Technical Report

NAG8-1342 Supplement 1
UAH Account: 5-34744
April 1, 1997 - March 31, 1998

Attitude Estimation Signal Processing: A First Report on Possible Algorithms and Their Utility

July 7, 1998

Prepared by: Vahid R. Riasati
Visiting Assistant Professor
Electrical Computer Engineering Department
The University of Alabama in Huntsville
Huntsville, Alabama

Introduction

In this brief effort, time has been of the essence. The data had to be acquired from APL/Lincoln Labs, stored, and sorted out to obtain the pertinent streams. This has been a significant part of this effort and hardware and software problems have been addressed with the appropriate solutions to accomplish this part of the task.

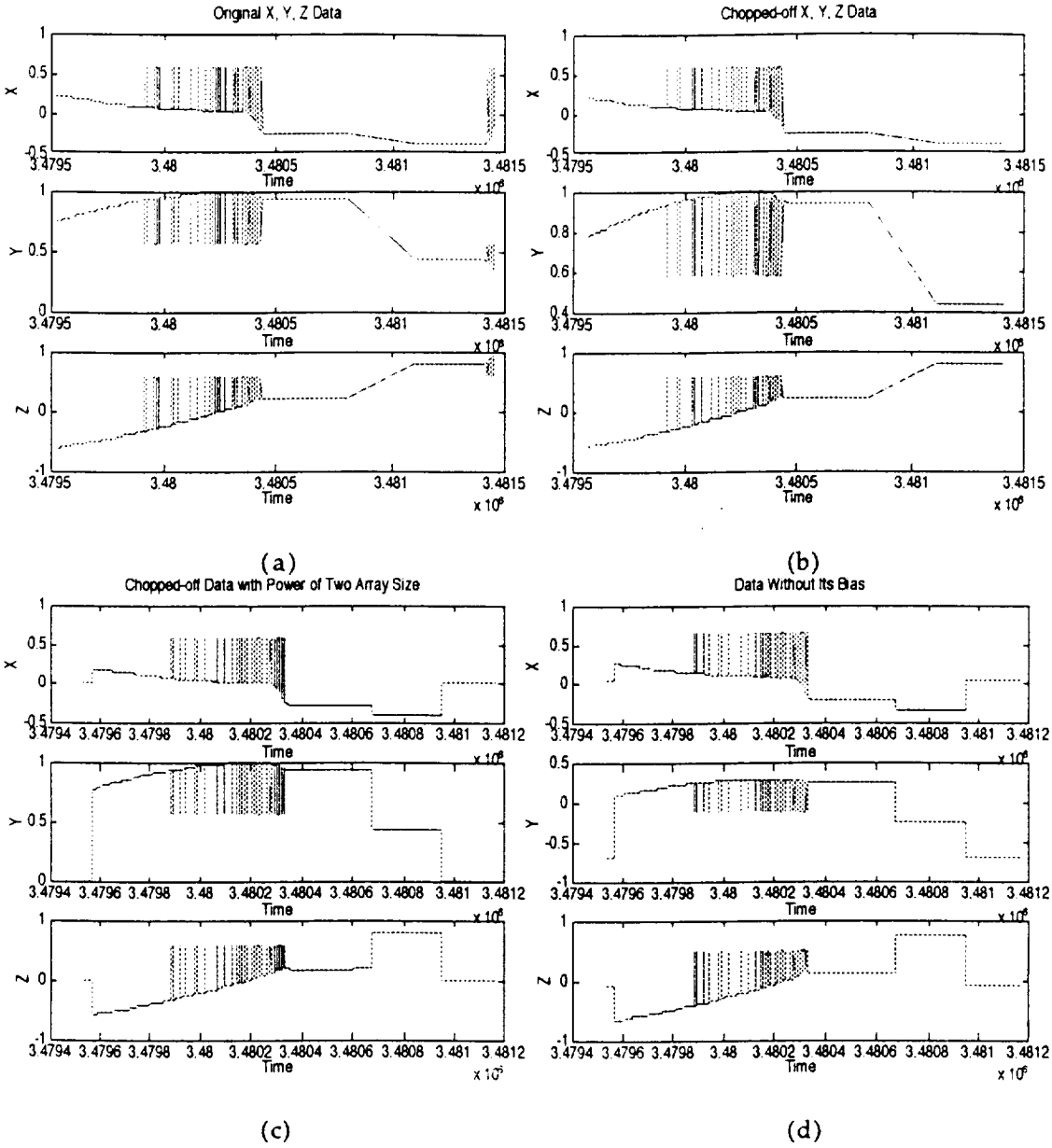
Passed this, some basic and important algorithms are utilized to improve the performance of the attitude estimation systems. These algorithms are an essential part of the signal processing for the attitude estimation problem as they are utilized to reduce the amount of the additive/multiplicative noise that in general may or may not change its structure and probability density function, pdf, in time. These algorithms are not currently utilized in the processing of the data, at least, we are not aware of their use in this attitude estimation problem. Some of these algorithms, like the variable thresholding, are new conjectures, but one would expect that someone somewhere must have utilized this kind of scheme before. The variable thresholding idea is a straightforward scheme to use in case of a slowly varying pdf, or statistical moments of the unwanted random process. The algorithms here are kept simple but yet effective for processing the data and removing the unwanted noise. For the most part, these algorithms can be arranged so that their consecutive and orderly execution would complement the preceding algorithm and improve the overall performance of the signal processing chain.

In any signal processing scenario, the utilization of particular algorithms are justified by evaluating the specific conditions of the data and the effort. In the current scenario, the overall goal is the development of some basic algorithms for pre-processing of data. These algorithms include a raw data thresholding at a constant threshold that would take out any extreme values that are just not expected to occur in relation to the other data samples in the

attitude control signal. This can be followed by a time windowing to reduce leakage problems from data segmentation, followed by a more fine-tuned thresholding scheme, perhaps similar to the variable thresholding scheme alluded to earlier. The result of this processing should reduce a significant amount of noise and clutter. At this point one could apply a frequency windowing scheme to limit the throughput information to that of the signal bandwidth estimate. Since signal precision information is of extreme importance in this scenario, one may want to limit the frequency of the signal by a window that has a cut-off frequency at a higher frequency than the bandwidth of the signal. This cut-off frequency information can be specified to the filter and the stopband for the filter can be adjusted for each particular signal. In the simulations a prompt is utilized that asks for the signal bandwidth information and the type of frequency limiting that is desired. These basic algorithms make up the pre-processing chain, a step towards the actual main stream processing of the data a linear estimator is also implemented for use with a Wiener filter to help extract the signal from the noisy and corrupted data.

Preconditioning

In all that follows the DC440200029star data, converted from quaternian format to x,y, and z coordinates, has been used. Prior to using the data the first and the last 425 samples are removed. By observing the data it has been determined that these data points should not be included as part of the data collection events due to transient effects of the system prior to the data collection event. The remaining data is loaded into the smallest array that can contain the data samples and is an integer power of two. The use of array sizes that are integer powers of two helps in processing time and is useful in the implementation of some signal processing algorithms. Figure (1) shows the original data, the chopped-off data, and the data in the smallest power of two array that would hold the data after 850 samples are removed.



