EE 411: Digital Signal Processing (Fall-2016)
Mini-Project on Fake Music Detection

Objective

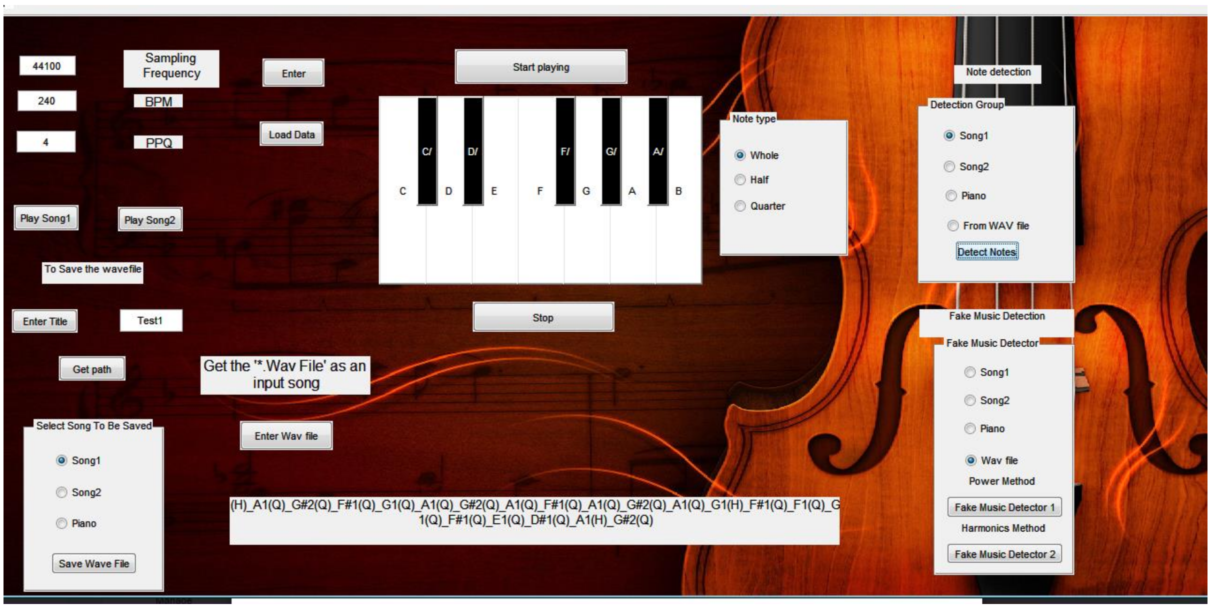
- **Note detection**: To identify the notes being played in a given piano song.
- To generate a **Matlab GUI for synthesis** of a song given various parameters.
- To detect if a song is **real or computer synthesised**.

Background

On a piano, the keyboard is divided into 8 octaves (with total 88 key frequencies ranging from 27.5 Hz to 4186.01Hz) the notes in one octave being twice the frequency of the notes in the next lower octave. For example, the reference note is the A above Middle-C A-440 middle-C which is usually called A-440 (or A4) because its frequency is 440 Hz. Each octave contains 12 notes (5 black keys and 7 white) and the ratio between the frequencies of the notes is constant between successive note. As a result, this ratio must be 2^(1/12).Musical notation shows which notes are to be played and their relative timing (half, quarter, or eighth).

Note Detection

The objective is to detect the notes been played in a given song and specify if the note is half note, full note or quarter note (A *quarter note* (half note) is a *note* played for one *quarter (half note)* of the duration of a whole *note* (or semibreve)).The first step was to segment the input song (using buffer function) into duration of quarter note (minimum duration) (we have assumed a quarter note of .25 seconds). Filtering process is carried out by using a second order band pass Butterworth filter and applying on the segment for all 88 frequencies. The detected note is the one which has the highest power among the notes detected in the given segment (for Table relating the key frequency with the notes see the appendix).



Sampling Frequency: 44100
BPM: 240
PPQ: 4
Enter Title: Test1
Select Song To Be Saved: [✓ Song1]
Note type: [✓ Whole]
Detection Group: [✓ Song1]
Fake Music Detector: [✓ Wav file]

```
(H)_A1(Q)_G#2(Q)_F#1(Q)_G1(Q)_A1(Q)_G#2(Q)_A1(Q)_F#1(Q)_A1(Q)_G#2(Q)_A1(Q)_G1(H)_F#1(Q)_F1(Q)_G
1(Q)_F#1(Q)_E1(Q)_D#1(Q)_A1(H)_G#2(Q)
```

