ECE 2713
Homework 6

Spring 2025 Dr. Havlicek

DUE: 04/22/2025, 11:59 PM

What to Turn In:

Submit your solution for this assignment electronically by uploading a file to the course Canvas page.

You can make the file with MS WORD or with any other editor that you prefer. Your turn-in file must be an MS WORD ".docx" file or a PDF file. To create PDF from MS WORD, print the file to PDF.

If you are using the Virtual Labs, make sure to save all your files before you log out!

As you work the problems, you can use the mouse to cut your Matlab code and resulting output from the command window and paste them into your turn-in file. You can also use the Matlab `diary` command to save a session log from the command window to a file like this:

```
diary my.txt
x = [1 2 3];
x(1)
diary off
```

This will save your command window session to a file called `my.txt` in the current directory. You can then open it with WordPad or WORD and paste it into your turn-in file.

For figures and graphs, you can use the File pulldown menu of the Matlab Figure Window to save them as JPEG or BMP files and then insert them into your turn-in file as pictures. To make the color work on the Virtual Labs, I had to open "Export Setup" from the File menu of the Matlab Figure window and **uncheck** the "custom color" box.

Make sure to include your name in your turn-in file and add a title at the top of the first page that says "ECE 2713" and "Homework 6." Number the problems and paste in your Matlab code, the resulting command window output, and the figures and graphs.

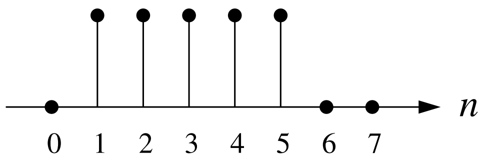
Introduction:

In this assignment, we are going to work with discrete-time signals that have a finite length. To make the mathematics work out right, it is *very important* that our finite-length signals will always start at $n = 0$ and end at $n = N - 1$. So, an 8-point signal $x[n]$ will have length $N = 8$. It will be defined from $n = 0$ to $n = 7$. Outside of this range of n's, the signal is not defined. It's important for you to understand that $x[n]$ is *not* equal to zero for other values of n; it does not even exist for other values of n!

- Remember from Homework 3: the default view in Matlab will have a command window and several panes such as "Current Folder" and "Workspace." I usually get rid of the "Workspace" pane and add a docked "Command History" pane. This can be done by clicking the "Layout" button on the HOME tab. The reason I like to have a "Command History" pane is because you can click previous commands there and drag them to the command window. Then you can execute them again. You can also edit them before you execute them. This can save a lot of typing.

The Assignment:

Here is a simple example of an 8-point signal $x_1[n]$:



We can write the signal $x_1[n]$ like this:

$$x_1[n] = \delta[n-1] + \delta[n-2] + \delta[n-3] + \delta[n-4] + \delta[n-5], \quad 0 \le n \le 7. \tag{1}$$

It's important to write that last part: "$0 \le n \le 7$," because it tells the reader that this signal $x_1[n]$ is *not defined* for other values of n. In other words, it tells the reader that $x_1[n]$ is an 8-point signal that only exists for $0 \le n \le 7$.

In Matlab, you can represent this signal with a vector (array)

```
x1n = [0 1 1 1 1 1 0 0];
```

You must always remember that our time index n starts at $n = 0$, but Matlab arrays start with index 1. The first element of the Matlab array is `x1n(1)`, but in terms of the actual finite-length signal this is $x_1[0]$, which is for $n = 0$. Similarly, the last element of the Matlab array is `x1n(8)`, but this is really $x_1[7]$ corresponding to $n = 7$.

Because the signal $x_1[n]$ is defined only for $0 \le n \le 7$ and *not* for all $n \in \mathbb{Z}$, it does *not* have a discrete-time Fourier transform (DTFT) $X_1(e^{j\omega})$. But it does have an 8-point *discrete Fourier transform* (DFT) $X_1[k]$. Like $x_1[n]$, $X_1[k]$ also has length $N = 8$. It is defined for $0 \le k \le 7$ and it takes complex values. If you want to graph it, you can graph the real part and the imaginary part, or you can graph the magnitude and the angle (phase). Usually, we graph the magnitude and the phase.

