TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN

PHẠM THỊ NGỌC YÊN
NGÔ HỮU TÌNH
LÊ TẤN HÙNG
NGUYỄN THỊ LAN HƯƠNG

CƠ SỞ MATLAB
VÀ ỨNG DỤNG

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN

PHẠM THỊ NGỌC YẾN - NGÔ HỮU TÌNH
LÊ TẤN HÙNG - NGUYỄN THỊ LAN HƯƠNG

CƠ SỞ MATLAB VÀ ỨNG DỤNG

PGS, TS TẠ DUY LIÊM
Hiệu đính và giới thiệu

Giáo trình cho các trường đại học và cao đẳng

In lần thứ 3 , có sửa chữa

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
Hà Nội – 2005

LỜI GIỚI THIỆU

Matlab (Matrix Laboratory) theo tên gọi của nó, là một công cụ phần mềm của Math Work, ban đầu nó được phát triển nhằm phục vụ chủ yếu cho việc mô tả các nghiên cứu kỹ thuật bằng toán học với những phần tử cơ bản là ma trận. Trong các lĩnh vực kỹ thuật chuyên ngành như điện và điện tử, vật lý hạt nhân, điều khiển tự động, robot công nghiệp, trong các ngành xử lý toán chuyên dụng như thống kê - kế toán và ngay cả trong lĩnh vực nghiên cứu về gien sinh học hay khí hậu và thời tiết... thường gặp những dữ liệu rời rạc (discret) ta có thể lưu trữ dưới dạng ma trận. Còn đối với hệ dữ liệu liên tục (continuous) như âm thanh, hình ảnh, hoặc đơn giản như các đại lượng vật lý tương tự (analog): điện áp, dòng điện, tần số, áp suất, lưu lượng... phải được biến đổi thành các tín hiệu số (digital) rồi mới tập hợp lại trong các file dữ liệu. Quá trình đó có thể được xử lý bằng các hàm toán học của Matlab.

Mức phát triển của Matlab ngày nay đã chứng tỏ nó là một phần mềm có giao diện cực mạnh cùng nhiều lợi thế trong kỹ thuật lập trình để giải quyết những vấn đề rất đa dạng trong nghiên cứu khoa học kỹ thuật.

Trước hết, các câu lệnh của Matlab được viết rất sát với các mô tả kỹ thuật khiến cho việc lập trình bằng ngôn ngữ này được thực hiện nhanh hơn, dễ hơn so với nhiều ngôn ngữ đã trở nên thông dụng như Pascal, Fortran... Những hàm sẵn có trong Matlab có cấu trúc thiết lập gần giống như ngôn ngữ C, bởi vậy người sử dụng không mất nhiều thì giờ học hỏi khi đã nắm được những vấn đề cơ bản của một số ngôn ngữ lập trình thông dụng.

Tiếp theo, Matlab không chỉ cho phép đặt vấn đề tính toán mà còn có thể xử lý dữ liệu, biểu diễn đồ họa một cách mềm dẻo, đơn giản và chính xác trong không gian 2D cũng như 3D, kể cả khả năng tạo hoạt cảnh cho những mô tả sinh động, bởi những công cụ như các thư viện chuẩn, các hàm sẵn có cho các ứng dụng đa dạng, các tệp lệnh ngày càng được mở rộng bởi 25 thư viện trợ giúp (Tools box) và bản thân các hàm ứng dụng được tạo lập bởi người sử dụng. Không cần nhiều đến kiến thức về máy tính cũng như kỹ thuật lập trình có tính xảo thuật, mà chỉ cần đến những hiểu biết cơ bản về lý thuyết số, toán ứng dụng, phương pháp tính và khả năng lập trình thông dụng, người sử dụng đã có thể dùng Matlab như một công cụ hữu hiệu cho lĩnh vực chuyên ngành của mình.

Sau hết, việc cài đặt Matlab thật là dễ dàng. Ta chỉ cần chú ý đôi chút nếu muốn dùng thêm các thư viện trợ giúp như Simulink, Fuzzy, Toolbox, DSP (Digital signal Processing) hay muốn tích hợp phần mềm này với một vài ngôn ngữ quen thuộc của người sử dụng như C, C++, Fortran... Matlab có thể hoạt động trên hầu hết các hệ máy tính, từ máy tính cá nhân (PC) đến các hệ máy tính lớn (SC); với các version 3.5 trở về trước, nó chạy trong môi trường MS-Dos., các version 4.0, 4.2, 5.1, 5.2 chạy trong môi trường Windows. Còn lại, các version Matlab khác cần đến môi trường tương tác Unix.

Được các công ty phần mềm hàng đầu trên thế giới phát triển, ngày nay Matlab đã trở thành công cụ phổ biến, đắc lực trong các môi trường công tác rất khác nhau, từ việc giảng dạy, đào tạo trong các nhà trường đại học và trung học chuyên nghiệp, đến việc triển khai ứng dụng trong các cơ sở nghiên cứu, sản xuất, dịch vụ và thương mại; từ các lĩnh vực khoa học cơ bản như toán học, vật lý, hóa học, sinh học... đến các lĩnh vực kỹ thuật công nghiệp, kinh tế, quốc phòng... Nhận thức rõ khả năng ứng dụng mạnh mẽ của nó, tập thể tác giả gồm các nhà chuyên môn của Trường cao đẳng Sư phạm Kỹ thuật I và Trường đại học Bách khoa Hà Nội đã dày công sưu tập và biên soạn cuốn "CƠ SỞ MATLAB VÀ ỨNG DỤNG" để kịp thời cung cấp cho bạn đọc một công cụ hữu hiệu của tin học ứng dụng, khiến cỗ máy tính của bạn trở nên thú vị hơn, hiệu quả hơn trong công việc hàng ngày.

Cuốn sách được soạn thảo cho mục đích tự đào tạo, có dàn ý sáng sủa, gồm các vấn đề mang tính hệ thống từ những khái niệm cơ bản đến kỹ thuật lập trình. Các tác giả đã trình bày hết sức tỷ mỷ những vấn đề đồ họa trong không gian 2D và 3D, kể cả các vấn đề mầu sắc và kiểm soát các hệ mầu, vốn là một khía cạnh được phần lớn bạn đọc quan tâm. Đặc biệt ở phần những hàm trong thư viện trợ giúp, những vấn đề xử lý tín hiệu số cũng như việc ứng dụng phần mềm Simulink để mô phỏng và phân tích một hệ thống hoạt động với các đặc tính tuyến tính hay phi tuyến, theo thời gian liên tục, gián đoạn hay hỗn hợp với nhiều tốc độ khác nhau, sẽ cung cấp cho bạn đọc những kiến thức nâng cao trong quá trình sử dụng Matlab.

Với niềm tin về sức thuyết phục của bản thân phần mềm Matlab, chúng tôi xin trân trọng giới thiệu cùng bạn đọc cuốn sách có giá trị này.

Hà Nội, Hè 1999

Pgs. Pts. Tạ Duy Liêm

LỜI NÓI ĐẦU

Các nhà khoa học, các kỹ sư và kỹ thuật viên luôn luôn quan tâm đến việc phát triển, nâng cao khả năng tính toán và xử lý trên máy tính những vấn đề chuyên môn rất đa dạng của họ. Nhưng để viết được một chương trình bằng ngôn ngữ lập trình cấp cao nhằm giải quyết những vấn đề như vậy, thường phải tốn nhiều công sức và thời gian, nhất là bên cạnh những kiến thức sâu sắc của chuyên ngành khoa học kỹ thuật, người lập trình còn phải có hiểu biết tường tận về hệ thống máy tính, về bản thân bộ môn toán học và những xảo thuật của kỹ thuật lập trình. Đôi khi điều đó là nan giải đối với các nhà chuyên môn kỹ thuật.

Để tạo điều kiện ứng dụng nhanh chóng và hiệu quả cho các nhà chuyên môn ngoài ngành tin học, các chuyên gia phát triển phần mềm đã thiết lập những công cụ trợ giúp cho những mục đích sử dụng đa dạng trên nhiều lĩnh vực chuyên môn khác nhau. Matlab cũng chính là một trong những phần mềm như vậy.

Matlab có thể làm được những gì? Matlab hoạt động ra sao? Ai có thể học và sử dụng Matlab?

Matlab là chương trình phần mềm trợ giúp cho việc tính toán và hiển thị. Nó có thể chạy trên hầu hết các hệ máy tính, từ máy tính cá nhân đến các hệ super computer. Matlab được điều khiển bởi tập các lệnh, tác động qua bàn phím trên cửa sổ điều khiển. Nó cũng cho phép một khả năng lập trình với cú pháp thông dịch lệnh - còn gọi là script file. Các lệnh hay bộ lệnh của Matlab lên đến con số hàng trăm và ngày càng được mở rộng bởi các phần Tools box (thư viện trợ giúp) hay thông qua các hàm ứng dụng được tạo lập bởi người sử dụng.

Các lệnh của Matlab rất mạnh và hiệu quả, nó cho phép giải các loại hình toán khác nhau và đặc biệt hữu dụng cho các hệ phương trình tuyến tính hay các bài toán ma trận. Cùng với 25 Tool box khác nhau, Matlab cho bạn một sự lựa chọn hoàn chỉnh và phong phú các công cụ trợ giúp đắc lực cho những linh vực nghiên cứu chuyên môn khác nhau.

Với Matlab, các vấn đề cần giải quyết của bạn sẽ được phân tích và xử lý theo 5 bước như sau:

Bước 1: Đặt vấn đề

Bài toán đưa ra cần được phân tích, biểu diễn một cách rõ ràng và cụ thể. Đây là bước mở đầu rất quan trọng, nó quyết định toàn bộ hướng giải quyết tiếp theo của bài toán đặt ra.

Bước 2: Mô tả các giá trị dữ liệu vào/ra.

Việc mô tả các thông tin cần giải đáp có liên quan trực tiếp đến các tham số được sử dụng trong tính toán, bởi vậy bước này cần được tiến hành cẩn trọng. Trong nhiều trường hợp, sơ đồ khối được sử dụng để xác định vị trí các luồng vào/ra, tuy nhiên đôi khi chúng chỉ là các hộp đen vì không thể xác định được luồng ra tại một điểm nào đó trong các bước. Mặc dầu vậy, ta vẫn chỉ ra được những thông tin để tính toán luồng ra.

- Bước 3: Các tính toán bằng tay với các tập dữ liệu đầu vào đơn giản

Đây là bước tiền để nhằm tìm kiếm những giải pháp cụ thể, bạn không nên bỏ qua kể cả đối với các bài toán đơn giản. Nếu trong bước này bạn chưa lấy được dữ liệu hay chưa tính được đầu ra thì có thể chuyển sang bước kế tiếp.

- Bước 4: Chuyển bài toán sang giải pháp bằng Matlab

Ở bước này bạn sẽ sử dụng các hàm toán, cũng là các lệnh để mô tả bài toán theo MatLab.

- Bước 5: Kiểm tra

Đây là bước cuối cùng trong tiến trình giải bài toán. Bài toán được kiểm tra bằng các dữ liệu đầu vào. Matlab thực hiện bài toán và cho bạn kết quả ở đầu ra.

Trong trường hợp không có kết quả hoặc kết quả sai thì điều đó có nghĩa là Matlab chưa thực hiện được bài toán, bạn cần kiểm tra lại cả tính toán bằng tay và thao tác bằng Matlab.

Để minh họa cụ thể, ta hãy lấy một ví dụ:

Đặt vấn đề: Giải bài toán "Tính khoảng cách giữa hai điểm cho trước trên một đường thẳng thuộc mặt phẳng xác định"

Mô tả: Điểm 1: P1 = (1, 5) Khoảng cách giữa hai điểm

Điểm 2: P2 = (4, 7)

Thao tác tay: Tính khoảng cách giữa hai điểm bằng công thức Pythagorean

Giải pháp bằng Matlab:

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

Kiểm tra:

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

MatLab ngày nay đã trở nên thông dụng và là một công cụ trợ giúp hữu hiệu cho các nhà chuyên môn, những sinh viên đang theo học trong các trường đại học và trung học chuyên nghiệp, các kỹ sư, cán bộ kỹ thuật ... nhằm giải quyết các vấn đề rất đa dạng trong công việc thường ngày của họ.

Để cung cấp cho bạn đọc thêm một công cụ hữu ích nữa của tin học ứng dụng, chúng tôi giới thiệu cuốn sách "Cơ sở Matlab và ứng dụng". Tập thể tác giả xin chân thành cảm ơn các bạn bè, đồng nghiệp đã đóng góp nhiều ý kiến bổ ích cho việc soạn thảo; cảm ơn các cán bộ biên tập của nhà xuất bản Khoa học và Kỹ thuật đã bỏ nhiều công sức để giúp cho cuốn sách sớm ra mắt phục vụ bạn đọc. Chúng tôi cũng xin cảm ơn Pgs. Pts. Tạ Duy Liêm đã giành thì giờ hiệu đính và giới thiệu cùng bạn đọc cuốn sách này. Chắc chắn lần xuất bản đầu tiên không thể tránh hết các thiếu sót, chúng tôi mong được sự chỉ giáo của bạn đọc và đồng nghiệp.

CÁC TÁC GIẢ

PHẦN THỨ NHẤT

CƠ SỞ MATLAB

CHƯƠNG 1

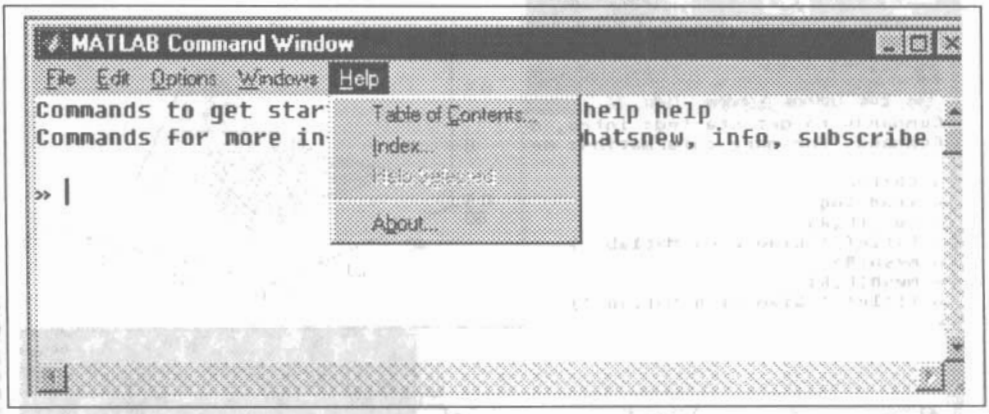
CÁC KHÁI NIỆM CƠ BẢN

1.1. HOẠT ĐỘNG CỦA MATLAB

Matlab 3.5 trở xuống hoạt động trong môi trường MS-Dos.

Matlab 4.0, 4.2, 5.1, 5.2 ... hoạt động trong môi trường Windows.

Còn các version Matlab khác thì làm việc với môi trường tương tác Unix.



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

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 nhấn chuột trên các icon hay còn gọi là các biểu tượng. Còn với môi trường Unix, Dos thông qua dòng lệnh

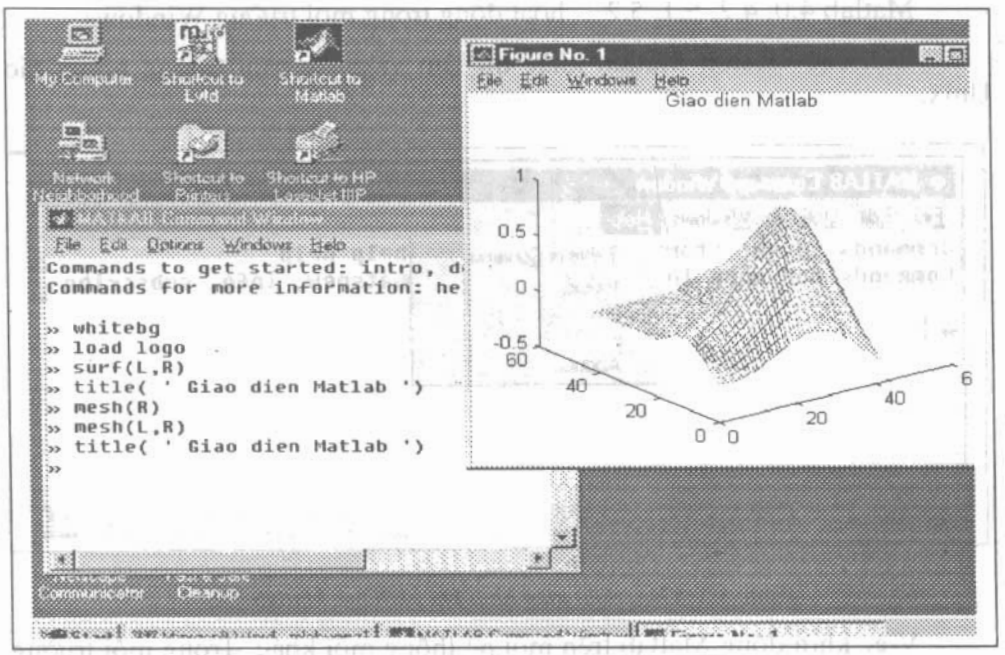
:\Matlab

Giao diện của Matlab sử dụng hai 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 đồ họa.

