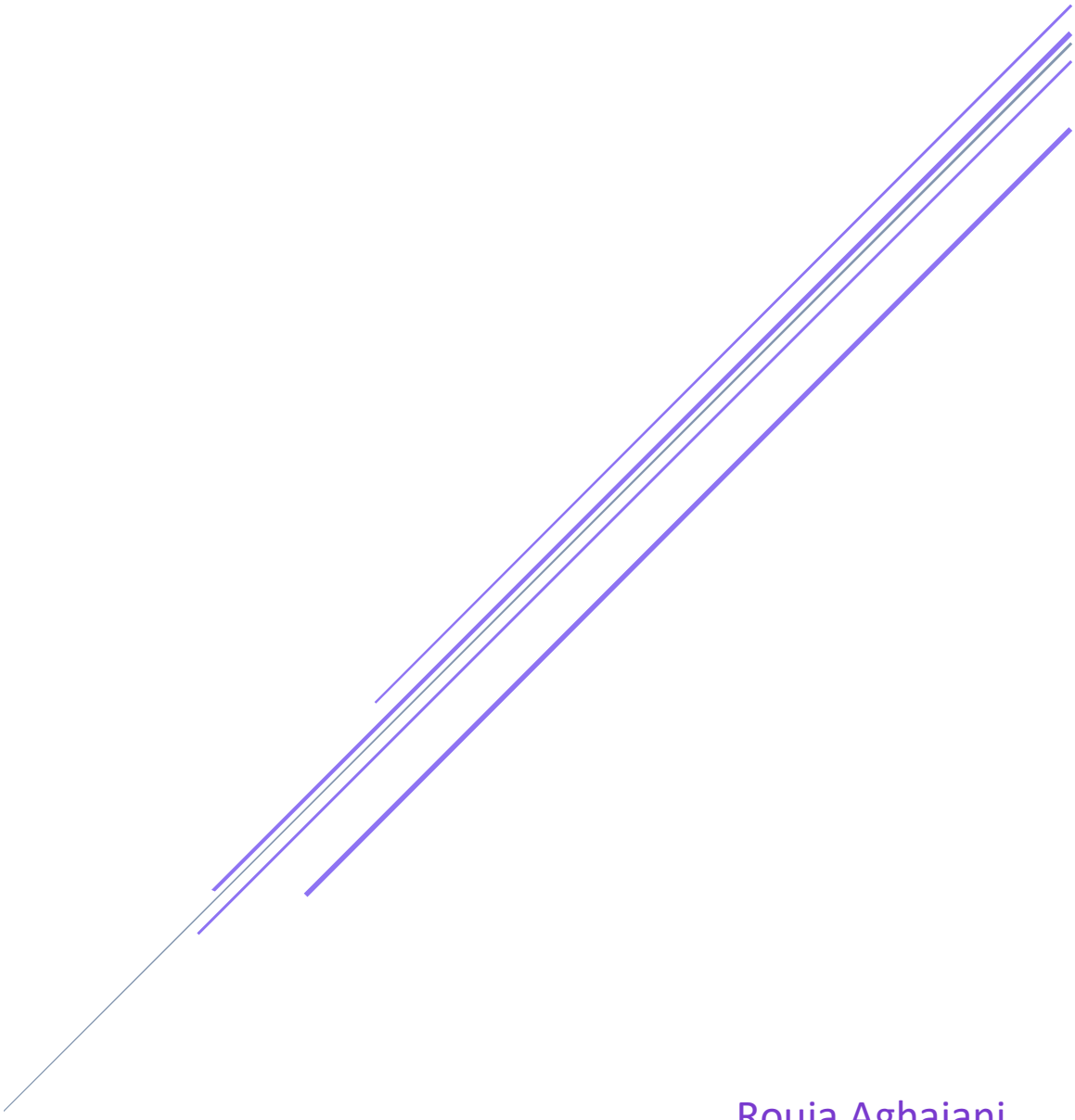
SIGNALS AND SYSTEMS

CA1



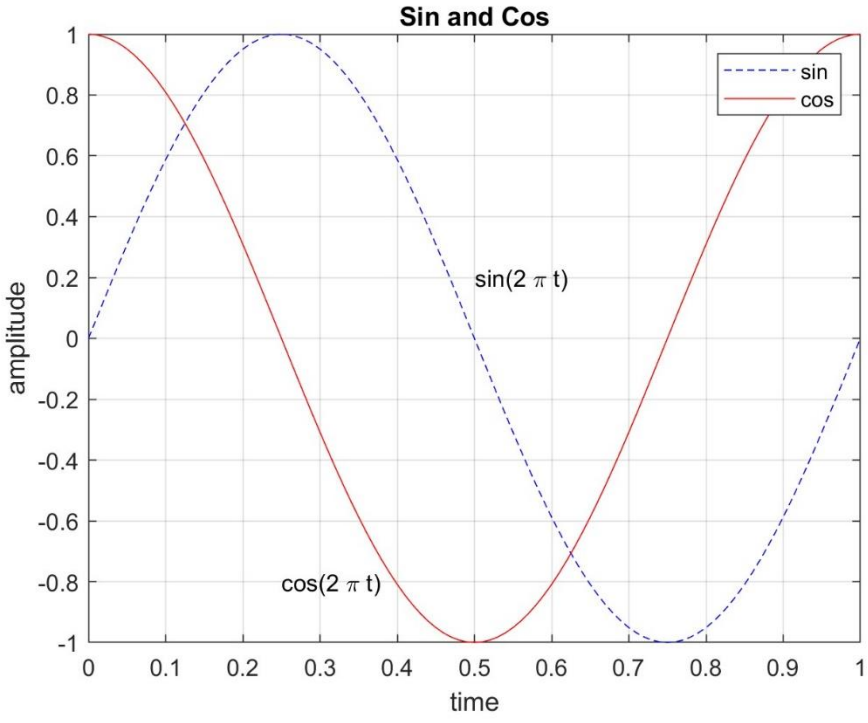
Rouja Aghajani
810101380

بخش اول:

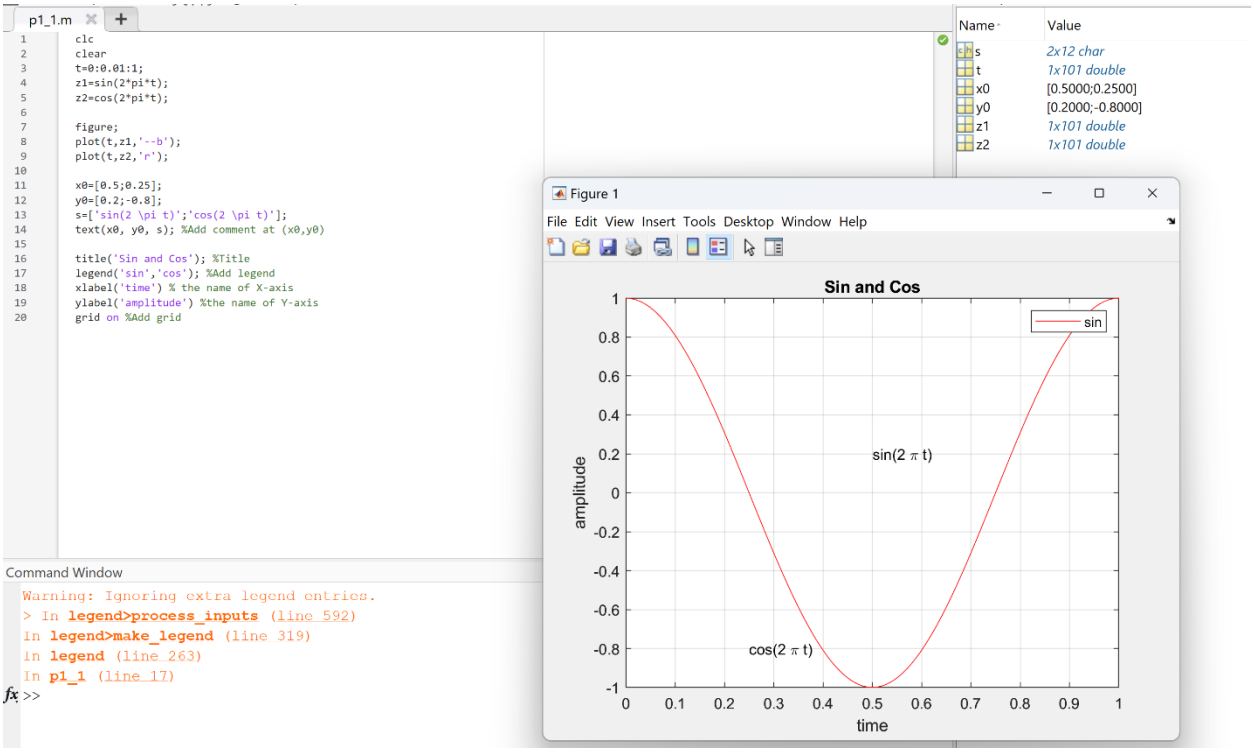
تمرین 1-1)

p1_1.m

```
clc
clear
t=0:0.01:1;
z1=sin(2*pi*t);
z2=cos(2*pi*t);

figure;
plot(t,z1,'--b');
hold on
plot(t,z2,'r');

x0=[0.5;0.25];
y0=[0.2;-0.8];
s=['sin(2 \pi t)';'cos(2 \pi t)'];
text(x0, y0, s); %Add comment at (x0,y0)

title('Sin and Cos'); %Title
legend('sin','cos'); %Add legend
xlabel('time') % the name of X-axis
ylabel('amplitude') %the name of Y-axis
grid on %Add grid
```



با حذف hold on :



Warning دریافتی به دلیل حذف hold on است که در legend مشکل ایجاد می کند.

ابتدا راجع به نتیجه حذف hold on : با حذف hold on، با اجرای خط بعد که دستور رسم نمودار کسینوس داده شده است، نمودار رسم شده قبلی یعنی سینوس پاک شده و سپس کسینوس رسم می شود. نتیجه آن است که با حذف hold on نمیتوان همزمان دو نمودار را در یک صفحه گراف نشان داد.

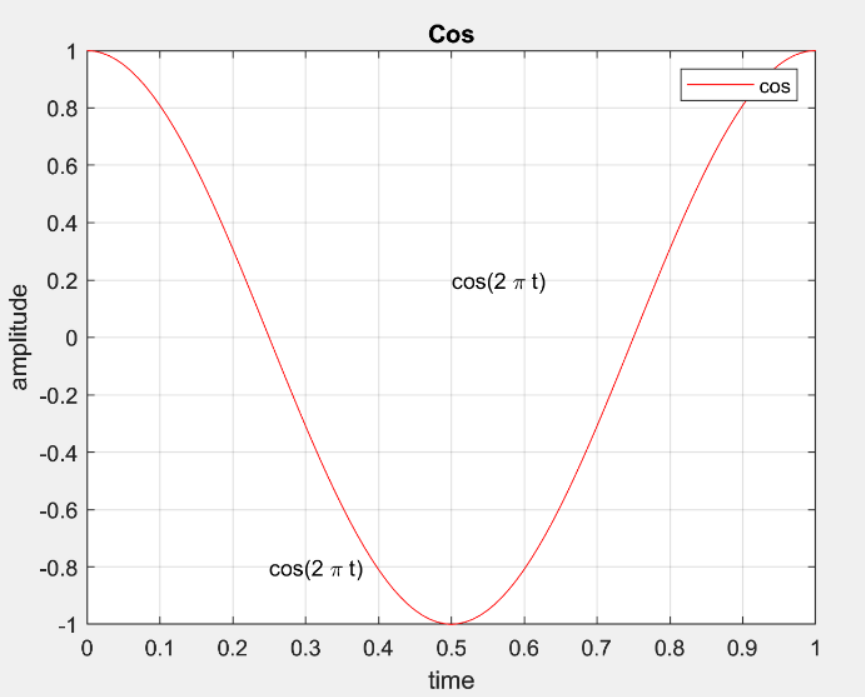
Warning برای legend: درواقع در legend گفتیم بنویسد نمودار با خط قرمز sin و با نقطه چین cos است. با حذف hold on، چون نمودار سینوس حذف می شود دیگر نمودار نقطه چینی باقی نمیماند که دستور legend بگوید آن نمودار بیانگر چه تابعی است. در نتیجه warning برای آن است که دستور legend ورودی مربوط به تابعی دارد که دیگر شکل آن موجود نیست.

همچنین باید نوشته $\sin(2\pi t)$ نیز حذف شود چون دیگر نمودار آن نمایش داده نمی شود.

با اعمال اصلاحات در نکات بالا:

p1_1.m

```
clc
clear
t=0:0.01:1;
z1=sin(2*pi*t);
z2=cos(2*pi*t);

figure;
plot(t,z1,'--b');
plot(t,z2,'r');

x0=[0.5;0.25];
y0=[0.2;-0.8];
s='cos(2 \pi t)';
text(x0, y0, s); %Add comment at (x0,y0)

title('Cos'); %Title
legend('cos'); %Add legend
xlabel('time') % the name of X-axis
ylabel('amplitude') %the name of Y-axis
grid on %Add grid
```



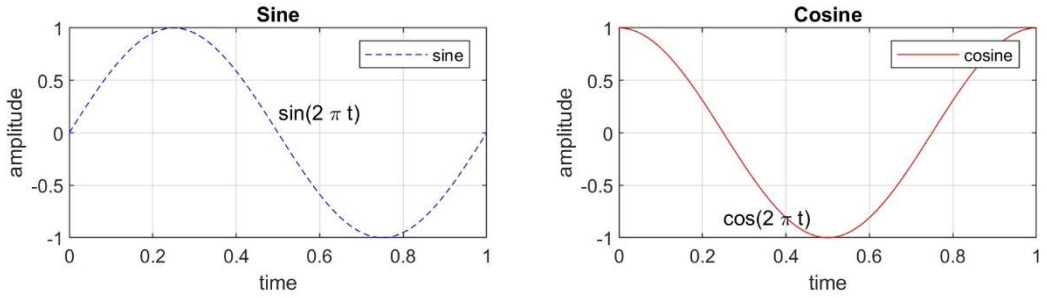
که در واقع در اینجا تنها نمودار کسینوس رسم میشود و خط 8 عملا بی تاثیر است.

