghiệm đã biết thì đa thức có thể dễ dàng viết bởi công thức  $f(x) = (x - 1)(x - 2)(x - 3)(x - 4)$ . Sau đó thực hiện việc nhân các nhân tử với nhau chúng ta sẽ có kết quả.

Việc này có thể được thực hiện với sự hỗ trợ của MATLAB như sau:

```
% Liệt kê các nghiệm của đa thức
myroot=[1 2 3 4];
```

```
% Lập đa thức bằng hàm poly()
p=poly(myroot);
% Hiển thị đa thức
poly2str(p, 'x');
```

**Ví dụ 4-3:** Cho đa thức  $f(x) = x^6 - 3x^4 + 5x^3 - 4x^2 + 3x + 2$ . Tính giá trị của hàm số tại  $x_0 = -3$

Chúng ta có thể thực hiện bài toán trên bằng MATLAB như sau:

```
% Liệt kê các hệ số của đa thức
P=[1 0 -3 5 -4 3 2];
% Tính giá trị bằng hàm polyval()
Val=polyval(p,-3)
```

Ngoài việc sử dụng các hàm thư viện nói trên, khi cài đặt bộ công cụ Toán biểu thức (Symbolic Math Toolbox), chúng ta có thể sử dụng hàm *solve()* để thực hiện giải phương trình hoặc hệ phương trình.

**Ví dụ 4-4:** Tìm nghiệm của phương trình  $x^2 + 4x + 1 = 0$

```
>>syms x;
>>solve(x^2+4*x+1==0)
```

#### 4.2.2.2. Giải bài toán Đại số tuyến tính

MATLAB cung cấp rất nhiều hàm thư viện hỗ trợ việc tính toán và giải các bài toán Đại số tuyến tính. Bằng cách sử dụng các hàm thư viện cơ bản này, lời giải (kết quả) các bài toán Đại số tuyến tính có thể dễ dàng được kiểm chứng. Các hàm cơ bản được liệt kê trong bảng sau:

Hàm	Ý nghĩa	Ví dụ
det()	Tính định thức của ma trận	det(A)
inv()	Tìm ma trận nghịch đảo	inv(A)
diag()	Xây dựng ma trận đường chéo; Trích xuất đường chéo của ma trận	diag(A)
trace()	Tìm vết của ma trận	trace(A)
rank()	Tìm hạng của ma trận	rank(A)
rref()	Tìm dạng thu gọn hàng Echelon	rref(A)
tril()	Tìm ma trận tam giác dưới	tril(A)
triu()	Tìm ma trận tam giác trên	triu(A)
eig()	Tìm các trị riêng và các véc-tơ riêng	eig(A)

*Xem thêm tài liệu hướng dẫn sử dụng MATLAB*

**Ví dụ 4-5:** Thực hiện giải hệ phương trình tuyến tính sau:

$$\begin{cases} x + 2y + 3z = 2 \\ x + y - z = 4 \\ x + 2y + z = 4 \end{cases}$$

Về mặt toán học, hệ phương trình tuyến tính trên có thể được giải theo nhiều cách. Để thực hiện giải bằng MATLAB chúng ta biểu diễn lại hệ phương trình trên như sau:

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & -1 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 4 \end{bmatrix} \Leftrightarrow \mathbf{A}\mathbf{v}=\mathbf{b}$$

Theo đại số tuyến tính, các nghiệm của hệ, tức là véc tơ  $\mathbf{v}$ , có thể tìm bằng cách tính  $\mathbf{v} = \mathbf{A} \backslash \mathbf{b}$  hoặc  $\mathbf{v} = \mathbf{A}^{-1}\mathbf{b}$ . Thực hiện trong MATLAB như sau:

```
A=[1 2 3; 1 1 -1; 1 2 1];
b=[2; 4; 4];
v=A\b
% Hoặc
v=inv(A)*b;
```

**Ví dụ 4-6:** Giải hệ phương trình tuyến tính bằng hệ thức Cramer

Xét lại ví dụ 4-6 ở trên, chúng ta biết rằng hệ phương trình đó còn có thể thực hiện giải bằng hệ thức Cramer. Trong đó:

$x = \frac{D_x}{D}$ ,  $y = \frac{D_y}{D}$ , và  $z = \frac{D_z}{D}$  với các giá trị  $D, D_x, D_y, D_z$  được tính như sau:

$$D = \begin{vmatrix} 1 & 2 & 3 \\ 1 & 1 & -1 \\ 1 & 2 & 1 \end{vmatrix}, D_x = \begin{vmatrix} 2 & 2 & 3 \\ 4 & 1 & -1 \\ 4 & 2 & 1 \end{vmatrix}, D_y = \begin{vmatrix} 1 & 2 & 3 \\ 1 & 4 & -1 \\ 1 & 4 & 1 \end{vmatrix}, \text{ và } D_z = \begin{vmatrix} 1 & 2 & 2 \\ 1 & 1 & 4 \\ 1 & 2 & 4 \end{vmatrix}$$

Phương pháp này có thể thực hiện giải trên MATLAB như sau:

```
A=[1 2 3;1 1 -1; 1 2 1];
b=[2; 4; 4];
% Lap cac ma tran cho cac dinh thuc
Ax=A; Ax(:,1)=b;
Ay=A; Ay(:,2)=b;
Az=A; Az(:,3)=b;
% Tinh cac dinh thuc
D=det(A);
Dx=det(Ax);
Dy=det(Ay);
Dz=det(Az);
% Tinh cac nghiệm
x=Dx/D;
y=Dy/D;
z=Dz/D;
```

Việc giải các hệ phương trình tuyến tính một cách trực tiếp phải sử dụng các phép toán ma trận (như định thức, ma trận nghịch đảo, ...). Đây là những phép toán có độ phức tạp tính toán lớn, đặc biệt khi kích thước của các ma trận đủ lớn. Để giảm nhỏ khối lượng tính toán chúng ta có thể áp dụng các phương pháp giải sử dụng các phép khử Gauss, Jordan, hoặc dạng thức thu gọn hàng Echelon (Reduced Row Echelon Form).

**Ví dụ 4-7:** Giải hệ phương trình tuyến tính theo phương pháp khử Gauss, Jordan, Reduced Row Echelon Form

Xét hệ phương trình tuyến tính cho trong ví dụ 4-6. Chúng ta sẽ sử dụng dạng thức thu gọn hàng Echelon để giải như sau:

```
A=[1 2 3;1 1 -1; 1 2 1];
b=[2; 4; 4];
% Lập ma trận mở rộng
Aext=[A b];
[B, pivot]=rref(Aext);
% Nếu B là ma trận đường chéo
% Nghiệm hệ sẽ là cột cuối cùng
v=Aext(:,4);
```

Không chỉ có khả năng hỗ trợ mạnh việc giải các hệ phương trình tuyến tính, MATLAB còn cho phép thực hiện phân tích các ma trận một cách dễ dàng. Trong ví dụ tiếp sau, chúng ta xem xét việc tính các trị riêng và vectơ riêng trong MATLAB.

**Ví dụ 4-8:** Cho ma trận  $A = \begin{bmatrix} 5 & 7 & -5 \\ 0 & 4 & -1 \\ 2 & 8 & -3 \end{bmatrix}$ . Tìm các trị riêng và các vectơ riêng của ma trận A

Các giá trị riêng của một ma trận được định nghĩa là các giá trị  $\lambda$  thỏa mãn phương trình đặc trưng  $\det(A - I\lambda) = 0$ . Như vậy, để tìm các giá trị riêng, chúng ta cần giải phương trình:

$$\begin{vmatrix} 5-\lambda & 7 & -5 \\ 0 & 4-\lambda & -1 \\ 2 & 8 & -3-\lambda \end{vmatrix} = 0 \Leftrightarrow \lambda^3 - 6\lambda^2 + 11\lambda - 6 = 0$$

Dễ dàng thấy phương trình đặc trưng có các nghiệm là  $\lambda_1 = 1$ ,  $\lambda_2 = 2$ ,  $\lambda = 3$ . Đây chính là các trị riêng của ma trận. Công việc trên có thể thực hiện dễ dàng với sự hỗ trợ của MATLAB như sau:

```
A=[5 7 -5; 0 4 -1; 2 8 -3];
% Phương trình đặc trưng của ma trận
P=poly(A);
% Tìm nghiệm của phương trình đặc trưng
lambda=roots(p);
```

Với giá trị riêng  $\lambda$ , vectơ trị riêng của ma trận là vectơ khác không  $\mathbf{x}$  thỏa mãn  $A\mathbf{x} = \lambda\mathbf{x}$ . Bằng cách thay lần lượt các giá trị trị riêng vào phương trình trên chúng ta sẽ tìm được các vectơ trị riêng.

Chúng ta có thể thực hiện việc tìm đồng thời các giá trị riêng và vectơ trị riêng đơn giản hơn trong MATLAB bằng cách sử dụng hàm *eig()* như sau:

```
A=[5 7 -5; 0 4 -1; 2 8 -3];
% Phương trình đặc trưng của ma trận
[x,lambda]=eig(A);
```

#### 4.2.2.3. Giải bài toán Giải tích

##### a) Tính tích phân

Tính tích phân là một phần quan trọng trong khối kiến thức giải tích. MATLAB cung cấp một số hàm thư viện cho phép tính toán không chỉ các giá trị các tích phân (tích phân xác định) mà còn cho phép tính toán công thức giải tích (tích phân không xác định).

Hàm	Ý nghĩa	Ví dụ
quad()	Tính giá trị tích phân đơn (tích phân xác định một lớp)	quad(f,0,1)
dblquad()	Tính giá trị tích phân kép (tích phân xác định hai lớp)	dblquad(f,0,2,0,3)
integral()	Tính giá trị tích phân đơn (tích phân xác định một lớp)	integral(f,0,Inf)
integral2()	Tính giá trị tích phân kép (tích phân xác định hai lớp)	integral(f,0,2,1,4)
quad2d()	Tính giá trị tích phân kép	quad2d(f,0,2,2,3)
int()	Tính tích phân (xác định, không xác định)	int(f,0,1) int(f)

Xem thêm tài liệu hướng dẫn sử dụng MATLAB

**Ví dụ 4-9:** Tính giá trị của tích phân  $I = \int_0^2 \frac{1}{x^3-2x-5} dx$

Việc tính toán giải tích giá trị tích phân trên không quá khó và được xem như bài tập luyện tập cho người học. Chúng ta có thể kiểm tra kết quả với sự hỗ trợ của MATLAB như sau:

```
>>f=inline('1./(x.^3-2*x-5)');
>>q=quad(f,0,2);
```

**Chú ý:** Hàm *quad()* sẽ được thay thế bởi hàm *integral()* trong các phiên bản MATLAB mới

**Ví dụ 4-10:** Thực hiện việc tính giá trị tích phân  $I = \int_1^2 \int_0^3 x^2 y dx dy$

Việc tính toán giá trị của tích phân bằng tay coi như một bài tập luyện tập cho người học. Ở đây chúng ta sẽ quan tâm đến việc sử dụng MATLAB để tính giá trị này.

```
f=inline('x.^2.*y','x','y');
```

```
q=dblquad(f,1,2,0,3);
```

**Chú ý:** Hàm *dblquad()* sẽ được thay thế bởi hàm *integral2()* trong các phiên bản MATLAB mới.

**Ví dụ 4-11:** Thực hiện tính giá trị tích phân  $I = \int_0^{\infty} e^{-x^2} \ln(x) dx$

Tương tự ví dụ 4-10, phần tính giá trị tích phân bằng tay coi như một bài tập luyện tập. Chúng ta chỉ minh họa thực hiện việc sử dụng hàm thư viện *integral()* để tính giá trị của tích phân này.

```
% Tạo một handle hàm cho biểu thức tính tích phân
f=@(x) exp(-x.^2).*log(x)
% Thực hiện tính tích phân
q=integral(f,0,Inf);
```

**Ví dụ 4-12:** Thực hiện tính giá trị tích phân  $I = \int_0^1 x \ln(1+x) dx$

Như đã trình bày ở trên, chúng ta có thể sử dụng hàm *quad()* hoặc hàm *integral()* để thực hiện việc tính toán giá trị các tích phân trên. Ngoài ra, chúng ta cũng có thể sử dụng hàm thư viện *int()* trong bộ công cụ Toán biểu thức (Symbolic Math Toolbox) để tính như sau:

```
% Khai báo biến biểu thức
syms x;
int(x*log(1+x),0,1);
```

Với hàm *int()* trong bộ công cụ Toán biểu thức (Symbolic Math Toolbox), chúng ta không những có thể tính toán giá trị các tích phân xác định mà còn có thể thực hiện việc tính toán giải tích các tích phân (tích phân không xác định).

**Ví dụ 4-13:** Thực hiện việc tính tích phân  $I = \int \frac{-2x}{(1+x^2)^2} dx$

Việc thực hiện tính toán tích phân trên hoàn toàn đơn giản coi như bài tập luyện tập cho người học. Kết quả tính toán có thể được kiểm nghiệm bằng MATLAB như sau:

```
% Khai báo biến biểu thức
syms x;
int(-2*x/(1+x^2)^2)
```

## ***b) Giải phương trình vi phân***

Một mảng kiến thức khác quan trọng cũng không kém trong giải tích và cũng là một mảng kiến thức khó đó là phương trình vi phân. MATLAB cung cấp khá nhiều hàm thư viện cho phép giải các phương trình vi phân từ bậc một đến các bậc cao hơn. Các phương trình vi phân thông thường (1 biến) hoặc từng phần (nhiều biến) đều có thể thực hiện giải bằng MATLAB. Hơn nữa, việc giải các phương trình vi phân không những có thể thực hiện bằng phương pháp số (numerical) mà còn có thể thực hiện giải tích (biểu thức toán học). Các hàm thư viện MATLAB hỗ trợ việc giải phương trình vi phân được liệt kê trong bảng:

Hàm	Ý nghĩa	Ví dụ
ode23()	Giải phương trình/hệ phương trình vi phân liên tục (non-stiff) bậc thấp	ode23(@f,[1 2],[0 4]);
ode45()	Giải phương trình/hệ phương trình vi phân liên tục (non-stiff) bậc trung bình	ode45(@f,[0 3],[2 8]);
ode113()	Giải phương trình/hệ phương trình vi phân liên tục (non-stiff) bậc thay đổi	ode113(@f,[0 2],[1 3]);
ode15s()	Giải phương trình/hệ phương trình vi phân không liên tục (stiff), bậc thay đổi	ode15s(@f,[0 2],[0 2]);
ode23s()	Giải phương trình/hệ phương trình vi phân không liên tục (stiff), bậc thấp	ode23s(@f,[0 1],[0 2])
ode23t()	Giải phương trình/hệ phương trình vi phân (medium stiff), theo phương pháp tứ giác	ode23t(@f,[0 1],[0 2])
dsolve()	Giải phương trình, hệ phương trình vi phân theo phương pháp tính toán số hoặc giải tích	dsolve('Dy=ay')

*Xem thêm tài liệu hướng dẫn sử dụng MATLAB*

**Ví dụ 4-14:** Thực hiện giải phương trình vi phân bậc nhất  $y' = ay$

Việc thực hiện giải tay bài này không khó và coi như một bài tập luyện tập đối với người học, ở đây chúng ta minh họa việc giải bài toán này trên MATLAB.

```
>>dsolve('Dy=a*y');
```

*Chú ý:* MATLAB sử dụng  $\text{diff}(y,n)$  cho đạo hàm bậc  $n$  của  $y$ , và thường được ký hiệu là  $Dy, D2y, \dots$

**Ví dụ 4-15:** Thực hiện giải phương trình vi phân bậc hai  $my'' + cy' + ky = 0$  với điều kiện đầu  $y(0) = 1$  và  $y'(0) = 0$ .

Bài toán trên có thể thực hiện giải trong MATLAB như sau:

```
>>dsolve('m*D2y+c*Dy+k*y=0','y(0)=1','Dy(0)=0');
```

**Ví dụ 4-16:** Thực hiện giải phương trình vi phân bậc hai  $y'' = -a^2y$  với điều kiện đầu  $y(0) = 1$  và  $y'(\frac{\pi}{a}) = 0$

Bài toán trên có thể thực hiện giải trong MATLAB như sau:

```
syms a y(t);
% Định nghĩa biên trung gian, y'
Dy=diff(y);
dsolve(diff(y,2)==-a^2*y,y(0)==1,Dy(pi/a)==0)
```

**Chú ý:** Mặc dù MATLAB là một công cụ mạnh hỗ trợ tính toán, nhưng có thể có một số trường hợp không tồn tại lời giải hoặc lời giải quá phức tạp mà MATLAB không thể tìm ra. Luôn ghi nhớ, MATLAB chỉ là công cụ hỗ trợ, MATLAB không thể làm thay chúng ta.

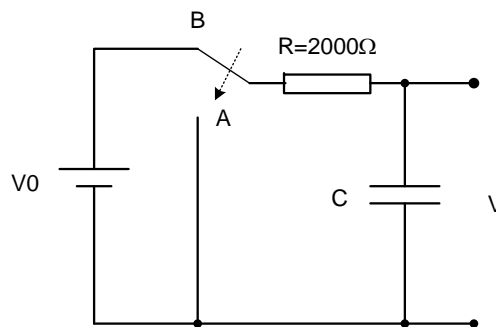
### c) Giải bài toán Phương pháp tính

Việc xấp xỉ đường, còn được gọi là phân tích hồi quy, là quá trình xây dựng một hàm giải tích phù hợp với một tập dữ liệu cho trước. Hàm xây dựng được sẽ được sử dụng như là một mô tả toán học đầy đủ của dữ liệu và sẽ được sử dụng cho các nghiên cứu tiếp theo của hệ thống. Quá trình xấp xỉ có thể dựa trên các dạng hàm từng biết như: tuyến tính, đa thức, mũ, lũy thừa, ... MATLAB cung cấp hàm thư viện *polyfit()* để thực hiện việc xấp xỉ đường.

**Ví dụ 4-17:** Trong quá trình thực nghiệm, đại lượng quan sát được là một dãy các cặp giá trị tọa độ sau (0.9,0.9), (1.5,1.5), (3,2.5), (4,5.1), (6,4.5), (8,4.9), và (9.5,6.3). Xây dựng hàm đa thức có bậc  $n = 1 \div 6$  xấp xỉ mô tả mô hình toán học cho dữ liệu quan sát được.

```
x=[0.9 1.5 3 4 6 8 9.5];
y=[0.9 1.5 2.5 5.1 4.5 4.9 6.3];
n=1; % Đa thức bậc 1
p=polyfit(x,y,n);
% Hiện thị đa thức xây dựng được
poly2str(p,'x');
% Vẽ minh họa việc xấp xỉ đường
xp=0.9:0.1:9.5;
yp=polyval(p,xp);
plot(x,y,'o',xp,yp);
xlabel('x'); ylabel('y');
```

**Ví dụ 4-18:** Trong quá trình thí nghiệm môn cơ sở cấu kiện điện tử, sinh viên được yêu cầu xác định giá trị của một tụ điện thông qua một thí nghiệm như sau. Sơ đồ mạch thí nghiệm như hình vẽ. Ban đầu chuyển mạch ở vị trí B, tụ được nạp đầy từ nguồn, sau đó chuyển mạch chuyển sang vị trí A đồng thời giá trị điện áp  $V$  được đo trong vòng 10s, mỗi lần đọc kết quả cách nhau 1s. Hãy xác định giá trị của điện áp  $V_0$  và độ lớn của tụ điện  $C$ .



t(s)	1	2	3	4	5	6	7	8	9	10
V(V)	9.4	7.31	5.15	3.55	2.81	2.04	1.26	0.97	0.74	0.58



Khi tụ điện được sạc đầy thì điện áp trên hai đầu cực tụ sẽ bằng  $V_0$ . Khi tụ điện phóng điện qua điện trở, điện áp trên hai bản cực tụ có thể mô tả theo hàm:

$$V = V_0 e^{-t/(RC)}$$

Từ công thức trên và từ dữ liệu thu được trong bảng, bằng cách sử dụng phương pháp xấp xỉ hàm để tìm công thức mô tả hàm của  $V$ , chúng ta có thể xác định được đại lượng  $\tau = RC$ , nghĩa là xác định được giá trị tụ điện  $C$ .

Từ biểu thức quan hệ trên, chúng ta thấy rằng giá trị  $V$  và  $t$  quan hệ với nhau theo hàm mũ. Để áp dụng hàm *polyfit()* chúng ta chuyển thành quan hệ trung gian tuyến tính giữa  $t$  và  $\ln(V)$  bằng việc lấy lô-ga-rít cơ số tự nhiên cả hai vế:  $\ln(V) = -\frac{1}{RC}t + \ln(V_0)$ . Biểu thức này có dạng  $y = mx + b$ . Giá trị  $m$  và  $b$  được xác định từ hàm *polyfit()*. Giá trị của tụ được xác định từ  $m$ :  $C = -1/Rm$ ; giá trị điện áp ban đầu được xác định từ  $b$ :  $V_0 = e^b$ .

```
R=2000;
t=1:10;
v=[9.4 7.31 5.15 3.55 2.81 2.04 1.26 0.97 0.74 0.58];
p=polyfit(t,log(v),1);
C=-1/(R*p(1));
V0=exp(p(2));
% Vẽ minh họa
tplot=0:0.1:10;
vplot=V0*exp(-tplot./(R*C));
plot(t,v,'o',tplot,vplot);
```

Ngoài việc sử dụng phương pháp xấp xỉ đường để tính toán các giá trị, MATLAB cũng cung cấp các hàm thư viện cho phép tính toán xấp xỉ theo phương pháp nội suy. Các hàm đó là *interp1()*, *interp2()*, ...

