Technical Note **LIGO-T1200329–v3** 2012/09/25

Machine Learning Techniques for Seismic Noise Classification and Interferometer Control Systems

Maria Okounkova — Mentors: Denis Martynov, Jenne Driggers, Rana Adhikari

1 Abstrat

One of the most significant low-frequency noise sources in the Laser Interferometer Gravitational Wave Observatory (LIGO) experiment is seismic noise. While 40 Meter Prototype Lab has noise cancellation filters in place for constant seismic noise, transient seismic disturbances, such as earthquakes, vehicles, and other unforeseen events, which generate noise in the interferometer signals, require adaptive filtering to be removed. This project involved gathering training data of seismic disturbances from known sources at the 40 Meter Lab and applying various machine learning algorithms for online classification, so that a noise source can be identified in real time, and a specific filter for the noise source may be applied.

Machine learning techniques have other potential applications to the LIGO project as well, including applications to interferometer control systems. Thus, I experimented with using recurrent neural networks as adaptive control systems in simulations of noisy dynamic systems in order to assess the feasibility of training a neural network to internally approximate their dynamics. With rigorous training, such a system could potentially be implemented in the 40 Meter Lab to control the interferometer test masses.

2 Motivation and Problem

One of the main current challenges to the LIGO (Laser Interferometer Gravitational Wave Observatory) project is seismic disturbance, which adds noise to the interferometer signals by moving the test masses in the optical cavity. These noisy signals, then, make the task of detecting gravitational waves highly difficult. The goal of this project is to use machine learning techniques to filter seismic noise in Gravitational Wave Detectors.

Specifically, seismic noise falls into the category of displacement noises, which can cause a differential change in the lengths of the Michelson Interferometer arms. By moving the test mass mirrors, seismic disturbances show up as strain noise in the interferometer signals. However, there is a distinction between stationary seismic noise, such as self-inflicted interferometer noise above 10 Hz due to HVAC systems and ground noise from the ground motion of the earth, and transient events such as earthquakes, trucks, cars, and other unpredictable seismic disturbances [**?**, p. 42].

The goal of this project is to create an online adaptive filtering system to filter the latter noise. While the LIGO 40m lab project currently has an Online Adaptive Filtering (OAF) system in place, using the Normalized Filtered-x Least Mean Squares algorithm, which updates its transfer function at each time step in order to minimize the squared error between the desired signal and the estimated, filtered signal, it is not fast enough to be able to filter out transient seismic disturbances online. This project aims to solve this problem through the use of machine learning algorithms to classify various types of incoming seismic noise in the time domain, in order to quickly determine the form of the filter which should be applied. For example, when the algorithm senses a seismic disturbance from a large truck near one of the LIGO buildings, it will signal to the control system to switch to the "truck" filter, transitioning back to Online Adaptive Filtering once the disturbance has passed.

Machine learning algorithms focused on classification of data range from the Perceptron

Learning Algorithm, to more sophisticated classification techniques, such as Radial Basis Functions, Kernel Methods, and Support Vector Machines [?, Lecture 1]. An input vector to a learning machine consists of a finite number of characteristics of the input. In supervised learning, or learning from data (as in our case), each training input vector is paired with a desired output vector. Classification of input vectors in a multi-dimensional space is possible as long as the data is linearly separable in some space if a non-linear transform is applied. When the data is a section of a time-series, as in our case, there are still machine learning algorithms which can process the entire data set, or, more cost-effectively, a decimation of the data set as an input vector [2, p. 159] Seismic disturbance should be linearly separable due to the distinctive bumps of the power spectral density of various seismic noise sources.

3 Classification of Seismic Noise - Data Aquisition

In order to develop or implement an algorithm to classify various sorts of seismic noise, one first has to decide what flavor of learning to implement - supervised or unsupervised. Generally, supervised learning occurs through the use of a training dataset and a testing dataset which already have input - desirable output pairs, and is much easier to implement in practice. Thus, my first step in the project was to gather a dataset of known seismic disturbances at the 40 Meter lab composed of { known input, known output } pairs. I managed to acquire enough data of 4 different classes to begin training neural networks for classification. Specifically, the classes are Local Earthquake (a magnitude 4 or greater earthquake within a 50 km radius of the 40m lab), Distant Earthquake (a magnitude 6 or greater earthquake outside of a 200 km radius of the 40m lab), Truck, and Quiet (in other words, ordinary seismic disturbances). I obtained the Earthquake data from the United States Geological Survey [1] and the truck data by both clocking trucks myself, and my inspecting the seismometer spectra at 9:00 - 10:00 am, when delivery trucks usually come to the neighboring labs. Typical time series of the 4 types of data are shown in figure 1:

Note the earthquake at 250 seconds in plot B, and the trucks at 200, 500, and 800 seconds in plot C. For any { input, output } pair, then, the output is a class vector of whatever form is required by the algorithm, and the input is the entire time signal. While certain features of the input signal may be picked out before sending the signal to the algorithm ahead of time, sending the entire signal as input allows the algorithm itself to decide on these features, thus supplying it with maximal information for both learning and subsequent classification. In order for the algorithm to be fast online, however, a decimated version of the signal can be used, provided that the signal retains its features. I found that a seismic disturbance retains its features to a decimation of up to 256, and used decimation by a factor of 8 in practice, lowering the sampling rate of the signal from $2048Hz$ to $256Hz$. The final dataset was comprised of three sorts of seismic disturbance - local earthquake, quiet, and truck. It was difficult to distinguish distant earthquake from seismometer tilt, since both occur at very low frequencies, and thus, for the time being, this data was excluded from training. The dataset was comprised of 224 signals, each 900 seconds in length, with roughly equal numbers of local earthquake, truck, and quiet signals. $\frac{2}{3}$rds of this data was used for training, and the final $\frac{1}{3}$ for testing and validation.

