Adaptive Control Using IIR Lattice Filters

Stephen J. Hevey

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Masters in Science

in

Electrical Engineering

April 24, 1998
Blacksburg, Virginia

Keywords: Adaptive Control, IIR Filters, Lattice Filters, Adaptive-Q

Adaptive Control Using IIR Lattice Filters

Stephen J. Hevey

(ABSTRACT)

This work is a study of a hybrid adaptive controller that blends fixed feedback control and adaptive feedback control techniques. This type of adaptive controller removes the requirement that information about the disturbance is known apriori. Additionally, the control structure is implemented in such a way that as long as the adaptive controller is stable during adaptation, the system consisting of the controller and plant remain stable.

The objective is to design and implement an adaptive controller that damps the structural vibrations induced in a multi-modal structure. The adaptive controller utilizes an adaptive infinite impulse response lattice filter for improved damping over the fixed feedback controller alone. An adaptive finite impulse response LMS filter is also implemented for comparison of the ability for both algorithms to reject harmonic, narrow bandwidth and wide bandwidth disturbances.

It is demonstrated that the lattice filter algorithm performs slightly better than the LMS filter algorithm in all three disturbance cases. The lattice filter also requires less than half the order of the LMS filter to get the same performance.

Acknowledgements

I would like to thank Dr. Baumann for his guidance as my committee chair and for his infinite patience. Without his help, I would probably still be scratching my head in the lab wondering why things are not working properly. I would also like to thank Dr. Bay and Dr. Vanlandingham for their suggestions and serving on my committee.

Special thanks go to my family. To my brothers, Mike and Matt, for their understanding when I was unable to spend time with them because I was too busy. To my mother and father, who always seem to know just what to say to lift my spirits. To Donna and Wade, for always supporting and encouraging me.

Additional thanks go to Dr. Baumann's family for understanding each time Bill spent a night or weekend helping out in the lab.

Finally, to my wife, Charlene, there are no words great enough to express my thanks for being there for me and understanding why I spent so many nights and weekend away from home.

Table of Contents

Table of Figures

Chapter 1 Adaptive Control Schemes

Most current methods of adaptive disturbance rejection applied to structures have used feedforward [1,2] or feedback [3,4] control algorithms utilizing an adaptive finite impulse response (FIR) filter. Adaptive feedforward controllers using FIR filters have been used successfully for active control for some time, with applications in controlling noise in ducts [20], cars [21], and aircraft [22].   Implementation of an adaptive feedforward controller has the advantages of inherent stability and simplicity in design. Unfortunately, a major problem with using feedforward control is the requirement that an independent measurement be obtained which is coherent with the disturbance.

An adaptive feedback control scheme removes the requirement that independent coherent measurements be obtained but sacrifices the inherent stability and simplicity of design associated with most feedforward designs. The adaptive feedback approach has been investigated by many for the reduction of acoustical noise. Applications include reducing the noise in air conditioning ducts [23], noise from sunroof air oscillations [24], and damping the sound field in a hearing protector [25].

One particular implementation of an adaptive feedback controller adds an adaptive filter to a standard linear quadratic gaussian (LQG) controller to enhance disturbance rejection in a feedback control loop. This controller is called a hybrid controller and has shown that the addition of a feedback loop to provide system damping results in faster convergence of the adaptive filter and lower filter orders for the same performance when compared with the undamped case [18,19]. One particular use of this feedback control design is referred to as Adaptive-Q feedback and employs a technique called a "neutralization loop" [5]. The neutralization loop breaks the feedback path that contains the adaptive filter in such a way that as long as the filter stays stable during adaptation, it cannot drive the control system unstable.

Currently, only FIR filters are used when implementing this adaptive feedback control approach. This thesis considers the use of an infinite impulse response (IIR) filter in place of the FIR filter. One potential benefit of using an IIR filter in place of the FIR filter is the reduction in the size of the required filter. A drawback to using an IIR filter is finding an adaptive algorithm for the filter that is reasonable to implement with a digital signal processor and ensures that the filter will stay stable during adaptation.

An adaptive IIR lattice filter utilizing a tapped state recursive lattice filter structure can be adapted in such a way as to guarantee stability [6]. This filter is adapted in time using a gradient descent algorithm. Although complex to implement, the algorithm has been simplified enough to allow implementation of the filter on a digital signal processor (DSP). Replacing the standard FIR filter with this adaptive IIR filter could potentially require fewer computations for the adaptive-Q controller when implemented on complex multi-modal plants.

The performance of the IIR lattice filter in the adaptive-Q feedback algorithm will be compared with the performance of an FIR filter in the same system. Simulations were performed, as well as a physical implementation of the system. The physical implementation is the only way to see the true complexity of the lattice algorithm and the only way to find any problems that exist which are not represented in simulation. Both systems were subjected to harmonic, narrow band, and wide band disturbances, and the results compared to the theoretically optimal performance.

Chapter 2 Control System Development

Consider a plant with a Linear Quadratic Gaussian (LQG) feedback controller added to the system. It has a layout as shown in Figure 1.

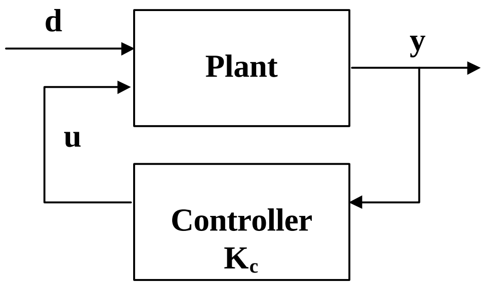


Figure 1 - Plant with LQG feedback controller

The Adaptive-Q controller builds off of the LQG controller by placing an adaptive filter into the controller as shown in Figure 2.

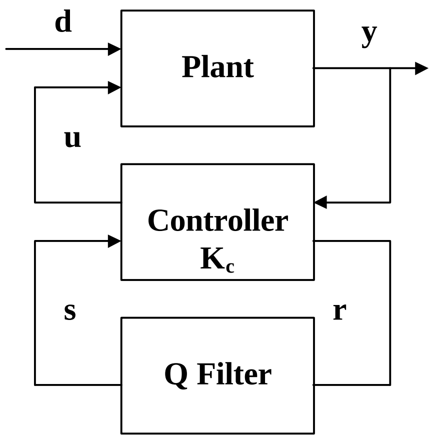


Figure 2 - Plant with LQG controller and Q Filter

This adaptive controller can be modeled as follows. Consider the plant with the state description:

$$x_{k+1} = A x_k + B u_k + E d_k$$
$$y_k = C x_k$$

where x_k are the states of the plant at time k. It is assumed that the controller knows nothing about the disturbance or any dynamics associated with the disturbance. With this in mind, the standard LQG framework will produce a state estimate feedback with state description:

$$\begin{aligned}
&\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L\,(y_k - \hat{y}_k) \\
&u_k = -(s_k + K\hat{x}_k) \\
&r_k = y_k - \hat{y}_k
\end{aligned}$$

where $\hat{x}_k$ is the state estimate at time k, u_k is the controller feedback, r_k is the output estimation error, s_k is a new control input required for the adaptive filter, K is the LQR state feedback gain, and L is the Kalman filter gain. The Kalman filter gain is calculated by assuming white noise acting on each of the modes of the system. In this way, some information about the disturbance is obtained for the state estimator design, but spectral information about the disturbance itself and how it enters the plant remains unknown.

The Q filter is placed between the error estimate, r_k, and the new input signal, s_k. Since u_k is fed directly into the state estimator, s_k produces no estimation error in either the state or the output [8]. Thus, the fixed feedback controller forces the transfer function between s_k and r_k to be identically equal to zero and a "neutralization loop" is formed. A proof of the transfer function being identically equal to zero is shown below.

Proof that $\mathbf{TF_{rs} \equiv 0}$

Consider the system with the following state equations:

$$\begin{aligned}
&x_{k+1} = Ax_k - B(s_k + K\hat{x}_k) + Ed_k \\
&\hat{x}_{k+1} = A\hat{x}_k - B(s_k + K\hat{x}_k) + L(Cx_k - C\hat{x}_k) \\
&r_k = Cx_k - C\hat{x}_k
\end{aligned}$$

Rewrite in matrix format:

$$\begin{bmatrix} x_{k+1} \\ \hat{x}_{k+1} \end{bmatrix} = \begin{bmatrix} A & -BK \\ LC & A-BK-LC \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} + \begin{bmatrix} -B \\ -B \end{bmatrix} s_k + \begin{bmatrix} E \\ 0 \end{bmatrix} d_k$$

$$r_k = \begin{bmatrix} C & -C \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix}$$

Transform using the Transform matrix T

$$T = \begin{bmatrix} I & 0 \\ I & -I \end{bmatrix}$$

and note that $T^{-1} = T$. In transfer function format,

$$\begin{aligned} \text{TF} &= \text{C (sI-A)}^{-1}\text{ B} \\ &= \text{C T}^{-1}\text{T (sI-A)}^{-1}\text{ T}^{-1}\text{T B} \\ &= \text{C T T (sI-A)}^{-1}\text{ T T B} \end{aligned}$$

Again rewrite into matrix format:

$$\begin{bmatrix} I & 0 \\ I & -I \end{bmatrix} \begin{bmatrix} A & -BK \\ LC & A-BK-LC \end{bmatrix} \begin{bmatrix} I & 0 \\ I & -I \end{bmatrix} = \begin{bmatrix} A-BK & BK \\ 0 & A-LC \end{bmatrix}$$

Multiply C and T matrices on left and T and B matrices on right:

$$\underbrace{\begin{bmatrix} C & -C \end{bmatrix} \begin{bmatrix} I & 0 \\ I & -I \end{bmatrix}}_{\begin{bmatrix} 0 & C \end{bmatrix}} \begin{bmatrix} A-BK & BK \\ 0 & A-LC \end{bmatrix}^{-1} \underbrace{\begin{bmatrix} I & 0 \\ I & -I \end{bmatrix} \begin{bmatrix} -B \\ -B \end{bmatrix}}_{\begin{bmatrix} -B \\ 0 \end{bmatrix}}$$

$$\begin{bmatrix} 0 & C \end{bmatrix} \begin{bmatrix} A-BK & BK \\ 0 & A-LC \end{bmatrix}^{-1} \begin{bmatrix} -B \\ 0 \end{bmatrix}$$

To find inverse of the inner matrix, consider

$$\begin{bmatrix} A-BK & BK \\ 0 & A-LC \end{bmatrix} \begin{bmatrix} (A-BK)^{-1} & ? \\ 0 & (A-LC)^{-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

The term containing the "?" mark is not needed because it is multiplied by zero as seen below.

$$[0 \quad C]\begin{bmatrix}(A-BK)^{-1} & ? \\ 0 & (A-LC)^{-1}\end{bmatrix}\begin{bmatrix}-B \\ 0\end{bmatrix}$$

$$\left[0 \quad C(A-LC)^{-1}\right]\begin{bmatrix}-B \\ 0\end{bmatrix}=0 \qquad \therefore \quad TF_{rs} \equiv 0$$

Since this neutralization loop "breaks" the feedback path between the control input, u_k, and the feedback estimation error, r_k, the Q filter does not appear as a new feedback loop in the system. As a result, this "broken path" closely resembles the adaptive feedforward control approach [5]. This has the advantage that as long as Q is chosen to be a stable adaptive filter, it will not drive the feedback control system unstable when placed across this neutralization loop. Additionally, it can be shown [7] that as the Q filter sweeps over the set of all stable transfer functions, the controller consisting of the fixed controller and Q, sweeps over the set of all stabilizing controllers for the plant. Thus, to find the best controller for the plant, one needs only to find the best stable Q.

Using mixed notation, the output y_k can be modeled as a combination of the disturbance input d_k and the filter input s_k, as shown below:

$$\begin{aligned} y_k &= T_{yd} d_k + T_{ys} s_k \\ s_k &= Q r_k \end{aligned}$$

T_{yd} is the transfer function from the disturbance input to y_k, and T_{ys} is the transfer function from the Q filter output to y_k. The problem of minimizing the output, y_k, can be reformulated as a system identification problem, provided both y_k and r_k are scalars

[8]. If both y_k and r_k are scalars, T_{ys} and Q commute, and the equations above can be rewritten as:

$$z_k = Qi_k + y_k$$

This is in the form of the standard output error model used in system identification as described by Ljung [12], where:

$$\begin{aligned}
&z_k = T_{yd}d_k \quad \textit{is the diturbance output} \\
&i_k = -T_{ys}r_k \quad \textit{is the adaptive filter input} \\
&y_k \quad \textit{is the residual error}
\end{aligned}$$

Now that the problem of minimizing y_k can be thought of as solving the system identification problem, system identification algorithms can be applied to the problem of adapting Q.

The final problem is finding an IIR adaptive filter that can be adapted using a system identification method. An adaptive IIR lattice filter using tapped-state recursive lattice filters can be used to solve a standard output error system identification problem [6]. Section 2.1 discusses the selection of a lattice filter as an adaptive IIR filter. Sections 2.2 and 2.3 discuss the structure of the lattice filter and the method of adaptation. Modification of the system identification problem for implementation with the adaptive-Q controller is discussed in section 2.4.

2.1 Choosing an Adaptive IIR Filter

When considering IIR filters, the direct form filter is the common structure of choice. This is true, in general, because when designing an algorithm which adapts the parameters a_k and b_k, the coefficients of the difference equation, described below, are manipulated directly.

$$y_k + a_1 y_{k-1} + \cdots + a_m y_{k-m} = b_0 u_k + b_1 u_{k-1} + \cdots + b_m u_{n-m}$$

Some problems exist in using the direct form filter for adaptive applications. First of all, ensuring stability of a time-varying direct form filter can be a major difficulty. It is often computationally a burden because the polynomial, A(z), made up of the a_k parameters, must be checked to see if it is minimum phase at each iteration [6]. Even if the stability was assured during adaptation, roundoff error causing limit cycles can plague the filter [11].

Parallel and cascade forms are often used as alternatives for direct form filters. These consist of an interconnection of first and second order filter sections, whose sensitivity to roundoff errors tends to be less drastic than for the direct form filter [6]. Since the filter is broken down into a factored form, the roundoff error associated with each factorization only affects that term. In the direct form filter, the factors are lumped together so that roundoff error in each term affects all of the factors in turn.

A larger problem exists for both parallel and cascade forms: the mapping from transfer function space to parameter space is not unique. Whenever the mapping from the transfer function space to the parameter space is not unique, additional saddle points in the error surface appear that would not be present if the mapping had been unique [13]. The addition of these saddle points can slow down the convergence speed if the parameter trajectories wander close to these saddle points. Examples illustrating this phenomenon are contained in [14]. For this reason, these filter forms are considered unsuitable for adaptive filtering [6].

A tapped-state lattice form has many of the desirable properties associated with common digital filters and avoids the problems discussed above. Section 2.2 discusses the structure of the tapped-state lattice form and the properties that make this filter form suitable for adaptive applications.

2.2 Tapped-state Recursive Lattice Filters

A tapped-state recursive lattice filter consists of a cascade of interconnected Schur recursion steps, denoted q, and a summation of weighted tap parameters, denoted n. The basic step in the Schur recursion structure is depicted in Figure 3.

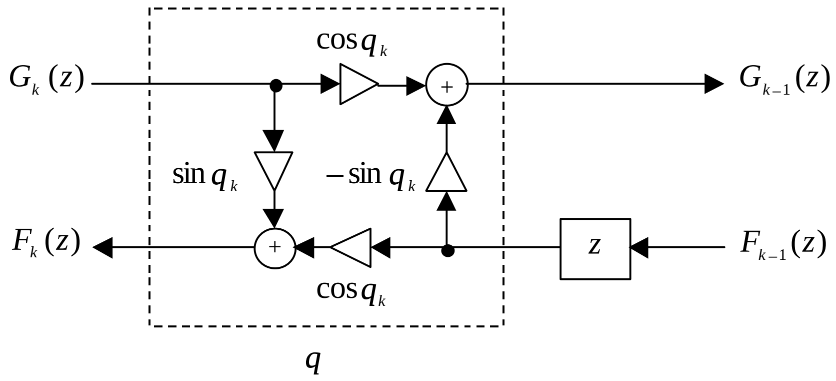


Figure 3 – Schur recursion

The Schur recursion contains rotation angles, q_k, that span the forward and backward signal path of the lattice structure. These Schur sections cascade to form the recursive part of the lattice filter as depicted in Figure 4.

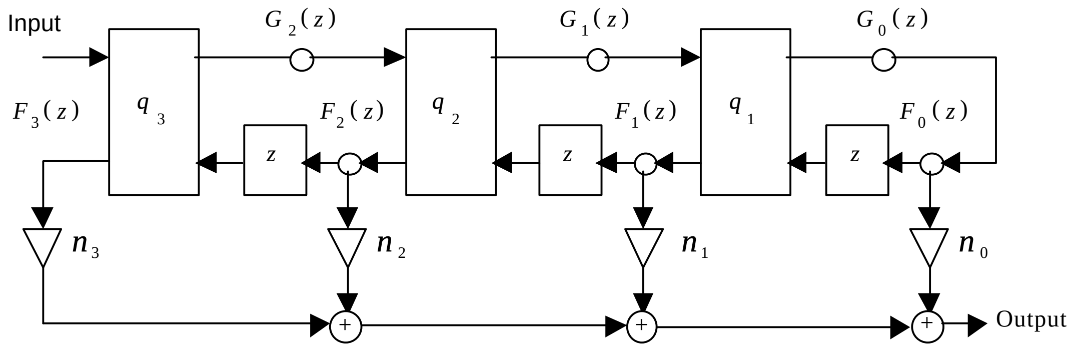


Figure 4 – Tapped-state recursive lattice filter

Summing the taps, $F(z)$, weighted by the n_i parameters, generates the output of the filter.

Due to the computational structure, the roundoff error in this filter is inherently low. Additionally, a tapped-state recursive lattice IIR filter is stable [6] when $|q_k| < p/2$ for all k. This filter is stable even for time-varying coefficients, as long as $|q_k| < p/2 - e$ for all k and some fixed e. Also, bounding the rotation angles in this way, forces the lattice parameterization to be unique [6]. These properties render the filter quite suitable for adaptive applications [6].

2.3 System ID Algorithm Using an Adaptive Lattice Filter

As stated in chapter Chapter 2 , the problem of minimizing the disturbance in the system can be reformulated as a system identification problem. Figure 5 depicts the standard system identification problem using an adaptive IIR filter.

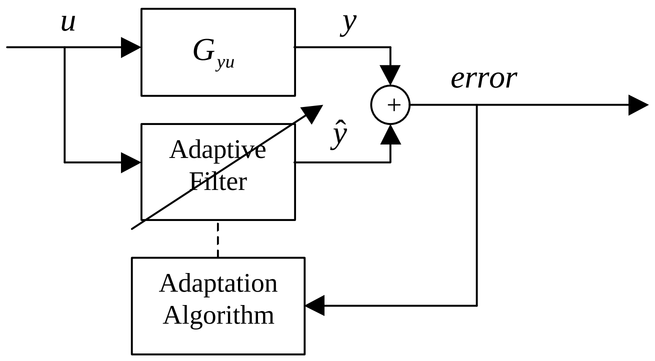


Figure 5– System ID problem using adaptive filter

This type of identification passes an input signal, u, through the plant G_{yu} and the adaptive filter. In Figure 5, y is the output of the plant, $\hat{y}$ is the estimate of the output of the plant, and the error signal is defined as, $error = y - \hat{y}$. The adaptation algorithm minimizes this error signal and calculates a gradient vector to adjust the weights of the adaptive filter. When the error signal reaches some small number close to zero, the systems have a closely matched frequency response.

An adaptive IIR filter using tapped-state recursive lattice filters can solve the standard system identification problem [6] described above. A simplified algorithm is considered because it reduces the computational complexity of the adaptation. This algorithm adapts both q_k and n parameters in the lattice filter by a small step size, m , opposite the direction of the calculated gradient. The gradients for the n parameters are calculated using the original lattice filter. A second lattice filter, called a post filter, is used to compute the gradient associated with the q_k parameters. Additionally, the q_k are bounded within $\pm p/2$ to guarantee stability for the lattice filter. This simplified form of the standard algorithm is presented below.

Simplified Partial Gradient Lattice Algorithm

Available at time n:

- Filter coefficients:

$$n_k(n), \quad k = 0,1,\cdots,M$$
$$q_k(n), \quad k = 1,2,\cdots,M$$

- Adaptive filter states:

$$\nabla_{n_k}(n-1), \quad k = 0,1,\cdots,M-1$$

  The variable $\nabla_{n_M}(n)$ will be computed below, but need not be stored.

- Post filter states:

$$x_k(n), \quad k = 0,1,\cdots,M-1$$

  The variable $x_M(n+1)$ will be computed below, but need not be stored.

- New data:

$$u(n) \quad \textit{Input Sample}$$
$$y(n) \quad \textit{reference Sample}$$

Adaptive filter computation:

```
    Let $g_M = u(n)$
    For $k = M, M-1, \cdots, 1$ do
        $\begin{bmatrix} g_{k-1} \\ \nabla_{n_k}(n) \end{bmatrix} = \begin{bmatrix} \cos q_k(n) & -\sin q_k(n) \\ \sin q_k(n) & \cos q_k(n) \end{bmatrix} \begin{bmatrix} g_k \\ \nabla_{n_{k-1}}(n-1) \end{bmatrix}$
    End {For}
    Let $\nabla_{n_0}(n) = g_0$
    Adaptive filter output:
        $\hat{y}(n) = \sum_{k=0}^{M} n_k(n) \nabla_{n_k}(n)$
    Output error:
        $e(n) = y(n) - \hat{y}(n)$
```

Filtered regressors:

```
    Let $g_M = 1$
    For $k = M, M-1, \cdots, 1$ do
        $\nabla_{q_k} = g_k x_{k-1}(n)$
        $g_{k-1} = g_k \cos q_k(n)$
    End {For}
```

Coefficient updates:

$$n_k(n+1) = n_k(n) + me(n)\nabla_{n_k}(n), \quad k = 0,1,\cdots,M$$
$$q_k(n+1) = q_k(n) + me(n)\nabla_{q_k}, \quad k = 1,2,\cdots,M$$

Test for instability:

```
For $k = 1,2,\cdots,M$ do
    If $|q_k(n+1)| > p/2$ set $q_k(n+1) = q_k(n)$
End {For}
```

Post filter computations:

```
Let $temp = -\hat{y}(n)$
For $k = M,\cdots,2,1$ do
    $\begin{bmatrix} temp \\ x_k\ (n+1) \end{bmatrix} = \begin{bmatrix} \cos q_k(n+1) & -\sin q_k(n+1) \\ \sin q_k(n+1) & \cos q_k(n+1) \end{bmatrix} \begin{bmatrix} temp \\ x_{k-1}(n) \end{bmatrix}$
End {For}
Let $x_0(n+1) = temp$
```

End Algorithm

The computational complexity of this algorithm is readily apparent. Implementation on a DSP processor is a difficult task, augmented by the need to find the sine and cosine of the rotation angles. Comparatively, the FIR LMS filter adaptation algorithm is essentially simple. Information on the LMS adaptation algorithm is contained in [12], [15], and [16]. The algorithm for an LMS filter is given below.

LMS Algorithm

Available at time n:

- Filter coefficients:

$$W_k(n), \quad k = 0,1,\cdots,M$$

- Adaptive filter states:

$$h_k(n-1), \quad k = 0,\cdots,M$$

- New data:

$$u(n) \quad \textit{Input Sample}$$
$$y(n) \quad \textit{reference Sample}$$

Adaptive filter computation:

```
Update filter states:

For $k = 1,2,\cdots,M$ do
        $h_k(n) = h_{k-1}(n-1)$
End {For}

$h_0(n) = u(n)$

Adaptive filter output:
        $\hat{y}(n) = \sum_{k=0}^{M} W_k(n) h_k(n)$
Output error:
        $e(n) = y(n) - \hat{y}(n)$
```

Coefficient updates:

$$W_k(n+1) = W_k(n) + m\, u(n) e(n)$$

End Algorithm

2.4 Modifying the system ID algorithm for use in the adaptive-Q approach

The majority of the adaptation described in the lattice algorithm above can be used to adapt the Q filter in the adaptive feedback control system. The main changes occur in the setup of the problem and the addition of some prefiltering. The system identification problem solved by the simplified partial gradient lattice algorithm can be described by the equation

$$e_k = T_{yu} u_k - Q u_k$$

where e_k is the output error, u_k is the input, and Q is the adaptive IIR lattice filter. This equation implies that adjusting the parameters of Q such that the error e_k is minimized results in Q modeling the plant described by T_{yu}.

From the discussion above, it was shown that the problem of adapting Q for the adaptive-Q control structure can be restated as a system identification problem. A copy of the resultant equation is shown below.

$$T_{yd}d_k = Q(-T_{ys}r_k) + y_k$$

where y_k is the output of the system, d_k is the disturbance, $-T_{ys}r_k$ is the estimation error, and Q is the adaptive lattice filter. In this case, the parameters of Q are adjusted using a gradient descent method such that the output of the system y_k is minimized. This is possible since the estimate, $\hat{y}_k$, is made using information about the plant only, so the output estimation error, $r_k = y_k - \hat{y}_k$, contains information that is coherent with the disturbance input.

Finding a relationship between these two equations allows the use of the simplified partial gradient lattice algorithm to adjust the parameters of Q. The equations that define this relationship are

$$\begin{aligned} e_k &= T_{yd}d_k \\ T_{yu}u_k &= y_k \\ u_k &= -T_{ys}r_k \end{aligned}$$

y_k is directly measured and r_k is the estimation error. From these equations the changes to the algorithm can be inferred. These changes are

1. Since the output needs to be driven to zero, the error e_k is changed to the output y_k.
2. The input to the Q filter is r_k instead of u_k.
3. To generate the n gradients, a second lattice filter Q_2 is added with $-T_{ys}r_k$ as its input.

4. The input to the post filter is the new estimate of the output, described by $Q_2 T_{ys} r_k$. The output of the post filter can still be used to calculate the gradients for the q_k parameters.

Figure 6 illustrates the setup of the adaptive lattice algorithm after the above changes have been made.

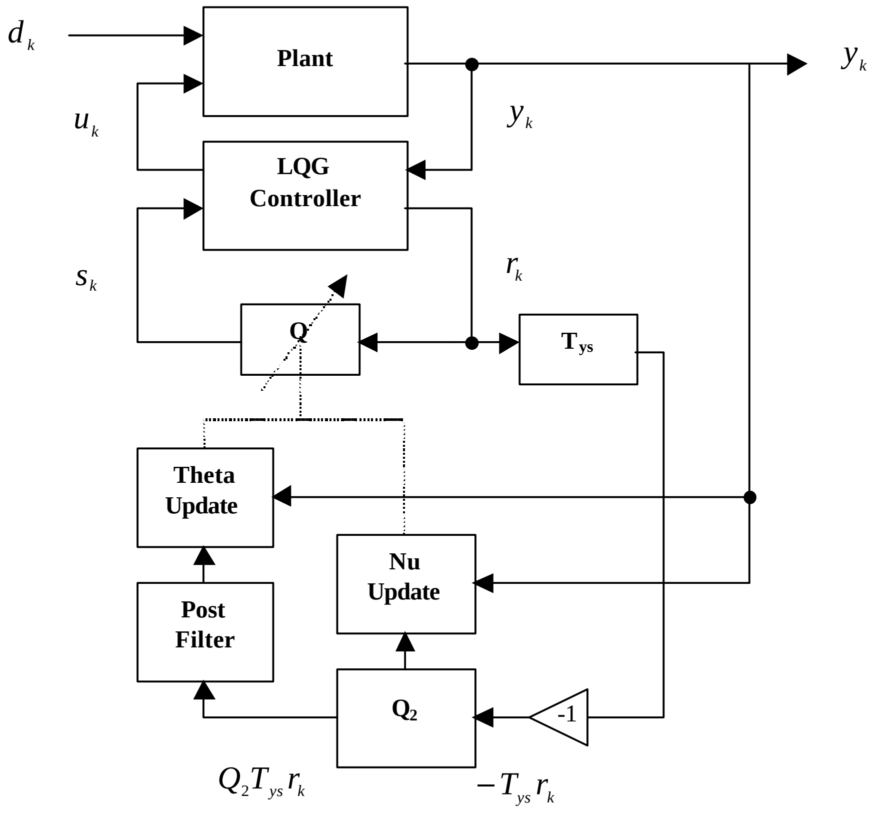


Figure 6 – Illustration of modified lattice algorithm

The revised algorithm that incorporates these changes is listed below.

Modified Partial Gradient Lattice Algorithm for Adaptive-Q System

Available at time n:

- Filter coefficients:

$$n_k(n), \quad k = 0,1,\cdots,M$$
$$q_k(n), \quad k = 1,2,\cdots,M$$

- Primary adaptive filter states:

$$Q_k(n-1), \quad k = 0,1,\cdots,M-1$$

  The variable $Q_M(n)$ will be computed below, but need not be stored.

- Secondary adaptive filter states:

$$\nabla_{n_k}(n-1), \quad k = 0,1,\cdots,M-1$$

  The variable $\nabla_{n_M}(n)$ will be computed below, but need not be stored.

- Post filter states:

$$x_k(n), \quad k = 0,1,\cdots,M-1$$

  The variable $x_M(n+1)$ will be computed below, but need not be stored.

- New data:

$$r\_filt(n) \quad \textit{Estimation Error (filtered through } T_{yu}\textit{)}$$
$$r(n) \quad \textit{Estimation Error (unfiltered)}$$
$$y(n) \quad \textit{System Output}$$

Primary adaptive filter computation:

```
Let $g_M = r(n)$
For $k = M, M-1, \cdots, 1$ do
    $\begin{bmatrix} g_{k-1} \\ Q_k(n) \end{bmatrix} = \begin{bmatrix} \cos q_k(n) & -\sin q_k(n) \\ \sin q_k(n) & \cos q_k(n) \end{bmatrix} \begin{bmatrix} g_k \\ Q_{k-1}(n-1) \end{bmatrix}$
End {For}
Let $Q_0(n) = g_0$
```

Filter output:

$$s(n) = \sum_{k=0}^{M} n_k(n) Q_k(n)$$

Secondary adaptive filter computation:

```
Let $g_M = r\_filt(n)$
For $k = M, M-1, \cdots, 1$ do
    $\begin{bmatrix} g_{k-1} \\ \nabla_{n_k}(n) \end{bmatrix} = \begin{bmatrix} \cos q_k(n) & -\sin q_k(n) \\ \sin q_k(n) & \cos q_k(n) \end{bmatrix} \begin{bmatrix} g_k \\ \nabla_{n_{k-1}}(n-1) \end{bmatrix}$
    End {For}
    Let $\nabla_{n_0}(n) = g_0$

    Filter output:
                    $s\_filt(n) = \sum_{k=0}^{M} n_k(n)\nabla_{n_k}(n)$
    Output error:
                    $e(n) = y(n)$
```

Filtered regressors:

```
    Let $g_M = 1$
    For $k = M, M-1, \cdots, 1$ do
                    $\nabla_{q_k} = g_k x_{k-1}(n)$
                    $g_{k-1} = g_k \cos q_k(n)$
    End {For}
```

Coefficient updates:

```
            $n_k(n+1) = n_k(n) + me(n)\nabla_{n_k}(n), \quad k = 0,1,\cdots,M$
            $q_k(n+1) = q_k(n) + me(n)\nabla_{q_k}, \quad k = 1,2,\cdots,M$
```

Test for instability:

```
    For $k = 1,2,\cdots,M$ do
        If $|q_k(n+1)| > p/2$ set $q_k(n+1) = q_k(n)$
    End {For}
```

Post filter computations:

```
    Let $temp = -s\_filt(n)$
    For $k = M,\cdots,2,1$ do
            $\begin{bmatrix} temp \\ x_k\ (n+1) \end{bmatrix} = \begin{bmatrix} \cos q_k(n+1) & -\sin q_k(n+1) \\ \sin q_k(n+1) & \cos q_k(n+1) \end{bmatrix} \begin{bmatrix} temp \\ x_{k-1}(n) \end{bmatrix}$
    End {For}
    Let $x_0(n+1) = temp$
```

End Algorithm

Again, as stated in section 2.3, the lattice algorithm is considerably more complex than the LMS algorithm. Since a digital signal processing chip is capable of implementing FIR filters with ease, it is a trivial mater to implement the LMS algorithm. The lattice algorithm, however, is not so fortunate. The lattice algorithm requires three lattice filters and the adaptation of both nu and theta parameters. Additionally, the sine and cosine of theta is required which adds a considerable amount of programming complexity. As a result, the performance of the lattice filter needs to be considerably better than the LMS filter for it to be an improvement.

Chapter 3 Experimental Setup

This section contains a description of the experimental setup used to validate the theoretical approach.

3.1 General System Overview

The experimental results were obtained using a cantilever beam, instrumented with three piezoceramic devices. One was used as a sensor and the other two as the disturbance and control input actuators. A Digital Signal Processing (DSP) board implements the controller design in mixed assembly and C code. Figure 7 below shows a block diagram of the setup.

