REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	29.Oct.99	DISSERTATION

4. TITLE AND SUBTITLE
ACTIVE NARROWBAND DISTURBANCE REJECTION ON AN ULTRA QUIET PLATFORM

5. FUNDING NUMBERS

6. AUTHOR(S)
MAJ EDWARDS STEPHEN G

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NAVAL POSTGRADUATE SCHOOL

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
FY99-325

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With AFI 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS	15. NUMBER OF PAGES
	233
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

ACTIVE NARROWBAND DISTURBANCE REJECTION ON AN ULTRA QUIET PLATFORM

by

Stephen G. Edwards

September 1999

Dissertation Supervisor: Brij N. Agrawal
Co-Advisor: Roberto Cristi

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1999	Doctoral Dissertation

4. TITLE AND SUBTITLE
ACTIVE NARROWBAND DISTURBANCE REJECTION ON AN ULTRA QUIET PLATFORM

5. FUNDING NUMBERS

6. AUTHOR(S)
Edwards, Stephen G.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT

Vibration isolation on spacecraft is needed for imaging sensors, microgravity experiments, and other sensitive payloads. The preferred method thus far has been passive isolation because of its simplicity and low cost. Active vibration isolation and disturbance rejection will soon be more common as space qualified sensors, actuators and processors become more capable and affordable, and performance requirements increase. Spacecraft disturbances are typically periodic vibrations which are most effectively controlled through feedforward techniques. A popular choice of feedforward control methods for disturbance rejection is the Multiple Error Least Mean Squares (LMS) algorithm which requires a separately measured disturbance-correlated signal in its implementation. A new technique called "Clear Box" makes extensive use of identification to bring out information that is normally hidden or not used by traditional control methods. It allows operation in an information-rich environment with built-in fault tolerance, the ability to control unanticipated disturbances, and the ability to select which modes to control (if saturation of the actuators is a possibility or concern), all without the need for a separately measured disturbance-correlated signal. Experiments using both Multiple Error LMS and Clear Box on an Ultra Quiet Platform provide an effective demonstration of the advantages of the Clear Box Algorithm, including a new Adaptive Basis Method which allows control of rapidly varying frequencies.

14. SUBJECT TERMS
Vibration Isolation, Narrowband Disturbances, Deterministic Disturbances, Disturbance Rejection, UQP, Clear Box, LMS, Filtered-x LMS, Multiple Error LMS, Active Control, Feedforward Control, System Identification, Adaptive Basis

15. NUMBER OF PAGES
248

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ACTIVE NARROWBAND DISTURBANCE REJECTION ON AN ULTRA QUIET PLATFORM

Stephen G. Edwards
Major, United States Air Force
B.S., U.S. Air Force Academy, 1986
M.S., Air Force Institute of Technology, 1990

Submitted in partial fulfillment of the
requirements for the degree of

**DOCTOR OF PHILOSOPHY IN
AERONAUTICAL AND ASTRONAUTICAL ENGINEEREING**

from the

NAVAL POSTGRADUATE SCHOOL
September 1999

Author: _____
Stephen Edwards

Approved by:

_____
Brij Agrawal, Professor
Dept. of Aeronautics and Astronautics
Dissertation Supervisor

_____
Roberto Cristi, Associate Professor
Dept. of Electrical and Computer Engineering
Co-Advisor

_____
Joshua Gordis, Associate Professor
Dept. of Mechanical Engineering

_____
Isaac Kaminer, Associate Professor
Dept. of Aeronautics and Astronautics

_____
Gerald Lindsey, Chairman
Dept. of Aeronautics and Astronautics

Approved by: _____
Gerald Lindsey, Chairman, Department of Aeronautics and Astronautics

Approved by: _____
Anthony Ciavarelli, Acting Associate Provost for Instruction

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Vibration isolation on spacecraft is needed for imaging sensors, microgravity experiments, and other sensitive payloads. The preferred method thus far has been passive isolation because of its simplicity and low cost. Active vibration isolation and disturbance rejection will soon be more common as space qualified sensors, actuators and processors become more capable and affordable, and performance requirements increase. Spacecraft disturbances are typically periodic vibrations which are most effectively controlled through feedforward techniques. A popular choice of feedforward control methods for disturbance rejection is the Multiple Error Least Mean Squares (LMS) algorithm which requires a separately measured disturbance-correlated signal in its implementation. A new technique called "Clear Box" makes extensive use of identification to bring out information that is normally hidden or not used by traditional control methods. It allows operation in an information-rich environment with built-in fault tolerance, the ability to control unanticipated disturbances, and the ability to select which modes to control (if saturation of the actuators is a possibility or concern), all without the need for a separately measured disturbance-correlated signal. Experiments using both Multiple Error LMS and Clear Box on an Ultra Quiet Platform provide an effective demonstration of the advantages of the Clear Box Algorithm, including a new Adaptive Basis Method which allows control of rapidly varying frequencies.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

The author wants to acknowledge the assistance of fellow students, faculty, and staff in recovering from a laboratory flood in January of 1999. This research would not have been completed on time without their assistance and support. A great deal of credit goes to the author's dissertation supervisor, Prof. Brij Agrawal, for unfailing support of the project, and for providing guidance which helped determine the focus area of the research. Also providing invaluable assistance were Prof. Roberto Cristi who acted as advisor for six months, Prof. Minh Phan of Princeton University who provided guidance for implementing the Clear Box Algorithm and had the initial idea for the Adaptive Basis Method, Prof. Richard Longman of Columbia University who was always available for consultation, Dr. Neil Goodzeit of Lockheed Martin Missiles & Space who co-developed the Clear Box Algorithm with Prof. Phan, and provided some of the computer code that was modified for use on the UQP, Dr. Eric Anderson at CSA Engineering who provided advice and guidance on platform hardware issues, and the technical support crew at dSPACE, Inc. who helped with technical issues regarding the digital signal processor system.

Finally the author wants to thank his family for their constant encouragement and support, without which this project would have been far less "disturbance free."

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Isolation of mechanical vibration on spacecraft is becoming more of a necessity. The use of larger solar arrays and apertures leads to the presence of increased vibration energy due to flexible modes in the spacecraft structure. At the same time there is an inevitable progression toward higher performance requirements for missions involving imaging sensors, interferometers, and microgravity experiments leading to the need for improved isolation of these sensitive payloads.

Passive isolation presents a reliable, low-cost solution that is effective for attenuating high frequency vibrations. Passive techniques typically are not used for low frequency vibration isolation since the delicate nature of the resulting mechanism is unable to withstand the launch environment. The problem is compounded by the presence of low frequency modes that result from the increased dimensions of modern spacecraft structures. These modes are excited by periodic disturbances or impulsive disturbances such as sudden structural deformation due to thermal stress, attitude control thrusters, etc. Sources of periodic excitation include solar array drive motors, cryo pumps, and momentum wheels.

Active control techniques allow a significant performance enhancement over passive methods, but require sensors, actuators, and processing equipment which must all be light weight and energy efficient. As space qualified sensors, actuators, and

processors become more capable and affordable active control will become the logical choice for achieving performance goals.

Control techniques for rejection of disturbances are numerous and include classical feedback, modern feedback, disturbance accommodating control, disturbance observers, repetitive control, adaptive control, adaptive inverse control, adaptive feedforward control, and neural networks [Ref. 1]. These methods require some degree of knowledge of the system dynamics or the disturbances to be controlled. Adaptive techniques are best suited to situations where the system dynamics and/or disturbance characteristics are changing with time. In most cases narrowband (periodic) disturbances are most effectively controlled through the use of feedforward techniques.

A widely accepted adaptive feedforward method for noise and vibration control is the Filtered-x LMS Algorithm [Ref. 2], or the Multi-Input Multi-Output (MIMO) implementation called the Multiple Error LMS Algorithm [Ref. 3]. A drawback of these methods is that they require a separately measured, disturbance-correlated reference signal which is adaptively filtered to form the control signal. In many cases a disturbance source may be unanticipated (an example is the thermally-induced solar array vibrations experienced by the Hubble Telescope), and these techniques are ineffective against such disturbances if the sensor placement does not provide a strong, well-correlated signal. LMS-derived methods also require prior knowledge of the system's dynamics, which may vary with time due to loading and environmental changes, or changes in orientation of the solar arrays and communication/sensing apertures. Finally, the LMS algorithms do not provide the ability to selectively cancel disturbances. This ability to know which

disturbance frequencies to reject and which to ignore is important because it prevents actuator saturation when disturbance frequencies are coincident with, or close to, system modes that are uncontrollable or weakly controllable. It also allows the use of lighter weight, less capable actuators when weight is a primary concern (e.g., in spacecraft applications).

A new technique called Clear Box approaches the control problem from a system identification perspective [Ref.s 4,5]. Identification brings out information that is normally hidden or not used in traditional "black box" disturbance rejection methods. Requiring only the knowledge of actuator inputs and (disturbance-corrupted) sensor outputs, the Clear Box Algorithm allows complete identification of both the system dynamics and the disturbance frequencies. The identification, which can be performed in the presence of unknown periodic disturbances, results in a "disturbance-free" ARX (AutoRegressive with eXogenous Input) model which is then used to calculate a "disturbance effect" signal. This disturbance effect signal and the ARX system model provide the information needed to intelligently identify and selectively cancel disturbances while taking into account limited actuation resources. All of the required information can be extracted from identification alone without requiring measurement of a separate disturbance-correlated signal.

Thus, the shortcomings of the LMS-based algorithms are addressed while allowing consistent reduction of the error to the sensor noise level. This dissertation also implements a new control version of the Clear Box Algorithm called the "Adaptive Basis

Method" that is capable of controlling rapidly varying disturbance frequencies by using adaptive basis functions to synthesize the control signal.

B. LITERATURE REVIEW

1. Background

The earliest efforts to actively control sound (acoustic vibration) are traced to Paul Lueg of Germany [Ref. 6] who attempted to control sound propagating in a duct through superposition of a 180 degree phase-shifted sound wave, resulting in destructive interference at the desired point in the duct. Problems with standing waves arose that could not be solved with the technology available at that time, and interest in the subject did not resurface until the 1950's in the United States. Olson's "Electronic Sound Absorber" [Ref.s 7,8] successfully minimized sound levels using a microphone and a canceling source (speaker) in a room environment. However, the effectiveness of the system rapidly diminished as the distance between the canceling source and the control point (the microphone) was increased. As technological advances continued, interest in the subject increased. The problem of controlling sound in a duct was eventually solved by Jessel [Ref. 9] and Swinbanks [Ref. 10] who eliminated the standing wave problem encountered by Lueg by employing multiple cancellation sources. Swinbanks' work was continued and improved upon by Poole [Ref.s 11,12] from 1976-1978.

Up to this point the techniques employed for sound reduction consisted mainly of simple gain and phase adjustments to cause destructive interference, and the dynamics of

the system were not used in determining the canceling signal. The availability of better processors in the late 1970's led to the first use of signal processing techniques to actively control sound and (for the first time) structural vibration. These new techniques used an analytical model of the physical system to determine the appropriate control signal, with a great deal of new work appearing in the mid 1980's. Some notable examples include control of exhaust noise [Ref. 13], spinning spacecraft [Ref. 14], mass-spring-damper systems [Ref.s 15,16], vibration isolation platforms [Ref.s 17,18], rotors [Ref.s 19,20], flexible beams [Ref.s 21,22], and industrial machinery [Ref. 23].

The "model reference" approaches used to this point were susceptible to performance degradation if the system parameters changed due to environmental, load, or structural changes. System identification techniques were still being refined, and many physical systems were too complex to model accurately using analytical methods. This led to an increasing interest in adaptive control which had attained a solid foundation by the late 1970's [Ref. 24].

2. Adaptive Control

Adaptive control allows optimal performance in the presence of modeling errors or changing system parameters since the control parameters adapt to minimize an error signal, and are not strictly dependent on the model of the system being controlled. Examples of adaptive control algorithms that emerged from early work include Least Mean Squares (LMS) [Ref.s 25,26,27], Recursive Least Mean Squares (RLMS) [Ref. 28], and the Adaptive Lattice Filter [Ref.s 29,30]. The LMS Algorithm (Widrow, et. al.,

1975) employs a Finite Impulse Response (FIR) filter to generate a control signal from a reference input. The RLMS Algorithm (Feintuch, 1976) employs an Infinite Impulse Response (IIR) filter to accomplish the same result, but has met with resistance due to an inability to prove convergence [Ref. 31].

The adaptive nature of these controllers creates an inherently nonlinear control system, which results in the stability and convergence properties being more difficult to analyze than linear control systems [Ref. 32]. Proofs of stability of the LMS Algorithm have so far been restricted to linearized systems operating under the restricted condition of slow adaptation [Ref.s 33,34,35,36]. The advantage gained by the adaptive techniques is that small errors in the system model are compensated for by the adaptive controller. However, larger changes in the system dynamics require periodic re-identification in order to maintain optimal performance and stability.

3. System Identification

There are many methods available for identification of system dynamics [Ref. 37, 38,39,40]. The techniques include both batch (off-line) or recursive (on-line), parametric or non-parametric, and time domain or frequency domain. While the presence of unknown disturbances or noise is assumed, these are generally assumed to be white noise or random disturbances. It is shown in Ref. 41 that models identified in the presence of these periodic disturbances can have serious defects that render them unusable for control applications. The Clear Box Algorithm presented in Chapter II and implemented on the UQP effectively removes the model defects associated with unknown periodic

disturbances that are present during the identification process, and allows implementation of an adaptive feedforward controller based on this disturbance-free system model.

4. Application of Vibration Control to Spacecraft

Spacecraft applications for vibration control are numerous [Ref.s 42,43,44], including launch load alleviation [Ref.s 45,46,47], isolating the effects of noise-producing equipment [Ref.s 48,49,50,51,52,53], isolation of sensitive optics [Ref.s 54,55,56,57,58,59,60,61], and microgravity experiment isolation [Ref.s 62,63,64,65, 66,67,68,69,70,71,72,73]. Hexapod (or "Stewart") platform [Ref. 74] configurations similar to that employed in this research allow control of vibration in six degrees of freedom using only linear actuators [Ref.s 75,76,77,78,79,80].

In general, there are three types of on-orbit spacecraft control implementations including 1) isolation of a noise source from the structure, 2) direct control of the flexible structural members, and 3) isolation of a sensitive payload article. Adaptive control methods are appropriate for all three types of control implementations. The UQP apparatus employed in this research is configured for sensitive payload isolation, but could be adapted for noise source isolation as well.

C. THESIS OVERVIEW

Chapter II presents a detailed outline of the basic theory that supports the control algorithms used in this dissertation. After reviewing the Least Mean Squares (LMS) algorithm, and its evolution into the Filtered-x LMS (FXLMS) algorithm for vibration

and noise control, the FXLMS LMS algorithm is expanded to a form for multi-input, multi-output (MIMO) systems called the Multiple Error LMS algorithm.

Following a theoretical summary of the LMS algorithm, the new "Clear Box" algorithm is presented, including a summary of two distinct approaches to formulating control signals. The first is the original "Sine/Cosine Method" where estimates of the disturbance frequencies are used to generate a control signal made up of the combination of sine & cosine pairs. The second "Adaptive Basis Method", developed during the course of this research, generates a control signal based on the disturbance effect η (and multiple time-shifted versions of it), and does not require disturbance frequency estimates.

Chapter III is an overview of the experimental setup used for the vibration isolation experiments. A description is given of the Ultra Quiet Platform (UQP), and its actuators and sensors. Also described are the support electronics including the data collection and processing equipment.

Chapter IV Describes the system identification experiments performed on the UQP, and describes the process used to build and verify a reference model of the system dynamics. This model is then compared with a model obtained using the Clear Box algorithm in order to verify the accuracy of the results.

Chapter V presents a summary of the disturbance rejection experiments that were performed on the UQP. The first experiments in each section use the Multiple Error LMS algorithm, followed by experiments using the two methods of the Clear Box algorithm. The experiments include a variety of narrowband disturbances, including

single and multiple frequencies (and harmonics), and also disturbances that are either constant or time-varying in frequency and amplitude. Also demonstrated is the ability of the Clear Box algorithms to handle the case where a disturbance frequency coincides with that of an uncontrollable mode of the system. The performance of each algorithm, and the conclusions to be drawn from the experiments are discussed in Chapter VI.

Chapter VII presents a summary of the results and conclusions. Also included is a description of the unique contributions of this research, and recommendations for further work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. REVIEW OF THEORY

A. INTRODUCTION

In Chapter I the advantages and disadvantages of several control methods were discussed. From the literature, the Filtered-x LMS (FXLMS) algorithm appears to be the accepted choice for active sound and vibration control due to its simplicity and ease of implementation. However, the Clear Box algorithm approaches the problem from a system identification perspective, and as a result the algorithm operates in an information-rich environment. This added information allows a more intelligent approach to the disturbance rejection problem.

This chapter provides a review of the mathematical formulation of the FXLMS algorithm and the extension to the MIMO version called Multiple Error LMS. Following this review, an outline of the Clear Box identification and control formulation follows. These formulations are the basis for control experiments performed Chapter V.

B. LMS ALGORITHM

Since the FXLMS algorithm is derived from the Least Mean Squares (LMS) algorithm, we first present the LMS adaptive algorithm in order to introduce the features of FXLMS.

1. Formulation of the LMS Algorithm

The goal of the LMS algorithm is to use an nth order digital FIR filter W to generate a feedforward control signal $y(k)$ and minimize the mean square error of $\varepsilon(k)$, which represents the difference between $y(k)$ and the disturbance signal $d(k)$ (shown in Figure II-1). This mean square error will, from this point on, be referred to as $\xi(k)$. The algorithm requires a "reference signal" $x(k)$ that is correlated with the disturbance signal $d(k)$ in order for the controller to perform well, as will be shown later.

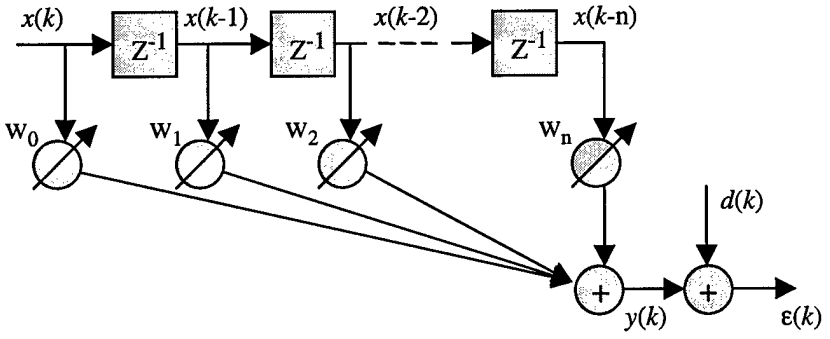

Figure II-1: LMS Filter

Each choice of the n+1 filter coefficients in W yields a different $\xi(k)$, leading to an n+2 dimensional performance surface. For example, a first order filter would have two coefficients (w_0 and w_1). Plotting the $\xi(k)$ with respect to these coefficients would result in the three-dimensional "performance surface" shown in Figure II-2.

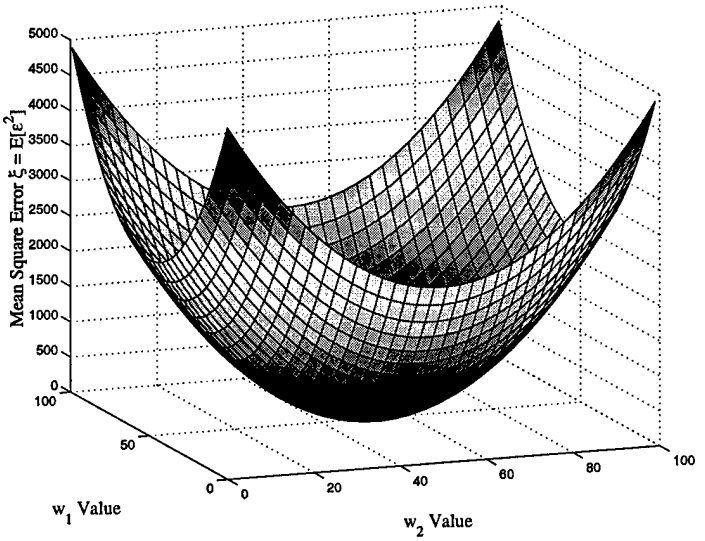

Figure II-2: Performance Surface

The LMS algorithm attempts to minimize $\xi(k)$ by "directing" the filter coefficients toward the minimum point on the performance surface. In general, gradient descent methods converge to a local minimum, unless the expression for $\xi(k)$ is convex in $\bar{W}$ in which case the local minimum is the global minimum. Referring again to Figure II-1, $\varepsilon(k)$ can be expressed as

$$\varepsilon(k) = d(k) + \bar{X}(k)^T \bar{W} \tag{2-1}$$

where

$$\bar{X}(k) = \begin{bmatrix} x(k) \\ x(k-1) \\ \vdots \\ x(k-n) \end{bmatrix}, \qquad \bar{W} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} \tag{2-2}$$

The mean square error $\xi(k)$ of $\varepsilon(k)$ is defined as

$$\xi(k) = E\left[\varepsilon(k)^2\right] \tag{2-3}$$

and it can be expressed in terms of the filter coefficients since

$$\varepsilon(k)^2 = d(k)^2 + \overline{W}^T \overline{X}(k)\overline{X}(k)^T \overline{W} + 2d(k)\overline{X}(k)^T \overline{W} \tag{2-4}$$

Applying the expectation operator to each term in (2-4) gives

$$E\left[\varepsilon(k)^2\right] = E\left[d(k)^2\right] + \overline{W}^T E\left[\overline{X}(k)\overline{X}(k)^T\right]\overline{W} + 2E\left[d(k)\overline{X}(k)^T\right]\overline{W} \tag{2-5}$$

or, in matrix form

$$\xi(k) = E\left[d(k)^2\right] + \overline{W}^T R\overline{W} + 2\overline{P}^T\overline{W} \tag{2-6}$$

where we define

$$\begin{aligned} R &= E\left[\overline{X}(k)\overline{X}(k)^T\right], \\ \overline{P} &= E\left[d(k)\overline{X}(k)\right] \end{aligned} \tag{2-7}$$

In this expression, R represents the input correlation matrix, and P represents the cross-correlation between $d(k)$ and the vector of delayed inputs, $\bar{X}(k)$. Note that there is a unique solution to the vector of filter weights, $\bar{W}$, if R is positive definite, which occurs when

$$n = 2f \tag{2-8}$$

and f is the number of frequency components in $d(k)$ [Ref. 81,82].

