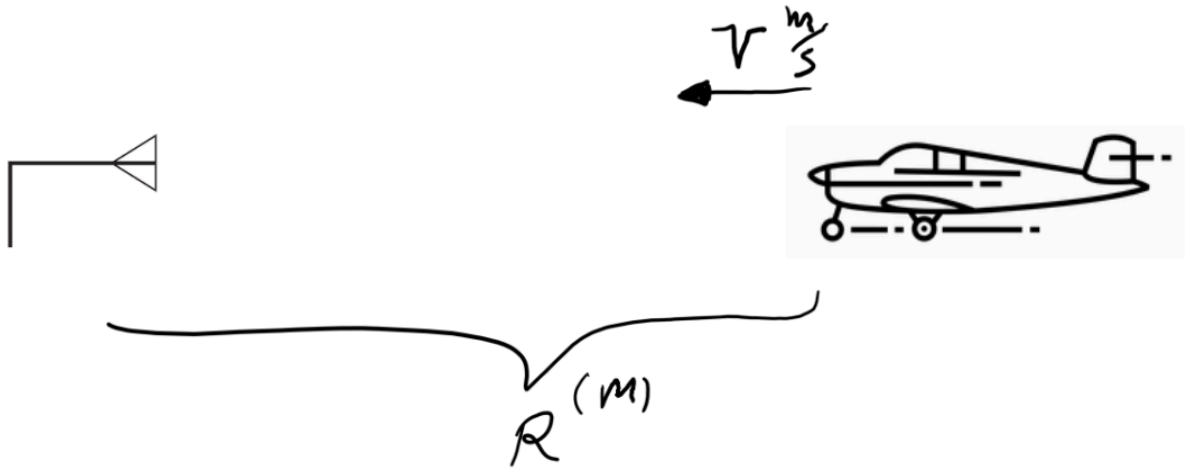
گزارش تمرین کامپیوتری پنجم سیگنال سیستم دکتر اخوان

پاییز 1403

محمدمهدی صمدی - 810101465

سپهر جمالی - 810101400

بخش اول)