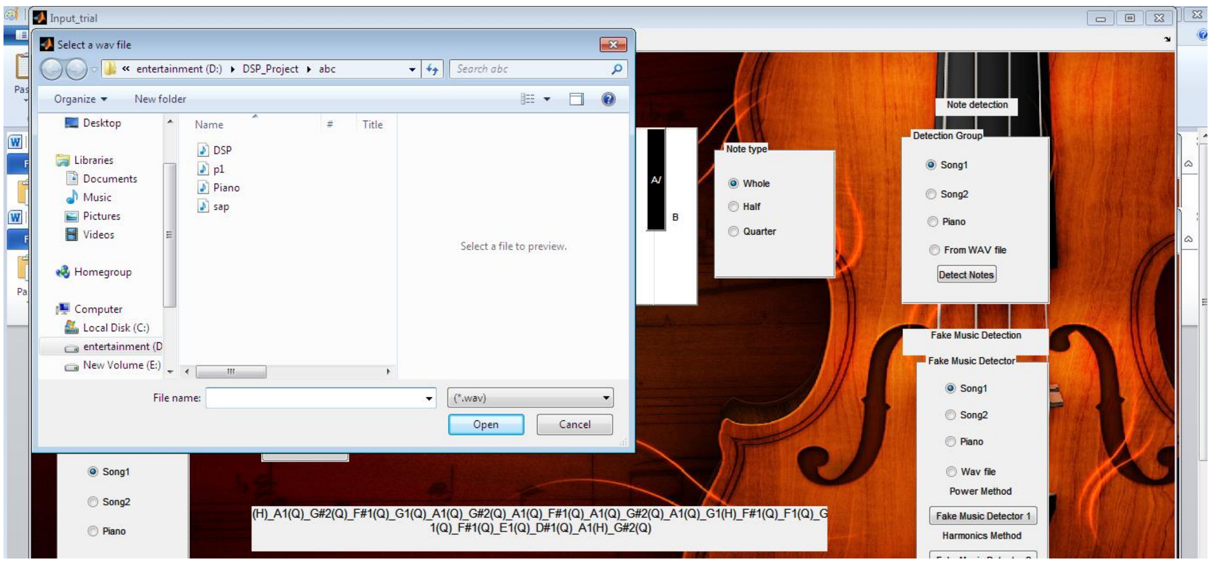
Fig 1: Indicating notes in Song1

Fake music Detection

The objective is to detect a fake music (synthesised through computer) and real piano music. We have used the criteria to differentiate between fake music from real one on basis of the pattern of the harmonics present in a given song and the exponential decay factor of the harmonics. A real music will always have harmonics present due the transition from one key frequency to another, which is not always true in the real music. Our aim is to filter all the fundamental 88 frequencies from the song and calculate the power of the harmonics. The power of the harmonics if greater than .06*original signal power will give rise to a fake music. (The threshold is set on basis of number of observations done on fake and real music).

Another criteria is to observe the pattern of harmonics present in a given signal. The song is first segmented by 50m sec frame. On each frame, we detect the frequency with maximum power and second maximum power by applying the filter with 88 piano frequencies, and calculate the amplitude of the harmonics present in the segment. An exponential function is applied which tells us the decaying factor. This is done for all segments. Variance is then calculated for all decaying parameters concatenated as a row vector for all segment present.

If the variance is less than 200 then it is fake music. (The threshold is set on basis of observations done on fake and real music.).**Detected output is shown in the command window as soon as it is tested on GUI.**



```
(H)_A1(Q)_G#2(Q)_F#1(Q)_G1(Q)_A1(Q)_G#2(Q)_A1(Q)_F#1(Q)_A1(Q)_G#2(Q)_A1(Q)_G1(H)_F#1(Q)_F1(Q)_G
1(Q)_F#1(Q)_E1(Q)_D#1(Q)_A1(H)_G#2(Q)
```

Fig 2: Input as a wave file to detect notes and Fake Music

Synthesizing a Song using Matlab GUI

The objective of this is to allow user to synthesize a song using matlab GUI. The user will add input matrix of the song with parameters such as sampling frequency , BPM and PPQ and the GUI will generate the desired song.

Timing in a Song

Musicians often think of the tempo, or speed of a song, in terms of “beats per minute” or BPM, where the beats are usually quarter notes.

beats_per_second = bpm/60;

seconds_per_beat = 1/beats_per_second;

seconds_per_pulse = seconds_per_beat / 4;

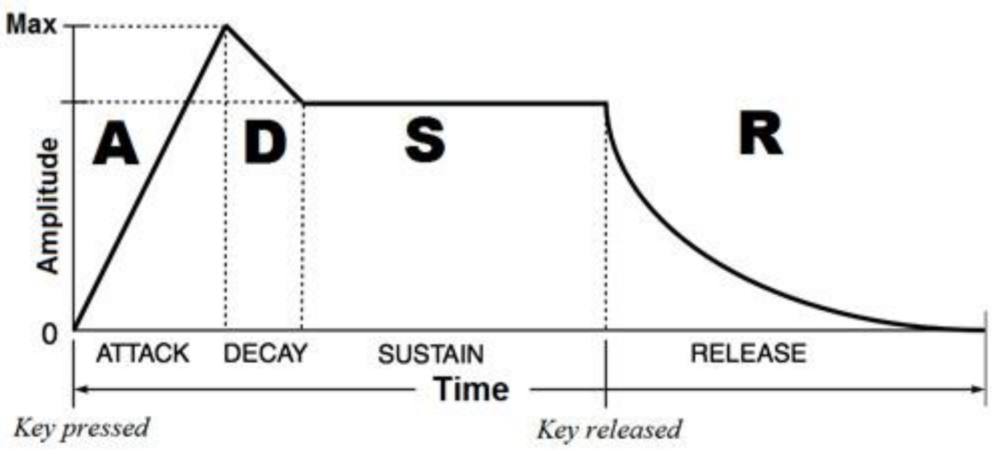
By knowing the pulses per second we can know the duration of pulses for the given song.

If a song has a BPM of 240 will play twice as fast as a song played at BPM of 120.

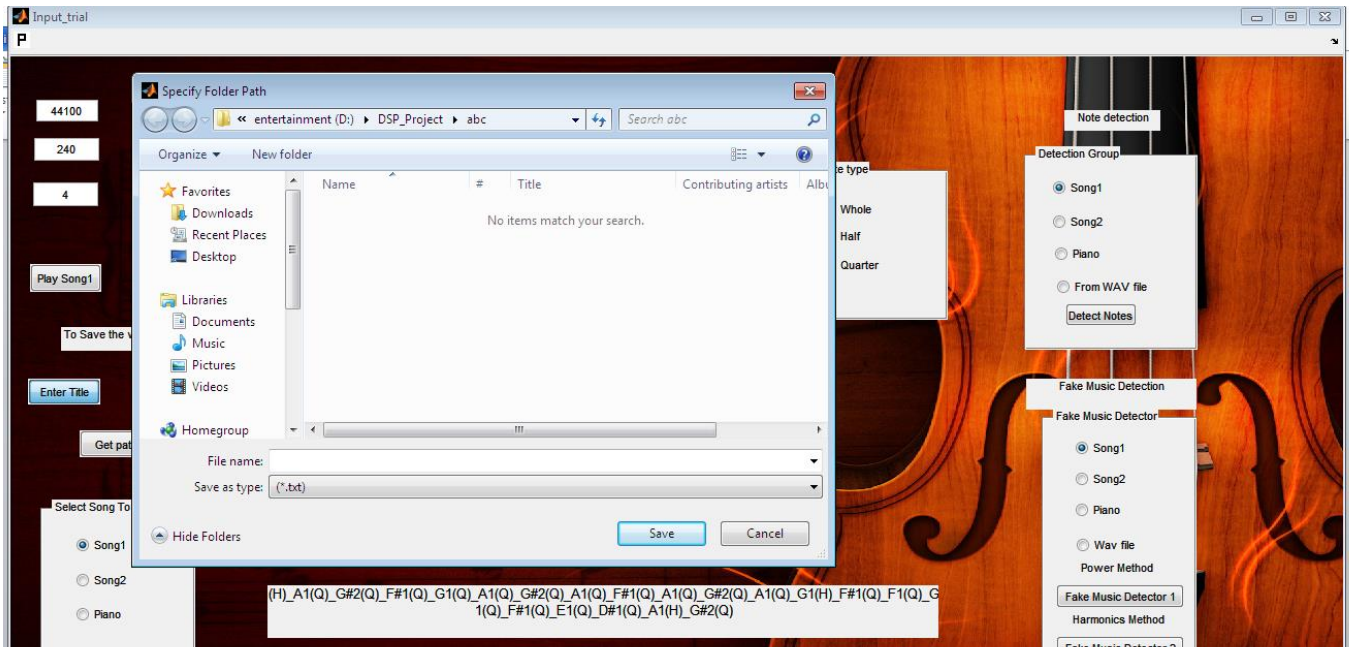
Another problem with the synthesizing of a music is that the notes are not continuous. Therefore, we can add small pause between the signal to generate a smooth signal.

