Signal&Systems CA#4

MahdiHajiSeyedHossein

Part 1

در این قسمت باید به رمزگذاری و همچنین رمزگشایی یک پیام از طریق موج های سینوسی بپردازیم .

آماده سازی دیتاست:

در این بخش تمام کاراکتر های گفته شده در پروژه که شامل حروف کوچک ، فاصله ، نقطه و ویدرگول و همچنین علامت تعجب و سمیکالن و البته کوتیشین است را در یک cell میریزیم و همچنین نام هر یک را در بالای هر باینری کاراکتر ذخیره سازی میکنیم .

```
function Mapset=MapSet()
    Mapset=cell(2,32);
    alphabet = 'abcdefghijklmnopqrstuvwxyz .,!";';
    for i = 1:32
        Mapset{1,i} = alphabet(i);
        Mapset{2,i} = dec2bin(i-1, 5);
    end

end
```

	1	2	3	4	5	6	7	8	9	10	11	
1	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	'j'	'k'	'l'
2	'00000'	'00001'	'00010'	'00011'	'00100'	'00101'	'00110'	'00111'	'01000'	'01001'	'01010'	'0101

نوشتن کد تابع کدینگ:

در این بخش تابعی با نام coding_amp مینویسیم که ورودی های آن پیام مورد نظر و همچنین rate در نظر گرفته شده برای پیام(سرعت ارسال پیام) است . و همچنین خروجی این تابع پیام کدگذاری شده به همراه کشیدن شکل پیام کدشده است .

در ابتدا باید با استفاده از دستور strcmp که در صورت پروژه مطرح شده بود تک تک کاراکتر های پیام را با استفاده از mapSet ساخته شده ، تبدیل به باینری میکنیم و در سلول binaryMessage ذخیره میکنیم و سپس در نهایت با استفاده از دستور cell2matrix این سلول را به آرایه تبدیل میکنیم که کار با آن راحت تر باشد .

```
3
4       Mapset = MapSet();
5       binaryMessage=[];
6
7       lengthOfMessage=strlength(message);
8
9       n=(strlength(message)*5)/rate;
10      % Ba tavajoh be Farz soal Fs = 100 Hz
11      fs=100;
12
13      for i=1:lengthOfMessage
14          for j=1:32
15              if strcmp(extract(message,i),Mapset(1,j))==1
16                  binaryMessage=[binaryMessage  Mapset(2,j)];
17              end
18          end
19      end
20
21
22      binaryMessage=cell2mat(binaryMessage);
23
```

حالا در قسمت بعدی تابع رشته باینری را به رشته باینری به طول سرعت پیام تبدیل می کند و به صورت یک ماتریس که شامل این رشته بیت ها به اندازه سرعت پیام است در همان متغییر binaryMessage ذخیره میکنیم .

```
27      stringLength = length(binaryMessage);
28      loopCounter = 1;
29      for k = 1 : rate : stringLength
30          index1 = k;
31          index2 = min(k + rate - 1, stringLength);
32          out{loopCounter} = binaryMessage(index1 : index2);
33          loopCounter = loopCounter + 1;
34      end
35
36      binaryMessage = out ;
37
```

سپس در سلولی به نام x تمام جایگشتهای ممکن برای یک رشته بیت به طول سرعت ارسال پیام را ذخیره می کنیم .

```
39      x = cell(1, 2^rate);
40
41      for i=0:2^rate-1
42          x{i+1} = dec2bin(i,rate);
43      end
44
```

یک ماتریس به نام y تعریف میکنیم که ضرایبی از sin (2*pi*t) که مربوط به هر یک از جایگشتها است را در آن میریزیم .

```
44
45          y=zeros(1,2^rate);
46          for i=1:2^rate
47              y(1,i)=(i-1)/(2^rate-1);
48          end
49
```

حال ماتریسی به نام t تعریف می کنیم که نمونه های صدتایی در هر ثانیه را در آن ریخته ایم.

سپس ماتریس binaryMessage را با هر عنصر سلول x با دستورstrcmp مقایسه میکنیم و هرجا که دو ورودی برابر شوند(خروجی دستورstrcmp مساوی 1 شد) ضرایب مربوطه به آن جایگشت را از ماتریس y انتخاب کرده و در encoded_message_coef میریزیم .

```
52          t=zeros(n,100);
53          for i=1:n
54              t(i,:)=linspace(i-1,i,fs);
55          end
56          encoded_message_coef=[];
57          for i=1:n
58              for j=1:2^rate
59                  if strcmp(binaryMessage(1,i),x(1,j))==1
60                      encoded_message_coef=[encoded_message_coef y(1,j)];
61                  end
62              end
63          end
64
```

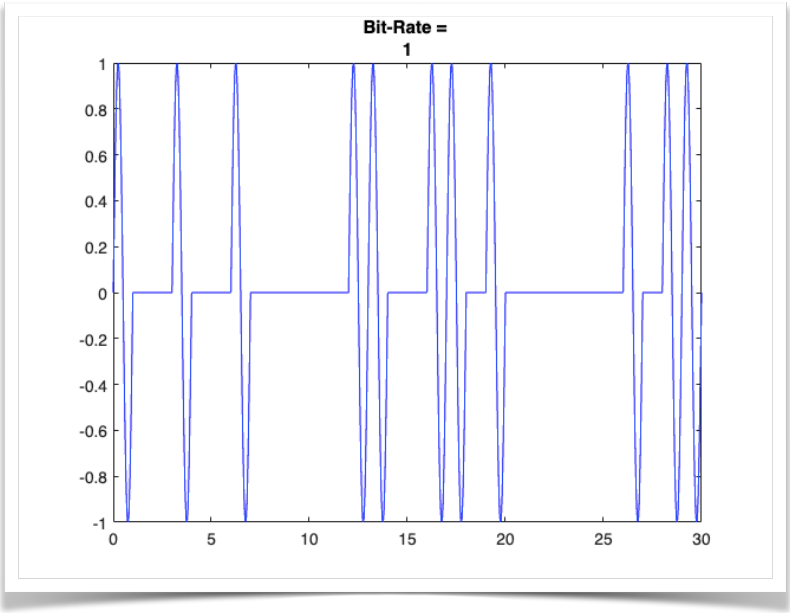
سپس ماتریس خروجی اصلی تابع را به اندازه ی مقدار ارایه t تعریف میکنیم (EncodMssage)
که در هر سطر ضریب مربوطه را در sin(2*pi*t) ضرب میکنیم ، و در ماتریس خروجی تابع میریزیم ،

```
66
67      EncodedMessage=zeros(n,100);
68      for i=1:n
69              EncodedMessage(i,:)=encoded_message_coef(1,i).*sin(2*pi*t(i,:));
70      end
71
```

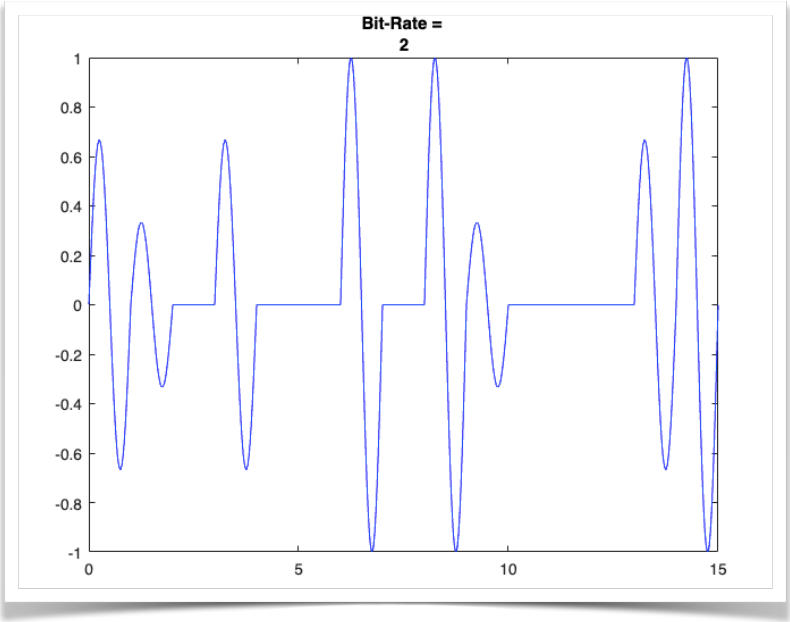
خروجی تابع همان طور که در صورت پروژه تضمین شده است ، یک ارایه به طول پنج برابر پیام
تقسیم بر سرعت ارسال پیام است که تضمین شده است که این مقدار همواره صحیح خواهد بود .

```
73      for k=1:n
74          title(["Bit-Rate = " , int2str(rate) ]);
75          plot(t(k,:),EncodedMessage(k,:),'b');
76          hold on
77      end
78  end
```

