# MỤC LỤC

# LỜI NÓI ĐẦU

Máy tính từ khi ra đời đã tạo điều kiện và hỗ trợ con người trong nhiều lĩnh vực của cuộc sống và ngày càng được coi như công cụ không thể thiếu trong học tập cũng như nghiên cứu.

Chính vì vậy, việc nâng cao và phát triển khả năng tính toán và xử lí của máy tính ngày càng được các nhà khoa học, kỹ sư các ngành quan tâm đến. Tuy nhiên để viết được một chương trình bằng ngôn ngữ lập trình cấp cao phục vụ tốt cho một lĩnh vực khoa học kỹ thuật đòi hỏi không những phải giỏi về toán học, các kiến thức về lập trình trên máy tính, hệ thống máy tính ... mà còn phải nắm rất vững các kiến thức về chuyên ngành đó. Người lập trình để đạt được những yêu này phải mất rất nhiều thời gian và tốn nhiều công sức.

Để tạo điều kiện cho các nhà khoa học thuộc các chuyên ngành khác, người ta đã xây dựng nên những phần mềm xử lý dữ liệu đơn giản, tiện lợi. Matlab là một trong những phần mềm như vậy và hiện nay đang được sử dụng rộng rãi. Nó không chỉ cho phép tính toán, mà còn cung cấp cho ta những công cụ cực mạnh biểu diễn, xử lí các dữ liệu, thông tin bằng đồ hoạ.

MATLAB là một phần mềm có rất nhiều ưu điểm để để các nhà khoa học, các kỹ sư lựa chọn:

* Dễ học và dễ sử dụng.

* Là một phần mềm mạnh, mềm dẻo, trong nhiều lĩnh vực khoa học kỹ thuật .

* Chính xác, đơn giản và trong sáng.

* Đang được các công ty phần mềm lớn trên thế giới ủng hộ và phát triển.

Trước hết MATLAB dễ học và dễ sử dụng: MATLAB có các thư viện chuẩn, các hàm sẵn có để bạn có thể sử dụng thuận lợi và dễ dàng. Mặt khác, bạn chỉ cần nắm được một số kiến thức toán học cơ bản về đại số và lượng giác, toán học cao cấp là có thể sử dụng MATLAB như một công cụ mạnh cho các ứng dụng của mình. MATLAB không đòi hỏi bạn phải có nhiều kiến thức về máy tính cũng như khả năng lập trình. Bạn có thể lập các chương trình ứng dụng cho chuyên ngành của bạn một cách tương đối dễ dàng, khi bạn nắm vững các kiến thức sau:

+ Toán ứng dụng cơ bản.

+ Lý thuyết số cơ bản

+ Một  chút về lập trình máy tính.

+ Phương pháp tính.

Matlab (Matrix Laboratory) là sản phẩm phần mềm của Math Work, đầu tiên được thiết kế trên cơ sở toán học, phục vụ chủ yếu đơn thuần cho toán học. Tuy nhiên, ngày nay nó được phát triển xa hơn nhiều so với Matlab nguyên thuỷ và là một phần mềm có giao diện cực mạnh và có khả năng lập trình để giải quyết các vấn đề, các bài toán trong rất nhiều lĩmh vực rất khác nhau của khoa học kĩ thuật như điện, phản ứng hạt nhân, tự động hoá, nghiên cứu về gien... Phần tử cơ bản của Matlab là ma trận. Các câu lệnh của Matlab viết tương tự như cách mô tả các vấn đề kĩ thuật bằng toán học, vì thế viết các chương trình bằng ngôn ngữ Matlab nhanh hơn và đơn giản hơn nhiều so với viết chương trình bằng các ngôn ngữ lập trình bậc cao như Pascan, Fotran, C.

Hơn thế nữa cấu trúc chương trình cũng như cấu trúc các hàm sẵn có trong MatLab được mô tả gần giống với ngôn ngữ lập trình C. ĐIều rất thuận lợi cho những người đã biết qua và sử dụng C cũng như một loại ngôn ngữ lập trình cơ bản khác  bất kỳ khác.

Thông thường, đối với các dữ liệu rời rạc: dữ liệu thống kê - kế toán, thông tin về khí hậu... được lưu dưới dạng ma trận. Còn đối với các hàm liên tục: sóng âm, âm thanh, hình ảnh... được biến đổi thành các tín hiệu số và được ghi lại trong các file dữ liệu. Sau đó, người ta sử dụng các hàm toán học của MATLAB đề xử lý chúng một cách dễ dàng.

Các vấn đề được sẽ được phân tích và giải quyết theo 5 bước như sau:

+ Phân tích và biểu diễn vấn đề một cách rõ ràng.

+ Mô tả các giá trị đầu vào và các giá trị đầu ra cần phải tính toán.

+ Thao tác với các ví dụ đơn giản

+ Viết chương trình bằng Matlab

+ Kiểm tra lại chương trình này bằng các bộ dữ liệu đa dạng.

Để nâng cao kỹ năng phân tích và giải quyết bài toán cần thực hành 5 bước trên một cách thuần thục. Từ đó sẽ tiếp cận và tìm ra được giải pháp đơn giản, dễ hiểu và hay nhất cho mỗi bài toán.

Dưới đây chúng ta sẽ làm một ví dụ theo theo từng bước trên để có thể hiểu kỹ hơn về phương pháp áp dụng cho bài toán cụ thể. Bài toán tính khoảng cách giữa hai đIểm trong mặt phẳng.

**Bước 1 Phân tích đặt vấn đề**:

Trong bước đầu tiên tiên này, bài toán đưa ra phải được xem xét đánh giá và đặt vấn đề một cách rõ ràng và cụ thể. ĐIều này cực kỳ quan trọng vì nó quyết định đến toàn bộ hướng đi của bài toán sau này. Cho ví dụ nêu ở trên thì vấn đề được nêu ra là:

Tính khoảng cách giữa 2 đIểm của đường thẳng trong mặt phẳng

**Bước 2 Mô tả dữ liệu vào ra:**

ở bước này việc mô tả thông tin cần giải quyết phải tiến hành cẩn trọng vì nó sẽ quyết định đến tham số được sử dụng và tính toán. Rất nhiều trường hợp, sơ đồ khối được sử dụng hữu hiệu để cho phép xác định vị trí luồng vào ra. Tuy nhiên một số trường hợp chúng chỉ là các hộp đen vì rằng chúng ta

không thể xác định luồng ra tại một đIểm nào đó trong các bước. Nhưng chúng ta có thể chỉ ra thông tin để tính toán luồng ra.

ĐIểm1 khoảng cáchgiữa 2 đIểm

ĐIểm 2

**Bước 3 Thao tác tay**:

Bước này dùng để thao tác và tính toán bằng tay sử dụng các tập dữ liệu đầu vào đơn giản. Nó là bước rất quan trọng và không nên bỏ qua kể cả với loại hình bài toán đơn giản. Đây là bước tiền đề để chúng ta đi vào cụ thể cho việc tìm ra giải pháp. Nếu ở đây chúng ta không thể lấy được dữ liệu hay tính được đầu ra thì chúng ta có thể chuyển sang bước kế tiếp.

Ví dụ: Với hai đIểm P1 và P2 có toạ độ ( 1,5 ) & ( 4,7 )

Khoảng cách giữa hai điểm bằng công thức Pythagorean

Bước 4 Giải pháp bằng MatLab

Tới bước này bài toán được chuyển đổi sang giải pháp MatLab. ĐIều đó có nghĩa chúng ta sẽ sử dụng các hàm toán học hay còn gọi là các lệnh. Và dưới đây bài toán của chúng ta sẽ được mô tả theo MatLab.

```
>> P1 = [ 1, 5 ]
>> P2 = [ 4, 7 ]
>> d  = sqrt ( sum ( P2-P1)^2 ) )
```

**Bước 5 Kiểm tra:**

Bước kiểm tra là bước cuối cùng trong chuỗi các tiến trình giải bài toán. Chúng ta nên kiểm tra bài toán bằng các dữ liệu đầu vào. Nếu MatLab thực hiện xong bài toán thì nó sẽ cho chung ta kết quả ở đầu ra.

```
>> d =
ans
        3.6056
```

Trường hợp không có kết quả hay kết quả sai thì có nghĩa MatLab chưa thực hiện được bài toán và chúng ta cần kiểm tra lại bài toán bằng cả hai phương pháp bằng tay và MatLab.

Ngày nay MatLab đã trở nên thông dụng và cực kỳ phổ biến ở hầu hết các trường đại học trên thế giới và là công cụ trợ giúp hữu hiệu cho các sinh viên, kỹ sư hay các nhà toán học trong nghiên cứu cũng như trong công việc thường ngày. Để giúp các bạn trẻ sinh viên có thêm một công cụ nữa trong tay, chúng tôi xin trình bầy sơ lược phần căn bản của MatLab và một số các ứng dụng của MatLab trong thực tế.

Bản in này chắc chắn sẽ có nhiều thiếu sót, chúng tôi rất mong được sự góp ý của các bạn đọc và các đồng nghiệp.

Các tác giả.

# CHƯƠNG 1

# *CÁC KHÁI NIỆM CƠ BẢN*

## 1.1. KHẢ NĂNG CỦA MATLAB

Matlab là gì ? Matlab hoạt động ra sao ?

Matlab có thể làm được những gì và ai có thể học và sử dụng Matlab ?

Sẽ có rất nhiều bạn đọc sẽ băn khoăn với các câu hỏi trên. Và ở chương này chúng ta sẽ cùng tìm ra lời giải đáp.

Matlab là chương trình phần mềm trợ giúp cho việc tính toán và hiển thị. Matlab có thể chạy trên hầu hết các hệ máy tính từ máy tính cá nhân đến các hệ máy tính lớn super computer.

Matlab được điều khiển bởi tập các bộ lệnh, tương tác bằng bàn phím trên cửa sổ điều khiển, đồng thời Matlab còn cho phép khả năng lập trình với cú pháp thông dịch lệnh hay còn gọi là script file. Các lệnh, bộ lệnh của Matlab lên đến con số hàng trăm và ngày càng được mở rộng bởi các phần Tools box trợ giúp, hay các hàm ứng dụng tạo ra bởi người sử dụng.

Các lệnh của Matlab rất mạnh và hiệu quả cho phép giải các loại hình toán khác nhau và đặc biệt hiệu quả cho các hệ phương trình tuyến tính cũng như thao tác trên các bài toán ma trận. Khônng những thế Matlab còn rất hữu hiệu trong việc trợ giúp thao tác và truy xuất đồ hoạ trong không gian 2D cũng như 3D cũng khả năng tạo hoạt cảnh cho việc mô tả bài toán một cách sinh động.

Cùng với trên 25 Tools box (thư viện trợ giúp) khác nhau Matlab đưa đến cho các bạn sự lựa chọn hoàn chỉnh và phong phú về các công cụ trợ giúp đắc lực cho những lĩnh vực khác nhau trên con đường nghiên cứu mà các bạn đã lựa chọn.

Dưới đây chúng tôi xin liệt kê một số lĩnh vực mà Matlab đã và đang giải quyết một cánh hiệu quả.

- Nghiên cứu và phát triển trong lĩnh vực công nghiệp.
- Giảng dạy, nghiên cứu lập các chương trình ứng dụng trong giảng dạy cho các môn kỹ thuật như toán, lý, hoá ... trong các trường phổ thông nhằm nâng cao khả năng tiếp thu cũng như ý sáng tạo trong học sinh.
- Giảng dạy và lập các chương trình giảng dạy về toán đặc biệt là các loại hình nguyên lý cơ bản và các phương trình tuyến tính cho sinh viên cũng như học sinh các trường kỹ thuật.
- Giảng dạy và nghiên cứu trong lĩnh vực kỹ thuật và khoa học bao gồm như: điện tử, lý thuyết điều khiển, vật lý , đồ hoạ , xử lý ảnh, vật liệu ...
- Giảng dạy và nghiên cứu trên mọi lĩnh vực có xuất hiện tính toán bao gồm toán kinh tế, hoá, cơ học, sinh học ...

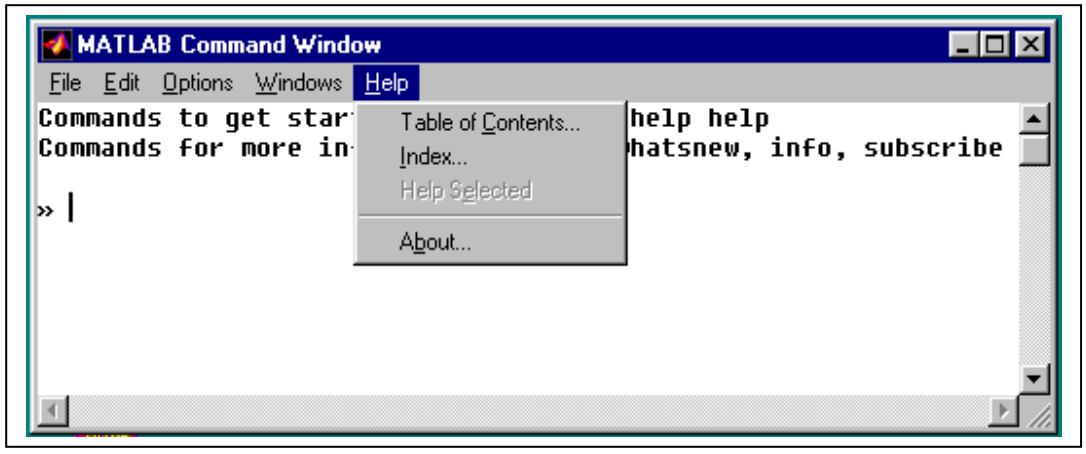
## 1.2 HOẠT ĐỘNG CỦA MATLAB

Muốn sử dụng được phần mềm MATLAB, trước tiên bạn phải cài đặt nó. Vì việc cài đặt MatLab cực kỳ dễ dàng nên ở đây chúng ta không đề cập đến. Tuy nhiên các bạn nên chú ý khi cài đặt nếu các bạn muốn sử dụng thêm các ToolBox của MatLab như Simulink, Fuzzy ToolBox, DSP (digital signal processing) .v.v. hay muốn tích hợp với MatLab một số ngôn ngữ lập trình quen thuộc mà bạn không muốn xa cách như C, C++, Fortran ...

Chương trình ứng dụng ở đây thường có ở các phiên bản sau:

MatLab 3.5 trở xuống với môi trường hoạt động là MS-Dos.

MatLab 4.0, 4.2, 5.1, 5.2 ... hoạt động trong môI trường Windows.




*Hình 1.1 Giao diện màn hình khi khởi tạo Matlab 4.2*

(ở đây chúng ta sẽ có ngay lệnh cơ bản dành cho việc giới thiệu chương trình là : intro, demo, help help ).

Và các version MatLab khác cho môi trường tương tác Unix.

Việc khởi động Matlab trên mỗi hệ thống mỗi khác. Trong môi trường Window hay Macintosh chương trình thường được khởi động thông qua việc click chuột trên các icon hay còn gọi là các biêu tượng. Còn với môi trường Unix, Dos thông qua dòng lệnh

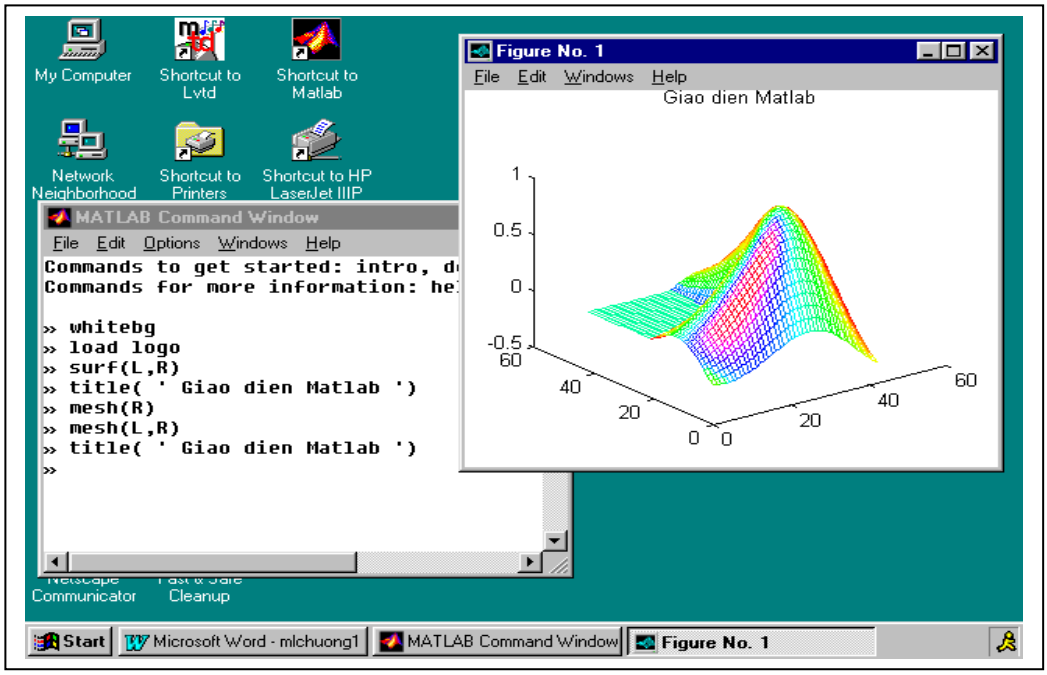
**:\Matlab**

Giao diện của MATLAB sử dụng 2 cửa sổ: cửa sổ thứ nhất được sử dụng để đưa các lệnh và dữ liệu vào đồng thời để in kết quả; cửa sổ thứ hai trợ giúp cho việc truy xuất đồ hoạ dùng để thể hiện những lệnh hay kết quả đầu ra dưới dạng graphics.

Việc ngắt chương trình đang thực hiện hoặc các chương trình thực hiện không đúng theo yêu cầu thông qua phím nóng Ctrl + C.

Để thoát ra khỏi môi trường làm việc Matlab, chúng ta có thể sử dụng lệnh của Matlab là :

```
>> quit                    %hoặc
>> exit
```




*Hình vẽ 1.2 Hai cửa sổ giao diện của Matlab*

### 1.2.1. Các phím chuyên dụng và các lệnh thông dụng hệ thống

| | |
|---|---|
| **↑ hoặc Ctrl + p** | Gọi lại lệnh vừa thực hiện trước đó |
| **↓ hoặc Ctrl + n** | Gọi lại lệnh đã đánh vào rất lâu trước đó |
| **→ hoặc Ctrl + f** | Chuyển con trỏ sang phải 1 ký tự |
| **← hoặc Ctrl + b** | Chuyển con trỏ sang trái 1 ký tự |
| **Ctrl + l hoặc Ctrl + ←** | Chuyển con trỏ tự sang trái 1 từ |
| **Ctrl + r hoặc Ctrl + →** | Chuyển con trỏ tự sang phải 1 từ |
| **Ctrl + a hay Home** | Chuyển con trỏ về đầu dòng |
| **Ctrl + k** | Xoá cho đến cuối dòng |

### Các lệnh hệ thống

| | |
|---|---|
| casesen off | -Bỏ thuộc tính phân biệt chữ hoa và chữ thường |
| casesen on | - Sử dụng thuộc tính phân biệt chữ hoa và chữ thường |
| clc | - Xoá cửa sổ dòng lệnh |
| clf | - Xoá cửa sổ đồ hoạ |
| computer | - Lệnh in ra một xâu kí tự cho biết loại máy tính |
| demo | - Lệnh cho phép xem các chương trình mẫu (minh hoạ khả năng của Matlab ) |
| exit, quit | - Thoát khỏi MATLAB |
| Ctr-c | - Dừng chương trình khi nó bị rơi vào tình trạng lặp không kết thúc |
| help | - Lệnh cho xem phần trợ giúp một số các lệnh được sử dụng trong Matlab |
| input | - Nhập dữ liệu từ bàn phím |
| load | - Tải các biến đã được lưu trong 1 file đưa vào vùng làm việc. |
| pause | - Ngừng tạm thời chương trình |
| save | - Lưu giữ các biến vào file có tên là *matlab.mat* |

## 1.3. BIẾN VÀ THAO TÁC CỦA CÁC BIẾN

### 1.3.1 Biến trong Matlab

Tên các biến trong Matlab có thể dài 19 ký tự bao gồm các chữ cái A-Z hay a-z cùng các chữ số cũng như 1 vài các ký tự đặc biệt khác nhưng luôn phải bắt đầu bằng chữ cái. Tên của các hàm đã được cũng có thể được sử dụng làm tên của biến với điều kiện hàm sẽ không được sử dụng trong suốt quá trình tồn tại của biến cho đến khi có lệnh clear xoá các biến trong bộ nhớ hay clear + tên của biến.

**clear** - Xoá cửa sổ đang sử dụng, xoá vùng nhớ dành cho các biến. Trong trường hợp này tất cả các biếnđược định nghĩa trước đó đều bị xoá.

**clear name** - Chỉ xoá biến có tên là name

**clear name1, name2, ...** - Chỉ xoá biến có tên được liệt kê sau lệnh clear ( name1, name2 ...)

**clear value** - Xoá biến theo giá trị cho trước

**pack** Lệnh được thực hiện nhằm mục đích sắp xếp lại các biến cũng như vùng chứa biến của bộ nhớ. Khi bộ nhớ của máy tính đầy lệnh pack cho phép tạo ra thêm vùng bộ nhớ cho biến mà không phải xoá đi các biến đã tồn tại. Công việc được thực hiên như sau:

1. Tất cả các biến trong bộ nhớ được lưới lại trên đĩa dưới file pack.tmp.

2. Vùng bộ nhớ cơ sở sẽ được giải phóng

3. Các biến sẽ được nạp (load) vào bộ nhớ từ file pack.tmp

4. File pack.tmp bị huỷ bỏ

**pack filename** Sắp xếp lại bộ nhớ với file trung gian có tên là: filename

Bình thường Matlab phân biệt khác nhau các biến tạo bởi chữ cái thường và chữ cái hoa. Các lệnh của Matlab nói chung thường sử dụng chữ cái thường. Việc phân biệt đó có thể được bỏ qua nếu chúng ta thực hiện lệnh

```
>> casensen
```

Kiểm tra sự tồn tại của các biến trong bộ nhớ thông qua bộ lệnh

**who** Hiển thị danh sách các biến đã được định nghĩa

**whos** Hiển thị các biến đã được định nghĩa cùng kích thước của

chúng và thông báo chúng có phải là số phức không.

**who global**

Hiển thị các biến cục bộ

**exist( namestr )**

Hiển thị các biến phụ thuộc vào cách các biến được định nghĩa trong chuỗi namestr. Hàm sẽ trả lại giá trị sau:

Nếu namestr là tên của 1 biến

Nếu namestr là tên của 1 file.m

Nếu namestr là tên của 1 MEX file

Nếu namestr là tên của hàm dịch bởi SIMULINK

Nếu namestr là tên của hàm được định nghĩa trước bởi Matlab .

### 1.3.2 Độ lớn của biến

Độ lớn hay chiều dài của biến vector cũng như ma trận có thể được xác định thông qua 1 số hàm có sẵn của Matlab.

**size ( A )**

Cho ra 1 vector chứa kích thước ma trận A. Phần tử đầu tiên của vector là số hàng của ma trận, phần tử thứ 2 là số cột của ma trận.

**[ m n ] = size( A )**

Trả giá trị độ lớn của ma trận A vào vector xác định bởi 2 biến m và n.

**size (A, p)**

Đưa ra giá trị số hàng của ma trận A nếu p <1 và số cột của A nếu p >= 2.

**size(x)**

Đưa ra vector mô tả độlớn của vector x. Nếu x là vector hàng m phần tử thì giá trị đầu của vector là m và giá trị thứ 2 là 1. Tường hợp x là vector n cột thì giá trị thứ nhất sẽ là 1 và thứ 2 là n.

**lengh(x)**

Trả giá trị chiều dài của vector x

**lengh(x)**

Trả giá trị chiều dài của ma trận A. Giá trị thu được sẽ là m nếu m>n và ngược lại sẽ là n néu n>m.

### 1.3.3 Một số biến được định nghĩa trước.

**ans**

Biến cho trước được gán cho phép tính cuối cùng của

công việc tính toán không biến gán.

**esp**

Trả ra độ chính xác của máy xác định bởi khoảng từ 1 đến 1 biến dấu phẩy động tiếp đó. Biến esp được sử dụng như là sai số trong 1 vài các phép toán. Người sử dụng có thể gán giá trị mới cho esp nhưng giá trị đó sẽ không bị xoá đi bởi hàm clear.

**realmax**

Đưa ra giá trị của số lớn nhất mà máy tính (chương trình) có thể tính toán được .

**realmin**

Đưa ra giá trị của số nhỏ nhất mà máy tính (chương trình) có thể tính toán được .

### 1.3.4 Số phức

#### a) Các phép toán đối với số phức:

| Phép toán | Kết quả |
|---|---|
| $c_1 + c_2$ | $(a_1 + a_2) + i(b_1 + b_2)$ |
| $c_1 - c_2$ | $(a_1 - a_2) + i(b_1 - b_2)$ |
| $c_1 . c_2$ | $(a_1 . a_2 - b_1 . b_2) + i(a_1 . a_2 + b_1 . b_2)$ |
| c1 | (a1. a2 - b1. b2) + i (a1. a2 - b1. b2) |
| c2 | $\frac{a_1^2+b_2^2}{a_1^2+b_2^2}$ |
| \| c1\| | $\sqrt{a_1^2 + b_2^2}$ (Độ lớn hay trị tuyệt đối của $c_1$) |
| $a_1^*$ | $a_1 - ib_1$ (số liên hợp của số phức) |

#### *b)* Một số hàm đặc biệt của số phức

**real(x)**

Hàm cho giá trị phần thực của số phức x. Nếu *x=a+ib* thì *real(x)=a*

**imag(x)**

Hàm trả lại giá trị phần ảo của số phức x. Nếu *x=a+ib* thì *imag(x)=b*

**conj(x)** Tính số liên hợp của số phức. Nếu *x=a+ib* thì *conj(x)=a-ib*

**abs(x)** Tính độ lớn, giá trị tuyệt đối của số phức.

**angle(x)** Tính góc có giá trị là *atan2(imag(x), real(x))*, giá trị góc nằm trong khoảng -π đến π.

## c) Toạ độ biểu diễn só phức

Chúng ta có thể biểu diễn số phức *a+ib* trên hệ trục toạ độ. Đối với hệ trục toạ độ đề các phần thực được biểu diễn trên trục x: *x=a*, phần ảo được biểu diễn trên trục y: *y=b*. Đối với hệ toạ đội cực số phức được biểu diễn bởi r, θ.

Trong đó:

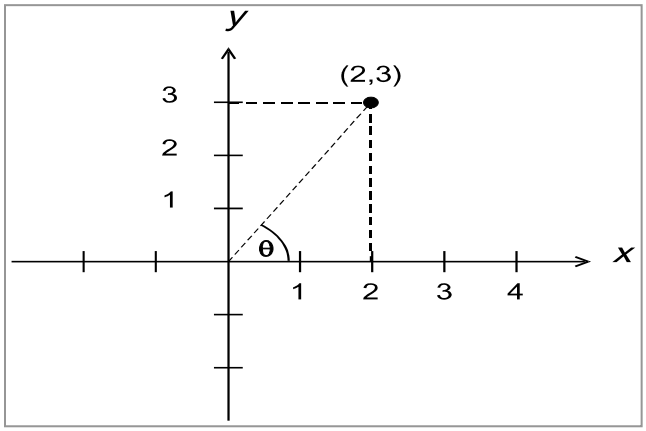
$$r = \sqrt{a_1^2 + b_2^2}$$

$$\theta = \tan^{-1} \frac{b}{a}$$

Ngược lại :

$a = r\cos\theta$

$b = r\sin\theta$




*Hình vẽ biểu diễn toạ độ số phức*

Trong hệ toạ độ cực: độ lớn *(magnitude)*, và pha *(phase)* của số phức sẽ được tính toán như sau:

```
>> r = abs(x);
>> theta = angle(x);
```

Biểu diễn số phức theo độ lớn và pha như sau:

```
>> y = r*exp(i*theta);
```

Trong hệ toạ độ đề các, phần thực *(real)* và phần ảo *(imaginary)* sẽ được tính toán như sau:

```
>> a = real(x);
>> b = imag(x);
```

Biểu diễn số phức: y = a + ib;

## 1.4. SƠ LƯỢC VỀ ĐỒ HOẠ TRONG MATLAB

MATLAB sử dụng lệnh X-Y Plots để vẽ đồ thị, biểu đồ cho các thông tin một các dễ dàng. Trong phần này, vẽ đồ thị tổng quát theo dữ liệu được lưu trong hai vector x,y. Trong trường hợp các biểu đồ hay đồ thị mong muốn được biểu diễn dưới dạng mẫu 3D thì đơn giản với Matlab chúng ta chỉ cần đổi sang dùng lệnh X-Y-Z Pots để vẽ.

### 1.4.1 Vẽ trong cửa sổ đồ hoạ của Matlab

**plot(x,y)** Vẽ đồ thị theo toạ độ x-y

**pot3(x,y,z)** Vẽ đồ thị theo toạ độ x-y-z

**title** Đưa các title và trong hình vẽ

**xlabel** Đưa các nhãn theo chiều x của đồ thị

**ylabel** Đưa các nhãn theo chiều y của đồ thị

**zlabel** Đưa các nhãn theo chiều z của đồ thị

**grid** Vẽ các đường gióng grid line trên đồ thị

**plot(y)** Vẽ đồ thị theo y bỏ qua chỉ số theo y

Nếu y là số ảo thì đồ thị được vẽ sẽ là phần thực và phần ảo của y.

```
>> plot( real ( y ), image ( y ) )
```

**plot(x,y,S)** plot ( x, y, s ) vẽ theo x,y

**plot(x,y,z,S)** plot ( X, Y, Z, S ) vẽ theo x, y, z với

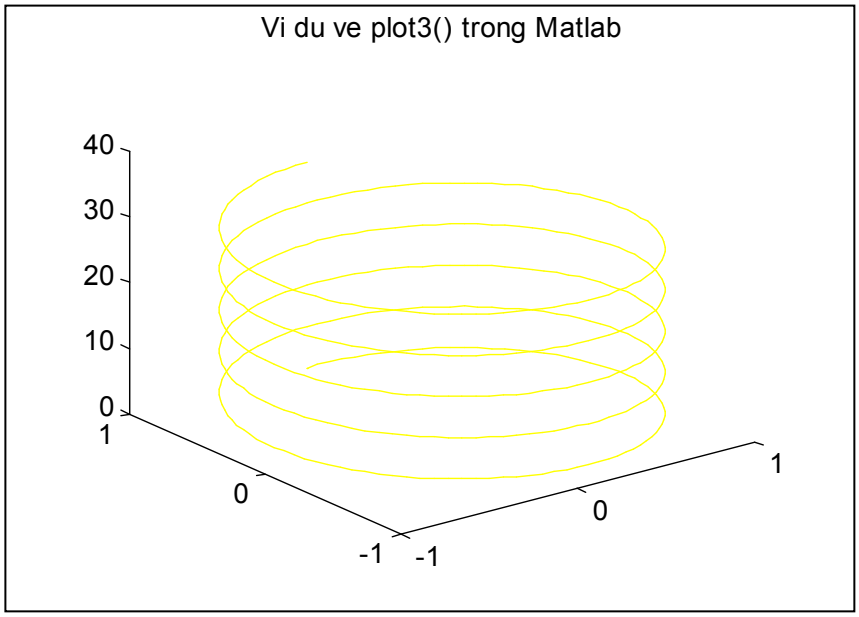
s là các chỉ số sẽ liệt kê ở chương sau

Ví dụ:

```
>> plot ( x, y, 'b+' )
```

Vẽ đồ thị theo x và y với mầu của đường là mầu xanh dương và ký tự tạo nên đường là dấu +

## Ví dụ hình helix:




*Hình 1.3 Hình Helix*

```
>>  t = 0:pi/50:10*pi;
>>  plot3(sin(t),cos(t),t);
>> Title(' Ví du ve plot3() trong Matlab ')
```

Matlab rất mạnh trong việc xử lý đồ hoạ. Ta sẽ đề cập vấn đề này rõ hơn ở chương sau.

### 1.4.2 In ấn trên màn hình đồ hoạ

Việc in các ảnh trên màn đồ hoạ có thể được thực hiện thông qua các menu lệnh may các lệnh của Matlab.

```
>> print
```

- In màn hình của cửa sổ đồ hoạ hiện thời ra máy in.

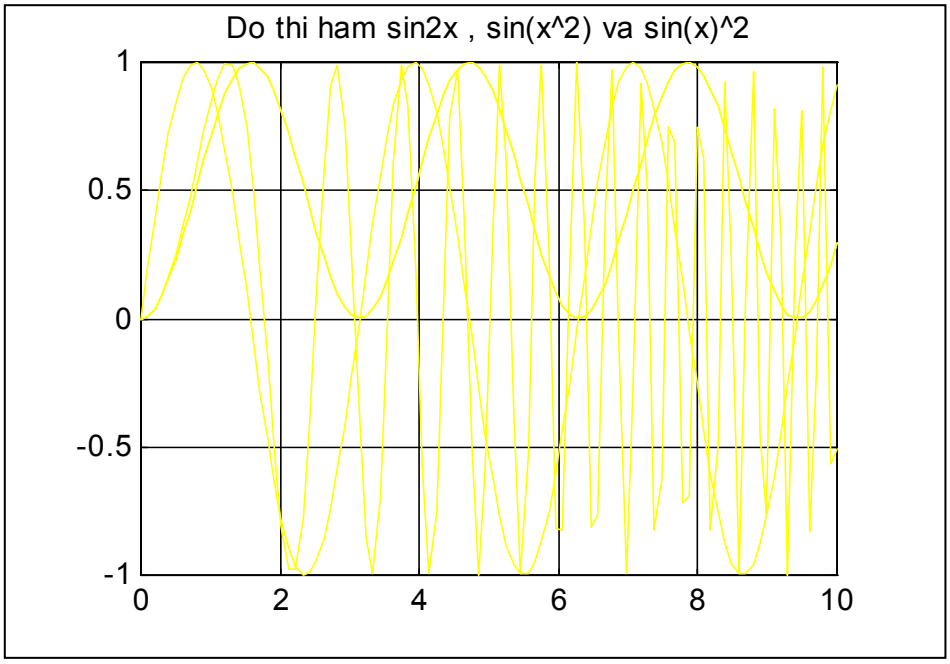
```
>> print filename
```

- In màn hình đồ hoạ ra file

```
>> print  esp filename
```

- copy màn đồ hoạ theo khuôn dạng eps. File thu được có thể đưa vào các trang văn bản

### 1.4.3 Một số ví dụ mô tả đồ hoạ .




*Hình 1.4 Hàm sin2x, cos(x)^2 và cos(x^2)*

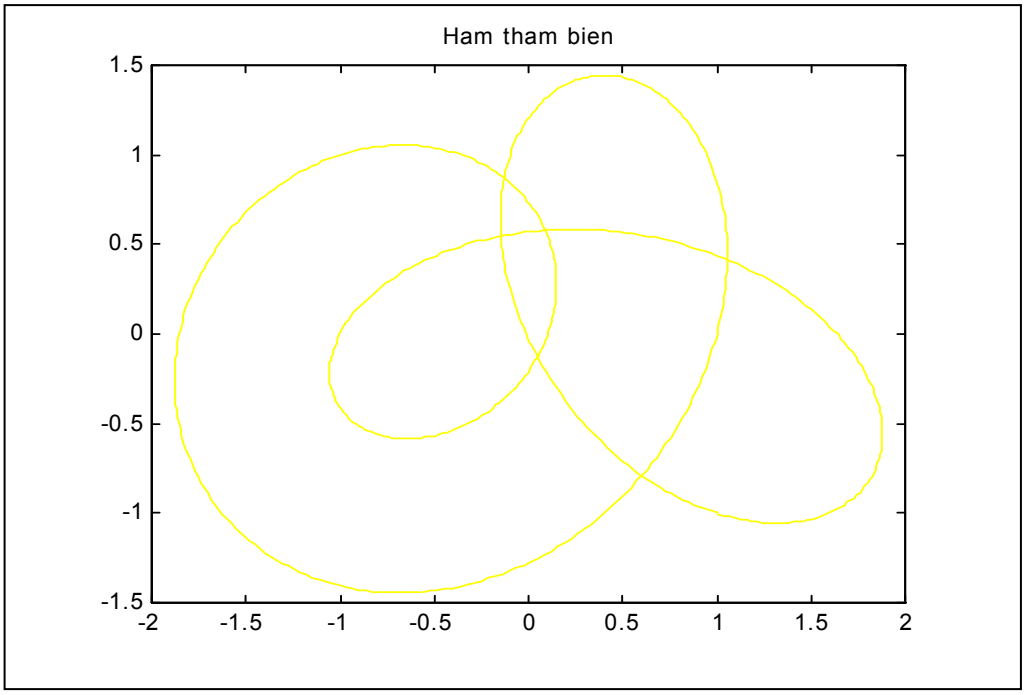
a) Ví dụ mô tả khả năng vẽ hàm đồ hoạ trong không gian 2D. Giả sử với hàm sin2x, cos(x)^2 và (cosx)^2 trong khoảng 0 < x < 10. Việc thao tác dễ dàng trên tập các lệnh sau.

```
>> hold on
>> x = línspace(0,10);
>> y1 = sin(2*x);
>> y2 = sin(x.^2);
>> y3 = (sin(x)).^2;
>> plot ( x,y1 ); plot ( x,y2 ); plot( x,y3 );
```

Hàm plot ( x,y ) sẽ cho ra trên màn hình đồ hoạ hàm y theo vector x

b) Hàm mô tả đường cong tham biến trong không gian 2D và 3D

Đường cong tham biến theo t với t trong khảng từ [ 0 2*pi ] cho kết quả Hình vẽ 1.5.




*Hình 1.5 Hàm tham biến 2D*

```
>> t = 0 : 0.01 : 2*pi ;
>> x = cos(t) – sin(3*t)
>> y = sin(t).*cos(t) – cos(3*t )
>> title(‘ Hàm tham biến ‘);
>> plot (x,y)
```

Với giá trị của t trong khảng từ [ 0  2*pi ] và khoảng của u là [ 0  1 ].

Đoạn chương trình sau cho ra hình vẽ hàm tham biến 3D.

```
>> t = 0 : 0.01 : 1 ;
>> u = 0: 0.01; 1;
>> x = u.*cos(t)./30 + 10;
>> y = u.*sin(t)./55 + 10;
>> z = 1;
>> title(‘ Hàm tham biến 3D ‘);
>> plot (x,y,z)
```

## 1.5. CÁC HÀM ÂM THANH TRONG MATLAB

Matlab cho phép tạo âm thanh thông qua các vector bởi lệnh sound.

| | |
|---|---|
| **sound ( y )** | - Gửi tín hiệu của vector y ra loa. Vector được xắp xếp với biên độ lớn nhất |
| **sound ( y , f )** | - Thực hiện công việc như hàm sound (y) với f là dải tần đo bởi Hz. Lệnh này không thực hiện trên các hệ máy SunSPARE. |
| **saxis** | - Trả giá trị giới hạn của trục âm thanh trong vector hiện hành. |
| **axis( [min max] )** | - Xét thang của trục âm thanh. Tăng giá trị sẽ cho âm thanh trầm hơn. Giảm giá trị sẽ cho âm thanh ồn hơn. |
| **saxis (str)** | - Xét trục âm thanh theo chuỗi srt. |

Ví dụ :

a) Tạo sóng hình sin trong khoảng sau:

```
>> x = sin ( linspace(0,10000,10000) );
>> sound ( x );
```

b) Một vài ví dụ với các âm thanh có sẵn được đưa ra bởi lệnh load.

```
>> load train;          % giá trị của âm thanh tầu hoả
>> sound ( y );         % được đưa vào tham số  y
>> load chirp;              % tiếng chim kêu
>> sound ( y );
```

Với Matlab trên hệ MS -Window cho phép người sử dụng thao tác với file âm thanh định dạng wav bằng bộ lệnh sau:

wavread ( fstr ) ≡ [y] = wavread (wavfile)

- Đọc dữ liệu âm thanh từ file.wav xác định bởi chuỗi fstr vào tham biến y.

[ y, Fs ] = wavread ( ... ) như trên với fs là tần số

wavwrite ( sv, f , wavfiles )

- Ghi dữ liệu âm thanh từ vector sv với tần số f vào file xác định bởi biến wavefile

# CHƯƠNG 2

# MA TRẬN VÀ CÁC PHÉP TOÁN CHO MA TRẬN

Trong phần này, ta sẽ xem xét các biến đơn, các đại lượng vô hướng cùng với các biến ma trận cùng các phép tính cơ bản, các hàm chức năng sẵn có và các toán tử được sử dụng trong phần mềm Matlab.

## 2.1 VECTOR - ĐẠI LƯỢNG VÔ HƯỚNG VÀ MA TRẬN

Khi giải quyết một vấn đề kỹ thuật nào đó, điều quan trọng là phải xem xét các dữ liệu liên quan tới vấn đề đó. Một số dữ liệu có giá trị đơn như diện tích hình vuông, một số dữ liệu liên quan tới nhiều đại lượng như toạ độ 1 điểm trong không gian gồm 3 giá trị x,y,z ...

Tất cả những dữ liệu này có dạng cấu trúc đặc biệt gọi là ma trận *(matrix).* Các phần tử của ma trận được sắp xếp theo hàng và cột. Một giá trị đơn có thể coi là một ma trận chỉ có duy nhất 1 hàng và 1 cột hay còn gọi là đại lượng vô hướng *(scalar).* Ma trận chỉ có một hàng hoặc một cột được gọi là vector. Để cập nhật tới 1 phần tử của ma trận ta sử dụng chỉ số hàng và cột của nó *(subscripts).*

Ví dụ: $C_{4,3}$

Kích thước của ma trận được thể hiện mxn có nghĩa là có m hàng và n cột.

### *2.1.1 Cách nhập giá trị cho ma trận hay các đại lượng vô hướng*

Có 4 cách liệt kê sau đây cho việc vào dữ liệu cho các biến vô hướng hay ma trận.

+ Liệt kê trực tiếp các phần tử của ma trận

+ Có thể đọc dữ liệu từ một file dữ liệu.

+ Sử dụng toán tử *(:).*

+ Vào số liệu trực tiếp từ bàn phím.

### ** Một số các quy định cho việc định nghĩa ma trận*

Tên ma trận phải được bắt đầu bằng chữ cái và có thể chứa tới 19 ký tự là số, chữ cái, hoặc dấu gạch dưới được đặt ở bên trái dấu bằng.

Bên phải của dấu bằng là các giá trị của ma trận được viết theo thứ tự hàng trong dấu ngoặc vuông.

Dấu chấm phẩy (;) phân cách các hàng. Các giá trị trong hàng được phân cách nhau bởi dấu phẩy (,) hoặc dấu cách; các giá trị có thể là số âm hay dương. Dấu thập phân được thể hiện là dấu chấm (.). Khi kết thúc nhập một ma trận phải có dấu (;).

#### a. Liệt kê trực tiếp:

Là cách định nghĩa ma trận một cách đơn giản nhất. Các phần tử của ma trận được liệt kê trong dấu ngoặc vuông.

```
>> A=[3,5];
>> B=[1.5,3.1];
>> C=[-1,0,0; -1,1,0; 1,-1,0; 0,0,2];
```

Có thể xuống dòng để phân biệt từng hàng ma trận.

Ví dụ:

```
>>C=[    -1   0   0
        -1   1   0
         1  -1   0
        0   0   2 ];
```

Khi số phần tử trên một hàng của ma trận quá lớn, ta có thể dùng dấu ba chấm (...) để thể hiện số phần tử của hàng vẫn còn. Và tiếp tục viết các phần tử ở dòng tiếp theo.

Ví dụ: Vector F có 10 phần tử ta có thể viết như sau:

```
>> F = [ 1,   52,    64,    197,    42, -42,...
        55,    82,    22,    109 ];
```

Bạn có thể định nghĩa một ma trận từ một ma trận khác như sau

```
>> B = [ 1.5,  3.1 ];
```

```
>> S = [ 3.0,   B  ];
```

Ma trận S có thể hiểu như sau:  S = [ 3.0, 1.5,  3.1];

Bạn có thể cập nhật tới từng phần tử một bằng cách sử dụng chỉ số của nó:

```
>> S(2) = -1.0;
```

Giá trị của phần tử thứ 2 trong ma trận S sẽ thay đổi từ 1.5 thành -1.0.

Bạn có thể mở rộng ma trận bằng cách thêm cho nó phần tử mới. Thực hiện lệnh sau:

```
>> S(4) = 5.5;
```

Ma trận S lúc này sẽ có 4 phần tử:  S = [ 3.0, -1.0,  3.1, 5.5 ];

Nếu ta thực hiện lệnh này:

```
>> S(8) = 9.5;
```

Thì ma trận S sẽ có 8 phần tử, các phần tử S(5), S(6), S(7) sẽ tự động nhận giá trị là 0.

### b. Có thể đọc dữ liệu từ một file dữ liệu đã có:

Thông qua lệnh load cho phếp nhập vào dữ liệu của ma trận lưu trữ trước trong đĩa

### c. Sử dụng toán tử (:)

Dấu hai chấm (:) được sử dụng để tạo vector từ ma trận. Điều này tạo điều kiện cho thuận lợi trong việc xử lý số liệu.

- Ví dụ: Muốn vẽ biểu đồ theo hệ toạ độ x,y cho 1 file dữ liệu nào đó, ta dễ dàng ghi các số liệu x vào 1 vector và các số liệu y vào 1 vector khác.

Tại vị trí của dấu (:) trong ma trận, nó đại diện cho tất cả các hàng hoặc tất cả các cột.

- Ví dụ: Các lệnh sau đây sẽ đưa tất cả các dữ liệu ở cột thứ nhất trong ma trận data1 vào vector x và toàn bộ dữ liệu ở cột thứ 2 của ma trận vào vector y:

```
>> x = data1 (: , 1);
>> y = data1 (: , 2);
```

Dấu hai chấm còn có thể sử dụng làm ký hiệu tổng quát trong ma trận mới. Nếu dấu hai chấm nằm ở giữa 2 số nguyên, thì nó đại diện cho tất cả các sõ nguyên nằm giữa 2 số nguyên đó. Ví dụ: dấu 2 chấm là ký hiệu tổng quát của vector H có chứa các số từ 1 đến 8.

```
>> H = 1:8;
```

Nếu dấu hai chấm nằm ở giữa 3 số, thì dấu 2 chấm đại diện cho tất cả các số có giá trị từ số thứ nhất đến số thứ 3, số thứ 2 được sử dụng làm mức tăng.

- Ví dụ: dấu 2 chấm là ký hiệu tổng quát trong vector hàng có tên TIME có chứa các số từ 0.0 đến 5.0 có mức tăng là 0.5:

```
>> TIME = 0.0 : 0.5 : 5.0;
```

Mức tăng âm được thể hiện trong ví dụ sau:

```
>> VALUES = 10 : -1 : 0;
```

Dấu hai chấm còn được sử dụng để chọn các ma trận con từ 1 ma trận khác.

- Ví dụ: Giả sử có ma trận C được cho như sau:

```
>> C=[ -1   0   0
  -1   1   0
   1  -1   0
  0   0   2 ];
```

Dùng lệnh:

```
>> C_PARTIAL_1 = C( : ,2:3);
>> C_PARTIAL_2 = C(3:4,1:2);
```

Ta sẽ nhận được ma trận sau:

```
C_PARTIAL_1 =[ 0   0
               1   0
              -1   0
               0   2  ];
```

```
C_PARTIAL_2 =[1   -1
              0    0 ];
```

Nếu dấu hai chấm định nghĩa các chỉ số không hợp lệ như C(5:6,:), thì sẽ có hiển thị thông báo lỗi.

Trong MATLAB ma trận rỗng *(empty matrix)* là giá trị hợp lệ. Ma trận rỗng có thể được định nghĩa như sau:

```
>> A = [ ];
>> B = 4 : -1 : 5
```

Ma trận rỗng khác với ma trận chỉ toàn số 0.

Cuối cùng, C(:) tương đương với một cột dài có chứa cột đầu tiên của ma trận C, tiếp đến là cột thứ hai của ma trận C, và cứ như vậy tiếp tục. Đây là toán tử rất mạnh của Matlab.

### d. Vào số liệu trực tiếp từ bàn phím.

Ta có thể nhập ma trận từ bàn phím.

Cú pháp:

```
>> Z = input('Nhập giá trị cho Z');
```

Khi thực hiện lệnh này, máy sẽ hiển thị xâu ký tự 'Nhập giá trị cho Z' và đợi người sử dụng nhập số liệu vào. Người sử dụng có thể gõ một biểu thức như sau [5.1 6.3 -18.0] để xác định giá trị của Z. Nếu người sử dụng chỉ gõ enter mà không nhập giá trị nào vào thì ma trận Z sẽ được coi là ma trận rỗng. Nếu lệnh kết thúc với dấu (;) thì giá trị của Z sẽ được hiển thị. Nếu không có dấu (;) thì không được hiển thị.

## 2.1.2 Hiển thị ma trận

Có nhiều cách để hiển thị ma trận. Cách đơn giản nhất gõ tên của ma trận rồi enter. Tuy nhiên, có một số lệnh được dùng để hiển thị ma trận với các phần tử ma trận được biểu diễn theo nhiều kiểu khác nhau.

Dạng mặc định là 5 chữ số có nghĩa sau dấu thập phân (gọi là *short format).* Một số dạng hiển thị khác được liệt kê dưới đây:

**format long** — Dạng số chữ số có nghĩa dài (15 chữ số có nghĩa sau dấu thập phân trở lên)

**format short** — Còn gọi là *default format* (có 5 chữ số có nghĩa)

**format short e** — Dạng số phẩy động ngắn (dưới $10^{15}$)

**format long e** — Dạng số phẩy động lớn (từ $10^{15}$ trở lên. Ví dụ: 6.023e+23)

**format** — Hiển thị dấu (âm, dương) của các phần tử của ma trận.

**format compact** — Cho phép giảm khoảng cách giữa các phần tử trong ma trận

**format loose** — Huỷ bỏ lệnh *format compact* trở lại chế độ hiển thị thông thường.

**disp** — Hiển thị thông báo trong dấu ngoặc đơn hoặc hiển thị nội dung của ma trận.

Ví dụ:

```
>>  disp(temp); disp(' độ F ');
```

Ta sẽ nhận được: 78 độ F

Trong đó *temp* là tên của ma trận chứa 1 giá trị nhiệt độ F là 78.

**fprintf**

Lệnh này cho phép in tham số đầu ra theo đúng dạng mà ta mong muốn: cả text và cả giá trị số. Trong lệnh này có thể có chứa cả những dòng trống. Cú pháp của nó như sau:

```
>> fprint( định dạng, ma trận);
```

Trong định dạng có thể chứa cả text và các ký hiệu dạng đặc biệt (%e, %f,%g, /n — được ghi trong cặp dấu nháy đơn) điều khiển cách in các giá trị của ma trận. Nếu sử dụng:

%e các giá trị được in ra dưới dạng số phẩy động.

%f các giá trị được in ra dưới dạng số phẩy tĩnh.

%g thì giá trị được in ra có thể có dạng số phẩy động hoặc tĩnh tuỳ thuộc vào bản thân nó.

\n thì 1 dòng trống sẽ được in ra.

Ví dụ:

```
>> fprintf( 'Nhiệt độ là: \n %4.1f độ F \n', temp);
```

Nghĩa là số vị trí dành để in giá trị của biến *temp* là 4 và một số sau dấu phảy.

Nó sẽ được hiển thị như sau:

Nhiệt độ là: 78.0 độ F

## 2.2 CÁC MA TRẬN ĐẶC BIỆT:

Matlab có sẵn một số hàm lưu các hằng, giá trị đặc biệt và các ma trận đặc biệt.

MATLAB có một số hàm để tạo ra các ma trận đặc biệt.

### 2.2.1 Ma trận ma phương (magic( n ) )

Ma phương bậc n là ma trận vuông cấp n bao gồm các số nguyên từ 1 đến $n^2$. Các số nguyên được sắp xếp sao cho tổng các phần tử trên một hàng, một cột, đường chéo là bằng nhau. Hàm của ma trận ma phương tổng quát chỉ cần một tham số là bậc của nó.

Ví dụ:

```
>> magic(4)
```

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

### 2.2.2 Ma trận 0 ( zeros )

Hàm zeros(m,n) là ma trận có kích thước mxn chứa toàn số 0. Nếu tham số của hàm chỉ có 1 giá trị thì hàm là ma trận vuông. Để tạo ra ma trận 0, dùng hàm zeros(n), zeros(m,n), zeros(A) với A là ma trận bất kỳ.

Ví dụ:

```
>> zeros ( 4 , 4 )
```

```
ans =
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

### 2.2.3 Ma trận 1 ( ones )

Hàm *ones* được định nghĩa giống như hàm *zeros* nhưng số 0 được thay bởi số 1.

ví dụ:

```
>> ones( 4 , 4 )
```

```
ans =
     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

### 2.2.4 Ma trận đường chéo đặc biệt (Identity Matrix)

Ma trận đường chéo là ma trận có các phần tử nằm trên đường chéo chính là 1, còn các phần tử ở vị trí khác là 0.

Ví dụ:

```
>> eye ( 4 )
```

ans =

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

Chú ý là không chỉ có ma trận vuông mới có đường chéo chính mà khái niệm này còn mở rộng cho cả ma trận chữ nhật.

### *2.2.5 Ma trận đường chéo mở rộng eye( m,n )*

Là ma trận đường chéo mở rộng với ma trận hình chữ nhật có m hàng, n cột. Các phần tử có chỉ số hàng và cột bằng nhau có giá trị là 1, tại các vị trí khác các phần tử có giá trị là không. Khi hàm chỉ có 1 giá trị tham số thì ma trận đường chéo mở rộng sẽ trở thành ma trận đường chéo. Ma trận này được tạo ra bởi hàm *eye(m,n); eye(n); eye (C)* (giống các định nghĩa trên).

Ví dụ:

```
>> eye ( 4,5 )
```

ans =

1 0 0 0 0

0 1 0 0 0

0 0 1 0 0

0 0 0 1 0

### 2.2.6 Ma trận Pascal ( pasca (n) )

Là ma trận chứa các giá trị của tam giác Pascal.

Ví dụ:

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

ans =

1 1 1 1

1 2 3 4

1 3 6 10

1 4 10 20

### *2.2.7 Các ma trận đặc biệt khác*

| | |
|---|---|
| **compan** | Companion matrix. |
| **gallery** | Several small test matrices. |
| **hadamard** | Hadamard matrix. |
| **hankel** | Hankel matrix. |
| **hilb** | Hilbert matrix. |
| **invhilb** | Inverse Hilbert matrix. |
| **kron** | Kronecker tensor product. |
| **rosser** | Classic symmetric eigenvalue test problem. |
| **toeplitz** | Toeplitz matrix. |
| **vander** | Vandermonde matrix. |
| **wilkinson** | Wilkinson's eigenvalue test matrix. |

## 2.3 CÁC PHÉP TOÁN VÔ HƯỚNG

### *2.3.1 Biểu thức số học:*

| Phép toán | Biểu thức số học | MATLAB |
|---|---|---|
| Cộng | a + b | a + b |
| Trừ | a - b | a - b |
| Nhân | a x b | a * b |
| Chia | a/b | a/b |
| Chia phải | a : b | a : b |
| Chia trái | b : a | b : a |
| Luỹ thừa | $a^b$ | a ^ b |

Ví dụ:

```
>> a = 3 ; b = 1.2;        % Phép nhập dữ liệu
>> a + b                   % Phép cộng ( trừ )
```

ans =

```
        4.2000
>> a/b                    % Phép chia ( nhân )
    ans =
        2.5000
>> b : a                  % Phép chia trái
    ans =
        1.2000    2.200
>>  a^b                   % Phép lũy thừa
    ans =
        3.7372
```

### 2.3.2 Thứ tự ưu tiên của các toán tử:

Tuy nhiên một số toán tử có thể kết hợp trong một biểu thức số học, khi đó điều quan trọng nhất là phải biết thứ tự ưu tiên của các toán tử trong biểu thức.

| Thứ tự ưu tiên | Toán tử |
|---|---|
| 1 | Ngoặc đơn |
| 2 | luỹ thừa |
| 3 | nhân và chia, từ trái qua phải |
| 4 | cộng và trừ, từ trái qua phải |

Ví dụ:

$$\frac{x^3 - 2x^2 + x - 6.3}{x^2 + 0.05005x - 3.14}$$

Nếu x là một giá trị vô hướng thì giá trị của f sẽ được tính theo các lệnh sau:

```
>> numerator = x^3 - 2*x^2 + x - 6.3;
>> denominator = x^2 + 0.05005*x - 3.14;
>> f = numerator/ denominator;
```

### 2.3.3 Các phép toán đối với vector

| Phép toán | Công thức | Viết dưới dạng Matlab |
|---|---|---|

| | | |
|---|---|---|
| Cộng | a + b | a + b |
| Trừ | a — b | a - b |
| Nhân mảng | a x b | a.*b |
| Chia phải mảng | a/b | a./b |
| Chia trái mảng | b/a | a.\b |
| Luỹ thừa mảng | $a^b$ | a.^b |

Các phép toán trên không chỉ áp dụng giữa các ma trận có kích thước bằng nhau mà còn áp dụng giữa các đại lượng vô hướng và đại có hướng.

Ví dụ: B = 3*A; C = A/5;

B = A.*; C = A./5;

Véc tơ B và C là véc tơ có kích thước bằng véc tơ A. Xét hai véc tơ như sau:

```
>> A = [ 2  5  6 ]
>> B = [ 2  3  5 ]
```

Tích hai véc là sẽ được viết như sau:

```
>> C = A.*B;
```

Véc tơ C sẽ chứa các phần tử như sau:

C = [ 4  15  30 ]

Matlab có hai phép chia:

Chia trái : C = A./B; % Giá trị của C thu được sẽ là: C = [ 1  1.667  1.2 ]

Chia phải: C = A.\B; % Giá trị của C thu được sẽ là: C = [ 1  0.6  0.883 ]

Toán tử mũ đối với véc tơ:

C = A.^2; % C = [ 4  25  36 ]

D = A.^B; % D = [ 4  125  7776 ]

E = 3.0.^A; % E = [ 9  243  729 ] Lệnh này còn có thể viết là: E = (3).^A;

Chú ý: E = 3.^A; % sẽ được xét sau.

E = 3 .^A; % Nếu có khoảng trống giữa số 3 và dấu chấm thì đúng

Các ví dụ trước xét cho các véc tơ, nhưng kết quả vẫn đúng cho các ma trận hàng và cột. Xét các lệnh sau:

C = [ 1:5; -1:-1:-5 ];

```
Z = ones(D);
S = D - Z;
P = D.*S;
SQ = D.^3;
```

Kết quả thu được sẽ là những ma trận như sau:

$$D = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ -1 & -2 & -3 & -4 & -5 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$S = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ -2 & -3 & -4 & -5 & -6 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 2 & 6 & 12 & 20 \\ 2 & 6 & 12 & 20 & 30 \end{bmatrix}$$

$$S = \begin{bmatrix} 1 & 8 & 27 & 64 & 125 \\ -1 & -8 & -27 & -64 & -125 \end{bmatrix}$$

Thông thường, các dữ liệu kỹ thuật được lưu dưới dạng ma trận. Để xử lý chúng một cách thuận tiện, phần mềm Matlab được xây dựng gồm nhiều hàm có thể xử lý các số liệu dưới dạng ma trận.

## 2.4. CÁC PHÉP TOÁN ĐỐI VỚI MA TRẬN

### *2.4.1 Ma trận chuyển vị*

Ma trận chuyển vị của ma trận A là một ma trận mới, trong đó cột của ma trận mới là hàng của ma trận gốc. Kí hiệu là $A^T$.

Ví dụ:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 \end{bmatrix}$$

Các phần tử hàng của ma trận A trở thành phần tử cột của ma trận $A^T$. Trong Matlab người ta kí hiệu ma trận chuyển vị là A'. Người ta sử dụng toán tử ma trận chuyển vị để chuyển vectơ hàng thành vectơ cột và ngược lại.

