ĐẠI HỌC THỦY LỢI
BỘ MÔN QUẢN LÝ TỔNG HỢP BIỂN VÀ ĐỚI BỜ

------ ☙✸❧ ------

MATLAB

TIN HỌC ỨNG DỤNG – HỌC PHẦN II

(Tài liệu tham khảo & Bài tập cho sinh viên Kỹ thuật Biển – K45)

Nguyễn Bá Tuyên
Nguyễn Quang Chiến

Hà Nội, tháng 08 năm 2007
Chỉnh lý, bổ sung 2024

MỤC LỤC

1. CHƯƠNG I: MATLAB CĂN BẢN

1.1. Matlab – ngôn ngữ của tính toán kỹ thuật

- MATLAB là một ngôn ngữ bậc cao và môi trường tương tác cho phép bạn tiến hành các nhiệm vụ tính toán có cường độ lớn nhanh hơn với các ngôn ngữ lập trình như C, C++ và Fortran.

- MATLAB viết tắt cho "Matrix Laboratory" - Phòng thí nghiệm ma trận. Ban đầu Matlab được thiết kế bởi Cleve Moler vào những năm 1970 để sử dụng như một công cụ dạy học. Từ đó đến nay nó đã được phát triển thành một bộ phần mềm thương mại rất thành công.

- Hiện nay MATLAB R14 là một bộ phần mềm cho công việc tính toán trong các ngành kỹ thuật, trong khoa học và trong lĩnh vực toán học ứng dụng.[1] Matlab cho ta một ngôn ngữ lập trình mạnh, giao diện đồ họa xuất sắc, và một phạm vi rất rộng các kiến thức chuyên môn. Matlab là một thương hiệu đã được thương mại hóa của tập đoàn ***MathWorks***, Massachusetts, USA (hiện là nhà cung cấp hàng đầu thế giới cho các phần mềm tính toán kỹ thuật và thiết kế dựa trên mô hình).

1.2. Khả năng và những ứng dụng của Matlab

- Một trong những tính năng tuyệt vời nhất của Matlab nhìn từ góc độ những nhà khoa học tính toán là thư viện dựng sẵn to lớn rất phong phú các chu trình tính toán và các công cụ hiển thị đồ họa.

- Matlab cho phép người dùng tiến hành rất nhiều các nhiệm vụ thông thường liên quan tới việc giải quyết các vấn đề một cách số học. Nó cho phép chúng ta dành nhiều thời gian hơn cho việc suy nghĩ, khuyến khích chúng ta thí nghiệm.

- Matlab ứng dụng những thuật toán hết sức được trân trọng vì vậy chúng ta có thể tin tưởng vào kết quả thu được.

- Các tính toán rất mạnh có thể được thực hiện chỉ với một hoặc hai câu lệnh.

- Bạn có thể xây dựng riêng cho mình những hàm toán học cho những ứng dụng đặc biệt.

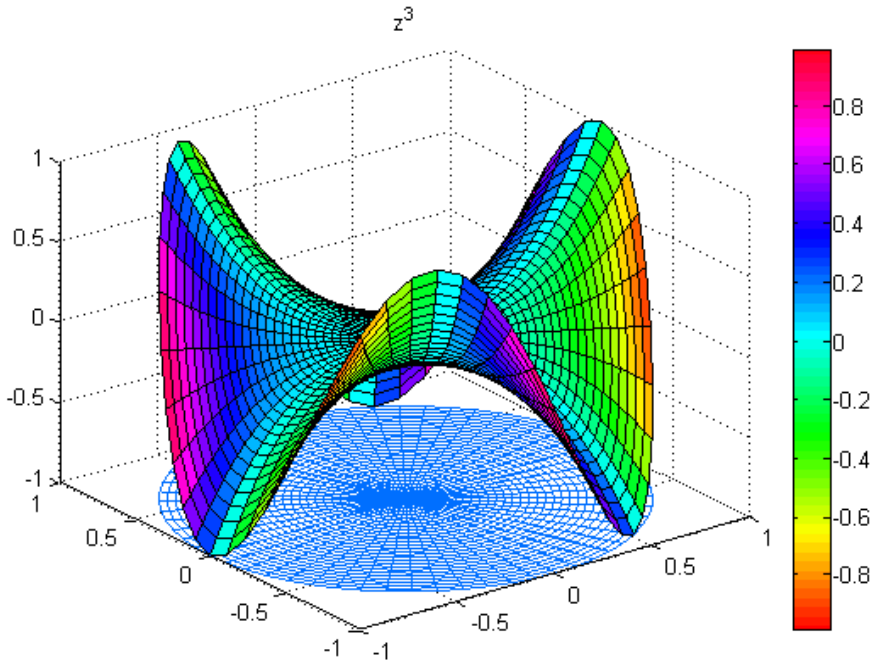
- Matlab cung cấp giao diện đồ họa tuyệt hảo, các hình từ Matlab có thể đem chèn vào LaTeX và các tài liệu Word.

- Tài liệu hướng dẫn này chỉ đem đến một cái nhìn thoáng qua về sức mạnh và sự linh hoạt của hệ thống Matlab. Để có được những hiểu biết sâu sắc và chi tiết hơn, xin tham khảo các giáo trình Matlab chuyên dụng khác hiện có trên thị trường.

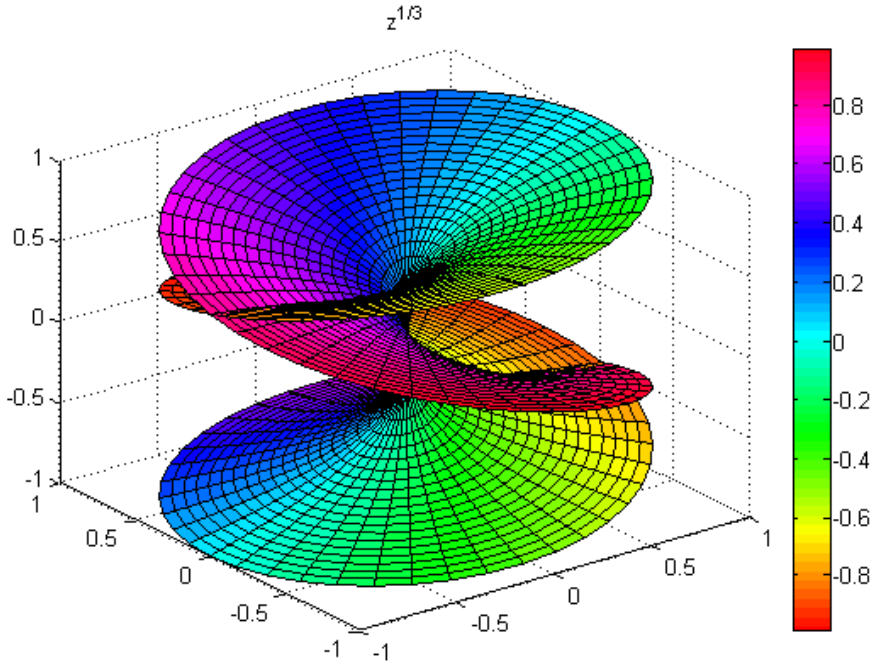
[1] Tên chính thức MATLAB viết chữ in, song để cho tiện chúng tôi sẽ viết thường trong suốt cuốn sách này.

VD: Hãy cùng tham khảo các demo của Matlab để xem ta có thể làm được những gì chỉ với một vài dòng lệnh đơn giản:

```
>> colormap(hsv(64))
>> z = cplxgrid(30);      % miền không gian số phức đơn vị z
>> cplxmap(z,z^3)         % vẽ đồ thị hàm x = z^3 – hinh1
>> cplxroot(3)            % vẽ đồ thị hàm y = z^(1/3) – hinh2
```



Hình 1: Đồ thị hàm $x = z^3$ trong không gian số phức



Hình 2: Đồ thị hàm $y = \sqrt[3]{z}$ trong không gian số phức

1.3. Đặc điểm của Matlab

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

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

Để minh họa điều này, ta xét một ví dụ rất đơn giản: so sánh phép cộng hai ***vécto*** trong ngôn ngữ lập trình Pascal và Matlab.

- Pascal biểu diễn một véctơ dưới dạng mảng (array), chẳng hạn có 3 vec-tơ A, B, C và A + B = C:

```
var
  A: array[1..5] of integer = (3, 7, 4, 2, 0);
  B: array[1..5] of integer = (-2, 4, 8, 5, 1);
  C: array[1..5] of integer;
  i: integer;
begin
  for i := 1 to 5 do
    begin
      C[i] := A[i] + B[i]
    end
end.
```

- Cách làm trong MatLab đơn giản hơn nhiều:

```
A = [3 7 4 2 0];
B = [-2 4 8 5 1];
C = A + B;
```

- Có được sự đơn giản nói trên là nhờ Matlab đã xây dựng sẵn khái niệm ***ma trận***. Dấu cộng trong dòng lệnh Matlab biểu thị phép cộng ma trận. Pascal không được như vậy; mảng chỉ là sự biểu diễn có thứ tự của các biến. Không có phép cộng ma trận, chỉ có phép cộng hai số – vì vậy chương trình Pascal dài hơn rất nhiều.

- Một đặc điểm nữa là tất cả các biến trong chương trình Pascal trên đều phải được khai báo. Trong Matlab các biến sẽ tự động hình thành trong mỗi câu lệnh gán.

Trong những năm gần đây, bên cạnh các ngôn ngữ lập trình truyền thống (C / C++ / Fortran), các *ngôn ngữ văn lệnh* (*scripting languages*) được sử dụng phổ biến hơn trong lĩnh vực nghiên cứu tính toán. Matlab là một trong các ngôn ngữ như vậy. Là một ngôn ngữ bậc cao, mỗi dòng lệnh Matlab thường có tác dụng tương đương với khoảng 10 dòng lệnh của C / C++. Người lập trình sẽ tốn ít thời gian gõ câu lệnh và tập trung hơn vào nội dung chương trình.

Tuy vậy các *ngôn ngữ lập trình biên dịch* như C / Fortran cho phép chương trình tính toán rất nhanh và tốc độ cũng là một yêu cầu rất quan trọng trong các chương trình tính lớn. Do đó một cách kết hợp thông minh là phần lõi tính toán có thể được viết bằng ngôn ngữ biên dịch, và các thao tác nhập xuất, xử lí, hiển thị số liệu được viết bởi ngôn ngữ văn lệnh như Matlab.

1.4. Cài đặt và khởi động Matlab

Chúng tôi sử dụng Matlab phiên bản 7.0 hay R14 (phát hành 2004) để giới thiệu những tính năng cơ bản nhất. Hiện nay Matlab đã phát triển đến phiên bản R2024b, với nhiều đặc điểm như tính toán hiệu năng cao trên nhiều nhân CPU, hoàn thiện hệ thống kiểm thử phần mềm (*testing framework*). Ngoài Windows (nền tảng được giới thiệu trong sách này), Matlab còn chạy trên UNIX (thực ra đây mới chính là nền tảng nguyên gốc để phát triển Matlab), Linux và Mac OS X. Vẻ ngoài giao diện đồ hoạ của Matlab cũng khác những hình in trên sách này song nguyên lý hoạt động vẫn giữ nguyên và các dòng mã lệnh hoàn toàn có hiệu lực.

1.4.1. Cài đặt Matlab 7.0

- Yêu cầu về cấu hình máy tính:

+ Bộ vi xử lý Pentium hoặc Pentium Pro

+ Windows 95 hoặc NT (WinXP home, XPprofessional đều được)

+ Bộ điều phối đồ họa 8 bit và card màn hình tối thiểu 256 màu

+ Dung lượng ổ cứng 25 MB cho tới hơn 1 GB (tùy thuộc vào cách cấu hình đĩa cứng, phân vùng đĩa, số hợp phần của Matlab được cài đặt), và tới 2,1 GB nếu cài đặt Matlab cùng với Simulink.

+ Bộ nhớ động (RAM) tối thiểu 16 MB (nên có bộ nhớ tối thiểu 128 MB)

+ Các khuyến nghị khác: Bộ nhớ bổ sung, card đồ họa bổ sung, card âm thanh, máy in, MS-Word 7.0 hoặc hơn, trình biên dịch C, Borlean, Micosoft (xây dựng file MEX), trình duyệt internet (để chạy Matlab Helpdesk online).

- Quá trình cài đặt Matlab 7.0 cho Windows XP (bộ gồm 2 đĩa CD):

+ Đưa đĩa CD vào ổ đọc. Nếu chương trình SETUP không tự động chạy thì nhấn đúp vào biểu tượng *setup.exe* để bắt đầu quá trình cài đặt.

+ Accept (chấp nhận) những thỏa thuận về bản quyền. sau đó click Next.

+ Nếu bạn cài theo kiểu mặc định (hay còn gọi là *Typical setup* - kiểu phổ biến), Matlab trên máy tính của bạn sẽ có các hợp phần cơ bản nhất để làm việc theo

các hướng dẫn trong tài liệu này. Theo các hướng dẫn trên màn hình. Cho đĩa CD thứ 2 vào khi được yêu cầu.

+ Nếu bạn cài đặt theo kiểu tùy chọn cá nhân (*Manual setup*) thì nhấn vào các hộp thành phần dấu 'v' nếu bạn muốn có tùy chọn đó. Nhấn tiếp nếu bạn không có ý định (có thể thêm vào sau này nếu muốn).

+ Trên màn hình hiển thị '*C:\MATLAB7*' là thư mục mặc định của quá trình cài đặt. Nếu bạn muốn cài đặt vào địa chỉ khác, hoặc đổi tên thư mục, thì bạn lựa chọn '*Browse*'.

+ Chi tiết hướng dẫn cài đặt xin xem file 'install_guide.pdf' trong đĩa CD1 (bản tiếng Anh).

- Cài đặt trên mạng:

Tuỳ theo hợp đồng kí kết với công ty MathWorks Inc. mà bạn có thể tải về và cài đặt sản phẩm (Matlab, Simulink) với phiên bản tương ứng sau khi đăng nhập vào tài khoản web.

1.4.2. Khởi động Matlab (Hệ điều hành Windows)

- Từ HĐH Windows, khởi động Matlab đơn giản bằng cách nháy đúp vào biểu tượng MATLAB trên màn hình, hoặc bằng cách chọn MATLAB từ Menu Start.

- Quá trình khởi động đưa người dùng đến *Cửa sổ lệnh*, nơi các dòng lệnh được biểu thị bằng hai dấu **>>**.

>>_

Đây là dấu hiệu cho thấy Matlab đang chờ bạn đánh một (câu) lệnh. Khi hoạt động trong chế độ máy tính cầm tay, tất cả các lệnh của Matlab được nhập vào dòng lệnh từ bàn phím.

- Matlab có thể được sử dụng theo nhiều chế độ và nhiều cách khác nhau;

+ Như một máy tính cầm tay cao cấp trong chế độ máy tính cầm tay

+ Như một ngôn ngữ lập trình bậc cao

+ Như một chu trình con gọi từ chương trình C

Trong tài liệu này chúng ta sẽ đi nghiên cứu chi tiết *2 chế độ đầu tiên*.

1.5. Quản lý không gian làm việc của Matlab

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

+ Cửa sổ trợ giúp (*Help window*)

+ Nút Start

+ Cửa sổ nhập lệnh (*Command window*)

+ Cửa sổ không gian làm việc (*Workspace window*)

+ Cửa sổ quá trình lệnh (*Command History window* - lịch sử)

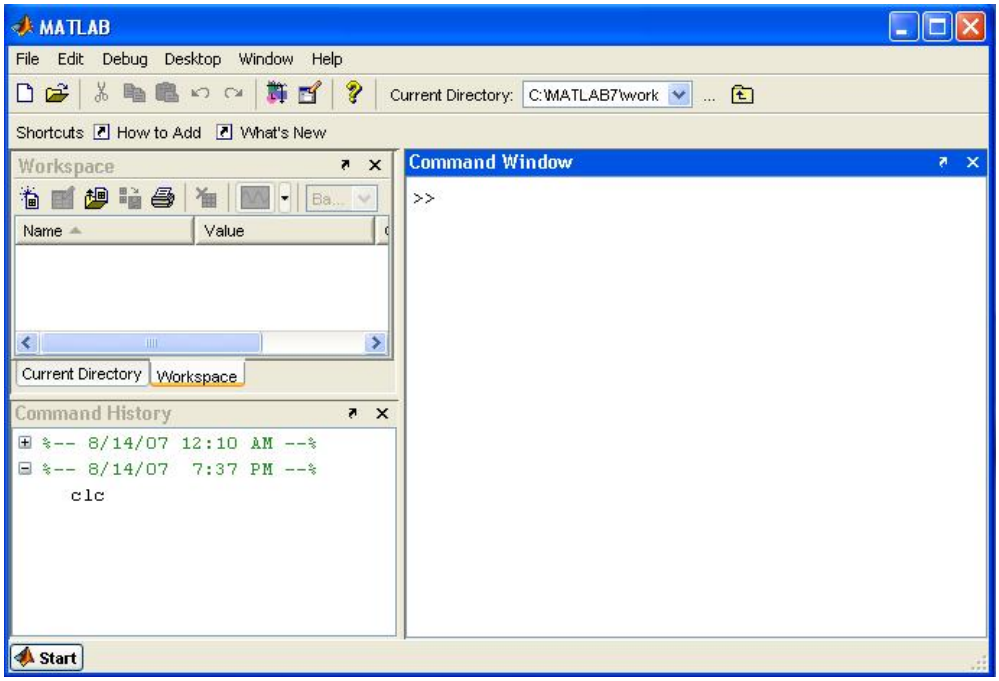
+ Cửa sổ biên tập mảng, vectơ, ma trận (*Array editor window*)

+ Cửa sổ địa chỉ thư mục hiện thời (*Current directory window)*

- Nút 'x' ở góc trên bên phải mỗi cửa sổ dùng để đóng chúng. Hiển thị lại cửa sổ bằng cách tích '√'vào tên cửa sổ tương ứng trong menu Desktop.

