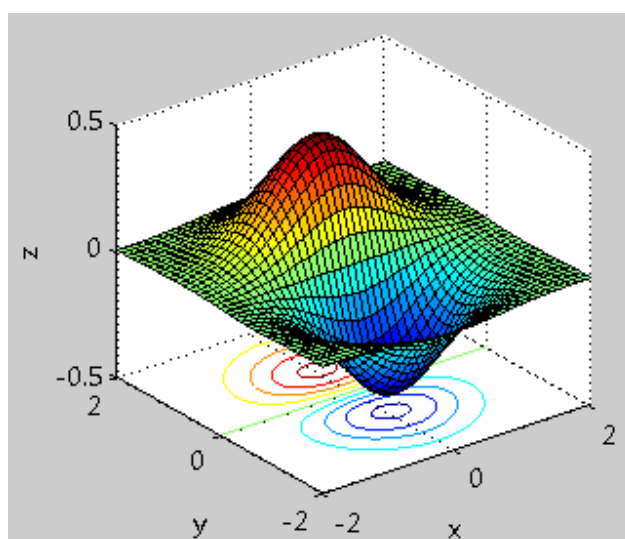


TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CƠ KHÍ – BỘ MÔN CƠ HỌC ỨNG DỤNG

TS. Nguyễn Quang Hoàng

Cơ sở

MATLAB và SIMULINK



Hà Nội 2010

Lời nói đầu

Hệ chương trình MATLAB là một công cụ xử lý số các hệ kỹ thuật từ đơn giản đến phức tạp. Chương trình này phù hợp với việc phân tích và tổng hợp nhanh các quá trình động lực đặc biệt trong nghiên cứu và phát triển, ngày nay Matlab đang được sử dụng nhiều trong công nghiệp. Matlab ngày càng có vai trò trong các trường đại học và cao đẳng kỹ thuật. Matlab có thể trợ giúp đắc lực các sinh viên và kỹ sư trong việc giải quyết các vấn đề tính toán số các bài toán kỹ thuật. Đặc biệt đối với sinh viên và kỹ sư ngành Cơ khí, điện, điện tử, cơ điện tử Matlab là một công cụ không thể thiếu.

Mục đích của cuốn sách là cung cấp cơ sở Matlab và Simulink cho các sinh viên kỹ thuật từ năm thứ hai sau khi đã có các kiến thức cơ bản về toán, vật lý, cơ học kỹ thuật cũng như kỹ thuật lập trình. Ngoài ra nếu có được thêm các kiến thức về kỹ thuật điều khiển, xử lý dữ liệu số người đọc có thể mở rộng thêm được các ứng dụng của Matlab và Simulink.

Nội dung của cuốn sách này được phân bố trong chín chương. Các chương từ một đến bảy trình bày việc sử dụng các lệnh của Matlab cho các bài toán cơ bản như tính toán trên vectơ, ma trận, đồ họa, giải phương trình vi phân thường, và một số phép biến đổi tích phân như Fourier, Laplace. Chương 8 giới thiệu về phần Simulink – một công cụ sử dụng các khối hàm để mô phỏng hệ. Trong mỗi chương, sau phần giới thiệu cách sử dụng các lệnh của Matlab đều có những ví dụ cụ thể và phần bài tập thực hành để người học có thể tự thực hành. Trong chương 9 một số bài toán kỹ thuật thường gặp trong lĩnh vực cơ học và kỹ thuật được trình bày.

Cuốn sách này được biên soạn trên cơ sở bài giảng của tác giả cho sinh viên ngành Cơ điện tử Trường Đại học Kinh doanh và Công nghệ Hà nội. Tuy nhiên, cuốn sách này không chỉ là tài liệu học tập cho sinh viên các trường đại học định hướng ứng dụng mà còn là tài liệu học tập cho sinh viên các trường đại học và cao đẳng kỹ thuật. Cuốn sách cũng là tài liệu bổ ích cho các kỹ sư trong công việc chuyên môn của họ.

Trong quá trình biên soạn chúng tôi đã nhận được sự trợ giúp quý báu của nhiều đồng nghiệp. Chúng tôi xin cảm ơn GS.TSKH. Nguyễn Văn Khang và PGS.TS. Nguyễn Phong Điền đã xem giúp bản thảo và có nhiều đề nghị cải tiến quý báu.

Mặc dù đã cố gắng nhiều, nhưng chắc chắn không tránh khỏi các sai sót, tác giả mong muốn nhận được sự góp ý của các bạn đồng nghiệp và của các em sinh viên để có điều kiện sửa chữa, hoàn thiện hơn trong các lần tái bản sau.

Mọi ý kiến đóng góp xin gửi về địa chỉ của tác giả:

TS. Nguyễn Quang Hoàng, Bộ môn Cơ học ứng dụng,

Trường Đại học Bách khoa Hà Nội.

E-mail: hoangnq-dam@mail.hut.edu.vn, hoặc Tel. 04.38680469.

Hà nội, tháng 10 năm 2010

Tác giả

Mục lục

Lời nói đầu	iii
Chương 1. Môi trường Matlab	
1.1 Matlab là gì?	1
1.2 Giao diện người sử dụng	2
1.3 Các phép tính số học cơ bản	3
1.4 Toán tử gán	4
1.5 Các định nghĩa toán học cơ bản	8
1.6 Số phức	11
1.7 Xử lý lỗi khi gõ lệnh	12
1.8 Kết thúc phiên làm việc với Matlab	12
1.9 Bài tập thực hành	13
Chương 2. Véc tơ, ma trận và Matlab	
2.1 Véc tơ và các phép tính trên véc tơ	15
Nhập véc tơ	15
Các phép tính cộng và trừ hai véc tơ cùng cỡ	16
Tạo véc tơ cỡ lớn từ các biến sẵn có	17
Tạo véc tơ có các phần tử cách đều	18
Các đặc trưng của véc tơ	19
Tích vô hướng và tích có hướng hai véc tơ	21
Tham chiếu đến các phần tử véc tơ	22
Một số hàm định sẵn cho véc tơ	23
2.2 Biểu diễn một đa thức bằng véc tơ	23
Nhập đa thức	23
Các phép tính trên đa thức	24
Phép nhân đa thức	24
Phép chia đa thức	25
Phép cộng và trừ đa thức	25
Không điểm hay nghiệm của phương trình đa thức	26
Xây dựng đa thức từ các không điểm cho trước	26
Giá trị của đa thức tại một điểm	26
Đạo hàm đa thức	27
2.3 Ma trận và các phép tính cơ bản trên ma trận	28
Nhập ma trận	28

Nhân ma trận với một số, phép cộng và trừ hai ma trận	28
Chuyển vị ma trận	29
Phép nhân ma trận	29
Một số phép tính cơ bản khác	30
Các ma trận đặc biệt	31
Tham chiếu đến các phần tử của ma trận	33
Ma trận khối	35
Phép nhân mảng hai ma trận cùng cỡ	36
Tính định thức và giải hệ phương trình đại số tuyến tính	37
Tìm hạng của ma trận	39
Tìm ma trận nghịch đảo và tựa nghịch đảo	41
Trị riêng và vectơ riêng của ma trận vuông	46
Phân tích ma trận vuông A thành tích các ma trận	47
2.4 Bài tập thực hành	51
 Chương 3. Lập trình trong Matlab	
3.1 Các kiểu dữ liệu	53
Kiểu vectơ và ma trận	53
Chuỗi ký tự (ký tự, xâu - string)	54
Kiểu ô, mảng (Cell-Array)	54
Kiểu cấu trúc	54
3.2 Soạn thảo Script file trong Matlab	56
Scripts	57
Các hàm, MATLAB - Function	60
3.3 Các vòng lặp và rẽ nhánh	62
Vòng lặp FOR	62
Vòng lặp WHILE	63
Lệnh if, cấu trúc if - else - end	65
Cấu trúc switch-case	66
3.4 Các Mat-File	71
3.5 Bài tập thực hành	71
 Chương 4. Đồ họa trong Matlab	
4.1 Đồ họa 2D	73
Đặt màu và kiểu đường cho đồ thị	76
Một số tùy chọn khi vẽ đồ thị 2D	77
Vẽ nhiều đồ thị trên cùng một hệ trục	79

Thêm các chú thích	81
Các lệnh axis	82
Đặt giới hạn miền vẽ với lệnh axis	83
Lệnh Subplots	84
Vẽ các đồ thị xếp chồng và lệnh linspace	87
Đồ thị theo tọa độ cực và Đồ thị với thang chia Logarith	89
Đồ thị của dữ liệu rời rạc	92
Vẽ biểu đồ với lệnh contour – vẽ đường đồng mức	96
Thêm chú thích trên đồ thị	101
Vẽ đồ thị các hàm có điểm không xác định	101
4.2 Các lệnh vẽ trong không gian – 3D	103
4.3 Bài tập thực hành	113
 Chương 5. Mìn hóa đường cong	
5.1 Mìn hóa bằng đa thức	115
5.2 Mìn hóa bằng hàm e mũ	120
5.3 Nội suy đa thức	122
5.3 Bài tập thực hành	125
 Chương 6. Phương trình vi phân thường và Matlab	
6.1 Giải phương trình vi phân bậc nhất với ode23 và ode45	127
6.2 Giải các phương trình vi phân bậc hai	133
6.3 Bài tập thực hành	139
 Chương 7. Các phép biến đổi tích phân và Matlab	
7.1 Phép biến đổi Laplace	141
7.2 Phép biến đổi Laplace ngược	143
7.3 Áp dụng Laplace giải phương trình vi phân	145
7.4 Phép biến đổi Fourier	148
7.5 Phép biến đổi Fourier ngược	151
7.6 Phép biến đổi Fourier một tín hiệu rời rạc	151
7.7 Phép biến đổi Fourier nhanh (FFT)	152
7.8 Bài tập thực hành	154
 Chương 8. Giới thiệu về Simulink	
8.1 Khái niệm về simulink	157

8.2	Nguyên lý hoạt động và việc thực hành trong simulink	159
	Khởi động simulink	159
	Các khối trong simulink	160
8.3	Một số ví dụ đơn giản	164
	Mô hình hóa một phương trình bằng sơ đồ khối	164
	Mô phỏng một quá trình động học	166
	Mô hình hóa một hệ động lực liên tục đơn giản	168
	Mô tả hệ dao động một bậc tự do	169
	Mô phỏng số tay máy một bậc tự do	171
8.4	Đơn giản sơ đồ simulink	174
	Đơn giản sơ đồ simulink bằng khốiFcn	174
	Đơn giản sơ đồ simulink bằng khối subsystem	175
	Kết hợp simulink và script file (m-file)	177
8.5	Xử lý kết quả mô phỏng	184
8.6	Bài tập thực hành	185
 Chương 9. Giải một số bài toán trong kỹ thuật bằng Matlab		
9.1	Bài toán hệ thanh	189
	Hệ thanh tĩnh định	189
	Hệ thanh siêu tĩnh	191
9.2	Bài toán uốn phẳng của dầm	193
9.3	Bài toán quỹ đạo chuyển động của viên đạn	199
9.4	Bài toán bắn trúng đích	203
9.5	Bài toán dao động	206
	Con lắc đơn	206
	Con lắc đơn dây treo đàn hồi	208
	Con lắc kép	211
	Dao động nhỏ của con lắc elliptic	213
	Hệ dao động ba bậc tự do	214
9.6	Phân tích động học cơ cấu	216
	Phân tích động học cơ cấu bốn khâu bản lề	218
9.7	Bài toán động học ngược rôbốt	221
	 Tài liệu tham khảo	 225

Môi trường Matlab

1.1 Matlab là gì?

MATLAB được viết tắt từ “**MA**Trix **LAB**oratory”, là một công cụ tính toán số và mô phỏng số, công cụ này được phát triển dựa trên các thư viện hàm tính toán số viết bằng ngôn ngữ lập trình FORTRAN. Matlab có giao diện thân thiện với người sử dụng.

Một định nghĩa khác: Matlab là một phần mềm đa năng tính toán số, thể hiện các số liệu bằng hình ảnh trực quan và là một ngôn ngữ đa năng cung cấp một môi trường linh hoạt cho việc tính toán kỹ thuật.

Matlab bao gồm nhiều công cụ để:

- thu thập dữ liệu (Data acquisition)
- phân tích và xử lý dữ liệu (Data analysis and exploration)
- hiển thị hình ảnh trực quan và xử lý hình ảnh (Visualization and image processing)
- tạo mẫu và phát triển thuật toán (Algorithm prototyping and development)
- mô hình hóa và mô phỏng (Modeling and simulation)
- lập trình và phát triển ứng dụng (Programming and application development).

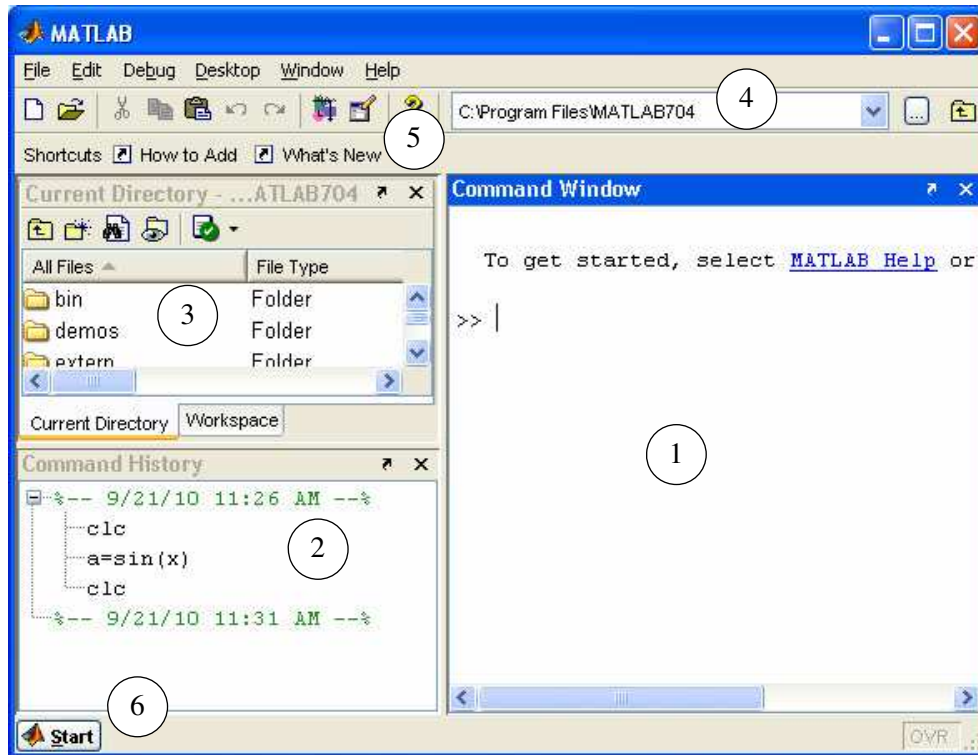
Matlab có thể chạy trên hầu hết các hệ máy tính: máy tính xách tay - Laptop, máy tính cá nhân PC, đến các hệ thống siêu máy tính (super computer). Matlab được điều khiển bởi các tập lệnh, tác động qua bàn phím trên cửa sổ điều khiển. Nó cũng cho phép một khả năng lập trình với cú pháp thông dịch lệnh – còn gọi m-file. Các lệnh hay bộ lệnh của Matlab lên đến con số hàng trăm và ngày càng được mở rộng nhờ các thư viện trợ giúp hay do người sử dụng tạo ra.

Để khởi động Matlab trong môi trường Windows bạn chỉ cần nhấp đúp chuột vào biểu tượng Matlab  có trên màn hình - Desktop.

Các lệnh của Matlab rất mạnh và hiệu quả, nó cho phép giải các loại bài toán khác nhau và đặc biệt khi xử lý các dữ liệu có cấu trúc kiểu vectơ và ma trận. Trong các phiên bản mới, người ta đã đưa việc tính toán ký tự vào phần mềm Matlab.

1.2 Giao diện người sử dụng

Giao diện của Matlab sau khi kích hoạt được thể hiện như trên hình 1-1 hoặc ở một dạng tương tự tùy theo lựa chọn của người sử dụng (bằng các lựa chọn khác nhau trong mục Desktop cho ta các dạng giao diện tương ứng).



Hình 1-1. Cửa sổ giao diện của Matlab

Các phần tử chính của giao diện này gồm:

1. Cửa sổ lệnh (Command Window).
2. Cửa sổ ghi lại các lệnh đã thực hiện (Command History).
3. Cửa sổ cho biết thư mục hiện thời (Current Directory) và danh sách các biến đang sử dụng (Workspace).
4. Thanh chỉ đường dẫn của thư mục hiện thời, cho phép chọn thư mục hiện thời.
5. Thanh Shortcut.
6. Nút Start.

1.3 Các phép tính số học cơ bản

Đối với người mới bắt đầu sử dụng Matlab, trước hết ta cần làm quen với các phép tính số học (cộng, trừ, nhân và chia: +, -, *, /). Với hai số a và b trong Matlab ta có thể thực hiện được các phép tính số học như liệt kê trong bảng dưới đây.

Phép tính	Cách viết toán học	cách viết trong Matlab
phép cộng	$c = a + b$	$c = a + b$
phép trừ	$c = a - b$	$c = a - b$
phép nhân	$c = a \cdot b$	$c = a * b$
phép chia	$c = a : b$	$c = a / b$
phép chia phải	$c = a : b$	$c = a / b$
phép chia trái	$c = b / a$	$c = a \backslash b$
lũy thừa	$c = a^b$	$c = a^b$

Ví dụ

```
>> a=3.5; b=7.5; % gán giá trị số cho hai biến
>> a+b % phép cộng hai số
ans =
    11
>> a-b % phép trừ hai số
ans =
    -4
>> a*b % phép nhân hai số
ans =
   26.2500
>> a/b % phép chia hai số, a:b (= phép chia phải)
ans =
    0.4667
>> a\b % phép chia trái (hiệu là b chia cho a, b/a)
ans =
    2.1429
>> a^b % phép lũy thừa, a mũ b
ans =
  1.2037e+004
```

Chú ý rằng, mọi dòng chữ sau dấu '%' đều được MATLAB bỏ qua và được sử dụng để diễn giải, bình luận. Dòng lệnh kết thúc với dấu chấm phẩy ';' sẽ không

xuất kết quả ra, còn dòng lệnh không có dấu ';' kết thúc (để trống) sẽ đưa kết quả ra khi dòng lệnh được thực hiện. Một dòng có thể được kéo dài bằng việc đánh '...' vào cuối dòng và tiếp tục câu lệnh (phép tính) ở dòng kế tiếp.

Thứ tự ưu tiên các phép toán

Khi tính toán một biểu thức gồm nhiều số hạng, nhiều phép tính thì thứ tự ưu tiên các toán tử rất quan trọng.

Thứ tự ưu tiên	Toán tử
1	Ngoặc đơn
2	Lũy thừa
3	Nhân và chia, từ trái qua phải
4	Cộng và trừ, từ trái qua phải

Ví dụ

Cần tính giá trị hàm

$$f(x) = \frac{x^2(x^4 - 5x^3 + 1.5) + 5(x + 10)}{x^2(x^2 + 2.5x + 5) + 1.5(x + 10)}$$

tại $x = x_0$ nào đó, chẳng hạn $x = 5$.

```
>> x=5;
>> tuso=x^2*(x^4-5*x^3+1.5)+5*(x+10);
>> mauso=x^2*(x^2+2.5*x+5)+1.5*(x+10);
>> f=tuso/mauso
f =
    0.1037
```

1.4 Phép gán

Trong Matlab dấu bằng "=" được sử dụng cho phép gán. Mặc dù trong cách viết thông thường ta hiểu dấu bằng thể hiện một phương trình, nhưng trong Matlab dấu bằng được định nghĩa là phép gán. Để phân biệt giữa hai cách thể hiện ta xét ví dụ sau. Nếu trong cửa sổ lệnh ta viết

```
>> x+18=120
```

Matlab sẽ báo lỗi với dòng hiển thị màu đỏ:

```
??? x+18=120
```

```
Error: The expression to the left of the equals sign is not a
valid target for an assignment.
```

Matlab không hiểu cách bạn viết một phương trình như trên giấy, mà nó chỉ có thể thực hiện được các phép tính và gán giá trị tính được cho một biến nào đó. Chẳng hạn ta có thể gán giá trị của phép tính $120-18$ cho biến x bằng cách viết

```
>> x=120-18
```

Một cách viết khác ta có thể sử dụng phép gán cho một phép tính lặp trong chương trình tính. Tức là ta có thể viết

```
>> x=x+12
```

Phép gán này chỉ hoạt động được nếu như trước đó ta đã có giá trị của biến x . Ví dụ, chuỗi các phép tính sau đây được thực hiện

```
>> x=32^3
```

```
x =  
32768
```

```
>> x=x+124
```

```
x =  
32892
```

Để sử dụng một biến trong vế phải của phép gán, chúng ta phải gán một giá trị cho biến đó trước khi sử dụng. Khi viết các dòng lệnh sau Matlab sẽ báo lỗi

```
>> x= 124
```

```
x =  
124
```

```
>> t=x+y
```

```
??? Undefined function or variable 'y'.
```

Lý do của lỗi trên là do biến y chưa được gán giá trị trước khi thực hiện cộng với biến x . Trong khi đó các dòng lệnh sau được thực hiện đúng.

```
>> x=124
```

```
x = 124
```

```
>> y=126
```

```
y = 126
```

```
>> t=x+y
```

```
t = 250
```

Trong nhiều phép gán (có thể là phép tính trung gian) nếu không muốn kết quả hiện ra dưới phép gán thì ta sử dụng dấu chấm phẩy (;) vào cuối biểu thức. Như thế trong cửa sổ lệnh ta không bị lãng phí không gian. Ví dụ

```
>> x=124;
```

```
>> y=126;
```

```
>> t=x+y
```

```
t = 250
```

Chúng ta cũng có thể gom nhiều phép gán trên cùng một dòng lệnh. Chẳng hạn

```
>> x=124; y=126; t=x+y
```

```
t = 250
```

Lưu ý rằng, các dấu “;” trong dòng lệnh trên báo cho Matlab biết là ta không muốn các giá trị của x và y hiện ra ở dòng dưới. Nếu muốn các giá trị này được hiển thị, ta thay chúng bằng các dấu phẩy “,”. Nếu ta không sử dụng dấu “;” hoặc dấu “,” giữa các phép gán Matlab sẽ báo lỗi.

<pre>>> x=124, y=126, t=x+y x = 124 y = 126 t = 250</pre>	<pre>>> x=124 y=126 t=x+y ??? x=124 y=126 t=x+y Error: Unexpected MATLAB expression.</pre>
--	--

Khi thực hiện nhiều phép tính, nếu muốn xóa bớt hoặc xóa tất cả các biến đã sử dụng để giảm bớt không gian sử dụng của bộ nhớ, ta sử dụng lệnh

clear tên-biến hoặc **clear all**

Trước khi làm việc đó ta nên xem lại danh sách các biến đã sử dụng bằng lệnh **who**. Khi thực hiện lệnh **who**, Matlab sẽ liệt kê cho bạn biết tất cả các biến đã được sử dụng. Chẳng hạn, với các lệnh đã thực hiện ở trên ta có

```
>> r=10; a=5^5; x=182; y=235; z=x+y
z =    417
>> V=4/3*pi*r^3
V =    4.1888e+003
>> t=x+a
t =    3307
>> who
Your variables are:
      V a r t x y z
```

Bằng lệnh **whos**, ta sẽ nhận được nhiều thông tin hơn về các biến như: tên biến sử dụng trong bộ nhớ, cỡ, số lượng bytes đã sử dụng, kiểu của biến.

```
>> whos

  Name      Size      Bytes  Class
  ----      -
V          1x1           8  double array
a          1x1           8  double array
r          1x1           8  double array
t          1x1           8  double array
x          1x1           8  double array
y          1x1           8  double array
z          1x1           8  double array

Grand total is 7 elements using 56 bytes
```

Với lệnh **clear** hoặc **clear all** tất cả không gian làm việc sẽ bị xóa. Khi không cần sử dụng đến một hay nhiều biến nào đó ta có thể xóa chúng bằng lệnh **clear var_name**. Chẳng hạn để xóa biến V, a và t ta thực hiện

```
>> clear V a t
```

Nếu muốn sử dụng lại dòng lệnh đã nhập vào cho các công việc tiếp theo (giữ nguyên hoặc để sửa đổi thành dòng lệnh mới), ta sử dụng hai mũi tên lên xuống trên bàn phím (\uparrow , \downarrow). Các dòng lệnh đã nhập vào sẽ xuất hiện và bạn có thể sửa đổi nếu cần. Các lệnh này có thể được sửa đổi ngay tại dòng lệnh hiện thời. Ta cũng có thể copy và paste các dòng lệnh ngay trên Command-Window.

Đối với các dòng lệnh dài phải viết xuống dòng thì trước khi xuống dòng, bạn phải kết thúc dòng thứ nhất bằng dấu chấm lửng – dấu ba chấm (...). Ví dụ

```
>> FirstClassHolders=109;
>> SecondClass=100;
>> Coach=121;
>> Crew=8;
>> TotalPeopleonPlane=FirstClassHolders + SecondClass + Coach...
+ Crew
TotalPeopleonPlane =    338
```

Dấu ba chấm “...” ở phía sau Coach trên dòng đó cho biết dòng lệnh này chưa kết thúc. Sau dấu ba chấm này bạn đánh ENTER, Matlab sẽ chuyển xuống dòng dưới và chờ các đầu vào tiếp theo.

Cho đến đây chúng ta đã biết cách đưa vào và sử dụng các biến cho các phép tính gán. Các kết quả tính toán đưa ra màn hình có bốn số sau dấu chấm nếu số đó có phần lẻ.

```
>> co = cos(0.2),    si = sin(0.2), tn = tan(0.2)
co = 0.9801
si = 0.1987
tn = 0.2027
```

Đó là định dạng short trong Matlab. Định dạng này đã được mặc định khi bạn khởi động và sử dụng Matlab. Nếu các kết quả với bốn số sau dấu chấm không đạt được độ chính xác yêu cầu của bạn, hãy thực hiện lệnh

```
>> format long
```

Matlab sẽ hiển thị kết quả tính với 14 số sau dấu chấm. Hãy quan sát ví dụ sau

```
>> format long
>> co = cos(0.2),    si = sin(0.2), tn = tan(0.2)
co = 0.98006657784124
si = 0.19866933079506
tn = 0.20271003550867

>> format short
```

```
>> co = cos(0.2), si = sin(0.2), tn = tan(0.2)
co = 0.9801
si = 0.1987
tn = 0.2027
```

Hãy so sánh các kết quả trên, chú ý rằng với *format short* thì số hạng thứ tư sau dấu chấm đã được làm tròn. Nếu bạn tính toán với ngành tài chính, liên quan đến đơn vị đo là tiền thì chỉ cần hai số thập phân sau dấu chấm là đủ, khi đó bạn sử dụng *format bank*. Sau đó các kết quả tính toán sẽ được làm tròn với hai số sau dấu chấm.

```
>> format bank
>> luonggio=35.55
luonggio = 35.55
>> luongtuan=luonggio*40
luongtuan = 1422.00
```

Với các số lớn, Matlab sẽ hiển thị bằng ký hiệu e mũ. Số 4.1264×10^5 sẽ được viết dạng e mũ là $4.1264e + 005$. Nếu muốn tất cả các số được hiển thị dạng e mũ thì bạn sử dụng lệnh *format short e* hoặc *format long e*. Cụ thể như trong ví dụ sau:

```
>> format short e
>> x=5.125*3.16
x = 1.6195e+001
>> format long e
>> x=5.125*3.16
x = 1.6195000000000000e+001
```

Nếu bạn gõ vào lệnh *format rat*, Matlab sẽ hiển thị kết quả bằng một phân số gần nhất với kết quả của bạn, ví dụ

```
>> format rat
>> x=5.125*3.16
x = 3239/200
```

1.5 Các định nghĩa toán học cơ bản

Để thuận tiện cho việc tính toán, trong Matlab người ta đã định nghĩa sẵn rất nhiều đại lượng toán học và các hàm cơ bản. Ví dụ như số π đã được định nghĩa sẵn với tên gọi là pi, và khi tính thể tích hình cầu bán kính R theo công thức $V = \frac{4}{3}\pi R^3$, trong Matlab ta thực hiện như sau

```
>> R = 2;
```



```
>> V = 4/3*pi*R^3
V = 33.5103
```

Một số quen thuộc khác trong toán học được sử dụng trong nhiều ứng dụng là số e với giá trị xấp xỉ bằng 2.718. Trong Matlab hàm mũ cơ số e , e^x được viết là $\exp(x)$. Dưới đây là một số ví dụ

```
>> exp(1)
ans = 2.7183
>> exp(2)
ans = 7.3891
```

Để tính căn của một số x ta sử dụng hàm $\text{sqrt}(x)$. Ví dụ

```
>> x = sqrt(9)
x = 3
>> y = sqrt(11)
y = 3.3166
```

Để tính lôgarít tự nhiên của số x , ta sử dụng hàm $\log(x)$. Ví dụ

```
>> log(3.2)
ans = 1.1632
>> x = 5; log(x)
ans = 1.6094
```

Đối với lôgarít cơ số 10 ta sử dụng hàm $\log_{10}(x)$

```
>> x = 3; log10(x)
ans = 0.4771
```

Dưới đây là các hàm toán học cơ bản

Các hàm toán học	
$\text{sqrt}(x)$	Tính căn của biến x
$\exp(x)$	Hàm e mũ, Exponential
$\expm1(x)$	Tính giá trị $\exp(x)-1$
$\log(x)$	Logarithm cơ số tự nhiên, cơ số e
$\log1p(x)$	Tính giá trị $\log(1+x)$
$\log_{10}(x)$	Logarithm cơ số 10
$\log_2(x)$	Logarithm cơ số 2
$\text{pow}_2(x)$	Hàm mũ cơ số 2, $\text{pow}_2(x) = 2^x$

<code>realpow(x,y)</code>	Tính x^y , với x, y là thực
<code>reallog(x)</code>	Logarithm cơ số tự nhiên của số thực
<code>realsqrt(x)</code>	Căn bậc hai của số lớn hơn hoặc bằng 0
<code>nthroot(x, n)</code>	Căn bậc n của số thực
<code>nextpow2(n)</code>	Trả lại số p đầu tiên sao cho $2^p \geq \text{abs}(n)$

Các hàm làm tròn và lấy phần dư (Rounding and remainder)	
<code>fix(x)</code>	Làm tròn số x bằng một số nguyên gần 0
<code>floor(x)</code>	Làm tròn số x bằng một số nguyên bé hơn gần nhất
<code>ceil</code>	Làm tròn số x bằng một số nguyên lớn hơn gần nhất
<code>round</code>	Làm tròn số x bằng một số nguyên gần nhất
<code>mod</code>	Lấy phần nguyên của phép chia
<code>rem</code>	Lấy phần dư của phép chia
<code>sign</code>	Hàm dấu

Các hàm lượng giác cơ bản như \sin , \cos , \tan , \cot đều được định nghĩa trong Matlab với đối số được mặc định cho bằng radian. Các hàm lượng giác ngược được mặc định là trả về giá trị radian. Các hàm này được định nghĩa trùng với tên hàm thường dùng, được viết bằng chữ nhỏ.

```
>> cos(pi/4)
ans = 0.7071
```

Để sử dụng các hàm lượng giác ngược như \arcsin , \arccos , \arctan ta chỉ việc thêm a vào phía trước tên của hàm lượng giác, ví dụ như $\text{asin}(x)$, $\text{acos}(x)$, $\text{atan}(x)$

```
>> format rat
>> atan(pi/3)
ans = 1110/1373
```

Bảng dưới đây liệt kê một số hàm lượng giác và các hàm lượng giác ngược:

Các hàm lượng giác - Trigonometric functions	
<code>sin</code>	Hàm \sin
<code>sind</code>	Hàm \sin với đối số là độ
<code>sinh</code>	Hàm \sin Hyperbolic
<code>asin</code>	Hàm \sin ngược, tức \arcsin

asind	Hàm sin ngược cho kết quả là độ
asinh	Hàm sin Hyperbolic ngược
cos	Hàm cos
cosd	Hàm cos với đối số là độ
cosh	Hàm cos Hyperbolic
acos	Hàm cos ngược, tức arccos
acosd	Hàm cos ngược cho kết quả là độ
acosh	Hàm cos Hyperbolic ngược
tan	Hàm tang, Tangent.
tand	Hàm tang với đối số là độ
tanh	Hàm tang Hyperbolic
atan	Hàm tang ngược, arctang
atand	Hàm tang ngược cho kết quả là độ
atan2	Hàm tang ngược cho kết quả từ -pi đến pi
atanh	Hàm tang Hyperbolic ngược
cot	Hàm cotang
cotd	Hàm cotang với đối số là độ
coth	Hàm cotang Hyperbolic
acot	Hàm cotang ngược, arccotang
acotd	Hàm cotang ngược cho kết quả là độ
acoth	Hàm cotang hyperbolic ngược

1.6 Số phức

Chúng ta cũng có thể đưa số phức vào trong Matlab. Nhớ lại rằng đơn vị phức được định nghĩa là căn của -1 . Trong Matlab số phức ký hiệu là i hoặc j

$$i = \sqrt{-1}$$

Một số phức có thể được viết dạng $z = x + iy$, trong đó x là phần thực và y là phần ảo của z . Việc nhập một số phức vào Matlab thật đơn giản, với i được mặc định là đơn vị ảo. Các phép tính trên số phức cũng được thực hiện một cách dễ hiểu. Ví dụ với hai số phức

$$\begin{aligned} a &= 2 + 3i \\ b &= 1 - i \\ \Rightarrow a + b &= 3 + 2i \end{aligned}$$

Phép tính này được thực hiện trong Matlab như sau

```
>> format short
>> a = 2 + 3i;
```

```
>> b = 1 - i;
>> c = a + b
c = 3.0000 + 2.0000i
```

Một số hàm liên quan đến số phức được liệt kê trong bảng dưới đây

Các hàm với số phức	
abs	Cho độ lớn hay modul véctơ phức
angle	Cho góc pha hay argument của số phức, radian
complex	Tạo lập dữ liệu phức từ các phần thực vào ảo
conj	Cho số phức liên hợp
imag	Cho phần ảo
real	Cho phần thực
isreal	Trả lại giá trị đúng, 1, nếu đối số là thực
cplxpair	Sắp xếp các số phức thành cặp liên hợp

Ví dụ với hai số phức z_1 và z_2 ta có:

<pre>>> format short >> z1=2.5+6.5i z1 = 2.5000 + 6.5000i >> z2=-0.5+4.5i z2 = -0.5000 + 4.5000i >> z1+z2 ans = 2.0000 +11.0000i >> z1-z2 ans = 3.0000 + 2.0000i >> z1*z2 ans = -30.5000 + 8.0000i</pre>	<pre>>> z1/z2 ans = 1.3659 - 0.7073i >> real(z1) ans = 2.5000 >> imag(z1) ans = 6.5000 >> abs(z1) ans = 6.9642 >> angle(z1) ans = 1.2036</pre>
--	--

1.7 Xử lý lỗi khi gõ lệnh

Như đã thấy ở trên, khi thực hiện các dòng lệnh bạn có thể làm không đúng và Matlab đã báo lỗi cho bạn. Error !. Nếu khi bạn kết thúc dòng lệnh bằng gõ phím ENTER và nhận ra rằng dòng lệnh trên có lỗi, bạn không nhất thiết phải gõ lại dòng lệnh đó, mà đơn giản bạn chỉ cần sử dụng các phím mũi tên lên hoặc xuống, \uparrow, \downarrow , để hiển thị lại dòng lệnh cần sửa. Sau khi sửa lỗi, và đánh phím ENTER Matlab sẽ đưa ra kết quả.

1.8 Kết thúc phiên làm việc với Matlab

Chúng ta vừa bắt đầu với một số lệnh cơ bản của Matlab, và bây giờ có thể bạn muốn ghi lại các công việc vừa thực hiện và thoát khỏi Matlab. Làm thế nào để kết thúc Matlab trên màn hình của bạn? Thật đơn giản bạn vào **exit** trong menu File, như khi sử dụng các chương trình khác trong Windows. Một cách khác là bạn gõ lệnh **quit** vào dấu nhắc trong cửa sổ lệnh và Matlab sẽ đóng lại. Kết thúc phiên làm việc.

1.9 Bài tập thực hành

Sử dụng Matlab để thực hiện tính các đại lượng sau:

1. $x = 12 \frac{13}{17}$;
2. $y = 16 \frac{13}{17} + 4^7$
3. $z = 9^{1.25}$
4. Đúng hai sai. Khi biến y chưa được gán giá trị, Matlab cho phép bạn thực hiện phép gán $x = y^2$ vào trong bộ nhớ để sử dụng cho việc tiếp theo.
5. Cho biết thể tích của hình trụ có chiều cao h và bán kính đáy r là $V = \pi r^2 h$, hãy sử dụng Matlab để tính thể tích hình trụ có chiều cao 12 cm và đường kính đáy là 4 cm.
6. Sử dụng Matlab tính giá trị sin của $\pi/3$ và biểu diễn kết quả dưới dạng một số hữu tỷ.
7. Hãy tạo một m-file để tính kết quả của $\sin(\pi/4)$, $\sin(\pi/3)$ và $\sin(\pi/2)$; biểu diễn chúng bằng các số hữu tỷ.

Chương 2.

Véc tơ, ma trận và Matlab

Các phép tính số học thông thường được thực hiện trên các số riêng lẻ được gọi là đại lượng vô hướng (scalar). Tuy nhiên trong rất nhiều bài toán ta cần thực hiện các phép tính lặp lại nhiều lần với những bộ số liệu. Công việc này được thực hiện một cách dễ dàng trong Matlab khi các bộ số liệu đó được lưu trữ dưới dạng mảng. Mảng trong Matlab có thể là một, hai hay nhiều chiều. Mảng một chiều còn gọi là véc tơ, mảng hai chiều là ma trận. Véc tơ và ma trận là hai đối tượng này được Matlab coi như kiểu dữ liệu chính của mình. Và do đó trong Matlab có rất nhiều công cụ hữu ích xử lý các mảng số kiểu véc tơ và ma trận. Chú ý rằng trong Matlab véc tơ được biểu diễn là một ma trận cột hoặc ma trận hàng.

2.1 Véc tơ và các phép tính trên véc tơ

Nhập véc tơ

Véc tơ là một mảng một chiều chứa các số. Matlab cho phép bạn tạo ra các véc tơ cột hoặc véc tơ hàng. Một véc tơ cột có thể được tạo ra trong Matlab bằng cách liệt kê các phần tử trong dấu ngoặc vuông [] và giữa các phần tử là dấu chấm phẩy “;”. Các véc tơ có số phần tử tùy ý. Chẳng hạn để tạo ra véc tơ cột với ba phần tử, ta viết

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

Các phép tính cơ bản trên véc tơ cột có thể được thực hiện với tên biến đã sử dụng để tạo ra nó. Nếu muốn nhân một số với một véc tơ, hay tích với một vô hướng. Giả sử muốn tạo ra một véc tơ có các phần tử bằng ba lần các phần tử của véc tơ vừa được tạo ra, tức là nhân véc tơ đã có với 3. Tất nhiên ta có thể gán số 3 này vào một biến, c, chẳng hạn. Ta thực hiện

```
>> c = 3;
>> b = c*a
```

```
b =
     6
     3
    12
```

Để tạo ra một vectơ hàng, chúng ta liệt kê các phần tử trong dấu ngoặc vuông [], và sử dụng dấu cách hoặc dấu phẩy để nằm giữa các phần tử. Ví dụ

```
>> v = [2 0 4]
v = 2 0 4
```

Hoặc sử dụng dấu phẩy

```
>> w = [1,1,9]
w = 1 1 9
```

Vectơ cột cũng có thể chuyển thành vectơ hàng và ngược lại nhờ phép chuyển vị. Giả sử có vectơ cột với n phần tử ký hiệu bởi:

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix}$$

chuyển vị của vectơ này là

$$\mathbf{v}^T = [v_1 \ v_2 \ \dots \ v_n].$$

Trong Matlab, phép chuyển vị được thể hiện bằng dấu nháy đơn (''). Chuyển vị vectơ hàng cho ta vectơ cột như trong ví dụ

```
>> a = [2; 1; 4];
>> y = a'
y = 2 1 4
```

Chuyển vị vectơ cột cho ta vectơ hàng

```
>> Q = [2 1 3]
Q = 2 1 3
>> R = Q'
R =
     2
     1
     3
```

Các phép tính cộng và trừ hai vectơ cùng cỡ

Các phép tính cộng và trừ hai vectơ tạo ra cho ta một vectơ mới. Để thực hiện được các phép tính cộng hoặc trừ, hai vectơ phải cùng dạng (cùng là cột hoặc cùng là hàng) và cùng có số phần tử. Các phép tính được thực hiện với tên biến của chúng. Ví dụ, thực hiện phép cộng hai vectơ

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

Và đây là phép trừ hai vectơ hàng

```
>> W = [3, 0, 3];
>> X = [2, 1, 1];
>> Y = W - X
Y =     1     -1     2
```

Tạo một vectơ từ các vectơ khác

Matlab cho phép bạn nối các vectơ lại với nhau để tạo ra một vectơ có nhiều phần tử. Gọi \mathbf{u} và \mathbf{v} là hai vectơ cột với số phần tử tương ứng là m và n , mà đã được tạo ra trong Matlab. Chúng ta có thể tạo ra vectơ \mathbf{w} với $m + n$ phần tử, trong đó m phần tử đầu là của vectơ \mathbf{u} và n phần tử cuối là của vectơ \mathbf{v} . Việc này được thực hiện bằng cách viết $\mathbf{w} = [\mathbf{u}; \mathbf{v}]$. Ví dụ

```
>> A = [1; 4; 5];
>> B = [2; 3; 3];
>> D = [A; B]
D =
     1
     4
     5
     2
     3
     3
```

Việc này cũng có thể thực hiện đối với các vectơ hàng. Để tạo vectơ hàng \mathbf{w} với $m + n$ phần tử từ hai vectơ hàng \mathbf{u} có m phần tử và \mathbf{v} có n phần tử, ta viết $\mathbf{w} = [\mathbf{u}, \mathbf{v}]$. Ví dụ


```
>> u = [12, 11, 9]
>> v = [1, 4];
>> w = [u, v]
w = 12 11 9 1 4
```

Tạo vectơ hàng có các phần tử cách đều

Matlab cung cấp công cụ để tạo ra vectơ có các phần tử cách đều với bước h , xuất phát từ giá trị a_1 và kết thúc tại giá trị a_n . Phần tử thứ k của vectơ này có giá trị là $a_k = a_1 + (k-1) \cdot h$. Toán tử hai chấm (:) cho ta thực hiện việc này với cú pháp như sau

$$\mathbf{a} = [a_1 : h : a_n]$$

Ví dụ để tạo ra một vectơ chứa danh sách các số chẵn từ 0 đến 10 ta viết

```
>> x = [0:2:10]
x = 0 2 4 6 8 10
```

Nhờ có kỹ thuật này mà việc vẽ đồ thị hàm số $y = f(x)$ trên đoạn $x = [a, b]$ được thực hiện một cách dễ dàng. Đoạn $[a, b]$ sẽ được chia đều với bước h đủ nhỏ thành một vectơ \mathbf{x} , sau đó tính giá trị của hàm số tại các điểm chia và ta nhận được một vectơ \mathbf{y} có số phần tử bằng số phần tử của vectơ \mathbf{x} , $y_i = f(x_i)$. Ví dụ

```
>> x = [0:0.1:1]
x = 0 0.10 0.20 0.30 0.40 0.50 0.60 0.70 0.80 0.90 1.00
```

Trường hợp $f(x) = e^x$ ta có

```
>> y = exp(x)
y =
Columns 1 through 9
1.0000 1.1052 1.2214 1.3499 1.4918 1.6487 1.8221 2.0138 2.2255
Columns 10 through 11
2.4596 2.7183
```

hoặc với $f(x) = x^2$

```
>> y = x.^2
y = 0 0.01 0.04 0.09 0.16 0.25 0.36 0.49 0.64 0.81 1.00
```

Chú ý rằng phép tính lũy thừa được ký hiệu bởi dấu mũ (^), và phép tính lũy thừa ở trên được viết là .^; nếu trong ví dụ trên bạn gõ vào >> y = x^2 thì Matlab sẽ báo lỗi !

```
??? Error using ==> mpower
Matrix must be square.
```

Các phép tính `.*` `./` `.^` sẽ được nói kỹ hơn ở phần sau.

Trong quá trình tạo vectơ với các phần tử cách đều, bạn có thể sử dụng bước h âm, tức là vectơ bạn tạo ra sẽ giảm đều từ giá trị đầu về giá trị cuối. Ví dụ để tạo ra dãy số giảm từ 100 về 80 với bước là 5, ta viết

```
>> u = [100:-5:80]
u = 100    95    90    85    80
```

Một cách khác để tạo ra vectơ hàng với các phần tử cách đều là sử dụng lệnh **linspace**. Lệnh `x = linspace(a,b)` cho ta một vectơ hàng `x` gồm 100 phần tử cách đều, $x_1 = a$ và $x_{100} = b$; trong khi đó lệnh `x = linspace(a,b,n)`, với n là số nguyên dương, cho ta một vectơ hàng `x` gồm n phần tử cách đều, $x_1 = a$ và $x_n = b$. Cả trong hai trường hợp Matlab tự động xác định bước h để cho ta giá trị đúng của các phần tử.

Matlab cũng cho phép bạn tạo một vectơ hàng n phần tử cách đều theo thang lôgarít bằng lệnh

```
logspace(a,b,n)
```

Lệnh này cho ta n phần tử nằm giữa 10^a và 10^b . Ví dụ

```
>> x = logspace(1,2,5)
x = 10.0000 17.7828 31.6228 56.2341 100.0000
```

Ta sẽ hãy thử lại với

```
>> log10(x) % và nhận được kết quả
ans =
    1.0000    1.2500    1.5000    1.7500    2.0000
```

Một ví dụ khác

```
>> logspace(-1,1,6)
ans = 0.1000 0.2512 0.6310 1.5849 3.9811 10.0000
```

Các đặc trưng của véctor

Lệnh **length** trả lại cho ta số các phần tử mà véctor đó chứa. Lệnh có thể áp dụng được cả với véctor hàng và véctor cột. Ví dụ

```
>> A = [2;3;3;4;5];
>> length(A)
ans = 5
>> B = [1, 1, 2];
>> length(B)
ans = 3
```

Để tìm các phần tử lớn nhất và nhỏ nhất của véctor ta sử dụng lệnh **max** và **min**. Ví dụ

```
>> A = [8 4 4 1 7 11 2 0];
>> max(A)
ans = 11
>> min(A)
ans = 0
```

Để tính tổng của tất cả các phần tử của véctor x, ta sử dụng lệnh **sum**. Ví dụ

```
>> A = [8 4 4 1 7 11 2 0];
>> sum(A)
ans = 37
```

Như ta biết độ dài của véctor $\mathbf{v} = [v_1 \ v_2 \ \dots \ v_n]^T$ được định nghĩa theo biểu thức

$$|v| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}.$$

Trong Matlab độ dài của véctor có thể được xác định bằng hai phép tính. Trước hết ta tính tích vô hướng của véctor này với chính nó bằng phép nhân phần tử (**.***). Sau đó khai căn của tổng bình phương các phần tử. Ví dụ

```
>> x = [8 4 4 1 7 2 0];
>> a = x.*x
a = 64 16 16 1 49 4 0
>> b=sum(a)
b = 150
>> mag=sqrt(b)
mag = 12.2474
```

Một cách khác ngắn gọn hơn là sử dụng lệnh **norm(x,2)**, lệnh này cho ta độ dài véctor x, theo như định nghĩa ở trên, hay còn gọi là chuẩn Euclid của véctor x.

```
>> norm(x,2)
ans = 12.2474
```

Trường hợp \mathbf{v} là một vectơ phức, thì độ dài của vectơ được định nghĩa như sau

$$|\mathbf{v}| = \sqrt{\mathbf{v}^\dagger \mathbf{v}}, \text{ với } \mathbf{v}^\dagger \text{ là số phức liên hợp của } \mathbf{v}.$$

Ví dụ với $\mathbf{v} = [i \ 1+2i \ 4]^T$, số phức liên hợp của nó là $\mathbf{v}^\dagger = [-i \ 1-2i \ 4]$ và ta tính được độ dài của \mathbf{v}

$$|\mathbf{v}|^2 = \mathbf{v}^\dagger \mathbf{v} = [-i \ 1-2i \ 4] \begin{bmatrix} i \\ 1+2i \\ 4 \end{bmatrix} = 22 \Rightarrow |\mathbf{v}| = \sqrt{\mathbf{v}^\dagger \mathbf{v}} = \sqrt{22}.$$

Trong Matlab số phức liên hợp nhận được bằng lệnh **conj(v)**. Các dòng lệnh sau sẽ tính cho ta độ dài của vectơ phức \mathbf{v} .

```
>> v=[i; 1+2i; 4];
>> u=conj(v)
u =
      0 - 1.0000i
      1.0000 - 2.0000i
      4.0000

>> s = sum(u.*v)
s = 22
>> mag_v=sqrt(s)
mag_v = 4.6904
```

Tất nhiên chúng ta có thể gộp các dòng lệnh trên trong một dòng như sau:

```
>> c = sqrt(sum(conj(v).*v))
c = 4.6904
```

Lệnh **abs** với đối số là vectơ trả lại cho ta một vectơ chứa giá trị tuyệt đối của các phần tử vectơ đó. Điều này thể hiện trong ví dụ sau

```
>> A = [-2 0 -1 9]
>> B = abs(A)
B = 2 0 1 9
```

Tích vô hướng và tích có hướng hai vectơ

Tích vô hướng (dot product) hai vectơ $\mathbf{a} = (a_1 \ a_2 \ \dots \ a_n)$ và $\mathbf{b} = (b_1 \ b_2 \ \dots \ b_n)$ là một số được xác định bởi

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \dots + a_n b_n = \sum_{i=1}^n a_i b_i$$

Trong matlab, tích vô hướng của hai vectơ \mathbf{a}, \mathbf{b} có thể được tính bằng lệnh `dot(a,b)`. Ví dụ

```
>> a = [1;4;7];      b = [2;-1;5];
>> c = dot(a,b)
c =      33
```

Tích vô hướng hai vectơ có thể được sử dụng để tính độ dài của một vectơ. Bằng cách lấy căn của tích vô hướng của vectơ \mathbf{a} với chính nó.

```
>> a = [0; 3; 4];
>> mag = sqrt(dot(a, a))
mag =      5
```

Phép nhân vô hướng đối với hai vectơ phức cũng được thực hiện như đối với hai vectơ thực.

```
>> u = [-i; 1 + i; 4 + 4*i];
>> dot(u,u)
ans =      35
```

Một phép tính quan trọng khác đó là tích có hướng của hai vectơ (cross product). Phép tính này chỉ áp dụng hai vectơ ba chiều, vectơ có 3 phần tử. Ví dụ

```
>> a = [1 2 3];      b = [2 3 4];
>> c = cross(a, b)
c = -1 2 -1
>> d = cross(b, a)
d =      1      -2      1
```

Lưu ý rằng tích vô hướng có tính chất hoán vị, còn tích có hướng thì không có tính chất này.

Tham chiếu đến các phần tử của vectơ

Trong Matlab có nhiều kỹ thuật có thể sử dụng để tham chiếu đến một hoặc nhiều phần tử của vectơ. Thành phần thứ i của vectơ \mathbf{v} có thể được tham chiếu tới bằng cách viết $v(i)$. Ví dụ

```
>> a = [12; 17; -2; 0; 4; 27];
>> a(2)
```

```
ans = 17
>> a(6)
ans = 27
```

Nếu muốn tham chiếu tới tất cả các phần tử của một vectơ, ta sử dụng toán tử hai chấm (:), ví dụ như $v(:)$, Matlab sẽ đưa ra tất cả các phần tử của vectơ đó.

```
>> a(:)
ans =
    12
    17
    -2
     0
     4
    27
```

Ta cũng có thể nhặt ra một dãy liên nhau các phần tử của vectơ. Bằng cách viết $v(i:j)$ Matlab sẽ nhặt ra các phần tử từ $v(i)$ đến $v(j)$. Ví dụ muốn có một vectơ ba phần tử từ ba phần tử đầu tiên của vectơ trên, ta viết

```
>> v = a(1:3)
v =
    12
    17
    -2
```

Một số hàm định sẵn cho vectơ

Trong Matlab có nhiều hàm đã được định nghĩa sẵn đối với biến vectơ như liệt kê dưới đây:

length(v)	Cho biết chiều dài, số phần tử của vectơ v
sum(v)	Tổng các phần tử của vectơ v
prod(v)	Tích các phần tử của vectơ v
min(v)	Phần tử nhỏ nhất của vectơ v
max(v)	Phần tử lớn nhất của vectơ v
sort(v)	Xếp các phần tử tăng dần
find(v)	Tìm các phần tử khác không trong v

2.2 Biểu diễn đa thức và các phép tính đa thức

Nhập đa thức

Đa thức có một vai trò quan trọng trong việc xấp xỉ hàm và phân tích ứng xử động lực trong miền tần số của các hệ động lực tuyến tính. Trong Matlab một đa thức được đưa vào dưới dạng một vectơ hàng. Khi đưa vào hay khai báo một đa thức ta cần biểu diễn dạng đầy đủ của nó, ví dụ xét đa thức

$$p_5(s) = a_5s^5 + a_4s^4 + a_2s^2 + a_1s$$

với dạng đầy đủ như sau

$$p_5(s) = a_5s^5 + a_4s^4 + 0s^3 + a_2s^2 + a_1s^1 + 0s^0.$$

Mỗi đa thức bậc n với các hệ số từ $a_n, a_{n-1}, \dots, a_1, a_0$ kể cả các hệ số ứng với giá trị 0, được biểu diễn bằng một vectơ hàng, bắt đầu từ bậc cao nhất a_n giảm đến a_0 .

Ví dụ đa thức sau đây

$$p_5(s) = 8s^5 + 3s^4 - 4s^2 + 6s$$

được biểu diễn trong Matlab bằng một vectơ hàng là

```
>> p = [8 3 0 -4 6 0]
      p = 8 3 0 -4 6 0
```

Matlab sẽ xác định bậc của đa thức từ số phần tử của vectơ

$$n = \text{length}(p) - 1$$

Ví dụ trong ví dụ trên

```
>> n = length(p)-1
      n = 5
```

Các phép tính trên đa thức

Phép nhân đa thức

Nhân hai đa thức $p_n(s)$ và $p_m(s)$ ta được đa thức $p_{n+m}(s)$ có bậc bằng tổng bậc của hai đa thức đã cho, và thực hiện trong Matlab bằng lệnh conv

$$p = \text{conv}(p_n, p_m)$$

Ví dụ tính tích hai đa thức

$$p_5(s) = 8s^5 + 3s^4 - 4s^2 + 6s \text{ và } p_2(s) = 3s^2 - 4s + 6$$

như sau:

```
>> p5=[8 3 0 -4 6 0];
>> p2=[3 -4 6];
>> p=conv(p5,p2)
```

```

p =      24      -23      36      6      34      -48      36      0
>> length(p)-1
ans = 7

```

Kết quả của phép nhân trên là một đa thức bậc 7

$$p_7(s) = 24s^7 - 23s^6 + 36s^5 + 6s^4 + 34s^3 - 48s^2 + 36s^1 + 0s^0$$

Phép chia đa thức

Chia đa thức $p_n(s)$ cho đa thức $p_m(s)$, $n \geq m$, ta được đa thức $q_{n-m}(s)$ có bậc bằng hiệu bậc của hai đa thức đã cho và một phần dư. Công việc này được thực hiện trong Matlab bằng lệnh `deconv`

$$[q, r] = \text{deconv}(p_n, p_m)$$

Ví dụ cần thực hiện phép chia $P_5(s)$ cho $P_2(s)$

$$p_5(s) = 6s^5 + 1s^4 - 12s^3 + 52s^2 - 46s + 39$$

$$p_2(s) = 3s^2 - 4s + 6$$

```

>> p5=[6 1 -12 52 -46 39];
>> p2=[3 -4 6];
>> [q,r]=deconv(p5,p2)
q = 2 3 -4 6
r = 0 0 0 0 2 3

```

Biểu diễn các kết quả này ta có

$$\frac{p_5(s)}{p_2(s)} = q + \frac{r(s)}{p_2(s)} \quad \Rightarrow \quad q(s) = 2s^3 + 3s^2 - 4s + 6, r(s) = 2s + 3$$

Trường hợp phép chia không dư thì $r(s) = 0$.

Phép cộng và trừ đa thức

Đối với các phép tính cộng và trừ hai đa thức, Matlab yêu cầu các đa thức này phải cùng bậc. Để làm việc này ta sẽ thêm các phần tử 0 vào phía trái của véc tơ ứng với đa thức có bậc nhỏ hơn, sao cho hai véc tơ có cùng độ dài. Trong Matlab không có hàm thực hiện việc cộng, trừ hai đa thức. Khi cộng hoặc trừ hai đa thức $p_n(s)$ và $p_m(s)$, $n > m$, ta được

$$p_a = p_n(s) + p_m(s)$$

$$p_s = p_n(s) - p_m(s)$$

và ta cần thực hiện như sau


```
pa = pn + [zeros(1,length(pn)-length(pm)) pm]
ps = pn - [zeros(1,length(pn)-length(pm)) pm]
```

Cụ thể

```
>> pa=p5+[zeros(1,length(p5)-length(p2)) p2]
pa = 6 1 -12 55 -50 45
>> ps=p5-[zeros(1,length(p5)-length(p2)) p2]
ps = 6 1 -12 49 -42 33
```

Không điểm hay nghiệm của phương trình đa thức

Hàm

roots(p)

tính cho ta các nghiệm của phương trình đa thức $p_n(s) = 0$. Các nghiệm tìm được có thể là thực hoặc phức. Ví dụ khi tìm nghiệm của các phương trình

$$p_5(s) = 6s^5 + 1s^4 - 12s^3 + 52s^2 - 46s + 39 = 0$$

$$p_2(s) = 3s^2 - 4s + 6 = 0$$

ta thực hiện và thu được

```
>> p2 = [3 -4 6];
>> p5 = [6 1 -12 52 -46 39];
>> roots(p2)
ans =
    0.6667 + 1.2472i
    0.6667 - 1.2472i
>> roots(p5)
ans =
   -2.6672
    0.7770 + 1.1589i
    0.7770 - 1.1589i
    0.4733 + 1.0138i
    0.4733 - 1.0138i
```

Xây dựng đa thức từ các không điểm cho trước

Hàm

p = poly(nu)

đưa ra cho ta một đa thức nhận các giá trị trong vectơ nu làm nghiệm.

Ví dụ cần đưa ra một đa thức nhận các giá trị sau là nghiệm

$$nu = [1-2i \quad 1+2i \quad 3 \quad 4]$$

```
>> nu = [1-2i 1+2i 3 4] ;
>> p4=poly(nu)
p4 = 1 -9 31 -59 60
```

Đa thức cần tìm sẽ là

$$p_4(s) = s^4 - 9s^3 + 31s^2 - 59s + 60$$

Ta có thể kiểm tra lại bằng lệnh roots(p)

```
>> roots(p)
ans =
    4.0000
```

```

3.0000
1.0000 + 2.0000i
1.0000 - 2.0000i

```

Giá trị của đa thức tại một điểm

Hàm

`polyval(p,s0)`

tính cho ta giá trị của đa thức tại điểm s_0 , $p(s_0)$. Ví dụ với đa thức $p_4(s)$ như trên, ta tính được $p_4(3)$:

```

>> polyval(p,3)
ans = 0

```

Đạo hàm đa thức

Hàm

`polyder(p)`

tính cho ta đạo hàm của đa thức $p_n(s)$, kết quả là một đa thức có bậc nhỏ hơn bậc của đa thức ban đầu là 1, $p_{n-1}(s)$. Ví dụ với đa thức $p_4(s)$ như trên, ta tính được $p_4'(s)$:

```

>> dp4=polyder(p)
dp4 = 4 -27 62 -59

```

hay

$$\frac{dp_4(s)}{ds} = 4s^3 - 27s^2 + 62s - 59.$$

Hàm

`polyder(pn,pm)`

tính cho ta đạo hàm của tích các đa thức $p_n(s), p_m(s)$. Lệnh này tương đương với hai lệnh: `p = conv(pn,pm)`; và `polyder(p)`.

Ví dụ:

```

>> p2 = [3 -4 6];
>> p5 = [6 1 -12 52 -46 39];
>> polyder(p2,p5)
ans = 126 -126 -20 840 -1254 1226 -432
>> p=conv(p2,p5)
p = 18 -21 -4 210 -418 613 -432 234
>> polyder(p)
ans = 126 -126 -20 840 -1254 1226 -432

```

Trong phần tiếp theo ta sẽ xem các kỹ thuật vừa nêu để tham chiếu đến các phần tử của một mảng số hai chiều tức ma trận.

2.3 Ma trận và các phép tính cơ bản trên ma trận

Nhập ma trận

Ma trận là một mảng số hai chiều, được sắp xếp theo hàng và cột. Để tạo một ma trận trong Matlab, chúng ta đặt từng hàng vào trong dấu ngoặc vuông [], các hàng được phân biệt với nhau bằng dấu chấm phẩy (;), và trong mỗi hàng các phần tử được phân biệt với nhau bằng dấu cách hoặc bằng dấu phẩy. Ví dụ với hai ma trận

$$\mathbf{A} = \begin{bmatrix} -1 & 6 \\ 7 & 11 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 7 & 4 \\ 3 & 0 & 1 \end{bmatrix}$$

ta đưa vào Matlab như sau:

```
>> A = [-1, 6; 7, 11]
A =
    -1     6
     7    11
>> B = [2, 0, 1; -1, 7, 4; 3, 0, 1]
B =
     2     0     1
    -1     7     4
     3     0     1
```

Nhân ma trận với một số, phép cộng và trừ hai ma trận

Các phép tính đối với vectơ như nhân với một số, cộng hoặc trừ hai vectơ cùng dạng, cùng cỡ, đều có thể áp dụng đối với ma trận. Tất nhiên khi cộng hoặc trừ hai ma trận thì yêu cầu các ma trận này phải cùng có số hàng m và số cột n , không nhất thiết m bằng n . Ví dụ

```
>> A = [-3, 6; 7, 10] ;
>> B = [5, 1; -1, 3] ;
>> C=2*A
C =
    -6    12
    14    20
>> A+B
ans =
     2     7
     6    13
>> A-B
ans =
    -8     5
     8     7
```

Chuyển vị ma trận

Chuyển vị một ma trận tức là sắp xếp lại hàng thành cột và cột thành hàng. Ví dụ

$$\mathbf{A} = \begin{bmatrix} 1 & -2 \\ 3 & 5 \\ 7 & 4 \end{bmatrix}, \quad \mathbf{A}^T = \begin{bmatrix} 1 & 3 & 7 \\ -2 & 5 & 4 \end{bmatrix}$$

Trong Matlab, ta sử dụng dấu nháy đơn để thực hiện chuyển vị một ma trận. Ví dụ

```
>> A = [-1 2 0; 6 4 1]
A =
    -1     2     0
     6     4     1
>> B = A'
B =
    -1     6
     2     4
     0     1
```

Nếu ma trận có chứa phần tử phức, phép chuyển vị ma trận sẽ cho ta ma trận liên hợp của ma trận đã cho. Ví dụ

```
>> C = [1 + i, 4 - i; 5 + 2*i, 3 - 3*i]
C =
    1.0000 + 1.0000i    4.0000 - 1.0000i
    5.0000 + 2.0000i    3.0000 - 3.0000i
>> D = C'
D =
    1.0000 - 1.0000i    5.0000 - 2.0000i
    4.0000 + 1.0000i    3.0000 + 3.0000i
```

Nếu muốn có ma trận chuyển vị của ma trận phức thì ta phải sử dụng kết hợp dấu chấm và nháy đơn (.'') thay vì sử dụng chỉ riêng nháy đơn (').

```
>> D = C.'
D =
    1.0000 + 1.0000i    5.0000 + 2.0000i
    4.0000 - 1.0000i    3.0000 - 3.0000i
```

Phép nhân ma trận

Xét hai ma trận **A** và **B**. Nếu **A** là ma trận cỡ $m \times p$ và **B** là ma trận cỡ $p \times n$, thì chúng có thể nhân được với nhau để tạo ra ma trận cỡ $m \times n$. Ta viết $\mathbf{C} = \mathbf{AB}$, và trong Matlab được viết là $\mathbf{C} = \mathbf{A} * \mathbf{B}$. Nên nhớ rằng nếu cỡ của hai ma trận không

phù hợp (số cột của ma trận **A** phải bằng số hàng của ma trận **B**, khi nhân **A** với **B**), thì Matlab sẽ thông báo lỗi. Error! Ví dụ cần nhân ma trận A với ma trận B sau

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}$$

Ở đây ta cần nhắc lại sự khác nhau giữa phép nhân hai ma trận và phép nhân mảng (tức tích phần tử hai ma trận)

```
>> A = [2 1; 1 2]; B = [3 4; 5 6];
>> A.*B
ans =
     6     4
     5    12

>> A*B
ans =
    11    14
    13    16
```

Và dưới đây là một ví dụ khác

$$\mathbf{A} = \begin{bmatrix} 1 & 4 \\ 8 & 0 \\ -1 & 3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} -1 & 7 & 4 \\ 2 & 1 & -2 \end{bmatrix}$$

Hai ma trận này có cỡ phù hợp để thực hiện phép nhân hai ma trận, tuy nhiên lại không phù hợp để thực hiện phép nhân mảng.

```
>> A = [1 4; 8 0; -1 3]; B = [-1 7 4; 2 1 -2];
>> C = A*B
C =
     7    11    -4
    -8    56    32
     7    -4   -10

>> C = A.*B
??? Error using ==> times
Matrix dimensions must agree.
```

Một số phép tính cơ bản khác

Về các phép tính ma trận, trong Matlab còn có những phép tính mà bạn có thể chưa gặp và chưa sử dụng trong giáo trình nhập môn về đại số tuyến tính. Chẳng hạn như Matlab cho phép cộng một số với một ma trận hay một véc-tơ, kết quả là tất cả các phần tử của ma trận đều được cộng với số đó. Ví dụ dưới đây chỉ ra điều đó

```
>> A = [1 2 3 4];
>> b = 2;
>> C = b + A
C =
     3     4     5     6
```

Các ma trận đặc biệt

Các ma trận đặc biệt như ma trận đơn vị, ma trận không, ma trận 1, ma trận kỳ diệu (magic matrix) đều có thể được tạo ra bằng các hàm sẵn có trong Matlab. Để tạo ra một ma trận đặc biệt loại này, ta chỉ việc gọi tên hàm với các đối số tương ứng.

