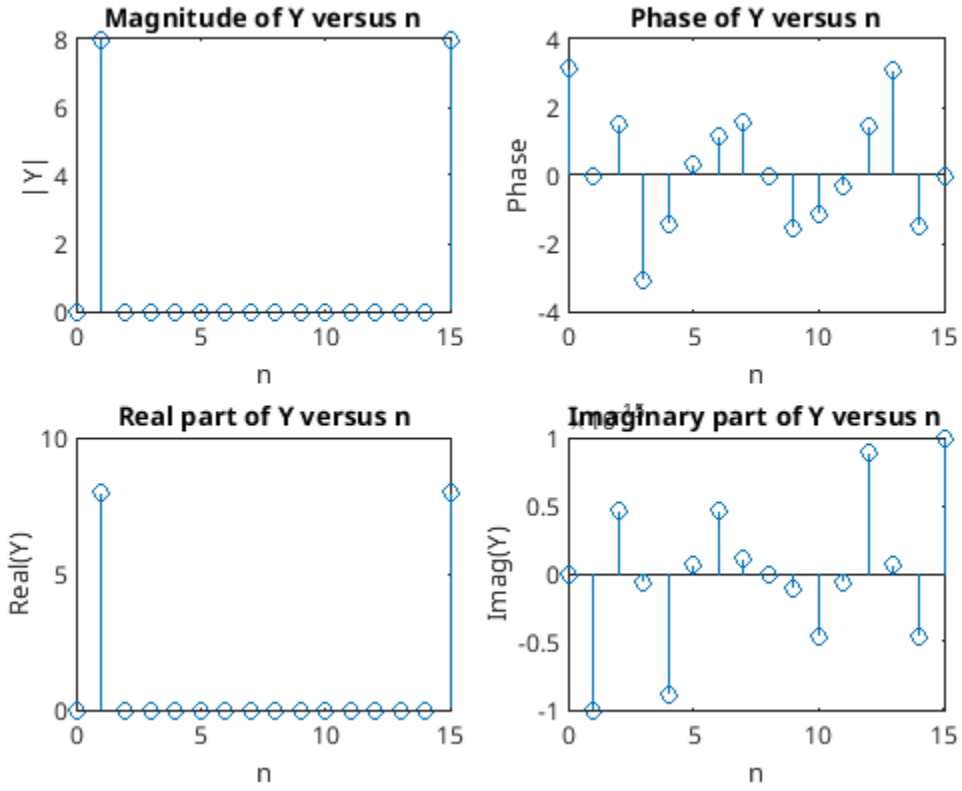
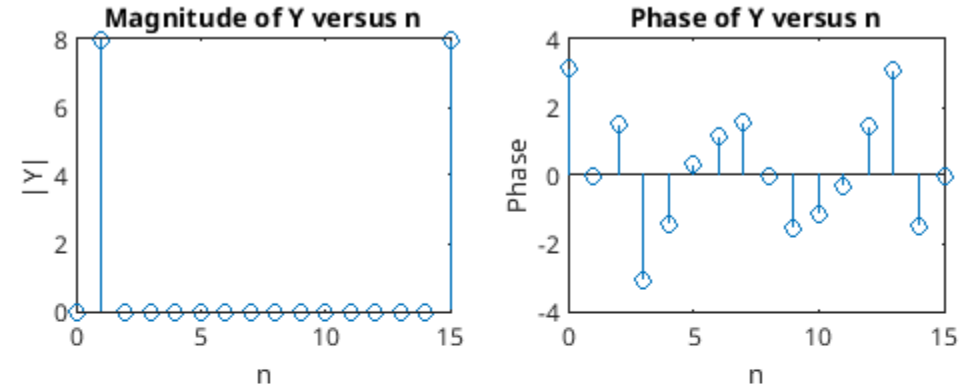
```
N=16;
n=0:(N-1);
C=1;
xn=cos(2*pi*C*n/N);
figure(1)
%stem(xn)
% There appears to be one full period in this range

% FFT
Y = fft(xn);

figure(2)
% Plot magnitude of Y versus n
subplot(2,2,1);
stem(n, abs(Y));
title('Magnitude of Y versus n');
xlabel('n');
ylabel('|Y|');

% Plot phase of Y versus n
subplot(2,2,2);
stem(n, angle(Y));
title('Phase of Y versus n');
xlabel('n');
ylabel('Phase');

% Plot real part of Y versus n
subplot(2,2,3);
stem(n, real(Y));
title('Real part of Y versus n');
xlabel('n');
ylabel('Real(Y)');

% Plot imaginary part of Y versus n
subplot(2,2,4);
stem(n, imag(Y));
title('Imaginary part of Y versus n');
xlabel('n');
ylabel('Imag(Y)');

%{
The 2 non-zero terms in the magnitude plot represent the frequency
components of the input. We have exactly 2 components because the cosine
function has a specific individual frequency. This is split into positive
and negative components because of symmetry in the frequency domain.

We have one component at an index of 1 because this corresponds to the
fundamental frequency of the input signal. Our fundamental frequency should
be 1/16 in this case, this could be a harmonic?

We have another component at index of 15 because this represents the
periodic frequencies wrapping around. This would be a frequency of 15/16,
```

```
or -1/16.

The magnitude of the non-zero components would be 8 because we normalize
our FFT by the length of our input - which is 16 samples long.
%}

%{
If the sampling frequency is 10khz:
Frequency Resolution: fs/N=10000/16=625hz
Length of the original data:N/fs=16/10000=1.6e-3 seconds
Actual Frequency: (C*fs)/N=10000/16=625Hz
%}
```

