Project II: Spectral Estimation

6.341 Discrete-Time Signal Processing

Jason Chang
Fall 2007

Part A. Writing and Using Spectral Estimation Functions

In this part of the project, we are interested in estimating the power spectral density (PSD). In real situations, we do not have access to the real PSD when we are trying to estimate it. However, in this project, we are given the exact PSD to compare our estimations with.

Theoretically, a good way to calculate the error between the estimated PSD, $\hat{P}\left(e^{j\omega}\right)$, and the actual PSD, $P\left(e^{j\omega}\right)$, is to use the mean squared error (MSE).

$$MSE = \int_0^{\pi} \left| P\left(e^{jk\pi/512}\right) - \hat{P}\left(e^{jk\pi/512}\right) \right|^2 d\omega$$

However, since we are using Matlab to simulate the estimated PSDs, we will be using discrete approximations using the DFT. Therefore, we will approximate the MSE with the following expression:

$$J = \frac{1}{512} \sum_{k=0}^{512} \left| P\left(e^{jk\pi/512}\right) - \hat{P}\left(e^{jk\pi/512}\right) \right|^2$$

The following code was used to estimate the errors between actual and approximated PSDs.

```
function [ error ] = calcerrorJC( actual, estimate )
%CALCERROR Calculates a discretized approximation to the MSE between actual
%   and estimate PSD based on the first 512 samples.  Assumes input is at
%   least 512 samples.  This should be used with 1024-PT DFT of real
%   signals.

error = sum((abs(actual(1:512)-estimate(1:512))).^2) / 512;
```

A.1. Periodogram-based Techniques

A.1.A – The Periodogram

The periodogram is calculated using the following steps:

1) Multiple the input, $x[n]$, with the window, $w[n]$, to get the windowed input, $v[n]$

$$v[n] = x[n]w[n]$$

2) Take the N-PT DFT of the windowed input to get $V[k]$

$$V[k] = \sum_{k=0}^{N-1} v[n] e^{j\frac{2\pi kn}{N}}$$

3) Calculate the values L, the window length, and U, the energy in the window

$$U = \frac{1}{L}\sum_{n=0}^{L}\left(w[n]\right)^2$$

4) Find the periodogram from the calculated values

$$P = \frac{1}{LU}\left|V[k]\right|^2$$

This method is rather straightforward to implement in Matlab. The only notes that need to be said are that the input and the window need to be of the same length to do a point-wise multiplication. Zeropadding or truncation of the window will be used to extend or curtail the window according to the length of the input. Also, we will use the Fast Fourier Transform (FFT) algorithm to implement the DFT.

```
function [ P ] = periodogramJC( x, window, NFFT )
%PERIODOGRAMJC Computes the periodogram of x[n] using the window, and
%   NFFT-PT DFT.  Note: window does not need to be rectangular.
%
%   [P] = periodogramJC(x, window, NFFT) computes the periodogram with a
%   specific window and using a specific point DFT.

% Make sure x[n] and w[n] are of the same length
if (length(x) > length(window))
    % zeropad the window
    window(length(window)+1:length(x)) = 0;
elseif (length(window) > length(x))
    % Assume x[n] is zero outside of the given x, and truncate the window
    window(length(x)+1:length(window)) = [];
end

% Find the windowed value v[n] = x[n]*w[n]
v = x.*window;

% Find the N-PT DFT of the windowed x[n]
V = fft(v, NFFT);

% Find the normalization factor U
U = 1/sum(window~=0) * sum(window.^2);

% Periodogram is just 1/UL * |V|^2
P = abs(V).^2 / (U*sum(window~=0));
```

It should be noted here that the input signals are of length 2048. However, the actual PSD that we are comparing our approximated PSDs to was found with a 1024-PT DFT. We have a few choices here with what we can do with the input signal to accurately compare it with the actual PSD calculated with a 1024-PT DFT.

1) Truncate the input signal to only 1024 points, and take the 1024-PT DFT

2) Time alias the input sequence to 1024 points, and take the 1024-PT DFT

3) Take the 2048-PT DFT, and drop every other sample

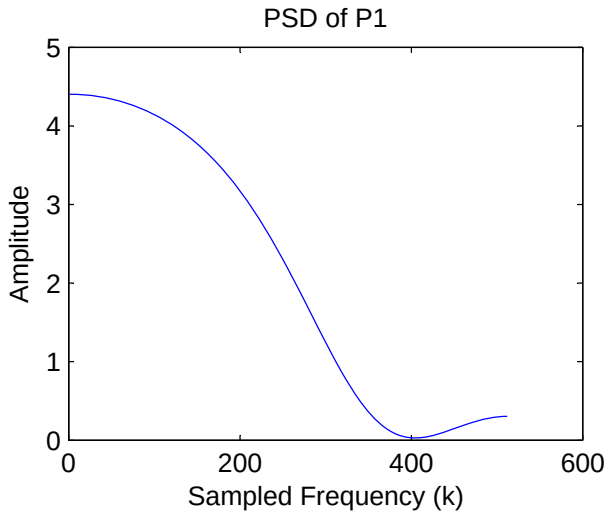
The 1st choice above is bad because we are losing all the data contained within the second half of the signal. All spectral power information in those samples will be lost, and in most cases, we will not get a good estimate of the entire PSD of the input signal.

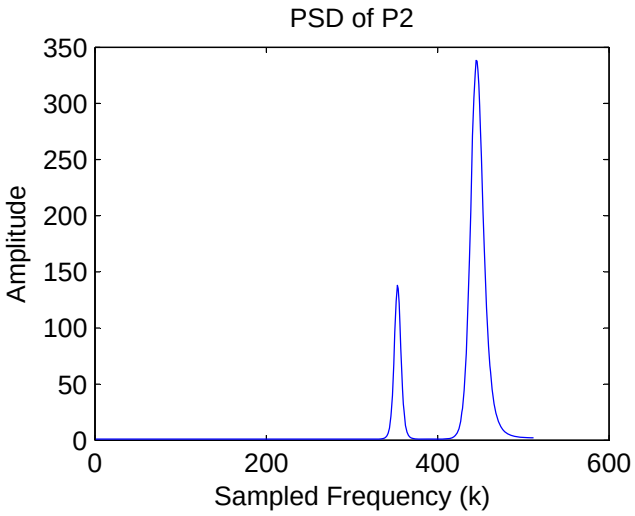
The 2nd choice and the 3rd choice should actually produce similar, if not the same results. Aliasing and taking the 1024-PT DFT is equivalent to just sampling the DTFT with 1024 points on the interval $[0,\pi]$. This is exactly the same as sampling the DTFT with 2048 points, but only looking at the odd samples. Therefore, for the lack of a better choice, we will use method 3 in calculating periodograms. We will also use this reasoning later on for the autoregressive spectrum estimates.

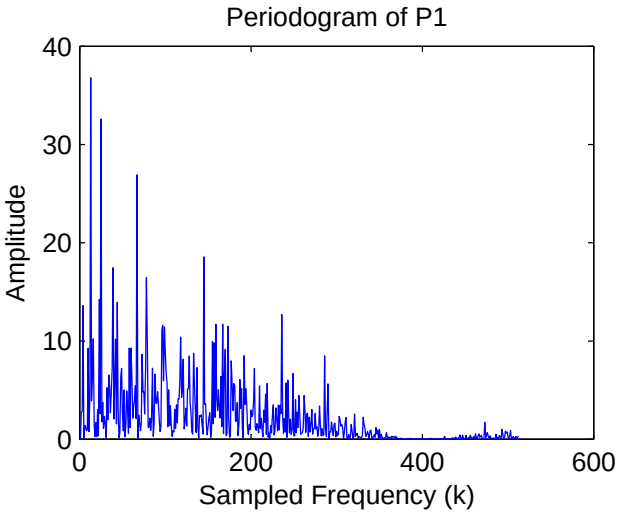
The following code was used to estimate the PSD with a periodogram:

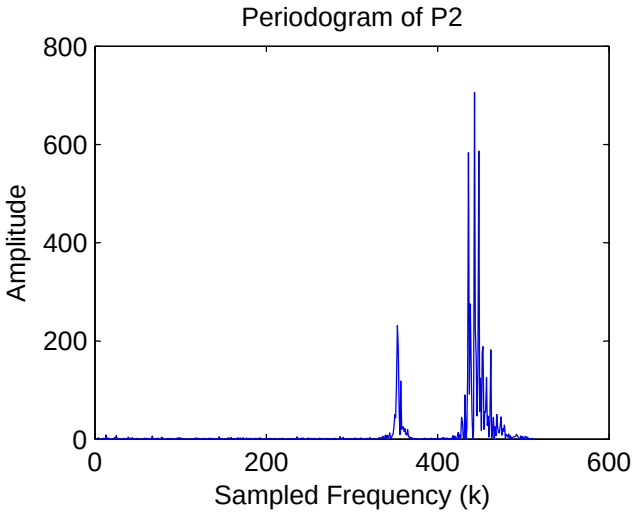
```
Pgram2048 = periodogramJC(v1, ones(1,2048), 2048);
Pgram1 = Pgram2048(1:2:2048);

Pgram2048 = periodogramJC(v2, ones(1,2048), 2048);
Pgram2 = Pgram2048(1:2:2048);

errors(1) = calcerrorJC(P1, Pgram1);
errors(2) = calcerrorJC(P2, Pgram2);
```

The following page contains the PSD estimates using a periodogram.


PSD of P1
Amplitude
Sampled Frequency (k)
0
1
2
3
4
5
0
200
400
600


PSD of P2
Amplitude
Sampled Frequency (k)
0
50
100
150
200
250
300
350
0
200
400
600


Periodogram of P1
Amplitude
Sampled Frequency (k)
0
10
20
30
40
0
200
400
600


Periodogram of P2
Amplitude
Sampled Frequency (k)
0
200
400
600
800
0
200
400
600

A.1.B – Welch's Method

Even though the project description puts Bartlett's Method before Welch's Method, the discussion here will first focus on the formulation of Welch's Method. The reason for this will become clear in the next section.

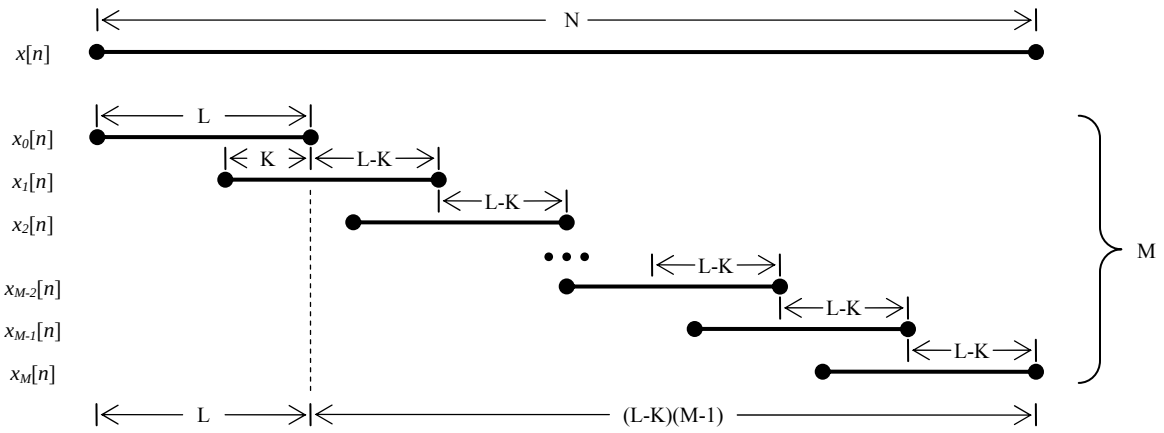
Welch's Method for the modified periodogram is calculated with the following steps:

1) Segment the input, $x[n]$, into a specified number of segments, each overlapping by a specified number of samples

2) Take the periodogram of each segment of the input

3) Average all the periodograms to obtain the modified periodogram of Welch's Method

This seems rather easy seeing how we have already written the periodogram function. However, one issue that needs to be discussed arises from the project description. In the handout, it states that we need to calculate Welch's Method using a certain number of segments, and a certain percentage of overlap. We need to find the actual starting and stopping indices for each segment of the input for our periodogram function, and it is not obviously clear how to find these numbers from the given parameters. If instead we were given the length of the segment, and the number of overlapping samples, picking out each segment would be easy. Let's formulate how to find these parameters from the given ones.



Clearly, we have the following relationship (where N is the length of the input, L is the length of a segment, K is the number of overlapping samples, and M is the number of segments):

$$N = L + (L - K)(M - 1)$$

In the specific case of this project, the number of overlapping samples is given as a percentage of the total segment. In other words, we have (where p is the percentage/100):

$$K = pL$$

$$N = L + (L - pL)(M - 1) = L + L(1 - p)(M - 1) = L\left[1 + (1 - p)(M - 1)\right]$$

$$L = \frac{N}{1+(1-p)(M-1)}$$

Notice that for a set of parameters, N, p, and M, there is no guarantee that L, the length of each segment, is an integer. Knowing this, we will use the approximation to truncate the input signal so that the length of each segment is an integer, and the number of overlapping samples is an integer. In other words, we have the following equations for calculating the parameters:

$$L = \left\lfloor \frac{N}{1+(1-p)(M-1)} \right\rfloor \qquad K = \lceil pL \rceil$$

It should be noted that this approximation may not be the best one to make because we are truncating data. However, the extra information we get from those few truncated samples is almost negligible. For a long input sample of 2048, truncating 8 samples (which is the maximum number of samples lost using this approximation for the parameters given) loses only 0.4% of the data. However, using this approximation greatly simplifies the code because we can keep the number of overlapping samples and the segment lengths the same.

Using these equations, the following code was used to calculate the new parameters for Welch's Method:

```
function [ lengthWindow, nOverlap ] = ...
    calcpwelchJC( lengthX, nSegments, percentOverlap )
%CALCPWELCHJC Calculates the parameters to use for pwelchJC with the given
%   specifications.
%
%   [lengthWindow, nOverlap] = ...
%      calcpwelchJC( lengthX, nSegments, percentOverlap)
%   lengthX is the total length of your input sequence
%   nSegments is the number of desired segments
%   percentOverlap is the approximated desired percentage overlap between
%      segments
%   lengthWindow is the suggested window size
%   nOverlap is the suggested number of overlapping samples

% Refer to the report for the derivations to these equations

% First find the ideal, non-integer values
L = lengthX / (1+(1-percentOverlap)*(nSegments-1));
K = percentOverlap * L;

% Round the number of overlapping samples up and the window length down
lengthWindow = floor(L);
nOverlap = ceil(K);
```

Now that these parameters are found, it's simple to segment the input signal and run Welch's Method on it. The following code was used to implement Welch's Method:

```
function [ P ] = pwelchJC( x, window, nSegments, nOverlap )
%PWELCHJC Uses Welch's method to calculate the modified periodogram.
%
%   [P] = pwelchJC(x, window, nSegments, nOverlap) computes the modified
%   periodogram using Welch's Method.  window describes the window to use
%   on x when calculating the segmented periodogram.  nSegments is the
%   number of segments to divide the input signal, x, into.  nOverlap is
%   the number of overlapping samples.
%
```

```
%   It should be noted that all the samples of x will most likely not be
%   used (unless the parameters are very specially chosen).  In most cases,
%   x will be truncated.  The upper bound to how many samples of x are
%   truncated is just (length(window) - nOverlap)
%
%   It is suggested to use calcpwelchJC.m to calculate parameters

L = length(window);

P = zeros(1, 1024);

for i=1:nSegments
    % Find the starting and ending indices
    xstart = (L - nOverlap)*(i-1) + 1;
    xend = L + (L - nOverlap)*(i-1);

    % Find the periodogram with 1024 PT DFT and add it to the total
    P = P + periodogramJC(x(xstart:xend), window, 1024);
end

% Find the average of all the periodograms
P = P ./ nSegments;
```

