Taylor Rembos
EEL 3135
Lab 3 Part 2

Lab 3: Music Synthesis with Sinusoidal Signals

Lab Part Two

2.1 Construction of the Bach Fugue

'Bach_fugue_3voices.wav' file
The MATLAB data file bach_fugue.mat is of the same structure as barukh_fugue.mat, also containing a theVoices array of structures. Add the three voices from theVoices in bach_fugue.mat, together to produce the Bach Fugue. Have your TA check this off. (Alternately, if you can't finish this in time, submit as a *.wav* file on Canvas.)

function playsong2.m

```
function song = playSong(theVoices)
% PLAYSONG Produce a sinusoidal waveform containing the combination
ofthe different notes in theVoices
% usage: song = playSong ()
% song = the output sinusoidal waveform
load('bach_fugue.mat')
fs = 8000;
spp = 0.25 %%% seconds per pulse, theVoices is measured in pulses with
4 pulsesper beat

% Create a vector of zeros with length equal to the total number
ofsamples in the entire song
song =
zeros(1,4*spp*fs*(theVoices(3).durations(length(theVoices(3).durations)
) + (theVoices(3).startPulses(length(theVoices(3).durations))))); %%%
vector of zeros

% Then add in the notes
    for i = 1:length(theVoices) % Cycle through each set of notes
        % Convert data arrays to appropriate units
        for j = 1:length(theVoices(i).noteNumbers) % Cycle througheach
note in aset
            keynum = theVoices(i).noteNumbers(j);
            note = key2note(1,keynum,theVoices(i).durations(j).*spp);
%%% create sinusoid of correct length to represent a single note
            locstart = theVoices(i).startPulses(j)*(fs/4)+1; %%% 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 (abs (song))); %% anti clipping xx
    soundsc(song,fs);
    audiowrite('Bach_fugue_3voices.wav',song,fs);
% Use audiowrite() to generate WAV file
end
```

2.2 Musical Tweaks - Enveloping

'bach_envelope.wav' file

Your job here is to improve the sound quality.

Hint 1: The function interp1()may be useful in this section.
Hint 2: You'll want to make sure that your note and E are the same length.
Hint 3: Previous generations of students have found the Hanning Window to be an effective envelope.

Implement an envelope to improve the sound of your Bach Fugue. Save it as *bach_envelope.wav* and submit it on Canvas.

script key2noteReal.m

```
function xx = key2noteReal(X,keynum,dur)
    % KEY2NOTE Produce a sinusoidal waveform corresponding to a given
piano key number
    % usage: xx = key2note (X, keynum, dur)
    % xx = the output sinusoidal waveform
    % X = complex amplitude for the sinusoid, X = A*exp(j*phi).
    % keynum = the piano keyboard number of the desired note
    % dur = the duration (in seconds) of the output note
% Hint 1: The function interp1()may be useful in this section.?
% Hint 2: You?ll want to make sure that your note and E are the same
length.
% Hint 3: Previous generations of students have found the Hanning
Window to be an effective envelope.

fs = 8000;
tt = (1/fs):(1/fs):dur;
freq = 440*(2^((keynum-49)/12));
a = 50;
b = 20;
c = 1.4285;
e = interp1([0,dur/a,dur/b,dur/c,dur], [0,1,0.8,0.02,0],tt,'spline');
% interp1 interpolation interp1(X,V,Xq), X vector (X=N), V vector
(length
% N), Xq array of size M
% interpolates to find Vq, the values of teh underlying fuction V=F(X)
at
% query points Xq
xx = e.*real(X*exp(j*2*pi*freq*tt));
```

function playSong2.m

```
function song = playSong(theVoices)
% PLAYSONG Produce a sinusoidal waveform containing the combination
ofthe different notes in theVoices
% usage: song = playSong ()
% song = the output sinusoidal waveform
load('bach_fugue.mat')
fs = 8000;
spp = 0.25 %%% seconds per pulse, theVoices is measured in pulses with
4 pulsesper beat
```

```
% Create a vector of zeros with length equal to the total number
ofsamples in the entire song
song =
zeros(1,4*spp*fs*(theVoices(3).durations(length(theVoices(3).durations)
) + (theVoices(3).startPulses(length(theVoices(3).durations))))); %%%
vector of zeros

% Then add in the notes
    for i = 1:length(theVoices) % Cycle through each set of notes
        % Convert data arrays to appropriate units
        for j = 1:length(theVoices(i).noteNumbers) % Cycle througheach
note in aset
            keynum = theVoices(i).noteNumbers(j);
            note =
key2noteReal(1,keynum,theVoices(i).durations(j).*spp); %%% create
sinusoid of correct length to represent a single note
            locstart = theVoices(i).startPulses(j)*(fs/4)+1; %%% 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 (abs (song))); %% anti clipping xx
    soundsc(song,fs);
    audiowrite('Bach_envelope.wav',song,fs);
% Use audiowrite() to generate WAV file
end
```

2.3 Musical Tweaks – Fourier Series of a Trumpet

Suppose the maximum frequency in the Bach Fugue is 1200 Hz. What is the minimum sampling frequency needed to synthesize, without aliasing, a trumpet sound containing nine harmonics?