Ma trận đơn vị là một ma trận vuông có các phần tử trên đường chéo chính bằng 1, còn các phần tử khác bằng 0. Ma trận 1 là ma trận mà tất cả các phần tử đều bằng 1, còn ma trận 0 là ma trận mà tất cả các phần tử đều bằng 0. Ma trận kỳ diệu cỡ $n \times n$ là ma trận vuông gồm n^2 phần tử, đó là các số từ 1, 2, 3 đến n^2 (với $n > 2$). Các số này được sắp xếp sao cho tổng các phần tử của mỗi hàng, mỗi cột và đường chéo là như nhau.

Bảng dưới đây liệt kê các hàm Matlab để tạo ra các ma trận đặc biệt nêu trên.

eye(m,m)	tạo ma trận đơn vị cỡ mxm
eye(m)	tạo ma trận đơn vị cỡ mxm
eye(m,n)	tạo ma trận đơn vị mở rộng cỡ mxn
zeros(m,n)	tạo ma trận không cỡ mxn
zeros(m,m)	tạo ma trận không cỡ mxm, hay ma trận vuông
zeros(m)	
ones(m,n)	cho ta một ma trận 1 cỡ mxn
ones(m,m)	cho ta một ma trận 1 cỡ mxm, hay ma trận vuông
ones(m)	
magic(n)	tạo ma trận magic vuông cỡ nxn, $n > 2$

Các ví dụ sau minh họa cho các hàm nêu trên.

<pre>>> eye(3) ans = 1 0 0 0 1 0 0 0 1</pre>	<pre>>> zeros(3,2) ans = 0 0 0 0 0 0</pre>
<pre>>> eye(2,2) ans = 1 0 0 1</pre>	<pre>>> zeros(2) ans = 0 0 0 0</pre>

<pre>>> eye(2,3) ans = 1 0 0 0 1 0 >> zeros(3,3) ans = 0 0 0 0 0 0 0 0 0</pre>	<pre>>> ones(2,2) ans = 1 1 1 1 >> ones(2) ans = 1 1 1 1 >> ones(2,3) ans = 1 1 1 1 1 1</pre>
<pre>>> magic(3) ans = 8 1 6 3 5 7 4 9 2</pre>	<pre>>> magic(5) ans = 17 24 1 8 15 23 5 7 14 16 4 6 13 20 22 10 12 19 21 3 11 18 25 2 9</pre>

Ngoài ra để tạo ra ma trận đường chéo, hay ma trận băng – ma trận gồm một số đường chéo, ta có thể sử dụng hàm **diag**. Hàm diag với đối số là ma trận sẽ trả lại một véc-tơ chứa đường chéo của ma trận, cũng với hàm diag nhưng biến số là một véc-tơ sẽ cho ta một ma trận đường chéo.

<pre>>> A1=[1 2 3; 3 4 5; 3 5 7] A1 = 1 2 3 3 4 5 3 5 7 >> Ad=diag(A1) Ad = 1 4 7</pre>	<pre>>> Add=diag(Ad) Add = 1 0 0 0 4 0 0 0 7</pre>
--	---

Với hàm diag ta có thể xây dựng được ma trận băng (band-matrix), diag(d,j) tạo ra ma trận đường chéo phụ (chỉ số j là số thứ tự của đường chéo phụ, với j dương đường chéo phụ phía trên đường chéo chính, với j âm đường chéo phụ phía dưới đường chéo chính).

diag(d,j) %Cho ma trận có đường chéo phụ j là véc-tơ d
diag(A,j) %Cho véc-tơ chứa đường chéo phụ j ma trận A

```
>> d=[4 4 4 4]      % cac phan tu tren duong cheo chinh
d =
     4     4     4     4
>> d1=[-2 -2 -2]    % cac phan tu tren duong cheo phu tren
d1 =
    -2    -2    -2
>> d2=[-1 -1 -1]    % cac phan tu tren duong cheo phu duoi
d2 =
    -1    -1    -1
>> Dd=diag(d)+diag(d1,1)+diag(d2,-1)
Dd =
     4    -2     0     0
    -1     4    -2     0
     0    -1     4    -2
     0     0    -1     4
>> diag(Dd,1)
ans =
    -2
    -2
    -2
```

Tham chiếu đến các phần tử của ma trận

Các phần tử, các cột hay các hàng của ma trận đều có thể được tác động đến nhờ cách đánh chỉ số của chúng. Chỉ số của các phần tử của ma trận là cặp số nguyên (i,j), i là chỉ số hàng và j là chỉ số cột. Các số này bắt đầu từ 1 (1, 2, ...), Matlab không sử dụng chỉ số 0 như một số ngôn ngữ lập trình khác. Ví dụ đối với ma trận

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

Chúng ta có thể nhặt ra một phần tử bất kỳ ở hàng i và cột j, A(i,j), như sau

```
>> A(2,3)
ans = 6
```

Để lấy tất cả các phần tử của cột j, ta viết A(:,j). Ví dụ cần lấy ra cột thứ hai của A:

```
>> c2 = A(:,2)
ans =
     2
     5
     8
```

Hay để lấy ra hàng thứ i ta viết A(i,:), ví dụ hàng thứ 3:


```
>> r3=A(3,:)
```

```
r3 =  
      7      8      9
```

Ta có thể lấy ra các phần tử của các cột từ cột i đến cột j, bằng cách viết $A(:,i:j)$. Ví dụ cần lấy ra cột 2 và cột 3, ta viết :

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

```
ans =  
      2      3  
      5      6  
      8      9
```

Từ một ma trận đã có ta có thể xây dựng được một ma trận con của nó, bằng cách lấy ra các phần tử thuộc hàng từ i1 đến i2 và các cột từ j1 đến j2. Ví dụ

```
>> A(2:3,1:2)
```

```
ans =  
      4      5  
      7      8
```

Nếu muốn thay đổi giá trị của phần tử nào đó trong ma trận, ta có thể sử dụng tham chiếu đến phần tử đó để gán cho nó một giá trị mới. Ví dụ muốn thay đổi giá trị của phần tử $A(1,1)$ bằng -8 , ta viết

```
>> A(1,1) = -8
```

```
A =  
    -8      2      3  
      4      5      6  
      7      8      9
```

Để tạo ra một mảng rỗng trong Matlab, đơn giản hãy để trống trong dấu ngoặc vuông $[]$. Cách viết này cũng có thể sử dụng để xóa bỏ hàng hoặc cột của ma trận. Ví dụ để xóa bỏ hàng 2 của A ta viết:

```
>> A(2,:)=[]
```

```
A =  
    -8      2      3  
      7      8      9
```

Bằng cách đó ta đã biến được ma trận cỡ 3×3 thành ma trận 2×3 . Nhờ cách tham chiếu các hàng và các cột ta có thể tạo ra được các ma trận mới. Trong ví dụ sau, ta sao chép cột thứ nhất của A bốn lần để tạo nên một ma trận mới cỡ 4×3 :

```
>> E = A([1,1,1,1],:)
```

```
E =
    -8     2     3
    -8     2     3
    -8     2     3
    -8     2     3
```

Hay một ví dụ khác, lấy hàng 1 và hàng 2 của **A** để tạo ra ma trận **F**:

```
>> F = A([1,2], :)
F =
    -8     2     3
     7     8     9
    -8     2     3
```

Ma trận khối

Xây dựng ma trận từ các ma trận con. Ví dụ cho các ma trận vuông **A** và **B**:

$$\mathbf{A} = \begin{bmatrix} 2 & 4 \\ 7 & 5 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 4 & 2 \\ 5 & 7 \end{bmatrix}, \quad \text{hãy tạo ra ma trận } \mathbf{Q} = \begin{bmatrix} \mathbf{0}_{2 \times 2} & \mathbf{E}_{2 \times 2} \\ -\mathbf{A}^{-1}\mathbf{B} & \mathbf{B}\mathbf{A}^T \end{bmatrix}.$$

```
>> A=[2 4; 7 5]
>> B=[4 2; 5 7]
>> Q=[zeros(2) eye(2); -A\B B*A']
Q =
     0         0         1         0
     0         0         0         1
     0        -1        16        38
    -1         0        38        70
```

Tạo dựng ma trận từ các ma trận con hay từ các vectơ, chẳng hạn như khi ta muốn chèn thêm một hàng hay cột có cỡ tương ứng vào một ma trận hay ghép nhiều ma trận thành một ma trận có cỡ lớn hơn.

<pre>>> A = [1 2; 3 4] A = 1 2 3 4 >> B = [11 12; 13 14] B = 11 12 13 14 >> C = [10 2; 3 6] C = 10 2 3 6</pre>	<pre>>> N = [C C2] N = 10 2 0 2 3 6 3 0 >> M = [A, B; N] M = 1 2 11 12 3 4 13 14 10 2 0 2 3 6 3 0</pre>
---	---

```
>> C2 = [0 2; 3 0]
```

```
C2 =
```

```
0    2
3    0
```

```
>> M=[A, B; C, C2]
```

```
M =
```

```
1    2   11   12
3    4   13   14
10   2    0    2
3    6    3    0
```

```
>> u=[0 2 3 0]
```

```
u =
```

```
0    2    3    0
```

```
>> K=[M; u]
```

```
K =
```

```
1    2   11   12
3    4   13   14
10   2    0    2
3    6    3    0
0    2    3    0
```

Chú ý các phần tử của ma trận có thể gọi đến với hai chỉ số (hàng, cột) hoặc chỉ cần một chỉ số. Khi sử dụng một chỉ số nghĩa là ma trận được lưu trữ như một véc-tơ do sự nối tiếp các cột với nhau.

Như thế, với ma trận m hàng n cột thì phần tử $A(i, j)$ chính là $A(i + m(j - 1))$. Ví dụ với ma trận vuông A cỡ 4×4 , thì

$$A_{(4,3)} \text{ chính là } A_{(12)} = A_{(4+4(3-1))}.$$

Các phép tính sau là tương đương

<code>sum(sum(A))</code>	<code>sum(A(:))</code>
<code>A(end,end)</code>	<code>A(end)</code>

Các phép nhân và chia phần tử hai ma trận cùng cỡ

Ngoài các phép tính như cộng, trừ và nhân thông thường hai ma trận có cỡ thích hợp, trong Matlab còn có các phép tính nhân phần tử và chia phần tử tương ứng của hai ma trận cùng cỡ. Các phép tính này còn được gọi là phép nhân mảng và phép chia mảng. Hai ma trận nhân phần tử được với nhau nếu chúng cùng cỡ, và thực hiện bởi dấu nhân phần tử, dấu chấm viết liền dấu nhân (`.*`). Dấu nhân phần tử này cũng áp dụng đối với hai véc-tơ cùng cỡ. Ví dụ

```
>> A = [12 3; -1 6]; B = [4 2; 9 1];
```

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

```
C =
```

```
48    6
-9    6
```

Có thể nhận thấy rằng, nhân phần tử hai ma trận sẽ cho một ma trận cùng cỡ được định nghĩa như sau

$$\mathbf{C} = \mathbf{A} .* \mathbf{B} \quad \text{với} \quad c_{i,j} = a_{i,j} b_{i,j}, \quad i = 1, \dots, m; j = 1, \dots, n.$$

Ta cũng có thể thực hiện các phép chia trái và chia phải của hai mảng cùng cỡ, phép tính này có thể hiểu là phép chia phần tử của hai ma trận cùng cỡ.

- phép chia trái phần tử ma trận **A** cho ma trận **B** ta viết là **C = A./B**
- phép chia phải phần tử ma trận **A** cho ma trận **B** ta viết là **C = A.\B** (tức là **B./A**)

Ví dụ

```
>> A = [2 4 6 8];    B = [2 2 3 1];
>> C = A./B
C =
    1    2    2    8
>> C = A.\B
C =
    1.0000    0.5000    0.5000    0.1250
```

Ngoài phép chia phần tử hai ma trận cũng cỡ, các phép tính nhân phần tử và mũ phần tử đều áp dụng được. Ví dụ

```
>> A = [2 2; 3 1];    B = [2 4; -1 6]
B =
     2     4
    -1     6
>> K = B.^2
K =
     4    16
     1    36
>> C = A.*B
C =
     4     8
    -3     6
```

Tính định thức và giải hệ phương trình đại số tuyến tính

Định thức của ma trận vuông là một số. Đối với ma trận vuông cỡ 2×2

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},$$

định thức của nó được xác định theo biểu thức

$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$

Để tính định thức ma trận **A** trong Matab, ta viết **det(A)**. Ví dụ đối với ma trận **A** cỡ 2×2

```
>> A = [1 3; 4 5];
>> det(A)
ans = -7
```

Hay đối với ma trận **B** cỡ 4×4 :

```
>> B = [3 -1 2 4; 0 2 1 8; -9 17 11 3; 1 2 3 -3];  
>> det(B)  
ans = -533
```

Như đã biết trong giáo trình đại số, việc tính định thức của ma trận thật phức tạp với nhiều phép tính. Đặc biệt là khi cỡ của ma trận lớn, công việc này nếu thực hiện bằng tay sẽ mất rất nhiều thời gian và sai sót là khó tránh khỏi. Nhưng việc này đối với Matlab thật đơn giản, như đã thấy trong hai ví dụ trên. Định thức của ma trận có thể được sử dụng để tìm nghiệm của hệ phương trình đại số tuyến tính, nếu hệ này có nghiệm. Xét hệ phương trình đại số tuyến tính sau:

$$5x + 2y - 9z = 44$$

$$-9x - 2y + 2z = 11$$

$$6x + 7y + 3z = 44$$

hay có thể viết lại dạng ma trận

$$\mathbf{Ax} = \mathbf{b},$$

với

$$\mathbf{A} = \begin{bmatrix} 5 & 2 & -9 \\ -9 & -2 & 2 \\ 6 & 7 & 3 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 44 \\ 11 \\ 44 \end{bmatrix}$$

Để tìm nghiệm của hệ trên, ta có thể sử dụng hai bước. Trước hết tìm định thức của ma trận hệ số **A**

```
>> A = [5 2 -9; -9 -3 2; 6 7 3]  
>> det(A)  
ans = 368
```

Ta thấy $\det(\mathbf{A})$ khác không, như vậy hệ phương trình trên có nghiệm duy nhất. Matlab cho phép tính được nghiệm một cách nhanh chóng nhờ phép tính chia trái $\mathbf{x} = \mathbf{A} \setminus \mathbf{b}$.

```
>> b = [44;11;5];  
>> x = A\b  
x =  
    -5.1250  
     7.6902  
    -6.0272
```

Tìm hạng của ma trận

Hạng của ma trận là số lớn nhất các hàng hoặc các cột độc lập tuyến tính. Nếu một véc tơ là độc lập tuyến tính đối với các véc tơ khác nghĩa là nó không thể biểu diễn là tổ hợp tuyến tính của các véc tơ đó. Ví dụ với ba véc tơ

$$\mathbf{u} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 3 \\ -4 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} 5 \\ -6 \end{bmatrix}$$

Ta thấy rằng $2\mathbf{u} + \mathbf{v} = \mathbf{w}$, như thế véc tơ \mathbf{w} phụ thuộc tuyến tính vào hai véc tơ \mathbf{u} và \mathbf{v} . Tức là ba véc tơ trên không độc lập tuyến tính. Một ví dụ khác, ba véc tơ sau

$$\mathbf{u} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ 7 \end{bmatrix}$$

là độc lập tuyến tính, bởi vì không một véc tơ nào có thể biểu diễn là tổ hợp tuyến tính của hai véc tơ còn lại.

Xét ma trận

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 2 & 0 & 4 \end{bmatrix}$$

ta thấy hàng thứ hai của ma trận bằng 2 lần hàng thứ nhất, tức là trong ma trận này chỉ có một hàng duy nhất. Do đó ma trận này có hạng bằng 1. Trong Matlab hạng của ma trận \mathbf{A} được xác định bằng lệnh `rank(A)`. Ví dụ

```
>> A = [0 1 0 2; 0 2 0 4];  
>> rank(A)  
ans = 1
```

Hay một ví dụ khác

$$\mathbf{B} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 0 & 9 \\ -1 & 2 & -3 \end{bmatrix}$$

Ta thấy cột thứ ba gấp ba lần cột thứ nhất. Do đó ba véc tơ cột của ma trận \mathbf{B} là phụ thuộc tuyến tính. Ta sẽ kiểm tra xem liệu cột thứ hai có thể biểu diễn thông qua cột thứ nhất không? Nếu có tức là tồn tại số $\alpha \neq 0$, sao cho

$$\begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix}$$

Rõ ràng là không thể tìm được số $\alpha \neq 0$. Như thế ta có thể kết luận rằng, trong ba vectơ cột của **B** có hai vectơ độc lập tuyến tính và hạng của ma trận **B** là 2. Hãy kiểm tra bằng Matlab:

```
>> B = [1 2 3; 3 0 9; -1 2 -3];
>> rank(B)
ans = 2
```

Bây giờ ta xem xét hệ phương trình đại số tuyến tính với m phương trình và n ẩn, viết ở dạng ma trận như sau

$$\mathbf{Ax} = \mathbf{b},$$

với ma trận **A** cỡ $m \times n$, vectơ **x** cỡ $n \times 1$ và **b** cỡ $m \times 1$.

Xét ma trận mở rộng bằng cách thêm vectơ cột **b** vào thành cột thứ $n + 1$ của ma trận **A**, $[\mathbf{A} \ \mathbf{b}]$. Hệ có nghiệm nếu $\text{rank}(\mathbf{A}) = \text{rank}([\mathbf{A} \ \mathbf{b}]) = r$. Nếu hạng $r = n$ hệ có nghiệm duy nhất, nếu hạng $r < n$, hệ có vô số nghiệm và có thể biểu diễn r ẩn là tổ hợp tuyến tính của $n - r$ ẩn còn lại.

Để minh họa, ta xét hệ sau

$$\begin{aligned} x - 2y + z &= 12 \\ 3x + 4y + 5z &= 20 \\ -2x + y + 7z &= 11 \end{aligned}$$

Ma trận hệ số **A** và vectơ **b**

$$\mathbf{A} = \begin{bmatrix} 1 & -2 & 1 \\ 3 & 4 & 5 \\ -2 & 1 & 7 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 12 \\ 20 \\ 11 \end{bmatrix}$$

Ma trận mở rộng

$$[\mathbf{A} \ \mathbf{b}] = \begin{bmatrix} 1 & -2 & 1 & 12 \\ 3 & 4 & 5 & 20 \\ -2 & 1 & 7 & 11 \end{bmatrix}$$

Bước một là ta nhập ma trận hệ số **A** và vectơ **b** vào trong Matlab:

```
>> A = [1 -2 1; 3 4 5; -2 1 7];
>> b = [12; 20; 11];
```

Sau đó tạo ma trận mở rộng

```
>> C = [A b]
C =
     1     -2      1     12
     3      4      5     20
    -2      1      7     11
```

```

3 4 5 20
-2 1 7 11

```

Và kiểm tra hạng của hai ma trận **A** và **C**

```

>> rank(A)
ans = 3
>> rank(C)
ans = 3

```

Vì hạng của hai ma trận **A** và **C** bằng nhau và bằng 3, nên hệ phương trình đã cho có nghiệm duy nhất. Nghiệm này tìm được nhờ phép chia trái

```

>> x = A\b
x =
4.3958
-2.2292
3.1458

```

Tìm ma trận nghịch đảo và ma trận tựa nghịch đảo

Nghịch đảo của ma trận vuông **A** là một ma trận, được ký hiệu là \mathbf{A}^{-1} , thỏa mãn quan hệ sau

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{E}, \text{ với } \mathbf{E} \text{ là ma trận đơn vị cùng cỡ.}$$

Xét phương trình ma trận sau

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

Nếu tồn tại ma trận nghịch đảo của **A**, thì nghiệm của phương trình trên được viết là

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}.$$

Trong thực tế không dễ dàng khi tìm ma trận nghịch đảo bằng tay với cây bút chì. So với các phép tính cộng, trừ, nhân ma trận thì việc tìm ma trận nghịch đảo khó hơn rất nhiều. Tuy nhiên, với Matlab thì việc này không khó khăn gì, bạn chỉ cần khai báo ma trận **A** và gọi lệnh **inv(A)**, Matlab sẽ tính và đưa ra cho ma trận nghịch đảo của ma trận **A**. Nhưng cần nhớ rằng không phải ma trận vuông **A** nào cũng có ma trận nghịch đảo. Điều kiện để tồn tại ma trận nghịch đảo là định thức của nó phải khác 0, $\det(A) \neq 0$, tức **A** là ma trận chính quy. Nếu $\det(A) = 0$, ta nói **A** là ma trận suy biến hay kỳ dị (singular matrix).

Ta bắt đầu với ví dụ đơn giản, tìm nghịch đảo của ma trận cỡ 2×2

$$\mathbf{A} = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$$

Trước hết kiểm tra xem đây có là ma trận chính quy:


```
>> A = [2 3; 4 5]
```

```
A =
     2     3
     4     5
```

```
>> det(A)
```

```
ans = -2
```

Vì $\det(A) \neq 0$, nên ta có thể tìm được ma trận nghịch đảo của nó.

```
>> inv(A)
```

```
ans =
    -2.5000    1.5000
     2.0000   -1.0000
```

Để khẳng định bạn có thể kiểm tra lại bằng tay

$$\mathbf{A}\mathbf{A}^{-1} = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} -2.5 & 1.5 \\ 2.0 & -1.0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Tiếp theo là tìm ma trận nghịch đảo của ma trận cỡ 4×4 :

```
>> S = [1 0 -1 2; 4 -2 -3 1; 0 2 -1 1; 0 0 9 8];
```

```
>> det(S)
```

```
ans = -108
```

Vì $\det(S) \neq 0$, nên ta có thể tìm được ma trận nghịch đảo của nó.

```
>> iS=inv(S)
```

```
iS =
    -0.9259    0.4815    0.4815    0.1111
    -0.6296    0.1574    0.6574    0.0556
    -0.5926    0.1481    0.1481    0.1111
     0.6667   -0.1667   -0.1667         0
```

Để kiểm tra lại ta tính tích hai ma trận:

```
>> iS*S
```

```
ans =
    1.0000         0         0         0
         0    1.0000         0    0.0000
         0         0    1.0000         0
         0         0    0.0000    1.0000
```

```
>> S*iS
```

```
ans =
    1.0000         0         0         0
    0.0000    1.0000   -0.0000         0
    0.0000   -0.0000    1.0000         0
         0         0         0    1.0000
```

Trong các ma trận tích này có những phần tử ngoài đường chéo không hoàn toàn bằng 0 và những phần tử trên đường chéo không hẳn bằng 1. Lý do là khi tính toán số không phải lúc nào cũng đạt được độ chính xác tuyệt đối, các kết quả chỉ có thể đạt được độ chính xác nhất định. Nếu sử dụng *format long e* ta sẽ thấy rõ thêm các phần tử này.

```
>> format long e
>> S*iS
ans =
Columns 1 through 2
    1.0000000000000000e+000                0
    1.110223024625157e-016    1.0000000000000000e+000
    1.110223024625157e-016   -2.775557561562891e-017
                                0                0
Columns 3 through 4
                                0                0
   -1.387778780781446e-016                0
    1.0000000000000000e+000                0
                                0    1.0000000000000000e+000
```

Có thể thấy rằng sai số nhận được ở đây rất bé, cỡ 10^{-16} .

Một trong những lý do tìm ma trận nghịch đảo là việc giải hệ phương trình đại số tuyến tính. Các ví dụ dưới đây sẽ chỉ ra một cách để nhận được nghiệm của phương trình $\mathbf{Ax} = \mathbf{b}$. Xét hệ phương trình

$$3x + 2y = 5$$

$$6x + 2y = 2$$

Từ đây ta biết được ma trận hệ số \mathbf{A} và vectơ vế phải \mathbf{b} :

$$\mathbf{A} = \begin{bmatrix} 3 & -2 \\ 6 & -2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

Ta sẽ nhập chúng vào Matlab:

```
>> A = [3 -2; 6 -2];
>> b = [5; 2];
```

Trước hết kiểm tra xem ma trận hệ số \mathbf{A} có phải là ma trận chính qui:

```
>> det(A)
ans = 6
```

Do đó tồn tại ma trận nghịch đảo, và tính được nghiệm theo:

```
>> x=inv(A)*b
x =
   -1.0000
   -4.0000
```

Cách mà ta vừa sử dụng để nhận được nghiệm của hệ phương trình đại số tuyến tính, chỉ áp dụng được khi ma trận hệ số là vuông và không suy biến. Trong hệ này ta thấy số phương trình bằng số ẩn cần tìm. Trường hợp số phương trình ít hơn số ẩn ta gọi là hệ hụt (thiếu) phương trình (underdetermined). Khi đó hệ phương trình có thể có vô số nghiệm thỏa mãn. Bởi vì ta chỉ có thể xác định được một số biến trong các ẩn đó, số còn lại có thể nhận các giá trị tùy ý. Ví dụ như hệ hai phương trình dưới đây với ba ẩn

$$x + 2y - z = 3$$

$$5y + z = 0$$

Sau khi biến đổi ta nhận được

$$x = 3 - 7y$$

$$z = -5y$$

Giá trị của hai biến x và z phụ thuộc vào biến y . Với mỗi giá trị của y ta sẽ nhận được các giá trị tương ứng của x và z . Hệ có vô số nghiệm.

Nếu viết hệ trên ở dạng ma trận $\mathbf{Ax} = \mathbf{b}$ với

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 5 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix}$$

Ta thấy ngay $\det(\mathbf{A}) = 0$. Để biết được nghiệm của hệ này có tồn tại không ta cần so sánh hạng của ma trận \mathbf{A} và ma trận mở rộng $\mathbf{C} = [\mathbf{A} \ \mathbf{b}]$, nếu chúng bằng nhau thì tồn tại nghiệm, và có thể sử dụng phép chia trái. Ví dụ cần giải hệ phương trình

$$3x + 2y - z = 7$$

$$4y + z = 2$$

ta thực hiện:

```
>> A= [3 2 -1; 0 4 1]; b = [7;2];
>> C = [A b]
C =
     3     2    -1     7
     0     4     1     2
>> rank(A)
ans = 2
>> rank(C)
ans = 2
>> x=A\b
x =
    2.0000
    0.5000
     0
```

Trong trường hợp này Matlab đã cho $z = 0$, thực tế z có thể nhận giá trị bất kỳ. Với một hệ phương trình có vô số nghiệm, thì ta có thể tìm được một nghiệm có độ dài nhỏ nhất chẳng hạn. Công thức của nghiệm có độ dài nhỏ nhất như sau

$$\mathbf{x} = \mathbf{A}^\dagger \mathbf{b} = \mathbf{A}^T [\mathbf{A} \mathbf{A}^T]^{-1} \mathbf{b}.$$

Ma trận $\mathbf{A}^\dagger = \mathbf{A}^T [\mathbf{A} \mathbf{A}^T]^{-1}$ được gọi là ma trận tựa nghịch đảo trái của ma trận \mathbf{A} . Trong Matlab đã sẵn có một hàm để tính ma trận tựa nghịch đảo này, đó là **pinv(A)**. Như thế nghiệm có chuẩn nhỏ nhất (tối ưu) của hệ trên được tính là

```
>> A= [3 2 -1; 0 4 1]; b = [7; 2];
>> x=pinv(A)*b
x =
    1.6667
    0.6667
   -0.6667
```

Hãy thử lại với

```
>> x=A'*inv(A*A')*b
x =
    1.6667
    0.6667
   -0.6667
```

Khi tính toán với các đại lượng, ta cần phải lưu ý đến thứ tự ưu tiên các phép toán, ví dụ khi giải hệ phương trình đại số tuyến tính

$$\mathbf{Kx} = 0.5 \mathbf{b} \Rightarrow \mathbf{x} = 0.5 \mathbf{K}^{-1} \mathbf{b}$$

Sau đây sẽ tính trong Matlab với các kiểu viết khác nhau

```
>> K=[2 2 3; 4 5 6; 7 8 9]; b=[2 3 4]';
>> x=0.5*K\b % cho ket qua sai
x =
     0
   -2.0000
    2.6667

>> x=0.5*(K\b) % cho ket qua dung
x =
     0
   -0.5000
    0.6667
>> x=0.5*inv(K)*b
x =
    0.0000
   -0.5000
    0.6667
```

```
>> x=K\b*0.5
x =
     0
-0.5000
 0.6667
```

Trị riêng và vectơ riêng của ma trận vuông

Cho \mathbf{A} là một ma trận vuông cấp n . Số λ được gọi là trị riêng và vectơ khác không \mathbf{x} là vectơ riêng của \mathbf{A} nếu chúng thỏa mãn điều kiện :

$$\mathbf{Ax} = \lambda\mathbf{x} \text{ hay } (\mathbf{A} - \lambda\mathbf{E})\mathbf{x} = \mathbf{0}$$

với \mathbf{E} là ma trận đơn vị.

Với mỗi giá trị λ_i , ta sẽ có vô số các vectơ \mathbf{x}_i thỏa mãn. Các vectơ riêng cùng tương ứng với một λ_i rõ ràng là phụ thuộc tuyến tính và chỉ khác nhau một hằng số α . Do đó ta có thể chọn một vectơ duy nhất làm cơ sở (ví dụ được chuẩn hoá theo chuẩn Euclide để có chuẩn bằng 1...). Tập hợp n vectơ riêng ứng với n trị riêng khác nhau tạo thành một hệ vectơ độc lập tuyến tính. Ma trận gồm các cột là các vectơ riêng của ma trận \mathbf{A} gọi là ma trận dạng riêng của \mathbf{A} (Modal matrix).

Nếu có hai ma trận vuông \mathbf{A} và \mathbf{B} cùng cấp n . Số λ và vectơ khác không \mathbf{x} thỏa mãn điều kiện :

$$\mathbf{Ax} = \lambda\mathbf{Bx} \text{ hay } (\mathbf{A} - \lambda\mathbf{B})\mathbf{x} = \mathbf{0}$$

thì số λ gọi là trị riêng suy rộng của hai ma trận \mathbf{A} và \mathbf{B} , vectơ \mathbf{x} là vectơ riêng tương ứng.

Việc tìm trị riêng và vectơ riêng trong Matlab được thực hiện bằng lệnh **eig()**:

Lệnh eig tính toán trị riêng và vectơ riêng của ma trận vuông	
<code>l = eig(A);</code>	cho một vectơ l chứa các trị riêng của ma trận vuông A .
<code>[V, D] = eig(A);</code>	cho ta ma trận chéo D chứa các trị riêng của A , ma trận V có các cột là các vectơ riêng của A thỏa mãn AV = VD .
<code>d = eig(A,B);</code>	cho ta vectơ chứa các trị riêng suy rộng của các ma trận vuông A và B .
<code>[V, D] = eig(A,B);</code>	cho ta ma trận chéo D chứa các trị riêng suy rộng của hai ma trận A và B , ma trận V có các cột là các vectơ riêng thỏa mãn AV = BVD .
<code>eig(A,B,'chol')</code>	tương tự như <code>eig(A,B)</code> đối với ma trận đối xứng A và ma trận đối xứng xác định dương B . Phương pháp tính ở đây dựa trên khai triển Cholesky ma trận B .

Các ví dụ sau minh họa việc tìm trị riêng và vectơ riêng trong Matlab

```

>> C=[2  1  0;
      1  4 -1;
      0 -1  7];
>> eig(C)
ans =
    1.5573
    4.1233
    7.3194

>> [V, D]=eig(C)
V =
    0.9119   -0.4064   -0.0571
   -0.4037   -0.8630   -0.3037
   -0.0742   -0.3000    0.9510

D =
    1.5573         0         0
         0    4.1233         0
         0         0    7.3194

```

Phân tích ma trận vuông **A** thành tích các ma trận

Phân tích **LU**

Với mỗi ma trận vuông **A** không suy biến, $\det(A) \neq 0$, luôn phân tích được thành tích của hai ma trận tam giác

$$\mathbf{A} = \mathbf{LU},$$

với **L** là ma trận tam giác dưới có các phần tử trên đường chéo chính bằng 1, và **U** ma trận tam giác trên.

Trong trường hợp có sử dụng sự hoán vị các hàng cho nhau trong quá trình phân tích thì, các phép hoán vị sẽ được thể hiện trong ma trận **P**, thỏa mãn

$$\mathbf{PA} = \mathbf{LU}$$

Ma trận hoán vị có tính chất sau

$$\mathbf{P}^T \mathbf{P} = \mathbf{E} \Rightarrow \mathbf{P}^T = \mathbf{P}^{-1}$$

Ví dụ

```

>> A = [-1  2  0; 4  1  8; 2  7  1]
A =
    -1     2     0
     4     1     8
     2     7     1

>> [P, L, U]=lu(A)

P =
    1.0000         0         0
    0.5000    1.0000         0
   -0.2500    0.3462    1.0000

L =
    4.0000    1.0000    8.0000
         0    6.5000   -3.0000
         0         0    3.0385

```

$$U = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Ý nghĩa của phân tích LU có thể thấy trong việc giải hệ $\mathbf{Ax} = \mathbf{b}$. Sau khi phân tích LU ta nhận được

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}$$

Đặt $\mathbf{y} = \mathbf{Ux}$, suy ra $\mathbf{Ly} = \mathbf{b}$. Như thế thay vì giải hệ $\mathbf{Ax} = \mathbf{b}$, ta giải hai hệ

$$\mathbf{Ly} = \mathbf{b} \text{ và } \mathbf{Ux} = \mathbf{y}$$

Do \mathbf{L} và \mathbf{U} là các ma trận tam giác nên việc giải hai hệ này khá đơn giản.

Với \mathbf{L} là ma trận tam giác dưới nên dễ dàng nhận được nghiệm \mathbf{y} . Sau đó giải tìm \mathbf{x} . Ví dụ cần giải hệ phương trình

$$3x + 2y - 9z = -65$$

$$-9x + 5y + 2z = 16$$

$$6x + 7y + 3z = 5$$

Nhập hệ vào Matlab, phân tích LU và giải tìm nghiệm:

```
>> A = [3 2 -9; -9 -5 2; 6 7 3]; b = [-65; 16; 5];
>> [L, U] = lu(A)
L =
   -0.3333    0.0909    1.0000
    1.0000         0         0
   -0.6667    1.0000         0
U =
   -9.0000   -5.0000    2.0000
         0    3.6667    4.3333
         0         0   -8.7273
>> x = U \ (L \ b)
x =
    2.0000
   -4.0000
    7.0000
```

Phân tích Cholesky

Ma trận vuông \mathbf{A} đối xứng xác định dương luôn phân tích được thành tích

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T,$$

với \mathbf{L} là ma trận tam giác dưới.

Ma trận vuông \mathbf{A} xác định dương nếu thỏa mãn $\mathbf{x}^T \mathbf{Ax} > 0$, $\forall \mathbf{x} \neq 0$.

Ví dụ

```
>> M = [2 3 4; 3 5 6; 4 6 9];
>> L=chol(M)
L =
    1.4142    2.1213    2.8284
         0    0.7071    0.0000
         0         0    1.0000

>> L'*L-M
ans =
    1.0e-015 *
         0    0    0
         0    0    0
         0    0    0
```

Phân tích QR

Phân tích QR là phép phân tích một ma trận vuông hoặc chữ nhật thành tích của ma trận trực giao và một ma trận tam giác trên:

$$\mathbf{A} = \mathbf{QR}$$

với $\mathbf{Q}^T \mathbf{Q} = \mathbf{E}$, \mathbf{R} là ma trận tam giác trên. Ví dụ

```
>> A = [2 3 4 5; 3 5 6 8; 4 6 9 10];
>> [Q, R] = qr(A)

Q =
   -0.3714    0.2491   -0.8944
   -0.5571   -0.8305   -0.0000
   -0.7428    0.4983    0.4472

R =
   -5.3852   -8.3563  -11.5131  -13.7415
         0   -0.4152    0.4983   -0.4152
         0         0    0.4472   -0.0000

>> Q*R - A
ans =
    1.0e-014 *
         0   -0.1332   -0.1776         0
    0.0444   -0.0888   -0.0888         0
         0   -0.0888         0    0.1776
```



```
>> Q*Q'
ans =

    1.0000    -0.0000    -0.0000
   -0.0000     1.0000    -0.0000
   -0.0000    -0.0000     1.0000

>> inv(Q)-Q'
ans =
    1.0e-015 *

         0         0         0
   -0.0833     0.1110    -0.0555
     0.1110    -0.1066    -0.2220
```

Phân tích SVD (Singular Value Decomposition)

Phân tích SVD là biểu diễn ma trận \mathbf{A} cỡ $m \times n$ dạng tích các ma trận

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

với \mathbf{U} là ma trận trực giao cỡ $m \times m$, \mathbf{V} là ma trận trực giao cỡ $n \times n$, và \mathbf{S} là ma trận thực cỡ $m \times n$ có dạng đường chéo chứa các giá trị kỳ dị (singular values) của \mathbf{A} theo trật tự từ lớn đến bé. Đó là các giá trị khai căn của các trị riêng của ma trận $\mathbf{A}^T \mathbf{A}$.

Ví dụ

```
>> A = [1 2; 2 3; 3 5];
>> [U,S,V] = svd(A)
U =
   -0.3092    0.7557   -0.5774
   -0.4998   -0.6456   -0.5774
   -0.8090    0.1100    0.5774
S =
    7.2071         0
         0    0.2403
         0         0
V =
   -0.5184   -0.8552
   -0.8552    0.5184
>> err = U*S*V'-A
err =
    1.0e-015 *

         0    0.4441
    0.8882    0.4441
    0.8882         0
```

2.4 Bài tập thực hành

1. Tìm độ dài của vectơ

$$\mathbf{a} = [-1 \ 7 \ 3 \ 2].$$

2. Tìm độ dài của vectơ phức

$$\mathbf{b} = [-1 + i \ 7i \ 3 \ -2 - 2i].$$

3. Nhập các số 1, 2, 3 để tạo ra một vectơ hàng và một vectơ cột.

4. Cho hai vectơ cột $\mathbf{a} = [1; 2; 3]$ và $\mathbf{b} = [4; 5; 6]$.

Hãy tìm tích phần tử hai vectơ.

5. Lệnh nào có thể tạo ra một ma trận cỡ 5×5 có các phần tử trên đường chéo chính bằng 1 còn lại là 0?

6. Cho hai ma trận

$$\mathbf{A} = \begin{bmatrix} 8 & 7 & 11 \\ 6 & 5 & 1 \\ 0 & 2 & 8 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 6 & 4 \\ 2 & 2 & 2 \end{bmatrix}$$

Hãy tính tích phần tử (tích mảng) và tích hai ma trận \mathbf{A} và \mathbf{B} .

7. Cho ma trận

$$\mathbf{A} = \begin{bmatrix} 8 & 7 & 11 \\ 6 & 5 & 1 \\ 3 & 2 & 8 \end{bmatrix}, \quad \text{từ A hãy tạo ra ma trận } \mathbf{B} = \begin{bmatrix} 3 & 2 & 8 \\ 3 & 2 & 8 \\ 6 & 5 & 1 \end{bmatrix}.$$

8. Tìm nghiệm của hệ phương trình :

$$x + 2y + 3z = 12$$

$$-4x + y + 2z = 13$$

$$9y - 8z = -1$$

Định thức của ma trận hệ số bằng bao nhiêu?

9. Hãy xem xét hệ sau có nghiệm duy nhất không? tại sao

$$x - 2y + 3z = 1$$

$$x + 4y + 3z = 2$$

$$2x + 8y + z = 3$$

10. Sử dụng phân tích LU tìm nghiệm của hệ sau:

$$x + 7y - 9z = 12$$

$$2x - y + 4z = 16$$

$$x + y - 7z = 16$$

Chương 3.

Lập trình trong Matlab

Trong các chương trước đã trình bày việc sử dụng các hàm, lệnh định nghĩa sẵn của Matlab để giải các bài toán đơn giản. Ngoài các lệnh, hàm định sẵn có này, Matlab còn là môi trường cho phép lập trình giải các bài toán phức tạp, hoặc tạo ra các hàm tiện lợi cho người sử dụng. Trong chương này sẽ trình bày khả năng lập trình trong Matlab.

3.1 Các kiểu dữ liệu

Trong Matlab có nhiều kiểu dữ liệu, sau đây chỉ trình bày một số kiểu liên quan đến dạng thể hiện và lưu trữ. Để có thể nhận được sự trợ giúp trực tiếp từ Matlab, bạn cần tận dụng câu lệnh *help*.

```
==> help class, help strfun, help struct
```

Kiểu véc tơ và ma trận

Biểu diễn số các ma trận hai hay nhiều chiều cũng như các véc tơ trong chương 2 là một kiểu dữ liệu đặc biệt. Mỗi phần tử của véc tơ hay ma trận theo chuẩn cần ô nhớ 8 Byte (class double) hoặc 4 Byte (class single). Các đại lượng phức cần một ô nhớ gấp đôi, phần thực và ảo được ghi riêng rẽ.

Ví dụ đối với mảng 3 chiều (3D), với cấu trúc K(hàng, cột, chiều sâu)

```
>> K(1,1,1)=2;      K(2,2,2)=4;      K(3,3,3)=5;
>> K(:, :, 3) % lop 3
ans =
     0     0     0
     0     0     0
     0     0     5
K(:, :, 2)      % lop 2
ans =
     0     0     0
```

```

0      4      0
0      0      0
K(:, :, 1)    % lop 1
ans =
2      0      0
0      0      0
0      0      0

```

Chuỗi ký tự (ký tự, xâu - string)

Chuỗi ký tự được đặt trong cặp dấu nháy đơn trên

```
>> 'Day la chuoi ky tu'
```

Mỗi phần tử của chuỗi ký tự (Ma trận ký tự) cần ô nhớ 2 Byte.

Kiểu ô, mảng (Cell-Array)

Các dữ liệu của các kiểu khác nhau, ví dụ chuỗi ký tự, các ma trận có cỡ khác nhau, có thể được sử dụng như các ô của một mảng. Các phần tử ô được gọi đến thông qua chỉ số của nó. Các phần tử của một ô được đặt trong dấu ngoặc nhọn {...}. Ví dụ

```

>> A(1,1)={ [1 2 3; 4 2 6; 1 7 9] };
>> A(1,2)={'Ma tran thu'};
>> A(2,1)={3+8i};
>> A(2,2)={0:pi/20:2*pi};    %hoac
>> A{1,1}=[1 2 3; 4 2 6; 1 7 9];
>> % goi den cac phan tu
>> A(1,1)
ans =
[3x3 double]
>> A{1,1}
ans =
1      2      3
4      2      6
1      7      9
>> A{1,2} % o (1,2)
ans = Ma tran thu
>> A(1,2)
ans = 'Ma tran thu'

```

Kiểu cấu trúc

Cấu trúc được sử dụng để làm việc với các kiểu dữ liệu khác nhau. Tên của một cấu trúc gồm hai phần, tên cấu trúc trước dấu chấm (.) và tên trường trong cấu

trúc sau dấu chấm, (struct_name.field_name). Các phần tử của cấu trúc được gọi đến qua tên và chỉ số.

Cú pháp:

```
% tạo một cấu trúc
structur=struct('name_1', value1, 'name_2', value2,.. )
% truy cập phần tử của cấu trúc
structur.name
hoặc
structur.name_1 = value_1;
structur.name_2 = value_2;
```

Ví dụ

```
>> A=[1 2 3; 4 2 6; 1 7 9];
>> my_structur=struct('data', A, 'dimension',[3 3])
my_structur =
    data: [3x3 double]
 dimension: [3 3]
>> my_structur.dimension
ans =
     3     3
>> my_structur.data
ans =
     1     2     3
     4     2     6
     1     7     9
>> my_structur(2).data=inv(A) % mở rộng trường data
my_structur =
1x2 struct array with fields:
    data
 dimension
>> my_structur.data % in ra ma tran A va inv(A)
ans =
     1     2     3
     4     2     6
     1     7     9
ans =
     4.0000    -0.5000    -1.0000
     5.0000    -1.0000    -1.0000
    -4.3333     0.8333     1.0000

>> % gọi đến các phần tử
>> my_structur(1).data(1,2) % phần tử A(1,2)
ans =
```

```

2
>> my_structur(2).data(1,2) % phan tu (1,2) cua inv(A)
ans =
-0.5000
>> my_structur(2).data(3,2) % phan tu (3,2) cua inv(A)
ans =
0.8333

```

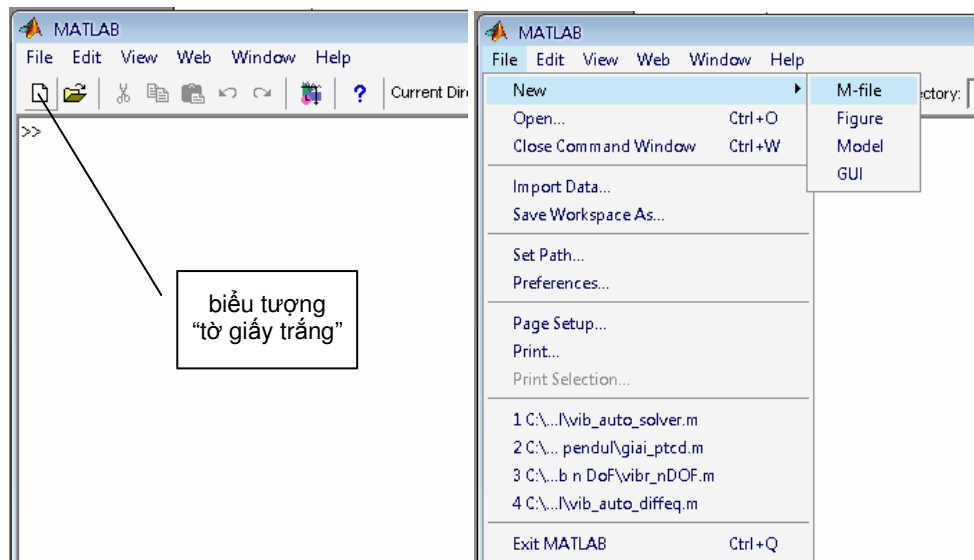
3.2 Soạn thảo Script file trong Matlab

Matlab là một ngôn ngữ tương tác, nghĩa là các lệnh được đánh tại dấu nhắc '>>' được thực hiện ngay sau khi kết thúc dòng lệnh bằng phím 'Enter'. Tuy nhiên thật đơn điệu khi đánh (gõ) một chuỗi lệnh dài mỗi khi sử dụng Matlab thực hiện công việc. Để tránh việc này trong Matlab có hai biện pháp mở rộng công năng: scripts và functions (dùng tệp văn bản và các hàm). Cả hai biện pháp này sử dụng loại tệp có tên là m-file (file có phần mở rộng là .m) được tạo ra nhờ trình soạn thảo. Ưu điểm của m-file là có thể ghi lại các dòng lệnh và có thể sửa đổi một cách dễ dàng mà không phải đánh lại toàn bộ danh sách các lệnh.

Script và Matlab-function	
function [out_1, . . .] = name(in_1, . . .)	
% Matlab-function / Hàm trong matlab	
@	Tham chiếu đến hàm function handle
nargin	Số tham số đầu vào
narout	Số tham số đầu ra
global	Định nghĩa, khai báo biến tổng thể / toàn cục
persistent	Định nghĩa, khai báo biến persistent
%, %{ . . . %}	Các dòng chú thích, diễn giải
...	Xuống dòng khi dòng lệnh quá dài
eval(str)	Xác định giá trị một xâu
feval(F, in_1, ..., in_n)	Xác định giá trị một hàm
inline('function','t',...)	Hàm ngay trong một script
clear function_name	Xóa các hàm
== > help function, help function_handle, help funfun	

Các thủ tục vào / ra	
<code>disp(string)</code>	Hiển thị một chuỗi ký tự, một xâu
<code>disp(variable)</code>	Hiển thị dữ liệu không được định dạng
<code>num2str(variable)</code>	Chuyển đổi một biến thực thành xâu ký tự
<code>int2str(variable)</code>	Chuyển đổi một biến nguyên thành xâu ký tự
<code>variable = input(string)</code>	Đưa vào một biến
<code>string = input(string, 's')</code>	Đưa vào một xâu ký tự
<code>pause</code>	Dừng cho đến khi gõ vào phím bất kỳ
<code>pause(n)</code>	Dừng lại một khoảng thời gian n giây
<code>fprintf(format, variable)</code>	In ra một dữ liệu có định dạng
<code>sprintf(format, variable)</code>	In ra một ký tự, một xâu
<code>tic operations toc</code>	Cho ta thời gian tính toán tic → toc
== > help function, help function_handle, help funfun	

Scripts



Hình 3-1.

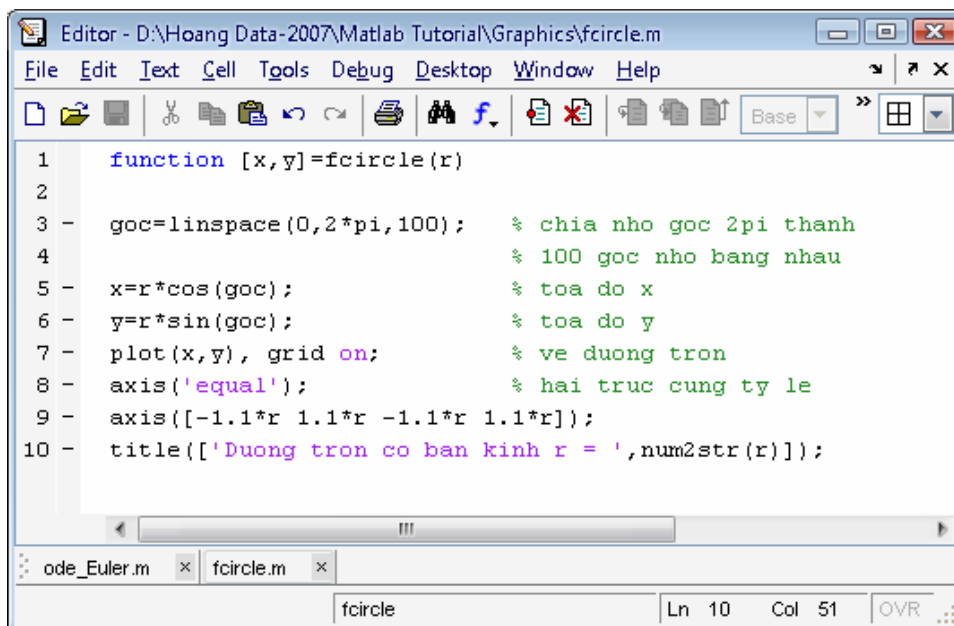
Matlab Script files là một chuỗi các lệnh được gõ vào trong cửa sổ soạn thảo và được ghi vào tệp có phần mở rộng là m (filename.m, hay được gọi là m-file). Để tạo m-file, trong menu chính bạn chọn New ==> M-file hoặc bạn nháy chuột vào biểu tượng “tờ giấy trắng” trên thanh công cụ (hình 3-1).

Việc chạy m-file tương đương với việc đánh toàn bộ các dòng lệnh trên cửa sổ lệnh tại dấu nhắc ‘>>’ của Matlab. Các biến sử dụng trong m-file được đặt vào trong không gian làm việc của Matlab. Không gian làm việc này là trống khi khởi động Matlab, và nó sẽ chứa tất cả các biến được định nghĩa trong phiên làm việc. Muốn xóa tất cả các biến ta sử dụng lệnh.

```
>> clear all
```

Một script-file có thể sử dụng các biến trong môi trường làm việc và đưa ra môi trường làm việc (workspace) các dữ liệu tạo ra. Các dữ liệu này tồn tại trong quá trình chạy chương trình và nó có thể được sử dụng cho các công việc xử lý tiếp theo và có thể hiển thị chúng bằng đồ thị chẳng hạn.

Script-file không chứa các phần khai báo và không bị giới hạn bởi begin / end. Các chú thích và diễn giải trong file được viết từng dòng một sau dấu %. Đối với MATLAB 7 ta có thể soạn một khối gồm nhiều dòng bắt đầu bằng %{ và kết thúc bởi %}. Để ngắt dòng trong một biểu thức hay một dòng lệnh ta sử dụng ba chấm liên tiếp ở cuối dòng, Matlab sẽ hiểu dòng dưới là nối tiếp của dòng trên.



Hình 3-2. Cửa sổ soạn thảo script – file, MATLAB 7

Việc gọi hay chạy một chương trình trong m-file đã được soạn thảo trước có thể được thực hiện bằng cách: trong cửa sổ lệnh sau dấu nhắc >> ta gõ tên m-file đó, cần nhớ là không có phần mở rộng (.m). Hoặc ta có thể chạy chương trình ngay trên cửa sổ soạn thảo bằng cách ấn F5, hoặc dùng chuột chọn Debug --> Run trên thanh Menu.

Dưới đây là một Script file thực hiện việc tính và đưa thể tích của một hình cầu, mà tham số bán kính được người sử dụng nhập vào từ bàn phím

```
function volume
r = input('Hay cho ban kinh cua hinh cau')
vol = (4/3)*pi*r^3;
disp('The tich hinh cau nay la:')
disp(vol)
```

Dưới đây là một số dòng lệnh minh họa cho **eval**, **feval**, **inline**, hàm ẩn tên và **fprintf**, **sprintf**, **disp**.

- **eval**

```
>> x='1/y*sin(y)'; % Bieu dien ham nhu mot xau ky tu
>> y=0.725;        % gan cho bien y mot gia tri
>> a=eval(x)        %
a =
    0.9147
```

Nếu có một hàm foo.m đã được soạn trong một m-file thì sau dòng lệnh

```
>> fooHandle=@foo
```

ba lệnh sau đây sẽ cho kết quả như nhau.

```
>> feval(fooHandle,8.75)
>> fooHandle(8.75)
>> foo(8.75)
```

- **inline**

```
>> g=inline('5*a+6.5*sin(b)') % hoac
>> g=inline('5*a+6.5*sin(b)','a','b') % a, b la hai bien
g =
    Inline function:
    g(a,b) = 5*a+6.5*sin(b)
>> c=0:2
c =
     0     1     2
>> b=pi/4
b =
```

```

0.78539816339745
>> z=g(c,b)
z =
4.59619407771256    9.59619407771256    14.59619407771256

```

Hàm ẩn tên, cú pháp

```

fhandle = @(argumlist) . . . . .

>> om=5; % gan gia tri 5 cho bien om
>> f = @(y) cos(om*y); % viet bieu thuc ham
>> t=linspace(0,2*pi/om); % chuoi thoi gian
>> f(t)

```

Định dạng hiển thị

```

>> x='1/y*sin(y)'; % Bieu dien ham nhu mot xau ky tu
>> y=0.725; % gan cho bien y mot gia tri
>> a=eval(x) %
a =
0.91466957608738
>> fprintf('%3.3f',a)
0.915
>> fprintf('% s %3.3f','Gan cho bien a gia tri', a)
Gan cho bien a gia tri 0.915
>> sprintf('%3.3f',a)
ans =
0.915
>> sprintf('%3.3e',a)
ans =
9.147e-001
>> disp(['a = ', num2str(a, '%4.3f')])
a = 0.915
>> disp(['a = ', sprintf('%4.3f',a)])
a = 0.915

```

Các hàm, MATLAB - Function

Ngoài các hàm sẵn có (built-in functions), Matlab cho phép tạo ra các hàm của riêng bạn, đó là các hàm dạng m-file. Các hàm m-file này có phần mở rộng là .m tương tự như các Script file. Tuy nhiên, điều khác với các script file là các hàm m-file được sử dụng với các *tham số vào* và *tham số ra*. Việc sử dụng các hàm m-file là rất cần thiết, chúng cho phép chia nhỏ một bài toán lớn thành nhiều môđun, mỗi

môđun sẽ được giải bằng một hàm. Các hàm m-file dạng này được gọi là Matlab-function với cấu trúc tổng quát như sau:

```
function [output 1, output 2] = function_name(input1,input2)
%
% Some comments that explain what the function does go here.
%
MATLAB command 1;
MATLAB command 2;
MATLAB command 3;
```

Tên của m-file lưu trữ hàm này phải lấy đúng tên hàm đã định nghĩa trong chương trình `function_name` và nó được gọi từ dòng lệnh của MATLAB hoặc từ một m-file khác như lệnh sau:

```
>> [output1, output2] = function_name(input1, input2)
```

Khi làm việc với các hàm m-file ta cần phân biệt hai loại biến được sử dụng: Biến toàn cục (*global variable*) và biến cục bộ hay biến địa phương (*local variable*). Biến toàn cục được khai báo theo cú pháp

```
global var_1 var_2
```

Các biến toàn cục được xóa khỏi môi trường tính toán bằng lệnh *clear* [tên biến]. Biến cục bộ chỉ được sử dụng trong một hàm, biến này không được liệt kê trong cửa sổ *workspace*, và do đó không thể tác động đến được trong môi trường làm việc.

Ngoài ra, các biến **persistent** trong một hàm có thể được thống nhất với cách khai báo

```
persistent var_1 var_2
```

Trái với các đại lượng đã được khai báo bằng **global**, các đại lượng với khai báo **persistent** chỉ được biết đến trong hàm mà nó được khai báo. Do đó các hàm khác không thể tác động được đến biến này. Các biến **persistent** chỉ được xóa khi hàm chứa nó thoát ra khỏi bộ nhớ (**clear** `function_name`) hoặc khi hàm được thay đổi sau đó được ghi lại.

Sau đây sẽ trình bày hai ví dụ để làm sáng tỏ việc thao tác đối với hàm. Ví dụ đầu tiên là việc soạn một hàm để vẽ các đường tròn có các bán kính khác nhau với tên hàm (`file_name=fcircle.m`, `function_name=fcircle`). Thông số đầu vào là bán kính r , đầu ra là tọa độ x , y của các điểm trên đường tròn. Trong hàm này có sử dụng lệnh `plot` để vẽ đường tròn. Hàm này được viết như sau:

```
function [x,y]=fcircle(r)
% chia nhỏ góc 2pi thành 100 góc nhỏ bằng nhau
goc=linspace(0,2*pi,100);
x=r*cos(goc);           % toa do x
y=r*sin(goc);           % toa do y
plot(x,y), grid on;     % ve duong tron
```

```
axis('equal'); % hai trục cùng tỷ lệ
axis([-1.1*r 1.1*r -1.1*r 1.1*r]);
title(['Đường tròn có bán kính r = ', num2str(r)]);
```

Trong môi trường làm việc (Workspace) hàm này có hai cách gọi với sau

```
>> fcircle(3) % vẽ đường tròn bán kính r = 3
>> [x,y]=fcircle(4) % vẽ đường tròn bán kính r = 4, và đưa
% ra màn hình hai vector chứa tọa độ các điểm
```

Hãy thực hiện và xem kết quả !

3.3 Các vòng lặp và rẽ nhánh

Khi lập trình tính toán trên dữ liệu kiểu ma trận, vector các vòng lặp điều khiển rất hữu ích và được sử dụng rộng rãi. Matlab đưa ra các dạng vòng lặp và rẽ nhánh: vòng lặp for, vòng lặp while, cấu trúc if-else-end và cấu trúc switch-case.

Các vòng lặp và rẽ nhánh	
for var_=dieukien, các lệnh end	Lệnh for
while dieukien, các lệnh end	Lệnh while
if dieukien ... <elseif> <else..> end	Lệnh if (nếu .. thì ..)
switch .. case .. <otherwise ..> end	Lệnh switch (chuyển)
break	Lệnh ngắt vòng lặp trong for và while

Vòng lặp FOR

Vòng lặp for cho phép thực hiện một nhóm lệnh lặp lại với số lần xác định với cú pháp như sau

```

for index = start: increment : finish
    statement_1,
    ...,
    statement_n
end

```

Ví dụ cần xây dựng một hàm sử dụng vòng lặp for để tính tổng các phần tử của một vectơ **v**. Trên thanh công cụ, ta lựa chọn New → m-file để mở một cửa sổ soạn thảo, trong cửa sổ này ta soạn nội dung sau

```

function tong = tongvector(v)
% ham tinh tong cac phan tu cua mot vector
n=length(v);           % tra lai so phan tu, chieu dai cua vector
s = 0;                  % khoi gan cho tong
for i = 1:n              % hoac for i = 1 : 1 : n
    s = s + v(i);
end;
tong = s;
% save with file name tongvector.m

```

Với hàm định nghĩa như vậy, ta có thể sử dụng ngay trên dấu nhắc trong cửa sổ lệnh, ví dụ

```

>> x=[1:2:50];
>> s=tongvector(x)
s =    625

```

Để kiểm tra với lệnh sẵn có của Matlab ta gọi lệnh sum

```

>> sum(x)
ans =    625

```

Một số ví dụ khác về sử dụng lệnh for:

<pre> n=36; for i = 1:n a(i) = sin(i*pi/n); end </pre>	<pre> n=5; for i = 1:n for j = 1:n a(i,j) = 1/(i+j-1); end end </pre>
--	---

Vòng lặp WHILE

While là lệnh lặp với số lần lặp không xác định. Cú pháp

```

WHILE expression
    statements
END

```

Ví dụ cần tính tổng sau

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{100} = \sum_{i=1}^{100} \frac{1}{i}$$

% Tính tổng S = 1+1/2+1/3+ ... +1/100	
<pre>% su dung vong lap while function tongS n=100; S=0; i=1; while i<=n S = S+1/i; i = i+1; end tong=S; disp('Voi n = 100, tong la :') disp(tong)</pre>	<pre>% su dung vong lap for function tong_Sn n=100; S=0; for i=1:n S=S+1/i; end tong=S; disp('Voi n = 100, tong la :') disp(tong)</pre>
<pre>% Ket qua la >> tong_S Voi n = 100, tong la : 5.1874</pre>	<pre>% Ket qua la >> tong_Sn Voi n = 100, tong la : 5.1874</pre>

Để viết thành một hàm có giao diện với người sử dụng ta viết một hàm như sau

```
function S = tong_Sn(n)
% su dung vong lap for
T=1;
while T
    disp('Vao so duong n: ')
    n = input('n = ')
    if n>1 T = 0; end
end
tong=0;
for i=1:n
    tong=tong+1/i;
end
S=tong;
disp(['Voi n = ', sprintf('%4d',n), ' Tong S(n) la'])
disp(S)

% save with file name tong_Sn.m
```

Dưới đây là một số ví dụ về việc sử dụng lệnh while:

```
while 1
    disp('Vao so n. Neu n <= 0, thi thoat chuong trinh.')
    n = input(n = ')
    if n <= 0, break, end
    r = rank(magic(n))
end
disp('Nhu vay do !.')

E = 0*A; F = E + eye(size(E)); N = 1;
while norm(E+F-E,1) > 0,
    E = E + F; F = A*F/N; N = N + 1;
end
```

Ví dụ tìm số n lớn nhất sao cho tổng sau nhỏ hơn 7

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} = \sum_{i=1}^n \frac{1}{i} < 7$$

```
function n = timn(S)
% tim so n lon nhat thoa man
% tongS(n) = 1+1/2+1/3+...+1/n < S
% gia tri S > 1 vao tu ban phim
T=1;
while T
    disp('Vao so duong S')
    S = input('S = ')
    if (S>1 & S<10) T = 0; end
end
format long
tong=0; i=0;
while(tong<S)
    i = i+1;
    tong = tong+1/i;
end
n=i-1;
tong=tong-1/i;
disp('So n tim duoc la: ')
disp(n)
disp('Gia tri cua tong S(n) la: ')
disp(tong)
```

Lệnh if, cấu trúc if - else - end

Lệnh điều kiện nếu - thì. Cú pháp

```
IF expression
    statements
ELSEIF expression
    statements
ELSE
    statements
END
```

Ví dụ

```
if I == J
    A(I,J) = 2;
elseif abs(I-J) == 1
    A(I,J) = -1;
else
    A(I,J) = 0;
end
```

<pre>k=5; for i=1:k for j=1:k if i==j A(i,j)=4; elseif(abs(j-i)==2) A(i,j)=1; else A(i,j)=0; end end end end</pre>	<pre>% ket qua chuong trinh >> A A = 4 0 1 0 0 0 4 0 1 0 1 0 4 0 1 0 1 0 4 0 0 0 1 0 4</pre>
--	--

Ma trận **A** trên có thể nhận được bằng các lệnh sau

```
>> k=5;
>> d1=4*ones(k,1);      d2=ones(k-2,1);
>> a=diag(d1)+diag(d2,-2) +diag(d2, 2)
```

Cấu trúc switch-case

Lệnh chuyển trong nhiều trường hợp dựa trên biểu thức và có dạng tổng quát như sau

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

Ví dụ

```
method = 'Bilinear';

switch lower(method)
    case {'linear','bilinear'}
        disp('Method is linear')
    case 'cubic'
        disp('Method is cubic')
    case 'nearest'
        disp('Method is nearest')
    otherwise
        disp('Unknown method.')
end
% ket qua la
Method is linear
```

Ví dụ

```
diem = 1;

switch diem
    case 1
        disp('Diem gioi')
    case 2
        disp('Diem tot')
    case 3
        disp('Diem kha')
    case 4
        disp('Diem trung binh')
    otherwise
        disp('Diem kem.')
end
```

Để thoát ra khỏi vòng lặp for hoặc while, ta sử dụng lệnh break. Khi chương trình chạy đến dòng lệnh này thì nó tự động nhảy ra khỏi vòng lặp có chứa break. Ví dụ

```
for p = 7:8
    for q = 3:5
        for r = 1:2
            fprintf('\n %3.0f, %3.0f, %3.0fn' , p, q, r)
        end
        if q == 4, break; end
    end
end
fprintf('\n Xuong dong \n')
% Ket qua la
    7,      3,      1n
    7,      3,      2n
    7,      4,      1n
    7,      4,      2n
    8,      3,      1n
    8,      3,      2n
    8,      4,      1n
    8,      4,      2n
Xuong dong
```

Ví dụ sau đây chỉ ra sự tương tác giữa các hàm khi lập trình. Xét phương trình vi phân

$$\dot{y} = f(t, y) \text{ với điều kiện đầu } y(t_0) = y_0.$$

Ta cần tìm nghiệm $y(t)$ trong khoảng thời gian $t = [t_0, t_f]$.