Musical Tweaks

The music generated through GUI will be sounding artificial because it is composed from sinusoids. We want that the signal should fade in quickly and fade out slowly. The quality of song can be increased by adding an Envelope to the signal. A standard way to define the envelope function is to divide E (t) into four sections: attack (A), delay (D), sustain (S), and release (R). Together these are called **ADSR**. The attack is a quickly rising front edge, the delay is a small short-duration drop, the sustain is more or less constant and the release drops quickly back to zero.



The user also gets a choice to save his/her song into a wave file at the desired path, and give a name to the song generated.



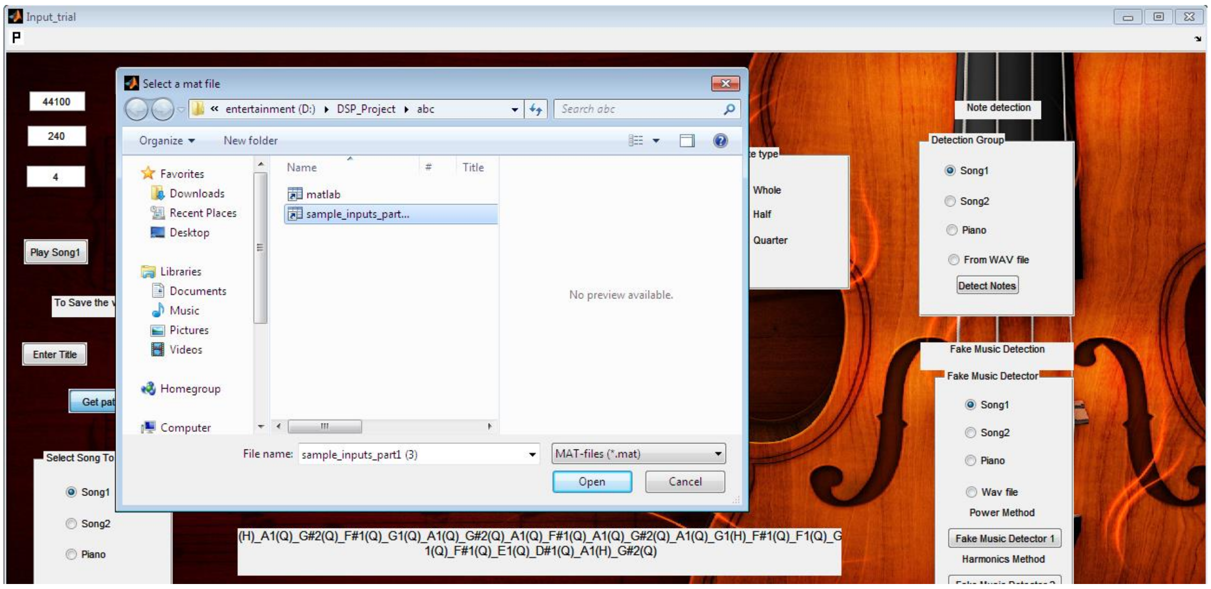
No items match your search.

File name:

Save as type: (*.txt)

(H)_A1(Q)_G#2(Q)_F#1(Q)_G1(Q)_A1(Q)_G#2(Q)_A1(Q)_F#1(Q)_A1(Q)_G#2(Q)_A1(Q)_G1(H)_F#1(Q)_F1(Q)_G
1(Q)_F#1(Q)_E1(Q)_D#1(Q)_A1(H)_G#2(Q)

Fig 3: To save the synthesized music in a wave file at a given location



File name: sample_inputs_part1 (3)

MAT-files (*.mat)

```
(H)_A1(Q)_G#2(Q)_F#1(Q)_G1(Q)_A1(Q)_G#2(Q)_A1(Q)_F#1(Q)_A1(Q)_G#2(Q)_A1(Q)_G1(H)_F#1(Q)_F1(Q)_G
1(Q)_F#1(Q)_E1(Q)_D#1(Q)_A1(H)_G#2(Q)
```

Fig 4: To take input as a .mat file for Synthesizing a song

Filter Design Requirements

The aim was to get the output such that only one frequency is dominant in a given segment for note detection. The Butterworth filter is a type of signal processing filter designed to have as flat a frequency response as possible in the pass band. Using Butterworth filter of 2nd order our design requirements were met. We designed 88 filters for all piano key frequencies to detect the note.

Pole-zero Plot for Butterworth Filter

The plot is show for Butterworth filter for middle tone frequency of 440Hz.