- Nút mũi tên cong bên cạnh nút 'x' dùng để tách các cửa sổ làm việc trong cửa sổ chính MATLAB thành cửa sổ con độc lập. Ấn nút này một lần nữa sẽ nhập một cửa sổ độc lập về cửa sổ chính của MATLAB.

- Cửa sổ *Help*, *History* sẽ được giới thiệu cụ thể trong *mục 1.7* và *mục 1.8*. Sau đây các cửa sổ làm việc còn lại sẽ được giới thiệu vắn tắt.



Hình 1.1: Giao diện của Matlab 7.0

**** Nút Start:*** ở góc dưới bên trái của màn hình Matlab, cho phép ta chạy các ứng dụng mẫu (*demos*), các công cụ và cửa sổ chưa hiển thị khi khởi động Matlab. Bằng cách đánh lệnh '*demo*' bạn có thể tiếp cận với một tập hợp sâu rộng những file trình diễn giá trị rất cao, vì đó là biểu hiện cho những khả năng của Matlab.

Ví dụ: Thử chạy Start -> Matlab -> Demos và chạy một ứng dụng mẫu trong cửa sổ Demo(s).

Ghi chú: Lệnh này sẽ xóa tất cả giá trị của các biến hiện có.

**** Cửa sổ lệnh:*** đã được đề cập ở mục 1.4.2. (Khởi động Matlab).

- Các diễn giải và câu (mệnh đề) của Matlab được đánh giá khi bạn gõ vào '*Cửa sổ lệnh*', và các kết quả tính toán cũng được thể hiện tại đây. Không giống như Fortran và các ngôn ngữ tính toán cần biên dịch khác, Matlab là một môi trường tương tác – bạn đưa ra một câu lệnh, và Matlab cố gắng thực thi nó ngay lập tức trước khi đòi hỏi 1 lệnh tiếp theo.

- Các diễn giải và câu cũng được sử dụng trong các M-file (sẽ được trình bày chi tiết ở chương V). Chúng thường có cấu trúc:

>> *biến = diễn giải* ↵

hoặc đơn giản là >> *diễn giải* ↵

- Các diễn giải thường được soạn bằng các *toán tử*, các *hàm*, và tên các *biến*, và được hiển thị trên màn hình sau khi ấn Enter. Các câu lệnh có dạng '*tên biến = diễn giải'* thì *diễn giải* đó sẽ được gán cho biến để sử dụng sau này. Khi '*tên biến*' và dấu '=' được bỏ đi thì kết quả của diễn giải sẽ được tự động gán cho biến có tên '*ans*' (hay answer – câu trả lời) và hiển thị trên màn hình.

- Một câu (lệnh) thông thường sẽ kết thúc ở cuối dòng. Tuy nhiên có thể tiếp tục một câu bằng ba dấu chấm '**...**' ở cuối dòng.

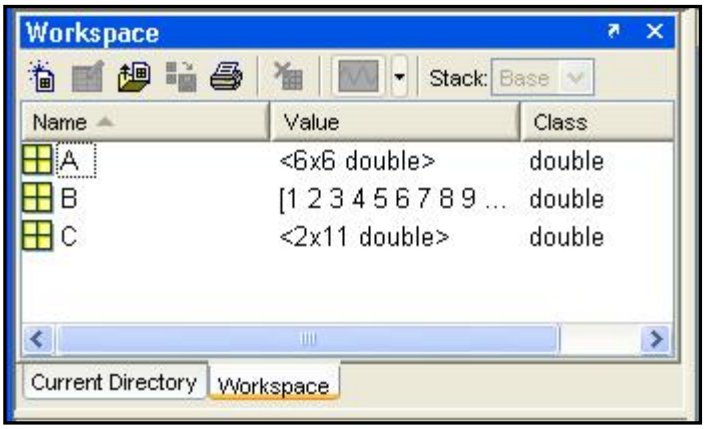
- Có thể đặt một vài câu lệnh trên cùng một hàng, ngăn cách bởi dấu phẩy '**,**' hoặc chấm phẩy '**;**'

- Nếu một câu lệnh kết thúc bằng dấu chấm phẩy ở cuối câu thì kết quả của lệnh đó sẽ không được hiển thị, tuy nhiên yêu cầu tính vẫn được thực hiện (phép tính hay phép gán vẫn được thực hiện, kết quả có trong *workspace*). Điều này là thiết yếu trong việc ẩn đi các kết quả trung gian không mong muốn (VD như khi thực hiện một loạt phép tính, hay tính toán với các ma trận lớn).

- Bạn có thể xóa trắng toàn bộ cửa sổ lệnh bằng lệnh

```
>> clc          ↵      % (clear command window)
```

hoặc vào menu Edit -> Clear Command Window. Khi thực hiện lệnh này, toàn bộ giá trị của các biến hiện có không thay đổi hay mất đi.



**** Cửa sổ không gian làm việc (workspace):***

Các biến và dữ liệu mà bạn nhập vào hoặc tính toán ra sẽ được Matlab lưu trong một phần gọi là '*không gian làm việc*'. Tất cả các biến, ngoại trừ những biến cục bộ thuộc về một M-file, sẽ được thể hiện trong không gian làm việc

- Lệnh '*who*' hoặc '*whos*' liệt kê các biến hiện có trong không gian làm việc.

VD: đánh lệnh '*whos*' vào cửa sổ lệnh, bạn sẽ thấy một danh sách các biến hiện có cùng kiểu loại và kích cỡ của chúng.

- Để biết giá trị hiện tại của một biến, bạn đánh vào tên biến tại dấu nhắc của cửa sổ lệnh và Enter.

- Để xóa một hàm hoặc biến khỏi không gian làm việc, ta sử dụng lệnh '*clear*':

>> clear tên_biến ↵

Bản thân lệnh '*clear*' sẽ xóa tất cả các biến hiện có (tương đương với '*clear all')*

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

Ví dụ: Có ma trận M, hãy thử click và thay đổi nó, thay đổi các phần tử, hay kích thước ma trận. Quay trở lại Cửa sổ lệnh và gõ 'M' rồi Enter, xem ma trận M của chúng ta sau khi thay đổi.

+ Bạn cũng có thể biên tập lại ma trận M bằng cách đánh lệnh

>> openvar ('C') ↵

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

1.6. Ghi & phục hồi dữ liệu

1.6.1. Lưu và phục hồi dữ liệu

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

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

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

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

>> save ↵ - lưu tất cả các biến trong Matlab theo kiểu nhị phân trong file *matlab.mat*

>> save dulieu ↵ - lưu tất cả các biến trong Matlab theo kiểu nhị phân trong file *dulieu.mat*

>> save dulieu A B C D -ascii ↵

- lưu các biến A, B, C, D theo dạng mã ASCII trong file *dulieu.mat*

1.6.2. Lưu một bộ dữ liệu (record)

1.6.3. Lưu một phiên (session)

Khi làm bài tập, việc lưu tất cả các thông số đầu vào và đầu ra của phiên làm việc với Matlab hiện tại của bạn cho việc in ấn sau này là rất hữu ích. Lệnh '*diary*' sử dụng cho mục đích này, sẽ lưu tất cả những thông số đầu vào và đầu ra ở giữa hai lệnh '*diary*' và '*diary off*'. Từ dấu nhắc ở dòng lệnh, bạn đánh:

>> diary('diary_file_name') ↵

>> ..... (các câu lệnh của bạn ở đây) ↵

>> diary off ↵

1.7. Sử dụng Help

- Trợ giúp và thông tin về các lệnh của Matlab có thể được tìm thấy theo nhiều cách:

+ Từ dòng lệnh bằng cách đánh lệnh '*help chủ đề*' (xem dưới đây)

+ Từ cửa sổ Help riêng biệt xuất phát ở *Menu Help*

+ Từ *helpdesk* của Matlab lưu trữ trên đĩa hoặc CD-rom, hoặc

+ Từ mạng Internet

- Từ dòng lệnh, đơn giản nhất hãy đánh lệnh '*help*' và Enter!

Kết quả: Matlab cho ta một bản tóm tắt về hệ thống trợ giúp.Một vài dòng đầu tiên của kết quả sẽ như sau:

HELP topics: (*Các chủ đề trợ giúp*)

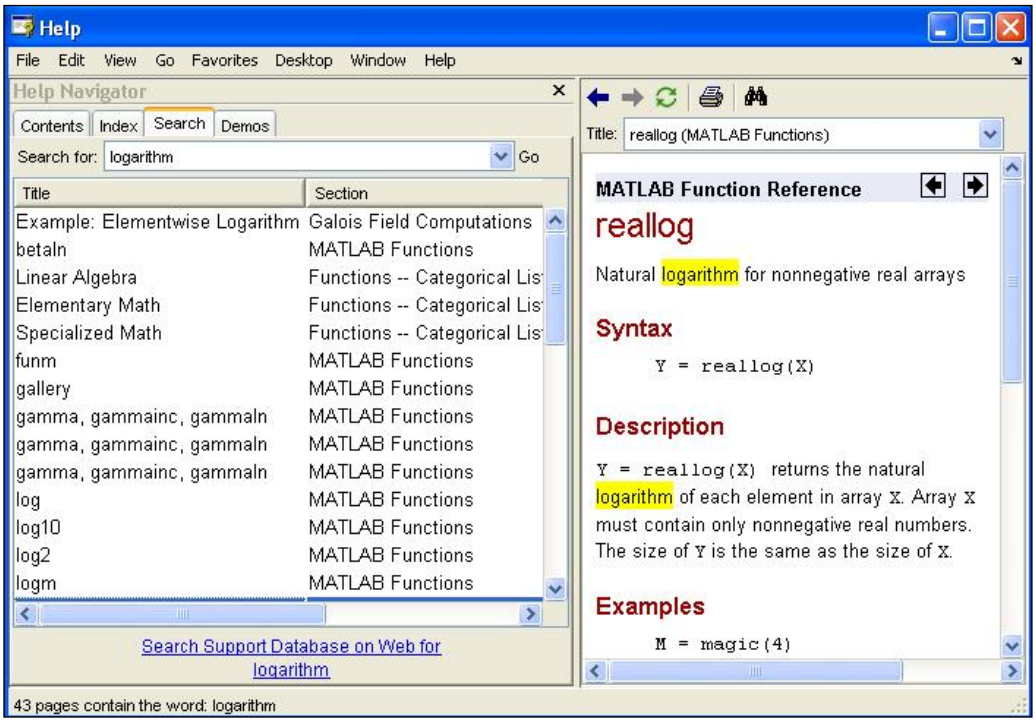
matlab/general	- Các lệnh với mục đích tổng quát.
matlab/ops	- (*operators*) Các toán tử và các ký tự đặc biệt...
matlab/lang	- (*language*) Ngôn ngữ lập trình...
matlab/elmat	- (*elementary*) Ma trận căn bản...
matlab/elfun	- (*elementary functions*) Các hàm toán căn bản.
matlab/specfun	- (*specialized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

- Thông thường bạn sẽ thấy cửa sổ text không đủ lớn để chứa tất cả thông tin từ một lệnh Matlab. Do đó bạn có thể sử dụng chức năng '*more on*' để xem thông tin theo từng trang màn hình, sau đó duyệt từng trang một bằng cách nhấn *phím bất kỳ*.

Đánh '*more off*' vào cửa sổ lệnh sẽ đưa Matlab trở về cách cư xử thông thường, không duyệt từng trang.

- Thông thường bạn có thể không nhớ chính xác tên của một *lệnh Matlab*, trong trường hợp đó bạn có thể sử dụng lệnh '*lookfor*' (tìm kiếm) như một sự trợ giúp.

Ví dụ, nếu bạn đánh vào dòng lệnh '*lookfor logarithm*', Matlab sẽ liệt kê tất cả các hàm từng biết của Matlab có liên quan tới Logarit.



Hình 1.3 : Giao diện cửa sổ Help của Matlab 7.0

1.8. History & Editing

- Trong giao diện mặc định của Matlab, cửa sổ '*Command History*' (lịch sử các lệnh) nằm ở góc phần tư bên dưới, phía trái. Trong cửa sổ này, các lệnh đã sử dụng trong các lần khởi động Matlab gần đây đều được lưu lại. Mỗi lần khởi động Matlab, toàn bộ các lệnh sử dụng trong lần đó sẽ được lưu lại dưới dạng một nhóm có thể đóng mở bằng nút biểu tượng '+' (expand)hoặc '-' (collapse) ở đầu dòng (dòng ghi mốc thời gian giữa hai dấu chú thích '*%*'):

VD: (+) *%-- 8/14/07 3:21 PM --%*

(-) *%-- 8/14/07 3:21 PM --%*

- Để gọi lại lệnh từ cửa sổ '*Command History*', bạn tìm đến lệnh đó bằng các thanh cuốn, rồi nháy đúp vào tên lệnh.

- Để gọi lại các lệnh bạn đã sử dụng từ dấu nhắc của cửa sổ lệnh, Matlab dùng các phím mũi tên (←↑→↓) trên bàn phím.

- Ví dụ, để gọi lại lệnh bạn gõ vào lúc gần nhất, bạn nhấn phím mũi tên lên (↑). Tiếp tục nhấn phím này, nó sẽ gọi tiếp lệnh trước đó. Phím mũi tên xuống gọi lại lệnh theo thứ tự ngược lại.

- Các phím mũi tên ← và → có thể dùng để thay đổi vị trí con trỏ trong dòng lệnh tại dấu nhắc của Matlab, như vậy chúng ta có thể sửa dòng lệnh. Thêm nữa, có thể dùng chuột cùng với bộ nhớ đệm để cắt, copy, dán và sửa văn bản tại dấu nhắc của dòng lệnh.

2. CHƯƠNG II: TÍNH TOÁN TRONG MATLAB

2.1. Matlab - một máy tính cá nhân

**** Giới thiệu các toán tử số học:***

- Các toán tử số học của Matlab hoạt động theo một cú pháp rất giống với cú pháp của các ngôn ngữ khác mà bạn có thể đã quen thuộc như Turbo Pascal, C, C++, Fortran, Java...

- Các toán tử cơ bản gồm có + - * / ^ = và chúng được dùng kết hợp với ngoặc đơn: **( )**. Toán tử '=' là toán tử gán. Toán tử ^ được dùng để cho lũy thừa: 2^4=16. Với những toán tử này ta có thể dùng Matlab như một máy tính cá nhân đơn giản.

**** Ví dụ:*** Bạn có thể đánh các lệnh dưới đây vào sau dấu nhắc lệnh: >>_.

```
>> A = 2 + 3/4*5
A =
    5.7500
>> B = 2^5 - 3*A
B =
   14.7500
>> A + B
ans =
   20.5000
```

Khi không có toán tử gán, Matlab trả kết quả của phép tính gần nhất vào biến '*ans = ...*' *(answer)*. Xem thêm mục 2.2.

**** Thứ tự ưu tiên tính toán:***

Trong ví dụ trên, Matlab đã tính như thế nào, 2 + 3/(4*5) hay 2 + (3/4)*5?

Matlab làm việc theo thứ tự ưu tiên sau:

1. các đại lượng trong ngoặc đơn,
2. lũy thừa 2 + 3^2 $\Rightarrow$ 2 + 9 = 11,
3. * /, làm việc từ trái qua phải (3*4/5 = 12/5),
4. + -, làm việc từ trái qua phải (3+4-5=7-5),

Vì vậy phép tính ở trên sẽ theo thứ tự ưu tiên 3.