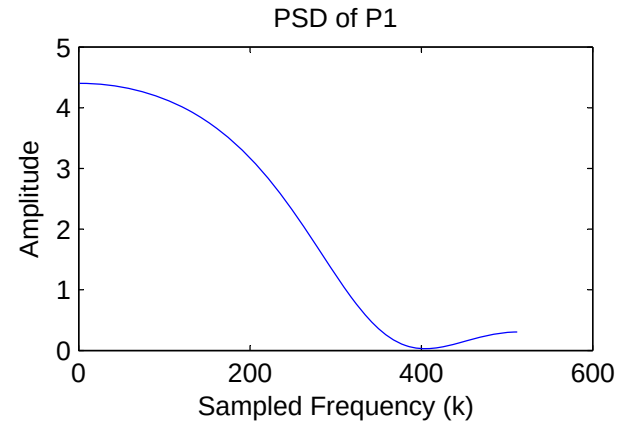
The following code was used to estimate the PSD with Welch's Method:

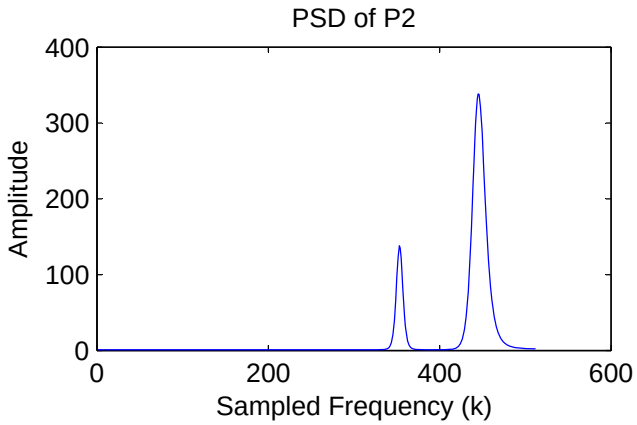
```
[lengthWindow, nOverlap] = calcpwelchJC(length(v1), 4, 0.125);
welch1125 = pwelchJC(v1, ones(1,lengthWindow), 4, nOverlap);
[lengthWindow, nOverlap] = calcpwelchJC(length(v1), 4, 0.25);
welch125 = pwelchJC(v1, ones(1,lengthWindow), 4, nOverlap);
[lengthWindow, nOverlap] = calcpwelchJC(length(v1), 4, 0.5);
welch150 = pwelchJC(v1, ones(1,lengthWindow), 4, nOverlap);

[lengthWindow, nOverlap] = calcpwelchJC(length(v2), 4, 0.125);
welch2125 = pwelchJC(v2, ones(1,lengthWindow), 4, nOverlap);
[lengthWindow, nOverlap] = calcpwelchJC(length(v2), 4, 0.25);
welch225 = pwelchJC(v2, ones(1,lengthWindow), 4, nOverlap);
[lengthWindow, nOverlap] = calcpwelchJC(length(v2), 4, 0.5);
welch250 = pwelchJC(v2, ones(1,lengthWindow), 4, nOverlap);

errors(1) = calcerrorJC(P1, welch1125);
errors(2) = calcerrorJC(P1, welch125);
errors(3) = calcerrorJC(P1, welch150);
errors(4) = calcerrorJC(P2, welch2125);
errors(5) = calcerrorJC(P2, welch225);
errors(6) = calcerrorJC(P2, welch250);
```

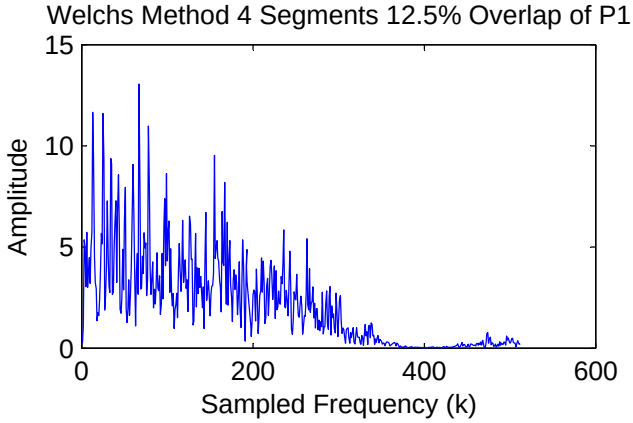
With the calculations, I used the following segment lengths and number of overlapping samples:

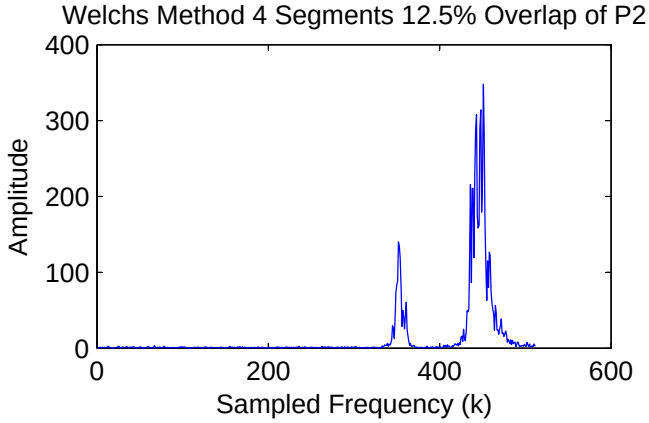
Welch's Method	Segment Length	# of Overlapping Samples	# of Truncated Samples
4 Segment 12.5% Overlap	564	71	8
4 Segment 25% Overlap	630	158	2
4 Segment 50% Overlap	819	410	2

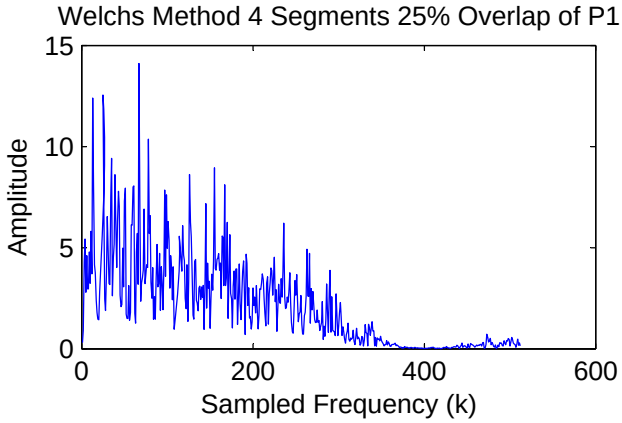
The following page contains the PSD estimates using Welch's Method.

