Signals and Systems

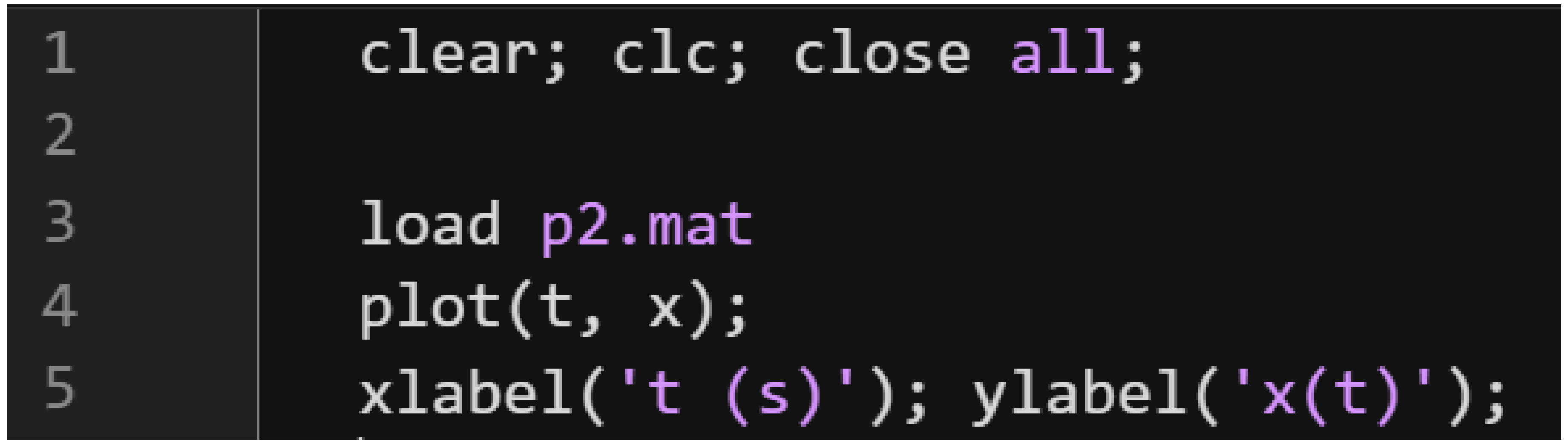
Project Report

Parsa KafshduziBukani 810102501

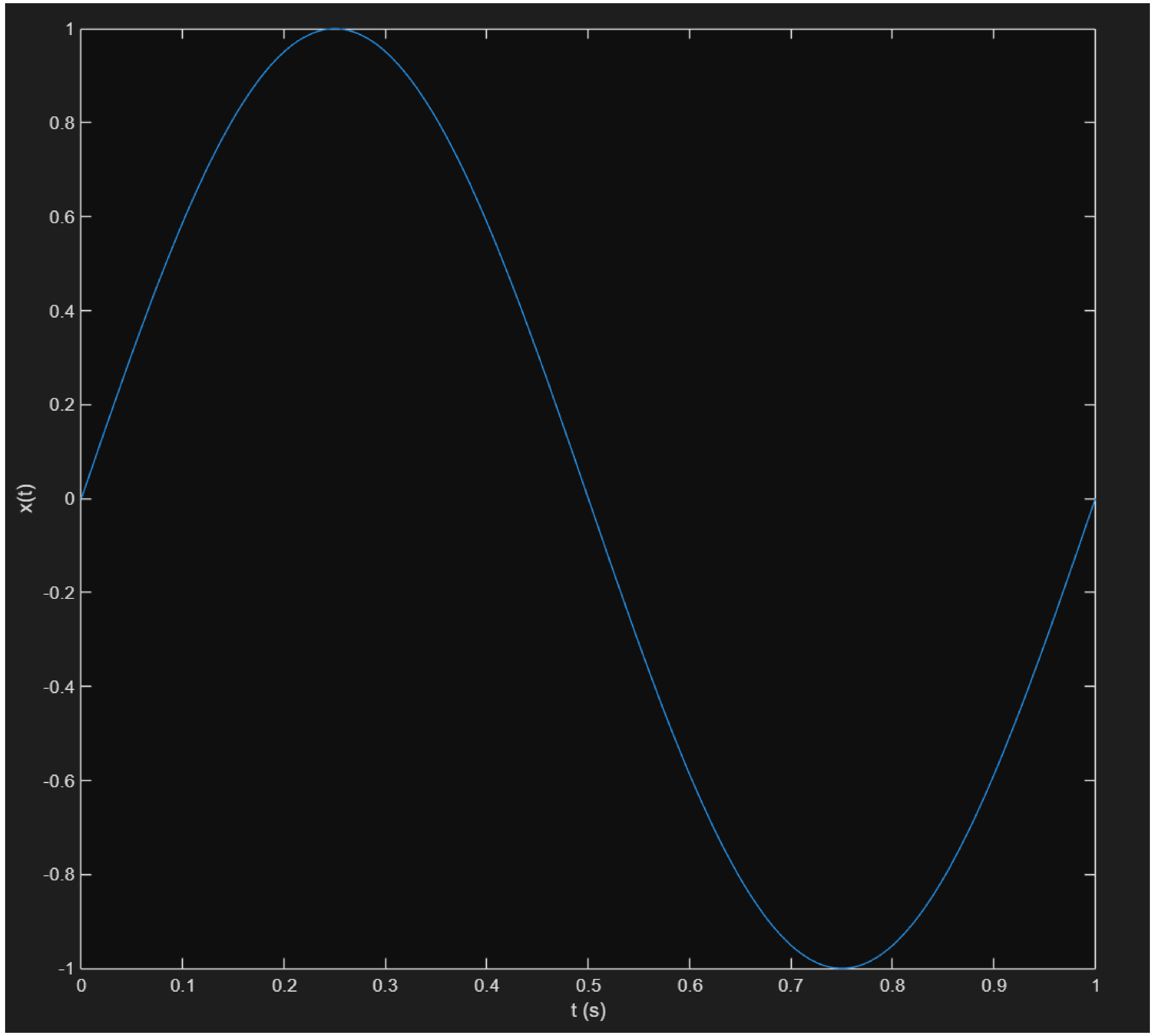
CA1

Section 2

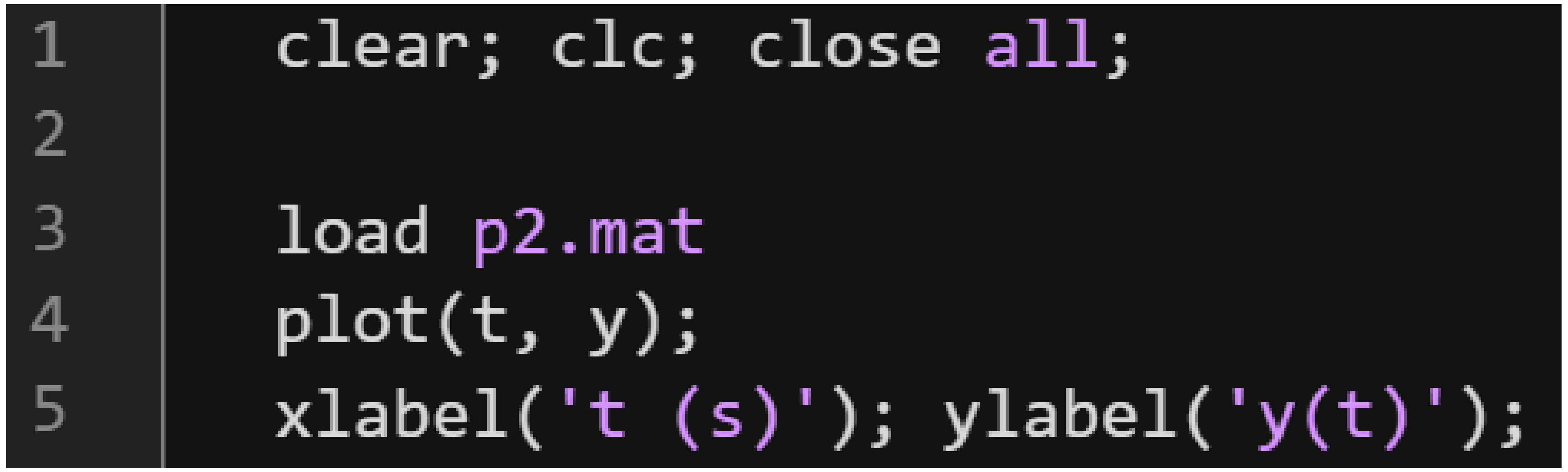
part 2.1:



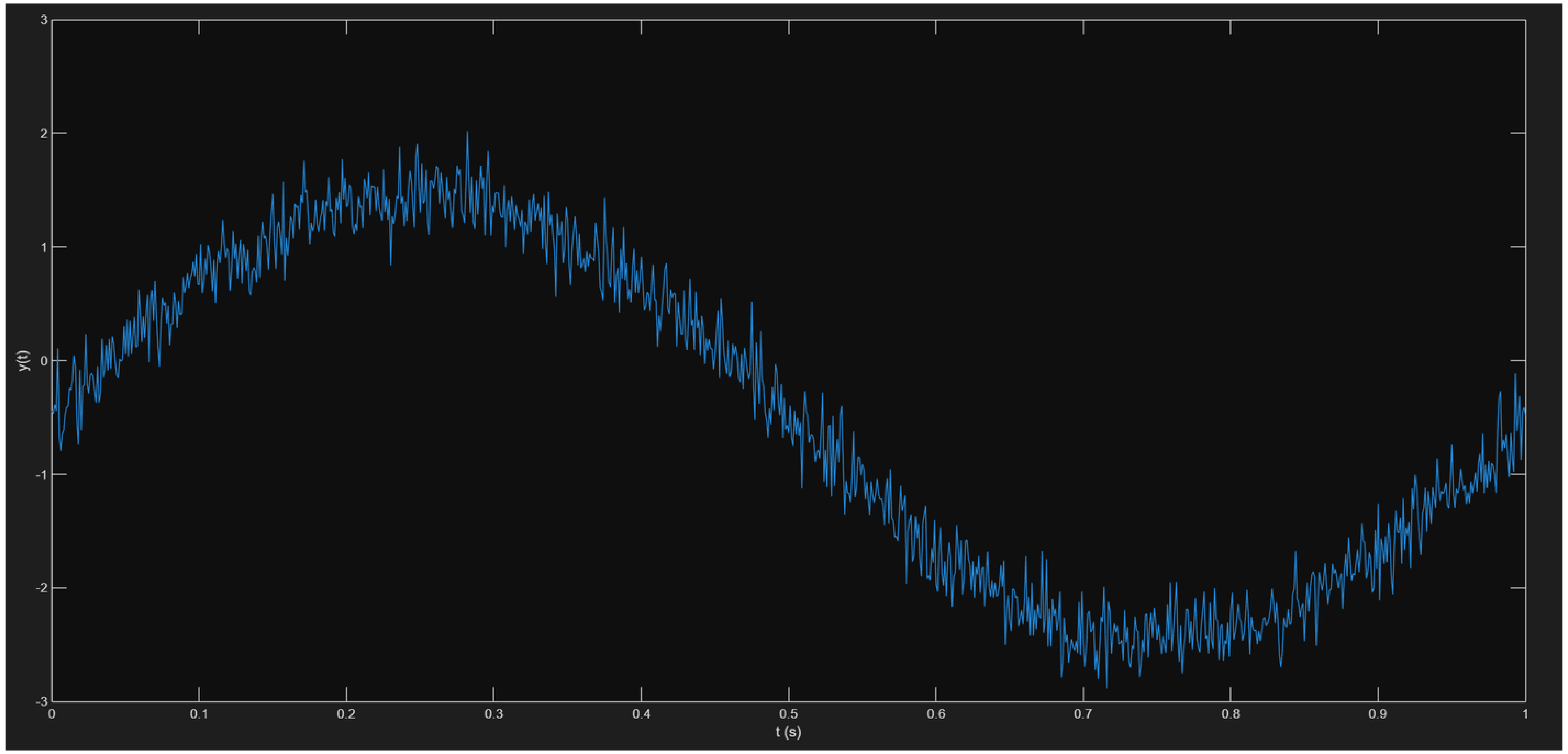
```
clear; clc; close all;

load p2.mat
plot(t, x);
xlabel('t (s)'); ylabel('x(t)');
```



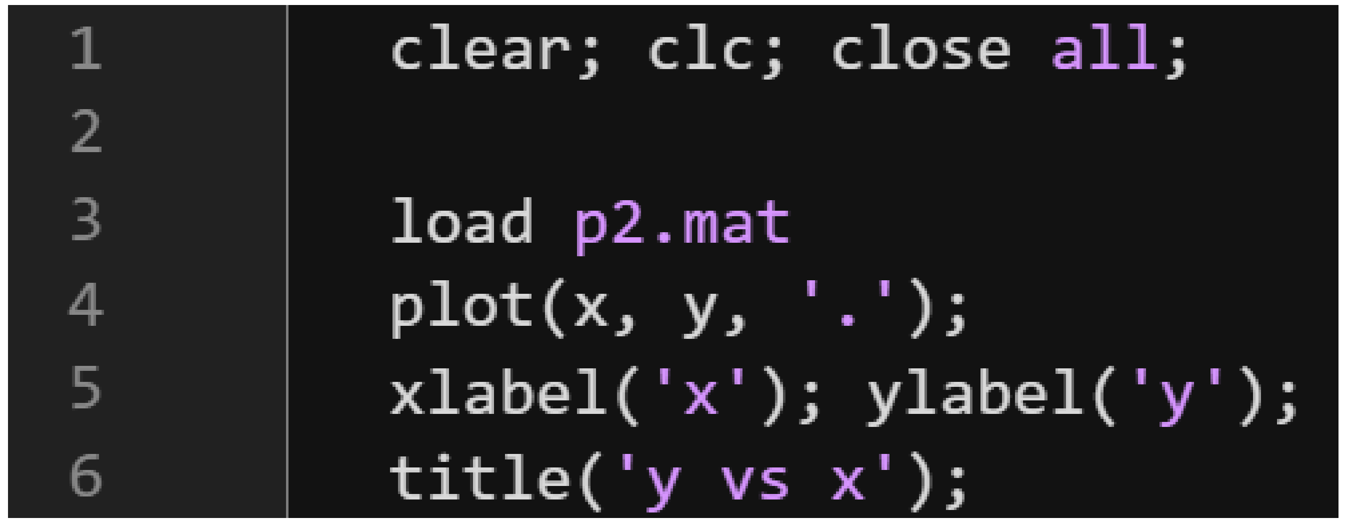
part 2.2:



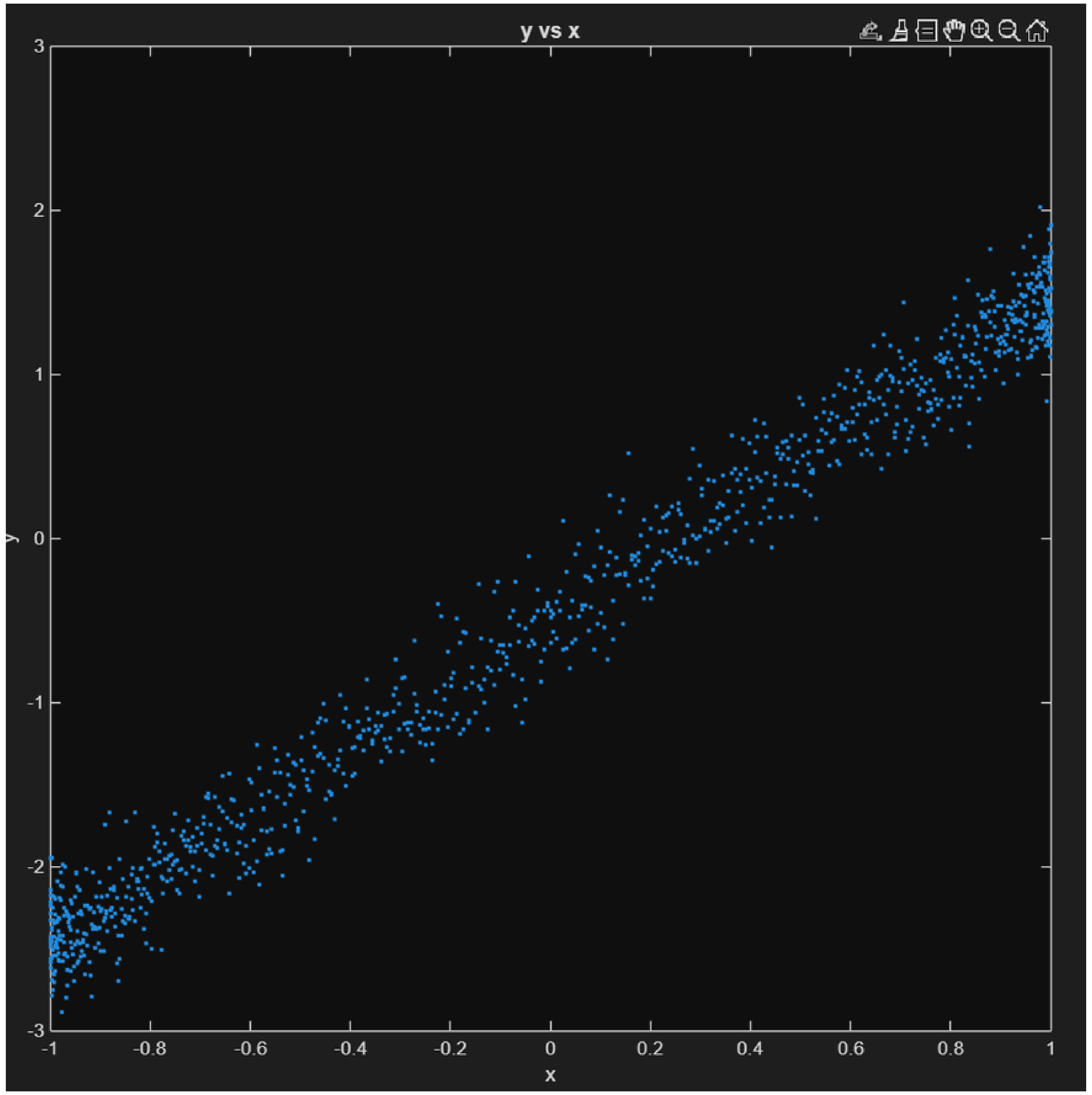
```
clear; clc; close all;

load p2.mat
plot(t, y);
xlabel('t (s)'); ylabel('y(t)');
```



part 2.3:



```
clear; clc; close all;

load p2.mat
plot(x, y, '.');
xlabel('x'); ylabel('y');
title('y vs x');
```



The scatter plot of y versus x shows an approximately linear trend, consistent with the idle model y=αx+β. The slope of the best-fit line estimates α (system gain) and the intercept at x=0 estimates β. Spread around the line is due to noise.

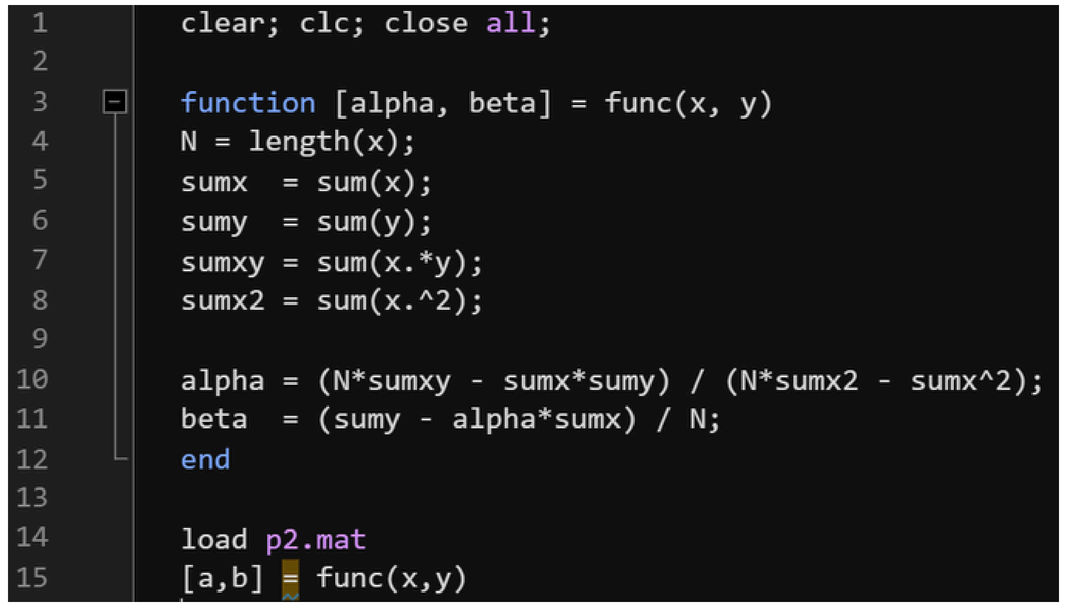
part 2.4:

To estimate the parameters α and β in the linear model y=αx+β, the least squares method was applied by minimizing ------->

$$f(\alpha,\beta) = \sum (y(t) - \alpha x(t) - \beta)^2$$

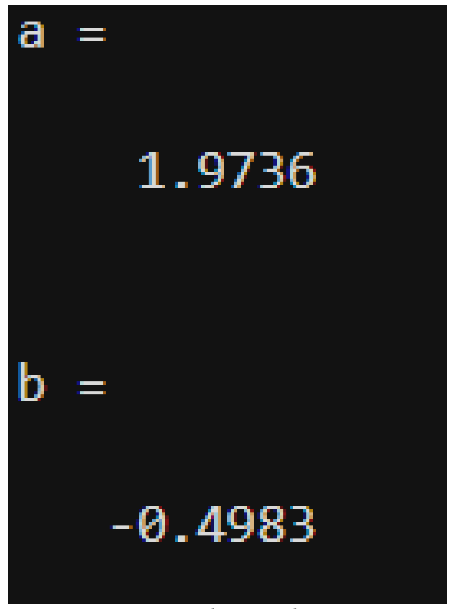
We minimize the total squared error By taking the partial derivatives of the cost function f(α,β) with respect to α and β and setting them to zero, we get two linear equations ------>

$$\alpha = \frac{N\sum xy - (\sum x)(\sum y)}{N\sum x^2 - (\sum x)^2}, \quad \beta = \frac{\sum y - \alpha \sum x}{N}$$



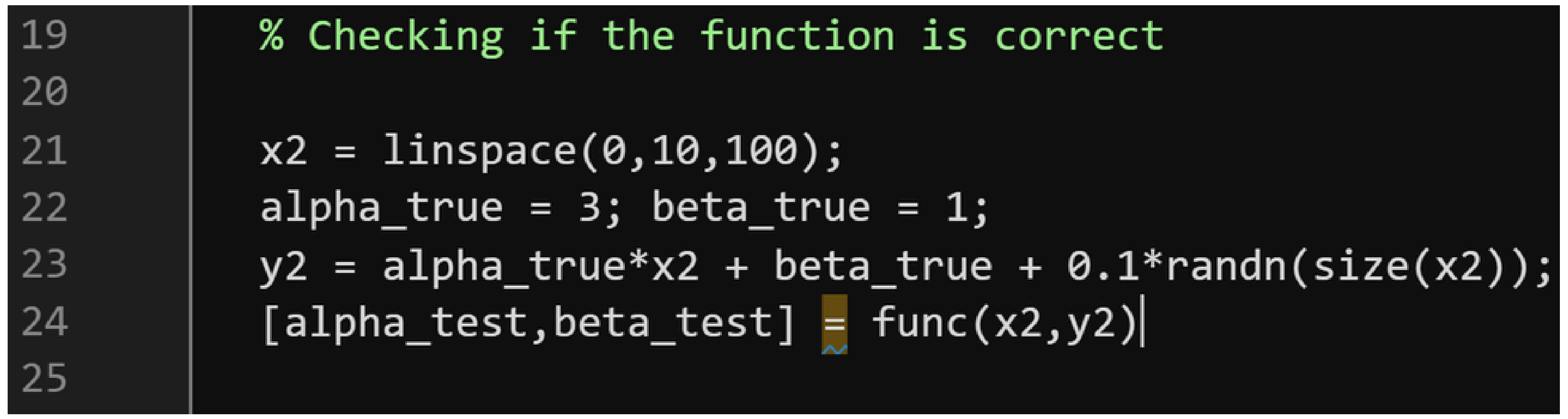
```
clear; clc; close all;

function [alpha, beta] = func(x, y)
N = length(x);
sumx  = sum(x);
sumy  = sum(y);
sumxy = sum(x.*y);
sumx2 = sum(x.^2);

alpha = (N*sumxy - sumx*sumy) / (N*sumx2 - sumx^2);
beta  = (sumy - alpha*sumx) / N;
end

load p2.mat
[a,b] = func(x,y)
```