### 2.4.2 Tích vô hướng là tích của hai ma trận cùng cỡ

Kí hiệu toán học là:

$$\text{dot-product} = A.B = \Sigma\, a_i.b_i$$

Trong Matlab:

```
dot-product = sum(A.*B);
```

Nếu cả A và B đều là vectơ cột hoặc hàng thì A.*B cũng là một vectơ. Nếu A là vectơ hàng và B là vectơ cột thì tích vô hướng được tính như sau:

```
dot-product = sum(A'.*B) = sum(A.*B');
```

Ví dụ:

```
>> A = [ 1 2 3; 4 5 6 ];
>> B = [ 3 4 5; 6 7 8];
>> C = A.*B                    % Phép nhân vô hướng 2 ma trận A và B
C =
     3     8    15
    24    35    48
>> sum(C)
ans =
    27    43    63
```

### 2.4.3 Nhân ma trận

C = AB

Trong đó: $C_{i,j} = \Sigma\, A_{ik}B_{kj}$

Số hàng của ma trận A phải bằng số cột của ma trận B. Chú ý là AB#BA (có thể tồn tại tích AB nhưng không tồn tại tích BA). Kí hiệu phép nhân ma trận trong Matlab:

Ví dụ: Với dữ liệu cho trong hai ma trận A và B. Phép nhân ma trận được thực hiện dưới đây.

```
>> B = B';                 % Đảo ma trận B để có số hàng, cột thích hợp cho
>> C = A* B;               % phép nhân ma trận A,B
```

C =

26 44

62 107

* Phép luỹ thừa:

Cú pháp: A^k =(A*A*....*A) #A.^k

```
>> A = [ A(:,1) A( :,2)]   % Chích 2 cột 1 và 2 của ma trận A để tạo
```

A = % ma trận A vuông cho phép luỹ thừa

1 2

4 5

```
>> C = A^3                 % Phép lũy thừa của ma trận A
```

C =

57 78

156 213

### 2.4.4 Các thao tác ma trận:

**a) Rotation (phép quay):**

Cú pháp:

```
>> B = rot90(A);
```

Các phần tử của ma trận A được quay một góc 90$^o$ theo ngược chiều kim đồng hồ.

Ví dụ:

$$A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 5 & -1 \\ 3 & 4 & 6 \end{bmatrix} \Longrightarrow B = rot90(A) = \begin{bmatrix} 0 & -1 & 6 \\ 1 & 5 & 4 \\ 2 & -2 & 3 \end{bmatrix}$$

Hàm rot90 cũng có tham số thứ hai để xác định thực hiện số lần quay của các phần tử trong ma trận A.

Ví dụ:

```
>> B = rot90(A);
```

```
>>C = rot90(B);
```

Hai dòng lệnh trên tương đương với dòng lệnh sau với tham số lần quay là 2

```
>> C = rot90(A,2);
```

**b) Đảo ma trận:**

Trong Matlab có hai hàm được sử dụng để đảo một ma trận tạo ra ma trận mới:

*fliplr(A)* hàm đảo các phần tử của ma trận ma trận A từ trái sang phải.

Ví dụ:

```
>> A = [  1     2     3
          4     5     6
          7     8     9 ];
>> B = fliplr(A)
```

B =

3 2 1

6 5 4

9 8 7

*flipud(B)* hàm đảo các phần tử của ma trận B từ trên xuống dưới. Ma trận thu được kết quả như sau:

```
>> C = flipud (B)
```

C =

9 8 7

6 5 4

3 2 1

**c) Reshape:**

Hàm này cho phép định dạng lại ma trận với số hàng và số cột khác với ma trận gốc. Số phần tử của ma trận gốc và ma trận đã định dạng lại phải bằng nhau. Hàm có ba tham số: tham số đầu là ma trận gốc, hai tham số còn lại là số hàng và số cột của ma trận mới.

Ví dụ:

$$A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 5 & -1 \end{bmatrix}$$

```
    | 3  4  6  |
```

```
>> B = reshape(A,1,9)
```

```
B =

     2   -2   3   1   5   4   0   -1   6
```

**d) Trích các phần tử từ một ma trận**

Các hàm diag, triu, tril cho phép trích các phần tử từ một ma trận. Có 3 hàm liên quan tới đường chéo chính:

**diag(A)** Lấy các phần tử trên đường chéo chính và lưu vào một vectơ cột.

**diag(A,k)** Chọn đường chéo tuỳ thuộc giá trị k. k=0 chọn đường chéo chính.

k>0 chọn đường chéo thứ k ở trên đường chéo chính

k<0,chọn đường chéo thứ k ở dưới đường chéo chính.

**A=diag(V)** Nếu V là vectơ ta được ma trận vuông A với vectơ là đường chéo chính.

**B=triu(A)** Sinh ra ma trận B cùng cỡ chứa các phần tử của A nằm trên đường chéo chính và phía trên đường chéo chính. Các vị trí khác bằng 0.

**triu(A.k)** Là ma trận cùng cỡ với A chứa số phần tử từ A ở ngay trên và ở phía trên đường chéo thứ k, các vị trí khác bằng 0.

**tril(A)** Là ma trận cùng cỡ với A chứa số phần tử từ A nằm dưới đường chéo chính. Các vị trí khác bằng 0.

**tril(A,k)** Là ma trận cùng cỡ với A chứa số phần tử từ A ở ngay trên và ở phía dưới đường chéo thứ k, các vị trí khác bằng 0.

Ví dụ: Với dữ liệu là ma trận A đã cho sau

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

Các hàm trích phần tử của ma trận được viết và thể hiện kết quả trên màn hình thể hiện :

```
>> diag(A)                    % Vector đường chéo của A
```

```
ans =

     1
```

```
     6
     11
>> diag(A,-1)                    % Vector đường chéo dưới, vị trí số 1của A
ans =
     5
     10
>> B=triu(A)                     % Phần trên của ma trận được lưu vào B
                                 % Các phần tử còn lại được cho = 0
B =
     1     2     3     4
     0     6     7     8
     0     0    11    12
>> B = triu(A,-1)                % Phần trên của ma trận tính từ đường chéo -1 được lưu vào B
                                 % Các phần tử còn lại được cho = 0
B =
     1     2     3     4
     5     6     7     8
     0    10    11    12
>>B = tril(A)                     % Phần dưới của ma trận được lưu vào B
B =                              % Các phần tử còn lại được cho = 0
     1     0     0     0
     5     6     0     0
     9    10    11     0
>> B = tril(A,-1)                % Phần dưới của ma trận tính từ đường chéo -1 được lưu vào
                                 % B. Các phần tử còn lại được cho = 0
B =
     0     0     0     0
     5     0     0     0
     9    10     0     0
```

# CHƯƠNG 3

# *LẬP TRÌNH TRONG MATLAB*

## 3.1 CÁC PHẦN TỬ CƠ BẢN CHO CHƯƠNG TRÌNH

### 3.1.1 Giới hạn của các giá trị tính toán trong MATLAB

Đối với phần lớn các máy tính, khoảng giá trị cho phép từ $10^{-308}$ đến $10^{308}$. Giả sử có những lệnh sau:

```
>> x = 2.5e200;
>> y = 1.0e200
>> z = x*y;
```

Tuy giá trị của x và y nằm trong khoảng cho phép. Nhưng giá trị của z là 2.5e400 lại nằm ngoài khoảng giá trị cho phép. Lỗi này được gọi là tràn số mũ trên *(exponent overflow).* Giá trị của kết quả quá lớn đối với vùng nhớ của máy tính. Trong Matlab, kết quả này được biểu diễn là ∝.

Tràn số mũ dưới *(exponent underflow).* Giả sử có những lệnh sau:

```
>> x = 2.5e-200;
>> y = 1.0e200
>> z = x/y;
```

Giá trị của z sẽ là 2.5e-400.

Trong Matlab, kết quả này được biểu diễn là 0. Chia cho 0 là một toán tử không hợp lệ. Nếu một giá trị có hạn được chia cho 0, kết quả nhận được sẽ là ∝.

Matlab sẽ in ra một lời cảnh báo và sử dụng giá trị ∝ để tiếp tục tính toán các phép tính sau đó.

### 3.1.2 Các ký tự đặc biệt

**[ ]** Dạng ma trận. Dùng để quy ước cho việc biểu diễn hay vào số liệu cho các biến vector hay ma trận. Các phần tử trong biến đó được cách nhau bởi dấu space hay dấu ‘,’ nếu trên cùng hàng hoặc cột. Các cột hay hàng sẽ phân cách nhau bởi dấu ‘;’ hay Enter.

ví dụ:

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

ans =

1  2  3

4  5  6

7  8  9
```

**( )** Dạng chỉ số. Dành cho các biến của hàm hay các chỉ số các phần tử trong ma trận khi cần được nhập hay biểu diễn.

**.** Phân tách giữa các chỉ số và các phần tử của ma trận

**;** Phân tách các ma trận, các lệnh, các hàng của ma trận

**>>** Dấu nhắc cho lệnh sau

**...** Thể hiện sự tiếp tục của lệnh ở dòng sau

**%** Phần chú giải dòng lệnh được ghi sau dấu này dùng để hiểu rõ nghĩa 1 dòng lệnh chứ không tham gia vào chương trình

**:** Cách ghi tổng quát ma trận

**\n** Dấu hiệu tạo dòng mới

### 3.1.3 Các giá trị đặc biệt

**pi** Giá trị của π tự động được đưa vào biến này ( 3.14156 ...)

**i, j** Các biến này có giá trị ảo √-1

**Inf** Biến này đại diện cho giá trị ∝ của MATLAB, thể hiện kết quả chia cho 0. Một lời cảnh báo sẽ hiện ra, nếu bạn muốn hiển thị kết quả

chia cho 0, giá trị hiển thị là ∝.

**NaN** Giá trị vô định, biểu thức không xác định: 0 chia 0.

**clock** Hàm cho biết giá trị của thời gian hiện tại bao gồm năm, tháng, ngày, giờ, phút, giây.

**date** Hàm cho biết giá trị hiện tại của ngày được cho bởi 1 xâu ký tự.

Ví dụ:

```
>> date
```

ans =

10-Jun-97

**eps** Hàm xác định độ chính xác của số thực trong quá trình tính toán

**ans** Biến này được dùng để chứa giá trị tính toán của biểu thức nhưng không ghi vào tên biến.

### 3.1.4 Biến string

Biến string trong Matlab được biểu diễn sử dụng như các biến số thông thường khác của Matlab. Điều đó có nghĩa biến được nhập, thao tác và lưu trữ trong các vector với mỗi phần tử của vector là 1 ký tự. Các ký tự được lưu trữ trong vector dưới dạng mã ASCII của chính nó, tuy nhiên khi hiển thị trên màn hình dòng ký tự sẽ được xuất hiện chứ không phải mã của chúng.

Việc xác định vị trí của mỗi phần tử của biến string thông qua chỉ số của nó trong vector. Ma trận của các ký tự hay string cũng có thể được sử dụng nhưng mỗi phần tử trong đó phải bằng nhau.

Ví dụ:

```
>> name = ' Trường Đại học Bách khoa Hà nội '
```

ans =

Trường Đại học bách khoa Hà nội

Matlab cho phép thao tác trên các ký tự theo ví dụ dưới đây.

#### *a) Đảo ngược chuỗi ký tự.*

```
Function d = dao_tu ( name )
for i = length (name) :-1 : 1
```

```
newname ( i ) = name( length(name) + 1 - i );
end
d = newname;
end
```

***b) Dùng 1 phần của chuỗi string.***

```
>> disp ( ' Trường tôi là : ', name ( 1:24 ));
```

```
ans =
Trường tôi là : Trường Đại học bách khoa
```

***c) Kết hợp các string khác nhau tạo ra 1 string mới.***

```
>> text1 = ' Tôi '; text2 = ' yêu ';
>> text = [ text1''text2''name ]
>> text
```

```
ans =
    Tôi yêu Trường Đại học bách khoa
```

## Các lệnh với biến string

| | |
|---|---|
| **abs (str )** | Trả lại giá trị là 1 vector với các phần tử của vector là các mã ASCII của các ký tự trong chuỗi str. |
| **setstr ( x )** | Chuyển vector x với các phần tử là các số nguyên trong khoảng 0 -> 255 thành chuỗi str theo mã ASCII. |
| **num2str ( f )** | Chuyển đổi đại lượng vô hướng f thành chuỗi string cho việc biểu diễn các số có dấu phẩy động. Lệnh này thường đi cùng với disp, x label hay các lệnh truy xuất đầu ra khác. Giá trị mặc định là 4 chữ số. |
| **num2str ( f,k )** | Chuyển đổi đại lượng vô hướng f thành chuỗi string cho việc biểu diễn các số có dấu phẩy động với k chữ số. |
| **int2str ( n )** | Chuyển đổi số nguyên n thành chuỗi string cho việc biểu diễn số nguyên đó. |
| **rats (x, strlen)** | Chuyển đổi số có dấu phẩy động x thành chuỗi string phân thức xấp xỉ cho việc biểu diễn sô. strlen là biến mô tả chiều dài của chuỗi với giá trị mặc định là 13 chữ số. |

| | |
|---|---|
| **hex2num (hstr )** | Chuyển đổi số theo hệ hexa thành chuỗi string biểu diễn các số theo hệ dec bao gồm cả dấu phẩy động. |
| **hex2dec (hstr)** | Chuyển đổi số theo hệ hexa thành chuỗi string biểu diễn các số nguyên theo hệ dec. |
| **dec2hex ( n )** | Chuyển đổi số theo hệ dec thành chuỗi string biểu diễn các số hệ hexadecimal. |

## 3.2. CÁC HÀM TOÁN HỌC

Matlab cũng sử dụng các hàm logarit, các hàm lượng giác, các hàm mũ, các hàm đại số ... để tính toán.

Các hàm này đúng đối với các tham số là các đại lượng vô hướng và cả ma trận. Nếu hàm được dùng đối với các tham số là ma trận thì hàm sẽ cho kết quả là một ma trận có cùng kích thước và mỗi phần tử của ma trận này có giá trị tương ứng với các phần tử của ma trận đã cho.

Tham biến và tham trị của hàm được đặt trong dấu ngoặc đơn đi cùng với tên hàm. Hàm có thể không có hoặc có nhiều tham số phụ thuộc vào định nghĩa của nó. Nếu hàm có nhiều tham số thì giá trị của các tham số sẽ được truyền đến theo đúng thứ tự của nó. Một số hàm đòi hỏi truyền tham số theo những đơn vị quy định.

Ví dụ như các hàm lượng giác thì đơn vị của các tham số phải là radian. Trong Matlab, một số hàm sử dụng tham số để truyền giá trị đầu ra. Ví dụ đối với hàm *zeros* có thể sử dụng một hoặc hai tham số, tham số thứ hai để chứa giá trị đầu ra.

Các hàm này không được đặt ở bên phải dấu bằng và biểu thức vì nó là giá trị chứ không phải là biến. Một hàm có thể là tham số của một hàm khác. Khi một hàm được sử dụng làm tham số nó, phải được đặt đúng vị trí. Theo mặc định tên hàm được viết bằng chữ thường trừ khi bạn sử dụng lệnh *case off.*

### *3.2.1 Hàm toán học cơ bản:*

| | |
|---|---|
| **abs(x)** | Hàm tính giá trị tuyệt đối của x |
| **sqrt(x)** | Hàm tính căn bậc hai của x |
| **round(x)** | Làm tròn x về số nguyên gần nhất |
| **fix(x)** | Làm tròn số x về 0 |

| | |
|---|---|
| **floor(x)** | Làm tròn về phía -∝ |
| **ceil(x)** | Làm tròn về phía ∝ |
| **sign(x)** | Hàm cho giá trị là -1 nếu x nhỏ hơn 0, giá trị bằng 0 nếu x bằng 0, có giá trị là 1 nếu x lớn hơn 0 |
| **rem(x,y)** | Hàm trả lại số dư của phép chia x cho y |
| **exp(x)** | Hàm tính giá trị của $e^x$ |
| **log(x)** | Hàm tính giá trị ln(x) |
| **log10(x)** | Hàm tính giá trị $\log_{10}(x)$ |

### *3.2.2 Hàm lượng giác cơ bản:*

Quy đổi radian ra độ và ngược lại được tính toán theo các lệnh sau:

```
>> angle_degrees = angle_radians*(180/pi);
>> angle_radians = angle_degrees*( pi/180);
```

| | |
|---|---|
| **sin(x)** | Tính sine của góc x, khi x có đơn vị đo là radian |
| **cos(x)** | Tính cos của góc x, khi x có đơn vị đo là radian |
| **tan(x)** | Tính cos của góc x, khi x có đơn vị đo là radian |
| **asin(x)** | Tính arcsine của x, khi x nằm trong khoảng [-1,1], hàm trả lại góc có giá trị radian trong khoảng -π/2 đến π/2 |
| **acos(x)** | Tính arccosine của x, khi x nằm trong khoảng [-1,1], hàm trả lại góc có giá trị radian trong khoảng 0 đến π |
| **atan(x)** | Tính arctangent của x trong khoảng -π/2 đến π/2 |
| **atan2(x,y)** | Tính arctangent của y/x trong khoảng -π đến π, tuỳ thuộc vào dấu của x và y |

Ví dụ:

```
>> x = -2*pi: 2: 2*pi          % Tạo lập vector x với các giá trị từ -2pi - 2pi
x =
   -6.2832   -4.2832   -2.2832   -0.2832    1.7168    3.7168    5.7168
>> sin(x)
```

```
ans =
    0.0000    0.9093   -0.7568   -0.2794    0.9894   -0.5440   -0.5366
>> atan(x)
ans =
   -1.4130   -1.3414   -1.1580   -0.2760    1.0434    1.3080    1.3976
```

### *3.2.3 Các hàm hyperbolic:*

| | |
|---|---|
| **sinh(x)** | Hàm tính hyperbolic sine của x |
| **cosh(x)** | Hàm tính hyperbolic cosine của x |
| **asinh(x)** | Hàm tính nghịch đảo của hyperbolic sine của x |
| **acosh(x)** | Hàm tính nghịch đảo của hyperbolic cosine của x |
| **atanh(x)** | Hàm tính nghịch đảo của hyperbolic tangent |

Ví dụ:

```
>> sinh(x)
ans =
 -267.7449  -36.2286   -4.8530   -0.2870    2.6936   20.5544  151.9660
>>  atanh(x)
ans =
  Columns 1 through 4
  -0.1605 + 1.5708i  -0.2379 + 1.5708i  -0.4697 + 1.5708i  -0.2911
  Columns 5 through 7
   0.6662 + 1.5708i   0.2758 + 1.5708i   0.1767 + 1.5708i
```

## 3.3 CÁC DẠNG FILE ĐƯỢC SỬ DỤNG TRONG MATLAB

### 3.3.1 Script file (M-files )

Các chương trình, thủ tục bao gồm các dòng lệnh theo một thứ tự nào đó do người sử dụng viết ra được lưu trữ trong các files có phần mở rộng là **.m.* File dạng này còn được gọi là script file. File được lưu dưới dạng ký tự ASCII và có thể sử dụng các chương trình soạn thảo nói chung để tạo nó.

Bạn có thể chạy file này giống như các lệnh, thủ tục của MATLAB. Tức là có thể gõ tên file không cần có phần mở rộng, sau đó enter. Khi sử dụng, nội dung của M-file không được hiển thị lên màn hình.

Về cấu trúc ngôn ngữ, toán tử hay các bộ lệnh của *.m file, chúng tôi xin giới thiệu kỹ hơn ở phần sau. Và dưới đây là một số lệnh hệ thống tương tác với *.m files thường gặp.

**echo** Lệnh cho phép xem các lệnh có trong **.m* files khi chúng được thực hiện

**type** Lệnh cho xem nội dung file, ngầm định file ở dạng M-file.

**what** Lệnh này cho biết tất cả các files M-file và MAT-file có trongvùng làm việc hiện hành hay không.

Ví dụ sau đây là 1 ví dụ đơn giản nhất đưa ra dòng lệnh HELLO ra màn hình cùng với 1 số yêu cầu. File tạo thành được lưu trữ dưới tên HELLO.m

```
% chương trình hello.m , Ví dụ về phần lập trình trong Matlab.
% Xin chào bạn ! Hãy làm quen với tôi
disp ( ' Xin chào ! Bạn là ai ? ');
name = input ( ' Tên bạn là gì ' );
d = date ;
answer = [ ' Hello ' name ' ! Hôm nay là ngày ' d ]
disp ( answer );
disp ( ' Chúc bạn 1 ngày tốt lành ' ) ;
```

Sau các ký tự % là chỉ dẫn cho hoạt động của file.m. Nó không tham gia vào hoạt động của chương trình và cũng không hiển thị lên màn hình trừ khi ta dùng lệnh help + tên file.

```
>> help hello
```

Chương trình hello.m , Ví dụ về phần lập trình trong Matlab.

Xin chào bạn ! Hãy làm quen với tôi

### 3.3.2 Hàm và tạo hàm trong Matlab

Các hàm do người sử dụng viết cũng được lưu trong M-file. Chúng được sử dụng giống như các hàm của Matlab. Các file hàm phải được viết theo một quy định chặt chẽ.

**** Các quy tắc viết hàm M-files***

***Function:***

1. Hàm phải được bắt đầu bằng từ *function*, sau đó lần lượt là tham số đầu ra, dấu bằng, tên hàm. Tham số đầu vào được viết theo tham số đầu vào và được bao trong ngoặc đơn. Dòng này định nghĩa tham số đầu vào và tham số đầu ra; phân biệt sự khác nhau giữa file hàm và các file script.

2. Một số dòng đầu tiên nên viết chú thích cho hàm. Khi sử dụng lệnh *help* với tên hàm, chú thích của hàm sẽ được hiển thị.

3. Các thông tin trả lại của hàm được lưu vào tham số (ma trận) đầu ra. Vì vậy luôn kiểm tra chắc chắn rằng trong hàm có chứa câu lệnh ấn định giá trị của tham số đầu ra.

4. Các biến (ma trận) cùng tên có thể được sử dụng bởi cả hàm và chương trình chỉ đến nó. Không có sự lộn xộn nào xảy ra vì các hàm và các chương trình đều được thực hiện một cách tách biệt. Các giá trị tính toán trong hàm, tham số đầu ra không chịu tác động của chương trình.

5. Nếu một hàm cho nhiều hơn một giá trị đầu ra phải viết tất cả các giá trị trả lại của hàm thành một vec tơ trong dòng khai báo hàm.

Ví dụ:

```
function  [ dist, vel, accel ] = motion(x)
                    % Cả ba giá trị phải được tính toán trong hàm
```

6. Một hàm có nhiều tham số đầu vào cần phải liệt kê chúng khi khai báo hàm.

Ví dụ:

```
function   error = mse(w, d)
```

7. Các biến đặc biệt *nargin* và *nargout* xác định số tham số đầu vào, số tham số đầu ra được sử dụng trong hàm. Các tham số này chỉ là biến cục bộ.

Ví dụ một hàm M-file sẽ được viết như sau:

```
function  c = chuvi(r)
    % Tính chu vi của đường tròn có bán kính r
    % Nếu hàm được áp dụng cho ma trận thì giá trị trả lại sẽ là
```

```
    % một ma trận tương ứng với mỗi phần tử có giá trị là
    % chu vi của đường tròn có bán kính tương ứng với mỗi
    % phần tử của véc tơ nguồn.
c = pi*2*r;
```

### 3.3.3 Files dữ liệu

Các ma trận biểu diễn thông tin được lưu trữ trong các files dữ liệu. Matlab phân biệt hai loại file dữ liệu khác nhau Mat-files và ASCII files.

Mat-files lưu các dữ liệu ở dạng số nhị phân, còn các ASCII file lưu các dữ liệu dưới dạng các kí tự ASCII. Mat-file thích hợp cho dữ liệu được tạo ra hoặc được sử dụng bởi chương trình Matlab. ASCII file được sử dụng khi các dữ liệu được chia sẻ (export - import) với các chương trình khác các chương trình của Matlab.

Khi muốn lưu các dữ liệu ta dùng lệnh save như sau:

```
>> save <tên file> x,y;
```

Lệnh này sẽ lưu các ma trận x,y vào file có tên là <tên file>, ngầm định các files này có phần mở rộng là *.mat. Để gọi các ma trận này, ta dùng lệnh:

```
>> load <tên file>;
```

ASCII files có thể được tạo bởi các chương trình soạn thảo nói chung hay các chương trình soạn thảo bằng ngôn ngữ máy. Nó cũng có thể được tạo ra bởi chương trình Matlab bằng cách sử dụng câu lệnh sau đây:

```
>> save <tên file>.dat <tên ma trận>/ascii;
```

Lúc này mỗi một hàng của ma trận được lưu ở một dòng của file dữ liệu. Phần mở rộng *.mat không được tự động thêm vào file ASCII. Tuy nhiên, phần mở rộng *.dat mà ta thêm vào sẽ dễ dàng phân biệt 2 loại Mat-files và ASCII files.

Để gọi ma trận loại này ta dùng lệnh sau:

```
>> load <tên file>.dat;
```

Lệnh này sẽ tự động đặt tên cho ma trận trùng với tên file.

Ví dụ:

```
>> x = 0; pi /60 ; 2*pi;
>> y = sin ( x );
>> t = [ x y ]
```

Ghi dữ liệu của t vào file có tên như sau : dl1.mat

```
>> save   dl1.mat    t
```

Việc lấy dữ liệu ra đạt được qua biến t thông qua lệnh load. Các tham số cần đến dữ liệu sẽ lấy qua biến t.

```
>> load    dl1
>> x = t ( : , 1 );
>> y = t ( : , 2 );
>> plot ( x, y ); grid on;
```

## 3.4. CÁC BIỂU THỨC QUAN HỆ VÀ LOGIC

### 3.4.1 Các phép toán quan hệ

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

Phép so sánh hai ma trận là phép so sánh từng phần tử của hai ma trận có cùng kích thước, kết quả sinh ra một ma trận cùng cỡ có các phần tử nhận giá trị 1 nếu phép so sánh là đúng, ngược lại phần tử nhận giá trị 0. Kết quả của phép toán quan hệ được gọi là bảng sự thật (ma trận 0-1).

### 3.4.2 Các phép toán logic:

| Toán tử logic | Ký hiệu |
|---|---|
| and | & |
| or | \| |
| not | ~ |

Biểu thức logic cho phép so sánh các bảng sự thật giống như các toán tử quan hệ. Biểu thức logic luôn là hợp lệ nếu 2 bảng sự thật có kích thước bằng nhau. Trong

biểu thức logic toán tử *not* có thể được đặt ở phía trước. Một biểu thức logic có thể chứa nhiều toán tử.

*Ví dụ:* `~ ( b == c | b == 5.5 );`

Thứ tự các toán tử trong biểu thức logic từ cao đến thấp là *not, and, or*. Tuy nhiên, cũng có thể dùng ngoặc đơn để thay đổi thứ tự này.

*Bảng các phép logic*

| A | B | ~A | A\|B | A&B |
|---|---|---|---|---|
| false | false | true | false | false |
| false | true | true | true | false |
| true | false | false | true | false |
| true | true | false | true | false |

Trong Matlab tất cả các giá trị khác 0 đều được coi là đúng *(true),* còn giá trị bằng 0 được coi là sai *(false).* Chính vì vậy, phải hết sức thận trọng khi điều khiển chương trình bằng các biểu thức quan hệ và logic.

### 3.4.3 Các hàm quan hệ và logic

**any(x)**

Hàm cho giá trị là 1 nếu một phần tử của x khác 0, ngược lại cho giá trị 0

**all(x)**

Hàm cho giá trị là 1 nếu tất cả các phần tử của ma trận x khác 0, ngược lại cho giá trị là 0.

**find(x)**

Hàm trả lại vector chứa chỉ số của các phần tử khác 0 của x. Nếu x là một ma trận thì chỉ số được chọn từ x(:) và là một vertor cột tạo nên bởi các cột của x.

**exist('A')**

Hàm trả lại giá trị 1 nếu A là biến, là 2 nếu A hoặc A.m là file, là 0 nếu A không tồn tại trong vùng làm việc. Tên biến phải được đặt trong dấu nháy đơn.

**isnan(x)**

Giá trị trả về là ma trận *ones* nếu các phần tử của ma trận x là *NaN*, ngược lại trả về ma trận *zeros*.

**finite(x)**

Giá trị trả về là ma trận *ones* nếu các phần tử của ma trận x là giá trị hữu hạn, trả về ma trận *zeros* khi chúng là vô hạn hoặc *NaN.*

**isempty(x)**

Giá trị trả về 1 nếu ma trận x là rỗng, và 0 nếu ngược

lại.

| | |
|---|---|
| isstr(x) | Giá trị trả về là 1 nếu x là một xâu, 0 nếu ngược lại. |
| strcmp(y1,y2) | So sánh hai xâu y1,y2. Giá trị trả về là 1 nếu 2 xâu giống hệt nhau và bằng 0 nếu ngược lại. So sánh ở đây bao gồm: phân biệt chữ hoa và chữ thường, các ký tự đầu dòng và các dấu cách có trong xâu. |

## 3.5. CẤU TRÚC CÂU LỆNH ĐIỀU KIỆN

### 3.5.1 Lệnh if đơn

Cú pháp: if <biểu thức logic>

nhóm lệnh;

end

Nếu biểu thức logic là đúng, nhóm lệnh sẽ được thực hiện. Nếu biểu thức logic là sai thì chương trình sẽ nhảy tới lệnh *end*.

Ví dụ:

```
if    a < 50
count = count +1;
sum = sum + a;
end
```

Trong trường hợp *a* là đại lượng vô hướng, nếu *a < 50*, thì *count* tăng thêm 1 và a được cộng vào *sum*; trái lại câu lệnh thứ 2 không được thực hiện. Trong trường hợp *a* là một ma trận thì *count* tăng thêm 1 và nó chỉ được cộng vào *sum* khi mọi phần tử của nó nhỏ hơn 50.

### 3.5.2 Lệnh *IF* lồng nhau:

Cú pháp: if <biểu thức logic 1>

nhóm lệnh A;

if <biểu thức logic 2>

nhóm lệnh B;

end

nhóm lệnh C;

end

nhóm lệnh D;

Nếu biểu thức điều kiện 1 đúng chương trình sẽ thực hiện các nhóm lệnh A và C; nếu biểu thức điều kiện 2 đúng nhóm lệnh B sẽ được thực hiện trước nhóm lệnh C. Nếu biểu thức điều kiện 1 sai chương trình thực hiện ngay nhóm lệnh D.

### 3.5.3 Mệnh đề *ELSE*:

Cú pháp: if <biểu thức logic 1>

nhóm lệnh A;

else

nhóm lệnh B;

end

Cho phép thực hiện nhóm lệnh A nếu biểu thức logic là đúng, ngược lại thực hiện nhóm lệnh B.

### 3.5.4 Mệnh đề *ELSEIF*:

Khi ta có một cấu trúc lồng nhiều câu lệnh *if-else*, rất khó xác định nhóm lệnh nào sẽ được thực hiện khi biểu thức logic đúng (hoặc sai). Trong trường hợp này, người ta sử dụng mệnh đề *elseif* làm chương trình trở nên trong sáng và dễ hiểu hơn.

Cú pháp: if <biểu thức logic 1>

nhóm lệnh A;

elseif <biểu thức logic 2>

nhóm lệnh B;

elseif <biểu thức logic 3>

nhóm lệnh C;

end

Các mệnh đề *elseif* có thể dùng nhiều hơn nữa. Nếu biểu thức 1 đúng thì thực hiện câu lệnh A, nếu biểu thức 1 sai và biểu thức 2 đúng thì chỉ có nhóm lệnh B được thực hiện. Nếu biểu thức 1, 2 sai và 3 đúng thì chỉ có nhóm C được thực hiện. Nếu có từ hai biểu thức logic trở lên đúng thì biểu thức logic đúng đầu tiên xác định nhóm lệnh sẽ được thực hiện. Nếu không có biểu thức điều kiện nào đúng thì không có lệnh nào trong cấu trúc *if - elseif* được thi hành.

Có thể kết hợp 2 mệnh đề *else* và *elseif*:

Cú pháp: if <biểu thức logic 1>
nhóm lệnh A;
elseif <biểu thức logic 2>
nhóm lệnh B;
elseif <biểu thức logic 3>
nhóm lệnh C;
else
nhóm lệnh D;
end

Nếu cả ba biểu thức logic đều sai thì nhóm lệnh D được thi hành. Đôi lúc, cấu trúc if-*elseif* còn được gọi là cấu trúc *case* bởi vì có một số trường hợp được kiểm tra. Mỗi trường hợp được kiểm tra bởi một biểu thức logic tương ứng.

Ví dụ sau đây minh hoạ các cấu trúc mệnh đề câu điều kiện. Chương trình được ghi trong file hello2.m

```
% Chương trình hello2 mô tả cấu trúc câu điều kiện trong Matlab
% Bài toán so sánh tuổi của bạn với số ngẫu nhiên sinh ra bởi hàm rand
disp ( ' Xin chào ! Rất hân hạnh được làm quen ');
x = fix ( 30* rand );
disp ( ' Tuổi của tôi trong khoảng từ 0-30 ');
gu = input ( ' Đưa vào tuổi của bạn : ');
if gu < x
        disp ( ' Bạn trẻ hơn tôi ');
elseif gu > x
        disp ( ' Bạn lớn hơn tôi ');
else
        disp ( ' Tuổi bạn bằng tuổi tôi  ');
end
```

### 3.5.5. Cú pháp câu điều kiện và break

Cú pháp: if <biểu thức logic > , break , end

Từ khoá break với câu lệnh if cho phép thoát ra khỏi vòng lặp nếu <biểu thức logic> trong câu điều kiện là đúng, ngược lại sẽ thực hiện nhóm lệnh tiếp theo trong vòng lặp đó.

***Ví dụ:***

Về nhập một số dương, nếu số đó < 0 thoát khỏi chương trình. Nếu số đó chia hết cho 2 hiện kết quả. Nếu số đó không chia hết cho 2 nhập số mới.

```
while 1
n = input ( ' Cho vào 1 số dương , thoát khi n < 0);
if n<=0 , break , end
while  n > 1
if rem( n , 2 ) == 0
  disp( ' Số dương cho vào chia hết cho 2 ', n );
break;
else
disp( ' Số dương cho vào không chia hết cho 2 ! Xin nhập số khác ');
end
end
```

## 3.6. CẤU TRÚC VÒNG LẶP

### 3.6.1 Vòng lặp FOR:

Cú pháp: for chỉ số = biểu thức
nhóm lệnh A;
end

Biểu thức là một ma trận (cũng có thể là một vectơ hay một đại lượng vô hướng), nhóm lệnh A được thi hành lặp đi lặp lại số lần bằng số cột của ma trận biểu thức. Mỗi lần lặp, chỉ số sẽ nhận giá trị của một phần tử của ma trận.

Chú ý: Nếu trường hợp ta không biết kích thước của vectơ, ta sử dụng hàm *length* để xác định số lần ta muốn lặp.

**** Qui tắc sử dụng vòng lặp FOR:***

+ Chỉ số của vòng lặp phải là biến.

+ Nếu ma trận biểu thức là ma trận rỗng thì vòng lặp *for* sẽ không thực hiện. Chương trình bỏ qua vòng lặp.

+ Nếu ma trận biểu thức là một đại lượng vô hướng. Vòng lặp được thực hiện một lần và chỉ số nhận giá trị của đại lượng vô hướng.

+ Nếu biểu thức ma trận là một vectơ hàng, sau mỗi lần lặp chỉ số lại lấy giá trị tiếp theo của vectơ.

+ Nếu biểu thức ma trận là ma trận, sau mỗi lần lặp chỉ số sẽ lấy giá trị của cột tiếp theo của ma trận.

+ Khi kết thúc vòng lặp, biến chỉ số nhận giá trị cuối cùng.

+ Nếu sử dụng toán tử (:) vào biểu thức ma trận:

```
For k = chỉ số đầu : gia số : chỉ số kết thúc;
```

Số lần thực hiện vòng lặp sẽ được tính theo công thức sau:

```
floor((kết thúc-bắt đầu) / gia số)) + 1;
```

Nếu giá trị là một số âm thì không thực hiện vòng lặp.

Nếu mốn thoát khỏi vòng lặp trước khi vòng lặp thực hiện xong để dò lỗi có trong vòng lặp, sử dụng lệnh *break* .

Ví dụ mô tả cho vòng lặp for được ghi trong file hello3.m

```
% Chương trình hello2 mô tả cấu trúc câu điều kiện trong Matlab
% Bài toán dự đoán 1 số ngẫu nhiên sinh ra từ hàm rand
% cho bởi các lần thử tạo bởi vòng lặp for
x = fix ( 100* rand );
n = 7;
test = 1;
for k  = 1:7
number = int2str( n);
disp ( ' Bạn có quyền dự đoán ' number ' lần ');
```

```
disp ( ' Số cần đoán nằm trong khoảng từ 0 – 100 ');
gu = input ( ' Đưa vào số bạn dự đoán ');
if gu < x
        disp ( ' Nhỏ hơn ');
elseif gu > x
        disp ( ' Lớn hơn ');
else
        disp ( ' Xin chúc mừng bạn đã đoán chính xác ');
        test = 0;
        break
end
n = n-1;
end
if  test > 0
disp ( ' Bạn không đoán ra rồi ');
numx = int2str( x );
disp ( ' Số đó là : ' numx);
end
```

### 3.6.2 Vòng lặp WHILE:

Là cấu trúc rất quan trọng.

Cú pháp: while < biểu thức>
nhóm lệnh A;
end

Nếu biểu thức đúng thì thực hiện nhóm lệnh A. Khi thực hiện xong thì lại kiểm tra điều kiện. Nếu điều kiện vẫn còn đúng thì nhóm lệnh A lại được thực hiện. Khi điều kiện sai, vòng lặp kết thúc. Trong nhóm lệnh A nên có các biến trong biểu thức, hoặc các giá trị của biểu thức không thay đổi. Nếu biểu thức luôn luôn đúng (hoặc có giá trị luôn khác không), vòng lặp sẽ bị quẩn. Để thoát khỏi vòng lặp quẩn, ta sử dụng Ctrl+C.

Ví dụ:

```
% Chương trình hello3 mô tả cấu trúc câu điều kiện while trong Matlab
% Bài toán cho ra từ hello trên màn hình với số lần nhập vào từ bàn phím
disp ( ' Xin chào ! Hello 3 ');
gu = input ( ' Nhập vào số lần in : ');
i=0;
while i~= gu
        disp ([ ' Hello '  i ]);
        i = i + 1;
end
```

# CHƯƠNG 4

# ĐỒ HOẠ 2 CHIỀU TRONG MATLAB

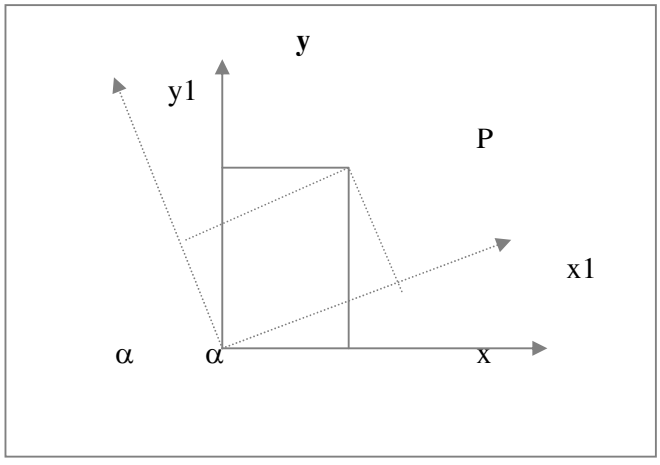
## 4.1. CÁC PHÉP BIẾN ĐỔI ĐỒ HOẠ

Nghịch đảo ma trận và định thức được giới thiệu trong phần này được hình dung như các phép biến đổi tạo nên các phép chuyển vị của các thực thể hình học hay các vector trong cửa sổ đồ hoạ của Matlab.

Nội dung của các phép biến đổi đó nhằm mô tả các phương pháp được sử dụng trong hầu hết các lĩnh vực kỹ thuật khác nhau như đồ hoạ máy tính hay robotic.

Ngoài ra để có thể khai thác tiềm năng về đồ hoạ của Matlab chúng tôi cũng xin liệt kê các hàm tương tác trên cửa sổ đồ hoạ của Matlab một cách chi tiết nhằm đem đến cho bạn một thư viện các hàm, hiệu ứng của các hàm cũng như phương pháp tiếp cận các hàm đó.

### 4.1.1. Quay hệ trục toạ độ trên mặt phẳng.




*Hình 4.1. Quay hệ trục trên mặt phẳng.*

Hình 4.1. hệ toạ độ của điểm P được biểu diễn bởi các đường liền nét x, y và điểm P trên hệ trục biểu diễn bởi các đường đứt nét $x_1$ và $y_1$. $x_1$ và $y_1$ quay 1 góc $\alpha$ ngược chiều kim đồng hồ so với 2 trục x, y. Quan hệ giữa hai cặp hệ trục toạ độ được biểu diễn bởi công thức sau:

$$x_1 = x \cdot \cos\alpha + y \cdot \sin\alpha \qquad (4.1)$$

$$y_1 = - x \sin\alpha + y \cos\alpha \qquad (4.2)$$

với $P = \begin{bmatrix} x \\ y \end{bmatrix}$ và $P_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$ ; $A = \begin{vmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{vmatrix}$

ta có thể viết 4.1 và 4.2 dưới dạn sau:

$$P_1 = AP \qquad (4.3)$$

Quan hệ nghịch đảo của hình 4.1 được thể hiện như sau:

$$x = x_1 \cdot \cos\alpha + y_1 \cdot \sin\alpha \qquad (4.4)$$

$$y = - x_1 \cdot \sin\alpha + y_1 \cdot \cos\alpha \qquad (4.5)$$

Điều đó có nghĩa ma trận biểu diễn phép quay 1 góc $-\alpha$ từ hệ toạ độ $x_1$, $y_1$ đến hệ toạ độ x, y là:

$$B = \begin{vmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{vmatrix}$$

Lúc đó 4.4 và 4.5 được viết thành

$$P = B \cdot B_1 \qquad (4.6)$$

### 4.1.2. Nghịch đảo ma trận.

Vậy quan hệ giữa A và B được hiểu ra sao?

Từ phương trình (4.5), (4.6) ta thấy rằng với 1 điểm P bất kỳ, sau hai phép chuyển ta đều thu được chính nó.

$$P = B\ A\ P$$

nếu loại bỏ biến P ta có thể viết như sau:

$$B.A = \begin{vmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{vmatrix} \begin{vmatrix} \sin\alpha & \cos\alpha \\ -\sin\alpha & \cos\alpha \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ 0 & 0 \end{vmatrix}$$

tương tự ta có:

$P_1 = ABP_1$

Một lần nữa ta thấy rằng **AB** chính bằng ma trận đơn vị và chúng ta có thể phát triển ma trận **B** là ma trận nghịch đảo của **A** và chúng ta thường biểu diễn $\mathbf{A^{-1}}$. Định nghĩa được phát triển như sau:

Cho một ma trận vuông **A**, ma trận nghịch đảo $\mathbf{A^{-1}}$ của **A** là ma trận sao cho khi nhân phải hay trái với **A** đều cho ta kết quả là ma trận đơn vị **I**.

$$\mathbf{A} \cdot \mathbf{A^{-1}} = \mathbf{A^{-1}} \cdot \mathbf{A} = \mathbf{I}$$

nghịch đảo ma trận 2 x 2 có thể dễ dàng tìm được theo cách sau. Ví dụ **a** là các phần tử của ma trận nghịch đảo từ **A**, việc nhân A . B cho kết quả như sau:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} \qquad \textbf{(4.8)}$$

Từ phương trình (4.8) cho kết quả sau:

$$b_{11} = \frac{a_{22}}{a_{11}a_{22} - a_{21}a_{22}} \qquad \textbf{(4.9)}$$

$$b_{12} = \frac{-a_{12}}{a_{11}a_{22} - a_{21}a_{12}} \qquad \textbf{(4.10)}$$

$$b_{21} = \frac{-a_{21}}{a_{11}a_{22} - a_{21}a_{12}} \qquad \textbf{(4.11)}$$

$$b_{22} = \frac{a_{11}}{a_{11}a_{22} - a_{21}a_{12}} \qquad \textbf{(4.12)}$$

Từ đây chúng ta dễ dàng thu đượ ma trận nghịch đảo B từ A. Tuy Matlab hàm nghịch đảo được viết sẵn trong thư việc và được gọi ra thông qua lệnh ***inv***. Với lệnh inv (A) cho ra ma trận nghịch đảo của A.

Ví dụ:

- Quay hệ trục toạ độ đi một góc $30^0$ sẽ được viết như sau:

```
>> alpha = 30
>> A=[cos(pi*alpha/180)      sin(pi*alpha/180)
-sin(pi*alpha/180)      cos(pi*alpha/180)]
A =     0.866       0.500
-0.500      0.866
```

- Ma trận nghịch đảo B tạo thành từ A

```
>> B = inv(A)
B = 0.866     -0.500
    0.500      0.866
```

- Nhân 2 ma trận A và, kết quả thu được như sau

```
>> A * B
ans = 1.000       0.000
```

```
        0.000        1.000
```

- Quay trục qua điểm x = 3 , y = 7

```
>>P1 = A*[3 ; 7]
P1 =
   6.0981
   4.5622
```

- Nghịch đảo lại ma trận hoàn trả lại toạ độ cho điểm

```
>> P = B * P1
   P =
        3
        7
```

### 4.1.3. Góc Euler.

Góc Euler là thao số quy ước để mô tả việc quay trong hệ không gian 3 chiều hay hệ toạ độ trực giác. Những tham số trên có rất nhiều ứng dụng trong lĩnh vực cơ khí. Có một vài cách định nghĩa khác nhau về góc Euler được biết đến như: Meitrovitch (1970), Guggenheimer (1977) và Czichos (1989).

Ở bài toán của chúng ta toạ độ của điểm P được xác định bởi hệ 3 giá trị x, y, z và chúng ta phải xác định ra điểm tương ứng $x_1$, $y_1$, $z_1$ sau khi quay hệ toạ độ đi 1 góc. Việc xác định chiều quay âm/ dương của hệ trục toạ độ thông qua quy tắc bàn tay phải.

Theo hình 4.2 với công thức được học trong phần đồ hoạ. Khi ta qua hệ toạ độ xung quanh trục z với một góc ψ. Điểm x*, y*, z* được tạo thành sẽ được mô tả theo công thức sau:

$$\begin{vmatrix} x^* \\ y^* \\ z^* \end{vmatrix} = X^* = \begin{vmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \end{vmatrix} = A.X \qquad (4.13)$$

Tiếp theo quay hệ trục quanh trục x với một góc θ. Hệ giá trị toạ độ mới của điểm thu được được viết dưới công thức:

$$\begin{vmatrix} x^{**} \\ y^{**} \\ z^{**} \end{vmatrix} = X^{**} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{vmatrix} \begin{vmatrix} x^* \\ y^* \\ z^* \end{vmatrix} = B.X^* \qquad (4.14)$$

Bước 3 quay hệ trục quanh trục z hướng ngược lại 1 góc ϕ. Giá trị toạ độ cuối cùng thu được sẽ là:

$$\begin{vmatrix} x_1 \\ y_1 \\ z_1 \end{vmatrix} = P_1 = \begin{vmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x** \\ y** \\ z** \end{vmatrix} = CX** \qquad (4.15)$$

Kết quả 3 tiến trình quay:

$$P_1 = A\ B\ C\ X = DX \qquad (4.16)$$

```
% Đoạn chương trình  ví dụ cho việc quay ma trận dưới các góc si, theta, fi
function  R = Elrotate (si, theta, fi)
A = [cos(si)     sin(si)        0
     -sin(si)    cos(si)        0
       0         0              1];
B = [1           0              0
       0         cos(theta)     sin(theta)
       0         -sin(theta)    cos(theta)];
C = [cos(fi)     sin(fi)        0
     -sin(fi)    cos(fi)        0
       0           0            1];
R = C * B * A;
```

Phép biến đổi Czichos (1989) được biểu diễn dưới công thức sau: Giả sử Cθ = cos (θ), Sθ = sin(θ)... ta có ma trận quay

$$R = \begin{vmatrix} C\phi C\psi - S\phi C\theta S\psi & C\phi C\psi + S\phi C\theta S\psi & S\phi S\theta \\ -S\phi C\psi - C\phi C\theta S\psi & -S\phi S\psi + C\phi C\theta C\psi & C\phi S\theta \\ S\theta S\psi & -S\theta C\psi & C\theta \end{vmatrix} \qquad 4.17)$$

Ví dụ:

Khi cho điểm P với các giá trị toạ độ [2. 5. 3] cho các góc quay là 30, 45, $20^0$. Việc biểu diễn bằng Matlab được viết như sau:

```
>> R = Elrotate( 30*pi/180 , 45*pi/180 , 20*pi/180 )
R =
   0.6929   0.6793   0.2418
  -0.6284   0.4044   0.6645
   0.3536  -0.6124   0.7071
>> X1 = R*[ 2 ; 5 ; 3 ]
X1 =
```

```
    5.5077
    2.7587
   -0.2334
>> X = inv( R ) * X1
X =
        2.0000
        5.0000
        3.0000
```

Trong hàm Elrotate tạo ra các biến trong A, B, C tuy nhiên khi kiểm tra bằng

```
>> A
```

hay dùng lệnh

```
>>Who
```

Các biến A, B, C cùng không xuất hiện vì A, B, C là các tham biến trong của hàm Elrotate và chỉ có tác dụng trong hàm.

## 4.2. PHÉP BIẾN ĐỔI AFFINE TRONG KHÔNG GIAN 2D

Các đối tượng hình hoạ được mô tả trong chương này có một ý nghĩa hết sức quan trọng trong các ứng dụng của các lĩnh vực kỹ thuật hiện đại như đồ họa mày tính hay robotics. Một nhóm các lệnh được sử dụng thường xuyên gọi là Affine transformation. Chúng bao gồm: translation, rotation, scaling... ở đây chúng ta sẽ cùng xem xét việc thực hiện chúng trong Matlab.

### 4.2.1 Tọa độ thuần nhất

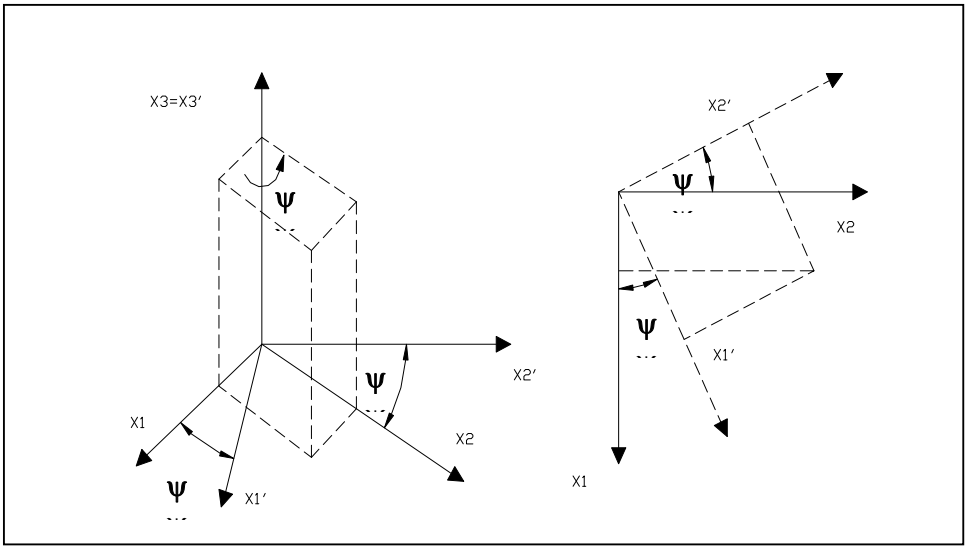
Trong thực tế để biểu diễn các phép biến đổi Affine người ta thường sử dụng phép biến đổi ma trận. Thường đồ hoạ máy tính và kỹ thuật robotics đòi hỏi concatenation của vài phép biến đổi. Điều đó được thực hiện bởi 1 loại các phép nhân ma trận. Việc nhân các ma trận chuyển đổi được thực hiện cùng với việc sử dụng hệ toạ độ thuần nhất.

Để giới thiệu hệ toạ độ thuần nhất chúng ta giả sử có 1 điểm P trong hệ toạ độ Đề các với 2 giá trị toạ độ x, y.

Việc biểu diễn P dưới dạng hệ toạ độ thuần nhất sẽ được viết như sau:

$$P = \begin{vmatrix} x_1 \\ y_1 \\ z_1 \end{vmatrix} \qquad \begin{aligned} x &= x_1 / W \\ y &= y_1 / W \end{aligned}$$

Nguyên nhân quan trọng trong việc biểu diễn theo hệ toạ độ thuần nhất là chúng cho phép biểu diễn các điểm ở xa vô cùng.




***Hình 4.2 Quay thực thể quanh trục x3***

Trong hệ toạ độ không gian thường (x, y) để biểu diễn điểm ở vô cùng ít nhất trong hai giá trị của điểm = α. Nhưng trong hệ toạ độ thuần nhất, việc biểu diễn chỉ thông qua bằng cách cho giá trị W = 0 với $x_1$, $y_1$ là những số hữu hạn bất kỳ.

Tuy nhiên chúng ta không đi sâu vào điểm này, bạn đọc có thể tham khảo các tài liệu về đồ hoạ máy tính và các bài toán chiếu phối cảnh, ở đây chúng ta chỉ nói về phương pháp tạo ra các phép biến đổi trong không gian đồ hoạ.

Với Hình 4.3 các điểm tạo nên hình vuông được cho các giá trị như sau:

$$P_1 = \begin{vmatrix} -0.5 \\ 0 \\ 1 \end{vmatrix}, \quad P_2 = \begin{vmatrix} -0.5 \\ 1 \\ 1 \end{vmatrix}, \quad P_3 = \begin{vmatrix} 0.5 \\ 1 \\ 1 \end{vmatrix}, \quad P_4 = \begin{vmatrix} 0.5 \\ 1 \\ 1 \end{vmatrix}$$

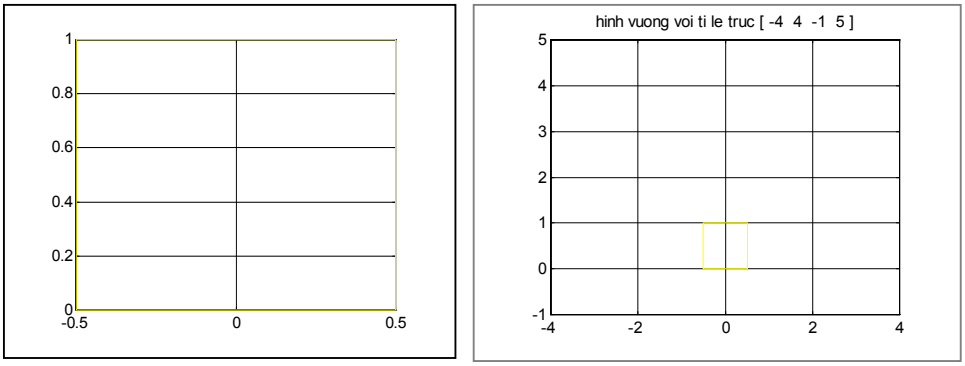
Như chúng ta đã biết giá trị toạ độ thứ ba - W được cho bằng 1, ở đây chúng ta đã sử dụng đến kỹ thuật phổ biến trong các bài toán đồ hoạ và với Matlab chúng ta dễ dàng thể hiện bằng các câu lệnh sau:

```
>> P1 = [-0.5;  0;  1]; P2 = [-0.5;  1;  1];
>> P3 = [0.5;  1;  1];  P4 = [0.5;  0;  1];
```

(Chú ý rằng ma trận P chứa 2 vector $P_1$ dùng cho việc đóng hình vuông)

Việc tạo ta hình vuông trên màn đồ hoạ thông qua biến square

```
>> square = [ P1      P2      P3      P4      P1] ;
>> plot ( square( 1,: ), square( 2,: ) )
>> axis([-4   4   -1   5]);
>> title ('hình vuông với tỉ lệ trục [ -4  4  -1  5 ]');
```




***Hình 4.3*** ***a. Hình vuông chuẩn*** ***b. Hình vuông sau khi thay đổi tỉ lệ trục toạ độ***

việc tạo ta hình vuông trên màn đồ hoạ thông qua biến square

```
>> square = [ P1     P2     P3     P4     P1 ] ;
>> plot ( square( 1,: ), square( 2,: ) )
>> axis([-4   4   -1   5]);
>> title ('hình vuông với tỉ lệ trục [ -4  4  -1  5 ]')
```

### 4.2.2 Phép chuyển dịch

Hàm biến đổi chuyển vị với các khoảng dx và dy song song trên 2 trục x và y.

```
function T = Translate ( dx , dy )
T = [1   0   dx;   0   1   dy;  0   0   1];
% ma trận dịch chuyển trong hệ toạ độ đồng nhất
```

$$\%\ T \quad = \quad \begin{vmatrix} 0 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{vmatrix}$$

Việc dịch chuyển hình vuông 1 khoảng đơn vị theo x và 2 khoảng đơn vị theo y được thể hiện bằng các dòng lệnh:

```
>> P = translate ( 1 , 2) * square
>> plot ( P( 1 , : ), P ( 2 , : ) );
>> axis ( [-3   3  -1   3] )
>> title (' Hình vuông thay đổi vị trí theo dx và dy ')
```


***Hình 4.4 Hình vuông dịch chuyển theo độ dài dx và dy***

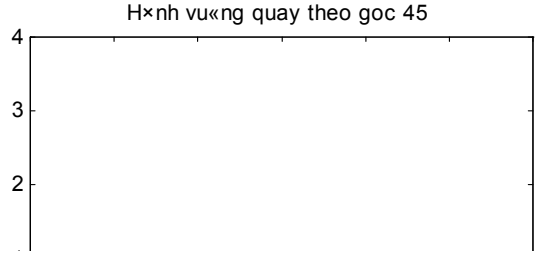
### 4.2.3. Phép quay

Hàm quay quanh gốc toạ độ với một góc ϕ bất kỳ ngược chiều kim đồng hồ được viết.

```
Function   R = rotate (fi)
```

$$\%\mathbf{R} = \begin{vmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

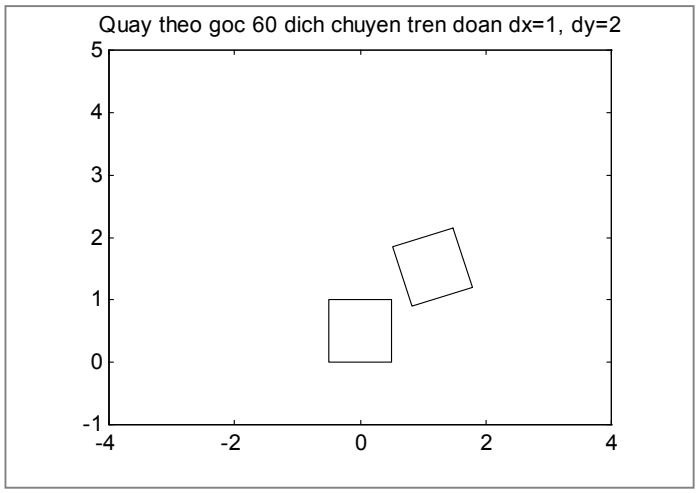
```
R =    [cos(fi)          -sin(fi)    0
        sin(fi)           cos(fi)     0
          0                 0         1]
```




***Hình 4.5 Hình vuông quay 1 góc 45 theo gốc toạ độ***

Hình 4.5 thu được thông qua dòng lệnh dưới Matlab như sau, với góc quay $\theta = 45^0$.

```
>> P = rotate (45*pi/180)*square
>> plot (P( 1 , : ), P ( 2 , : ) ), axis ([-3   3  -1   3])
>> title ('Hình vuông quay theo goc 45 ')
```




***Hình 4.6 Tổ hợp của hai phép biển đổi***

Hay hình 4.6 thu được 1 cách dễ dàng trên cơ sở kết hợp của 2 phương pháp chuyển đổi.