Việc ngắt chương trình đang thực hiện hoặc các chương trình thực hiện không đúng theo yêu cầu đều 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 1.2 *Hai cửa sổ giao diện của Matlab*

1.2. 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 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 sang trái 1 từ
Ctrl + r hoặc Ctrl + →	Chuyển con trỏ 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 một vài các ký tự đặc biệt khác nhưng luôn phải bắt đầu bằng chữ cái. Tên của các hàm đã được đặt cũng có thể được sử dụng làm tên của biến với điều kiện hàm này 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 biến được 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 hiện như sau:

1. Tất cả các biến trong bộ nhớ được nạp 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 có sự phân biệt 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 off (bỏ thuộc tính phân biệt chữ hoa và chữ thường)

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 một số hàm có sẵn của Matlab.

size (A)	Cho ra một vector chứa kích thước ma trận A. Phần tử đầu tiên của vector là số hàng của ma trận, phần tử thứ hai là số cột của ma trận.
[m n] = size(A)	Trả giá trị độ lớn của ma trận A vào vector xác định bởi hai 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 độ lớn của vector x. Nếu x là vector hàng m phần tử thì giá trị đầu của vector là m và giá trị thứ hai là 1. Trường hợp x là vector cột n thì giá trị thứ nhất sẽ là 1 và thứ hai là n.
lengh(x)	Trả giá trị chiều dài của vector x
lengh(A)	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 nếu 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 tính toán của máy xác định bởi khoảng từ 1 đến một biến dấu phẩy động tiếp đó. Biến esp được sử dụng như là sai số trong một vài 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	$(a_1 . a_2 - b_1 . b_2) + i\,(a_1 . a_2 - b_1 . b_2)$
c2	$a_1^2 + b_2^2 \qquad a_1^2 + b_2^2$
abs(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 $-\pi$ đến π.

c) Toạ độ biểu diễn số phức

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

Trong đó:

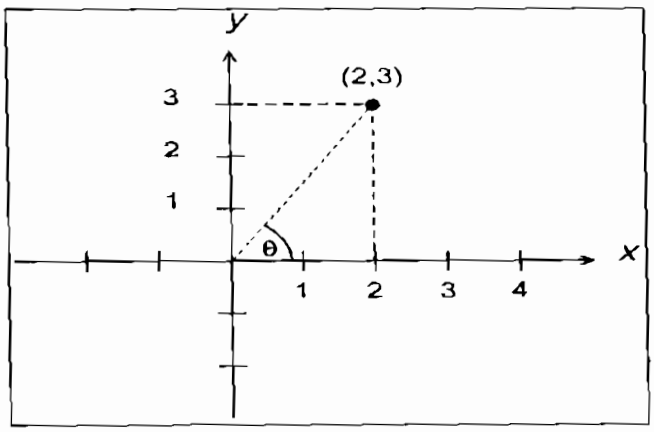
$r = \sqrt{a_1^2 + b_2^2}$

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

Ngược lại :

$a = r\cos\theta$

$b = r\sin\theta$



Hình 1.3 ***Biểu diễn 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 đêcác, 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 HỌA TRONG MATLAB

1.4.1. Các lệnh thông dụng trong đồ họa bằng Matlab

Matlab sử dụng lệnh X-Y Plots để vẽ đồ thị, biểu đồ cho các thông tin một cách dễ dàng. Trong không gian 2D, 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 trong không gian 3D thì đơn giản với Matlab, chúng ta chỉ cần đổi sang dùng lệnh X-Y-Z Plots để vẽ.

plot(x,y)	Vẽ đồ thị theo toạ độ x-y
plot(x,y,z)	Vẽ đồ thị theo toạ độ x-y-z
title	Đưa các title vào 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 giống grid line trên đồ thị
plot(y)	Vẽ đồ thị theo y bỏ qua chỉ số theo y Nếu y là số phức 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

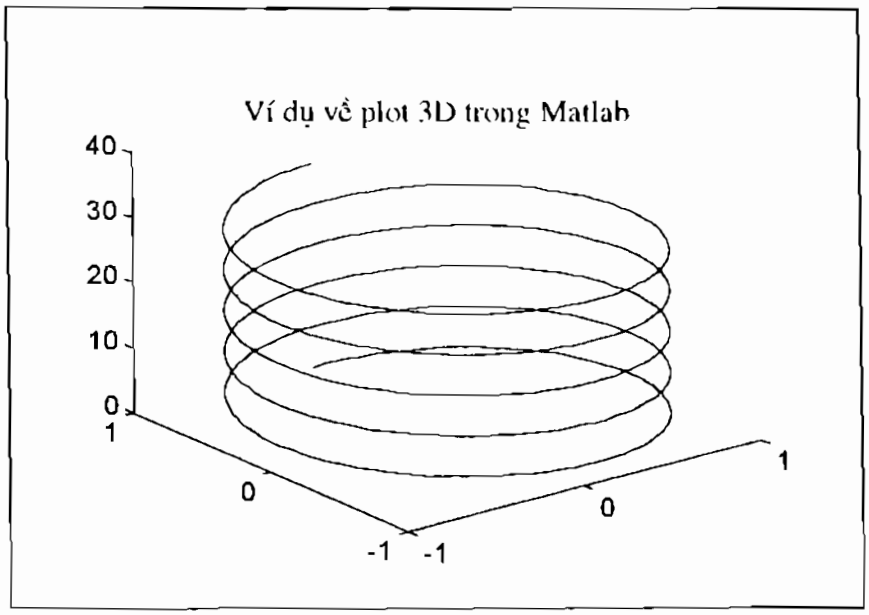
plot(x,y,z,S) plot (X, Y, Z, S) vẽ theo x, y, z với

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 +



Hình 1.4 *Đường Helix trong không gian 3D.*

```
>>  t = 0:pi/50:10*pi;
>>  plot3(sin(t),cos(t),t);
>>  Title(' Vi du ve plot3D trong Matlab ')
```

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

1.4.1.2. In ấn trên màn hình đồ họa

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

>> print

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

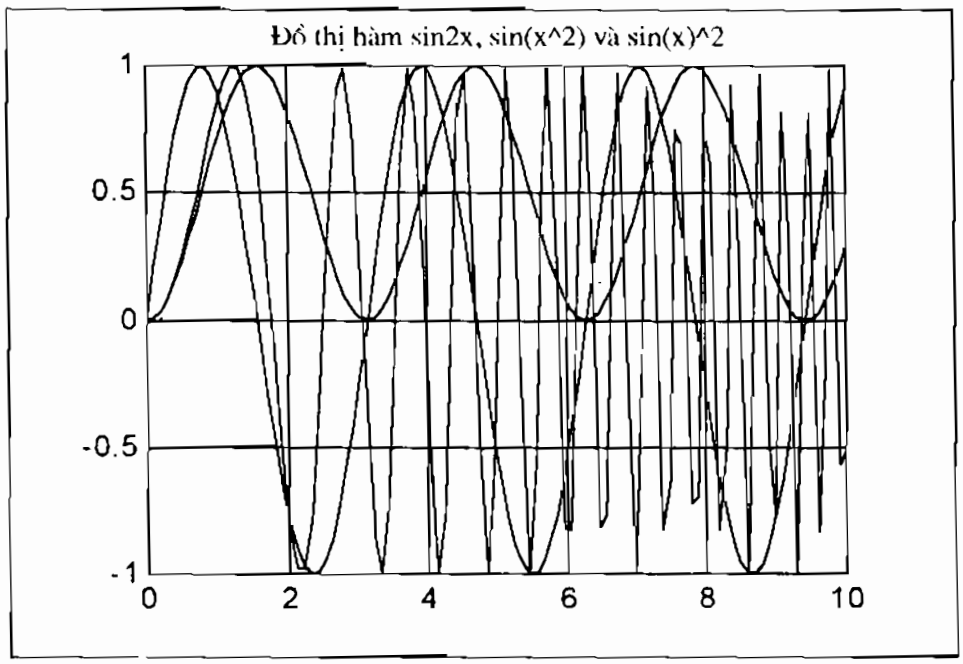
>> print filename

- 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 họa



Hình 1.5 *Hàm sin2x, sin(x)^2 và sin(x^2)*

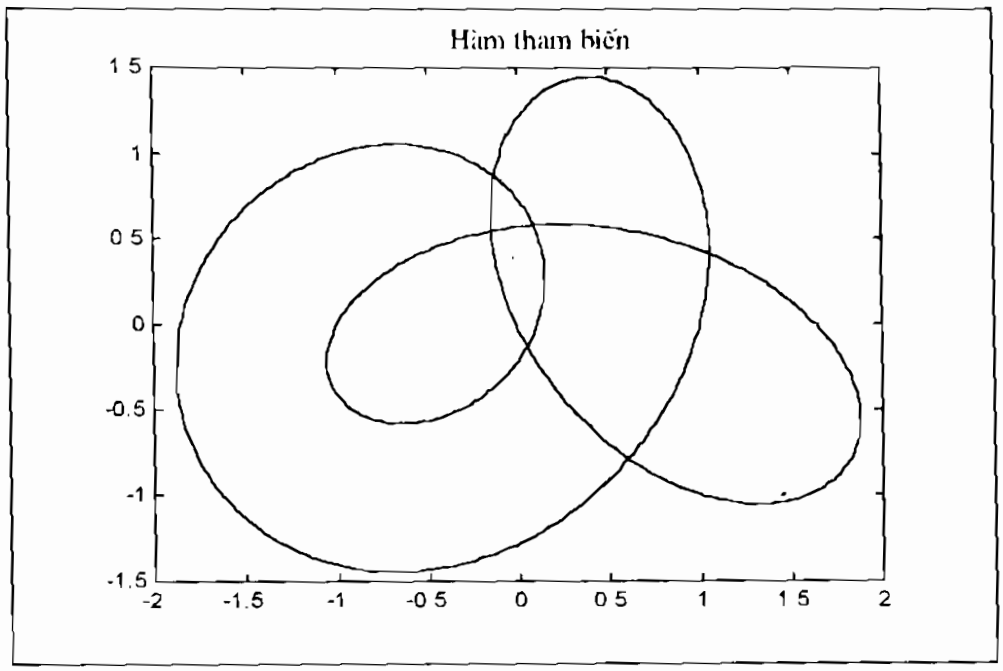
a) Ví dụ mô tả khả năng vẽ hàm đồ hoạ trong không gian 2D. Giả sử với hàm sin2x, cos(x)^2 và (cosx)^2 trong khoảng 0 < x < 10. Việc thao tác dễ dàng trên tập các lệnh sau.

```
>> hold on
>> x = linspace(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 khoảng từ [0 2*pi] cho kết quả như trên hình 1.6.



Hình 1.6 *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 sắp 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 hơn. Giảm giá trị sẽ cho âm trầm hơn.
saxis (str)	- Xét trục âm thanh theo chuỗi str.

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 MA TRẬN

Trong chương này, ta sẽ xem xét các biến đơn, các đại lượng vô hướng, 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, cần 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 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 truy nhập 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 bởi (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 bốn cách vào dữ liệu cho các biến vô hướng hay ma trận.

+ Liệt kê trực tiếp các phần tử của ma trận

+ Đọc dữ liệu từ một file dữ liệu

+ Sử dụng toán tử *(:)*

+ Vào số liệu trực tiếp từ bàn phím.

*** Một số 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, chúng 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 ];
```

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];

Có thể truy nhập 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.

Cũng 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:

```
>> 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. Đọc từ một file dữ liệu đã có

Thông qua lệnh load cho phép 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 số nguyên nằm giữa 2 số nguyên đó. Ví dụ: dấu " : " 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 nó 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 " : " 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 " : " còn được sử dụng để chọn các ma trận con từ một 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 " : " 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 hiểu 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 phẩy.

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 (zero)

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 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 (pascal(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

Khi một số toán tử có thể kết hợp trong một biểu thức số học, thì 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 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 lượng 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 tơ là véc tơ C 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:

```
D = [ 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:

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

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

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

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

2.4. CÁC PHÉP TOÁN 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:

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

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

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

2.4.2. Tích vô hướng 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: $C_{ij} = \Sigma\, A_{ik}B_{kj}$

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 cho thích hợp
>> 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)]   % Trích hai cột 1 và 2 của ma trận A
```

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° theo ngược chiều kim đồng hồ.

Ví dụ:

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

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:

$$A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 5 & -1 \\ 3 & 4 & 6 \end{bmatrix}$$

```
>> 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 CỦA LẬP 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 $\sqrt{-1}$
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 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
```

d) 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 số. 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 decimal bao gồm cả dấu phẩy động.
hex2dec (hstr)	Chuyển đổi số theo hệ hexa thành chuỗi string biểu diễn các số nguyên theo hệ decimal.
dec2hex (n)	Chuyển đổi số theo hệ decimal thành chuỗi string biểu diễn các số hệ hexadecimal.

3.2. CÁC HÀM TOÁN HỌC TRONG MALAB

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 $-\infty$
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 sin 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 tan của góc x, khi x có đơn vị đo là radian
asin(x)	Tính arcsin của x, khi x nằm trong khoảng [-1,1], hàm trả lại góc có giá trị radian trong khoảng $-\pi/2$ đến $\pi/2$
acos(x)	Tính arccos 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 arctang của x trong khoảng $-\pi/2$ đến $\pi/2$
atan2(x,y)	Tính arctang của y/x trong khoảng $-\pi$ 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 sin của x
cosh(x)	Hàm tính hyperbolic cosine của x
asinh(x)	Hàm tính nghịch đảo của hyperbolic sin của x
acosh(x)	Hàm tính nghịch đảo của hyperbolic cos của x
atanh(x)	Hàm tính nghịch đảo của hyperbolic tangent x

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 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.

Ta 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.

Cấu trúc ngôn ngữ, toán tử hay các bộ lệnh của *.m file, chúng tôi giới thiệu kỹ hơn ở phần sau. 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.

Sau đây là ví dụ đơn giản nhất đưa dòng lệnh HELLO ra màn hình cùng với một 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 cần 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 files 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	true

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 hai 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. Lệnh *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. Lệnh *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:

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 muốn 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*

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 HAI CHIỀU TRONG MATLAB

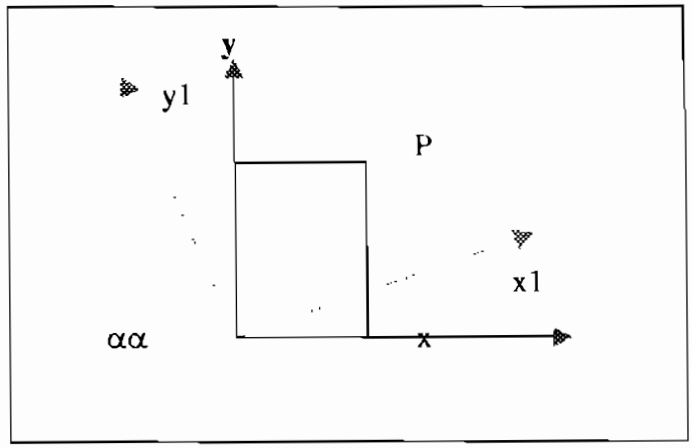
4.1. CÁC PHÉP BIẾN ĐỔI ĐỒ HOẠ

Nghịch đảo ma trận và định thức 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 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 các hàm tương tác trên cửa sổ đồ hoạ của Matlab cũng được liệt kê một cách chi tiết nhằm đem đến cho bạn một thư viện các hàm, hiệu ứng của các hàm 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.*

Trên 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 đã quay được biểu diễn bởi các đường đứt nét x_1 và y_1. x_1 và y_1, quay một góc α ngược chiều kim đồng hồ so với hai 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 dạng sau:

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

Quan hệ nghịch đảo của hai cặp toạ độ 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 một 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 :

$$P = B \cdot P_1 \quad . \qquad (4.6)$$

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:

$$\mathbf{B.A} = \begin{vmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{vmatrix} \begin{vmatrix} \sin\alpha & \cos\alpha \\ -\sin\alpha & \cos\alpha \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ 0 & 0 \end{vmatrix} \qquad (4.7)$$

tương tự ta có:

$P_1 = ABP_1$

Một lần nữa ta thấy rằng **AB** chính bằng ma trận đơn vị và có thể phát biểu ma trận **B** là ma trận nghịch đảo của **A** và biểu diễn là $\mathbf{A^{-1}}$.

Định nghĩa ma trận nghịch đảo được phát biểu 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**.

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

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:

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

Từ phương trình (4.8) cho kết quả sau:

$$b_{11} = \frac{a_{22}}{a_{11}a_{22} - a_{21}a_{12}} \qquad (4.9)$$

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

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

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

Từ đây chúng ta dễ dàng thu được ma trận nghịch đảo B từ A. Trong Matlab, hàm nghịch đảo được viết sẵn trong thư viện 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 B, 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 tham 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 này, toạ độ của điểm P được xác định bởi hệ 3 giá trị x, y, z và chúng ta phải xác định 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.

Trên hình 4.2 với công thức được học trong phần đồ hoạ, khi ta quay hệ toạ độ xung quanh trục z một góc ψ, điểm x*, y*, z* 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 đã quay 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 một góc ϕ. Giá trị toạ độ cuối cùng thu được sẽ là:

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

Kết quả 3 tiến trình quay:

$$\mathbf{P_1 = A\,B\,C\,X = DX} \qquad (4.16)$$

```
% 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

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

Ví dụ:

Khi cho điểm P với các giá trị toạ độ [2. 5. 3] cho các góc quay là 30^0, 45^0, 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 ) * X,
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 đồ hoạ được mô tả trong chương này có một ý nghĩa hết sức quan trọng trong ứng dụng của các lĩnh vực kỹ thuật hiện đại như đồ họa máy tính hay robotics. Một nhóm các lệnh được sử dụng thường xuyên gọi là Affine transformation. Chúng bao gồm: translation, rotation, scaling... ở đây ta hãy 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 một xích ghép nối liên tục (concatenation) của vài phép biến đổi. Điều đó được thực hiện bởi một loạt 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 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:

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

Ưu việt của việc biểu diễn theo hệ toạ độ thuần nhất là chúng cho phép biểu diễn các điểm ở xa vô cùng.