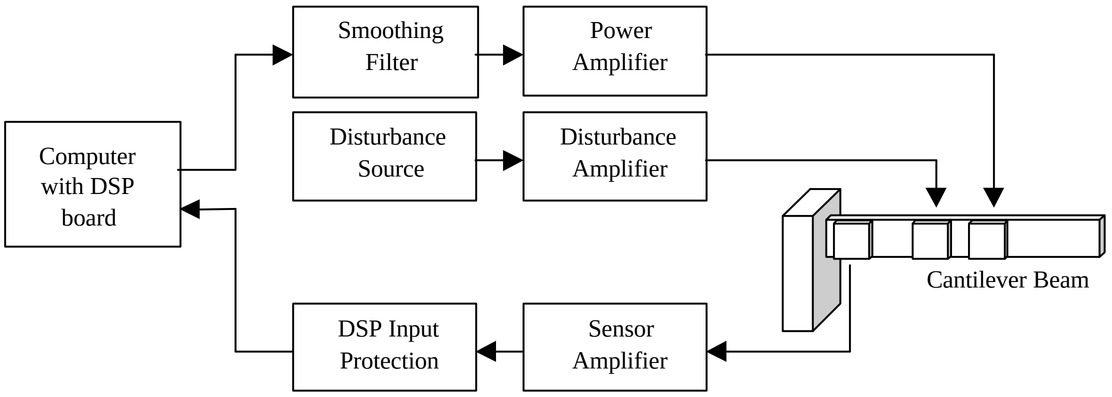


Figure 7 – Experimental setup

3.2 Beam

The beam is made of a piece of aluminum held rigid at one end. Three piezoceramic elements were attached to the beam for use as the feedback sensor, disturbance actuator, and control actuator. For best performance, the actuators were adhered to the beam in areas of high strain [9]. The location of the piezoceramic elements and size of the beam are shown in Figure 8.

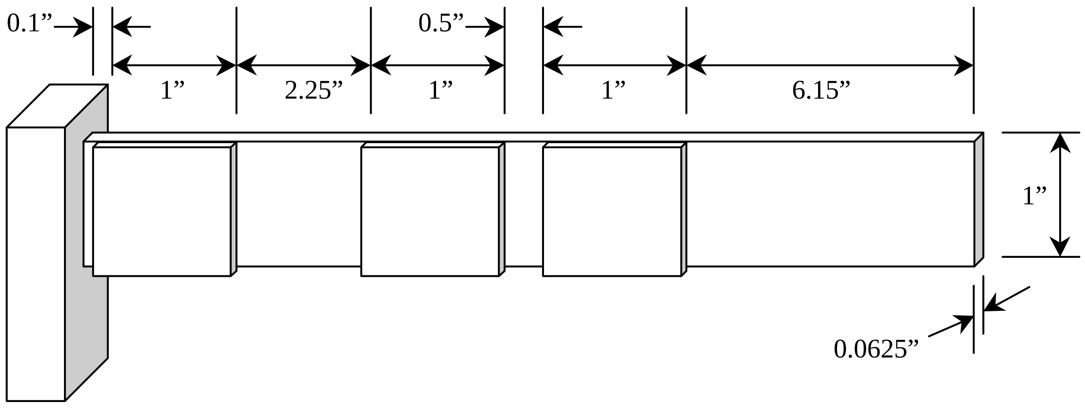


Figure 8 – Cantilever beam

3.3 Sensor Amplifier

The sensor amplifier is set up in a high pass filter configuration. An LF411 (JFET) operational amplifier was chosen for this circuit because of its high input impedance, which is required when measuring voltage from a piezoceramic. The amplifier configuration is chosen as a high pass circuit, with cutoff frequency at 0.07 Hz, to counter the DC shifts that can occur in the sensor signal. This is mainly due to the extremely high sensitivity of the piezoceramic devices to low frequencies (< 1 Hz). Additionally, it is desired to measure small disturbances in the system, so the amplifier was set with a gain of 10. The circuit diagram for the sensor amplifier is shown in Figure 9.

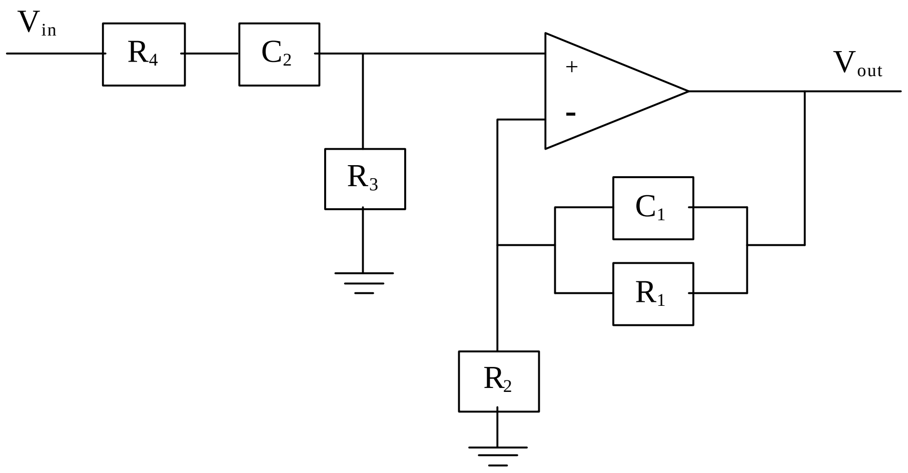


Figure 9 – Sensor amplifier

The transfer function associated with this configuration is

$$\frac{V_{out}}{V_{in}} = \frac{s^2 C_1 C_2 R_1 R_2 R_3 + s C_2 R_3 (R_1 + R_2)}{s^2 C_1 C_2 R_1 R_2 (R_3 + R_4) + s(C_2 R_2 (R_3 + R_4) + C_1 R_1 R_2) + R_2}$$

The component values are

$$\begin{aligned} C_1 &= 0.472\ nF \\ C_2 &= 0.223\ mF \\ R_1 &= 10.0\ K\Omega \\ R_2 &= 1.0\ K\Omega \\ R_3 &= 10.0\ M\Omega \\ R_4 &= 82.0\ K\Omega \end{aligned}$$

3.4 Smoothing Filter

A smoothing filter is required to smooth out the high frequency components from the DSP digital outputs. The filter is set up using a Sallen-Key low pass circuit configuration. The gain is set to 1 V/V and the cutoff frequency is set to 500 Hz. To keep the final order of the system as small as possible, the filter is only second order.

The circuit diagram for the smoothing filter is shown in Figure 10.

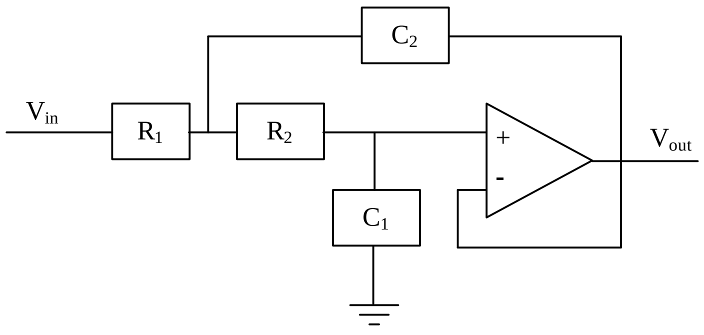


Figure 10 – Smoothing filter

The transfer function associated with this configuration is

$$\frac{V_{out}}{V_{in}} = \frac{1}{s^2 R_1 R_2 C_1 C_2 + sC_1(R_1 + R_2) + 1}$$

The component values are

$$\begin{aligned} R_1 &= 1.0\ M\Omega \\ R_2 &= 1.0\ M\Omega \\ C_1 &= 220\ pF \\ C_2 &= 470\ pF \end{aligned}$$

3.5 Power Amplifier

A power amplifier was needed to drive the piezoceramic devices to higher voltage levels than a standard 741 operational amplifier is capable of supplying. The power amplifier is a PB58 power booster from APEX Microtechnology. The supply voltages for the device were set at ±30V and the gain for the system is approximately 10.

The power booster is not capable of driving the capacitive load of the piezoceramic actuator without compensation. To compensate the amplifier, the resistor isolation technique described in APEX application note 24 was used. This technique moves the lowest frequency pole in the open loop and add a zero near this pole such that the modified open loop passes through the zero dB point at a slope of 20 dB/decade instead of 40 dB/decade. To set the value of R_{ISO}, first find the first open loop pole given by the equation

$$fp = \frac{1}{2pC_L R_0}$$

where C_L is the capacitance of the load and R_0 is the output resistance of the PB58. For the cantilever beam system, the parameters are

$$\begin{aligned} C_L &\approx 15mF \\ R_0 &= 35\Omega \end{aligned}$$

Then choose an R_{ISO} such that the modified pole frequency is within one decade of the new zero frequency, and the open loop response passes through the zero dB point at 20 dB/decade. The equations for the modified pole and added zero frequencies are

$$fp_{\mathrm{mod}} = \frac{1}{2pC_L(R_0 + R_{ISO})}$$

$$fz = \frac{1}{2pC_L R_{ISO}}$$

The value for R_{ISO} was chosen to be 16Ω. All other component values are given below. The circuit configuration with R_{ISO} in place is shown below in Figure 11.

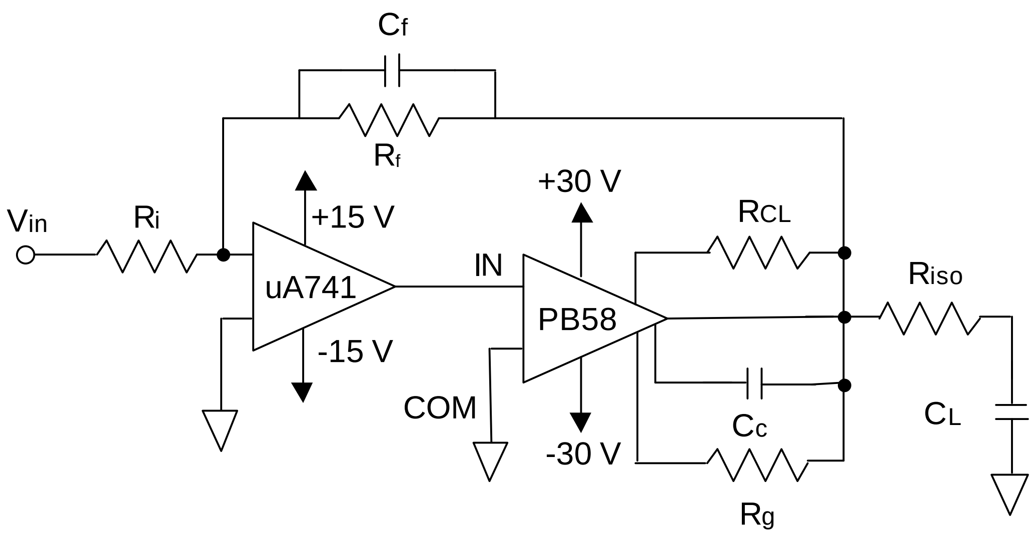


Figure 11 – Power amplifier

Gain Set:

$$R_G = [(A_v - 1) * 3.1K] - 6.2K$$

$$A_v = \frac{R_G + 6.2K}{3.1K} + 1$$

The component values are

$$\begin{aligned}
R_I &= 6.2\ K\Omega \\
R_F &= 60\ K\Omega \\
R_G &= 0\ \Omega \\
R_{CL} &= 2\ \Omega \\
R_{ISO} &= 16\ \Omega \\
C_F &= 10\,pF \\
C_c &= 10\,pF
\end{aligned}$$

3.6 Disturbance Amplifier

The disturbance amplifier is a 741 operational amplifier set up in a standard inverting configuration. The gain was set at 10 V/V allowing the amplifier to act as a buffer and driver for the piezoceramic actuator.

3.7 DSP Input Protection

The input to the DSP card is limited to ± 3 volts, so a voltage limiter was needed to prevent overdriving the inputs. The input protection is made using the natural diode voltage drops associated with a 1N7001 diode. The circuit diagram is shown below in Figure 12. This circuit clamps the voltage between ± 2.5 volts.

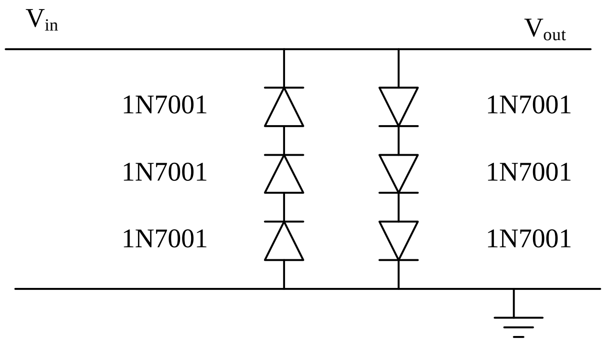


Figure 12 – DSP input protection

3.8 DSP Signal Processing Board

The DSP board used was manufactured by Spectrum Signal Processing [17]. It uses a Texas Instruments TMS320C31 32-bit floating-point microprocessor. The board has two analog input and two analog output channels and is capable of I/O speed well in excess of the required 2000 Hz used in the implementation of the controller. The DSP board plugs into an ISA bus on a standard personal computer and uses software written by Spectrum as a processor interface.

One item of major concern was discovered during operation of the Spectrum board. The system has a four sample latency output delay inherent to the system. These delays were confirmed by Spectrum Signal Processing, but are not mentioned in the accompanying manuals. Since the delays were present during the system ID, they were accounted for in the control system.

3.9 Other Equipment

Several pieces of additional test equipment were used during the implementation of the controller. These include a Tektronix TDS210 Digital Oscillosope, a Hewlett Packard 35665A Dynamic Signal Analyzer, two Elenco XP-656 power supplies, an Elenco XP-760 power supply, and a Tektronix FG502 Function Generator.

Chapter 4 System Identification

The adaptive control system requires a model of the control to output transfer function to create a fixed LQG controller that generates the neutralization loop. A copy of the system is also needed to prefilter the input data for the secondary lattice filter. The required model needs to contain the magnitude and phase information from the control input to the system output. Additionally, a model of the transfer function from the disturbance input to the system output is needed so that a proper simulation of the experimental setup can be conducted on the computer. The disturbance model is not used in the controller design because in a realistic application, information about how the disturbance enters the system is unknown. The best way to get these system models is to use standard system identification techniques.

The process of system identification requires the measurement of the system's output response to a white noise signal entered at the input. To generate this noise signal, a pseudo random binary signal (PRBS) was generated in assembly language using the method described in Horowitz and Hill [10]. This PRBS signal is sent to the control actuator through the signal conditioning electronics described in section 3.1. This is important to consider so that the model contains all magnitude and phase information associated with the smoothing filter, cantilever beam, signal amplifier, and DSP input protection circuitry.

The gathering of this input-output data was performed using the sysid.asm program. A listing for this program is found in Appendix A. The input-output information is then converted from binary to ASCII using a second program written in C-code and was named Bin2asci. A listing of Bin2asci is found in Appendix A. Conversion to ASCII is required so that the rest of the system identification can be generated using a series of Matlab m-files.

A listing of each of the m-files required for the system identification is found in Appendix A. They are named: etfest.m, ident.m, rduce.m, svmod.m, and savtf.m.

etfest.m computes the empirical transfer function estimate (ETFE) for the input-output data. The ETFE is a ratio of the discrete Fourier transform (DFT) of the output data to the DFT of the input data. A Hamming window is used to smooth the data. Etfest.m relies on the Matlab m-file etfe.m to calculate the ETFE of the input-output data. ident.m computes a least squares estimate of the input-output data. A 34th order model was chosen to fit all of the nuances associated with the data. A visual comparison of the ETFE and the least squares fit was made to determine the quality of the fit. rduce.m reduces the order of the data using a balanced state space model routine in conjunction with a model order reduction routine. The model was reduced to 28th order because it could capture all of the system modes accurately. The magnitude and phase response of the least squares fit and the ETFE are plotted together in Figure 13 and Figure 14 below.

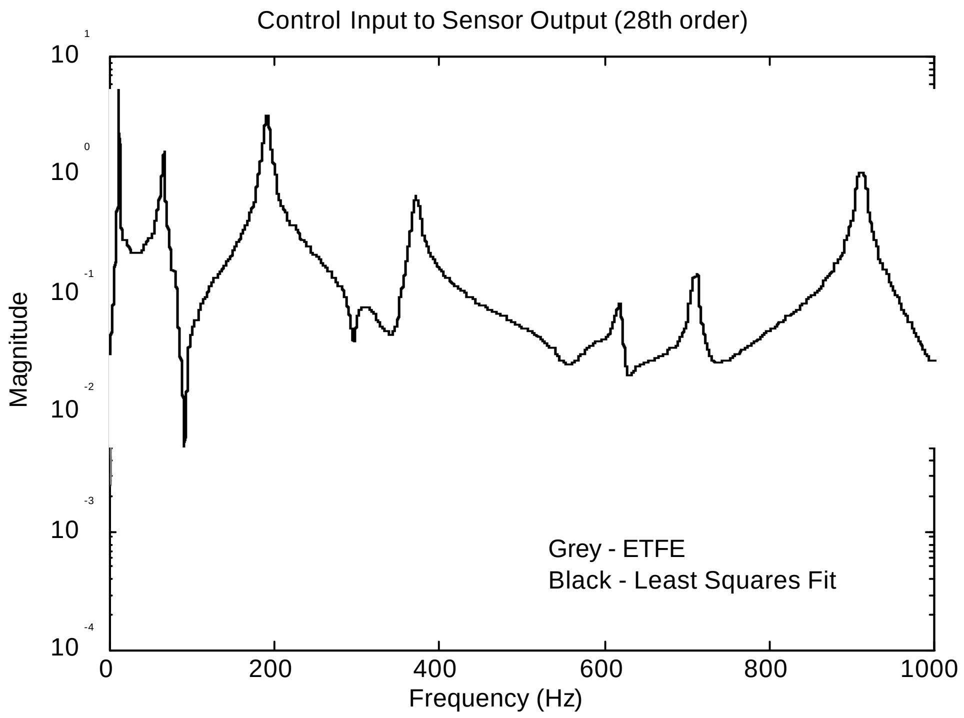


Figure 13 – Magnitude comparison between ETFE and 28th order least squares fit

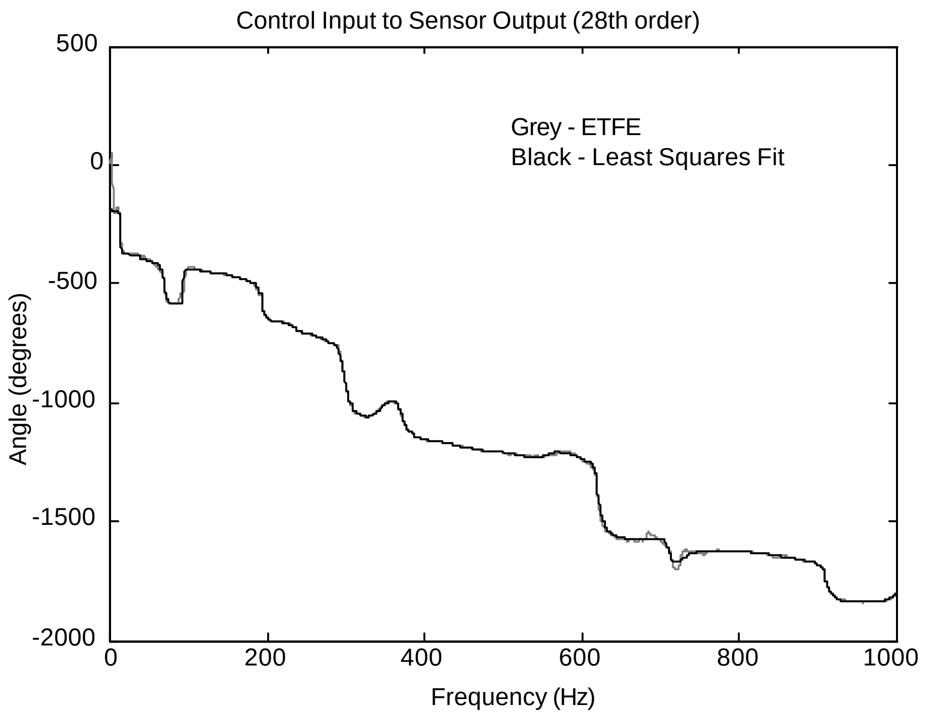


Figure 14 – Phase comparison between ETFE and 28th order least squares fit

It is important to note that modes between 600 and 800 hertz are high frequency modes aliased down to lower frequencies. These modes do not exist in the continuous time system, but do exist when the system is driven by a zero order hold.

svmod.m saves the system model to disk in state space form. savtf.m uses the state space model generated by the system identification routines to compute a fixed LQG controller for the system. The controller information is saved in transfer function form so that the data can be filtered efficiently by the DSP chip. The LQG controller was designed to create the neutralization loop for the adaptive IIR filter and does very little rejection of the disturbance in the beam. With this LQG controller, the disturbance rejection performed by the adaptive IIR filter can be readily seen.

Additionally, the model of the system that was generated from the system ID had a different structure than the model used in Chapter 2 for the theoretical discussion. In the theoretical case, the disturbance and control signals entered into a system described by one set of state equations. This is illustrated in Figure 15.

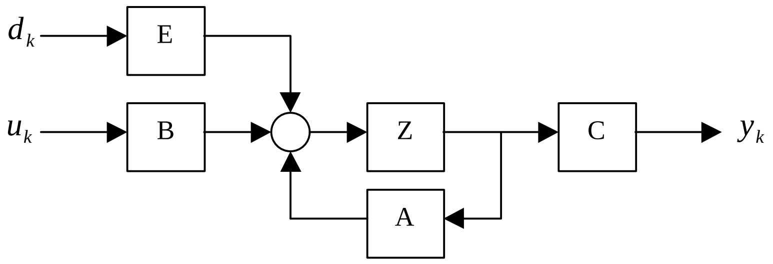


Figure 15 – Model structure used in theoretical discussion

For the model created by the system ID, the disturbance and control state equations are modeled separately then summed to produce the output, see Figure 16.

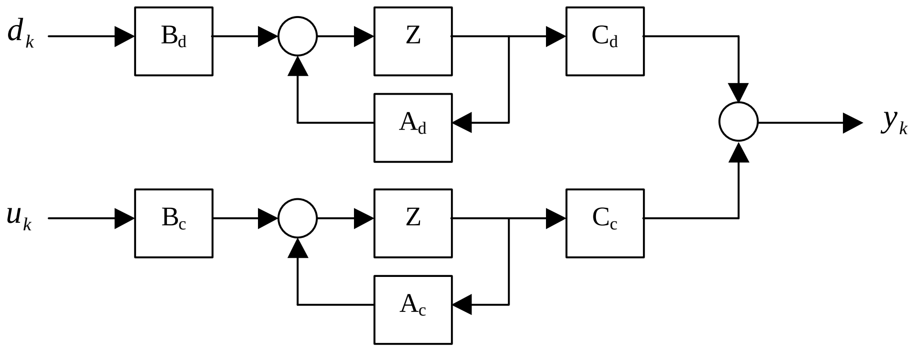


Figure 16 – Model structure created by system identification

Both systems have an identical response, so this difference will not affect the results.

Chapter 5 LQG Fixed Feedback Controller

The LQG controller is required to generate the neutralization loop for the adaptive-Q technique and for rejecting any transient disturbances in the system. The LQG controller is designed to give a minimal amount of damping so that the effects of the adaptive filter on the system are easy to see.

The LQG controller design was performed in Matlab using the LQR and LQE commands. Since the control system requires no spectral information or knowledge of how the disturbance enters the plant, the disturbance is modeled as white noise acting on each mode of the system, independently. The process noise covariance is set to 1 and the measurement noise covariance is set to 1e-4. The cost function is defined in the standard LQG form

$$e_k^T e_k = x_k^T Q x_k + u_k^T R u_k$$

The state weighting is $Q = C^T C$ where C is the output matrix of the system model on which the control system design is based. The control weighting for the controller, R, is set to 0.96 to provide damping of only the larger modes in the system. The closed loop LQG disturbance to output simulation response and the open loop ETFE spectral response from disturbance input to system output are plotted together in Figure 17.

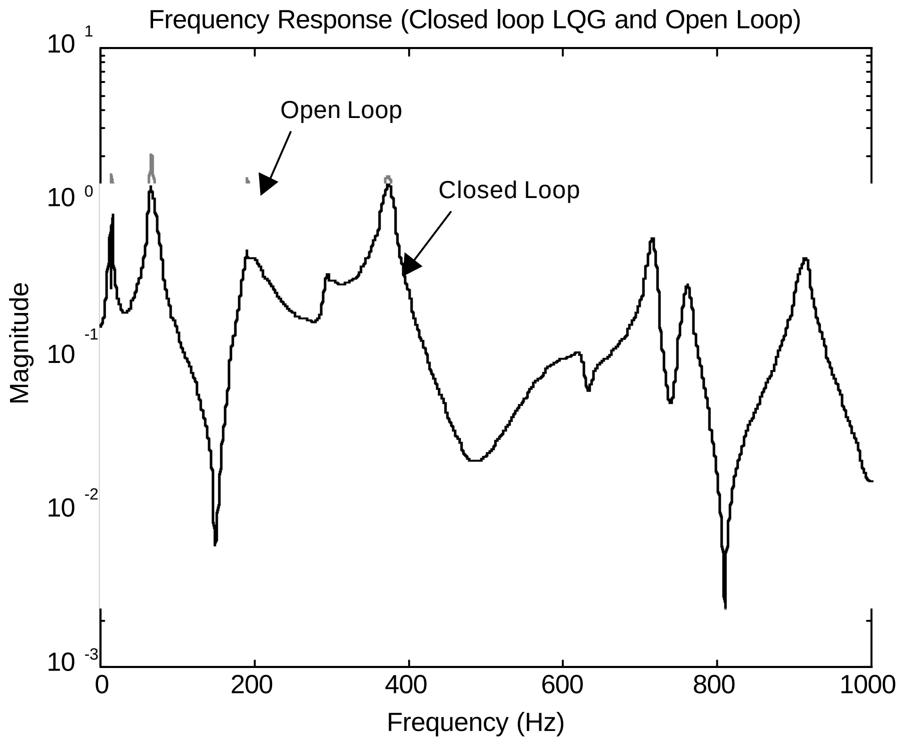


Figure 17 – Open loop and closed loop frequency response of system

5.1 Closed Loop System Model

The closed loop system with the plant modeled from the system ID and the LQG fixed feedback controller in place is shown in Figure 18. The location of the adaptive filter is also shown. This diagram is useful when parsing through the Matlab m-files and assembly code.

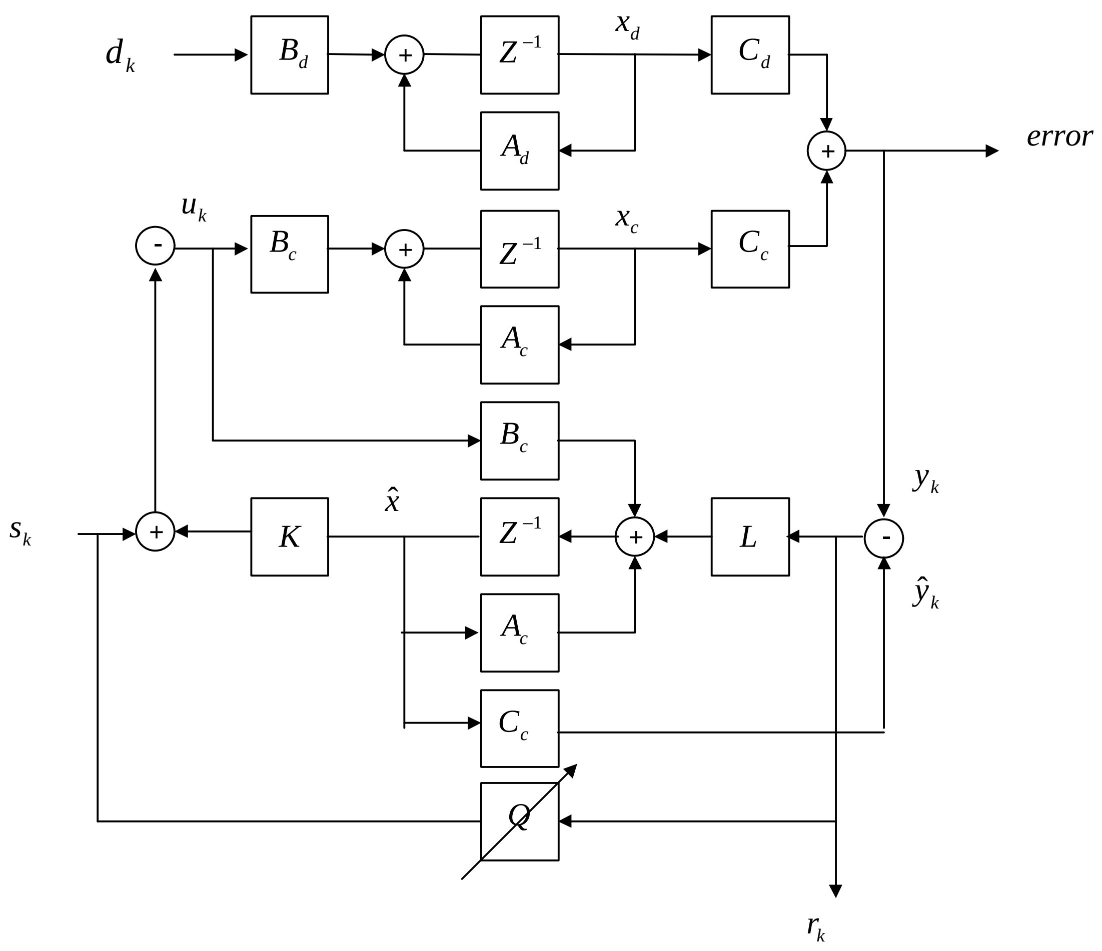


Figure 18 – System block diagram

Chapter 6 Implementation of Controller

The implementation of the digital controller in the system as described above was performed in mixed Assembly and C-code. Several considerations had to be taken into account when implementing the controller.   These include understanding the delays in the system, making sure the processing time is less than the sampling speed, and the effects of the digital controller at frequencies nearing the Nyquist rate. A brief description of each of these effects is discussed in sections 5.1 through 5.3 below.

The Assembly and C-code written for the controller that uses an LMS FIR filter can be found in Appendix B. The code pertaining to the controller using IIR lattice filters can also be found in Appendix B. These listings include the controller code, as well as makefiles, header files, and .cmd files needed for processor implementation.

6.1 System Delays

Delay is important in any implementation of a control system. Added delay can adversely affect the phase of a system to the point where stability can no longer be maintained. Additionally, it is important to have an exact model of the beam for the neutralization loop to work properly. Any delay in the actual system not included in the model of the system will result in erratic behavior of the adapting filter.

Delay in the system occurs because the output cannot be calculated before the input is measured. As a result, the calculations resulting from an input sample will not be sent out until the next sampling instant. This delay needs to be accounted for in the fixed LQG controller only. The delay does not need to be accounted for in the adaptive filter since any delay is incorporated into the filter itself.

To account for the delay in the fixed controller, estimates of the future system states need to be based on the current delayed states, the previous output, and the current delayed input. A mathematical description for this controller is

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + Bu_k + AL(y_k - \hat{y}_k)$$
$$u_{k+1|k} = -K\hat{x}_{k+1|k}$$

$x_{k+1|k}$ denotes the value of x at time $k+1$ based on information obtained at time k.

6.2 Processing Time

Implementing the control system requires a computationally intensive program. To make sure the system is capable of computing the output of the controller between the sampling instants, the controller must be programmed in Assembly code. Assembly code is perfectly suited for the implementation of the many digital filters required to make the fixed controller and provides a way to streamline the gradient descent algorithm required for the lattice filters.

6.3 Frequencies near the Nyquist rate

When running the fixed controller on the system, the response of the system did not match the predicted response from the computer. The magnitude of the modes near the Nyquist rate of 1 kHz, specifically the mode at 900 Hz, tended to increase by one or two decibels when the fixed controller was applied. It is currently unknown why the magnitude of the modes increase when they should be decreasing.

Chapter 7 Optimal Controller

The performance of the optimal controller is used as a baseline comparison to the experimental and simulated controllers' performances. It is assumed that the optimal performance of the system to a harmonic disturbance is the noise floor in the experimental case and zero for simulation. In the narrow and wide band cases, the optimal performance is referenced to the optimal disturbance rejection controller described below.

The optimal controller consists of augmenting the information of how the disturbance enters the plant and the disturbance spectrum to the model and finding the best performance using an LQG disturbance rejection controller. In this way, the estimator has exact knowledge of the disturbance spectrum allowing the control effort to compensate in the best possible manner.

The current state representation for the system is:

$$\begin{aligned} x_{k+1} &= A x_k + B u_k + E\, d_k \\ y_k &= C x_k \end{aligned}$$

To create the optimal controller, it is assumed that the controller knows the spectrum of the disturbance signal and how it enters the plant. The disturbance spectrum is modeled as:

$$\begin{aligned} \bar{x}_{k+1} &= A_d \bar{x}_k + B_d w_k \\ d_k &= C_d \bar{x}_k \end{aligned}$$

$\bar{x}_k$ are the states of the disturbance input and w_k is white noise.

