

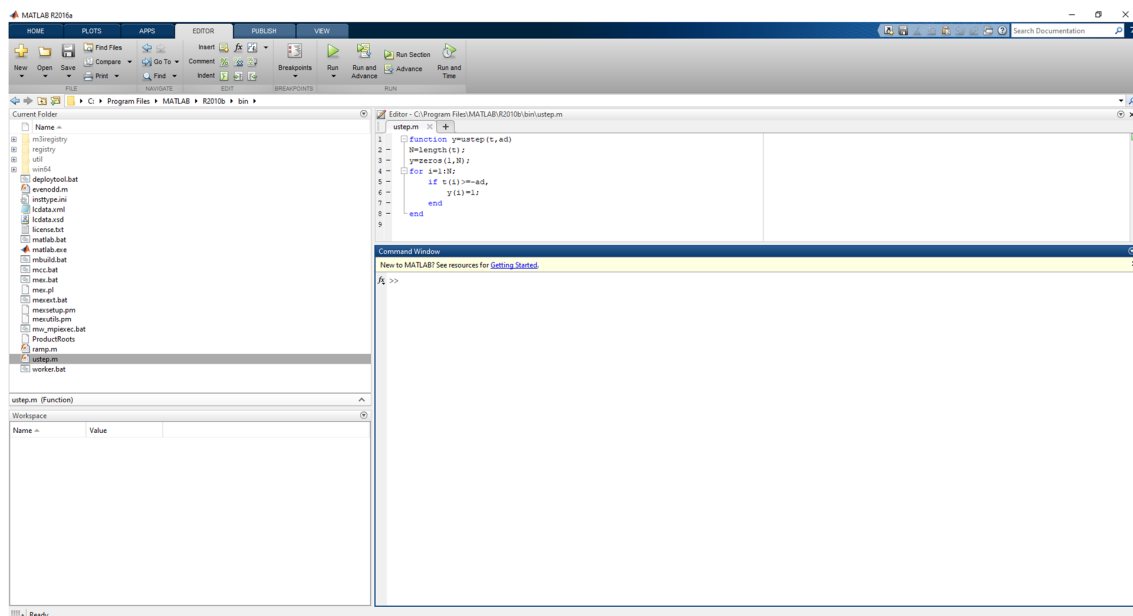
Phần 1. Giới thiệu chung về Matlab

I. Giới thiệu về phần mềm Matlab

- Matlab là gì?
- + Matrix Laboratory
- + Là ngôn ngữ lập trình bậc cao cho các ngành khoa học và kỹ thuật.
- + Được phát triển bởi Mathwork Inc.
- Khả năng của Matlab:
 - + Tính toán khoa học (một công cụ tính toán rất mạnh)
 - + Dụng cụ vẽ đồ thị hàm số.
 - + Thực hiện các thuật toán.
 - + Mô phỏng các hệ thống khác nhau.
 - + Giao tiếp được với các phần mềm được viết trên các ngôn ngữ khác (C, C++, Java, Fortran)
- Các lĩnh vực ứng dụng:
 - + Kỹ thuật
 - + Khoa học
 - + Y sinh
 - + Kinh tế
 - + ...

II. Thông tin căn bản:

- Giao diện phần mềm



- Dấu nhắc lệnh của Matlab: >>
 - Bạn có thể viết các lệnh Matlab sau dấu nhắc lệnh này.
 - Ví dụ về sử dụng Matlab để tính toán:
-

```
>> 3+2*5
ans =
13
>> sqrt(1+6*8)-3^2
ans =
-2
>> 48/(6+2)
ans =
6
>>
```

- Các kí hiệu phép toán trong Matlab

Kí hiệu	Phép toán	Ví dụ
+	Cộng	$2 + 3$
-	Trừ	$2 - 3$
*	Nhân	$2 * 3$
/	Chia	$2 / 3$

- Biến trong Matlab: Chúng ta có thể gán giá trị cho các biến trong Matlab

+ Ví dụ:

```
>> x = 3
x =
3
>> y = 5
y =
5
>> x+y*2
ans=
13
>>
```

- Dấu chấm phẩy: Thêm dấu chấm phẩy vào cuối một câu lệnh sẽ ngừng việc hiển thị ra màn hình kết quả của nó.

+ Ví dụ:

```
>> x = 3;
>> y = 2;
>> z = (x^2-1)/y;
>> z
z =
4
>>
```

- Biến ans: Nếu chúng ta không gán kết quả của một biểu thức cho một biến nào đó, nó sẽ được lưu vào biến ans một cách mặc định.

- Một số hàm cơ bản trong Matlab:

cos(x)	Hàm tính cos	abs(x)	Hàm lấy giá trị tuyệt đối
sin(x)	Hàm tính sin	sign(x)	Hàm xét dấu
tan(x)	Hàm tính tan	max(x)	Hàm tìm giá trị lớn nhất
acos(x)	Hàm tính arccos	min(x)	Hàm tìm giá trị nhỏ nhất
asin(x)	Hàm tính arcsin	ceil(x)	Hàm làm tròn lên
atan(x)	Hàm tính arctan	floor(x)	Hàm làm tròn xuống
exp(x)	Hàm lũy thừa e	rem(x)	Lấy phần dư sau khi thực hiện phép chia
sqrt(x)	Hàm căn bậc hai	angle(x)	Hàm lấy góc pha
log(x)	Hàm logarit tự nhiên	conj(x)	Hàm lấy liên hợp
log10(x)	Hàm logarit cơ số 10		

- Có thể sử dụng lệnh help command để lấy thông tin chi tiết về từng hàm.

- Hằng số:

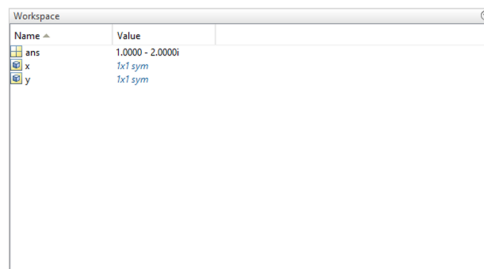
pi	Số π , $\pi = 3.14159\dots$
i , j	Đơn vị ảo i , $\sqrt{-1}$
Inf	Vô cùng, ∞
NaN	Not a number

+ Nên tránh sử dụng i hoặc j làm biến.

+ Ví dụ:

```
> clear all
>> (3*i)^2
ans =
-9
>> i = 2; % nên tránh sử dụng như này bởi vì i đã được sử dụng
      % mặc định làm hằng số.
>> (3*i)^2
ans=
36
>>
```

- Matlab workspace



Name	Value
ans	1.0000 - 2.0000i
x	1x1 sym
y	1x1 sym

- + Là tập hợp các biến đã được định nghĩa.
- + Có thể sử dụng lệnh `who` để hiển thị tất cả các biến đang có.

