Full Name: ____

EEL 4750 / EEE 5502 (Fall 2018) – Code #06 **Due Date:** **Nov. 20, 2018**

Question #1: *(1 pts)* How many hours did you spend on this homework?

Question #2: *(6 pts)* *Filters and Windows*

Consider three different systems, defined by the following equations:

$$\text{System 1:}\quad Y[k] = \begin{cases} 0 & \text{for } |Y[k]| < 20 \\ Y[k] & \text{for otherwise} \end{cases}$$

$$\text{System 2:}\quad y[n] = \frac{1}{10}\sum_{m=0}^{9} x[n-m]$$

$$\text{System 3:}\quad y[n] = \frac{1}{2}y[n-100] + \frac{1}{2}x[n]$$

(a) What does system 1 do?

(b) What type of filter is system 2? High-pass, low-pass, band-pass, or other?

(c) Verify the behavior of system 2 by plotting the approximate DTFT with

```
plot((0:9999)/10000*2*pi, abs(fft(h),10000))
```

where `h` is the impulse response the signal. The second input for `fft` adds zeros to the end of the input signal before computing the FFT, making frequency-domain resolution more fine.

(d) Sketch its pole-zero plot for system 3.

(e) What type of filter is system 3? High-pass, low-pass, band-pass, or other?

(f) Practically, speaking what does system 3 do?

Question #3: *(7 pts)* *The Inverse STFT*

In the previous coding assignment, you created the short-time Fourier transform (STFT). The STFT is a widely used tool for audio signal processing. The benefit of the STFT is that we can more independently alter frequency and time. We will now work with this transform some more.

Attached with this assignment is a `STFT_func` MATLAB function. This function computes the STFT of the signal $x[n]$, modifies the STFT (or will after this assignment – currently it does nothing), and then computes the inverse STFT to get an output $y[n]$. In this problem, you will be implementing a system at these lines:

```
% ***** PERFORM PROCESSING HERE AND ASSIGN ySTFT *****
ySTFT(:,m) = xSTFT(:,m);                    % REPLACE THIS
% **************************************************
```

For each part below, plot the magnitude of the STFT for the original music signal *and* the modified music signal. Use `axis` to zoom-in on same relevant (i.e., non-zero) information for each part.

(a) Load `chiptune_noise.wav`. Implement system 1 from Question #2 across the frequency domain for each frame and plot the STFT. How does the audio change?

(b) Load `chiptune_normal.wav`. Implement system 2 from Question #2 over the time domain of the STFT and plot the STFT. Do **not** use convolution or the DFT/FFT to solve this. Solve the difference equation directly. How does the audio change?

(c) Load `chiptune_noaudio.wav`. Implement system 3 from Question #2 over the time domain of the STFT and plot the STFT. Do **not** use convolution or the DFT/FFT to solve this. Solve the difference equation directly. How does the audio change?

Question #4: *(7 pts)* *Overlap-Add*
You may have noticed choppiness in the reconstructed time signal. This is because there is no smooth transition from one frame to another. In this question, we will fix this problem. To remove the choppiness, change the `STFT_func` function such that:

- Each time you compute the FFT, shift the frame by $W/2$ instead of W. Hence, each frame will have a 50% overlap with an adjacent frame.
- After computing each IFFT, multiply the length-W time-domain signal with a Hann window
$$w[n] = \frac{1}{2}\left[1 - \cos\left(\frac{2\pi n}{W-1}\right)\right] .$$
This creates a smooth transition between frames.
- After computing each IFFT and multiplying each frame by a Hann window, sum the overlapping components of the adjacent frames.

This process is known as overlap-add.

For each part below, plot the magnitude of the STFT for the original music signal *and* the modified music signal. Use `axis` to zoom-in on same relevant (i.e., non-zero) information for each part.

(a) Load `chiptune_noise.wav`. With your new overlap-add STFT, implement system 1 from Question #2 across the frequency domain for each frame and plot the STFT. How does the audio change?

(b) Load `chiptune_normal.wav`. With your new overlap-add STFT, implement system 2 from Question #2 over the time domain of the STFT and plot the STFT. Do **not** use convolution or the DFT/FFT to solve this. Solve the difference equation directly. How does the audio change?

(c) Load `chiptune_noaudio.wav`. With your new overlap-add STFT, implement system 3 from Question #2 over the time domain of the STFT and plot the STFT. Do **not** use convolution or the DFT/FFT to solve this. Solve the difference equation directly. How does the audio change?