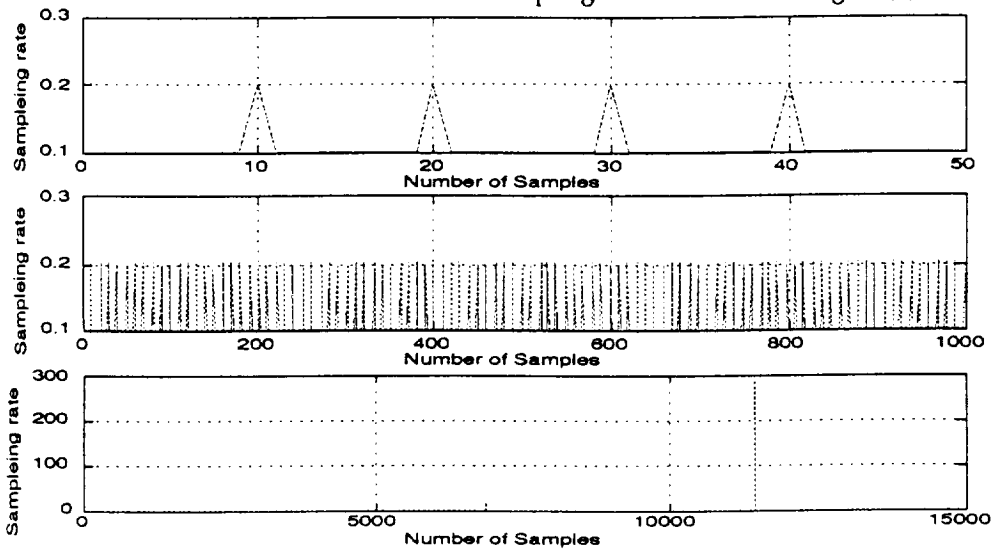
Figure(1), (a) The original coordinates in X, Y, and Z, (b) the data after removal of transient data that is not part of the data collection event, (c) the data after it has been placed in an array that is a the smallest power two which can contain this data, (d) the data after the constant bias has been removed.

To work with the data more effectively and be able to observe the relative magnitude of each of the frequency components of the signal more clearly, the dc bias of the data is

removed. The constant bias information is in general of little use in signal processing scenarios since this information relates to no variation in the signal. Hence, all that is really needed is the actual value of the bias so that it can be added back to the signal at the end of the processing chain. The constant bias is added back in to the signal because for most electronic circuits the signal is useless unless the correct bias level is used. Ultimately, the goal is to recover the original command signal from the noisy and corrupted data; this includes the bias and relative delays of the data. Most of the algorithms, with the exception of the Linear Estimator and Wiener filter, presented here can be thought of as preludes to the actual signal recovery and estimation algorithms.

Interpolation

A comparison of figures (1b) and (1c) reveals a change in the appearance of the data. This is attributed to the variation in the sampling rate as shown in figure(2).

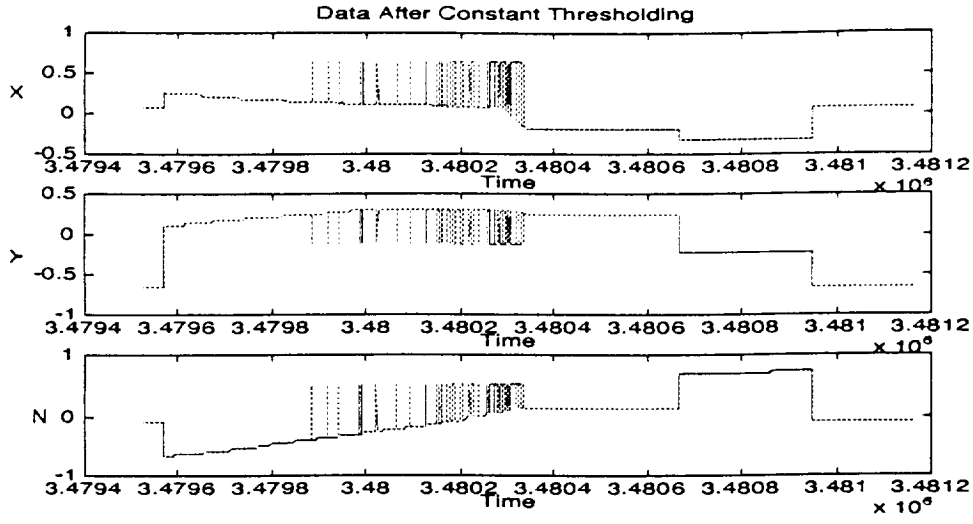


Figure(2), The variation in the sample rate is depicted at different sample densities. This shows a wide range of variation in the sampling rate for consecutive samples.

The data presented in this report is not sampled at a regular spacing. The relative sample spacing for consecutive samples shown in figure (2) indicates this clearly. This irregularity implies that some type of curve fitting or interpolation must be utilized to estimate the necessary samples that are missing. This problem is significant and should be addressed prior to the actual implementation of the simulated algorithms that are discussed in this report.

Constant Thresholding

To provide basic removal of unlikely noise events a constant thresholding has been implemented to remove data events that are three standard deviations above the standard deviation, SD, of the data. A constant pdf is assumed for the SD calculation of the data. This is justified by noting that there are a number of processes in this data and some of these processes are changing with time; these processes are usually smooth and decreasing as the random variable moves away from its mean value, or at worst the probability of the event occurring will remain constant as the random variable moves away from its mean. Even if the pdf were constant for a random variable, the mere fact that there are several additive random variables in this data would imply, by the central limit theorem, that the pdf of the sum of these variables will fall off as variable's values move away from the mean value. Hence, by calculating a generic SD for the data using a constant density and a threshold value that is two or three times larger than this SD unlikely events that would follow any usual density should be removed from the data. This is a good first attempt at thresholding, because the chances that this process would remove any of the command data are extremely small. This is due to a high threshold relative to the variation of the data and the increase in the possibility-of-occurrence for highly unlikely events in the moment calculations through the use of a constant pdf.



