% **Tune-Up #5** **October 15, 2024**

% The tuneup is to solve homework problem 5.1.

% **Intro**. A step function u[n] is a function

% that turns at the origin and stays on. This

% can model turning on a switch and leaving it

% on indefinitely. Mathematically, u[n] is

% 1 when n >= 0

% 0 otherwise.

% In Matlab, one can implement u[n] as ( n >= 0 ).

% The logical operator >= returns 1 if true and

% 0 if false.



% **Part (a).** Make a plot of u[n] for -5 <= n <= 10.

% Describe what you see.

n = -5 : 10;

unitstep = ( n >= 0 );

figure;

stem(n, unitstep);

xlabel('n');

ylabel('u[n]');

ylim([-0.5 1.5]);

% In the plot, the signal is zero/off when

% n < 0 and one/on when n >= 0. This is a

% step function -- the signal takes a step up

% at n = 0 from amplitude 0 to amplitude 1.



% **Part (b).** We can use the unit-step sequence

% to represent other sequences that are zero

% for n < 0. Plot x[n] = (0.5)^n u[n]

% for -5 <= n <= 10. Describe what you see.

n = -5:10;

unitstep = ( n >= 0 );

x = (0.5 .^ n ) .\* unitstep;

figure;

stem(n, x);

xlabel('n');

ylabel('0.5^n u[n]');

ylim([-0.5 1.5]);

% In the plot, signal is zero when n < 0 and a

% a decaying exponential sequence when n >= 0.



% **Part (c).** Apply a four-point averaging

% filter to x[n] and plot the result

% Solution #1: Using the filter command

averagingFilterCoeffs = [ 1/4, 1/4, 1/4, 1/4 ];

y = filter(averagingFilterCoeffs, 1, x);

figure;

stem(n, y);

xlabel('n');

ylabel('Output signal using the filter command');

ylim([-0.5 1.5]);

% In the plot, the output signal is zero when

% n < 0. The output signal for 0 <= n <= 2

% corresponds to a partial response by the filter

% to the change in the input signal at the origin

% Once we reach n = 3, the sliding window of input

% samples would be filled.

% Solution #2: Using the convolution command, conv.

% When the input signal and the impulse response
% are finite length, convolution will produce a signal

% whose length is the length of the input signal plus

% the length of the impulse response minus one.

averagingFilterCoeffs = [ 1/4, 1/4, 1/4, 1/4 ];

y = conv(averagingFilterCoeffs, x);

numberExtraSamples = length(averagingFilterCoeffs) - 1;

n = -5 : 10 + numberExtraSamples;

figure;

stem(n, y);

xlabel('n');

ylabel('Convolution of x[n] and h[n]');

ylim([-0.5 1.5]);

% In the plot, the output signal is zero when n < 0. The output

% signal for 0 <= n <= 2 corresponds to a partial response by the

% filter to the change in the input signal at the origin. Once

% we reach n = 3, the sliding window of input samples would be

% filled. We also see the trailing response from convolution.



% Solution #3: Using filter command to perform convolution

averagingFilterCoeffs = [ 1/4, 1/4, 1/4, 1/4 ];

numberExtraSamples = length(averagingFilterCoeffs) - 1;

xZeroPadded = [x zeros(1, numberExtraSamples)];

y = filter(averagingFilterCoeffs, 1, xZeroPadded);

n = -5 : 10 + numberExtraSamples;

figure;

stem(n, y);

xlabel('n');

ylabel('Using the filter command to mimic the conv comand');

ylim([-0.5 1.5]);

% In the plot, the output signal is zero when n < 0. The output

% signal for 0 <= n <= 2 corresponds to a partial response by the

% filter to the change in the input signal at the origin. Once

% we reach n = 3, the sliding window of input samples would be

% filled. We also see the trailing response from convolution.