**Ví dụ 4-19:** Bảng dữ liệu sau là dữ liệu thu được trong quan sát giá trị của hàm  $f(x) = 1.5^x \cos(2x)$ . Sử dụng phương pháp nội suy để tính các giá trị còn lại của hàm (tại các điểm khác trong khoảng quan sát). Vẽ và minh họa kết quả.

$x$	0	1	2	3	4	5
$f(x)$	1.0	-0.6242	-1.4707	3.2406	-0.7366	-6.3717

Với số điểm dữ liệu đã cho, chúng ta sử dụng phương thức nội suy đa đường (spline)

```
>>x=0:1.0:5;
>>y=[1.0 -0.6242 -1.4707 3.2406 -0.7366 -6.3717];
>>xi=0:0.1:5;
>>yi=interp1(x,y,xi,'spline');
>>yfun=1.5^xi.*cos(2*xi);
>>plot(x,y,'o',xi,yfun,xi,yi,'-');
>>xlabel('x');
>>ylabel('y');
```

### 4.3. ỨNG DỤNG MATLAB TRONG XỬ LÝ TÍN HIỆU SỐ

#### 4.3.1. Giới thiệu chung

Không chỉ là công cụ lựa chọn của các sinh viên ngành kỹ thuật nói chung, các kỹ sư các nhà khoa học nói riêng, MATLAB còn được biết đến như một bộ công cụ nổi bật cho việc học tập, nghiên cứu và triển khai thử nghiệm các thuật toán các phương pháp xử lý tín hiệu số mới. MATLAB được trang bị không chỉ một bộ công cụ Xử lý tín hiệu số cho phép các phép xử lý cơ bản tín hiệu số mà còn được trang bị cả những bộ công cụ xử lý tín hiệu số nâng cao như Hệ thống xử lý tín hiệu số (DSP System Toolbox), bộ công cụ xử lý ảnh số (Digital Image Processing Toolbox), cho đến các bộ công cụ hỗ trợ thu thập tín hiệu âm thanh, hình ảnh để thực hiện xử lý. Trong phần này, chúng ta xem xét một số ví dụ điển hình của việc áp dụng lợi thế của bộ công cụ MATLAB và giải các bài toán xử lý tín hiệu số.

Để hiểu rõ hơn các kiến thức cơ bản của Xử lý tín hiệu số, người học có thể tham khảo thêm trong các tài liệu bài giảng [18], [19], [20].

#### 4.3.2. Một số bài toán xử lý tín hiệu số giải bằng MATLAB

##### 4.3.2.1. Lấy mẫu tín hiệu, biểu diễn tín hiệu

**Ví dụ 4-20:** Cho một tín hiệu liên tục có hàm mô tả:

$$y(t) = 20 \sin(50\pi t) + 40 \cos(100\pi t) - 10 \sin(300\pi t).$$

Thực hiện xác định tần số lấy mẫu theo tiêu chuẩn Nyquist. Sử dụng tần số lấy mẫu tìm được thực hiện lấy mẫu tín hiệu và minh họa tín hiệu trước và sau lấy mẫu.

Chúng ta thấy tín hiệu cho gồm 3 thành phần với các tần số lần lượt là 25Hz, 50Hz, và 150Hz. Theo tiêu chuẩn Nyquist, tần số lấy mẫu phải thỏa mãn tối thiểu bằng 2 lần tần số lớn nhất. Do đó, tần số lấy mẫu theo tiêu chuẩn Nyquist là  $f_s = 300\text{Hz}$ . Giả sử xem xét tín hiệu trong vòng 0.4s. Khi đó, bài toán có thể thực hiện trong MATLAB như sau:

```
Fs=300;
Ts=1/Fs;
t=0:Ts:0.4;
n=1:length(t);
yt=20*sin(50*pi*t)+40*cos(100*pi*t)-10*sin(300*pi*t);
yn=20*sin(50*pi*n*Ts)+40*cos(100*pi*n*Ts)-10*sin(300*pi*n*Ts);
% Hiển thị
subplot(2,1,1); plot(t,yn);
title('Tín hiệu'); xlabel('Thời gian (s)'); ylabel('Biên độ');
subplot(2,1,2); stem(1:length(t),yn);
title('Tín hiệu lấy mẫu');
xlabel('Chỉ số thời gian - n'); ylabel('Biên độ');
```

**Ví dụ 4-21:** Cho một tín hiệu rời rạc được mô tả bởi công thức toán học như sau:

$$x(n) = \begin{cases} 1 + \frac{n}{3} & \text{với } -3 \leq n \leq -1 \\ 1 & \text{với } 0 \leq n \leq 3 \\ 0 & \text{với các giá trị khác} \end{cases}$$

- Biểu diễn tín hiệu trên đồ thị
- Tìm tín hiệu  $x(n-4)$  và biểu diễn tín hiệu này
- Tìm tín hiệu  $x(-n)$  và biểu diễn tín hiệu này

Chúng ta thực hiện giải bài này với sự hỗ trợ của MATLAB như sau:

```
% - a: -----
n1=-3:-1;
x1=1+n1/3;
n2=0:3;
x2=ones(1,length(n2));
x=[zeros(1,5), x1, x2, zeros(1,5)];
n=[[-5:-1]+n1(1) n1 n2 n2(end)+[1:5]];
figure;
stem(n,x);
xlabel('Chỉ số thời gian - n ');
ylabel('Biên độ');
title('Biểu diễn tín hiệu');

% - b: -----
x4=zeros(1,length(n));
x4(5:end)=x(1:end-4);
figure;
stem(n,x4);
xlabel('Chỉ số thời gian - n ');
ylabel('Biên độ');
title('Biểu diễn tín hiệu trễ x(n-4)');

% - c: -----
nf=fliplr(n);
xf=fliplr(x);
figure;
stem(nf,xf);
xlabel('Chỉ số thời gian - n ');
ylabel('Biên độ');
title('Biểu diễn tín hiệu trễ x(-n)');
```

#### 4.3.2.2. Phân tích tín hiệu và hệ thống rời rạc trong miền thời gian

Các hàm thư viện cơ bản cho phép phân tích tín hiệu và hệ thống rời rạc trong miền thời gian được liệt kê trong bảng sau:

Hàm	Ý nghĩa	Ví dụ
conv()	Thực hiện tính tích chập	conv(x1,x2)
xcorr()	Tính tương quan chéo giữa hai tín hiệu hoặc tự tương quan của tín hiệu	xcorr(x1,x2)
filter()	Tìm đáp ứng ra của hệ thống	filter(a,b,x)
impz()	Tìm đáp ứng ra của hệ thống khi kích thích vào là xung Dirac	impz(a,b,N)
filt()	Xây dựng hàm truyền đạt (H(z)) của hệ thống	filt(a,b,Ts)

*Xem thêm tài liệu hướng dẫn sử dụng MATLAB*

**Ví dụ 4-23:** Một hệ thống tuyến tính bất biến (LTI) được mô tả bởi phương trình sai phân sau:

$$y(n) = 0.45x(n) + 0.5x(n-1) + 0.45x(n-2) + 0.53y(n-1) - 0.46y(n-2)$$

- Tìm hàm đáp ứng xung của hệ thống, vẽ minh họa hàm đáp ứng xung
- Tìm hàm truyền đạt của hệ thống, xác định điểm không và điểm cực, biểu diễn các điểm không và điểm cực trên mặt phẳng phức
- Vẽ minh họa đáp ứng biên độ phổ và pha của đáp ứng tần số của hệ thống
- Tìm đáp ứng ra của hệ thống với các kích thích vào là:

$$x(n) = \cos\left(\frac{20\pi n}{256}\right) + \cos\left(\frac{200\pi n}{256}\right) \text{ với } 0 \leq n \leq 299$$

Phương trình sai phân đã cho có thể viết lại như sau:  $y(n) - 0.53y(n-1) + 0.46y(n-2) = 0.45x(n) + 0.5x(n-1) + 0.45x(n-2)$ . Việc thực hiện giải theo lý thuyết bài toán trên được coi là bài tập luyện tập cho người học, ở đây chúng ta minh họa việc giải bài toán với sự hỗ trợ của MATLAB.

```
num=[0.45 0.5 0.45];
den=[1 -0.53 0.46];
% Biểu diễn hàm đáp ứng xung (như phần filter)
y=impz(num,den,50);
stem(y); title('Đáp ứng xung');
xlabel('Chỉ số thời gian (n)');
ylabel('Biên độ');
% -- Tìm hàm truyền đạt ---
Hz=filt(num,den,0.1);
% Biểu diễn điểm cực và không trên mặt phẳng phức
```

```

zplane(num,den);
% -- Vẽ đáp ứng biên độ phổ và pha -
freqz(num,den);
% -- Tính đáp ứng ra -----
% Tạo tín hiệu kích thích
n=0:299;
xn=cos(2*pi*10*n/256)+cos(2*pi*100*n/256);
% Tính đáp ứng ra với kích thích xn
y=filter(num,den,xn);
% Vẽ minh họa kết quả
figure;
subplot(2,1,1); stem(n,xn);
title('Tín hiệu vào');
xlabel('Chỉ số thời gian - n');
ylabel('Biên độ');
subplot(2,1,2); stem(n,y);
title('Tín hiệu ra');
xlabel('Chỉ số thời gian - n');
ylabel('Biên độ');

```

#### 4.3.2.3. Phân tích tín hiệu và hệ thống rời rạc trong miền z

Việc phân tích tín hiệu và hệ thống trong miền z cho phép chúng ta phân tích hoặc tổng hợp hệ thống một cách dễ dàng nhờ các phép toán có thể thực hiện đơn giản trong miền này. MATLAB cung cấp nhiều hàm thư viện hỗ trợ việc phân tích tín hiệu và hệ thống trong miền z. Một số hàm cơ bản thực hiện phân tích tín hiệu và hệ thống trong miền z cho trong bảng sau.

Hàm	Ý nghĩa	Ví dụ
filt()	Định nghĩa hàm truyền đạt của hệ thống	filt(a,b,Ts)
ztrans()	Thực hiện biến đổi z của một hàm số được định nghĩa theo công thức toán biểu thức	ztrans(f)
iztrans()	Thực hiện biến đổi z ngược của biểu thức hàm số được định nghĩa theo công thức toán biểu thức	iztrans(f)
zplane()	Biểu diễn các điểm cực, không của hàm truyền đạt trên mặt phẳng z	zplane(a,b)
residuez()	Thực hiện phân tích hàm truyền đạt $H(z)$ thành tổng các thành phần hữu tỷ	[r,p,k]=residuez(a,b)
freqz()	Trả về đáp ứng tần số, pha của hệ thống	freqz(a,b)

*Xem thêm tài liệu hướng dẫn sử dụng MATLAB*

**Ví dụ 4-24:** Xét một hệ thống tuyến tính bất biến có đáp ứng xung là:

$$h = \{1, \sqrt{2}, 1\}$$

- Tìm hàm truyền đạt  $H(z)$
- Tìm các điểm cực, điểm không của hàm truyền đạt  $H(z)$ , biểu diễn các điểm không và điểm cực trên mặt phẳng phức.
- Vẽ đáp ứng phổ pha và phổ biên độ của hệ thống
- Tính và vẽ minh họa  $H(z)|_{z=e^{j\omega}}$  với số điểm  $N = 16$

Dễ dàng có hàm truyền đạt của hệ thống:  $H(z) = 1 + \sqrt{2}z^{-1} + z^{-2}$ . Ta có thể tìm bằng MATLAB như sau:

```
num=[1 sqrt(2) 1];  
den=1;  
Hz=filt(num,den,0.1);
```

Do hàm truyền đạt của hệ thống không phải là hàm hữu tỉ (theo  $z^{-1}$ ) nên dễ thấy hàm truyền đạt có điểm cực kép tại gốc tọa độ, và điểm không là nghiệm của phương trình  $H(z) = 0$ . Các điểm không được tìm với sự hỗ trợ của MATLAB như sau:

```
num=[1 sqrt(2) 1];  
z=roots(num)
```

Các điểm cực và không của hàm truyền đạt có thể biểu diễn trên mặt phẳng phức với sự hỗ trợ của hàm thư viện `zplane`.

```
num=[1 sqrt(2) 1];  
den=1;  
zplane(num,den);
```

Đáp ứng phổ biên độ và pha của hệ thống có thể được biểu diễn với sự hỗ trợ của hàm `freqz()`

```
num=[1 sqrt(2) 1];  
den=1;  
[Hw,w]=freqz(num,den);  
% Vẽ phổ pha  
figure;  
plot(w/(2*pi),abs(Hw));  
xlabel('Tan so chuan hoa');  
ylabel('Bien do');  
title('Pho bien do');  
figure;  
plot(w/(2*pi),angle(Hw));  
xlabel('Tan so chuan hoa');  
ylabel('Goc pha');  
title('Pho pha');
```

Việc tính trong câu d) chính là thực hiện tính các giá trị  $H(z)$  trên đường tròn đơn vị, nói cách khác chính là tính các giá trị của biến đổi Fourier rời rạc. Ta có thể thực hiện với sự hỗ trợ của MATLAB như sau:

```
H=fftshift(fft(num,16));
N=length(H);
fqv=[-0.5:1/N:0.5-1/N];
stem(fqv,abs(H));
title('Gia tri tuyet doi ham truyen dat');
xlabel('Tan so chuan hoa');
ylabel('Bien do');
```

#### 4.3.2.4. Phân tích tín hiệu và hệ thống rời rạc trong miền tần số

Danh sách một số hàm thư viện cơ bản cho phép phân tích tín hiệu và hệ thống rời rạc trong miền tần số được cho trong bảng sau.

Hàm	Ý nghĩa	Ví dụ
fourier()	Thực hiện tìm biến đổi Fourier của biểu thức định nghĩa bởi hàm toán biểu thức	fourier(f)
ifourier()	Thực hiện tìm biến đổi Fourier ngược của biểu thức định nghĩa bởi hàm toán biểu thức	ifourier(f)
fft()	Thực hiện biến đổi Fourier rời rạc theo thuật toán biến đổi nhanh	fft(x,N)
ifft()	Thực hiện biến đổi Fourier rời rạc ngược theo thuật toán biến đổi nhanh	ifft(y,N)

*Xem thêm tài liệu hướng dẫn sử dụng MATLAB*

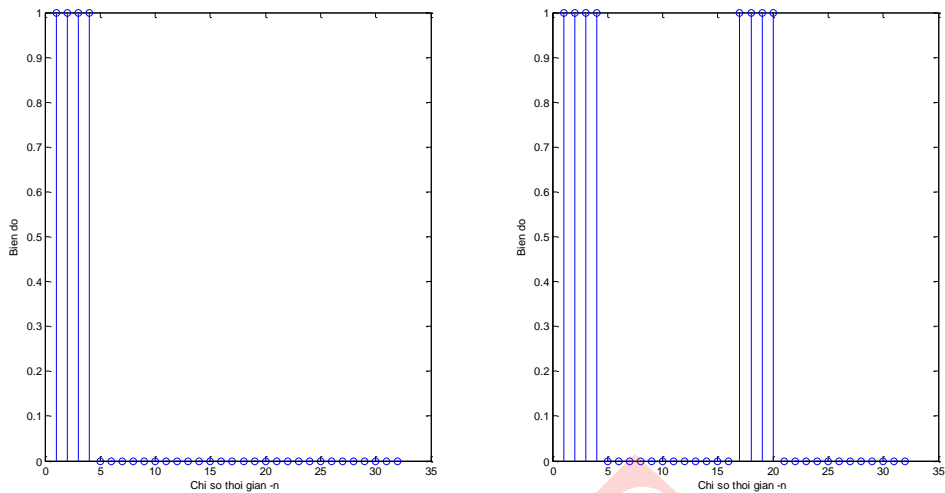
**Ví dụ 4-25:** Tính và vẽ phổ của tín hiệu số được mô tả bởi:

$$x(n) = \begin{cases} 1 & \text{nếu } n = 0 \div 7 \\ 0 & \text{các giá trị còn lại} \end{cases}$$

Bài này có thể thực hiện giải với sự hỗ trợ của MATLAB như sau

```
x=ones(1,8);
X=fft(x,128);
subplot(3,1,1);stem(x);
xlabel('n');ylabel('x(n)');
subplot(3,1,2);stem(abs(X));
xlabel('k');ylabel('Bien do pho');
subplot(3,1,3);stem(angle(X));
xlabel('k');ylabel('Pho pha');
```

**Ví dụ 4-26:** Cho hai tín hiệu minh họa trong hình vẽ, thực hiện phân tích phổ và biểu diễn phổ của các tín hiệu, nhận xét kết quả.



Chúng ta thấy rằng tín hiệu phía bên trái là không tuần hoàn trong khi đó tín hiệu bên phải có thể coi là gần tuần hoàn. Để thực hiện phân tích phổ của tín hiệu chúng ta sử dụng hàm thư viện *fft()* trong MATLAB như sau:

```

xd=[ones(1,4) zeros(1,12)];
xleft=[xd zeros(1,16)];
xright=[xd xd];
% Tính phổ của các tín hiệu
f=0:63;
xleft_f=abs(fft(xleft,64));
xright_f=abs(fft(xright,64));
% Vẽ minh họa
subplot(2,2,1); stem(xleft);
xlabel('Chi so thoi gian - n');
ylabel('Bien do');
title('Tin hieu');
subplot(2,2,2); plot(f,xleft_f);
xlabel('Tan so');
ylabel('Bien do pho');
title('Pho tin hieu');
subplot(2,2,3); stem(xright);
xlabel('Chi so thoi gian - n');
ylabel('Bien do');
title('Tin hieu');
subplot(2,2,4); plot(f,xright_f);
xlabel('Tan so');
ylabel('Bien do pho'); title('Pho tin hieu');

```



## 4.4. ỨNG DỤNG MATLAB TRONG LÝ THUYẾT ĐIỀU KHIỂN TỰ ĐỘNG

### 4.4.1. Giới thiệu chung

Phần này sẽ giới thiệu sơ lược về bộ công cụ Hệ thống điều khiển (Control System Toolbox) và ứng dụng minh họa thực hiện một số nhiệm vụ (phân tích/tổng hợp/đánh giá) đơn giản của bài toán điều khiển.

Các trình bày ở đây dựa trên giả định rằng người học đã có những kiến thức cơ bản về các hệ thống động và hiểu biết cơ bản về Lý thuyết điều khiển tuyến tính. Trong trường hợp cần củng cố kiến thức cơ bản, người học có thể tham khảo trong các tài liệu [21]; [22].

### 4.4.2. Một số bài toán điều khiển giải bằng MATLAB

#### 4.4.2.1. Biểu diễn hệ thống thông qua hàm truyền đạt, hệ phương trình trạng thái

MATLAB với bộ công cụ Hệ thống điều khiển cung cấp các hàm thư viện thuận tiện cho phép định nghĩa và biểu diễn một hệ thống tuyến tính bất biến. Các hàm cơ bản được liệt kê trong bảng sau:

Hàm	Ý nghĩa	Ví dụ
tf()	Biểu diễn hàm truyền đạt cho hệ thống từ hệ số của tử số và mẫu số hàm truyền đạt	tf(num,den,...)
ss()	Biểu diễn mô hình không gian trạng thái của một hệ thống từ các ma trận trạng thái	ss(A,B,C,D)
zpk()	Biểu diễn mô hình điểm không, điểm cực và hệ số của hệ thống	zpk(z,p,k)
tf2ss()	Thực hiện chuyển đổi biểu diễn hàm truyền đạt hệ thống sang mô hình không gian trạng thái	tf2ss(Hs)

*Xem thêm tài liệu hướng dẫn sử dụng bộ công cụ Hệ thống điều khiển*

**Ví dụ 4-27:** Xây dựng một hàm truyền đạt cho hệ thống tuyến tính bất biến liên tục theo thời gian với biểu thức hàm truyền đạt cho bởi công thức:

$$H(s) = K \frac{s}{Ts+1}, K = 2, T = 4$$

```
>> K=2;  
>> T=4;  
>> num=[K, 0];  
>> den=[T, 1];  
>> H=tf(num,den);
```

**Ví dụ 4-28:** Xây dựng mô hình không gian trạng thái của hệ thống có mô tả như sau:

$$\begin{cases} \frac{dx_1}{dt} = x_2 \\ \frac{dx_2}{dt} = -4x_1 - 2x_2 + 2u \\ y = x_1 \end{cases}$$

Từ mô tả của hệ thống đã cho, dễ dàng thấy được các ma trận trạng thái:

$A = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix}$ ,  $B = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ ,  $C = [1 \ 0]$ , và  $D = [0]$ . Khi đó, mô hình không gian trạng thái của hệ thống có thể được xây dựng bởi hàm `ss()` trong MATLAB như sau:

`A=[0 1; -4 -2];`

`B=[0; 2];`

`C=[1, 0];`

`D=[0];`

`sm=ss (A, B, C, D) ;`

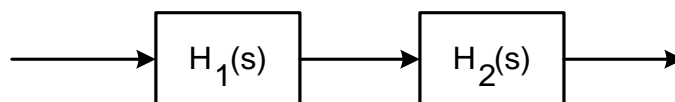
#### 4.4.2.2. Kết hợp các hệ thống: nối tiếp, song song, hồi tiếp

Các hàm cơ bản trong MATLAB cho phép tìm hàm tổng hợp từ các mô hình nối tiếp, song song hoặc hồi tiếp của các hệ thống được liệt kê trong bảng sau:

Hàm	Ý nghĩa	Ví dụ
<code>series()</code>	Kết hợp hai hệ thống mắc nối tiếp	<code>series(H1,H2)</code>
<code>parallel()</code>	Kết hợp hai hệ thống mắc song song	<code>parallel(H1,H2)</code>
<code>feedback()</code>	Kết hợp hai hệ thống mắc tạo thành một vòng hồi tiếp	<code>feedback(H1,H2,-1)</code>

Xem thêm trong tài liệu hướng dẫn sử dụng bộ công cụ Hệ thống điều khiển

**Ví dụ 4-29:** Xét hệ thống gồm hai khối mắc nối tiếp như hình vẽ, trong đó  $H_1(s) = \frac{1}{s+2}$  và  $H_2(s) = \frac{3}{s+4}$ . Tìm hàm truyền đạt tổng của hệ thống.



