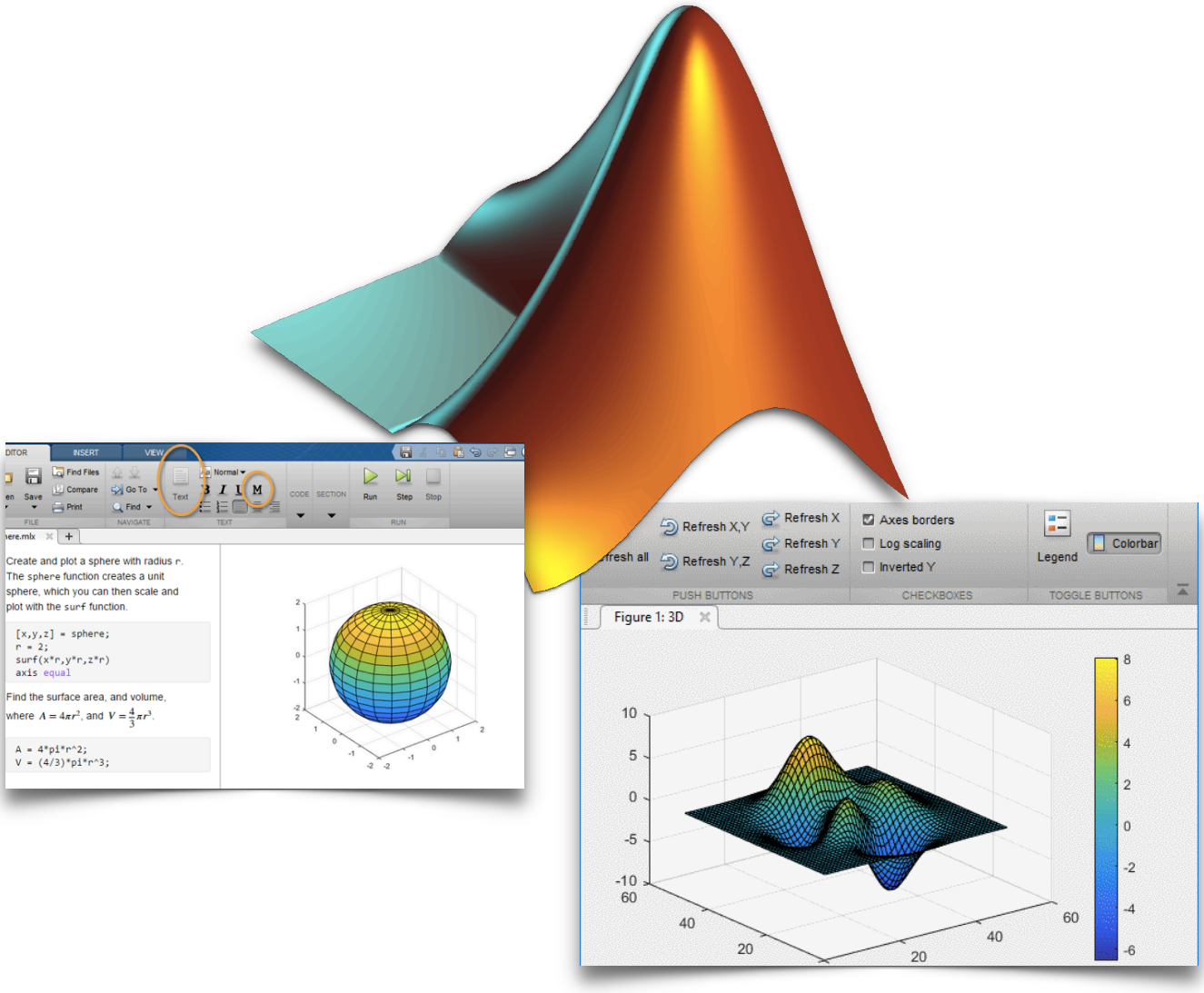
Project 1

Introduction to MATLAB



Part 1

Part 1.1

- توضیح خط به خط هر کد :

/MATLAB Drive/p1_1.m

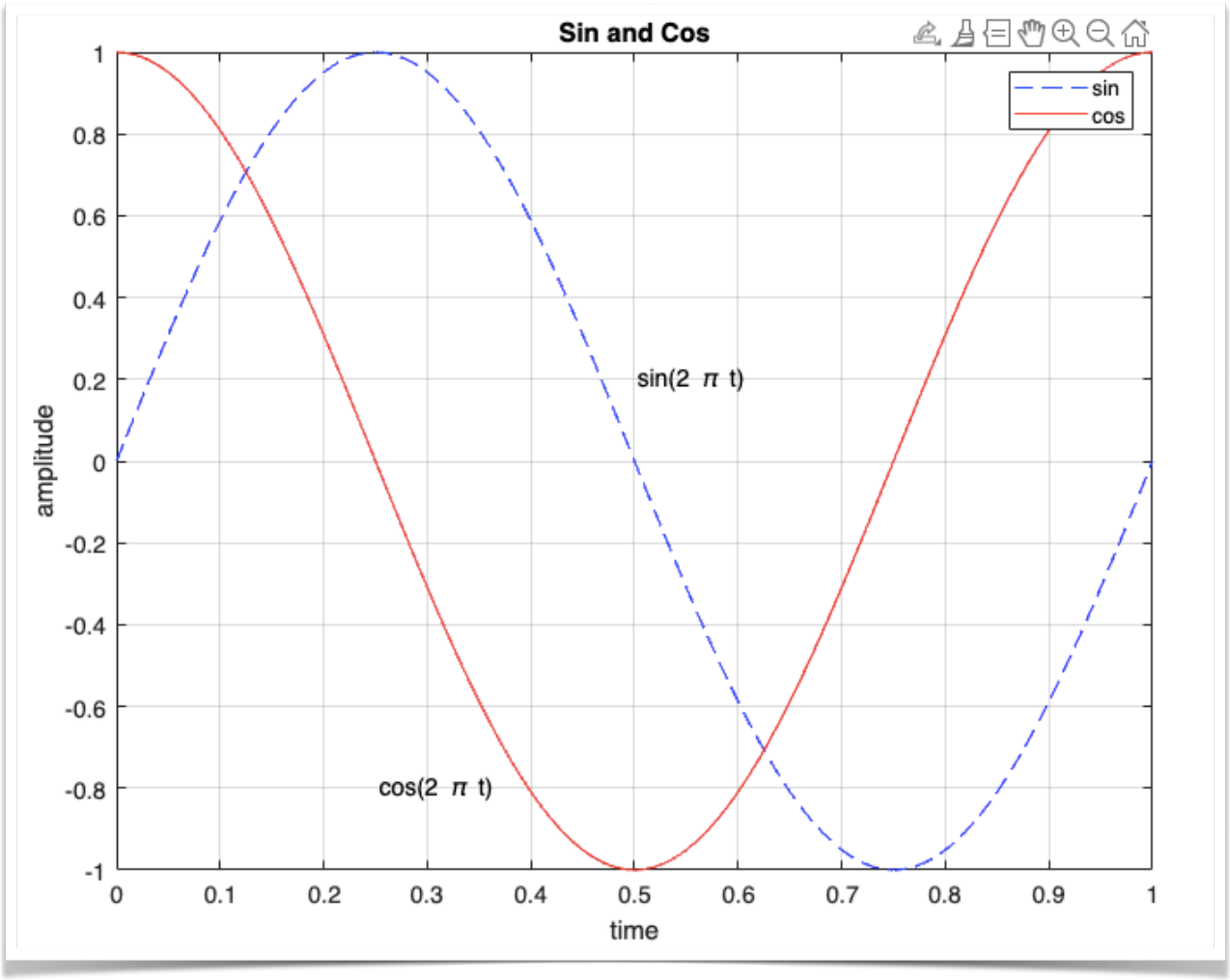
```
1   t = 0 : 0.01 : 1;
2   z1 = sin( 2 * pi * t );
3   z2 = cos( 2 * pi * t );
4
5
6   figure;
7   plot (t , z1 , '--b')
8   hold on
9   plot (t , z2 , 'r')
10
11  x0 = [0.5 ; 0.25];
12  y0 = [0.2 ; -0.8];
13
14
15  s = ['sin(2 \pi t)' ; 'cos(2 \pi t)'];
16  text(x0 , y0 , s); %add a comment at (x0 , y0)
17
18
19  title('Sin and Cos'); % Title
20  legend('sin' , 'cos') % add legends
21
22
23  xlabel('time') % the name of X-axis
24  ylabel('amplitude') % the name of Y-axis
25
26  grid on % add grid
27
28
```

Name	Value	Size	Class
s	'sin(2 \pi t)c...	1×24	char
t	*1×101 double*	1×101	double
x0	[0.5000;0.2...	2×1	double
y0	[0.2000;-0....	2×1	double
z1	*1×101 double*	1×101	double
z2	*1×101 double*	1×101	double

