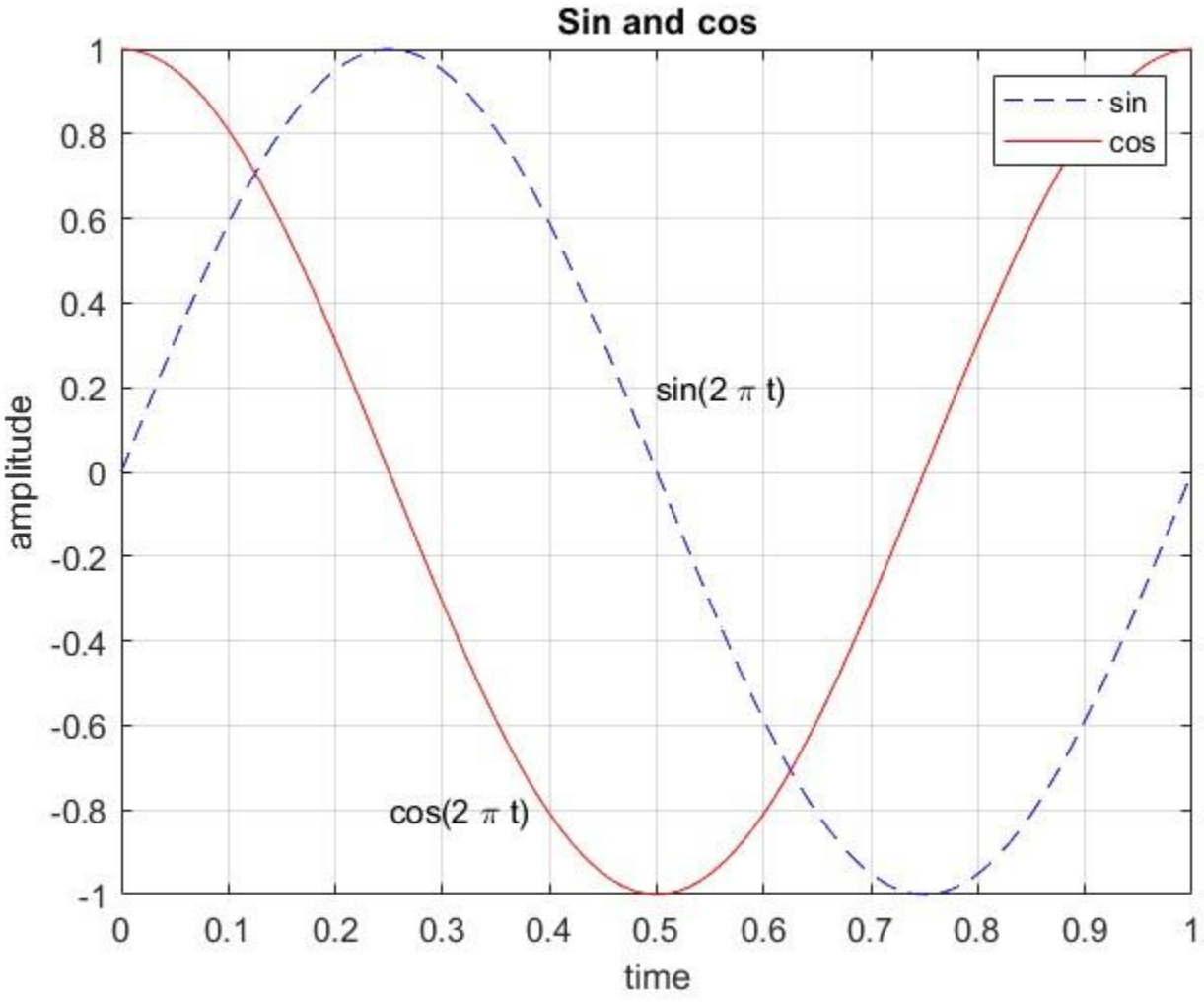
گزارش کار تمرین کامپیوتری یک سیگنال و سیستم