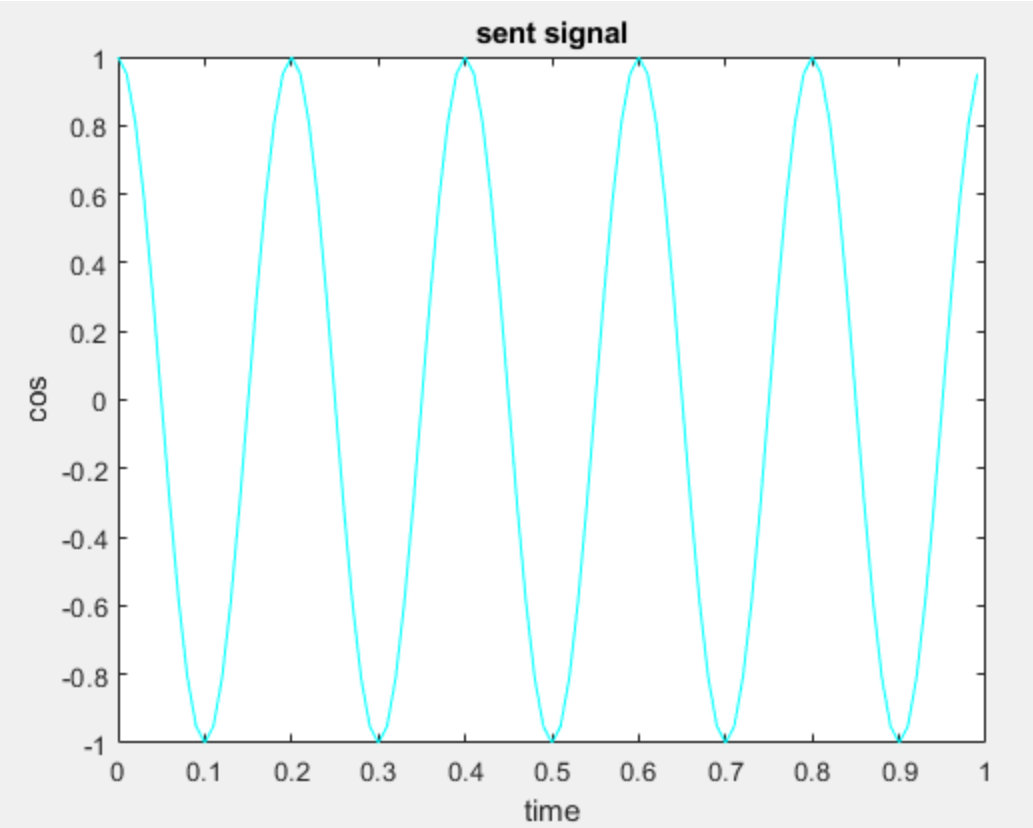


تمرین 1_1:

اسکریپت روبه‌رو سیگنال فرستنده را ساخته و پلات می‌کند. خروجی در تصویر صفحه بعد است.

```
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, 'Color', 'cyan');
xlabel('time'); ylabel('cos');
title('sent signal');
```



تمرین 1_2:

اسکریپت روبه‌رو سیگنال دریافتی را با فرض‌های ذکر شده در صورت سوال می‌سازد.

سرعت 180 km/h را به واحد m/s می‌بریم.

همچنین فاصلهر ا از واحد km به m می‌بریم.

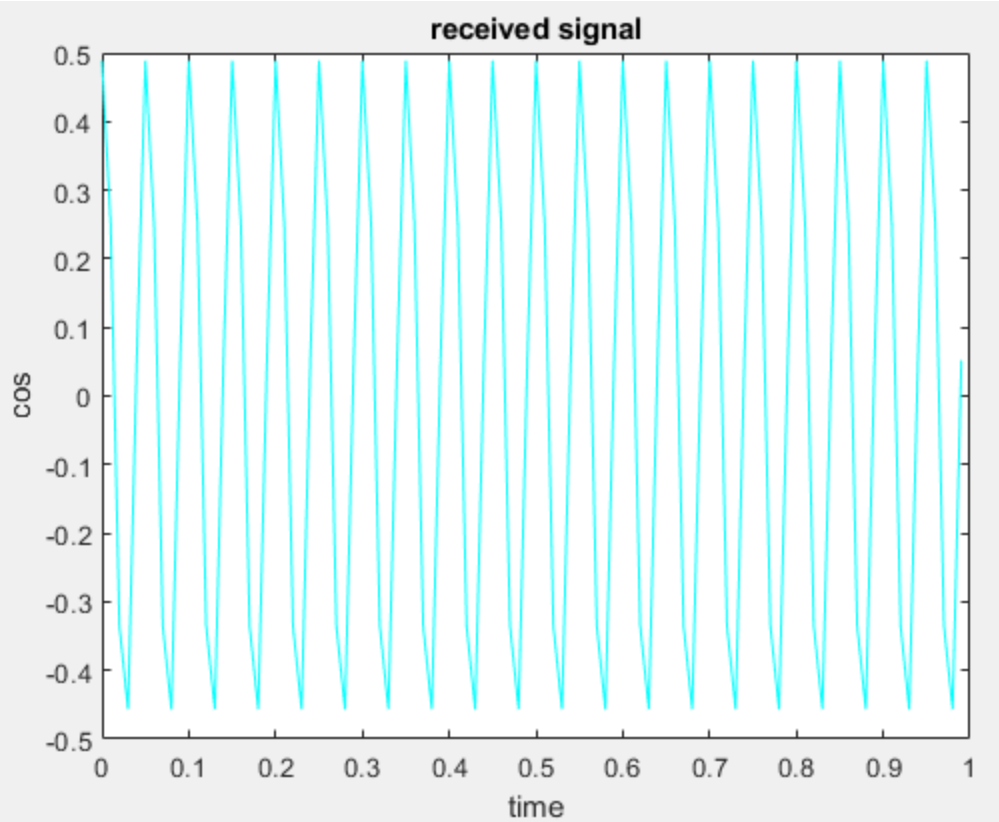
```
fc = 5; fs = 100;
tstart = 0; tend = 1;
v = 50; beta = 0.3; fd = beta*v;
C = 3e8; p = 2/C;
R = 250000; td = p*R;
ts = 1/fs; t = tstart:ts:tend-ts;
alpha = 0.5;
y = alpha*cos(2*pi*(fc+fd)*(t-td));
figure; plot(t,y, color='cyan');
xlabel('time');
ylabel('cos');
title('received signal');
```

