گزارش پروژه سیگنال و سیستم
گروه دکتر امینی

محمد تسلیمی ۹۹۱۰۱۳۲۱

۲۵ تیر ۱۴۰۱

۱ پرسش های

۱) کد زده شده در زیر آمده است:

```
function [downsampled_Fs, resampled_audio] = import_audio(path, song_num, format)

    [audio, Fs] = audioread( strcat(path,"music", num2str(song_num), format));

    % getting mean over right and left channels
    audio = (audio(:,1) + audio(:,2))/2;
    %%% audioMono
    % downsample the audio to 8 KHz

    downsampled_Fs = 8000;
    %%% resampled_audio
    resampled_audio = resample(audio, downsampled_Fs, Fs);
  end

```

همانطور که در کد بالا مشخص است ابتدا با دستور audioread آهنگ مورد نظر را import کرده ایم و میانگین دوکانال آن رو گرفته و در نهایت با دستور resample موزیک با نرخ نمونه برداری ۸۰۰۰ تولید شده است.

۲) در این سوال تبدیل فوریه به طرز خاصی پیاده سازی شده است که در زیر آمده است:

```
function single_sided_power_spectrum = FFT(X)
  %%% single_sided_power_spectrum

  fft_signal =  ((abs(fft(X))/length(X)).^2);
  fft_signal = fft_signal(1:floor(length(X)/2) + 1);
  fft_signal(2: end -1)=fft_signal(2:end -1) * 2;
  single_sided_power_spectrum = fft_signal;
end

```

در این سوال مراحلی که در پروژه خواسته شده بود پیاده سازی شده است.

۳) کد زده شده در این سوال به صورت زیر است:

```
function [time, freq, time_freq_mat] = STFT(audio, Fs, window_time)
  window_length = Fs*window_time;
```

```
    window_num = floor(length(audio)/(window_length/2));
    time_freq_mat = zeros(1+floor(window_length/2),window_num
-1);
    % calculating fft using an overlapping sliding window
    for i = 1:window_num-1
      begin = ((floor((window_length/2))*(i-1))+ 1);
      End = floor(window_length*i/2) + floor(window_length/2)
+ 1;
      X = audio(begin:End);
      temp = FFT(X);
      time_freq_mat(:,i) = temp + 0.001*rand(length(temp), 1);

      %%% time_freq_mat
    end
    % time and freq vectors
    time = (window_time/2)*(1:(window_num-1));
    freq = Fs*(0:floor(window_length/2))/window_length;
 end

```

در این سوال یک پنجره لغزان با طول تعریف شده روی آهنگ حرکت میکند و در هر پنجره تبدیل فوریه گرفته میشود. در این بخش پنجره های لغزان ۵۰ درصد همپوشانی دارند. همچنین در این بخش به تبدیل فوریه هر پنجره مقداری نویز اضافه شده است. نویز به این دلیل اضافه شده است که اگر در آخر اهنگ صدایی نبود، در آن زمان ها سیگنال صفر شده و تابع find anchor point که براساس ماکزیمم گیری از تبدیل فوریه بدست آمده است، نمیتواند ماکزیمم گیری کند. به همین دلیل نویز اضافه کرده ایم که تابع به مشکل نخورد.

۴) در این بخش به توضیح فایل create database میپردازیم. در این فایل ابتدا مسیر های پوشه های تابع های مورد نیاز اضافه شده است. سپس دیتابیس که اطلاعات رو به صورت key and value ذخیره میکند، ساخته میشود. در ادامه ابتدا نام آهنگ های موجود در پوشه musics گرفته میشود تا با استفاده از نام ها، آهنگ ها لود شوند و پردازش شوند. پس از اینکه نام ها گرفته شده فرایند زیر برای تک تک موزیک ها انجام میشود تا اثر انگشت آن تولید شود. ابتدا موزیک با تابع import audio لود میشود. سپس تابع STFT روی آن اعمال میشود و با خروجی آن anchor point های آهنگ با تابع find anchor point مشخص میشوند. سپس با استفاده از anchor point ها hash tag با تابع create hash tag برای آهنگ ساخته میشود. در نهایت hash key and hash value های ساخته شده به دیتا بیس اضافه شده و ذخیره میشوند.