2*1200 Hz = 2400 Hz

Using a sampling frequency of 44100 Hz, construct your Bach Fugue in trumpet. Submit this as a *.wav* file on Canvas.

'bach_trumpet.wav' file

script key2noteReal.m

```
function xx = key2noteReal(X,keynum,dur)
    % KEY2NOTE Produce a sinusoidal waveform corresponding to a given
piano key number
    % usage: xx = key2note (X, keynum, dur)
    % xx = the output sinusoidal waveform
    % X = complex amplitude for the sinusoid, X = A*exp(j*phi).
    % keynum = the piano keyboard number of the desired note
    % dur = the duration (in seconds) of the output note
% Hint 1: The function interp1()may be useful in this section.?
% Hint 2: You?ll want to make sure that your note and E are the same
length.
% Hint 3: Previous generations of students have found the Hanning
Window to be an effective envelope.

fs = 44100;
tt = (1/fs):(1/fs):dur;
freq = 440*(2^((keynum-49)/12));
a = 50;
b = 20;
c = 10/7;
e = interp1([0,dur/a,dur/b,dur/c,dur], [0,1,0.8,0.02,0],tt,'spline');
% interp1 interpolation interp1(X,V,Xq), X vector (X=N), V vector
(length
% N), Xq array of size M
% interpolates to find Vq, the values of teh underlying fuction V=F(X)
at
% query points Xq
% s = spline( x , y , xq ) returns a vector of interpolated values s
corresponding to the query points in xq
har1 = 0.1155*real(X*exp(j*1*2*pi*freq*tt)*exp(j*(-2.199)));
har2 = 0.3417*real(X*exp(j*2*2*pi*freq*tt)*exp(j*1.6727));
har3 = 0.1789*real(X*exp(j*3*2*pi*freq*tt)*exp(j*(-2.5454)));
har4 = 0.1232*real(X*exp(j*4*2*pi*freq*tt)*exp(j*0.6607));
har5 = 0.0678*real(X*exp(j*5*2*pi*freq*tt)*exp(j*(-2.0390)));
har6 = 0.0473*real(X*exp(j*6*2*pi*freq*tt)*exp(j*2.1597));
har7 = 0.0260*real(X*exp(j*7*2*pi*freq*tt)*exp(j*(-1.0467)));
har8 = 0.0045*real(X*exp(j*8*2*pi*freq*tt)*exp(j*1.8581));
har9 = 0.0020*real(X*exp(j*9*2*pi*freq*tt)*exp(j*(-2.3925)));
xx = e.*(har1+har2+har3+har4+har5+har6+har7+har8+har9);
```

function playSong2.m

```
function song = playSong(theVoices)
% PLAYSONG Produce a sinusoidal waveform containing the combination
ofthe different notes in theVoices
% usage: song = playSong ()
% song = the output sinusoidal waveform
load('bach_fugue.mat')
fs = 44100;
spp = 0.25 %%% seconds per pulse, theVoices is measured in pulses with
4 pulsesper beat

% Create a vector of zeros with length equal to the total number
ofsamples in the entire song
song =
zeros(1,4*spp*fs*(theVoices(3).durations(length(theVoices(3).durations)
) + (theVoices(3).startPulses(length(theVoices(3).durations))))); %%%
vector of zeros

% Then add in the notes
    for i = 1:length(theVoices) % Cycle through each set of notes
        % Convert data arrays to appropriate units
        for j = 1:length(theVoices(i).noteNumbers) % Cycle througheach
note in aset
            keynum = theVoices(i).noteNumbers(j);
            note =
key2noteReal(1,keynum,theVoices(i).durations(j).*spp); %%% create
sinusoid of correct length to represent a single note
            locstart = theVoices(i).startPulses(j)*(fs/4)+1; %%% 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 (abs (song))); %% anti clipping xx
    soundsc(song,fs);
    audiowrite('Bach_trumpet.wav',song,fs);
% Use audiowrite() to generate WAV file
end
```

Extra Credit is coming…