The 8-point DFT $X_1[k]$ is given by

$$X_1[k] = \sum_{n=0}^{7} x_1[n] e^{-j2\pi kn/8}, \quad 0 \le k \le 7. \tag{2}$$

It gives us a way to write an 8-point signal like $x_1[n]$ as a sum of the eight DFT basis functions $e^{j0\frac{2\pi}{8}n}$, $e^{j1\frac{2\pi}{8}n}$, $e^{j2\frac{2\pi}{8}n}$, $e^{j3\frac{2\pi}{8}n}$, $e^{j4\frac{2\pi}{8}n}$, $e^{j5\frac{2\pi}{8}n}$, $e^{j6\frac{2\pi}{8}n}$, and $e^{j7\frac{2\pi}{8}n}$. Notice that each one of these basis functions is a complex sinusoid and their frequencies are given by $k(2\pi/8) = k\omega_0$, where $\omega_0 = 2\pi/N$ and $0 \leq k \leq 7$. As always, the transform (2) is the inner product (dot product) between the signal $x_1[n]$ and the basis functions.

To compute the DFT of $x_1[n]$, we have to compute the complex number $X_1[k]$ for each k from 0 to 7. So all together, we have to "do" the equation (2) eight times – once for each k.

For each time that we do the equation (2), the sum requires eight complex multiply-add operations. So the overall computational complexity is 64 complex multiply-add operations. More generally, for an N-point signal the computational complexity is N^2 complex multiply-add operations.

The *fast Fourier Transform* (FFT) is a family of fast algorithms that rearrange the terms of the sum (2) in a tricky way that makes maximum re-use of partial products. This reduces the computational complexity from N^2 complex multiply-add operations to $N\log(N)$ complex multiply-adds. For an 8-point signal, it reduces the complexity from 64 multiply-adds to 40 multiply-adds. For $N = 4096$, it reduces the complexity from about 16.8 million multiply-adds to 49,152 multiply-adds.

Once you have used the DFT equation (2) to compute the eight inner products $X_1[k]$ for $k = 0, 1, 2, \ldots, 7$, then you can write the signal $x_1[n]$ by adding up these inner products times the basis functions. This is called the *inverse discrete Fourier transform* (IDFT). It is given by

$$x_1[n] = \frac{1}{8}\sum_{k=0}^{7} X_1[k]e^{j2\pi kn/8}, \quad 0 \leq n \leq 7. \tag{3}$$

Matlab provides a built-in function `fft` that uses the FFT algorithm to compute the DFT in Eq. (2). Matlab also provides a built-in function `ifft` that uses the FFT algorithm to compute the IDFT (3).

