Signals and Systems

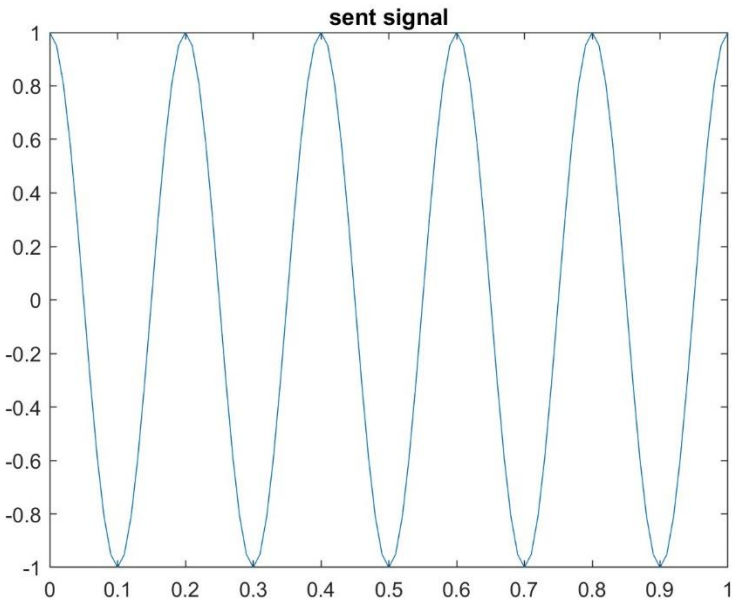
CA6

Rouja Aghajani – 810101380

تمرین 1-1)

```
fc = 5;
tstart = 0;
tend = 1;
fs = 100;
t = tstart:(1/fs):tend;
signal = cos(2*pi*fc*t);

plot(t,signal)
title('sent signal')
```

تصویر سیگنال رسم شده:

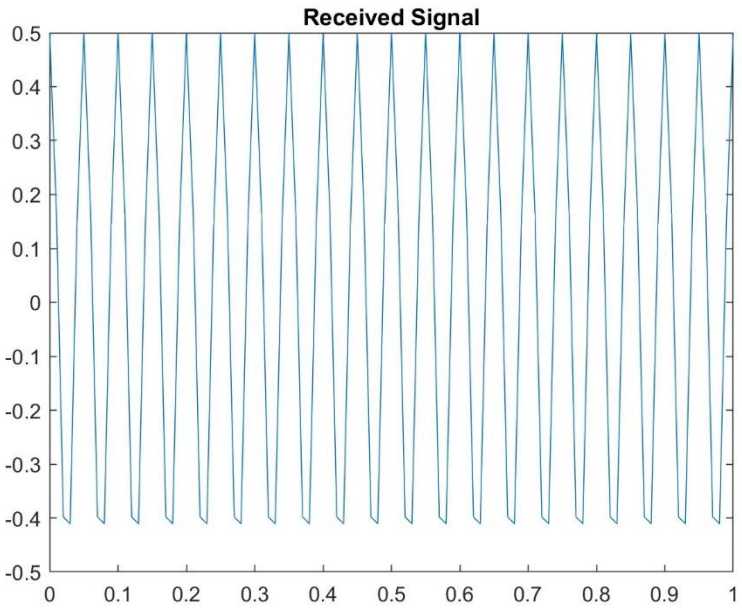


با داشتن فرکانس، مدت زمان شروع و پایان، و استفاده از تابع کسینوس سیگنال دست آمد.

تمرین 1-2)

```
fc = 5;
tstart = 0;
tend = 1;
fs = 100;
t = tstart:(1/fs):tend;

V=180*10/36;
R=250*1000;
beta=0.3;
alpha=0.5;
fd=beta*V;
c=3*1e9;
ro=2/c;
td=ro*R;

receivedSignal=alpha*cos(2*pi*(fc+fd)*(t-td));
plot(t,receivedSignal)
title('Received Signal')
```

شکل موج رسم شده:



با استفاده از فرمول های داده شده در صورت سوال و داشتن تمام پارامتر ها، آن ها را در تابع کسینوس قرار داده و هر پارامتر را طبق فرمول داده شده جایگذاری کرده تا سیگنال نهایی تولید شود.