سوال۵) کد زده شده برای ساخت لیست به صورت زیر است:

```
  clc; close all;
```

```
    list = [];
    % searching for found hash-keys in the database
    for i = 1:length(hash_key)
    key_tag = [num2str(hash_key(i, 1)), '*', num2str(hash_key(i,
    2)), '*',  num2str(hash_key(i, 3))];
    if (isKey(database, key_tag))
    temp1 = split(database(key_tag),'+');
    for j = 1:length(temp1)
    temp2 = split(temp1{j},'*');
    list = [list; [str2num(temp2{1}),str2num(temp2{2}),
   hash_value(i,2)]];
    end
    end
    end

```

در این بخش برای ساختن لیست میخواهیم با توجه به hash key های بدست آمده از آهنگ تست، از دیتا بیس آهنگ مربوطه را پیدا کنیم. برای اینکار تمام hash key های بدست آمده را طی کرده(با فور لوپ) سپس key tag متناظر با hash key فعلی را مطابق آنچه در دیتابیس ذخیره کرده ایم میسازیم. حال چک میکنیم که آیا این key tag در دیتابیس وجود دارد یا نه. اگر وجود داشت از انجا که ممکن است برای هر hash key تعدادی hash value وجود داشته باشد، برای هر hash value (با استفاده از دستور های split and for) شماره آهنگ، زمان آن در آهنگ اصلی و زمان آن در آهنگ تست را به لیست اضافه میکنیم. این کار را برای همه ی hash key ها انجام داده تا لیست تکمیل گردد.

سوال۶) داده ساختار hashmap ،داده ساختاری است که اطلاع را به صورت key-value pair ذخیره میکند. مرتبه سرچ در آن $O(1)$ است. این یعنی که زمان سرچ در آن تقریبا ثابت است. پس این داده ساختار، داده ساختار مطلوبی برای این کار میباشد.

سوال۷) در این سوال میخواهیم ببنیم هریک از آهنگ های تست مربوط به کدام آهنگ هست. و همچنین زمان شروع آهنگ تست در آهنگ اصلی را نیز تعیین کنیم. برای تعیین زمان شروع آهنگ تست در آهنگ اصلی ابتدا تمام زمان های اون سطر هایی که به آهنگ تشخیص داده شده توسط الگوریتم مربوط میشود را در یک لیست ذخیره کرده و سپس میانه آن را بدست می آوریم. برای بدست آوردن زمان شروع آهنگ این عدد را بر ۲۰ و سپس بر ۶۰ تقسیم میکنیم تا به دقیقه تبدیل شود. تقسیم بر بیست با توجه به windows length صورت میگیرد. همچنین میانه رو انتخاب میکنیم چون برخی از زمان ها ممکن است پرت باشد. زیرا بخشی از اهنگ ممکن است در جاهای مختلف آن تکرار شده باشد و در این صورت تمام زمان های تکرار آن در لیست می آید. در لیست تعداد زیادی از زمان ها نزدیک بهم میباشد ولی تعداد کمی نیز این زمان ها فاصله دارند. به همین

دلیل از میانه استفاده کرده ایم. حال به برسی موزیک های تست میپردازیم.(توجه کد زده شده برای تشخیص زمان موزیک تست در موزیک اصلی در فایل search database در زیر بخشی که لیست تشکلیل میشود زده شده است.)