1. Consider the following Matlab code which computes the DFT of the signal $x_1[n]$ in (1) and plots the DFT magnitude and phase as functions of k. The program also plots the DFT magnitude as a function of the Matlab array index and as a function of the radian digital frequency $k(2\pi/8)$. Type in this code and run it. You can type it in line-by-line at the command prompt or you can create an *m*-file.

```
%-----------------------------------------------------
% P1
%
% - Create and plot the signal x_1[n] as a function of n.
% - Compute the DFT X_1[k].  Plot the magnitude and phase
%   as functions of k.
% - Plot the DFT magnitude as a function of the matlab
%   array index.
% - Plot the DFT magnitude as a function of the discrete
%   radian frequency w.
% - Compute and plot the IDFT.
%
n = 0:7;                            % time variable
x1n = [0 1 1 1 1 1 0 0];            % our 8-point signal
X1k = fft(x1n);                     % compute the DFT
X1kmag = abs(X1k);                  % magnitude of the DFT
X1karg = angle(X1k);                % phase of the DFT

% plot the signal
figure(1);
stem(n,x1n);
axis([0 7 0 1.5]);
title('Original Signal');
xlabel('n');
ylabel('x_1[n]');

% plot DFT magnitude and phase as functions of k
k = 0:7;                            % frequency index
figure(2);
stem(k,X1kmag); ylim([0 6]);
title('DFT Magnitude');
xlabel('k');
ylabel('|X_1[k]|');
figure(3);
stem(k,X1karg);
title('DFT Phase');
xlabel('k');
ylabel('arg(X_1[k])');

% plot DFT magnitude as a function of Matlab index
Matlab_idx = [1:8];                 % Matlab index
figure(4);
stem(Matlab_idx,X1kmag); ylim([0 6]);
title('DFT Magnitude');
xlabel('Matlab index');
ylabel('|X_1[index]|');

% plot DFT magnitude as a function of discrete frequency
%  (radians per sample)
w = [0:2*pi/8:7*2*pi/8];            % discrete frequency
```

```
figure(5);
stem(w,X1kmag); ylim([0 6]);
title('DFT Magnitude'); ylim([0 6]);
xlabel('discrete radian frequency \omega');
ylabel('|X_1[\omega]|');

% Compute and plot the IDFT
x2n = ifft(X1k);
figure(6);
stem(n,x2n);
axis([0 7 0 1.5]);
title('IDFT');
xlabel('n');
ylabel('IDFT');
```

People often refer to the eight numbers $X_1[k]$ as "the DFT coefficients" of the signal $x_1[n]$. Here is a table that shows, for each DFT coefficient $X_1[k]$, the Matlab array index, the DFT frequency index k, the digital frequency ω in radians per sample, and the digital frequency f in cycles per sample:

Matlab array index	1	2	3	4	5	6	7	8
DFT freq index k	0	1	2	3	4	5	6	7
ω, rad/sample	$0(2\pi/8)$	$1(2\pi/8)$	$2(2\pi/8)$	$3(2\pi/8)$	$4(2\pi/8)$	$5(2\pi/8)$	$6(2\pi/8)$	$7(2\pi/8)$
f, cycles/sample	0/8	1/8	2/8	3/8	4/8	5/8	6/8	7/8

Now we need to remember two important things about discrete-time complex sinusoids. First, we only need frequencies from $-\pi$ to π to make all the possible graphs. Second, subtracting any integer multiple of 2π from the frequency does not change the graph.

In the table above, notice that the radian frequencies for the DFT coefficients with $k = 5$, 6, and 7 are all $\geq \pi$. Subtracting 2π from the frequency for the $k = 5$ DFT coefficient, we get $5(2\pi/8) - 8(2\pi/8) = -3(2\pi/8)$. This does not change the graph of the basis function at all. So we can think of the $k = 5$ DFT coefficient as being for frequency $+5(2\pi/8)$, or, equivalently, as being for frequency $-3(2\pi/8)$.

Similarly, we can think of the $k = 6$ DFT coefficient as being for frequency $+6(2\pi/8)$ or for frequency $6(2\pi/8)-8(2\pi/8) = -2(2\pi/8)$. And we can think of the $k = 7$ DFT coefficient as being for frequency $+7(2\pi/8)$ or for frequency $-1(2\pi/8)$.

Finally, we can think of the $k = 4$ DFT coefficient as being for frequency $+4(2\pi/8)$ or for frequency $4(2\pi/8) - 8(2\pi/8) = -4(2\pi/8)$. Notice that this is the "$N/2$" coefficient, where $N = 8$ is the length of the signal. People often refer to this DFT coefficient as being for both of the frequencies $\pm 4(2\pi/8)$.