To converge toward the global minimum of the performance surface each filter weight is updated along the gradient of the surface, given by

$$\nabla \stackrel{\Delta}{=} \frac{\partial \xi}{\partial W} = \left[\frac{\partial \xi}{\partial w_0} \quad \frac{\partial \xi}{\partial w_1} \quad \cdots \quad \frac{\partial \xi}{\partial w_L} \right] = 2R\bar{W} + 2\bar{P} \tag{2-9}$$

The minimum error occurs at the global minimum point, where $\nabla = 0$, and $\bar{W} = \bar{W}^*$ (the * indicates the optimum solution). From Eq. (2-9) we now have

$$0 = 2R\bar{W}^* + 2\bar{P} \tag{2-10}$$

which results in the matrix form of the Weiner-Hopf equation [Ref. 83]

$$\bar{W}^* = -R^{-1}\bar{P} \tag{2-11}$$

Using this solution to determine an expression for the minimum mean square error of $\varepsilon(k)$ from Eq. (2-6) we obtain

$$\begin{aligned}
\xi_{\min} &= E\left[d(k)^2\right] + \overline{W}^{*T} R \overline{W}^{*} + 2\overline{P}^T \overline{W}^{*} \\
&= E\left[d(k)^2\right] + \left[R^{-1}\overline{P}\right]^T R R^{-1}\overline{P} - 2\overline{P}^T R^{-1}\overline{P} \\
&= E\left[d(k)^2\right] - \overline{P}^T R^{-1}\overline{P} \\
\xi_{\min} &= E\left[d(k)^2\right] - \overline{P}^T \overline{W}^{*}
\end{aligned} \tag{2-12}$$

To update the filter weights, we use

$$\overline{W}(k+1) = \overline{W}(k) + \mu(-\nabla(k)) \tag{2-13}$$

where μ is the "adaptation rate", which is a positive value. The exact computation of the gradient at time step k, $\nabla(k)$, is inefficient to calculate. The simplicity of the LMS algorithm comes from employing an estimate of the gradient. Since from Eq. (2-3) we have $\xi(k) = E\left[\varepsilon(k)^2\right]$, we can simply remove the expectation operator to obtain an estimate of $\xi(k)$. Now we use $\xi(k) \cong \varepsilon(\mathrm{k})^2$, and recall that $\varepsilon(k) = d(k) + \overline{X}(k)^T \overline{W}$ to obtain the estimate of the gradient at step k

$$\hat{\nabla}(\mathrm{k}) = \begin{bmatrix} \dfrac{\partial(\varepsilon(\mathrm{k})^2)}{\partial w_0} \\ \vdots \\ \dfrac{\partial(\varepsilon(\mathrm{k})^2)}{\partial w_L} \end{bmatrix} = 2\varepsilon(\mathrm{k}) \begin{bmatrix} \dfrac{\partial\varepsilon(\mathrm{k})}{\partial w_0} \\ \vdots \\ \dfrac{\partial\varepsilon(\mathrm{k})}{\partial w_L} \end{bmatrix} = 2\varepsilon(\mathrm{k})\bar{X} \tag{2-14}$$

Now the update to the filter weights is accomplished at each time step using the measured error and the reference input as follows

$$\bar{W}(k+1) = \bar{W}(k) - 2\mu\varepsilon(k)\bar{X} \tag{2-15}$$

The maximum adaptation rate that can be used (without causing instability) is given by

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{2-16}$$

where $\lambda_{\max}$ is the largest eigenvalue of the input correlation matrix R [Ref. 84]. An alternative upper bound is 1/tr[R], which is more restrictive but easier to calculate [Ref. 85]. The resulting convergence of the filter coefficients $\bar{W}$ achieves a global minimum mean square error $\xi(k)$.

2. Formulation of the Filtered-x LMS Algorithm

Note that the structure of the problem outlined for the LMS algorithm, above, differs from the disturbance rejection problem of interest here. Figure II-3 shows that there is a physical plant between the output of the filter W and the signal $\varepsilon(k)$.

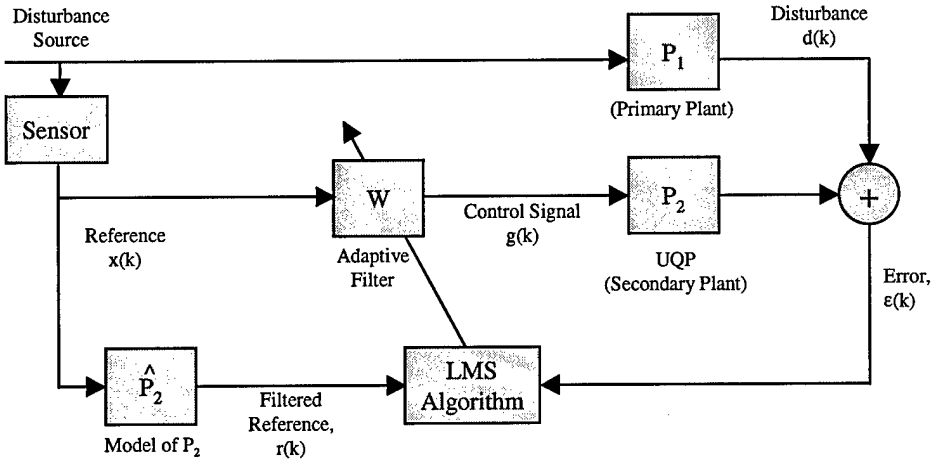

Figure II-3: Filtered-x LMS System Representation

The disturbance source acts on the physical system through the plant P_1 (with unknown dynamics) referred to as the "primary plant". A sensor provides a disturbance-correlated reference signal $x(k)$. The Filtered-x LMS algorithm determines an appropriate feedforward control signal $g(k)$ that acts on the system through the "secondary plant" P_2, which in this case is the UQP's active strut system (see Chapter III for a complete description of the UQP experiment apparatus). The error at the system output $\varepsilon(k)$ is the result of the actions of the disturbance source and $g(k)$ on the primary

and secondary plants, respectively. The existence of the plant P_2 between the filter and $\varepsilon(k)$ requires a change in the way the LMS algorithm is applied.

The key is to start with a form of the problem that minimizes the mean square error of $\varepsilon(k)$ at the output of the filter W, as shown in Figure II-4 a). Next we simply duplicate the P_2 box for an equivalent representation in Figure II-4 b). Finally we assume that the filter W is changing very slowly with time. We make this assumption so that the property of commutability between W and P_2 holds (i.e. $WP_2 = P_2W$), and thus make the final transition to Figure II-4 c). Adding the primary plant path results in the final configuration of the disturbance rejection problem originally shown in Figure II-3.

The reason for the "Filtered-x" designation of the algorithm comes from the fact that the reference input $x(k)$ is filtered by a model of the secondary plant before being used in the LMS algorithm to update the filter weights. Thus, the only difference between the equations used to implement the LMS and Filtered-x LMS algorithms is that the term $\overline{X}$ used in Eq. (2-15) consists of a filtered version of the original signal.

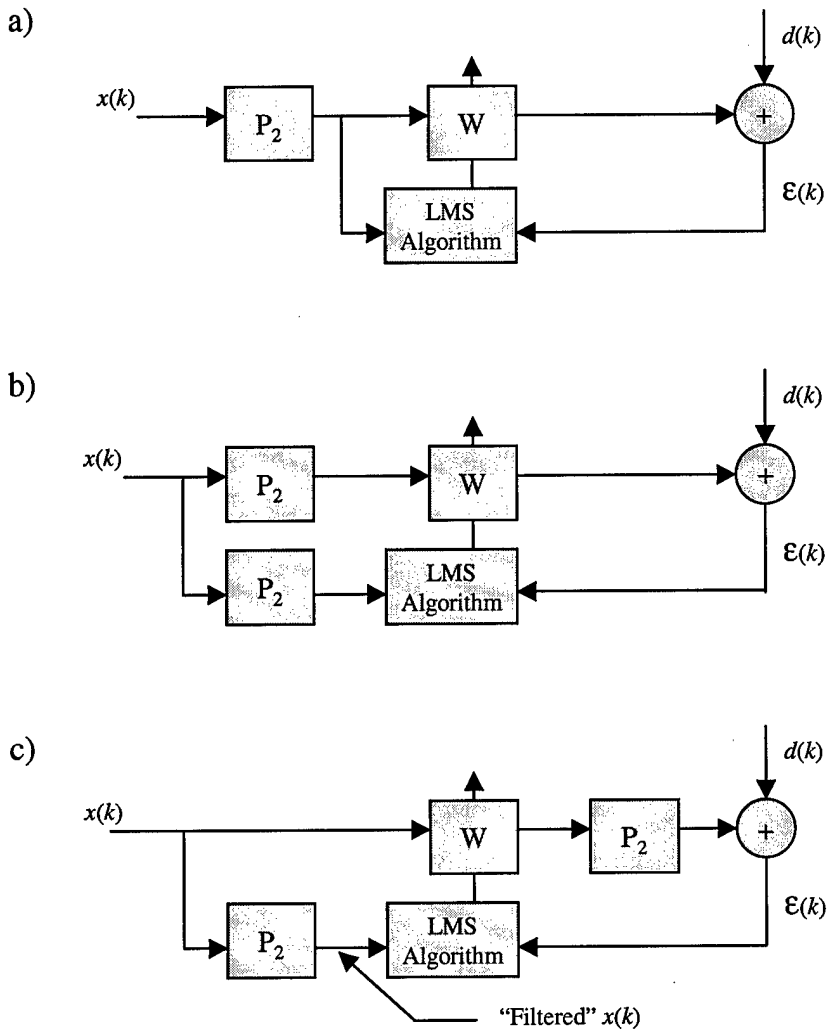

Figure II-4: Transformation to Filtered-*x* LMS

3. Extension to the Multiple Error LMS Algorithm

The UQP is configured similarly to a Stewart Platform [Ref. 86], which employs orthogonal struts to minimize coupling between a given strut's actuator and neighboring sensors on other struts. Under perfectly decoupled conditions the UQP could employ six

single-input-single-output (SISO) Filtered-x LMS controllers with no loss of performance. However, coupling does exist between struts (as will be shown in Chapter IV), and better performance results if a multi-input multi-output (MIMO) controller is used.

The Filtered-x LMS algorithm has been extended to a MIMO version [Ref. 87] called the Multiple Error LMS algorithm. For the development of this algorithm it is assumed that there are M actuators, and L sensors. Again, there is a reference signal $x(k)$ which passes through a "primary plant" before being sensed at the system output (Figure II-1) as $d(k)$. The disturbance at the lth sensor is represented by $d_l(k)$.

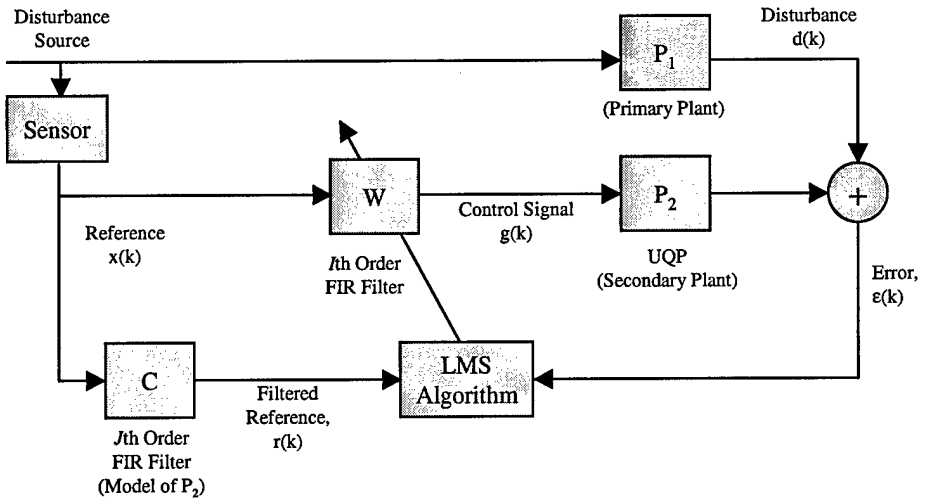

Figure II-5: Multiple Error LMS Algorithm

The plant model used to filter the reference signal is a Jth order Finite Impulse Response (FIR) filter C whose coefficients c_{lmj} indicate the jth coefficient ($j = 1, \ldots, J$)

for the filter that models the dynamics between the mth actuator and the lth sensor. The resulting filtered signal $r(k)$ includes LxM elements similarly indicated by $r_{lm}(k)$. The M control signals in $g(k)$ are generated by filtering the reference signal with an Ith order FIR filter W whose coefficients are w_{mi}. Finally, the error signal at each of the L sensors is indicated by $\varepsilon_l(k)$, an expression for which is

$$\varepsilon_l(k) = d_l(k) + \sum_{m=1}^{M} \sum_{j=0}^{J-1} c_{lmj} \sum_{i=0}^{I} w_{mi}(n-j)x(n-i-j) \tag{2-17}$$

As long as each $d_l(k)$ is partially correlated with $x(k)$ it is possible to reduce the error at each sensor through the proper choice of the coefficients w_{mi}. By defining the total error as

$$J = E\left\{\sum_{l=1}^{L} \varepsilon_l^2(k)\right\}, \tag{2-18}$$

from Eq.s (2-17) and (2-18) it is clear that J is a quadratic function of each of the coefficients w_{mi}, indicating that gradient descent methods allow convergence to a global minimum J. The differential of J with respect to one coefficient is

$$\frac{\partial J}{\partial w_{mi}} = 2E\left\{\sum_{l=1}^{L} \varepsilon_l(k) \frac{\partial \varepsilon_l(k)}{\partial w_{mi}}\right\} \tag{2-19}$$

Partially differentiating Eq. (2-17) with respect to w_{mi} we obtain

$$\frac{\partial \varepsilon_l(k)}{\partial w_{mi}} = \sum_{j=0}^{J-1} c_{lmj} x(k-i-j) \tag{2-20}$$

The above quantity is the same as that obtained by filtering the reference signal, delayed by i samples, with the FIR filter C, which is denoted by $r_{lm}(k-i)$. Thus we have

$$\frac{\partial \varepsilon_l(k)}{\partial w_{mi}} = r_{lm}(k-i) \tag{2-21}$$

Adjusting each filter coefficient in W by the negative of the gradient expression in Eq. (2-19), and using the expression in Eq. (2-21), we obtain

$$w_{mi}(k+1) = w_{mi}(k) - 2\mu \sum_{l=1}^{L} \varepsilon_l(k) r_{lm}(k-i) \tag{2-22}$$

where μ is, once again, the adaptation rate. When $L = M = 1$ this algorithm reduces to the result obtained in Eq. (2-15) for the LMS and FXLMS algorithms.

The assumption of time invariance in the w_{mi} filter coefficients is equivalent, in practice, to assuming that the filter coefficients change only slowly compared to the timescale of the response of the system to be controlled [Ref. 88].

4. Stability

Proofs of stability of the LMS Algorithm have so far been restricted to linearized systems operating under the restricted condition of slow adaptation [Ref.s 89,90,91,92, 93], and are typically limited to analysis of the single-input single-output case, although recent proofs have addressed the MIMO case [Ref. 94,95].

To maintain stability the adaptation rate μ for the SISO Filtered-x LMS Algorithm must be chosen less than the upper bound set in Eq. (2-16). A recent text by Fuller et. al. [Ref. 96] offers a guideline for the Multiple Error LMS Algorithm adaptation rate

$$0 < \mu < \frac{1}{2\overline{r}^2 I} \tag{2-23}$$

where $\overline{r}^2$ is the mean square level of the filtered reference signal $r(k)$, and I is the order of the adaptive filter.

C. CLEAR BOX ALGORITHM

1. System Representation

We approach the identification problem by assuming that the system can be represented by a linear discrete-time state space model of the form

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + B_d d(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2-24}$$

where $x(k)$ is an $n\times 1$ state vector, $u(k)$ is an $m\times 1$ input vector, and $y(k)$ is a $q\times 1$ output vector. Similarly the system A, B, and C matrices have dimensions $n\times n$, $n\times m$, and $q\times n$, respectively. The system is represented in Figure II-6. It is assumed that nothing is known except for the recorded system input $u(k)$, the disturbance-corrupted output data measurements $y(k)$, an upper bound on the true system order n, and an upper bound on the number of disturbance frequencies f.

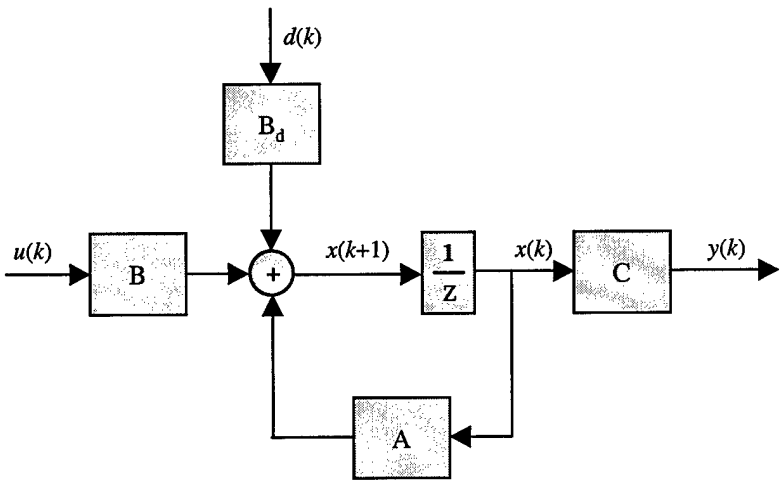

Figure II-6: System Representation for Clear Box Algorithm

2. *p*-Step Ahead Predictions

Eq. (2-24) is a one-step-ahead prediction of the state, based on the state and system input at the current time. By a recursive procedure we can determine the equations for a *p*-step-ahead predictor given by

$$x(k+p) = A^p x(k) + \mathbf{C}\, u_p(k) + \mathbf{C}_d d_p(k) \tag{2-25}$$

where $u_p(k)$ and $d_p(k)$ are vectors of the control inputs and disturbances,

$$u_p(k) = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+p-1) \end{bmatrix}, \qquad d_p(k) = \begin{bmatrix} d(k) \\ d(k+1) \\ \vdots \\ d(k+p-1) \end{bmatrix} \tag{2-26}$$

The matrices $\mathbf{c}$ and $\mathbf{c}_d$ are of a form similar to the controllability matrices associated with the control input and disturbance excitation, respectively, and are given by

$$\mathbf{C} = [A^{p-1}B, \ldots, AB, B], \qquad \mathbf{C}_d = [A^{p-1}B_d, \ldots, AB_d, B_d] \tag{2-27}$$

The output equation can similarly be propagated forward in time, with the result

$$y_p(k) = \mathbf{O}\, x(k) + \mathbf{T}\, u_p(k) + \mathbf{T}_d d_p(k) \tag{2-28}$$

where the elements of the equation are defined by

$$y_p(k) = \begin{bmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+p-1) \end{bmatrix}, \quad \mathbf{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{p-2} \\ CA^{p-1} \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ CB & 0 & \ddots & \ddots & \vdots \\ CAB & CB & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ CA^{p-2}B & \cdots & CAB & CB & 0 \end{bmatrix}, \quad \mathbf{T}_d = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ CB_d & 0 & \ddots & \ddots & \vdots \\ CAB_d & CB_d & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ CA^{p-2}B_d & \cdots & CAB_d & CB_d & 0 \end{bmatrix} \tag{2-29}$$

$\mathbf{o}$ is a $pq \times n$ matrix similar in form to the system observability matrix. $\mathbf{T}$ is a $pq \times pm$ Toeplitz matrix with its elements corresponding to the $q \times m$ system Markov parameters CB, CAB, ... , $CA^{p-2}B$, whose elements are the system response to a unit pulse applied at each control input.

3. Removing Dependence on the Initial State and the Disturbance Time History

Equations (2-25) and (2-28) represent the system input-output mapping, and at this point their solution depends on the initial state $x(k)$ and the disturbances $d_p(k)$. The goal is for the algorithm to rely solely on the input and output time histories ($u_p(k)$ and $y_p(k)$). Note that the system input signal $u_p(k)$ must have sufficiently rich frequency

content to excite all of the modes of the system. Application of band-limited white noise to the system can satisfy this condition.