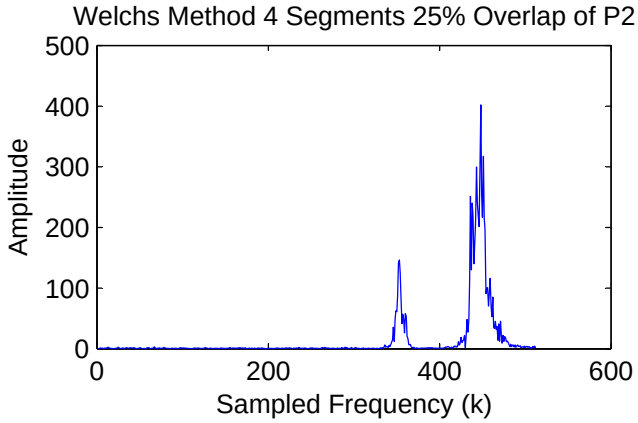

PSD of P1
Amplitude
Sampled Frequency (k)
0
1
2
3
4
5
0
200
400
600

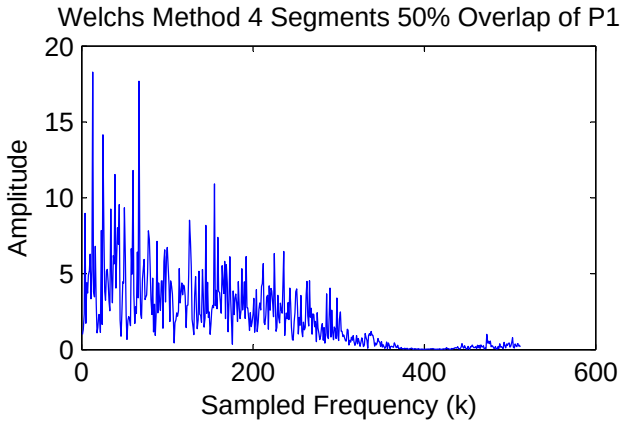

PSD of P2
Amplitude
Sampled Frequency (k)
0
100
200
300
400
0
200
400
600

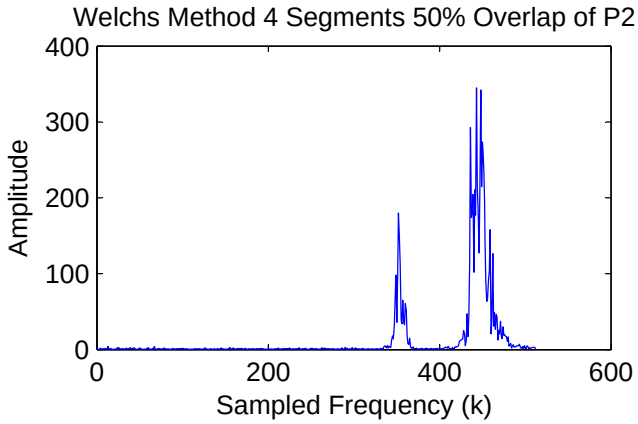

Welchs Method 4 Segments 12.5% Overlap of P1
Amplitude
Sampled Frequency (k)
0
5
10
15
0
200
400
600


Welchs Method 4 Segments 12.5% Overlap of P2
Amplitude
Sampled Frequency (k)
0
100
200
300
400
0
200
400
600


Welchs Method 4 Segments 25% Overlap of P1
Amplitude
Sampled Frequency (k)
0
5
10
15
0
200
400
600


Welchs Method 4 Segments 25% Overlap of P2
Amplitude
Sampled Frequency (k)
0
100
200
300
400
500
0
200
400
600


Welchs Method 4 Segments 50% Overlap of P1
Amplitude
Sampled Frequency (k)
0
5
10
15
20
0
200
400
600


Welchs Method 4 Segments 50% Overlap of P2
Amplitude
Sampled Frequency (k)
0
100
200
300
400
0
200
400
600

A.1.C – Bartlett's Method

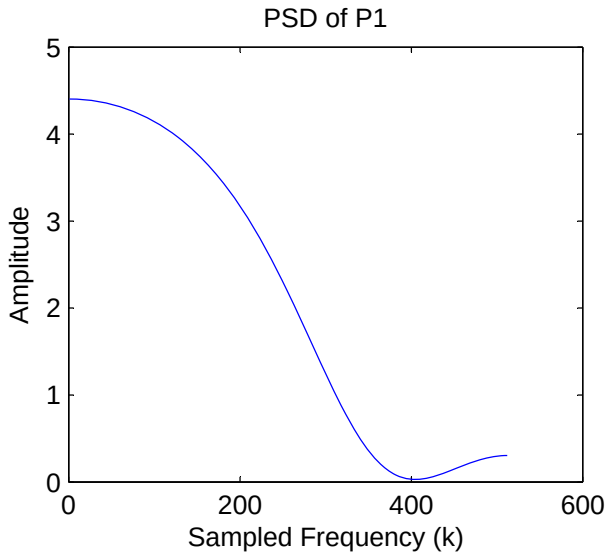
Bartlett's Method is averaging non-overlapping windowed samples of the input signal. This modified periodogram is just a special case of Welch's Method. Thus, we can use the previously written functions to easily implement Bartlett's Method.

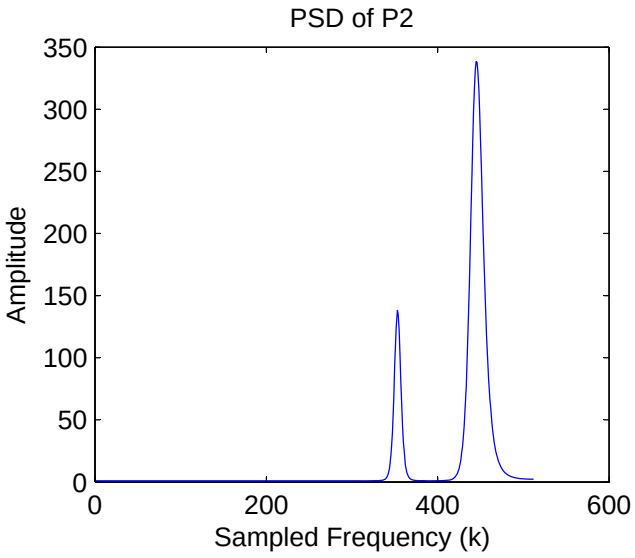
```
function [ P ] = bartlettJC( x, nSegments )
%BARTLETTJC Uses Bartlett's method to calculate the modified periodogram.
%
%   [P] = bartlettJC(x, nSegments) computes the modified periodogram using
%   Bartlett's Method.  nSegments is the number of segments to divide the
%   input signal, x, into.

[lengthWindow, nOverlap] = calcpwelchJC(length(x), nSegments, 0);
P = pwelchJC(x, ones(1,lengthWindow), nSegments, nOverlap);
```

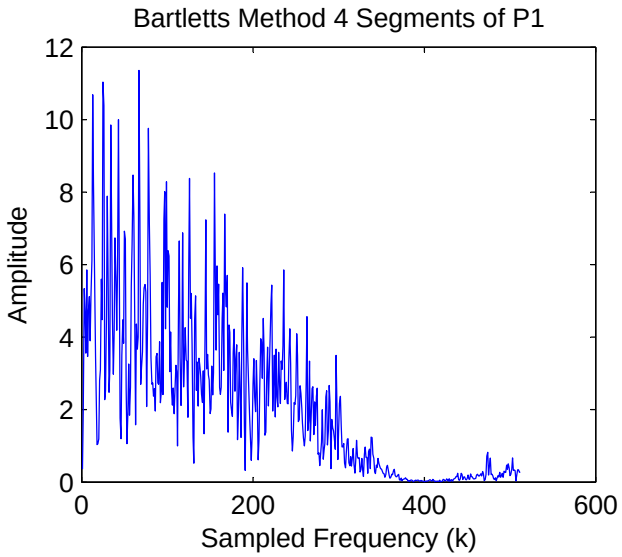
The following code was used to estimate the PSD with Bartlett's Method:

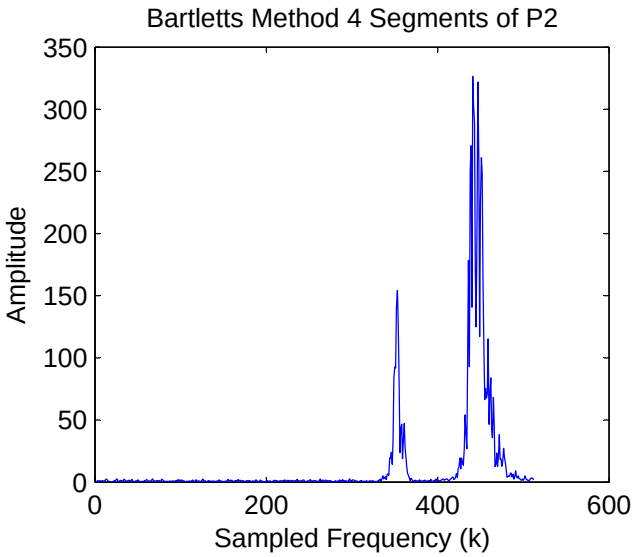
```
bartlett14 = bartlettJC(v1, 4);
bartlett116 = bartlettJC(v1, 16);
bartlett24 = bartlettJC(v2, 4);
bartlett216 = bartlettJC(v2, 16);

errors(1) = calcerrorJC(P1, bartlett14);
errors(2) = calcerrorJC(P1, bartlett116);
errors(3) = calcerrorJC(P2, bartlett24);
errors(4) = calcerrorJC(P2, bartlett216);
```

