گزارش‌کار تمرین کامپیوتری اول درس سیگنال و سیستم

محمد مهدی صمدی 810101465

بخش اول

تمرین 1_1)

قطعه کد استفاده شده در این بخش را در تصویر 1 مشاهده می‌کنیم.

```
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);

title('Sin and Cos');
legend('sin', 'cos');
xlabel('time');
ylabel('amplitude');
grid on;
```

تصویر 1 - کد بخش 1_1

توضیح خط به خط کد تصویر 1:

1: تعریف بازه زمانی به صورت آرایه‌ای با شروع از 0 و اتمام در 1 با گام‌هایی به طول 0.01

2: تعریف متغیر با فرمول $sin(2\pi t)$

3: تعریف متغیر با فرمول $cos(2\pi t$

5: باز کردن صفحه نمایش پلات

6: پلات کردن متغیر سینوسی تعریف شده بر اساس زمان

7: نگه داشتن پلات قبلی به همراه پلاتی که در آینده رسم خواهد شد

8: پلات کردن متغیر کسینوسی تعریف شده بر اساس زمان

10و 11: تعریف دو آرایه دو بعدی با ابعاد 2X1 برای مختصات متن‌های نشان داده شده روی نمودار

12: متن‌های نشان داده شده روی نمودار

13: نشان دادن متن‌های آماده شده

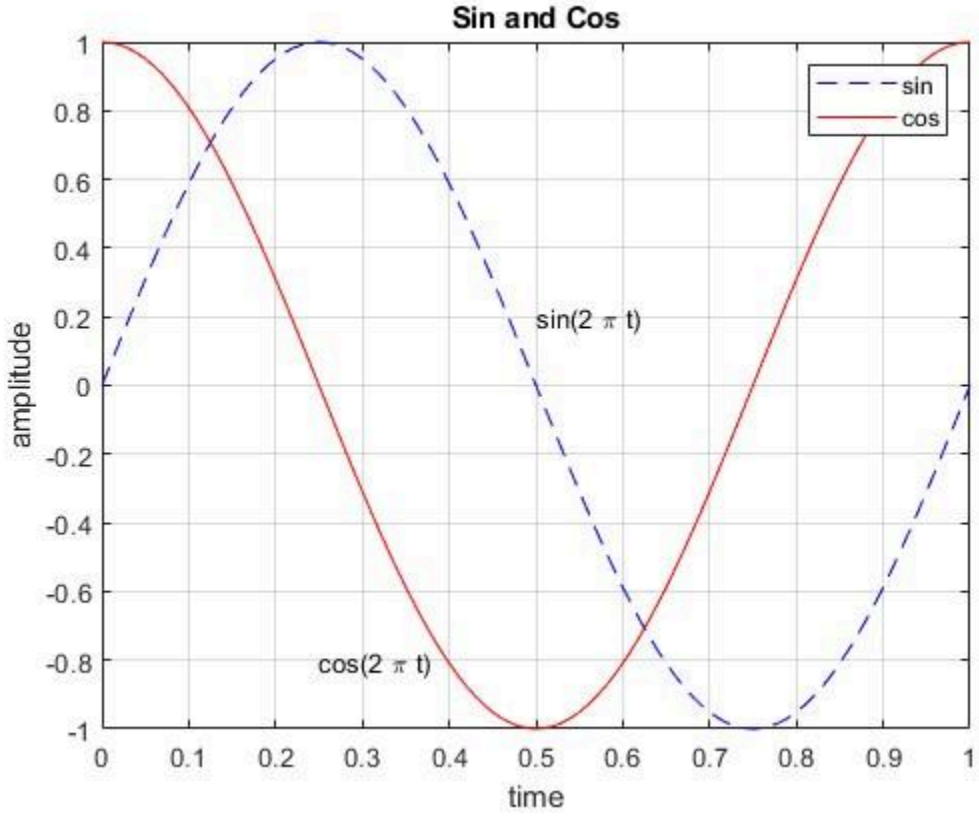
15: ست کردن title برای کل نمودار

16: ست کردن title برای هر کدام از دو نمودار که آن‌ها را تشخیص دهیم

17و 18: ست کردن title محور افقی و عمودی

19: نمایش نمودار به صورت شطرنجی

نتیجه اجرای کد بالا را در تصویر 2 مشاهده می‌کنیم



تصویر 2 - نمودار نهایی بخش 1_1

اگر از دستور hold on استفاده نکنیم، نمودار اول همزمان با نمودار دوم نمایش داده نمی‌شود.

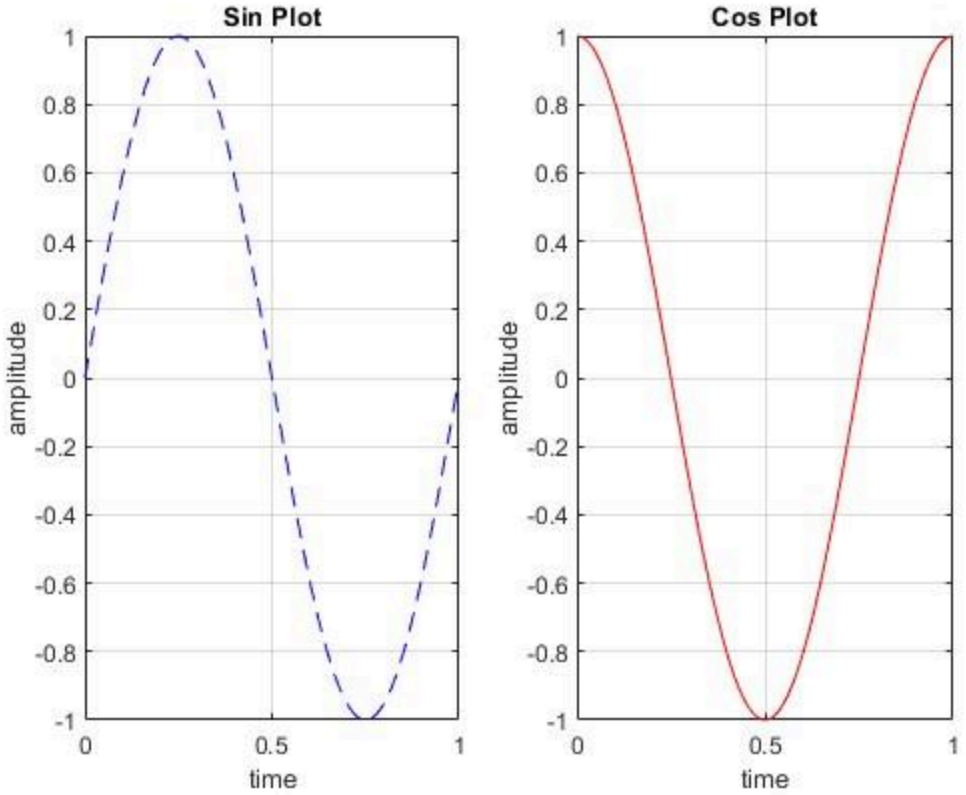
تمرین 1_2)

در این بخش از ما خواسته شده تا دو نمودار را کنار هم رسم کنیم. از دستور subplot کمک می‌گیریم. قطعه کد استفاده شده در این بخش را در تصویر 3 مشاهده می‌کنیم.

