LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note	**LIGO-T07XXXX-00-X**	September 27, 2007

Newtonian Noise Suppression by Adaptive Filtering

Elena Gasparri

Distribution of this document:

LIGO Scientific Collaboration

California Institute of Technology
LIGO Project, MS 18-34
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, Room NW17-161
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
Route 10, Mile Marker 2
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

http://www.ligo.caltech.edu/

Abstract

The next generation of gravitational wave observatories are being designed to detect space-time strains down to 10 Hz where there is an except abundance of astrophysical sources (like the massive black hole mergers and the gravitational stochastic background). At such low frequencies the dominant noise buffeting the suspended mirrors of the interferometers is likely to be the seismic noise dued to the motion of the ground and the fluctuating Newtonian gravity gradients arising from density fluctuations in the ground and in the surrounding air. A way to achieve noise suppression is to perform active noise cancellation using the adaptive filter technique. In this article is explained how a such adaptive filter is been implemented and finally are shown the results obtained testing this filter at the 40-meter prototype laboratory.

1 Introduction

LIGO (Laser Interferometer Gravitational-Wave Observatory) is a project that aims to detect gravitational waves from astronomical sources. Currently there are two LIGO sites, one in Hanford, Washington State, and one in Livingston, Louisiana.

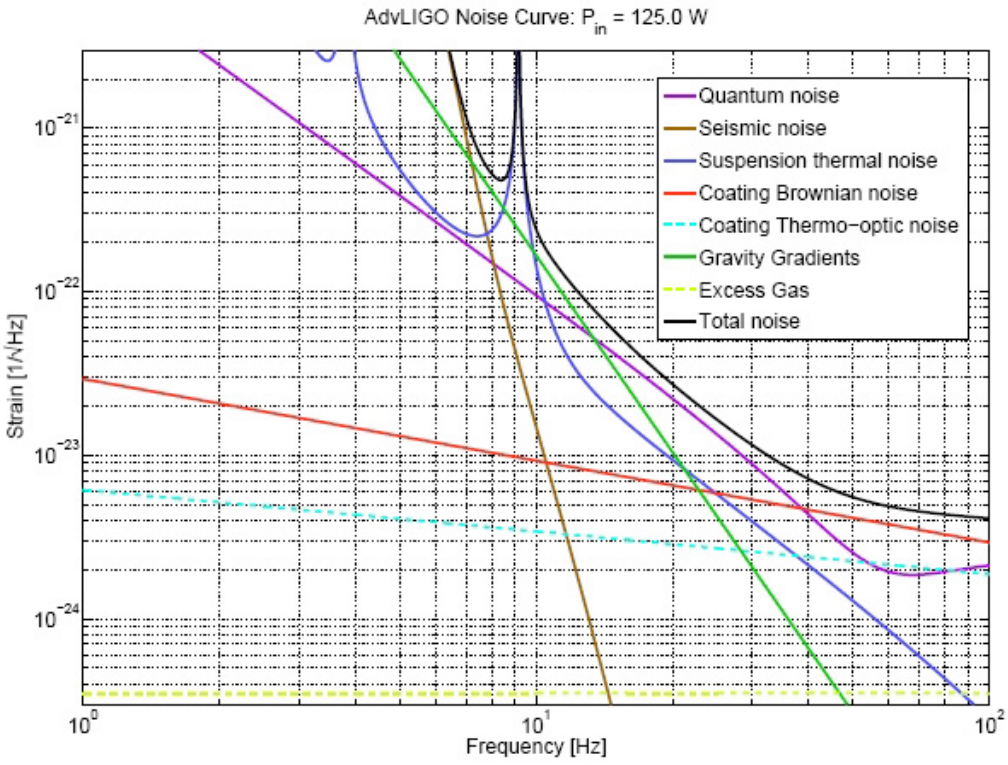

Figure 1: Amplitude spectra of the noise in Advanced LIGO.

As is predicted in General Relativity by Einstein in 1918 the gravitational radiation is a ripple in the curvature of space-time so when a gravitational wave passes through the interferometer, the space-time in the local area is altered. Depending on the source of the wave and its polarization, this results in an effective change in the length of one or both of the arms of the interferometer. Therefore the possibility to detect a gravitational wave is strictly related to the precision with which is possible to measure the difference of the length of the arms.
Gravitational waves from massive black hole mergers and stochastic background are expected to be detected at frequency below 10 Hz; in this frequency region the most important type of noise are the seismic noise, the gravity gradient noise and the suspension thermal noise (Fig.1) that act on the test masses altering their position. The seismic noise is due to the motion of the ground, the gravity gradient noise instead, is due to the fluctuation of the gravity force. This fluctuation can be caused by seismic vibration that alter the conformation of the ground but also by the fluctuation of the density of the earth itself. While it seems possible to isolate the test masses from the seismic vibration down to frequency as low as 3 Hz ([4]), it is not possible to achieve isolation from the Newtonian noise because the gravitational field can not be shielded. Thus, gravity gradients constitute an ultimate low-frequency noise source.

2 Active Noise Cancellation and Adaptive Filtering

As was explained in section 1 we can not obtain a reduction of Newtonian noise by increasing the isolation of the system; the only way to achieve suppression from the gravity gradient noise is the *active noise cancellation*.

