Project Motivation:

In our first project we estimated the amplitude, frequency, and phase of a discrete time sinusoid using the DTFT and it's discrete counterpart the DFT.

Our investigation revealed that the DTFT magnitude of a *complex* sinusoid always has a peak at the desired frequency. The amplitude and phase of the sinusoid can then be determined by analyzing the amplitude and angle the DTFT at that point. As the DFT, implemented in Matlab using the FFT, is simply a sampled version of the DTFT, it will correctly determine the sinusoid's parameters only when the DTFT is sampled at its peak. We found that appending zeros to the sinusoidal data simply results in a DFT with more points, increasing the likelihood that a point in the DFT is in close proximity to the desired peak. (See Figure 1.)

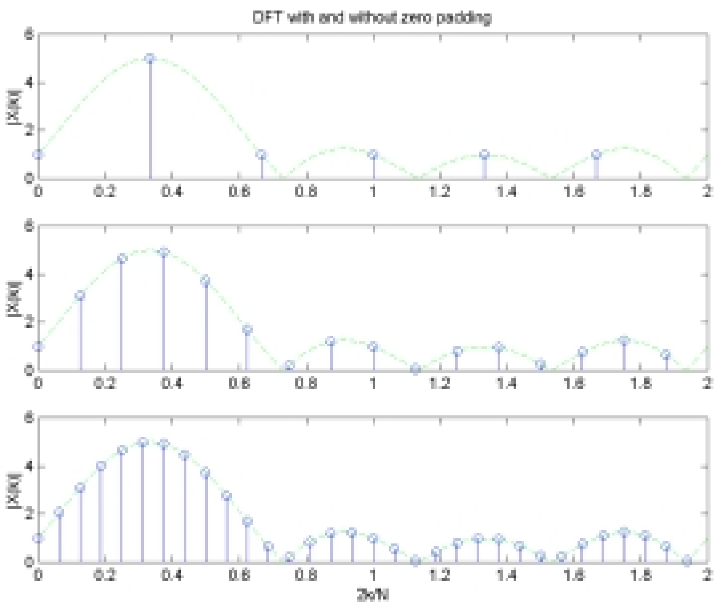

Figure 1: *The stems in the above figure correspond to the points in the DFT magnitude of a sinusoid with radian frequency π/3 while the dotted line is the DTFT magnitude. The top plot is the DFT without zero padding. The middle plot is the DFT of the sinusoid padded with zeros to length 16, while the bottom plot is the DTF of the data padded to length 32.*

We encountered some problems when using purely real sinusoids. Using Euler's formula, it was shown that the DTFT of a real sinusoid has two components corresponding to each of the complex parts of the sinusoid. These components add together causing interference and cancellation in the DTFT. As a result, the peak of the DTFT magnitude is often located at a location other than frequency of sinusoid. The estimation of the amplitude and phase also suffer. (See Figure 2.)