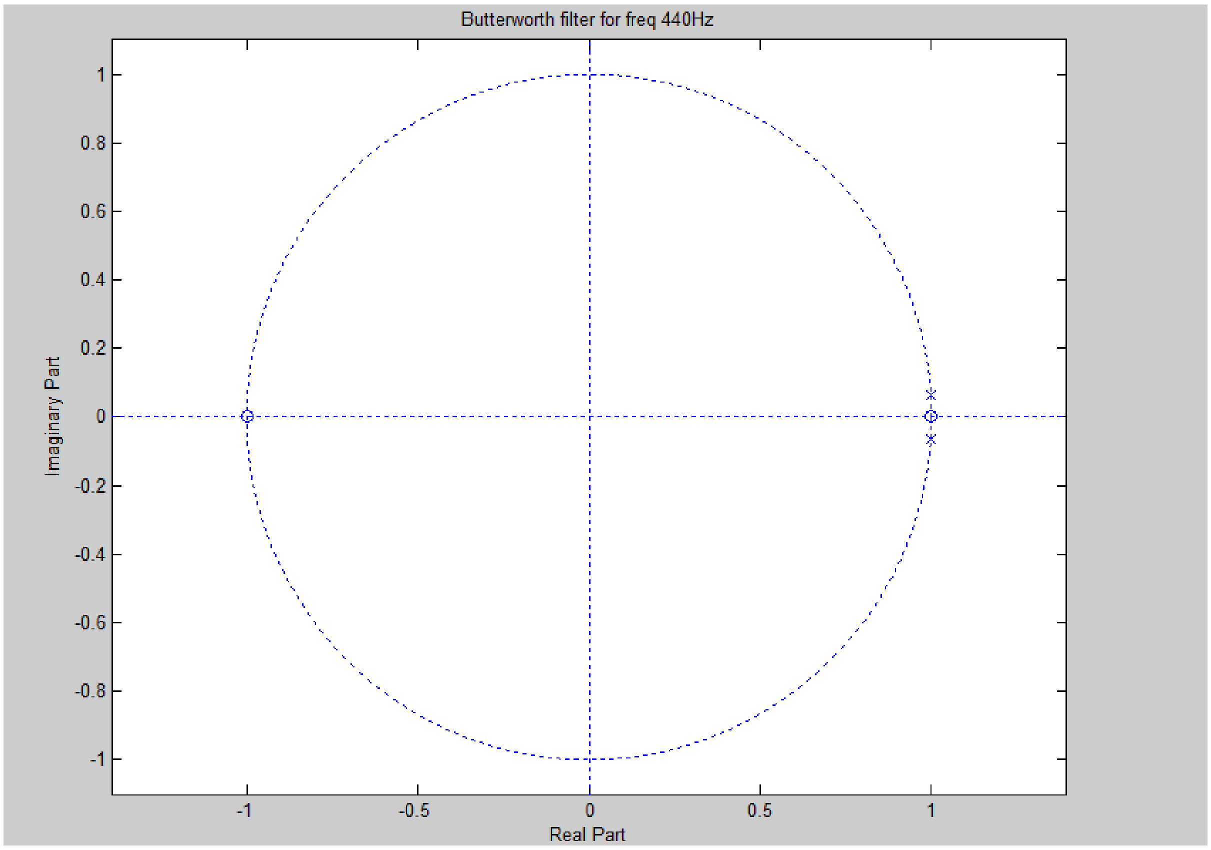


Fig 5: Pole-Zero Plot for Butterworth Filter

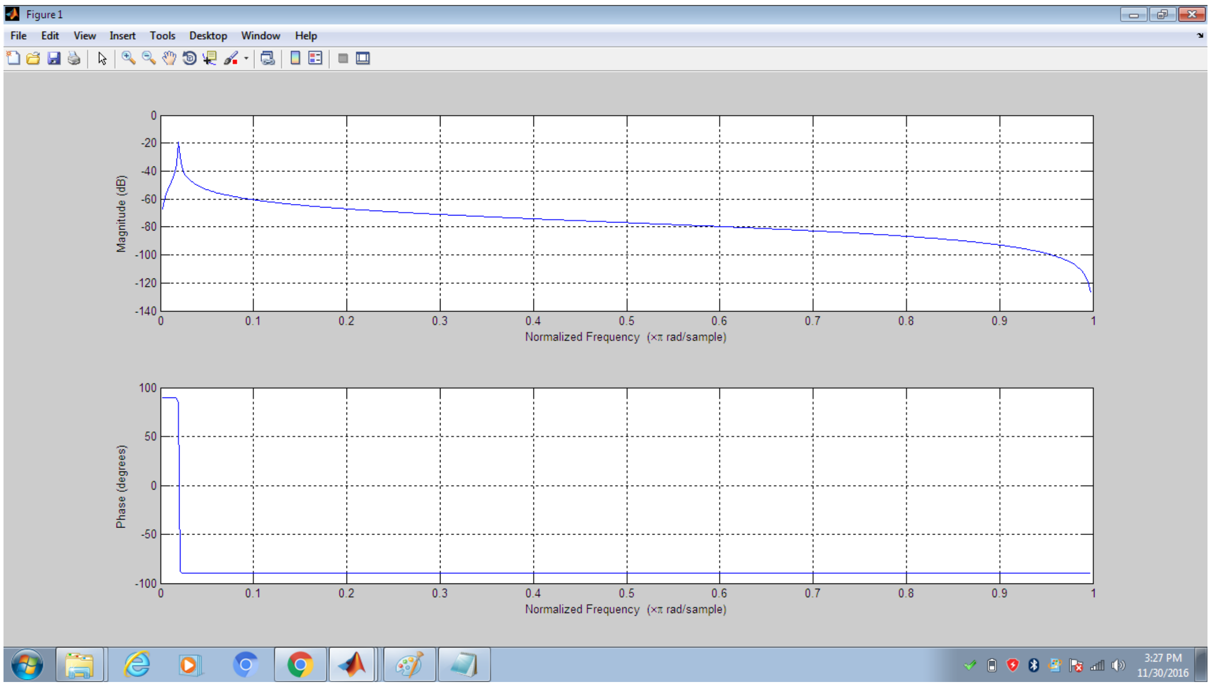


Fig6 :Magnitude and Phase response for Butterworth Filter

The second filter used to suppress the piano frequencies for fake detection was a Notch filter. Notch filter attenuates signals within a very narrow band of frequencies. The requirement was to find the power of the signal without harmonics of the fundamental frequencies.