With this in mind, the state equations for the optimal controller can be formed as follows:

$$\begin{aligned} \tilde{x}_{k+1} &= A_{aug} \tilde{x}_k + B_{aug} u_k + L(y_k - \hat{y}_k) \\ u_k &= -K \tilde{x}_k \end{aligned}$$

$\tilde{x}$ is the optimal estimate of the states of the plant and disturbance, $A_{aug}, B_{aug}, C_{aug}$, and E_{aug} are the state matrices augmented with the disturbance description as follows:

$$A_{aug} = \begin{bmatrix} A & EC_d \\ 0 & A_d \end{bmatrix}$$
$$B_{aug} = \begin{bmatrix} B \\ 0 \end{bmatrix}$$
$$C_{aug} = \begin{bmatrix} C & 0 \end{bmatrix}$$
$$E_{aug} = \begin{bmatrix} 0 \\ B_d \end{bmatrix}$$

The LQG controller that has been augmented with the disturbance states is then placed on the original system plant. The state equations for the plant with the optimal controller are:

$$\begin{bmatrix} x_{k+1} \\ \tilde{x}_{k+1} \end{bmatrix} = \begin{bmatrix} A & -BK \\ LC & A_{aug} - B_{aug}K - LC_{aug} \end{bmatrix} \begin{bmatrix} x_k \\ \tilde{x}_k \end{bmatrix} + \begin{bmatrix} -B \\ -B_{aug} \end{bmatrix} s_k + \begin{bmatrix} E \\ 0 \end{bmatrix} d_k$$
$$y_k = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x_k \\ \tilde{x}_k \end{bmatrix}$$

s_k is the new control input where the adaptive filter enters the system, d_k is the colored disturbance input, and y_k is the system output.

When designing the optimal controller, a process noise covariance of 1, a measurement noise covariance of 1e-4, and a control penalty of 1e-8 was used. To find the control penalty, a start value of 1e-2 was assumed and was continuously adjusted down until no further performance improvement was noticed.

The narrow band filter is a fifth order Butterworth filter with a center frequency of 189 Hz and a 20 Hz bandwidth. It is placed at the largest mode in the system so that the disturbance rejection performance is easier to measure.

The wide band filter is a fifth order Butterworth filter with a center frequency of 300 Hz and a bandwidth of 250 Hz. It is considered a wide band filter because it covers two of the largest modes in the system and is about 1/3rd the size of the zero to 1 kHz spectrum being considered. These modes are 189 Hz and 367 Hz.

Calculation of the optimal controller was performed in Matlab. The code for the optimal controller can be found in Appendix C, and is named optcntrl.m. Figures showing the performance of the optimal controller are place along with the figures of the simulated LMS and lattice filter controller performances in Chapter 8.

Chapter 8 Simulations vs. Optimal Controller

This section compares the simulations of the controller using an LMS algorithm, the controller using a lattice filter algorithm, and the optimal controller. Generation of the optimal controller is described in Chapter 7. The Matlab M-files used for these simulations are given in Appendix D.

8.1 Harmonic Simulation

For the harmonic simulation, a 1 V_{peak} sinusoid with a frequency of 189 Hz was chosen for the disturbance. Note that this frequency corresponds to one of the strongest modes on the beam. The damping provided by the fixed feedback controller reduced the disturbance down to 0.5 V_{peak}. Figure 19 below shows the closed loop response with an adaptive lattice filter and the closed loop response with an adaptive LMS filter. In both cases, 3rd order filters were used and the step size for the two systems were identical.

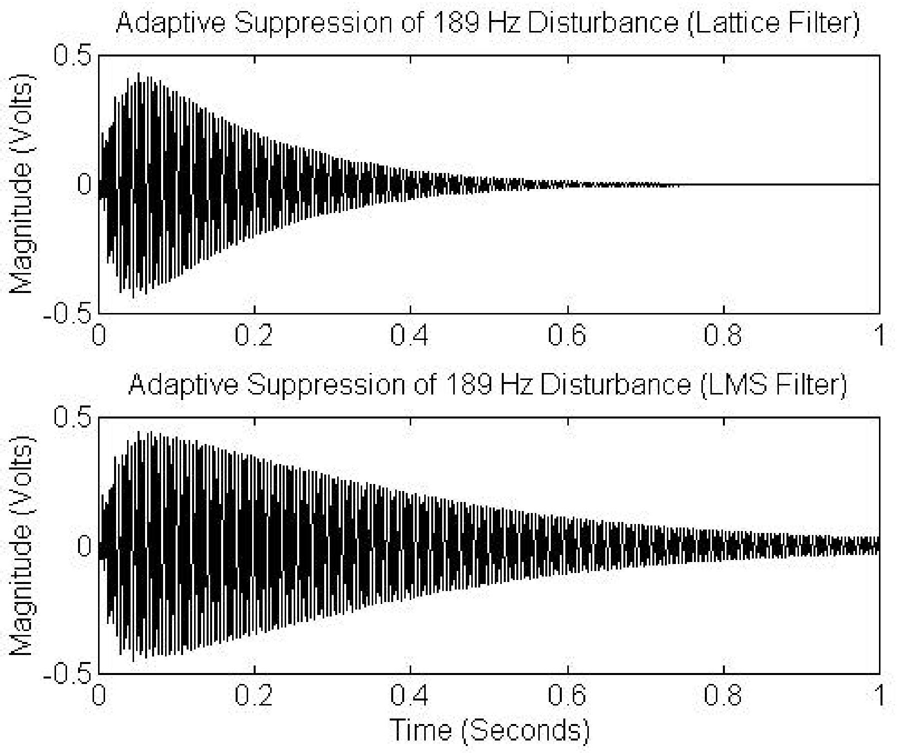


Figure 19 – Response of IIR and FIR adaptive-Q controller to 189 Hz harmonic disturbance

Notice that both systems damp the disturbance down to zero volts. Since the lattice filter has both poles and zeros to adapt, the harmonic disturbance was damped out faster.

Trying to get the LMS filter to perform as well as the lattice filter required an increased order for the LMS filter. It takes a seventh order LMS filter to get the system to damp the sinusoid as well as the third order lattice filter.

8.2 Narrow Band Simulation

The narrow band filter is a second order discrete time Butterworth filter centered at 189 Hz and has a bandwidth of 60 Hz. A pseudo random binary signal was generated and passed through this filter to generate the narrow band input disturbance. To keep the comparisons among the various controllers as similar as possible, the same random number generator seed was used. The seed was set to 931916785 in Matlab and the normal random generator RANDN was used to create the pseudo random binary sequence. Additionally, a random measurement noise of 0.01 $V_{p\text{-}p}$ was added to the signal.

The spectral performance of the optimal controller to the disturbance input is shown in Figure 20 below.

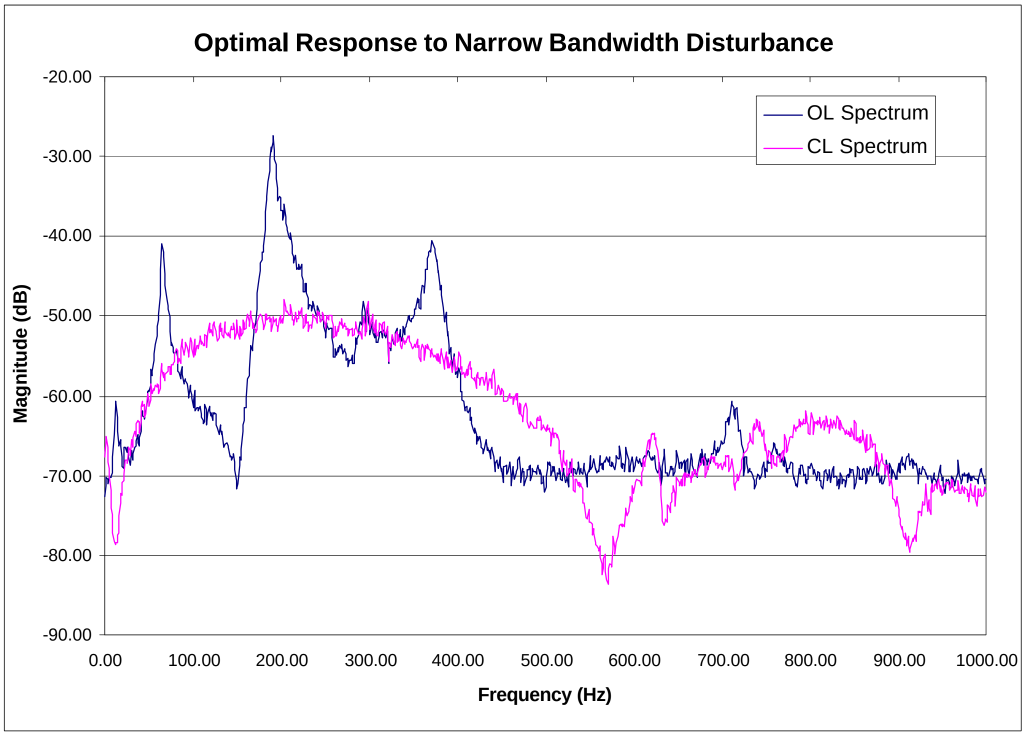


Figure 20 – Matlab simulated performance of the optimal controller to a narrow band disturbance centered at 189 Hz

By looking at the reduction in the magnitude at 189 Hz, it is determined that the optimal disturbance rejection controller is capable of reducing the disturbance by 23 dB. This is twenty seven times smaller than the open loop response.

The adaptive algorithm using an LMS filter was subjected to the same narrow band disturbance. Its step size parameter, m , was set to 5 and the filter order was adjusted to 7th order. The initial run of the system is shown in Figure 21.

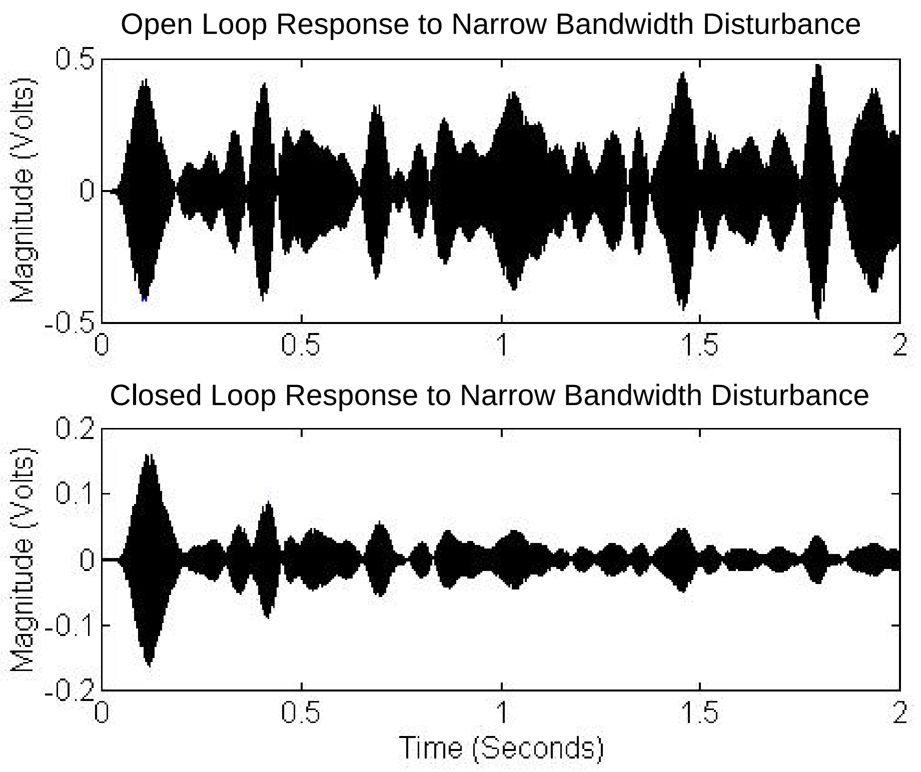


Figure 21 - Initial adaptation of the LMS algorithm to a narrow bandwidth disturbance centered at 189 Hz

The LMS algorithm takes about 2 seconds to converge to near its steady-state value. The algorithm was left to fully converge for one minute. A comparison between the open and closed loop spectral response of the LMS algorithm after this one minute of convergence is shown in Figure 22 below.

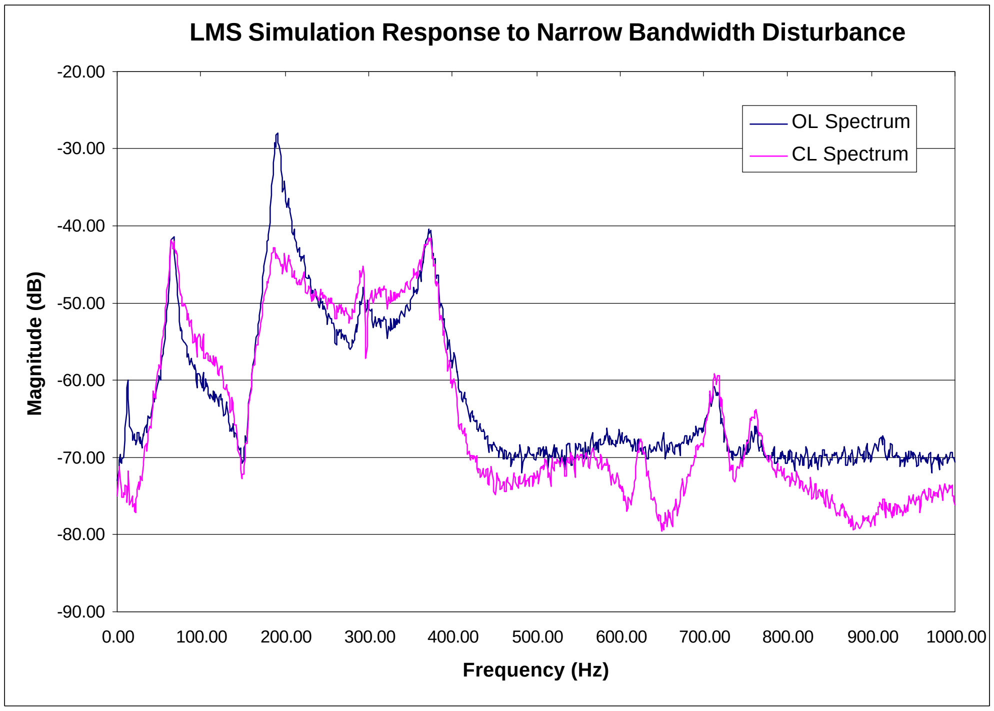


Figure 22 – Matlab simulated response of the LMS algorithm to the narrow band disturbance input after one minute of convergence.

The LMS algorithm reduced the disturbance in the system from 0.5 V_{peak} to about 0.023 V_{peak} after one minute. This is close to the result produced by the optimal controller and corresponds to a reduction of 15 dB as seen in Figure 22. This is twenty three times smaller than the open loop response.

The adaptive lattice filter algorithm was subjected to the same disturbance as the LMS algorithm and the optimal controller. The order of the lattice filter was set to 3rd order and the step size parameters for the n and q parameters were set to 5 and 0.1 respectively. The initial run of the lattice filter algorithm is shown in Figure 23.

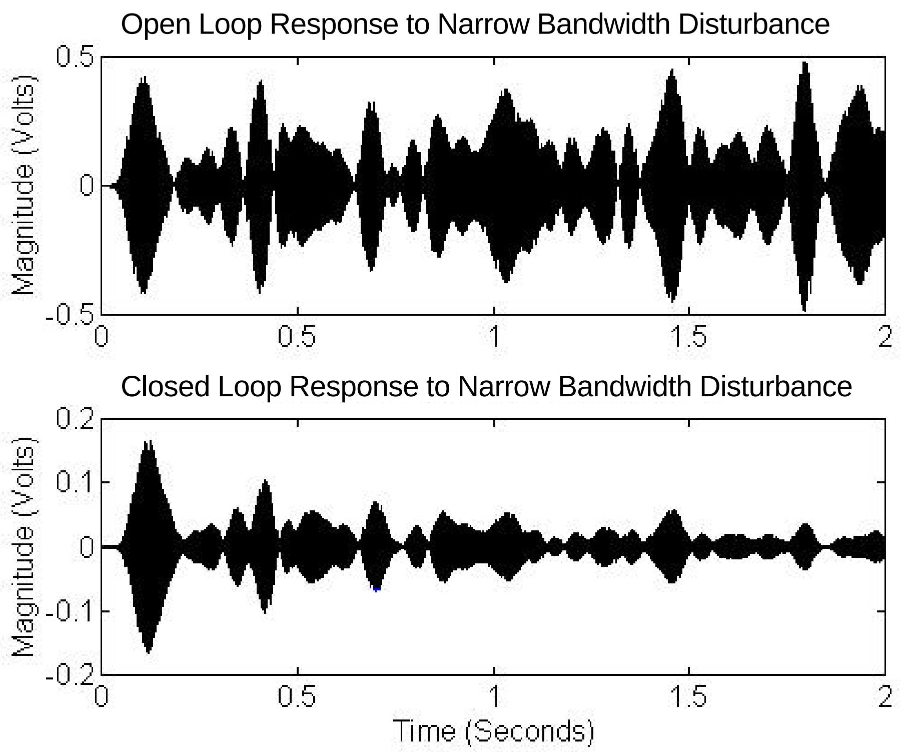


Figure 23 - Initial adaptation of the lattice filter algorithm to a narrow bandwidth disturbance centered at 189 Hz.

Like the LMS algorithm, the lattice filter algorithm took about 2 seconds to converge to near its steady-state value. The filter was left to converge for one minute so that a comparison could be made to the LMS algorithm. The open and closed loop spectral response of the lattice filter algorithm after one minute of convergence is shown in Figure 24.

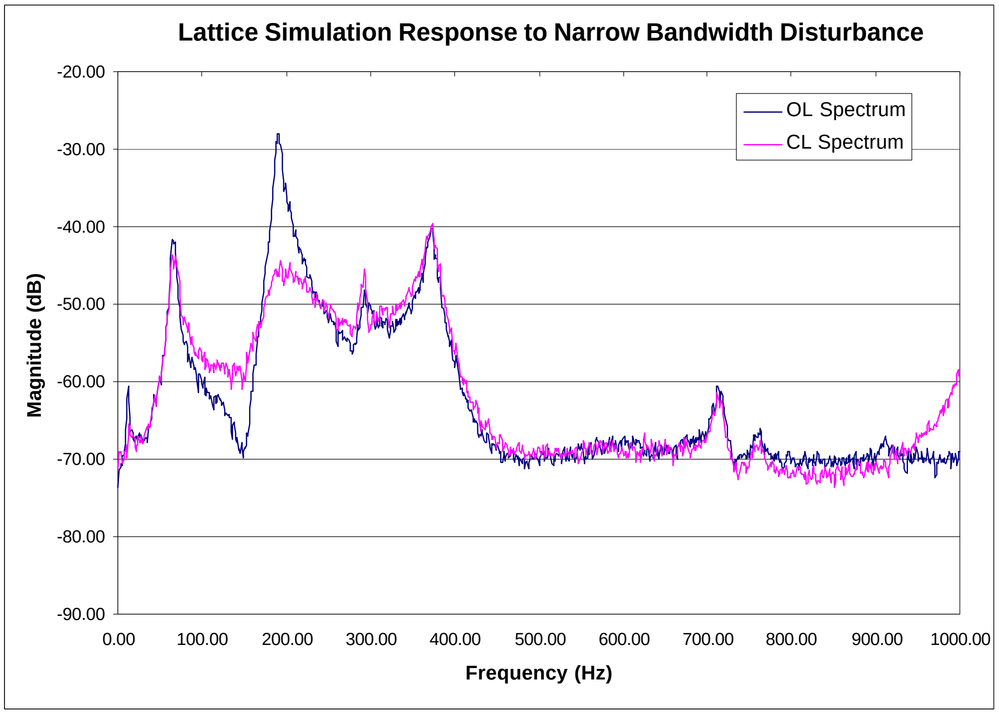


Figure 24 – Matlab simulated response of the adaptive lattice filter algorithm to the narrow band disturbance input after one minute of convergence.

The lattice filter algorithm converged to 0.02 V_{peak}. This convergence is better than the LMS algorithm's convergence and looking at the 189 Hz mode in Figure 24, corresponds to a reduction of 18 dB. This is about twenty five times smaller than the open loop response.

Overall, the improved performance of the lattice filter algorithm over the LMS algorithm is not very large. A 3rd order instead of 7th order filter was used, and the convergence of the lattice filter algorithm was a little better than the LMS algorithm. Since the lattice filter algorithm is more complex to implement than the LMS algorithm, the lattice filter algorithm is not a good choice for narrow bandwidth disturbances.

8.3 Wide Band Simulation

To see if the adaptive lattice filter is an improvement over the LMS algorithm, a wide bandwidth disturbance is considered. The wide bandwidth disturbance is generated using a butterworth filter with a center frequency of 300 Hz and a bandwidth of 250 Hz. This filter covers two of the largest modes in the system. The seed of the random number generator is the same one used in the narrow bandwidth case.

The performance of the optimal controller is shown below in Figure 25.

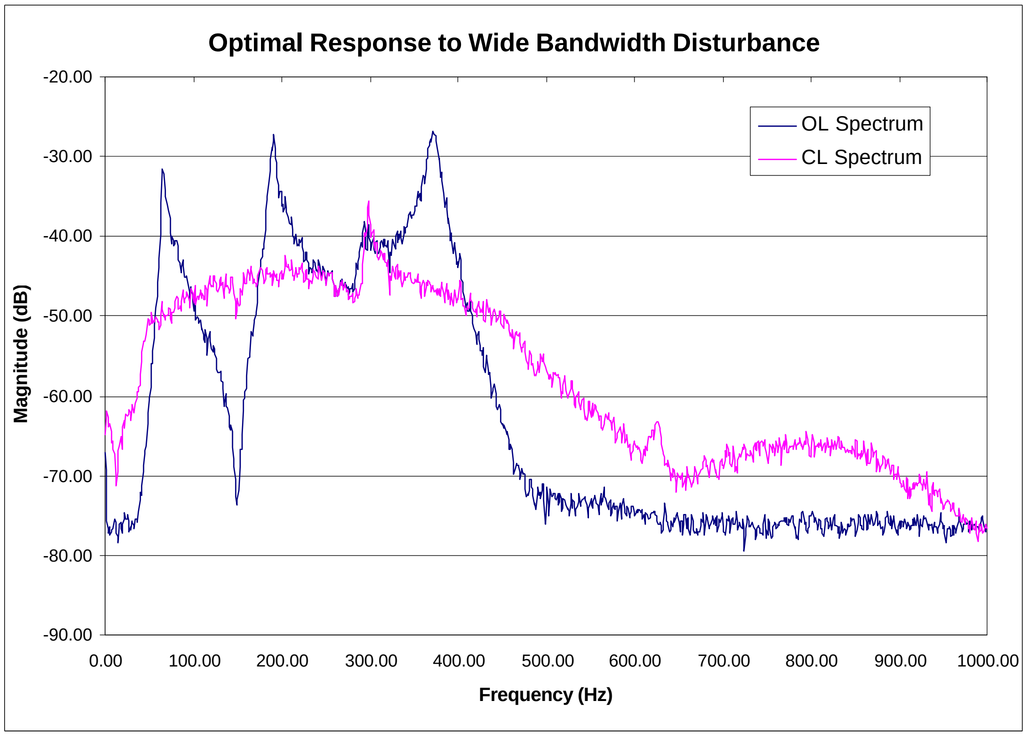


Figure 25 – Matlab simulated performance of the optimal controller to a wide band disturbance centered at 300 Hz.

The optimal controller was able to reduce the disturbance 17 dB at 189 Hz and 20 dB at 367 Hz.

The same wide band disturbance was sent through the adaptive controller using the LMS algorithm. The order of the filter was set to twelve. Higher orders did not seem to improve the performance of the controller. The step size for the weights of the filter was set to 5. The initial response of the LMS algorithm to the disturbance is shown Figure 26.

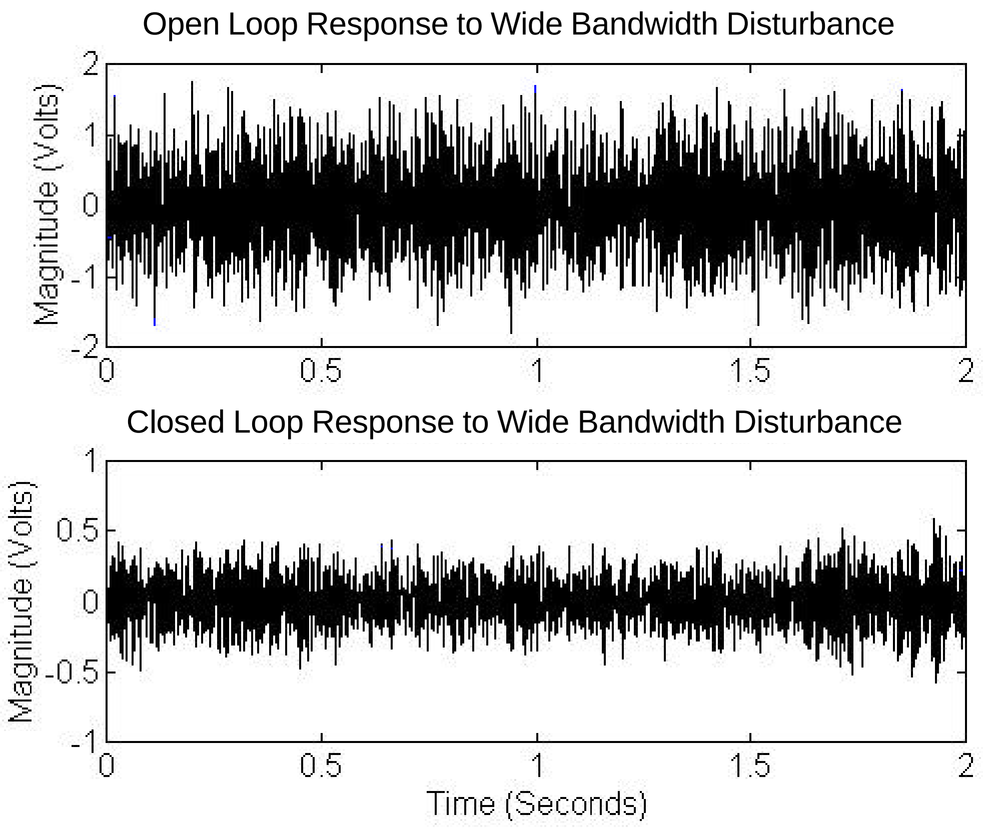


Figure 26 - Initial adaptation of the LMS algorithm to a wide band disturbance centered at 300 Hz.

Figure 26 shows no visible evidence that the LMS algorithm is converging, however, the filter parameters did slowly change during several minutes of running. The simulation was left to adapt for 5 minutes to make sure the filter fully converges. The open loop and closed loop spectral response of the LMS algorithm after the 5 minutes of convergence is shown in Figure 27.

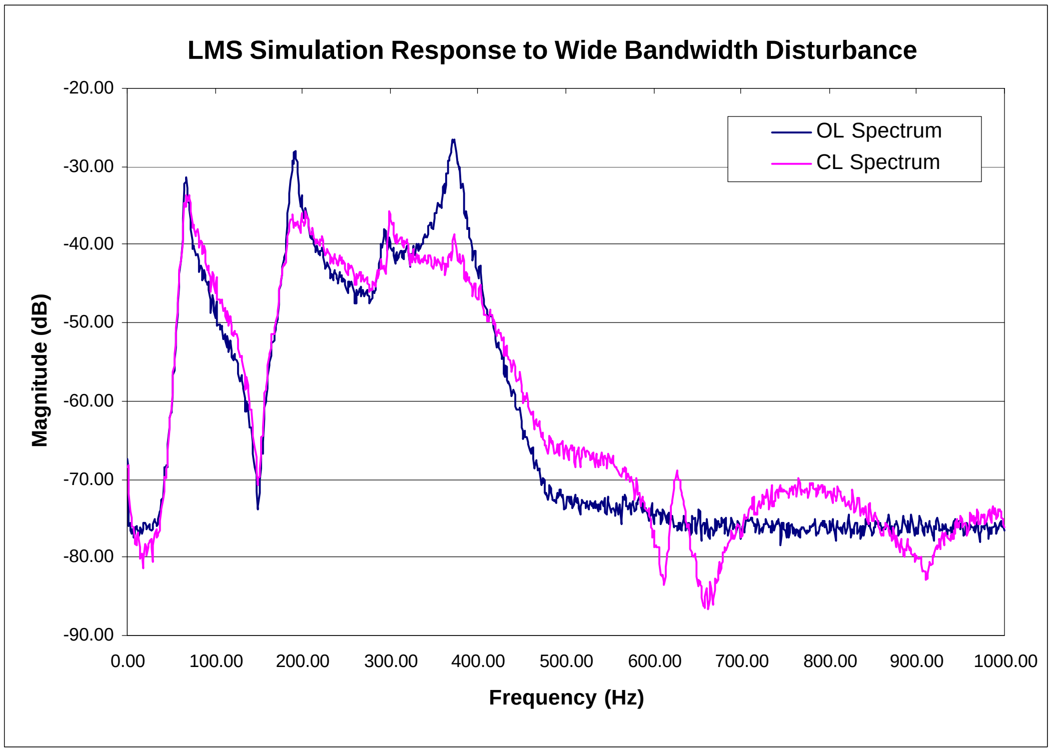


Figure 27 – Matlab simulated response of the LMS algorithm to the wide band disturbance input after five minutes of convergence.

The LMS algorithm damped the disturbance 9 dB at 189 Hz and 13 dB at 367 Hz. The optimal controller performance is 1.3 times better than the converged LMS algorithm performance. This is around 3 times smaller than the open loop response.

The lattice filter algorithm was subjected to the same wide band disturbance as the LMS algorithm and the optimal controller. The order of the lattice filter was set to 5rd order and the step size parameters for the n and q parameters were set to 5 and 0.08 respectively. The q parameter step size was reduced because larger step sizes resulted in the system going unstable. The initial run of the lattice filter algorithm is shown in Figure 28.

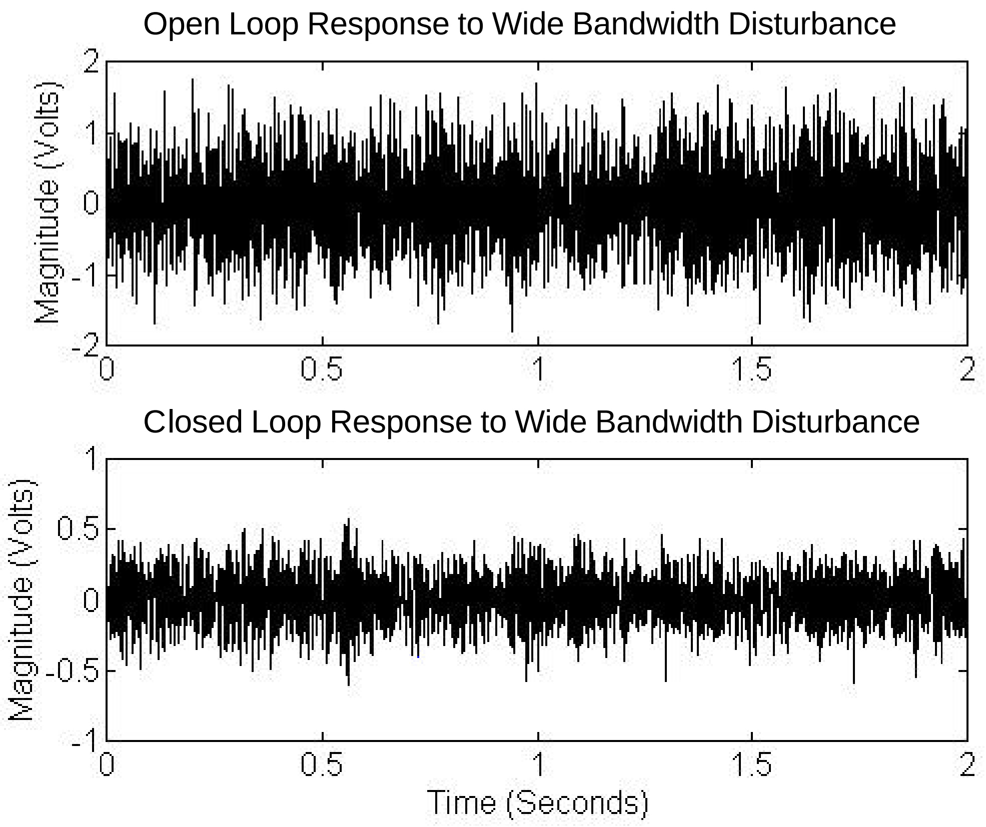


Figure 28 - Initial adaptation of the lattice filter algorithm to a wide band disturbance centered at 300 Hz.

The lattice filter algorithm is slowly converging to its optimal performance. The simulation was left to run for five minutes. After the five minutes, the performance was greatly improved. The performance is shown in Figure 29 below.