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

figure;
subplot(1, 2, 1);
plot(t, z1, '--b');
title('Sin Plot')
xlabel('time');
ylabel('amplitude');
grid on;

subplot(1, 2, 2);
plot(t, z2, 'r');
title('Cos Plot');
xlabel('time');
ylabel('amplitude');
grid on;
```

تصویر 3 - کد بخش 1_2

دستور subplot سه آرگومان دریافت کرده که اولین آن‌ها تعداد سطرها، دومین‌شان تعداد ستون‌ها و سومین آرگومان شماره نموداری که قرار است درباره آن اطلاعات بدهیم را نشان می‌دهند. خروجی کد بالا را در تصویر چهارم مشاهده می‌کنیم.



تصویر 4 - نمودار نهایی بخش 1_2

بخش دوم

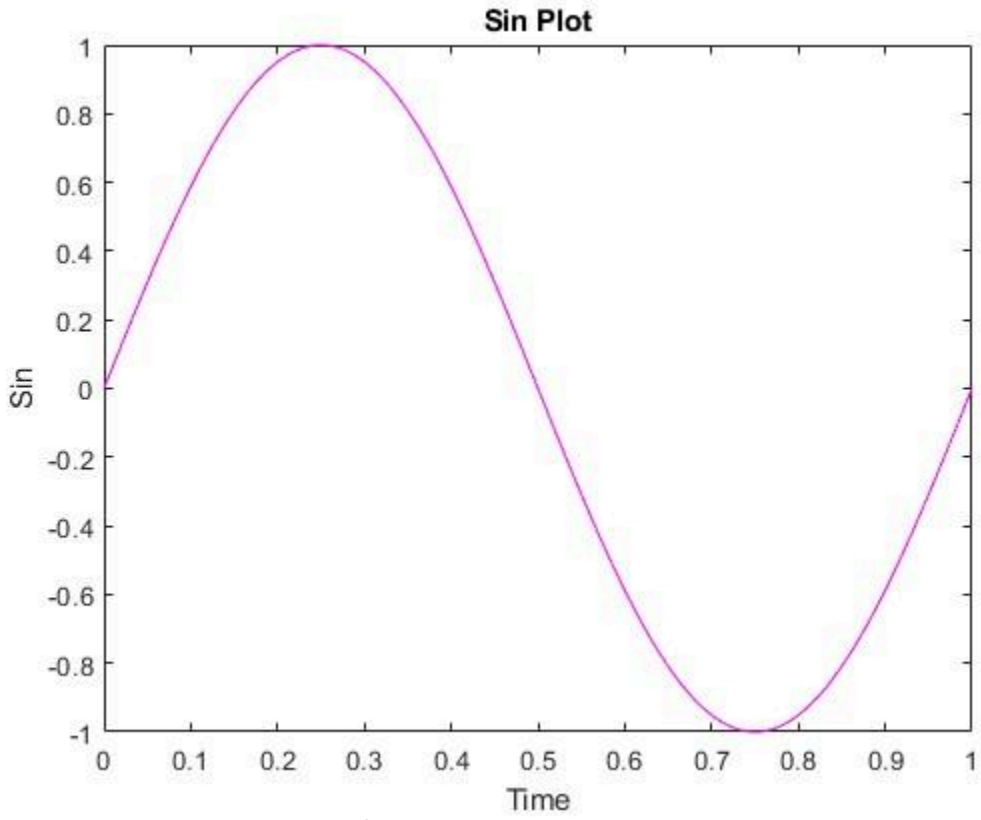
تمرین 2_1)

کد استفاده شده در این بخش در تصویر 5 آورده شده است.

```
load p2.mat;
plot(t, x, 'm');
title('Sin Plot');
xlabel('Time');
ylabel('Sin');
grid off;
```

تصویر 5 - کد بخش 1_2

کد تصویر 5 در ابتدا فایل داده شده را لود کرده و سپس متغیر x را بر حسب متغیر زمانی t پلات می‌کند. نمودار حاصل را در تصویر 6 مشاهده می‌کنیم.



تصویر 6 - نمودار نهایی بخش 2_1

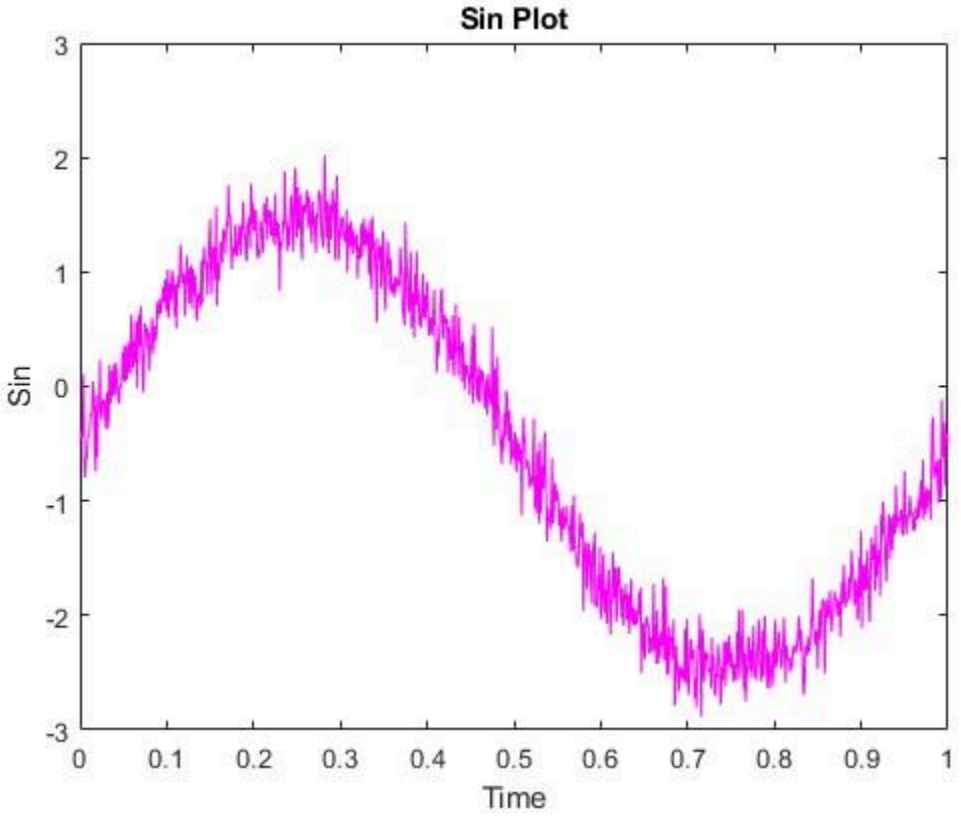
تمرین 2_2)

کد استفاده شده در این بخش در تصویر 7 آورده شده است.