تمرین 1-3)

```
clc
clear
fc = 5;
tstart = 0;
tend = 1;
fs = 100;
t = tstart:(1/fs):tend-1/fs;

V=180*10/36;
R=250*1000;
beta=0.3;
alpha=0.5;
fd=beta*V;
c=3*1e9;
ro=2/c;
td=ro*R;
receivedSignal=alpha*cos(2*pi*(fc+fd)*(t-td));

N=length(receivedSignal);
m=fftshift(fft(receivedSignal));
theta = angle(m);
m=abs(m);
[~,col]=find(m==max(m));
pha=abs(theta(col(2)));
fnew=(col(2)-N/2-1)*fs/N;
fdfound=fnew-fc;
foundV=fdfound/beta*36/10;
foundR=pha/(2*pi*(fdfound+fc)*ro)*0.001;
fprintf("found distance: %f km\n", foundR);
fprintf("found velocity: %f m/s\n" , foundV);
```

خروجی:

```
Command Window
  found distance: 250.000000 km
  found velocity: 180.000000 m/s
```

توضیح:

جهت بدست آوردن سرعت و فاصله متحرک مد نظر، می توان از فرکانس و فاز سیگنال دریافتی بهره گرفت. مطابق فرمول داده شده فرکانس برابر با مجموع fd و fc است و با داشتن fc، fd و در نتیجه سرعت جسم قابل محاسبه است.

برای یافتن خواسته سوال از تبدیل فوریه استفاده می کنیم تا تاخیر زمانی را بدست بیاوریم (ابتدا در حوزه فرکانس بدست می آید) و سپس سرعت و فاصله را محاسبه کنیم. بدین منظور ابتدا سیگنال را به فضای فوریه برده، سپس فرکانس و فاز غالب را بدست می آوریم که همان فرکانس و فاز جدید گفته شده در راهنمایی سوال می باشد. در نهایت با استفاده از دو فرمول زیر سرعت و فاصله محاسبه می شود:

$$f_d = f_{new} - f_c \ , \qquad V = \frac{f_d}{\beta} \times 3.6 \ , \qquad R = \frac{\varphi_{new} \times c}{2 \times \pi \times (f_c + f_d) \times 2}$$

که مقادیر محاسبه شده نیز بالا نمایش داده شده اند.

تمرین 1-4)

```
1   fc = 5;
2   tstart = 0;
3   tend = 1;
4   fs = 100;
5   t = tstart:(1/fs):tend-1/fs;
6
7   V=180*10/36;
8   R=250*1000;
9   beta=0.3;
10  alpha=0.5;
11  fd=beta*V;
12  c=3*1e9;
13  ro=2/c;
14  td=ro*R;
15  receivedSignal=alpha*cos(2*pi*(fc+fd)*(t-td));
16  sigma = 0.01;
17
18  receivedSignal=receivedSignal+sigma*randn(size(receivedSignal));
19  plot(receivedSignal);
20  title("Received Signal With Added Noise")
21  N=length(receivedSignal);
22  m=fftshift(fft(receivedSignal));
23  theta = angle(m);
24  m=abs(m);
25  [~,col]=find(m==max(m));
26  pha=abs(theta(col(2)));
27  fnew=(col(2)-N/2-1)*fs/N;
28  fdfound=fnew-fc;
29  foundV_withnoise=fdfound/beta*36/10;
30  foundR_withnoise=pha/(2*pi*(fdfound+fc)*ro)*0.001;
31  fprintf("Sigma of the noise: %f\n" , sigma);
32  fprintf("found distance: %f km\n", foundR_withnoise);
33  fprintf("found velocity: %f m/s\n" , foundV_withnoise);
```

و مقدار خروجی بدست آمده:

```
Command Window
found distance: 250.000000 km
found velocity: 180.000000 m/s
>> p1_4
Sigma of the noise: 0.001000
found distance: 250.771548 km
found velocity: 180.000000 m/s
>> p1_4
Sigma of the noise: 0.010000
found distance: 249.853357 km
found velocity: 180.000000 m/s
>> p1_4
Sigma of the noise: 0.100000
found distance: 237.545399 km
found velocity: 180.000000 m/s
>> p1_4
Sigma of the noise: 1.000000
found distance: 20205.624100 km
found velocity: 372.000000 m/s
>> p1_4
Sigma of the noise: 1.400000
found distance: 10757.754593 km
found velocity: 240.000000 m/s
```