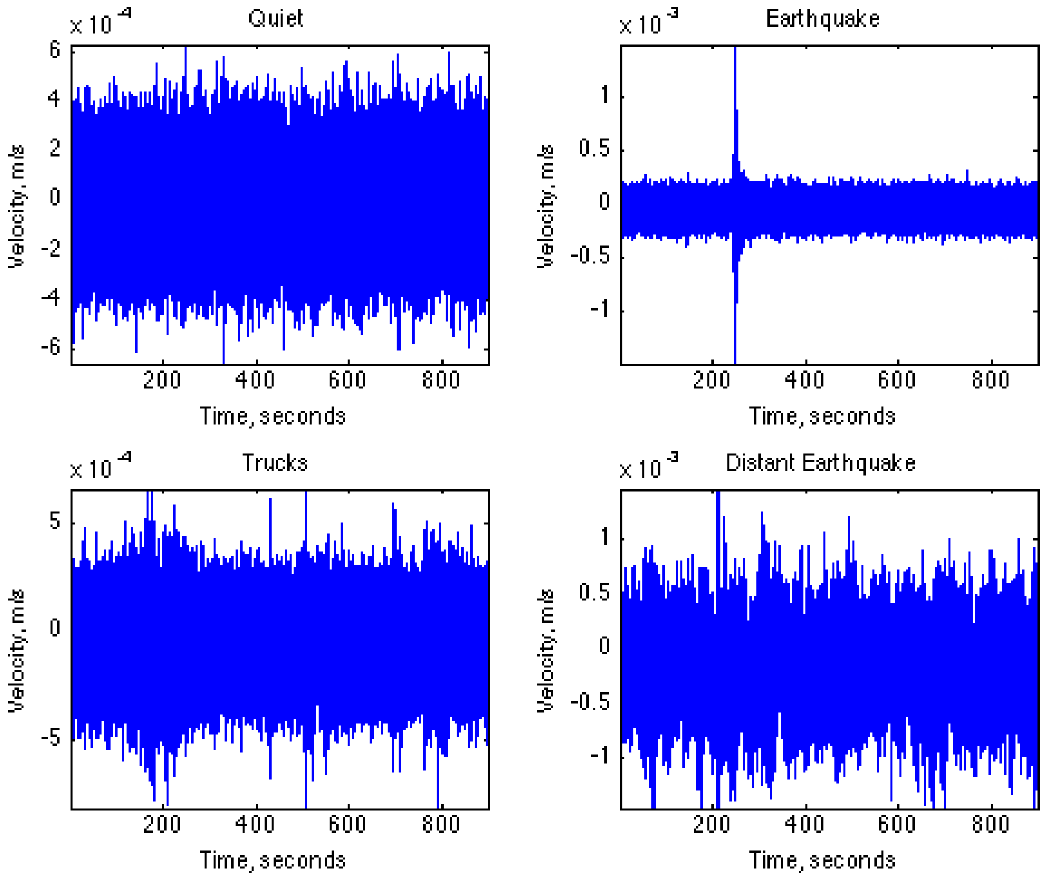

Figure 1: Time Series Spectra of Various Seismic Noise Sources at the 40 Meter

4 Neural Networks

After experimenting with various machine learning algorithms, starting from the basic perception to k-means clustering, I found that neural networks offered the most promising results. neural networks, at the core, are a means of function approximation using non-linear functions, and thus can be used to learn and approximate systems with complicated, highly-non linear dynamics. Thus, in practice, neural networks span most function spaces, and allow for system identification, modeling, and classification. Likewise, various flavors of neural networks exists, including feed-forward, time-delayed input, and recurrent neural networks. The basic architecture for a neural network is shown in figure 2.

Essentially, a neural network is comprised of an input layer of neurons, an output layer of neurons, and a number of hidden layers. For each neuron, the output of the previous layer is multiplied by a vector of weights and then passed through an activation function, which may be linear or non-linear in nature, as in equation[1]. In practice, most activation functions are sigmoidal and bear the form of equation[2].

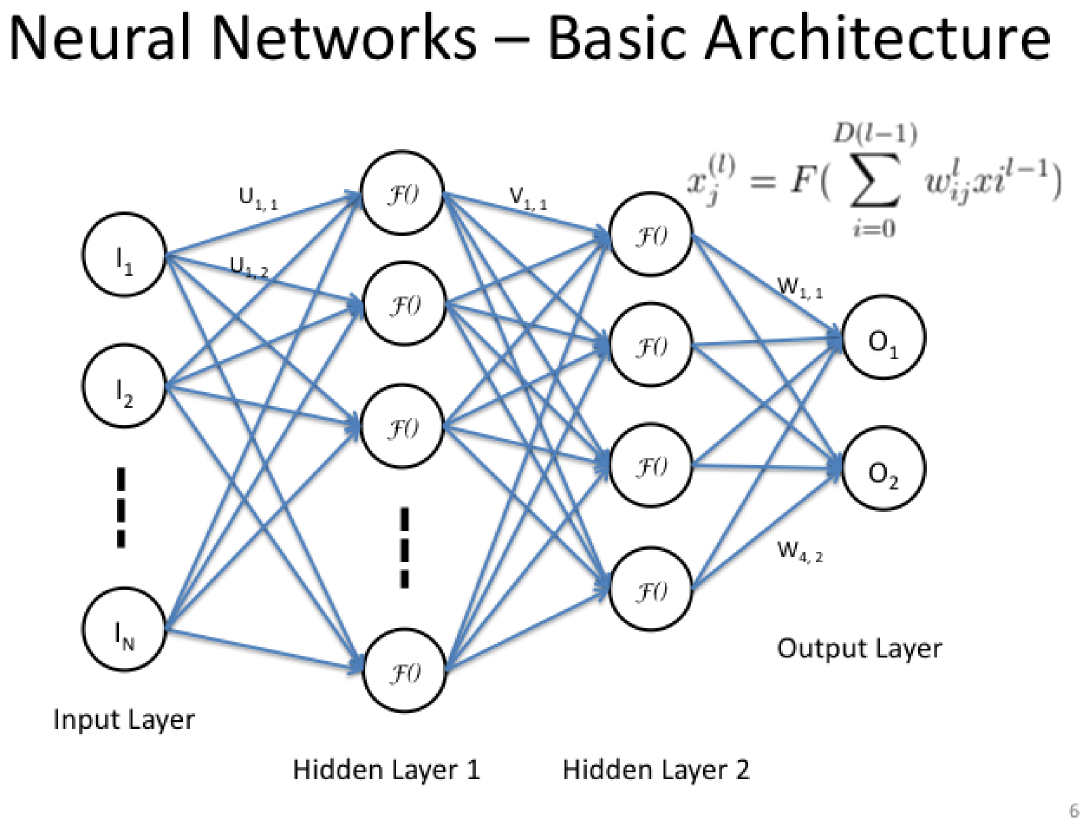

Figure 2: Neural Networks - Basic Architecture

$$x_j^{(l)} = F(\sum_{i=0}^{D(i-1)} w_{ij}^l x_i^{l-1}) \tag{1}$$

$$F(u) = \frac{1}{1+e^{-u}} \tag{2}$$

Using this algorithm, an output layer of neurons is generated from an input layer of neurons. Now, a neural network learns a certain function through adjusting its weight vectors. Most commonly, this is performed through back-propagation, where the gradient of some error function is calculated with respect to each weight, and the weights are adjusted accordingly. Most commonly, the error function during supervised learning takes the form of equation[3], where the error is the squared difference between the desired output and the output of the neural network, for each neuron.

$$e = (O_{desired} - O_{NN})^2 \tag{3}$$

In order to calculate the gradient of this error with respect of each weight as in equation[4], we use the chain rule as in equation[5], so that we only have to calculate the partial derivative at the output layer, and can then use the relationships between the layers themselves to back propagate this error throughout the neural network.

$$\nabla e(w) = \frac{\partial e(w)}{\partial w_{ij}^{l}} \forall i, j, l \tag{4}$$

$$\frac{\partial e(w)}{\partial w_{ij}^{l}} = \frac{\partial e(w)}{\partial s_{j}^{l}} \times \frac{\partial s_{j}^{(l)}}{\partial w_{ij}^{l}} \tag{5}$$

Thus, we obtain a weight adjustment function of the form of equation[6] and equation[7]. The adjustment parameter in equation[7], μ determines the rate of convergence and smoothness of the learning curve. If μ is large, then learning will be faster, although the learning curve may also be sharper and sub-optimal training results will be achieved. If μ is made small, however, then the training process is lengthy, and many epochs will be required to achieve convergence.

$$\frac{\partial e(w)}{\partial s_{j}^{l}} = \delta_{j}^{(l)} \tag{6}$$

$$\Delta w_{ij}^{(l)} = -\mu x_{i}^{(l-1)} \delta_{j}^{(l)} \tag{7}$$

5 Separability in Frequency Domain

In order to train a neural network to classify various seismic noises, I had to determine the form of input to use for the input neurons, given a signal. Initially, I experimented with the possibility of passing the entire time-domain signal of a seismic disturbance to the algorithm, or a decimated version thereof, and allowing the algorithm to distinguish key features on its own. Such a technique, however, would require vast amounts of neurons and multiple hidden layers.