Pole-Zero Plot for Notch Filter

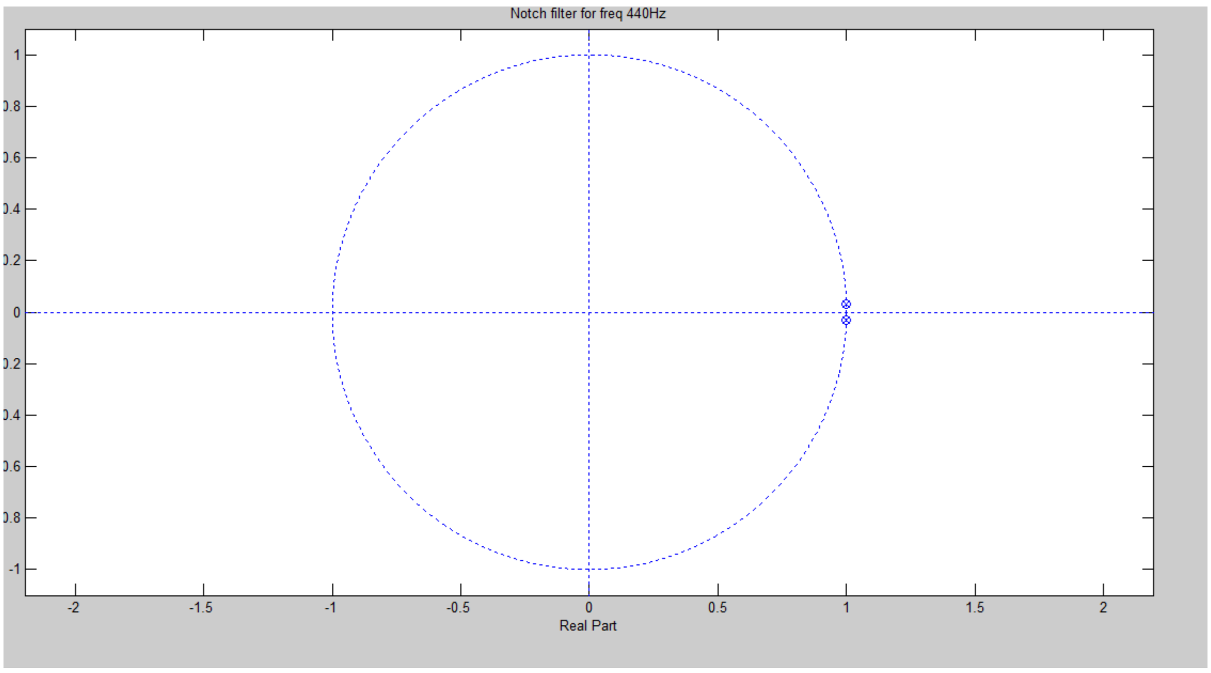


Fig 7: Pole-Zero Plot for Notch Filter

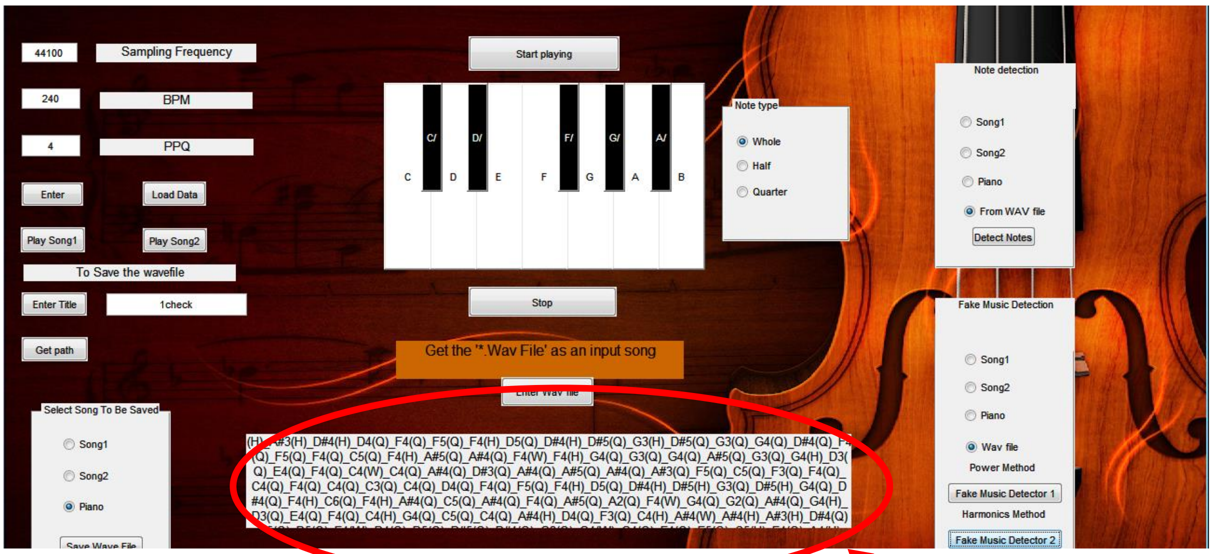
Requirements Constraint and Assumptions

- Calculating the BPM of a song: we have assumed the song to be detected has a BPM (beats per minute) of 240 at 4 PPQ (pulse per quarter).
- **Filter Design:** Designing of filters consumed much of the time for generating the desired filter response.
- Insufficient knowledge of Matlab GUI.