خروجی این کد در صفحه بعد قرار دارد.

فرکانس در سیگنال فرستنده 5 بود پس در بازه زمانی [0, 1] شاهد کسینوس در 5 دوره تناوبش بودیم.

اما فرکانس سیگنال گیرنده 15+5 بود پس در همین بازه شاهد کسینوس در 20 دوره تناوبش هستیم.

همچنین می‌توان مشاهده کرد که دامنه سیگنال گیرنده نصف سیگنال فرستنده است.



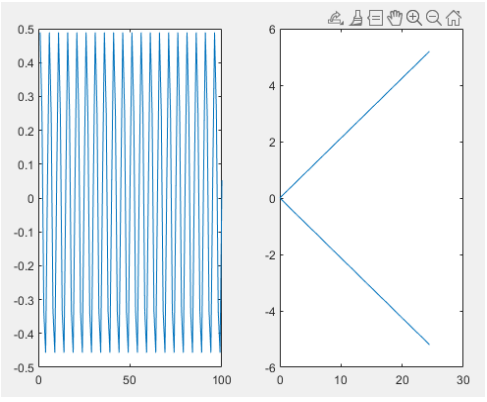
تمرین 1_3)

در تابع نوشته شده برای این بخش، ابتدا تبدیل فوریه سیگنال گرفته شده و با سپس با دستور fftshift تقارن حول صفر ایجاد می‌شود. بعد از آن فاز و اندازه تبدیل فوریه را پیدا کرده. می‌دانیم تبدیل فوریه کسینوس دو پیک دارد. با تابع find آن دو را پیدا می‌کنیم و از پیک دوم (فرکانس مثبت) برای ادامه کار استفاده می‌کنیم. فرکانس new را طبق راهنمایی سوال و از فرمول زیر محاسبه می‌کنیم:

$$f_{new} = (idx_{max} - \frac{N}{2} - 1) \times \frac{f_s}{N}$$

اندیس ماکزیمم از نصف سایز تبدیل فوریه کم شده و در ضریبی ضرب شده تا فرکانس متناظر پیدا شود. همانطور که می‌دانیم $f_{new} = f_d + f_c$. حال که f_c و f_{new} را داریم می‌توانیم f_d را پیدا کنیم و از روی آن سرعت را محاسبه کنیم. همچنین فاصله با کمک فاز سیگنال و طبق راهنمایی صورت سوال محاسبه شده است.

خروجی کد این بخش:



```
calculated distance is 250000
calculated speed is 50
```

کد تابع نیز در صفحه بعد آورده شدند.

```
function p13(y)
    fc = 5; fs = 100;
    B = 0.3; C = 3e8; p = 2/C;
    dtft = fftshift(fft(y));
    phase = angle(dtft);
    fabs = abs(dtft);
    [~,col] = find(fabs==max(fabs));
    pos_peak_idx = col(2);
    phase = abs(phase(pos_peak_idx));
    fnew = (pos_peak_idx-(length(dtft)/2)-1)*(fs/(length(dtft)));
    fd = fnew-fc; td = phase/(fnew*2*pi);
    v = (fd/B); R = td/p;

    figure;
    subplot(1, 2, 1); plot(y);
    subplot(1, 2, 2); plot(dtft);
    fprintf("calculated distance is %d\n", round(R, 1));
    fprintf("calculated speed is %d\n", round(v, 1));
```

