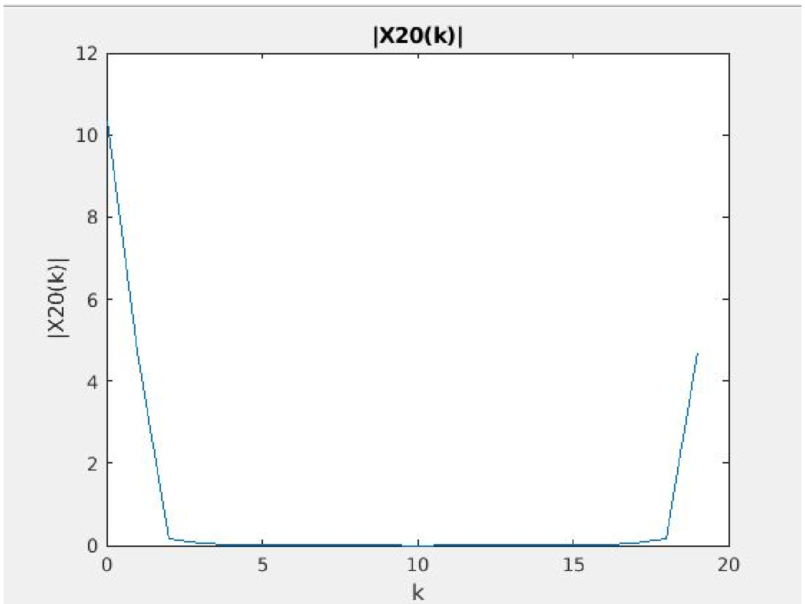
ECE 438 Lab 6b

David Dang & Benedict Lee

2.1 Shifting the Frequency Range

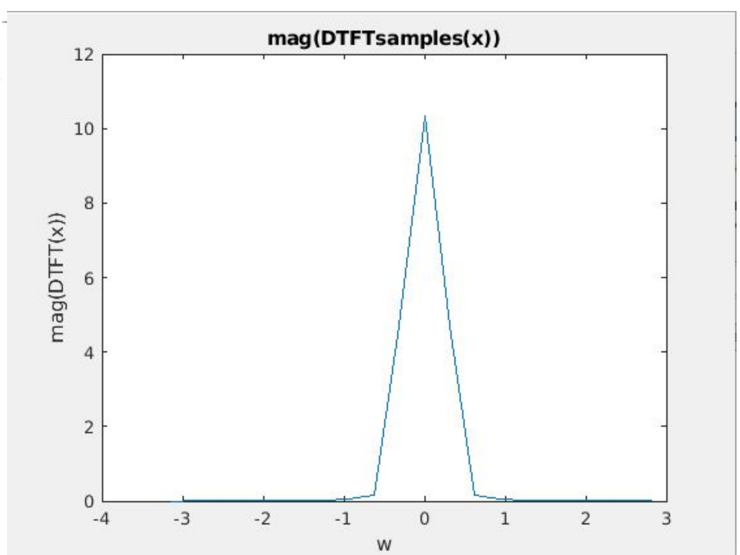
INLAB REPORT: Hand in the plot of the |X20(k)|. Circle the regions of the plot corresponding to low frequency components.



INLAB REPORT:

1. Hand in the code for your function DTFT samples.
2. Hand in the plot of the magnitude of the DTFT samples.

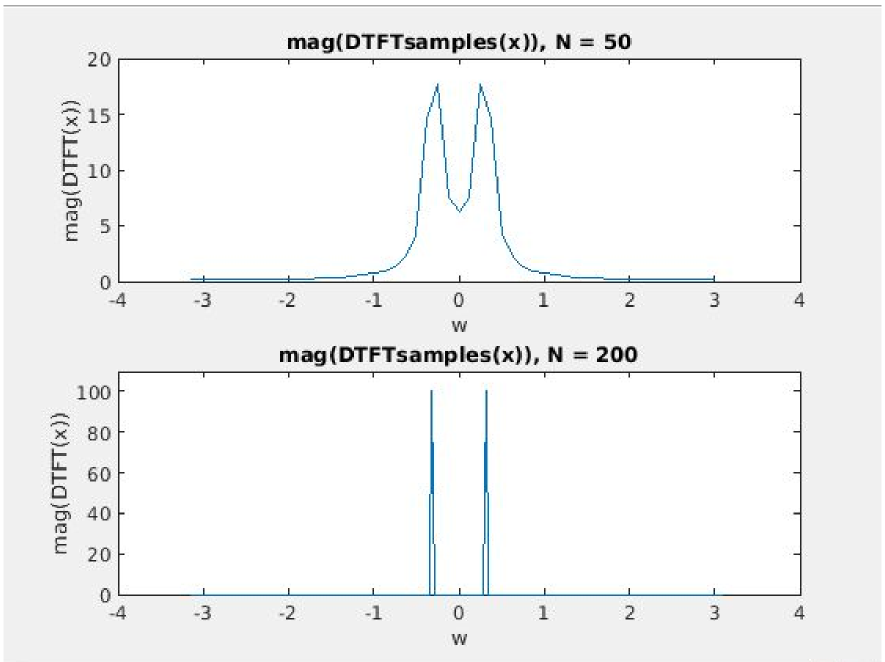
```
function [X,w] = DTFTsamples(x)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
N = length(x);
k = 0:N-1;
w = (2*pi*k)/N;
w(w>=pi) = w(w>=pi)-2*pi;

X = DFTsum(x);

X = fftshift(X);
w = fftshift(w);


end
```



2.2 Zero Padding

INLAB REPORT:

1. Submit your two plots of the DTFT samples for N= 50 and N= 200.
2. Which plot looks more like the true DTFT?
3. Explain why the plots look so different.