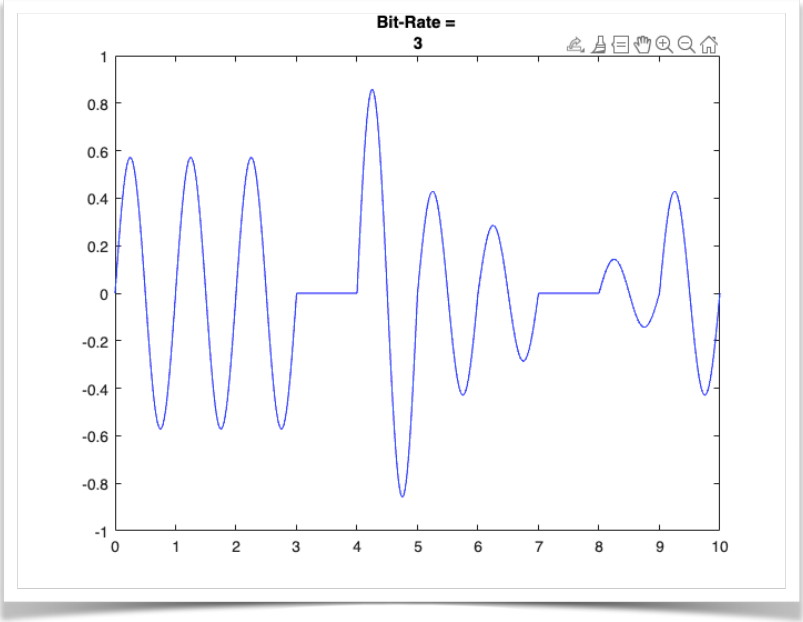
نوشتن تست پیام سیگنال:



پیام کلمه signal با bit_rate=1



پیام کلمه signal با bit_rate=2



پیام کلمه signal با bit_rate=3

نوشتن دیکود کردن پیام:

تابعی به نام گفته شده تعریف میکنیم که به عنوان ورودی پیام کدگذاری شده و سرعت ارسال پیام را میگیرد. ابتدا تابع mapsetرا فراخوانی میکنیم.

```
function decoded_message=decoding_amp(encoded_message,bit_rate)

    Mapset=MapSet();
```

مانند بخش قبل سلولی به نام x تعریف میکنیم که تمام جایگشتهای ممکن رشتههای باینری به طول سرعت ارسال اطلاعات را در آن ذخیره میکنیم و در ماتریسی به نام y ضرایب مربوط به هرجایگشت را ذخیره میکنیم.ماتریس دیگری به نام t تعریف میکنیم که نمونه های صدتایی در هرثانیه را در آن میریزیم.

```
    correlation=[];
    num=size(encoded_message);
    fs=100;
    t=zeros(num(1),100);
    for i=1:num(1)
        t(i,:)=linspace(i-1,i,fs);
    end
    % Calculating correlation
    for k=1:num(1)
        corr_2d=0.01*sum((2*sin(2*pi*t(k,:))).*(encoded_message(k,:)));
        correlation=[correlation corr_2d];
        correlation=double(correlation);
    end
    y=zeros(1,2^bit_rate);
    for i=1:2^bit_rate
        y(1,i)=(i-1)/(2^bit_rate-1);
    end
```

حال با نوشتن یک حلقه بین ماتریس ورودی پیام کدگذاری شده و 2*(2*pi*t)sin در هر ثانیه کورووليشن میگیریم و یک ضریب ۰.۰۱ در کورولیشن ضرب میکنیم تا مقایسه راحتتر شود و در سلول coroolation ذخیره میکنیم در نهایت آن را به float تغییر میدهیم.

```
14        % Calculating correlation
15        for k=1:num(1)
16            corr_2d=0.01*sum((2*sin(2*pi*t(k,:))).*(encoded_message(k,:)));
17            correlation=[correlation corr_2d];
18            correlation=double(correlation);
19        end
```

حال برای اینکه بتوانیم پیام را decode کنیم ماتریسی به نام y_m به اندازه 2 به توان bit_rate تشکیل میدهیم و در آن میانگین هر دو درایه متوالی ماتریس y را ذخیره میکنیم.

حال corollation را با y، y_mمقایسه میکنیم، اگر بین عدد وسط و عدد کوچکتر بود x متناظر با عدد کوچکتر را به آن نسبت میدهیم و اگر بین عدد وسط و عدد بزرگتر بود x متناظر به عدد بزرگتر را به آن نسبت میدهیم و در w ذخیره میکنیم.

حال با دستور cell2mat سلول w را به ماتریس تبدیل میکنیم .

```
28        end
29
30        x = cell(1, 2^bit_rate);
31        for i=0:2^bit_rate-1
32            x{i+1} = dec2bin(i,bit_rate);
33        end
34        w=[];
35        % Decision Making
36        for i=1:num(1)
37            for h=1:2^bit_rate-1
38                if abs(correlation(1,i))>y_m(1,h) && abs(correlation(1,i))<= y(1,h+1)
39                    w=[w x(1,h+1)];
40                end
41                if abs(correlation(1,i))< y_m(1,h) && abs(correlation(1,i))>= y(1,h)
42                    w=[w x(1,h)];
43                end
44                if abs(correlation(1,i))>1
45                    w=[w x(1,2^bit_rate)];
46                end
47
48            end
49        end
50        w=cell2mat(w);
```

حال تا الان توانستیم پیام رمزگذاری شده را به رشته باینری تبدیل کنیم. در سلول mapset به هر کاراکتر عدد 5 بیتی باینری نسبت دادیم. بنابراین رشته بیت باینری W را 5 تا 5 تا با فراخواندن یک قطعه کد تکراری که در فرایند رمزگذاری استفاهد کردیم جدا میکنیم هر 5 تا بیت جدا شده را با سطر دوم سلول mapset با دستور strcmp مقایسه کرده و با هرکدام که برابر بود کاراکتر مورد نظر را نسبت میدهیم و در خروجی تابع ذخیره می کنیم. سپس با دستور strjoin کاراکترهای انتخاب شده را به یک دیگر میچسبونیم .

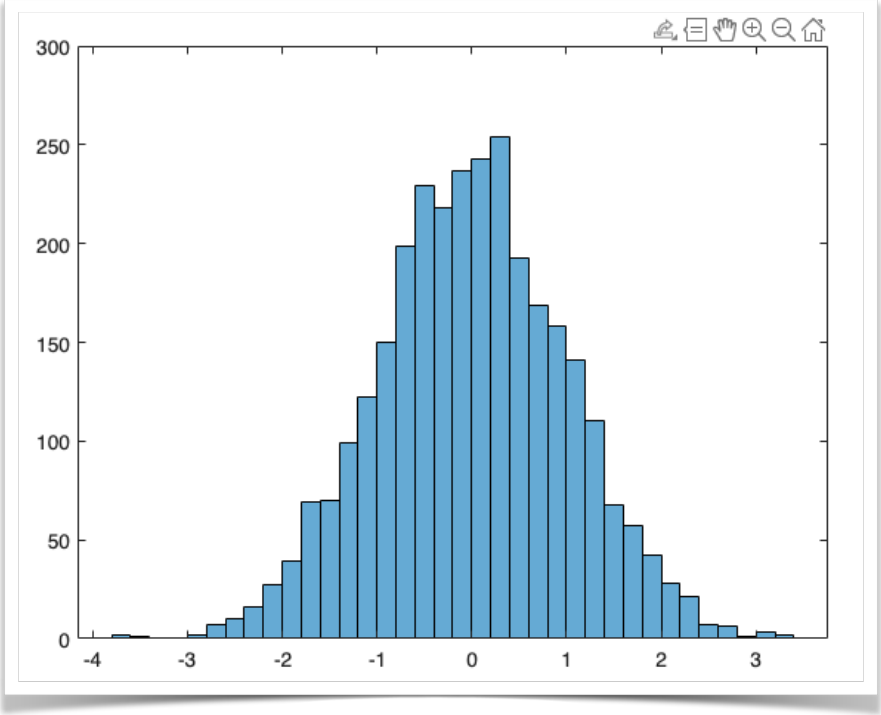
```
    stringLength = length(w);
    loopCounter = 1;
    for k = 1 : 5 : stringLength
        index1 = k;
        index2 = min(k + 5 - 1, stringLength);
        out{loopCounter} = w(index1 : index2);
        loopCounter = loopCounter + 1;
    end
    w = out ;


    decoded_message=[];
    Mapset_length=size(Mapset);
    message_length=num(1)*bit_rate/5;
    for m=1:message_length
        for n=1:Mapset_length(1,2)
            if strcmp(w(1,m),Mapset(2,n))==1
                decoded_message=[decoded_message  Mapset(1,n)];
            end
        end
    end
    decoded_message=strjoin(decoded_message,'');
    disp(decoded_message)
end
```

```
>> decoding_amp(r1 , 1 );
signal
>> decoding_amp(r2 , 2 );
signal
>> decoding_amp(r3 , 3 );
signal
>>
```

