Nima Mohammadi

Digital Communications

Homework #3

1. Consider a compact disk player that uses pulse-code modulation to record audio signals whose bandwidth is B = 15kHz. If the quantizer uses 512 levels and the waveform encoder uses binary waveforms (e.g., square pulses) determine the minimum possible bit rate.

   The signal has bandwidth B, so the sampling rate must be at least $f_s = 2B = 30\text{kHz}$

   For representing 512 levels, the quantizer needs at least 9 bits ($512 = 2^9$)

   The PCM bit rate is therefore $r = f_s n = 30 \times 9 = 270\text{kHz}$

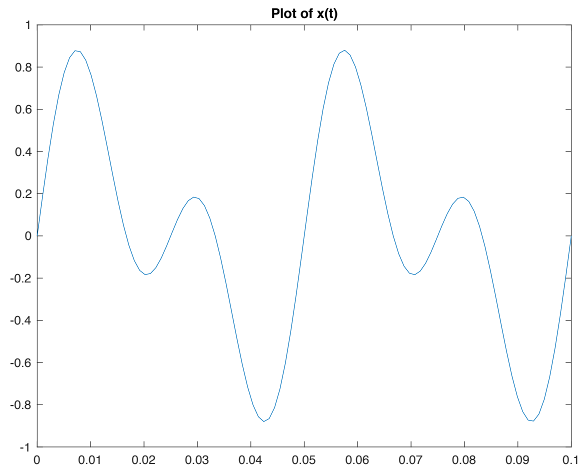
2. Consider the signal $x(t) = \cos(20\pi t)\sin(60\pi t)$ sampled at a frequency of 1kHz.

   2.a. Plot the signal in Matlab from t = 0 to t=0.1s.

MATLAB/Octave code

```
%% Q2.a
fs = 1000;                          % sampling frequency
t = linspace(0, .1, .1*fs);         % time vector
x = cos(20*pi*t).*sin(60*pi*t);
plot(t, x);
title('Plot of x(t)’);
```

Plot

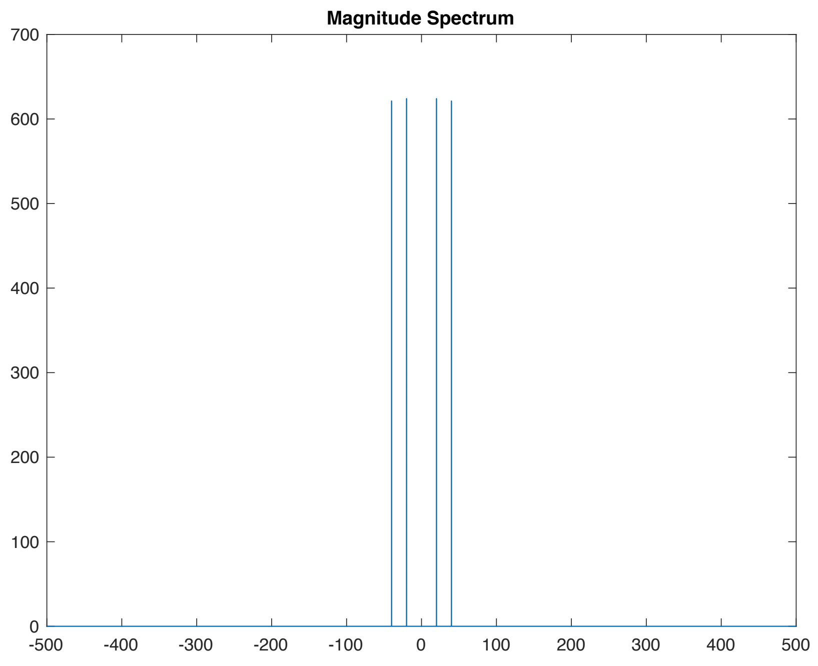


2.b. Plot the magnitude spectrum of the signal. Use the Matlab command `fftshift` to center the spectrum at zero. Use at least 10 seconds of data to create the spectrum.

