به نام خدا

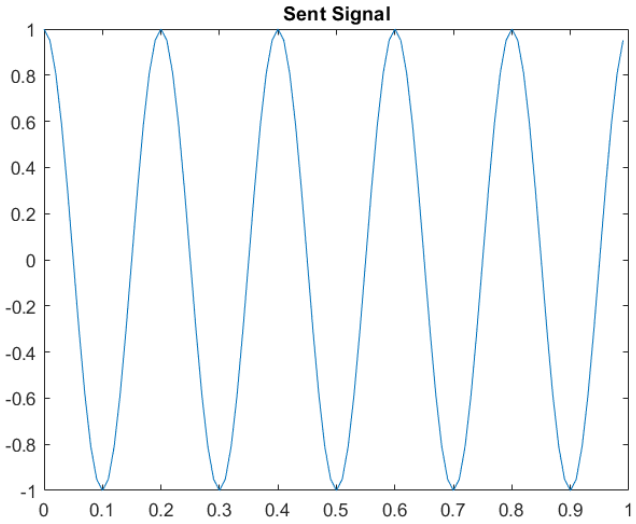
سید محمد حسین مظهری ۸۱۰۱۰۱۵۲۰ پروژه ۶ درس سیگنال و سیستم

پارسا دقیق 810101419

بخش اول :

تمرین 1-1)

سیگنال ارسالی:

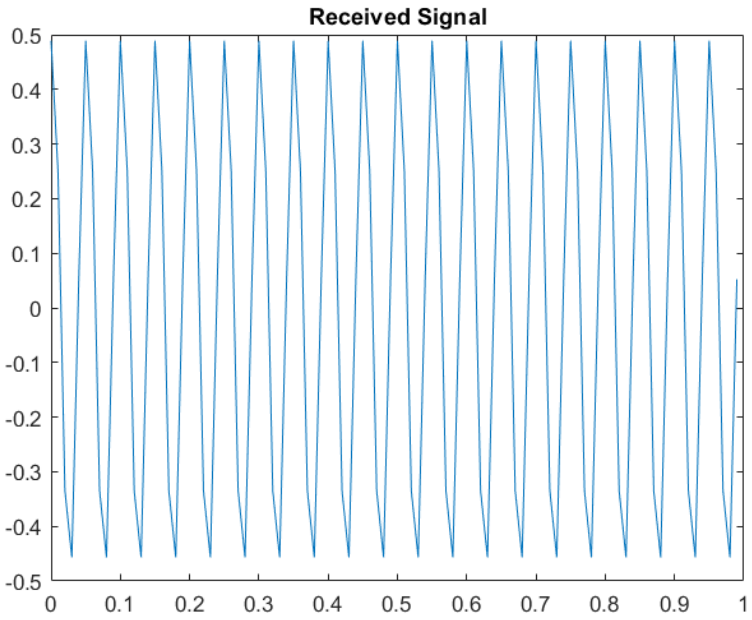


اسکریپت:

```
clc;
clear;

% Creating sent signal
fc = 5;
fs = 100;
tstart = 0;
tend = 1;

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

x = cos(2*pi*fc*t);
figure
plot(t,x)
title('Sent Signal')
```

تمرین ۲-۱)

سیگنال دریافتی :



اسکریپت :

```
clc;
clear;

% Creating Received signal
fc = 5;
fs = 100;
tstart = 0;
tend = 1;
v = (180*1000)/3600;
R = 250*1000;
B = 0.3;
a = 0.5;
C = 3e8;

ts = 1/fs;
t = tstart:ts:tend-ts;
fd = B*v;
p = 2/C;
td = p*R;

y = a*cos(2*pi*(fc+fd)*(t-td));
figure
plot(t,y)
title('Received Signal')

save('received1.mat',"y");
```

تمرین ۳-۱)