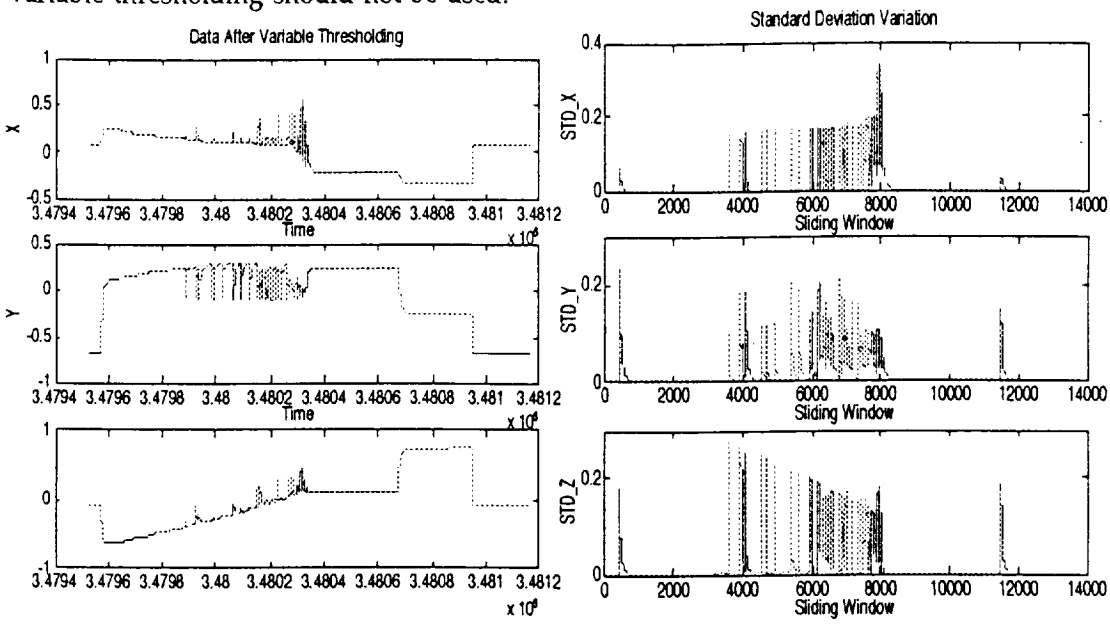
Figure(3) (a) The data after constant thresholding. STD_X=0.2317, STD_Y=0.3559, STD_Z=0.4200; Th1=3.*STD_X;Th2=3.*STD_Y; Th3=3.*STD_Z;

Variable Thresholding

The variable thresholding starts with the assumption that most random processes are not stationary and change in time. So, the idea of a wide-sense stationary process can be used to calculate the standard deviations along windowed segments of the data. In other words, adjustable segments in the temporal data are utilized on the basis that relative to the windowed segments of data the variation of the noise structure is slow and is not significant over the window where the first and the second statistical moments are being calculated.

The size of the window is chosen rather arbitrarily for the examples used in this report, however, this choice must include the sampling rate, or in other words the largest frequency component of the signal. The current results are obtained with sliding windows that contain 10 samples. Figure (4) shows the results of applying the variable thresholding idea to the data, along with the variation of the SD of the data. The variation in the SD of the data

is provided as an instructive measure to help in communicating the relative change in the variance that is utilized in thresholding the various windowed segments as the algorithm progressively includes new samples. A basically flat plot for the SD would imply that variable thresholding should not be used.



(a) (b)

Figure(4) (a) The data after variable thresholding of window size ten is applied to the data (b) the variation in the standard deviation of the data over the progressive ten sample windows.

Time Windowing

The purpose of data windowing is to modify the relationship between the spectral estimate Pk at a discrete frequency and the actual underlying continuous spectrum P(f) at nearby frequencies. In general, the spectral power in one "bin" k contains leakage from frequency components that are actually s bins away. This is due to the limitation that is put on the function by the finite size of the FT window. There is quite substantial leakage even from moderately large values of s.

When a run of N sampled points for periodogram spectral estimation is selected, in effect, an infinite run of sampled data cj is multiplied by a window function in time, one which is zero except during the total sampling time ND, and unity during that time. In other words, the data are windowed by a square window function. By the convolution theorem (interchanging the roles of f and t), the Fourier transform, FT, of the product of the data with this square window function is equal to the convolution of the data's FT with the window's FT. This is nothing more than the square of the discrete FT of the unity window function.

$$W(s) = \frac{1}{N^2}\left[\frac{\sin(\pi s)}{\sin\left(\frac{\pi s}{N}\right)}\right]^2 = \frac{1}{N^2}\left|\sum_{k=0}^{N-1} e^{j2\pi sk/N}\right|^2$$

The reason for the leakage at large values of s, is that the square window function turns on and off so rapidly. Its FT has substantial components at high frequencies. To remedy this situation, we can multiply the input data cj, j=0, 1, ..., N-1 by a window function wj that changes more gradually from zero to a maximum and then back to zero as j ranges from 0 to N-1.

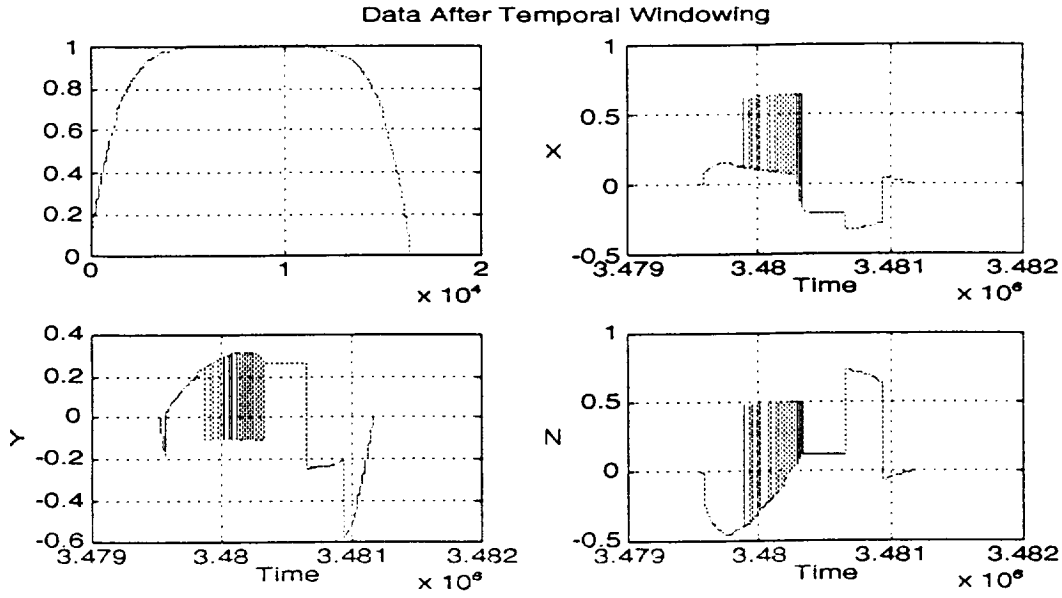
There are a variety of different windows that have been utilized by different people, these include the Hanning window, the Parzen window, and the Welch window. The difference between these various windows is measured by various metrics including the sidelobe fall-off (dB per octave), 3-dB bandwidth, highest sidelobe level (dB), et cetera. Roughly speaking, the principle tradeoff is between making the central peak as narrow as possible versus making the tails of the distribution fall off as rapidly as possible.Window functions that rise smoothly from zero to unity quickly (~in the first 10% of the data) stay maximum and fall off quickly (~in the last 10% of the data) are thought to be more desirable since they are usually more narrow in the main lobe of the leakage function, however, this must be traded with the widening of the leakage tail by a significant factor (the reciprocal of 10%, a factor of ten). If we distinguish between the width of a window (number of samples for which it is at its maximum value) and its rise/fall time (number of samples during which it rises and falls); and

if we distinguish between the FWHM (full width to half maximum value) of the leakage function's main lobe and the leakage width (full width that contains half of the spectral power that is not contained in the main lobe); then these quantities are related roughly by

$$FWHMinbins \sim \frac{N}{windowwidth}$$

$$leakagewidthinbins \sim \frac{N}{windowrise, falltime}$$

Figure(5) shows a variation of the Welch window which is used in the simulations.