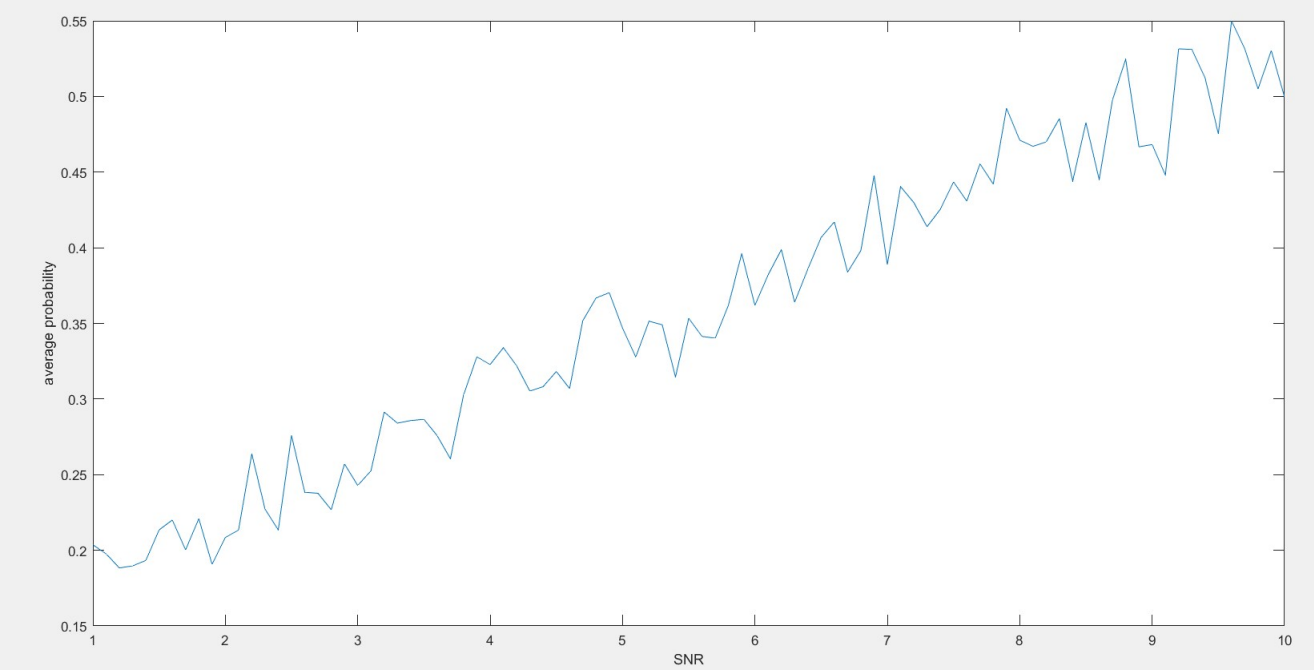
- موزیک تست ۱ مربوط به آهنگ شماره ۵ میشود و در آهنگ اصلی تقریبا پس از ۱ دقیقه و ۳۸ ثانیه شروع میشود.
- موزیک تست ۲ مربوط به آهنگ شماره ۲ میشود و در آهنگ اصلی تقریبا پس از ۳ دقیقه و ۱۵ ثانیه شروع میشود.
- موزیک تست ۳ مربوط به آهنگ شماره ۱۳ میشود و در آهنگ اصلی تقریبا پس از ۱ دقیقه و ۵۸ ثانیه شروع میشود.
- موزیک تست ۴ مربوط به آهنگ شماره ۳۶ میشود و در آهنگ اصلی تقریبا پس از ۲ دقیقه و ۷ ثانیه شروع میشود.
- موزیک تست ۵ مربوط به آهنگ شماره ۴۳ میشود و در آهنگ اصلی تقریبا پس از یک دقیقه و ۴۰ ثانیه شروع میشود.
- موزیک تست ۶ مربوط به آهنگ شماره ۴۹ میشود و در آهنگ اصلی تقریبا پس از ۲ دقیقه و ۳۰ ثانیه شروع میشود.
- موزیک تست ۷ مربوط به آهنگ شماره ۱۷ میشود و در آهنگ اصلی تقریبا پس از یک دقیقه و ۲۸ ثانیه شروع میشود.
- موزیک تست ۸ مربوط به آهنگ شماره ۱۱ میشود و در آهنگ اصلی تقریبا پس از ۲ دقیقه و ۷ ثانیه شروع میشود.
- موزیک تست ۹ مربوط به آهنگ شماره ۳۲ میشود و در آهنگ اصلی تقریبا پس از ۴ ثانیه شروع میشود.
- موزیک تست ۱۰ مربوط به آهنگ شماره ۱ میشود و در آهنگ اصلی تقریبا پس از ۲ دقیقه و ۱۹ ثانیه شروع میشود.