1. تخصیص یک ارایه از 0 تا 1 به گام های 0.01 ای
2. تخصیص یک ارایه به بعد ۱۰۰۰ برای سینوس دوپی برابر t
3. تخصیص یک ارایه به بعد ۱۰۰۰ برای کسینوس دوپی برابر t
6. تخصیص یک تصویر
7. کشیدن نموداری به محور افقی t و محور عمودی z1 و همچنین خطوط تکه تکه به رنگ آبی
8. نگه داشتن نمودار قبلی در کنار نمودار جدید
9. کشیدن نموداری به محور افقی t و محور عمودی z2 و همچنین خطوط به رنگ قرمز
11. تعریف ارایه ای به ابعاد 2*1
12. تعریف ارایه ای به ابعاد 2*1
15. تخصیص یک ارایه از متن به اندازه ی 2*1
16. نوشتن متون آرایه خط ۱۵ ، در نقاط تخصیص داده شده در خطوط ۱۱ و ۱۲
19. تخصیص تایتل sin and cos
20. کشیدن legend

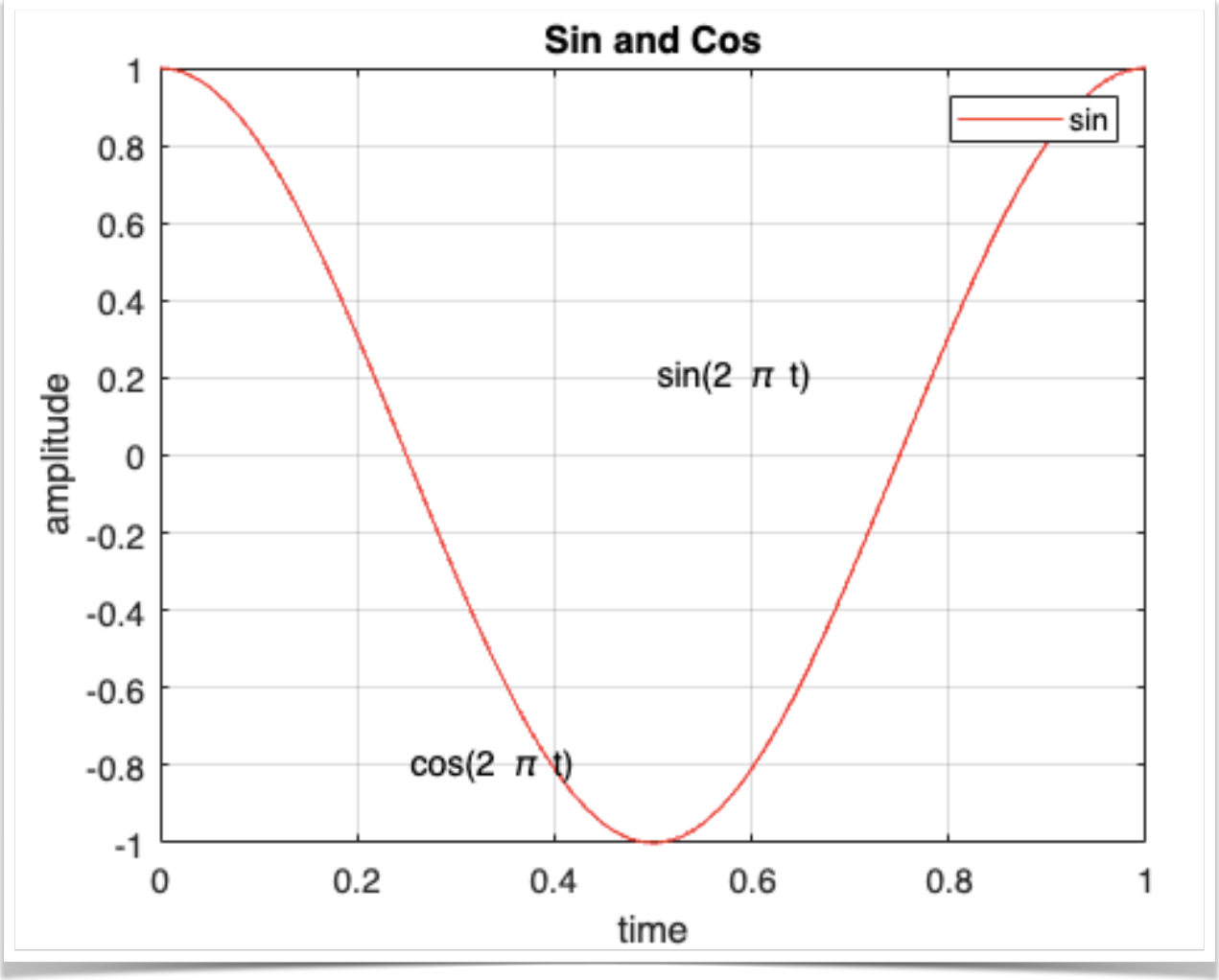
26. کشیدن خطوط کمرنگ gird روی نمودار

- شکل نهایی نمودار :

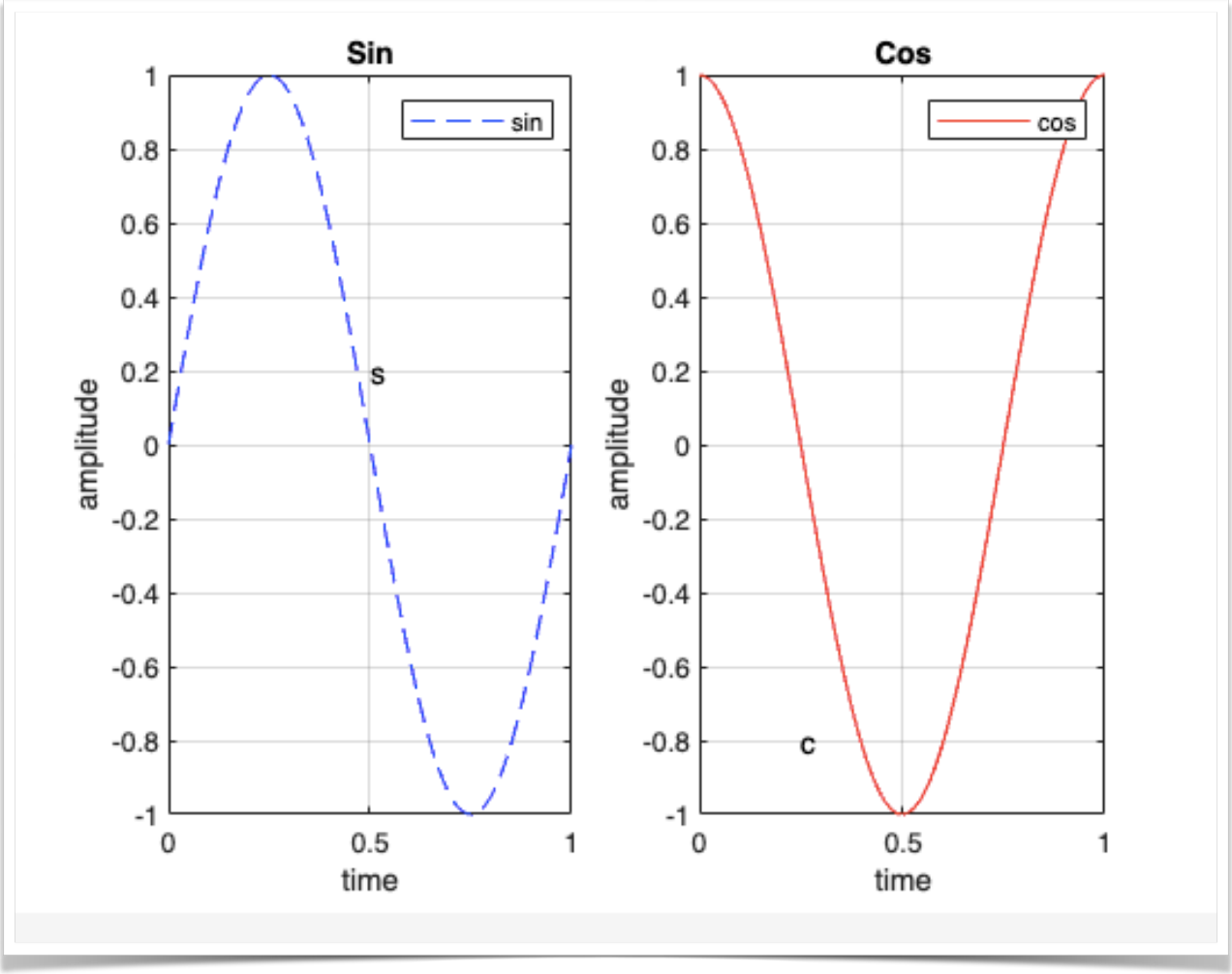


- اگر از دستور hold on استفاده نکنیم :

اگر از این دستور استفاده نکنیم دو نمودار باهم در یک شکل نمی آیند در واقع انگار شکل اول کشیده میشود و پاک میشود و سپس شکل دوم کشیده میشود .