- The plot where N = 200 looks more like the true DTFT of sin(0.1πn). The plots look different because there are more zeroes in the plot with N = 200 which provides us with a finer sampling of the DTFT

3.1 Implementation of Divide-and-Conquer DFT

INLAB REPORT: Do the following:

1. Submit the code for your function dcDFT.
2. Determine the number of multiplies that are required in this approach to computing an N point DFT. (Consider a multiply to be one multiplication of real or complex numbers.)

HINT: Refer to the diagram of Fig. 1, and remember to consider the N/2 point DFTs.

```
function [X] = dcDFT(x)
%UNTITLED6 Summary of this function goes here
%   Detailed explanation goes here

j = sqrt(-1);
N = length(x);
k = 0:(N/2 - 1);
x0 = x(1:2:N);
x1 = x(2:2:N);
X0 = DFTsum(x0);
X1 = DFTsum(x1);
Wkn = exp((-j*2*pi*k)/N);
X(1:(N/2)) = X0 + Wkn.*X1;
X((N/2)+1:N) = X0 - Wkn.*X1;


end
```

- There are (N^2)/2 + N multiplies
    1. X0 = x(1:2:N) → N/2
    2. X1 = x(2:2:N) → N/2
    3. 'for' loops in DFTsum → (N^2)/4
    4. DFTsum called twice → (N^2)/2
    5. Wkn multiplied twice → (N^2)/2 + N

3.2 Recursive Divide and Conquer

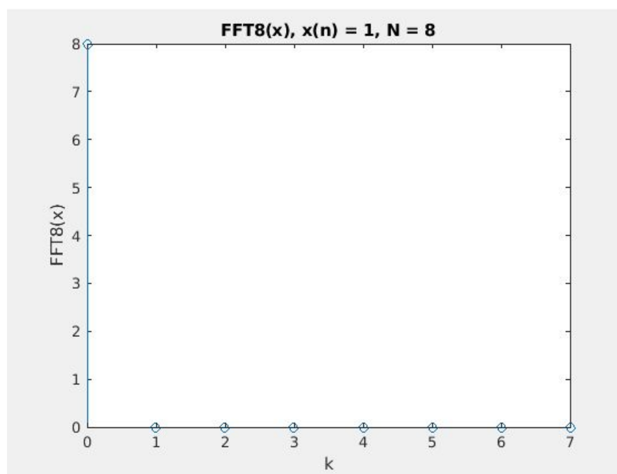
INLAB REPORT:

1. Submit the code for your functions FFT2, FFT4 and FFT8.
2. List the output of FFT8 for the case x(n) = 1 for N= 8.
3. Calculate the total number of multiplies by twiddle factors required for your 8-point FFT. (A multiply is a multiplication by a real or complex number.)
4. Determine a formula for the number of multiplies required for an N= 2^p point FFT. Leave the expression in terms of N and p. How does this compare to the number of multiplies required for direct implementation when p= 10?

```
function [X] = FFT2(x)
%UNTITLED10 Summary of this function goes here
%   Detailed explanation goes here

N = length(x);
X = zeros(1, N);
X(1) = x(1) + x(2);
X(2) = x(1) - x(2);

end
```

```
function [X] = FFT4(x)
%UNTITLED12 Summary of this function goes here
%   Detailed explanation goes here

j = sqrt(-1);
N = length(x);
k = 0:(N/2 - 1);
x0 = x(1:2:N);
x1 = x(2:2:N);
X0 = FFT2(x0);
X1 = FFT2(x1);
Wkn = exp((-j*2*pi*k)/N);
X(1:(N/2)) = X0 + Wkn.*X1;
X((N/2)+1:N) = X0 - Wkn.*X1;


end
```

```
function [X] = FFT8(x)
%UNTITLED13 Summary of this function goes here
%   Detailed explanation goes here

j = sqrt(-1);
N = length(x);
k = 0:(N/2 - 1);
x0 = x(1:2:N);
x1 = x(2:2:N);
X0 = FFT4(x0);
X1 = FFT4(x1);
Wkn = exp((-j*2*pi*k)/N);
X(1:(N/2)) = X0 + Wkn.*X1;
X((N/2)+1:N) = X0 - Wkn.*X1;

end
```



- For the 8-point fft, there are 3N multiplies of the twiddle factors (FFT8 calls FFT4 twice and FFT8 also has twiddle factor multiplies, so (N/2)*6 = 3N)
- For a N = 2^p fft, there are ((2^(p-1)) - 1)*N multiplies of the twiddle factor. If p = 10, there are 511N multiplies of the twiddle factor

INLAB REPORT: Submit the code for your fft_stage function.

```
function [X] = fft_stage(x)
%UNTITLED14 Summary of this function goes here
%   Detailed explanation goes here

N = length(x);

if N == 2
   X = FFT2(x);
   return
elseif N > 2
    j = sqrt(-1);
    N = length(x);
    k = 0:(N/2 - 1);
    x0 = x(1:2:N);
    x1 = x(2:2:N);
    X0 = fft_stage(x0);
    X1 = fft_stage(x1);
    Wkn = exp((-j*2*pi*k)/N);
    X(1:(N/2)) = X0 + Wkn.*X1;
    X((N/2)+1:N) = X0 - Wkn.*X1;
end

end
```