**** Bộ các toán tử của Matlab:*** (*Xem thêm Help/Arithmetic operators*)

Toán tử	**Mô tả**
+	Cộng
-	Trừ
.*	Nhân mảng cùng kích thước (nhân phần tử với phần tử)

./	Chia mảng cùng kích thước (chia phần tử với phần tử)
.\	Chia mảng trái
:	Toán tử Hai chấm
.^	Lũy thừa mảng, lũy thừa từng phần tử với phần tử
.'	Chuyển vị mảng
'	Chuyển vị ma trận - ma trận liên hợp (MTLH phức)
*	Nhân (ma trận). Đại số tuyến tính.
/	Chia (ma trận), B/A ~ B*inv(A), chính xác hơn B/A=(A'\B')
\	Chia (ma trận) trái. A\B ~ inv(A)*B
^	Lũy thừa ma trận

2.2. Biến trong Matlab

**** Ví dụ:***

```
>> 2^3-3
ans =
     5
>> ans*6
ans =
    30
```

- Kết quả của phép tính thứ nhất được Matlab gán cho biến '*ans*', biến này được sử dụng cho phép tính thứ hai, qua đó giá trị của nó đã được thay đổi (được gán lại).

- Chúng ta có thể sử dụng tên riêng do ta đặt để lưu các giá trị số

```
>> x = 2^3-3
x =
     5
>> y = x*6
y =
    30
```

từ đó '*x*' có giá trị bằng 5 và '*y*' bằng 30. Chúng có thể được sử dụng cho những tính toán tiếp theo.

- Đây là các ví dụ về về *câu lệnh gán:* các *giá trị* được gán cho các *biến*. Cần phải gán một giá trị cho mỗi biến trước khi sử dụng biến đó trong câu lệnh tiếp theo.

**** Quy tắc đặt tên biến:***

- Tên biến hợp lệ cấu tạo bởi các chữ và số, bắt đầu bằng chữ. Nên đặt tên biến phản ánh giá trị mà nó đại diện cho.

Các tên sau hợp lệ:

Hsig, Let2try, Dhaluu, T1, V2, z25c5

Các tên sau không hợp lệ:

chu-ky, 2P, %x, @wru

- Chiều dài tên biến: Mặc dù tên biến có thể có độ dài tùy ý, nhưng Matlab sẽ chỉ sử dụng N ký tự đầu tiên của tên, vì vậy các biến khác nhau không được có N ký tự đầu tiên đều giống nhau.

```
N = namelengthmax        ↵
N =
    63
```

- Bạn có thể sự dụng hàm '*isvarname*' để kiểm tra tính hợp lệ của tên biến. Hàm trả về giá trị 1 nếu tên hợp lệ và 0 nếu tên đó không hợp lệ.

```
isvarname 8th_column     ↵
ans =
     0                   % Not valid - begins with a number
```

- Thông thường, tên biến không phụ thuộc vào chữ hoa - chữ thường. Vì thế '*xyz*' sẽ giống như '*xYz*'.

- Tránh đặt tên biến trùng với tên các hàm chuẩn, hoặc các từ khóa của Matlab. Vì như vậy thông thường bạn sẽ không thể sử dụng các hàm, từ khóa của Matlab nữa.

VD: Nếu bạn gán cho 1 biến tên là '*sqrt*' một giá trị, thì bạn sẽ không thể sử dụng hàm căn bậc hai (sqrt) nữa!

- Matlab đã đăng ký trước rất nhiều từ khóa (xem bằng lệnh '*iskeyword*'):

'break' 'case' 'catch' 'continue' 'else' 'elseif' 'end' 'for' 'function' 'global' 'if' 'otherwise' 'persistent' 'return' 'switch' 'try' 'while'

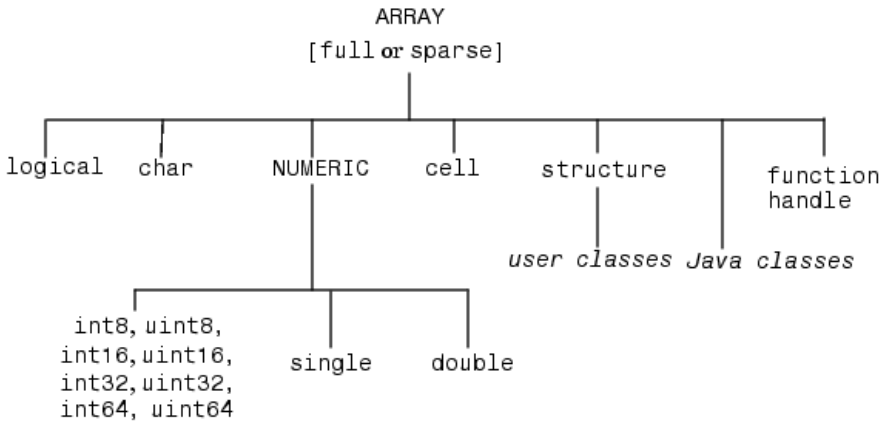
và các hàm, xem danh sách trong *Help/Functions/Categorical List*); và các hằng số. Một số hằng số và hàm thông dụng có thể xem ở mục 2.5 và 2.6.

2.3. Các kiểu dữ liệu - Định dạng kết quả

**** Tổng quát về các kiểu dữ liệu:***

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

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



Hình 2.1. Các kiểu dữ liệu của Matlab

**** Định dạng kết quả:***

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

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

format short	*: dấu phẩy thập phân cố định, 5 chữ số*
format long	*: dấu phẩy cố định, 15 chữ số*
format short e	*: ký hiệu khoa học, 5 chữ số*
format long e	*: ký hiệu khoa học, 15 chữ số*
format short g	*: dấu phẩy cố định hoặc di động, 5 chữ số*
format long g	*: dấu phẩy cố định hoặc di động, 15 chữ số*
format hex	*: format dạng Hexa (hệ 16)*
format '+'	*: dương (+), âm (-), và ký tự trắng (blank) ứng với 0*
format bank	*: Dollars và cents*
format rat	*: tỷ lệ xấp xỉ integer*

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

2.4. Các kiểu dữ liệu số & số phức

- Integer: ví dụ như -5 hay 9888.

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

- Một dạng ngắn cực kỳ hiệu quả cho việc nhập các số rất lớn hoặc rất bé là dạng ký hiệu 'e'. Chẳng hạn -1.23456e-7 là dạng ngắn của -1.23456 x 10^{-7}; và 8.76e+12 là dạng viết ngắn của 8.76 x 10^{12}. Ví dụ:

```
>> 1.23e-2        ↵
ans =
      0.0123
>> 5e6            ↵
ans =
      5000000
```

- Số phức: được nhập vào dưới dạng ***3+2*i*** hoặc ***3+2*sqrt(-1)***.

- Chuỗi: là một mảng tập hợp của các ký tự, được nhập vào dưới dạng 'abc' hoặc 'vi du day la mot chuoi'.

Trên đây là những kiểu dữ liệu cơ bản mà bạn sẽ rất thường dùng trong khóa học này. Để biết danh sách đầy đủ hơn, bạn có thể dùng lệnh '*help datatypes*' từ cửa sổ nhập lệnh.

2.5. Các ký tự, Chuỗi và Văn bản

- Khả năng xử lý văn bản trong tính toán rất hữu ích cho việc nhập/xuất kết quả từ/tới màn hình và file lưu trên đĩa. Để có thể quản lý văn bản, một loại dữ liệu là '*character*' được đưa vào Matlab. Một mảnh của văn bản đơn giản là một *chuỗi (vectơ)* hay một *mảng* các *ký tự*.

VD:
```
>> t1='A'          ↵
```

sẽ gán giá trị *A* cho một mảng ký tự tên '*t1*', kích thước 1 x 1.

```
>> t2='BCDE'       ↵
```

sẽ gán giá trị *BCDE* cho một mảng ký tự tên '*t2*', kích thước 1 x 4.

- Các chuỗi có thể được cộng với nhau bằng cách sử dụng các toán tử thao tác trong mảng.

VD:
```
>> t3=[t1, t2]     ↵
```

sẽ gán giá trị *ABCDE* cho một mảng ký tự tên '*t3*', kích thước 1 x 5.

```
>> t4=[t3, ' la 5 ky tu dau tien    ';...
'trong bang chu cai latinh.']              ↵
```

sẽ gán giá trị 'ABCDE la 5 ky tu dau tien ' '
'trong bang chu cai latinh.'

cho một mảng ký tự tên '*t4*', kích thước 2 x 26.

- Cần chú ý rằng số các ký tự ở hai dòng phải bằng nhau, nếu không việc thực thi câu lệnh trên sẽ dẫn tới một lỗi:

```
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.
```

- Dấu ba chấm '...' thể hiện rằng câu lệnh còn tiếp tục ở dòng sau.

**** Chuyển đổi giữa chuỗi và số***

- Đôi khi chúng ta cần chuyển một chuỗi thành một số tương ứng, hoặc ngược lại. Các công việc chuyển đổi này được thực hiện bởi lệnh:

- *'str2num':* chuyển một chuỗi thành số tương ứng

và hai lệnh:

- *'num2str':* chuyển một số thực thành chuỗi tương ứng
- *'int2str':* chuyển một chuỗi thành số tương ứng

- Những lệnh này rất hữu ích trong việc tạo ra các nhãn và tiêu đề một cách tự động, chẳng hạn như

2.6. Các hằng số dựng sẵn

Matlab định nghĩa sẵn nhiều hàm số rất hữu ích, bao gồm:

- ***pi****, π = 3.141592654...*
- ***i*** *và* ***j****, cả hai đều bằng phần ảo của số phức, =* ***sqrt(-1)***
- ***inf****, 'infinity' hay 'vô cùng'*
- ***NaN****, 'not-a-number' hay 'không phải là một số'*
- ***ans****, luôn được gán cho kết quả của lệnh tính trước đó*
- ...

Bạn nên tránh gán lại giá trị khác cho các hằng số nêu trên nếu có thể. Chỉ có một ngoại lệ là i và j, vì cả hai thường được sử dụng như các chỉ số của vòng lặp. Việc gán lại giá trị khác cho các hằng số này là chấp nhận được vì số phức luôn có thể thu được bằng cách sử dụng ***sqrt(-1)***.

2.7. Các hàm dựng sẵn

- Cũng như những ngôn ngữ bậc cao khác, Matlab thực thi các '*function*' (hàm) nhiều hơn '*procedure*' (chương trình con). Các hàm này bao gồm căn bậc hai (*sqrt*), lũy

thừa (exp), logarit (log, log10, log2), giá trị tuyệt đối (abs), và các hàm lượng giác (sin, cos, tan, atan,...). Ví dụ:

```
>> sin(45)        ↵
ans =
      0.8509
```

trả kết quả bằng sin của 45 radians... mà thực ra ý định của bạn là tính sin của 450, sẽ bằng $\frac{45\pi}{180}$ radians:

```
>> sin(45/180*pi) ↵
ans =
      0.7071
```

và kiểm tra lại xem bạn có thu được $\frac{\sqrt{2}}{2}$ như đã định không

```
>> sqrt(2)/2          ↵
ans =
      0.7071
```

- Chú ý rằng tất cả các tính toán của Matlab đều có lỗi làm tròn, mà đôi khi bạn lại thấy một cách không mong đợi. Ví dụ, bạn không nên ngạc nhiên khi thấy

```
>> tan(pi)        ↵
ans =
      -1.2246e-016
```

Nên nhớ rằng lỗi làm tròn có mặt ở khắp nơi, và chúng ta nên đơn giản tiếp nhận kết quả này như tan(π)=0.

**** Danh mục các hàm dựng sẵn phổ biến:***

Các hàm lượng giác:

sin	- hàm sin.
sind	- sin của argument tính theo độ.
sinh	- sin hypecbolic.
asin	- arcsin, hay hàm nghịch đảo của hàm sin.
asind	- hàm nghịch đảo của hàm sin, kết quả theo độ.
asinh	- hàm nghịch đảo của hàm sin hyperbolic.
cos	- hàm cos.
cosd	- cos của argument tính theo độ.
cosh	- cos hyperbolic.
acos	- hàm nghịch đảo của hàm cos.
acosd	- hàm nghịch đảo của hàm cos, kết quả theo độ.
acosh	- hàm nghịch đảo của hàm cos hyperbolic.

tan	- hàm tang.
tand	- tang của argument tính theo độ.
tanh	- tang hyperbolic.
atan	- hàm nghịch đảo của hàm tang.
atand	- hàm nghịch đảo của hàm tang, kết quả theo độ.
atan2	- hàm nghịch đảo của hàm tang 4 góc phần tư.
atanh	- hàm nghịch đảo của hàm tang hyperbolic.
cot	- hàm côtang.
cotd	- côtang của argument tính theo độ..
coth	- côtang hyperbolic.
acot	- hàm nghịch đảo của hàm côtang.
acotd	- hàm nghịch đảo của hàm côtang, kết quả theo độ.
acoth	- hàm nghịch đảo của hàm côtang hyperbolic.

Các hàm lũy thừa:

exp	- hàm mũ.
expm1	- tính chính xác exp(x)-1.
log	- logarit cơ số tự nhiên.
log1p	- tính chính xác log(1+x).
log10	- logarit cơ số 10.
reallog	- loga cơ số tự nhiên của số thực.
realsqrt	- căn bậc hai của một số >= 0.
sqrt	- căn bậc hai.
nthroot	- nghiệm thực bậc n của các số thực.

Các hàm liên quan đến số phức:

abs	- giá trị tuyệt đối.
angle	- góc pha.
complex	- xây dựng dữ liệu về số phức từ các phần thực và ảo.
conj	- liên hợp của phức.
imag	- phần ảo của phức.
real	- phần thực của phức.
isreal	- hàm logic, trả về giá trị '*true*' với mảng số thực.

cplxpair - sắp xếp các số về các cặp liên hợp phức.

Các hàm làm tròn và phần dư:

fix - làm tròn về phía 0.

floor - làm tròn về phía âm vô cùng.

ceil - làm tròn về phía dương vô cùng.

round - làm tròn về phía số nguyên gần nhất.

mod - mô đun (lấy phần dư của phép chia).

rem - lấy phần dư của phép chia (tương tự mod)

sign - hàm lấy dấu của một biến, trả về +1, 0, -1 (+, 0, -).

Ví dụ:

```
mod([1    2    3    4    5    6    7],3)        ↵
ans =
     1     2     0     1     2     0     1
```

2.8. Các phép toán quan hệ

**** Các toán tử quan hệ (so sánh):***

Toán tử	Cú pháp	Mô tả
<	A < B	Nhỏ hơn
<=	A <= B	Nhỏ hơn hoặc bằng
>	A > B	Lớn hơn
>=	A >= B	Lớn hơn hoặc bằng
==	A == B	Bằng
~=	A ~=B	Không bằng

**** Mô tả:***

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

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

```
>> X = 5*ones(3,3)          ↵
     X =
          5     5     5
          5     5     5
```

```
           5     5     5
>> Y= [1 2 3; 4 5 6; 7 8 9]   ↵
      Y =
           1     2     3
           4     5     6
           7     8     9
>> X >= Y                    ↵
      ans =
           1     1     1
           1     1     0
           0     0     0
```

Kết quả được trả về dưới dạng một ma trận có cùng kích thước với ma trận X và Y. Trong đó mỗi phần tử có giá trị 1 (đúng - true) hoặc 0 (sai - false) tùy thuộc vào sự logic của phép so sánh X>=Y.

2.9. Các phép toán logic

- Matlab biểu diễn ***đúng*** và ***sai*** bởi các số nguyên tố ***1*** và ***0***:

đúng = 1, sai = 0 (true = 1, false = 0)

- Ví dụ trong quá trình tính toán, biến x (x là một đại lượng vô hướng) nhận một giá trị bất kỳ, chúng ta có thể tiến hành các phép kiểm tra logic cho nó:

x == 2	xem x có bằng 2 không?
x ~= 2	xem x có khác 2 không?
x > 2	xem x có lớn hơn 2 không?
x < 2	xem x có nhỏ hơn 2 không?
x >= 2	xem x có lớn hơn hoặc bằng 2 không?
x <= 2	xem x có nhỏ hơn hoặc bằng 2 không?

- Đặc biệt chú ý tới cách viết trong Matlab là phép kiểm tra ***bằng nhau*** sử dụng hai dấu bằng viết liền nhau ==.

Ví dụ chúng ta có:

```
>> x = pi          ↵
x =
      3.1416
>> x ~= 3          ↵
ans =
      1
>> x ~= pi         ↵
ans =
```

```
     0
```

- Khi X là một véc tơ hay ma trận, các phép kiểm tra này sẽ được tiến hành cho từng phần tử của X.

Ví dụ ở mục trước (2.8), phép kiểm tra xem X có lớn hơn hoặc bằng Y hay không cho ta kết quả:

```
>> X >= Y                ↵
ans =
     1     1     1
     1     1     0
     0     0     0
```

- Chúng ta có thể phối hợp các phép kiểm tra logic, ví dụ $4 \le Y \le 6$

```
>> Y>=4 & Y<=6           ↵
ans =
     0     0     0
     1     1     1
     0     0     0
```

**** Các toán tử logic chính trong Matlab và ý nghĩa của chúng***

&	***và (and)***
|	***hoặc (or)***
~	***không (not)***

Chẳng hạn như `~=` nghĩa là không bằng

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

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

- khi tính phân bố lưu tốc theo chiều sâu (tính từ đáy kênh), ta có thể bỏ qua lớp biên với chiều sâu nước z thỏa mãn điều kiện *z <= delta* (*delta* là chiều dày lớp biên).
- khi tính tần suất vượt quá một ngưỡng N nào đó của chiều cao sóng trong bộ số liệu đo đạc về sóng – véctơ H, ta có thể kiểm tra điều kiện *H(i) > N*.