تمرین 1-2)

با استفاده از دستور subplot و ایندکس دهی به صورت(2,2,2) و (2,2,1) توانستیم دو نمودار خواسته شده را در کنار هم رسم کنیم. توجه شود اگر ایندکس دهی به صورت (2,1,1) و (2,1,2) بود، زیر هم رسم میشدند. درواقع ایندکس ها به صورت اولین سطر و اولین ستون، دومین ستون و اولین سطر و.. می باشند.

p1_2.m

```
clc
clear
t=0:0.01:1;
z1=sin(2*pi*t);
z2=cos(2*pi*t);

x0=[0.5;0.25];
y0=[0.2;-0.8];

s1='sin(2 \pi t)';
s2='cos(2 \pi t)';

figure('Position', [90, 90, 800, 400]);

subplot(2,2,1);
plot(t,z1,'--b');

text(x0(1), y0(1), s1); % Add label for subplot 1
title('Sine'); %Title
grid on %Add grid
legend('sine');
xlabel('time') % the name of X-axis
ylabel('amplitude') %the name of Y-axis


subplot(2,2,2);
plot(t,z2,'r');

text(x0(2), y0(2), s2); % Add label for subplot 2
legend('cosine');
title('Cosine'); %Title
xlabel('time') % the name of X-axis
ylabel('amplitude') %the name of Y-axis
grid on %Add grid
```



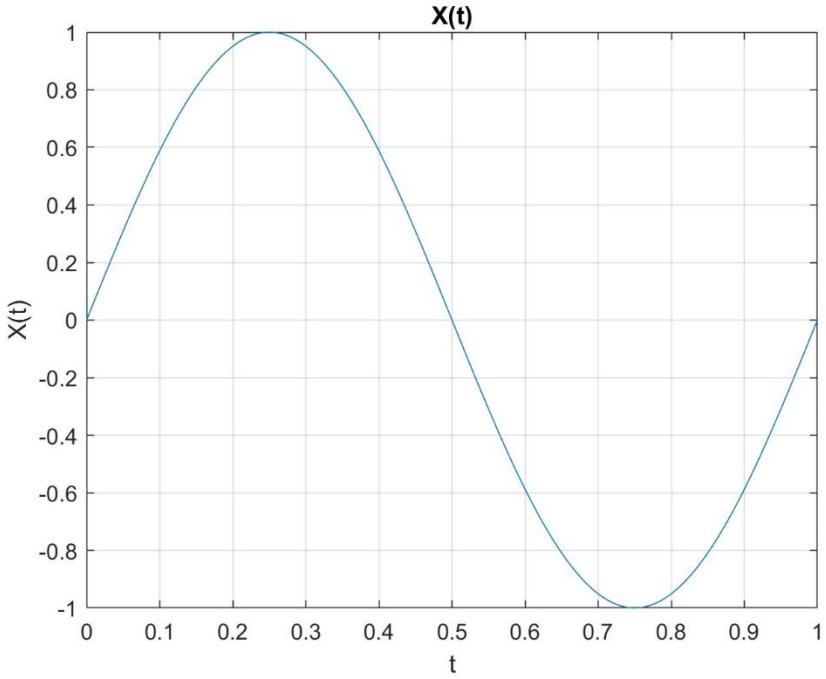
بخش دوم)

تمرین 1-2)

مطابق خواسته سوال، x(t) با دستور plot رسم گردید

p2_1.m

```
clc
plot(t,x)
title('X(t)');
ylabel('X(t)');
xlabel('t');
grid on
```

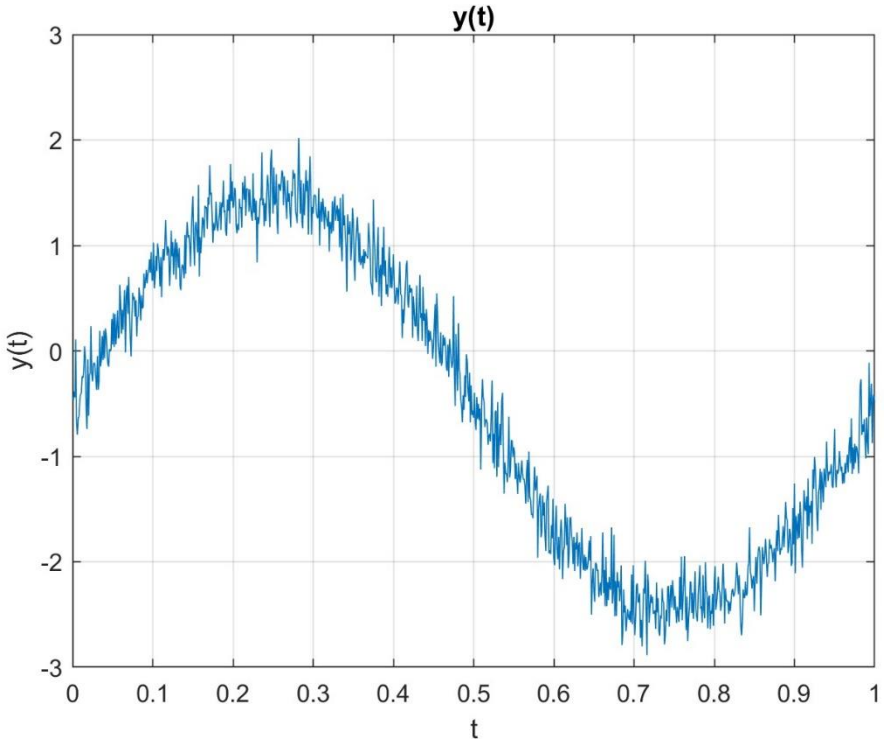


تمرین 2-2)

مطابق خواسته سوال y(t) با دستور plot رسم شد. مشاهده شد y(t) شکلی مانند تابع سینوس ولی با نویز دارد.

p2_2.m

```
clc
plot(t,y)
title('y(t)');
ylabel('y(t)');
xlabel('t');
grid on
```



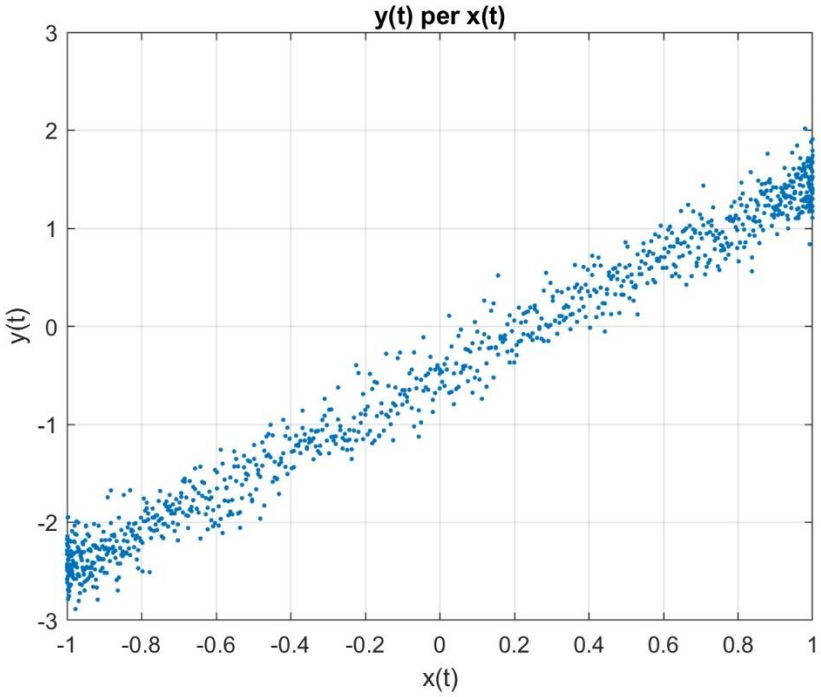
تمرین 2-3)

برای رسم y بر حسب x، با استفاده از دستور plot مقادیر محور افقی همان مقادیر x(t) و بردار افقی همان بردار x(t) است و مقادیر و بردار عمودی در نمودار مقادیر y(t) می باشد و بدین ترتیب y(t) برحسب x(t) رسم می شود.