تمرین 1_4)

در اسکریپت این بخش سیگما را با شروع از 0 و با گام‌های 0.01 تا 2 می‌بریم و در هر مرحله پیام‌هایی چاپ می‌شود که شامل سیگمای کنونی، فاصله محاسبه‌شده و سرعت محاسبه‌شده هستند.

```
y = cell2mat(struct2cell(load("received_p12.mat")));
for sigma=0:0.01:2
    fprintf("simulating with sigma %2f\n", sigma);
    p13(y + sigma*randn(1,length(y)));
    fprintf("\n");
end
```

با بررسی پیام‌ها به این نتیجه می‌رسیم که محاسبه فاصله با کوچک‌ترین نویز دچار خطا می‌شود در حالی که محاسبه سرعت تا سیگماهای حدود 1 همچنان درست است. البته به علت رندوم بودن ماهیت نویز در هر اجرا، اولین خطا متفاوت است اما به طور میانگین عددی در همین حدود می‌باشد. در نتیجه می‌توان گفت پارامتر فاصله حساسیت بیشتری به نویز دارد.

```
simulating with sigma 0.000000
calculated distance is 250000
calculated speed is 50

simulating with sigma 0.010000
calculated distance is 2.526192e+05
calculated speed is 50

simulating with sigma 0.020000
calculated distance is 2.506335e+05
calculated speed is 50
```

```
simulating with sigma 0.780000
calculated distance is 4.599828e+05
calculated speed is 50

simulating with sigma 0.790000
calculated distance is 4.494123e+06
calculated speed is 3.670000e+01

simulating with sigma 0.800000
calculated distance is 1.769240e+04
calculated speed is 50

simulating with sigma 0.810000
calculated distance is 1.453710e+04
calculated speed is 50
```

تمرین 1_5)

رابطه سیگنال خروجی بدین صورت است:

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

اسکریپت این بخش. بسیار مشابه اسکریپت بخش 1_2 است و صرفا دو سیگنال ساخته و با هم جمع شده‌اند.

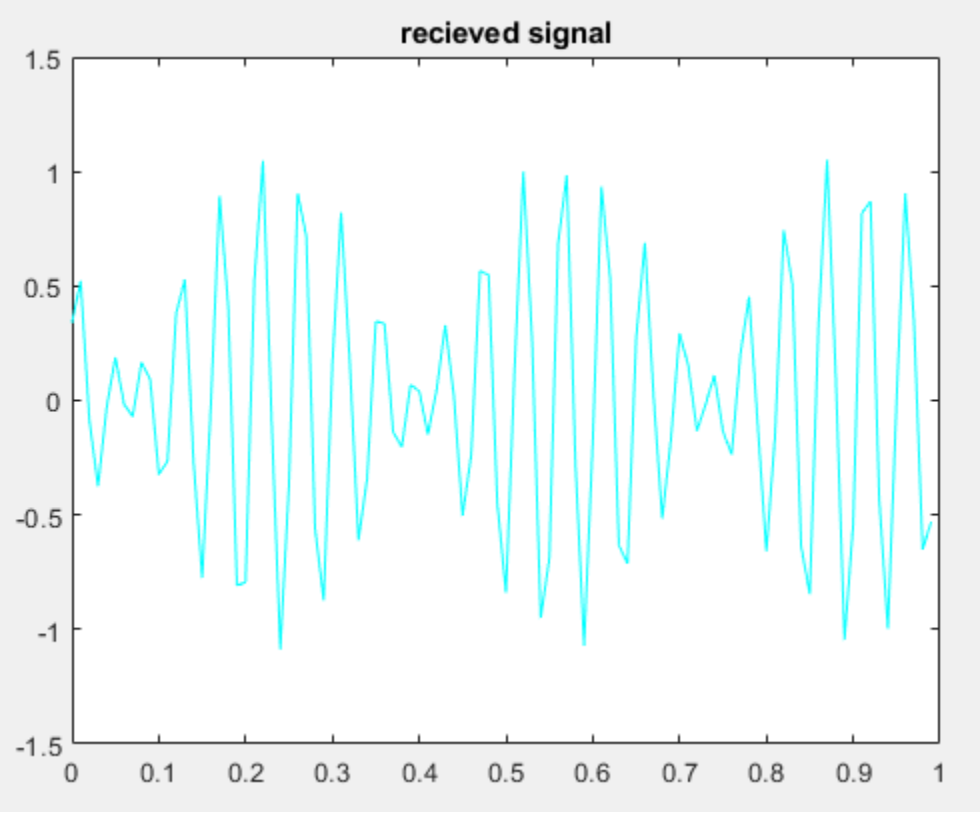
```
fc = 5; fs = 100;
tstart = 0; tend = 1;
B = 0.3; C = 3e8; p = 2/C;
ts = 1/fs;
t = tstart:ts:tend-ts;

v1 = 50; R1 = 2501000; a1 = 0.5;
v2 = 60; R2 = 200000; 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, color='cyan');
title('recieved signal');
save('recieved_p15.mat',"y_double");
```