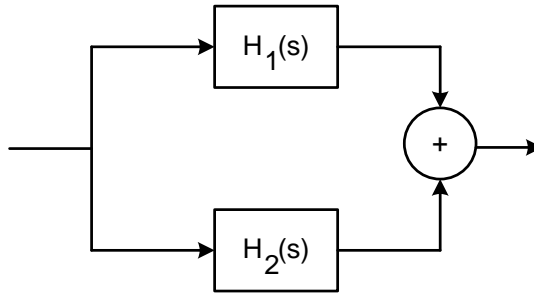
Chúng ta biết rằng, với hệ thống gồm các khối mắc nối tiếp thì hàm truyền đạt tổng (trong miền  $s$ ) sẽ là tích của các hàm truyền đạt thành phần. Kết quả này có thể kiểm nghiệm đơn giản trong MATLAB như sau:

`>>H1=tf (1, [1 2]) ;`

`>>H2=tf (3, [1 4]) ;`

`>>H=series (H1,H2) ;`

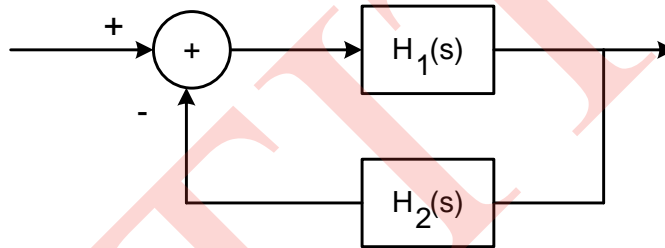
**Ví dụ 4-30:** Xét một hệ thống gồm hai khối mắc song song như hình vẽ, trong đó  $H_1(s) = \frac{1}{s+2}$  và  $H_2(s) = \frac{3}{s+4}$ . Tìm hàm truyền đạt tổng của hệ thống.



Với hệ thống gồm các khối mắc song song, hàm truyền đạt tổng của hệ thống (trong miền  $s$ ) sẽ là tổng của các hàm truyền đạt. Kết quả được kiểm nghiệm với sự hỗ trợ của MATLAB như sau:

```
>>H1=tf(1,[1 2]);
>>H2=tf(3,[1 4]);
>>H=parallel(H1,H2);
```

**Ví dụ 4-31:** Xét một hệ thống hồi tiếp như sơ đồ hình vẽ, trong đó  $H_1(s) = \frac{1}{s+2}$  và  $H_2(s) = \frac{3}{s+4}$ . Tìm hàm truyền đạt tổng của hệ thống.



Theo lý thuyết điều khiển tự động, hàm truyền đạt của hệ thống được tính bởi công thức  $H(s) = \frac{H_1(s)}{1+(s)H_2(s)}$ . Kết quả này có thể kiểm nghiệm với sự hỗ trợ của MATLAB như sau:

```
>>H1=tf(1,[1 2]);
>>H2=tf(3,[1 4]);
>>H=feedback(H1,H2,-1);
```

#### 4.4.2.3. Phân tích hệ thống

Với sự hỗ trợ của MATLAB, việc phân tích hệ thống chẳng hạn như xét tính ổn định... có thể dễ dàng thực hiện với kết quả trực quan. Danh sách các hàm thư viện cơ bản cho phép phân tích hệ thống cho trong bảng sau:

Hàm	Ý nghĩa	Ví dụ
pole()	Xác định các điểm cực của hàm truyền đạt của hệ thống	pole(H)
tzero()	Xác định các điểm không của hàm truyền đạt của hệ thống	tzero(H)
pzmap()	Định vị các điểm cực và điểm không và vẽ trên mặt phẳng $s$	pzmap(H)

step()	Đáp ứng của hệ thống với kích thích vào là xung bước nhảy	step(H)
impulse()	Đáp ứng của hệ thống với kích thích vào là xung Dirac	impulse(H)
bode()	Vẽ đồ thị Bode của đáp ứng tần số	bode(H)
nyquist()	Vẽ đồ thị Nyquist của đáp ứng tần số	nyquist(H)
rlocus()	Vẽ quỹ đạo nghiệm số (Root locus) hàm truyền đạt	rlocus(H)
rlocfind()	Tìm giới hạn hệ số độ lợi cho hệ thống ổn định	rlocfind(H)

*Xem thêm tài liệu hướng dẫn sử dụng bộ công cụ Hệ thống điều khiển*

**Ví dụ 4-32:** Xét một hệ thống điều khiển có hàm truyền đạt  $H(s) = \frac{s+0.5}{s^2+2s+4}$ .

- Tìm các điểm không, điểm cực của hàm truyền đạt, biểu diễn các điểm không và điểm cực trên mặt phẳng phức.
- Minh họa đáp ứng của hệ thống với kích thích vào là bước nhảy đơn vị, là xung Dirac

Bài toán trên có thể giải với sự hỗ trợ của MATLAB như sau:

```
num=[1 0.5];
den=[1 2 4];
Hs=tf(num,den);
% Tính toán điểm cực
p=pole(Hs)
% Tính toán điểm không
z=tzero(Hs)
% Minh họa điểm cực và điểm không trên mặt phẳng phức
pzmap(Hs)
% Tìm đáp ứng của hệ thống với kích thích bước nhảy đơn vị
Tf=10;
step(Hs,Tf);
% Tìm đáp ứng của hệ thống với kích thích xung Dirac
impulse(Hs,Tf);
```

**Ví dụ 4-33:** Một hệ thống tuyến tính có hàm truyền đạt  $H(s) = \frac{10}{s^2+2s+10}$ . Tín hiệu kích thích vào  $u(t) = 2\sin(3t)$ .

- Tính đáp ứng ra của hệ thống trong khoảng 10s đầu tiên. Biểu diễn trên cùng một đồ thị tín hiệu kích thích và tín hiệu đáp ứng ra.

- b) Khảo sát hoạt động của hệ thống với các tần số khác  $\omega_1 = 1$ ,  $\omega_2 = 2$ ,  $\omega_3 = 5$ ,  $\omega_4 = 10$

Chúng ta thực hiện giải bài toán trên với sự hỗ trợ của MATLAB như sau:

```
num=10;
den=[1 2 10];
Hs=tf(num,den);
% Xét khoảng thời gian 10s đầu tiên
t=0:0.05:10;
u=2*sin(3*t);
% Tính đáp ứng ra ứng với kích thích vào u(t)
y=lsim(Hs,u,t);
% Biểu diễn kích thích và đáp ứng ra
plot(t,u,'r',t,y,'b');
grid on; xlabel('Thời gian (s)'); ylabel('Biên độ');
% Xét khoảng tần số khảo sát
omega = 1:0.1:10;
% Tính hệ số đáp ứng biên độ và pha
[K,p]=bode(Hs,omega); % hoặc vẽ bằng lệnh bode(num,den,omega)
% Minh họa
subplot(2,1,1); semilogx(omega,20*log10(K(:)));
xlabel('Tan so goc (omega)'); ylabel('Biên độ');
subplot(2,1,2); semilogx(omega,20*log10(p(:)));
xlabel('Tan so goc (omega)'); ylabel('Pha');
% Phần vẽ đặc tính biên độ và pha có thể thực hiện bởi
% lệnh bode(num,den,omega)
```

Theo lý thuyết điều khiển tự động, một hệ thống vòng kín được gọi là ổn định theo tiêu chuẩn BIBO nếu với các kích thích vào có độ lớn hữu hạn thì hệ thống cũng tạo ra các đáp ứng ra có độ lớn hữu hạn. Để đánh giá tính ổn của một hệ thống chúng ta có nhiều phương pháp.

Phương pháp thứ nhất là dựa trên vị trí các điểm cực của hàm truyền đạt vòng kín: Một hệ thống vòng kín được cho là ổn định nếu các điểm cực có phần thực có giá trị âm, nói cách khác, các điểm cực phải nằm ở nửa bên trái của mặt phẳng phức. Như vậy, bằng cách sử dụng MATLAB, từ biểu diễn trực quan các điểm cực của hàm truyền đạt hệ thống là chúng ta có thể có ngay kết luận về tính ổn định.

**Ví dụ 4-34:** Xét một hệ thống điều khiển có hàm truyền đạt vòng kín (hàm truyền đạt tổng quát của hệ thống) cho bởi biểu thức  $H(s) = \frac{s+5}{s^5-3s^4+4s^3+10s^2+5s-10}$ . Hãy cho biết hệ thống đã cho có ổn định không?

Thực hiện đánh giá tính ổn định của hệ thống dựa trên vị trí các điểm cực, chúng ta có thể giải bài này với sự hỗ trợ của MATLAB như sau:

```

num=[1 5];
den=[1 -3 4 10 5 -10];
Hs=tf(num,den);
% Tính các điểm cực bằng hàm roots() của mẫu số
poles=roots(den)
% Hoặc bằng hàm thông số cực, không, hệ số
[z,p,k]=zpkdata(Hs,'v');
% Hoặc bằng cách biểu diễn trực quan các điểm cực, không
pzmap(Hs);

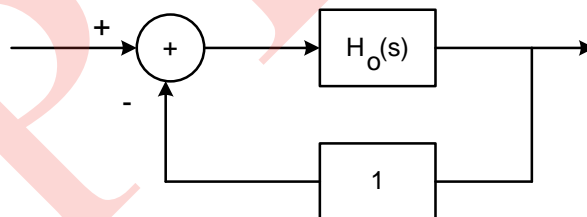
```

Sau khi có các điểm cực, đối chiếu với tiêu chuẩn chúng ta thấy hệ thống có các điểm cực nằm ở nửa bên phải của mặt phẳng phức nên dễ dàng kết luận: hệ thống là không ổn định.

Phương pháp thứ hai dựa trên điều kiện ổn Nyquist: Sự ổn định của một hệ thống vòng kín có thể được xác định thông qua hàm truyền đạt của hệ thống vòng hở thành phần của hệ thống: (1) Nếu hàm truyền đạt vòng hở của hệ thống không có các điểm cực không ổn định và tất cả các điểm cực của hàm truyền đạt vòng hở của hệ thống nằm ở nửa trái của mặt phẳng phức: Hệ thống vòng kín ổn định nếu đường Nyquist của hệ thống vòng hở không bao điểm  $-1 + 0i$ ; Hoặc (2) Nếu hàm truyền đạt vòng hở của hệ thống có các điểm cực không ổn định: Hệ thống vòng kín ổn định nếu đường Nyquist của hệ thống vòng hở bao điểm  $-1 + 0i$  theo chiều ngược kim đồng hồ số lần bằng số điểm cực không ổn định của hệ thống vòng hở.

**Ví dụ 4-35:** Xét một hệ thống hồi tiếp có sơ đồ như hình vẽ, trong đó:

$H_o(s) = \frac{10}{(1+10s)(1+s)}$ . Sử dụng tiêu chuẩn ổn định Nyquist đánh giá xem hệ thống có ổn định không?



Dựa trên lý thuyết về điều kiện ổn định theo tiêu chuẩn Nyquist, chúng ta có thể giải bài toán với sự hỗ trợ của MATLAB như sau:

```

% Dùng biến trung gian
s=tf('s');
Ho=10/((1+10*s)*(1+s));
% Kiểm tra xem có điểm cực không ổn định
[z,p,k]=zpkdata(Ho,'v')
% Minh họa đường Nyquist để đánh giá tính ổn định
nyquist(Ho); grid;

```

Quan sát kết quả chúng ta thấy, hệ thống vòng hở thành phần không có điểm cực không ổn định, đường Nyquist không bao điểm  $-1 + 0i$ , do đó có thể kết luận rằng: Hệ thống vòng kín là ổn định.

**Ví dụ 4-36:** Cho hệ thống điều khiển có hàm truyền đạt  $H(s) = \frac{2s^2+5s+1}{s^2+2s+3}$ . Minh họa quỹ đạo nghiệm số của hệ thống.

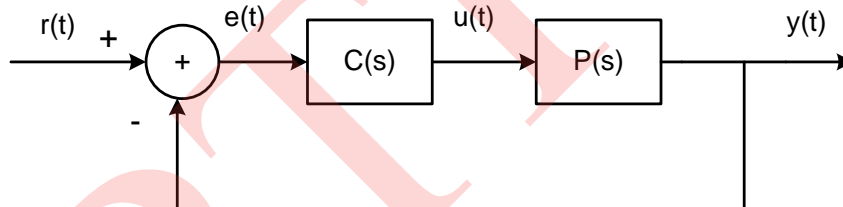
Với sự hỗ trợ của MATLAB, bài toán có thể giải như sau:

```
num=[2 5 1];
den=[1 2 3];
H=tf(num,den);
rlocus(H);
title('Quỹ đạo nghiệm số');
```

#### 4.4.2.4. Tổng hợp (thiết kế) hệ thống

Với bộ công cụ Điều khiển hệ thống và các hàm thư viện của MATLAB, việc thực hiện bài toán thiết kế hệ thống điều khiển cũng có thể được thực hiện một cách đơn giản và dễ dàng. Tuy nhiên, bài toán thiết kế hệ thống điều khiển là một bài toán rất phức tạp, trong phần này chúng ta chỉ minh họa với một ví dụ đơn giản việc thiết kế một hệ thống điều khiển bù PID nối tiếp.

Sơ đồ của hệ thống như hình vẽ, trong đó  $P(s)$  là hàm truyền đạt của quá trình cần điều khiển,  $C(s)$  là hàm truyền đạt cần được xây dựng theo kiểu bù nối tiếp để thực hiện việc điều khiển quá trình  $P(s)$ .



Dễ dàng có hàm truyền đạt vòng hở của hệ thống  $H_o(s) = C(s)P(s)$  và hàm truyền đạt hệ thống  $H(s) = \frac{H_o(s)}{1+H_o(s)}$

Bài toán thiết kế là xây dựng hàm  $C(s)$  để thu được một hệ thống đạt yêu cầu các chỉ tiêu kỹ thuật cụ thể, chẳng hạn như ổn định, điều khiển chính xác trong quá trình ổn định, thời gian quá độ nhỏ,...

**Ví dụ 4-37:** Thiết kế bộ điều khiển loại PID (khâu tỷ lệ - tích phân - vi phân) cho quá trình cần điều khiển có hàm truyền đạt  $P(s) = \frac{1}{(1+10s)(1+s)(1+0.2s)}$

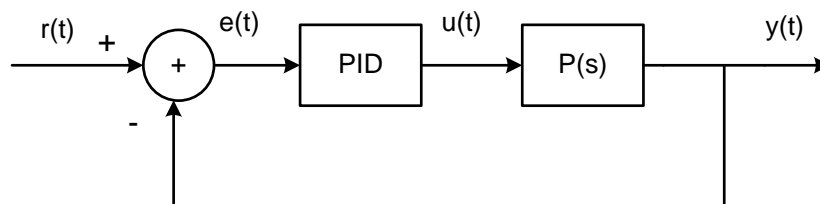
```
% Tạo biến trung gian
s=tf('s');
Ps=1/((1+10*s)*(1+s)*(1+0.2*s));
% Biểu diễn P(s) dạng cực-không-hằng
P=zpk(Ps);
% Lập hàm truyền đạt các khâu thành phần
% Khâu tỷ lệ
Cp=1;
```

```

% Khâu tỷ lệ - tích phân
Cpi=(1+10*s)/(10*s);
% Khâu tỷ lệ - vi phân
Cpd=(1+s)/(1+0.1*s);
% Khâu tỷ lệ - tích phân - vi phân
Cpid=Cpi*Cpd;
Cpid=zpk(Cpid);
% Vẽ đồ thị Bode để xác định một số tham số quan trọng
[mag,phase,w]=bode(Cpid*P);
% Tính độ khuếch đại biên với pha biên ~60o
kpid=margin(mag,phase-60,w);
% Hàm truyền đạt đầy đủ của khâu bù C(s)
Cpid=kpid*Cpid;
% -- Kiểm tra hệ thống tổng hợp được
% Hàm truyền đạt của hệ thống
H=feedback(Cpid*P,1);
% Đáp ứng của hệ thống với kích thích bước nhảy
figure(1); step(H);
title('Dap ung he thong voi kích thích buoc nhay');
xlabel('Thoi gian (s)'); ylabel('Bien do');
% -- Minh họa tín hiệu điều khiển u(t)
Upid=feedback(Cpid,P);
t=0:0.05:10;
u=step(Upid,t);
figure(2); plot(t,u);
title('Tin hieu dieu khien');
xlabel('Thoi gian (s)'); ylabel('Bien do');

```

**Ví dụ 4-38:** Một hệ thống điều khiển có sơ đồ như hình vẽ, trong đó khối điều khiển PID có hàm truyền đạt  $H_{pid}(s) = K \frac{(s+4)(s+2)}{s}$ , hệ thống cần điều khiển có hàm truyền đạt  $P(s) = \frac{1}{s^2-1}$ . Khảo sát giới hạn của  $K$  để hệ thống vẫn thỏa mãn ổn định.



Dễ dàng thấy rằng đây là bài toán điều hình của việc thiết kế hệ thống dùng quỹ đạo nghiệm số (Root locus design). Phương trình đặc trưng của hệ thống vòng kín là:

$$1 + H_{pid}(s)P(s) = 0.$$



Với sự hỗ trợ của MATLAB, việc tìm giá trị giới hạn của  $K$  có thể được thực hiện như sau:

```
num=[1 6 8];
den=[1 0 -1 0];
H=tf(num,den);
rlocus(H);
title('Do thi quy dao nghiem so');
% Khao sat, tim gia tri K gioi han
grid on;
axis('equal');
[K,poles]=rlocfind(H);
```

#### 4.4.2.5. Mô phỏng hệ thống điều khiển trên Simulink

Simulink là bộ công cụ rất mạnh của MATLAB cho phép xây dựng các mô hình khối mô phỏng các hệ thống một cách đơn giản, trực quan. Các nhóm khối cơ bản trong Simulink được liệt kê trong bảng sau:

Nhóm khối	Các khối cơ bản thuộc nhóm
Nguồn (Source)	Step: Khối tạo ra tín hiệu hàm bước nhảy Signal Generator: Khối tạo các dạng sóng cơ bản
Đích (Sink)	Scope: Khối hiển thị XY Graph: Hiển thị tương quan tín hiệu (quan sát pha)
Khối xử lý (Math Operations)	Sum: Khối thực hiện phép cộng tín hiệu Gain: Khối thực hiện khâu tỉ lệ 1/s: Khối vi phân

*Xem thêm tài liệu hướng dẫn sử dụng MATLAB Simulink*

**Ví dụ 4-39:** Một hệ thống điều khiển hoạt động được mô tả theo phương trình vi phân sau:

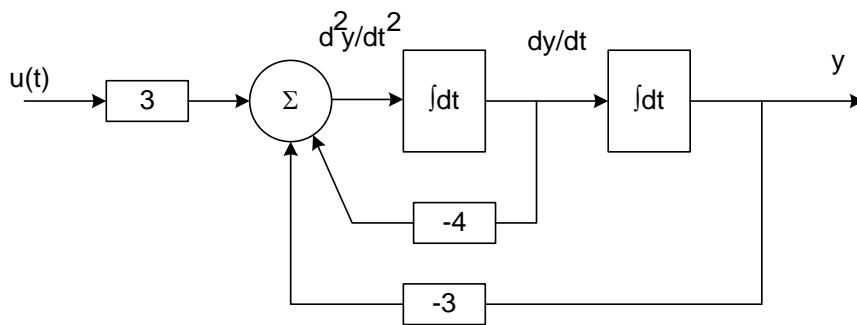
$$\frac{d^2y}{dt^2} + 4\frac{dy}{dt} + 3y = 3u(t)$$

Sử dụng bộ công cụ mô phỏng Simulink khảo sát hệ thống và hiển thị đáp ứng ra khi kích thích vào là  $u(t)$  (bước nhảy đơn vị). Biết điều kiện đầu  $y(0) = 0.5$  và  $y'(0) = 0$ .