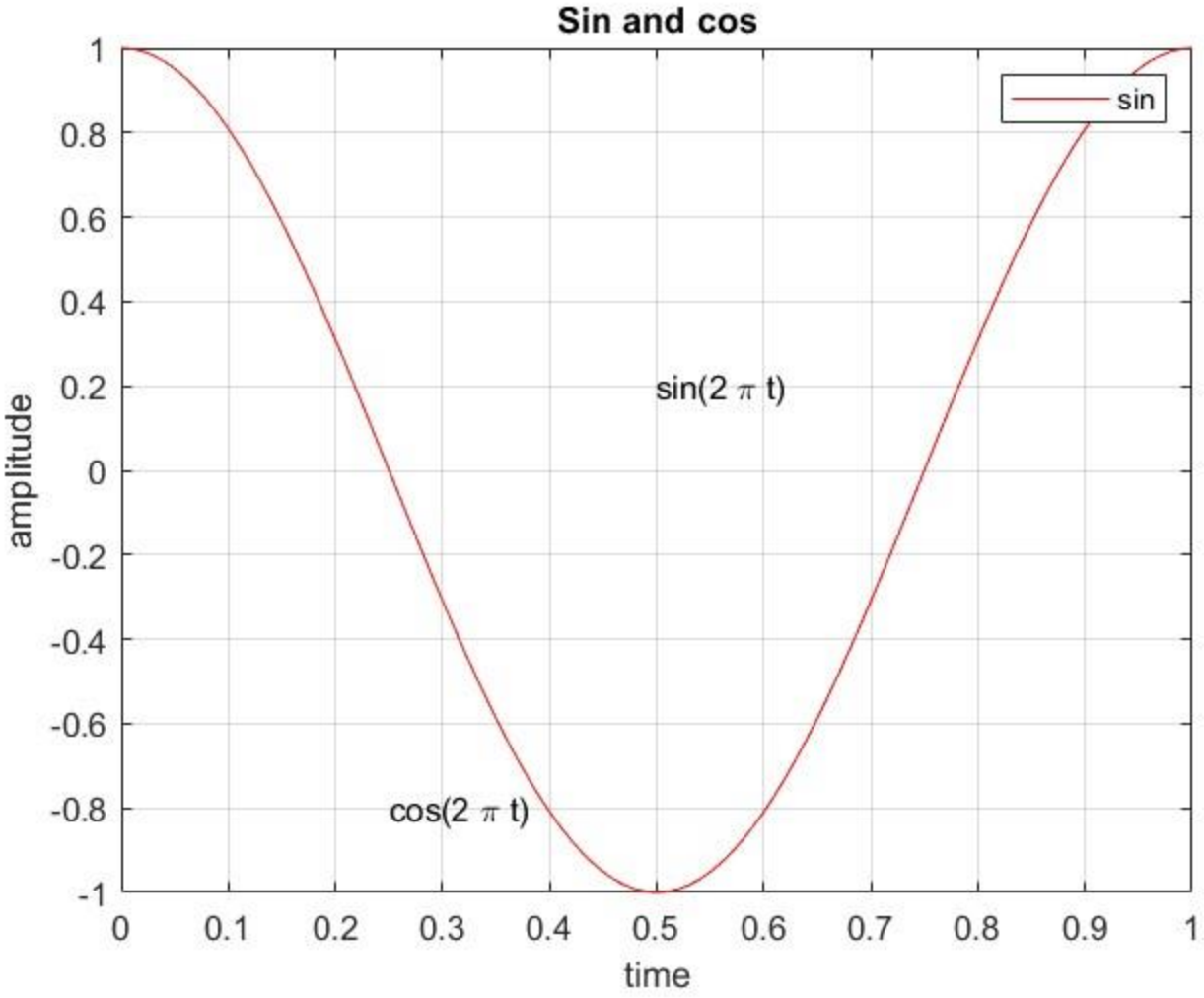
مرضیه موسوی-810101526

1-1

شکل نهایی به صورت زیر است:



در صورت حذف خط 7 نمودار اول که مربوط به سینوس است روی فیگور نمی ماند:

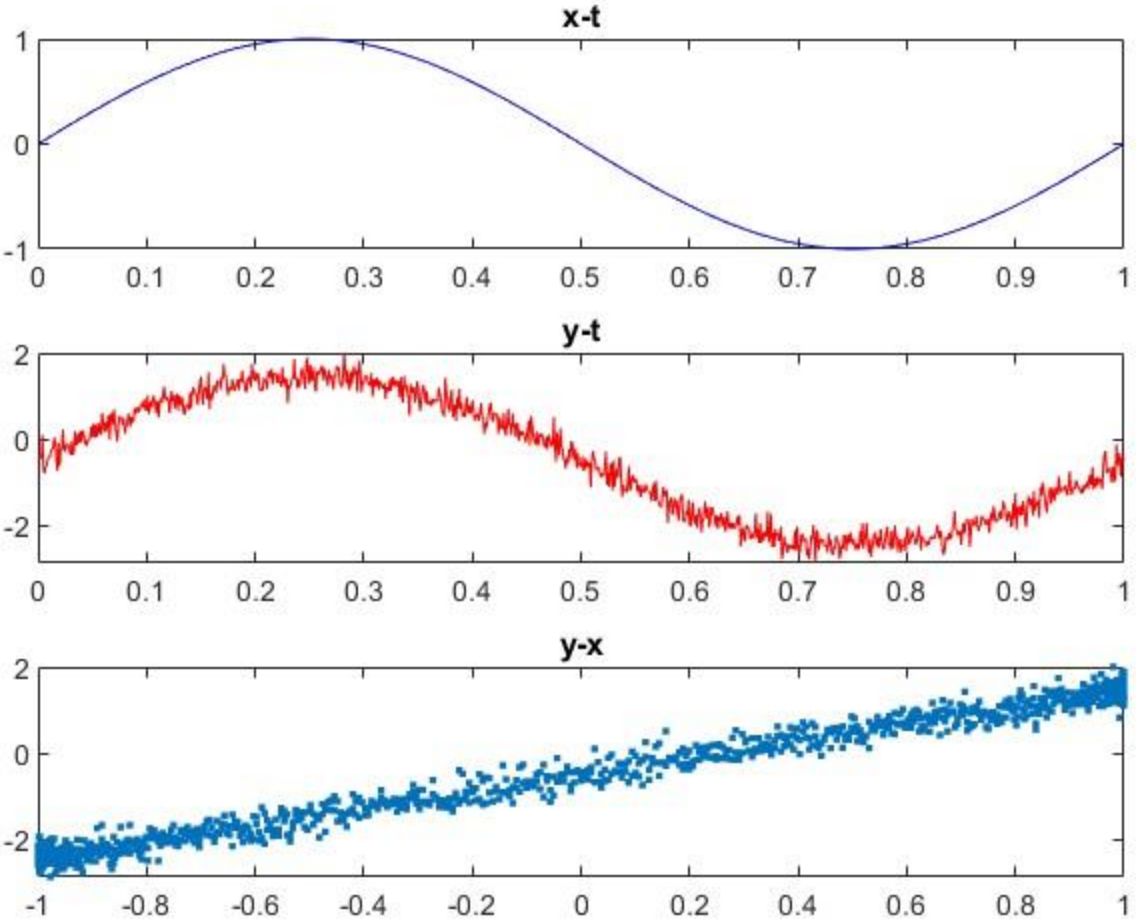

Sin and cos
sin
sin(2 π t)
cos(2 π t)
amplitude
time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

```
t=0:0.01:1;
z1=sin(2*pi*t);
z2=cos(2*pi*t);
figure;
subplot(2,1,1);
x0=[0.5,0.25];
y0=[0.2,-0.8];
s1=["sin(2 \pi t)"]
s2=['cos(2 \pi t)']
plot(t,z1,'--b')
text(0.5,0.25,s1);
title ("Sin");
legend('sin');
xlabel('time');
ylabel('amplitude');
grid on;
subplot(2,1,2);
plot(t,z2,'r')
text(0.2,-0.8,s2);

title ("cos");
legend('cos');
xlabel('time');
ylabel('amplitude');
grid on;
```

بخش دوم:

شکل نمودار های بخش یک تا سه به شرح زیر است:



به نظر می رسد شیب نمودار y-x که همان ضریب آلفا در رگرسیون خطی ماست تقریبا برابر 2 و عرض از مبدا که همان ضریب بتا در رگرسیون خطی است تقریبا کمی کوچکتر از 0 است.