همچنین دیده می شود نمودار رسم شده نقاطی پراکنده ولی گویا در امتداد خطی راست (درجه اول بر حسب x(t)) می باشد.

p2_3.m

```
1  clc
2  plot(x,y,'.')
3  title('y(t) per x(t)');
4  ylabel('y(t)');
5  xlabel('x(t)');
6  grid on
```



طبق فرمول داده شده:

$$\boldsymbol{y(t) = \alpha\, x(t) + \beta}$$

با رسم نمودار y(t) بر حسب x(t)، شیب نمودار بدست آمده همان α در فرمول بالا و عرض از مبدا نیز همان β در فرمول بالا می باشد.

تمرین 2-4)

از روش حداقل مربعات استفاده می کنیم.

مجموع خطاها به صورت زیر بدست می آید:

$$S = \sum_{i=1}^{m} (y_i - \overline{y_i})^2 = \sum_{i=1}^{m} \left(y_i - a_n x_i^n - a_{n-1} x_i^{n-1} - \cdots - a_1 x_i - a_0\right)^2$$

حال باید چند جمله ای برازش داده شده تا جای ممکن به نقاط جدولی نزدیک باشند تا خطا حداقل شود. به این منظور، باید مشتق خطا نسبت به تمامی ضرایب مجهول صفر شود؛ در نتیجه:

$$\frac{\partial S}{\partial a_j} = 0 \qquad\qquad j = 0\,, 1\,, \ldots, n$$

در این صورت معادلات نرمال با صورت زیر در آمده و با حل دستگاه معادلات بدست آمده، ضرایب مجهول محاسبه خواهند شد.

$$\begin{cases} ma_0 + (\sum x_i)\, a_1 + \cdots + (\sum x_i^n) a_n = \sum y_i \\ \sum x_i a_0 + (\sum x_i^n) a_1 + \cdots + (\sum x_i^{n+1})\, a_n = \sum x_i\, y_i \\ \vdots \\ (\sum x_i^n)\, a_0 + (\sum x_i^{n+1})\, a_1 + \cdots + (\sum x_i^{2n})\, a_n = \sum x_i^n\, y_i \end{cases}$$

تابع نوشته شده:

```
function [b , a] = p2_4(X , Y)
    syms b;
    syms a;
    fx=@(q,p)((q-b-a*p).*(q-b-a*p));
    counter=1:length(X);
    wantedSigma=sum(fx(Y(1,counter),X(1,counter)));
    equation1=diff(wantedSigma,b);
    equation2=diff(wantedSigma,a);
    eqns=[equation2==0,equation1==0];
    vars = [b a];
    [B, A] = solve(eqns,vars);
    b=eval(B);
    a=eval(A);
end
```

تابع تعریف می شود. دو متغیری که مقدارشان مورد سوال است نیز تعریف می شوند. تابعی که طبق روش کار ذکر شده باید از آن نسبت به متغیر های ذکر شده مشتق گرفته شود، تعریف می شود. یک شمارشگر برای انجام عملیات جمع برای تمام داده ها تعریف می شود. آنگاه عبارتی که باید از آن مشتق گرفته شود، بدست می آید. از عبارت بدست اورده شده نسبت به دو متغیر خواسته شده مشتق گرفته می شود تا طبق توضیح خطای آن کمترین مقدار خود شود و مقدار آن دو نیز بدست آید.

دو مشتق حساب شده برابر صفر قرار داده می شوند. متغیرهای مورد نظر انتخاب می شوند. مقدار دو متغیر بدست می آیند. توجه شود این دو مقدار ابتدا در دو متغیر A و B ذخیره می شوند، چراکه کسری هستند. آنگاه در مرحله بعد، با دستور eval مقدار آن دو به صورت عدد اعشاری حساب شده و آنگاه درون دو متغیر a و b جایگذاری می شوند.

تست با نویز و بدون نویز:

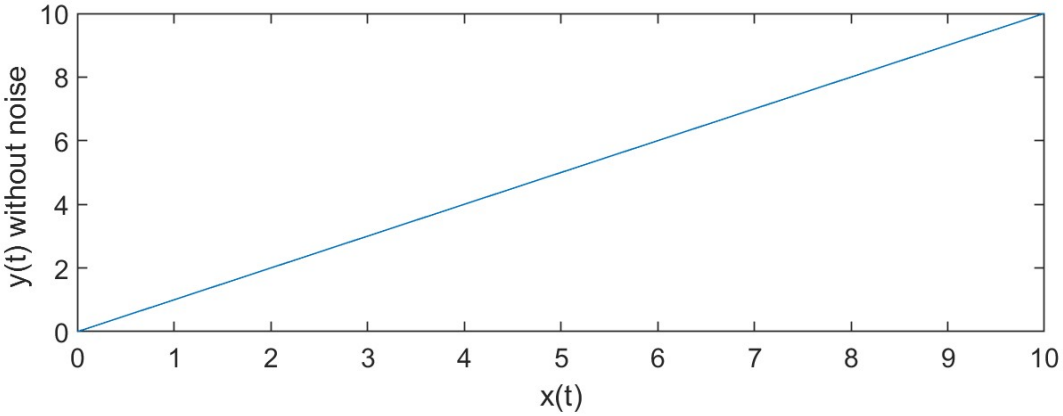
```
Xvals = 0:0.1:10;
Yvals = 0:0.1:10;
noise = 0.8*rand(1,length(Xvals));
[beta, alpha] = p2_4(Xvals, Yvals);
subplot(2,1,1);
plot(Xvals,Yvals);
ylabel('y(t) without noise ');
xlabel('x(t)');
subplot(2,1,2);
plot(Xvals,(Yvals+noise));
ylabel('y(t) with noise');
xlabel('x(t)');
fprintf('Test without noise:\n');
fprintf('alpha = %f and beta = %f , the line equation is: y = %f * x + %f \n',alpha,beta,alpha,beta);
[beta, alpha] = p2_4(Xvals, (Yvals + noise));
fprintf('Test with noise:\n');
fprintf('alpha = %f and beta = %f , the line equation is: y = %f * x + %f \n',alpha,beta,alpha,beta);

```

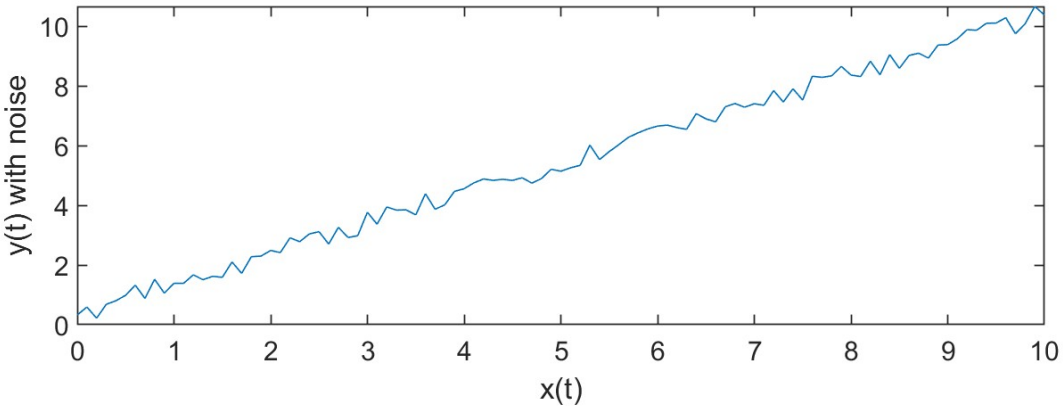
Command Window

```
>> p2_4_test
Test without noise:
alpha = 1.000000 and beta = 0.000000 , the line equation is: y = 1.000000 * x + 0.000000
Test with noise:
alpha = 1.009161 and beta = 0.369755 , the line equation is: y = 1.009161 * x + 0.369755
```