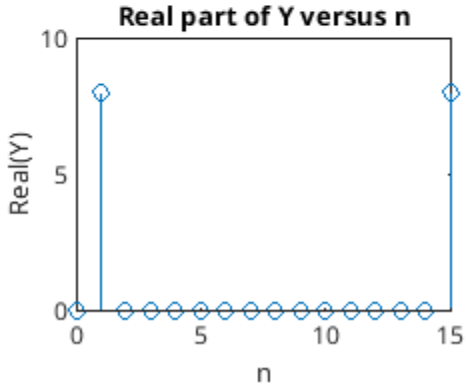


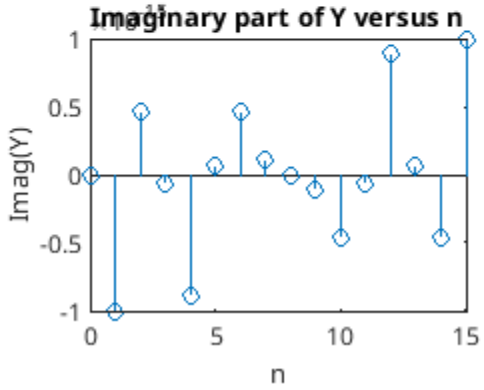

```
xn=sin(2*pi*C*n/N);

Y=fft(xn);

figure(3)
% Plot magnitude of Y versus n
subplot(2,2,1);
stem(n, abs(Y));
title('Magnitude of Y versus n');
xlabel('n');
ylabel('|Y|');

% Plot phase of Y versus n
subplot(2,2,2);
stem(n, angle(Y));
title('Phase of Y versus n');
xlabel('n');
ylabel('Phase');

% Plot real part of Y versus n
subplot(2,2,3);
stem(n, real(Y));
title('Real part of Y versus n');
xlabel('n');
ylabel('Real(Y)');
```

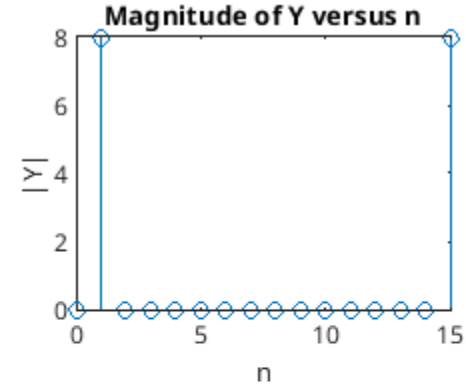
```
% Plot imaginary part of Y versus n
subplot(2,2,4);
stem(n, imag(Y));
title('Imaginary part of Y versus n');
xlabel('n');
ylabel('Imag(Y)');

%{
The magnitude of Y does not appear to change.
This is likely v=because the FFT will just compute the magnitude of the
frequency components. Since frequency doesn't change, neither will
magnitude

Y =

  Columns 1 through 10

  -0.0000 + 0.0000i  -0.0000 - 8.0000i  -0.0000 - 0.0000i   0.0000 -
0.0000i   0.0000 - 0.0000i   0.0000 - 0.0000i   0.0000 - 0.0000i   0.0000 +
0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Columns 11 through 16

   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
-0.0000 + 0.0000i  -0.0000 + 8.0000i

We see that Y is entirely imaginary.
%}
```

