

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
ĐẠI HỌC CÔNG NGHỆ TP.HCM**

# **T** **HỰC HÀNH XỬ LÝ TÍN** **HIỆU SỐ**

Biên soạn:

ThS. Phạm Hùng Kim Khánh

[www.hutech.edu.vn](http://www.hutech.edu.vn)

**THỰC HÀNH XỬ LÝ TÍN HIỆU SỐ**

Ấn bản 2014



# MỤC LỤC

MỤC LỤC .....	1
HƯỚNG DẪN .....	3
<b>BÀI 1: PHẦN MỀM MATLAB.....</b>	<b>5</b>
1.1 KHỞI ĐỘNG MATLAB .....	5
1.2 CÁC VẤN ĐỀ CƠ BẢN .....	8
1.2.1 Các phép toán và toán tử .....	8
1.2.2 Khai báo biến .....	9
1.2.3 Các lệnh thường dùng .....	9
1.3 LẬP TRÌNH TRONG MATLAB .....	10
1.3.1 Các phát biểu điều kiện if, else, elseif .....	10
1.3.2 Switch .....	10
1.3.3 While .....	10
1.3.4 For .....	11
1.3.5 Break: .....	11
1.4 MA TRẬN .....	11
1.4.1 Các thao tác trên ma trận .....	11
1.4.2 Vector.....	15
1.4.3 Đa thức.....	15
1.5 ĐỒ HOẠ .....	16
1.5.1 Các lệnh vẽ .....	16
1.5.2 Tạo hình vẽ .....	16
1.5.3 Kiểu đường vẽ .....	16
1.5.4 Vẽ với hai trục y.....	17
1.5.5 Vẽ đường cong 3-D.....	18
1.5.6 Vẽ nhiều trục tọa độ .....	18
1.5.7 Đặt các thông số cho trục.....	19
1.5.8 Đồ hoạ đặc biệt.....	23
1.5.9 Đồ hoạ 3D.....	25
1.5.10 Thực hành vẽ đồ thị.....	26
1.6 CÁC FILE VÀ HÀM .....	28
1.6.1 Script file (file kịch bản) .....	28
1.6.2 File hàm.....	28
1.6.3 Các hàm toán học cơ bản .....	29
1.6.4 Các phép toán trên hàm toán học.....	30

1.6.5 Thực hành trên script và function .....	31
<b>BÀI 2: TÍN HIỆU RỜI RẠC THEO THỜI GIAN .....</b>	<b>38</b>
2.1 CÁC TÍN HIỆU SƠ CẤP .....	38
2.2 CÁC PHÉP TOÁN .....	39
2.3 KIỂM TRA TÍNH CHẤT TUYẾN TÍNH VÀ BẤT BIẾN .....	39
2.4 HỆ LTI .....	41
<b>BÀI 3: BIẾN ĐỔI Z .....</b>	<b>44</b>
3.1 CÁC ĐIỂM CỰC VÀ ĐIỂM KHÔNG .....	44
3.2 PHÂN TÍCH DÙNG PHƯƠNG PHÁP THẲNG DƯ .....	45
3.3 BIẾN ĐỔI Z VÀ Z NGƯỢC .....	46
<b>BÀI 4: BIẾN ĐỔI FOURIER RỜI RẠC.....</b>	<b>49</b>
4.1 TÍNH DTFT.....	49
4.2 FFT VÀ CÁC TÍNH CHẤT .....	50
<b>BÀI 5: BỘ LỌC SỐ FIR .....</b>	<b>53</b>
5.1 CÁC LOẠI BỘ LỌC .....	53
5.2 PHƯƠNG PHÁP CỬA SỔ.....	55
5.3 PHƯƠNG PHÁP LẤY MẪU TẦN SỐ.....	58
5.4 PHƯƠNG PHÁP LẶP .....	60
<b>BÀI 6: BỘ LỌC SỐ IIR .....</b>	<b>65</b>
6.1 THIẾT KẾ BỘ LỌC TƯƠNG TỰ.....	65
6.2 THIẾT KẾ BỘ LỌC SỐ.....	69
TÀI LIỆU THAM KHẢO .....	73

# HƯỚNG DẪN

## MÔ TẢ MÔN HỌC

Thực hành xử lý tín hiệu số là môn học hỗ trợ cho môn Xử lý tín hiệu số của chuyên ngành Kỹ thuật Điện tử Truyền thông. Môn học này dựa trên MATLAB để kiểm chứng các lý thuyết đã học trong môn Xử lý tín hiệu số.

## NỘI DUNG MÔN HỌC

- Bài 1. Phần mềm MATLAB: cơ bản về MATLAB, cách lập trình cũng như cách xử lý ma trận, vẽ đồ thị trong MATLAB.
- Bài 2: Tín hiệu rời rạc theo thời gian: cách biểu diễn tín hiệu và hệ thống rời rạc theo thời gian, các tính chất và đáp ứng xung của hệ LTI.
- Bài 3: Biến đổi  $z$  và  $z$  ngược: chuyển tín hiệu từ miền thời gian sang miền  $z$ , các tính chất của biến đổi  $z$  và chuyển tín hiệu hữu tỷ trên miền  $z$  sang miền thời gian.
- Bài 4: Biến đổi Fourier rời rạc: chuyển tín hiệu rời rạc trên miền thời gian sang miền tần số, dùng các thuật toán FFT để xác định biến đổi Fourier rời rạc.
- Bài 5: Bộ lọc số FIR: thiết kế bộ lọc FIR theo yêu cầu cho trước.
- Bài 6: Bộ lọc số IIR: thiết kế bộ lọc IIR theo yêu cầu cho trước.

## KIẾN THỨC TIỀN ĐỀ

Môn học Thực hành Xử lý tín hiệu số đòi hỏi sinh viên có nền tảng về Xử lý tín hiệu số.

## YÊU CẦU MÔN HỌC

Người học phải dự học đầy đủ các buổi lên lớp và làm bài tập đầy đủ.

## CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Để học tốt môn này, người học cần thực hành theo hướng dẫn, làm các bài tập; đọc trước bài mới và tìm thêm các thông tin liên quan đến bài học.

## **PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC**

Môn học được đánh giá gồm:

- Điểm quá trình: 30%. Hình thức và nội dung do giảng viên quyết định, phù hợp với quy chế đào tạo và tình hình thực tế tại nơi tổ chức học tập.
- Điểm thi: 70%. Hình thức bài thi trên máy tính trong 60 phút.

# BÀI 1: PHẦN MỀM MATLAB

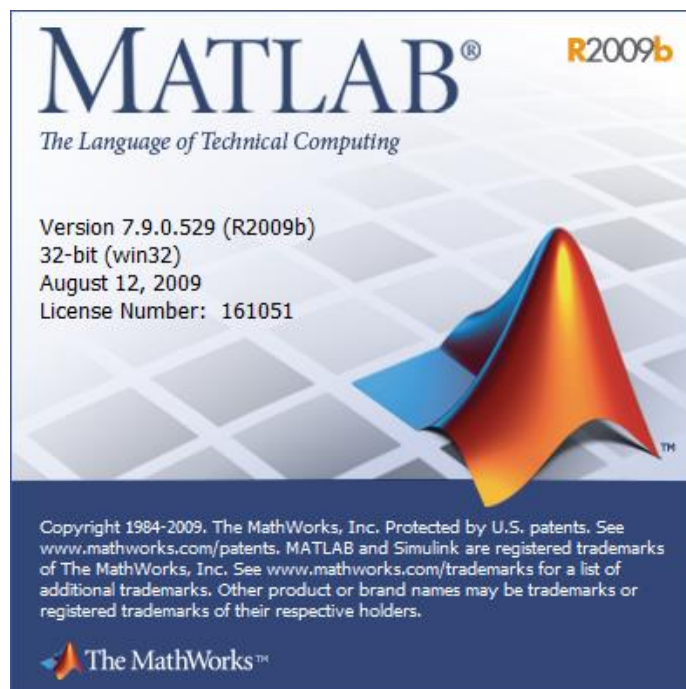
Sau khi học xong bài này, người học có thể:

- *Sử dụng phần mềm MATLAB.*
- *Thực hiện tạo các script file hay function và lưu trữ trên MATLAB.*
- *Biết được các công cụ cơ bản trên MATLAB.*

## 1.1 KHỞI ĐỘNG MATLAB

MATLAB (Matrix laboratory) là phần mềm dùng để giải các bài toán kỹ thuật, đặc biệt là các bài toán liên quan đến ma trận. MATLAB cung cấp các toolboxes, tức các hàm mở rộng môi trường MATLAB để giải quyết các vấn đề đặc biệt như xử lý tín hiệu số, hệ thống điều khiển, mạng neuron, fuzzy logic, mô phỏng v.v.

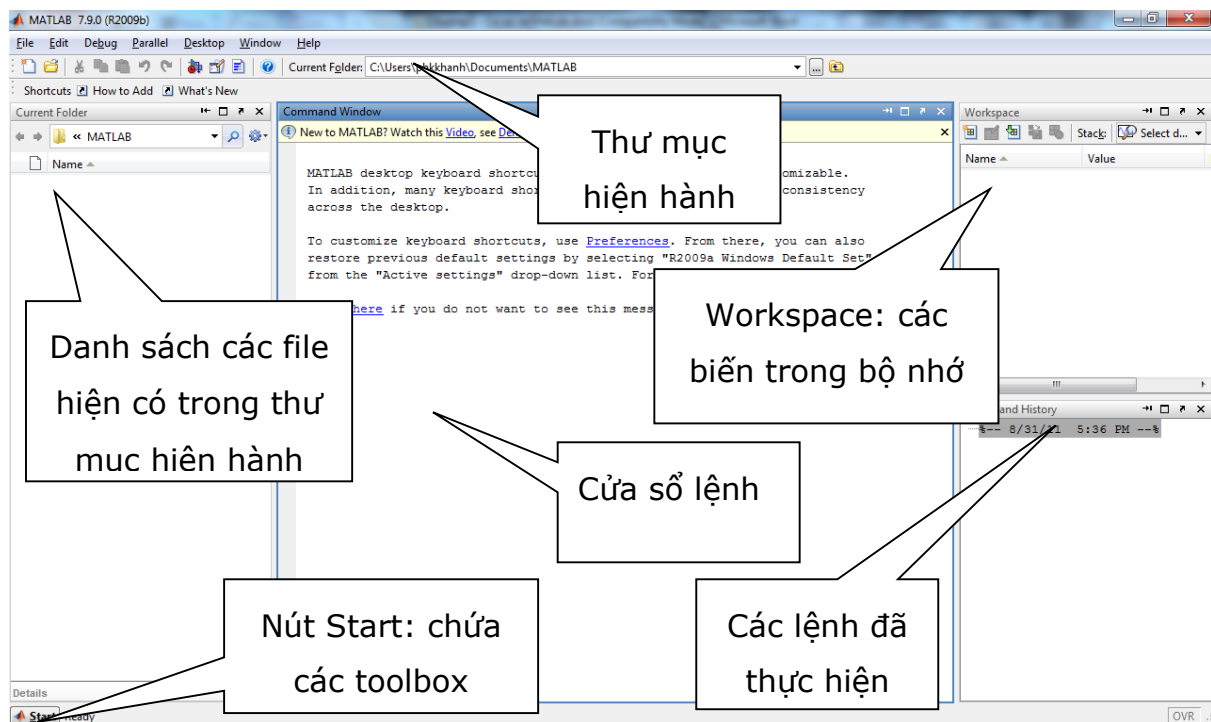
Cửa sổ biểu tượng của chương trình MATLAB:



Hình 1.1 - Cửa sổ khởi động của MATLAB



## Cửa sổ làm việc của MATLAB:

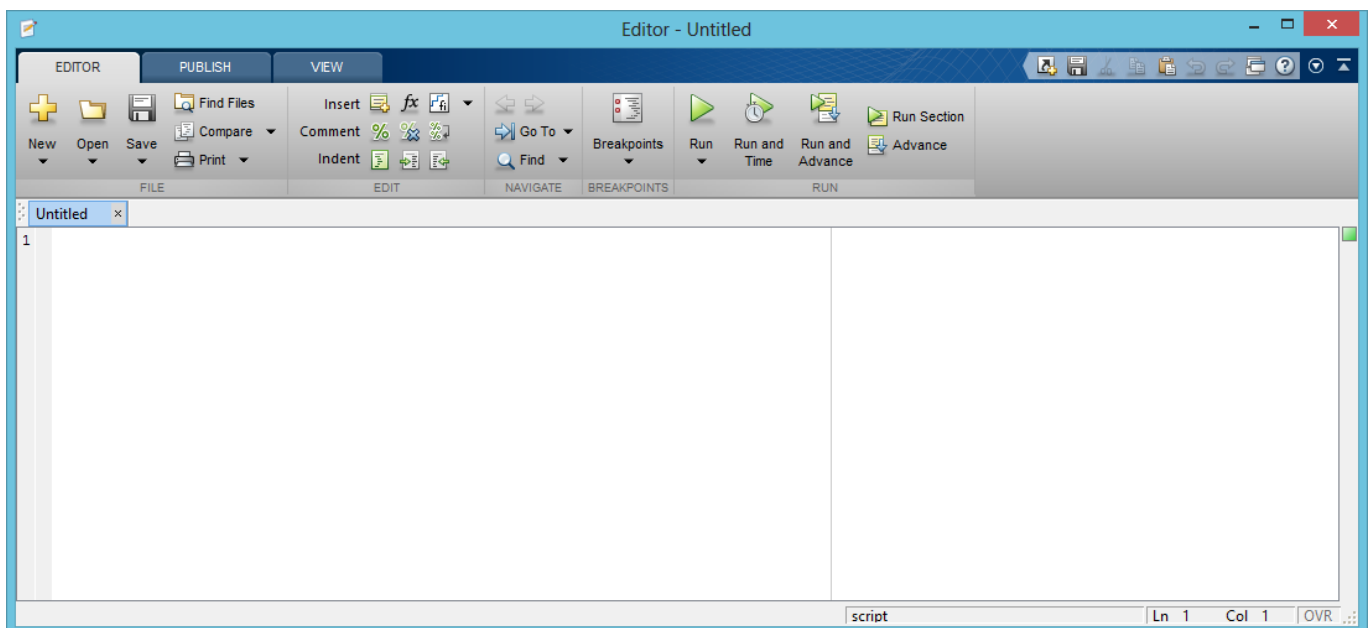


Hình 1.2 – Cửa sổ làm việc của MATLAB

### • Cửa sổ lệnh (Command window):

Là cửa sổ giao tiếp chính của MATLAB bởi đây là nơi nhập giá trị các biến, hiển thị giá trị, tính toán giá trị của biểu thức, thực thi các hàm có sẵn trong thư viện hoặc các hàm do người dùng lập trình ra trong M-files. Các lệnh được nhập sau dấu nhắc '>>' và thực thi lệnh bằng phím Enter. Để mở chương trình soạn thảo trong MATLAB, gõ lệnh:


```
>>edit
```



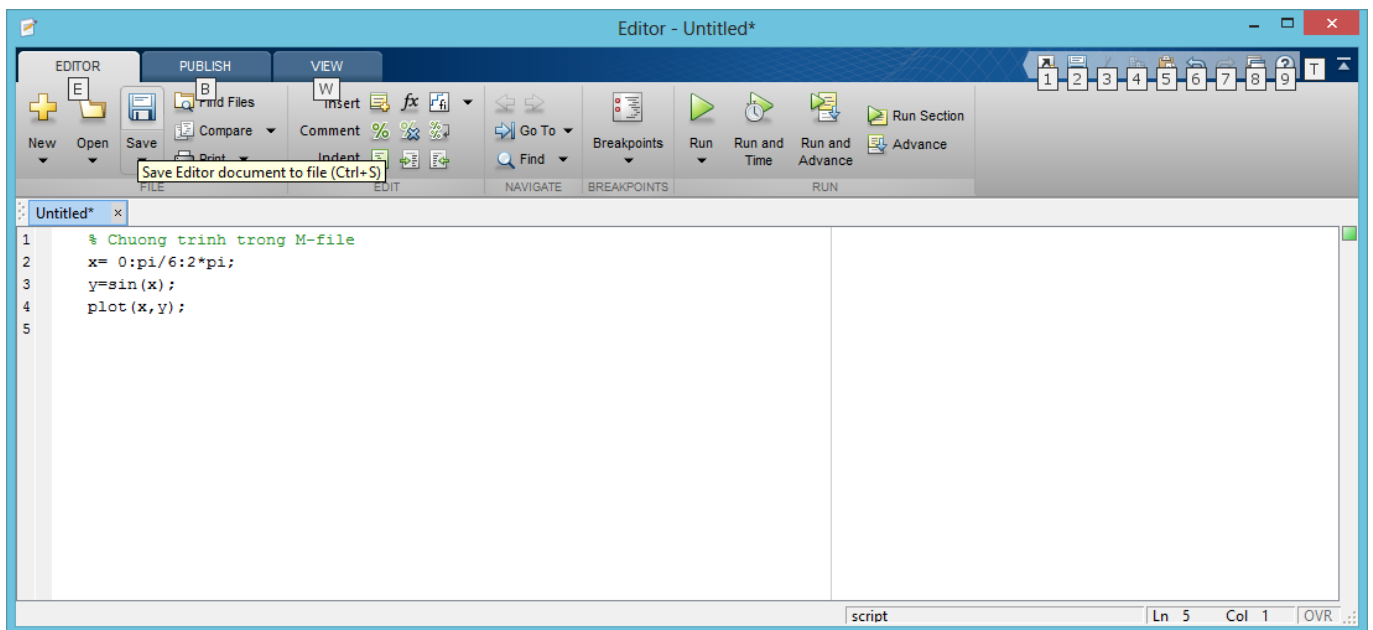
Hình 1.3 – Cửa sổ edit để soạn script file hay function

Sau đó nhập vào đoạn chương trình sau:

```
% Chương trình trong M-file  
  
x= 0:pi/6:2*pi;  
  
y=sin(x);  
  
plot(x,y);
```

Lưu chương trình với tên file *plot\_sin.m* bằng cách nhấn Ctrl+S hay nhấn vào biểu tượng Save .

