1. **Boxcar Averager** [2 points] A sliding window averager has a window length of three samples. On the same axes, plot the input and output of this averager given the input samples listed below. Assume that the input at times *before* and *after* these samples is equal to zero.

   0 0.2 0 -0.1 0.3 0.8 1.2 0.9 1.1 1.2 0.8 1.1 1.2 0.8 1.1 0 -0.2 -0.1 0 -0.2

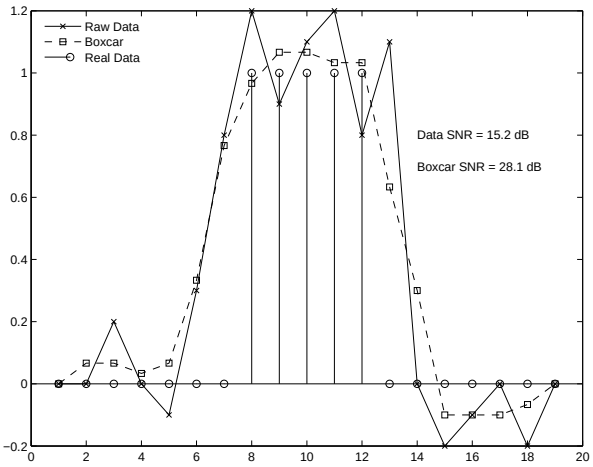
   If the actual signal trying to be represented by these samples is:

   0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

   then comment on the pros / cons of the sliding averager relative to recovering the actual signal.

   Qualitatively you observed a reduction in noise, but the quantitative estimate of SNR improvement did not achieve the theoretical $\sqrt{3}$ because you didn't take into account what the "true" signal should be (I didn't tell you what is was); performing mean and standard deviation operations over the entire signal includes the variability of the actual signal (which in this case was a `rect`).

   Remember that SNR = $\frac{P_{signal}}{P_{noise}}$. If we look at the SNR improvement in the data from Problem Set #6 in the region of the `rect` that has actual signal (non-zero values), then we get the following:



   Looking at the SNRs calculated above, how does the SNR improvement compare with the "expected" $\sqrt{3}$? Why might this be?

   The SNR improvement is $\frac{10^{28.1/20}}{10^{15.2/20}} \simeq 4.4 > \sqrt{3}$. This is partially due to the fact that the $\sqrt{N}$ rule-of-thumb was derived under the assumption of a normal distribution of data points; having such a small number of points violates that assumption.

2. **Exponentially-Weighted Averager** An exponentially-weighted averager that produces an output signal that is a weighted sum of a series of input signals described by:

   $S_o = \frac{1}{m}\left[S_n + \left(\frac{m-1}{m}\right) S_{n-1} + \left(\frac{m-1}{m}\right)^2 S_{n-2} + \ldots\right]$,

   where m is the exponential weighting factor.

   - Demonstrate how m affects the weighting of more recent or later samples in the exponential averager. [1 point]
   - We derived the SNR improvement associated with an equally-weighted and boxcar averager to be $\sqrt{N}$, where N is the number of samples being averaged. Derive the SNR improvement for the exponentially-weighted averager (hint - it should be a function of m). [1 point]

- We also discussed how the boxcar averager acted as a low-pass filter on the data. Sketch the transfer function of the boxcar averager relative to a sinusoidal signal oscillating at ω_o, and demonstrate how if the averaging window is too long, then you are just left with the DC offset of the sinusoidal signal in the output. [1 point]
- Compare and contrast the frequency-domain transfer function of a boxcar averager with that of an exponentially-weighted averager. How does the transfer function of the exponentially-weighted averager change as a function of m? [1 point]

$$S_o = \frac{1}{m}\left[S_n + \left(\frac{m-1}{m}\right)S_{n-1} + \left(\frac{m-1}{m}\right)^2 S_{n-2} + ...\right]$$

Lets see how the weights change as a function of m for 30 samples (Figure 1).