Observations

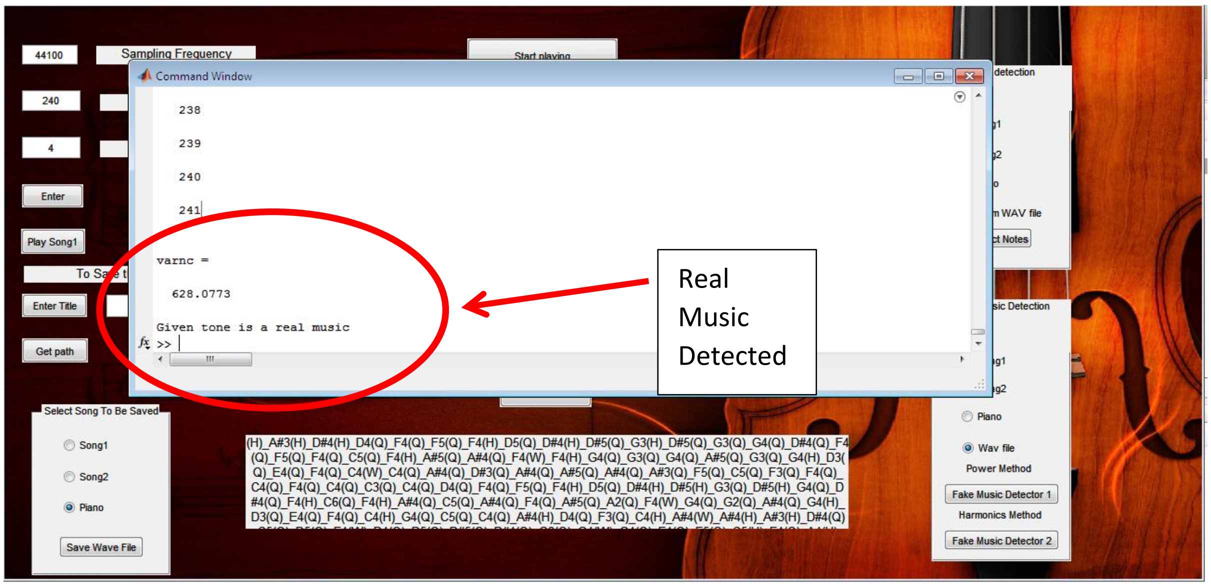
Music detection of a real song

(Input file used Real_music.wav)



```
(H)_A#3(H)_D#4(H)_D4(Q)_F4(Q)_F5(Q)_F4(H)_D5(Q)_D#4(H)_D#5(Q)_G3(H)_D#5(Q)_G3(Q)_G4(Q)_D#4(Q)_F4
(Q)_F5(Q)_F4(Q)_C5(Q)_F4(H)_A#5(Q)_A#4(Q)_F4(W)_F4(H)_G4(Q)_G3(Q)_G4(Q)_A#5(Q)_G3(Q)_G4(H)_D3(
Q)_E4(Q)_F4(Q)_C4(W)_C4(Q)_A#4(Q)_D#3(Q)_A#4(Q)_A#5(Q)_A#4(Q)_A#3(Q)_F5(Q)_C5(Q)_F3(Q)_F4(Q)_
C4(Q)_F4(Q)_C4(Q)_C3(Q)_C4(Q)_D4(Q)_F4(Q)_F5(Q)_F4(H)_D5(Q)_D#4(H)_D#5(H)_G3(Q)_D#5(H)_G4(Q)_D
#4(Q)_F4(H)_C6(Q)_F4(H)_A#4(Q)_C5(Q)_A#4(Q)_F4(Q)_A#5(Q)_A2(Q)_F4(W)_G4(Q)_G2(Q)_A#4(Q)_G4(H)_
D3(Q)_E4(Q)_F4(Q)_C4(H)_G4(Q)_C5(Q)_C4(Q)_A#4(H)_D4(Q)_F3(Q)_C4(H)_A#4(W)_A#4(H)_A#3(H)_D#4(Q)
```

Detected Note sequence

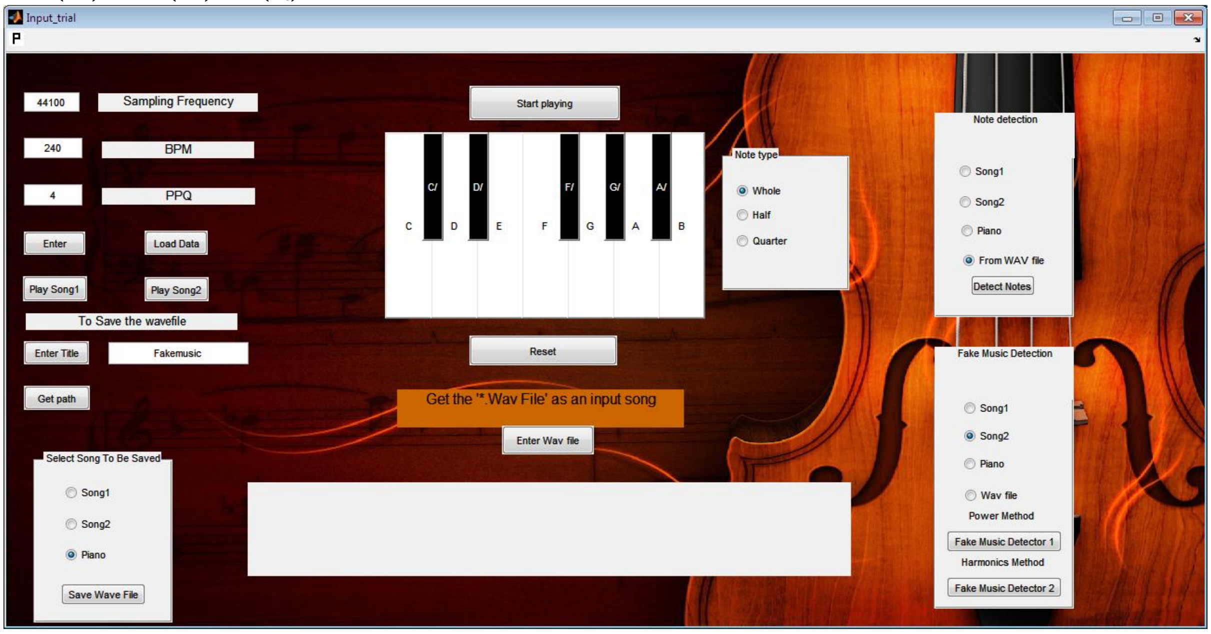


```
  238

  239

  240

  241

varnc =

  628.0773

Given tone is a real music
>>
```