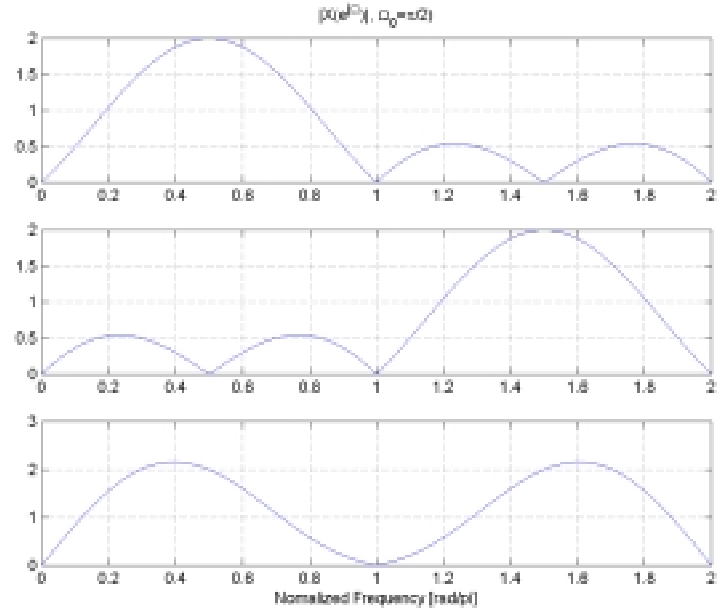

Figure 2: *The top plot is the DTFT magnitude of $x_1[n]$, where $x_1[n] = Ae^{j\Omega 0n+\phi}$. The middle plot is DTFT magnitude of $x_2[n] = Ae^{-j\Omega 0n+\phi}$. The bottom plot is DTFT magnitude of $x[n] = (x_1[n]+ x_2[n])/2 = A\cos(\Omega_0 n+\phi)$.*

Next, we investigated some possible ways to mitigate the problems associated with frequency estimation of a real sinusoid using the DFT. One of the more promising alternatives was to use a greater number of data samples in addition to zero padding. Using a longer data vector results in a DFT with less interference between the two components of the transform. Problems still exist, however, especially when the sinusoidal frequency is close to 0 or π.

A New Model:

The problem with the DFT is not that we aren't using it correctly, but that it isn't the right model for this particular problem. As was shown, the DTF is a good tool for estimating the parameters of a complex sinusoid, but it fails when applied to real sinusoidal signals. A new model is derived as follows:

Using the trigonometric identity $\cos(A+B) = \cos(A)\cos(B) - \sin(A)\sin(B)$, the real sinusoid $d[n] = A\cdot\cos(\Omega_0 n + \theta)$ can be rewritten as

$$A\cdot\cos(\theta)\cos(\Omega_0 n) - A\cdot\sin(\theta)\sin(\Omega_0 n) = a\cdot\cos(\Omega_0 n) + b\cdot\sin(\Omega_0 n)$$

where

$$a = A\cdot\cos(\theta)$$
$$b = -A\cdot\sin(\theta)$$

We define the following signals:

Data signal:	$d[n] = A \cdot \cos(\Omega_0 n + \theta)$
Sinusoidal Model:	$x[n] = a \cdot \cos(\Omega_1 n) + b \cdot \sin(\Omega_1 n)$
Squared Error:	$E(a,b) = \sum_n (d[n] - x[n])^2$

In order to fit the sinusoid model to the data, we want to minimize the square error with respect to *a* and *b*. Note that the sinusoidal model has frequency Ω_1 while the sinusoidal data has frequency Ω_0. Because we do not know Ω_0 *a priori*, Ω_1 is varied from 0 to π, and a new squared error determined at each step.

$$\frac{\partial E(a,b)}{\partial a} = 0 \qquad \frac{\partial E(a,b)}{\partial b} = 0$$

$$\frac{\partial E(a,b)}{\partial a} = \frac{\partial}{\partial a} \sum_{n=0}^{N-1} \left[d[n] - a \cdot \cos(\Omega_1 n) - b \cdot \sin(\Omega_1 n) \right]^2 = 0$$

$$\frac{\partial E(a,b)}{\partial a} = 2 \sum_{n-0}^{N-1} \left[d[n] - a \cdot \cos(\Omega_1 n) - b \cdot \sin(\Omega_1 n) \right] \cdot \left(-\cos(\Omega_1 n) \right) = 0$$

$$\sum_{n=0}^{N-1} a \cdot \cos(\Omega_1 n) \cdot \cos(\Omega_1 n) + \sum_{n=0}^{N-1} b \cdot \sin(\Omega_1 n) \cdot \cos(\Omega_1 n) = \sum_{n=0}^{N-1} d[n] \cdot \cos(\Omega_1 n)$$

$$\underline{d} = d[n] \qquad \underline{c} = \cos(\Omega_1 n) \qquad \underline{s} = \sin(\Omega_1 n)$$

$$a \cdot \underline{c}^T \underline{c} + b \cdot \underline{c}^T \underline{s} = \underline{c}^T \underline{d}$$