Figure(5) Top left to bottom right: Window, X-data, Y-data, and Z-data after windowing.

After removing the mean of the data, the power spectrum should contain a large signal at a low frequency, then small signals centered at progressively higher frequencies, the noise. These small signals are what we would like to eliminate.

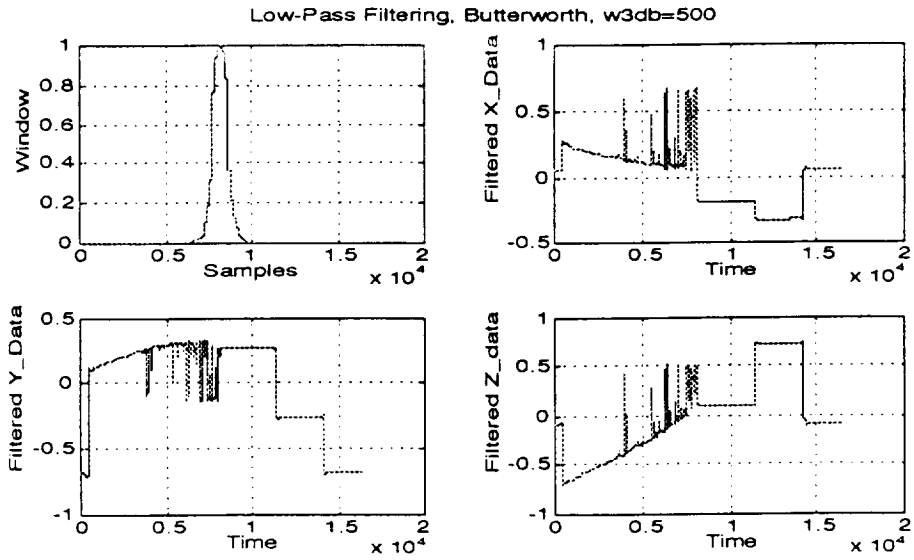
Low-Pass Filtering

Windowing the data in the frequency domain can make it smoother than it is now by removing some of the high frequency data. This is sometimes referred to as low pass filtering. Remember, however, that only a minimal range of the highest frequency information should be filtered out. This can be done by implementing a frequency windowing scheme with an adjustable cut-off frequency. One such window is a simple triangular function given by,

$f(w)=1-|w|$ for $-1<w<1$;

$=0$ elsewhere

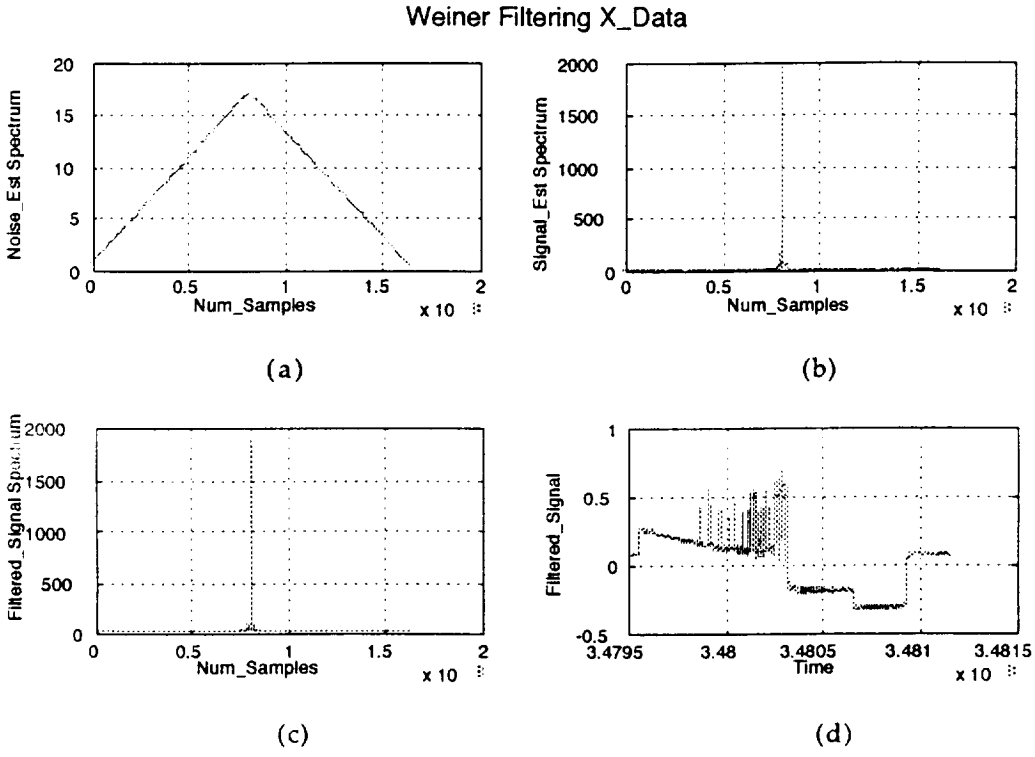
where w is a normalized frequency. A variation of this is to allow $|w|$ to take on powers higher than one. This will reduce the amount of high frequency information that is allowed, while increasing the magnitude of the low frequency contents of the data. The slope of the isosceles triangle relative to its base can be modified to incorporate the desired 3dB bandwidth. Other Low-pass filters that are implemented in the simulation are the Butterworth and the Square filters. Figure (6) shows the frequency characteristics of a second-order Butterworth filter and the results of its application to the data.



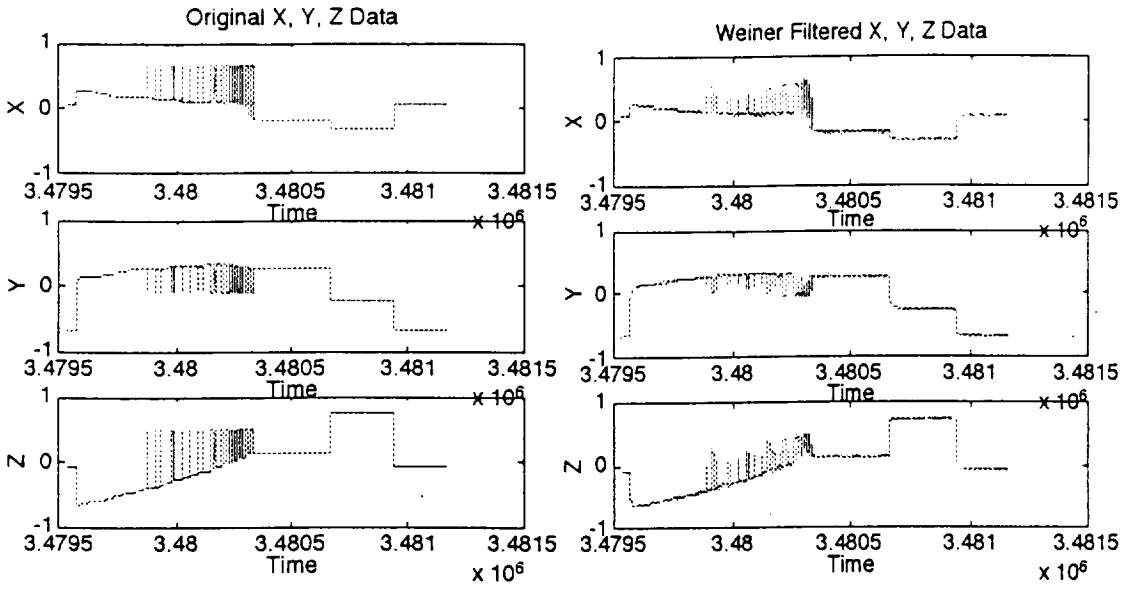
Figure(6) From top left corner to the bottom right corner the figures show a second-order low-pass Butterworth filter with a cut-off frequency of w3db=500 samples, and the results of applying the filter to the X_data, Y_data, and Z_data.