```
load p2.mat;
plot(t, y, 'm');
title('Sin Plot');
xlabel('Time');
ylabel('Sin');
grid off;
```

تصویر 7 - کد بخش 2_2

کد تصویر 7 همانند بخش قبل، در ابتدا فایل داده شده را لود کرده و سپس متغیر y را بر حسب متغیر زمانی t پلات می‌کند. نمودار حاصل را در تصویر 8 مشاهده می‌کنیم.



تصویر 8 - نمودار نهایی بخش 2_2

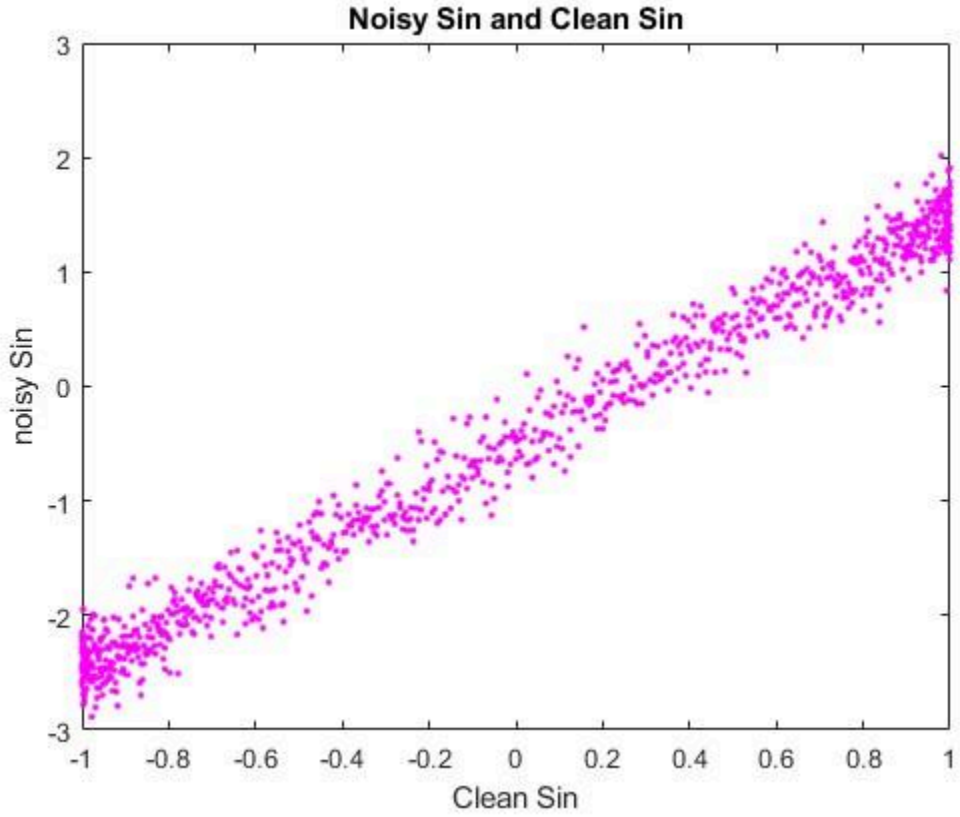
تمرین 2_3)

کد استفاده شده در این بخش در تصویر 9 آورده شده است.

```
load p2.mat
plot(x, y, '.m');
title('Noisy Sin and Clean Sin');
xlabel('Clean Sin');
ylabel('noisy Sin');
grid off;
```

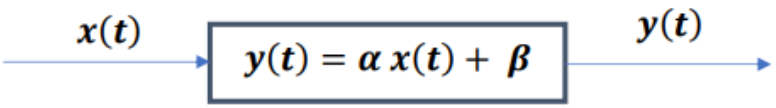
تصویر 9 - کد بخش 3_2

کد تصویر 9 در ابتدا فایل داده شده را لود کرده و سپس متغیر y را بر حسب متغیر x پلات می‌کند. نمودار حاصل را در تصویر 10 مشاهده می‌کنیم.



تصویر 10 - نمودار نهایی بخش 3_2

عرض از مبدا و شیب خط بالا به ترتیب همان متغیرهای β و α هستند که در صورت پروژه هم اشاره شده (تصویر 11)



تصویر 11 - معادله سیستم سوال 2

تمرین 2_4)

در این بخش قصد داریم با استفاده از Linear Regression و تابع هزینه Mean Squared Error پارامتر های β و α مورد بحث در بخش قبل را تخمین بزنیم. توضیحات ریاضی مربوط به این بخش در تصویر 12 آورده شده است.

$$\text{Estimate } \alpha \to \hat{\alpha} \qquad \text{Estimate } \beta \to \hat{\beta} \qquad \text{Estimate } y = \alpha x + \beta \to \hat{y} = \hat{\alpha} x + \hat{\beta}$$

$$\frac{\partial MSE}{\partial \beta} = \frac{2}{n}\sum_i^n \frac{\partial (y_i - \hat{y}_i)}{\partial \beta} \times (y_i - \hat{y}_i) = \frac{2}{n}\sum_i^n \frac{\partial (\alpha x_i + \beta - \hat{\alpha} x_i - \hat{\beta})}{\partial \beta} \times (y_i - \hat{y}_i) = \frac{2}{n}\sum_i^n 1 \times (\alpha x_i + \beta - \hat{\alpha} x_i - \hat{\beta}) = 0$$

$$n \times \hat{\beta} = \sum_i^n (\alpha x_i + \beta - \hat{\alpha} x_i) \quad \to \quad \hat{\beta} = \frac{1}{n}\sum_i^n (y_i - \hat{\alpha} x_i) \quad \to \quad \hat{\beta} = \frac{1}{n}\sum_i^n y_i - \frac{\hat{\alpha}}{n}\sum_i^n x_i \quad \to \quad \hat{\beta} = \bar{y} - \hat{\alpha}\bar{x}$$

$$\frac{\partial MSE}{\partial \alpha} = \frac{2}{n}\sum_i^n \frac{\partial (y_i - \hat{y}_i)}{\partial \alpha} \times (y_i - \hat{y}_i) = \frac{2}{n}\sum_i^n \frac{\partial (\alpha x_i + \beta - \hat{\alpha} x_i - \hat{\beta})}{\partial \alpha} \times (y_i - \hat{y}_i) = \frac{2}{n}\sum_i^n x_i \times (\alpha x_i + \beta - \hat{\alpha} x_i - \hat{\beta}) = 0$$

$$\to \sum_i^n \hat{\alpha} x_i^2 = \sum_i^n x_i \times (y_i - \hat{\beta}) \quad \to \quad \sum_i^n \hat{\alpha} x_i^2 = \sum_i^n x_i \times (y_i - \bar{y}) + \hat{\alpha} x_i \bar{x}$$

$$\to \sum_i^n \hat{\alpha} x_i (x_i - \bar{x}) = \sum_i^n x_i \times (y_i - \bar{y}) \quad \to \quad \hat{\alpha} \times (\sum_i^n (x_i^2) - n \times \bar{x}^2) = (\sum_i^n x_i y_i) - n \times \bar{x}\bar{y}$$

$$\to \hat{\alpha} = \frac{n \times \bar{x}\bar{y} - \sum_i^n (x_i y_i)}{n \times \bar{x}^2 - \sum_i^n (x_i^2)} \quad \to \quad \hat{\alpha} = \frac{\bar{x}\bar{y} - \frac{1}{n} \times \sum_i^n (x_i y_i)}{\bar{x}^2 - \frac{1}{n} \times \sum_i^n (x_i^2)} \quad \to \quad \hat{\alpha} = \frac{mu(x)mu(y) - mu(xy)}{mu(x)^2 - mu(x^2)}$$

تصویر 12 - محاسبات ریاضی تخمین پارامترهای β و α

برای محاسبه پارامترهای ذکر شده توسط روشی که در بالا توضیح داده شده، تابعی نوشتم که در تصویر 13 قابل مشاهده است.