Để giải bài toán bằng phương pháp Euler, ta sẽ chia khoảng thời gian trên thành N đoạn, với độ dài hay còn gọi là bước thời gian là $h = (t_f - t_0) / N$. Gọi $z_i = z(t_i)$ là trị gần đúng của hàm $y(t)$ tại thời điểm $t_i = t_0 + ih$, ($i = 1, 2, \dots, N$), theo phương pháp Euler dãy các giá trị z_i được tính theo công thức sau

$$z_0 = y(t_0), \quad z_i = z_{i-1} + h \cdot f(t_{i-1}, z_{i-1})$$

Khi tiến hành thực hiện trong Matlab ta sẽ soạn ba m-file: một mô tả phương trình vi phân, một mô tả phương pháp gần đúng Euler, và một chương trình chính.

Cụ thể để giải phương trình vi phân

$$\dot{y} = \frac{1}{m}(F_0 - by), y(t_0) = 0, \quad t_0 = 0, \quad t_f = 10.$$

Ta biết nghiệm chính xác của phương trình này là

$$y(t) = \frac{F_0}{b} [1 - e^{-(b/m)t}]$$

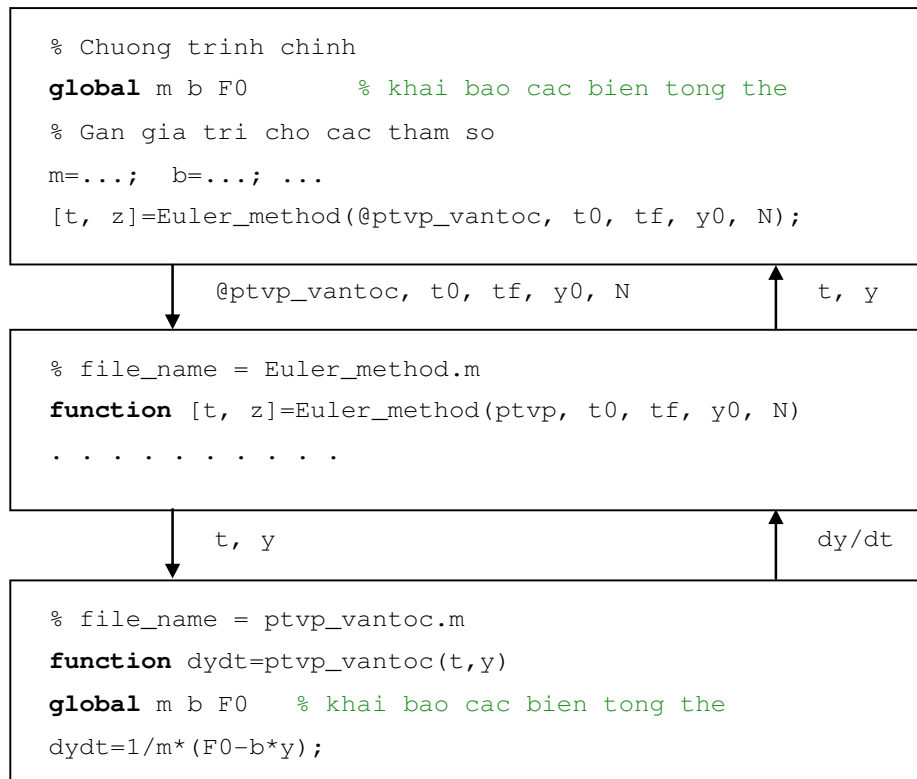
Dưới đây là ba m-file được soạn.

```
% file_name = ptvp_vantoc.m
function dydt=ptvp_vantoc(t,y)
global m b F0 % khai bao cac bien tong the
dydt=1/m*(F0-b*y);
```

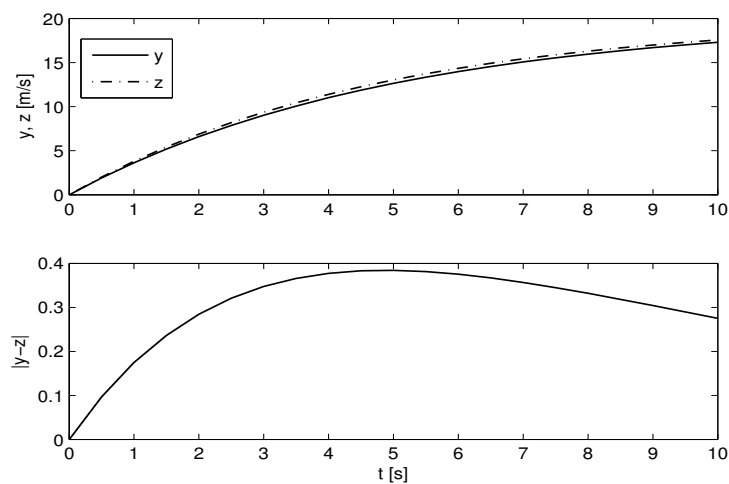
```
= = = = =
% file_name = Euler_method.m
function [t, z]=Euler_method(ptvp, t0, tf, y0, N)
% Cac bien:
% ptvp - ten file chua ve phai cua phuong trinh vi phan can
giai
% t0 - thoi diem dau
% tf - thoi diem cuoi (t0 < tf)
% y0 - gia tri ham tai thoi diem dau
% N - so doan ma khoang thoi gian (tf-t0) duoc chia ra
global m b F0 % khai bao cac bien tong
h = (tf - t0)/N; % buoc thoi gia
t = t0+[0:N]*h; % vector chua cac diem c
z(1,:) = y0;
for k = 1:N
    z(k + 1,:) = z(k,:)+h*ptvp(t(k),z(k,:));
end
```

```
= = = = =
% file_name = tuy_chon__.m
% chuong trinh chinh
global m b F0 % khai bao cac bien tong the
m=1; b=0.2; F0=4; % Gan gia tri cho cac tham so
t0=0; tf=10; N=20; y0=0;
% nghiem gan dung z
[t, z]=Euler_method(@ptvp_vantoc, t0, tf, y0, N);
%nghiem chinh xac
y=F0/b*(1-exp(-b/m.*t));
figure(1)
subplot(211)
plot(t,y,'k',t,z,'-k','Linewidth',1);
% so sanh ket qua tinh voi nghiem chinh xac
legend('y','z')
ylabel('y, z [m/s]');
subplot(212)
plot(t,abs(z-y),'k','Linewidth',1);
% sai so cua phuong phap
xlabel('t [s]'); ylabel('|y-z|');
```

Khi chạy chương trình chính các chương trình sẽ trao đổi dữ liệu theo sơ đồ sau:



Khi chạy chương trình chính cho ta các kết quả như trên đồ thị dưới đây.



Hình 1. Đồ thị các hàm $y(t)$, $z(t)$ và hiệu $y(t)-z(t)$

3.4 Các Mat-File

Mat-file là tệp có phần mở rộng là mat và do đó được gọi là tệp chấm mat (filename.mat). Mat-file là những tệp nhị phân được nén để lưu trữ kết quả số. Các file này được sử dụng để ghi các kết quả mà đã được tạo ra bởi chuỗi các lệnh, chỉ thị của Matlab. Chẳng hạn, để ghi giá trị của hai biến x_1 và x_2 trong file có tên dulieu.mat ta đánh dòng lệnh

```
>> x1=5;  
>> x2=10;  
>> save dulieu.mat x1 x2
```

Ghi tất cả các biến hiện hành trong một file được thực hiện bằng cách gõ dòng lệnh

```
>> save filename.mat
```

Muốn tải (đọc) một mat-file vào MATLAB ta gõ dòng lệnh

```
>> load filename (or load filename.mat)
```

3.5 Bài tập thực hành

1. Viết một hàm Matlab hỏi người sử dụng bán kính và chiều cao của hình trụ, sau đó tính toán và đưa ra màn hình diện tích toàn phần của hình trụ và thể tích của hình trụ.

2. Viết một chương trình con có sử dụng lệnh while để tính tổng $S(x,n)$

$$S(x, n) = 1 + x + x^2 + \dots + x^n$$

3. Viết một chương trình con có sử dụng lệnh for để tính tổng $S(x,n)$

$$S(x, n) = 1 + 1/x + 1/x^2 + \dots + 1/x^n$$

4. Viết một chương trình con có sử dụng lệnh while để tính $\sqrt{2}$ theo phương pháp lặp Newton-Raphson, theo công thức

$$x_{n+1} = \frac{x_n^2 + 2}{2x_n}$$

Hãy lặp để đạt độ chính xác bốn số sau dấu phẩy, tức $|x_{n+1} - x_n| < 10^{-5}$.

5. Hãy sử dụng vòng lặp while để tìm thương và số dư khi chia hai số nguyên a cho b , $a > b$.

Đồ họa trong Matlab

Đồ họa là một trong những ứng dụng quan trọng nhất của một gói công cụ toán sử dụng trên máy tính, và Matlab cũng không là một ngoại lệ. Thông thường chúng ta muốn thể hiện bằng hình ảnh các hàm mà quá khó khi vẽ bằng tay hoặc để thể hiện các dữ liệu tạo ra hoặc thu được từ thực nghiệm. Việc thể hiện các kết quả dưới dạng đồ thị 2D và 3D có ý nghĩa quan trọng để làm rõ các quá trình khảo sát. Matlab cung cấp rất nhiều các khả năng để thực hiện việc này. Trong chương này, các câu lệnh và các kỹ thuật sử dụng trong Matlab thực hiện nhiệm vụ này sẽ được trình bày.

4.1 Đồ họa 2D

Trước hết ta bắt đầu với việc vẽ đồ thị hàm một biến $y = f(x)$. Công việc này trong Matlab bao gồm ba bước:

- Định nghĩa hàm cần vẽ $f(x)$,
- Xác định miền giá trị của biến $x = [a, b]$,
- Gọi hàm plot(x,y) của Matlab.

Khi xác định miền giá trị của đối số, ta phải báo cho Matlab biết bước của biến mà chúng ta cần xác định giá trị của hàm. Một bước chia phù hợp là rất cần thiết để có thể nhận được đường cong trơn của hàm số. Nếu bước chia lớn đồ thị sẽ bị gãy khúc, còn nếu quá nhỏ sẽ gây lãng phí bộ nhớ và thời gian tính toán. Để minh họa điều này, ta xem các ví dụ sau đây.

Vẽ đồ thị hàm $y = \sin x$ trong miền $0 \leq x \leq 10$. Để bắt đầu ta định nghĩa miền cần vẽ (x_{start}, x_{end}) cùng với bước h của nó theo cú pháp:

$$x = [x_{start} : h : x_{end}]$$

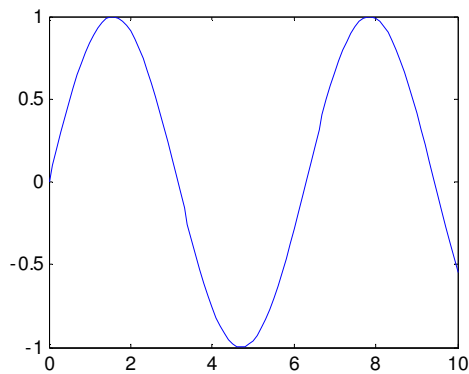
với dòng lệnh này, miền vẽ sẽ được chia thành $n = (x_{end} - x_{start}) / h + 1$ đoạn bằng nhau nhờ các điểm chia x_k , $k = 1, 2, \dots, n$; Giá trị của các điểm chia này là

$$x_k = x_{start} + (k-1) \cdot h, \quad k = 1, 2, \dots, n$$

Kết quả của việc chia miền là một véc tơ gồm n phần tử được gán cho biến x . Sau đó ta tính giá trị của hàm số tại các điểm vừa chia, và cũng nhận được một véc tơ cùng cỡ với biến x . Với hàm đơn giản như $y = \sin x$ thì ta có thể tính véc tơ y theo cách viết thông thường. Khi đã có hai véc tơ x và y cùng cỡ, ta chuyển sang bước ba đó là sử dụng lệnh `plot(x,y)` để vẽ đồ thị. Thực hiện ba dòng lệnh sau

```
>> x=[0:0.1:10];
>> y=sin(x);
>> plot(x,y)
```

Sau khi gõ ENTER giây lát, Matlab mở ra một cửa sổ mới với tên Figure 1, trong đó có hiện lên đồ thị hàm số $y = \sin x$. Kết quả của ba dòng lệnh trên được thể hiện như trên hình 4-1.



Hình 4-1. Đồ thị hàm $y = \sin x$ với $h = 0.1$

Bây giờ ta sẽ vẽ lại hàm trên với bước chia là 1, tăng 10 lần so với trường hợp trên.

Thực hiện lệnh

```
>> x=[0:0:10];
```

Và thử vẽ lại bằng lệnh, nhưng Matlab đã báo lỗi

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

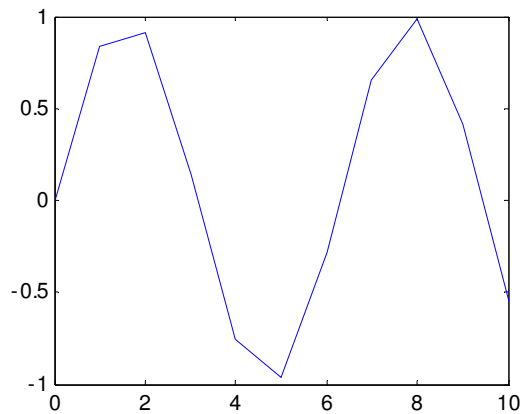
```
??? Error using ==> plot
```

```
Vectors must be the same lengths.
```

Chuyện gì đã xảy ra, chúng ta đã định nghĩa hàm $y = \sin x$ tại sao Matlab không vẽ mà lại báo lỗi. Ở đây chúng ta đã chia lại miền vẽ, số phần tử của véc tơ x đã thay đổi, nhưng ta chưa tính lại các giá trị của y , véc tơ này vẫn giữ lại các giá trị của lần tính trước và như vậy hai véc tơ x và y có số phần tử khác nhau. Đó là lý do. Để vẽ được ta cần tính lại với véc tơ y , hãy sử dụng phím lên xuống (\uparrow , \downarrow) để chọn lại dòng lệnh hoặc viết lại.

```
>> y = sin(x);
>> plot(x,y)
```

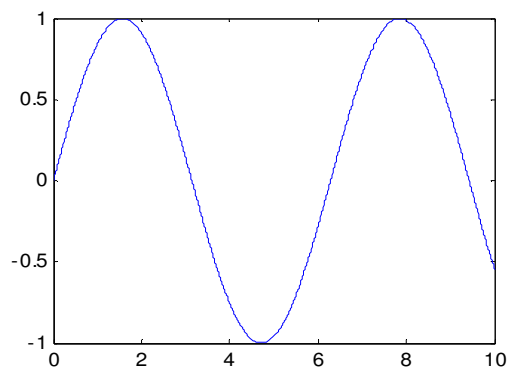
Và nhận được đồ thị trong cửa sổ Figure 1 như trên hình 4-2. Ta thấy đồ thị này bị gãy khúc, nhìn không giống với đồ thị hình sin quen thuộc.



Hình 4-2. Đồ thị hàm $y = \sin x$ với $h = 1.0$

Bây giờ ta thực hiện lại ba dòng lệnh trên với bước $h = 0.01$, tức là nhỏ hơn 10 lần so với lần vẽ đầu tiên. Kết quả là được đồ thị như trên hình 4-3, đường cong trên hình này rất trơn.

```
>> x = [0:0.01:10], y=sin(x);
>> plot(x,y)
```

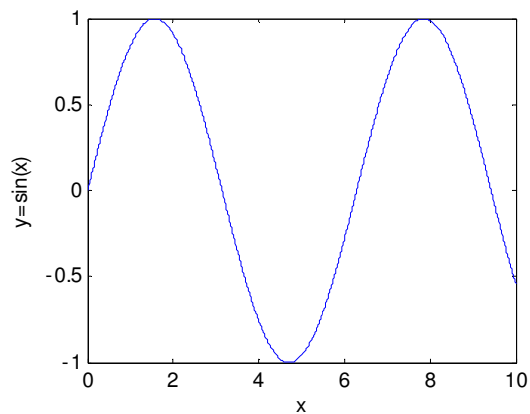


Hình 4-3. Đồ thị hàm $y = \sin x$ với $h = 0.01$

Để làm rõ hơn ta cần đưa thêm vào đồ thị tên của các trục tọa độ và biểu thức của hàm được vẽ. Việc này sẽ được thực hiện bằng các lệnh xlabel và ylabel. Các lệnh này có thể được sử dụng với một đối số, là tên mà bạn muốn đặt cho từng trục. Tên

của trục cần phải đặt trong hai dấu nháy đơn ‘ten trục’. Các lệnh đặt tên này có thể được viết cùng dòng với lệnh plot, nhưng nhớ phải có dấu phẩy giữa các lệnh.

```
>> x=[0:0.01:10];
>> y=sin(x);
>> plot(x,y), xlabel('x'), ylabel('y=sin(x)');
```



Hình 4-4a. Đồ thị hàm $y = \sin x$ với $h = 0.01$ cùng với nhãn của trục tọa độ

Đặt màu và kiểu đường cho đồ thị

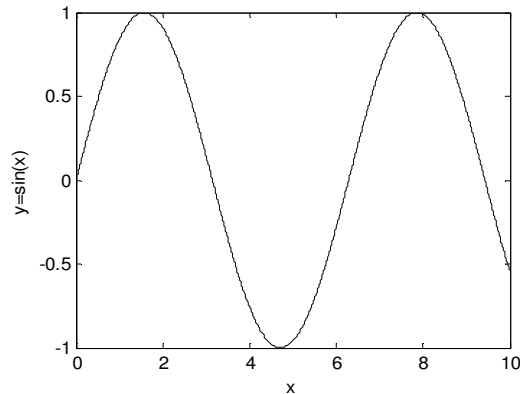
Khi bạn vẽ đồ thị một hàm một biến, đồ thị hiện lên với màu đã mặc định là xanh dương (blue). Hoặc khi vẽ nhiều đồ thị trên cùng một hệ trục tọa độ Matlab sẽ chọn các màu khác nhau, mỗi màu cho một đường. Nếu in màu thì đồ thị rất đẹp và rõ, tuy nhiên nếu không có màu các đồ thị này rất mờ. Để đổi màu cho đồ thị ta cần bổ sung thêm tham số cho lệnh plot. Các màu trong Matlab được ký hiệu như trong bảng 3-1 và đưa vào lệnh plot trong dấu nháy đơn, như một ký tự.

Bảng 3-1. Ký hiệu màu sắc

Màu	Ký hiệu
black – đen	k
blue – xanh da trời	b
cyan – lục lam	c
green – xanh lá cây	g
red – đỏ	r
magenta – đỏ tươi	m
yellow – vàng	y
white – trắng	w

Ví dụ ta muốn đồ thị hình sin trên có màu đen:

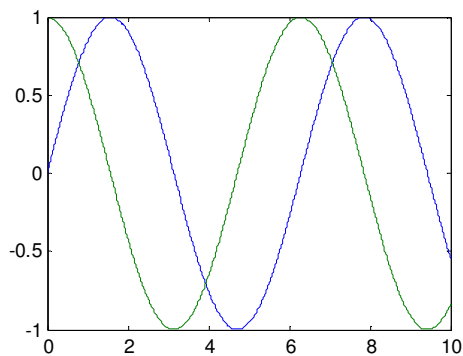
```
>> plot(x,y,'k'), xlabel('x'), ylabel('y=sin(x)');
```



Hình 4-4b. Đồ thị hàm $y = \sin x$ màu đen

Nếu bạn không khai báo màu trong plot, Matlab tự động chọn màu cho các đường. Xem ví dụ sau:

```
>> x=[0:0.01:10];  
>> y=sin(x);  
>> z=cos(x);  
>> plot(x,y, x,z)
```



Hình 4-4c. Đồ thị hai hàm $y = \sin x$ và $z = \cos x$

Nếu đồ thị của bạn có nét quá mảnh, khi in có thể mờ, bạn có thể bổ sung thêm lựa chọn Linewidth (độ mảnh của đường). Matlab mặc định độ mảnh của đường là 1, nếu muốn làm nổi bật bạn có thể chọn bằng 2. Cách bổ sung lựa chọn này như sau

```
>> plot(x,y, 'k-', 'Linewidth', 2)
```

Một số tùy chọn khi vẽ đồ thị 2D

Như trên đã trình bày một số thủ tục cơ bản nhất để Matlab đưa ra đồ thị của hàm một biến. Bây giờ chúng ta sẽ xem xét kỹ hơn một số tùy chọn (options) bổ sung để có nhiều thông tin được hiển thị trên đồ thị. Bạn có thể đưa tên của đồ thị bằng

lệnh title. Lệnh title với phần tên đồ thị để trong cặp dấu nháy đơn, ' ', tức là ở dạng sâu ký tự, Matlab sẽ hiện thị tên đó lên phía trên đồ thị. Chẳng hạn muốn vẽ đồ thị $x = e^{-2t} \sin t$, trong khoảng $0 \leq t \leq 4$, với t là thời gian tính bằng giây (s), bước thời gian là $h = 0.01$ s. Ta muốn đặt tên cho đồ thị này là 'Dao động có cản' (Dao động có cản), thì sẽ làm thế nào?.

Trước hết chia khoảng thời gian cần thể hiện

```
>> t = [0:0.01:4];
```

Sau đó xác định giá trị của hàm tại các điểm chia

```
>> x = exp(-2*t)*sin(t);
```

Tuy nhiên, với dòng lệnh trên Matlab sẽ báo lỗi!

```
??? Error using ==> mtimes
```

```
Inner matrix dimensions must agree.
```

Lý do của lỗi trên là, với t là một vectơ hàng, Matlab sẽ tính các đại lượng $\exp(-2*t)$ và $\sin(t)$ cũng là các vectơ hàng, và với phép nhân thông thường Matlab không thực hiện được, vì chưa được định nghĩa trong nó.

Để vẽ được hàm trên, cách thứ nhất là sử dụng hàm *fplot*. Hàm *fplot* này tự động chọn bước chia, và hàm cần vẽ được nhập vào lệnh như là một chuỗi ký tự, trong cặp dấu nháy đơn. Với cú pháp

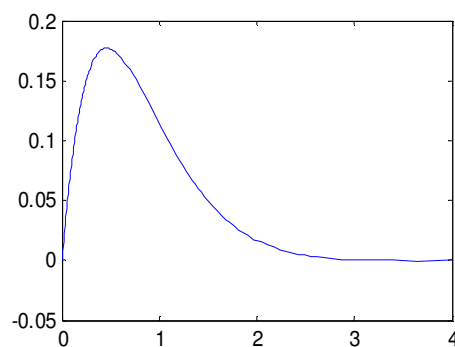
```
fplot('biểu thức hàm', [t_start, t_end])
```

Với hàm dao động có cản trên, ta viết

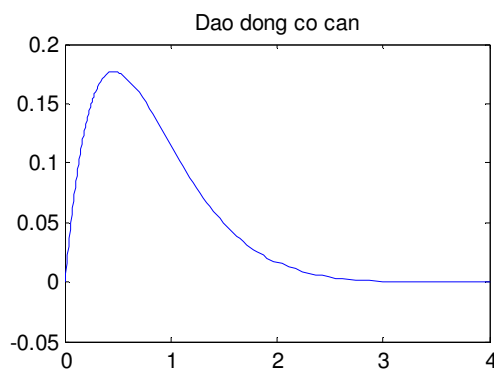
```
>> fplot('exp(-2*t)*sin(t)', [0, 4])
```

Matlab nhanh chóng đưa ra đồ thị như trên hình 3-5. Nếu muốn thêm tên của đồ thị vào, ta sử dụng lệnh như dòng dưới đây và nhận được đồ thị như trên hình 3-6.

```
>> title('Dao động có cản')
```



Hình 3-5.



Hình 3-6.

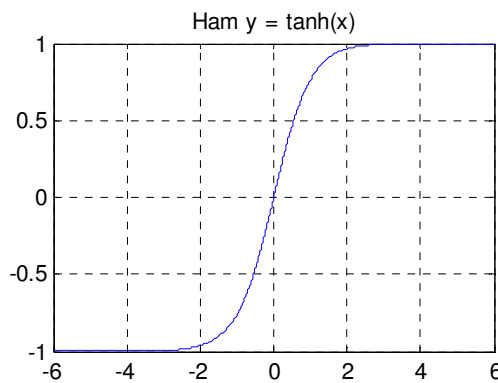
Một phương án khác là ta sử dụng các phép tính nhân, mũ, chia phần tử (`.*`, `.^`, `./`) để viết biểu thức của hàm số vào dòng lệnh của Matlab. Với hàm số dao động có cản, ta viết như sau:

```
>> t = [0:0.01:4];
>> x=exp(-2*t).*sin(t);
>> plot(t,x), title('Dao dong co can')
```

Kết quả nhận được như trên hình 3-6.

Một ví dụ khác là vẽ đồ thị hàm $y = \tanh x$, trong miền $-6 \leq x \leq 6$, với các đường dóng xuất hiện trên đồ thị. Lệnh `grid on` sau lệnh `plot` sẽ hiển thị các đường dóng.

```
>> x = [-6:0.01:6];
>> y = tanh(x);
>> plot(x,y), title('Ham y = tanh(x)'), grid on
```



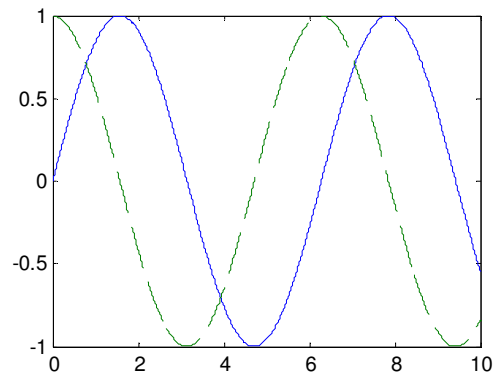
Hình 3-7. Đồ thị được vẽ cùng với lệnh `grid on`

Vẽ nhiều đồ thị trên cùng một hệ trục

Trong nhiều trường hợp, ta cần thể hiện nhiều đường trong cùng một hệ trục tọa độ. Chẳng hạn cần vẽ đồ thị các hàm $y_1 = \sin x$, và $y_2 = \cos x$, $0 \leq x \leq 10$, trong cùng một hệ trục. Chúng ta sẽ vẽ hàm $y_1(x)$ bằng nét liền và $y_2(x)$ bằng nét đứt. Thực hiện các dòng lệnh sau và nhận được kết quả như trên hình 3-8.

Thông số thể hiện trong dấu nháy ở dòng lệnh cuối của các dòng lệnh (`'--'`) cho Matlab biết rằng hàm y_2 sẽ được vẽ bằng đường nét đứt. Quan sát trên đồ thị ta thấy, Matlab đã tự động chọn hai màu khác nhau cho hai đường. Trong Matlab có bốn kiểu đường đã được định nghĩa (nét liền, nét đứt, nét chấm gạch, và nét chấm), ngoài ra có rất nhiều kiểu để đánh dấu các điểm trên đồ thị như liệt kê trong bảng

```
>> x=[0:0.01:10];
>> y2=cos(x);
>> y1=sin(x);
>> plot(x,y1, x,y2,'--')
```



Hình 3-8. Đồ thị y1 (nét liền) và y2 nét đứt

Bảng 3-2. Ký hiệu kiểu đường

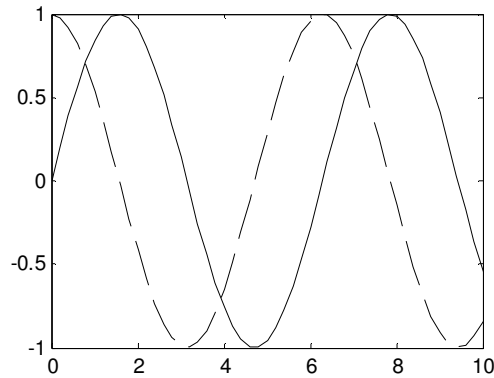
Kiểu đường (line style)		
-	solid line	đường nét liền
--	dashed line	đường nét đứt
-.	dashdot line	đường nét chấm gạch
:	dotted line	đường nét chấm
(none)	no line	(không hiện đường lên)

Bảng 3-3. Ký hiệu đánh dấu

Kiểu đánh dấu (marker)					
.	point	điểm	v	triangle (down)	tam giác
o	circle	vòng tròn	^	triangle (up)	
x	x-mark	chữ x	<	triangle (left)	
+	plus	dấu cộng	>	triangle (right)	
*	star	dấu sao	p	pentagram	
s	square	dấu vuông	h	hexagram	
d	diamond	kim cương			

Để các đồ thị cùng màu đen (k) và có kiểu đường khác nhau một nét liền (-) và một nét đứt (--), ta viết dòng lệnh (hình 3-9)

```
>> x=[0:0.01:10];
>> y2=cos(x);
>> y1=sin(x);
>> plot(x,y1,'k-', x,y2,'k--')
```



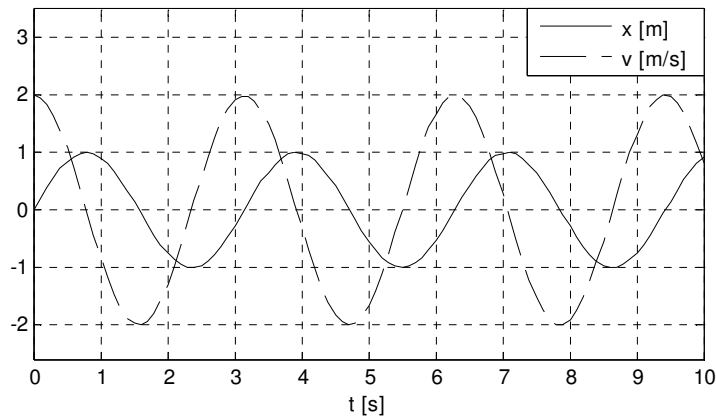
Hình 3-9. Đồ thị y_1 (nét liền) và y_2 nét đứt

Nếu không khai báo kiểu đường Matlab tự động chọn nét liền, đó là kiểu đường mặc định.

Thêm các chú thích

Một đồ thị có ý nghĩa vật lý, nếu trên đó thể hiện được các đại lượng với thứ nguyên hay đơn vị đo rõ ràng. Và trong trường hợp có nhiều đồ thị trên cùng một hình, thì cần phải chỉ rõ từng đường. Với lệnh *legend* Matlab cho phép ta điền các chú giải lên hình vẽ. Ví dụ ta cần vẽ đồ thị hai hàm số theo thời gian tính bằng giây, một là dịch chuyển với đơn vị đo là mét, và một là vận tốc với đơn vị đo là mét/giây:

$$x = \sin(2t) \text{ m}, \quad v = 2 \cos(2t) \text{ m/s}.$$



Hình 3-10. Đồ thị các hàm với legend

Thực hiện các dòng lệnh sau

```
>> t = [0:0.1:10];
```

```
>> x = sin(2*t);    v = 2*cos(2*t);
>> plot(t,x,'k-', t,v,'k--'), grid on, xlabel('t [s]')
>> legend('x [m]', 'v [m/s]'), axis([0 10 -2.6 3.5])
```

Cho ta kết quả như trên hình 4-10 với các chú thích về kiểu đường và thứ nguyên của đại lượng cần hiển thị.

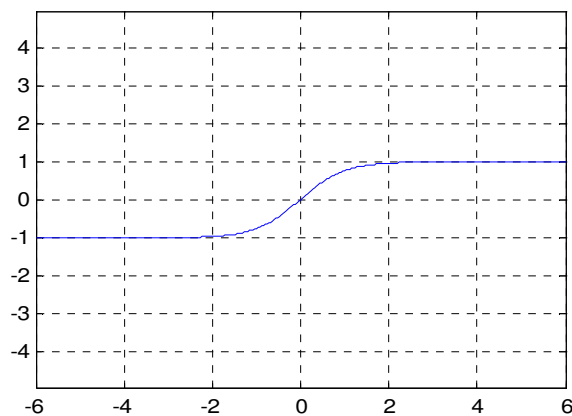
Các lệnh axis

Matlab cho phép bạn điều chỉnh các trục tọa độ sử dụng trong đồ thị hai chiều theo cách sau đây. Nếu thêm vào lệnh *axis square* vào cùng dòng với lệnh *plot*, thì đồ thị sẽ hiện ra trong một hình vuông, Matlab đã kéo các trục để phần vẽ trở nên vuông. Hãy thực hiện dòng lệnh và quan sát

```
>> plot(x,y), title('Ham y = tanh(x)'), axis square
```

Nếu bạn thêm lệnh *axis equal* vào cùng dòng với lệnh *plot*, thì các trục sẽ được điều chỉnh với cùng một tỷ lệ xích (hình 3-11).

```
>> plot(x,y), grid on, axis equal
```



Hình 3-11. Đồ thị được vẽ cùng với lệnh grid on và axis equal

Như đã thấy, với các lựa chọn khác nhau cho ta các đồ thị tương ứng, sử dụng lựa chọn nào là do bạn quyết định để nhận được đồ thị trong miền vẽ là hợp lý, chẳng hạn ta không nên để đồ thị hiện ra trong một diện tích vẽ quá lớn như trên hình 3-11. Giới hạn của trục tung từ -5 đến +5 là không cần thiết. Lãng phí diện tích!. Để Matlab tự động chọn giới hạn cho các trục ta sử dụng lệnh *axis auto*. Với

```
>> plot(x,y), title('Ham y = tanh(x)'), grid on, axis auto
```

Kết quả nhận được một đồ thị như trên hình 3-11.

Đặt giới hạn miền vẽ với lệnh axis

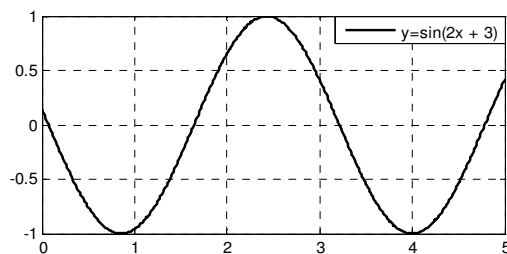
Để giới hạn miền vẽ đồ thị, ta sử dụng lệnh axis với cách viết

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

Chẳng hạn muốn vẽ đồ thị hàm $y = \sin(2x + 3)$, $0 \leq x \leq 5$, có thể chúng ta đã biết rằng giá trị của hàm này chỉ nằm trong giới hạn $-1 \leq y \leq 1$. Để giá trị giới hạn của trục tung trong khoảng đó, ta sử dụng chuỗi các dòng lệnh sau, trong đó lệnh axis có tác dụng giới hạn miền vẽ của các trục tọa độ.

```
>> x = [0:0.01:5];  
>> y = sin(2*x + 3);  
>> plot(x,y,'k','Linewidth',1.5), axis([0 5 -1 1]), grid on,  
>> legend('y=sin(2x + 3)')
```

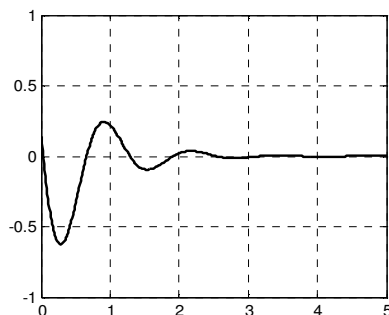
Các dòng này cho kết quả như trên hình 3-12.



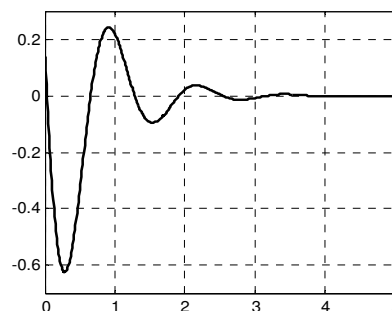
Hình 3-12.

Bây giờ ta vẽ đồ thị hàm $y = e^{-3x/2} \sin(5x + 3)$, với $0 \leq x \leq 5$. Trước hết thử với $-1 \leq y \leq 1$:

```
>> x = [0:0.01:5];  
>> y = exp(-1.5*x).*sin(5*x+3);  
>> plot(x,y, 'k', 'Linewidth',2), axis([0 5 -1 1]), grid on
```



Hình 3-13a.



Hình 3-13b.

Như thấy trên hình 3-13a, giới hạn của trục y nên được hiệu chỉnh, trong khoảng từ $-0.7 \leq y \leq 0.3$ (hình 3-13). Với hình 3-13b ta có thể quan sát đồ thị hàm số chi tiết hơn.

```
>> plot(x,y, 'k', 'Linewidth',2), axis([0 5 -0.7 0.3]), grid on
```

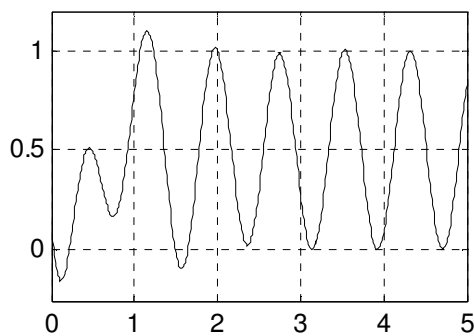
Một thủ thuật để giới hạn trục y một cách tự động, ta có thể sử dụng hàm **min** và **max** để tìm các giá trị nhỏ nhất và lớn nhất của véc-tơ y. Từ đó đưa ra các giới hạn hợp lý. Ví dụ đối với hàm số

$$y = e^{-3x/2} \sin(5x + 3) + \sin^2(2x), \text{ với } 0 \leq x \leq 5.$$

Thực hiện

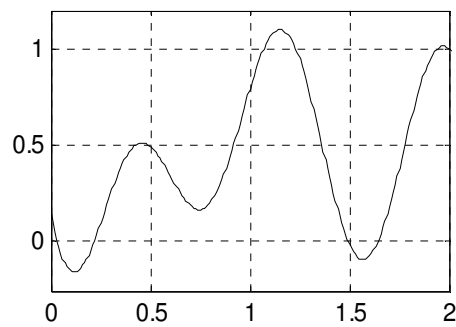
```
>> x = [0:0.01:5];
>> y = exp(-1.5*x).*sin(5*x+3)+sin(4*x).^2;
>> plot(x,y, 'k', 'Linewidth',2), grid on
>> axis([min(x) max(x) min(y) max(y)])
>> axis([min(x) max(x) min(y)-0.1 max(y)+0.1])
```

Kết quả nhận được là hình 3-14a.



a) với

```
axis([min(x) max(x) min(y)-0.1 max(y)+0.1])
```



b) với

```
axis([0 2 min(y)-0.1 max(y)+0.1])
```

Hình 3-14

Hình 3-14b là kết quả của các dòng lệnh dưới đây.

```
>> plot(x,y, 'k', 'Linewidth',2), grid on
>> axis([0 2 min(y)-0.1 max(y)+0.1])
```

Lệnh Subplots

Lệnh subplot cho phép ta vẽ nhiều đồ thị trong nhiều hệ trục tọa độ vào cùng một hình vẽ. Lệnh này được gọi với cú pháp *subplot(m, n, p)*, trong đó *m* và *n* cho biết số hàng và số cột trong hình vẽ như là một ma trận hay mảng. Số *p* cho biết đồ thị sẽ được vẽ vào ô thứ mấy trong mảng. Để làm sáng tỏ, ta thực hiện các ví dụ

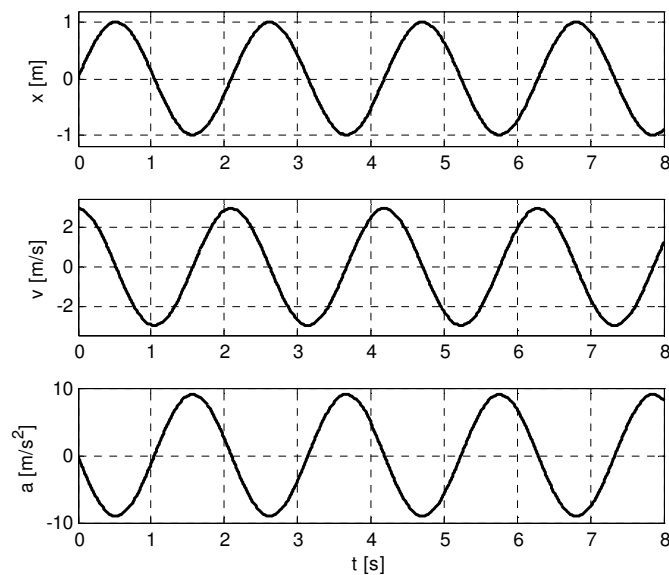
sau đây. Hãy vẽ đồ thị của ba hàm sau đây trên ba hệ trục vào cùng một hình vẽ, trong đó biến thời gian t tính bằng giây, với $0 \leq t \leq 8$:

$$x = \sin 3t \text{ m}, \quad v = 3 \cos 3t \text{ m/s}, \text{ và } a = -9 \sin 3t \text{ m/s}^2.$$

Trong trường hợp này hình vẽ sẽ có ba hàng và một cột, ($m = 3, n = 1$), số p sẽ chạy từ 1 đến 3. Thực hiện các dòng lệnh

```
>> t = [0:0.01:8];
>> x = sin(3*t); v=3*cos(3*t); a=-9*sin(3*t);
>> subplot(3,1,1)
>> plot(t,x,'k-','Linewidth',1.5), grid on, ylabel('x [m]'),
>> axis([0 8 -1.2 1.2])
>> subplot(3,1,2)
>> plot(t,v,'k-','Linewidth',1.5), grid on, ylabel('v [m/s]'),
>> axis([0 8 -3.5 3.5])
>> subplot(3,1,3)
>> plot(t,a,'k-','Linewidth',1.5), grid on, ylabel('a [m/s^2]'),
>> axis([0 8 -10 10])
>> xlabel('t [s]')
```

Kết quả nhận được như trên hình 3-15.



Hình 3-15. Sử dụng subplot (3,1, ...)

Tiếp theo là ví dụ về việc vẽ bốn đồ thị vào một hình bốn ô, hai hàng và hai cột, ($m = 2, n = 2$), số p sẽ chạy từ 1 đến 4.

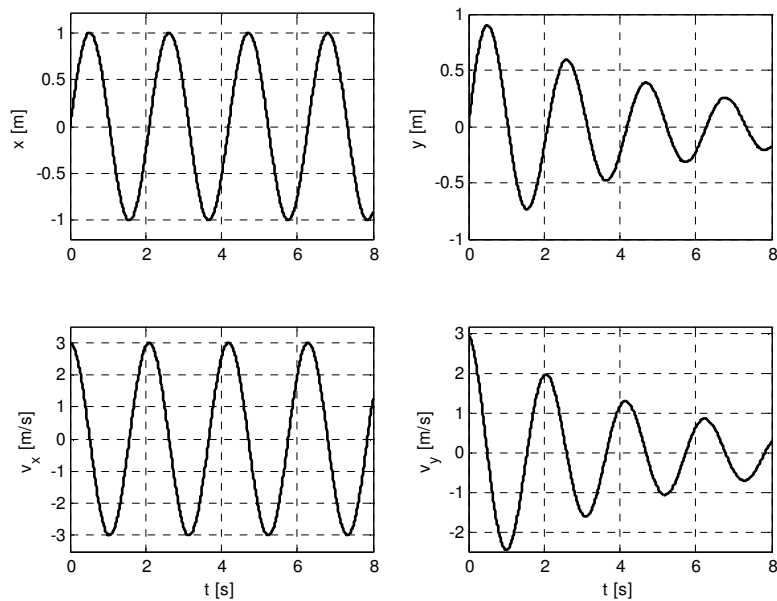
$$x = \sin 3t \text{ m}, \quad y = e^{-0.2t} \sin 3t \text{ m},$$

$$v_x = 3 \cos 3t \text{ m/s}, \quad v_y = -0.2e^{-0.2t} \sin 3t + 3e^{-0.2t} \cos 3t \text{ m/s}.$$

Thực hiện các dòng lệnh

```
>> t = [0:0.01:8];
>> x = sin(3*t);          vx=3*cos(3*t);
>> y = exp(-0.2*t).*sin(3*t);
>> vy = -0.2*exp(-0.2*t).*sin(3*t)+3*exp(-0.2*t).*cos(3*t);
>> subplot(2,2,1)
>> plot(t,x,'k-','Linewidth',2), grid on, ylabel('x [m]'),
>> axis([0 8 -1.2 1.2])
>> subplot(2,2,3)
>> plot(t,vx,'k-','Linewidth',2), grid on, ylabel('v_x [m/s]'),
>> axis([0 8 -3.5 3.5])
>> xlabel('t [s]')
>> subplot(2,2,2)
>> plot(t,y,'k-','Linewidth',2), grid on, ylabel('y [m]'),
>> axis([0 8 -1 1])
>> subplot(2,2,4)
>> plot(t,vy,'k-','Linewidth',2), grid on, ylabel('v_y [m/s]'),
>> axis([0 8 -2.5 3.2])
>> xlabel('t [s]')
```

Kết quả nhận được như trên hình 3-16.



Hình 3-16. Sử dụng subplot (2, 2, ...)

Cần lưu ý rằng thứ tự các ô (tức là số p) trong mảng $m \times n$ được đánh số hết hàng này qua hàng khác, tương tự như khi sử dụng một chỉ số để tham chiếu vào phần tử của ma trận. Như trong trường hợp trên hàng một gồm các ô 1 và 2, và hàng hai gồm các ô 3 và 4, theo thứ tự từ trái qua phải.

Vẽ các đồ thị xếp chồng và lệnh `linspace`

Giả sử chúng ta đã có đồ thị một hàm số và quyết định vẽ thêm đồ thị một hàm nữa vào cùng với đồ thị đã có. Chúng ta có thể làm việc này với hai lần gọi hàm `plot` cùng với cụm từ `hold on` để báo cho Matlab biết cần giữ lại đồ thị của hàm thứ nhất đã có.

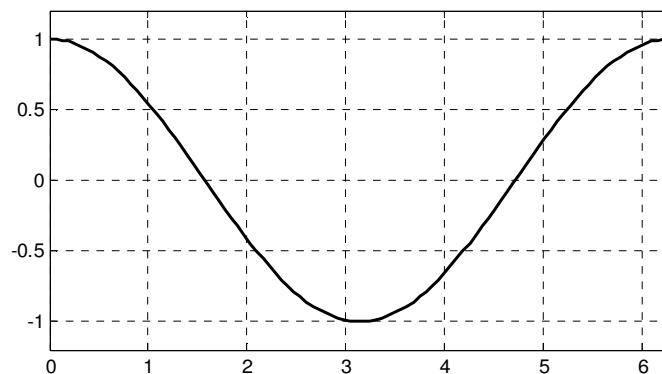
Trong ví dụ sau chúng ta sẽ vẽ đồ thị của các hàm $\cos(x)$ và $\sin(x)$ và đặt chúng vào cùng một hình vẽ. Trước hết chúng ta sẽ học thêm một lệnh mới có thể sử dụng để tạo ra dữ liệu cho trục hoành x . Đó là lệnh `linspace`, với hai cách gọi như sau:

<code>x = linspace(a,b)</code>	Matlab tự động chia khoảng (a, b) bằng 100 điểm cách đều
<code>x = linspace(a,b,n)</code>	Matlab chia khoảng (a, b) bằng n điểm cách đều

Bây giờ để vẽ các hàm sin và cos trong khoảng từ 0 đến 2π , ta thực hiện các dòng lệnh sau:

```
>> x = linspace(0,2*pi);
>> plot(x,cos(x),'k-','Linewidth',2), grid on
>> axis([0 2*pi -1.2 1.2])
```

Kết quả cho ta là đồ thị hàm $\cos(x)$ như trên hình 3-17.

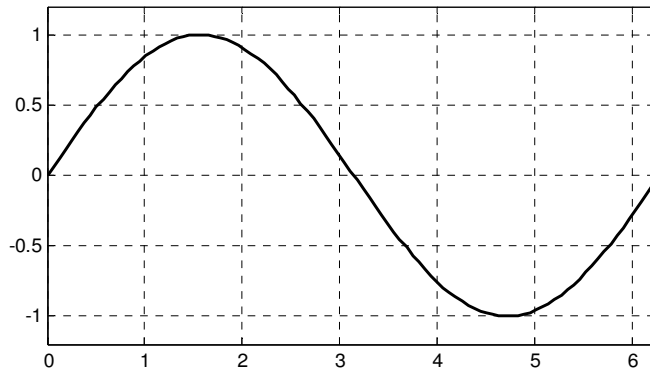


Hình 3-17. Đồ thị hàm thứ nhất

Nếu tiếp tục với dòng lệnh

```
>> plot(x,sin(x),'k-','Linewidth',2), grid on
>> axis([0 2*pi -1.2 1.2])
```

ta nhận được một hình vẽ mới đồ thị hàm $\sin(x)$, như thế hình vẽ của hàm $\cos(x)$ vừa vẽ đã bị xóa, và Matlab đã tạo ra một hình vẽ mới



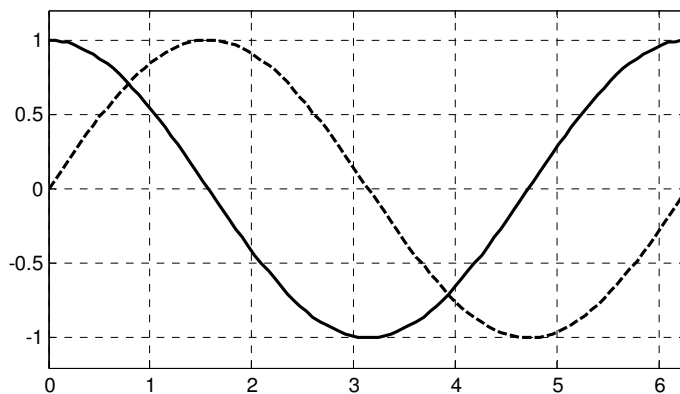
Hình 3-18. Đồ thị hàm thứ hai, không sử dụng hold on

Để cả hai đồ thị cùng tồn tại trên một hình vẽ, thì trước khi vẽ hàm thứ hai ta phải sử dụng lệnh *hold on*. Với dòng thông báo này Matlab sẽ giữ lại đồ thị đã vẽ trước đó. Trong trường hợp trên ta cần làm như sau

```
>> x = linspace(0,2*pi);
>> plot(x,cos(x),'k-', 'Linewidth',2), grid on
>> axis([0 2*pi -1.2 1.2])
```

Nếu tiếp tục với dòng lệnh

```
>> hold on
>> plot(x,sin(x),'k--', 'Linewidth',2), grid on
>> axis([0 2*pi -1.2 1.2])
>> hold off
```



Hình 3-19. Đồ thị hai hàm cùng hệ trục tọa độ, có sử dụng hold on

Và nhận được kết quả như trên hình vẽ 3-19. Để phân biệt hai đường ta đã chọn các kiểu đường nét đứt cho đồ thị thứ hai. Khi muốn kết thúc tác dụng của lệnh *hold on* ta sử dụng lệnh *hold off*.

Đồ thị theo tọa độ cực và Đồ thị với thang chia Logarith

Trong các phần trước ta đã vẽ đồ thị các hàm số trong hệ trục tọa độ đề các vuông góc, tuy nhiên trong kỹ thuật có nhiều đường cong nếu biểu diễn theo tọa độ cực sẽ đơn giản hơn. Ví dụ như khi vẽ đường cong có tên là đường xoắn ốc Acximet có phương trình trong dạng tọa độ cực là

$$r = a\varphi, \text{ với } a \text{ là hằng số và } \varphi \text{ là tọa độ góc.}$$

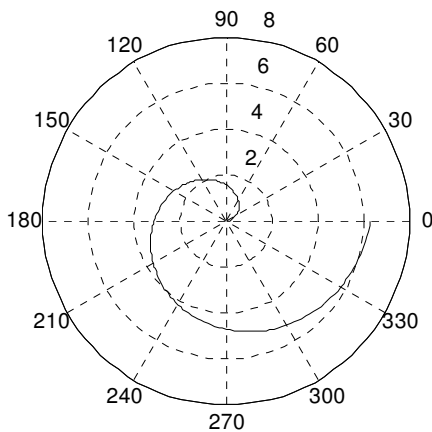
Tất nhiên là có sự chuyển đổi giữa tọa độ cực và tọa độ đề các theo công thức

$$x = r \cos \varphi = a\varphi \cos \varphi, \quad y = r \sin \varphi = a\varphi \sin \varphi.$$

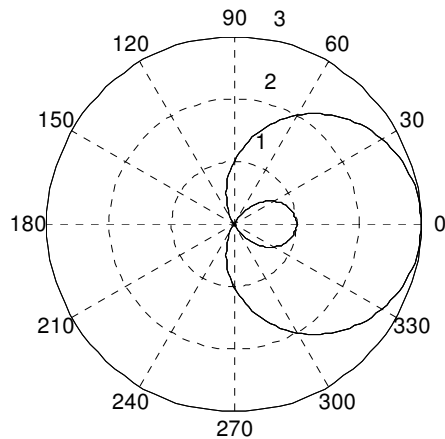
Để vẽ đường xoắn ốc này trong phạm vi $0 \leq \varphi \leq 2\pi$, ta thực hiện việc chia đều miền cần vẽ, tính toán các giá trị của r tương ứng và sử dụng lệnh *polar* như trong các dòng lệnh sau:

```
>> a = 1;
>> phi=[0:pi/90:2*pi];
>> r = a*phi;
>> polar(phi, r, '-k');
```

Kết quả cho ta hình vẽ 3-20.



Hình 3-20. Đường xoắn ốc Ácximét



Hình 3-21. Đồ thị tọa độ cực của phương trình $r = 1 + 2 \cos \theta$

Một ví dụ khác là vẽ đồ thị hàm $r = 1 + 2 \cos \theta$ trong tọa độ cực, với $0 \leq \theta \leq 6\pi$. Các dòng lệnh sau thực hiện công việc này

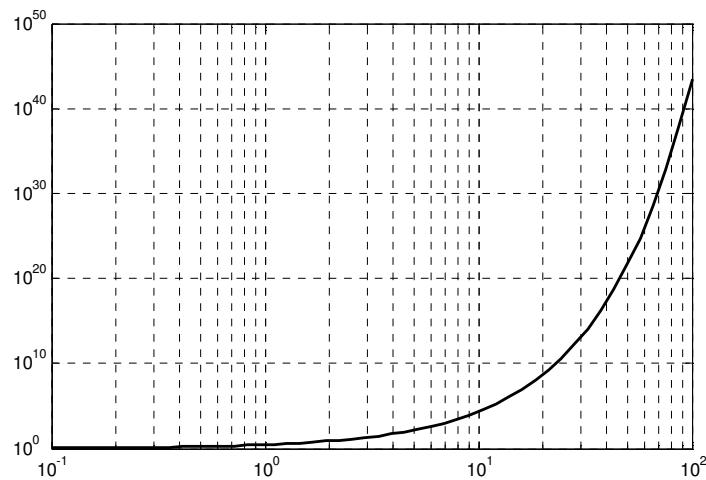
```
>> theta=[0:pi/90:6*pi];
>> r = 1+2*cos(theta);
>> polar(r,theta,'-k');
```

Kết quả cho ta hình vẽ 3-21.

Trong kỹ thuật điều khiển, các đường đặc tính biên độ và góc pha theo tần số tác động thường được biểu diễn theo thang logarith. Để vẽ các đồ thị với thang logarith này, chúng ta có thể sử dụng kiểu vẽ log-log bằng lệnh loglog. Lệnh vẽ loglog này được sử dụng tương tự như lệnh plot, chỉ khác là trong khi lệnh plot sử dụng thang đo thường (thang thập phân) còn lệnh loglog lại sử dụng thang đo log trên cả hai trục tung và hoành. Nghĩa là với hàm số $y = f(x)$, thì đồ thị sẽ hiển thị các giá trị $\log(x)$ và $\log(y)$ trên hai trục. Ví dụ ta sẽ vẽ hàm $y = e^x$, $0.1 \leq x \leq 100$ trong thang logarith bằng các dòng lệnh

```
>> x = logspace(-1,2);
>> loglog(x,exp(x),'k-'), grid on
```

Kết quả nhận được như trên hình 4-22.



Hình 3-22. Đồ thị hàm $y = e^x$ trong thang loglog

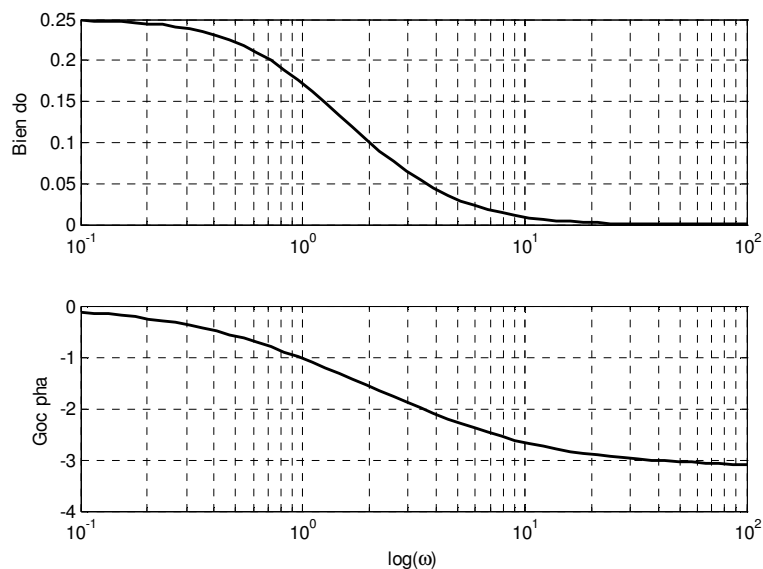
Ngoài ra ta có thể vẽ các đồ thị với lệnh *semilogx* hoặc *semilogy*, với các lệnh này chỉ có một trục x hoặc y tương ứng sử dụng thang logarith còn trục kia sử dụng thang thập phân. Trong ví dụ sau đây ta sẽ vẽ đồ thị biên độ và góc pha của các hàm biến phức sau:

$$G_1(s) = \frac{1}{1+s}, \quad G_2(s) = \frac{4}{(1+s)(s+4)}$$

với s là biến phức $s = i\omega$, $0.1 \leq \omega \leq 100$.

```
>> s = logspace(-1,2)*i;
>> G2 = 1./((s+1).*(s+4));
>> absG2 = abs(G2); pha2 = phase(G2);
>> subplot(2,1,1)
>> semilogx(imag(s), absG2, 'k-', 'Linewidth', 2), grid on
>> ylabel('Bien do')
>> subplot(2,1,2)
>> semilogx(imag(s), pha2, 'k-', 'Linewidth', 2), grid on
>> xlabel('log(\omega)'), ylabel('Goc pha')
```

Kết quả cho ta hình vẽ 3-23.



Hình 3-23. Đồ thị độ lớn và góc pha của hàm phức $G_2(s)$ trong thang semilogx

Trong kỹ thuật điều khiển, khi nghiên cứu đáp ứng biên độ tần số và góc pha tần số, người ta thường sử dụng thang đo log đối với trục hoành, và trục tung được tính theo dexibel, dB đối với đồ thị biên độ tần số. Còn đối với đồ thị góc pha, trục tung được thể hiện bằng độ. Trong ví dụ sau ta sẽ vẽ hai đồ thị dạng này với subplot và semilogx.

Hàm biên độ tần số

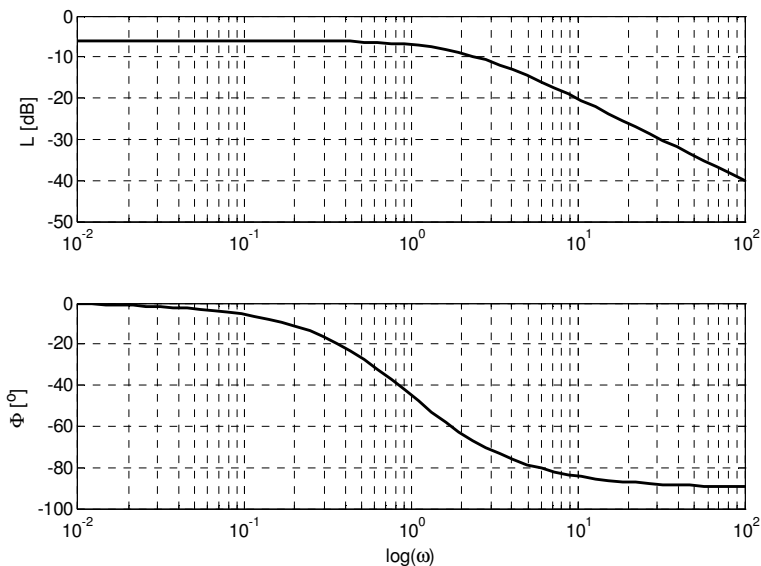
$$L(\omega) = -20 \lg \sqrt{4 + \omega^2}$$

Hàm góc pha

$$\phi(\omega) = -\arctan(\omega T)$$

Thực hiện các dòng lệnh:

```
>> T=1;
>> x = logspace(-2,2);
>> L = -20*log10(sqrt(4+x.^2));
>> pha = -atand(x*T);
>> semilogx(x,L)
>> semilogx(x,pha)
>> subplot(2,1,1)
>> semilogx(x, L, 'k-', 'Linewidth', 2), grid on
>> ylabel('L [dB]')
>> subplot(2,1,2)
>> semilogx(x, pha, 'k-', 'Linewidth', 2), grid on
>> xlabel('log(\omega)'), ylabel('\Phi [^\circ]')
```



Hình 3-24. Đồ thị độ lớn và góc pha của hàm phức $G_2(s)$ trong thang semilogx

Đồ thị của dữ liệu rời rạc

Không phải lúc nào ta cũng có hoặc tạo ra các dữ liệu liên tục, khi cần phải thể hiện bằng biểu đồ, đồ thị của các đại lượng rời rạc ta làm thế nào? Chẳng hạn như muốn thể hiện bằng biểu đồ điểm tổng của năm bạn sinh viên An, Nam, Thu, Đông và Hạ với số điểm tương ứng là 67, 56, 97, 80 và 98. Trước hết, ta tạo hai mảng, một mảng chứa danh sách sinh viên và mảng kia chứa điểm số tương ứng. Danh sách sinh viên đóng vai trò như biến x, nên ta tạo một mảng với năm phần tử:

```
>> x = [1:5];
```

Bởi vì chúng ta không mô hình bằng một hàm liên tục, nên không cần thiết phải xác định bước tăng của x. Ta có bước bằng 1, như thế Matlab tạo ra 5 điểm từ 1 đến 5. Sau đó ta tạo véc tơ y với 5 phần tử chứa điểm số tương ứng:

```
>> y = [67, 56, 97, 80, 98];
```

Và bây giờ ta có thể vẽ đồ thị, và đưa tên các bạn sinh viên vào bằng lệnh *set*. Việc này như có vẻ tương đối phức tạp:

```
>> plot(x,y,'o',x,y),
>> set(gca,'XTicklabel',['An'; 'Nam'; 'Thu'; 'Dong'; 'Ha']),
>> set(gca,'XTick',[1:5]),
>> axis([1 5 0 100]),xlabel('Sinh vien'),
>> ylabel('Diem so'), title('Diem thi thang 10, 2010')
```

Khi chạy các dòng này, Matlab sẽ báo lỗi.

```
??? Error using ==> vertcat
```

```
All rows in the bracketed expression must have the same
number of columns.
```

Lý do nằm ở dòng,

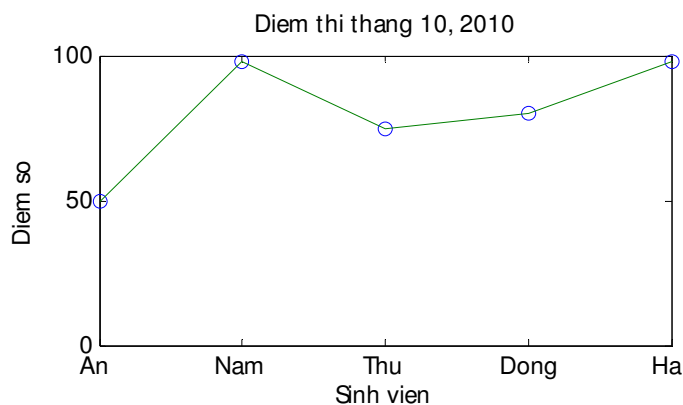
```
>> set(gca,'XTicklabel',['An'; 'Nam'; 'Thu'; 'Dong'; 'Ha']),
```

ở đây do tên của các bạn sinh viên có số ký tự khác nhau, để hiển thị được các tên này lên, ta xem tên dài nhất có bao nhiêu ký tự, sau đó thêm ký tự trắng vào các tên ngắn hơn sao cho số ký tự trong tất cả các tên là như nhau. Trong trường hợp ở đây, 'Dong' là tên dài nhất có 4 ký tự, do đó tên 'An' cần thêm 2, tên 'Nam' thêm 1, tên 'Thu' thêm 1, và tên 'Ha' thêm 2.

Sửa dòng lệnh trên thành

```
>> set(gca,'XTicklabel',['An  '; 'Nam '; 'Thu '; 'Dong'; 'Ha  ']),
```

Ta sẽ nhận được đồ thị biểu diễn như hình 4-25.