Linear Estimation and Wiener Filtering

The pre-processing algorithms that have been discussed should be followed by a series of algorithms that attempt to recognize and extract the command signal from the noisy data. As an example of this type of filtering algorithm a simple adaptive Wiener filter has been implemented to reduce the undesirable noise in the data. This filter uses a best-linear-fit to estimate the noise information and utilizes this information to extract the signal spectra from the noise; this is used to reduce the cluttering pink-noise in the attitude data. Figure(7) shows the results of applying the Wiener filter to the data.



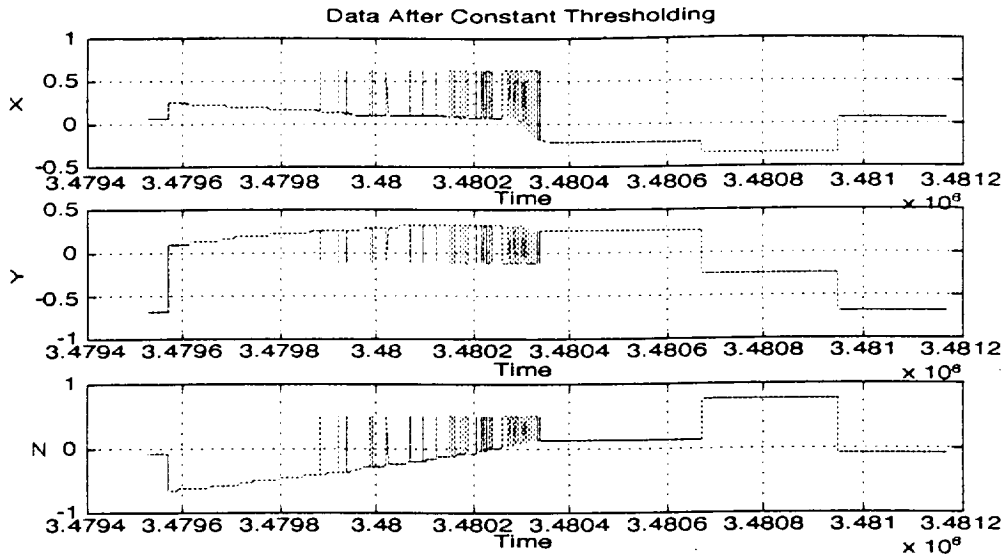
Figure(7), The Linear Estimation Wiener filter results, a) the estimate used for the noise spectrum, b) the noise and signal spectrum, c) the filtered signal spectrum, d) the signal after filtering. Figure(8) shows the results of all three coordinate data after wiener filtering has been applied to extract the signal.



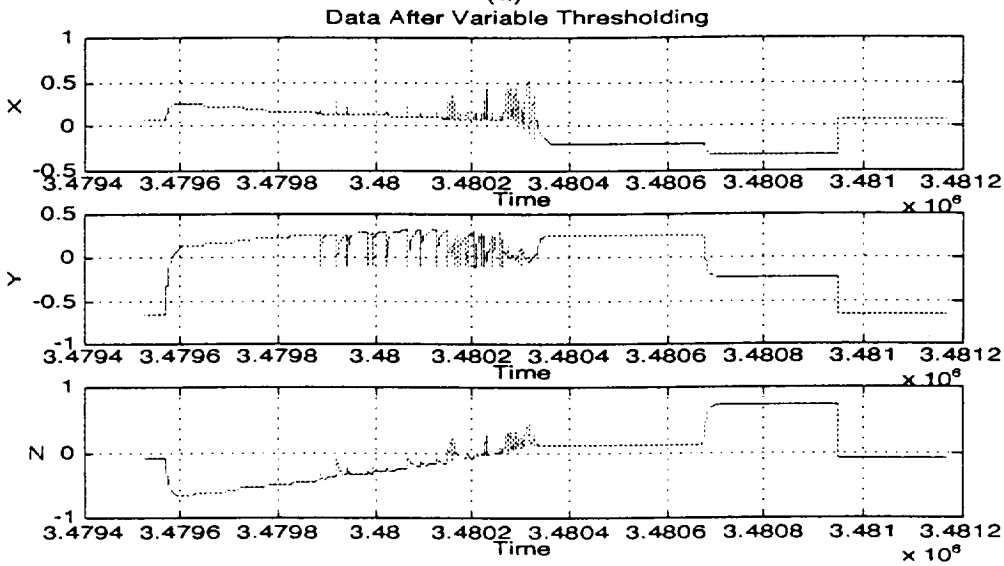
Figure(8), The X, Y, and Z coordinate data points are depicted prior and post Wiener Filtering.

The noise estimation can be made more sophisticated to include a more accurate model for the variations of the spectrum. This should improve the performance of the filter. Segmentation of the data should also help in the implementation of a more effective Wiener filter.

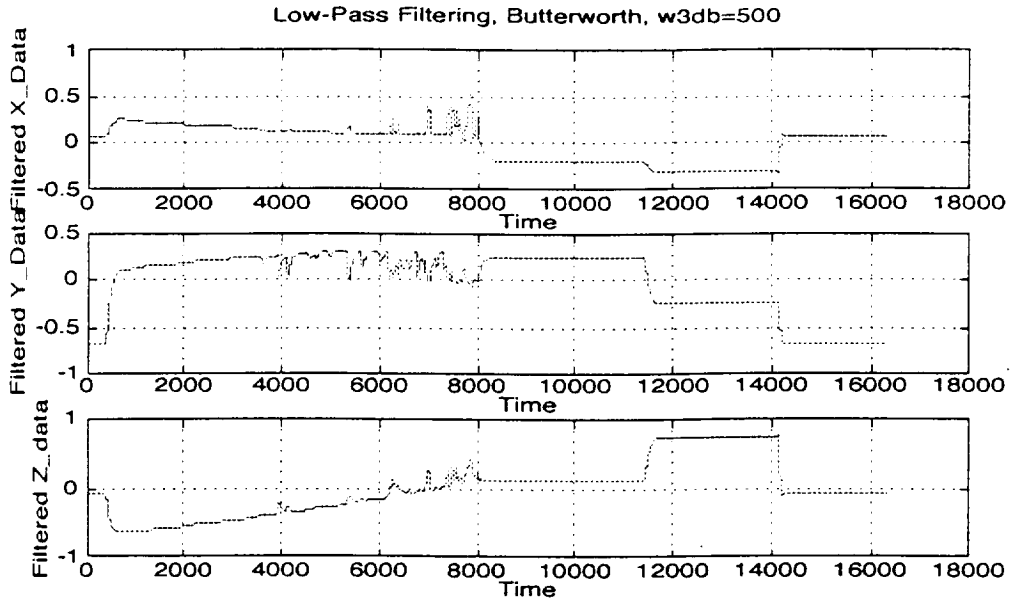
So far each of the algorithms have been applied to the original data to see the effect of each algorithm independent of the other algorithms. Another approach to utilizing these methods in the simulation is to implement two or more of these algorithms together, for example, the Butterworth filter may be utilized after the constant and variable thresholding in conjunction with the linear estimator and Wiener filtering of the data. The results of this particular combination of algorithms are presented in Figure(9).