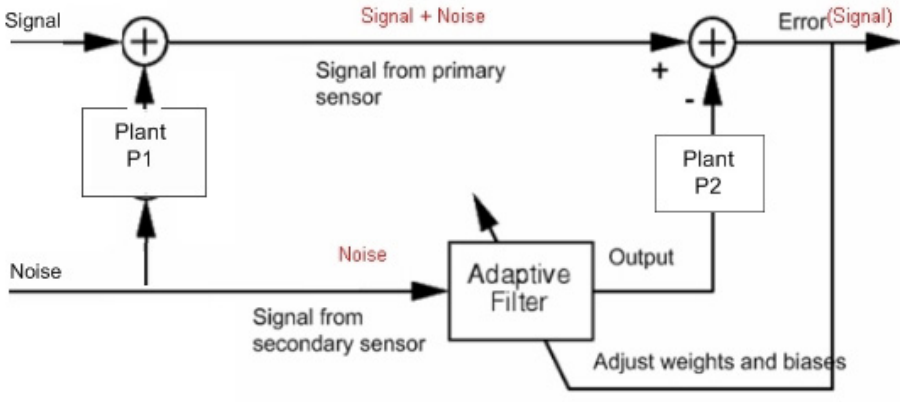

Figure 2: Active noise cancellation scheme.

In Fig.2 is illustrated the basic scheme for an active-canceller; the circuit receives two input: the

primary input is the signal that we want to measure plus the noise that contaminate the signal, the secondary input (Reference input) is a signal from an independent measure of the noise. Thus the reference input is uncorrelated with the signal but correlated in some unknown way with the noise in the primary input. The reference input is filtered to produce an output that is a close replica of the noise in the primary input. Then subtracting this output from the primary input we perform the noise cancellation.
Because we do not know the correlation between the noise in the primary input and the reference input we can not use a fixed filter, we need an adaptive filter to produce the right output.
An *Adaptive Filter* is a filter that automatically adjusts its own impulse response through a recursive algorithm that respond to an error signal dependent on the filter's output. Thus with this kind of algorithm, the filter can operate under changing conditions and can readjust itself continuously to minimize the error signal.

2.1 Least-Mean Square Algorithm

The least-mean square (LMS) algorithm is a linear adaptive filtering algorithm that consist of two basic processes: the first one is a *filtering process*, which involves computing the output of a real, FIR filter ($\hat{\mathbf{d}}(n|x_n)$) produced by a set of tap inputs (x(n), ..., x(n-M+1)), and generating an estimation error (e(n)) by comparing this output to a desired response (d(n)); the second one is an *adaptive process*, which involves the automatic adjustment of the tap weights of the filter ($\hat{w}_i(n)$) in accordance with the estimation error ([3]).
The details of the FIR filter are shown in Fig.3. The tap input *x(n), x(n-1), ..., x(n-M+1)* form the elements of the M-by-1 *tap-input-vector* ***x****(n)*, where M-1 is the number of delay elements (the order of the filter); correspondingly the tap weights $\hat{w}_0(n)$, $\hat{w}_1(n)$, ..., $\hat{w}_{M-1}(n)$ form the elements of the M-by-1 tap-weight vector $\hat{\mathbf{w}}(n)$.
The LMS algorithm for the adaption is:

1. *Filter Output*: $y(n) = \hat{\mathbf{w}}^T(n)\mathbf{x}(n)$
2. *Estimation error*: $e(n) = d(n) - y(n)$
3. *Tap-weight adaptation*: $\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{x}(n)e(n)$

The point 1 and 2 define the estimation error $e(n)$, the computation of which is based on the current estimate of the tap-weight vector $\mathbf{w}\hat{(n)}$. The formula in the third point allow us to calculate the successive value of the weight vector in a recursive way. The second term in the formula represents the correction that is applied to the current estimate of the weights and it originates from the calculus of the gradient of the mean-squared error produced by the LMS algorithm in the

estimate of the primary input.

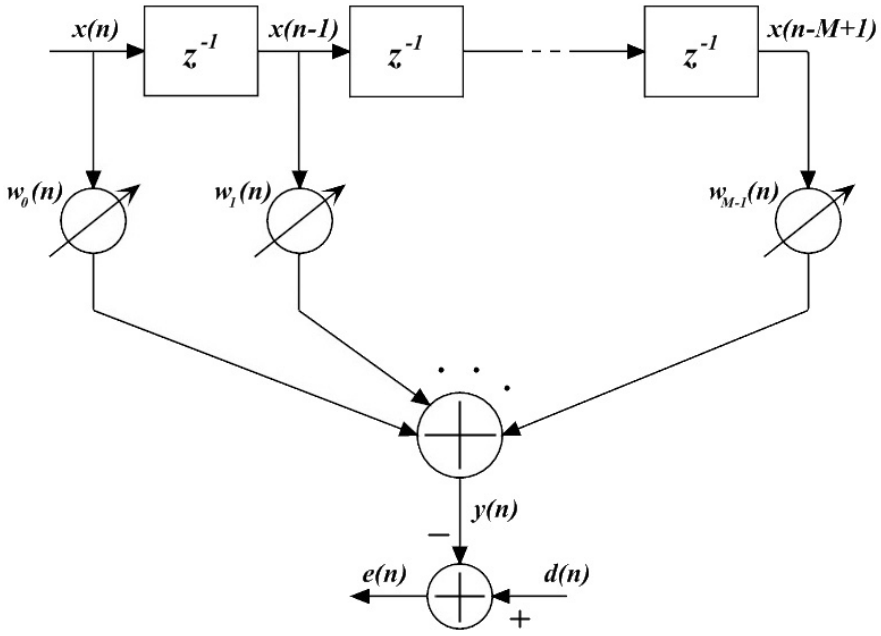

Figure 3: Transversal filter detail.