So let's draw the table again, but this time we will subtract 2π from the frequencies of the $k = 4$, 5, 6, and 7 DFT coefficients. It's important for you to remember that this does not change anything. The signals $e^{j5\frac{2\pi}{8}n}$ and $e^{-j3\frac{2\pi}{8}n}$ have exactly the same graph. They are just two different ways of writing the same DFT basis signal.

So here's the new table:

Matlab array index	1	2	3	4	5	6	7	8
DFT freq index k	0	1	2	3	4	5	6	7
ω, rad/sample	$0(2\pi/8)$	$1(2\pi/8)$	$2(2\pi/8)$	$3(2\pi/8)$	$\pm 4(2\pi/8)$	$-3(2\pi/8)$	$-2(2\pi/8)$	$-1(2\pi/8)$
f, cycles/sample	0/8	1/8	2/8	3/8	$\pm 4/8$	$-3/8$	$-2/8$	$-1/8$

From this new table, you can see that the DFT coefficients in the first half of the array are for the positive frequencies, but the coefficients in the second half of the array are actually for the *negative* frequencies.

So, when we look at a DFT array in practice, we usually *swap* the left and right sides of the array so that the negative frequency coefficients are on the left, the zero frequency (DC) coefficient is in the center, and the positive frequency coefficients are on the right.

Here is what the table looks like *after* we swap the left and right halves of the array:

NEW Matlab index	1	2	3	4	5	6	7	8
OLD Matlab index	5	6	7	8	1	2	3	4
DFT freq index k	4	5	6	7	0	1	2	3
ω, rad/sample	$\pm 4(2\pi/8)$	$-3(2\pi/8)$	$-2(2\pi/8)$	$-1(2\pi/8)$	$0(2\pi/8)$	$1(2\pi/8)$	$2(2\pi/8)$	$3(2\pi/8)$
f, cycles/sample	$\pm 4/8$	$-3/8$	$-2/8$	$-1/8$	0/8	1/8	2/8	3/8

This is called the *centered DFT*. Matlab provides a built-in function `fftshift` to center the DFT array for you. Matlab also provides a built-in function `ifftshift` to un-center it. If you center a DFT, then you must **always** un-center it before you try to invert!

So, for example, if you wanted to compute the magnitude and phase of the centered DFT and then invert, you could do it like this:

```
X1kshift = fftshift(fft(x1n));   % compute centered DFT
X1kmag = abs(X1kshift);          % centered spectral magnitude function
X1karg = angle(X1kshift);        % centered spectral phase function
x2n = ifft(ifftshift(X1kshift)); % YOU MUST UN-CENTER before you invert!
```

2. Modify the Matlab code in Problem 1 to compute and plot the magnitude and phase of the centered DFT for the signal $x_1[n]$ in (1). Plot the centered magnitude and phase as functions of the radian frequency ω and of the Hertzian frequency f. Also compute and plot the inverse DFT.

Hint: you may find the following Matlab statements helpful for making the "x-axis" quantities to use in the `stem` commands for radian frequency and Hertzian frequency:

```
w = [-4*2*pi/8:2*pi/8:3*2*pi/8];  % Radian discrete freq

f = [-0.5:1/8:3/8];               % Hertzian discrete freq
```

Hint: here are some more helpful Matlab statements:

```
n = 0:7;                          % time variable
x1n = [0 1 1 1 1 1 0 0];          % our 8-point signal
X1k = fftshift(fft(x1n));         % compute the centered DFT
X1kmag = abs(X1k);                % magnitude of the centered DFT
X1karg = angle(X1k);              % phase of the centered DFT

% plot centered DFT magnitude as a function of radian freq
w = [-4*2*pi/8:2*pi/8:3*2*pi/8];  % radian discrete freq
figure(1);
stem(w,X1kmag); ylim([0 6]);
title('Centered DFT Magnitude');
xlabel('discrete radian frequency \omega');
ylabel('|X_1[\omega]|');
```

3. As you saw in Problems 1 and 2, plotting the DFT magnitude and phase as functions of radian frequency ω is a little bit inconvenient. In Problem 2, we wanted the x-axis of these plots to go from $-\pi$ to $3(2\pi/8)$. But what we got were plots from -4 to 3. Matlab doesn't like it when the first and last ticks on the x-axis are irrational numbers like π.