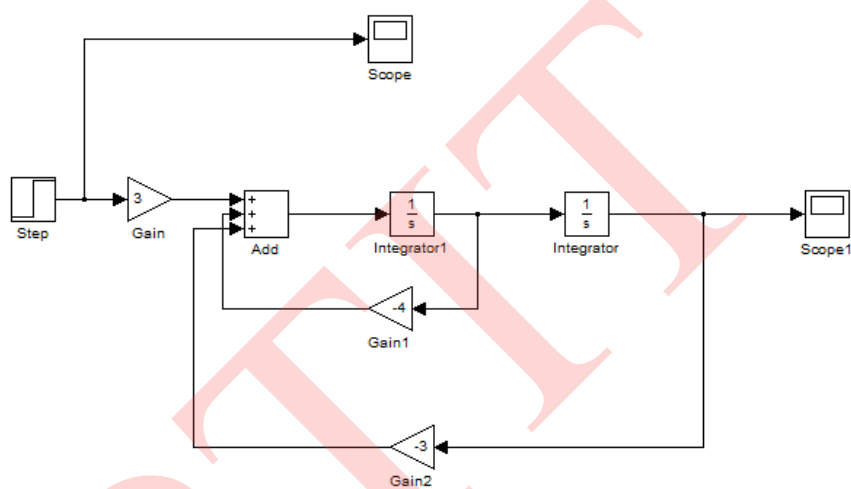
Từ phương trình mô tả hệ thống, chúng ta thấy kích thích vào là  $u(t)$  và tín hiệu ra là  $y(t)$ . Chúng ta viết lại phương trình như sau:

$$\frac{d^2y}{dt^2} = -4\frac{dy}{dt} - 3y + 3u(t)$$

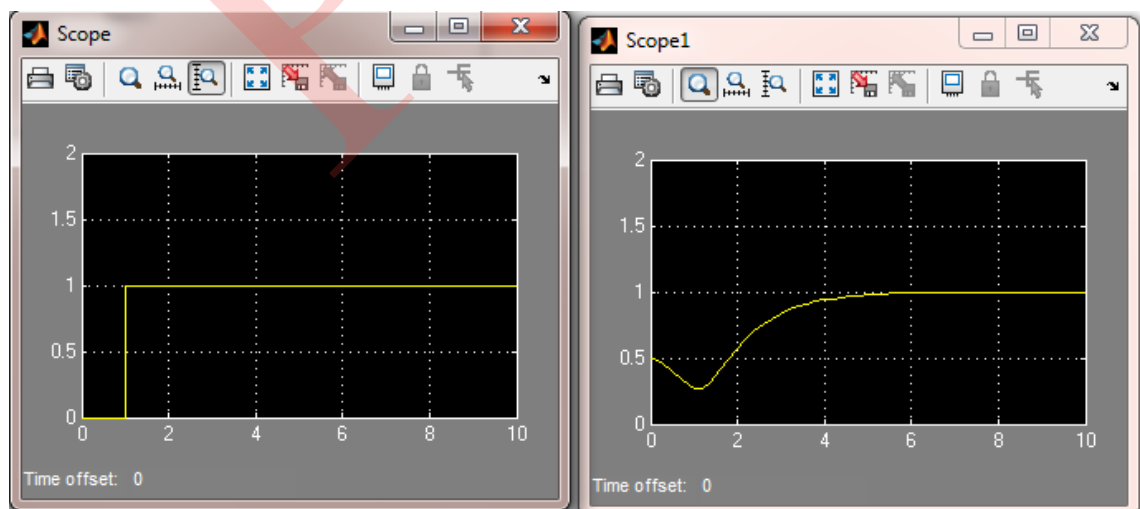
Với phương trình này chúng ta có thể biểu diễn hệ thống theo sơ đồ khối như sau:



Từ sơ đồ khối của hệ thống, chúng ta cần ba bộ nhân hằng số (khối gain) với các hệ số là 3, -4 và -3; hai bộ tích phân (khối 1/s) và một bộ cộng tổng ba đầu vào (khối add). Nguồn tạo tín hiệu đầu vào là khối tạo tín hiệu hàm bước nhảy đơn vị (khối step). Để hiển thị tín hiệu chúng ta sử dụng hai Ô-xi-lô (khối scope), một cho tín hiệu ra  $y(t)$  và một cho tín hiệu vào  $u(t)$ . Sơ đồ hệ thống trong simulink như sau:



Sau khi thiết lập các điều kiện đầu, thực hiện mô phỏng chúng ta có kết quả như sau.

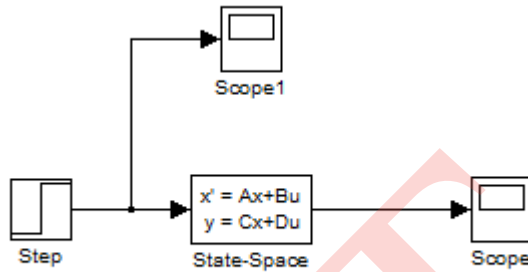


**Ví dụ 4-40:** Cho hệ thống được mô tả bởi hệ phương trình trạng thái như sau:

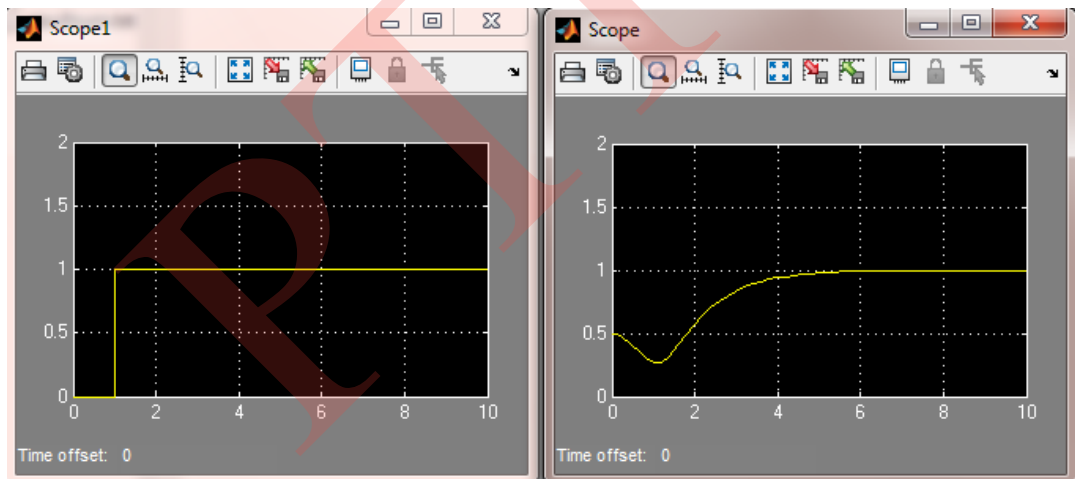
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -4 & -4 \\ 3/4 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 4 \\ 0 \end{bmatrix} u_0(t), \quad y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} u_0(t)$$

Thực hiện mô phỏng hệ thống bằng Simulink, tìm đáp ứng ra với kích thích hàm bước nhảy đơn vị. Biết rằng hệ thống có điều kiện đầu  $x_1 = 0, x_2 = 0.5$

Từ hệ phương trình trạng thái đã cho, dễ dàng thấy rằng  $A = \begin{bmatrix} -4 & -4 \\ 3/4 & 0 \end{bmatrix}$ ,  $B = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$ ,  $C = [0 \ 1]$ , và  $D = [0]$ . Việc thực hiện mô phỏng hệ thống điều khiển khi hệ thống được mô tả bằng phương trình trạng thái trên Simulink rất dễ dàng. Ngoài khối tín hiệu kích thích (hàm bước nhảy – khối step), các khối Ô-xi-lô (khối scope) để quan sát tín hiệu, chúng ta chỉ cần một khối mô hình trạng thái để mô tả mô hình không gian trạng thái của hệ thống. Sơ đồ hệ thống mô phỏng như sau:



Kích đúp khối không gian trạng thái và nhập các ma trận trạng thái A, B, C, và D, cùng các điều kiện đầu của hệ thống. Sau đó thực hiện chạy mô phỏng chúng ta có kết quả như sau:



## 4.5. ỨNG DỤNG MATLAB TRONG TRUYỀN THÔNG

### 4.5.1. Giới thiệu chung

MATLAB cung cấp hàng loạt hàm thư viện trong bộ công cụ Hệ thống truyền thông (Communication System Toolbox), tạo khả năng dễ dàng cho việc học tập nghiên cứu trong lĩnh vực truyền thông thông tin. Với sự hỗ trợ của bộ công cụ này cùng với các hàm thư viện khác của MATLAB, chúng ta không chỉ thuận tiện trong việc minh họa trực quan sinh động hóa những kiến thức lý thuyết, triển khai tiến hành phân tích, thiết kế các hệ thống cũng như đánh giá các hệ thống truyền thông. Trong phần này, chúng ta sẽ làm quen với cách sử dụng các công cụ của MATLAB để thực hiện giải các bài toán trong lĩnh vực truyền thông thông tin.

qua các ví dụ cụ thể. Các ví dụ được trình bày từ dễ đến khó với mặc định người học đã có một chút hiểu biết về lĩnh vực truyền thông.

Trong trường hợp cần trang bị một số kiến thức cơ bản về lĩnh vực truyền thông, người học có thể tham khảo thêm trong các tài liệu [23], [24].

## 4.5.2. Một số bài toán truyền thông giải bằng MATLAB

### 4.5.2.1. Biểu diễn tín hiệu liên tục, rời rạc

Trong phần đồ họa của MATLAB, chúng ta đã phần nào thấy được sức mạnh của MATLAB trong việc biểu diễn trực quan các đồ thị. Bằng việc kết hợp khả năng đó cùng với các hàm thư viện của bộ công cụ Hệ thống truyền thông, chúng ta có thể tạo và biểu diễn trực quan các tín hiệu liên tục, rời rạc, tuần hoàn, không tuần hoàn. Việc biểu diễn tín hiệu không chỉ có thể thực hiện trong miền thời gian, hay trong miền tần số mà còn cả trong miền kết hợp thời gian-tần số.

Một số hàm cơ bản cho việc tạo và biểu diễn tín hiệu được liệt kê trong bảng sau:

Hàm	Ý nghĩa	Ví dụ
sin()	Tạo tín hiệu theo hàm sin	$y = \sin(x)$
cos()	Tạo tín hiệu theo hàm cos	$y = \cos(x)$
sawtooth()	Tạo tín hiệu là dãy xung răng cưa với đỉnh là $\pm 1$ và chu kỳ $2\pi$	$y = \text{sawtooth}(x)$
square()	Tạo tín hiệu là dãy xung vuông với chu kỳ $2\pi$ với độ rộng tùy chọn	$y = \text{square}(x)$
tripuls()	Tạo một tín hiệu xung tam giác với độ lớn đỉnh cực đại bằng 1	$y = \text{tripuls}(x, 1e-3)$
rectpuls()	Tạo một tín hiệu xung vuông	$y = \text{rectpuls}(x, 1e-3)$

*Xem thêm trong tài liệu hướng dẫn sử dụng bộ công cụ Xử lý tín hiệu số và Hệ thống truyền thông*

**Ví dụ 4-41:** Biểu diễn tín hiệu tương tự.

Thực hiện tạo và biểu diễn khoảng 1,5s của một tín hiệu là dãy xung răng cưa có tần số 50Hz

```
t=0:1/1e+4:1.5;
f=50;
s=sawtooth(2*pi*f*t);
plot(t,s);
axis([0 0.2 -1.2 1.2]);
xlabel('Thời gian (s)');
ylabel('Biên độ');
title('Tín hiệu dãy xung răng cưa');
```

Thực hiện tạo và biểu diễn khoảng 1,5s của một tín hiệu sin có tần số 50Hz

```
t=0:1/1e+4:1.5;
f=50;
s=sin(2*pi*f*t);
plot(t,s);
axis([0 0.2 -1.2 1.2]);
xlabel('Thời gian (s)');
ylabel('Biên độ');
title('Tín hiệu sin');
```

Thực hiện tạo và biểu diễn 2s của một xung tam giác với độ rộng xung bằng 20ms

```
t=-1:1/1e+4:1;
w=20e-3;
s=tripuls(t,w);
plot(t,s);
axis([-0.2 0.2 -0.2 1.2]);
xlabel('Thời gian (s)');
ylabel('Biên độ');
title('Tín hiệu xung tam giác');
```

Thực hiện tạo và biểu diễn 2s của một xung chữ nhật với độ rộng xung bằng 20ms

```
t=-1:1/1e+4:1;
w=20e-3;
s=rectpuls(t,w);
plot(t,s);
axis([-0.2 0.2 -0.2 1.2]);
xlabel('Thời gian (s)');
ylabel('Biên độ');
title('Tín hiệu xung chữ nhật');
```

**Ví dụ 4-42:** Biểu diễn tín hiệu số

Thực hiện tạo và biểu diễn 50 mẫu đầu tiên của dãy xung Dirac

```
n=0:49;
xn=zeros(1,length(n));
xn(1)=1;
stem(n,xn);
axis([-1 50 -0.2 1.2]);
xlabel('Thời gian (n)');
ylabel('Biên độ');
title('Dãy xung Dirac');
```

Thực hiện tạo và biểu diễn 50 mẫu đầu tiên của dãy hàm mũ với hệ số mũ  $a = 0.5$

```
n=0:49;
```

```

a=0.5;
xn=a.^n;
stem(n,xn);
axis([-1 50 -0.2 1.2]);
xlabel('Thoi gian (n)');
ylabel('Bien do');
title('Day ham mu');

```

**Ví dụ 4-43:** Biểu diễn phổ của tín hiệu

Thực hiện biểu diễn tín hàm Dirac trong miền thời gian và miền tần số.

```

N=1000;
xt_delta=[ones(1,1) zeros(1,N)];
Xf_delta=fft(xt_delta,512);
Power_Xf=Xf_delta.*conj(Xf_delta)/512;
Power_Xf=Power_Xf/max(Power_Xf);
f=1000*(0:256)/512;
figure;
subplot(2,1,1); plot(1:length(xt_delta),xt_delta);
axis([-1 length(xt_delta) -0.2 1.2]);
xlabel('Thoi gian'); ylabel('Bien do');
title('Xung Dirac');
subplot(2,1,2); plot(f,Power_Xf(1:257));
axis([- (max(f)-min(f))*0.05 max(f)*1.1 0 1.1]);
xlabel('Tan so'); ylabel('Bien do pho');
title('Pho cua ham Dirac');

```

**Ví dụ 4-44:** Biểu diễn tín hiệu trong miền thời gian-tần số (spectrogram)

Một tín hiệu có hàm toán học mô tả như sau:

$$y(t) = \sin(100\pi t) - \sin(120\pi t) + \cos(240\pi t).$$

Quan sát đồ hình phổ-thời gian (spectrogram) của tín hiệu.

```

t=0:0.01:3;
f1=50;
f2=60;
f3=120;
y=sin(2*pi*f1*t)-sin(2*pi*f2*t)+cos(2*pi*f3*t);
spectrogram(y,128,32);

```

#### 4.5.2.2. Xử lý tín hiệu và hệ thống tương tự

Một số hàm cơ bản hỗ trợ xử lý tín hiệu tương tự cho trong bảng sau:

Hàm	Ý nghĩa	Ví dụ
-----	---------	-------

ammod	Thực hiện điều chế biên độ	$y = \text{ammod}(s, F_c, F_s)$
amdemod	Thực hiện giải điều chế biên độ	$r = \text{amdemo}(y, F_c, F_s)$
fmmod	Thực hiện điều chế tần số	$y = \text{fmmod}(s, F_c, F_s, \text{freqdev})$
fmdemod	Thực hiện giải điều chế tần số	$r = \text{fmdevmod}(y, F_c, F_s, \text{freqdev})$
pmmod	Thực hiện điều chế pha	$y = \text{pmmod}(s, F_c, F_s, \text{phasedev})$
pmdemod	Thực hiện giải điều chế pha	$r = \text{pmdemod}(y, F_c, F_s, \text{phasedev})$
ssbmod	Thực hiện điều chế đơn biên nén sóng mang	$y = \text{ssbmod}(s, F_c, F_s)$
ssbdemod	Thực hiện giải điều chế đơn biên nén sóng mang	$r = \text{ssbdemod}(y, F_s, F_s)$

*Xem thêm trong tài liệu hướng dẫn sử dụng bộ công cụ Hệ thống truyền thông*

**Ví dụ 4-45:** Biểu diễn tín hiệu điều chế

Một tín hiệu sin có tần số 10Hz được đưa vào thực hiện điều chế biên độ với tần số sóng mang 300Hz. Thực hiện tạo và biểu diễn tín hiệu trước và sau điều chế.

```
f=10;
fc=300;
fs=8e+3;
t=0:1/fs:0.1;
s=sin(2*pi*f*t);
y=ammod(s,fc,fs);
figure;
subplot(2,1,1); plot(t,s);
xlabel('Thời gian (s)');
ylabel('Biên độ');
title('Tín hiệu vào');
subplot(2,1,2); plot(t,y);
xlabel('Thời gian (s)');
ylabel('Biên độ');
title('Tín hiệu điều chế biên độ');
```

Một tín hiệu sin hai kênh: trong đó kênh 1 gồm có thành phần chính tần số 300Hz và một thành phần hài bậc hai; kênh 2 gồm có thành phần chính tần số 150Hz và một thành phần hài bậc 6. Tín hiệu này được đưa vào thực hiện điều chế tần số với sóng mang có tần số 3000Hz. Giả sử hệ thống điều chế có hằng số lệch tần (frequency deviation constant) bằng 50. Thực hiện tạo và biểu diễn tín hiệu trước và sau điều chế.

```
f1=300;
f2=150;
```

```

fc=3000;
freqdev=50;
fs=8e+3;
t=0:1/fs:1-1/fs;
s1=sin(2*pi*f1*t)+2*sin(2*pi*2*f1*t);
s2=sin(2*pi*f2*t)+2*sin(2*pi*6*f1*t);
s=[s1 s2];
y=fmmmod(s,fc,fs,freqdev);
figure;
subplot(2,1,1); plot(t,s);
xlabel('Thời gian (s)');
ylabel('Biên độ');
title('Tin hiệu vào');
subplot(2,1,2); plot(t,y);
xlabel('Thời gian (s)');
ylabel('Biên độ');
title('Tin hiệu điều chế biên độ');

```

#### 4.5.2.3. Mã hóa nguồn

Danh sách các hàm thư viện cơ bản hỗ trợ việc làm việc với mã hóa nguồn trong MATLAB được cho trong bảng sau:

Hàm	Ý nghĩa	Ví dụ
huffmandict()	Thực hiện mã hóa và xây dựng bộ từ điển mã hóa Huffman	huffmandict(sym,prob)
huffmanenco()	Thực hiện mã hóa một văn bản theo bảng mã Huffman xây dựng được từ huffmandict()	huffmanenco(msg,dict)
huffmandeco()	Thực hiện giải mã bản mã hóa Huffman	huffmandeco(rxmsg,dict)
arithenco()	Thực hiện mã hóa theo phương pháp mã hóa số học	arithenco(msg,freq)
arithdeco()	Thực hiện giải mã của bản mã hóa số học	arithdeco(code,freq,len)

*Xem thêm tài liệu hướng dẫn sử dụng bộ công cụ Hệ thống truyền thông*

#### **Ví dụ 4-46:** Mã hóa Huffman

Một nguồn rời rạc không nhớ  $X$  gồm các ký tự 1,2,3,4 với các xác suất xuất hiện lần lượt là  $1/2$ ,  $1/4$ ,  $1/8$  và  $1/8$ . (1) Thực hiện việc mã hóa nguồn theo phương pháp mã hóa Huffman. (2) Thực hiện sử dụng bộ mã xây dựng được mã hóa đoạn văn bản gồm các ký tự sau: 1341212



```

symbols=1:4;
p=[0.5 0.25 0.125 0.125];
[dict,avglen]=huffmandict(symbols,p);
% hien thi bo ma xay dung duoc
for ii=1:length(symbols)
    disp(['Ky tu: ',num2str(dict{ii,1}),'->', num2str(dict{ii,2})]);
end;
% Nhập bản tin cần mã hóa
msg=[1 3 4 1 2 1 2];
% Mã hóa bản tin
coded_msg=huffmanenco(msg,dict);

```

#### 4.5.2.4. Mã hóa kênh

Danh sách các hàm cơ bản cho phép làm việc với mã hóa kênh trong MATLAB được liệt kê trong bảng sau:

Hàm	Ý nghĩa	Ví dụ
cyclopoly()	Tạo đa thức sinh cho mã vòng tuyến tính (mã xyclic tuyến tính)	cyclopoly(15,5)
bchgenpoly()	Tạo đa thức sinh cho mã BCH	bchgenpoly(15,5)
rsngenpoly()	Tạo đa thức sinh cho mã RS	rsngenpoly(15,5)
cyclgen()	Tạo ma trận sinh và ma trận kiểm tra cho mã vòng tuyến tính	cyclgen(7,[1 0 1 1])
hamgen()	Tạo ma trận sinh và ma trận kiểm tra cho mã Hamming nhị phân	hamgen(m)
poly2trellis()	Chuyển mô tả mã từ đa thức sang giản đồ Trellis	poly2trellis(2,[35 31])
encode()	Thực hiện mã hóa mã khối tuyến tính	encode(msg,l,k,'cyclic')

*Xem thêm tài liệu hướng dẫn sử dụng bộ công cụ Hệ thống truyền thông*

**Ví dụ 4-47:** Xét một mã vòng tuyến tính  $C(7,4)$  với đa thức sinh  $g(x) = 1 + x + x^3$ . Xây dựng ma trận sinh và ma trận kiểm tra cho mã trên.

```

genpoly=[1 1 0 1];
[P,G]=cyclgen(7,genpoly);

```

#### 4.5.2.5. Hệ thống thông tin số

Như chúng ta đã biết, trong các hệ thống thông tin, chỉ có một khoảng tần số xác định được ấn định cho việc truyền dẫn thông tin. Do đó, khi thực hiện thông tin liên lạc, nếu tần số của tín hiệu thông tin không nằm trong khoảng cho phép này hoặc không phù hợp với việc

truyền dẫn trên kênh thông tin thì tín hiệu phù hợp cho truyền dẫn sẽ được thực hiện thay đổi theo thông tin cần truyền. Việc thay đổi này được gọi chung là việc điều chế. Phía phát sẽ thực hiện điều chế tín hiệu phát, ở phía thu sẽ thực hiện việc ngược lại, bóc tách để khôi phục tín hiệu ban đầu thông qua quá trình giải điều chế.