To use this iterative algorithm we only need to set an initial values for the tap-weight input vector ($\hat{\mathbf{w}}(0)$), a value for μ that is the $step-size$ parameter and the number of taps M. If the initial value of $\hat{\mathbf{w}}$ ($\hat{\mathbf{w}}(0)$) is unknown (as it is in the most of the case) it could be set to zero; to set the value of the step-size parameter, μ, there is a formula in the literature that define his low and his upper limit:

$$0 < \mu < \frac{2}{Tap - Input\ Power} \tag{1}$$

where the tap-input power is the sum of the mean-square values of all the tap inputs in the transversal filter. The convergence of the algorithm is assured by choosing μ in this interval. If a small value of μ is chosen, the adaption is slow, which is equivalent to the LMS algorithm having a long "memory", on the other hand, if μ is large, the adaption is relatively fast, but at the expense of an increase in the average excess mean-square error after adaption. Thus the reciprocal of the parameter μ may be viewed as the memory of the LMS algorithm.

3 Application

In order to check the feasibility of the adaptive filter to active cancel the noise, it is been decided to test the filter at 40-meter prototype laboratory.
The goal is to achieve seismic noise reduction on the position signal of the suspension using the signals from six accelerometers. These accelerometers are divided in two block: one is disposed onto a support of the ITMX chamber and the other one onto a support of the beam splitter chamber. Each accelerometer block supplies three signal for x, y, z direction, thus actually we can use these six signal for reference input and build a multiple input single output LMS adaptive filter to achieve the best noise reduction.

3.1 Implementation of the filter with MATLAB

MATLAB has several tool to build the adaptive filter like the *Filter Designed Toolbox* or the *Neural Network Toolbox*, otherwise is possible to make the filter just implement the algorithm in MATLAB. Initially I focused on the Neural Network toolbox but then I decided to simply implement the LMS algorithm (always using MATLAB).

3.1.1 The SISO LMS adaptive filter

To start I made the plot of coherences between the position signal of the suspension and the signals from the accelerometers.
From the Fig.4 we can see that we have a good coherence in the region of frequency below 10 Hz for both the x-axis of the east and west accelerometer. So the thing that we can deduce is that probably we will obtain a good noise cancellation in this range of frequency.
Initially I developed a program for a single-input single-output LMS adaptive filter. The primary input ($dpos$) is the signal for the position of the suspension (PRM), the reference input ($daccx$) is the signal from the x-axis of the east accelerometer. The program uses a modify LMS algorithm called *the sign algorithm* that is useful for increase the performance of the program in "quality of speed".
The difference between the standard LMS algorithm and this is in the way of calculate the new value of the weights vector: the new algorithm replaces the error with only his sign and put an average value (equal 2, like an error step) in the place of the absolute value of the error.

$$\hat{w}(n+1) = \hat{w}(n) + 2 * \mu * sign(e(n)) * \mathbf{u}(n) \tag{2}$$

In this way we can avoid the possibility of a divergence of the algorithm when the reference input and the primary input have the amplitude of the signal very different (as it is in our case).
The program is in section 5.1 of this report. In the first part I simply make some operation on the suspension signal and on the accelerometer signal: first I resample the signals at the same sample rate (32 Hz), than I subtract from the signals their mean so that they are all with zero empirical mean. Now, with these signals, is possible start to work.
The next step consist on set the parameters of the algorithm such as the step-size μ and the filter order M (that is equal to the number of delay). These two parameters are the most important values of the algorithm and from their depends the goodness of the filter as it is been explained in section section 2.1. In order to find the correct value for μ and M, I made some plot of the response of the filter changing the value of these parameters. Notice that the values of μ for which we can have the convergence of the algorithm are in a particular range that can be calculated using the formula (1): $0 < \mu < 0.0165$.