Giải thích đoạn chương trình trên: Dòng 1 là dòng chú thích, các chuỗi ở phía sau dấu **%** sẽ không được dịch. Dòng 2 định nghĩa vector  $x$  trong khoảng từ 0 đến  $2\pi$  và mỗi giá trị cách nhau một khoảng  $\pi/6$ . Dòng 3 gán biến  $y = \sin(x)$  và dòng 4 vẽ đồ thị trong đó  $x$  là trục hoành và  $y$  là trục tung.



Hình 1.4 – Lưu file trong cửa sổ Edit

Thực thi chương trình trên trong Command window bằng dòng lệnh sau:

```
>>plot_sin
```

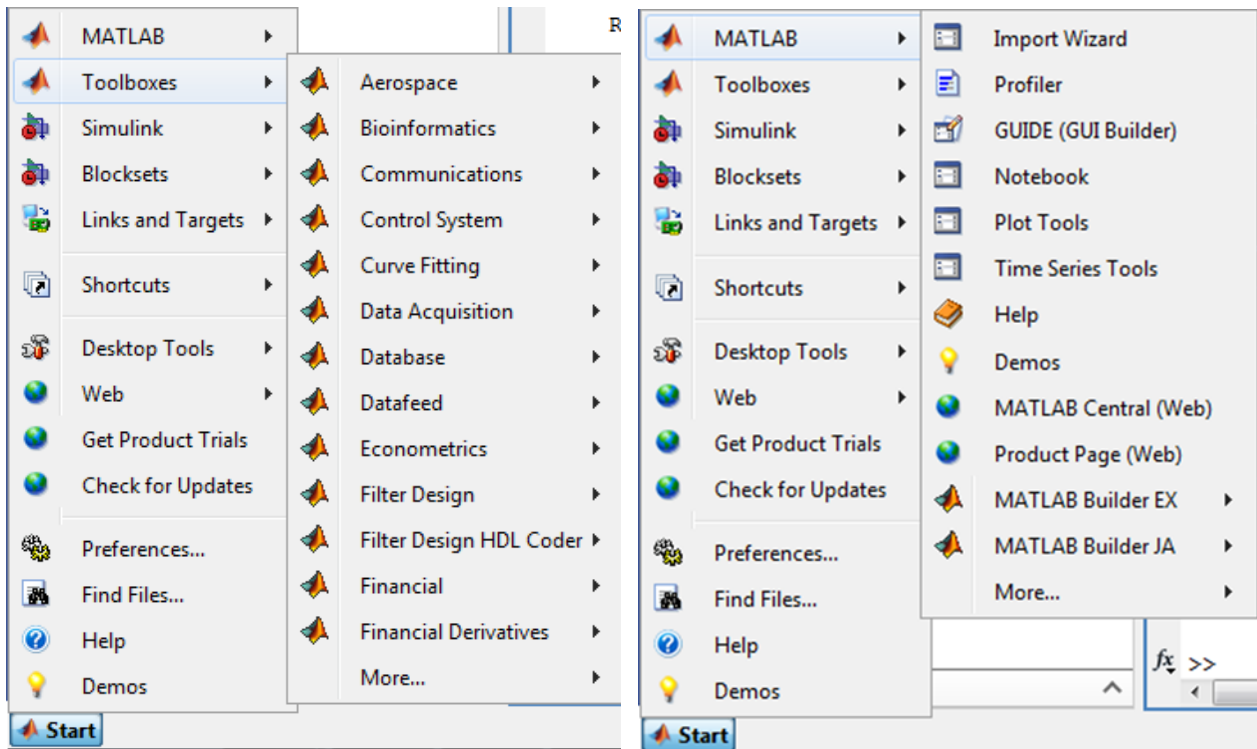
- **Cửa sổ Command History:**

Các dòng đã nhập trong Command window (các dòng này có thể là dòng nhập biến, có thể là dòng lệnh) được giữ lại trong cửa sổ Command History và cửa sổ này cho phép ta sử dụng lại những lệnh đó bằng cách nhấp đúp chuột lên các lệnh hay biến đó

- **Cửa sổ Workspace:**

Là cửa sổ thể hiện tên các biến bạn sử dụng cùng với kích thước vùng nhớ (số bytes), kiểu dữ liệu (lớp), các biến được giải phóng sau mỗi lần tắt chương trình. Cửa sổ Workspace cho phép thay đổi giá trị của biến bằng cách nhấn phím chuột phải lên các biến và chọn **Edit**.

- **Nút Start:**



Hình 1.5 – Nút Start

- **Cửa sổ Edit:**

Là cửa sổ dùng để soạn thảo chương trình ứng dụng, được khởi động bằng lệnh edit trong Command Window. Chương trình soạn thảo trong cửa sổ edit có 2 dạng:

- + Dạng Script file : tập hợp các câu lệnh viết dưới dạng liệt kê, không có biến dữ liệu vào và biến ra.
- + Dạng function: có biến dữ liệu vào và biến ra.

## 1.2 CÁC VẤN ĐỀ CƠ BẢN

### 1.2.1 Các phép toán và toán tử

Các phép toán: + , - , \* , / , \ (chia trái) , ^ (mũ) , ` (chuyển vị hay số phức liên hiệp).

Các toán tử quan hệ : < , <= , > , >= , == , ~=

Các toán tử logic : & , | (or) , ~ (not)

Các hằng:

pi      3.14159265

i, j	số ảo
eps	sai số $2^{-52}$
realmin	số thực nhỏ nhất $2^{-1022}$
realmax	số thực lớn nhất $2^{1023}$
inf	vô cùng lớn
NaN	Not a number

### Chú ý :

+ Các lệnh kết thúc bằng dấu chấm phẩy, MATLAB sẽ không thể hiện kết quả trên màn hình.

+ Các chú thích được đặt phía sau dấu %.

+ Trong quá trình nhập nếu các phần tử trên một hàng dài quá ta có thể xuống dòng bằng toán tử ba chấm (. . .)

### 1.2.2 Khai báo biến

- Phân biệt chữ hoa và chữ thường.
- Không cần phải khai báo kiểu biến.
- Tên biến phải bắt đầu bằng ký tự và không được có khoảng trắng.
- Không đặt tên trùng với các tên đặc biệt của MATLAB.
- Để khai báo biến toàn cục (sử dụng được trong tất cả chương trình con), phải dùng thêm từ khoá **global** phía trước.

### 1.2.3 Các lệnh thường dùng

```
>>help tên_hàm % tham khảo help của hàm
```

```
>>lookfor 'chuỗi' %Tìm kiếm chuỗi
```

**Ví dụ:** >>lookfor 'filter' % Tìm các hàm có liên quan đến mạch lọc

```
>>clc % Xoá màn hình
```

```
>>clear tên_biến % Xoá biến
```

```
>>clear all %Xoá tất cả các biến
```

```
>> clf %Xoá figure
```

```
>>save % Lưu các biến hiện có trong bộ nhớ
```

```
>>load % Lấy nội dung các biến đã lưu
```

```
>>who % liệt kê các biến trong bộ nhớ
```

```
>>whos % liệt kê chi tiết các biến trong bộ nhớ
```

```
>>which % Xác định vị trí của hàm hay file
```

**Ví dụ:** >>which plot %Xác định vị trí của hàm plot

>>what % Liệt kê các file có trong một thư mục

## 1.3 LẬP TRÌNH TRONG MATLAB

---

### 1.3.1 Các phát biểu điều kiện if, else, elseif

Cú pháp của if:

```
if <biểu thức điều kiện>
    <phát biểu>
end
```

Nếu <biểu thức điều kiện> cho kết quả đúng thì phần lệnh trong thân của if được thực hiện. Các phát biểu else và elseif cũng tương tự.

### 1.3.2 Switch

Cú pháp của switch như sau:

```
switch <biểu thức>
    case n1
        <lệnh 1>
    case n2
        <lệnh 2>
    . . . . .
    case nn
        <lệnh n>
    Otherwise
        <lệnh n+1>
end
```

### 1.3.3 While

Vòng lặp while dùng khi không biết trước số lần lặp. Cú pháp của nó như sau:

```
while <biểu thức>
    <phát biểu>
end
```

### 1.3.4 For

Vòng lặp for dùng khi biết trước số lần lặp. Cú pháp như sau:

```
for <chỉ số>=<giá trị đầu>:<mức tăng>:<giá trị cuối>
```

### 1.3.5 Break:

Phát biểu break để kết thúc vòng lặp for hay while mà không quan tâm đến điều kiện kết thúc vòng lặp đã thoả mãn hay chưa.

## 1.4 MA TRẬN

### 1.4.1 Các thao tác trên ma trận

#### 1.4.1.1 Nhập ma trận

Ma trận là một mảng có m hàng và n cột. Trường hợp ma trận chỉ có một phần tử (ma trận 1x1) ta có một số. Ma trận chỉ có một cột hay một hàng được gọi là một vector. Ta có thể nhập ma trận vào MATLAB bằng nhiều cách:

- Nhập một danh sách các phần tử từ bàn phím.
- Nạp ma trận từ file.
- Tạo ma trận nhờ các hàm có sẵn trong MATLAB.
- Tạo ma trận nhờ hàm tự tạo.

Khi nhập ma trận từ bàn phím ta phải tuân theo các quy định sau:

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

#### 1.4.1.2 Chỉ số

Phần tử ở hàng i cột j của ma trận có ký hiệu là  $A(i,j)$ . Tuy nhiên, ta cũng có thể tham chiếu tới phần tử của mảng nhờ một chỉ số, ví dụ  $A(k)$ . Trong trường hợp này, ma trận được xem là một cột dài tạo từ các cột của ma trận ban đầu.

Như vậy viết  $A(8)$  có nghĩa là tham chiếu phần tử  $A(4, 2)$  (nếu ma trận có 4 hàng). Lưu ý rằng các chỉ số của ma trận thường bắt đầu từ 1.

#### 1.4.1.3 Toán tử ":"

Toán tử ":" là một toán tử quan trọng của MATLAB. Nó xuất hiện ở nhiều dạng khác nhau. Biểu thức 1:10 là một vector hàng chứa 10 số nguyên từ 1 đến 10.

```
>>1:10
```

```
>>100:-7:50 %tạo dãy số từ 100 đến 51, cách đều nhau 7
```

```
>>0: pi/4: pi %tạo một dãy số từ 0 đến  $\pi$ , cách đều nhau  $\pi/4$ 
```

Các biểu thức chỉ số có thể tham chiếu tới một phần của ma trận.  $A(1:k,j)$  xác định k phần tử đầu tiên của cột j. Ngoài ra toán tử ":" tham chiếu tới tất cả các phần tử của một hàng hay một cột.

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

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

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

#### 1.4.1.4 Tạo ma trận bằng hàm có sẵn

MATLAB cung cấp một số hàm để tạo các ma trận cơ bản:

- **zeros** tạo ra ma trận mà các phần tử đều là 0.

```
>>z = zeros(2, 4)
```

- **ones** tạo ra ma trận mà các phần tử đều là 1.

```
>>x = ones(2, 3)
```

```
>>y = 5*ones(2, 2)
```

- **rand** tạo ra ma trận mà các phần tử ngẫu nhiên phân bố đều.

```
>>d = rand(4, 4)
```

- **randn** tạo ra ma trận mà các phần tử ngẫu nhiên phân bố chuẩn.

```
>>e = randn(3, 3)
```

- **magic(n)** tạo ra ma trận cấp n gồm các số nguyên từ 1 đến  $n^2$  với tổng các hàng bằng tổng các cột và bằng tổng các đường chéo ( $n \geq 3$ ).

- **pascal(n)** tạo ra tam giác Pascal.

```
>>pascal(4)
```

- **eye(n)** tạo ma trận đơn vị

```
>>eye(3)
```

- **eye(m,n)** tạo ma trận đơn vị mở rộng

```
>>eye(3,4)
```

### 1.4.1.5 Nối ma trận

Ta có thể nối các ma trận có sẵn thành một ma trận mới. Ví dụ:

```
>>a = ones(3, 3);
>>b = 5*ones(3, 3);
>>c = [a+2; b]
```

### 1.4.1.6 Xoá hàng và cột

Ta có thể xoá hàng và cột từ ma trận bằng dùng dấu [].

```
>>b(:, 2) = [] ; %xoá cột thứ 2
>>b(1:2:5) = []; % xoá các phần tử bắt đầu từ 1 đến 5 và cách 2
(1,3,5) rồi sắp xếp lại ma trận.
```

### 1.4.1.7 Các lệnh xử lý ma trận

Cộng :  $X = A + B$

Trừ :  $X = A - B$

Nhân :  $X = A*B$

$X = A.*B$  nhân các phần tử tương ứng với nhau, yêu cầu 2 ma trận A và B phải có cùng kích thước.

Chia :  $X = A/B$  lúc đó  $X = A * \text{inv}(B)$

$X = A \setminus B$  lúc đó  $X = \text{inv}(A) * B$

$X=A./B$  chia các phần tử tương ứng với nhau, 2 ma trận A và B có cùng kích thước.

Luỹ thừa :  $X = A^2$

$X = A.^2$

Nghịch đảo:  $X = \text{inv}(A)$

Định thức:  $d = \text{det}(A)$

Để tạo ma trận trong MATLAB ta chỉ cần liệt các phần tử của ma trận trong cặp dấu ngoặc vuông ([...]). Các phần tử trên cùng hàng được phân biệt bởi dấu phẩy (,) hoặc khoảng trắng (space). Các hàng của ma trận, phân cách nhau bởi dấu chấm phẩy (;). Ví dụ, nhập ma trận A có 4 hàng, 4 cột như sau:

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



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

Để truy xuất đến từng phần tử của ma trận ta dùng chỉ số phần tử tương ứng. Ví dụ, phần tử ở hàng thứ 2, cột thứ 3 của A là A(2,3).

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

**Bài 1.1.** Cho ma trận  $A = [2 \ 4 \ 1; \ 6 \ 7 \ 2; \ 3 \ 5 \ 9]$ , sinh viên dùng các lệnh cần thiết để:

- Lấy dòng đầu tiên của ma trận A.
- Tạo ma trận B bằng 2 dòng cuối cùng của A.
- Tính tổng các phần tử trên các cột của A. (gợi ý: tính tổng các phần tử trên cột 1: `sum(A(:,1))`).
- Tính tổng các phần tử trên các dòng của A.

**Bài 1.2.** Cho ma trận  $A = [2 \ 7 \ 9 \ 7; \ 3 \ 1 \ 5 \ 6; \ 8 \ 1 \ 2 \ 5]$ , sinh viên giải thích kết quả của các lệnh sau:

- A'
- A(:, [1 4])
- A([2 3], [3 1])
- reshape(A, 2, 6)
- A(:)
- [A; A(end, :)]
- A(1:3, :)
- [A ; A(1:2, :)]
- sum(A)
- sum(A')
- [ [ A ; sum(A) ] [ sum(A, 2) ; sum(A(:)) ] ]

**Bài 1.3.** Giải hệ phương trình sau:

$$2x_1 + 4x_2 + 6x_3 - 2x_4 = 0$$

$$x_1 + 2x_2 + x_3 + 2x_4 = 1$$

$$2x_2 + 4x_3 + 2x_4 = 2$$

$$3x_1 - x_2 + 10x_4 = 10.$$

**Gợi ý:**  $A.X = B \rightarrow X = \text{inv}(A).B$

**Bài 1.4.** Chứng tỏ rằng  $(A+B)C=AC+BC$ , với:

$$A = \begin{bmatrix} 10 & -2 \\ 20 & 4 \\ 3 & 6 \end{bmatrix}, B = \begin{bmatrix} 3 & 1 \\ -10 & 2 \\ 0 & 5 \end{bmatrix} \text{ và } C = \begin{bmatrix} -3 & 4 \\ 6 & 1 \end{bmatrix}$$

### 1.4.2 Vector

Vector thực chất là ma trận có kích thước  $n \times 1$  hay  $1 \times n$ , nên ta có thể tạo ra vector như cách tạo ra ma trận. Ngoài ra, có thể dùng một số cách sau:

```
>>x=0:0.1:1
```

```
>>y=linspace(1, 10, 20) % vector 20 phần tử cách đều nhau từ 1
den 10
```

```
>>z=rand(10,1) ; tạo 10 số ngẫu nhiên phân bố đều
```

**Bài 1.5.** Cho vector  $x = [3 \ 1 \ 5 \ 7 \ 9 \ 2 \ 6]$ , giải thích kết quả của các lệnh sau:

- $x(3)$
- $x(1:7)$
- $x(1:end)$
- $x(1:end-1)$
- $x(6:-2:1)$
- $x([1 \ 6 \ 2 \ 1 \ 1])$
- $\text{sum}(x)$

**Bài 1.6.** Tạo một vector  $x$  có 100 phần tử, sao cho:  $x(n) = (-1)^{n+1}/(2n+1)$  với  $n = 0 - 99$ .

### 1.4.3 Đa thức

Các đa thức trong MATLAB được mô tả bằng các vector hàng với các phần tử của vector chính là các hệ số của đa thức, xếp theo thứ tự số mũ giảm dần. Ví dụ, đa thức  $m = s^4 - s^3 + 4s^2 - 5s - 1$  được biểu diễn là:

```
>>m=[1 -1 4 -5 -1]
```

Để xác định giá trị của đa thức, ta dùng hàm **polyval**. Ví dụ, xác định giá trị của đa thức tại điểm  $s=2$ :

```
>> polyval(m, 2)
```

Để xác định nghiệm của đa thức, ta dùng hàm **roots**. Ví dụ:

```
>> roots(m)
```

**Bài 1.7.** Cho phương trình  $ax^2+bx+c=0$ , giải phương trình dùng hàm **roots**.

```
>>m=[a b c];
```

```
>>x=roots(m)
```

Thay đổi các giá trị khác nhau của a, b và c tương ứng trong 2 cách giải trên.