Bảng liệt kê các hàm thư viện cơ bản hỗ trợ hệ thống thông tin số:

Hàm	Ý nghĩa	Ví dụ
pskmod	Thực hiện điều chế dịch khóa pha	$y = \text{pskmod}(x, M)$
pskdemod	Thực hiện giải điều chế dịch khóa pha	$y = \text{pskdemod}(r, M)$
fskmod	Thực hiện điều chế dịch khóa tần	$y = \text{fskmod}(x, M, \text{freq\_sep}, \text{nsamp})$
fskdemod	Thực hiện giải điều chế dịch khóa tần	$y = \text{fskdemod}(x, M, \text{freq\_sep}, \text{nsamp})$
qammod	Thực hiện điều chế cầu phương biên độ	$y = \text{qammod}(x, M)$
qamdemod	Thực hiện giải điều chế cầu phương biên độ	$r = \text{qamdemod}(y, M)$

*Xem thêm tài liệu sử dụng bộ công cụ Hệ thống truyền thông*

**Ví dụ 4-48:** Biểu diễn tín hiệu điều chế số

Tạo một dãy tín hiệu tin tức ngẫu nhiên gồm 1000 ký hiệu. Thực hiện điều chế dịch khóa pha (PSK) với mức điều chế  $M = 8$  cho dãy tín hiệu tạo được. Giả sử tín hiệu điều chế bị tác động bởi nhiễu trắng. Minh họa đồ thị chòm sao của tín hiệu điều chế trước và sau khi chịu tác động của nhiễu.

```

N=1000;
M=8;
msg=randi([0 M-1],N,1);
Txpsk=pskmod(msg,M);
noise=(randn(N,1)+i*randn(N,1))*0.15;
Rxpsk=Txpsk+noise;
scatterplot(Txpsk);title('8-PSK, Không nhiễu');
scatterplot(Rxpsk);title('8-PSK, Có nhiễu');

```

Tạo ngẫu nhiên một dãy thông tin gồm 1000 ký hiệu. Thực hiện điều chế dịch khóa tần số (FSK) cho tín hiệu với mức điều chế  $M = 4$ , khoảng cách tần số bằng 8, số mẫu bằng 8, tần số lấy mẫu là 32Hz. Biểu diễn phổ của tín hiệu sau điều chế.

```

M=4;
freqsep=8;

```

```

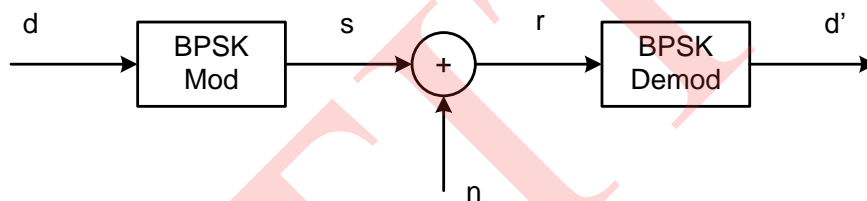
nsamp=8;
Fs=32;
msg=randi([0 M-1],1000,1);
y=fskmod(msg,M,freqsep,nsamp,Fs);
yLen=length(y);
freq=[-Fs/2:Fs/yLen:Fs/2-Fs/yLen];
Syy=10*log10(fftshift(abs(fft(y)))));
plot(freq,Syy);

```

#### 4.5.2.6. Đánh giá hệ thống số

Bằng việc kết hợp các hàm thư viện của MATLAB, chúng ta có thể dễ dàng triển khai các đoạn chương trình đánh giá chất lượng các hệ thống truyền thông số. Trong phần này chúng ta sẽ xem xét thông qua một ví dụ đánh giá một hệ thống thông tin số đơn giản.

**Ví dụ 4-49:** Đánh giá chất lượng (tỷ lệ lỗi bit BER) của hệ thống điều chế BPSK trên kênh AWGN



```

clear all;
N=1e+6;
rand('state',100);
randn('state',200);
% Chuẩn bị cho phía phát
EbN0dB=-3:10;
msg=rand(1,N)>0.5;
noise=(randn(1,N)+i*randn(1,N))/sqrt(2);
% BPSK modulation
txsym=2*msg-1;
% Lap mo phong kênh AWGN cho moi gia trị EbN0dB
nError=zeros(1,length(EbN0dB));
for ii=1:length(EbN0dB)
    % Tac dong nhieu tren kênh
    rxsym=txsym+10^(-EbN0dB(ii)/20)*noise;
    % BPSK demodulation
    rxmsg=real(rxsym)>0;
    % Dem so bit loi
    nError(ii)=size(find(msg-rxmsg),2);
end

```

```

end;
simBER=nError/N;
theoryBER=0.5*erfc(sqrt(10.^(EbN0dB/10)));
%
figure;
semilogy(EbN0dB,theoryBER,'b.-\ ',EbN0dB,simBER,'rs\ ');
axis([-3 10 1e-5 0.5]);
legend('Ket qua ly thuyet','Ket qua mo phong');
grid on;
xlabel('E_b/N_0 (dB)');
ylabel('Ty le loi bit');
title('Ty le loi bit cua he thong dieu che BPSK tren kenh AWGN');

```

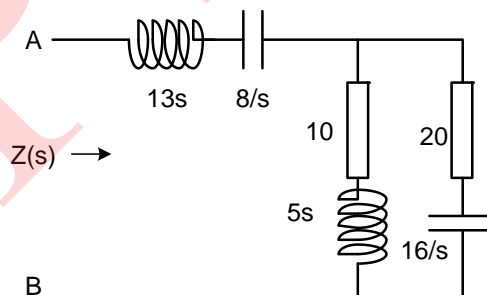
## 4.6. ỨNG DỤNG MATLAB TRONG KỸ THUẬT ĐIỆN TỬ

### 4.6.1. Giới thiệu chung

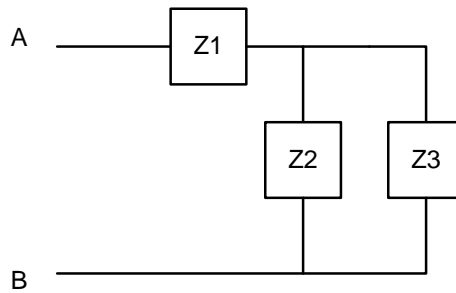
Mặc dù MATLAB không xây dựng một bộ công cụ riêng biệt phục vụ giải các bài toán thuộc lĩnh vực điện tử, nhưng bằng sự kết hợp các hàm thư viện của MATLAB và một số bộ công cụ phổ biến chúng ta có thể dễ dàng thực hiện việc giải các bài toán kỹ thuật điện tử. Trong phần này, chúng ta sẽ làm quen việc ứng dụng MATLAB vào giải các bài toán Kỹ thuật điện tử thông qua một số ví dụ minh họa.

### 4.6.2. Một số bài toán Kỹ thuật điện tử giải bằng MATLAB

**Ví dụ 4-50:** Xét một đoạn mạch như hình vẽ, các giá trị cho có đơn vị  $\Omega$ . Thực hiện việc tính trở kháng  $Z(s)$  và  $Y(s)$ .



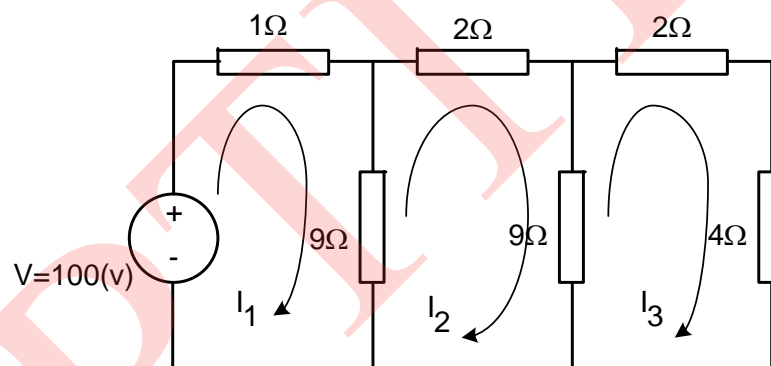
Việc thực hiện tính trở kháng  $Z(s)$  và  $Y(s)$  bằng tay được coi như một bài tập luyện tập cho người học. Chúng ta sẽ tập trung vào việc sử dụng MATLAB để hỗ trợ giải bài toán này. Trước hết, để thuận tiện chúng ta biểu diễn lại đoạn mạch đã cho như sau:



Từ sơ đồ biểu diễn trên ta thấy  $Z(s) = Z_{AB} = Z_1 \text{ nt } (Z_2 // Z_3)$  và  $Y(s) = 1/Z(s)$

```
syms s;
Z1=13*s+8/s;
Z2=10+5*s;
Z3=20+16/s;
Z=Z1+(Z2*Z3)/(Z2+Z3);
Zs=simplify(Z);
pretty(Zs)
```

**Ví dụ 4-51:** Xét một sơ đồ mạch điện như hình vẽ. Sử dụng phương pháp dòng điện vòng tìm giá trị dòng điện trên các vòng được biểu diễn trên sơ đồ.



Sử dụng phương pháp dòng điện vòng cho các vòng 1, 2, và 3 trong hình vẽ, ta có hệ phương trình sau:

$$\begin{cases} 10I_1 - 9I_2 = 100 \\ -9I_1 + 20I_2 - 9I_3 = 0 \\ -9I_2 + 15I_3 = 0 \end{cases}$$

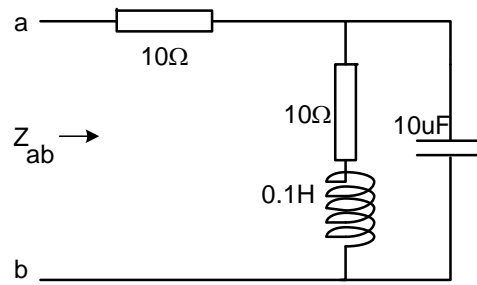
Hệ phương trình trên tương đương với hệ phương trình tuyến tính  $RI = V$  trong đó:

$$R = \begin{bmatrix} 10 & -9 & 0 \\ -9 & 20 & -9 \\ 0 & -9 & 15 \end{bmatrix}, V = \begin{bmatrix} 100 \\ 0 \\ 0 \end{bmatrix}, \text{ và } I = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix}$$

Do đó, dễ dàng có thể sử dụng MATLAB để giải hệ phương trình như sau:

```
R=[10 -9 0; -9 20 -9; 0 -9 15];
V=[100; 0; 0];
I=R\V
```

**Ví dụ 4-52:** Xét đoạn mạch điện như hình vẽ. Khảo sát giá trị trở kháng  $Z_{ab}$  theo tần số  $\omega$ .



Từ sơ đồ mạch ta có:  $Z_{ab} = Z_1 \text{ nt } (Z_2 // Z_3)$ , trong đó  $Z_1 = 10$ ,  $Z_2 = 10 + j0.1\omega$ , và  $Z_3 = \frac{1}{j10^{-5}\omega}$ .

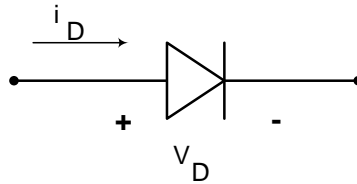
Giả sử chúng ta cần khảo sát trở kháng trong khoảng  $\omega = 0 \div 2000$ . Ta có thể giải bài toán với sự trợ giúp của MATLAB như sau:

```
omega = 0:1:2000;
z1 = 10;
z2 = 10 + j * 0.1 * omega;
z3 = 1./(j * 10^-5 * omega);
z = z1 + (z2.*z3)./(z2 + z3);
% Khảo sát phần thực theo omega: tọa độ Decarde
plot(omega, real(z));
xlabel('Tan so goc'); ylabel('Do lon phan thuc');
% Khảo sát phần ảo theo omega: tọa độ Decarde
plot(omega, imag(z));
% Minh hoa trong tọa độ cực
magrnd=round(abs(z));
theta=angle(z);
figure;
polar(theta,magrnd);
title('Tro khang tren toa do cuc');
```

**Ví dụ 4-53:** Xét một diode bán dẫn như hình vẽ. Dòng điện chạy qua diode được xác định bởi công thức

$$i_D = i_0(e^{\frac{qV_D}{kT}} - 1)$$

Trong đó  $V_D$  là điện áp đặt trên hai đầu của diode;  $i_0$  là dòng dò (giả sử bằng  $2 \times 10^{-4}A$ );  $k$  là hằng số Boltzman;  $q$  là giá trị điện tích của một điện tử;  $T$  là nhiệt độ làm việc; và  $i_D$  là dòng điện chạy qua diode.



Sử dụng MATLAB, thực hiện tính

- Các giá trị dòng điện chạy qua Diode với các giá trị điện áp  $V_D$  thay đổi từ  $-0.2V$  đến  $+0.25V$  với bước thay đổi bằng  $0.01V$
- Tương tự câu a) nhưng với các giá trị nhiệt độ bằng  $70^\circ K$ ,  $75^\circ K$ , và  $80^\circ K$
- Vẽ minh họa kết quả của câu b)

Bài toán này có thể giải với sự hỗ trợ của MATLAB như sau:

```
i0=2e-4;
k=1.38e-23;
q=1.602e-19;
vd=-0.2:0.01:0.25;
Tf=[70 75 80];
for ii=1:length(Tf)
    % Với mỗi giá trị T, tính với Vd thay đổi
    id=i0*(exp((q*vd)/(k*Tf(ii)))-1);
    % Minh họa với các kiểu đường khác nhau
    if ii==1
        plot(vd,id,'-o');
        hold on;
    elseif ii==2
        plot(vd,id,'-');
    else
        plot(vd,id,':o');
        hold off;
    end;
end;
legend('70^o K','75^o K','80^o K');
grid on;
title('Dong I_D vs. V_D');
xlabel('V_D'); ylabel('I_D');
```

## 4.7. BÀI TẬP CHƯƠNG 4

### 4.7.1. Bài tập: Ứng dụng MATLAB trong giải toán

**Bài 4-1.** Cho phương trình  $2x^3 - 6x + 1 = 0$ .

a) Thực hiện vẽ minh họa đồ thị trên đoạn  $[-2,2]$  để thấy tính đúng đắn của chứng minh “Phương trình có 3 nghiệm trên đoạn  $[-2,2]$ )

b) Tìm các nghiệm của phương trình

**Bài 4-2.** Sử dụng MATLAB vẽ minh họa để thấy rằng các phương trình sau có nghiệm

a)  $2x^2 - 5x^3 - 2x - 1 = 0$

b)  $2^x + 3^x = 6^x$

**Bài 4-3.** Sử dụng MATLAB vẽ minh họa tính đúng đắn của bất đẳng thức sau

$$\frac{1}{2} - \frac{x}{8} < \frac{1}{x} - \frac{1}{e^x - 1} < \frac{1}{2} \text{ với } \forall x \in [0; +\infty)$$

**Bài 4-4.** Cho hệ phương trình tuyến tính 
$$\begin{cases} x_1 + 0.306x_3 = 101.48 \\ x_2 + 0.702x_3 = 225.78 \\ -2x_1 + x_2 = 0 \end{cases}$$

Thực hiện giải hệ phương trình đã cho bằng các phương pháp khác nhau trên MATLAB.

**Bài 4-5.** Cho hệ phương trình tuyến tính

$$\begin{cases} 2x + 2y - z = 1 \\ x - 5y + z = 2 \\ 3x + 4y - 3z = 9 \end{cases}$$

Thực hiện giải hệ phương trình đã cho bằng các phương pháp khác nhau trên MATLAB

**Bài 4-6.** Tính giá trị của các tích phân

a)  $I_1 = \int_0^1 \frac{1}{(1+x)^2} dx$

b)  $I_2 = \int_0^1 \frac{1}{4x^2 + 4x + 5} dx$

c)  $I_3 = \int_0^1 x e^{-x} dx$

**Bài 4-7.** Tìm các tích phân

a)  $I_1 = \int \frac{1}{x^5 - x^2} dx$

b)  $I_2 = \int \frac{1}{(x-2)(x^2+1)^2} dx$

c)  $I_3 = \int \frac{x}{x^4 - 1} dx$

**Bài 4-8.** Tính diện tích hình phẳng giới hạn bởi đường cong  $y = x^2 - x + 3$  và đường thẳng  $y = 2x + 1$

**Bài 4-9.** Giải hệ phương trình vi phân



$$\begin{cases} y' = 5y - 2x \\ y' = 7y - 4x \end{cases}$$

với các điều kiện đầu  $y_1(0) = 0 = y_2(0)$

**Bài 4-10.** Giải phương trình vi phân  $y'' = xy$

**Bài 4-11.** Cho  $x, y, z$  là các số thực không âm thỏa mãn  $x^2 + y^2 + z^2 = 2$ . Thực hiện vẽ đồ thị và tìm cực tiểu của biểu thức

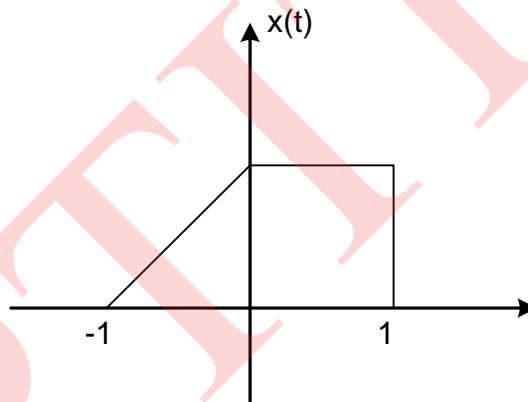
$$P = \frac{x^2}{x^2 + yz + x + 1} + \frac{y + z}{x + y + z + 1} - \frac{1 + yz}{9}$$

#### 4.7.2. Bài tập: Ứng dụng MATLAB trong Xử lý tín hiệu số

**Bài 4-12.** Một tín hiệu tuần hoàn  $x(t) = \Lambda(t)$  ( $|t| \leq 4$ ) có chu kỳ  $T = 8$ .

- Biểu diễn tín hiệu đã cho
- Tính và vẽ minh họa phổ pha và biên độ của tín hiệu đã cho

**Bài 4-13.** Cho một tín hiệu được mô tả bởi đồ thị như hình vẽ



- Biểu diễn lại tín hiệu đã cho trong MATLAB
- Tìm và vẽ minh họa phổ biên độ và pha của tín hiệu đã cho.

**Bài 4-14.** Sử dụng MATLAB, xây dựng các hàm thư viện biểu diễn các dãy tín hiệu cơ bản đã học: Xung Dirac, Hàm bước nhảy đơn vị, Dãy mũ, Hàm chữ nhật, Hàm tam giác, Dãy sin, ...