To eliminate Eq. (2-25)'s dependence on $x(k)$ and $d_p(k)$ additional degrees of freedom are introduced into the model by adding and subtracting $My_p(k)$ to the right side, resulting in

$$x(k+p) = A^p x(k) + \mathbf{C}\, u_p(k) + \mathbf{C}_d d_p(k) + My_p(k) - My_p(k) \tag{2-30}$$

where M is arbitrary, and of dimension $n \times pq$. Substitution for $y_p(k)$ (from Eq. (2-28)) into the above equation yields

$$\begin{aligned} x(k+p) &= A^p x(k) + \mathbf{C}\, u_p(k) + \mathbf{C}_d d_p(k) + M\left(\mathbf{O}\, x(k) + \mathbf{T}\, u_p(k) + \mathbf{T}_d d_p(k)\right) - My_p(k) \\ &= \left(A^p + M\mathbf{O}\right)x(k) + \left(\mathbf{C} + M\mathbf{T}\right)u_p(k) + \left(\mathbf{C}_d + M\mathbf{T}_d\right)d_p(k) - My_p(k) \end{aligned} \tag{2-31}$$

In order to eliminate $x(k)$ and $d_p(k)$ from Eq. (2-31), M must be chosen to satisfy the following two conditions (for all values of k),

$$A^p + M\mathbf{O} = 0 \tag{2-32}$$

$$\left(\mathbf{C}_d + M\mathbf{T}_d\right)d_p(k) = 0 \quad \forall k \tag{2-33}$$

so that Eq. (2-31) becomes

$$x(k+p) = \left(\mathbf{C} + M\mathbf{T}\right)u_p(k) - My_p(k) \tag{2-34}$$

Equation (2-33) represents a set of constraints that must be satisfied for all values of $k = 1,\ 2, \ldots,\ N,\ N+1, \ldots$. These constraints can be grouped together such that

$$\left(\mathbf{C}_d + M\mathbf{T}_d\right)\mathbf{D} = 0 \tag{2-35}$$

where

$$\mathbf{D} = \left[d_p(1),\ d_p(2), \ldots\ d_p(N),\ d_p(N+1), \ldots\right] \tag{2-36}$$

Since the rows of the matrix D are made up of the time-shifted histories of the disturbance signal, and assuming that there are f distinct frequencies present in the disturbance signal, there is a limit to the possible rank of D even if the available time history is infinite. The maximum rank, ρ, of D is equal to $2f$, or $2f+1$ if any of the disturbances has non-zero mean. The result of this observation is that the maximum number of constraints represented by Eq. (2-35) is $n\rho$ (where n is the true order of the system). In order to show that a solution M exists that meets all of the required constraints, we will use the following arguments.

Let D_f be formed from ρ linearly independent columns of D, so that it is now $\rho \times \rho$. Similarly the dimensions of T_d and c_d can be reduced to $pq \times \rho$ and $n \times \rho$, respectively. Combining Eq.s (2-32) and (2-35), the equations that M must satisfy are

$$M\left[\mathbf{O},\ \mathbf{T}_d\mathbf{D}_f\right] = -\left[A^p,\ \mathbf{C}_d\mathbf{D}_f\right] \tag{2-37}$$

where, again, M is $n \times pq$, o is $pq \times n$, and A^p is $n \times n$. From the dimensions of the matrices we see that Eq. (2-37) represents $n^2 + n\rho$ linear equations in $n \times pq = npq$ unknowns in M. A solution for M exists if $\left[\mathbf{O},\ \mathbf{T}_d\mathbf{D}_f\right]$ is full (column) rank, and p is chosen such that $nqp \geq n^2 + \rho n$. Recalling that $\rho = 2f + 1$, this condition for the existence of M can be expressed as

$$p \geq \frac{n + 2f + 1}{q} \tag{2-38}$$

Thus, if an upper bound on the true system order n and the maximum number of frequencies that need to be controlled are known, p can be chosen such that a solution M exists. [Ref. 97]

4. Disturbance-Corrupted System Model

The condition expressed in Eq. (2-38) assures that the p-step ahead state prediction in Eq. (2-34) is valid. Pre-multiplying Eq. (2-34) by C results in an input-output model of the form

$$y(k+p) = C\left(\mathbf{C} + M\mathbf{T}\right)u_p(k) - CMy_p(k) \tag{2-39}$$

Shifting the time index back by p steps gives, for $k \geq p$

$$y(k) = C\left(\mathbf{C} + M\mathbf{T}\right)u_p(k-p) - CMy_p(k-p) \tag{2-40}$$

where

$$u_p(k-p) = \begin{bmatrix} u(k-p) \\ u(k-p+1) \\ \vdots \\ u(k-1) \end{bmatrix}, \qquad y_p(k-p) = \begin{bmatrix} y(k-p) \\ y(k-p+1) \\ \vdots \\ y(k-1) \end{bmatrix}$$

So we now have a model that predicts the next system output condition, given the last p sets of disturbance-corrupted input-output data. The input-output model in Eq. (2-40) has the same form as

$$y(k) = \alpha_1 y(k-1) + \alpha_2 y(k-2) + \cdots + \alpha_p y(k-p) + \beta_1 u(k-1) + \beta_2 u(k-2) + \cdots + \beta_p u(k-p) \tag{2-41}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_p$ and $\beta_1, \beta_2, \ldots, \beta_p$ are the model coefficients. These coefficients are related to the matrices in Eq. (2-40) as follows

$$\left[\alpha_p, \alpha_{p-1}, \ldots, \alpha_1\right] = -CM, \qquad \left[\beta_p, \beta_{p-1}, \ldots, \beta_1\right] = C\left(\mathbf{C} + M\mathbf{T}\right) \tag{2-42}$$

Note that since this model is derived from disturbance-corrupted data it will include modes that are not part of the system dynamics. Assuming the disturbances are narrowband (sinusoidal) in nature, the extraneous modes in Eq. (2-41) will have frequencies corresponding to those of the disturbances.

5. Disturbance-Free System Model

The identification of the system model may need to be accomplished under various conditions, depending on the application and the quality of the sensor data available. The system's dynamic model may be relatively constant, or may be rapidly time varying. For the UQP application, the dynamics of the system (from actuator input to sensor output) are fairly constant over time, and thus the identification can be accomplished in batch mode a single time, or periodically if conditions warrant (if a malfunction occurs, or adjustments are made, etc.) If the system model is known to

change with time, then the identification can be accomplished recursively so that constant updates are available.

a) Identification from Disturbance Corrupted Data

Assuming that the system identification must be done in the presence of disturbances that cannot be "turned off", the available input-output data will be corrupted by disturbance effects that are not part of the true system dynamics. The assumption is also made that the sensor data is contaminated with some degree of noise. In this case a least squares solution of the ARX model coefficients is warranted, and can be found using

$$\left[C\left(\mathbf{C}+M\mathbf{T}\right),\ -CM\right]=YV^{T}\left(VV^{T}\right)^{+} \tag{2-43}$$

where the Y and V matrices are formed from input-output data as follows,

$$Y=\left[y(p),\, y(p+1),\ldots,\, y(l)\right],\qquad V=\begin{bmatrix} u_p(0) & u_p(1) & \cdots & u_p(l-p) \\ y_p(0) & y_p(1) & \cdots & y_p(l-p) \end{bmatrix} \tag{2-44}$$

with $u_p(k)$ and $y_p(k)$ as in Eq. (2-26). The α_i and β_i coefficients are found from Eq.s (2-42) and (2-43). For an ARX model of order p to be generated, at least p data samples must be available. Since noise is assumed to be present, the use of substantially more data points will have the affect of averaging and will generally give a better result.

When using noisy data it is generally useful to choose p large enough to include all of the disturbance modes plus a number of "noise modes" or over-parameterization modes. At this point the identified system model coefficients will include three types of modes; true system modes, disturbance modes, and noise modes. The disturbance modes need to be removed in order to obtain the disturbance-free model. This process is described in the next two sections.

b) Disturbance Identification through Modal Decomposition

To facilitate the removal of the disturbance modes it is convenient to convert the ARX model to an equivalent state space observable canonical form

$$\begin{cases} z(k+1) = A_p z(k) + B_p u(k) \\ y(k) = C_p z(k) \end{cases} \tag{2-45}$$

where

$$A_p = \begin{bmatrix} \alpha_1 & I & 0 & \cdots & 0 \\ \alpha_2 & 0 & I & \ddots & \vdots \\ \alpha_3 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & I \\ \alpha_p & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad B_p = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_p \end{bmatrix}, \quad C_p = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{2-46}$$

Conversion of this model to modal form yields the state space equations

$$\begin{cases} w(k+1) = \Lambda w(k) + \Gamma u(k) \\ y(k) = \Omega w(k) \end{cases} \tag{2-47}$$

where Λ, Γ, and Ω are formed via similarity transformation;

$$\Lambda = T^{-1} A_p T, \qquad \Gamma = T^{-1} B_p, \qquad \Omega = C_p T \tag{2-48}$$

Assuming A_p is diagonalizable, the columns of the transformation matrix, T, are the eigenvectors of A_p. Each oscillatory mode of A_p results in a pair of eigenvectors that are complex conjugates of each other. In this case one column of T is formed from the real part of this eigenvector pair, and a second column is taken from the imaginary part. Any non-oscillatory (real) modes of the system will result in a single real eigenvector column added to T (the corresponding real eigenvalue is λ_i). Building the transformation matrix in this manner results in the diagonalized, or decoupled transition matrix

$$\Lambda = diag\left\{ \lambda_1, \ldots, \lambda_{n_r}, \begin{bmatrix} \sigma_1 & \omega_1 \\ -\omega_1 & \sigma_1 \end{bmatrix}, \ldots, \begin{bmatrix} \sigma_{\frac{n_c}{2}} & \omega_{\frac{n_c}{2}} \\ -\omega_{\frac{n_c}{2}} & \sigma_{\frac{n_c}{2}} \end{bmatrix} \right\} \tag{2-49}$$

where n_r and n_c are the number of real and complex modes, respectively, and $n_r + n_c = pq$. The complex eigenvalue pairs are represented by the 2×2 block matrices on the diagonal of Λ, the ith complex eigenvalue pair is $\sigma_i \pm j\omega_i$, where $i = 1, 2, \ldots, \frac{n_c}{2}$.

The output and input influence matrices are given by

$$\Omega = \left[c_r^{(1)}, c_r^{(2)}, \ldots, c_c^{(1)}, c_c^{(2)}, \ldots \right], \qquad \Gamma = \begin{bmatrix} b_r^{(1)} \\ b_r^{(2)} \\ \vdots \\ b_c^{(1)} \\ b_c^{(2)} \\ \vdots \end{bmatrix} \tag{2-50}$$

where $c_r^{(i)}, c_c^{(i)}$ and $b_r^{(i)}, b_c^{(i)}$ are the respective output and input influence coefficients associated with the real eigenvalues $\lambda_r^{(i)}$ and the complex conjugate eigenvalue pairs $\sigma_c^{(i)} \pm j\omega_c^{(i)}$. The pulse response of the i^{th} real or complex conjugate mode is given by

$$P_r^{(i)}(k) = c_r^{(i)} (\lambda_r^{(i)})^{k-1} b_r^{(i)}, \quad \text{and} \quad P_c^{(i)}(k) = c_c^{(i)} \begin{bmatrix} \sigma_c^{(i)} & -\omega_c^{(i)} \\ \omega_c^{(i)} & \sigma_c^{(i)} \end{bmatrix}^{k-1} b_c^{(i)} \tag{2-51}$$

and the total system pulse response is

$$P_t(k) = \sum_{i=1}^{n_r} P_r^{(i)}(k) + \sum_{i=1}^{n_c} P_c^{(i)}(k) = \sum_{i=1}^{n_m} P^{(i)}(k) \tag{2-52}$$

Again, n_r is the number of real eigenvalues, n_c is the number of complex conjugate pairs, and $n_m = n_r + n_c$.

Now it is possible to calculate the relative contribution of each mode to the total system pulse response by taking the inner product of each *modal* pulse response with the *total* system pulse response (over N samples), as

$$S^{(i)} = \sum_{k=1}^{N} P_t(k) \cdot P^{(i)}(k) \tag{2-53}$$

If the i^{th} mode's pulse response is well correlated with the total system pulse response then the elements of the $q \times m$ matrix $S^{(i)}$ will be relatively large, which is an indication that a mode is one of the true system modes, as opposed to a noise mode. To obtain a scalar "modal pulse response norm" discriminator for each mode [Ref. 98], the elements of $S^{(i)}$ can be summed so that

$$s^{(i)} = \sum_{k=1}^{q} \sum_{l=1}^{m} S_{kl}^{(i)} \tag{2-54}$$

The lower portion of Figure IV-10 on page 87 shows an example of how these norms are used to isolate the true system modes from the other modes present in the model (noise and disturbance modes).

The periodic disturbance modes are oscillatory with their z-plane poles being very close to the unit circle (just inside or just outside depending on the particular set of data used). This results in near zero (positive or negative) damping for the disturbance modes, which is to be expected for forced oscillations. Typically, if p is chosen large enough, the damping ratios of the disturbance modes will be at least one or two orders of magnitude smaller than those of the noise modes or true system modes, and they are easily identified (see the upper portion of Figure IV-10). Also, the accuracy of the identified frequency improves as p increases. The frequencies of the disturbances are thus readily determined when the model is converted to this modal form.

c) Disturbance Free ARX Model and the Disturbance Effect

Once the disturbance modes have been identified in the modal state space model, they can be removed by eliminating the corresponding rows and columns from Λ, Γ, and Ω. The resulting matrices are the "disturbance-free" versions, which are denoted by a "bar" on top; $\overline{\Lambda}$, $\overline{\Gamma}$, and $\overline{\Omega}$. These are simply the $\overline{A}$, $\overline{B}$, and $\overline{C}$ matrices converted by a similarity transformation, and they describe the same system. The only difference is that the $\overline{\Lambda}$, $\overline{\Gamma}$, $\overline{\Omega}$ model form uses a different state vector w related to x by the transformation $x(k) = Tw(k)$. Since the "intermediate step" of calculating the state

vector is not needed in this problem, and in both cases the same y(k) output is the result, it is equivalent to use "$\bar{A}$", "$\bar{B}$", and "$\bar{C}$" to describe the disturbance-free model in state space form.

Recall from Section 3 that a system input-output model in ARX form was found by using additional freedom introduced by the matrix M to eliminate the dependence of the model on the initial state and the disturbance input(s). The same approach is used here, but the goal is only to remove the dependence on the initial state and retain the "disturbance effect" for later use.

The p-step ahead state prediction equation using the disturbance-free system parameters is thus

$$\bar{x}(k+p) = \left(\bar{A}^p + \bar{M}\bar{\mathbf{O}}\right)\bar{x}(k) + \left(\bar{\mathbf{C}} + \bar{M}\bar{\mathbf{T}}\right)u_p(k) + \left(\bar{\mathbf{C}}_d + \bar{M}\bar{\mathbf{T}}_d\right)d_p(k) - \bar{M}y_p(k) \quad (2\text{-}55)$$

The $\bar{\mathbf{C}}$, $\bar{\mathbf{O}}$, and $\bar{\mathbf{T}}$ matrices are formed the same way as in Eq.s (2-27) and (2-29), but $\bar{A}$, $\bar{B}$, and $\bar{C}$ are used in place of A, B, and C. To remove the dependence on the initial state $\bar{x}(k)$, the condition that must be met is

$$\bar{A}^p + \bar{M}\bar{\mathbf{O}} = 0 \quad (2\text{-}56)$$

Since $\bar{\mathbf{O}}$ is full column rank (the identified system is fully observable), $\bar{M}$ can be found from

$$\bar{M} = -\left(\bar{A}\right)^{p}\left(\bar{\mathbf{O}}\right)^{+} \tag{2-57}$$

as long as p is chosen so that $pq \geq n$. This choice for the "arbitrary" matrix $\bar{M}$ eliminates the dependence on the initial state, but leaves the disturbance term intact. Pre-multiplying Eq. (2-55) by $\bar{C}$ gives the result

$$\bar{y}(k) = -\bar{C}\bar{M}y_p(k-p) + \bar{C}\left(\bar{\mathbf{C}} + \bar{M}\bar{\mathbf{T}}\right)u_p(k-p) + \bar{C}\left(\bar{\mathbf{C}}_d + \bar{M}\bar{\mathbf{T}}_d\right)d_p(k-p) \tag{2-58}$$

where

$$y_p(k-p) = \begin{bmatrix} u(k-p) \\ u(k-p+1) \\ \vdots \\ u(k-1) \end{bmatrix}, \qquad u_p(k-p) = \begin{bmatrix} u(k-p) \\ u(k-p+1) \\ \vdots \\ u(k-1) \end{bmatrix}, \qquad d_p(k-p) = \begin{bmatrix} d(k-p) \\ d(k-p+1) \\ \vdots \\ d(k-1) \end{bmatrix}$$

The entire term in Eq. (2-58) involving $d_p(k-p)$ (the last term on the right side) is simply a linear combination of the disturbance inputs to the system, and represents the forced response of the system to these disturbances. With this term referred to as $\eta(k)$ Eq. (2-58) becomes

$$\bar{y}(k) = -\bar{C}\bar{M}y_p(k-p) + \bar{C}\left(\bar{\mathbf{C}} + \bar{M}\bar{\mathbf{T}}\right)u_p(k-p) + \eta(k) \tag{2-59}$$

where this is in the form of an ARX (Auto-Regressive with eXogenous input) model

$$y(k) = \bar{\alpha}_1 y(k-1) + \bar{\alpha}_2 y(k-2) + \ldots + \bar{\alpha}_p y(k-p) + \bar{\beta}_1 u(k-1) + \bar{\beta}_2 u(k-2) + \ldots + \bar{\beta}_p u(k-p) + \eta(k) \tag{2-60}$$

and the coefficients are given by

$$\left[\bar{\alpha}_p, \bar{\alpha}_{p-1}, \ldots, \bar{\alpha}_1\right] = -\bar{C}\bar{M}, \qquad \left[\bar{\beta}_p, \bar{\beta}_{p-1}, \ldots, \bar{\beta}_1\right] = \bar{C}\left(\bar{\mathbf{c}} + \bar{M}\bar{T}\right) \tag{2-61}$$

The models given in Eq.s (2-41) and (2-60) are equivalent, but two very important differences are that 1) the coefficients are now representative of the disturbance-free system, and 2) the disturbance effect has been separated from the system dynamics. Note that the disturbance effect separation was accomplished mode-by-mode through column and row elimination. If desired, selected disturbance modes can be left in the system model, and would remain absorbed in the $\bar{\alpha}$ and $\bar{\beta}$ coefficients. This is desirable if a particular mode is determined to be uncontrollable or weakly controllable. This selectivity will be discussed further in Section 6 below.

It is possible to calculate the disturbance effect directly by rearranging Eq. (2-60) in the form

$$\eta(k) = y(k) - \bar{\alpha}_1 y(k-1) - \bar{\alpha}_2 y(k-2) - \ldots - \bar{\alpha}_p y(k-p) \\ - \bar{\beta}_1 u(k-1) - \bar{\beta}_2 u(k-2) - \ldots - \bar{\beta}_p u(k-p) \tag{2-62}$$

This allows a real-time calculation of the disturbance effect based on the last p measurements of the input-output data. The calculated disturbance effect in Eq. (2-62) contains all of the frequencies that were 1) identified as disturbance modes, and 2) removed from the system model.

6. Control Formulation

a) Choice of Control Signal Basis Functions

At this point the disturbance-free system model and the disturbance effect are known. For a time-invariant system, the disturbance-free model need only be found once through batch processing of the input-output data and elimination of the disturbance modes. Using this model, the disturbance effect is calculated from Eq. (2-62) in real-time and is available at each time step (after p data samples are taken). From Eq. (2-60), the feedforward control $u_f(k)$ that cancels the steady-state disturbances must satisfy

$$\bar{\beta}_1 u_f(k-1) + \bar{\beta}_2 u_f(k-2) + \ldots + \bar{\beta}_p u_f(k-p) = -\eta(k) \tag{2-63}$$

If the identified system model is non-minimum phase, the system zeros present in the $\bar{\beta}$ coefficients will cause the control signal to grow, unbounded, if an

attempt is made to recursively calculate $u_f(k)$ from Eq. (2-63). The alternative is to take advantage of the knowledge that the control signal needs to be made up of periodic components to cancel the periodic disturbances. Two options for such an approach are presented here which exploit the fact that the disturbance effect signal contains all of the disturbance frequencies (assuming they were not intentionally retained in the system's ARX model coefficients).

The first option is to use the sine function as a basis for the control signal. Analysis of the disturbance effect signal $\eta(k)$ yields estimates of the disturbance frequencies, and for each frequency present a sine/cosine "basis set" can be used to cancel the disturbance. Using the frequency estimates the coefficients for the sine and cosine terms are calculated recursively.

The second option exploits the fact that the *exact* disturbance frequencies are present in the disturbance effect signal $\eta(k)$. The time history of $\eta(k)$, like the sine function, is a basis that can be used to generate the disturbance-canceling control signal. The cosine function is simply a time-shifted sine function, and a similar time shifting of $\eta(k)$ will yield a valid basis set. The attractiveness of this option is that the disturbance frequencies do not need to be estimated since the exact frequencies are already present in $\eta(k)$.

The choice of which control formulation to use depends mainly on the rate of frequency variation expected in the disturbances, and somewhat on the available processing power. All of these considerations will be discussed in Chapter VI, but for now the steps needed to implement the two options are discussed below.

b) Control Formulation Using the Sine/Cosine Basis Set

The first step in generating a sine/cosine-based control signal is to estimate, as accurately as possible, the frequencies of the disturbances, and this can be done in two different ways. The first uses the system input-output data to generate a full system model as described in Section 5.b). The disturbance frequencies are then estimated by finding the modes that have near-zero damping. A major shortcoming of this approach is that the control signal needs to be turned off while the input-output data is recorded, otherwise the disturbance modes will be suppressed by the control signal, and will not be manifested in the system model. Repeatedly turning off the control signal will likely result in an unacceptable loss of performance. Also, for time-invariant systems the full system model does not have to be recalculated unless the system has been damaged or adjusted, and repeated re-identification is computationally inefficient.

The second approach to estimating the disturbance frequencies requires fitting an autoregressive (AR) model to the disturbance effect data. Such a model would have the form

$$\eta(k) = \gamma_1 \eta(k-1) + \gamma_2 \eta(k-2) + \ldots + \gamma_\tau \eta(k-\tau) \tag{2-64}$$

and the model order τ must be chosen such that

$$q\tau > 2f \tag{2-65}$$

where f is the number of disturbance frequencies present in $\eta(k)$. The AR model can then be analyzed to find the frequencies of the modes with near-zero damping (the disturbance frequencies). This can be done in a similar fashion to the system identification method used earlier (conversion to modal state-space form, etc.), or the roots of the difference equation in Eq. (2-64) can be used to find the z-plane pole locations (which yield the frequencies and damping ratios). Since the disturbance effect is the same whether or not the control system is operating, there is no need to turn off the controller to identify the disturbance frequencies.