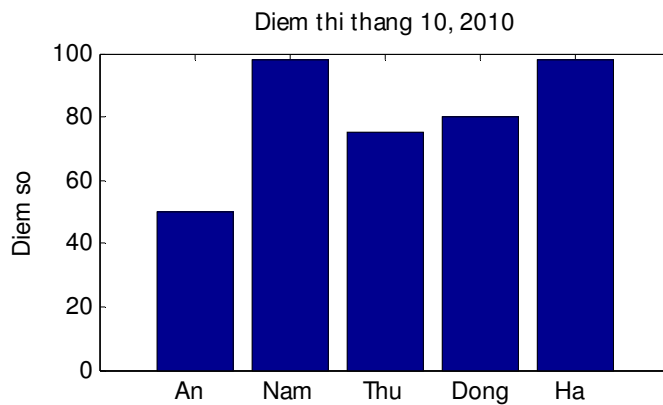
Hình 3-25. Biểu diễn điểm số sinh viên với lệnh set

```
>> plot(x,y,'o',x,y),
>> set(gca,'XTicklabel',{'An ','Nam ','Thu ','Dong','Ha '}),
>> set(gca,'XTick',[1:5]),
>> axis([1 5 0 100]),xlabel('Sinh vien'),
>> ylabel('Diem so'), title('Diem thi thang 10, 2010')
```

Chúng ta cũng có thể biểu diễn các điểm trên bằng các cột hai chiều với lệnh `bar(x,y)`. Lệnh vẽ cùng với các nhãn tên như sau

```
>> x = [1:5];
>> y = [50,98,75,80,98];
>> bar(x,y), % xlabel('Sinh vien'),
>> set(gca,'XTicklabel',{'An ','Nam ','Thu ','Dong','Ha '}),
>> ylabel('Diem so'), title('Diem thi thang 10, 2010')
```

Kết quả nhận được là biểu đồ như trên hình 4-26



Hình 3-26. Biểu diễn điểm số sinh viên với lệnh `bar`

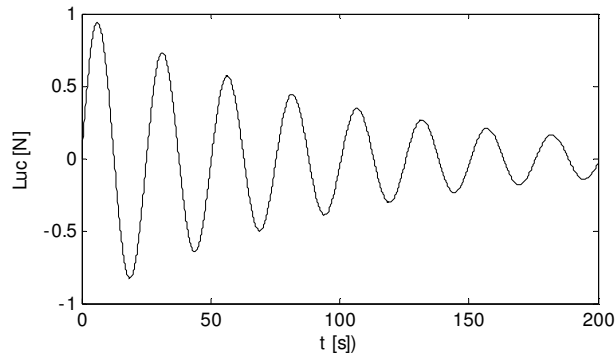
Một cách khác để biểu diễn các điểm dữ liệu rời rạc là *stem plot*. Lệnh `stem plot` tạo ra đồ thị của một hàm với dữ liệu của các điểm nhất định. Tại mỗi điểm tọa độ (x,y) được đánh dấu bằng marker, và một đường thẳng đứng kéo từ marker đến trục hoành. Để làm ví dụ, ta xét hàm đáp ứng của một lò xo khi có lực tác dụng như sau:

$$f(t) = e^{-\beta t} \sin(t/4), \quad \beta = 0.01, \quad 0 \leq t \leq 200$$

Trước hết hàm này được vẽ bằng lệnh `plot` với bước thời gian $h = 0.1$. Thực hiện các dòng lệnh

```
>> t = [0:0.1:200];
>> f = exp(-0.01*t).*sin(t/4);
>> plot(t,f, 'k-'), xlabel('t [s]'),ylabel('Luc [N]')
```

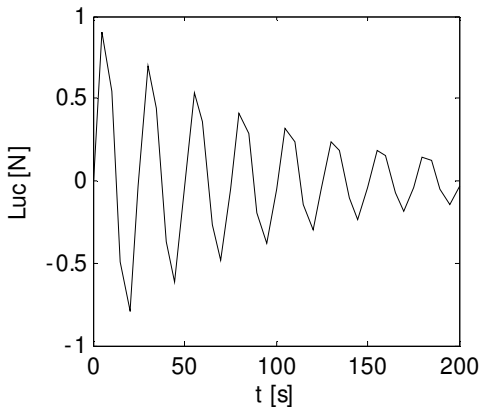
ta nhận được đồ thị như trên hình 4-27, đó là dao động tắt dần.



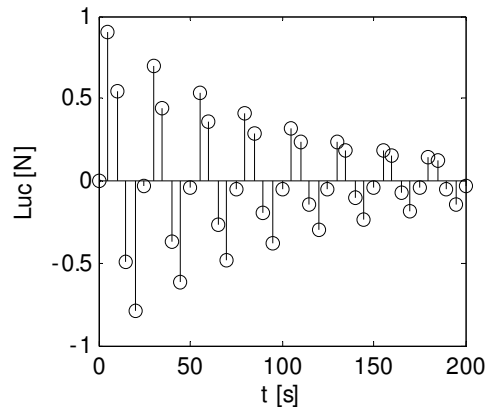
Hình 3-27. Vẽ hàm $f(t)$ với lệnh plot

Để tạo ra được dữ liệu rời rạc, ta lấy bước thời gian tương đối lớn $h = 5$ s, (giả sử rằng tín hiệu cứ sau 5 giây lại được lấy mẫu). Sau đó thực hiện vẽ lại các tín hiệu với các lệnh plot và stem như dưới đây, và thu được các đồ thị tương ứng (hình 4-28a và 3-28b).

```
>> t = [0:5:200];
>> f = exp(-0.01*t).*sin(t/4);
>> plot(t,f,'k-'), xlabel('t [s]'),ylabel('Luc [N]')
>> stem(t,f,'k-'), xlabel('t [s]'),ylabel('Luc [N]')
```



Hình 3-28a. Đồ thị tín hiệu với plot



Hình 3-28b. Đồ thị tín hiệu với stem

Chú ý rằng, các lựa chọn đối với kiểu đường khi sử dụng plot(x,y) đều có thể áp dụng khi sử dụng stem. Ví dụ bạn muốn lựa chọn đường nét liền, đường chấm gạch, . . . hay muốn chọn màu cho các đường (xanh dương là màu mặc định). Chúng ta cũng có thể bôi màu cho các dấu đánh trên hình bằng việc lựa chọn 'fill' trong stem. Và ta cũng có thể chọn tùy ý các kiểu đánh dấu (marker) bao gồm dấu

vuông (s), kim cương (d), sao năm điểm (p), vòng tròn (c), gạch chéo (x), dấu sao (*), và chấm (.). Bạn hãy thử với dòng lệnh sau và xem đồ thị !

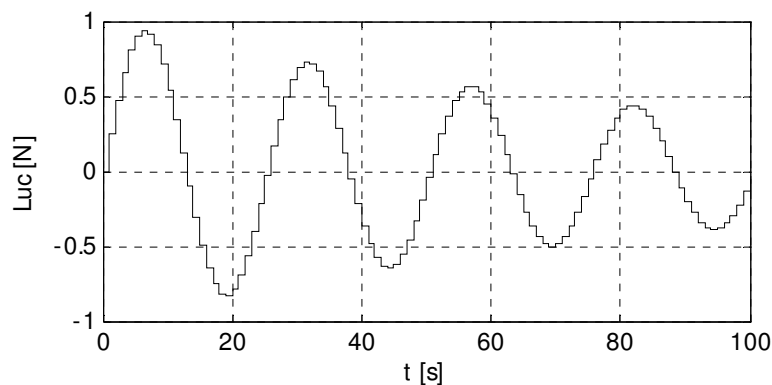
```
>> stem(t,f, '--db', 'fill'), xlabel('t [s]'),ylabel('Luc [N]')
```

Nếu muốn thể hiện một tín hiệu lấy mẫu và được giữ là hằng số cho đến lần lấy mẫu tiếp theo, ta sử dụng lệnh `stairs(x,y, ...)`. Ví dụ tín hiệu liên tục trên được lấy mẫu với chu kỳ 1 s (mỗi giây lấy mẫu một lần), và được giữ là hằng, thì ta biểu diễn với các dòng lệnh sau:

```
>> f = exp(-0.01*t).*sin(t/4);
```

```
>> stairs(t,f,'k-'), xlabel('t [s]'),ylabel('Luc [N]'), grid on
```

Kết quả cho ta một đồ thị như sau



Hình 3-29. Đồ thị tín hiệu với stairs

Vẽ biểu đồ với lệnh `contour` – vẽ đường đồng mức

Khi vẽ đồ thị của hàm hai biến ta có rất nhiều cách thể hiện khác nhau, nếu thể hiện nó trong một mặt phẳng thì sử dụng các đường đồng mức là cách thể hiện thích hợp. Cách thể hiện này giống như khi ta chụp ảnh từ trên xuống một quả đồi có ruộng bậc thang. Về mặt toán học, giả sử cần thể hiện các đường đồng mức của hàm hai biến $z = f(x, y)$, trong một miền hình chữ nhật $a \leq x \leq b$, $c \leq y \leq d$, ta thực hiện các bước sau: Trước hết chia miền chữ nhật trên bằng một lưới các điểm chia (x_i, y_j) bằng lệnh `meshgrid`. Kết quả của lệnh `meshgrid` đối với hai biến độc lập là một ma trận chứa các điểm chia với bước chia định nghĩa cho từng biến. Giả sử các biến nằm trong miền $-5 \leq x \leq 5$, $-3 \leq y \leq 3$, với bước chia 0.1 cho cả hai biến ta sẽ gọi lệnh `meshgrid` như sau:

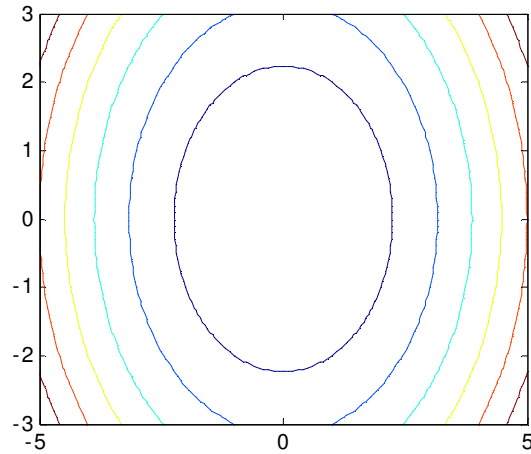
```
>> [x,y] = meshgrid(-5:0.1:5,-3:0.1:3);
```

Tiếp đó là tính giá trị của hàm f tại các điểm chia bằng cách sử dụng các phép tính phần tử (`.* ./ .^`):

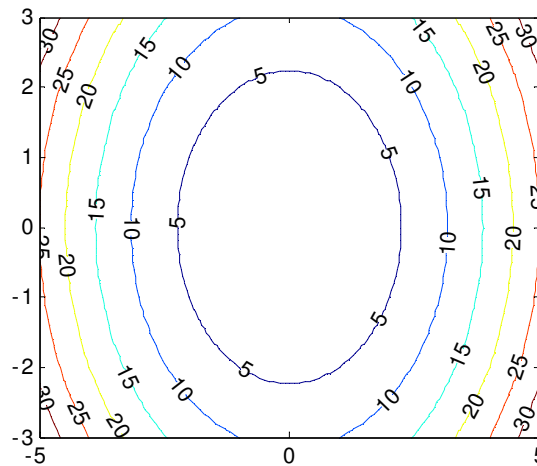
```
>> z = x.^2 + y.^2;
```

Sau khi gọi lệnh `contour`, Matlab sẽ đưa ra các đường đồng mức với các màu khác nhau thể hiện các giá trị của hàm số.

```
>> contour(x,y,z)
```



Hình 4-30. Các đường đồng mức của hàm $z = x^2 + y^2$



Hình 4-31. Thêm giá trị cho các đường mức của hàm $z = x^2 + y^2$

Tuy nhiên nhìn vào hình trên ta chưa biết giá trị của mỗi mức là bao nhiêu, để thấy rõ hơn, ta sẽ bổ sung thêm các giá trị của hàm z tại mỗi mức bằng lệnh `set`. Các giá trị về độ cao của mỗi mức cùng với màu của đường vẽ được lấy ra nhờ phép gán sau:

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

Từ đó ta có thể ghi giá trị các mức lên đồ thị nhờ lệnh `set`:

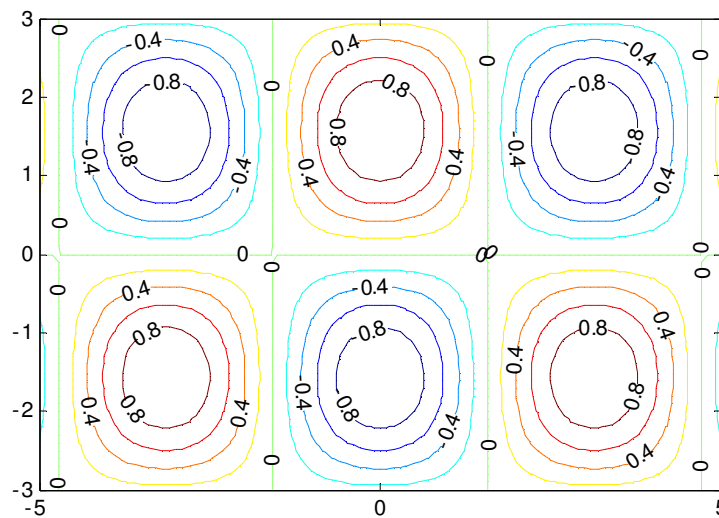
```
>> set(h, 'ShowText', 'on', 'TextStep', get(h, 'LevelStep') * 2)
```

Với dòng lệnh này Matlab đưa ra cho ta đồ thị (hình 4-31).

Trong ví dụ sau ta vẽ các đường đồng mức của hàm $z = \cos(x)\sin(y)$ bằng các dòng lệnh:

```
>> [x,y] = meshgrid(-5:0.1:5,-3:0.1:3);
>> z = cos(x).*sin(y);
>> [C,h] = contour(x,y,z);
>> set(h, 'ShowText', 'on', 'TextStep', get(h, 'LevelStep') * 2)
```

Kết quả của các lệnh trên là hình 4-32.



Hình 4-32. Các đường đồng mức của hàm $z = \cos x \sin y$

Tiếp theo đây là sử dụng lệnh *contour3*, nếu sử dụng lệnh với hai tham số *contour3(z, n)* ta sẽ nhận được bản đồ với *n* mức. Ví dụ đối với hàm $z = \cos(x)\sin(y)$, nếu sử dụng lệnh

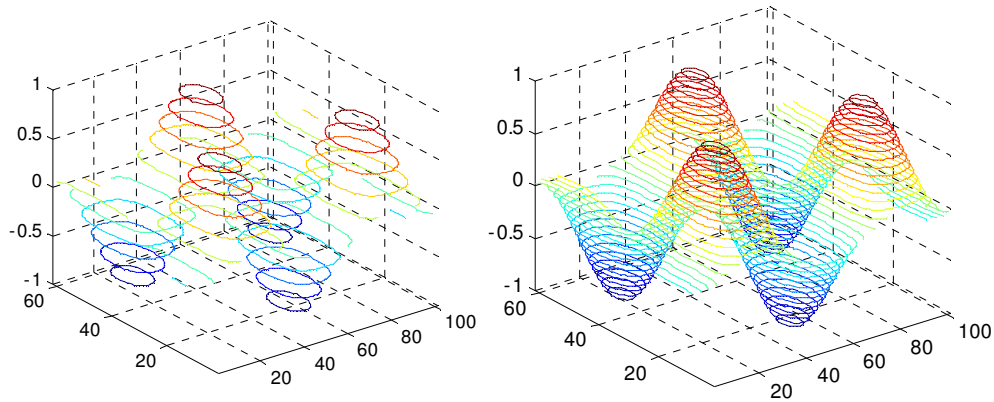
```
>> contour3(z,10) % hoặc >> contour3(x,y,z,30);
```

kết quả như trên hình 4-33.

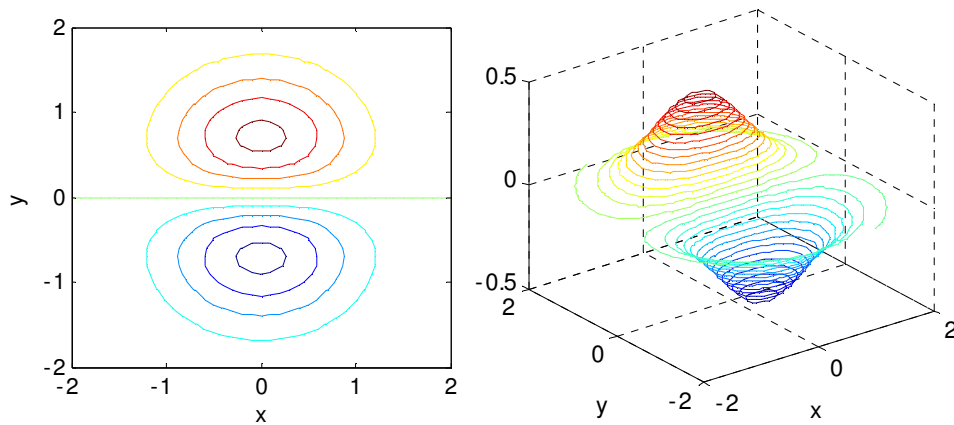
Với *contour3*, các đường mức hiện ra trực quan hơn, nhưng đôi khi vẫn chưa đáp ứng được yêu cầu. Để có hình ảnh chuyên nghiệp hơn ta cần Matlab cung cấp nhiều thông tin hơn của đồ thị. Trong ví dụ sau ta xét hàm $z = ye^{-(x^2+y^2)}$ trong miền $-2 \leq x, y \leq 2$. Trước hết sử dụng lệnh *contour* và *contour3* cho ta các hình 4-34.

```
>> [x,y] = meshgrid(-2:0.1:2);
>> z = y.*exp(-x.^2-y.^2);
>> contour(x,y,z), xlabel('x'), ylabel('y')
```

```
>> contour3(z,30),xlabel('x'),ylabel('y')
```



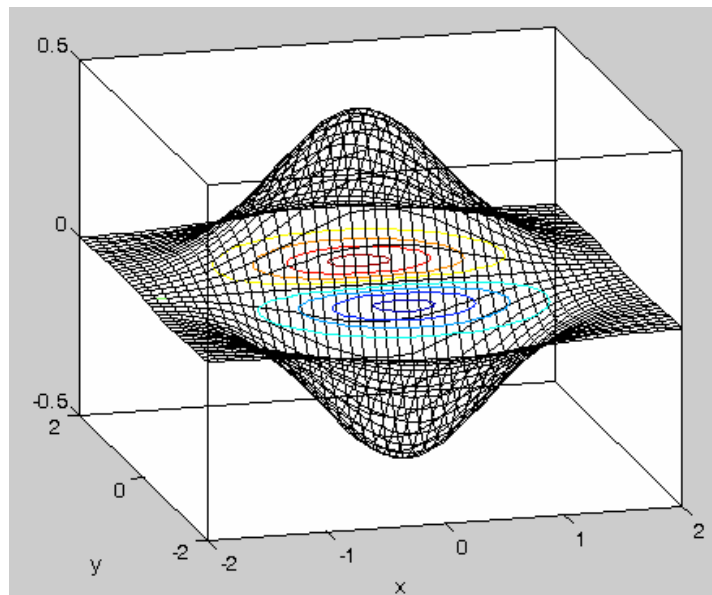
Hình 4-33. Lệnh `contour3` đối với hàm $z = \cos x \sin y$, $n = 10$ và $n=30$



Hình 4-34. Lệnh `contour` và `contour3` đối với hàm $z = ye^{-(x^2+y^2)}$

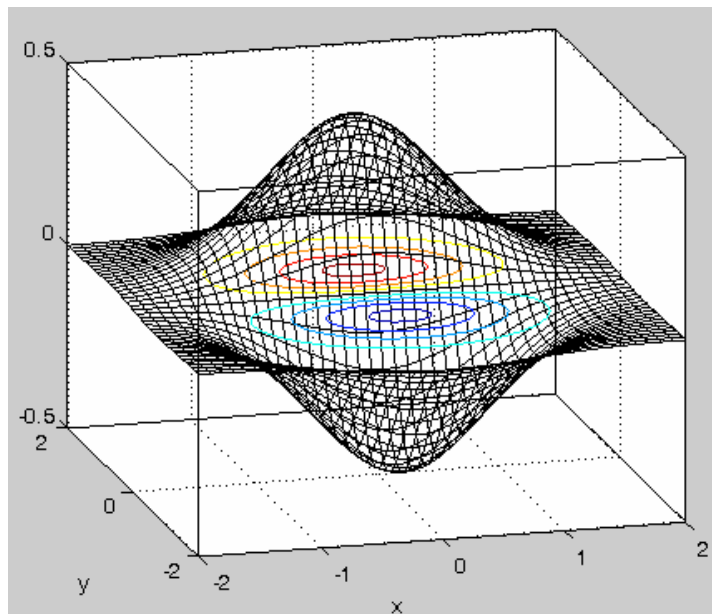
Bằng cách thêm vào lệnh `surface` sau đây, chúng ta có thể nhận được một hình ảnh đẹp hơn như sau, (hình 4-35a và b).

```
>> [x,y] = meshgrid(-2:0.1:2);
>> z = y.*exp(-x.^2-y.^2);
>> surface(x,y,z,'EdgeColor',[.8 .8 .8],'FaceColor','none'),
>> surface(x,y,z,'EdgeColor',[0 0 0],'FaceColor','none'),
>> grid off, view(-15,20)
```

Hình 4-35a. Lệnh surface đối với hàm $z = ye^{-(x^2+y^2)}$, grid off

```
>> surface(x,y,z,'EdgeColor',[0 0 0],'FaceColor','none'),
>> grid on, view(-15,20)
```



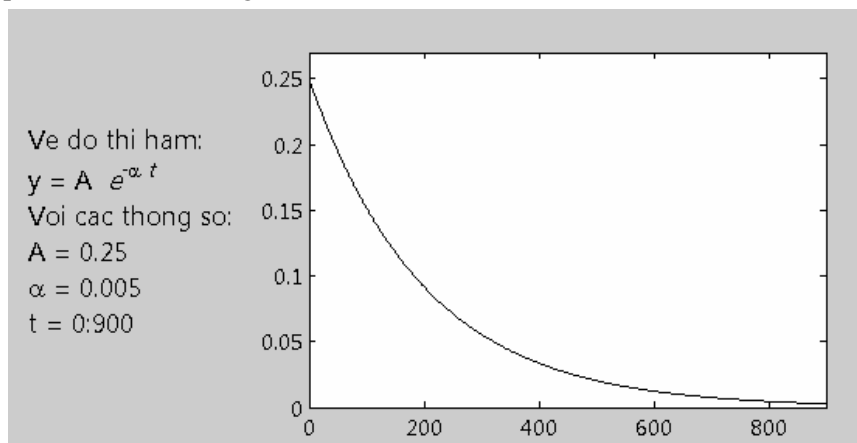
Hình 4-35b. Lệnh surface đối với hàm $z = ye^{-(x^2+y^2)}$, grid on

Thêm chú thích trên đồ thị

Khi muốn thể hiện chú thích bên ngoài cửa sổ của đồ thị, ta thực hiện như ví dụ dưới đây.

```
figure(1), clf
h=axes('Position',[0 0 1 1], 'Visible','off');
% Xac lap he truc 1
axes('Position',[0.35 0.1 0.6 0.8]);
% Xac lap he truc 2
t = 0:900; % vector chua thoi gian
plot(t,0.25*exp(-0.005*t),'k','lineWidth',1); % ve ham y(t)
axis([0 max(t) 0 0.27]);
% dat cac dong chu va dinh vi tren cua so do thi
str(1) = {'Ve do thi ham:'}; % hoac str(1) = 'Ve do thi ham: ';
str(2) = {'y = A {\it e}^{\alpha {\it t}}'};
str(3) = {'Voi cac thong so:'};
str(4) = {'A = 0.25'};
str(5) = {'\alpha = 0.005'};
str(6) = {'t = 0:900'};
set(gcf,'CurrentAxes',h);
text(0.025, 0.5, str,'FontSize',12);
```

Kết quả cho ta đồ thị cùng với các chú thích như hình 3-36.



Hình 3-36. Đồ thị hàm số cùng các chú thích

Vẽ đồ thị các hàm có điểm không xác định

Giả sử cần vẽ đồ thị hàm $f(x)$ trong khoảng $[a, b]$, nhưng trong khoảng này hàm số không xác định tại 2 điểm c và d . Để tránh các điểm kỳ dị, trong trường hợp này ta chia khoảng $[a, b]$ thành ba đoạn nhỏ $[a, c - \varepsilon]$, $[c + \varepsilon, d - \varepsilon]$, và $[d + \varepsilon, b]$, với ε là số dương bé. Sau đó vẽ đồ thị hàm số trên các đoạn nhỏ đó.

Chẳng hạn ta cần vẽ đồ thị hàm sau đây trong đoạn $[a, b] = [0, 3]$

$$f(x) = \frac{4(1-x^2)}{(10-7x^2)(1-x^2)-2}$$

thấy rằng trong đoạn này hàm số không xác định tại hai điểm

$$x_1 = c = 0.7990, \quad x_2 = d = 1.3380$$

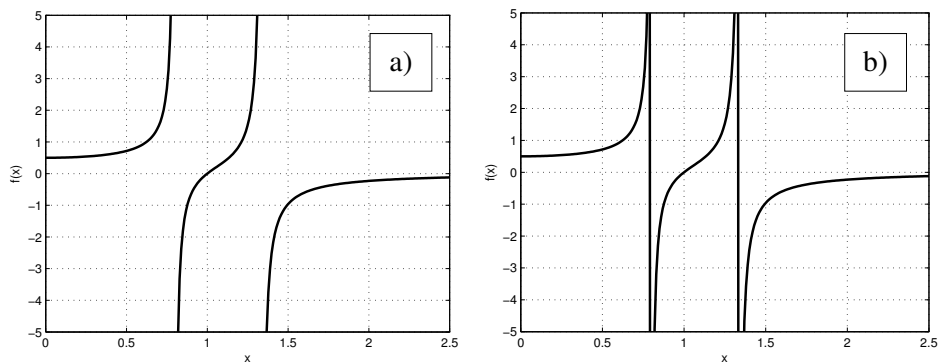
```
% matlab code in a m-file
a = 0;          b = 2.5;    c = 0.7990;    d = 1.3380;
dx = 0.01;      epsi = dx;
x1 = [a:dx:c-epsi]; x2 = [c+epsi:dx:d-epsi]; x3 = [d+epsi:dx:b];

for i=1:length(x1), f1(i)=f(x1(i)); end
for i=1:length(x2), f2(i)=f(x2(i)); end
for i=1:length(x3), f3(i)=f(x3(i)); end

mif=min([min(f1), min(f2), min(f3)]);
mxm=max([max(f1), max(f2), max(f3)]);
plot(x1,f1,'k-', x2,f2, 'k-', x3,f3,'k-','LineWidth',2), grid on
xlabel('x'), ylabel('f(x)'), axis([a, b, -5, 5])
axis on fill
```

Kết quả là hình 3-37a, nếu không sử dụng kỹ thuật chia miền thành các đoạn nhỏ thì đồ thị nhận được sẽ như hình 3-37b.

```
% matlab code in a m-file
a = 0;    b = 2.5;    dx = 0.01;
% không chia thành các đoạn nhỏ
x=[a:dx:b];
for i=1:length(x), y(i)=f(x(i)); end
plot(x, y, 'k-', 'LineWidth',2), grid on
xlabel('x'), ylabel('f(x)'), axis([a, b, -5, 5])
```



Hình 3-37. Đồ thị hàm số có điểm không xác định

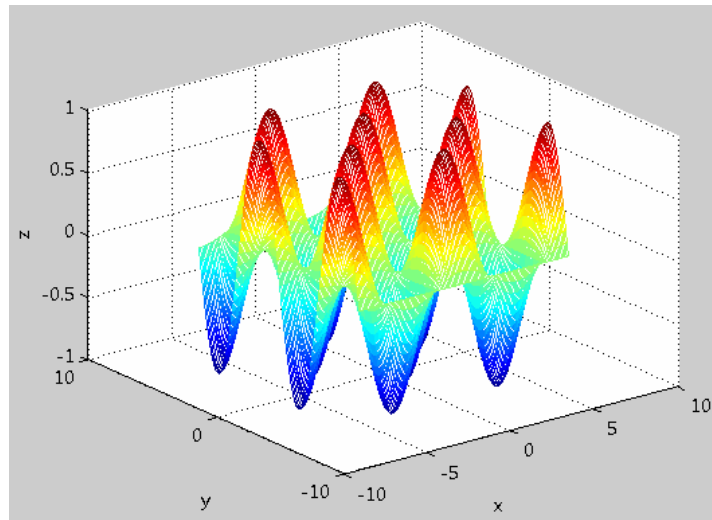
4.2 Các lệnh vẽ trong không gian – 3D

Như trên đã trình bày có thể sử dụng lệnh `surface` để thể hiện hình ảnh 3D. Trong Matlab còn có nhiều khả năng khác để thể hiện đồ họa 3D, chúng ta có thể sử dụng các lệnh như liệt kê trong bảng 3-4.

Bảng 3-4. Các lệnh hiển thị đồ họa 3D	
<code>plot3(x,y,z <,plotstil>)</code>	<code>surfl(x,y,z <,color>)</code>
<code>comet3(x,y,z <,comet length>)</code>	<code>patch(x,y <,z> ,color)</code>
<code>mesh(x,y,z <,color>)</code>	<code>waterfall(x,y,z...<,...>,...)</code>
<code>surf(x,y,z <,color>)</code>	<code>contour3(x,y,z... <,...>,...)</code>
<code>surfc(x,y,z <,color>)</code>	<code>contour(x,y,z... <,...>,...)</code>
<code>[X, Y]= meshgrid(x_vector, y_vector)</code>	
<code>box <on off></code>	
<code>view(horizontal, vertical)</code>	
<code>zlabel(string)</code>	

Trong ví dụ sau, ta sẽ minh họa việc sử dụng lệnh `mesh` để thể hiện một hàm hai biến trong không gian, $z = \cos x \sin y$, với $-2\pi \leq x, y \leq 2\pi$. Đưa vào các dòng lệnh sau

```
>> [x,y] = meshgrid(-2*pi:0.1:2*pi);
>> z = cos(x).*sin(y);
>> mesh(x,y,z),xlabel('x'),ylabel('y'),zlabel('z')
```



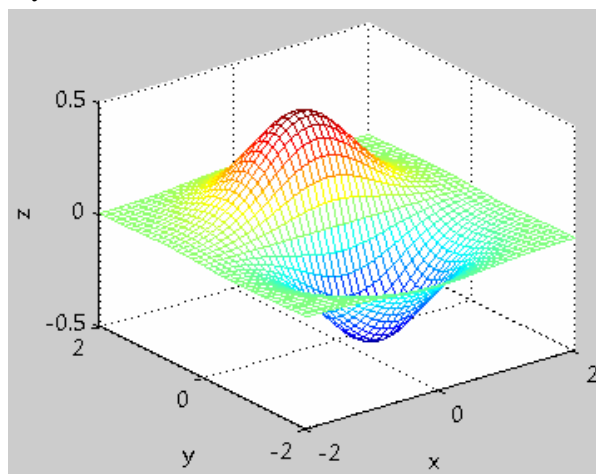
Hình 3-38. Đồ thị hàm $z = \cos x \sin y$ với lệnh `mesh`

Quan sát dòng lệnh cuối cùng, ta thấy *mesh* như là một sự mở rộng của lệnh *plot(x,y)* vào không gian 3D. Kết quả của các dòng lệnh trên được đưa ra như trên hình 4-38.

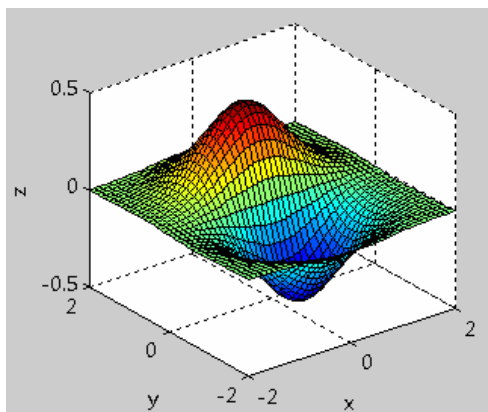
Dưới đây là sử dụng *mesh* để thể hiện hàm $z = ye^{-x^2-y^2}$ với miền giới hạn $-2 \leq x, y \leq 2$. Chúng ta đưa vào các dòng lệnh

```
>> [x,y] = meshgrid(-2:0.1:2);
>> z = y.*exp(-x.^2-y.^2);
>> mesh(x,y,z), xlabel('x'), ylabel('y'), zlabel('z')
```

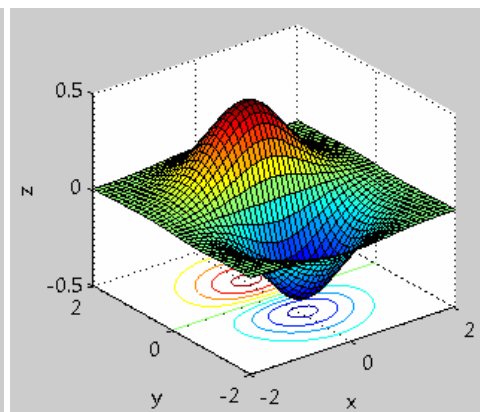
Đồ thị của hàm này thể hiện như trên hình 4-39.



Hình 3-39. Đồ thị hàm $z = ye^{-x^2-y^2}$ với lệnh *mesh*



Hình 3-40. Đồ thị hàm $z = ye^{-x^2-y^2}$ với lệnh *surf(x,y,z)*



Hình 3-41. Đồ thị hàm $z = ye^{-x^2-y^2}$ với lệnh *surfc(x,y,z)*

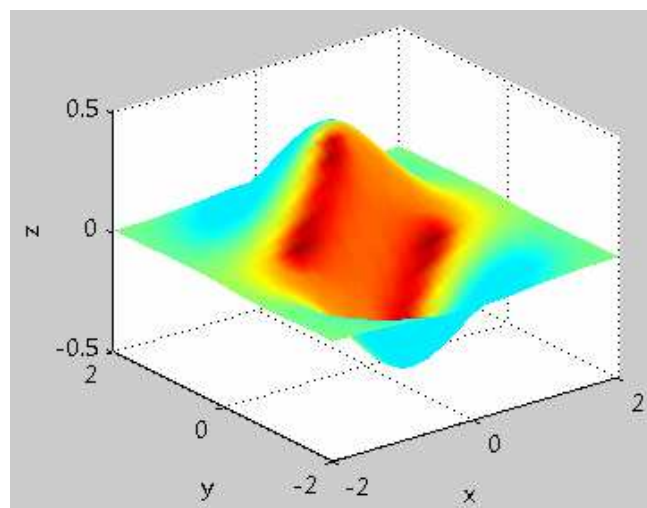
Để hiển thị màu trên các ô lưới của bề mặt đồ thị, ta sử dụng lệnh *surf* hoặc *surfc*. Thật đơn giản, ta thay thế lệnh *mesh* trên bởi *surf* hoặc *surfc* như dòng dưới đây:

```
>> surf(x,y,z), xlabel('x'), ylabel('y'), zlabel('z')
>> surfc(x,y,z), xlabel('x'), ylabel('y'), zlabel('z')
```

Kết quả cho ta đồ thị như hình 4-40. Trong hình vẽ trên màu của các ô lưới được tô tỷ lệ với chiều cao (tọa độ z) của các điểm. Nếu sử dụng lệnh *surf* ta nhận được hình 4-41.

Nếu sử dụng lệnh *surf* (chữ 'l' cho biết bề mặt được chiếu sáng) cho ta một bề mặt trơn, không hiển thị các đường chia lưới, với lựa chọn màu hoặc xám. Khi áp dụng lệnh *surf* đối với hàm $z = ye^{-x^2-y^2}$ cho ta đồ thị như trên hình 4-42.

```
>> surf(x,y,z), xlabel('x'), ylabel('y'), zlabel('z')
>> shading interp
>> color map(gray);
```



Hình 3-42. Đồ thị hàm $z = ye^{-x^2-y^2}$ với lệnh *surf*(x,y,z)

Bề mặt được đánh bóng với các lựa chọn *flat*, *interp*, hoặc *faceted*. Nếu chọn *flat* thì giá trị màu của mỗi ô là hằng số và không hiện các đường lưới. Lựa chọn *interp* báo cho Matlab biết cần nội suy màu trong các ô để có được bản đồ màu được trơn liên tục.

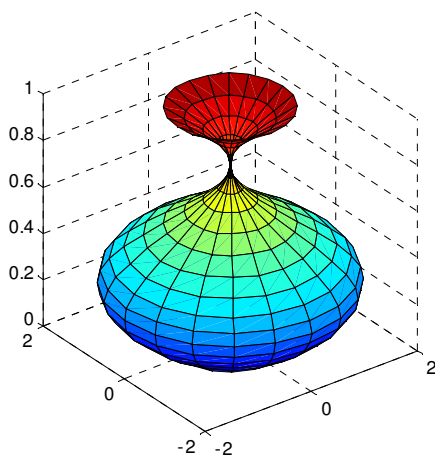
Để vẽ các mặt tròn xoay hay mặt cầu, ta sử dụng các lệnh *cylinder* hoặc *sphere*. Các dòng lệnh dưới đây minh họa công việc đó.

```
>> t = 0:pi/10:2*pi;
>> [X,Y,Z] = cylinder(1+sin(t));
>> surf(X,Y,Z), axis square
```

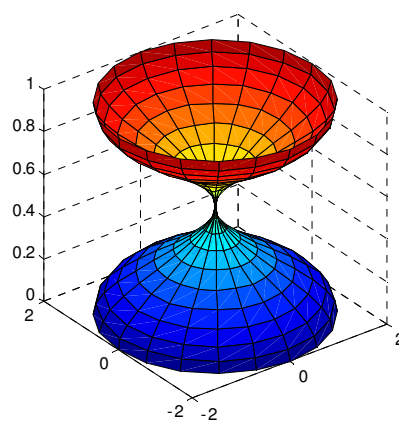
Kết quả cho ta một hình ảnh trông như giọt nước, Hình 4-43. Một ví dụ khác với lệnh *cylinder* và *faceted shading*.

```
>> t = 0:pi/10:2*pi;
>> [X,Y,Z] = cylinder(1+cos(t));
>> surf(X,Y,Z), axis square
>> shading faceted
```

Kết quả là hình 4-44.



Hình 4-43. Tạo hình ảnh bằng lệnh *cylinder* và lựa chọn flat shading



Hình 4-44. Lệnh *cylinder* vẽ hàm $1+\cos(t)$ với faceted shading

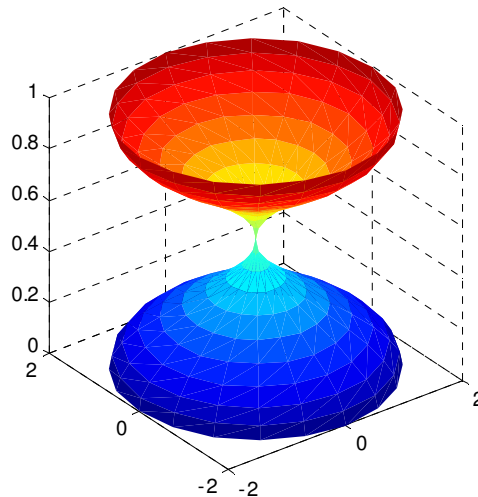
Nếu sử dụng lựa chọn *shading interp*, chúng ta sẽ nhận được hình ảnh như trên hình 4-45.

```
>> t = 0:pi/10:2*pi;
>> [X,Y,Z] = cylinder(1+cos(t));
>> surf(X,Y,Z), axis square, shading interp
```

Có rất nhiều khả năng biểu diễn đồ họa trọng không gian ba chiều (3D) từ đồ thị các điểm, các đường đơn giản cho đến những đối tượng gần như thực tế. Ở đây ta chỉ có thể giới thiệu phạm vi ứng dụng rất hẹp, một số lệnh được giới thiệu và làm

sáng tỏ bằng các thí dụ cụ thể. Để có cái nhìn tổng quan về đồ họa 3D, bạn đọc có thể tham khảo từ trợ giúp ngay trên màn hình Matlab với lệnh:

```
>> help graph3d
```



Hình 4-45. Lệnh cylinder vẽ hàm $1+\cos(t)$ với shading interp

Ngoài ra sự trợ giúp trực tuyến (online help) mô tả chi tiết các lệnh với nhiều ví dụ, đặc biệt trong phần tìm kiếm theo chủ đề (search) bạn có thể tìm được các chỉ dẫn theo yêu cầu. Chẳng hạn như *plot3* cho phép vẽ các đường đồ thị và các đánh dấu trong không gian 3D.

```
>> t = linspace(0, 4*pi, 100);
>> plot3(sin(t), cos(t), t, '-k');
```

Ví dụ vẽ một đường trong không gian 3D. Chẳng hạn ta cần thể hiện dao động cường bức hệ khối lượng – lò xo

$$x(t) = A \cos(\Omega t - \psi), \quad \dot{x}(t) = -A\Omega \sin(\Omega t - \psi)$$

trong không gian chuyển động (t, x, \dot{x}) . Ngoài ra, ta còn muốn thể hiện các hình chiếu của đường cong 3D trên các mặt phẳng (t, x) và (t, \dot{x}) cũng như trong mặt phẳng pha (x, \dot{x}) . Các công việc này sẽ được thực hiện trong m-file sau:

```
om_0=10; % tan so dao dong rieng khong can
q=1; eta_0=1;
Omega=eta_0*om_0; % tan so kích động
```

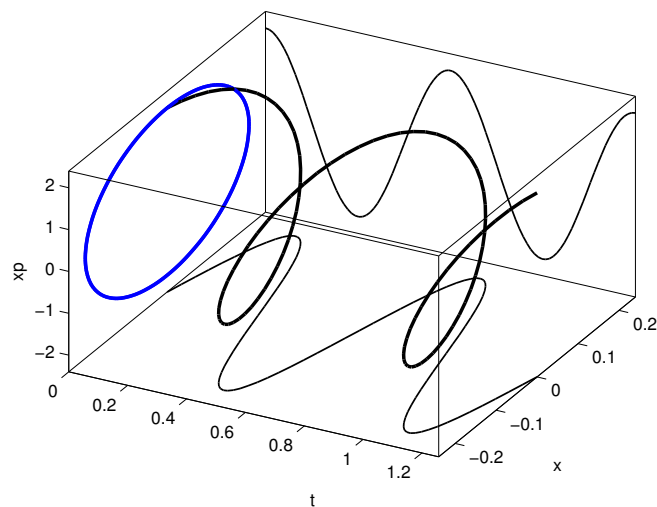


```

D=[0.025, 0.050, 0.100, 0.200]; % cac he so can Lehr D khac nhau
ii=1;
t=linspace(0, 4*pi/Omega, 100);
A=q/om_0^2/sqrt((1.0-eta_0^2)^2+4*D(ii)^2*eta_0^2);
% bien do
Psi=atan2(2.0*D(ii)*eta_0, 1-eta_0^2);
% goc pha
x=A*cos(Omega*t-Psi); % ham x(t)
xp=-Omega*A*sin(Omega*t-Psi); % ham xp(t) hay x_dot(t)
figure('name','3D')
plot3(t, x, xp, 'k','Linewidth',2), hold on
% chuyen dong khong gian
plot3(t, x, 1.2*min(xp)*ones(size(t)), 'k','Linewidth',1)
% hinh chieu trong mat phang (t, xp)
plot3(t, 1.2*max(x)*ones(size(t)), xp, 'k','Linewidth',1)
% hinh chieu trong mat phang (t, x)
plot3(t*0, x,xp, 'b','Linewidth',2)
% qui dao pha (x, xp)
axis([t(1) t(end), 1.2*[min(x) max(x)], 1.2*[min(xp) max(xp)]]);
box on
xlabel('t'); ylabel('x'); zlabel('xp');
view(28,42);

```

Chạy chương trình sẽ cho ta đồ thị như hình 3-46.



Hình 3-46. Đồ thị dao động trong không gian chuyển động, cùng các hình chiếu

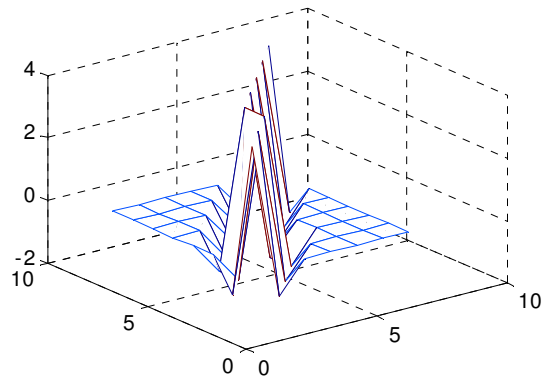
Với lệnh *mesh* ta có thể biểu diễn đồ thị 3D một ma trận, trong đó các chỉ số của phần tử thể hiện trong một mặt phẳng còn giá trị của các phần tử biểu diễn bằng độ cao tương ứng.

```
>> mesh(M)
```

Chẳng hạn ta có thể biểu diễn đồ họa 3D ma trận sau:

```
>>C=[4 -1 0 0 0 0 0;  
-1 4 -1 0 0 0 0;  
0 -1 4 -1 0 0 0;  
0 -1 4 -1 0 0 0;  
0 0 -1 4 -1 0 0;  
0 0 0 -1 4 -1 0;  
0 0 0 0 -1 4 -1;  
0 0 0 0 0 -1 4];
```

```
>> mesh(C)
```



Hình 3-47. Hiển thị một ma trận

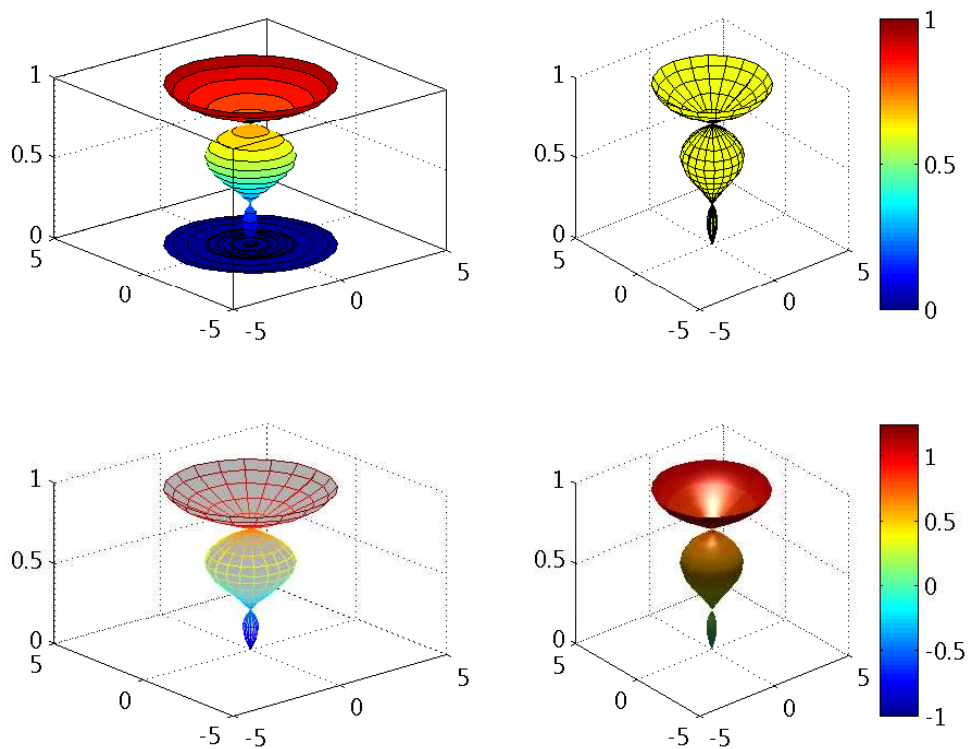
Ví dụ sau đây sử dụng kết hợp các lệnh vẽ 3D và subplot, kết quả nhận được là hình 4-48.

```
t=0:pi/10: 2*pi;  
[X, Y, Z] = cylinder(0.5*t.*cos(t));  
  
figure(1), clf reset  
subplot(221) % do thi thu nhat  
surf(X,Y,Z,'MeshStyle','row'), hold on  
surf(X,Y,Z*0,'MeshStyle','row');  
view(-40,30);  
box on  
subplot(222) % do thi thu hai  
surf(X,Y,Z,0.6*ones(size(Z)))  
caxis([0 1]);  
view(-40,30);  
colorbar  
subplot(223) % do thi thu ba  
surf(X,Y,Z,'EdgeColor','interp','FaceColor',[0.7 0.7 0.7])  
caxis([0 1]);  
view(-40,30);
```

```

subplot(224)                                % do thi thu tu
surf(X,Y,Z,'FaceLighting','phong',...
      'EdgeColor','none','AmbientStrength',0.25)
caxis([-1 1.25]);
view(-40,30);
colorbar
light('Position', [-2, 2, 10]) % Light position x, y, z
lighting phong

```



Hình 3-48. Hiển thị bằng surf với thanh màu colorbar

Một số ví dụ khác về đồ họa trong không gian 3 chiều:

Biểu diễn dao động cường độ hệ một bậc tự do có cản

$$x(t) = A \sin(\Omega t + \psi) + B e^{-\delta t} \sin(\omega t + \varphi)$$

$$\dot{x}(t) = A \Omega \cos(\Omega t + \psi) + B \omega e^{-\delta t} \cos(\omega t + \varphi) - B \delta e^{-\delta t} \sin(\omega t + \varphi)$$

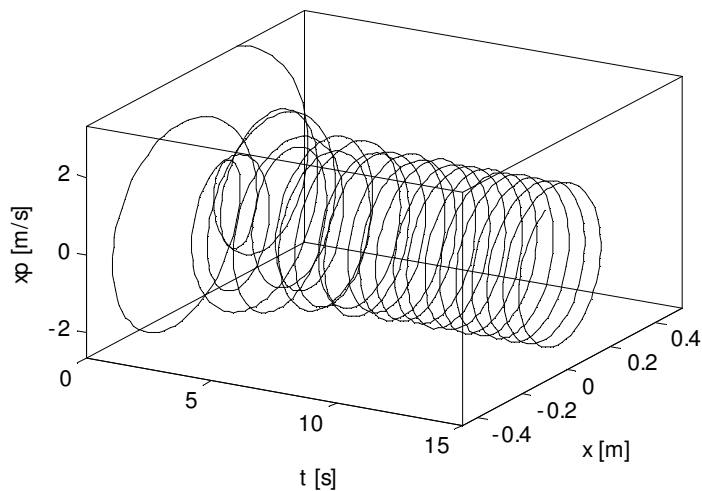
Thực hiện trong Matlab với các dòng chương trình

```

omega=4;          Omega=10;
A=0.2;           B=0.4;
delta=0.4;       psi=0.4;   phi=0.3;

t=[0:0.01:15];
x=A*sin(Omega*t+phi)+B*exp(-delta*t).*sin(omega*t+psi);
xp=A.*cos(Omega*t+phi)*Omega-B*delta.*exp(-
delta*t).*sin(omega*t+psi)+B.*exp(-
delta*t).*cos(omega*t+psi)*omega;
figure(1)
plot3(t, x, xp, '-k', 'LineWidth',1)
xlabel('t [s]'); ylabel('x [m]'); zlabel('xp [m/s]');
view(30,30);
box on
axis([0, t(end), min(x), max(x), min(xp), max(xp)])

```



Hình 3-49. Hiển thị đồ thị dao động trong không gian chuyển động

Vẽ đồ thị hàm hai biến:

$$z = f(x, y) = e^{-0.2 \cdot (x^2 + y^2)} \cdot \sin(x^2 + y^2)$$

```

x = [-3 : 0.1 : 3];
y = [-3 : 0.1 : 3];
[X, Y] = meshgrid(x, y);

f=exp(-0.2*(X.^2+Y.^2)).*sin(X.^2+Y.^2);
%hoac Z=exp(-0.2*(X.^2+Y.^2)).*sin(X.^2+Y.^2);
mxmf=max(max(f)); % tìm giá trị lớn nhất của hàm f

```

```

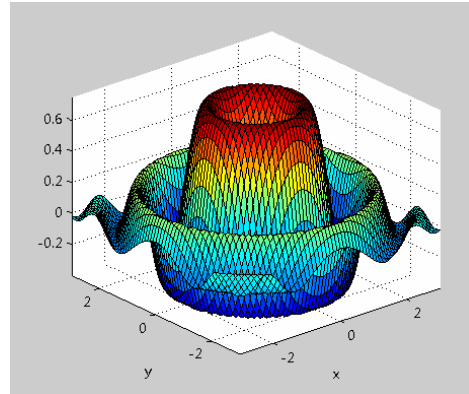
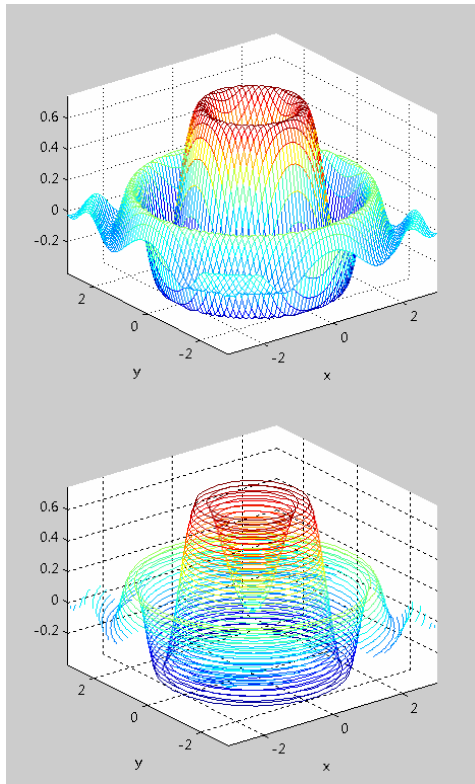
mif=min(min(f));    % tìm giá trị nhỏ nhất của hàm f

figure(1)           % sử dụng mesh
mesh(X,Y,f);
axis([-3, 3, -3, 3, mif, mxf]);
xlabel('x'); ylabel('y');

figure(2)           % sử dụng surf
surf(X,Y,f)
view(-40,30);
axis([-3, 3, -3, 3, mif, mxf]);
xlabel('x'); ylabel('y');

figure(3)           % sử dụng contour3
contour3(X,Y,f,30);
axis([-3, 3, -3, 3, mif, mxf]);
xlabel('x'); ylabel('y');

```



Hình 3-50. Hiển thị hàm hai biến trong không gian 3D, với mesh, surf và contour

4.3 Bài tập thực hành

1. Hãy vẽ đồ thị hàm $y = \tan(x)$, với $0 \leq x \leq 1$, và bước chia là 0.1. Đặt và đưa vào tên cho các trục là x và y.

2. Cho hiển thị thêm đồ thị hình $\sin(x)$ vào đồ thị vừa vẽ như là một đồ thị thứ hai. (lưu ý sử dụng hold on).

3. Tạo một vectơ hàng chứa các điểm chia trong khoảng $-\pi \leq x \leq \pi$, với bước chia 0.2. Cũng khoảng xác định trên, với lệnh linspace hãy chia thành 100 điểm và thành 50 điểm.

4. Hãy tạo các điểm chia để vẽ đồ họa không gian với các miền xác định sau

a) $-3 \leq x \leq 2$ và $-5 \leq y \leq 5$ với cùng bước chia 0.1.

b) $-5 \leq x \leq 5$ và $-5 \leq y \leq 5$ với cùng bước chia 0.2.

5. Vẽ đường cong là quỹ đạo chuyển động của một điểm trong không gian với lệnh plot3, biết

$$x = e^{-t} \cos t, \quad y = e^{-t} \sin t, \quad z = t$$

Không cần gán tên cho các trục, nhưng cho hiển thị các đường dóng.

Chương 5.

Mịn hóa đường cong

Trong kỹ thuật khi đo đạc ta nhận được các số liệu rời rạc, từ các số liệu đo này ta cần biểu diễn các đường để biết được đặc tính của hệ khảo sát và có thể suy ra được những giá trị lân cận các giá trị đã đo. Matlab đã cung cấp sẵn các hàm cho phép đưa ra được các đường biểu diễn phù hợp nhất với bộ số liệu đã có. Trong chương này ta sẽ trình bày các kỹ thuật đơn giản thực hiện công việc đó.

5.1 Mịn hóa bằng đa thức

Lệnh `polyfit(x,y,n)` cho ta một đa thức bậc n dạng

$$p_n(x) = p_1x^n + p_2x^{n-1} + \dots + p_{n-1}x^2 + p_nx^1 + p_{n+1}$$

các hệ số của đa thức này được suy ra hai vectơ số liệu \mathbf{x}, \mathbf{y}

$$\mathbf{x} = [x_1 \ x_2 \ x_3 \ \dots \ x_m], \ \mathbf{y} = [y_1 \ y_2 \ y_3 \ \dots \ y_m], \text{ với } m > n$$

nhờ phương pháp sai số bình phương bé nhất.

Trường hợp đơn giản nhất, $n = 1$, đa thức có dạng một phương trình đường thẳng

$$y = ax + b$$

Theo phương pháp sai số bình phương bé nhất hai hệ số a, b được tìm như sau. Sai số giữa giá trị của số liệu đo y_i và giá trị tính theo đường thẳng vừa đưa ra được tính theo công thức

$$e_i = (ax_i + b) - y_i, \quad i = 1, 2, \dots, m$$

Như thế tổng bình phương các sai số là

$$S(a, b) = \sum_{i=1}^m e_i^2 = \sum_{i=1}^m (ax_i + b - y_i)^2$$

Bài toán đặt ra ở đây là tìm a, b sao cho hàm S đạt cực tiểu. Đạo hàm S theo các biến a, b ta nhận được

$$\frac{\partial S}{\partial a} = \sum_{i=1}^m 2(ax_i + b - y_i)x_i = 0, \quad \frac{\partial S}{\partial b} = \sum_{i=1}^m 2(ax_i + b - y_i) = 0$$

$$\text{hay} \quad a \sum_{i=1}^m x_i x_i + b \sum_{i=1}^m x_i - \sum_{i=1}^m x_i y_i = 0, \quad a \sum_{i=1}^m x_i + mb - \sum_{i=1}^m y_i = 0$$

Từ đây giải được

$$b = \frac{\bar{y} \sum_{i=1}^m x_i x_i - \bar{x} \sum_{i=1}^m x_i y_i}{\sum_{i=1}^m x_i x_i - m\bar{x}^2}, \quad a = \frac{\sum_{i=1}^m x_i y_i - m\bar{x} \bar{y}}{\sum_{i=1}^m x_i x_i - m\bar{x}^2}$$

với các giá trị trung bình

$$\bar{x} = \frac{1}{m} \sum_{i=1}^m x_i, \quad \bar{y} = \frac{1}{m} \sum_{i=1}^m y_i$$

Hay

$$a = \frac{\sum_{i=1}^m y_i (x_i - \bar{x})}{\sum_{i=1}^m x_i (x_i - \bar{x})}, \quad b = \bar{y} - \bar{x}a.$$

Ví dụ 1. Cho biết các số liệu đo được như sau

x	6	8	10	12	14	16	18	20	22	24
y	3.94	3.8	4.1	3.87	4.45	4.33	4.12	4.43	4.6	4.5

Nếu biểu diễn các số liệu trên bằng một đường thẳng, $y = ax + b$. Hãy tìm các giá trị của a, b và tìm giá trị của y khi $x = 13$.

Trước hết ta cần nhập hai vectơ số liệu x và y vào Matlab.

```
>> x = [6: 2 : 24]
```

```
x =
```

```
6      8      10      12      14      16      18      20      22      24
```

```
>> y = [3.94 3.8 4.1 3.87 4.45 4.33 4.12 4.43 4.6 4.5]
```

```
y =
```

```
3.94    3.80    4.10    3.87    4.45    4.33    4.12    4.43    4.60    4.50
```

Tiếp theo sử dụng lệnh polyfit để tìm các hệ số của đa thức

```
>> p = polyfit(x,y,1)
```

```
p =    0.0392    3.6267
```


Theo cách biểu diễn đa thức bằng một vectơ bắt đầu từ hệ số ứng với số hạng có bậc lũy thừa lớn nhất, ta có

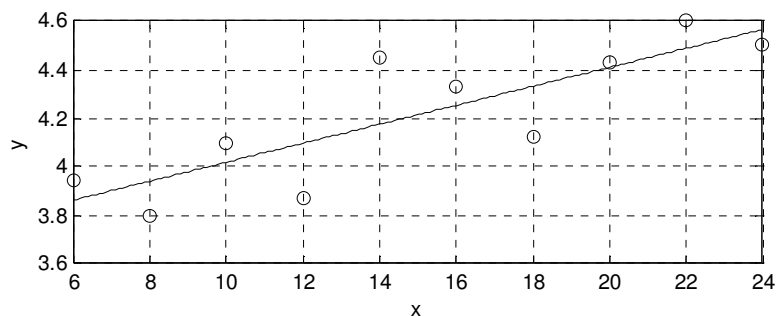
```
>> a=p(1)
a = 0.0392
>> b=p(2)
b = 3.6267
```

Giá trị cần tìm tại $x = 11$ được tính bằng lệnh polyval(p,x)

```
>> polyval(p,11)
ans = 4.0574
```

Để thấy được rõ đường thẳng cùng với các số liệu đã cho, ta vẽ chúng trên cùng một đồ thị như trên hình 5-1 bằng các dòng lệnh:

```
>> x1=[6:0.1:24];
>> y1=a*x1+b;
>> plot(x,y,'ko', x1,y1, 'k-'), grid on,
>> xlabel('x'), ylabel('y')
```



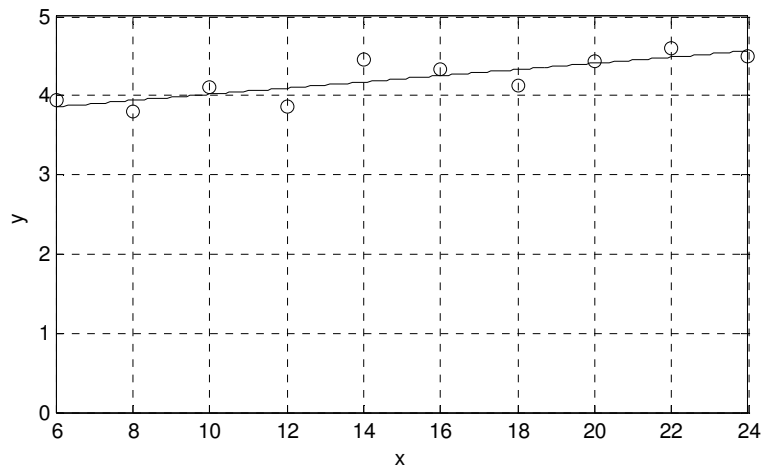
Hình 5-1a. Đường thẳng cùng với các điểm dữ liệu

Nếu tính theo đường thẳng vừa tìm thì giá trị tại các điểm x_i cho trong bảng sẽ là

```
>> w = a*x+b
w = 3.8616 3.9399 4.0182 4.0965 4.1748 4.2532 4.3315 ...
    4.4098 4.4881 4.5664
```

Khi chỉnh lại các trục tọa độ bằng lệnh axis, ta thấy các điểm dữ liệu nằm rất gần và phân bố hai bên đường thẳng $y = ax + b$ vừa tìm được.

```
>> axis([6 24 0 5])
```



Hình 5-1b. Đường thẳng cùng với các điểm dữ liệu

Xét tổng bình phương của sai số là

```
>> w = a*x+b;
>> e = w-y;
>> S1 = sum(e.*e)
S1 = 0.2274
```

Nếu các điểm trên được làm mịn bằng đường bậc 2:

```
>> n=2;
>> p = polyfit(x,y,n);
p =
    -0.0002    0.0443    3.5940
>> a=p(1); b=p(2); c = p(3);
>> w = a*x.^2+b*x+c;
>> e = w-y;
>> S2= sum(e.*e)
S2 = 0.2272
```

Ta thấy sai số có nhỏ hơn ($S2 < S1$) nhưng không đáng kể.

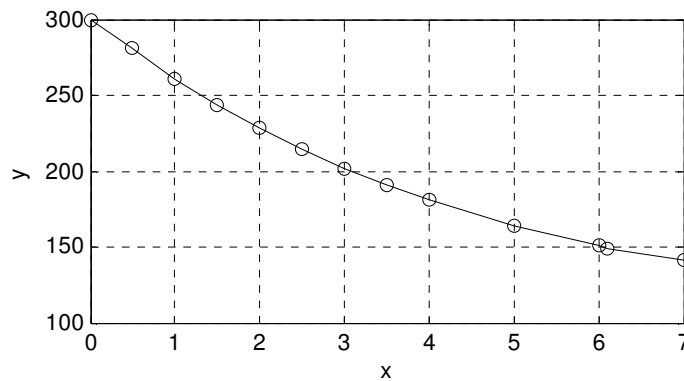
Ví dụ 2. Cho bộ số liệu sau

x	0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	5.0	6.0	6.1	7.0
y	300	281	261	244	228	214	202	191	181	164	151	149	141

Hãy tìm một đa thức bậc 3 phù hợp với số liệu cho. So sánh sai số so với trường hợp nếu sử dụng đa thức bậc hai.

Trước hết nhập số liệu vào hai véc-tơ và nối các điểm ta được

```
>> x = [0, 0.5, 1.0, 1.5, 2, 2.5, 3, 3.5, 4, 5, 6, 6.1, 7];
>> y = [300, 281, 261, 244, 228, 214, 202, 191, 181, 164, 151, 149, 141];
>> plot(x,y, 'ko', x,y, 'k-'), grid on, xlabel('x'), ylabel('y')
```



Hình 5-2. Đường nối các điểm dữ liệu

Từ hình 5-2 ta thấy các điểm phân bố không theo đường thẳng, nên ta sẽ thử với các đa thức bậc 2 và 3.

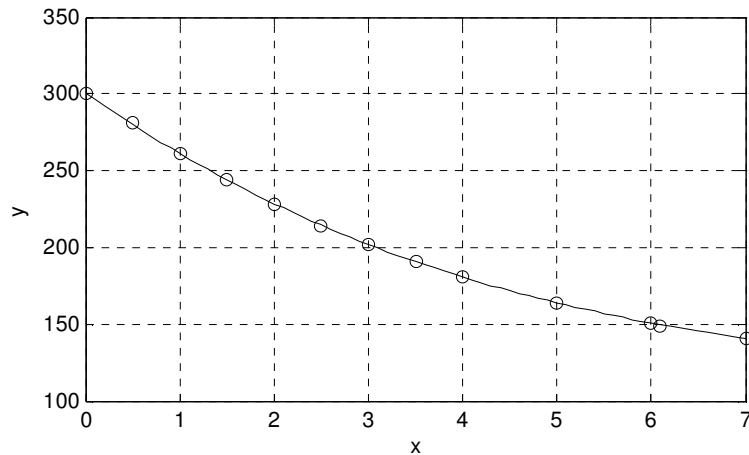
```
>> n=2; % Xấp xỉ bằng đa thức bậc hai
>> p = polyfit(x,y,n)
>> a=p(1); b=p(2); c = p(3);
>> w = a*x.^2+b*x+c;
>> e = w-y;
>> S2= sum(e.*e)
S2 = 15.4079

>> n=3; % Xấp xỉ bằng đa thức bậc ba
>> p = polyfit(x,y,n)
>> a=p(1); b=p(2); c = p(3); d = p(4);
>> w = a*x.^3+b*x.^2+c*x+d;
>> e = w-y;
>> S3= sum(e.*e)
S3 = 2.7933
```

So sánh S2 và S3 ta thấy đường cong bậc 3 sẽ phù hợp hơn đường cong bậc 2.

```
>> p = polyfit(x,y,3)
p =
    -0.1290    3.8030   -43.1278   300.5634
>> a = p(1); b = p(2); c = p(3); d = p(4);
>> x3 = [0:0.1:7];
>> y3 = a*x3.^3+b*x3.^2+c*x3+d;
```

```
>> plot(x,y,'ko', x3,y3, 'k-'), grid on, xlabel('x'),ylabel('y')
```



Hình 5-3. Đường bậc 3 và các điểm dữ liệu cho

Ta thấy các điểm dữ liệu cho nằm ngay trên hoặc rất gần đường cong bậc ba, như vậy việc chọn đa thức bậc 3 trong trường hợp này là hợp lý.