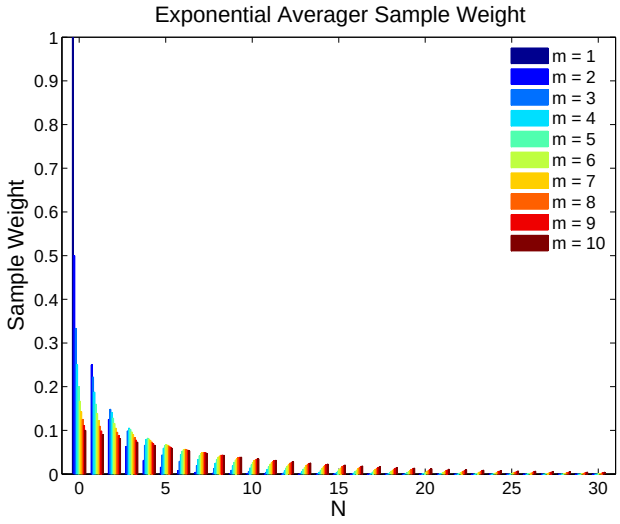


Figure 1: Weights for samples in an exponential averager as a function of m.

Notice that as m increases, the impact of older samples increases. In the limiting case (m = 1), we simply have a delta function for the most recent sample. Question to consider: How does the overall energy of the output signal change as a function of m? Is it constant?

The SNR improvement of the exponential averager is a function of m. There is no SNR improvement for $m = 1$.

For the exponential averager, the new average estimate (A_m) can be related to the previous estimate (A_{m-1}) with a new input (I_m), as:

$$A_m = A_{m-1} + \frac{I_m - A_{m-1}}{m} = \frac{1}{M}\sum_{i=1}^{m}\left(\frac{M-1}{M}\right)^{m-i} I_i, \tag{1}$$

where M is the number of samples with history in the averager.

The derivation of the SNR associated with the exponential averager can be expressed after expansion of Equation 1 as:

$$\lim_{m\to\infty} \mathrm{SNR_o} \frac{\left[1-(1-\frac{1}{M})^m\right]}{\left[\frac{1-(1-\frac{1}{M})^{2m}}{2M-1}\right]^{\frac{1}{2}}} = \mathrm{SNR_o}(2M-1)^{\frac{1}{2}}, \tag{2}$$

where $\mathrm{SNR_o}$ is the original signal SNR.

The filtering characteristics of the boxcar averager, as a function of the length of the boxcar, are shown in Figure 2. Notice how for a single sample boxcar length (i.e., a delta function), all frequencies are passed (i.e., a uniform transfer function), but as the length of the boxcar increases, the passband of the transfer function reduces down to something that, in the extreme, would be a delta function at $f = 0$, only allowing DC characteristics to be preserved.