Thus, I examined the various signals in the frequency domain, as in figure 3. The truck signal has a characteristic bump at around $30Hz$ due to the resonant frequency of the tires with the ground, and the earthquake signal has a much higher gain. Furthermore, if the differences between the signals are examined in various frequency bands, as in figure 4, then there is even greater linear separability, as the truck, quiet, and earthquake signals each seem to occupy a distinct region, so much that lines could be drawn to separate them.

However, both of these cases were generated with a Fast Fourier Transform, which is difficult to implement online, in real time, and thus bandpass filtering would have to be used to roughly separate the strength of a seismic signal into its component frequencies.

Over the course of my summer in the 40 Meter Lab, I constructed filters for the Bandwith Limited Root Mean Square process of examining a signal in various frequency bands in real time. More about this can be read in Appendix A. The 40 Meter lab currently separates

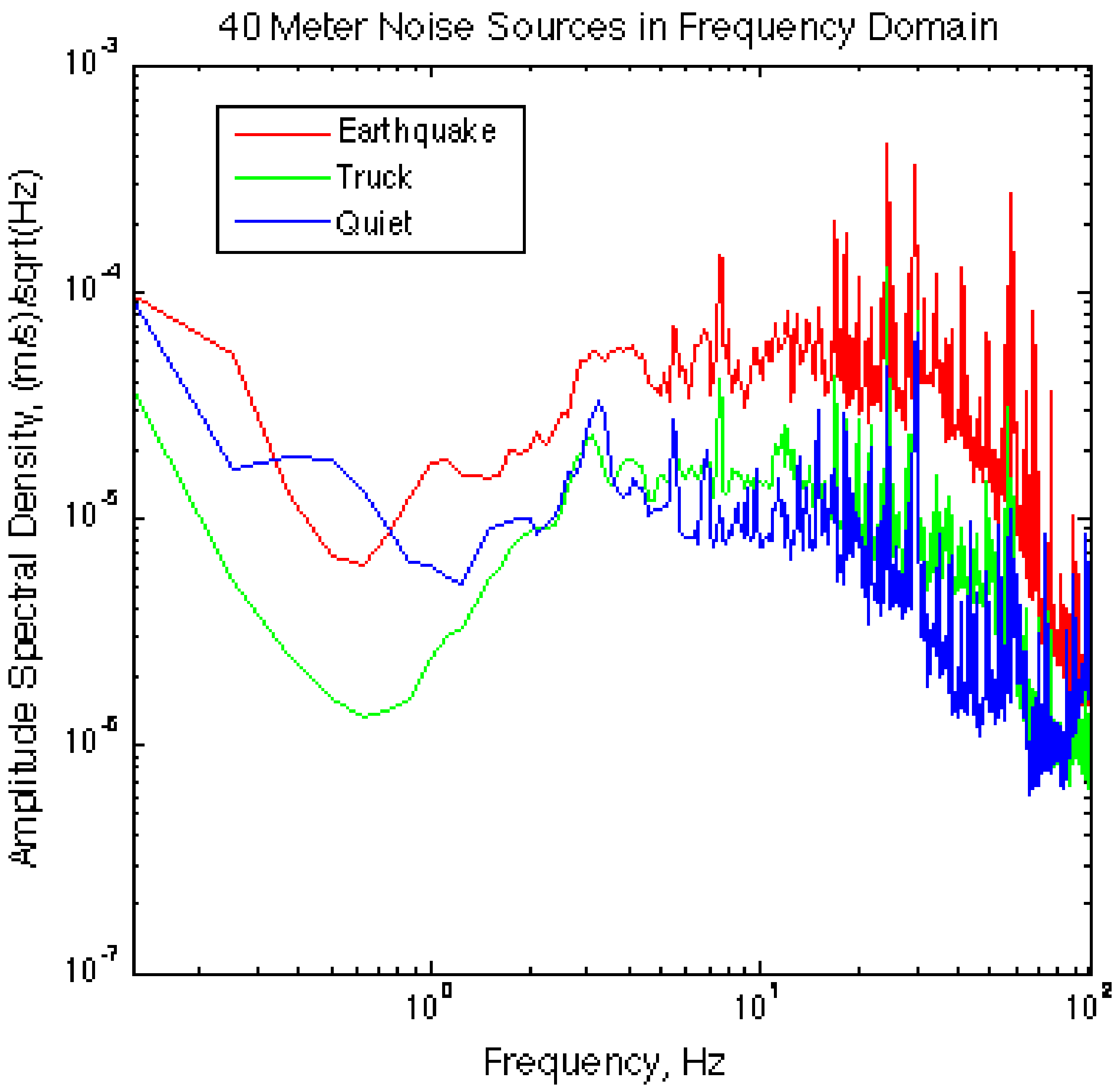

Figure 3: 40 Meter Lab Noise Sources in Frequency Domain

seismometer signals into 0.01 - 0.03 Hz, 0.03 - 0.1 Hz, 0.1 - 0.3 Hz, 0.3 - 1.0 Hz, 1.0 - 3.0 Hz, 3.0 - 10.0 Hz, and 10.0 - 30.0 Hz frequency bands, seven in total.

However, though the first two bands would be instrumental in classifying distant earthquakes, which arrive at the 40 Meter Lab traveling at low frequencies, these bands are not currently informative in practice as they are dominated by seismometer tilt and seismometer noise.

6 Training

The training input to the neural network thus included the squared Fast Fourier Transform of the normalized signal in the 0.1 - 0.3, 0.3 - 1.0, 1.0 - 3.0, 3.0 - 10.0 and 10.0 - 30.0 Hz frequency bands. Because training is performed offline, FFTs, rather than BLRMS may be used in order to ideally separate the signal, with no residuals, as in bandpass filtering.