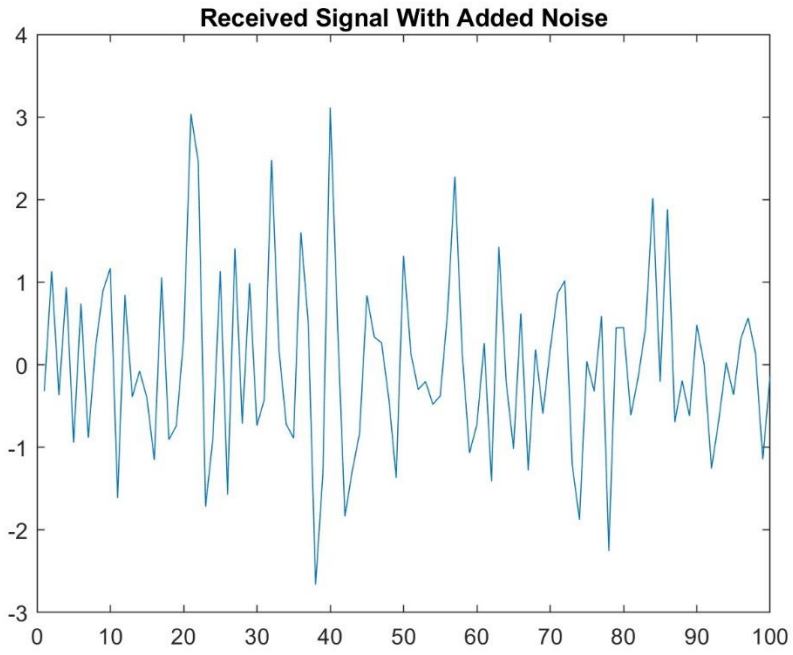
در بالا خروجی سوال قبل جهت مقایسه ذکر شده است.

از دستور randn جهت اضافه کردن نویز استفاده شده است.

مشاهده می شود با اعمال نویز با سیگمای 0.001 هم فاصله و هم سرعت صحیح بدست می آیند، ولی با افزایش سیگما فاصله نادرست بدست آمده ولی سرعت همچنان درست است. مشاهده می شود سرعت در سیگمای 1 دچار خطا می شود که در مقایسه با فاصله، می تواند نویز بیشتری را تحمل کند.

علت این پدیده آن است که برای داشتن سرعت دقیق یا موقعیت مکانی دقیق، باید سیگنال گسترده در فرکانس و کوچک در حوزه زمان یا برعکس داشته باشیم. در واقع بین تخمین دقیق سرعت و موقعیت مکانی یک trade-off وجود دارد که بستگی به نیاز ما، می توانیم محاسبه یکی را به شرط دقت کمتر در دیگری، دقیق بدست بیاوریم.

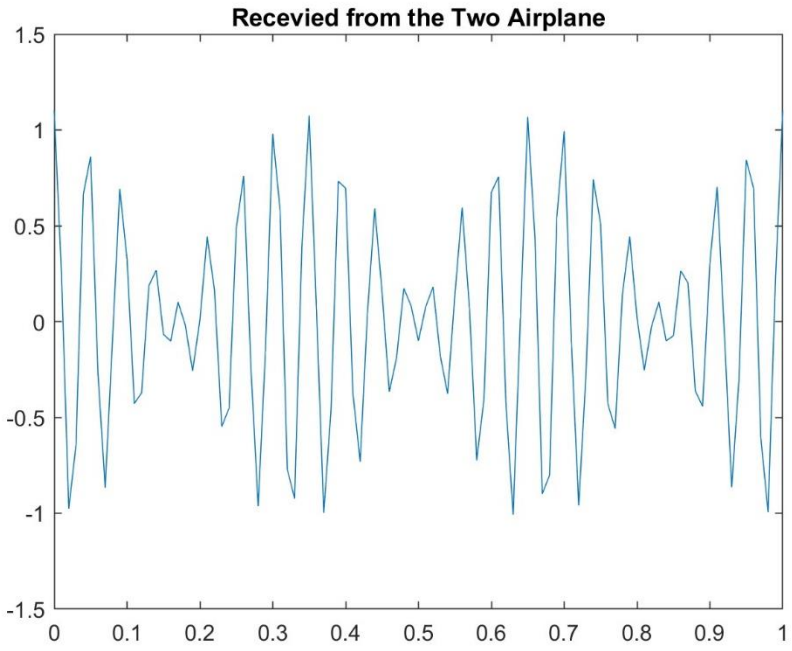
سیگنال با نویز با سیگمای 1 که در این صورت در محاسبه سرعت و فاصله خطا وجود دارد:



تمرين 1-5)

```
fc = 5;
tstart = 0;
tend = 1;
fs = 100;
t = tstart:(1/fs):tend;
V=180*10/36;
R=250*1000;
beta=0.3;
alpha=0.5;
fd=beta*V;
c=3*1e9;
ro=2/c;

R1=250*1000;
R2=200*1000;

V1=180*10/36;
V2=216*10/36;

alpha1=0.5;
alpha2=0.6;

fd1=round(beta*V1);
fd2=round(beta*V2);

td1=ro*R1;
td2=ro*R2;

receivedSignal1=alpha1*cos(2*pi*(fc+fd1)*(t-td1));
receivedSignal2=alpha2*cos(2*pi*(fc+fd2)*(t-td2));
recieved=receivedSignal1+receivedSignal2;
figure()
plot(t,recieved)
title('Recevied from the Two Airplane')
```

تصویر سیگنال دریافتی از دو هواپیما:



سیگنال های خروجی دو هواپیما را مانند بخش قبل بدست آوردیم، آنگاه با هم جمع کردیم.

تمرین 1-6)

```
clc
clear
fc = 5;
tstart = 0;
tend = 1;
fs = 100;
t = tstart:(1/fs):tend-1/fs;
V=180*10/36;
R=250*1000;
beta=0.3;
alpha=0.5;
fd=beta*V;
c=3*1e9;
ro=2/c;

R1=250*1000;
R2=200*1000;

V1=180*10/36;
V2=216*10/36;

alpha1=0.5;
alpha2=0.6;

fd1=round(beta*V1);
fd2=round(beta*V2);

td1=ro*R1;
td2=ro*R2;

receivedSignal1=alpha1*cos(2*pi*(fc+fd1)*(t-td1));
receivedSignal2=alpha2*cos(2*pi*(fc+fd2)*(t-td2));
recieved=receivedSignal1+receivedSignal2;

N=length(recieved);
m2=fftshift(fft(recieved));
theta2 = angle(m2);
m2=abs(m2);
[mx,col]=maxk(m2,4);
fnew1=(col(2)-N/2-1)*fs/N;
pha1=abs(theta2(col(2)));
fnew2=(col(4)-N/2-1)*fs/N;
pha2=abs(theta2(col(4)));
fdfound1=fnew1-fc;
fdfound2=fnew2-fc;

Vfound1=fdfound1/beta*36/10;
Vfound2=fdfound2/beta*36/10;
Rfound1=pha1/(2*pi*(fdfound1+fc)*ro)*0.001;
Rfound2=pha2/(2*pi*(fdfound2+fc)*ro)*0.001;

fprintf("V1 = %f, R1 = %f\n", Vfound1, Rfound1);
fprintf("V2 = %f , R2 = %f" , Vfound2, Rfound2);
```

مقادیر بدست آمده:

```
Command Window
V1 = 216.000000, R1 = 200.000000
V2 = 180.000000 , R2 = 250.000000>>
```

توضیح:

جهت بدست آوردن سرعت و فاصله دو جسم، ابتدا سیگنال حاصل ر ا به حوزه فوریه برده تا فرکانس های غالب آن بدست آیند. سپس دو فرکانس غالب را که دلیل وجودشان دو جسم مد نظر است بدست می آیند. سپس با استفاده از این فرکانس ها و فاز متناظرشان، فرکانس داپلر و تاخیر زمانی هر جسم بدست آمده و در نهایت از این دو جهت محاسبه سرعت و فاصله اجسام استفاده می شود.