پلات:



تمرین 1_6)

خروجی این بخش به این صورت است:

```
calculated distances are 200000 and 2500000
calculated speeds are 60 and 50
```

تابع این بخش بدین صورت است:

```
function p16(y)
    fc = 5; fs = 100;
    B = 0.3; C = 3e8; p = 2/C;

    dtft = fftshift(fft(y)); phase = angle(dtft); fabs = abs(dtft);
    max1 = 1; max2 = 1;
    for i=2:length(fabs)
        if (fabs(i) > fabs(max1))
            max1 = i;
        end
    end
    if (fabs(1) == max(fabs))
        max2 = 2;
    end
    for i=2:length(fabs)
        if ((fabs(i) > fabs(max2)) && (fabs(i) ~= fabs(max1)))
            max2 = i;
        end
    end
    [~, col1] = find(fabs==fabs(max1)); [~, col2] = find(fabs==fabs(max2));

    phase1 = abs(phase(col1(2)));
    fnew1 = (col1(2)-(length(y)/2)-1)*(fs/(length(y)));
    fd1 = fnew1-fc; td1 = phase1/(fnew1*2*pi);
    phase2 = abs(phase(col2(2)));
    fnew2 = (col2(2)-(length(y)/2)-1)*(fs/(length(y)));
    fd2 = fnew2-fc; td2 = phase2/(fnew2*2*pi);
    v1 = fd1/B; R1 = td1/p; v2 = fd2/B; R2 = td2/p - 1000;

    fprintf("calculated distances are %d and %d \n", round(R1, 1), round(R2, 1));
    fprintf("calculated speeds are %d and %d\n", round(v1, 1), round(v2, 1));
```

مانند بخش 1_3 عمل می‌کنیم اما چون دو سیگنال داریم، باید دو ماکزیمم پیدا کنیم. زیرا هر جسم یک ماکزیمم دارد. مگر اینکه پیک هر دو جسم یکسان باشد که در این تمرین چون فرکانس دو سیگنال متفاوت است این اتفاق نمی‌افتد. Max1 همان اندیس مقدار ماکزیمم است. اما Max2 اندیس دومین مقدار بیشینه است. نقاط پیک یک جسم در اندیس‌های Max1 و مقدار پیک دیگری در اندیس‌های MAx2 اتفاق می‌افتند. با این منطق و به مانند بخش 1_3 دو جسم را پیدا کرده و همانطور که در خروجی مشخص است سرعت و فاصله به درستی تشخیص داده شدند.