Việc biểu diễn các điểm này chỉ thông qua $W = 0$ với x_1, y_1 là một số hữu hạn bất kỳ. Bạn đọc có thể tham khảo các tài liệu về đồ hoạ máy tính và các bài toán chiếu phối cảnh, ở đây ta chỉ nói về phương pháp tạo ra các phép biến đổi trong không gian 2D.

Với h. 4.2 các điểm tạo nên hình vuông được cho bởi các giá trị

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

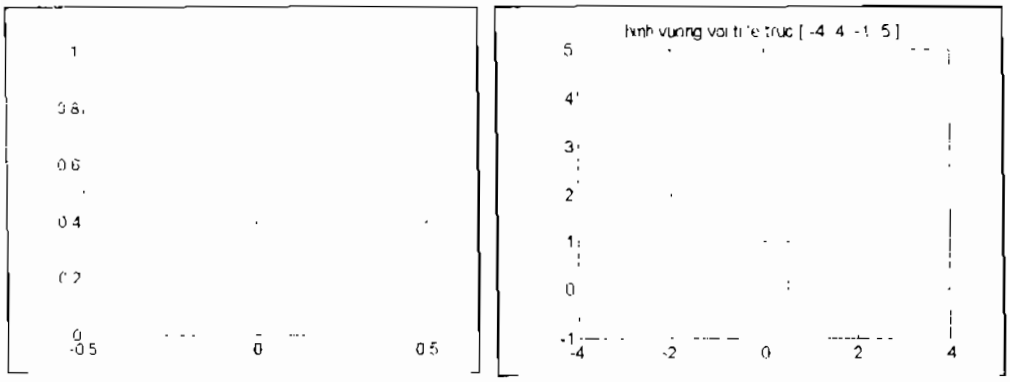
Như đã thấy, ở đây giá trị toạ độ thứ 3, W được cho bằng 1. Đó là một giải pháp của kỹ thuật đồ họa và trong Matlab viết như sau:

```
>> 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 hai vector P_1 dùng cho việc đóng hình vuông)

Việc tạo ra 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 4.2 ***a. Hình vuông chuẩn*** ***b. Hình vuông sau khi thay đổi tỉ lệ trục toạ độ***

4.2.2 Phép chuyển dịch tịnh tiến

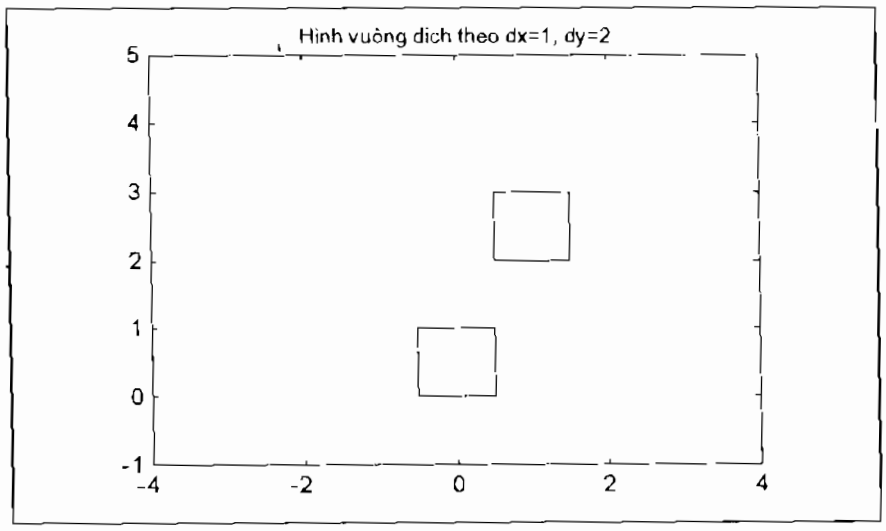
Hàm chuyển vị tịnh tiến với các khoảng dx và dy song song trên 2 trục x và y.

```
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
```

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

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 ( [-4   4  -1   3] )
>> title (' Hình vuông thay đổi vị trí theo dx và dy ')
```



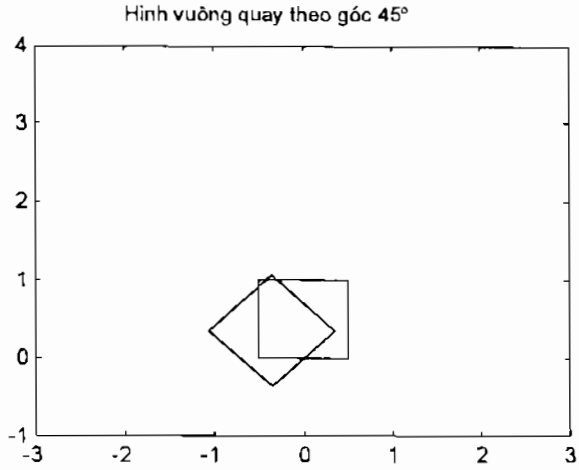
4.2.3 Phép quay

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

Function R = rotate (fi)

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

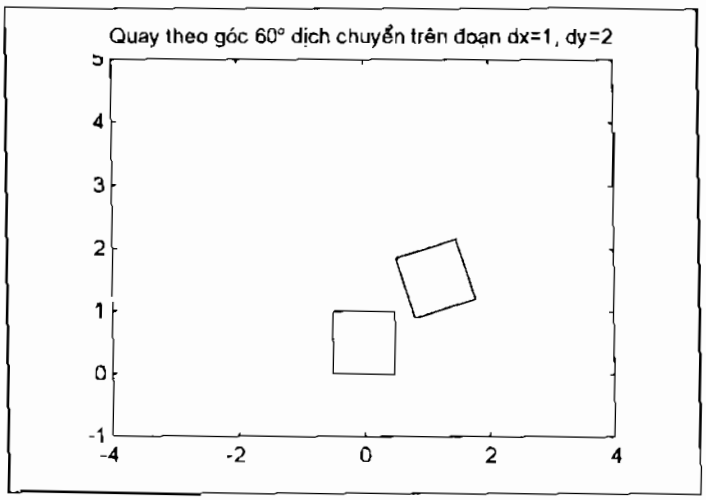
```
R =   [cos(fi) - sin(fi)    0
       sin(fi)   cos(fi)    0
          0      0          1]
```



Hình 4.4 ***Hình vuông quay một góc 45 theo gốc toạ độ***

Hình 4.4 thu được thông qua dòng lệnh Matlab như sau, với góc quay $\theta = 45^0$.

```
>> P = rotate (45*pi/180)*square
>> plot (P( 1 , : ), P ( 2 , : ) ), axis ([-3  3  -1  4])
>> title ('Hinh vuông quay theo goc 45 ')
```



Hình 4.5 ***Tổ hợp của hai phép biến đổi***

Hình 4.5 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  4])
>> title ('Hình vuông quay và dịch chuyển ')
```

4.2.4 Phép tỉ lệ (Scaling)

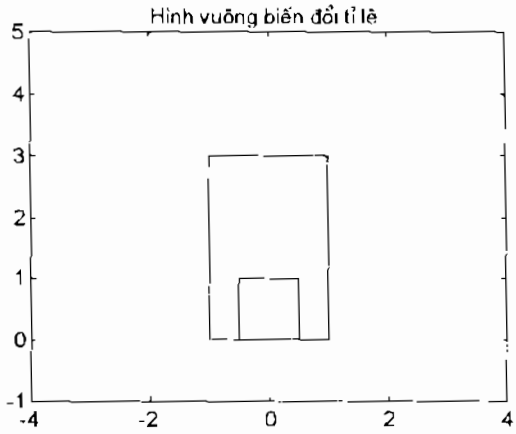
Hàm dưới đây cho phép biến hình theo một tỷ lệ nhất định. Việc biến đổi tỷ lệ được thực hiện qua phép nhân ma trận S với Sx, Sy là hai hệ số biến đổi.

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

```
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 )
```

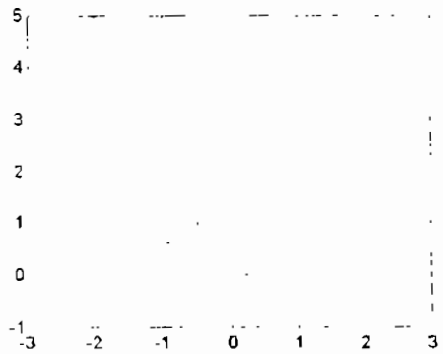


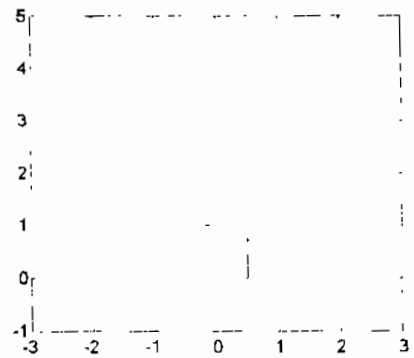
Hình 4.6 ***Phép biến đổi tỉ lệ***

Hình 4.6 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 kết hợp lẫn với nhau một cách dễ dàng. Việc chuyển đổi thứ tự 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 ([-3  3  -1  5])
>> title ('Hình vuông quay')
```





```
>> P = translate(1,1) *rotate(30*Pi/180)*Scale(2,2) *square
>> plot (P( 1 , : ), P ( 2 , : ) ), axis ([-3  3  -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 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 (xi,yi) với y là các điểm trên trục tung và xi 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 với chỉ số tương ứng xác định bởi chương trình.
plot (x,A)	Vẽ ra theo các cột của A với 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 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,	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

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

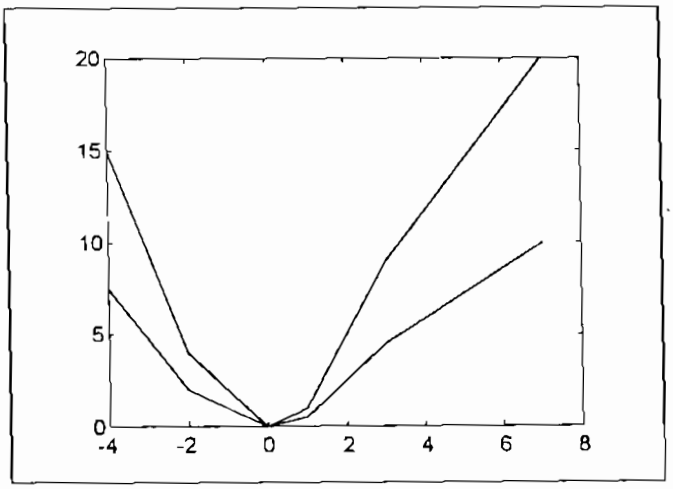
Kiểu đường	Kiểu đường	Mầu sắc	Mầu sắc
điểm	- đường liền nét	Y vàng	C xanh lá mạ
* sao	-- đường đứt nét	G xanh lá cây	W trắng
X chữ cái x	-. đường chấm gạch	M đỏ tươi	R đỏ
O chữ cái o	: đường chấm	B xanh lam	K đen
+ dấu cộng			

Kiểu đường thẳng có thể là tổ hợp của hai 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:

a. 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 );
```



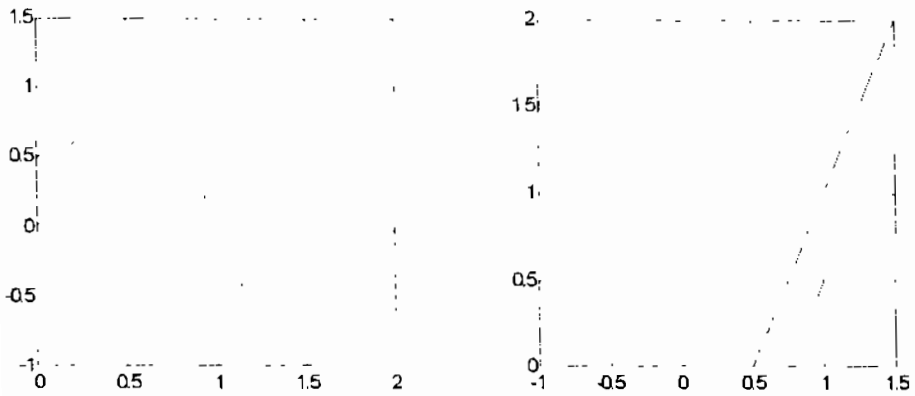
Hình 4.9 ***Đồ thị hàm y và y/2 theo x***

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 của A và X

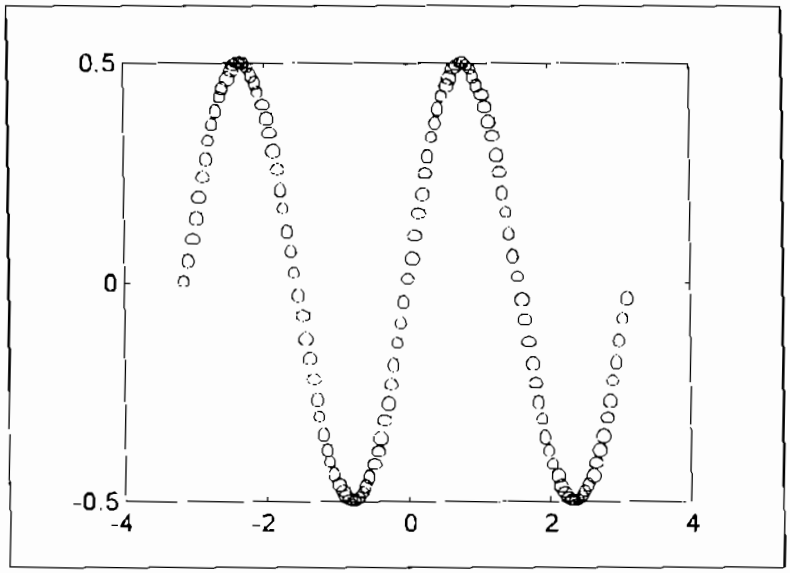
```
>>x = 0 : 0  1 : 2;
>>A = [sin (pi*x); 0.5 + 0.5*x];
>>plot (A,x)
```



Hình 4.10 ***Đồ thị của ma trận A trên trục x và x trên A***

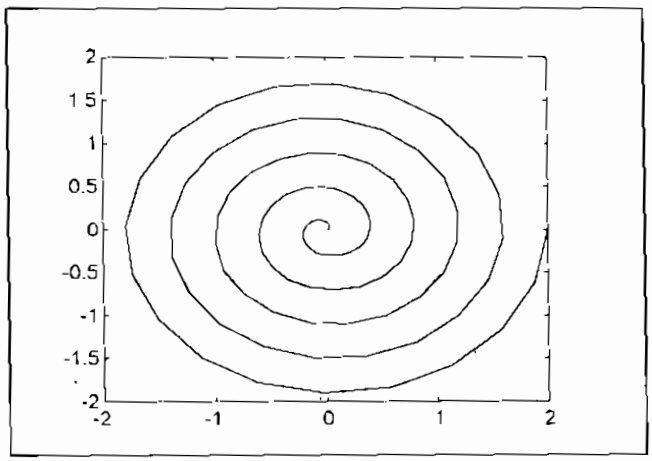
c) Hàm y = sin(x). cos(x) với các điểm là các vòng tròn nhỏ

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



Hình 4.11 *Hàm y = sin(x) · cos(x)*

d) Hàm plot với tham số phức.