```
function [a, b] = linear_regressor(x, y)
    len = length(x);
    mu_x = mean(x);
    mu_y = mean(y);
    xx = zeros(1, len);
    xy = zeros(1, len);

    for i = 1:len
        xx(i) = x(i) * x(i);
        xy(i) = x(i) * y(i);
    end

    mu_xx = mean(xx);
    mu_xy = mean(xy);

    a = ((mu_x * mu_y) - mu_xy) / ((mu_x * mu_x) - mu_xx);
    b = mu_y - (a * mu_x);
end
```

تصویر 13 - تابع تخمین پارامترهای خط

با استفاده از این تابع دو پارامتر خواسته شده را تخمین می‌زنیم. نتیجه در تصویر 14 آورده شده است.

```
load p2.mat
[a, b] = linear_regressor(x, y);
display(a);
display(b);
```

```
Command Window
>> p2_4

a =

    1.9736


b =

   -0.4983
```

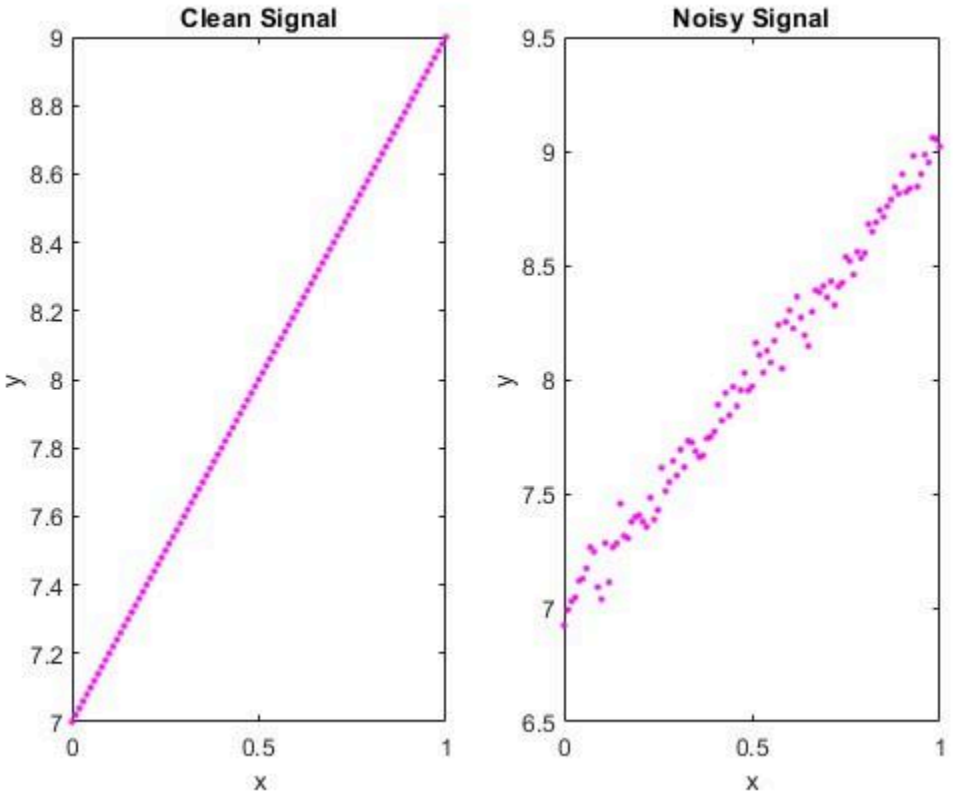
تصویر 14 - نتیجه اجرای تابع تخمین

جهت اطمینان از درستی تابع نوشته شده، دوباره و با مقادیر جدید از آن استفاده می‌کنیم (تصویر 15)

```
x = 0:0.001:1;
a = 2;
b = 7;
y = a * x + b;
len = length(x);

noisy_y = zeros(1, len);
noise = 0.2 * randn(1, len);

for i=1:len
    noisy_y(i) = y(i) + noise(i);
end

figure;
subplot(1, 2, 1);
plot(x, y, '.m');
title('Clean Signal')
xlabel('x')
ylabel('y')

subplot(1, 2, 2);
plot(x, noisy_y, '.m');
title('Noisy Signal')
xlabel('x')
ylabel('y')