And similarly for $\frac{\partial E(a,b)}{\partial b} = 0$, we can derive the following equation:

$$a \cdot \underline{c}^T \underline{s} + b \cdot \underline{s}^T \underline{s} = \underline{s}^T \underline{d}$$

We now have two equations with two unknowns.

$$\begin{bmatrix} \underline{c}^T \underline{c} & \underline{c}^T \underline{s} \\ \underline{c}^T \underline{s} & \underline{s}^T \underline{s} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \underline{c}^T \underline{d} \\ \underline{s}^T \underline{d} \end{bmatrix}$$

Solving for *a* and *b*,

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \underline{c}^T \underline{c} & \underline{c}^T \underline{s} \\ \underline{c}^T \underline{s} & \underline{s}^T \underline{s} \end{bmatrix}^{-1} \begin{bmatrix} \underline{c}^T \underline{d} \\ \underline{s}^T \underline{d} \end{bmatrix}$$

The values of *a* and *b*, and the resulting squared error, are computed for many values of Ω_1. The frequency at which the minimum error occurs is the best estimate of the frequency of the sinusoidal data, Ω_0. The amplitude and phase are computed using the following formulas:

$$A = \sqrt{a^2 + b^2}$$
$$\theta = \angle(a + jb)$$

Comparing the Results

Matlab was used to implement both the DFT and linear least squares (LLS) methods of sinusoidal parameter estimation. To make the comparisons fair, both the DFT and LLS methods used the same number of discrete frequency points between 0 and π, L = 512. The number of data points, N, was also varied. A sequence of test cases is shown on the following pages.

Example 1: We begin by trying to estimate the parameters of a sinusoid with A = 1, $\Omega_0 = \pi/2$, and $\theta = 0$. N = 8 data points are used.

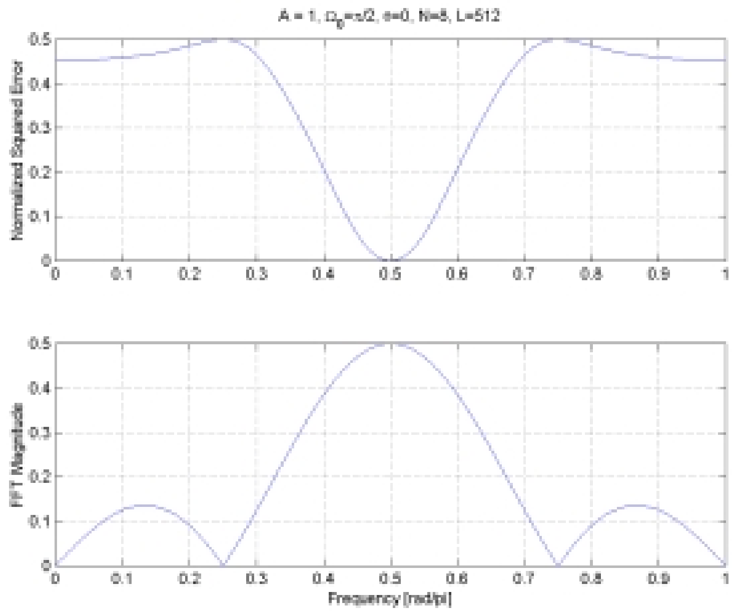

Figure 3: *The top plot shows the squared error as a function of frequency. The lower plot shows the DFT magnitude. Each method uses 512 points. The actual sinusoidal parameters are A = 1,* $\Omega_0 = \pi/2$*, and* $\theta = 0$*. N = 8 data points were used.*

Linear Least Squares Method			
	A	$\mathbf{\Omega_0}$	$\boldsymbol{\theta}$
Actual Value	1.0000	1.5708	0.0000
Estimated Value	1.0000	1.5708	0.0000
a = 1.0000		**b** = 0.0000	

DFT Method			
	A	$\mathbf{\Omega_0}$	$\boldsymbol{\theta}$
Actual Value	1.0000	1.5708	0.0000
Estimated Value	1.0000	1.5708	0.0000

Table 1: *Both The Linear Least Squares and the DFT methods are able to correctly identify the sinusoidal parameters in this instance.*

Example 2: This example is the same as the last, except a phase of $\pi/4$ is added to the sinusoid. Notice that the Linear Least Squares method correctly estimates the sinusoidal parameters, while the DFT method fails.