توابع مورد استفاده در تست:

تست با x و y داده شده:


y(t) without noise
x(t)
0
1
2
3
4
5
6
7
8
9
10

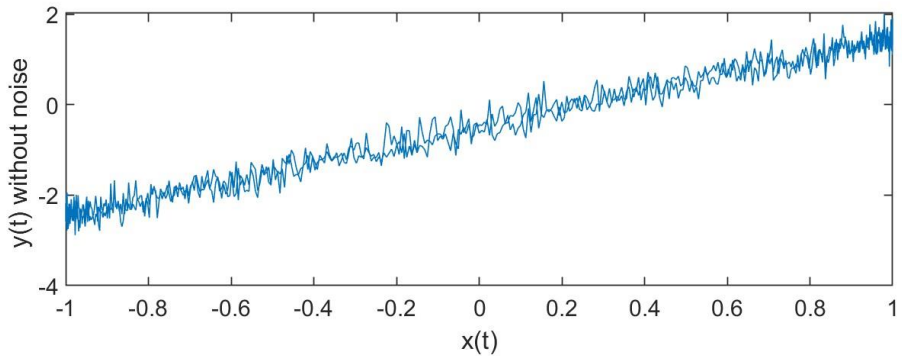

y(t) with noise
x(t)
0
1
2
3
4
5
6
7
8
9
10

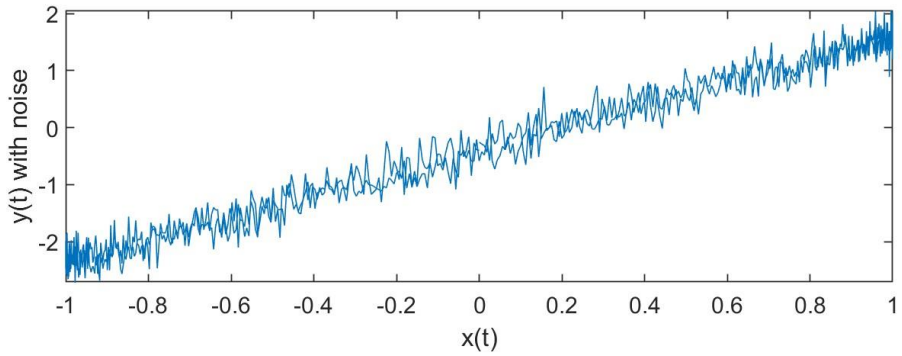
```
%Xvals = 0:0.1:10;
%Yvals = 0:0.1:10;
noise = 0.2*rand(1,length(x));
[beta, alpha] = p2_4(x, y);
subplot(2,1,1);
plot(x,y);
ylabel('y(t) without noise ');
xlabel('x(t)');
subplot(2,1,2);
plot(x,(y+noise));
ylabel('y(t) with noise');
xlabel('x(t)');
fprintf('Test without noise:\n');
fprintf('alpha = %f and beta = %f , the line equation is: y = %f * x + %f \n',alpha,beta,alpha,beta);
[beta, alpha] = p2_4(x, (y + noise));
fprintf('Test with noise:\n');
fprintf('alpha = %f and beta = %f , the line equation is: y = %f * x + %f \n',alpha,beta,alpha,beta);

```

Command Window

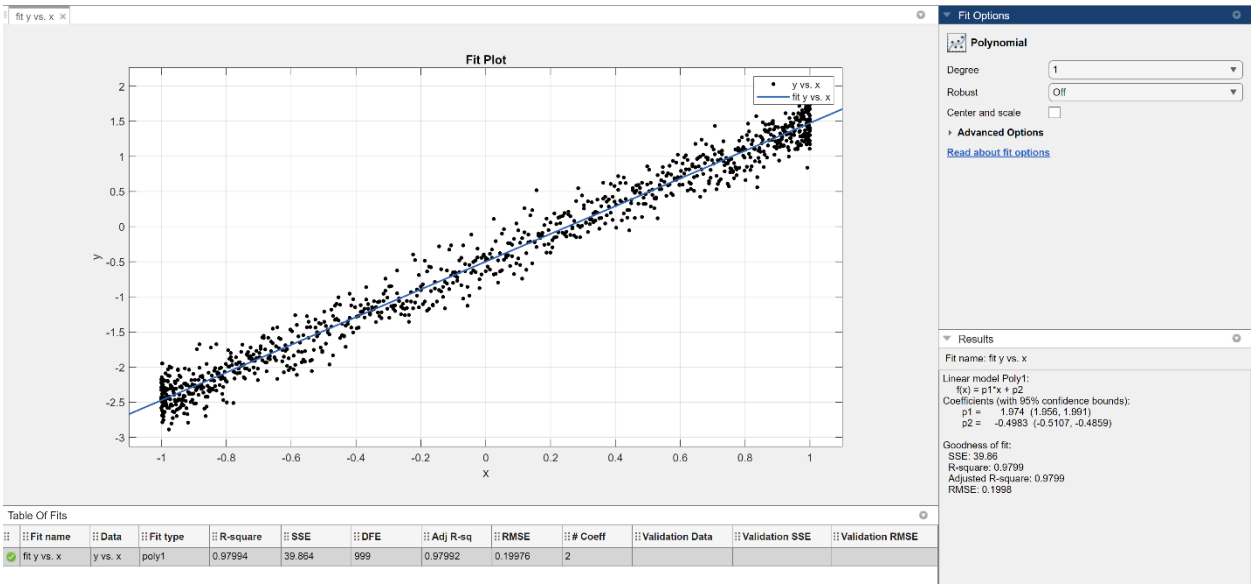
```
>> load('p2.mat')
>> p2_4_test
Test without noise:
alpha = 1.973579 and beta = -0.498339 , the line equation is: y = 1.973579 * x + -0.498339
Test with noise:
alpha = 1.972087 and beta = -0.397860 , the line equation is: y = 1.972087 * x + -0.397860
```

توابع مورد تست:





تمرین 2-5)



و مشاهده میشود ضرایب بدست آمده از این روش به صورت زیر می باشند:

```
Linear model Poly1:
    f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
       p1 =       1.974  (1.956, 1.991)
       p2 =     -0.4983  (-0.5107, -0.4859)
```

