**For each section using MATLAB:**

1. Create a new \*.m (or \*.mlx) file and call init() in the first line. Save it. Remember, no spaces in the file name!
2. Copy last lab’s init.m, fft\_ifft.m, make\_plot.m and make\_stem.m (or \*.mlx) functions to the same directory. Use the init and plot files defining fig\_num as a global so that both make\_plot and make\_stem can use it.

A quiz will be given at the beginning (1st 10 minutes) of each lab covering the content of the prelab. One quiz will be dropped. NO make-up quizzes will be given.

**Prelab:**

1. Based on the script below, determine the value of each expression.
   1. What are the values of t?
   2. How many zeros are in the y1 array?
   3. What are the values of y2 when the script finishes?
   4. What are the values of y3 when the script finishes?

T=5;

t=0:T/5:T

y1=zeros(size(t))

y2=zeros(size(t))

t\_greater\_than\_2=find(t>2);

y2(t\_greater\_than\_2)=3\*t(t\_greater\_than\_2)

y3=zeros(size(t))

t\_between\_1\_and\_3=find(t>=1 & t<=3);

y3(t\_between\_1\_and\_3)=3

**Note: You may change variable names and comments to make them make more sense to you. Section 1 will be used with several homework problems.**

**Section 1: FFT and IFFT –** Create a new script for the code below.

init();

N=16; %number of samples in time and freq domain

n=0:N-1; %index for freq domain.

T=9; %signal period

Ts=T/N; %sample period

t=0:Ts:T-Ts;

1. Calculate Ts and ws (sample angular frequency = 2\*pi/Ts)

Ts = \_\_\_\_\_\_\_\_sec ws = \_\_\_\_\_\_\_\_rad/sec

1. Complete the table for the t array

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.0000 | 0.5625 |  |  |  | 2.8125 | 3.3750 | 3.9375 | 4.5000 | 5.0625 |
| 5.6250 | 6.1875 |  |  | 7.8750 | 8.4375 |

1. Complete the MATLAB code (for 7 < t <= 9) and add it to the script to create the input waveform shown in figure 1.

Note: The plot command in make\_plot.m (or \*.mlx) was **changed from plot(t,y) to plot(t,y,’o’)** to show the data points seen in figure 1. Making this change is recommended, but not required.

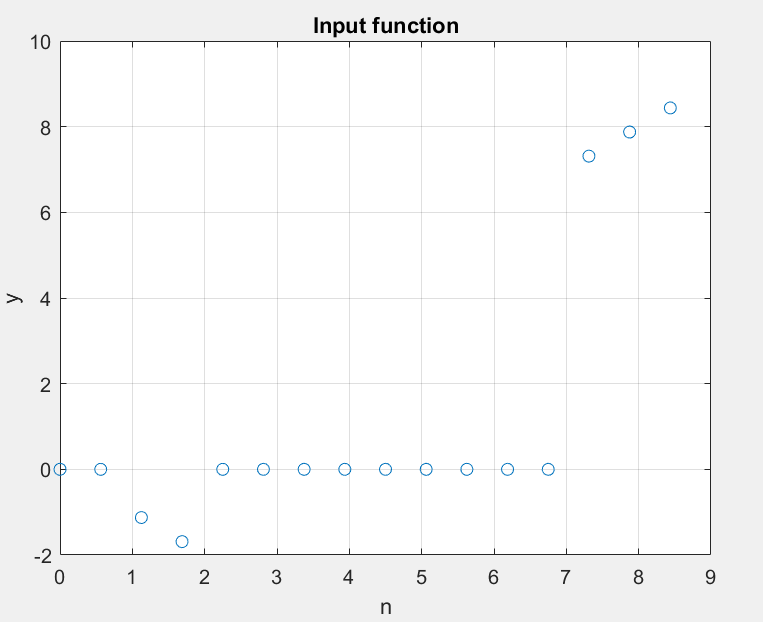
y=zeros(size(t));

t\_between\_1\_and\_2=find(t>=1 & t<=2);

y(t\_between\_1\_and\_2)=-t(t\_between\_1\_and\_2);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

make\_plot(t,y,'Input function','n','y');



1. Copy fft\_ifft.m (or \*.mlx) from last lab and call it from your script after the make\_plot command. Make sure you include returned variables in your function call.

The fft\_ifft function will take the Fourier transform and then inverse Fourier transform (returning the original function). In future labs we will manipulate the Fourier transform data. To do this we need to be able to interpret it and interpretation of the Fourier data is difficult and take practice. Frequency, spectrum and Fourier are all used to refer to the fft data. The subscript m is generally used in the frequency domain and n is used in the time domain. Sometimes m and n are interchanged.