```
>> P = rotate (30*Pi/180)*translate(1,2)*square
>> plot (P( 1 , : ), P ( 2 , : ) ), axis ([-4   4  -1   5])
>> title ('Hình vuông quay và dịch chuyển ')
```

### 4.2.4 Phép tỉ lệ (Scaling)

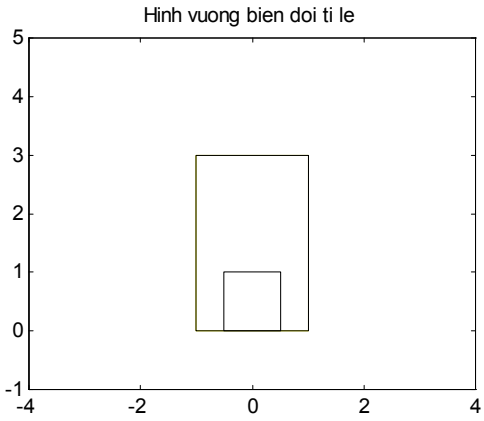
Hàm dưới đây cho phép biến đổi tỷ lệ của hình theo 1 tỷ lệ nhất định. Việc biến đổi tỷ lệ được thực hiện qua phép nhân ma trận S vơ Sx, Sy là 2 hệ số biến đổi.

$$\% \; S = \begin{vmatrix} S\,x & 0 & 0 \\ 0 & S\,y & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

```
function S = scale (Sx, Sy)
S = [ Sx  0  0;  0  Sy  0;  0  0  1 ]
```

Ví dụ việc biến đổi hình vuông 2 lần theo x và 3 lần theo y được thực hiện nhờ các lệnh sau.

```
>> P = scale (2,3)*square
>> plot (P( 1 , : ), P ( 2 , : ) )
>> title ('Hình vuông thay đổi tỉ lệ theo x = 2 theo y = 3')
```

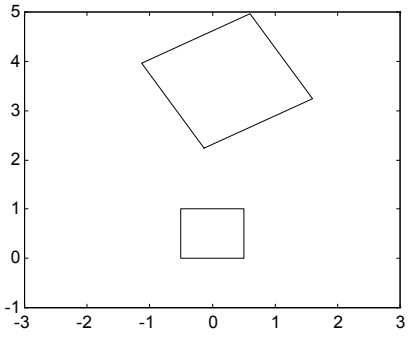



***Hình 4.7 Phép biến đổi tỉ lệ***

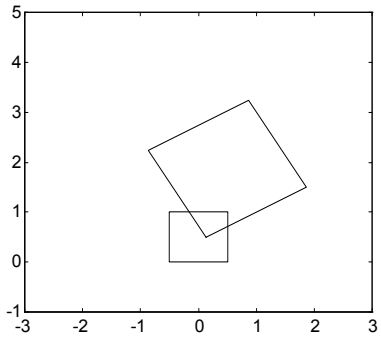
Hình 4.7 cho thấy sự biến đổi của hình vuông qua hàm scale. Việc thay đổi đó thực hiện qua gốc toạ độ.

Phép chuyển vị, phép quay hay scale đều có thể hết hợp lẫn với nhau 1 cách dễ dàng. Việc chuyển đổi vị trí của các phép biến đổi sẽ đem đến các hình ảnh khác nhau.

```
>> P = Scale(2,2)*roate(30*Pi/180)translate(1,1)*square
>> plot (P( 1 , : ), P ( 2 , : ) ), axis ([-4   4  -1   5])
>> title ('Hình vuông quay')
```




***Hình 4.8***




***Hình 4.9***

***ảnh 4.8 khác với ảnh 4.9 với phép đảo vị trí phép toán***

```
>> P = translate(1,1) *rotate(30*Pi/180)*Scale(2,2) *square
>> plot (P( 1 , : ), P ( 2 , : ) ), axis ([-4   4  -1   5])
>> title ('Hình vuông quay')
```

## 4.3. CÁC HÀM CHUẨN ĐỂ BIỂU DIỄN ĐỒ HOẠ 2 CHIỀU

Như đã giới thiệu qua với bạn đọc ở phần trên, Matlab là công cụ rất mạnh cho việc xử lý và thể hiện dưới dạng đồ hoạ. Hàm Plot trong Matlab thường được sử dụng liên tục cho việc vẽ và tạo lập các dạng đồ thị 2 chiều. Dạng đơn giản nhất để thể hiện dữ liệu là Plot nhưng kiểu đường hay mầu của đường được xác định trên biến str của hàm. Bảng dưới đây sẽ cho phép chúng ta nắm đầy đủ mọi thông tin về hàm.

### 4.3.1 Các bộ lệnh vẽ

#### a. Lệnh PLOT

| | |
|---|---|
| **plot ( x,y )** | Vẽ theo vector y và x (y trục tung của hệ, x trục hoành). Đồ thị thu được sẽ là tập của các điểm (xi,yi). |
| **plot ( y )** | Vẽ ra tập các điểm (i,yi) với y là các điểm trên trục tung và i là các điểm trên trục hoành. |
| **plot ( z )** | Vẽ theo tập các số ảo ( real( z ), image( z ) ). Trục hoành là tập các số thực và trục tung là tập các số ảo. |
| **plot ( A)** | Vẽ ra theo các cột của A theo chỉ số tương ứng xác định bởi tự chương trình. |
| **plot ( x,A )** | Vẽ ra theo các cột của A theo chỉ số tương ứng xác định bởi vector x. |
| **plot ( A,x )** | Vẽ x theo A. Nếu A là ma trận m hàng n cột, vector x có thể theo chỉ số tương ứng trên m hoặc n nếu chiều dài của x = m hay bằng n. Vector x có thể là ma trận hàng hay cột. |
| **plot ( A,B )** | Vẽ các cột của A theo các cột của B. Nếu A và B là 2 ma trận có cùng độ lớn mxn thì chúng ta sẽ thu được m đồ thị n điểm . |
| **plot ( ..., str )** | Vẽ hàm có các biến số xác định như trên và các chỉ số về mầu sắc và kiểu đường theo biến str |
| **plot( x1, y1, str1, x2, y2, str2, ... )** | Vẽ hàm vector y1, y2, ..., yn theo vector x1, x2, ..., xn ( y trục tung của hệ, x trục hoành ). Đồ thị thu được sẽ lấy giá trị về mầu sắc và kiểu đường tương ứng trong str1, str2, ..., str n |

Các giá trị của biến str trong hàm Plot về mầu sắc hay kiểu dáng của đường được liệt kê theo bảng dưới đây.

| Kiểu đường | Kiểu đường | Mầu sắc | Mầu sắc |
|---|---|---|---|

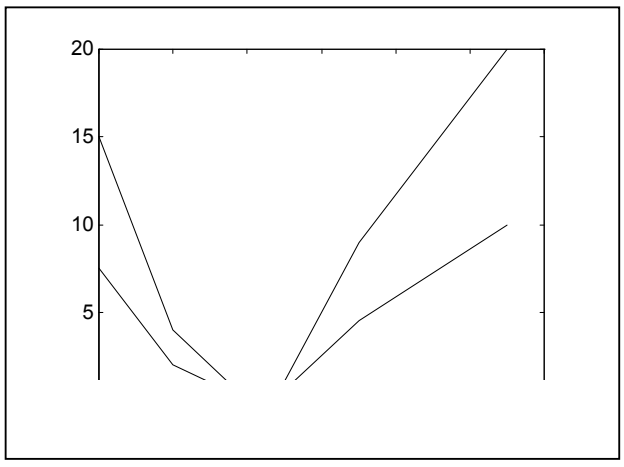
| | | | |
|---|---|---|---|
| . điểm | - đường liền nét | Y vàng | C xanh lá mạ |
| * sao | -- đường đứt nét | G xanh lá cây | W trắng |
| X chữ cái x | -. đường chấm gạch | M đỏ tươi | R đỏ |
| O chữ cái o | : đường chấm | B xanh lam | K đen |
| + dấu cộng | | | |

Kiểu đường thẳng có thể là tổ hợp của 2 hay nhiều yếu tố. Ví dụ 'y- -' là ký hiệu cho đường vàng liền nét hay ' b+' là đường xanh các dấu cộng. Độ rộng của đường hay kích thước các ký hiệu có thể thay đổi tuỳ ý. Các trục toạ độ sẽ tự động thay đổi phù hợp với đơn vị của đồ thị nếu không có sự tác động nào khác của người sử dụng.

Ví dụ:

Vào dữ liệu cho các biến số x,y

```
>> x = [ -4  -2  0  1  3  7 ]
>> y = [ 15  4  0  1  9  20 ]
>> plot( x,y,'w'); hold on
>> plot( x,y/2 );
```




***Hình 4.10 Đồ thị hàm y và y/2 theo x***

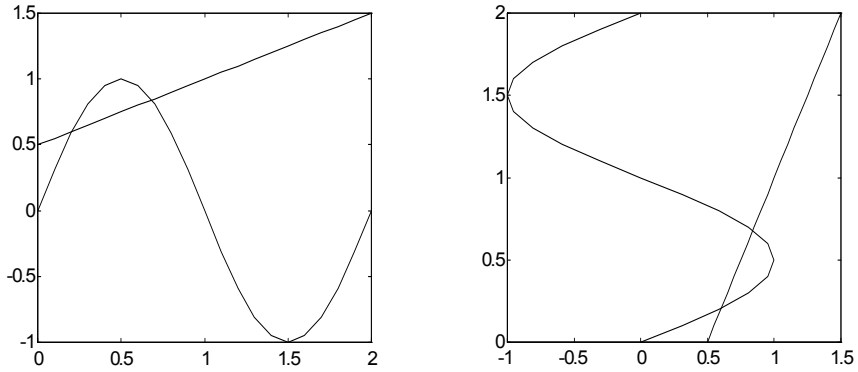
b. Đồ thị hàm sin(x) và x/2 + 1/2

```
>>x = 0 : 0  1 : 2;
```

```
>>A = [sin (pi*x); 0.5 + 0.5*x];
>>plot (x,A)
```

* Trục của đồ thị xoay khi ta giao hoán vị trí cua A và A

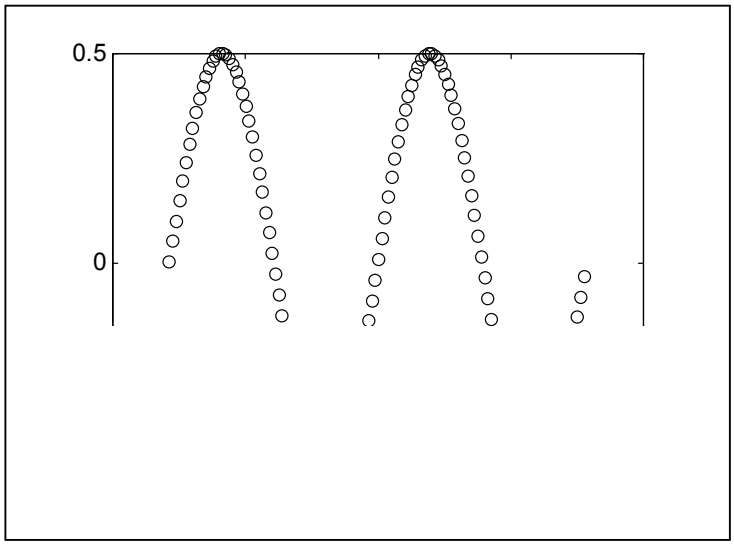
```
>>x = 0 : 0  1 : 2;
>>A = [sin (pi*x); 0.5 + 0.5*x];
>>plot (A,x)
```




***Hình 4.11 Đồ thị của ma trận A trên trục x và x trên A***

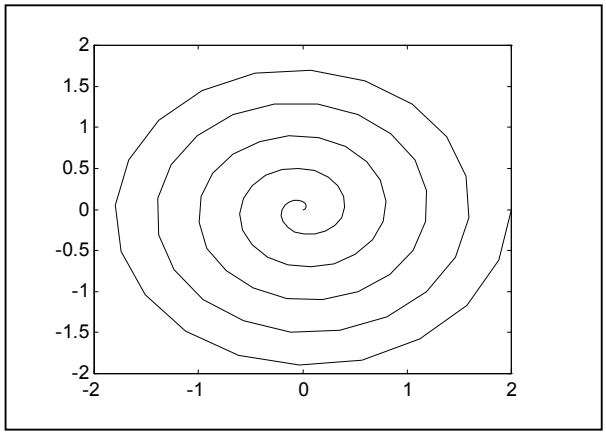
c) Hàm y = sin(x). cos(x) với các điểm là các vòng tròn nhỏ

```
>> x = - pi0 : 0.05 : pi;
>> A = sin(x).*cos(x) ;
>> plot( x,A );
```




***Hình 4.12***

d) Hàm plot với tham số phức.




***Hình 4.13 Số phức được biểu diễn dưới dạng xoắn ốc.***

```
>> R = linespace (0,2);                    % tạo vector
>> tt = linespace (0,10*pi);               % tạo vector của góc
>> [x,y] = plot2cart(tt,r);                % chuyển toạ độ
>> z = x + i*y  ;                          %tạo vector số phức
>> plot(z) ;                               % in ra màn đồ hoạ
```

e) Tạo một file *.m thực hiện chuỗi lệnh sau

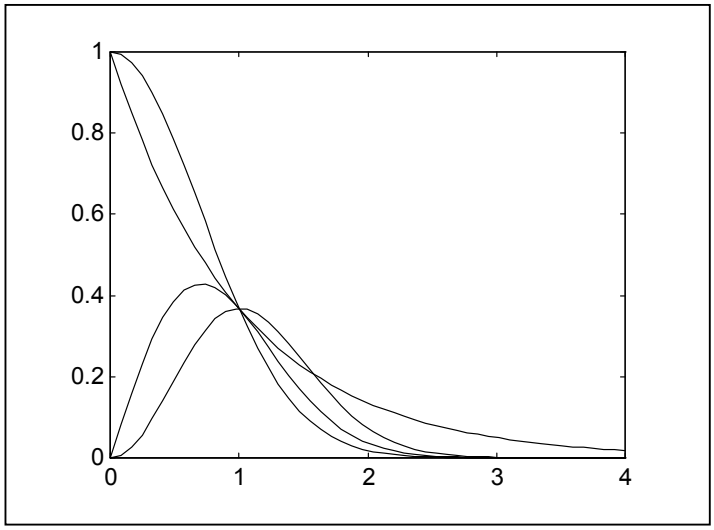
```
n = input ( ' Số điểm n = ');
a = input (' Khoảng xác định trên a = ');
b = input (' Khoảng xác định dưới b = ');
y = [ ];
e1 = [ ]; e2 = [ ];  e3 = [ ]; e4 = [ ];        % xoá e1 - e4
for i = 1 : n
xx = a + ( b - a ) * (i - 1) / (n - 1);
x(i) = xx;
e1(i) = exp( -(xx^2) );
e2(i) = xx^2 * exp ( -(xx^2) );
e3(i) = xx*exp ( -(xx^2) );
e4(i) = exp (-xx);
end
```

Script file trên sau khi thực hiện với tham số:

Số điểm n = 50

Khoảng xác định trên a = 0

Khoảng xác định dưới b = 5




***Hình 4.14 a Đồ thị của các hàm lấy từ ví dụ trên***

```
plot(x,e1,'w',x,e2,'w',x,e3,'w',x,e4,'w')
```

Các lệnh trên tương đương với chuỗi các lệnh dưới đây. Tuy nhiên với các bộ lệnh dưới cho phép chúng ta dễ dàng trong khi đọc cũng như vào dữ liệu.

```
>> x = linespace (a,b,u)
>> e1 = exp (-x^2);
>> e2 = (X^2) * exp (-x ^ 2);
>> e3 = x . * exp (-x^2);
>> e4 = exp (-x)
```

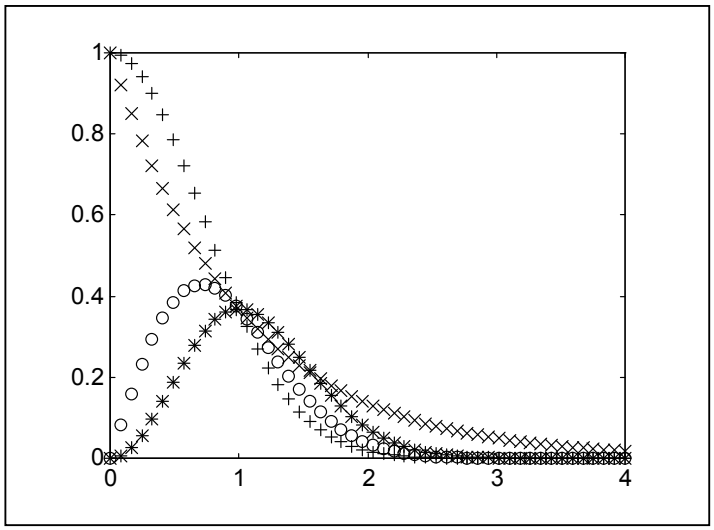
với

u = 50

a = 0

b = 30

```
>> plot (x, e1, 't', x, e2, 't', x, e3, x, 't', x, e3, '0', x, eu, 'x')
```



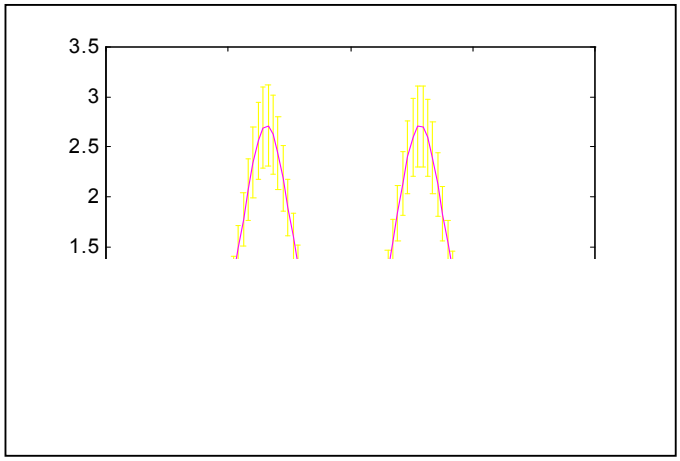

***b. Vẽ đồ thị cùng với khoảng sai số.***

**Errorbar (x, y, e, str)** Vẽ vector y theo x cùng với các thanh sai số có độ dài ei trên dưới yi.

**Errorbar (x, y, ℓ, u)** Vẽ đồ thị vector y theo x với ei và u là đoạn sai số của yi tương ứng với phần dưới và trên.

Ví dụ: Tạo đồ thị có khoảng sai số là 15%

```
>> x = linspace (0, 10, 50);
>> y = exp (sin (x));
>> delta = 0.15 *y;
>> errorbar ( x, y, delta)
```




***Hình 4.15 Đồ thị cùng đường sai số 15%***

### c. Vẽ hoạt hình ( comet )

Lệnh comet plot cho phép người sử dụng vẽ theo từng điểm trên màn hình gây hiệu ứng hoạt hoạ khi vẽ. Dưới đây là một số trong bộ lệnh comet.

**Comet (x, y)** Vẽ vector y trên trục x. Nếu tham số vào không có hay thiếu thì chương trình tự định ra chỉ số

**Comet (x, y, ℓ)** Vẽ theo hàm comet với phần kéo dài ℓ khi không khai báo chỉ số ℓ thì chương trình tự lấy giá trị = 0.1

### d.Hàm đồ hoạ.

**fplot (fku,lim,str)** Dùng để vẽ một hàm toán học bất kỳ được khai báo bởi mảng ký tự. Mảng ký tự có thể là các hàm chuẩn hay được định nghĩa bởi người sử dụng trong file M fku.m.

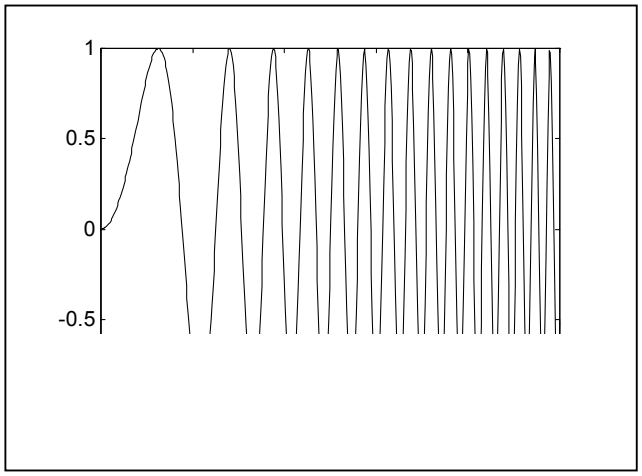
Vector lim = [Xmin Xmax] dùng để giới hạn khoảng xác định của đồ hoạ. Nó có thể bao gồm 4 thành phần trong đó thành phần thứ 3 và 4 là khoảng xác định trên trục y. Nếu biến str không khai báo trong hàm thì chương trình sẽ tự lấy các giá trị mặc định về kiểu đường hay mầu cho phần đồ hoạ.

**fplot ( fcu, lim, str, tol )** Vẽ đồ thị như trên với sai số liên quan nhỏ hơn giá trị tol

Ví dụ:

Dùng hàm fplot vẽ phương trình sin $x^2$

```
>> fplot(' sin(x^2)',[ 0 , 10 ] );
```




*Hình 4.16 Phương trình sin $x^2$ qua hàm fplot()*

## 4.3.2.Các hệ toạ độ trong mặt phẳng

Hàm plot cho phép người sử dụng vẻtên toạ đồ Đề các. Tuy nhiên 1 số bài toán trong kỹ thuật lại yêu cầu các hệ toạ độ khác. Để đáp ứng nhu cầu đó Matlab cung cấp 1 loạt các hàm cho phép tạo dựng đồ họa trên các loại hệ toạ độ.

**polar ( thet, r )** - Vẽ trên hệ tọa độ cực. Các phần từ của vector theta là các biến đo bằng radian và các phần tử của vector r là khoảng cách đến điểm gốc.

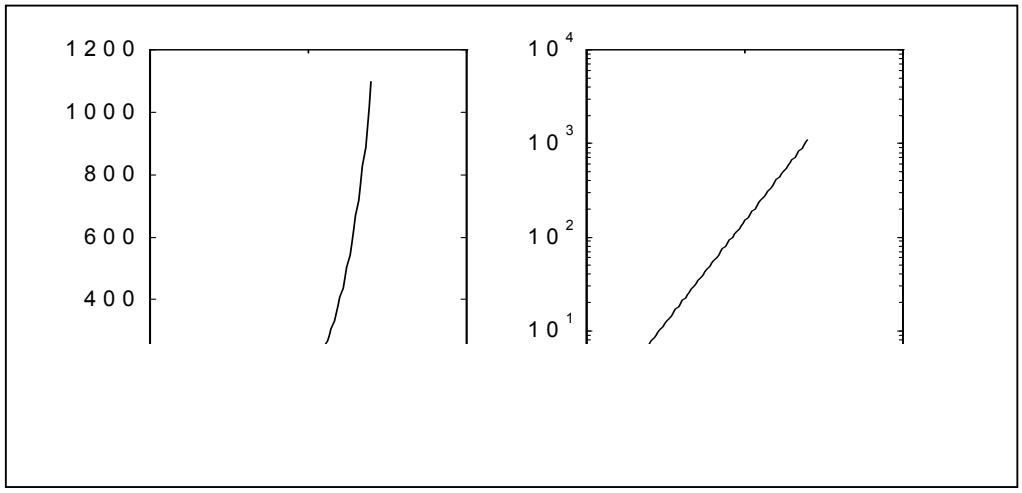
**semilogx ( x, y )** - Cho phép vẽ trên hệ toạ độ nửa trục loga, thay log10 được sử dụng cho trục x. Điều đó cũng tương đương với việc chúng ta viết plot (log10(x,y) nhưng sẽ không có lỗi với cả trong trường hợp log10(0).

**semilogy ( x,y )** - Vẽ trên hệ toạ độ của trục loga. Thang đo log10 được sử dụng cho trục y. Điều đó tương đương plot (x,log10(y) và cũng sẽ không báo lỗi khi viết log10(0).

**loglog ( x,y )** - Hàm cho phép vẽ trên hệ toạ độ loga 2 trục của hệ toạ độ đều dựa trên thang đo log10. Điều đó tương đương với việc plot(log10(x), log10(y)) và cũng không báo lỗi nếu ta sử dụng log10(0).

Ví dụ: a)

```
>> x = linespace (0,7);                    % tạo giá trị x
>> y = exp(x)                              % tạo y theo x
>> subplot (x,1,1);  plot( x,y );          % vẽ hàm chuẩn
>> subplot(2,1,2);   semilogy( x,y );      % vẽ hàm loga
```

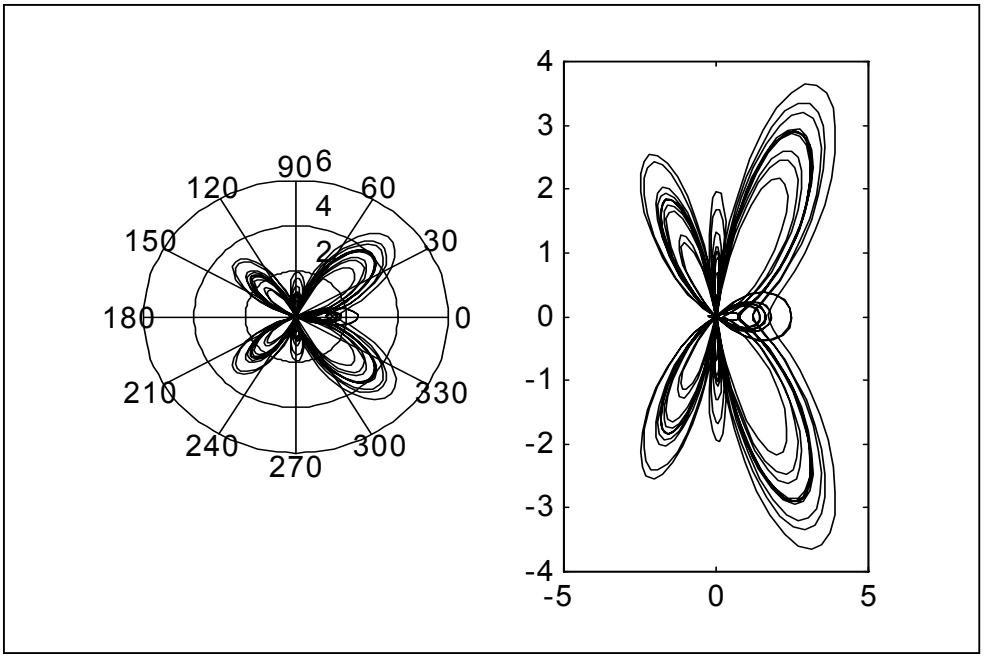



***Hình 4.16***

b)Vẽ hàm sau trên hê toạ độ cực theo công thức sau:

$$R = e^{cost} - 2\cos 4 + \left(\sin\frac{t^5}{12}\right)$$

```
>> t = linspace (0,22*pi,1100);
>> r = exp ( cos( t ) ) - 2*cos (u*t)+sin ( t./12) ).^5;
>> subplot (2,1,1);
>> p = polar(t,r);                    % vẽ trên hệ toạ độ cực
>> subplot (2,1,2)
>> [ x , y ] = pol2cart(t,r)          % giá trị từ hệ toạ độ cực sang hệ Đề các
>> plot(x,y);                         % polar_to_carttesian
```




*Hình 4.17 trên hệ toạ độ cực*

### 4.3.3. Mặt phẳng đồ hoạ cho số phức.

| | |
|---|---|
| **quiver ( x , y )** | Vẽ mũi tên cho mỗi cặp của hệ toạ độ cho bởi xij và yij cùng biến số và độ lớn là dxi và dyi. |
| **quiver ( x , y , dx , dy )** | Vẽ 1 mũi tên với tọa độ xi yi cùng biến số và độ lớn là tập dxij và dyij. |
| **quiver** | Vẽ mũi tên như trên nhưng hệ số tỷ lệ được cho bởi giá |

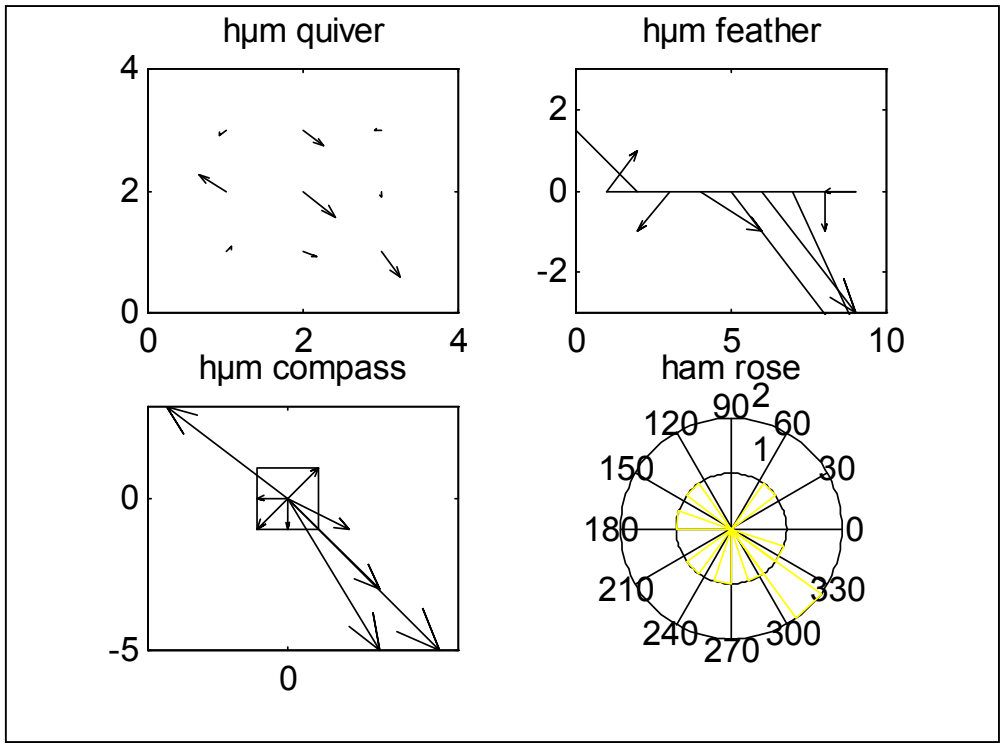
| | |
|---|---|
| **( x, y , ... , s )** | trị s.<br>Nếu s không được khai báo thì giá trị mặc định là 1 |
| **quiver**<br>**( x, y, ... , str )** | Vẽ 1 mũi tên với kiểu mẫu đường được xác định thông qua biến str |
| **feather ( z )** | Vẽ mũi tên chỉ ra phần thực và ảo của các phần tử hay ma trận của các số ảo z. |
| **feather ( x, y )** | Tương tự với feather(x+y*i) |
| **feather ( z, str )** | Vẽ mũi tên với việc sử dụng kiểu đường thẳng str |
| **compass ( z )** | Vẽ mũi tên khởi tạo từ gốc chỉ ra phần thực và ảo của các phần tử trong ma trận số ảo z |
| **compass ( x, y )** | Tương đương hàm compass (x + y*i) |
| **compass (z, str)** | Vẽ mũi tên sử dụng kiểu đường và mẫu sắc được định nghĩa bởi str |
| **rose ( v )** | Vẽ biểu đồ đối với biểu đồ tròn cho phép thể hiện tần suất của đối số trong vector v. |
| **rose ( u )** | Tương tự nhưng với khoảng xác định u |
| **rose ( x )** | Vẽ biểu đồ đối số với x là vector của các khoảng xác định. |

Ví dụ: Ma trận z được xác định như sau:

$$\% \; z = \begin{vmatrix} 1+i & 2-i & 3-5i \\ -4+3i & 5-5i & i \\ -1-i & 3-3i & -1 \end{vmatrix}$$

```
clf;
z =     [ (1 + i)     (2 - i)     (3 - 5*i)
          (-4 + 3*i)  (5 -5*i)    (i)
          (-1 - i)    (3 - 3*i)   (-1)    ]
subplot(2,2,1); quiver(real(z), imag(z));
title('hàm quiver');
subplot(2,2,2);feather(z);
title('hàm feather');
subplot(2,2,3); compass(z);
```

```
title('hàm compass');
subplot(2,2,4); rose(angle(z(:)));
title ( 'ham rose ');
```




***Hình 4.18 Đồ hoạ thu được từ các hàm Quiver, Teather, Compass, Rose.***

### 4.3.4. Lệnh kiểm soát.

**figure ( gef )** Hiển thị cửa sổ graphics hiện hành. Lệnh figure cũng có thể dùng để kích hoạt cửa sổ graphic hay tạo ra một cửa sổ đồ hoạ mới.

**clf** Lệnh dùng để xoá cửa sổ đồ hoạ hiện thời. Việc xoá vẫn thực hiện kể cả khi chúng ta đã dùng lệnh hold on

**clg** Lệnh xoá tương tự như elf và có thể không tồn tại ở các version mới của Matlab.

**clc** Lệnh xoá màn hình lệnh

**home** Chuyển con trỏ đến vị trí 'home' là vị trí ở trên bên trái màn hình.

**hold on** Giữ lại tất cả màn hình đã vẽ. Các lệnh sau sẽ thêm vào màn hình đồ hoạ chứ không xoá màn hình cũ đi.

| | |
|---|---|
| **hold off** | Là trạng thái mặc định của màn hình đồ hoạ khi ở trạng thái này các thừ thể đồ hoạ mới sẽ thay thế các thực thể cũ trên màn hình. |
| **hold** | Chuyển trạng thái từ on sang off và ngược lại |
| **ishold** | Trả giá trị 1 vài trạng thái hold là on trường hợp còn lại = off |
| **subplot** | Lệnh subplot được sử dụng để vẽ nhiều đồ thị lên cùng một màn hình đồ hoạ. subplot không dùng để vẽ mà chỉ dùng để xác định hay chia vùng màu đồ hoạ. |
| **subplot ( m, u, p )** | Chia màn hình đồ hoạ làm m hàng, n cột và p là phần cửa sổ hiện thời. Các cửa sổ con của màn hình đồ hoạ được đánh số theo hàm từ trái sang phải, từ trên xuống dưới. |
| **subplot** | Đưa màu đồ hoạ về chế độ mặc định là màn hình đơn. Điều đó tương đường subplot(1,1,1) |

Ví dụ:

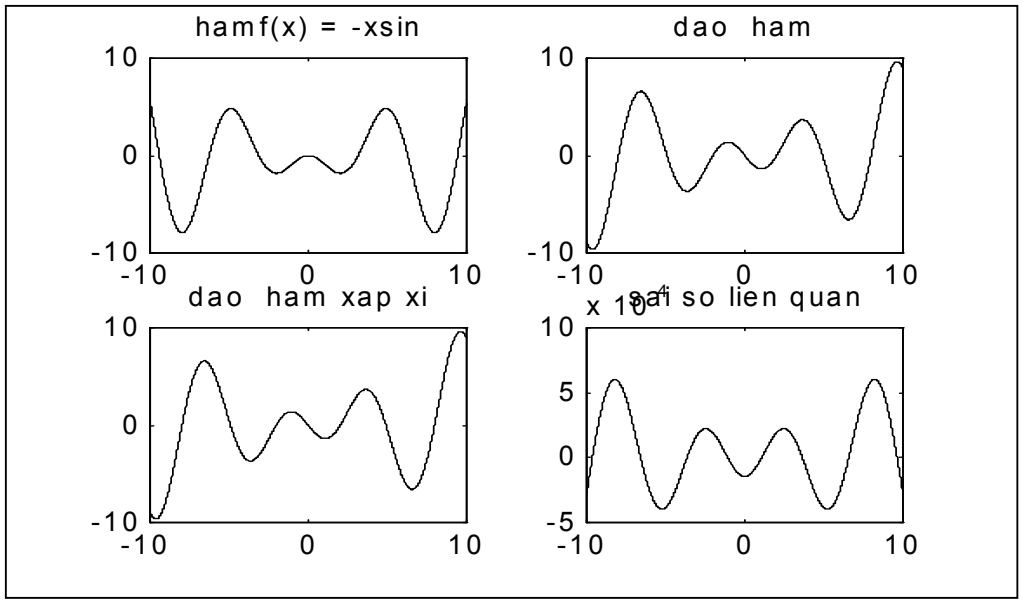
a) Tạo ma trận với các số ngẫu nhiên. Đoạn chương trình được ghi vào file *.m bất kỳ

```
clc; cfg;                          % xoá màn hình tương tác và màn đồ hoạ
for i = 1 : 25
home                               % đưa con trỏ về vị trí 'home'
A = rand(5)                        % tạo và in ma trận
end
```

b) Tạo hàm số sau:

```
% f(x) = -xsinx
% f'(x) = -xcosx-sinx
x = linspace (-10,10,1000)         % tạo ma trận x
y11 = (-x) *sin(x);                % tạo giá trị f
y12 = (-x) * cos(x) - sin (x);     % đạo hàm
y21 = diff(y11)./(x(2)-x(1));      % đạo hàm xấp xỉ
y22 = (y21 - y12 (1:999))./norm(y12);
subplot(2,2,1); plot (x,y11);
title ('hamf(x) = -xsin(x)');
```

```
subplot(2,2,2); plot (x,y12);
title ('đạo hàm');
subplot (2,2,3); plot (x(1:999),y21);
title ('đạo hàm xấp xỉ');
subplot(2,2,u); plot (x(1:999),y22);
title('sai số liên quan')
```




***Hình 4.19 Hàm -xsin(x) và các hàm liên quan***

Hàm subplot có thể sử dụng cho đồ hoạ 3 chiều và subplot có thể thay đổi kích thước

Ví dụ:

Hàm Mandelbrot và việc hiển thị bằng 3 phương pháp khác nhau với các giá trị đặc trưng phù hợp.

$z_0 = 0$

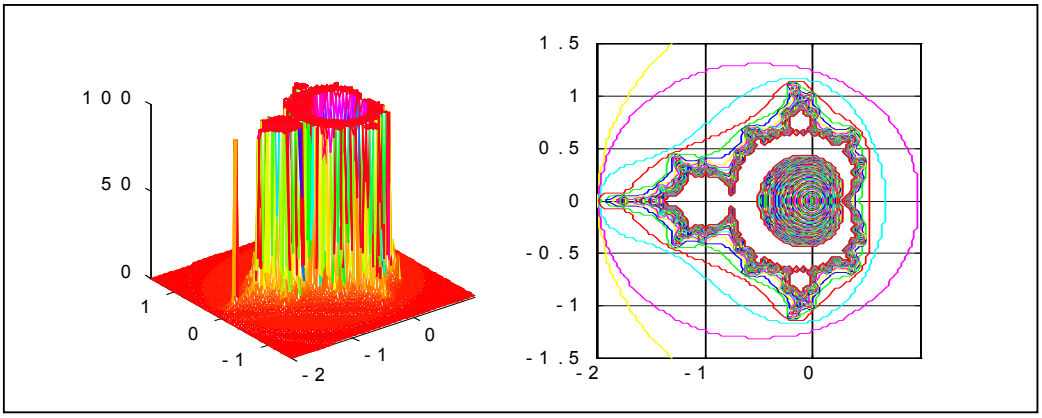
$z_{i+1} = z_i^2 + c$

Nếu $z_i$ là sai số thì c không phải là tậo của Mandelbrot. Số lặp lại tại mỗi điểm c của mặt phẳng phức được ghi vào ma trận Madelbrot cùng với việc giải vector

```
clear;
epsilon = 1e-14;                    % sai số
renum=input('số điểm thực renum =  ');
imnum=input('số điểm  ảo imnum = ' );
```

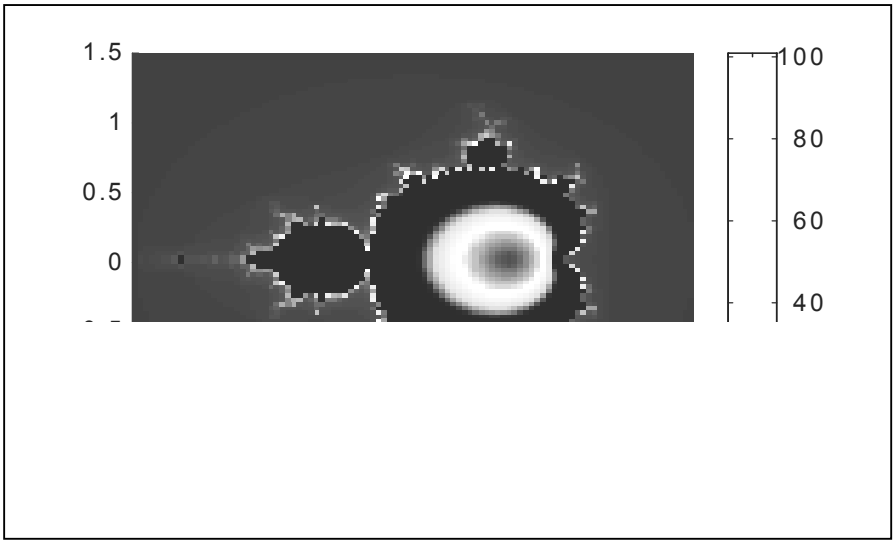
```
remin=-2; immin=-15;                    % khoảng chặn dưới
remax=1; immax = 1.5;                   % khoảng chặn trên
reval1 = linspace ( remin,  remax, renum );
imval1 = linspace( immin, immax, imnum );
[reval, imval] = meshgrid(reval1,imval1);   % tạo lưới grid trong khoảng
imvalreal =i mval;
imval = imval*i ;
cgrid = reval + imval;
% ---------------------------------------
for reind =1: renum                     % Vòng lặp cho phần thực của các số
        disp( [' reind = ', int2str( reind )] );
        for imind = 1 : imnum           % Vòng lặp cho phần thực của các số
        c = cgrid (reind, imind);
        numc = 0;
        zold = 0.0  +  i * 0.0;
        z = zold^2 + c;
                while( ( abs(z) <=2 ) & ( abs(z-zold) >= epsilon ) & ...
                ( numc < 100 ) )
                numc = numc + 1;
                zold = z;
                z = zold^2 + c;
                end;                    % End của còng lặp while
        Mandelbrot (reind, imind) = numc;
        end
end
% Các chức năng đồ hoạ hiển thị hàm Mandelbrot với
% 3 phương pháp khác nhau
% ---------------------------------------------------------------
clf;                            % xoá màu đồ hoạ
whitebg ( 'k' );                % thiết lập màn hình đen
subplot ( 2, 2, 1 );
mesh ( reval1, imval1, Mandelbrot );
axis( [-2  1  -1.5  1.5  0  100] )
subplot ( 2, 2, 2 );
contour (reval1, imval1, Mandelbrot, 100);
```

```
grid;
subplot ( 2, 1, 2)
surf ( reval, imvalreal, Mandelbrot );
view ( 2 );
shading flat;
colormap ( flipud ( jet ) );          % Xác lập hệ mầu JET
colorbar;                             % Hiển thị thanh bar mầu
axis ( [-2  1  -1.5  1.5] );
```




***Hình 4.15 hàm Madelbrot hiển thị 3 cách***

***a. In theo lưới*** ***b. Theo contour***




***c. In theo phổ mầu***

### 4.3.5 Thao tác và kiểm soát màn hình đồ hoạ

#### Axes , scaling và zooming.

Các trục khi vẽ thường được tự động biến đổ tỷ lệ kích thước sao cho khít với việc thể hiện các điểm trên màn hình cho phép có khung nhìn tốt nhất. Các giá trị thu được qua các hàm min và max.

*Ví dụ:*

```
[ min(x), min(y), max(x),min(y), min(x), max(y), max(x), max(y) ]
```

Tuy nhiên trong một số trường hợp mục đích hiển thị của người sử dụng khác so với việc tự động dàn xếp của Matlab. Chính vì vậy lệnh axis cho phép chúng ta thay đổi tỷ lệ của trục hay zoom trên cơ sở sử dụng mouse.

#### a. Axis:

| | |
|---|---|
| **axis** | - Trả lại giá trị khoảng giới hạn của lệnh in hiện thời vào 1 mảng. Với không gian 2D ta có [xmin xmax ymin ymax] và [xmin xmax ymin ymax zmin zmax] cho không gian 3D. |
| **axis ( υ )** | - Xét tỷ lệ theo vector $\upsilon$ với xmin $\upsilon_1$, xmax = $\upsilon_2$, ymin = $\upsilon_3$ ymax = $\upsilon_4$ với không gian 3D zmin = $\upsilon_5$ và zmax = $\upsilon_6$ |
| **axis (axis)** | - Khoá tỷ lệ giữ không cho Matlab tự động thay đổi tỷ lệ khi thêm các thực thể mới vào màn hình đồ hoạ. |
| **axis (str)** | - Đưa ra các kết quả khác nhau phụ thuộc vào chuỗi str |
| **'auto'** | - Cho phép Matlab tự động thay đổi tỷ lệ |
| **'equal'** | - Đưa ra trục toạ độ có tỷ lệ x,y tương đương |
| **'ij'** | - Quay trục y với hướng dưới cho chiều dương, trên cho chiều âm |
| **'x,y'** | - Xét lại trục y với hướng ban đầu (ngược với 'ij') |
| **'image'** | - Thay đổi kích thước của màn hình đồ hoạ sao cho các điểm có kích thước trên chiều dài và bề rộng như nhau |
| **'square'** | - Thay đổi màn hình đồ hoạ để tạo ra cửa sổ vuông |
| **'normal'** | - Thay đổi màn hình đồ hoạ cho ra kích thước ban đầu |
| **'off'** | - Dấu các trục ghi kích thước, không cho hiển thị |
| **'on'** | - Hiển thị các trục bị dấu |

#### b. Grid

| | |
|---|---|
| **grid on** | - Bật lưới vẽ trên màn hình đồ hoạ |
| **grid off** | - Tắt lưới |

**grid** - Chuyển trạng thái của lưới từ on sang off hay ngược lại

**c.zoom**

**zoom on** - Cho phép người sử dụng phóng to đồ hoạ 2 chiều bằng cách kích trái chuột lên trên màn hình, phải chuột sử dụng để thu nhỏ, nó cũng cho phép lựa chọn vùng bằng "click và drag" (kéo thả). Tỷ lệ sẽ thay đổi để vùng lựa chọn phù hợp với màn hình đồ hoạ.

**zoom off** - Loại bỏ lệnh zoom

**zoom out** - Thay đổi cho đầy khít màn hình

**zoom** - Chuyển trạng thái giữa on và off

Ví dụ:
a) Ví dụ cho việc biểu diễn đường tròn.

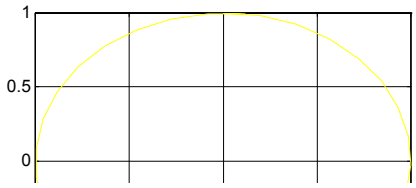
```
>> t = 0 : 0.2 : 2*pi + 0.2;
>> x = sin (t);
>> y = cos (t);
>> plot ( x,y,1,-1 ); grid on
```

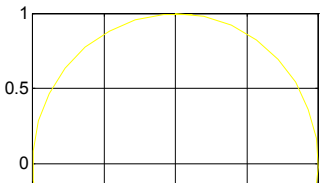
b) Thay đổi để tạo ra đường tròn đúng hình dạng

```
>>axis ('square');
>>grid on
```

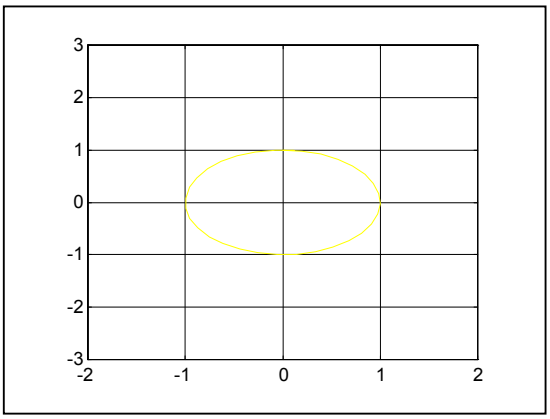
c) Bộ lệnh sau cho ra màn hình 13.15b

```
>> axis ('normal');
>> grid on;
>> axis ( [-2  2  -3  3] )
```




***Hình 4.20 a) Trước lúc căn chuẩn b) Sau khi căn chuẩn square***




*c) Sau khi trả lại trạng thái normal*

### 4.3.6. Văn bản trong màn hình đồ hoạ

Trong phần này chúng ta cùng đề cập đến các lệnh tạo text lên màn hình đồ hoạ. Tập các lệnh như title, zlabel cho phép viết các chữ chuẩn. Còn với text cho phép viết chữ lên mọi nơi thuộc màn hình đồ hoạ. Các lệnh viết chữ đều áp dụng trên cơ sở lệnh subplot

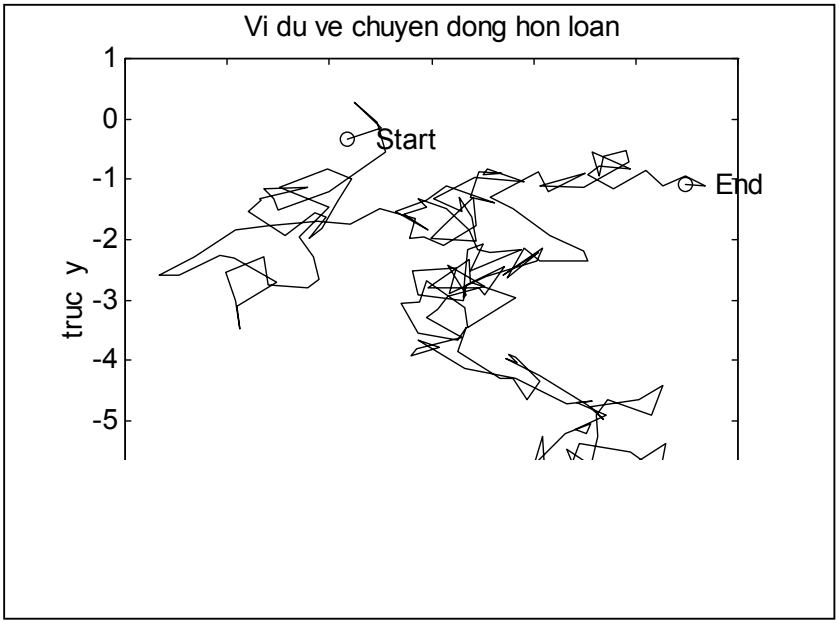
| | |
|---|---|
| **title ( txt )** | Viết mấy ký tự txt như dòng tiêu đề trên đỉnh căn giữa màn đồ hoạ |
| **xlabel ( txt )** | Viết mảng ký tự txt như như nhau căn giữa trục x |
| **ylabel ( txt )** | Viết mảng ký tự txt như như nhau căn giữa trục y |
| **zlabel ( txt )** | Viết mảng ký tự txt như như nhau căn giữa trục z |
| **text( x, y , txt )** | Viết chuỗi txt lên màn đồ hoạ tại vị trí x, y. Giá trị toạ độ x,ycó cùng tỷ lệ với lệnh plot. Nếu x và y là 2 vector thì giá trị txt được viết tại vị trí (xi, yi). Nếu txt là vector thì các giá trị txt được viết ra tại vị trí xi, yi |
| **text(x,y,txt,'sc')** | Viết ra chuỗi ký tự txt tại vị trí x, y trong hệ toạ độ với 2 điểm giới hạn là 0,0 và 1, 1 |
| **gtext ( txt )** | Viết ra chuỗi ký tự txt tại vị trí được xác định bởi dấu+ hay con trỏ được điều khiển bởi chuột. |
| **legend (st1,st2,...)** | Đưa ra màn hình các chuỗi ký tự st1, st2... trong hình hộp box mà vị trí của box có thể được điều khiển bở chuột |
| **legend (l1, st1, l2, st2...)** | Dùng như lệnh legend(st1, st2, ...) với l1 và l2 là kiểu của đường thẳng |
| **legen off** | Loại bỏ chức năng legend khỏi màn hình đồ hoạ |

Lệnh chuyển đổi từ số sang chuỗi có thể được dùng trong việc in bao gồm sprinf, num2str, int2str.

Ví dụ:

Chương trình mô tả chuyển động hỗn loạn bằng các bước chuyển động tự do.

```
n = input (' Nhap gia tri n = ');
x = cumsum( rand ( n,1) - 0.5 );
y = cumsum( rand ( n,1 ) - 0.5 );
clf;
plot (x,y);
hold on;
plot (x(1), y(1), '0', x(n), y(n),'0');
axs = axis;                                  % lấy giá trị min max
scale = axs(2) - axs(1);
text(x(1) + scale/30, y(1), 'start');
text(x(n) + scale/30, y y(n), 'kết thúc');
hold off;
xlabel('trục x'); ylabel('trục y');
title ('chuyển độn hỗn loạn');
```




***Hình 4.21 cho ra với số bước hoạt động n = 200***

### 4.3.7. Đọc dữ liệu từ màn đồ hoạ.

Lệnh ginput được sử dụng để lấy dữ liệu từ màn hình đồ hoạ. Lệnh này sẽ dùng để thay thế con trỏ trên cửa sổ. Con trỏ sẽ được dịch chuyển thông qua con chuột hay bàn phím bởi người sử dụng. Khi ấn chuột hay phím enter thì giá trị toạ độ sẽ được chuyển vào Matlab. Nếu giá trị toạ độ điểm không xác định thì Matlab sẽ giữ lại cho đến khi có lần dữ liệu khác.

***[ x, y ] = ginput**

Đọc toạ độ điểm từ màn hình đồ hoạ và trao kết quả cho 2 vector x, y. Vị trí của điểm được xác định bởi mouse hay bàn phím.

***[ x, y ] = ginput ( n )**

Đọc n toạ độ điểm từ màn hình đồ hoạ

***[ x, y, t ] = ginput ( ... )**

Trả giá trị toạ độ cho x và y; t là mảng ký tự tương ứng với 1 là phím trái chuột, 2 là phím phải, 3 là phím giữa. Nếu bàn phím được sử dụng thì t sẽ nhận giá trị cho bởi mã ASCII của phím.

***[ x, y ] = ginput( ..., 's' )**

Đọc giá trị toạ độ với giới hạn của màn đồ họa trong khoảng từ 0 đến 1.

***Waitforbuttonpress**

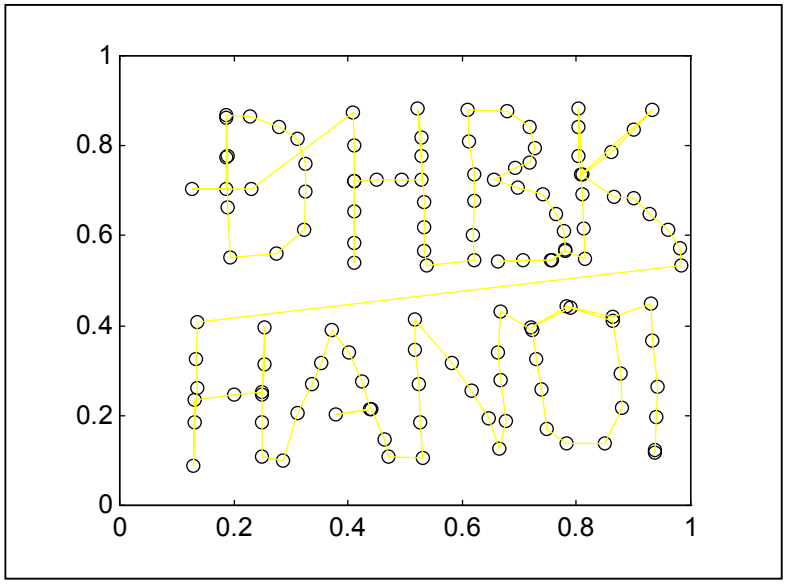
Dừng Matlab cho đến khi tác động lên chuột hay bàn phím. Nếu ấn chuột thì lệnh sẽ trả giá trị 0 nếu bàn phím sẽ trả giá trị 1.

Ví dụ:

Ví dụ cho sau đây sẽ minh hoạ cho việc dùng ginput và ***waitforbuttonpress*** trong lập trình Matlab để tạo nên nhiều tương tác đơn giản trên màn đồ họa.