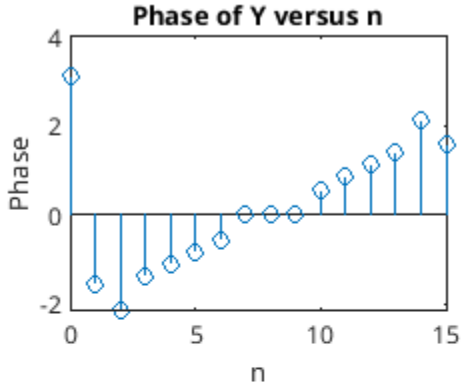

Magnitude of Y versus n
|Y|
n

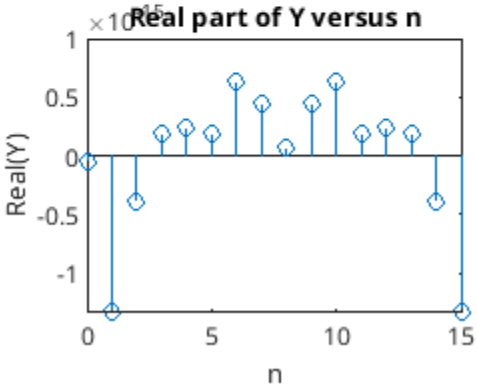
Phase of Y versus n
Phase
n

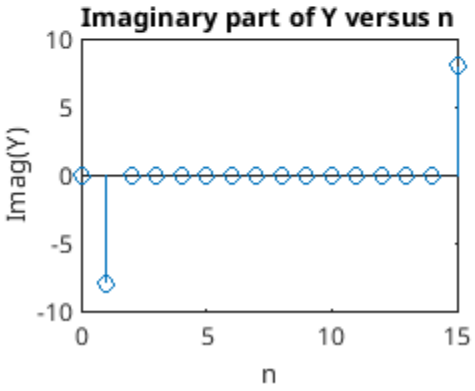

Real part of Y versus n
Real(Y)
n


Imaginary part of Y versus n
Imag(Y)
n


Magnitude of Y versus n
|Y|
n

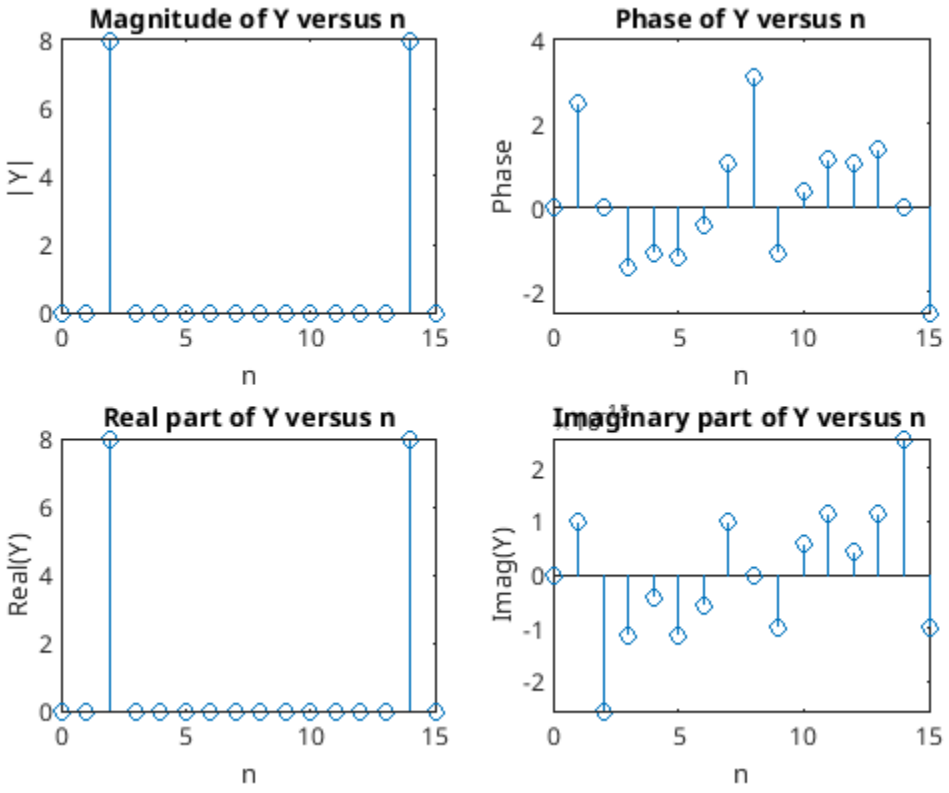

Phase of Y versus n
Phase
n

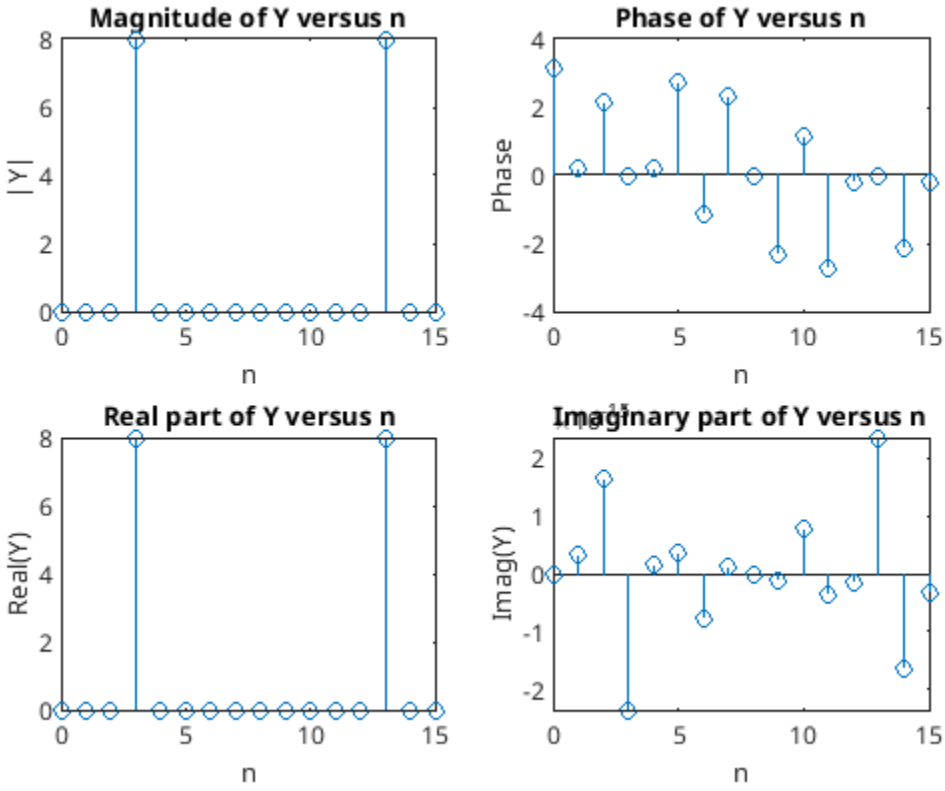

Real part of Y versus n
Real(Y)
n


Imaginary part of Y versus n
Imag(Y)
n

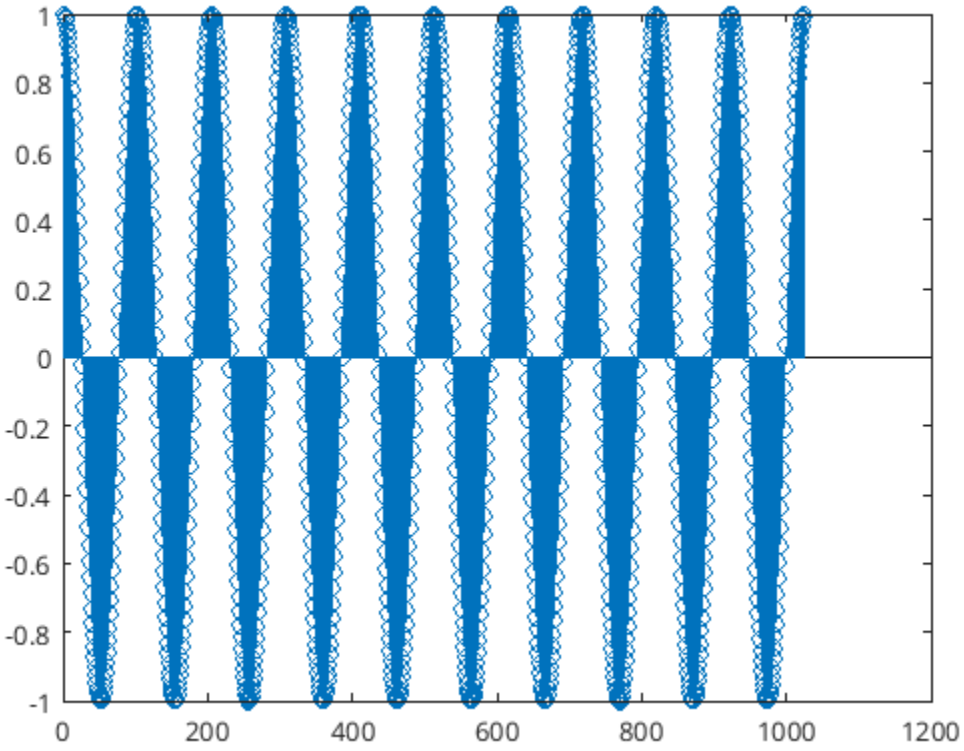
```
xn=cos(2*pi*2*n/16);

Y=fft(xn);

figure(4)
% Plot magnitude of Y versus n
subplot(2,2,1);
stem(n, abs(Y));
title('Magnitude of Y versus n');
xlabel('n');
ylabel('|Y|');

% Plot phase of Y versus n
subplot(2,2,2);
stem(n, angle(Y));
title('Phase of Y versus n');
xlabel('n');
ylabel('Phase');

% Plot real part of Y versus n
subplot(2,2,3);
stem(n, real(Y));
title('Real part of Y versus n');
xlabel('n');
ylabel('Real(Y)');

% Plot imaginary part of Y versus n
subplot(2,2,4);
stem(n, imag(Y));
title('Imaginary part of Y versus n');