Part 1.2

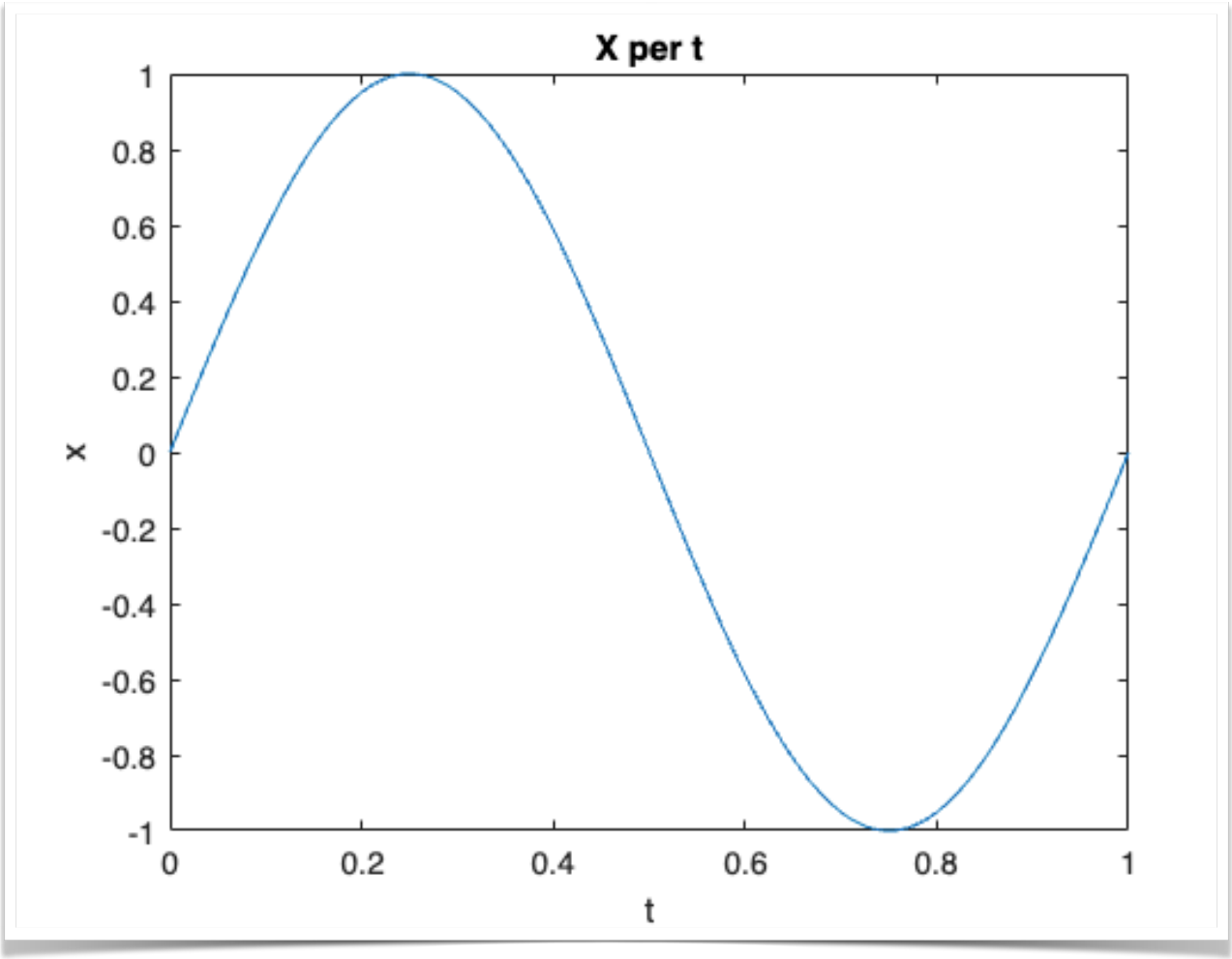


```
t = 0 : 0.01 : 1;
z1 = sin( 2 * pi * t );
z2 = cos( 2 * pi * t );



x0 = [0.5 ; 0.25];
y0 = [0.2 ; -0.8];
s = ['sin(2 \pi t)' ;'cos(2 \pi t)'];


figure;
subplot(1 , 2 , 1);
plot (t , z1 , '--b')
xlabel('time') % the name of X-axis
ylabel('amplitude') % the name of Y-axis
title('Sin'); % Title
legend('sin') % add legends
grid on % add grid
text(x0(1) , y0(1) , s(1)); %add a comment at (x0 , y0)



subplot(1 , 2 , 2);
plot (t , z2 , 'r')
xlabel('time') % the name of X-axis
ylabel('amplitude') % the name of Y-axis
title('Cos'); % Title
legend('cos') % add legends
grid on % add grid
text(x0(2) , y0(2) , s(2)); %add a comment at (x0 , y0)