تمرین 1_7:

خیر تمایز آن دو ممکن نخواهد بود. در واقع 4 قله برابر خواهند بود و ما یک سرعت برای هر دو پیدا می‌کنیم اما فاصله هر کدام قابل ردیابی نیست. اگر بخواهیم فاصله را هم پیدا کنیم، حداقل اختلاف سرعت دو جسم باید به گونه اي باشد که اختلاف فرکانس ها در حوزه فوریه از رزولوشن فرکانسی ما (یک هرتز) بیشتر باشد تا ما بتوانیم پارامتر های دو جسم را درست تخمین بزنیم

$$\Delta f_{new} = \Delta f_d > 1\,hz \rightarrow \Delta v > \frac{10}{3}\ \frac{m}{s}\ = \frac{10}{3} \times 3.6 = 12 \frac{kn}{h}$$

بنابرین حداقل اختلاف سرعت دو جسم باید 12 کیلومتر بر ساعت باشد تا بتوانیم فاصله را درست تخمین بزنیم.

تمرین 1_8:

بله تمایز آن دو ممکن است. چون سرعت‌ها متفاوت است، 4 مقدار پیک مورد نیاز را داریم (در واقع یعنی دو قله از هم جدا هستند). حال با اندیس‌ها پیدا شده می‌توان فاصله دو جسم را پیدا کرد که برابر هم خواهند شد.

تمرین 1_9:

اگر سرعت همه اجسام متفاوت باشد (در واقع حداقل 12 کیلومتر بر ساعت تفاوت داشته باشند)، با این رزولوشن فرکانسی قادر به پیدا کردن همه آنان هستیم. مانند بخش 1_6 که بیشینه اول و دوم را پیدا کردیم، این دفعه چون تعداد اجسام نامشخص است، باید پیدا کردن بیشینه‌ها را تا جایی ادامه دهیم که مثلا مقدار آخرین بیشینه از حد مشخصی کمتر نشود. به این علت که نویز سیگنال را به عنوان جسم در نظر نگیریم. حال اگر k مقدار بیشینه پیدا شود، هر کدام دو اندیس می‌دهند که قرینه هم در حوزه فرکانسی هستند. با k اندیس در فرکانس مثبت، سرعت اجسام و فاصله آنان را پیدا می‌کنیم.

بخش دوم)

تمرین 2_1:

تابع زیر چهار ورودی دارد. نوت‌های مورد نیاز، فرکانس متناظر هر نوت، نوت‌ها و مدت زمان پخش هر کدام در آهنگ خواسته شده و در آخر نام فایل. به ازای هر نوت در آهنگ خواسته شده، آن را در نوت‌ها پیدا کرده تا فرکانس متناظرش را بردارد و سیگنال سینوسی متناظرش را بسازد و در انتهای آهنگ ساخته شده تا الان بگذارد. البته بعد از هر نوت یک بخش خالی می‌گذاریم تا فاصله بین هر دو نوت مشخص باشد. زمان بخش خالی را 0.025 گذاشتیم که در واقع $\frac{T}{20}$ است. در انتها موسیقی ساخته شده در فایل نوشته شده و پخش می‌شود.

```
function make_song(notes, freqs, song, name)
    tstart = 0; trest = 0.025; fs = 8000; ts = 1/fs;
    t_silence = tstart:ts:trest-ts; silence = zeros(size(t_silence));

    music = [];
    for i=1:length(song)
        note = song{1,i}; duration = song{2,i};
        [~, index] = find(notes==note);
        fc = freqs(index); t = tstart:ts:duration-ts;
        wave = sin(2*pi*fc*t);
        music = [music wave]; music = [music silence];
    end

    audiowrite(name, music, fs);
    sound(music);
```