```
n = figure;                                  % tạo cửa sổ đồ hoạ mới
disp ('vẽ các đường trong màn đồ hoạ');
disp ('bằng trái chuột');
disp ('kết thúc bằng phím phải chuột');
[x, y, t] = input(1);                        % đọc toạ độ từ màn đồ hoạ
plot (x, y, '0');
xϕ = x; yϕ = y;
hold; axis ( [0  1  0  1] )                  % khoá trục
while t ~= 2                                 % nếu không ấn phải chuột
[x, y, t] = ginput(1);
plot ( x,y,'0' );
xϕ = [xϕ    x];
yϕ = [yϕ    y];
```

```
end
line ( xϕ, yϕ );
disp ( ' ấn vào hình vẽ ' );
waitforbuttonpress;                    % đợi cho đến khi ấn vào phím
delete ( n)
```




***Hình 4.22 Tương tác màn hình đồ hoạ bằng chuột và bàn phím***

# CHƯƠNG 5

# ĐỒ HOẠ TRONG KHÔNG GIAN 3 CHIỀU

## 5.1. CÁC HÀM TẠO LẬP CONTOUR.

Lệnh contour trong không gian 2D và 3D đều được vẽ bởi hàm hai biến z = f(x,y) tương ứng với 2 hàm contour và contour3. Hai lệnh trên chỉ có thể được sử dụng trên lưới tứ giác.

### 5.1.1. Contour plots.

| | |
|---|---|
| **contour ( z )** | - Vẽ contour với các giá trị trong ma trận z. Các phần tử được dịch và biểu diễn trên mặt phẳng x, y. Nếu x là ma trận m x n thì tỷ lệ trên các trục tương ứng sẽ là n, m. |
| **contour ( z, n )** | - Vẽ được contour cho n cấp độ. Nếu n không xác định thì hàm sẽ lấy giá trị mặc định n = 10. |
| **contour ( z,v )** | - Vẽ đường contour với cấp độ được xác định bởi một vector |
| **contour(x,y,z)** | - Vẽ đường contour với giá trị thuộc ma trận z. Các thước tỷ lệ được xác định trên 2 trục tương ứng cho bởi vector x và y. |
| **contour( x, y ,z , n )** | - Vẽ trên n cấp độ với x, y là vector tỉ lệ trên các trục. |
| **contour ( x, y, z ,v )** | - Vẽ đường contour có cấp độ xác định bởi vector υ và tỷ lệ trên các trục được xác định bởi x và y |
| **contour ( ..,' str ')** | - Vẽ đường contour với việc sử dụng kiểu và màu sắc của đường được xác định bởi biến str. |
| **contour ( ... )** | - Tính toán cho việc thu dữ liệu vào ma trận c bởi việc sử dụng contour và clabel mà không vẽ đường, c là ma trận |

2 dòng chứa dữ liệu vẽ.

**contour3(x,y,z,n )** - Vẽ đường contour n mức độ trong không gian 3 chiều, nó không thể hiện các đường chiếu xuống mặt phẳng x,y việc trả giá trị vào ma trận contur cho bởi lệnh clabel.

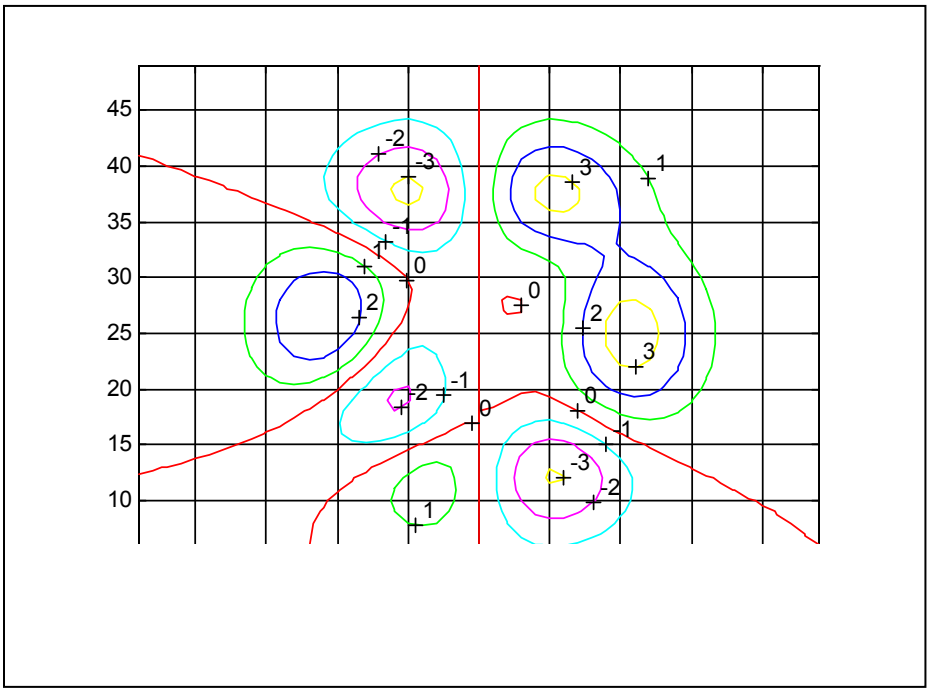
**clabel ( c )** - Cho chỉ số mức độ của contour c. Vị trí được xác định ngẫu nhiên. Ma trận c là ma trận contour được cho ra bởi lệnh contour hoặc contours.

**clabel ( c, υ )** - Trả lại giá trị chỉ số mức độ được xác định trong ma trận υ .

**clabel (c,'manual ' )** - Cho người sử dụng đưa ra chỉ số xác định mức độ tại điểm con trỏ tác động lên. Người sử dụng có thể dịch chuyển con trỏ bằng chuột hay bàn phím. Việc vào giá trị có thể thông qua phím chuột hay số trên bàn phím. Tiến trình kết thúc khi ấn phím enter.

## 5.1.2 Ví dụ

a) Giả sử ma trận z được mô tả như mặt của hàm 2 biến. Qua giá trị của z ta thu được đồ thị contour hình 5.1 bằng chuỗi lệnh dưới đây.

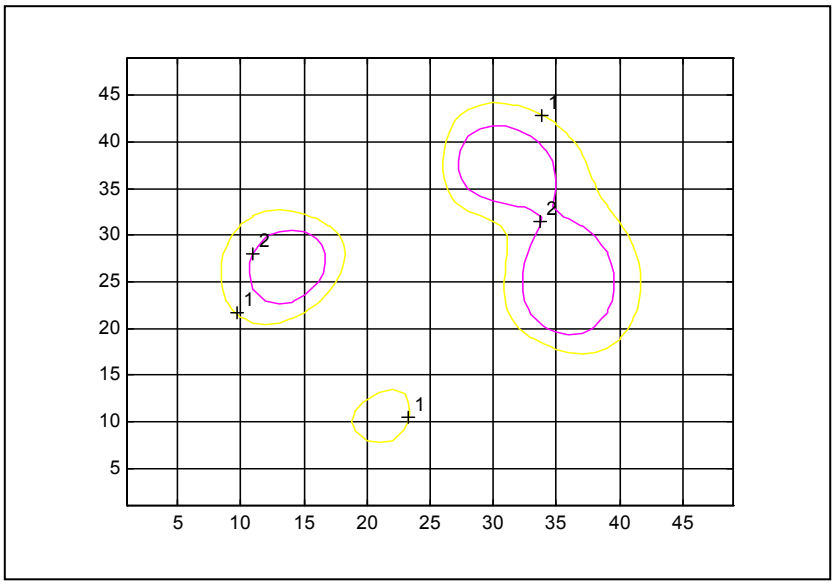



*Hình 5.1 Đồ thị contour cho bởi ví dụ 5.1a*

```
>> subplot ( 2 , 1 , 1 )
```

```
>>[X,Y] = meshgrid(-3:1/8:3);
>>z = peaks(X,Y).* sin(X)
>>v1 = -4 : -1;
>> v2 = 0 : 4 ;
>> contour ( z, v1, ' k ' );          % vẽ đường đặc với z dương
>> hold on;
>> contour ( z,v2 , 'k--' );          % vẽ đường đặc soloid với z âm
>> hold off;
>> subplot ( 2 , 1 , 1 );
>> c = contour ( z );
>> clabel ( c );                      % tạo nhẫn cho đường contour
>> grid on
```

b) Với ví dụ b chúng ta sử dụng zsmall. Chương trình chỉ thể hiện 2 mức độ như vậy zsmall lấy 2 giá trị 1 và 2.




*Hình 5.2 Đồ thị contour 2 mức độ cho bởi ví dụ 5.1b*

```
>> v = 1: 2 ;
>> zsmall = z;
>> c = contour ( zsmall ,v );
>> clabel ( c ) ;
>> size (c);
```

## 5.2. LƯỚI - GRID.

Để tạo được các đường contour chúng ta luôn phải tính các giá trị của z. Điều đó được mô tả như sau:

Ta xác định lưới của vùng nơi chúng ta sẽ vẽ đường contour. Vùng được xác định bởi 2 vector x và y với chiều dài n và m tương ứng với các giá trị x và y trên lưới. Giả sử rằng khoảng cách của các phần tử trên x và y là không bằng nhau. Khi ta xây dựng lưới mằng lệnh.

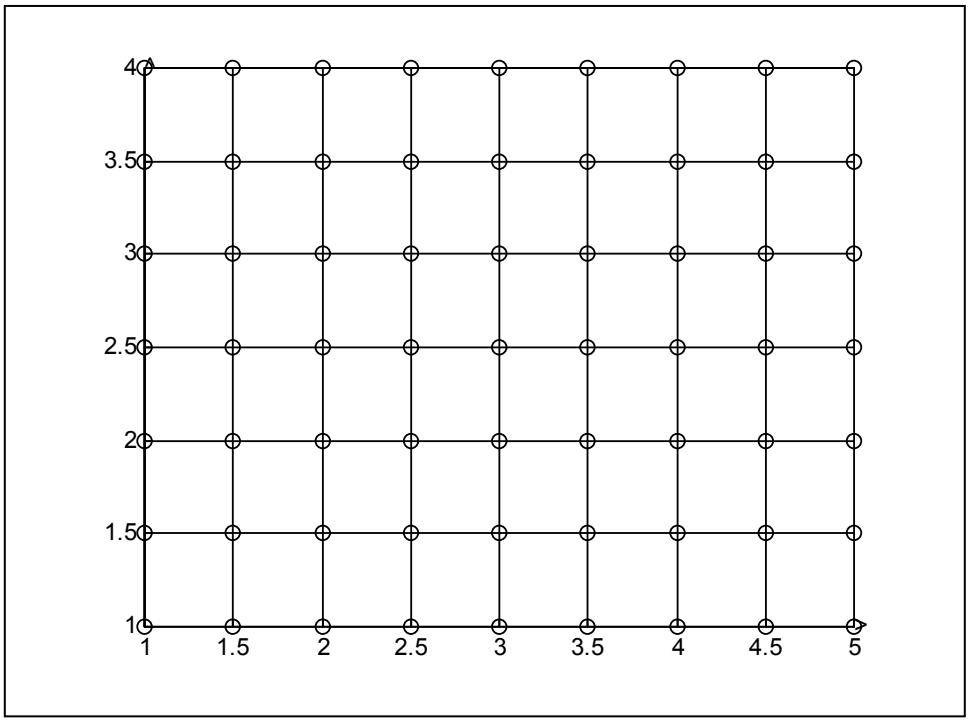
```
>> [ u  v ] = meshgrid (x,y);
```

Giá trị toạ độ điểm của lưới được lưu trữ vào 2 ma trận u, v.

- u chứa vector x với m dòng.

- v chứa vector y với n cột.

Hình vẽ dưới cho thấy ảnh của lưới [u , v]




*Hình 5.3 Lưới 4 x 5 tương ứng với x và y.*

Việc tạo lưới trụ hay lưới cầu cũng được thực hiện tương tự

### 5.2.1. Lệnh tạo lưới.

```
>> [ u, v ] = mesgrid ( x , y )
```

Đưa ra ma trận định dạng lưới theo toạ độ trục x,y từ 2 vector x, y. Vector có chiều dài n chứa toạ độ x và vector y có chiều dài m chứa toạ độ y. Ma trận u, v tạo thành có độ lớn tương ứng m x n. Ma trận biểu diễn bao trùm miền chữ nhật. Cặp toạ độ tương ứng ($u_{ij}$ và $y_{ij}$) với i = 1, ..., m j = 1, ..., n. Giá trị $z_{ij} = f(u_{ij}, v_{ij})$ tương đương với lệnh z = f( u,v )

```
>> [u,v,w] = meshgrid ( x , y , z )
```

Tạo ma trận lưới 3 chiều từ hàm 3 biến

```
>> [x, y, z] = cylinder ( r , n )
```

Trả giá trị tạo nên ma trận toạ độ điểm tương tự như meshgrid. Toạ độ tạo thành được xây dựng bởi mặt của hình trụ hoặc nón. Bán kính của hình trụ được biểu diễn bởi vector r tương ứng với n đường tròn tạo nên hình. Nếu n không được khai báo thì hàm sẽ lấy giá trị mặc định n = 20. Nếu cả r và n đều không được khai báo thì giá trị mặc định của hàm r = 1 và n = 20.

```
>> cylinder (r, u)
```

Vẽ hình trụ theo các dữ liệu đầu vào r và u.

```
>> [x,y,z] = sphere(n)
```

Trả các giá trị toạ độ không gian của hình cầu vào ma trận x, y, z với n là số mảnh bằng nhau của hình theo cách thể hiện hình theo tỷ lệ (n+1) x (n+1)

```
>> sphere(n)
```

In hình cầu ra màn hình thay vào việc trả giá trị vào các ma trận.

**Ví dụ 5.2 :**

Giả sử ta muốn định nghĩa lưới U,V trên 1 đơn vị mặt vuông với 5 diểm trên trục x và 4 điểm trên trục y.

* Đầu tiên ta phải định nghĩa 2 vector x và y.

```
>> x = linspace ( 0, 1, 5 );
>> y = linspace ( 0, 1, 4 );
>> [ u , v ] = meshgrid ( x , y )
```

* Tiếp theo

- Là tính toán giá trị hàm z = f(x,y) trên miền vùng đã định nghĩa lưới.
- Z = f(uv,)

Ví dụ:

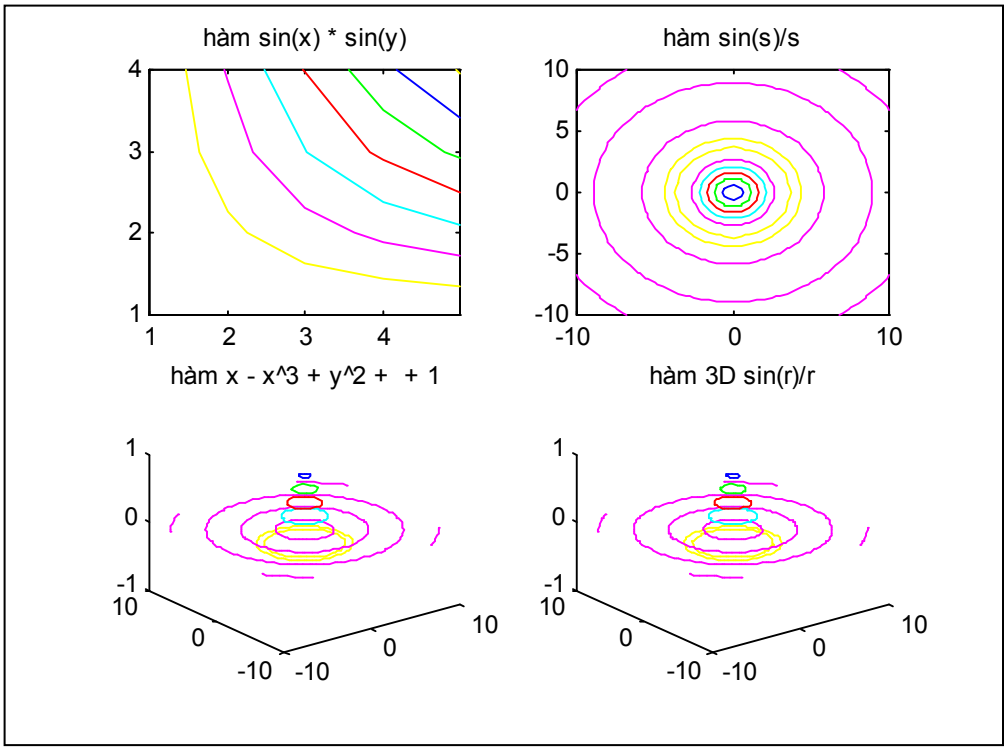
a) Giả sử chúng ta cần vẽ đường contour của 3 hàm sau:

$z_1 = f(x,y) = \sin x \cdot \sin y$ x, y $\in [0, \pi)$

$z_2 = f(x,y) = x - x^3 + y^2 + 1 \quad x,y \in [-5, 5]$

$z_3 = f(x,y) = \sin x\ ((x^2 + y^2)^{1/2}/(x^2 + y^2)^{1/2} \quad x,y \in [-10, 10]$

Đoạn chương trình sau tạo ra bởi lưới và các giá trị của hàm. Sau đó với hàm plot sẽ đưa kết quả ra màn hình đồ họa.



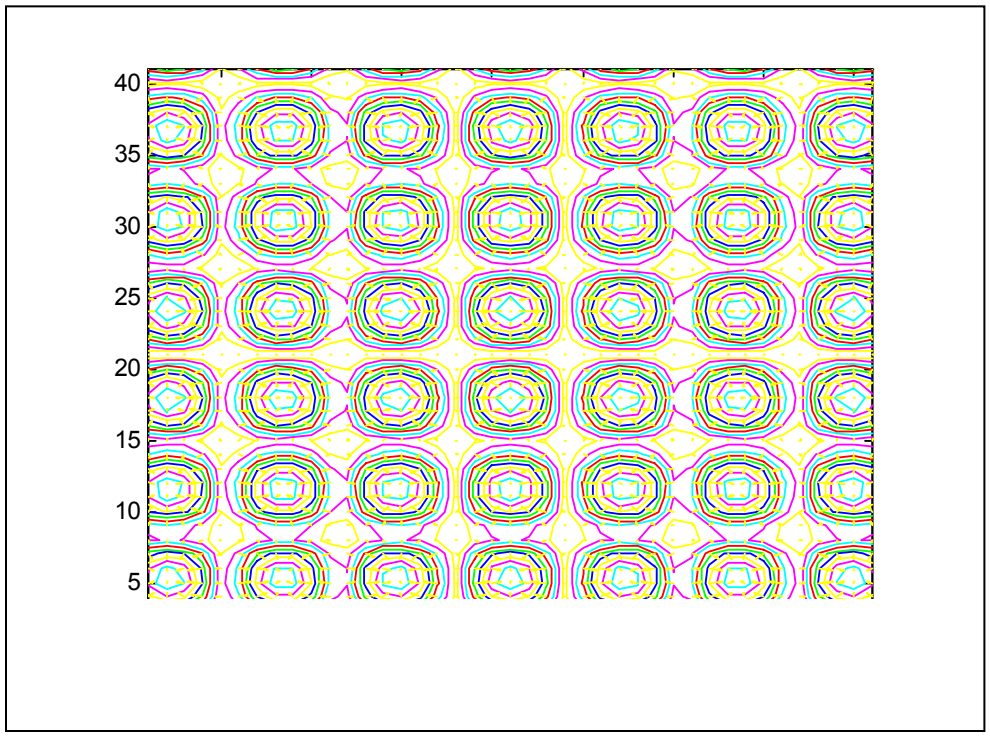

*Hình 5.4 Hình vẽ cho bởi ví dụ 5.2 a*

Phần chương trình nguồn của ví dụ 5.2 a

```
>> X = 0 : 0.2 : 3*pi;
>> Y = 0 : 0.25 : 5*pi;
>> [X,Y] = meshgrid (x,y);
>> z1 = sin (X).* sin(Y);
>> x = -5 : 0.25 : +5;
>> y = x;
>> [X,Y] = meshgrid( x, y );
>> z2 = X - X.^3 + Y.^2 + 1;
>> x = -10 : 0.5 : 10;
>> y = x
```

```
>> [X,Y] = meshgrid (x,y);
>> r = sqrt(X.^2 + Y.^2) + esp;
>> z3 = sin(r)./r;
>> clf;
>> subplot (2,2,1); contour(z1);
>> title ('hàm sin(x) * sin(y)');
>> subplot ( 2,2,2 ); contour ( x,y,z3 );
>> title ('hàm sin(σ)/σ');
>> subplot(2,2,3); contour3(x,y,z3);
>> title ('hàm x - x^3 + y^2 +  + 1');
>> subplot (2,2,4); contour3 (x,y,z3);
>> title (hàm 3D sin(r)/r');
```

b) Để thực sự làm sáng tỏ hình ảnh của hàm, chúng ta sẽ vẽ đường contour như là vẽ gradients. Giá trị gradient được tính bởi lệnh gradient và có thể được đưa ra màn hình bởi lệnh quiver.




***Hình 5.5 Mô tả cho ví dụ 5.2 b***

Hình vẽ sau cho ra bởi đoạn mã chương trình.

```
>> [x,y] = meshgrid ( -pi/2 : 0.1: pi/2 , -pi : 0.2 : pi );
```

```
>> z = abs ( sin( Y ).* cos( X ) );
>> [ DX, DY ] = gradient ( z, 0.1, 0.2 )
>> contour ( z );
>> hold on;
>> quiver ( DX, DY );
>> hold off;
```

## 5.3. ĐỒ HOẠ 3 CHIỀU.

Matlab sẽ tự dàn xếp cảnh nhìn và góc nhìn với bộ lệnh plot3 và bản chất plot3 tương đương với plot chỉ khác plot3 yêu cầu thêm vector thứ 3 hay ma trận đối số. Kiểm tra màu của đường có thể thay đổi thông qua biến string.

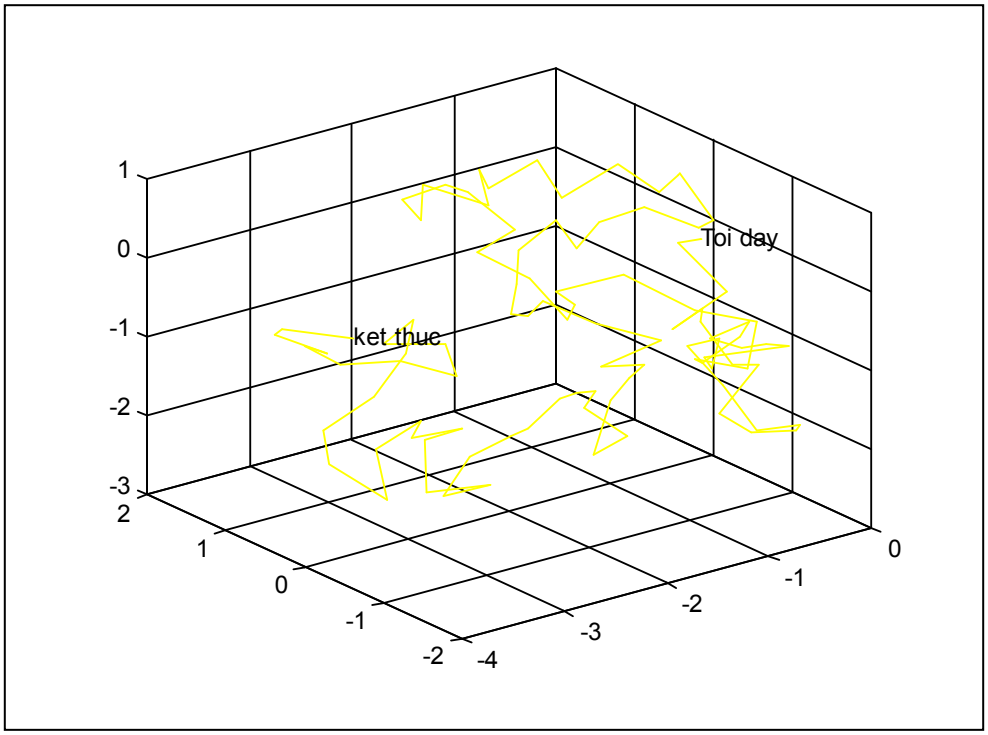
### 5.3.1 Lệnh vẽ đồ hoạ 3D thông thường

| | |
|---|---|
| **plot3( x, y, z )** | - Vẽ đồ hoạ thông qua điểm xác định bởi (xi, yi, zi). Các vector x, y, z phải có độ dài bằng nhau. |
| **plot3( X, Y,Z )** | - Vẽ đồ hoạ với các cột của ma trận X, Y, Z các ma trận phải có độ lớn như nhau, đồng thời chiều dài của các cột trong ma trận phải bằng nhau. |
| **plot3(x,y,z,srt )** | - Vẽ đồ hoạ tương tự lệnh trên với màu và kiểu đường được xác định bởi biến srt. |
| **plot3 ($x_1$, $y_1$, $z_1$, $srt_1$, $x_2$, $y_2$, $z_2$, $str_2$,...)** | - Vẽ đồ hoạ tại ($x_1$, $y_1$, $z_1$) với màu và kiểu đường xác định bởi $str_1$ và tương tự $str_2$ cho $x_2$, $y_2$, $z_2$... Nếu $str_1$ và $str_2$ không được định nghĩa Matlab sẽ tự chọn màu và kiểu cho đường. |

**Ví dụ 5.3**

Ví dụ trên sẽ tạo ra chương trình mô phỏng chuyển động hỗn loạin n bước trong không gian 3D.

```
n = input ( 'số bước chuyển động' );
x = cumsum ( rand (1,n ) -0.5 );
y = cumsum ( rand( 1,n ) -0.5 );
z = cumsum ( rand( 1,n ) -0.5 );
plot3 ( x,y,z );
text ( x( 1), y( 1 ), z( 1 ), 'Tới đây');
text ( x( n), y( n ), z( n ), 'kết thúc');
```




*Hình 5.6 Mô tả chuyển động hỗn loạn trong không gian 3D*

### 5.3.2 Các lệnh vẽ hoạt hình 3D

**comet 3 (x)** - Cũng tương tự như lệnh comet trong khoong gian 2D. Comet3 cho ra hình ảnh chuyển động hoạt hình mô phỏng lại quá trình vẽ

**comet3(x,y,z)** - Vẽ mô phỏng quá trình vẽ của hàm z = f(x,y). Điều đó có nghĩa điểm vẽ xác định bởi (xi, yi,zi)

**comet3(x,y,z,p)** - Cho ra tiến trình vẽ mô phỏng tương tự như trên với đôi kéo dài tính theo p. Chiều dài của p được cho trước như vector y. Nếu p không được xác định thì hàm số lấy giá trị mặc định là tập của các giá trị 0.1

Chữ trong cửa sổ không gian 3D được thể hiện tương tự như các bộ lệnh trong không gian 2D như title, text, xlabel, ylabel và zlabel.

## 5.4 MẶT LƯỚI TRONG KHÔNG GIAN 3D.

Matlab cho phép tạo ra các mặt lưới trên màn hình đồ hoạ của hàm z = f(x,y) theo từng bước sau đây.

- Xây dựng lưới grid

- Tính giá trị z = f(u,v) với U và V là 2 ma trận điểm toạ độ của các giá trị trên trục x và y tương ứng.

- Vẽ mặt lưới bằng lệnh đồ hoạ cho phép trong Matlab. Chú ý rằng lưới grid không cần thiết cho loại lưới tứ giác. Trong các trường hợp khác toạ độ lưới phải được cho vào khi gọi hàm.

## 5.4.1 Bộ lệnh tạo lưới

**mesh ( z )** - In các giá trị trong ma trận z như là các độ cao trên mặt lưới grid hình chữ nhật. Nối các điểm đó với các điểm xung quanh tạo nên mặt lưới (mesh)

**mesh ( z , c )** - Vẽ các giá trị của z lên trên mặt lưới grid chữ nhật với màu sắc của điểm được xácđịnh bởi tập các biến trong ma trận c.

**mesh( u, v, z, c )** - Vẽ hàm mặt lưới trên dữ liệu là các phần tử trong ma trận z. Các điểm láng giềng trong lưới được nối với nhau bởi các đường thẳng. Đồ hoạ được vẽ trong không gian 3D với góc chiếu phối cảnh với phần tử zij là chiều cao trên lưới grid(Uij, Vij)).

- Điểm nhìn được lấy tự động để được góc nhìn phối cảnh rộng nhất. Vị trí điểm nhìn có thể được thay đổi thông qua hàm view.

U: ma trận toạ độ theo x

V: ma trận toạ độ theo y

Z: ma trận toạ độ theo z

Zij = f(Uij, Vij)

C: ma trận màu cho mỗi điểm.

Nếu ma trận C không xác định thì C = Z được sử dụng. Nếu U và V là 2 vector có chiều dài m và n tương ứng thì z là ma trận có kích thước m x n và mặt lưới được xác định bởi 3 điểm (uij, Vi, Zij).

**meshc (...)** - Dùng để vẽ bước lưới cho các bề mặt lưới tương tự như lệnh mesh nhưng đồng thời vẽ thêm đường contour ở dưới bề mặt lưới

**meshz (...)** - Dùng để vẽ mặt lưới tương tự như lệnh mesh nhưng có thêm lưới grid trên mặt x,y

| | |
|---|---|
| **waterfall(...)** | - Tương tự như lệnh mesh nhưng lưới grid chỉ được vẽ theo 1 hướng. |
| **hidden on** | - Giữ cho Matlab không vẽ các đường khuất sau mặt lưới tạo bởi lệnh mesh |
| **hidden off** | - Cho phép Matlab vẽ các đường khuất sau mặt lưới |
| **hidden** | - Chuyển trạng thái hidden từ on sang off hoặc ngược lại |

## 5.4.2 Quay ma trận đồ hoạ 3D

Việc quay các ma trận đồ hoạ có thể được thao tác thông qua lệnh rot90.

| | |
|---|---|
| **rot90(A)** | - Trả lại giá trị của ma trận ảnh A qua phép quay $90^0$ theo chiều kim đồng hồ, lệnh thường được sử dụng với lệnh mesh |
| **rot90(A,k)** | - Trả lại giá trị ma trận A quay theo chiều kim đồng hồ 1 góc k * 90 |

Dưới đây là một số ví dụ minh hoạ cho các lệnh đồ hoạ trên đây là sáng tỏ ý nghĩa và cánh dùng cuả lệnh hay làm.

**Ví dụ 5.4:**

a) Trong Matlab có sẵn một số ma trận ảnh chữ có tên Matlabmatrix. Vì lý do ma trận quá to chúng ta chỉ quan sát trên cơ sở những gì tạo thành từ nó.

```
>> clf;
>> subplot(2,2,1); mesh( Matlabmatrix );
>> title('gcs nhìn chuẩn');
>> subplot(2,2,2); mesh(Matlabmatrix);
>> view([1 -4  2]);axis([0  200  0  20  0  3]);
>> title('viewed từ điểm [1  -4  2]');
>> subplot(2,2,3); mesh (Matlabmatrix);
>> view([-1  -2  -7]);
>> title('nhìn dưới lên 1 điểm nhìn [-1  -2  -7];);
>> subplot(2,2,4); spy (Matlabmatrix);
>> title('cấu trúc của ma trận ảnh Matlabmatrix');
```

với lệnh spy( ) cho phép mô tả một cách rõ ràng nhất về ma trận điểm ảnh.

b) Ví dụ b dùng Matlab để mô tả các mặt hình học sau:

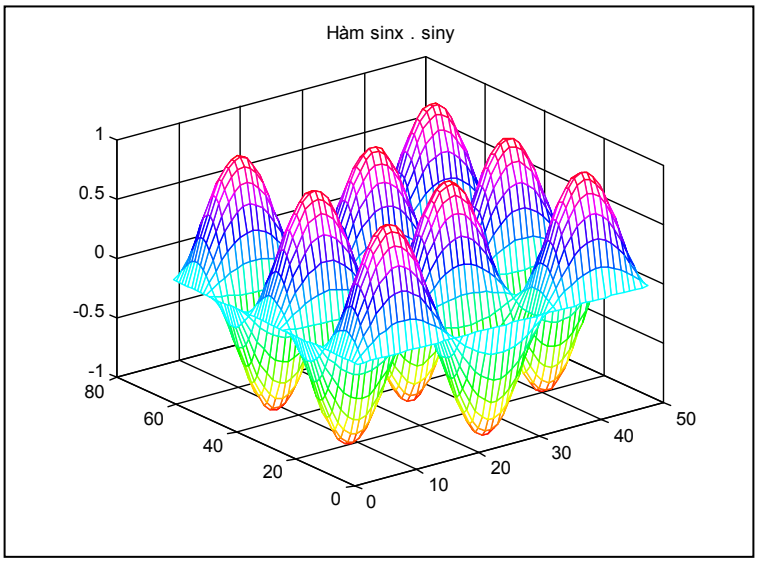
$z_1 = f(x,y) = \sin x \cdot \sin y \quad x,y \in [0, \pi]$

$z_2 = f(x,y) = +x = x^3 + y^2 + 1 \; x, y \in [-3, 3]$

$z_3 = f(x,y) = \sin\left(\sqrt{x^2 + y^2}\right) / \sqrt{(x^2 + y^2)} x, y \in [-8,8]$

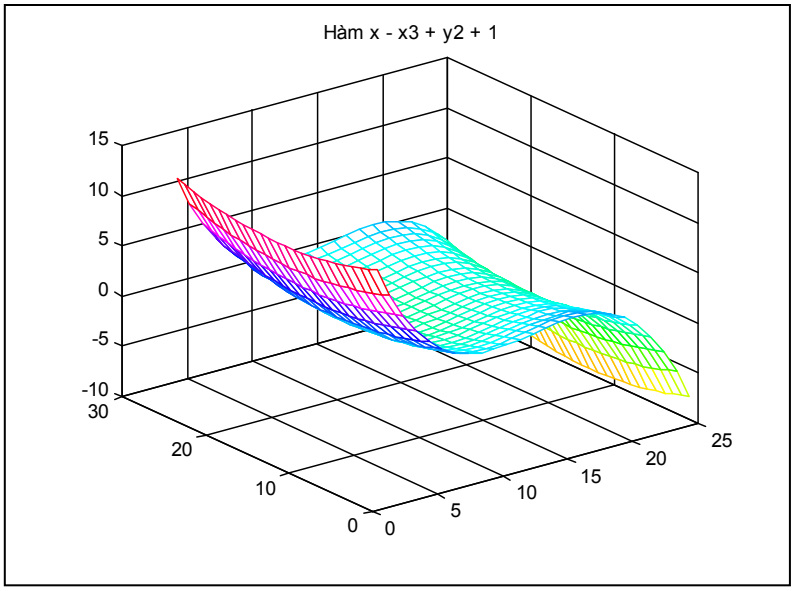
Việc định nghĩa x, y và $z_1$, $z_2$, $z_3$ được mô tả với các khoảng xác định như sau:

```
>> x = 0 : 0.2 : 3*pi;
>> y = 0 : 0.25 : 5*pi;
>> [X,Y] = meshgrid(x,y);
>> z1 = sin(X) . *sin(Y);
>> subplot(2,2,1); mesh(z1);
>> title('hàm sinx . siny');
```




***Hình 5.7 Mô tả ví dụ 5.4 a Hàm sin x * cos x***

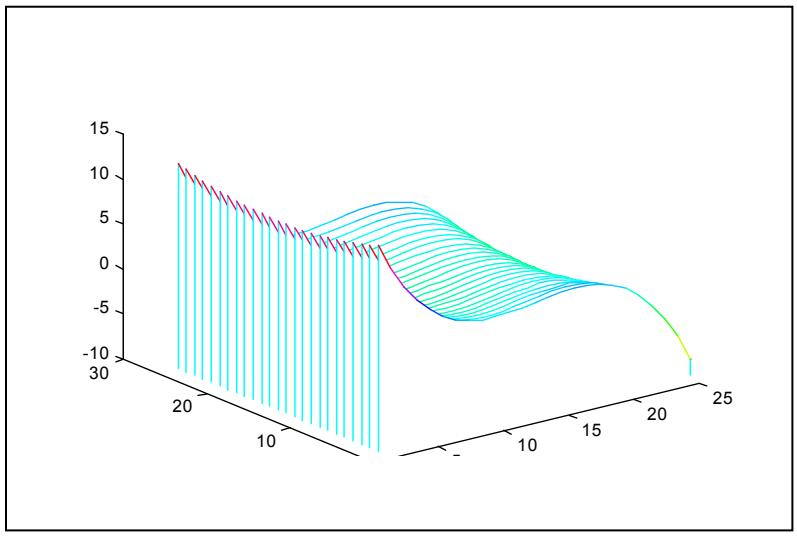
```
>> x = 03 : 0.25 : 3
>> y = x
>> [X,Y] = meshgrid (X,Y);
>> z2 = X - X.^3 + Y.^2 + 1;
>> subplot ( 2, 2, 2 ); mesh ( z2 );
>> title (' Hàm x - x^3 + y^2 + 1 ');
```




*Hình 5.8 Mô tả ví dụ 5.4 b Hàm z = X - X.^3 + y.^2 +1*

```
>> subplot ( 2, 2, 3 );
>> waterfall ( z2 );
```

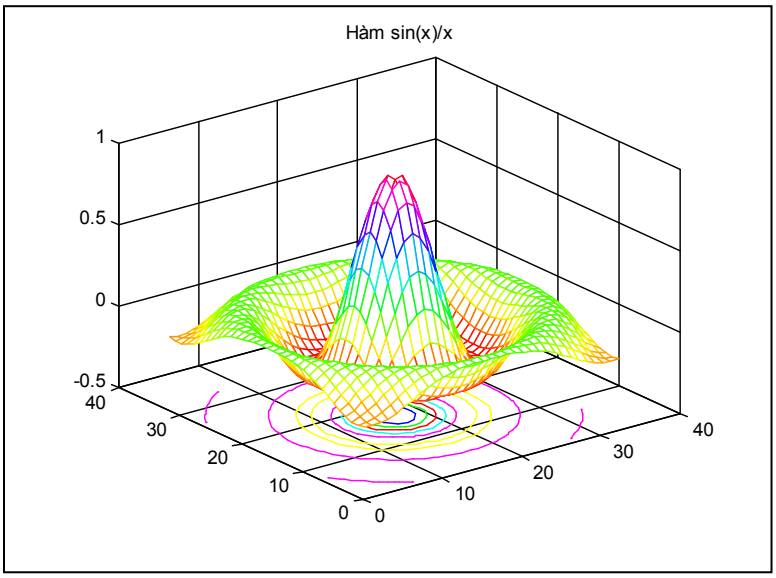
Hiệu ứng waterfall cho phép hiển thị các đường mô tả chiều cao của của từng đỉnh trên lưới.




*Hình 5.9 Mô tả hiệu ứng waterfall*

```
>> x = -8 : 0.5 : 8;
>> y = x;
>> [X,Y] = meshgrid ( x, y );
```

```
>> r = sqrt ( X.^2+Y.^2 );
>> z3 = sin ( r )./ r;
>> subplot ( 2, 2, 4 ); meslice ( z3 );
>> title ( ' Đồ hoạ hàm sin(x)/x ' );
```

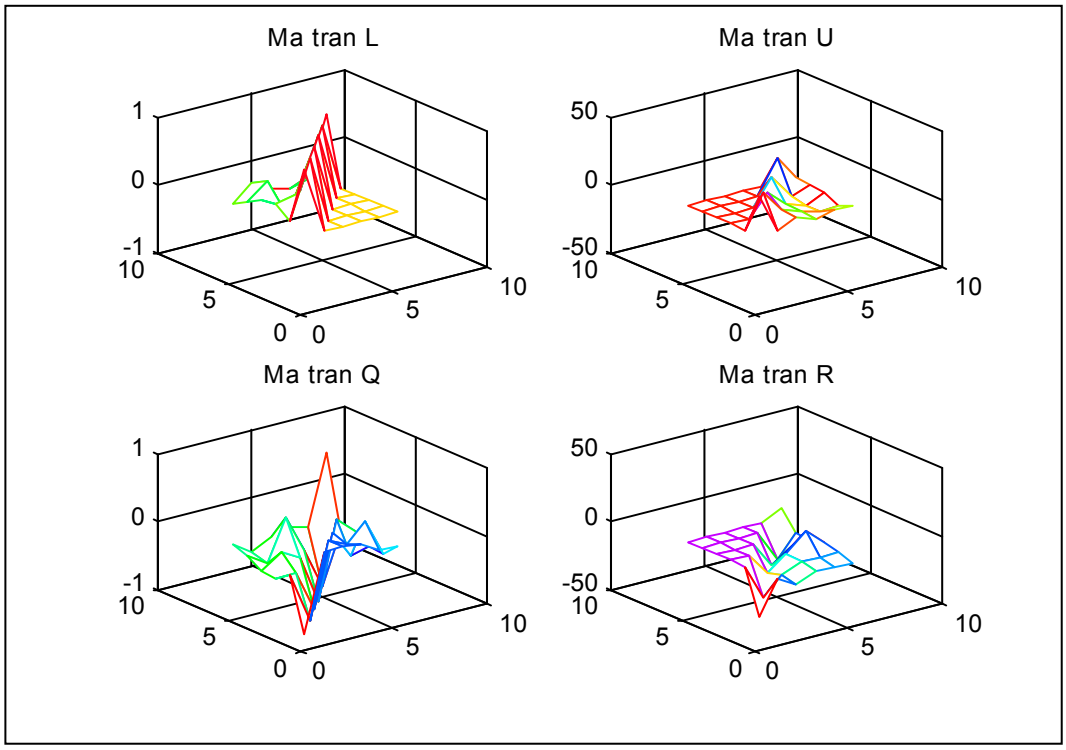



***Hình 5.10 Đồ thị lưới của phương trình hàm sin( x ) / x***

c) Ví dụ c xây dựng các ma trận LU và QR thông qua 2 hàm ℓu và qr từ ma trận A. Mã chương trình sau đây sẽ cho kết ra quả thu được lên màn hình đồ hoạ Hình 5.11

```
If ~ exist ( 'A' )
A=input ('Vào số liệu cho A :' )
else
disp ( 'Ma trận A đã tồn tại' );
end
[ L, U ] = lu ( A );
[ Q , R ] = qr ( A );
disp( ' Press any key to continue ' );
pause;
elf;
subplot ( 2, 2, 1 );
 mesh ( L ); title ( ' Ma trận L ' );
subplot ( 2 , 2 , 2 )
```

```
mesh ( U ), title ( 'Ma trận U');
subplot ( 2 , 2 , 3 );
mesh ( Q ), title( 'Ma trận Q' );
subplot ( 2 , 2 , 4 );
mesh ( R ), title ( 'Ma trận R' );
```




*Hình 5.11 Mô tả kết quả ví dụ 5.4 c*

Kết quả thu được với hàm bất kỳ là tên của đoạn chương trình cho ta dữ liệu sau trong phần trên chúng ta làm quen với những hàm tạo mặt lưới trong không gian 3D. Tuy nhiên khi tạo ra các bề mặt lưới có độ bóng hay ánh sáng tương tác lên bề mặt thì các hàm hay độ lệch được sử dụng sẽ khác và mang thêm thông tin về các dạng dữ liệu đó.

Để tìm hiểu về các thông tin các loại, dạng ánh sáng hay các giải thuật tạo bóng bề mặt, bạn đọc có thể tìm hiểu thêm trong giáo trình đồ hoạ hoặc giáo trình CAD.

## 5.5 ĐỒ HOẠ BỀ MẶT.

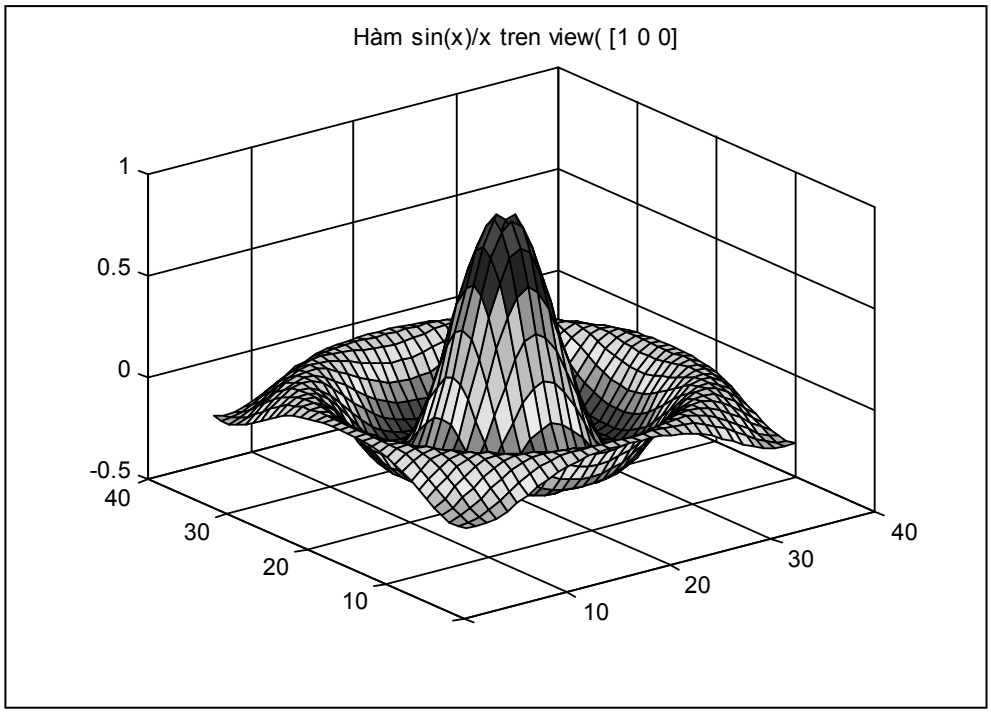
| | |
|---|---|
| **surf ( X, Y, Z, C )** | - Tạo mặt ba chiều lên màn hình đồ hoạ xác định bởi các toạ độ xij, yij, zij. Nếu x và y là các vector có độ dài m, n tương ứng, z là ma trận tương ứng m x n và bề mặt được định nghĩa bởi xi, yj và zij.<br>- Nếu X, Y không được định nghĩa Matlab sẽ sử dụng lưới grid hình chữ nhật đến giá trị của lưới được xác định bởi giá trị các phần tử trong ma trận C.<br>- Nếu C không được xác định thì giá trị mặc định của C = Z. |
| **surfc ( X,Y, Z, C )** | - Hàm thực hiện các chức năng tương tự như surf(...) ngoại trừ việc bao gồm cả chức năng vẽ các đường contour của mặt lên mặt phẳng dưới bề mặt lưới. |
| **surfl ( X , Y , Z , ls )** | - Tương tự như hàm surf (...) nhưng cần có thêm ánh sáng theo hướng ls = [ v,h ] hoặc ls = [ x, y, z ] mà trong đó các biến số tương tự như ở lệnh view. |
| **surfc ( X, Y, Z, ls, r )** | - Hàm thực hiện các chức năng như trên, tuy nhiên người sử dụng có thể cho thêm các thông tin liên quan như ánh sáng xung quanh, độ phản xạ khuyếch tán, phản xạ dải và độ phản chiếu.<br>- r = [ ambient, diffuse, specular, spread ] |
| **surfnorm ( X, Y, Z )** | - Hàm tạo bề mặt lưới với các chỉ số chức năng khác ở mức độ bình thường hay mặc định.<br>- [ Nx, Ny, Nz ] |
| **surfmorm ( X, Y, Z )** | - Đưa giá trị đơn vị vào bề mặt tạo các ma trận X,Y,Z nhưng không vẽ hàm lên màn đồ hoạ nxij, nyij, nzij là vector đơn vị xác định bởi xij, yij, zij. Giá trị đơn vị có độ dài 1. |
| **difuse(Nx,Ny,Nz,ls)** | - Trả lại độ phản xạ của mặt khuyếch tán cùng với các thành phần đơn vị cho bởi: Nx, Ny, Nz sử dụng luật Lambert ls ở đây là vị trí của nguồn sáng xác định bởi vecotr 3 thành phần. |
| **specular(Nx,Ny,Nz,lsN)** | - Trả lại độ phản xạ bề mặt cho các thành phần đơn vị Nx, Ny, Nz sử dụng nguồn sáng ls và góc nhìn v. |
| **pcolor ( Z )** | - Vẽ một mảng màu nhằm tạo với mỗi ô là 1 mầu xác định bởi các phần tử trong ma trận Z. |

| | |
|---|---|
| **pcolor( X,Y,Z )** | - Giống với lệnh surf(Z,Y,Z) và với góc nhìn view(2). |
| **file ( x, y, c )** | - Vẽ 1 đa giác với các góc xác định bởi toạ độ trong vector c. Nếu c là vecotr có cùng chiều dài với x và y. Nếu x và y là ma trận thì đa giác được vẽ bởi mỗi cột. |

## Ví dụ 5.5

a) Vẽ hình phương trình sinr/r với đồ thị đường mức ở dưới.

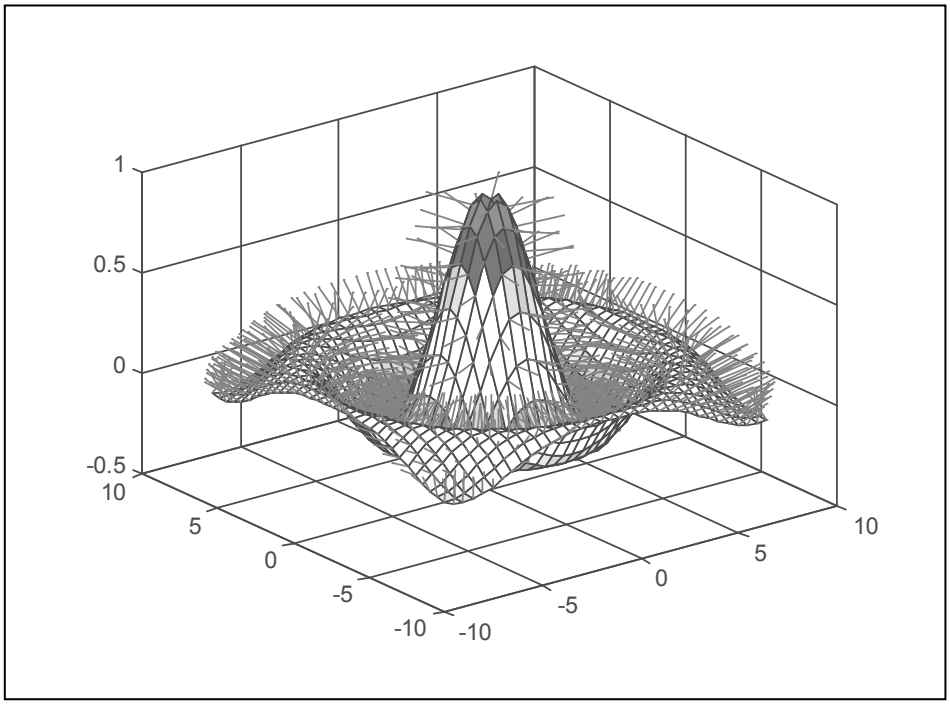
```
>> x = -8 : 0.5 : 8; y = x
>> [X Y] = meshgrid ( x, y );
>> R = sqrt ( X.^2+Y.^2 ) + eps;
>> Z = sin ( R)./ R;
>> surfc ( X,Y,Z );
>> title ( 'Hàm sin r/r' );
```




*Hình 5.11 Mô tả mặt lưới sin (r )/ r. Với dải mầu C phụ thuộc vào Z*

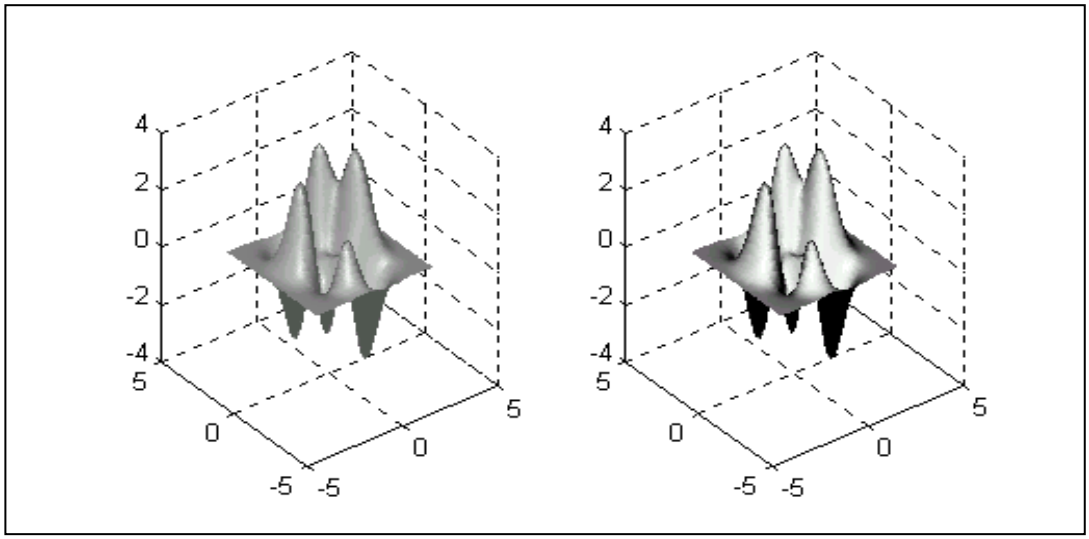
b) Với giá trị X,Y,Z xác định như ở phần a) với lệnh

```
>> surfnorm( X,Y,Z )
>>grid on
```



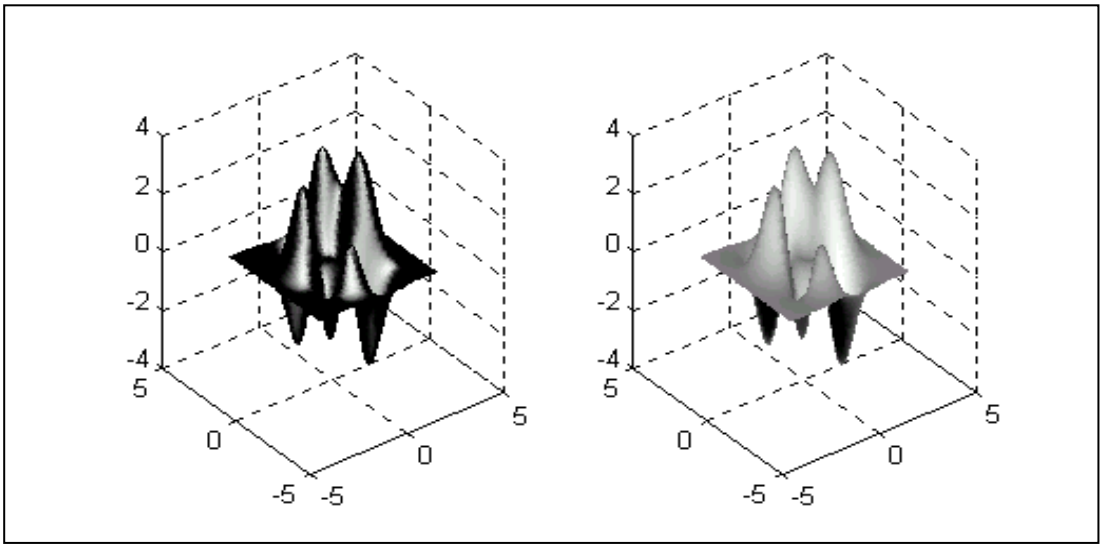

*Hình 5.12 Cho ra mặt lưới với đặc tính normal*

```
[ X  Y ] = meshgrid ( -3:1/8:3 );
Z = peaks( X,Y ).*sin ( X );
[ Nx Ny Nt ] = surfnorm ( Z );
S = [ -3  -3  2 ]              % vị trí nguồn sáng
k1 = [ 0, 1, 0, 0 ]            % mức độ phản xạ
k2 = [ 0, 0, 1, 1 ]            % mức độ ánh sáng xung quanh
```




***Hình 5.13 Mặt lưới với các độ phản xạ cho bởi nguồn sáng S***

```
surfl ( X, Y, Z, S ); shading interp;
surfl ( X,Y, Z, S, k1); shading interp;
```




***Hình 5.14 Mặt lưới 3 chiều với các mô hình ánh sáng khác nhau.***

```
surfl ( X,Y, Z, S, k2); shading interp;
D = diffuse ( Nx, Ny, Nz, S );
surf ( X, Y, Z, D ); shading interp;
colormap ( gray );
```

## 5.6 ĐIỂM NHÌN VÀ PHÉP PHỐI CẢNH.

Đồ hoạ sẽ dễ dàng được quan sát và gần với thực tế hơn nếu được nhìn từ các góc khác nhau. Lệnh view được dùng để thay đổi góc nhìn trên màn hình đồ hoạ. Nó cho phép khả năng xác định đồng thời cả điểm nhìn lẫn góc, phương độ nhìn và độ cao. Phép chiếu phối cảnh còn có thể thay đổi thông qua lệnh viewtx.

### View

>> view ( v, h )

- Xét góc nhìn cho màn đồ hoạ. Thanh v là góc phương vị với chiều dương trên mặt phẳng x, y được tính theo chiều kim đồng hồ. Chiều cao trên mặt phẳng được xác định mới thang đo h.

>> [ v  h ] = view

- Trả lại góc trên mặt phẳng x,y vào v và chiều cao trên mặt phẳng vào h.

>> View ( r )

- Đặt điểm nhìn vào vị trí xác định bởi r = [ x , y , z ]

>> view ( n )

- Xét góc nhìn phụ thuộc theo giá trị của n.

n = 2. Góc nhìn chuẩn hai chiều. Hay top-down nhìn từ trên xuống.

n = 3. Góc nhìn chuẩn 3D cho bởi ma trận 4 x 4 để chuyển đổi dữ liệu khi vẽ các thực thể đồ họa.

>> View ( T )

- Sử dụng góc nhìn xác định bởi ma trận 4 x 4T khi vẽ đồ hoạ.

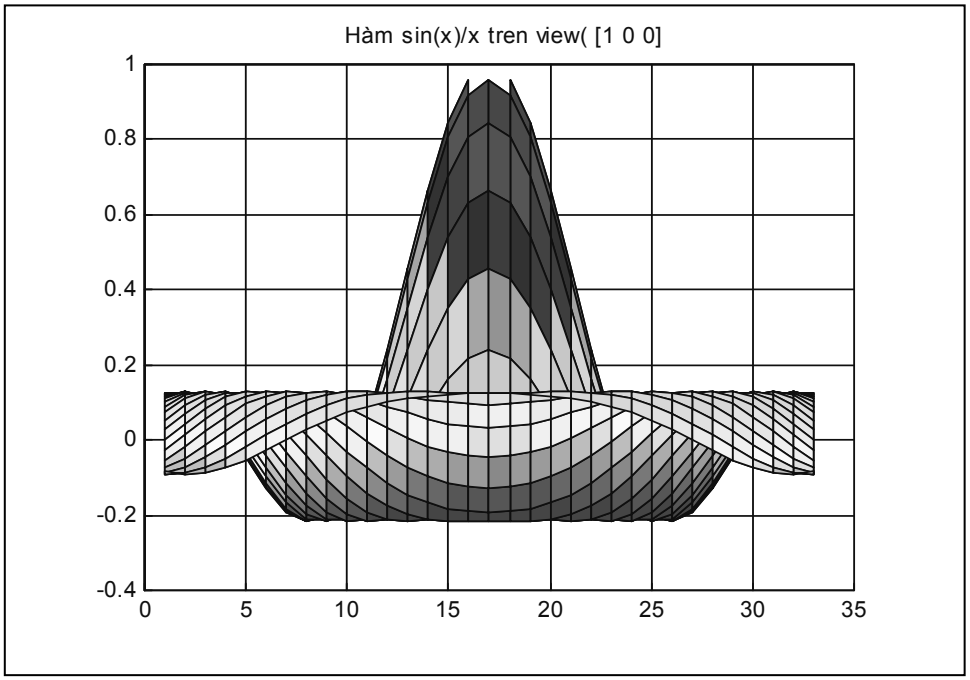
Viewtx ( v , h , s , r )

- Trả lại giá trị ma trận 4 x 4 xác định điểm nhìn và hướng nhìn.

### Ví dụ 5.6

Mô hình mặt sin (r)/r với góc nhìn từ cạnh sang

```
>> z = sin ( r )./ r;
>> surf( z );
>> title ( ' Ham sin(x)/x tren tren view [ 1   0   0 ]');
 >>grid on
>> view ( [1  0  0 ] ) ;
```




*Hình 5.15 bill với điểm nhìn từ cạnh sang*

b)Với lệnh view cho phép nhìn 3 chiều với hình ảnh 2 chiều

Ví dụ hình quả bóng với góc nhìn

```
>> view ( [ 1  1  1 ] )
```

Hình quả bóng mặt phẳng với góc nhìn trong không gian 3D

Lệnh surf và mesh có thể được sử dụng để vẽ các hàm trong cả hệ lưới grid không đều.

## Ví dụ 5.7

a) Chúng ta muốn nghiên cứu những số Mach trong lĩnh vực hàng không. Việc tính toán và tạo ra lưới gỉd đơn giản chỉ bằng mấy dòng lệnh text và hình vẽ số được tạo bởi Matlab. Lưới grid sẽ được cất vào hai ma trận $X_1$, $Y_1$ bởi ma trận mach chứa các giá trị $S_0$ Mach.

```
>> surf ( X1, Y1, Mach );
>> view ( 2 );
>> axis ( [-0.5  1.5  -1  1 ] );
>> shading interp;
```

Để nhìn thấy grid chúng ta sử dụng lệnh mesh với các ma trận cố định. Tuy nhiên muốn hiển thị được lưới các bạn cần phải sưu tầm được dữ liệu của ma trận X1 và Y1.

```
>> mesh( X1, Y1, ones ( size ( X1) ) ) ;
>> view ( 2 );
>> axis ( [-0.5   1.5   -1   1 ] );
```

## 5.7 SLICE TRONG KHÔNG GIAN 3D

Để nghiên cứu những hãng đồ hoạ 3 biến Matlab cung cấp cho chúng ta lệnh slice. Lệnh này dùng để vẽ cắt lát trong không gian 3D với mầu tại mỗi điểm trên bề mựat lưới tương ứng với các giá trị của hàm tại điểm đó.

>> slice ( V, xs, ys, zx, nx )

Vẽ phân lớp của hàm ba biến xác định bởi ma trận V. Ma trận V là tập của nx lớp dưới tính trên ba ma trận tạo bởi lệnh meshgrid cùng ba tham biến vector xs, ys và zs sẽ xác định những lát vẽ.

### Ví dụ 5.8

Cho hàm F(x,y,z) = $X^2$ + $Y^2$ + $Z^2$ trong một hình khối có giá trị

[ -1  1 ] x [ -1  1 ] x [ -1  1 ]

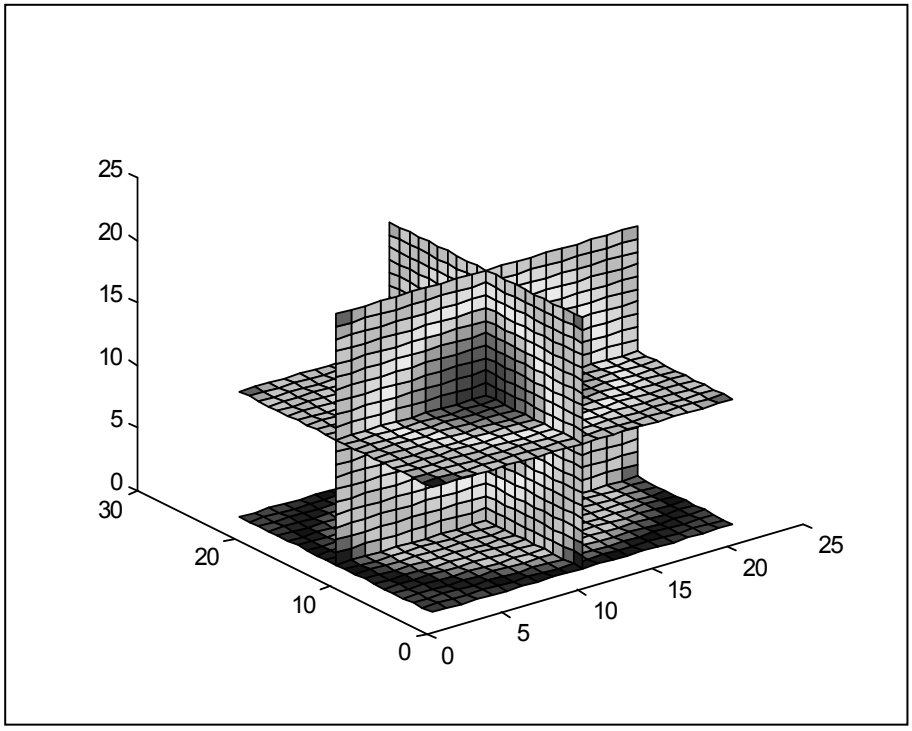
Đầu tiên chúng ta định nghĩa lưới grid trong không gian 3D thông qua hàm meshgrid và tính các giá trị của hàm F(x,y,z) thông qua các điểm trên lưới grid đó.