نویز گوسی:

/MATLAB Drive/CA4/Part1/GaussNoise.m

```
1   noise = randn(1 , 3000);
2
3
4   disp(["Mean for the Gaussian Noise is : " , mean(noise) ]);
5   disp(["Var for the Gaussian Noise is : " , var(noise) ]);
6   histogram(noise)
```




```
>> GaussNoise
    "Mean for the Gaussian Noise is : "    "-0.00038315"

    "Var for the Gaussian Noise is : "    "0.98829"

>>
```

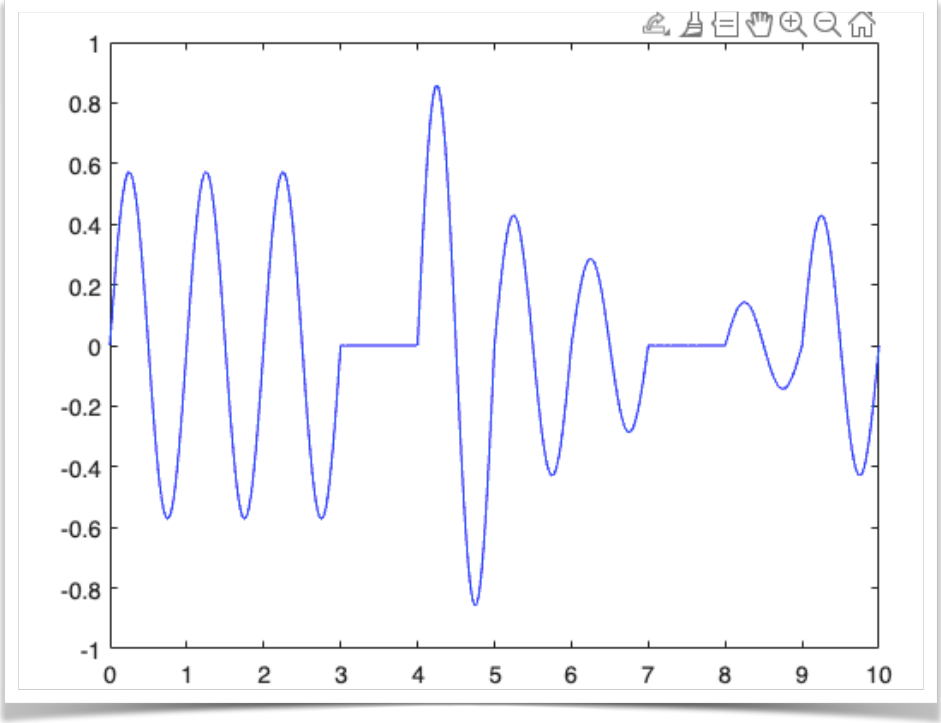
نویز اضافه کردن به پیام:

```
function message_with_noise=addingNoise(coded , sigma)
    num=size(coded);
    fs=100;
    encoded_message_size=size(coded);

    noise=sigma*randn(encoded_message_size(1),encoded_message_size(2));

    message_with_noise=coded+noise;
    t=zeros(num(1),fs);
    figure()

    for i=1:num(1)
        t(i,:)=linspace(i-1,i,fs);
    end

    for k=1:num(1)
        plot(t(k,:),message_with_noise(k,:),'b');
        hold on
    end
end

```

```


r1 = coding_amp("signal" , 1);
r2 = coding_amp("signal" , 2);
r3 = coding_amp("signal" , 3);

sigma = 0.0001;

r1n = addingNoise(r1 , sigma);
r2n = addingNoise(r2 , sigma);
r3n = addingNoise(r3 , sigma);

decoding_amp(r1n , 1);
decoding_amp(r2n , 2);
decoding_amp(r3n , 3);

```

حال شکل موج دارای نویز به صورت زیر است : (فقط برای rate=3 نمایش دادم)

(برای سیگمای ۰.۰۰۰۱)

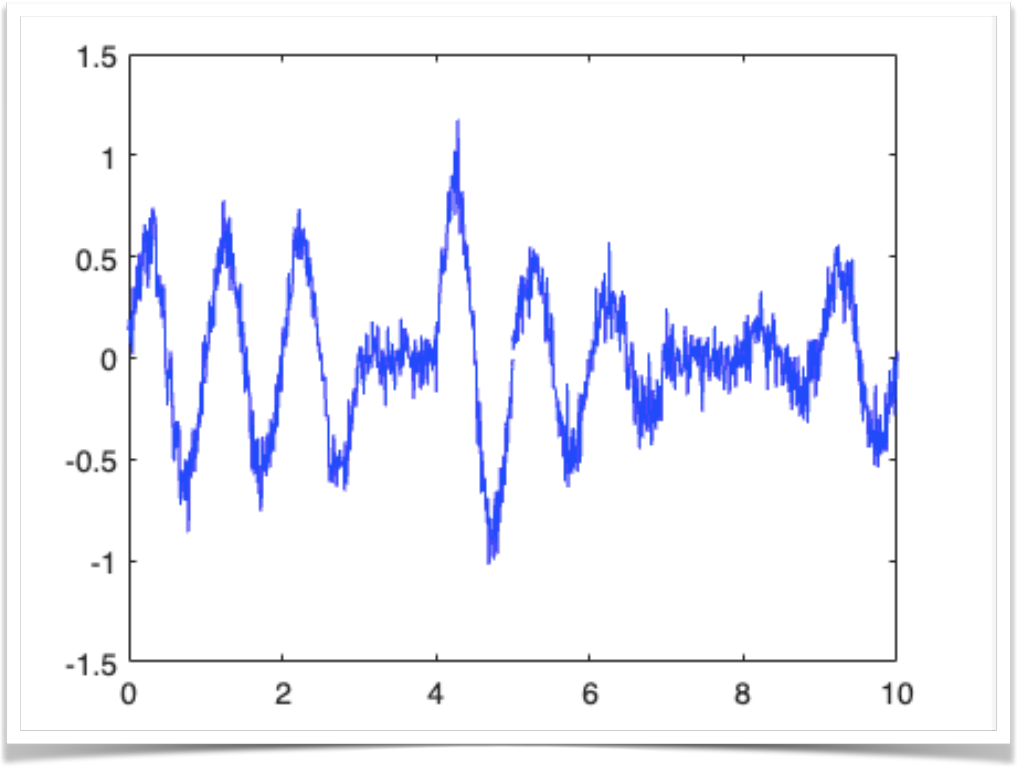


حال نتایج سیگمای 0.0001 را مشاهده میکنیم :

```
>> decodeCodedMessageWithNoise
signal
signal
signal
>>
```

حال شکل موج دارای نویز به صورت زیر است : (فقط برای rate=3 نمایش دادم)

(برای سیگمای ۱)