   Because of this, people often plot the DFT magnitude and phase using a *normalized* radian frequency axis. The normalized frequency is given by ω/π. Then, -1 on the normalized frequency axis corresponds to $\omega = -\pi$ and $+1$ on the normalized frequency axis corresponds to $\omega = +\pi$. This makes the Matlab plots turn out a little bit nicer. Here are some Matlab statements that show you how to do this:

   ```
   w = [-4*2*pi/8:2*pi/8:3*2*pi/8];
   stem(w/pi,X1kmag);
   xlabel('\omega/\pi');
   ```

   Modify your Matlab code from Problem 2 to plot the centered DFT magnitude for $x_1[n]$ as a function of normalized frequency (i.e., using normalized frequency on the horizontal axis).

It's important for you to know about normalized frequency because the IIR digital filter design routines in the Matlab Signal Processing Toolbox require you to specify the passband and stopband edge frequencies in units of normalized frequency.

Now, as we said back on page 2, the finite-length signal $x_1[n]$ in (1) does not have a DTFT $X_1(e^{j\omega})$. But suppose that we make a new signal $\hat{x}_1[n]$ by adding zeros to both sides of $x_1[n]$ so that the new signal is defined for all $n \in \mathbb{Z}$. In other words, we make the new signal

$$\hat{x}_1[n] = \begin{cases} x_1[n], & 0 \le n \le 7, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Then $\hat{x}_1[n]$ *does* have a DTFT $\widehat{X}_1(e^{j\omega})$. The relationship between the 8-point DFT $X_1[k]$ and the DTFT $\widehat{X}_1(e^{j\omega})$ is that $X_1[k]$ is given by eight equally spaced samples of $\widehat{X}_1(e^{j\omega})$ going from $\omega = 0$ to $\omega = 7(2\pi/8)$. The 8-point centered DFT of $x_1[n]$ is given by eight equally spaced samples of $\widehat{X}_1(e^{j\omega})$ going from $\omega = -\pi$ to $\omega = 3(2\pi/8)$.

To investigate this further, let's compute the DTFT $\widehat{X}_1(e^{j\omega})$. Unfortunately, $\hat{x}_1[n]$ is not in our DTFT Table on the course formula sheets. However, the signal

$$\hat{x}_0[n] = \begin{cases} 1, & |n| \le 2, \\ 0, & |n| > 2, \end{cases} \tag{5}$$

is in our DTFT Table. And $\hat{x}_1[n] = \hat{x}_0[n-3]$. In other words, our signal $\hat{x}_1[n]$ in (4) is a time shifted version of $\hat{x}_0[n]$ in (5) and the DTFT of $\hat{x}_0[n]$ is in the table. According to the table,

$$\widehat{X}_0(e^{j\omega}) = \frac{\sin\left(\frac{5}{2}\omega\right)}{\sin(\omega/2)}. \tag{6}$$

Applying the DTFT time shifting property, we get

$$\widehat{X}_1(e^{j\omega}) = e^{-j3\omega}\widehat{X}_0(e^{j\omega}) = \frac{\sin\left(\frac{5}{2}\omega\right)}{\sin(\omega/2)}e^{-j3\omega}. \tag{7}$$