MATLAB/Octave code

```
%% Q2.b
t2 = linspace(0, 10, 10*fs);      % time vector
x2 = cos(20*pi*t2).*sin(60*pi*t2);
n = length(x2);
X = fft(x2);
% f = (0:n-1)*(fs/n);            %frequency range

Z = fftshift(X);
fshift = (-n/2:n/2-1)*(fs/n);    % zero-centered frequency range
powershift = abs(Z).^2/n;        % zero-centered power
plot(fshift, powershift);
title('Magnitude Spectrum')
```

Plot



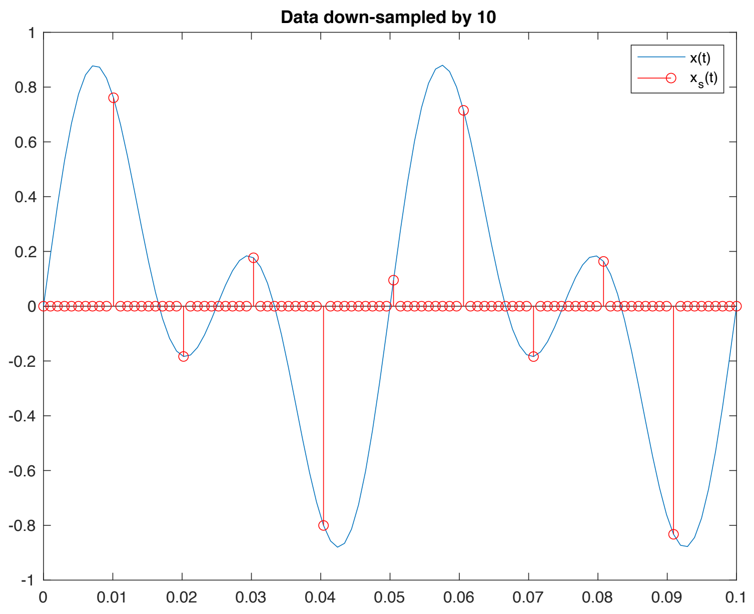
2.c. Now down-sample the signal by a factor of 10.

2.d. Plot the signal on the same graph as x(t) using `stem(t(1:10:end), y(1:10:end),'r');`

MATLAB/Octave code

```
%% Q2.c and Q2.d
y = zeros(1, length(x));
y(1:10:end) = x(1:10:end);
plot(t, x);
hold on
stem(t, y, 'r');
legend('x(t)','x_s(t)');
title('Data down-sampled by 10')
hold off;
```

Plot

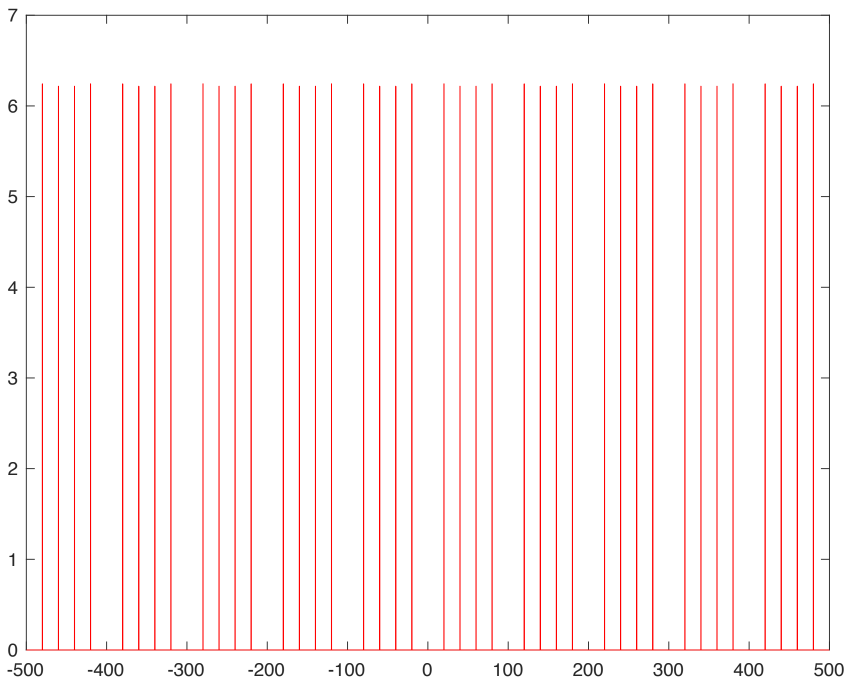


2.e. Plot the magnitude spectrum and compare to the original spectrum.