نتیجه :

```
R:250
v:180
```

اسکریپت:

```
clc;
clear;

fc = 5;
fs = 100;
B = 0.3;
C = 3e8;
p = 2/C;

% Analyzing Recieved signal
y = load("received1.mat");
y = struct2cell(y);
y = cell2mat(y);
fourier = fftshift(fft(y));
phase = angle(fourier);
fabs = abs(fourier);
[row,col] = find(fabs==max(fabs));
phase = abs(phase(col(2)));
fnew = (col(2)-(length(y)/2)-1)*(fs/(length(y)));

fd = fnew-fc;
td = phase/(fnew*2*pi);
v = (fd/B)*(3600/1000);
R = (td/p)/1000;
```

با توجه به راهنمایی سوال ، ابتدا سیگنال دریافتی را به حوزه فوریه می بریم. سپس فرکانس غالب سیگنال را استخراج میکنیم . (با استفاده از دستور max مقدار قله را در حوزه فوریه پیدا کرده و با استفاده از دستور find، index متناسب با آن را استخراج میکنیم و با استفاده از این index، فرکانس غالب را محاسبه میکنیم .

$$Frequency = (index - N/2 - 1)(f_s / N)$$

با استفاده از index استخراج شده و دستور angle ، فاز سیگنال را محاسبه میکنیم . حال با استفاده از فرکانس و فاز ، f_d و t_d را به محاسبه میکنیم و سپس با آنها Vو R را به دست می آوریم .

تمرین ۴-۱)

در این تمرین ، طی چندین مرحله به سیگنال نویز گوسی اضافه کرده و پارامتر ها را محاسبه کردیم.

پارامتر فاصله حساسیت بسیار زیادی به نویز دارد ، به شکلی که در همان ابتدا و با σ= ۰,۰۱ ، این پارامتر دچار خطا میشود .

فاصله :

```
sigma:0.01
v:180
R:247.808
```

اما پارامتر سرعت حساسیت کمتری به نویز نشان می دهد، به صورتی که تا σ= ۰,۳۱ درست عمل میکند.

سرعت :

```
sigma:0.31
v:180
R:13.5867
sigma:0.32
v:372
R:63.8144
```

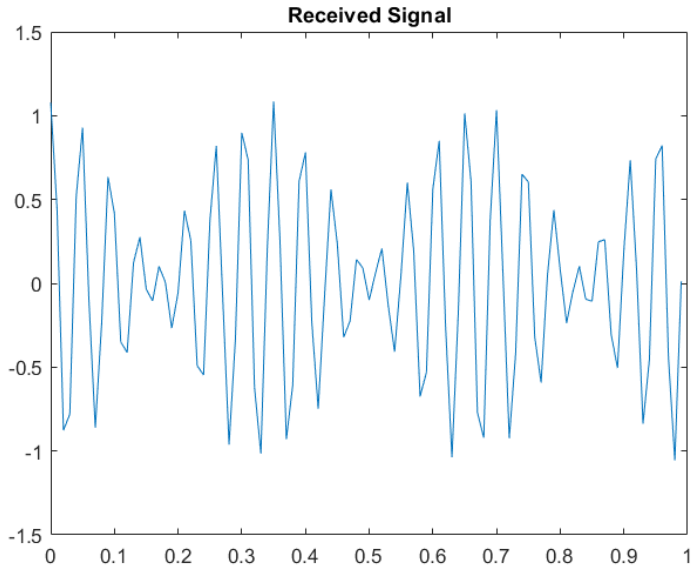
اسکریپت:

```
clc;
clear;

fc = 5;
fs = 100;
B = 0.3;
C = 3e8;
p = 2/C;

y = load("received1.mat");
y = struct2cell(y);
y = cell2mat(y);

% Adding noise to the received signal and analyzing
for i=1:100
    sigma = 0.01*i;
    noise = sigma*randn(1,length(y));
    y = y + noise;


    fourier = fftshift(fft(y));
    phase = angle(fourier);
    fabs = abs(fourier);
    [row,col] = find(fabs==max(fabs));
    phase = abs(phase(col(2)));
    fnew = (col(2)-(length(y)/2)-1)*(fs/(length(y)));

    fd = fnew-fc;
    td = phase/(fnew*2*pi);
    v = (fd/B)*(3600/1000);
    R = (td/p)/1000;
end
```