اسکریپت زیر آهنگ داده شده در صورت پروژه را با کمک تابع بالا می‌سازد.

```
T = 0.5;
notes = ["C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"];
freqs = [523.25, 554.37, 587.33, 622.25, 659.25, 698.46, ...
         739.99, 783.99, 830.61, 880, 932.33, 987.77];

song = {'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#', 'E', 'F#', 'F#', 'E', 'F#', 'F#', 'D';
        T/2, T/2, T, T, T, T/2, T/2, T/2, T/2, T/2, T/2, T, 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, T, T/2, T/2, T, T/2, T/2, T, T, T};

make_song(notes, freqs, song, 'love_story.wav');
```

تمرین 2_2:

در این بخش ریف آهنگ sweet child o' mine برداشته شده و نوت‌های آن به تابع داده شده. خروجی در فایل wav. ذخیره شده است. اسکریپت این بخش که همانند بخش قبل است:

```
T = 0.5;
notes = ["C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"];
freqs = [523.25, 554.37, 587.33, 622.25, 659.25, 698.46, ...
         739.99, 783.99, 830.61, 880, 932.33, 987.77];

song = {'D', 'D', 'A', 'G', 'G', 'A', 'F#', 'A', ...
        'C', 'D', 'A', 'G', 'G', 'A', 'F#', 'A', ...
        'G', 'D', 'A', 'G', 'G', 'A', 'F#', 'A', ...
        'D', 'D', 'A', 'G', 'G', 'A', 'F#', 'A', ...
        'C', 'D', 'A', 'G', 'G', 'A', 'F#', 'A', ...
        'G', 'D', 'A', 'G', 'G', 'A', 'F#', 'A', ...
        'D', 'D', 'A', 'G', 'G', 'A', 'F#', 'A', ...
        'C', 'D', 'A', 'G', 'G', 'A', 'F#', 'A', ...
        'G', 'D', 'A', 'G', 'G', 'A', 'F#', 'A';
        T/2, T/2, T/2, T/2, T/2, T/2, T/2, T/2, ...
        T/2, T/2, T/2, T/2, T/2, T/2, T/2, T/2, ...
        T/2, T/2, T/2, T/2, T/2, T/2, T/2, T/2, ...
        T/2, T/2, T/2, T/2, T/2, T/2, T/2, T/2, ...
        T/2, T/2, T/2, T/2, T/2, T/2, T/2, T/2, ...
        T/2, T/2, T/2, T/2, T/2, T/2, T/2, T/2, ...
        T/2, T/2, T/2, T/2, T/2, T/2, T/2, T/2, ...
        T/2, T/2, T/2, T/2, T/2, T/2, T/2, T/2, ...
        T/2, T/2, T/2, T/2, T/2, T/2, T/2, T/2, ...
        };

make_song(notes, freqs, song, 'mysong.wav');
```

Size: 309 KB (316,844 bytes)

سیاز خروجی تابع 158400 است. همچنین حجم فایل تقریبا 309 کیلوبایت است.

پس $316844 \times 8 = 2534752$ بیت داریم که اگر این مقدار را تقسیم بر تعداد سمپل‌ها کنیم، هر سمپل در 16 بیت ذخیره شده است:

$$\frac{2534752}{158400} = 16$$

تمرین 2_3:

برای این بخش تابعی نوشته شده که آن را بخش بخش توضیح می‌دهم.

- خط 4 و 5: تعریف دو آرایه. اولی نوت کنونی که در حال ذخیره آن هستیم را نشان می‌دهد و آرایه‌ی دوم نوت‌های قبلی ذخیره‌شده را دارد.
- خط 6 تا 16: حلقه روی کل موزیک می‌زنیم. هر وقت به اندیسی برسیم که مقدار موزیک در آن اندیس و اندیس بعد برابر صفر است، به این معناست که به بخش خالی که میان نوت‌ها اضافه کرده بودیم رسیده‌ایم. بخش کنونی که از بعد از آخرین تکه خالی تا قبل شروع این تکه خالی است را در detected_notes ذخیره می‌کنیم و ارایه اول را خالی می‌کنیم تا از اول شروع

به کار کنیم. سپس برای اینکه از تمام تکه خالی بگذریم با یک حلقه وایل اندیس i را جلو می‌بریم.