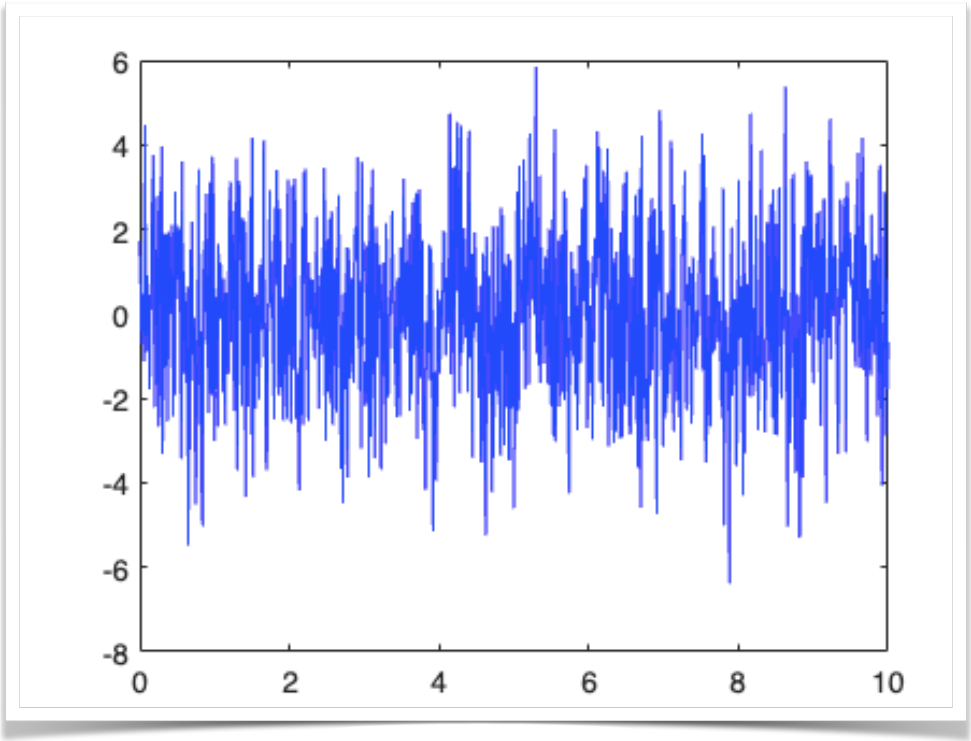


حال نتایج سیگمای 1 را مشاهده میکنیم :

```
>> decodeCodedMessageWithNoise
signal
signal
signal
>>
```

حال شکل موج دارای نویز به صورت زیر است : (فقط برای rate=3 نمایش دادم)

(برای سیگمای 2)



حال نتایج سیگمای 2 را مشاهده میکنیم :

```
>> decodeCodedMessageWithNoise
sih!al
k;zpz;
vch;;;
>>
```

بخش ۸-۱:

بیشترین واریانس نویزی که rate =3 به آن مقاوم بود چیزی حدود 0.2 بود .

بیشترین واریانس نویزی که rate =2 به آن مقاوم بود چیزی حدود1 بود.

بیشترین واریانس نویزی که rate =1 به آن مقاوم بود چیزی حدود 1.8 بود.

.

بخش ۱۰-۱:

علاوه بر نویز عوامل دیگری مثل تلفات در سیگنال موثر است که اینها نیز باعث میشوند از یه حدی بیشتر نتوان سرعت ارسال پیام را افزایش داد.

بخش ۱۱-۱:

بله .این کار عملکرد سیگنال را نسبت به نویز مقاومتر میکند.

در این روش عملا پاور سیگنال را تقویت میکنیم مثلا با ضرب کرد ۱۰ به جای۲ در قبل از سینوس مقدار بیشنیه کورووليشن از 1 به 5 تغییر میکند و همین باعث میشودکه فاصله بین threahold های درنظر گرفته بیشتر شود و در نتیجه خطا کمتر میشود.

سرعت اینترنت خانگی:

سرعت اینترنت adsl خونه ما چیزی حدود 32 مگابیت بر ثانیه است . یعنی ۳۲،۰۰۰،۰۰۰ بیت در ثانیه !

در این پروژه سرعت انتقال ما حداکثر ۳ بیت بر ثانیه بود :)))

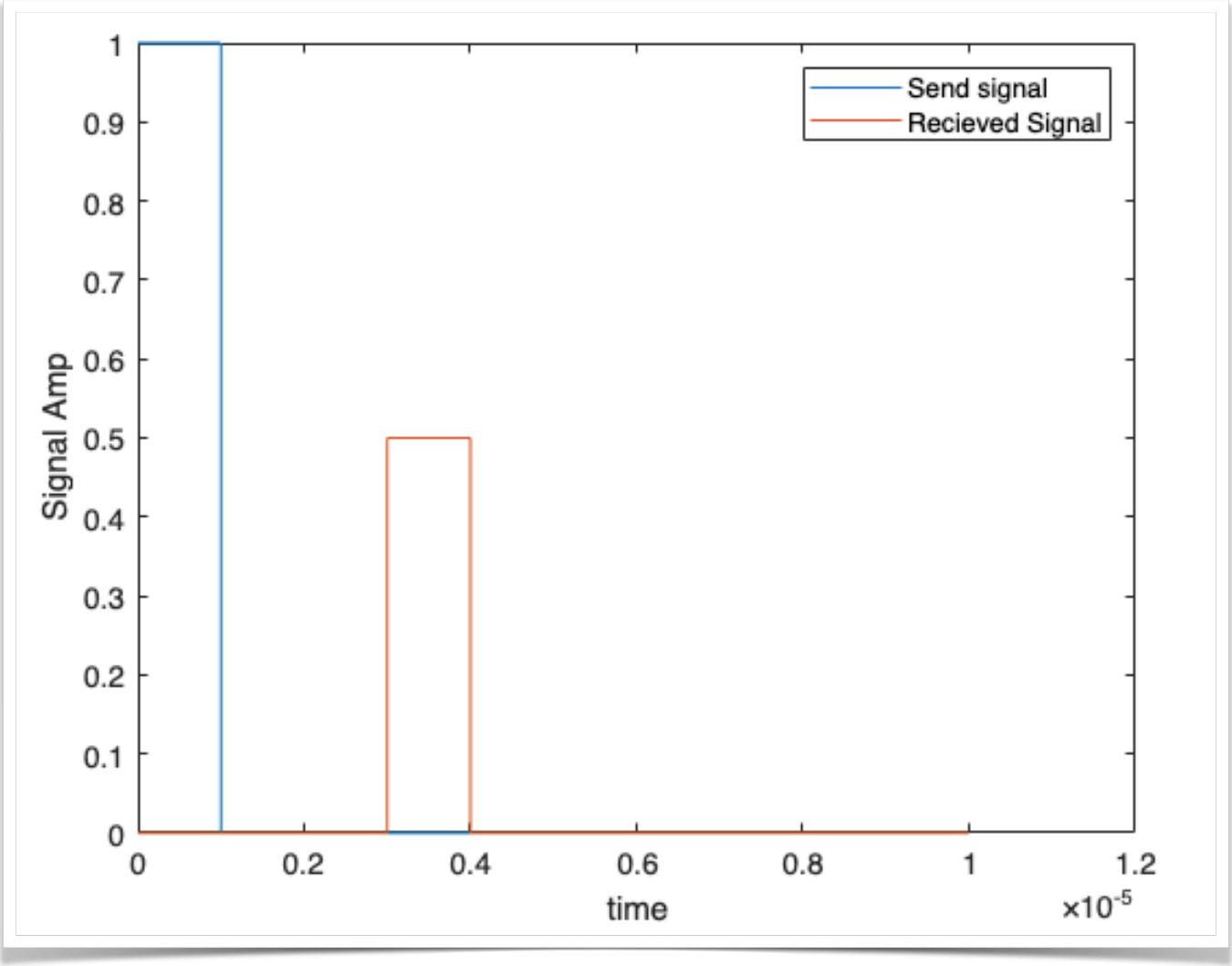
Part 2

در این قسمت باید یک رادار ماشین را که فاصله ی ماشین تا اشیا را تشخیص میدهد را پیاده سازی کنیم .