5.2 Mịn hóa bằng hàm e mũ

Trong nhiều trường hợp khi mịn hóa bằng đa thức không đáp ứng được yêu cầu đặt ra, ta cần phải tìm một phương án khác. Biểu diễn các số liệu thông qua hàm e mũ là một phương án nên được xem xét. Giả sử bộ số liệu (x,y) có thể lấy hàm

$$y = be^{ax}$$

là một xấp xỉ.

Bây giờ ta cần tìm hai hệ số a và b để các điểm dữ liệu nằm gần đường cong nhất. Nếu áp dụng trực tiếp phương pháp sai số bình phương bé nhất ta sẽ được một hệ phương trình phi tuyến đối với các ẩn a và b . Để tránh điều đó, ta lấy lôgarit cơ số tự nhiên hai vế được

$$\ln y = ax + \ln b$$

Đặt $w = \ln y$, $z = x$, và $p_1 = a$, $p_2 = \ln b$ bài toán trở thành tìm các hệ số p_1, p_2 xấp xỉ các số liệu $(z, w) \equiv (x, \ln y)$ bằng đường thẳng

$$w = p_1 z + p_2$$

Như vậy ta có thể sử dụng lệnh `polyfit(z,w,1)` để tìm các hệ số p_1, p_2 . Hay ta có thể sử dụng trực tiếp lệnh

```
>> p = polyfit(x, log(y), 1) % trong Matlab log(x) = ln(x)
```

Từ đó xác định được các hệ số a và b

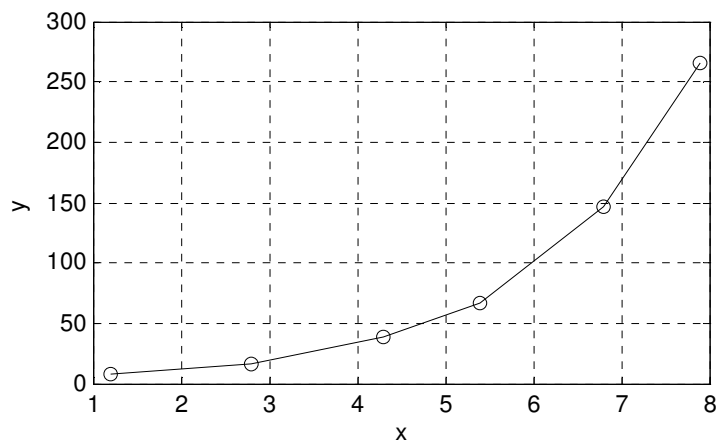
$$a = p_1, \quad b = \exp(p_2)$$

Ví dụ xét bộ số liệu sau

x	1.2	2.8	4.3	5.4	6.8	7.9
y	7.5	16.1	38.9	67.0	146.6	266.2

Nhập số liệu vào Matlab

```
>> x = [1.2  2.8  4.3  5.4  6.8  7.9];
>> y = [7.5  16.1  38.9  67.0  146.6  266.2];
>> plot(x,y,'ko', x,y, 'k-'), grid on, xlabel('x'),ylabel('y')
```



Hình 5-4. Đường nối các điểm dữ liệu

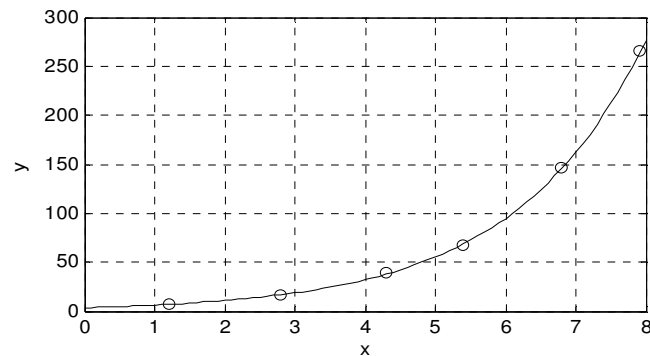
Xấp xỉ bằng hàm e mũ

```
>> p = polyfit(x,log(y),1)
p =
    0.5366    1.3321
>> a = p(1); b=exp(p(2));
>> w = b*exp(a*x);
>> err = w-y;
>> S3= sum(err.*err)
S3 = 17.6259
>> x3 = [0:0.1:8];
>> y3 = b*exp(a*x3);
>> plot(x,y,'ko', x3,y3, 'k-'), grid on,xlabel('x'),ylabel('y')
```

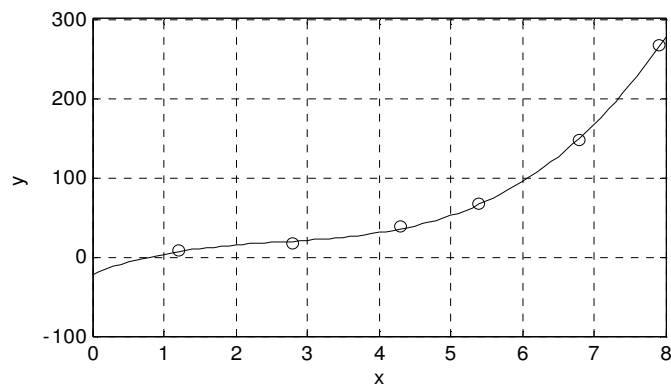
Sau đây ta thử xấp xỉ bằng đa thức bậc 3:

```
>> p = polyfit(x,y,3) % Xap xi bang duong bac ba
p = 1.4497 -11.2896 34.9691 -21.6503
```

```
>> a = p(1); b=p(2); c = p(3); d = p(4);
>> w = a*x.^3+b*x.^2+c*x+d;
>> err = w-y;
>> S3= sum(err.*err)
S3 = 40.1435
>> x3 = [0:0.1:8];
>> y3 = a*x3.^3+b*x3.^2+c*x3+d;
>> plot(x,y,'ko', x3,y3, 'k-'), grid on, xlabel('x'),ylabel('y')
```



Hình 5-5. Đường xấp xỉ e mũ và các điểm dữ liệu



Hình 5-6. Đường xấp xỉ bậc 3 và các điểm dữ liệu cho

So sánh hai trường hợp trên ta thấy, xấp xỉ đường cong đối với các số liệu đã cho bằng hàm e mũ hợp lý hơn khi sử dụng đa thức bậc 3.

5.3 Nội suy đa thức

Lệnh `polyfit(x,y,n)` cho ta một đa thức bậc n mà đồ thị của nó là đường gần với đường tạo ra khi nối các điểm cho bởi bộ số liệu, còn nội suy đa thức cũng cho ta

một đường cong nhưng đi qua các điểm số liệu đó. Có nhiều phương pháp để tạo ra đa thức đi qua các điểm cho như phương pháp nội suy theo công thức Lagrange, phương pháp Newton, phương pháp Neville, phương pháp nội suy với đường spline bậc 3, Chi tiết về các phương pháp này có thể xem trong các tài liệu về giải tích số và phương pháp số.

Trong phần này chỉ trình bày việc sử dụng các lệnh *interp* của Matlab để thực hiện tìm các giá trị nội suy từ các số liệu sẵn có. Bài toán đặt ra là cho biết các giá trị tương ứng của hai đại lượng x và y như trong bảng số liệu

x	x_1	x_2	x_3	...	x_n
y	y_1	y_2	y_3	...	y_n

Hãy tìm giá trị của biến y^* ứng với giá trị nào đó của biến x^* với $x_1 \leq x^* \leq x_n$. Bài toán này trong Matlab được giải quyết dễ dàng với lệnh *interp*. Cách gọi lệnh này như sau:

```
y_star = interp1(x,y,x_star)
```

Nếu x là véc-tơ gồm các phần tử nguyên từ 1 đến n ($x = 1:n$), với $n = \text{length}(y)$, thì ta có thể gọi lệnh

```
y_star = interp1(y,x_star)
```

Để nói rõ hơn phương pháp nào được sử dụng trong việc nội suy này ta sử dụng cú pháp

```
y_star = interp1(x, y, x_star, method)
```

với *method* được chọn là một trong các phương pháp sau

- 'nearest' - nội suy theo lân cận gần nhất
- 'linear' - nội suy tuyến tính
- 'spline' - nội suy sử dụng đường spline bậc 3
- 'pchip' - nội suy bậc 3 từng đoạn
- 'cubic' - tương tự như 'pchip'

Nếu giá trị x^* nằm ngoài khoảng $[x_1 \dots x_n]$, khi đó ta gọi phép tính giá trị $y^* = y(x^*)$ là phép tính ngoại suy và sử dụng lệnh với cú pháp:

```
y_star = interp1(x, y, x_star, method, 'extrap')
```

Sau đây xét ví dụ:

```
>> x = 1:20; y = sin(x);
>> x_star = 1.5;
>> y_star = interp1(x,y, x_star)
```

```

y_star =
    0.8754
>> y_star = interp1(x,y, x_star, 'linear')
y_star =
    0.8754
>> y_star = interp1(x,y, x_star, 'cubic')
y_star =
    0.9008
>> y_star = interp1(x,y, x_star, 'nearest')
y_star =
    0.9093
>> y_star = interp1(x,y, x_star, 'spline')
y_star =
    1.0200

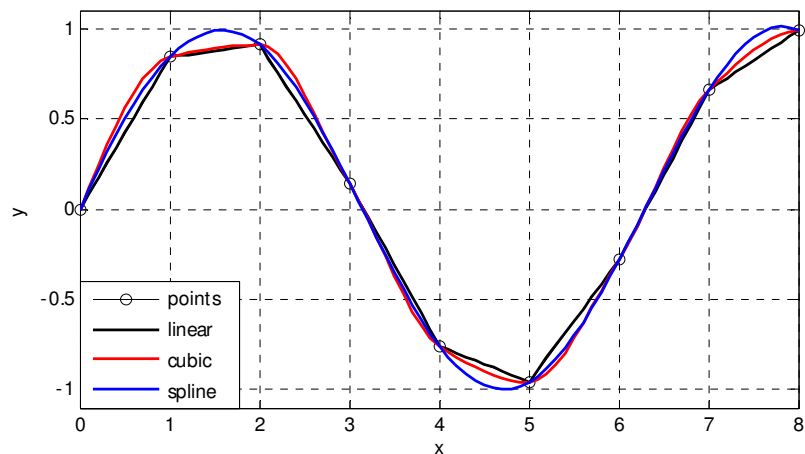
```

Như thế các phương pháp nội suy khác nhau sẽ cho ta các kết quả khác nhau. Để thấy rõ hơn kết quả của các phương pháp nội suy, ta xét ví dụ sau

```

x = 0:8; y = sin(x); plot(x,y,'-ko'), hold on
xi = 0:0.1:8; yi1 = interp1(x,y, xi, 'linear');
yi2 = interp1(x,y, xi, 'cubic');
yi3 = interp1(x,y, xi, 'spline');
plot(xi,yi1,'-k','linewidth',1.5)
plot(xi,yi2,'-r','linewidth',1.5)
plot(xi,yi3,'-b','linewidth',1.5), grid on
xlabel('x'), ylabel('y'), axis([0 8, -1.1 1.1])
legend('points','linear','cubic','spline')

```



Hình 5-7. Kết quả nội suy bằng đường bậc 3 và đường spline

Dưới đây là một ví dụ khác.

```
>> x = [1.2; 2.8; 4.3; 5.4; 6.8; 7.0];
>> y = [7.5; 16.1; 38.9; 67.0; 146.6; 266.2];
>> x_star = 100;
>> y_star = interp1(x,y, x_star, 'spline')
y_star =
    15.1333
>> y_star = interp1(x,y, x(1), 'spline')
y_star =
     7.5000
>> y_star = interp1(x,y, x(3), 'spline')
y_star =
    38.9000
```

5.4 Bài tập thực hành

1. Để xác định môđun đàn hồi của nhôm, người ta thực hiện ba thử nghiệm trên một thanh nhôm và thu được kết quả như trong bảng số liệu sau

	Ứng suất (MPa)	34.5	69.0	103.5	138.0
biến dạng (mm/m)	Lần đo 1	0.46	0.95	1.48	1.93
	Lần đo 2	0.34	1.02	1.51	2.09
	Lần đo 3	0.73	1.10	1.62	2.12

Xác định quan hệ tuyến tính giữa ứng suất và biến dạng trung bình của ba lần đo, từ đó đưa ra môđun đàn hồi E của nhôm, theo công thức $\sigma = \varepsilon E \Rightarrow E = \sigma / \varepsilon$.

2. Tìm phương trình đường thẳng $y = ax + b$ thể hiện quan hệ tuyến tính của bộ số liệu cho trong bảng.

x	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
y	3.076	2.810	2.588	2.297	1.981	1.912	1.653	1.478	1.399	1.018	0.794

3. Bảng số liệu dưới đây cho biết khối lượng m (kg) và suất tiêu hao nhiên liệu φ (km/lít) của một số xe ô tô sản xuất bởi Ford và Honda năm 1999. Tìm phương trình đường thẳng $\varphi = a + bm$ thể hiện quan hệ tuyến tính giữa khối lượng và suất tiêu hao nhiên liệu.

Model	M(kg)	φ (km/lít)
Contour	1310	10.2
Crown Victoria	1810	8.1
Escort	1175	11.9
Expedition	2360	5.5
Explorer	1960	6.8
F-150	2020	6.8

Ranger	1755	7.7
Taurus	1595	8.9
Accord	1470	9.8
CR-V	1430	10.2
Civic	1110	13.2
Passport	1785	7.7

4. Tỷ trọng tương đối của không khí thay đổi theo chiều cao đo được như trong bảng.

h (km)	0	1.525	3.050	4.575	6.10	7.625	9.150
ρ	1	0.8617	0.7385	0.6292	0.5328	0.4481	0.3741

Tìm phương trình bậc hai $\rho = a + bh + ch^2$ thể hiện quan hệ giữa độ cao và tỷ trọng không khí, từ đó tính tỷ trọng của không khí ở độ cao $h = 10.5$ km.

5. Độ nhớt động học μ_k của nước thay đổi theo nhiệt độ T được cho như trong bảng dưới đây

T (°C)	0	21.1	37.8	54.4	71.1	87.8	100
μ_k (10^{-3} m ² /s)	1.79	1.13	0.696	0.519	0.338	0.321	0.296

Xác định đường bậc ba thể hiện quan hệ này, từ đó tính độ nhớt động học tại các nhiệt độ $T = 10^\circ, 30^\circ, 60^\circ$, và 90° °C

6. Xác định a và b để hàm $f(x) = ax^b$ phù hợp với bộ số liệu theo nghĩa sai số bình phương nhỏ nhất.

x	0.5	1.0	1.5	2.0	2.5
y	0.49	1.60	3.36	6.44	10.16

Phương trình vi phân thường và Matlab

Phương trình vi phân mô tả các quá trình động lực là phương trình chứa các biến trạng thái, đạo hàm của nó phụ thuộc vào biến độc lập. Trong chương này chúng ta sẽ học làm thế nào để sử dụng Matlab tìm nghiệm số của phương trình vi phân với các điều kiện đầu. Trong Matlab có nhiều công cụ, hàm giải phương trình với các phương pháp khác nhau.

6.1 Giải phương trình vi phân bậc nhất với ode23 và ode45

Để giải số phương trình vi phân thường, trước hết ta cần định nghĩa phương trình cần phải giải. Để làm ví dụ, ta xét phương trình vi phân sau

$$\frac{dy}{dt} = \cos(t)$$

Với phương trình đơn giản này ta có thể viết ra được nghiệm chính xác (nghiệm giải tích) $y(t) = \sin(t) + C$, với C là hằng số tích phân phụ thuộc vào điều kiện đầu. Nghiệm này sẽ được sử dụng để so sánh với nghiệm tìm bằng phương pháp số. Để định nghĩa phương trình vi phân ta tạo một m-file với nội dung

```
function ydot = eq1(t,y)
ydot = cos(t);
```

Hàm này sẽ tự động được lấy tên là eq1.m khi ta ghi lại.

Để giải phương trình vi phân, ta gọi hàm ode23, đó là chương trình giải số bằng phương pháp tích phân sử dụng công thức Runge-Kutta cấp hai và ba. Cú pháp của câu lệnh như sau

```
[t, y] = ode23('func_name', [t_start, t_end], y(0))
```

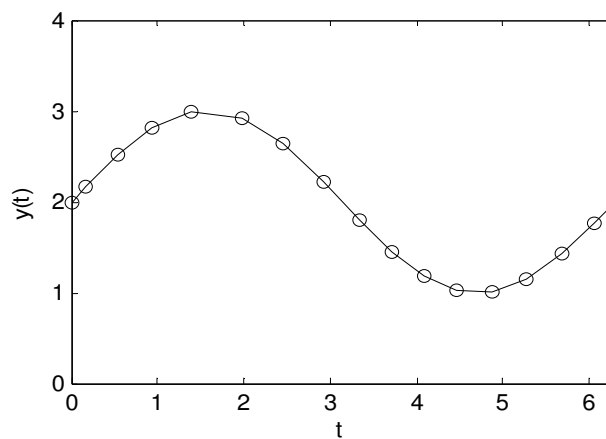
Hàm của chúng ta có tên eq1, với khoảng thời gian tích phân $0 \leq t \leq 2\pi$, và giả sử rằng có điều kiện đầu $y(0) = 2$. Từ cú pháp trên ta viết

```
>> [t, y] = ode23('eq1', [0, 2*pi], 2)
```

Đây là phương trình đơn giản, nên Matlab nhanh chóng đưa ra kết quả. Để so sánh với nghiệm giải tích, ta vẽ chúng trên một đồ thị. Với nghiệm giải tích

```
>> f = sin(t) + 2;
>> plot(t,y,'ko', t, f,'k-'), xlabel('t'), ylabel('y(t)'),
>> axis([0 2*pi 0 4])
```

Kết quả của lệnh plot trên là đồ thị như trên hình 6-1. Ta thấy rằng nghiệm số (các dấu tròn) rất gần với nghiệm giải tích, như trên hình vẽ thì khó có thể thấy được sai lệch giữa hai phương pháp. Nghiệm nhận được bằng lệnh ode23 trong trường hợp này khá chính xác.



Hình 6-1. Nghiệm số và nghiệm giải tích của ptvp $dy/dt = \cos(t)$, ode23

Để thấy được sai số tương đối giữa nghiệm số và nghiệm giải tích ta sẽ đưa ra đại lượng

$$e(t) = \left| \frac{f(t) - y(t)}{f(t)} \right|$$

Giá trị của sai số này tại các điểm thời gian, được xác định bằng các dòng lệnh

```
>> err_y = abs((f-y)./f);
>> for i = 1:1:size(y)
    err_y(i) = abs((f(i)-y(i))/f(i));
end

>> err_y =
ans =
    1.0e-003 *
         0
    0.0001
    0.0091
    0.0301
```

```

0.0743
0.2115
0.2876
0.3780
0.4659
0.5597
0.6480
0.6907
0.6050
0.4533
0.3164
0.2414
0.2129

```

Ta thấy sai số tương đối trên khá nhỏ, sai số lớn nhất tìm được bằng lệnh max

```

>> eymax = max(err_y)
eymax = 6.9075e-004

```

Hàm ode45 sử dụng phương pháp Runge-Kutta bậc cao hơn, bậc 4 và bậc 5. Chúng ta sẽ sử dụng lệnh này để giải phương trình vi phân trên, bằng cách viết tương tự

```

>> [t, w] = ode45('eq1', [0, 2*pi], 2)

```

So sánh với nghiệm giải tích, ở đây ta cần tính lại

```

>> f = sin(t) + 2;
>> plot(t,w,'ko', t, f,'k-'), xlabel('t'), ylabel('y(t)'),
>> axis([0 2*pi 0 4])

```

Với lý do có thể là số điểm chia sử dụng trong ode45 khác với số điểm chia sử dụng trong ode23. Hình 6-2 là đồ thị các nghiệm $f(t)$ và $w(t)$. Để so sánh với phương pháp giải tích, ta tính sai số tương đối với các dòng lệnh:

```

>> err_w = abs((f-w)./f); >> err_w = zeros(size(w));
>> for i = 1:1:size(w)
    err_w(i) = abs((f(i)-w(i))/f(i));
end

```

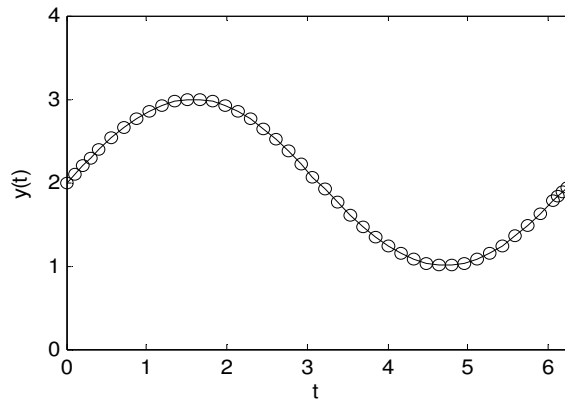
và tìm được

```

>> ewmax = max(err_w)
ewmax = 4.9182e-006
>> ss = eymaxx/ewmax
ss = 140.4473

```

Sai số này nhỏ hơn cỡ 140 lần so với khi sử dụng lệnh ode23.



Hình 6-2. Nghiệm số và nghiệm giải tích của ptvp $dy/dt = \cos(t)$, ode45

Lưu ý rằng số điểm chia sử dụng trong ode23 nhỏ hơn số điểm chia khi sử dụng ode45. Hãy so sánh length(y) và length(w).

Trong ví dụ sau ta sẽ sử dụng ode45 để giải phương trình vi phân

$$\frac{dy}{dt} = -5y + 5F(t), \text{ với kích động } F(t) = te^{-t/t_c} \cos(\omega t),$$

với các số liệu cho trong bảng

Hằng số thời gian, t_c	Tần số ω
0.01 s	628 rad/s
0.1 s	6.28 rad/s

Trước hết định nghĩa phương trình vi phân trong một m-file, trong đó các thông số t_c và ω được khai báo ở dạng biến toàn cục (global tc w):

```
function ydot = eq2(t,y)
global tc w
F = t*exp(-t/tc)*cos(w*t);
ydot = 5*(F - y);
% save with file name eq2.m
```

Khi ghi lại phần soạn thảo trên Matlab tự động lấy tên hàm eq2 đặt tên cho m-file.

Trong cửa sổ lệnh, ta khai báo tc và w là các biến toàn cục

```
>> global tc w
```

Trước hết sử dụng các thông số trong dòng đầu của bảng:

```
>> tc = 0.01; w = 628;
```

Đưa vào thời gian cuối `final_time`, cùng với điều kiện đầu $y(0)$:

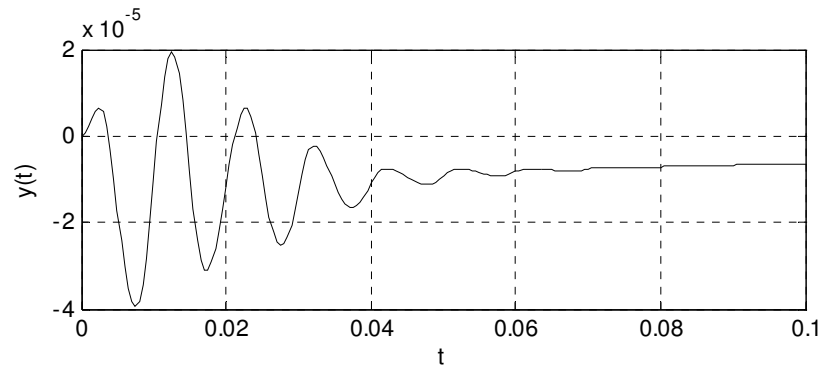
```
>> final_time = 0.1; y0 = 0;
```

Sau đó gọi lệnh `ode45` để giải theo mẫu cú pháp:

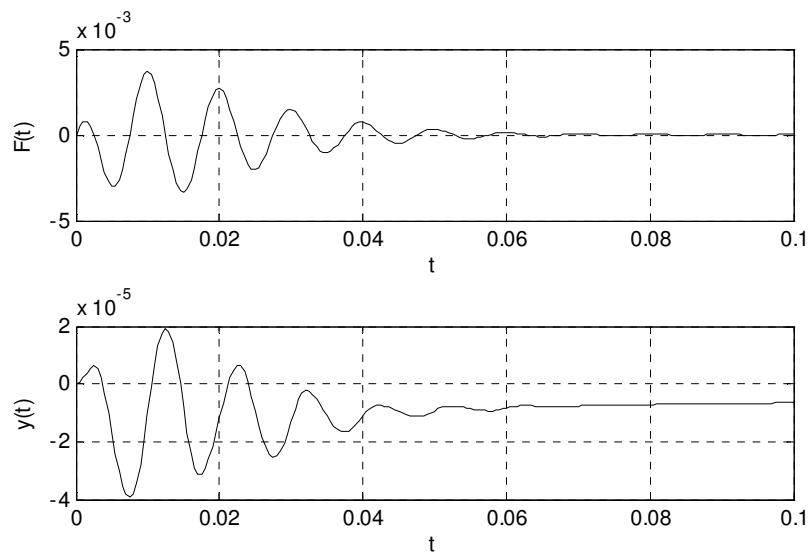
```
>> [t,y] = ode45('eq2',[0:0.0005:final_time],y0);
```

```
>> plot(t,y,'k-'), xlabel('t'), ylabel('y(t)'), grid on
```

Kết quả được thể hiện như trên hình 6-3.



Hình 6-3. Nghiệm số với $t_c = 0.01$ và $w = 628$



Hình 6-4. So sánh kích động $F(t)$ và đáp ứng $y(t)$

Để so sánh với hàm kích động $F(t)$, ta viết vào dòng lệnh:

```
>> F = t.*exp(-t./tc).*cos(w*t);
```

```
>> subplot(2,1,1); plot(t,F,'k-'), xlabel('t'), ylabel('F(t)'),
```

```
>> grid on
>> subplot(2,1,2); plot(t,y,'k-'), xlabel('t'), ylabel('y(t)'),
>> grid on
```

Và nhận được đồ thị như trên hình 6-4.

Từ hình trên ta thấy lực kích động lớn hơn đến cả 100 lần đáp ứng, nhưng chúng có dạng khá giống nhau. Cả hai đều có dạng dao động tắt dần.

Bây giờ ta thực hiện giải phương trình vi phân đã cho với các số liệu:

```
>> tc = 0.1; w = 6.28;
```

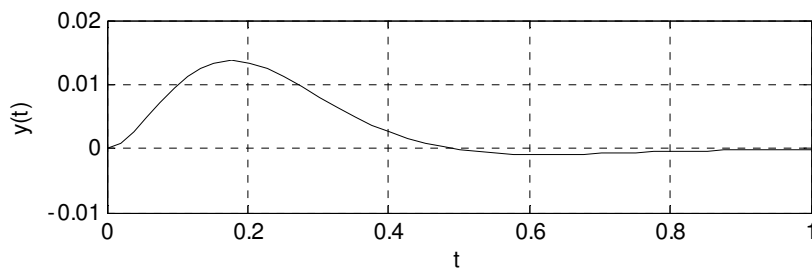
Bởi vì hằng số thời gian tăng 10 lần nên, ta cho thời gian tích phân cũng tăng lên 10 lần:

```
>> final_time = 1.0;
```

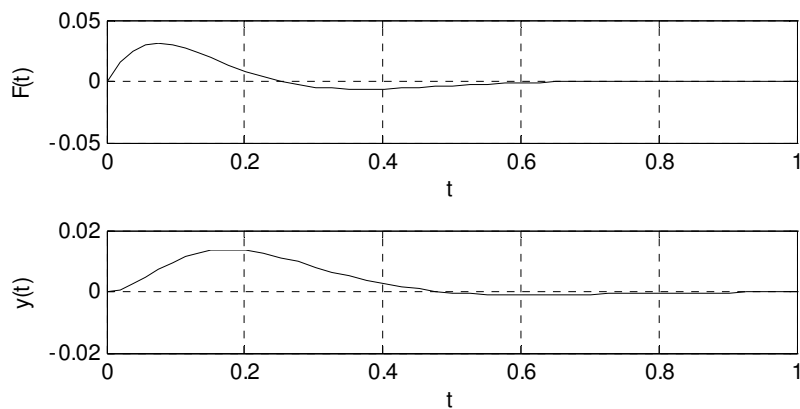
Gọi lại lệnh

```
>> [t,y] = ode45('eq2',[0 final_time],y0);
>> plot(t,y,'k-'), xlabel('t'), ylabel('y(t)'), grid on
```

Kết quả trong trường hợp này được thể hiện trên hình 6-5 và 4-6.



Hình 6-5. Nghiệm số với $tc = 0.1$ và $w = 6.28$



Hình 6-6. So sánh kích động $F(t)$ và đáp ứng $y(t)$ với $tc = 0.1$ và $w = 6.28$

6.2 Giải các phương trình vi phân bậc hai

Trong phần này sẽ trình bày việc sử dụng các lệnh ode23 và ode45 của Matlab để giải các phương trình vi phân bậc cao hay hệ phương trình vi phân bậc một. Đối với một phương trình vi phân bậc cao, chẳng hạn cấp hai

$$\frac{d^2 y}{dt^2} = f\left(t, y, \frac{dy}{dt}\right)$$

ta luôn có thể được biến đổi thành hệ hai phương trình vi phân cấp 1. Bằng cách đặt

$$y_1 = y, \quad y_2 = \frac{dy}{dt} \equiv \frac{dy_1}{dt}$$

ta nhận được các phương trình vi phân bậc nhất tương đương là

$$\frac{dy_1}{dt} = y_2, \quad \frac{dy_2}{dt} = f(t, y_1, y_2)$$

Nghiệm của phương trình này cần hai điều kiện đầu $y_1(t_o) = y_{1o}$, $y_2(t_o) = y_{2o}$.

Trước hết xét việc giải hệ hai phương trình vi phân cấp một sau

$$\frac{dx}{dt} = -x^2 + y, \quad \frac{dy}{dt} = -x - xy$$

với các điều kiện đầu $x(0) = 0$, $y(0) = 1$. Vẽ đồ thị các kết quả nhận được $x(t)$, $y(t)$ và đồ thị quỹ đạo pha (x, y) .

Để giải hệ trên bằng ode45, ta viết một m-file thể hiện vế phải của hệ trên, ở đây để thuận tiện cho việc biểu diễn vectơ, ta đặt

$$x_1 = x, \quad x_2 = y \Rightarrow \frac{dx_1}{dt} = -x_1^2 + x_2, \quad \frac{dx_2}{dt} = -x_1 - x_1 x_2$$

và m-file được viết như sau:

```
function xdot = eqx(t,x);  
xdot = zeros(2,1);  
xdot(1) = -x(1)^2 + x(2);  
xdot(2) = -x(1) - x(1)*x(2);  
% save with file name eqx.m
```

Dòng lệnh Matlab sau sẽ giải hệ trong khoảng thời gian từ 0 đến 10 s:

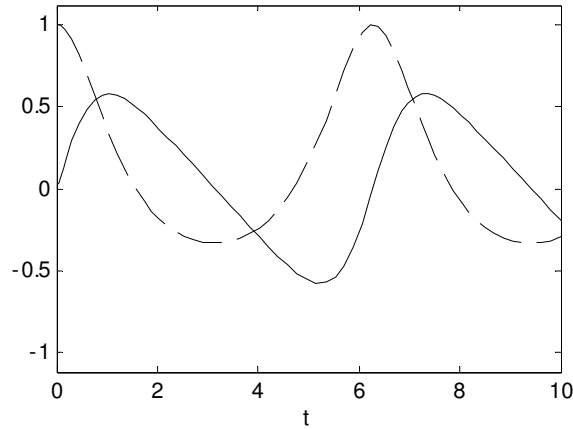
```
>> [t,x] = ode45('eqx',[0 10],[0,1]);
```

Nghiệm cần tìm x được ghi lại bằng hai vectơ cột, cột thứ nhất $x_1 = x(:,1)$ và cột thứ hai $x_2 = x(:,2)$. Với lệnh vẽ

```
>> plot(t,x(:,1),t,x(:,2),'--'),xlabel('t'),
```

```
>> axis([0 10 -1.12 1.12])
```

Ta nhận được đồ thị hai hàm x_1 và x_2 theo thời gian như trên hình 6-6.

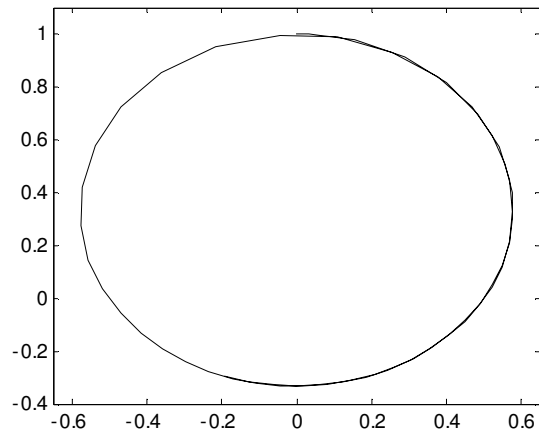


Hình 6-7. Hàm x tức x_1 nét liền, và hàm y tức x_2 nét đứt

Để vẽ quỹ đạo pha, sử dụng lệnh sau

```
>> plot(x(:,1),x(:,2), 'k-'), axis([-0.65 0.65 -0.4 1.1])
```

và ta nhận được hình ảnh pha như trên hình 6-8.



Hình 6-8. Quỹ đạo pha (x_1, x_2)

Ví dụ dưới đây trình bày việc giải phương trình vi phân bậc hai. Hãy tìm nghiệm của phương trình vi phân sau

$$\ddot{y} + 16y = \sin(4.3t) \text{ với điều kiện đầu } y(0) = 0, \dot{y}(0) = 0.$$

Bằng cách đặt $x_1 = y$, $x_2 = \dot{y}$, ta nhận được hệ hai phương trình vi phân cấp một:

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= -16x_1 + \sin(4.3t)\end{aligned}$$

Vế phải của hệ trên được thể hiện trong một m-file như sau:

```
function xdot = eqx2(t,x);
xdot = zeros(2,1);
xdot(1) = x(2);
xdot(2) = sin(4.3*t)-16*x(1);
% save with file name eqx2.m
```

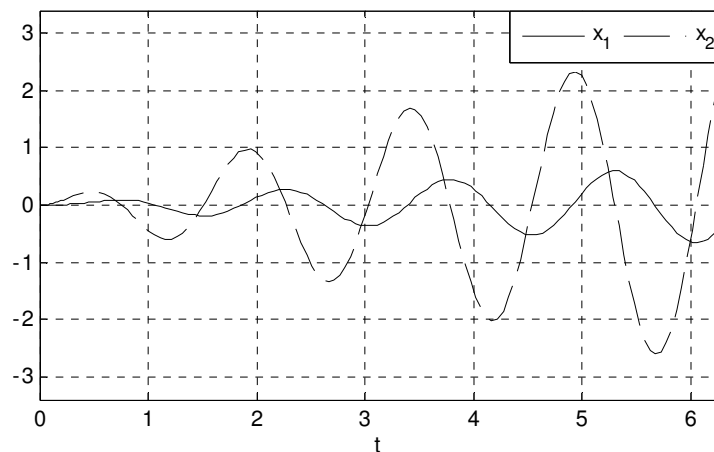
Ta gọi lệnh ode45 để nhận được nghiệm số của phương trình. Do hệ có hàm sin kích động, nên ta chọn khoảng thời gian để tính là $0 \leq t \leq 2\pi$:

```
>> [t,x] = ode45('eqx2',[0 2*pi],[0,0]);
```

Với dòng lệnh plot

```
>> plot(t,x(:,1),t,x(:,2),'--'),xlabel('t'), axis([0 2*pi -3 3])
```

ta nhận được đồ thị như trên hình 6-9.

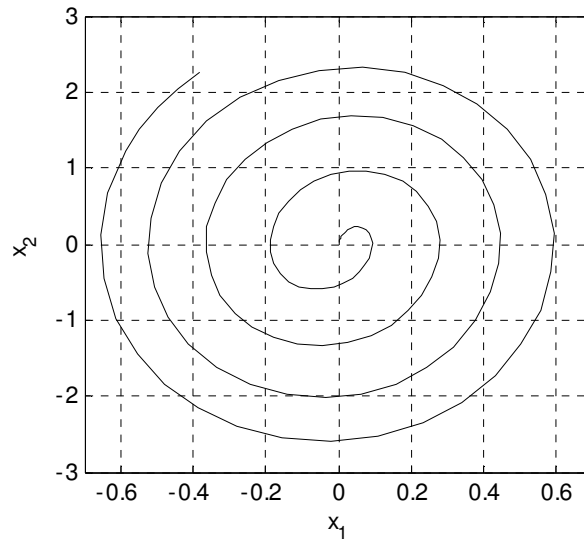


Hình 6-9. Ứng xử của hệ với điều kiện đầu $[0, 0]$

Qua đồ thị ta thấy rằng, các hàm x_1 và x_2 tăng theo thời gian, và x_2 có biên độ lớn hơn biên độ x_1 . Để tạo ra đồ thị quỹ đạo pha, ta sử dụng lệnh

```
>> plot(x(:,1), x(:,2),'k-'), xlabel('x_1'), ylabel('x_2')
>> grid on, axis([-0.7 0.7 -3 3])
```

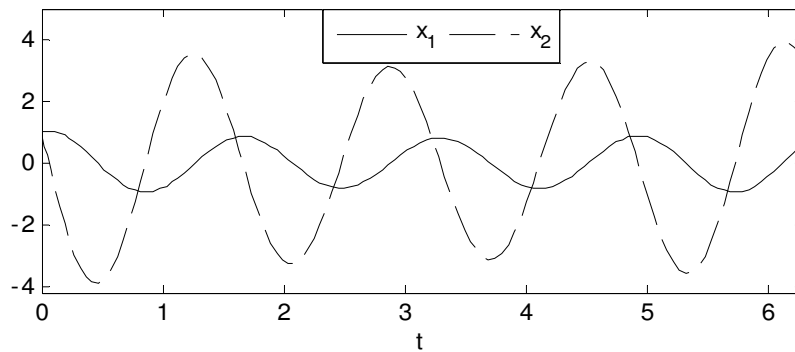
và nhận được kết quả như trên hình 6-10.



Hình 6-10. Hình ảnh quỹ đạo pha với điều kiện đầu $[0, 0]$

Để so sánh, ta sẽ giải hệ trên với điều kiện đầu $y(0) = \dot{y}(0) = 1$. Lặp lại các dòng lệnh trên, ta nhận được đồ thị như trên các hình 6-11,

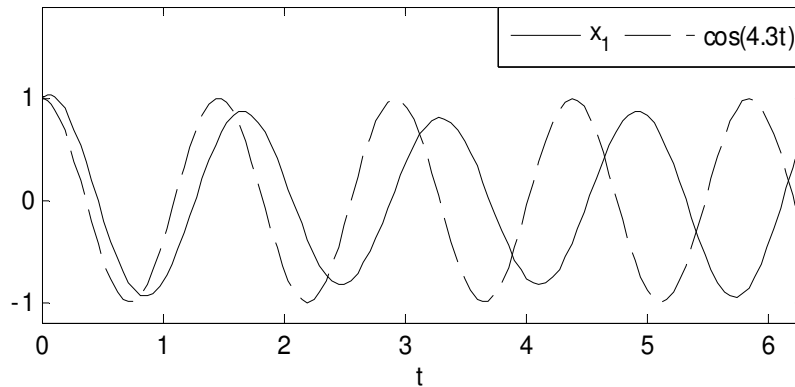
```
>> [t,x] = ode45('eqx2',[0 2*pi],[1,1]);
>> plot(t,x(:,1), 'k-',t,x(:,2),'k--'), xlabel('t'),
>> axis([0 2*pi -4.2 4.2]), legend('x_1','x_2')
```



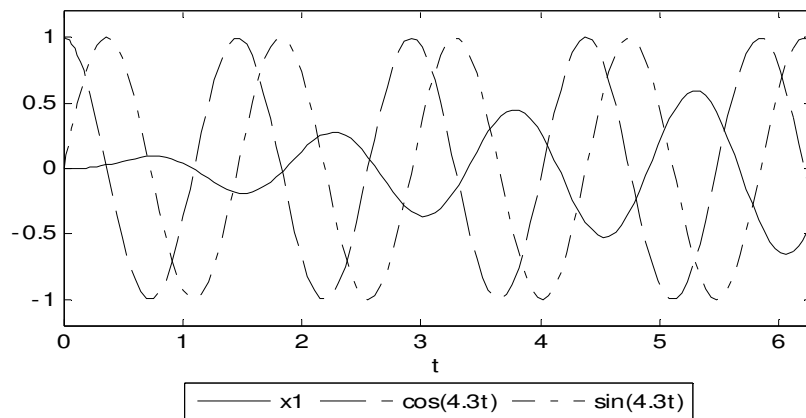
Hình 6-11. Ứng xử của hệ với điều kiện đầu $[1, 1]$,

Như thấy trên đồ thị, nghiệm của hệ đã thay đổi rất nhiều khi điều kiện đầu thay đổi từ $[0, 0]$ sang $[1, 1]$. Các nghiệm tăng sớm hơn ngay trong những giây đầu tiên. Để so sánh x_1 với $\cos(4.3t)$ ta vẽ hai đại lượng này trên cùng một hệ trục tọa độ, như trên hình 6-12.

```
>> f1=cos(4.3*t);
>> plot(t,x(:,1),'k-', t,f1,'k--'), xlabel('t'),
>> axis([0 2*pi -1.2 1.9]), legend('x_1','cos(4.3t)')
```



Hình 6-12. So sánh nghiệm x_1 với $\cos(4.3t)$, điều kiện đầu $[1, 1]$

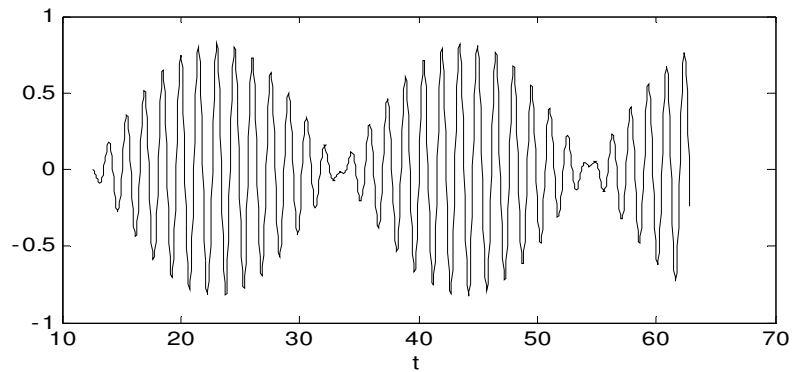


Hình 6-13. So sánh x_1 với \sin và $\cos(4.3t)$, khi điều kiện đầu là $[0, 0]$

Hai đường này bắt đầu từ hai điểm rất gần nhau, nhưng chúng xa dần khi thời gian tăng lên. Nhưng so với trường hợp trước, khi điều kiện đầu là $[0, 0]$, ta thấy các nghiệm tăng dần (hình 6-13), như vậy có thể kết luận là các nghiệm không tăng lên mãi theo thời gian. Để đánh giá được nghiệm này, ta cần phải tính toán nghiệm trong một khoảng thời gian dài hơn, như trường hợp sau đây:

```
>> [t,x] = ode45('eqx2',[4*pi 20*pi],[0,0]);
>> plot(t,x(:,1)), xlabel('t')
```

Kết quả được đưa ra như trên hình 6-14.



Hình 6-14. Hiện tượng phách trong trường hợp điều kiện đầu $[0, 0]$

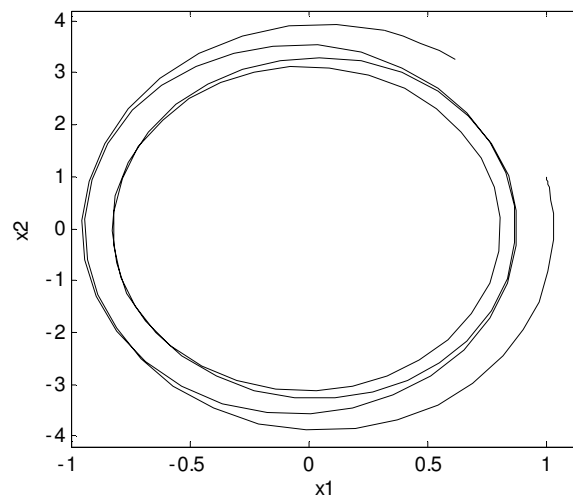
Ta thấy hiện tượng phách (beat phenomena) đã xảy ra với hệ, đó là hiện tượng biên độ dao động tăng giảm, tăng giảm theo chu trình. Hiện tượng này xảy ra khi tần số lực kích động gần với tần số riêng của hệ. Xem xét kỹ phương trình vi phân

$$\ddot{y} + 16y = \sin(4.3t) \text{ suy ra } \ddot{y} + 4^2 y = \sin(4.3t)$$

ta thấy tần số riêng của hệ là 4 và tần số lực kích động là 4.3.

Hình ảnh quỹ đạo pha trong trường hợp điều kiện đầu là $[1, 1]$ được đưa ra như trên hình 6-15.

```
>> [t,x] = ode45('eqx2',[0 2*pi],[1, 1]);
>> plot(x(:,1), x(:,2),'k-'),xlabel('x1'), ylabel('x2'),
>> axis([-1 1.15 -4.2 4.2])
```



Hình 6-15. Quỹ đạo pha trong trường hợp điều kiện đầu là $[1, 1]$

6.3 Bài tập thực hành

Giải phương trình vi phân với các điều kiện đầu sau:

1. $\frac{dy}{dt} = -2.3y, \quad y(0) = 0, \quad t_{end} = 10$
2. $\frac{dx_1}{dt} = 2x_1x_2, \quad \frac{dx_2}{dt} = -x_1^2, \quad x_1(0) = x_2(0) = 0, \quad t_{end} = 10$
3. $\frac{dx_1}{dt} = x_2, \quad \frac{dx_2}{dt} = -x_1, \quad x_1(0) = x_2(0) = 1, \quad t_{end} = 10$
4. $\frac{dy}{dt} = -ty + 1, \quad y(0) = 1, \quad t_{end} = 10$
5. $\frac{dy}{dt} = t^2y, \quad y(0) = 1, \quad 0 \leq t \leq 2$

6. Phương pháp số nào được sử dụng trong ode23 và ode45?

7. Sử dụng ode23 và ode45 giải phương trình vi phân

$$\frac{dy}{dt} = y, \quad y(0) = 1, \quad t_{end} = 10$$

Hãy cho biết số điểm đã được sử dụng với ode23 và ode45.

8. Giải phương trình vi phân sau

$$\frac{dy}{dt} = \frac{2}{\sqrt{1-t^2}}, \quad y(0) = 1, \quad -1 < t < 1$$

9. Giải và vẽ đồ thị và quỹ đạo pha của phương trình vi phân sau

$$\frac{d^2y}{dt^2} - 2\frac{dy}{dt} + y = e^{-t}, \quad y(0) = 2, \quad \dot{y}(0) = 0$$

10. Phương trình vi phân dao động cưỡng bức của hệ tuyến tính n bậc tự do được cho dạng ma trận như sau

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{B}\dot{\mathbf{q}} + \mathbf{C}\mathbf{q} = \mathbf{f}(t), \quad \text{với điều kiện đầu } \mathbf{q}(0) = \mathbf{q}_0, \quad \dot{\mathbf{q}}(0) = \dot{\mathbf{q}}_0.$$

với \mathbf{q} là vectơ tọa độ suy rộng, $\mathbf{M}, \mathbf{B}, \mathbf{C}$ là các ma trận vuông cấp n , tương ứng là ma trận khối lượng, ma trận cản, và ma trận độ cứng. Vectơ lực kích động $\mathbf{f}(t)$.

Bằng cách hạ bậc ta nhận được

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{M}^{-1}(\mathbf{f}(t) - \mathbf{B}\mathbf{v} - \mathbf{C}\mathbf{q}) \end{aligned}$$

Hãy viết các m-file thể hiện các phương trình vi phân và thực hiện việc giải số tìm dao động của hệ, với

$$\mathbf{M} = \begin{bmatrix} 12 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 20 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 10 & -10 & 0 \\ -10 & 20 & -10 \\ 0 & -10 & 10 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 1000 & -1000 & 0 \\ -1000 & 2000 & -1000 \\ 0 & -1000 & 1000 \end{bmatrix}, \quad \mathbf{f}(t) = \begin{bmatrix} 5 \sin 10t \\ 0 \\ 0 \end{bmatrix}.$$

Các phép biến đổi tích phân và Matlab

Các phép biến đổi như Laplace, z và Fourier được sử dụng rộng rãi trong khoa học và kỹ thuật. Qua các phép biến đổi này các tín hiệu được chuyển đổi sang một cách biểu diễn mới. Trong chương này, các lệnh cơ bản của Matlab sử dụng cho các phép biến đổi Laplace, Fourier và Fourier nhanh được trình bày.

7.1 Phép biến đổi Laplace

Biến đổi Laplace một hàm theo thời gian $f(t)$ được đưa ra bởi công thức tích phân

$$\ell\{f(t)\} = \int_0^{\infty} f(t)e^{-st} dt$$

Chúng ta thường ký hiệu biến đổi Laplace của hàm $f(t)$ là $F(s)$. Hàm $F(s)$ được gọi là hàm ảnh của hàm $f(t)$:

$$F(s) = \ell\{f(t)\}$$

Một trong những ứng dụng của phép biến đổi Laplace là để biến đổi phương trình vi phân tuyến tính thành một phương trình đại số. Như chúng ta biết việc giải phương trình đại số sẽ dễ dàng hơn so với việc giải một phương trình vi phân, tuy nhiên không phải lúc nào cũng như vậy. Với Matlab, việc biến đổi Laplace một hàm theo thời gian được thực hiện một cách thật đơn giản. Chúng ta có thể tìm bảng về phép biến đổi Laplace của các hàm cơ bản trong các sách về phương trình vi phân, kỹ thuật điện, kỹ thuật điều kiện,...

Để biến đổi Laplace trong Matlab, ta gọi hàm `laplace(f(t))`. Việc này chỉ tính toán được với hàm được đưa vào ở dạng ký tự chữ, symbolic. Ví dụ đơn giản nhất là tìm hàm ảnh của một hằng số, hàm $f(t) = a$. Trước hết chúng ta định nghĩa các biến ký tự:

```
>> syms s t
```

Hãy thử với $f(t) = 1$, ta gõ vào dòng lệnh

```
>> laplace(1)
```

```
??? Function 'laplace' is not defined for values of class 'double'.
```

Matlab báo lỗi, vì 1 không phải là kiểu symolic. Do đó ta cần khai báo

```
>> syms a
```

Sau đó gọi hàm laplace đối với hằng số a:

```
>> laplace(a)
```

```
ans = 1/s^2
```

Tiếp theo là ví dụ biến đổi laplace của một số hàm lũy thừa của t , $f(t) = t^m$

```
>> laplace(t^2)
```

```
ans = 2/s^3
```

```
>> laplace(t^7)
```

```
ans = 5040/s^8
```

```
>> laplace(t^5)
```

```
ans = 120/s^6
```

Từ đây có thể suy luận ra công thức quen thuộc của phép biến đổi Laplace

$$F(s) = \mathcal{L}\{t^m\} = \frac{m!}{s^{m+1}}$$

Sau đây là hàm ảnh của một số hàm hay được sử dụng trong khoa học và kỹ thuật.

Trước hết là hàm e mũ giảm $f(t) = e^{-bt}$:

```
>> laplace(exp(-b*t))
```

```
ans = 1/(s+b)
```

Biến đổi Laplace các hàm sin và cos:

```
>> laplace(cos(w*t))
```

```
ans = s/(s^2+w^2)
```

```
>> laplace(sin(w*t))
```

```
ans = w/(s^2+w^2)
```

Chúng ta cũng có thể biến đổi laplace các hàm sin- và cos hyperpol:

```
>> laplace(cosh(b*t))
```

```
ans = s/(s^2-b^2)
```

Phép biến đổi Laplace có tính chất tuyến tính, nghĩa là

$$\mathcal{L}\{\alpha_1 f_1(t) + \alpha_2 f_2(t)\} = \alpha_1 F_1(s) + \alpha_2 F_2(s)$$

Chúng ta sẽ kiểm nghiệm lại tính chất tuyến tính này qua ví dụ

```
>> f = 5 + exp(-2*t)
```

```
>> laplace(f)
```

```
ans = 5/s+1/(s+2)
```

7.2 Phép biến đổi Laplace ngược

Để có thể ứng dụng phép biến đổi Laplace cho các bài toán kỹ thuật, phép biến đổi Laplace ngược là cần thiết. Nếu $\ell\{f(t)\} = F(s)$, thì phép biến đổi Laplace ngược ký hiệu là

$$\ell^{-1}\{F(s)\} = f(t), \quad t \geq 0,$$

toán tử ℓ^{-1} biến đổi hàm ảnh $F(s)$ về hàm gốc của nó $f(t)$. Ví dụ

$$\ell^{-1}\left(\frac{\omega}{s^2 + \omega^2}\right) = \sin \omega t, \quad t \geq 0$$

Thông thường để tìm hàm gốc $f(t)$ từ hàm ảnh $F(s)$, người ta hay sử dụng phương pháp tra bảng "các hàm gốc – hàm ảnh tương ứng".

Với Matlab việc tìm hàm gốc từ hàm ảnh tương đối đơn giản, bạn chỉ việc sử dụng lệnh `ilaplace(F(s))`. Dưới đây là một số ví dụ về việc sử dụng hàm `ilaplace` đối với các hàm đơn giản:

```
>> ilaplace(1/s^3)
ans = 1/2*t^2
>> ilaplace(1/s^7)
ans = 1/720*t^6
>> ilaplace(2/(w+s))
ans = 2*exp(-w*t)
>> ilaplace(s/(s^2+4))
ans = cos(2*t)
```

Tuy nhiên trong các bài toán thực tế ta ít gặp các hàm ảnh đơn giản như trên. Chẳng hạn cần tìm hàm gốc của hàm ảnh sau

$$F = \frac{5-3s}{2+5s}$$

Ta cần nhập hàm này vào, với khai báo s ở dạng symbolic:

```
>> syms s
>> F = (5-3*s)/(2+5*s);
```

Khi biến đổi ngược hàm này ta nhận được hàm gốc trong đó có hàm delta Dirac:

```
>> ilaplace(F)
ans = -3/5*dirac(t)+31/25*exp(-2/5*t)
```

Ở đây cần nhắc lại định nghĩa hàm delta Dirac

$$\delta(t) = \begin{cases} 0 & t > 0 \\ \infty & t = 0 \end{cases} \quad \text{và} \quad \int_{-\infty}^{+\infty} \delta(t-a)f(t)dt = f(a).$$

Một ví dụ phức tạp hơn

$$F = \frac{-s^2 - 9s + 4}{s^2 + s + 2}$$

```
>> F = (-s^2 - 9*s + 4)/(s^2 + s + 2);
>> ilaplace(F)
ans = -dirac(t) - 8*exp(-1/2*t)*cos(1/2*7^(1/2)*t)+20/7*7^(1/2)*
exp(-1/2*t)*sin(1/2*7^(1/2)*t)
```

Hay một hàm đơn giản hơn

$$F = \frac{1}{(s+7)^2}$$

```
>> f = ilaplace(1/(s+7)^2)
f = t*exp(-7*t)
```

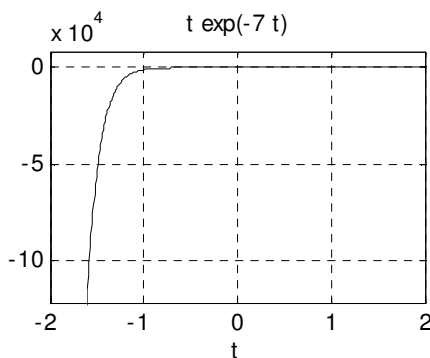
Để vẽ đồ thị các hàm gốc theo thời gian ta có thể sử dụng lệnh ezplot theo cú pháp

```
ezplot(f, [t_start, t_end])
```

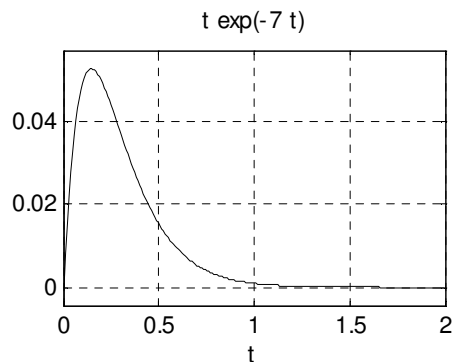
Với hàm f trên ta gõ lệnh

```
>> ezplot(f, [-2, 2])
>> ezplot(f, [0, 2])
```

và nhận được đồ thị như trên hình 7-1 và 7-2.



Hình 7-1. Hàm te^{-7t} , $-2 \leq t \leq 2$



Hình 7-2. Hàm te^{-7t} , $0 \leq t \leq 2$

Một ví dụ khác là tìm hàm gốc và vẽ đồ thị của nó, khi biết hàm ảnh

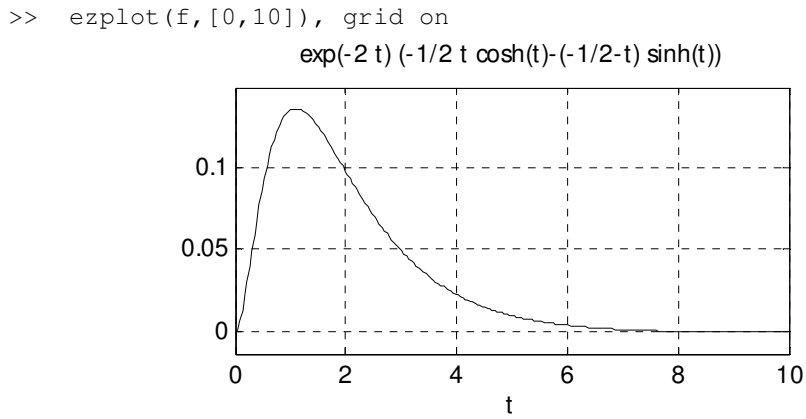
$$F = \frac{2s + 3}{(s+1)^2(s+3)^2}$$

Gọi hàm ilaplace và nhận được hàm gốc như sau:

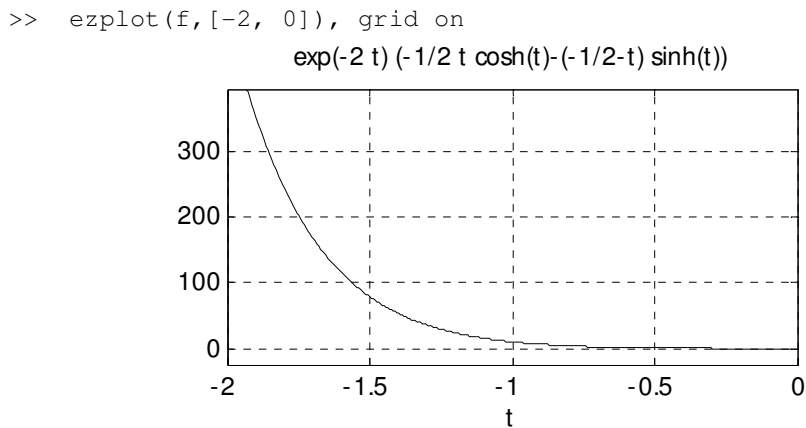
```
>> f = ilaplace((2*s+3)/((s+1)^2*(s+3)^2))
```

```
f = exp(-2*t)*(-1/2*t*cosh(t)+(1/2+t)*sinh(t))
```

Vẽ đồ thị hàm gốc trong 10 giây đầu tiên ta nhận được đồ thị như trên hình 7-3.



Hình 7-3. Hàm gốc với $0 \leq t \leq 10$



Hình 7-4. Hàm gốc với $-2 \leq t \leq 0$

7.3 Áp dụng phép biến đổi Laplace giải phương trình vi phân

Nhờ phép biến đổi Laplace ta có thể biến đổi phương trình vi phân tuyến tính hệ số hằng số về dạng phương trình đại số, từ đó giải tìm hàm ảnh nghiệm của phương trình vi phân. Sau đó sử dụng biến đổi Laplace ngược để tìm hàm gốc – nghiệm của phương trình vi phân. Trước hết nêu ra một tính chất nữa của phép biến đổi Laplace, theo định lý vi phân nếu $\mathcal{L}\{f(t)\} = F(s)$ thì ta có

$$\ell\left\{\frac{df}{dt}\right\} = sF(s) - f(0), \quad \ell\left\{\frac{d^2f}{dt^2}\right\} = s^2F(s) - sf(0) - \dot{f}(0).$$

Tuy nhiên chúng ta không thể sử dụng lệnh Laplace tác động trực tiếp lên phép tính đạo hàm. Để giải phương trình vi phân tuyến tính hệ số hằng bằng phép biến đổi Laplace, ta cần phải biến đổi Laplace phương trình vi phân đã cho và đưa kết quả đó vào Matlab. Sau đó sử dụng phép biến đổi Laplace ngược để nhận được nghiệm. Ví dụ sau đây sẽ minh họa công việc đó.

Xét hệ dao động khối lượng – lò xo – cản nhớt, được mô tả bằng phương trình vi phân

$$m\ddot{x} + b\dot{x} + cx = u(t) + \alpha\dot{u}(t)$$

Trước hết sử dụng biến đổi Laplace hai vế

$$\ell\{m\ddot{x} + b\dot{x} + cx\} = \ell\{u(t) + \alpha\dot{u}(t)\}$$

ta nhận được

$$m[s^2X(s) - s\dot{x}(0) - \dot{x}(0)] + b[sX(s) - x(0)] + cX(s) = U(s) + \alpha[sU(s) - u(0)]$$

Từ đây giải được

$$X(s) = \frac{[1 + \alpha s]}{[ms^2 + bs + c]}U(s) - \frac{\alpha u(0) + msx(0) + m\dot{x}(0) + bx(0)}{[ms^2 + bs + c]}$$

Thay các thông số của hệ $m = 1, b = 1.2, c = 1$ và với điều kiện đầu $x(0) = 0, \dot{x}(0) = 0$ vào công thức ta có

$$X(s) = \frac{[1 + \alpha s]}{s^2 + 1.2s + 1}U(s) - \frac{\alpha u(0)}{s^2 + 1.2s + 1}.$$

Sau đây ta sẽ tìm hàm gốc $x(t)$ từ hàm ảnh $X(s)$ và vẽ các đồ thị hàm này với các thông số khác nhau của α , $\alpha = 0, 2, 5$. Cho biết hàm kích động $u(t)$ là hàm bước nhảy đơn vị hay còn gọi là hàm Heaviside

$$u(t) = \begin{cases} 0 & t < 0 \\ 1 & 0 \leq t \end{cases} \quad \text{hay } u(t) = \text{heaviside}(t).$$

Hàm này được vẽ với dòng lệnh

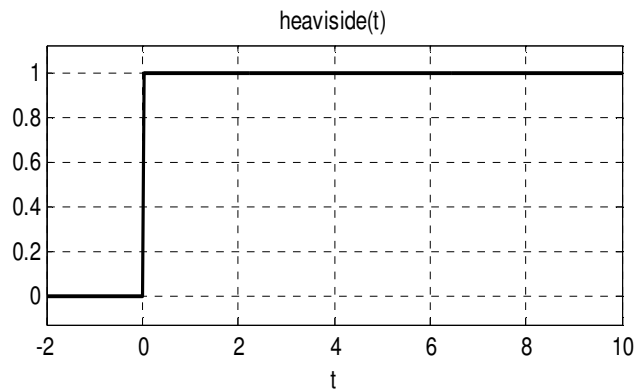
```
>> ezplot(heaviside(t), [-2, 10]), grid on
```

```
>> laplace(heaviside(t))
```

```
ans = 1/s
```

và như vậy

$$X(s) = \frac{1 + \alpha s}{s^2 + 1.2s + 1} \frac{1}{s}, \quad \text{với } u(0) = 0.$$



Hình 7-5. Đồ thị hàm heaviside(t), $-2 \leq t \leq 10$

Để đưa ba giá trị của α vào ta đặt: $a1 = 0$, $a2 = 2$, $a3 = 5$:

```
>> a1 = 0; a2 = 2; a3 = 5;
```

Mẫu số của hàm ảnh $X(s)$ được đưa vào dạng

```
>> d = s^2 + (1.2)*s + 1;
```

Với mỗi giá trị của hệ số α ta có

```
>> X1 = ((1+ a1*s)/d)*(1/s);
```

```
>> X2 = ((1+ a2*s)/d)*(1/s);
```

```
>> X3 = ((1+ a3*s)/d)*(1/s);
```

Và bây giờ sử dụng biến đổi Laplace ngược

```
>> x1 = ilaplace(X1)
```

```
x1 = -exp(-3/5*t)*cos(4/5*t)-3/4*exp(-3/5*t)*sin(4/5*t)+1
```

```
>> x2 = ilaplace(X2)
```

```
x2 = 1-exp(-3/5*t)*cos(4/5*t)+7/4*exp(-3/5*t)*sin(4/5*t)
```

```
>> x3 = ilaplace(X3)
```

```
x3 = 1-exp(-3/5*t)*cos(4/5*t)+11/2*exp(-3/5*t)*sin(4/5*t)
```

Sử dụng các lệnh subplot, ezplot ta nhận được đồ thị hàm $x(t)$ trong các trường hợp như trên hình 7-6.

```
>> subplot(3,1,1),
```

```
>> ezplot(x1,[0 16]), grid on, axis([0 16 -0.1 1.5])
```

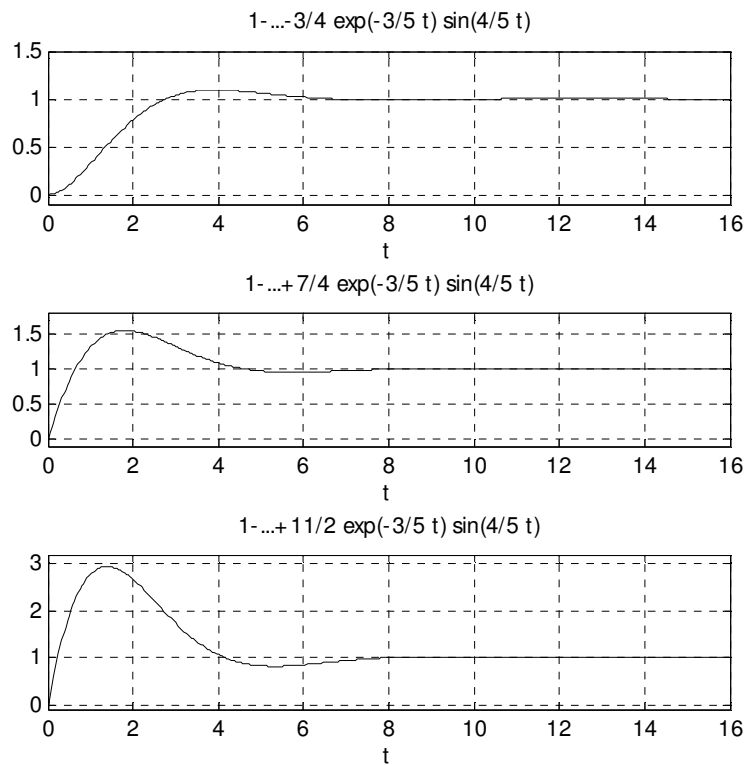
```
>> subplot(3,1,2),
```

```
>> ezplot(x2,[0 16]), grid on, axis([0 16 -0.1 1.8])
```

```
>> subplot(3,1,3),
```

```
>> ezplot(x3,[0 16]), grid on, axis([0 16 -0.1 3.2])
```

Kết quả nhận được như trên hình 7-6.