```
>> [ X, Y, Z ] = meshgrid ( -1 : .1 : 1 , -1 : .1 : 1 , -1 : 0.1 : 1 );
>> V = X.^2 + Y.^2 + Z.^2;
```

Số trên được tính tại $21^3$ điểm và chúng ta phải chọn những mảnh nào song song với trục toạ độ cần vẽ. Vector [1 3 2] cho biết rằng chúng ta muốn vẽ những mảnh 1, 3 và 2, 1.

Điều đó được thực hiện qua lệnh sau:

```
>> slice ( V, [ 1  1 ] , [ 1  1 ] , [ 1  1  1], 21 )
```




*Hình vẽ 5.16 với các mảnh được xác định bởi mặt phẳng X = 11, Y = 11, Z = 11 cùng với các mầu tương ứng.*

## 5.8 MẦU SẮC VÀ KIỂM SOÁT CÁC HỆ MẦU.

Trong lĩnh vực đồ hoạ, việc kiểm soát ánh sáng và màu sắc là những chức năng không thể thiếu được để cho ra những hình ảnh thật sắc nét. Trong Matlab người sử dụng được cung cấp một số hàm để kiểm soát màu sắc, ánh sáng, độ bóng v.v... của những hình ảnh được tạo ra.

Ví dụ:

Lệnh shading cho phép đặt cấu hình của việc in ra bề mặt lưới. Bề mặt có thể được vẽ ra có hoặc không có lưới cộng với thang mầu nội suy.

### 5.8.1 Các thuộc tính bề mặt.

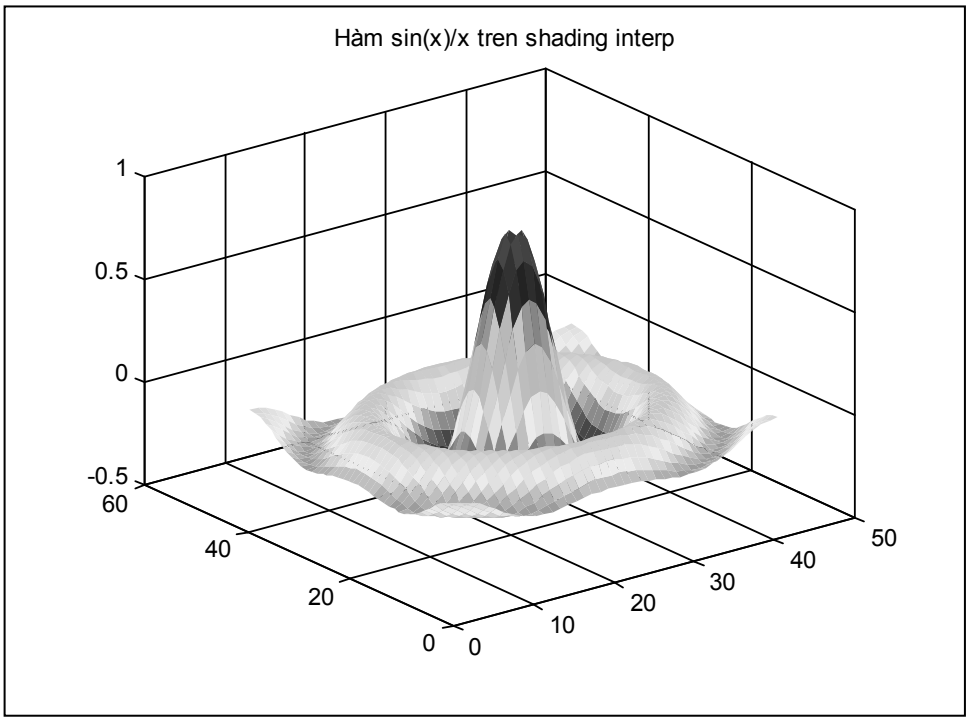
Kiểu shading, type : Dùng để vẽ bề mặt cùng một số thuộc tính sau

- Faceted dùng để vẽ lưới trên bề mặt và đây là kiểu mặc định của hệ thống

- Interp sử dụng những mầu nội suy trên bề mặt
- Plat tất cả các bề mặt được vẽ với cùng một màu từ các đỉnh của bề mặt.

**Ví dụ 5.9:**

Hàm sin (R) /R với hiệu ứng bóng shading interp




*Hình 5.15 Phân bố mầu trên bề mặt lưới với hiệu ứng bóng*

```
x = -10 : 0.5 : 10;
y = x;
[X,Y] = meshgrid ( x, y );
r = sqrt ( X.^2+Y.^2 );
z3 = sin ( r )./ r;
graymon;
surf( z3 );
shading interp;
title ( ' H    m sin(x)/x tren shading interp');
grid on
```

## 5.8.2. Giới thiệu về các hệ màu trong màn hình đồ hoạ.

Mô hình màu là một kỹ thuật cho việc biểu diễn mầu sắc của một thể mầu trên một hệ toạ độ mầu ba chiều bao gồm tập các màu nhỏ thành phần có thể trông thấy được trong hệ thống toạ độ mầu thuộc một gam màu đặc trưng.

Ví dụ như mô hình màu RGB (Red, Green, Blue): là một đơn vị tập các mầu thành phần sắp xếp theo hình lập phương của hệ trục toạ độ Đề các dùng biểu diễn một mầu bất kỳ.

Mục đích của mô hình màu là cho phép biểu diễn và chuyển đổi theo quy ước một số loại mầu từ gam mầu nay sang và phù hợp các màu sắc của các gam màu khác.

Màu căn bản trong loại gam màu cho các màn hình CRT (Cathode ray tube) được xác định bởi các màu gốc RGB, chúng ta có thể nhìn thấy trong mảng màu này một gam màu là một tập hợp nhỏ hơn của tất cả các màu có thể nhìn thấy được, vì vậy một mô hình màu không thể được sử dụng để định rõ tất cả có thể nhìn thấy.

Ba mô hình màu định hướng phần cứng là:

- RGB được sử dụng với các màn hình CRT.
- YIQ được sử dụng trong hệ thống ti vi mầu băng tần rộng
- CMY (xanh tím, đỏ tươi, vàng) sử dụng cho một số thiết bị in màu.

Không một mô hình màu nào trong các mô hình màu thưc tế trên có tính dễ sử dụng, bởi vì chúng không có mối quan hệ trực tiếp với các ý niệm màu của trực giác của con người bao gồm:

- Hue - sắc màu.
- Sturation - độ bão hoà.
- Lighness - độ sáng.

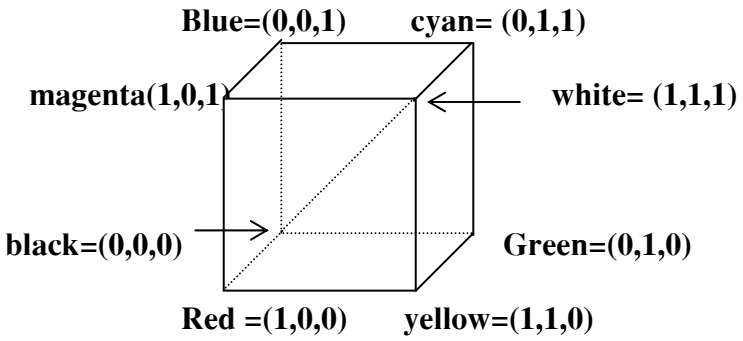
Bởi vậy các mô hình màu khác nhau đã được phát triển nhằm đến việc sử dụng chỉ cho một tiêu chí nhất định.

Chúng ta cùng tìm hiểu ba mô hình màu HSV, HLS và HVC, với mỗi mô hình màu cho ta một phương tiện phục vụ cho mỗi mục đích tiếp cận khác nhau khi thể hiện mầu sắc. Sự tồn tại của các mô hình mầu nêu trên dẫn đến nhu cầu về sự biến đổi từ một mô hình mầu sang mô hình RGB dựa theo sự biến đổi trong khoảng không gian mầu (X,Y,Z) của CIE (Commission Internationale de l' éclairage). Sự biến đổi này rất quan trọng bởi vì CIE là một tiêu chuẩn rộng khắp thế giới đối với tất cả các mô hình màu.

## 5.8.3. Mô hình màu RGB ( Red - Green - Blue ).

Màu đỏ, xanh lá cây, xanh gia trời (RGB) được sử dụng rộng rãi trên màn hình CRT và các loại màn hìnhđồ hoạ Raster mầu dựa vào hệ toạ độ Đề các. Những màu trên mô hình RGB được xây dựng trên cơ sở thêm vào từ những màu gốc, điều đó tạo nên sự đóng góp riêng của từng màu gốc để mang lại kết qủa.

Tập hợp màu nhỏ thành phần sắp xếp theo khối lập phương đơn vị được chỉ ra trong hình 5.17. Đường chéo chính của khối lập phương với sự cân bằng về số lượng từng màu gốc tương úng với các mức độ xám với đen là(0, 0,0 ) và trắng (1, 1, 1) .




*Hình 5.17 Mô hình không gian mầu RGB*

Gam màu được thể hiện trong hệ màu RGB được xác định bằng những đặc tính của hiện tượng phát quang của các chất phốt pho trong màn hình CRT. Hai màn CRT với 2 loại chất phốt pho khác nhau sẽ cho ra các gam màu khác nhau. Sự biến đổi màu được định rõ trong gam màu của một CRT so với gam màu của một CRT khác. Chúng ta có thể thay đổi gam mầu của một CRT này sang một CRT khác thông qua các ma trận chuyển đổi $M_1$ và $M_2$ từ không gian màu RGB của từng màn hình tới không fgian màu (X,Y,Z) . Công thức biến đổi :

$$\begin{vmatrix} X \\ Y \\ Z \end{vmatrix} = \begin{vmatrix} Xr & Xg & Xb \\ Yr & Yg & Yb \\ Zr & Zg & Zb \end{vmatrix} \begin{vmatrix} R \\ G \\ B \end{vmatrix}$$

Với Xr, Xg, Xb là các trọng số tương ứng với các mầu trong hệ RGB của màn hình, tương tự với Y, Z. Việc xác định M là hệ số chọn màu thông qua ma trận 3x3 của các trọng số trên. Chúng ta viết lại công thức như sau:

$$\begin{vmatrix} X \\ Y \\ Z \end{vmatrix} = M \begin{vmatrix} R \\ G \\ B \end{vmatrix}$$

Với M1 và M2 là những ma trận hệ số, sự biến đổi qua lại giữa gam màu của hai màn hình theo CIE được mô tả bằng $M_2^{-1} * M_1$. Điều đó có nghĩa việc biến đổi đó thông qua RGB của màn hình một tới RGB của màn hình hai. Nếu màu $C_1$ là gam màu của màn hình một nhưng nó không là gam màu của màn hình hai, màu tương ứng

$C_2 = M_2.M_1.C_1$ sẽ ở bên ngoài khối lập phương đơn vị và vì vậy sẽ không thể hiển thị được. Việc chuyển đổi tuy đơn giản nhưng không phải là giải pháp thoả mãn cho mọi giá trị. Vấn đề này có thể giải quyết bằng cách thay thế các giá trị của R, G hoặc B khi các giá trị này nhỏ hơn 0 bằng 0 và lớn hơn 1 bằng 1.

Các độ sắc màu cho mỗi mô hình phốt pho GRB luôn có sẵn như là các thông số kỹ thuật của công nghệ CRT. Nêú không, các thiết bị so màu cũng có thể được sử dụng để đo trực tiếp các giá trị toạ độ màu, hay một thiết bị đo quang phổ cũng có thể được sử dụng để đo P(λ) và sau đó chúng có thể được biến đổi thành toạ độ màu bằng các phương trình. (*), (**) và (***).

$$k = \frac{100}{\int Pw(\lambda)\,\overline{y}\,\lambda\,d\lambda} \quad (*)$$

$$x = \frac{X}{(X+Y+Z)}, y = \frac{Y}{(X+Y+Z)}, z = \frac{Z}{(X+Y+Z)} \quad (**)$$

$$X = \frac{x}{y}Y, Y = Y, Z = \frac{1-x-y}{y}Y \quad (***)$$

Biểu thị các toạ độ thông qua $(X_r, Y_r)$ cho màu đỏ, $(X_g, Y_g)$ cho màu xanh và $(X_b.Y_b)$ cho màu xanh da trời và xác định $C_r$ như sau :

$$Cr = Xr + Yr + Zr$$

Chúng ta có thể tính cho màu đỏ gốc theo:

$$Xr = Xr/(Xr+Yr+Zr) = Xr/Cr,\ Xr = xr*Cr$$

$$Yr = Yr/(Xr+Yr+Zr) = Yr/Cr,\ Yr = yr*Cr$$

$$Zr = (1 - xr - yr) = Zr/(Xr+Yr+Zr) = Z_r/C_r,\ Zr = zr*Cr$$

Với cách xác địng tượng tự cho Cg và Cb phương trình có thể được viết như sau **:**

$$\begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{bmatrix} = \begin{pmatrix} \mathbf{xrCr} & \mathbf{xgCg} & \mathbf{xbCb} \\ \mathbf{yrCr} & \mathbf{ygCg} & \mathbf{ybCb} \\ \mathbf{(1-xr - yr)Cr} & \mathbf{(1-xg -yg)\ Cg} & \mathbf{(1-xb - yb)Cb} \end{pmatrix} \begin{bmatrix} \mathbf{R} \\ \mathbf{G} \\ \mathbf{B} \end{bmatrix} \quad \mathbf{(4*)}$$

Các ẩn số Cr, Cg và Cb có thể được tìm bằng một trong hai cách. Cách thứ nhất, những thể sáng Yr, Yg và Yb của màu đỏ màu xanh da trời sáng nhất có thể được

đo với một quang kế chất lượng cao. Những thước đo thể sáng này có thể được kết hợp với các đại lượng $y_r$, $y_b$ và $y_g$ đã biết để tính các giá trị.

$$Cr = Yr/y_r, Cg = Yg/y_g , Cb = Yb/y_b$$

Những giá trị này sau đó được thay thế vào phương trình (4*) và ma trận chuyển đổi M được diễn tả trong quan hệ của các đại lượng đã biết ($x_r$, $y_r$) , ($x_g$, $y_g$), ($x_b$, $y_b$), Yr, Yg,Yb.

Chúng ta cũng có thể loại những biến không biết từ phương trình (4*) nếu chúng ta biết hoặc đo được các giá trị Xw, Yw và Zw của màu trắng được tạo ra khi R=G=B=1. Trong trường hợp này phương trình (4*) có thể được viết lại như sau:

$$\begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} = \begin{bmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ (1-x_r - y_r) & (1-x_g - y_g) & (1-x_b - y_b) \end{bmatrix} \begin{bmatrix} C_R \\ C_G \\ C_B \end{bmatrix}$$

Lời giải cho Cr,Cg,Cb là các giá trị cần tìm và với những giá trị kết quả thu được được thay thế cho phương trình (4*). Mặt khác các giá trị của màu trắng được xác định bởi $x_w$ ,$y_w$,$z_w$ và Yw trong trường hợp này sẽ được tìm ra với phương trình trên cho chúng ta các đại lượng cần tìm:

$$Xw = x_w Yw/y_w, \quad Zw = z_w Yw/y_w.$$

### 5.8.4 Mô hình màu CMY (Cyan, Magenta, Yellow - xanh, đỏ tươi, vàng)

Ba mầu CMY là mầu bù tương ứng cho các màu đỏ, xanh lá cây, xanh da trời và chúng được sử dụng như những bộ lọc loại trừ các màu này từ ánh sáng trắng. Vì vậy MCY còn được gọi là các mầu bù loại trừ của các màu gốc RGB.

Tập hợp màu thành phần biểu diễn trong hệ toạ độ Đề-các cho mô hình màu CMY cũng giống như cho mô hình màu RGB ngoại trừ màu trắng (ánh sáng trắng ) được thay thế màu đen (không có ánh sáng) ở tại nguồn sáng. Các màu thường được tạo thành bằng cách loại bỏ hoặc được bù từ ánh sấng trắng hơn là được thêm vào những mầu tối.

Những kiến thức về CMY là quan trọng, khi xem xét các thiết bị in màu trên giấy. Chẳng hạn như in tĩnh điện hay máy in phun. Khi bề mặt giấy được bao phủ bởi lớp mực màu xanh tím, sẽ không có tia màu đỏ phản chiếu từ bề mặt đó. Mầu xanh tím đã loại bỏ phần mầu đỏ phản xạ khi có tia sáng trắng, mà bản chất là tổng của 3 màu đỏ, màu xanh lá cây, xanh da trời.

Vì thế ta có thể coi màu xanh tím (cyan) là màu trắng trừ đi màu đỏ và đó cũng là màu xanh da trời cộng màu xanh lá cây. Tương tự như vậy ta có màu đỏ thẫm (magenta) thụ màu xanh lá cây (green) vì thế nó tương đương với màu đỏ cộng màu xanh da trời. Và cuối cùng màu vàng (yellow) hấp thụ màu xanh da trời, nó sẽ bằng mầu đỏ cộng với màu xanh lá cây.

Khi bề mặt của thực thể được bao phủ bởi xanh tím và vàng, chung sẽ hấp thụ hết các phần mầu đỏ và xanh dương của bề mặt. Khi đó chỉ tồn tại duy nhất mầu xanh lá cây bị phản xạ từ sự chiếu sáng của ánh sáng trắng. Trong trường hợp khi bề mặt được bao phủ bởi cả 3 mầu xanh tím, vàng và đỏ thẫm, hiện tượng hấp thụ xảy ra trên cả 3 mầu đỏ, xanh lá cây và xanh da trời, do đó là màu đen sẽ là mầu của bề mặt. Những mối liên hệ này có thể được miêu tả bởi phương trình sau:

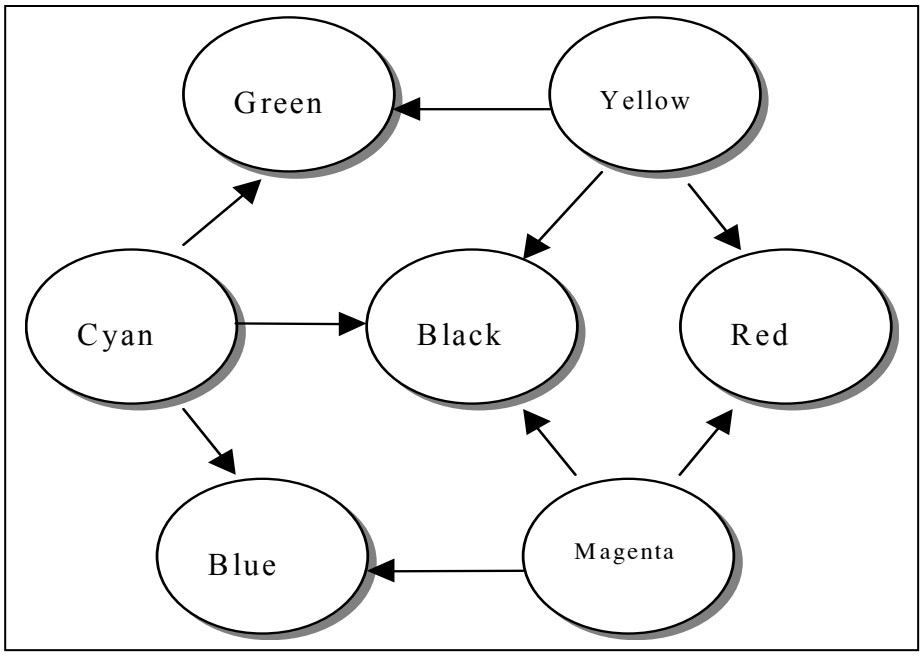
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Véc tơ đơn vị cột RGB miêu tả cho màu trắng và CMY miêu tả cho màu đen .

Sự biến đổi từ RGB thành CMY là:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

Công thức biến đổi đơn giản này có thể được xử dụng cho việc biến đổi tám màu tạo thành từ ttỏ hợp của 3 màu đỏ, xanh lá cây, xanh da trời thành tám màu tổ hợp của màu xanh tím, đỏ thẫm và màu vàng. Sự biến đổi này rất được ứng dụng rất hiệu quả trong công nghệ in phun và in Xerox.




*Hình 5.18 Các màu bù (cyan, magenta, yellow) và sự pha trộn giữa chúng.*

***Mô hình CMYK**

Một mô hình màu khác tương tự, CMYK, sử dụng thêm mầu màu đen (viết tắt là K) như màu thứ tư, được sử dụng trong quá trình in bốn màu của việc in ấn trong một số thiết bị in ấn.

Với các chỉ số kỹ thuật CMY quy định, màu đen được sử dụng để thay thế cho các vị trí có thành phần ngang bằng theo C,M,Y. Mối quan hệ sau được viết theo công thức:

$$K = \min(C, M, Y) ;$$

$$C = C - K ;$$

$$M = M - K;$$

$$Y = Y - K ;$$

## 5.8.5. Mô hình màu YIQ .

Mô hình màu YIQ là mô hình mầu được ứng dụng trong truyền hình mầu băng tần rộng tại Mỹ, và do đó nó có mối quan hệ chặt chẽ với màn hình đồ hoạ màu raster. YIQ là sự thay đổi của RGB cho khả năng truyền phát và tính tương thích với ti vi đen trắng thế hệ trước. Tín hiệu truyền sử dụng trong hệ thống NTSC ( National Television System Committee).

Thành phần Y của YIQ không phải là màu vàng nhưng là thể sáng và được xác định giống như màu gốc Y của CIE. Chỉ thành phần Y của một tín hiệu ti vi màu được thể hiện trên những ti vi đen trắng. Màu được mã hoá trong 2 thành phần còn lại là I và Q. Mô hình YIQ sử dụng hệ toạ độ Đề-Các 3 chiều với tập các thành phần nhìn thấy được biểu diễn như một khối đa diện lồi trong khối lập phương RGB.

Sự biến đổi RGB thành YIQ được xác định theo công thức sau:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0,596 & -0.275 & -0.321 \\ 0.212 & -0.532 & 0.311 \end{pmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Những đại lượng trong hàng đầu tiên phản ánh mối liên hệ quan trọng của màu xanh lá cây và màu đỏ và mối liên hệ không quan trọng của màu sáng xanh da trời. Nghịch đảo của ma trận biến đổi RGB thành YIQ được sử dụng cho sự biến đổi YIQ thành RGB.

Phương trình trên được viết với giả sử chỉ số màu RGB dựa trên cơ sở tiêu chuẩn Phosphor RGB NTSC với các giá trị (toạ độ) theo CIE là

Red Green Blue

| | | | |
|---|---|---|---|
| x | 0.67 | 0.21 | 0.14 |
| y | 0.33 | 0.71 | 0.08 |

Và cho những điểm trắng phát sáng C là: $x_w$ =0.31, $y_w$ = 0.316 và $Y_w$ = 100.0.

Các chỉ định rõ trong mô hình màu YIQ giải quyết vấn đề tiềm ẩn tạo tiền đề cho việc phát triển rộng rãi cho sự truyền phát vô tuyến băng tần rộng.

Hai màu khác nhau được hiển thị cùng nhau trên màn hình màu sẽ khác nhau, nhưng khi được biến đổi thành YIQ và được hiển thị trên màn hình đen trắng chúng lại có thể giống nhau. Vấn đề này có thể được tránh bởi vệc định rõ hai màu với hai giá trị Y khác nhau trong không gian của mô hình màu YIQ .

Mô hình màu YIQ khai thác hai thuộc tính hữu ích của hệ thống hiển thị.

Thứ nhất hệ thống này thay đổi trong thể sáng nhạy hơn là sự thay đổi trong màu sắc hoặc sự bão hoà. Khả năng của chúng ta để phân biệt không gian đa màu yếu hơn là đơn màu. Điều này đưa ra giả thiết rằng nhiều bit (đơn vị đo thông tin) của dải tần có thể được sử dụng tượng trưng cho Y hơn là được sử dụng để tượng trưng cho I và Q vì nó cung cấp độ phân giải cao hơn trong Y.

Thứ hai các đối tượng bao phủ phần rất nhỏ của vùng cảm giác màu hạn chế của chúng ta, điều này có thể được chỉ rõ tương xứng với màu một chiều hơn là màu hai chiều. Giả thiết này cho I, Q hoặc cả hai có thể có một dải tần thấp hơn Y.

Hệ thống mã NTSC của mô hình màu YIQ vào trong tín hiệu truyền băng tần rộng sử dụng các thuộc tính đó đạt giá trị lớn nhất về số lượng của thông tin được chuyển dao trong sự kết hợp dải tần: 4MHz được ấn định cho Y, 1.5 cho I ,và 0.6 cho Q .

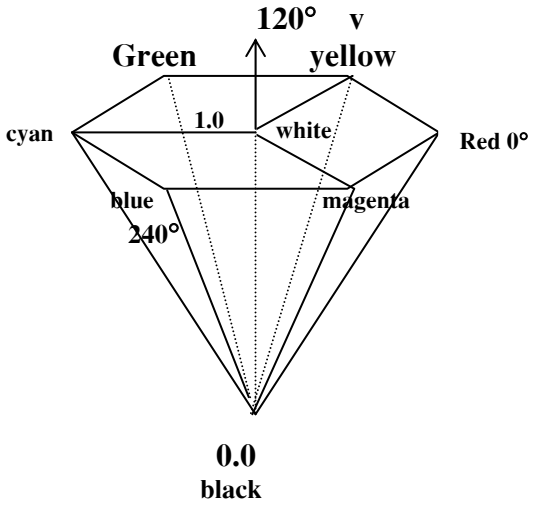
## 5.8.6. Mô hình màu HSV ( Hue, Saturation, Value )

Các mô hình màu RGB, CMY, YIQ được định hướng cho phần cứng trái ngược với mô hình màu HSVcủa Smith [SMIT78] hay còn được gọi là mô hình HSB với B là Brightness (độ sáng) được định hướng người sử dụng dựa tren cơ sở nền tảng về trực giác về tông màu, sắc mầu và sắc thái mỹ thuật.

Hệ thống toạ độ có dạng hình trụ và tập mầu thành phần của không gian bên trong mô hình màu được xác định là hình nón sáu cạnh hoặc là hình chóp sáu cạnh như trong hình 518. Đỉnh hình nón sáu cạnh khi V=1 chứa đựng mối quan hệ giữa các màu sáng. Những màu trên mặt phẳn với V=1 đều không nhận màu sáng .

Màu sắc (hue) hoặc H được đo bởi góc quanh trục đứng với màu đỏ là 0° màu xanh lá cây là 120°, màu xanh da trời là 240° xem hình 5.18. Các màu bổ xung trong hình chóp HSV ở 180° đối diện với màu khác. Giá trị của S là một dãy số truyền từ giá trị 0 trên đường trung tâm (trục V) đến 1 trên các mặt bên có hình dạng tam giác của hình nón sáu cạnh. Sự bão hoà được đo tương đối cho gam màu tương ứng với mô hình màu này, dĩ nhiên điều này là một tập hợp nhỏ của toàn bộ biểu đồ màu CIE do đó sự boã hoà 100% trong mô hình ít hơn 100% sự kích thích tinh khiết.

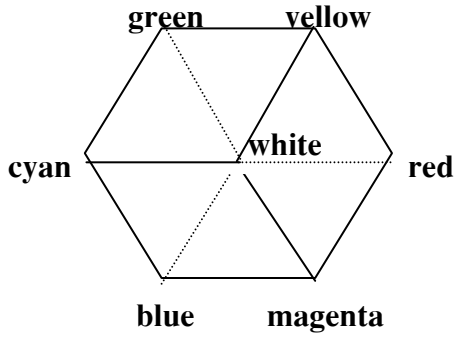
Hình nón sáu cạnh, đường cao V với các đỉnh là điểm gốc. Điểm ở đỉnh là màu đen và có giá trị toạ độ mầu V = 0, tại cái điểm này giá trị của H và S là không liên quan với nhau. Điểm có S =0 và V=1 là điểm màu trắng, những giá trị trung gian của V đối với S = 0 (trên đường thẳng qua tâm) là các màu xám. Khi S = 0 giá trị của H phụ thuộc được gọi bởi các quy ước không xác định ngược lại khi S khác 0 giá trị của H sẽ là phụ thuộc.




*Hình 5.19 Mô hình mầu HSV*

Ví dụ như đối với màu đỏ thuần xác định tại H= 0, V=1, S = 1,

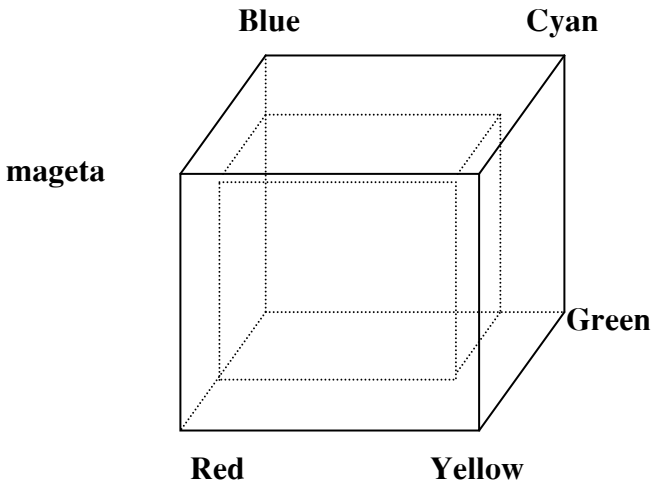
Như vậy một màu nào đó V = 1, S =1 là giống như màu thuần khiết trong mỹ thuật được sử dụng như điểm khởi đầu trong các màu pha trên. Thêm màu trắng phù hợp để giảm S (không có sự thay đổi V) sự chuyển màu được tạo ra bởi việc giữ S =1 và giảm V. Sắc thái được tạo ra bởi việc giữ cả hai S và V. Dĩ nhiên sự thay đổi H tương ứng để lựa chọn một chất màu cần thiết để bắt đầu. Chẳng hạn như H, S và V phù hợp với khái niệm màu của hội hoạ và rất chính xác.




*Hình 5.20 Hình chiếu bằng mô hình mầu HSV*

Điểm cao nhất của hình nón sáu cạnh HSV phù hợp với hình chiếu được nhìn dọc hình chéo chính của khối lập phương màu RGB. Từ màu trắng hướng đến màu đen được chỉ ra trong hình 5.20.

Khối lập phương RGB có các khối lập phương nhỏ bên trong, như được minh hoạ trong hình 5.21. Mỗi hình lập phương nhỏ khi được nhìn thấy dọc theo đường chéo chính của nó giống như hình sáu cạnh ở hình 5, trừ những hình nhỏ hơn. Mỗi mặt V bất biến trong khoảng không gian HSV tương ứng với những cái nhìn thấy của một hình lập phương nhỏ bên trong của khoảng không gian RGB.



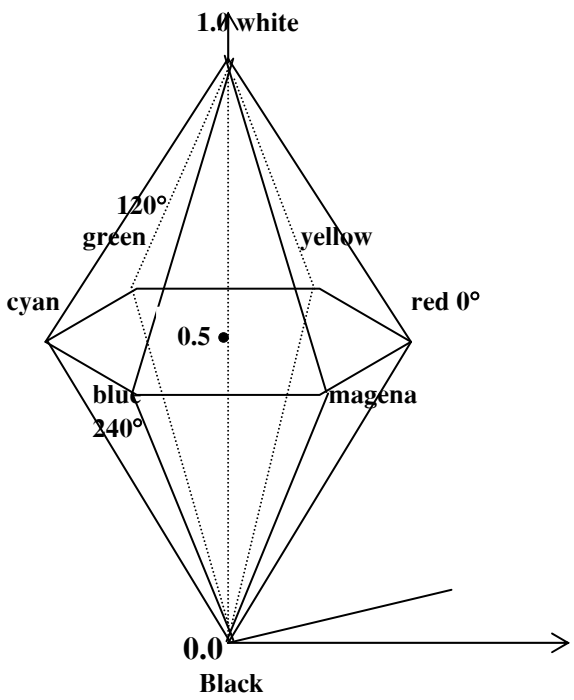

*Hình 5.21 Khối lập phương RGB và một khối lập phương nhỏ bên trong*

Đường chéo chính của không gian RGB trở thành trục V của không gian HSV. Chẳng hạn chúng ta có thể nhìn thấy bằng trực giác sự phù hợp giữa RGB và HSV thuật toán Va,Vb xác định đúng sự phù hợp bởi việc cung cấp các sự biến đổi từ một mô hình tới một mô hình khác.

## 5.8.7. Mô hình màu HLS.( Hue, Light, Saturation - màu sắc, ánh sáng, sự bão hoà )

Mô hình mầu HLS được xác định bởi tập hợp hình nón sáu cạnh đôi của không gian hình trụ như nhìn thấy trong hình 5.21. Màu sắc là góc quanh trục đứng của hình nón sáu cạnh đôi với màu đỏ tại góc 0° (một vài cuộc thảo luận của HSL cho màu xanh gia trời là điểm 0° và chúng ta đặt màu đỏ tại 0° cho sự chắc chắn với mô hình HSV).

Các màu sẽ xác định theo thứ tự giống như trong biểu đồ CIE khi ranh giới của nó bị xoay ngược chiều kim đồng hồ: Màu đỏ, màu vàng, màu xanh lá cây, màu xanh tím, màu xanh da trời và đỏ thẫm. Điều này cũng giống như thứ tự xắp xếp trong mô hình hình nón sáu cạnh đơn HSV.




*Hình 5.22 Mô hình màu hình nón sáu cạnh đôi HLS.*

Tóm lại chúng ta có thể coi mô hình HLS như một sự biến dạng của mô hình HSV mà trong đó mô hình này màu trắng được kéo hướng lên hình nón sáu cạnh phía trên từ mặt V = 1. Như với mô hình hình nón sáu cạnh đơn, phần bổ xung của một màu sắc được đặt ở vị trí 180° hơn là xung quanh hình nón sáu cạnh đôi, sự bão hoà được đo xung quanh trục trục đứng, từ 0 trên trục tới 1 trên bề mặt.

Độ sáng (lighness) = 0 cho màu đen (tại điểm mút thập nhất của hình nón sáu cạnh đôi) và bằng 1 cho màu trắng (tại đầu mút cao nhất). Thêm nữa thuật ngữ sắc màu, độ sáng và sự bão hoà trong mô hình này cũng tương tự như như các thuật ngữ được giới thiệu ở các mục trên nhưng xác định một cách không chính xác.

Các thủ tục trong VIa và VIb và thực hiện sự biến đổi giữa HLS và RGB. Chúng được sự sửa đổi từ các quy định bởi Metrick để dời tới H không xác định khi S=0 đến khi H=0 cho màu đỏ hơn là cho màu xanh

Mô hình HLS cũng giống mô hình HSV là dễ sử dụng. Tất cả các màu xám có S=0 nhưng các màu bão hoà lớn nhất là tại S = 1, L = 0.5 nếu thiết bị đo điện kế được sử dụng để xác định tham số mô hình màu, thực tế L phải là 0.5 để đạt tới màu mạnh nhất đó là một bất lợi của mô hình HSV trong trường hợp S =1 và V = 1 đạt được giống nhau.

Tuy nhiên tương tự như mô hình HSV các màu của mảng L =0.5 tất cả chúng đều giống nhau là không nhận màu sáng. Vì thế hai màu khác nhau nhận độ sáng như nhau sẽ có giá trị của L khác nhau. Hơn nữa hoặc mô hình HLS, hoặc một trong các mô hình khác được thảo luận ở phần này có cảm giác đơn điệu.

Hệ thống màu Tektronix TekHVC (Hue, Value, Chroma) phát triển gần đây là một sự sửa đổi của các mô hình CIE LUV cùng loại đã cung cấp một không gian màu được đo và nhận biết được khoảng cách giữa các màu là xấp xỉ như nhau. Điều này là một lợi thế quan trọng của hai mô hình CIE LUV và TekHVC trong đó các chi tiết của sự biến đổi từ mô hình CIE sang mô hình TekHVC đã được tách ra.

Tuy nhiên chúng ta hiểu từ sự chuyển đổi đó từ CIE XYZ sang CIE LUV là không phức tạp. Như vậy chúng ta mong rằng các không gian màu cùng loại sẽ được nhận biết và sử dụng rộng rãi trong tương lai.

## 5.8.8 Các bộ lệnh chuyển đổi mô hình mầu

Matlab sử dụng những hệ màu khác nhau dùng để vẽ bề mặt lưới. Bảng mầu là ma trận m x 3 với mỗi hàm gồm 3 giá trị để xác định các mầu sắc thành phần theo thứ tự đỏ, xanh lục, xanh dương (R,G,B). Màu sắc của bề mặt bề mặt được ấn định bởi chỉ số của bảng mầu, chỉ số này thường được tính tương quan với giá trị từ min đến max của bề mặt.

Lệnh colormap được sử dụng để ấn định màu sắc cho bề mặt lưới.

### * colormap

**colormap (C )** - Xét C thành giá trị bảng mầu dùng hiện thời, ma trận C có thể là một trong những bản mầu chuẩn của Matlab hoặc do người sử dụng tự định ngũa ra colormap

**colormap** - Lệnh dùng để trả giá trị bảng mô hình hiện tại đang dùng vào ma trận m x 3.

**colorbar** - Lệnh dùng để vẽ ra dải mầu thẳng đứng trong màn hình đồ hoạ hiện thời.

**colorbar ( 'horiz' )** - Vẽ ra dải mầu nằm ngay trong màn hình đồ hoạ thời. Dưới đây là liệt kê của 11 bản đồ màu của Matlab.

- gray (m) đưa ra dải mầu xám tuyến tính trên m mức độ.

- hsv (m) đưa ra dải mầu sáng bão hoà chạy từ giá trị đỏ qua giá trị xanh đến giá trị đỏ. Hệ mầu hsv được xác định trên ba chỉ số hulg, saturation, volume.

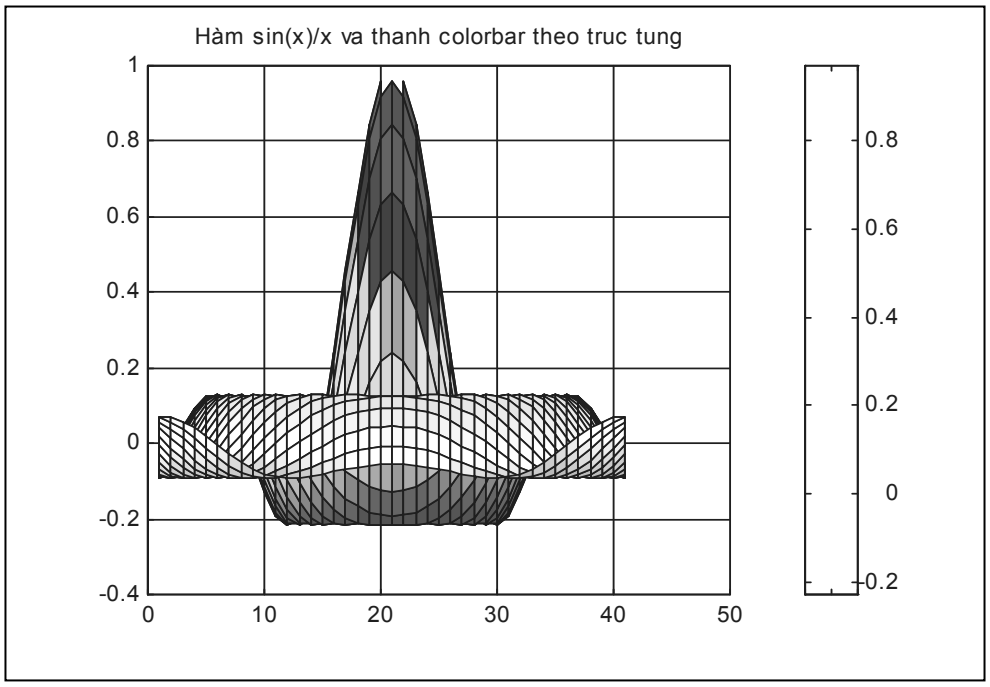
- hot (m) đưa ra gam mầu nóng là hỗn hợp giữa màu đen lẫn đỏ xen giữa vàng lẫn trắng.

- cool (m) đưa ra gam màu lạnh hỗn hợp giữa cyan (xanh) và màu magenta khác

- bone (m) đưa ra giá trị của dải màu gam màu phớt

xanh.

- copper (m) đưa ra dải mầu đồng
- pink (m) đưa dải biến đổi theo màu hồng
- flag (m) đưa ra mầu theo mầu cờ UK và US (đỏ trắng hoặc xanh cùng màu đen liên tiếp thành chuỗi.
- prism (m) đưa ra chuỗi màu gồm 6 máuau: đỏ, da cam, vàng, xanh lục, xanh dương và tím.
- jet (m) đưa ra bảng mầu tương tự hệ hsv với giá trị đi từ đỏ đến xanh
- white (m) đưa ra gam mầu trắng cho hệ thống




*Hình 5.23 Dải mầu của mặt lưới và thang bậc mầu của thanh colorbar*

## 5.8.9 Thao tác với mầu sắc.

**rgb2hsv ( C )** - Chuyển giá trị của ma trận (m x 3)C từ hệ mầu rgb sang hệ mầu hsv

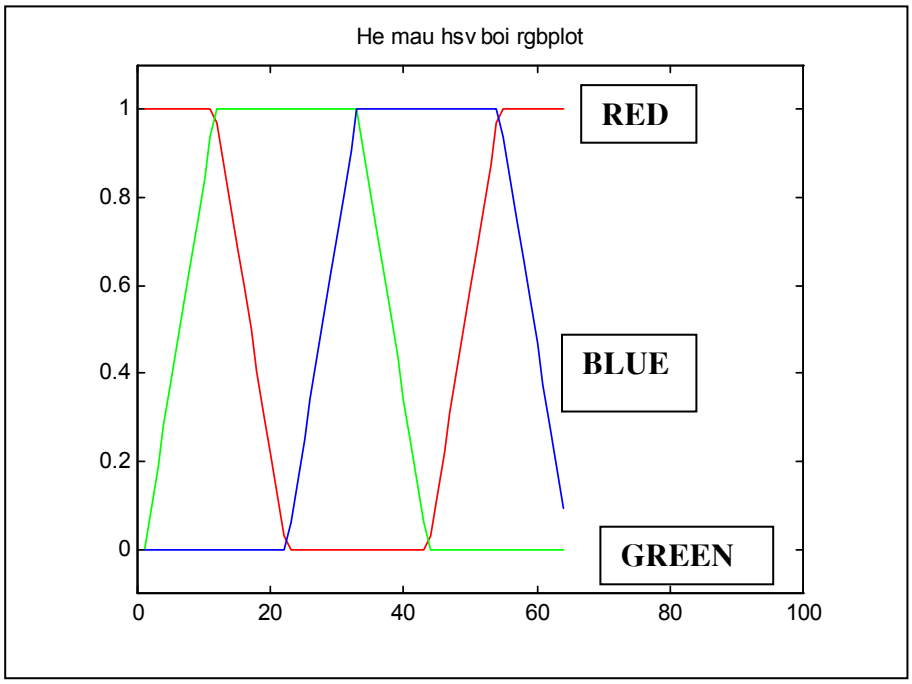
**hsv2rgb ( C )** - Chuyển ma trận C (m x 3) từ hệ mầu hsv sang hệ màu rgb.

| | |
|---|---|
| **rgbplot ( C )** | - Vẽ đồ thị của bảng mầu rgb xác định bởi ma trận C với các cột tương ứng cho 3 mầu đỏ, xanh lục, xanh dương. |
| **caxis ( v )** | - Đưa ra khoảng xác định giới hạn của bảng mầu với v = [$v_{min}$ $v_{max}$]: $v_{min}$, $v_{max}$ là cá thành phần giới hạn dưới và trên của dải mầu . |
| **caxis** | - Đưa ra khoảng xác định giá trị của bảng mầu hiện thời |
| **caxis ('auto')** | - Xét lại dải mầu cho hệ thống với giá trị được lấy từ hệ thống Matlab. |
| **spinmap ( t, s )** | - Quay bảng màu trong thời gian t giây sử dụng bước nhảy s. Nếu s không được định nghĩa thì giá trị mặc định = 2. Nếu t không được xác định thì thời gian mặc định là 3s. |
| **spinmap( inf )** | - Quay bảng màu không xác định thời gian |
| **brighten ( s )** | - Sử dụng bảng mầu sáng nếu s ∈ (-1, 0) và bảng mầu thẫm nếu s ∈ (-1,0). |
| **Nt=brighten(c,s)** | - Đưa ra bảng mầu đậm nhật của ma trận C mà không vẽ lại màn hình. |
| **contrast ( c, m )** | - Đưa ra bảng mầu có chiều dài m từ bảng mầu ma trận C. Nếu m không xác định thì chiều dài của bảng sẽ lấy chiều dài của ma trận. |
| **whitebg** | - Chuyển mầu nền của màn đồ hoạ từ đen - trắng hoặc ngược lại. |
| **whitebg ( str )** | - Xét mầu nền theo chuỗi str, hay vevtor hệ mầu rgb. |
| **graymon** | - Xét biến số cho màn hình đen trắng |

**Ví dụ 5.10:**

Cho ra đồ thị của hệ mầu hsv trên cơ sở lệnh rgbplot. Ba đường đồ thị chỉ định cho 3 mầu trên cơ sở tỉ lệ tham gia thành phần của mỗi mầu.

```
>> rgbplot ( hsv);
>>title ( 'Hệ màu hsv bởi rgbplot' );
>>axis ( [0  100  -0.1  1.1] );
```




***Hình 5.24. Hình vẽ quan hệ bảng mầu hsv với lệnh rgbplot***

# BÀI TẬP ỨNG DỤNG PHẦN 1

## Bài 1

Xây dựng hàm bậc nhất y = ax + b với các tham số a,b được đưa vào từ bàn phím. Truy xuất kết quả lên màn hình đồ hoạ

***Bài giải:***

```
%A.1 Vẽ theo phương trình hàm bậc nhất
% y = ax + b

clg
a=0;b=0;c=0;d=0;e=0;
disp('Khong gian hai chieu')
disp('Ve do thi ham bac nhat y = ax + b');
a=input('Vao he so bac nhat ; a = ');
b=input('Vao he so tu do : b = ');
x=-5:0.1:5;
y=a*x+b;
hold on
  plot(x,y,'m-')
  plot(y,zeros(x),'c-')
  plot(zeros(x),x,'c-')
  text(-1,-1.5,'O')
  text(-0.05,max(y),'^')
  text(max(x),0,'>')
  title('Ham bac nhat')
hold off
clc
```

## Bài 2

Xây dựng hàm bậc hai $y = ax^2 + bx + c$ với các tham số a, b, c được đưa vào từ bàn phím. Truy xuất kết quả lên màn hình đồ hoạ

*Bài giải:*

```
%B.1 Vẽ theo phương trình hàm bậc 2
%  y = ax^2 + bx + c

disp('Next : Ham so bac hai')
pause
clg
disp('Ve do thi ham bac hai y = ax^2 + bx +c');
a=input('Vao he so bac hai ; a = ');
b=input('Vao he so bac nhat : b = ');
c=input('Vao he so tu do c = ');
x=-3:0.1:3;
y=a*(x.^2)+b*x+c;
hold on
  plot(x,y,'m-')
  plot(y,zeros(x),'c-')
  plot(zeros(x),x,'c-')
  text(-1,-1.5,'O')
  text(-0.05,max(y),'^')
  text(max(x),0,'>')
  title('Ham bac hai')
hold off
clc
```

## Bài 3

Xây dựng hàm bậc hai  y = 1/( ax + b ) với các tham số a, b được đưa vào từ bàn phím. Truy xuất  kết quả lên màn hình đồ hoạ

```
% Vẽ theo phương trình hàm Ham so
%   y = 1/( ax + b )

disp('Next : Ham so y=1/(ax+b)')
pause
clg
disp('Ve do thi ham  y =1/(ax + b)');
a=input('Vao he so bac nhat ; a = ');
b=input('Vao he so tu do : b = ');
x=-5:0.1:5;
y=1./(a*x+b);
```

```
hold on
  plot(x,y,'m-')
  plot(y,zeros(x),'c-')
  plot(zeros(x),x,'c-')
  text(-1,-1.5,'O')
  text(-0.05,max(y),'^')
  text(max(x),0,'>')
  title('Ham y=1/(ax+b)')
hold off
clc
```

## Bài 4

Xây dựng hàm r = a* phi với các tham số a được đưa vào từ bàn phím. Truy xuất kết quả lên màn hình đồ hoạ với hệ toạ độ dùng là hệ toạ độ cực

*Bài giải:*

```
% Ví dụ về hệ toạ độ cực
disp('Next : He toa do cuc')
pause
clg
% D.1 Vẽ đường xoan ốc
%  r = a* phi
disp('Ve duong xoan oc : r = a*tt')
pause
clg
a=input('Vao he so a = ');
tt=0:0.1:8*pi;
r=a*tt;
axis('equal','off')
polar(tt,r)
title('Duong xoan oc')
disp('Ve nhieu lan')
pause
axis('equal','off')
for m=1:8
  hold on
  r1=r*m;
  polar(tt,r1)
  hold off
 end
```

## Bài 5

Xây dựng hàm r = a*cos ( phi ) + b với các tham số a,b được đưa vào từ bàn phím. Truy xuất kết quả lên màn hình đồ hoạ với hệ toạ độ dùng là hệ toạ độ cực

*Bài giải:*

```
%D.2 Đường ốc sên r = a*cos ( phi ) + b
disp('Next :duong oc sen r=a*cos(tt)+b')
pause
clg
a=input('Vao he so a = ');
b=input('Vao he so b = ');
tt=0:0.1:8*pi;
r=a*cos(tt)+b;
axis('equal','off')
polar(tt,r)
title('Duong oc sen')
disp('Ve nhieu lan')
pause
for m=1:8
  hold on
  r1=r*m;
  polar(tt,r1)
  hold off
 end
```

## Bài 6

Xây dựng hàm Astroit với các tham số a được đưa vào từ bàn phím. Truy xuất kết quả lên màn hình đồ hoạ với hệ toạ độ dùng là hệ toạ độ cực

*Bài giải:*

```
%D.3 Đường astroit
disp('Next :duong Astroit ')
pause
clg
a=input('Vao he so a = ');
tt=0:0.1:8*pi;
r=a*sqrt(abs(1-sin(3*tt)/4));
polar(tt,r)
title('Duong Astroit')
disp('Ve nhieu lan')
pause
```

```
for m=1:8
  hold on
  r1=r*m;
  polar(tt,r1)
  hold off
 end
```

## Bài 7

Xây dựng phương trình đường Lemniscat Becnulli với các tham số a được đưa vào từ bàn phím. Truy xuất kết quả lên màn hình đồ hoạ với hệ toạ độ dùng là hệ toạ độ cực

*Bài giải:*

```
% D.4 Đường Lemniscat Becnulli

disp('Next :duong Lemniscat Becnulli')
pause
clg
a=input('Vao he so a = ');
tt=0:0.1:8*pi;
r=a*sqrt(abs(2*cos(2*tt)));
axis('equal','off')
polar(tt,r)
title('Duong xoan oc')
disp('Ve nhieu lan')
pause
for m=1:8
  hold on
  r1=r*m;
  polar(tt,r1)
  hold off
 end
```

## Bài 8

Dùng hàm bucky để xây dựng hình giả 3chiều. Truy xuất kết quả lên màn hình đồ hoạ

*Bài giải:*

```
%Không gian 3D
```

```
disp('Khong gian ba chieu ')
pause
clg
```

%e.1 Vẽ hình quả bóng

```
disp('Ve qua bong da')
[B,V]=bucky;
H=sparse(60,60);
k=31:60;
H(k,k)=B(k,k);
x=V(:,1);
y=V(:,2);
gplot(H,V,'m-')
axis('equal','off');
hold on
gplot(B-H,V,'c-')
hold off
```

## Bài 9

Vẽ hàm đồ thị trong không gian 3 chiều. Dùng plot3()

*Bài giải*

%e.2 Vẽ đường có hình ảnh không gian

```
disp('Ve duong co hinh anh khong gian')
pause
clg
t=0:pi/50:8*pi;
plot3(sin(t),cos(t),t);
```

## Bài 10

Vẽ một số bề mặt ví dụ trong không gian 3 chiều với các tham số tuỳ chọn. Mặt parabolloit, mặt trụ.

*Bài giải*

%e.3 Vẽ mặt không gian 3D

```
disp('Next: Ve mat khong gian ba chieu')
disp('Ve Parabolloit')
pause
clg
```

```
t=-5:0.1:5;
[x,y]=meshdom(t,t);
z=x.^2+y.^2;
mesh(z)
title('Paraboloit')
disp('Next: Mat tru sinh boi y=x^2')
pause
clg
z=sqrt(x.^4+y.^2);
mesh(z)
title('Mat tru')
pause
```

## Bài 11

Xây dựng menu trong môi trường Matlab và thực hiện một số các thao tác xây dựng các hàm đồ hoạ đơn giản. Bao gồm: Vẽ một hình cầu, phương trình đường sin(x)^2, sin(x^2)*exp(-x), sin(1/x)^2/x và tich phân xác định của hàm bất kỳ.

*Bài giải*

```
function Thuctap(action);
%Thuctap    Chuong trinh nay ve mot do hoa bao gom chuc %
            nang ve mot so ham va tich phan
%Nhung viec can lam:
% Nut1 :Sphere (Hinh cau)
% Nut2,3,4 : Phuong trinh cac ham co ban
% Nut 5 : Tich phan xac dinh
if nargin<1,
    action='batdau';
end;
if strcmp(action,'batdau'),
   figNumber=figure( ...
       'Name','Bai tap', ...
       'NumberTitle','on', ...
     'Visible','off');

% Nhung thong tin de tao cac phim chuc nang
    labelColor=[0.8 0.8 0.8];
    yInitPos=0.9;
    xPos=0.8;
    btnLen=0.12;
    btnWid=0.10;
```

```
% Khoang cach giua nut va nhan cua lenh tiep theo
    kc=0.03;

% Khung nen cho cac phim chuc nang :The CONSOLE frame
    frmBorder=0.01;
    yPos=0.01;
    frmPos=[xPos+0.02 yPos-frmBorder btnLen+4*frmBorder
0.9+11*frmBorder];
    uicontrol( ...
        'Style','frame', ...
        'Units','normalized', ...
        'Position',frmPos, ...
     'BackgroundColor',[1 0 0]);

% Nut 1 hien thi lai do thi hinh cau
    btnNumber=1;
    yPos=0.90-(btnNumber-1)*(btnWid+kc);
    labelStr='Nut1';
    callbackStr='Thuctap(''Sphere'')';

% Cac thong tin chung ve kieu nut kich hoat.
    btnPos=[xPos+4*frmBorder yPos-kc btnLen btnWid];
    uicontrol( ...
        'Style','pushbutton', ...
        'Units','normalized', ...
       'Position',btnPos, ...
       'String',labelStr, ...
     'Callback',callbackStr);

% Nut 2 hien thi ham sin(x)^2
    btnNumber=2;
    yPos=0.90-(btnNumber-1)*(btnWid+kc);
    labelStr='Nut2';
    callbackStr='hinhcau1(''nut2'')';

%day la nap lai ham thuctap
% Cac thong tin chung ve kieu nut kich hoat.
    btnPos=[xPos+4*frmBorder yPos-kc btnLen btnWid];
    uicontrol( ...
        'Style','pushbutton', ...
        'Units','normalized', ...
       'Position',btnPos, ...
       'String',labelStr, ...
     'Callback',callbackStr);
```

```
% Nut 3 hien thi ham sin(x^2)*exp(-x)
    btnNumber=3;
    yPos=0.90-(btnNumber-1)*(btnWid+kc);
    labelStr='Nut3';
    callbackStr='hinhcau1(''nut3'')';

% Cac thong tin chung ve kieu nut kich hoat.
    btnPos=[xPos+4*frmBorder yPos-kc btnLen btnWid];
    uicontrol( ...
        'Style','pushbutton', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
        'Callback',callbackStr);

% Nut 4 hien thi ham sin(1/x)^2 / x
    btnNumber=4;
    yPos=0.90-(btnNumber-1)*(btnWid+kc);
    labelStr='Nut4';
    callbackStr='hinhcau1(''nut4'')';

% Cac thong tin chung ve kieu nut kich hoat.
    btnPos=[xPos+4*frmBorder yPos-kc btnLen btnWid];
    uicontrol( ...
        'Style','pushbutton', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
        'Callback',callbackStr);

% Nut 5 Tinh va hien thi vung tich phan cua mot ham
    btnNumber=5;
    yPos=0.90-(btnNumber-1)*(btnWid+kc);
    labelStr='Nut5';
    callbackStr='hinhcau1(''nut5'')';

% Cac thong tin chung ve kieu nut kich hoat.
    btnPos=[xPos+4*frmBorder yPos-kc btnLen btnWid];
    uicontrol( ...
        'Style','push', ...
        'Units','normalized', ...
        'Position',btnPos, ...
        'String',labelStr, ...
```

```
        'Callback',callbackStr);

% Nut kich hoat phan thong tin giai thich
labelStr='Info';
    callbackStr='hinhcau1(''info'')';

%day la nap lai ham thuctap
    uicontrol( ...
        'Style','push', ...
        'Units','normalized', ...
        'Position',[xPos+4*frmBorder 0.22 btnLen btnWid],
...
        'String',labelStr, ...
        'Callback',callbackStr);

% Nut xoa man hinh do hoa:The CLOSE button.
    labelStr='Close';
    callbackStr='close(gcf)';
    uicontrol( ...
        'Style','push', ...
        'Units','normalized', ...
        'Position',[xPos+4*frmBorder 0.05 btnLen btnWid],
...
        'String',labelStr, ...
        'Callback',callbackStr);

% vE HINH CAU
     clc reset;
      set(gca,'XTick',[],'YTick',[],'ZTick',[]);

% Reset the arrow and the nextplot information for this
window.
    set(figNumber, ...
     'Nextplot','new', ...
     'Visible','on');

elseif strcmp(action,'info');
    ttlStr=get(gcf,'Name');
     hlpStr=['      BAI TAP CHUYEN DE MATLAB      '
          '                                        '
          '     Sinh Vien: Nguyen thi Nhung        '
          '                                        '
          ' File name: thuctap.m Ver 1.0          '];
     helpfun(ttlStr,hlpStr);
```

```
%end;
% end cua if strcmp(action, ...
elseif strcmp(action,'nut2')

% Ve hinh NUT 2
      x=0:0.05:5;
      y=sin(x.^2);
       plot(x,y);
        title('Ham sinx^2');
   elseif strcmp(action,'nut3')

% ve hinh NUT 3
     clc reset;
     x = 0:0.1:4;
     y = sin(x.^2).*exp(-x);
     stem(x,y)
     title('ham y=sin(x^2)*e^-x');
elseif strcmp(action,'nut4')

%ve hinh  NUT 4

     x=logspace(-2,0,500);
     plot(x,((sin(1./x)).^2)./x);
     set(gca,'XScale','log','YScale','linear');
     title('Ham y=(sin(1/x)^2)/x');
elseif strcmp(action,'nut5')

%ve hinh NUT 5
  fplot('humps',[0,2]), hold on
  patch([0.5 0.5:0.02:1 1 0.5],
        [0 humps(0.5:0.02:1) 0 0],'r');
  hold off
  title('Tinh tich phan xac dinh.'),grid
elseif strcmp(action,'Nut 1')

% Ve hinh cau
    clc reset;
    n=25;
    [x,y,z]=sphere(n)
    surf(x,y,z),...
    title('3-DIMENSION DO THI HINH CAU')
    clc reset;
    set(gca,'XTick',[],'YTick',[],'ZTick',[]);
end;
```

## Bài 12

Ví dụ về 2 hình cầu lồng nhau cà các phương pháp tô mầu ( rendering ) trong Matlab.

*Bài giải*

```
[xx yy zz] = sphere;
s = surf(xx,yy,zz);
set(s, 'EdgeColor', 'r', 'FaceColor', 'none');
axis off;
set(gca, 'DataAspectRatio' , [1  1 1]);
light;
set(s, 'LineWidth', 6)
hold on;
[xx yy zz] = sphere;
s2=surf(xx/2, yy/2, zz/2);
set(s2, 'CData', rand(21), 'FaceColor', 'interp')
colormap(cool(100))
lighting phong;
set(gca, 'CameraViewAngle',  7);
set(gcf, 'color', [1 1 1]);
```

## Bài 13

Xây dựng và vẽ hình đường B-Spline trong không gian 2D và 3D từ các điểm kiểm soát được vào từ bàn phím hay các file dữ liệu. Trên cơ sở đường cong phát triển thành mặt B-spline.

*Bài giải*

```
 s=2;s1=0;
while s>1
s1=s1+1;
s=input('Neu tiep tuc thi s<1=');
n=input('n=');
```