:Q1

این بخش مشابه تمرین اول است که تماما همان پارت از تمرین یک در این قسمت به صورت مشابه تکرار شده است .

```
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("Signal Amp");
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
Signal Amp
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

:Q2

این بخش باید به پیاده سازی رادار مشابه به تمرین اول بپردازیم با این تفاوت که به جای template_matching و corrolation باید از کانولووشن که در متلب با دستور conv شناخته میشود استفاده کنیم .

یک نکته مهم که در این جا قابل ذکر است size ، خروجی تابع conv است . که در ویدیوی کلاس در بخش کانولووشن به آن اشاره شده است .

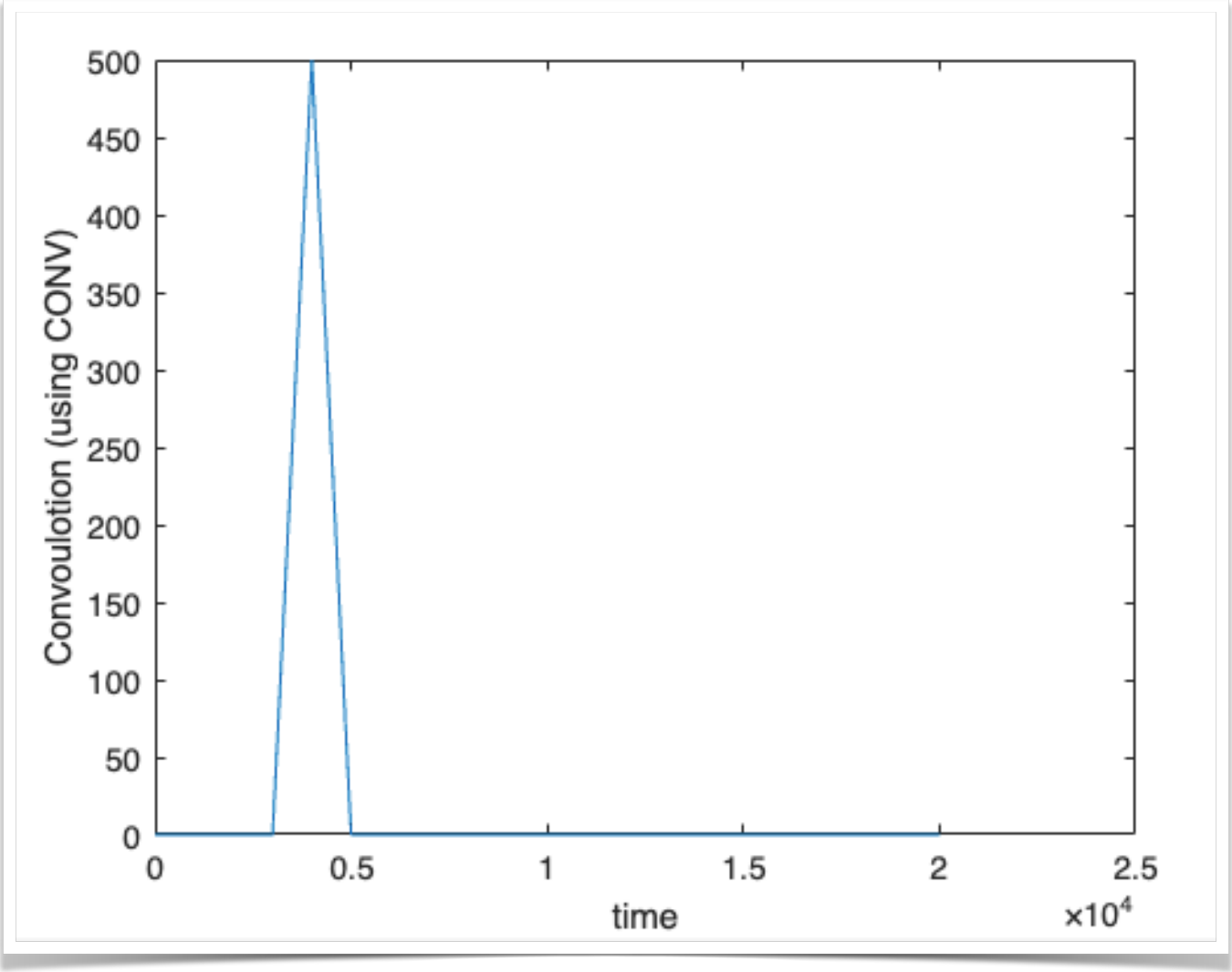
نکته ای که دارد این است که طول سیگنال sent در بازه ی [0 , 10000] است . و هم چنین طول بازه ی سیگنال .received هم در بازه ی بین [0 , 10000] است بنابراین طبق نکته ای که در کلاس گفته شد طول بازه ی خروجی کانولووشن باید مقدار [first1 + first2 , end1 + end2] باشد .

پس بنابراین طول سیگنال خروجی کانولووشن [0 , 20000] خواهد بود .

کد برنامه و خروجی در پایین ذکر شده اند :

```
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;


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;



% conv instruciton :
ro=conv(sent,recieved);

[MAXRO,positionOfMax]=max(ro);

finalTd=(positionOfMax - round(tau/ts) )*ts;
finalR=finalTd*speedOfLight/2 ;

timeFrame = size(ro);
plot(timeFrame(2),ro);
xlabel("time");
ylabel("Convoulotion (using CONV)");

disp(["Final R usign CONV is : " , int2str(finalR)])


```

```
>> Q2

    "Final R usign CONV is : "    "450"
```


Convoulotion (using CONV)
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
time
×10^4

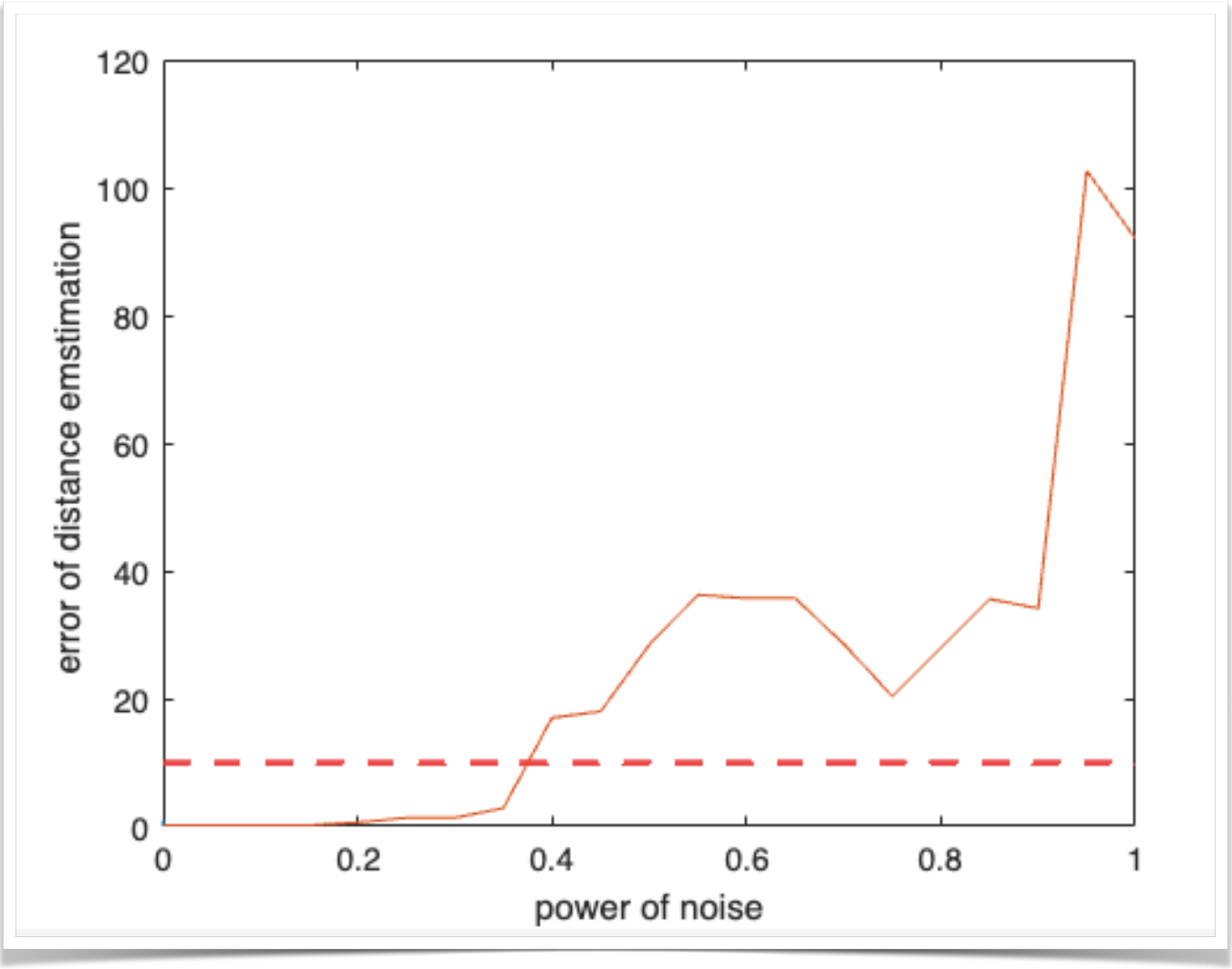
:Q3

این بخش باید مقداری نویز به سیگنال دریافتی اضافه کنیم و دقت آن تا آنجایی که با تلورانس ۱۰ متر به ما جواب صحیح بدهد را بسنجیم .

برای اینکه دقت برنامه بیشتر شود برای هر قدرت نویز ۱۰ با تست میکنیم و میانگین آن ۱۰ تا را به عنوان خطای آن قدرت نویز در نظر میگیریم .