سوال۸) در این سوال یک فایل به اسم Question8 ساخته شده که در پوشه اصلی پروژه است و در آن تابع گرفتن بیست ثانیه از آهنگ پیاده سازی شده است. برای تست از آهنگ شماره دو استفاده شده است. کد زده شده برای گرفتن بیست ثانیه از آهنگ به صورت زیر است:(توجه کنید برای اجرای این فایل هر دفعه باید کل بخش ها اجرا شود.)

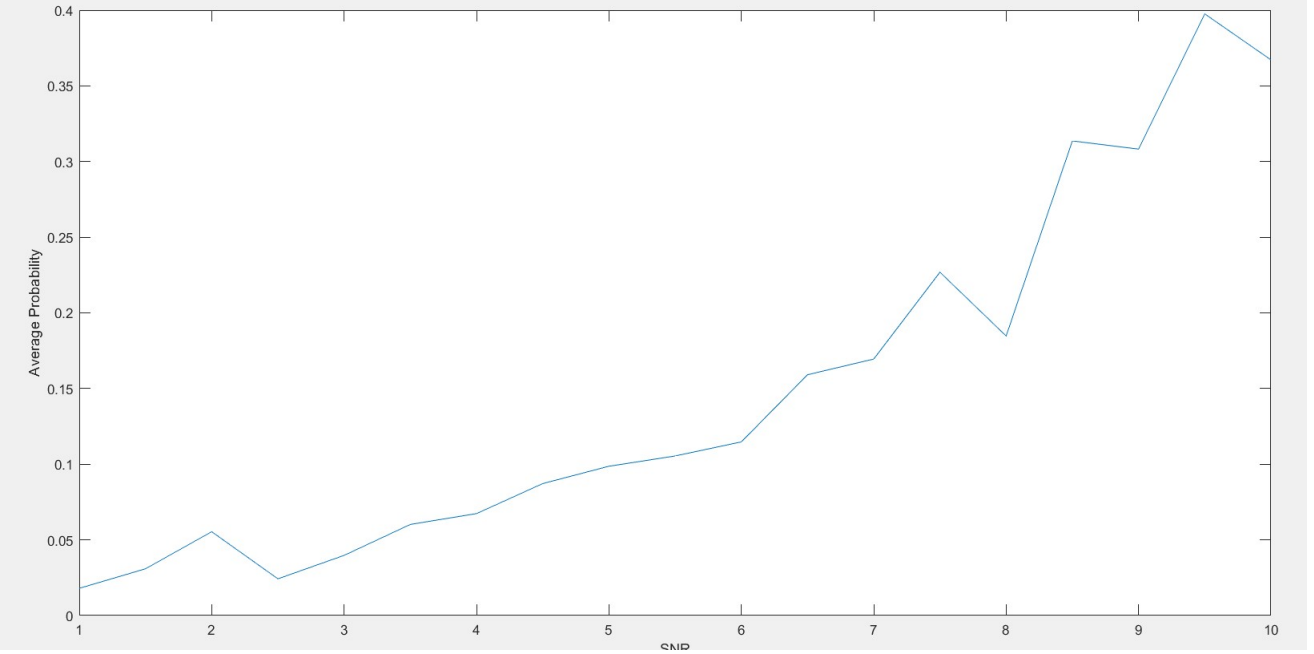
```
% Question 8
clc; clear;
addpath('musics/');
[audio, Fs] = audioread("music2.mp3");
downsampled_Fs = 8000;
resampled_audio = resample(audio, downsampled_Fs, Fs);
leng = 4*Fs; % getting 20s of music 2
test_audio = resampled_audio(1:leng);
std = 0.1;
SNR = 1; % 1: SNR = 20, 2: SNR = 10, SNR = 5, SNR = 2, SNR =
1,
noise = randn(1, leng)*std/db2mag(SNR);
test_audio = test_audio + noise;
audiowrite("music24.wav", test_audio, downsampled_Fs);

