Assignment: Homework 3

What to Hand In

1. Your code with appropriate comments and a **readme file**. You are responsible for making us understand and know how to run your code.
2. A report with answers, figures, and explanations.
3. Any sound files asked for in the problems.

How to Hand It In

1. Put all your files in one folder. Compress this folder and name it <firstname>_<lastname>_HW3.zip. For example, "Zhiyao_Duan_HW3.zip".
2. Submit via blackboard to the appropriate entry**.**

When to Hand It In

It is due immediately at the beginning of the day (i.e., 0:00AM) specified on the course calendar. **Late assignments will receive a grade deduction of 20% of the full grade each day.**

Problems (10 points in total)

1. (3 points) Windowing, zero padding
    a. (0.5 points) Generate a discrete-time rectangular window with length of M=101 points. Plot its magnitude spectrum and the phase spectrum. The horizontal axis should be **normalized angular frequency in** $[-\pi, \pi]$. The vertical axes should be **magnitude in dB and wrapped phase**, respectively. Label the axes appropriately. The figures may not look correct, because there are too few points to represent the spectra.
    b. (0.5 points) Zero-pad your rectangular window with different times of zero padding (e.g., 2, 5, 10, 50, 100, 500) and plot the magnitude and phase spectra. You should see the details of the spectra better and better.
    c. (0.5 points) Change the length of the rectangular window (e.g. M = 11, 51, 101, 501, 1001) and apply 100 times zero padding. Plot the magnitude and phase spectra. You should see that the main lobe (and side lobes) are getting narrower and narrower. You might need to zoom in the spectra when you use longer windows.
    d. (0.5 point) Generate a Hamming window and a Hann window with a length of 101 points, apply 100 times zero-padding, and plot the magnitude spectrum. Compare their magnitude spectra with the rectangular window magnitude spectra. Discuss the changes of main lobe width, side lobe height, and side lobe roll off rate.
    e. (0.5 points) Generate a 2-second long mixture signal x[n] with sampling rate of 8000 Hz. It is a summation of two sinusoids with the same peak amplitude and initial phase. The frequency of the first sinusoid is 1000 Hz, and the frequency of the second sinusoid is 1010 Hz. According to the ideal time-frequency resolution tradeoff, if we take 100 ms long chunk of the signal for Fourier analysis, the sinusoids should be resolved. Try to take a 100 ms long chunk of the signal and apply a rectangular window, then zero-pad the windowed signal with 100 times zero

padding, and perform FFT to see the magnitude spectrum. The horizontal axis should be **frequency in Hz**, while the vertical axis should be **magnitude in dB**. Label the axes appropriately. It's OK to only show the positive frequency range. Are the sinusoids resolved?

f. (0.5 points) Vary the window length to find the minimum length that you feel the two sinusoids are resolved for the rectangular window. Now repeat the experiment with Hamming and Hann windows. Vary the window length to find the minimum lengths that you feel the two sinusoids are resolved. Do you see a good match between the values you found and the theoretical values of the main lobe width for these windows derived in class?

2. (3 points) Short-time Fourier Transform (STFT) and spectral peak interpolation
   a. (0.5 points) Read the attached "violin.wav" into Matlab. The sampling rate of the signal is 22,050 Hz. Apply Short-time Fourier Transform (STFT) with Hamming window with length of 1024 points (i.e., about 46ms) and hop size of 512 points (i.e., about 23 ms), and 4 times zero padding. Stack all the magnitude (in dB) spectra (only the positive frequency range) into a spectrogram. The spectrogram should be a matrix with size of 2049 * #Frames, where 2049 is the number of bins in the positive frequency range. Display the magnitude spectrogram using the "imagesc" function. Label the horizontal axis **in time (second)** and vertical axis **in frequency (Hz)**.
   b. (1 point) Write a function that can detect significant peaks in the logarithmic-magnitude spectrum. By "significant peaks", I mean local maxima whose magnitude is high enough (e.g., higher than the global maximum of the spectrum minus 30dB). The peaks generally correspond to the harmonics of the signal. The syntax of the function is as follows:

```
function peak = PeakExtract(MagSpec, MagTh)
% Peak detection from magnitude spectrum
% Input
%   - MagSpec   : magnitude spectrum measured in dB(only
contains the positive frequency part)
%   - MagTh     : the absolute threshold (peaks above the
maximum amplitude minus this value are considered)
% Output
%   - peak      : frequency bins of detected peaks
```

   c. (0.5 points) Write a function to perform quadratic interpolation of the peaks detected by your peak detection function.

```
function [peak, peakAmp] = PeakInterpolate(MagSpec, peak)
% Quadratic interpolation
% Input
%   - MagSpec           : magnitude spectrum measured in dB
%   - peak              : frequency bins of detected peaks
% Output
%   - peak              : interpolated peak frequencies (in
frequency bins)
%   - peakAmp           : interpolated peak amplitudes (in dB)
```

d. (1 point) Plot the magnitude spectrum of the 20-th frame, with interpolated peaks marked by red circles. What are the frequencies of the peaks? Since most peaks correspond to the harmonics of the fundamental frequency, can you infer the fundamental frequency?

3. (4 points) Analysis, modification and re-synthesis through STFT
   a. (1 point) Zero the **linear-amplitude spectrogram** outside the frequency range [300Hz, 3400Hz]. This is the frequency range that is used in telephone signals. Plot the modified log-amplitude spectrogram to verify your processing. Convert the new spectrogram back to the time domain signal using Overlap-Add technique. Save the time domain signal as "violin_telephone.wav". Does it sound like a voice coming out of a telephone?
   b. (0.5 points) Now let's try to change the speed of the violin sound. The naive way is to resample the sound (i.e., change the sampling rate) but save it with the original sampling rate. For example, the original sampling rate of the violin signal was 22,050 Hz, and there were in total 276,220 samples in the piece. If we use "resample" function to change its sampling rate to 11,025 Hz, only half of the samples will remain. Now if we save the new signal with the original sampling rate 22,050 Hz to the hard drive and play it back, the length of the violin signal is halved (i.e., its tempo is doubled). Use this method to generate and save a tempo-doubled signal as "violin_resample.wav". Listen to it. Do you feel the tempo is doubled? Do you feel that the pitch is changed?
   c. (2.5 points) Now we want to change the speed of the audio signal without changing the frequency. In fact, we want to make speed change and pitch change independent to each other. This is the technique called phase vocoder. Write a function that allows one to modify the speed and the pitch of the signal separately. The syntax is as follows:

```
function y = SpeedPitchChange(x, fs, alpha, beta)
% Independently modify speed and pitch of the audio
%
% Input
%   - x         : the input audio data
%   - fs        : sampling rate
%   - alpha     : speed factor (alpha = new speed / old speed)
%   - beta      : pitch factor (beta = new pitch / old pitch)
% Output
%   - y         : the output audio data, with the same
sampling rate but new speed and new pitch
```

   The general process is as follows:
   1. (0.5 points) resample the input audio to achieve wanted pitch change
   2. (0.5 points) calculate the spectrogram of the resampled audio (using an appropriate frame length, hop, and window function)

3. (1 point) interpolate the magnitude spectrogram and adjust the phase spectrogram to achieve wanted speed change
4. (No point) convert the interpolated spectrogram back to time domain with overlap-add technique
5. (0.5 points) Test your function on the violin signal with parameters alpha = 1.5 (i.e., new speed is 1.5 times as fast as before) and beta = 2^(2/12) (i.e., new pitch is 2 semitones higher than before). Save your modified signal as "violin_phasevocoder.wav". Enjoy your "career" as a DJ!