VD:

```
>> H = [4.7506   1.1557   3.0342   2.4299   4.4565... (shift + ↵)
    3.8105    2.2823    0.0925    4.1070    2.2235];      ↵
>> N = 3;          ↵
>> kiemtra = H > N      ↵
kiemtra =
```

```
     1     0     1     0     1     1     0     0     1     0
>> H = H.* kiemtra      ↵
H =
    4.7506   0   3.0342   0   4.4565   3.8105   0   0   4.1070   0
```

Nhờ đó véctơ H bây giờ chỉ còn chứa các số liệu sóng có chiều cao lớn hơn ngưỡng 3 m.

2.10. Kết hợp nhiều lệnh trên một dòng; Ẩn kết quả tính.

Dấu phẩy (**,**) và dấu chấm phẩy (**;**) là những ký tự có ý nghĩa đặc biệt trong Matlab, và sẽ chứng tỏ là rất hữu ích.

**** Kết hợp nhiều lệnh trên một dòng:***

Toán tử phẩy (**,**) được dùng để nhóm nhiều lệnh trên một dòng, ví dụ:

```
>> x=3.5, y=-5.0, x^3 - y          ↵
x =
      3.5000
y =
      -5
ans =
      47.8750
```

**** Ẩn kết quả tính (Surpessing output)***

- Thông thường, chúng ta không muốn theo dõi tất cả các tính toán trung gian, hay ta muốn ẩn đi một câu lệnh, một diễn giải. Khi đó ta dùng dấu chấm phẩy (**;**). Ví dụ:

```
>> x=3.5; y=-5.0; x^3 - y          ↵
ans =
      47.8750
```

Trong ví dụ trên, kết quả của hai lệnh gán đầu tiên đã được ẩn đi.

3. CHƯƠNG III: VECTƠ

3.1. Giới thiệu

- Đại số tuyến tính là trái tim và là phần hồn của Matlab. Trong thực tế thì ban đầu Matlab là từ viết tắt của "matrix laboratory'. Vì vậy hơn bất kỳ ngôn ngữ nào khác, Matlab khuyến khích và trông đợi bạn tận dụng mọi khả năng của các ***mảng***, ***vécto*** và ***ma trận***.

*** Một vài thuật ngữ trong chương III (Véctơ) và IV (Ma trận):**

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

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

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

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

Nghĩa là kích thước của mảng `r` sẽ là 2x3 (2 hàng, 3 cột).

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

- ***Vécto*** là một ma trận mà một chiều chỉ có chỉ số =1. Cụ thể, một ***véctơ hàng*** là một ma trận chỉ có một hàng (kích thước *1 x n*), còn một ***véctơ cột*** là một ma trận chỉ có một cột (kích thước *m x 1*).

*- Mặc dù khái niệm **mảng** tổng quát hơn và ít tính chất toán học hơn một **ma trận**, nhưng hai thuật ngữ này vẫn thường được dùng lẫn với nhau. Hơn nữa, Matlab đôi khi không có một sự phân biệt chính thức nào, thậm chí là giữa một đại lượng vô hướng và một ma trận kích thước 1x1.*

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

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

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

- Sau đây chúng ta sẽ xem xét lần lượt hai loại véctơ chính trong Matlab: véctơ hàng và véctơ cột.

3.2. Véctơ hàng

Véctơ hàng là chuỗi các số được phân cách bởi dấu phẩy hoặc khoảng trống. Số lượng các đầu số được gọi là 'chiều dài' của véctơ, và mỗi đầu số thường được nhắc đến như 'phần tử', hoặc 'hợp phần' của véctơ. Cú pháp cơ bản để nhập 1 véctơ là một chuỗi các giá trị được bao trong cặp ngoặc vuông [ ]. Ví dụ:

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

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

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

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

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

- Như đã đề cập ở trên, chúng ta có thể xem hoặc thay đổi giá trị của những phần tử riêng biệt của véctơ:

```
>> w(2) = -2, w(3)
w =
      1   -2   3
```

```
ans =
      3
```

3.3. Vectơ cột

Véctơ cột có cấu tạo tương tự như véctơ hàng. Khi định nghĩa véctơ cột, các phần tử được phân cách nhau bởi ký tự '***;***' hoặc bởi '***newlines***'. Ví dụ:

```
>> c = [ 1; 3; sqrt(5)] ↵
c =
      1.0000
      3.0000
      2.2361
>> c2 = [3                (shift + ↵)
4                         (shift + ↵)
5]                        ↵
c2 =
      3
      4
      5
>> c3 = 2*c - 3*c2        ↵
c3 =
      -7.0000
      -6.0000
      -10.5279
```

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

3.4. Toán tử hai chấm (:)

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

```
>> x = 1:4          ↵
x =
     1 2 3 4
>> y = 3:7          ↵
y =
     3 4 5 6 7
>> z = 1:-1         ↵
z =
   Empty matrix: 1-by-0
```

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

```
>> 0.32:0.1:0.6    ↵
ans =
      0.3200 0.4200 0.5200
>> -1.4:-0.3:-2    ↵
ans =
      -1.4000 -1.7000 -2.0000
```

- Toán tử '**:**' còn được dùng để trích xuất một phần của véctơ. Giả thiết chúng ta có véc tơ

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

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

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

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

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

Hãy xem `r(1:2:7) ↵` cho ta kết quả như thế nào?

3.5. Làm việc với vectơ & ma trận (mảng)

**** Các phép toán số học:***

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

```
v1 =     |1     2     3|
v2 =     |4     5     6|
>> v1+v2    ↵
ans =
     5   7   9
>> v3=3*v1  ↵
v3 =
     3   6   9
>> v4=2*v1-3*v2   ↵
v4 =
   -10   -11   -12
>> v5=[10 11 12 13];    ↵
>> v4+v5
??? Error using ==> plus
Matrix dimensions must agree.
```

- Một véc tơ cũng có thể nhân dược với một đại lượng vô hướng (một số), thao tác được Matlab tiến hành ***với từng phần tử***.

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

```
>> v1.*v2          ↵
ans =
     4   10   18
>> v2./v1          ↵
```

```
ans =
    4.0000   2.5000   2.0000
```

Toán tử lũy thừa có thể được sử dụng theo hai cách, với lũy thừa số vô hướng hoặc lũy thừa véctơ:

```
>> v2.^2          ↵
ans =
    16   25   36
>> v2.^v1         ↵
ans =
     4   25  216
```

Lý do Matlab cần các toán tử '**.**' này sẽ được làm rõ hơn trong chương 5.

- Kí hiệu **.*** thực ra có ý nghĩa của một phép nhân ma trận, tương ứng với **.*** cho các véctơ.

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

VD: đoạn mã lệnh dưới đây tính toán giá trị biểu thức $2\sqrt{x} + \frac{x}{y} - x^3 \cos(\pi.y)$ theo-từng-phần-tử. Tính với mỗi một phần tử trong véctơ x và y:

```
>> x = [1 2 3]; y = [4 5 6];              ↵
>> s = 2*sqrt(x) + x./y - x.^3.*cos(pi*y) ↵
s =
      1.2500   11.2284  -23.0359
```

Lưu ý các phép tính của các đại lượng vô hướng trên các véctơ khác như thế nào với cách làm việc phần tử với phần tử, ví dụ: `2 * sqrt(x)` rõ ràng là nhân số vô hướng với véctơ, trong khi `x/y` thì khác, vì vậy ở đây ta cần phải sử dụng `x./y`

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

**** Ghép các véctơ:***

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

```
>> w = [1 2 3], z = [8 9]      ↵
>> cd = [2*z,-w], sort(cd)     ↵
w =
      1 2 3
z =
      8 9
cd =
      16 18 -1 -2 -3
ans =
      -3 -2 -1 16 18
```

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

**** Các lệnh cho thông tin về ma trận (véc tơ):***

size - kích thước theo mỗi chiều

length - kích thước của chiều dài nhất (đặc biệt là cho véctơ)

ndims - số chiều

find - các chỉ số của các phần tử khác 0

**** Chuyển vị ma trận:***

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

```
>> w, w', c, c'   ↵
w =
      1 -2 3
ans =
      1
      -2
      3
c =
      1.0000
      3.0000
      2.2361
ans =
      1.0000 3.0000 2.2361
>> t = w + 2*c'   ↵
t =
      3.0000 4.0000 7.4721
>> T = 5*w'-2*c   ↵
T =
      3.0000
      -16.0000
      10.5279
```

3.6. Xử lý dữ liệu với các hàm dựng sẵn cho vectơ & ma trận

(Xem thêm Help và bài giảng trên lớp)

**** Sắp xếp dữ liệu: sort***

**** Tìm giá trị lớn nhất: max***

**** Tìm giá trị nhỏ nhất: min***

**** Tính tổng: sum***

**** Tìm giá trị trung bình: mean***

**** Tìm độ lệch quân phương: std***

4. CHƯƠNG IV: MA TRẬN ĐẠI SỐ & TUYẾN TÍNH

4.1. Định nghĩa và khởi tạo ma trận

- Ma trận là dạng cấu trúc dữ liệu cơ bản của Matlab. Và như đã đề cập ở chương trước, các véctơ chẳng qua là những dạng đặc biệt của ma trận có kích thước (1×n) hoặc (m×1).

- Từ dấu nhắc của cửa sổ nhập lệnh, đánh lệnh `help elmat` và `help matfun` để có một danh sách các ***lệnh*** và ***hàm*** làm việc với ma trận trong Matlab.

- ***Cú pháp*** của việc định nghĩa và khởi tạo ma trận rất giống với những gì bạn đã thấy với véctơ: các khoảng trống (hoặc dấu phẩy) phân cách các phần tử trong một hàng, và các dấu chấm phẩy là ký hiệu cho biết bắt đầu một hàng mới sau đó.

Ví dụ khi bạn đánh:

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

Matlab sẽ đưa ra kết quả

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

Và biến A bây giờ chứa một ma trận 3×4.

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

`A(3,2) = 0` sẽ thay thể giá trị phần tử cột 2 hàng cuối cùng của ma trận A thành 0.

- Có nhiều lệnh để khởi tạo một số dạng ma trận đặc biệt, ví dụ

`zeros(n,m)`	↵	-	tạo ma trận với tất cả các phần tử = 0
`ones(n,m)`	↵	-	tạo ma trận với tất cả các phần tử = 1
`eye(n)`	↵	-	tạo ma trận đơn vị kích thước n × n.

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

4.2. Một số ma trận đặc biệt

* Các lệnh dùng để xây dựng ma trận và mảng:

eye - ma trận đơn vị

zeros	-	ma trận với tất cả các phần tử = 0
ones	-	ma trận với tất cả các phần tử = 1
diag	-	ma trận đường chéo (hoặc trích xuất một đường chéo)
toeplitz	-	ma trận với mỗi đường chéo bằng 1 hằng số
triu	-	ma trận tam giác trên
tril	-	ma trận tam giác dưới
rand	-	ma trận với các phần tử ngẫu nhiên (từ –1 đến 1)
linspace	-	ma trận với các phần tử cách đều nhau
cat	-	móc nối các ma trận với nhau theo một chiều đã định
repmat	-	xây dựng ma trận mới bằng cách lặp một véc tơ theo 1 chiều (hoặc nhiều chiều) đã định

Xem help để có hướng dẫn chi tiết hơn về cách sử dụng các hàm này.

4.3. Các phép toán với từng phần tử trong ma trận

4.4. Các phép toán với ma trận

4.5. Giải phương trình đại số

4.6. Giải hệ phương trình đại số tuyến tính

4.7. Tìm nghiệm của đa thức

4.8. Giải phương trình phi tuyến

4.9. Giải phương trình vi phân

4.10. Các lệnh hữu ích khác

`inv(A)`, nghịch đảo ma trận

`det(A)`, định thức ma trận

`trace(A)`, vết ma trận

`cond(A)`, số điều kiện, biểu diễn tình trạng (tốt/xấu) của ma trận

`norm(A)`, chuẩn ma trận

`eig(A)`, giá trị riêng và vectơ riêng của ma trận

Nội dung chương này gắn với bài giảng trên lớp. Sinh viên xem thêm phần trợ giúp và tham khảo thêm cuốn "Phương pháp số cho kĩ sư", tác giả S. Chapra và R. Canale, được dịch từ tiếng Anh làm tài liệu cho sinh viên chuyên ngành.

5. CHƯƠNG V: SCRIPTS VÀ FUNCTIONS (M-FILES)

5.1. Giới thiệu M-file

- Các ***M-file*** là các file ASCII (file text) bình thường chứa các (câu) lệnh Matlab. Một điều thiết yếu là các file đó có phần mở rộng là '*.m*' (VD: *baitap2.m*) và vì lý do này, chúng thường được biết đến dưới cái tên các m-files.

- Có hai loại m-file: ***Script*** và ***Function***. Các Script và Function files cũng hoạt động gần như các Procedures và Functions trong các ngôn ngữ lập trình thông dụng khác.

- Về cơ bản nội dung của một script file được hiểu giống hệt như khi nội dung đó được gõ vào tại dấu nhắc cửa sổ nhập lệnh. Hiểu đơn giản thì nó chỉ thực hiện một chuỗi các câu lệnh của Matlab. Tuy nhiên trong thực tế nhiều người dùng ưa thích sử dụng Matlab bằng cách đánh tất cả các lệnh vào script file và chạy (các) file đó. Ưu điểm của phương pháp sử dụng script là:

  • Tạo ra và xem xét, chỉnh sửa một chuỗi nhiều dòng lệnh (thường là 4, 5 dòng trở lên).

  • Có thể dễ dàng xem lại hoặc thực hiện lại công việc của bạn sau này.

  • Chạy các tính toán (công việc) đòi hỏi cường độ cao của CPU trên nền, xử lý kết quả và lưu lại tự động, cho phép bạn log-off (liên quan tới UNIX).

5.2. Biên soạn và thực thi M-file

- **Biên soạn:** Matlab cung cấp cho ta một công cụ biên soạn các m-file khá tốt, đó là ***Matlab editor***. Tuy nhiên bạn có thể tự do sử dụng các ứng dụng soạn thảo khác cho file text của Windows như Notepad, Textpad...

- Bạn có thể khởi động Matlab Editor bằng nhiều cách: Từ menu File/New/M-file, hoặc nhấn tổ hợp phím tắt 'Ctrl – N', hay cách nhấn vào nút '*New Doccument*' trên thanh công cụ, cách đánh vào cửa sổ nhập lệnh '***edit***' và tên file (nếu file chưa tồn tại trong thư mục hiện thời, Matlab sẽ hỏi bạn để khẳng định rằng bạn muốn tạo ra một file mới với tên như vậy)

- Soạn thảo các câu lệnh của bạn và *Save*.

- Để biết trong thư mục hiện tại (*current directory*) có những m-file nào, bạn có thể sử dụng lệnh

```
>> what     ↵
```

- Để xem nội dung của một m-file, bạn nháy đúp vào file đó để mở nó ra hoặc đánh lệnh

```
>> type  tên_file ↵
```

- **Thực thi:** Để có thể thực thi một m-file, nó cần phải tồn tại trong thư mục hiện thời (xem cửa sổ Current Directory). Bạn có thể di chuyển giữa các thư mục trong ổ cứng gần giống như với trình duyệt Exprorer của Windows, hoặc dùng lệnh ***editpath*** (*path* là đường dẫn đến thư mục mà Matlab sẽ tìm kiếm file ở đó).

- **Biên dịch:** không cần thiết biên dịch cả hai loại M-file của Matlab. Muốn thực hiện các lệnh chứa trong file này rất đơn giản, bạn chỉ cần đánh tên file (không cần phần mở rộng '*.m*') từ dấu nhắc cửa sổ lệnh . Các chỉnh sửa đã tiến hành với file và ghi lại vào ổ đĩa sẽ được thực thi khi bạn gọi function hay script đó lần sau.

Ví dụ gọi thực thi các lệnh có trong file *baitap2.m* như sau:

```
>> baitap2        ↵
```

Chỉ có các thông số đầu ra sẽ được thể hiện trên màn hình, chứ không phải bản thân các câu lệnh.

- Để có thể xem các câu lệnh có trong file cùng lúc với các thông số đầu ra, bạn đánh lệnh

```
>> echo on        ↵
```

và lệnh '*echo off*' sẽ tắt chức năng này.

5.3. Chú thích (comments)

- Một dạng câu quan trọng trong M-file là câu chú thích, được bắt đầu bằng ký tự phần trăm (*%*). Bất cứ phần text nào sau ký tự '*%*' trên một dòng lệnh sẽ được Matlab bỏ qua không thực hiện (trừ trường hợp ký tự % là một phần của chuỗi ký tự giữa hai dấu nháy ‘ ’).