Data After Constant Thresholding
X
Y
N
Time
x 10^6

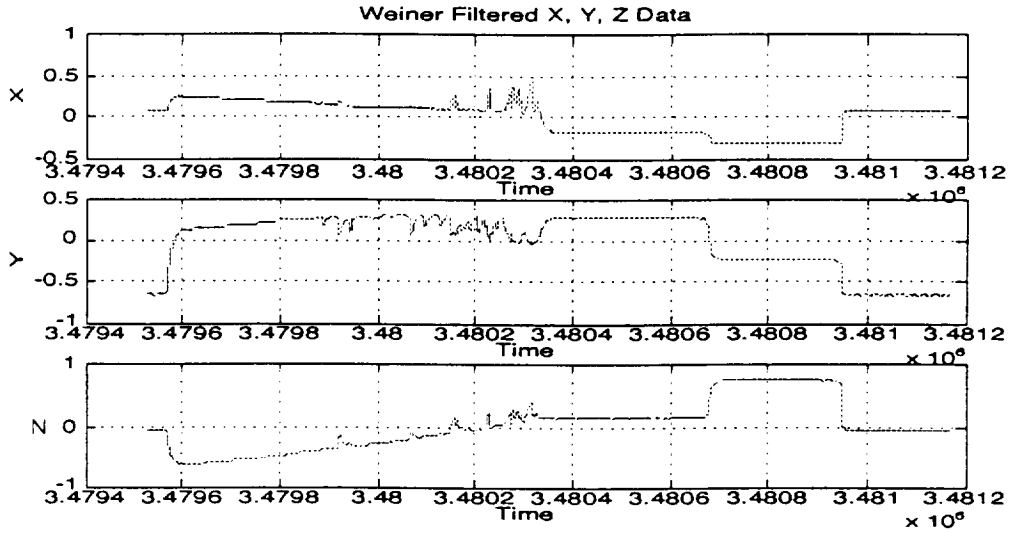
(a)


Data After Variable Thresholding
X
Y
N
Time
x 10^6

(b)



(c)



(d)

Figure(9), One possible combination of algorithms for a signal processing chain. All three data streams are processed by a) a constant threshold, b) a variable threshold, c) a low-pass Butterworth filter, and d) the linear estimator Wiener filter system.

Hence, a variety of different schemes for processing of the data can be utilized merely by re-configuring the code to this purpose. To facilitate this, the code has been implemented in

a modular format and flags have been incorporated to easily allow for the selection of the algorithms of choice. The appropriateness of the utility of these techniques depends on a number of variables, including the statistics of the data and the random structure variation. However, the basic algorithms are applicable to most data sets and should be useful in rejecting and reducing noise and estimating the signal.

The Remaining Work

If one were to compare this effort to writing an essay, it could safely be said that the outline for the essay has been done. Now, the work of writing and re-organizing the contents, spell-checking and grammatical correctness, of the essay must begin. The body of the essay can be compared to a series of tests of a consistent array of low-pass filters to determine their applicability to the particular data types that are under consideration here, then the results of these test must be evaluated with reasonable merits and the best filter(s) should be used. Of course a set of metrics are necessary to define "best" as far as these data are concerned. A remodeling of the noise estimation algorithm should be utilized to improve the performance of the Wiener filter. A set of three to five analytical models should be used and tested for performance evaluation on a set of typical attitude data sets. Similar metrics to those used for the low-pass filter evaluation could be utilized, even thought adjustments must be made to make the distinction between the objective function of each algorithm. Finally, a series of tests including the "best" choices of algorithms is necessary to choose the various algorithm-sets that are to be used in series for the over-all signal processing chain. At the end of these tests, a series of algorithms should be chosen for use with the various data sets. Finally, since all this falls under algorithm design and development, one should determine a global metric for the evaluation of the entire process. One method for achieving this is to compare the final

estimated command signal with the actual command signal, then do the same comparison with the signal prior to any processing. A relative metric of these two comparisons would then enable a measure of improvement which would determine the net worth of the signal extraction and noise removal algorithms. This metric is essential because it would help in answering a basic question that should always be asked and is often omitted in signal processing scenarios: is the processing time, energy and cost worth the improvement that is achieved in the final signal? The answer to this question is subjective from a signal processing point of view and has a strong bias towards the application and purpose of the signal processing chain.

Appendix

This section contains a copy of the code that is used to simulate the relavent algorithms discussed in this report. All of the results and figures have been implemented by the use of this code.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
% Comment Box:
%
% This code implements some preprocessing algorithms for
%  for the MSX data. The 'bulk data' is loaded in to the
%  code and the pertinent data is extracted and the dc-bias
%  is removed. After a constant thresholding algorithm the
%  regional variations of the data is utilized to remove data
%  samples that do not conform to the norm of the regional data
%  in a statistical sense. A temporal windowing scheme is also
%  implemented to reduce the frequency leakage of the window.
%  Three different low-pass filters are also implemented, any
%  one of which can be used to limit the high-frequency noise
%  input to the system. Finally a linear estimator is used
%  to estimate the noise in the data. This estimate is
%  used to extract the signal from the noise in a Wiener filter
%  approach.
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
%
% Load the data and extract pertinent information
%
```