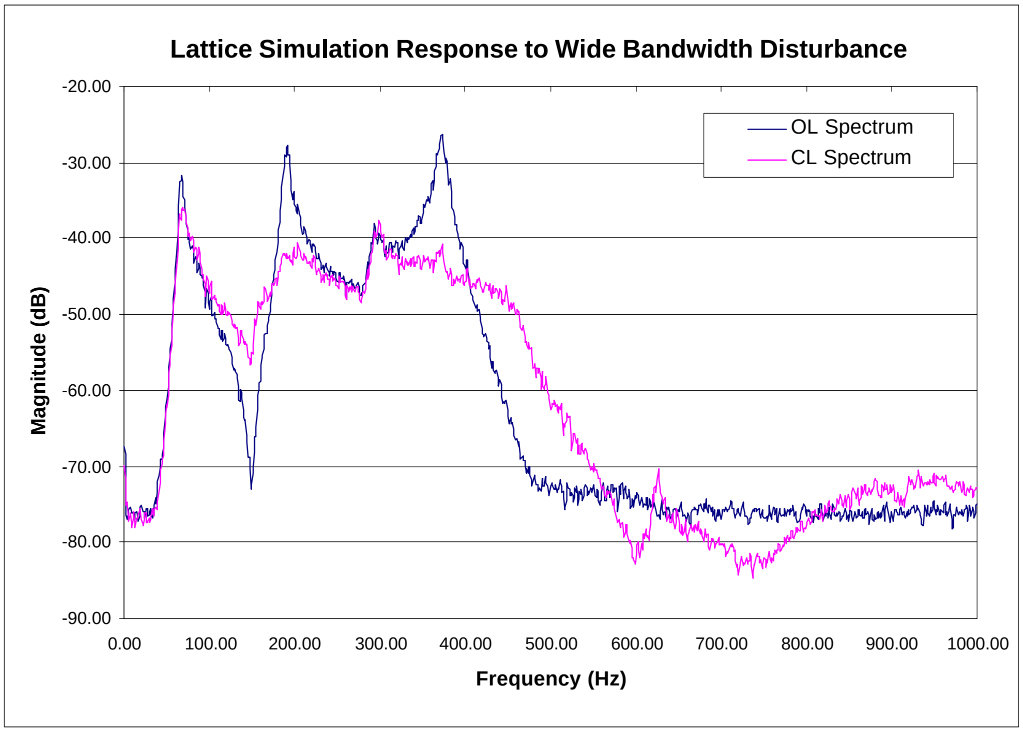


Figure 29 – Matlab simulated response of the lattice filter algorithm to the wide band disturbance input after five minutes of convergence.

The lattice filter algorithm converged yielding a reduction in the disturbance of 12 dB at 189 Hz and 15 dB at 367 Hz. This convergence is better than the LMS algorithm's convergence and is close to the optimal controller's performance. The reduction is about 5.6 times smaller than the open loop response.

Overall, the improved performance of the lattice filter algorithm over the LMS algorithm is quite substantial. A 5th order instead of 12th order filter was used, and the reduction in the disturbance for the lattice algorithm was almost two times the reduction achieved by the LMS algorithm. For disturbances that span many modes of the system, the lattice filter algorithm is a significant improvement over the LMS algorithm.

Chapter 9 Experimental Results

The experimental results consist of a comparison between the controller using an LMS filter, the controller using the lattice filter, and the theoretically optimal controller.   The power spectrum at the output of the beam is measured using a Hewlett Packard 35665A Dynamic Signal Analyzer. The measurements are set to average 10 runs for the purpose of reducing the white noise in each measurement. Additionally, a uniform window is used to smooth the data. Comparisons are made by checking the reduction in the largest mode of concern for each case.

9.1   LQG Controller

The LQG controller was designed with a control weighting of 0.96. This is the same weighting that was used in the simulations. The LQG controller was designed to give a minimal amount of damping to the system. The effect of the LQG controller on a harmonic disturbance is shown in Figure 30.

Measurements in of the reduction of the disturbance are taken using the HP analyzer marker function. The marker function can identify a peak in the open loop spectral response, set this as a zero decibel point, then measure the change in the peaks magnitude relative to this point when the LQG control loop is closed around the system.

Additionally, the noise floor is plotted with the harmonic plot. The noise floor is the response of the system, measured at the output, with no inputs. It can be seen from Figure 30 that there is a second harmonic generated by the function generator which has a frequency of 378 Hz. Because of this additional harmonic, the adaptive controller requires a larger order adaptive filter to reduce the disturbance to the noise floor. The larger order is required because two harmonics need to be reduced in the system, not just the main harmonic.

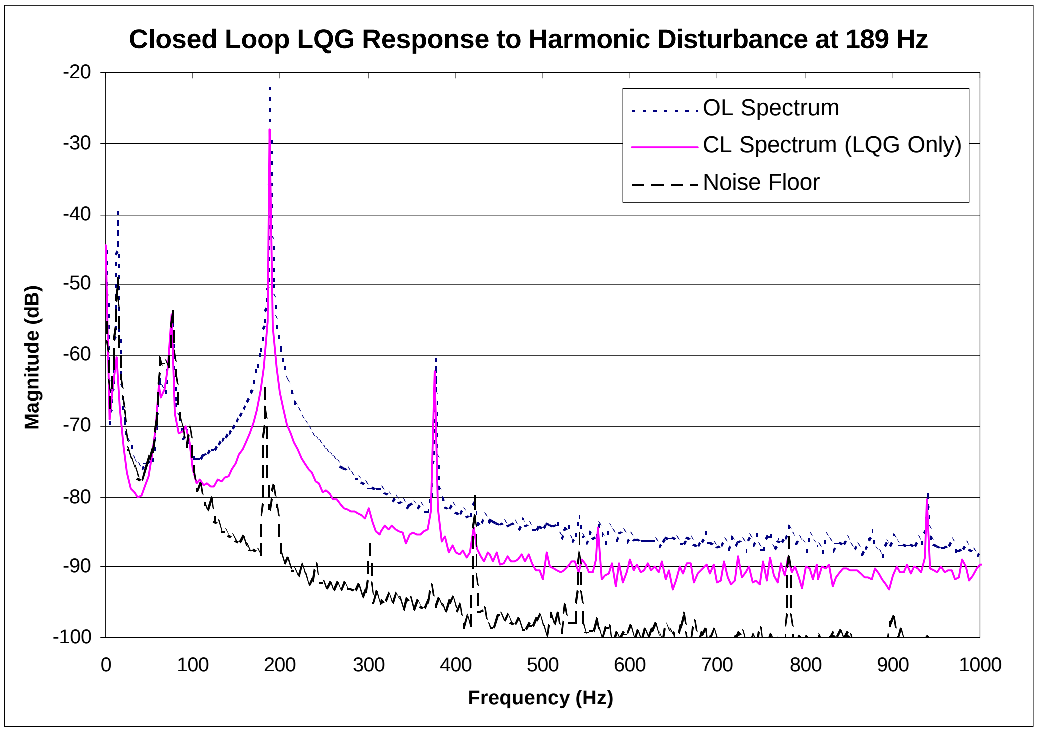


Figure 30 – Open and closed loop response of the control system to a harmonic disturbance of 189 Hz.

The LQG controller reduced the harmonic disturbance by 7.5 dB at 189 Hz. This leaves plenty of room for the adaptive algorithms to show their ability to affect the disturbance.

The closed loop LQG response to a narrow band disturbance is shown in Figure 31, below.

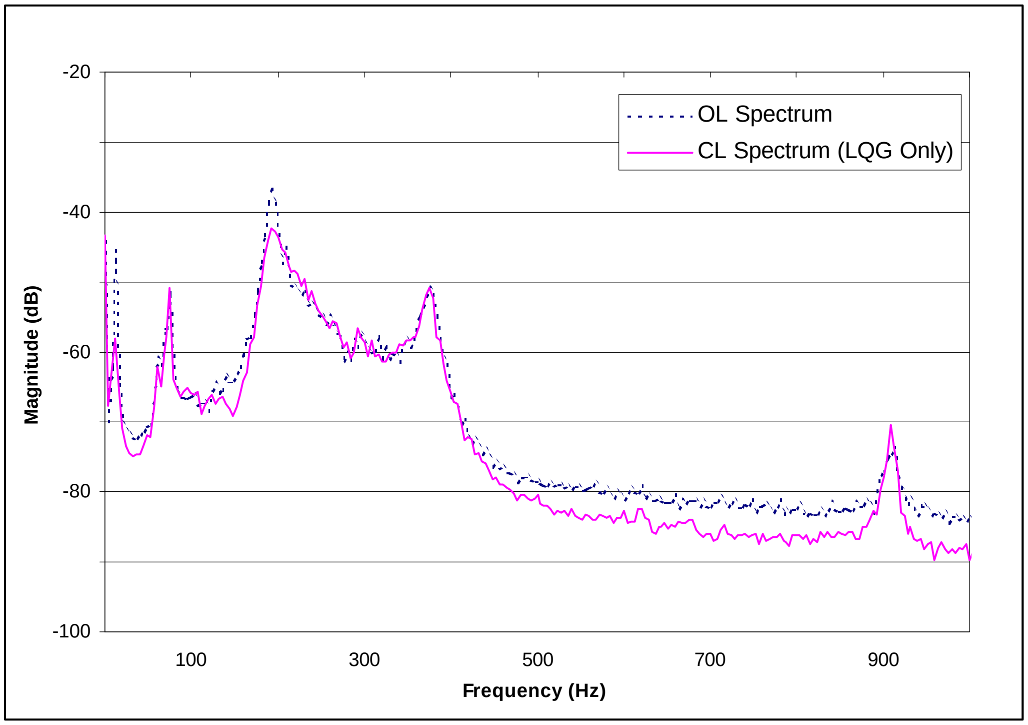


Figure 31 – Closed loop response of the beam to a narrow band disturbance centered around 189 Hz.

The dominant mode in the narrow band case is the mode at 189 Hz. This is expected since the narrow band filter is centered at this frequency. The LQG controller reduced the disturbance at this mode by 5 dB.

The effect of the LQG controller on a system with a wide bandwidth disturbance is shown in Figure 32.

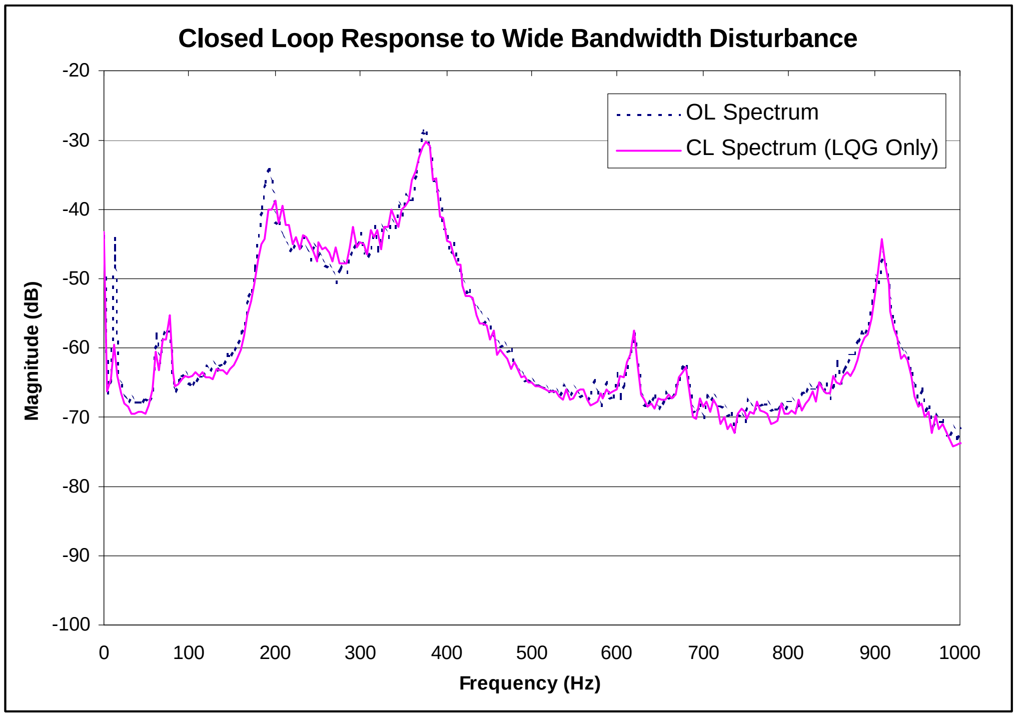


Figure 32 – Closed loop response of the beam to a wide bandwidth disturbance centered around 300 Hz and having a bandwidth of 250 Hz.

There are two dominant modes in the wide bandwidth disturbance case. The modes are located at 189 Hz and 367 Hz. The mode at 189 Hz was reduced by 5 dB and the mode at 367 Hz was reduced by 2 dB.

9.2 Verification of Neutralization Loop

The adaptive filter needs to be placed in the neutralization loop to prevent feedback through the controller. In practice, an exact model of the system cannot be made, so the neutralization loop generates same non-zero transfer function between s_k and r_k. To measure the relative size of the transfer function in the experimental setup, the HP analyzer is used. A pseudo random binary sequence is generated using the signal processing board. This is used as a white noise input into s_k. The analyzer is set up to measure the power spectral density at r_k with no disturbance input at d_k. The resulting measurement is shown below in Figure 33.

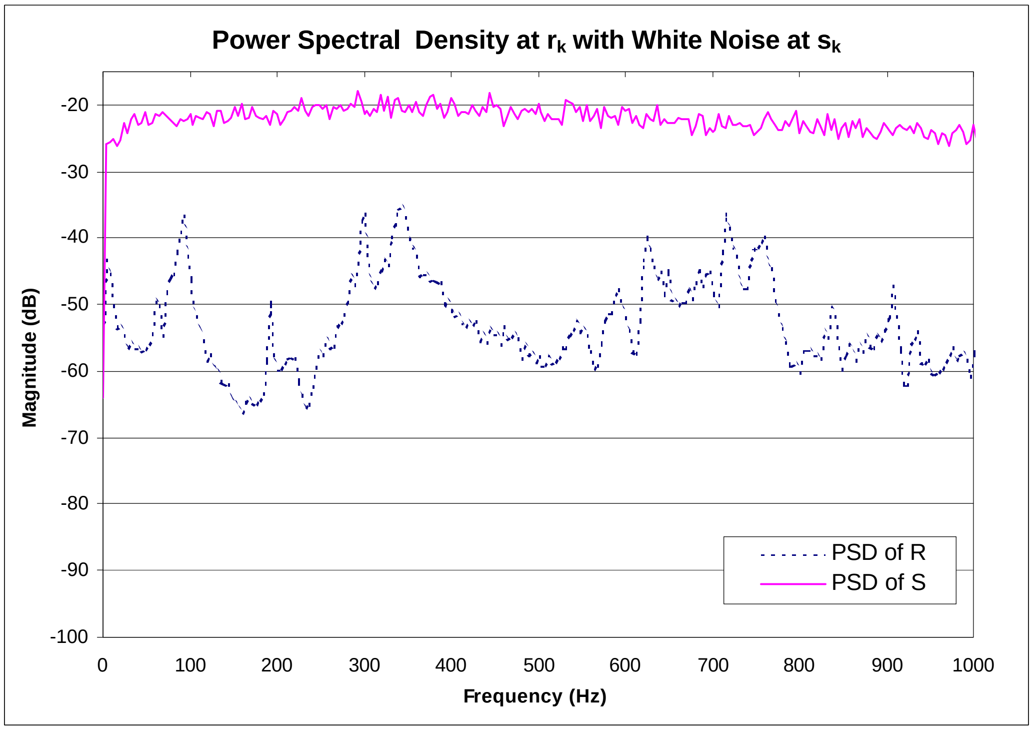


Figure 33 – Power spectral density at r_k with white noise at s_k.

This plot of the spectrum shows the true transfer function is indeed not zero. The majority of the signal resides at the modes of the system where the errors have more energy and so are magnified.

9.3 Harmonic Disturbance

The harmonic disturbance was generated from a Tektronix FG502 function generator. The disturbance was sent to the piezoceramic actuator that represents the disturbance input. A National Semiconductor uA741 operational amplifier with a gain of 1 V/V was used as a buffer between the piezoceramic element and the function generator.

For the LMS algorithm, the step size, μ, was set to 0.75. Larger step size would cause a saturation of the controller. The experimental setup was set to run 3 to 4 times longer than the simulations to make up for the reduction in the step size. The filter was adjusted to a 3rd order filter. The effect of the LMS algorithm on the harmonic disturbance is shown in Figure 34.

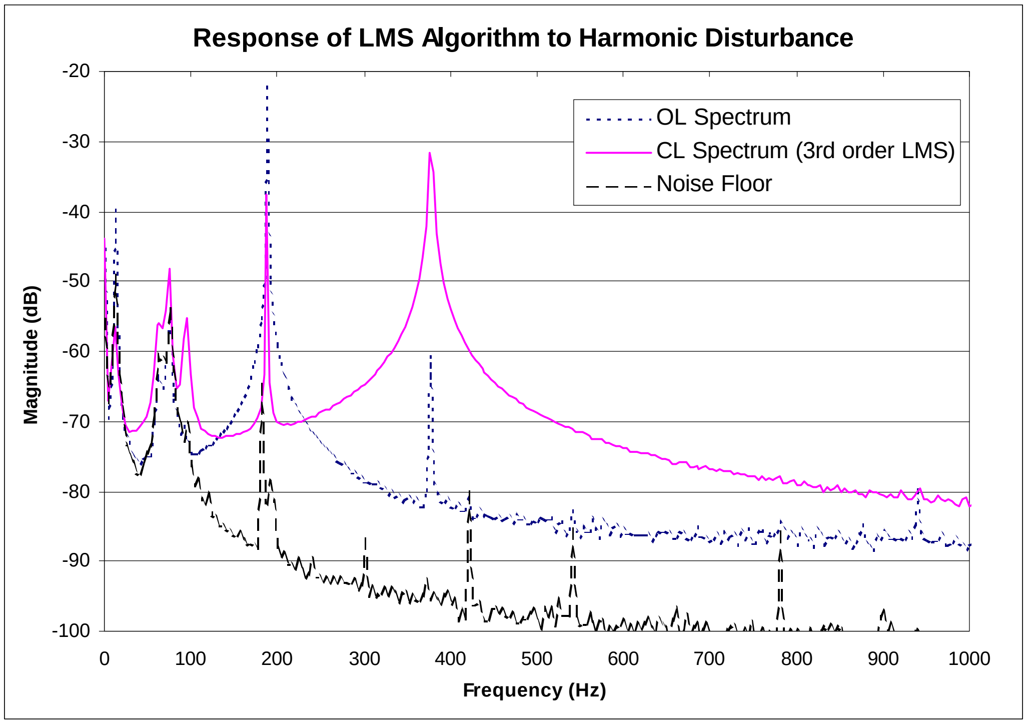


Figure 34 – Response of the LMS algorithm using a 3rd order LMS filter to a harmonic disturbance.

The harmonic disturbance was reduced 17.6 dB by the LMS algorithm. During the adaptation, a mode at 378 Hz was excited. As a result, the overall disturbance rejection for the system is not optimal. This harmonic excitation is a second harmonic created by the function generator. To reduce the mode at 189 Hz and the mode at 378 Hz, a larger order filter is needed. The filter order was increased by one and tested until the filter was large enough the reduce both modes. This resulted in a 7th order LMS filter. The response of the filter is shown below in Figure 35.

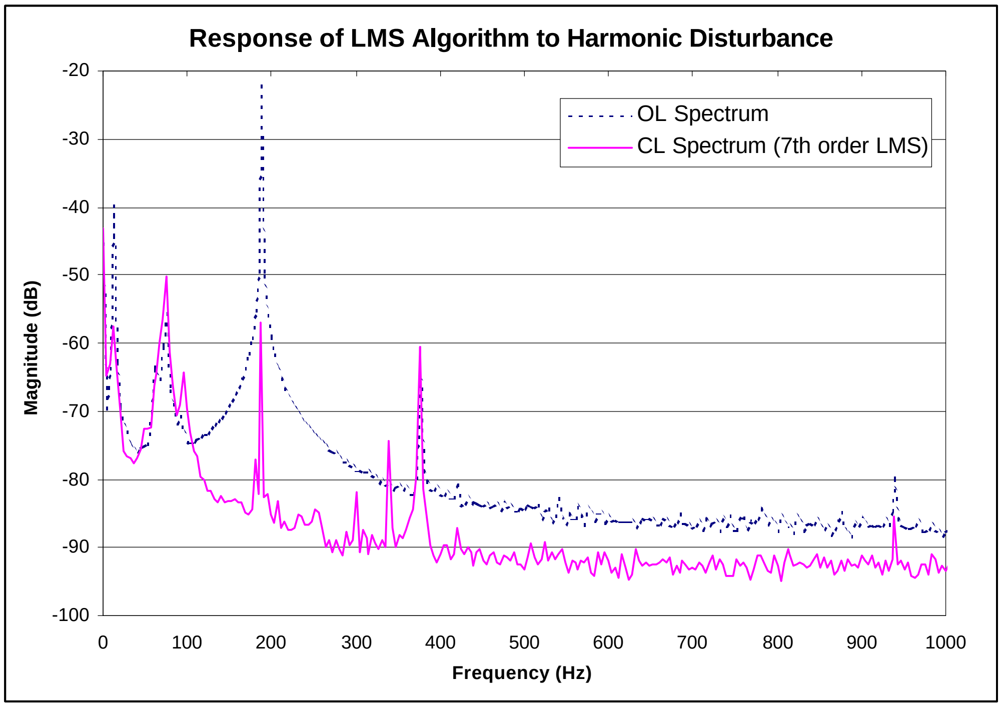


Figure 35 – Response of the LMS algorithm to a harmonic disturbance located at 189 Hz.

The algorithm reduced the harmonic disturbance by 36 dB. This was enough to drive the disturbance into the noise floor.

Next, the lattice algorithm was used to remove the same harmonic disturbance. Again, the second harmonic at 378 Hz appeared and required a larger filter order to reduce the noise. The filter was set at first order then increased by one until the system could reduce the disturbance. The filter was required to be 3rd order and the step size for the *n* and *q* parameters were set to 0.75 and 0.1 respectively. Figure 36 shows the effect of the lattice algorithm on the 189 Hz harmonic disturbance.

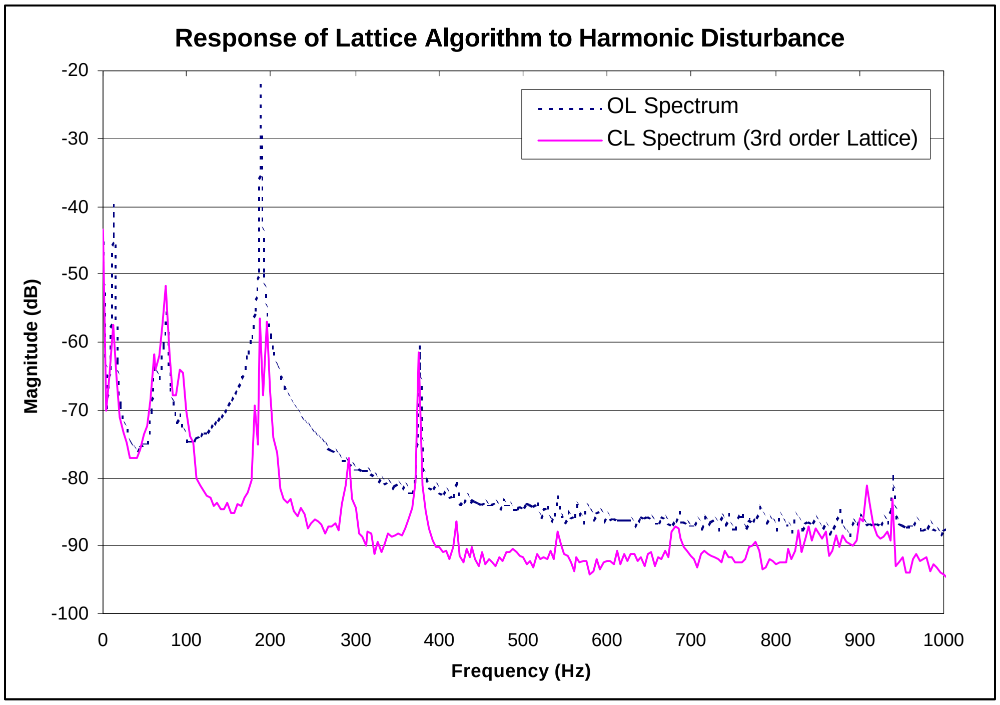


Figure 36 – Response of the lattice algorithm to a harmonic disturbance located at 189 Hz.

The lattice algorithm was capable of reducing the disturbance by 37 dB. The 367 Hz mode was reduced as it was with the 7th order LMS filter. This is because the lattice filter is an IIR filter and can adapt both numerator and denominator coefficients in its transfer function. This gives the filter more freedom to fit the 189 Hz and 367 Hz modes.

9.4 Narrow Band Disturbance

To create the narrow bandwidth disturbance, a Krohn-Hite model 3202 analog filter was used. This particular filter was used because it was readily available and could generate fourth order filters with relative ease. The filter breakpoints were set to 180 Hz and 200 Hz. Additionally, a uA741 operational amplifier was placed at the output of the filter to boost the signal and serve as a driver for the piezoceramic actuator. It was unnecessary to use a power booster to drive the disturbance piezoceramic element because it did not need an output greater than the ±10 volts available from a 741 op-amp. Additionally, the 741 is internally compensated so no additional components were required to keep the amplifier and piezoceramic device combination stable. For the narrow bandwidth disturbance case, the gain of the 741 amplifier was set to 10 V/V. The frequency spectrum of this narrow bandwidth filter is shown in Figure 37

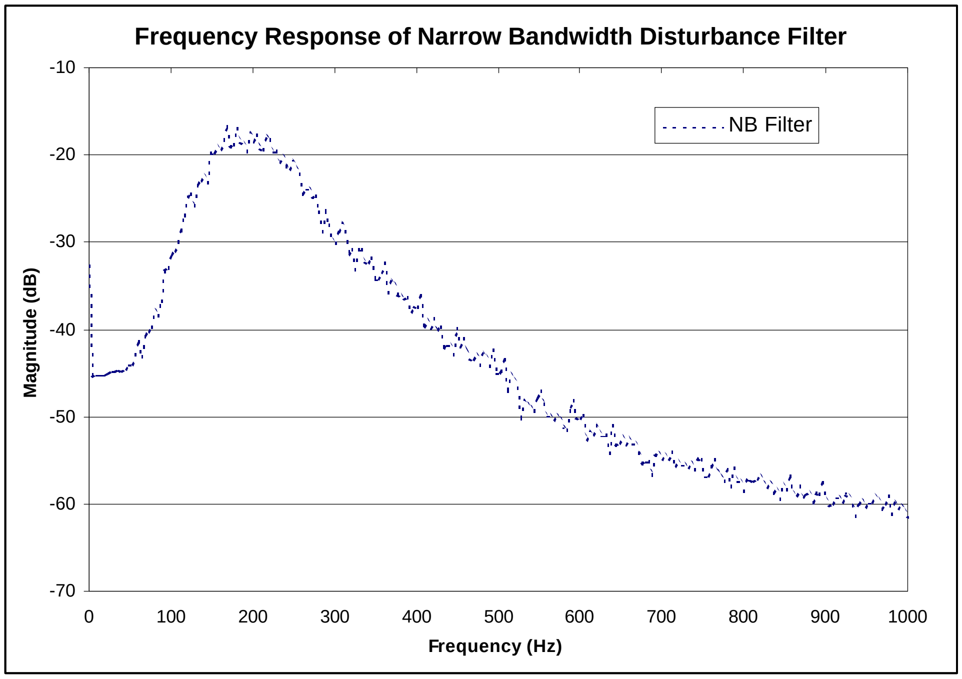
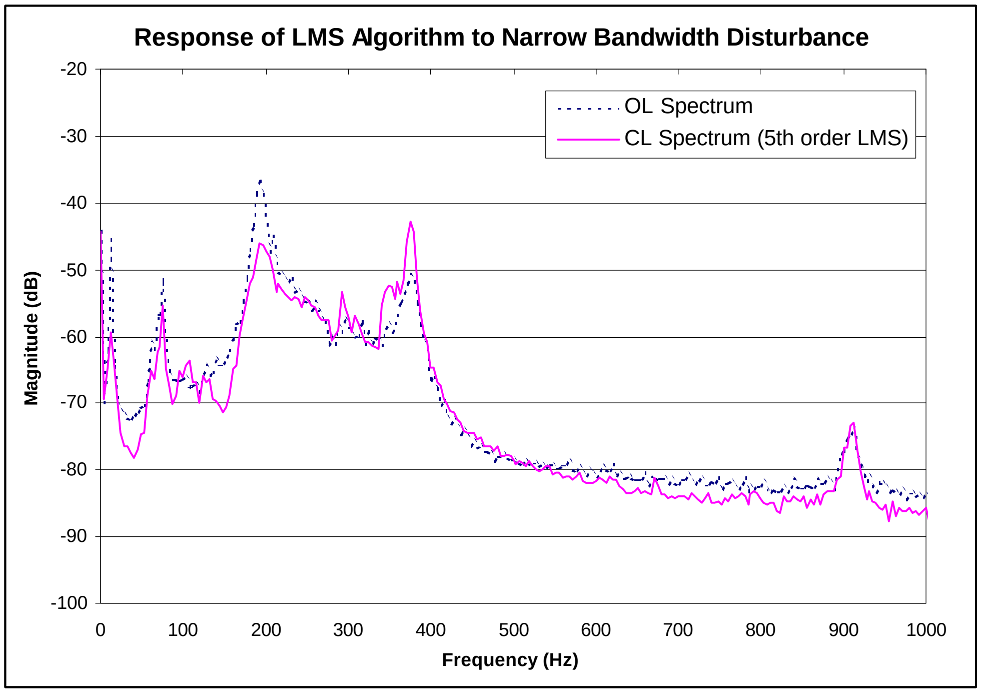


Figure 37 – Frequency response of the narrow bandwidth filter.

Since this filter places most of the disturbance power at the 189 Hz mode, the reduction in its magnitude is used to measure the performance of the LMS and lattice algorithms.

First, the ability of the LMS algorithm to reduce the narrow bandwidth disturbance is considered. The step size for the LMS algorithm was left at 0.75 and the filter was adjusted to 7th order based on the results of the harmonic disturbance experimental run. The effect of the LMS algorithm on the narrow bandwidth disturbance is shown Figure 38. A copy of the simulation response is also present for referencing purposes.



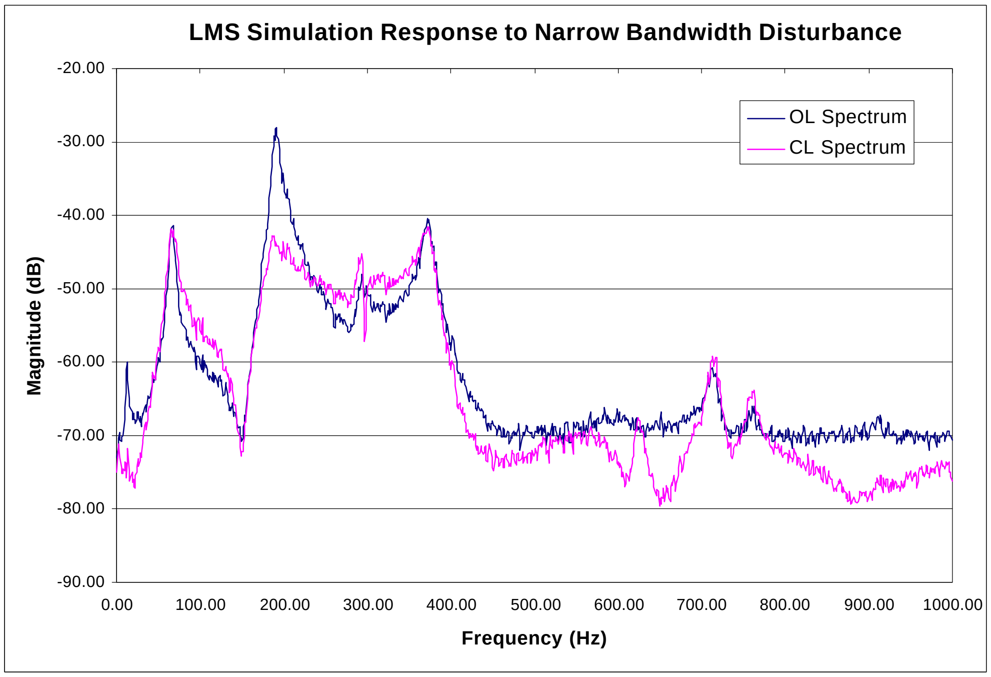


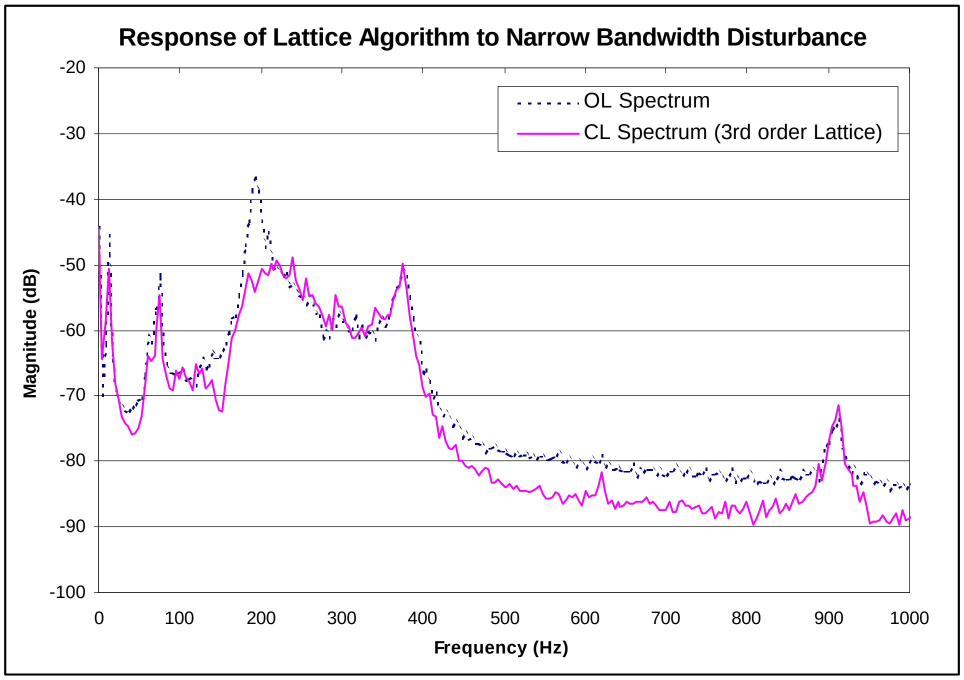
Figure 38 – Response of the LMS algorithm to a narrow bandwidth disturbance.