[a1, b1] = linear_regressor(x, y);
[a2, b2] = linear_regressor(x, noisy_y);
```

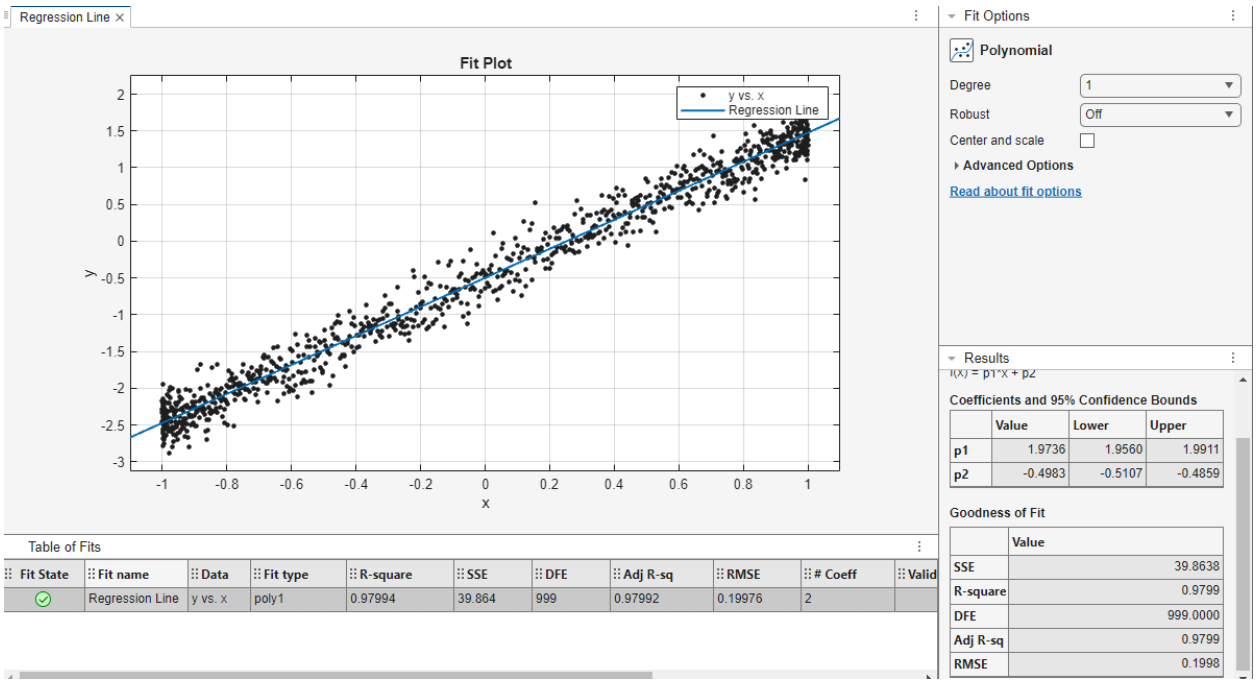
تصویر 15 - تولید سیگنال نویزی و تخمین پارامترهای آن

نتیجه کد تصویر 15 را در زیر مشاهده می‌کنیم.



تمرین 2_5)

این بار از ابزار آماده متلب برای فیت کردن منحنی روی دیتا داده شده استفاده می‌کنیم (تصویر 17)



تصویر 17 - نتیجه فیت کردن منحنی

توضیحاتی درباره خروجی بالا:

- یک منحنی چندجمله‌ای درجه اول (خط) روی دیتا فیت کردیم.
- مقدار پارامترهای تخمینی زده شده توسط تابع نوشته شده با تابع آماده متلب دقیقا یکسان هستند.
- معیار R squared، برای سنجش مدل‌های Regression به کار می‌رود. این معیار نشان می‌دهد چه مقدار از واریانس متغیر وابسته توسط متغیر(های) مستقل توضیح داده می‌شود. در این مدل ما R squared برابر 0.98 گرفتیم که عدد بسیار خوبی است.

بخش سوم

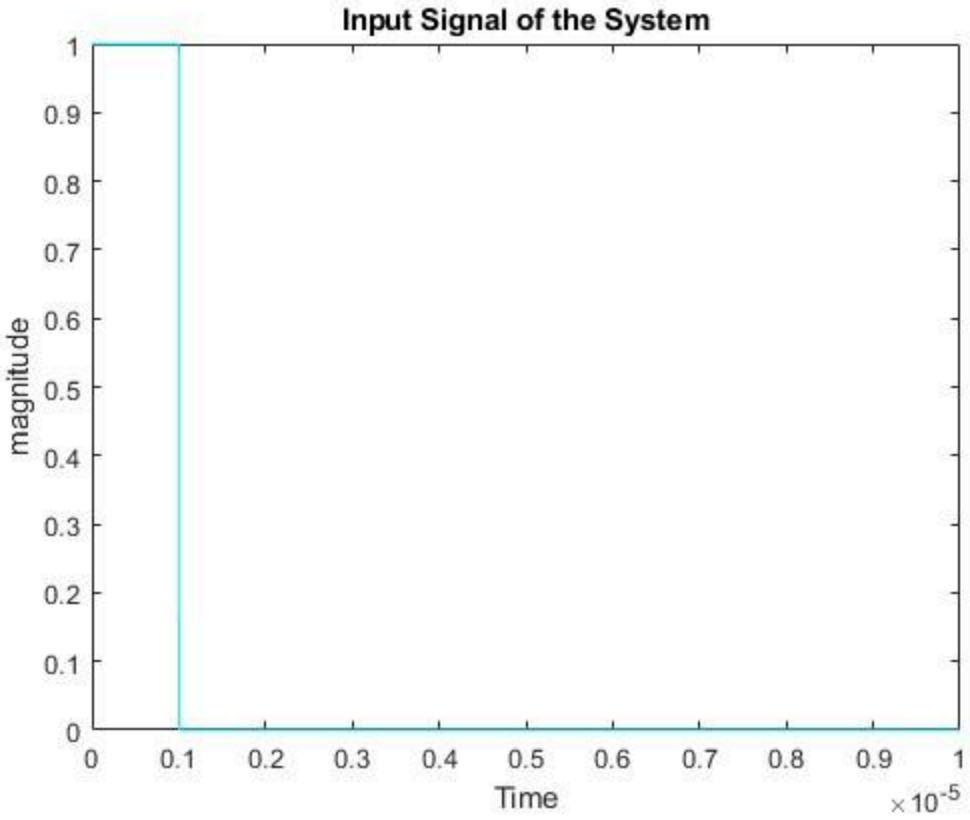
تمرین 3_1)

ts	اندازه هر گام زمانی / فاصله زمانی سمپل‌های متوالی	10^{-9}
fs	فرکانس سیگنال / تعداد سمپل‌ها در یک واحد زمانی	$\frac{1}{ts} = 10^{9}$
T	دوره تناوب / کوچک‌ترین بازه زمانی تکرار سیگنال	10^{-5}
tau	طول زمانی سیگنال پالس	10^{-6}
nums	تعداد کل سمپل‌ها	$\frac{T}{ts} = 10^{4}$
nump	تعداد سمپل‌های بخش پالس	$\frac{tau}{T} * nums = 10^{3}$

تعداد کل سمپل‌ها به این معناست که چند سمپل در یک دوره تناوب جای می‌گیرند.
تعداد پالس‌ها به این معناست که چند سمپل از یک تناوب، متعلق به بخش پالس هستند.
در آخر دو آرایه ones و zeros با سایزهای محاسبه‌شده با هم concatenate شده‌اند.

```
function signal=create_input_signal()
    ts = 1e-9; % time step
    t = ts:ts:1e-5; % time axis
    fs = 1e9; % frequency
    T = 1e-5; % period
    tau = 1e-6; % pulse width or duration of a single radar pulse
    alpha = 1; % value of the pulse signal

    num_samples = int32(T / ts);
    num_pulses = (tau / T) * num_samples;