**Bài 4-15.** Một hệ thống rời rạc tuyến tính được mô tả bằng phương trình sai phân

$$y(n) + 1.5y(n-1) + 2y(n-2) = x(n) + 0.5x(n-1)$$

- Tìm hàm truyền đạt của hệ thống
- Biểu diễn các điểm không, điểm cực (nếu có) của hàm truyền đạt hệ thống trên mặt phẳng phức
- Tìm đáp ứng ra khi kích thích vào là xung bước nhảy đơn vị

**Bài 4-16.** Một hệ thống rời rạc tuyến tính được mô tả bằng phương trình sai phân

$$y(n) - 0.5y(n-1) + 0.125y(n-2) = x(n) + x(n-1)$$

- a) Tìm hàm truyền đạt của hệ thống
- b) Tìm đáp ứng xung của hệ thống
- c) Tìm đáp ứng ra của hệ thống khi kích thích vào là hàm bước nhảy đơn vị

**Bài 4-17.** Sử dụng hàm thư viện *fourier()* tìm biến đổi Fourier của các hàm tín hiệu sau

- a)  $y(t) = te^{-t^2}$
- b)  $y(t) = -e^t u(t) + 3\delta(t)$
- c)  $y(t) = u(t)$

**Bài 4-18.** Hệ thống LTI được mô tả bởi phương trình sai phân sau:

$$y(n) = 0.8y(n-1) + x(n)$$

- a) Xác định  $H(e^{j\omega})$
- b) Tính và vẽ minh họa đáp ứng ra ở trạng thái ổn định ứng với kích thích vào

$$x(n) = \cos(0.05\pi n) u(n)$$

#### 4.7.3. Bài tập: Ứng dụng MATLAB trong Lý thuyết điều khiển

**Bài 4-19.** Khảo sát các đặc tính của các khâu động học cơ bản:

- a) Khâu tỷ lệ  $H(s) = k$  với các hệ số  $k=-2, k=2$
- b) Khâu tích phân  $H(s) = \frac{k}{s}$  với  $k = 1; k = 5$
- c) Khâu quán tính  $H(s) = \frac{k}{1+Ts}$  với  $k = 2, T = 10$
- d) Khâu vi phân thực tế  $H(s) = \frac{kTs}{Ts+1}$  với  $k = 20, T = 0.1$
- e) Khâu bậc hai  $H(s) = \frac{k}{T^2s^2+2dT s+1}$  với  $k = 20, T = 10, d = 0:0.25:1$
- f) Khâu tích phân bậc hai  $H(s) = \frac{4}{s^2(1+10s)}$

**Bài 4-20.** Một hệ thống điều khiển có phương trình hàm truyền đạt:

$$H(s) = \frac{2}{s^2 + 2s + 4}$$

- a) Tìm các điểm không, điểm cực (nếu có) của hệ thống. Biểu diễn các điểm không và điểm cực trên mặt phẳng phức.
- b) Tìm đáp ứng ra của hệ thống và vẽ minh họa khi kích thích vào là: (1) Hàm bước nhảy, (2) Hàm Dirac. Giả sử chỉ quan sát 10s đầu tiên.
- c) Khảo sát và vẽ minh họa đáp ứng tần số của hệ thống.

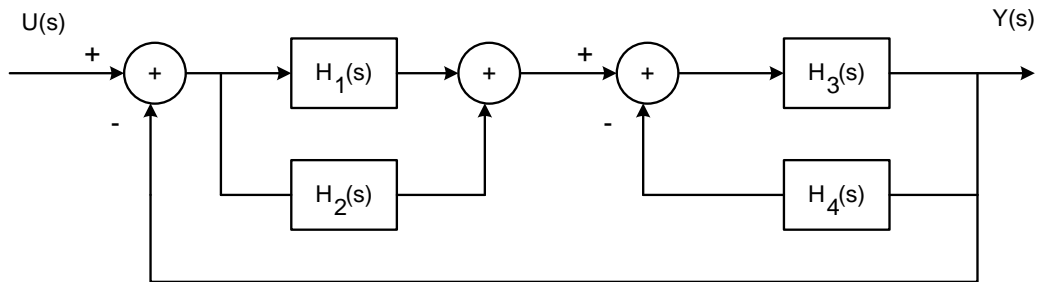
**Bài 4-21.** Xét tính ổn định của các hệ thống sau

- a) Hệ thống vòng kín có hàm truyền đạt vòng  $H(s) = -\frac{5}{(1-10s)(1+0.1s)}$ , hồi tiếp với hệ số  $-1$

b) Hệ thống vòng kín có hàm truyền đạt vòng  $H(s) = \frac{1-s}{(1+s)(1+0.5s)}$

**Bài 4-22.** Cho hệ thống điều khiển có sơ đồ như hình vẽ, trong đó:

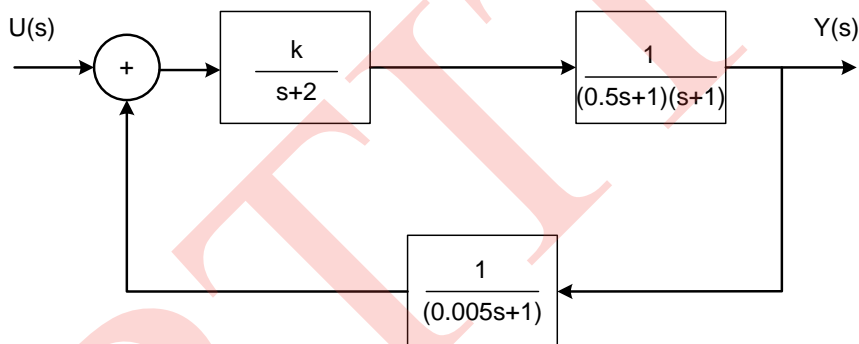
$$H_1(s) = \frac{s+1}{(s+3)(s+5)}, H_2(s) = \frac{s}{s^2+2s+8}, H_3(s) = \frac{1}{s}, \text{ và } H_4(s) = \frac{1}{s+2}.$$



a) Tính hàm truyền đạt của hệ thống

b) Thực hiện khảo sát các đặc tính hệ thống

**Bài 4-23.** Cho hệ thống điều khiển vòng kín có sơ đồ như hình vẽ, trong đó giá trị  $k$  có thể là 8; 17.564411; 20

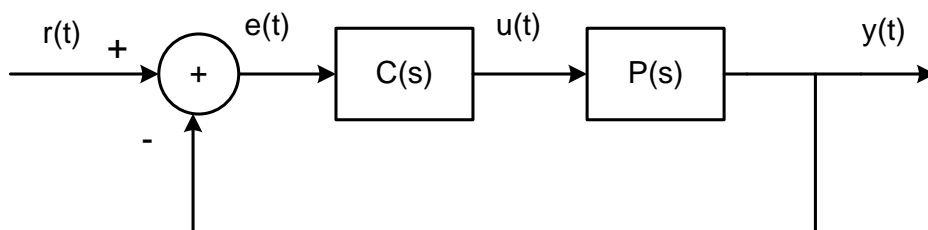


a) Xác định hàm truyền đạt của hệ thống

b) Khảo sát các đặc tính của hệ thống vòng kín trong miền thời gian

c) Khảo sát các đặc tính trong miền tần số Nyquist và đồ thị Bode của hệ thống hở

**Bài 4-24.** Xét một hệ thống điều khiển có sơ đồ như hình vẽ, trong đó quá trình cần điều khiển có hàm truyền đạt  $P(s) = \frac{1}{(1+10s)(1+s)(1+0.2s)}$



Thực hiện thiết kế khối điều khiển có hàm truyền đạt  $C(s)$ , với các trường hợp:

a) Khối điều khiển là bộ điều khiển loại P (khâu tỷ lệ)

b) Khối điều khiển là bộ điều khiển loại PI (khâu tỷ lệ - tích phân)

c) Khối điều khiển là bộ điều khiển loại PD (khâu tỷ lệ - vi phân)

**Bài 4-25.** Hệ thống bậc nhất với trễ thời gian được mô tả bởi phương trình vi phân sau:  $\dot{x} = \frac{1}{T}(-x + Ku(t - \tau))$ , trong đó tín hiệu vào  $u(t)$  và tín hiệu ra là  $x(t)$ .

a) Thực hiện xây dựng sơ đồ hệ thống trên Simulink

b) Khảo sát các tính chất của hệ thống trên Simulink với  $K = 1, T = 2, \tau = 3$

**Bài 4-26.** Một mạng bậc 4 được mô tả bởi phương trình vi phân sau:

$$\frac{d^4 y}{dt^4} + a_3 \frac{d^3 y}{dt^3} + a_2 \frac{d^2 y}{dt^2} + a_1 \frac{dy}{dt} + a_0 y(t) = u(t)$$

trong đó,  $a_3 = -2, a_2 = 1$ , và  $a_1 = a_0 = 1$ . Điều kiện đầu  $y(0) = y'(0) = y''(0) = y'''(0) = 0$

a) Xây dựng sơ đồ mô phỏng mạng trong simulink

b) Thực hiện mô phỏng quan sát sự biến đổi của tín hiệu kích thích và các mức tín hiệu ra (sau mỗi khâu tích phân).

#### 4.7.4. Bài tập: Ứng dụng MATLAB trong Truyền thông

**Bài 4-27.** Một hệ thống phát thanh sử dụng phương pháp điều chế song biên (DSB-AM), trong đó tín hiệu mang tin tức được mô tả bởi hàm  $m(t)$  như sau:

$$m(t) = \begin{cases} 1 & 0 \leq t \leq t_0/3 \\ -2 & t_0/3 < t \leq 2t_0/3 \\ 0 & \text{khoảng còn lại} \end{cases}$$

Biết sóng mang  $c(t) = \cos(2\pi f_c t)$ ,  $t_0 = 0.15s$ , và  $f_c = 250Hz$ .

a) Thực hiện biểu diễn minh họa tín hiệu điều chế  $u(t)$

b) Tính và biểu diễn minh họa phổ bên độ và pha của tín hiệu mang tin tức  $m(t)$  và tín hiệu điều chế  $u(t)$

**Bài 4-28.** Một hệ thống phát thanh sử dụng phương pháp điều chế tần số, trong đó tín hiệu mang tin tức được mô tả như sau:

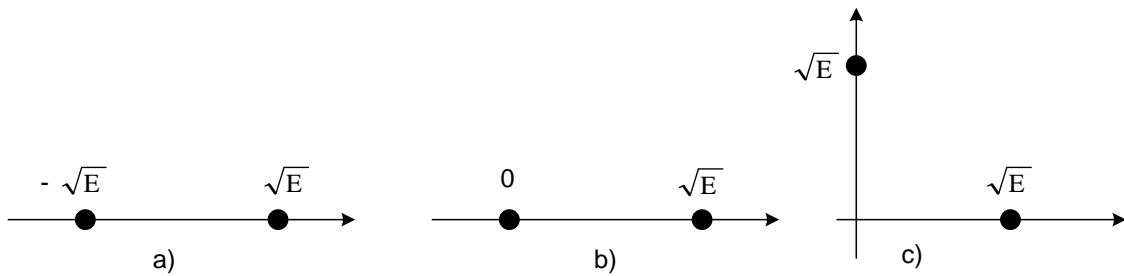
$$m(t) = \begin{cases} 1 & 0 \leq t \leq t_0/3 \\ -2 & t_0/3 < t \leq 2t_0/3 \\ 0 & \text{khoảng còn lại} \end{cases}$$

Biết sóng mang  $c(t) = \cos(2\pi f_c t)$ ,  $f_c = 200Hz$ ,  $t_0 = 0.15s$ , hằng số điều tần  $k_f = 50$

a) Vẽ minh họa tín hiệu điều chế  $u(t)$

b) Xác định và vẽ minh họa phổ của tín hiệu mang tin tức  $m(t)$  và tín hiệu điều chế  $u(t)$

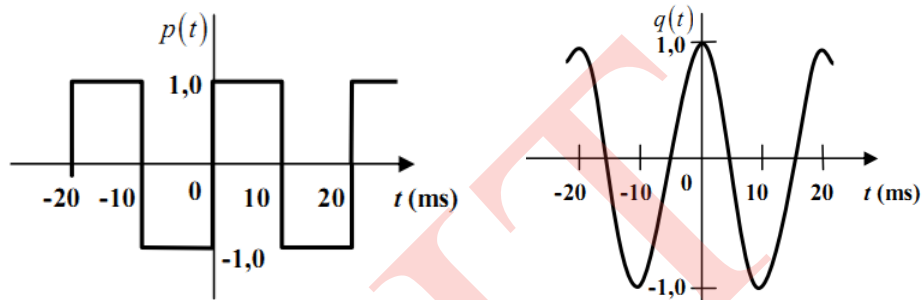
**Bài 4-29.** Một hệ thống truyền thông số thực hiện việc điều chế đơn giản với các đồ hình chòm sao tín hiệu như sau.



a) Sử dụng MATLAB/Simulink thực hiện mô phỏng theo phương thức Monte Carlo so sánh chất lượng hoạt động của các phương thức điều chế đã cho (theo tỷ lệ lỗi bit BER)

b) Sử dụng kiến thức đã biết, giải thích kết quả thu được trong câu a)

**Bài 4-30.** Cho tín hiệu  $p(t)$  và  $q(t)$  được mô tả bởi đồ thị như hình vẽ:

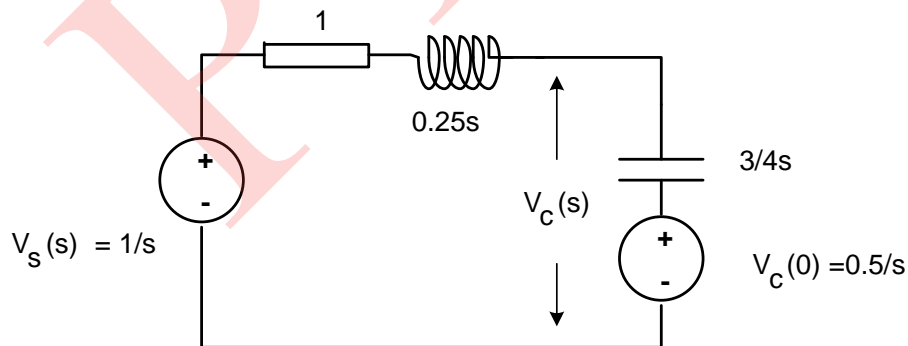


a) Biểu diễn minh họa lại các tín hiệu trong MATLAB

b) Tính và minh họa tương quan chéo của các tín hiệu đã cho

#### 4.7.5. Bài tập: Ứng dụng MATLAB trong Kỹ thuật điện tử

**Bài 4-31.** Cho sơ đồ mạch điện như hình vẽ

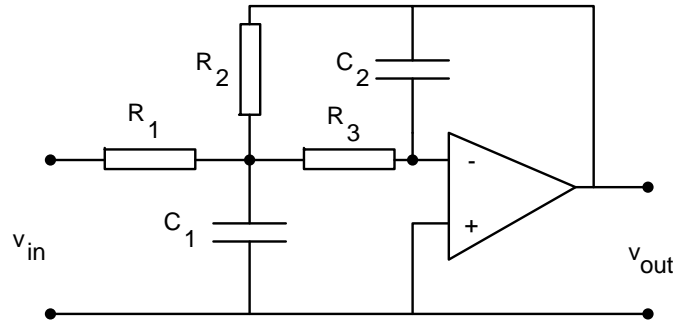


a) Xây dựng biểu thức  $V_C(s)$

b) Sử dụng hàm thư viện chuyển đổi Laplace ngược, tìm biểu thức điện áp trên tụ  $C$  trong miền thời gian  $v_C(t)$

c) Vẽ minh họa biểu thức  $v_C(t)$  trong 10s đầu tiên

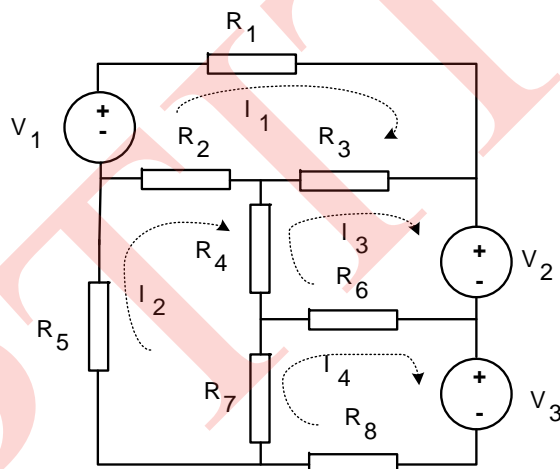
**Bài 4-32.** Cho sơ đồ mạch điện như hình vẽ



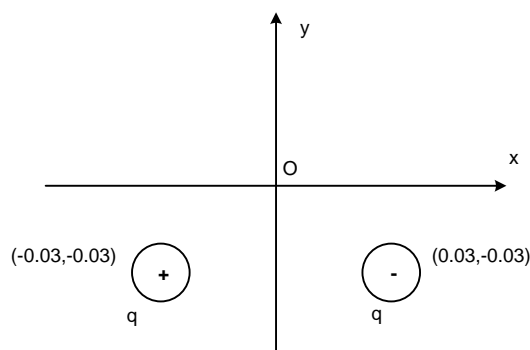
Biết  $R_1 = 0.2M$ ,  $R_2 = 40k$ ,  $R_3 = 50k$ ,  $C_1 = 25nF$ ,  $C_2 = 10nF$

- Tìm biểu thức hàm truyền đạt của mạch đã cho
- Khảo sát hàm truyền đạt của hệ thống theo tần số góc  $\omega$

**Bài 4-33.** Cho sơ đồ như hình vẽ. Biết  $V_1 = 22V$ ,  $V_2 = 12V$ ,  $V_3 = 44V$ ,  $R_1 = 20\Omega$ ,  $R_2 = 12\Omega$ ,  $R_3 = 15\Omega$ ,  $R_4 = 7\Omega$ ,  $R_5 = 16\Omega$ ,  $R_6 = 10\Omega$ ,  $R_7 = 10\Omega$ ,  $R_8 = 15\Omega$ . Xác định các dòng điện trên các điện trở.

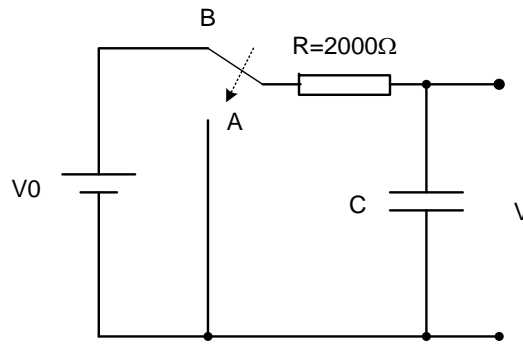


**Bài 4-33.** Một cặp điện tích trái dấu có độ lớn điện tích là  $q = 12 \times 10^{-19}C$  như hình vẽ. Sử dụng MATLAB tính cường độ điện trường và vẽ biên độ cường độ điện dọc theo trục  $x$  trong đoạn  $[-8,8]$



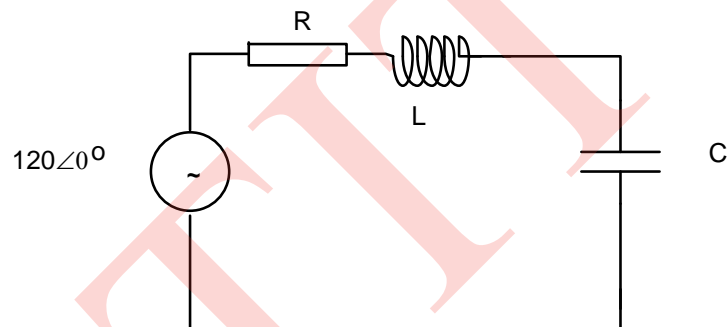
**Bài 4-34.** Trong quá trình thí nghiệm môn cơ sở cấu kiện điện tử, sinh viên được yêu cầu xác định giá trị của một tụ điện thông qua một thí nghiệm như sau. Sơ đồ mạch thí nghiệm như hình vẽ. Ban đầu chuyển mạch ở vị trí B, tụ được nạp đầy từ nguồn, sau đó

chuyển mạch chuyển sang vị trí A đồng thời giá trị điện áp  $V$  được đo trong vòng 10s, mỗi lần đọc kết quả cách nhau 1s. Hãy xác định giá trị của điện áp  $V_0$  và độ lớn của tụ điện  $C$ .



t(s)	1	2	3	4	5	6	7	8	9	10
V(V)	9.5	7.35	5.25	3.65	2.85	2.05	1.25	0.95	0.75	0.61

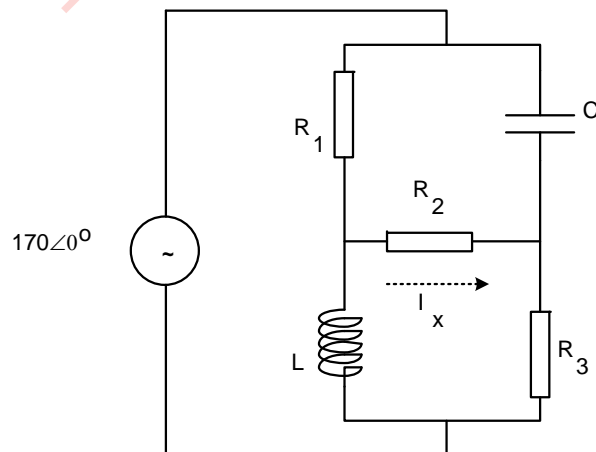
**Bài 4-35.** Một mạch RLC nối tiếp như hình vẽ, trong đó nguồn cung cấp là nguồn xoay chiều  $120\angle 0^\circ$  (V),  $R = 120\Omega$ ,  $L = 0.15mH$ ,  $C = 0.26nF$ .



Sử dụng MATLAB tính và vẽ:

- Biên độ dòng điện trong mạch khi tần số thay đổi từ  $100kHz \div 10MHz$ .
- Pha dòng điện trong mạch khi tần số thay đổi từ  $100kHz \div 10MHz$ .

