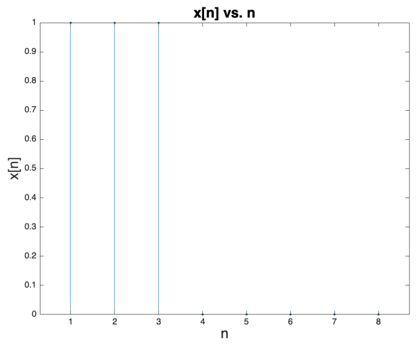
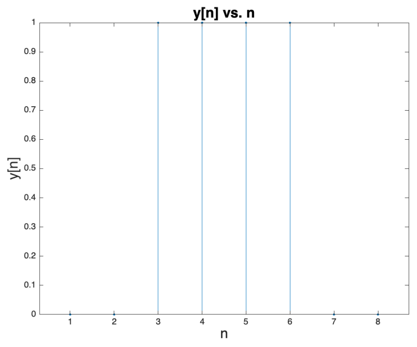
Lab 4

Problem 1: Implementing periodic convolution

In this problem, we will implement periodic convolution and compare it with the frequency domain implementation.

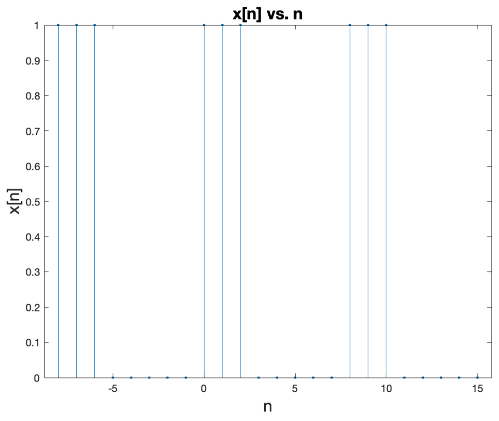
a) We start by creating and plotting one period of the two periodic signals $x[n]$ and $y[n]$:





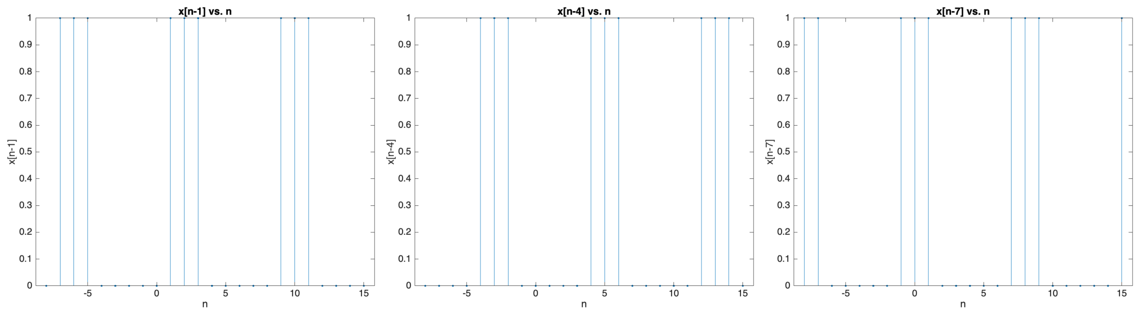
```
n = 1:8
x_arr = zeros(1,8);
y_arr = zeros(1,8);
x_arr(1:3) = 1;
y_arr(3:6) = 1;

stem(n, x_arr, '.')
title('x[n] vs. n', 'FontSize', 16)
xlabel('n', 'FontSize', 16);
ylabel('x[n]', 'FontSize', 16);

stem(n, y_arr, '.')
title('y[n] vs. n', 'FontSize', 16)
xlabel('n', 'FontSize', 16);
ylabel('y[n]', 'FontSize', 16);
```

b) Plotting $x[n]$ for $n = -8$ to $n = 15$, we observe the following:



```
n = -8:15;
x_arr3 = zeros(1, length(n));
for i = 0:2
    x_arr3(8*i+1:8*(i+1))=x_arr
end

stem(n, x_arr3, '.')
title('x[n] vs. n', 'FontSize', 16)
xlabel('n', 'FontSize', 16);
ylabel('x[n]', 'FontSize', 16);
```