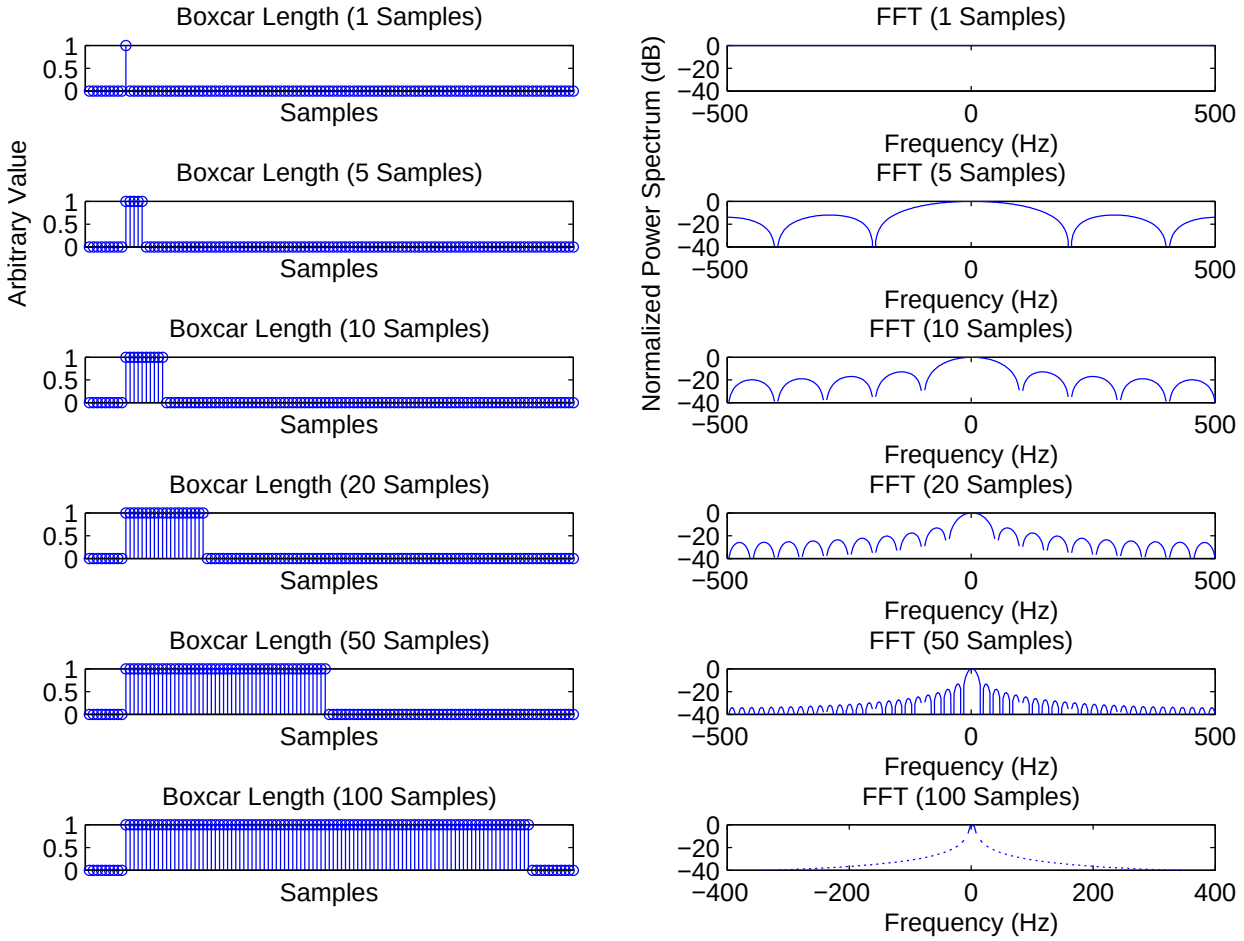


Figure 2: Magnitude transfer functions for different length boxcar averagers. The sampling frequency was arbitrarily set to 1 kHz.

To compare/contrast, the power spectra of the exponential averager for 3 different m weights are shown in Figure 3. Notice how for increasing m, the passband of the exponential averager transfer function becomes significantly limited, effectively acting as a low-pass filter, similar to what happens with the boxcar averager, but to less of an extreme.

Code for various plots shown in these solutions is as follows:

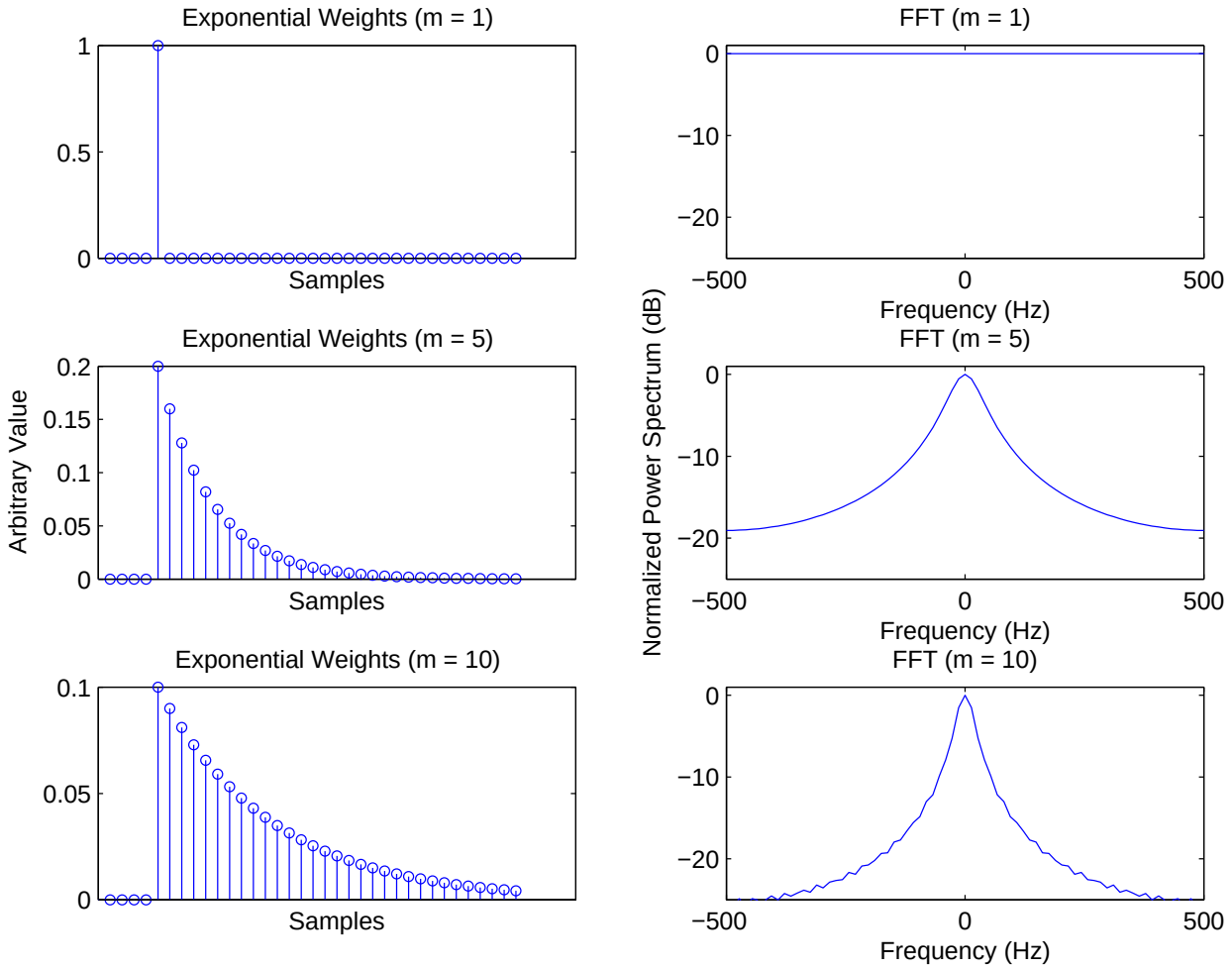


Figure 3: Magnitude transfer functions for differently weighted exponential averagers . The sampling frequency was arbitrarily set to 1 kHz.

```
% sliding_average.m
%
% BME154L (Spring 2012)
% Problem Set #6 (Sliding Window Averager)
%
% Mark Palmeri

data = [ 0 0.2 0 -0.1 0.3 0.8 1.2 0.9 1.1 1.2 0.8 1.1 0 -0.2 -0.1 0 -0.2 ];

% pad the data with a 0 on the front and back to do a 3-sample boxcar
data = [0 data 0];

figure;
plot(data,'-x');
hold on;
```

```
% 3-sample boxcar
data_boxcar = zeros(1,length(data));
data_boxcar(2:end-1) = (data(1:end-2) + data(2:end-1) + data(3:end))/3;
plot(data_boxcar,'--s');

% real signal
real = [0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0];
stem(real,'-')

% compute SNR
nonzero = find(real ~= 0)
snr_data = 20*log10(mean(data(nonzero))/std(data(nonzero)))
snr_data_boxcar = 20*log10(mean(data_boxcar(nonzero))/std(data_boxcar(nonzero)))

legend('Raw Data','Boxcar','Real Data','Location','Northwest');
legend boxoff
text(14.0,0.8,sprintf('Data SNR = %.1f dB',snr_data));
text(14.0,0.7,sprintf('Boxcar SNR = %.1f dB',snr_data_boxcar));

function []=boxcar_fft();
% function []=boxcar_fft();
%
% Compute the FFT of the boxcar averager with a different number of samples
%
% INPUTS: None
%
% OUTPUTS: None
%
% EXAMPLE: boxcar_fft
%
% Mark Palmeri (mlp6)
% BME154L S12 PS6
% 2012-03-19

% define the number of samples for the boxcar averager
N=[1 5 10 20 50 100];

% define the sampling frequency to be able to compute the frequency axis for the FFT
fs = 1000; % Hz

for n=1:length(N),
    box = zeros(300,1);
    box(100:100+N(n)-1) = 1;
    boxcarFFT = fft(box);
    f = linspace(-fs/2,fs/2,length(boxcarFFT));
    subplot(length(N),2,n*2-1);
    stem(box,'MarkerSize',4);
    xlim([90 210]);
    title(sprintf('Boxcar Length (%i Samples)',N(n)));
    set(gca,'XTick',[])
```

```
    xlabel('Samples');
    if(n == 2),
        ylabel('Arbitrary Value');
    end;
    subplot(length(N),2,n*2);