    pulse = alpha * ones(1, num_pulses);
    rest_of_signal = zeros(1, num_samples - num_pulses);
    signal = cat(2, pulse, rest_of_signal);
```

تصویر 18 - کد تولید سیگنال پالس



تصویر 19 - سیگنال پالس ورودی

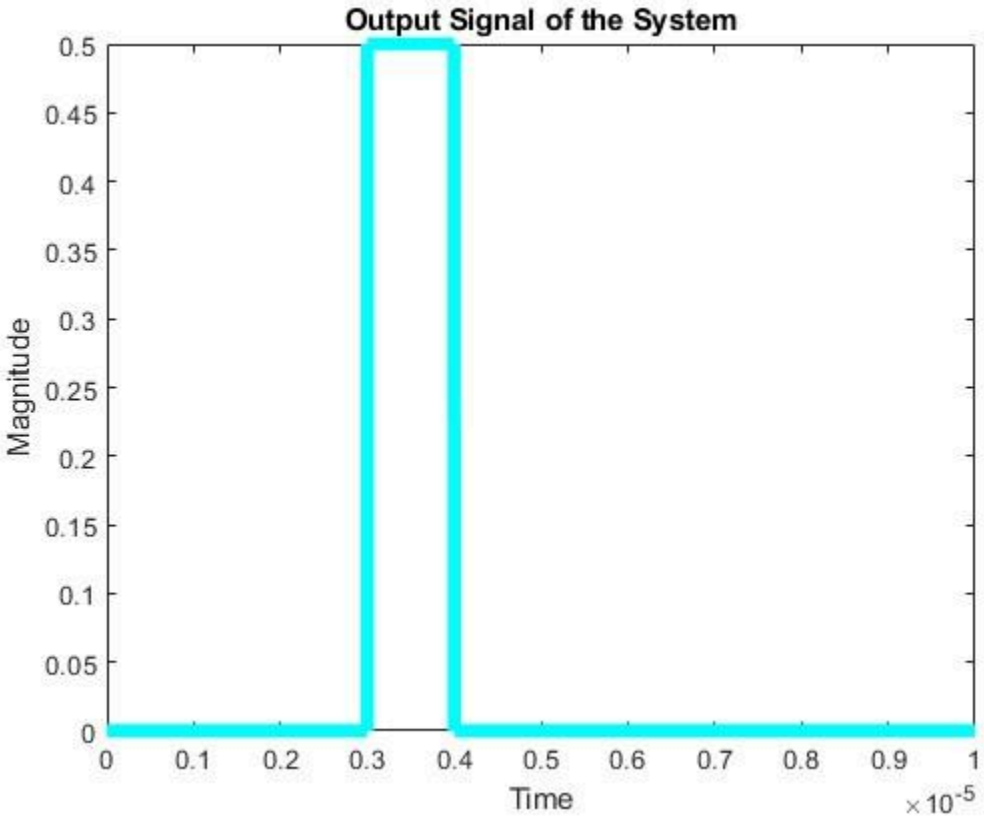
تمرین 3_2)

سیگنال خروجی در حالت ایده‌آل (بدون نویز)، شیفت داده شده سیگنال ورودی است. در واقع این سیگنال در ابتدا با صفر شروع می‌شود، سپس با مقدار یک (یا آلفا) ادامه میابد و در انتها دوباره صفر می‌شود. سه سیگنال گفته شده در آخر concatenate می‌شوند. محاسبات اندازه سیگنال‌ها همانند بخش 3_1 است و توضیحی نیاز ندارد.

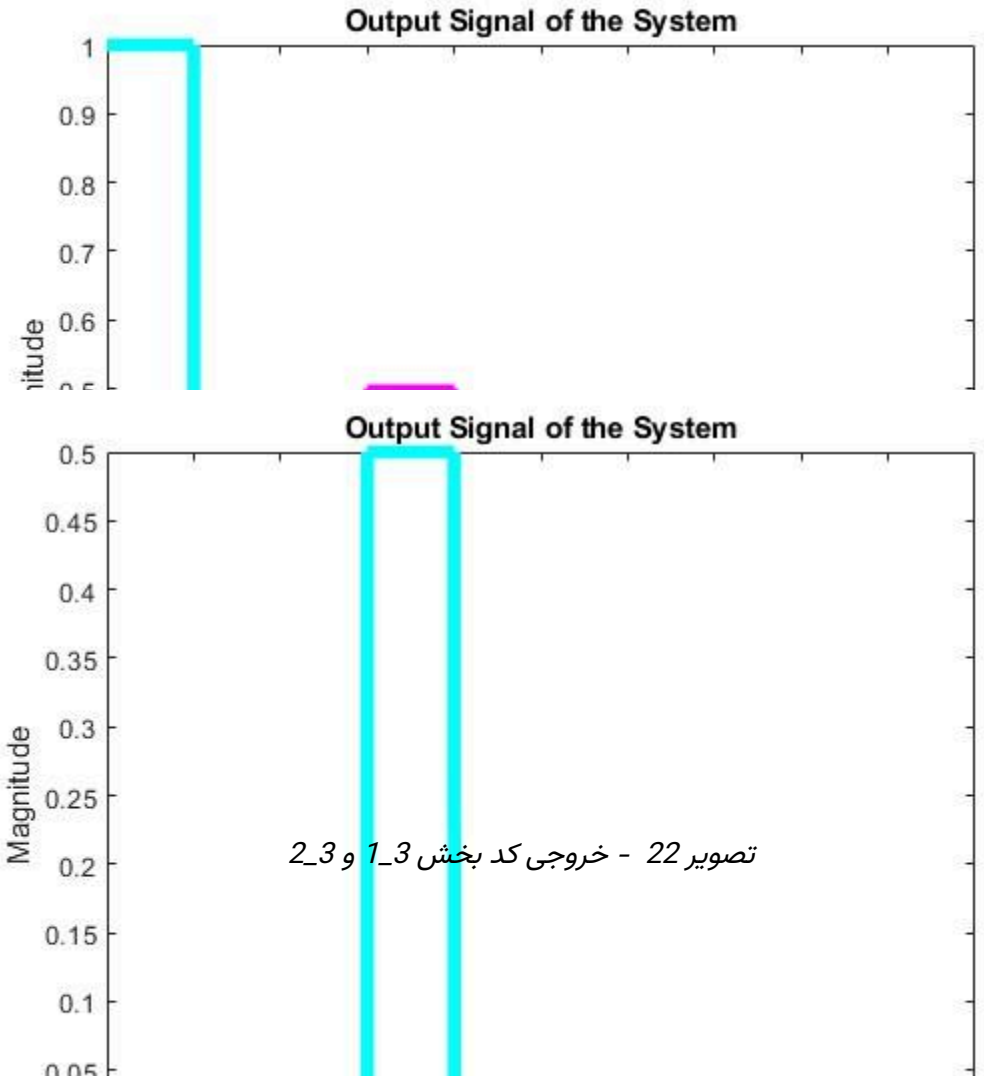
```
function signal=create_output_signal()
    ts = 1e-9; % time step
    t = ts:ts:1e-5; % time axis
    fs = 1e9; % frequency
    T = 1e-5; % period
    tau = 1e-6; % pulse width or duration of a single radar pulse
    alpha = 0.5; % value of the pulse signal

    R = 450;
    C = 3e8;
    td = 2 * R / C

    num_samples = int32(T / ts);

    num_zeros_1 = int32(td / ts);

    num_pulses = (tau / T) * num_samples;

    pulse = alpha * ones(1, num_pulses);

    num_zeros_2 = num_samples - num_zeros_1 - num_pulses;

    signal = cat(2, zeros(1, num_zeros_1), pulse, zeros(1, num_zeros_2));
```

تصویر 20 - کد بخش 3_2

خروجی کد بالا به این صورت است (تصویر 21). پالس خروجی در زمان $3e-6$ تا $4e-6$ جای گرفته است. همچنین دو سیگنال ورودی و خروجی همراه هم در تصویر 22 قابل مشاهده هستند.



تصویر 21 - خروجی کد بخش 3_2



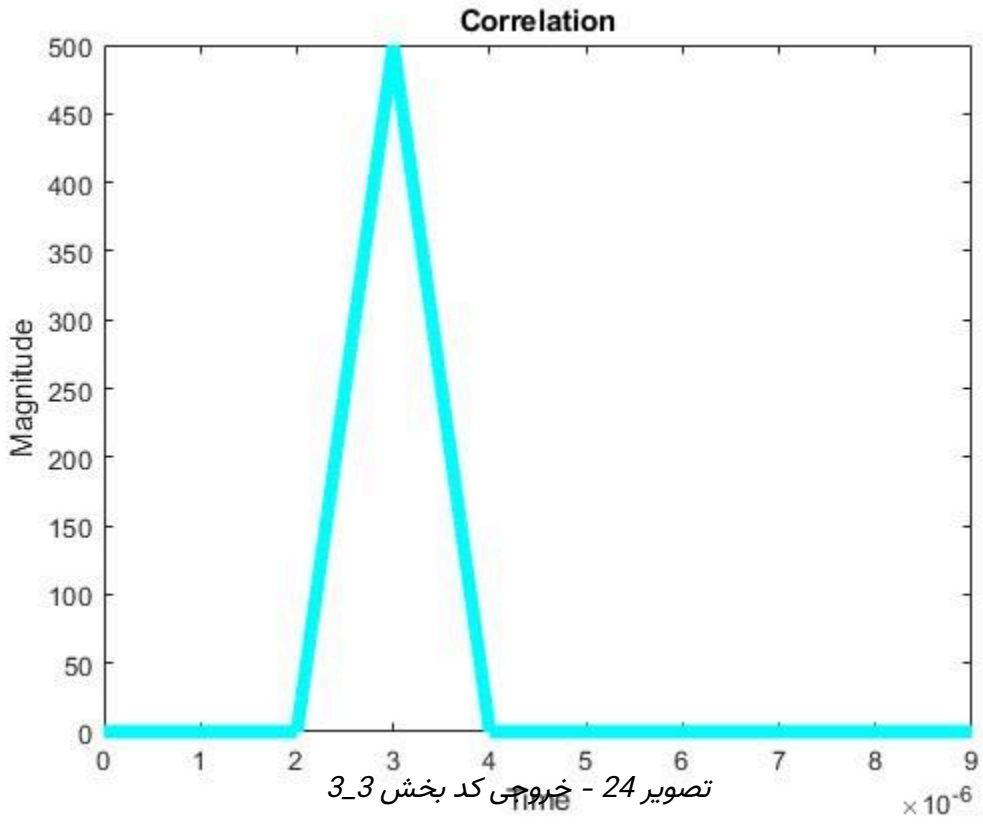
تصویر 22 - خروجی کد بخش 3_1 و 3_2

تمرین 3_3)

با روش Correlation به حل این بخش می‌پردازیم. به معیاری نیاز داریم که شباهت سیگنال خروجی و ورودی بسنجد. از آنجایی که سیگنال‌های ما پالس هستند (همه جا صفر به جز بخشی یک)، از ضرب داخلی استفاده می‌کنیم. هر جا ضرب داخلی ماکسیمم شد، احتمالا الگوی مد نظر یافت شده است.