The LMS algorithm reduced the disturbance 11 dB at the dominant mode of 189 Hz. Compared to the simulated response, the algorithm performs only a little worse on the experimental setup than was predicted in simulation.

The response of the lattice algorithm to the narrow bandwidth disturbance was looked at next. The filter was left at 3rd order and the step size for the n and q parameters were set to 0.75 and 0.2 respectively. The narrow bandwidth disturbance for this case is the same one used as the disturbance for the LMS algorithm.

Figure 39 shows the effect of the lattice algorithm to a narrow bandwidth disturbance.

The lattice algorithm reduced the disturbance by 15 dB. The simulations predicted the lattice algorithm would reduce the disturbance more than the LMS algorithm. This is true in the experimental setup as well.



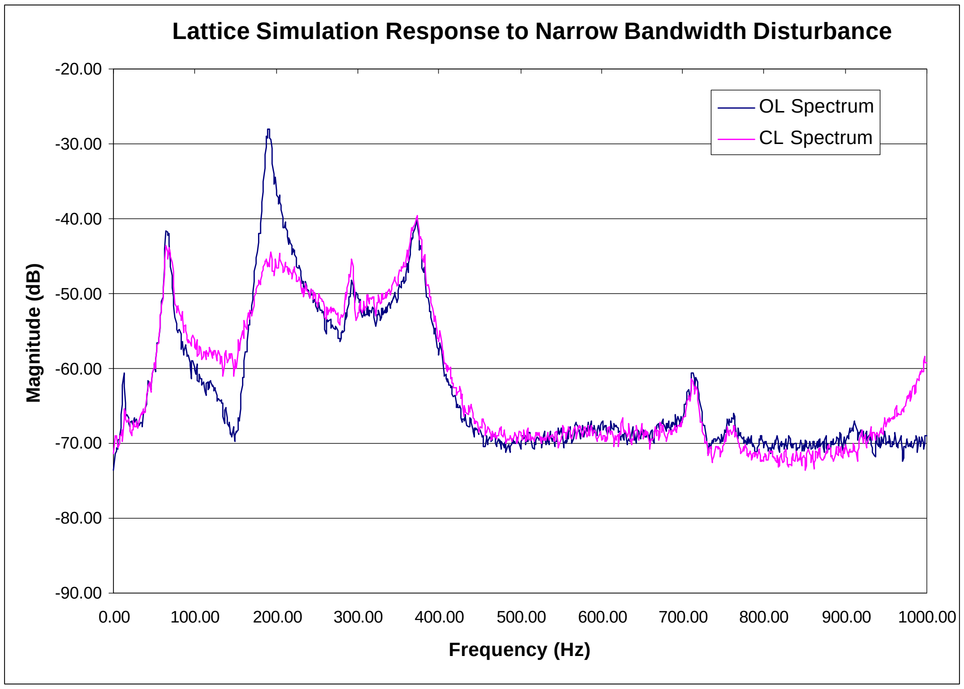


Figure 39 – Response of the lattice algorithm to a narrow bandwidth disturbance.

9.5 Wide Band Disturbance

To create the wide bandwidth disturbance, the analog filter used to create the narrow bandwidth filter was adjusted. The filter breakpoints were set at 175 Hz and 425 Hz. This provided a filter with a center frequency of 300 Hz and a bandwidth of 250 Hz. This filter was selected to cover two modes in the beam, the 189 Hz and 367 Hz modes. To measure the amount of disturbance rejection, it is convenient to measure the reduction in the size of the 367 Hz mode. Additionally, the uA741 operational amplifier gain was adjusted to 20 V/V. The frequency spectrum of the wide bandwidth filter is shown in Figure 40.

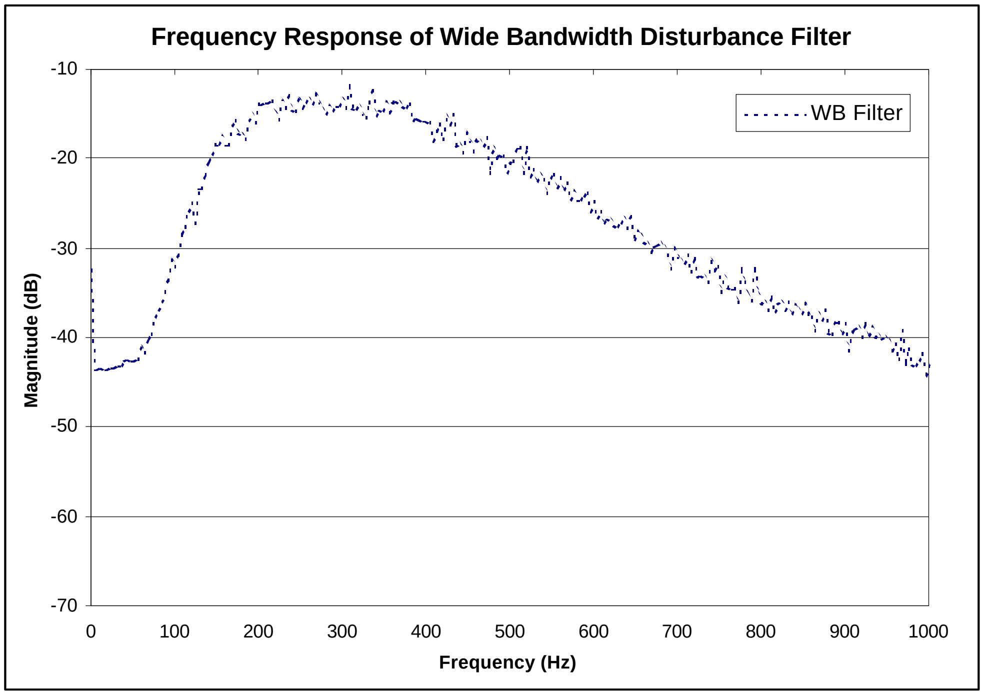
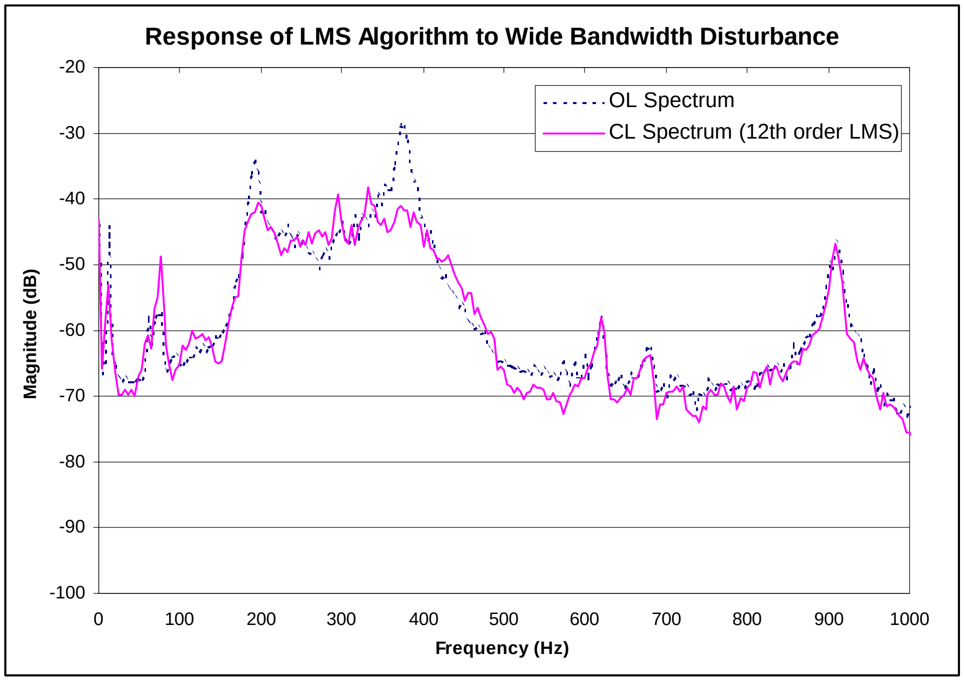


Figure 40 – Frequency response of the wide bandwidth filter.

The step size for the LMS algorithm was left at 0.75. The filter was adjusted to 12th order based on the results of the simulation. The effect of the LMS algorithm on the wide bandwidth disturbance is shown in Figure 41.



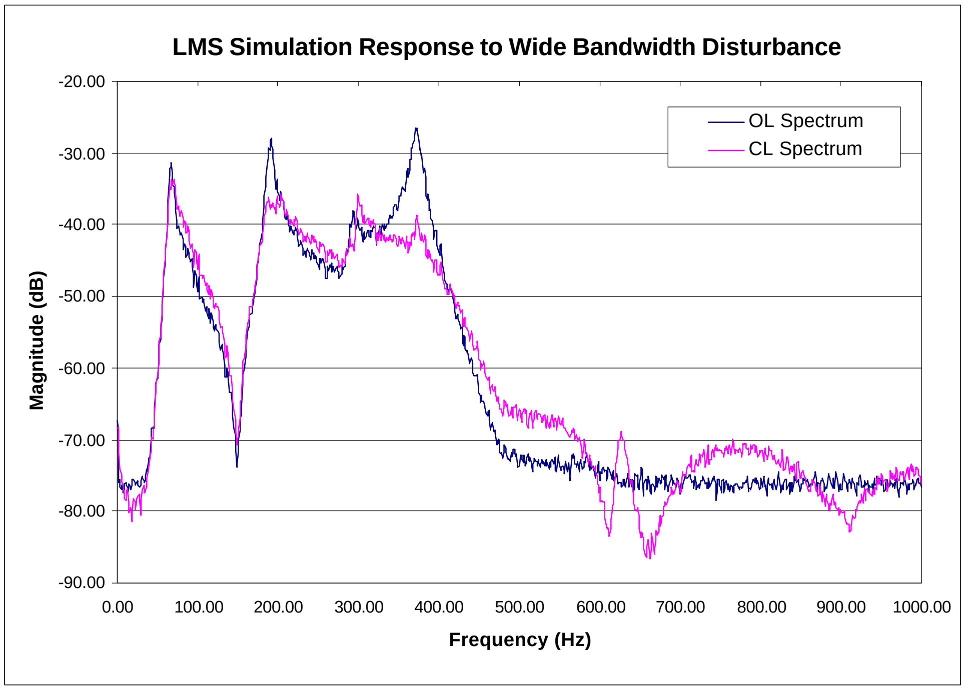
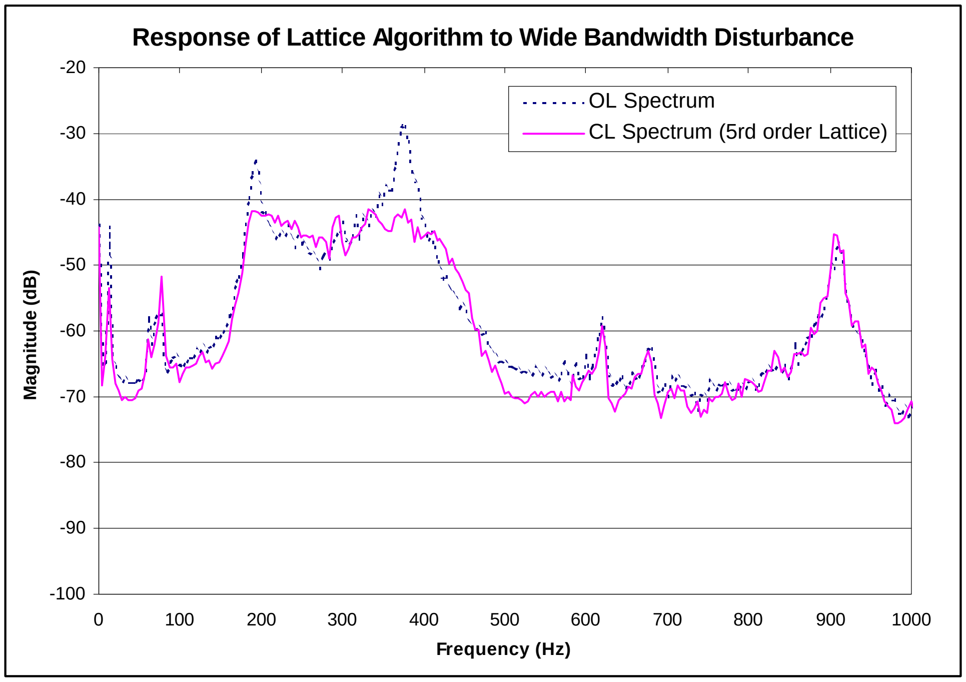


Figure 41 – Response of the LMS algorithm to a wide bandwidth disturbance.

The LMS algorithm reduced the disturbance 6 dB at the 189 Hz mode and 12 dB at the 367 Hz mode. This is close to the 9 dB and 13 dB reduction achieved by the simulation. The convergence time for this case was longer than in simulation because of the smaller step sizes that were required. Remember that large step sizes would result in a saturation of the controller. This is also true for the lattice case discussed below.

The lattice algorithm filter was adjusted to 5th order. The step size for the n and q parameters were left at 0.75 and 0.2 respectively. The wide bandwidth disturbance for this case is the same one used as the disturbance for the LMS algorithm. Figure 42 shows the effect of the lattice algorithm to the wide bandwidth disturbance.



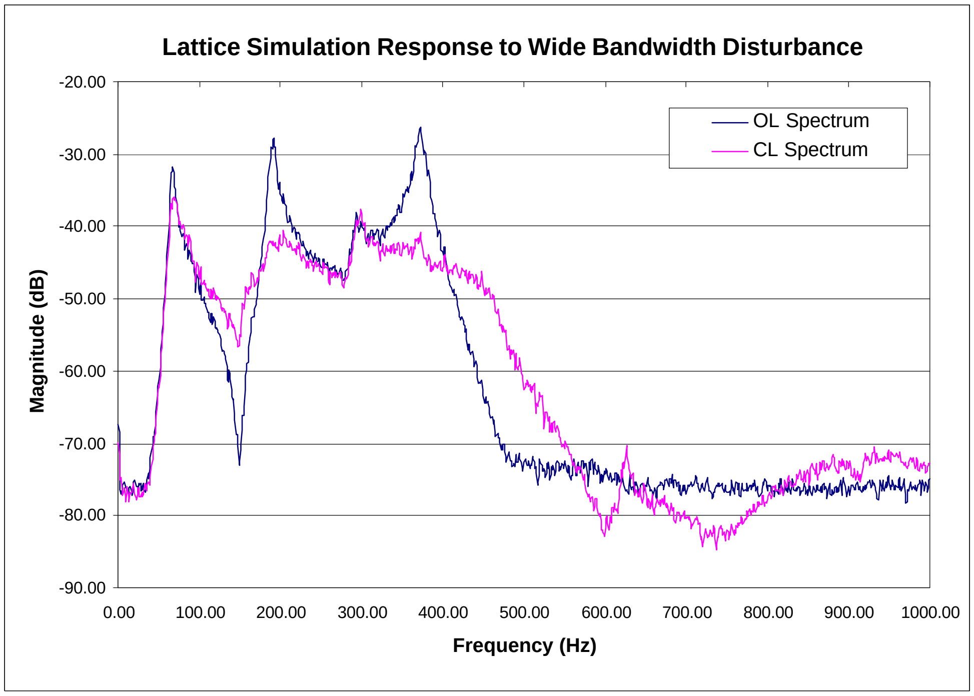


Figure 42 – Response of the lattice algorithm to a wide bandwidth disturbance.

The lattice algorithm reduced the disturbance by 8 dB at the 189 Hz mode and 14 dB at the 367 Hz mode. This is the similar to the amount of rejection achieved by the simulation. Again, as in the simulation, the lattice algorithm performs better than the LMS algorithm with a smaller order filter.

The results for Chapter 8 and these results show the true benefit of using an adaptive lattice filter in place of an adaptive LMS filter; the order of the filter required is cut almost in half. Unfortunately, the lattice filter algorithm is a complex algorithm to implement. As a result, the lattice algorithm should only be used when needed for complex multi-modal, possibly MIMO, plants.

Chapter 10 Conclusions

This thesis contains the theoretical and experimental results obtained for two forms of an adaptive disturbance rejection controller. A disturbance rejection controller using an FIR and an IIR adaptive filter were implemented and their results compared. The results show the improvement in the performance of the controller using an IIR adaptive filter over the same controller using an FIR adaptive filter.

Despite the improvement in the performance of the controller over the LMS algorithm, the LMS algorithm is a better choice for most applications. The LMS algorithm is much easier to implement on a digital signal processor and it has an overall good performance, as shown by the experimental results.

If the lattice filter controller were used on a multi-input multi-output (MIMO) system with many modes, it would improve the computation time and performance enough to make it worthwhile to implement. This potential is worth exploring in future work, as it could be capable of removing disturbances in systems such as complex space structures, noise in close environments, and numerous other situations.

Also, making the system fully adaptive so that the system works with plants whose parameters vary with time would be useful. It would open up a wide range of applications for the adaptive lattice and LMS control structure.

References

1. Conover, W. B., 1956, “Fighting Noise with Noise”, *Noise Control*, Vol. 2, pp. 78-82

2. Boucher, Elliot, and Nelson, 1991, “The Effect of Errors in the Plant Model on the Performance of Algorithms for Adaptive Feedforward Control”, *Proceedings of the Institute of Electrical Engineers*, Vol. 138, 313-319.

3. Olson, H and May, E., 1953, “Electronic Sound Absorber”, *Journal of the Acoustic Society of America*, Vol. 25, No. 6, pp. 1130-1136.

4. Nelson, p, Elliot, S., 1992, “Active Control of Sound”, Academic Press, San Diego, 1992.

5. Tay, T.T. and Moore, J.B., 1991, “Enhancement of Fixed Controllers via Adaptive-Q Disturbance Estimate Feedback”, *Automatica*, Vol. 27, No. 1 pp. 39-53.

6. Regalia, Phillip A., *Adaptive IIR Filtering in Signal Processing and Control*, Marcel Dekker, Inc., 1995.

7. Moore, J.B., Glover, K., and Telford, A., 1990, “All Stabilizing Controllers as Frequecny-Shaped State Estimate Feedback*”, IEEE Transactions on Automatic Control*, Vol. 35, No. 2 pp. 203-208.

8. Baumann, W.T., 1997, “An Adaptive Approach to Structural Vibration Suppression”, *Journal of Sound and Vibration*, pp. 121-133.

9. Miller, Daniel, 1995, “Adaptive Feedback Technique for Unmodeled Disturbance Rejection”, *Master’s Thesis Virginia Polytechnic Institute and State University*.

10. Horrowitz, Paul and Hill, Winfield, *The Art of Electronics, Second Addition*, Cambridge University Press, 1994.

11. Classen, Mecklenbrauker, and Peek, 1976, “Effects of Quantization and Overflow in Recursive Digital Filters”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 24, pp. 517-529

12. Ljung, Lennart, *System Identification Theory for the User*, Prentice Hall, 1987.

13. M. Nayeri and W. Jenkins, 1989, “Alternate realizations of adaptive IIR filters and properties of their performance surfaces”, *IEEE Transactions on Circuits and Systems*, Vol. 36, pp. 485-496.

14. J. Shynk, 1987, “Performance of alternate adaptive IIR filter realizations”, *Proc. Asilomar Conf. Circuits, Systems, and Computers*, Pacific Grove, CA.

15. Haykin, Simon, *Adaptive Filter Theory*, Prentice Hall, 1986.

16. Astrom, Karl and Wittenmark, Bjorn, *Adaptive Control - Second Edition*, Addison-Wesley Publishing Company, Inc., 1995.

17. Spectrum Signal Processing, “PC/C31 Board User Guide”, Version 1.04, Burnaby, B.C.

18. Saunders, Robertshaw, and Burdisso, 1996, “A Hybrid Structural Control Approach for Narrowband and Impulsive Disturbance Rejection”, *Noise Control Engineering Journal*, Vol. 44, pp. 11-21.

19. Clark, R. L., 1995, “A Hybrid Autonomous Control Approach”, *American Society of Mechanical Engineers Journal of Dynamic Systems, Measurement, and Control*, Vol. 117, pp. 232-240.

20. Lueg, P., 1936, "Process of Silencing Sound Oscillations*", U.S. Patent No. 2,043,416*.

21. Elliot, Stothers, McDonald, Quinn, Saunders, and Nelson, 1988, "The Active Control of Engine Noise Inside Cars", *Proc. Inter-Noise 88,* pp. 989-990.

22. Elliot, Nelson, Stothers, and Boucher, 1990, "In-flight Experiments on the Active Control of Propeller-induced Cabin Noise", *Journal of Sound and Vibration,* Vol. 140, pp. 219-238.

23. Eghtesadi, Hong, and Leventhall, 1983, "The Tight-Coupled Monopole Active Attenuator in a Duct", *Noise Control Engineering Journal*, Vol. 20, No. 1, pp. 16-20.

24. Smothers, Saunders, McDonald, and Elliot, 1993, "Adaptive Feedback Control of Sunroof Flow Oscillations", *Proceedings of the Institute of Acoustics,* Vol. 15, pp. 383-393.

25. Wheeler, P, and Smeatham, 1992, "On Spatial Variability in the Attenuation Performance of Active Hearing Protectors", *Applied Acoustics*, Vol. 36, pp. 159-162.

Appendix A

A.1 SYSID.ASM

```
*
* SYSTEM ID ROUTINE
*
        .global     RESET,INIT,INPUT
        .global     M,SENSOR_ADDR
        .global     RANDOM_ADDR,RANDOM
        .global     ISR0,ISR1,ISR2,ISR3
        .global     CH_0,CH_1

*
* RESET VECTOR SPECIFICATION
*
        .sect  "init"           ;
RESET  .word  INIT              ; RESET VECTOR (LOADS INIT ADDRESS TO PC)

        .sect  ".int0"
        BR     ISR0
        .sect  ".int1"
        BR     ISR1
        .sect  ".int2"
        BR     ISR2
        .sect  ".int3"
        BR     ISR3
*
*
        .data
BUSADDR .word  00808064h        ; ADDRESS OF BUS CONTROL REGISTER
BUSDATA .word  00000900h        ; VALUE FOR LSI CARD GIVES ZERO
                                ; WAIT STATES, 32k BANK SIZE, EXTERNAL
                                ; READY CONTROL
*
* INITIALIZE CONSTANTS
*

M           .set   16384        ; NUMBER OF POINTS

CH_0        .word  550002h      ; CH_0 ADDRESS
TM1         .word  550005h      ; TIMER 1 REGISTER
CH_1        .word  550006h      ; CH_1 ADDRESS
UCR         .word  550008h      ; USER CONTROL REG.
ACR         .word  55000Ah      ; AMELIA CONTROL REG.
IMR         .word  55000Bh      ; INTERRUPT MASK REG.
```

```
CFR           .word   55000Fh            ; CONFIGURATION REGISTER

VALUES          .word   0A4000000h           ; A4000000h
                .word   0E8010000h           ; E8010000h 2000 Hz
                .word   00B20000h            ; 00B20000h
                .word   08DFF0000h           ; 8DFF0000h
                .word   00010000h            ; 00010000h

*SEED           .word   0AC1178h             ; Random Startup SEED
*SEED           .word   01AB903h             ; Random Startup SEED
*SEED           .word   00BA61Ch             ; Random Startup SEED
*SEED           .word   039CF27h             ; Random Startup SEED
SEED          .word   0FE633Dh               ; Random Startup SEED



END           .word   40000000h
MID           .word   08000000h

NORM_1          .float  1.09225E+4           ; Represents 1V
NORM_2          .float  9.155552843E-5       ;

*
* SPECIFY ARRAYS
*

RANDOM          .usect  "RAND",M             ; RANDOM OUTPUT ARRAY
SENSOR        .usect  "SENS",M               ; SENSOR INPUT ARRAY

*
* SPECIFY POINTERS TO ARRAYS
*

RANDOM_ADDR     .word   RANDOM       ; RANDOM ARRAY ADDRESS
SENSOR_ADDR     .word   SENSOR       ; SENSOR ARRAY ADDRESS

*
* INITIALIZE POINTERS AND ARRAYS
*
                .text

INIT:   LDP     BUSADDR,DP                   ; DATA PAGE POINTER SET PRIOR TO
                                     ; USING DIRECT ADDRESSING MODE
        LDI     @BUSADDR,AR0                 ;
        LDI     @BUSDATA,R0                  ;
        STI     R0,*AR0              ; SET BUS. REG. FOR LSI CARD
```

```
          LDP    @SENSOR_ADDR,DP        ; SET DATA PAGE
          LDI    @SENSOR_ADDR,AR0       ; SET POINTER TO SENSOR[]
          LDI    @RANDOM_ADDR,AR1       ; SET POINTER TO RANDOM[]

          LDI    0,R0                   ; R0 = 0
          RPTS   M                      ; REPEAT M+1 TIMES
          STI    R0,*AR0++(1)     ;

          LDI    @SEED,R0         ;
          LDI    M,RC                   ;

          RPTB   RNDN

          LDI    @END,R1
          AND    R0,R1
          LDI    @MID,R2
          AND    R0,R2
          LDI    -27,R3
          LSH    R3,R2
          LDI    -30,R3
          LSH    R3,R1
          XOR    R1,R2
          LDI    1,R3
          LSH    R3,R0
          OR3    R0,R2,R0

          LDF    @NORM_1,R3       ; LOAD VALUE STORED AT NORM_1
          FLOAT  R2,R1
          SUBF   0.5,R1
          MPYF   3,R1
          MPYF3  R3,R1,R2         ; CONVERT (+-)3V TO ORIGINAL FORM

          BNN    POSITIVE

          FIX    R2,R3
          LDI    0FFFFh,R1
          ADDI   R1,R3

          BR     SENDOUT

POSITIVE:

          FIX    R2,R3                  ; CONVERT TO INTEGER

SENDOUT:
          LDI    16,R1                  ; 0000XXXX / FFFFXXXX => XXXX0000
          ASH     R1,R3

          STI    R3,*AR1++(1)

RNDN    NOP


*
* PREPARE FOR ID ROUTINE
*

          LDI    @VALUES,R0             ; SET USER CONTROL REGISTER
          LDI    @UCR,AR0               ; FOR BURR_BROWN DAUGHTER
MOD.
          STI    R0,*AR0          ; VAL = A4000000h

          LDI    @VALUES+1,R0           ; SET TIMER 1 REGISTER
          LDI    @TM1,AR0         ; FOR BURR_BROWN DAUGHTER MOD.
          STI    R0,*AR0          ; VAL = E8010000h ( 2 kHz )

          LDI    @VALUES+2,R0           ; SET AMELIA CONTROL REGISTER
          LDI    @ACR,AR0               ; FOR BURR_BROWN DAUGHTER
MOD.
          STI    R0,*AR0          ; VAL = 00B20000h

          LDI    @VALUES+3,R0           ; SET CONFIGURATION REGISTER
          LDI    @CFR,AR0         ; FOR BURR_BROWN DAUGHTER MOD.
          STI    R0,*AR0          ; VAL = 8DFF0000h

          LDI    @VALUES+4,R0           ; SET INTERRUPT MASK REGISTER
          LDI    @IMR,AR0         ; FOR BURR_BROWN DAUGHTER MOD.
          STI    R0,*AR0          ; VAL = 00010000h

          LDI    @RANDOM_ADDR,AR0       ; SET POINTER TO RANDOM[]
          LDI    @SENSOR_ADDR,AR1       ;  SET POINTER TO SENSOR[]
          LDI    M,RC                   ; M INTO REPEAT COUNTER

          OR     01h,IE                 ; ENABLE INTERRUPT 0

          RPTB    LOOP            ;

          LDI    @CH_0,AR2              ; AR2 CONTAINS I/O ADDR OF CH 0
          LDI    *AR0++(1),R1           ; LOAD VALUE STORED AT
RANDOM[M]
```

```
        STI    R1,*AR2               ; OUTPUT THROUGH D/A

        PUSH   RS                            ; SAVE REPEAT START
        PUSH   RE                            ; SAVE REPEAT END

        OR     2000h,ST              ; ENABLE ALL INTERRUPTS SET IN IE REG.
        IDLE                                 ; WAIT FOR INTERRUPT
        AND     1FFFh,ST             ; DISABLE ALL INTERRUPTS

        POP    RE                            ; RESTORE REPEAT END
        POP    RS                            ; RESTORE REPEAT START
LOOP   NOP

        AND     0FFEh,IE             ; DISABLE INTERRUPT ZERO

        NOP
        NOP

WAIT   BR     WAIT                           ; END IN TERMINATION LOOP

*********************************************************************
***
ISR0
        PUSH   ST
        PUSH   DP
        PUSH   IE

        LDI    @IMR,AR2              ; IMR FOR BURR_BROWN DAUGHTER MOD.
        LDI    *AR2,R0               ; READ INTERRUPT MASK TO CLEAR

        AND     0FFFEh,IF            ; RESET INTERRUPT FLAG

        LDI    @CH_0,AR2                     ; AR2 CONTAINS I/O ADDR OF CH 0
        LDI    *AR2,R0               ; READ VALUE FROM A/D PORT
        STI    R0,*AR1++(1)          ; STORE VALUE IN SENSOR[M]

        POP    IE
        POP    DP
        POP    ST

        RETI                                 ; RETURN FROM INTERRUPT

*********************************************************************
***

ISR1   RETI  ; INT1
```

```
ISR2    RETI  ; INT2
ISR3    RETI  ; INT3
ISR4    RETI  ; XINT0
ISR5    RETI  ; RINT0
ISR6    RETI  ; TINT0
ISR7    RETI  ; TINT1
ISR8    RETI  ; DINT0

***********************************************************************
***

        .end
```

A.2 BIN2ASCI.C

```
#include <stdio.h>
#include <io.h>
#include <ctype.h>

main (argc,argv)

char *argv[];
int argc;

{
  FILE *fopen(), *fp_in, *fp_out;
  char ch;

  if (argc != 3)
  {
    printf ("correct usage is:\n");
    printf ("bin2asci <input_file> <output_file>\n");
    exit (0);
  }

  fp_in= fopen(argv[1],"rb");
  fp_out= fopen(argv[2],"w");

  if (fp_in == NULL)
  {
    printf ("input file could not be opened.\n");
    exit (1);
  }

  convert_file (fp_in,fp_out);

  printf ("DONE\n");

  fclose (fp_in);
  fclose (fp_out);
  return (1);
}

convert_file (input,output)

FILE *input, *output;

{
  unsigned short ch[2];
```

```
   long length;
   float out;
   int i,hold,handle;

   printf ("\n");

   handle=fileno(input);
   length=filelength(handle);

  printf ("Size of binary file: %lu\n",length);
  printf ("Please wait . . . \n\n");

   length=length / 4;

   for (i=0; i<length; i++)
   {
      fread (ch, sizeof(unsigned short), 2, input);

      if ( ch[1] > 32767 )
      {
            hold=ch[1] - 65535;
      }
      else
      {
            hold=ch[1];
      }

      out=hold/10922.5;

      fprintf (output,"%f\n",out);

   }

}
```