xlabel('n');
ylabel('Imag(Y)');


% C=3

xn=cos(2*pi*3*n/16);

Y=fft(xn);

figure(5)
% Plot magnitude of Y versus n
subplot(2,2,1);
stem(n, abs(Y));
title('Magnitude of Y versus n');
xlabel('n');
ylabel('|Y|');

% Plot phase of Y versus n
subplot(2,2,2);
stem(n, angle(Y));
title('Phase of Y versus n');
xlabel('n');
ylabel('Phase');
```

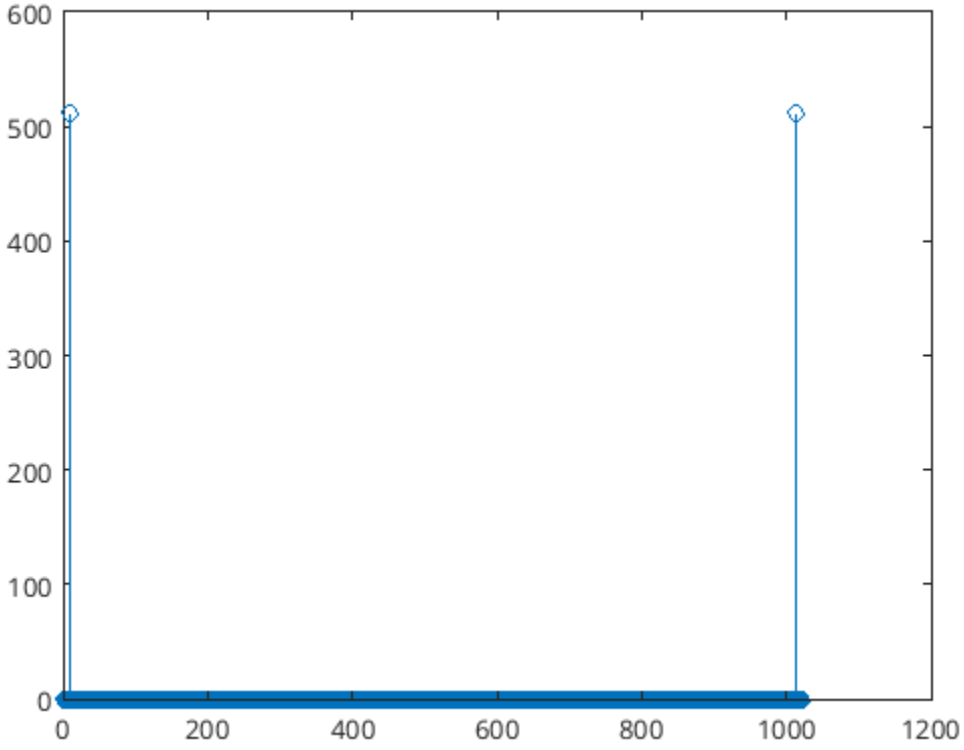
```
% Plot real part of Y versus n
subplot(2,2,3);
stem(n, real(Y));
title('Real part of Y versus n');
xlabel('n');
ylabel('Real(Y)');

% Plot imaginary part of Y versus n
subplot(2,2,4);
stem(n, imag(Y));
title('Imaginary part of Y versus n');
xlabel('n');
ylabel('Imag(Y)');

%{
Here, we see that changing C to 2 shifted the frequency response from
indeces 1 and 15 to indeces 2 and 14. The phase and imaginary plots are
slightly different as well, because of the change in frequency.

Changing C to 3 shifts the indeces again to 3 and 13, and because of the
change in frequency we will again see a change in phase and imaginary
component.
%}
```



