Full Name: ______________________________

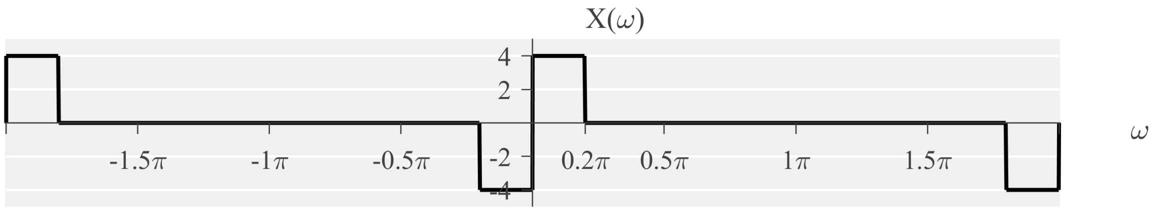
EEL 4750 / EEE 5502 (Fall 2021) – HW #09 **Due: 4:00 PM ET, Nov. 08, 2021**

Concept Questions 09

Question #1: Complete the Canvas questions here: `https://ufl.instructure.com/courses/437179/assignments/4812585`

Theory Questions 09

Question #1: Consider the DTFT of $x[n]$ shown below



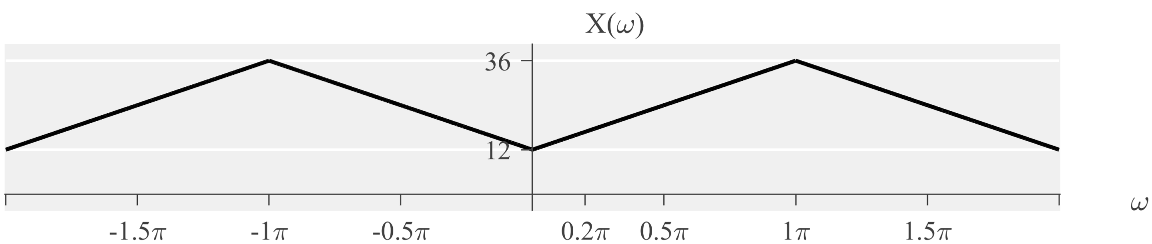
(a) Sketch the DTFT of $x[n]$ after downsampling by 4 (without an anti-aliasing filter)

(b) Sketch the DTFT of $x[n]$ after downsampling by 8 (without an anti-aliasing filter)

(c) Sketch the DTFT of $x[n]$ after downsampling by 4 (with an anti-aliasing filter)

(d) Sketch the DTFT of $x[n]$ after downsampling by 8 (with an anti-aliasing filter)

Question #2: Consider the DTFT of $x[n]$ shown below



(a) Sketch the DTFT of $x[n]$ after upsampling by 2 (without an interpolation filter)

(b) Sketch the DTFT of $x[n]$ after upsampling by 4 (without an interpolation filter)

(c) Sketch the DTFT of $x[n]$ after upsampling by 2 (with an interpolation filter)

(d) Sketch the DTFT of $x[n]$ after upsampling by 4 (with an interpolation filter)

Implementation Questions 09

Question #1: Included is a low-pass filter function `lpf_func(x,wc,P)` that uses the difference approximation method to design and apply a P-th order Butterworth low pass filter to signal `x` with a cut-off frequency of `wc` (normalized to be between 0 and π). Use $P = 10$.

(a) Create a function `y = downsample_func(x, N)` that downsamples signal `x` by `N`. The output signal `y` should have a length of `ceil(Nx/N)`, where `Nx` is the length of the original signal and `ceil` is the ceiling function.

(b) Create a function `y = upsample_func(x, M)` that upsamples signal `x` by `M`. The output signal `y` should have a length of `Nx*M`, where `Nx` is the length of the original signal.

(c) Create a function `y = downsample_antialias_func(x, N)` that combines `lpf_func` with your downsample function to downsample signal `x` by `N` after passing thru an anti-aliasing filter.

(d) Create a function `y = upsample_interp_func(x, N)` that combines `lpf_func` with your upsample function to upsample signal `x` by `M` before passing thru an interpolation filter.

Question #2: Use the provided chirp function `chirp` to create a signal $x[n]$

$$x[n] = \cos\left(2\pi(f_1/(2N_x))n^2\right)$$

where `nx=0:Nx-1`, `Nx = 256`, and `f1=1/8` as the maximum frequency. Compute and plot the resulting signal and frequency magnitude representation for the following scenarios.

(a) Compute $x[n]$ after downsampling by 2 (with no anti-aliasing).

(b) Compute $x[n]$ after downsampling by 5 (with no anti-aliasing).

(c) Compute $x[n]$ after downsampling by 5 (with anti-aliasing).

Question #3: Use the provided chirp function `chirp` to create a new signal $x[n]$ where `nx=0:Nx-1`, `Nx = 64`, and `f1=1/8` as the maximum frequency. Compute and plot the resulting signal and frequency magnitude representation for the following scenarios.

(a) Compute $x[n]$ after upsampling by 2 (with no interpolation).

(b) Compute $x[n]$ after upsampling by 5 (with no interpolation).

(c) Compute $x[n]$ after upsampling by 5 (with interpolation).

Question #4: Load `urquan.wav`, containing audio from the Ur-Quan Masters game (`http://sc2.sourceforge.net/`). We will speed up the audio with downsampling and the STFT.

(a) First, use your downsample function to downsample (without anti-aliasing) the audio by 3.

(b) Use the provided `STFT = stft_func(x, W)` (similar to what you previously made) to compute the STFT of the original audio with $W = 600$. Downsample (across time) the STFT for each frequency. Do **not** use an anti-aliasing filter.[1] Then use the provided `x = istft_func(STFT, W)` function to compute the inverse STFT.

(c) How are your results different for the previous two questions?

[1] The STFT is complex-valued and traditional filtering does not work / make sense in this context.