**Bài 4-36.** Cho sơ đồ mạch điện như hình vẽ, trong đó:  $R_1 = 85\Omega$ ,  $R_2 = 100\Omega$ ,  $R_3 = 50\Omega$ ,  $Z_C = -j100\Omega$ ,  $Z_L = j200\Omega$ .



Sử dụng phương pháp điện áp nút tìm dòng điện  $I_x$

# PHỤ LỤC

## 1. Danh sách một số hàm trong Signal Processing Toolbox

### a) Các hàm về dạng tín hiệu

+ Các hàm tạo tín hiệu

<b>buffer</b>	Đệm vectơ tín hiệu vào ma trận khung dữ liệu
<b>chirp</b>	Tạo tín hiệu quét tần số
<b>demod</b>	Giải điều chế
<b>diric</b>	Hàm Dirichlet hay hàm sinc tuần hoàn
<b>gauspuls</b>	Tạo xung Gaussian điều chế (RF)
<b>gmonopuls</b>	Xung đơn Gaussian
<b>modulate</b>	Điều chế
<b>pulstran</b>	Chuỗi xung
<b>rectpuls</b>	Tạo mẫu xung chữ nhật không chu kỳ
<b>sawtooth</b>	Tạo xung răng cưa hoặc xung tam giác
<b>seqperiod</b>	Tính chu kỳ tín hiệu
<b>sinc</b>	Hàm Sinc
<b>sptool</b>	Mở công cụ xử lý tín hiệu số
<b>square</b>	Tạo xung vuông
<b>tripuls</b>	Tạo mẫu xung tam giác không có chu kỳ
<b>vco</b>	Bộ dao động điều khiển bằng điện áp (VCO)
<b>shiftdata</b>	Dịch dữ liệu
<b>unshiftdata</b>	Dịch dữ liệu ngược
<b>strips</b>	Vẽ đồ thị theo dải
<b>udecode</b>	Giải mã các số lượng tử 2n mức ở đầu vào thành các điểm lấy mẫu đầu ra.
<b>uencode</b>	Lượng tử hóa và mã hóa các điểm lấy mẫu ở đầu vào thành các số nguyên đầu ra.

+ Các hàm đo đặc tính tín hiệu

<b>midcross</b>	Giá trị tham chiếu ở giữa của dạng sóng hai mức
<b>pulseperiod</b>	Chu kỳ của xung 2 mức (xung vuông)
<b>pulsesep</b>	Phân tách giữa các xung 2 mức
<b>pulsewidth</b>	Độ rộng xung hai mức
<b>settlingtime</b>	Đặt thời gian cho dạng sóng 2 mức



<b>statelevels</b>	Đánh giá mức trạng thái của dạng sóng hai mức bằng phương pháp biểu đồ
<b>falltime</b>	Sườn âm (xuống) của dạng sóng 2 mức
<b>risetime</b>	Sườn dương (lên) của dạng sóng 2 mức
<b>slewrate</b>	Tốc độ quay của dạng sóng 2 mức
<b>overshoot</b>	Số liệu overshoot của dạng sóng 2 mức
<b>undershoot</b>	Số liệu undershoot của dạng sóng 2 mức
<b>dutycycle</b>	Hệ số lấp đầy
<b>peak2peak</b>	Giá trị đỉnh – đỉnh
<b>peak2rms</b>	Chuyển đổi giá trị đỉnh sang hiệu dụng
<b>rms</b>	Giá trị hiệu dụng
<b>rssq</b>	Tổng bình phương
<b>bandpower</b>	Công suất băng tần
<b>enbw</b>	Băng tần nhiễu tương đương
<b>sfdR</b>	Dải động sai lệch tự do
<b>sinad</b>	Tỷ số tín hiệu trên nhiễu và suy hao
<b>snr</b>	Tỷ số tín hiệu trên nhiễu
<b>thd</b>	Méo tuần hoàn tổng
<b>toi</b>	Điểm chặn thứ ba
<b>marcumq</b>	Hàm Marcum Q tổng quát

***b) Các hàm chập và tương quan***

<b>cconv</b>	Chập vòng modulo-N
<b>conv</b>	Tích chập và nhân đa thức
<b>convmtx</b>	Ma trận chập
<b>corrmtx</b>	Ma trận dự liệu để đánh giá ma trận tự tương quan
<b>xcorr</b>	Tương quan chéo
<b>xcorr2</b>	Tương quan chéo hai chiều
<b>xcov</b>	Hiệp phương sai chéo

***c) Các phép biến đổi***

<b>bitrevorder</b>	Hoán vị dữ liệu theo thứ tự đảo ngược
<b>cceps</b>	Phân tích phổ phức
<b>czt</b>	Biến đổi Z
<b>dct</b>	Biến đổi cosine rời rạc (DCT)

<b>digitrevorder</b>	Hoán vị đầu vào thành các số có thứ tự đảo ngược
<b>dftmtx</b>	Ma trận biến đổi Fourier rời rạc
<b>fft</b>	Biến đổi Fourier nhanh
<b>fwht</b>	Biến đổi Walsh–Hadamard nhanh
<b>goertzel</b>	Biến đổi Fourier rời rạc với thuật toán Goertzel bậc 2
<b>hilbert</b>	Phân tích tín hiệu thời gian rời rạc bằng biến đổi Hilbert
<b>icceps</b>	Phổ phức nghịch đảo
<b>idct</b>	Biến đổi cosine rời rạc ngược
<b>ifwht</b>	Biến đổi Walsh–Hadamard nhanh ngược
<b>rceps</b>	Tái tạo phổ thực và pha tối thiểu

#### ***d) Các bộ lọc tương tự và bộ lọc số***

##### ***+ Các bộ lọc tương tự***

<b>bilinear</b>	Biến đổi song tuyến bộ lọc tương tự - số
<b>besselap</b>	Bộ lọc thông thấp Bessel nguyên mẫu
<b>besself</b>	Thiết kế bộ lọc tương tự Bessel
<b>buttap</b>	Bộ lọc Butterworth nguyên mẫu
<b>butter</b>	Thiết kế bộ lọc Butterworth
<b>cheb1ap</b>	Bộ lọc tương tự thông thấp Chebyshev loại I nguyên mẫu
<b>cheb2ap</b>	Bộ lọc tương tự thông thấp Chebyshev loại II nguyên mẫu
<b>cheby1</b>	Thiết kế bộ lọc Chebyshev loại I (dải thông gọn)
<b>cheby2</b>	Thiết kế bộ lọc Chebyshev loại II (dải chắn gọn)
<b>ellip</b>	Thiết kế bộ lọc Elliptic
<b>ellipap</b>	Bộ lọc tương tự thông thấp Elliptic nguyên mẫu
<b>freqs</b>	Đáp ứng tần số của các bộ lọc tương tự
<b>lp2bp</b>	Biến đổi bộ lọc tương tự thông thấp sang bộ lọc thông dải
<b>lp2bs</b>	Biến đổi bộ lọc tương tự thông thấp sang bộ lọc chặn dải
<b>lp2hp</b>	Biến đổi bộ lọc tương tự thông thấp sang bộ lọc thông cao
<b>lp2lp</b>	Thay đổi tần số cắt cho bộ lọc tương tự thông thấp

##### ***+ Thiết kế bộ lọc số***

<b>designfilt</b>	Thiết kế các bộ lọc số
<b>digitalFilter</b>	Bộ lọc số
<b>butter</b>	Thiết kế bộ lọc Butterworth
<b>buttord</b>	Tần số cắt và bậc của bộ lọc Butterworth

<b>cheby1</b>	Thiết kế bộ lọc Chebyshev loại I (dài thông gọn)
<b>cheb1ord</b>	Bậc của bộ lọc Chebyshev loại I
<b>cheby2</b>	Thiết kế bộ lọc Chebyshev loại II (dài chặn gọn)
<b>cheb2ord</b>	Bậc của bộ lọc Chebyshev loại II
<b>ellip</b>	Thiết kế bộ lọc Elliptic
<b>ellipord</b>	Các bộ lọc elliptic bậc nhỏ nhất
<b>polyscale</b>	Sắp xếp nghiệm của đa thức
<b>polystab</b>	Ổn định đa thức
<b>impinvar</b>	Phương pháp xung bất biến để biến đổi bộ lọc tương tự sang bộ lọc số
<b>yulewalk</b>	Thiết kế bộ lọc số đệ quy
<b>cfirpm</b>	Thiết kế bộ lọc FIR phức và pha phi tuyến
<b>fir1</b>	Thiết kế bộ lọc đáp ứng xung hữu hạn theo phương pháp cửa sổ
<b>fir2</b>	Thiết kế bộ lọc đáp ứng xung hữu hạn theo phương pháp lấy mẫu tần số
<b>fircls</b>	Thiết kế bộ lọc FIR nhiều băng, bình phương tối thiểu
<b>fircls1</b>	Thiết kế bộ lọc FIR, pha tuyến tính, thông thấp và thông cao, bình phương tối thiểu
<b>firls</b>	Thiết kế bộ lọc FIR pha tuyến tính
<b>firpm</b>	Thiết kế bộ lọc FIR Parks-McClellan tối ưu
<b>firpmord</b>	Đánh giá bậc của bộ lọc FIR Parks-McClellan tối ưu
<b>gaussdesign</b>	Thiết kế bộ lọc Gaussian FIR dạng xung
<b>intfilt</b>	Thiết kế bộ lọc FIR nội suy
<b>kaiserord</b>	Các thông số đánh giá thiết kế bộ lọc FIR cửa sổ Kaiser
<b>maxflat</b>	Chuẩn hóa thiết kế bộ lọc số Butterworth
<b>rcosdesign</b>	Thiết kế bộ lọc dạng xung FIR cosine nâng
<b>sgolay</b>	Thiết kế bộ lọc Savitzky-Golay
<b>fdatool</b>	Mở file thiết kế và công cụ phân tích
<b>fvtool</b>	Mở công cụ bộ lọc ảo
<b>sptool</b>	Mở công cụ xử lý tín hiệu tương tác

+ *Phân tích các bộ lọc số*

<b>angle</b>	Góc pha
<b>fdatool</b>	Mở file thiết kế và công cụ phân tích
<b>filternorm</b>	2-norm or infinity-norm of digital filter

<b>filtord</b>	Bậc của bộ lọc
<b>firtype</b>	Kiểu bộ lọc FIR pha tuyến tính
<b>freqz</b>	Đáp ứng tần số của bộ lọc số
<b>fvtool</b>	Mở công cụ bộ lọc ảo
<b>grpdelay</b>	Trễ trung bình của bộ lọc (trễ nhóm)
<b>impz</b>	Đáp ứng xung của bộ lọc số
<b>impzlength</b>	Chiều dài đáp ứng xung
<b>info</b>	Thông tin về bộ lọc số
<b>phasedelay</b>	Trễ pha của bộ lọc số
<b>phasez</b>	Đáp ứng pha của bộ lọc số
<b>residuez</b>	z-transform partial-fraction expansion
<b>stepz</b>	Đáp ứng nhảy bậc của bộ lọc số
<b>zerophase</b>	Đáp ứng pha-điểm không của bộ lọc số
<b>zplane</b>	Vẽ điểm cực – điểm không
<b>isallpass</b>	Xác định liệu bộ lọc có phải là thông toàn bộ
<b>isfir</b>	Xác định liệu bộ lọc số có đáp ứng xung hữu hạn
<b>islinphase</b>	Xác định liệu bộ lọc có pha tuyến tính
<b>isminphase</b>	Xác định liệu bộ lọc có pha tối thiểu
<b>ismaxphase</b>	Xác định liệu bộ lọc có pha tối đa
<b>isstable</b>	Xác định liệu bộ lọc có ổn định

+ Xây dựng bộ lọc số

<b>cell2sos</b>	Chuyển đổi mảng bậc 2 thành ma trận
<b>dspfwiz</b>	Mở panel FDA Tool Realize Model để tạo mô phỏng bộ lọc
<b>latc2tf</b>	Chuyển đổi các thông số bộ lọc lưới để biến đổi dạng hàm
<b>tf2latc</b>	Chuyển đổi các thông số hàm truyền đạt của bộ lọc sang dạng bộ lọc lưới
<b>residuez</b>	z-transform partial-fraction expansion
<b>sos2cell</b>	Biến đổi ma trận các thành phần bậc 2 thành dạng mảng
<b>sos2ss</b>	Biến đổi các thông số thành phần bậc 2 của bộ lọc số sang dạng không gian trạng thái
<b>sos2tf</b>	Biến đổi dữ liệu các thành phần bậc 2 của bộ lọc số sang dạng hàm
<b>sos2zp</b>	Biến đổi dữ liệu các thông số thành phần bậc 2 của bộ lọc số sang dạng điểm không – điểm cực
<b>ss</b>	Biến đổi bộ lọc số sang dạng không gian trạng thái

<b>ss2sos</b>	Biến đổi các thông số từ dạng không gian trạng thái sang dạng thành phần bậc 2.
<b>ss2tf</b>	Biến đổi các thông số bộ lọc từ dạng không gian trạng thái sang dạng hàm truyền đạt
<b>ss2zp</b>	Biến đổi các thông số của bộ lọc từ dạng không gian trạng thái sang dạng điểm không – điểm cực
<b>tf</b>	Biến đổi bộ lọc số sang hàm truyền đạt
<b>tf2sos</b>	Biến đổi dữ liệu hàm truyền đạt của bộ lọc số sang dạng các thành phần bậc 2.
<b>tf2ss</b>	Biến đổi các thông số bộ lọc từ dạng hàm truyền đạt sang dạng không gian trạng thái
<b>tf2zp</b>	Biến đổi các thông số bộ lọc từ dạng hàm truyền đạt sang dạng điểm không – điểm cực.
<b>tf2zpk</b>	Biến đổi các thông số bộ lọc từ dạng hàm truyền đạt sang dạng điểm không – điểm cực – độ lợi.
<b>zp2sos</b>	Biến đổi các thông số bộ lọc từ dạng điểm không – điểm cực – độ lợi sang dạng các thành phần bậc hai.
<b>zp2ss</b>	Biến đổi các thông số bộ lọc từ dạng điểm không – điểm cực – độ lợi sang dạng không gian trạng thái.
<b>zp2tf</b>	Biến đổi các thông số bộ lọc từ dạng điểm không – điểm cực – độ lợi sang dạng hàm truyền đạt.
<b>zpk</b>	Biến đổi bộ lọc số thành dạng điểm không – điểm cực – độ lợi.
<b>fftfilt</b>	Bộ lọc FIR dựa trên FFT sử dụng phương pháp overlap-add
<b>filter</b>	Dữ liệu bộ lọc đệ quy (IIR) hoặc không đệ quy (FIR)
<b>filtfilt</b>	Bộ lọc số điểm không – pha
<b>filtic</b>	Các điều kiện đầu để xây dựng bộ lọc chuyển vị trực tiếp dạng II
<b>filt2block</b>	Tạo các khối bộ lọc trong simulink
<b>latcfilt</b>	Thiết lập bộ lọc lưới hoặc thang lưới
<b>medfilt1</b>	Lọc trung tuyến 1 chiều
<b>sgolayfilt</b>	Lọc Savitzky-Golay
<b>sosfilt</b>	Bộ lọc số IIR bậc 2

#### ***e) Phân tích phổ***

<b>bandpower</b>	Công suất băng tần
<b>cpsd</b>	Mật độ phổ công suất chéo
<b>db</b>	Chuyển đổi số đo năng lượng hoặc công suất sang decibel
<b>db2mag</b>	Chuyển đổi decibels (dB) sang độ lớn

<b>db2pow</b>	Chuyển đổi decibels (dB) sang công suất
<b>findpeaks</b>	Tìm giá trị đỉnh
<b>mag2db</b>	Chuyển đổi độ lớn sang decibels (dB)
<b>mscohere</b>	Bình phương biên độ
<b>pburg</b>	Đánh giá mật độ phổ công suất theo phương pháp Burg
<b>pcov</b>	Đánh giá mật độ phổ công suất theo phương pháp hiệp phương sai
<b>peig</b>	Phổ giả ngẫu nhiên dùng phương pháp vectơ riêng
<b>periodogram</b>	Đánh giá mật độ phổ công suất
<b>pmcov</b>	Đánh giá mật độ phổ công suất theo phương pháp hiệp phương sai có sửa đổi
<b>pmtm</b>	Đánh giá mật độ phổ công suất multitaper
<b>pmusic</b>	Phổ giả ngẫu nhiên dùng thuật toán MUSIC
<b>pow2db</b>	Chuyển đổi công suất sang decibels (dB)
<b>pwelch</b>	Đánh giá mật độ phổ công suất Welch
<b>rooteig</b>	Tần số và công suất sử dụng phương pháp vectơ riêng
<b>rootmusic</b>	Thuật toán MUSIC gốc
<b>sinad</b>	Tỷ số tín hiệu trên nhiễu và méo
<b>snr</b>	Tỷ số tín hiệu trên nhiễu
<b>spectrogram</b>	Phân tích phổ dùng biến đổi Fourier thời gian ngắn
<b>sptool</b>	Mở công cụ xử lý tín hiệu số
<b>tffestimate</b>	Đánh giá hàm truyền đạt
<b>pyulear</b>	Đánh giá mật độ phổ công suất — Y

+ *Phân tích phổ theo phương pháp cửa sổ*

<b>bartlett</b>	Cửa sổ Bartlett
<b>barthannwin</b>	Cửa sổ Bartlett-Hann sửa đổi
<b>blackmanharris</b>	Cửa sổ 4 thứ hạng cực tiểu Blackman-Harris
<b>blackman</b>	Cửa sổ Blackman
<b>bohmanwin</b>	Cửa sổ Bohman
<b>chebwin</b>	Cửa sổ Chebyshev
<b>enbw</b>	Băng tần nhiễu tương đương
<b>gausswin</b>	Cửa sổ Gaussian
<b>hamming</b>	Cửa sổ Hamming
<b>hann</b>	Cửa sổ Hann (Hanning)
<b>kaiser</b>	Cửa sổ Kaiser

<b>parzenwin</b>	Cửa sổ Parzen
<b>rectwin</b>	Cửa sổ chữ nhật
<b>taylorwin</b>	Cửa sổ Taylor
<b>triang</b>	Cửa sổ tam giác
<b>tukeywin</b>	Cửa sổ Tukey
<b>window</b>	Công chức năng cửa sổ
<b>wintool</b>	Mở công cụ phân tích và thiết kế cửa sổ
<b>wvtool</b>	Mở công cụ cửa sổ ảo

## 2. Danh sách một số hàm trong Symbolic Math Toolbox

### a) Tính toán với các biến

#### + Các biến chữ, các mô tả và hàm

<b>argnames</b>	Nhập các biến cho hàm tượng trưng
<b>children</b>	Mô tả phụ các số hạng của mô tả tượng trưng
<b>disp</b>	Hiển thị đầu vào
<b>display</b>	Hiển thị đầu vào
<b>formula</b>	Mô tả toán học định nghĩa hàm tượng trưng
<b>funtool</b>	Tính toán hàm
<b>pretty</b>	Điều chỉnh hiển thị
<b>sym</b>	Tạo các đối tượng tượng trưng
<b>symfun</b>	Tạo hàm tượng trưng
<b>syms</b>	Tạo nhanh các biến và hàm tượng trưng
<b>symvar</b>	Tìm các biến trong các mô tả tượng trưng, ma trận và hàm

#### + Các phép toán số học

<b>Arithmetic Operations</b>	Thực hiện các phép số học
<b>ceil</b>	Làm tròn ma trận về phía dương vô cùng
<b>conj</b>	Liên hợp phức của symbolic
<b>cumprod</b>	Tích tích lũy symbolic
<b>cumsum</b>	Tổng tích lũy symbolic
<b>fix</b>	Làm tròn về 0
<b>floor</b>	Làm tròn ma trận symbolic về phía dương vô cùng
<b>frac</b>	Tách các phần tử của ma trận về dạng phân số
<b>imag</b>	Phần ảo của số phức
<b>minus</b>	Phép trừ symbolic

<b>mod</b>	Phép modulus các phần tử của ma trận symbolic
<b>plus</b>	Phép cộng symbolic
<b>quorem</b>	Thương và phần dư
<b>real</b>	Phần thực của số phức
<b>round</b>	Làm tròn các phần tử của ma trận symbolic
<b>+ Các phép toán quan hệ</b>	
<b>eq</b>	Định nghĩa phương trình
<b>ge</b>	Định nghĩa quan hệ lớn hơn hoặc bằng
<b>gt</b>	Định nghĩa quan hệ lớn hơn
<b>le</b>	Định nghĩa quan hệ nhỏ hơn hoặc bằng
<b>lt</b>	Định nghĩa quan hệ nhỏ hơn
<b>ne</b>	Định nghĩa quan hệ không bằng
<b>isequaln</b>	Kiểm tra các đối tượng symbolic bằng nhau, và coi các giá trị NaN là bằng nhau
<b>+ Các phép logic</b>	
<b>and</b>	Phép logic AND
<b>not</b>	Phép logic NOT
<b>or</b>	Phép logic OR
<b>xor</b>	Phép logic XOR
<b>all</b>	Kiểm tra xem tất cả các phương trình và các bất đẳng thức của mảng symbolic có hợp lệ hay không
<b>any</b>	Kiểm tra xem có ít nhất một phương trình hoặc một bất đẳng thức của mảng symbolic có hợp lệ hay không
<b>isequaln</b>	Kiểm tra xem các đối tượng symbolic có bằng nhau, coi các giá trị NaN là bằng nhau
<b>isfinite</b>	Kiểm tra xem các phần tử của mảng symbolic có phải hữu hạn
<b>isinf</b>	Kiểm tra xem các phần tử của mảng symbolic có phải vô cùng
<b>isnan</b>	Kiểm tra xem các phần tử của mảng symbolic có phải NaN
<b>logical</b>	Kiểm tra tính hợp lệ của phương trình hoặc bất đẳng thức
<b>+ Chuyển đổi giữa biến và số</b>	
<b>char</b>	Chuyển đổi các đối tượng symbolic dạng dạng chuỗi ký tự
<b>double</b>	Chuyển đổi ma trận symbolic sang dạng số của MATLAB
<b>int8, int16, int32, int64</b>	Chuyển đổi ma trận symbolic sang dạng số nguyên ấn định
<b>poly2sym</b>	Chuyển véc tơ hệ số của đa thức sang dạng đa thức
<b>single</b>	Chuyển đổi ma trận symbolic sang dạng single
<b>sym</b>	Tạo các đối tượng symbolic



<b>sym2poly</b>	Chuyển đổi symbolic sang đa thức số
<b>symfun</b>	Tạo các hàm symbolic
<b>uint8, uint16, uint32, uint64</b>	Chuyển đổi ma trận symbolic sang số nguyên không âm định
<b>vpa</b>	Các phép toán với độ chính xác tùy chọn theo yêu cầu.