- Mục đích chính của tính năng này là cho phép bổ sung các câu chú thích (comments) vào script file, mô tả rõ ràng hơn mục đích, tính năng các lệnh, đoạn, vòng lặp, biến...

- Hơn nữa, khối các câu chú thích đầu tiên trong một M-file sẽ hoạt động như một hướng dẫn sử dụng m-file của bạn, và sẽ hiện ra ở cửa sổ nhập lệnh khi bạn sử dụng lệnh help + tên_m-file.

Ví dụ: giả sử trong file `baitap2.m` của bạn có nội dung sau:

```
% script nay giai quyet cac bai tap ve nha
% lien quan toi kien thuc chuyen nganh ky thuat bien
%
z= rand(1);                 % muc nuoc bien
a=omega*t*sin(2*pi);        % bien do song...
```

Thì khi một người dùng khác ngồi vào máy tính của bạn, muốn biết những thông tin cơ bản nhất xem file `baitap2.m` viết về vấn đề gì, họ có thể đánh vào cửa sổ lệnh:

```
help baitap2      ↵
```

và kết quả nhận được sẽ là

```
script nay giai quyet cac bai tap ve nha
lien quan toi kien thuc chuyen nganh ky thuat bien
```

5.4. Các hàm m-file (function m-files)

- Trước tiên chúng ta cần phân biệt các hàm m-file và các hàm số dựng sẵn, hàm trong một dòng.

- Hàm dựng sẵn, VD như sqrt( ), log( ), exp( ), sin( )...
- Hàm trong 1 dòng (inline function): là cách đơn giản nhất mà người dùng có thể định nghĩa một hàm, VD: Dòng lệnh dưới đây sẽ khai báo hàm $f(x) = x.\sin(x) + 2$ và tính giá trị hàm tại $x = 5$ bằng cách chuyển hàm này cho lệnh ***inline*** của Matlab trong một cặp dấu nháy ‘ ’:

```
>> f = inline('x*sin(x)+2'), f(5)
f =
     Inline function:
     f(x) = x*sin(x)+2
ans =
   -2.7946
```

- Hàm với m-file: Dùng cho các hàm phức tạp hơn, chẳng hạn như có chứa các vòng lặp, câu điều kiện... bạn cần dùng m-file để khai báo các hàm đó.

- Sau nữa chúng ta cần phân biệt các hàm m-file và các script-file:

- Script m-file, như đã đề cập ở phần trước, không phải là một hàm. Nó không có các tham số đầu vào cũng như đầu ra, và đơn giản nó chỉ thực hiện một chuỗi các câu lệnh của Matlab, với các biến được định nghĩa trong khong gian làm việc.
- Hàm m-file khác với script m-file ở chỗ nó có một dòng định nghĩa hàm, qua đó liên hệ giữa các tham số đầu vào và đầu ra.

Hàm là cách chủ yếu để phát huy khả năng của Matlab. So với các script, các hàm có khả năng phân chia nhiệm vụ tốt hơn nhiều.

Một ví dụ về hàm trong Matlab có thể tham khảo bài tập 4 (tính diện tích tam giác), chương 8 trong giáo trình này.

**** Các bước chính cần tuân theo khi khai báo một hàm trong Matlab là:***

- Đặt têm cho hàm, lưu ý rằng tên đó không được xung đột với các tên đã được Matlab dành trước. Trong ví dụ này tên hàm là *dientich* vì vậy các định nghĩa của nó sẽ được lưu trong một file tên là *dientich.m*

- Dòng đầu tiên của file này cần có dạng thức như sau:

```
function[các outputs] = tên_hàm(các inputs)
```

Lấy ví dụ trong bài toán của chúng ta, biến đầu ra S (diện tích) là một hàm số của các biến đầu vào a, b, c (là chiều dài của ba cạnh). Do đó dòng đầu tiên của m-file hàm dientich sẽ là:

```
function [S] = area(a,b,c)
```

- Soạn thảo hướng dẫn sử dụng cho hàm (không bắt buộc, xem thêm phần chú thích - Comments). Mô tả ngắn gọn mục đich của hàm và làm thế nào để sử dụng nó. Các dòng này cần bắt đầu bằng ký tự %, hay chính là các dòng chú thích mà ta đã đề cập, và Matlab sẽ bỏ qua nó khi thực thi hàm.

- Cuối cùng và cũng là quan trọng nhất: soạn thảo mã lệnh thực thi nội dung của hàm. Đi cùng với nội dung ta cũng cần đầy đủ các câu chú thích để người dùng khác có thể hiểu được quá trình ta đang làm.

Một m-file hàm hoàn chỉnh có thể trông như sau (theo ví dụ trên của chúng ta)

```
function [A] = dientich(a,b,c)
% Tinh dien tich cua mot tam giac
% khi biet chieu dai 3 canh la a, b va c.
% Dau vao:
% a,b,c: Chieu dai cua cac canh
% Dau ra:
% A: Dien tich tam giac
% Cach su dung (cu phap):
% Dientichcantinh = dientich(2,3,4);
% Nguoi viet: Ng.Ba.Tuyen, 2007.
    s = (a+b+c)/2;
    A = sqrt(s*(s-a)*(s-b)*(s-c));
%%%%%%%%% ket thuc dientich %%%%%%%%%%
```

Ta thấy rằng chú thích ở đây khá đầy đủ, và người dùng sau có thể dễ dàng hiểu được nội dung cũng như cách sử dụng hàm *dientich* bằng cách đánh lệnh *help dientich* ↵ từ cửa sổ nhập lệnh, hướng dẫn thu được sẽ như sau:

```
>> help dientich        ↵
  Tinh dien tich cua mot tam giac
  khi biet chieu dai 3 canh la a, b va c.
  Dau vao:
  a,b,c: Chieu dai cua cac canh
  Dau ra:
  A: Dien tich tam giac
  Cach su dung (cu phap):
  Dientichcantinh = dientich(2,3,4);
  Nguoi viet: Ng.Ba.Tuyen, 2007.
```

Thử sử dụng hàm vừa lập để tính diện tích một tam giác khác:

```
>> dientich(4, 5, 7)
ans =
    9.7980
```

Như vậy chúng ta đã đi qua các bước cơ bản từ khai báo một hàm bằng m-file, soạn thảo nội dung, mã lệnh, và sử dụng hàm. Hãy sử dụng help để có hiểu biết sâu hơn về hàm trong Matlab.

** Một khía cạnh quan trọng khác của hàm M-file là hầu hết các hàm xây dựng trong Matlab (trừ những hàm lõi toán học) đều là các M-file mà bạn có thể đọc và copy. Đây là một cách rất tốt để học hỏi, luyện tập lập trình – và cũng là một mẹo.*

5.5. Câu lệnh rẽ nhánh (if và switch)

- Thông thường một hàm cần rẽ nhánh tùy theo các điều kiện thực thi. Matlab cung cấp cho ta các công cụ để làm việc này cũng như hầu hết các ngôn ngữ lập trình khác.

**** Câu lệnh if...elseif...else...end***

- Ví dụ sau minh họa hầu hết các tính năng của ***if***

```
if isinf(x) | ~isreal(x)
    disp('So lieu dau vao xau!')
    y = NaN;
elseif (x == round(x)) && (x > 0)
    y = prod(1:x-1);
else
    y = gamma(x);
end
```

- Ta thấy các điều kiện cho câu lệnh if có thể liên quan tới các toán tử quan hệ đã đề cập ở chương 2, cũng có thể liên quan tới các hàm cho ta giá trị logíc (isinf, ~isreal... để kiểm tra xem x có phải là số vô cùng, hay x không phải là số thực...).

**** Câu lệnh switch...case...case... case...otherwie...end***

- Bộ câu lệnh ***if/elseif*** chỉ hữu ích trong trường hợp chỉ có một vài lựa chọn. Còn khi có một số lượng lớn các lựa chọn khả dĩ, thông thường ta dùng ***switch*** để thay thế. Ví dụ:

```
switch donvi
    case 'Chieudai'
        disp('met')
    case 'The tich'
        disp('lit')
    case 'Thoi gian'
        disp('giay')
    otherwise
        disp('Toi chiu thua')
end
```

- Diễn giải của lệnh ***switch*** có thể là một chuỗi hoặc một số. Trường hợp đầu tiên phù hợp với ***case*** thì các lệnh của nó sẽ được thực thi.

- Có thể sử dụng ***otherwise*** hoặc không. Trong trường hợp có sử dụng, thì Matlab thực thi các lệnh sau ***otherwise*** nếu không có trường hợp nào phù hợp với các ***case***.

5.6. Vòng lặp (for và while)

**** Vòng lặp for...end***

- Được sử dụng khi ta muốn lặp một đoạn mã lệnh cho một số lần tùy ý (thực ra ta sẽ ít dùng đến nó trong Matlab hơn là trong các ngôn gnữ lập trình khác, bởi vì Matlab cung cấp cho ta toán tử dấu hai chấm, **:**)

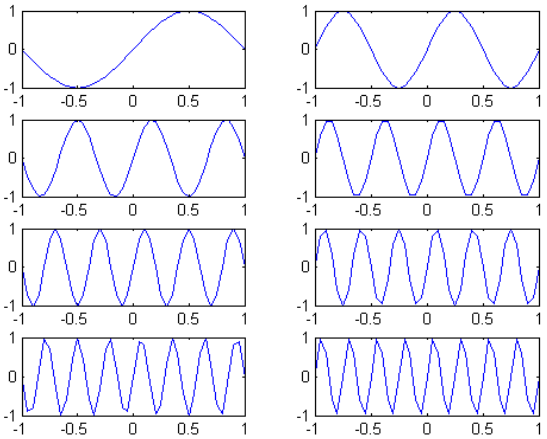
- Ví dụ, vẽ đồ thị sin(n.π.x) trong khoảng -1≤ x ≤1 với các giá trị khác nhau của n = 1, 2, ...8.

- Thực thi: Chúng ta có thể đưa ra 8 lệnh vẽ riêng rẽ, nhưng sẽ dễ dàng hơn nhiều nếu ta sử dụng một vòng lặp. Dạng đơn giản nhất của nó sẽ là

```
>> x = -1:.05:1;          ↵
>> for n = 1:8            (shift + ↵)
subplot(4,2,n), plot(x,sin(n*pi*x))        (shift + ↵)
end                       ↵
```

Tất cả các lệnh giữa hai dòng bắt đầu bằng '***for***' và kết thúc bằng '***end***' đều được lặp đi lặp lại với n=1 lần thứ nhất, n=2 lần thứ 2... cho tới khi n=8.

Lệnh subplot tạo ra một mảng 4x2 cửa sổ đồ thị con trong một đồ thị chính (dãy đồ thị, xem Mục 6.6). Ở lần lặp thứ *n*, một hình sẽ được vẽ lên cửa sổ đồ thị con thứ *n*.



*Hình 5.1: Minh họa cho vòng lặp **for**...**end***

**** Vòng lặp while...end***

- Được sử dụng khi bạn muốn thực hiện lặp đi lặp lại một đoạn mã lệnh của Matlab cho tới khi một điều kiện (logic) nào đó được thỏa mãn, nhưng ta không thể nói trước nó sẽ cần lặp bao nhiêu lần. Khi đó chúng ta có thể sử dụng vòng lặp này.

- Ví dụ, tìm giá trị lớn nhất của n sao cho tổng dưới đây vẫn nhỏ hơn 100?

$$1^2 + 2^2 + 3^2 + ... + n^2$$

- Mã lệnh cho Matlab thực thi nhiệm vụ trên:

```
>> S = 1; n = 1;                ↵
>> while S+ (n+1)^2 < 100       (shift + ↵)
   n = n+1; S = S + n^2;        (shift + ↵)
   end                          ↵
>> [n, S]          ↵
ans =
      6   91
```

- Ví dụ khác: Tìm giá trị gần đúng của nghiệm phương trình *x=cos(x)* ?

5.7. Đọc dữ liệu từ file và ghi ra file

- Nhập dữ liệu trực tiếp từ bàn phím sẽ trở nên không thể (không thực tế) khi

- Lượng dữ liệu quá lớn
- Dữ liệu đó được dùng cho phân tích nhiều lần

Trong những trường hợp này thì người sử dụng Matlab sẽ chọn cách nhập/xuất dữ liệu với ***file dữ liệu***.

- Hai lệnh save và load mà ta đã học ở chương 2 cũng có chức năng ghi và đọc giá trị của các biến vào/từ đĩa.

- Khi làm việc với file dữ liệu, một điều cốt yếu cần lưu ý là ***định dạng của dữ liệu*** phải đúng. Định dạng dữ liệu là chìa khóa quyết định việc biên dịch dữ liệu. Có hai dạng file dữ liệu: formatted và unformatted (có định dạng và không định dạng).

- File dữ liệu có định dạng sử dụng cách định dạng chuỗi để khai báo chính xác xem dữ liệu được lưu ở vị trí nào và như thế nào.
- File dữ liệu không định dạng thì khác, nó chỉ định rõ định dạng của số.

Cách đơn giản nhất để học cách ***làm việc với file dữ liệu*** là thông qua ví dụ sau:

Giả sử dữ liệu dạng số được lưu trong file có tên 'table.dat' trong thư mục hiện hành, dữ liệu như sau

```
100   2256
200   4564
300   3653
400   6798
500   6432
```

3 lệnh sau

```
>> fid = fopen('table.dat','r');
>> a = fscanf(fid,'%3d%4d');
>> fclose(fid);
```

sẽ lần lượt làm các nhiệm vụ:

- Mở một file để đọc, việc này được chỉ định bằng chuỗi '***r***', (***r*** là viết tắt của *read*). Biến ***fid*** được gán cho một giá trị bằng 1 số nguyên tố duy nhất, đặc trưng cho file sẽ sử dụng (số này còn gọi là số chỉ thị của file). Sau này mỗi khi nhắc đến file này chúng ta sẽ sử dụng số chỉ thị ***fid***.
- Đọc vào bộ nhớ từng cặp số từ file (file có số chỉ thị là ***fid***), một số có 3 chữ số và một số có 4 chữ số.
- Đóng file (file có số chỉ thị là ***fid***).

Quá trình này tạo ra một véc tơ cột chứa các phần tử 100 2256 200 4564 ...500 6432. Véctơ này có thể được chuyển đổi về ma trận 5x2 bằng lệnh:

```
A = reshape(2,2,5)';    ↵
```

Tuy nhiên, thường ta sẽ không đọc/ghi file thủ công như vậy. Matlab hỗ trợ một loạt định dạng file dữ liệu quan trọng. Với các file số liệu phân tách (delimited), Matlab cung cấp lệnh `dlmread` và `dlmwrite`. Lệnh đọc file này được minh hoạ ở Mục 7.4.

Hãy xem thêm trợ giúp Help để biết thêm thông tin về các định dạng file này.

6. CHƯƠNG VI: ĐỒ THỊ DẠNG ĐƯỜNG

6.1. Biểu diễn đường quá trình

Trường hợp đơn giản nhất là biểu diễn sự biến thiên tăng giảm số liệu trong một dãy. Chẳng hạn với dãy số liệu mực nước z đo được ta có thể biểu diễn dưới dạng đường quá trình như sau:

```
z = [-0.05  0.18  0.28  0.33  0.19  0  -0.26  -0.35  -0.31  -0.22  0.05
0.14  0.31  0.38 0.18  0.09  -0.11  -0.20  -0.36  -0.11  0.08];
plot(z)
```

Chú ý ở lệnh gán trên rằng ở lệnh gán trên, mã lệnh kéo dài xuống dòng dưới do hạn chế bề ngang của tài liệu. Khi lập trình *không* ấn **Enter** vì máy sẽ hiểu nhầm z thành một ma trận.

Lệnh `plot(z)` sẽ vẽ biểu đồ dạng đường với số liệu cho bởi vec-tơ z. Trường hợp này trục hoành sẽ đánh số thứ tự lần lượt 1, 2,... Điều này không giúp ích gì trong trường hợp thông thường khi trục hoành cần biểu thị khoảng cách không gian hoặc thời gian. Chẳng hạn, nếu số liệu z ở ví dụ trên là mực nước đo được tại các thời điểm 0 s, 10 s, 20 s... (cách nhau 10 giây) thì ta có thể bổ sung mã lệnh như sau:

```
figure;
t = 0:10:200;
plot(t, z);
```

Ở đây lệnh `figure` có tác dụng tạo ra một hình mới.

Chú ý: Các vec-tơ cần vẽ thường có rất nhiều phần tử, và do đó ta cần dùng dấu ; ở cuối câu lệnh để ngăn không cho máy hiện lại nội dung của toàn bộ vec-tơ.

Tóm lại để vẽ biểu đồ dạng đường nói chung ta theo 3 bước sau:

1) Phát sinh một vec-tơ chứa các tọa độ x của các điểm

2) Phát sinh một vec-tơ chứa các tọa độ y, có thể là từ số liệu sẵn có, hoặc là một hàm tính từ các giá trị tương ứng của x. Trong trường hợp sau cần lưu ý phép tính cần được áp dụng cho từng phần tử một.