```

این بخش برای SNR های مختلف تکرار شد. از $SNR = 20$ شروع کردیم. در $SNR = 20$ موزیک را با احتمال ۲۰ درصد درست تشخیص داد در حالی که موزیک اصلی را با احتمال ۹۹ درصد درست تشخیص می دهد. در $SNR = 10$ با احتمال ۱۸ درصد درست تشخیص میدهد. در $SNR = 5$ با احتمال ۱۴ درصد درست تشخیص میدهد. در $SNR = 2$ در با احتمال ۱۲ درصد درست تشخیص میدهد. و اما در $SNR = 1$ درست تشخیص نمیدهد. موزیک های تست شده در پوشه test musics میباشد که از شماره ۲۰ تا ۲۴ نام گذاری شده است.

سوال۹) در این سوال یک فایل به اسم Question9 درپوشه اصلی پروژه درست شده است. نتیجه ران کردن این کد برای موزیک یک و پنج مطابق شکل زیر است. همانطور که در شکل مشخص است با افزایش SNR میانگین احتمال افزایش میابد که منطقی است. اما افزایش آن نوسان دارد که آن هم منطقی است زیرا با توجه به بیست ثانیه انتخابی از آهنگ احتمال تشخیص آهنگ تغییر میکند به طوری که زمانی که بخش بی کلام آهنگ انتخاب میشود احتمال تشخیص آن با افزودن نویز کم میشود اما اگر آن بخش از آهنگ تست با کلام باشد احتمال تشخیص درست آن افزایش میابد.
آهنگ شماره نه با درصد افزایش بیشتر SNR رسم شده زیرا مدت ران کردن آن خیلی زیاد بود.