با دستور max دو پیک بدست می آیند که در واقع فرکانس های آنها پیدا می شوند. سپس با این فرکانس ها فاز و طبق فرمول هایی که پیش تر بیان شد سرعت و فاصله بدست می آیند. همچنین چون fs دقت را کم می کند، از دستور round فرکانس را گرد کرده تا مسافت محاسبه شده زیاد و با خطای زیاد نباشد.

تمرین 1-7)

در این حالت امکان تفکیک دو سیگنال در حوزه فوریه را نخواهیم داشت. در پروژه قبل مشاهده شد که چون اختلاف دو سیگنال تک تن کمتر از 1Hz بود، دو سیگنال در حوزه فوریه قبال تمییز دادن نبودند. در صورتی که اختلاف دو سرعت 12Km/h یا 3.33 m/s باشد، اختلاف فرکانس ها به یک متر رسیده و پارامترها قابل استخراج درست خواهند بود.

$$\Delta V = \frac{\Delta f}{\beta} \ , suppose\ \beta = 0.3 \ \rightarrow \ \Delta v = \frac{1}{0.3} = 3.33$$

تمرین 1-8)

در این صورت فرکانس های fd1 و fd2 برای دو جسم متفاوت ولی td یکسان خواهد بود. به علت تفاوت در فرکانس های داپلر، با بردن سیگنال در حوزه فوریه می توانیم فرکانس های غالب را استخراج کرده و از طریق آنها مانند بخش های قبل سرعت و فاصله اجسام را بیابیم.

در نتیجه بله، با داشتن دو جسم با فاصله برابر ولی سرعت های متفاوت می توان سرعت و فاصله آن ها را استخراج کرد.

تمرین 1-9)

سیگنال در حوزه فوریه به تعداد اجسام متحرک پیک دارد. پس ابتدا سیگنال دریافتی را به حوزه فوریه برده و پیک های آن را بدست می آوریم (بدین صورت که هر نقطه در مقادیر مثبت را با نقطه قبل آن مقایسه کرده و اگر این مقدار بزرگ باشد نقطه یافت شده پیک است).

سپس برای هر نقطه بدست آمده فرکانس و فاز مربوطه را محاسبه میکنیم. Fd از تفاوت fc و فرکانس هر قله بدست می آید. با داشتن β سرعت جسم و با داشتن فاز تاخیر زمانی و در نهایت فاصله هر جسم محاسبه خواهد شد.

تمرین 2-1)

```
tstart=0;
tend=0.5;
T=0.5;

fs=8000;
tau=25e-3;

t=tstart:1/fs:tend-1/fs;


silence=zeros(1, round(tau*fs));



Notes=["B" , "A#" , "A" , "G#" , "G" , "F#" , "F" , "E" , "D#" , "D" , "C#" , "C"];
NoteFrequencies=[987.77 , 932.33 , 880 ,830.61 ,783.99,739.99,698.46 , 659.25 , 622.25 , 587.33 , 554.37 , 523.25];

inputSongNotes = {'D','D','G','F#','D','D','E','E','D','F#','D','E','D','E','F#','E','D','E','E','D','F#','D','E','D','E','D','F#','E','D','E','D','F#','E','D','D','E','F#
inputSongNoteDurations = [T/2, T/2, T, T, T, T/2, T/2, T/2, T/2, T/2, T/2, T, T, T, T, T/2, T/2, T/2, T/2, T/2, T/2, T, T, T/2, T/2, T, T, T, T/2, T/2, T, T, T/2, T/2,


song=[];
for i=1:length(inputSongNotes)

    [~,num]=find(Notes==inputSongNotes{1,i});
    t = tstart:1/fs:(inputSongNoteDurations(i)-1/fs);
    y=sin(2*pi*NoteFrequencies(num)*t);
    song=[song y silence];

end
sound(song)
audiowrite('p2_1.wav',song,fs);
```