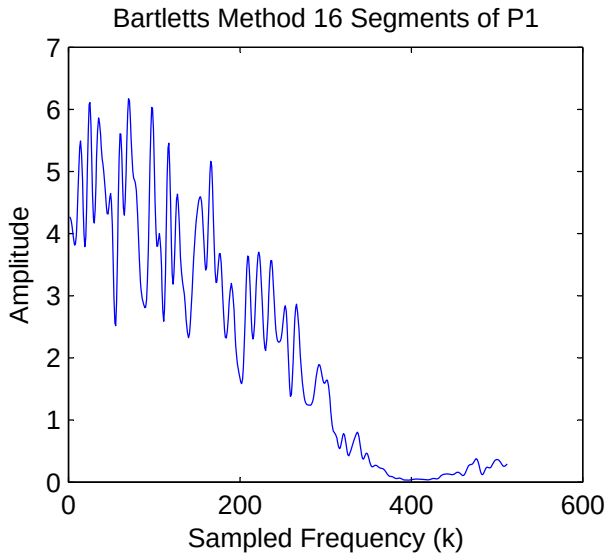
The following page contains the PSD estimates using Bartlett's Method.

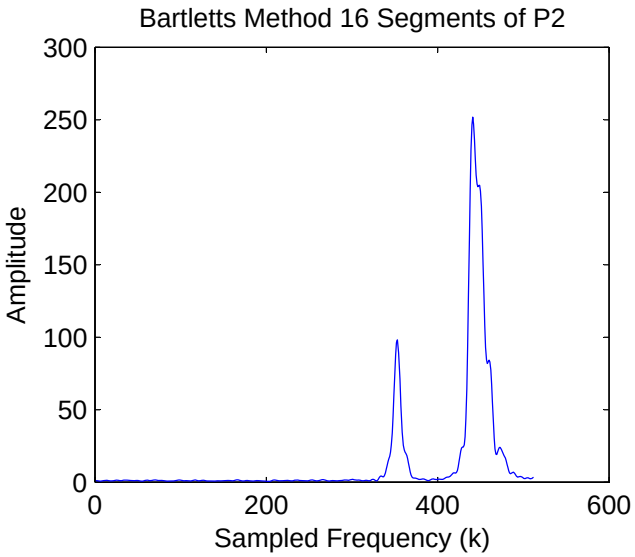

PSD of P1
Amplitude
Sampled Frequency (k)
0
1
2
3
4
5
0
200
400
600


PSD of P2
Amplitude
Sampled Frequency (k)
0
50
100
150
200
250
300
350
0
200
400
600


Bartletts Method 4 Segments of P1
Amplitude
Sampled Frequency (k)
0
2
4
6
8
10
12
0
200
400
600


Bartletts Method 4 Segments of P2
Amplitude
Sampled Frequency (k)
0
50
100
150
200
250
300
350
0
200
400
600


Bartletts Method 16 Segments of P1
Amplitude
Sampled Frequency (k)
0
1
2
3
4
5
6
7
0
200
400
600


Bartletts Method 16 Segments of P2
Amplitude
Sampled Frequency (k)
0
50
100
150
200
250
300
0
200
400
600

A.1 – Results

	Error for *v1* and *P1*	Error for *v2* and *P2*
Periodogram	1.055×10^{1}	2.082×10^{3}
Bartlett 4 Segments	1.972×10^{0}	3.966×10^{2}
Bartlett 16 Segments	3.746×10^{-1}	2.607×10^{2}
Welch 4 Segments 12.5% Overlap	2.098×10^{0}	3.727×10^{2}
Welch 4 Segments 25% Overlap	2.091×10^{0}	2.804×10^{2}
Welch 4 Segments 50% Overlap	2.953×10^{0}	4.725×10^{2}

Surprisingly, in both cases, Bartlett's Method with 16 segments did the best according to our error measure. However, it should be noted that both Welch's Method and Bartlett's Method with 4 segments do somewhat similar.

Clearly, the straight periodogram of the signal always does the worse. This is because with a modified periodogram, we are able to average multiple segmented periodograms. This averaging reduces the variance, thus reducing the energy of the error with which we are concerned.

The biggest drawback of using many smaller segments of the data and averaging to reduce the variance is that we are windowing the signal to smaller chunks of data. When we take the periodogram of a smaller window of data, the effect is convolving the spectrum with a sinc shaped waveform (actually sin(·)/sin(·)) in the frequency domain. This will basically smear the peaks in our estimated PSD. The smaller the window we use in the time domain, the wider the sinc shaped waveform is in the frequency domain. Therefore, when we window the signal more and more, we are losing resolution in the frequency domain as a result of improving our variance. The reason why our 16 segment Bartlett Method is doing better than the other methods is because we are windowing our signal more to reduce our variance, but not to the point of losing too much resolution. In *v1*, there are no resolution problems because there is only one peak. In *v2*, there are two peaks, and they are separated by about 90 frequency samples of a 1024-PT DFT. We know that this 90 sample separation corresponds to just:

$$\omega = \frac{k}{N} = \frac{90}{1024} = 0.08789 \frac{\text{rads}}{\text{sample}}$$

With 16 windows of a 2048 sample sequence, each window is just a rectangle of width 128. Given the DTFT of a rectangle, we know the following:

$$rect\left[\frac{n - W/2}{W}\right] \Leftrightarrow \frac{\sin\left(\left(W+1\right)\frac{\omega}{2}\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\omega\frac{W}{2}}$$

$$rect\left[\frac{n-64}{128}\right] \Leftrightarrow \frac{\sin\left(\frac{129}{2}\omega\right)}{\sin\left(\frac{1}{2}\omega\right)} e^{-j64\omega}$$

The first zero of the transform occurs at $\omega = 2\pi/129 = 0.0487$ rads/sample. Therefore, it makes sense why 16 windows of length 128 are still able to resolve the two sinusoids in *v2* separated by 0.08789 rads/sample. Therefore, there are not really many drawbacks to using 16 segments, but the gains in the decreasing variance will decrease our overall error. This explains the results we are seeing, and why Bartlett's Method with 16 segments produced the best results.

A.2. Other Techniques

Another method to estimate the PSD of a signal is to use a parametric modeling to try and fit the PSD to a shape given the signal. One way to do this is solving the Yule-Walker equations. Not much Matlab code is needed for this part, because we can use *pyulear* to compute the PSD estimate. Note the need to change from a 2048-PT DFT to a 1024-PT DFT again.