Real Music Detected

```
(H)_A#3(H)_D#4(H)_D4(Q)_F4(Q)_F5(Q)_F4(H)_D5(Q)_D#4(H)_D#5(Q)_G3(H)_D#5(Q)_G3(Q)_G4(Q)_D#4(Q)_F4
(Q)_F5(Q)_F4(Q)_C5(Q)_F4(H)_A#5(Q)_A#4(Q)_F4(W)_F4(H)_G4(Q)_G3(Q)_G4(Q)_A#5(Q)_G3(Q)_G4(H)_D3(
Q)_E4(Q)_F4(Q)_C4(W)_C4(Q)_A#4(Q)_D#3(Q)_A#4(Q)_A#5(Q)_A#4(Q)_A#3(Q)_F5(Q)_C5(Q)_F3(Q)_F4(Q)_
C4(Q)_F4(Q)_C4(Q)_C3(Q)_C4(Q)_D4(Q)_F4(Q)_F5(Q)_F4(H)_D5(Q)_D#4(H)_D#5(H)_G3(Q)_D#5(H)_G4(Q)_D
#4(Q)_F4(H)_C6(Q)_F4(H)_A#4(Q)_C5(Q)_A#4(Q)_F4(Q)_A#5(Q)_A2(Q)_F4(W)_G4(Q)_G2(Q)_A#4(Q)_G4(H)_
D3(Q)_E4(Q)_F4(Q)_C4(H)_G4(Q)_C5(Q)_C4(Q)_A#4(H)_D4(Q)_F3(Q)_C4(H)_A#4(W)_A#4(H)_A#3(H)_D#4(Q)
```

Input given from piano keys {A4(Q)_B4(Q)_G5(Q)_C5(Q)_D5(Q)_E5(Q)_F5(Q)_A#4(W)_G#6(W)_F#5(W)_F#5(W)_C#5(W)_D#5(W)_D#5(W)_F1(Q)} and saved with name Fakemusic


Input_trial

44100 Sampling Frequency

240 BPM

4 PPQ

To Save the wavefile

Enter Title: Fakemusic

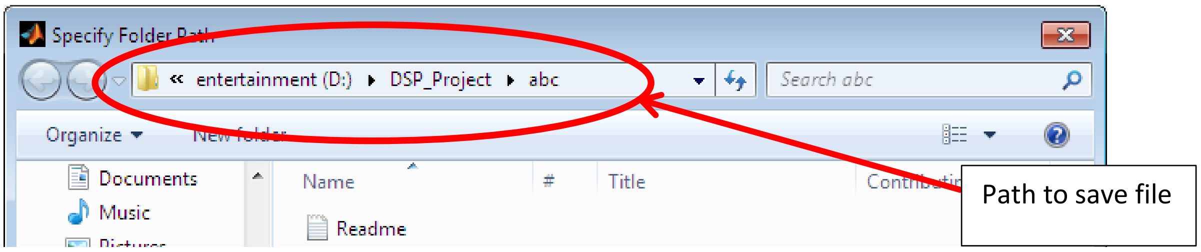
Get the '*.Wav File' as an input song

Select Song To Be Saved: Song1 Song2 [✓ Piano]

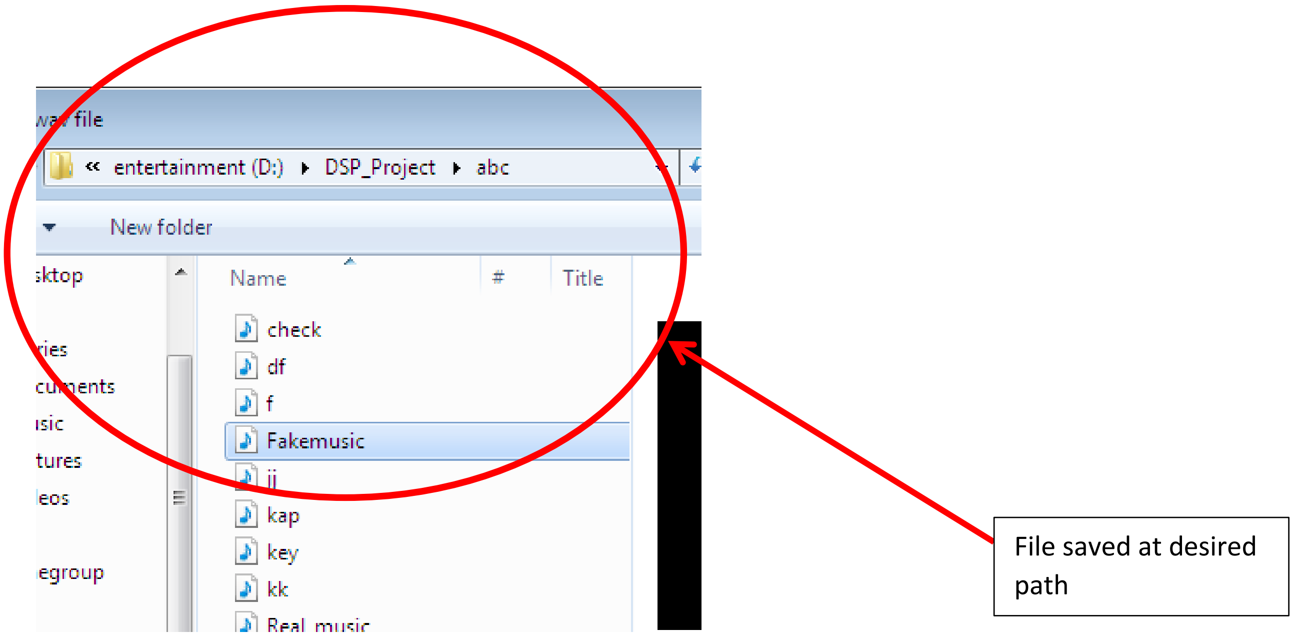
Note type: [✓ Whole] Half Quarter

Note detection: Song1 Song2 Piano [✓ From WAV file]