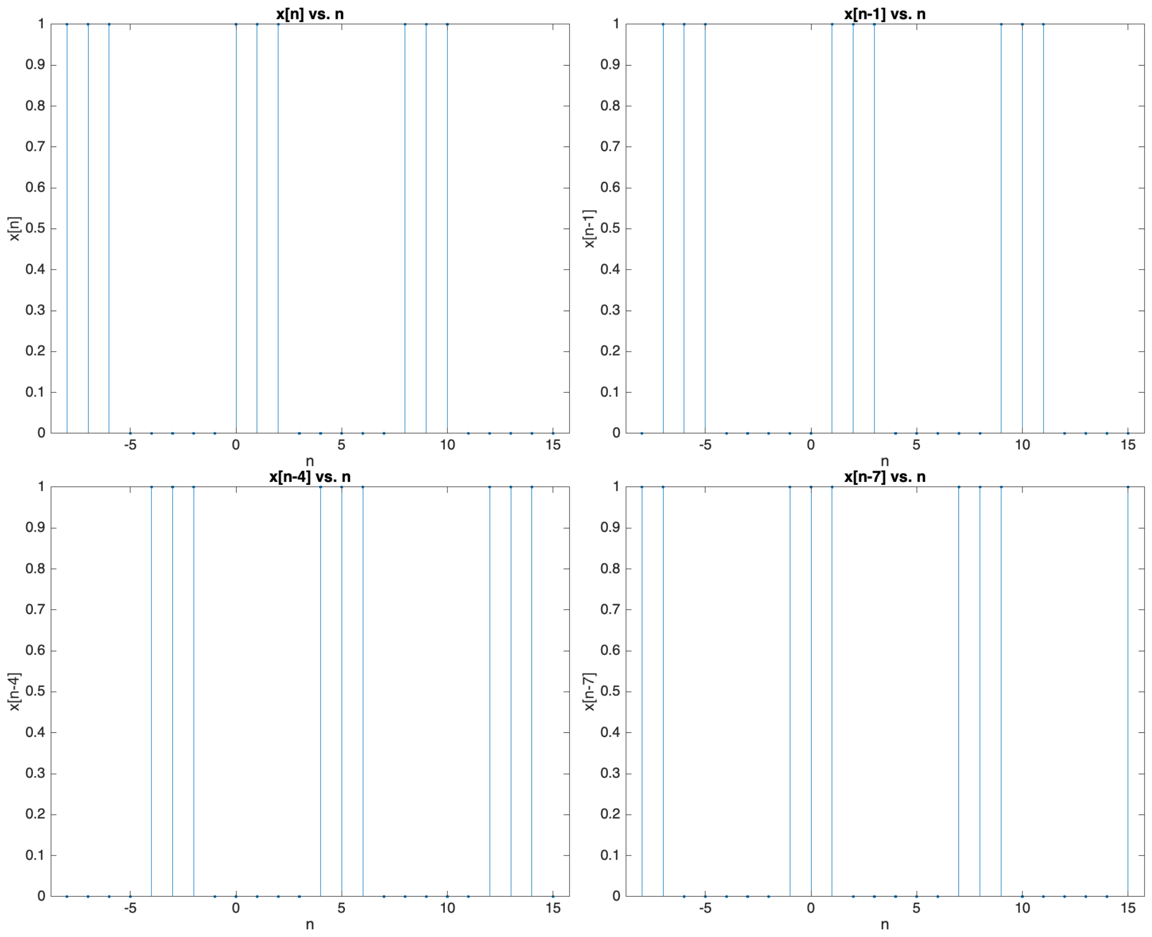
c) Plotting $x[n-1]$, $x[n-4]$, and $x[n-7]$ for the same values of n, we obtain the following plots:



```
x_ext=x_arr3;
x_ext(25:48)=x_arr3;

shiftval=1;
stem(n, x_ext(9-shiftval:32-shiftval), '.');
title('x[n-1] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-1]', 'FontSize', 16);

shiftval=4;
stem(n, x_ext(9-shiftval:32-shiftval), '.');
```

```
title('x[n-4] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-4]', 'FontSize', 16);

shiftval=7;
stem(n, x_ext(9-shiftval:32-shiftval), '.');
title('x[n-7] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-7]', 'FontSize', 16);
```