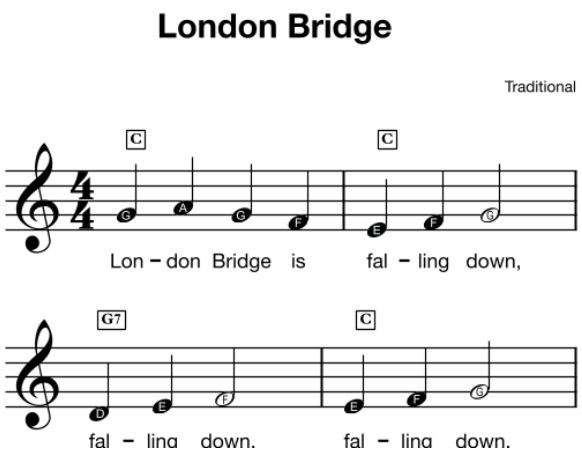
ابتدا از نت های ممکن و فرکانس هایشان دو ماتریس تشکیل میدهیم که هر نت و فرکانس متناظر با آن در یک ایندکس هستند. سپس نت های موسیقی ورودی و زمان متناظر با هر نوت را نیز در دو ماتریس قرار می دهیم. با داشتن ماتریس نت های ورودی، هر نت را با نت های ممکن مقایسه کرده، نت متناظر را میابیم (مثلا یافتن نت A در ماتریس حاوی تمام نت ها با داشتن نت A در نت های ورودی) و با داشتن ایندکس آن، فرکانس مدنظر را پیدا میکنیم. آنگاه با داشتن فرکانس، بدست آوردن مدت زمان آن نت و قرار دادن این دو مورد یعنی f و t در یک تابع سینوسی صدای مد نظر را تولید میکنیم. در نهایت بعد هر نت یک وقفه با طول معین در صورت سوال قرار می دهیم.

تمرین 2-2)

```
tstart=0;
tend=0.5;
T=0.5;

fs=8000;
tau=25e-3;

t=tstart:1/fs:tend-1/fs;


silence=zeros(1, round(tau*fs));



Notes=["B" , "A#" , "A" , "G#" , "G" , "F#" , "F" , "E" , "D#" , "D" , "C#" , "C"];
NoteFrequencies=[987.77 , 932.33 , 880 ,830.61 ,783.99,739.99,698.46 , 659.25 , 622.25 , 587.33 , 554.37 , 523.25];

inputSongNotes = {'G','A','G','F','E','F','G','D','E','F','E','F','G'};
inputSongNoteDurations = [T,T,T,T,T,T,2*T,T,T,2*T,T,T,2*T];


song=[];
for i=1:length(inputSongNotes)

    [~,num]=find(Notes==inputSongNotes{1,i});
    t = tstart:1/fs:(inputSongNoteDurations(i)-1/fs);
    y=sin(2*pi*NoteFrequencies(num)*t);

    song=[song y silence];

end
sound(song)
audiowrite('mysong.wav',song,fs);
```

در اینجا نیز مطابق بالا نت ها و مدت زمان هر نوت را وارد کرده و موسیقی متناظر با آن را در نهایت بدست می آوریم.

نت ورودی:



تمرین 2-3)

```
function [decoded] = musicDCD(music_address)
    fs=8000;
    [amp,~]=audioread(music_address);



    Notes=["B" , "A#" , "A" , "G#" , "G" , "F#" , "F" , "E" , "D#" , "D" , "C#" , "C"];
    NoteFrequencies=[987.77 , 932.33 , 880 ,830.61 ,783.99,739.99,698.46 , 659.25 , 622.25 , 587.33 , 554.37 , 523.25];




    notes={};
    en=length(amp);
    n=1;
    while en~=0
        for i=2:en
            if amp(i)==0 && amp(i+1)==0
                break
            end
        end
        y=amp(1:i-1);
        notes(n)={y};
        n=n+1;
        amp=amp(i+200:en);
        en=length(amp);

    end

    for i=1:length(notes)

        y=cell2mat(notes(i));
        N=length(y);
        m=abs(fftshift(fft(y)));
        [row,~]=find(m==max(m));
        frq=(row(2)-N/2-1)*fs/N;
        for n=1:length(NoteFrequencies)
            trsh=2;
            if abs(NoteFrequencies(n)-frq)<trsh
                decoded(1,i)=Notes(n);
                decoded(2,i)=N/4000;
            end
        end
    end

end
```

```
function musicRebuilder(given_song)
    tstart=0;
    T=0.5;

    fs=8000;
    tau=25e-3;

    silence=zeros(1, round(tau*fs));

    Notes=["B" , "A#" , "A" , "G#" , "G" , "F#" , "F" , "E" , "D#" , "D" , "C#" , "C"];
    NoteFrequencies=[987.77 , 932.33 , 880 ,830.61 ,783.99,739.99,698.46 , 659.25 , 622.25 , 587.33 , 554.37 , 523.25];

    song=[];
    for i=1:length(given_song)

        [~,num]=find(Notes==given_song{1,i});
        t = tstart:1/fs:(str2double(given_song{2,i})*T-1/fs);
        y=sin(2*pi*NoteFrequencies(num)*t);

        song=[song y silence];

    end
    sound(song)
    audiowrite('p2_3.wav',song,fs);

end
```

```
music = "p2_1.wav";
decoded = musicDCD(music);
musicRebuilder(decoded);
```