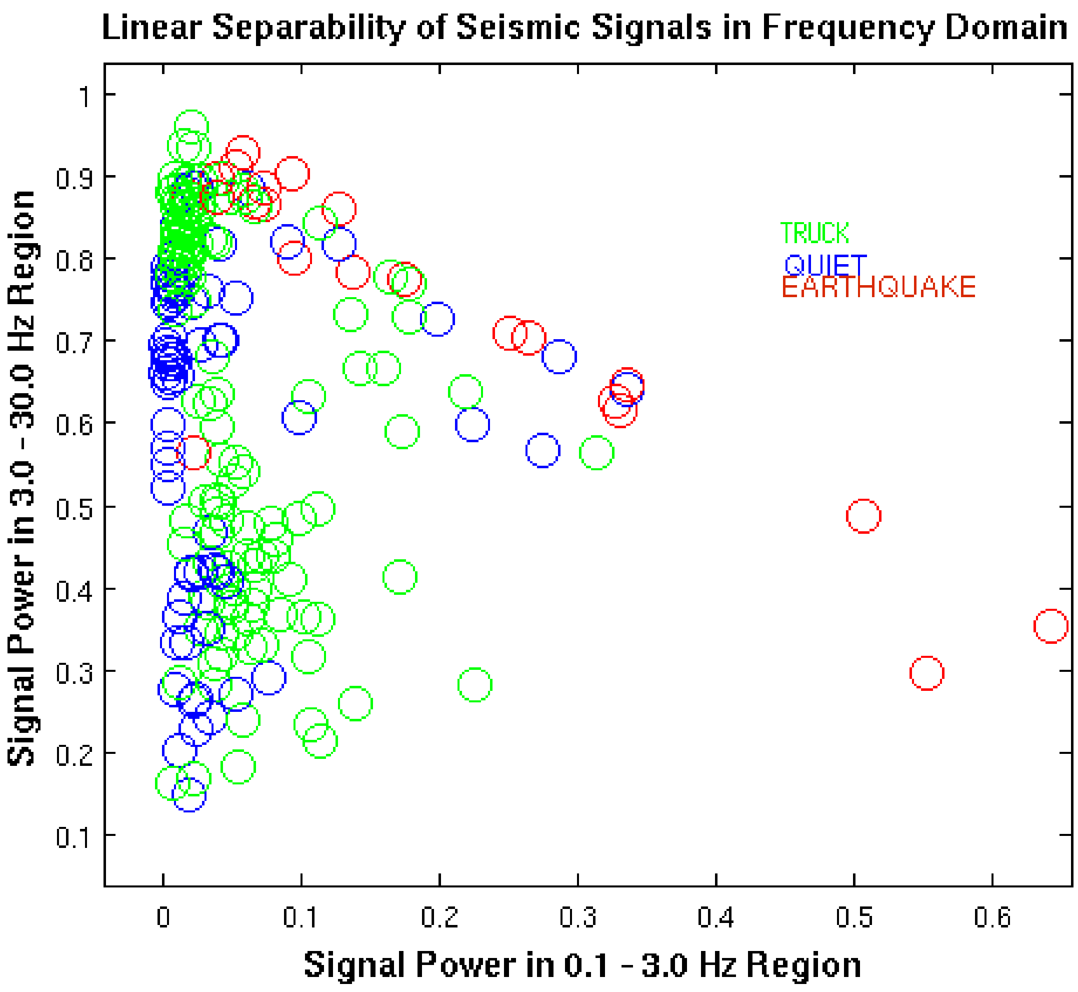

Figure 4: Linear Separability of Seismic Signals in Frequency Domain

Because the various seismometers have had various gains throughout the history of the 40 Meter Lab, and their gains continue to be varied from time to time, the input signal must be normalized to avoid any dependance on gain.

The final neural network included one input layer of these 5 frequency bands, one hidden layer of 15 neurons and one output layer of 3 neurons, with desired output vectors of the form $Earthquake : [1, 0, 0] Truck : [0, 1, 0] Quiet : [0, 0, 1]$.

The learning curve for the current feed-forward neural network is pictured in figure 5, and involved 250,000 training epochs of gradient descent back propagation with $\mu = 0.175$.

7 Classification Code

The final classification code for seismic noise was written in C, and sits in the Simulink model for C1PEM. Each of the three seismometers in this model has 3 such neural networks (for the x, y, and z directions). The code accepts the normalized BLRMS signal in the five frequency

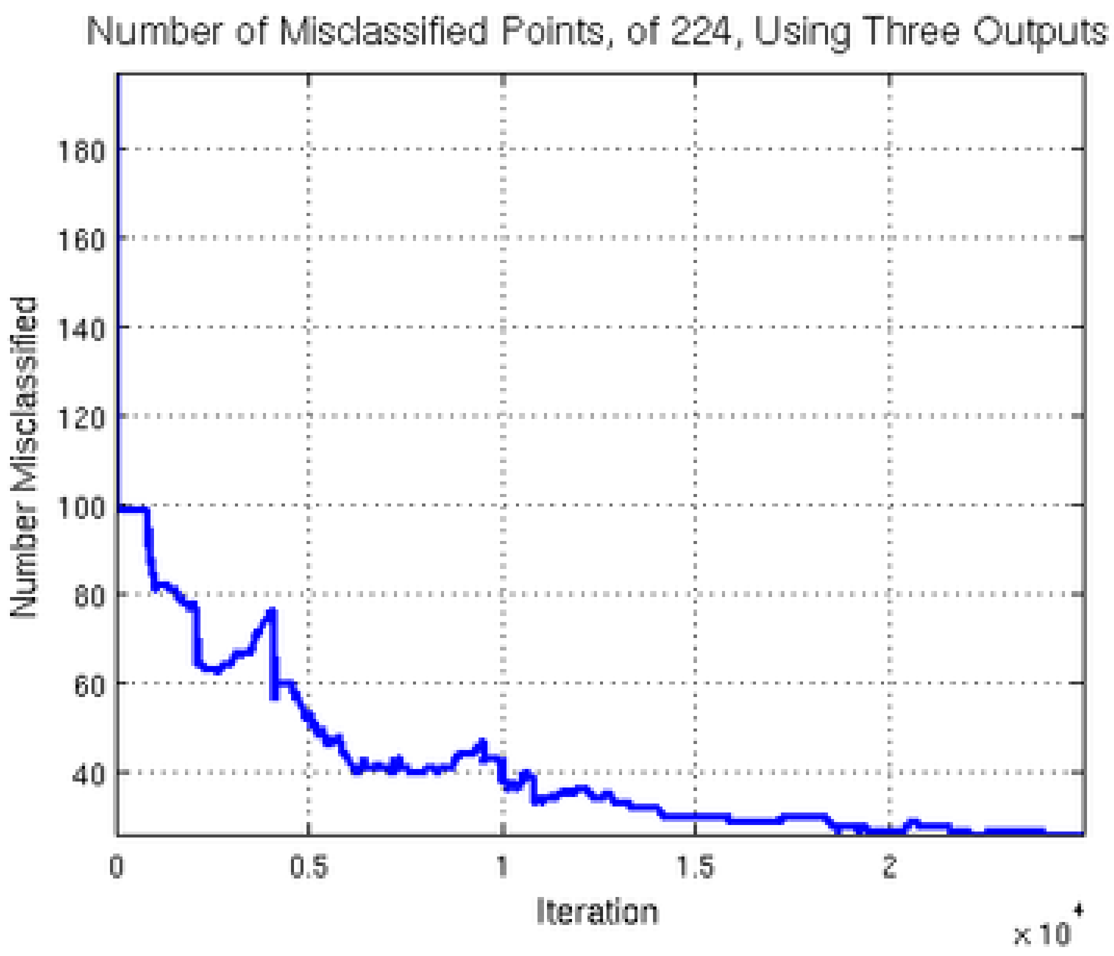

Figure 5: PEM Model Classification Widget

bands, and outputs a triple in the form of $[Earthquake, Truck, Quiet]$ which demonstrates how much of each type of seismic noise the signal represents. The code is fast in real-time not only because it is written in C and uses BLRMS rather than some soft of attempt at an online FFT, but also because the activation functions, rather than including terms of the form e^{-u}, which are time-consuming to computer, are really parametrized parabolas of the form

$$\begin{aligned} f(u) &= 0 : u < -A \\ f(u) &= \frac{(u+A)^2}{2A^2} : -A < u \leq 0 \\ f(u) &= 1 - \frac{(u-A)^2}{2A^2} : -A < u \leq 0 \\ f(u) &= 1 : u > A \end{aligned} \tag{8}$$

where A is as large as possible. For $A = 10^3$, the function looks like figure 6