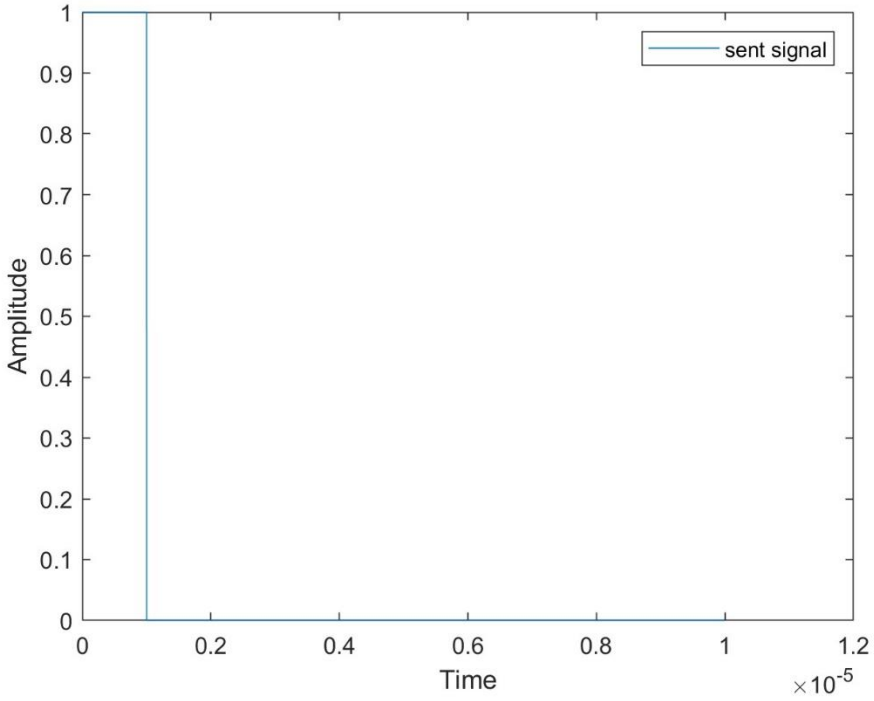
یعنی α=1.974 و β=-0.4983 که برای هر یک بازه اطمینان 95 درصدی نیز ارائه شده است. همچنین مشاهده می شود مقدار بدست آمده با مقادیر محاسبه شده در بخش قبل همخوانی دارد ونزدیک هم هستند.

بخش سوم:

1-3)

ابتدا یک وکتور صفر تعریف میشود، آنگاه با توجه به اطلاعات داده شده مقدار یک در بازه مد نظر (ts\tau) قرار داده شده، لذا سیگنال تعریف و در نهایت با دستور plot رسم می شود.

```
ts = 1e-9;
T = 1e-5;
tau = 1e-6;
t=0:ts:T;
t_len=length(t);
sent_signal = zeros(1,t_len);
sent_signal(1:round(tau/ts))=1;


plot(t,sent_signal);
ylabel("Amplitude");
xlabel("Time");
legend('sent signal')
```

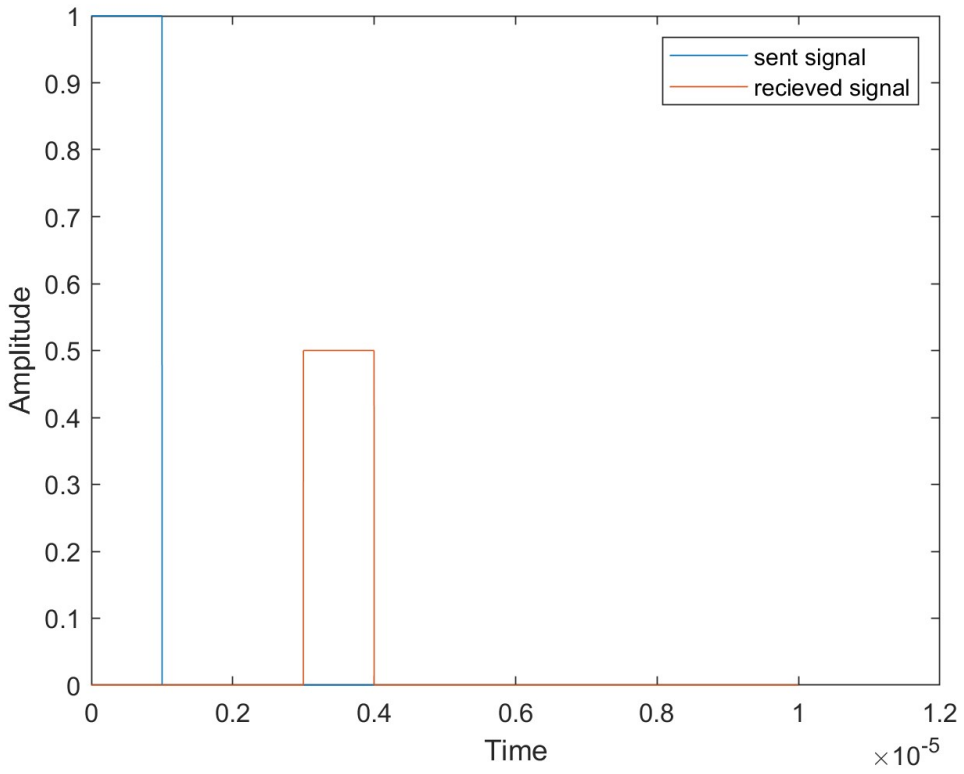


2-3) با توجه به بازه 450 متر داده شده، سیگنال دریافتی ابتدا به شکل وکتور صفر تعریف شده، فاصله 450 متر نیز با توجه به سرعت نور محاسبه می شود که چند ثانیه و در چه محل زمانی قرار دارد. آنگاه با استفاده از محل زمانی پیدا شده،

مقدار سیگنال دریافتی در محل موردنظر یک شده و در نتیجه سیگنال دریافتی نیز پیدا و سپس با دستور plot رسم می شود.

در کد زیر هم سیگنال دریافتی و هم سیگنال ارسالی و نحوه رسم آن نوشته شده است.

```
ts = 1e-9;
T = 1e-5;
tau = 1e-6;
t=0:ts:T;
t_len=length(t);
sent_signal = zeros(1,t_len);
sent_signal(1:round(tau/ts))=1;

subplot(2,2,1);
plot(t,sent_signal);
ylabel("Amplitude");
xlabel("Time");
hold on;

alpha = 0.5;
R = 450;
C = 3e8; %speed of lights
recieved = zeros(1,t_len);

td = 2*R/C;
recieved(round(td/ts):round((td+tau)/ts)-1) = alpha*1;
plot(t,recieved);
legend('sent signal','recieved signal');
hold on;
```