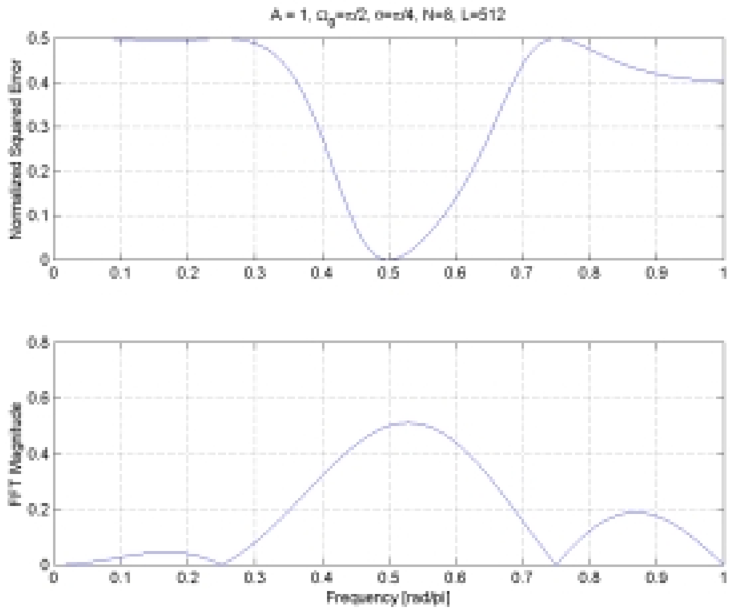

Figure 4: *The top plot shows the squared error as a function of frequency. The lower plot shows the DFT magnitude. Each method uses 512 points. The actual sinusoidal parameters are* $A = 1$, $\Omega_0 = \pi/2$, *and* $\theta = \pi/4$. $N = 8$ *data points were used.*

Linear Least Squares Method			
	A	Ω_0	θ
Actual Value	1.0000	1.5708	0.7854
Estimated Value	1.0000	1.5708	0.7854
a = 0.7071		**b** = -0.7071	

DFT Method			
	A	Ω_0	θ
Actual Value	1.0000	1.5708	0.7854
Estimated Value	1.0229	1.6628	0.4633

Table 2: *The Linear Least Squares method is able to correctly identify the sinusoidal parameters, but the DFT method fails.*

Example 3: This example is the same as the last, except the number of data points was increased from N = 8 to N = 30. Notice that the Linear Least Squares method still correctly estimates the sinusoidal parameters, while the performance of DFT improves.