Ví dụ:

```
>> x = 3;
>> y = 5;
>> z = x^2 + y;
>> sqrt(16)/2;
>> who
Your variables are:
ans x      y
>>
```

- + Dùng lệnh `clear` để xóa một phần hoặc tất cả workspace

Ví dụ:

```
>> x = 3;
>> y = 5;
>> z = x^2 + y;
>> who
Your variables are:
x      y      z
>> clear x
>> who
Your variables are:
ans y      z
>> clear
>> who
>> y+1
Undefined function or variable 'y'. % thông báo lỗi
```

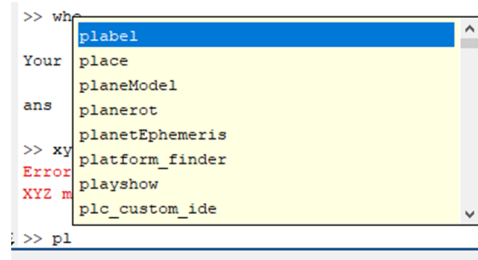
- Lịch sử lệnh:

- + Khi bạn bắt đầu viết một câu lệnh, có thể sử dụng nút mũi tên lên và xuống trên bàn phím để gọi lại các lệnh đã được nhập vào từ lúc trước.
- + Một khi mà lệnh đã được gọi lại, bạn có thể sử dụng mũi tên sang trái và sang phải để chỉnh sửa câu lệnh.

- Hỗ trợ hoàn thiện câu lệnh:

+ Sau khi viết vài kí tự, bạn có thể sử dụng nút Tab trên bàn phím để hiện ra các lệnh và biến bắt đầu với các kí tự đó.

+ Ví dụ: nhập vào p1 sau đó nhấn Tab sẽ hiện ra như hình.



- Lệnh help

+ Cấu trúc:

help command_name

+ Ví dụ:

```
>> help sqrt
```

```
sqrt    Square root.
```

```
sqrt(X) is the square root of the elements of X. Complex  
results are produced if X is not positive.
```

```
See also sqrtm, realsqrt, hypot.
```

```
Reference page for sqrt
```

```
Other functions named sqrt
```

- Lệnh lookfor

+ Cấu trúc:

```
lookfor keyword
```

+ Lệnh này sẽ trả lại kết quả là các lệnh có từ keyword ở phần đầu tiên trong file trợ giúp.

+ Ví dụ: tìm các lệnh để chơi nhạc

```
>> lookfor audio
```

audioplayer	- Audio player object.
audiorecorder	- Audio recorder object.
audioOscillator	- Generate tunable audio waves
audioPluginInterface	- Specify the interface of the
generated audio plugin	
audioPluginParameter	- Specify one parameter of a
generated audio plugin	
audioTestBench	- Audio Test Bench for audio plug-
in MATLAB classes.	
crossoverFilter	- Multiband audio crossover filter

```

audioBrokenLinksMapping      - Broken links restoration mapping
for Audio System Toolbox blocks
audioRelinkFigure            - : Restore the figure handle to
the block's user data
audioUnlinkFigure            - : Remove the figure handle from
the block's user data
audio_links                  - Display and return library link
information
...

```

III. Vector và ma trận:

Vector và ma trận là nền móng của Matlab.

- Matlab được thiết kế đặc biệt cho các phép toán vectơ và ma trận.
- Các phép toán trực tiếp với vector và ma trận thường nhanh hơn nhiều so với sử dụng các cấu trúc vòng lặp vô hướng.
- Hãy cố gắng sử dụng vector và ma trận nhiều nhất có thể và hạn chế các cấu trúc vòng lặp (vì nó khá chậm ở Matlab).

1. Vector.

- Định nghĩa vector hàng: (sử dụng `,` để ngăn cách)

```

>> a = [3, 1, 2, 8]           % [ ] chỉ ra đây là 1 vector.
a =
    3     1     2     8

```

- Định nghĩa vector cột: (sử dụng `;` để ngăn cách)

```

>> b = [3;1;2;8]
b =
     3
     1
     2
     8

```

- Phép chuyển vị: `.'`

```

>> b.'
ans =
     3     1     2     8

```

- Các cách định nghĩa vectơ:

+ Bắt đầu tại 1, kết thúc tại 10 với bước là 2.

```

>> c = [1:2:10]               % ngoặc vuông là không bắt buộc
c =
     1     3     5     7     9

```

+ Bắt đầu tại 5, kết thúc tại 8, với bước là 1.

```

>> d = 5:8

```

```
d =  
    5      6      7      8
```

+ Bắt đầu tại 6, kết thúc tại 0, với bước là -2.

```
>> e = [6:-2:0]
```

```
e =  
    6      4      2      0
```

+ Tạo 5 điểm cách đều nhau giữa 2 và 20.

```
>> linspace(2, 20, 5)
```

```
ans =  
    2.0000    6.5000   11.0000   15.5000   20.0000
```

- Lấy thông tin về một vectơ:

Coi a = [3 1 2 8].

+ Lấy phần tử thứ 3 của một vectơ:

```
>> a(3)
```

```
ans =
```

```
    2
```

+ Phần tử thứ 2, 3, 4.

```
>> a(2:4)
```

```
ans =
```

```
    1      2      8
```

+ Phần tử thứ 2 và 4.

```
>> a(2:2:4)
```

```
ans =
```

```
    1      8
```

+ Phần tử thứ 1 và 4.

```
>> a([1,4])
```

```
ans =
```

```
    3      8
```

2. Ma trận:

- Định nghĩa một ma trận 2×3 :

```
>> a = [2, 5, 7; 1, 4, 9]
```

```
a =
```

```
    2      5      7
```

```
    1      4      9
```

- Chuyển vị ma trận: .'

```
>> a.'
```

```
ans =
```

```
    2      1
```

```

5      4
7      9

```

- Lấy phần tử nằm ở hàng thứ nhất, cột thứ ba.

```

>> a(1, 3)
ans =
7

```

- Lấy hàng đầu tiên.

```

>> a(1, :)
ans =
2      5      7

```

- Lấy cột thứ hai.

```

>> a(:, 2)
ans =
5
4

```

- Lấy phần tử thứ 2 và 3 ở hàng thứ 2.

```

>> a(2, [2, 3])
ans =
4      9

```

- Lấy các phần tử ở cột 1 và cột 3.

```

>> a(:, [1, 3])
ans =
2      7
1      9

```

- Tạo một ma trận chứa hàng 2 và hàng 3, cột 1 và cột 3 của một ma trận có sẵn.

$$b = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```

>> b([2, 3], [1, 3])
ans =
4      6
7      9

```

- Đổi chỗ cột 1 và cột 3

```

>> b(:, [3, 2, 1])
ans =
3      2      1
6      5      4
9      8      7

```


- Xây dựng một ma trận với các vector hàng:

```
>> v1 = [1 3 5];  
>> v2 = [2 4 6];  
>> m = [v1; v2]  
m =  
     1     3     5  
     2     4     6
```

- Xây dựng một ma trận với các vector cột:

```
>> c1 = [1; 2];  
>> c2 = [3; 4];  
>> c3 = [5; 6];  
>> m2 = [c1, c2, c3]  
m2 =  
     1     3     5  
     2     4     6
```

- Các phép tính với ma trận:

Cho các ma trận sau: $m1 = \begin{bmatrix} 2 & 5 & 7 \\ 1 & 4 & 9 \end{bmatrix}$, $m2 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, $v1 = \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}$, $v2 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

+ Phép cộng và phép trừ (các ma trận phải cùng kích thước)

```
>> m1-m2
```

```
ans =
```

```
     1     3     4  
    -3    -1     3
```

```
>> v1+v2
```

```
Error using +
```

```
Matrix dimensions must agree. % thông báo lỗi do các ma trận không  
                               % có cùng kích thước
```

```
>> v1+v2.'
```

```
ans =
```

```
     8    10    12
```

+ Phép nhân từng phần tử ở cùng vị trí: .*

```
>> m1.*m2
```

```
ans =
```

```
     2    10    21  
     4    20    54
```

+ Phép chia các phần tử ở cùng vị trí: ./

```
>> v1./v2
```

```

Error using ./
Matrix dimensions must agree.
>> v1./v2.'
ans =
    7    4    3
+ Phép nhân ma trận: *
>> m1*m2
Error using *
Inner matrix dimensions must agree.
% lỗi do 2 ma trận không có kích thước thỏa mãn
>> m1*m3
ans =
    31    30    23
    31    33    20
+ Tìm ma trận nghịch đảo
>> inv(m3)
ans =
   -0.2778    0.3889    0.0556
    0.0556   -0.2778    0.3889
    0.3889    0.0556   -0.2778
+ Cộng vectơ v1 vào hàng 2 của ma trận m1, sau đó gán kết quả vào một ma trận mới mat1
>> mat1 = m1;
>> mat1(2, :) = mat1(2, :) + v1
mat1 =
     2     5     7
     8    12    18
+ Thay thế cột thứ 3 của ma trận m2 với cột thứ nhất của ma trận m1, sau đó gán kết quả vào
một ma trận mới mat2.
>> mat2 = m2;
>> mat2(:, 3) = m1(:, 1)
mat2 =
     1     2     2
     4     5     1
- Các phép tính với đối tượng vô hướng:
>> m1+2
ans =
     4     7     9
     3     6    11

```

```
>> m1*2
ans =
     4     10     14
     2      8     18
>> m1/2
ans =
     1.0000     2.5000     3.5000
     0.5000     2.0000     4.5000
```

- Phép chuyển vị (.'') và chuyển vị phức ('')

$$m4 = \begin{bmatrix} 2+i & 5+2i & 7+3i \\ 1 & 4 & 9 \end{bmatrix}$$

+ Phép chuyển vị: .'

```
>> m4.'
ans =
     2.0000 + 1.0000i     1.0000
     5.0000 + 2.0000i     4.0000
     7.0000 + 3.0000i     9.0000
```

+ Phép chuyển vị phức: ''

```
>> m4''
ans =
     2.0000 -1.0000i     1.0000
     5.0000 -2.0000i     4.0000
     7.0000 -3.0000i     9.0000
```

- Phép tính logic.

$$m1 = \begin{bmatrix} 2 & 5 & 7 \\ 1 & 4 & 9 \end{bmatrix}, m2 = \begin{bmatrix} 2 & 2 & 3 \\ 4 & 5 & 9 \end{bmatrix}$$

Ví dụ: lấy tất cả các phần tử ở m1 mà > 4.

```
>> logic_mat = (m1 > 4)
logic_mat =
     0      1      1
     0      0      1
>> m1(logic_mat)
ans =
     5
     7
     9
```

Cách khác:

```
>> m1(m1>4)
ans =
     5
     7
     9
```

- Kích thước ma trận: `size()`

$$m1 = \begin{bmatrix} 2 & 5 & 7 \\ 1 & 4 & 9 \end{bmatrix}$$

```
>> size(m1)
ans =
     2     3
>> [m,n] = size(m1)
m =
     2
n =
     3
```

- Đường chéo của một ma trận:

$$m1 = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}, v1 = \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}$$

+ `diag(a)`: trả lại đường chéo chính của matrix a.

```
>> diag(m3)
ans =
     1
     1
     1
```

+ `diag(v1)`: tạo một ma trận đường chéo nhận véc tơ v1 làm đường chéo của nó.

```
>> diag(v1)
ans =
     7     0     0
     0     8     0
     0     0     9
```

- Tạo các ma trận đặc biệt

+ Ma trận “1”:

```
>> ones(2, 3)
ans =
     1     1     1
```

```
1      1      1
```

+ Ma trận “0”:

```
>> zeros(1, 2)
ans =
0      0
```

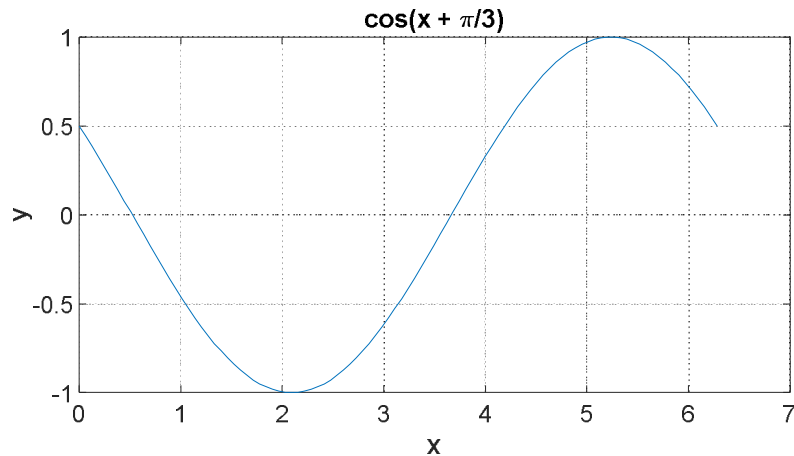
+ Ma trận đơn vị:

```
>> eye(3)
ans =
1      0      0
0      1      0
0      0      1
```

IV. Vẽ đồ thị:

- Vẽ đồ thị đơn giản:

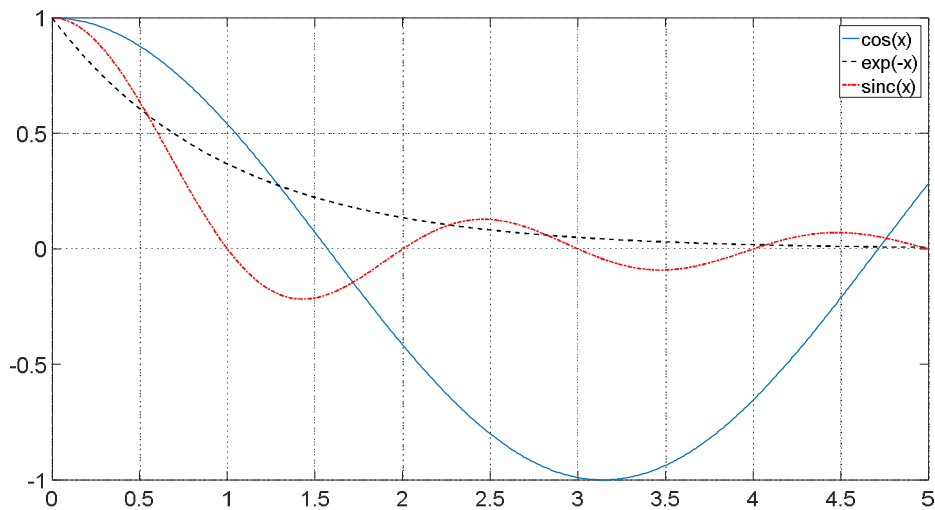
```
>> x = [1 2 3 4 5];
>> y = [3 -1 2 -3 -4]; % véctơ x và y phải cùng độ dài.
>> plot(x, y)
>> x = linspace(0, 2*pi, 100);
>> y = cos(x + pi/3);
>> plot(x, y)
>> xlabel('x'); % đặt một nhãn cho trục x
>> ylabel('y'); % đặt một nhãn cho trục y
>> title('cos(x + \pi/3)'); % đặt tiêu đề cho đồ thị
>> set(gca, 'fontsize', 24); % đặt cỡ chữ
>> grid on; % bật lưới
>> print -djpegfig.jpg; % lưu hình vào một file.
```



- Vẽ nhiều đường cong trên cùng một đồ thị: sử dụng lệnh “hold on”

```
>> x = [0:0.01:5];
>> y1 = cos(x);
>> y2 = exp(-x);
```

```
>> y3 = sinc(x);
>> plot(x, y1, 'linewidth', 2);
>> hold on;
>> plot(x, y2, 'k--', 'linewidth', 2); % đen, gạch đứt
>> plot(x, y3, 'r-.', 'linewidth', 2); % đỏ, gạch chấm
>> legend('cos(x)', 'exp(-x)', 'sinc(x)'); % thêm chú thích
>> set(gca, 'fontsize', 24);
```



- Sử dụng lệnh “help plot” để tìm thêm nhiều loại đường thẳng và màu sắc.

- Căn chỉnh một đồ thị: sử dụng lệnh axis

```
>> x = [-20:0.01:20];
>> plot(x, sinc(x))
>> axis([-5, 5 -0.3 1]); % x range: [-0.5, 0.5], y-range:
                                                                    [-0.3, 1]
```

- Tắt một hoặc nhiều đồ thị.

```
+ figure(1); % make figure 1 the current figure
```

```
+ close; % close the current figure
```

```
+ close all; % close all figures
```

V. Sử dụng M-file:

- Chúng ta có thể viết một dãy các lệnh Matlab trong một file bên ngoài với phần mở rộng là *.m. Những file này thường được gọi là m-file.

- Thực thi m-file sẽ thực thi tất cả các lệnh trong file.

1. Tạo một script file:

- Trong cửa sổ Matlab, nhấn vào File → New → Script để mở trình biên dịch mặc định.

- Trong cửa sổ biên dịch, nhập vào chuỗi lệnh sau:

```
x = [0:0.01:10];
y1 = cos(x);
```

```

y2 = exp(-x);
plot(x, y1);
hold on;
plot(x, y1+y2, 'r--');

```

- Lưu file vừa tạo lại bằng cách nhấn vào File → Save → test.m trong cửa sổ biên dịch.
- Thực thi file vừa tạo bằng cách nhấn vào Debug → Run test.m trong cửa sổ biên dịch.

2. Chạy thử m-file:

- Chuyển sang cửa sổ command window.
- Tại dấu nhắc lệnh, nhập:

```
>> test % chú ý rằng không ghi thêm .m vào câu lệnh.
```

- Quan trọng: m-file được thực thi phải nằm trong thư mục làm việc hiện tại.

```

>> pwd % hiển thị thư mục làm việc hiện tại
ans=
c:\skydrive\teaching\ELEG3124\Matlab
>> ls *.m % liệt kê tất cả các m-file nằm trong thư mục
                                làm việc hiện tại

test.m

```

3. Viết hàm:

- Một hàm có thể được định nghĩa và lưu vào một m-file riêng biệt.
- Ví dụ:

```

function y = average(x)
% function y = average(x)
% compute the average of a vector x, and return the value to y
N_element = length(x);
y = sum(x)/N_element;

```

- Buộc phải bắt đầu với function
- Lưu vào một m-file: trong cửa sổ editor, nhấn vào File → Save → average.m (tên của m-file phải là tên của hàm).
- Những chú thích sau tiêu đề của function sẽ được hiển thị khi bạn nhập lệnh help average vào dòng lệnh.

4. Gọi hàm:

- Trong cửa sổ command window, nhập vào:

```

>> x = 1:10;
>> y = average(x)
>> z = sqrt(x);
>> average(z)

```

5. Hàm với nhiều đầu vào và/hoặc nhiều đầu ra:

```

function[addition, difference] = total_diff(x, y)
% function [total, difference] = total_diff(x, y)
% find the sum and difference between two vectors
addition = x + y;

```

```
difference = x - y;
```

VI. Các cấu trúc điều khiển chương trình trong Matlab:

1. Cấu trúc if...end

```
x = 10;  
y = sqrt(x)-x/3;  
if y < 0  
    'y nhỏ hơn 0'  
    y = y + 1;  
end
```

2. Cấu trúc if...else...end

```
x = 10;  
y = sqrt(x)-x/3+1;  
if y < 0  
    'y nhỏ hơn 0'  
    y = y + 1;  
else  
    'y lớn hơn hoặc bằng 0'  
    y = y - 1;  
end
```

3. Cấu trúc if...elseif...else...end

```
y = 0;  
if y < 0  
    'y nhỏ hơn 0'  
    y = y + 1;  
elseif y == 0  
    'y bằng 0'  
else  
    'y lớn hơn 0'  
    y = y - 1;  
end
```

4. Cấu trúc for...end

```
for mm = 1:2:10  
    y(mm) = mm^2;  
end
```

* Cách khác (hiệu quả hơn): $y = [1:2:10].^2$;

5. Vòng lặp đôi:

```
A = [1 3 2; 4 -1 0];  
[n_row, n_col] = size(A);  
  
for mm = 1:n_row  
    row_avg(mm) = mean(A(mm, :));  
    for nn = 1:n_col  
        B(mm, nn) = A(mm, nn).^2;  
    end  
end
```

6. Cấu trúc while...end

```
x = 1;  
while x <= 20
```



```

x = 3*x + 2;
end

```

VII. Phép toán sử dụng biến tượng trưng:

- Phân tích, giải quyết và sử dụng các biểu thức toán học sử dụng biến tượng trưng. (tất cả các phần bên trên đều dựa trên tính toán số học: biến phải có giá trị).
- Trong phép toán này, biến không cần phải có giá trị cụ thể.
- Ví dụ, chúng ta có thể sử dụng phép toán với biến tượng trưng để thực hiện các phép nguyên hàm, vi phân.

1. Khai báo biến tượng trưng:

- Khai báo các biến a, b, x như là biến tượng trưng (chúng không cần phải có một giá trị số cụ thể).
- Định nghĩa một hàm dùng biến tượng trưng $f(x) = x^a e^{-bx}$

```

>> syms a b x      % khai bao cac bien tuong trung
>> f = x^a*exp(-b*x) % su dung bien tuong trung dinh nghĩa một
ham so
f =
x^a/exp(b*x)

```

2. Phép vi phân: lệnh `diff`

Tìm vi phân bậc nhất của hàm số $f(x) = x^a e^{-bx}$.

```

>> diff_f = diff(f, x) % vi phân
diff_f =
(a*x^(a - 1))/exp(b*x) - (b*x^a)/exp(b*x)

>> simplify(diff_f) % yêu cầu Matlab đơn giản hóa kết quả.
ans =
(x^(a - 1)*(a - b*x))/exp(b*x)

```

3. Phép tích phân: lệnh `integral`

- Tìm tích phân không xác định của hàm số $f_2(x) = x e^{-ax}$

```

>> syms a x
>> f2 = x*exp(-a*x)
f2 =
x/exp(a*x)

>> int_f = int(f2, x) % tích phân không xác định
int_f =
-(a*x + 1)/(a^2*exp(a*x))

```

- Tích phân xác định: $\int_0^{10} x e^{-ax} dx$

```
>> int(f2, x, 0, 10)
ans =
1/a^2 - (10*a + 1)/(a^2*exp(10*a))
```

4. Câu lệnh thay thế:

- Lệnh subs sẽ thay thế một số vào vị trí của một biến tượng trưng trong biểu thức sử dụng biến tượng trưng.

- Ví dụ 1: tính giá trị hàm số $f(x) = x^a e^{-bx}$ khi $a = 2, b = 1$, và $x = 3$.

```
>> syms a b x
>> f = x^a*exp(-b*x)
f =
x^a/exp(b*x)
>> subs(f, {a, b, x}, {2, 1, 3})
ans =
0.4481
```

- Ví dụ 2: Tính tích phân $\int_0^{10} x e^{-ax} dx$ khi $a = 2$.

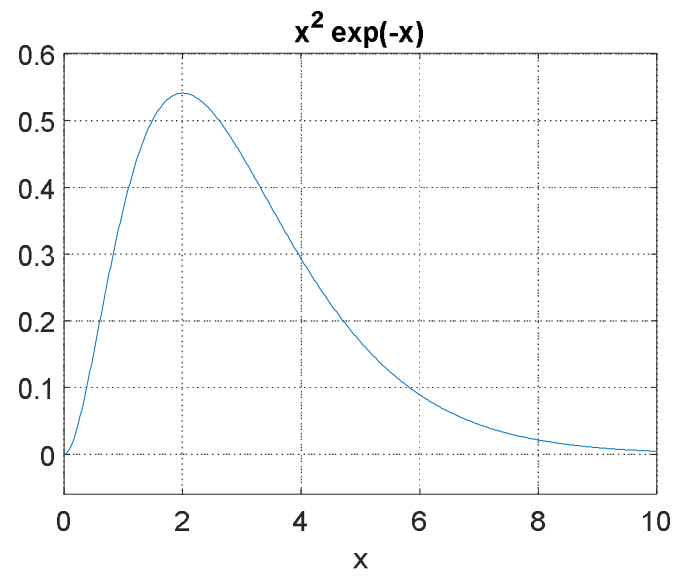
```
>> syms a x
>> f2 = x*exp(-a*x)
f2 =
x/exp(a*x)
>> int_f2 = int(f2, 0, 10)
int_f2 =
1/a^2 - (10*a + 1)/(a^2*exp(10*a))
>> subs(int_f2, a, 2)
ans =
0.2500
```

5. Vẽ đồ thị hàm số với lệnh ezplot:

Ví dụ: vẽ đồ thị hàm số $f(x) = x^a e^{-bx}$ khi $a = 2$ và $b = 1$ cho biến x trong khoảng từ $[0;10]$

```
>> syms x y a b
>> f = x^a*exp(-b*x);
>> f3 = subs(f, {a, b}, {2, 1})
f3 =
x^2/exp(x)
```

