Lab 1 Part 1

Table of Contents

1.1 - A Function to Play a Note

```
type key_to_note.m
```

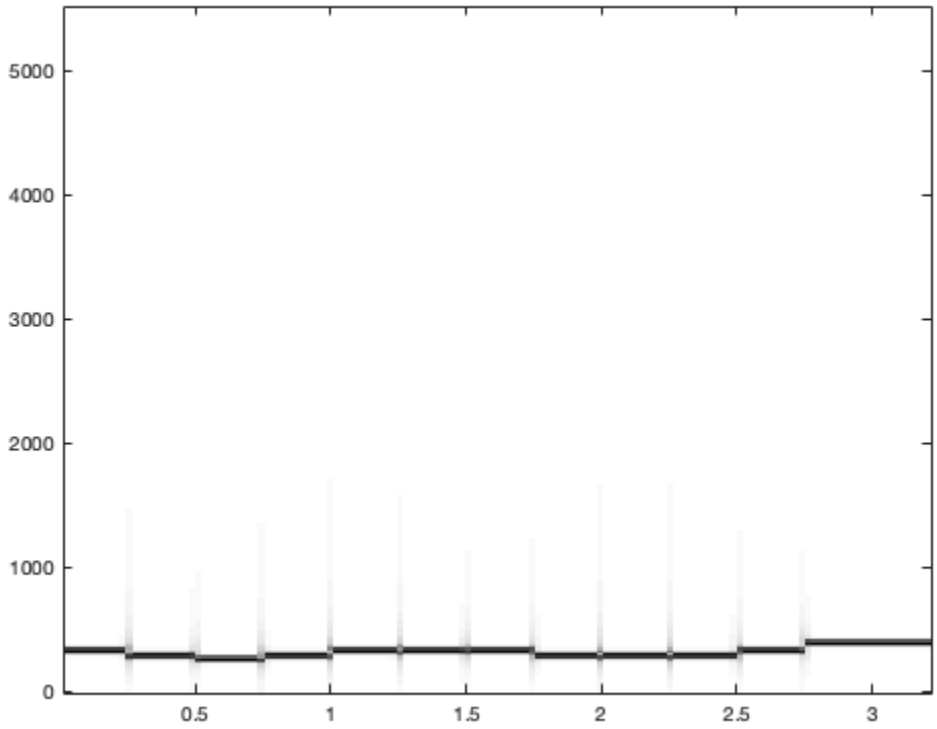
```
function xx = key_to_note(X, keynum, dur)
    %{
    KEY_TO_NOTE: Produce a sinusoidal waveform corresponding to a
 given piano key number Input Args:
    -X: amplitude (default = 1)
    -keynum: number of the note on piano keyboard -dur: duration of
 the note (in seconds)
    Output:
    -xx: sinusoidal waveform of the note
    %}
    fs = 11024;
    tt = 0:(1/fs):dur-1/fs;
    freq = 440 * (   2^( (keynum-49)/12 )   );
    xx = real( X*exp(j*2*pi*freq*tt) );
end
```

1.2 - Synthesize a Song - Mary Had a Little Lamb that NEVER Grew Up!

```
type play_mary.m
play_mary
```

```
% --------------play_mary.m-------------- %

%Notes: C D E F G
%Key #40 is middle-C
mary.keys = [44 42 40 42 44 44 44 42 42 42 44 47 47];
mary.durations = 0.25 * ones(1,length(mary.keys));
fs = 11024; % 11025 Hz also works
xx = zeros(1, sum(mary.durations)*fs);
n1 = 1;
```