**Bài 1.8.** Giải phương trình  $x^3-2x^2+4x+5=0$ . Kiểm chứng kết quả thu được bằng hàm **polyval**. Sinh viên có nhận xét gì về kết quả kiểm chứng.

**Bài 1.9.** Lặp lại bài 1.8 cho phương trình  $x^7-2=0$ .

## 1.5 ĐỒ HOẠ

---

### 1.5.1 Các lệnh vẽ

MATLAB cung cấp một loạt hàm để vẽ biểu diễn các vector cũng như giải thích và in các đường cong này.

**plot:** đồ họa 2-D với số liệu 2 trục vô hướng và tuyến tính

**plot3:** đồ họa 3-D với số liệu 2 trục vô hướng và tuyến tính

**loglog:** đồ họa với các trục x, y ở dạng logarit

**semilogx:** đồ họa với trục x logarit và trục y tuyến tính

**semilogy:** đồ họa với trục y logarit và trục x tuyến tính

### 1.5.2 Tạo hình vẽ

Hàm **plot** có các dạng khác nhau phụ thuộc vào các đối số đưa vào. Ví dụ nếu y là một vector thì plot(y) tạo ra một đường quan hệ giữa các giá trị của y và chỉ số của nó. Nếu ta có 2 vector x và y thì plot(x,y) tạo ra đồ thị quan hệ giữa x và y.

```
>>t = [0:pi/100:2*pi];
```

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

```
>>plot(t,y); % Vẽ hình sin từ 0- $2\pi$ 
```

```
>>grid on
```

### 1.5.3 Kiểu đường vẽ

Ta có thể dùng các kiểu đường vẽ khác nhau khi vẽ hình.

```
>>t = [0:pi/100:2*pi];
```

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

```
>>plot(t,y,'.') % vẽ bằng đường chấm chấm
```

Để xác định màu và kích thước đường vẽ, ta dùng các tham số sau:

**LineWidth** : độ rộng đường thẳng, tính bằng số điểm

**MarkerEdgeColor** : màu của các cạnh của khối đánh dấu

**MarkerFaceColor** : màu của khối đánh dấu

**MarkerSize** : kích thước của khối đánh dấu

Màu được xác định bằng các tham số:

**r**: red      **m** magenta      **g**: green      **y**: yellow

**b**: blue      **k**: black      **c**: cyan      **w**: white

Các dạng đường thẳng xác định bằng:

- đường liền      -- đường đứt nét

: đường chấm chấm      -. đường chấm gạch

Các dạng điểm đánh dấu xác định bằng:

**+** dấu cộng      **.** điểm      **o** vòng tròn      **x** chữ thập      **\*** dấu sao

**s** hình vuông      **d** hạt kim cương      **v** tam giác hướng xuống

**^** tam giác hướng lên      **<** tam giác sang trái

**>** tam giác sang phải      **h** lục giác      **p** ngũ giác

```
>>x = -pi : pi/10 : pi;
```

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

```
>>plot(x,y,'--rs','LineWidth',2,'MarkerEdgeColor','k',...
'MarkerFaceColor','g','MarkerSize',10)
```

Để vẽ hai hàm trên cùng một đồ thị, ta dùng lệnh:

```
>>hold on
```

### 1.5.4 Vẽ với hai trục y

Hàm **plotyy** cho phép tạo một đồ thị có hai trục y. Ta cũng có thể dùng **plotyy** để cho giá trị trên hai trục y có kiểu khác nhau nhằm tiện so sánh.

```
>>t = 0:900;
```

```
>>A = 1000;
```

```
>>b = 0.005;  
>>a = 0.005;  
>>z2 = sin(b*t);  
>>z1 = A*exp(-a*t);  
>>[haxes, hline1, hline2] = plotyy(t,z1,t,z2,'semilogy','plot');
```

### 1.5.5 Vẽ đường cong 3-D

Nếu  $x$ ,  $y$ ,  $z$  là 3 vector có cùng độ dài thì `plot3` sẽ vẽ đường cong 3D.

```
>>t = 0:pi/50:10*pi;  
>>plot3(sin(t),cos(t),t)  
>>axis square;  
>>grid on
```

### 1.5.6 Vẽ nhiều trục tọa độ

Dùng hàm ***subplot*** để vẽ nhiều trục tọa độ.

```
>>subplot(2,3,5) %2,3: xác định có 2 hàng, 3 cột  
% 5: chọn trục thứ 5 (đếm từ trái sang phải, trên xuống dưới)  
>>x=linspace(0,2*pi);  
>>y1=sin(x);  
>>y2=cos(x)  
>>y3=2*exp(-x).*sin(x);  
>>x1=linspace(-2*pi,2*pi);  
>>y4=sinc(x1);  
>>subplot(221);  
>>plot(x,y1);  
>>title('Ham y = sinx');  
>>subplot(222);  
>>plot(x,y2);  
>>title('Ham y = cosx');  
>>subplot(223);  
>>plot(x,y3);
```

```
>>title('Ham y = 2e^{-x}sinx');

>>subplot(224);

>>plot(x1,y4);

>>title('Ham y = $$\sin \pi x \over \pi x$$','interpreter','latex');
```

### 1.5.7 Đặt các thông số cho trục

Khi ta tạo một hình vẽ, MATLAB tự động chọn các giới hạn trên trục tọa độ và khoảng cách đánh dấu dựa trên dữ liệu dùng để vẽ. Tuy nhiên ta có thể mô tả lại phạm vi giá trị trên trục và khoảng cách đánh dấu theo ý riêng. Ta có thể dùng các hàm sau:

**axis** đặt lại các giá trị trên trục tọa độ.

**axes** tạo một trục tọa độ mới với các đặc tính được mô tả.

**get** và **set** cho phép xác định và đặt các thuộc tính của trục tọa độ đang có.

**gca** trở về trục tọa độ cũ.

#### 1.5.7.1 Giới hạn của trục và chia vạch trên trục

MATLAB chọn các giới hạn trên trục tọa độ và khoảng cách đánh dấu dựa trên số liệu dùng để vẽ. Dùng lệnh **axis** có thể đặt lại giới hạn này. Cú pháp của lệnh:

```
axis[xmin , xmax , ymin , ymax]

>>x = 0:0.025:pi/2;

>>plot(x,tan(x),'-ro')

>>axis([0 pi/2 0 5])
```

MATLAB chia vạch trên trục dựa trên phạm vi dữ liệu và chia đều. Ta có thể mô tả cách chia nhờ thông số **xtick** và **ytick** bằng một vector tăng dần.

```
>>x = -pi:.1:pi;

>>y = sin(x);

>>plot(x,y)

>>set(gca,'xtick',-pi:pi/2:p);

>>set(gca,'xticklabel',{'-pi','-pi/2','0','pi/2','pi'})
```

#### 1.5.7.2 Ghi nhãn lên các trục tọa độ

MATLAB cung cấp các lệnh ghi nhãn lên đồ họa gồm:

**title** thêm nhãn vào đồ hoạ.

**xlabel** thêm nhãn vào trục x.

**ylabel** thêm nhãn vào trục y.

**zlabel** thêm nhãn vào trục z.

**legend** thêm chú giải vào đồ thị.

**text** hiển thị chuỗi văn bản ở vị trí nhất định.

**gtext** đặt văn bản lên đồ hoạ nhờ chuột.

```
>>x = -pi:.1:pi;
>>y = sin(x);
>>plot(x,y)
>>xlabel('t = 0 to 2\pi','FontSize',16)
>>ylabel('sin(t)','FontSize',16)
>>title('\it{Gia tri cua sin tu zero đến 2 pi}','FontSize',16)
```

Ta có thể thêm văn bản vào bất kỳ chỗ nào trên hình vẽ nhờ hàm **text**.

```
>>text(3*pi/4,sin(3*pi/4),' \leftarrow sin(t)=0.707','FontSize',1
2)
```

Ta có thể sử dụng đối tượng văn bản để ghi chú các trục ở vị trí bất kỳ. MATLAB định vị văn bản theo đơn vị dữ liệu trên trục. Ví dụ để vẽ hàm  $y = Ae^{\alpha t}$  với  $A = 0.25$ ,  $t = 0$  đến 900 và  $\alpha = 0.005$ :

```
>>t = 0:900;
>>plot(t,0.25*exp(-0.005*t))
```

Để thêm ghi chú tại điểm  $t = 300$ , ta viết:

```
>>text(300,.25*exp(-.005*300),...
'\bullet\leftarrow\fontname{times}0.25\ite^{(-0.005\itt)}
at,...
\itt=300','FontSize',14)
```

Tham số **HorizontalAlignment** và **VerticalAlignment** định vị văn bản so với các toạ độ  $x$ ,  $y$ ,  $z$  đã cho.

Để thêm các ký tự đặc biệt, ta dùng format dạng Tex:

Character Sequence	Symbol	Character Sequence	Symbol	Character Sequence	Symbol
\alpha	$\alpha$	\upsilon	$\upsilon$	\sim	$\sim$
\beta	$\beta$	\phi	$\Phi$	\leq	$\leq$
\gamma	$\gamma$	\chi	$\chi$	\infty	$\infty$
\delta	$\delta$	\psi	$\psi$	\clubsuit	$\clubsuit$
\epsilon	$\epsilon$	\omega	$\omega$	\diamondsuit	$\diamondsuit$
\zeta	$\zeta$	\Gamma	$\Gamma$	\heartsuit	$\heartsuit$
\eta	$\eta$	\Delta	$\Delta$	\spadesuit	$\spadesuit$
\theta	$\theta$	\Theta	$\Theta$	\leftrightarrow	$\leftrightarrow$
\vartheta	$\vartheta$	\Lambda	$\Lambda$	\leftarrow	$\leftarrow$
\iota	$\iota$	\Xi	$\Xi$	\uparrow	$\uparrow$
\kappa	$\kappa$	\Pi	$\Pi$	\rightarrow	$\rightarrow$
\lambda	$\lambda$	\Sigma	$\Sigma$	\downarrow	$\downarrow$
\mu	$\mu$	\Upsilon	$\Upsilon$	\circ	$\circ$

\nu	$\nu$	\Phi	$\Phi$	\pm	$\pm$
\xi	$\xi$	\Psi	$\Psi$	\geq	$\geq$
\pi	$\pi$	\Omega	$\Omega$	\propto	$\propto$
\rho	$\rho$	\forall	$\forall$	\partial	$\partial$
\sigma	$\sigma$	\exists	$\exists$	\bullet	$\bullet$
\varsigma	$\varsigma$	\ni	$\ni$	\div	$\div$
\tau	$\tau$	\cong	$\cong$	\neq	$\neq$
\equiv	$\equiv$	\approx	$\approx$	\aleph	$\aleph$
\Im	$\Im$	\Re	$\Re$	\wp	$\wp$
\otimes	$\otimes$	\oplus	$\oplus$	\oslash	$\oslash$
\cap	$\cap$	\cup	$\cup$	\supseteq	$\supseteq$
\supset	$\supset$	\subseteq	$\subseteq$	\subset	$\subset$
\int	$\int$	\in	$\in$	\circ	$\circ$
\rfloor	$\rfloor$	\lceil	$\lceil$	\nabla	$\nabla$
\lfloor	$\lfloor$	\cdot	$\cdot$	\ldots	$\ldots$
\perp	$\perp$	\neg	$\neg$	\prime	$\prime$



\bf — Bold font

\it — Italic font

\sl — Oblique font (rarely available)

\rm — Normal font

\fontname{fontname}

\fontsize{fontsize}

\color(colorSpec)

Các chỉ số trên và dưới thực hiện bằng ^ và \_.

```
>>title('\ite^{i\omega_0\tau} = cos(\omega_0\tau) + i
sin(\omega_0\tau)')
```

Kết quả:  $e^{i\omega_0\tau} = \cos(\omega_0\tau) + i\sin(\omega_0\tau)$

Để thêm các công thức toán học, ta dùng dạng LaTeX. Một số ví dụ:

```
>>text('units','inch', 'position',[.2 5], ...
'fontsize',14, 'interpreter','latex', 'string',...
['$$\hbox {magic(3) is } \left( {\matrix{ 8 & 1 & 6 \cr'...
'3 & 5 & 7 \cr 4 & 9 & 2 } } \right)$$']);
>>text('units','inch', 'position',[.2 4], ...
'fontsize',14, 'interpreter','latex', 'string',...
['$$\left[ {\matrix{\cos(\phi) & -\sin(\phi) \cr'...
'\sin(\phi) & \cos(\phi) \cr}} \right]'...
'\left[ \matrix{x \cr y} \right]$$']);
>>text('units','inch', 'position',[.2 3], ...
'fontsize',14, 'interpreter','latex', 'string',...
['$$L\{f(t)\} \equiv F(s) = \int_0^{\infty}\!\!\!{e^{-st}}'...
'f(t)dt}$$']);
>>text('units','inch', 'position',[.2 2], ...
'fontsize',14, 'interpreter','latex', 'string',...
['$$e = \sum_{k=0}^{\infty} {1 \over {k!}} } $$']);
>>text('units','inch', 'position',[.2 1], ...
```

```
'fontsize',14, 'interpreter','latex', 'string',...
['\$\$m \ddot y = -m g + C_D \cdot \{1 \over 2\}'...
'\rho {\dot y}^2 \cdot A\$'];
>>text('units','inch', 'position',[.2 0], ...
'fontsize',14, 'interpreter','latex', 'string',...
'\$\$\int_{0}^{\infty} x^2 e^{-x^2} dx =
\frac{\sqrt{\pi}}{4}\$');
```

Các khai báo cụ thể tham khảo tại: <http://www.latex-project.org/>

## 1.5.8 Đồ hoạ đặc biệt

### 1.5.8.1 Khối và vùng

Đồ hoạ khối và vùng biểu diễn số liệu là vector hay ma trận. MATLAB cung cấp các hàm đồ hoạ khối và vùng:

**bar** hiển thị các cột của ma trận  $m \times n$  như là  $m$  nhóm, mỗi nhóm có  $n$  bar.

**barh** hiển thị các cột của ma trận  $m \times n$  như là  $m$  nhóm, mỗi nhóm có  $n$  bar nằm ngang.

**bar3** hiển thị các cột của ma trận  $m \times n$  như là  $m$  nhóm, mỗi nhóm có  $n$  bar dạng 3D.

**bar3h** hiển thị các cột của ma trận  $m \times n$  như là  $m$  nhóm, mỗi nhóm có  $n$  bar dạng 3D nằm ngang.

Mặc định, mỗi phần tử của ma trận được biểu diễn bằng một bar.

```
>>y = [5 2 1
6 7 3
8 6 3
5 5 5
1 5 8];
>>bar(y)
```

Sau đó nhập lệnh **bar3(y)** ta có đồ thị 3D.

### 1.5.8.2 Xếp chồng đồ thị

Ta có thể xếp chồng số liệu trên đồ thị thanh bằng cách tạo ra một trục khác trên cùng một vị trí và như vậy ta có một trục y độc lập với bộ số liệu khác.

```
>>TCE = [515 420 370 250 135 120 60 20];
>>nhdo = [29 23 27 25 20 23 23 27];
>>ngay = 0:5:35;
>>bar(ngay,nhdo)
>>xlabel('Ngày')
>>ylabel('Nhiệt độ (^{o}C)')
```

Để xếp chồng một số liệu lên một đồ thị thanh ở trên, có trục thứ 2 ở cùng vị trí như trục thứ nhất ta viết :

```
>>h1 = gca;
```

và tạo trục thứ 2 ở vị trí trục thứ nhất trước nhất vẽ bộ số liệu thứ 2

```
>>h2 = axes('Position',get(h1,'Position'));
>>plot(days,TCE,'LineWidth',3)
```

Để trục thứ 2 không gây trở ngại cho trục thứ nhất ta viết :

```
>>set(h2,'YAxisLocation','right','Color','none','XTickLabel',[])
>>set(h2,'XLim',get(h1,'XLim'),'Layer','top')
```

Để ghi chú lên đồ thị ta viết:

```
>>text(11,380,'Mat do','Rotation',-55,'FontSize',16)
>>ylabel('TCE Mat do (PPM)')
>>title('Xếp chồng đồ thị','FontSize',16)
```

### 1.5.8.3 Đồ họa vùng

Hàm **area** hiển thị đường cong tạo từ một vector hay từ một cột của ma trận. Nó vẽ các giá trị của một cột của ma trận thành một đường cong riêng và tô đầy vùng không gian giữa các đường cong và trục x.

```
>>Y = [5 1 2
      8 3 7
      9 6 8
      5 5 5
      4 2 3];
>>area(Y)
```

### 1.5.8.4 Đồ thị pie

Đồ thị **pie** hiển thị theo tỉ lệ phần trăm của một phần tử của một vector hay một ma trận so với tổng các phần tử. **pie** và **pie3** tạo ra đồ thị 2D và 3D.

```
>>X = [19.3 22.1 51.6;
34.2 70.3 82.4;
61.4 82.9 90.8;
50.5 54.9 59.1;
29.4 36.3 47.0];

>>x = sum(X);

>>explode = zeros(size(x));

>>[c,offset] = max(x);

>>explode(offset) = 1;

>>h = pie(x,explode);

>>h = pie3(x,explode);
```

## 1.5.9 Đồ hoạ 3D

### 1.5.9.1 Các hàm cơ bản

Hàm **mesh** và **surf** tạo ra mặt 3D từ ma trận dữ liệu. Gọi ma trận dữ liệu là  $z$  mà mỗi phần tử của nó  $z(i,j)$  xác định tung độ của mặt thì  $\text{mesh}(z)$  tạo ra một lưới có màu thể hiện mặt  $z$  còn  $\text{surf}(z)$  tạo ra một mặt có màu  $z$ .