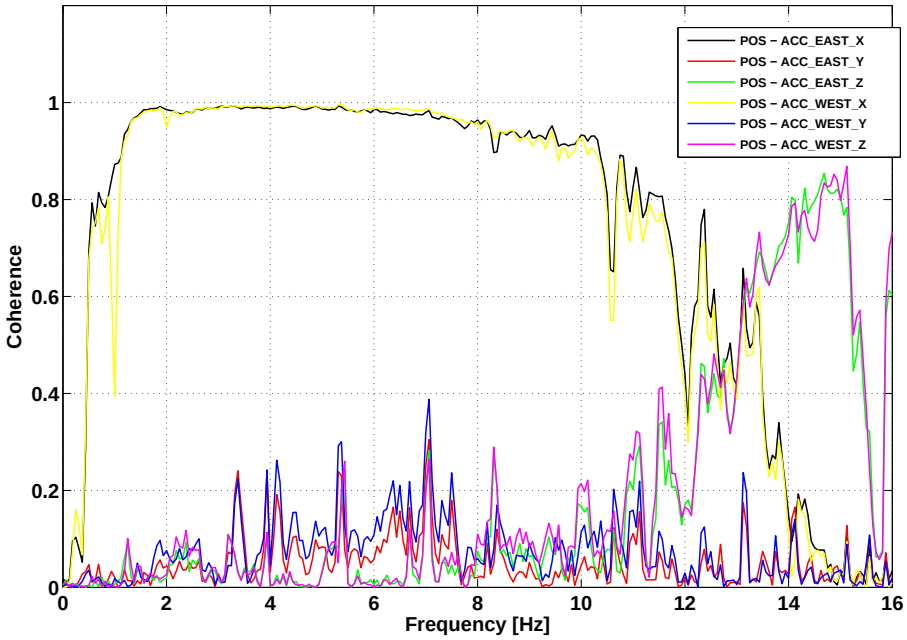

Figure 4: Coherence between the position signal and the accelerometers signals

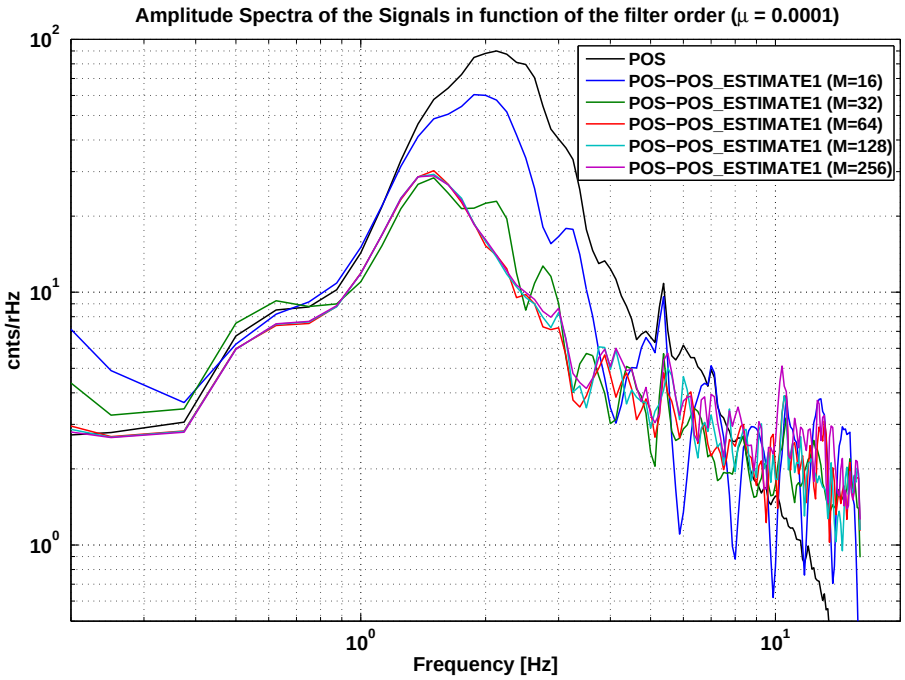

Figure 5: Plot of the amplitude of the spectrum in function of the filter order M for μ equal 0.0001.

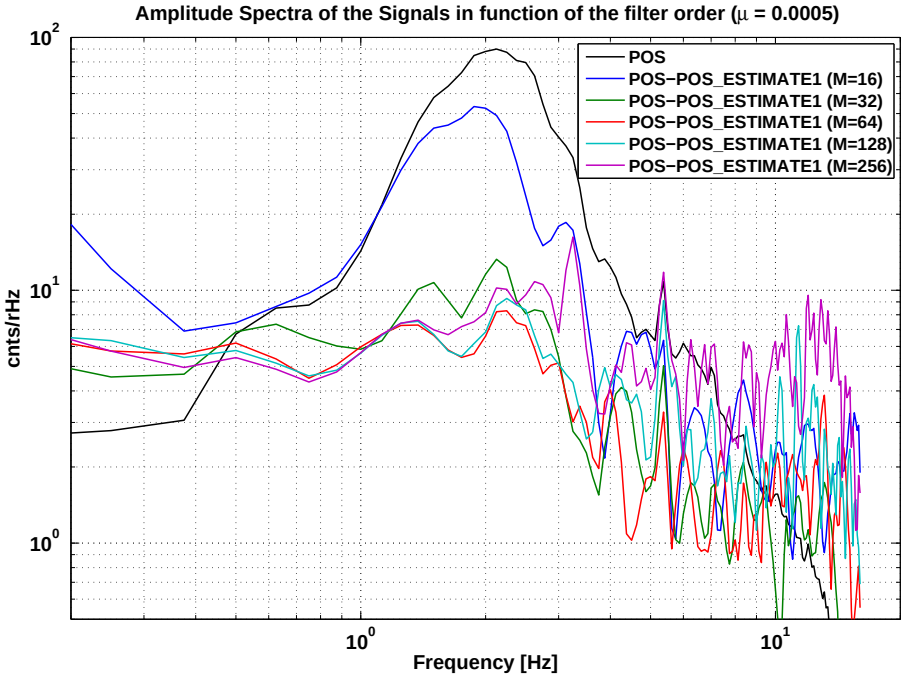

Figure 6: Plot of the amplitude of the spectrum in function of the filter order M for μ equal 0.0005.

Observing the plots in figure 5, 6 and 7, it is possible to see that with the value of μ = 0.0005 and M = 64 we obtain the best noise suppression (with a SISO LMS algorithm).
It is knew that the LMS adaptive filter converges to the Optimal Wiener Filter so, in order to check how good is the noise suppression on the position signal we can compare the amplitude spectra of the position signal filtered with the optimal wiener filter with the amplitude spectra filtered with the SISO LMS adaptive filter. From Fig.8 is possible to see that at low frequency the SISO LMS filter achieves the same noise reduction of the optimal wiener filter; at 10 Hz frequency the SISO LMS filter is not good as the optimal wiener filter. The main reason of this lower performance is that the LMS filter is an adaptive filter so maybe it needs a longer sequence of data to adapt his weights (the curve in figure 8 is been obtained with one hour of data for the adaption); in other words the adaption process is not finished so the values of the $M - matrix$ that we have are not the best.
However the results are interesting because the importance of this filter is that first of all it is an adaptive filter so it does not need to know all the signal or to calculate the correlation matrix: it is able to adjust the filter with only the knowledge of the previous values of the signal and his calculus is proportional to M, and not to M^2 like in the case of the optimal filter, so it is faster.