By adopting a feedforward control signal of the form

$$u_f(k) = \sum_{i=1}^{L} [a_i \cos(\omega_i k\Delta t) + b_i \sin(\omega_i k\Delta t)] \tag{2-66}$$

(where Δt is the controller sample time) it is possible to cancel the identified disturbance frequencies ω_i, $i = 1, 2, \ldots, L$. To solve for the control coefficients, Eq. (2-66) is substituted into Eq. (2-63) to obtain a set of equations that is linear in a and b, and thus a solution can be obtained recursively in real-time.

Recognizing that periodic re-identification of the disturbance frequencies could result in abrupt changes in the ω_i values, it is desirable to replace the term $\omega_i k\Delta t$ in (2-66) with $\theta_i(k)$ which is updated at each time step using

$$\theta_i(k) = \theta_i(k-1) + \Delta\theta_i, \qquad \text{and} \qquad \Delta\theta_i = \omega_i \Delta t \tag{2-67}$$

Eq. (2-63) can now be put in the form

$$\eta(k) = -\phi_f^T(k)\psi_f \tag{2-68}$$

where

$$\phi_f^T(k) = \left[G_1(k), \ldots, G_L(k), H_1(k), \ldots, H_L(k)\right], \qquad \psi_f = \begin{bmatrix} a_1 \\ \vdots \\ a_L \\ b_1 \\ \vdots \\ b_L \end{bmatrix} \tag{2-69}$$

and the $q \times m$ matrices $G_i(k)$ and $H_i(k)$, $i = 1, 2, \ldots, L$, are given by

$$G_i(k) = \sum_{l=1}^{p} \overline{\beta}_l \cos\left(\theta_i(k) - l\Delta\theta_i\right), \qquad H_i(k) = \sum_{l=1}^{p} \overline{\beta}_l \sin\left(\theta_i(k) - l\Delta\theta_i\right) \tag{2-70}$$

Equation (2-68) is in a form which can be solved with Recursive Least Squares (RLS) using the following set of equations

$$\psi_f(k) = \psi_f(k-1) + L_f(k)\left[\eta(k) - \hat{\eta}(k)\right] \tag{2-71}$$

where

$$\hat{\eta}(k) = -\phi_f^T(k)\psi_f(k-1) \tag{2-72}$$

$$L_f(k) = P_f(k-1)\phi_f(k)\left[\phi_f^T(k)P_f(k-1)\phi_f(k) + \lambda_f I_{q\times q}\right]^{-1} \tag{2-73}$$

$$P_f(k) = \frac{\left[P_f(k-1) - L_f(k)\phi_f^T(k)P_f(k-1)\right]}{\lambda_f} \tag{2-74}$$

where λ_f is the "forgetting factor" which determines the relative data weighting [Ref. 99].

As soon as the first disturbance frequency estimate is available, the recursive solution of the control coefficients is initiated, and the control signal is calculated from Eq. (2-66), using the $\theta_i(k)$ substitution discussed earlier. The initial a_i and b_i values are set to zero, and the initial covariance matrix, $P_f(0) = \gamma I_{2mL\times 2mL}$ where γ is large compared to the order of magnitude of the parameters to be estimated. Convergence to steady-state values will occur if the disturbance frequencies and amplitudes are constant and the frequency estimates are exact. If the frequencies and/or amplitudes are varying with time, then recursive estimation and regular disturbance frequency estimates allow the coefficients to track the correct solution. The performance of the controller depends on the accuracy of the frequency estimates provided. If the estimates are inaccurate, or the true disturbance frequency drifts between estimates, the recursive estimation allows the coefficients to "cycle" at a frequency equal to the difference between the true and estimated frequencies.

To demonstrate this, assume ω_t is the true disturbance frequency, and ω_e is the estimated frequency, and define

$$\omega_\Delta = \omega_t - \omega_e \,, \quad \text{or} \quad \omega_t = \omega_e + \omega_\Delta \tag{2-75}$$

If the recursively calculated feedforward control signal $u_f(k)$ is correctly compensating for the disturbance, as will be demonstrated in Chapter V, then it must be composed of sinusoidal components that have frequency ω_t such that

$$u_f(k) = \gamma \sin(\omega_t k) + \delta \cos(\omega_t k), \quad \text{where} \quad k \equiv k\Delta t \tag{2-76}$$

However, the control signal is calculated from estimates of the disturbance frequency, ω_e, and so we also have

$$u_f(k) = \alpha(k)\sin(\omega_e k) + \beta(k)\cos(\omega_e k) \tag{2-77}$$

where $\alpha(k)$ and $\beta(k)$ are the time-varying feedforward coefficients. Equating the two expressions for $u_f(k)$, we obtain

$$\gamma \sin(\omega_t k) + \delta \cos(\omega_t k) = \alpha(k)\sin(\omega_e k) + \beta(k)\cos(\omega_e k) \tag{2-78}$$

Substituting for ω_t from Eq. (2-75) we have

$$\gamma \sin((\omega_e + \omega_\Delta)k) + \delta \cos((\omega_e + \omega_\Delta)k) = \alpha(k)\sin(\omega_e k) + \beta(k)\cos(\omega_e k) \tag{2-79}$$

Using trigonometric identities for angle summations, and rearranging terms yields

$$\begin{aligned} &\gamma\left[\sin(\omega_e k)\cos(\omega_\Delta k) + \cos(\omega_e k)\sin(\omega_\Delta k)\right] + \\ &\delta\left[\cos(\omega_e k)\cos(\omega_\Delta k) - \sin(\omega_e k)\sin(\omega_\Delta k)\right] = \alpha(k)\sin(\omega_e k) + \beta(k)\cos(\omega_e k) \end{aligned} \tag{2-80}$$

Grouping terms we obtain

$$\begin{aligned} &\left[\gamma\cos(\omega_\Delta k) - \delta\sin(\omega_\Delta k) - \alpha(k)\right]\sin(\omega_e k) + \\ &\left[\gamma\sin(\omega_\Delta k) + \delta\cos(\omega_\Delta k) - \beta(k)\right]\cos(\omega_e k) = 0 \end{aligned} \tag{2-81}$$

Setting the bracketed terms equal to zero, and using

$$\begin{aligned} A &\equiv \sqrt{\gamma^2 + \delta^2} \\ \phi_1 &\equiv \tan^{-1}\left(\frac{-\delta}{\gamma}\right) \\ \phi_2 &\equiv \tan^{-1}\left(\frac{\gamma}{\delta}\right) \end{aligned} \tag{2-82}$$

we arrive, finally, at expressions for $\alpha(k)$ and $\beta(k)$;

$$\alpha(k) = A\sin(\omega_\Delta k + \phi_1),$$
$$\beta(k) = A\sin(\omega_\Delta k + \phi_2) \tag{2-83}$$

which demonstrates that if the frequency estimate is incorrect, the control signal coefficients will vary with time, and will cycle sinusoidally at a frequency ω_Δ equal to the difference between the true and estimated frequencies. An experimental demonstration of this cycling is provided in Section V.C.1.b).

A unique feature of the Clear Box algorithm is that it allows selective control of each identified disturbance frequency. If the disturbance frequency happens to be located at a frequency which is uncontrollable (or weakly controllable) the resulting control signal required to cancel the disturbance would have a large magnitude, which could saturate the actuator(s). To prevent this from happening, logic is implemented that prevents any attempt to control these frequencies. To accomplish this, the disturbance frequency in question is simply eliminated from the list of disturbance frequencies sent to the DSP controller. In general, both the system dynamics and the disturbance frequencies may be varying with time. In this case, disturbance frequencies can be selected for control based on the amount that each uncontrolled disturbance affects the response of the system, and also on the magnitude of the control signal required to control the disturbance. In Eq. (2-66), the Euclidean norm of the coefficients associated with each sine/cosine pair represents the magnitude of the control signal required for that disturbance frequency.

To determine the effect each disturbance, if uncontrolled, would have on the system output, Eq. (2-60) is used to obtain

$$y_d(k) - \bar{\alpha}_1 y_d(k-1) - \bar{\alpha}_2 y_d(k-2) - \ldots - \bar{\alpha}_p y_d(k-p) = \eta(k) \tag{2-84}$$

Assuming the system is linear, it is known that the steady state response to the disturbances present is a linear combination of the harmonic components

$$y_d(k) = \sum_{i=1}^{L} [c_i \sin(\omega_i k \Delta t) + d_i \cos(\omega_i k \Delta t)] \tag{2-85}$$

Substituting Eq. (2-85) into Eq. (2-84) gives an expression that is linear in the response coefficients, c_i and d_i. Batch or recursive methods can be used to solve for these coefficients, and the Euclidean norm of each sine/cosine pair represents the magnitude of the system response for each disturbance frequency. Those frequencies that have a low ratio of system response magnitude to control signal magnitude can thus be de-selected for control if actuator saturation is a possibility.

c) Control Formulation Using the Disturbance Effect as a Basis Set

In applications where the disturbance frequencies are known to vary, an approach that does not rely on frequency estimation is desirable. Such an approach would not only eliminate the processing requirements of estimating the disturbance

frequencies, but would also improve performance since the frequency estimates would not be old or otherwise inaccurate. With this motivation, a new Adaptive Basis Method is presented to address the case of rapidly varying disturbances, and to eliminate the need for disturbance frequency estimates.

Comparison of Eq.s (2-41) and (2-60) shows that all of the frequencies eliminated from the disturbance-corrupted system model are now present in the disturbance effect term $\eta(k)$. Consider an example where it is desired to control a single sinusoidal disturbance with a frequency of 25 Hz using a controller operating at a sampling rate of 400 Hz. This means there will be 16 data samples per disturbance period, and $\eta(k)$ will contain a single sinusoid of the same frequency (25 Hz). Shifting this sinusoid by 90 degrees, or 4 samples, results in an orthogonal signal and gives a basis set similar to the sine/cosine functions, and a controller of the form

$$u_f(k) = \psi_1 \eta(k) + \psi_2 \eta(k-4) \tag{2-86}$$

can be used to satisfy the controller requirement expressed in Eq. (2-63). The ψ_1 and ψ_2 coefficients can be solved for recursively in the same manner as a_i and b_i in Eq. (2-69).

Note that it is not necessary to have orthogonal basis functions for the controller to operate successfully. As long as the functions are not linearly dependent (in the example this would occur if the signal were shifted by a multiple of 8 samples), then the disturbance can be completely cancelled. The method works equally well when there

are f disturbance frequencies, as long as there are N time-shifted $\eta(k)$ basis functions such that

$$N \geq 2f + 1 \tag{2-87}$$

For the general case, a feedforward control signal $u_f(k)$ must satisfy Eq. (2-63), which is repeated here for convenience,

$$\bar{\beta}_1 u_f(k-1) + \bar{\beta}_2 u_f(k-2) + \ldots + \bar{\beta}_p u_f(k-p) = -\eta(k)$$

The assumed form of the control signal is

$$u_f(k) = \psi_1 \eta(k-\Delta_1) + \psi_2 \eta(k-\Delta_2) + \ldots + \psi_N \eta(k-\Delta_N) \tag{2-88}$$

where Δ_i $(i = 1, \ldots, N)$ are the number of samples that the disturbance effect has been shifted to generate the N basis functions. Each Δ_i value is chosen by the operator, and the guidelines below should be followed when selecting them.

$$\begin{aligned} \Delta_i &\geq 1 && \forall\, i \\ \Delta_i &\neq \Delta_j && \forall\, i, j \\ |\Delta_i - \Delta_j| &\neq |\Delta_j - \Delta_k| && \forall\, i, j, k \end{aligned} \tag{2-89}$$

The first guideline prevents any problems with causality by using the disturbance effect that is delayed by at least one time sample. The second ensures that two functions do not have the same time shift (such a pair would be identical functions). The third introduces a random characteristic to the time shifting, and prevents linear dependence of the basis functions for any given disturbance frequency.

Note that $u_f(k)$ is $m\times 1$, $\eta(k)$ is $q\times 1$, and thus each ψ_i matrix is $m\times q$. For the UQP application the computational requirements of the algorithm can be reduced by noticing that the disturbance effect signal for sensor i has the same frequencies as the signal in sensor j ($\forall\, i, j$). It is likely that the disturbance effect signal at one sensor could have amplitude or phase differences, but each strut's control coefficients can be recursively adjusted to account for unique amplitude and phase requirements. Thus, the $\eta(k)$ time history for any strut can be used as the basis function for controlling all of the struts, and using this information a new form of the control signal is chosen as

$$u_f(k)=\psi_1\eta^*(k-\Delta_1)+\psi_2\eta^*(k-\Delta_2)+\ldots+\psi_N\eta^*(k-\Delta_N) \tag{2-90}$$

where $\eta^*(k)$ is the scalar disturbance effect signal at the strut chosen to act as the "basis strut". Now each ψ_i parameter ($i=1,\ldots,N$) has dimension $m\times 1$ instead of $m\times q$. The first step toward solution of these parameters is to progressively time shift Eq. (2-90) as follows

$$
\begin{aligned}
u_f(k-1) &= \psi_1\eta^*(k-\Delta_1-1)+\psi_2\eta^*(k-\Delta_2-1)+\ldots+\psi_N\eta^*(k-\Delta_N-1)\\
u_f(k-2) &= \psi_1\eta^*(k-\Delta_1-2)+\psi_2\eta^*(k-\Delta_2-2)+\ldots+\psi_N\eta^*(k-\Delta_N-2)\\
&\vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \qquad\qquad\qquad \vdots\\
u_f(k-p) &= \psi_1\eta^*(k-\Delta_1-p)+\psi_2\eta^*(k-\Delta_2-p)+\ldots+\psi_N\eta^*(k-\Delta_N-p)
\end{aligned}
\tag{2-91}
$$

Substitution of these expressions into Eq. (2-63) results in

$$
\begin{aligned}
&\bar{\beta}_1\psi_1\eta^*(k-\Delta_1-1)+\bar{\beta}_1\psi_2\eta^*(k-\Delta_2-1)+\ldots+\bar{\beta}_1\psi_N\eta^*(k-\Delta_N-1)+\\
&\bar{\beta}_2\psi_1\eta^*(k-\Delta_1-2)+\bar{\beta}_2\psi_2\eta^*(k-\Delta_2-2)+\ldots+\bar{\beta}_2\psi_N\eta^*(k-\Delta_N-2)+\ldots+\\
&\bar{\beta}_p\psi_1\eta^*(k-\Delta_1-p)+\bar{\beta}_p\psi_2\eta^*(k-\Delta_2-p)+\ldots+\bar{\beta}_p\psi_N\eta^*(k-\Delta_N-p)=-\eta(k)
\end{aligned}
\tag{2-92}
$$

Since $\eta^*(k)$ is a scalar sequence, Eq. (2-92) can be rearranged as

$$
\begin{aligned}
&\bar{\beta}_1\eta^*(k-\Delta_1-1)\psi_1+\bar{\beta}_1\eta^*(k-\Delta_2-1)\psi_2+\ldots+\bar{\beta}_1\eta^*(k-\Delta_N-1)\psi_N+\\
&\bar{\beta}_2\eta^*(k-\Delta_1-2)\psi_1+\bar{\beta}_2\eta^*(k-\Delta_2-2)\psi_2+\ldots+\bar{\beta}_2\eta^*(k-\Delta_N-2)\psi_N+\ldots+\\
&\bar{\beta}_p\eta^*(k-\Delta_1-p)\psi_1+\bar{\beta}_p\eta^*(k-\Delta_2-p)\psi_2+\ldots+\bar{\beta}_p\eta^*(k-\Delta_N-p)\psi_N=-\eta(k)
\end{aligned}
\tag{2-93}
$$

and rewritten in vector form as

$$
-\eta(k)=\Phi^T(k)\Psi(k) \tag{2-94}
$$

where we define the following vectors

$$\Phi^T(k) = \begin{bmatrix} \phi_1(k) & \phi_2(k) & \cdots & \phi_N(k) \end{bmatrix} \tag{2-95}$$

$$\phi_i(k) = \sum_{j=1}^{p} \overline{\beta}_j \eta^*(k - \Delta_i - j), \qquad i = 1, 2, \ldots, N \tag{2-96}$$

and

$$\Psi(k) = \begin{bmatrix} \psi_1(k) \\ \psi_2(k) \\ \vdots \\ \psi_N(k) \end{bmatrix} \tag{2-97}$$

Each parameter $\phi_i(k)$ is a $q \times m$ matrix, making $\Phi^T(k)$ a $q \times Nm$ matrix. The vector $\Psi(k)$ of the feedforward control parameters has dimension $Nm \times 1$.

Having the set of linear equations in $\phi_i(k)$ in the form of Eq. (2-94) facilitates use of the RLS algorithm described below

$$\begin{aligned} \hat{\Psi}(k) &= \hat{\Psi}(k-1) + L(k)\left[-\eta(k) - \Phi^T(k)\hat{\Psi}(k-1)\right] \\ P(k) &= P(k-1) - L(k)\left[\Phi^T(k)P(k-1)\right] \end{aligned} \tag{2-98}$$

where

$$L(k) = P(k-1)\Phi(k)\left[\Phi^T(k)P(k-1)\Phi(k) + R\right]^{-1} \tag{2-99}$$

The estimated feedforward control coefficients in $\hat{\Psi}(k)$ are used in the control input calculation in Eq. (2-90). The computational requirements of the above algorithm are almost identical to that of the sine/cosine controller described in the previous section, except there is no need to compute disturbance frequency estimates.

In the case where there are uncontrollable modes known to be present in the system, or if actuator control ability is limited, it may be desirable to use selective cancellation of disturbances (as with the sine/cosine method). This can be implemented using the Adaptive Basis Method through selective filtering of the disturbance effect signal (demonstrated in Section V.E.2).

7. Stability

A stability analysis conducted in Ref. 100 (for a SISO system controlling a single disturbance frequency) demonstrates theoretically that the Clear Box Sine/Cosine Method has a phase margin of ± 90 degrees at the frequency of the disturbance, assuming a slow adaptation rate (which equates to a forgetting factor close to one). Equivalently, the error of the identified system dynamics can be off by as much as 90 degrees before instability is induced. This is similar to the results of stability analyses of the LMS-based algorithms referenced in Section B.4, indicating that the average values of the sensor error and the disturbance-correlated signal (either $x(k)$ or $\eta(k)$) must at least be of the same sign in order for the error to be reduced [Ref. 101].

The Clear Box Adaptive Basis Method is very similar to the Sine/Cosine Method in the case of a single disturbance frequency (assuming a low level of noise in the

disturbance effect signal, and the use of two basis functions, or $N = 2$). Thus the stability analysis referenced earlier could be adapted to show a phase margin for the Adaptive Basis Method that is similar to the other two controllers. While the added nonlinear effects of a changing/adapting set of controller basis functions inhibits analysis of the method for the general case, experimental assessment of the controller's stability can offer some indication of its ability to perform well in less than ideal conditions. This will be accomplished in Chapter V.

D. ALTERNATE APPROACHES

Two approaches are described here that may offer alternative ways to achieve good performance against time-varying frequencies and/or achieve greater computational efficiency. These approaches are recommended for implementation in future work.

The Adaptive Basis Method was developed to allow the Clear Box Algorithm to control rapidly varying frequencies with little or no loss of performance (over the static frequency case). This same goal can be achieved if the Sine/Cosine Method's frequency estimates are significantly more accurate. To achieve this, the most recent disturbance frequency estimates can be fit to a polynomial curve. This curve can then be extrapolated forward in time to the time steps prior to the next frequency update. This is shown in Figure II-7 using a simple third order polynomial fit to eight data points of past disturbance frequency estimates. In this hypothetical situation, the last estimate was given at time step $k = 8000$, and the next update is not expected until time step $k = 9000$. The current approach of the Sine Cosine Method (as implemented on the

UQP) holds the last estimate until the next update, in which case the estimate becomes increasingly inaccurate as time progresses. The proposed approach allows extrapolation by evaluating the polynomial for the current time step at each time step until the next update is available. At that time the polynomial is re-fit to the available data, and the process repeated until the next update. The resulting improvement in frequency estimation should allow better performance in the case of rapidly varying frequencies.

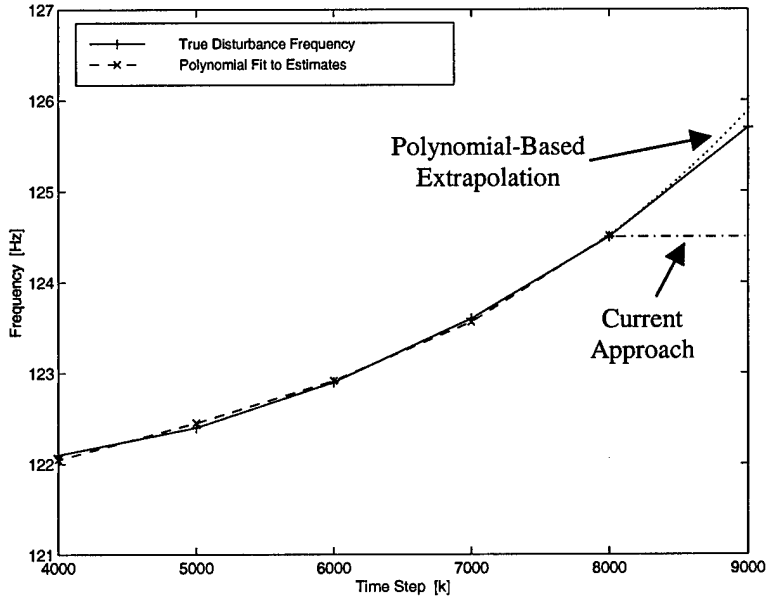

Figure II-7: Improving the Sine/Cosine Method's Frequency Estimates Using a Polynomial Curve Fit

The second proposed approach involves forming a hybrid controller that uses the disturbance effect signal $\eta(k)$ from the Clear Box Algorithm to act as the

reference signal $x(k)$ for the Multiple Error LMS Algorithm. The block diagram for such a controller is shown in Figure II-8.