```
recieved(round(td/ts) : round((td+tau)/ts)-1 ) = 1 * alpha;

powers = 0 : 0.05 : 1 ;
errorsOfPowers = [];

for power= 0 : 0.05 : 1
    allErrorsOfThisPower = [];
    for iteration=1:10

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


        distance = convolute(sent , recieved);

        realDistance = 450;

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


    end

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

plot(powers, errorsOfPowers);

% Add a vertical line at x = 10
yline(10, '--', 'LineWidth', 2 , 'Color', 'r');

hold off;
```

```
function [finalR] = convolute(sent,recieved)
    ts = 1e-9;  % Sampling time
    tau = 1e-6; % Time constant
    speedOfLight = 3e8;

    ro=conv(sent,recieved);

    [MAXRO,positionOfMax]=max(ro);

    finalTd=(positionOfMax - round(tau/ts) )*ts;
    finalR=finalTd*speedOfLight/2 ;

end
```



برای خطا طبق گزارش کار عدد ۱۰ را به عنوان خطای critical در نظر میگیریم .

Part 3

در این قسمت با استفاده صدای ضبط شده خود باید به پیاده سازی اعلام نوبت در بانک بپردازیم .

آماده سازی دیتاست:

در ابتدا باید اعداد زیر را ضبط کنیم :

1 , 2 , 3 , ... , 20

30 , 40 , 50 , 60 , 70 , 80 , 90

همچنین باید به ضبط صداهای " ُ " ، " شماره ی " ، " به بادجه ی " بپردازیم .

این فایل هارا تماما در یک دایرکتوری با نام voiceDataSet در کنار فایل اصلی برنامه آپلود میکنیم .

حالا باید عدد ورودی شماره ی مشتری را بررسی کنیم :

- اگر عدد بین ۱ تا ۲۰ باشد ، چون این اعداد تلفظ خاص خود را دارند باید همان ویس مخصوص خودشان را پخش کنیم .
- اگر عدد بیشتر از ۲۰ بود و یکان آن صفر بود کافی است که وویس رقم دهگان آن را پخش کنیم .
- اگر عدد بیشتر از ۲۰ بود اما یکان آن صفر نبود باید از بهم پیوستن وویس رقم دهگان و لفظ ذخیره شده ی "ُ" و همچنین وویس مربوط به رقم یکان صدا را ارسال کرد .

اما کار برای شماره ی بادجه راحت است چون بادجه فقط بین ۱ تا ۹ میباشد .

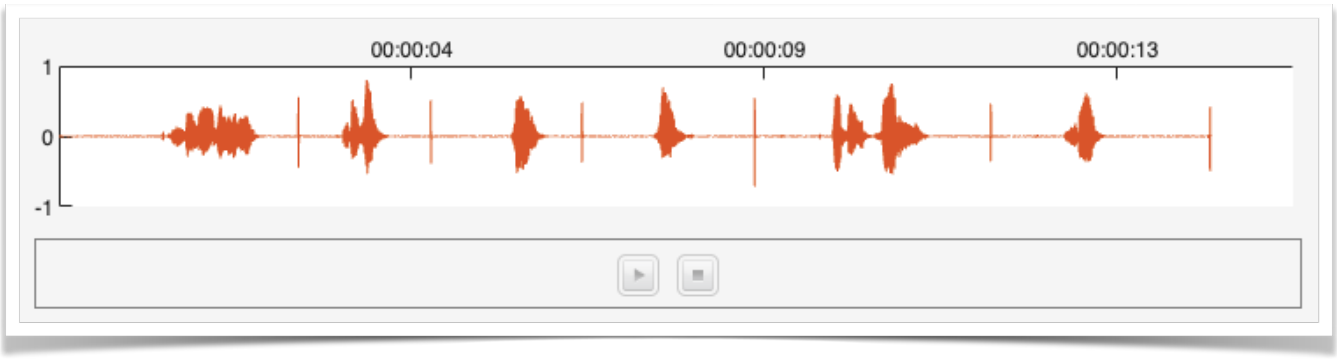
پیاده سازی برنامه:

حالا باید ویس های مربوط به لفظ "شماره ی " و عدد مورد نظر مشتری و لفظ " به بادجه ی " و همچنین شماره ی بادجه را به هم وصل کنیم و صدا را با Rate خود ویس ها تولید کنیم .

همچنین اگر عدد مربوط به مشتری خارج از بازه ی ۱ تا ۹۹ باشد و یا شماره ی بادجه خارج از بازه ی ۱ تا ۹ باشد برنامه با ارور به پایان میرسد .

```
if (budget < 1) || (budget > 9)
    disp("The budget is NOT valid :(");
    return;
end
if (number < 1) || (number > 99)
    disp("Your Input number is NOT valid :(");
    return;
end

directory = dir('VoiceDataSet');

files = {directory.name};
% Dar inja . and .. ro hazf mikonim
files = files(3:end);

voices = containers.Map;

for i = 1:length(files)
    addrr = fullfile('VoiceDataSet', files{i}) ;
    [audioData, sampleRate] = audioread(addrr);