```
load DC4402029coord;
m=size(DC4402029coord,1);
clg, t=DC4402029coord(:,1);clear a1;clear a2,clear a3
figure(1),subplot(311),plot(t,DC4402029coord(:,2)),
title('Original X, Y, Z Data')
ylabel('X'),xlabel('Time')
figure(1),subplot(312),plot(t,DC4402029coord(:,3)),
ylabel('Y'),xlabel('Time')
figure(1),subplot(313),plot(t,DC4402029coord(:,4))
ylabel('Z'),xlabel('Time'),    pause

figure(1),subplot(311),plot(t(425:m-425),DC4402029coord(425:m-425,2))
title('Chopped-off X, Y, Z Data')
ylabel('X'),xlabel('Time')
figure(1),subplot(312),plot(t(425:m-425),DC4402029coord(425:m-425,3))
ylabel('Y'),xlabel('Time')
figure(1),subplot(313),plot(t(425:m-425),DC4402029coord(425:m-425,4))
ylabel('Z'),xlabel('Time'),pause
%
% Choose the appropriate FLAGS for each desiered operation
% FLAG=1 => Perform the operation
%
bias_rem=1;dc_rem=0;th_c=1;th_v=1;tw=0;LPF=1;Est_filter=1;
%
% Extract the size of the data and utilize an arrary that
% is the smallest integer multiple of 2 and can accept all
% of the data.
%
if m>2^9;
k=2^(fix(log2(m))+1);    del_t=t(2)-t(1);
k1=2^(fix(log2(m))-9);t1=zeros([k,1]);
t1=[t(1):del_t:(k-1).*del_t+t(1)]';
else
s='Data set is not large enough';
end
a1=zeros(k,1);a2=zeros(k,1);a3=zeros(k,1);
a1(425:m-425)=DC4402029coord(425:m-425,2);
a2(425:m-425)=DC4402029coord(425:m-425,3);
a3(425:m-425)=DC4402029coord(425:m-425,4);
figure(2),subplot(311),plot(t1,a1)
title('Chopped-off Data with Power of Two Array Size ')
ylabel('X'),xlabel('Time')
figure(2),subplot(312),plot(t1,a2)
ylabel('Y'),xlabel('Time')
figure(2),subplot(313),plot(t1,a3)
ylabel('Z'),xlabel('Time'),pause
%clear DC4402029coord
%
% Bias Removal
%
if bias_rem==1
dc1=mean(a1);dc2=mean(a2);dc3=mean(a3);
a1=a1-dc1;a2=a2-dc2;a3=a3-dc3;
end;
```

```
%
% DC Removal
%
if dc_rem==1
LFR=3;
A1=fftshift(fft(a1));A2=fftshift(fft(a2));A3=fftshift(fft(a3));
dc1=A1(k/2-LFR:k/2+LFR);dc2=A2(k/2-LFR:k/2+LFR);dc3=A3(k/2-LFR:k/2+LFR);
A1(k/2-LFR:k/2+LFR)=zeros(size([k/2-LFR:k/2+LFR]));
A2(k/2-LFR:k/2+LFR)=zeros(size([k/2-LFR:k/2+LFR]));
A3(k/2-LFR:k/2+LFR)=zeros(size([k/2-LFR:k/2+LFR]));
a1=ifft(fftshift(A1));a2=ifft(fftshift(A2));a3=ifft(fftshift(A3));
end
%
% Constant Thresholding
%
if th_c==1
std1=std(a1);th1=mean(a1);
std2=std(a2);th2=mean(a2);
std3=std(a3);th3=mean(a3);
for i=1:m
if abs(a1(i))>=(3.*std1), a1(i)=th1; end;
if abs(a2(i))>=(3.*std2), a2(i)=th2; end;
if abs(a3(i))>=(3.*std3), a3(i)=th3; end;
end
figure(4),subplot(311),plot(t1,a1)
title('Data After Constant Thresholding')
ylabel('X'),xlabel('Time')
figure(4),subplot(312),plot(t1,a2)
ylabel('Y'),xlabel('Time')
figure(4),subplot(313),plot(t1,a3)
ylabel('Z'),xlabel('Time'),%pause
end;
%
% Adaptive Thresholding Using Absolute Values
%
if th_v==1
l=1
rss=10;
for j=1:1
for i=rss+1:m-850
a1_temp=a1(i-rss:i);  std1_temp(i)=std(a1_temp);th1=mean(a1(i-rss:i));
a2_temp=a2(i-rss:i);  std2_temp(i)=std(a2_temp);th2=mean(a2(i-rss:i));
a3_temp=a3(i-rss:i);  std3_temp(i)=std(a3_temp);th3=mean(a3(i-rss:i));
if  abs(a1(i))>=(th1+0.05.*th1.*std1_temp(i)),  a1(i)=th1;  end;
if  abs(a2(i))>=(th2+0.05.*th1.*std2_temp(i)),  a2(i)=th2;  end;
if  abs(a3(i))>=(th3+0.05.*th1.*std3_temp(i)),  a3(i)=th3;  end;
end;
figure(5),subplot(311),plot(t1,a1)
title('Data After Variable Thresholding')
ylabel('X'),xlabel('Time')
figure(5),subplot(312),plot(t1,a2)
ylabel('Y'),xlabel('Time')
figure(5),subplot(313),plot(t1,a3),%pause,
ylabel('Z'),xlabel('Time')
```

```
figure(6),subplot(311),plot(std1_temp)
title('Standard Deviation Variation')
ylabel('STD_X'),xlabel('Sliding Window')
figure(6),subplot(312),plot(std2_temp)
ylabel('STD_Y'),xlabel('Sliding Window')
figure(6),subplot(313),plot(std3_temp)
ylabel('STD_Z'),xlabel('Sliding Window'),pause
end

end;
%
% Temporal Windowing
%
if tw==1
ll=1
for l=1:k;
window(l)=1-(((l-1)-.5.*(k-1))./(0.5.*(k+1))).^6;
end;
figure(7),subplot(221),plot(window),grid
a1=a1.*window';
a2=a2.*window';
a3=a3.*window';
figure(7),subplot(222),plot(t1,a1),grid
ylabel('X'),xlabel('Time')
figure(7),subplot(223),plot(t1,a2),grid
ylabel('Y'),xlabel('Time')
figure(7),subplot(224),plot(t1,a3),grid
ylabel('Z'),xlabel('Time'),pause
gtext('Data After Temporal Windowing')
end;
%
% Low-pass filtering: 1) Square, 2) Triangular, 2) Butterworth
%
if LPF==1
w3db=input('what is the 3dB cutoff (in #of samples) for the filter '),
s=input('Which LPF would you like to use?(square, triangular, butterworth) ','s')
LPS=strcmp(s,'square')
LPT=strcmp(s,'triangular')
LPB=strcmp(s,'butterworth')
lll=1
% Low-Pass Filtering
Y=zeros([1,k]);
if LPS==1
Y(k/2-w3db:k/2+w3db)=1;
end;
if LPT==1
Y(k/2:k/2+2.*w3db)=1-(1./(2.*w3db)).*[0:2.*w3db];
Y(k/2-2.*w3db:k/2)=(1./(2.*w3db)).*[0:2.*w3db];
end;
if LPB==1
for i=1:k
Y(i)=1./(1+((i-k/2)./w3db).^4);
```