```
>> ezplot(f3, [0, 10])
```



Bài 1. Tín hiệu liên tục

I. Hàm bước nhảy đơn vị (unit step) và hàm dốc đơn vị (ramp)

Bài 1. Viết hàm $y = \text{ustep}(t)$ để biểu diễn hàm bước nhảy đơn vị.

Bài 2. Viết hàm $y = \text{uramp}(t)$ để biểu diễn hàm dốc đơn vị.

Bài 3. Sử dụng các hàm vừa viết, vẽ đồ thị của các tín hiệu liên tục sau trên đoạn $-10 \leq t \leq 10$

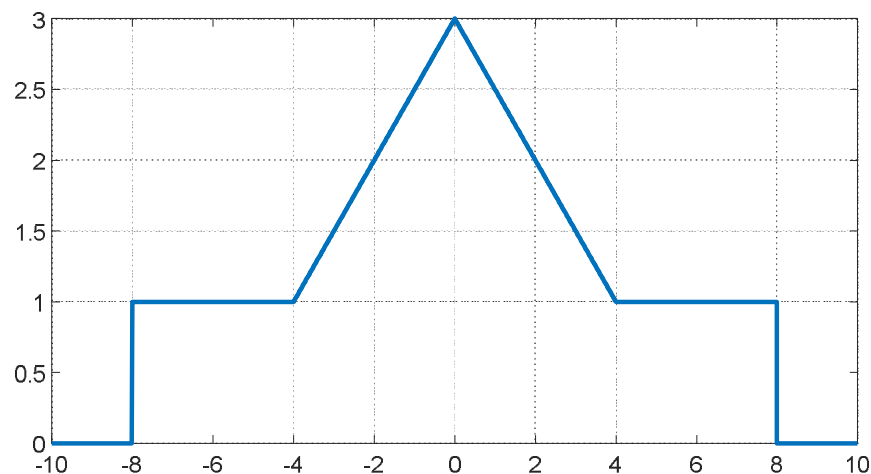
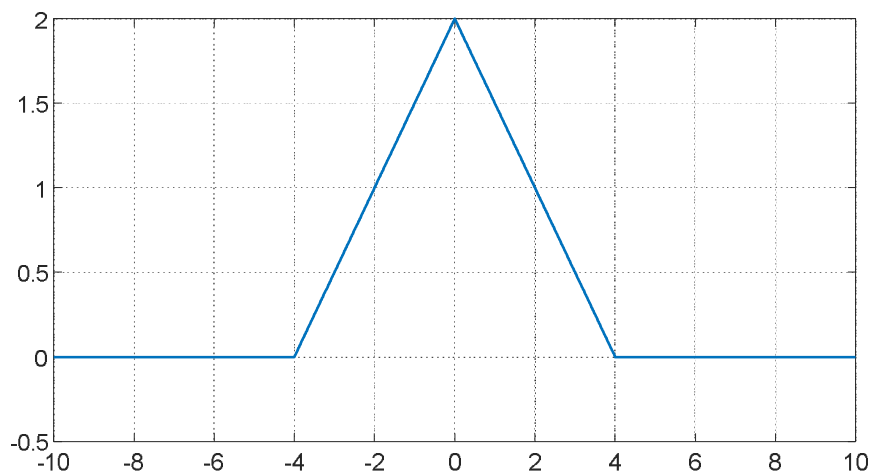
$$+ 5u(t-2).$$

$$+ 3r(t+5).$$

$$+ y(t) = 2r(t+2,5) - 5r(t) + 3r(t-2) + u(t-4).$$

$$+ y(t) = \sin(t) * [u(t+3) - u(t-3)].$$

Bài 4. Sử dụng hai hàm trên để tạo ra các tín hiệu có đồ thị như sau:



II. Tín hiệu chẵn, lẻ:

Bài 1. Xây dựng hàm số trả về kết quả là phần chẵn và phần lẻ của một tín hiệu như sau:

```

function [ye,yo] = evenodd(y)
% even/odd decomposition
% y: analog signal
% ye, yo: even and odd components
% USE [ye,yo] = evenodd(y)
%
yr = flipplr(y);
ye = 0.5*(y + yr);
yo = 0.5*(y - yr);

```

Bài 2. Sử dụng hàm số trên để tìm phần chẵn và phần lẻ của các tín hiệu liên tục sau và vẽ đồ thị của tín hiệu chính cũng như phần chẵn và phần lẻ của nó trong cùng một đồ thị sử dụng các dạng đường thẳng và màu sắc khác nhau: (giả sử $-10 \leq t \leq 10$).

$$y(t) = 2r(t+2,5) - 5r(t) + 3r(t-2) + u(t-4)$$

III. Tổng của các tín hiệu tuần hoàn:

Vẽ dạng của các tín hiệu sau trên đoạn $-10 \leq t \leq 10$. Tín hiệu đó có phải là tín hiệu tuần hoàn hay không? Nếu có, tìm chu kỳ của nó?

a. $x_1(t) = 1 + 1,5 \cos(2\pi\Omega_0 t) - 0,6 \cos(4\Omega_0 t)$ với $\Omega_0 = \frac{\pi}{10}$.

b. $x_2(t) = 1 + 1,5 \cos(6\pi t) - 0,6 \cos(4\Omega_0 t)$ với $\Omega_0 = \frac{\pi}{10}$.

III. Năng lượng, công suất của một tín hiệu:

Năng lượng của một tín hiệu trong khoảng $\left[-\frac{T}{2}; \frac{T}{2}\right]$ được định nghĩa là

$$E = \int_{-\frac{T}{2}}^{+\frac{T}{2}} |x(t)|^2 dt. \text{ Công suất của nó thì được định nghĩa là } P = \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} |x(t)|^2 dt. \text{ Tìm}$$

năng lượng và công suất của tín hiệu sau trên đoạn $-10 \leq t \leq 10$ bằng cách sử dụng công cụ biến tượng trưng của Matlab.

$$x(t) = e^{-t} \cos(2\pi t) u(t).$$

IV. Phép dịch, phép co giãn và phép đảo tín hiệu:

Bài 1. Vẽ đồ thị của các hàm số sau trên cùng một đồ thị: $x(t)$, $x(t-2)$ và $x(t+2)$ (giả sử $-10 \leq t \leq 10$) với: $x(t) = e^{-|t|}$.

Bài 2. Vẽ đồ thị của các hàm số sau trên cùng một đồ thị: $x(t)$, $x(2t)$ và $x(0,5t)$ (giả sử $-10 \leq t \leq 10$) với: $x(t) = e^{-|t|}$.

Bài 3. Vẽ đồ thị của các hàm số sau trên cùng một đồ thị: $x(t)$ và $x(-t)$ (giả sử $-10 \leq t \leq 10$) với: $x(t) = e^{-|t|}$.

Bài 2. Hàm tuyến tính

Bài 1. Tần số và nốt nhạc

Tần số của các nốt nhạc được cho trong bảng dưới đây

Bảng 1. Tần số các nốt

Note	C	D	E	F	G	A	B
Freq (Hz)	262	294	330	349	392	440	494

Mỗi nốt nhạc là một tín hiệu hình sin với một tần số nhất định. Ví dụ sau đây sẽ cho thấy cách để chơi một nốt nhạc bằng Matlab.

```
% example of a sinusoidal signal
% playing and plotting a pure tone
%
T = 2;           % time to play the note is 2 sec
Fs = 8000;       % sampling frequency is 8000 Hz
t=0:1/Fs:T;      % vector of time instants
Amp = 1;         % amplitude of the tone
ph=0;           % phase of the tone
fb = 494;        % frequency of middle B
N=300;

x = Amp*sin(2*pi*fb*t+ph); % vector x contains the values of
                           % the sinusoidal of frequency fb
                           % try with x = Amp*exp(-t).*sin(2*pi*fb*t+ph);

plot(t(1:300),x(1:300));

% play the note
sound(x,Fs);
```

Hãy viết một chương trình Matlab để chơi bản nhạc sau: CCGGAAG--, FFEEDDC--,
Bạn có thể căn chỉnh độ dài của từng nốt để có kết quả tốt hơn.