code implementation



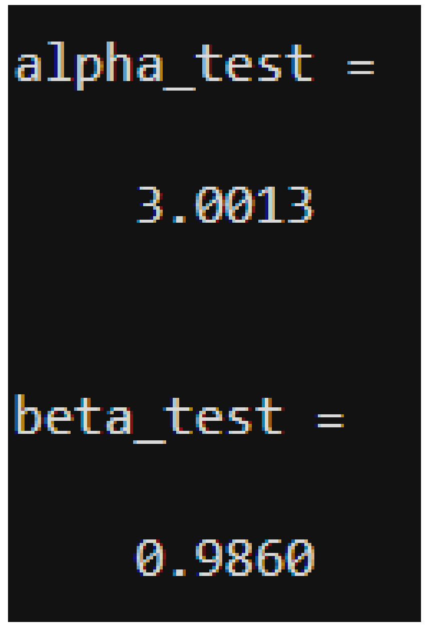
```
a =

    1.9736


b =

   -0.4983
```

output



```
% Checking if the function is correct

x2 = linspace(0,10,100);
alpha_true = 3; beta_true = 1;
y2 = alpha_true*x2 + beta_true + 0.1*randn(size(x2));
[alpha_test,beta_test] = func(x2,y2)

```

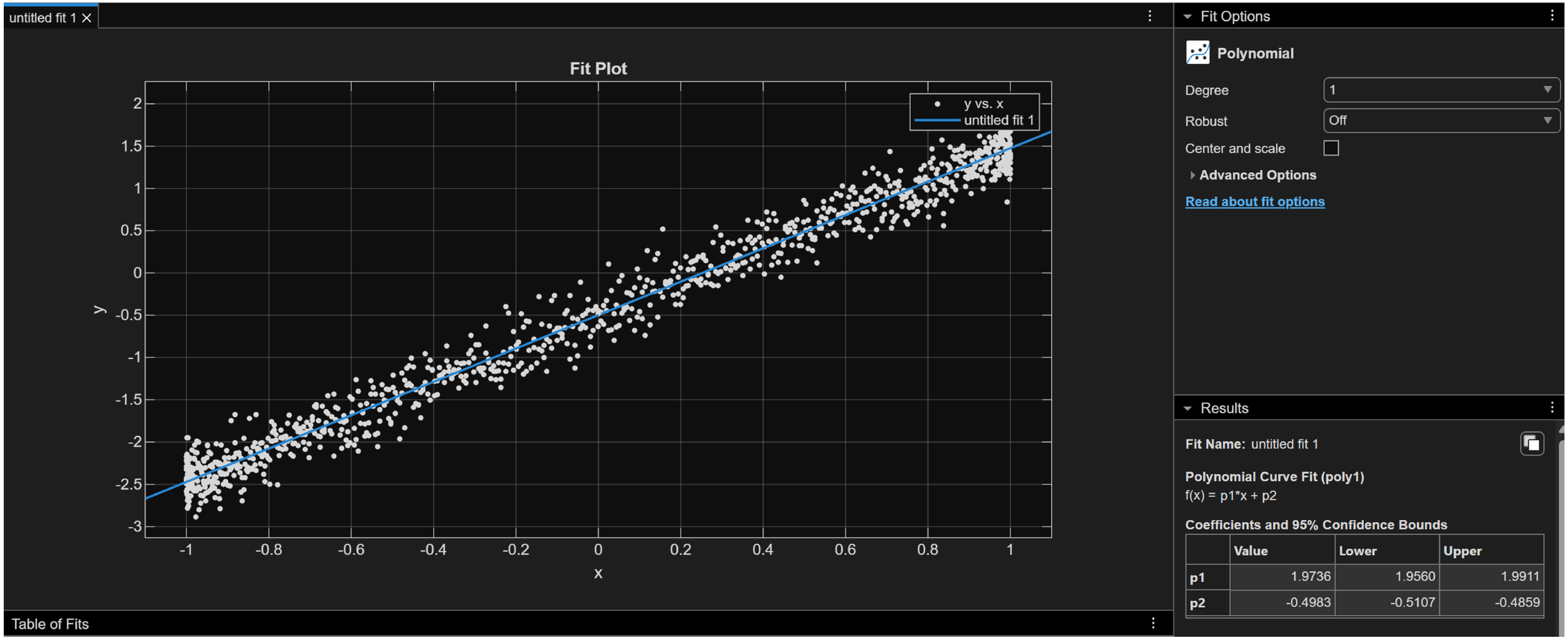
function check



```
alpha_test =

    3.0013


beta_test =

    0.9860
```

output

part 2.5:



Polynomial

Degree: 1

Robust: Off

Center and scale: ☐

Fit Name: untitled fit 1

Polynomial Curve Fit (poly1)