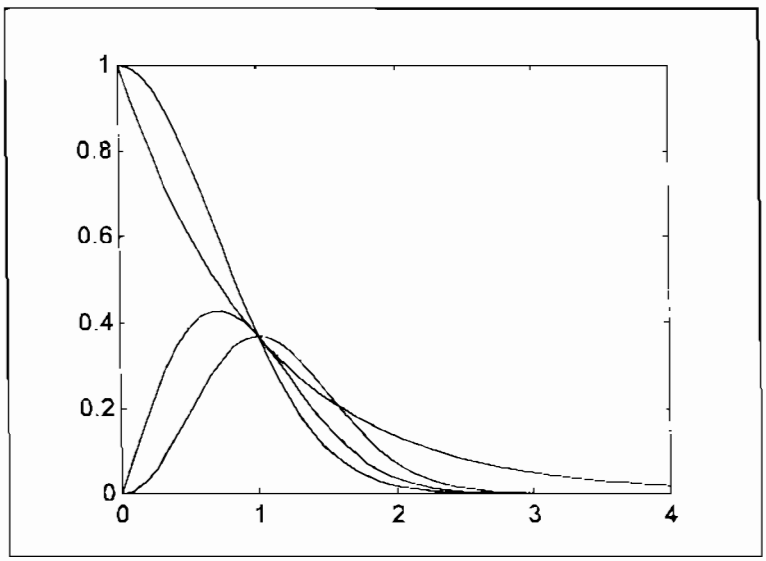
Hình 4.12 *Số phức được biểu diễn dưới dạng xoắn ốc*

```
>> R = linespace (0,2);                    % tạo vector
>> tt = linespace (0,10*pi);               % tạo vector của góc
>> [x,y] = plot2cart(tt,r);                % chuyển toạ độ
```

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 4.13 ***Đồ thị của các hàm e1, e2, e3, e4***

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)
```

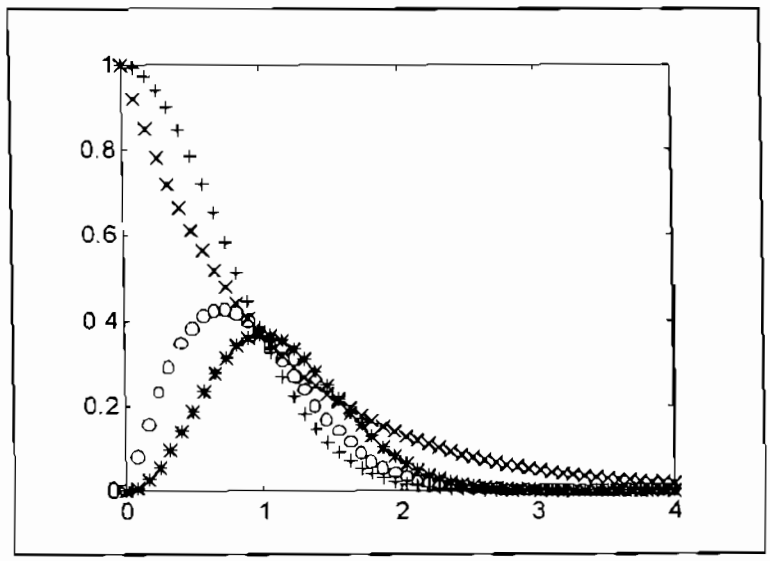
với

u = 50

a = 0

b = 30

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

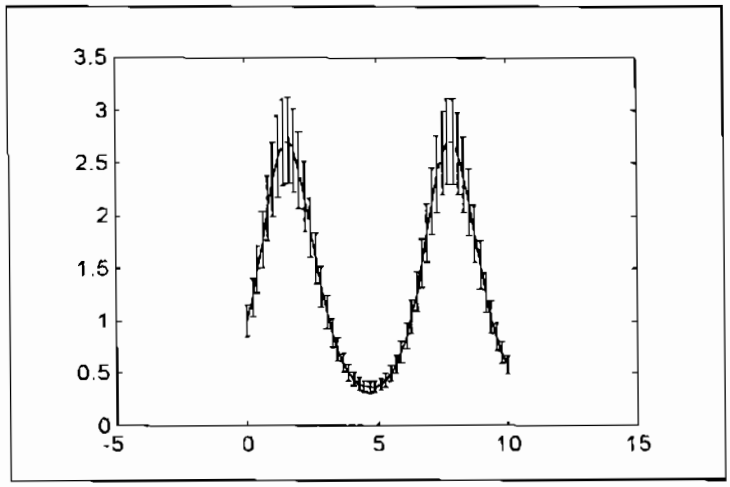


Hình 4.14. ***Vẽ đồ thị cùng với khoảng sai số***

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 l, 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 khoảng sai lệch 15%***

b. 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

c. Hàm đồ hoạ

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

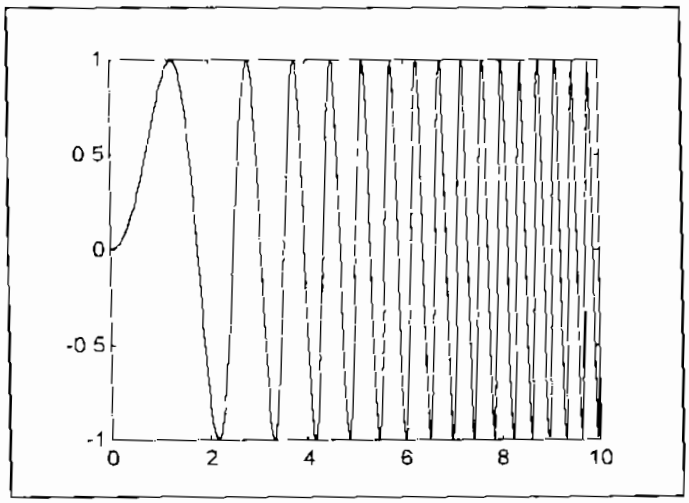
Vector lim = [Xmin Xmax] dùng để giới hạn khoảng xác định của đồ hoạ. Nó có thể bao gồm 4 thành phần trong đó thành phần thứ 3 và 4 là khoảng xác định trên trục y. Nếu biến str không khai báo trong hàm thì chương trình sẽ tự lấy các giá trị mặc định về kiểu đường hay mầu cho phần đồ hoạ.

fplot (fku, 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 vẽ với toạ đồ Đề các. Tuy nhiên một số bài toán trong kỹ thuật lại yêu cầu các hệ toạ độ khác. Để đáp ứng nhu

cầu đó Matlab cung cấp một 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 tử của vector theta là các biến đo bằng radian và các phần tử của vector r là khoảng cách đến điểm gốc.

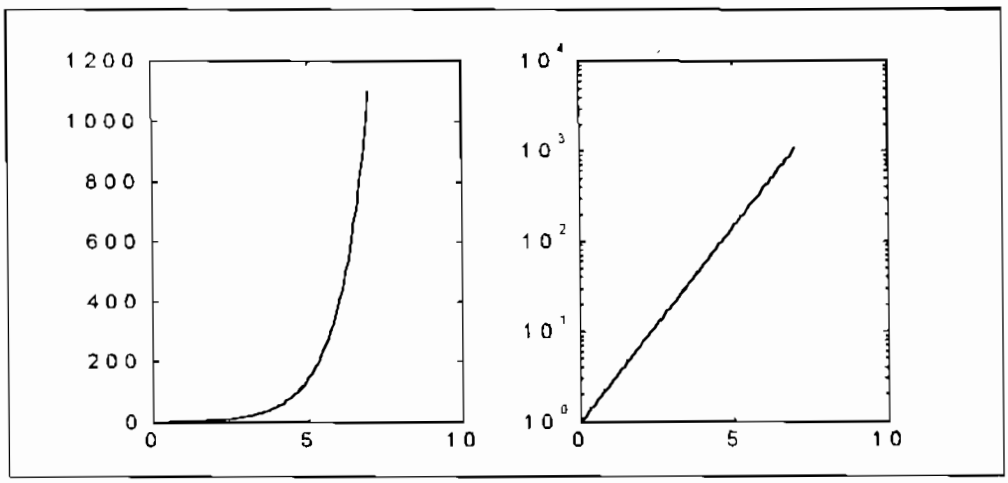
semilogx (x, y) - Cho phép vẽ trên hệ toạ độ của trục loga, thang đo log10 được sử dụng cho trục x. Điều đó cũng tương đương với việc chúng ta viết plot (log10(x,y) nhưng sẽ không có lỗi với cả trong trường hợp log10(0).

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
```

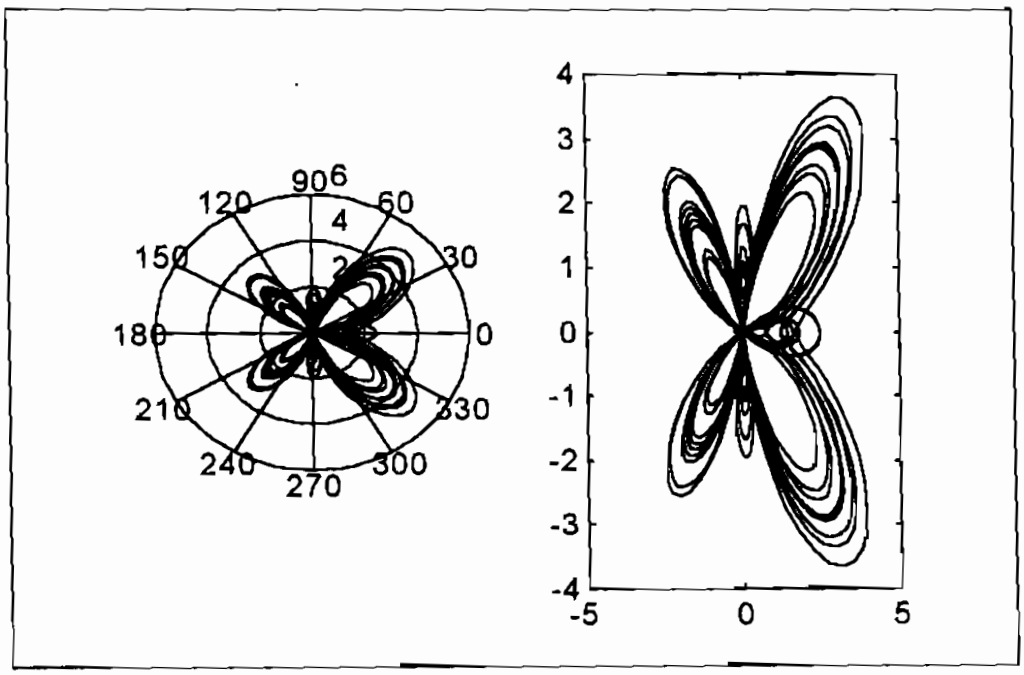


Hình 4.17 ***Đồ họa trên hệ trục Đề-các thang đo log10***

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

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

```
>> 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 4.18 *Đồ họa trên hệ toạ độ cực*

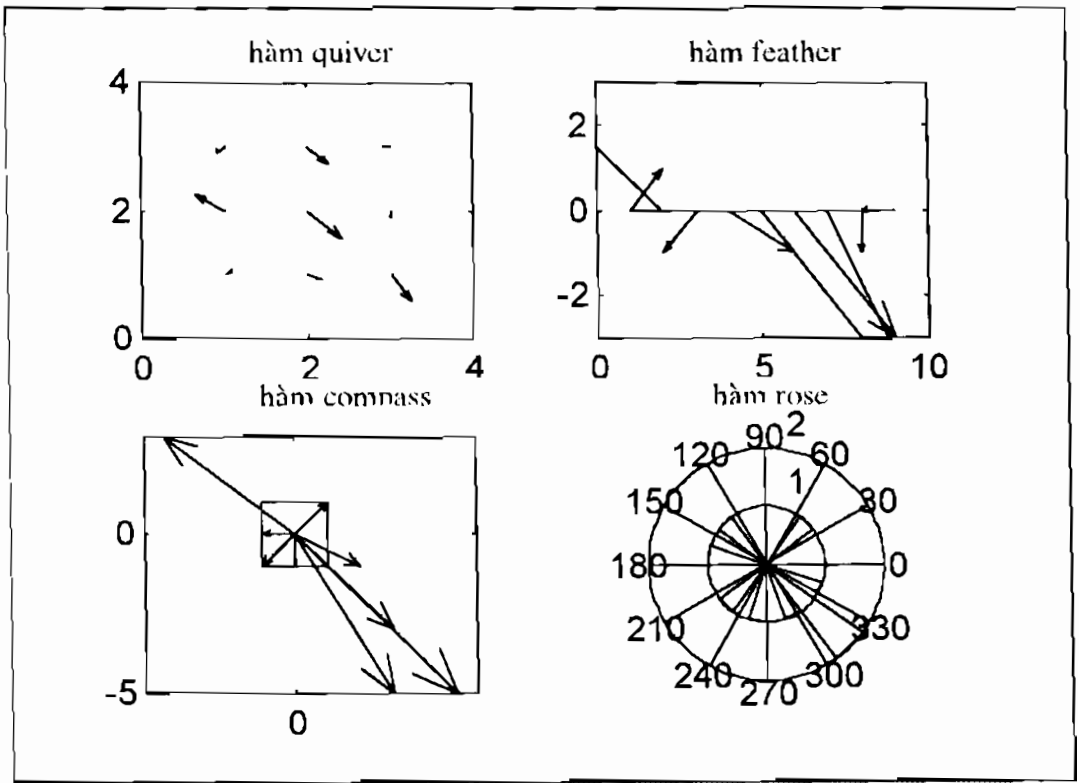
4.3.3 Mặt phẳng đồ hoạ cho số phức

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 (x, y , ... , s)	Vẽ mũi tên như trên nhưng hệ số tỷ lệ được cho bởi giá trị s. Nếu s không được khai báo thì giá trị mặc định là 1
quiver (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;      [ (1 + i)    (2 - i)      (3 - 5*i)
z =       (-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 4.19 *Đồ hoạ thu được từ các hàm Quiver, Feather, 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 thực 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 một vài trạng thái hold là on, trường hợp còn lại là 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, n, 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 đương subplot(1,1,1)

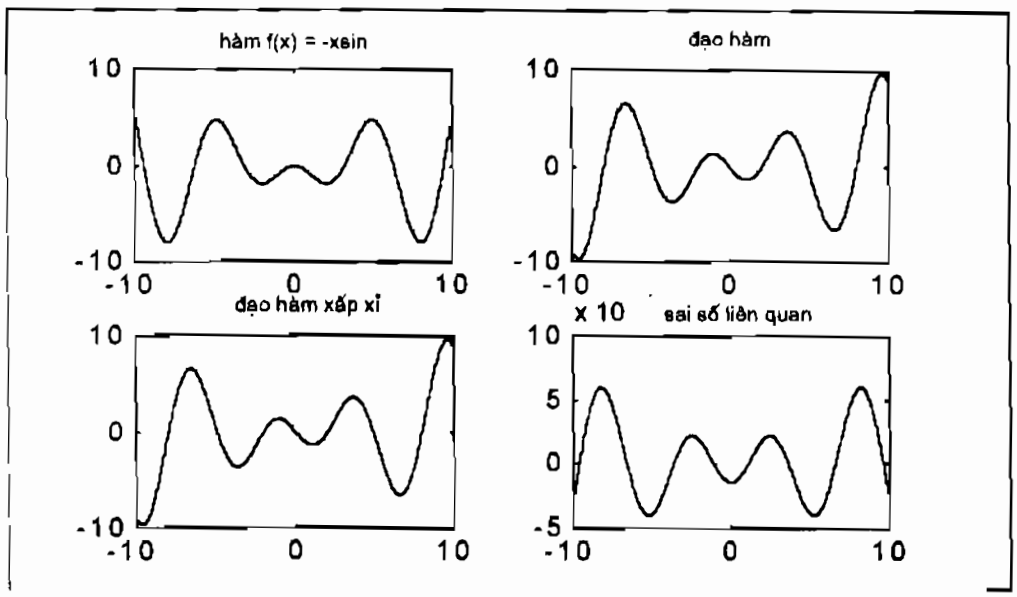
Ví dụ:

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

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

Ví dụ:

Hàm Mandelbrot và việc hiển thị bằng ba 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 tập của Mandelbrot. Số lặp lại tại mỗi điểm c của mặt phẳng phức được ghi vào ma trận Madelbrot cùng với việc giải vector

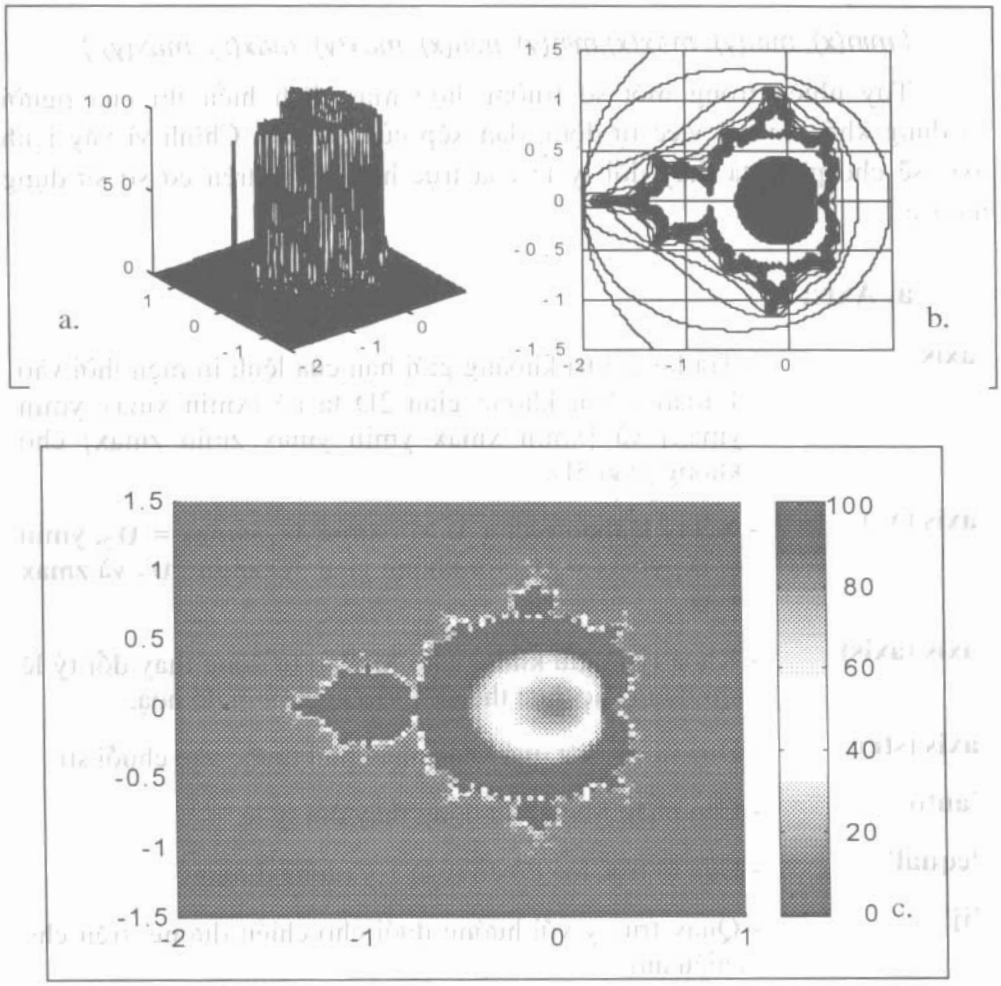
```
clear;
epsilon = 1e-14;                          % sai số
renum=input('số điểm thực renum = ');
imnum=input('số điểm  ảo imnum = ' );
remin=-2; immin=-15;                      % khoảng chặn dưới
remax=1; immax = 1.5;                     % khoảng chặn trên
reval1 = linspace ( remin,  remax, renum );
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 công 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 ba  mầu
axis ( [-2  1  -1.5  1.5] );
```



Hình 4.21 ***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 đổi tỷ lệ kích thước sao cho thích hợp 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 sẽ cho phép 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 υ với xmin υ_1, xmax = υ_2, ymin = υ_3 ymax = υ_4 với không gian 3D zmin = υ_5 và zmax = υ_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 phím bên trái chuột lên trên màn hình. Còn kích phím bên 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 màn hình

zoom - Chuyển trạng thái giữa on và off

Ví dụ:

a) Ví dụ cho việc biểu diễn đường tròn

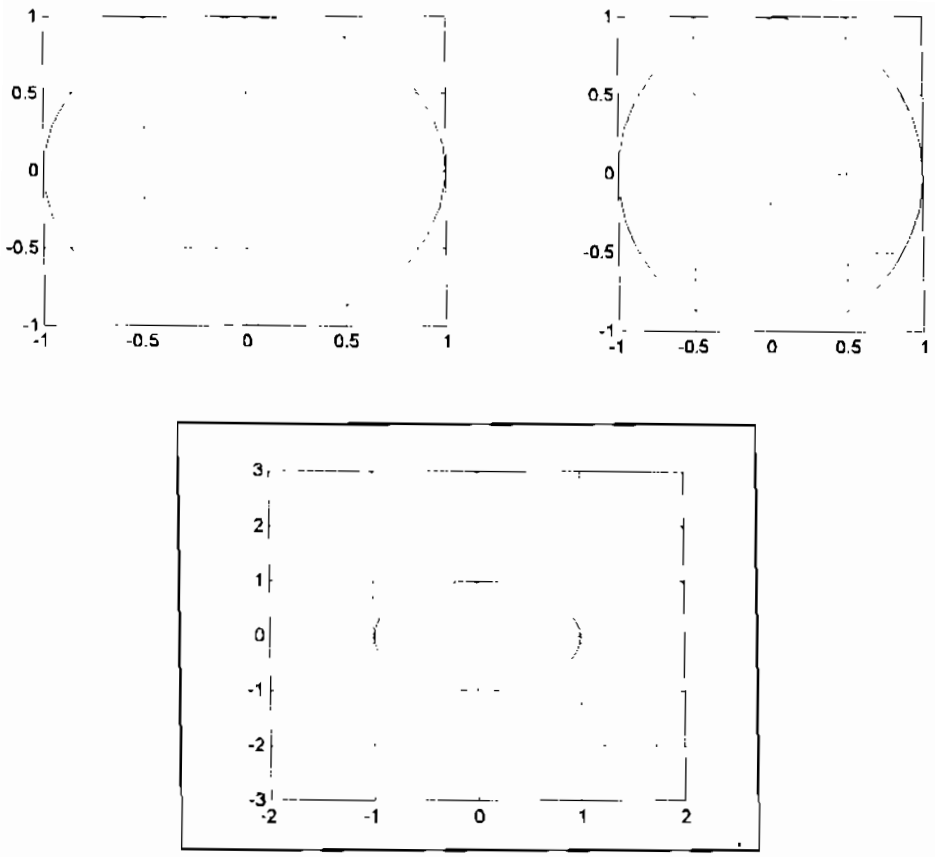
```
>> t = 0 : 0.2 : 2*pi + 0.2;
>> x = sin (t);
>> y = cos (t);
>> plot ( x,y,1,-1 ); grid on
```

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

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

c) Bộ lệnh sau cho ra màn (hình 4.22c)

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



Hình 4.22 ***Đồ họa Zoom-on***

a) Trước lúc căn chuẩn b) Sau khi căn chuẩn square c) Sau khi trở lại trạng thái normal

4.3.6 Văn bản trong màn hình đồ hoạ

Phần này 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,yeó 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 hai 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 bởi 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

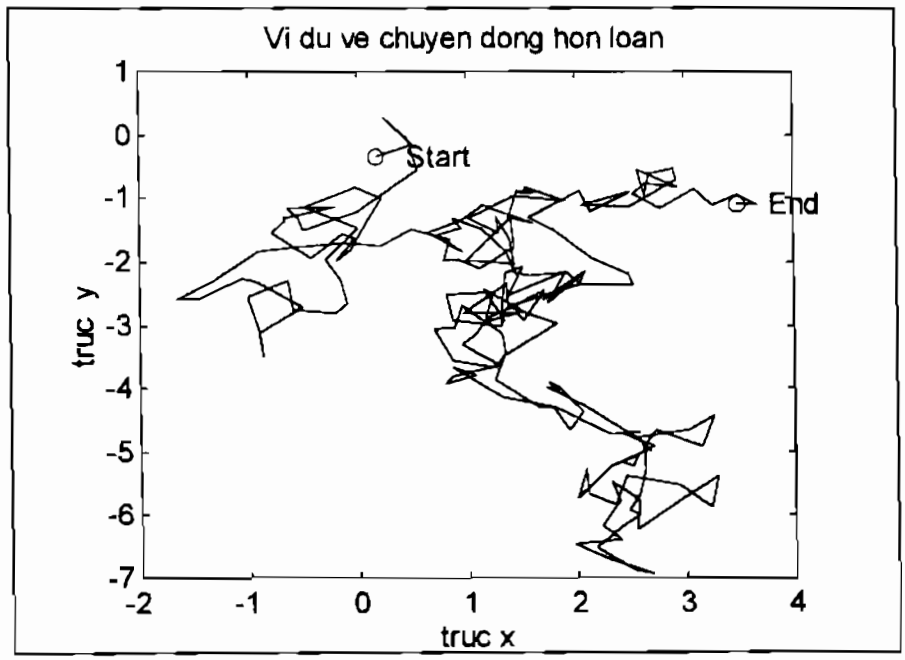
Lệnh chuyển đổi từ số sang chuỗi có thể được dùng trong việc in bao gồm sprinf, num2str, int2str.

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 động hỗn loạn');
```