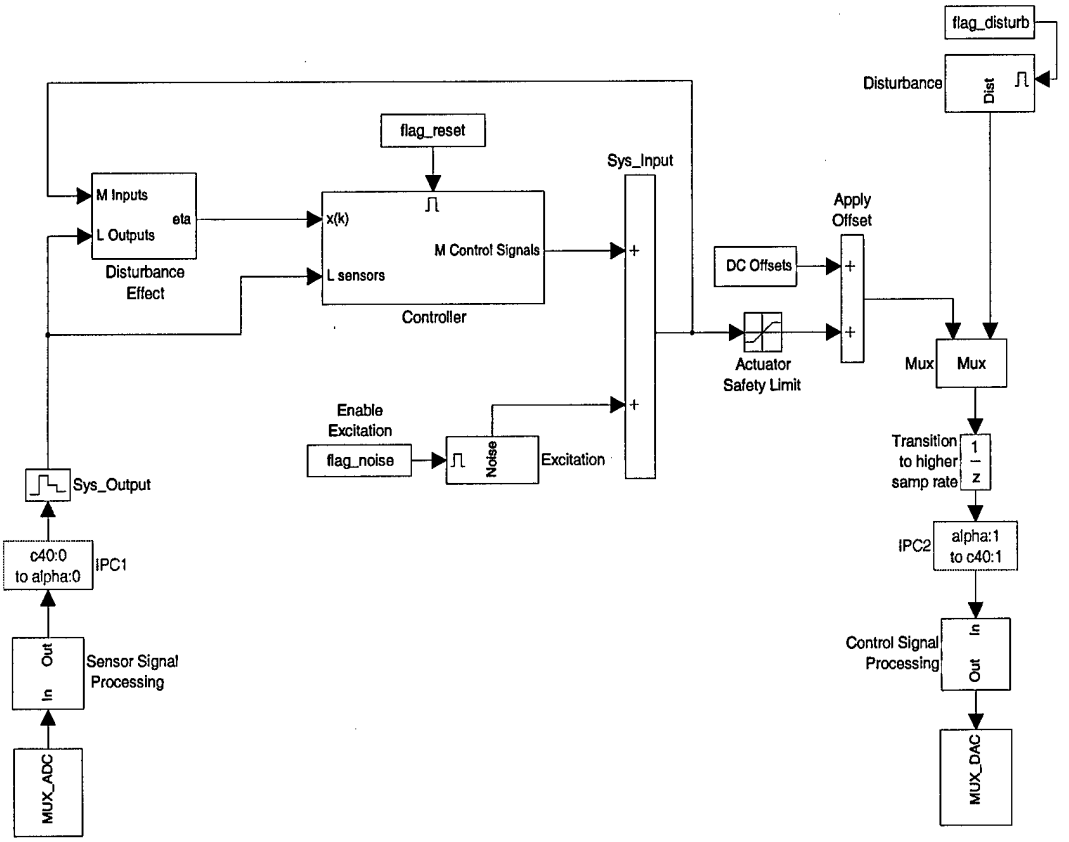

Figure II-8: Hybrid Controller Block Diagram

The advantages of this approach are that the identification provided by the Clear Box component allows control of time-varying systems since the Clear Box system model can be translated into the FIR filter form needed by the LMS algorithm. Control of unanticipated disturbances and harmonics is possible since this information is

available in the disturbance effect signal, without the need for the separate sensor to provide the disturbance correlated signal. At the same time the processing requirements are significantly less than that of the Clear Box Algorithm, although slightly more than the Multiple Error LMS Algorithm since the disturbance effect must be calculated at each time step. Also, the selective control capability is available through filtering of the $\eta(k)$ signal.

A disadvantage of the hybrid method is that the order of the FIR model required to control lightly damped systems may still be impractically large, and thus the required increase in the order of the model would negate some of the processing gains associated with this approach. Also, although the selective disturbance control option is available, the decision regarding which disturbances to control is slightly hindered by the fact that the magnitude of the control signal needed to control each disturbance is unknown (since sine & cosine coefficients are not used). The decision would have to be made based on noted transmission zeros in the system model, and the relative significance of each disturbance frequency to the total system output (similar to the approach used with the Adaptive Basis Method).

E. SUMMARY

The wide acceptance of the FXLMS (and extension to the MIMO Multiple Error LMS) algorithm for active control of sound and vibration is based on its ease of implementation and adequate performance in many applications. However, the Clear Box control formulation approaches the problem from a system identification

perspective, allowing greater insight into the physical processes, and selectivity in controlling disturbances.

The next chapter discusses the experimental setup, and Chapter IV discusses the results of the system identification experiments. Both the Multiple Error LMS Algorithm and Clear Box Algorithm (including both methods discussed above) are implemented on the UQP in experiments conducted in Chapter V, and the observed advantages and disadvantages are discussed in Chapter VI.

III. EXPERIMENT SETUP

A. UQP

The "Ultra Quiet Platform", or UQP, on which the experiments were performed was built by CSA Engineering of Palo Alto, CA, and is configured similarly to a six degree of freedom "cubic" Stewart Platform. In such a system, the struts are arranged as if they were on the edges of a cube, thus providing for three orthogonal pairs of actuators. The advantage of such an arrangement is that control in six degrees of freedom is possible using all linear actuators, and actuator coupling is minimized. [ref. 102]

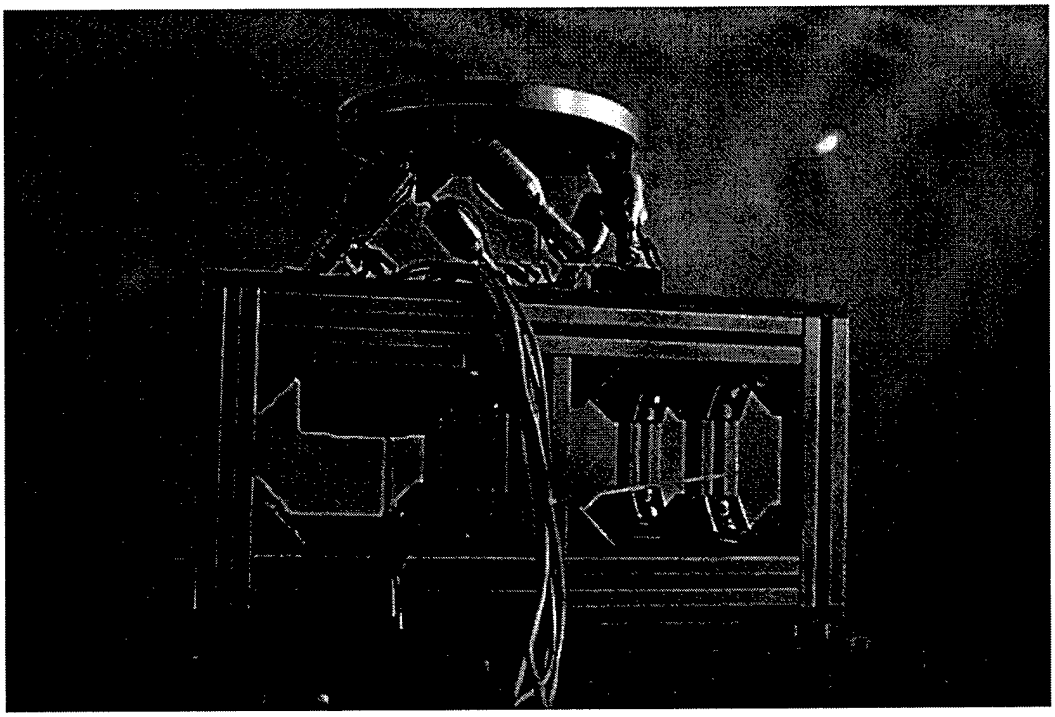

Figure III-1: UQP and Satellite Bus Mockup

The platform is mounted on a spacecraft bus mockup, to which is mounted an Aura bass shaker which serves as the disturbance source. The entire experiment sits on sixteen rubber feet attached to a 3800 lb. Newport RS4000 isolation table. The table is mounted on four Newport I-2000 series Laminar Flow Isolator pneumatic pedestals which help to further isolate the experiment from floor vibrations.

1. Smart Struts

The six struts that support the top plate each consist of an actuator, sensor, and a passive isolation stage (shown in Figure III-2). The struts are mounted on each end using flexible mounts to minimize the transmission of bending moments.

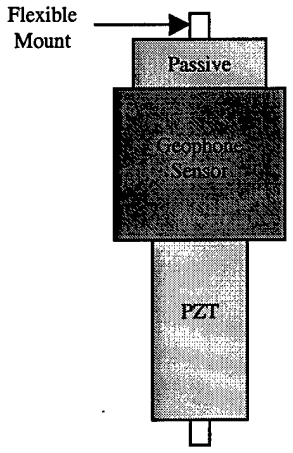

Figure III-2: Smart Strut Configuration

a) Actuators

The piezoceramic stack actuators (PZTs) in each strut convert control signal voltages to physical movement of the strut. The maximum displacement of the actuators is 50 μm, which is enough for vibration isolation applications, but not for platform pointing/steering. Although piezoceramics are considered "hard" actuators, the intended application is narrowband disturbance rejection and hard or soft actuators work equally well.

b) Sensors

The Geospace GS-11D geophone sensors consist of wire coils supported by soft springs under the influence of a magnetic field, and the sensors provide a signal proportional to velocity. The GS-11D model has a natural frequency of 14 Hz (double pole with damping factor ζ=0.8), above which the sensitivity is fairly constant. Below 14 Hz the response decays rapidly.

The geophone sensors were selected for this experiment because of their low cost and ruggedness. An aerospace quality accelerometer might be a more logical choice for an actual spacecraft application, and would extend the useable control bandwidth to lower frequencies.

c) Passive Isolation

A degree of passive isolation is achieved through use of a flexure with damping material. This passive isolation stage is in series with the active stage, and the resulting six platform suspension modes have frequencies between 25-80 Hz. An advantage of this series configuration is that if a flexible structure/payload is mounted on the top of the platform, the passive stage tends to minimize any control/structure interaction. Additionally, if a strut experiences on-orbit failure there exists some degree of isolation from the spacecraft bus.

2. Disturbance Source

The source of disturbances for the disturbance rejection experiments is an Aura bass shaker (model AST-1B-4, 25 W, 4 ohm). The shaker is mounted to the underside of the spacecraft bus' top plate (Figure III-3), and is driven by sinusoidal signals generated by the digital signal processor (DSP). The disturbance generator code can be modified to output a variety of disturbance profiles, from single static frequencies to multiple disturbance frequencies that have time-varying amplitudes and frequencies.

B. SUPPORT ELECTRONICS

1. Hardware Interface

The UQP experiment requires power amplification for the actuators and signal conditioning for the geophone sensors. These are provided via a PCB Piezotronics

790A06 6-channel power amplifier (± 200 V, ± 100 mA), and a CSA Engineering Active Vibration Control System (AVCS) signal conditioning unit supplied with the UQP. The disturbance generator is powered by a Kepco BOP 20-10M amplifier. Figure III-3 shows the configuration of the experiment.

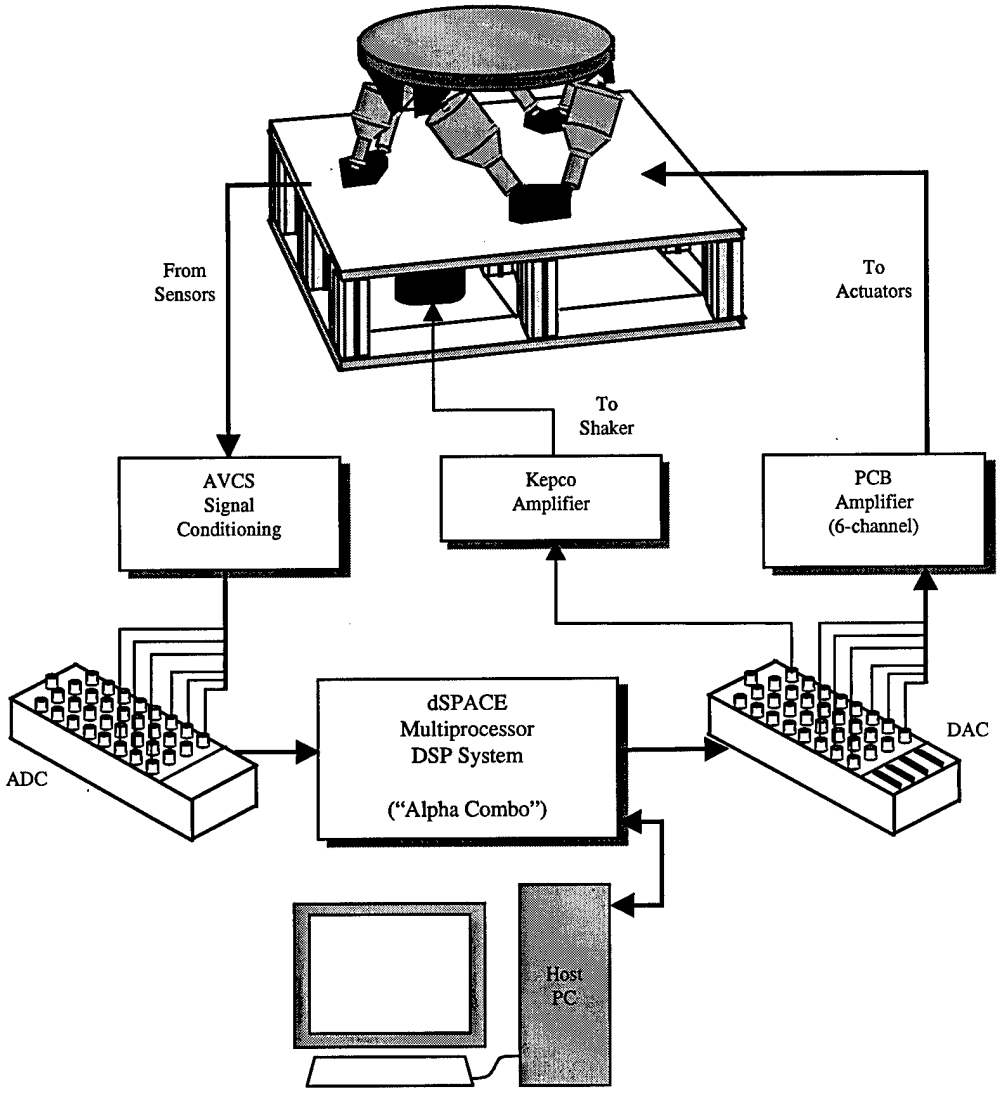

Figure III-3: Experiment Overview

2. Digital Signal Processor

The control function is performed by the combination of a dSPACE DSP system and a host PC (see Figure III-4). The dSPACE system includes an "alpha combo" multiprocessor system consisting of a Texas Instruments C40 50 MHz processor with 512KB of memory, and a Digital Equipment Corporation Alpha 500 MHz processor with 2MB of memory. The C40 performs all of the input/output functions such as interfacing with the analog to digital converter (ADC) and digital to analog converter (DAC) boards, and data transfer to and from the host PC.

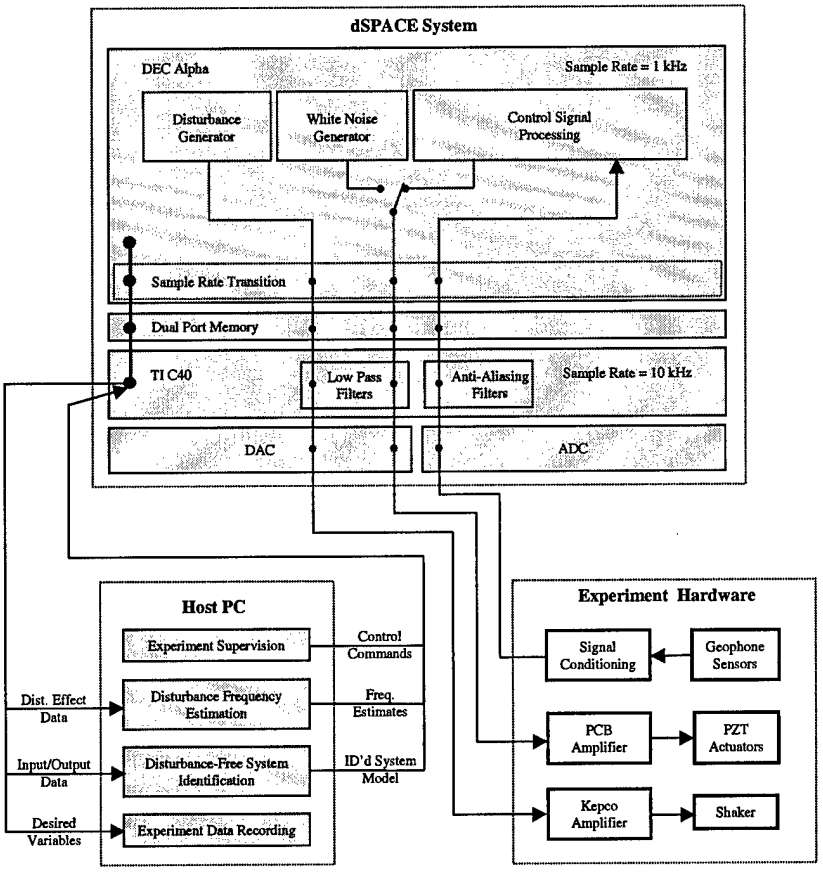

Figure III-4: Control Processing

The ADC is a DS2003 32 channel unit with the resolution set at 12 bits. The DAC is a DS2103 (also 32 channels), with the resolution fixed at 14 bits. On both units the input/output voltage range can be selected as either ±5 volts, or ±10 volts. To prevent aliasing, the ADC samples data at a rate of 10 kHz and then digital anti-aliasing filters (3^{rd} order Chebyshev) are employed on the C40 board with a corner frequency of 200 Hz. Analog filtering before the ADC is unnecessary due to the low system response above 5 kHz (see further discussion in Chapter IV). This arrangement provides 5x oversampling of the highest frequencies passed (unattenuated) by the filters. Low-pass filters are again employed on the output to smooth the "stair-step" quality of the control signal. This prevents any excessive excitation of the actuators at the 1 kHz sample rate of the control signal calculation.

3. Host Personal Computer (PC)

The host PC (A Dell Optiplex GX1, 450MHz) serves several functions. Coding for the control algorithms is performed in the Matlab/Simulink environment using C-coded "S-Functions" to perform the more specialized tasks. dSPACE software on the host PC allows automatic DSP code generation and downloading when working in the Matlab environment. While the controller is running on the DSP, the host PC performs supervisory functions and also interfaces with the user.

Before the start of each experiment the user is given the option to perform a complete system identification (see Chapter IV). Once the identification is complete the host PC downloads the updated system model to the DSP. While the controller is

running on the DSP the PC also performs periodic estimates of the disturbance frequencies (when using the Clear Box Sine/Cosine Method), and downloads these updated estimates. Experiment data is also captured by the PC using the dSPACE MTrace utility. Multiple captures can be performed simultaneously allowing data to be collected for disturbance frequency updates at the same time as different variables are captured for post-experiment analysis.

C. SOFTWARE

Many different software tools were used during the process of coding, monitoring, and analysis (summarized in Table III-1). The dSPACE software was designed to work with Matlab and Simulink, and allows rapid transition from a "block diagram" representation of the control algorithm to real-time code running on the DSP. Except for compilation of the C-coded S-Functions, all code generation and downloading is handled by the dSPACE RTI software (working in combination with the Matlab Real Time Workshop).

The MLIB and MTrace utilities from dSPACE allows the host PC's experiment supervisor (Matlab code) to acquire, process, store, and download data while the DSP was running. This allows a high degree of automation to be built into the experiment.

The Simulink block diagram is used to divide functions between the C40 and Alpha processors by separating the tasks with "Inter-Processor Communication" (IPC) blocks. More details are provided in Chapters IV and V.

Software Tools	Version	Function
Matlab	5.2.1.1420	
Real-Time Workshop	2.2.1	Real-time code generation
Simulink	2.2.1	Algorithm development
dSPACE RTI 1003	3.2	Real-time interface to Simulink
dSPACE RTI-MP	3.2	Real-time interface for multiprocessor systems
MTrace	3.1	Data acquisition from DSP
MLIB	3.1	Data downloading to DSP
Compilers		
AXP-GCC GNU	2.7.2	C compiler for DEC Alpha
TMS-320	4.70	Compiler for TI C40
Microsoft Visual C++	5.0	Compiler for S-Functions

Table III-1 : Software Versions and Functions

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SYSTEM IDENTIFICATION EXPERIMENTS

A. EARLY IDENTIFICATION WORK

System identification experiments performed in [Ref. 103] on the UQP at the Naval Postgraduate School resulted in six SISO models from the input of each actuator's amplifier to the output of the corresponding strut's sensor. An example of the frequency response obtained from this work (see Figure IV-1) shows that there is only one lightly damped mode which dominates the response at 1.4 kHz, and a smaller one just above 1 kHz. The remainder of the response appears well damped. The response lacks some fidelity below 10 Hz due to 1) sensor noise , 2) limited data record length, and 3) limited model order. The data record length is limited since a relatively high sample rate (5 kHz) was needed to capture the 1.4 kHz mode, and the available memory was finite.