### 1.5.9.2 Đồ thị các hàm hai biến

Bước thứ nhất để thể hiện hàm 2 biến  $z=f(x,y)$  là tạo ma trận  $x$  và  $y$  chứa các tọa độ trong miền xác định của hàm. Hàm  $\text{meshgrid}$  sẽ biến đổi vùng xác định bởi 2 vector  $x$  và  $y$  thành ma trận  $x$  và  $y$ . Sau đó ta dùng ma trận này để đánh giá hàm.

```
>>[x,y] = meshgrid(-8:.5:8);

>>r = sqrt(x.^2 + y.^2);
```

Ma trận  $r$  chứa khoảng cách từ tâm của ma trận. Tiếp theo ta dùng hàm  $\text{mesh}$  để vẽ hàm.

```
>>z = sin(r)./r;

>>mesh(z)
```

### 1.5.10 Thực hành vẽ đồ thị

**Bài 1.10.** Vẽ đồ thị hàm số  $y_1 = \sin x \cdot \cos 2x$  và hàm số  $y_2 = \sin x^2$  trong  $[0-2\pi]$ , trên cùng hệ trục tọa độ, ta lần lượt thực hiện như sau:

```
>>x=0:0.01:2*pi;
>>y1=sin(x).*cos(2*x); %nhân tương ứng từng phần tử
>>plot(x,y1)
>>grid on %hiện thị lưới
```

Sau khi thu được đồ thị hàm  $y_1$ , để vẽ  $y_2$  trên cùng đồ thị, ta thực hiện:

```
>>hold on %giữ hình, mặc nhiên là hold off
>>y2=sin(x.^2); %lưu thừa từng phần tử
>>plot(x,y2,'k') %đường vẽ có màu đen
>>axis([0 4*pi -1.25 1.25]) %định lại tọa độ hiện thị
```

Ta có thể đặt nhãn cho các trục cũng như tiêu đề cho đồ thị:

```
>>xlabel('Time')
>>ylabel('Amplitude')
>>title('y1=sinx.cos2x and y2=sin(x^2)')
>>legend('sinx.cos2x','sinx^2')
```

**Bài 1.11.** Thực hiện như trên nhưng dùng các hàm semilogx, semilogy, loglog thay thế cho plot.

**Bài 1.12.** Thực hiện như trên cho hàm số  $y = \frac{e^{-x}}{2e^{-x}+2}$

**Bài 1.13.** Vẽ hàm số  $r = \sin(5\theta)$  trong tọa độ cực:

```
>>theta=0:0.05:2*pi;
>>r=sin(5*theta);
>>polar(theta,r)
```

**Bài 1.14.** Vẽ hàm số  $r = 2\sin(\theta) + 3\cos(\theta)$

**Bài 1.15.** Vẽ hàm số  $2x^2 + y^2 = 10$  ở dạng tọa độ cực.

Gợi ý:  $x = r\cos\theta$ ,  $y = r\sin\theta$

**Bài 1.16.** Dùng hàm text xuất các công thức sau ra trục tọa độ:

$$\int_{-\infty}^{\infty} \frac{e^{-x} + e^x}{\sqrt{2x+1}} \sin(\omega(x-\tau)) dx$$

$$\sum_{n=-1}^{\infty} z^{-n} x(n)$$

$$\ddot{y} = x \begin{bmatrix} x & x^2 & x^3 \\ x^2 & x^3 & x \\ x^3 & x & x^2 \end{bmatrix}$$

- MATLAB cung cấp nhiều hàm vẽ đồ thị 3D, chẳng hạn: **plot3** - dùng để vẽ các đường trong không gian 3 chiều; **mesh** và **surf** - dùng để vẽ vật thể 3D.

### Bài 1.17. Vẽ đồ thị 3D bằng hàm **plot3**:

```
>>t=0:pi/50:10*pi;

>>x=sin(t);

>>y=cos(t);

>>z=t;

>>subplot(121), plot3(x,y,z)

>>grid on

>>subplot(122), plot3(x,y,t.^2)

>> grid on
```

### Bài 1.18. Vẽ mặt paraboloid $z=x^2+y^2$ trong không gian 3 chiều:

```
>>close all

>>t=-5:0.1:5;

>> [x,y]=meshgrid(t); %dinh luoi ve

>>z=x.^2+y.^2;

>> subplot(2,2,1), mesh(z)

>> title('mesh(z)')

>> subplot(2,2,2), meshc(z)

>> title('meshc(z)')

>> subplot(2,2,3), meshz(z)

>> title('meshz(z)')

>> subplot(2,2,4), waterfall(z)
```

```
>> title('waterfall(z)')
```

**Bài 1.19.** Nhận xét về các hàm vẽ trên.

**Bài 1.20.** Vẽ mặt  $z = \frac{\sin(\sqrt{x^2+y^2})}{2(\sqrt{x^2+y^2})}$  dùng hàm **surf** và **mesh**.

## 1.6 CÁC FILE VÀ HÀM

### 1.6.1 Script file (file kịch bản)

Kịch bản là M-file đơn giản nhất, không có đối số. Nó dùng khi thi hành một loạt lệnh MATLAB theo một trình tự nhất định. Ta xét ví dụ tạo ra các số Fibonacci nhỏ hơn 1000.

```
f = [1 1];
i = 1;
while(f(i)+f(i+1))<1000
    f(i+2)= f(i) +f(i+1);
    i = i + 1;
end
plot(f)
```

Ta lưu đoạn mã lệnh này vào một file tên là *fibonacci.m*. Đây chính là một script file. Để thực hiện các mã chứa trong file fibonacci.m từ cửa sổ lệnh ta nhập:

```
>>fibonacci
```

Và nhấn enter. Kết quả MATLAB sẽ vẽ ra đồ thị của chuỗi Fibonacci.

### 1.6.2 File hàm

Hàm là M-file có chứa các đối số. Ta có một ví dụ về hàm:

```
function y = tb(x)

%Tính trị trung bình của các phần tử

[m,n] = size(x);

if m == 1
    m = n;
end

y = sum(x)/m;
```

Từ ví dụ trên ta thấy một hàm M-file gồm các phần cơ bản sau :

- Một dòng định nghĩa hàm: **function y = tb(x)** gồm từ khoá **function**, đối số trả về **y**, tên hàm **tb** và đối số vào **x**.

- Dòng kế tiếp là dòng trợ giúp đầu tiên. Vì đây là dòng văn bản nên nó phải đặt sau **%**. Nó xuất hiện khi ta nhập lệnh `help <tên hàm>`. Phần văn bản này giúp người dùng hiểu tác dụng của hàm.

- Thân hàm chứa mã MATLAB.

- Các lời giải thích dùng để cho chương trình rõ ràng. Nó được đặt sau dấu **%**.

Cần chú ý là tên hàm phải bắt đầu bằng ký tự và **cùng tên với file chứa hàm**. Tên hàm là **tb** thì tên file cũng là **tb.m**.

Từ cửa sổ MATLAB ta gõ lệnh:

```
>>z = 1:99;
```

```
>>tb(z)
```

Các biến khai báo trong một hàm của MATLAB là biến địa phương. Các hàm khác không nhìn thấy và sử dụng được biến này. Muốn các hàm khác dùng được biến nào đó của hàm ta cần khai báo nó là **global**.

Nếu hàm có nhiều thông số ngõ vào và ngõ ra thì khai báo như sau:

```
function [y1,y2,y3] = tb(x1,x2,x3)
```

Lưu ý rằng trong một file .m có thể có nhiều hàm nhưng hàm đầu tiên phải có tên trùng với tên file.

### 1.6.3 Các hàm toán học cơ bản

`exp(x)` hàm mũ cơ số e

`sqrt(x)` căn bậc hai của x

`log(x)` logarit cơ số e

`log10(x)` logarit cơ số 10

`abs(x)` module của số phức x (giá trị tuyệt đối của số thực)

`angle(x)` argument của số phức a

`conj(x)` số phức liên hợp của x

`imag(x)` phần ảo của x

`real(x)` phần thực của x



sign(x) dấu của x

cos(x) tính cos

sin(x) tính sin

tan(x) tính tang

acos(x) tính  $\cos^{-1}$

asin(x) tính  $\sin^{-1}$

atan(x) tính  $\tan^{-1}$

cosh(x) tính  $\frac{e^x + e^{-x}}{2}$

coth(x) tính  $\cosh(x)/\sinh(x)$

sinh(x) tính  $\frac{e^x - e^{-x}}{2}$

tanh(x) tính  $\sinh(x)/\cosh(x)$

acosh(x) tính  $\cosh^{-1}$

acoth(x) tính  $\coth^{-1}$

asinh(x) tính  $\sinh^{-1}$

atanh(x) tính  $\tanh^{-1}$

## 1.6.4 Các phép toán trên hàm toán học

### a. Biểu diễn hàm toán học

MATLAB biểu diễn các hàm toán học bằng cách dùng các biểu thức đặt trong M-file. Ví dụ để khảo sát hàm:

$$f(x) = \frac{2}{x+1} + \frac{1}{x^2+2} - 1$$

Ta tạo ra một file, đặt tên là **ab.m** có nội dung:

```
function y = ab(x)
y = 2./(x + 1) + 1./(x.^2 + 2) - 1 ;
```

Cách thứ hai để biểu diễn một hàm toán học trên dòng lệnh là tạo ra một đối tượng inline từ một biểu thức chuỗi. Ví dụ ta có thể nhập từ dòng lệnh như sau:

```
>>f = inline('2./(x + 1) + 1./(x.^2 + 2) - 1');
```

Ta có thể tính trị của hàm tại  $x = 2$  như sau:

```
>>f(2)
```

### b. Vẽ đồ thị của hàm

Hàm **fplot** vẽ đồ thị hàm toán học giữa các giá trị đã cho.

```
>>fplot(@ (x) [tan(x),sin(x),cos(x)], 2*pi*[-1 1 -1 1])
```

### c. Tìm cực tiểu của hàm

Cho một hàm toán học một biến, ta có thể dùng hàm **fminbnd** của MATLAB để tìm cực tiểu địa phương của hàm trong khoảng đã cho.

```
>>f=inline('1./((x-0.3).^2+0.01)+1./(x.^2+0.04)-6');
```

```
>>x = fminbnd(f,0.3,1)
```

Hàm **fminsearch** tương tự hàm **fminbnd** dùng để tìm cực tiểu địa phương của hàm nhiều biến.

Ta có hàm **three\_var.m**:

```
function b = three_var(v)

x = v(1);

y = v(2);

z = v(3);

b = x.^2 + 2.5*sin(y) - z^2*x^2*y^2;
```

Và bây giờ tìm cực tiểu đối với hàm này bắt đầu từ  $x = -0.6$ ,  $y = -1.2$ ;  $z = 0.135$

```
>>v = [-0.6 -1.2 0.135];
```

```
>>a = fminsearch('three_var',v)
```

### d. Tìm điểm zero

Hàm **fzero** dùng để tìm điểm không của hàm một biến. Ví dụ để tìm giá trị không của hàm lân cận giá trị -0.2, ta viết:

```
>>f=inline('1./((x-0.3).^2+0.01)+1./(x.^2+0.04)-6');
```

```
>>a = fzero(f,-0.2)
```

## 1.6.5 Thực hành trên script và function

### 1.6.5.1 Script

Tập hợp các dòng lệnh của MATLAB được sắp xếp theo một cấu trúc nào đó và lưu thành file có phần mở rộng \*.m được gọi là script file. Ta có thể chạy file này từ cửa sổ lệnh giống hệt như các lệnh của MATLAB. Cấu trúc của một script file như sau:

```
% Phần viết sau dấu '%' ở đây dùng cho lệnh help
```

```
% Thông thường phần này mô tả chức năng, cách sử dụng,
% ví dụ minh họa hay những lưu ý đặc biệt mà tác giả mong muốn
trợ
% giúp cho người sử dụng.
[global tênbiến1, tênbiến2,... ] % Khai báo biến toàn cục % (nếu
có)
<các câu lệnh> % phần trình bày câu lệnh
```

### Khởi động MATLAB Editor:

```
>>edit
```

### Tạo một script file có tên vd.m, với nội dung như sau:

```
% Doan script file nay hien thi loi chao trong 2s. Sau do hien
thi logo cua MATLAB roi thoat
close all
% ----- Tao mot cua so do hoa -----
figure('Color',[0 0 0],...
'Name','Welcome to MATLAB Experiments',...
'NumberTitle','off',...
'MenuBar','none');
% ----- Hien thi loi chao -----
text( 'String','Welcome to MATLAB',...
'Color',[.25 .25 .25],...
'Position',[0.01 .501],...
'FontSize',32,...
'FontAngle','italic');
text( 'String','Welcome to MATLAB',...
'Color','w',...
'Position',[0 .5],...
'FontSize',32,...
'FontAngle','italic');
axis off;
```

```

pause(2); % dung trong 2 giay

% ---- Hien thi logo cua MATLAB -----

logo

clear all

close

% ket thuc script file

```

Sau khi lưu file này, từ cửa sổ lệnh của MATLAB, nhập:

```
>>help vd
```

Để thi hành script file vừa soạn, nhập:

```
>>vd
```

### 1.6.5.2 Sử dụng các tool xây dựng sẵn

MATLAB hỗ trợ một thư viện hàm rất phong phú, xây dựng trên các giải thuật nhanh và có độ chính xác cao. Ngoài các hàm cơ bản của MATLAB, tập hợp các hàm dùng để giải quyết một ứng dụng chuyên biệt nào đó gọi là Toolbox, ví dụ: Xử lý số tín hiệu (Digital Signal Processing), Điều khiển tự động (Control), mạng neural (Neural networks), ...

```

help <ten toolbox> % chuc nang toolbox

>>help control % liet ke ham cua control toolbox

```

Ta có thể tìm kiếm các hàm liên quan bằng cách cung cấp cho hàm **lookfor** của MATLAB một từ khóa:

```

lookfor <tu khoa timkiem>

>>lookfor filter % tìm các hàm liên quan đến mạch lọc

```

### 1.6.5.3 Xây dựng hàm

Xây dựng hàm cũng được thực hiện tương tự như script file. Tuy nhiên, đối với hàm ta cần quan tâm đến các tham số truyền cho hàm và các kết quả trả về sau khi thực hiện. Có 3 điểm cần lưu ý:

- Tên hàm phải được đặt trùng với tên file lưu trữ.
- Phải có từ khóa **function** ở dòng đầu tiên.
- Trong một hàm có thể xây dựng nhiều hàm con (điều này không có trong script file).

```
function [out1,out2,...]=tenham(in1,in2,...)
```

```

% -----
% Hiện thị khi người sử dụng dùng lệnh help tenham
% -----
[global <tênbiến1, tênbiến2, ...>] %khai báo biến toàn cục (nếu
có)

<Các câu lệnh thực hiện hàm>

out1=kết quả 1 %kết quả trả về của hàm
out2=kết quả 2
...
% Các hàm con (nếu có)
function [subout1,subout2,...]=tenhamcon(subin1,subin2,...)
<Các câu lệnh của hàm con>

```

**Bài 1.21.** Xây dựng hàm ***gptb2*** để giải phương trình bậc hai  $ax^2+bx+c=0$ . Nội dung hàm như sau:

```

function [x1,x2]=gptb2(a,b,c)
% Giai phuong trinh bac hai ax^2+bx+c=0
% [x1,x2]=gptb2(a,b,c)
% Trong do: x1,x2 la nghiệm
% a, b, c la 3 he so cua phuong trinh
if nargin<3
error('Error! Nhap 3 he so cua phuong trinh')
elseif a==0
x1=-c/b;
x2=[];
else
delta = b^2 - 4*a*c;
x1 = (-b+sqrt(delta))/(2*a);
x2 = (-b-sqrt(delta))/(2*a);
end

```

Lưu tên file là ***gptb2.m*** và kiểm tra kết quả:

```
>>help gptb2
>>[x1,x2]=gptb2(1,6,-7)
>>[x1,x2]=gptb2(2,7,14)
>>[x1,x2]=gptb2(0,4,3)
>>[x1,x2]=gptb2(1,6)
```

**Bài 1.22.** Cho biết ý nghĩa của từ khóa nargin?

**Bài 1.23.** Viết lại hàm này để kết quả chỉ trả về nghiệm số thực.

**Bài 1.24.** Xây dựng hàm **vdcongdb(a,m,method)** để vẽ một số đường cong trong hệ tọa độ cực, với a là bán kính và m là số đường cong vẽ trên cùng trục tọa độ. Trường hợp này hàm không trả về giá trị.

