% Tune-Up Tuesday #7 for October 23, 2018

% Play *x*[*n*], which is an 800 Hz tone for 3s at a sampling rate of 8000 Hz:

fs = 8000;

Ts = 1/fs;

t = 0 : Ts : 3;

f0 = 800;

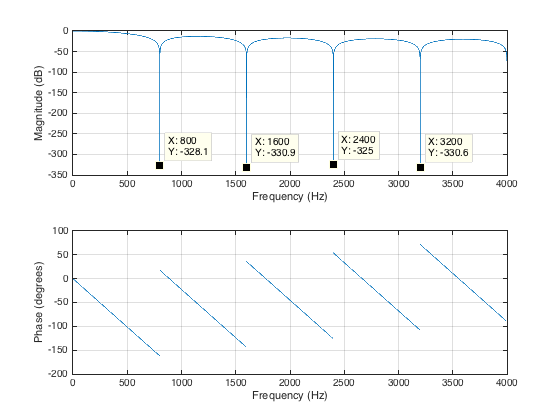
x = cos(2\*pi\*f0\*t);

sound(x, fs); % please use the sound command for tune-up

pause(3);

% Discrete-time frequency for the cosine is

% w0 = 2\*pi\*f0/fs = 2\*pi\*800/8000 = 0.2\*pi

% (a) Define an impulse response *h*[*n*] of an averaging filter of 10 coefficients.

h10 = (1/10)\*ones(1, 10);

% (b) Plot its magnitude/phase response

freqz(h10); % not shown

% (c) At what frequencies (in Hz) does the magnitude response equal zero?

% *Hint: You can use the data cursor tool in the freqz plot window*.

% For 1 Hz accuracy in freqz and

% horizontal axis in Hz, use

freqz(h10, 1, fs, fs); % plot on right

% 800, 1600, 2400, 3200, 4000 Hz

% -800, -1600, -2400, -3200 Hz

% Are the frequencies harmonically related?

% Yes, over frequencies captured via sampling at sampling rate fs,

% i.e. –fs/2 to fs/2, integer multiples are 800 Hz have been

% zeroed out except 0 Hz.

% Can you give a formula for the frequencies in terms for an *N*-point averaging filter?

% fs/N, 2\*fs/N, 3\*fs/N, etc.

% -fs/N, -2\*fs/N, -3\*fs/N, etc.

% (d) Filter *x*[*n*] using the averaging filter *h*[*n*] and play the result:

y10 = filter(h10, 1, x);

sound(y10, fs);

pause(3);

% Playback is silent because the filter filters out (rejects)

% the frequency of the input sinusoid (800 Hz). **See Epilog.**

% (e) Filter *x*[*n*] using a five-point averaging filter and play the result

h5 = (1/5)\*ones(1, 5);

y5 = filter(h5, 1, x);

sound(y5, fs);

pause(3);

freqz(h5, 1, fs, fs);

% From the freqz plot, the filter reduces amplitude of the cosine at

% 800 Hz (0.2pi) by about -3.7 dB. AdB = 20 log10 A = -3.7 dB, which

% means that A = 10^(-3.7/20) = 0.653. **See Epilog**.

% (f) Filter *x*[*n*] using a 15-point averaging filter and play the result

h15 = (1/15)\*ones(1, 15);

y15 = filter(h15, 1, x);

sound(y15, fs);

pause(3);

freqz(h15);

% From the freqz plot, the filter reduces amplitude of the cosine at

% 800 Hz (0.2pi) by about -13.37 dB, which is a gain of 0.2145. **See Epilog**.

% (g) Filter *x*[*n*] using a 20-point averaging filter and play the result

h20 = (1/20)\*ones(1, 20);

y20 = filter(h20, 1, x);

sound(y20, fs);

pause(3);

freqz(h20);

% Playback is silent because the averaging filter filters out (rejects)

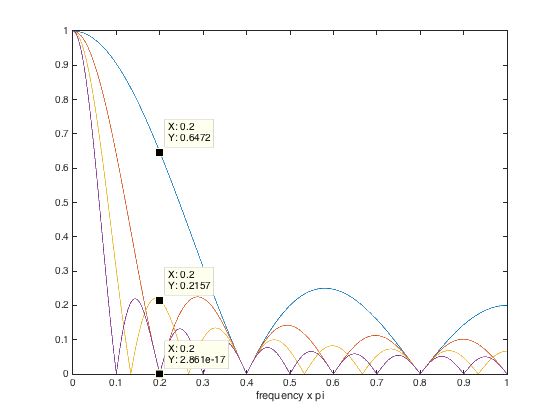
% the frequency of the input sinusoid (800 Hz). **See Epilog.**

% ***Epilog***. Here we superimpose the magnitude responses for the four averaging

% filters: 5-point (blue), 10-point (red), 15-point (yellow), and 20-point (purple).

% The data cursor indicates the magnitude response at 0.2\*pi (i.e. 800 Hz).

% Lowpass filter: passes low frequencies and attenuates high frequencies.



fs = 8000;

for N = [5 10 15 20]

coeffs = (1/N)\*ones(1, N);

[H, W] = freqz(coeffs, 1, fs);

plot( W/pi, abs(H) );

hold on;

end

xlabel('frequency x pi');

% *N*-point averaging filter: (a) extent in positive frequencies that have magnitude

% response in linear units close to 1 is proportional to 2\*pi/*N,* and (b) zeros out

% discrete-time frequencies that are multiples of 2\*pi/N but not multiples of 2\*pi.