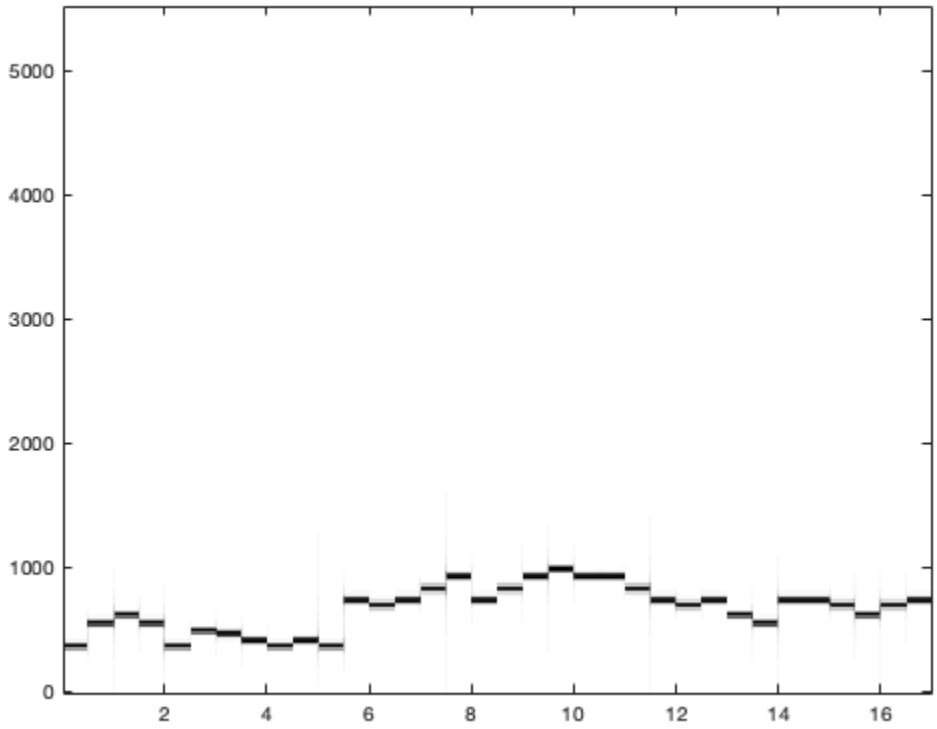
```
for kk = 1:length(mary.keys)
    keynum = mary.keys(kk);
    tone = key_to_note(1, keynum, mary.durations(kk));
    n2 = n1 + length(tone) - 1;
    xx(n1:n2) = xx(n1:n2) + tone;
    n1 = n2 + 1;
end

soundsc(xx,fs);
plotspec(xx,fs,512);
audiowrite("mary.wav", xx, fs)
```



1.4 - The Evenly-Timed First Voice

```
type play_first_voice_even
play_first_voice_even


% --------------play_first_voice_even.m-------------- %
load barukh_fugue

fs = 11024; % 11025 Hz also works
n1 = 1;

durations = 0.5 * ones(1,length(theVoices(1).noteNumbers));
xx = zeros(1, ceil(fs*sum(durations)));
```

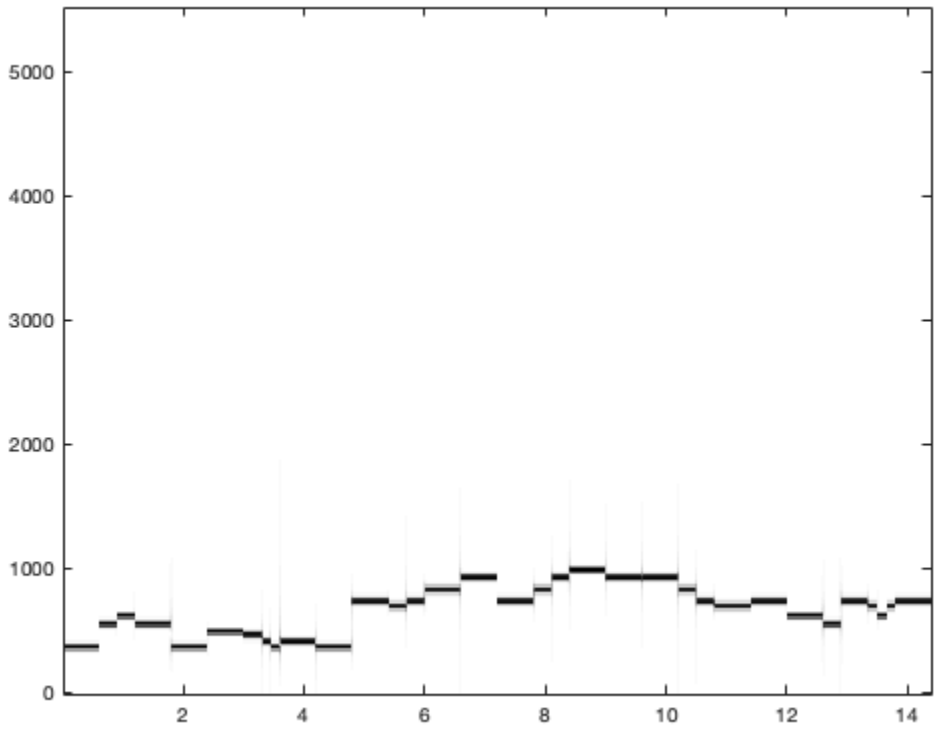
```
for kk = 1:length(theVoices(1).noteNumbers)
    keynum = theVoices(1).noteNumbers(kk);
    tone = key_to_note(1, keynum, durations);
    n2 = n1 + length(tone) - 1;
    xx(n1:n2) = xx(n1:n2) + tone;
    n1 = n2 + 1;
end

soundsc(xx,fs);
plotspec(xx,fs,512);
audiowrite("first_voice_even.wav", xx, fs)
```