Tuỳ theo giá trị của tham số 'method' mà ta vẽ đồ thị tương ứng:

Nếu method = 'Becnulli': Vẽ đường Becnulli:  $r = a\sqrt{|2\cos 2\theta|}$

Nếu method = 'Astroit': Vẽ đường Astroit:  $r = a\sqrt{\left|1 - \frac{\sin 3\theta}{4}\right|}$

Nếu method = 'Xoanoc': Vẽ đường xoắn ốc:  $r = \cos\theta + 1$

Nội dung hàm như sau:

```
function vdcongdb(a,m,method)
% Ve duong cong trong toa do cuc: vdcongdb(a,m,method)
% method = 'Becnulli' - Ve duong Lemniscat Becnulli:
% r=a*sqrt(abs(2*cos(2*theta)))
% 'Astroit' - Ve duong Astroit:
% r=a*sqrt(abs(1-sin(3*theta)/4))
% 'Xoanoc' - Ve duong xoan oc:
% r=a*cos(theta)+1
% Voi: a-ban kinh; m-so duong cong ve tren cung he truc
if nargin<3
error('Vui long nhap du 3 thong so cua ham')
else
theta=0:0.01:2*pi; method=upper(method);
switch method
```

```

case 'BECNULLI'

r=a*sqrt(abs(2*cos(2*theta)));

case 'ASTROIT'

r=a*sqrt(abs(1-sin(3*theta)/4));

case 'XOANOC'

r=a*cos(theta)+1;

otherwise

error('Chon: ''Becnuli'', ''Astroit'' hoac ''Xoanoc''')

end % end of switch

% ve do thi

close all; figure('Color','w');

for k=1:m

hold on

r1=r*k;

mau=[rand(1,1) rand(1,1) rand(1,1)];

h=polar(theta,r1);

set(h,'color',mau,'LineWidth',2);

axis equal;

end % end of for

hold off;

axis off

end % end of if

```

Kiểm tra lại hoạt động của hàm, ví dụ:

```

>>help vdcongdb

>>vdcongdb(1,5,'Becnuli')

>>vdcongdb(1,5,'Astroit')

>>vdcongdb(1,5,'Xoanoc')

>> vdcongdb(1,5,'saikieu')

>> vdcongdb(5,'becnulli')

```

**Bài 1.25.** Xây dựng hàm **dudoan()** để dự đoán kết quả sau mỗi lần tung một xúc xắc đồng nhất, 6 mặt. Nội dung hàm như sau:

```
function dudoan()

% Du doan ket qua sau moi lan tung ngau nhien mot xuc xac 6 mat
% Chuong trinh lap lai cho den khi nguoi su dung khong doan tiep
%

tiep = 'y'; sai=0; dung=0;

disp('Chao mung ban den voi chuong trinh du doan xuc xac!')

while(lower(tiep)=='y')

doan=input('Moi ban du doan ket qua (1-6):');

kqua=tungxx;

if (doan ~= kqua)

disp('Xin loi, ban da doan sai!')

sai=sai+1;

else

disp('Xin chuc mung!')

dung=dung+1;

end

tiep=input('Ban muon choi tiep(''y''/''n''):');

end

disp(['Dung ' num2str(dung) ' trong tong so ' num2str(sai+dung)
' lan doan'])

% subfunction -----

function mat = tungxx()

mat=floor(6*rand(1,1))+1;

% end
```

Kết luận về sự khác nhau giữa script file và hàm không có tham số vào.

**Bài 1.26.** Viết chương trình in tam giác Pascal  $n$  dòng trong màn hình đồ họa với  $n$  được nhập từ bàn phím.



# BÀI 2: TÍN HIỆU RỜI RẠC THEO THỜI GIAN

Sau khi học xong bài này, người học có thể:

- Biết được các tín hiệu sơ cấp.
- Thực hiện các phép toán đơn giản.
- Tính năng lượng của tín hiệu.
- Xác định các tính chất của hệ rời rạc.

## 2.1 CÁC TÍN HIỆU SƠ CẤP

Hàm xung đơn vị:

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

```
>>n = -10:10;
```

```
>>delta=zeros(1,10) 1 zeros(1,10);
```

```
>>stem(n,delta);
```

**Bài 2.1.** Viết chương trình và vẽ dạng tín hiệu hàm bước đơn vị  $u(n)$ .

Hàm bước đơn vị:

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

**Bài 2.2.** Viết chương trình và vẽ dạng tín hiệu hàm mũ  $(1/2)^n u(n)$ ,  $3^n u(n)$ .

Hàm mũ:

$$x(n) = \begin{cases} a^n & n \geq 0 \\ 0 & n < 0 \end{cases} = a^n u(n)$$

**Bài 2.3.** Tính năng lượng tín hiệu  $x(n) = (1/2)^n u(n)$  trong khoảng  $(-10,10)$ ;  $(0,1000)$ ;  $(0,1e6)$ .

**Bài 2.4.** Tính năng lượng và công suất tín hiệu  $x(n) = (1 - 2^n)u(n)$  trong khoảng  $(-10, 10)$ ;  $(0, 1000)$ ;  $(0, 1e6)$ .

Năng lượng của tín hiệu trong khoảng  $[-N, N]$ :

$$E_N = \sum_{n=-N}^N |x(n)|^2$$

Năng lượng tín hiệu:

$$E = \lim_{N \rightarrow \infty} E_N$$

Công suất trung bình của tín hiệu:

$$P = \lim_{N \rightarrow \infty} \frac{E_N}{2N + 1}$$

## 2.2 CÁC PHÉP TOÁN

- Dịch:  $x(n) \rightarrow x(n - k)$
- Ảnh gương:  $x(n) \rightarrow x(-n)$
- Co trên miền thời gian:  $x(n) \rightarrow x(\mu n)$
- Cộng:  $y(n) = x_1(n) + x_2(n)$
- Nhân:  $y(n) = x_1(n)x_2(n)$
- Co biên độ:  $y(n) = Ax(n)$

Dịch phải xung đơn vị 5 mẫu:

```
>>delta=zeros(1,15) 1 zeros(1,5)];
>>stem(n,delta);
```

**Bài 2.5.** Viết chương trình và vẽ dạng tín hiệu hàm  $u(n - 3)$ ,  $u(n + 2)$ .

**Bài 2.6.** Viết chương trình và vẽ dạng tín hiệu hàm  $x(n) = 2u(n - 3) + \delta(n - 2)$  trong khoảng  $(-10, 10)$ . Từ đó vẽ các tín hiệu  $x(-n)$ ,  $2x(n)$ ,  $x(2n)$ .

**Bài 2.7.** Viết function thực hiện cộng và nhân 2 tín hiệu. Ví dụ thực hiện cho 2 tín hiệu  $x_1(n) = \{1, -1, 2, 3, -2\}$  và  $x_2(n) = \{-2, -2, 1, 1, -4\}$ .

↑

↑

## 2.3 KIỂM TRA TÍNH CHẤT TUYẾN TÍNH VÀ BẤT BIẾN

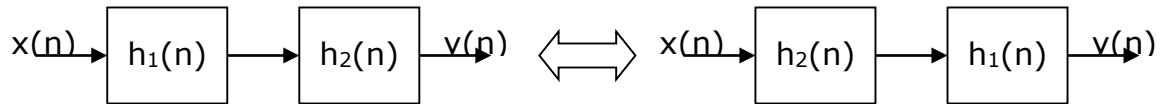
Hệ thống H bất biến theo thời gian nếu và chỉ nếu:

$$y(n) = H[x(n)] \rightarrow y(n - k) = H[x(n - k)]$$

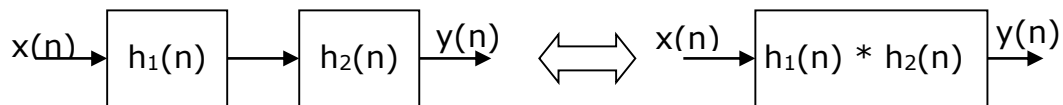
Hệ thống là tuyến tính nếu và chỉ nếu:

$$H[a_1x_1(n) + a_2x_2(n)] = a_1H[x_1(n)] + a_2H[x_2(n)]$$

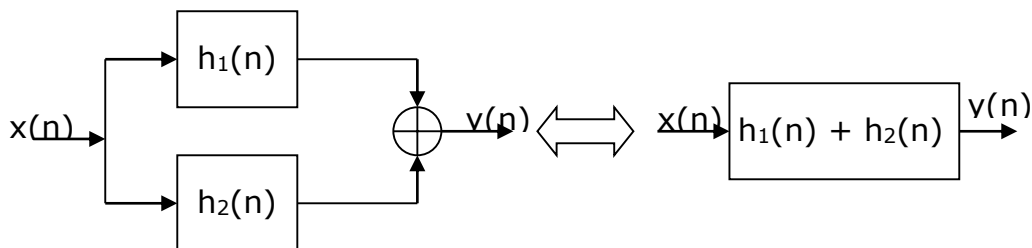
Tính giao hoán:



Tính kết hợp:



Tính phân phối:



**Bài 2.8.** Xét hệ thống  $y(n) = nx(n)$ .

```
>>n = -10:10;

>>x = randn(size(n)); %Tín hiệu x ngẫu nhiên

>>y = n.*x; %y(n) = nx(n)

>>ynk = [0 0 0 0 y]; %Dịch phải y(n) 4 mẫu -> y(n - 4)

>>x1 = [0 0 0 0 x]; %Dịch phải x(n) 4 mẫu

>>n1 = [n 11:14]; % Bỏ sung thêm giá trị cho n

>>yn = n1.*x1; % yn = H[x(n - 4)]

>>subplot(211), stem(n1,ynk), title('y(n - k)');

>>subplot(212), stem(n1,yn), title('H[x(n - k)]');
```

Kết luận về tính chất bất biến theo thời gian của  $y(n) = nx(n)$ .

**Bài 2.9.** Xác định tính chất bất biến theo thời gian của hệ thống có phương trình  $y(n) = x(-n)$  và  $y(n) = x(n)\cos(0.5n)$ .

**Bài 2.10.**

```

>>clear all

>>clf

>>n = -10:10;

>>x1 = randn(size(n)); %Tín hiệu x1 ngẫu nhiên
>>x2 = randn(size(n)); %Tín hiệu x2 ngẫu nhiên
>>a1 = 3; a2 = -2; %a1, a2 tùy ý
>>y1 = n.*x1;
>>y2 = n.*x2;
>>y = n.*(a1*x1 + a2*x2);
>>subplot(211), stem(n,a1*y1+a2*y2);
>>title('a_1y_1(n)+a_2y_2(n)');
>>subplot(212), stem(n,y);
>>title('H[a_1x_1(n)+a_2x_2(n)]');

```

Kết luận về tính chất tuyến tính của  $y(n) = nx(n)$ .

**Bài 2.11.** Xác định tính chất tuyến tính của hệ thống có phương trình  $y(n) = x^2(n)$  và  $y(n) = x(n^2)$

## 2.4 Hệ LTI

Phương trình sai phân:

$$y(n) = -\sum_{k=1}^N a_k y(n-k) + \sum_{k=0}^M b_k x(n-k)$$

**Bài 2.12.** Xét hệ thống có phương trình sai phân:  $y(n) = 0.3x(n) + 0.2x(n-1) - 0.3x(n-2) - 0.9y(n-1) + 0.9y(n-2)$ . Xác định đáp ứng xung đơn vị của hệ thống.

```

>>N = 40;

>>num = [0.3 0.2 -0.3];
>>den = [1 0.9 -0.9];

>>h = impz(num,den,N); % N: số lượng mẫu tính toán
>>stem(h);

```

Xác định ngõ ra khi biết đáp ứng xung và ngõ vào:

```

>>x = randn(1,10);

```

```
>>y = conv(x,h);
>>subplot(311),stem(x);
>>subplot(312),stem(h);
>>subplot(313),stem(y);
```

**Bài 2.13.** Kiểm tra tính giao hoán và kết hợp:

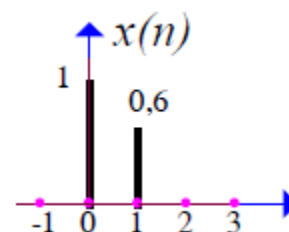
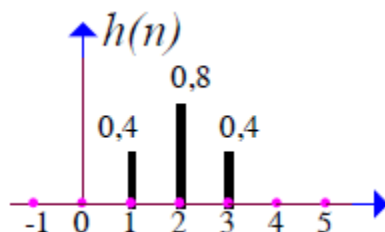
```
>>h1 = [1 2 -2 -3]; Hệ thống 1
>>h2 = [-2 0 3 1]; Hệ thống 2
>>h = conv(h1,h2);
>>N = 30;
>>x = randn(1,N);
>>y11 = conv(x,h1);
>>y1 = conv(y11,h2);
>>y21 = conv(x,h2);
>>y2 = conv(y21,h1);
>>y = conv(x,h);
>>subplot(311),stem(y1);
>>title('y(n) = (x*h_1(n))*h_2(n)');
>>subplot(312),stem(y2);
>>title('y(n) = (x*h_2(n))*h_1(n)');
>>subplot(313),stem(y);
>>title('y(n) = x*(h_1(n)*h_2(n))');
```

**Bài 2.14.** Kiểm tra tính giao hoán và kết hợp của hai hệ thống ghép liên tiếp sau:

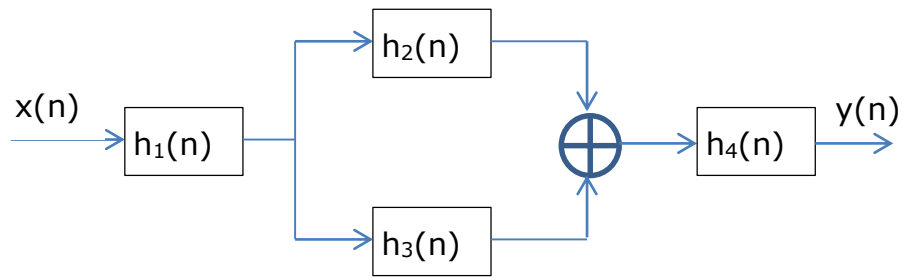
Hệ thống 1:  $y(n) = 2x(n) - 0.5x(n-1) + 0.5x(n-3) + 0.1y(n-1)$

Hệ thống 2:  $y(n) = 0.3x(n) + 0.2x(n-2) - 0.1y(n-2)$

**Bài 2.15.** Xác định ngõ ra của hệ thống sau:



**Bài 2.16.** Xác định đáp ứng xung tương đương của hệ thống sau:



$$h_1 = \{1, 0, -1, 3\}; \quad h_2 = \{2, -2, 1\}; \quad h_3 = \{3, 4, -1, 1\}; \quad h_4 = \{-3, 5, 6, -1, 1\}$$

$\uparrow \qquad \qquad \qquad \uparrow \qquad \qquad \qquad \uparrow \qquad \qquad \qquad \uparrow$

Viết chương trình xác định ngõ ra của hệ thống khi ngõ vào là  $x(n) = (-2)^n u(n)$  (tính toán cho giá trị  $n$  từ -20 đến 20):

- Dùng theo sơ đồ trên.
- Dùng đáp ứng xung tương đương.

# BÀI 3: BIẾN ĐỔI Z

Sau khi học xong bài này, người học có thể:

- Biết cách chuyển đổi tín hiệu trên miền thời gian rời rạc sang miền  $z$ .
- Biết các tính chất của biến đổi  $z$ .
- Biểu diễn hàm hệ thống của LTI có quan hệ vào – ra là phương trình sai phân hệ số hằng bằng biến đổi  $z$  hữu tỉ.

## 3.1 CÁC ĐIỂM CỰC VÀ ĐIỂM KHÔNG

Biến đổi  $z$  của tín hiệu rời rạc  $x(n)$ :

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}$$

$X(z)$  là hàm hữu tỉ:

$$X(z) = \frac{N(z)}{D(z)} = \frac{\sum_{k=0}^M b_k z^{-k}}{\sum_{k=0}^N a_k z^{-k}}$$

Giả sử  $a_0 \neq 0$  và  $b_0 \neq 0$ :

$$X(z) = \frac{N(z)}{D(z)} = \frac{b_0 z^{-M} z^M + \frac{b_1}{b_0} z^{M-1} + \dots + \frac{b_M}{b_0}}{a_0 z^{-N} z^N + \frac{a_1}{a_0} z^{N-1} + \dots + \frac{a_N}{a_0}}$$

Do  $N(z)$  và  $D(z)$  là các đa thức theo  $z$  nên có thể biểu diễn như sau:

$$X(z) = G z^{N-M} \frac{\prod_{k=1}^M (z - z_k)}{\prod_{k=1}^N (z - p_k)}$$

Để biểu diễn trên đồ thị, điểm cực được đánh dấu bằng  $x$  và điểm không được đánh dấu bằng  $o$ .

**Bài 3.1.** Xác định điểm cực và không dựa vào hàm  $z$ -plane:

```
>>num = [1 2 3]; % Tử số
>>den = [2 4 7]; % Mẫu số
>>zplane(num,den);
```

Ta có thể vẽ các điểm cực và điểm không nếu đã biết điểm cực và điểm không bằng cách đưa thông số vào hàm `zplane` ở dạng vector cột:

```
>>zero = [-1 1+j*1];
>>pole = [j*2 -1+j];
>>zplane(zero',pole');
```

Để xác định điểm cực và không, ta dùng hàm **tf2zp**: `[z,p,k] = tf2zp(num,den)` trong đó `z`, `p` là các điểm cực và không lưu dạng vector hàng, `k` là hệ số khuếch đại:

```
>>num = [1 2 3]; % Tử số
>>den = [2 4 7]; % Mẫu số
>>[z,p,k] = tf2zp(num,den)
```

Nếu đã cho điểm cực và điểm không, ta có thể xác định lại biểu thức của biến đổi `z` bằng hàm **zp2tf**: `[num,den] = zp2tf(z,p,k)` (`z`, `p` ở dạng vector cột)

```
>>zero = [-1 1+j*1];
>>pole = [j*2 -1+j];
>>k = 2;
>>[num,den] = zp2tf(zero',pole',k)
```

**Bài 3.2.** Xác định và vẽ điểm cực, điểm không của các hàm hệ thống sau:

$$H(z) = \frac{2 + 5z^{-1} + 4z^{-2} + 5z^{-3} + 3z^{-4}}{5 + 45z^{-1} + 2z^{-2} + z^{-3} + z^{-4}}$$

Từ đó xác định các miền hội tụ có thể có. So sánh với lý thuyết.