```
% program to play the ABC song with pure tone
%
clc
clear all
%clf reset

% MUSICAL SCALE
s=2^(1/12);

R=0;           % This is a rest

A0=110;
A0s=A0*s;
B0=A0s*s;
C0=B0*s;
```

```

C0s=C0*s;
D0=C0s*s;
D0s=D0*s;
E0=D0s*s;
F0=E0*s;
F0s=F0*s;
G0=F0s*s;
G0s=G0*s;

A=220;
As=A*s;
B=As*s;
C=B*s;           % Middle C
Cs=C*s;
D=Cs*s;
Ds=D*s;
E=Ds*s;
F=E*s;
Fs=F*s;
G=Fs*s;
Gs=G*s;

A2=440;
A2s=A2*s;
B2=A2s*s;
C2=B2*s;
C2s=C2*s;
D2=C2s*s;
D2s=D2*s;
E2=D2s*s;
F2=E2*s;
F2s=F2*s;
G2=F2s*s;
G2s=G2*s;

Fs=8000;
t=0:1/Fs:0.6;
t2=0:1/Fs:1.2;

N=300;
A=2;
nC=A*exp(-0.5*t).*cos(2*pi*C*t);
nC2=A*exp(-0.5*t2).*cos(2*pi*C*t2);
nG=A*exp(-0.5*t).*cos(2*pi*G*t);
nGC=A*exp(-0.5*t).*cos(2*pi*G*t);
nGC2=A*exp(-0.5*t2).*cos(2*pi*G*t2);
nG2=A*exp(-0.5*t2).*cos(2*pi*G*t2);
nA=A*exp(-0.5*t).*cos(2*pi*A2*t);
nF=A*exp(-0.5*t).*cos(2*pi*F*t);
nE=A*exp(-0.5*t).*cos(2*pi*E*t);
nD=A*exp(-0.5*t).*cos(2*pi*D*t);
nDG2=A*exp(-0.5*t2).*cos(2*pi*D*t2);

x=[nC,nC,nGC,nG,nA,nA,nGC2,nF,nF,nE,nE,nD,nD,nC2,...
    nG,nG,nF,nF,nE,nE,nDG2,nG,nG,nF,nF,nE,nE,nDG2,...
    nC,nC,nGC,nG,nA,nA,nGC2,nF,nF,nE,nE,nD,nD,nC2];
sound(x,Fs);

```

Question 2 (Fourier Series of a Trumpet – MATLAB)

Since instruments playing musical notes create periodic signals, musical signals have Fourier expansions. The Fourier series can be truncated to a finite number of terms and still do a good job of representing the musical signal.

For example, the trumpet is play note B. We know from the Table 1 that, note B has a frequency of 494 Hz. So the period of note B is $T = 1/494$ seconds. Therefore, the trumpet signal can be expanded as a Fourier series which is a sum of sinusoids at frequencies of $\{494, 2(494), 3(494), \dots\}$ Hertz. Using values in Table 2,

Table 2. Amplitudes and phases of truncated Fourier series coefficients

Amplitude	Phase (radians)
$A_1 = 0.1155$	$\theta_1 = -2.1299$
$A_2 = 0.3417$	$\theta_2 = +1.6727$
$A_3 = 0.1789$	$\theta_3 = -2.5454$
$A_4 = 0.1232$	$\theta_4 = +0.6607$
$A_5 = 0.0678$	$\theta_5 = -2.0390$
$A_6 = 0.0473$	$\theta_6 = +2.1597$
$A_7 = 0.0260$	$\theta_7 = -1.0467$
$A_8 = 0.0065$	$\theta_8 = +1.8581$
$A_9 = 0.0020$	$\theta_9 = -2.3925$

the nine-term finite Fourier series approximation to the trumpet signal is:

$$\begin{aligned} x(t) &= \underbrace{A_0}_{\text{DC}} + \underbrace{A_1 \cos[2\pi(1)494t - \theta_1]}_{\text{FUNDAMENTAL}} \\ &\quad + \underbrace{A_2 \cos[2\pi(2)494t - \theta_2]}_{\text{HARMONIC}} + \dots \\ &= \sum_{k=0}^9 A_k \cos[2\pi(k)494t - \theta_k]. \end{aligned}$$

Note that $A_0 = 0$.

- (a) Using values in Table 2, write an M-file to synthesize the trumpet signal from nine-term finite Fourier series approximation. Then listen to the signal using `sound` function. We can store 44100 samples for each second of the trumpet in `x` variable in MATLAB and therefore the first command in your program should be:

```
t = linspace(0,1,44100); F=494;
```

- (b) Plot the signal $x(t)$ within its 3 periods (i.e., about 4.5×10^{-3} seconds).

- (c) Repeat part (a) and (b) for $\theta_k = 0$. Does changing the phases effect the sound of the signal?

```
% example of Fourier series
% synthesis of a trumpet playing note B
%
close all;
clear all;
```

```
C = [0.1155 0.3417 .1789 .1232 .0678 .0473 .0260 0.0065 0.0020];
```



```
Th = zeros(1,9);  
B = 494;  
F = [B 2*B 3*B 4*B 5*B 6*B 7*B 8*B 9*B];  
  
Fs=8000;  
t=0:1/Fs:2;  
x=1*C*cos(2*pi*F'*t);  
  
sound(x,Fs);
```

Bài 3: Tích chập, phép biến đổi Fourier và lọc tín hiệu

I. Tích chập và lọc tín hiệu âm thanh bằng bộ lọc thông thấp lý tưởng.

Chương trình dưới đây là một ví dụ của việc truyền một tín hiệu âm thanh qua một bộ lọc thông thấp với tần số cắt là 1500Hz. [Chú ý rằng, cốt lõi của chương trình này là hàm `conv` có sẵn của MATLAB.]

```
% doc file
[data, Fs, Nbits]=wavread('female_voice.wav');

% Lưu ý: trong các phiên bản Matlab từ 2014 trở đi lệnh wavread này được thay
% thế bởi % lệnh audioread
data = data(:, 1).';
% Fs: tần số lấy mẫu; Ts: Thời gian lấy mẫu
Ts = 1/Fs;

% phát lại âm thanh bị hỏng
sound(data, Fs);

% vectơ thời gian
t = [-10:Ts:10];

% tần số cắt của bộ lọc là 1500Hz
wb = 1500*2*pi;

% bộ lọc thông thấp lý tưởng với tần số cắt wb
% biến đổi fourier: rect(w/wb)
ht = wb/(2*pi)*sinc(wb*t/(2*pi));

% đầu vào: dữ liệu, đáp ứng tuyến tính, bất biến: ht
% đầu ra: y = tích chập của dữ liệu với ht
y = conv(data, ht, 'same');

% chuẩn hóa âm thanh đã xử lý tránh sự cắt xén
y = y/max(abs(y));
% phát lại âm thanh đã được xử lý
sound(y, Fs);
```

II. Phép biến đổi Fourier và lọc tín hiệu bằng bộ lọc Butterworth bậc 5

1. Tự tạo các hàm của bạn

Trong cửa sổ soạn thảo, bạn hãy viết chương trình tạo hàm `FourierTransform` để tính ảnh Fourier của một tín hiệu như dưới đây. Sau đó lưu vào với tên là `FourierTransform.m`.

[Chú ý rằng, cốt lõi của hàm tự tạo này là hàm `fft` có sẵn của MATLAB. Hàm `fft` thực hiện thuật toán biến đổi Fourier nhanh (*Fast Fourier Transform*).]