تمرین 2-4

باید از loss function داده شده یکبار نسبت به آلفا و بار دیگر نسبت به بتا مشتق بگیریم و با حل کردن معادلات داده شده آلفا و بتا مورد نظر را بیابیم.

مشتق تابع loss داده شده نسبت به آلفا:

$$\frac{\partial L}{\partial a} = -2\sum_{i=1}^{n}(y_i - (ax_i + b))x_i = 0$$

مشتق تابع loss داده شده نسبت به بتا:

$$\frac{\partial L}{\partial b} = -2\sum_{i=1}^{n}(y_i - (ax_i + b)) = 0$$

بنابراین باید از دو معادله ی زیر آلفا و بتا را بدست آوریم:

$$n\beta + a\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$$

$$a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i$$

اسکریپت متلب به صورت زیر است:

که مطابق با معادلات به دست آمده است.

```
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
```

بخش تست:

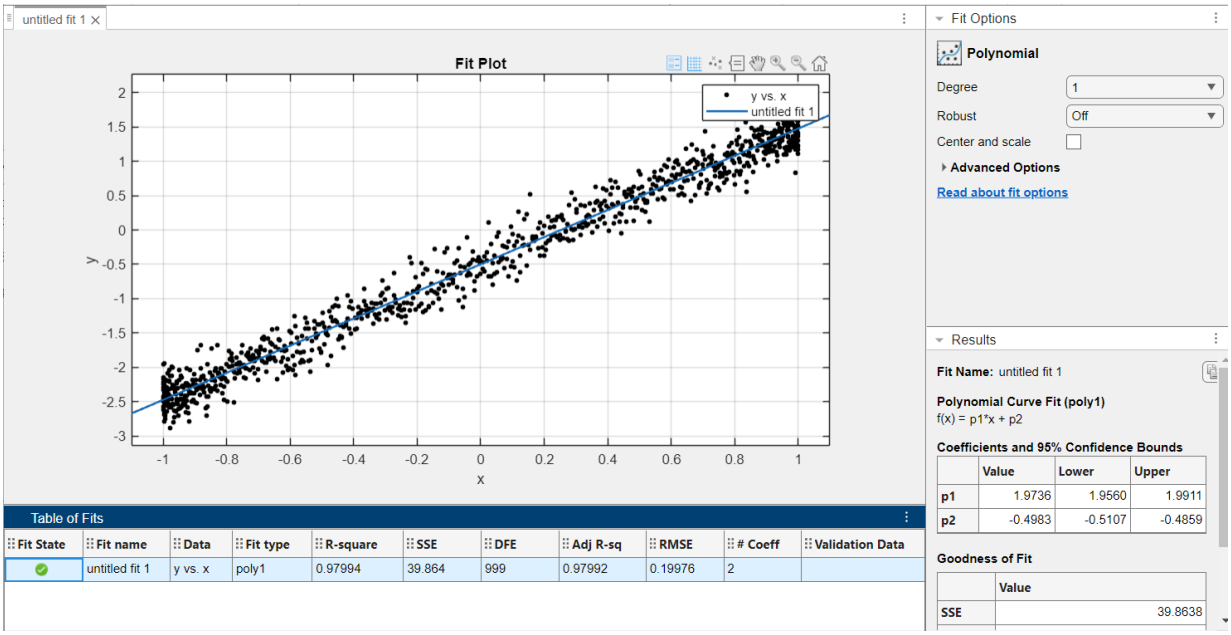
اسکریپت زیر برای تست نوشته شده:

```
x_2 = [1 2 3 4 5];
alpha=2;
beta=3;
dimension = [1, 5];
random_matrix = randn(dimension);
y_2=alpha*x_2+beta+random_matrix;
[aprim,bprim]=estimateCoefficients(x_2, y_2);
```

همانطور که در کامند مشاهده می کنید آلفا و بتای بدست آمده نزدیک به آلفا و بتای محاسبه شده می باشد:

```
>> estimate_q_with_noise_test
The value of alpha  is: 1.4165
The value of beta is: 4.9265
>> estimate_q_with_noise_test
The value of alpha  is: 1.9893
The value of beta is: 3.1228
>> estimate_q_with_noise_test
The value of alpha  is: 2.0492
The value of beta is: 2.5260
```