```

Part 2

Part 2.1

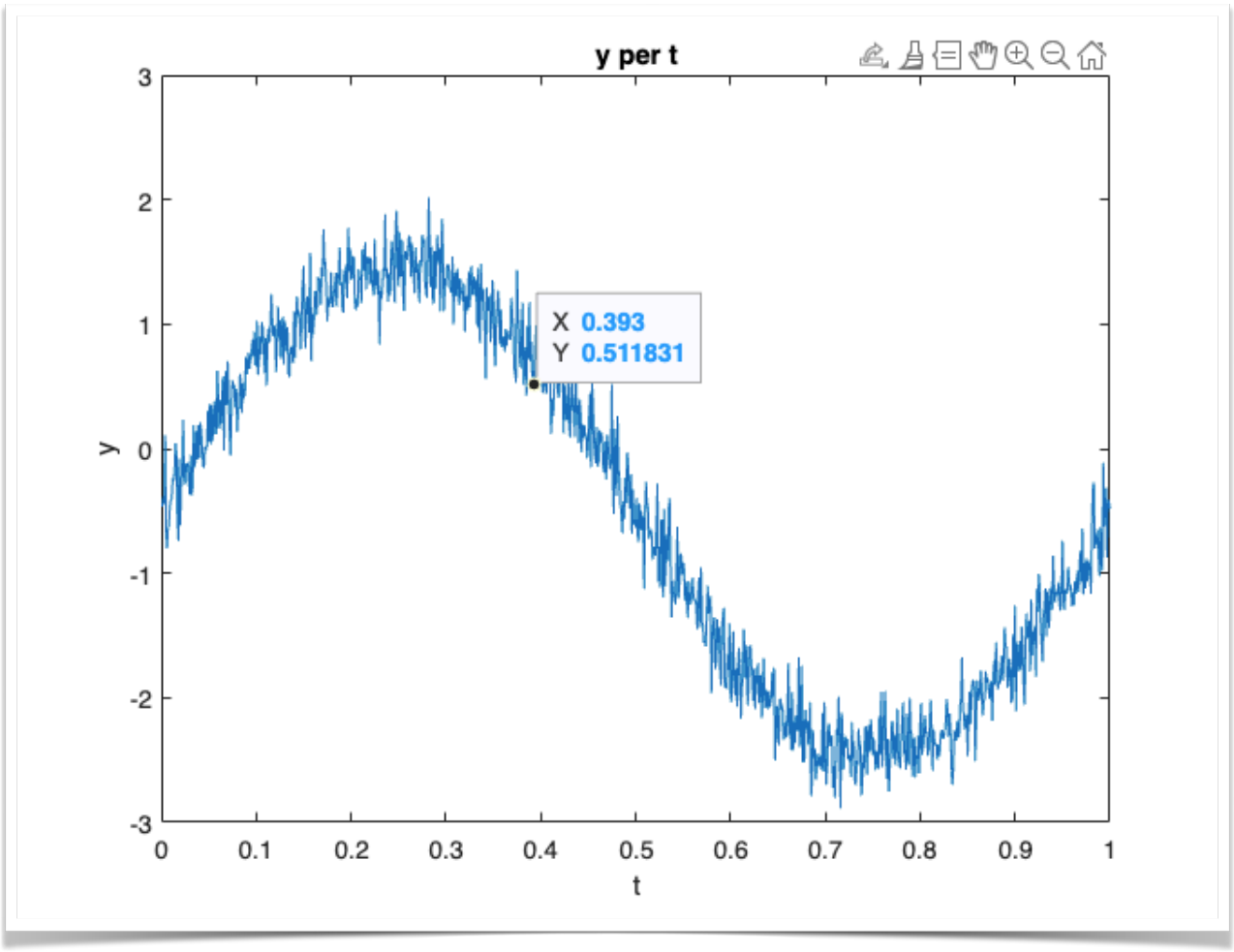
- نمودار x بر حسب t



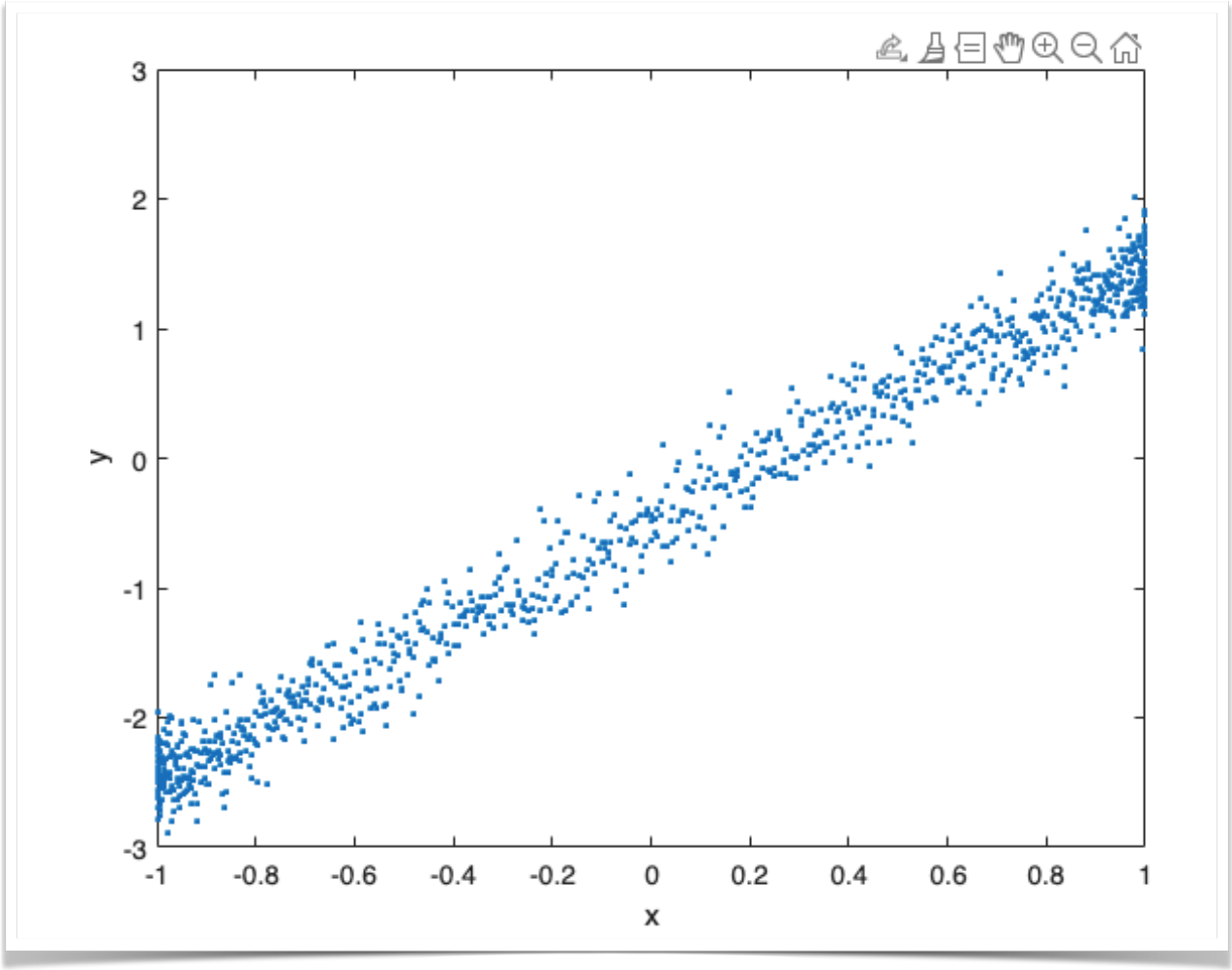
Part 2.2

- نمودار y بر حسب t

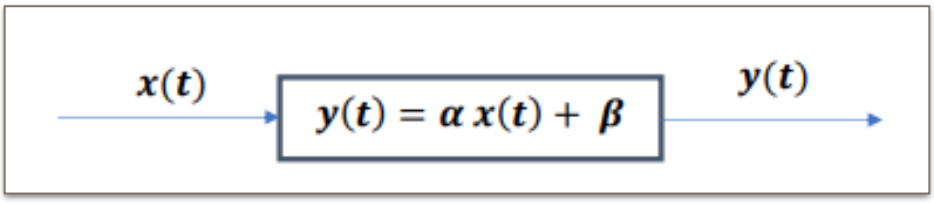
```
plot (t , y);title ("y per t");ylabel('y');xlabel('t');
```



Part 2.3

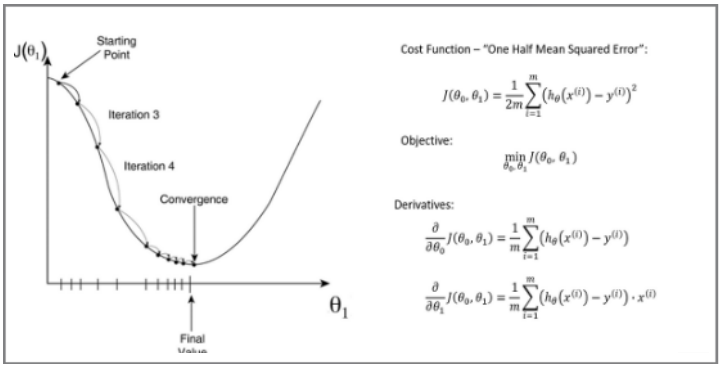


- چون رابطه ی سیستم رابطه ای خطی است پس میتوان گفت که از رابطه ی زیر پیروی میکنند :



- بنابراین شیب نمودار نمایانگر alpha و عرض از مبدا آن نمایانگر beta است .

Part 2.4



- با استفاده از روش gradient desent میتوانیم با گام های ثابت به سمت ضرایب بهینه حرکت کنیم به این صورت که ابتدا دو مقدار رندوم برای alpha ، beta در نظر میگیریم و سپس مقدار خطا را حساب میکنیم و با گام های learning_rate میتوانیم به سمت ضرایب مناسب حرکت کنیم به این صورت که مشتق تابع هزینه نسبت به الفا x میشود ، و همین طور مشتق تابع هزینه نسبت به بتا مقدار 1 میشود .

```
function [alpha , beta] = p2_4(x, y)

    if length(x) ~= length(y)
        error('Input vectors x and y must have the same length.');
    end

    alpha = rand();
    beta = rand();

    learning_rate = 0.01;
    num_iterations = 1000;

    for iter = 1:num_iterations

        y_pred = alpha * x + beta;

        error = y_pred - y;


        alpha = alpha - learning_rate * (2 / length(x)) * sum(error .* x);
        beta = beta - learning_rate * (2 / length(x)) * sum(error);

        cost = sum(error.^2) / (2 * length(x));
        fprintf('Iteration %d: Cost = %f\n', iter, cost);
    end

    fprintf('Final Slope (m) = %f\n', alpha);
    fprintf('Final Intercept (b) = %f\n', beta);


end
```

- نمونه ی اولیه بدون نویز :

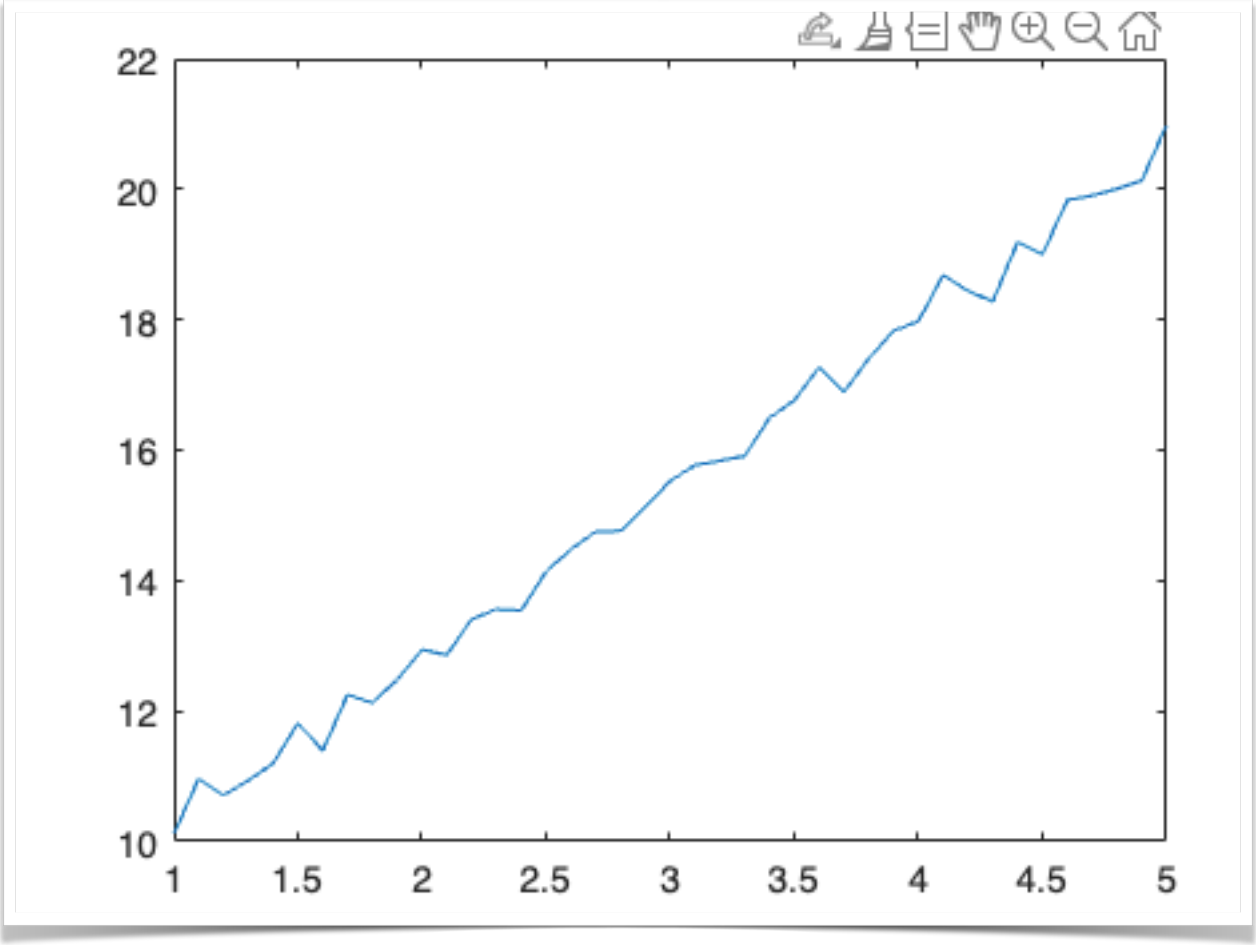
```
>> testx = [ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10];
>> testy = 3.3*testx + 7.7 ;
>> testy

testy =

   11.0000   14.3000   17.6000   20.9000   24.2000   27.5000   30.8000   34.1000   37.4000   40.7000

>> |
```

```
Iteration 99998: Cost = 0.000000
Iteration 99999: Cost = 0.000000
Iteration 100000: Cost = 0.000000
Final Slope (m) = 3.300000
Final Intercept (b) = 7.700000

ans =

    3.3000

>> ;
```