A.3 ETFE.M

```
%XX=1; %% use Control input to output program flow
XX=-1; %% use Disturbance input to output program flow

sizeof=16384;

if XX>0
  load c:\matlab\research\data\cinput.dat
  u=cinput(1:sizeof);
  clear cinput;

  load c:\matlab\research\data\coutput.dat
  y=coutput(1:sizeof);
  clear coutput;
else
  load c:\matlab\research\data\dinput.dat
  u=dinput(1:sizeof);
  clear dinput;

  load c:\matlab\research\data\doutput.dat
  y=doutput(1:sizeof);
  clear doutput;
end

%%% Empirical Transfer Function Estimate %%%

M=512;
N=4096;
T=1;

ghh_y=etfe([ y u ],M,N,T);
[freq_vect,amp,ang] = getff(ghh_y,1,1);
freq_vect=freq_vect.*(2000/(2*pi));

figure
semilogy(freq_vect,amp,'y')
if XX>0
  title('Control Input to Sensor Output (ETFE)')
else
  title('Disturbance Input to Sensor Output (ETFE)')
end
xlabel('Frequency (Hz)')
ylabel('Magnitude')
axis([0 1000 1e-4 10])
```

```
figure
plot(freq_vect,ang,'y')
if XX>0
  title('Control Input to Sensor Output (ETFE)')
else
  title('Disturbance Input to Sensor Output (ETFE)')
end
xlabel('Frequency (Hz)')
ylabel('Angle (degrees)')
```

A.4 IDENT.M

```
%%% Initial Model Orders (ARX Model) %%%

if XX>0
  na=34; %%% Control %%%
  nb=34;  % 34 %
  nk=0;
else
  na=34; %%% Disturbance %%%
  nb=34;  % 34 %
  nk=0;
end

%%% Least Squares Fit %%%

th_ls_y=arx([y u],[na nb nk]);
[num_ls_y,den_ls_y]=th2tf(th_ls_y);

[a_ls,b_ls,c_ls,d_ls,k_ls,x0_ls]=th2ss(th_ls_y);
clear k_ls; clear x0_ls;
clear th_ls_y;

max(abs(roots(den_ls_y)))

%%% Least Squares Frequency Response %%%

[h_ls_y,w]=freqz(num_ls_y,den_ls_y,N);
mag_ls_y=abs(h_ls_y);
phase_ls_y=(unwrap(angle(h_ls_y))*(180/pi));

clear num_ls_y;
clear den_ls_y;
%clear h_ls_y;

%%% Plot %%%
```

```
figure
semilogy(freq_vect,amp,'y')
hold on
semilogy(freq_vect,mag_ls_y,'g')
hold off
if XX>0
  title('Control Input to Sensor Output (Least Squares Fit)')
else
  title('Disturbance Input to Sensor Output (Least Squares Fit)')
end
xlabel('Frequency (Hz)')
ylabel('Magnitude')
axis([0 1000 1e-4 10])

figure
plot(freq_vect,ang,'y')
hold on
plot(freq_vect,phase_ls_y,'g')
hold off
if XX>0
  title('Control Input to Sensor Output (Least Squares Fit)')
else
  title('Disturbance Input to Sensor Output (Least Squares Fit)')
end
xlabel('Frequency (Hz)')
ylabel('Angle (degrees)')
```

A.5 RDUCE.M

```
%%% Discrete Balanced Realization %%%

Gc = dgram(a_ls,b_ls);
Go = dgram(a_ls',c_ls');

[Vx,Dx]=eig(Gc);
Gc_pd=Vx*abs(Dx)*inv(Vx);

R=chol(Gc_pd);

[ux,D,V] = svd(R*Go*R');
D = D.*sign(ux'*V);

Trans = R'*V*diag(diag(D).^(-.25));
a_red = Trans\a_ls*Trans;
b_red = Trans\b_ls;
c_red = c_ls*Trans;

gram_diag = diag(dgram(a_red,b_red))';

clear Dx; clear Vx; clear ux;
clear R;  clear D;  clear V;
clear Gc_pd; clear Gc; clear Go;

%%% Discrete Model Reduction %%%

if XX>0
  [ab,bb,cb,db]=dmodred(a_red,b_red,c_red,d_ls,[29,30,31,32,33,34]);
else
  [ab,bb,cb,db]=dmodred(a_red,b_red,c_red,d_ls,[29,30,31,32,33,34]);
end

%clear a_red; clear b_red; clear c_red;
%clear a_iv; clear b_iv; clear c_iv; clear d_iv;

%%% Frequency Response %%%

[top,bot]=ss2tf(ab,bb,cb,db);

[h,w]=freqz(top,bot,N);
red_mag=abs(h); red_ang=unwrap(angle(h))*(180/pi);

max(abs(roots(bot)))
clear h; clear top;
```

```
if XX>0
  [num_c,den_c]=ss2tf(ab,bb,cb,db);
else
  [num_d,den_d]=ss2tf(ab,bb,cb,db);
end

%%% Plot %%%

figure
semilogy(freq_vect,amp,'y')
hold on
semilogy(freq_vect,red_mag,'m')
hold off
if XX>0
  title('Control Input to Sensor Output (reduced to 28th order)')
else
  title('Disturbance Input to Sensor Output (reduced to 28th order)')
end
xlabel('Frequency (Hz)')
ylabel('Magnitude')
axis([0 1000 1e-4 10])

figure
plot(freq_vect,ang,'y')
hold on
plot(freq_vect,red_ang,'m')
hold off
if XX>0
  title('Control Input to Sensor Output (reduced to 28th order)')
else
  title('Disturbance Input to Sensor Output (reduced to 28th order)')
end
xlabel('Frequency (Hz)')
ylabel('Angle (degrees)')
```

A.6 SVMOD.M

```
if XX>0
  save c:\matlab\research\model\Ac.dat ab -ascii -tabs;
  save c:\matlab\research\model\Bc.dat bb -ascii -tabs;
  save c:\matlab\research\model\Cc.dat cb -ascii -tabs;
  save c:\matlab\research\model\Dc.dat db -ascii -tabs;
else
  save c:\matlab\research\model\Ad.dat ab -ascii -tabs;
  save c:\matlab\research\model\Bd.dat bb -ascii -tabs;
  save c:\matlab\research\model\Cd.dat cb -ascii -tabs;
  save c:\matlab\research\model\Dd.dat db -ascii -tabs;
end
```

A.7 SAVTF.M

```
N=4096;
ld_1x_md;

%%%%% Finding the Value of K %%%%%

R=0.96
K=dlqr(Ac,Bc,Cc'*Cc,R);  %% 0.960

%%%%% Finding the Value of L %%%%%

L=Ac*(dlqe(Ac,Bc,Cc,1,1e-4));

fresp2
%break

%%%%% Generate Transfer Functions %%%%%

[num_uy,den_uy]=ss2tf(Ac,Bc,Cc,0);
[num_dy,den_dy]=ss2tf(Ad,Bd,Cd,0);
[num_uz,den_uz]=ss2tf(Ac-L*Cc,Bc,K,0);
[num_yz,den_yz]=ss2tf(Ac-L*Cc,L,K,0);
[num_uyh,den_uyh]=ss2tf(Ac-L*Cc,Bc,Cc,0);
[num_yyh,den_yyh]=ss2tf(Ac-L*Cc,L,Cc,0);

save c:\matlab\research\model\num_dy.dat num_dy -ascii -tabs;
save c:\matlab\research\model\den_dy.dat den_dy -ascii -tabs;

save c:\matlab\research\model\num_uy.dat num_uy -ascii -tabs;
save c:\matlab\research\model\den_uy.dat den_uy -ascii -tabs;
save c:\matlab\research\model\num_uz.dat num_uz -ascii -tabs;
save c:\matlab\research\model\den_uz.dat den_uz -ascii -tabs;

save c:\matlab\research\model\num_yz.dat num_yz -ascii -tabs;
save c:\matlab\research\model\den_yz.dat den_yz -ascii -tabs;

save c:\matlab\research\model\num_uyh.dat num_uyh -ascii -tabs;
save c:\matlab\research\model\den_uyh.dat den_uyh -ascii -tabs;

save c:\matlab\research\model\num_yyh.dat num_yyh -ascii -tabs;
save c:\matlab\research\model\den_yyh.dat den_yyh -ascii -tabs;
```

Appendix B

B.1 ADAPT.C for LMS code

```
/*******************************************************/
/*                                                     */
/*  Adaptive LMS Routine                               */
/*                                                     */
/*******************************************************/

#include <math.h>
#include <stdlib.h>
#include "tfmod8.h"
#include "initvecs.h"

/*** defined constants ***/

#define  M          15      /* order of lattice filter */
#define  ORDER        28      /* order of system */

/*** variables ***/

float  mu;

float nu[M];
float ux[M];
float uf[M];

float  u, y, s;

int    seed;
int    i;

/*** function prototypes ***/

extern void init_io(void);
extern void filter(void);

/*******************************************************/
/*                                                     */
/*  Main Program: Adapt.c                              */
/*                                                     */
/*******************************************************/

void main(void)
{
  mu = 0.75;

  seed = 16671549;

  for (i=0; i < M; i++)
  {
    nu[i]=0.0;
    ux[i]=0.0;
    uf[i]=0.0;
  }

  u = 0.0;
  y = 0.0;
  s = 0.0;

  i = 1;

  init_io();

  while (i > 0)
  {
    filter();
  }

}
```

B.2 INITVECS.H for LMS code

```
float u_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float y_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float za_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float zb_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float yha_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float yhb_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float r_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float rb_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float rp_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
```

B.3 TFMOD8.H for LMS code

```
const float num_uy[29] =  {  0.0000000e+000, -1.5904315e-004,  4.2409415e-004, -4.9284109e-002,
1.9893541e-001, -3.1758832e-001,  3.1991080e-001, -2.5978062e-001,  1.6495562e-001,
1.3093236e-002, -2.6836145e-001,  4.0050110e-001, -3.8240303e-001,  3.3581286e-001,
-2.5461830e-001,  9.4364512e-002,  9.4201270e-002, -1.9247663e-001,  2.0855357e-001,
-2.0134025e-001,  1.4533480e-001, -4.3740757e-002, -3.3162386e-002,  3.8873874e-002,
-2.9871692e-002,  3.2910294e-002, -3.7478365e-002,  2.3224869e-002, -9.5386817e-004};

const float den_uy[29] =   {  1.0000000e+000, -3.2865226e+000,  4.6603098e+000, -3.2893508e+000,
4.1916269e-001,  1.6767313e+000, -3.1235753e+000,  4.3896353e+000, -3.9919281e+000,
1.0311235e+000,  2.2520882e+000, -3.7354589e+000,  4.1809189e+000, -4.2092994e+000,
2.8734239e+000,  4.5056639e-002, -2.5650890e+000,  3.2889473e+000, -2.7153134e+000,
1.4414222e+000,  8.5999084e-002, -1.1600308e+000,  1.1084441e+000, -3.9557971e-001,
-1.2658787e-002, -1.2413373e-001,  2.8652604e-001, -1.8344510e-001,  5.6467152e-002};

const float num_uz[29] =  {  0.0000000e+000,  4.4880500e-002, -7.8615094e-002,  3.5876926e-002,
1.7895885e-003, -7.4897554e-003,  1.7182858e-002, -6.9484365e-002,  8.1450245e-002,
-1.6089545e-002, -6.8375095e-004, -1.4705265e-002, -1.0106208e-002,  4.9611529e-002,
-4.2922180e-002,  3.5838479e-003, -2.6290233e-004,  4.8756548e-003,  1.3150029e-002,
-2.3821924e-002,  1.0976638e-003,  1.3481585e-002, -1.0471705e-003, -6.0964111e-003,
3.2565513e-003,  6.9405164e-003, -1.8129320e-003, -4.1746617e-003,  1.4782862e-004};

const float den_uz[29] =   {  1.0000000e+000, -2.6732026e+000,  3.4256695e+000, -3.0453783e+000,
2.1656934e+000, -1.0075518e+000, -1.1942929e+000,  3.4116253e+000, -4.0199864e+000,
3.6430676e+000, -3.0497623e+000,  1.9259230e+000, -4.0241287e-002, -1.5552926e+000,
2.0543575e+000, -2.0193335e+000,  1.7315302e+000, -1.0097455e+000,  3.4233749e-002,
4.0383223e-001, -3.4999218e-001,  2.7868387e-001, -3.2353246e-001,  3.2916130e-001,
-1.1928593e-001,  3.3412439e-003,  3.3099793e-003, -2.1129510e-003,  5.5094230e-004};

const float num_yz[29] =  {  0.0000000e+000, -1.9824451e-001,  6.5783635e-001, -7.1735758e-001,
2.5201745e-001,  4.6119829e-001, -1.1576058e+000,  1.6506510e+000, -2.0641613e+000,
1.8259873e+000, -9.6363275e-001, -2.5721269e-002,  7.9107211e-001, -1.2764404e+000,
1.5898900e+000, -1.3052523e+000,  6.1665461e-001,  7.2693068e-002, -4.5457441e-001,
5.8160015e-001, -4.9605233e-001,  2.2580249e-001, -4.6555229e-002, -8.4845841e-003,
-7.5207262e-002,  8.1771196e-002, -1.2539931e-002, -1.4700859e-002,  1.1441345e-002};

const float den_yz[29] =   {  1.0000000e+000, -2.6732026e+000,  3.4256695e+000, -3.0453783e+000,
2.1656934e+000, -1.0075518e+000, -1.1942929e+000,  3.4116253e+000, -4.0199864e+000,
3.6430676e+000, -3.0497623e+000,  1.9259230e+000, -4.0241287e-002, -1.5552926e+000,
2.0543575e+000, -2.0193335e+000,  1.7315302e+000, -1.0097455e+000,  3.4233749e-002,
4.0383223e-001, -3.4999218e-001,  2.7868387e-001, -3.2353246e-001,  3.2916130e-001,
-1.1928593e-001,  3.3412439e-003,  3.3099793e-003, -2.1129510e-003,  5.5094230e-004};

const float num_uyh[29] ={  0.0000000e+000, -1.5904315e-004,  4.2409415e-004, -4.9284109e-002,
1.9893541e-001, -3.1758832e-001,  3.1991080e-001, -2.5978062e-001,  1.6495562e-001,
1.3093236e-002, -2.6836145e-001,  4.0050110e-001, -3.8240303e-001,  3.3581286e-001,
-2.5461830e-001,  9.4364512e-002,  9.4201270e-002, -1.9247663e-001,  2.0855357e-001,
-2.0134025e-001,  1.4533480e-001, -4.3740757e-002, -3.3162386e-002,  3.8873874e-002,
-2.9871692e-002,  3.2910294e-002, -3.7478365e-002,  2.3224869e-002, -9.5386817e-004};

const float den_uyh[29] = {  1.0000000e+000, -2.6732026e+000,  3.4256695e+000, -3.0453783e+000,
2.1656934e+000, -1.0075518e+000, -1.1942929e+000,  3.4116253e+000, -4.0199864e+000,
3.6430676e+000, -3.0497623e+000,  1.9259230e+000, -4.0241287e-002, -1.5552926e+000,
2.0543575e+000, -2.0193335e+000,  1.7315302e+000, -1.0097455e+000,  3.4233749e-002,
4.0383223e-001, -3.4999218e-001,  2.7868387e-001, -3.2353246e-001,  3.2916130e-001,
-1.1928593e-001,  3.3412439e-003,  3.3099793e-003, -2.1129510e-003,  5.5094230e-004};

const float num_yyh[29] ={  0.0000000e+000,  6.1331992e-001, -1.2346403e+000,  2.4397244e-001,
1.7465307e+000, -2.6842832e+000,  1.9292824e+000, -9.7800999e-001, -2.8058302e-002,
2.6119441e+000, -5.3018505e+000,  5.6613819e+000, -4.2211602e+000,  2.6540068e+000,
-8.1906648e-001, -2.0643901e+000,  4.2966192e+000, -4.2986929e+000,  2.7495472e+000,
-1.0375900e+000, -4.3599127e-001,  1.4387147e+000, -1.4319766e+000,  7.2474101e-001,
-1.0662714e-001,  1.2747497e-001, -2.8321606e-001,  1.8133215e-001, -5.5916210e-002};

const float den_yyh[29] = {  1.0000000e+000, -2.6732026e+000,  3.4256695e+000, -3.0453783e+000,
2.1656934e+000, -1.0075518e+000, -1.1942929e+000,  3.4116253e+000, -4.0199864e+000,
3.6430676e+000, -3.0497623e+000,  1.9259230e+000, -4.0241287e-002, -1.5552926e+000,
2.0543575e+000, -2.0193335e+000,  1.7315302e+000, -1.0097455e+000,  3.4233749e-002,
4.0383223e-001, -3.4999218e-001,  2.7868387e-001, -3.2353246e-001,  3.2916130e-001,
-1.1928593e-001,  3.3412439e-003,  3.3099793e-003, -2.1129510e-003,  5.5094230e-004};

const float num_hp[7] =  { 7.60790003e-001, -4.56474002e+000, 1.14118500e+001, -1.52158000e+001,
1.14118500e+001, -4.5647400e+000, 7.60790003e-001 };

const float den_hp[7] =  { 1.00000000e+000, -5.45387055e+000, 1.24164881e+001, -1.51024974e+001,
1.03499935e+001, -3.78890907e+000, 5.78801429e-001 };
```

B.4 INITIO.ASM for LMS code

```
*****************************************************************
*
* INITALIZE I/O ROUTINE
*
*****************************************************************

        .global      _init_io

*****************************************************************
*
* Set C31 for LSI Control
*
*****************************************************************
               .data

BUSADDR .word   00808064h      ; ADDRESS OF BUS CONTROL REGISTER
BUSDATA .word   00000900h      ; VALUE FOR LSI CARD GIVES ZERO
                               ; WAIT STATES, 32k BANK SIZE,
                               ; EXTERNAL READY CONTROL

*****************************************************************
*
* INITIALIZE CONSTANTS
*
*****************************************************************

CH_0         .word   550002h       ; CH_0 ADDRESS
TM1          .word   550005h       ; TIMER 1 REGISTER
CH_1         .word   550006h       ; CH_1 ADDRESS
UCR          .word   550008h       ; USER CONTROL REG.
ACR          .word   55000Ah       ; AMELIA CONTROL REG.
IMR          .word   55000Bh       ; INTERRUPT MASK REG.
CFR          .word   55000Fh              ; CONFIGURATION REGISTER

VALUES         .word   0A4000000h         ; A4000000h
               .word   0E8010000h         ; E8010000h 2 kHz
*              .word   0FA010000h         ; FA010000h 8 kHz
               .word   000B20000h         ; 00B20000h
               .word   08DFF0000h         ; 8DFF0000h
               .word   000010000h         ; 00010000h

*****************************************************************
*
* INITALIZE I/O Routine
*
*****************************************************************
               .text

_init_io:

        LDP     BUSADDR,DP              ; DP POINTER SET PRIOR TO
                                        ; USING DIR. ADDR. MODE
        LDI     @BUSADDR,AR0            ;
        LDI     @BUSDATA,R0             ;
        STI     R0,*AR0                 ; SET BUS. REG. FOR LSI CARD

        LDP     @VALUES,DP              ; SET DATA PAGE POINTER
        LDI     @VALUES,R0              ; SET USER CONTROL REGISTER
        LDI     @UCR,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = A4000000h

        LDI     @VALUES+1,R0            ; SET TIMER 1 REGISTER
        LDI     @TM1,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = FFC40000h

        LDI     @VALUES+2,R0            ; SET AMELIA CONTROL REGISTER
        LDI     @ACR,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = 00B20000h

        LDI     @VALUES+3,R0            ; SET CONFIGURATION REGISTER
        LDI     @CFR,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = 8DFF0000h

        LDI     @VALUES+4,R0            ; SET INTERRUPT MASK REGISTER
        LDI     @IMR,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = 00010000h

        RETS

***************************************************************
```

B.5 FILTER.ASM for LMS code

```
*****************************************************************
*
* Adaptive-Q Filter Routine
*
*****************************************************************

        .global         _filter
        .global         _mu, _y, _u
        .global         ISR0,ISR1,ISR2,ISR3

        .global         _num_uy, _den_uy
        .global         _num_uz, _den_uz
        .global         _num_yz, _den_yz
        .global         _num_uyh, _den_uyh
        .global         _num_yyh, _den_yyh
        .global         _num_hp, _den_hp

        .global         _za_vect, _zb_vect
        .global         _yha_vect, _yhb_vect
        .global         _u_vect, _y_vect
        .global         _r_vect, _s, _seed
        .global         _rb_vect, _rp_vect

        .global         _nu
        .global         _ux, _uf

*****************************************************************
*
* Interrupt Vector Specifications
*
*****************************************************************

        .sect   ".int0"
        BR      ISR0
        .sect   ".int1"
        BR      ISR1
        .sect   ".int2"
        BR      ISR2
        .sect   ".int3"
        BR      ISR3

*****************************************************************
*
* Vector Addresses & Constants
*
*****************************************************************
        .data

ord             .set    28              ;
M               .set    15              ;

NORM_1      .float  1.09225e+4          ; REPRESENTS 1V
NORM_2      .float  9.155552843e-5      ;
```

```
END           .word   40000000h                 ;
MID           .word   08000000h                 ;

IMR           .word   55000Bh                   ; INTERRUPT MASK REG.
CH_0          .word   550002h                   ; CH_0 ADDRESS
CH_1          .word   550006h                   ; CH_1 ADDRESS

numuy_addr       .word     _num_uy              ;
denuy_addr       .word     _den_uy              ;
numuz_addr       .word     _num_uz              ;
denuz_addr       .word     _den_uz              ;
numyz_addr       .word     _num_yz              ;
denyz_addr       .word     _den_yz              ;
numuyh_addr      .word     _num_uyh             ;
denuyh_addr      .word     _den_uyh             ;
numyyh_addr      .word     _num_yyh             ;
denyyh_addr      .word     _den_yyh             ;
numhp_addr       .word     _num_hp              ;
denhp_addr       .word     _den_hp              ;

yvect_addr       .word     _y_vect              ;
uvect_addr       .word     _u_vect              ;
zavect_addr      .word     _za_vect             ;
zbvect_addr      .word     _zb_vect             ;
yhavect_addr     .word     _yha_vect            ;
yhbvect_addr     .word     _yhb_vect            ;
rvect_addr       .word     _r_vect              ;
rpvect_addr      .word     _rp_vect             ;
rbvect_addr      .word     _rb_vect             ;

nu_addr          .word     _nu                  ;
ux_addr          .word     _ux                  ;
uf_addr          .word     _uf                  ;

******************************************************************
*
* filter Routine
*
******************************************************************
                 .text

_filter:

*********** Output U & Input Y ***********************************

        OR      01h,IE                          ; ENABLE INTERRUPT 0
                                                ;
        OR      2000h,ST                        ; ENABLE ALL INT IN IE REG
        IDLE                                    ; WAIT FOR INTERRUPT
        AND     1FFFh,ST                        ; RESET GIE TO ZERO
                                                ;
        AND     0FFEh,IE                        ; DISABLE INTERRUPT ZERO

******************************************************************
```

```
          LDI    @zavect_addr,AR0         ;
          LDI    @zbvect_addr,AR1         ;
          LDI    @yhavect_addr,AR2        ;
          LDI    @yhbvect_addr,AR3        ;
          LDI    @rvect_addr,AR4          ;
          LDI    @rbvect_addr,AR5         ;
          LDI    @rpvect_addr,AR6         ;
          LDI    @yvect_addr,AR7          ;
                                          ;
          LDI    ord-1,IR0                ;
          LDI    ord,IR1                  ;
                                          ;
          LDI    ord-1,RC                 ;
          RPTB   shift1                   ;
          LDF    *+AR0(IR0),R0            ;
       || LDF    *+AR1(IR0),R1            ;
          LDF    *+AR2(IR0),R2            ;
       || LDF    *+AR3(IR0),R3            ;
          LDF    *+AR4(IR0),R4            ;
       || LDF    *+AR5(IR0),R5            ;
          LDF    *+AR6(IR0),R6            ;
       || LDF    *+AR7(IR0),R7            ;
          STF    R0,*+AR0(IR1)            ;
       || STF    R1,*+AR1(IR1)            ;
          STF    R2,*+AR2(IR1)            ;
       || STF    R3,*+AR3(IR1)            ;
          STF    R4,*+AR4(IR1)            ;
       || STF    R5,*+AR5(IR1)            ;
          STF    R6,*+AR6(IR1)            ;
       || STF    R7,*+AR7(IR1)            ;
          SUBI   1,IR0                    ;
shift1:   SUBI   1,IR1                    ;
                                          ;
          LDI    @yvect_addr,AR0          ;
                                          ;
          LDF    @_y,R0                   ;
          STF    R0,*AR0                  ;

*************** TF from U to Za *********************************

          LDI    @numuz_addr,AR0                  ;
          LDI    @uvect_addr,AR1                  ;
          LDI    @denuz_addr,AR2                  ;
          LDI    @zavect_addr,AR3                 ;
                                                  ;
          LDF    0.0,R0                           ;
          LDF    0.0,R2                           ;
                                                  ;
          LDI    ord,RC                           ;
          RPTS                                    ;
          MPYF3  *AR0++(1),*AR1++(1),R0           ;
       || ADDF3  R0,R2,R2                         ;
          ADDF3  R0,R2,R2                         ;
                                                  ;
          LDF    0.0,R0                           ;
                                                  ;
          LDI     ord-1,RC                        ;
          RPTS                                    ;
          MPYF3  *++AR2(1),*++AR3(1),R0           ;
       || SUBF3  R0,R2,R2                         ;
          SUBF3  R0,R2,R2                         ;
                                                  ;
          STF     R2,@_za_vect                    ;

**************** TF from Y to Zb *********************************

          LDI     @numyz_addr,AR0                 ;
          LDI     @yvect_addr,AR1                 ;
          LDI     @denyz_addr,AR2                 ;
          LDI     @zbvect_addr,AR3                ;
                                                  ;
          LDF     0.0,R0                          ;
          LDF     0.0,R2                          ;
                                                  ;
          LDI     ord,RC                          ;
          RPTS                                    ;
          MPYF3  *AR0++(1),*AR1++(1),R0           ;
       || ADDF3  R0,R2,R2                         ;
          ADDF3  R0,R2,R2                         ;
                                                  ;
          LDF     0.0,R0                          ;
                                                  ;
          LDI     ord-1,RC                        ;
          RPTS                                    ;
          MPYF3  *++AR2(1),*++AR3(1),R0           ;
       || SUBF3  R0,R2,R2                         ;
          SUBF3  R0,R2,R2                         ;
                                                  ;
          STF     R2,@_zb_vect                    ;

**************** TF from U to Y_HAT *****************************

          LDI     @numuyh_addr,AR0                ;
          LDI     @uvect_addr,AR1                 ;
          LDI     @denuyh_addr,AR2                ;
          LDI     @yhavect_addr,AR3               ;
                                                  ;
          LDF     0.0,R0                          ;
          LDF     0.0,R2                          ;
                                                  ;
          LDI     ord,RC                          ;
          RPTS                                    ;
          MPYF3  *AR0++(1),*AR1++(1),R0           ;
       || ADDF3  R0,R2,R2                         ;
          ADDF3  R0,R2,R2                         ;
                                                  ;
          LDF     0.0,R0                          ;
                                                  ;
          LDI     ord-1,RC                        ;
          RPTS                                    ;
          MPYF3  *++AR2(1),*++AR3(1),R0           ;
       || SUBF3  R0,R2,R2                         ;
```

```
            SUBF3  R0,R2,R2                          ;
                                                     ;
            STF    R2,@_yha_vect                     ;

**************** TF from Y to Y_HAT ******************************

            LDI    @numyyh_addr,AR0                  ;
            LDI    @yvect_addr,AR1                   ;
            LDI    @denyyh_addr,AR2                  ;
            LDI    @yhbvect_addr,AR3                 ;
                                                     ;
            LDF    0.0,R0                            ;
            LDF    0.0,R2                            ;
                                                     ;
            LDI    ord,RC                            ;
            RPTS                                     ;
            MPYF3  *AR0++(1),*AR1++(1),R0            ;
         || ADDF3  R0,R2,R2                          ;
            ADDF3  R0,R2,R2                          ;
                                                     ;
            LDF    0.0,R0                            ;
                                                     ;
            LDI    ord-1,RC                          ;
            RPTS                                     ;
            MPYF3  *++AR2(1),*++AR3(1),R0            ;
         || SUBF3  R0,R2,R2                          ;
            SUBF3  R0,R2,R2                          ;
                                                     ;
            STF    R2,@_yhb_vect                     ;

**************** Y - Y_HAT **************************************

            LDF    @_yha_vect,R0                     ;
            LDF    @_yhb_vect,R1                     ;
            LDF    @_y_vect,R3                       ;
            ADDF   R0,R1                             ;
            SUBF   R1,R3                             ;
            STF    R3,@_r_vect                       ;

**************** TF from R to RX *********************************

            LDI    @numhp_addr,AR0                   ;
            LDI    @rvect_addr,AR1                   ;
            LDI    @denhp_addr,AR2                   ;
            LDI    @rbvect_addr,AR3                  ;
                                                     ;
            LDF    0.0,R0                            ;
            LDF    0.0,R2                            ;
                                                     ;
            LDI    6,RC                              ;
            RPTS                                     ;
            MPYF3  *AR0++(1),*AR1++(1),R0            ;
         || ADDF3  R0,R2,R2                          ;
            ADDF3  R0,R2,R2                          ;
                                                     ;
            LDF    0.0,R0                            ;
```

```
                                                ;
          LDI      5,RC                         ;
          RPTS                                  ;
          MPYF3  *++AR2(1),*++AR3(1),R0         ;
     ||   SUBF3  R0,R2,R2                       ;
          SUBF3  R0,R2,R2                       ;
                                                ;
          STF      R2,@_rb_vect                 ;


**************** TF from R to R_BAR *******************************

          LDI      @numuy_addr,AR0              ;
          LDI      @rbvect_addr,AR1             ;
          LDI      @denuy_addr,AR2              ;
          LDI      @rpvect_addr,AR3             ;
                                                ;
          LDF      0.0,R0                       ;
          LDF      0.0,R2                       ;
                                                ;
          LDI      ord,RC                       ;
          RPTS                                  ;
          MPYF3  *AR0++(1),*AR1++(1),R0         ;
        || ADDF3  R0,R2,R2                      ;
          ADDF3  R0,R2,R2                       ;
                                                ;
          LDF      0.0,R0                       ;
                                                ;
          LDI      ord-1,RC                     ;
          RPTS                                  ;
          MPYF3  *++AR2(1),*++AR3(1),R0         ;
        || SUBF3  R0,R2,R2                      ;
          SUBF3  R0,R2,R3                       ;
                                                ;
          STF      R3,@_rp_vect                 ;

***** W = W + MU * ERR * UX *************************************

          LDI      @ux_addr,AR0                 ;
          LDI      @nu_addr,AR1                 ;
                                                ;
          LDF      @_mu,R0                      ;
          LDF      @_y_vect,R2                  ;
          MPYF   R2,R0                          ;
                                                ;
          LDI      M-1,RC                       ;
          RPTB    WLOOP                         ;
          MPYF3  R0,*AR0++(1),R2                ;
          ADDF   *AR1,R2                        ;
WLOOP:             STF      R2,*AR1++(1)        ;

*****************************************************************

          LDI      @ux_addr,AR0                 ;
          LDI      @uf_addr,AR1                 ;
                                                ;
```

```
            LDI     M-2,IR0                         ;
            LDI     M-1,IR1                         ;
                                                    ;
            LDI     M-2,RC                          ;
            RPTB    shift4                          ;
            LDF     *+AR0(IR0),R0                   ;
         || LDF     *+AR1(IR0),R1                   ;
            STF     R0,*+AR0(IR1)                   ;
         || STF     R1,*+AR1(IR1)                   ;
            SUBI    1,IR0                           ;
shift4:     SUBI    1,IR1                           ;

*           LDF     0.0,R0                          ;
            LDF     @_rp_vect,R0                    ;
*           SUBF    R1,R0                           ;
            STF     R0,@_ux                         ;

            LDF     @_rb_vect,R0                    ;
            STF     R0,@_uf                         ;

*****************************************************************

            LDI     @nu_addr,AR0                    ;
            LDI     @uf_addr,AR1                    ;

            LDF     0.0,R0                          ;
            LDF     0.0,R2                          ;

            LDI     M-1,RC                          ;
            RPTS                                    ;
            MPYF3   *AR0++(1),*AR1++(1),R0          ;
         || ADDF3   R0,R2,R2                        ;
            ADDF3   R0,R2,R2                        ;

            STF     R2,@_s                          ;

*****************************************************************

            LDI     @uvect_addr,AR0                 ;
                                                    ;
            LDI     ord-1,IR0                       ;
            LDI     ord,IR1                         ;
                                                    ;
            LDI     ord-1,RC                        ;
            RPTB    shift5                          ;
            LDF     *+AR0(IR0),R0                   ;
            STF     R0,*+AR0(IR1)                   ;
            SUBI    1,IR0                           ;
shift5:     SUBI    1,IR1                           ;

********* U_VECT = -(S+(Za+Zb)) ********************************

            LDF     @_za_vect,R0                    ;
            LDF     @_zb_vect,R1                    ;
            LDF     @_s,R2                          ;
            ADDF    R0,R1                           ;
```

```
*       LDF     0.0,R2                          ;
*       SUBF    R1,R2                           ;
        ADDF    R1,R2                           ;
        LDF     0.0,R3                          ;
        SUBF3   R2,R3,R2                        ;
                                                ;
        LDF     2.5,R0                          ;
        CMPF    R2,R0                           ;
        BLT     LT                              ;
        LDF     -2.5,R0                         ;
        CMPF    R2,R0                           ;
        BGT     GT                              ;
        STF     R2,@_u_vect                     ;
        STF     R2,@_u                          ;
        B       OUT                             ;
LT:     STF     R0,@_u_vect                     ;
        STF     R0,@_u                          ;
        B       OUT                             ;
GT:     STF     R0,@_u_vect                     ;
        STF     R0,@_u                          ;
OUT:    NOP                                     ;

**********************************************************************

        LDF     @NORM_1,R3      ; LOAD NORM_1
        LDF     @_u,R0          ; LOAD U
                        ;
        MPYF    R3,R0           ; NORMALIZE BETWEEN 7FFF & 8000
        BGE     POS_2           ; BRANCH IF INPUT VALUE +VE
                                ;
        FIX     R0,R3           ; CONVERT FLOATING POINT TO INT.
        LDI     0FFFFh,R1       ;
        ADDI    R1,R3           ; ADD 65535 TO NEG. VALUE
                                ;
        BR      OUT_2           ; GOTO SENDOUT

POS_2:
        FIX     R0,R3           ; CONVERT FLOATING POINT TO INT.

OUT_2:

        LDI     @CH_0,AR2       ; AR2 - CH_0 I/O ADDR
                                ;
        LDI     16,R1           ; SHIFT VALUE
        ASH     R1,R3           ; FFFFXXXX / 0000XXXX => XXXX0000

        STI     R3,*AR2         ; OUTPUT THROUGH D/A CH_0

        RETS                    ;

********************************************************************
ISR0
        PUSH    ST              ; SAVE STATUS REGISTER
        PUSH    DP              ; SAVE DATA PAGE POINTER
        PUSH    IE              ; SAVE INTERRUPT ENABLE
                                ;
```

```
            LDI    @IMR,AR2          ; IMR FOR DAUGHTER MOD.
            LDI    *AR2,R0           ; CLEAR INTERRUPT MASK
                                     ;
            AND     0FFFEh,IF        ; RESET INTERRUPT FLAG

*********** Generate White Noise *********************************

            LDI    @_seed,R0         ;
                                     ;
            LDI    @END,R1           ;
            AND     R0,R1            ;
            LDI    @MID,R2           ;
            AND     R0,R2            ;
            LDI    -27,R3            ;
            LSH     R3,R2            ;
            LDI    -30,R3            ;
            LSH     R3,R1            ;
            XOR     R1,R2            ;
            LDI    1,R3              ;
            LSH     R3,R0            ;
            OR3     R0,R2,R0;
            STI        R0,@_seed     ;
                                     ;
            LDF     @NORM_1,R3       ; LOAD VALUE STORED AT NORM_1
            FLOAT   R2,R1            ;
            SUBF    0.5,R1           ;
            MPYF    3,R1             ;
            MPYF3   R3,R1,R2         ; CONVERT (+-)3V TO ORIGINAL FORM
                                     ;
            BNN     POS_1            ;
                                     ;
            FIX     R2,R4            ;
            LDI     0FFFFh,R1        ;
            ADDI    R1,R4            ;
                                     ;
            BR      OUT_1            ;
                                     ;
POS_1:
                                     ;
            FIX     R2,R4            ; CONVERT TO INTEGER
                                     ;
OUT_1:
                                     ;
            LDI     16,R1            ; 0000XXXX / FFFFXXXX => XXXX0000
            ASH      R1,R4           ;

*********** Output U & Input Y *********************************

            LDI     @CH_0,AR2        ; AR2 - CH_0 I/O ADDR
            LDI     @CH_1,AR3        ; AR3 - CH_1 I/O ADDR
                                     ;
            STI     R4,*AR3          ; OUTPUT THROUGH D/A CH_1
            LDI     *AR2,R0          ; READ A/D PORT

            LDI     -16,R1           ; SHIFT VALUE
            LDF     @NORM_2,R3       ; LOAD VALUE STORED AT NORM_2
```

```
                                        ;
            ASH     R1,R0               ; XXXX0000 => FFFFXXXX / 0000XXXX
                                        ;
            FLOAT   R0,R2               ; CONVERT TO FLOATING POINT VALUE
            MPYF3   R3,R2,R0            ; NORMALIZE VALUE TO (+-)3V

            STF     R0,@_y              ; STORE IN Y

                                        ;
            POP     IE                  ; RESTORE INTERRUPT ENABLE
            POP     DP                  ; RESTORE DATA PAGE POINTER
            POP     ST                  ; RESTORE STATUS REGISTER
                                        ;
            RETI                        ; RETURN FROM INTERRUPT

******************************************************************
ISR1    RETI  ; INT1
ISR2    RETI  ; INT2
ISR3    RETI  ; INT3
ISR4    RETI  ; XINT0
ISR5    RETI  ; RINT0
ISR6    RETI  ; TINT0
ISR7    RETI  ; TINT1
ISR8    RETI  ; DINT0

******************************************************************
```