## b) Tính toán

### + Giải phương trình

#### – Các phương trình đại số và các hệ thống

<b>equationsToMatr ix</b>	Chuyển đổi tập các phương trình tuyến tính sang dạng ma trận
<b>finverse</b>	Đảo hàm
<b>linsolve</b>	Giải hệ phương trình tuyến tính biểu diễn theo dạng ma trận
<b>poles</b>	Các điểm cực của mô tả hoặc hàm
<b>solve</b>	Giải phương trình và hệ phương trình
<b>vpasolve</b>	Giải số

#### – Các phương trình vi phân và hệ thống

<b>dsolve</b>	Giải phương trình và hệ phương trình vi phân thuần nhất
<b>odeToVectoField</b>	Chuyển đổi các phương trình vi phân bậc cao thành hệ các phương trình vi phân bậc 1

### + Thao tác với công thức và rút gọn

#### – Rút gọn

<b>simplify</b>	Rút gọn đại số
<b>simplifyFraction</b>	Rút gọn phân số symbolic

#### – Sắp xếp công thức

<b>coeffs</b>	Các hệ số của đa thức
<b>collect</b>	Lựa chọn các hệ số
<b>combine</b>	Kết hợp các số hạng của cấu trúc đại số giống nhau
<b>compose</b>	Phân tách hàm
<b>expand</b>	Mở rộng đa thức symbolic và các hàm cơ bản
<b>factor</b>	Phân tách
<b>horner</b>	Biểu diễn đa thức theo kiểu lồng ghép
<b>numden</b>	Tử số và mẫu số
<b>rewrite</b>	Ghi lại các mô tả thành số hạng mới
<b>subexpr</b>	Ghi các mô tả symbolic thành dạng các số hạng mô tả phụ chung

– Thay thế	
<b>subs</b>	Thay thế symbolic
+ <i>Tính toán</i>	
– Vi phân	
<b>diff</b>	Tính vi phân của biến symbolic hoặc hàm
– Tích phân	
<b>int</b>	Tích phân xác định và bất định
<b>rsums</b>	Đánh giá tổng Riemann
– Phân tích véctor	
<b>curl</b>	Curl của trường véctor
<b>divergence</b>	Divergence của trường véctor
<b>gradient</b>	Véctor gradient của hàm vô hướng
<b>hessian</b>	Ma trận Hessian của hàm vô hướng
<b>jacobian</b>	Ma trận Jacobian
<b>laplacian</b>	Phép Laplacian của hàm vô hướng
<b>potential</b>	Thế của trường véctor
<b>véctorPotential</b>	Thế véctor của trường véctor
– Chuỗi số	
<b>cumprod</b>	Tích tích lũy
<b>cumsum</b>	Tổng tích lũy
<b>rsums</b>	Đánh giá tổng Riemann
<b>symprod</b>	Tích của các chuỗi
<b>symsum</b>	Tổng của các chuỗi
<b>taylor</b>	Mở rộng chuỗi Taylor
<b>taylortool</b>	Tính toán chuỗi Taylor
– Giới hạn	
<b>limit</b>	Tính toán giới hạn
– Biến đổi	
<b>fourier</b>	Biến đổi Fourier
<b>ifourier</b>	Biến đổi Fourier ngược
<b>ilaplace</b>	Biến đổi Laplace ngược
<b>iztrans</b>	Biến đổi Z ngược
<b>laplace</b>	Biến đổi Laplace
<b>ztrans</b>	Biến đổi Z

+ *Đại số tuyến tính*

– *Các phép tính ma trận và biến đổi*

<b>bernstein</b>	Các đa thức Bernstein
<b>bernsteinMatrix</b>	Ma trận Bernstein
<b>cat</b>	Ghép các mảng dài thành kích thước nào đó
<b>diag</b>	Tạo hoặc trích các đường chéo của ma trận
<b>expm</b>	Tính phép mũ ma trận
<b>horzcat</b>	Nối các ma trận theo chiều ngang (hàng)
<b>inv</b>	Lấy ma trận nghịch đảo
<b>pinv</b>	Lấy phép đảo Moore-Penrose của ma trận
<b>reshape</b>	Sắp xếp mảng
<b>size</b>	Tính kích thước ma trận
<b>sort</b>	Sắp xếp các phần tử trong một véc tơ hoặc 1 ma trận
<b>toeplitz</b>	Ma trận Toeplitz
<b>tril</b>	Lấy nửa dưới đường chéo chính của ma trận
<b>triu</b>	Lấy nửa trên đường chéo chính của ma trận
<b>vertcat</b>	Nối các ma trận theo chiều đứng (cột)

– *Phân tích ma trận*

<b>adjoint</b>	Ma trận liên hợp của ma trận vuông
<b>charpoly</b>	Đa thức riêng của ma trận
<b>cond</b>	Điều kiện số của ma trận
<b>curl</b>	Curl của trường véc tơ
<b>det</b>	Tính định thức của ma trận
<b>divergence</b>	Divergence (độ tản) của hàm véc tơ
<b>gradient</b>	Véc tơ gradient (biến thiên) của hàm vô hướng
<b>hessian</b>	Ma trận Hessian của hàm vô hướng
<b>jacobian</b>	Ma trận Jacobian
<b>laplacian</b>	Laplacian của hàm vô hướng
<b>minpoly</b>	Đa thức tối giản của ma trận
<b>norm</b>	Chuẩn của ma trận hoặc véc tơ
<b>potential</b>	Thế của trường véc tơ
<b>sqrtn</b>	Căn bậc hai của ma trận
<b>véc tơPotential</b>	Thế véc tơ của trường véc tơ

– *Không gian véc tơ và không gian con*

<b>colon</b>	(Dấu :) tạo các véc tơ, chỉ số mảng và vòng lặp
<b>colspace</b>	Không gian cột của ma trận

<b>null</b>	Tạo hệ cơ sở của không gian không của ma trận
<b>numel</b>	Tìm số phần tử của một mảng
<b>orth</b>	Cơ sở trực giao cho hạng của ma trận
<b>rank</b>	Tính hạng của ma trận
<b>rref</b>	Tính toán giảm hàng theo dạng bậc thang
<b>size</b>	Tính kích thước ma trận

– *Trị riêng và vectơ riêng*

<b>charpoly</b>	Đa thức riêng của ma trận
<b>eig</b>	Giá trị riêng vectơ riêng của ma trận
<b>jordan</b>	Dạng Jordan của ma trận

– *Phân tách ma trận*

<b>chol</b>	Phân tích thừa số Cholesky
<b>jordan</b>	Dạng Jordan của ma trận
<b>lu</b>	Phân tích thừa số LU
<b>qr</b>	Phân tích thừa số QR
<b>svd</b>	Phân tích giá trị đơn ma trận dạng biểu thức toán học

– *Các phương trình tuyến tính*

<b>cond</b>	Điều kiện số của ma trận
<b>det</b>	Tính định thức của ma trận
<b>equationsToMatrix</b>	Chuyển đổi tập các phương trình tuyến tính sang dạng ma trận
<b>inv</b>	Tính ma trận nghịch đảo
<b>linsolve</b>	Giải hệ phương trình tuyến tính biểu diễn theo dạng ma trận
<b>norm</b>	Chuẩn của ma trận hoặc vectơ
<b>rank</b>	Tính hạng của ma trận
<b>rref</b>	Tính toán giảm hàng theo dạng bậc thang

+ *Giả định*

<b>assume</b>	Thiết lập giả định với đối tượng biểu thức (toán học)
<b>assumeAlso</b>	Thêm thiết lập giả định với đối tượng biểu thức
<b>assumptions</b>	Hiển thị tập các giả thiết/giả định trên biến dạng biểu thức (toán học)
<b>clear all</b>	Xóa các biến trên workspace và reset MuPAD
<b>isAlways</b>	Kiểm tra xem liệu các phương trình hoặc bất phương trình có chứa tất các giá trị của các biến của nó
<b>isfinite</b>	Kiểm tra các phần tử của mảng có phải giá trị hữu hạn không
<b>isinf</b>	Kiểm tra các phần tử của mảng có phải là vô cùng

<b>isnan</b>	Kiểm tra các phần tử của mảng có phải là NaN (không phải số)
<b>logical</b>	Kiểm tra tính hợp lệ của phương trình và bất phương trình.
<b>reset</b>	Đóng cơ chế MuPAD
<b>+ Đa thức</b>	
<b>charpoly</b>	Đa thức riêng của ma trận
<b>coeffs</b>	Các hệ số của đa thức
<b>minpoly</b>	Đa thức tối thiểu của ma trận
<b>poly2sym</b>	Chuyển đổi từ dạng vectơ hệ số của đa thức sang dạng đa thức
<b>sym2poly</b>	Chuyển đổi từ dạng đa thức sang dạng vectơ hệ số của đa thức
<b>+ Các hàm toán học</b>	
<b>– Các hằng số</b>	
<b>catalan</b>	Hằng số Catalan
<b>eulergamma</b>	Hằng số Euler-Mascheroni
<b>– Logarit</b>	
<b>log</b>	Tính logarit tự nhiên của các phần tử của ma trận
<b>log10</b>	Tính logarit cơ số 10 của các phần tử của ma trận
<b>log2</b>	Tính logarit cơ số 2 của các phần tử của ma trận
<b>mfun</b>	Ước lượng giá trị của hàm toán học đặc biệt
<b>mfunlist</b>	Liệt kê các hàm đặc biệt dùng với hàm <b>mfun</b>
<b>– Các hàm lượng giác</b>	
<b>sin</b>	Hàm sin
<b>cos</b>	Hàm cosin
<b>tan</b>	Hàm tang
<b>cot</b>	Hàm cotang
<b>sec</b>	Hàm secant ( $\sec \theta = 1 / \cos \theta$ )
<b>csc</b>	Hàm cosecant ( $\csc \theta = 1 / \sin \theta$ )
<b>asin</b>	Hàm arcsine
<b>acos</b>	Hàm arccos
<b>atan</b>	Hàm arctang
<b>acot</b>	Hàm arccotang
<b>asec</b>	Hàm arcsecant
<b>acsc</b>	Hàm arccosecant
<b>– Các hàm Hyperbolic</b>	
<b>sinh</b>	Hàm sine hyperbol

<b>cosh</b>	Hàm cosine hyperbol
<b>tanh</b>	Hàm tang hyperbol
<b>coth</b>	Hàm cotang hyperbol
<b>sech</b>	Hàm secant hyperbol
<b>csch</b>	Hàm cosecant hyperbol
<b>asinh</b>	Hàm arcsin hyperbol
<b>acosh</b>	Hàm arccos hyperbol
<b>atanh</b>	Hàm arctang hyperbol
<b>acoth</b>	Hàm arccotang hyperbol
<b>asech</b>	Hàm arcsecant hyperbol
<b>acsch</b>	Hàm arccosecant hyperbol

– Các số phức

<b>abs</b>	Giá trị tuyệt đối của số thực hoặc số phức
<b>angle</b>	Góc pha
<b>atan2</b>	Hàm atan cho 4 góc 1/4
<b>imag</b>	Phần ảo của số phức
<b>real</b>	Phần thực của số phức
<b>sign</b>	Dấu của số thực hoặc số phức

– Các hàm đặc biệt

#### Hàm Dirac và Heaviside

<b>dirac</b>	Xung Dirac
<b>heaviside</b>	Hàm nhảy bậc Heaviside
<b>mfunlist</b>	Liệt kê các hàm đặc biệt dùng với hàm <b>mfun</b>
<b>rectangularPulse</b>	Tạo xung chữ nhật
<b>triangularPulse</b>	Tạo xung tam giác

#### Các hàm Gamma

<b>beta</b>	Hàm Beta
<b>factorial</b>	Tính giai thừa
<b>gamma</b>	Hàm Gamma
<b>gammain</b>	Hàm Gamma Logarit
<b>igamma</b>	Hàm Gamma không đầy đủ
<b>nchoosek</b>	Hệ số của nhị thức

#### Các hàm Zeta và Polylogarithms

<b>catalan</b>	Hằng số Catalan
<b>dilog</b>	Hàm Dilogarithm
<b>psi</b>	Hàm Digamma

<b>zeta</b>	Hàm Riemann zeta
<b>Các hàm Airy và Bessel</b>	
<b>airy</b>	Hàm Airy
<b>besseli</b>	Hàm Bessel sửa đổi loại 1
<b>besselj</b>	Hàm Bessel loại 1
<b>besselk</b>	Hàm Bessel sửa đổi loại 2
<b>bessely</b>	Hàm Bessel loại 2
<b>Tích phân lượng giác và tích phân hàm mũ</b>	
<b>coshint</b>	Hàm tích phân cosine Hyperbolic
<b>cosint</b>	Hàm tích phân cosine
<b>dawson</b>	Tích phân Dawson
<b>ei</b>	Tích phân hàm mũ một đối số
<b>eulergamma</b>	Hằng số Euler-Mascheroni
<b>expint</b>	Tích phân hàm mũ
<b>fresnelc</b>	Tích phân cosine Fresnel
<b>fresnels</b>	Tích phân sine Fresnel
<b>logint</b>	Tích phân hàm Logarit
<b>sinhint</b>	Tích phân sine Hyperbol
<b>sinint</b>	Tích phân sine
<b>ssinint</b>	Hàm tích phân sin dịch
<b>Các hàm lỗi (Error Functions)</b>	
<b>erf</b>	Hàm lỗi
<b>erfc</b>	Hàm lỗi bù
<b>erfcinv</b>	Hàm lỗi bù nghịch đảo
<b>erfi</b>	Hàm lỗi ảo
<b>erfinv</b>	Hàm lỗi đảo
<b>Hypergeometric and Whittaker Functions</b>	
<b>hypergeom</b>	Hình học siêu hình
<b>whittakerM</b>	Hàm Whittaker M
<b>whittakerW</b>	Hàm Whittaker W
<b>Tích phân elliptic (Elliptic Integrals)</b>	
<b>ellipke</b>	Tích phân elliptic đầy đủ loại 1 và 2
<b>ellipticCE</b>	Tích phân elliptic đầy đủ bù loại 2
<b>ellipticCK</b>	Tích phân elliptic đầy đủ bù loại 1
<b>ellipticCPi</b>	Tích phân elliptic đầy đủ bù loại 3
<b>ellipticE</b>	Tích phân elliptic loại 2

<b>ellipticF</b>	Tích phân elliptic thiếu loại 1
<b>ellipticK</b>	Tích phân elliptic đầy đủ loại 1
<b>ellipticPi</b>	Tích phân elliptic loại 3
<b>Các hàm Lambert W và Wright (Lambert W and Wright Functions)</b>	
<b>lambertw</b>	Hàm Lambert W
<b>wrightOmega</b>	Hàm omega Wright
<b>+ Các thao tác trên số</b>	
<b>double</b>	Chuyển ma trận dạng biến chữ sang dạng số
<b>ceil</b>	Làm tròn ma trận về phía dương vô cùng (làm tròn lên)
<b>conj</b>	Liên hợp phức
<b>fix</b>	Làm tròn về 0 (Round toward zero)
<b>floor</b>	Làm tròn về phía âm vô cùng (làm tròn xuống)
<b>imag</b>	Phần ảo của số phức
<b>max</b>	Phần tử lớn nhất
<b>min</b>	Phần tử nhỏ nhất
<b>real</b>	Phần thực của số phức
<b>round</b>	Làm tròn số
<b>+ Lý thuyết số</b>	
<b>bernoulli</b>	Các đa thức và số Bernoulli
<b>euler</b>	Các đa thức và số Euler
<b>harmonic</b>	Hàm (số) điều hòa (Harmonic)
<b>mod</b>	Phép tính modulo (chia lấy phần dư)
<b>quorem</b>	Thương và phần dư
<b>+ Đồ họa</b>	
– Hàm vẽ	
<b>ezcontour</b>	Vẽ đường viền
<b>ezcontourf</b>	Vẽ viền và tô
<b>ezmesh</b>	Vẽ lưới 3-d
<b>ezmeshc</b>	Kết hợp và lưới và viền
<b>ezplot</b>	Plot symbolic expression, equation, or function
<b>ezplot3</b>	Vẽ đường cong thông số dạng 3-D
<b>ezpolar</b>	Vẽ trong tọa độ cực
<b>ezsurf</b>	Vẽ bề mặt tô màu 3 – D
<b>ezsurf</b>	Kết hợp vẽ bề mặt và viền



## TÀI LIỆU THAM KHẢO

- [1]. La Văn Hiến, *Nhập môn MATLAB*, NXB. ĐHQG TP. HCM, 2003
- [2]. Nguyễn Phùng Quang, *Matlaab & Simulink dành cho kỹ sư Điều khiển tự động*, Nhà xuất bản Khoa học Kỹ thuật, 2004
- [3]. Phạm Hồng Liên, *MATLAB và ứng dụng trong viễn thông*, NXB ĐHQG TPHCM, 2006
- [4]. The MathWorks (1999 – 2001 – 2004), *Learning MATLAB (student version)*, The MathWorks Inc., USA
- [5]. Mathworks Inc., *MATLAB ® Primer*, 2014
- [6]. Mathworks Inc., *Symbolic Math Toolbox: User's Guide*, 2014
- [7]. Mathworks Inc., *DSP System Toolbox: Getting Started Guide*, 2014
- [8]. Mathworks Inc., *Control System Toolbox: User's Guide*, 2014
- [9]. Mathworks Inc., *Symbolic Math Toolbox: User's Guide*, 2014
- [10]. Mathworks Inc., *Signal Processing Toolbox: User's Guide*, 2004
- [11]. Mathworks Inc., *MATLAB Document*, <http://www.mathworks.com/help/matlab/>
- [12]. Đỗ Phi Nga, *Bài giảng Đại số*, Học viện Công nghệ BCVT, 2009.
- [13]. Đỗ Phi Nga, *Bài giảng toán cao cấp 2: Học phần Đại số - Dành cho ngành Quản trị Kinh doanh*, Học viện Công nghệ BCVT, 2009.
- [14]. Nguyễn Thị Dung, *Bài giảng toán cao cấp 1: Học phần Giải tích – Dành cho ngành Quản trị Kinh doanh*, Học viện Công nghệ BCVT, 2009.
- [15]. Lê Bá Long, *Bài giảng toán kỹ thuật – Dành cho sinh viên ngành Điện tử - Viễn thông*, Học viện Công nghệ BCVT, 2009.
- [16]. Jean-Marie Monier, *Giáo trình toán Đại số*, NXB Giáo dục, 2007
- [17]. Jean-Marie Monier, *Giáo trình toán Giải tích*, NXB Giáo dục, 2007
- [18]. Nguyễn Quốc Trung, *Xử lý tín hiệu và lọc số*, Tập 1,2, NXB KHKT, 2006
- [19]. Đặng Hoài Bắc, Lê Xuân Thành, *Bài giảng xử lý tín hiệu số*, Học viện Công nghệ BCVT, 2013.
- [20]. Hà Thu Lan, *Bài giảng xử lý tín hiệu số*, Học viện Công nghệ BCVT, 2009.
- [21]. Huỳnh Thái Hoàng, *Lý thuyết điều khiển tự động*, Đại học BK Tp. Hồ Chí Minh, 2006
- [22]. Đặng Hoài Bắc, Vũ Anh Đào, *Cơ sở điều khiển tự động*, Học viện Công nghệ BCVT, 2014
- [23]. Ngô Đức Thiện, Bùi Thị Dân, *Bài giảng Truyền thông số*, Học viện Công nghệ BCVT, 2010.
- [24]. Lê Nhật Thăng, Trần Thúy Hà, *Truyền dẫn số*, Học viện CNBCVT, 2013.