- نمونه ی اولیه با نویز کم :

- برای اینکه نویز کم باشد سعی کردم اون رو به روش min_max اسکیل کنم تا فقط مقدار های نویز میان ۰ تا ۱ باشند .

```
1
2   alpha = 2.6;
3   beta = 7.1;
4
5   % x values
6   testx = 1 : 0.1 :5 ;
7
8
9   % noise and scale it between 0 to 1
10  noise = randn (1 , length(testx));
11
12  min_noise = min(noise);
13  max_noise = max(noise);
14
15  scaled_noise = (noise - min_noise) / (max_noise - min_noise);
16
17  % y values with noise and linear fucntion
18  testy = alpha*testx + beta + scaled_noise;
19
20  plot (testx , testy);
21  fprintf ("real Alpha is %f and real Beta is %f" , alpha , beta);
22  [guessa , guessb] = p2_4(testx , testy);
23  fprintf ("guessed Alpha is %f and guessed Beta is %f" , guessa , guessb)
24
```

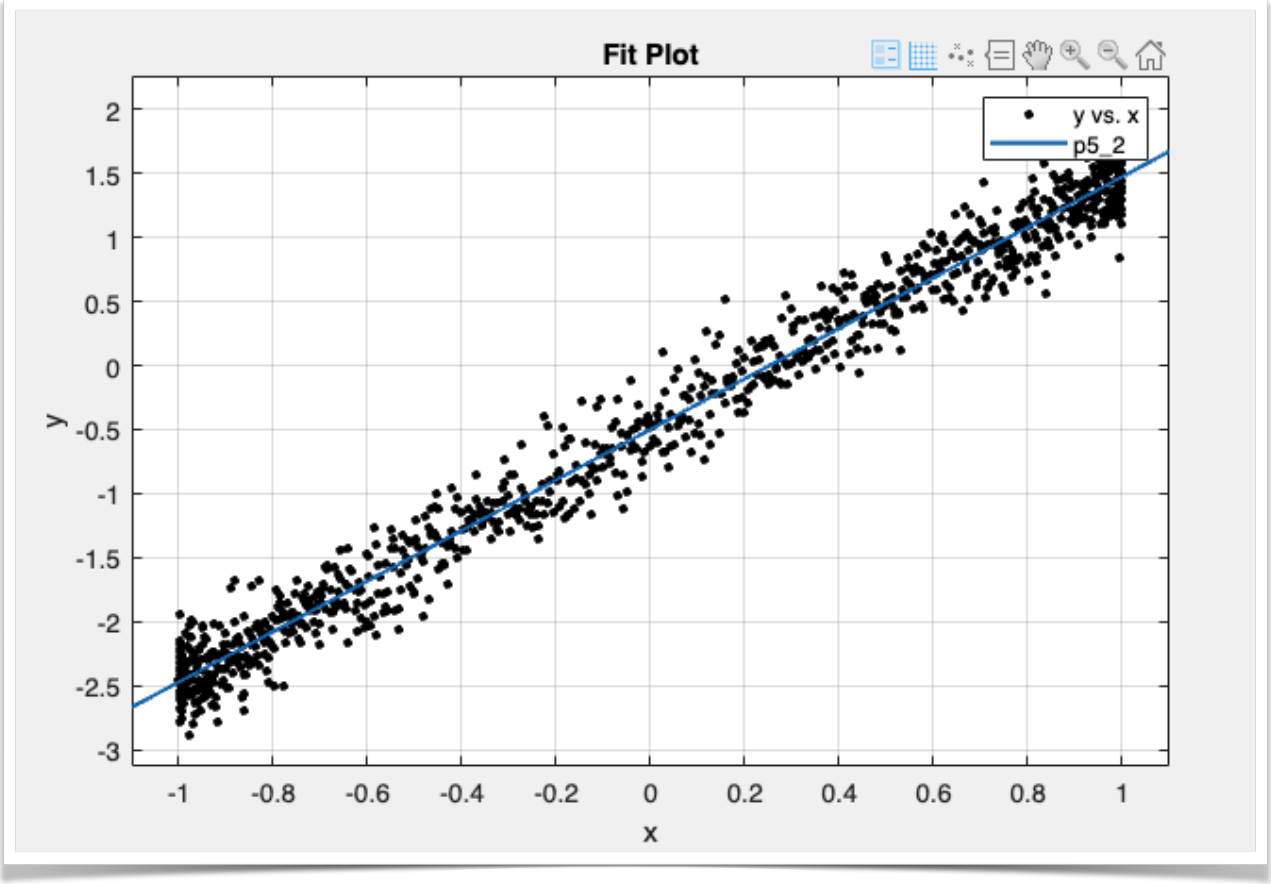



```
Iteration 99992: Cost = 0.031019
Iteration 99993: Cost = 0.031019
Iteration 99994: Cost = 0.031019
Iteration 99995: Cost = 0.031019
Iteration 99996: Cost = 0.031019
Iteration 99997: Cost = 0.031019
Iteration 99998: Cost = 0.031019
Iteration 99999: Cost = 0.031019
Iteration 100000: Cost = 0.031019
Final Slope (m) = 2.601633
Final Intercept (b) = 7.485813
>>
```

- فراخوانی تابع بر روی مقادیر داده شده در صورت پروژه :

```
Iteration 99974: Cost = 0.019912
Iteration 99975: Cost = 0.019912
Iteration 99976: Cost = 0.019912
Iteration 99977: Cost = 0.019912
Iteration 99978: Cost = 0.019912
Iteration 99979: Cost = 0.019912
Iteration 99980: Cost = 0.019912
Iteration 99981: Cost = 0.019912
Iteration 99982: Cost = 0.019912
Iteration 99983: Cost = 0.019912
Iteration 99984: Cost = 0.019912
Iteration 99985: Cost = 0.019912
Iteration 99986: Cost = 0.019912
Iteration 99987: Cost = 0.019912
Iteration 99988: Cost = 0.019912
Iteration 99989: Cost = 0.019912
Iteration 99990: Cost = 0.019912
Iteration 99991: Cost = 0.019912
Iteration 99992: Cost = 0.019912
Iteration 99993: Cost = 0.019912
Iteration 99994: Cost = 0.019912
Iteration 99995: Cost = 0.019912
Iteration 99996: Cost = 0.019912
Iteration 99997: Cost = 0.019912
Iteration 99998: Cost = 0.019912
Iteration 99999: Cost = 0.019912
Iteration 100000: Cost = 0.019912
Final Slope (m) = 1.973579
Final Intercept (b) = -0.498339

ans =

    1.9736

>>
```