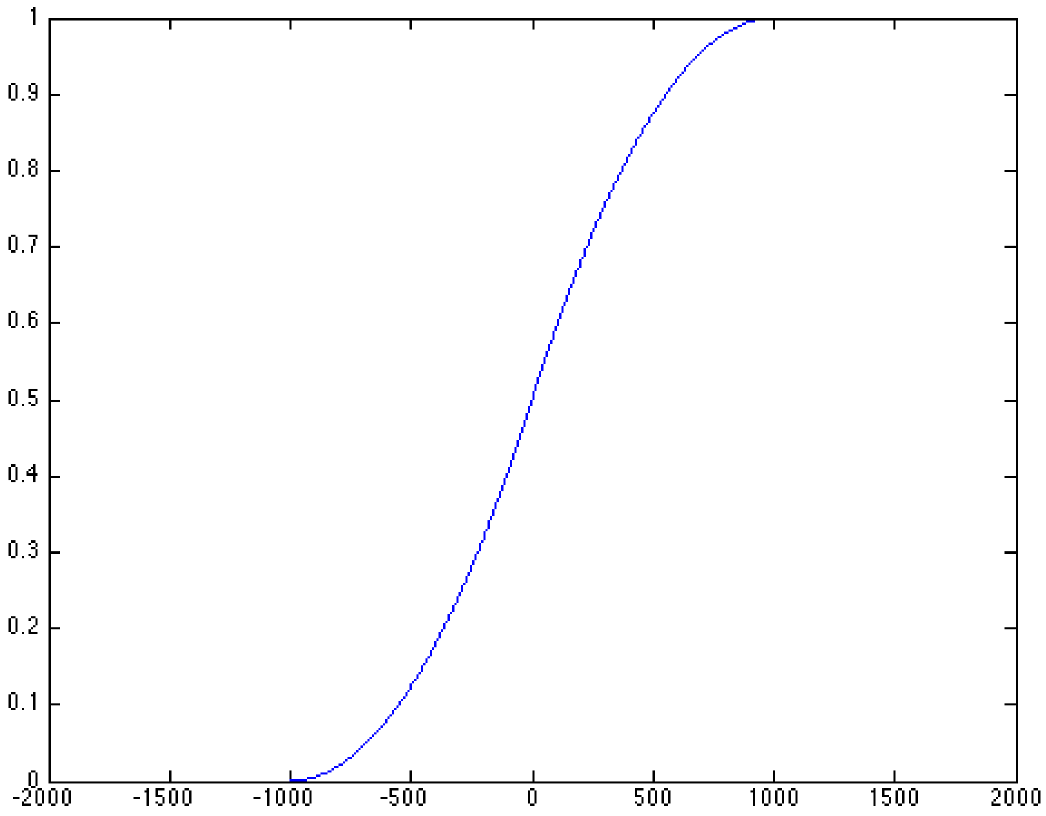

Figure 6: Hyperboloid Activation Function

8 Classification Widget

Finally, the results of classification must be presented in some manner. For this, I developed the widgets shown in figures 7 and 8. The first can be found in every PEM model, for each seismometer axis, and not only shows the seismic noise class, but also the confidence in the classification, which, for class A given that the signal is of class B, is the value of the A output neuron divided by the value of the B output neuron. In this case, the signal is classified as that of a truck, although to some extent it could be an earthquake, but is certainly not quiet. The second widget is an overview of this process for the nine seismometer channels, where darker shades of red indicate higher confidence that the signal is of that form.

9 Potential for Future Research

This project was the first attempt at a step in achieving sentience in the LIGO project through the use of advanced learning algorithms. In the field of adaptive filtering, there is potential to not only explore further classification of noise, but using classification, or the noise signal itself to generate an optimal filter through a learning algorithm. In the