- خط 18 تا 28: حال روی بخش‌های تشخیص داده شده حلقه زده و نوت هر کدام را پیدا می‌کنیم. از آن‌جایی که هر نوت را با یک تابع سینوس/کسینوس ذخیره کردیم، با تبدیل فوریه گرفتن و شیفت دادن حاصل تبدیل، دو ضربه در فرکانس‌های متقارن خواهیم داشت. با دستور find اندیس‌هایی که مقدار ماکزیمم دارند (همان اندیس‌های ضربه) را پیدا کرده که می‌دانیم 2 تا هستند، اولی فرکانس منفی و دومی مثبت. ما اندیس فرکانس مثبت را برمی‌داریم و فرکانسش را می‌سازیم. حالا در آرایه freqs ورودی به دنبال فرکانسی می‌گردیم که اختلاف بسیار کمی (در حد 10) با فرکانس پیدا شده توسط تبدیل فوریه داشته باشد. اندیس متناظرش در ارایه notes را پیدا کرده و نوت مد نظر را به جواب‌ها می‌افزاییم. مدت زمان اجرای این نوت را هم می‌توان از سایزش تقسیم بر فرکانس نمونه‌برداری پیدا کرد.

```
function music_notes=make_notes(music, fs, notes, freqs)
    music = music.';
    i = 0;
    cur_part_of_the_music = [];
    detected_notes = [];
    while(i<length(music)-1)
        i = i+1;
        if (music(i) == 0 && music(i+1) == 0)
            detected_notes{length(detected_notes)+1} = cur_part_of_the_music;
            cur_part_of_the_music = [];
            while(music(i) == 0 && i < length(music))
                i = i+1;
            end
        end
        cur_part_of_the_music(length(cur_part_of_the_music)+1) = music(i);
    end

    thresh = 10;
    for i=1:length(detected_notes)
        x = detected_notes{i};
        dtft = fftshift(fft(x));
        fabs = abs(dtft);
        [~, col] = find(fabs == max(fabs));
        frequency = (col(2)-(length(x)/2)-1)*(fs/(length(x)));
        idx = find(abs(freqs-frequency)<thresh);
        music_notes{1, i} = notes(idx);
        music_notes{2, i} = length(x)/fs;
    end
```

حال با این اسکریپت فایل ذخیره‌شده بخش قبل را باز کرده و به تابع بالا می‌دهیم.

```
T = 0.5;
notes = ["C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"];
freqs = [523.25, 554.37, 587.33, 622.25, 659.25, 698.46, ...
         739.99, 783.99, 830.61, 880, 932.33, 987.77];

[music,fs] = audioread("mysong.wav");
music_notes = make_notes(music, fs, notes, freqs);
```

خروجی این بخش به صورت زیر است:

music_notes

2x72 cell

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	"D"	"D"	"A"	"G"	"G"	"A"	"F#"	"A"	"C"	"D"	"A"	"G"	"G"	"A"	"F#"
2	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500

music_notes

2x72 cell

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	"A"	"G"	"D"	"A"	"G"	"G"	"A"	"F#"	"A"	"D"	"D"	"A"	"G"	"G"	"A"
2	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500

music_notes

2x72 cell

	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
1	"F#"	"A"	"C"	"D"	"A"	"G"	"G"	"A"	"F#"	"A"	"G"	"D"	"A"	"G"	"G"
2	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500

music_notes

2x72 cell

	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	"A"	"F#"	"A"	"D"	"D"	"A"	"G"	"G"	"A"	"F#"	"A"	"C"	"D"	"A"	"G"
2	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500

music_notes

2x72 cell

	61	62	63	64	65	66	67	68	69	70	71	72
1	"G"	"A"	"F#"	"A"	"G"	"D"	"A"	"G"	"G"	"A"	"F#"	"A"
2	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500

که دقیقا همان نوت‌هایی است که در بخش 2_2 دادیم.