3) Thực hiện lệnh vẽ `plot(x, y)`

Trong các biểu đồ ta phải ghi tiêu đề của biểu đồ và điền các đại lượng và đơn vị lên các trục (x, y). Có thể thực hiện việc này bằng cách:

```
title('Tiêu đề biểu đồ')
```

```
xlabel('Tiêu đề trục x')
ylabel('Tiêu đề trục y')
```

Chẳng hạn với ví dụ về mẫu quan trắc mực nước theo thời gian như trên ta có thể viết:

```
title('Qua trinh muc nuoc thuc do')
xlabel('t (s)')
ylabel('z (m)')
```

Kết quả biểu đồ thu được như trên Hình 6.1. Trong một số bản vẽ kĩ thuật, cần biểu diễn các số mũ hoặc kí hiệu toán học. Chẳng hạn với số mũ, có thể dùng dấu ^. Hãy thử viết `v(m^3)` để thu được V(m^3).

Matlab tự động căn chỉnh phạm vi của trục tung và trục hoành sao cho có thể hiển thị toàn bộ số liệu cần vẽ. Tuy vậy trong một số trường hợp ta cần thể hiện từng phần của biểu đồ, hoặc vì tính thẩm mỹ mà có thể chỉnh sửa phạm vi của các trục. Câu lệnh như sau:

```
axis([xmin xmax ymin ymax])
```

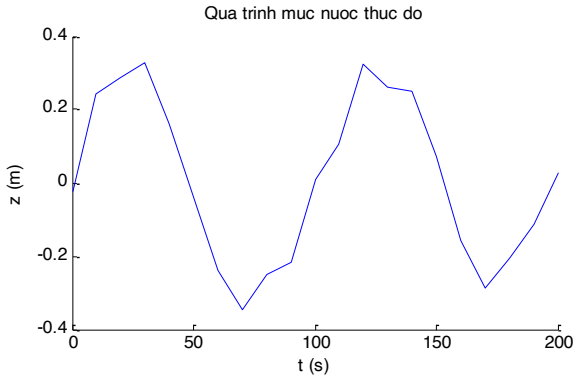
trong đó:

xmin, ***xmax*** lần lượt là giới hạn trái và phải của trục hoành

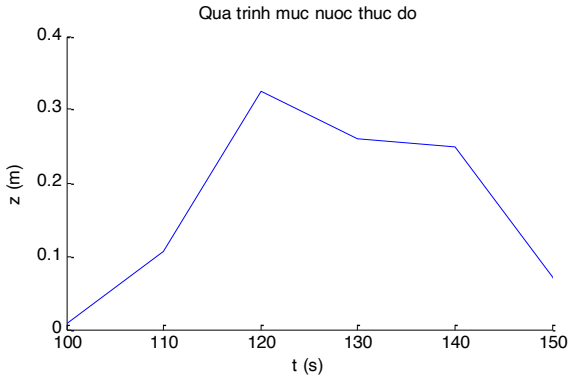
ymin, ***ymax*** lần lượt là giới hạn dưới và trên của trục tung.

Ta cần nhập cả 4 giá trị nói trên của vec-tơ phạm vi trục. Chẳng hạn khi muốn biểu thị mực nước chỉ trong khoảng thời gian từ t = 100 s đến t = 150 s, và hạn chế cao độ mặt nước từ 0 đến 0.4 m, ta gõ lệnh sau để thu được biểu đồ như Hình 6.2:

```
axis([100 150 0 0.4])
```



Hình 6.1: Ví dụ về biểu đồ dạng đường



Hình 6.2: Biểu đồ sau khi chỉnh lại phạm vi trục

Để xóa toàn bộ đồ thị hiện thời, ta gõ lệnh:

```
clf
```

6.2. Lựa chọn màu vẽ, nét vẽ

Trường hợp có nhiều đường nét vẽ khác nhau, ta nên phân biệt bằng những kiểu và màu nét vẽ khác nhau. Những lựa chọn này có ngay trong câu lệnh plot.

```
plot(x, y, 'lựa chọn')
```

Trong đó, ***lựa chọn*** là một chuỗi kí tự có 3 phần qui định như sau:

md--

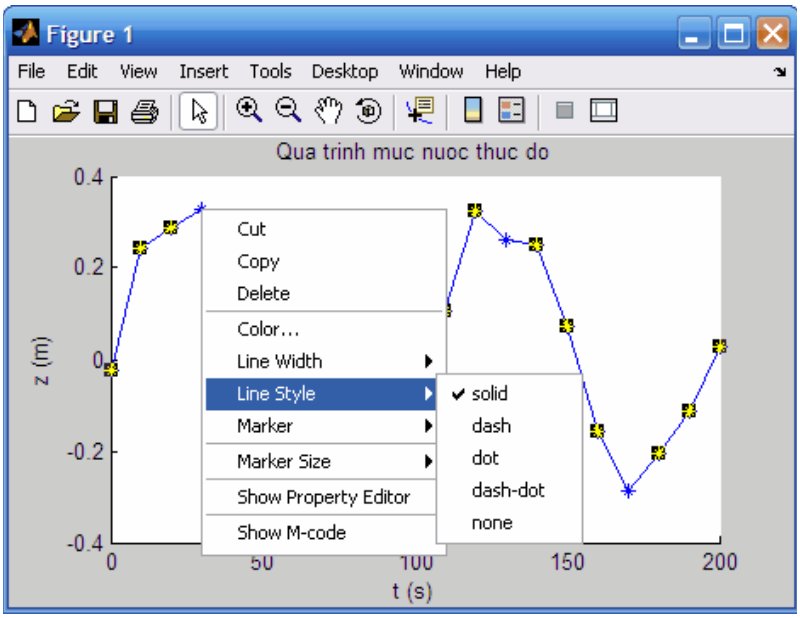
với ***m*** là một kí tự chỉ màu vẽ, thường là chữ đầu của từ tiếng Anh tương ứng
d là kí hiệu đánh dấu các điểm nút
-- là một hay hai kí tự thể hiện kiểu nét vẽ

Ví dụ:

`plot(x, y, 'r')` vẽ đồ thị nét vẽ màu đỏ (**r**ed). Không có lựa chọn kiểu nét vẽ cụ thể, do đó máy sẽ vẽ kiểu nét liền (mặc định).

`plot(x, y, 'g--')` vẽ đồ thị nét vẽ xanh lục (**g**reen). Nét vẽ được chọn là kiểu nét đứt (--).

`plot(x, y, 'b*-')` vẽ đồ thị nét vẽ xanh lam (**b**lue). Nét vẽ được chọn là kiểu nét liền với các điểm dấu sao (*-).



*Hình 6.3: Kiểu nét vẽ cùng các thuộc tính khác có thể lựa chọn trực tiếp sau khi chọn chế độ **Edit Plot** và nháy phải chuột vào đường biểu đồ*

Như vậy có thể tổ hợp các kiểu màu và nét vẽ khác nhau, theo bảng sau đây[2]:

Màu vẽ		*Nét vẽ*		*Điểm nút*	
r	red, đỏ	-	nét liền	*	dấu sao
g	green, xanh lục	--	nét đứt	+	dấu cộng
b	blue, xanh lam	:	nét chấm	s	hình vuông
k	black, đen	.-	nét chấm gạch	^	tam giác

Theo mặc định, khi có một đường mới được vẽ ra thì đường cũ sẽ biến mất. Để vẽ nhiều đường trên cùng một biểu đồ ta cần gõ lệnh:

```
hold on
```

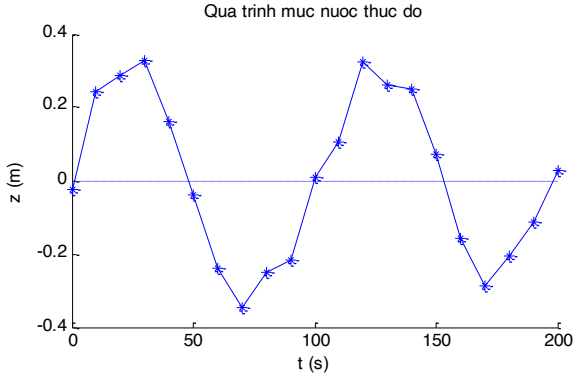
trước khi vẽ các đường tiếp theo.

Chẳng hạn ta có thể thêm một đường nét đứt nằm ngang biểu thị mực nước bằng 0 theo cách sau:

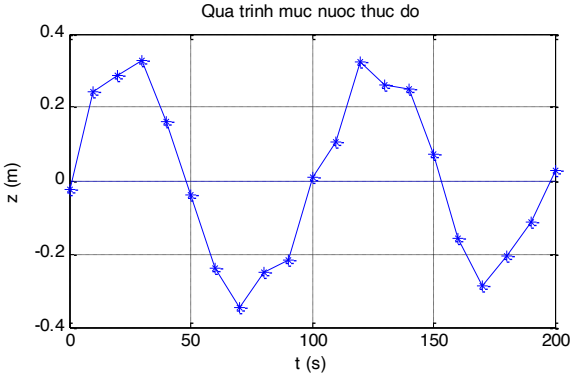
```
hold on; plot([0 200],[0 0],':');
```

Bản thân trên đường mực nước có thể thêm các điểm dấu *:

```
plot(t,z,'*-');
```



Hình 6.4: Biểu đồ dạng đường với hai nét vẽ có kiểu khác nhau



Hình 6.5: Biểu đồ có khung và đường dóng

Nếu muốn đóng khung đồ thị và tạo các đường dóng ta lần lượt gõ vào các lệnh:

2 Thực ra ta có thể lựa chọn kiểu và nét vẽ trực tiếp trên biểu đồ của Matlab bằng cách chọn Edit Plot (hoặc nút  trên thanh công cụ) rồi nháy phải chuột vào đường cần chỉnh và lựa chọn **Line Style** (kiểu đường), **Line Width** (bề rộng) hoặc **Marker** (kiểu điểm nút) (xem Hình 3).

```
box on
grid on
```

Ngược lại, để xóa các đường đóng khung và đường dóng, chỉ cần gõ:

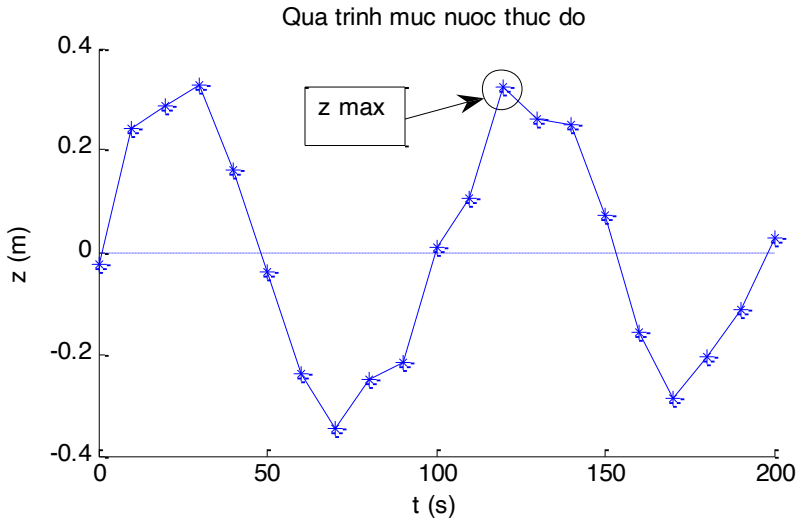
```
box off
grid off
```

6.3. Tạo các chú thích, chú giải trên hình vẽ

Một biểu đồ mặc dù mang nhiều thông tin nhưng đôi khi ta vẫn muốn làm rõ thêm bằng những chú thích (thường là những nét vẽ đơn giản bằng tay và chú thích bằng chữ). Chẳng hạn, với ví dụ trên ta muốn chỉ ra trên biểu đồ vị trí mực nước đạt cực đại (như Hình 6.6). Cách làm có thể theo các bước sau:

- Vào menu **Insert – Ellipse**, vẽ một vòng tròn vào vị trí đỉnh của đường quá trình Vào menu **Insert – Text Box**, vạch ra một khung chữ nhật và gõ vào **`z max`** vào đó.[3]
- Vào menu **Insert – Arrow**, vạch mũi tên chỉ từ khung chữ vào vòng tròn

Muốn xóa các chú thích, chỉ cần chọn đối tượng cần xóa, ấn **Delete**.



Hình 6.6: Biểu đồ với các chú thích

**** Chú giải (legend):***

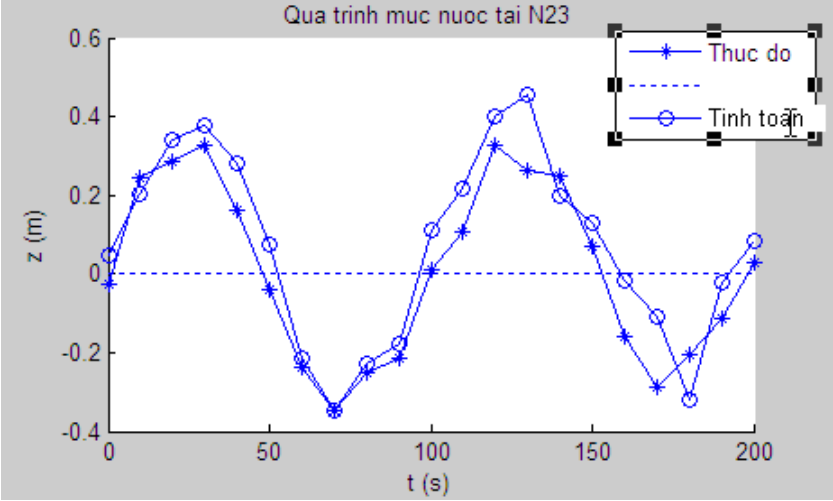
Khi có nhiều đường biểu đồ trên một đồ thị, cần có chú giải (legend) để phân biệt chúng. Giả sử cũng trên đồ thị quá trình mực nước ở trên, ta bổ sung thêm một đường mực nước tính toán được từ mô hình:

```
z2 =[0.05  0.2  0.34  0.38  0.28  0.08  -0.21  -0.35  -0.23  -0.18
0.11  0.22  0.4  0.45  0.2  0.13  -0.02  -0.11  -0.32  -0.02  0.09];
hold on; plot(t,z2,'-o');
```

3 Muốn có z_{max}, hãy thử gõ **`z_{max}`**. Gõ **`z_max`** có được không, tại sao?

```
title('Qua trinh muc nuoc tai N23');
```

Trên thanh công cụ, ấn nút (hoặc menu **Insert – Legend**). Sau đó hãy nháy đúp phần chú giải mới xuất hiện, gõ vào tên chú giải cần đặt (Hình 6.7).



Hình 6.7: Chỉnh sửa tên chú giải tương ứng với các đường quá trình

6.4. Xóa đường biểu đồ, lưu biểu đồ

- Nếu vẽ sai một đường biểu đồ nào đó ta có thể xóa bằng cách chọn menu **Tools – Edit Plot**; chọn đường biểu đồ đó, ấn **Delete** hoặc chọn **Delete** trong danh mục khi nháy phải chuột trên Hình 3.

- MatLab có thể lưu lại biểu đồ chúng ta tạo ra, dưới dạng nhiều dạng file ảnh chuẩn hiện nay: *.gif, *.png, *.jpeg, *.emf, *.eps v.v... Bên cạnh đó, Matlab còn có một dạng file riêng gọi là *.fig, trong đó lưu toàn bộ thông tin của các đường, nét, điểm... trên biểu đồ. Do vậy mà file fig chi tiết hơn đồng thời thường có kích thước file lớn.
- Để lưu biểu đồ trong Matlab ta chọn Menu **File – Save**, sau đó nhập tên cho file hình mà ta muốn lưu. Cũng có thể nhấn vào biểu tượng Save ( ) trên thanh công cụ. Chú ý chọn kiểu file hình (**Save as type**) thích hợp.

6.5. Đồ thị Logarit

Trong một số trường hợp, các đồ thị logarit cần được sử dụng, chẳng hạn biểu đồ đường cấp phối hạt. Muốn đặt thang logarit với trục hoành ta chỉ cần thay tên lệnh `plot` bằng `semilogx`.