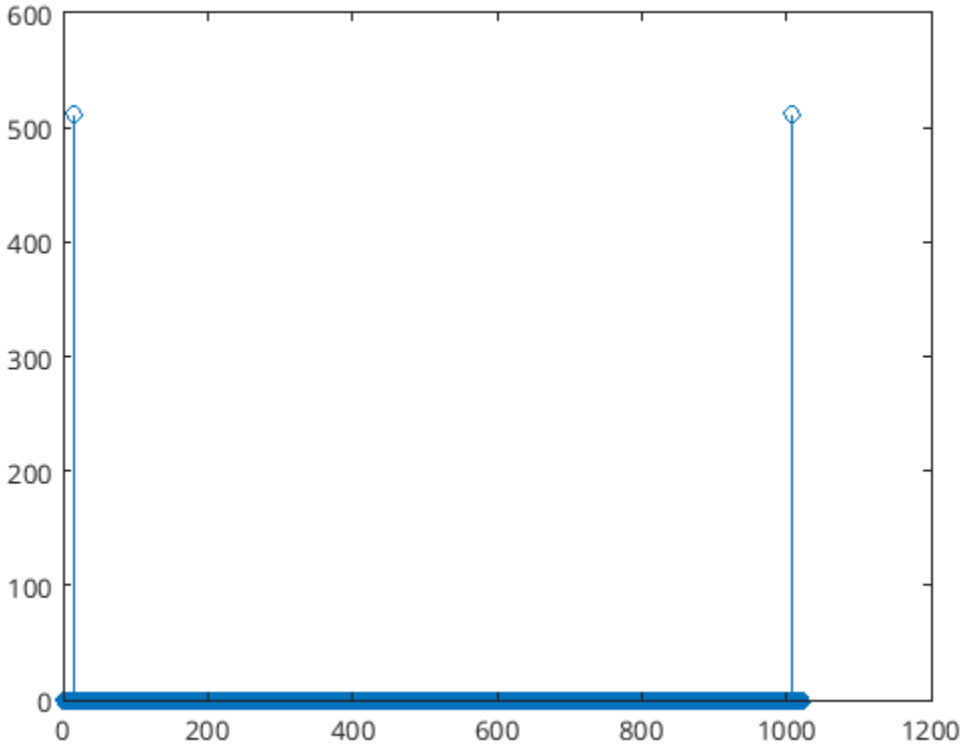



```
N=1024;
C=10;
n=0:(N-1);
xn=cos(2*pi*C*n/N);
figure(6);
stem(xn);
% There appear to be 11 samples

Y=fft(xn);

figure(7);
stem(n,abs(Y))
% There is one peak at X=1014 Y=512
```