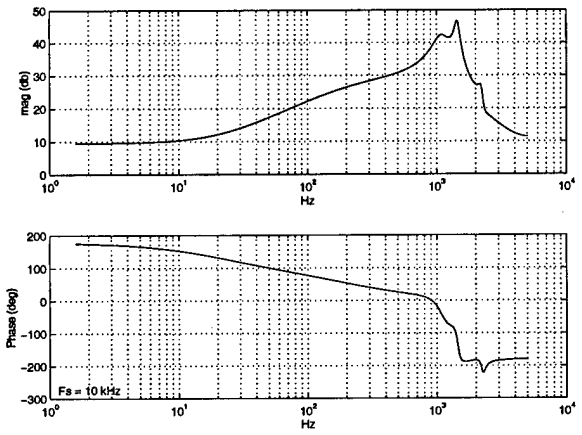

Figure IV-1: Frequency Response, Strut #1 Early Model

B. SAMPLE RATE SELECTION

The disturbance frequencies of interest for the UQP application are approximately 20-300 Hz, and the system model must be accurate in that frequency range. To improve the fidelity of the model at low frequencies the sample rate is reduced, allowing longer time-duration records. Another contributing factor is that the real time control calculations are relatively complex, given that a 6-input, 6-output MIMO controller is used. These factors led to a decision to reduce the sampling rate from 5 kHz (used in the previous identification work) to 1 kHz.

C. ANTI-ALIASING FILTERS

At sampling frequencies below ~3 kHz aliasing of the 1.4 kHz mode occurs, and thus it is clear that anti-aliasing filters are needed. As discussed in Chapter III, and shown in Figure III-4, the use of a 10 kHz sample rate for the A/D conversion allows accurate capturing of all frequencies below 5 kHz. This allows digital anti-aliasing filters to be used before the sample rate is reduced to 1 kHz. The main advantage of using digital filters is the ability to rapidly reconfigure the filter to any desired type, corner frequency, and order. The disadvantage is that it requires processing, and for a spacecraft application where processing power is scarce it may be advantageous to use analog filters (weight and reliability must also be considered).

For this project we have chosen 3rd order Chebyshev filters with 1 dB of passband ripple and a 200 Hz corner frequency. This configuration gives sufficient attenuation of the system response above the Nyquist frequency of 500 Hz.

D. SYSTEM ID DATA COLLECTION

To facilitate the acquisition of system identification data, a program was coded for the DSP (controlled by the Host PC) that first turns on a white noise source (input to all six struts), and then records input-output data as long as possible according to the available memory. Recording twelve channels of data at a rate of 1 kHz, the longest allowable data record is 11 seconds.

Figure IV-2 shows the Simulink block diagram which represents the source for the DSP code. The white noise source is simply a six-channel random number generator with normal distribution and zero mean. Also present are a disturbance generator which can generate up to five sinusoidal disturbances. These disturbances can be constant or time-varying in frequency and amplitude, depending upon user responses to prompts from the Host PC. An impulse generator function allows the unit pulse response to be obtained for each strut, enabling comparison with the identified model's pulse response.

All of these functions are located in "enabled subsystem" Simulink blocks which allow them to be turned on and off based on commands sent from the Host PC. The Host PC code also starts and stops all data recording, and returns the data to the workspace for subsequent system identification calculations.

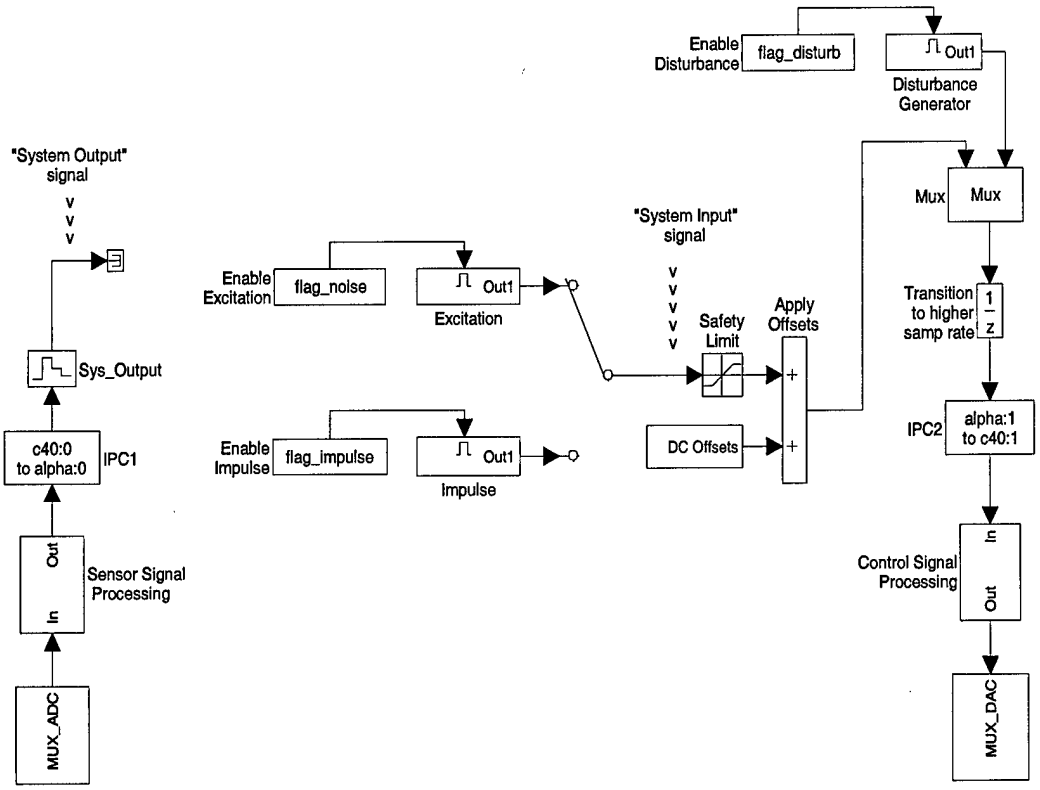

Figure IV-2: System Identification Simulink Diagram

Note in Figure IV-2 that a DC offset is applied to the actuator input signal. This is needed to keep the drive signal in the middle of the actuators' operating range of 0-100 Volts, thus an offset of 50 V is applied. The "IPC" blocks are for InterProcessor Communication between the Texas Instruments C40 board (the elements below the IPC blocks) and the Digital Equipment Corp. Alpha board (the elements above the IPC blocks). The C40 is operating at 10 kHz, and the Alpha is operating at 1 kHz, which

requires a sampling rate transition when going from one to the other. The zero-order hold and unit delay blocks provide this transition. The system model identified from the input-output data takes into account the added delay due to the zero-order hold and unit delay transition blocks.

The "sensor signal processing" is the anti-alias filtering discussed earlier. Also of note is the "control signal processing" that occurs prior to D/A conversion. In the case where there is a sinusoidal input to the actuator, the effect of the sampling rate transition is a sinusoid with stair steps occurring once every sampling period at the lower rate. Without filtering, this stair step effect excites the actuators at the slow sampling frequency (1 kHz) which degrades performance. The solution is to employ digital low pass filtering in a manner similar to that used for the anti-aliasing filters. After some experimentation it was determined that good "smoothing" was obtained using a 4^{th} order Chebyshev lowpass filter at 400 Hz. The amount of delay caused by the filtering is frequency dependent, but it is on the order of one half of the lower sampling period (1/2*0.001 sec), as can be seen in Figure IV-3. The disturbance signal is also filtered before it goes to the shaker (2^{nd} order Chebyshev lowpass at 300 Hz) to prevent the unwanted 1 kHz disturbance from exciting the platform.

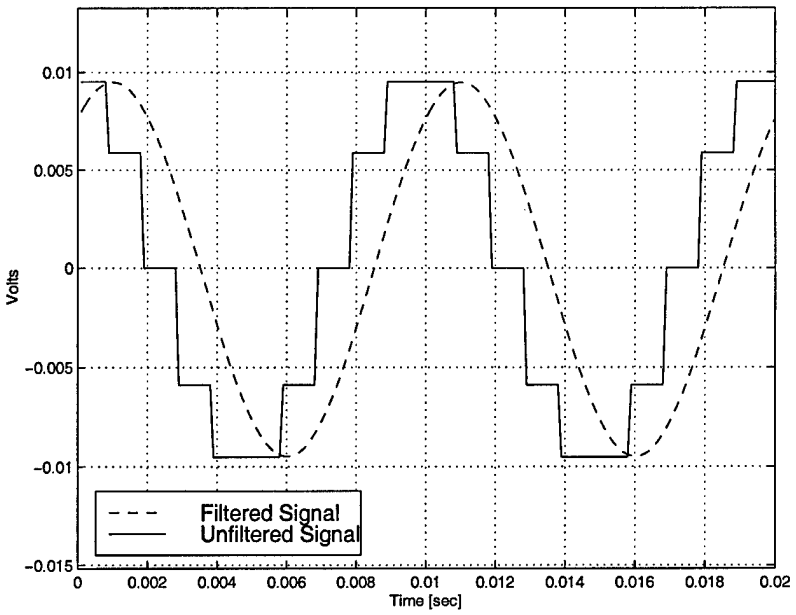

Figure IV-3: Effect of Low Pass Filtering on Control Signals

E. OKID REFERENCE MODEL

To obtain an initial MIMO model of the UQP system dynamics the Observer/Kalman Filter Identification (OKID) software is used to generate a state space model. This model also serves to verify the identification results obtained from the Clear Box algorithm in the next section. There is extensive literature available on OKID [Ref.s 104,105,106]. Suffice it to say that the technique serves to obtain an accurate estimate of the system's state space model from a time history of input-output data. For this reason it is referred to as a "time-domain" approach.

The desired model order is determined by the selection of p, the equivalent order of the model for each input-output pair. For the reference model, p has been selected as

40 which (for $q = 6$) results in a MIMO state space model with an A matrix of dimension 240×240. Shown below in Figure IV-4 is the frequency response at the sensor outputs of all six struts for an input to the strut #1 actuator (the frequency response is very similar for inputs to the other struts). The uppermost magnitude plot is for the output at the strut #1 sensor. This is to be expected since it is most directly coupled to the strut #1 actuator. The effect of the anti-aliasing filters at 200 Hz can be seen by the sharp reduction of the response above this frequency.

The remaining five plots in Figure IV-4 show the coupling that exists between the strut #1 actuator and the sensors on the other five struts. In general the coupling is strongest with the neighboring struts on either side, and in the frequency range of 30-150 Hz. The realization that this coupling is strongest within the intended control bandwidth led to a decision to implement a full MIMO controller instead of six SISO controllers.

A notable system characteristic is a steadily decreasing response below the Geophone sensor's natural frequency of 14 Hz. This puts a lower limit on the useful control bandwidth of the UQP. Also notable are several damped modes from 30-100 Hz, which are suspension modes due to the passive isolation stage on each strut. These modes may be altered in a zero-g environment, and in a spacecraft application the system identification would need to be re-accomplished upon reaching orbit.

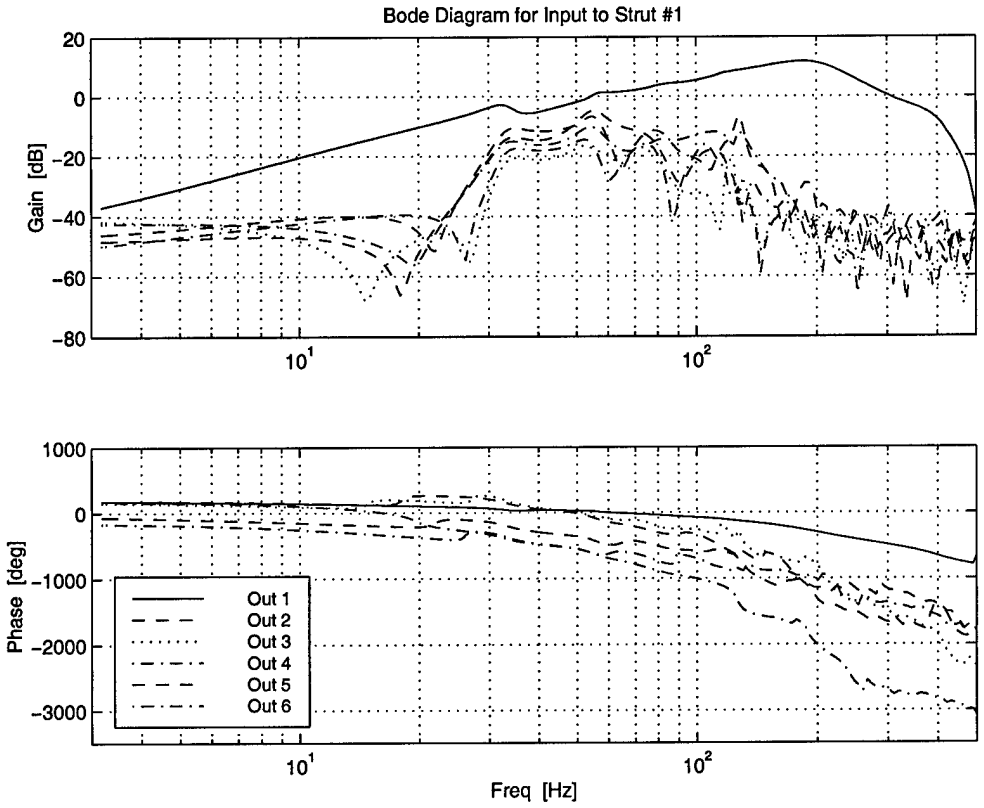

Figure IV-4: OKID Reference Model Frequency Response

F. CHECKING THE ACCURACY OF THE REFERENCE MODEL

Two methods are used to verify that the OKID model obtained above is an accurate representation of the UQP system's dynamics. First, the true pulse response of the system is obtained and compared against that of the model. Second, a new set of input-output data is obtained, and the actual system output is compared to that of the model (using the same input data).

1. Pulse Response

Using the pulse response capabilities of the Simulink code shown in Figure IV-2, the true "UQP pulse response" is obtained by inputting an impulse to one strut at a time. Using the OKID state space reference model obtained above, the corresponding "model pulse response" is generated. These responses (36 input-output pairs) match closely, as can be seen by the examples shown in Figure IV-5 and Figure IV-6 below.

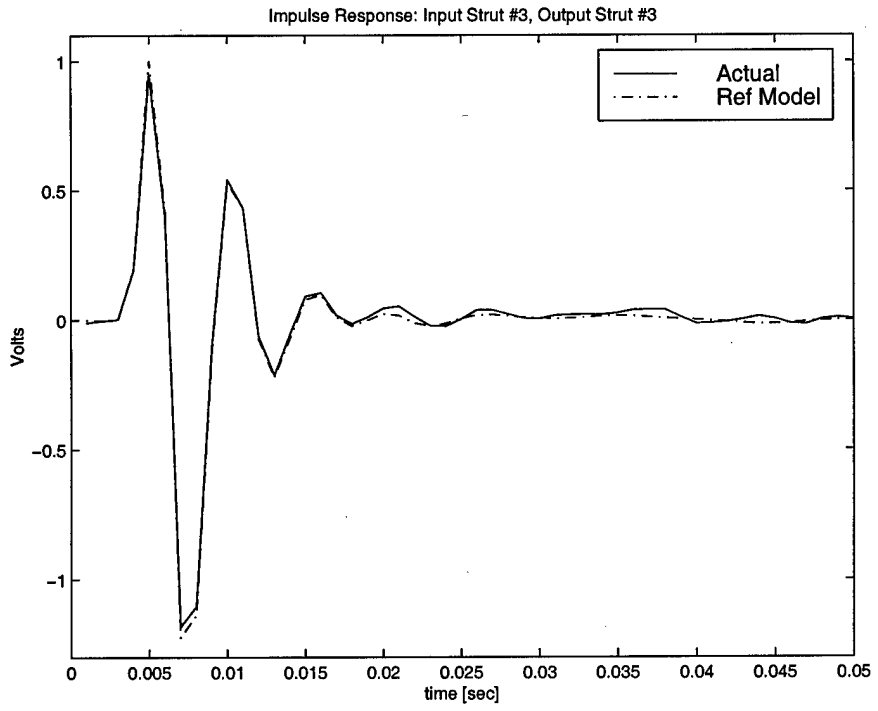

Figure IV-5: Impulse Response of Model vs. Actual (#1)

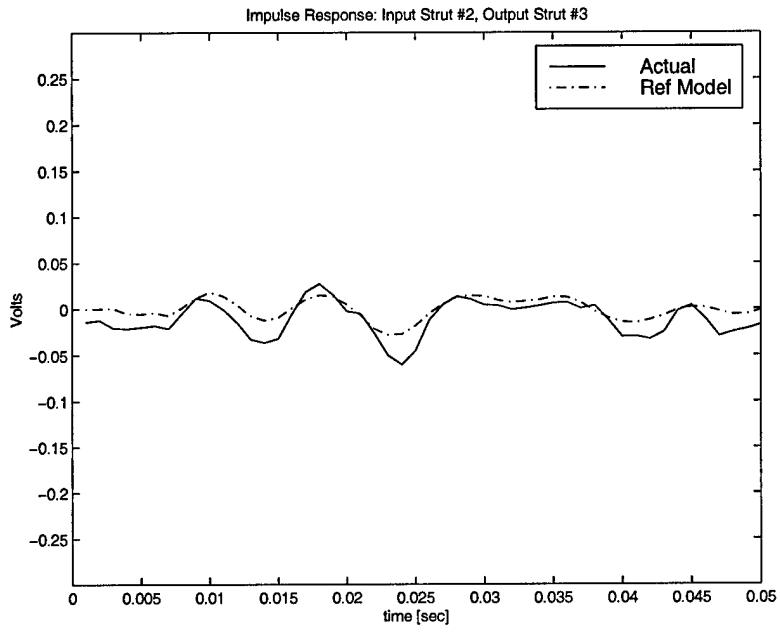

Figure IV-6: Impulse Response of Model vs. Actual (#2)

2. Response to Random Input Data

Another test of the reference model's accuracy is accomplished through use of a set of "verification data", which is different from the data used to generate the identified model. A second set of UQP response data is obtained using random inputs, and the input data is then applied to the OKID reference model. If the model is accurate, the resulting simulated output should match that of the UQP hardware. The samples shown below in Figure IV-7 show that the outputs of the model match those of the UQP very well. After the first 0.02 seconds (the system's settling time), the two plots are almost indistinguishable. At this point it can be stated that the OKID model is of sufficient accuracy to serve as a reference against which other identification results can be compared.

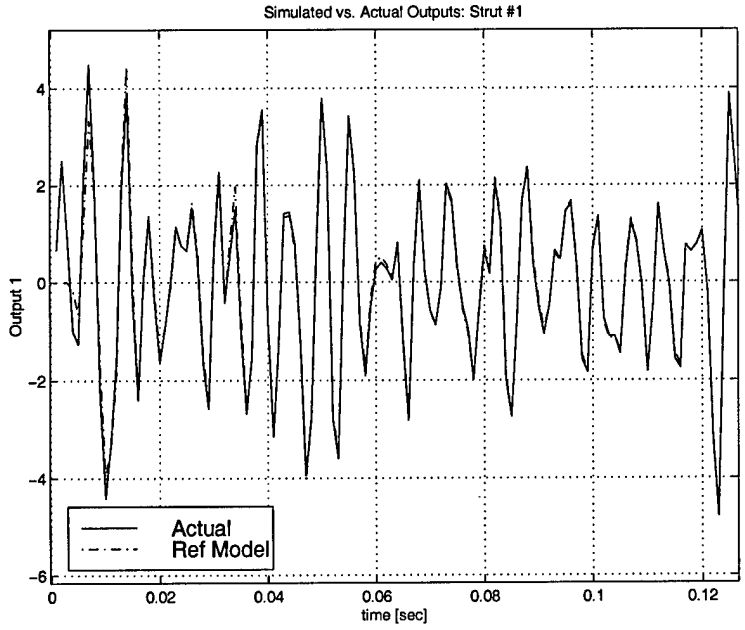

Figure IV-7: Actual vs. Simulated Response to Random Inputs

G. DISTURBANCE SOURCE FREQUENCY RESPONSE

Using the same technique as in Section E above, the response of the six strut sensors to inputs to the Aura bass shaker (the disturbance source) is obtained. None of the control algorithms implemented require a model of this "primary plant", but it is useful for determining which frequencies cause the largest system response.

A set of input-output data is obtained for the shaker-to-sensor system, and the resulting OKID state space model is used to obtain the frequency response in Figure IV-8 below.

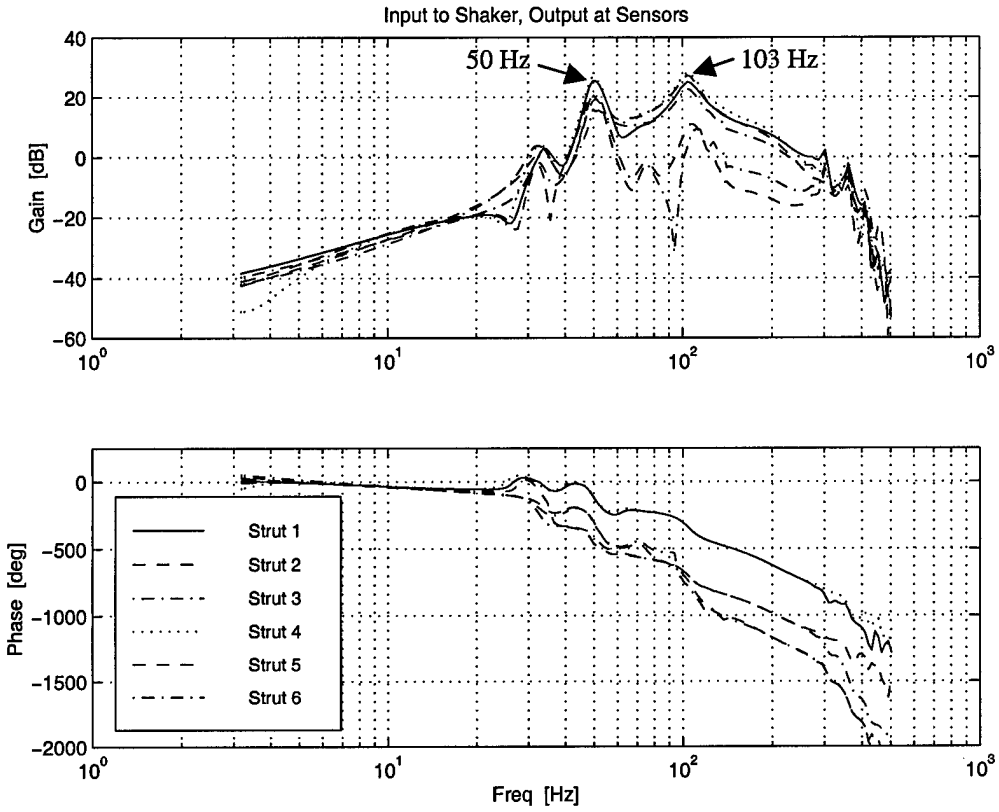

Figure IV-8: Shaker Frequency Response

From this response it is clear that the sensors are most sensitive to disturbance frequencies at 50 and 103 Hz. In general, struts 1, 4, 5, and 6 are more sensitive to disturbances due to their proximity to the shaker. Struts 2 and 3 are on the opposite side of the platform from the shaker, which explains their lower sensitivity. There is a significant dip in the response of strut #3 to a 90 Hz disturbance.