Chẳng hạn ta cần vẽ đường cấp phối hạt với mẫu bùn cát sau:

Đường kính (mm)	Khối lượng (mg)
d < 0.15	900
0.15 < d < 0.21	2900
0.21 < d < 0.30	16000
0.30 < d < 0.42	20100
0.42 < d < 0.60	8900
0.60 < d	1200
(Toàn bộ)	50000

Trước khi vẽ đồ thị hãy tính tỉ lệ bùn cát tương ứng với mỗi khoảng đường kính và tỉ lệ cộng dồn:

```
KhoiLuong = [900  2900  16000  20100  8900  1200];
TiLe = KhoiLuong / 50000;
```

Hàm cumsum giúp ta tính cộng dồn, chẳng hạn:

```
>> cumsum(TiLe)
ans =
0.0180    0.0760    0.3960    0.7980    0.9760    1.0000
```

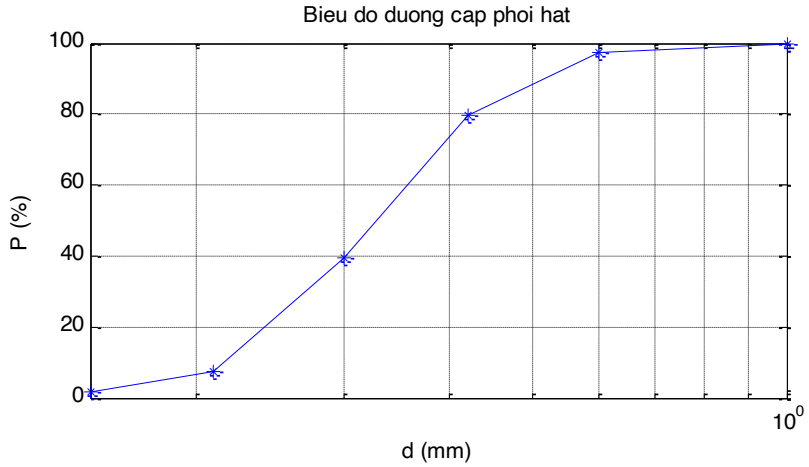
Ta cần tính tỉ lệ P theo phần trăm, cho nên:

```
P = cumsum(TiLe) * 100;
```

Đường kính (mm)	Khối lượng (mg)	Tỉ lệ (%)	Tỉ lệ cộng dồn, P (%)	Đường kính d (mm)
d < 0.15	900	1.8	1.8	0.15
0.15 < d < 0.21	2900	4.8	7.6	0.21
0.21 < d < 0.30	16000	32.0	39.6	0.30
0.30 < d < 0.42	20100	40.2	79.8	0.42
0.42 < d < 0.60	8900	17.8	97.6	0.60
0.60 < d	1200	2.4	100	1
(Toàn bộ)	50000	100		

Số liệu dùng để vẽ đồ thị là hai cột sau cùng: tỉ lệ cộng dồn (P) và đường kính (d). Có P (%) khối lượng bùn cát mịn hơn d (mm). Ở đây đã giả thiết rằng đường kính lớn nhất bằng 1 mm như là giới hạn trên của biểu đồ.

```
d = [0.15  0.21  0.30  0.42  0.60  1];
title('Bieu do duong cap phoi hat');
xlabel('d (mm)');
ylabel('P (%)');
semilogx(d,P,'*-');
grid on; box on;
```



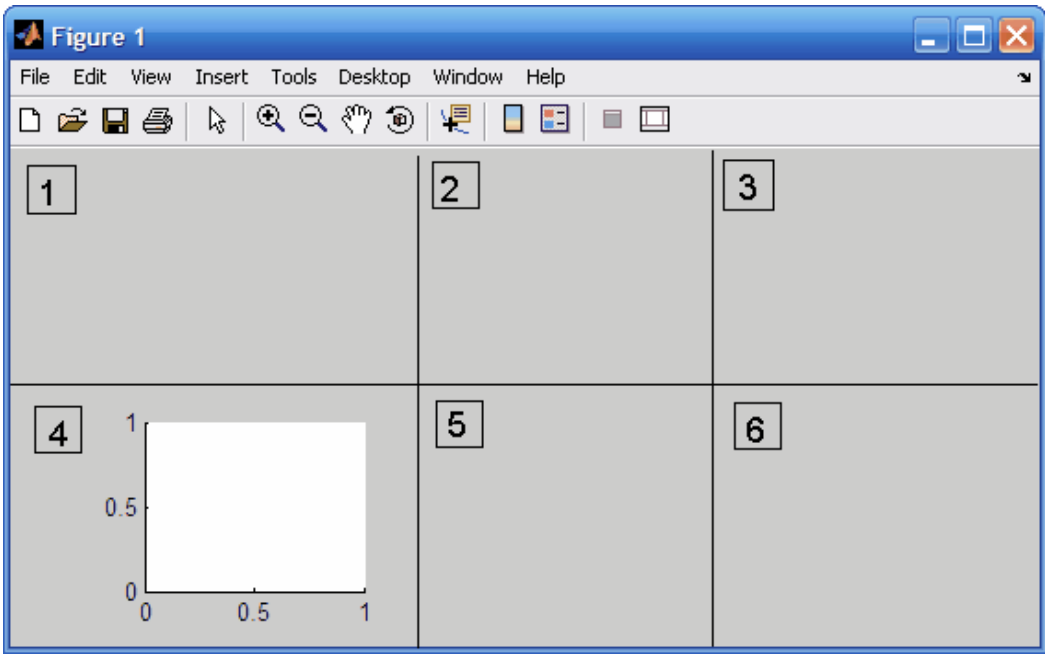
Hình 6.8: Ví dụ với biểu đồ có trục theo thang logarit

Các đồ thị với thang logarit trên trục *y* và trên cả 2 trục cũng được thực hiện tương tự với các câu lệnh lần lượt là **`semilogy`** và **`loglog`**.

6.6. Dãy biểu đồ

Bằng lệnh **`figure`** ta có thể tạo ra nhiều hình vẽ độc lập trên nhiều cửa sổ. Tuy vậy, nhiều lúc ta muốn có một dãy (hoặc bảng) các biểu đồ xếp kế tiếp nhau, có kích thước bằng nhau để tiện việc so sánh. Matlab hỗ trợ hệ thống subplot (biểu đồ nhỏ) với câu lệnh có dạng chung như sau:

```
subplot(m,n,k);
```



Hình 6.9: Vị trí các biểu đồ trong dãy tạo bằng lệnh subplot

Sẽ tạo ra một bảng gồm (***m*** × ***n***) biểu đồ nhỏ (***m*** hàng và ***n*** cột). Tiếp đó hình thứ ***k*** (tính từ trên xuống dưới, trái qua phải) sẽ được kích hoạt và chuẩn bị được vẽ.

Chẳng hạn, sau khi thực hiện lệnh

```
subplot(2,3,4);
```

ta được kết quả như Hình 6.9.

Gọn hơn nữa ta có thể viết (trong trường hợp ***m***, ***n***, ***k*** < 10):

```
subplot 234
```

Xét một ví dụ đơn giản: ta cần vẽ đường quá trình mực nước (z) và vận tốc dòng chảy (v) theo thời gian (t), nhưng trên hai biểu đồ khác nhau. Để có sự đối chiếu về thời gian giữa hai biểu đồ ta nên xếp chúng theo một cột dọc. Như vậy ***m*** = 2 và ***n*** = 1.

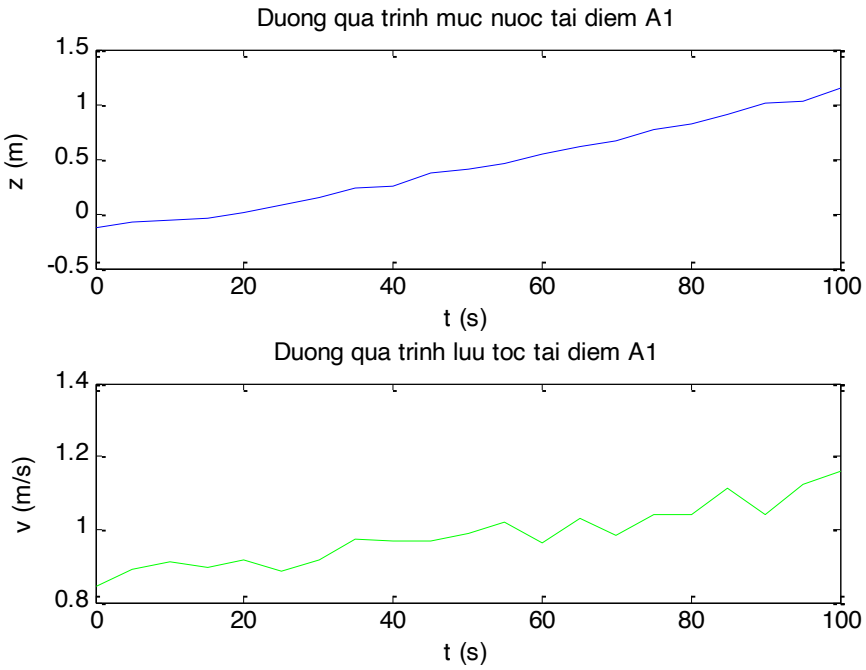
```
t = 0:5:100;
z = [-0.14  -0.08  -0.05  -0.04  0.01  0.07  0.15  0.23  0.25  0.37  0.4
      0.45  0.55  0.6  0.66  0.76  0.82  0.91  1  1.03  1.14];
v = [0.84   0.89  0.91  0.89  0.91  0.88  0.92  0.97  0.97  0.97  0.99
      1.02  0.96  1.03  0.98  1.04  1.04  1.11  1.04  1.12  1.16];

subplot 211
plot(t, z); xlabel('t (phut)'); ylabel('z (m)');
title('Duong qua trinh muc nuoc tai diem A1');

subplot 212
plot(t, v, 'g'); xlabel('t (phut)'); ylabel('v (m/s)');
title('Duong qua trinh luu toc tai diem A1');
```

Kết quả như trên Hình 6.10.

Matlab còn có lệnh `plotyy` cho phép hai trục tung với hai đại lượng khác nhau (chẳng hạn `z` và `v`) trên cùng một biểu đồ (và tô màu trục theo màu đường vẽ từng đại lượng đó). Đây là cách biểu thị thông tin linh hoạt và rất hiệu quả.



Hình 6.10: Hai đường quá trình trên hai subplot

7. CHƯƠNG VII: ĐỒ THỊ KHÔNG GIAN

Trong kỹ thuật rất cần các biểu đồ không gian. Thường gặp nhất là các bình đồ (bản đồ địa hình của một khu vực nhỏ). Bên cạnh đó, biểu đồ không gian còn có thể được dùng để biểu diễn độ sâu địa hình hay một trường không gian nói chung (nhiệt độ, khí áp ...). Trong tất cả các trường hợp nói trên, luôn có một biến (z) được biểu diễn theo hai biến không gian trên mặt đất (x, y).

7.1. Các dạng cơ bản

Các dạng cơ bản

Matlab hỗ trợ nhiều phương pháp biểu diễn số liệu; trong tài liệu này ta làm quen với một số loại cơ bản, đó là:

- Mảng màu (color patch)
- Đường đồng mức (contour)
- Bề mặt 3 chiều (surface)

Hãy bắt đầu với một ví dụ đơn giản là mô phỏng địa hình đáy biển của một khu vực giả tưởng được cho trong ma trận z. Đáy biển này có dạng tương tự như mặt cắt ngang cân bằng (Dean) theo phương trình:

$$z = -0.1\, y^{2/3}$$

Với trục x dọc theo bãi biển và trục y hướng ra khơi. Ta xét lưới tọa độ trên mặt bằng, các điểm nút lưới có phạm vi $0 < x < 200$, $0 < y < 200$. Khoảng cách giữa các điểm nút lưới theo phương x là $\Delta x = 10$, theo phương y là $\Delta y = 4$. Nghĩa là tọa độ của tất cả các điểm nút lưới có dạng:

$$x_i, y_i = \begin{bmatrix} 0{,}0 & 10{,}0 & \dots & 200{,}0 \\ 0{,}4 & 10{,}4 & \dots & 200{,}4 \\ \vdots & \vdots & \ddots & \vdots \\ 0{,}200 & 10{,}200 & \dots & 200{,}200 \end{bmatrix}$$

Lệnh meshgrid sẽ giúp ta làm điều đó:

```
[x, y] = meshgrid(0:10:200, 0:4:200);
```

(Một lần nữa cần chú ý dấu ; ở cuối câu lệnh, nếu không rất nhiều con số sẽ được in ra màn hình!)

Lệnh **meshgrid** lấy hai thông số: thông số vec-tơ thứ nhất là các tọa độ trên trục x, thông số vec-tơ thứ hai là các tọa độ trên trục y. Kết quả ta được một ma trận các

điểm trên một vùng hình chữ nhật (51 × 21) có các tọa độ tương ứng dựng từ các vec-tơ trên[4]. Sau đó mảng z sẽ được tính từ mảng y theo công thức Dean:

```
z = -0.1 * y .^ (2/3);
```

Trong đó cần chú ý phép tính lũy thừa cho từng phần tử.

Câu lệnh đơn giản nhất để hiển thị mảng màu biểu thị *z* là:

```
pcolor(x, y, z)
```

Bảng màu mặc định được dùng có tên là jet.[5] Ta có thể thay bảng màu này bằng một số bảng màu khác, chẳng hạn:

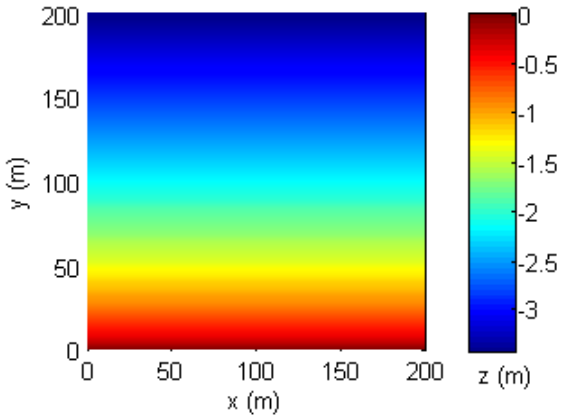
```
colormap(gray); pcolor(x, y, z);
```

Cần hiện tên các trục *x* và *y* theo cách tương tự như ta đã làm với biểu đồ dạng đường:

```
xlabel('x(m)'); ylabel('y(m)');
```

Có thể hiển thị thanh chú thích bên cạnh bảng màu bằng lệnh:

```
colorbar
```



Hình 7.11: Bảng màu biểu thị địa hình đáy biển với mặt cắt ngang cân bằng theo Dean

Viết thêm dòng chữ z(m) bên cạnh thanh chỉ dẫn màu này bằng cách **Insert – Text Box**. Xóa bỏ đường bao của hình chữ nhật bằng cách nháy phải chuột – **Line Style – None**.

Chú ý rằng khi hiện các bản đồ địa hình / bản vẽ mặt bằng, trong hầu hết các trường hợp ta muốn đặt những tỷ lệ bằng nhau trên hai trục *x* và *y*. Khi đó ta dùng lệnh:

```
axis equal;
```

---

4 x và y đều là ma trận 2 chiều. Do các điểm trên lưới xếp theo hàng nên ta nhận thấy các phần tử giống nhau trên mỗi cột của x (các điểm có cùng tọa độ x); và tương tự đối với y.

$$x=\begin{bmatrix} 0 & 10 & \cdots & 200 \\ 0 & 10 & \cdots & 200 \\ \vdots & \vdots & & \vdots \\ 0 & 10 & \cdots & 200 \end{bmatrix} \text{ và } y=\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 4 & 4 & \cdots & 4 \\ \vdots & \vdots & & \vdots \\ 200 & 200 & \cdots & 200 \end{bmatrix}$$

5 Bảng màu `jet` này đã được thay bằng bảng màu parula mới làm mặc định trong Matlab. Bảng parula không chứa màu đỏ nên nhìn đỡ chói hơn. Việc chọn bảng màu phù hợp rất quan trọng để biểu diễn các đại lượng khác nhau!

Tiếp theo ta có khung của đồ thị lại cho vừa với bảng màu:[6]

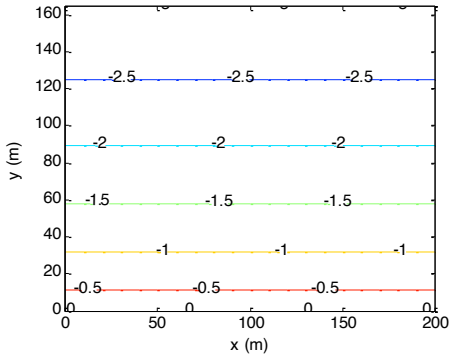
```
axis tight;
```

Bên cạnh cách dùng biểu đồ mảng màu, còn có thể biểu thị dưới dạng đường đồng mức. Cách này thường dùng với các bản in lên giấy:[7]

```
[C,h] = contour(x,y,z);
```

Các đường đồng mức của đáy biển đã xuất hiện song song với trục *x*, nhưng cần phải có giá trị số trên đường đồng mức:

```
clabel(C,h);
```



Hình 7.12: Đường đồng mức biểu diễn địa hình đáy biển với mặt cắt theo Dean

Và cũng như đối với mảng màu, ta có thể căn chỉnh các trục một cách hợp lý:

```
axis equal tight;
xlabel('x(m)'); ylabel('y(m)');
```

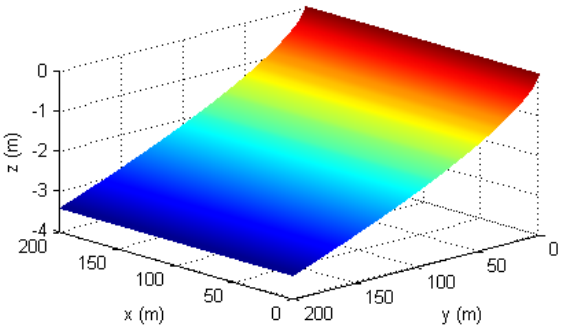
Ta cũng có thể biểu diễn địa hình dưới dạng không gian (3 chiều) sử dụng lệnh surf hoặc mesh:

```
figure; surf(x, y, z);
figure; mesh(x, y, z);
```

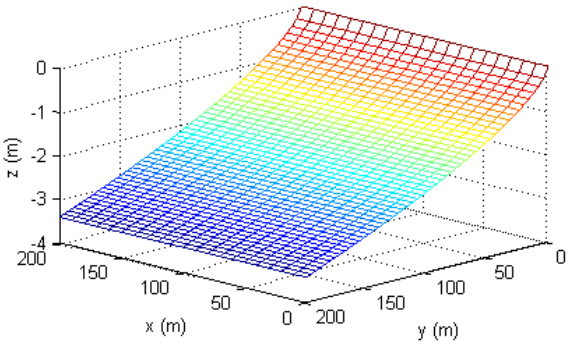
Sau đó hãy sử dụng công cụ xoay hình bằng cách ấn nút **Rotate 3D** trên thanh công cụ ().