(2-5



همانطور که در تصویر مشخص است نتایج همخوانی دارد

بخش سه:

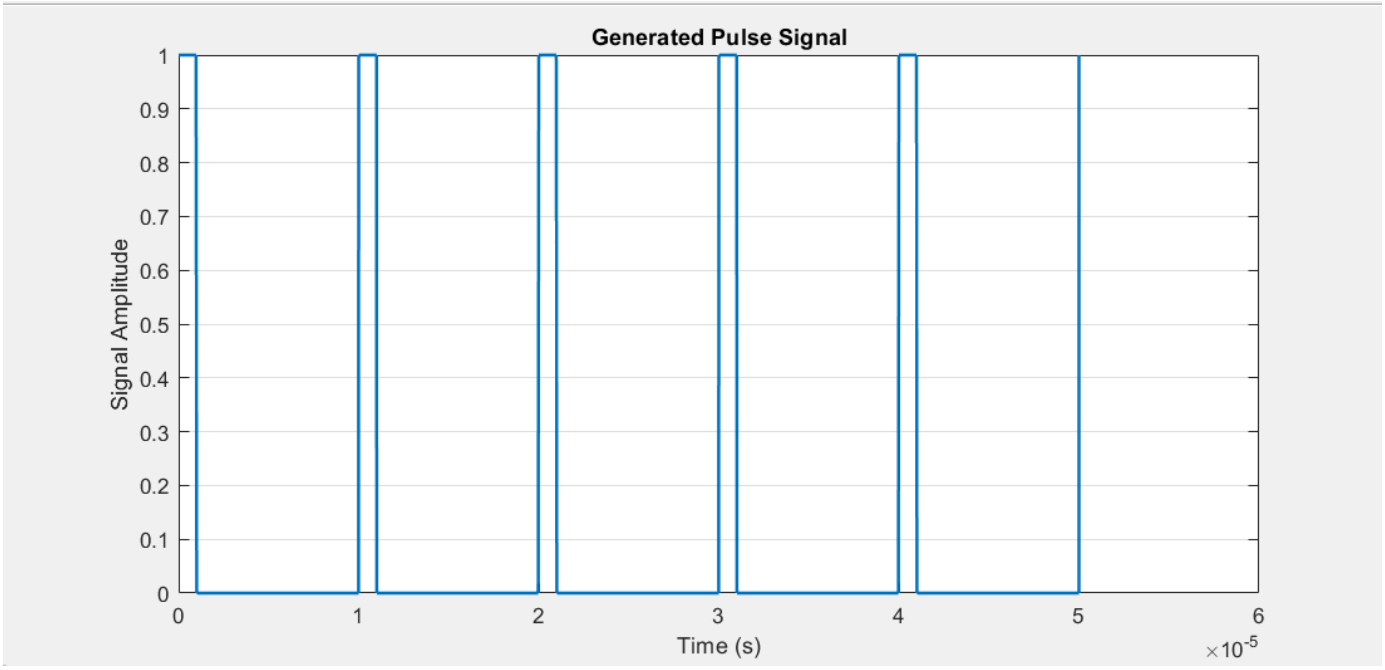
(1-3

اسکریپت متلب به صورت زیر است:

```
ts = 1e-9;
T = 1e-5;
tau = 1e-6;
t = 0:ts:5*T;
signal = zeros(size(t));
signal(mod(t, T) < tau) = 1;
figure;
plot(t, signal, 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Signal Amplitude');
title('Generated Pulse Signal');
grid on;
```

همانطور که مشخص است ابتدا وکتوری از 0 به اندازه بازه زمانیمان گرفتیم.سپس در هر بازه زمانی اگر بین 0 و tau را یک قرار دادیم.(tهایی که در آن بازه هستند)

شکل سیگنال به صورت زیر است:



(3-2

اسکریپت متلب به صورت زیر است:

```
T = 1e-5;
ts = 1e-9;
tau = 1e-6;
R=450;
c = 299792458;
t_d=2*R/c;
t = 0:ts:5*T;
alpha=0.5;
signal = zeros(size(t));
for n = 0:floor(max(t)/T)
    start_time = n * T + t_d;
    end_time = start_time + tau;
    signal(t >= start_time & t < end_time) = alpha;
end
figure;
plot(t, signal, 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Signal Amplitude');
title('recieved Pulse Signal');
grid on;
```

همانطور که مشخص است یک وکتور سیگنال به اندازه ی بازه زمانی و با مقدار 0 گرفته ایم و سپس در هر دوره ی تناوب ابتدا زمان شروع سیگنال و زمان پایان آن را حساب کرده ایم و آن بخش را به اضافه ی آلفا کردیم.