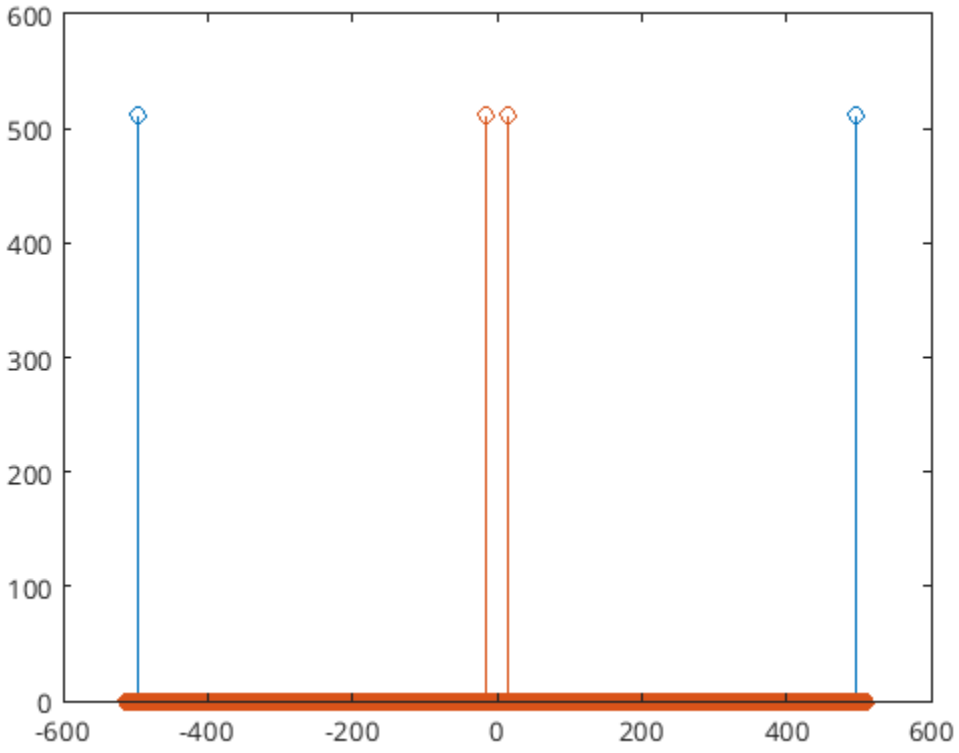

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000
1200


600
500
400
300
200
100
0
0
200
400
600
800
1000
1200

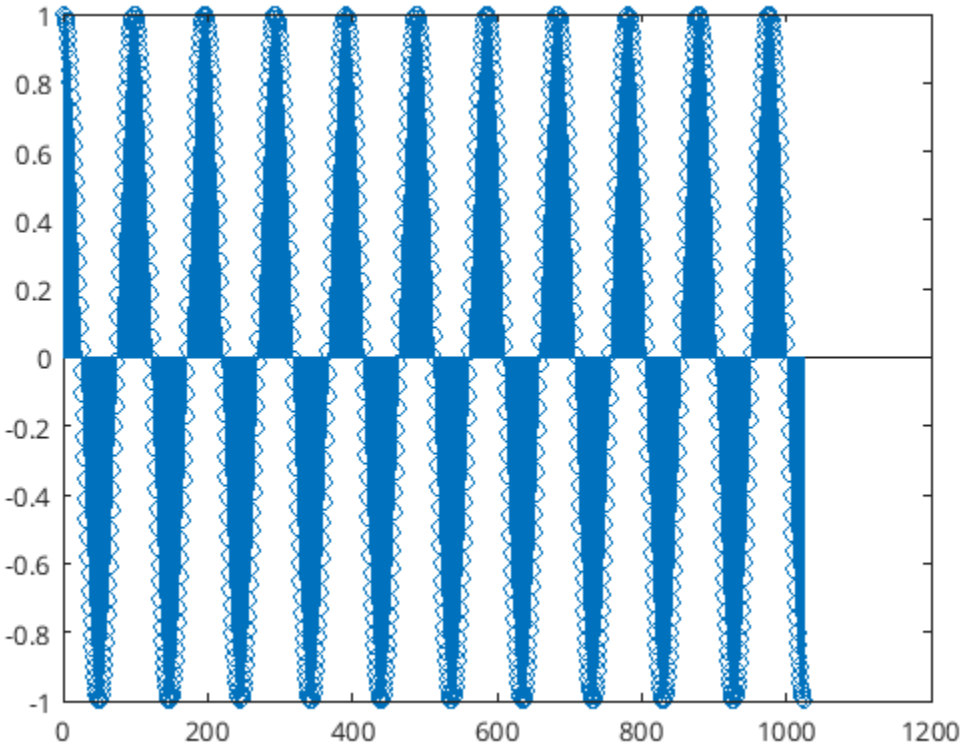
```
C=15;
xn=cos(2*pi*C*n/N);
figure(8)
Y=fft(xn);
stem(n,abs(Y))
% This time the peak is at (1009, 512)
```

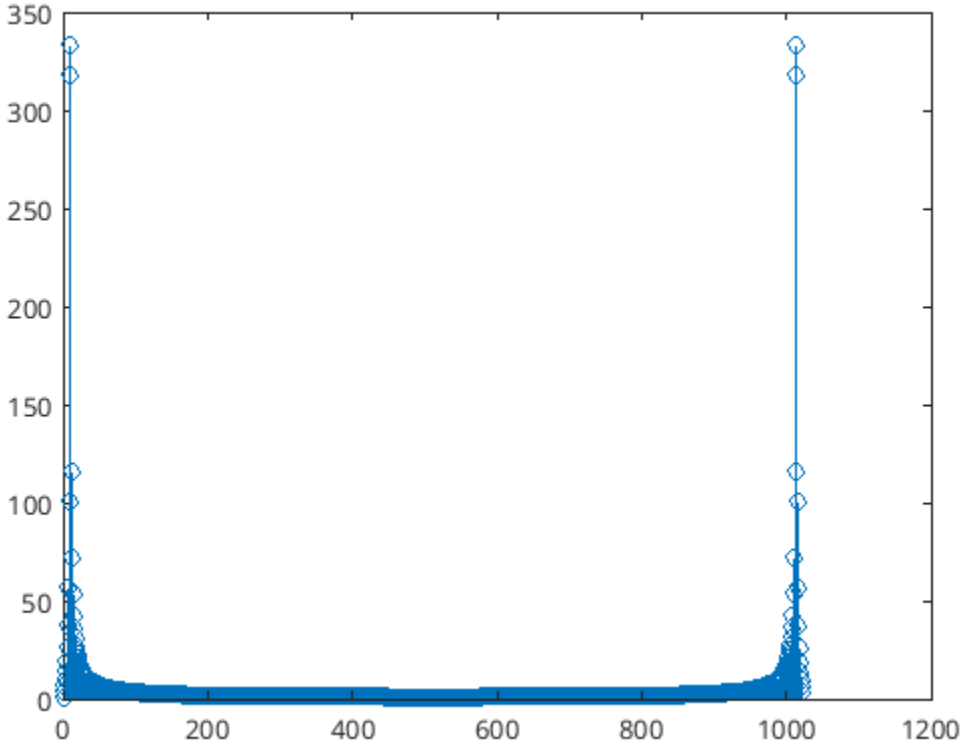



```
n=(-N/2:(N/2-1));
xn=cos(2*pi*C*n/N);
figure(9)
Y=fft(xn);
stem(n,abs(Y))
hold on;
fftshift(Y);
stem(n, abs(fftshift(Y)))
hold off;
```