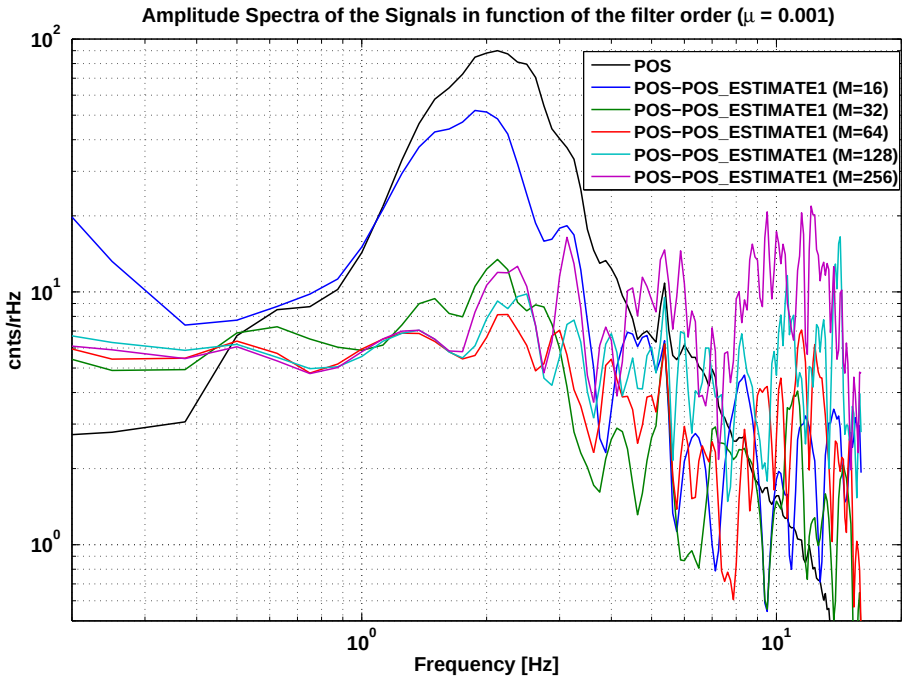

Figure 7: Plot of the amplitude of the spectrum in function of the filter order M for μ equal 0.001.

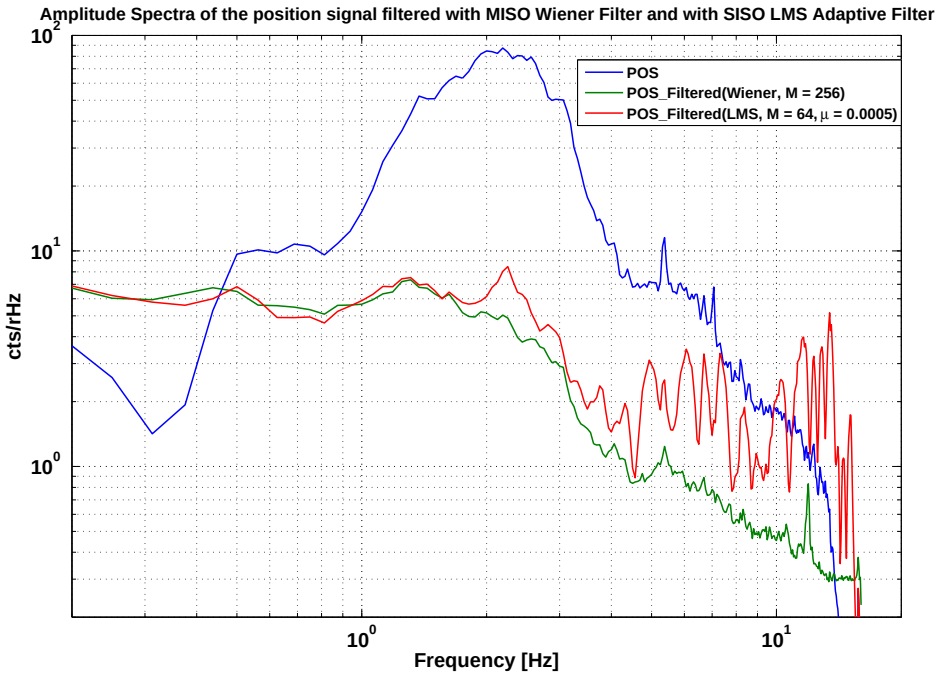

Figure 8: Comparison of the results obtained on filtering the position signal with an optimal MISO wiener filter of order 256 and with an LMS SISO adaptive filter of order 64.

3.1.2 MISO LMS adaptive filter

The next step of the project is to develop a filter the receives in input all the accelerometer signals instead of only one: so the goal is to build a MISO LMS adaptive filter.
This type of filter is not well known: the idea is to adapt a different weight array for each reference input and to use as error signal for the adaption of the weights, the difference between the position signal and the total output (that is the sum of all the single output Y_i (Fig.9)).

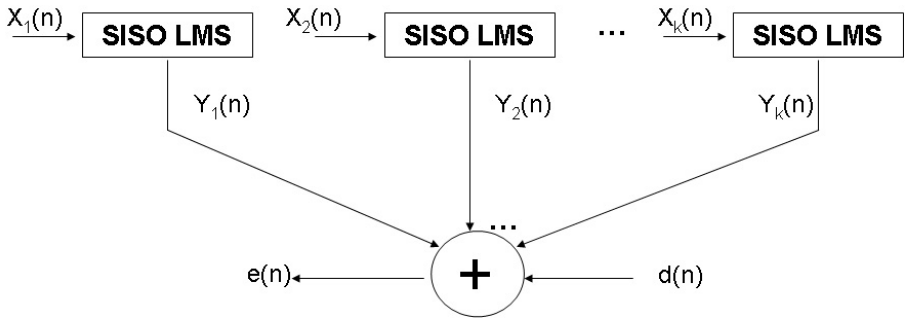

Figure 9: MISO LMS adaptive filter scheme.

To choose the best value for the step-size parameter (μ) and the filter order (M), I made three different plot for three different value of μ and, for a fixed μ, I varied the value of M (Fig.10, 11 and 12).