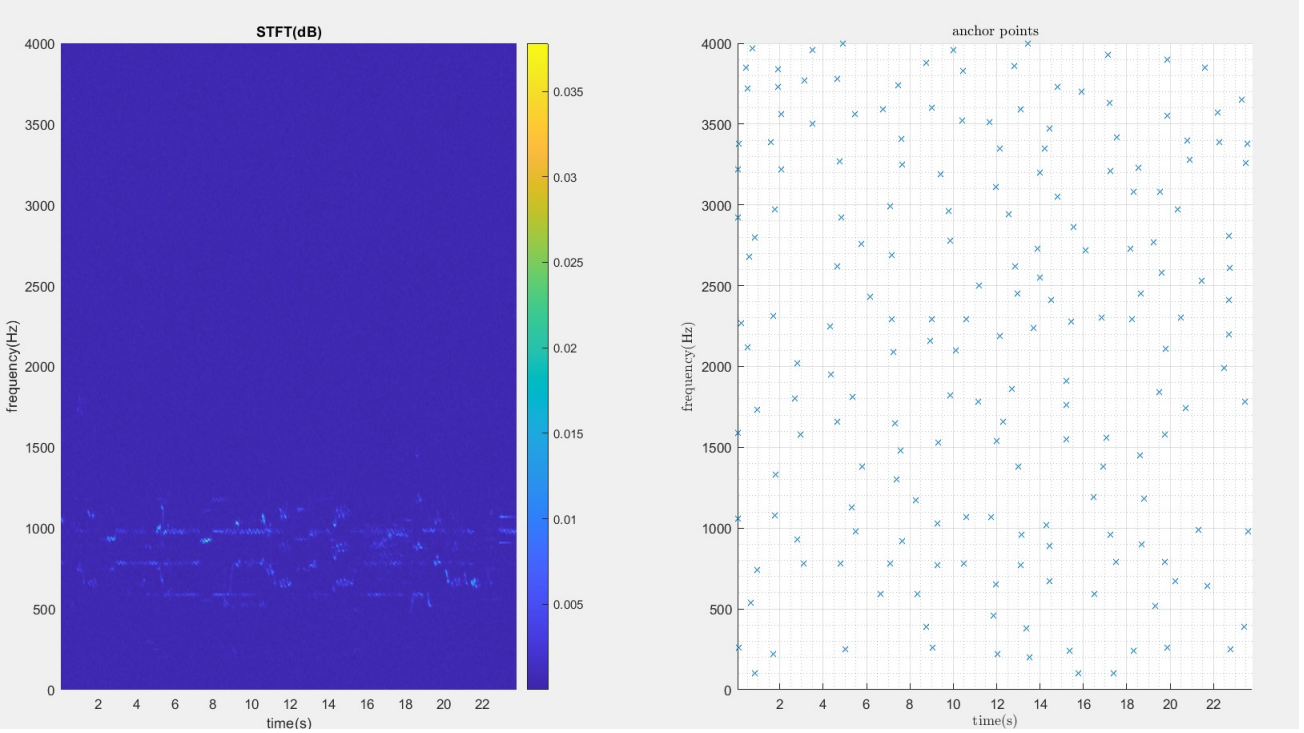


شکل ۱: نمودار میانگین احتمال بر حسب SNR برای آهنگ شماره یک

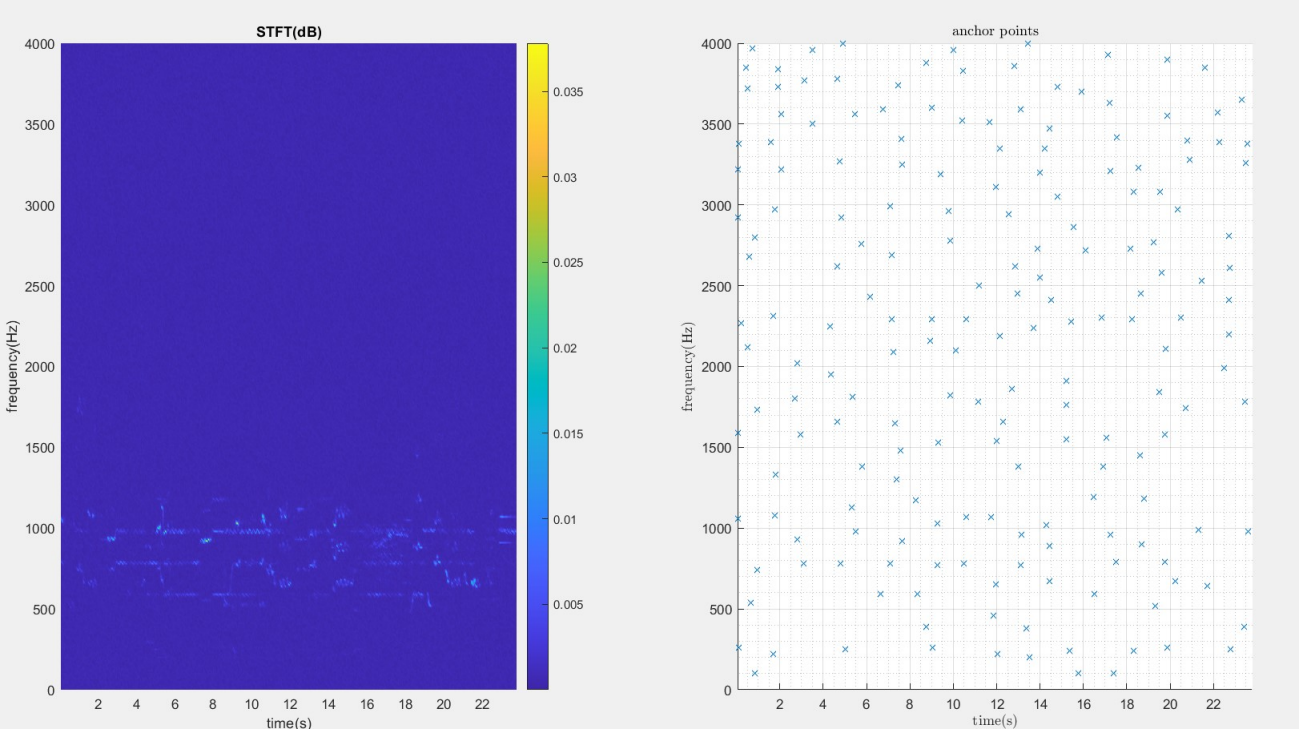


شکل ۲: نمودار میانگین احتمال بر حسب SNR برای آهنگ شماره ۹

سوال۱۰) در این سوال صدا ابتدا در کنار بلندگوی لپ تاپ ضبط شد و استفاده شد(music30) که آهنگ درست تشخیص داده شد. ولی دفعه دوم گوشی رو از بلند گو دور و کنار باد کولر گرفتم و در این حالت که خیلی صدای متفرقه داشت الگوریتم درست کار نکرد.