Hình 7-6. Đáp ứng của hệ theo thời gian

7.4 Phép biến đổi Fourier

Phép biến đổi Fourier một hàm $f(t)$ được định nghĩa theo công thức tích phân

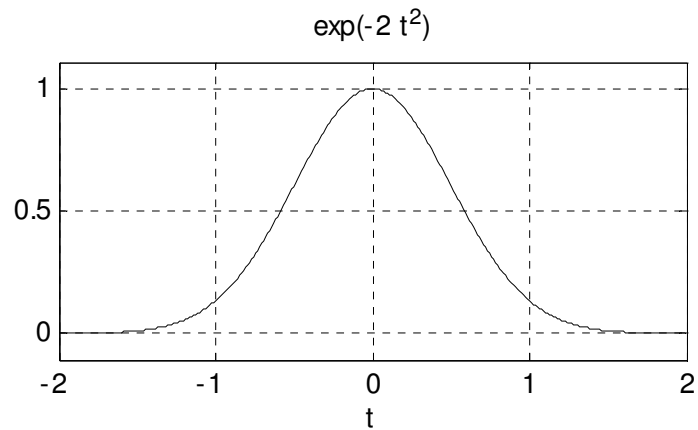
$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t} dt$$

Chúng ta có thể tìm hàm ảnh của phép biến đổi Fourier trong Matlab bằng lệnh `fourier`. Ý nghĩa của phép biến đổi Fourier cho phép biến đổi một tín hiệu trong miền thời gian $f(t)$ sang biểu diễn trong miền tần số $F(\omega)$. Ví dụ xét biến đổi Fourier của hàm điều hòa $\sin(t)$ cho ta hai hàm delt Dirac:

```
>> syms t
>> fourier(sin(t))
ans = i*pi*(-dirac(w-1)+dirac(w+1))
```


Biến đổi Fourier hàm phân bố Gaussian (hình 7-7) và vẽ đồ thị các hàm này ta được hình 7-8.

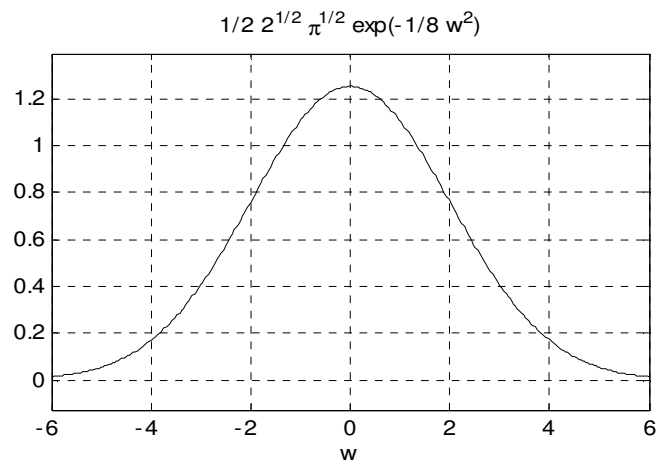
```
>> syms t
>> f = exp(-2*t^2);
>> ezplot(f,[-2,2]), grid on
```



Hình 7-7. Hàm phân bố Gaussian, e^{-2t^2}

Nhờ lệnh fourier ta nhận được hàm ảnh

```
>> FT = fourier(f)
FT = 1/2*2^(1/2)*pi^(1/2)*exp(-1/8*w^2)
```



Hình 7-8. Kết quả biến đổi Fourier hàm Gaussian

Hàm ảnh này cũng có dạng tương tự như hàm phân bố Gaussian ban đầu nhưng có dải rộng hơn và đỉnh cao hơn.

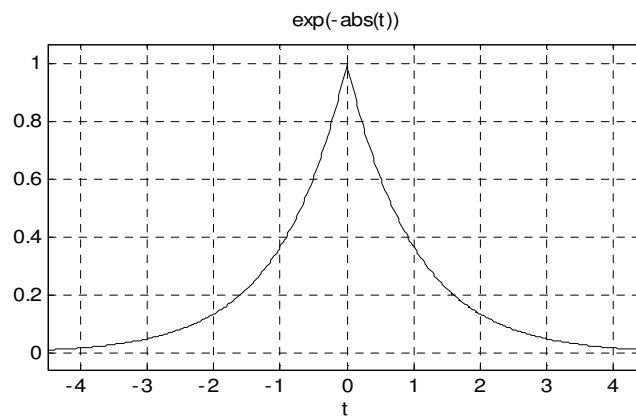
Một ví dụ khác với

$$f(t) = e^{-|x|} = \begin{cases} e^x & x < 0 \\ e^{-x} & 0 < x \end{cases}$$

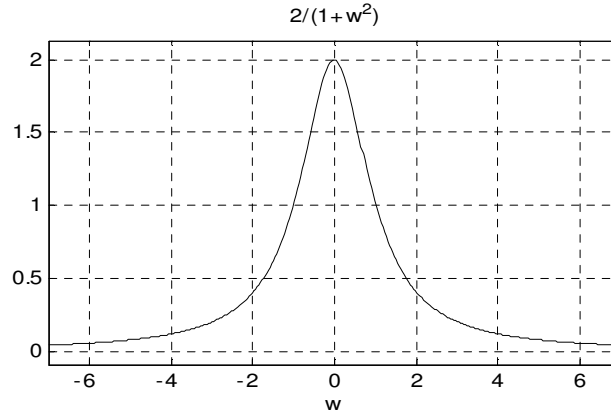
Ta sẽ định nghĩa hàm này trong Matlab bằng cách gõ vào dòng sau:

```
>> syms x
>> f = exp(-abs(x));
```

Hàm này có đồ thị như trên hình 7-9.



Hình 7-9. Đồ thị hàm $f(t) = e^{-|x|}$



Hình 7-10. Kết quả biến đổi Fourier hàm $f(t) = e^{-|x|}$

Thực hiện biến đổi Fourier và nhận được kết quả như trên hình 7-10.

```
>> FT = fourier(f)
FT = 2/(w^2+1)
```

7.5 Phép biến đổi Fourier ngược

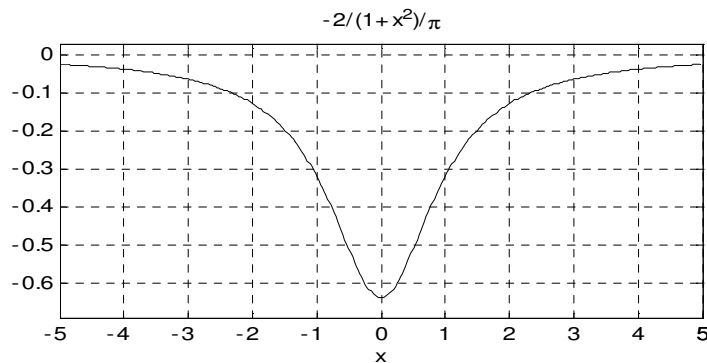
Phép biến đổi Fourier ngược trong Matlab được thực hiện với lệnh *ifourier*. Ví dụ với hàm $F(\omega) = -2e^{-|\omega|}$ khi biến đổi ngược tìm hàm $f(x)$ ta thực hiện:

```
>> f = ifourier(-2*exp(-abs(w)))
>> ezplot(f,[-5,5]), grid on
```

ta được

```
f = -2/(x^2+1)/pi
```

Đồ thị của hàm này như được vẽ trên hình 7-11.



Hình 7-11. Kết quả biến đổi Fourier hàm $f(t) = e^{-|x|}$

7.6 Phép biến đổi Fourier một tín hiệu rời rạc

Cho một tín hiệu ở dạng vectơ số, x_0, x_1, \dots, x_{N-1} , qua phép biến đổi Fourier rời rạc ta được một dãy số phức ở dạng vectơ gồm N phần tử xác định theo công thức

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn}, \quad k = 0, 1, \dots, N-1$$

ở đây theo công thức Euler có thể biểu diễn dạng

$$e^{-\frac{2\pi i}{N} kn} = \cos(2\pi kn / N) - i \sin(2\pi kn / N).$$

Ta có thể biểu diễn vectơ ảnh phức $\mathbf{X} = [X_0, X_1, \dots, X_{N-1}]$ bằng độ dài A_k và góc pha φ_k theo liên hệ

$$A_k = |X_k| = \sqrt{\text{Re}(X_k)^2 + \text{Im}(X_k)^2},$$

$$\varphi_k = \arg(X_k) = \text{atan2}(\text{Im}(X_k), \text{Re}(X_k))$$

Phép biến đổi ngược được thực hiện theo công thức

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn}, \quad n = 0, 1, \dots, N-1$$

7.7 Phép biến đổi Fourier nhanh (FFT)

Hàm fft của Matlab sử dụng để tính số khi thực hiện biến đổi Fourier nhanh các véc-tơ số. Việc sử dụng lệnh fft này thể hiện trong ví dụ sau.

Giả sử có tín hiệu chuẩn

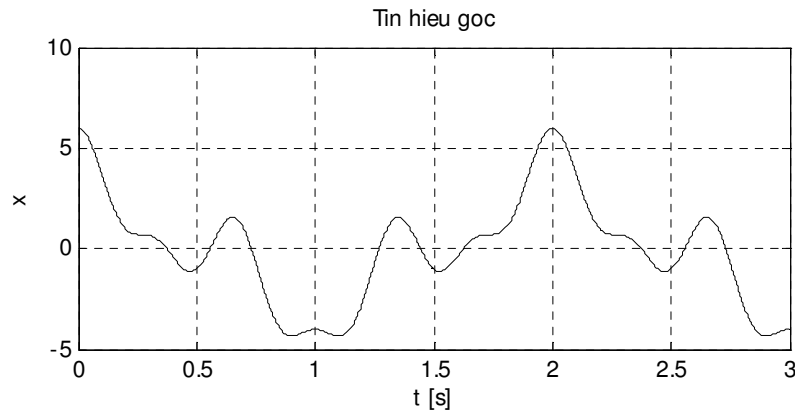
$$x(t) = 3 \cos(\pi t) + 2 \cos(3\pi t) + \cos(6\pi t)$$

Ta sẽ tạo ra một véc-tơ số chứa tín hiệu này trong khoảng thời gian 10 giây. Ta sẽ cho tín hiệu này bị nhiễu để sử dụng fft khảo sát các tần số chứa trong tín hiệu.

Trước hết chia khoảng thời gian thành các điểm rời rạc

```
>> t = [0:0.001:6]; % Time vector
>> N = length(t);
>> x = 3*cos(pi*t) + 2*cos(3*pi*t) + cos(6*pi*t);
>> plot(t(1:N/2), x(1:N/2), 'k-', 'linewidth', 1), grid on
>> xlabel('t [s]'), ylabel('x')
>> title('Tín hiệu gốc')
```

Sau đó tính véc-tơ tín hiệu chuẩn tại các điểm thời gian. Tín hiệu chuẩn trong ba giây đầu tiên được vẽ như trên hình 7-12.

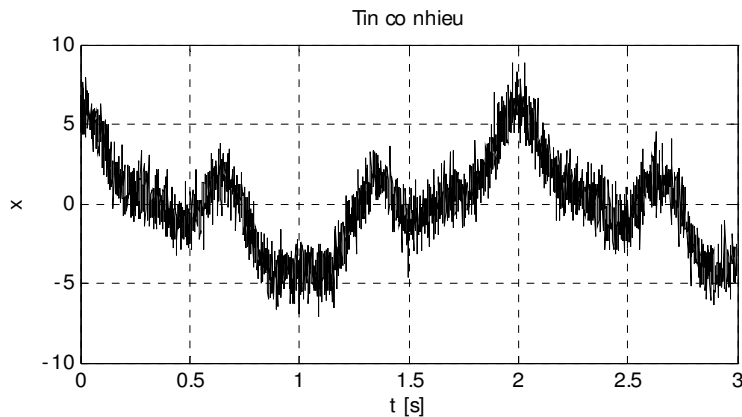


Hình 7-12. Tín hiệu chuẩn trong ba giây đầu tiên

Để tạo nhiễu, chúng ta cho tạo ra các số ngẫu nhiên bằng lệnh randn(m) và cộng thêm vào tín hiệu chuẩn.

```
>> x_noisy = x + randn(size(t)); % Tao tin hieu nhieu
>> plot(t(1:N/2),x_noisy(1:N/2), 'k-', 'linewidth',1), grid on
>> xlabel('t [s]'), ylabel('x'),
>> title('Tin co nhieu')
```

Tín hiệu nhiễu được vẽ ra như trên hình 7-13.



Hình 7-13. Tín hiệu có nhiễu trong ba giây đầu tiên

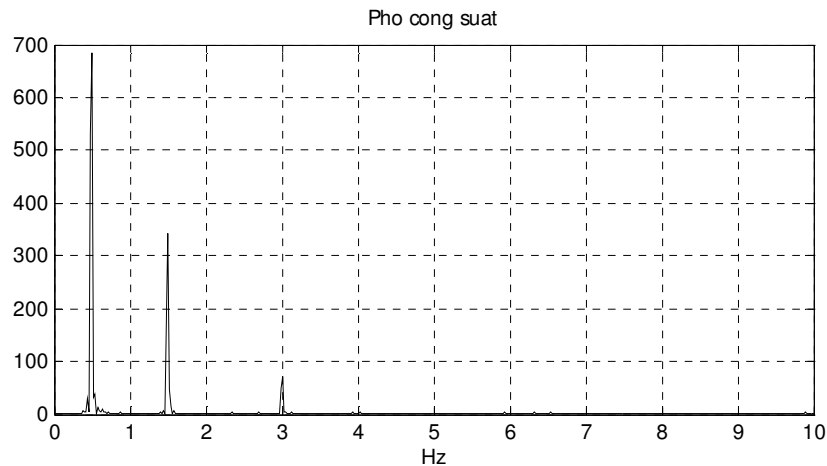
Bây giờ ta sẽ sử dụng phép biến đổi Fourier nhanh để phân tích xem trong tín hiệu nhiễu có chứa những tần số nào. Phép biến đổi Fourier nhanh dựa trên n điểm tín hiệu được thực hiện bởi lệnh `fft(f, n)`. Để thực hiện ví dụ này, ta soạn một Script như sau :

```
% Su dung phan tich Fourier nhanh tim tan so trong tin hieu
Fs = 20; % Tan so lay mau, Fs >= 2 tan so cua tin hieu
T = 1/Fs; % Chu ky lay mau
L = 600; % Chieu dai tin hieu,
t = (0:L-1)*T; % Cac thoi diem lay mau
x = 3*cos(pi*t)+2*cos(3*pi*t)+cos(6*pi*t);
x_noisy = x + randn(size(t)); % Tao tin hieu nhieu

NFFT = 2^nextpow2(L); % Next power of 2 from length of y
N = NFFT; % so diem su dung trong fft, la 2^z
FT = fft(x_noisy, N); % goi ham fft
P = FT.*conj(FT)/N; % cong suat trong tin hieu,
% hay binh phuong bien do
f = Fs/2*linspace(0,1,N/2+1)
plot(f,P(1:N/2+1), 'k-', 'linewidth',1), grid on
xlabel('Hz'), title('Pho cong suat')
```

Công suất trong tín hiệu có thể tìm được bằng tích của hàm ảnh Fourier với liên hợp phức của nó và chia cho tổng số điểm.

Khi vẽ ra đồ thị phổ công suất ta thấy các tần số trong tín hiệu xuất hiện một cách rõ nét. Trong hình 7-14 ta thấy rõ ba đỉnh ứng với ba tần số trong tín hiệu chuẩn ban đầu $x(t) = 3 \cos(\pi t) + 2 \cos(3\pi t) + \cos(6\pi t)$. Rõ ràng là trong tín hiệu này chưa ba tần số: $f_1 = 0.5$, $f_2 = 1.5$, $f_3 = 3$. Độ cao của các đỉnh phản ánh các biên độ trong tín hiệu chuẩn ứng với các tần số, hình 7-14. Hãy so sánh !



Hình 7-14. Phổ công suất của tín hiệu sau khi phân tích Fourier

7.8 Bài tập thực hành

- Thực hiện biến đổi Laplace các hàm gốc sau

$$g(t) = te^{-3t}, \quad f(t) = 8 \sin 5t - e^{-t} \cos 2t$$

- Thực hiện biến đổi Laplace ngược các hàm ảnh sau

$$F(s) = \frac{1}{s} - \frac{2s^2 + 3}{s^2 + 9}, \quad H(s) = \frac{s}{s^2 - 49} - \frac{3}{s^2 - 9}$$

- Sử dụng biến đổi Laplace giải phương trình vi phân sau

$$\frac{dy}{dt} + y = 2u(t)$$

với $u(t)$ là hàm heaviside. Cho tất cả các điều kiện đầu bằng 0, vẽ ra và nhận xét ứng xử cường bức của hệ.

- Thực hiện biến đổi Fourier các hàm gốc sau

$$g(x) = x^2, \quad f(x) = x \cos x$$

5. Thực hiện biến đổi Fourier ngược hàm ảnh sau

$$F(\omega) = \frac{1}{1 + i\omega}$$

6. Kết quả biến đổi Fourier nhanh vectơ

$a = [2, 4, -1, 2]$ là gì?

```
>> a=[2 4 -1 2] ; >> fft(a,4)
```

7. Cho tín hiệu chuẩn $x(t) = \sin(\pi t) + 2 \sin(4\pi t)$. Hãy tạo thêm nhiễu cho tín hiệu bằng cách sử dụng $x + \text{randn}(\text{size}(t))$. Hãy tính và vẽ ra mật độ phổ công suất, chỉ ra đỉnh cao nhất và tần số tương ứng.

Giới thiệu về Simulink

Công cụ simulink trong Matlab cho phép mô phỏng các hệ động lực với một giao diện đồ họa đặc biệt bằng các khối kết nối với nhau. Mục đích của chương này là giới thiệu về simulink thông qua các ví dụ. Sau khi thiết lập các phương trình vi phân mô tả hệ, một sơ đồ khối tương ứng được đưa ra và từ đó sử dụng các khối của simulink để mô phỏng hệ.

8.1 Khái niệm về Simulink

Simulink là một phần mở rộng của MATLAB được phát triển bởi Mathworks Inc. Simulink là một gói chương trình để mô hình hóa, mô phỏng và phân tích các hệ động lực. Với simulink ta có thể mô hình hóa các hệ thống tuyến tính và phi tuyến liên tục theo thời gian, rời rạc theo thời gian, hoặc hỗn hợp cả liên tục và rời rạc theo thời gian. Các hệ cũng có thể là đa tốc độ khác nhau, nghĩa là các phần khác nhau có thể được lấy mẫu hoặc cập nhật số liệu ở các tốc độ khác nhau.

Để mô hình hóa simulink cung cấp một giao diện đồ họa (GUI) cho việc xây dựng các mô hình như là một sơ đồ khối, sử dụng các thao tác nhấn và kéo chuột. Với giao diện này, bạn có thể vẽ ra mô hình giống như khi bạn phác thảo nó bằng bút chì trên giấy. Đây là điều khác xa so với gói mô phỏng đã giới thiệu trong phần giải phương trình vi phân, ở đó yêu cầu chúng ta phải thiết lập các phương trình vi phân hoặc các phương trình sai phân trong một ngôn ngữ hoặc lập trình. Simulink bao gồm một thư viện toàn bộ các khối như khối cấp nguồn tín hiệu (sources), khối các phần tử tuyến tính và phi tuyến, các mối nối, và khối nhận và hiển thị tín hiệu (sinks). Người sử dụng cũng có thể thay đổi để tạo ra các khối riêng của mình.

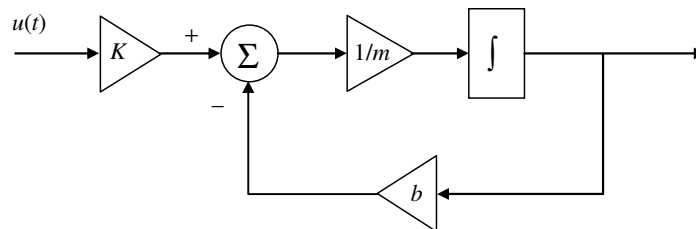
Các mô hình trong simulink được xây dựng có thứ bậc hay còn gọi là mô hình phân cấp, do đó người sử dụng có thể xây dựng mô hình theo hướng từ dưới lên hay từ trên xuống. Người sử dụng vừa có thể quan sát hệ thống ở mức tổng quan, vừa có thể quan sát ở mức độ chi tiết bằng cách nhấn kép chuột vào từng khối con của hệ thống. Với cách xây dựng như thế, người sử dụng có thể thấy được một hệ được tổ chức như thế nào và các phần của nó tương tác với nhau ra sao.

Sau khi xây dựng được mô hình, người sử dụng có thể mô phỏng nó. Việc chọn các phương pháp tích phân có thể từ menu của simulink hoặc bằng cách gõ vào các lệnh từ bàn phím trong cửa sổ lệnh của Matlab. Việc sử dụng menu đặc biệt tiện lợi cho các công việc có tính chất tương tác lẫn nhau, còn cách sử dụng dòng lệnh thích hợp khi thực hiện một loạt các mô phỏng. Bằng việc sử dụng các khối Scope hay các khối hiển thị khác, người sử dụng có thể quan sát các kết quả khi mô phỏng đang chạy. Ngoài ra, người sử dụng có thể thay đổi các thông số và quan sát được ngay tác động của các tham số thay đổi đến ứng xử của hệ. Các kết quả mô phỏng có thể được đưa vào trong môi trường làm việc của Matlab cho việc xử lý tiếp theo và hiển thị bằng hình ảnh.

Để phân tích mô hình simulink cung cấp các công cụ tuyến tính hóa và cắt xén (trimming), các công cụ này có thể được truy cập từ dòng lệnh Matlab, và nhiều các công cụ khác trong Matlab và các ứng dụng của nó. Do Matlab và Simulink đã được tích hợp nên người sử dụng có thể mô phỏng, phân tích, và sửa đổi các mô hình của mình trong cả hai môi trường trong bất kỳ thời điểm nào.

Như đã nói trên mô hình xây dựng trong simulink là đồ khối với những đường kết nối tương ứng. Để mô hình hóa được một hệ thống trong simulink, đòi hỏi người sử dụng phải có kiến thức về mô tả hệ thống bằng sơ đồ khối. Người sử dụng cần phải mô tả một hệ thống như được cấu tạo từ các khối thành phần có đầu vào và đầu ra, tương tác giữa các phần tử được thể hiện bằng các đường nối giữa các khối, đó là các dòng tín hiệu. Chẳng hạn ta có thể tạo ra một sơ đồ khối của một hệ động lực được mô tả bằng phương trình vi phân sau

$$m\dot{v} + bv = Ku(t) \quad (*)$$



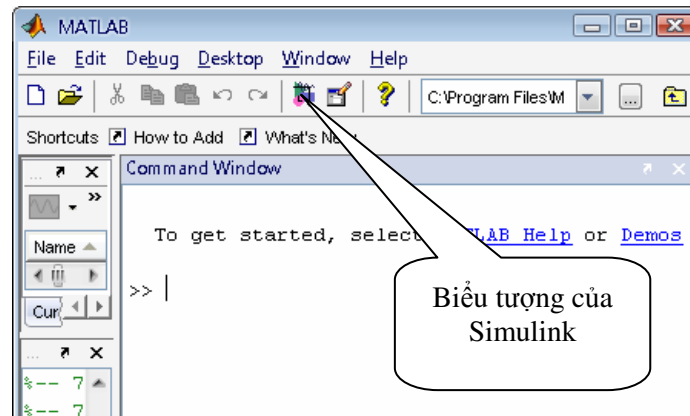
Hình 8-1. Sơ đồ khối của hệ động lực (*)

Dạng mô tả đồ thị trên sẽ được đưa vào simulink để mô phỏng hệ. Để thực hiện tốt việc mô hình hóa trên đòi hỏi người sử dụng phải có kiến thức về kỹ thuật điều khiển và lý thuyết hệ thống. Ở đây không thể đi sâu vào các lĩnh vực này, người sử dụng có thể tham khảo thêm trong các tài liệu về kỹ thuật điều khiển và lý thuyết hệ thống. Trong phần này, giới hạn chỉ giới thiệu simulink để giải số các phương trình vi phân mô tả hệ động lực.

8.2 Nguyên lý hoạt động và việc thực hành trong simulink

Khởi động simulink

Để khởi động simulink ta có thể thực hiện theo hai cách: Trên màn hình ta dùng chuột nhấp kép vào biểu tượng simulink

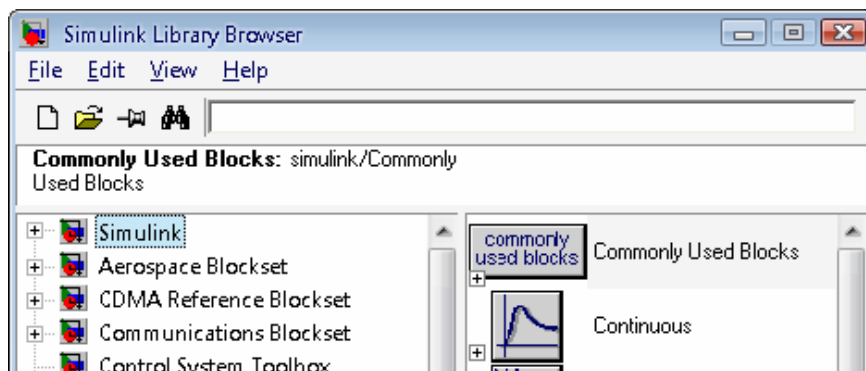


Hình 8-2. Khởi động simulink

Hoặc trong cửa sổ lệnh của Matlab, từ bàn phím ta gõ lệnh simulink vào sau dấu nhắc

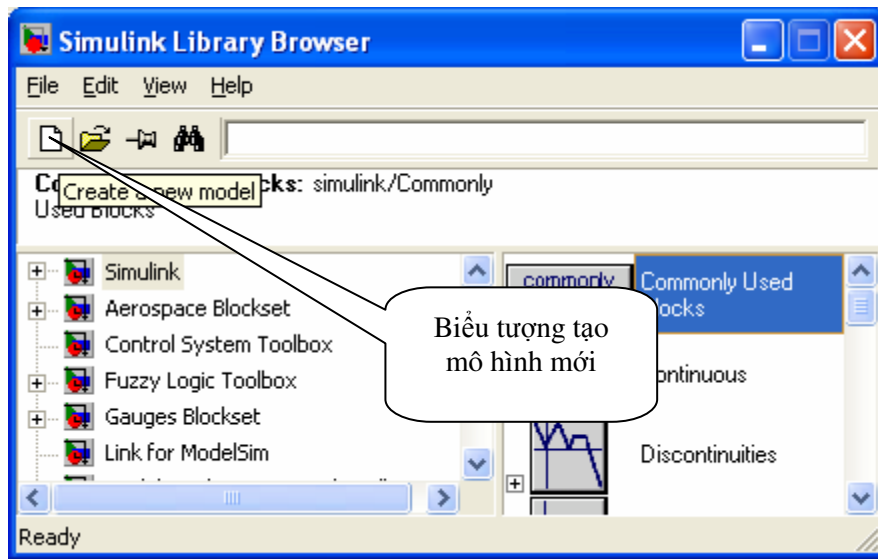
```
>> simulink
```

Khi lệnh simulink được thực hiện, cửa sổ thư viện simulink sẽ hiện ra như sau



Hình 8-3. Cửa sổ với các thư viện trong simulink

Để tạo một mô hình mới, để ấn mới ta sẽ nhấn chuột vào biểu tượng tờ giấy trắng.













Hình 8-4. Khởi tạo một mô hình mới trong simulink















Hoặc từ menu ta chọn File --> New --> Model Ctrl +N.


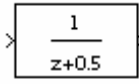

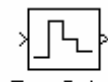
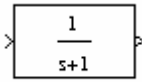

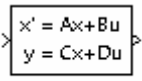
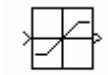
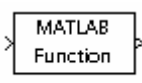
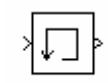
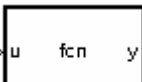
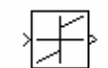
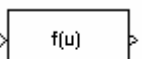

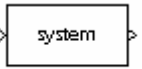
Sau đó ta sẽ sử dụng chuột để kéo các khối trong thư viện vào trong mô hình mới vừa mở, kết nối các khối này để tạo ra một mô hình mô tả hệ.


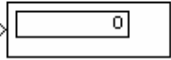




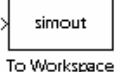

Các khối trong simulink

Các phần tử trong source (nguồn)			
	Constant	Cấp một hằng số, hay một véc tơ hằng	
	Ramp	Cấp một tín hiệu theo đường xiên (đường dốc)	
	Pulse Generator	Cấp một tín hiệu dạng xung	
	Step	Cấp một tín hiệu dạng bước nhảy	
	Sine Wave	Cấp một tín hiệu hình sin	
	Repeating Sequence Stair	Cấp một tín hiệu bậc thang	

 Clock	Đồng hồ cấp thời gian	 Repeating Sequence	Cấp một tín hiệu răng cưa
 Random Number	Cấp một số ngẫu nhiên	 Band-Limited White Noise	Cấp một tín hiệu nhiễu

Các khối hàm toán học, Math Operations			
 Add	Tính tổng hai tín hiệu	 Tín hiệu	Tính tổng hai tín hiệu
 Product	Nhân hai tín hiệu	 Subtract	Tính hiệu hai tín hiệu
 Abs	Cho giá trị tuyệt đối của tín hiệu	 Divide	Nhân/chia hai tín hiệu
 Dot Product	Tính tích vô hướng hai vectơ	 Gain	Nhân tín hiệu với một số (khuếch đại tín hiệu)
 Sum of Elements	Tính tổng các phần tử	 Sign	Hàm dấu (cho dấu của biến x)
 Product of Elements	Tính tích các phần tử	 Sine Wave Function	Hàm sin
 Math Function	Một hàm toán học exp, sin, cos,...	 Trigonometric Function	Hàm lượng giác

Các khối hàm toán học liên tục và không liên tục			
 Derivative	đạo hàm một tín hiệu theo thời gian	 Discrete Transfer Fcn	khối hàm truyền bậc nhất hệ rời rạc
 Integrator	toán tử tích phân	 Zero-Order Hold	khối giữ tín hiệu trích mẫu bậc 0
 Transfer Fcn	khối hàm truyền bậc nhất hệ liên tục	 First-Order Hold	khối giữ tín hiệu trích mẫu bậc 1
 State-Space	khối mô tả phương trình trạng thái hệ tuyến tính	 Saturation	Giữ tín hiệu bão hòa
 MATLAB Fcn	tạo một hàm định nghĩa bởi người sử dụng	 Memory	Ghi nhớ tín hiệu của bước trước
 Embedded MATLAB Function	tạo một hàm nhúng định nghĩa bởi người sử dụng	 Coulomb & Viscous Friction	Khối mô tả ma sát Coulomb và ma sát nhớt
 Fcn	$f(u)$ là một hàm nào đó của Matlab	 Transport Delay	Khối làm trễ tín hiệu
 S-Function			

Các khối kết trong sink	
 Scope	Vẽ ra đồ thị tín hiệu theo thời gian
 Display	Hiển thị giá trị của tín hiệu
 Out1	Một đầu tín hiệu ra, sử dụng cho hệ con
 Terminator	Sử dụng để kết thúc các tín hiệu ra, (tránh để chương trình không cảnh báo cổng tín hiệu không được
 XY Graph	Cho phép biểu diễn hai tín hiệu dạng đồ thị, y(x)
 To File	Để ghi lại tín hiệu vào một file.mat
 To Workspace	Để chuyển tín hiệu vào workspace
 Stop Simulation	Sử dụng để dừng chương trình mô phỏng (khi tín hiệu bằng 0, chương trình sẽ dừng lại.

8.3 Một số ví dụ đơn giản

Một trong những vấn đề hay nhầm lẫn đối với người mới sử dụng simulink là làm thế nào để mô hình hóa các phương trình. Dưới đây sẽ trình bày một số thí dụ nhằm minh họa việc mô hình hóa khối các phương trình toán học.

a) Mô hình hóa một phương trình bằng sơ đồ khối

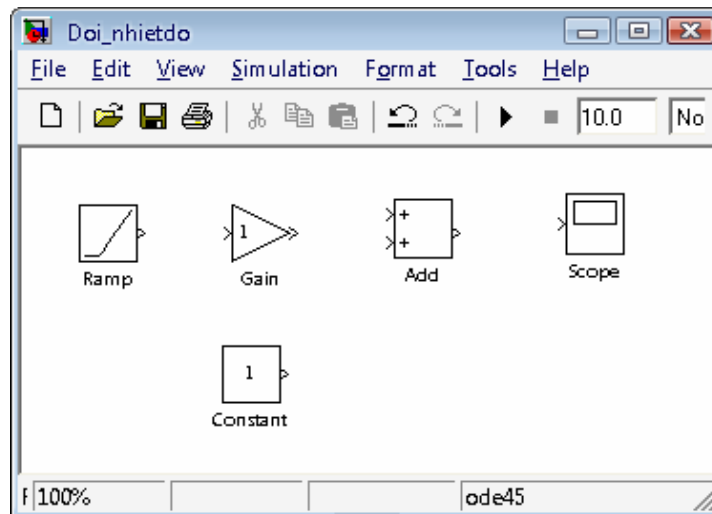
Ví dụ ta cần chuyển đổi từ thang nhiệt độ C sang thang nhiệt độ Fahrenheit, phương trình mô tả quan hệ nhiệt độ giữa hai thang đo này như sau

$$T_F = \frac{9}{5} T_C + 32$$

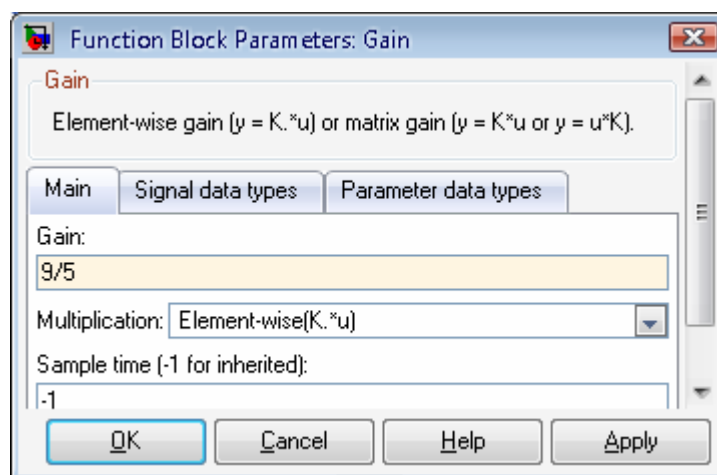
Trước hết cần chú ý đến những khối để xây dựng mô hình

Các khối cần được kéo vào model window	Vị trí các khối trong thư viện Simulink
Ramp	Sources
Constant	Sources
Gain	Math Operation
Sum	Math Operation
Scope	Sinks

Sau đó, ta đưa các khối trên vào trong một cửa sổ mô hình (tạo ra từ Menu File --> New model).



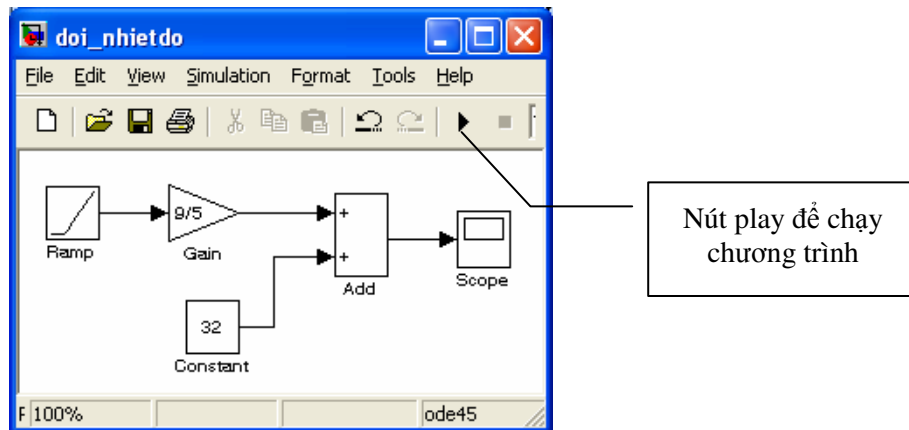
Hình 8-5. Các khối sử dụng trong mô hình



Hình 8-6. Gán giá trị thông số trong khối Gain

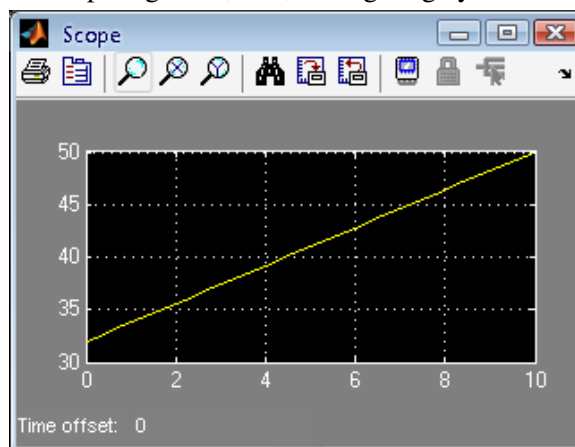
Ở đây có thông số trong hai khối Constant và Gain cần thay đổi. Việc này được thực hiện bằng cách nhấn kép chuột vào khối tương ứng, thay đổi giá trị sau đó nhấn chuột vào nút Close để đặt các giá trị mới và đóng cửa sổ hội thoại lại. Chẳng hạn khi thay đổi giá trị trong Gain ta có cửa sổ hội thoại như trên hình 8-6.

Sau khi sử dụng chuột nối các khối với nhau, cho ta một mô hình khối mô tả phương trình trên như sau



Hình 8-7. Sơ đồ khối chuyển đổi thang nhiệt độ

Khối Ramp cung cấp nhiệt độ trong thang độ C. Mô khối này và đặt cho tham số Initial output giá trị 0. Khối Gain nhân giá trị nhiệt độ của khối Ramp với hằng số 9/5. Khối Sum (Add) cộng giá trị 32 với kết quả sau khối Gain và cho ra giá trị nhiệt độ theo thang Fahrenheit. Mô khối Scope để quan sát kết quả. Để chạy chương trình ta chọn Start từ Simulation menu hoặc là nhấn chuột vào nút play ► trên thanh công cụ. Mô phỏng sẽ thực hiện trong 10 giây.



Hình 8-8.

b) Mô phỏng một quá trình động học

Trong ví dụ này ta sẽ sử dụng Simulink để mô tả một phương trình động học sau:

$$x(t) = A \sin(\omega t + \varphi),$$

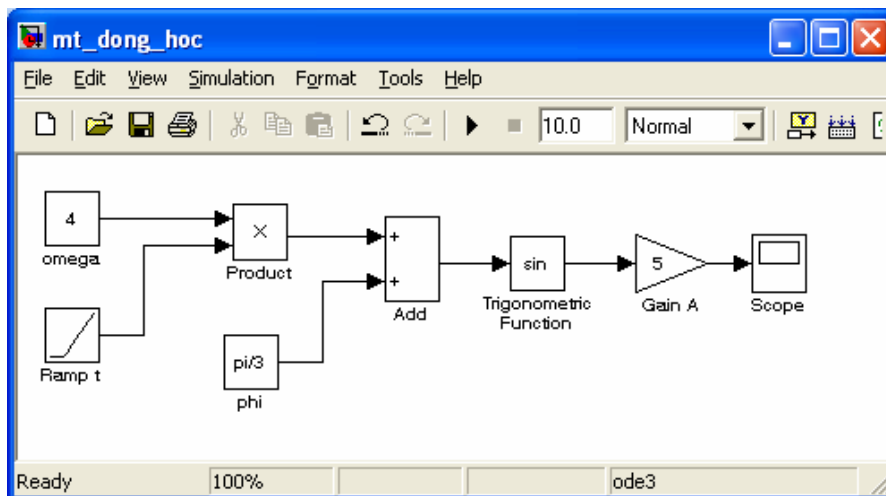
trong đó dịch chuyển x là hàm của thời gian t , tần số góc ω và góc pha ban đầu φ . Trong ví dụ này các thông số được nhận các giá trị: $A = 2$, $\omega = 4$, $\varphi = \pi / 3$.

Các bước thực hiện xây dựng mô hình như sau

1. Từ thư viện Simulink sử dụng chuột kéo các khối sau đây vào trong Model Window


Các khối cần được kéo vào model window	Vị trí các khối trong thư viện Simulink
Constant	Sources
Ramp	Sources
Gain	Math Operation
Sum	Math Operation
Product	Math Operation
Trigonometry Function	Math Operation
Mux	Signal Routing
Scope	Sinks


2. Nối các khối như trên sơ đồ dưới đây



Hình 8-9. Sơ đồ mô tả phương trình động học $x(t)$

Nhấn đúp chuột vào các khối ta có thể thay đổi giá trị các thông số.

3. Kiểm tra lại xem các khối đã được nối chính xác chưa, sau đó chạy chương trình bằng cách nhấn chuột vào biểu tượng play  hoặc (CTRL+T). Để xem kết quả mô

phòng hãy nhấn chuột vào khối Scope. Để hiển thị toàn bộ đồ thị, hãy nhấn chuột vào biểu tượng ống nhòm .

Cần chú ý rằng trong khối Trigonometric Function ta có thể chọn được nhiều hàm lượng giác khác nhau.

Ta có thể nhận được một kết quả tương tự bằng các dòng lệnh tại dấu nhắc trong cửa sổ lệnh như sau:

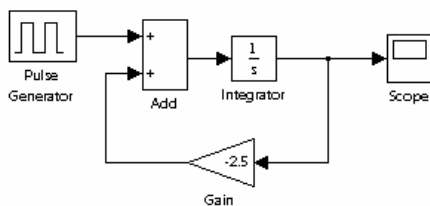
```
>> t=(0:.01:10); A=2; phi=pi/3; omega=4;
>> x=A*sin(omega*t+phi);
>> plot(t,x); grid
```

c) Mô hình hóa một hệ động lực liên tục đơn giản

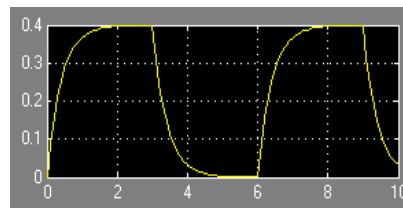
Xét phương trình vi phân mô tả một hệ động lực liên tục như sau

$$\dot{x}(t) = -2.5x(t) + u(t)$$

trong đó $u(t)$ là kích động dạng xung, có biên độ là 1 và chu kỳ là 6 s, độ rộng xung là 50%. Khối tích phân Integrator thực hiện việc tích phân từ $\dot{x}(t)$ để tạo ra $x(t)$. Các khối khác sử dụng trong mô hình này là Gain và Sum. Để tạo ra tín hiệu xung, ta kéo khối Pulse Generator từ thư viện Source, kích đúp chuột vào khối và thay đổi một số tham số trong đó. Trong mô hình này, để đổi hướng của khối Gain, ta chọn khối nhấn chuột phải và chọn **Flip Block** trong menu **Format**. Sau khi kết nối các khối, chạy chương trình ta thu được kết quả hiển thị trong Scope như trên hình 8-10.



a) sơ đồ simulink



b) kết quả mô phỏng trong 10 giây

Hình 8-10.

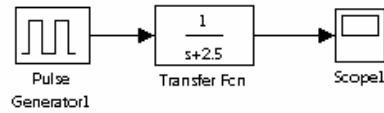
Nếu áp dụng biến đổi Laplace đối với hệ động lực tuyến tính trên, ta có thể sử dụng khối Transfer Fcn để mô tả hệ. Theo cách này ta biến đổi phương trình vi phân trên như sau

$$\dot{x}(t) = -2.5x(t) + u(t) \Rightarrow sX(s) = -2.5X(s) + U(s)$$

Giải tìm $X(s)$ ta được

$$X(s) = \frac{1}{s + 2.5} U(s)$$

Sử dụng chuột kéo khối Transfer Fcn từ thư viện ra, thay đổi các tham số của tử số và mẫu số cho phù hợp. Trong trường hợp này tử số là 1 và mẫu số là $s + 2.5$. Các hệ số của tử và mẫu số được xếp thành dưới dạng các vectơ lần lượt giảm theo số mũ của s . Trong trường hợp này, tử số là [1] (hoặc chỉ viết 1) và mẫu số là [1 2.5]. Mô hình bây giờ trở nên đơn giản, hình 8-11. Nhấn nút play sẽ cho ta kết quả mô phỏng phù hợp với mô hình trên.



Hình 8-11.

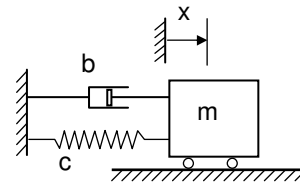
Lưu ý trong quá trình xây dựng mô hình không nên quên ghi lại với một cái tên thích hợp để dễ nhớ. Các dữ liệu ghi lại sẽ nhận phần mở rộng của tên tệp là mdl.

d) Mô tả hệ dao động một bậc tự do

Khảo sát một hệ dao động gồm khối lượng - lò xo - giảm chấn như trên hình bên. Mô hình toán học mô tả hệ

$$m\ddot{x}(t) + b\dot{x}(t) + cx(t) = f(t)$$

trong đó m - khối lượng hệ (kg), b - hệ số cản nhớt (N.s/m), c - hệ số cứng lò xo (N/m), $f(t)$ - lực kích động (N).



Hình 8-12.

Sau đây sẽ minh họa việc xây dựng mô hình hệ trong simulink để mô phỏng hệ với lực kích động là một hàm bước nhảy đơn vị.


Bước 1.

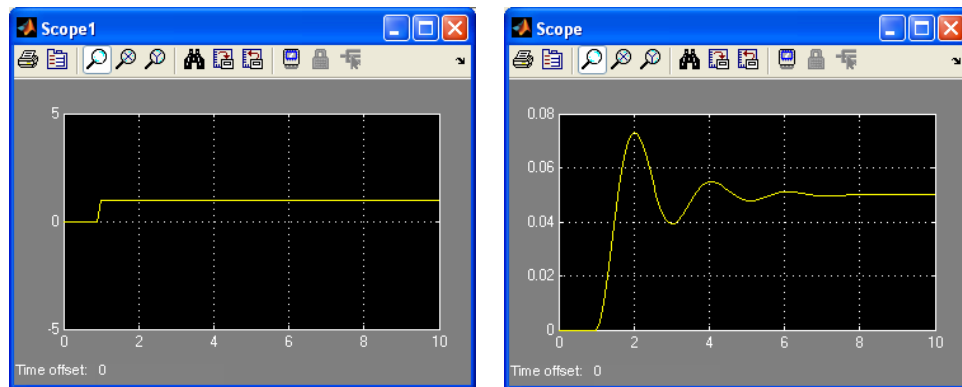
Trong simulink, ta mở một cửa sổ mô hình mới (CTRL+N) sau đó sử dụng chuột kéo các khối sau đây từ thư viện simulink vào cửa sổ mô hình.

Các khối cần được kéo vào model window	Vị trí các khối trong thư viện Simulink
Step	Sources
Constant	Sources
Sum	Math Operation
Product	Math Operation
Divide	Math Operation
Integrator	Continuous
To Workspace	Sinks
Scope	Sinks

Người sử dụng có thể thay đổi các giá trị khác nhau và quan sát ứng xử của hệ. Sử dụng tổ hợp phím (CTRL+E) khối hội thoại sau đây xuất hiện (hình 8-14). Trong hộp hội thoại này ta có thể chọn phương pháp tích phân, thời gian đầu, thời gian cuối của quá trình tích phân, bước tích phân,... Chẳng hạn, ở trên đã sử dụng phương pháp Runge-Kutta cấp 4 (ode4), bước thời gian là $\Delta t = 0.1 \text{ s}$. Nhấn OK để đóng hộp thoại.

Bước 4.

Chạy chương trình mô phỏng (CTRL+T) hoặc nhấn chuột vào nút play . Để xem ứng xử hệ ta nhấn kép chuột vào Scope. Với các số liệu trên ta có kết quả như trên đồ thị.



Kích động bước nhảy, hàm $f(t)$

Đáp ứng của hệ $x(t)$

Hình 8-15

Quan sát đáp ứng hệ, ta thấy trong giây đầu tiên khi chưa có lực kích động khối lượng m không chuyển động, khi có lực tác dụng khối lượng m chuyển động (thực hiện dao động), và sau khoảng 8 giây khối lượng dừng lại ở vị trí mới.

e) Mô phỏng số tay máy một bậc tự do

Xét tay máy một bậc tự do được dẫn động bởi động cơ một chiều (hình 8-16). Phương trình vi phân chuyển động của tay máy sau khi đã đơn giản hóa như sau

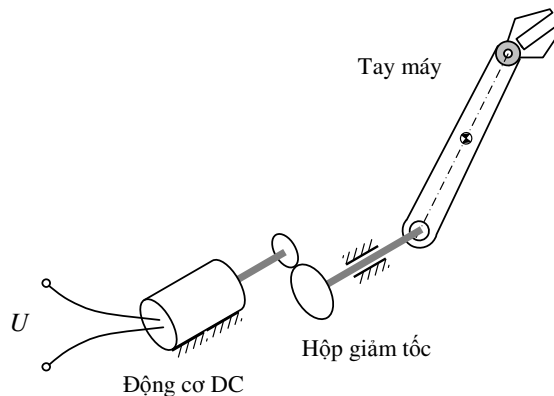
$$(J_o + J_m r^2) \ddot{q} = (K_m r / R_a) U - (K_m K_e r^2 / R_a + b_1 r^2 + b_2) \dot{q} - mgl \cos(\varphi)$$

$$\dot{\varphi} = q$$

hay $(J_o + J_m r^2) \ddot{\varphi} + (K_m K_e r^2 / R_a + b_1 r^2 + b_2) \dot{\varphi} + mgl \cos(\varphi) = (K_m r / R_a) U$.

Các thông số của hệ sử dụng trong phương trình trên gồm:

- Tay máy là vật rắn quay quanh trục ngang cố định qua O, khối lượng m , khối tâm C, $OC = l$. Mômen quán tính đối với trục quay O, J_O . Hệ số cản tại ổ trục b_2 .
- Động cơ điện một chiều: điện trở R_a , cuộn cảm có hệ số tự cảm $L \approx 0$, hằng số phản sức điện động của rôto, K_e ; hằng số mômen K_m . Ở đây θ là góc quay của trụ động cơ, $\dot{\theta}$ là vận tốc góc trên trục động cơ. Mômen quán tính khối của rôto J_m . Hệ số cản nhớt tại ổ trục b_1 (ma sát nhớt tại ổ trục).
- Hộp giảm tốc có tỷ số truyền $r = \omega_1 / \omega_2 = r_2 / r_1$, khối lượng các bánh răng không đáng kể.



Hình 8-16. Mô hình tay máy một bậc tự do

Sau đây sẽ trình bày việc sử dụng simulink để mô tả phương trình vi phân và mô phỏng. Trong mô phỏng này bỏ qua hệ số cản b_2 (cản trên tay máy) và hệ số cản b_1 được viết là b , điện áp U chuyển thành V . Mômen quán tính J_O viết thành J . Như thế phương trình vi phân chuyển động của tay máy trở thành:

$$(J_O + J_m r^2) \ddot{\varphi} + (K_m K_e r^2 / R_a + b r^2) \dot{\varphi} + mgl \cos(\varphi) = (K_m r / R_a) V.$$

Bước 1.

Trong simulink, ta mở một cửa sổ mô hình mới (CTRL+N) sau đó sử dụng chuột kéo các khối sau đây từ thư viện simulink vào cửa sổ mô hình.

Các khối cần được ghép vào model window	Vị trí các khối trong thư viện Simulink
Constant	Sources
Sum	Math Operation
Product	Math Operation
Divide	Math Operation

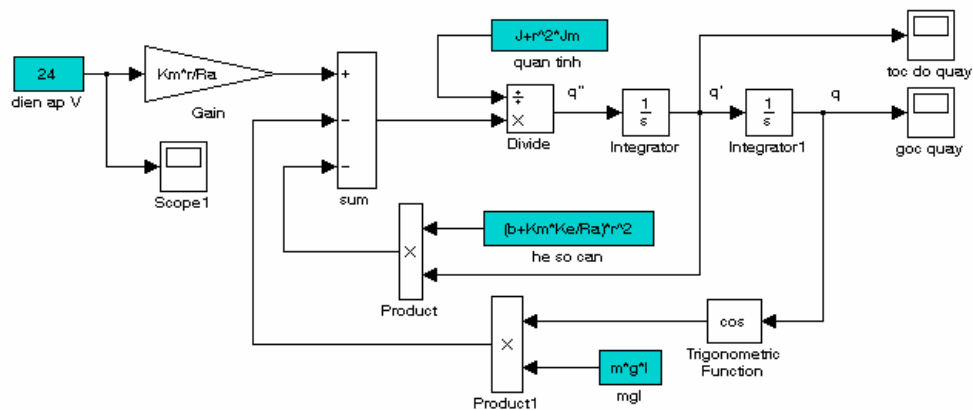
Gain	Math Operation
Trigonometric functions	Math Operation
Integrator	Continuous
Scope	Sinks

Bước 2.

Viết lại phương trình vi phân chuyển động trên dưới dạng sau

$$\ddot{q} = \frac{1}{(J + J_m r^2)} \cdot \left\{ (K_m r / R_a) V - [(b + K_m K_e / R_a) r^2] \dot{q} - mgl \cos(q) \right\}$$

Dựa vào phương trình này ta nối các khối như trong sơ đồ. Ở đây có một số khối cần được xoay 180° để thực hiện việc này, hãy chọn khối sau đó nhấn chuột phải rồi chuyển đến Format và chọn Flip Block hoặc Rotate Block.



Hình 8-17. Sơ đồ mô tả tay máy một bậc tự do (he_co_dien_1doff.mdl)


Bước 3.

Để gán các giá trị cho các thông số ta có thể soạn một m-file như sau

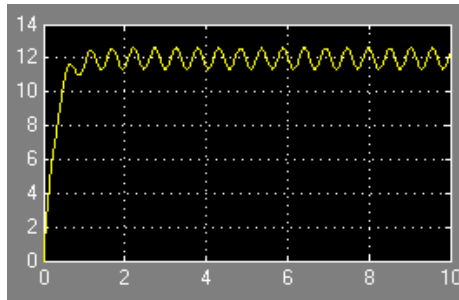
```
% file khai bao so lieu cho cac bien
m=2;    J=1;    l=0.4;
g=9.81; r=2;
Ke=1;   Km=1;   Ra=1;
Jm=0.001;    b=0.001;
% save voi ten file *.m
```

Ấn F5 chạy m-file này để Matlab nhận các giá trị các thông số sử dụng trong simulink.

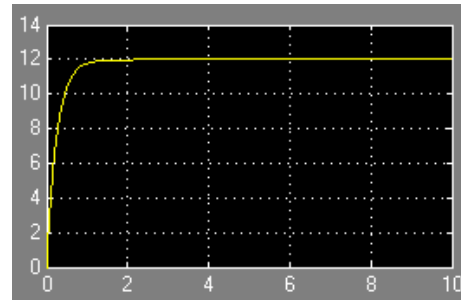
Bước 4.

Chạy chương trình mô phỏng (CTRL+T) hoặc nhấn chuột vào nút play . Để xem ứng xử hệ ta nhấn kép chuột vào Scope.

Với đầu vào điện áp $U = 24 \text{ V}$, ta thu được kết quả tốc độ quay của tay máy như đồ thị.



Trọng tâm không nằm trên trục quay,
 $L = 0.4$



Trọng tâm nằm trên trục quay, $L = 0$

Hình 8-18. Đáp ứng của hệ (tốc độ quay của tay máy, \dot{q})

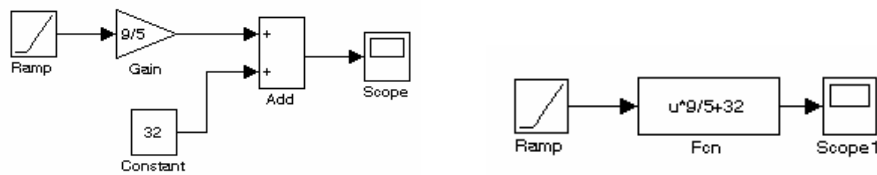
Quan sát đáp ứng của hệ, ta thấy tốc độ quay của tay máy tăng từ giá trị ban đầu 0 sau đó bình ổn dao động quanh giá trị 12 rad/s. Trọng tâm tay máy không nằm trên trục quay cố định làm nguyên nhân tốc độ quay dao động. Khi trọng tâm tay máy nằm trên trục quay ta thấy tốc độ quay đạt được giá trị hằng số sau khoảng 2 giây. Người sử dụng có thể thay đổi các giá trị khác nhau và quan sát ứng xử của hệ.

8.4 Đơn giản sơ đồ simulink

Đối với các hệ động lực phức tạp chẳng hạn như con lắc kép hay một tay máy hai bậc tự do. Các phương trình vi phân mô tả hệ như thế rất phức, nếu thực hiện tất cả các phép tính bằng các khối của simulink thì sơ đồ mô tả rất phức tạp và rất có thể dẫn tới các lỗi khó kiểm soát. Trong các trường hợp như vậy, ta có thể sử dụng các khối Fcn, subsystem để làm đơn giản mô hình bằng các mô đun. Một cách khác nữa là kết hợp giữa simulink với các m-file. Dưới đây là ví dụ mô tả công việc này.

Đơn giản sơ đồ simulink bằng khối Fcn

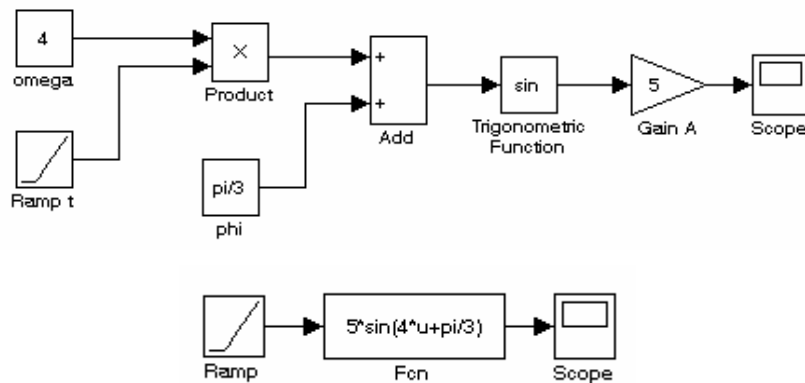
Khối **Fcn** trong thư viện **User-defined Functions** cho phép thực hiện các phép tính đại số thông thường. Do đó thay vì sử dụng nhiều khối để tính toán một biểu thức $y = f(x)$, ta chỉ cần sử dụng một khối duy nhất Fcn, mà trong đó viết biểu thức của hàm $f(x)$. Ví dụ thay vì sử dụng sơ đồ như trên hình 8-7. cho bài toán chuyển đổi thang nhiệt độ, ta chỉ cần sử dụng một sơ đồ đơn giản như sau



Hình 8-19. Ví dụ về khối Fcn

Một thí dụ khác là mô tả động học phương trình dao động sau:

$$x = 5 \sin(4t + \pi / 3)$$



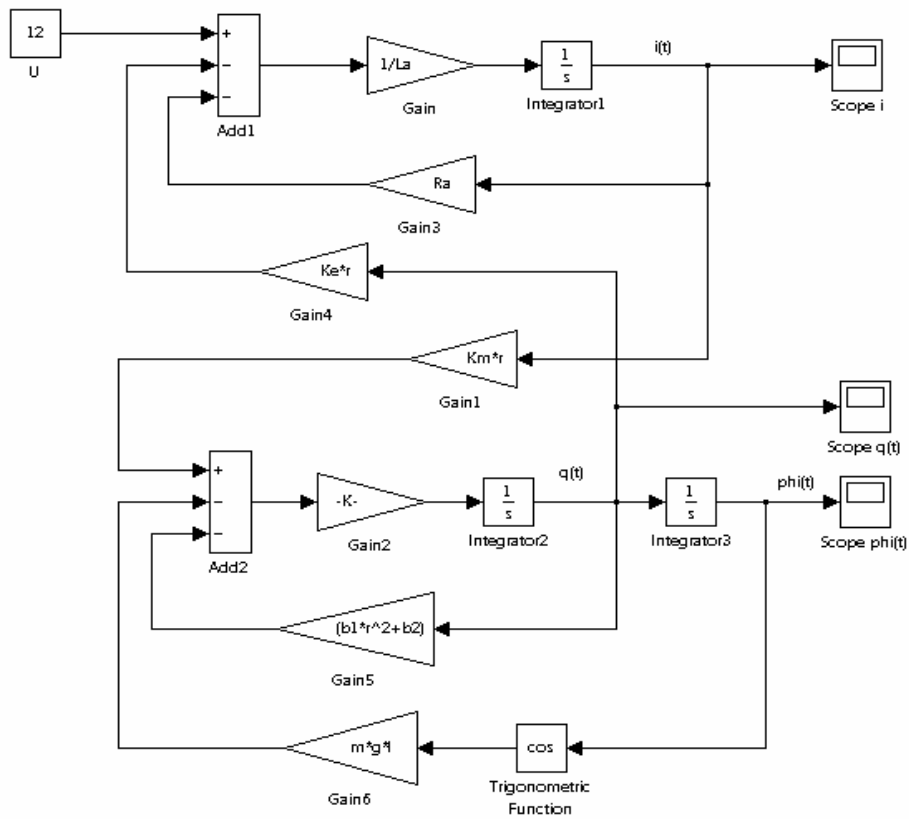
Hình 8-20. Ví dụ về khối Fcn

Đơn giản sơ đồ simulink bằng khối subsystem

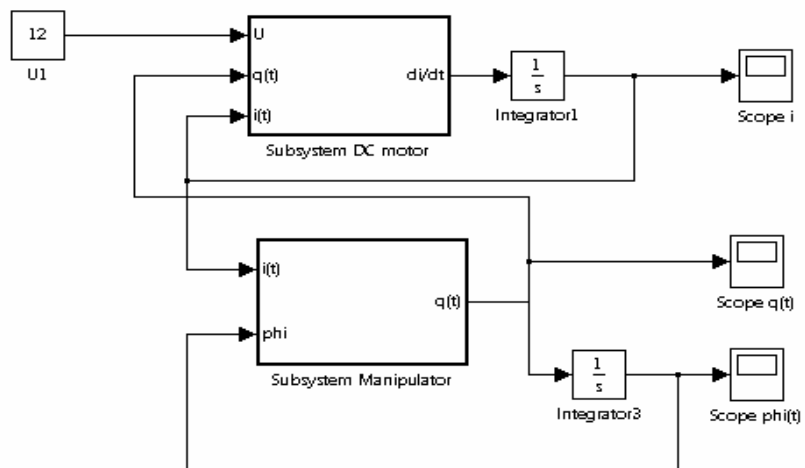
Để đơn giản sơ đồ simulink mô tả hệ phức tạp, trước hết ta chia hệ ban đầu thành các hệ con với các đầu vào ra rõ ràng. Mỗi mô đun này sẽ được mô tả bằng một subsystem trong simulink. Để thấy được sự tiện lợi của khối subsystem, ta sẽ xét ví dụ hệ cơ điện mô tả bởi phương trình vi phân:

$$\begin{aligned} \frac{di}{dt} &= \frac{1}{L_a} (U - R_a i - K_e r q) \\ \frac{dq}{dt} &= \frac{1}{(J_o + J_m r^2)} [K_m r i - (b_1 r^2 + b_2) q - mgl \cos(\varphi)] \\ \frac{d\varphi}{dt} &= q \end{aligned}$$

Với hệ ba phương trình vi phân trên, sơ đồ simulink được thiết kế như hình 8-21 dưới đây. Sơ đồ mô tả này tương đối phức tạp với nhiều khối và các đường nối có phần chồng chéo nhau.