```
k=input('k=');
for i=1:(n+4)

if i<(k+1)
u(i)=0;
elseif i>n
u(i)=n-k+1;
else
u(i)=i-k;
end
end
x=input('Nhap vao n toa do Px=');
y=input('Nhap vao n toa do Py=');
z=input('Nhap vao n toa do Pz=');

m=input('vao khoang can ve(1,2..n)=');

for i=1:(n+3)
if u(i)< u(i+1)
if u(i)==m-1
N(i,1)=1;
else
N(i,1)=0;
end
else
N(i,1)=0;
end
end
for i=1:(n+3)
t=N(i,1);
end
t=0;
for U=(m-1):0.125:m
%U=0.125*i;
t=t+1;
for j=2:4
if j==2
for l=1:(n+k-2)
if u(l+j-1)==u(l)
if u(l+j)==u(l+1)
N(l,j)=0;
else
N(l,j)=(u(l+j)-U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
```

```
else
if u(l+j)==u(l+1)
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l));
else
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l)) +(u(l+j)-
U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
end
end
end

if j==3

for l=1:(n+k-3)
if u(l+j-1)==u(l)
if u(l+j)==u(l+1)
N(l,j)=0;
else
N(l,j)=(u(l+j)-U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
else
if u(l+j)==u(l+1)
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l));
else
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l)) +(u(l+j)-
U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
end
end
end

if j==4
X(t)=0;Y(t)=0;Z(t)=0;
for l=1:(n+k-4)
if u(l+j-1)==u(l)
if u(l+j)==u(l+1)
N(l,j)=0;
else
N(l,j)=(u(l+j)-U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
else
if u(l+j)==u(l+1)
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l));
else
```

```
N(l,j)=(U-u(j))*N(l,j-1)/(u(l+j-1)-u(l)) +(u(l+j)-
U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
end
end

end
end
for l=1:n
X(t)=X(t)+x(l)*N(l,k);
Y(t)=Y(t)+y(l)*N(l,k);
Z(t)=Z(t)+z(l)*N(l,k);
end
end %U
%hold on;
if s1==1
subplot(2,1,1);
plot3(X,Y,Z);
line(x,y,z);
hold on
subplot(2,1,2);
hold on
plot(X,Y,'M');
line(x,y);
hold on
 end
if s1==2
subplot(2,1,1);
plot3(X,Y,Z);
line(x,y,z);
hold on
subplot(2,1,2);

plot(X,Y,'G');
line(x,y);
hold on
 end
if s1==3
subplot(2,1,1);
plot3(X,Y,Z);
line(x,y,z);
hold on
subplot(2,1,2);
```

```
plot(X,Y,'R');
line(x,y);
hold on
 end
end
elseif strcmp(action,'Plane');

n=input('n=');
m=input('m=');
k=input('Vao bac k=');
h=input('vao bac h=');
q='y';
while q=='y'
q=input('Neu tiep tuc thi danh y nguoc lai la n =');
for i=1:(n+k)

if i<(k+1)
u(i)=0;
elseif i>n
u(i)=n-k+1;
else
u(i)=i-k;
end
end
for i=1:(m+h)

if i<(h+1)
w(i)=0;
elseif i>m
w(i)=m-h+1;
else
w(i)=i-h;
end
end
%x=input('Nhap vao n.m toa do Px=');
%y=input('Nhap vao n.m toa do Py=');
%z=input('Nhap vao n.m toa do Pz=');
x(1,1)=-3;x(1,2)=-3;x(1,3)=-3;x(1,4)=-3;
x(2,1)=-1;x(2,2)=-1;x(2,3)=-1;x(2,4)=-1;
x(3,1)=1;x(3,2)=1;x(3,3)=1;x(3,4)=1;
x(4,1)=3;x(4,2)=3;x(4,3)=3;x(4,4)=3;

y(1,1)=0;y(1,2)=3;y(1,3)=3;y(1,4)=0;
y(2,1)=3;y(2,2)=5;y(2,3)=5;y(2,4)=3;
```

```
y(3,1)=3;y(3,2)=5;y(3,3)=5;y(3,4)=3;
y(4,1)=5;y(4,2)=5;y(4,3)=5;y(4,4)=5;

z(1,1)=5;z(1,2)=3;z(1,3)=-3;z(1,4)=-5;
z(2,1)=5;z(2,2)=3;z(2,3)=-3;z(2,4)=-5;
z(3,1)=5;z(3,2)=3;z(3,3)=-3;z(3,4)=-5;
z(4,1)=5;z(4,2)=3;z(4,3)=-3;z(4,4)=-5;
v=input('vao khoang can ve cua u(1,2..n)=');
g=input('vao khoang can ve cua w(1,2..n)=');
for i=1:(n+k-1)
if u(i)< u(i+1)
if u(i)==v-1
N(i,1)=1;
else
N(i,1)=0;
end
else
N(i,1)=0;
end
end
for i=1:(n+k-1)
t=N(i,1);
end

for i=1:(m+h-1)
if w(i)< w(i+1)
if w(i)==g-1
M(i,1)=1;
else
M(i,1)=0;
end
else
M(i,1)=0;
end
end
for i=1:(m+h-1)
t1=M(i,1);
end

X1=[];Y1=[];Z1=[];
for U=(v-1):0.1:(v-0.1)
t=0;
for W=(g-1):0.1:(g-0.1)
t=t+1;
```

```
for i=2:h
if i==2

for l=1:(m+h-2)
if w(l+i-1)==w(l)
if w(l+i)==w(l+1)
M(l,i)=0;
else
M(l,i)=(w(l+i)-W)*M(l+1,i-1)/(w(l+i)-w(l+1));
end
else
if w(l+i)==w(l+1)
M(l,i)=(W-w(l))*M(l,i-1)/(w(l+i-1)-w(l));
else
M(l,i)=(W-w(l))*M(l,i-1)/(w(l+i-1)-w(l)) +(w(l+i)-
W)*M(l+1,i-1)/(w(l+i)-w(l+1));
end
end
end
end
if i==3

for l=1:(m+h-3)
if w(l+i-1)==w(l)
if w(l+i)==w(l+1)
M(l,i)=0;
else
M(l,i)=(w(l+i)-W)*M(l+1,i-1)/(w(l+i)-w(l+1));
end
else
if w(l+i)==w(l+1)
M(l,i)=(W-w(l))*M(l,i-1)/(w(l+i-1)-w(l));
else
M(l,i)=(W-w(l))*M(l,i-1)/(w(l+i-1)-w(l)) +(w(l+i)-
W)*M(l+1,i-1)/(w(l+i)-w(l+1));
end
end
end
end
end
for j=2:k
if j==2
for l=1:(n+k-2)
```

```
if u(l+j-1)==u(l)
if u(l+j)==u(l+1)
N(l,j)=0;
else
N(l,j)=(u(l+j)-U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
else
if u(l+j)==u(l+1)
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l));
else
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l)) +(u(l+j)-
U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
end
end
end
if j==3
for l=1:(n+k-3)
if u(l+j-1)==u(l)
if u(l+j)==u(l+1)
N(l,j)=0;
else
N(l,j)=(u(l+j)-U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
else
if u(l+j)==u(l+1)
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l));
else
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l)) +(u(l+j)-
U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
end
end
end

if j==4
for l=1:(n+k-4)
if u(l+j-1)==u(l)
if u(l+j)==u(l+1)
N(l,j)=0;
else
N(l,j)=(u(l+j)-U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
else
```

```
if u(l+j)==u(l+1)
N(l,j)=(U-u(l))*N(l,j-1)/(u(l+j-1)-u(l));
else
N(l,j)=(U-u(j))*N(l,j-1)/(u(l+j-1)-u(l)) +(u(l+j)-
U)*N(l+1,j-1)/(u(l+j)-u(l+1));
end
end
end
end
end  % kt for j=2:k

X1(t)=0;Y1(t)=0;Z1(t)=0;
for l=1:n
for i=1:m
X1(t)=X1(t)+x(l,i).*N(l,k).*M(i,h);
Y1(t)=Y1(t)+y(l,i).*N(l,k).*M(i,h);
Z1(t)=Z1(t)+z(l,i).*N(l,k).*M(i,h);
end
end
plot3(X1,Y1,Z1);
hold on
view([3 3 6])
end
end

view([3 3 3])
end
```

# PHẦN 2

# ỨNG DỤNG VỀ XỬ LÝ TÍN HIỆU SỐ

## 1. TÍN HIỆU VÀ XỬ LÝ TÍN HIỆU

Khái niệm về tín hiệu và một khái niệm rộng. Nó được định nghĩa ở Websur (1998) như là một "sự đếm được của lượng vật lý hay xung (như một hiệu điện thế, dòng, hoặc từ thông ) bởi một bản tin hoặc thông tin mà có thể truyền đi được". Ví dụ như thông tin mong muốn có thể là nhiệt độ và tín hiệu điện thế tỉ lệ với nhiệt độ này.

Nhiều sách cho rằng tín hiệu thay đổi như một hàm của thời gian. Một số tín hiệu là liên tục ; nhiệt độ không khí, sóng biển tại một điểm... Một số tín hiệu khác là rời rạc, ví dụ như các thư truyền đi (gửi đi) theo mã Morse. Các tín hiệu có thể rời rạc vì chúng nhận được bởi sự lấy mẫu thông tin không liên tục, ví dụ như nhiệt độ của khí quyển và áp suất được truyền đi theo một khoảng thời gian nhất định bởi vô tuyến. ***Máy tính có thể xử lý tín hiệu rời rạc thôi.***

Để xử lý bằng máy tính, hầu hết các tín hiệu có thể thể hiện theo một chuỗi các số 1,0 . Tín hiệu được sinh ra bởi cảm biến (sensor), ví dụ như nhiệt điện trở, hay tốc độ kế sinh ra. Chuỗi số 1 chiều thực khi được lấy mẫu tại các khoảng không đổi. Việc số hoá các hình ảnh sinh ra chuỗi số 2 chiều. Như các chuỗi số và các ma trận được thể hiện trên các đường khác nhau. Chúng có thể được lọc khỏi nhiễu, có thể được modul hoá và có thể được xử lý để làm rõ hình ảnh, hoặc nén trong một khoảng động. Khi mà các thao tác có thể được thực hiện ở thời gian hoặc tần số chủ đạo, khi chọn tần số chủ đạo và thuật toán đúng thì thông thường có kết quả là tăng hiệu ứng. Matlab dùng để giải thuật toán nhanh và chọn tần số chủ đạo.

Phần này ta xử dụng Toolbox xử lý tín hiệu (**Signal Processing**). ***Nó gồm hơn 70 hàm số khác nhau để*** phân tích số tín hiệu số và xây dựng bộ lọc với những đặc

tính cho trước. Ở chương này ta chỉ xét xét một số ví dụ sử dụng những hàm chính hay dùng, còn những hàm khác thì bạn đọc có thể tự tham khảo trong sách hướng dẫn sử dụng.

## 2. HÀM LỌC.

Một trong những hàm số hay dùng để xử lý chuỗi là hàm ***filter*** (hàm lọc). Lọc số có ở mọi nơi trong quá trình xử lý tín hiệu. Trường hợp đơn giản khi 1 tín hiệu x có thêm nhiễu (noise). Chúng ta có thể loại bỏ chúng bằng cách sử dụng lọc.

Hãy xem xét trường hợp tín hiệu hiện thị trong một thiết bị đo được do người tự đọc. Việc đọc sẽ rất khó khăn vì số là thay đổi số là thay đổi theo thời gian, khi có nhiễu của cảm biến, nhưng điều này có thể được cải thiện nếu chúng ta chọn hiển thị tại một khoảng nhất định, không phải giá trị thực x mà là một tổng các trọng số trung bình và giá trị cuối cùng hiện thị và đầu vào mới, có nghĩa là

$$y_n = k_1.y_{n-1} + k_2.x_n$$

Trong đó $y_{n-1}$ là giá trị hiện thị cuối, và $x_n$ là đầu vào mới.

Ví dụ: nếu ta chọn k = 0.9 và $k_2 = 0.1$

```
» t = linspace (0, 10, 100)                ; thời gian cơ bản
» s = sin (2*pi/5*t)                       ; tín hiệu
» [t,c] = size(t) ; n = 0.1 * rand(r, c)   ; nhiễu
» x = s + n                                ; đầu vào có nhiễu
» y(1) = x(1)                              ; điều kiện đầu
» for i = 2 : 100
» y(i) = 0.9* Y(i-1) + 0.1 * x(i)
```

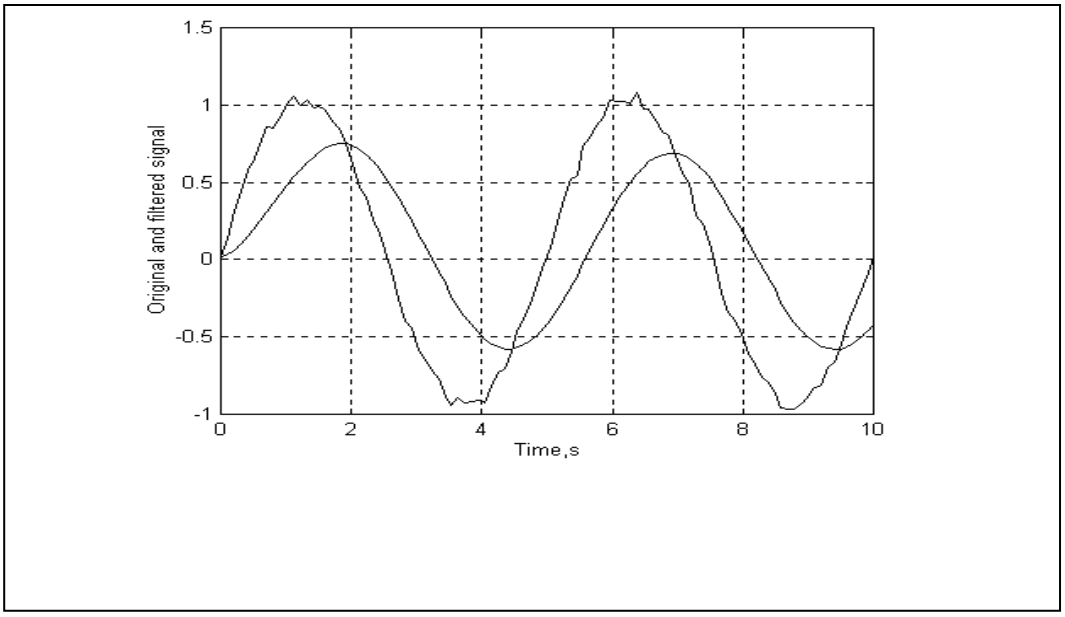
$$y_n = 0.9.y_{n-1} + 0.1.x_n$$

Mối quan hệ này được xác định với $y_n$ mà n > 1. Ta không quan tâm đến việc là giá trị $y_1$ được xác định như thế nào ( ta sẽ xem xét điều kiện sau) . Chọn $y_1 = x_7 \rightarrow$ sử dụng Matlab

Bạn có thể so sánh x và y ***bằng lệnh chấm điểm plot***

```
» plot(t, x, t, y)
```

Kết quả như hình vẽ (I.1)




***Hình 1.1 Làm thẳng tín hiệu nhiễu- Ví dụ 1***

Ta có thể tính theo 2 giá trị bước trước

$$y_n = 0.9.y_{n-1} + 0.05.x_n + 0.05.x_{n-1}$$

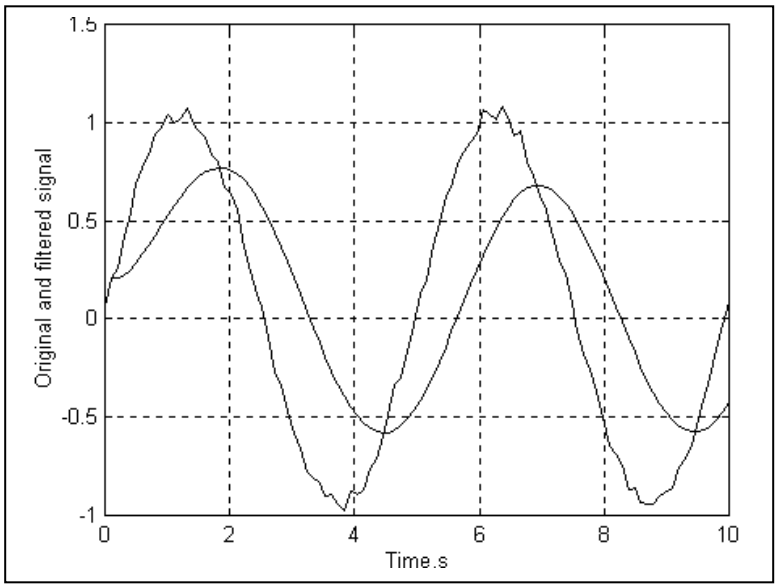
Trong trường hợp này 2 giá trị tự do $y_1$ , và $y_2$. khi đó thì y được sinh ra như sau:

```
» t = linspace (0, 10, 100)                    ; thời gian cơ bản
» s = sin (2*pi/5*t)                           ; tín hiệu
» [t,c] = size(t) ; n = 0.1 * rand(r, c)       ; nhiễu
» x = s + n                                    ; đầu vào có nhiễu
» y(1) = x(1)                                  ; điều kiện đầu
» for i = 3 : 100
» y(i) = 0.9* Y(i-1) + 0.05 * x(i) + 0.05*x(i-1):
» End
```

Cũng như trường hợp trước chúng ta cũng làm

```
» plot(t, x, t, y)
```

* Lọc số có thể giới thiệu đơn giản bắt đầu từ lọc tương tự đơn giản lọc RC - thông thấp theo ví dụ




***Hình 1.2 Làm thẳng tín hiệu nhiễu- Ví dụ 2***

Mạch xoay chiều như hình vẽ

Với thông số như sau:

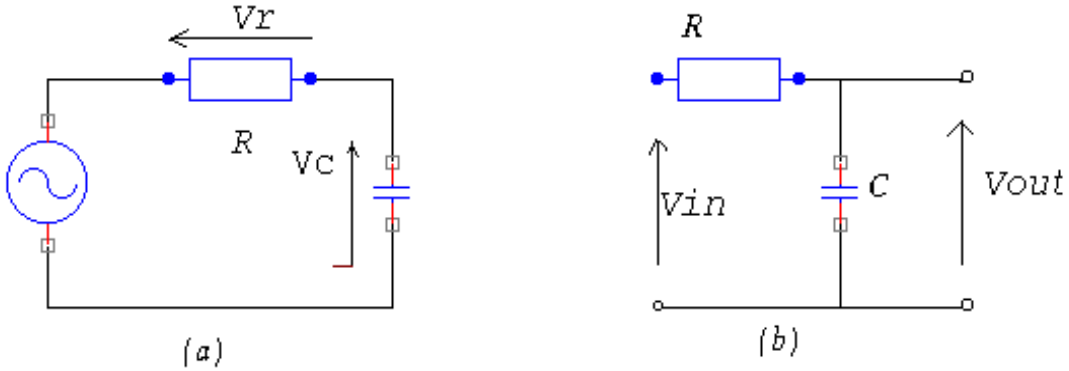
V, 10mV, 50 MHz, pha 0

R, 15Ω

C, 100pF

Điện trở tương đương của tụ điện đọc tính như sau

$$Z_c = \frac{1}{j\omega C} = -j\frac{1}{\omega C}$$




Để đưa các thông số của mạch vào ta làm như sau

```
» V= 0.01;                          %V
» R= 15;                            % Ohm
» omega= 2*pi*50*10^6;              %rad/s
» C= 100*10^(-12);                  %F
» Zc= -j/(omega*C);                 %Ohm
```

Ta tính được tổng trở của mạch

```
» Z= R+ Zc
Z=
        16.0000 -31.8310i
```

Và dòng điện

```
» I = V/Z
I =
        1.2606e-04+ 2.5097e-04i
```

Điện áp rơi trên điện trở

```
» Vr = R*I
Vr=
        0.0018 + 0.0039i
```

Và qua tụ điện chúng ta có thể đồ thị biên pha . Sử dụng MATLAB ***subplot***

```
» subplot (1, 2, 1)
```

Các thông số 1, 2, 1 có ý nghĩa là 1 và 2 ma trận của hình vẽ sinh ra và chấm điểm theo một điểm đầu. Trong MATLAb 3.5 các thông số này sẽ được viết là (121)

Gồm các giá trị V, Vr, Vc như pha cuối cùng. Để định nghĩa pha chúng ta cần đưa điểm gốc của chúng

```
» VV= [0 V]; VVc = [0 Vc];  VVr = [0 Vr]
```

Tiếp đến ta chấm điểm điện thế pha

```
» plot (real(VV) , imag(VV)
```

Để quan sát được thang góc pha chính xác của trục thật cần phải như đối với trục ảo . Có nghĩa là khung chấm điểm là hình vuông

```
» axis('square')
```

Khung của các trục sẽ được sinh ra như sau

```
» axis ([0  0.012  -0.006  0.006])
```

Phần mở rộng của cả hai trục là 0.012. Các giá trị này có thể được nhận bởi thử nghiệm và sai số, như được hiện thị ở đồ thị , khi đánh giá trục tốt nhất và dùng câu lệnh ***axis*** .Hoặc ta có thể sử dụng một hàm khác là ***abs*** để tìm giá trị biên pha lớn nhất và để điều chỉnh lại trục

Chúng ta cũng có thể dùng hàm ***hold*** để giữ cho đồ thị chồng lên hai hình khác,Vr và Vc

```
» hold on
» plot(real(VVr), imag(VVr))
» plot(real(VVc), imag(VVc))
```

Toàn bộ đồ thị sẽ được viết như sau:

```
» title('(a)')
» xlabel('Real'), ylabel('Imaginary')
```

Bạn cũng có thể xác định các hình bằng cách tự đánh dấu các điểm tức là dùng hàm ***gtext***

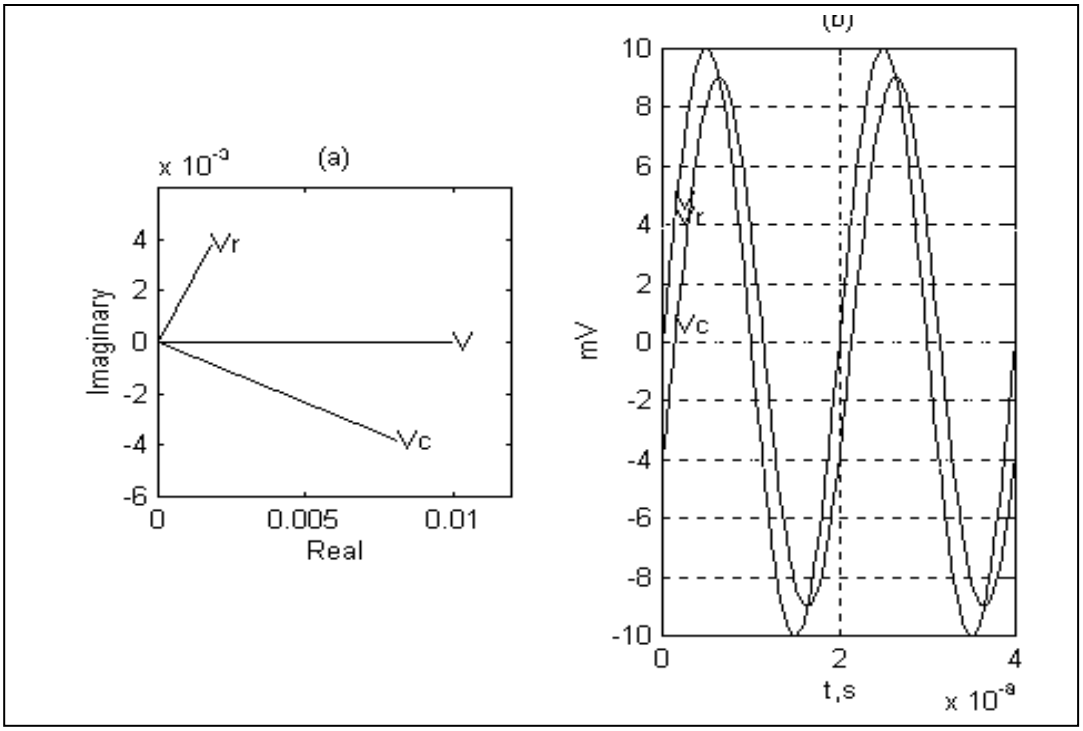
```
» text(real(V), imag(V), 'V')
» text(real(Vr), imag(Vr), 'Vr')
» text(real(Vc), imag(Vc), 'Vc')
```

Sau đó chấm điểm toàn bộ

```
» hold off
```

Ta cũng có thể tính tổng của điện áp như sau(xem thêm hìnhvẽ):

```
» Vr + Vc
ans =
        0.0100 - 0.0000i
» 180*(angle(Vr) - -angle(Vc))/pi
ans =
        90
```




***Hình 1.3. Diện áp trên mạch nói tiếp RC***

Để có được đồ thị và tính toán điện áp ta làm theo những bước sau

```
» f= 50*10^6;                    % tần số, Hz
» T = 1/f;                       % chu kỳ, s
» omega= 2*pi*f;                 %tần số góc, rad/s
» t=0: T/50 :  2*T;              % chuỗi các giá trị
» v = V*sin(omega*t);
» vr = abs(Vr)*sin(omega*t  + angle(Vr));
» vc = abs(Vc)*sin(omega*t + angle(Vc));
```

Để có được điểm như phần tử thứ hai của ma trận đồ thị .

```
» subplot(1, 2, 2)
```

và hcấm điểm theo những lệnh sau:

```
» plot (t, 1000*v, t, 1000*vr, t, 1000*vc)
```

```
» grid, title('(b)')
» xtabel('t,s'), ylabel('mV')
»text(t(5), 1000*v(5), 'V')
»text(t(20), 1000*v(20), 'Vr')
»text(t(50), 1000*v(5), 'Vc')
```

Như ta đã biết mạch RC là một ***mạch lọc thông thấp***. Giả sử ta cho vào đầu vào 10mV, bây giờ ta xem đầu ra của tụ điện. Ký hiệu điện áp vào là $V_{in}$ và điện áp ra $V_{out}$ .

$$Z = R + Rc = R + \frac{1}{j\omega C}$$

$$I = \frac{V_{in}}{Z} = \frac{1}{R - j/\omega C} V_{in}$$

$$Vout = ZcI = \frac{1}{j\omega C} \frac{1}{R - j/\omega C} V_{in}$$

Hàm ***đáp ứng tần*** (dùng hàm biến đổi) của bộ lọc xác định bởi tỉ số

$$H(j\omega) = \frac{Vout}{Vin} = \frac{1}{1 + j\omega RC}$$

Giá trị $\omega_c = 1/RC$ gọi là ***tần số cắt*** ở dạng không thứ nguyên

$$H(j\omega) = \frac{1}{1 + j\omega/\omega_c}$$

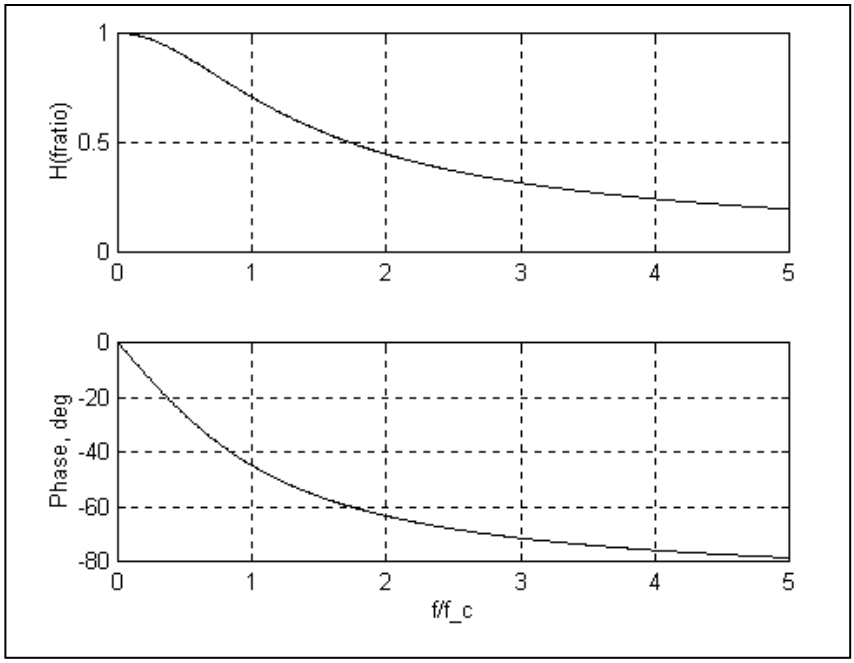
hoặc

$\omega/\omega_c = 2\pi f/2\pi f_c = f/f_c$

Chúng ta nhận được dạng không thứ nguyên của hàm biến đổi

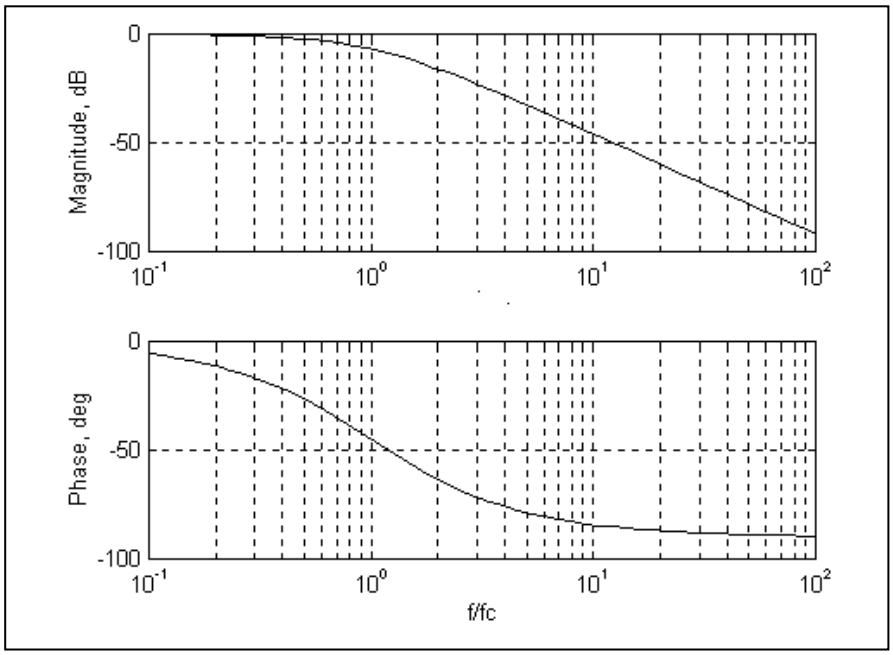
$$H(f) = \frac{1}{1 + jf/f_c}$$

Chúng ta sẽ vẽ biên và pha của hàm biến đổi theo hai đồ thị theo tỉ số tần và theo dãy giá trị biến đổi




***Hình 1.4 Đồ thị biên và tần của mạch thông thấp RC***

```
» fratio= 0 :  0.01:  5;
» H = ones(size(fratio))/(1+j*fratio);
```




***Hình 1.5. Mạch thông thấp dự tính trước***

Vẽ đồ thị theo logarithm

Bây giờ ta xem xét bài toán theo quan điểm xử lý tín hiệu

$$\frac{L}{R}\frac{di}{dt}+i=\frac{v}{R}$$

Với τ = L/R - hằng số thời gian của mạch - y = i và x = *v*/R

Đẳng thức trở thành

$$\tau\frac{dy}{dt}+y=x \qquad (1.1)$$

Ta giả thiết rằng y được lấy mẫu trong khoảng $T_S$ nhỏ so với thời gian τ. Do đó ta có thể xấp xỉ

$$\frac{dy}{dt}=\frac{y_n-y_{n-1}}{T_S}$$

Trong đó $y_n$ - lấy được từ lần đo thứ n của giá trị y, và $y_{n-1}$ của 1 bước trước. Đẳng thức (1.1) được viết lại như sau

$$\tau\frac{y_n-y_{n-1}}{T_S}+y_n=x_n \qquad (1.2)$$

Ta lại có $$y_n=\frac{b_1}{a_1}x_n-\frac{a_2}{a_1}y_{n-1}$$

(1.3)

Biểu thức này ta có được từ bộ lọc đầu tiên

Viết cho trường hợp tổng quát

$$a_1y_n+a_2y_{n-1}+\ldots+a_{n_a}y_{n-n_{a+1}}=b_1x_n+b_2x_{n-1}+\ldots+b_{n_b}x_{n-n_b}$$

(1.4)

hoặc

$$y_n = \frac{b_1}{a_1}x_n+\frac{b_2}{a_2}x_{n-1}+\ldots+\frac{b_{n_b}}{a_1}x_{n-n_{b+1}}-\frac{a_2}{a_1}y_{n-1}-\ldots-\frac{a_{na}}{a_1}y_{n-n_{a+1}}$$

(1.5)

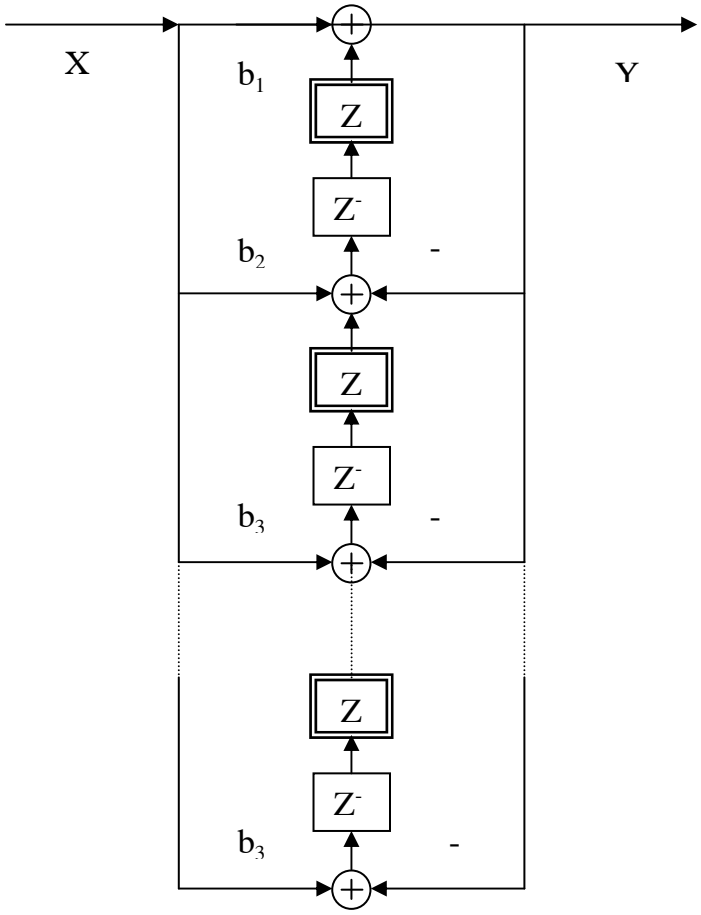
Sử dụng MATLAB ở đây là bộ lọc N tăng, N = max($n_a$ ,$n_b$) - 1

Ta có 2 chuỗi

$A = [a_1\ a_2\ , ..., a_{na}]$

$B = [b_1, b_2, ..., b_{nb}]$

Điều này có thể thấy rõ qua *đồ thị 1.6* trong đó $a_i$ và $b_i$ các hệ số được làm bình thường hoá bằng cách chia cho $a_1$ hệ số của $y_n$ . Trong hình 2 là cho ký hiệu thao tác dịch thời gian *unitary time shift operator*). Cho x - rời rạc của tín hiệu f(t) bị lấy mẫu theo chu kỳ $T_S$, do đó $x(n) = f(n.T_S)$ và $z_x(k)$ xác định thành phần thứ k $z_x(k) = x(k+1)$. Tương tự như vậy thành phần thứ của $z^{-1}$ n là x(k-1) z - ký hiệu thích hợp cho việc thao tác số nhanh (xin đọc thêm về biến đổi z trong các giáo trình về xử ký tín hiệu).




***Hình 1.6. Chuyển đổi trực tiếp ở dạng II***

Việc thể hiện lại chỉ ra trên hình 1.6 gọi là dạng chuyển đổi II và là mô hình tối thiểu việc cất giữ (tức là tiết kiệm bộ nhớ nhất). Trên thực tế tại mỗi bước này chúng ta cần cất mỗi giá trị của trạng thái $z_1, z_2, ..., z_n$ được thể hiện trên đồ họa bằng khung đúp ở đường trung tâm (xem hình vẽ)

Bộ lọc được thực hiện như sau:

(1) Thực hiện

(a) Vector trạng thái đầu, thường được đánh dấu bằng $Z_i$ và được đặt vào bộ trễ $Z_j$ không xuất hiện, tất cả các giá trị $Z_j$ sẽ đặt = 0

(b) Giá trị của A và B được chuẩn hoá bằng chia cho a1

(2) Vòng lặp chính. Tại các bước này, chức năng lọc thể hiện theo những thao tác sau:

(a) Sinh ra $y_n$ từ quan hệ $y_n = b_1x_n + Z_1$

(b) Đặt vào $Z_j$ với j từ 1 đến N-1 trong

$$Z_j = b_jx_n + Z_{j+1} - a_jy_n$$

và đối với j = N ta tính như sau:

$$Z_N = b_Nx_n - a_N.y_n$$

Vòng lặp được tiếp tục cho đến khi chuỗi x hội tụ

(3) Bước cuối cùng. Giá trị $Z_j$ được sao chép đến vector ra $Z_f$

Đẳng thức (1.5) có thể được thực hiện bởi MATLAB bằng chức năng lọc. Ở đây có 4 cách gọi hàm lọc.

y = filter (B, A, x)

y = filter (B, A, x, $Z_i$)

[y,$Z_f$] = filter (B, A, x)

[y,$Z_f$] = filter (B, A, x, $Z_i$)

Trong đó $Z_i$ và $Z_f$ là một chuỗi các thông số tối ưu, liên hệ với trạng thái đầu và trạng thái cuối, đã được mô tả như ở trên. Hàm lọc có thể trả về giá trị cuối cùng của Z như dãy $Z_f$, nếu nó gọi 2 biến số bên vế trái:

```
»[y, Zf] = filter (B, A, x)
```

Ví dụ: cho

```
» x = [1, 2, 1, 2, 1, 2 , 1, 2, 1, 2];
» B = [0.5];
» A = [1, -0.25, -0.25];
```

Gọi hàm lọc như sau:

Các tần số được tính toán như sau

| Bước | Vào | Ra | $Z_1$ | $Z_2$ |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | 1 | .5 | 0.125 | 0.1250 |
| 2 | 2 | 1.125 | 0.4063 | 0.2813 |
| 3 | 1 | 0.9063 | 0.5078 | 0.2266 |
| 4 | 2 | 1.508 | 0.6035 | 0.3770 |
| 5 | 1 | 1.104 | 0.6528 | 0.2759 |
| 6 | 2 | 1.653 | 0.6891 | 0.4132 |
| 7 | 1 | 1.189 | 0.7105 | 0.2973 |
| 8 | 2 | 1.71 | 0.7249 | 0.4276 |
| 9 | 1 | 1.225 | 0.7338 | 0.3062 |
| 10 | 2 | 1.734 | 0.7397 | 0.4335 |

Trong bảng này sinh ra bởi MATLAB M-file, số được làm tròn 4 số sau dấu phẩy.

```
» y(:)
        ans =
0.5000
1.1250
0.9063
1.5078
1.1035
1.6258
1.1891
1.7338
```

Người đọc có thể tính lại số này theo mô tả ở *hình 1.6* (nói ở trên). Cuối cùng, sau khi xác định chuỗi số x, A và B liên hệ lại hàm lọc như được chỉ ra

```
» [Y,Zf] = filter (B, A, x, Zi) ;
```

Chúng ta nhận ra chuỗi y được tính toán bởi tệp M, và giá trị lớn nhất của $Z_1$ và $Z_2$ trên bảng.

```
» Zf
Zf =
0.7397
        0.4335
```

```
» Zf
Zf =
0.7397
        0.4335
```

## 3. GỌI HÀM LỌC VỚI ĐIỀU KIỆN ĐẦU

Nếu bạn biến các điều kiện đầu để cất vào khâu trễ, bạn có thể gọi các bộ lọc bởi.

Thông thường dùng nếu ta có một tín hiệu x dài không thể cất gửi vào bộ nhớ của máy tính của bạn. Bạn có thể trạt x ra thành điểm gọi là $x_1$, $x_2$, ... , $x_n$ như trong MATLAB

Và bộ lọc phân biệt được chúng chính xác. Giá trị cuối cùng , $Z_f$ của các khâu trễ tại mỗi bước được sử dụng như là các giá trị đầu, $Z_i$, cho bước tiếp theo. Để hiểu rõ ta

```
»x = [x1 ; x2 ; ... ; xn]
```

dựng 1 chuỗi với 100 số ngẫu nhiên.

```
» x = rand(100,1) ;
```

Chúng ta trặt chúng thành nhiễu chuỗi nhỏ

```
» x1 = x(1 : 25) ;
» x2 = x(26 : 50) ;
» x3 = x(51 : 75) ;
» x4 = x(76 : 100)
```

Bây giờ chúng ta xây dựng bộ lọc liên hệ với các thành phần của các chuỗi con. Khi xây dựng chúng sử dụng các giá trị của trạng thái cuối cùng, Zf, sinh ra bởi lần gọi đầu tiên như vậy một trạng thái đầu Zi, cho cho lần gọi thứ 2, cứ như vậy:

Để kiểm tra bạn có thể sử dụng khi chuỗi số x

```
» y = filter(b, a, x) ;
```

Và bạn thay đổi kết quả được sinh ra bởi "partial"

```
» max(max(abs(y-[y1 ; y2 ; y3 ; y4]))) ;
ans =
        0
```

Như chúng ta nhiều khi thấy vector $Z_i$ là phần chọn và bạn có thể làm ra. Trong trường hợp này hàm lọc sử dụng vector không chuẩn với chiều dài N toàn là 0 (vector 0). Nếu Z xuất hiện nó cần phải dài N, bằng với bộ lọc đã được định trước , và 1 + N như ở phần trên MATLAB.

Để xác định bộ lọc của N, chúng ta cần cung cấp N trạng thái như là giá trị của vector Zi, trong trạng thái để chuỗi A & B. Ta có N, trạng thái đầu cho đầu vào ở trong trong dạng của vectơ y, có chiều dài N và cần giá trị đầu của y đồng thời với $y_i$. Nếu chúng ta viết N đẳng thức đầu của thuật toán lọc, bạn nhận được mối quan hệ sau:

$$y_1 = Z_{i1} + b_1 + b_1x_1 \quad (1.6)$$

$$y_2 = Z_{i2} + [b_1x_2 + b_2x_1] - [a_2 - y_1] \quad (1.7)$$

$$y_3 = Z_{i3} + [b_1x_3 + b_2x_2 + b_3x_1] - [a_2 - y_2 + a_1y_3] \quad (1.8)$$

.

.

.

$$y_k = Z_{ik} \sum_{i=1}^{k} b_i x_{k+1-i} - \sum_{i=1}^{k-1} a_{i+1} y_{k-i} \quad (1.9)$$

.

.

.

$$y_N = Z_{iN} \sum_{i=1}^{N} b_i x_{N+1-i} - \sum_{i=1}^{N-1} a_{i+1} y_{N-i} \quad (1.10)$$

Nếu chúng ta muốn N giá trị đầu của bộ lọc phù hợp với $y_i$ , chúng ta có thể trừ $y_i$ với y trong N đẳng thức, và tìm ra $Z_i$ . Hàm lọc được viết để giải quyết vấn đề này. Hãy làm những bước chuẩn bị sau để gọi tệp ***filteric.m.***

** ***Chú ý:*** *Trạng thí dụ này dùng tín hiệu giả.*

Như trong ví dụ : Cho X hiện thị 1000 điểm đầu của tín hiệu bị lấy mẫu tại 100 $H_Z$ . Chúng ta muốn cho qua tín hiệu từ 30 Hz (lọc thấp) của điểm 5, và nhận được tín hiệu y như y(i) = x(i) và i =1 đến 5

Đầu tiên ta xây dựng tín hiệu với

```
» x = rand(1000 , 1) ;
```

Tiếp đến chúng ta xác định thông số của bộ lọc theo thí dụ như ở phần tổng quan của toolbox xử lý số tín hiệu.

```
» yi = x(1 : 5) ;
```

Các trạng thái đầu được xác định như sau

```
» [b, a] = butter (5 , 30/50)
b =
        0.1084 0.5419 0.0837 1.0837 0.5149 0.1084
a =
        1.000          0.9853 0.9738 0.3864 0.1112 0.0113
```

Và trạng thái đầu của khâu trễ

```
» Zi = filteric (b, a, x, yi) ;
```

Chúng ta liên hệ với hàm lọc

```
» y = filter(b, a, x, Zi) ;
```

Chúng ta có thể hiện thị 5 nhóm đầu của chuỗi vào và ra và thay đổi cho chúng bằng nhau

```
        » [ X(1:5), Y(1:5) ]
ans=
        0.2190      0.2190
        0.0970      0.0470
        0.6789      0.6789
        0.6793      0.6793
        0.9347      0.9347
```

# 4. THIẾT KẾ CÁC BỘ LỌC SỐ

## 4.1. CÁC ĐỊNH NGHĨA :

Một trong những vấn đề chung nhất xuất hiện trong xử lý số tín hiệu là cấu trúc của bộ lọc với các đặc tính biên tại các tần số khác nhau. Một trong tools (công cụ) trong toolbox xử lý tín hiệu là 2 hàm ***yulewalk*** và ***remez.***

Chúng ta gọi chúng với bộ lọc số H của N điểm, đặt tần số lấy mẫu số liệu x, sinh ra tần số mới y, quan hệ với x theo đẳng thức.

$$a_1 y_n + a_2 y_{n-1} + \ldots a_{N+1}\, y_{n-N} =$$

$$\mathbf{b_1 x_n + b_2 x_{n-1} + \ldots b_{N+1}\, x_{n-N}} \qquad (1.12)$$

Các hệ số $B = [b_1, b_2, \ldots, b_{N+1+}]$ và $A = [a_1, a_2, \ldots, a_{N+1}]$ đều xác định các hệ số $\neq 0$. Song chúng ta có thể giả thiết chúng chuẩn theo $a_1$. Hơn thế nữa, tại các hệ số cuối

cùng $a_{N+1}$ hoặc $b_{N+1}$ có thể khác 0, trong các trường hợp khác bị lọc cần xác định vector thu gọn A và B, và chúng cần nhỏ hơn N. Hàm trong toolbox MATLAB sinh ra các hệ số của bộ lọc . (***yulewalk, cheb1*** và các hàm khác ...) luôn sinh ra các hệ số qui chuẩn, thành phần của hàm lọc (*filter*).

Khi sử dụng thao tác dịch thời gian như xác định phần trước, bộ lọc H được biểu diễn bằng hàm phân thức sau.

$$H(z) = \frac{B(x)}{H(z)} = \frac{b_1 + b_2 Z^{-1} + b_N Z^{-(N-1)}}{a_1 + a_2 Z^{-1} + \ldots + a_N Z^{-(N-1)}} \qquad (1.13)$$

Với $a_1 = 1$ và hệ số lớn nhất $a_N$ và $b_N \neq 0$

Rất tiếc là trong tất cả các version của MATLAB ta khi dùng ***help yulewalk*** không được đúng lắm, chúng sẽ hiện ra đoạn văn bản như sau

YULEWALK RECUSIVE FILTER DESIGN USING A LEAST-SQUARES METHOD.

[B,A] = yulewalk(N,F,M) finds the N-th order recursive filter

coefficients B and A such that the filter:

$$\frac{B(z)}{A(z)} = \frac{b(1) + b(2)z + \ldots.. + b(n)\, z^{-(n-1)}}{1 + a(1)z + \ldots.... + a(n)z^{-(n-1)}}$$

Trong đó chỉ số của A bị sai dịch như sau n = N + 1 , song cho ví dụ, bộ lọc của 4 sẽ được xác định bởi vectors

$B = [b_1, b_2, b_3, b_4, b_5]$

$A = [a_1, a_2, a_3, a_4, a_5]$

Với $a_1 = 1$ và hệ số cuối b5 hoặc a5 $\neq 0$

Dạng như đã nói ở trên bị giới hạn vì lỗi ở trong sách sử dụng đúng của tương ứng với hàm số như bộ lọc.

Nếu các nhóm $a_2, a_3, \ldots, a_N$ đều bằng 0 thì bộ lọc sẽ gọi FIR (bộ lọc đáp ứng xung hữu hạn). ***yulewalk*** được dùng để tổng hợp bộ lọc IIR , khi hàm ***remez*** được sử dụng cho FIR

### 4.2 Xác định đặc tính tần của bộ lọc.

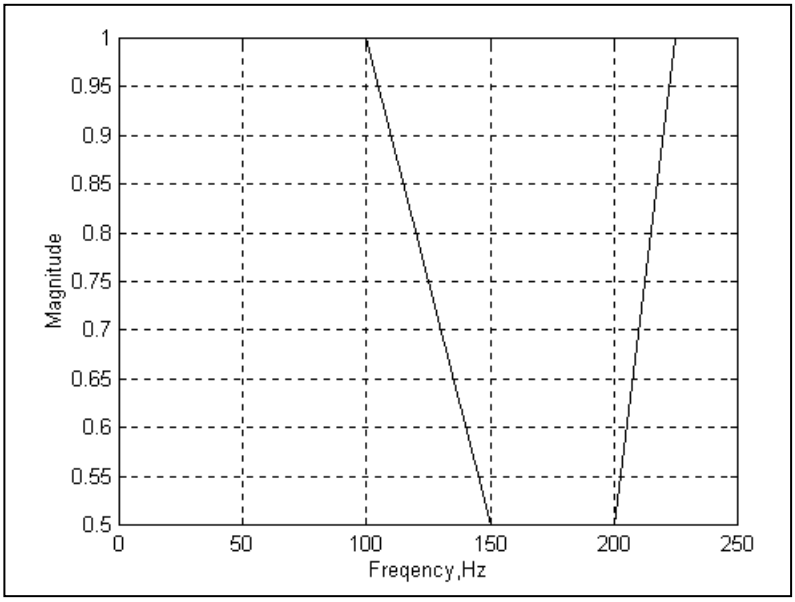
MATLAB cho phép ta định nghĩa số của tần số $fr_1$ , $fr_2$ , ..., $fr_i$ ,..., $fr_k$ và biến tương ứng $mag_1$, $mag_2$ , ..., $mag_i$,..., $mag_k$ và mô tả bộ lọc số gần đúng (xấp xỉ) với đáp ứng của bộ lọc tương tự.

Tần số đáp ứng của bộ lọc số phụ thuộc vào tần suất lấy mẫu . Một nửa tần số đáp ứng của lấy mẫu được gọi là ***Nyquist*** .Một số hàm của MATLAB được nhanh chóng đẩy vào vùng tần số không thứ nguyên , có nghĩa là bằng việc định nghĩa tần số không thứ nguyên 50/(1000/2) = 0.1 và 150 Hz sẽ tương ứng với 150/(1000/2) = 0.3.

Để xác định đặc tính của bộ lọc, chúng ta cần có 2 chuỗi : 1 là tần số không thứ nguyên, f=[$f_1$, $f_2$ ,..., $f_k$ ] và 1 tương ứng với biên m = [$m_1$, $m_2$, ..., $m_k$] MATLAB đòi hỏi $f_1 = 0$ và $f_k = 1$.

Như trong ví dụ , chúng ta giả thiết rằng tín hiệu x được lấy mẫu ở 500Hz và chúng ta muốn xây dựng bộ lọc với tần số biên như sau:

| Từ | Đến | Biên |
|---|---|---|
| 0 | 100 | 1.0 |
| 100 | 150 | giảm đến từ 1 đến 0.5 |
| 150 | 200 | luôn là hằng 0.5 |
| 200 | 225 | tăng đều từ 0.5 đến 1 |
| 200 | 250 | - |




***Hình 1.7 Đặc tính tần của bộ lọc***

Chúng ta đưa đặc tính bộ lọc được mô tả vào MATLAB bởi

```
» fHz0 = [0  1000  150  200  225   250];
» m0 = [1.0  10   0.5   0.1  1.0   1.0] ;
```

Để kiểm tra chúng ta có thể vẽ đồ thị như hình vẽ 1.8

```
» plot (fHz0 , m0) ;
```

MATLAB dùng công cụ  để xây dựng bộ lọc số IIR và FIR với những đặc tính nhất định. Hơn thế nữa trong phần tổng quan về xử lý tín hiệu chúng ta đã xem xét những vấn đề ưu nhược điểm, ở đây chúng ta đề cập đến hàm ***yulewalk*** cho IIR và ***remez*** cho tổng hợp FIR. Để sử dụng hàm ***yulewalk***, chúng ta cần qui các tần số thành không thứ nguyên

```
» fs = 500;
» f0 = fHz0 / (fs/2) ;
```

Trong đó fs là tần số lấy mẫu được xấp xỉ đúng nhất, bằng phương pháp bình quân  phương nhỏ nhất. Chúng ta thử bộ lọc 6 điểm:

```
» [bIIR, aIIR] = yulewalk (6, f0, m0) ;
```

Bây giờ chúng ta có thể kiểm tra lại việc xấp xỉ của chúng ta bằng cách so sánh đáp ứng của bộ lọc được xác định bởi [bIIR, aIIR] với đáp ứng đã có được. Chúng ta đã  biết là đáp ứng của bộ lọc H(z) tại tần số ω rad/s được cho bởi giá trị H(z) cho Z = $e^{i.\omega/fs}$ trong đó fs là tần số lấy mẫu. Như chúng ta đã giả thiết cần 5 điểm đặt trước trên

trục x, như 50 điểm từ 0 đến tần số ***Nyquist***, tính rad/s. Trong MATLAB ta có được chúng theo tần số Hz cùng

```
» fHz1 = linspace (0, 250, 50) ;
```

và chuyển chúng thành rad/s bằng

```
» om1 = 2 * pi * fHz1 ;
```

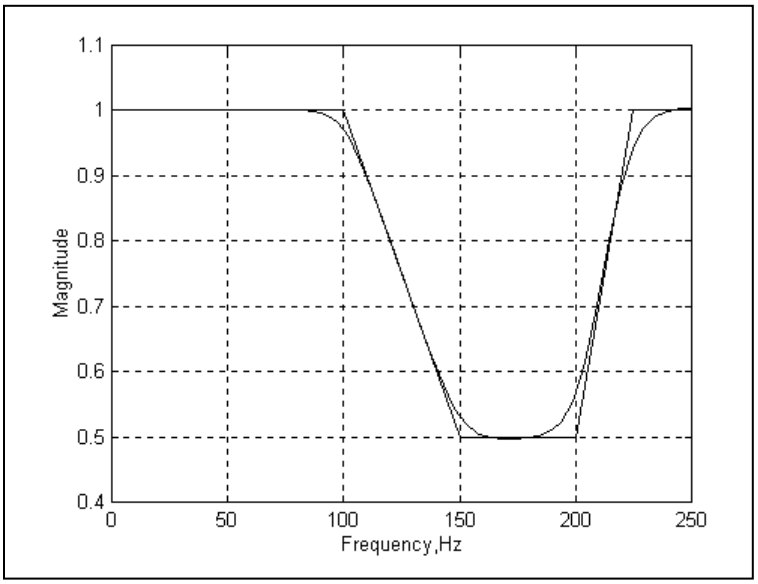
Chúng ta muốn tính đáp ứng biên của bộ lọc dùng lệnh sau

```
» 2 = exp (sqrt (-1) * om1 / fs) ;
» mIIR = abs (polyval (bIIR, z). / polyval (aIIR, z)) ;
```

Bây giờ bạn có thể so sánh sự trùng của đáp ứng cho trước và đáp ứng thực.

```
» plot (tHz0, m0, fHz1, mIIR) ;
```

Những kết quả này được mô tả trong hình 1.5. Nếu như sự xấp xỉ không tốt như giả định của ta thì chúng ta cần tăng số điểm cho trước của bộ lọc.




***Hình 1.8. Bộ lọc IIR định nghĩa và bộ lọc thực***

Bây giờ chúng ta giải quyết cùng một vấn đề sử dụng FIR. Bộ lọc FIR có thể có số điểm cho trước lớn hơn để đạt được việc so sánh chúng ta dùng bộ lọc với số điểm là 20:

```
» bFIR = remez (20, f0, m0) ;
```

Hàm ***remez*** cho ta chuỗi b, tất cả bộ lọc FIR a = [1].

Bạn có thể kiểm tra lại kết quả bằng hình vẽ.

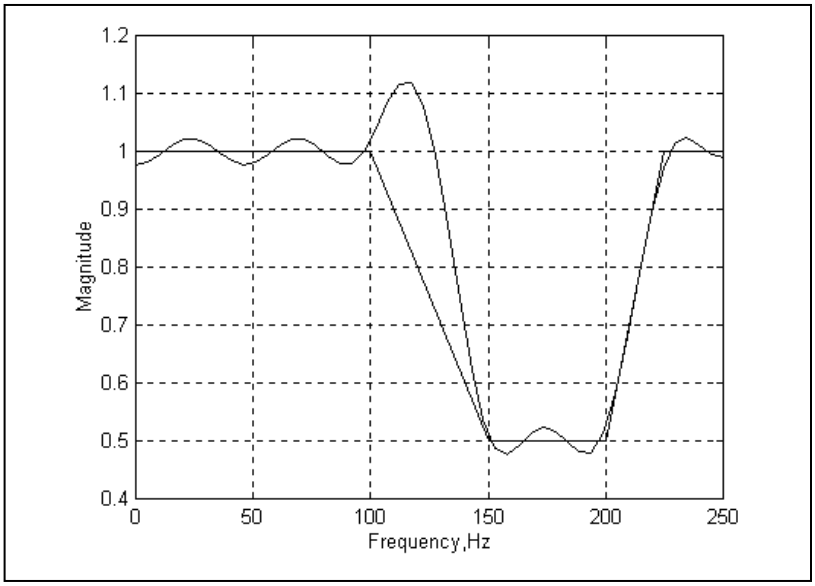
```
» mFIR = abs (polyval (bFIR, z)) ;
» plot (fHz0, m0, fHz1, mFIR) ;
```

Kết quả đồ thị như *hình vẽ 1.8*.

Ngoài ra trong toolbox xử lý số tín hiệu có bổ xung thêm một số hàm để tổng hợp bộ lọc IIR: ***cheby1, cheby2, ellipt***, hàm số ***yulewalk*** đòi hỏi hai chuỗi số: 1 - tần số, 2- là đáp ứng biên. Chúng ta cần đưa thêm xác định kiểu lọc: thông cao, thông thấp, thông giữa và các trạng thái của biến đổi nhỏ (ripple). Trong trường hợp tổng hợp FIR cũng tương tự IIR dùng ***remez*** hoặc ***fir1 và fir2.***

Để tính toán đáp ứng tần số của bộ lọc số, MATLAB dùng hàm tần số ***freqz***, nhanh hơn là dùng thẳng tính toán của ***H (squtt (-1) * om / fz).***

Để hiểu thêm quan hệ của phương pháp này, mời bạn đọc thêm sách hướng dẫn sử dụng




***Hình 1.9. Sự xác định và thực hiện của hàm lọc FIR***

**Ví dụ:** Tách 2 sóng hình sin từ tổng của chúng.

Bộ lọc này dùng tần số để phân biệt hoặc tách thành phần cosin từ tín hiệu tổng hợp. Như ở trong ví dụ, ta xây dựng tín hiệu đơn từ 2 sóng hình sin, một với tần số 100Hz, một là 400Hz, trong khoảng thời gian 0.1 giây. Tần số lấy mẫu là 2000 Hz.