(music31) تصاویر مربوط به نمودار ها در شکل زیر آماده است:(در این بخش از آهنگ شماره دو استفاده شده.)

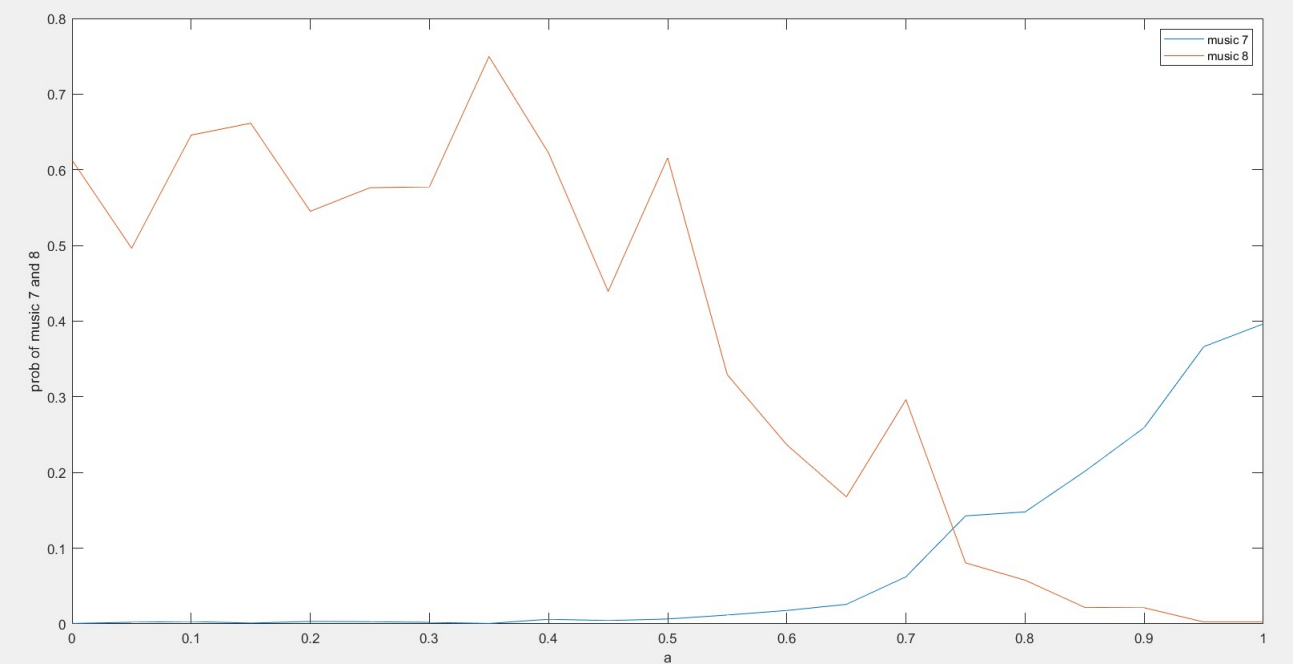


شکل ۳: first test (music30)



شکل ۴: second test (music31)

سوال ۱۱) در این سوال موزیک ۷ و ۸ برای تست انتخاب شده است. زمانی که ضرایب هر دوموزیک نیم بود. الگوریتم آهنگ ۸ رو تشخیص داد. این موضوع از نمودار زیر نیز مشخص است. در این سوال آلفا با step های 0.05 تغییر میکند.



شکل ۵: نمودار تصمیم گیری الگوریتم با تغییر ضریب آلفا