Part 2.5



Coefficients and 95% Confidence Bounds

	Value	Lower	Upper
p1	1.9736	1.9560	1.9911
p2	-0.4983	-0.5107	-0.4859

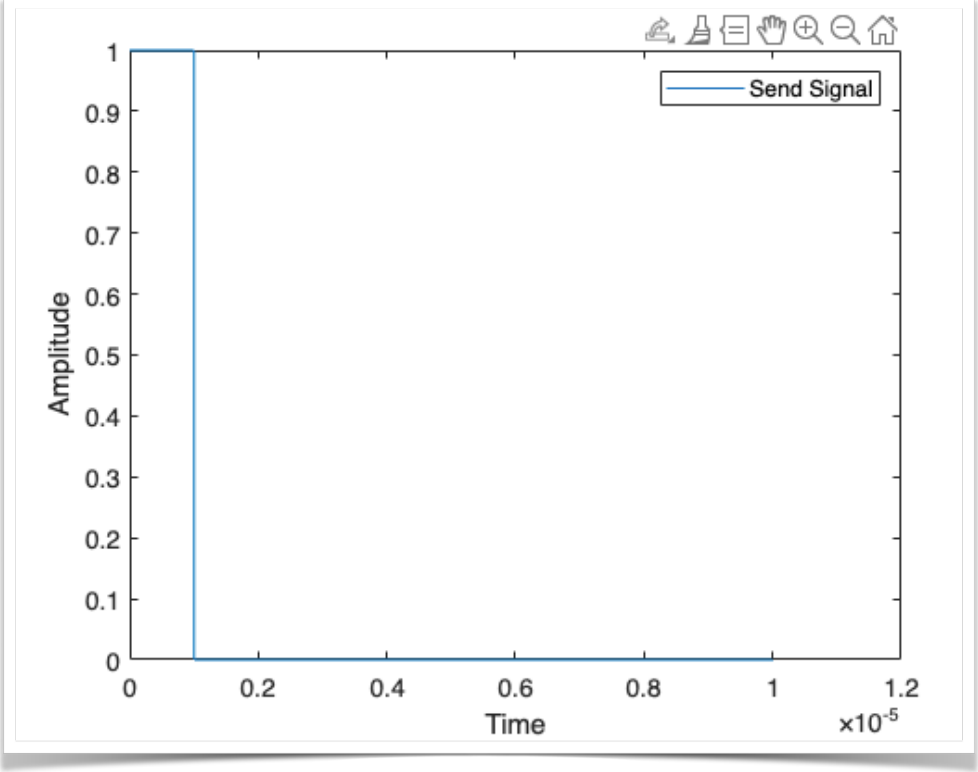

CURVE FITTER
New
Open
Save
Duplicate
FILE
Select Data
Exclusion Rules
Validation Data
DATA
Polynomial
Exponential
FIT TYPE
Auto
Manual
Fit
FIT
Fit Plot
Residuals Plot
Contour Plot
Prediction Bounds
None
VISUALIZATION
Export
EXPORT
p5_2
Fit Plot
y vs. x
p5_2
x
y
Fit Options
Polynomial
Degree
1
Robust
Off
Center and scale
Read about fit options
Results
Value
Lower
Upper
p1
1.9736
1.9560
1.9911
p2
-0.4983
-0.5107
-0.4859
Table of Fits
DFE
Adj R-...
RMSE
Co...
Validation D...
Validation S...
Validation RM...
999
0.97992
0.19976
2
SSE
39.8638
R-square
0.9799
DFE
999.0000
Adj R-sq
0.9799
RMSE
0.1998

- مقایسه ی مقادیر بدست آمده از تابع نوشته شده با مقادیر curve fitter
  - مقادیر محاسبه شده در تابع من
    - Alpha =1.973579
    - Beta = -0.498339
  - مقادیر محاسبه شده در curve fitter
    - Alpha = 1.9736
    - Beta = -0.4983

Part 3

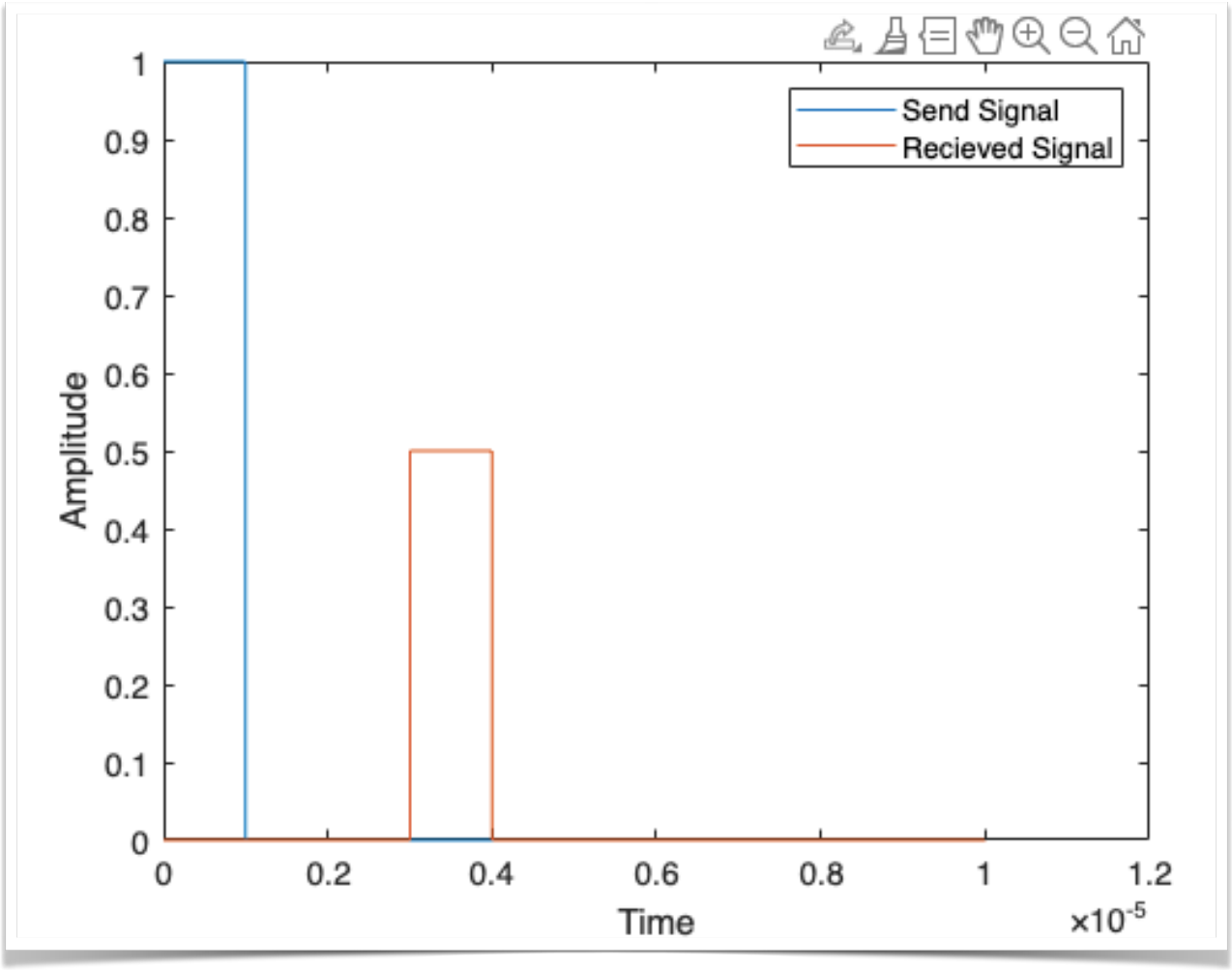
Part 3.1

```
ts = 1e-9;  % Sampling time
T = 1e-5;   % Total time duration
tau = 1e-6; % Time constant
t=0:ts:T;
tlen=length(t);
x=zeros(1,tlen);
x(1:round(tau/ts))=1;


plot(t,x );
hold on;

```



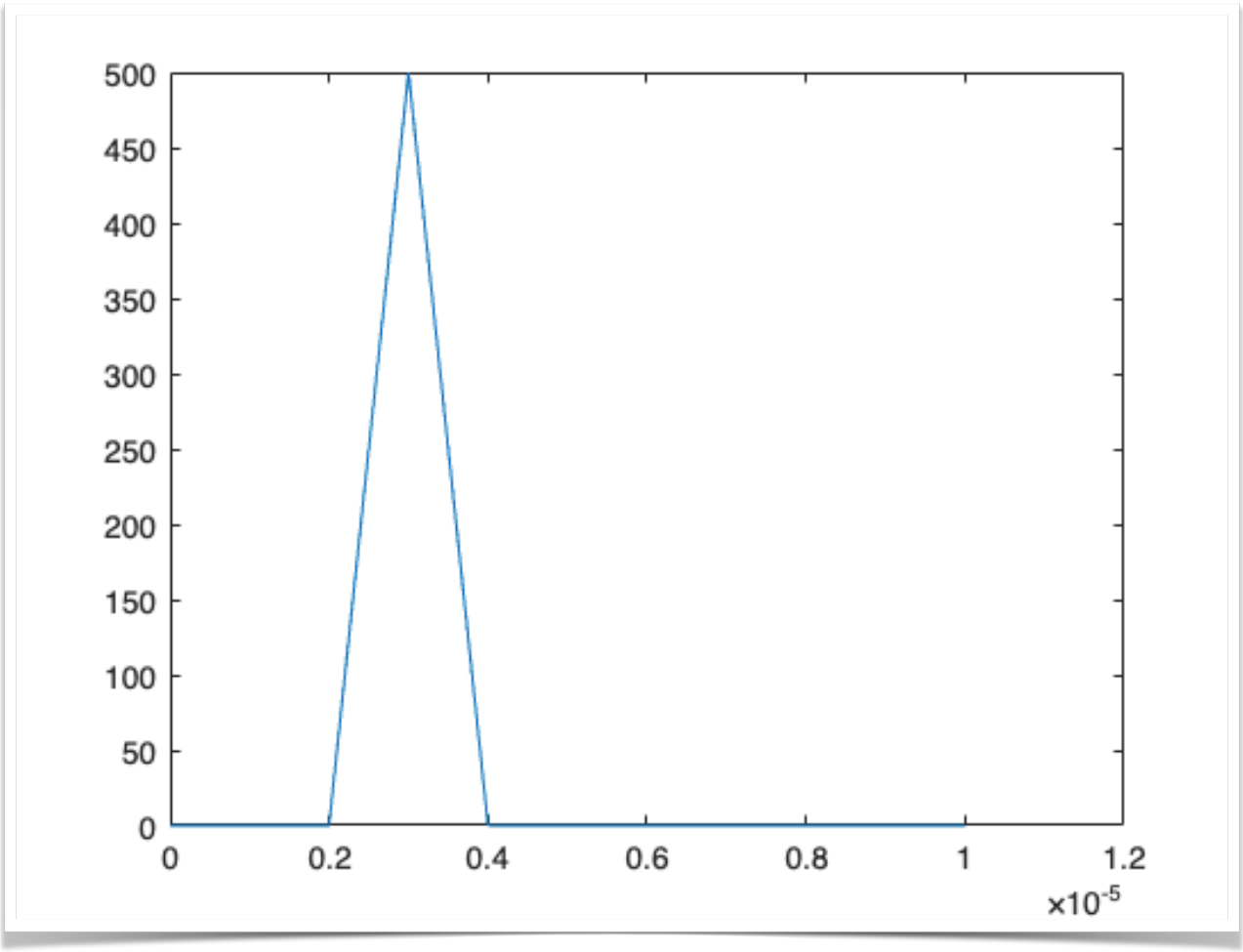
Part 3.2

```
ts = 1e-9;  % Sampling time
T = 1e-5;   % Total time duration
tau = 1e-6; % Time constant
t=0:ts:T;
tlen=length(t);
sent=zeros(1,tlen);
sent(1:round(tau/ts))=1;


plot(t,sent);
xlabel("Time");
ylabel("Amplitude");
hold on;


alpha = 0.5 ;
recieved = zeros(1,tlen);
speedOfLight = 3e8;
R = 450;
td = 2 * R / speedOfLight;
recieved(round(td/ts) : round((td+tau)/ts)-1 ) = 1 * alpha;



plot(t,recieved);
legend("Send Signal" ,"Recieved Signal");
hold on;