The following code was used to estimate the PSD with parametric modeling:

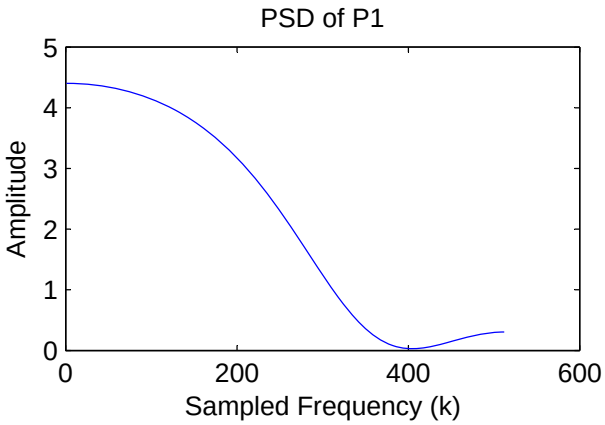
```
yw22048 = pyulear(v1, 2, 2048)'*pi;
yw42048 = pyulear(v1, 4, 2048)'*pi;
yw82048 = pyulear(v1, 8, 2048)'*pi;
yw12 = yw22048(1:2:length(yw22048));
yw14 = yw42048(1:2:length(yw42048));
yw18 = yw82048(1:2:length(yw82048));

yw22048 = pyulear(v2, 2, 2048)'*pi;
yw42048 = pyulear(v2, 4, 2048)'*pi;
yw82048 = pyulear(v2, 8, 2048)'*pi;
yw22 = yw22048(1:2:length(yw22048));
yw24 = yw42048(1:2:length(yw42048));
yw28 = yw82048(1:2:length(yw82048));

errors(1) = calcerrorJC(P1, yw2);
errors(2) = calcerrorJC(P1, yw4);
errors(3) = calcerrorJC(P1, yw8);
```

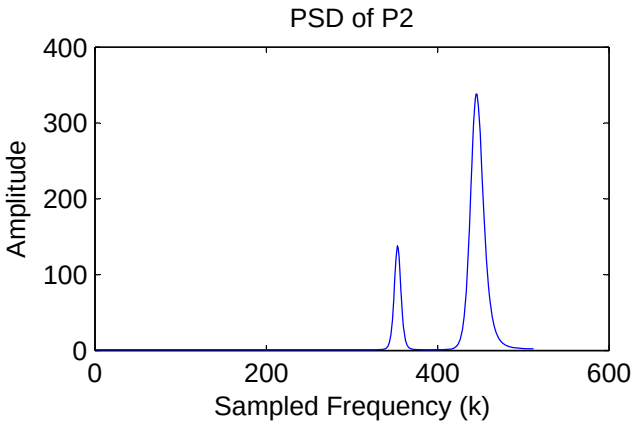
Note the factor of π in calculating the autoregressive estimate. This factor comes from Matlab's function pyulear normalizing the frequencies, and the factor of π is used to bring the amplitude of the PSD to where it should be.

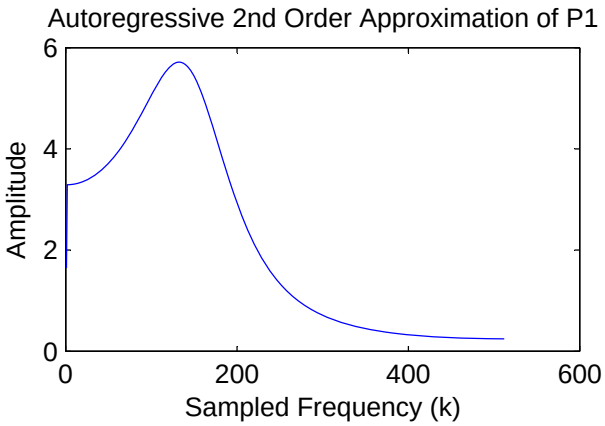
The following table contains the errors from each estimate:

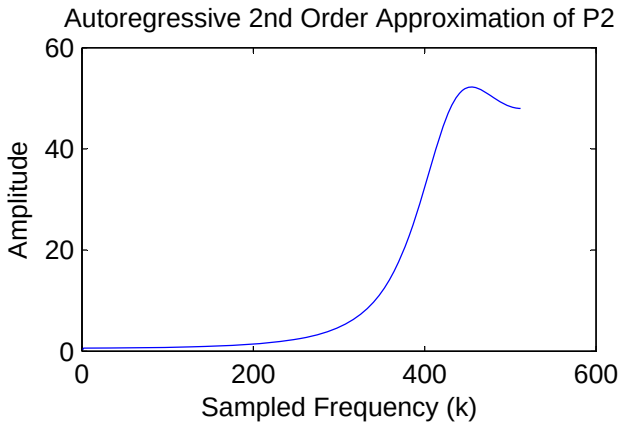
	Error for *v1* and *P1*	Error for *v2* and *P2*
Order 2	5.927×10^{-1}	2.292×10^{3}
Order 4	9.865×10^{-2}	2.203×10^{3}
Order 8	4.553×10^{-2}	8.796×10^{2}

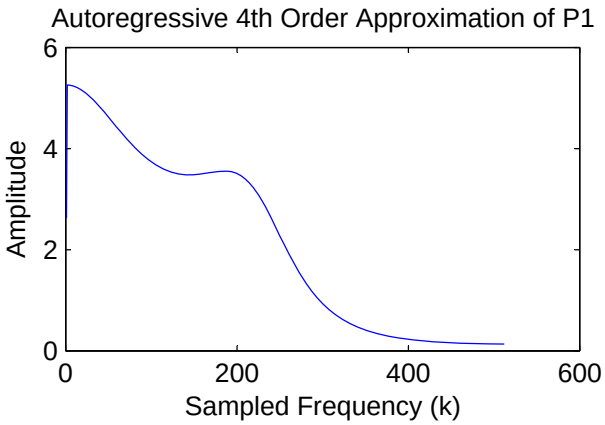
The following page contains the PSD estimates using autoregressive parametric modeling.

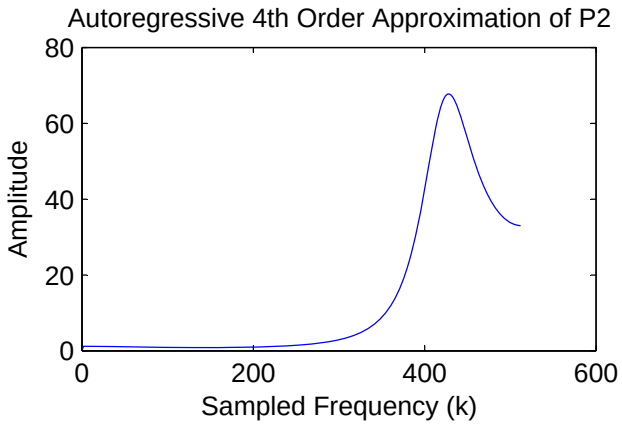

PSD of P1
Amplitude
Sampled Frequency (k)
0
1
2
3
4
5
0
200
400
600

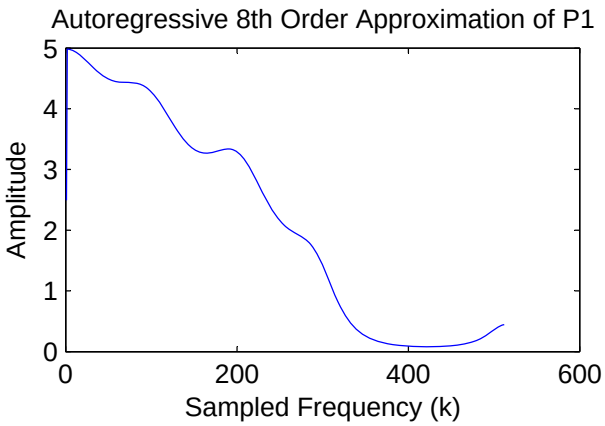

PSD of P2
Amplitude
Sampled Frequency (k)
0
100
200
300
400
0
200
400
600

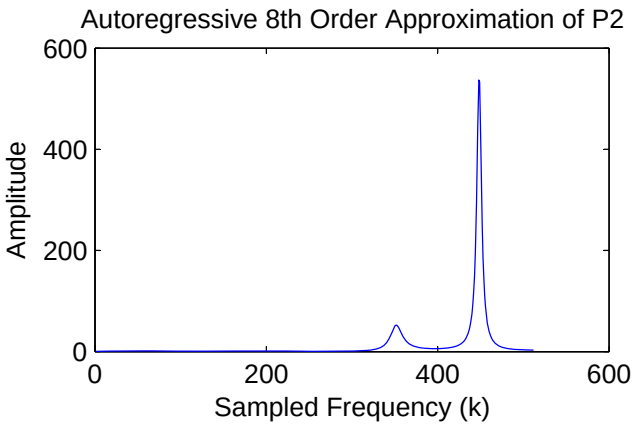

Autoregressive 2nd Order Approximation of P1
Amplitude
Sampled Frequency (k)
0
2
4
6
0
200
400
600


Autoregressive 2nd Order Approximation of P2
Amplitude
Sampled Frequency (k)
0
20
40
60
0
200
400
600


Autoregressive 4th Order Approximation of P1
Amplitude
Sampled Frequency (k)
0
2
4
6
0
200
400
600


Autoregressive 4th Order Approximation of P2
Amplitude
Sampled Frequency (k)
0
20
40
60
80
0
200
400
600


Autoregressive 8th Order Approximation of P1
Amplitude
Sampled Frequency (k)
0
1
2
3
4
5
0
200
400
600


Autoregressive 8th Order Approximation of P2
Amplitude
Sampled Frequency (k)
0
200
400
600
0
200
400
600

A.3. Reflection

The following table summarizes all spectral estimation attempts and their corresponding errors. The grayed errors are the lowest achieved with the prescribed methods.