     % bedoone .wav suffix
    voices(files{i}(1:end-4)) = audioData;
end

if (number < 21)
    concatenatedVoice = [voices('Shomareye'); voices(int2str(number)); voices('BeBadgeye'); voices(int2str(budget))];
end
if (mod(number, 10) == 0)
    concatenatedVoice = [voices('Shomareye'); voices(int2str(number)); voices('BeBadgeye'); voices(int2str(budget))];
end
if ~((number < 21) || (mod(number, 10) == 0))
    concatenatedVoice = [voices('Shomareye'); voices(int2str(number - mod(number, 10))); voices('va'); voices(int2str(mod(number, 10)));voices('BeBad
end
```

همان طور هم که انتظار داریم ویس تولید شده از اجتماع ویس های مورد نیاز تولید میگردد :



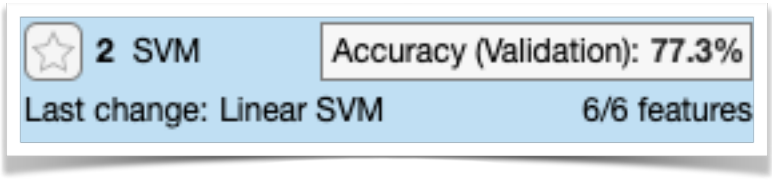
Part 4

در این قسمت با استفاده از dataset افراد تشخیص دیابت را با ML پیاده سازی میکنیم .

:Q1

با استفاده از دیتاست training یک مدل SVM پیاده سازی میکنیم که با استفاده از تمام feature های موجود در دیتاست Label هر یک را حدس بزند .

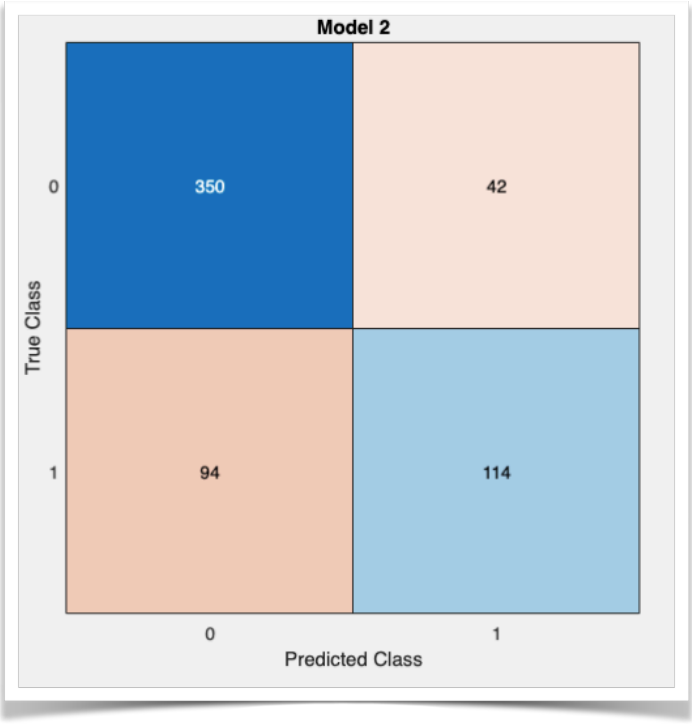
حال به گزارش دقت مدل روی دیتاست train میپردازیم :



همچنین با استفاده از Confusion Matrix به تحلیل حدس های ماشین میپردازیم :

دیابت نداشته و مدل هم تشخیص داده که دیابت ندارد

دیابت نداشته اما مدل تشخیص داده که دیابت دارد



دیابت داشته اما مدل تشخیص داده که دیابت ندارد

دیابت داشته و مدل تشخیص داده که دیابت دارد

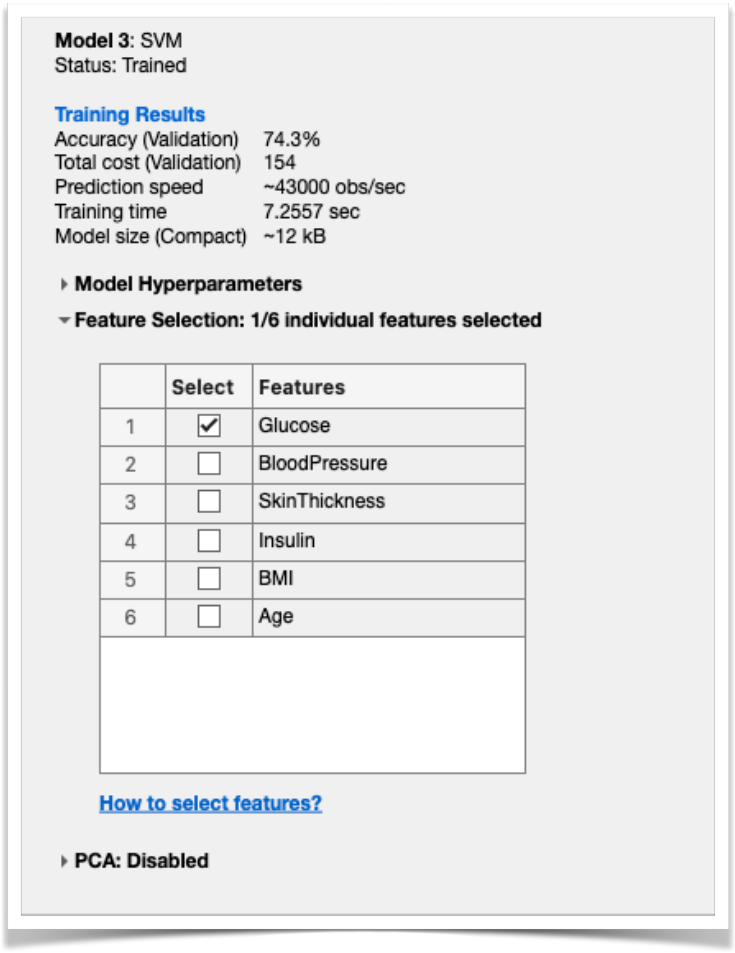
Model 2: SVM
Status: Trained

Training Results

Accuracy (Validation)	77.3%
Total cost (Validation)	136
Prediction speed	~8400 obs/sec
Training time	4.2221 sec
Model size (Compact)	~25 kB

	Select	Features
1	☑	Glucose
2	☑	BloodPressure
3	☑	SkinThickness
4	☑	Insulin
5	☑	BMI
6	☑	Age

- ساخت مدل بر پایه ی ویژگی Clucose



Model 3: SVM
Status: Trained

Training Results

Accuracy (Validation)	74.3%
Total cost (Validation)	154
Prediction speed	~43000 obs/sec
Training time	7.2557 sec
Model size (Compact)	~12 kB

▸ **Model Hyperparameters**

▾ **Feature Selection: 1/6 individual features selected**

	Select	Features
1	☑	Glucose
2	☐	BloodPressure
3	☐	SkinThickness
4	☐	Insulin
5	☐	BMI
6	☐	Age

How to select features?

▸ **PCA: Disabled**

- ساخت مدل بر پایه ی ویژگی BloodPressure

Model 4: SVM
Status: Trained

Training Results

Accuracy (Validation)	65.3%
Total cost (Validation)	208
Prediction speed	~82000 obs/sec
Training time	1.5189 sec
Model size (Compact)	~14 kB

Model Hyperparameters

Feature Selection: 1/6 individual features selected

	Select	Features
1	☐	Glucose
2	☑	BloodPressure
3	☐	SkinThickness
4	☐	Insulin
5	☐	BMI
6	☐	Age

How to select features?

PCA: Disabled

- ساخت مدل بر پایه ی ویژگی skin Thinckness

Model 5: SVM
Status: Trained

Training Results

Accuracy (Validation)	65.3%
Total cost (Validation)	208
Prediction speed	~61000 obs/sec
Training time	1.5738 sec
Model size (Compact)	~14 kB

▸ **Model Hyperparameters**

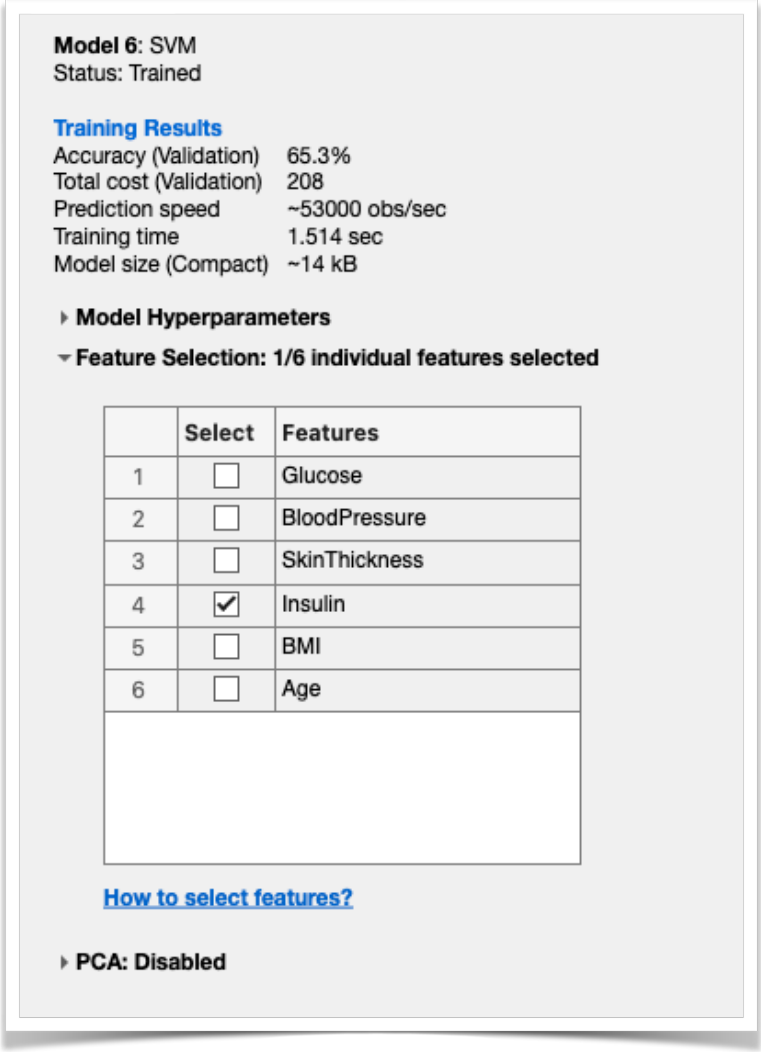
▾ **Feature Selection: 1/6 individual features selected**

	Select	Features