(3-3

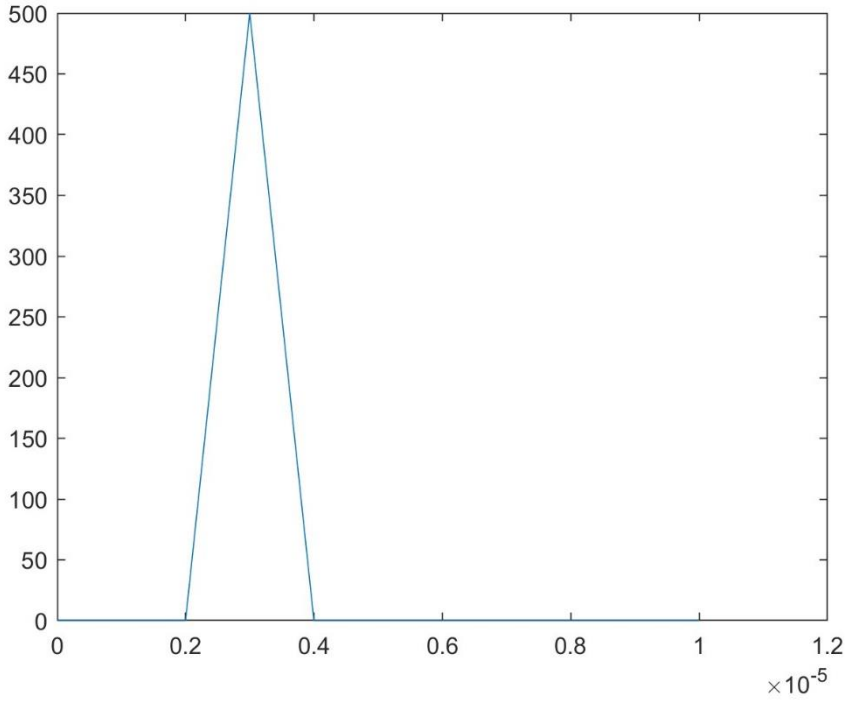
```
clc
clear
ts = 1e-9;
T = 1e-5;
tau = 1e-6;
t=0:ts:T;
t_len=length(t);
sent_signal = zeros(1,t_len);
sent_signal(1:round(tau/ts))=1;

subplot(2,2,1);
plot(t,sent_signal);
ylabel("Amplitude");
xlabel("Time");
hold on;

alpha = 0.5;
R = 450;
C = 3e8; %speed of lights
recieved = zeros(1,t_len);

td = 2*R/C;
recieved(round(td/ts):round((td+tau)/ts)-1) = alpha*1;
noise = 0.1*randn(1,t_len);
recieved=noise + recieved;
plot(t,recieved);
legend('sent signal','recieved signal');
hold on;
temp = ones(1,round(tau/ts));
ro=zeros(1,t_len);
for i=1:t_len-round(tau/ts)
    ro(i)=sum(recieved(i:i+round(tau/ts)-1).*temp);
end
subplot(2,2,2);
plot(t,ro);
legend('correlation result');
[amplitude, distance_in_time] = max(ro);
distance=distance_in_time*1e-9*C/2;
disp(distance);
```

Command Window

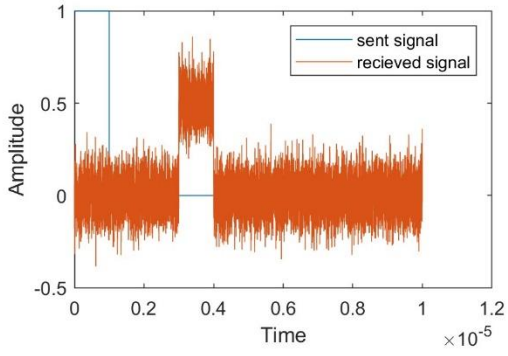
```
    450
```

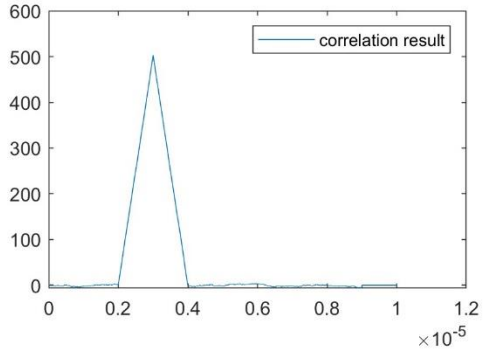
با حلقه for، سیگنال دریافتی با یک سیگنال که در بازه معین و برابر با بازه ای که سیگنال در آن 1 است، برابر 1 است ضرب داخلی شده و در نهایت به روش correlation مکان جسم شناسایی می شود.

درواقع حاصل این ضرب داخلی همه جا صفر است غیر از جایی که سیگنال با جسم همپوشانی دارد، و مقدار این ضرب داخلی هرجا همپوشانی بیشتر باشد بالاتر است، لذا با correlation و یافتن نقطه ماکسیمم حاصل از این روش مکان جسم شناسایی می شود.