MATLAB/Octave code

```
%% Q2.e
y2 = zeros(1, length(x2));
y2(1:10:end) = x2(1:10:end);
Y = fft(y2);
Z2 = fftshift(Y);
powershift2 = abs(Z2).^2/n;
plot(fshift, powershift2, 'r');
```

Plot



Explanation

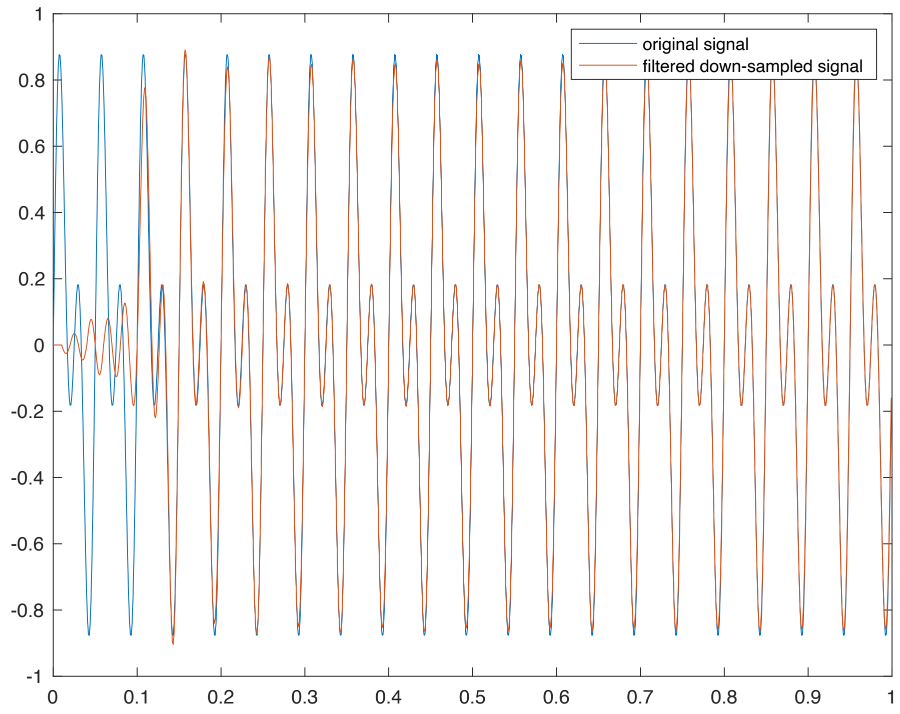
We can observe that aliasing, an artifact of sampling, has occurred. A frequency spectrum of the samples produces equally strong responses at all frequencies which are integer multiplications of the fundamental frequency of the signal. Without collateral information, the frequency of the original function is ambiguous. So the functions and their frequencies are said to be aliases of each other. Aliasing matters when one attempts to reconstruct the original waveform from its samples. As we will see below, we can use a sinc filter, as an ideal low-pass filter, to remove all frequency components above a given cutoff frequency.

2.f. Let h(t) = sinc(100(t-10)) be the impulse response of a causal filter. (NOTE: We are using Matlab's definition for the sinc function.) The delay allows us to have a causal filter. Pass the sampled signal through this filter and examine the output. Plot the time domain signal along with the samples. Does the new signal match the samples? Does it match the original signal?