B.6 ADAPT.CMD for LMS code

```
/******************************************************/
/* This linker command file is used to specify how the link        */
/* phase is to be performed. Input files, output files and         */
/* other options are specified in this command file.               */
/******************************************************/

/* Specify input files                                              */
adapt.obj
init_io.obj
filter.obj

/* Specify linker output files                                     */
-m adapt.map          /* Generate MAP File                          */
-o adapt.out          /* name executable output module              */
                      /* (default a.out)                            */

/* Specify linker options                                           */
-c                    /* Link Using C Conventions                   */
-stack 0x100          /* Stack                                      */
-heap  0x100          /* Heap                                       */
-l rts30.lib          /* Get Run-time Support                       */
```

```
/* Describe memory configuration for linker memory allocation      */
/* scheme. Specify 32K in Bank 0 and (optional) 512K in Bank 1   */
/* this allows the command file to be used for all C31              */
/* standard memory variants                                          */

/* give names to the various regions of memory available            */
/* to the C31                                                        */
/* use last reflection of Bank0; allow space for monitor             */

MEMORY
      {
      BANK0:  origin = 0478400h        length = 00007bffh
      BANK1:  origin = 0480000h        length = 0001ffffh
      DPRAM:  origin = 0500000h        length = 000007ffh
      IRAM0:  origin = 0809800h        length = 00000400h
      IRAM1:  origin = 0809c00h        length = 000003bfh
      VECS:   origin = 0809FC0h        length = 00000040h
      }

/* specify how each output section is to be                          */
/* allocated into memory regions                                     */
SECTIONS
      {
      .text:      > BANK0
      .bss:       > BANK1
      .data:      > BANK1
      .const:     > BANK1
      .cinit:     > BANK1
      .stack:     > BANK1
      .sysmem:    > BANK1
      .init:      > VECS
      .int0   0809fc1h: {} /* external interrupt 0                       */
      .int1   0809fc2h: {} /* external interrupt 1                       */
      .int2   0809fc3h: {} /* external interrupt 2                       */
      .int3   0809fc4h: {} /* external interrupt 3                       */
      .int4   0809fc5h: {} /* sport 0 tx                                 */
      .int5   0809fc6h: {} /* sport 0 rx                                 */
      .int6   0809fc9h: {} /* timer 0 int                                */
      .int7   0809fcah: {} /* timer 1 int                                */
      .int8   0809fcbh: {} /* DMA channel                                */
      }

/* end of adapt.cmd                                                  */
```

B.7 MAKEFILE for LMS code

```
@ECHO OFF
REM *************************************************************
REM *
REM *  Adaptive-Q Filter Makefile
REM *
REM *************************************************************
@ECHO ON

ac30 adapt.c
cg30 adapt
asm30 adapt.asm
asm30 init_io.asm
asm30 filter.asm
lnk30 adapt.cmd
```

B.8 ADAPT.C for Lattice code

```
/*************************************************/
/*                                               */
/*  adapt.c                                      */
/*                                               */
/*************************************************/

#include <math.h>
#include <stdlib.h>
#include "tfmod8.h"
#include "initvecs.h"

/*** defined constants ***/

#define  M          8      /* order of lattice filter */
#define  ORDER      28     /* order of system */

/*** variables ***/

float  mu_theta;
float  mu_nu;

float QLatticeParams[M+1];
float DLatticeParams[M+1];
float PFLatParams[M+1];
float gradient[M+1];
float nu[M+1];
float sines[M];
float cosines[M];
float theta[M];
float old_theta[M];

float  u, y, s, gm, gm_hold, post_input;

int    seed;
int    i, j;

/*** function prototypes ***/

extern void init_io(void);
extern void filter(void);
extern void postfilt(void);

/**************************************************/
/*                                                */
/*  Main Program: Adapt.c                         */
/*                                                */
/**************************************************/

void main(void)
{

  mu_theta = 0.001;
  mu_nu = 0.75;

  seed = 16671549;

  for (i=0; i < M+1; i++)
  {
    QLatticeParams[i]=0.0;
    DLatticeParams[i]=0.0;
    PFLatParams[i]=0.0;
    nu[i]=0.0;
  }

  for (i=0; i < M; i++)
  {
    sines[i]=0.0;
    cosines[i]=1.0;
    theta[i]=0.0;
    old_theta[i]=0.0;
  }

  u = 0.0;
  y = 0.0;
  s = 0.0;

  i = 1;

  init_io();

  while (i > 0)
  {
    filter();

    for (j=0; j < M; j++)
    {
      if (abs(theta[j]) > (3.1415926/2))
      {
        theta[j]=old_theta[j];
      }
      theta[j]=0.00;
      sines[j]=sin(theta[j]);
      cosines[j]=cos(theta[j]);
    }
    postfilt();
  }

}
```

B.9 INITVECS.H for Lattice code

```
float u_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float y_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float za_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float zb_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float yha_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float yhb_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float r_vect[29] =  { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float rp_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

float rb_vect[29] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
```

B.10 TFMOD8.H for Lattice code

```
const float num_uy[29] =  {  0.0000000e+000, -1.5904315e-004,  4.2409415e-004, -4.9284109e-002,
1.9893541e-001, -3.1758832e-001,  3.1991080e-001, -2.5978062e-001,  1.6495562e-001,
1.3093236e-002, -2.6836145e-001,  4.0050110e-001, -3.8240303e-001,  3.3581286e-001,
-2.5461830e-001,  9.4364512e-002,  9.4201270e-002, -1.9247663e-001,  2.0855357e-001,
-2.0134025e-001,  1.4533480e-001, -4.3740757e-002, -3.3162386e-002,  3.8873874e-002,
-2.9871692e-002,  3.2910294e-002, -3.7478365e-002,  2.3224869e-002, -9.5386817e-004};

const float den_uy[29] =    {  1.0000000e+000, -3.2865226e+000,  4.6603098e+000, -3.2893508e+000,
4.1916269e-001,  1.6767313e+000, -3.1235753e+000,  4.3896353e+000, -3.9919281e+000,
1.0311235e+000,  2.2520882e+000, -3.7354589e+000,  4.1809189e+000, -4.2092994e+000,
2.8734239e+000,  4.5056639e-002, -2.5650890e+000,  3.2889473e+000, -2.7153134e+000,
1.4414222e+000,  8.5999084e-002, -1.1600308e+000,  1.1084441e+000, -3.9557971e-001,
-1.2658787e-002, -1.2413373e-001,  2.8652604e-001, -1.8344510e-001,  5.6467152e-002};

const float num_uz[29] =  {  0.0000000e+000,  4.4880500e-002, -7.8615094e-002,  3.5876926e-002,
1.7895885e-003, -7.4897554e-003,  1.7182858e-002, -6.9484365e-002,  8.1450245e-002,
-1.6089545e-002, -6.8375095e-004, -1.4705265e-002, -1.0106208e-002,  4.9611529e-002,
-4.2922180e-002,  3.5838479e-003, -2.6290233e-004,  4.8756548e-003,  1.3150029e-002,
-2.3821924e-002,  1.0976638e-003,  1.3481585e-002, -1.0471705e-003, -6.0964111e-003,
3.2565513e-003,  6.9405164e-003, -1.8129320e-003, -4.1746617e-003,  1.4782862e-004};

const float den_uz[29] =    {  1.0000000e+000, -2.6732026e+000,  3.4256695e+000, -3.0453783e+000,
2.1656934e+000, -1.0075518e+000, -1.1942929e+000,  3.4116253e+000, -4.0199864e+000,
3.6430676e+000, -3.0497623e+000,  1.9259230e+000, -4.0241287e-002, -1.5552926e+000,
2.0543575e+000, -2.0193335e+000,  1.7315302e+000, -1.0097455e+000,  3.4233749e-002,
4.0383223e-001, -3.4999218e-001,  2.7868387e-001, -3.2353246e-001,  3.2916130e-001,
-1.1928593e-001,  3.3412439e-003,  3.3099793e-003, -2.1129510e-003,  5.5094230e-004};

const float num_yz[29] =  {  0.0000000e+000, -1.9824451e-001,  6.5783635e-001, -7.1735758e-001,
2.5201745e-001,  4.6119829e-001, -1.1576058e+000,  1.6506510e+000, -2.0641613e+000,
1.8259873e+000, -9.6363275e-001, -2.5721269e-002,  7.9107211e-001, -1.2764404e+000,
1.5898900e+000, -1.3052523e+000,  6.1665461e-001,  7.2693068e-002, -4.5457441e-001,
5.8160015e-001, -4.9605233e-001,  2.2580249e-001, -4.6555229e-002, -8.4845841e-003,
-7.5207262e-002,  8.1771196e-002, -1.2539931e-002, -1.4700859e-002,  1.1441345e-002};

const float den_yz[29] =    {  1.0000000e+000, -2.6732026e+000,  3.4256695e+000, -3.0453783e+000,
2.1656934e+000, -1.0075518e+000, -1.1942929e+000,  3.4116253e+000, -4.0199864e+000,
3.6430676e+000, -3.0497623e+000,  1.9259230e+000, -4.0241287e-002, -1.5552926e+000,
2.0543575e+000, -2.0193335e+000,  1.7315302e+000, -1.0097455e+000,  3.4233749e-002,
4.0383223e-001, -3.4999218e-001,  2.7868387e-001, -3.2353246e-001,  3.2916130e-001,
-1.1928593e-001,  3.3412439e-003,  3.3099793e-003, -2.1129510e-003,  5.5094230e-004};

const float num_uyh[29] ={  0.0000000e+000, -1.5904315e-004,  4.2409415e-004, -4.9284109e-002,
1.9893541e-001, -3.1758832e-001,  3.1991080e-001, -2.5978062e-001,  1.6495562e-001,
1.3093236e-002, -2.6836145e-001,  4.0050110e-001, -3.8240303e-001,  3.3581286e-001,
-2.5461830e-001,  9.4364512e-002,  9.4201270e-002, -1.9247663e-001,  2.0855357e-001,
-2.0134025e-001,  1.4533480e-001, -4.3740757e-002, -3.3162386e-002,  3.8873874e-002,
-2.9871692e-002,  3.2910294e-002, -3.7478365e-002,  2.3224869e-002, -9.5386817e-004};

const float den_uyh[29] =  {  1.0000000e+000, -2.6732026e+000,  3.4256695e+000, -3.0453783e+000,
2.1656934e+000, -1.0075518e+000, -1.1942929e+000,  3.4116253e+000, -4.0199864e+000,
3.6430676e+000, -3.0497623e+000,  1.9259230e+000, -4.0241287e-002, -1.5552926e+000,
2.0543575e+000, -2.0193335e+000,  1.7315302e+000, -1.0097455e+000,  3.4233749e-002,
4.0383223e-001, -3.4999218e-001,  2.7868387e-001, -3.2353246e-001,  3.2916130e-001,
-1.1928593e-001,  3.3412439e-003,  3.3099793e-003, -2.1129510e-003,  5.5094230e-004};

const float num_yyh[29] ={  0.0000000e+000,  6.1331992e-001, -1.2346403e+000,  2.4397244e-001,
1.7465307e+000, -2.6842832e+000,  1.9292824e+000, -9.7800999e-001, -2.8058302e-002,
2.6119441e+000, -5.3018505e+000,  5.6613819e+000, -4.2211602e+000,  2.6540068e+000,
-8.1906648e-001, -2.0643901e+000,  4.2966192e+000, -4.2986929e+000,  2.7495472e+000,
-1.0375900e+000, -4.3599127e-001,  1.4387147e+000, -1.4319766e+000,  7.2474101e-001,
-1.0662714e-001,  1.2747497e-001, -2.8321606e-001,  1.8133215e-001, -5.5916210e-002};

const float den_yyh[29] =  {  1.0000000e+000, -2.6732026e+000,  3.4256695e+000, -3.0453783e+000,
2.1656934e+000, -1.0075518e+000, -1.1942929e+000,  3.4116253e+000, -4.0199864e+000,
3.6430676e+000, -3.0497623e+000,  1.9259230e+000, -4.0241287e-002, -1.5552926e+000,
2.0543575e+000, -2.0193335e+000,  1.7315302e+000, -1.0097455e+000,  3.4233749e-002,
4.0383223e-001, -3.4999218e-001,  2.7868387e-001, -3.2353246e-001,  3.2916130e-001,
-1.1928593e-001,  3.3412439e-003,  3.3099793e-003, -2.1129510e-003,  5.5094230e-004};

const float num_hp[7] =  { 7.60790003e-001, -4.56474002e+000, 1.14118500e+001, -1.52158000e+001,
1.14118500e+001, -4.5647400e+000, 7.60790003e-001 };

const float den_hp[7] =  { 1.00000000e+000, -5.45387055e+000, 1.24164881e+001, -1.51024974e+001,
1.03499935e+001, -3.78890907e+000, 5.78801429e-001 };
```

B.11 INITIO.ASM for Lattice code

```
******************************************************************
*
* INITALIZE I/O ROUTINE
*
******************************************************************

        .global      _init_io

******************************************************************
*
* Set C31 for LSI Control
*
******************************************************************
               .data

BUSADDR .word   00808064h      ; ADDRESS OF BUS CONTROL REGISTER
BUSDATA .word   00000900h      ; VALUE FOR LSI CARD GIVES ZERO
                               ; WAIT STATES, 32k BANK SIZE,
                               ; EXTERNAL READY CONTROL

******************************************************************
*
* INITIALIZE CONSTANTS
*
******************************************************************

CH_0        .word   550002h    ; CH_0 ADDRESS
TM1         .word   550005h    ; TIMER 1 REGISTER
CH_1        .word   550006h    ; CH_1 ADDRESS
UCR         .word   550008h    ; USER CONTROL REG.
ACR         .word   55000Ah    ; AMELIA CONTROL REG.
IMR         .word   55000Bh    ; INTERRUPT MASK REG.
CFR         .word   55000Fh    ; CONFIGURATION REGISTER

VALUES        .word  0A4000000h       ; A4000000h
               .word   0E8010000h     ; E8010000h 2 kHz
*              .word   0FA010000h     ; FA010000h 8 kHz
               .word   000B20000h     ; 00B20000h
               .word   08DFF0000h     ; 8DFF0000h
               .word   000010000h     ; 00010000h

******************************************************************
*
* INITALIZE I/O Routine
*
******************************************************************
               .text

_init_io:

        LDP    BUSADDR,DP             ; DP POINTER SET PRIOR TO
                                      ; USING DIR. ADDR. MODE
        LDI    @BUSADDR,AR0           ;
        LDI     @BUSDATA,R0             ;
        STI     R0,*AR0                 ; SET BUS. REG. FOR LSI CARD

        LDP     @VALUES,DP              ; SET DATA PAGE POINTER
        LDI     @VALUES,R0              ; SET USER CONTROL REGISTER
        LDI     @UCR,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = A4000000h

        LDI     @VALUES+1,R0            ; SET TIMER 1 REGISTER
        LDI     @TM1,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = FFC40000h

        LDI     @VALUES+2,R0            ; SET AMELIA CONTROL REGISTER
        LDI     @ACR,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = 00B20000h

        LDI     @VALUES+3,R0            ; SET CONFIGURATION REGISTER
        LDI     @CFR,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = 8DFF0000h

        LDI     @VALUES+4,R0            ; SET INTERRUPT MASK REGISTER
        LDI     @IMR,AR0                ; FOR BURR_BROWN DAUGHTER MOD.
        STI     R0,*AR0                 ; VAL = 00010000h

        RETS

******************************************************************
```

B.12 FILTER.ASM for Lattice code

```
******************************************************************
*
* Adaptive-Q Filter Routine
*
******************************************************************

        .global         _filter
        .global         _mu_theta, _mu_nu, _y, _u, _s
        .global         ISR0,ISR1,ISR2,ISR3

        .global         _num_uy, _den_uy
        .global         _num_uz, _den_uz
        .global         _num_yz, _den_yz
        .global          _num_uyh, _den_uyh
        .global         _num_yyh, _den_yyh
        .global         _num_hp, _den_hp

        .global         _za_vect, _zb_vect
        .global         _yha_vect, _yhb_vect
        .global         _u_vect, _y_vect
        .global         _r_vect, _old_theta
        .global         _rb_vect, _rp_vect

        .global         _PFLatParams

        .global         _QLatticeParams
        .global         _DLatticeParams
        .global         _nu, _sines, _cosines, _theta
        .global         _gm_hold
        .global         _gradient, _seed

******************************************************************
*
* Interrupt Vector Specifications
*
******************************************************************

        .sect   ".int0"
        BR      ISR0
        .sect   ".int1"
        BR      ISR1
        .sect   ".int2"
        BR      ISR2
        .sect   ".int3"
        BR      ISR3

******************************************************************
*
* Vector Addresses & Constants
*
******************************************************************
        .data

ord             .set    28              ;
M               .set    8               ;

NORM_1          .float  1.09225e+4      ; REPRESENTS 1V
NORM_2          .float  9.155552843e-5  ;

END             .word   40000000h       ;
MID             .word   08000000h       ;

IMR             .word   55000Bh         ; INTERRUPT MASK REG.
CH_0            .word   550002h         ; CH_0 ADDRESS
CH_1            .word   550006h         ; CH_1 ADDRESS

numuy_addr      .word   _num_uy         ;
denuy_addr      .word   _den_uy         ;
numuz_addr      .word   _num_uz         ;
denuz_addr      .word   _den_uz         ;
numyz_addr      .word   _num_yz         ;
denyz_addr      .word   _den_yz         ;
numuyh_addr     .word   _num_uyh        ;
denuyh_addr     .word   _den_uyh        ;
numyyh_addr     .word   _num_yyh        ;
denyyh_addr     .word   _den_yyh        ;
numhp_addr      .word   _num_hp         ;
denhp_addr      .word   _den_hp         ;

yvect_addr      .word   _y_vect         ;
uvect_addr      .word   _u_vect         ;
zavect_addr     .word   _za_vect        ;
```

```
zbvect_addr         .word     _zb_vect              ;
yhavect_addr        .word     _yha_vect             ;
yhbvect_addr        .word     _yhb_vect             ;
rvect_addr          .word     _r_vect               ;
rpvect_addr         .word     _rp_vect              ;
rbvect_addr         .word     _rb_vect              ;

q_addr              .word     _QLatticeParams
d_addr              .word     _DLatticeParams

post_addr           .word     _PFLatParams

nu_addr             .word     _nu                   ;
grad_addr           .word     _gradient             ;
sin_addr .word      _sines                          ;
cos_addr            .word     _cosines              ;

theta_addr          .word     _theta                ;
oldth_addr          .word     _old_theta            ;

******************************************************************
*
* filter Routine
*
******************************************************************
                    .text

_filter:

*********** Output U & Input Y ************************************

          OR      01h,IE                           ; ENABLE INTERRUPT 0
                                                   ;
          OR      2000h,ST                         ; ENABLE ALL INT IN IE REG
          IDLE                                     ; WAIT FOR INTERRUPT
          AND     1FFFh,ST                         ; RESET GIE TO ZERO
                                                   ;
          AND     0FFEh,IE                         ; DISABLE INTERRUPT ZERO

******************************************************************

          LDI       @zavect_addr,AR0        ;
          LDI       @zbvect_addr,AR1        ;
          LDI       @yhavect_addr,AR2       ;
          LDI       @yhbvect_addr,AR3       ;
          LDI       @rvect_addr,AR4         ;
          LDI       @rpvect_addr,AR5        ;
          LDI       @rbvect_addr,AR6        ;
          LDI       @yvect_addr,AR7         ;
                                            ;
          LDI       ord-1,IR0               ;
      LDI           ord,IR1                 ;
                                            ;
          LDI       ord-1,RC                ;
          RPTB      shift1                  ;
          LDF       *+AR0(IR0),R0           ;
```

```
    ||      LDF     *+AR1(IR0),R1           ;
            LDF     *+AR2(IR0),R2           ;
    ||      LDF     *+AR3(IR0),R3           ;
            LDF     *+AR4(IR0),R4           ;
    ||      LDF     *+AR5(IR0),R5           ;
            LDF     *+AR6(IR0),R6           ;
    ||      LDF     *+AR7(IR0),R7           ;
            STF     R0,*+AR0(IR1)           ;
    || STF          R1,*+AR1(IR1)           ;
            STF     R2,*+AR2(IR1)           ;
    || STF          R3,*+AR3(IR1)           ;
            STF     R4,*+AR4(IR1)           ;
    || STF          R5,*+AR5(IR1)           ;
            STF     R6,*+AR6(IR1)           ;
    ||      STF     R7,*+AR7(IR1)           ;
            SUBI    1,IR0                   ;
shift1:     SUBI    1,IR1                   ;
                                            ;
            LDI     @yvect_addr,AR0         ;
                                            ;
            LDF     @_y,R0                  ;
            STF     R0,*AR0                 ;

**************** TF from U to Za *********************************

            LDI     @numuz_addr,AR0                 ;
            LDI     @uvect_addr,AR1                 ;
            LDI     @denuz_addr,AR2                 ;
            LDI     @zavect_addr,AR3                ;
                                                    ;
            LDF     0.0,R0                          ;
            LDF     0.0,R2                          ;
                                                    ;
            LDI     ord,RC                          ;
            RPTS                                    ;
            MPYF3   *AR0++(1),*AR1++(1),R0          ;
         || ADDF3   R0,R2,R2                        ;
            ADDF3   R0,R2,R2                        ;
                                                    ;
            LDF     0.0,R0                          ;
                                                    ;
            LDI     ord-1,RC                        ;
            RPTS                                    ;
            MPYF3   *++AR2(1),*++AR3(1),R0          ;
         || SUBF3   R0,R2,R2                        ;
            SUBF3   R0,R2,R2                        ;
                                                    ;
            STF     R2,@_za_vect                    ;

**************** TF from Y to Zb ********************************

            LDI     @numyz_addr,AR0                 ;
            LDI     @yvect_addr,AR1                 ;
            LDI     @denyz_addr,AR2                 ;
            LDI     @zbvect_addr,AR3                ;
                                                    ;
```

```
          LDF     0.0,R0                        ;
          LDF     0.0,R2                        ;
                                                ;
          LDI     ord,RC                        ;
          RPTS                                  ;
          MPYF3   *AR0++(1),*AR1++(1),R0        ;
       || ADDF3   R0,R2,R2                      ;
          ADDF3   R0,R2,R2                      ;
                                                ;
          LDF     0.0,R0                        ;
                                                ;
          LDI     ord-1,RC                      ;
          RPTS                                  ;
          MPYF3   *++AR2(1),*++AR3(1),R0        ;
       || SUBF3   R0,R2,R2                      ;
          SUBF3   R0,R2,R2                      ;
                                                ;
          STF     R2,@_zb_vect                  ;

**************** TF from U to Y_HAT ******************************

          LDI     @numuyh_addr,AR0              ;
          LDI     @uvect_addr,AR1               ;
          LDI     @denuyh_addr,AR2              ;
          LDI     @yhavect_addr,AR3             ;
                                                ;
          LDF     0.0,R0                        ;
          LDF     0.0,R2                        ;
                                                ;
          LDI     ord,RC                        ;
          RPTS                                  ;
          MPYF3   *AR0++(1),*AR1++(1),R0        ;
       || ADDF3   R0,R2,R2                      ;
          ADDF3   R0,R2,R2                      ;
                                                ;
          LDF     0.0,R0                        ;
                                                ;
          LDI     ord-1,RC                      ;
          RPTS                                  ;
          MPYF3   *++AR2(1),*++AR3(1),R0        ;
       || SUBF3   R0,R2,R2                      ;
          SUBF3   R0,R2,R2                      ;
                                                ;
          STF     R2,@_yha_vect                 ;

**************** TF from Y to Y_HAT ******************************

          LDI     @numyyh_addr,AR0              ;
          LDI     @yvect_addr,AR1               ;
          LDI     @denyyh_addr,AR2              ;
          LDI     @yhbvect_addr,AR3             ;
                                                ;
          LDF     0.0,R0                        ;
          LDF     0.0,R2                        ;
                                                ;
          LDI     ord,RC                        ;
          RPTS                                   ;
          MPYF3  *AR0++(1),*AR1++(1),R0          ;
       || ADDF3  R0,R2,R2                        ;
          ADDF3  R0,R2,R2                        ;
                                                ;
          LDF    0.0,R0                          ;
                                                ;
          LDI    ord-1,RC                        ;
          RPTS                                   ;
          MPYF3  *++AR2(1),*++AR3(1),R0          ;
       || SUBF3  R0,R2,R2                        ;
          SUBF3  R0,R2,R2                        ;
                                                ;
          STF    R2,@_yhb_vect                   ;

**************** Y - Y_HAT ***************************************

          LDF    @_yha_vect,R0                   ;
          LDF    @_yhb_vect,R1                   ;
          LDF    @_y_vect,R3                     ;
          ADDF   R0,R1                           ;
          SUBF   R1,R3                           ;
          STF    R3,@_r_vect                     ;

**************** TF from R to R_BAR *******************************

          LDI    @numhp_addr,AR0                 ;
          LDI    @rvect_addr,AR1                 ;
          LDI    @denhp_addr,AR2                 ;
          LDI    @rbvect_addr,AR3                ;
                                                ;
          LDF    0.0,R0                          ;
          LDF    0.0,R2                          ;

                                                ;
          LDI    6,RC                            ;
          RPTS                                   ;
          MPYF3  *AR0++(1),*AR1++(1),R0          ;
       || ADDF3  R0,R2,R2                        ;
          ADDF3  R0,R2,R2                        ;
                                                ;
          LDF    0.0,R0                          ;
                                                ;
          LDI    5,RC                            ;
          RPTS                                   ;
          MPYF3  *++AR2(1),*++AR3(1),R0          ;
       || SUBF3  R0,R2,R2                        ;
          SUBF3  R0,R2,R3                        ;
                                                ;
          STF    R3,@_rb_vect                    ;

**************** TF from R_BAR to R_POST **************************

          LDI    @numuy_addr,AR0                 ;
          LDI    @rbvect_addr,AR1                ;
          LDI    @denuy_addr,AR2                 ;
```

```
          LDI     @rpvect_addr,AR3             ;
                                               ;
          LDF     0.0,R0                       ;
          LDF     0.0,R2                       ;
                                               ;
          LDI     ord,RC                       ;
          RPTS                                 ;
          MPYF3   *AR0++(1),*AR1++(1),R0       ;
       || ADDF3   R0,R2,R2                     ;
          ADDF3   R0,R2,R2                     ;
                                               ;
          LDF     0.0,R0                       ;
                                               ;
          LDI     ord-1,RC                     ;
          RPTS                                 ;
          MPYF3   *++AR2(1),*++AR3(1),R0       ;
       || SUBF3   R0,R2,R2                     ;
          SUBF3   R0,R2,R3                     ;

          STF     R3,@_rp_vect                 ;

***** CALCULATE GRADIENT ******************************************

          LDI     @grad_addr,AR0               ;
          LDI     @post_addr,AR1               ;
          LDI     @cos_addr,AR2                ;
          LDF     1.0,R0                       ;
                                               ;
          LDI     M-1,RC                       ;
          RPTB    GTH                          ;
          MPYF3   R0,*++AR1(1),R2              ;
          STF     R2,*AR0++(1)                 ;
GTH:      MPYF3   R0,*AR2++(1),R0              ;
                                               ;
***** NU = NU + MU*ERR*D *****************************************

          LDI     @d_addr,AR0                  ;
          LDI     @nu_addr,AR1                 ;

          LDF     @_mu_nu,R0                   ;
          LDF     @_y_vect,R2                  ;
          MPYF    R2,R0                        ;
                                               ;
          LDI     M,RC                         ;
          RPTB    NULOOP                       ;
          MPYF3   R0,*AR0++(1),R2              ;
          ADDF    *AR1,R2                      ;
NULOOP:           STF     R2,*AR1++(1)         ;

****** THETA = THETA + MU*ERR*GRADIENT ***************************

          LDI     @oldth_addr,AR0              ;
          LDI     @theta_addr,AR1              ;

          LDI     M-1,RC                       ;
          RPTB    OTL                          ;
          LDF     *AR1++(1),R0                    ;
OTL:      STF     R0,*AR0++(1)                    ;

          LDI     @grad_addr,AR2                  ;
          LDI     @theta_addr,AR3                 ;
                                                  ;
          LDF     @_mu_theta,R6                   ;
          LDF     @_y_vect,R7                     ;
          MPYF    R6,R7                           ;
                                                  ;
          LDI     M-1,RC                          ;
          RPTB    SLOOP                           ;
          MPYF3   *AR2++(1),R7,R0                 ;
          ADDF    *AR3,R0                         ;

SLOOP:    STF     R0,*AR3++(1)                    ;

          RETS                                    ;

*****************************************************************
ISR0
          PUSH    ST             ; SAVE STATUS REGISTER
          PUSH    DP             ; SAVE DATA PAGE POINTER
          PUSH    IE             ; SAVE INTERRUPT ENABLE
                                 ;
          LDI    @IMR,AR2        ; IMR FOR DAUGHTER MOD.
          LDI    *AR2,R0         ; CLEAR INTERRUPT MASK
                                 ;
          AND     0FFFEh,IF      ; RESET INTERRUPT FLAG

*********** Generate White Noise ********************************

          LDI    @_seed,R0        ;
                                 ;
          LDI    @END,R1         ;
          AND     R0,R1          ;
          LDI    @MID,R2         ;
          AND     R0,R2          ;
          LDI    -27,R3          ;
          LSH     R3,R2          ;
          LDI    -30,R3          ;
          LSH     R3,R1          ;
          XOR     R1,R2          ;
          LDI    1,R3            ;
          LSH     R3,R0          ;
          OR3     R0,R2,R0       ;
          STI      R0,@_seed     ;
                                 ;
          LDF     @NORM_1,R3     ; LOAD VALUE STORED AT NORM_1
          FLOAT   R2,R1          ;
          SUBF    0.5,R1         ;
          MPYF    3,R1           ;
          MPYF3   R3,R1,R2       ; CONVERT (+-)3V TO ORIGINAL FORM
                                 ;
          BNN     POS_1          ;
                                 ;
```

```
            FIX     R2,R4               ;
            LDI     0FFFFh,R1           ;
            ADDI    R1,R4               ;
                                        ;
            BR      OUT_1               ;
                                        ;
POS_1:
                                        ;
            FIX     R2,R4               ; CONVERT TO INTEGER
                                        ;
OUT_1:
                                        ;
            LDI     16,R1               ; 0000XXXX / FFFFXXXX => XXXX0000
            ASH     R1,R4               ;

*********** Output U & Input Y *********************************

            LDI     @CH_0,AR2           ; AR2 - CH_0 I/O ADDR
            LDI     @CH_1,AR3           ; AR3 - CH_1 I/O ADDR
                                        ;
            STI     R4,*AR3             ; OUTPUT THROUGH D/A CH_1
            LDI     *AR2,R0             ; READ A/D PORT CH_0
                                        ;
            LDI     -16,R1              ; SHIFT VALUE
            LDF     @NORM_2,R3          ; LOAD VALUE STORED AT NORM_2
                                        ;
            ASH     R1,R0               ; XXXX0000 => FFFFXXXX / 0000XXXX
                                        ;
            FLOAT   R0,R2               ; CONVERT TO FLOATING POINT VALUE
            MPYF3   R3,R2,R0            ; NORMALIZE VALUE TO (+-)3V
            STF     R0,@_y              ; STORE IN Y
                                        ;
            POP     IE                  ; RESTORE INTERRUPT ENABLE
            POP     DP                  ; RESTORE DATA PAGE POINTER
            POP     ST                  ; RESTORE STATUS REGISTER
                                        ;
            RETI                        ; RETURN FROM INTERRUPT

*****************************************************************
ISR1        RETI  ; INT1
ISR2        RETI  ; INT2
ISR3        RETI  ; INT3
ISR4        RETI  ; XINT0
ISR5        RETI  ; RINT0
ISR6        RETI  ; TINT0
ISR7        RETI  ; TINT1
ISR8        RETI  ; DINT0

*****************************************************************
```