	Error for *v1* and *P1*	Error for *v2* and *P2*
Periodogram	1.055×10^{1}	2.082×10^{3}
Bartlett 4 Segments	1.972×10^{0}	3.966×10^{2}
Bartlett 16 Segments	3.746×10^{-1}	2.607×10^{2}
Welch 4 Segments 12.5% Overlap	2.098×10^{0}	3.727×10^{2}
Welch 4 Segments 25% Overlap	2.091×10^{0}	2.804×10^{2}
Welch 4 Segments 50% Overlap	2.953×10^{0}	4.725×10^{2}
Autoregressive Order 2	5.927×10^{-1}	2.292×10^{3}
Autoregressive Order 4	9.865×10^{-2}	2.203×10^{3}
Autoregressive Order 8	4.553×10^{-2}	8.796×10^{2}

Clearly, the autoregressive all-pole modeling estimator worked very well for the first signal, and not so well on the second signal. This is because *v1* had a smoother PSD which could be more easily modeled with small ordered all-pole systems. The second signal was best estimated with Bartlett's Method or Welch's Method. The errors were both comparable. Upon further inspection, it seems like the extra 12 segments between the two Bartlett's Methods really made a difference. I tried implementing Welch's Method with 16 segments, and achieved approximately the same result as Bartlett's Method with 16 segments. The main concept to get from the second signal is that there are two main peaks existing in the PSD. To achieve the best results, we should use the largest window possible without losing any resolution. Either method will return similar results because the main contributor to the increased success is the averaging to reduce the variance.

In conclusion, smoother PSDs are more easily approximated with somewhat small ordered autoregressive modeling, while PSDs that change faster are more easily approximated with Bartlett's or Welch's Method for modified periodograms. If instead we aimed at identifying specific sinusoidal components, we can see what would happen from *v2*, which is composed of two sinusoidal components.

Taking the local maxima from each method, we have the following sinusoid estimates:

	Peak 1		Peak 2	
	Freq. Index (k)	Amplitude	Freq. Index (k)	Amplitude
Actual PSD	353	138.0	445	338.3
Periodogram	353	231.0	443	705.7
Bartlett 4 Segments	353	154.3	441	326.5
Bartlett 16 Segments	353	98.2	441	251.8
Welch 4 Segments 12.5% Overlap	352	140.1	451	348.1
Welch 4 Segments 25% Overlap	353	146.5	448	401.9
Welch 4 Segments 50% Overlap	352	179.9	443	344.9
Autoregressive Order 2	455	52.2	N/A	N/A
Autoregressive Order 4	428	67.7	N/A	N/A
Autoregressive Order 8	352	52.2	448	536.8

Notice that the autoregressive method still does poorly, especially with orders smaller then 8 when it can not even pick out two peaks. However, what is more interesting is that Bartlett's Method with 4 segments does better than with 16 segments. This can be attributed to the poorer resolution achieved by the 16 segment windows. As stated previously, more windows in Bartlett's Method corresponds to smaller windows for each periodogram. This windowing operation is equivalent to convolving the frequency domain with a wider sinc shaped waveform. This convolution will effectively stretch the peak in the frequency axis, and shrink the amplitude. With poorer resolution, the 16 segment Bartlett's Method will lose some of the vital sinusoid information because the amplitude is decreased. Therefore, if we know the signal is composed solely of a few sinusoids, fewer segments may actually be preferred.

I expected that Welch's Method with more overlap would have done better than Welch's Method with smaller overlap because of the same reasoning: more overlap corresponds to longer segments. The 50% overlap did do better than the 25% overlap to some extent, but oddly enough, the 12.5% overlap did the best. The only reasoning I see fit for this is that for this specific sequence, something random occurred and produced an odd result. It is important to keep in mind that we are indeed still estimating power spectral densities from random signals. Thus, only looking at one or two signals to compare estimation techniques is not very thorough. To accurately compare methods, we should consider many more signals, and signals with longer lengths. However, I still believe that in general, given the signal consists of two sinusoidal components, longer windows should do better in both Bartlett's and Welch's Method.

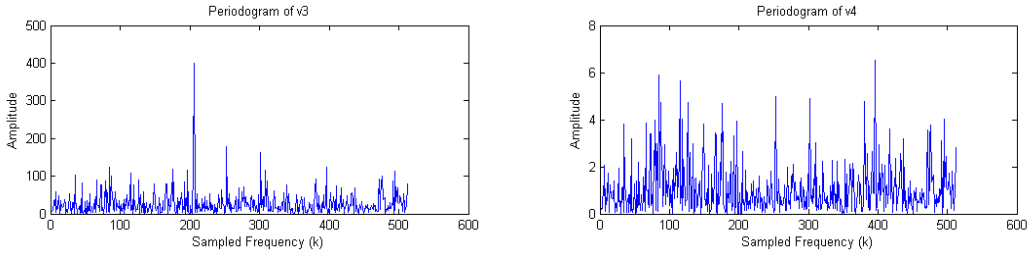
Part B. Analysis of an Unknown Signal

Given two signals, *v3* and *v4*, we were asked to analyze them and determine which signal came from either three to five sinusoids on top of white noise, or an autoregressive process of order 8. Let's first try to think of what these two random signals' PSDs should look like if we had the exact curve.

The PSD of one sinusoid should have one high peak at one frequency, and be nearly zero everywhere else. Multiple sinusoids will have peaks at each frequency and be zero everywhere else as long as they are separated by enough that we can resolve them. White noise will just have a constant bias floor that is equal to the variance of the noise. Therefore, the ideal PSD will look like three to five (depending on how many sinusoids it actually consists of) peaks plus some constant bias. Of course, with a random signal, we will usually have some variance in the estimate, and therefore, the bias from the white noise will just look like very fast (but somewhat small compared to the sinusoid peaks) oscillations throughout the whole signal.

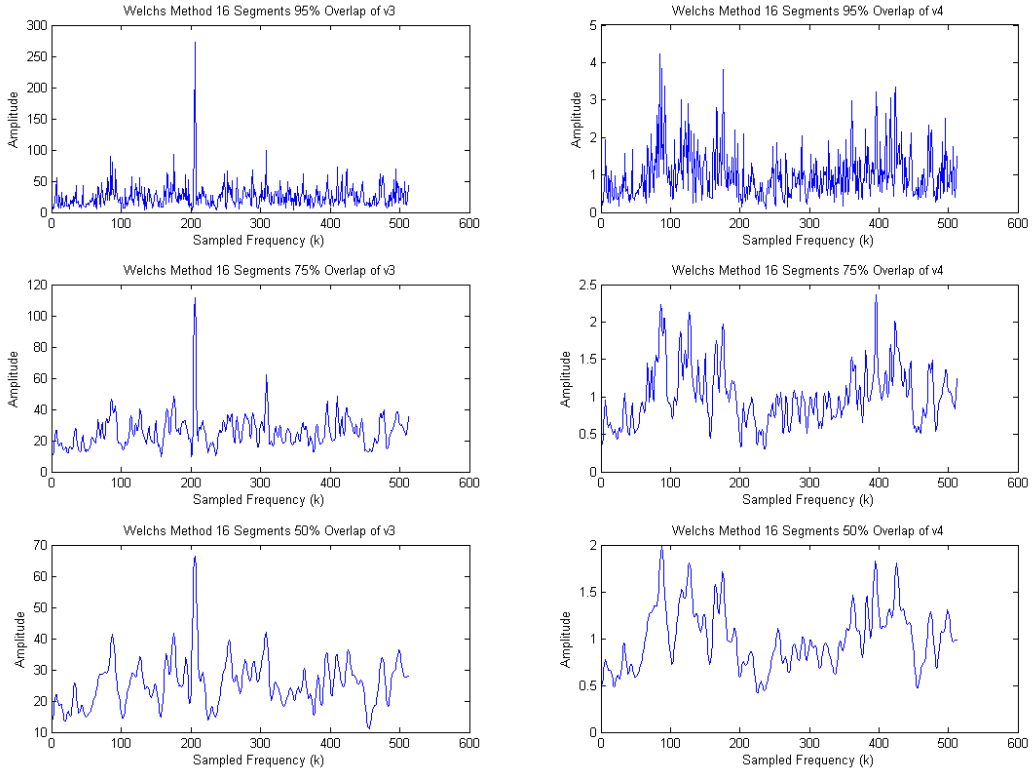
The PSD of an autoregressive process of order 8 should look fairly smooth. It will most likely not have very large and discrete peaks. The underlying shape of the estimate should therefore look something like a polynomial.