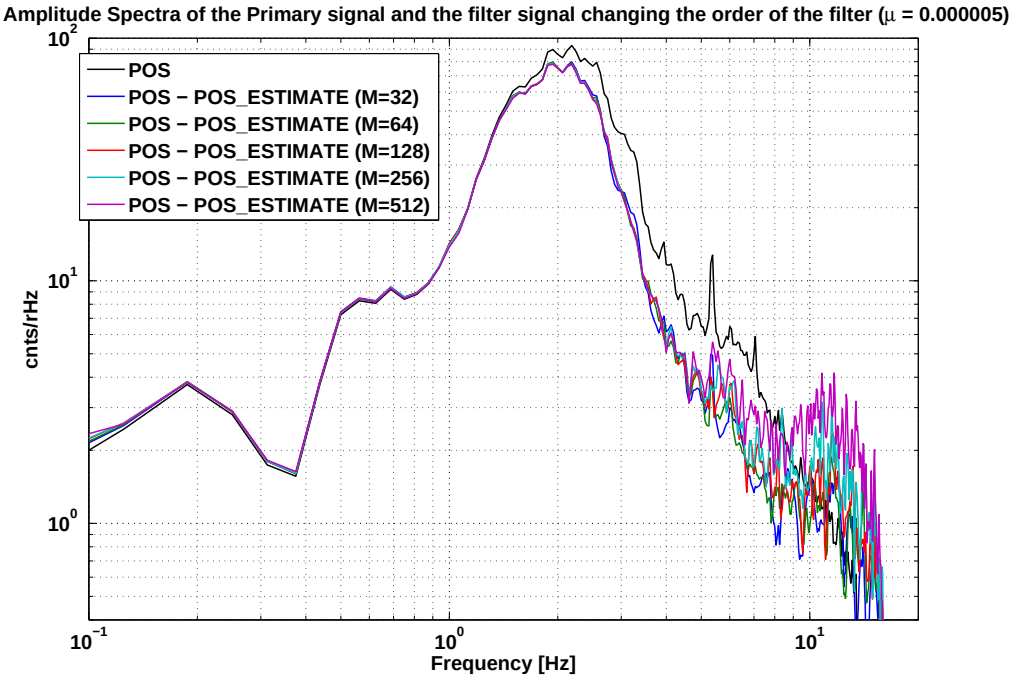

Figure 10: Plot of the amplitude of the spectrum in function of the MISO filter order M, for $\mu = 0.000005$.

The program for the MISO LMS adaptive filter is shown in section 5.2. The results are worse respect to the SISO LMS adaptive filter but, it is to note that for the adaption of this MISO filter is been used the same sequence of data (one hour data). In figure 12 (in particular) it is clear that the MISO algorithm needs a longer sequence of data to adapt his weights, infact the spectra at high frequency has a strange behavior that is characteristic of a not finished adaption process.

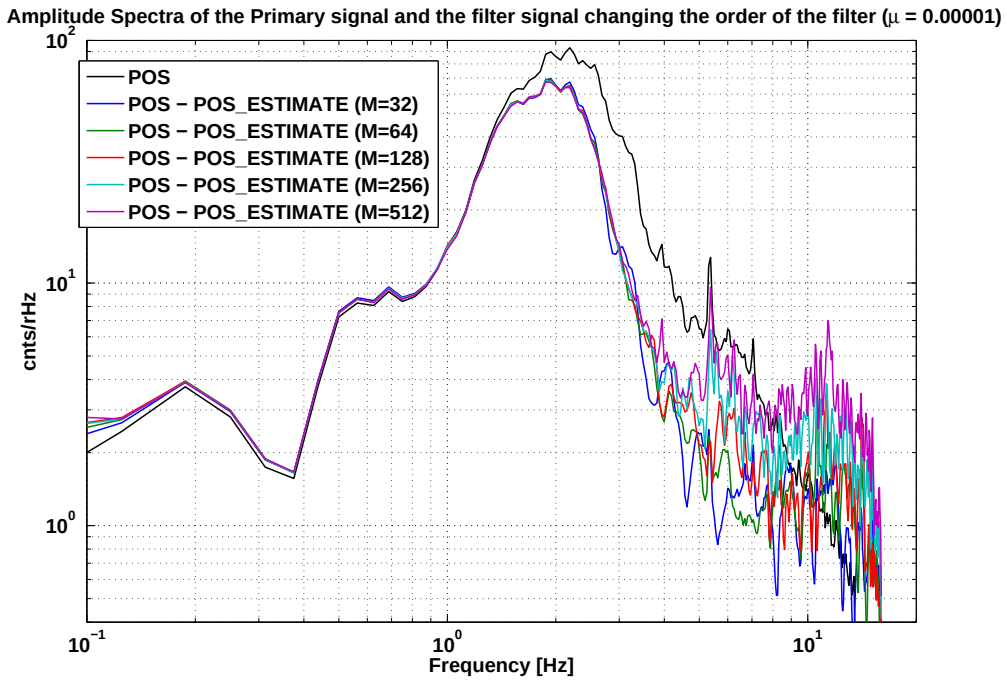

Figure 11: Plot of the amplitude of the spectrum in function of the MISO filter order M, for $\mu = 0.00001$.