4. Consider the following Matlab code which plots the magnitude and phase of the DTFT $\widehat{X}_1(e^{j\omega})$ together with the magnitude and phase of the centered DFT of $x_1[n]$:

```
%----------------------------------------------------------
% P4a
%
% Show that the DFT of the finite-length signal is given
% by samples of the DTFT of the zero-extended signal.
% - plot the DTFT magnitude of x1hat from -pi to pi.
%   - plot the centered DFT magnitude of x_1[n] on the
%     same graph.
% -plot the DTFT phase of x1hat from -pi to pi.
%   - plot the centered DFT phase of x_1[n] on the same
%     graph.
%

% Frequency vector for plotting the DTFT.  Use 1000 points.
w = linspace(-pi,pi,1000);

% The DTFT was computed analytically (i.e., with paper and
% pencil - not by computer)
X1hat = sin(2.5*w)./sin(w/2) .* exp(-3*j*w);
X1hatmag = abs(X1hat);
X1hatarg = angle(X1hat);

% Now compute the 8-point DFT of the finite-length signal
x1n = [0 1 1 1 1 1 0 0];  % our 8-point signal
k =  -4:3;           % frequency index for the centered DFT
X1k = fftshift(fft(x1n));
X1kmag = abs(X1k);
X1karg = angle(X1k);

figure(1);
plot(w,X1hatmag,'-b');    % plot the DTFT magnitude
axis([-pi pi 0 6]);
hold on;                  % makes the next plot come out on the
                          %   same graph
plot(k*2*pi/8,X1kmag,'ro');  % plot the centered DFT magnitude
hold off;                    %   using a symbol, but no line
                             %   and no stem.
title('Magnitude of DTFT and centered 8-pt DFT');
xlabel('\omega','FontSize',14);
ylabel('$|\widehat X_1(e^{j\omega})|$, $|X_1[\omega]|$',...
       'Interpreter','latex','FontSize',14);
legend('DTFT','DFT');

figure(2);
plot(w,X1hatarg,'-b');    % plot the DTFT phase
axis([-pi pi -4 5]);
hold on;
plot(k*2*pi/8,X1karg,'ro');  % plot the centered DFT phase
hold off;
```

```
title('Phase of DTFT and centered 8-pt DFT');
xlabel('\omega','FontSize',14);
ylabel('$\arg\widehat X_1(e^{j\omega})$, $\arg X_1[\omega]$',...
       'Interpreter','latex','FontSize',14);
legend('DTFT','DFT');
```

(a) Type in this code and run it. You can type it in line-by-line at the command prompt or you can create an *m*-file.

(b) Note that if we add zeros to the right side of $x_1[n]$, then it will make the finite-length signal and the DFT longer. But it will not change $\hat{x}_1[n]$ and $\widehat{X}_1(e^{j\omega})$. If, for example, we change $x_1[n]$ to

```
x1n = [0 1 1 1 1 1 0 0 0 0 0 0];
```

then the length of both $x_1[n]$ and $X_1[k]$ are increased to $N = 12$. But $\hat{x}_1[n]$ is still given by (4) and $\widehat{X}_1(e^{j\omega})$ is still given by (7).

This gives us a way to sample the DTFT $\widehat{X}_1(e^{j\omega})$ with arbitrary density by using zero padding to increase the length of $x_1[n]$.

Modify the Matlab code in Problem 4(a) to plot the magnitude and phase of the DTFT together with the magnitude and phase of the centered DFT of $x_1[n]$ for a length of $N = 16$. To do this, you must change three things in the program P4a. First, you must increase the length of `x1n` to $N = 16$ by appending zeros to the end of the signal. Second, you must change the DFT frequency index to go from $k = -8$ to $k = 7$ instead of $k = -4$ to $k = 3$. Third, in the plot commands for `X1kmag` and `X1karg`, you must change the DFT frequency vector from `k*2*pi/8` to `k*2*pi/16`. You should also change the titles of the plots to reflect the fact that it is now a 16-point DFT.

Okay, that's it for Homework 6. But since you are thinking about the DFT right now, here is some more information that will be *very* helpful in the upcoming Design Project. In the Design Project, you will work with digital audio signals. The length N will be much larger than here in Homework 6. In some cases it will be tens of thousands.