d) Plotting `circshift(x_arr, shift_val)` for `shift_val` sequentially taking $0, 1, 4, 7$, we can observe the following plots:

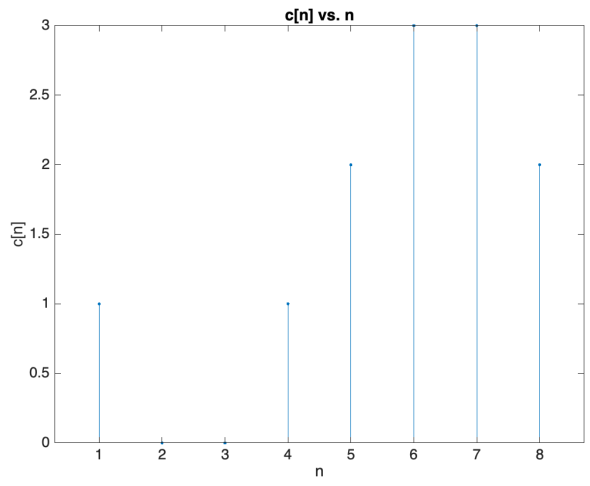


```
stem(n, circshift(x_arr3, 0), '.');
title('x[n] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n]', 'FontSize', 16);

stem(n, circshift(x_arr3, 1), '.');
title('x[n-1] vs. n', 'FontSize', 16);
```

```
xlabel('n', 'FontSize', 16);
ylabel('x[n-1]', 'FontSize', 16);

stem(n, circshift(x_arr3, 4), '.');
title('x[n-4] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-4]', 'FontSize', 16);

stem(n, circshift(x_arr3, 7), '.');
title('x[n-7] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-7]', 'FontSize', 16);
```

We see here that the signals we obtain through our `circshift(x_arr, shift_val)` implementation are exactly the same as our initial periodic time shift implementation in parts **(b)** and **(c)**.

e) In this part, we will implement periodic convolution $c[n] = x[n]\tilde{*}y[n]$. We will leverage Matlab's `circshift` command in order to convolve with period signals, considering periodic convolution is between two signals of infinite length with values repeating every N units, where N is the period of the signal. We observe the periodic convolution $c[n]$ below:



```
N = 8;
c_arr = zeros(1, N);
% Remember that convolution has to happen over any one period.
% 1:N is convenient
for idx = 1:N
    % First flip
    y_flipped = y_arr(N:-1:1);

    % Then advance using circshift
    y_shifted = circshift(y_flipped, idx-1);
    c_arr(idx) = sum(x_arr.*y_shifted);
end

stem(1:N, c_arr, '.')
title('c[n] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('c[n]', 'FontSize', 16);
```

f) We can compute the discrete time Fourier transform coefficients, a_k of $x[n]$ and b_k of $y[n]$, using the Matlab command `fft`. It is important to note the Matlab equations for DTFS,