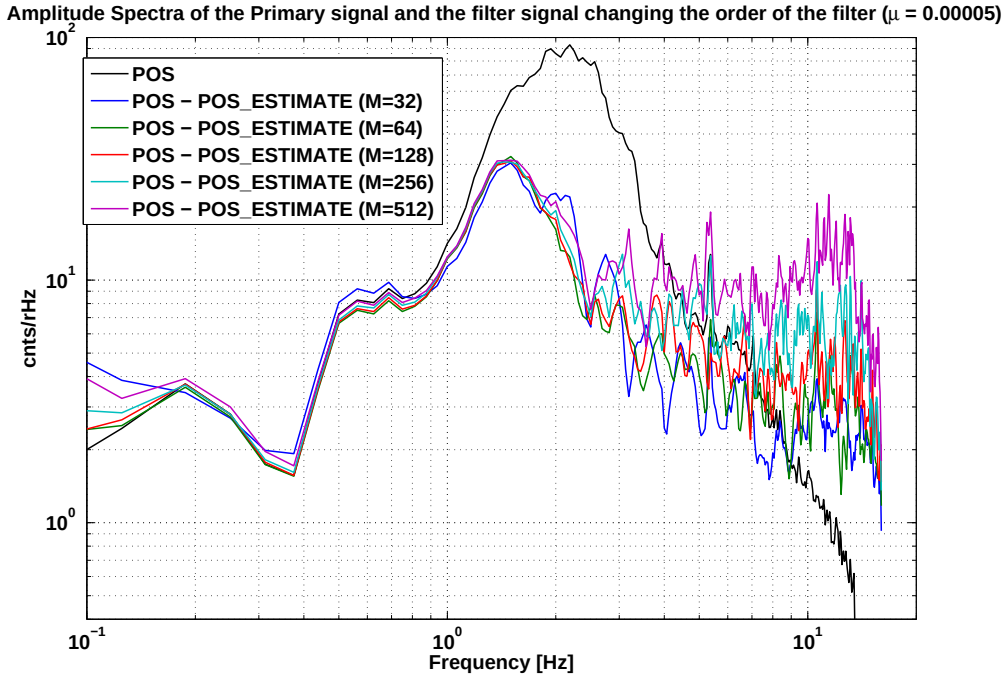

Figure 12: Plot of the amplitude of the spectrum in function of the MISO filter order M, for $\mu = 0.00005$.

3.1.3 Neural Network Toolbox

The Neural Network Toolbox is a Matlab tool that can be used to build an Adaptive Filter.
The Neural Network are composed of simple elements that operating in parallel; the network function is determined by the connections between elements (weights) and the ability of the network relies in the fact that by a process of adaption, call learning, we can set the network to perform basically any function. Commonly neural network are adjusted, or trained, so that a particular input leads to a specific target output: in our particular case, in order to perform active noise cancellation, it suffices to combine a linear neural network with a tapped delay line (Fig.13).

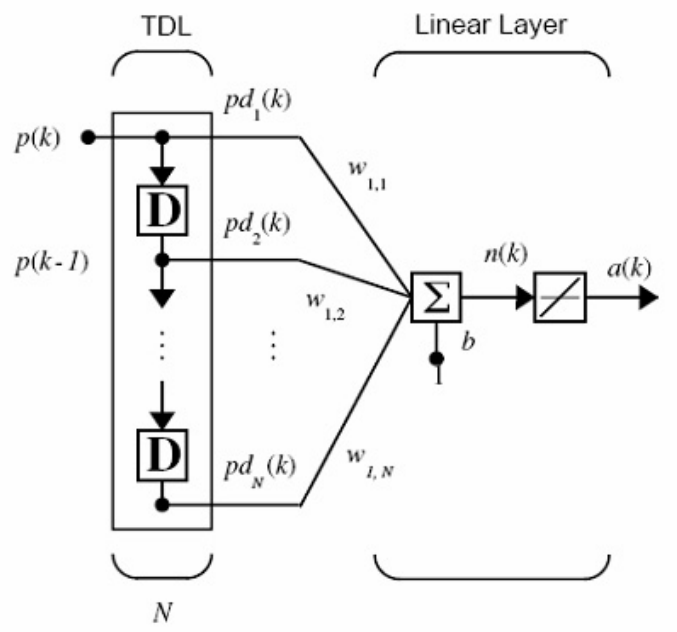

Figure 13: Adaptive filter with Neural Network.

A Neural Network is defined as *linear* when the transfer function of the system (that convert the neuron in the output) itself is linear (this kind of transfer function are called *purelin* in Matlab); a *tapped delay line* (TDL) is an element that produces at the output an N-dimensional vector, **pd**(k), made up of the input signal at the current time, the previous input signal, etc. So the output of an adaptive linear neural network is defined as:

$$a(k) = n(k) = purelin(\mathbf{W}\mathbf{p} + b) = \sum_{i=1}^{N} w_{1,i} p(k-i+1) + b \tag{3}$$

where a(k) is the output of the network, that is equal to the neuron n(k), that is the multiplication of the weights vector for the output of the TDL.
Before use the filter, the values of the weights have to be adjusted: the neural network toolbox has

a function called *adapt*, that makes this automatically. This function use the LMS algorithm to recursive change the values of $\mathbf{W}_i(k)$ by comparing the output of the filter to the desired output, in the standard way that it is been explained in section 2.1.
However, when finally I tried to make a such adaptive filter with this toolbox I encountered some problem with the adaption. Actually the fact that the matlab has this function already built inside does not allow to customize it a lot: this is the reason for which I decide to implement myself the LMS algorithm for the adaption.

4 Summary and Further Work

The SISO LMS adaptive filter and the MISO LMS adaptive filter seems to be promising for active cancel the noise on the test masses of the interferometer: these filter are fast and simple to implement.
To improve their performance we can try to make the adaption process with a longer sequence of data; moreover, for the MISO LMS Adaptive Filter, there are other algorithm that can be implemented like the *Multiple Error LMS Algorithm* ([5], [6]).