```
function [distance, correlation]=find_distance(signal, template)
    len_t = length(template);
    len_s = length(signal);
    correlation = zeros(1, len_s - len_t - 1);

    for i = 1 : length(correlation)
        correlation(i) = dot(template, signal(i:i+len_t-1));
    end

    [~, idx] = max(correlation);
    C = 3e8 ;
    ts = 1e-9;
    td = (idx - 1) * ts;
    distance = td * C / 2;
end
```

تصویر 23 - کد بخش 3_3



تصویر 24 - خروجی کد بخش 3_3

همانطور که مشخص است در زمان $3e-6$ الگوی مدنظر دوباره آغاز شده است که با مقدار تئوری همخوانی دارد. همچنین فاصله 450 متر تخمین زده شد که صحیح است.

تمرین 3_4)

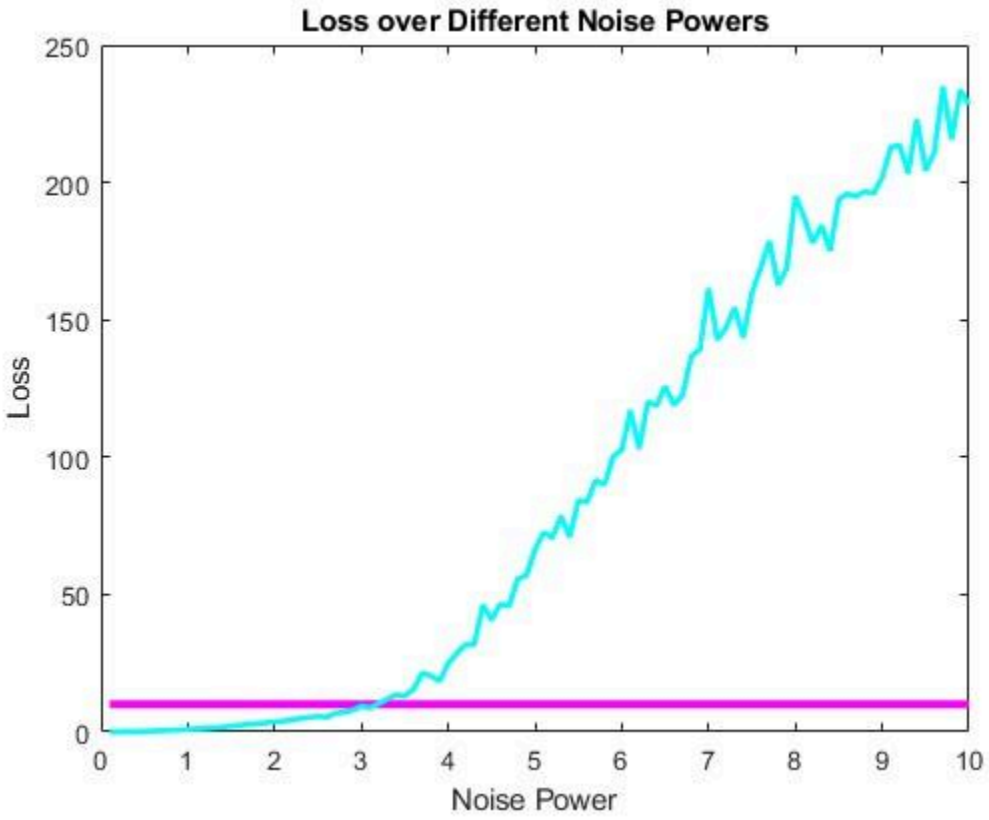
```
function [noise_powers, losses]=simulate_noises()
    ts = 1e-9; % time step
    T = 1e-5; % period
    tau = 1e-6; % pulse width or duration of a single radar pulse
    num_samples = int32(T / ts);
    num_pulses = (tau / T) * num_samples;
    R = 450;

    signal = create_output_signal();
    len_s = length(signal);
    pulse = ones(1, num_pulses);

    num_repeats = 1000;
    noise_powers = 0.1:0.1:10;
    len_n = length(noise_powers);
    losses = zeros(1, len_n);

    for i=1:len_n
        loss = zeros(1, num_repeats);
        for j=1:num_repeats
            noise = noise_powers(i) * randn(1, len_s);
            noisy_signal = signal + noise;
            [distance, ~] = find_distance(noisy_signal, pulse);
            loss(j) = abs(R - distance);
        end
        losses(i) = mean(loss);
    end
    fprintf('%s', 'Finished Simulating!');
end
```

تصویر 25 - کد بخش 3_4

تابع بالا به تعداد num_repeats دفعه، با نویزهای مختلف تخمین فاصله را انجام می‌دهد و در نهایت میانگین loss ها را برمی‌گرداند.

همانطور که در نمودار صفحه بعد (تصویر 26) مشخص است، با افزایش قدرت نویز میزان خطا هم افزایش میابد.

در صورت تمرین گفته شده که تا خطای 10 متر قابل چشم‌پوشی است. همانطور که از نمودار زیر مشخص است تا زمانی که قدرت نویز از حدود 3 بیشتر نشده باشد، همچنان با دقت قابل قبولی کار می‌کند.



تصویر 26 - خروجی کد بخش 3_4

بخش چهارم

تمرین 4_1)

در ابتدا فایل را با دستور audioread در محیط MATLAB باز می‌کنیم. سپس با استفاده از اطلاعات دریافت شده می‌توان گفت که:

- سیگنال از 376832 سمپل تشکیل شده است
- فرکانس سیگنال 48000 است
- مدت زمان ویس 7.850667 ثانیه است