(3-3

می دانیم کورلیشن دو سیگنال مربعی یک سیگنال مثلثی می باشد.در این سوال اگر سیگنالی به عنوان باکس کانولوشن و سیگنال دریافتی را باهم کانولوشن بگیریم چون ماکسیمم سیگنال مثلثی ایجاد شده در td می باشد بنابراین td بدست می آید و با بدست آورد td می توانیم R را حساب کنیم.

اسکریپت متلب به صورت زیر است:

```
ts = 1e-9;
T = 1e-5;
tau = 1e-6;
t = 0:ts:T;
t_len=length(t);

c = 300000000;
R=450;

t_d=2*R/c;
alpha=0.5;
signal2 = zeros(t_len);

for n = 0:floor(max(t)/T)
    start_time = n * T + t_d;
    end_time = start_time + tau;
    signal2(t >= start_time & t < end_time) = alpha;
end
N = 0:ts:tau;
correlationBox= ones(1,length(N));
ro= zeros(1,length(t)-length(N));
for i=1:t_len-length(N)
    h=signal2(i:i+length(N)-1);
    ro(i)=sum(correlationBox .* h);
```

```
        for i=1:t_len-length(N)
            h=signal2(i:i+length(N)-1);
            ro(i)=sum(correlationBox .* h);

        end
        plot(t(1:t_len-length(N)),ro);
        [val,index]=max(ro);
        t_d=index*ts;
        r=t_d*c/2;
        disp(r);
```

سیگنال دریافتی را با روشی که قبلا رسم کرده ایم دوباره وکتور آن را بدست می آوریم.کورلیشن باکس را نیز بدست می آوریم و سپس ro که سیگنال خروجی است را بدست می آوریم.مکسیمم مقدار آن در نقطه ی tdاست و با استفاده از td بدست آمده R را محاسبه می کنیم.

(3-4

به اسکریپت قبلی یک حلقه اضافه می کنیم.حلقه ی بیرونی قدرت نویز را تعیین می کند.عدد رندومی به دست آورده و آن را در قدرت نویز ضرب می کنیم و به سیگنال دریافتی اصلی اضافه می کنیم.از روش تمرین قبلی استفاده می کنیم و r را در هر 100 نمونه بدست آورده و میانگین می گیریم.اگر میانگین بین 440 و 460 بود یعنی خطای ده درصدی داشت تقریب ما درست است.در غیر این صورت تقریب ما مشکل دارد.

```
ts = 1e-9;
T = 1e-5;
tau = 1e-6;
t_len=((1e-5)/(1e-9))+1;
t = 0:ts:T;
c = 3e8;
R=450;
t_d=2*R/c;
alpha=0.5;
signal2 = zeros(size(t));
for n = 0:floor(max(t)/T)
    start_time = n * T + t_d;
    end_time = start_time + tau;
    signal2(t >= start_time & t < end_time) = alpha;
end
num_sample=100;
noise_power=20;
for j=1:0.5:noise_power
    sum_r=0;
    for i=1:num_sample
        noise=j*rand(1,t_len);

       noisy_signal=signal2+noise;
        N = 0:ts:tau;
       correlationBox= ones(1,length(N));
       out_core= zeros(1,t_len-length(N));
    for k=1:t_len-length(N)
    h=noisy_signal(k:k+length(N)-1);

    out_core(k)=sum(correlationBox .* h);
```

```
    end
  [val,index]=max(out_core);
  t_d=index*ts;
  r=t_d*c/2;
  sum_r=sum_r+r;
    end

  mean_r=(sum_r/num_sample);
  disp("the distance is");
  disp(mean_r);
  if (mean_r>440 && mean_r<460)

      disp("The distance estimated correctly");
      disp(" ");
  else
      disp("The distance didn't estemated correctly");
      disp(" ");


  end
end
```