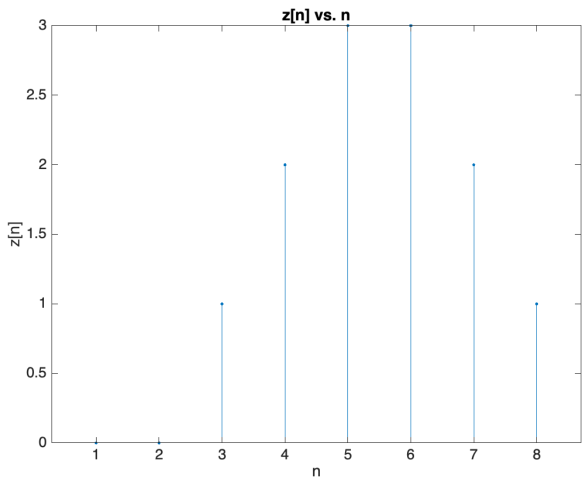
$$a_k^{\text{Matlab}} = \sum_{n=1}^{N} x[n]e^{-j\omega_0(k+1)(n-1)} \tag{1}$$

and

$$x[n] = \frac{1}{N}\sum_{k=1}^{N} a_k^{\text{Matlab}} x[n]e^{-j\omega_0(k-1)(n+1)}, \tag{2}$$

are slightly different than our typical application. Using `fft`, we can obtain said coefficients. We can then compute $c_k = a_k b_k$, and the signal $z[n]$ that has c_k as its coefficients through the `ifft` command.

g) We can observe the plot of $z[n]$ below:



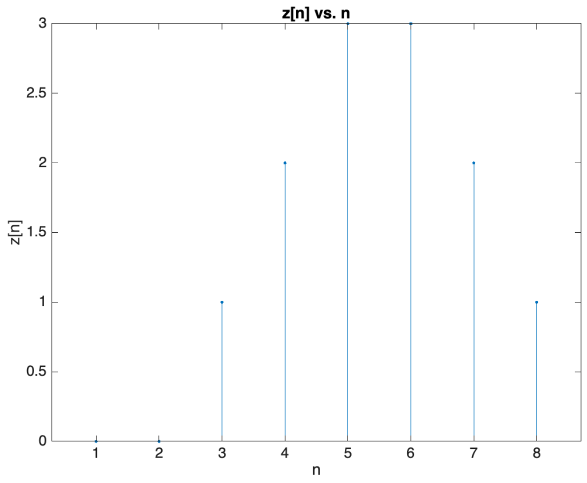
```
a_k = fft(x_arr);
b_k = fft(y_arr);
c_k = a_k .* b_k;
z_arr = ifft(c_k);

stem(1:N, z_arr, '.');
title('z[n] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
```

```
ylabel('z[n]', 'FontSize', 16);
```

We can then conclude it is similar to the one we obtained in **(e)**, its only difference being the time shift. We can directly compare values to analyze this, and we can observe

$$z[n] = c[n-1]. \tag{3}$$

h) Comparing this to a linear convolution, which assumes the signal is aperiodic, we can obtain the following plot:



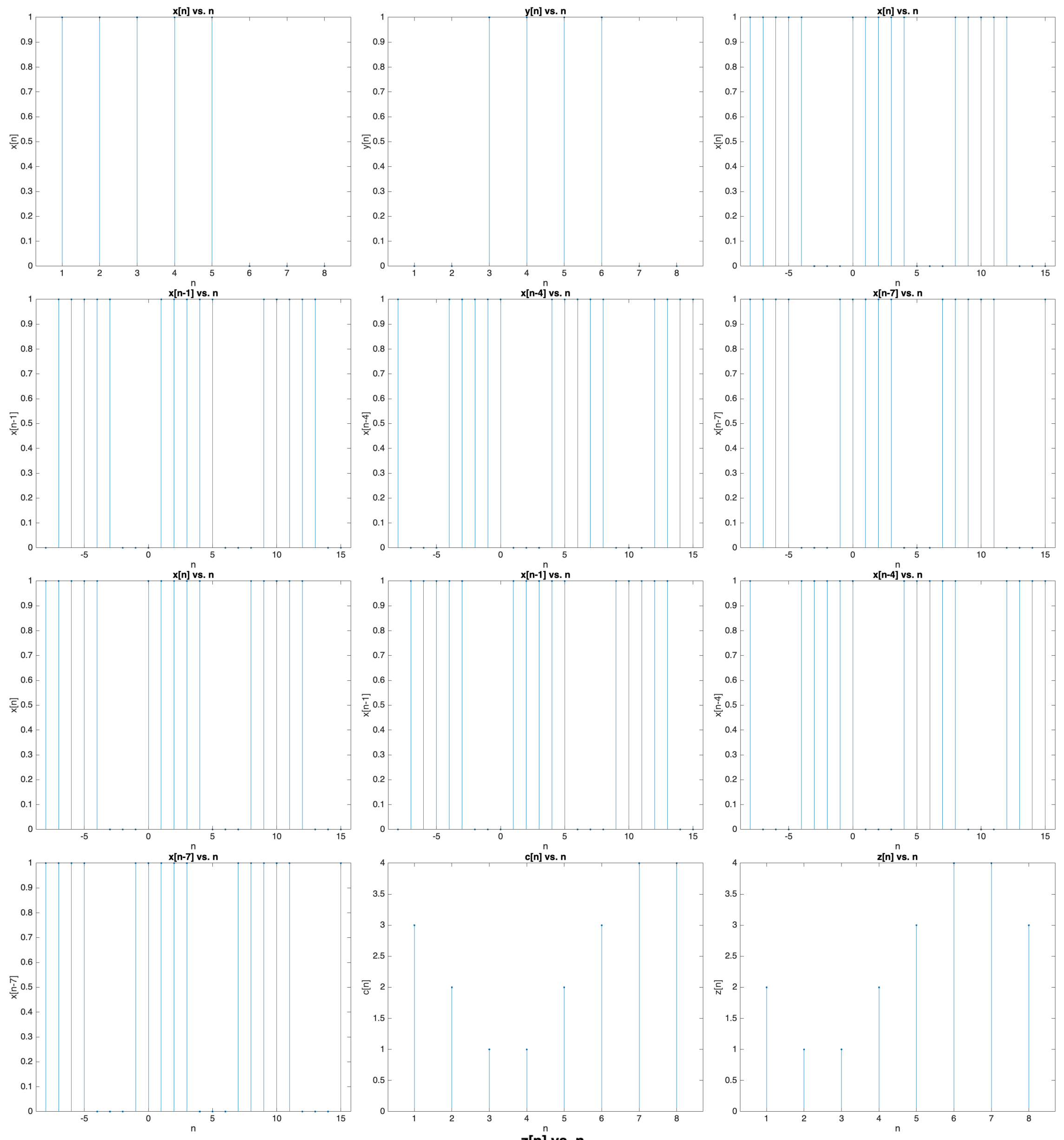
```
z_conv_linear = conv(x_arr, y_arr);
z_conv_linear = z_conv_linear(1:N)

stem(1:N, z_conv_linear, '.');
title('z[n] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('z[n]', 'FontSize', 16);
```

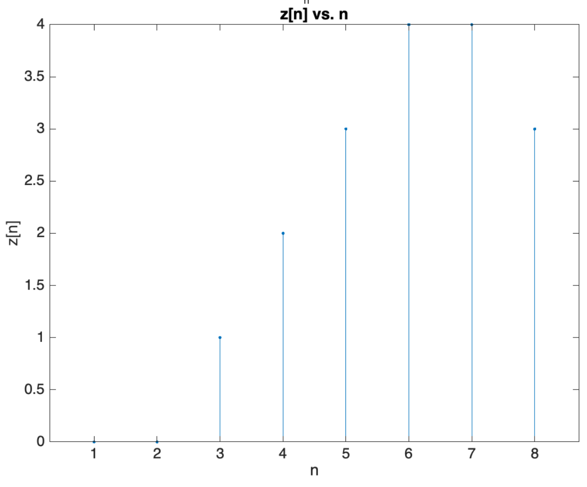
This plot is exactly the same as the one obtained in **(f)** and **(g)**, where it is

$$z[n] = c[n-1] \tag{4}$$

in relation to the one obtained in **(e)**.