```
voice_file_path = './myVoice.wav';
[x, fs] = audioread(voice_file_path);
fprintf('There are %d Samples in the Discrete Signal\n', length(x));
fprintf('The Frequency of the Signal is %d\n', fs);
fprintf('Duration of the Voice File is %f Seconds\n', length(x) / fs);
sound(x, fs);
```

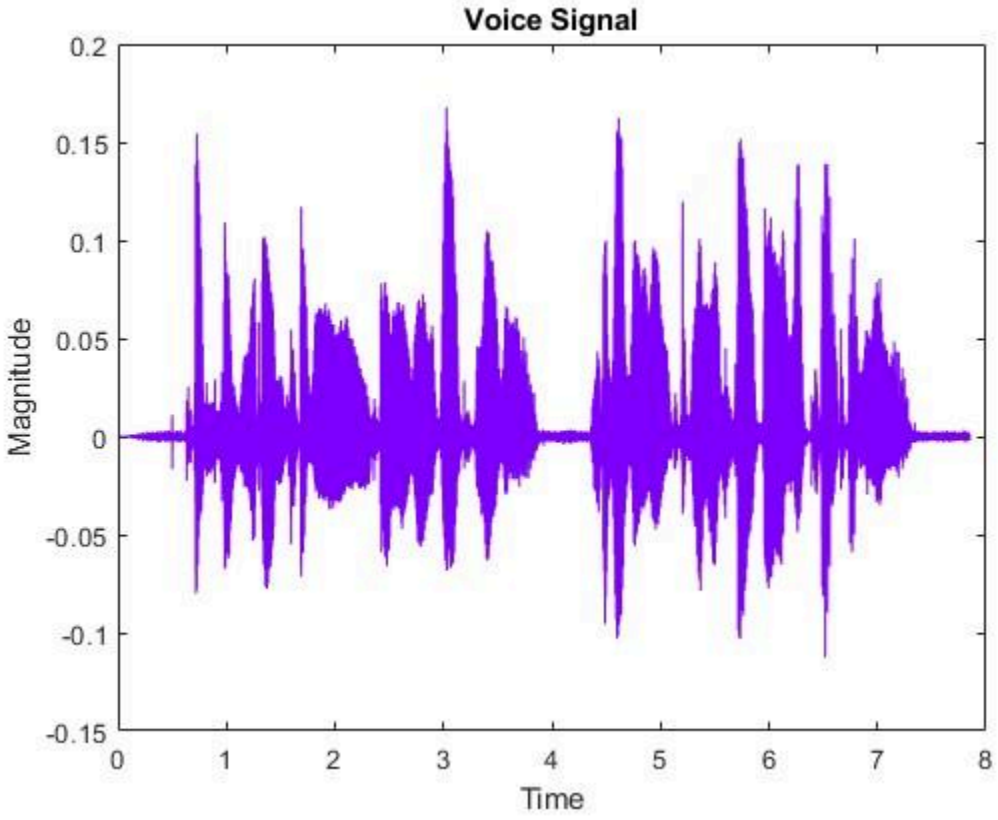
تمرین 4_2)

کد زیر فایل را باز می‌کند، آن را ذخیره می‌کند و برحسب زمان به ثانیه رسم می‌کند.

```
voice_file_path = './voice.wav';
[x, fs] = audioread(voice_file_path);
ts = 1 / fs;
num_samples = length(x);
duration = length(x) / fs;
t = ts:ts:duration;

audiowrite('x.wav', x, fs);

figure;
plot(t, x, 'Color', [0.5, 0, 1]);
title('Voice Signal');
xlabel('Time');
ylabel('Magnitude');
```

تصویر 27 - کد بخش 4_2

نمودار خروجی کد بالا را در زیر مشاهده می‌کنیم.



تصویر 28 - نمودار سیگنال ویس بخش 4

بخش 4_3)

تابع زیر سیگنال دریافتی را دو برابر تند یا کند می‌کند. هر دو حالت را بررسی می‌کنیم:

- تند: یکی در میان سیگنال‌ها را برداشته و دور می‌ریزد. N سمپل را به سقف $\frac{N}{2}$ سمپل می‌رساند.
- کند: بین هر دو سیگنال، میانگین آن‌ها را قرار می‌دهد. N سمپل را به $2N + 1$ سمپل می‌رساند.

توجه کنید که فرکانس سیگنال خروجی با سیگنال ورودی یکسان است و کاری با آن پارامتر نداریم.
اگر سرعت درخواستی برابر 2 یا 0.5 نباشد اروری چاپ می‌شود.

```
function y=p4_3(x, speed)
    num_samples = length(x);
    y = zeros(1, num_samples);

    if speed == 2
        num_output_samples = floor((num_samples + 1) / 2);
        y = zeros(1, num_output_samples);
        for i=1:num_output_samples
            y(i) = x(2 * i);
        end

    elseif speed == 0.5
        num_output_samples = num_samples * 2 - 1;
        y = zeros(1, num_output_samples);
        for i=1:num_samples-1
            y(2*i - 1) = x(i);
            y(2*i) = (x(i) + x(i + 1)) / 2;
        end
        y(2*num_samples - 1) = x(num_samples);
    else
        fprintf('Please Set Speed as 2 or 0.5\n');
        return;
    end
end
```

تصویر 29 - کد بخش 4_3

بخش 4_4)

در این بخش باید تابع قسمت قبل را تعمیم دهیم به گونه‌ای که هر سرعتی بین 0.5 تا 2 که مضرب 0.1 باشد را اجرا کند.

- وقتی سرعت سیگنالی تغییر می‌کند و همان فرکانس قبل را دارد، تعداد سمپل‌ها هم تغییر می‌کند.
- بسته به بزرگ‌تر یا کوچک‌تر بودن سرعت از یک، تعداد سمپل‌ها می‌تواند کمتر یا بیشتر از سمپل‌های اولیه شود.
- هر سمپل در سیگنال جدید می‌تواند یک سمپل از سیگنال اولیه یا ترکیبی از دو سمپل متوالی در آن باشد.

با توجه به نکات بالا این ایده تابع زیر را پیاده‌سازی می‌کنیم.

```
function y=p4_4(x, speed)
    old_len = length(x);
    new_len = round(old_len / speed);
    y = zeros(1, new_len);

    if rem(speed, 0.1) ~= 0
        fprintf('Speed argument must be divisible by 0.1!\n');
        return;
    end

    if (speed > 2) || (speed < 0.5)
        fprintf('Speed argument must be between 0.5 and 2!\n');
        return;
    end

    for i = 1:new_len
        org_idx = (i - 1) * speed + 1;
        if org_idx <= old_len
            low_idx = floor(org_idx);
            up_idx = ceil(org_idx);
            if low_idx == up_idx
                y(i) = x(low_idx);
            else
                delta_y = (org_idx - low_idx) * (x(up_idx) - x(low_idx));
                y(i) = x(low_idx) + delta_y;
            end
        end
    end
end
```

تصویر 30 - کد بخش 4_4

توضیح کد بالا:

- 2-4: تعریف چند متغیر
- 6-9: چک کردن اینکه سرعت داده شده مضرب 0.1 است یا خیر
- 11-14: چک کردن اینکه سرعت داده شده میان 0.5 و 2 است یا خیر
- 16: حلقه روی کل سمپل‌های جدید
- 17: پیدا کردن اندیس متناظر در سمپل اولیه
- 19-20: پیدا کردن کف و سقف اندیس متناظر
- 21-22: حالتی که سمپل متناظر سمپل کنونی در سیگنال اولیه وجود دارد
- 23-25: حالتی که سمپل متناظر وجود ندارد و باید آن را تخمین بزنیم

-پایان گزارش کار تمرین کامپیوتری اول-