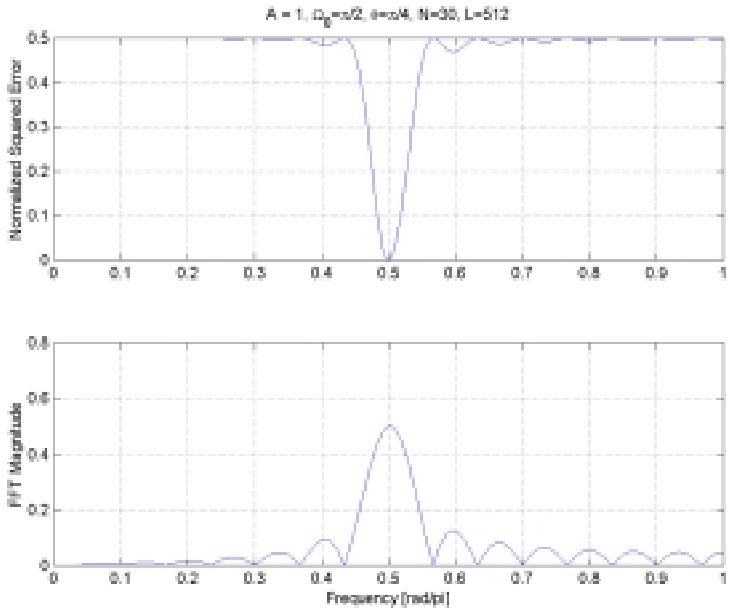

Figure 5: *The top plot shows the squared error as a function of frequency. The lower plot shows the DFT magnitude. Each method uses 512 points. The actual sinusoidal parameters are A = 1,* $\Omega_0 = \pi/2$*, and* $\theta = \pi/4$*. The number of data points was increased from N = 8 to N = 30.*

Linear Least Squares Method			
	A	Ω_0	θ
Actual Value	1.0000	1.5708	0.7854
Estimated Value	1.0000	1.5708	0.7854
a = 0.7071		**b** = -0.7071	

DFT Method			
	A	Ω_0	θ
Actual Value	1.0000	1.5708	0.7854
Estimated Value	1.0017	1.5769	0.6964

Table 3: *The Linear Least Squares method once again is able to correctly identify the sinusoidal parameters. The DFT method still fails, but is much more accurate due to the increase in the number of data points from 8 to 30.*

Example 4: The goal of this example is to show that the Linear Least Squares method will correctly estimate all the parameters of a sinusoidal as long as frequency of the sinusoid falls exactly on one of the bins (i.e. $\Omega_0 = \Omega_1$ for some discrete value of Ω_1).

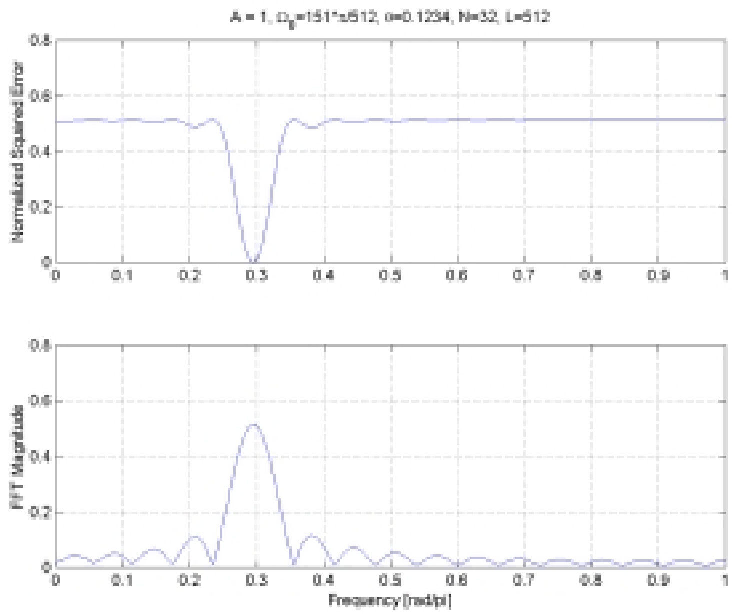

Figure 6: *The top plot shows the squared error as a function of frequency. The lower plot shows the DFT magnitude. Each method uses 512 points. The actual sinusoidal parameters are A = 1, $\Omega_0 = 151*\pi/512$, and $\theta = 0.1234$. N = 32 data points were used.*

Linear Least Squares Method			
	A	Ω_0	θ
Actual Value	1.0000	0.9265	0.1234
Estimated Value	1.0000	0.9265	0.1234
a = 0.7071		**b** = -0.7071	

DFT Method			
	A	Ω_0	θ
Actual Value	1.0000	0.9265	0.1234
Estimated Value	1.0297	0.9265	0.0996

Table 4: *The Linear Least Squares method correctly identifies the sinusoidal parameters. The DFT method correctly estimates Ω_0.*

