Music 421A
Spring 2019-2020
Homework #8
Filter Banks
One week assignment

Theory Problems

1. (10 pts) Suppose the window transform $W(\omega)$ is a *lowpass filter* with cut-off frequency $\omega_c = 2\pi/R$ and infinite side-lobe suppression. That is, $W(\omega) = 0$ for $|\omega| \geq \omega_c$.

   (a) (5 pts) In this case, show that

   $$\sum_{m=-\infty}^{\infty} w(n - mR) = \frac{1}{R} W(0)$$

   irrespective of the shape of w or the shape of $W(\omega)$ in the interval $(-\omega_c, \omega_c)$.

   (b) (5 pts) Specify the set of useable frame step sizes R' such that

   $$\sum_{m=-\infty}^{\infty} w(n - mR') = \text{constant}.$$

2. (10 pts) **Constant-Overlap-Add Condition**

   (a) (5 pts) For windows in the 1-, 2-, and 3-term Blackman-Harris families, (*i.e.*, rectangular, generalized Hamming, and Blackman family), use the Poisson Summation Formula to determine the set of all hop-size values giving constant overlap-add.

   (b) (5 pts) Why does the Kaiser window not overlap-add to a constant exactly for $R > 1$? What range of hop sizes should be used and why? [Characterize the valid hop sizes in terms of one or more spectral properties of the window. The matlab function `ckola.m`[1] may be used to check your conclusions.]

[1] `http://ccrma.stanford.edu/~jos/sasp/hw/ckola.m`

Lab Assignment

1. (30 pts) In this problem, you will implement a sinusoidal modeling framework that takes any quasi-periodic signal, tracks its frequency and amplitude, and resynthesizes the signal using a bank of sinusoidal oscillators. You should submit the original, resynthesized, and pitch-shifted signals along with your code.

   (a) (10 pts) You will use the `findpeaks.m` function you wrote in Homework 5 to get the frequencies and amplitudes of a given audio signal. Implement a matching algorithm in `create_partial_tracks.m` that creates frequency and amplitude tracks across successive frames. This is known as **partial tracking**. For more information, see papers by McAulay and Quatieri[2] and/or Serra and Smith.[3] The pseudo-code given in `create_partial_tracks.m` implements the MQ algorithm.

```
function [freq_tracks,amp_tracks] = create_partial_tracks(freqs,peaks,delta)

% create partial tracks according to McAulay-Quarieri algorithm
% freqs - detected frequency peaks
% peaks - complex amplitudes of detected peaks
% delta - margin of frequency difference allowed in Hz
% RETURNS :
% freq_tracks - matrix of frequency tracks (nframes x npeaks)
% amp_tracks - matrix of amplitude tracks (nframes x npeaks)

[nframes,nfreqs] = size(freqs);
for k = 1:nframes-1
for n = 1:nfreqs
cur_freq = freqs(k,n);

%-- Step 1 : check if current track is dead, i.e, check if
%abs(cur_freq - freqs(k+1,:) >= delta). If track is dead,
%frequency is matched with itself in next frame and amplitude
%set to 0. If on the other hand, there exists a frequency in
%the next frame k+1, freqs(m,k+1), that lies within the matching
%interval about current frequency, and is the closest frequency,
%then it is declared to be a candidate match --%


%--Step 2 :  check if candidate match is better matched to any of
%the remaining unmatched frequencies of frame k. If there is no
%better match, finalize candidate and eliminate it from further
%consideration. If there is a better match, then the candidate
%match frequency is better matched to the frequency freqs(k,n+1).
```

[2] https://ieeexplore.ieee.org/abstract/document/1164910
[3] `https://ccrma.stanford.edu/~jos/parshl/`

```
%There can be two cases :

%--Case 1 :  freqs(k,n)-freqs(k+1,m-1) lies below matching
%interval => track is marked dead, frequency is matched to itself
%and amplitude matched to 0.

%--Case 2 : or else freqs(k+1,m-1) is finalized as a match
%to freqs(k,n) --%


%--Step 3 : when all frequencies of frame k gave been tested
%and assigned to continuous tracks or dying tracks, there may
%remain frequencies in frame k+1 for which no matches have been
made. Then those frequencies are considered to be "born" in frame
k, and a new frequency is created in frame k with zero amplitude --%

end
```

(b) (5 pts) Complete the code in `additive_synthesis.m` to return a sum of real sinusoids given their frequencies and amplitudes. Use linear interpolation to smooth each amplitude and frequency trajectory from one frame to another.

```
function [x] = additive_synthesis(amp_lims,freq_lims,M,fs)

% amp_lims - amplitudes in current and next frame (npeaks x 2)
% freq_lims - frequencies in current and next frame (npeaks x 2)
% M - window length
% fs - sampling rate (Hz)
% RETURNS:
% x - sum of npeaks sinusoids of length M samples
...
end
```

(c) (10 pts) For the signal bird_chirps.wav[4], using a Hann window of length $M = 1023$ samples and a hopSize of $R = \frac{M+1}{8}$, break the signal into overlapping frames and find the peak frequencies and amplitudes in each frame with $maxPeaks = 20$. Zero pad to a factor of 4 for FFT computation. Find the frequency peaks and their amplitudes in each frame using `findpeaks.m`. Link the found peaks using `create_partial_tracks.m`. Resynthesize the signal with `additive_synthesis.m` using overlap-add.

(d) (2 pts) Now pitch shift the original signal up by an octave and resynthesize it.

(e) (3 pts) Change R, M and $maxPeaks$ and listen to how different the resynthesized signals sound. How does each affect analysis?

[4] `http://ccrma.stanford.edu/~jos/hw421/hw8/bird_chirps.wav`