**Bài 3.3.** Xác định biểu thức của biến đổi `z` có các điểm cực 0.5; 0.75; 1+j0.5; 1-j0.5 và các điểm không 0.3; 0.1; 2-j2; 2+j2 với hệ số khuếch đại `k` = 0.7

## 3.2 PHÂN TÍCH DÙNG PHƯƠNG PHÁP THẶNG DƯ

Phân tích thành các thừa số theo phương pháp thặng dư:

$$\frac{N_1(z)}{D(z)} = \frac{A_1}{1 - p_1 z^{-1}} + \dots + \frac{A_N}{1 - p_N z^{-1}}$$

**Bài 3.4.** Xác định các hệ số của biểu thức biến đổi `z` bằng hàm **residuez**:

```
>>num = [1 2 3]; % Tử số
>>den = [2 4 7]; % Mẫu số
```



```
>>[A,p,k] = residuez(num,den);
```

Ta cũng có thể dùng hàm **residuez** để xác định lại tử số và mẫu số:

```
>>[num,den] = residuez(A,p,k);
```

Ghi lại công thức biến đổi và so sánh với kết quả ban đầu.

**Bài 3.5.** Phân tích biểu thức sau dùng phương pháp thẳng dư:

$$H(z) = \frac{1 - 4.2z^{-1} + 0.8z^{-2}}{1 - 2.5z^{-1} + 3z^{-2} - z^{-3}}$$

$$G(z) = \frac{1 + z^{-1}}{(1 + 2z^{-1})^2(1 - z^{-1})}$$

Tính toán lại kết quả theo lý thuyết.

**Bài 3.6.** Cho hệ thống có phương trình vào / ra là phương trình sai phân hệ số hằng:  $y(n) = x(n) - 2x(n-2) + 0.81y(n-1)$ . Xác định  $H(z)$ , từ đó viết chương trình:

- Xác định và vẽ các điểm cực, không.
- Phân tích dùng phương pháp thẳng dư.

### 3.3 BIẾN ĐỔI Z VÀ Z NGƯỢC

Xét hệ LTI biểu diễn bằng phương trình sai phân hệ số hằng:

$$y(n) = - \sum_{k=1}^N a_k y(n-k) + \sum_{k=0}^M b_k x(n-k)$$

Hàm hệ thống của hệ LTI biểu diễn bằng phương trình sai phân hệ số hằng:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^M b_k z^{-k}}{1 + \sum_{k=1}^N a_k z^{-k}}$$

Biến đổi z ngược:

$$X(z) = \frac{N(z)}{D(z)} = \sum_{k=0}^{M-N} c_k z^{-k} + \frac{N_1(z)}{D(z)}$$

(Nếu bậc của tử số nhỏ hơn bậc của mẫu số)

$$\sum_{k=0}^{M-N} c_k z^{-k} \stackrel{z}{\leftrightarrow} \sum_{k=0}^{M-N} c_k \delta(n-k)$$

Để tính thành phần còn lại, ta phân tích thành các thừa số theo phương pháp thẳng dư:

$$\frac{N_1(z)}{D(z)} = \frac{A_1}{1 - p_1 z^{-1}} + \dots + \frac{A_N}{1 - p_N z^{-1}}$$

Nếu các giá trị  $p_j = \dots = p_m$  thì chuyển các số hạng từ  $A_j$  đến  $A_m$  thành:

$$\frac{A_j}{1 - p_N z^{-1}} + \frac{A_{j+1}}{(1 - p_N z^{-1})^2} + \dots + \frac{A_{j+m-1}}{(1 - p_N z^{-1})^m}$$

Áp dụng kết quả:

$$\frac{A}{1 - az^{-1}} \xleftrightarrow{z} Aa^n u(n), ROC: |z| > |a|$$

**Bài 3.7.** Dùng hàm **ztrans** để biến đổi z ở dạng công thức:

```
>>syms n x
>>x = 2^n;
>>ztrans(x)
>>x = (-1/2)^n;
>>ztrans(x)
```

**Bài 3.8.** Xác định biến đổi z của các hàm sau:

- $x(n) = (-2)^{n-1}u(n)$
- $x(n) = n3^n u(n)$
- $x(n) = n^2 4^n u(n)$

**Bài 3.9.** Biến đổi z ngược theo giá trị bằng hàm impz.

```
>>num = [1 1 2];
>>den = [1 -1 2];
>>L = 50; %Số lượng mẫu cần tính
>>x = impz(num,den,L) % x là biến đổi z ngược
>>impz(num,den,L); % Vẽ trên đồ thị
```

**Bài 3.10.** Xác định và vẽ 100 mẫu đầu tiên của biến đổi z ngược của hàm:

$$X(z) = \frac{0.9 + 0.7z^{-1} + 0.1z^{-2} - z^{-3} + 0.5z^{-4}}{1 + 0.5z^{-1} - 0.2z^{-3} + 2z^{-4} + z^{-5}}$$

**Bài 3.11.** Thực hiện lại bài 3.10 với kết quả ở dạng công thức.

Để xác định biến đổi z ngược ở dạng công thức, ta dùng hàm **residuez** để phân tích thành dạng biểu thức đơn giản và dùng các kết quả biến đổi ngược để xác định.

**Bài 3.12.** Ta cũng có thể xác định biến đổi z ngược bằng cách dùng hàm iztrans.

```
>>syms F z  
>>F = 2*z^(-1)/(1-3*z^(-1));  
>>iztrans(F)
```

**Bài 3.13.** Xác định biến đổi z ngược của các hàm sau:

$$X(z) = \frac{z^{-1} + z^{-2}}{\left(1 + \frac{1}{2}z^{-1}\right)^2 (1 + 2z^{-1})}$$

$$Y(z) = \frac{2z^{-1}}{(3 + 4z^{-1} + z^{-2})(1 + z^{-1})}$$

# BÀI 4: BIẾN ĐỔI FOURIER RỜI RẠC

Sau khi học xong bài này, người học có thể:

- Chuyển tín hiệu rời rạc trên miền thời gian sang miền tần số bằng cách dùng biến đổi Fourier rời rạc.
- Tính toán biến đổi Fourier bằng một số thuật toán.

## 4.1 TÍNH DTFT

Biến đổi Fourier rời rạc (DTFT) mô tả như sau:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-jn\omega} = |X(e^{j\omega})|e^{j\Phi(\omega)}$$

**Bài 4.1.** Tính và vẽ DTFT có dạng:

$$X(e^{-j\omega}) = \frac{1 + e^{-j\omega} - 2e^{-j2\omega}}{1 + 0.5e^{-j\omega}}$$

```
>>w = linspace(-4*pi,4*pi,512); % Tạo 512 giá trị từ -4π đến 4π
>>num = [1 1 -2];
>>den = [1 0.5];
>>h = freqz(num,den,w);
>>subplot(211),plot(w/pi,abs(h));
>>xlabel('\omega/\pi');
>>ylabel('Bien do');
>>title('Pho bien do |X(e^{j\omega})|');
>>subplot(212),plot(w/pi,angle(h));
>>xlabel('\omega/\pi');
>>ylabel('Pha [rad]');
```

```
>>title('Pho pha arg(X(e^{j\omega}))');
```

Để khảo sát chuỗi  $x(n)$  hữu hạn, ta cho thông số thứ 2 của hàm **freqz** là 1.

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

```
>>h = freqz(x,1,w);
```

**Bài 4.2.** Tính và vẽ DTFT trong khoảng  $[-\pi, \pi]$ :

$$X(e^{-j\omega}) = \frac{0.8 + 3e^{-j\omega} - 0.2e^{-j2\omega} + 0.1e^{-j3\omega}}{1 + 0.1e^{-j\omega} - 0.4e^{-j2\omega} + 0.7e^{-j3\omega}}$$

**Bài 4.3.** Khảo sát DTFT của  $x(n) = [1 \ -2 \ 2 \ -3 \ 3 \ 4 \ 0 \ -1]$  trong khoảng  $[-\pi, \pi]$

Tính chất dịch thời gian:

$$x(n-m) \xleftrightarrow{DTFT} e^{-j\omega m} X(e^{j\omega})$$

**Bài 4.4.** Khảo sát tính chất dịch thời gian: Tính và vẽ DTFT trong khoảng  $[-\pi, \pi]$  của  $x(n-3)$  với  $x(n)$  cho như bài 4.3.

Tính chất dịch tần số:

$$e^{-j\omega_0 n} x(n) \xleftrightarrow{DTFT} X(e^{j(\omega-\omega_0)})$$

**Bài 4.5.** Khảo sát tính chất dịch tần số: Tính và vẽ DTFT trong khoảng  $[-\pi, \pi]$  của  $x(n)e^{-j3n}$  với  $x(n)$  cho như bài 4.3.

Tính chất đảo thời gian:

$$x(-n) \xleftrightarrow{DTFT} X(e^{-j\omega})$$

**Bài 4.6.** Khảo sát tính chất đảo thời gian: Tính và vẽ DTFT trong khoảng  $[-\pi, \pi]$  của  $x(-n)$  với  $x(n)$  cho như bài 4.3. Dùng hàm **fliplr** để chuyển  $x(n)$  thành  $x(-n)$  và nhân thêm hệ số  $e^{j\omega(L-1)}$  khi biến đổi.

**Bài 4.7.** Thực hiện lại từ 4.3 đến 4.6 với  $x(n)$  cho như bài 4.2.

## 4.2 FFT VÀ CÁC TÍNH CHẤT

**Bài 4.8.** Dùng hàm **fft** và **ifft** để tính DFT và IDFT của  $x(n)$ :

```
>>N = 32;
```

```
>>x = randn(1,N);
```

```
>>y = fft(x,N);
```

```
>>x1 = ifft(y,N);
```

```

>>subplot(321),stem(abs(x));

>>title('x(n)');xlabel('n');

>>subplot(323),stem(abs(y));

>>title(['FFT(n,' N ')']);xlabel('k');ylabel('Bien do');

>>subplot(324),stem(angle(y));

>>title(['FFT(n,' N ')']); xlabel('k');ylabel('Pha');

>>subplot(325),stem(abs(x1));

>>title(['IFFT(FFT(n,' N '))']);

>>xlabel('k');ylabel('Bien do');

>>subplot(326),stem(angle(x1));

>>title(['IFFT(FFT(n,' N '))']);

>>xlabel('k');ylabel('Pha');

```

**Bài 4.9.** Xác định và vẽ FFT 16 điểm của  $x(n)$  cho tùy ý.

**Bài 4.10.** Tạo function ***cshift*** để dịch vòng một chuỗi  $m$  giá trị:

```

function out = cshift(x,m)

m0 = m;

if abs(m0) > length(x)

    m0 = rem(m0,length(x));

end

while (m0<0)

    m0 = m0 + length(x);

end

out= [x(length(x)-m0+1:length(x)) x(1:length(x)-m0)];

```

Khảo sát tính chất dịch vòng của DFT-N điểm:

```

>>N = 20;

>>m = 3;

>>x = randn(1,N);

>>y = fft(x,N);

>>x1 = cshift(x,m); %Có thể thay bằng hàm circshift như sau: x1
= circshift(x',m)';

```

```
>>y1 = fft(x1,N);  
>>k = 0:N-1;  
>>y2 = exp(-j*2*pi*k*m/N).*y;  
>>subplot(221),stem(abs(y));title('y1');  
>>subplot(222),stem(angle(y)); title('y1');  
>>subplot(223),stem(abs(y2));title('y2');  
>>subplot(224),stem(angle(y2)); title('y2');
```

**Bài 4.11.** Viết chương trình khảo sát tính chất dịch vòng trên miền tần số.

**Bài 4.12.** Viết chương trình khảo sát tính chất chập vòng. Dùng hàm `cconv` để tính tích chập vòng.

**Bài 4.13.** Viết chương trình khảo sát tính chất đảo trên miền thời gian.

# BÀI 5: BỘ LỌC SỐ FIR

Sau khi học xong bài này, người học có thể:

- Thiết kế một bộ lọc số FIR dựa theo các thông số cho trước.

## 5.1 CÁC LOẠI BỘ LỌC

Điều kiện đối xứng và phản đối xứng của mạch lọc:

$$h(n) = \pm h(M - 1 - n)$$

Dựa trên tính chất đối xứng hay phản đối xứng của chuỗi đáp ứng xung và chiều dài  $N$  của chuỗi đáp ứng xung, người ta phân loại bộ lọc FIR làm 4 loại sau:

- Bộ lọc FIR loại 1:  $h(n)$  đối xứng,  $M$  lẻ,  $\beta = 0$ ,  $\alpha = (M - 1)/2$
- Bộ lọc FIR loại 2:  $h(n)$  đối xứng,  $M$  chẵn,  $\beta = 0$ ,  $\alpha = (M - 1)/2$
- Bộ lọc FIR loại 3:  $h(n)$  phản đối xứng,  $M$  lẻ,  $\beta = \pi/2$ ,  $\alpha = (M - 1)/2$
- Bộ lọc FIR loại 4:  $h(n)$  phản đối xứng,  $M$  chẵn,  $\beta = \pi/2$ ,  $\alpha = (M - 1)/2$

Đáp ứng tần số của FIR cho từng loại:

- Loại 1:

$$H(\omega) = \left[ \sum_{n=0}^{\frac{M-1}{2}} a(n) \cos \omega n \right] e^{-j \frac{(M-1)}{2} \omega} \text{ với } \begin{cases} a(0) = h\left(\frac{M-1}{2}\right) \\ a(n) = 2h\left(\frac{M-1}{2} - n\right) \text{ với } 1 \leq n \leq \frac{M-1}{2} \end{cases}$$

- Loại 2:

$$H(\omega) = \left[ \sum_{n=1}^{\frac{M}{2}} b(n) \cos \omega \left( n - \frac{1}{2} \right) \right] e^{-j \frac{(M-1)}{2} \omega} \text{ với } b(n) = 2h\left(\frac{M}{2} - n\right) \text{ với } 1 \leq n \leq \frac{M}{2}$$

- Loại 3:

$$H(\omega) = \left[ \sum_{n=1}^{\frac{M-1}{2}} c(n) \sin \omega n \right] e^{-j \left( \frac{\pi}{2} - \frac{(M-1)}{2} \omega \right)} \text{ với } c(n) = 2h\left(\frac{M-1}{2} - n\right) \text{ với } 1 \leq n \leq \frac{M-1}{2}$$

- Loại 4:

$$H(\omega) = \left[ \sum_{n=1}^{\frac{M}{2}} d(n) \sin \omega \left( n - \frac{1}{2} \right) \right] e^{-j \left( \frac{\pi}{2} - \frac{(M-1)}{2} \omega \right)} \text{ với } d(n) = 2h\left(\frac{M}{2} - n\right) \text{ với } 1 \leq n \leq \frac{M}{2}$$



**Bài 5.1.** Xác định đáp ứng tần số của bộ lọc FIR loại 1 từ chuỗi đáp ứng xung. Tạo function `FIR_t1` như sau:

```
function [a,w,L,Hr] = FIR_t1(h)

M = length(h);

L = (M-1)/2;

a = [h(L+1) 2*h(L:-1:1)];

n = [0:1:L];

w = linspace(0,2*pi,100)';

Hr = cos(w*n)*a';

stem(Hr);
```

Lưu file với tên ***FIR\_t1.m***. Thực hiện tính toán với đáp ứng xung  $h_1 = [1.5 \ -2.5 \ 3 \ -2.5 \ 1.5]$ :

```
>>h1 = [1.5 -2.5 3 -2.5 1.5];

>> [a,w,L,Hr]=FIR_t1(h1);
```

**Bài 5.2.** Xác định đáp ứng tần số cho bộ lọc FIR loại 2: Viết function ***FIR\_t2***. Thực hiện tính toán với đáp ứng xung  $h_2 = [1.5 \ -2.5 \ 3 \ 3 \ -2.5 \ 1.5]$ .

**Bài 5.3.** Xác định đáp ứng tần số cho bộ lọc FIR loại 3: Viết function ***FIR\_t3***. Thực hiện tính toán với đáp ứng xung  $h_3 = [1.5 \ -2.5 \ 3 \ 2.5 \ -1.5]$ .

**Bài 5.4.** Xác định đáp ứng tần số cho bộ lọc FIR loại 4: Viết function ***FIR\_t4***. Thực hiện tính toán với đáp ứng xung  $h_4 = [1.5 \ -2.5 \ 3 \ -3 \ 2.5 \ -1.5]$ .

**Bài 5.5.** Cho đáp ứng xung của bộ lọc FIR như sau:  $h = [-1 \ 2 \ 1.3 \ -2.2 \ 0.6 \ 3 \ 0.6 \ -2.2 \ 1.3 \ 2 \ -1]$ . Đáp ứng xung  $h(n)$  đối xứng với  $M$  lẻ nên đây là bộ lọc FIR loại 1.

a. Biểu diễn đáp ứng xung:

```
>>h = [-1 2 1.3 -2.2 0.6 3 0.6 -2.2 1.3 2 -1];

>>M = length(h);

>>n = 0:M-1;

>>subplot(221); stem(n,h);

>>title('Đáp ứng xung');xlabel('n');ylabel('h(n)');
```

b. Các hệ số của bộ lọc:

```
>>[a,w,L,Hr] = FIR_t1(h);
>>subplot(222);stem(0:L,a);
>>title('Cac he so a(n)');xlabel('n');ylabel('a(n)');
```

c. Đáp ứng tần số:

```
>>w = linspace(0,2*pi,100)';
>>subplot(223);plot(w,Hr);
>>title('Dap ung tan so');xlabel('\omega');ylabel('H(\omega)');
```

d. Phân bố cực - không:

```
>>subplot(224); zplane(h,1);
>>title('Bieu do cuc - khong');xlabel('Thuc');ylabel('Ao');
```

**Bài 5.6.** Thực hiện lại bài 5.5 với đáp ứng xung  $h = [-1 \ 2 \ 1.3 \ -2.2 \ 0.6 \ 3 \ 3 \ 0.6 \ -2.2 \ 1.3 \ 2 \ -1]$ .