(4-3

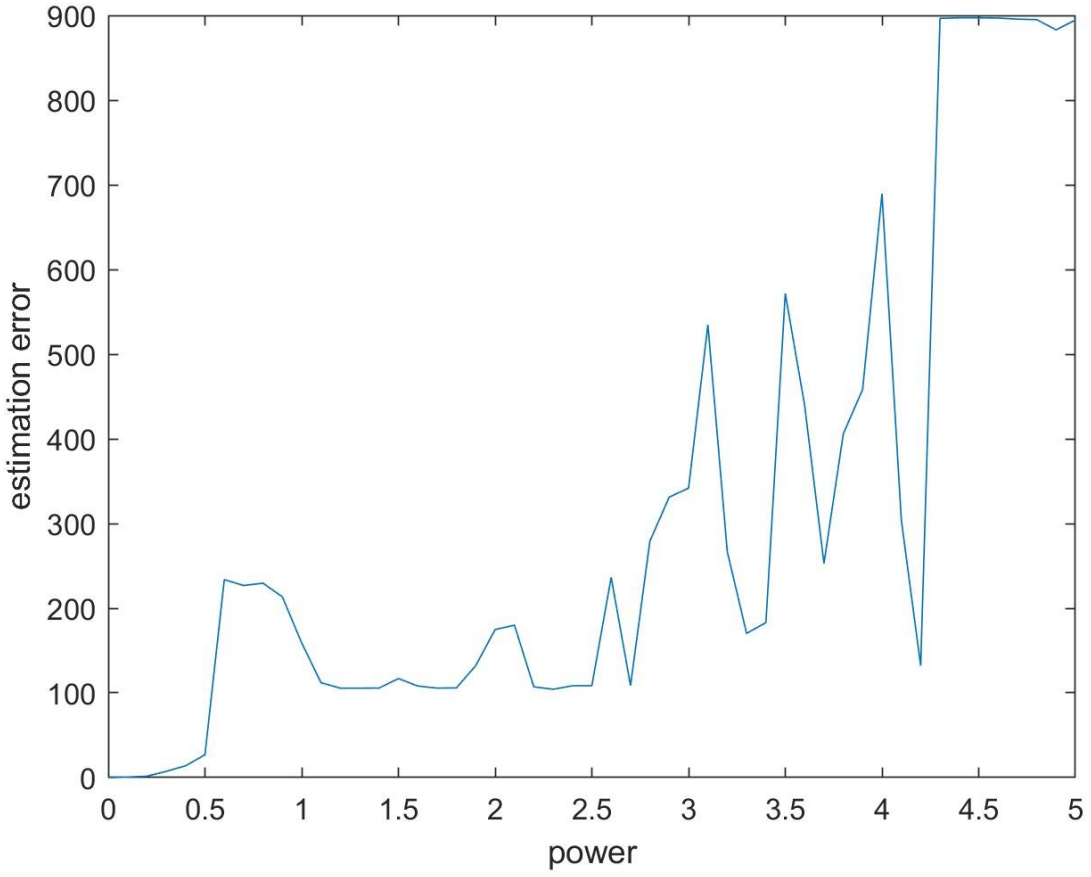




نمودار بدست آمده با نویز کم به صورت بالا می باشد. حال برای یک بازه از نویز:

```
clc
clear
ts = 1e-9;
T = 1e-5;
tau = 1e-6;
t=0:ts:T;
t_len=length(t);
sent_signal = zeros(1,t_len);
sent_signal(1:round(tau/ts))=1;

alpha = 0.5;
R = 450;
C = 3e8; %speed of lights
recieved = zeros(1,t_len);

td = 2*R/C;
temp = ones(1,round(tau/ts));
ro=zeros(1,t_len);
recieved(round(td/ts):round((td+tau)/ts)-1) = alpha*1;
powers = 0 :0.1:5;
error_of_each_power = zeros(50,2);
p=1;
currentError=zeros(100,1);
```

```
p=1;
currentError=zeros(100,1);
for power = 0 :0.1:5
    currentError=zeros(100,1);
    for iteration = 1:100
        noise = power*randn(1,t_len);
        recieved = recieved + noise;
        ro = zeros(1,t_len);
        for i=1:t_len-round(tau/ts)
            ro(i)=sum(recieved(i:i+round(tau/ts)-1).*temp);
        end
        [amplitude, distance_in_time] = max(ro);
        distance=distance_in_time*1e-9*C/2;
        error = abs(R-distance);
        currentError(iteration,1) = error;
    end
    error_of_each_power(p,1) = [mean(currentError)];
    error_of_each_power(p,2) = power;
    p=p+1;
end
plot(powers, error_of_each_power(:,1)');
xlabel('power');
ylabel('estimation error');
for i=1:10
    if error_of_each_power(i,1)>=10
        fprintf('maximum noise power for 10m error is: %f', error_of_each_power(i,2));
        break;
    end
end
```




```
maximum noise power for 10m error is: 0.400000>>
```

همانطور که انتظار میرود، با افزایش نویز دقت سنجش فاصله کاهش می یابد.

همچنین با چندبار اجرای برنامه، قدرت نویز برای کمتر از 10 متر شدن خطا بین 0.3 تا 0.5 بدست آمد.

بخش چهارم:

1-4)

Sampling frequency نشان می دهد در هر ثانیه چند نمونه از سیگنال ورودی (در اینجا صدای ضبط شده) گرفته شده است.

p4_1.m

```
clc
clear
[x, sampleFrequency] = audioread("recording.wav");
fprintf("sampling frequency: %d \n" , sampleFrequency);

```

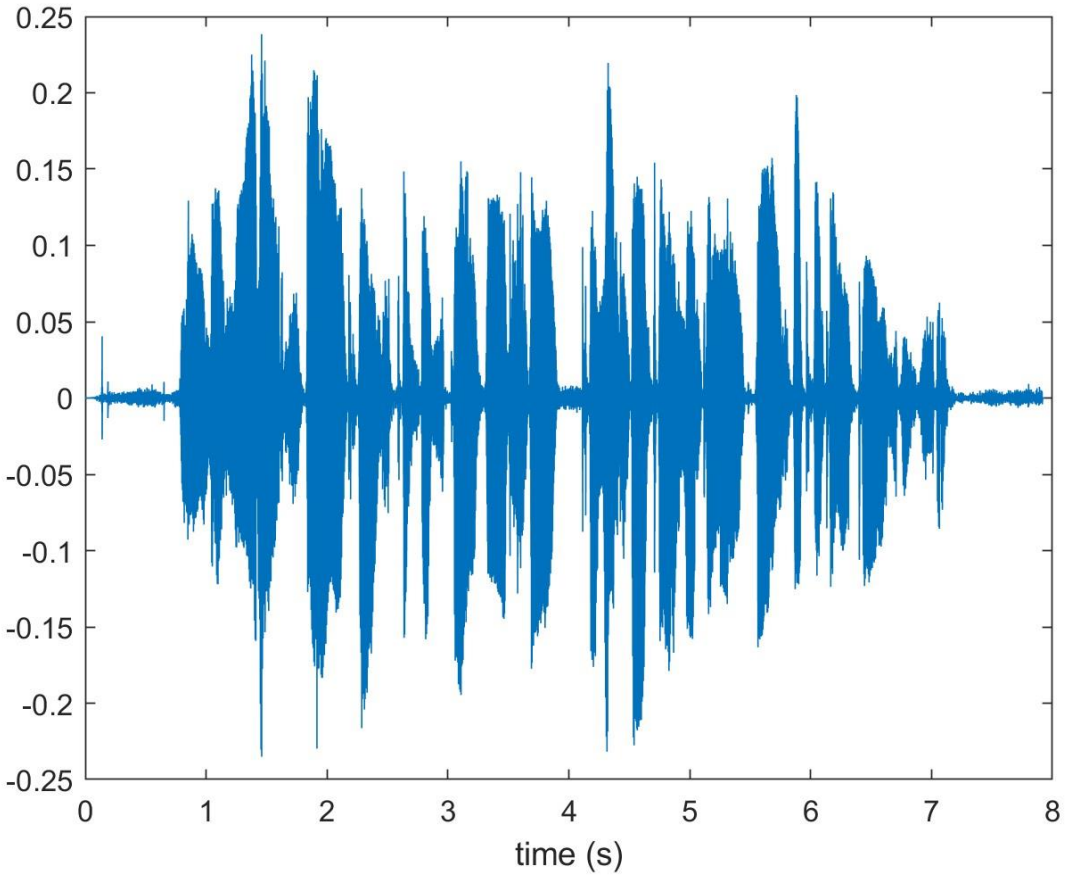
Command Window

```
sampling frequency: 48000
```

(2-4

با دستور پلات حاصل نمونه گیری از صدا رسم می شود. در واقع طول فایل را بدست آوردیم، آنگاه بر فرکانس نمونه گیری تقسیم کردیم.

```
clc
clear
[x, sampleFrequency] = audioread("recording.wav");
N=length(x);
t = linspace(0, N/sampleFrequency, N);
plot(t, x);
xlabel('time (s)');
```



(3-4

جهت 0.5 برابر کردن سرعت، x جدیدی تعریف میکنیم که طولش دوبرابر طول فایل ورودی اولیه باشد. سپس بین هر دو سمپل از صدا که پشت سر هم هستند میانگین گرفته شده و درنهایت صدا با سرعت نصف صدای اولیه تولید می شود.

برای 2 برابر کردن سرعت،x جدیدی تعریف میکنیم که در آن از هر دو سمپل از صدای ورودی یکی حذف شده (یکی درمیان ذخیره میکنیم) و خروجی به گونه ای است که صدا با همان فرکانس سمپل گیری قبلی ولی با سرعت دو برابر تولید می شود.

p4_3.m

```
function p4_3 (voice, speed)
    if speed~= 2 && speed ~=0.5
        error('Invalid speed input!');
    end
    [x, samplingFrequency] = audioread(voice);

    if speed == 2
        newX = x(1:2:end);
    end
    if speed == 0.5
        newX = zeros(2*length(x)-1, size(x,2));
        newX(1:2:end , :) = x;
        newX(2:2:end-1,:) = (x(1:end-1,:)+x(2:end,:)) /2;
    end
    audiowrite("newVoice.wav", newX, samplingFrequency);
end
```

(4-4

p4_4.m

```
function p4_4 (voice, speed)
    if speed > 2 || speed < 0.5
        error('Invalid speed input!');
    end
    [x, samplingFrequency] = audioread(voice);

    speed = round(speed,1);
    newX = x(round(linspace(1,size(x,1), size(x,1)/speed)));
    audiowrite("newVoice.wav", newX, samplingFrequency);
end
```

در واقع بردار (وکتور) خطی جدیدی می سازیم که همان سایز x را با گام هایی به طول size(x,1)/round(speed,1) طی میکند.