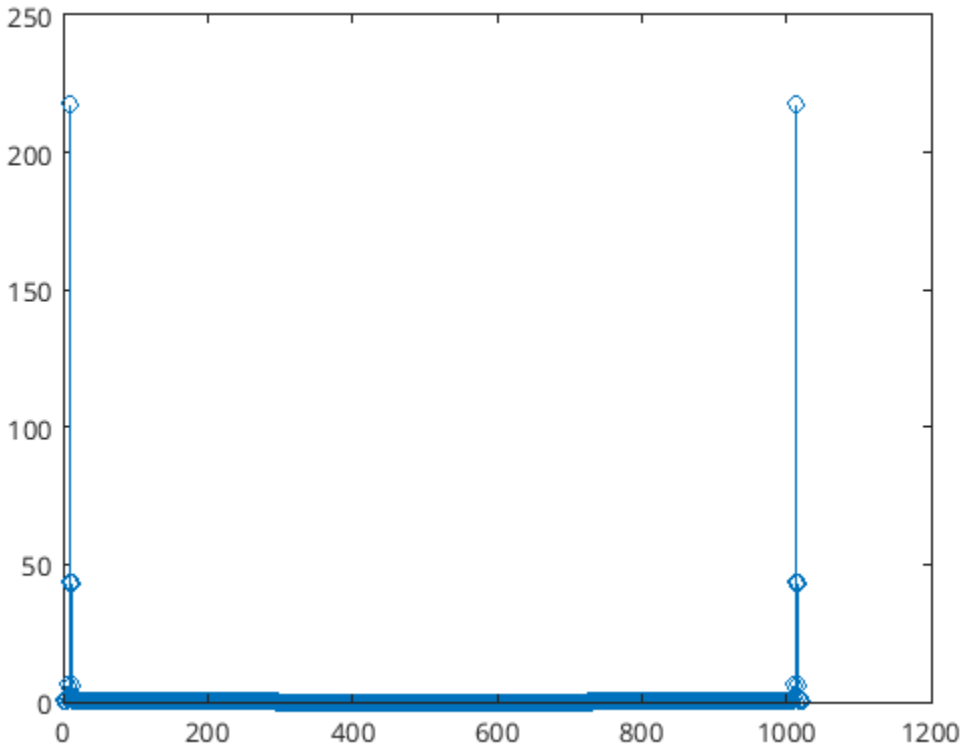



```
N=1024;
C=10.5;
n=0:(N-1);
xn=cos(2*pi*C*n/N);
Y=fft(xn);
figure(10)
stem(xn)
figure(11)
stem(n, abs(Y))

%{
This "bin leakage" is clearly not an accurate representation for the
frequency of the system. There are frequency components that are
represented in the graph that are not present, and if this were a more
complicated signal with smaller frequency components, the bin leakage could
likely obscure some of those smaller components.
%}
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000
1200

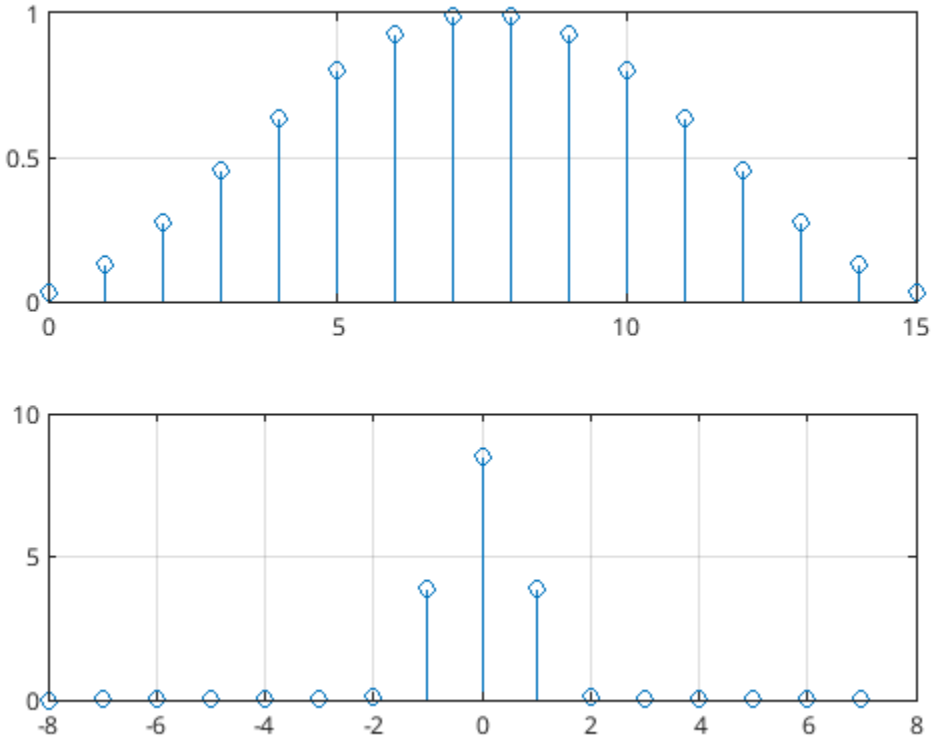

350
300
250
200
150
100
50
0
0
200
400
600
800
1000
1200

```
y=xn.*hanning(N).';
Y=fft(y);
figure(12)
stem(n, abs(Y))


%{
The Hanning window seems to "smooth" the window. Since it is applied to the
signal before it is transformed, it must apply this "smoothing" affect in
the time domain. It seems to reduce the leakage from the previous plot and
make the side lobes slightly smaller.
%}
```