```


Send Signal
Recieved Signal
Amplitude
Time
×10^-5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2

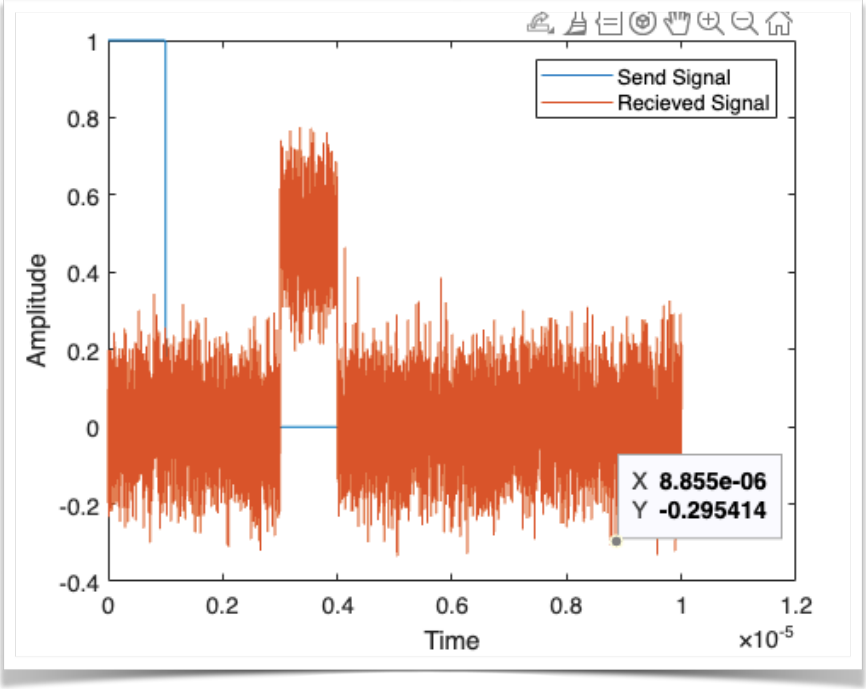
Part 3.3

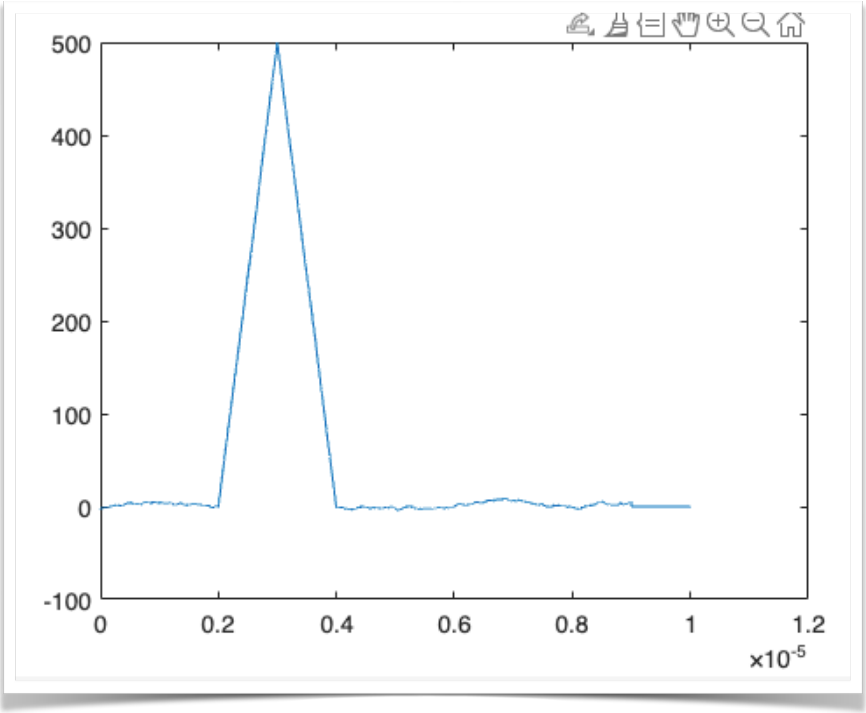
```
ts = 1e-9;  % Sampling time
T = 1e-5;   % Total time duration
tau = 1e-6; % Time constant
t=0:ts:T;
tlen=length(t);
sent=zeros(1,tlen);
sent(1:round(tau/ts))=1;


plot(t,sent);
xlabel("power of noise");
ylabel("error of distance emstimation ");
hold on;


alpha = 0.5 ;
recieved = zeros(1,tlen);
speedOfLight = 3e8;
R = 450;
td = 2 * R / speedOfLight;
recieved(round(td/ts) : round((td+tau)/ts)-1 ) = 1 * alpha;


plot(t , recieved);
legend("Send signal" , "Recieved Signal");
hold on;



ro=zeros(1,length(t));
for i=1:length(t)-round(tau/ts)
    tempp=zeros(1,length(t));
    tempp(i:i+round(tau/ts)-1)=1;
    ro(i)=sum(tempp .* recieved);
end
figure
plot(t , ro);


[amplitude , timeDist] = max(ro);
timeDist = timeDist *1e-9
distance = timeDist * speedOfLight /2

```




```
>> p3_1_withoutNoise

timeDist =

   3.0000e-06

distance =

   450

>>
```