1. Using figure 2 (Spectrum Amplitude) and the “data cursor”, click on the plot to help you complete the list below of cm values (some are given)
2. DC: m = 0 and |cm| = 1.301
3. 1st harmonic: m = 1 and |cm| = 1.44885\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. -1st harmonic: m = 15 and |cm| = 1.449
5. 2nd harmonic: m = 2 and |cm| = 1.33892
6. -2nd harmonic: m = 14 and |cm| = 1.339
7. (T/F) The DC value is the average value of the waveform: false, just the center value

MATLAB’s arrangement of the FFT is confusing and inconvenient. Change the fft\_ifft.m function (or \*.mlx) so it shifts the spectrum putting the DC value in the center. Changes are highlighted.

function [m\_ctr,cm\_ctr,yy] = fft\_ifft(t,~~n,~~y,N)

% Calculate, display F(m).

% NOTE: Matlab fft() returns N times spectrum so N is divided out

% Matlab ifft() used later will scale it back up by N

m\_ctr=-N/2:N/2-1;

cm\_ctr = fftshift(fft(y,N)/N);

make\_stem(m\_ctr,abs(cm\_ctr),'shifted spectrum','m(center)','abs(cm)');

% Reconstruct y (called yy) using inverse FFT (IFFT).

% NOTE: Matlab fft() returns N times spectrum so N is was divided out

% Matlab ifft() now expects fft() scale up by N

yy = real(ifft(N\*fftshift(cm\_ctr))); % scrub imaginary vestiges

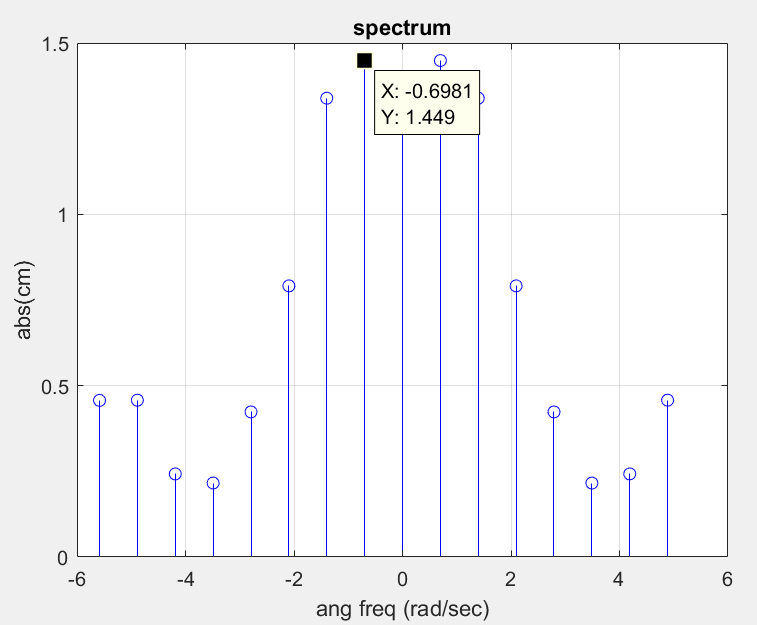
make\_plot(t,yy,'Reconstructed Waveforms','seconds','reconstructed y');

end

1. Repeat the previous exercise using figure 3 (Shifted Spectrum Amplitude) and the “data cursor” to click on the plot. Use these results to complete the list below of cm values.
2. DC: m\_ctr = 0\_ and |cm| = 1.301
3. 1st harmonic: m\_ctr = 1 and |cm| = 1.44885
4. -1st harmonic: m\_ctr = -1 and |cm| = 1.449
5. 2nd harmonic: m\_ctr = 2 and |cm| = 1.33892
6. -2nd harmonic: m\_ctr = -2 and |cm| = 1.339

The spectrum can be further enhanced by displaying the actual frequency and not just harmonic number. The frequency of any coefficient can be calculated by multiplying the harmonic number by the fundamental frequency. Add a make\_stem in your script after the call to fft\_ifft using the angular frequency for the x-axis variable.