Matlab Code for Comparing DFT and LLS:

```
% Project 3 vs Project 1
% Compare sinusoid estimation using the the FFT with the Linear
% Least Squares method

clear
N = 30;                    % N is the # of samples in the data
n = [0:N-1]';              % n is the discrete time index

A = 1;                     % Amplitude
Omega0 = pi/2;             % Frequency
Theta = pi/4;              % Angle

d = A*cos(Omega0*n+Theta); % Data signal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Using L-1 values of Omega between 0 and pi (not inclusive), compute the
% values of a and b that minimize the squared error.  Store the error and
% the values of a and b for each value of Omega.

L = 512;

for k = 1:L-1;

   Omega1 = pi*k/L;

   c = cos(Omega1*n);
   s = sin(Omega1*n);

   M = [c'*c c'*s;
        c'*s s'*s];

   r(1,1) = c'*d;
   r(2,1) = s'*d;

   ab = inv(M)*r;
   ab_stored(:,k) = ab;

   x = [c s]*ab;
   e(k) = sum((d-x).^2);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Find the frequency at which the minimum error occurs.  Compute estimates
% for Omega0, A, and Theta.

[dummy,I] = min(e);     % Get index of minimum error
abEst = ab_stored(:,I); % Find the ab associated with the min error

OmegaEst = pi*I/L;                      % Find an estimate for Omega0
AEst = sqrt(abEst'*abEst);              % Find an estimate for A
ThetaEst = angle(abEst(1)-j*abEst(2)); % Find an estimate for Theta

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Make a table with the vital Linear Least Squares parameters

fprintf('\n');
fprintf('            Linear Least Squares Method           \n');
fprintf('------------------------------------------------- \n');
fprintf('|                  |    A     | Omega0   |  Theta   | \n');
fprintf('|------------------|----------|----------|----------| \n');
fprintf('| Actual Value     | %5.4f   | %5.4f   | %5.4f   | \n',A,Omega0,Theta);
fprintf('| Estimated Value | %5.4f   | %5.4f   | %5.4f   |
\n',AEst,OmegaEst,ThetaEst);
fprintf('------------------------------------------------- \n');
fprintf('         a = %5.4f       b = %5.4f          \n',abEst(1),abEst(2));
fprintf('------------------------------------------------- \n');
fprintf('\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Use the FFT to estimate Omega0, A, and Theta. Use L+1 values of Omega
between 0 and pi (inclusive).

D = fft(d,L*2)/N;
D = D(1:L+1);
DMag = abs(D);
[dummy,I] = max(DMag);      % Get index of minimum error
OmegaEst = pi*(I-1)/L;      % Find an estimate for Omega0
AEst = 2*DMag(I);           % Find an estimate for A
ThetaEst = angle(D(I));     % Find an estimate for Theta

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Make a table with the vital FFT parameters

fprintf('\n');
fprintf('                     FFT Method                   \n');
fprintf('------------------------------------------------- \n');
fprintf('|                  |    A     | Omega0   |  Theta   | \n');
fprintf('|------------------|----------|----------|----------| \n');
fprintf('| Actual Value     | %5.4f   | %5.4f   | %5.4f   | \n',A,Omega0,Theta);
fprintf('| Estimated Value | %5.4f   | %5.4f   | %5.4f   |
\n',AEst,OmegaEst,ThetaEst);
fprintf('------------------------------------------------- \n');
fprintf('\n');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the error signal and the FFT magnitude.

OmegaLLS = [1:L-1]/L;
OmegaFFT = [0:L]/L;

subplot(211),plot(OmegaLLS,e/N)
title('A = 1, \Omega_0=\pi/2, \theta=\pi/4, N=30, L=512'),grid
ylabel('Normalized Squared Error')

subplot(212)
plot(OmegaFFT,DMag),grid
ylabel('FFT Magnitude'),xlabel('Frequency [rad/pi]')
```