**Bài 5.7.** Thực hiện lại bài 5.5 với đáp ứng xung  $h = [-1 \ 2 \ 1.3 \ -2.2 \ 0.6 \ 3 \ -0.6 \ 2.2 \ -1.3 \ -2 \ 1]$ .

**Bài 5.8.** Thực hiện lại bài 5.5 với đáp ứng xung  $h = [-1 \ 2 \ 1.3 \ -2.2 \ 0.6 \ 3 \ -3 \ -0.6 \ 2.2 \ -1.3 \ -2 \ 1]$ .

## 5.2 PHƯƠNG PHÁP CỦA SỐ

**Bài 5.9.** Tạo hàm *ideal\_lp* xác định đáp ứng xung của bộ lọc thông thấp lý tưởng theo tần số cắt  $\omega_c$  và chiều dài chuỗi đáp ứng xung.

```
function hd = ideal_lp(wc,M)
alpha = (M-1)/2;
n = [0:1:(M-1)];
m = n - alpha + eps;
hd = sin(wc*m)./(pi*m);
```

**Bài 5.10.** Tạo hàm *freqz\_m* tính toán độ lớn và pha của đáp ứng tần số, hàm trễ nhóm.

```
function [db,mag,pha,grd,w] = freqz_m(b,a)
```

```

% db = Do lon tuong doi theo dB tren doan tu 0 den pi
% mag = Do lon tuyet doi tren doan tu 0 den pi
% pha = Dap ung pha tren doan tu 0 den pi
% grd = Tre nhom tren doan tu 0 den pi
% w = Cac mau tan so doan tu 0 den pi
% b = Cac he so da thuc tu so cua H(z) (voi FIR: b=h)
% a = Cac he so da thuc mau so cua H(z) (voi FIR: a=[1])

[H,w] = freqz(b,a,1000,'whole');
H = (H(1:1:501))';
w = (w(1:1:501))';
mag = abs(H);
db = 20*log10((mag+eps)/max(mag));
pha = angle(H);
grd = grpdelay(b,a,w);

```

**Bài 5.11.** Thiết kế bộ lọc thông thấp theo phương pháp cửa sổ Hamming với các tham số như sau:

$$\omega_p = 0.2\pi; \omega_s = 0.3\pi; R_p = 0.25 \text{ dB}; A_s = 50 \text{ dB}$$

Việc thiết kế bộ lọc là quá trình tìm ra các tham số, hay chuỗi đáp ứng xung của bộ lọc, thoả mãn các yêu cầu chỉ tiêu kỹ thuật cho trước, cụ thể là một số hoặc tất cả các tham số tuyệt đối (absolute specification) sau:

- Tần số cắt dải thông  $\omega_p$
- Tần số cắt dải thông  $\omega_s$
- Bề rộng dải quá độ  $\Delta\omega$
- Độ gợn sóng dải thông  $\delta_p$
- Độ gợn sóng dải chặn  $\delta_s$

Các tham số thường được cho dưới dạng đơn vị dB:

- Độ gợn sóng dải thông theo dB:

$$R_p = -20 \log \frac{1-\delta_p}{1+\delta_p}$$

- Độ suy giảm dải chặn theo dB:

$$A_s = -20 \log \frac{\delta_s}{1+\delta_p}$$

Các hàm cửa sổ trong MATLAB: **boxcar** hay **rectwin**, **bartlett**, **hanning**, **hamming**, **blackman**, **Kaiser**.

```
>>wp = 0.2*pi; ws = 0.3*pi;
>>tr_width = ws - wp;
>>M = ceil(6.6*pi/tr_width) + 1;
>>n = [0:1:M-1];
>>wc = (ws+wp)/2;
>>hd = ideal_lp(wc,M);
>>w_ham = (hamming(M))';
>>h = hd.*w_ham;
>>[db,mag,pha,grd,w] = freqz_m(h,[1]);
>>delta_w = 2*pi/1000;
>>Rp = -(min(db(1:1:wp/delta_w+1)))
>>As = -round(max(db(ws/delta_w+1:1:501)))
```

a. Đáp ứng xung:

```
>>subplot(221); stem(n,hd);
>>axis([0,M-1,-0.1,0.3]);
>>title('Đáp ứng xung lý tưởng');xlabel('n'); ylabel('h_d(n)');
```

b. Cửa sổ Hamming:

```
>>subplot(222); stem(n,w_ham);
>>axis([0,M-1,0,1.1]);
>>title('Cửa sổ Hamming');xlabel('n'); ylabel('w(n)');
```

c. Đáp ứng tần số:

```
>>subplot(223); stem(n,h);
>>axis([0,M-1,-0.1,0.3]);
```

```
>>title('Đáp ứng tần số');xlabel('n'); ylabel('h(n)');
```

d. Đáp ứng tần số theo dB:

```
>>subplot(224); plot(w,db);
```

```
>>axis([0,1,-100,10]);
```

```
>>title('Đáp ứng tần số theo dB');xlabel('\omega'); ylabel('dB');
```

**Bài 5.12.** Thiết kế bộ lọc thông dải theo phương pháp cửa sổ Blackman với các tham số như sau:

$$\omega_{s1} = 0.2\pi; \omega_{p1} = 0.35\pi; \omega_{p2} = 0.65\pi; \omega_{s2} = 0.8\pi; R_p = 1 \text{ dB}; A_s = 60 \text{ dB}$$

Quá trình thực hiện như bài 5.11. Về lý thuyết, đáp ứng xung lý tưởng của bộ lọc thông dải lý tưởng là hiệu đáp ứng xung của hai bộ lọc thông thấp lý tưởng. Dùng hàm **idea\_lp** như bài 5.9 để xác định đáp ứng xung của bộ lọc thông dải lý tưởng trong đó các tần số cắt có thể chọn là trung bình của các tần số cắt dải thông và dải chặn. Thông số độ rộng dải chuyển tiếp để tính chiều dài chuỗi đáp ứng xung có thể chọn là giá trị nhỏ nhất của độ rộng hai dải chuyển tiếp, từ dải chặn lên dải thông  $[\omega_{s1}, \omega_{p1}]$  và từ dải thông xuống dải chặn  $[\omega_{p2}, \omega_{s2}]$ .

## 5.3 PHƯƠNG PHÁP LẤY MẪU TẦN SỐ

Xét đáp ứng xung  $h(n)$  của hệ thống LTI có đáp ứng tần số là:

$$H(\omega) = \sum_{n=0}^{M-1} h(n)e^{-j\omega n}$$

Ta chỉ định một tập tần số như sau:

$$\omega_k = \frac{2\pi}{M}(k + \alpha), \begin{cases} k = 0, 1, \dots, \frac{M-1}{2} \text{ nếu } M \text{ lẻ} \\ k = 0, 1, \dots, \frac{M}{2} - 1 \text{ nếu } M \text{ chẵn} \\ \alpha = 0 \text{ hay } \frac{1}{2} \end{cases}$$

Khi đó:

$$H(k + \alpha) = H\left(\frac{2\pi}{M}(k + \alpha)\right) = \sum_{n=0}^{M-1} h(n)e^{-j\frac{2\pi}{M}(k+\alpha)n}$$

Tập hợp các giá trị  $\{H(k + \alpha)\}$  gọi là các mẫu tần số của  $H(\omega)$ . Trong trường hợp  $\alpha = 0$ ,  $\{H(k)\}$  tương ứng với DFT  $M$  điểm của  $h(n)$ .

**Bài 5.13.** Thiết kế bộ lọc thông thấp theo phương pháp lấy mẫu tần số với các tham số như sau:

$$\omega_p = 0.2\pi; \omega_s = 0.3\pi; R_p = 0.25 \text{ dB}; A_s = 50 \text{ dB}$$

Chọn đáp ứng xung có chiều dài 60 ứng với 60 mẫu tần số trong khoảng  $[0, 2\pi)$ . Dải thông có độ rộng là  $0.2\pi$  tương đương với 7 mẫu nhận giá trị 1. Giả sử quá trình tối ưu hoá chỉ ra nên chọn dải chuyển tiếp 2 mẫu nhận các giá trị  $T_1 = 0.5925$  và  $T_2 = 0.1099$ . Mẫu các tần số được cho như sau:

$$H(k) = [1, 1, 1, 1, 1, 1, 1, T_1, T_2, 0, 0, \dots, 0, 0, T_2, T_1, 1, 1, 1, 1, 1, 1]$$

(43 số 0)

```
>>M = 60; alpha = (M-1)/2; L = 0:M-1; wl = (2*pi/M)*L;

>>Hk = [ones(1,7), 0.5925, 0.1099, zeros(1,43), 0.1099, 0.5925,
ones(1,6)];

% Đáp ứng tần số mẫu lý tưởng

>>Hdr = [1, 1, 0, 0]; wdl = [0, 0.2, 0.3, 1];

% Đáp ứng tần số lý tưởng để biểu diễn đồ thị

>>k1 = 0:floor((M-1)/2); k2 = floor((M-1)/2)+1:M-1;

>>angH = [-alpha*(2*pi)/M*k1, alpha*(2*pi)/M*(M-k2)];

>>H = Hk.*exp(j*angH);

>>h = real(ifft(H,M));

>>[db, mag, pha, grd, w] = freqz_m(h, 1);

>>[a, ww, L, Hr] = FIR_t2(h);
```

a. Chuỗi mẫu tần số:

```
>>subplot(221); plot(wl(1:31)/pi, Hk(1:31), 'o', wdl, Hdr);

>>axis([0, 1, -0.1, 1.1]);

>>title('Cac mau tan so: M=60, T2 = 0.5925, T1 = 0.1099');

>>xlabel(f['\pi']); ylabel('Hr(k)');
```

b. Đáp ứng xung của bộ lọc thực tế:

```
>>subplot(222); stem(L, h);
```

```
>>axis([-1,M,-0.1,0.3]);
>>title('Dap ung xung');
>>xlabel('n'); ylabel('h(n)');
```

c. Biên độ của đáp ứng tần số:

```
>>subplot(223); plot(ww/pi,Hr,wl(1:31)/pi,Hk(1:31),'o');
>>axis([0,1,-0.2,1.2]);
>>title('Bien do cua dap ung tan so');
>>xlabel('f[*\pi]'); ylabel('Hr(w)');
```

d. Biên độ của đáp ứng tần số theo dB:

```
>>subplot(224); plot(w/pi,db);
>>axis([0,1,-100,10]); grid
>>title('Bien do cua dap ung tan so ');
>>xlabel('f[*\pi]'); ylabel('dB');
```

**Bài 5.14.** Thiết kế bộ lọc thông dải theo phương pháp lấy mẫu tần số với các tham số như sau:

$$\omega_{s1} = 0.2\pi; \omega_{p1} = 0.35\pi; \omega_{p2} = 0.65\pi; \omega_{s2} = 0.8\pi; R_p = 1 \text{ dB}; A_s = 60 \text{ dB}$$

Chọn đáp ứng xung có chiều dài 40 ứng với 40 mẫu tần số trong khoảng  $[0, 2\pi)$ . Dải thông có độ rộng là  $0.3\pi$  tương đương với 7 mẫu nhận giá trị 1. Giả sử quá trình tối ưu hoá chỉ ra nên chọn dải chuyển tiếp 2 mẫu nhận các giá trị  $T_1 = 0.109021$  và  $T_2 = 0.59417456$ . Mẫu các tần số được cho như sau:

$$H(k) = [0, 0, 0, 0, 0, T_1, T_2, 1, 1, 1, 1, 1, 1, 1, T_2, T_1, 0, \dots, 0, T_1, T_2, 1, 1, 1, 1, 1, 1, 1, T_2, T_1, 0, 0, 0, 0]$$

(9 số 0)

Quá trình thực hiện tương tự bài 5.13.

## 5.4 PHƯƠNG PHÁP LẬP

Đáp ứng tần số của 4 loại bộ lọc FIR:

$$H(\omega) = P(\omega)Q(\omega)$$

Hàm sai số giữa bộ lọc thực tế và bộ lọc lý tưởng:

$$E(\omega) = W(\omega)[H_d(\omega) - H(\omega)] = W(\omega)Q(\omega) \left[ \frac{H_d(\omega)}{Q(\omega)} - P(\omega) \right]$$

Ta định nghĩa:

$$\hat{W}(\omega) = W(\omega)Q(\omega) \quad \hat{H}_d(\omega) = \frac{H_d(\omega)}{Q(\omega)}$$

Khi đó:

$$E(\omega) = \hat{W}(\omega)[\hat{H}_d(\omega) - P(\omega)]$$

Parks và McClellan đã đưa ra giải pháp sử dụng thuật toán Remez để tìm ra đáp ứng xung của bộ lọc tối ưu, tức là gần đúng theo nghĩa Chebyshev đối với một bộ lọc lý tưởng, cho giá trị M là chiều dài của chuỗi đáp ứng xung với các điều kiện ràng buộc về độ gợn sóng ở dải thông và dải chặn như sau:

1. Xác định loại bộ lọc, tính giá trị R và xây dựng các hàm  $W(\omega)$ ,  $Q(\omega)$ .

Loại bộ lọc	R	P( $\omega$ )	Q( $\omega$ )
FIR loại 1	$\frac{M-1}{2}$	$\sum_{n=0}^R \bar{a}(n)\cos\omega n$	1
FIR loại 2	$\frac{M}{2} - 1$	$\sum_{n=0}^R \bar{b}(n)\cos\omega n$	$\cos(\omega/2)$
FIR loại 3	$\frac{M-1}{2} - 1$	$\sum_{n=0}^R \bar{c}(n)\cos\omega n$	$\sin\omega$
FIR loại 4	$\frac{M}{2} - 1$	$\sum_{n=0}^R \bar{d}(n)\cos\omega n$	$\sin(/2)\omega$

2. Xây dựng bài toán gần đúng bằng cách xác định các hàm  $\hat{W}(\omega)$ ,  $H_d(\omega)$ .
3. Sử dụng thuật toán trao đổi Remez để tìm ra hàm tối ưu P( $\omega$ ).

① Chọn lấy R+2 điểm rời rạc, giả sử đó là các cực trị của hàm sai số.

② Trên cơ sở tại R+2 điểm rời rạc nói trên, hàm E( $\omega$ ) luân phiên đổi dấu và có trị tuyệt đối bằng một giá trị  $\delta$  nào đó, tính nội suy lại giá trị  $\delta$  và hàm P( $\omega$ ), từ đó tính ra hàm sai số E( $\omega$ ), tính được cực trị thực của hàm sai số.



③ Xem xét xem các giá trị rời rạc được chọn ban đầu có thực sự là các điểm mà hàm sai số  $E(\omega)$  đạt cực trị và có trị tuyệt đối bằng nhau hay không. Nếu không, tìm các điểm tại đó  $E(\omega)$  đạt cực trị.

④ Trong các điểm cực trị của  $E(\omega)$  lấy ra  $R+2$  điểm và quay về lặp lại từ bước 2.

⑤ Lặp lại các bước 2, 3, và 4 cho đến khi tập hợp các điểm rời rạc hội tụ.

⑥ Từ tập các điểm rời rạc cuối cùng, tính ra hàm  $P(\omega)$ , từ đó tính ra các hệ số của  $P(\omega)$ .

4. Tính các giá trị của chuỗi đáp ứng xung  $h(n)$ .

Khi chọn giá trị  $M$  càng chuẩn thì kết quả là thu được bộ lọc có hàm đáp ứng tần số càng gần với yêu cầu bài toán. Nếu như với giá trị  $M$  nào đó mà chưa thoả mãn được yêu cầu thì phải tăng giá trị  $M$  đến khi nào thoả mãn các điều kiện ràng buộc cho  $\delta_p$  và  $\delta_s$  (hay  $A_s$  và  $R_p$ ). Một công thức lựa chọn ban đầu cho chiều dài  $M$  của đáp ứng xung là:

$$M_0 = \frac{-20 \log \sqrt{\delta_1 \delta_2} - 13}{14.6 \Delta f} \text{ với } \Delta f = \frac{\omega_s - \omega_p}{2\pi}$$

Trong MATLAB, tìm đáp ứng xung của bộ lọc tối ưu với giá trị  $M$  và hàm đáp ứng tần số lý tưởng cho trước được thực hiện bởi hàm **firpm**.