تمرین ۵-۱) رابطه سیگنال دریافتی:

$$y = 0.5\cos(2\pi(5 + 15)(t - 0.0017)) + 0.6\cos(2\pi(5 + 18)(t - 0.0013))$$

نتیجه:



اسکریپت:

```
clc;
clear;

% Creating Received signal
fc = 5;
fs = 100;
tstart = 0;
tend = 1;
B = 0.3;
C = 3e8;
p = 2/C;
ts = 1/fs;
t = tstart:ts:tend-ts;

v1 = (180*1000)/3600;
R1 = 250*1000;
a1 = 0.5;

v2 = (216*1000)/3600;
R2 = 200*1000;
a2 = 0.6;

fd1 = B*v1;
td1 = p*R1;
y1 = a1*cos(2*pi*(fc+fd1)*(t-td1));

fd2 = B*v2;
td2 = p*R2;
y2 = a2*cos(2*pi*(fc+fd2)*(t-td2));
y_double = y1+y2;

figure
plot(t,y_double)
title('Received Signal')
save('received2.mat',"y_double");
```

تمرین ۶-۱)

روش محاسبه فاصله و سرعت دو جسم دقیقا ماندن تمرین ۱-۳ است . اما تفاوتی که میان این دو تمرین وجود دارد در پیدا کردن قله هاست . در تمرین قبل چون تنها یک جسم داشتیم ، دو قله به مقادیر بیشینه یکسان داشتیم . (یک قله در فرکانس مثبت و دیگری در فرکانس قرینه فرکانس اول) اما از چون در این تمرین ۲ جسم داریم ، ۴ قله خواهیم داشت . پس به جای استفاده از دستور find , max از دستور maxk استفاده میکنیم. عملکرد این دستور بدین صورت است که فقط نقطه حداکثر را مانند max پیدا نمیکند بلکه از نقطه بیشینه شروع کرده و به تعدادی که از آن بخواهیم ، index مقادیر حوزه فوریه سیگنال را به صورت نزولی به ما میدهد. حال که index های مد نظر پیدا شدند ، بقیه محاسبات را دقیقا مانند تمرین قبل انجام میدهیم ، با این تفاوت که در این تمرین این محاسبات باید دو مرتبه انجام شوند.

اسکریپت:

```
clc;
clear;

fc = 5;
fs = 100;
B = 0.3;
C = 3e8;
p = 2/C;

% Analyzing Recieved signal
y = load("received2.mat");
y = struct2cell(y);
y = cell2mat(y);
fourier = fftshift(fft(y));
phase = angle(fourier);
fabs = abs(fourier);
[row,col] = maxk(fabs,4);

phase1 = abs(phase(col(4)));
fnew1 = (col(4)-(length(y)/2)-1)*(fs/(length(y)));
fd1 = fnew1-fc;
td1 = phase1/(fnew1*2*pi);
v1 = (fd1/B)*(3600/1000);
R1 = (td1/p)/1000;

phase2 = abs(phase(col(2)));
fnew2 = (col(2)-(length(y)/2)-1)*(fs/(length(y)));
fd2 = fnew2-fc;
td2 = phase2/(fnew2*2*pi);
v2 = (fd2/B)*(3600/1000);
R2 = (td2/p)/1000;
```

نتیجه :

```
v1:180
R1:250
v2:216
R2:200
```

تمرین ۷-۱) خیر ، در این صورت قادر نخواهیم بود فاصله ی آن ها را استخراج بکنیم. اگر سرعت دو جسم داده شده برابر باشند ، فرکانس های دو سیگنال دریافتی از دو جسم هم برابر خواهند بود ، بنابرین در حوزه فوریه قله ها قابل تمایز از یکدیگر نخواهند بود و تنها یک قله خواهیم داشت . ما با index این قله ها میتوانیم سرعت این دو جسم را استخراج کنیم (زیرا با این index میتوانیم فرکانس را محاسبه کنیم و فرکانس ها برابر هستند)، اما در مورد فاز ، این index به ما مجموع فاز ها را میدهد و ما راهی برای تمایز دادن فاز ها از یکدیگر نداریم . پس قادر به استخراج فاصله دو جسم نیستیم .