Without much direction, let's first just blindly take the periodogram to get an idea of what the signals' spectral energies look like. The following two plots are the periodograms of *v3* and *v4* computed using a 1024-PT DFT.



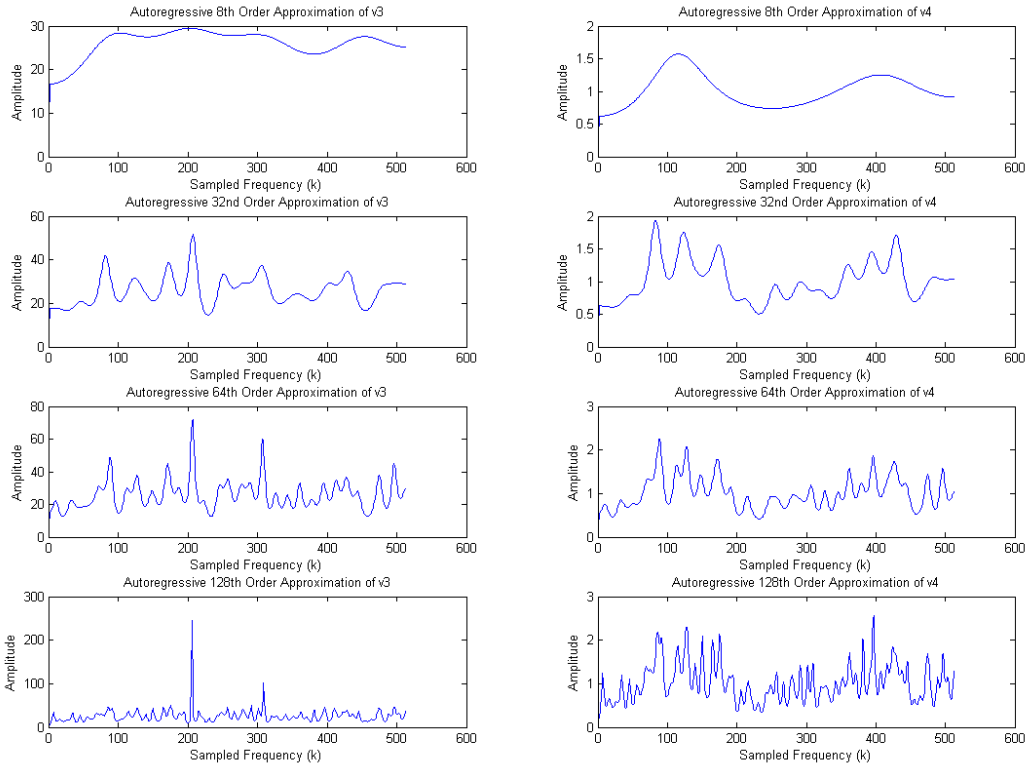
As an initial guess, it looks like *v3* is the sinusoidal PSD, and *v4* is the autoregressive PSD. Clearly, *v3* has one very large peak at around k=200. *v4* on the other hand doesn't really have very large peaks, but rather peaks of the same amplitude throughout the entire periodogram. This initial guess already seems pretty accurate, but let's continue with our analysis to make sure this argument is sound.

Instead of taking a straight periodogram, let's use Welch's Method to try and get a better approximation to the PSD with less variance. As stated previously, more segments decrease the variance. Shorter windows worsen the resolution. Here, we will try to use 32 segments, with a fairly large set of percentages to overlap. This larger overlap allows us to try and lower the variance while still combating the resolution problem. Even though this has some underlying problems (reusing portions of the signal many times), it will still give us a good idea on the PSD of the signal. The following figures depict estimates to the PSD using Welch's Method.

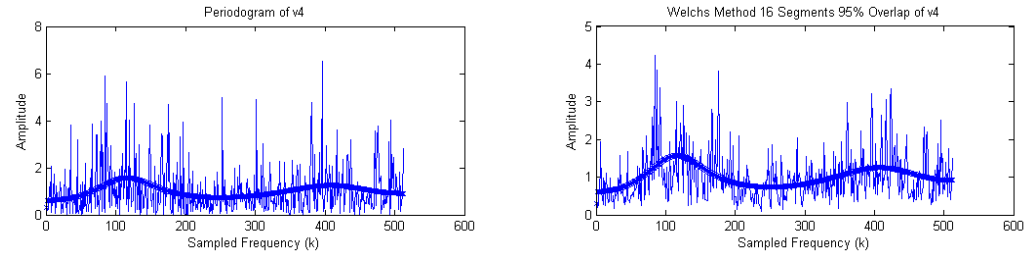


Again, these figures lead us to believe that *v3* is the sinusoidal signal. Notice the large peak at k=200, most likely corresponding to one sinusoid. It looks like another one may exist at the frequency index k=30. Also, we can see why resolution becomes a problem as we decrease the overlap percentage, in effect, decreasing the window size. Notice how the very sharp peak in the 95% case becomes wider and wider as the overlap decreases. The figures on the right still seem to not have any dominating peaks. They still seem to have a rather smooth underlying shape, which indicates that *v4* is most likely not composed of sinusoids.

To explore even further, we can model each input with an autoregressive estimate. Here, we will take a few different orders for the autoregressive estimate to illustrate a point. The figures are attached on the following page. If one of the signals was indeed generated from a PSD that was an autoregressive order 8 system, then we should be able to model the PSD with an order 8 system very well. In other words, the shape of the model should not change much as we increase the order (though it may oscillate more). Clearly, the figures on the left change a lot between the 8^{th} order case and the 128^{th} order case. Notice how peaks appear in the 128^{th} order case that were not apparent at all in the 8^{th} order case. However, for the figures on the right, the general underlying shape does not change. There is always a bump in the energy (and thus greater variance) around k=100, and another one at k=400. These observations lead us again to believe that *v3* is composed of sinusoids, and that *v4* is an autoregressive process of order 8.



Another interesting point is that since we know the autoregressive process is of order 8, we can try overlaying the approximation of order 8 on all the PSD estimates. The shape from this approximation should match the underlying shape from the periodogram and modified periodograms. The following two figures show the 8th order autoregressive estimate (thick line) overlaid on top of the periodogram and Welch's Method (thin line).



Clearly, this shape matches the underlying shape of the periodograms. At higher energies, the variance is higher and thus the actual periodogram may stray farther from the shape. However, this result shows that *v4* is indeed the result from an 8th order autoregressive random process.

All of the above arguments conclude the same thing. Thus, the only plausible solution to the proposed problem is that *v3*'s PSD is composed of a few sinusoids on top of white noise, and *v4*'s PSD is the output to an 8th order autoregressive process.