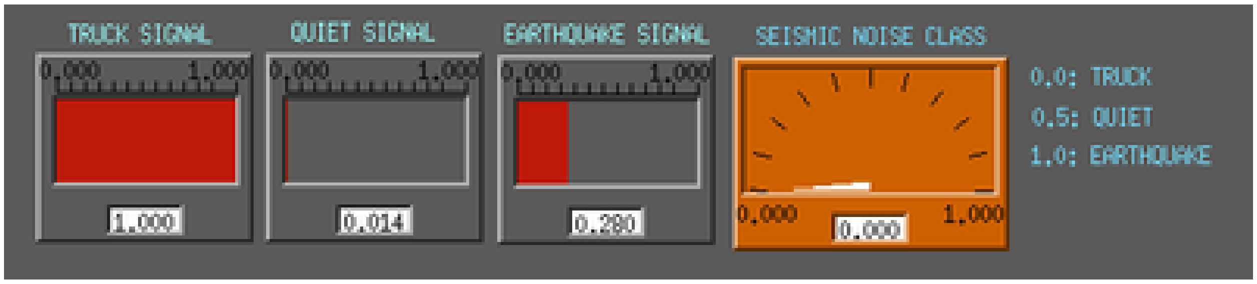

Figure 7: PEM Model Classification Widget

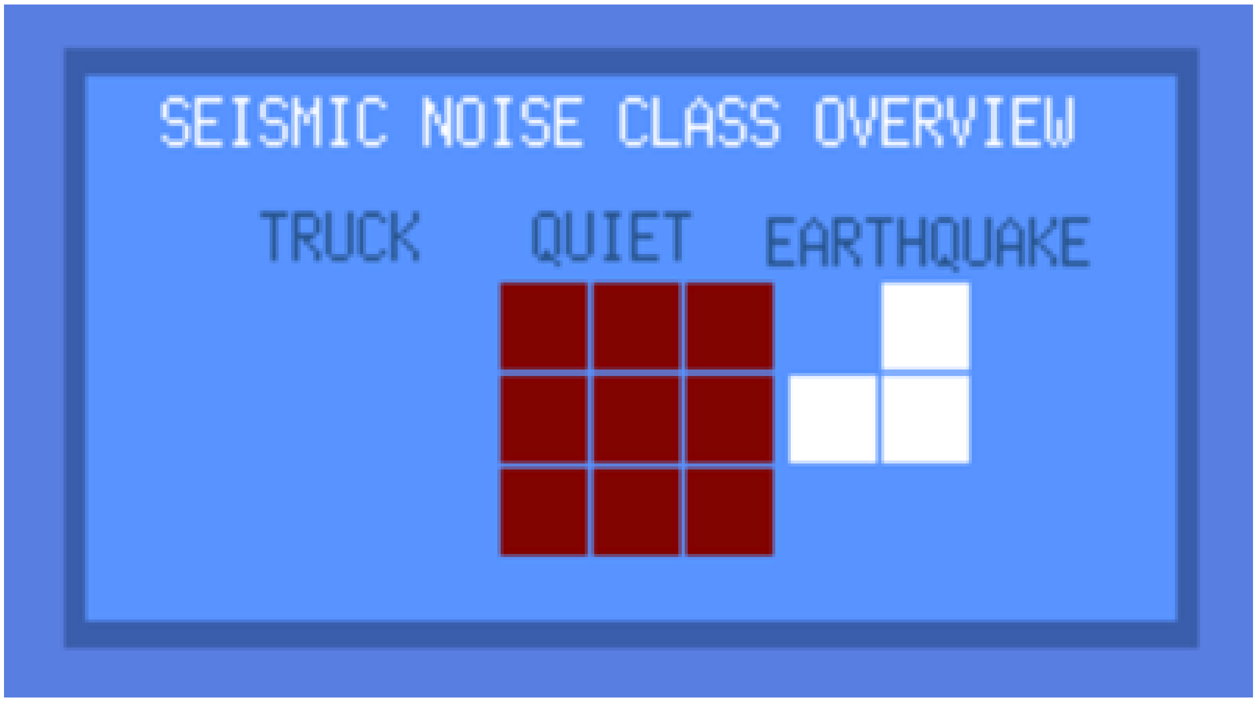

Figure 8: Overview Classification Widget

overall field of interferometer control, there is a plethora of possibilities for machine learning applications, starting with test mass control.

Over the past few weeks, since leaving Caltech, I have experimented with the idea of using neural networks to control a plant based on an error signal. Thus far, I have achieved promising results in plant identification, which is necessary in order to perform a gradient descent back-propagation learning algorithm for a neural network controller which actuates on the plant. In my case, I have built a neural network which learns the dynamics of the plant, plus noise, and is thus able to simulate the affect of actuation on the actual plant.

10 Machine Learning For Control Systems - Future Research

Rana Adhikari and I read Charles W. Anderson's paper 'Learning to Control an Inverted Pendulum Using Neural Networks' [3], in which he outlines the process of using a feedforward neural network with unsupervised learning to generate a necessary force, given the displacement, angle, and the time derivates thereof of an inverted pendulum, to keep the pendulum from falling and the cart from sliding off-center. A similar approach is also outlined in 'Neuronlike Adaptive Elements that Can Solve Difficult Learning Control Problems' [4]. We wondered whether it would be possible to apply a similar principle to the test masses of the interferometer. While the dynamics of the test mass suspensions can be approximated, environmental factors such as temperature and seismic noise essentially make the actual dynamics of the system difficult to directly model. Thus, a neural network could be used for function approximation, to continuously learn the dynamics of the system and output the necessary forces to keep the test masses in resonance. This, however, is a difficult problem since, in order to safely train such a network, a complex model of the optical cavity is necessary.

Likewise, in order to better approximate the dynamics, a neural network architecture which takes into account past values of a time series is preferable. Recurrent neural networks have a suitable architecture for this operation[5], as each neuron in a given layer sums not only the current input multiplied by a weight, but also the previous T outputs of the neuron, where T is the time delay. In fact, there is a large body of literature on using recurrent neural networks for system identification [6], [7], [8].

I spent some time this summer working on neural network controllers in both Matlab and C, using an recurrent neural network based on the gradient descent weight update algorithm outlined by Mikael Boden in 'A guide to recurrent neural networks and back propagation' [9] and Kumpati S. Narendra in 'Adaptive Control Using Neural Networks' [10]. The plant, in this case, was a driven, damped harmonic oscillator, and the cost function was, is the displacement of the oscillator squared. However, this could be changed to the velocity of the system squared, or, in a more complicated system, the displacement between two test masses. Likewise, activity can be measured within a certain frequency band and used as the error using band-pass limiting filters.

11 Acknowledgements

This work would not have been possible without the LIGO Scientific Collaboration, the NSF, and the Caltech SURF Program. I am very grateful to Denis Martynov, Jenne Driggers, and Rana Adhikari, as well as the other members of the 40 Meter Lab for all of their guidance and support throughout this research project.

References

[1] *U.S. Geological Survey* Earthquake Search
http://earthquake.usgs.gov/earthquakes/eqarchives/epic/

[2] Principe, Jose C. *Dynamic Neural Networks and Optimal Signal Processing* Handbook of Neural Network Signal Processing Ed. Yu Hen Hu, Jenq-Neng Hwang CRC Press, 2002

[3] Anderson, Charles W. *Learning to Control an Inverted Pendulum Using Neural Networks* IEEE Control Systems Magazine April 1989

[4] Barto, Andrew G., Sutton, Richard S., Anderson, Charles W. *Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems* IEEE Transactions on Systems, Man, and Cybernetics, Vol SMC-13, No. 5 September - October 1983

[5] Tutschku, K. *Recurrent Multilayer Perceptrons for Identification and Control: The Road to Applications* University of Wurzburg, Institute of Computer Science Research Report Series, Report No. 118 June 1995

[6] Feldkamp, Lee A., Puskorius, Gintaras V. *A Signal Processing Framework Based on Dynamic Neural Networks with Application to Problemts in Adaptive Filtering, and Classification* Proceedings of the IEEE, Vol. 86, No. 11 November 1998

[7] Li, Xiaoou, Yu, Wen *Dynamic system identification via recurrent multilayer perceptrons* Information Sciences, 147 (2002) 45-63 April 2002

[8] Amari, Shun-Ichi, and Cichocki, Andrzej *Adaptive Blind Signal Processing - Neural Network Approaches* Proceedings of the IEEE, Vol. 86, No. 10 October1998

[9] Boden, Mikael *A guide to recurrent neural networks and backpropagation* School of Information Science, Computer and Electrical Engineering Halmstad University November 13, 2001

[10] Narendra, Kumpati S. *Adaptive Control Using Neural Networks* Neural Networks for Control, Chapter 5 Ed. Miller, W. Thomas, Sutton, Richard S., Werbos, Paul J. MIT Press, 1990

[11] *Simulink Model Reference Controller Block Documentation* http://www.mathworks.com/help/toolbox/nnet/ug/bss37c6-1.html

[12] Williams, Ronald J. *Training Recurrent Networks Using the Extended Kalman Filter* IEEE 1992

12 Appendix A: BLRMS

In order to determine the strength of signal at various frequencies in real time (in other words, without performing a Fourier Transform), one can use the band-limited root mean square method, in which a signal is sent through a set of digital bandpass filters in parallel, and the root mean square of the signal is calculated within each passband in order to determine the amplitude of the signal. I worked on developing a series band-limited root mean square channels for the seismometer signals from Guralp 1, Guralp 2, and the Streckeisen seismometer.

Currently, each signal is band-pass filtered into the following bands:

$$0.01 - 0.03|0.03 - 0.1|0.1 - 0.3|0.3 - 1|1 - 3|3 - 10|10 - 30 \text{ Hz}$$

I added the $0.01 - 0.03$ and $0.03 - 0.1$ Hz channels specifically so that distant earthquake could be observed. Suppose we wanted to find calculate the band-limited root mean square in the $1 - 3$ Hz region for the Guralp 1 Z signal. The first step is to construct a digital bandpass filter to remove the signal that is not in the $1 - 3$ Hz region. A Butterworth bandpass filter with cut-off frequencies at 1 Hz and 3 Hz serves this purpose in theory, as shown in figure 9

Note that the ratio of the input to output is 1 in the passband (in decibels, $20 \cdot \log_{10}$(the ratio) $= 20 \cdot \log_{10}(1) = 0dB$), and is -40 dB in the next frequency order at 10 Hz (meaning that the ratio is .01, or that 99% of the incoming signal is filtered out at this frequency).