```
end;
end;
figure(8),subplot(221),plot(abs(Y)),grid
ylabel('Window'),xlabel('Samples')
A1=fftshift(fft(a1));A2=fftshift(fft(a2));A3=fftshift(fft(a3));
a=ifft(fftshift(A1.*(Y'.^1.0)));
aa=ifft(fftshift(A2.*(Y'.^1.0)));
aaa=ifft(fftshift(A3.*(Y'.^1.0)));
figure(8),subplot(222),plot(real(a)),grid
ylabel('Filtered   X_Data'),xlabel('Time')
figure(8),subplot(223),plot(real(aa)),grid
ylabel('Filtered   Y_Data'),xlabel('Time')
figure(8),subplot(224),plot(real(aaa)),grid
ylabel('Filtered   Z_data'),xlabel('Time')
gtext('Low-Pass Filtering, Butterworth, w3db=500')

figure(8),subplot(311),plot(real(a)),grid
ylabel('Filtered   X_Data'),xlabel('Time')
figure(8),subplot(312),plot(real(aa)),grid
ylabel('Filtered   Y_Data'),xlabel('Time')
figure(8),subplot(313),plot(real(aaa)),grid
ylabel('Filtered   Z_data'),xlabel('Time')
gtext('Low-Pass Filtering, Butterworth, w3db=500')
pause
end;
%
% Estimation and sub-optimal (Wiener) filtering
%
noise_estPSD=zeros(k,1);
index=[0:k/2-1];
if  Est_filter==1
        if LPF==1
                A1=fftshift(fft(a));A2=fftshift(fft(aa));A3=fftshift(fft(aaa));
                w3db_Est=w3db;
        else
                w3db_Est=input('what is the 3dB cutoff (in #of samples) for the Est_filter '),
                A1=fftshift(fft(a1));A2=fftshift(fft(a2));A3=fftshift(fft(a3));
        end;
sd=sum(abs(A1(w3db_Est:k/2)));
sdi=sum(abs(A1(w3db_Est:k/2)).*abs(index(w3db_Est:k/2)'));
si=sum(abs(index(w3db_Est:k/2)));
sii=sum(abs(index(w3db_Est:k/2)).*abs(index(w3db_Est:k/2)));
beta=(sdi-si.*sd)./(sii-si.^2);
alfa=sd-((sdi-si.*sd)./(sii-si.^2)).*si;
noise_estPSD(1:k/2)=alfa+beta.*index;
noise_estPSD(k/2+1:k)=noise_estPSD(k/2:-1:1)';
figure(9),subplot(221),plot(abs(noise_estPSD)),grid
ylabel('Noise_Est   Spectrum'),xlabel('Num_Samples')
sig_estPSD=sqrt(abs(A1.*conj(A1)-(noise_estPSD.^2)));
figure(9),subplot(222),plot(abs(sig_estPSD)),grid
ylabel('Signal_Est   Spectrum'),xlabel('Num_Samples')
fil_sig_PSD=((A1)./(1+((noise_estPSD)./(sig_estPSD+0.1)));
figure(9),subplot(223),plot(abs(fil_sig_PSD)),grid
ylabel('Filtered_Signal   Spectrum'),xlabel('Num_Samples')
```

```
signal1=ifft(fftshift(fil_sig_PSD));
figure(9),subplot(224),plot(t1,real(signal1)),grid
ylabel('Filtered_Signal'),xlabel('Time')
gtext('Weiner  Filtering  X_Data'),pause

sd=sum(abs(A2(w3db_Est:k/2)));
sdi=sum(abs(A2(w3db_Est:k/2)).*abs(index(w3db_Est:k/2)'));
si=sum(abs(index(w3db_Est:k/2)));
sii=sum(abs(index(w3db_Est:k/2)).*abs(index(w3db_Est:k/2)));
beta=(sdi-si.*sd)./(sii-si.^2);
alfa=sd-((sdi-si.*sd)./(sii-si.^2)).*si;
noise_estPSD(1:k/2)=alfa+beta.*index;
noise_estPSD(k/2+1:k)=noise_estPSD(k/2:-1:1)';
figure(10),subplot(221),plot(abs(noise_estPSD)),grid
ylabel('Noise_Est  Spectrum'),xlabel('Num_Samples')
sig_estPSD=sqrt(abs(A2.*conj(A2)-(noise_estPSD.^2)));
figure(10),subplot(222),plot(abs(sig_estPSD)),grid
ylabel('Signal_Est  Spectrum'),xlabel('Num_Samples')
fil_sig_PSD=((A2))./(1+((noise_estPSD)./(sig_estPSD+0.1)));
figure(10),subplot(223),plot(abs(fil_sig_PSD)),grid
ylabel('Filtered_Signal  Spectrum'),xlabel('Num_Samples')
signal2=ifft(fftshift(fil_sig_PSD));
figure(10),subplot(224),plot(t1,real(signal2)),grid
ylabel('Filtered_Signal'),xlabel('Time')
gtext('Weiner  Filtering  Y_Data'),pause

sd=sum(abs(A3(w3db_Est:k/2)));
sdi=sum(abs(A3(w3db_Est:k/2)).*abs(index(w3db_Est:k/2)'));
si=sum(abs(index(w3db_Est:k/2)));
sii=sum(abs(index(w3db_Est:k/2)).*abs(index(w3db_Est:k/2)));
beta=(sdi-si.*sd)./(sii-si.^2);
alfa=sd-((sdi-si.*sd)./(sii-si.^2)).*si;
noise_estPSD(1:k/2)=alfa+beta.*index;
noise_estPSD(k/2+1:k)=noise_estPSD(k/2:-1:1)';
figure(11),subplot(221),plot(abs(noise_estPSD)),grid
ylabel('Noise_Est  Spectrum'),xlabel('Num_Samples')
sig_estPSD=sqrt(abs(A3.*conj(A3)-(noise_estPSD.^2)));
figure(11),subplot(222),plot(abs(sig_estPSD)),grid
fil_sig_PSD=((A3))./(1+((noise_estPSD)./(sig_estPSD+0.1)));
figure(11),subplot(223),plot(abs(fil_sig_PSD)),grid
ylabel('Signal_Est  Spectrum'),xlabel('Num_Samples')
ylabel('Filtered_Signal  Spectrum'),xlabel('Num_Samples')
signal3=ifft(fftshift(fil_sig_PSD));
figure(11),subplot(224),plot(t1,real(signal3)),grid
ylabel('Filtered_Signal'),xlabel('Time')
gtext('Weiner  Filtering  Z_Data'),pause
%
% Replacing the dc bias back in to the signal
%
A1=fft(a1);A2=fft(a2);A3=fft(a3);
A1(1:1)=dc1;A2(1:1)=dc2;A3(1:1)=dc3;
a1=ifft(A1);a2=ifft(A2);a3=ifft(A3);
figure(12),subplot(311),plot(t1,real(a1))
title('Original X, Y, Z Data')
```

```
ylabel('X'),xlabel('Time')
figure(12),subplot(312),plot(t1,real(a2))
ylabel('Y'),xlabel('Time')
figure(12),subplot(313),plot(t1,real(a3))
ylabel('Z'),xlabel('Time')

A1=zeros(size([1:k]));A2=A1;A3=A2;
A1=fft(signal1);A2=fft(signal2);A3=fft(signal3);
A1(1:1)=dc1;A2(1:1)=dc2;A3(1:1)=dc3;
a1wf=ifft(A1);a2wf=ifft(A2);a3wf=ifft(A3);
figure(13),subplot(311),plot(t1,real(a1wf))
title('Weiner Filtered X, Y, Z Data')
ylabel('X'),xlabel('Time')
figure(13),subplot(312),plot(t1,real(a2wf))
ylabel('Y'),xlabel('Time')
figure(13),subplot(313),plot(t1,real(a3wf))
ylabel('Z'),xlabel('Time')
end;
```