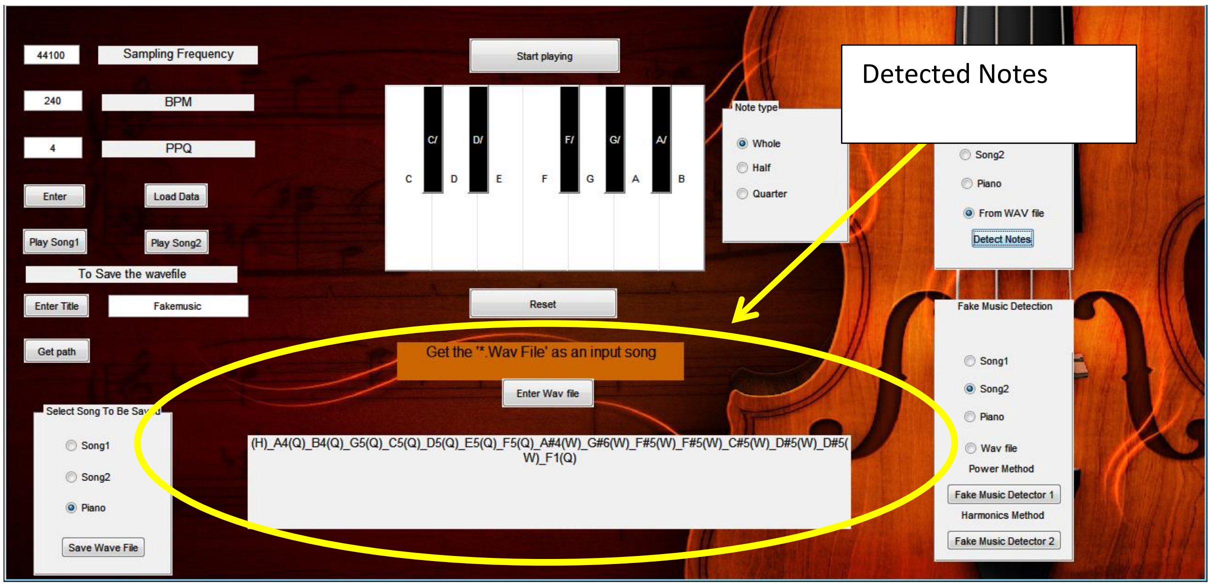
Fake Music Detection: Song1 [✓ Song2] Piano Wav file


Specify Folder Path

« entertainment (D:) ▸ DSP_Project ▸ abc


wav file

« entertainment (D:) ▸ DSP_Project ▸ abc



44100 Sampling Frequency

240 BPM

4 PPQ

To Save the wavefile

Enter Title: Fakemusic

Note type: [✓ Whole] Half Quarter

Song2 Piano [✓ From WAV file]

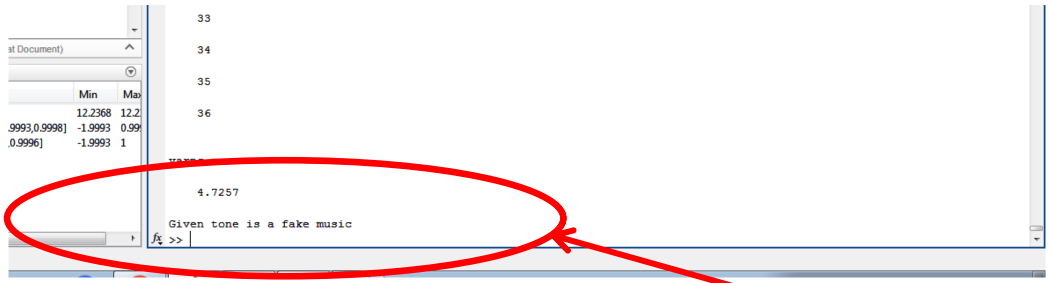
Get the '*.Wav File' as an input song

(H)_A4(Q)_B4(Q)_G5(Q)_C5(Q)_D5(Q)_E5(Q)_F5(Q)_A#4(W)_G#6(W)_F#5(W)_F#5(W)_C#5(W)_D#5(W)_D#5(W)_F1(Q)

Select Song To Be Saved: Song1 Song2 [✓ Piano]

Fake Music Detection: Song1 [✓ Song2] Piano Wav file

Detected Notes



```
at Document)
                    Min       Max
                    12.2368   12.2
.9993,0.9998]       -1.9993   0.99
,0.9996]            -1.9993   1
```

```
    33

    34

    35

    36

varp

    4.7257

Given tone is a fake music
>>
```

Fake music detected

Appendix

Table relating piano key frequencies with the notes

88	**C8**	4186.01
87	**B7**	3951.07
86	**A♯7/B♭7**	3729.31
85	**A7**	**3520**
84	**G♯7/A♭7**	3322.44
83	**G7**	3135.96
82	**F♯7/G♭7**	2959.96
81	**F7**	2793.83
80	**E7**	2637.02
79	**D♯7/E♭7**	2489.02
78	**D7**	2349.32
77	**C♯7/D♭7**	2217.46
76	**C7**	2093
75	**B6**	1975.53
74	**A♯6/B♭6**	1864.66
73	**A6**	**1760**
72	**G♯6/A♭6**	1661.22
71	**G6**	1567.98
70	**F♯6/G♭6**	1479.98
69	**F6**	1396.91
67	**D♯6/E♭6**	1244.51
65	**C♯6/D♭6**	1108.73
64	**C6**	1046.5
63	**B5**	987.767
27	**B2**	123.471
26	**A♯2/B♭2**	116.541
25	**A2**	**110**
24	**G♯2/A♭2**	103.826
23	**G2**	97.9989
22	**F♯2/G♭2**	92.4986
21	**F2**	87.3071
20	**E2**	82.4069
19	**D♯2/E♭2**	77.7817
18	**D2**	73.4162
17	**C♯2/D♭2**	69.2957
16	**C2**	65.4064
15	**B1**	61.7354
14	**A♯1/B♭1**	58.2705
13	**A1**	**55**
12	**G♯1/A♭1**	51.9131
10	**F♯1/G♭1**	46.2493
9	**F1**	43.6535
8	**E1**	41.2034
7	**D♯1/E♭1**	38.8909
6	**D1**	36.7081
5	**C♯1/D♭1**	34.6478
4	**C1**	32.7032
3	**B0**	30.8677
2	**A♯0/B♭0**	29.1352
1	**A0**	**27.5**

References

[1] DSP first-A multimedia Approach by James H. McClellan.

[2] ADSR envelope. http://en.wikipedia.org/wiki/ADSR_envelope

[3] https://en.wikipedia.org/wiki/Piano_key_frequencies

[4] http://we15hang.blogspot.in/2012/02/matlab-gui-inserting-background-image.html

[5] https://userscontent2.emaze.com/images/d88f5fe6-f12f-4561-ab5a-89432717b148/5083aab6-f174-42a5-8c52-e2d273027c35.jpg

[6] https://www.clear.rice.edu/elec301/Projects01/beat_sync/beatalgo.html

[7] https://web.eecs.umich.edu/~fessler/course/100/p/project1.pdf

[8] http://www.songsofthecosmos.com/encyclopedia_of_modern_music/A/ADSR.html