```
» fS = 2000 ;
```

```
» t = 0: (1/fS) : 0.1 ;
» x1 = sin(2 + pi * 100 x t) ;
» x2 = sin(2 * pi * 400 * t) ;
» x = x1 + x2 ;
```

Tín hiệu x xuất hiện trên đường 1 ở hình 1.10. Bây giờ ta sử dụng ***yulewalk*** để mô tả bộ lọc thông thấp và thông cao. Tần số cơ bản được xác định bởi

```
» fH20 = [0  225  275  1000] ;
```

Biên đặc biệt của bộ lọc thông thấp là

```
» m10 = [11  00] ;
```

Và bộ lọc thông cao là

```
» mh0 = [00  11] ;
```

Tần số mô tả không thứ nguyên là

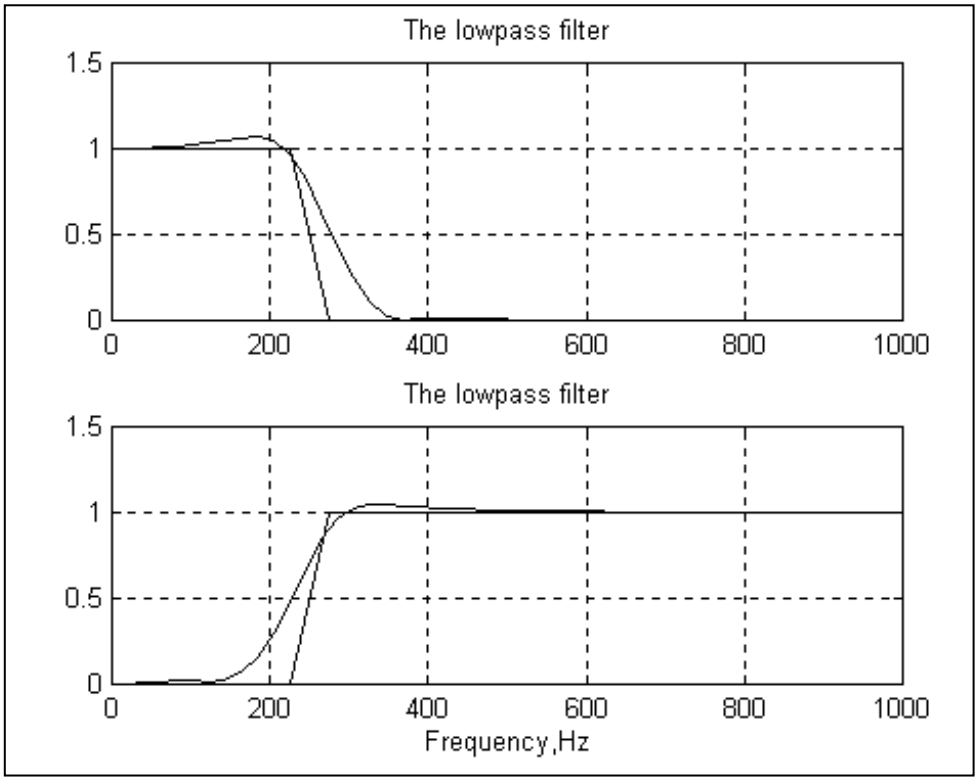
```
» f = fH20/(fS/2) ;
```

Các thông số của bộ lọc thông thấp được tính bởi

```
» [b1 , a1] = yulewalk (6,f0, mh0) ;
```

Để kiểm tra lại chất lượng của bộ lọc chúng ta tính và chấm điểm ở các tần số của chúng với lệnh sau:

***Hình1.10 Đặc tính tần của bộ lọc***




```
» fHz1 = linspace (0, fs/2 , 50) ;
» om1 = 2 * pi *H21 ;
» Z = exp (sqrt(-1) * 0m1/fs) ;
» m1 = abs(polyval(b1,z) . /polyval(a1,z)) ;
» mh = abs(polyval(bh, z)./polyval (ah,z)) ;
```

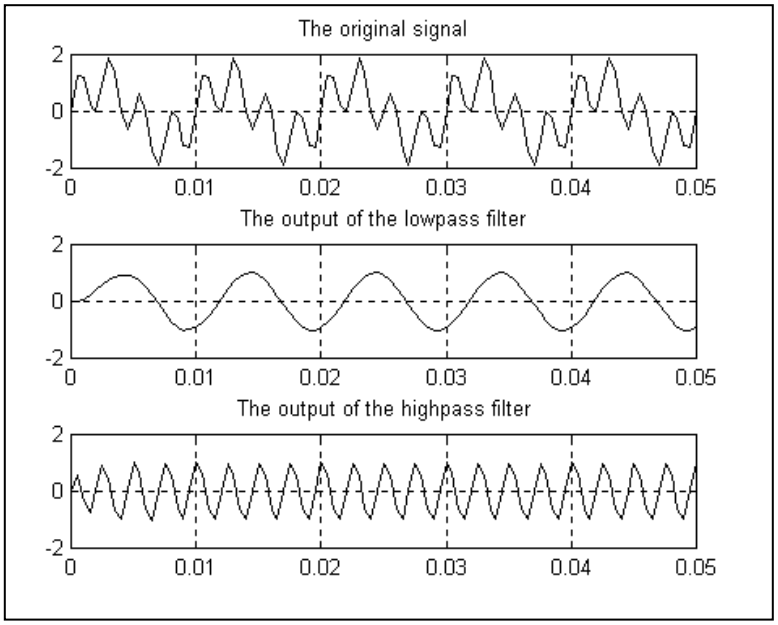
Chúng ta có thể so sánh đặc tính của bộ lọc thông cao với đặc điểm sau:

```
» plot (fH20, mh0 ,fH21, ml) ;
```

Đặc tính tần của 2 bộ lọc có thể nhìn thấy trên ***hình 1.9***. Sai lệch với kết quả không xa. Chúng ta lọc được tín hiệu

```
» y1 = filter (bl, al, x) ;
```

Và chấm điểm y2 , có thể nhìn thấy thành phần 100Hz.




***Hình 1.11. Chỉ ra tín hiệu gốc của x và đầu ra của 2 bộ lọc y1 và y2 trong thời gian 0.05 giây.***

## 4.3. Biến đổi nửa tuyến tính Tustin

Thông thường ta có hàm biến đổi H(s) của bộ lọc tuyến tính xác định tần số chủ đạo, và muốn xấp xỉ nó với bộ lọc số $H_d(z)$ ở đây số bước chung để chuyển bộ lọc tương tự thành số "tương đương" , thì cần đọc thêm bộ lọc ***Franlin Powell và Workman (viết 1990)***

Một trong những khả năng để đạt được s= s(z) trong các biến số z, điều này được xấp xỉ gần nhất .

$$\mathbf{H_d(z) = H(s(z))\ ;} \qquad (1.14)$$

Biến đổi

$$s(z) = \frac{2}{Ts}.\frac{z-1}{z+1} \qquad (1.15)$$

Gốc là biến đổi Tustin" và MATLAB dùng hàm ***bilinear***

Trong biểu thức (1.15) Ts là đoạn lấy mẫu là z, là thao tác dịch thời gian

Biến đổi này gần với luật biến đổi ***tranpezoidal integration*** và chúng có tác dụng ở trong khoảng tác dụng của lấy xấp xỉ ***trapezoidal*** trong một số trường hợp nếu hàm đủ bằng phẳng trong một không gian lấy mẫu đủ ngắn . Biến đổi **Tustin** sẽ cho ta phép biến đổi đại số chuyển bộ lọc tương tự về bộ lọc số.

*Muốn hiểu kỹ hơn về phần này bạn nên đọc kỹ lý thuyết xử lý tín hiệu.*

Như trong ví dụ, chúng ta có thể mô tả bộ lọc số với đặc tính tương tự (2 tầng), lọc thông thấp và hàm biến đổi.

$$H(s) = \frac{\omega_n^2}{S^2 + 2\xi\omega_n s + \omega_n^2} \qquad (1.16)$$

Trong đó $\omega_n$ = 30rad/s , $\xi$= 0.6 và ký hiệu lấy mẫu tại 100Hz. Để giải quyết vấn đề này trên MATLAB ta định nghĩa tần số lấy mẫu

```
» f1 = 100 ;
```

Các thông số của bộ lọc

```
» ωn = 3.0 ; zi = 0.6 ;
```

Số là

```
» num = [ωn ^2] ;
```

Số lần của đặt tên, để có công suất của s là

```
» den = [1 2 * zi * ωn  ωn ^2] ;
```

Bạn có thể nhận được bộ lọc số tương đương

```
» [a,b] = bilinear (num, den , fs) ;
```

Để so sánh bộ lọc số, định trước [a,b] , ta vẽ 2 đương quan hệ theo tần số (đơn vị rad/s)

```
» om = linspace(0,300) ;
```

Tiếp theo ta tính tần số đáp ứng này của bộ lọc số bởi.

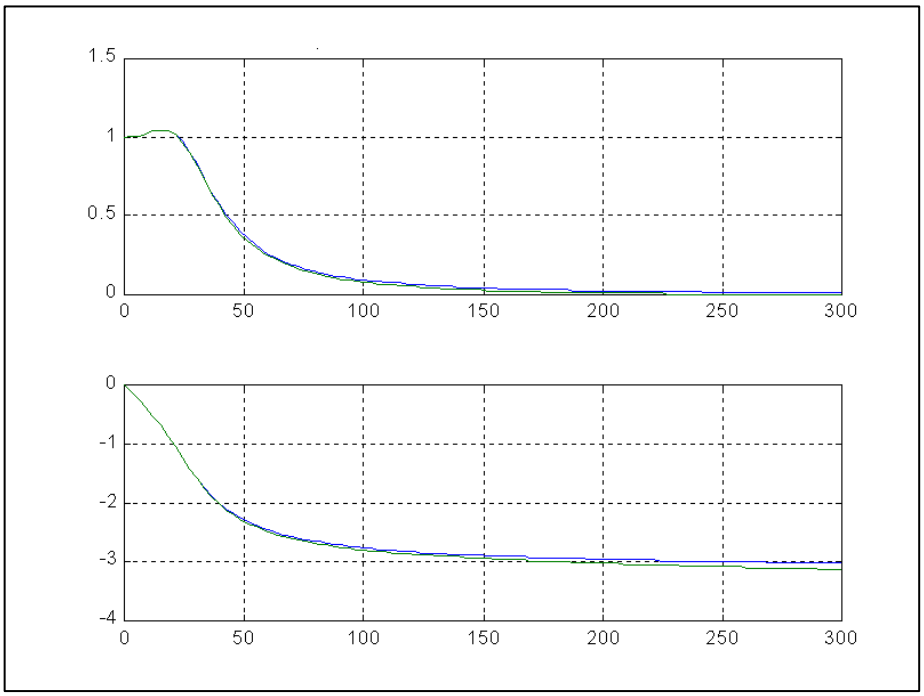
```
» z = exp(s/fs) ;
» hz = polyval(b,z) ./ polyval (a,z) ;
```

Và chúng ta so sánh biến của 2 đáp ứng này bằng

```
» subplot (2, 1, 1) ;
```

```
» plot (om, abs(hs), om, abc(hx))
» subplot (2 ,1, 2) ;
» plot (om, angle(hs), om, angle(hs)) ;
```

Kết quả thu được trên *hình vẽ 1.12*.




***Hình 1.12 Sự xấp xỉ bộ lọc tương tự và bộ lọc IIR***

# 5. BIẾN ĐỔI FOURIER RỜI RẠC

Một tín hiệu f(t) của một biến liên tục t có chu kỳ lặp lại T nếu

f(1 + T) = f(t). Nếu giá trị chu kỳ nhỏ nhất dương thì chu kỳ đó gọi là chu kỳ cơ bản của f.

Tương tự tín hiệu số d = [... , d(-1) , d(0) , d(1) , d(2) , ...]

gọi là có chu kỳP, nếu với mỗi số nguyên dương k : d(k)= d(k+P) .

Nếu ta lấy mẫu hàm số chu kỳ f của chu kỳ thực T tại các thời điểm lấy mẫu $T_S$, và $T_S$ là ước số của T, ta gọi $T_S = T/N$ , lần lấy mẫu d (bị lấy mẫu vesion d) của f là một tín hiệu số lặp với chù kỳ N. Gọi là $d(k) = f(k.T_S)$

$d(k+N) = f((k+N) . T_S)$

$= f((k+N) . T/N)$

$= f(k . T/N + T)$

$= f(k . T/N)$

$= f(k . T_S)$

$= d(k)$

Sự lặp lại của tín hiệu theo chu kỳ N là phân bố đều bới số N , ví dụ đối với mỗi lần 1 đến N $\rightarrow$ trong MATLAB dùng vector x với chiều dài N. Vector x gọi là biểu diễn chủ đạo của tín hiệu d và được xác định một cách đơn giản bởi chuỗi của thành phần như sau x(h) = d(h) với h = 1 đến N.

Từ thời gian biểu diễn chủ đạo x của d sử dụng chu kỳ lặp của tín hiệu, do đó dễ dàng xây dựng với quan hệ : d(h) = x(k), đối với k có điều kiện sau

(1) $1 \leq k \leq N$

(2) (h - k) chia hết cho N.

Với mỗi đầu vào x(k) của tín hiệu x có thể là một số thực hoặc phức. Đặc thù là bạn có thể thấy thời điểm đó phân biệt tín hiệu thực, những tín hiệu khác lấy từ việc tính toán hoặc từ quan điểm lý thuyết. Cho trường hợp chung thì f(k) là số phức. Và giả định rằng tín hiệu thực trên thực tế phần biến đều bằng 0. Biến đổi Fourier rời rạc, DFT, của một chuỗi số x độ dài N là một chuỗi khác X, cũng có độ dài N. Biến đổi Fourier gọi là biến đổi ngược IFT (Inverse Fourier Transform). Ta sẽ còn quay lại 2 khái niệm này ở phần sau.

IFT và DFT dùng trong MATLAB rất có giá vì sử dụng thuật toán FFT (Biến đổi Fourier nhanh). FFT là một thuật toán rất được phổ biến Coolly và Tukey (1965). Thuật toán này cần xấp xỉ Nlog (N) các phép toán để tính DFT, so với $N^2$ phép toán cho phép bình thường.

Biến đổi Fourier có 2 bài toán ứng dụng chính. Đầu tiên là tiện cho người dùng. Nhiều thao tác trên tín hiệu nhanh hơn khi dùng trên tần số chính. Có khoảng 15 hàm số trên Toobox xử lý tín hiệu độc lập tác dụng lên các ứng dụng các hàm trực tiếp trên thời gian chủ đạo, theo cách chuyển vector (gốc) thành tần số chính, ứng dụng hàm xấp xỉ và chuyển kết quả ngược trở lại thời gian chủ đạo.

Ứng dụng thứ 2 của DFT để nhận dạng các thành phần tần số của tín hiệu , như ta sẽ thấy ở mục sau.

## 6. GIỚI THIỆU TÓM TẮT DFT (BIẾN ĐỔI FOURIER RỜI RẠC)

Trong $C^N$ (hay còn được ký hiệu $R^N$ không gian N chiều), ví dụ chúng ta có cơ bản, ký hiệu bởi $i_1, i_2, ... ,i_N$ và được xác định bởi.

$i_1 = (1, 0, 0, ..., 0)$

$i_2 = (0, 1, 0,..., 0)$

$i_3 = (0, 0, 1, ..., 0)$

.

.

.

$i_N = (0, 0, 0, ..., 1)$

$C^N$, hơn nửa có chấm điểm (dot product), ký hiệu bởi (!) và xác đinh như sau: Nếu

$x = [x_1, x_2,..., x_N]$

$y = [y_1, y_2,..., y_N]$

Các điểm sinh ra là:

$$<x/y> = \sum_{h=1}^{N} x_h \ \bar{y}_h$$

Trong đó $\bar{y}_h$ gía trị trung bình, liên hợp phức $y_h$ khi xác định thông thường của điểm trong $R^N$ và chú ý MATLAB sử dụng như đã được giới thiệu phần đầu: Phần ma trận và đồ họa .

Phần bù số ảo của $y_h$ khi xác định thông thường của điểm trong $R^N$ và chú ý MATLAB sử dụng giới thiệu phần đầu :Phần ma trận và đồ họa .

Chúng ta còn gọi ảnh của vector x lên vector y khác không là vector

$$x_y = <x/y> = \frac{y}{< y / y >}$$

Vector này còn gọi là thành phần của x lên hướng của y.

Phần cơ bản của $R^N$ (hoặc $C^N$) gọi là trực giao cùng với điểm <1>, nếu dot product của mỗi phần tử của phần cơ bản là = 0 . Ví dụ , vector $i_1$, $i_2$, ..., $i_N$ được xác định là trực giao cơ bản (orthogonal basic) của $R^N$ (or $C^N$).

Có một nguyên tắc : Đối với các trực giao cơ bản của $R^N$ (hoặc $C^N$) , vector = tổng của các thành phần trên hướng của vector của phần cơ bản.

Trong các trường hợp khác, đặc tính này không thay đổi đối với phần cơ bản tự nhiên mà còn cho bất kỳ một trực giao cơ bản nào

Chúng ta biết rằng họ của N vector dài N $e_m = [e_m(h)]$ được xác định như sau

$$e_m(h) = \exp(2\pi i \frac{(m-1)(h-1)}{N}) \qquad 1 \le m \le N.$$

Hình dáng trực giao cơ bản đối với $C^N$ với ánh xạ đến điểm được xác định trước. Hơn thế nữa đối với các h, $\langle e_h | e_h \rangle = N$ . Để chứng minh điều này bạn có thể tìm thấy trong các sách về xử lý tín hiệu, như Oppenheim và Shafer (1975). Bạn có thể thay đổi kết quả sử dụng MATLAB, với N = 16.

Trong ma trận E chúng ta có thể xây dựng hàng thứ m biểu diễn vector $e_m$. Hãy đánh dòng lệch sau:

```
» N = 16 ;
» for m = 1 : N ;
        for n = 1 : N ;
                E(m, n) = exp(2 * pi * sqrt(-1) * (m-1) * (n - 1/N) ;
        end
end
```

Cho ta biết cấu trúc của ma trận D như các (dot product) điểm sinh ra của $e_h$ và $e_k$ trên vị trí (h, k). Thay đổi D được cho bởi MATLAB như sau

```
» D = E * E'.
```

Và bằng N * eye(N, N).

Biến đổi Fourier rời rạc X, của vector x chiều dài N, được xác định như

$x(h) = \langle x | e_h \rangle$. Bởi nơi X(h) được biểu diễn cho hệ số nhân, biên của x theo hướng của $e_h$ . Thao tác của phần x từ X, gọi là biến đổi Fourier ngược, mà được dùng trong MATLAB bởi hàm ***ift***. Như tần suất của khả năng tách ly (decomposition property), ***ift*** được biểu diễn thao tác

$$x = \sum_{h=1}^{N} \frac{X(h)}{N} e_h \qquad (1.17)$$

Ví dụ 1.2: Chúng ta thay đổi công thức (1.17) cho vector ngẫu nhiên của 128 điểm .

```
» N = 128 ;
» x = read (1, N) ;
» X = ift(x) ;
» t = (0 : (N - 1)/N ;
» for h = 1 : N
                    yy(h, :) = X(h)/N*exp(2*pi)2*pi*sqrt(-1)*(h-1)*t) ;
   end
» y = sum(yy) ;
```

Bây giờ chúng ta so sánh k và y, ở đồ thị khác

```
» plot (1 : N, x, 1: N, y)
```

Hoặc gọi số

```
» max (abs(x-y))
```

**Ví dụ 1.3.** Nói ngoài lề tác dụng của mã (code)

Vì kích cỡ lớn của một chuỗi, giải quyết vấn đề trong MATLAB trong vùng tín hiệu và xử lý ảnh cần có 1 kỹ năng rất cao. Có 2 thao tác: một có hiệu quả lớn đó là - vùng bộ nhớ động và vòng lặp (for). Vùng bộ nhớ động để chỗ cho việc cất ma trận, hoặc khi chúng ta tăng kích cỡ của một ma trận thoát ra. Điều đó đòi hỏi thời gian rất lớn. Lặp For cần phải trợ giúp việc xếp lại thao tác chuỗi như + , - : , .. với khả năng nhanh hơn ở biên.

Để mô tả những ý này chúng ta quay trở lại ở ví dụ 1.6 trong đó khi tăng ma trận E, chúng ta tăng hiệu quả của mã của chia vùng. Lời giải nhanh hơn nhận được nhờ vùng giải ma trận và loại bỏ 1 vòng với vector hoá. Viết một M-file và chạy chúng

```
N = 16 ;
P1 : zeros(N, N) ;
I : [0 : (N-1)]/N ;
for m = 1 : N
            P1(m, :) = exp(2*pi*sqrt(-1)*(m-1)*I) ;
end
```

Đánh

```
» pi1 - E
```

Bạn sẽ nhận được ma trận 0 khích thước 16 x 16.

Chúng ta cần dùng vòng lặp for để tính toán chuyển FFT của một ma trận

```
» P2 = fft + (eye(N ,N)) ;
```

Bạn có thể kiểm tra kết quả với

```
» P2 - E
```

Trong thời gian này, có giá trị 0 chúng ta có sai lệch rất nhỏ cho đến sai số bằng số. Có thể so sánh thời gian tính với giá trị N lớn hơn.

## 7. PHỔ NĂNG LƯỢNG

x là tín hiệu lặp lại theo thời gian với chu kỳ T, được lấy mẫu tại khoảng $T_S = T/N$, và X là biến đổi Fourier rời rạc.

Vector cơ bản $e_h$ được xác định ở phần trên $\exp(2\pi i \frac{h-1}{T} t)$ của tần số (h-1)/T Hz. Vector $((X(h)/N)e_h$ , là chiếu của x theo hướng của $e_h$. Đối với h > 1, vector này thường được xem như thành phần của x của tần số (h - 1)/THz. Đầu vào của vector x, như x(h) còn gọi là thành phần thứ h của x. Tổng trung bình của x là X(1)/N và có khi được gọi là thành phần DC của tín hiệu x , rất hay dùng bởi kỹ sư điện , $(x(2)/N)e_2$ và $(x(N)/N)e_N$, liên quan đến chu kỳ T, được gọi là các thành phần cơ bản của tín hiệu này.

Hình 1.12 chỉ ra quan hệ giữa các thành phần này của ***x = fft (s)*** và tần số của các thành phần theo x.

Đối với các h = 1, 2, ..., N -1 , nhóm x(1 + h) và x(1+N-h) được gọi là liên hợp . (xem thêm bài tập 1.1)

Chuỗi của các phần tử $|x(h)|^2 /N$ gọi là phổ năng lượng của x, như là đối với bình quân phương của tín hiệu hay phép nhân của nó, được thể hiện là năng lượng (công suất) . Trong trường hợp này, lý thuyết bền vững lại là quan hệ rất quan trọng .

MATLAB

Nếu x là rời rạc của tín hiệu có chu kỳ T tần số lấy mẫu là fs = N/T Hz và X là biến đổi Fourier.

| | |
|---|---|
| 1. X(1) | Liên quan đến DC của tín hiệu |
| 2. Đối với b ≤ N/2+1(X(b)) | liên quan với tần số b-1/T = b-1/N .$f_s$ Hz |

3. Với f ≤ fs/2 Thân số f Hz liên hệ với bin b = $\frac{N.f}{fs}$

***Hình 1.13 Mối quan hệ giữa tần số của các thành phần của tín hiệu và DFT (biến đổi Fourier rời rạc)***

Bình phương của biên của vector x, được xác định bằng

$$<x/x> = \sum_{h=1}^{N} (x_h)^2$$

Có thể tính bằng cách sử dụng biểu thức

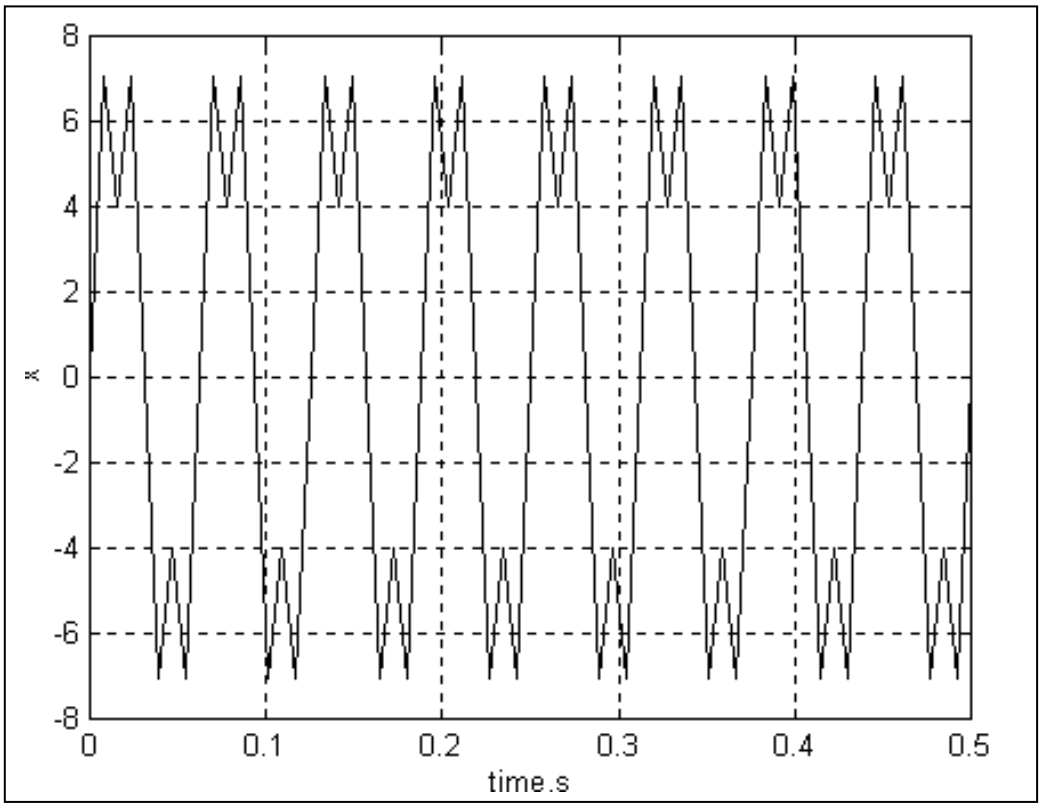
$$<x/x> = \sum_{h=1}^{N} \frac{(X_h)^2}{N} \qquad (1.19)$$

Như các thành phần hợp lại $<e_h/e_k> = \Phi$ khi $h \neq k$ khi đó nhóm $|X_h|^2/N$ thể hiện công suất của thành phần của k của tần số f = (h - 1)fs/N. Do đó biểu thức (1.19) chỉ ra công suất của tín hiệu  = tổng của các công suất của các thành phần .

Đó là 1 trong dạng của lý thuyết Parseval (1755 - 1836)

**Ví dụ 1.4**: Tách phổ tần số của tín hiệu

Chúng ta sẽ lấy tổng của hai ký hiệu tần số khác nhau và biên độ cũng khác nhau và cũng xem xét phổ năng lượng (Công suất) của nó như thế nào. Có một điều khó trong ví dụ hai này là giữ  đúng công suất ảnh hưởng ứng với tần số và làm sao cho biên của công suất ảnh hưởng đến biên độ $a_1 = 7$ và tần số $f_1 = 16$Hz và tín hiệu 2  $x_2$ biên độ $a_2 = 3$ và tần số $f_2 = 48$Hz tần số lấy mẫu là 128Hz và gọi tổng của chúng.




*Hình 1.14. Tổng của hai tín hiệu hình sin*

```
» N = 512;                          % số điểm
» b = 1 : N;                        % bins
» Ts = 1/128;                       % Khoảng lấy mẫu theo giây
» fs = 1/Ts;                        % Tần số lấy mẫu theo Hz
» ts = Ts x (b - 1)                 % Khoảng lấy mẫu
» a1 = 7  ;  f1 = 16;
» x1 = a1 * sin (2 * pi * f1 * ts) ;   % tín hiệu đều
» a2 = 3; f2 = 48
» x2 = a2 * sin (2 * pi * f2 * ts) ;   % tín hiệu thứ hai
» x = x1 + x2;
```

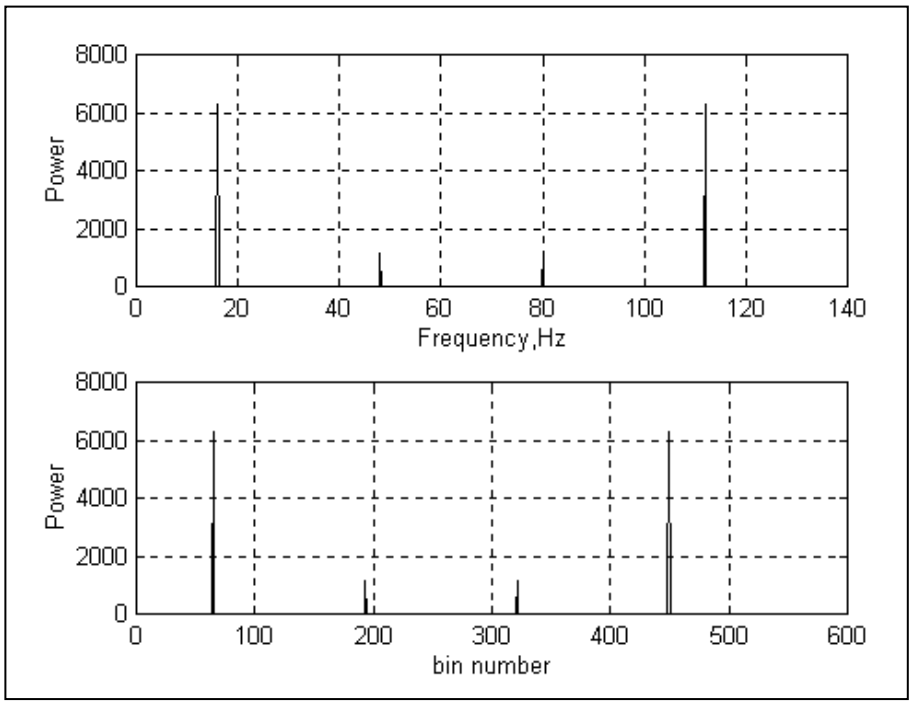
Nếu bạn nhìn thấy kết quả tín hiệu (hình 1.13) đưa lệnh sau vào

```
» plot (ts, x)
» xlabel ('Time, s'), y label ('x')
```

Chúng ta có thể xây dựng và chấm điểm của phổ công suất.

```
» X = fft(x);                       % DFT của x
» pwr = x * cosj (X) / N            % Công suất của tín hiệu
» frs = (b = 1), N * fs             % Các tần số
» Plot (frs, pwr)                   % chấm điểm phổ công suất
```

Kết quả chấm điểm ở trên hình 1.14. Tín hiệu $x_1$ đạt công suất ở điểm 65, với tần số $f_1 = 16H_2$ (65 = 1 + 512 x 16 / 128) và ở bin 449, phần chậm hơn vì liên hợp của số tín hiệu 65 = 1 + 65 được cất trong bin 449 = 1 + 512 - 64




*Hình I.15 Phổ năng lượng của tín hiệu x=x1+x2*

Phổ công suất được chỉ ra trên hình 1.14. Ta có thể kiểm tra **pwr (65) = $(a_1/2)^2$.N**. Tương tự như vậy đối với tín hiệu $x_2$. Công suất ở bin 193 và 321 và

**pwr (193) = pwr (321) = $(a_2/2)^2$ N**

**Ví dụ 1.5**: Nhận dạng tần số và thành phần công suất chính

Trong thí dụ này chúng ta sẽ phân tích tín hiệu tam giác của chu kỳ S = 5 giây và điểm nhảy biên độ 1 vào thành phần tần số của chúng sử dụng 512 điểm lấy mẫu. Chúng ta quan tâm đến việc tìm phần trăm nào của công suất tổng là thành phần trong tín hiệu được nhận từ gốc, bằng cách tách các thành phần từ 4 thành phần. Chúng ta còn muốn biết bằng cách nào làm xấp xỉ tín hiệu với tín hiệu chuẩn. Đầu tiên, chúng ta xây dựng phương án rời rạc x của tín hiệu bằng lấy mẫu nó tại 512 điểm bằng nhau.

```
» T = S;
» N = 512;
» t = linspace (0,T, N + 1) ; t = (1 : N);
» x1 = 2 * t/T - 1/2 ;    x2 = 2*(T - t) / T - 1 / 2;
» x = min (x1, x2);              % tín hiệu tam giác và xây dựng phổ công suất của chúng:
```

```
» b = 1 : N                    % Khoảng lấy mẫu và tần số
» X = fft (x);
» Ts = T / N ; fs = N/T % bằng (b - 1) / N * fs
» prw = X * conj (X) / N;
```

Để kiểm tra kết quả của chúng ta, chúng ta có thể dùng đẳng thức Parseval. Những số sau phải bằng

```
»[sum (pow) norm (x)^ 2]
ans =
        42.6680        42.6680
```

Dễ dàng nhận thấy các tần số này gồm thành phần lớn nhất của công suất, sử dụng hàm ***sort*** với quay trở lại các phần tử của ***pow*** bằng cách tăng điểm:

```
» [spow, spos] = sort (pow);
```

Chúng ta tìm chữ số của 4 tần số thành phần công suất lớn nhất:

```
» m = 4; spos (N: -1 : (N - m + 1)
```

Chúng ta có thể thấy các tần số này cấu thành trên 512, 2, 510 và 4. Bây giờ chúng ta xây dựng tín hiệu xấp xỉ

```
» X4 = zesos (X);              % Vùng đổ xấp xỉ X
» h = [512  2  510  4];
» X4 (h) = X (h);              % chép binh cầu thành công suất cao
```

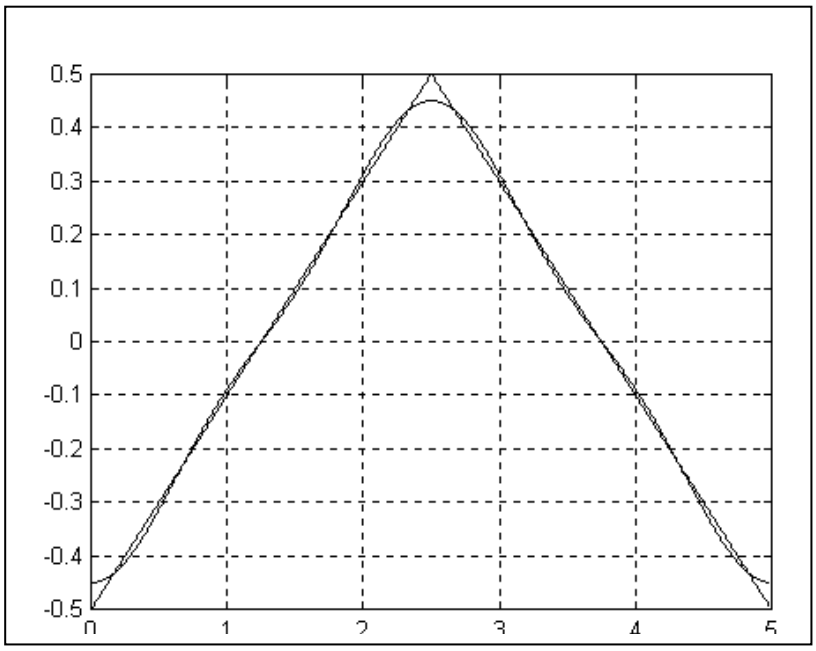
Phần trăm của công suất tạo thành trên 4 thành phần chỉ đạo được đưa ra bởi

```
» pere = 100 * (norm (X4) / norm (X))^2
```

Kết luận, 99,7698 % của công suất được tạo thành trên 4 nhóm, tương ứng với tần số cơ bản 0.2 Hz; liên quan đến bin số 2, tần số truyền đạt của nó liên quan đến bin 512, giao động thứ 2, 0.6 Hz liên quan đến bin số 4, và hệ số truyền của nó, liên quan đến bin 510. Chúng ta sẽ sử dụng kết qủa trong ví dụ 1.8

Những dòng sau sẽ chỉ ra làm thế nào tiến gần đến tín hiệu tam giác góc được xấp xỉ, xem hình 1.15

```
» x4 = ift (X4);
» plot (f, [x; x4]) ,  grid
» xlabel (‘ t ’), ylabel (‘ Tín hiệu tam giác và sự xấp xỉ chú ý)
```




*Hình 1.16. Sự xấp xỉ của tín hiệu hình tam giác*

## 8. PHẦN LƯỢNG GIÁC MỞ RỘNG CỦA TÍN HIỆU

Mục đích của phần này là chỉ ra làm thế nào vesion rời rạc của tín hiệu, chu kỳ T và lấy mẫu ở khoảng Ts = T/N, có thể nhanh chóng tổ hợp tuyến tính của hình sin và cosin theo dạng sau

$$x = \sum_{b-1}^{N} (Ah\cos(2\pi(h-1)\frac{t}{T}) + B_h \sin(2\pi(h-1)\frac{t}{T}) \qquad (1-20)$$

Đối với t của Ts chúng ta nhìn thấy rằng nếu X đánh dấu chuyển đổi Fourier của x, đẳng thứ nhất (1.17)

$$\text{x} = \sum_{h=1}^{N} \frac{X(h)}{N} cxp(2\pi i(h-1)\frac{1}{T} \qquad (1-21)$$

Sử dụng cách Euler, và gọi R và I tương ứng phần thực và phần ảo của X, đẳng thứ (1 - 21) sẽ được lại như sau.

$$x = \sum_{h=1}^{N} (\frac{R_h}{N}\cos(2\pi(h-1)\frac{t}{T}) - \frac{I_h}{N}\sin(2\pi(h-1)\frac{t}{T})) + i\sum_{h=1}^{N} (\frac{R_h}{N}\sin(2\pi(h-1)\frac{t}{T}) - \frac{I_h}{N}\cos(2\pi(h-1)\frac{t}{T}))$$

Đồng nhất thật đúng đối với mỗi x, nhưng có thể làm đơn giản hoá khi x là số thực. Trong trường hợp đó, chúng ta biết ưu tiên là thành phần ảo của đẳng thức (1.22), phải triệt tiêu, dùng đồng nhất thức (1.20) cho

$A_h = R_h / N$

$B_h = - I_h / N$ và h chạy từ 1 đến N

Biểu thức (1.20) được gọi là lượng giác mở rộng của x

**Ví dụ 1.6:**

Trong ví dụ sau chúng ta sẽ biến đổi biểu thức (1.20) cho năm giây và vector ngẫu nhiên của 128 nhóm.

```
» T = 5 ;                          % Khoảng thời gian, giây
» N = 128;                         % Chiều dài của vector
» t = linspace (0, T, N + 1);
» t = t (1 : N);                   % thời gian lấy mẫu
» x = rand (t);                    % vector ngẫu nhiên
» X = stt (x);                     % DFT của  nó
»A = real (X) / N;                 % Hệ số cosine
»B = -imag (X) / N;                % Hệ số sin
»sum cos      Zeros (N, N);
»for h = 1 : N
        sumcos (h : ) = A (h) * cos (2 * pi * (h - 1) * t/T);
        sumsin (h, ☺= - B (h) * sin (2 * pi * (h - 1) * t/N);
end
» y = sum (sumcos * sumsin);
```

Bây giờ so sánh x và y, đồ họa của chúng

```
» plot (t, x, t, y)
```

hoặc tính số

```
» Max (abs (x - y))
```

Trong version của chúng ta MATLAB có kết quả là 2.142e - 19

**Ví dụ 1.7**: **Phân tích lượng giác của tín hiệu tam giác**

Bây giờ chúng ta muốn phân tích tín hiệu tam giác x tính trong ví dụ 1.5 trong thành phần lượng giác của nó và kiểm tra kết quả. Nếu chữ số N = 512 xuất hiện trong

nhóm tiếp theo của lệnh thì rất lớn cho bộ nhớ của máy tính của bạn, bạn có muốn giảm nó thành số nhỏ, như 32

```
» T = 5;
» N = 512;
» t = linspace (0, T, N + 1); t = (1 : N);
» x1 = 2 * t / T - 1/2 ; x2 = 2 * (T - t) / T - 1/2;
» x = min (x1, x2);                          % tín hiệu tam giác
» plot (t, x)
```

Chúng ta tính hệ số của sines và cosine.

```
» X = fft (x);
» A = real (X) / N;                          % hệ số cosine
» B = - imag (X) / N);                       % hệ số sine
» sumcos = zeros (N, N);
» sumsin = zeros (N, N);
» for h = 1 : N
        sumcos (h, : )  = A(h) * cos (2 * pi * (h - 1) * t/T);
        sumsin (h, : ) = B (h) * sin (2 * pi * (h - 1) * t/T);
end
» y = sum (sumcos + sumsin);
```

Chúng ta có thể kiểm tra các kết quả bằng cách so sánh x và y, đồ họa của chúng

```
» plot (t, x, t, y);
```

và số

```
»max (abs (x - y))
```

## 9. NHỮNG TÍN HIỆU TẦN SỐ CAO VÀ KÝ HIỆU

Ở hình 1.12 đã chỉ ra sự tương ứng giữa công suất của tín hiệu và biến đổi Fourier của nó đối với các tần số đến tần số Nyquist. Điều này trở nên thú vị để xem điều gì xảy ra khi chúng ta lấy mẫu tại khoảng thời gian Ts hằng số tín hiệu tuần hoàn liên tục của tần số cao đến tần số Nyquist $N_f = 1/ (2Ts)$. Như chúng ta nhìn thấy ở đây, version lấy mẫu của tín hiệu đồng nhất với tín hiệu khác tần số thấp. Hiện tượng này gọi là dấu hiệu từ $C_1$ , từ ý nghĩa Latin "other", những cái khác. Để

nhấn mạnh ý này chúng ta chọn T là 5 giây, N = 16 lấy mẫu trong một chu kỳ, và hiện ra theo khoảng lấy mẫu với Ts = T/N và tần số mẫu với fs > 1/Ts.

Tín hiệu tuần hoàn với chu kỳ T có chu kỳ cơ bản của T nối T/k với k phù hợp. Chúng ta chỉ ra tần số của nó k/T, với f nhỏ. Cũng như tín hiệu, cho khoảng cách sin (2πft) và cos (2πft). Tần số f có thể luôn viết như sau

**f = fapp + nfs**

Trong đó n và số nguyên và $0 \le |f_{app}| < N_f$. Nó dễ dàng kiểm tra rằng tại các t bội số của Ts như sau **t = hTs, sin (2πft) = sin (2πf$_{app}$t).** Thực tế

**sin(2πft) = sin (2π (f$_{app}$ + nfs) t)**

= sin (2π (f$_{app}$ + nfs) hTs)

= sin (2π (f$_{app}$ hTs + 2πnfshTs)

= sin (2π (f$_{app}$ hTs + 2πnh)

= sin (2π f$_{app}$)

Song tín hiệu $x = sin(2\pi ft)$, tần số f, khi lấy mẫu ở tần số fs, là không thể phân biệt được từ tín hiệu $x_1 = \sin(2\pi f_{app} t)$ của tần số thấp $f_{app}$ MATLAB cho phép chúng ta giải quyết vấn đề và biểu diễn các dấu hiệu. *Hãy dùng m tệp sau; alias.m:*

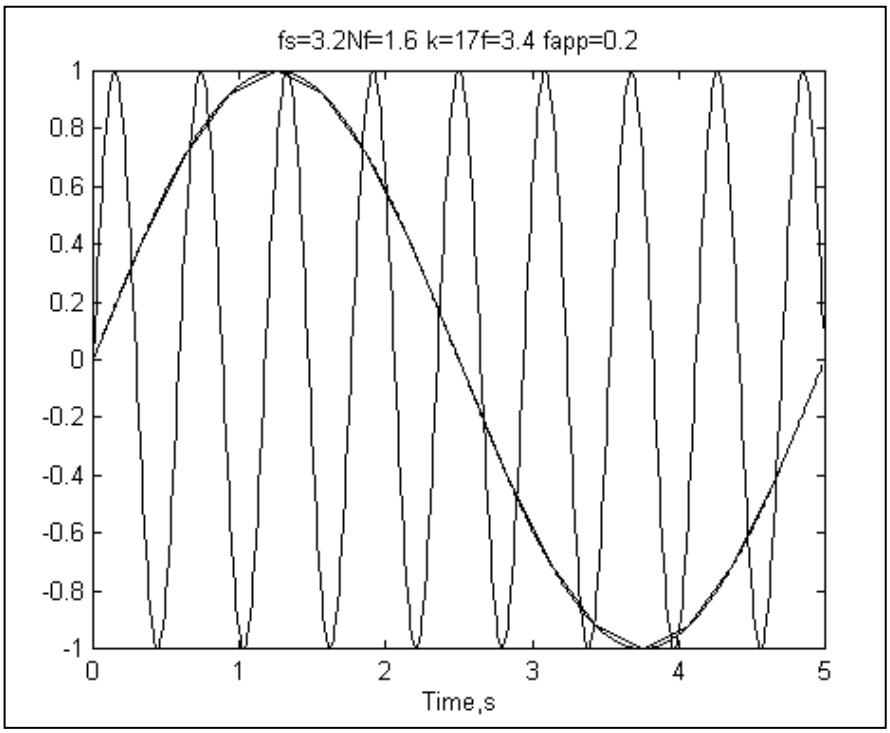
```
T = 5 ;                          % tần  số cơ bản
Np = 512;                        %Số điểm để chấm
t = linspace(0,T,Np+1;
t = t(1:Np);                     % tìm  độ phân giải của thời gian
                                 %để chấm điểm
N=16;                            % số điểm lấy mẫu
Ts =T/N;                         % khoảng lấy mẫu
fs =1/Ts;                        % tần số lấy mẫu
ts = Ts*(0:(N-1));               % khoảng thời gian lấy mẫu
Nf = 1/(2*Ts);                   % Tần số Nyquist

f = k/T;                         % tần số liên tục
                                 % tín hiệu
x = sin(2*pi*f*t);               % tín hiệu, độ phân giải cao
xs = sin(2*pi*f*ts);             % tín hiệu, lấy mẫu phân giải
% tìm fapp, như sau: f =n*fs+fapp
n = round(f/fs);
```

```
fapp = f-n*fn;
xa = sin(2*pi*fapp*t);
plot(t,[x;xa],ts,xs,'0');
str1 = ['fs = ', num2str(fs), 'Nf = ',num2str(Nf)];
str2 = ['k = ', num2str(k), 'f = ',num2str(f)];
str3 = [fapp=',  num2str(fapp)];
str = [str1, '   ',str2, '   ', str3];
title(str);
```

Chạy chúng với lệnh sau

```
» k= 17;          alias
```




***Hình 1.17 tín hiệu tần số cao lấy mẫu như một tần số thấp.***

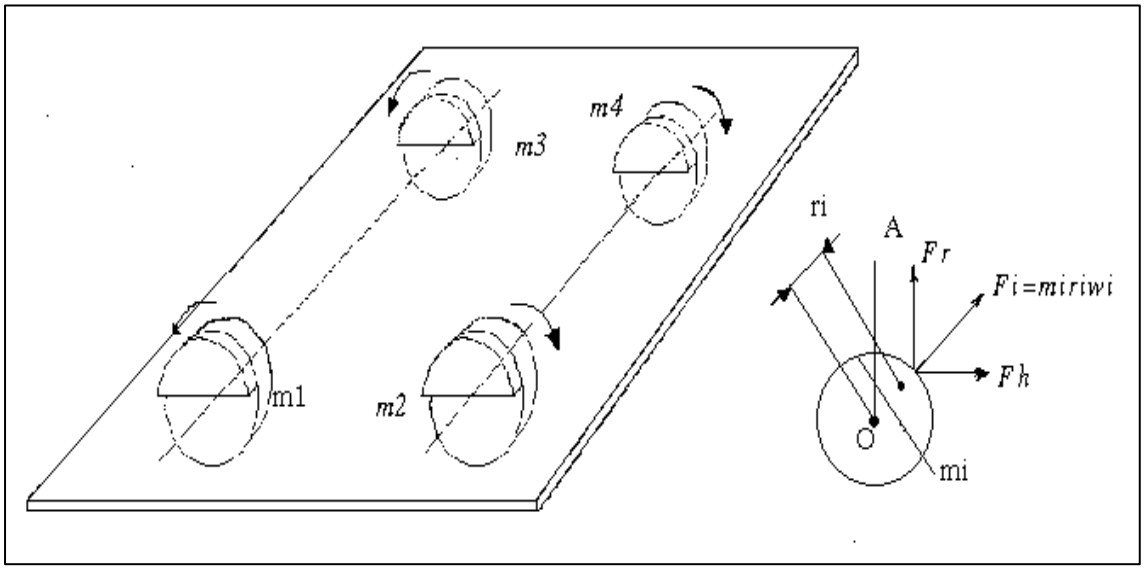
**Ví dụ 1.8:** Giao động của một tấm

Việc tính toán ở ví dụ 1.5 và 1.7 có thể có một ứng dụng kỹ thuật mô tả trong: Máy kiểm tra giao động. Ví dụ đơn giản có dạng như hình 1.17. Các bộ phận hoạt động của máy là 4 trục quay, không có khối lượng giao động $m_1$ đến $m_4$. Như mô tả ở trên hình1.17 (a), khối lượng không giao động có thể đoạn của vòng trong làm bằng sắt (thép) và tựa trên đĩa quay. Khối lượng $m_1$ và $m_2$ bằng nhau, nhưng quay theo hai hướng

đối nhau, và cũng như vậy đối với khối lượng $m_3$ và $m_4$. Một trong những bộ phận được chỉ chi tiết trên hình 1.17 (b). Cho rằng khoảng cách giữa trục quay qua điểm 0 và tâm của khối lượng không giao động, $m_i$ là $r_i$ . Giả sử khối lượng quay quanh điểm 0 với tốc độ $\omega_i$. Lực hướng tâm đặt vào tâm của khối lượng không giao động bằng $\boldsymbol{F_i = m_i r_i \omega^2_i}$. Nếu chuyển động bắt đầu từ trục thẳng đứng OA và hướng quay theo chiều kim đồng hồ, sau thời gian t góc giữa OA và hướng của $\boldsymbol{F = \omega_i t}$. Thành phần thẳng đứng của lực hướng tâm là $\boldsymbol{F_v = m_i r_i \omega^2_i . cos \omega_i t}$, và thành phần nằm ngang là $F_h = m_i r_i \omega^2_i . sin\ \omega_i t$ . Đối với khối lượng bên phải ở đây bằng khối lượng mà quay hướng ngược, bắt đầu từ trục đứng.

Nó sẽ đặt lực hướng tâm khi mà thành phần thẳng đứng = $F_v$, khi thành phần ngang = $-F_h$. Thành phần nằm ngang giao động quanh điểm, khi thành phần thẳng đứng lên cao, *sinh ra lực đàn hồi* $\boldsymbol{= 2\ m_i r_i \omega^2_i\ .\ cos\ \omega_i t}$. Điều quan trọng là lực này và thành phần của chúng, sản phẩm $\boldsymbol{m_i r_i}$ biểu diễn mômen tĩnh của khối lượng theo trục quay.

Nếu hai cặp đếm khối lượng quay sắp xếp trên cùng một bàn đàn hồi và *tỉ số giữa môment và góc quay của chúng có thể tính (gần đúng),* thì có thể tổng hợp được các xung đàn hồi của các hình dạng khác nhau. Chúng ta hãy thử xấp xỉ dạng sóng được phân tích trong ví dụ 1.5 và 1.7. Chúng ta gọi cho 4 thành phần tạo nên năng lượng chủ yếu. Đó là giao động đầu tiên với tần **số *0.2 Hz***, và liên kợp của nó, giao động thứ 3, tần số 0.6 Hz, và liên hợp của nó. Liên hợp tương ứng theo chiều ngược lại với các tần số ***0.2 Hz và 0.6 Hz***. Điều đó có nghĩa là cặp khối lượng không giao động quay theo hướng ngược lại như hình 1.17, sẽ sinh ra lực tương ứng với cặp liên hợp trong phần lượng giác mở rộng của lực. Biên độ của các thành phần tỷ lệ theo hệ số với lượng giác mở rộng. Chúng bằng 0.2026N cho tần số 0.2 Hz và -0.2Hz và 0.0225N cho tần số 0.6Hz và -0.6Hz.




***Hình 1.18. Máy kiểm tra rung động***

Chúng ta bắt đầu thiết kế máy đàn hồi bằng cách đưa vận tốc góc của khối lượng không giao động theo rad/s.

```
» omega 1 = 2 * pi * 0.2 , omega2 = 2 * pi * 0.6
   omega 1 =
                1.2566
   omega 2 =
                3.7699
```

Tiếp theo chúng ta đưa biên của lực được sinh ra bởi trọng lượng không giao động

```
» F1 = 0.2026 ; F2 = 0.0225;
```

Môment trọng lượng, $m_1r_1$, $m_2r_2$ (kgm), sinh ra những lực sau

```
» r1m1 = F1 / omega1 ^ 2
   r1m1 =
      0 . 1283
» r2m2 = F2 / omega 2 ^ 2
   r2m2 =
      0.0016
```

Chúng ta giả thiết là khối lượng không giao động là 1 đoạn của vòng tròn dày 0.02m, làm bằng thép có khối lượng riêng 7850 kg/$m^3$. Môment tĩnh của vùng segment (tính ra $m^3$) là

```
» S1 = r1m1 / (0.02 * 7850)
   S1 =
      8.1718 e - 04
»S2 = r2m2 / (0.02 * 7850)
   S2 =
      1.0084 e - 05
```

Điều này có thể chỉ ra rằng môment này của vùng segmen của vòng tròn phụ thuộc vào tổ hợp của nó t và = $t^3$ / 12. Dùng công thức sau để tính tổ hợp của segment của vòng, theo m,

```
»  t1 = (12 * S1)^ (1/3)
t1 =
      0.2140
» t2 = (12 * S2)^ (1/3)
```

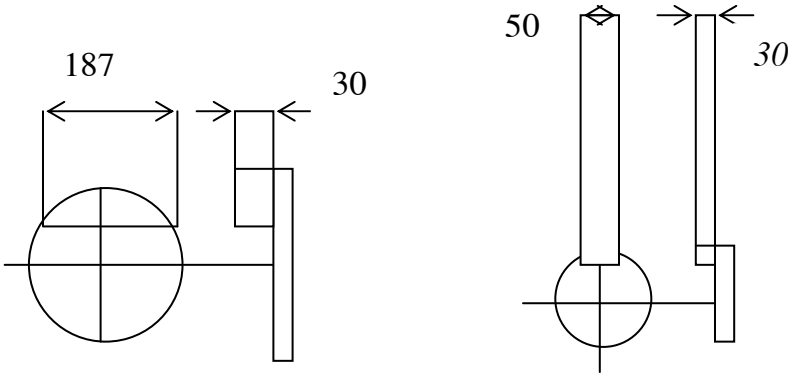
```
t2 =
        0.0495
```

Chúng ta kiểm tra nếu giảm hợp của khối lượng m1, m2 bằng cách tăng chiều dày của chúng đến 0.03m:

```
» S1 = r1m1 / (0.03 * 7850)
 S1 =
        5.4479 e - 04
» t1 = (12 * S1)^ (1/3)
t1 = 0.1870
```

Chúng ta sẽ chỉ ra thành khối lượng giao động thiết kế theo việc làm sinh ra lực thẳng đứng khi đồ thị thời gian xấp xỉ hình tam giác. Bắt đầu bằng việc xác định trục thời gian.

```
» t = 0; 0.02 : 10;
```




***Hình 1.19 Kích thước của khối lượng không giao động và tiếp tục viết các điều hoà chính***

```
» f1 = 2* r1m1 * omega1 ^ 2 * cos (omega1 * t);
» f2 = 2 * r2m2 * omega2 ^ 2 * cos (omega2 * t);
```

Chấm các điểm nhận được

```
» plot (t, (f1 + f2)
» grid
» title (‘Tổng hợp lực đàn hồi hình tam giác')
» x label ( ‘ t, S’)
» y label (‘ F, N’)
```

Thử kiểm tra trên hình vẽ chu kỳ của sóng tam giác là năm giây và biên độ của lực đàn hồi là 0.45N, gần với 0.5N. Như hình 1.19.

# PHẦN BÀI TẬP

## 1) Mệnh đề liên hợp

a/ Thay đổi những mệnh đề sau cho vector x xác định trong MATLAB bởi N = 128; x = rand (1, N);

Nếu x là số thực có chiều dài N và x là biến đổi Fourier rời rạc, đối với mỗi h trong khoảng [1, N - 1], x (1 + N - h) là số phức liên hợp của x (1 + h)

b/ Nếu bạn theo hướng toán học, chứng minh mệnh đề cho mỗi vector thực x.

Giả thiết là $x = x$ và $e_{1+h} = e_{1+N-h}$

## 2) Xác định đặc tính tần của bộ lọc

Một số hàm của MATLAB như ***yulewalk*** và ***remez***, xây dựng các hệ số của bộ lọc số như thế này; xấp xỉ với tần số mô tả các tính chất

a/ Xây dựng hàm ***deffiltm.m*** cho phép người dùng xác định đặc tính tần của bộ lọc khi nháy vào điểm trên biên plane tần số với chuột và quay về chuỗi của tần số không thứ nguyên (như các tần số qui chuẩn vơí tần số Nyquist), $f_0$, và biên M.

b/ Kiểm tra hàm số. Xác định ý nghĩa của ***deffilt.m*** được xây dựng ở (a) của bộ lọc số này, trên tần số lấy mẫu tại 100Hz (tức là tần số Nyguist là 50Hz), có những đặc tính sau:

| Biên | Tần số |
|---|---|
| 1.0 | 0 |
| 1.0 | 10 |
| 0.5 | 20 |
| 0.5 | 30 |
| 1.0 | 40 |
| 1.0 | 50 |

## 3) Mô tả IIR - yulewalk

Tín hiệu được lấy mẫu tại 800Hz. Chúng ta muốn dùng hàm ***yulewalk*** để thiết kế bộ lọc IIR với xấp xỉ bộ lọc F, xác định bởi đặc tính tần sau:

| Từ Hz | Đến He | Biên |
|---|---|---|
| 0 | 100 | 0 |
| 100 | 150 | Tăng tuyến tính từ 0 đến 2 |
| 150 | 180 | 2 |
| 180 | 200 | Giảm đều từ 0.5 |
| 200 | 240 | 0.5 |
| 200 | 300 | Tăng đều từ 0.5 đến 1 |
| 300 | 400 | 1 |

a/ Viết chuỗi $f_0$ và $m_0$ để xác định đặc tính củ bộ lọc từ yêu cầu bằng ***yulewalk***

Hint: Viết tần số như bội của tần số Nyquist.

b/ Thay đổi lời giải đúng vào (a) bằng chấm điểm $m_0$ ***versus*** $f_0$.

c/ Sử dụng hàm ***yulewalk***, tìm các hệ số của bộ lọc cho 6 điểm, 8, 10 bằng cách xấp xỉ bộ lọc đã đưa ra.

d/ So sánh đặc tính đồ hoạ của bộ lọc nhận được với F.

## 4) Kiểm tra bộ lọc với đầu vào hình sin

Khi giải bài (3) bạn có chuỗi

bIIR6 = [0.51 69 - 0.7337 0.6589 - 0.69890.4929 - 0.1354 0.1355]

aIIR6 = [10000 - 0.3217 1.2452 - 0.089 0.5872 - 0.0185 0.1643]

biểu diễn các hệ số của bộ lọc số. Nếu bạn không cất chúng thì hãy đưa vào bằng tay. Bạn muốn kiểm tra rằng bộ lọc này có đặc tính tần yêu cầu bằng cách kiểm lại nó theo số điền vào hình sin, như sau

(a) Xây dựng phần rời rạc của tín hiệu s = sin (2πft) lấy mẫu ở 800Hz trong thời gian 1giây, đối với f = 100Hz.

(b) Dùng hàm lọc filter qua S , qua bộ lọc xác định với hệ số của bIIR6 và aIIR6 và gọi kết quả fs. Chấm điểm fs như một hàm thời gian, đối với t ttrong khoảng [0.5, 0.6], sau thời gian đủ cho có tác động của trạng thái sẽ xoá.

(c) Kiểm tra ở các bộ lọc có đặc tính tần rời rạc

(ví dụ tín hiệu 100Hz, chính xác 0,5)

(d) Thay đổi bộ lọc có đặc tính tần như (3)

(e) Lặp lại câu c cho tần số 100, 150, 180, 200, 240 và 300 Hz

## 5) Thiết kế FIR - remez

Tín hiệu lấy mẫu tại 400Hz. Chúng ta muốn sử dụng hàm ***remez*** để thiết kế FIR lọc số cùng với hàm lọc F xấp xỉ, xác định bởi đặc tính tần như sau:

| Từ Hz | Đến Hz | Biên |
|---|---|---|
| 0 | 25 | 1 |
| 25 | 50 | Giảm tuyến tính từ 1 đến 0 |
| 50 | 100 | 0 |
| 100 | 150 | Tăng tuyến tính từ 0 đến 1 |
| 150 | 200 | 1 |

a/ Viết hai chuỗi $f_0$ và $m_0$ để xác định đặc tính cảu bộ lọc trên dựa vào ***remez***.

b/ Thay đổi cho đúng với lời giải (a), bằng cách chấm điểm $m_0$ theo $f_0$

c/ Sử dụng hàm ***remez*** tìm hệ số của bộ lọc cho 10 điểm, 20, 30 (gọi chúng tương ứng với bFIR10, bFIR20, bFIR30) và xấp xỉ bộ lọc được đưa ra.

d/ So sánh đồ hoạ đặc tính của bộ lọc nhận được với F

## 6) Hàm Bilinear cùng với tính toán khoảng lấy mẫu

Hàm chuyển đổi của bộ lọc thông thấp được xác định theo mặt s như sau:

$$H(s) = \frac{1}{1+s/p} \cdot \frac{w^2 n}{w_n^2 + 2\xi w_n s + s^2}$$

với P = 6 rad/s; $w_n$ = 15rad/s và $\xi$ = 0.6

a/ Hãy viết H(s) như là tỷ số của 2 đa thức ***num*** và ***den*** tính nhanh chúng nhờ sử dụng chuyển đổi MATLAB (chuỗi các hệ số của việc giảm năng lượng của s)

Yêu cầu : Sử dụng hàm ***cour*** .

b/ Chấm các điểm biên đáp ứng của bộ lọc được xác định bởi hàm tỉ số H(s), trong khoảng 0.1 đến 100 rad/s, bằng biện thị các tần số theo thang logarithm và biên theo dB.

c/ Chấm điểm đáp ứng pha trên cùng một khoảng, bằng biện thị tần số theo rad/s và pha theo độ (sử dụng ***unwarp*** nếu có khả năng cài đặt vào máy của bạn).

d/ Tìm các hệ số của số tương đương của nó, lấy tổng các tần số lấy mẫu của 50Hz

Chú ý: Sử dụng hàm **bilinear.**

## 7) Xấp xỉ khoảng lấy mẫu cho hàm bilinear

Hàm biến đổi của bộ lọc thông thấp, xác định theo s như sau:

$$H(s) = \frac{1}{1 + s/p} \cdot \frac{w^2 n}{w_n^2 + 2\xi w_n s + s^2}$$

với P = 8rad/s, $w_n$ = 20rad/s và $\xi$ = 0.65. Bạn muốn có bộ lọc số tương đương và muốn xác định tần số lấy mẫu với sử dụng help của MATLAB. Viết mẫu Chương trình bằng MATLAB sau:

a/ Nhắc người sử dụng hàm số lấy mẫu fs;

b/ Sử dụng hàm **bilinear**, cấu tạo bộ lọc số tương đương ứng với fs1

c/ Hiển thị ra đặc tính và pha của bộ lọc tương tự và bộ lọc số tương đương.

d/ Lặp lại bước (a), (b), (c) cho đến khi người dùng thoả mãn, lúc đó thoát ra khỏi vòng lặp.

e/ Dựa trên vòng lặp trên hiện thị tần số lấy mẫu và các thông số của bộ lọc số.

f/ So sánh đồ thị của đáp ứng của bộ lọc số được xây dựng ở (d) với bộ lọc tương tự đầu tiên.

## 8) FFT và bins.

X là 256 điểm FFT, có được bằng DFT đến chuỗi x có 256 điểm, có được bởi tần số của tín hiệu lặp lại tại 64Hz. Nhìn vào phổ công suất, chúng ta thấy có nhảy ở bin thứ 33. Điều đó chỉ ra khả năng thành phần của tần số lên tín hiệu gốc (có nghĩa là dấu hiệu có thể được thực hiện)

dùng công thức ở hình 1.12

## 9) Aliasing (ký tự)

Giả sử có tín hiệu được xác định ở bài 8. Nếu ký tự là hiện tại, với các tần số khác có cùng bước nhảy không?

## 10) Phổ công suất của tín hiệu tam giác.

Tín hiệu lặp lại hình tam giác được lấy mẫu tại 256Hz theo thời gian khoản 0.5 giây, sinh ra chuỗi x có 4 điểm của giá trị 0 với 16 điểm giá trị 1.

a/ Chấm điểm x như một hàm theo thời gian, mỗi khoảng 0.5 giây.

b/ Tính toán chấm điểm phổ tần của x

c/ Chỉ ra 5 tần số đầu tiên nó gồm những công suất lớn nhất .

*Chú ý*: Sử dụng hàm ***sort***

d/ Xấp xỉ x bởi 5 giao động (gọi kết quả x ppr5) và chấm điểm x và ***xapprs***.

## 11) Lọc và tín hiệu

Bộ lọc thông thấp F được xác định với chuỗi của các hệ số a và b có được bằng cách sử dụng lệnh MATLAB.