B.13 POSTFILT.ASM for Lattice code

```
******************************************************************
*
* Post Filter Routine
*
******************************************************************

        .global         _postfilt

        .global         _PFLatParams
        .global         _sines, _cosines
        .global         _gm_hold, _post_input
        .global         _y, _u, _s

        .global         _QLatticeParams
        .global         _DLatticeParams
        .global         _nu, _sines, _cosines

        .global         _za_vect, _zb_vect
        .global         _u_vect, _y_vect
        .global         _r_vect, _old_theta
        .global         _rb_vect, _rp_vect

******************************************************************
*
* Vector Addresses & Constants
*
******************************************************************
        .data

ord             .set    28              ;
M               .set    8               ;

NORM_1          .float  1.09225e+4      ; REPRESENTS 1V
NORM_2          .float  9.155552843e-5  ;

END             .word   40000000h       ;
MID             .word   08000000h       ;

CH_0            .word   550002h         ; CH_0 ADDRESS
CH_1            .word   550006h         ; CH_1 ADDRESS

q_addr          .word   _QLatticeParams ;
d_addr          .word   _DLatticeParams ;
post_addr       .word   _PFLatParams    ;

nu_addr         .word   _nu             ;
sin_addr        .word   _sines          ;
cos_addr        .word   _cosines        ;

yvect_addr      .word   _y_vect         ;
uvect_addr      .word   _u_vect         ;
zavect_addr     .word   _za_vect        ;
zbvect_addr     .word   _zb_vect        ;
```

```
rpvect_addr        .word     _rp_vect              ;
rbvect_addr        .word     _rb_vect              ;

*****************************************************************
*
* filter Routine
*
*****************************************************************
                   .text

_postfilt:

************* D Lattice Filter ***********************************

            LDI       @cos_addr,AR1                ;
            LDI       @d_addr,AR2                  ;
            LDI       @sin_addr,AR3                ;
                                                   ;
            LDF       0.0,R0                       ;
            LDF       0.0,R1                       ;
            LDF       0.0,R2                       ;
                                                   ;
            LDF       @_rp_vect,R3                 ;
            STF       R3,@_gm_hold                 ;
                                                   ;
            LDI       M-1,RC                       ;
                                                   ;
            RPTB      LATLP1                       ;
                                                   ;
            LDF       @_gm_hold,R3                 ;
                                                   ;
            MPYF3 R3,*AR1,R0                       ;
            MPYF3 *+AR2(1),*AR3,R1                 ;
            SUBF3 R1,R0,R2                         ;
                                                   ;
            MPYF3 R3,*AR3++(1),R0                  ;
            MPYF3 *+AR2(1),*AR1++(1),R1            ;
            ADDF3 R0,R1,R3                         ;
                                                   ;
            STF       R3,*AR2++(1)                 ;
            STF       R2,@_gm_hold                 ;
LATLP1: nop                                        ;
                                                   ;
            STF       R2,*AR2                      ;
                                                   ;
            LDI       @d_addr,AR0                  ;
            LDI       @nu_addr,AR1                 ;
                                                   ;
            LDF       0.0,R0                       ;
            LDF       0.0,R2                       ;
                                                   ;
            LDI       M,RC                         ;
            RPTS                                   ;
            MPYF3 *AR0++(1),*AR1++(1),R0           ;
         || ADDF3 R0,R2,R2                         ;
            ADDF3 R0,R2,R2                         ;
```

```
                                                  ;
        LDF     0.0,R0                            ;
        SUBF    R2,R0                             ;
        STF     R0,@_post_input                   ;

************* Q Lattice Filter ***********************************

        LDI     @cos_addr,AR1                     ;
        LDI     @q_addr,AR2                       ;
        LDI     @sin_addr,AR3                     ;
                                                  ;
        LDF     0.0,R0                            ;
        LDF     0.0,R1                            ;
        LDF     0.0,R2                            ;

        LDF     @_rb_vect,R3                      ;
        STF     R3,@_gm_hold                      ;
                                                  ;
        LDI     M-1,RC                            ;
                                                  ;
        RPTB    LATLP2                            ;
                                                  ;
        LDF     @_gm_hold,R3                      ;
                                                  ;
        MPYF3   R3,*AR1,R0                        ;
        MPYF3   *+AR2(1),*AR3,R1                  ;
        SUBF3   R1,R0,R2                          ;
                                                  ;
        MPYF3   R3,*AR3++(1),R0                   ;
        MPYF3   *+AR2(1),*AR1++(1),R1             ;
        ADDF3   R0,R1,R3                          ;
                                                  ;
        STF     R3,*AR2++(1)                      ;
        STF     R2,@_gm_hold                      ;
LATLP2: nop                                       ;
                                                  ;
        STF     R2,*AR2                           ;
                                                  ;
        LDI     @q_addr,AR0                       ;
        LDI     @nu_addr,AR1                      ;
                                                  ;
        LDF     0.0,R0                            ;
        LDF     0.0,R2                            ;
                                                  ;
        LDI     M,RC                              ;
        RPTS                                      ;
        MPYF3   *AR0++(1),*AR1++(1),R0            ;
     || ADDF3   R0,R2,R2                          ;
        ADDF3   R0,R2,R2                          ;
                                                  ;
        STF     R2,@_s                            ;

*****************************************************************

        LDI     @uvect_addr,AR0                   ;
                                                  ;
        LDI     ord-1,IR0                        ;
        LDI     ord,IR1                          ;
                                                 ;
        LDI     ord-1,RC                         ;
        RPTB    shift4                           ;
        LDF     *+AR0(IR0),R0                    ;
        STF     R0,*+AR0(IR1)                    ;
        SUBI    1,IR0                            ;
shift4: SUBI    1,IR1                            ;

********* U_VECT = -(S + (Za + Zb)) **********************************

        LDF     @_za_vect,R0               ;
        LDF     @_zb_vect,R1               ;
        LDF     @_s,R2                     ;
        ADDF    R0,R1                      ;
*       LDF     0.0,R2                     ; %%% Uncomment for LQG Only %%%
        LDF     0.0,R3                     ;
        ADDF    R1,R2                      ;
        SUBF3   R2,R3,R2                   ;
                                           ;
        LDF     2.5,R0                     ;
        CMPF    R2,R0                      ;
        BLT     LT                         ;
        LDF     -2.5,R0                    ;
        CMPF    R2,R0                      ;
        BGT     GT                         ;
        STF     R2,@_u_vect                ;
        STF     R2,@_u                     ;
        B       OUT                        ;
LT:     STF     R0,@_u_vect                ;
        STF     R0,@_u                     ;
        B       OUT                        ;
GT:     STF     R0,@_u_vect                ;
        STF     R0,@_u                     ;
OUT:    NOP                                ;

*********** WRITE U TO OUTPUT REGISTER **********************

        LDF     @NORM_1,R3                 ; LOAD NORM_1
        LDF     @_u,R0                     ; LOAD U
                                           ;
        MPYF    R3,R0                      ; NORMALIZE BETWEEN 7FFF & 8000
        BGE     POS_2                      ; BRANCH IF INPUT VALUE +VE
                                           ;
        FIX     R0,R3                      ; CONVERT FLOATING POINT TO INT.
        LDI     0FFFFh,R1                  ;
        ADDI    R1,R3                      ; ADD 65535 TO NEG. VALUE
                                           ;
        BR      OUT_2                      ; GOTO SENDOUT

POS_2:
        FIX     R0,R3                      ; CONVERT FLOATING POINT TO INT.

OUT_2:
```

```
            LDI     @CH_0,AR2                   ; AR2 - CH_0 I/O ADDR
                                                ;
            LDI     16,R1                       ; SHIFT VALUE
            ASH      R1,R3                      ; FFFFXXXX / 0000XXXX => XXXX0000
                                                ;
            STI     R3,*AR2                     ; OUTPUT TO D/A CH_0 REG.


************* Post State Lattice Filter *************************

            LDI       @cos_addr,AR1             ;
            LDI       @post_addr,AR2            ;
            LDI       @sin_addr,AR3             ;
                                                ;
            LDF       0.0,R0                    ;
            LDF       0.0,R1                    ;
            LDF       0.0,R2                    ;
                                                ;
            LDF       @_post_input,R3           ;
            STF       R3,@_gm_hold              ;
                                                ;
            LDI       M-1,RC                    ;
                                                ;
            RPTB      LATLP3                    ;
                                                ;
            LDF       @_gm_hold,R3              ;
                                                ;
            MPYF3 R3,*AR1,R0                    ;
            MPYF3 *+AR2(1),*AR3,R1              ;
            SUBF3  R1,R0,R2                     ;
                                                ;
            MPYF3 R3,*AR3++(1),R0               ;
            MPYF3 *+AR2(1),*AR1++(1),R1 ;
            ADDF3 R0,R1,R3                      ;
                                                ;
            STF       R3,*AR2++(1)              ;
            STF       R2,@_gm_hold              ;
LATLP3: nop                                     ;
                                                ;
            STF       R2,*AR2                   ;
                                                ;
            RETS
```

B.14 ADAPT.CMD for Lattice code

```
/******************************************************/
/* This linker command file is used to specify how the link          */
/* phase is to be performed. Input files, output files and            */
/* other options are specified in this command file.                  */
/******************************************************/

/* Specify input files                                                 */
adapt.obj
init_io.obj
filter.obj
postfilt.obj

/* Specify linker output files                                         */
-m adapt.map          /* Generate MAP File                             */
-o adapt.out          /* name executable output module                 */
                      /* (default a.out)                               */

/* Specify linker options                                              */
-c                    /* Link Using C Conventions                      */
-stack 0x100          /* Stack                                         */
-heap  0x100          /* Heap                                          */
-l rts30.lib          /* Get Run-time Support                          */


/* Describe memory configuration for linker memory allocation     */
/* scheme. Specify 32K in Bank 0 and (optional) 512K in Bank 1,  */
/* this allows the command file to be used for all C31                 */
/* standard memory variants                                            */

/* give names to the various regions of memory available              */
/* to the C31                                                          */
/* use last reflection of Bank0; allow space for monitor              */

MEMORY
    {
    BANK0:  origin = 0478400h       length = 00007bffh
    BANK1:  origin = 0480000h       length = 0001ffffh
    DPRAM:  origin = 0500000h       length = 000007ffh
    IRAM0:  origin = 0809800h       length = 00000400h
    IRAM1:  origin = 0809c00h       length = 000003bfh
    VECS:   origin = 0809FC0h       length = 00000040h
    }

/* specify how each output section is to be                            */
/* allocated into memory regions                                    */
SECTIONS
    {
    .text:     > BANK0
    .bss:      > BANK1
    .data:     > BANK1
    .const:    > BANK1
    .cinit:    > BANK1
    .stack:    > BANK1
    .sysmem:   > BANK1
    .init:     > VECS
    .int0   0809fc1h: {} /* external interrupt 0                   */
    .int1   0809fc2h: {} /* external interrupt 1                   */
    .int2   0809fc3h: {} /* external interrupt 2                   */
    .int3   0809fc4h: {} /* external interrupt 3                   */
    .int4   0809fc5h: {} /* sport 0 tx                             */
    .int5   0809fc6h: {} /* sport 0 rx                             */
    .int6   0809fc9h: {} /* timer 0 int                            */
    .int7   0809fcah: {} /* timer 1 int                            */
    .int8   0809fcbh: {} /* DMA channel                            */
    }

/* end of adapt.cmd                                                 */
```

B.15 MAKEFILE for Lattice code

```
@ECHO OFF
REM ************************************************************
REM *
REM *  Adaptive-Q Filter Makefile
REM *
REM ************************************************************
@ECHO ON

ac30 adapt.c
cg30 adapt
asm30 adapt.asm
asm30 init_io.asm
asm30 filter.asm
asm30 postfilt.asm
lnk30 adapt.cmd
```

Appendix C

C.1 OPTCNTRL.M

```
%
% Simulation of Clamped Beam
%
%

loadmodl;

%%% Narrow Band (189 Hz center 20 Hz BW) %%%
%[num,den]=butter(5,[0.179 0.199]);

%%% Wide Band (300 Hz center 250 Hz BW) %%%
[num,den]=butter(5,[0.075 0.425]);

%%%%% Harmonic Disturbance %%%%%

Time=0:0.0005:2;

randn('seed',s);
prbs=sign(randn(size(Time,2),1));
Input=dlsim(num,den,prbs);

ord=size(Ac,0);

%%%%% Closed Loop System %%%%%

[Af,Bf,Cf,Df]=tf2ss(num,den);
ordf=size(Af,0);

Adh=[ Af zeros(ordf,ord); Bd*Cf Ad];
Bdh=[ Bf; zeros(ord,1) ];
Cdh=[ zeros(1,ordf) Cd ];
Ddh=[ 0 ];

ordd=size(Adh,0);

Aaug=[ Adh zeros(ordd,ord); zeros(ord,ordd) Ac];
Baug=[ zeros(ordd,1); Bc ];
Caug=[ Cdh  Cc ];
Daug=[ 0 ];
Eaug=[ Bdh; zeros(ord,1) ];

ordaug=size(Aaug,0);
```

```
%%%%% Finding the Value of K %%%%%

R=1e-8; %% 1e-8 narrow and wide
K=dlqr(Aaug,Baug,Caug'*Caug,R);  %% 0.930

%%%%% Finding the Value of L %%%%%

L=Aaug*(dlqe(Aaug,Eaug,Caug,1,1e-4));

%Acl=[Ad zeros(ord,ord) zeros(ord,ordaug);zeros(ord,ord) Ac -Bc*K; L*Cd L*Cc
Aaug-Baug*K-L*Caug];
%Bcl=[Bd; zeros(ord,1); zeros(ordaug,1)];
%Ccl=[Cd Cc zeros(1,ordaug)];
%Dcl=[ 0 ];

%[magcl,phacl,w]=dbode(Acl,Bcl,Ccl,Dcl,0.0005);
%[magol,phaol,w]=dbode(Ad,Bd,Cd,Dd,0.0005,1,w);

%semilogy(w/(2*pi),magcl,'r',w/(2*pi),magol,'y');

%break

%%%%% Initialization of States %%%%%

X=zeros(ord,1);
Xd=zeros(ord,1);
X_hat=zeros(ordaug,1);

Y=zeros(size(Input,1),1);

for i=1:size(Input,1)

%%%%% Compute Outputs %%%%%

  Y(i,1)=Cc*X+Cd*Xd;
  Y_hat=Caug*X_hat;

%%%%% Update States %%%%%

  X=Ac*X+Bc*(-K*X_hat);
  X_hat=Aaug*X_hat+Baug*(-K*X_hat)+L*(Y(i,1)-Y_hat);
  Xd=Ad*Xd+Bd*Input(i,:);

end
```

```
figure
subplot(2,1,1);
plot(Time,Input);
title('Wide Band Disturbance');
ylabel('Magnitude (Volts)');
subplot(2,1,2);
plot(Time,Y)
title('Optimal Disturbance Rejection Controller');
ylabel('Magnitude (Volts)');
xlabel('Time (Seconds)');
```

Appendix D

D.1 LMSFILT.M

```
%
%  LMS Algorithm Simulation of Clamped Beam Controller
%
%

loadmodl;

%%% Harmonic Frequency %%%
f=189; % Hz %
Time=0:0.0005:3;
Input=sin(2*pi*f*Time)';

%%% Narrow Band (189 Hz center 20 Hz BW) %%%
%[num,den]=butter(5,[0.179 0.199]);

%%% Wide Band (300 Hz center 250 Hz BW) %%%
%[num,den]=butter(5,[0.075 0.425]);

%Time=0:0.0005:2;
%randn('seed',s);
%prbs=sign(randn(size(Time,2),1));
%Input=dlsim(num,den,prbs);

%%%%% LMS Parameters %%%%%

M=3; % LMS Order %

w=zeros(M,1);
u=zeros(M,1);
uf=zeros(M,1);

mu=0.75;

ord=size(Ac,0);
ordd=size(Ad,0);

%%%%% Finding the Value of K %%%%%

Q=Cc'*Cc;
R=0.96;
K=dlqr(Ac,Bc,Q,R);
```

```
%%%%% Finding the Value of L %%%%%

L=dlqe(Ac,Bc,Cc,1,1e-4);
L=Ac*L;

%%%%% Initialization of States %%%%%

X=zeros(ord,1);
Xd=zeros(ordd,1);
X_hat=zeros(ord,1);
X_filt=zeros(ord,1);
Y_hat=0.0;
Y_filt=0.0;
S=0.0;

Y=zeros(size(Input,1),1);

for i=1:size(Input,1)

%%%%% Compute Outputs %%%%%

  S=w'*uf;

  Y(i,1)=Cc*X+Cd*Xd;
  Y_hat=Cc*X_hat;
  Y_filt=Cc*X_filt;

  if Y(i,1) > 5.0
    error('Too Big')
  end

%%% LQG %%%
%  X=Ac*X+Bc*(-K*X_hat);
%  X_hat=Ac*X_hat+Bc*(-K*X_hat)+L*(Y(i,1)-Y_hat);
%  Xd=Ad*Xd+Bd*Input(i,:);
%  X_filt=Ac*X_filt+Bc*(Y(i,1)-Y_hat);

%%% LMS %%%

  X=Ac*X+Bc*(-S-K*X_hat);
  X_hat=Ac*X_hat+Bc*(-S-K*X_hat)+L*(Y(i,1)-Y_hat);
  Xd=Ad*Xd+Bd*Input(i,:);
  X_filt=Ac*X_filt+Bc*(Y(i,1)-Y_hat);

%%%%% LMS Algorithm %%%%%
```

```
  err=Y(i,:);

  for j=M:-1:1
    w(j,1)=w(j,1)-mu*err*u(j,1);
  end

%%%%% Update Filter Inputs %%%%%

  for j=M:-1:2
    u(j,1)=u(j-1,1);
    uf(j,1)=uf(j-1,1);
  end

  u(1,1)=-Y_filt;
  uf(1,1)=(Y(i,1)-Y_hat);

end

figure
subplot(2,1,1);
plot(Time,Input);
title('Wide Band Disturbance');
ylabel('Magnitude (Volts)');
subplot(2,1,2);
plot(Time,Y)
title('Adaptive Supression of Disturbance');
ylabel('Magnitude (Volts)');
xlabel('Time (Seconds)');
```

D.2 HARMONIC.M

```
%
% Harmonic Simulation of Clamped Beam Controller
%
%

loadmodl;

%%% Harmonic Frequency %%%
f=189; % Hz %

%%%%% Harmonic Disturbance %%%%%

Time=0:0.0005:3;
Input=sin(2*pi*f*Time)';

[Out]=nocntrl(Ad,Bd,Cd,Dd,Input);

%%%%% Lattice Parameters %%%%%

M=3; % Lattice Order %

mu_nu=0.75;
mu_theta=0.03;

ord=size(Ac,0);
ordd=size(Ad,0);

%%%%% Lattice Filter Initialization %%%%%

Q_Latt=zeros(M+1,1);
D_Latt=zeros(M+1,1);
P_Latt=zeros(M+1,1);

nu=zeros(M+1,1);

grad_th=zeros(M,1);
Theta=zeros(M,1);

%%%%% Finding the Value of K %%%%%

Q=Cc'*Cc;
R=0.96;
K=dlqr(Ac,Bc,Q,R);
```

```
%%%%% Finding the Value of L %%%%%

L=dlqe(Ac,Bc,Cc,1,1e-4);
L=Ac*L;

%%%%% Initialization of States %%%%%

X=zeros(ord,1);
Xd=zeros(ordd,1);
X_hat=zeros(ord,1);
X_dist=zeros(ord,1);
X_filt=zeros(ord,1);

Y=zeros(size(Input,1),1);

for i=1:size(Input,1)

%%%%% Compute Outputs %%%%%

  Y(i,1)=Cc*X+Cd*Xd+0.01;
  Y_hat=Cc*X_hat;
  Y_filt=Cc*X_filt;

  if Y(i,1) > 5.0
    error('Too Big')
  end

%%%%% Lattice Filter ( D Filter ) %%%%%

  gm=Y_filt;

  for j=M:-1:1
    gm_hold=gm;
    gm=[cos(Theta(j)) -sin(Theta(j))]*[gm_hold; D_Latt(j)];
    D_Latt(j+1)=[sin(Theta(j)) cos(Theta(j))]*[gm_hold; D_Latt(j)];
  end
  D_Latt(1)=gm;

  s_filt=nu'*D_Latt;

%%%%% Lattice Filter ( Q Filter ) %%%%%

  gm=(Y(i,1)-Y_hat);

  for j=M:-1:1
    gm_hold=gm;
    gm=[cos(Theta(j)) -sin(Theta(j))]*[gm_hold; Q_Latt(j)];
    Q_Latt(j+1)=[sin(Theta(j)) cos(Theta(j))]*[gm_hold; Q_Latt(j)];
  end
  Q_Latt(1)=gm;

  s=nu'*Q_Latt;

%%%%% Update States %%%%%

  X=Ac*X+Bc*(-s-K*X_hat);
  X_hat=Ac*X_hat+Bc*(-s-K*X_hat)+L*(Y(i,1)-Y_hat);
  Xd=Ad*Xd+Bd*Input(i,:);
  X_filt=Ac*X_filt+Bc*(Y(i,1)-Y_hat);

%%%%% Adaptive Update %%%%%

  err=Y(i,:);

  gamma=1;
  for j=M:-1:1
    grad_th(j)=gamma*P_Latt(j);
    gamma=gamma*cos(Theta(j));
  end

  nu=nu+mu_nu*err*D_Latt;

  old_theta=Theta;
  Theta=Theta+mu_theta*err*grad_th;

  for j=M:-1:1
    if abs(Theta(j))>pi/2
      Theta(j)=old_theta(j);
    end
  end

  gm=-s_filt;
  for j=M:-1:1
    gm_hold=gm;
    gm=[cos(Theta(j)) -sin(Theta(j))]*[gm_hold; P_Latt(j)];
    P_Latt(j+1)=[sin(Theta(j)) cos(Theta(j))]*[gm_hold; P_Latt(j)];
  end
  P_Latt(1)=gm;

end

figure
```

```
subplot(2,1,1);
plot(Time,Input);
title('Harmonic Disturbance');
ylabel('Magnitude (Volts)');
subplot(2,1,2);
plot(Time,Y)
title('Adaptive Supression of Disturbance');
xlabel('Time (Seconds)');
ylabel('Magnitude (Volts)');

spectrum(Out,Y);

[top,bot]=ss2tf(Ac,Bc,Cc,Dc);
[h,w]=freqz(top,bot,4096);
sys_mag=abs(h);

[a,b]=lat2dir(nu,sin(Theta));

[h,w]=freqz(b,a,4096);
filt_mag=abs(1./h);

figure
semilogy(w*2000/(2*pi),sys_mag,'y')
hold on
semilogy(w*2000/(2*pi),filt_mag,'r')
hold off
xlabel('Frequency (Hz)')
ylabel('Magnitude')
```

D.3 NBANDWB.M

```
%
% Narrow Band and Wide Band Simulation of Clamped Beam Controller
%
%

loadmodl;

%%% Narrow Band (189 Hz center 20 Hz BW) %%%
%[num,den]=butter(5,[0.179 0.199]);

%%% Wide Band (300 Hz center 250 Hz BW) %%%
[num,den]=butter(5,[0.075 0.425]);

Time=0:0.0005:2;

randn('seed',s);

prbs=sign(randn(size(Time,2),1));
Input=dlsim(num,den,prbs);

[Out]=nocntrl(Ad,Bd,Cd,Dd,Input);

%%%%% Lattice Parameters %%%%%

M=3; % Lattice Order %

mu_nu=5;
mu_theta=0.10;

ord=size(Ac,0);
ordd=size(Ad,0);

%%%%% Lattice Filter Initialization %%%%%

Q_Latt=zeros(M+1,1);
D_Latt=zeros(M+1,1);
P_Latt=zeros(M+1,1);

nu=zeros(M+1,1);

grad_th=zeros(M,1);
Theta=zeros(M,1);

%%%%% Finding the Value of K %%%%%
```

```
Q=Cc'*Cc;
R=0.96;
K=dlqr(Ac,Bc,Q,R);

%%%%% Finding the Value of L %%%%%

L=dlqe(Ac,Bc,Cc,1,1e-4);
L=Ac*L;

%%%%% Initialization of States %%%%%

X=zeros(ord,1);
Xd=zeros(ordd,1);
X_hat=zeros(ord,1);
X_dist=zeros(ord,1);
X_filt=zeros(ord,1);

Y=zeros(size(Input,1),1);

for i=1:size(Input,1)

%%%%% Compute Outputs %%%%%

  Y(i,1)=Cc*X+Cd*Xd+randn*0.005;
  Y_hat=Cc*X_hat;
  Y_filt=Cc*X_filt;

  if Y(i,1) > 5.0
    error('Too Big')
  end

%%%%% Lattice Filter ( D Filter ) %%%%%

  gm=Y_filt;

  for j=M:-1:1
    gm_hold=gm;
    gm=[cos(Theta(j)) -sin(Theta(j))]*[gm_hold; D_Latt(j)];
    D_Latt(j+1)=[sin(Theta(j)) cos(Theta(j))]*[gm_hold; D_Latt(j)];
  end
  D_Latt(1)=gm;

  s_filt=nu'*D_Latt;

%%%%% Lattice Filter ( Q Filter ) %%%%%
```

```
  gm=(Y(i,1)-Y_hat);

  for j=M:-1:1
    gm_hold=gm;
    gm=[cos(Theta(j)) -sin(Theta(j))]*[gm_hold; Q_Latt(j)];
    Q_Latt(j+1)=[sin(Theta(j)) cos(Theta(j))]*[gm_hold; Q_Latt(j)];
  end
  Q_Latt(1)=gm;

  s=nu'*Q_Latt;

%%%%% Update States %%%%%

%%% LQG %%%
%  X=Ac*X+Bc*(-K*X_hat);
%  X_hat=Ac*X_hat+Bc*(-K*X_hat)+L*(Y(i,1)-Y_hat);
%  Xd=Ad*Xd+Bd*Input(i,:);
%  X_filt=Ac*X_filt+Bc*(Y(i,1)-Y_hat);

%%% Lattice %%%
  X=Ac*X+Bc*(-s-K*X_hat);
  X_hat=Ac*X_hat+Bc*(-s-K*X_hat)+L*(Y(i,1)-Y_hat);
  Xd=Ad*Xd+Bd*Input(i,:);
  X_filt=Ac*X_filt+Bc*(Y(i,1)-Y_hat);

%%%%% Adaptive Update %%%%%

  err=Y(i,:);

  gamma=1;
  for j=M:-1:1
    grad_th(j)=gamma*P_Latt(j);
    gamma=gamma*cos(Theta(j));
  end

  nu=nu+mu_nu*err*D_Latt;

  old_theta=Theta;
  Theta=Theta+mu_theta*err*grad_th;

  for j=M:-1:1
    if abs(Theta(j))>pi/2
      Theta(j)=old_theta(j);
    end
  end
  gm=-s_filt;
  for j=M:-1:1
    gm_hold=gm;
    gm=[cos(Theta(j)) -sin(Theta(j))]*[gm_hold; P_Latt(j)];
    P_Latt(j+1)=[sin(Theta(j)) cos(Theta(j))]*[gm_hold; P_Latt(j)];
  end
  P_Latt(1)=gm;

end

figure
subplot(2,1,1);
plot(Time,Input);
title('Narrow Band Disturbance');
ylabel('Magnitude (Volts)');
subplot(2,1,2);
plot(Time,Y)
title('Adaptive Supression of Disturbance');
ylabel('Magnitude (Volts)');
xlabel('Time (Seconds)');

spectrum(Out,Y);

[top,bot]=ss2tf(Ac,Bc,Cc,Dc);
[h,w]=freqz(top,bot,4096);
sys_mag=abs(h);

[a,b]=lat2dir(nu,sin(Theta));
[h,w]=freqz(b,a,4096);
filt_mag=abs(1./h);

figure
semilogy(w*2000/(2*pi),sys_mag,'y')
hold on
semilogy(w*2000/(2*pi),filt_mag,'r')
hold off
xlabel('Frequency (Hz)')
ylabel('Magnitude')
```

D.4 LOADMODL.M

```
%
% Loads the Model of the Clamped Beam
%

load c:\matlab\research\model\Ac.dat;
load c:\matlab\research\model\Bc.dat;
load c:\matlab\research\model\Cc.dat;
load c:\matlab\research\model\Dc.dat;

load c:\matlab\research\model\Ad.dat;
load c:\matlab\research\model\Bd.dat;
load c:\matlab\research\model\Cd.dat;
load c:\matlab\research\model\Dd.dat;
```

VITA

Stephen Hevey was born in Boston, Massachusetts to Richard and Kathleen Hevey. He was raised for most of his life in Malvern, Pennsylvania. Stephen received a Bachelors of Science from Penn State University in 1994, where he studied Electrical Engineering. He married Charlene Slone and moved to Virginia shortly after graduation. Steve accepted a job at Vatell Corporation in Blacksburg, Virginia and started working part time on his Masters in Electrical Engineering at Virginia Tech. In 1998 he completed his Masters of Science degree and received a promotion to Vice President of Engineering at Vatell.