**Bài 5.15.** Tạo script file sau để biểu diễn bộ lọc trên đồ thị:

```
wp = 0.2*pi; ws = 0.3*pi; Rp = 0.25; As = 50;

delta_w = 2*pi/1000;

wsi = ws/delta_w+1;

delta1 = (10^(Rp/20)-1)/(10^(Rp/20)+1);

delta2 = (1+delta1)*(10^(-As/20));

deltaH = max(delta1,delta2);

deltaL = min(delta1,delta2);

weights = [delta2/delta1 1];

deltaf = (ws-wp)/(2*pi);

M = ceil((-20*log10(sqrt(delta1*delta2))-13)/(14.6*deltaf)+1)

f = [0 wp/pi ws/pi 1];

m = [1 1 0 0];
```

```

h = firpm(M-1,f,m,weights);

[db,mag,pha,grd,w] = freqz_m(h,[1]);

Asd = -max(db(wsi:1:501))

while Asd<As

    M = M+1

    [h,ERR,RES] = firpm(M-1,f,m,weights);

    [db,mag,pha,grd,w] = freqz_m(h,[1]);

    Asd = -max(db(wsi:1:501))

end

n = [0:1:M-1];

subplot(221); stem(n,h);

axis([0,M-1,-0.1,0.3]);

title('Dap ung xung');

xlabel('n'); ylabel('h(n)');

subplot(222); plot(w/pi,db); grid;

axis([0,1,-80,10]);

title('Dap ung tan so theo dB');

xlabel('f[*\pi]'); ylabel('dB');

subplot(223); plot(w/pi,mag); grid;

axis([0,1,-0.2,1.2]);

title('Dap ung tan so');

xlabel('f[*\pi]'); ylabel('Hr(\omega)');

subplot(224); plot(RES.fgrid,RES.error); grid;

axis([0,1,-0.0150,0.0150]);

```

```
title('Đáp ứng tần số của lõi');  
xlabel('f[*\pi]'); ylabel('Er(\omega)');
```

**Bài 5.16.** Thực hiện như bài 5.15 cho bộ lọc thông dải theo phương pháp lấy mẫu tần số với các tham số như sau:

$$\omega_{s1} = 0.2\pi; \omega_{p1} = 0.35\pi; \omega_{p2} = 0.65\pi; \omega_{s2} = 0.8\pi; R_p = 1 \text{ dB}; A_s = 60 \text{ dB}$$

# BÀI 6: BỘ LỌC SỐ IIR

Sau khi học xong bài này, người học có thể:

- Thiết kế một bộ lọc IIR dựa theo các thông số cho trước.

## 6.1 THIẾT KẾ BỘ LỌC TƯƠNG TỰ

Các yêu cầu thiết kế cho bộ lọc tương tự dựa trên điều kiện giới hạn các chỉ tiêu kỹ thuật cho bình phương biên độ của hàm đáp ứng tần số. Hàm truyền  $H_a(s)$  của một hệ thống tuyến tính bất biến tương tự là biến đổi Laplace hàm đáp ứng xung  $h_a(t)$  của hệ thống và  $H_a(s)$  chính bằng tỷ số giữa biến đổi Laplace của tín hiệu đầu ra với biến đổi Laplace của tín hiệu đầu vào. Hàm đáp ứng tần số  $H_a(\omega)$  của một hệ thống tuyến tính bất biến là hàm thể hiện đáp ứng của hệ thống với đầu vào là các giá trị tần số khác nhau. Do đó, hàm đáp ứng tần số là biến đổi Fourier hàm đáp ứng xung  $h_a(t)$  của hệ thống và hàm  $H_a(\omega)$  cũng chính là hàm hệ thống  $H_a(s)$  đánh giá trên trục ảo.

Đối với các hệ thống thực hiện được về mặt vật lý, đáp ứng xung bao giờ cũng là một hàm thực. Do đó, hàm truyền luôn đối xứng qua trục thực. Mặt khác, một hệ thống tương tự là nhân quả và ổn định nếu và chỉ nếu tất cả các điểm cực của hàm truyền đặt nằm ở nửa bên trái của mặt phẳng  $s$  hoặc nằm ở gốc tọa độ. Khi ta xét đến bình phương biên độ của hàm hệ truyền đạt, các điểm cực của hàm số này sẽ phân bố trên tất cả các góc phần tư của mặt phẳng  $S$ . Lúc này, việc xét đáp ứng tần số của hệ thống cũng thuận tiện và tất cả các điểm cực nằm bên trái mặt phẳng  $S$  của hàm  $|H_a(s)|^2$ .

Yêu cầu về chỉ tiêu kỹ thuật của bộ lọc thông thấp tương tự thường được cho dưới dạng như sau:

$$\frac{1}{1+\epsilon^2} \leq |H_a(\omega)| \leq 1, |\omega| \leq \omega_p$$

$$0 \leq |H_a(\omega)| \leq \frac{1}{A^2}, |\omega| \geq \omega_s$$

với  $\omega_p$  và  $\omega_s$  lần lượt là các tần số cắt dải thông và tần số cắt dải chặn,  $\epsilon$  là tham số gợn sóng và  $A$  là tham số suy giảm. Quan hệ giữa  $\epsilon$  và  $A$  với  $R_p$  và  $A_s$  hay  $\delta_p$  và  $\delta_s$ :

$$R_p = -10 \log \frac{1}{1+\epsilon^2} \rightarrow \epsilon = \sqrt{10^{\frac{R_p}{10}} - 1}$$

$$A_s = -10 \log \frac{1}{A^2} \rightarrow A = 10^{\frac{A_s}{20}}$$

$$\frac{1 - \delta_p}{1 + \delta_p} = \sqrt{\frac{1}{1 + \epsilon^2}} \rightarrow \epsilon = \frac{2\sqrt{\delta_p}}{1 - \delta_p}$$

$$\frac{\delta_s}{1 + \delta_p} = \frac{1}{A} \rightarrow A = \frac{1 + \delta_p}{\delta_s}$$

Có 4 định dạng cơ bản thường được vận dụng trong quá trình thiết kế bộ lọc tương tự là: bộ lọc Butterworth, bộ lọc Chebyshev-1, bộ lọc Chebyshev-2 và bộ lọc Elliptic. Các hàm MATLAB có thể sử dụng cho bài thực hành này là:

**freqs**: trả về đáp ứng tần số của một hệ thống tương tự khi biết hàm truyền dưới dạng phân thức hữu tỷ.

**impz**: trả về đáp ứng xung của một hệ thống tương tự khi biết hàm truyền dưới dạng phân thức hữu tỷ

**buttap, cheb1ap, cheb2ap, ellipap**: trả về các điểm không, điểm cực, và độ lợi trong thiết kế của một hàm truyền bộ lọc thông thấp bậc N, tần số cắt đã được chuẩn hoá bằng 1 với các định dạng lần lượt là Butterworth, Chebyshev-I, Chebyshev-II, và Elliptic.

**impinvar, bilinear**: trả về các hệ số của đa thức tử số và đa thức mẫu số hàm truyền đạt của hệ thống số xuất phát từ hệ thống tương tự qua các phương pháp chuyển đổi bất biến xung và song tuyến tính.

**butter, cheby1, cheby2, ellip**: trả về các hệ số của đa thức tử số và đa thức mẫu số hàm truyền của bộ lọc số dựa trên tham số đầu vào là các tần số cắt, phương pháp chuyển đổi được sử dụng trong các hàm này là phương pháp biến đổi song tuyến tính.

Hàm **freqs** của MATLAB trả về đáp ứng tần số của một hệ thống tương tự khi biết trước hệ số của đa thức tử số và đa thức mẫu số của hàm truyền đạt  $H_a(s)$ . Trong nhiều trường hợp, để thuận tiện ta cần tìm thêm các thông số: hàm độ lớn của đáp ứng tần số, hàm pha của đáp ứng tần số, hàm trễ nhóm, thể hiện độ lớn theo thang decibels.

**Bài 6.1.** Tạo hàm tính đáp ứng tần số **freqs\_m** nhằm tính các thông số trên:

```
function [db,mag,pha,w] = freqs_m(b,a,wmax);

% db = Độ lớn tương đối theo dB trên đoạn từ 0 đến wmax
% mag = Độ lớn tuyệt đối trên đoạn từ 0 đến wmax
% pha = Đáp ứng pha trên đoạn từ 0 đến wmax
```

```
% w = Cac mau tan so tren doan tu 0 den wmax
% b = Cac he so da thuc tu so cua Ha(s)
% a = Cac he so da thuc mau so cua Ha(s)
% wmax = Tan so cuc dai theo don vi rad/sec tren doan
% tan so mong muon tim dap ung tan so
w = [0:1:500]*wmax/500;
H = freqs(b,a,w);
mag = abs(H);
db = 20*log10((mag+eps)/max(mag));
pha = angle(H);
```

Hàm **cheb1ap** trả về danh sách các điểm không, điểm cực và độ lợi của hàm truyền đạt cho thiết kế bộ lọc dạng Chebyshev I, tần số cắt đã được chuẩn hoá.

Đáp ứng tần số của bộ lọc thông thấp Chebyshev I bậc N:

$$|H_a(\omega)| = \frac{1}{1 + \epsilon^2 T_N^2\left(\frac{\omega}{\omega_c}\right)}$$

Trong đó  $\epsilon$  là tham số gợn sóng dải thông,  $\omega_c$  là tần số cắt và  $T_N(x)$  là đa thức Chebyshev bậc N:

$$T_N(x) = \begin{cases} \cos[N\cos^{-1}(x)] & -1 \leq x \leq 1 \\ \cosh[N\cosh^{-1}(x)] & |x| > 1 \end{cases}$$

Giá trị thích hợp của N cho như sau:

$$N = \left\lceil \frac{\log(g + \sqrt{g^2 - 1})}{\log(\omega_r + \sqrt{\omega_r^2 - 1})} \right\rceil$$

Với:

$$g = \sqrt{\frac{A^2 - 1}{\epsilon^2}} \quad \omega_r = \frac{\omega_s}{\omega_p}$$

**Bài 6.2.** Tạo hàm **u\_chb1ap** nhằm trả về hệ số của các đa thức tử số và đa thức mẫu số của hàm truyền đạt cho thiết kế bộ lọc dạng Chebyshev I có tần số cắt tùy ý:

```
function [b,a] = u_chb1ap(N,Rp,Omegac)
% Bo loc thong thap dang Chebyshev-1
% tan so cat khong duoc chuan hoa
```

```

% [b,a] = u_chblap(N,Rp,Omegac)

% b = cac he so da thuc tu so cua Ha(s)

% a = cac he so da thuc mau so cua Ha(s)

% N = Bac cua bo loc Chebyshev-I

% Rp = Do gon dai thong theo don vi dB; Rp > 0

% Omeac = tan so cat theo don vi radians/sec

[z,p,k] = cheblap(N,Rp);

a = real(poly(p));

aNn = a(N+1);

p = p*Omeac;

a = real(poly(p));

aNu = a(N+1);

k = k*aNu/aNn;

B = real(poly(z));

b0 = k;

b = k*B;

```

Hàm số mô tả ở trên trả về hàm truyền với bậc N cho trước. Bậc của bộ lọc có thể lựa chọn cho phù hợp tối ưu với các chỉ tiêu kỹ thuật yêu cầu đầu vào.

**Bài 6.3.** Tạo hàm **afd\_chb1** trả về thiết kế bộ lọc thông thấp tương tự, định dạng Chebyshev có bậc tối ưu:

```

function [b,a] = afd_chb1(Wp,Ws,Rp,As)

% Analog Lowpass Filter Design: Chebyshev-1

% [b,a] = afd_chb1(Wp,Ws,Rp,As)

% b = cac he so da thuc tu so cua Ha(s)

% a = cac he so da thuc mau so cua Ha(s)

% Wp = tan so cat dai thong theo don vi rad/sec; Wp > 0

% Ws = tan so cat dai chan theo don vi rad/sec; Ws>Wp > 0

% Rp = Do gon dai thong theo don vi dB; (Rp > 0)

% As = Do suy giam dai chan theo don vi +dB; (Ap > 0)

if Wp <= 0

```

```

        error('Tan so cat dai thong phai lon hon 0')
    end

    if Ws <= Wp

        error('Tan so cat dai thong phai lon hon tan so cat dai chan')
    end

    if (Rp <= 0) | (As<0)

        error('Do gon dai thong va/hoac Do suy giam dai chan phai lon
hon 0')
    end

    ep = sqrt(10^(Rp/10)-1);
    A = 10^(As/20);
    OmegaC = Wp;
    OmegaR = Ws/Wp;
    g = sqrt(A*A-1)/ep;
    N = ceil(log10(g+sqrt(g*g-1))/log10(OmegaR+sqrt(OmegaR*OmegaR-
1)));
    fprintf('\n*** Bac cua bo loc Chebyshev-1 = %2.0f\n',N);
    [b,a] = u_chblap(N,Rp,OmegaC);

```

**Bài 6.4.** Thiết kế bộ lọc thông thấp tương tự, định dạng Chebyshev-I với các tham số đầu vào như sau:

$$\omega_p = 0.2\pi; \omega_s = 0.3\pi; R_p = 1 \text{ dB}; A_s = 16 \text{ dB}$$

Viết chương trình tính và biểu diễn trên đồ thị:

- Độ lớn của đáp ứng tần số
- Đáp ứng pha của bộ lọc
- Độ lớn tương đối tính theo dB của đáp ứng tần số
- Đáp ứng xung của bộ lọc tương tự

## 6.2 THIẾT KẾ BỘ LỌC SỐ

Biến đổi song tuyến tính là phép ánh xạ từ mặt phẳng  $s$  sang mặt phẳng  $z$  định nghĩa như sau:



$$S = \frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}}$$

Với bộ lọc tương tự cho trước có hàm hệ thống  $H_a(s)$ , bộ lọc số được thiết kế như sau:

$$H(z) = H_a\left(\frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}}\right)$$

**Bài 6.5.** Chuyển đổi bộ lọc với các tham số đã cho ở bài 6.4 sang bộ lọc số bằng phương pháp biến đổi song tuyến. Hàm **bilinear** cho phép thực hiện việc chuyển đổi này.

Tạo script file **bai605.m** như sau:

```
wp = 0.2*pi; % digital Passband freq in Hz
ws = 0.3*pi; % digital Stopband freq in Hz
Rp = 1; % Passband ripple in dB
As = 15; % Stopband attenuation in dB
T = 1; Fs = 1/T; % Dat T=1
OmegaP = (2/T)*tan(wp/2);
OmegaS = (2/T)*tan(ws/2);
% Tinh toan bo loc tuong tu:
[cs, ds] = afd_chb1(OmegaP, OmegaS, Rp, As);
% Bien doi song tuyen tinh:
[b, a] = bilinear(cs, ds, Fs);
[db, mag, pha, grd, w] = freqz_m(b, a);
```

a. Đáp ứng tần số:

```
subplot(221); plot(w/pi, mag);
axis([0, 1, 0, 1.2]); grid
title('Dap ung tan so');
xlabel('f[*\pi]'); ylabel('|H_r(\omega)|');
```

b. Hàm độ lớn tương đối tính theo dB:

```
subplot(222); plot(w/pi, db);
axis([0, 1, -30, 10]); grid
title('Dap ung tan so theo dB');
xlabel('f[*\pi]'); ylabel('dB');
```

## c. Hàm đáp ứng pha:

```
subplot(223); plot(w/pi, pha/pi);
axis([0,1,-1,1]); grid
title('Đáp ứng pha');
xlabel('f[*\pi]'); ylabel('Angle(H_r(\omega))');
```

## d. Trễ nhóm theo tần số:

```
subplot(224); plot(w/pi, grd);
axis([0,1,0,15]); grid
title('Group Delay');
xlabel(' f[*\pi]'); ylabel('Samples');
```

Gõ lệnh:

```
>> bai605
```

tại cửa sổ lệnh để kiểm tra kết quả.

Phương pháp bất biến xung thực hiện lấy mẫu bộ lọc tương tự để tạo ra bộ lọc số:

$$h(n) = h_a(nT)$$

Đáp ứng tần số của bộ lọc số có quan hệ với đáp ứng tần số của bộ lọc tương tự như sau:

$$H(\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} H_a \left[ \frac{(\omega - 2\pi k)}{T} \right]$$

**Bài 6.6.** Thực hiện yêu cầu của bài 6.5 theo phương pháp bất biến xung, dùng hàm **impinvar** của MATLAB. So sánh kết quả thu được với câu trên.

**Bài 6.7.** Tạo hàm **zmapping** thực hiện việc chuyển đổi băng tần số, trả về hàm truyền của bộ lọc mới với tham số đầu vào là hàm truyền đạt của bộ lọc thông thấp, hàm đa thức thể hiện phép đổi biến số độc lập:

```
% function [bz,az] = zmapping(bZ,aZ,Nz,Dz)

bzord = (length(bZ)-1)*(length(Nz)-1);
azord = (length(aZ)-1)*(length(Dz)-1);

bz = zeros(1,bzord+1);

for k = 0: bzord
    pln = [1];
```

```

    for l = 0:k-1
        pln = conv(pln,Nz);
    end
    pld = [1];
    for l = 0:bzord-k-1
        pld = conv(pld,Dz);
    end
    bz = bz+bZ(k+1)*conv(pln,pld);
end
az = zeros(1,azord+1);
for k = 0:azord
    pln = [1];
    for l = 0:k-1
        pln = conv(pln,Nz);
    end
    pld = [1];
    for l = 0:azord-k-1
        pld = conv(pld,Dz);
    end
    az = az+aZ(k+1)*conv(pln,pld);
end
az1 = az(1); az = az/az1; bz=bz/az1;

```

**Bài 6.8.** Viết chương trình chuyển đổi từ bộ lọc thông thấp theo thiết kế của bài 6.5 sang bộ lọc thông cao có tần số cắt  $\omega_c=0.6\pi$ . Tính và biểu diễn trên đồ thị:

- Độ lớn của đáp ứng tần số
- Hàm đáp ứng pha của bộ lọc
- Hàm độ lớn tương đối tính theo dB của đáp ứng tần số
- Trễ nhóm theo tần số.

# TÀI LIỆU THAM KHẢO

1. Phạm Hùng Kim Khánh. *Xử lý tín hiệu số*. ĐH Công nghệ TPHCM.
2. Sanjit K. Mitra. *Digital Signal Processing Laboratory using MatLab*.
3. Đinh Đức Anh Vũ, Vũ Tuấn Thanh, Lê Trọng Nhân, Tôn Thất Đại Hải. *Giáo trình Thực hành Xử lý tín hiệu số*. Bộ môn Kỹ thuật Máy tính, ĐH Bách Khoa TPHCM.
4. Hồ Văn Sung. *Thực hành Xử lý tín hiệu số với MatLab*. NXB Khoa học và Kỹ thuật.