خروجی به صورت زیر است:

```
The distance estimated correctly

the distance is
  450.0570

The distance estimated correctly

the distance is
  450.0330

The distance estimated correctly

the distance is
  450.1545

The distance estimated correctly

the distance is
  450.0930

The distance estimated correctly

the distance is
  450.2130

The distance estimated correctly

the distance is
  450.0255
```

```
The distance didn't estemated correctly

the distance is
  467.8065

The distance didn't estemated correctly

the distance is
  456.2775

The distance estimated correctly

the distance is
  468.9450

The distance didn't estemated correctly

the distance is
  505.6140

The distance didn't estemated correctly

the distance is
  503.8365

The distance didn't estemated correctly
```

بخش چهار)

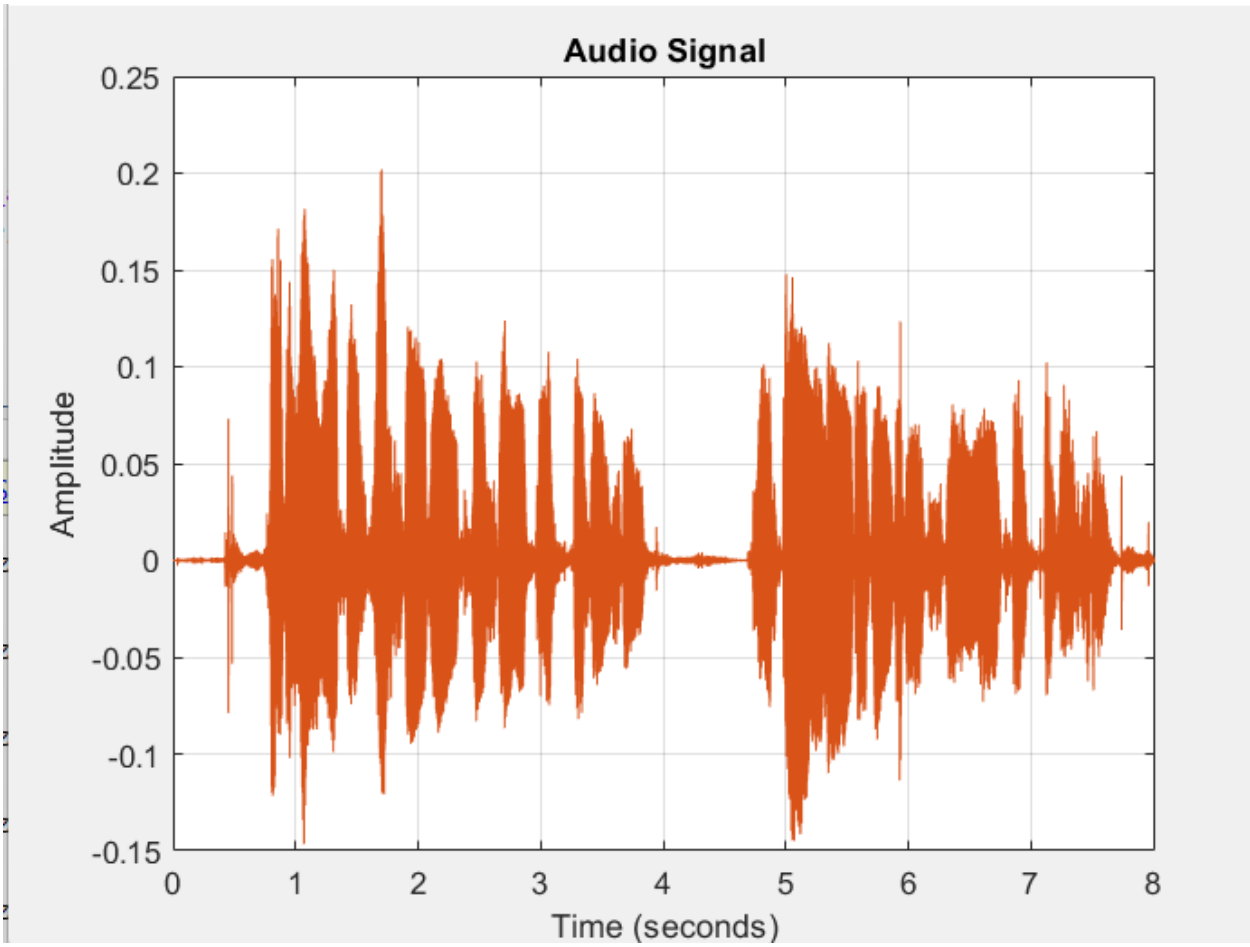
(4-1

ابتدا با استفاده از دستور audio read فرکانس نمونه برداری و دیتای صوت را خواندیم.برای نمایش اندازه ی دیتای خوانده شده را تفسیم بر فرکانس کردیم.بازه زمانی را تا آخرین t قرار دادیم و نمودار را کشیدیم

```
[audioData, fs] = audioread("Recording (4).m4a");
disp(['Sampling Frequency: ', num2str(fs), ' Hz']);

t = (0:length(audioData)-1) / fs;


figure;
plot(t, audioData);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Audio Signal');
grid on;
xlim([0 max(t)]);
```



(4-2

نحوه ی سیو کردن صوت به صورت زیر است:

```
outputFileName = 'output_audio.wav';
audiowrite(outputFileName, audioData, fs);
```

(4-3

ابتدا برای مقاوم سازی شرط گذاشتیم و ولیو های نامعتبر را نپذیرفتیم.همچنین سایز وکتور صوت جدید را با استفاده از اندازه ی صوت اولیه بدست آوردیم:

```
function p4_3(filename,speed)
[audioData, fs] = audioread("Recording (4).m4a");
n=length(audioData);
n_two_times=n/2;
n_half=2*n;
if (speed ~=2 && speed~=0.5)
    disp("not valid")
```

برای سرعت دو برابر یکی درمیان دیتاها را حذف کردیم:

برای سرعت نصف اولین و آخرین دیتا متناظر صوت اولیه می باشد و برای بقیه از میانگین دو دیتای مجاور استفاده کردیم

```
function p4_3(filename,speed)
[audioData, fs] = audioread("Recording (4).m4a");
n=length(audioData);
n_two_times=n/2;
n_half=2*n;
if (speed ~=2 && speed~=0.5)
    disp("not valid")
elseif (speed==2)
new_audio=zeros(1,n_two_times);
for i=1:n_two_times
    new_audio(i)=audioData(2*i);
end
outputFileName = 'output_audio_twotimes.wav';
audiowrite(outputFileName, new_audio, fs);
sound(new_audio,fs);
else
    new_audio=zeros(1,n_half);
    new_audio(1)=audioData(1);
    new_audio(n_half)=audioData(n);
    for i=2:n_half-1
        new_audio(i)=(audioData(i-1)+audioData(i+1))/2;
    end
    outputFileName = 'output_audio_half.wav';
    audiowrite(outputFileName, new_audio, fs);
    sound(new_audio,fs);



end
```

(4-3

برای مقاوم سازی کد مقادیر بزرگتر از 2 و کوچکتر از نیم و هرچه مضرب یک دهم نبود را نپذیرفتیم

از تابع resample برای ساخت فایل جدید استفاده می کنیم.

```
function p4_4(filename,speed)

  outputFileName = 'output_audio_extended.wav';
if(((speed*10) <5) || ((speed*10)>20) || (mod(speed*100,10)~=0) )
    disp("not valid")
else
    [audioData, fs] = audioread(filename);


          new_sound=resample(audioData,10,speed*10);
          audiowrite(outputFileName, new_sound, fs);
sound(new_sound,fs)  ;




end
```

هر دو تابع را در part 4 تست کردیم:

```
[audioData, fs] = audioread("Recording (4).m4a");
disp(['Sampling Frequency: ', num2str(fs), ' Hz']);

t = (0:length(audioData)-1) / fs;


figure;
plot(t, audioData);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Audio Signal');
grid on;
xlim([0 max(t)]);

outputFileName = 'output_audio.wav';
audiowrite(outputFileName, audioData, fs);

p4_3('Recording (4).m4a', 2);
p4_4('Recording (4).m4a', 0.5);
```