So, assume that N is an even positive integer (like 1024, for example). For an N-point finite-length signal $x[n]$ defined for $0 \le n \le N-1$, the N-point DFT is given by

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N}, \quad 0 \le k \le N-1. \tag{8}$$

If the values of the N-point signal $x[n]$ are stored in the Matlab array `xn`, then the Matlab statement

```
Xk = fft(xn);
```

will place the N complex-valued DFT coefficients in the Matlab array `Xk`. As the Matlab index ranges from 1 to N, the DFT frequency index k ranges from 0 to $N-1$. The Matlab array elements `Xk(1)` through `Xk(N)` contain the DFT coefficients $X[0]$ through $X[N-1]$, which are for radian digital frequencies going from $\omega = 0$ to $\omega = (N-1)\frac{2\pi}{N}$ in steps of $\frac{2\pi}{N}$. If $N = 8$, then everything is exactly as shown in the table on page 5.

However, it's usually more intuitive to work with the centered DFT. The Matlab statement

```
Xk = fftshift(fft(xn));
```

will again place the N complex-valued DFT coefficients in the Matlab array `Xk`. But this time,

- the first half of the Matlab array, i.e., Matlab array elements `Xk(1)` through `Xk(N/2)`, will contain the DFT coefficients $X[N/2]$ through $X[N-1]$ which are for radian digital frequencies going from $\omega = -\pi$ to $\omega = -\frac{2\pi}{N}$ in steps of $\frac{2\pi}{N}$.
- the Matlab array element `Xk(N/2 + 1)` will contain the DFT DC coefficient $X[0]$, which is for radian digital frequency $\omega = 0$.
- the last half of the Matlab array, i.e., Matlab array elements `Xk(N/2 + 2)` through `Xk(N)`, will contain the DFT coefficients $X[1]$ through $X[N/2-1]$ which are for radian digital frequencies going from $\omega = \frac{2\pi}{N}$ to $\omega = \pi - \frac{2\pi}{N}$ in steps of $\frac{2\pi}{N}$.

In other words, in an N-point centered DFT array, the radian digital frequency goes from $-\pi$ to $\pi - \frac{2\pi}{N}$ in steps of $\frac{2\pi}{N}$. If $N = 8$, then everything is exactly as shown in the second table on page 6.

Discrete Hertzian frequency (i.e., cycles per sample instead of radians per sample) is obtained by dividing the radian digital frequency ω by 2π. In an N-point centered DFT array, the Hertzian digital frequency goes from $-\frac{1}{2}$ cycle per sample to $\frac{1}{2} - \frac{1}{N}$ cycles per sample in steps of $\frac{1}{N}$.

Normalized frequency is obtained by dividing the radian digital frequency ω by π. In an N-point centered DFT array, the normalized frequency goes from -1 to $1 - \frac{2}{N}$ in steps of $\frac{2}{N}$.

The N-point inverse DFT (IDFT) is given by

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j2\pi kn/N}, \quad 0 \le n \le N-1. \tag{9}$$

For an un-centered DFT array, the N-point IDFT can be computed using the Matlab statements

```
Xk = fft(xn);
xn = ifft(Xk);
```

The Matlab statements for a centered DFT array are:

```
Xk = fftshift(fft(xn));
xn = ifft(ifftshift(Xk));
```

A Final Note: The DFT is usually written using the special "shorthand" symbol

$$W_N = e^{-j2\pi/N}.$$

For any *fixed* value of N, W_N is a *constant*. Notice that the symbol W_N has the conjugation operation built into it. In terms of W_N, the DFT and IDFT equations (8) and (9) become

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad 0 \leq k \leq N-1, \tag{10}$$

and

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn}, \quad 0 \leq n \leq N-1. \tag{11}$$

Although this is the way that you will usually see the DFT written, we are not using the "W_N notation" in this assignment in the interest of simplicity.