5 MATLAB Program

5.1 SISO LMS adaptive filter program

```
tic
clear all; close all;

load bsdat

accex = bs(1).data;
accey = bs(2).data;
accez = bs(3).data;
accwx = bs(4).data;
accwy = bs(5).data;
accwz = bs(6).data;

pos = bs(7).data;
```

```
srate = 32; % New sample rate
R = bs(1).rate/srate;  % Downsample to 64 Hz (Nyquist 32 Hz)

disp('Downsampling...')
daccx = decimate(accex, R);
daccy = decimate(accey, R);
daccz = decimate(accez, R);

dpos = decimate(bs(7).data,bs(7).rate/srate);

% Zero empirical mean.
daccx = detrend(daccx);  % Reference input
daccy = detrend(daccy);  % Reference input
daccz = detrend(daccz);  % Reference input
dpos = detrend(dpos);    % Primary input

% Parameter
mu = 0.005;              % Step - size
N = length(daccx);       % Length of the Primary and Reference signals
M = 64;                  % Size of the weights array (Filter order)

disp('Initialization...')
W = zeros(M,1);          % Weights array
Y = zeros(N,1);          % Position estimate (filter output)
E = zeros(N,1);          % Position - Position estimate (filter error or
                         % primary signal filtered)

X = zeros(N+M-1,1);      % Array to make the delay on the reference input
X(M:N+M-1) = daccx(1:N);

disp('Calculating the weights array...')
for n = M : N+M-1
   Y(n-M+1) = W'*flipud(X(n-M+1:n));
   E(n-M+1) = dpos(n-M+1) - Y(n-M+1);
   W = W + 2*mu*sign(E(n-M+1))*flipud(X(n-M+1:n));
end
```

```
disp('Filtering...')
Out=zeros(N,1);
for n = M : N
   Out(n) = W'*flipud(daccx(n-M+1:n));
   E1(n) = dpos(n) - Out(n);
end

% Spectra of the Signals
nfft = 16*srate;
% Spectra of the primary input
[pdpos,f] = pwelch(dpos, hanning(nfft), nfft/2, nfft, srate);
% Spectra of the filtered prymary input
[pE1,f] = pwelch(E1,hanning(nfft), nfft/2, nfft, srate);
toc
```

5.2 MISO LMS adaptive filter program

```
tic
clear all; close all;

load bsdat

accex = bs(1).data;
accey = bs(2).data;
accez = bs(3).data;
accwx = bs(4).data;
accwy = bs(5).data;
accwz = bs(6).data;

pos = bs(7).data;

srate = 32; % New sample rate
R = bs(1).rate/srate;  % Downsample to 64 Hz (Nyquist 32 Hz)

disp('Downsampling...')
daccex = decimate(accex, R);
daccey = decimate(accey, R);
```

```
daccez = decimate(accez, R);
daccwx = decimate(accwx, R);
daccwy = decimate(accwy, R);
daccwz = decimate(accwz, R);

dpos = decimate(bs(7).data,bs(7).rate/srate);

% Zero empirical mean.
daccex = detrend(daccex);  % Reference input
daccey = detrend(daccey);  % Reference input
daccez = detrend(daccez);  % Reference input
daccwx = detrend(daccwx);  % Reference input
daccwy = detrend(daccwy);  % Reference input
daccwz = detrend(daccwz);  % Reference input
dpos = detrend(dpos);    % Primary input

% Reference input matrix
Ref = [daccex daccey daccez daccwx daccwy daccwz];
% Parameter
[N, I] = size(Ref);    % N is the Length of the Primary and Reference ...
                         %signals and I is the number of ref. signals
mu = 0.00001;          % Step - size
M = 32;                % Size of the weights array (Filter order)

disp('Initialization...')
E = zeros(N,1);         % Error signal for the adaption
                        % primary signal filtered)
W = zeros(M,I);         % Weights array
Y = zeros(N,1);         % Position estimate (filter output for adaption)
Out=zeros(N,1);         % Final output of the filter
E1=zeros(N,1);          % Error or position signal filtered
X = zeros(N+M-1,I);     % Array to make the delay on the reference input
X(M:N+M-1,:) = Ref;

disp('Adaption of weights array...')
for n = M : N+M-1
    for i = 1:I
        Y(n-M+1) = Y(n-M+1)+(W(:,i))'*flipud(X(n-M+1:n,i));
```

```
    end
    E(n-M+1) = dpos(n-M+1) - Y(n-M+1);
    W = W + 2*mu*sign(E(n-M+1))*flipud(X(n-M+1:n,:));
end

disp('Filtering...')
for n = M : N
    for i = 1:I
        Out(n) = Out(n) + (W(:,i))'*flipud(Ref(n-M+1:n,i));
    end
    E1(n) = dpos(n) - Out(n); %Position signal filtered
end
toc
```

References

[1] Y. Huang, J. Benesty and J. Chen, *Acoustic MIMO Signal Processing*.

[2] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*.

[3] S. Haykin, *Adaptive Filter Theory*.

[4] S. A. Hughes and K. S. Thorne, Physical Review D 58 (1998) 1-27.

[5] S. C. Douglas, *"Analysis of the Multiple-Error and Block Least-Mean-Square Adaptive Algorithms"*, IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing, vol. 42, no. 2, pp. 92-101, February 1995. [engr.smu.edu/ douglas/conference/ICASSP94paper.ps]

[6] S. G. Edwards, *"Disturbance Identification and Rejection Experiments On an Ultra Quiet Platform"*, Paper AAS 99-342, Proceedings of AAS/AIAA Astrodynamics Specialist Conference, Girdwood, AK, 16-19 August 1999. [http://www.ligo.caltech.edu/ rana/docs/UQP-MIL.pdf]