```
» [b, a] = butter (5, 0.5)
```

Tín hiệu x, được xác định như sau:

```
» Ts = 1/100 ; t = Ts * (1 : 500);
» f = 25 ; x = sin 92 * pi * f * t);
```

Có sin 2πft và được lấy mẫu lại 100Hz trong đoạn [0, 5] sử dụng hàm ***filter*** mà không xác định giới hạn trạng thái zi (có nghĩa là MATLAB tự xác định 0), tìm tín hiệu y, có được bởi cho x qua bộ lọc F, và nhận dạng hiệu quả ưu tiên bằng cách chấm điểm y như hàm của trung đoạn [0,0.2).

## 12) Bộ lọc với mô tả trạng thái tới hạn.

Bộ lọc thông thấp F được xác định bằng 1 chuỗi của các hệ số a và b có được nhờ sử dụng dòng lệnh của MATLAB

```
» [b, a] = butter (5, 0.5)
```

Tín hiệu x được xác định như sau.

```
» Ts = 1/100 ;f = 25  ;        x = sin (2 + pi * z * Ts * (1 : 500) ;
```

Tìm vector Z; như lệnh sau

```
»y = filter (b,a,x,zi)
```

Sinh ra vector y bắt đầu với 5 zeros. Kiểm tra câu trả lời bằng cách chấm điểm bắt đầu 30 nhóm của y.

*** Chú ý**: Dùnghàm ***filteric.***

# CÁC HÀM THƯ VIỆN THÔNG DỤNG TRONG TOOLBOX - DSP XỬ LÝ TÍN HIỆU SỐ

## 1. HÀM SINH RA CÁC DẠNG SÓNG

| | |
|---|---|
| **Chirp** | Phát hàm cosin |
| **Diric** | hàm tuần hoàn sinc |
| **Gauspull** | Phát xung Gaussian |
| **Pulstran** | Phát một dãy xung |
| **Rectpuls** | Phát hình vuông lấy mẫu không tuần hoàn |
| **sawtooth** | Hàm răng cưa |
| **sinc** | Hàm sinc hoặc sin(pi*x)/(pi*x) |
| **square** | Hàm sóng bình phương |
| **tripuls** | Máy phát hình thang lấy mẫu không tuần hoàn |

## 2. PHÂN TÍCH BỘ LỌC VÀ THỰC HIỆN CHÚNG

| | |
|---|---|
| **Abs** | Giá trị tuyệt đối của số ảo |
| **Angle** | Góc pha |
| **Conv** | quay |
| **Fftfilt** | Thực hiện bộ lọc over lap-add |
| **Filter** | Thực hiện bộ lọc |

| | |
|---|---|
| **filtfilt** | Bộ lọc pha không |
| **filtic** | Bộ lọc xác định điều kiện đầu |
| **freqs** | Biến đổi Laplace tần số đáp ứng |
| **freqspace** | Đặt tần số cho đáp ứng tần số |
| **freqz** | Biến đổi z tần số đáp ứng |
| **grpdelay** | Một nhóm trễ |
| **impz** | Đáp ứng xung (rời rạc) |
| **latcfilt** | Thực hiện bộ lọc Lattice |
| **unwrap** | Không bó pha |
| **upfirdn** | Bộ lọc FIR không lấy mẫu, lấy mẫu xuống |
| **zplane** | Chấm điểm cực rời rạc |

## 3.CÁC BIẾN ĐỔI HỆ TUYẾN TÍNH

| | |
|---|---|
| **Convmtx** | Ma trận quay( Ma trận chuyển vị, hay nghịc đảo) |
| **latc2tf** | Lưới và hoặc lưới bậc thang để truyền hàm chuyển đổi |
| **poly2rc** | Đa phương đến hệ số biến đổi |
| **rc2poly** | Hệ số phản xạ để biến đổi đa phương |
| **residuez** | Miền mở rộng thập phân của biến đổi z |
| **sos2ss** | Chuyển đổi các vùng thứ hai đến đặt trạng thái chuyển đổi |
| **sos2tf** | Chuyển đổi các vùng thứ hai để truyền hàm chuyển đổi |
| **sos2zp** | Chuyển đổi các vùng thứ hai đến trường không |
| **ss2sos** | Đặt trạng thái để đạt điểm thứ hai của vùng chuyển đổi |
| **ss2zp** | Đặt trạng thái đến chuyển đổi trường không |
| **ss2tf** | Đặt trạng thái để truyền hàm chuyển đổi |
| **tf2latc** | Truyền hàm đến lưới hoặc chuyển đổi lưới hình thang |
| **tf2ss** | Truyền hàm đến chuyển đổi trạng thái |

| | |
|---|---|
| **tf2zp** | Truyền hàm đến vùng chuyển đổi trường không |
| **zp2sos** | Chuyển đổi từ trường không đến vùng đặt thứ hai |
| **zp2ss** | Chuyển đổi từ trường không đến điểm trạng thái |
| **zp2tf** | Chuyển đổi từ trường không đến hàm truyền |

## 4. THIẾT KẾ BỘ LỌC SỐ IIR

| | |
|---|---|
| **butter** | Thiết kế hàm lọc đơn giản nhất |
| **cheby1** | Thiết kế bộ lọc ChebyshewI |
| **cheby2** | Thiết kế bộ lọc ChebyshewII |
| **ellip** | Thiết kế bộ lọc Elliptic (dạng Ellip) |
| **maxflat** | Thiết kế bộ lọc thông thấp được sinh ra một cách đơn giản nhất |
| **yulewalk** | Thiết kế bộ lọc Yule-Walker |

## 5. CHỌN BỘ LỌC CHO TRƯỚC IIR

| | |
|---|---|
| **Buttord** | Chọn bộ lọc đơn giản Butterworth cho trước |
| **cheb1ord** | Chọn bộ lọc Chebyshew 1 cho trước |
| **cheb2ord** | Chọn bộ lọc Chebyshew 2 cho trước |
| **ellipord** | Chọn bộ lọc Ellip cho trước |

## 6. THIẾT KẾ BỘ LỌC FIR

| | |
|---|---|
| **cremez** | Thiết kế bộ lọc FIE số phức và hiệu ứng (méo nhỏ )ripple pha không tuyến tính |
| **fir1** | Thiết kế cửa sổ cở bản của bộ lọc FIR- -thấp ,cao, thông giữa,dừng,tích |
| **fir2** | Thiết kế cửa sổ cơ bản của bộ lọc FIR -Đáp ứng tuỳ ý |

| | |
|---|---|
| **fircls** | Thiết kế bộ lọc ở điều kiện bình phương lớn nhất- Đáp ứng tuyf ys |
| **fircls1** | Thiết kế bộ lọc FIR ở điều kiện bình phương lớn nhất- thông thấp và thông cao |
| **girrcos** | Thiết kế bộ lọc FIR cosine lớn dần |
| **firls** | Thiết kế bộ lọc FIR- đáp ứng tuỳ ý cùng với vùng chuyển đổi |
| **inflit** | Thiết bọ lọc FIR nội suy |
| **kaiserord** | Chọ điểm đặt cửa sổ cơ bản của bộ lọc sử dụng cửa sổ Kaiser |
| **remez** | Thiết kế hàm lọc tối ưu FIR Parks-McChellan |
| **remezord** | Chọn hàm lọc đặt trước Parks-McChellan |

## 7. CÁC CHUYỂN ĐỔI

| | |
|---|---|
| **czt** | Biến đổi Z |
| **dct** | Biến đổi Cosine rời rạc |
| **dftmtx** | Ma trận biến đổi Fourier rời rạc |
| **fft** | Biến đổi Fourier nhanh |
| **fftshift** | Chuyển đổi vector halves |
| **hilbert** | Biến đổi Hilbert |
| **idct** | Biến đổi cosin rời rạc ngược |
| **ifft** | Biến đổi fourier ngược nhanh |

## 8. XỬ LÝ TÍN HIỆU THỐNG KÊ VÀ PHÂN TÍCH PHỔ

| | |
|---|---|
| **cohere** | Hàm đánh giá trặt trẽ |
| **corrcoef** | Hệ số hiệu chỉnh (hệ số bù) |
| **cov** | Ma trận sai lệch |

| | |
|---|---|
| **csd** | Mật độ phổ cắt nhau |
| **pburg** | Định lượng phổ công suất theo phương pháp Burg |
| **pmtm** | Định lượng phổ công suất theo phương pháp Thomson |
| **pmusic** | Định lượng phổ công suất theo phương pháp amm nhacj |
| **psd** | Định lượng phổ công suất theo phương pháp Welch |
| **pyulear** | Định lượng phổ công suất theo phương pháp Yule-Walker |
| **spectrum** | psd, csd, dính kết và tổ hợp tfe |
| **tfe** | Đánh giá hàm truyền |
| **xcorr** | Hàm bù (hiệu chỉnh) giao nhau |
| **xcov** | Hàm sai lệch |

## 9. CÁC CỬA SỔ TÍN HIỆU

| | |
|---|---|
| **Bartlett** | Cửa sổ Bartlett |
| **Blackman** | Cửa sổ Blackman |
| **Boxcar** | Cửa sổ Boxcar |
| **Chebwin** | Cửa sổ Chebwin |
| **hamming** | Cửa sổ hamming |
| **hamning** | Cửa sổ hamning |
| **kaiser** | Cửa sổ Kaiser |
| **triang** | Cửa sổ có dạng tam giác |

## 10. THÔNG SỐ KHI MÔ HÌNH HOÁ

| | |
|---|---|
| **invfreqs** | Bộ lọc tương tự phù hợp với đáp ứng tần số |
| **invfreqz** | Bộ lọc rời rạc phù hợp với đáp ứng tần số |
| **lpe** | Các hệ số tuyến tính đoán trước sự dụng phương pháp tự bù |

| | |
|---|---|
| **prony** | Bộ lọc rời rạc Prony phù hợp với đáp ứng thời gian |
| **stmcb** | |

## 11. CÁC THAO TÁC ĐẶC BIỆT

| | |
|---|---|
| **decimate** | Lấy mẫu lại số liệu ở khoảng lấy mẫu thấp nhất |
| **deconv** | Quay ngược trước |
| **demod** | Mô hình hoá để chạy mô phỏng quá trình truyền tin |
| **dpss** | Rời rạc miền không gian tần số |
| **dpssclear** | Chuyển miền không gian tần số rời rạc vào niền cơ sở dữ liệu |
| **dpssload** | Nạp vào miền không gian tần số rời rạc từ miền cơ sở dữ liệu |
| **dpsssave** | Cất miền không gian tần số rời rạc vào miền cơ sở dữ liệu |
| **interp** | Lấy mẫu lại số liệu ở khoảng lấy mẫu cao hơn |
| **interp1** | Nội suy một chiều chung cho toolbox |
| **medfilt1** | Sự lọc điểm giữa một chiều |
| **modulate** | MOdul hoá để mô phỏng các quá trình truyền tin |
| **resample** | Lấy mẫu lại tần số với khoảng lấy mẫu mới |
| **specgram** | ảnh phổ, đối với tốc độ , tín hiệu |
| **spline** | Nội suy theo hình hộp |
| **vco** | Tạo giao động điều khiển áp |

## 12. LÀM MẪU BỘ LỌC TƯƠNG TỰ THÔNG THẤP

| | |
|---|---|
| **besselap** | Làm mẫu bộ lọc Bessel |
| **buttap** | Làm mẫu bộ lọc Butter |
| **cheb1ap** | Làm mẫu bộ lọc Chebyshev dạng 1( Sai nhỏ ở giữa giải |

thông)

| | |
|---|---|
| **cheb2ap** | Làm mẫu bộ lọc Chebyshev dạng 2( Sai nhỏ ở cuối giải thông) |
| **ellipap** | Làm mẫu bộ lọc  dạng Ellip |

## 13. CHUYỂN ĐỔI TẦN SỐ (DỊCH TẦN SỐ)

| | |
|---|---|
| **lp2bp** | Biến đổi bộ lọc thông thấp thành thông theo dải |
| **lp2bs** | Biến đổi bộ lọc thông thấp thành thông đỉnh |
| **lp2hp** | Biến đổi bộ lọc thông thấp thành thông cao |
| **lp2lp** | Biến đổi bộ lọc thông thấp thành thông thấp |

## 14. RỜI RẠC HOÁ BỘ LỌC

| | |
|---|---|
| **bilinear** | Sự chuyển đổi nủa tuyến tính với vùng được chọn trước |
| **impinvar** | Chuyển đổi xung bất biến tương tự thành số |

## 15.NHỮNG HÀM KHÁC

| | |
|---|---|
| **besself** | Thiết kế bộ lọc tương tự Bessel |
| **conv2** | quay hai chiều |
| **cplxpair** | Vector đặt trước vào bộ số phức liên hợp |
| **fft2** | Biến đổi Fourier nhanh hai chiều |
| **ifft2** | Chuyển đổi ngược hai chiều Fourier nhanh |
| **polystab** | Sự bên vững đa dạng |
| **stan** | chấm điểm số liệu tần số rời rạc |
| **strips** | Chấm điểm phóng ra |
| **xcorr2** | giao bù hai chiều |

# TOOL BOX SIMULINK

## 1.THẾ NÀO LÀ SIMULINK?

**Simulink** là một phần mềm dùng để mô hình hoá, mô phỏng và phân tích một hệ thống động. **Simulink** cung cấp cho ta hệ thống tuyến tính, hệ phi tuyến, các mô hình trong thời gian liên tục hay gián đoạn hay một hệ lai bao gồm cả liên tục và gián đoạn. Hệ thống cũng có thể có nhiều tốc độ khác nhau có nghĩa là các phần khác nhau lấy mẫu và cập nhật số liệu ở tốc độ khác nhau.

Để mô hình hoá **Simulink** cung cấp một giao diện đồ hoạ để xâu dựng mô hình như là một sơ đồ khối sử dụng thao tác " nhấn và kéo" chuột. Với giao diện này bạn có thể xây dựng mô hình như ta xây dựng trên giấy. Đây là sự khác xa các bản mô phỏng trước mà nó yêu cầu ta đưa vào các phương trình vi phân và các phương trình sai phân bằng một ngôn ngữ hay chương trình.

**Simulink** cũng bao gồm toàn bộ thư viện các khối như khối nhận tín hiệu, các nguồn tín hiệu, các phần tử tuyến tính và phi tuyến, các đầu nối. Ta cũng có thể thay đổi hay tạo ra các khối riêng của mình. Các mô hình là có thứ bậc, bạn có thể xây dựng mô hình theo cách từ dưới lên hay từ trên xuống. Bạn có thể xem hệ thống ở mức cao hơn, khi đó ta nháy kép và khối để xem xét chi tiết mô hình. Cách này cho phép ta hiểu sâu sắc tổ chức của mô hình và tác động qua lại của các phần như thế nào.

Sau khi tạo ra được một mô hình, ta cũng có thể mô phỏng nó trong **Simulink** hay bằng nhập lệnh trong cửa sổ lệnh của MATLAB. Các Menu đặc biệt thích hợp cho các công việc có sự tác động qua lại lẫn nhau, trong khi sử dụng dòng lệnh hay được dùng để chạy một loạt các mô phỏng. Sử dụng các bộ Scope và các khối hiển thị khác ta có thể xem kết quả trong khi đang chạy mô phỏng. Hơn nữa bạn có thể thay đổi thông số và xem có gì thay đổi một cách trực tiếp.

Kết quả mô phỏng có thể đặt vào MATLAB để xử lý đưa ra máy in hay hiển thị. Công cụ phân tích mô hình bao gồm cả công cụ tuyến tính hoá và "trimming" mà ta có thể truy nhập từ dòng lệnh của MATLAB, hơn nữa ta cũng có rất nhiều công cụ trong MATLAB và các bộ chương trình ứng dụng của nó. Và bởi vì MATLAB và

**Simulink** đã được tích hợp nên ta có thể mô phỏng, phân tích và sửa chữa mô hình trong cả hai môi trường tại bất kỳ điểm nào.

Để xem xét một chương trình cách tốt nhất là ta xem xét một vài ví dụ.

# 2. BÀI TOÁN THỨ NHẤT

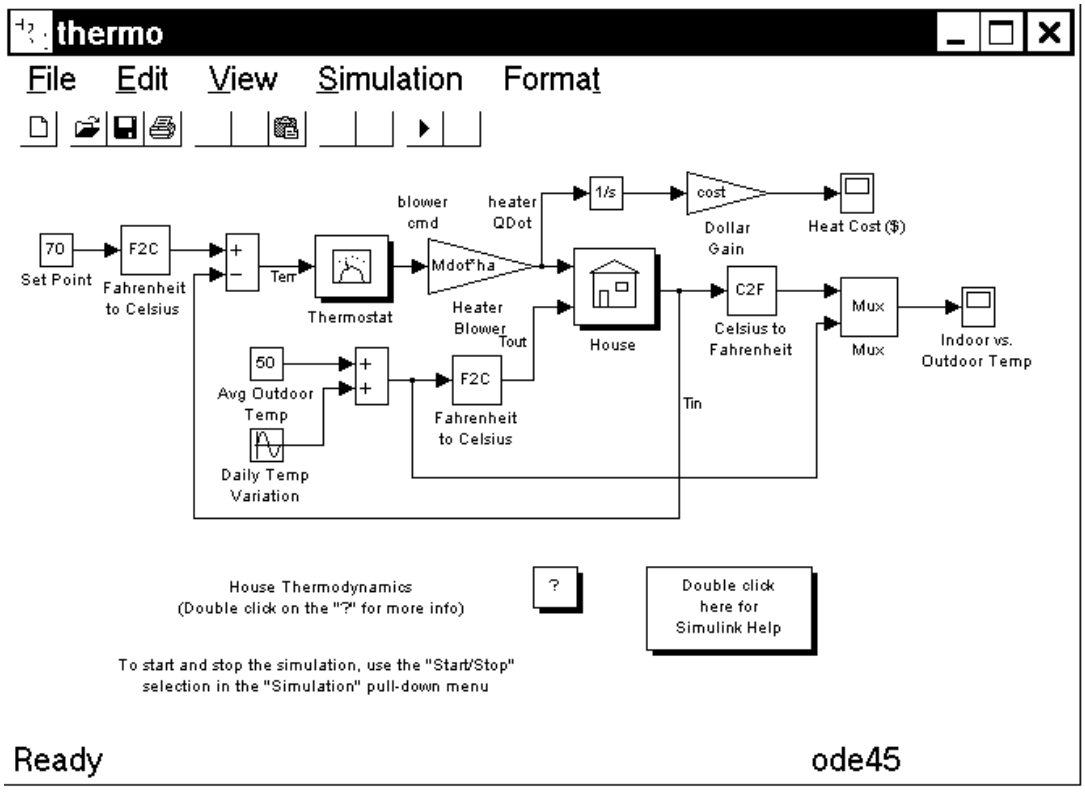
## *2.1 Đặt bài toán cho mô hình*

Một ví dụ đáng chú ý của **Simulink** là mô hình nhiệt động học của một ngôi nhà.

Để chạy mô hình này ta thực hiện các bước dưới đây:

1. Chạy MATLAB.

2. Để chạy mô hình ta đánh "Thermo" trong cửa sổ lệnh của MATLAB. Lệnh này sẽ chạy **Simulink** và tạo ra một cửa sổ chứa mô hình sau (Hình 2.2.1)




*Hình 2.1 Sơ đồ mô hình mô tả bằng Simulink*

Khi bạn xem mô hình,**Simulink** sẽ đưa ra hai khối hiển thị có tên "Indoor vs Outdoor Temp" và " Heat cost".

3.Để bắt đầu mô phỏng, vào menu **Simulation** và chọn lệnh **Start** ( Hoặc ấn

phím **Start** trên thanh công cụ của cửa sổ **Simulink**). Khi chạy mô phỏng, nhiệt độ trong và ngoài nhà sẽ hiển thi trong khối Scope "Indoor vs Outdoor Temp" và số tiền nhiệt phải trả sẽ xuất hiện trong khối Scope " Heat Cost".

4. Để dừng mô phỏng, chọn lệnh **Stop** trong menu **Simulation** ( Hoặc ấn phím **Pause** trên thanh công cụ ).

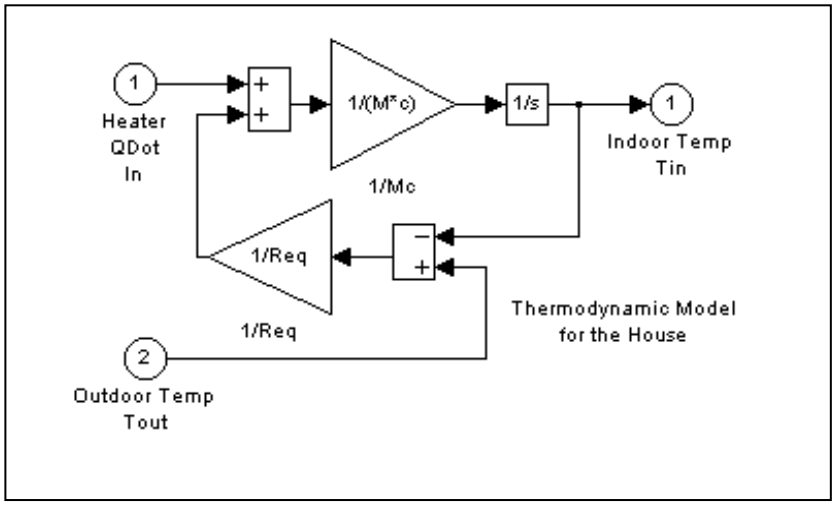
5.Khi bạn đã kết thúc việc chạy mô hình này, đóng mô hình bằng lệnh **Close** từ Menu **File**.

## *2.2 Mô tả mô hình bài toán*

Mô hình mô phỏng nhiệt động của ngôi nhà là một mô hình đơn giản. Máy điều nhiệt được đặt tại $70^0$ F và bị tác động bởi nhiệt độ bên ngoài biến đổi theo luật hình sin có biên độ là $15^0$ xung quanh nhiệt độ $50^0$. Đây là sự mô phỏng sự thay đổi nhiệt độ hàng ngày.

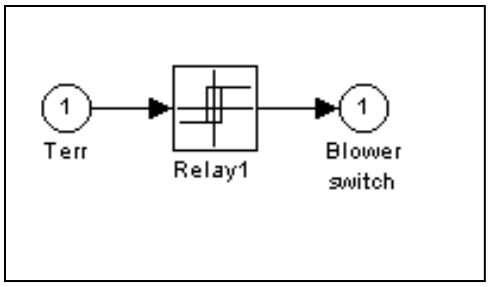
Mô hình sử dụng các hệ con để đơn giản hoá sơ đồ mô hìnhvà tạo ra hệ thống có thể sử dụng được. Hệ con là một nhóm các khối mà được đại diện bởi hệ con. Mô hình náy có 5 hệ con: máy điều nhiệt, nhà và 3 hệ biến đổi nhiệt độ ( hai hệ biến đổi từ $^0$F sang $^0$C và một biến đổi từ $^0$C sang $^0$F).

Nhiệt độ bên trong và ngoài nhà được cấp tới hệ con "House",và nó sẽ luôn cập nhật nhiệt độ trong nhà.Nhấp kép vào khối 'House" để xem các khối cơ bản của hệ phụ này.



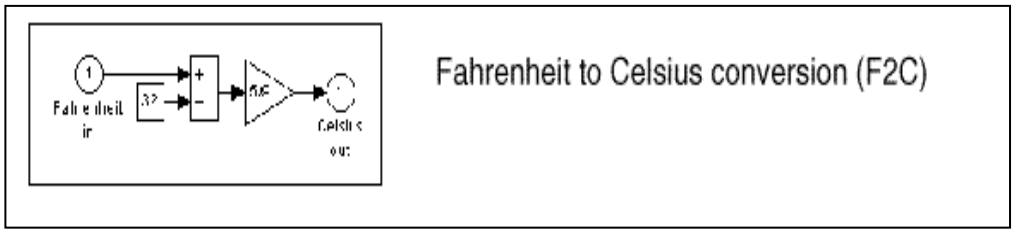

*Hình 2.2 Mô hình nhiệt động của ngôi nhà*

Mô hình hệ con ổn định nhiệt là hoạt động của máy ổn nhiệt, nó quyết định khi nào hệ thống nhiệt bật hay tắt. Nháy kép vào khối để xem các khối cơ bản của hệ này.




***Hình 2.3 Mô hình máy ổn nhiệt***

Cả nhiệt độ bên trong và bên ngoài nhà được biến đổi từ $^0$F sang $^0$C bởi một hệ con chung.




***Hình 2.4 Mô hình hệ biến đổi từ độ F sang độ C***

Khi nhiệt được bật,tiền nhiệt phải trả sẽ được tính toán và hiển thị trên khối "Heat Cost", nhiệt độ bên trong nhà được hiển thị trên khối "Indoor Temp".

## *2.3Thử lại một số quá trình*

Có một số quá trình mà ta cần thử lại để xem mô hình đáp ứng như thế nào đối với các thông số khác nhau.

- Một khối hiển thị bao gồm vùng hiển thị tín hiệu và điều khiển mà nó cho phép ta lựa chọn khoảng tín hiệu hiển thị, phóng to từng phần tín hiệu và thực hiện các công việc khác. Trục hoành biểu diễn thời gian và trục tung biểu diễn giá trị của tín hiệu.
- Khối hằng số có tên là "Setpoint" đặt nhiệt độ yêu cầu trong nhà. Mở khối này ra và đặt giá trị tới $80^0$ F khi đang chay mô phỏng. Xem nhiệt độ bên trong nhà và tiền nhiệt thay đổi. Cũng như vậy ta cũng có thể thay đổi nhiệt độ bên ngoài và xem ảnh hưởng của nó đối với mô hình.
- Điều chỉnh độ biến đổi nhiệt độ hằng ngày bởi việc mở khối phát sóng sin có tên "Daily Temp Variation" và thay đổi thông số biên độ.

## 2.4 Hiệu quả của việc mô phỏng quá trình.

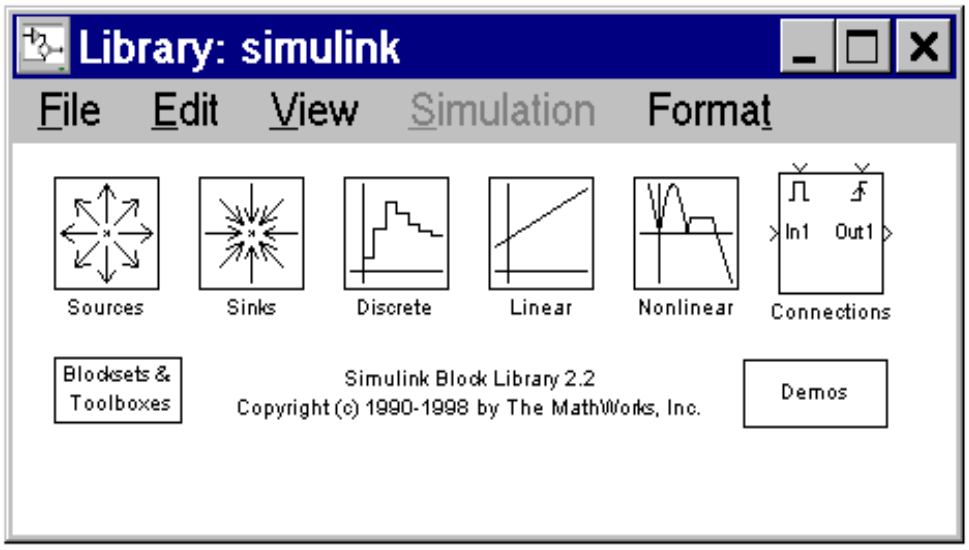
Ví dụ này làm sáng tỏ một vài công việc đã được sử dụng để xây dựng mô hình.

- Chạy mô phỏng bao gồm đặt các thông số và bắt đầu mô phỏng với lệnh **Start**.
- Bạn có thể gói gọn toàn bộ các khối có liên quan trong một khối đơn gọi là hệ con.
- Bạn có thể tạo ra biểu tượng của mình và thiết kế một hộp đối thoại cho một khối công việc sử dụng "masking".Trong mô hình nhiệt tất cả các hệ con được tạo ra biểu tượng sử dụng "Masking".
- Khối hiển thị hiển thị ra đồ hoạ như một máy hiện sóng thực sự. Khối hiển thị hiển thị tín hiệu vào của nó.

## 2.5 Các ví dụ có thể sử dụng khác của *Simulink*

Các ví dụ khác làm sáng tỏ khái niệm về mô hình có thể được sử dụng. Bạn có thể xem các ví dụ này từ cửa sổ thư viện của Simulink.

1. Đánh "simulink" trong cửa sổ lệnh của Matlab. Cửa sổ thư viện các khối sẽ xuất hiện.

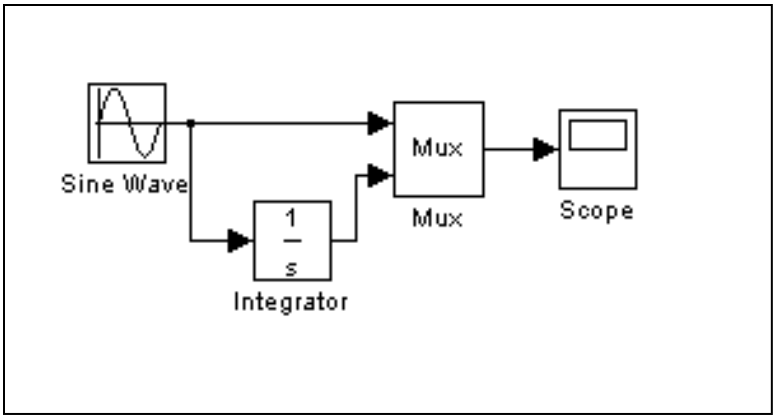



**Hình 2.5 Cửa sổ các thư viện của Simulink**

2. Nhấp kép vào biểu tượng " Demos". Cửa sổ "Matlab demos" sẽ xuất hiện. Cửa sổ này có một vài ví dụ đáng quan tâm mà nó làm sáng tỏ đặc điểm sử dụng của **Simulink**.

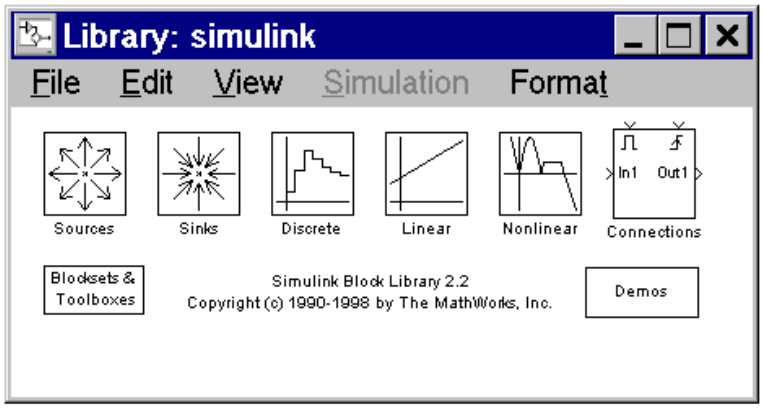
## 3. PHƯƠNG PHÁP XÂY DỰNG MÔ HÌNH

Ví dụ này sẽ trình bầy cho ta cách xây dựng một mô hình như thết nào, cách sử dụng các lệnh và các thao tác bạn sẽ sử dụng để xây dựng mô hình của mình. Ta sẽ xây dựng mô hình tích phân sóng sin và hiển thị kết quả cùng với sóng sin. Sơ đồ khối của mô hình như sau:




*Hình 2.6 Mô hình tích phân sóng hình sin*

Đánh lệnh "simulink" từ cửa số lệnh của Matlab để hiển thị cửa sổ thư viện Simulink và nếu không có cửa sổ mô hình nào được mở thì một cửa sổ mô hình mới được tạo ra. Cửa sổ thư viện của Simulink như sau:



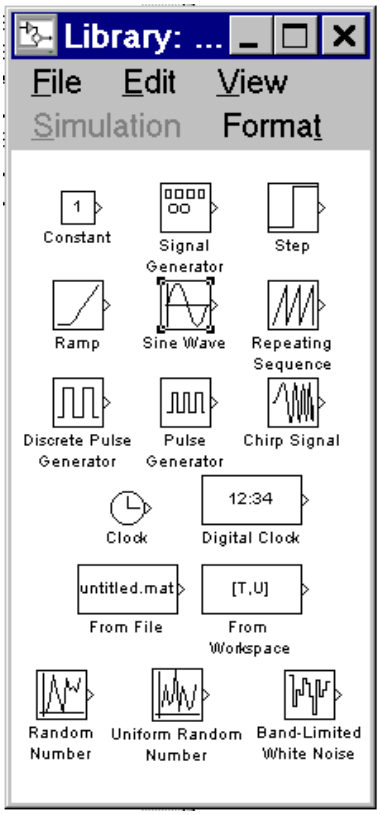

*Hình 2.7 Cửa sổ thư việncủa Simulink*

Trong mô hình này bạn lấy các khối sau từ các thư viện:

- Thư viện các nguồn tín hiệu (Khối phát sóng sin).
- Thư viện các khối nhận tín hiệu(Khối hiển thị).
- Thư viện các hàm tuyến tính (Khối tích phân).

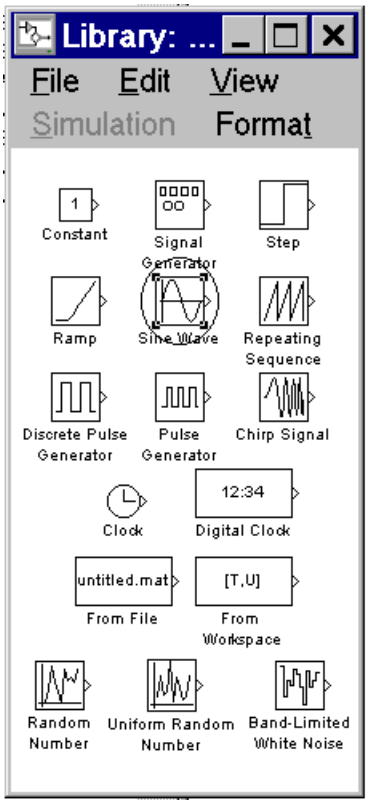
- Thư viên các đầu nối ( Khối chuyển mạch).

Mở thư viện các nguồn tín hiệu để vào khối sóng sin. Để mở một thư viện ta nháy kép vào nó. **Simulink** sẽ hiển thị một cửa sổ chứa tất cả các khối của thư viện đó. Trong thư viện nguồn tín hiệu tất cả các khối đều là nguồn tín hiệu. Thư viện nguồn tín hiệu được thể hiện như hình 2.8



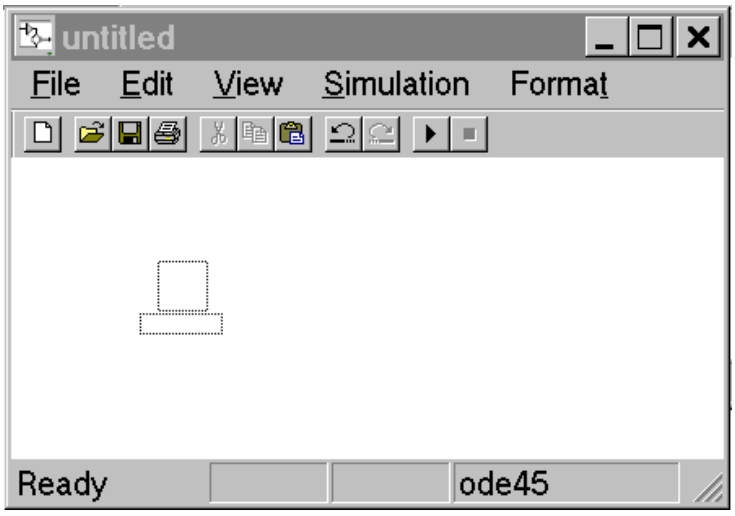

***Hình 2.8 Cửa sổ thư viện nguồn tín hiệu***

Bạn thêm khối vào mô hình của bạn bằng cách chép nó từ thư viện hay từ mô hình khác. Trong bài tập này bạn cần chép khối phát sóng hình sin. Đặt con trỏ trên khối ấn và giữ phím chuột, kéo khối tới cửa sổ mô hình.



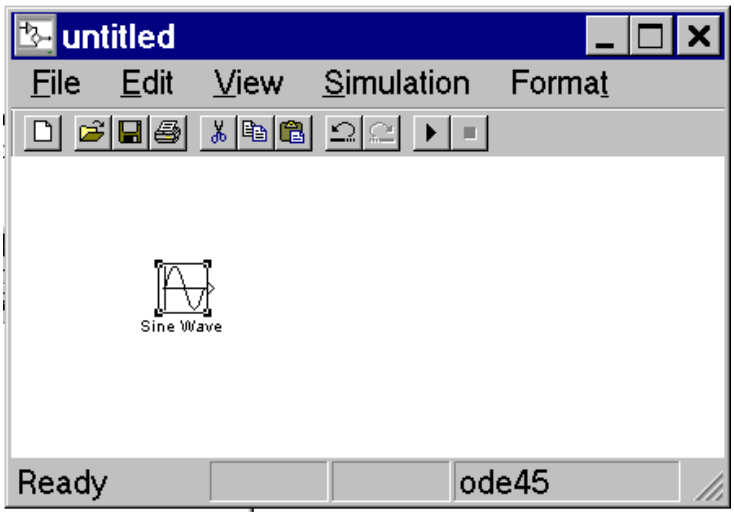

*Hình 2.9 Copy khối sin vào mô hình*

Khi bạn di chuyển khối bạn có thể thấy khối và tên của nó di chuyển cùng với con trỏ.




*Hình 2.10 Khối và tên khối di chuyển cùng con trỏ*

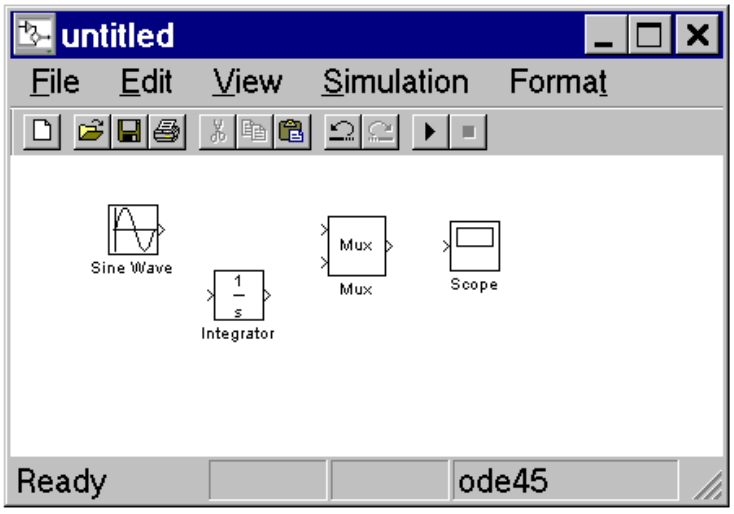
Khi con trỏ tới nơi bạn cần đặt khối trong mô hình bạn nhả phím chuột, một bản copy của khối phát hình đã ở trong mô hình của bạn.




*Hình 2.11 Cửa sổ mô hình khi bạn đã copy khối sóng sin*

Theo cách này chép những khối còn lại vào mô hình của bạn. Bạn có thể di chuyển khối trong mô hình sử dụng kỹ thuật như khi bạn chép khối, hoặc có thể di chuyển khối trong khoảng nhỏ bằng cách chọn khối và ấn các phím mũi tên.

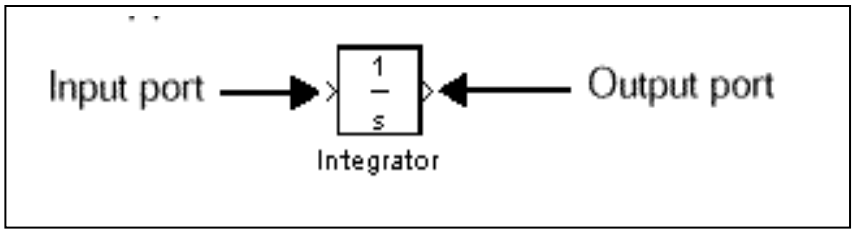
Với tất cả các khối đã chép cửa sổ mô hình như sau:




*Hình 2.12 Cửa sổ mô hình với các khối đã copy*

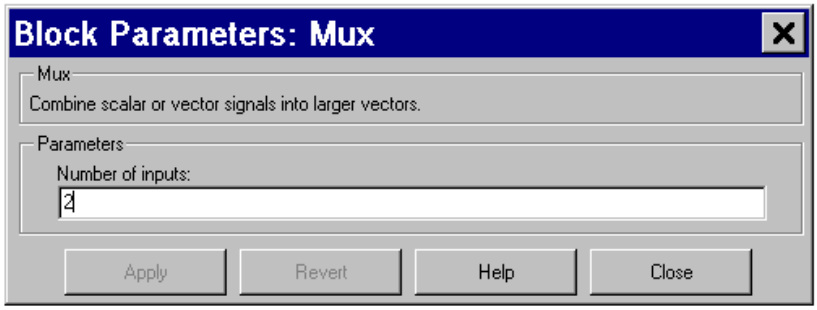
Nếu bạn xem kỹ từng khối, bạn thấy dấu > ở bên phải khối sin và dấu ở bên trái khối MUX. Dấu ở đầu ra một khối là cổng ra, ở đầu vào một khối là cổng vào. Tín

hiệu đi từ đầu ra một khối tới đàu vào khối khác theo một đường nối. Khi một cổng đã được nối thì biểu tượng của cổng cũng mất đi.



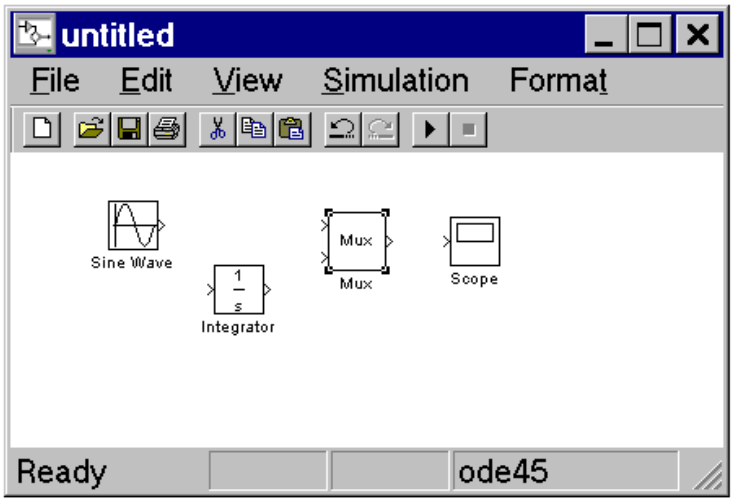

*Hình 2.13 Đầu vào và ra của một khối*

Bạn thấy rằng khối MUX có ba cổng vào nhưng chỉ có 2 tín hiệu vào. Để thay đổi số cổng vào bạn mở khối MUX bằng cách nháy kép trên khối và thay đổi giá trị thông số " Number of Input" tới 2. Sau đó ấn phím **Close**, **Simulink** sẽ điều chỉnh số cổng vào.



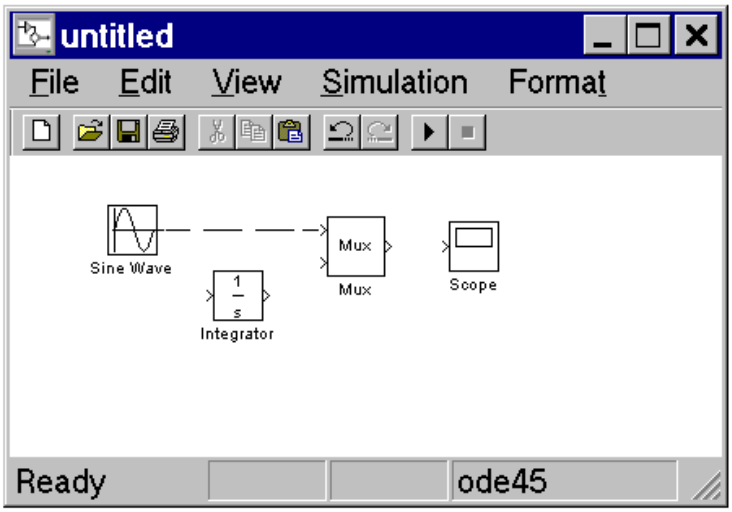

*Hình 2.14 Cửa sổ thông số khối MUX*

Hiện nay ta có thể nối các khối. Nối đầu ra khối phát sinh tới đầu vào trên của khối MUX. Đặt con trỏ tới đầu ra của khối sin, lúc đó con trỏ sẽ thay đổi thành một chữ thập nhỏ.



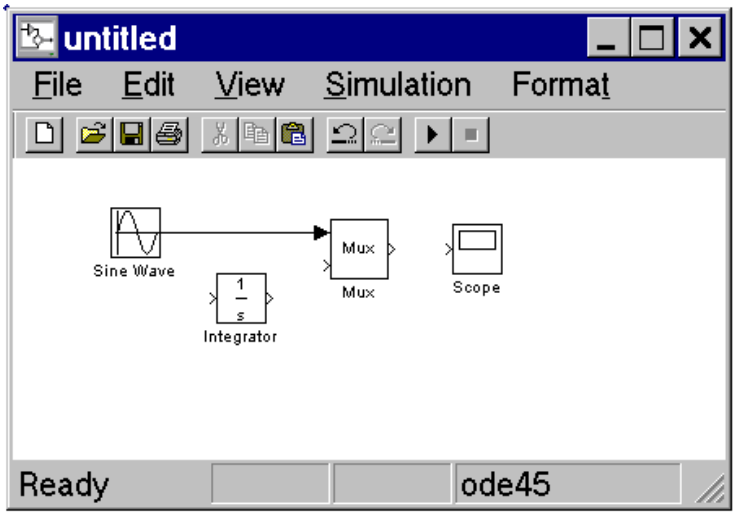

*Hình 2.15 Cửa sổ mô hình trước khi nối dây*

Giữ và kéo chuột tới đầu vào của khối MUX. Chú ý đường là nét đứt khi phím chuột vẫn giữ và con trỏ sẽ thay đổi thành chữ thập kép khi nó lại gần khối MUX.




*Hình 2.16 Cửa sổ mô hình khi đang nối dây*

Ta nhả phím chuột ra và các khối đã được nối. Bạn có thể nối bằng cách nhả phím chuột khi con trỏ ở bên trong khối. Khi đó đường nối sẽ nối vào cổng gần vị trí con trỏ nhất.

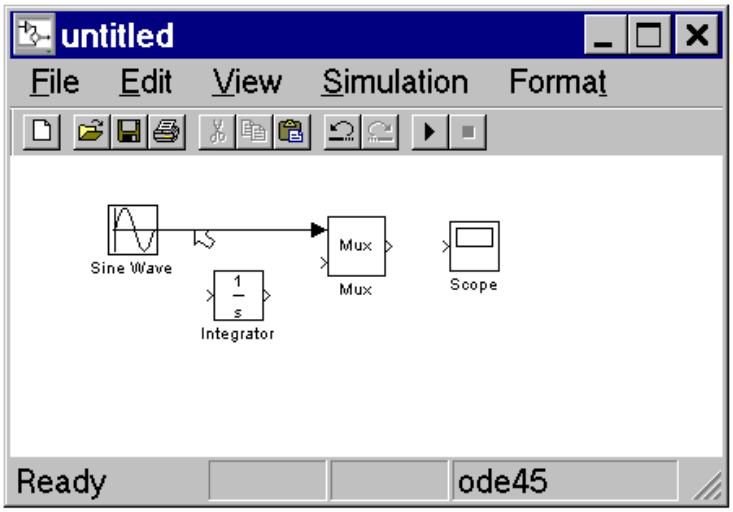



*Hình 2.17 Hình sin nối vào đầu trên khối MUX*

Phần lớn các đường nối từ đầu ra của khối tới đầu vào của khối khác. Có đường nối từ một đường tới đầu vào của một khối gọi là đường rẽ nhánh.

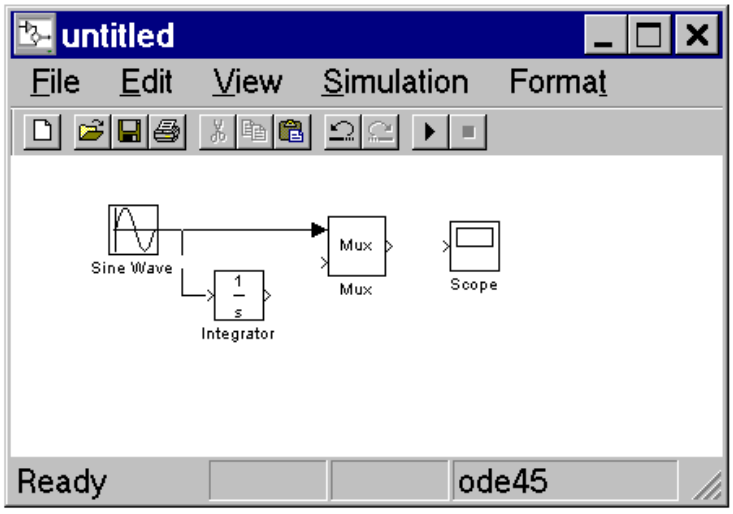
Vẽ đường rẽ nhánh có sự khác biệt nhỏ so với vẽ đường. Để nối đường đã có ta thực hiện theo các bước sau:

1. Đặt vị trí con trỏ ở trên đường cần rẽ nhánh.



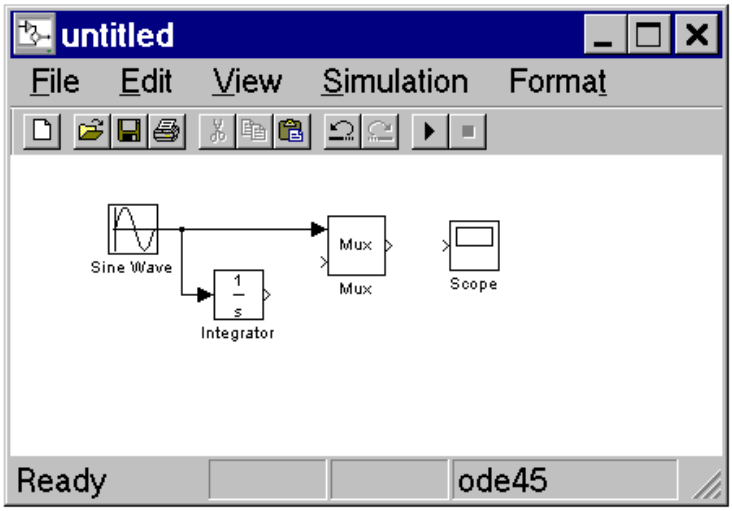

*Hình 2.18 Con trỏ đặt vào điểm cần rẽ nhánh*

2. Ấn và giữ phím **Ctrl**, ấn và giữ phím chuột kéo con trỏ tới đầu vào của khối.



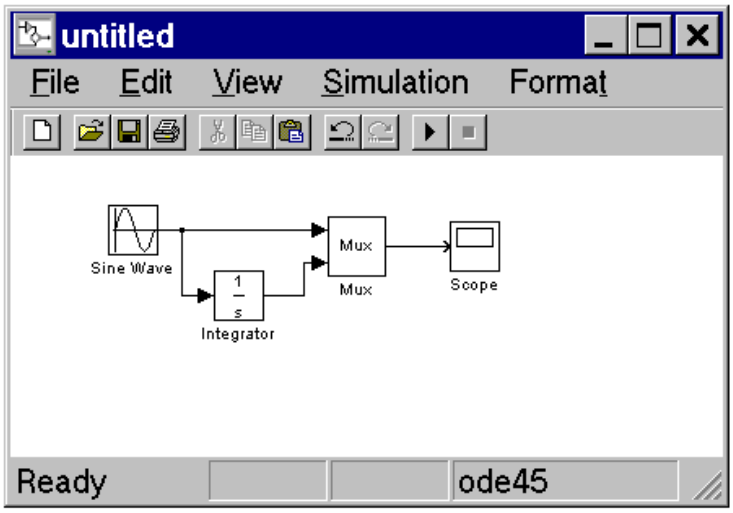

*Hình 2.19 Nối các khối*

3. Nhả phím chuột, **Simulink** sẽ vẽ một đường từ điểm bắt đầu tới cổng vào của khối.



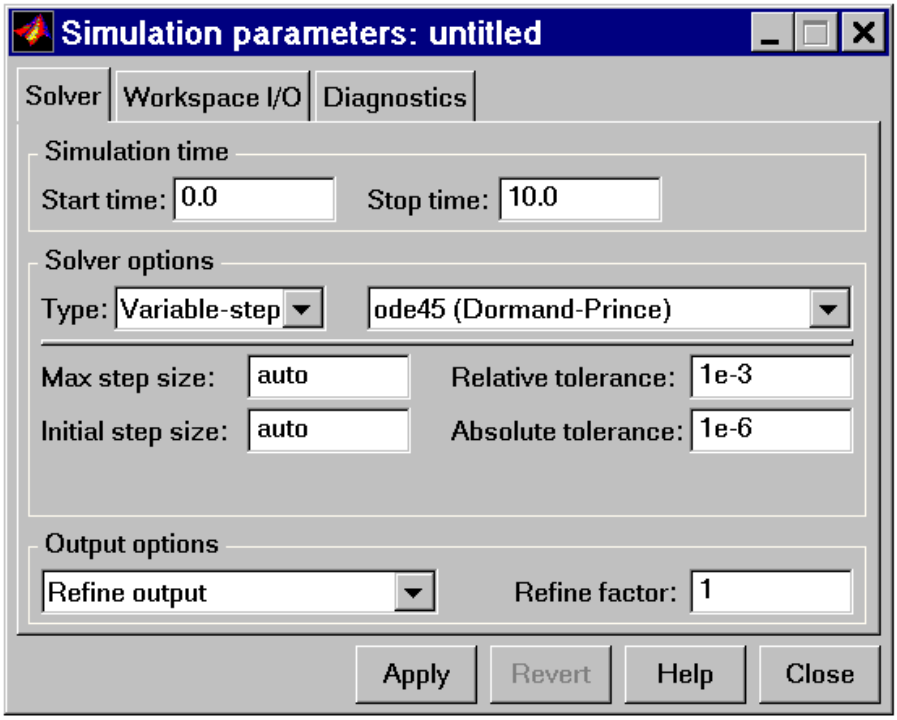

*Hình 2.20 Đường nối dây rẽ nhánh*

Kết thúc việc nối dây, mô hình như sau:




*Hình 2.21 Cửa sổ mô hình khi ta đã vẽ xong*

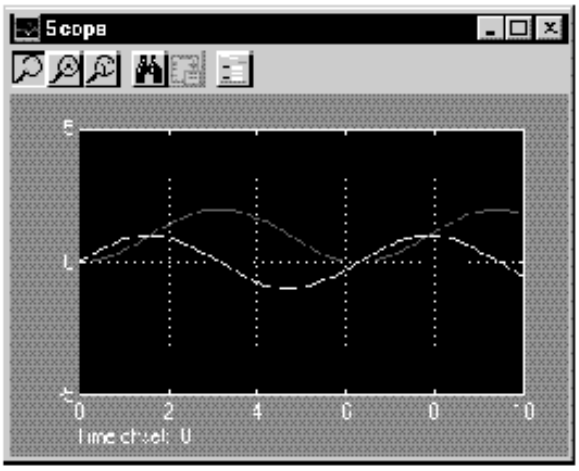
Bây giờ ta mở khối Scope để hiển thị tín hiệu ra và chạy mô phỏng trong 10s. Đầu tiên ta phải đặt thông số mô phỏng bằng lệnh **Parameter** trong menu **Simulation**, hộp hội thoại xuất hiện. Chú ý **Stoptime** đặt là 10.0s.




*Hình 2.22 Hộp đối thoại Parameter của Simulink*

Để đóng hộp đối thoại **Simulink Parameter** ta ấn phím **Close**. **Simulink** sẽ áp dụng các thông số ta đặt và đóng cửa sổ hội thoại.

Chọn **Start** trong menu **Simulation** và xem sự thay đổi của đầu vào khối hiển thị.




*Hình 2.23 Cửa sổ hiển thị tín hiệu ra khối Scope*

Để lưu mô hình nay sử dụng lệnh **Save** trong menu **File** và nhập tên và vị trí của file. File này chứa các mô tả của mô hình.

Để kết thúc Simulink và Matlab chọn lệnh **exit Matlab** trong menu **File** hoặc ta đánh lệnh **Quit** trong cửa sổ lệnh Matlab. Nếu bạn muốn thoát khỏi **Simulink** mà không thoát khỏi Matlab đóng tất cả các cửa sổ của **Simulink.**