Hình 4.23 ***Chuyển động hỗn loạn với bước hoạt động n = 200***

4.3.7. Đọc dữ liệu từ màn hình 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**

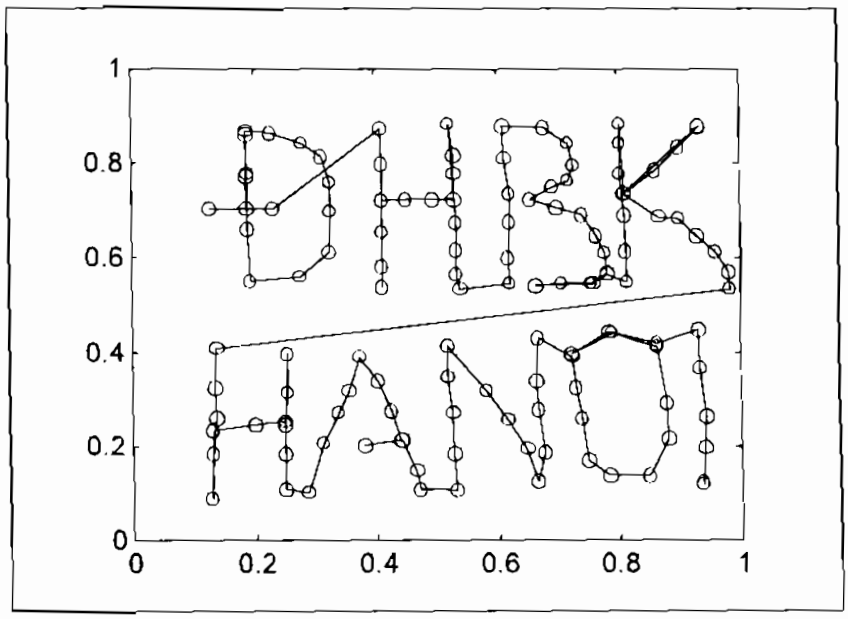
Dừng Matlab cho đến khi tác động lên chuột hay bàn phím. Nếu ấn chuột thì lệnh sẽ trả giá trị 0 nếu bàn phím sẽ trả giá trị 1.

Ví dụ 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.

Ví dụ:

```
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 phim
delete ( n)
```



Hình 4.24 ***Tương tác màn hình đồ hoạ bằng chuột và bàn phím***

CHƯƠNG 5

ĐỒ HOẠ TRONG KHÔNG GIAN BA CHIỀU

5.1. CÁC HÀM TẠO LẬP BIÊN DẠNG (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 sử dụng trên lưới tứ giác.

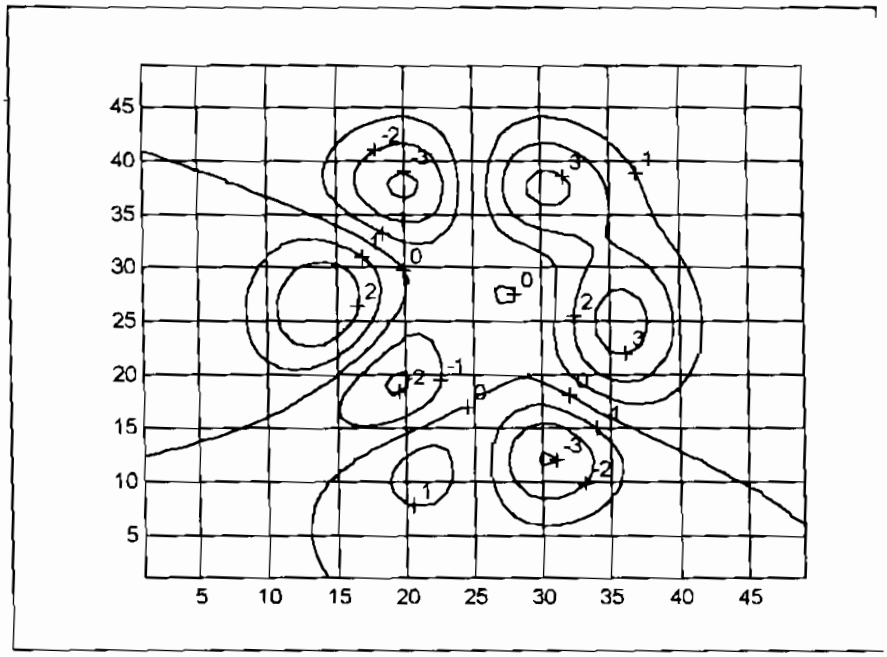
Các hàm tạo lập contour gồm có:

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 z là ma trận m x n thì tỷ lệ trên các trục tương ứng sẽ là n, m.
contour (z, n)	- Vẽ contour cho n cấp độ. Nếu n không xác định thì hàm sẽ lấy giá trị mặc định n = 10.
contour (z,v)	- Vẽ contour với cấp độ được xác định bởi một vector v
contour(x,y,z)	- Vẽ 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 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 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 hai dòng chứa dữ liệu vẽ.

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

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.

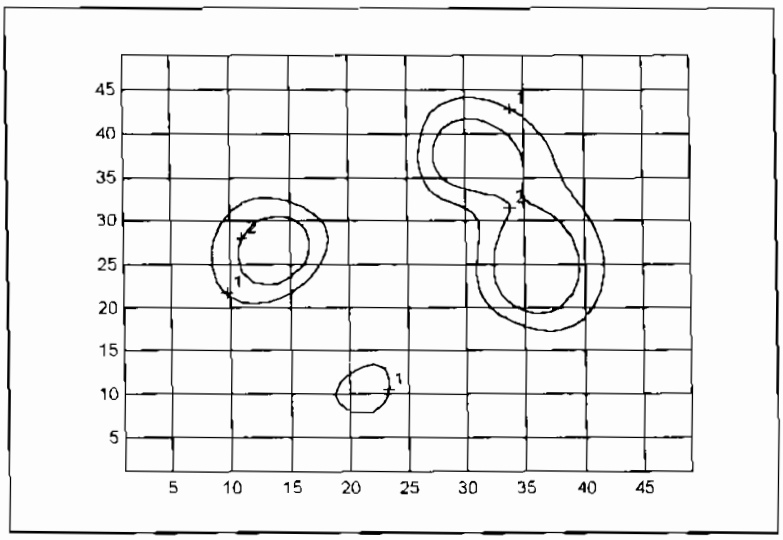


Hình 5.1 *Đồ thị contour cho bởi ví dụ a*

```
>> 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 ta sử dụng zsmall. Chương trình chỉ thể hiện 2 mức độ như vậy zsmall lấy 2 giá trị 1 và 2.

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



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

5.2. LƯỚI - GRID

Để vẽ được các contour trước hết khi xác định lưới của vùng, nơi ta sẽ vẽ đường contour. Vùng này 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 khoảng cách của các phần tử trên x và y là không bằng nhau, lưới được tạo ra bởi lệnh:

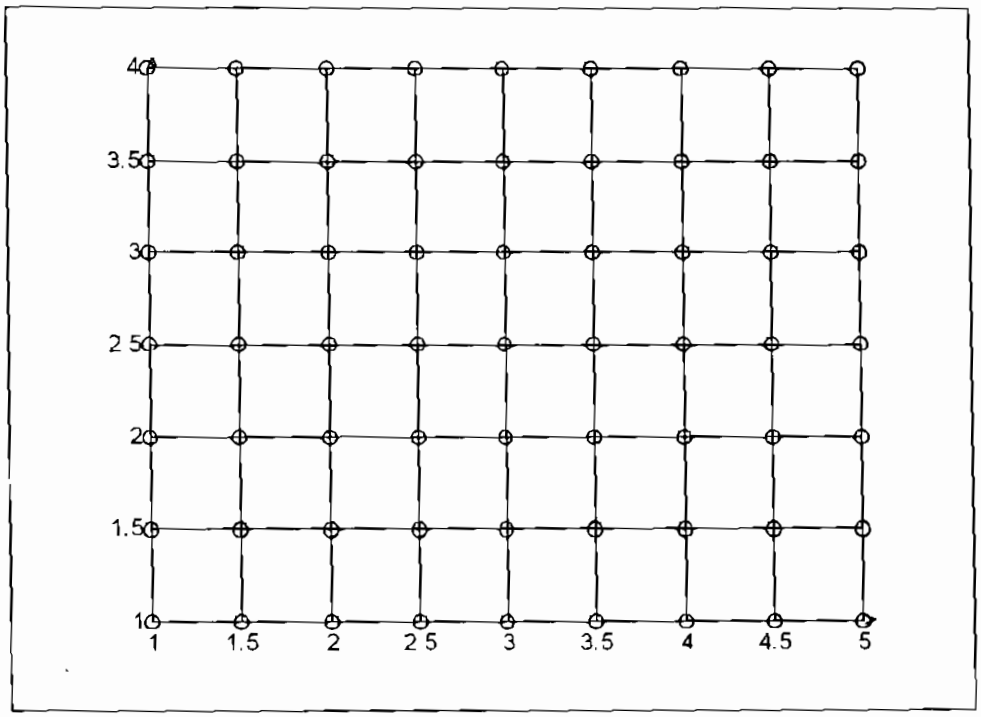
>> [u v] = meshgrid (x,y);

Trong đó, 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 5.3cho thấy ảnh của lưới [u , v]



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

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

Lệnh tạo lưới

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

đưa ra ma trận định dạng lưới theo toạ độ x,y từ 2 vector tương ứng 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)

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) × (n+1)

>> sphere(n)

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

Ví dụ

Cần định nghĩa lưới U,V trên 1 đơn vị mặt vuông với 5 điểm 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

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

- Z = f(uv,)

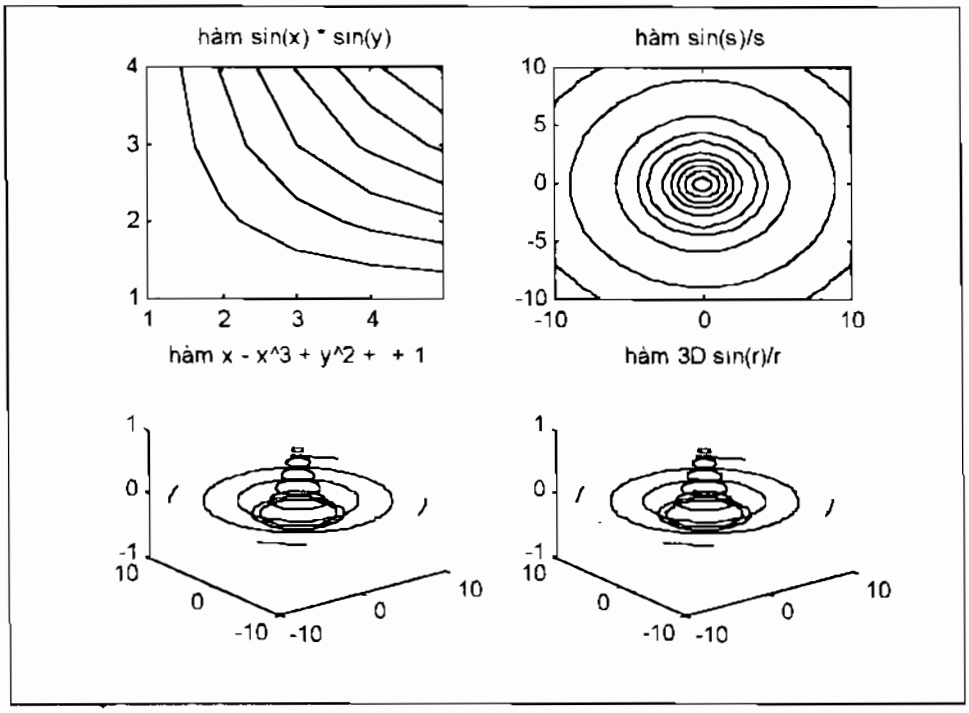
a) Giả sử cần vẽ đường contour của 3 hàm sau:

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

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

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

Đoạn chương trình sau tạo 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 a***

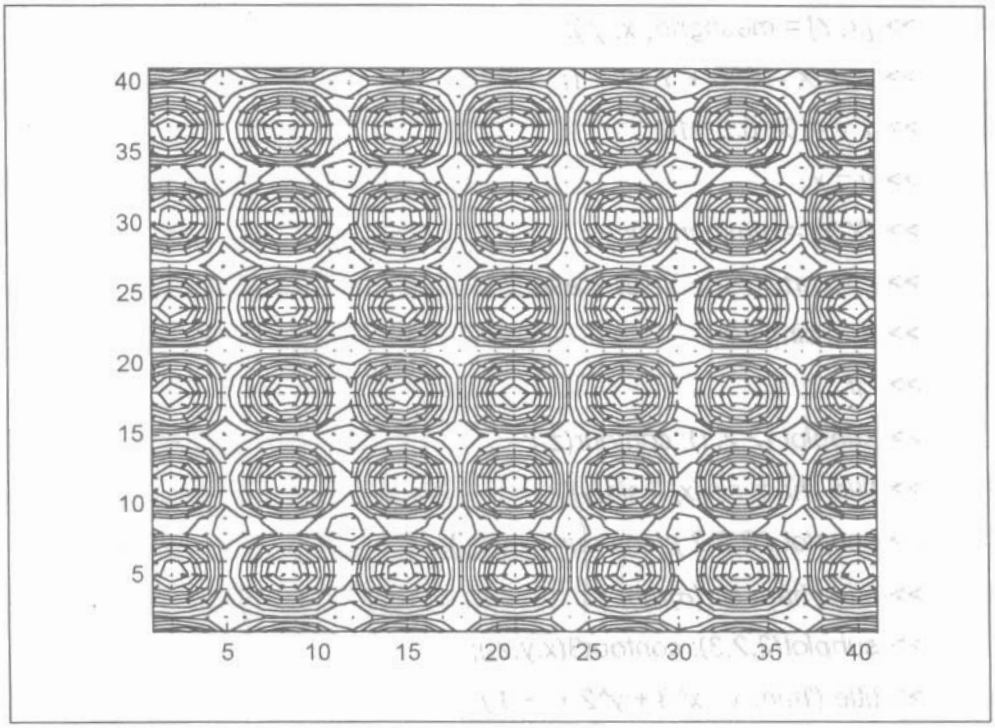
Phần chương trình nguồn của ví dụ a