Now that we have band-passed the signal in order to obtain almost only signal at frequencies $1 - 3$ Hz, we have to perform the root mean square operation:

$$S_{RMS} = \sqrt{\frac{1}{N}\sum_{n=0}^{N} \text{Signal}^2} \tag{9}$$

Note that we use a sum rather than an integral since out signal is discrete. Suppose we had a signal of the form $A \sin \omega t$. In this case, we would obtain

$$S_{RMS} = \sqrt{\frac{1}{N}\sum_{n=0}^{N} \left\{A \sin \omega t\right\}^2} = \sqrt{\frac{1}{N}\sum_{n=0}^{N} \left\{\frac{A(1 - \cos 2\omega t)}{2}\right\}^2}$$

In this case, the squared signal would also include higher frequencies that $1 - 3$ Hz in the $\cos(2\omega t)$ term, so we have to apply a low-pass filter to remove this term and leave only the signal corresponding to the A term in order to measure the magnitude. For our purposes, a 4th order lowpass filter which cuts off at $0.7Hz$ (in other words, its transfer function has 4 poles at 0.7) low-pass filters enough of the signal, as shown in 10.

After the low-pass filter, the root of the signal is taken. However, how do we determine the order of filter which works in practice? A useful method is to look at the coherence of a signal before a filter is applied with the signal after the filter is applied. In the bottom plot of figure 11, we can see that in the $1 - 3$ Hz region, the signal is coherent with itself both through low-pass filtering and through band-pass filtering. The top plot of 11, in turn, shows the whole RMS process. The BP_1_3_IN1 signal shows the original seismometer signal, the

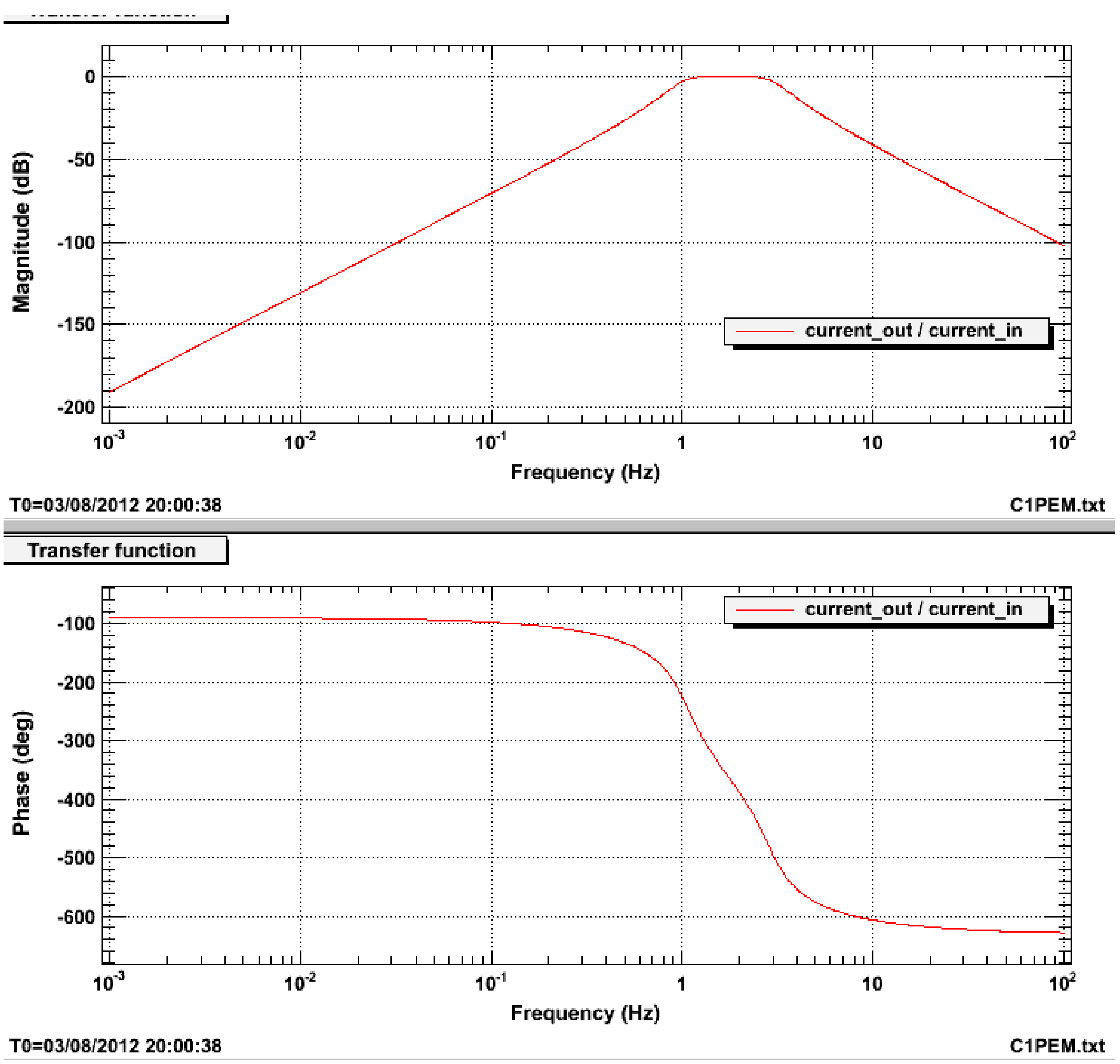

Figure 9: An order 3 Butterworth Bandpass 1-3 Hz Filter

BP_1_3_OUT signal shows the signal after it has been band-passed in the 1-3 Hz region and a gain has been applied to convert from counts to m/s. Note how the magnitude of the signal is greatest in the 1-3 Hz region for BP_1_3_OUT, as expected. LP_1_3_IN1 shows the signal after it has been squared, and thus corresponds to the square of the BP_1_3_OUT signal, as shown on the graph. LP_1_3_OUT, in turn, corresponds to LP_1_3_IN1 after it has been low-passed. Note that there is less magnitude at higher frequencies than in LP_1_3_IN1 , as expected.