6 Hai lệnh trên có thể được gộp lại thành một lệnh **`axis equal tight`**

7 Cách viết này có vẻ khác với câu lệnh thông thường, tuy nhiên theo Matlab, các lệnh vẽ đều có thể viết dưới dạng hàm. Chẳng hạn, hãy gõ thử lệnh **`zzz = plot(t, z).`** Mặt khác, có gì khác nếu ta gõ lệnh **`clabel(contour(x,y,z));`** ?



Hình7. 13: Biểu diễn mặt cắt Dean dưới dạng mặt 3 chiều



Hình 7.14: Biểu diễn mặt cắt Dean dưới dạng lưới 3 chiều

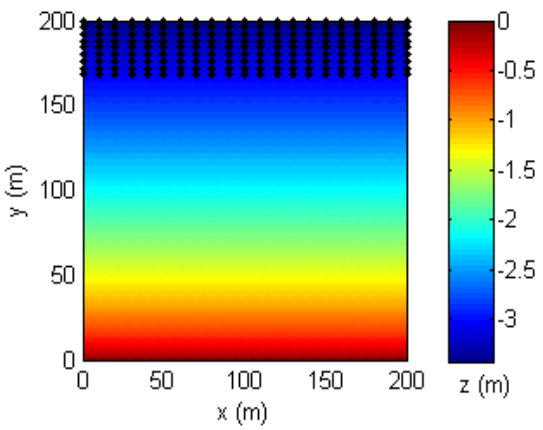
7.2. Chỉ định các vị trí trong không gian 2 chiều

Khi thể hiện dữ liệu không gian bằng mảng màu, trong một số trường hợp ta phải chỉ ra những vị trí thỏa mãn một điều kiện nào đó. Cách đơn giản nhất là đánh dấu chúng bằng những điểm riêng.

Chẳng hạn, hãy chỉ ra các vị trí có $z < -3$ trong ví dụ trên. Có hai bước như sau:

- Tìm các chỉ số (i,j) trong ma trận có giá trị thỏa mãn $z(i,j) < -3$
- Chấm các điểm xác định bởi $x(i,j)$, $y(i,j)$ lên mảng màu

```
ind = find(z < -3);
hold on;
plot(x(ind), y(ind), ... 'k.');
```



Hình 7.15: Đánh dấu trên mảng màu 2 chiều

7.3. Mặt cắt địa hình

Trong nhiều trường hợp cần biểu diễn một mặt cắt địa hình từ biểu đồ hai chiều. Và tổng quát hơn, có thể là phân bố độ sâu nước, áp suất, độ mặn, hay bất kì một biến đặc trưng nào dọc theo một tuyến xác định trên mặt bằng.

Số liệu của mặt cắt được lấy từ 1 cột (hoặc hàng) từ mảng 2 chiều tương ứng. Chẳng hạn, mặt cắt có $x = 100$ m (tương ứng với cột thứ 11 (trong số 21 cột) trong ma trận z):

```
ymc1 = y(11,:);
zmc1 = z(11,:);
plot(ymc1, zmc1, '*-');
xlabel('y (m)'); ylabel('z (m)');
```

7.4. Trường véctơ

Trong các bài toán kỹ thuật thủy động lực thường yêu câu mô phỏng trường dòng chảy. Biểu diễn dòng chảy trong không gian 2 chiều được hỗ trợ rất tốt trong Matlab bằng câu lệnh:

```
quiver(x, y, u, v)
```

trong đó ***x***, ***y*** là hai ma trận chứa tọa độ của tất cả các điểm theo phương ngang (thường được phát sinh bằng lệnh meshgrid); ***u*** và ***v*** là hai ma trận chứa thành phần lưu tốc của tất cả các điểm (***u*** là lưu tốc theo phương ***x***, ***v*** là lưu tốc theo phương ***y***).

Giả sử trong thư mục hiện thời đã có file `data_u.txt` và `data_v.txt`, mỗi file chứa một ma trận giá trị `u` và `v`. Khi đó, hai ma trận này được đọc như sau:

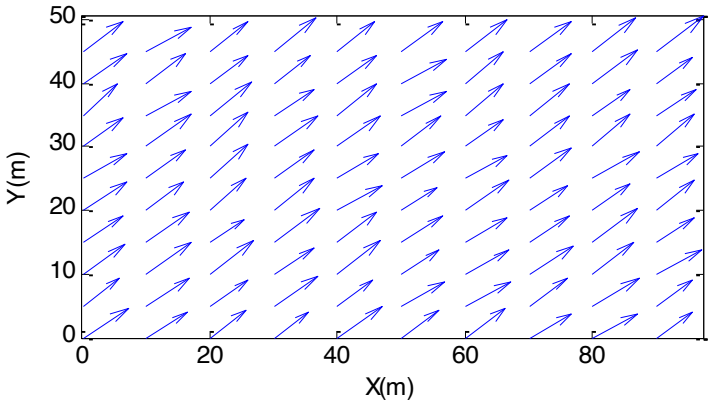
```
u = dlmread('data_u.txt');
v = dlmread('data_v.txt');
```

Giả sử `u` và `v` đã đọc có kích thước giống nhau = số điểm trên trục x × số điểm trên trục y:

```
[sizex, sizey] = size(u);
```

Ta muốn biểu diễn lên hình với, chẳng hạn, Δx = 10 m và Δy = 5 m:

```
dx = 10; dy = 5;
[x, y] = meshgrid(0:dx:(sizex – 1)*dx, 0:dy:(sizey -1) * dy);
figure;
quiver(x, y, u, v);
xlabel('X(m)'); ylabel('Y(m)');
axis equal tight;
```



Hình 16: Biểu diễn trường vec-tơ

8. PHẦN BÀI TẬP

Bài tập số 1:

Tính toán sóng theo số liệu sau: $T = 8$ s, $H_0 = 2$ m, $\alpha_0 = 30°$.

• Gán giá trị cho 3 biến T, Ho, alpha0.
• Tính C_o.
• Tính L_o.
• Mở 1 file **disperse.m** bằng cách gõ

```
>> edit disperse
```

• Với file mới mở hãy gõ vào nội dung sau:

```
function [L] = disperse(h, Lo)
L = Lo;
err = Inf;
while err > 0.1;
Ltry = Lo * tanh(2*pi*h / L);
err = abs(Ltry - L);
L = Ltry;
end;
```

• Ta đã thiết lập được hàm tính L từ các giá trị h và Lo cho trước
• Hãy thiết lập một vectơ h chứa độ sâu khác nhau: 3 m, 3.2 m, ..., 4 m. Tính L tại các độ sâu cho bởi h.
• Tính C tương ứng với L
• Tính **sin_alpha** theo công thức $\sin\alpha = \sin\alpha_0 * C / Co$. (chú ý đơn vị độ !)
• Tính alpha (dùng lệnh asin)
• Tính Kr = $\text{sqrt}(\cos\alpha_0 / \cos\alpha)$
• Tính Ksh theo công thức Ksh =
• Tính H = Ho × Ksh × Kr
• Ghi lại các giá trị của H
•Viết kết quả ra file **ketqua.txt**

Bài tập số 2

Quan hệ giữa vận chuyển bùn cát S và lưu tốc u có quan hệ dạng: $S = au^b$. Để xác định các hệ số a và b người ta tiến hành thí nghiệm và thu được kết quả sau:

u (m/s)	0.2	0.35	0.57	0.68	0.81	0.96	1.12
S (m^3/s.m)	0.0002	0.0018	0.0159	0.0282	0.0609	0.1258	0.2858

1) Hãy nhập các giá trị *u* và *S* vào hai vectơ, sau đó tính X và Y là logarit tương ứng của *u* và *S*.
2) Vẽ đồ thị điểm của X và Y.
3) Hai hệ số *a* và *b* được xác định bằng cách dựa vào quan hệ: $\log S = \log a + b \log u$, hay $Y = b\,X + \log a$.

Sử dụng hàm polyval để tính *b* và log *a* trong phương trình trên như sau:

```
p = polyval(X, Y, 1)
```

Phần tử đầu của vectơ p chính là *b*; phần tử thứ 2 của p bằng log *a*.
Hãy tính *a*.

4) Vẽ đồ thị đường thẳng: $Y = b\,X + \log a$ lên cùng hệ trục với các điểm chấm ở câu 2).

Bài tập số 3

1) Một chuỗi số liệu đo đạc vận tốc dòng chảy được phát sinh bởi:

```
t = 0:0.5:48;
```

N là dãy số ngẫu nhiên có chiều dài bằng `length(t)`

```
u = 0.4 + 0.12 sin(2π t / 24) + 0.05 N
```

2) Tính ứng suất tiếp tại ven bờ, biết
$C = 60\ m^{0.5}/s$
$\tau = \rho\, g\, u^2 / C^2$

Thiết lập subplot 3 hàng × 1 cột. Hai vùng trên vẽ hai đồ thị $u \sim t$, $\tau \sim t$.

3) Nếu bờ có thành phần đất sét với ứng suất tiếp tới hạn $\tau_c = 0.65\ N/m^2$, hãy chỉ ra xem có bao nhiêu thời điểm xuất hiện $\tau > \tau_c$.

4) Công thức xói lở đường bờ được xác định bởi:

Hãy tính khoảng cách xói lở E.

$E = 10\,(\tau - \tau_c)$ nếu $\tau > \tau_c$
$E = 0$ trường hợp còn lại

Vẽ $E \sim t$ lên vùng đồ thị dưới cùng tạo bởi `subplot`.

Bài tập số 4

Cho véctơ `X = [10 20 30]` và `Y = [10 40 30]`.

1. Tính diện tích tam giác tạo bởi 3 đỉnh có tọa độ X(1)Y(1), X(2)Y(2), X(3)Y(3).
2. Tổng quát hơn, tính diện tích đa giác n cạnh (trên mặt phẳng 2 chiều) với tọa độ xi và yi của đỉnh thứ i cho bởi phần tử thứ i của véctơ X và Y tương ứng (do đó X và Y là 2 véctơ có cùng chiều dài = n).

LỜI GIẢI

Bài tập số 1

Hướng dẫn: tạo một script file có tên *baitap1.m*, soạn thảo nội dung dưới đây, và ghi lại. Sau đó chạy file này, ta sẽ thu được kết quả.

```
% baitap1
% Inputs:   T,  H0, alpha0
% Outputs:  C0, L0, L
%-------------------------------------------------------------
clear all;

g = 9.81;
T = 8;
H0 = 2;
alpha0 = 30;
%-------------------------------------------------------------

L0 = g*T^2/(2*pi)
C0 = L0/T;
h = [3:0.2:4]

% n = length(h);
% for i = 1:n
%     L(i) = disperse(h(i), L0);
% end;

L = disperse(h,L0);                 % Wave length

C = L/T                             % Wave celerity
sin_alpha=sin(alpha0)/C0*C
alpha = asin(sin_alpha)             % arcsin
Kr = sqrt(cos(alpha0)./cos(alpha))  % Refraction coefficient
k = 2*pi./L                         % Wave number
Ksh = sqrt(1./tanh(k.*h)/(1+2*k.*h/sinh(2*k.*h))) % Shoaling coeff
H = H0*Ksh.*Kr                      % Wave height
```

Bài tập số 2

Hướng dẫn: tạo một script file có tên *baitap2.m*, soạn thảo nội dung dưới đây, và ghi lại. Sau đó chạy file này, ta sẽ thu được kết quả.

(Bài này làm theo các bước như trên lớp: tính log(u), log(S), vẽ lên trục tọa độ thường (không vẽ lên trục loga vì ở đây ta đã tính giá trị của loga rồi), và xác định các hệ số a, b của đường thẳng một cách gần đúng trên đồ thị).

Bài tập số 3

Hướng dẫn: tạo một script file có tên *baitap3.m*, soạn thảo nội dung dưới đây, và ghi lại. Sau đó chạy file này, ta sẽ thu được kết quả.

```
clear all;
t = 0:0.5:48
N = rand(1,length(t));
u = 0.4+0.12*sin(2*pi*t/24)+0.05*N
plot(N);
hold on;
plot(u,'+');
C = 60;
Rho = 1000;
g = 9.81;
To = Rho*g/C^2*u.^2
plot(t,To)
hold on
grid on
Toc = 0.65
10 * (To - Toc) .* (To > Toc) + 0 .* (To <= Toc)
```

Bài tập số 4

Hướng dẫn: Lần lượt thực hiện các bước sau:
1. Tạo một script file có tên *baitap4.m*, soạn thảo nội dung dưới đây, và ghi lại.

```
function Sdagiac = dientich(x,y);
% Tinh dien tich da giac

n = length (x);
m = length (y);
if m ~= n
    'Error: 2 vecto X va Y can co chieu dai bang nhau!'
else
    Sdagiac = 0;
    for i = 1:(n-1)
        Sdagiac = Sdagiac + 0.5*(y(i)+y(i+1))*(x(i+1)-x(i));
    end
```

```
    Sdagiac = Sdagiac + 0.5*(y(n)+y(1))*(x(1)-x(n));
end
```

2. Vừa rồi ta đã tạo ra một hàm mới trong Matlab, với tên là hàm *dientich*. Sử dụng hàm này ta có thể dễ dàng tính được diện tích đa giác nói chung và tam giác nói riêng bằng cách gọi hàm từ cửa sổ nhập lệnh

```
>> dientich(X,Y)    ↵
```

Lưu ý rằng hàm tính diện tích ngầm định rằng 2 véc tơ X, Y ở đây đã chứa số liệu cho trước về tọa độ các đỉnh của đa giác theo như quy ước. Hơn nữa, khi nhập tọa độ ta đánh số các đỉnh từ 1 đến n theo chiều kim đồng hồ. (Nếu thứ tự các điểm nhập vào thuận chiều kim đồng hồ thì sao? Hãy thử tính ra kết quả xem!)

* Để tính diện tích tam giác S với chiều dài 3 cạnh cho trước là a, b, c, bạn có thể thử thuật toán khác như sau:

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

với $p=(a+b+c)/2$ là một nửa chu vi.

9. TÀI LIỆU THAM KHẢO:

1. Matlab R14 - Helps & Demos, The Mathworks, Inc., 2004
2. David F. Griffiths, *An introduction to Matlab*, Department of Mathematics-The University of Dundee – Scotland. (Added materials by Ulf Carlsson KTH Stockholm Sweden), 1997-2005.
3. John M. Stockie, *A Whirlwind Tour of MATLAB for Students of CS 3113*, Department of Mathematics and Statistics, University of New Brunswick – Canada, 2003
4. Bill Mason, *Introduction to Matlab*, Northeastern University - College of Computer and Information Science – USA, 2003.
5. Tobin A. Driscoll, *Crash course in MATLAB*, Department of Mathematical Sciences - University of Delaware – USA, 2006
6. Nguyễn Hoàng Hải & Nguyễn Việt Anh, *Lập trình Matlab và ứng dụng*, NXB Khoa học và Kỹ thuật – Hà Nội, 2005
7. Nguyễn Phùng Quang, *Matlab & Simulink dành cho kỹ sư điều khiển tự động*, NXB Khoa học & Kỹ thuật – Hà Nội, 2006
8. Nguyễn Phương Thảo, *Programming in Matlab*, handouts, 2007
9. ĐH. Thủy Lợi, *Matlab version 7.0 cơ bản, Khóa tin học nâng cao cho cán bộ giảng dạy của dự án 95 bộ NN&PTNT*, handouts, 2007
10. Knight A., *Basics of MatLab® and beyond*, CRC Press, 2000
11. Timothy A. Davis & Kermit Sigmon, *Matlab® Primer*, Chapman & Hall/CRC
12. R. J. Braun, *Beginning Matlab Exercises*, Department of Mathematical Sciences – University of Delaware – USA.