```
>> X = 0 : 0.2 : 3*pi;
>> Y = 0 : 0.25 : 5*pi;
>> [X,Y] = meshgrid (x,y);
>> z, = 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 làm sáng tỏ hình ảnh của hàm, có thể vẽ contour như là vẽ gradients. Giá trị gradient được tính bởi lệnh gradient và có thể được đưa ra màn hình bởi lệnh quiver.

```
>> [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;
```



Hình 5.5 *Mô tả cho ví dụ b*

Hình 5.5 cho bởi đoạn mã chương trình viết ở trên.

5.3. ĐỒ HOẠ BA 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.

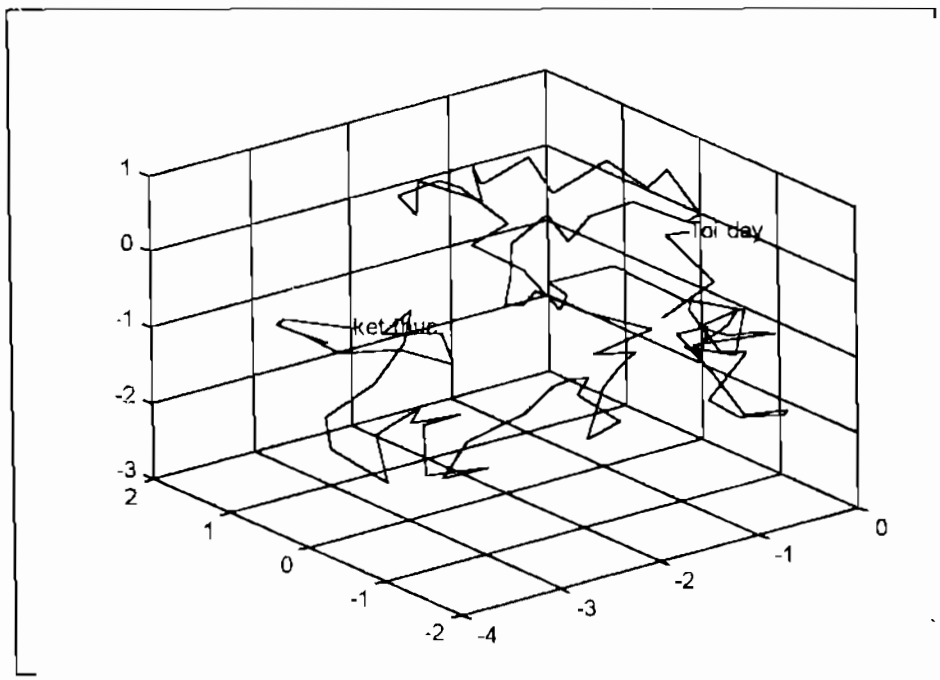
plot3(x,y,z,str) - Vẽ đồ hoạ tương tự lệnh trên với màu và kiểu đường được xác định bởi biến str.

plot3 (x_1, y_1, z_1, str_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

Chương trình mô phỏng chuyển động hỗn loạn 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)	- Tương tự như lệnh comet trong không 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).
comet3(x,y,z,p)	- Cho ra tiến trình vẽ mô phỏng tương tự như trên với độ 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 xác định 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 lân cận trong lưới được nối với nhau bởi các đường thẳng. Đồ hoạ được vẽ trong không gian 3D với góc chiếu phối cảnh, trong đó phần tử zij là chiều cao trên lưới grid(Uij, Vij)).

- Điểm nhìn được lấy tự động để có 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 hai 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	- Matlab không vẽ các đường khuất sau mặt lưới tạo bởi lệnh mesh
hidden off	- Matlab sẽ 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 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)	- Tra 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 nói trên.

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

```
>> clf;
>> subplot(2,2,1); mesh( Matlabmatrix );
>> title('gcs nhin 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) Dùng Matlab để mô tả các mặt hình học sau:

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

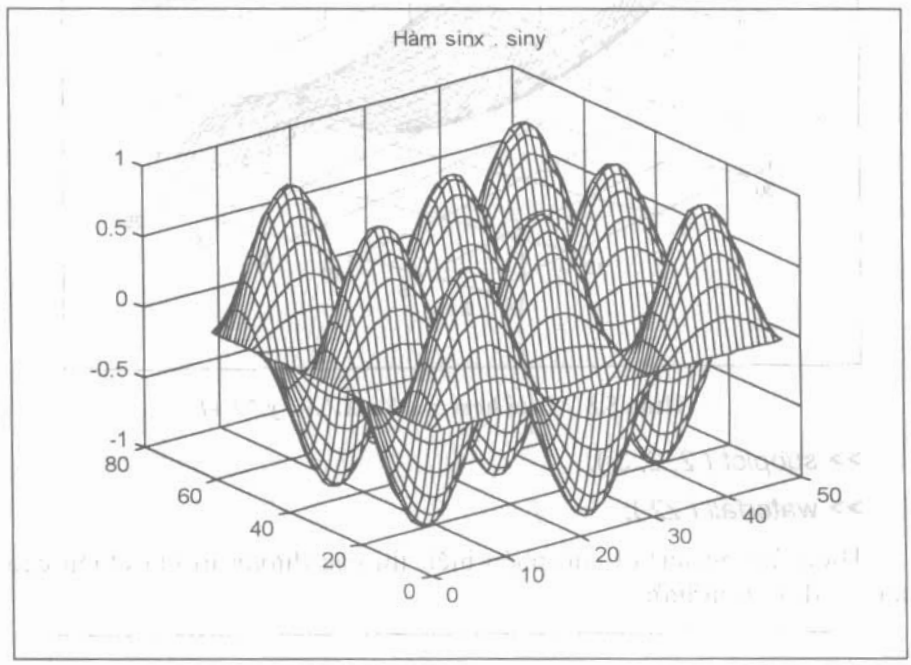
$z_2 = f(x,y) = +x = x^3 + y^2 + 1 \; x, y \in [-3, 3]$

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

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

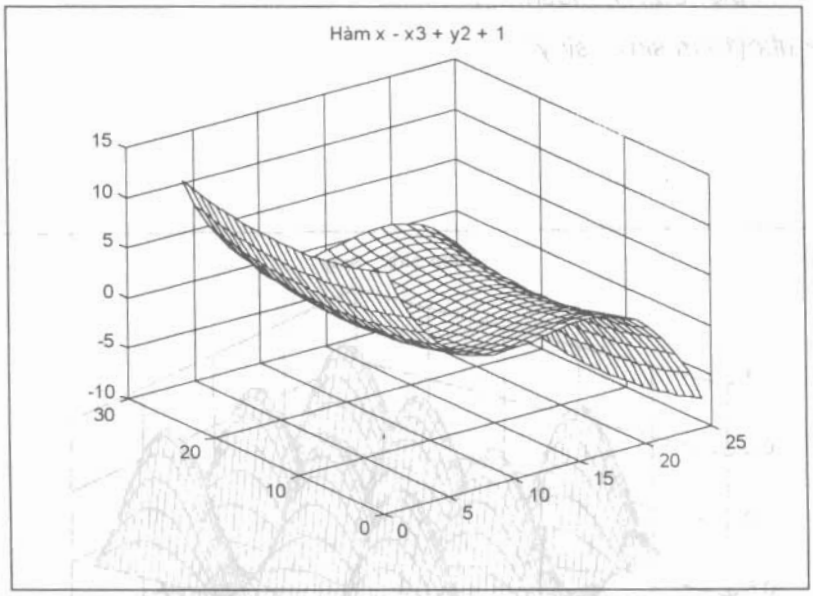
```
>> x = 0 : 0.2 : 3*pi;
>> y = 0 : 0.25 : 5*pi;
>> [X,Y] = meshgrid(x,y);
>> z1 = sin(X) . *sin(Y);
```

```
>> subplot(2,2,1); mesh(z1);
>> title('hàm sinx . siny');
```



Hình 5.7 *Mô tả hàm sin x * sin y*

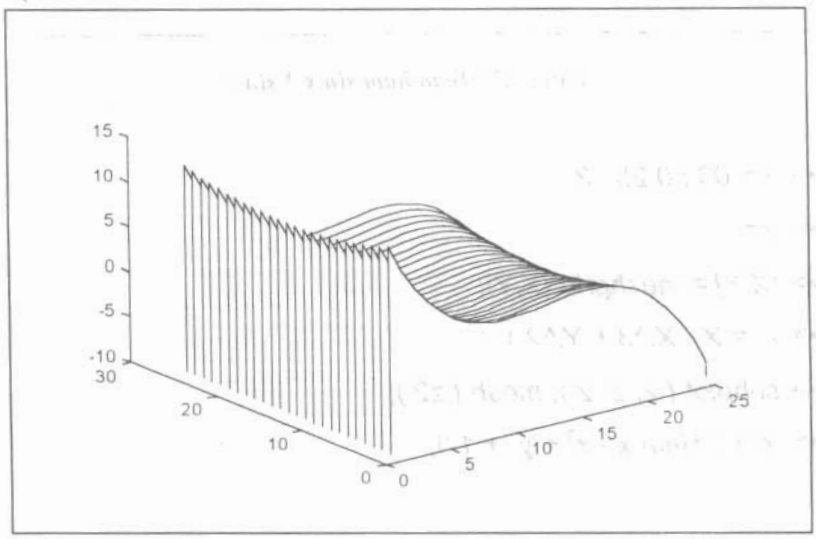
```
>> x = 03 : 0.25 : 3
>> y = x
>> [X,Y] = meshgrid (X,Y);
>> z2 = X - X.^3 + Y.^2 + 1;
>> subplot ( 2, 2, 2 ); mesh ( z2 );
>> title (' Hàm x - x^3 + y^2 + 1 ');
```



Hình 5.8 *Mô tả hàm z = X - X.^3 + y.^2 +1*

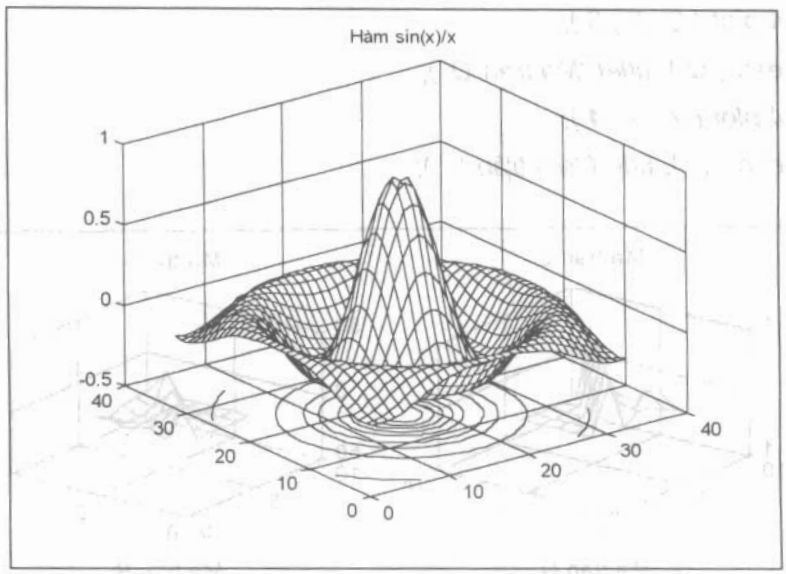
```
>> subplot ( 2, 2, 3 );
>> waterfall ( z2 );
```

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 ' );
```

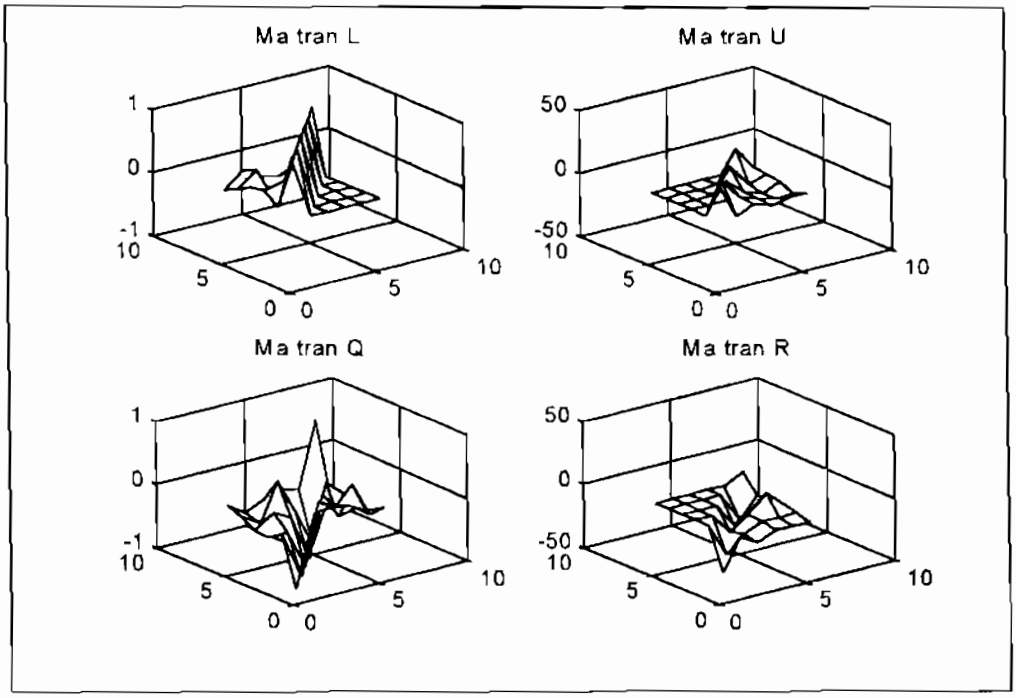


Hình 5.10 *Đồ thị lưới của hàm sin(x) / x*

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 như trên 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 các ma trận LU và QR*

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.

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

5.5. ĐỒ HỌA 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 thì z là ma trận tương ứng m x n và bề mặt được định nghĩa bởi xi, yj và zij.

- 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.

- 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(...) trừ 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], 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.

- 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.