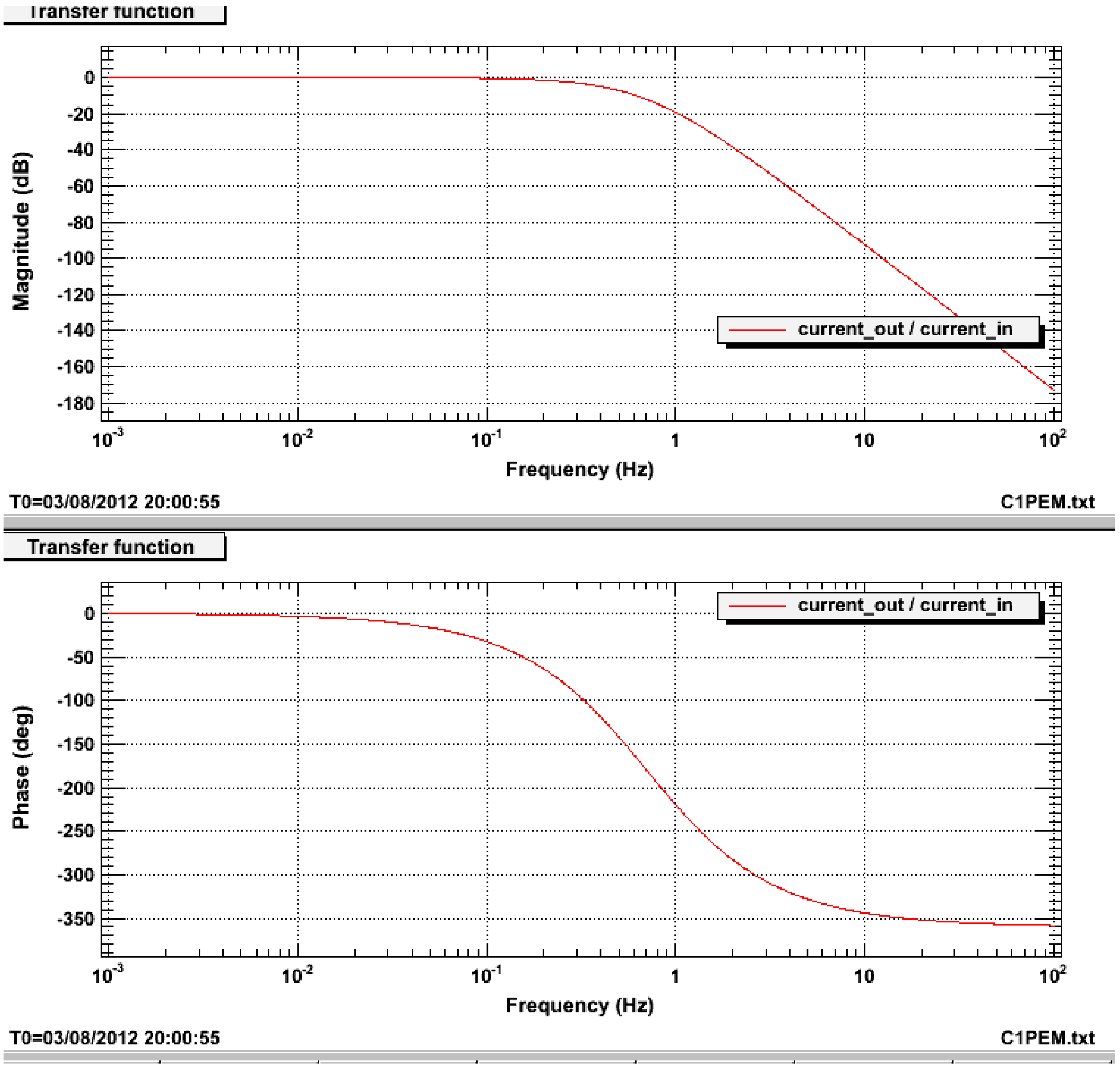

Figure 10: An order 4 Lowpass 0.7 Hz Filter

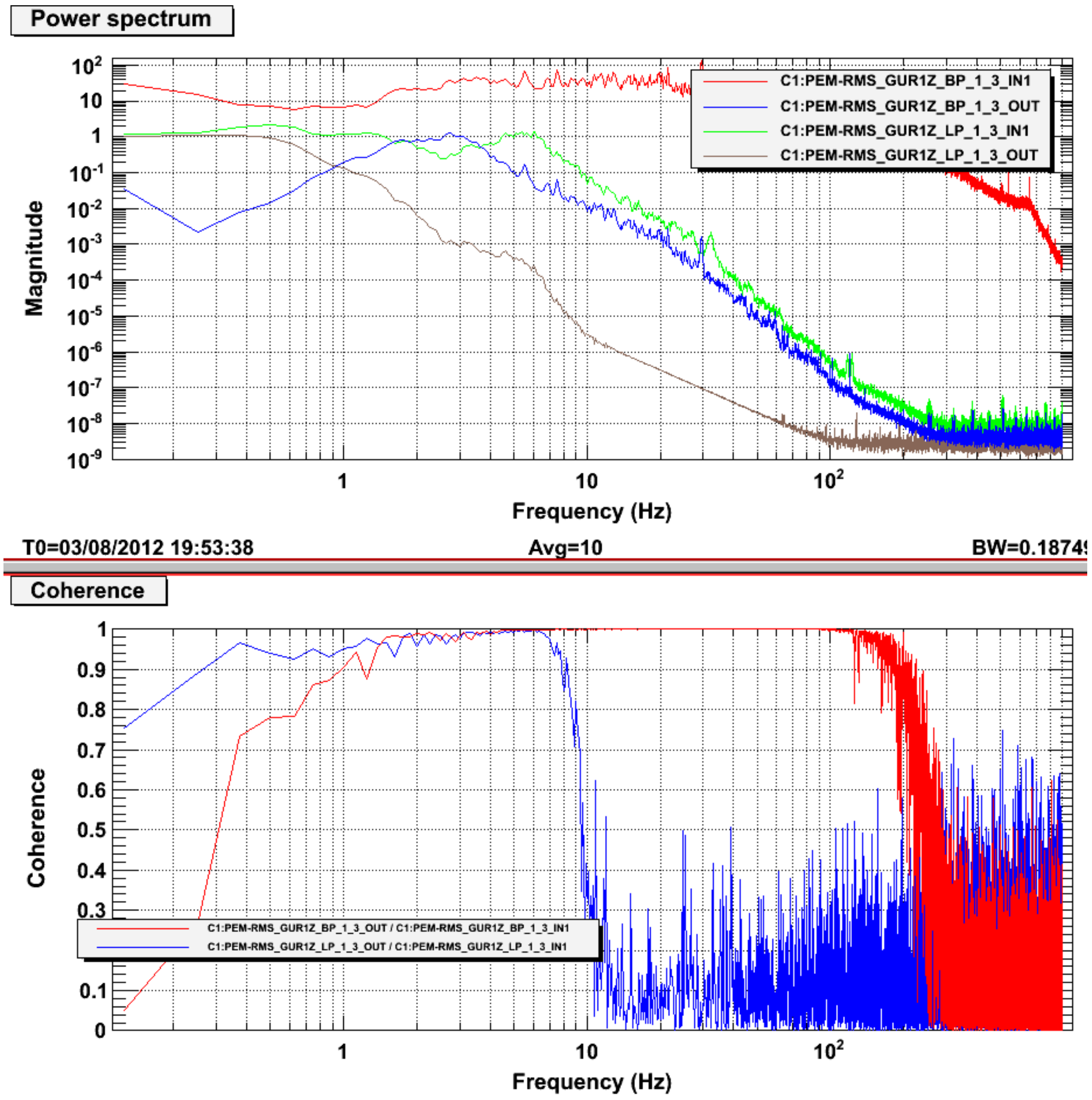

Figure 11: Guralp 1 Z BLRMS process