MATLAB/Octave code

```
%% Q2.f
t2 = linspace(0, 10, 10*fs);
x2 = cos(20*pi*t2).*sin(60*pi*t2);
h = sinc(100*t2-10);

y2 = zeros(1, length(x2));
y2(1:10:end) = x2(1:10:end);

y2_filtered = conv(h, y2);
% y2_filtered = filter(h, 1, y2);  % Alternatively

plot(t2(1:1000), x2(1:1000));
hold on
% stem(t2(1:1000), y2(1:1000), 'r');
% hold on
plot(t2(1:1000), y2_filtered(1:1000));
legend('original signal', 'filtered down-sampled signal')
```

Plot



Explanation

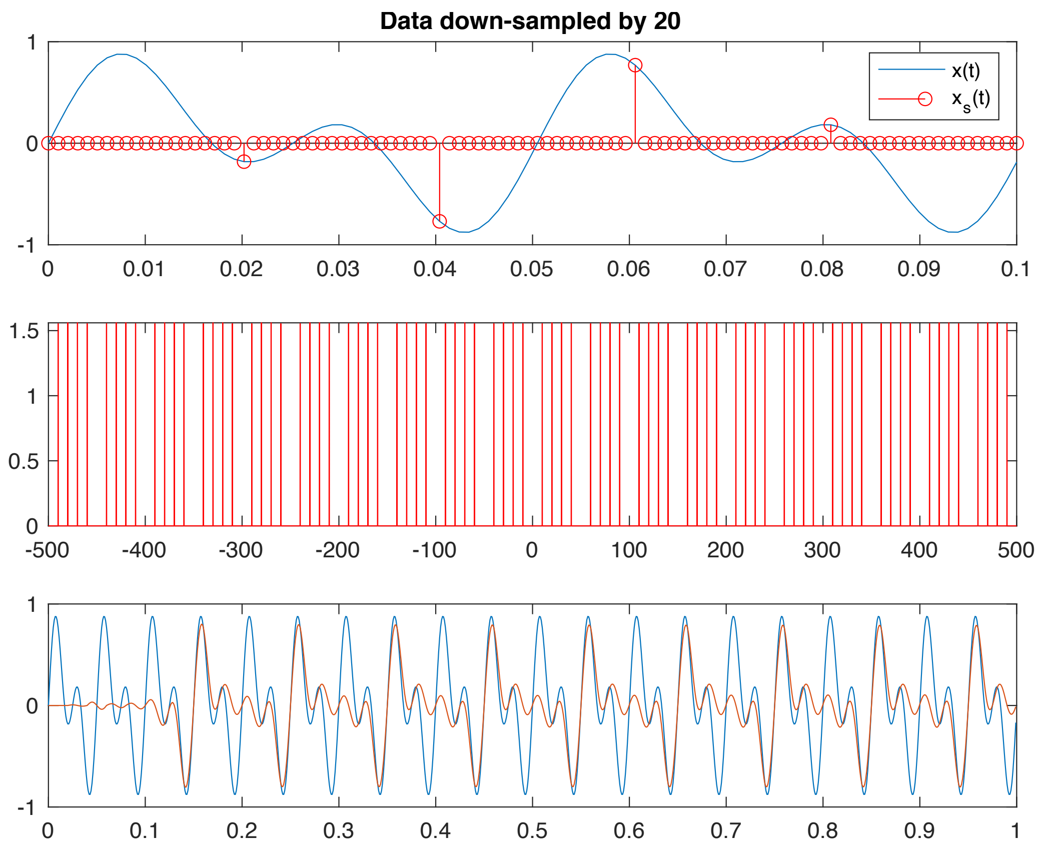
As we mentioned above, we can use a sinc function as an ideal low-pass filter to remove unwanted components. However, an ideal filter would require an infinitely long period of time to remove those. By delaying the sinc function we get a causal filter which is a requirement for the filter to be realizable. As it can be seen from the plot, proper reconstruction is achieved (except for the initial segment which is because it takes time for the filter to take effect).

2.g. Repeat steps c-f using a down-sampling factor of 20. Have things changed? Why or why not?