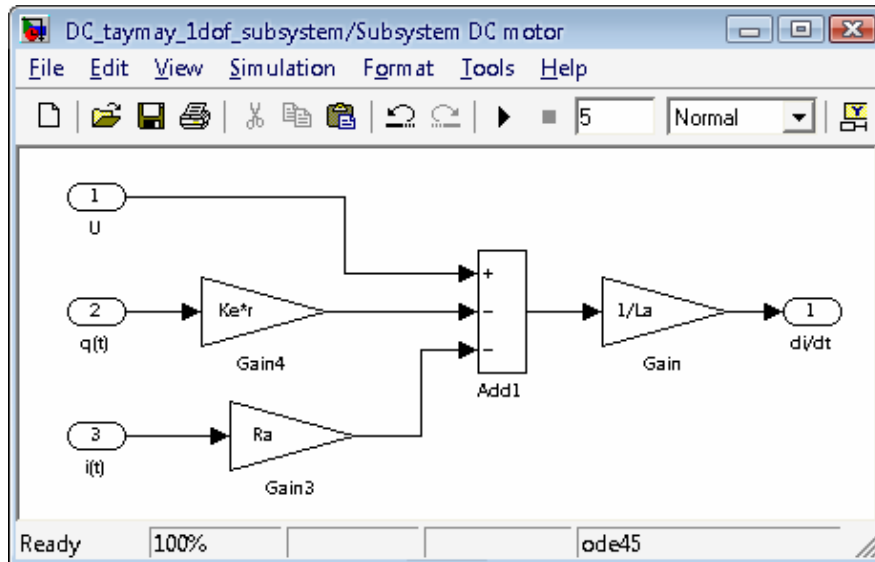


Hình 8-21. DC_taymay_1dof.mdl

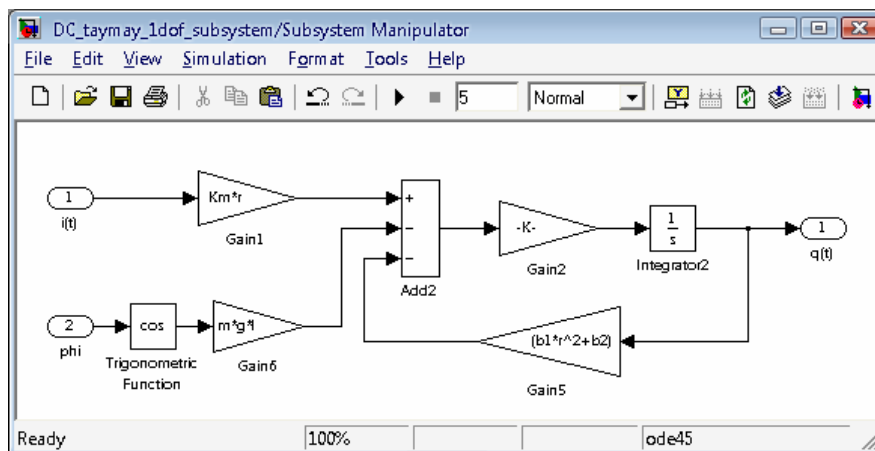


Hình 8-22. DC_taymay_1dof_subsystem.mdl

Nhờ có kỹ thuật hệ con (subsystem) ta xây dựng được một sơ đồ đơn giản hơn, với hai hệ con. Một hệ mô tả động lực học động cơ và hệ còn lại mô tả động lực học tay máy. Tất nhiên trong mỗi hệ con ta phải mô tả chi tiết các phần tử của hệ (hình 8-22,23,24).



Hình 8-23.

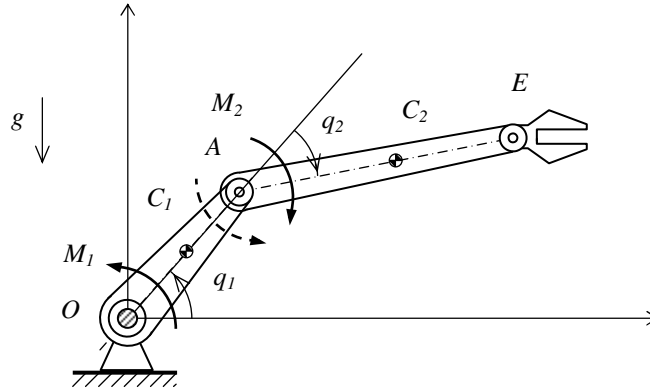


Hình 8-24.

Kết hợp simulink và script file (m-file)

Xét hệ tay máy hai bậc tự do chuyển động trong mặt phẳng thẳng đứng. Khâu 1 chiều dài $OA = l_1$, khối tâm C_1 , $OC_1 = a_1$, khối lượng m_1 , mômen quán tính đối với trục qua khối tâm là J_1 . Khâu 2 chiều dài $AE = l_2$, khối tâm C_2 , $AC_2 = a_2$, khối

lượng m_2 , mômen quán tính đối với trục khối tâm là J_2 . Động cơ 1 gắn liền với giá cố định tạo ra mômen M_1 tác dụng lên khâu 1, động cơ 2 gắn liền với khâu 1 tạo ra mômen M_2 tác dụng lên khâu 1.



Hình 8-25. Tay máy hai bậc tự do

Trước hết sử dụng phương trình Lagrange loại 2 để thiết lập phương trình vi phân chuyển động cho hệ. Cơ hệ gồm hai vật: khâu OA quay quanh trục ngang cố định, và khâu AB chuyển động song phẳng. Chọn các tọa độ suy rộng: $\mathbf{q}^T = [q_1, q_2]$, q_1 là góc quay của khâu 1, thanh OA, q_2 là góc quay của khâu 2, thanh AE, so với khâu 1.

Thiết lập PTVP CĐ bằng phương trình Lagrange loại 2

$$\frac{d}{dt} \frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} = - \frac{\partial \Pi}{\partial q_i} + Q_i^* \quad (i = 1, 2)$$

1) Tính động năng của cơ hệ

$$T = \frac{1}{2} J_{O_1} \omega_1^2 + \frac{1}{2} m v_{C_2}^2 + \frac{1}{2} J_{C_2} \omega_2^2 \quad (1)$$

$$J_{O_1} = J_{C_1} + m_1 a_1^2, \quad \omega_1^2 = \dot{q}_1^2, \quad \omega_2^2 = (\dot{q}_1 + \dot{q}_2)^2,$$

$$x_{C_2} = l_1 \cos q_1 + a_2 \cos(q_1 + q_2)$$

$$y_{C_2} = l_1 \sin q_1 + a_2 \sin(q_1 + q_2),$$

$$\dot{x}_{C_2} = -l_1 \dot{q}_1 \sin q_1 - a_2 (\dot{q}_1 + \dot{q}_2) \sin(q_1 + q_2)$$

$$\dot{y}_{C_2} = l_1 \dot{q}_1 \cos q_1 + a_2 (\dot{q}_1 + \dot{q}_2) \cos(q_1 + q_2),$$

Từ đó có :

$$v_{C2}^2 = \dot{x}_{C2}^2 + \dot{y}_{C2}^2 = l_1^2 \dot{q}_1^2 + a_2^2 (\dot{q}_1 + \dot{q}_2)^2 + 2l_1 a_2 \dot{q}_1 (\dot{q}_1 + \dot{q}_2) \cos q_2$$

Thay các kết quả trên vào (1) ta được :

$$T = \frac{1}{2} (J_{C1} + m_1 a_1^2 + J_{C2} + m_2 a_2^2 + m_2 l_1^2) \dot{q}_1^2 + (J_{C2} + m_2 a_2^2 + m_2 l_1 a_2 \cos q_2) \dot{q}_1 \dot{q}_2 + \frac{1}{2} (J_{C2} + m_2 a_2^2) \dot{q}_2^2 \quad (2)$$

2) Hàm thế năng

$$\Pi = m_1 g a_1 \sin q_1 + m_2 g [l_1 \sin q_1 + a_2 \sin(q_1 + q_2)];$$

3) Tính lực suy rộng Q_1 và Q_2 , ta có :

$$\begin{aligned} Q_1 &= M_1 - m_1 g a_1 \cos q_1 - m_2 g [l_1 \cos q_1 + a_2 \cos(q_1 + q_2)] \\ Q_2 &= M_2 - m_2 g a_2 \cos(q_1 + q_2). \end{aligned} \quad (3)$$

Thay (2), (3) vào phương trình Lagrange loại 2 ta có phương trình vi phân chuyển động của cơ hệ:

$$\begin{aligned} (J_{C1} + m_1 a_1^2 + J_{C2} + m_2 a_2^2 + m_2 l_1^2 + 2m_2 l_1 a_2 \cos q_2) \ddot{q}_1 + \\ (J_{C2} + m_2 a_2^2 + m_2 l_1 a_2 \cos q_2) \ddot{q}_2 - m_2 l_1 a_2 \dot{q}_2 (2\dot{q}_1 + \dot{q}_2) \sin q_2 \\ + m_1 g a_1 \cos q_1 + m_2 g [l_1 \cos q_1 + a_2 \cos(q_1 + q_2)] = M_1 \\ \\ (J_{C2} + m_2 a_2^2 + m_2 l_1 a_2 \cos q_2) \ddot{q}_1 + (J_{C2} + m_2 a_2^2) \ddot{q}_2 \\ + m_2 l_1 a_2 \dot{q}_1^2 \sin q_2 + m_2 g a_2 \cos(q_1 + q_2) = M_2. \end{aligned}$$

Viết phương trình vi phân chuyển động trên dạng ma trận như sau:

$$\mathbf{M}(\mathbf{q}) \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{u}$$

với

- ma trận khối lượng:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} J_{C1} + m_1 a_1^2 + J_2 + m_2 (l_1^2 + a_2^2 + 2l_1 a_2 \cos q_2) & J_{C2} + m_2 (a_2^2 + l_1 a_2 \cos q_2) \\ J_{C2} + m_2 (a_2^2 + l_1 a_2 \cos q_2) & J_{C2} + m_2 a_2^2 \end{bmatrix}$$

- ma trận chứa các thành phần côriôlis và lực ly tâm:

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -m_2 l_1 a_2 \dot{q}_2 \sin q_2 & -m_2 l_1 a_2 (\dot{q}_1 + \dot{q}_2) \sin q_2 \\ m_2 l_1 a_2 \dot{q}_1 \sin q_2 & 0 \end{bmatrix}$$

- véc tơ chứa lực do trọng lượng:

$$\mathbf{g}(\mathbf{q}) = \begin{bmatrix} g(m_1 a_1 + m_2 l_1) \cos q_1 + g m_2 a_2 \cos(q_1 + q_2) \\ g m_2 a_2 \cos(q_1 + q_2) \end{bmatrix}$$

- véctơ lực/mômen do các động cơ điều khiển:

$$\mathbf{u} = \begin{bmatrix} M_1 & M_2 \end{bmatrix}^T.$$

Trong trường hợp này, nếu hoàn toàn thực hiện các phép tính bằng các khối simulink, thì ta phải sử dụng rất nhiều khối, kết nối các khối này sẽ tạo nên một hệ tương đối phức tạp. Để tránh điều này, ta sẽ thực hiện kết hợp simulink và m-file, bằng cách mô tả các phương trình vi phân chuyển động trong m-file.

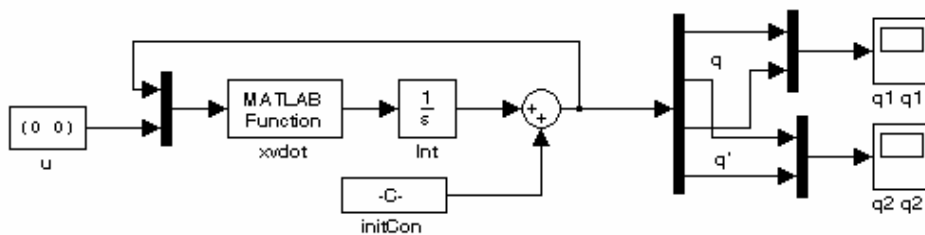
Sau đây sẽ minh họa việc sử dụng kết hợp simulink và m-file để xây dựng mô hình mô phỏng hệ.

Bước 1.

Trong simulink, ta mở một cửa sổ mô hình mới (CTRL+N) sau đó sử dụng chuột kéo các khối sau đây từ thư viện simulink vào cửa sổ mô hình.

Các khối cần được kết vào model window	Vị trí các khối trong thư viện Simulink
Constant	Sources
Sum	Math Operation
MATLAB Fcn	User-Defined Functions
Integrator	Continuous
Mux	Signal Routing
Demux	Signal Routing
Scope	Sinks

Sử dụng chuột để nối các khối như dưới đây.



Hình 8-26. Sơ đồ simulink cho tay máy hai bậc tự do

Bước 2.

Biến đổi phương trình vi phân cấp hai

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{u}$$

về cấp một như sau

$$\dot{\mathbf{q}} = \mathbf{v}$$

$$\dot{\mathbf{v}} = \mathbf{M}(\mathbf{q})^{-1}[\mathbf{u} - \mathbf{C}(\mathbf{q}, \mathbf{v})\mathbf{v} - \mathbf{g}(\mathbf{q})]$$

Trên cơ sở hệ phương trình vi phân cấp một này, ta soạn một m-file (với tên file **xvdot.m**) như dưới đây.

```
== =====
function ydot=xvdot(in)
% Phương trình vi phân chuyển dạng ma trận:
% M(q)*q'' + C(q,q')q' + g(q) = tau
% M(q) = ma trận khối lượng
% C(q,q')q' = vector chứa lực Coriolis và lực ly tâm
% g(q) = vector chứa lực do trọng trường
% tau = vector lực/momen điều khiển
% các biến sử dụng hay các tọa độ và vận tốc suy rộng
q1=in(1);    q2=in(2); q1_dot=in(3);  q2_dot=in(4);
q_dot=[q1_dot; q2_dot];
% control vector
u1=in(5);    u2=in(6);  tau=[u1; u2];
% Các tham số của hệ
% khâu 1
m1=21.20;    %[kg]
J1=0.21;     %[kg m^2]
l1=0.25;     %[m]
a1=0.15;     %[m]
% khâu 2
m2=15.22;    %[kg]
J2=0.18;     %[kg m^2]
l2=0.45;     %[m]
a2=0.19;     %[m]
% gia tốc trọng trường
g=9.81;      %m/s^2
% ma trận khối lượng hay ma trận quán tính
m11 = J1+J2+m1*a1^2+m2*l1^2+m2*a2^2+2*m2*l1*a2*cos(q2);
m12 = J2+m2*a2^2+m2*l1*a2*cos(q2);
m21 = m12;    m22 = J2+m2*a2^2;
M = [m11, m12; m21, m22];
```



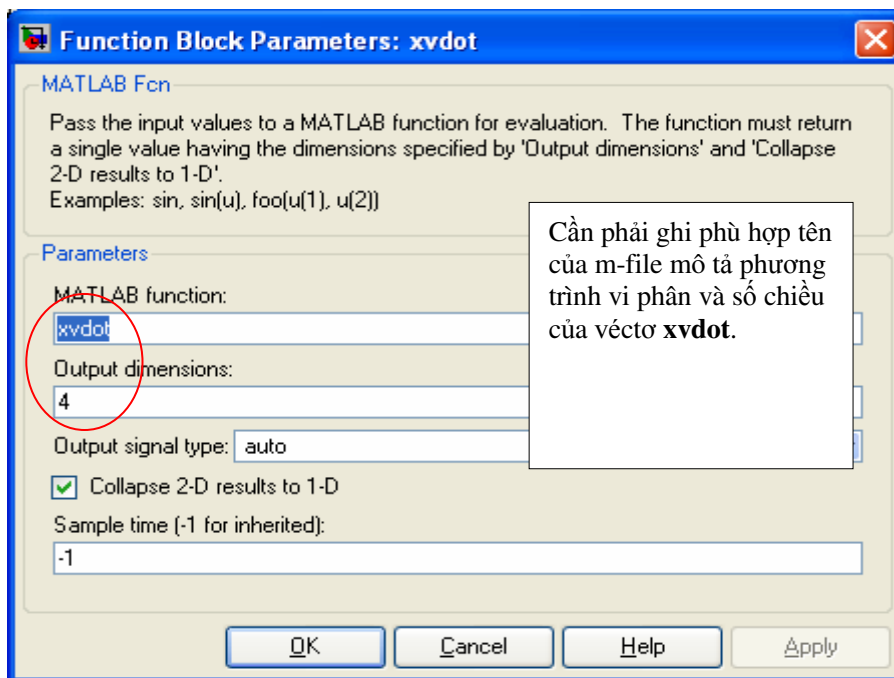
```

% Coriolis forces
c11 = -m2*l1*a2*sin(q2)*q2_dot;
c12 = -m2*l1*a2*sin(q2)*(q2_dot+ q1_dot);
c21 = m2*l1*a2*sin(q2)*q1_dot;          c22 = 0;
C = [c11, c12; c21, c22];
% vector lực do trọng trường
gq=[g*(m1*a1+m2*l1)*cos(q1)+g*m2*a2*cos(q1+q2);
    g*m2*a2*cos(q1+q2)];
% Dưa phương trình vi phân về dạng
% y_dot = f(t,y)
q_dot=[q1_dot; q2_dot];
q_2dot=inv(M)*(tau - C*q_dot - gq);
ydot=[q_dot; q_2dot];
=====

```

Bước 3.


Nháy đúp chuột vào khối (MATLAB Fcn), một hộp hội thoại xuất hiện và ta sẽ thay đổi hai thông số cho phù hợp (lấy đúng tên của m-file vừa soạn trên điền vào ô MATLAB Fcn: **xvdot**, lấy đúng số chiều của vectơ **xvdot** điền vào ô Output dimensions: **4** như hình dưới đây.



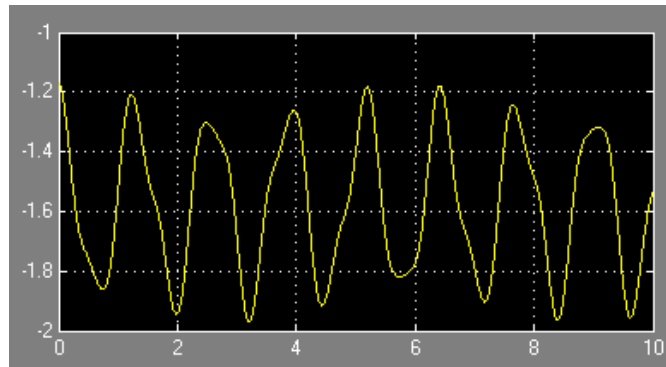
Hình 8-27. Hộp thoại khai báo các thông số của khối MATLAB-Fcn

Sau đó nhấn OK để đóng hộp thoại.

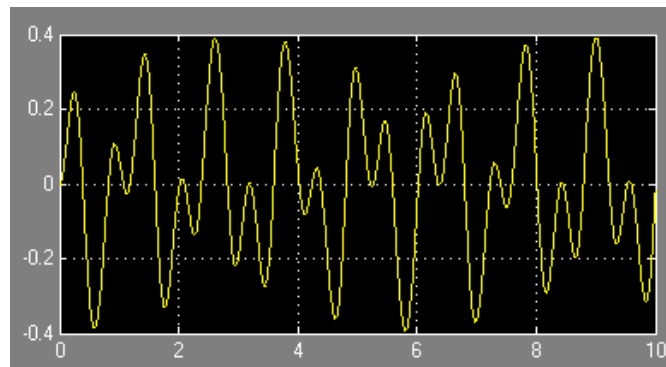
Bước 4.

Chạy chương trình mô phỏng (CTRL+T) hoặc nhấn chuột vào nút play . Để xem ứng xử hệ ta nhấn kép chuột vào Scope.

Với điều kiện đầu $q_1 = -\pi/2 + 0.4 \text{ rad}$, $q_2 = 0$, $\dot{q}_1 = 0$, $\dot{q}_2 = 0$ ($[-\pi/2 + 0.4 \ 0, 0 \ 0]$), và không có các mômen điều khiển kết quả mô phỏng nhận được như trên hình 8-28.



Đồ thị góc khâu 1 theo thời gian

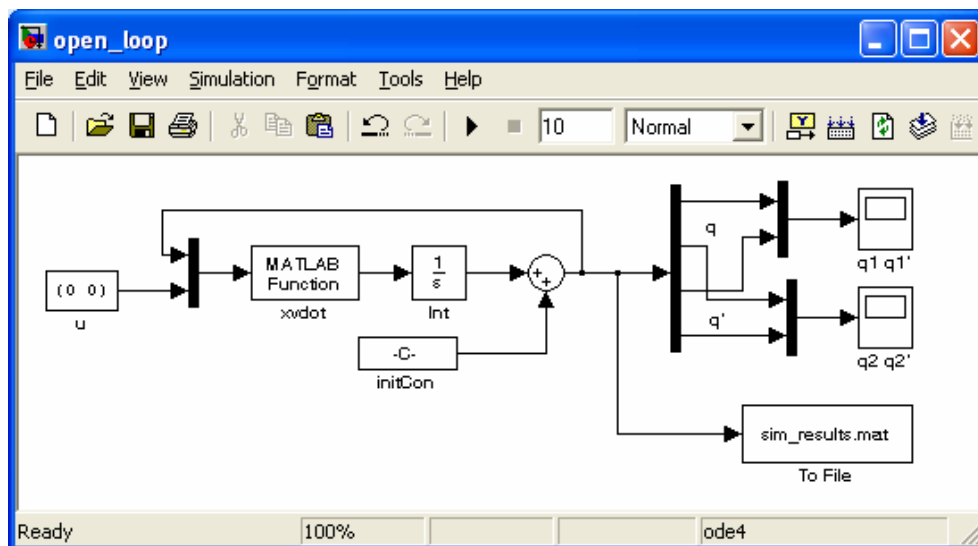


Hình 8-28. Đồ thị góc khâu 2 theo thời gian

Người sử dụng có thể thay đổi các điều kiện đầu trong khối initCon (initial Condition), thay đổi giá trị mômen trong khối đầu vào u chạy chương trình và quan sát kết quả.

8.5 Xử lý kết quả mô phỏng

Khối Scope cho ta kết quả mô phỏng theo thời gian, tuy nhiên nếu trực tiếp sử dụng đồ thị này cho các công việc như viết báo cáo, trình bày trong bài một văn bản thì không được thuận tiện, đặc biệt nếu để in ấn bởi vì nền đen của nó. Để tránh điều này, ta sẽ lưu trữ kết quả mô phỏng vào một tệp có phần mở rộng là mat, (được gọi là mat-file), với một tên biến phù hợp. Để làm việc này ta cần lấy khối *To File* trong thư viện sink để lưu trữ kết quả vào mat-file. Trong khối này ta có thể đặt tên tệp và đặt tên cho biến cần lưu trữ kết quả. Các kết quả này sẽ được đọc ra bởi lệnh load, và thuận tiện cho các việc xử lý tiếp theo. Để minh họa ta thực hiện mô phỏng hệ theo sơ đồ trên hình 8-29. Các kết quả mô phỏng này sẽ được lưu trữ vào tệp có tên (sim_results.mat), với tên biến qv. Biến qv này sẽ có 5 cột, cột thứ nhất là thời gian t, bốn cột còn lại là chứa các biến $q_1, q_2, \dot{q}_1, \dot{q}_2$.

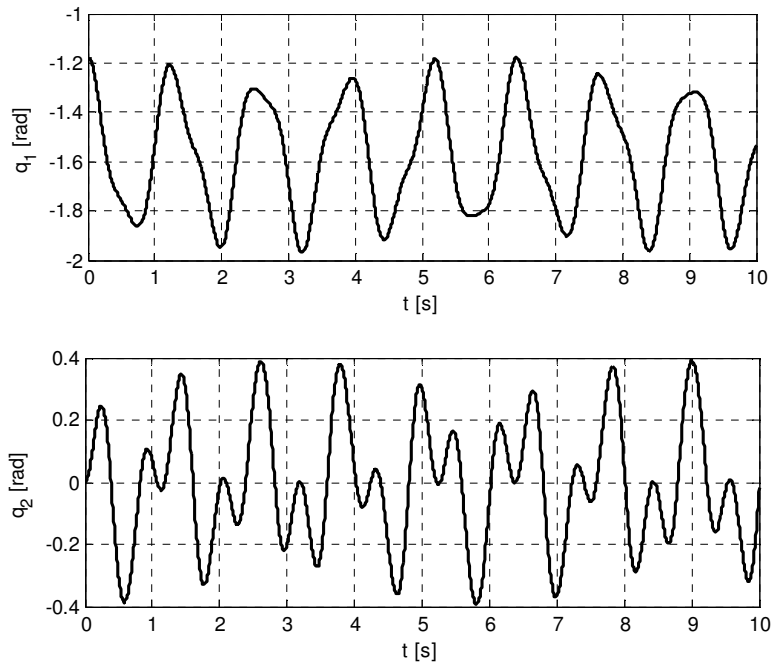


Hình 8-29. Sử dụng khối To file trong sink để lưu kết quả

Để vẽ đồ thị các biến này theo thời gian, ta thực hiện các dòng lệnh sau trong cửa sổ lệnh, hoặc có thể soạn sẵn một script.

```
load sim_results.mat
t=qv(1,:);
q1=qv(2,:);    q2=qv(3,:);
q1_dot=qv(4,:); q2_dot=qv(5,:);
figure(1)
plot(t,q1,'-k','linewidth',2), grid on
xlabel('t [s]'); ylabel('q_1 [rad]');
```

```
figure(2)
plot(t,q2,'-k','linewidth',2), grid on
xlabel ('t [s]'); ylabel ('q_2 [rad]');
% save ve_q_t.m
```



Hình 8-30. Kết quả mô phỏng – đồ thị các góc theo thời gian

8.6 Bài tập thực hành

Xây dựng mô hình simulink để mô phỏng các hệ cho bởi các phương trình vi phân sau :

1. Phương trình vi phân chuyển động của con lắc vật lý

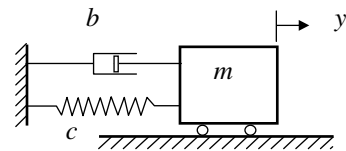
$$J_o \ddot{\varphi} + b \dot{\varphi} + mgl \sin \varphi = 0, \text{ với } J_o = 1, b = 0.1, mgl = 10,$$

Lấy các điều kiện đầu $\varphi_o = 0.30$, $\dot{\varphi}_o = 0$.

2. Hệ dao động gồm khối lượng - lò xo - cản nhớt có phương trình vi phân mô tả như sau:

$$\ddot{y} + \frac{b}{m} \dot{y} + \frac{c}{m} y = 0$$

với $m = 2 \text{ kg}$, $b = 40 \text{ N.s/m}$, $c = 450 \text{ N/m}$

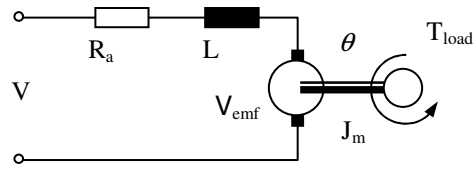


Điều kiện đầu là $y(0) = 0.01 \text{ m}$, $\dot{y}(0) = 0$. Thực hiện mô phỏng bằng simulink trong khoảng thời gian $t = 0 \dots 2 \text{ s}$ và đưa ra các kết quả dạng đồ thị $y(t)$ và $\dot{y}(t)$.

3. Cho sơ đồ động cơ điện một chiều như trên hình vẽ và phương trình vi phân mô tả như sau:

$$J_m \frac{d^2 \theta}{dt^2} + b \frac{d\theta}{dt} = T(t) = K_T i(t),$$

$$L \frac{di}{dt} + R i(t) + B_b \frac{d\theta}{dt} = V(t)$$

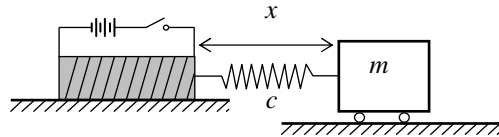


Sơ đồ động cơ điện một chiều

Hãy hạ bậc đưa hệ trên về hệ phương trình vi phân bậc nhất với các biến trạng thái

$$\mathbf{x} = [\theta, \dot{\theta}, i]^T$$

4. Khối sắt nhiễm từ có khối lượng m được nối với một lò xo độ cứng c , chiều dài L . Khối này ở trạng thái nghỉ tại $x = L$, khi bật nam châm điện xuất hiện một lực đẩy $F = k/x^2$ tác dụng lên khối sắt.



Phương trình vi phân mô tả chuyển động của khối này là

$$m\ddot{x} = k/x^2 - c(x - L)$$

Sử dụng phương pháp ode45 tích phân phương trình vi phân trên với các thông số của hệ $m = 1 \text{ kg}$, $c = 120 \text{ N/m}$, $k = 5 \text{ N} \cdot \text{m}^2$, $L = 0.20 \text{ m}$.

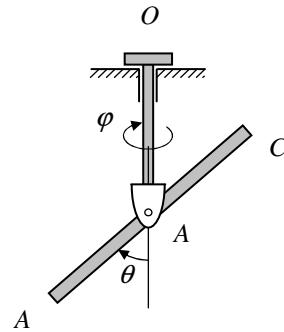
5. Thanh đồng chất ABC được nối bằng bản lề tron B với thanh OA quay được quanh trục đứng. Bỏ qua ma sát, phương trình vi phân chuyển động của hệ là

$$\ddot{\theta} = \dot{\varphi}^2 \sin \theta \cos \theta,$$

$$\ddot{\varphi} = -2\dot{\theta}\dot{\varphi} \cot \theta$$

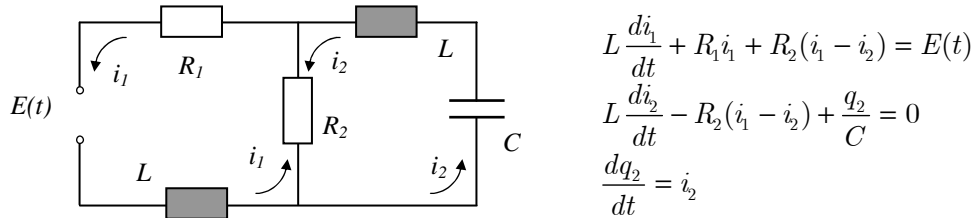
Giải hệ phương trình trên bằng phương pháp ode4 với điều kiện đầu

$$\theta(0) = \pi/12 \text{ rad}, \dot{\theta}(0) = 0, \varphi(0) = 0, \dot{\varphi}(0) = 20 \text{ rad/s}$$



đưa ra kết quả dạng đồ thị trong khoảng $t = 0 \dots 2 \text{ s}$.

6. Cho sơ đồ mạch điện như trên hình vẽ. Áp dụng định luật Kirchhoff ta nhận được phương trình vi phân của mạch điện như sau:



Cho biết các số liệu $R_1 = 4 \, \Omega$, $R_2 = 10 \, \Omega$, $L = 0.032 \, \text{H}$, $C = 0.53 \, \text{F}$ và điện áp đặt lên mạch có dạng

$$E(t) = \begin{cases} 20 \, \text{V} & \text{khi } 0 < t < 0.005 \, \text{s} \\ 0, & \text{khi } t \geq 0.005 \, \text{s} \end{cases}$$

Hãy giải và vẽ ra đồ thị các dòng điện $i_1(t)$, $i_2(t)$ theo thời gian $t = 0 \dots 0.05 \, \text{s}$.

7. Phương trình vi phân dao động cưỡng bức của hệ tuyến tính n bậc tự do được cho dạng ma trận như sau

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{B}\dot{\mathbf{q}} + \mathbf{C}\mathbf{q} = \mathbf{f}(t), \text{ với điều kiện đầu } \mathbf{q}(0) = \mathbf{q}_0, \quad \dot{\mathbf{q}}(0) = \dot{\mathbf{q}}_0.$$

với \mathbf{q} là vector tọa độ suy rộng, $\mathbf{M}, \mathbf{B}, \mathbf{C}$ là các ma trận vuông cấp n , tương ứng là ma trận khối lượng, ma trận cản, và ma trận độ cứng. Vectơ lực kích động $\mathbf{f}(t)$.

Hãy viết m-file và kết hợp với simulink thực hiện việc giải tìm nghiệm của hệ dao động với các số liệu

$$\mathbf{M} = \begin{bmatrix} 12 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 20 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 10 & -10 & 0 \\ -10 & 20 & -10 \\ 0 & -10 & 10 \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} 1000 & -1000 & 0 \\ -1000 & 2000 & -1000 \\ 0 & -1000 & 1000 \end{bmatrix}, \quad \mathbf{f}(t) = \begin{bmatrix} 5 \sin 10t \\ 0 \\ 0 \end{bmatrix}.$$

Chương 9.

Giải một số bài toán trong kỹ thuật bằng Matlab

Trong chương này chúng ta sẽ sử dụng Matlab để giải một số bài toán trong kỹ thuật. Đó là các bài toán xác định ứng lực trong các thanh và xác định đường đàn hồi, biểu đồ mômen uốn và lực cắt của dầm uốn phẳng trong Sức bền vật liệu; bài toán quỹ đạo chuyển động của viên đạn; xác định góc xiên để viên đạn bắn trúng đích; các bài toán dao động của con lắc đơn và kép; bài toán phân tích động học cơ cấu; bài toán động học ngược rôbot công nghiệp. Với mỗi bài toán phương pháp giải và chương trình Matlab được đưa ra cùng với các kết quả tính toán ở dạng số hoặc dạng đồ thị.

9.1 Bài toán hệ thanh

Trong thực tế kỹ thuật, ta hay gặp các kết cấu có thể được mô hình hóa như một hệ thanh không trọng lượng nối với nhau bằng các bản lề trơn, được gọi là các nút, tải trọng chỉ tác dụng tại các nút. Đối với các hệ thanh tĩnh định, bằng phương pháp cân bằng nút ta sẽ thiết lập được một hệ phương trình đại số tuyến tính cho việc xác định ứng lực trong các thanh.

Trong các ví dụ sau ta xét việc thiết lập các phương trình và giải các phương trình này trong Matlab.

Hệ thanh tĩnh định

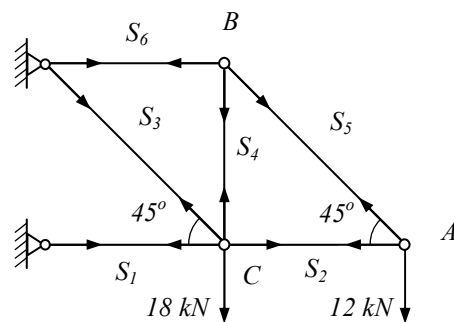
Ví dụ 1. Cho hệ thanh phẳng có kết cấu và chịu tải trọng như trên hình vẽ.

Tách và viết phương trình cân bằng cho các nút A, B và C, ta thu được các phương trình sau:

Nút C:

$$\begin{aligned}-S_1 + S_2 - S_3 \cos 45^\circ &= 0 \\ S_3 \sin 45^\circ + S_4 &= 18\end{aligned}$$

Nút A:



Hình 9-1

$$S_2 + S_5 \cos 45^\circ = 0$$

$$S_5 \sin 45^\circ = 12$$

Nút B:

$$S_5 \cos 45^\circ - S_6 = 0$$

$$S_4 + S_5 \sin 45^\circ = 0$$

Các phương trình trên được viết dạng ma trận như sau:

$$\begin{bmatrix} -1 & 1 & -\cos 45^\circ & 0 & 0 & 0 \\ 0 & 0 & \sin 45^\circ & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -\cos 45^\circ & 0 \\ 0 & 0 & 0 & 0 & \sin 45^\circ & 0 \\ 0 & 0 & 0 & 0 & \cos 45^\circ & 1 \\ 0 & 0 & 0 & -1 & -\sin 45^\circ & 0 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 12 \\ 0 \\ 12 \\ 0 \\ 0 \end{bmatrix}$$

Sau khi nhập ma trận hệ số A và véc tơ b trong Matlab, giải ra ta nhận được:

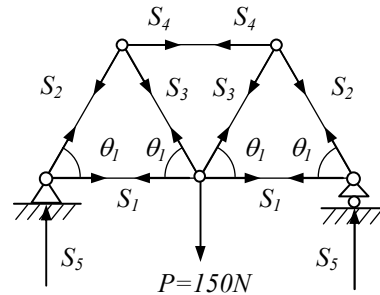
```
>> c45 = cos(pi/4); s45 = sin(pi/4);
>> A = [-1 1 -c45 0 0 0;
        0 0 s45 1 0 0;
        0 -1 0 0 -c45 0;
        0 0 0 0 s45 0;
        0 -1 0 0 c45 1;
        0 0 0 -1 -s45 0];
>> b = [0; 12; 0; 12; 0; 0];
>> x = A^-1*b % hoặc x = inv(A)*b
x =
-42.0000
-12.0000
 42.4264
-12.0000
 16.9706
-24.0000
```

Kết quả trên cho biết lực trong thanh 3 lớn nhất, các thanh 1, 2, 4, 6 chịu nén và các thanh 3, 5 chịu kéo.

Ví dụ 2. Cho hệ thanh phẳng có kết cấu và chịu tải trọng như trên hình 9-2. Sử dụng phương pháp cân bằng nút cho hệ giàn phẳng (đối xứng) như trên hình vẽ và thu được hệ phương trình đại số tuyến tính

$$\begin{bmatrix} c & 1 & 0 & 0 & 0 \\ 0 & s & 0 & 0 & 1 \\ 0 & 0 & 2s & 0 & 0 \\ 0 & -c & c & 1 & 0 \\ 0 & s & s & 0 & 0 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 150 \\ 0 \\ 0 \end{bmatrix}$$

với $s = \sin \theta, c = \cos \theta$ và các ẩn cần tìm là S_i ($i = 1, \dots, 5$). Hãy viết chương trình Matlab để tính các lực này, khi cho trước góc θ . Chạy chương trình với $\theta = 56^\circ$.



Hình 9-2

Triển khai trong Matlab như sau:

```
function Vidu92
disp('Vao goc theta () so duong n: ')
theta = input('theta = ')
c = cosd(theta); s = sind(theta);
A = [c 1 0 0 0;
     0 s 0 0 1;
     0 0 2*s 0 0;
     0 -c c 1 0;
     0 s s 0 0];
b = [0; 0; 150; 0; 0];
x=A^-1*b;
disp('Ung luc trong cac thanh: ')
x
```

Chạy chương trình và nhập $\theta = 56$ (độ), ta nhận được kết quả, đó là ứng lực trong các thanh.

```
x =
    161.7802
   -90.4663
    90.4663
  -101.1763
    75.0000
```

Hệ thanh siêu tĩnh

Đối với các hệ thanh siêu tĩnh, do có các liên kết thừa, các phương trình cân bằng nút không đủ để ta giải tìm ra ứng lực trong các thanh. Để giải được bài toán này chúng ta cần phải chú ý thêm các điều kiện ràng buộc, các điều kiện liên kết. Các điều kiện này sẽ thể hiện sự phụ thuộc lẫn nhau biến dạng dài của các thanh. Dưới đây là một ví dụ.

Ví dụ 3. Xét hệ thanh siêu tĩnh đơn giản như trên hình 9-3.

Quan sát thấy, do nút A bị chặn không cho di chuyển theo phương ngang, nên tại đó xuất hiện lực liên kết N.

Viết phương trình cân bằng nút được

$$-S_1 \cos 45^\circ - S_2 \cos 30^\circ - N = 0$$

$$S_1 \sin 45^\circ - S_2 \sin 30^\circ = 18000$$

Theo quan hệ giữa lực tác dụng và biến dạng dài của thanh ta có

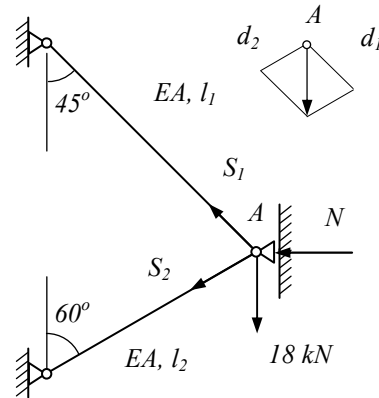
$$\Delta l_1 = S_1 l_1 / EA, \quad \Delta l_2 = S_2 l_2 / EA$$

Theo hình vẽ (A chỉ di chuyển theo phương đứng), ta có

$$\Delta l_1 \cos 45^\circ + \Delta l_2 \cos 30^\circ = 0$$

$$\Rightarrow \frac{S_1 l_1}{EA} \cos 45^\circ + \frac{S_2 l_2}{EA} \cos 30^\circ = 0$$

$$\Rightarrow S_1 l_1 \cos 45^\circ + S_2 l_2 \cos 30^\circ = 0$$



Hình 9-3

Như thế ta có hệ sau

$$\begin{bmatrix} \cos 45^\circ & \cos 30^\circ & 1 \\ \sin 45^\circ & -\sin 30^\circ & 0 \\ l_1 \cos 45^\circ & l_2 \cos 30^\circ & 0 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ N \end{bmatrix} = \begin{bmatrix} 0 \\ 18000 \\ 0 \end{bmatrix}$$

Triển khai trong Matlab

```
function Vidu9_3
alpha = 45;    beta = 60;
ca = cosd(alpha);  sa = sind(alpha);
cb = cosd(beta);   sb = sind(beta);
EA = 2*210000;      % A = 2 cm^2,    E = 2.1e+7 N/cm^2
l1 = 2;    l2 = 1.7; % m
A = [ca      cb      1;
     sa      -sb     0;
     l1*ca    l2*cb   0]
b = [0; 18000; 0]
x = A^-1*b;
disp('Ung luc trong cac thanh (N): ')
S1 = x(1)
S2 = x(2)
N = x(3)
```

9.2 Bài toán uốn phẳng của dầm

Vẽ biểu đồ mômen uốn, biểu đồ lực cắt và đường đàn hồi của một dầm là một bài toán tương đối phức tạp đặc biệt đối với dầm siêu tĩnh. Trong phần này, phương pháp sai phân hữu hạn được trình bày để giải quyết bài toán này.

Như ta biết trong giáo trình Sức bền vật liệu, phương trình vi phân đường đàn hồi của dầm có độ cứng chống uốn không đổi, $EI = \text{const}$, chịu uốn phẳng có dạng

$$EI \frac{d^4 w}{dx^4} = q(x)$$

với $q(x)$ là cường độ lực phân bố, hàm chuyển vị cần tìm $w(x)$ cần phải thỏa mãn thêm các điều kiện biên của dầm.

Mômen uốn và lực cắt được tính theo hàm chuyển vị $w(x)$

$$M(x) = -EI(x)w''(x), \quad Q(x) = -[EI(x)w''(x)]'.$$

Bây giờ ta sẽ sử dụng phương pháp sai phân hữu hạn để giải gần đúng phương trình vi phân cấp bốn trên với các điều kiện biên cho trước. Chia dầm với chiều dài L thành $n-1$ đoạn bằng nhau nhờ các điểm chia cách đều $x_i = (i-1)h$, $i = 1, 2, \dots, n$. Thay vì tìm hàm $w(x)$, $0 \leq x \leq L$, ta chỉ tìm nghiệm tại các điểm chia, $w_i = w(x_i)$. Sử dụng công thức sai phân trung tâm ta có

$$w^{(4)} = \frac{1}{h^4} (w_{i+2} - 4w_{i+1} + 6w_i - 4w_{i-1} + w_{i-2})$$

với $i = 1, 2, \dots, n$. Để thấy rõ các nút ảo (bên ngoài miền khảo sát) ta sẽ viết cụ thể hơn các phương trình trên

$$EI(w_{-1} - 4w_0 + 6w_1 - 4w_2 + w_3) = h^4 q_1$$

$$EI(w_0 - 4w_1 + 6w_2 - 4w_3 + w_4) = h^4 q_2$$

$$EI(w_1 - 4w_2 + 6w_3 - 4w_4 + w_5) = h^4 q_3$$

...

$$EI(w_{k-2} - 4w_{k-1} + 6w_k - 4w_{k+1} + w_{k+2}) = h^4 q_k, \quad i = k$$

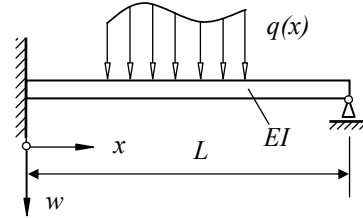
...

$$EI(w_{n-3} - 4w_{n-2} + 6w_{n-1} - 4w_n + w_{n+1}) = h^4 q_{n-1}$$

$$EI(w_{n-2} - 4w_{n-1} + 6w_n - 4w_{n+1} + w_{n+2}) = h^4 q_n$$

Mômen uốn và lực cắt được tính theo công thức sai phân




$$M_{bi} = -\frac{EI}{h^2} (w_{i-1} - 2w_i + w_{i+1}), \quad Q_i = -\frac{EI}{2h^3} [-w_{i-2} + 2w_{i-1} - 2w_{i+1} + w_{i+2}]$$



Hình 9-4.

Ta thấy có bốn ẩn nằm ngoài miền khảo sát là : $w_{-1}, w_0, w_{n+1}, w_{n+2}$. Các ẩn này sẽ được khử dựa vào các điều kiện biên như liệt kê trong bảng 9-1 dưới đây.

Bảng 9-1. Các điều kiện biên của dầm

Liên kết	tại đầu dầm $x_1 = 0$:	tại đầu dầm $x_n = L$:
Ngàm 	$w(0) = 0 : w_1 = 0$ $w'(0) = 0 : w_0 = w_2$	$w(L) = 0 : w_n = 0$ $w'(L) = 0 : w_{n+1} = w_{n-1}$
Gối 	$w(0) = 0 : w_1 = 0$ $w''(0) = 0 : w_0 = -w_2$	$w(L) = 0 : w_n = 0$ $w''(L) = 0 : w_{n+1} = -w_{n-1}$
Tự do 	$w''(0) = 0 : w_0 = 2w_1 - w_2$ $w'''(0) = 0 :$ $w_{-1} = 2w_0 - 2w_2 + w_3$	$w''(L) = 0 : w_{n+1} = 2w_n - w_{n-1}$ $w'''(L) = 0 :$ $w_{n+2} = 2w_{n+1} - 2w_{n-1} + w_{n-2}$

Nếu tại điểm x_i có lực tập trung tác dụng thì coi như lực này phân bố đều trên đoạn dài $h = x_{i+1/2} - x_{i-1/2}$, do đó ta lấy $q_i = F / h$. Còn đối với trường hợp mômen M tập trung, ta thay bằng hệ hai lực ngược chiều nhau đặt tại các điểm x_{i+1}, x_{i-1} , với $q_{i-1} = q_{i+1} = M / 2h^2$ (do ngẫu lực).

Ví dụ 1. Một dầm tiết diện không đổi chiều dài l , mômen chống uốn $EI = \text{const}$, một đầu ngàm còn đầu kia tự do, chịu tải trọng phân bố đều cường độ q_0 trên toàn bộ chiều dài. Hãy sử dụng phương pháp sai phân hữu hạn tính và vẽ ra đường chuyển vị, biểu đồ mômen và biểu đồ lực cắt. Biết các thông số hệ: $l = 5 \text{ m}$; $q_0 = 500 \text{ N/m}$; $E = 210000 \text{ N/mm}^2$; $I = 6 \cdot 10^6 \text{ mm}^4$; với số đoạn chia $n = 2000$.

Lời giải

Chú ý đến các điều kiện biên:

Ngàm tại $x = 0$:

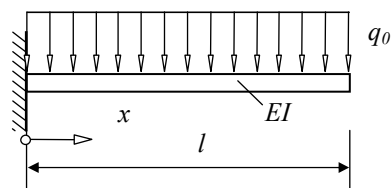
$$w(0) = 0 : w_1 = 0$$

$$w'(0) = 0 : w_0 = w_2$$

Đầu tự do $x = L$

$$w''(L) = 0 : w_{n+1} = 2w_n - w_{n-1}$$

$$w'''(L) = 0 : w_{n+2} = 2w_{n+1} - 2w_{n-1} + w_{n-2}$$



Hình 9-6

Thay vào hai phương trình đầu và hai phương trình cuối trong hệ trên ta nhận được

$$EIw_1 = 0$$

$$EI(w_{-1} - 8w_2 + w_3) = h^4 q_1, \quad EI(7w_2 - 4w_3 + w_4) = h^4 q_2$$

$$EI(-4w_2 + 6w_3 - 4w_4 + w_5) = h^4 q_3$$

$$EI(w_2 - 4w_3 + 6w_4 - 4w_5 + w_6) = h^4 q_4$$

...

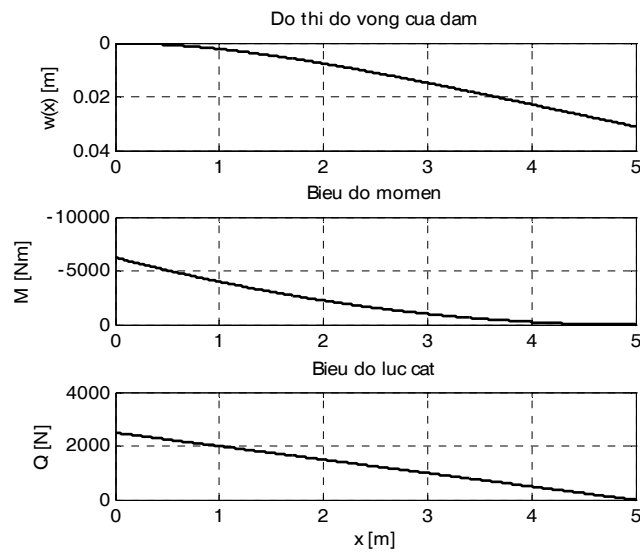
$$EI(w_{n-4} - 4w_{n-3} + 6w_{n-2} - 4w_{n-1} + w_n) = h^4 q_{n-2}$$

$$EI(w_{n-3} - 4w_{n-2} + 5w_{n-1} - 2w_n) = h^4 q_{n-1}$$

$$EI(2w_{n-2} - 4w_{n-1} + 2w_n) = h^4 q_n$$

Hay ở dạng ma trận

$$\begin{bmatrix} 1 & -8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & -4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & 6 & -4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -4 & 5 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -4 & 2 \end{bmatrix} \begin{bmatrix} w_{-1} \\ w_2 \\ w_3 \\ w_4 \\ \dots \\ \dots \\ w_{n-2} \\ w_{n-1} \\ w_n \end{bmatrix} = \frac{q_0 h^4}{EI} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \dots \\ \dots \\ 1 \\ 1 \\ 1 \end{bmatrix}$$



Hình 9-7. Kết quả tính toán (xem Beam_SPHH_Vd1.m)

Nhận xét thấy ma trận hệ số có dạng băng (5 đường chéo), để tiết kiệm bộ nhớ ta sử dụng 5 vectơ để lưu trữ ma trận này. Triển khai trong Matlab, ta nhận được kết quả dạng đồ thị như hình 9-7.

Giá trị mômen uốn cực đại tại ngàm $x = 0$, $M_{\max} = 6249,9$ Nm. So sánh với giá trị chính xác là $M_{\max} = \frac{1}{2}ql^2 = 6250$ Nm, ta thấy sai số tương đối là 0.0016 %.

Ví dụ 2. Một dầm tiết diện không đổi chiều dài l , mômen chống uốn $EI = \text{const}$, một đầu ngàm còn đầu kia đặt trên gối, chịu tải trọng là lực tập trung F . Hãy sử dụng phương pháp sai phân hữu hạn tính và vẽ ra đường chuyển vị, biểu đồ mômen và biểu đồ lực cắt.

Lời giải

Chú ý đến các điều kiện biên

Ngàm tại $x = 0$:

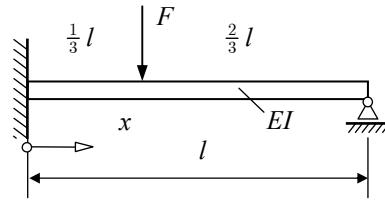
$$w(0) = 0: \quad w_1 = 0$$

$$w'(0) = 0: \quad w_0 = w_2$$

Gối tại $x = L$:

$$w(L) = 0 \quad w_n = 0$$

$$w''(L) = 0 \quad w_{n+1} = -w_{n-1}$$



Hình 9-8

Thay vào hai phương trình đầu và hai phương trình cuối trong hệ trên ta nhận được

$$EI(w_{-1} - 8w_2 + w_3) = 0$$

...

$$EI(7w_2 - 4w_3 + w_4) = 0$$

$$EI(w_{n-4} - 4w_{n-3} + 6w_{n-2} - 4w_{n-1}) = 0$$

$$EI(-4w_2 + 6w_3 - 4w_4 + w_5) = 0$$

$$EI(w_{n-3} - 4w_{n-2} + 5w_{n-1}) = 0$$

$$EI(w_2 - 4w_3 + 6w_4 - 4w_5 + w_6) = 0$$

$$EI(w_{n-2} + w_{n+2}) = 0$$

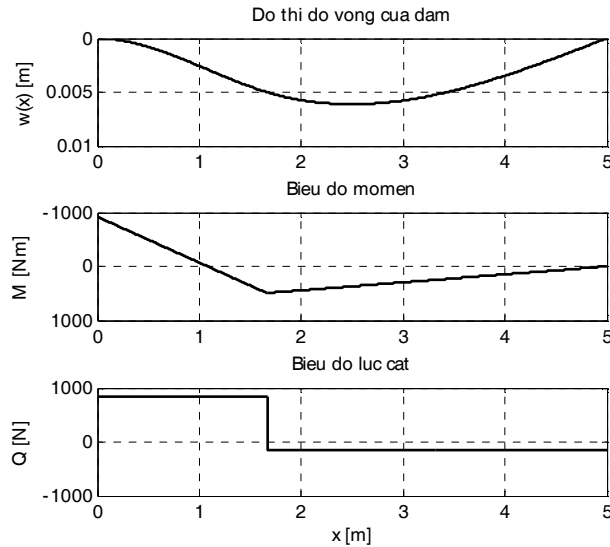
Hay ở dạng ma trận

$$\begin{bmatrix} 1 & -8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7 & -4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & 6 & -4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -4 & 6 & -4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -4 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_{-1} \\ w_2 \\ w_3 \\ w_4 \\ \dots \\ \dots \\ w_{n-2} \\ w_{n-1} \\ w_{n+2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ \dots \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Lực tập trung được coi như lực phân bố trên đoạn dài h , tức là ta có

$$q_{iF} = F / h.$$

Như thế vectơ vế phải có các phần tử bằng 0, trừ phần tử ứng với $x_{iF} = l/3$. Triển khai và thu được kết quả như trên hình 9-9.

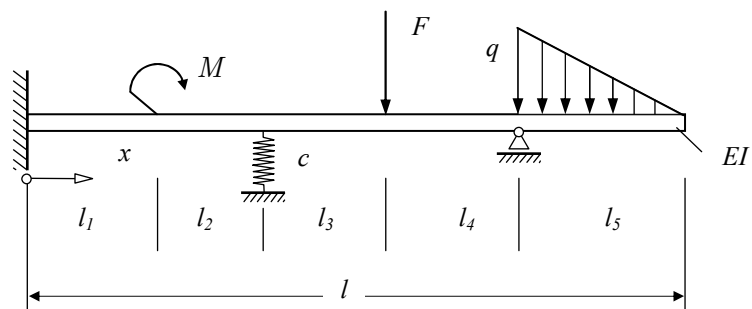


Hình 9-9. Kết quả tính toán (xem Beam_SPHH_Vd2.m)

Ví dụ 3. Một dầm tiết diện không đổi chiều dài l , mômen chống uốn bằng hằng số, $EI = \text{const}$, có liên kết và chịu tải trọng như trên hình vẽ. Hãy sử dụng phương pháp sai phân hữu hạn tính và vẽ ra đường chuyển vị, biểu đồ mômen và biểu đồ lực cắt. Biết các thông số hệ: $l = 5 \text{ m}$, $l_1 = l_2 = l_3 = l_4 = l_5$; $q = 300 \text{ N/m}$; $M = 50 \text{ Nm}$; $F = 200 \text{ N}$; $E = 210000 \text{ N/mm}^2$; $I = 10^5 \text{ mm}^4$; $c = 10^6 \text{ N/m}$; sử dụng 3000 đoạn chia.

Lời giải

Các điều kiện biên tại hai đầu dầm $x = 0, x = L$ như trong ví dụ 1 và ví dụ 2.



Hình 9-10. Kết quả tính toán (xem Beam_SPHH_Vd3.m)

Trong ví dụ này ta cần chú ý đến các lực, ngẫu lực tập trung, gối mềm và gối cứng:

Lực tập trung tác dụng tại vị trí iF suy ra $q(iF) = F/h$.

Mômen tập trung tại vị trí iM được thay bằng hai lực ngược chiều cùng trị số tác dụng tại hai nút (trước và sau)

$$q(iM-1) = -M/2h^2, \quad q(iM+1) = M/2h^2$$

Gối cứng $w_{(i=iGc)} = 0$, tức là dòng thứ i của ma trận A chỉ có phần tử $a(i,i) = 1$, $a(i,j) = 0, i \neq j$, $b(i) = 0$.

Gối mềm, coi lực lò xo, $F_{lx} = -cw_i$, như là lực tập trung tác dụng tại nút

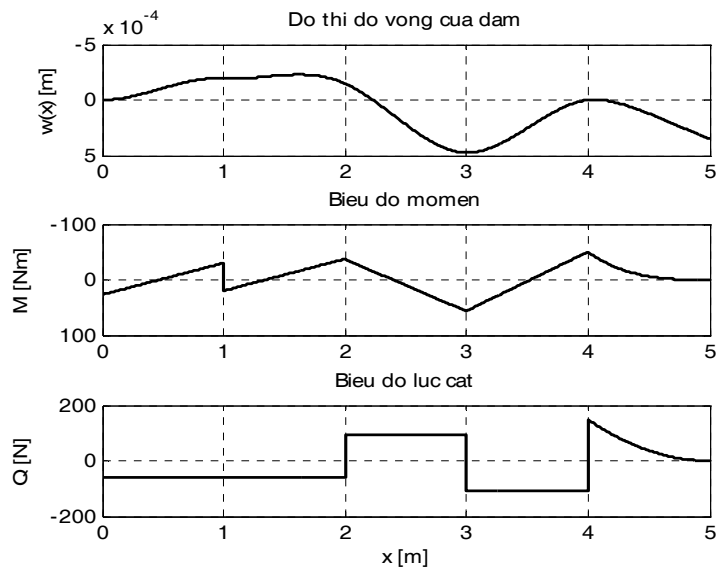
$$q_{iGm} = -(c/h)w_{iGm},$$

Như thế phương trình với chỉ số iGm được viết lại thành

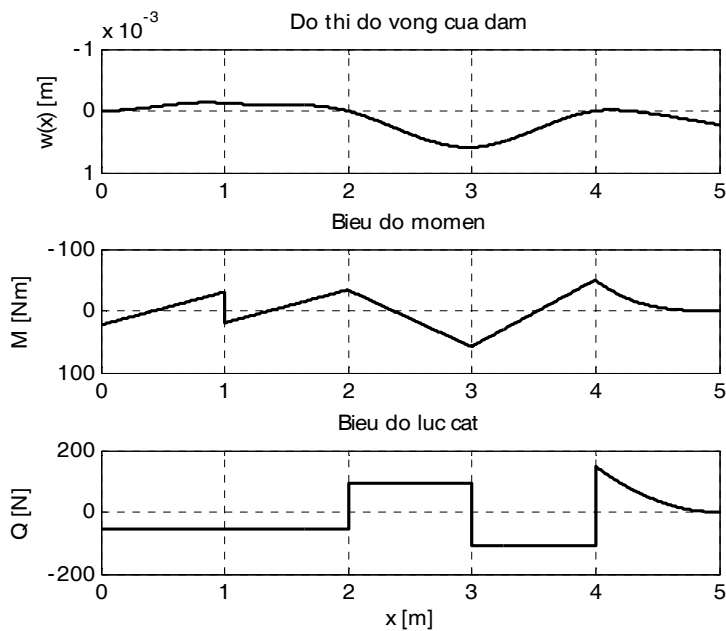
$$w_{i-2} - 4w_{i-1} + \left(6 - \frac{h^4}{EI} \frac{c}{h}\right)w_i - 4w_{i+1} + w_{i+2} = 0 \quad (\text{gối mềm tại đây})$$

Thực hiện trong Matab với số điểm chia $n = 3001$, số đoạn chia là 3000, thu được kết quả như trên hình 9-11.

Nếu cho hệ số cứng lò xo tăng lên rất lớn ($c = 10^{15}$ N/m) thì ta có thể coi như gối cứng và thu được kết quả như trên hình 9-12.



Hình 9-11. Kết quả tính toán (xem Beam_SPHH_Vd3.m)



Hình 9-12. Kết quả tính toán khi gối mềm trở thành gối cứng (xem Beam_SPHH_Vd3.m)

9.3 Bài toán quỹ đạo chuyển động của viên đạn

Trong bài toán này ta xét chuyển động của viên đạn khối lượng m được bắn lên với vận tốc v_0 , góc nghiêng của nòng súng so với phương ngang là α . Khảo sát chuyển động của viên đạn trong các trường hợp: a) bỏ qua lực cản của không khí và b) có lực cản tỷ lệ tuyến tính và bình phương vận tốc.

Phương trình vi phân chuyển động của viên đạn trong mặt phẳng đứng

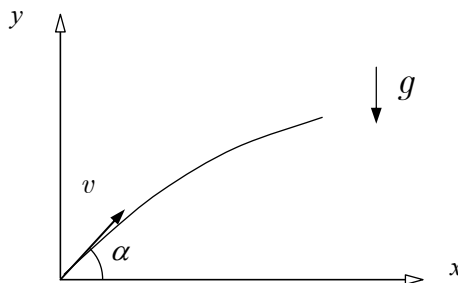
$$m\ddot{x} = -F_{cx}$$

$$m\ddot{y} = -mg - F_{cy}$$

Lực cản không khí tỷ lệ vận tốc

$$F_{c1} = c_1 v = c_1 \sqrt{\dot{x}^2 + \dot{y}^2} : F_{c1x} = c_1 \dot{x}, \quad F_{c1y} = c_1 \dot{y}$$

Lực cản không khí tỷ lệ bình phương vận tốc



Hình 9-13.

$$F_{c_2} = c_q v^2 = c_q (\dot{x}^2 + \dot{y}^2)$$

$$F_{c_{2x}} = c_n v^2 \dot{x} / v = c_n \dot{x} \sqrt{\dot{x}^2 + \dot{y}^2}, \quad F_{c_{2y}} = c_n v^2 \dot{y} / v = c_n \dot{y} \sqrt{\dot{x}^2 + \dot{y}^2}$$

Hạ bậc nhận được phương trình vi phân như sau

$$\dot{x} = v_x$$

$$\dot{v}_x = -\frac{1}{m} (c_l \dot{x} + c_n \dot{x} \sqrt{\dot{x}^2 + \dot{y}^2})$$

$$\dot{y} = v_y$$

$$\dot{v}_y = -\frac{1}{m} (mg + c_l \dot{y} + c_n \dot{y} \sqrt{\dot{x}^2 + \dot{y}^2})$$

Trước hết viết một m-file mô tả phương trình vi phân trên, trong đó các thông số khối lượng và các hệ số cần được khai báo là biến toàn cục.

```
function ydot = dandao(t,y)
% ptvp cd cua vien dan: m khoi luong, cl he so can tuyen tinh,
% cn he so can ty le binh phuong. vector y = [y1, y2, y3, y4]';
% y1 la toa do x, y2 la van toc vx
% y3 la toa do y, y4 la van toc vy

global m cl cn
g = 9.81; % gia toc trong trung
v = (y(2)^2 + y(4)^2)^0.5; ydot = zeros(4,1);
ydot(1) = y(2);
ydot(2) = -1/m*(cl*y(2) + cn*v*y(2));
ydot(3) = y(4);
ydot(4) = -1/m*(m*g + cl*y(4) + cn*v*y(4));
```

Lệnh ode45 được sử dụng để giải phương trình vi phân với các điều kiện đầu

$$x(0) = y(0) = 0, \dot{x}(0) = v_o \cos \alpha, \dot{y}(0) = v_o \sin \alpha$$

Ở đây ta sẽ khảo sát quỹ đạo chuyển động của viên đạn có khối lượng $m = 25$ kg, vận tốc tại nòng súng $v_o = 300$ m/s, với các góc bắn khác nhau từ 10 đến 90 độ.

Từ kết quả này ta có thể chỉ ra được tầm xa và độ cao của viên đạn.

a) Trường hợp bỏ qua lực cản không khí, ta cho các hệ số cản $c_l = c_n = 0$

```
global m cl cn
m = 25; cl = 0.0; cn = 0.0;
t_start = 0; t_end = 100.0;
vo = 300; alpha = [10 : 10 : 90];
N = length(alpha); ymax = zeros(1,N);

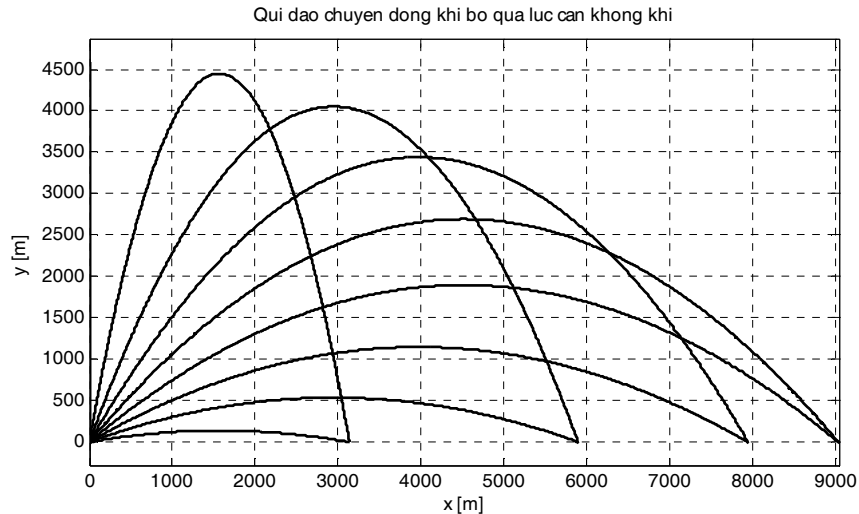
for i = 1:length(alpha)
    y1 = 0; y2 = vo*cosd(alpha(i));
```

```

y3 = 0; y4 = vo*sind(alpha(i));
y0 = [y1 y2 y3 y4]'; % vector chua dieu kien dau
[t,y] = ode45('dandao',[0: 0.05: t_end],y0);
ymax(i) = max(y(:,3));
% tim thoi gian chuyen dong
for k = 10:length(t)
    if y(k,3)>=0 && y(k+1,3)<0
        t_ground(i) = t(k); % thoi gian tiep dat
        ki = k+1;
    end
end
tamxa(i) = y(ki, 1); % Xac dinh tam xa va tam cao
tamcao(i) = max(y(1:ki, 3));
plot(y(1:ki,1),y(1:ki,3),'k-', 'linewidth',1.5),
xlabel('x [m]'), ylabel('y [m]'), grid on
hold on
end
axis equal
title('Qui dao chuyen dong khi bo qua luc can khong khi')
disp(['goc ban (do) ', 'Thoi gian bay (s) ', ' tam xa (m)', '
tam cao (m)'])

for i = 1:length(alpha)
disp([sprintf('%4d',alpha(i)), sprintf(' %6.3f',t_ground(i)),
sprintf(' %6.3f',tamxa(i)), sprintf(' %6.3f',tamcao(i))])
end

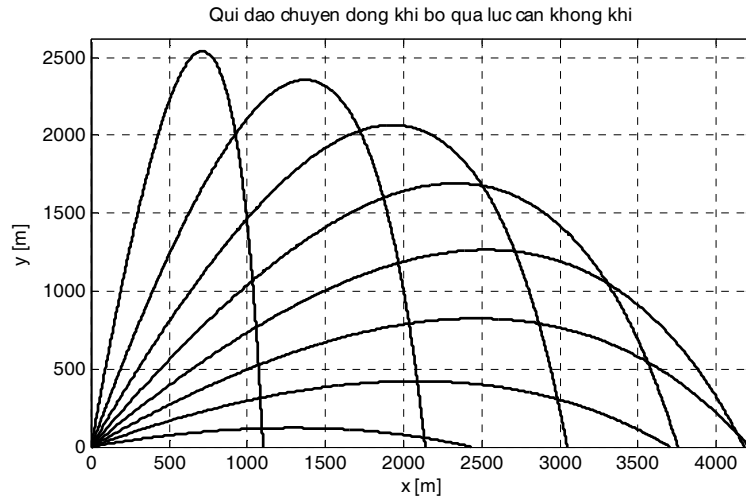
```



Hình 9-14. Quỹ đạo chuyển động của viên đạn khi không có cản với các góc bắn khác nhau 10, 20, . . . , 90°

Từ các mô phỏng trên, ta có thể tổng hợp các kết quả trong bảng 9-2.

b) Trường hợp có tính đến lực cản tuyến tính của không khí: $c_l = 1, c_n = 0$



Hình 9-15. Quỹ đạo chuyển động của viên đạn khi có cản tuyến tính với các góc bắn khác nhau 10, 20, ..., 90°

Từ các mô phỏng trên, ta có thể tổng hợp các kết quả trong bảng 9-3.