1.5 - The Correctly-Timed First Voice

```
type play_first_voice
play_first_voice


% --------------play_first_voice.m-------------- %
load barukh_fugue

fs = 11024; % 11025 Hz also works
n1 = 1;

xx = zeros(1, ceil(fs*0.15*sum(theVoices(1).durations)));
```

```
for kk = 1:length(theVoices(1).noteNumbers)
    keynum = theVoices(1).noteNumbers(kk);
    tone = key_to_note(1, keynum, 0.15*theVoices(1).durations(kk));
    n2 = n1 + length(tone) - 1;
    xx(n1:n2) = xx(n1:n2) + tone;
    n1 = n2 + 1;
end

soundsc(xx,fs);
plotspec(xx,fs,512);
audiowrite("first_voice.wav", xx, fs)
```



1.6 - Silence and startPulses: Construction of the Better Fugue

```
type playSong

load better_fugue.mat
song = playSong(theVoices);
audiowrite('better_fugue.wav',song/max(song),11024);

load barukh_fugue.mat
song = playSong(theVoices);
audiowrite('barukh_fugue.wav',song/max(song),11024);
```

```
function song = playSong(theVoices)
    %{
    PLAYSONG: Produce a sinusoidal waveform containing the combination
 of the different notes in theVoices
    Input Args:
    -theVoices: structure contains noteNumbers, durations, and
 startpulses vectors for multiple voices.
    Output:
    -song: vector that represents discrete-time version of a musical
 waveform
    Usage: song = playSong()
    %}

    fs = 11024;
    bpm = 120;
    bps = bpm / 60;
    spb = 1 / bps;
    spp = spb / 4; %seconds per pulse, theVoices is measured in pulses
 with 4 pulses per beat

    maxIndex = 1;
    for i = 1:length(theVoices)
        lastNoteNumber = length(theVoices(i).noteNumbers);
        lastNote = key_to_note(0.5,
 theVoices(i).noteNumbers(lastNoteNumber),
 theVoices(i).durations(lastNoteNumber)*spp);
        currentIndex = spp*fs*theVoices(i).startPulses(lastNoteNumber)
 + length(lastNote) - 1;
        if currentIndex > maxIndex
            maxIndex = currentIndex;
        end
    end

    song = zeros(1, maxIndex); %Create a vector of zeros with length
 equal to the total number of samples in the entire song

    %Then add in the notes
    for i = 1:length(theVoices)
        for j = 1:length(theVoices(i).noteNumbers)
            keynum = theVoices(i).noteNumbers(j);
            dur = theVoices(i).durations(j)*spp;
            note = key_to_note(0.5, keynum, dur); %Create sinusoid of
 correct length to represent a single note
            locstart = spp*fs*theVoices(i).startPulses(j); % Index of
 where note starts
            locend = locstart + length(note) - 1; % index of where
 note ends
            song(locstart:locend) = song(locstart:locend) + note;
        end
    end
    song = song/max(song);
    soundsc(song, fs);
end
```

Published with MATLAB® R2019a