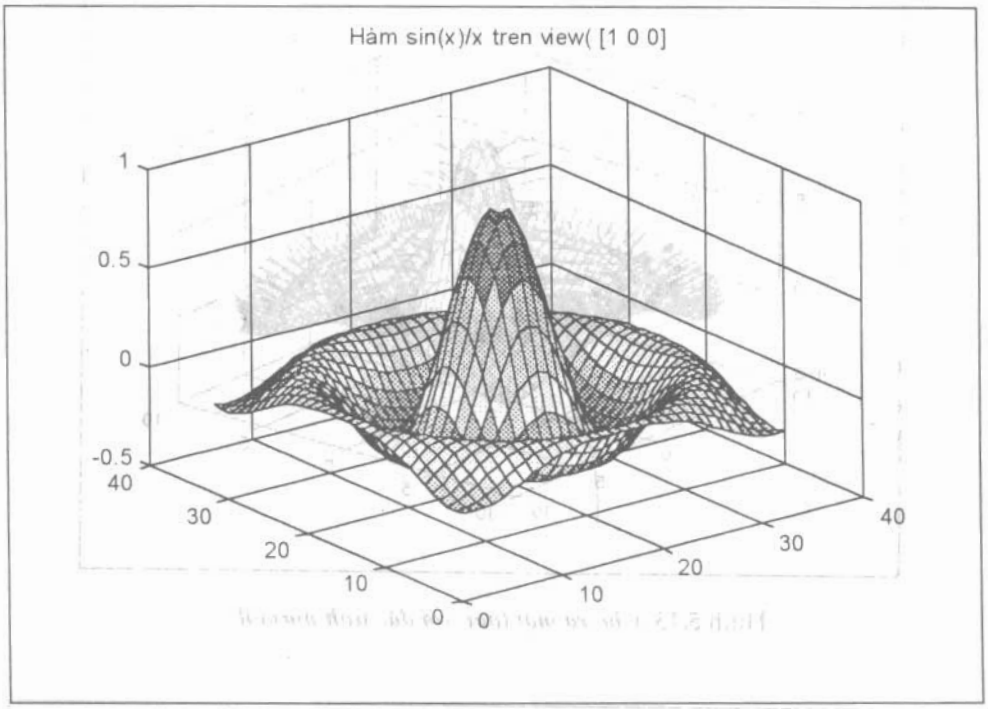
- [Nx, Ny, Nz]

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

Sau đây là một vài ví dụ minh họa.

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

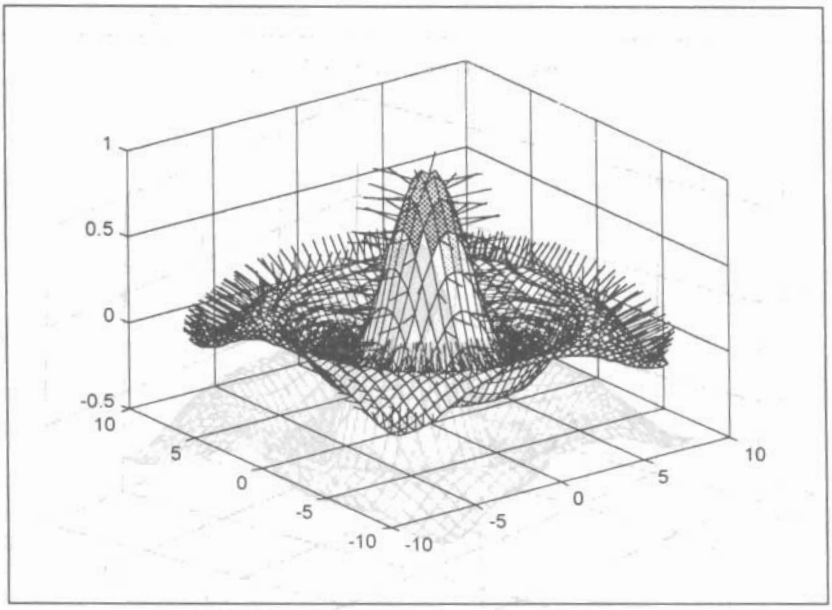
```
>> x = -8 : 0.5 : 8; y = x
>> [X Y] = meshgrid ( x, y );
>> R = sqrt ( X.^2+Y.^2 ) + eps;
>> Z = sin ( R)./ R;
>> surfc ( X,Y,Z );
>> title ( 'Hàm sin r/r' );
```



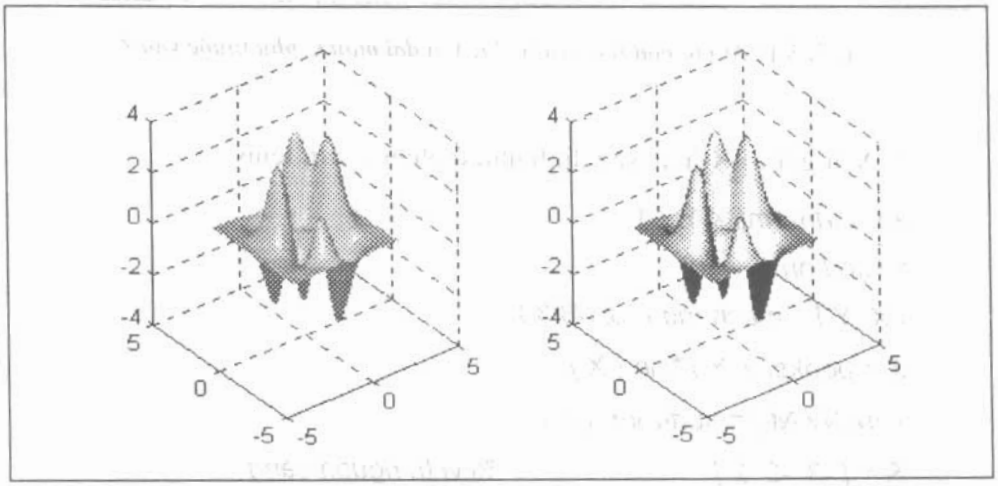
Hình 5.12 *Mô tả mặt lưới sin (r)/ r. Với dải mầu C phụ thuộc vào Z*

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

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



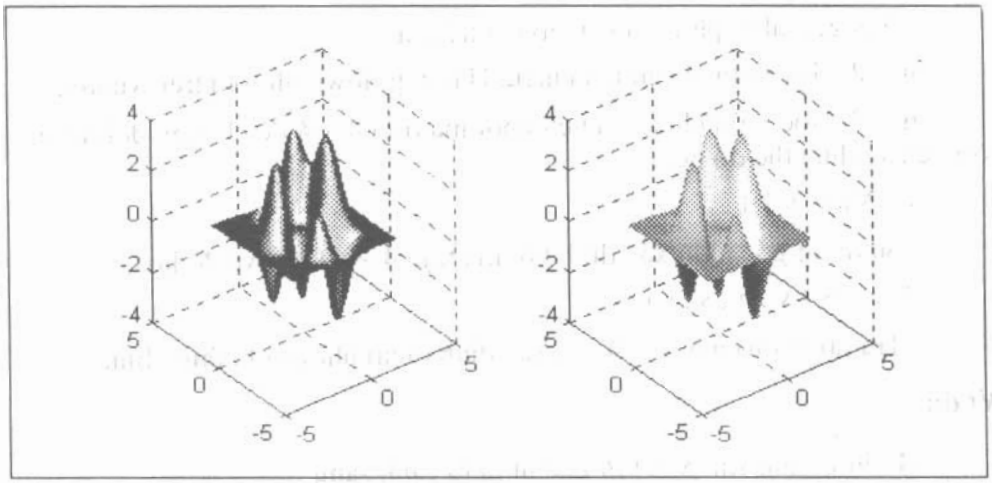
Hình 5.13 *Cho ra mặt lưới với đặc tính normal*



Hình 5.14 *Mặt lưới với các độ phản xạ cho bởi nguồn sáng S*

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

surfl (X,Y, Z, S, k_1); shading interp;



Hình 5.15 *Mặt lưới 3 chiều với các mô hình ánh sáng khác nhau.*

surfl (X,Y, Z, S, k_2); 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

Việc quan sát đồ hoạ sẽ dễ dàng và gần với thực tế hơn nếu được nhìn từ các góc khác nhau. Lệnh view 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 với 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 × 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 × 4T khi vẽ đồ hoạ.

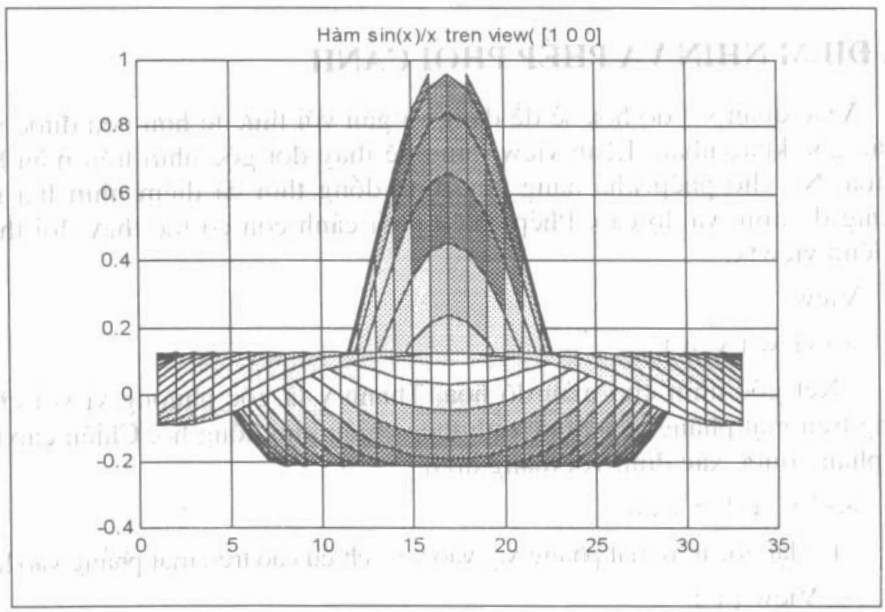
Viewtx (v , h , s , r)

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

Ví dụ

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

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



Hình 5.16 *Mặt sin(x)/r 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 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

Khi nghiên cứu những số Mach trong lĩnh vực hàng không, việc tính toán và tạo ra lưới grid 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 (X_1, Y_1, Mach);

>> view (2);

>> axis ([-0.5 1.5 -1 1]);

>> shading interp;

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

>> mesh(X_1, Y_1, ones (size (X_1))) ;

>> view (2);

>> axis ([-0.5 1.5 -1 1]);

5.7. SLICE TRONG KHÔNG GIAN 3D

Để nghiên cứu những đồ hoạ 3 biến Matlab dùng lệnh slice. Lệnh này để vẽ cắt lát trong không gian 3D với mầu tại mỗi điểm trên bề mặt 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 lưới 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

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]

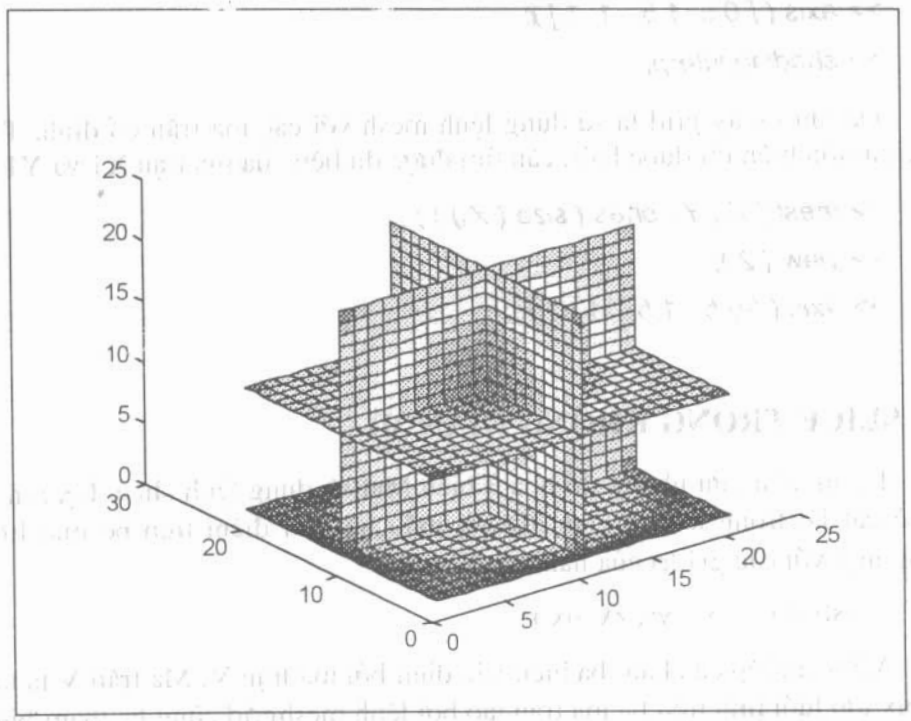
Đầu tiên định nghĩa lưới grid trong không gian 3D thông qua hàm meshgrid và tính các giá trị của hàm F(x,y,z) thông qua các điểm trên lưới grid đó.

```
>> [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 ta phải chọn những mảnh nào song song với trục toạ độ cần vẽ. Vector [1 3 2] cho biết những mảnh 1, 3 và 2, 1 sẽ được vẽ ra..

Điều đó được thực hiện qua lệnh :