```
function [f,X]=FourierTransform(t,x)
% computes the Fourier transform of signal x(t)
% ns: length(x)=number signal points
```

```

% dt: signal point spacing
%
% Transform computed with N points, where N=2*ns

ns=size(x,2); dt=t(2)-t(1);
N=2*ns; df=1/(N*dt);
xp=zeros(1,N); nns=sum(t<0);
xp(1:ns-nns)=x(nns+1:ns); xp(N-nns+1:N)=x(1:nns);
Xf=dt*fft(xp); n2=ceil(N/2);
if n2==N/2; X(1:n2-1)=Xf(n2+2:N); X(n2:N)=Xf(1:n2+1);
    f=(-n2+1)*df:df:n2*df; no=n2;
else; X(1:n2-1)=Xf(n2+1:N); X(n2:N)=Xf(1:n2);
    f=(-n2+1)*df:df:(n2-1)*df; end;

```

Tương tự, bạn hãy mở một cửa sổ soạn thảo khác và viết chương trình tạo hàm `IFourierTransform` để tính ảnh Fourier ngược như dưới đây. Sau đó lưu vào với tên là `IFourierTransform.m`. [Chú ý rằng, cốt lõi của hàm tự tạo này là hàm `ifft` có sẵn của MATLAB.]

```

function [t, x] = IFourierTransform(f, X)
% computes the inverse Fourier transform of X(f)
% ns: length(X)=number of transform points
% df: transform point spacing
%
% Signal computed with N points, where N=ns

ns=length(X); df=f(2)-f(1);
N=ns; dt=1/(N*df);
Xp=zeros(1,N); Xp(1:ns)=X;
nns=sum(f<0);
Xpp(1:ns-nns)=Xp(nns+1:ns); Xpp(N-nns+1:N)=Xp(1:nns);
xf=N*df*ifft(Xpp); n2=ceil(N/2);
if n2==N/2; x(1:n2-1)=xf(n2+2:N); x(n2:N)=xf(1:n2+1);
    t=(-n2+1)*dt:dt:n2*dt;
else; x(1:n2-1)=xf(n2+1:N); x(n2:N)=xf(1:n2);
    t=(-n2+1)*dt:dt:(n2-1)*dt; end;

```

2. Loại tín hiệu điện tim

Điện tâm đồ (*Electrocardiogram - ECG*) là đồ thị ghi những thay đổi của dòng điện sinh học trong tim người được tạo ra khi quả tim co bóp theo nhịp. Dòng điện này tuy nhỏ (khoảng một phần nghìn vôn) nhưng được máy ghi điện khuếch đại lên và ghi lại trên điện tâm đồ. Điện tâm đồ được sử dụng trong y học để phát hiện các bệnh về tim.

Số liệu điện tim của một người được lưu trong file `hum3hb.mat` có sẵn trong máy tính ở PTN. Các giá trị điện áp được lưu trong vector `hb` và thời gian lấy mẫu là 0.002 giây được lưu trong biến `T`. Để giảm thiểu nhiễu tần số cao trong máy ghi điện tim, ta sử dụng một bộ lọc với đáp ứng xung là

$$h(t) = 568e^{-300t} - e^{-243t} [485\cos(176t) - 668\sin(176t)] - e^{-93t} [83\cos(285t) + 255\sin(285t)], \quad t > 0.$$

Chương trình `main.m` của MATLAB sau đây được sử dụng để vẽ tín hiệu điện tim và phổ biên độ của nó.

```
% main.m
load hum3hb;
x=hb; tf=(size(x,2)-1)*T;
t=0:T:tf;
[f,Xf]=FourierTransform(t,hb);
figure(1);
subplot(2,1,1)
plot(t,x,'linewidth',1);
grid; axis([0 2.5 -0.5 1.5]);
xlabel('t (sec)');ylabel('x(t)');
subplot(2,1,2)
plot(f,abs(Xf),'linewidth',1);
grid; axis([-150 150 0 0.12]);
xlabel('f (Hz)');ylabel('|X(f)|');
```

Câu hỏi 1: Hãy chạy chương trình trên và nhận xét về đồ thị thời gian của tín hiệu điện tim và phổ của nó. [Gợi ý: Trên đồ thị phổ, bạn thấy có những nhiễu nào?]

Tiếp theo, hãy viết tiếp vào chương trình `main.m` đoạn lệnh sau chương trình sau để có được đồ thị đáp ứng biên độ và đáp ứng pha của bộ lọc.

```
h=(568*exp(-300*t)-485*exp(-243*t).*cos(176*t)...
+668*exp(-243*t).*sin(176*t)-83*exp(-93*t).*cos(285*t)...
-255*exp(-93*t).*sin(285*t));
[f,Hf]=FourierTransform(t,h);
figure(2)
subplot(2,2,[1 2])
plot(t,h,'linewidth',1); grid;
axis([0 0.5 -50 150]);
xlabel('t (sec)'); ylabel('h(t)');
subplot(2,2,3)
plot(f,abs(Hf),'k','linewidth',1); grid;
axis([-150 150 0 1.2]);
xlabel('f (Hz)'); ylabel('|H(j2\pi f)|');
subplot(2,2,4)
```

```

angleH=unwrap(angle(Hf))+2*pi;
plot(f,angleH,'k','linewidth',1); grid;
axis([-150 150 -10 10]);
xlabel('f (Hz)'); ylabel('\angle H(f)');

```

Câu hỏi 2: Đây là loại bộ lọc gì? [Gợi ý: thông thấp, thông cao,...?]. Vùng tần số mà bộ lọc cho đi qua?

Cuối cùng, bạn hãy vẽ phổ biên độ của tín hiệu ra cùng với tín hiệu ra của bộ lọc bằng đoạn lệnh sau đây (viết tiếp vào chương trình main.m).

```

Yf=Xf.*Hf;
figure(3)
subplot(2,1,1)
plot(f,abs(Yf),'r','linewidth',1); grid;
axis([-150 150 0 0.12]);
xlabel('f (Hz)'); ylabel('|Y(f)|');
[t2,y]=IFourierTransform(f,Yf);
subplot(2,1,2)
plot(t2,y,'r','linewidth',1); grid;
axis([0 2.5 -0.5 1.5]);
xlabel('t (sec)'); ylabel('y(t)');

```

Câu hỏi 3: Nhận xét về tác dụng của bộ lọc khi so sánh các đồ thị trong các Hình 1 và 3.