1. From figure 4 complete the list below of cm values (plot given below)
2. DC: = 0 and |cm(0)| = 1.301
3. 1st harmonic: = 0.698132 rad/sec and |cm(1)| = 1.44885
4. -1st harmonic: = = - 0.698132 rad/sec and |cm(-1)| = 1.449 (shown on plot)
5. 2nd harmonic: = 1.39626 rad/sec and |cm(2)| = 1.33892
6. -2nd harmonic: = -1.39626 rad/sec and |cm(-2)| = 1.339

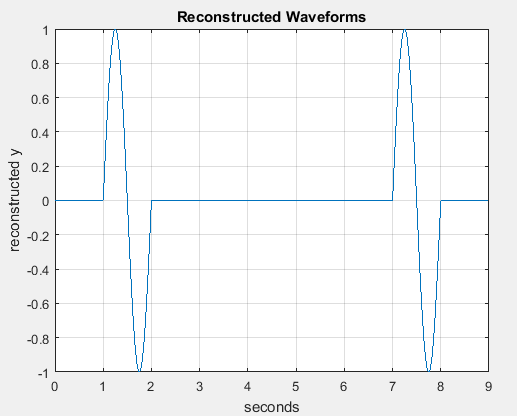


Submit

1. Section 1 blanks completed (hand written ok) in your report.

**Section 2: Other waveforms**

1. Create a new script and copy the script from section 1 into it.
2. Change the function to the sine wave described below.
   1. Set N to 1024.
   2. If you changed make\_plot to contain an ‘o’, remove it.
   3. Write new MATLAB code to generate the y shown below (period of sine pulse is Tp=1) where the function is
   4. Run the program with the new input and print figure 3, the reconstructed waveform.

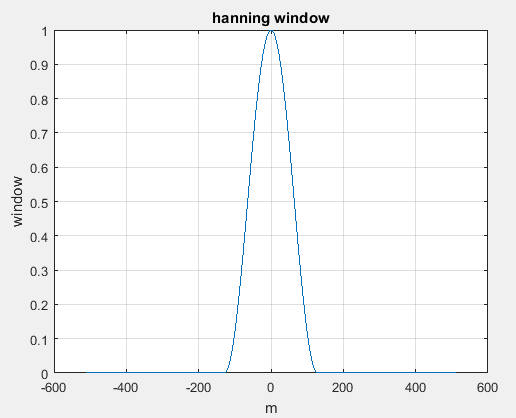


Prepare Figure 3, the reconstructed waveform, for sign-off.

**Section 3: Windowing**

One application for transforms is data compression that is, using less data to describe the original signal. In imaging applications, low frequency data is much more important than high frequency data. Part of the compression process includes tossing some of the high frequency cm values. Doing this can cause an overshoot or ringing at simple discontinuities (Gibb’s phenomenon). Windowing provides a way to remove high frequency terms while, at the same time, smoothing the remaining terms avoiding large discontinuities and ringing. The sample window below will be multiplied by the spectrum. Its gradual decent to zero will prevent ringing.

Windowing requires creating an array the same size as the spectrum (1024 data points in this case) with zero for the high frequency values. In the example below, the values of m = -512 to -128 and 128 to 512 are zero. The window is between -128 and 128 and gradually tappers to zero on both ends.



1. Create a new script and copy the script below into it.

N=16;

M=3;

m=-N/2:N/2-1

cm = % step a

make\_stem(m,cm,'spectrum','m','cm');

win= % step b

m\_between\_negM\_and\_posM= % step c

win( ) = hanning(2\*M+1) % step d

make\_stem(m,win,'window','m','win');

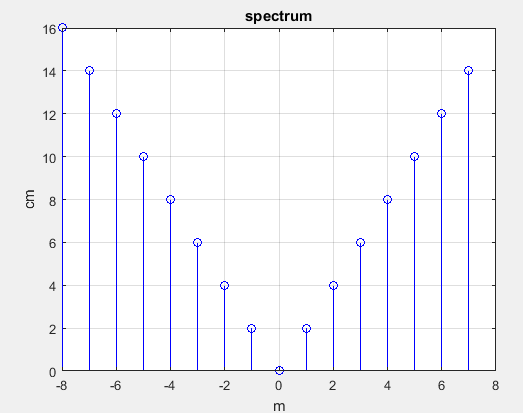
cm\_win= % step e

make\_stem(m,cm\_win,'windowed spectrum','m','cm\_win');

1. Follow the instructions below to practice creating a window. Each step helps you complete a highlighted line of code in the script.
   1. Define cm as a simple function 2\*abs(m). Complete the table and verify that you get the same plot. Remember not to use a semi-colon at the end of the cm assignment so you can see the values of cm in the transcript window.

cm

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 16 | 14 | 12 | 10 | 8 | 6 | 4 | 2 | 0 | 2 |
| 4 | 6 | 8 | 10 | 12 | 14 |



* 1. Define win as a zero vector the same size as cm. To do this refer to previous labs to see how the zeros() and size() function were used. Complete the table and verify that you get the same plot.

win

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |

The window will now be placed in the center of the win vector.