حداقل اختلاف سرعت دو جسم باید به گونه ای باشد که اختلاف فرکانس ها در خوزه فوریه از رزولوشن فرکانسی ما بیشتر ابشد تا بتوانیم پارامتر های دو جسم را به درستی تخمین بزنیم :

$$\delta\_f = 1/T = 1\ \text{Hz}$$

$\Delta f\_new > 1 \rightarrow \Delta f\_d > 1 \rightarrow \Delta V > 3.33\ (m/s) \rightarrow \Delta V > 12\ (km/h)$

بنابراین حداقل اختلاف سرعت دو جسم باید 12 (km/h) باشد تا بتوانیم پارامترهای دو جسم را درست تخمین بزنیم.

تمرین ۸-۱)

بله ، در این صورت قادر خواهیم بود سرعت ها و فاصله ی آنها را استخراج کنیم . در این تمرین بر خلاف تمرین قبل ، فرکانس سیگنال های دریافتی از دو جسم متفاوت هستند . بنابر این در حوزه فوریه قله ها جدا از یکدیگر هستند و هرکدام index متفاوتی دارند . به همین دلیل به راحتی و با روش توضیح داده شده در تمرین های قبل میتوانیم پارامتر های دو جسم را استخراج کنیم .

تمرین ۹-۱)

برای اینکه تعداد اجسام را بفهمیم، باید سیگنال دریافتی را به حوزه فوریه ببریم . حال باید تعداد قله ها را در حوزه فوریه پیدا کنیم . نصف تعداد قله ها در حوزه فوریه ، تعداد اجسام است . (از آنجایی که هر فرکانس دو قله ایجاد میکند ، یک قله در فرکانس مثبت و یک قله در فرکانس منفی ، به همین دلیل نصف تعداد قله ها تعداد اجسام است .) با استفاده از index این قله ها هم میتوانیم پارامتر های اجسام را استخراج کنیم .

بخش دوم :

تمرین ۲-۱)

•تنظیمات اولیه و تعریف فرکانس نت‌ها:

- کد با پاک کردن متغیرها و صفحه نمایش شروع می‌شود و سپس فرکانس نت‌های مختلف مانندA ، B، Cو غیره را تعریف می‌کند.
- فرکانس نمونه‌برداری به ۸۰۰۰ هرتز تنظیم می‌شود و گام زمانی براساس آن محاسبه می‌شود.

•ایجاد بردارهای زمانی و وقفه‌ها:

- مدت زمان نت‌ها (۰,۵ ثانیه و ۰,۲۵ ثانیه) و بردارهای زمانی مربوط به آنها ایجاد می‌شود.
- یک بردار از صفرها برای وقفه بین نت‌ها تعریف می‌شود.

•ساخت سیگنال صوتی:

- با استفاده از ترکیب امواج سینوسی برای هر نت و وقفه‌های بین آنها، یک سیگنال صوتی تولید می‌شود.
- سیگنال شامل نت‌های مختلفی مانندD ، G، #Fو غیره است که به ترتیب و با وقفه‌های مشخص در آرایه‌ی `signal` ذخیره می‌شود.
- در نهایت سیگنال تولید شده پخش می‌شود.

```
A=880;
B=987.77;
C=523.25;
D=587.33;
E=659.25;
F=698.46;
G=783.99;
A_sharp=932.33;
C_sharp=554.37;
D_sharp=622.25;
F_sharp=739.99;
G_sharp=830.61;
freq=[A,B,C,D,E,F,G,A_sharp,C_sharp,D_sharp,F_sharp,G_sharp];
ts=1/8000;
T1=0.5;
t1=0:ts:T1-ts;
T2=0.25;
t2=0:ts:T2-ts;
pause_zeros=zeros(1,200);
```

```
signal=[];
signal=[signal,sin(2*pi*D*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*D*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*G*t1)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*F_sharp*t1)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*D*t1)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*D*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*E*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*E*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*D*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*F_sharp*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*D*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*E*t1)];
signal=[signal,pause_zeros];
```

```
signal=[signal,sin(2*pi*E*t1)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*D*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*D*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*E*t1)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*F_sharp*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*E*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*F_sharp*t1)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*F_sharp*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*E*t2)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*F_sharp*t1)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*F_sharp*t1)];
signal=[signal,pause_zeros];
signal=[signal,sin(2*pi*D*t1)];
signal=[signal,pause_zeros];
sound(signal);
```

تمرین۲-۲)

این بخش کد به منظور تولید و پخش یک سیگنال صوتی برای آهنگ "Twinkle, Twinkle, Little Star" استفاده می‌شود.