MATLAB/Octave code

```
%% Q2.g
y3 = zeros(1, length(x2));
y3(1:20:end) = x2(1:20:end);

subplot(3, 1, 1)
plot(t(1:100), x2(1:100));
hold on
stem(t(1:100), y3(1:100), 'r');
legend('x(t)','x_s(t)');
title('Data down-sampled by 20')

subplot(3, 1, 2)
Y = fft(y3);
Z2 = fftshift(Y);
powershift2 = abs(Z2).^2/n;
plot(fshift, powershift2, 'r');

subplot(3, 1, 3)
y3_filtered = conv(h, y3);

plot(t2(1:1000), x2(1:1000));
hold on
plot(t2(1:1000), y3_filtered(1:1000));
```

Plot



Explanation

This time, we can see that filtering the down-sampled signal has not led to a full reconstruction of the original signal. While one of the components seems to have been recovered, the same does not go for the other component. This has occurred as the Nyquist criterion is not satisfied. Basically, the sampling rate is not high enough to capture all the information from the original signal.

3. Consider an analog power signal with a bandwidth of 200kHz. It is to be converted to a digital signal using pulse code modulation with (binary) polar non-return to zero encoding and square pulses. It can be assumed that the original analog signal is uniformly distributed. The required quantization SNR (or SDR) is 35dB.

a) Determine the minimum data rate of the digital signal that allows the system to exceed the required quantization SNR.

The bandwidth of the analog signal is 200kHz, so the minimum sampling frequency is $f_s = 2 \times 200\text{kHz} = 400\text{kHz}$

The SNR of PCM is $\text{SNR} = (1.8 + 6N)\text{dB}$

$1.8 + 6N = 35\text{dB}$

$N = 6$bits is required.

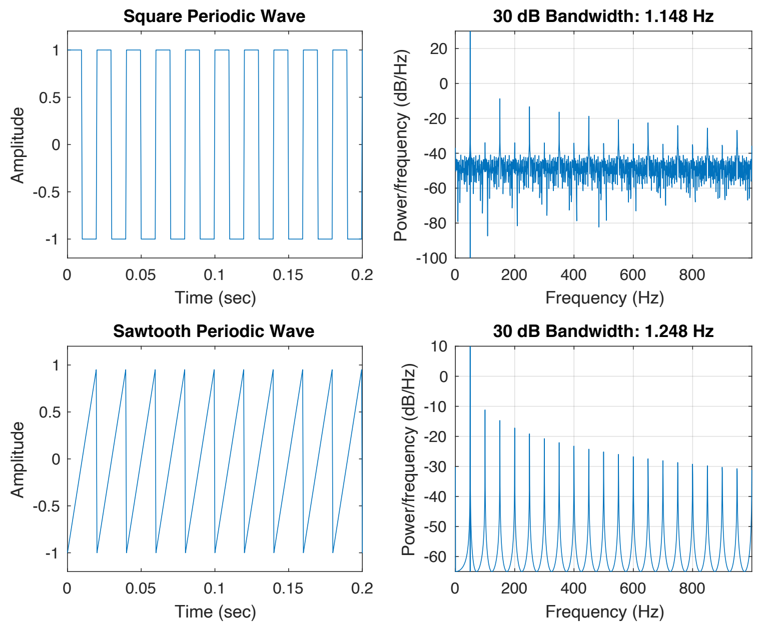
Then the minimum data rate would be

$N \times f_s = 6 \times 400000 = 2.4\text{Mbps}$

b) What is the minimum bandwidth required for the resulting digital signal if the bandwidth is measured as the 30dB-bandwidth?

c) What is the minimum 30dB bandwidth if the pulse is a triangular pulse (use a Matlab plot to help determine this).

I am having problem with this section of the question. But I believe it can be done similar to code below in MATLAB



```
fs = 2000;
t = 0:1/fs:1.5;
x1 = square(2*pi*50*t);
x2 = sawtooth(2*pi*50*t);

figure;

subplot(2, 2, 1)
plot(t, x1)
axis([0 0.2 -1.2 1.2])
xlabel('Time (sec)')
ylabel('Amplitude')
title('Square Periodic Wave')

subplot(2, 2, 2)
powerbw(x1, fs, [], 30)

subplot(2, 2, 3)
plot(t,x2)
axis([0 0.2 -1.2 1.2])
xlabel('Time (sec)')
ylabel('Amplitude')
title('Sawtooth Periodic Wave')

subplot(2, 2, 4)
powerbw(x2, fs, [], 30)
```