```
clear; clf;
subplot(2,1,1);
N=16;
n=0:(N-1);
x=hanning(N);
stem(n,x);
grid;

subplot(2,1,2)
nn=-N/2:(N/2-1);
X=fftshift(fft(hanning(N)));
stem(nn, abs(X));
grid;
```

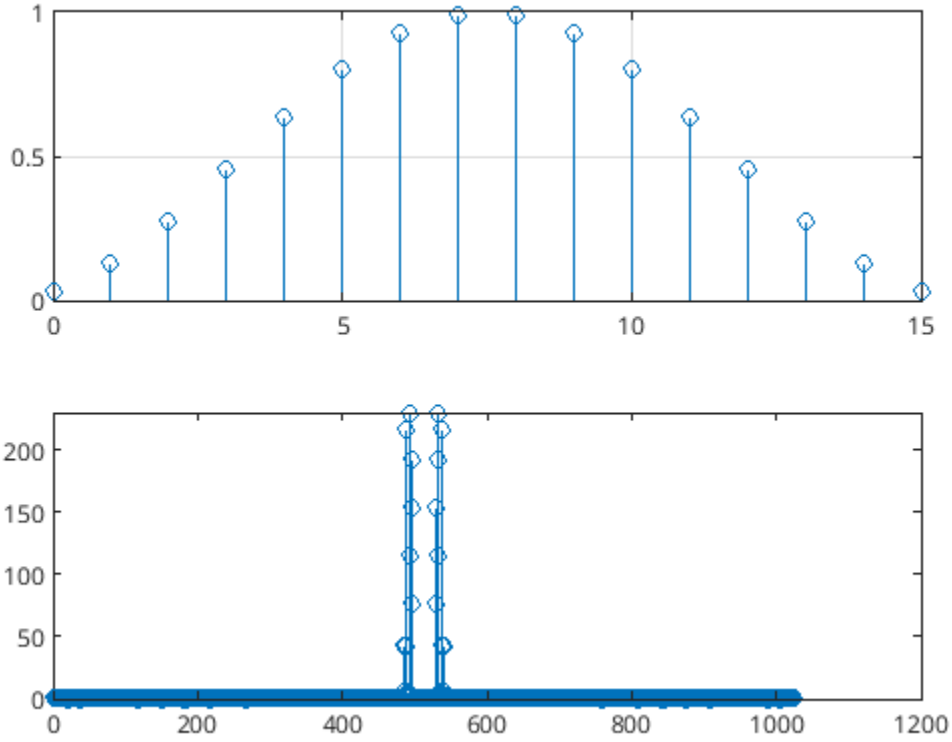
```
%{
Dot multiplying a time domain vector by a data taper window (hanning
window) should be how we achieve windowing.
It should effectively "taper" the edges
%}
```




```
load('hunt.mat')
N=1024;
windowed=Signal.*hanning(N).';
Y=abs(fftshift(fft(windowed))); % Plotting this isn't helpful.

% Attempt 1
%[~, i]=findpeaks(Y); % Find local peaks from Y and output their index
%magnitudes=abs(Y(i));
% Originally I got 280 contacts.

% Attempt 2
[~, i]=findpeaks(Y, MinPeakHeight=0.000001*max(abs(Y)));
fprintf('Number of target: %d\n', length(i))
% Now I get 4.
% This seems more reasonable. Should check graph to visually verify
stem(Y)
% Graph shows more than 4, so I adjusted multiplier from 0.1 to 0.00001.
% New answer: 6. Adding another 0 changes to 22, and 1 more changes to 218.
% This seems like a drastic change - 22 seems like the most reasonable
% answer (Since I can't imaging 218 being correct, but overshooting seems
% like the best option. I can't think of any mathematical way to determine
```

```
% what I should choose as a multiplier, but this does seem as though I
% should be able to determine a logical multiplier in a simpler fashion

% Contacts: 22
magnitudes=abs(Y(i));
frequencies=Y(i);
%{
magnitudes =

  Columns 1 through 19

    0.0256    0.0002    0.0003    0.0003    0.0003    0.0003    0.0004
0.0004    0.0005  217.4413  230.5689  230.5689  217.4413    0.0005
0.0004    0.0004    0.0003    0.0003    0.0003

  Columns 20 through 22

    0.0003    0.0002    0.0256


frequencies =

  Columns 1 through 19

    0.0256    0.0002    0.0003    0.0003    0.0003    0.0003    0.0004
0.0004    0.0005  217.4413  230.5689  230.5689  217.4413    0.0005
0.0004    0.0004    0.0003    0.0003    0.0003

  Columns 20 through 22

    0.0003    0.0002    0.0256
%}
```

Number of target: 22




```
%{
Windows are used to mitigate two primary issues: spectral leakage and
frequency resolution.
Spectral leakage occurs when the frequency ocntent of a signal is
misaligned from the discrete frequency "bins" in the FFT. This can cause
energy from the signal to "leak" into adjacent frequency bins. Windows are
used to "taper" the edges of the signal, reducing this leakage at the
borders of a signal and suppressing spectral leakage. Windows can help to
improve frequency resolution by narrowing the main lobe of the window's
frequency response - this can help the user to more easily differentiate
between tightly spaced components.

In this project, we explored relative advantages and drawbacks to using
rectangular windows vs hanning windows. A rectangular window can provide
poor frequency resolution because of its wide main lobe and tendency to
induce spectral leakage. While it seemed to do a good job representing the
original signal amplitude, it made it somewhat difficult to interpret the
frequency information. Hanning windows provide better frequency resolution
by reducing spectral leakage and tapering the window. Hanning windows do
perform slightly worse when it comes to representing signal amplitude
however.

Using these concepts, we applied windowing techniques to sonar data in
order to improve frequency resolution and reduce spectral leakage. By
comparing the results obtained with regular and hanning windows, we
observed how the choice of window affects the detection and
```

```
characterization of sonar targets. We were able to determine a number of
hypothetical sonar targets using the improved resolution provided by the
hanning window.

I got some information online when I was looking for context/explanations.
https://community.sw.siemens.com/s/article/window-types-hanning-flattop-
uniform-tukey-and-exponential
https://dsp.stackexchange.com/questions/208/what-should-be-considered-when-
selecting-a-windowing-function-when-smoothing-a-t
%}
```

Published with MATLAB® R2023b