تنظیمات اولیه:

- کد با پاک کردن متغیرها و صفحه نمایش شروع می‌شود و فرکانس نمونه‌برداری به ۸۰۰۰ هرتز تنظیم می‌شود.
- فرکانس نت‌های مختلف(C4 ، D4، E4، F4، G4، A4، B4، C5) تعریف می‌شود.

2. **ایجاد بردارهای زمانی و وقفه‌ها:**
   - مدت زمان نت‌ها به ۰,۵ و ۰,۲۵ ثانیه تنظیم می‌شود و بردارهای زمانی برای این مدت زمان‌ها ایجاد می‌شود.
   - یک بردار از صفرها برای ایجاد وقفه بین نت‌ها تعریف می‌شود.
3. **ساخت سیگنال صوتی:**
   - با استفاده از ترکیب امواج سینوسی برای هر نت و وقفه‌های بین آنها، یک سیگنال صوتی تولید می‌شود.
   - این سیگنال شامل نت‌های مختلفی از آهنگ "Twinkle, Twinkle, Little Star" است.
4. **پخش و ذخیره صدا :**
   - در نهایت سیگنال تولید شده با استفاده از تابع `sound`پخش می‌شود و با استفاده از تابع `audiowrite` به عنوان یک فایل صوتی ذخیره می‌شود

```
Fs = 8000;
ts = 1/Fs;

C4 = 261.63;
D4 = 293.66;
E4 = 329.63;
F4 = 349.23;
G4 = 392.00;
A4 = 440.00;
B4 = 493.88;
C5 = 523.25;

T1 = 0.5;
T2 = 0.25;

t1 = 0:ts:T1-ts;
t2 = 0:ts:T2-ts;

paus = zeros(1, 200);

melody = [];
```

```
% Twinkle, Twinkle, Little Star
melody = [melody, sin(2*pi*C4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*C4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*G4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*G4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*A4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*A4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*G4*T2)]; melody = [melody, paus];

melody = [melody, sin(2*pi*F4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*F4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*E4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*E4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*D4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*D4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*C4*T2)]; melody = [melody, paus];

% Repeat the first part
melody = [melody, sin(2*pi*G4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*G4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*F4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*F4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*E4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*E4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*D4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*D4*t1)]; melody = [melody, paus];

melody = [melody, sin(2*pi*C4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*C4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*G4*t1)]; melody = [melody, paus];
```

```
% Repeat the first part
melody = [melody, sin(2*pi*G4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*G4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*F4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*F4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*E4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*E4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*D4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*D4*t1)]; melody = [melody, paus];

melody = [melody, sin(2*pi*C4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*C4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*G4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*G4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*A4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*A4*t1)]; melody = [melody, paus];
melody = [melody, sin(2*pi*G4*T2)]; melody = [melody, paus];

sound(melody, Fs);
audiowrite('mysong.wav', melody, Fs);
```

تمرین ۲-۳)

استخراج نت‌ها و مدت زمان آنها:

- بخش‌هایی از سیگنال که غیر صفر هستند به عنوان نت‌های موسیقی استخراج می‌شوند و در سلول‌ها ذخیره می‌شوند.
- با استفاده از FFT) تبدیل فوریه سریع)، فرکانس اصلی هر نت و مدت زمان آن محاسبه می‌شود.
- نت‌های استخراج شده و مدت زمان‌های آنها در آرایه‌ای به نام `final_notes` ذخیره می‌شوند و نمایش داده می‌شوند.

```
 counter=1;
 note_cells=cell(1,1000);
 start_index=1;
 for i=3:length(signal)
     if signal(i)~=0 && signal(i-1)==0
         start_index=i-1;
     end
     if signal(i)==0 && signal(i-1)~=0
         segment=(signal(start_index:i-1));
         note_cells(1,counter)={segment};
         counter=counter+1;
     end
 end
 note_cells = note_cells(~cellfun('isempty',note_cells));
 frequencies=zeros(1,length(note_cells));
 durations=zeros(1,length(note_cells));
for j=1:length(note_cells)
    segment=cell2mat(note_cells(1,j));
    fft_result=fftshift(fft(segment));
    len=length(segment);
    scaling=len/2000;
    if scaling==1
        step=4;
        durations(j)=0.25;
    elseif scaling==2
        step=2;
        durations(j)=0.5;
    end
    freq_vec=0:step:(4000-step);
    half_spectrum=fft_result((len/2+1):len);
    [~,max_index]=max(half_spectrum);
    frequencies(j)=freq_vec(max_index);
end
Index=zeros(1,length(note_cells));
final_notes=cell(2,length(note_cells));
```

```
for w=1:length(note_cells)
    Index(w)=find(min(abs(freq-frequencies(w)))==abs(freq-frequencies(w)));
    if Index(w)==1
        final_notes(1,w)={'A'};
    elseif Index(w)==2
        final_notes(1,w)={'B'};
    elseif Index(w)==3
        final_notes(1,w)={'C'};
    elseif Index(w)==4
        final_notes(1,w)={'D'};
    elseif Index(w)==5
        final_notes(1,w)={'E'};
    elseif Index(w)==6
        final_notes(1,w)={'F'};
    elseif Index(w)==7
        final_notes(1,w)={'G'};
    elseif Index(w)==8
        final_notes(1,w)={'A#'};
    elseif Index(w)==9
        final_notes(1,w)={'C#'};
    elseif Index(w)==10
        final_notes(1,w)={'D#'};
    elseif Index(w)==11
        final_notes(1,w)={'F#'};
    elseif Index(w)==12
        final_notes(1,w)={'G#'};
    end
end
for k=1:length(note_cells)
    final_notes(2,k)=num2cell(durations(k));
end
```

```
disp(final_notes);
```

```
Columns 1 through 9

  {'D'     }    {'D'     }    {'G'     }    {'F#'    }    {'D'     }    {'D'     }    {'E'     }    {'E'     }    {'D'     }
  {[0.2500]}    {[0.2500]}    {[0.5000]}    {[0.5000]}    {[0.5000]}    {[0.2500]}    {[0.2500]}    {[0.2500]}    {[0.2500]}

Columns 10 through 18

  {'F#'    }    {'D'     }    {'E'     }    {'D'     }    {'E'     }    {'F#'    }    {'E'     }    {'D'     }    {'E'     }
  {[0.2500]}    {[0.2500]}    {[0.5000]}    {[0.5000]}    {[0.5000]}    {[0.5000]}    {[0.5000]}    {[0.2500]}    {[0.2500]}

Columns 19 through 27

  {'E'     }    {'D'     }    {'F#'    }    {'D'     }    {'E'     }    {'D'     }    {'E'     }    {'D'     }    {'F#'    }
  {[0.2500]}    {[0.2500]}    {[0.2500]}    {[0.2500]}    {[0.5000]}    {[0.5000]}    {[0.2500]}    {[0.2500]}    {[0.5000]}

Columns 28 through 36

  {'E'     }    {'D'     }    {'E'     }    {'D'     }    {'F#'    }    {'E'     }    {'D'     }    {'D'     }    {'E'     }
  {[0.5000]}    {[0.5000]}    {[0.2500]}    {[0.2500]}    {[0.5000]}    {[0.5000]}    {[0.2500]}    {[0.2500]}    {[0.5000]}

Columns 37 through 44

  {'F#'    }    {'E'     }    {'F#'    }    {'F#'    }    {'E'     }    {'F#'    }    {'F#'    }    {'D'     }
  {[0.2500]}    {[0.2500]}    {[0.5000]}    {[0.2500]}    {[0.2500]}    {[0.5000]}    {[0.5000]}    {[0.5000]}
```