H. CLEAR BOX MODEL

As outlined in Section II.C.5 on page 32, the Clear Box algorithm takes the approach of first identifying the system model (even in the presence of periodic disturbances and sensor noise). The result is a "disturbance free" model which can be expressed in any chosen form (state space, ARX, etc.).

As a test of the algorithm's ability to correctly identify the UQP system dynamics model in the presence of disturbances, a new set of input-output data is obtained while the shaker (disturbance source) is driven by a signal consisting of two sinusoids at 70 and 103 Hz. These frequencies were chosen to show that the algorithm will detect with equal accuracy disturbance frequencies to which the system is least and most sensitive (see Figure IV-8). The data is then used to construct a disturbance-corrupted model, the frequency response of which is shown in Figure IV-9. Note the effects of the two disturbance modes at 70 and 103 Hz.

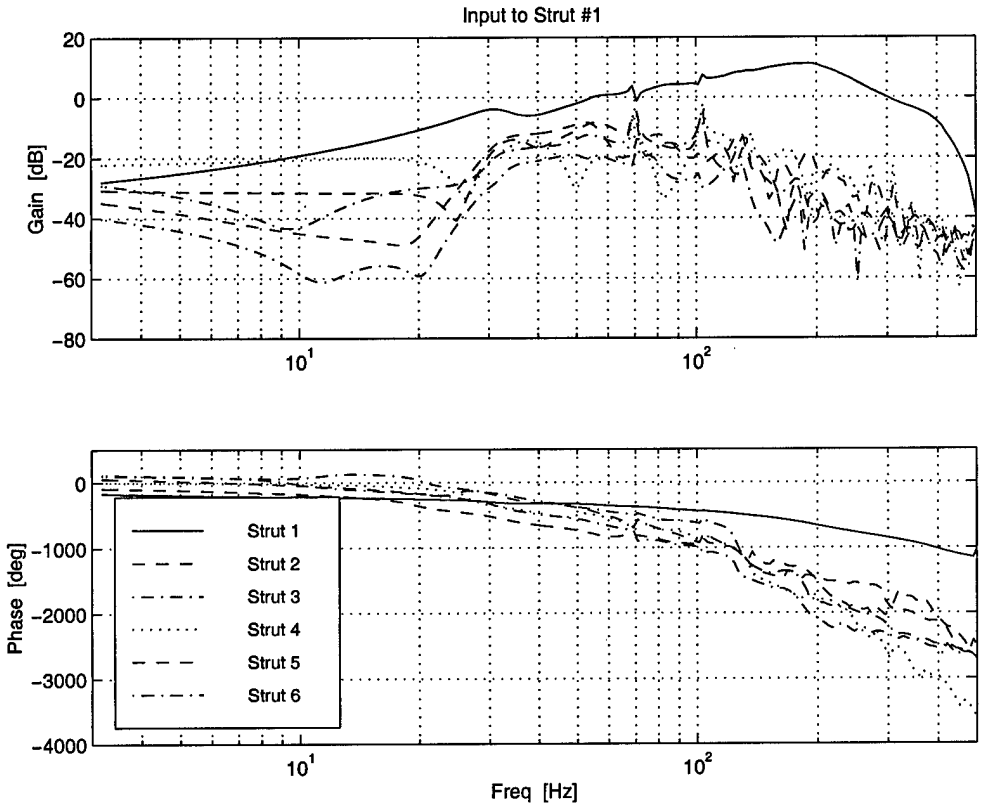

Figure IV-9: Clear Box System ID, Disturbance-Corrupted

The disturbance-corrupted model is then converted to modal form so that each mode can be analyzed. The top portion of Figure IV-10 shows the frequencies of the identified modes and their associated damping ratios. The disturbance modes are identified by the fact that they are below the damping threshold, which is set by the user. The bottom portion of Figure IV-10 shows the relative contribution of each mode to the total system pulse response. The modes with a greater "modal pulse response norm" are the true system modes, and the remainder are the noise and disturbance modes.

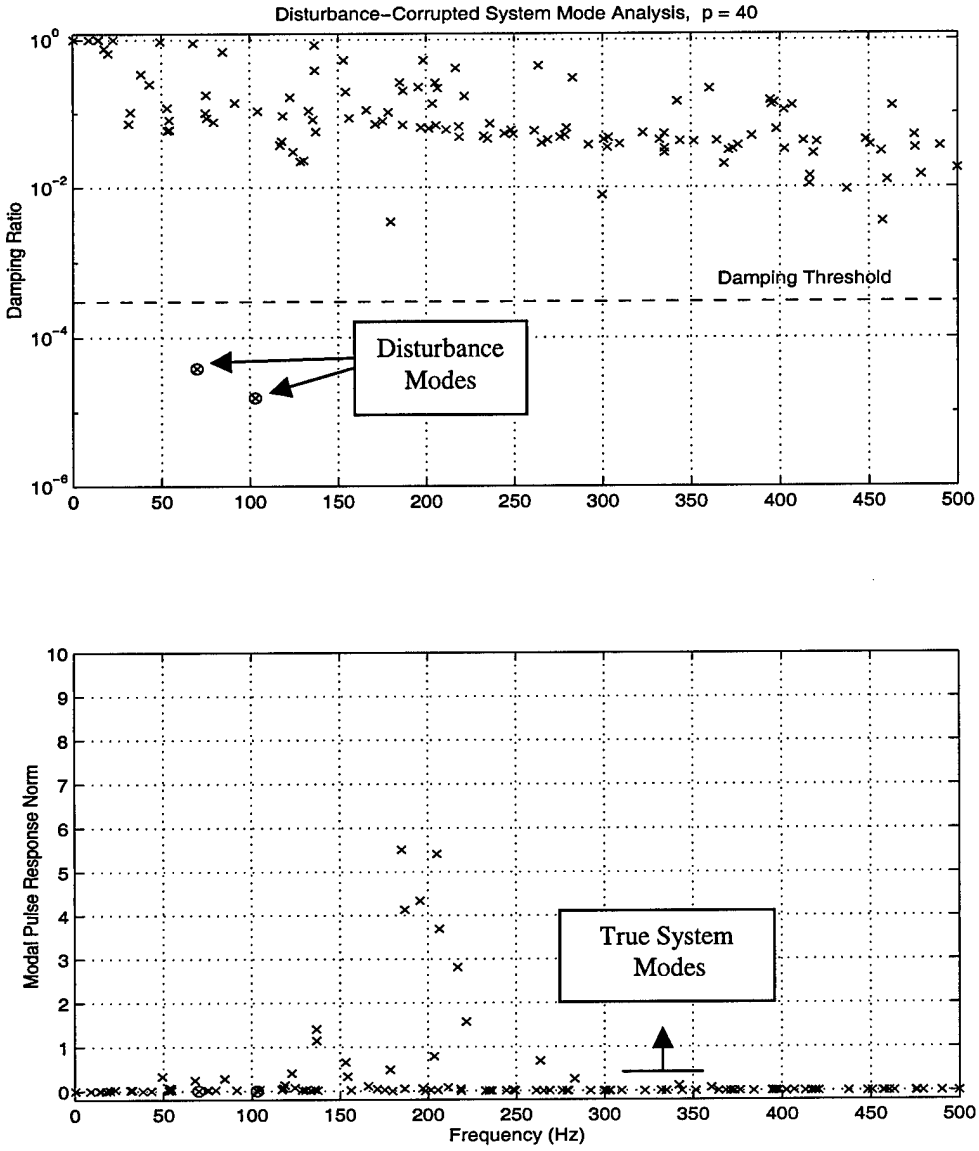

Figure IV-10: Clear Box System ID, Analysis of Modes

As discussed in Section II.C.5.a) (starts on page 33), there are three types of modes present; true system modes, disturbance modes, and noise modes. The true system modes are determined by their large contribution to the system's pulse response. In this

case the anti-aliasing filters used at 200 Hz tend to dominate the response due to the highly damped nature of the UQP in this low frequency region.

After identifying the disturbance modes, they are removed from the system model. The resulting model has the frequency response shown in Figure IV-11, which matches very closely with that of the OKID reference model shown earlier in Figure IV-4. The noise modes can be removed if desired, although retaining them has the effect of filtering the noise associated with the system response [Ref. 107].

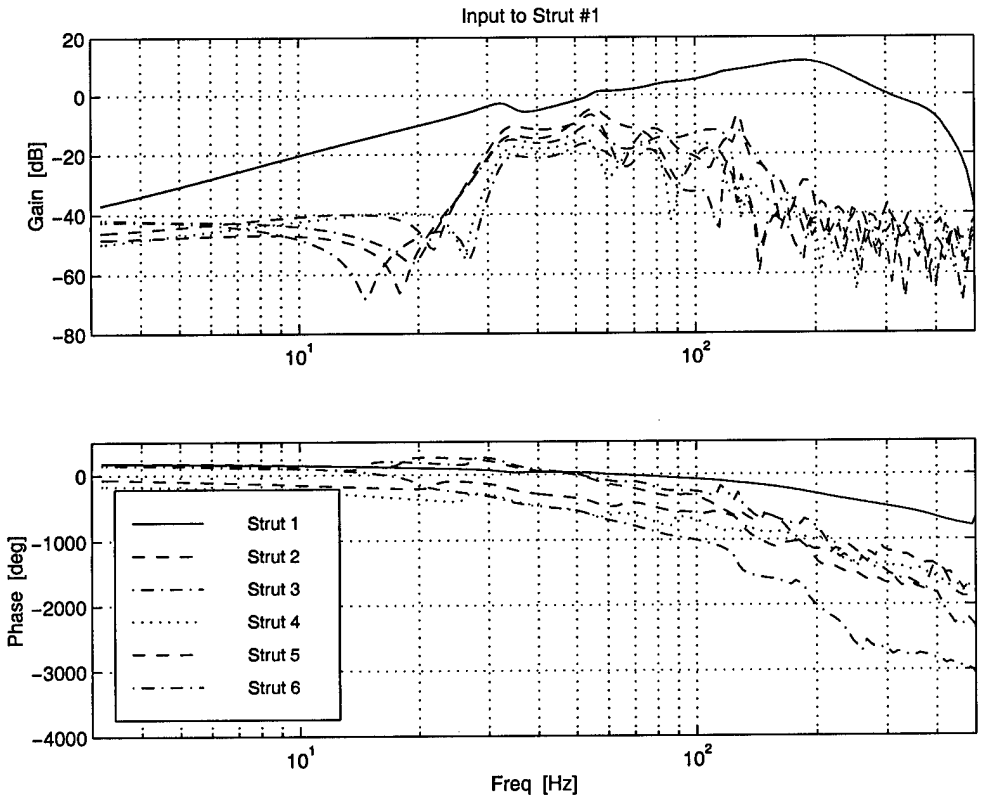

Figure IV-11: Clear Box System ID, Disturbance-Free

I. CONVERSION TO ARX MODEL

Now that the disturbance-free state space model has been determined (through elimination of the disturbance modes) the methods of Section II.C.5.c) on page 38 are used to convert the model to ARX form for implementation on the digital signal processor. The model has the form

$$\begin{aligned} y(k) = {} & \bar{\alpha}_1 y(k-1) + \bar{\alpha}_2 y(k-2) + \ldots + \bar{\alpha}_p y(k-p) + \\ & \bar{\beta}_1 u(k-1) + \bar{\beta}_2 u(k-2) + \ldots + \bar{\beta}_p u(k-p) + \eta(k) \end{aligned} \tag{4-1}$$

where, again, the $\bar{\alpha}$ and $\bar{\beta}$ coefficients are the disturbance-free system model, and $\eta(k)$ is the disturbance effect. This form of the model is used for the Clear Box algorithm, which is implemented in the next Chapter. The disturbance effect, which is calculated by rearranging Eq. (4-1) above such that

$$\begin{aligned} \eta(k) = {} & y(k) - \bar{\alpha}_1 y(k-1) - \bar{\alpha}_2 y(k-2) - \ldots - \bar{\alpha}_p y(k-p) \\ & - \bar{\beta}_1 u(k-1) - \bar{\beta}_2 u(k-2) - \ldots - \bar{\beta}_p u(k-p) \end{aligned} \tag{4-2}$$

plays a key role in both the "Sine/Cosine Method" and the "Adaptive Basis Method" of the Clear Box algorithm (discussed in Section II.C.6 starting on page 42). It contains all of the information needed to cancel any disturbance frequency that is not retained in the system model coefficients.

J. CONVERSION TO FIR FILTER MODEL

Recall from Section II.B.3 (page 20) that the Multiple Error LMS algorithm requires a Model in the Form of a Finite Impulse Response (FIR) filter. Such a model is obtained directly from the disturbance-free state space model obtained earlier in this Chapter, using either the OKID reference model or the Clear Box model. In practice the method used to determine the disturbance-free model depends on whether or not the disturbances can be turned off while data is taken. If the disturbances are always present, then the Clear Box model is the only alternative which achieves accurate results.

To obtain an FIR model using the disturbance-free state space model, we simply calculate the system Markov parameters (pulse response coefficients) using the $\overline{A}, \overline{B}, \overline{C}, \overline{D}$ matrices. Recall that if there are M actuators and L sensors, then

$$\hat{y}(k) = C_{LM0}u(k) + C_{LM1}u(k-1) + \cdots + C_{LM(J-1)}u\big(k-(J-1)\big) \tag{4-3}$$

where

$$\begin{aligned} C_{LM0} &= \overline{D} \\ C_{LMj} &= \overline{C}\overline{A}^{j-1}\overline{B}, \qquad j = 1, 2, \ldots, J-1 \end{aligned} \tag{4-4}$$

and also

$$C_{LMj} = \begin{bmatrix} c_{11j} & c_{12j} & \cdots & c_{1Mj} \\ c_{21j} & c_{22j} & \cdots & c_{2Mj} \\ \vdots & \vdots & \ddots & \vdots \\ c_{L1j} & c_{L2j} & \cdots & c_{LMj} \end{bmatrix}, \qquad j = 0, 2, \ldots, J-1 \tag{4-5}$$

In the case of the Multiple Error LMS algorithm, the FIR model

$$C = \left[C_{LM0},\ C_{LM1},\ \ldots,\ C_{LM(J-1)} \right] \tag{4-6}$$

is used to filter the disturbance-correlated reference signal $x(k)$ to give the "filtered-x" signal $r(k)$ where

$$r_{lm}(k) = \sum_{j=0}^{J-1} c_{lmj} x(k-j) \tag{4-7}$$

This signal, along with the sensor error $\varepsilon(k)$ is used in the LMS algorithm to adapt the feedforward controller's filter coefficients $w(k)$ (see Eq. (2-22)).

K. SUMMARY

In this chapter sets of input-output data were used to build a model of the UQP system dynamics. In this case the "system" includes all subsystems such as the actuators, struts, sensors, filters, rate transition delays, and any dynamics associated with D/A and A/D conversion. To obtain a reference model, band-limited white noise inputs were used, and outputs obtained in the absence of any user-generated, or other disturbances (there are always some disturbances associated with room and sensor noise). The resulting OKID model was found to have a pulse response that matched that of the actual

system, and a set of verification data confirmed that the model and the actual system react very similarly to random inputs.

The Clear Box algorithm's system identification routine was used, in the presence of sinusoidal disturbances, to generate a disturbance-corrupted model which was converted to a disturbance-free model using disturbance mode elimination. The model was then converted to ARX form for implementation on the DSP.

Finally, it was shown how the model can be transformed into a Finite Impulse Response filter, which is the form of the model needed for implementation of the Multiple Error LMS algorithm. The next chapter will demonstrate how these models are used in disturbance rejection experiments on the UQP.

V. DISTURBANCE REJECTION EXPERIMENTS

A. INTRODUCTION

The experiments performed in this chapter were designed to reveal the strengths and weaknesses of the three controllers investigated in this research. The Multiple Error LMS algorithm, the recently developed Clear Box algorithm's "Sine/Cosine Method", and the Clear Box "Adaptive Basis Method" (developed during the course of this research) are all demonstrated in a variety of disturbance environments.

The initial experiments are with static disturbances that do not change their amplitude or frequency (a variety of single and multiple disturbance cases are explored). The next set of experiments show how the controllers perform with slowly and rapidly varying disturbances. Finally, consideration is given to the case of an uncontrollable system mode, and what happens when the disturbance frequency coincides with that mode. Discussion of the results of these experiments is presented in Chapter VI.

B. IMPLEMENTATION

1. Multiple Error LMS

The Multiple Error LMS algorithm was implemented on the UQP using the Matlab/Simulink environment as discussed in Chapter III. The Simulink block diagram is shown in Figure V-1, which contains many of the same building blocks as the code used for the system identification experiments performed in Chapter IV. In this case the controller requires a disturbance-correlated signal to function properly. A simple unit

delay is used to model a sensor transfer function, and the resulting signal is referred to as $x(k)$ or "the reference signal". The "Controller" subsystem in the Simulink diagram consists of the "*C*", "*W*", and "LMS Algorithm" blocks in Figure II-5, and uses C code to implement the Multiple Error LMS Algorithm in accordance with Eq. (2-22).

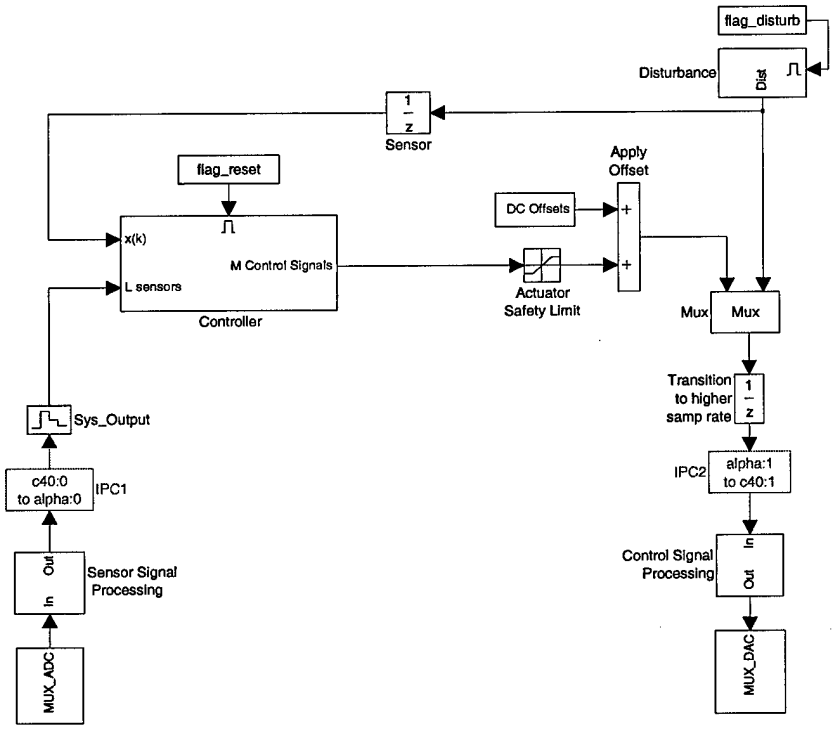

Figure V-1: Multiple Error LMS Controller Simulink Diagram

2. Clear Box

The Clear Box Algorithm (either method) is implemented in much the same way as the Multiple Error LMS Algorithm. The most significant difference is that the Clear Box controllers do not require a sensor to provide a disturbance correlated reference

signal (Figure V-2). The algorithm provides its own disturbance correlated signal through calculation of the disturbance effect, as discussed in Section II.C.5.c). White noise excitation is added to the system input when taking data for system identification. Although system ID can be accomplished as often as necessary, it is generally only accomplished at the beginning of an experiment. The Sine/Cosine Method of Clear Box requires additional processing to identify the disturbance frequencies, and this function is performed by the host PC using data downloaded from the DSP.