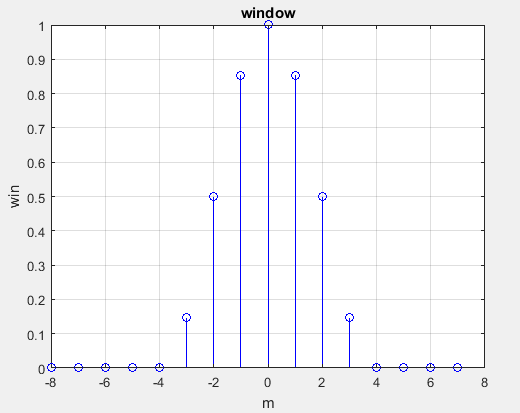
* 1. Use the find() command to locate the m values between –M and M.
  2. Load the Hanning window values in the center of the win vector, from -M to M.

The most common error in this step is “Unable to perform assignment because the left and right sides have a different number of elements.” Make sure the size of “m\_between\_negM\_and\_posM” is equal to 2\*M+1. If not, fix the find command making sure you include the end points, –M and M.

Complete the table and verify that you get the same plot.

win

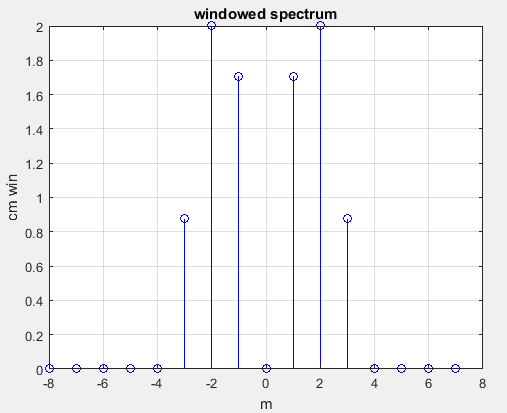
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0.1464 | 0.5000 | 0.8536 | 1.000 | 0.8536 |
| 0.5000 | 0.1464 | 0 | 0 | 0 | 0 |



* 1. Define cm\_win by multiplying the cm and win vectors (remember to use a dot). Complete the table and verify that you get the same plot.

cm\_win

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0.8787 | 2.0000 | 1.7071 | 0 | 1.7071 |
| 2.0000 | 0.8787 | 0 | 0 | 0 | 0 |



In the next step the fft\_ifft function will be modified to include the Hanning windowing function.

1. Copy the fft\_ifft function and call it fft\_hanning\_ifft.
2. Add hanning to the copied function and add Mwin to the parameter list. Mwin is the number of non-zero terms in the window.
3. Use the steps below to complete the highlighted lines of code.
   1. Complete the assignment to win making it the same size as cm\_ctr and filling it with zeros.
   2. Fill the center of win with the hanning window.
   3. Create cm\_ctr by multiplying cm with the new window.
4. Replace cm\_ctr with cm\_ctr\_win in the yy function so the windowed coefficients are used to regenerate yy.

function [m\_ctr,cm\_ctr\_win,yy] = fft\_hanning\_ifft(t,y,N,Mwin)

% Calculate, display F(m).

% NOTE: Matlab fft() returns N times spectrum so N is divided out

% Matlab ifft() used later will scale it back up by N

m\_ctr=-N/2:N/2-1;

cm\_ctr = fftshift(fft(y,N)/N);

make\_stem(m\_ctr,abs(cm\_ctr),'shifted spectrum','m(center)','abs(cm)');

win= ;

win( ) = hanning(2\*Mwin+1)';

cm\_ctr\_win= ;

% Reconstruct y (called yy) using inverse FFT (IFFT).