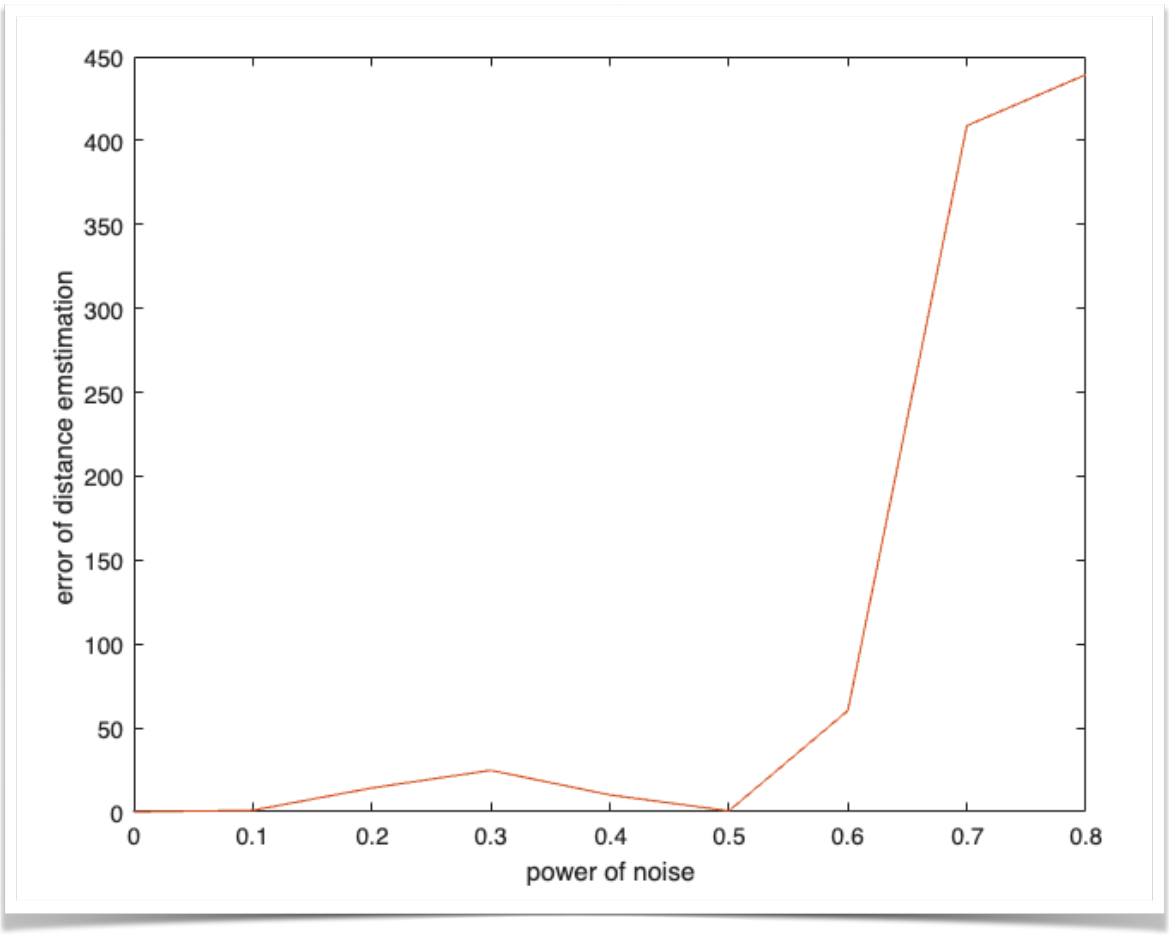
Part 3.4


Send Signal
Recieved Signal
Amplitude
Time
×10^-5
X 8.855e-06
Y -0.295414


×10^-5

- حالا یک لوپ برای هر قدرت نویز (powerOfNoise) در نظر میگیرم و هر بار یک نویز با آن قدرت تولید میکنیم و سپس خطا فاصله یابی را بدست می آوریم و میانگین این خطا ها را خطای آن قدرت گزارش میکنیم .

```
alpha = 0.5 ;
recieved = zeros(1,tlen);
speedOfLight = 3e8;
R = 450;
td = 2 * R / speedOfLight;
recieved(round(td/ts) : round((td+tau)/ts)-1 ) = 1 * alpha;

powers = 0 : 0.1 : 1 ;
errorsOfPowers = [];

for power= 0 : 0.1 : 1
    allErrorsOfThisPower = [];
    for iteration=1:100

        noise = power*randn(1 , tlen);
        recieved = recieved + noise;



        ro=zeros(1,length(t));
        for i=1:length(t)-round(tau/ts)
            tempp=zeros(1,length(t));
            tempp(i:i+round(tau/ts)-1)=1;
            ro(i)=sum(tempp .* recieved);
        end



        [amplitude , timeDist] = max(ro);
        timeDist = timeDist * 1e-9;
        distance = timeDist * speedOfLight /2;

        realDistance = 450;

        allErrorsOfThisPower = [allErrorsOfThisPower , abs(realDistance - distance) ];


    end

    errorsOfPowers = [errorsOfPowers , mean(allErrorsOfThisPower)];
end

plot(powers , errorsOfPowers);
```



- همانطور که انتظار داریم به مرور با افزایش قدرت نویز خطای ما رو به افزایش میرود . این خطای گزارش شده برای هر مقدار آلفای قدرت نویز میانگین چندیدن بار تست است . به این دلیل که نویز ماهیت تصادفی دارد

Part 4

Part 4.1

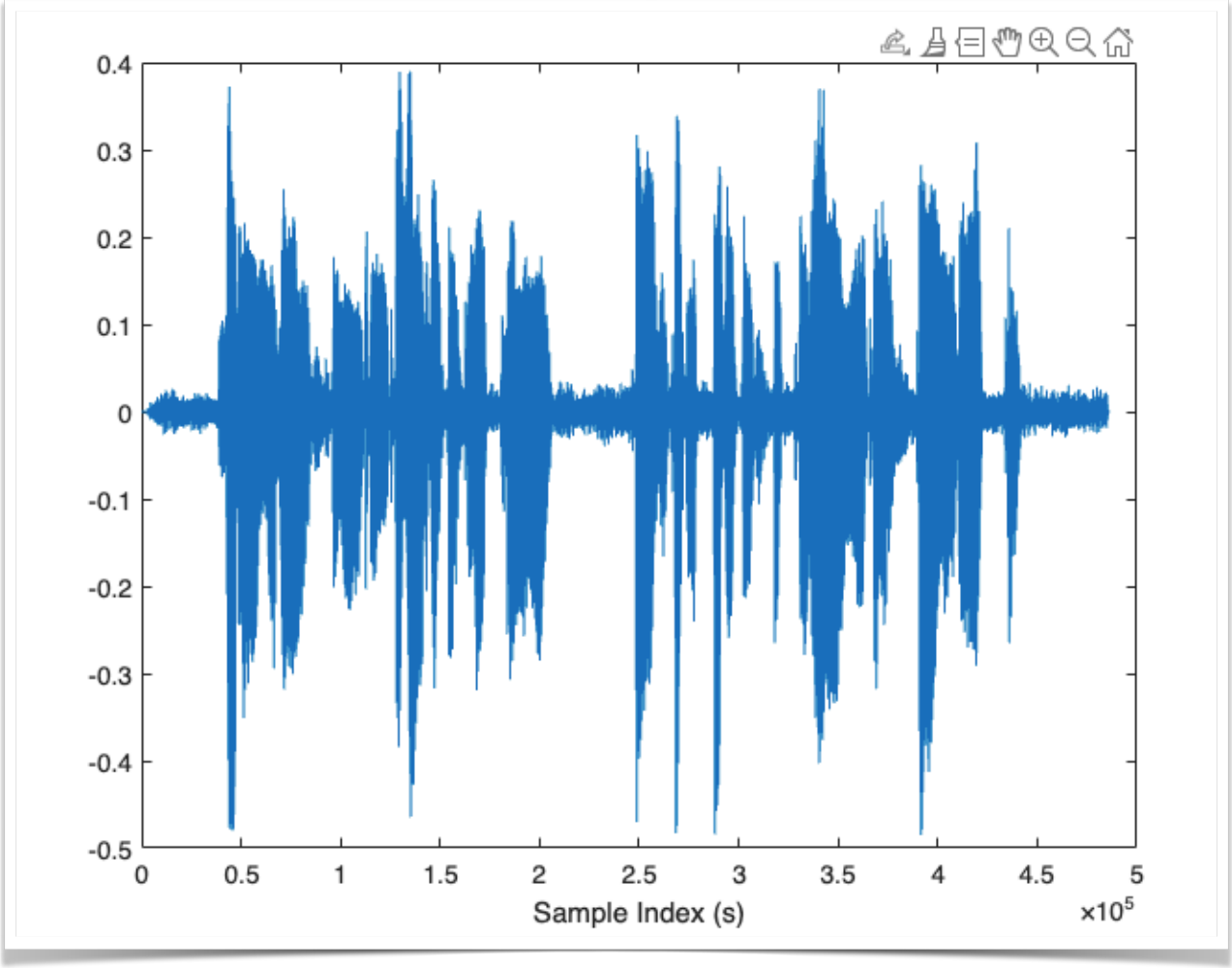
- فرکانس نمونه برداری نشان می دهد که در هر ثانیه، چند نمونه از سیگنال پیوسته گرفته شده است

```
1
2       [x,samplingFrequency ]= audioread("myVoice.wav" );
3
4       fprintf("Sampleing Frequency is : %d" , samplingFrequency);
5
6
```

```
>> p4_3
Sampleing Frequency is : 48000
>>
```

```
audiowrite("newVoice.wav" , x , Fs);
```

Part 4.2




```
[x,samplingFrequency ]= audioread("myVoice.wav" );

fprintf("Sampleing Frequency is : %d" , samplingFrequency);

%
% plot (x);
% xlabel("Sample Index");
% ylabel("Amplitude");s
%
%
timeArray = 1 : length(x) ;

timeArray = timeArray';

plot (timeArray , x);
xlabel("Sample Index (s)")
```

Part 4.3

- برای دوبرابر کردن سرعت صدا ، با طی کردن یکی در میان بین ایندکس های نمونه مقدار هر یک را حذف میکنیم بدین صورت انگار طول ارایه x را نصف کرده ایم و دست به Sample Frequecy نزده ایم برای همین سرعت صدا دو برابر میشود.
- همچنین برای نیم برابر کردن سرعت صدا باید یک آرایه به طول دو برابر آرایه اولیه تولید کنیم ، و با میانگین گیری بین ایندکس زوج و فرد پشت سر هم صدای اولیه یک صدا با طول دو برابر تولید کنیم .

```
function speedTheVoice(inputFile, speed)

    if speed ~= 0.5 && speed ~= 2
        error('speed input is invalid .');
    end


    [x, Fs] = audioread(inputFile);

    if speed == 2
        x = x (1 : 2 : end);
    end

    if speed == 0.5
        y = zeros(2*length(x)-1, size(x, 2));

        y(1:2:end, :) = x;
        y(2:2:end-1, :) = (x(1:end-1, :) + x(2:end, :)) / 2;

    end


    audiowrite("newEditedVoice.wav" , x , Fs);

end
```

Part 4.4

```
function newX = p4_4(inputFile, speed)

    if speed < 0.5 || speed > 2
        error('speed input is invalid .');
    end




    [x, Fs] = audioread(inputFile);

    %correct speed to a float number that has only one digit after
    %floationg point
    speed = round( speed , 1 );

    newX = x( round ( linspace (1 , size(x , 1 ) , size(x , 1)/speed) ) ) ;


    audiowrite("newVoice.wav", newX , Fs);

    fprintf("Done");
end
```

- با استفاده از دستور linspace یک وکتور خطی میسازیم از یک تا سایز x . با گام هایی به طول size(x, 1)/speed