i) Repeating parts **(a)** through **(h)** with the new signal, we can analyze the plots pictured below, and we can observe a similar relationship between each convolution method. However, one significant difference for the linear convolution result is the loss of data in this method. At the beginning of the signal, we have zeros where the other plots have non-zero values, which was not as apparent in the initial attempts as our values there were meant to be zeros. With this new $x[n]$, however, they are not meant to be zeros. We can verify this as both other methods give non-zero values for $n = 1$ and $n = 2$. They are still equivalent results for all other values, however.


x[n] vs. n
y[n] vs. n
x[n] vs. n
x[n-1] vs. n
x[n-4] vs. n
x[n-7] vs. n
x[n] vs. n
x[n-1] vs. n
x[n-4] vs. n
x[n-7] vs. n
c[n] vs. n
z[n] vs. n


z[n] vs. n
z[n]
n

```matlab
clear
n = 1:8;
x_arr = zeros(1,8);
y_arr = zeros(1,8);
x_arr(1:5) = 1;
y_arr(3:6) = 1;

stem(n, x_arr, '.');
title('x[n] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n]', 'FontSize', 16);

stem(n, y_arr, '.');
title('y[n] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('y[n]', 'FontSize', 16);

n = -8:15;
x_arr3 = zeros(1, length(n));
for i = 0:2
    x_arr3(8*i+1:8*(i+1))=x_arr;
end

stem(n, x_arr3, '.');
title('x[n] vs. n', 'FontSize', 16)
xlabel('n', 'FontSize', 16);
ylabel('x[n]', 'FontSize', 16);

x_ext=x_arr3;
x_ext(25:48)=x_arr3;

shiftval=1;
stem(n, x_ext(9-shiftval:32-shiftval), '.');
title('x[n-1] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-1]', 'FontSize', 16);

shiftval=4;
stem(n, x_ext(9-shiftval:32-shiftval), '.');
title('x[n-4] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-4]', 'FontSize', 16);

shiftval=7;
stem(n, x_ext(9-shiftval:32-shiftval), '.');
title('x[n-7] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
```

```
ylabel('x[n-7]', 'FontSize', 16);

stem(n, circshift(x_arr3, 0), '.');
title('x[n] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n]', 'FontSize', 16);

stem(n, circshift(x_arr3, 1), '.');
title('x[n-1] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-1]', 'FontSize', 16);

stem(n, circshift(x_arr3, 4), '.');
title('x[n-4] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-4]', 'FontSize', 16);

stem(n, circshift(x_arr3, 7), '.');
title('x[n-7] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('x[n-7]', 'FontSize', 16);

N = 8;
c_arr = zeros(1, N);
% Remember that convolution has to happen over any one period.
% 1:N is convenient
for idx = 1:N
    % First flip
    y_flipped = y_arr(N:-1:1);

    % Then advance using circshift
    y_shifted = circshift(y_flipped, idx-1);
    c_arr(idx) = sum(x_arr.*y_shifted);
end

stem(1:N, c_arr, '.')
title('c[n] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('c[n]', 'FontSize', 16);

a_k = fft(x_arr);
b_k = fft(y_arr);
c_k = a_k .* b_k;
z_arr = ifft(c_k);

stem(1:N, z_arr, '.');
title('z[n] vs. n', 'FontSize', 16);
```

```
xlabel('n', 'FontSize', 16);
ylabel('z[n]', 'FontSize', 16);

z_conv_linear = conv(x_arr, y_arr);
z_conv_linear = z_conv_linear(1:N);

stem(1:N, z_conv_linear, '.');
title('z[n] vs. n', 'FontSize', 16);
xlabel('n', 'FontSize', 16);
ylabel('z[n]', 'FontSize', 16);
```