```
>> slice (V, [ i 1 ], [1 1 1], [1 1 1], 21)
```



Hình 5.17 *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

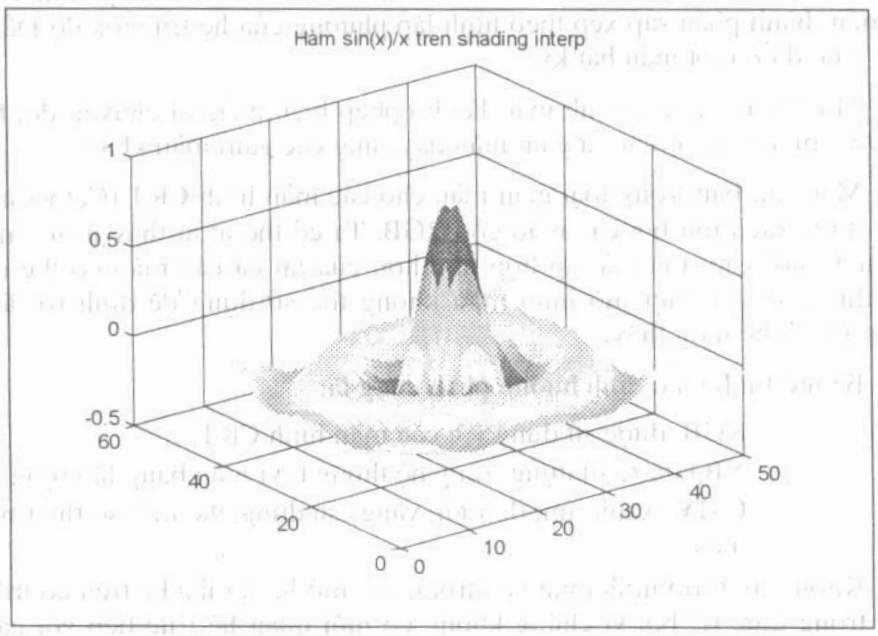
- Faceted dùng để vẽ lưới trên bề mặt và đây là kiểu mặc định của hệ thống

- 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

Hàm sin (R) /R với hiệu ứng bóng shading interp



Hình 5.18 *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 các hệ màu trong màn hình đồ hoạ

Mô hình màu là kỹ thuật biểu diễn mầu sắc của một thể mầu trên một hệ toạ độ mầu ba chiều bao gồm tập các màu 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 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. Ta có thể nhìn thấy trong mảng màu này mỗi gam màu là tập hợp nhỏ hơn của tất cả các màu có thể nhìn thấy được, vì vậy một mô hình màu không thể sử dụng để định rõ tất cả những gì 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 kể trên có thể sử dụng trong thực tế, 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.

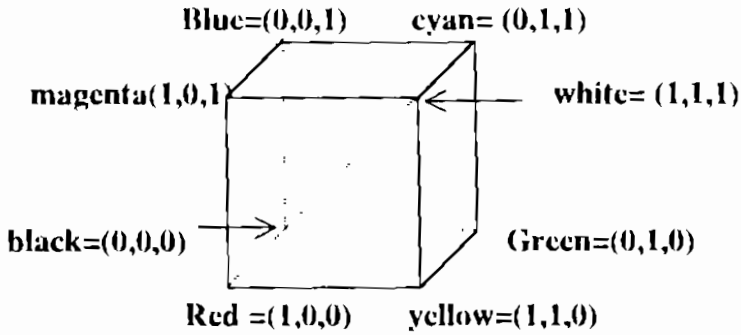
Bởi vậy mỗi mô hình màu khác nhau được phát triển chỉ sử dụng cho một tiêu chí nhất định.

Dưới đây ta hãy cùng tìm hiểu ba mô hình màu HSV, HLS và HVC, trong đó mỗi mô hình màu cho ta một phương tiện phục vụ cho một mục đích tiếp cận khác nhau khi thể hiện mầu sắc. Sự tồn tại của các mô hình mầu nêu trên cũng đồng thời 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) do Uỷ ban Quốc tế về chiếu sáng CIE (Commission Internationale de l' éclairage) quy định. Quy tắc biến đổi này rất quan trọng bởi vì CIE là một tiêu chuẩn được áp dụng 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 da trời (RGB) được sử dụng rộng rãi trên màn hình CRT và các loại màn hình đồ hoạ Raster mầu dựa vào hệ toạ độ Đề các. Những màu trên mô hình RGB được xây dựng trên cơ sở thêm vào những màu gốc, điều đó tạo nên sự đóng góp riêng của từng màu gốc để mang lại kết qủa.

Tập hợp màu hành phần sắp xếp theo khối lập phương đơn vị được chỉ ra trong hình 5.19. Đường chéo chính của khối lập phương với sự cân bằng về số lượng từng màu gốc tương ứng, đi từ mức độ đen là(0, 0,0) cho đến trắng (1, 1, 1) .



Hình 5.19 *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 gian 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 X, 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 $C_2 = M_2 M1.C_1$ 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. Nếu 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 $P(\lambda)$ và sau đó chúng có thể được biến đổi thành toạ độ màu bằng các phương trình (5-1), (5-2) và (5-3).

$$k = \frac{100}{\int P_w(\lambda)\, y\lambda\, d\lambda} \qquad (5\text{-}1)$$

$$x = \frac{X}{(X+Y+Z)}, y = \frac{Y}{(X+Y+Z)}, z = \frac{Z}{(X+Y+Z)} \qquad (5\text{-}2)$$

$$X = \frac{x}{y}Y,\ Y = Y,\ Z = \frac{1-x-y}{y}Y \qquad (5\text{-}3)$$

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 :

$$Cr = Xr + Yr + Zr$$

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 định tương tự cho Cg và Cb phương trình có thể được viết như sau :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{pmatrix} xrCr & xgCg & xbCb \\ yrCr & ygCg & ybCb \\ (1-xr-yr)Cr & (1-xg-yg)Cg & (1-xb-yb)Cb \end{pmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (5\text{-}4)$$

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 yr, yb và yg đã biết để tính các giá trị:

$$Cr = Yr/yr, Cg = Yg/yg,\ Cb = Yb/yb$$

Những giá trị này sau đó được thay thế vào phương trình (5-4) 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.

Ta cũng có thể loại những biến không biết từ phương trình (5-4) nếu biết được 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 (5-4) có thể được viết lại như sau:

$$\begin{bmatrix} Xw \\ Yw \\ Zw \end{bmatrix} = \begin{pmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ (1-x_r - y_r) & (1-x_g - y_g) & (1-x_b - y_b) \end{pmatrix} \begin{bmatrix} C_R \\ C_G \\ C_B \end{bmatrix}$$

Lời giải cho Cr,Cg,Cb là các giá trị cần tìm với những giá trị đã biết được thay vào phương trình (5-4). Mặt khác các giá trị của màu trắng xác định bởi x_w ,y_w,z_w. trong đó Yw sẽ được tìm ra với phương trình trên:

$$Xw=x_wYw/y_w,\ Zw=z_wYw/y_w.$$

5.8.4 Mô hình màu CMY (Cyan, Magenta, Yellow - xanh tím, đỏ tươi, vàng)

Ba mầu CMY là các mầu bù tương ứng cho các màu đỏ, xanh lá cây, xanh da trời và chúng được sử dụng như những bộ lọc để 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 sáng 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 tươi (magenta) hấp 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, chúng 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:

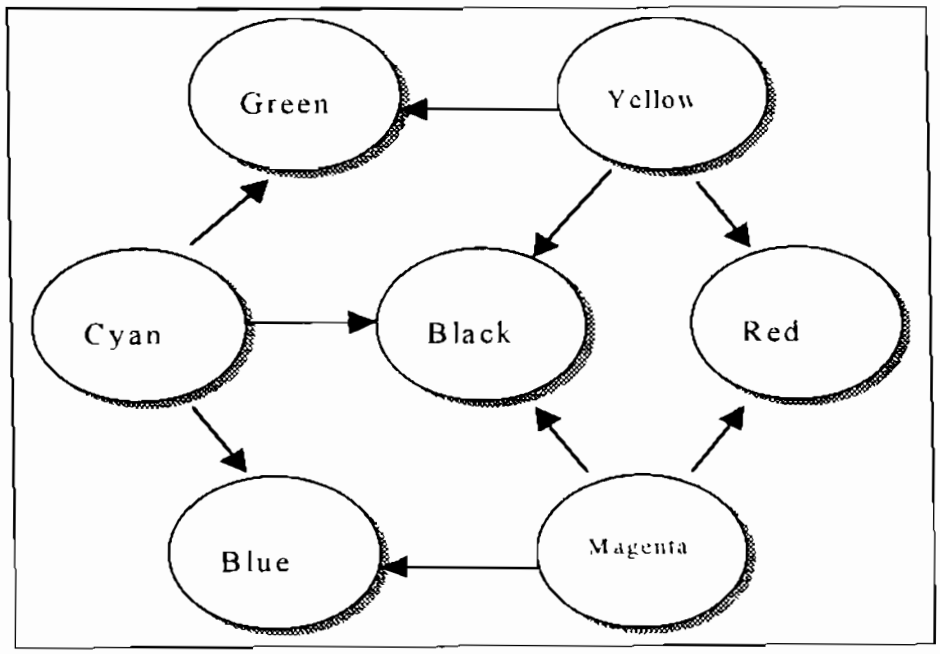
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

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 tổ 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 được ứng dụng rất hiệu quả trong công nghệ in phun và in Xerox.



Hình 5.20 *Các màu bù (cyan, magenta, yellow) và sự pha trộn giữa chúng*

Mô hình CMYK

Một mô hình màu khác, CMYK là mô hình sử dụng thêm mầu đen (viết tắt là K) như màu thứ tư, được sử dụng trong quá trình in bốn màu trên 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 trăng 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 hai thành phần còn lại là I và Q. Mô hình YIQ sử dụng hệ toạ độ Đêcác ba 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 mức quan trọng của màu xanh lá cây và màu đỏ cũng như mức 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 thiết 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 cụ thể trong mô hình màu YIQ, tạo tiền đề cho việc phát triển rộng rãi 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 việc 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 mắt con người 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 mắt con người, điều này có thể được chỉ rõ khả năng tương xứng với màu một chiều hơn là màu hai chiều. Giả thiết này cho I, Q hoặc cả hai là chúng có thể có một dải tần thấp hơn Y.

Hệ thống mã NTSC của mô hình màu YIQ 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.

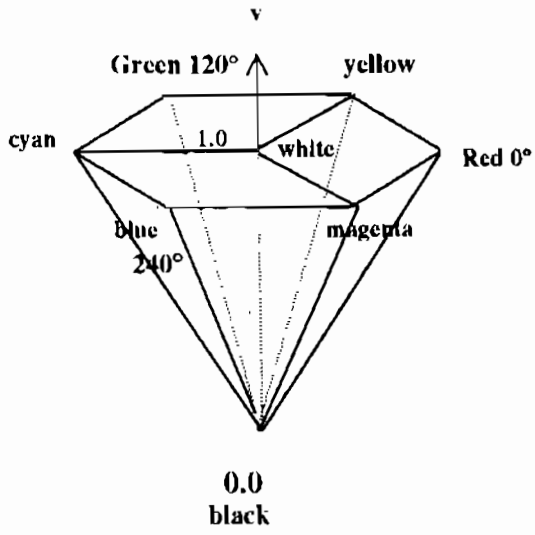
5.8.6 Mô hình màu HSV (Hue, Saturation, Value)

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), định hướng cho người sử dụng trên cơ sở 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 chóp sáu cạnh như trong hình 5.21. Đáy hình chóp sáu cạnh khi V=1 chứa đựng mối quan hệ giữa các màu sáng. Những màu trên mặt phẳng đáy chóp 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°. Các màu bổ xung trong hình chóp HSV ở 180° đối diện với màu cơ bản. Giá trị của S là một dãy số truyền từ giá trị 0 đến 1 trên đường trung tâm (trục V). Sự bão hoà được đo tương đối cho gam màu tương ứng với mô hình màu này, và là một tập hợp nhỏ trong toàn bộ biểu đồ màu CIE do đó sự bão hoà 100% trong mô hình ít hơn 100% sự kích thích tinh xác.

Trong hình nón sáu cạnh, đường cao V với điểm ở đỉnh là màu đen và có giá trị toạ độ mầu V = 0, tại 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.

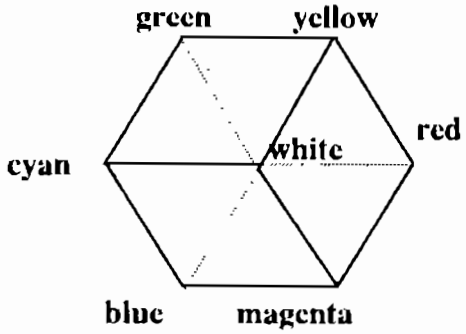


Hình 5.21 ***Mô hình mầu IIS***

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

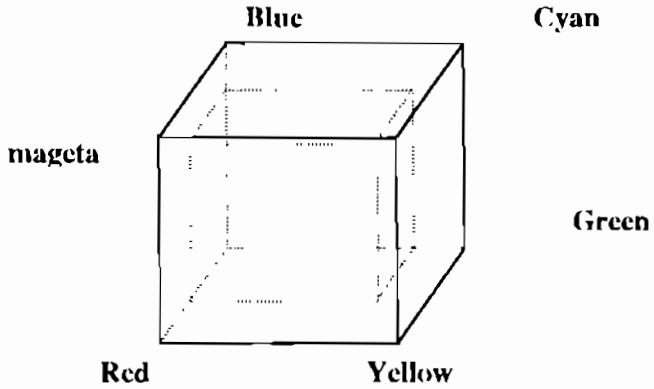
màu được tạo ra bởi việc giữ S =1 và giảm V. Sắc thái được tạo ra bởi việc giữ cả hai S và V cố định. Dĩ nhiên sự thay đổi H tương ứng sẽ lựa chọn được một chất màu cần thiết. Bởi vậy H, S và V phù hợp với khái niệm màu của hội hoạ và rất chính xác.



Hình 5.22 ***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 5.22.

Khối lập phương RGB có các khối lập phương nhỏ bên trong, như được minh hoạ trong hình 5.23. Mỗi hình lập phương nhỏ khi được nhìn thấy dọc theo đường chéo chính của nó giống như hình sáu cạnh ở hình 5-22, trừ những hình nhỏ hơn. Mỗi mặt V bất biến trong khoảng không gian HSV tương ứng với 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***

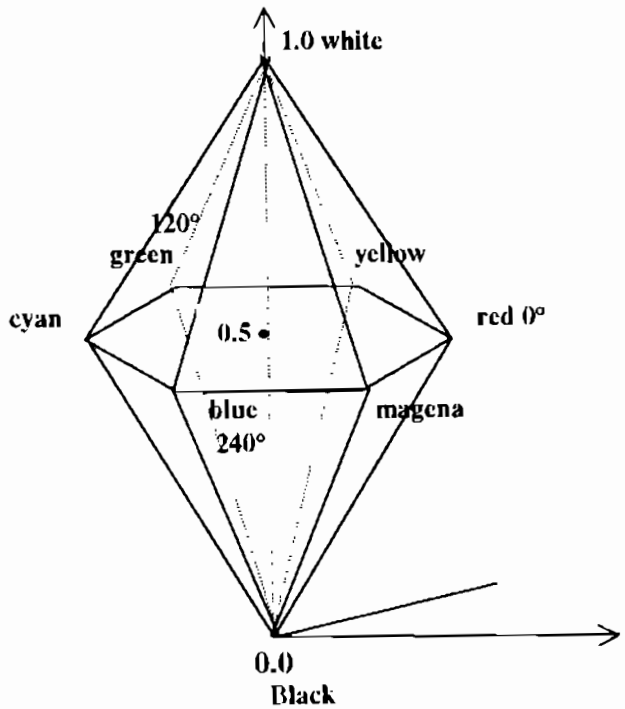
Đường chéo chính của không gian RGB trở thành trục V của không gian HSV. Như vậy ta có thể nhìn thấy bằng trực giác sự phù hợp giữa RGB

và HSV và thuật toán Va,Vb xác định đúng sự phù hợp ấy trong khi biến đổi từ một mô hình sang một mô hình khác.

5.8.7. Mô hình màu HLS (Hue, Light, Saturation - màu sắc, độ sáng, sự bão hoà)

Mô hình mầu HLS được xác định bởi tập hợp hai hình chóp sáu cạnh chung đáy như hình 5-24. Màu sắc là góc quanh trục Black-White, kể từ màu đỏ tại góc 0° Mặc dù trong thảo luận về HSL có ý kiến cho màu xanh da trời là điểm 0° nhưng ta vẫn 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 sắp xếp trong mô hình hình nón sáu cạnh đơn HSV.



Hình 5.24 Mô hình màu hình nón sáu cạnh đôi HLS.

Tóm lại có thể coi mô hình HLS như một sự biến dạng của mô hình HSV trong đó màu trắng được kéo hướng lên đỉnh chóp sáu cạnh từ mặt

V = 0,5. Tương tự mô hình hình chóp sáu cạnh đơn, phần bổ xung của màu sắc được đặt ở vị trí 180° xung quanh hình nón sáu cạnh đôi, sự bão hoà được đo xung quanh trục đứng, từ 0 trên trục tới 1 trên bề mặt.

Độ sáng (lighness) = 0 cho màu đen (tại điểm mút thấp nhất của hình nón sáu cạnh đôi) và bằng 1 cho màu trắng (tại đầu mút cao nhất). 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 thực hiện sự biến đổi giữa HLS và RGB thực chất là phép biến đổi để dời tới H không xác định khi S=0 và đến khi H=0 cho màu đỏ hơn là cho màu xanh.

Mô hình HLS 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 thì 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.

Tuy nhiên tương tự như mô hình HSV các màu của mảng L =0.5 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.

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 có thể đo và nhận biết được khoảng cách giữa các màu là xấp xỉ như nhau. Điều này thể hiện 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 phân tích.

Tuy nhiên sự chuyển đổi đó từ CIE XYZ sang CIE LUV là không phức tạp. Như vậy các không gian màu cùng loại sẽ được nhận biết và sử dụng rộng rãi trong tương lai.

5.8.8 Các 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 da trời (R,G,B). Màu sắc của 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 nghĩa ra colormap

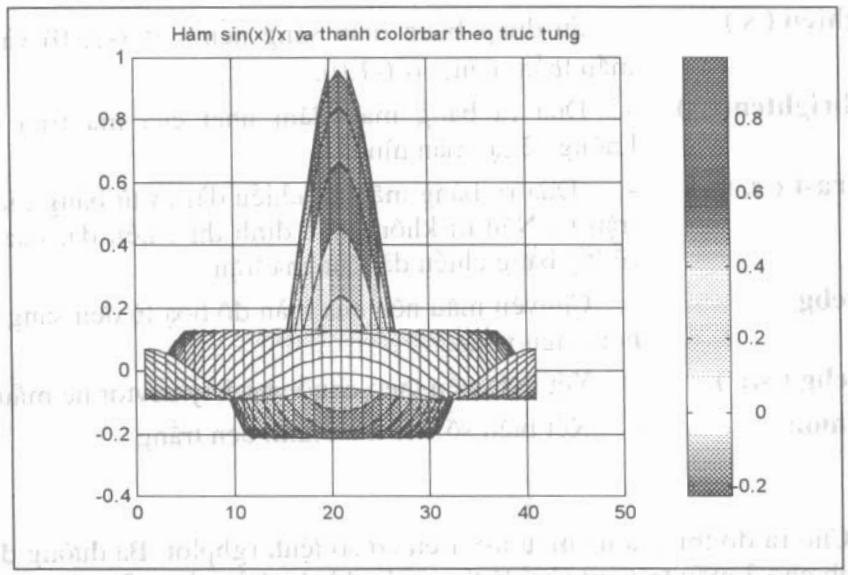
colormap - Trả giá trị bảng mô hình hiện tại đang dùng vào ma trận m x 3.

colorbar - Vẽ ra dải mầu thẳng đứng trong màn hình đồ hoạ hiện thời.

colorbar ('horiz') - Vẽ ra dải mầu nằm ngang 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 màu: 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 5.25 *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 × 3) C từ hệ mầu rgb sang hệ mầu hsv

hsv2rgb (C) - Chuyển ma trận C (m × 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 da trời.

caxis (v) - Đưa ra khoảng xác định giới hạn của bảng mầu với $v = [v_{min}\ v_{max}]$: v_{min}, v_{max} là các 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.

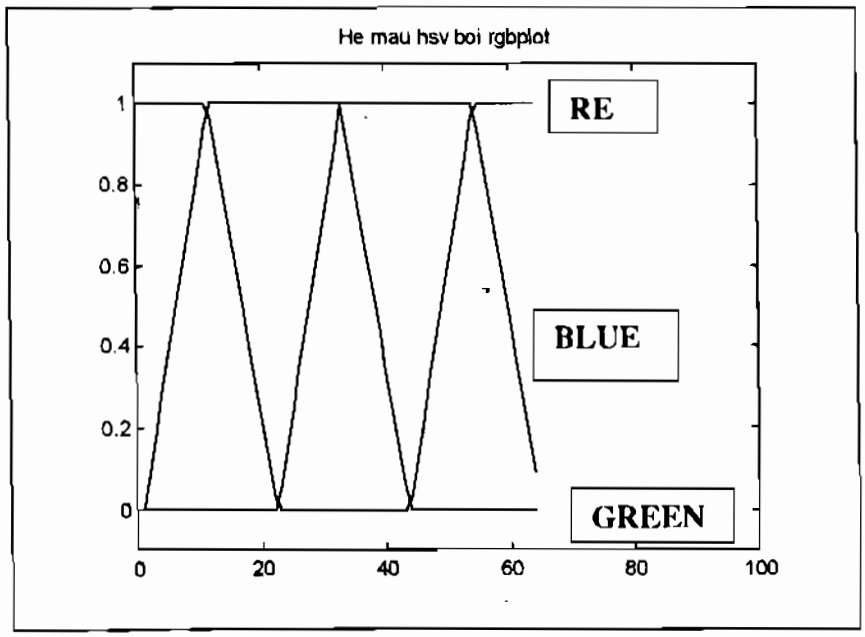
spinmap(inf) - Quay bảng màu không xác định thời gian

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 nhạt 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 bằng chiều dài của ma trận.
whitebg	- Chuyển mầu nền của màn đồ hoạ từ đen sang 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

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 5.26 ***Hệ màu hsv trên cơ sở lệnh rgbplot.***

BÀI TẬP ỨNG DỤNG PHẦN THỨ NHẤT

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 nghịch đảo 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 xoắn
% r = a* phi
disp('Ve duong xoan : 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')
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 xoắn ốc 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 xoan oc')
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 paraboloit, 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 tích 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 và 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
```