با استفاده از یک لوپ while، تا زمانی که amplitude سیگنال به صفر نرسیده آن را بخش بخش میکنیم تا نوت ها را پیدا کنیم. برای یافتن نوت ها نیزباید ارایه ای با طول بیشتر از 1 و شامل صفر پیدا کنیم. (میتوانیم در سیگنال سینوسی صفر داشته باشیم ولی دو نقطه کنار هم صفر نمی شوندن و حداکثر یک صفر داریم). با داشتن مدت زمان سکوت بین هر نوت، با یافتن نوت اول بقیه نوت ها به ترتیب بدست می آیند.

همچنین یک threshold نیز جهت تشخیص نوت ها تعریف شده است. هربار در حوزه فوریه پیک ها پیدا شده و در واقع فرکانس ها شناسایی می شوند، مدت زمان هریک نیز ذخیره می شوند. سپس با مقایسه فرکانس نوت یافت شده با فرکانس هایی که برای هر نوت داریم، نوت مربوط شناسایی و مدت زمان آن نیز در سطر اول و و دوم سلولی ذخیره می شود. در نهایت این سلول به عنوان خروجی تحویل داده می شود.

با استفاده از این سلول به عنوان ورودی تابع دوم، اکنون هم نوت ها و هم مدت زمان هر یک را داریم (مدت زمان به string ذخیره شده که آن را به عدد با دستور str2double تبدیل می کنیم). ادامه کار مانند بخش های قبل است، نوت مربوط پیدا شده، فرکانس آن یافت شده و مدت زمانی آن نیز بدست می آید، با قرار دادن این مقادیر در تابع سینوس و قرار دادن سکوت بعد هر نوت موسیقی نهایی تشکیل می شود.

نت های خروجی نیز مشابه نت های داده شده در تمرین 2-1 م باشد که خواسته سوال است:

```
>> p2_3
  Columns 1 through 13

    "D"     "D"     "G"     "F♯"    "D"     "D"     "E"     "E"     "D"     "F♯"    "D"     "E"     "D"
    "0.5"   "0.5"   "1"     "1"     "1"     "0.5"   "0.5"   "0.5"   "0.5"   "0.5"   "0.5"   "1"     "1"

  Columns 14 through 26

    "E"     "F♯"    "E"     "D"     "E"     "E"     "D"     "F♯"    "D"     "E"     "D"     "E"     "D"
    "1"     "1"     "1"     "0.5"   "0.5"   "0.5"   "0.5"   "0.5"   "0.5"   "1"     "1"     "0.5"   "0.5"

  Columns 27 through 39

    "F♯"    "E"     "D"     "E"     "D"     "F♯"    "E"     "D"     "D"     "E"     "F♯"    "E"     "F♯"
    "1"     "1"     "1"     "0.5"   "0.5"   "1"     "1"     "0.5"   "0.5"   "1"     "0.5"   "0.5"   "1"

  Columns 40 through 44

    "F♯"    "E"     "F♯"    "F♯"    "D"
    "0.5"   "0.5"   "1"     "1"     "1"
```