**بخش دوم**

clc, clearvars, **close** **all**;  
**load**('p2.mat')

**2 – 1)**

%% **2** **-** **1**  
**figure**;  
**plot**(t, x);  
**xlabel**("t"), **ylabel**("x"), **title**("x per t");



**2 – 2)**

%% 2 - 2  
figure;  
plot(t, y);  
xlabel("t"), ylabel("y"), title("y per t");



**2 – 3)** از آنجایی که می‌دانیم رابطه ی y بر حسب x خطی می باشد، و به صورت می باشد، بنابراین شیب خط برابر و عرض از مبدا آن برابر می باشد.

%% 2 - 3  
figure;  
plot(x, y, '.');  
xlabel("x"), ylabel("y"), title("y per x");

****

2 – 4) برای محاسبه‌ی α و β باید با استفاده از مشتق گیری ضمنی از تابع می‌توانیم α و β را حساب کنیم.

**A black and white math equation

AI-generated content may be incorrect.**

**function** [alpha, beta] = **p2\_4**(x,y)  
 alpha = sum((x - mean(x)) .\* (y - mean(y))) / sum((x - mean(x)).^2);  
 beta = mean(y) - alpha \* mean(x);  
**end**

%% 2 - 4  
[alpha, beta] = p2\_4(x, y)  
figure;  
plot(x, y, '.');  
xlabel("x"), ylabel("y"), title("y per x and fitted curve");  
hold on;  
t1 = -1:0.001:1;  
plot(t1, alpha .\* t1 + beta, "Color", "Red", "LineWidth", 2);  
legend("Data", "fitted line");

alpha =  
  
 1.9736  
  
  
beta =  
  
 -0.4983

****

تست تابع با مقادیر دیگر:

clc, clearvars, close all;

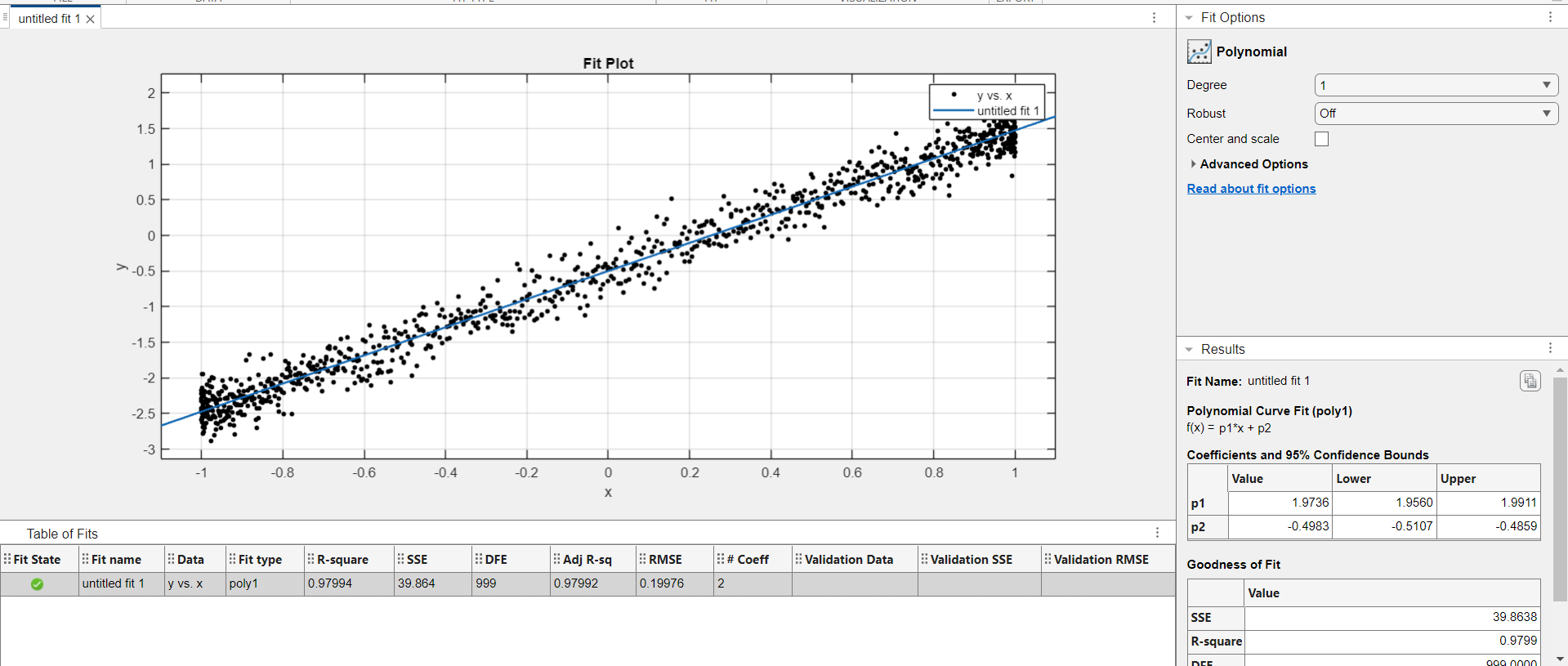
alphaTrue = 2.5;  
betaTrue = 1.0;  
t = 0:0.001:1;  
x = sin(2 \* pi \* t);  
y = alphaTrue \* x + betaTrue;  
noiseLevel = 0.1;  
yNoisy = y + noiseLevel \* randn(size(y));  
[alphaEstimated, betaEstimated] = p2\_4(x, yNoisy);  
fprintf('real alpha %.4f estimated alpha %.4f\n', alphaTrue, alphaEstimated)  
fprintf('real beta %.4f estimated beta %.4f\n', betaTrue, betaEstimated)  
figure;  
plot(x, yNoisy, '.');  
xlabel("x"), ylabel("y"), title("y per x and fitted curve");  
hold on;  
plot(x, alphaEstimated \* x + betaEstimated, 'r', 'LineWidth', 2);  
legend("Data", "fitted line");  
grid on;

real alpha 2.5000 estimated alpha 2.5048  
real beta 1.0000 estimated beta 0.9968