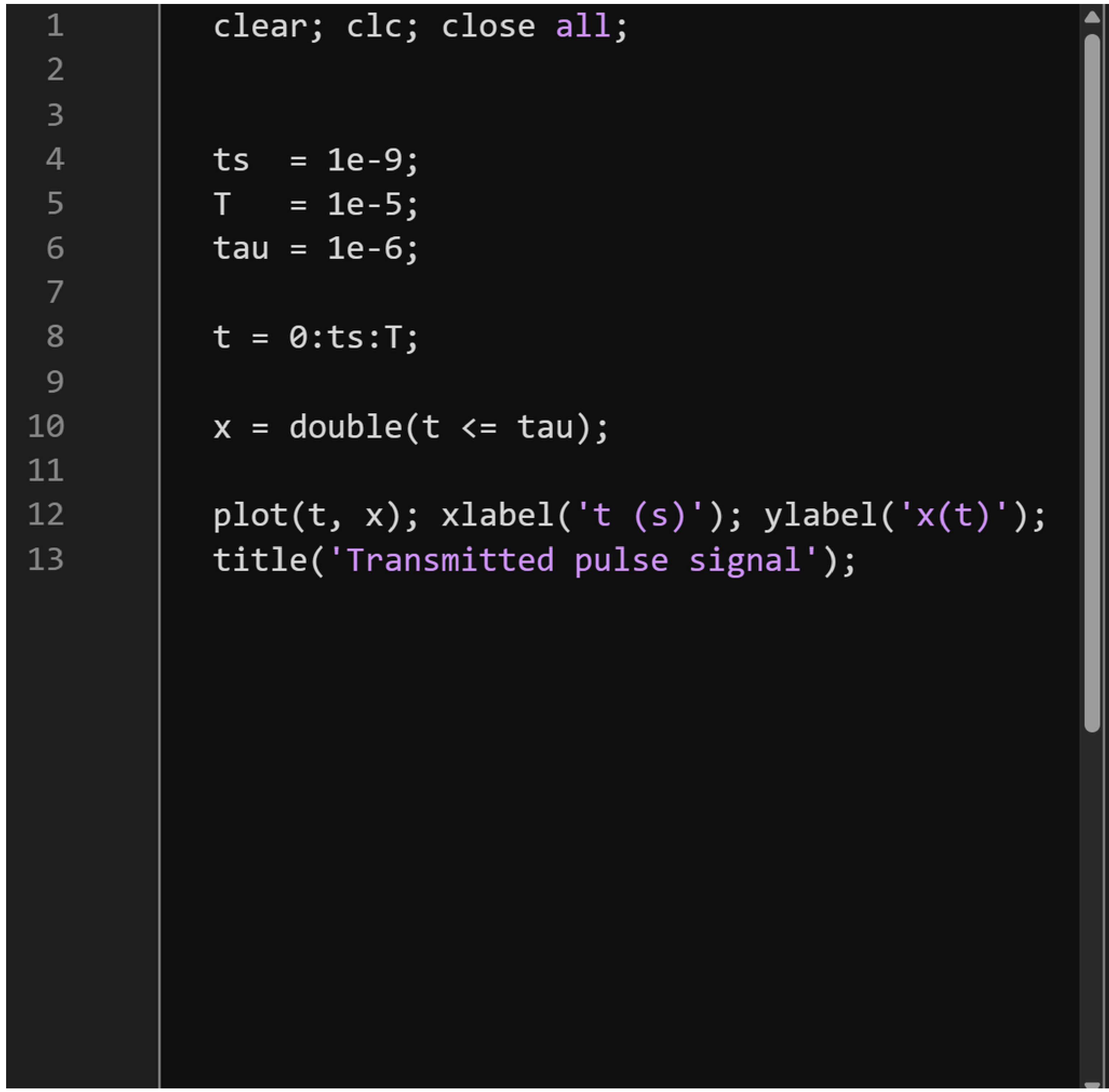
f(x) = p1*x + p2

Coefficients and 95% Confidence Bounds

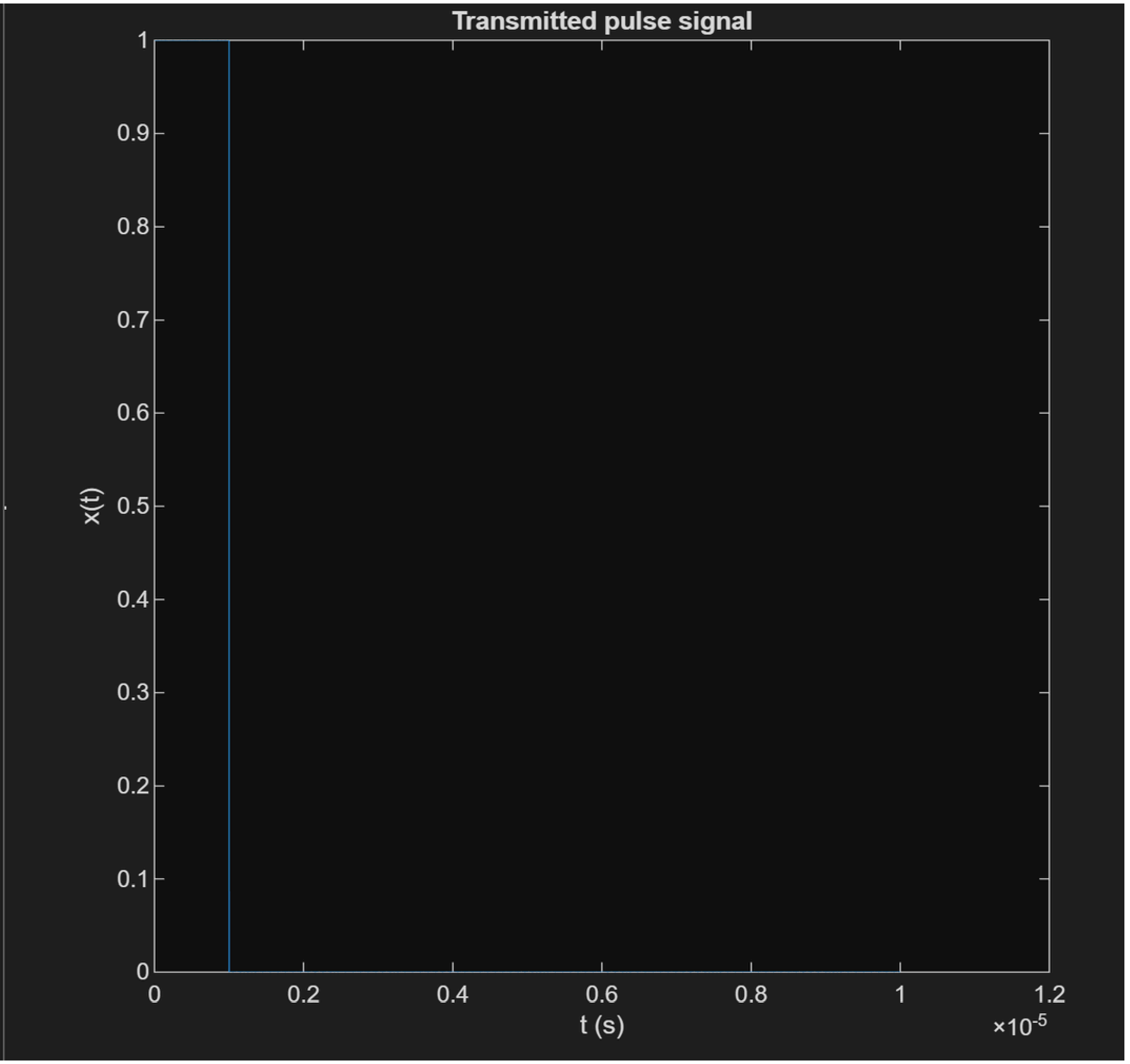
	Value	Lower	Upper
p1	1.9736	1.9560	1.9911
p2	-0.4983	-0.5107	-0.4859

Section 3

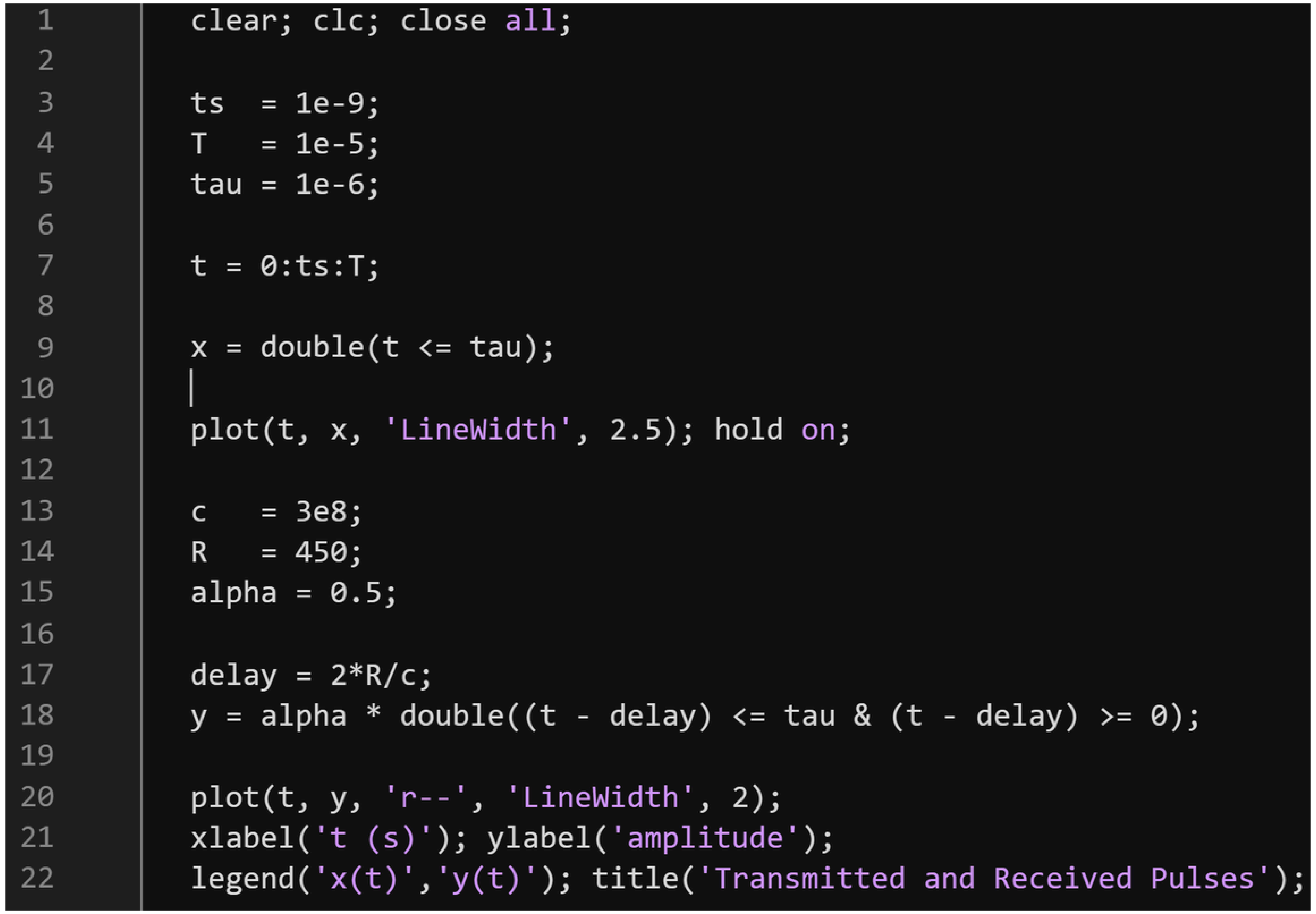
part 3.1:



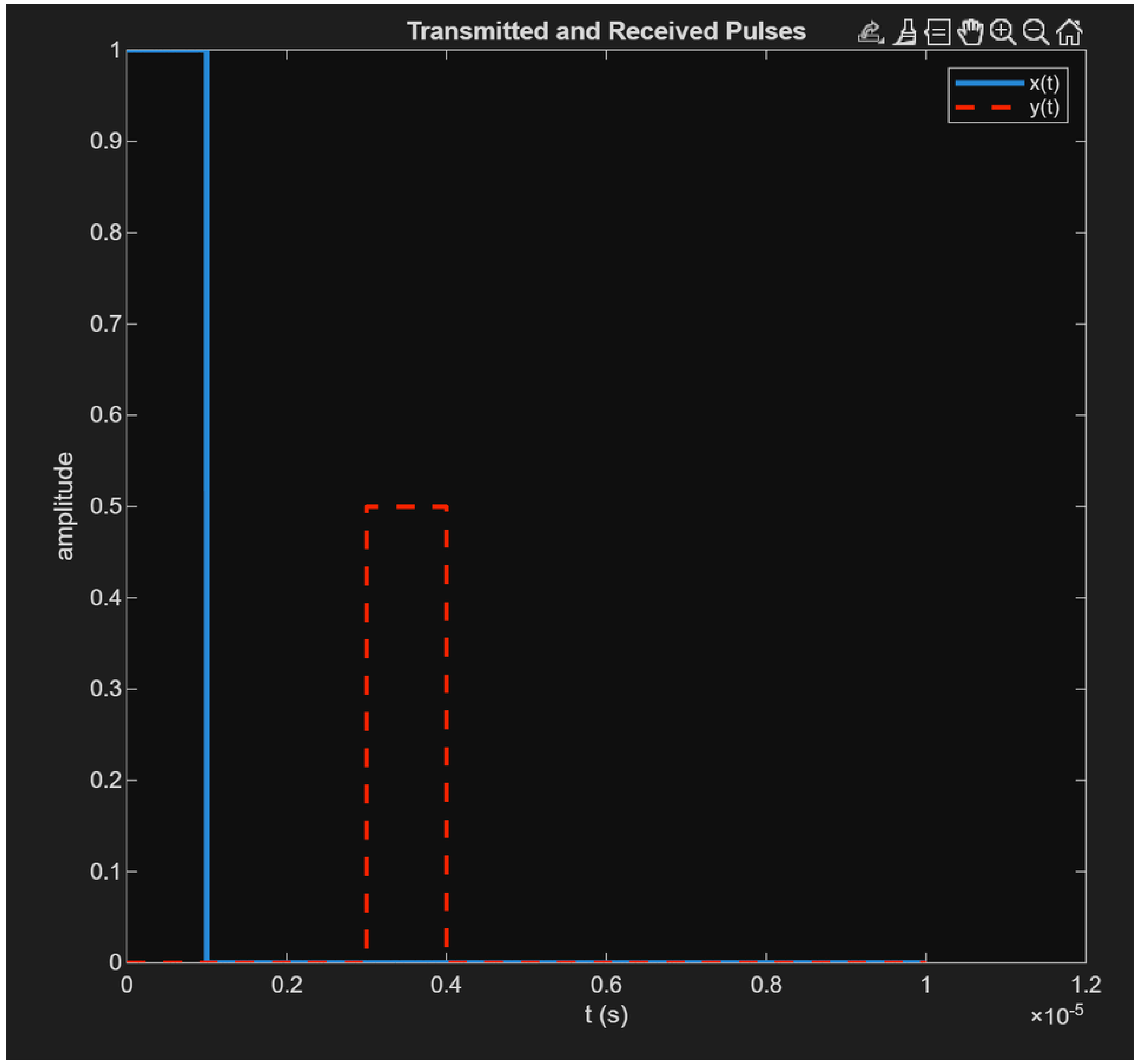
```
clear; clc; close all;


ts  = 1e-9;
T   = 1e-5;
tau = 1e-6;

t = 0:ts:T;

x = double(t <= tau);

plot(t, x); xlabel('t (s)'); ylabel('x(t)');
title('Transmitted pulse signal');
```



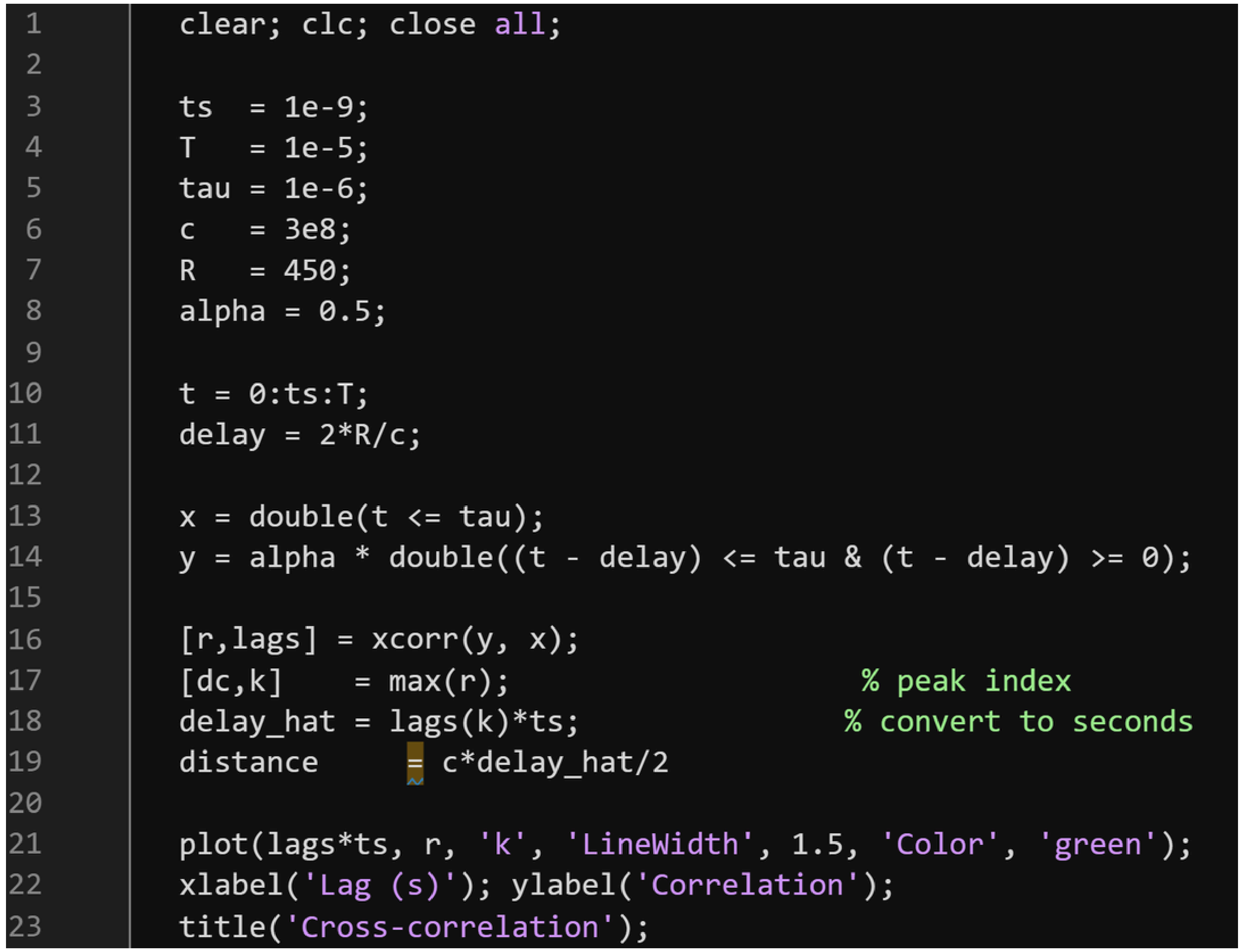
part 3.2:



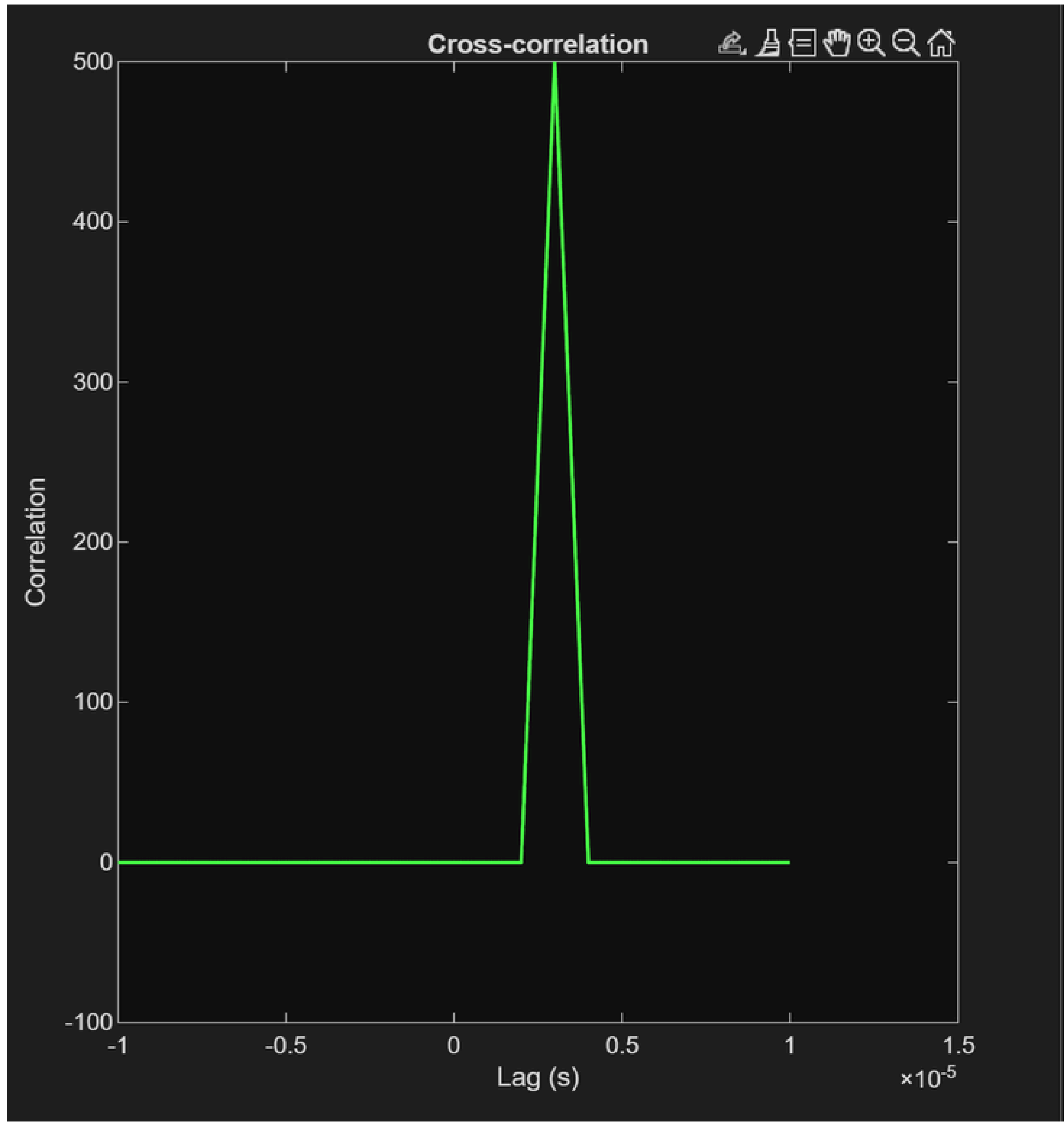
```
clear; clc; close all;

ts  = 1e-9;
T   = 1e-5;
tau = 1e-6;

t = 0:ts:T;

x = double(t <= tau);

plot(t, x, 'LineWidth', 2.5); hold on;

c   = 3e8;
R   = 450;
alpha = 0.5;

delay = 2*R/c;
y = alpha * double((t - delay) <= tau & (t - delay) >= 0);

plot(t, y, 'r--', 'LineWidth', 2);
xlabel('t (s)'); ylabel('amplitude');
legend('x(t)','y(t)'); title('Transmitted and Received Pulses');
```

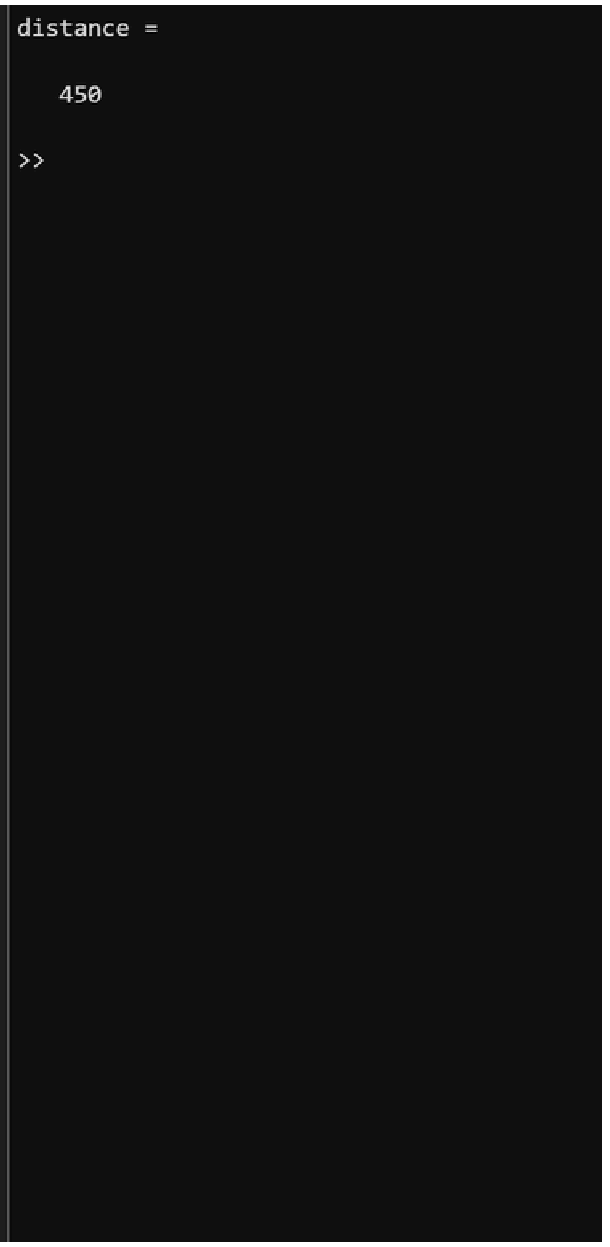


part 3.3:



```
clear; clc; close all;

ts  = 1e-9;
T   = 1e-5;
tau = 1e-6;
c   = 3e8;
R   = 450;
alpha = 0.5;

t = 0:ts:T;
delay = 2*R/c;

x = double(t <= tau);
y = alpha * double((t - delay) <= tau & (t - delay) >= 0);

[r,lags] = xcorr(y, x);
[dc,k]    = max(r);                        % peak index
delay_hat = lags(k)*ts;                  % convert to seconds
distance     = c*delay_hat/2

plot(lags*ts, r, 'k', 'LineWidth', 1.5, 'Color', 'green');
xlabel('Lag (s)'); ylabel('Correlation');
title('Cross-correlation');
```