Bảng 9-2.

Kết quả mô phỏng khi không có cản

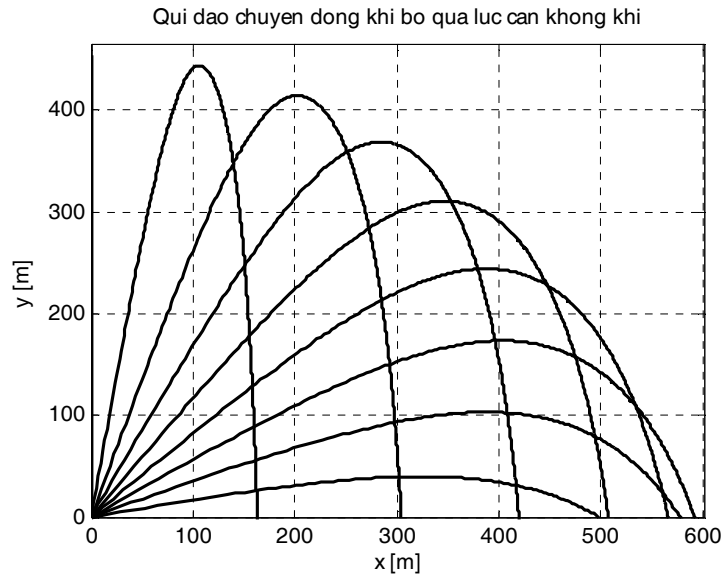
Góc bắn (độ)	Thời gian bay (s)	Tầm xa (m)	Tầm cao (m)
10	10.60	3146.461	138.319
20	20.90	5905.968	536.595
30	30.55	7950.113	1146.789
40	39.30	9043.155	1895.302
50	46.85	9044.022	2691.851
60	52.95	7950.000	3440.366
70	57.45	5899.847	4050.560
80	60.20	3138.691	4448.835
90	61.15	0000.000	4587.154

Bảng 9-3.

Kết quả mô phỏng khi có cản tỷ lệ

Góc bắn (độ)	Thời gian bay (s)	Tầm xa (m)	Tầm cao (m)
10	9.95	2435.036	121.402
20	18.60	3705.257	422.214
30	26.10	4213.168	823.918
40	32.45	4179.547	1263.938
50	37.70	3755.920	1691.532
60	41.80	3046.891	2066.010
70	44.75	2137.729	2355.927
80	46.55	1100.429	2538.861
90	47.15	0000.000	2601.335

c) Trường hợp lực cản không khí tỷ lệ bình phương vận tốc: $c_l = 0, c_n = 0.1$



Hình 9-16. Quĩ đạo chuyển động của viên đạn khi cản tỷ lệ bình phương vận tốc với các góc bắn khác nhau 10, 20, . . . , 90°

Từ các mô phỏng trên, ta có thể tổng hợp các kết quả trong bảng 9-4 sau

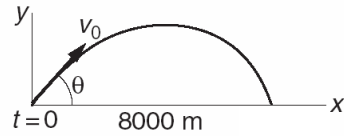
Bảng 9-4. Kết quả mô phỏng khi có cản tỷ lệ bậc nhất và bậc 2

Góc bắn (độ)	Thời gian bay (s)	Tầm xa (m)	Tầm cao (m)
10	5.40	498.857	39.726
20	8.70	578.956	103.061
30	11.35	592.698	173.318
40	13.65	566.398	244.065
50	15.65	507.712	310.553
60	17.35	419.968	368.575
70	18.60	304.410	414.172
80	19.45	162.797	443.653
90	19.70	000.000	453.958

9.4 Bài toán bắn trúng đích

Xét bài toán: Một quả đạn khối lượng m bay tự do trong không khí chịu lực cản khí động học $F_d = cv^2$, với v là vận tốc và c là hệ số cản. Phương trình vi phân chuyển động của viên đạn là

$$\begin{aligned}
m\ddot{x} &= -cv\dot{x}, \\
m\ddot{y} &= -mg - cv\dot{y}, \\
v &= \sqrt{\dot{x}^2 + \dot{y}^2}
\end{aligned}$$



Biết rằng vận tốc của viên đạn khi thoát khỏi miệng là nòng v_0 . Hãy xác định góc nghiêng θ để viên đạn đạt mục tiêu ở khoảng cách 8 km. Sử dụng các số liệu $m = 20$ kg, $c = 3.2 \times 10^{-4}$ kg/m, $v_0 = 500$ m/s, và $g = 9.80665$ m/s².

Phương pháp giải bài toán như sau, trước hết ta bắn thử với góc bắn θ nào đó giải phương trình vi phân chuyển động ta nhận được điểm tiếp đất với tầm xa $L = L(\theta)$, so sánh điểm rơi này mục tiêu ta được một sai lệch

$$r(\theta) = L(\theta) - L_{mt}$$

Từ phương trình sai số này cho ta xác định được góc bắn cần thiết.

Trong matlab ta cần xây dựng các m-file sau:

```
function F = dEqs(t,y) % First-order differential equations.
m = 20; c = 3.2e-4; g = 9.80665;
x_ = y(1); vx = y(2); y_ = y(3); vy = y(4);
v = sqrt(vx^2+vy^2);
xdot = vx; vxdot = 1/m*(-c*v*vx);
ydot = vy; vydot = 1/m*(-m*g-c*v*vy);

F = [xdot, vxdot, ydot, vydot]';

function y = inCond(v0, theta)
y = [0 v0*cosd(theta) 0 v0*sind(theta)]';

function r = residual(theta)
global tSTART tSTOP h k v0
k=k+1
[tSol,ySol] = ode45(@dEqs,[tSTART:h:tSTOP],inCond(v0, theta));
% tìm thời gian chuyển động
for j = 50:length(tSol)
    if ySol(j,3)>=0 && ySol(j+1,3)<0
        t_ground(j) = tSol(j); ki = j+1;
    end
end
% Tầm xa L(theta)
tamxa = ySol(ki, 1); r = tamxa-8000;

function root = newtonRaphson2(func,x,tol)
if nargin == 2; tol = 1.0e-8; end
if size(x,1) == 1; x = x'; end % x must be column vector
for i = 1:30
    [jac,f0] = jacobian(func,x);
    if sqrt(dot(f0,f0)/length(x)) < tol root = x; return end
```

```

    dx = jac\(-f0);
    x = x + dx;
    if sqrt(dot(dx,dx)/length(x)) < tol*max(abs(x),1.0)
        root = x; return
    end
end
error('Too many iterations')

function [jac,f0] = jacobian(func,x)
% Returns the Jacobian matrix and f(x).
h = 1.0e-4;          n = length(x);
jac = zeros(n);      f0 = feval(func,x);
for i =1:n
    temp = x(i);
    x(i) = temp + h;
    f1 = feval(func,x);
    x(i) = temp;
    jac(:,i) = (f1 - f0)/h;
end

function main

global tSTART tSTOP h k  v0
k=0; %so lan ban thu
tSTART = 0; tSTOP = 150;
h = 0.01;  v0 = 500;
theta = 40; % thu ban voi theta = 40 do

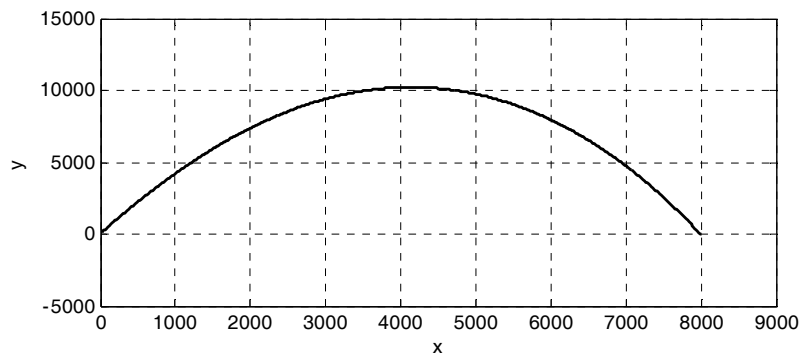
theta = newtonRaphson2(@residual,theta);
[tSol,ySol] = ode45(@dEqs,[tSTART:h:tSTOP],inCond(v0, theta));

% tim thoi gian chuyen dong
for j = 50:length(tSol)
    if ySol(j,3)>=0 && ySol(j+1,3)<0
        t_ground(j) = tSol(j); ki = j+1;
    end
end
end
% Tam xa L(theta)

tSTOP = tSol(ki)
[t,y] = ode45(@dEqs,[tSTART:h:tSTOP],inCond(v0, theta));
plot(y(:,1), y(:,3), 'k-', 'Linewidth', 2), grid on
xlabel('x'), ylabel('y')
disp('Goc ban can tim (do)')
theta

```

Kết quả thu được góc bắn cần thiết để đạt mục tiêu là $\theta = 77.8777^\circ$. Thời gian bay của viên đạn là 91.35 giây. Quỹ đạo chuyển động của viên đạn như trên hình 9-17.



Hình 9-17. Quỹ đạo chuyển động của viên đạn

9.5 Bài toán dao động

Con lắc đơn

Xét con lắc toán là một quả cầu nhỏ, khối lượng m treo vào dây mềm không khối lượng, không giãn. Phương trình vi phân chuyển động của con lắc trong trường hợp không cản:

$$ml^2\ddot{\varphi} + mgl \sin \varphi = 0 \Rightarrow \ddot{\varphi} = -(g/l) \sin \varphi$$

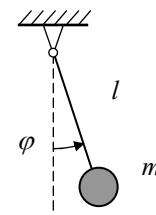
Nếu kể đến lực cản tỷ lệ với vận tốc hệ số k , lực cản $F_c = kv = kl\dot{\varphi}$, ta có

$$ml^2\ddot{\varphi} = -mgl \sin \varphi - kl^2\dot{\varphi}$$

Đặt $y_1 = \varphi$, $y_2 = \dot{\varphi}$, ta có

$$\dot{y}_1 = y_2$$

$$\dot{y}_2 = -\frac{1}{ml^2}(mgl \sin y_1 + kl^2 y_2) = -(g/l) \sin y_1 - (k/m)y_2$$



Hình 9-18

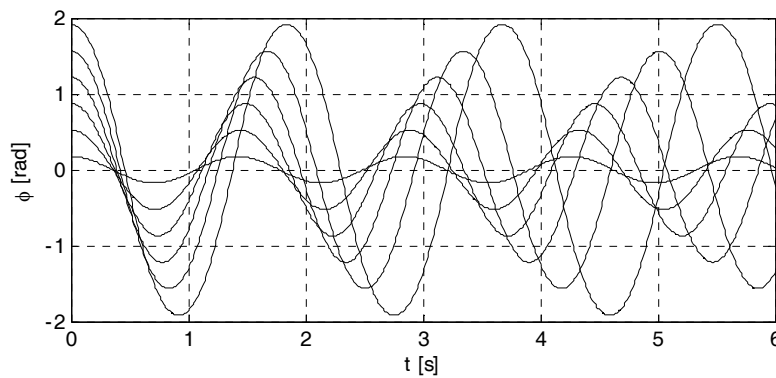
Triển khai trong Matlab

```
function ydot = conlac_don(t,y)
global m l k
g = 9.81;      % gia toc trong truong
v = l*y(2);
ydot = zeros(2,1);
ydot(1) = y(2);
ydot(2) = -g/l*sin(y(1)) - k/m*y(2);
```

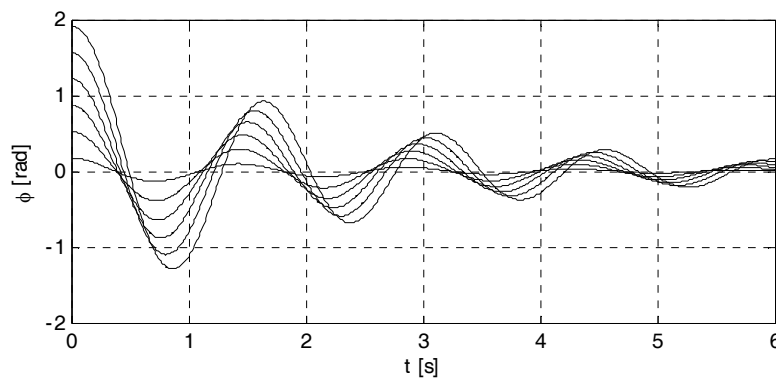

Trong phần sau ta sẽ khảo sát chuyển động của con lắc có khối lượng $m = 0.25$ kg, chiều dài dây $l = 0.5$ m, với các góc lệch ban đầu khác nhau từ 10 đến 110 độ. Các kết quả mô phỏng được đưa ra như trên hình 9-19, 9-20 và 9-21.

```
global m l k
m = 0.25; l = 0.5;
    k = 0.0; % không cản
% k = 0.1; % có cản

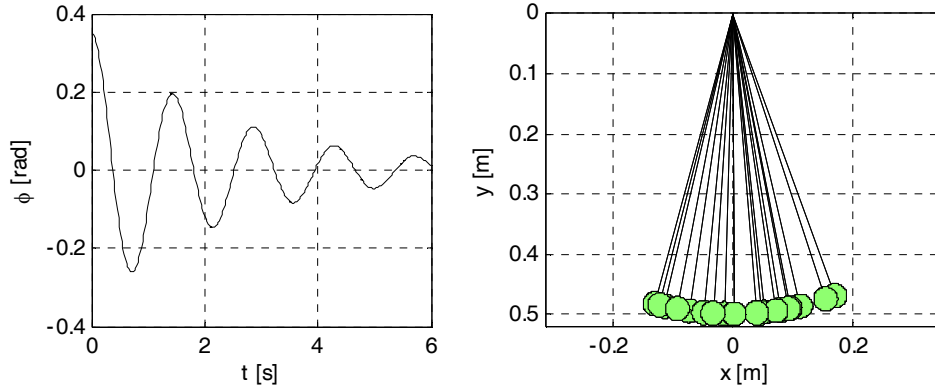
t_start = 0;    t_end = 6.0;
phi = [10 : 20 : 110];
for i = 1:length(phi)
    y1 = phi(i)*pi/180;    y2 = 0;    y0 = [y1 y2]';
    [t,y] = ode45('conlac_don',[0: 0.02: t_end],y0);
    plot(t, y(:,1),'k-', 'linewidth',1), grid on
    xlabel('t [s]'), ylabel('\phi [rad]'),
    hold on
end
```



Hình 9-19. Đồ thị theo thời gian của góc lắc khi không có cản, $k = 0$



Hình 9-20. Đồ thị theo thời gian của góc lắc khi có cản, $k = 0.2$

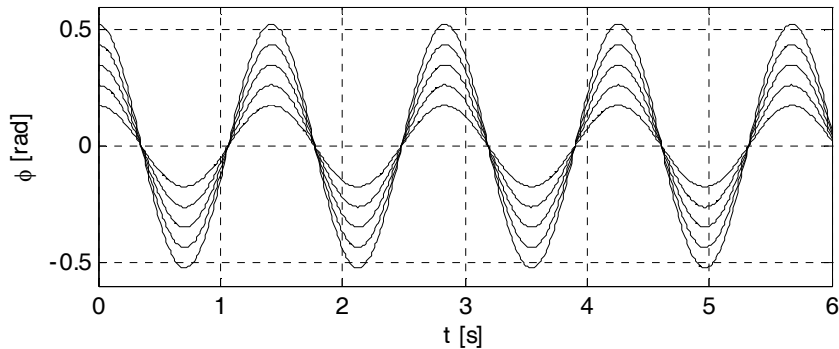


Hình 9-21. Chuyển động của con lắc, với $k = 0.20$

Xét trường hợp dao động nhỏ, ta có thể tuyến tính hóa quanh vị trí cân bằng và nhận được phương trình vi phân:

$$\dot{y}_1 = y_2, \quad \dot{y}_2 = -(g/l)y_1 - (k/m)y_2$$

Kết quả mô phỏng trong trường hợp không cản với các điều kiện đầu khác nhau được đưa ra như trên hình 9-22. Từ trên hình vẽ ta thấy rằng con lắc dao động với chu kỳ không đổi, không phụ thuộc vào điều kiện đầu.



Hình 9-22. Dao động nhỏ, không cản $k = 0$

Con lắc đơn dây treo đàn hồi

Xét con lắc là một quả cầu nhỏ được treo vào lò xo có độ cứng c , chiều dài tự nhiên là l (hình 9-23). Hệ hai bậc tự do với các tọa độ suy rộng s, φ . Động năng và thế năng của hệ:

$$T = \frac{1}{2} m[(l+s)^2 \dot{\varphi}^2 + \dot{s}^2], \quad \Pi = \frac{1}{2} cs^2 - mg(l+s) \cos \varphi$$

Sử dụng phương trình Lagrange loại 2 ta nhận được phương trình vi phân chuyển động của con lắc như sau:

$$m(l+s)^2\ddot{\varphi} + 2m(l+s)\dot{s}\dot{\varphi} + mg(l+s)\sin\varphi = 0$$

$$m\ddot{s} - m(l+s)\dot{\varphi}^2 - mg\cos\varphi + cs = 0$$

Triển khai trong Matlab

```
function ydot = conlac_danhroi(t,y)
% ptvp cd cua con lac dan hoi (m, c, L)
% vector y = [q1, q2, v1, v2]'
global m c L
g = 9.81;
q1 = y(1);    q2 = y(2);
qd1 = y(3);   qd2 = y(4);
qd = [qd1; qd2];

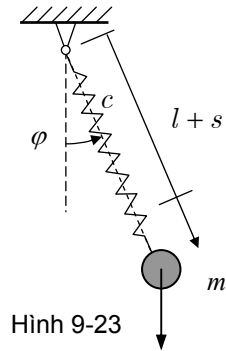
M_q = [m,      0;      0,      m*(L+q1)^2];
C_q = [ 0, -m*(L+q1)*qd2;  m*(L+q1)*qd2, -m*(L+q1)*qd1];
g_q = [c*q1-m*g*cos(q2); m*g*(L+q1)*sin(q2)];
ydot = zeros(4,1);
qdot = qd;
qd_dot = inv(M_q)*(-C_q*qd - g_q);
ydot = [qdot; qd_dot];
% save conlac_danhroi.m

=====

% main program
global m L c
m = 0.5; L = 0.5; c = 50; % gan cac thong so he
t_start = 0; t_end = 3.0;
q10 = 0.05; q20 = 30.0;
y1 = q10; y2 = q20*pi/180;
y3 = 0; y4 = 0;
y0 = [y1 y2 y3 y4]';

[t,y] = ode45('conlac_danhroi',[0: 0.01: t_end],y0);
figure(1)
subplot(2,1,1), plot(t, y(:,1),'k-','linewidth',1), grid on
ylabel('q_1 [m]')
subplot(2,1,2), plot(t, y(:,2),'k-','linewidth',1), grid on
ylabel('q_2 [rad]'), xlabel('t [s]'),

figure(2)
x0 = 0; y0 = 0;
hold on, plot(x0,y0,'o-','linewidth',1)
r = 0.02;
for i = 1:5:length(t)/2
x1(i) =(L+y(i,1))*sin(y(i,3)); y1(i)=-(L+y(i,1))*cos(y(i,3));
x_day = [x0, x1(i)]; y_day = [y0, y1(i)];
```



```

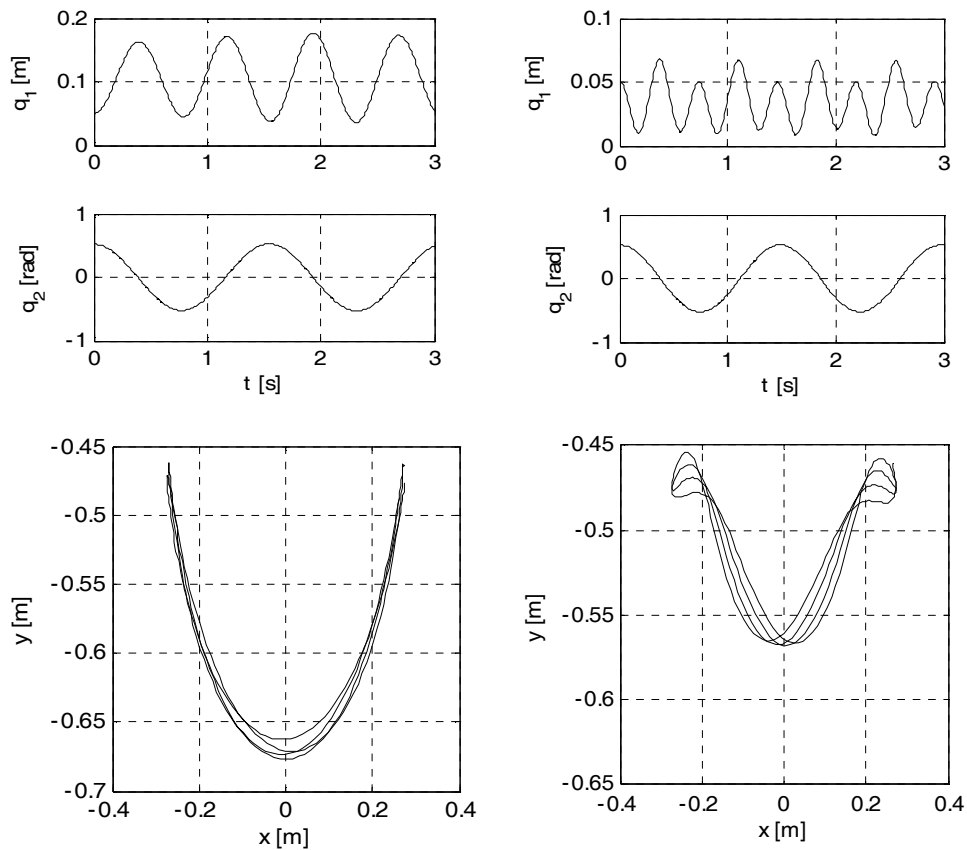
plot(x_day, y_day, 'k-', 'linewidth', 1), grid on, axis equal
tam1 = [x1(i), y1(i)];
vehinhtron(tam1, r);
end
xlabel('x [m]'), ylabel('y [m]')
axis([-0.8*L 0.8*L -1.5*L 0.05*L]);

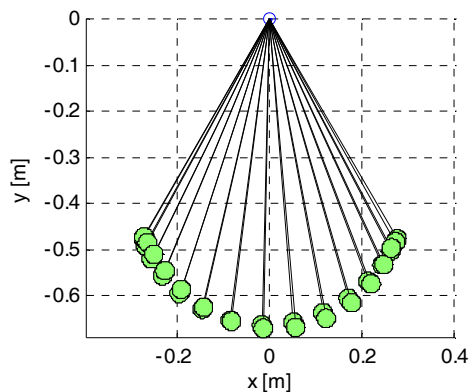
figure(3)
for i = 1:length(t)
    xC(i) = (L+y(i,1))*sin(y(i,3));
    yC(i) = -(L+y(i,1))*cos(y(i,3));
end
plot(xC, yC, 'k-', 'linewidth', 1), grid on

```

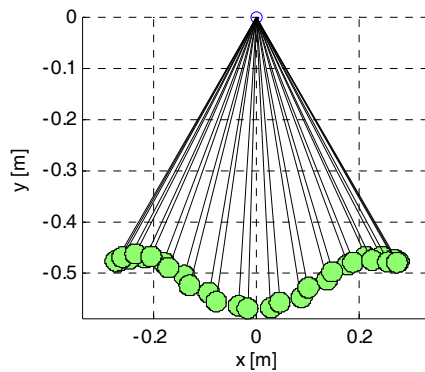
=====

Dưới đây là kết quả mô phỏng với các số liệu khác nhau của lò xo, và các điều kiện đầu khác nhau.





$m = 0.5; \quad L = 0.5; \quad c = 50;$
 $q_{10} = 0.05; \quad (\text{mét})$
 $q_{20} = 30.0; \quad (\text{độ})$



$m = 0.5; \quad L = 0.5; \quad c = 150;$
 $q_{10} = 0.05; \quad (\text{mét})$
 $q_{20} = 30.0; \quad (\text{độ})$

Hình 9-24. Kết quả mô phỏng con lắc đàn hồi

Con lắc kép

Phương trình vi phân chuyển động của con lắc kép nhận được nhờ phương trình Lagrange 2 được viết gọn lại ở dạng ma trận như sau:

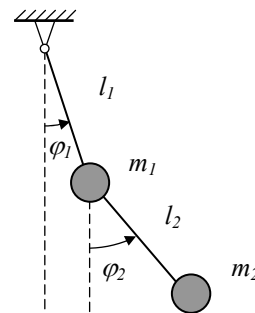
$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = 0$$

với

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} m_1 l_1^2 + m_2 l_2^2 & m_2 l_1 l_2 \cos(q_1 - q_2) \\ m_2 l_1 l_2 \cos(q_1 - q_2) & m_2 l_2^2 \end{bmatrix}$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} 0 & m_2 l_1 l_2 \sin(q_1 - q_2) \dot{q}_2 \\ -m_2 l_1 l_2 \sin(q_1 - q_2) \dot{q}_1 & 0 \end{bmatrix}$$

$$\mathbf{g}(\mathbf{q}) = \begin{bmatrix} (m_1 l_1 + m_2 l_2) g \sin q_2 & m_2 g l_2 \sin q_2 \end{bmatrix}^T.$$



Hình 9-25. Con lắc kép

Triển khai trong Matlab

```
function ydot=conlactoan_kep(t,y)
% vector y = [q1, q2, v1, v2]'
global m1 m2 l1 l2
g = 9.81;
q1 = y(1);    q2 = y(2);    qd1 = y(3);    qd2 = y(4);
qd = [qd1; qd2];
```

```

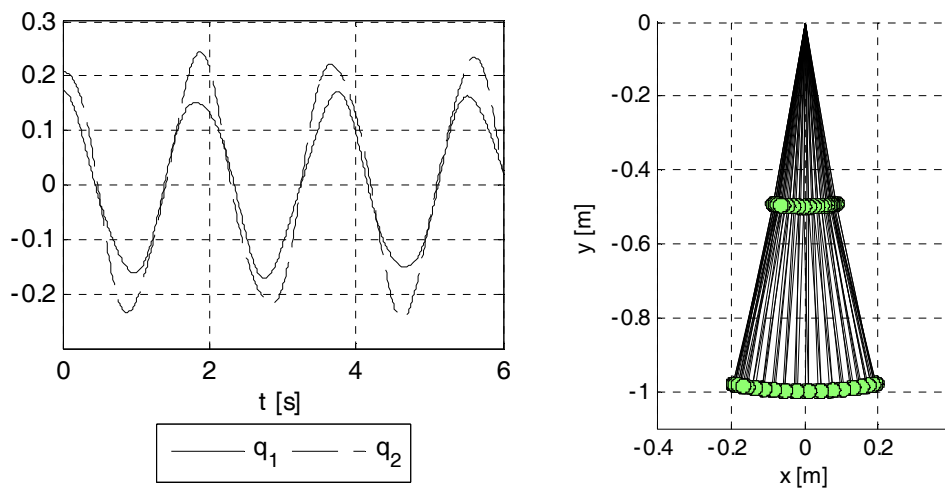
M_q = [m1*l1^2+m2*l1^2,      m2*l1*l2*cos(q1-q2);
        m2*l1*l2*cos(q1-q2), m2*l2^2   ];
C_q = [ 0,      m2*l1*l2*sin(q1-q2)*qd2;
        -m2*l1*l2*sin(q1-q2)*qd1,      0];
g_q = [g*l1*sin(q1)*(m1+m2); g*m2*l2*sin(q2)];

ydot = zeros(4,1);
qdot  = qd;
qd_dot = inv(M_q)*(-C_q*qd - g_q);
ydot   = [qdot; qd_dot];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

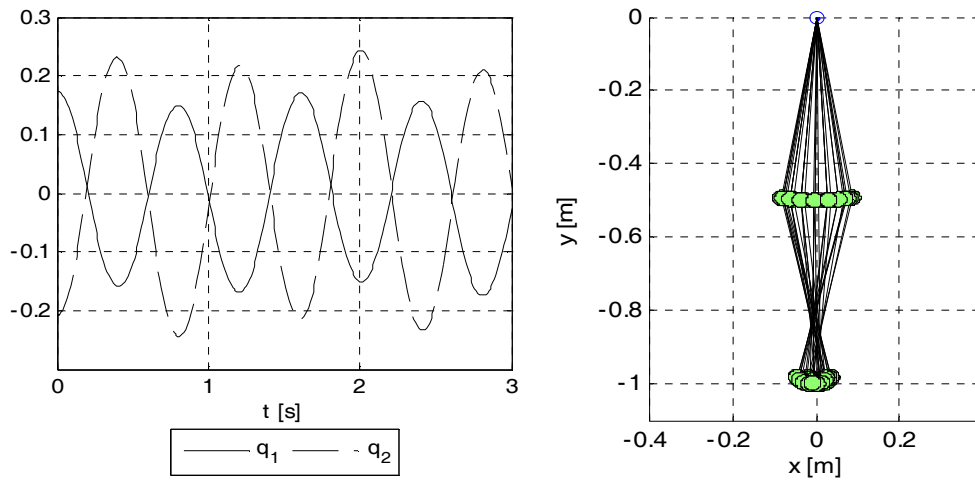
% main program
global m1 m2 l1 l2
m1 = 0.25;  l1 = 0.5;
m2 = 0.25;  l2 = 0.5;
t_start = 0;    t_end = 6.0;
q10 = 10;      q20 = 12;
y1 = q10*pi/180;  y2 = q20*pi/180;  y3 = 0;  y4 = 0;
y0 = [y1 y2 y3 y4]';
[t,y] = ode45('conlactoan_kep',[0: 0.01: t_end],y0);
plot(t, y(:,1),'k-', t, y(:,2),'k--', 'linewidth',1), grid on
legend('q_1','q_2'), xlabel('t [s]')
axis([0 t_end -0.3 0.3]);

```

Chạy chương trình với các điều kiện đầu khác nhau cho ta các kết quả như trên hình 9-26 và 9-27.



Hình 9-26. Kết quả mô phỏng với điều kiện đầu $q_{10} = 10$; $q_{20} = 12$;



Hình 9-27. Kết quả mô phỏng với điều kiện đầu $q_{10} = 10$; $q_{20} = -12$;

Dao động nhỏ của con lắc elliptic

Xét con lắc elliptic như hình 9-28. Xe A có khối lượng m_1 chuyển động trên đường ngang, lò xo có độ cứng c . Dây AB có chiều dài l , khối lượng không đáng kể và luôn căng. Tải trọng được coi như chất điểm có khối lượng m_2 . Biết khi $x = 0$ lò xo không bị biến dạng.

Sử dụng phương trình Lagrange loại 2, ta nhận được phương trình vi phân chuyển động:

$$(m_1 + m_2)\ddot{x} + m_2l\ddot{\varphi} \cos \varphi - m_2l\dot{\varphi}^2 \sin \varphi + cx = 0,$$

$$m_2l^2\ddot{\varphi} + m_2l\ddot{x} \cos \varphi + m_2gl \sin \varphi = 0$$

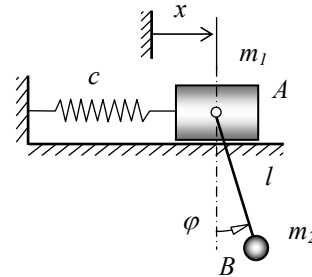
Trường hợp xét dao động nhỏ, coi $\sin \varphi \approx \varphi$, $\cos \varphi \approx 1$ và bỏ qua số hạng phi tuyến $\dot{\varphi}^2$, ta nhận được phương trình vi phân dao động nhỏ.

$$(m_1 + m_2)\ddot{x} + m_2l\ddot{\varphi} + cx = 0, \quad m_2l\ddot{x} + m_2l^2\ddot{\varphi} + m_2gl\varphi = 0$$

hay ở dạng ma trận

$$\begin{bmatrix} m_1 + m_2 & m_2l \\ m_2l & m_2l^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\varphi} \end{bmatrix} + \begin{bmatrix} c & 0 \\ 0 & m_2gl \end{bmatrix} \begin{bmatrix} x \\ \varphi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Từ đây ta sẽ sử dụng Matlab tìm các tần số dao động riêng và các dạng dao động riêng.



Hình 9-28

```

function Daodongnho_enliptic
m1 = 50; m2 = 40;      % kg
l = 5;                 % m
g = 9.81;              % m/s^2
c = 2000;              % N/m

M = [m1 + m2,    m2*l; m2*l m2*l^2];
C = [c,          0;    0,    m2*g*l];

n = size(M,1);
[V, D] = eig(C,M);
for i=1:n ome(i)=sqrt(D(i,i)); end

ome, v1 = V(:,1), v2 = V(:,2)
x = [0:1:n];
subplot(2,1,1)
plot(x, [0; V(:,1)], 'k-', 'linewidth',1), grid on, ylabel('v_1')
subplot(2,1,2)
plot(x, [0; V(:,2)], 'k-', 'linewidth',1), grid on, ylabel('v_2')

```

Các tần số riêng và các dạng
dao động riêng

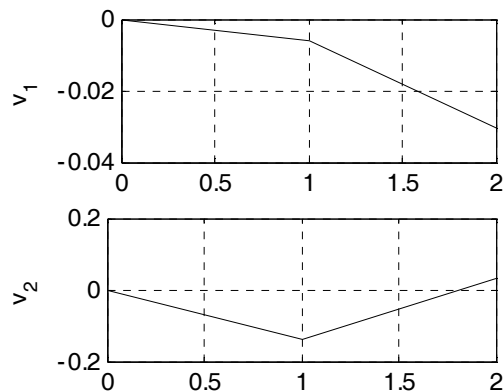
```

ome =
    1.3727
    6.4535

v1 =
   -0.0062
   -0.0303

v2 =
   -0.1413
    0.0297

```



Hình 9-29. Các dạng dao động riêng

Hệ dao động ba bậc tự do

Cho hệ dao động ba bậc tự do như trên hình 9-30. Các lò xo tuyến tính có độ cứng c_i , khối lượng các vật nặng là m_i , ($i = 1, 2, 3$). Gọi x_i là dịch chuyển của các khối lượng kể từ vị trí mà các lò xo không bị biến dạng. Áp dụng phương pháp tách vật (hoặc phương trình Lagrange loại 2) nhận được phương trình vi phân chuyển động ở dạng ma trận như sau:

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\mathbf{x} = \mathbf{p}, \quad \mathbf{x} = [x_1 \ x_2 \ x_3]^T$$

với

$$\mathbf{M} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} m_1 g \\ m_2 g \\ m_3 g \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} c_1 + c_2 + c_3 + c_5 & -c_3 & -c_5 \\ -c_3 & c_3 + c_4 & -c_4 \\ -c_5 & -c_4 & c_4 + c_5 \end{bmatrix},$$

Cho biết các thông số của hệ:

$$c_1 = c_2 = c_3 = 10 \text{ N/mm}, \quad c_4 = c_5 = 5 \text{ N/mm},$$

$$m_1 g = m_3 g = 98.1 \text{ N}, \quad m_2 g = m_3 g.$$

1. Tìm vị trí cân bằng tĩnh của hệ, bằng cách giải hệ phương trình đại số tuyến tính để tìm biến dạng của các lò xo

$$\mathbf{C}\mathbf{x} = \mathbf{p}.$$

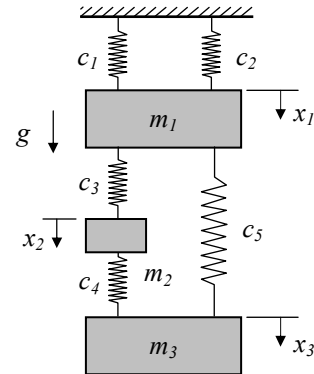
2. Tìm các tần số dao động riêng và các dạng dao động riêng của hệ từ phương trình dao động tự do

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\mathbf{x} = \mathbf{0}.$$

Bài toán được giải trong Matlab bằng một m-file như sau:

```
function Daodon_3DOF
m1 = 10; m2 = 5; m3 = m1; % kg
c1 = 10000; % N/m
c2 = c1; c3 = c1; c4 = 5000; c5 = c4;
g = 9.81; % m/s^2

M = diag([m1, m2, m3]);
C = [ c1+c2+c3+c5, -c3, -c5;
      -c3, c3+c4, -c4;
      -c5, -c4, c4+c5];
p = g*[m1; m2; m3];
x0 = inv(C)*p % tính do dãn tĩnh của các lò xo
n = size(M,1);
[V, D] = eig(C,M);
for i=1:n ome(i)=sqrt(D(i,i)); end
ome'
```



Hình 9-30

```

x = [0:1:n];
subplot(3,1,1)
plot(x,[0; V(:,1)], 'k-', 'linewidth',1), grid on, ylabel('v_1')
subplot(3,1,2)
plot(x,[0; V(:,2)], 'k-', 'linewidth',1), grid on, ylabel('v_2')
subplot(3,1,3)
plot(x,[0; V(:,3)], 'k-', 'linewidth',1), grid on, ylabel('v_3')

```

Độ dẫn tĩnh của lò xo

```

x0 =
    0.0123
    0.0201
    0.0260

```

Các tần số dao động riêng

```

ome =
    21.2572
    48.5851
    68.4662

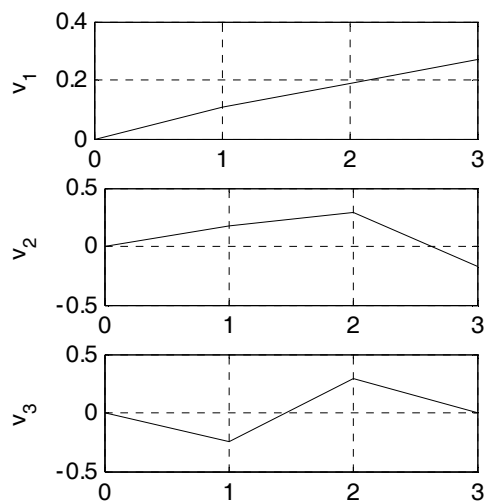
```

Các vectơ dạng riêng

```

v1 =
    0.1053
    0.1875
    0.2671
v2 =
    0.1756
    0.2847
   -0.1692
v3 =
   -0.2410
    0.2895
   -0.0066

```



Hình 9-31. Các dạng dao động riêng

9.6 Phân tích động học cơ cấu

Bài toán phân tích động học cơ cấu dựa trên các phương trình liên kết thường dẫn đến việc giải hệ phương trình đại số phi tuyến. Sau đây trình bày việc giải hệ phương trình đại số phi tuyến bằng phương pháp Newton-Raphson.

Bài toán: cần tìm nghiệm \mathbf{x} thỏa mãn hệ phương trình đại số phi tuyến

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \quad \mathbf{f} \in \mathbb{R}^n, \quad \mathbf{x} \in \mathbb{R}^n$$

hay viết dưới dạng hệ n phương trình

$$f_k(x_1, x_2, \dots, x_n) = 0, \quad k = 1, 2, \dots, n.$$

Để thiết lập các bước của bài tính, trước hết ta khai triển Taylor hàm $f(\mathbf{x})$ tại lân cận \mathbf{x}_0

$$\begin{aligned} f(\mathbf{x}_0 + \delta\mathbf{x}) &= f(\mathbf{x}_0) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}_0)\delta\mathbf{x} + O(\delta\mathbf{x})^2 \\ &= f(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0)\delta\mathbf{x} + O(\delta\mathbf{x})^2 \end{aligned}$$

với $\mathbf{J}(\mathbf{x}_0)$ là ma trận Jacobi được xác định tại điểm \mathbf{x}_0

$$\mathbf{J}(\mathbf{x}_0) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}(\mathbf{x}_0)$$

Giả sử \mathbf{x}_0 là một xấp xỉ ban đầu của nghiệm cần tìm và để $(\mathbf{x}_0 + \delta\mathbf{x})$ là một giá trị gần nghiệm hơn (chính xác hơn), ta cho $\mathbf{f}(\mathbf{x}_0 + \delta\mathbf{x}) = \mathbf{0}$. Từ đó nhận được một hệ phương trình đại số tuyến tính đối với $\delta\mathbf{x}$

$$\mathbf{J}(\mathbf{x}_0)\delta\mathbf{x} = -\mathbf{f}(\mathbf{x}_0) \text{ suy ra } \delta\mathbf{x} = -\mathbf{J}^{-1}(\mathbf{x}_0) \cdot \mathbf{f}(\mathbf{x}_0).$$

Lấy giá trị $\mathbf{x}_1 = \mathbf{x}_0 + \delta\mathbf{x}$ là xấp xỉ gần đúng mới của nghiệm. Lặp lại nhiều lần công việc trên, nếu phương pháp hội tụ sẽ cho ta nghiệm của hệ phương trình đại số phi tuyến.

Thuật giải

Bước 1. Khởi gán $k = 0$, chọn xấp xỉ ban đầu $\mathbf{x}^{(0)}$.

Bước 2. Tính $\mathbf{f}(\mathbf{x}^{(k)})$. Nếu $\|\mathbf{f}(\mathbf{x}^{(k)})\| < \varepsilon$ thì dừng, nếu không thì tiếp tục từ 3.

Bước 3. Tính ma trận Jacobi tại $\mathbf{x}^{(k)}$, tức là $\mathbf{J}(\mathbf{x}^{(k)})$.

Bước 4. Giải $\mathbf{J}(\mathbf{x}^{(k)})\delta\mathbf{x} = -\mathbf{f}(\mathbf{x}^{(k)})$ để tìm $\delta\mathbf{x}^{(k)}$.

Bước 5. Lấy $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}$.

Bước 6. Tăng k , $k = k + 1$. Nếu $k > M$ với M là số bước lặp đã chọn trước, thì dừng. Nếu không, tiếp tục từ Bước 2.

Trong thuật giải trên ta đã thoát ra khỏi vòng lặp trong hai tình huống. Khi không đạt được độ chính xác cần thiết, ta buộc phải kết thúc sau một số bước lặp nhất định (bước 6). Quá trình lặp sẽ kết thúc tốt nếu như ta đạt được độ chính xác chọn trước (bước 2). Chú ý rằng ở đây ta chọn điều kiện dừng bằng cách xét chuẩn Euclide của vector \mathbf{f} . Cũng có thể thay điều kiện này bằng các điều kiện khác như

$$\|\delta \mathbf{x}\| < \varepsilon.$$

Sự thành công của phương pháp phụ thuộc vào việc chọn xấp xỉ ban đầu. Nếu chọn được điểm xuất phát tốt, phương pháp hội tụ rất nhanh. Ngược lại, thì khó có thể đoán được và yêu cầu thuật toán phải dừng lại sau một số bước lặp nhất định.

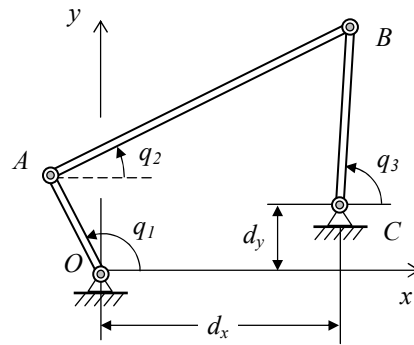
Khi áp dụng phương pháp Newton-Raphson để giải bài toán, ngoài việc nhập các hàm $f(\mathbf{x})$ vào, còn yêu cầu phải có ma trận Jacobi $\mathbf{J}(\mathbf{x})$. Trong trường hợp nếu việc đạo hàm $\partial f_i / \partial x_j$ gặp khó khăn, hoặc khó thực hiện được ta có thể sử dụng xấp xỉ sau đây để tính ma trận Jacobi

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(\mathbf{x} + \mathbf{e}_j h) - f_i(\mathbf{x})}{h}$$

với h là bước nhỏ và \mathbf{e}_j là vectơ đơn vị theo hướng của trục x_j .

Phân tích động học cơ cấu bốn khâu bản lề.

Cho cơ cấu 4 khâu bản lề OABC như hình 9-32. Các kích thước của cơ cấu $OA = l_2$, $AB = l_3$, $BC = l_4$, d_x và d_y . Hãy viết các phương trình liên kết cho cơ cấu, từ đó giải bài toán vị trí của cơ cấu bằng phương pháp Newton-Raphson. Cho biết $q_1 = q_{10} + \Omega t$, hãy xác định các góc $\varphi_3(t)$ và $\varphi_4(t)$ tại các thời điểm t_k , $k = 1, 2, 3, \dots$ ứng với các góc quay của thanh OA, $q_1 = q_{10} + \Omega t_k$. Cho biết các thông số của cơ cấu $l_1 = 0.20$, $l_2 = 0.60$, $l_3 = 0.40$, $d_x = 0.45$, $d_y = 0.12$ m.



Hình 9-32

Từ hình vẽ ta viết được các phương trình liên kết

$$f_1(q_1, q_2, q_3) = l_1 \cos q_1 + l_2 \cos q_2 - d_x - l_3 \cos q_3 = 0$$

$$f_2(q_1, q_2, q_3) = l_1 \sin q_1 + l_2 \sin q_2 - d_y - l_3 \sin q_3 = 0$$

Tính được các ma trận Jacobi

$$\mathbf{J}_{q_{23}} = \begin{bmatrix} -l_2 \sin q_2 & l_3 \sin q_3 \\ l_2 \cos q_2 & -l_3 \cos q_3 \end{bmatrix}, \quad \mathbf{J}_{q_1} = \begin{bmatrix} -l_1 \sin q_1 \\ l_1 \cos q_1 \end{bmatrix}$$

Đạo hàm các phương trình liên kết theo thời gian cho ta phương trình để giải bài toán vận tốc và gia tốc:

$$\mathbf{J}_{q_{23}} \dot{\mathbf{q}}_{23} + \mathbf{J}_{q_1} \dot{q}_1 = 0 \Rightarrow \dot{\mathbf{q}}_{23} = -\mathbf{J}_{q_{23}}^{-1} \mathbf{J}_{q_1} \dot{q}_1$$

$$\mathbf{J}_{q_{23}} \ddot{\mathbf{q}}_{23} + \dot{\mathbf{J}}_{q_{23}} \dot{\mathbf{q}}_{23} + \mathbf{J}_{q_1} \ddot{q}_1 + \dot{\mathbf{J}}_{q_1} \dot{q}_1 = 0 \Rightarrow \ddot{\mathbf{q}}_{23} = -\mathbf{J}_{q_{23}}^{-1} (\dot{\mathbf{J}}_{q_{23}} \dot{\mathbf{q}}_{23} + \mathbf{J}_{q_1} \ddot{q}_1 + \dot{\mathbf{J}}_{q_1} \dot{q}_1)$$

với

$$\mathbf{J}_{q_{23}} = \begin{bmatrix} -\dot{q}_2 l_2 \cos q_2 & \dot{q}_3 l_3 \cos q_3 \\ -\dot{q}_2 l_2 \sin q_2 & \dot{q}_3 l_3 \sin q_3 \end{bmatrix}, \quad \mathbf{J}_{q_1} = \begin{bmatrix} -l_1 \cos q_1 \\ -l_1 \sin q_1 \end{bmatrix} \dot{q}_1$$

Triển khai trong Matlab với các m-file

1. Các phương trình liên kết

```
function f = bkbl_ptlk(q1, q2, q3)
global l1 l2 l3 dx dy
f1 = l1*cos(q1) + l2*cos(q2) - l3*cos(q3) - dx;
f2 = l1*sin(q1) + l2*sin(q2) - l3*sin(q3) - dy;
f=[f1; f2];
```

2. Các ma trận Jacobi

```
function [J1, J23] = bkbl_Jacob(q1,q2,q3)
global l1 l2 l3 dx dy
J1 = [-l1*sin(q1); l1*cos(q1)];
J23 = [-l2*sin(q2), l3*sin(q3);
       l2*cos(q2), -l3*cos(q3)];
```

3. Đạo hàm của ma trận Jacobi

```
function [dJ1, dJ23] = bkbl_Jacdot(q1,q2,q3, dq1, dq2,dq3)
global l1 l2 l3 dx dy
dJ1 = -l1*dq1*[cos(q1); sin(q1)];
dJ23 = [-dq2*l2*cos(q2), dq3*l3*cos(q3);
        -dq3*l2*sin(q2), dq3*l3*sin(q3)];
```

4. Chương trình chính

```
function bkbl_main
global l1 l2 l3 dx dy
l1 = 0.12; l2 = 0.70; l3 = 0.40; dx = 0.50; dy = 0.10;
%cho goc cua tay quay OA
q10 = 0.5; % rad
omega = 1.0; % rad/s
t = linspace(0, 5*pi/omega, 360);
a(1) = 0.2; b(1) = 0.542; % chon xap xi ban dau cho q2, q3
for i=1:length(t)
    q1(i)= q10+omega*t(i);
    dq1(i) = omega; ddq1(i) = 0;
    f = ptlk_bkbl(q1(i), a(1), b(1));
    k = 1;
    while (norm(f,2) > 1.0e-10 && k < 30)
        [J1, J23] = Jacob_bkbl(q1(i), a(1), b(1));
        deltax = inv(J23)*(-f);
        a(2) = a(1)+deltax(1);
```

```

        b(2) = b(1)+deltax(2);
        a(1) = a(2);      b(1) = b(2);
        f = ptlk_bkbl(q1(i), a(1), b(1));
        k = k+1;
    end
    q2(i)=a(1);      q3(i)=b(1);
    % van toc
    [J1, J23] = Jacob_bkbl(q1(i), q2(i), q3(i));
    dq23 = -inv(J23)*J1*dq1(i);
    dq2(i) = dq23(1);      dq3(i) = dq23(2);
    % gia toc
    [dJ1, dJ23] =
        Jacdot_bkbl(q1(i),q2(i),q3(i),dq1(i),dq2(i),dq3(i));
    ddq23 = -inv(J23)*(dJ23*dq23 + J1*ddq1(i) + dJ1*dq1(i));
    ddq2(i) = ddq23(1);      ddq3(i) = ddq23(2);
end

figure(1)
plot(t,q2,'k-', t, dq2,'k--', 'linewidth',1), grid on
xlabel('t [s]'), legend('q_2', 'dq_2')
axis([0 max(t) -0.5 1.1])
figure(2)
plot(t,q3,'k-', t, dq3,'k--', 'linewidth',1), grid on
xlabel('t [s]'), legend('q_3', 'dq_3')
axis([0 max(t) -0.8 2.2])

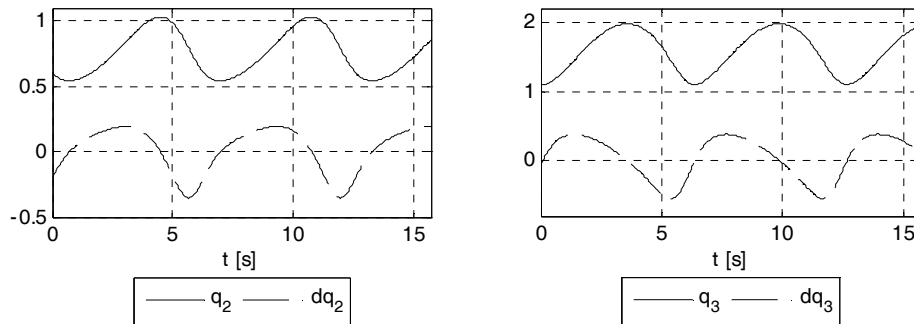
x0 = 0; y0 = 0; r = 0.005;
figure(3), hold on
plot(x0,y0,'o-','linewidth',1), plot(dx,dy,'o-','linewidth',1)
for i = 1:5:length(t)/2
    x1(i) = l1*cos(q1(i));      x2(i) = x1(i) + l2*cos(q2(i));
    y1(i) = l1*sin(q1(i));      y2(i) = y1(i) + l2*sin(q2(i));
    x4(i) = dx;                  x3(i) = dx + l3*cos(q3(i));
    y4(i) = dy;                  y3(i) = dy + l3*sin(q3(i));

    x_day = [x0, x1(i), x2(i), x3(i), x4(i), x0];
    y_day = [y0, y1(i), y2(i), y3(i), y4(i), y0];

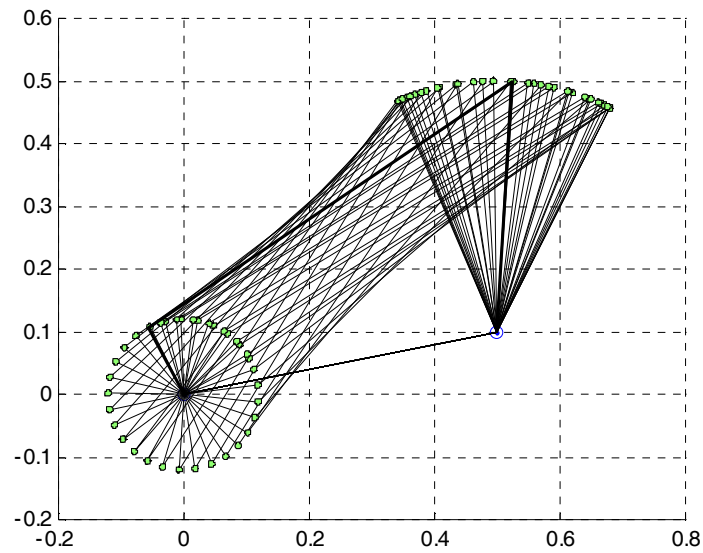
    plot(x_day, y_day, 'k-','linewidth',1), grid on, axis equal
    tam1 = [x1(i), y1(i)];      vehinhtron(tam1,r);
    tam2 = [x2(i), y2(i)];      vehinhtron(tam2,r);
end
axis([-0.2 0.8 -0.2 0.6])

```

Đồ thị của các góc q_2, q_3 cùng các đạo hàm \dot{q}_2, \dot{q}_3 được thể hiện như trên hình 9-33. Cấu hình của cơ cấu tại một số vị trí được chỉ ra trên hình 9-34.



Hình 9-33. Đồ thị các góc q_2 và q_3 theo thời gian



Hình 9-34. Cấu hình cơ cấu bốn khâu bản lề

9.7 Bài toán động học ngược robot

Trong bài toán động học ngược robot, ta cần xác định các góc khớp – hay tọa độ khớp – khi muốn bàn kẹp của robot ở một vị trí mong muốn. Trong phần này ta xét một tay máy hai bậc tự do phẳng có sơ đồ như trên hình 9-35. Cần tìm các tọa độ suy rộng q_1, q_2 để bàn kẹp E ở một vị trí mong muốn cho trước x_E, y_E . Biết chiều dài các khâu là $OA = L_1 = 0.50 \text{ m}$, $AE = L_2 = 0.70 \text{ m}$.

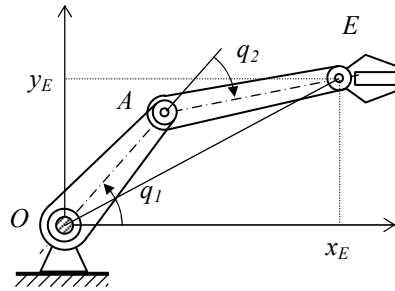
Đối với bài toán này, robot có cấu trúc khá đơn giản, ta có thể giải bằng phương pháp hình học, bằng cách giải tam giác OAE khi biết chiều dài ba cạnh của nó.

Ở đây ta sẽ sử dụng phương pháp giải hệ phương trình phi tuyến, để giải bài toán này, vì lẽ phương pháp có thể được mở rộng dễ dàng cho các rôbot có số bậc tự do lớn hơn và có cấu trúc phức tạp hơn.

Mối liên hệ giữa tọa độ điểm E với các tọa độ suy rộng được xác định bởi

$$x_E = l_1 \cos q_1 + l_2 \cos(q_1 - q_2)$$

$$y_E = l_1 \sin q_1 + l_2 \sin(q_1 - q_2)$$



Hình 9-35

Đây là một hệ hai phương trình phi tuyến đối

với hai ẩn q_1, q_2 . Phương pháp Newton-Raphson được sử dụng để giải hệ này, với ma trận Jacobi như sau

$$\mathbf{J}_q = \begin{bmatrix} -l_1 \sin q_1 & l_2 \sin(q_1 - q_2) \\ l_1 \cos q_1 & -l_2 \cos(q_1 - q_2) \end{bmatrix}$$

Triển khai trong Matlab với các m-file sau:

1. Phương trình liên kết

```
function f = taymay2dof_bkbl(q1, q2, xE, yE)
global l1 l2
f1 = l1*cos(q1) + l2*cos(q1 - q2) - xE;
f2 = l1*sin(q1) + l2*sin(q1 - q2) - yE;
f=[f1; f2];
```

2. Ma trận Jacobi

```
function Jq = taymay2dof_Jacob(q1, q2)
global l1 l2
Jq = [-l1*sin(q1), l2*sin(q1 - q2);
      l1*cos(q1), -l2*cos(q1 - q2)];
```

3. Chương trình chính

```
function taymay2dof_main
global l1 l2
l1 = 0.50; l2 = 0.70;
n = 20; % so diem chia
xE = linspace(1.1, 0.1, n);
yE = linspace(0.1, 1.1, n);

a(1) = 0.5; b(1) = 0.2; % chon xap xi ban dau cho q1, q2
for i=1:n
    f = taymay2dof_ptlk(a(1), b(1), xE(i), yE(i));
    k = 1;
    while (norm(f,2) > 1.0e-10 && k < 30)
```



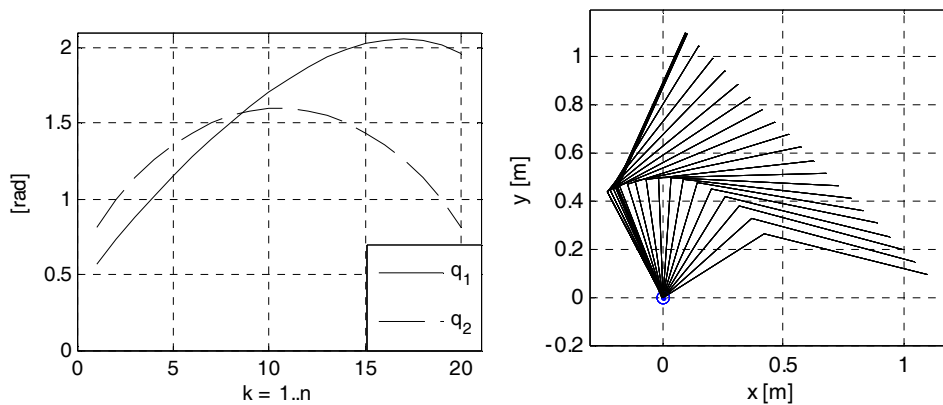
```

Jq = taymay2dof_Jacob(a(1), b(1));
dq = inv(Jq)*(-f);
a(2) = a(1)+dq(1);          b(2) = b(1)+dq(2);
a(1) = a(2);                b(1) = b(2);
f = taymay2dof_ptlk(a(1), b(1), xE(i), yE(i));
k = k+1;
end
q1(i) = a(1);    q2(i) = b(1);
end
k = 1:1:n;
figure(1)
plot(k, q1, 'k-', k, q2, 'k--', 'linewidth',1), grid on
legend('q_1', 'q_2'), xlabel('k = 1..n'), ylabel('[rad]')
axis([0 n+1 0 2.1])

figure(2)
x0 = 0; y0 = 0; plot(x0,y0,'o-', 'linewidth',1), hold on
for i = 1:n
    xA(i) = l1*cos(q1(i)); xE(i) = xA(i)+l2*cos(q1(i)-q2(i));
    yA(i) = l1*sin(q1(i)); yE(i) = yA(i)+l2*sin(q1(i)-q2(i));
    x_day = [x0, xA(i), xE(i)];
    y_day = [y0, yA(i), yE(i)];
    plot(x_day, y_day, 'k-', 'linewidth',1), grid on, axis equal
end

```

Chạy chương trình cho ta các kết quả như trong bảng 9-5. Đồ thị của các góc khớp và cấu hình tương ứng của rôbốt được đưa ra như trên hình 9-36.



Hình 9-36. Đồ thị các tọa độ q_1, q_2 và cấu hình của rôbốt

Bảng 9-5. Kết quả tính toán bài toán động học ngược

k	Cho trước		Giải tìm được	
	x_E [m]	y_E [m]	q_1 [rad]	q_2 [rad]
1	1.1000	0.1000	0.5701	0.8152
2	1.0474	0.1526	0.7334	0.9965
3	0.9947	0.2053	0.8806	1.1411
4	0.9421	0.2579	1.0181	1.2600
5	0.8895	0.3105	1.1488	1.3584
6	0.8368	0.3632	1.2738	1.4387
7	0.7842	0.4158	1.3931	1.5024
8	0.7316	0.4684	1.5063	1.5499
9	0.6789	0.5211	1.6123	1.5816
10	0.6263	0.5737	1.7098	1.5974
11	0.5737	0.6263	1.7974	1.5974
12	0.5211	0.6789	1.8739	1.5816
13	0.4684	0.7316	1.9380	1.5499
14	0.4158	0.7842	1.9889	1.5024
15	0.3632	0.8368	2.0257	1.4387
16	0.3105	0.8895	2.0478	1.3584
17	0.2579	0.9421	2.0545	1.2600
18	0.2053	0.9947	2.0444	1.1411
19	0.1526	1.0474	2.0148	0.9965
20	0.1000	1.1000	1.9596	0.8152

Tài liệu tham khảo

- [1] Ottmar Beucher: *Matlab und Simulink: Grundlegende Einführung für Studenten und Ingenieure in der Praxis*. Pearson Studium Verlag; Auflage 3 (2006).
- [2] Helmut Bode: *MATLAB-Simulink Analyse und Simulation dynamischer Systeme*. Springer Verlag, 2006.
- [3] Jörg Kahlert: *Simulation technischer Systeme: Eine beispielorientierte Einführung*. Vieweg & Sohn-Verlag, Wiesbaden, 2004.
- [4] Jaan Kiusalaas: *Numerical Methods in Engineering with MATLAB*. Cambridge University Press, 2005.
- [5] David McMahon: *MATLAB Demystified - A Self-Teaching Guide*. McGraw-Hill, 2007.
- [6] John H. Mathews, Kurtis D. Fink: *Numerical methods using MATLAB*. 3. Ed. Prentice Hall 1999.
- [7] Dieter Pietruszka: *MATLAB in der Ingenieurpraxis. Modellbildung, Berechnung und Simulation*. 2. Ed. Vieweg+Teubner Verlag, 2006.
- [8] L.F. Shampine, I. Gladwell, S. Thomson: *Solving ODEs with Matlab*. Cambridge University Press, 2003.
- [9] Won Young Yang, Wenwu Cao, Tae-Sang Chung, John Morris: *Applied Numerical Methods Using Matlab*. John Wiley & Sons, Inc., 2005.
- [10] D. Gross, W. Hauger, W. Schnell, J. Schröder: *Technische Mechanik, Band 1: Statik*. (8. Auflage). Springer - Verlag, Berlin 2004.
- [11] D. Gross, W. Hauger, W. Schnell, J. Schröder: *Technische Mechanik, Band 2: Elastostatik*. (8. Auflage). Springer -Verlag, Berlin 2005.
- [12] D. Gross, W. Hauger, W. Schnell, J. Schröder: *Technische Mechanik, Band 3: Kinetik*. (8. Auflage). Springer -Verlag, Berlin 2004.
- [13] Dankert/Dankert: *Technische Mechanik: Statik, Festigkeitslehre, Kinematik/Kinetik*. (3. Auflage). Teubner B.G. 2004.
- [14] R.C. Hibbele: *Engineering Mechanics –Strength of Materials* (10.Edition). Prentice Hall, Upper Saddle River , New Jersey 2004.
- [15] Nguyễn Hoàng Hải. Nguyễn Việt Anh: *Lập trình Matlab và ứng dụng*. NXB Khoa học và Kỹ thuật, Hà Nội 2006.
- [16] Nguyễn Văn Khang: *Dao động kỹ thuật (in lần 4)*. NXB Khoa học và Kỹ thuật, Hà Nội 2005.
- [17] Đinh Văn Phong: *Phương pháp số trong cơ học*. Nhà xuất bản Khoa học và Kỹ thuật, 1999.

- [18] Nguyễn Phùng Quang: *Matlab & Simulink dành cho kỹ sư điều khiển tự động*. NXB Khoa học và Kỹ thuật, Hà Nội 2006.
- [19] Đỗ Sanh: *Cơ học kỹ thuật, tập 1 và tập 2*. Nhà xuất bản Giáo dục, Hà nội 2008.
- [20] Phạm Thị Ngọc Yến, Ngô Hữu Tình, Lê Tấn Hùng và Nguyễn Thị Lan Hương: *Cơ sở Matlab & ứng dụng*. NXB Khoa học và Kỹ thuật, Hà Nội, 2007.
- [21] Lê Quang Minh, Nguyễn Văn Vượng: *Sức bền vật liệu, Tập 1,2,3*. NXB Giáo dục, Hà nội 1997.
- [22] Đặng Việt Cường, Nguyễn Nhật Thăng, Nhữ Phương Mai: *Sức bền vật liệu, tập 1 và 2*. NXB Khoa Học và Kỹ Thuật, 2002.
- [23] Phạm Minh Hoàng: *Maple và các bài toán ứng dụng*. NXB Khoa học và Kỹ thuật, 2008.