1	☐	Glucose
2	☐	BloodPressure
3	☑	SkinThickness
4	☐	Insulin
5	☐	BMI
6	☐	Age

How to select features?

▸ **PCA: Disabled**

- ساخت مدل بر پایه ی ویژگی Insulin



Model 6: SVM
Status: Trained

Training Results

Accuracy (Validation)	65.3%
Total cost (Validation)	208
Prediction speed	~53000 obs/sec
Training time	1.514 sec
Model size (Compact)	~14 kB

▸ **Model Hyperparameters**

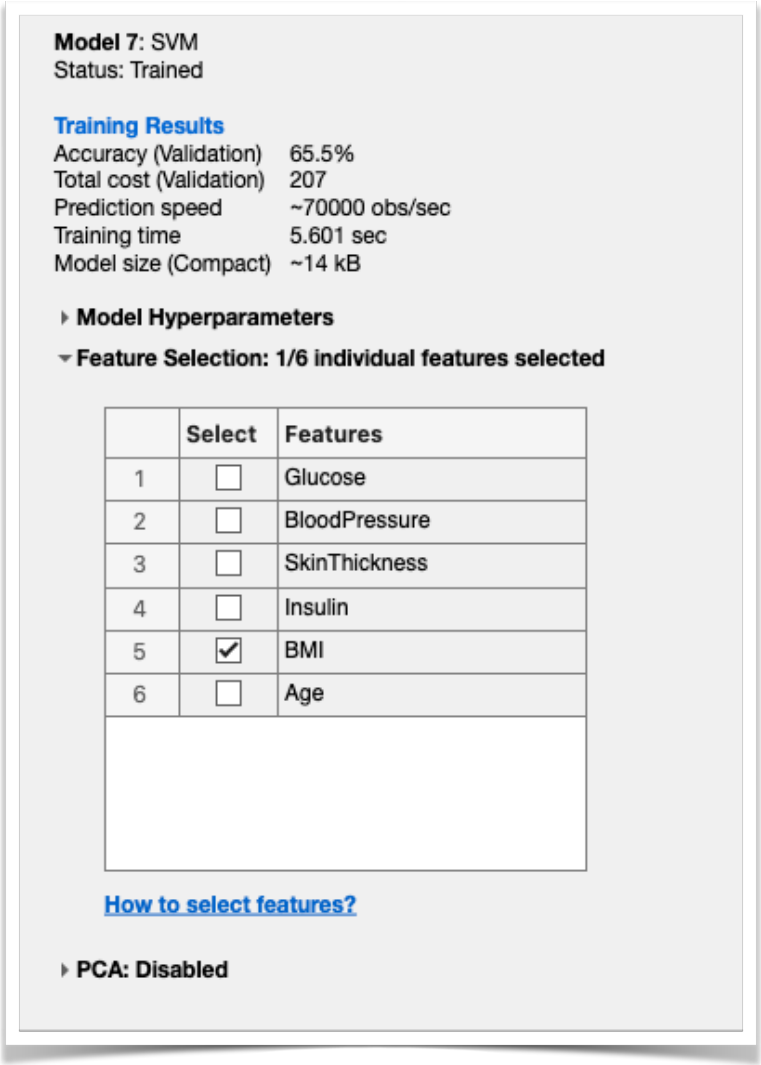
▾ **Feature Selection: 1/6 individual features selected**

	Select	Features
1	☐	Glucose
2	☐	BloodPressure
3	☐	SkinThickness
4	☑	Insulin
5	☐	BMI
6	☐	Age

How to select features?

▸ **PCA: Disabled**

- ساخت مدل بر پایه ی ویژگی BMI



Model 7: SVM
Status: Trained

Training Results

Accuracy (Validation)	65.5%
Total cost (Validation)	207
Prediction speed	~70000 obs/sec
Training time	5.601 sec
Model size (Compact)	~14 kB

Model Hyperparameters

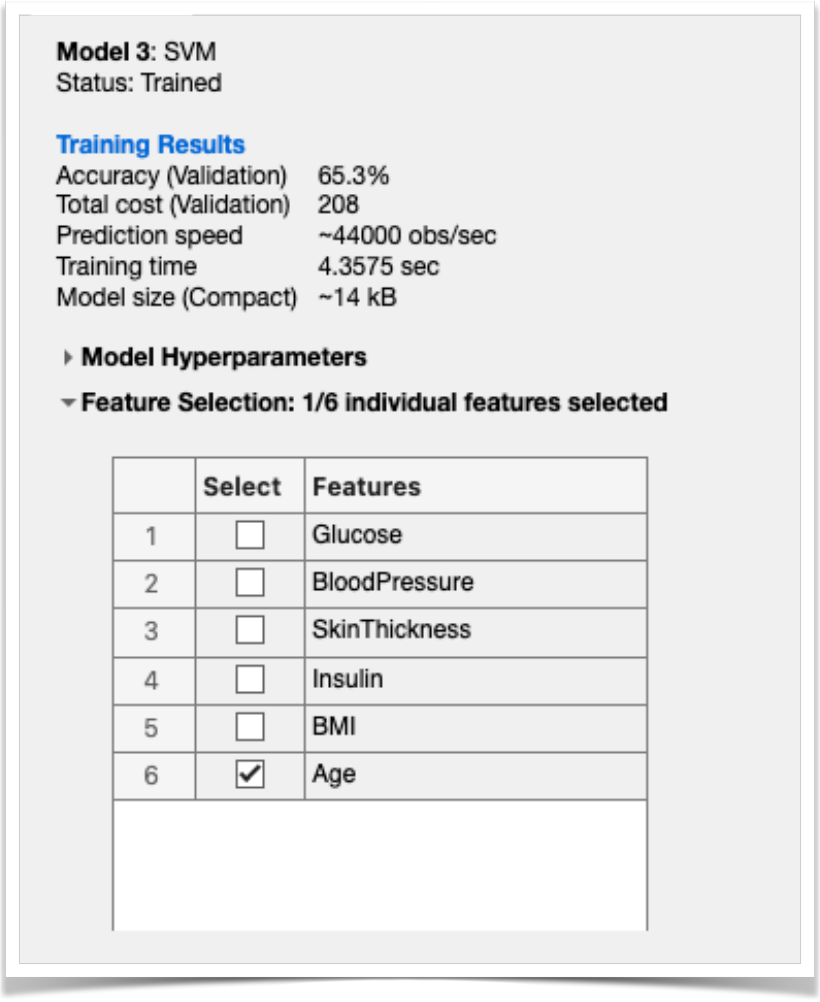
Feature Selection: 1/6 individual features selected

	Select	Features
1	☐	Glucose
2	☐	BloodPressure
3	☐	SkinThickness
4	☐	Insulin
5	☑	BMI
6	☐	Age

How to select features?

PCA: Disabled

- ساخت مدل بر پایه ی ویژگی Age



Model 3: SVM
Status: Trained

Training Results

Accuracy (Validation)	65.3%
Total cost (Validation)	208
Prediction speed	~44000 obs/sec
Training time	4.3575 sec
Model size (Compact)	~14 kB

▸ **Model Hyperparameters**

▾ **Feature Selection: 1/6 individual features selected**

	Select	Features
1	☐	Glucose
2	☐	BloodPressure
3	☐	SkinThickness
4	☐	Insulin
5	☐	BMI
6	☑	Age

بر اساس مدل های بالا بیشترین دقت مدل برای مدلی است که بر مبنای Glucose ساخته شده باشد . که این دقت مقدار 74.3% را داراست . البته مدلی که بر مبنای تمامی ویژگی ها باشد با دقت 77% در ابتدا قرار دارد .

:Q3

در پارت از برنامه یک کد نوشته ام که چک میکند که چقدر از Label های حدس زده شده با واقعیت تشابه داد که این مقدار روی همان train data ما تست شده است . همان طور که در صورت پروژه آمده این مقدار تا حد دقیقی با مقدار دقت در زمان ساخت مدل تطابق دارد .

```
>> Q3

sizeOfDF =

   600

Accuracy training.csv : %d77.5
>>
```

```
1
2   a = TrainedModel.predictFcn(diabetes_training);
3   sizeOfDF = height(diabetes_training)
4
5
6   accept = 0;
7
8   for i = 1:sizeOfDF
9       if (a(i,1) == diabetes_training{i, 'label'})
10          accept = accept + 1;
11      end
12  end
13
14  % accuracy
15  accuracy = accept / sizeOfDF * 100;
16
17  disp(['Accuracy training.csv : %d', num2str(accuracy)]);
18
19
```

:Q4

دقت در فاز validation را با استفاده از کد زیر بدست میاوریم :

```
1
2   b= TrainedModel.predictFcn(diabetes_validation);
3
4   accept = 0;
5   sizeOfDF = height(diabetes_validation)
6
7   for i = 1:sizeOfDF
8       if (a(i,1) == diabetes_validation{i, 'label'})
9           accept = accept + 1;
10      end
11  end
12  accuracy = accept / sizeOfDF * 100;
13
14  disp(['Accuracy validation.csv: ', num2str(accuracy)]);
15
```

```
>> Q4

sizeOfDF =

   100

Accuracy validation.csv: 50
>>
```