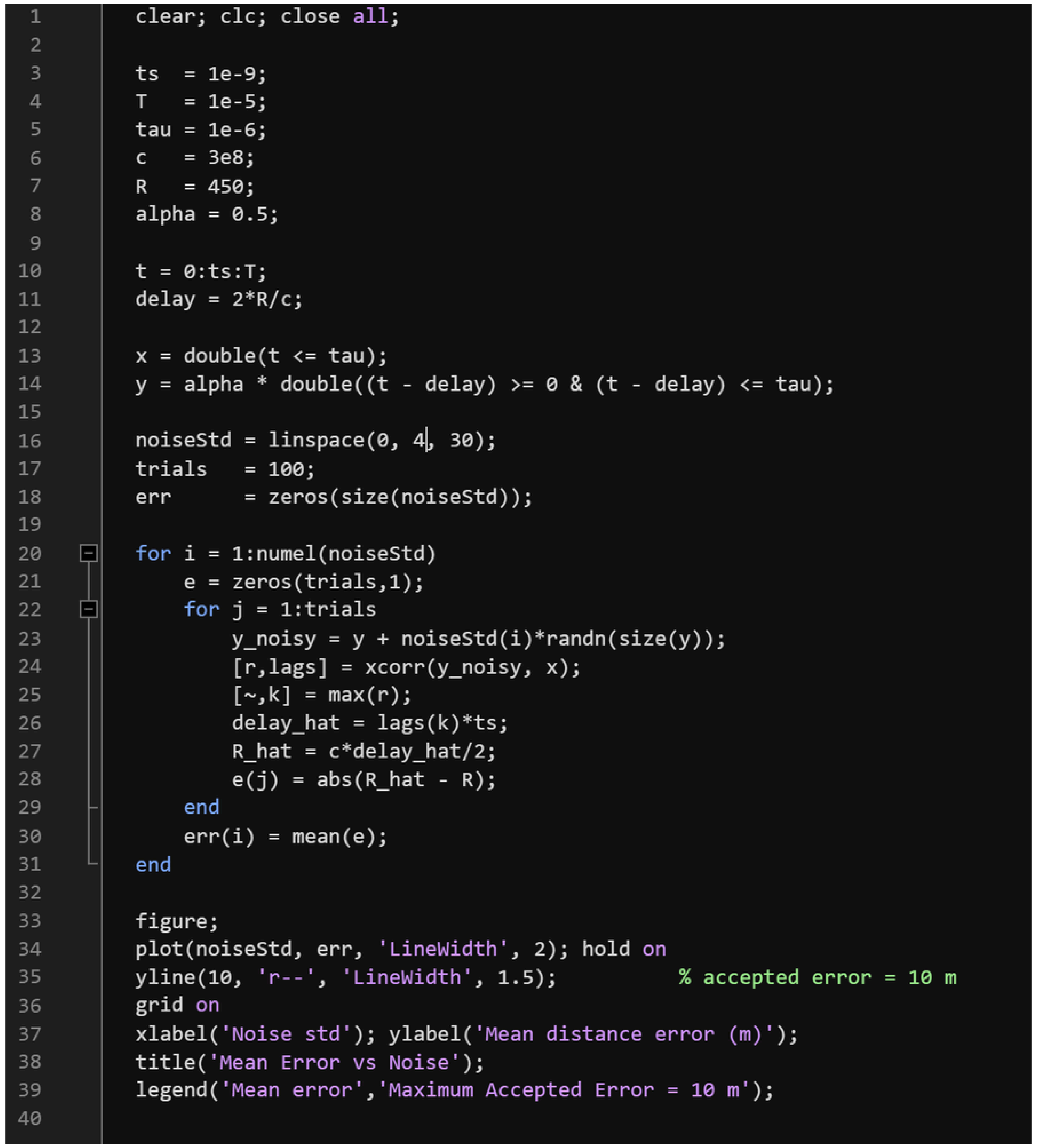




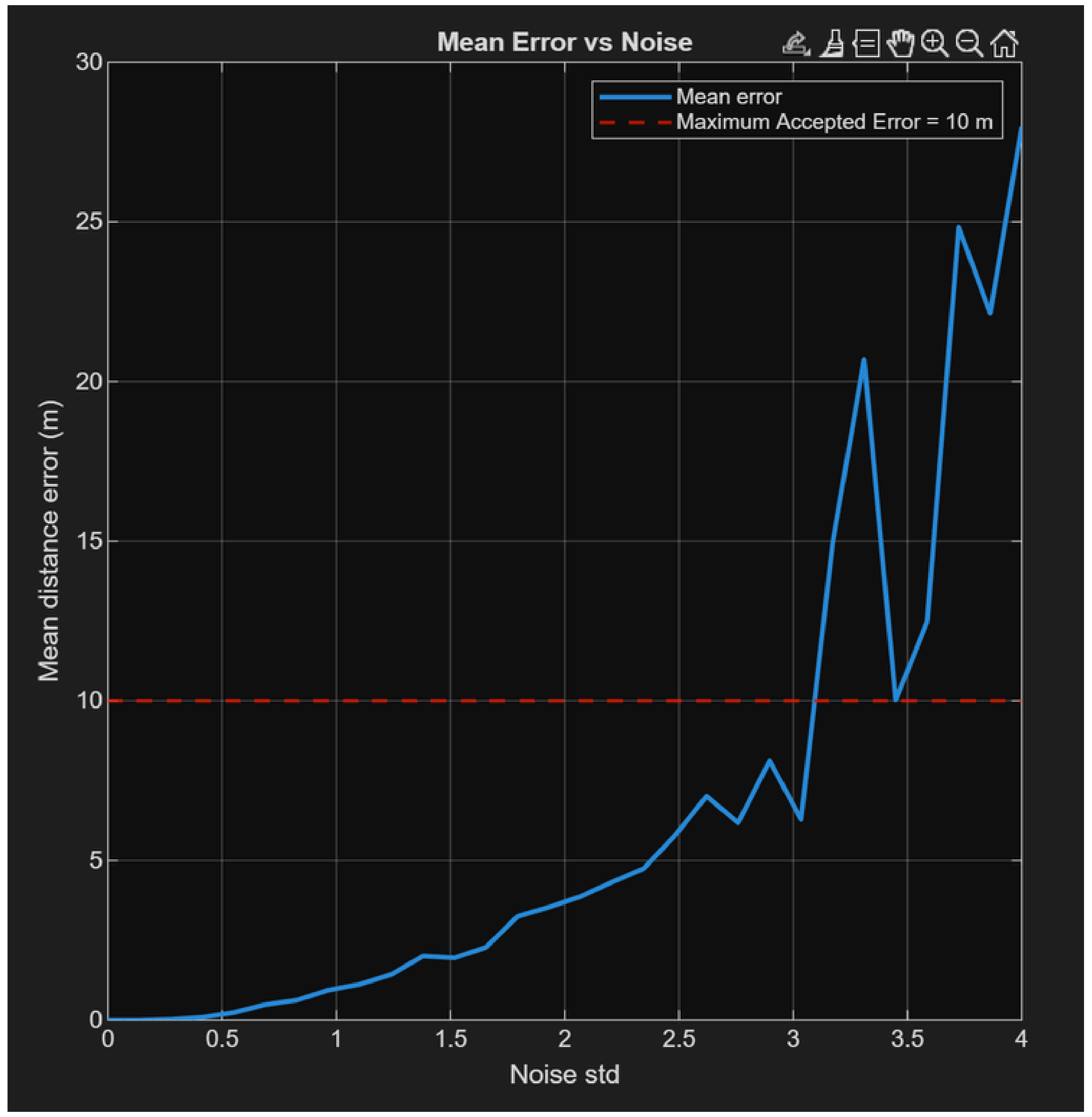
```
distance =

   450

>>
```

We calculate the cross-correlation between the transmitted and received pulses.
The correlation peak shows where the two signals best align — this gives the time delay of the echo.
From that delay, the target distance is estimated as **R = c × delay /2**

part 3.4:



```matlab
clear; clc; close all;

ts  = 1e-9;
T   = 1e-5;
tau = 1e-6;
c   = 3e8;
R   = 450;
alpha = 0.5;

t = 0:ts:T;
delay = 2*R/c;

x = double(t <= tau);
y = alpha * double((t - delay) >= 0 & (t - delay) <= tau);

noiseStd = linspace(0, 4, 30);
trials   = 100;
err      = zeros(size(noiseStd));

for i = 1:numel(noiseStd)
    e = zeros(trials,1);
    for j = 1:trials
        y_noisy = y + noiseStd(i)*randn(size(y));
        [r,lags] = xcorr(y_noisy, x);
        [~,k] = max(r);
        delay_hat = lags(k)*ts;
        R_hat = c*delay_hat/2;
        e(j) = abs(R_hat - R);
    end
    err(i) = mean(e);
end

figure;
plot(noiseStd, err, 'LineWidth', 2); hold on
yline(10, 'r--', 'LineWidth', 1.5);          % accepted error = 10 m
grid on
xlabel('Noise std'); ylabel('Mean distance error (m)');
title('Mean Error vs Noise');
legend('Mean error','Maximum Accepted Error = 10 m');
```



We gradually increase the noise level and, for each level, repeat the distance estimation 100 times with new random noise. The average error of these trials is taken as the error for that noise level. Finally, the mean error is plotted versus noise strength — as noise increases, estimation accuracy decreases.

From roughly noise std ≈ 3 onward, the error crosses the 10 m limit, meaning accurate distance detection is no longer reliable beyond that noise level.