    plot(f,fftshift(20*log10(abs(boxcarFFT)./max(abs(boxcarFFT)))));
    ylim([-40 0]);
    xlabel('Frequency (Hz)');
    if(n == 2),
        ylabel('Normalized Power Spectrum (dB)');
    end;
    title(sprintf('FFT (%i Samples)',N(n)));
end;

print('-depsc2','boxcar_fft.eps');

function []=exp_weight_ave()
% function []=exp_weight_ave()
%
% Demonstrate the properties of the exponentially-weighted averager (compared
% w/ a boxcar averager).  [BME154L S12 PS6.3]
%
% INPUTS: None
%
% OUPUTS: None
%
% EXAMPLE: exp_weight_ave
%
% Mark Palmeri (mlp6)
% 2012-03-16

% define coefficient weight
% m = 1 is a limiting case since only the S_n term will be non-zero,
% which makes this impulse response look like a delta function

N = [0:30]; % number of samples in the averager to show

for m=1:10,
    for n=N,
        S(n+1,m) = (1/m)*((m-1)/(m))^n;
    end;
end;

figure;
bar(N,S);
xlim([-1 31]);
set(gca,'XTick',[0:5:30]);
xlabel('N','FontSize',18);
ylabel('Sample Weight','FontSize',18);
title('Exponential Averager Sample Weight','FontSize',18);
```

```
legend('m = 1','m = 2','m = 3','m = 4','m = 5','m = 6','m = 7','m = 8','m = 9','m = 10
legend boxoff

set(gca,'FontSize',14)

print('-depsc2','exp_weight_ave.eps');

% lets do some Fourier analysis on these...

fs = 1000; % Hz

expFFT = zeros(1000,100);
M=[1 5 10];
for m=1:length(M),
    expbox(45:45+max(N),m) = S(:,M(m));
    expFFT = fft(expbox(:,m));
    f = linspace(-fs/2,fs/2,length(expFFT));
    subplot(length(M),2,m*2-1);
    stem(expbox(:,m),'MarkerSize',4);
    xlim([40 40+max(N)+10]);
    set(gca,'XTick',[]);
    title(sprintf('Exponential Weights (m = %i)',M(m)));
    xlabel('Samples');
    if(m == 2),
        ylabel('Arbitrary Value');
    end;
    subplot(length(M),2,m*2);
    plot(f,fftshift(20*log10(abs(expFFT)./max(abs(expFFT)))));
    ylim([-25 1]);
    xlabel('Frequency (Hz)');
    if(m == 2),
        ylabel('Normalized Power Spectrum (dB)');
    end;
    title(sprintf('FFT (m = %i)',M(m)));
end;

print('-depsc2','exp_weight_ave_fft.eps');
```