*این تابع را با مقادیر دیگر تست کردم. مقدار و را جایگذاری کردم و همچنین تابع را قرار دادم. مقادیر به دست آمده، به ترتیب برابر 2.5048 و 0.9968 بودند که تقریب خوبی می باشد.*

**2 – 5)** بله دقیقا برابر هستند:



**بخش سوم**

3-1 )

  
%% 3 - 1  
ts = 1e-9;  
T = 1e-5;  
tau = 1e-6;  
t = 0:ts:T;  
x = zeros(size(t));  
x(1:(tau/ts)) = 1;  
figure;  
plot(t, x, "LineWidth", 2);  
xlim([0, T]);  
ylim([-0.2, 1.2]);  
xlabel("t"), ylabel("x(t)"), title("Input Signal");

2 – 3 )

R = 450;  
c = 3e8;  
td = 2 \* R / c;  
outputSignal = zeros(size(t));  
alpha = 0.5;  
outputSignal(td/ts:(td+tau) / ts) = alpha;  
figure;  
plot(t, outputSignal, "LineWidth", 2);  
xlim([0, T]);  
ylim([-0.2, 1.2]);  
xlabel("t"), ylabel("x(t)"), title("Output Signal");



*3 – 3) نمودار مقدار* correlation *به دست آمده بر پایه زمان:*

*حال نقطه ی ماکسیمم این نمودار را پیدا میکنیم. این نقطه، می باشد. با توجه به رابطه ی زیر، فاصله ی جسم یا همان* R *بدست می آید:*

sigma = calc\_corr(t, tau, ts, outputSignal);  
plot(t, sigma, "LineWidth", 3);  
[maxSum, maxCorr] = max(ro);  
td = t(maxCorr);  
estimatedR = td \* c / 2

**function** **sigma** = **calc\_corr**(t, tau, ts, inputSignal)  
 sigma = zeros(size(t));  
 **for** i=1:length(t)-(tau/ts)  
 temp = zeros(size(t));  
 temp (i:i+(tau/ts)-1) = 1;  
 sigma(i) = sum(temp .\* inputSignal);  
 **end**  
**end**

که مقدار تقریبی 449.8500 به دست آمد و به 450 خیلی نزدیک است.

estimatedR =  
  
 449.8500



تابع calc\_corr در اینجا به این صورت کار می‌کند که هرسری یک سیگنال نمونه با اندازه‌‌‌هایی که داشتیم تولید می‌شود و با تابع ورودی جمع می‌شود. در نهایت هم مجموع تمام این‌ها برمی‌گردد. حال در برنامه‌ی اصلی td را که از ماکزیمم تابع خروجی به دست آوردیم، برمیداریم و R را محاسبه می‌کنیم.

3 – 4)   
متأسفانه سیگنال‌های ورودی اغلب نویزی هستند. برای شبیه‌سازی نویز، اعداد تصادفی بر اساس توزیع نرمال به سیگنال اضافه می‌کنیم. سپس نویز را افزایش می‌دهیم تا ببینیم تخمین فاصله چقدر از مقدار واقعی فاصله می‌گیرد. (فاصله واقعی ۴۵۰ متر است.)

برای کاهش اثر تصادفی بودن نویز، هر سطح نویز را ۱۰۰ بار تکرار می‌کنیم و میانگین خطاها را محاسبه می‌کنیم. این روند را ادامه می‌دهیم تا زمانی که خطا به 10 متر برسد.

noiseLevel = 0;  
numIterations = 100;  
distance = 450;  
noiseIncrement = 0.1;  
errorValues = zeros(1, numIterations);  
estimatedDistances = zeros(1, numIterations);  
lastError = 0;  
currentIndex = 1;  
  
**while**(lastError < 10)  
 totalError = 0;  
 totalDistance = 0;  
   
 **for** iteration = 1 : numIterations  
 correlationOutput = calc\_corr(t, tau, ts, outputSignal + noiseLevel \* randn(size(outputSignal)));  
 [~, maxCorrIndex] = max(correlationOutput);  
 estimatedDistance = t(maxCorrIndex) \* c / 2;  
 totalError = totalError + abs(distance - estimatedDistance);  
 totalDistance = totalDistance + estimatedDistance;  
 **end**  
   
 errorValues(currentIndex) = totalError / numIterations;  
 estimatedDistances(currentIndex) = totalDistance / numIterations;  
 lastError = errorValues(currentIndex);  
 currentIndex = currentIndex + 1;  
 noiseLevel = noiseLevel + noiseIncrement;  
**end**  
  
lastValidIndex = currentIndex - 1;  
noiseRange = (0 : noiseIncrement : noiseIncrement \* (lastValidIndex - 1));  
  
subplot(1, 2, 1);  
plot(noiseRange, errorValues(1:lastValidIndex), 'LineWidth', 3);  
title('Error');  
xlabel('noise');  
ylabel('error');  
  
subplot(1, 2, 2);  
plot(noiseRange, estimatedDistances(1:lastValidIndex), 'LineWidth', 3);  
title('Estimated Distance');  
xlabel('noise');  
ylabel('R');