% NOTE: Matlab fft() returns N times spectrum so N is was divided out

% Matlab ifft() now expects fft() scale up by N

yy = real(ifft(N\*fftshift(cm\_ctr\_win))); % scrub imaginary vestiges

make\_plot(t,yy,'Reconstructed Waveforms','seconds','reconstructed y');

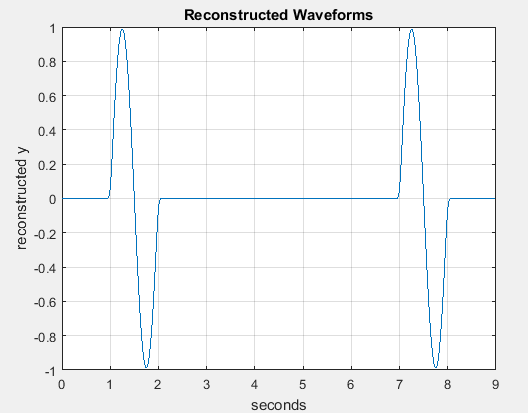
end

1. Edit the script from section 2 so it calls the new windowing fft function (new code below). Rerun the script and verify the plot below.

Mwin=128; %2\*Mwin+1 terms kept in freq domain after windowing

[m\_ctr,cm\_ctr,yy] = fft\_hanning\_ifft(t,y,N,Mwin);

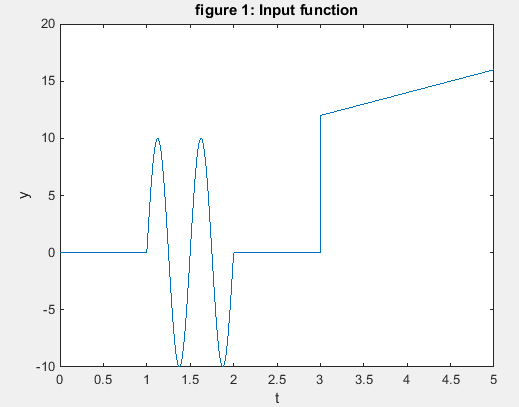
The reconstructed waveform uses +/- 128 point instead of 1024. Compare the original waveform to the reconstructed waveform. Loosing 768 data points (1024 – 128\*2) makes slight differences, especially near fast changing areas like the peak and start of the sine waves. Sending the transformed data and not the original waveform will save time and is nearly lossless.



Prepare figure 4 with N = 1024 and Mwin = 128 for sign-off.

**Section 4: Even more waveforms**

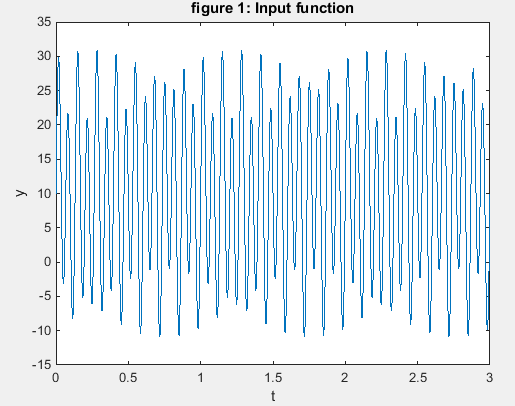
1. Create a new script and copy the script from section 2 (modified in section 4).
2. Write new MATLAB code to generate the y shown below (A=10 & Tsine=1/2) where the function is zero everywhere except the equations below. The function repeats every 5 sec (T=5).
3. Rerun the program with the new input and print figure 6.



Prepare figure 4 with N = 1024 and Mwin=63 for sign-off.

**Section 5: Reading the spectrum**

1. Create a new script and copy the code from the last section.
2. Write new MATLAB code to generate the y shown below. The function repeats every 3 sec (T=3).



1. What is the sample period
2. What is the fundamental frequency

The maximum frequency the system can sample without aliasing is the Nyquist frequency. The system Nyquist frequency is: .

1. (T/F) This systems Nyquist frequency is much greater than the frequency of either of the input sine waves.

Examine the spectrum.

* 1. What is the frequency of the 24th harmonic?
  2. What is m equal to at each of the spikes in the spectrum? Fill in the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -45 |  | 0 | 24 |  |

* 1. Convert these sample numbers to frequency? Fill in the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -15 Hz |  |  | 8 Hz |  |

Submit in your report

1. Section 5 table with the blanks completed (hand written ok).
2. Section 5 spectrum with N = 1024 and M=128.

**Report:**

Create your own cover page.

Submit your cover page, the requested prints (sections 1 and 5 only) and this sign-off sheet on the second page.

All reports due in next schedule lab period. Late penalty: -30%/week.

*Sign-offs*

*Name*

Section 2: Other waveforms

|  |  |  |
| --- | --- | --- |
|  |  | / / |
| Signature |  | Date |

Section 3: Windowing

|  |  |  |
| --- | --- | --- |
|  |  | / / |
| Signature |  | Date |

Section 4: Even more waveforms

|  |  |  |
| --- | --- | --- |
|  |  | / / |
| Signature |  | Date |