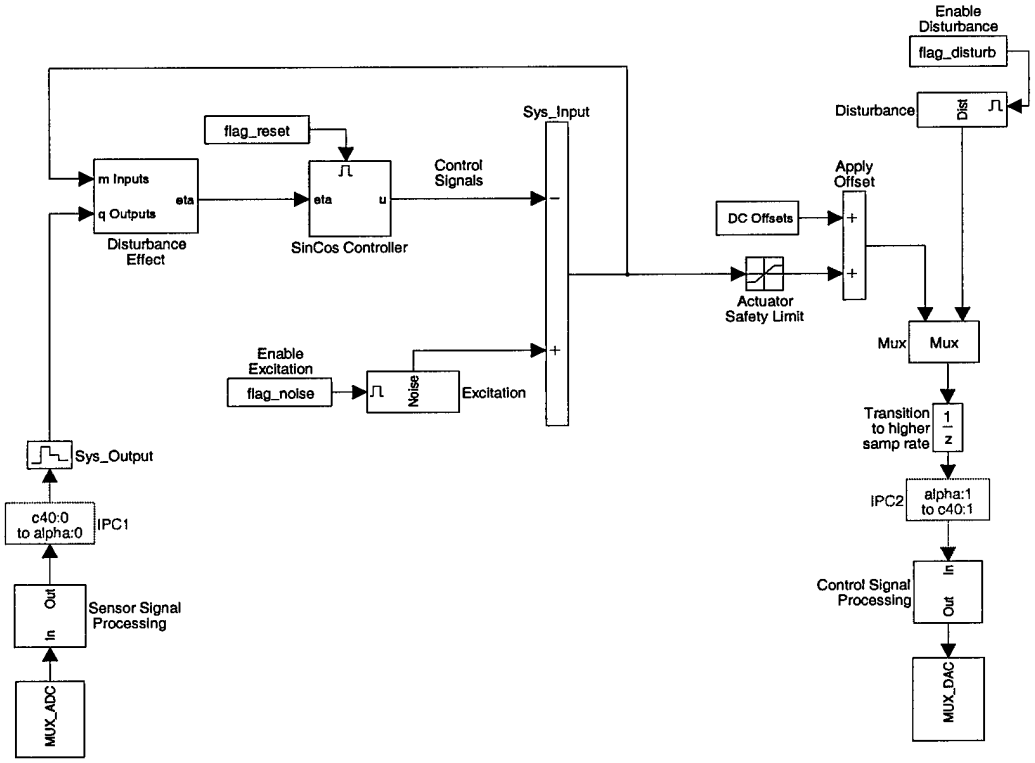

Figure V-2: UQP Clear Box Controller Simulink Diagram

C. REJECTING STATIC DISTURBANCES

The first set of experiments demonstrates the relative ability of each controller to reject a constant sinusoidal disturbance at a single, unchanging frequency. This case allows a methodical investigation into the effects of altering any of the available parameters (adaptation rate, recursive forgetting factor, model order, etc.) The second set of experiments shows behavior in the presence of multiple disturbance frequencies.

1. Single Static Disturbance Frequency

a) Multiple Error LMS

The first experiment attempts to cancel a 120 Hz disturbance which is imparted on the system by means of the Aura bass shaker mounted to the bottom of the satellite bus mockup, to which the UQP is mounted. The amplitude of the disturbance is chosen such that there is a significant output at the strut sensors, without going beyond the linear range of the shaker. To do so would introduce harmonics of the fundamental frequency, which is a case that will be explored later. The results of the first experiment can be seen in Figure V-3, which shows all six strut sensor outputs during the course of the experiment, with the controller being initiated at approximately 0.5 seconds. The result is a 40 dB drop in the overall RMS level of the sensor outputs.

A more revealing way to view the result of the control action is to look at the power spectrum of one of the sensors (strut #1 was selected) before and after the controller is activated as shown in Figure V-4. Clearly the energy at 120 Hz has been

removed from the sensor signal. There is, however, some (minor) negative impact on the frequencies immediately surrounding the disturbance frequency.

The control signals for the six struts, shown in Figure V-5, are determined by filtering the reference signal. The coefficients for this control filter W converge via gradient descent to the "optimal" solution at a rate that is determined by the value of the adaptation rate μ and (as will be seen later) by the mean square level of the filtered reference signal $r(k)$. In this experiment the adaptation rate was selected as 0.1. Figure V-6 shows the coefficients associated with strut #1. In this case a tenth order filter is used, but for a single, zero-mean disturbance frequency only two are required. Note that each of the six sensors has a small offset bias that is removed for data display purposes.

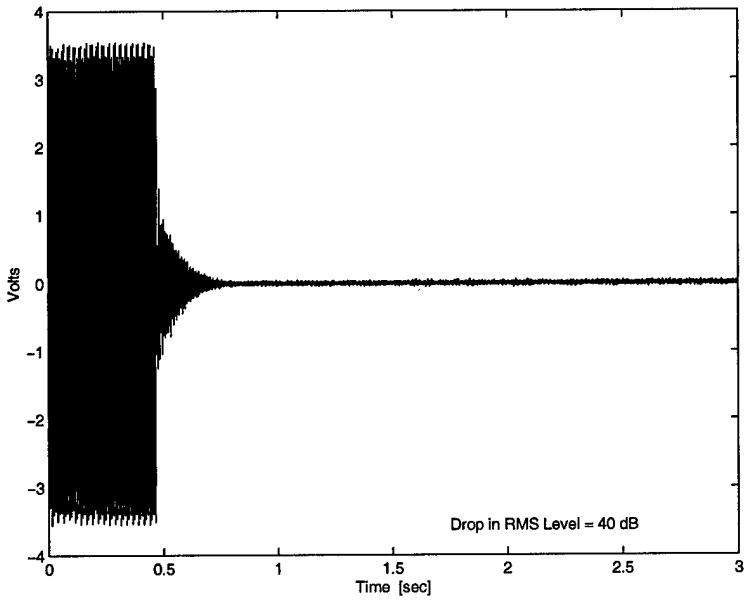

Figure V-3: Multiple Error LMS Results for a 120 Hz Disturbance

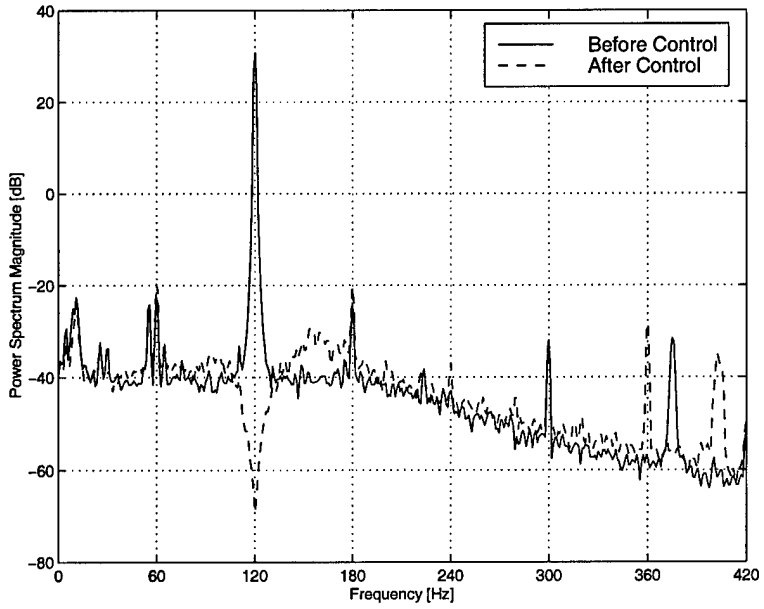

Figure V-4: Power Spectrum Comparison, 120 Hz Disturbance

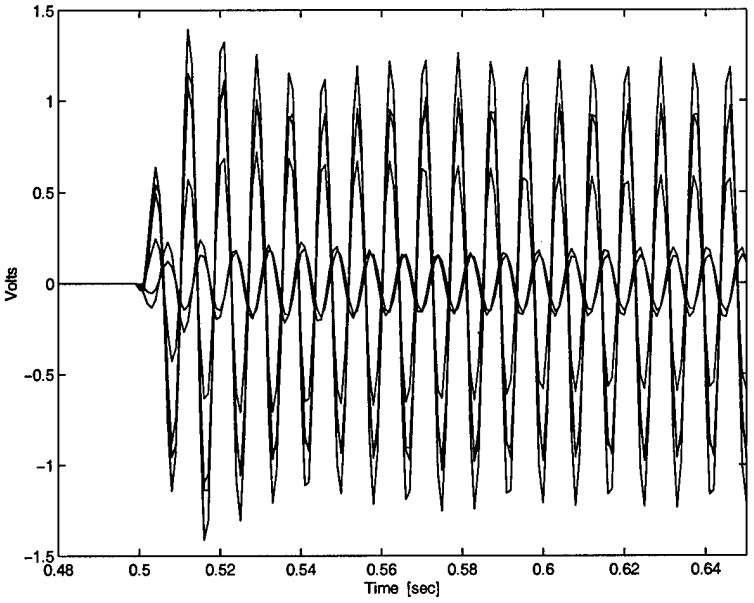

Figure V-5: Control Signals, Struts 1-6, 120 Hz Disturbance

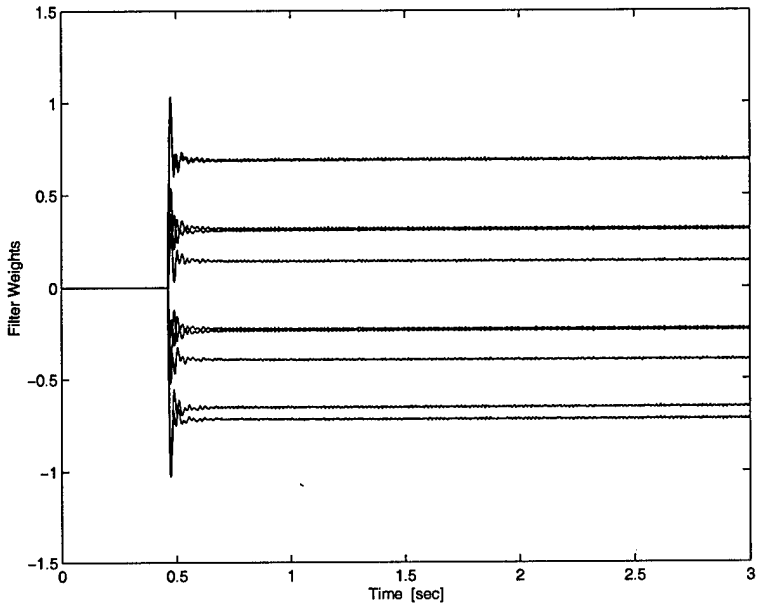

Figure V-6: Coefficient Convergence, μ= 0.08, 120 Hz Disturbance

The selection of μ has a great impact on the performance of the controller. To demonstrate this, two more experiments are run using different adaptation rates. A lower rate of 0.01 (Figure V-7) shows that slower convergence results, and the coefficient values are also less "noisy". Increasing the rate to 0.13 results in instability (Figure V-8), so care must be taken to chose a conservative (low) adaptation rate.

Also important to the selection of the adaptation rate is a knowledge of the bounds on the disturbance frequencies to be controlled. A given adaptation rate μ results in greatly different performance for high and low frequency disturbances. The reason for this is that the mean square level $\overline{r}^2$ of the filtered reference signal $r(k)$ increases with increasing frequency since the magnitude of the plant model generally

increases with frequency (Figure IV-11). Recall that the upper bound for μ for stable operation is inversely related to $\bar{r}^2$ (Eq. (2-23)), and thus the effective adaptation rate is affected by the disturbance frequency. As a demonstration, the original adaptation rate of 0.08 is used on two disturbances, one at 40 Hz, and another at 150 Hz. The resulting coefficient conversion is seen in Figure V-9 and Figure V-10. The controller converges very slowly for the 40 Hz disturbance, and exhibits instability for any frequency at or above 150 Hz. In general, the best performance for a given disturbance frequency is obtained by "tuning" the adaptation rate in a manner inversely proportional to $\bar{r}^2$.

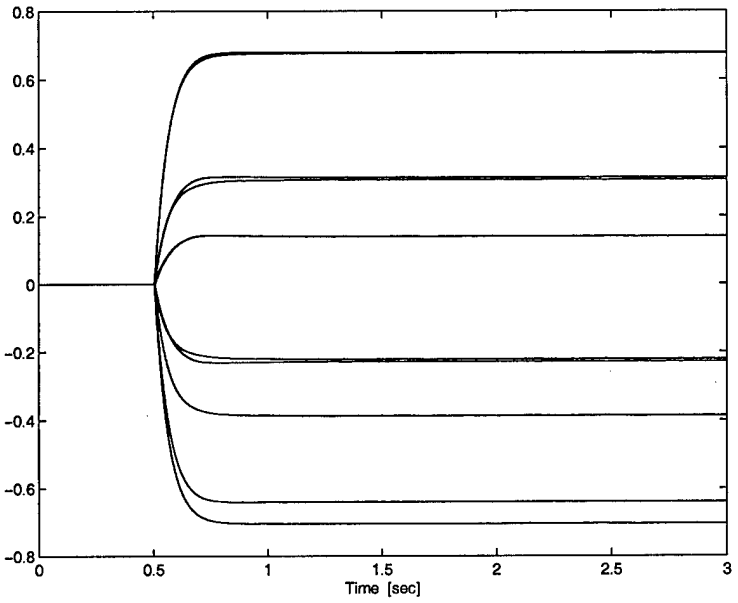

Figure V-7: Coefficient Convergence, mu=0.01, 120 Hz Disturbance

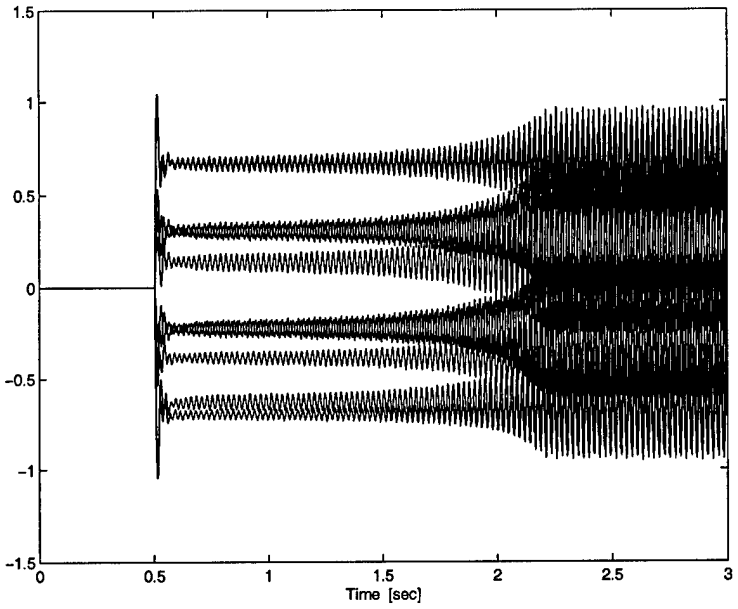

Figure V-8: Coefficient Convergence, mu=0.13, 120 Hz Disturbance (Unstable)

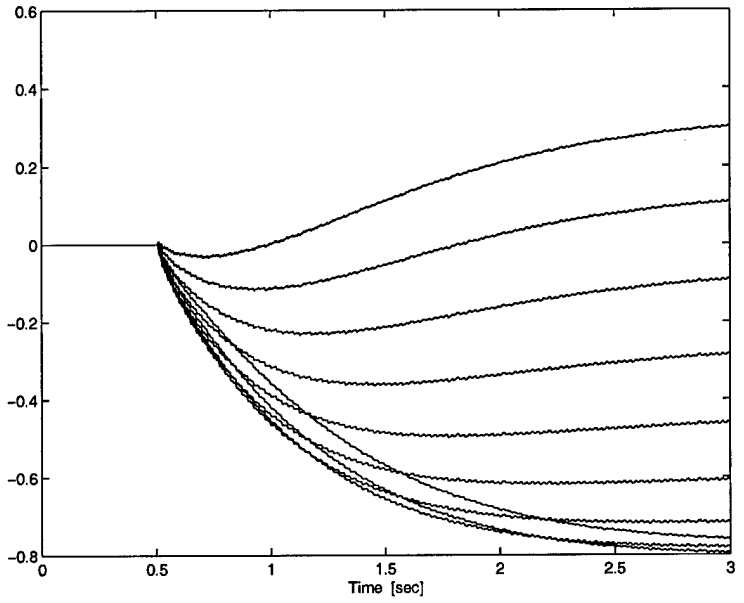

Figure V-9: Coefficient Conversion, mu=0.08, 40 Hz Disturbance

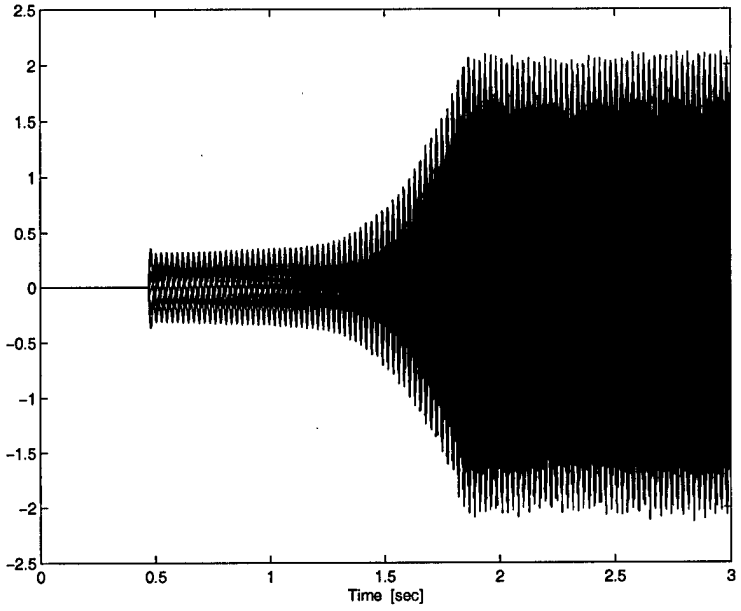

Figure V-10: Coefficient Convergence, mu=0.08, 150 Hz Disturbance (Unstable)

Returning to the case of the 120 Hz disturbance, it is possible to characterize the steady state performance by looking at the RMS level of each strut's sensor, and comparing it to that of the sensor's background noise level. After each experiment is completed, a brief pause allows the transients to die out, and a reading of the "background" sensor noise levels is taken. These noise levels change depending on the vibration and acoustic environment in the room, but the RMS level is generally from 0.012 – 0.016 Volts. Figure V-11 shows that the overall sensor output RMS level is reduced to a level just above that of the sensor noise, down 40 dB from the pre-control level of 1.61 Volts.

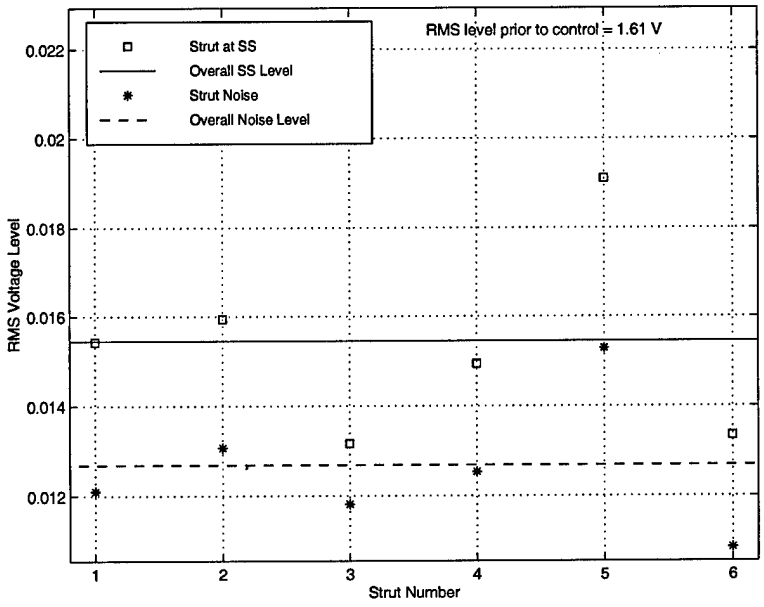

Figure V-11: Steady State and Sensor Noise Levels, 120 Hz Disturbance

To fully characterize the performance of the Multiple Error LMS algorithm for a single static frequency disturbance, experiments are run over a range of frequencies using three different values for I, the control filter order. In each experiment, the adaptation rate is tuned for best performance. The results are shown in Figure V-12, which shows both the dB reduction in the RMS level for each frequency, and the relative "position" of the steady state levels in comparison to the sensor noise level (positive levels are above the noise level). Note that the amount of dB drop is related to the initial level of the disturbance, so if the same voltage signal is sent to the shaker for each frequency, then the response will vary according to the frequency response of the shaker (Figure IV-8 on page 84). Clearly the dB drop in Figure V-12 is related to this shaker frequency response. Thus the more accurate measure of the performance of the controller

is to compare the steady state noise level with the sensor noise level. Note that the filter order has little effect on the controller performance for a single, static disturbance frequency.

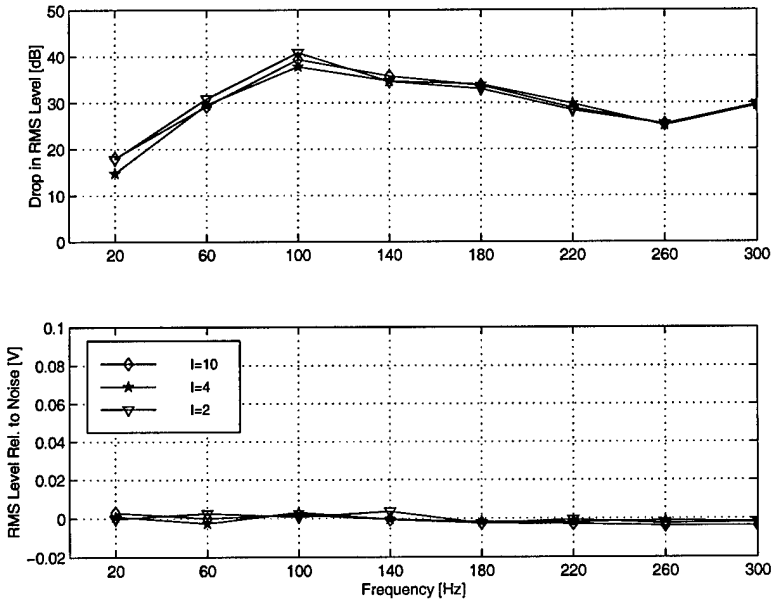

Figure V-12: Multiple Error LMS Performance vs. Frequency

Finally, to test the stability of the Multiple Error LMS Algorithm, an impulsive disturbance is imparted to the experiment while the controller is operating. Return of the control coefficients to pre-control levels would give an indication of stability. Figure V-13 and Figure V-14 show the resulting sensor outputs and control coefficients, respectively. They indicate that the Multiple Error LMS Algorithm is stable for this type of disturbance.

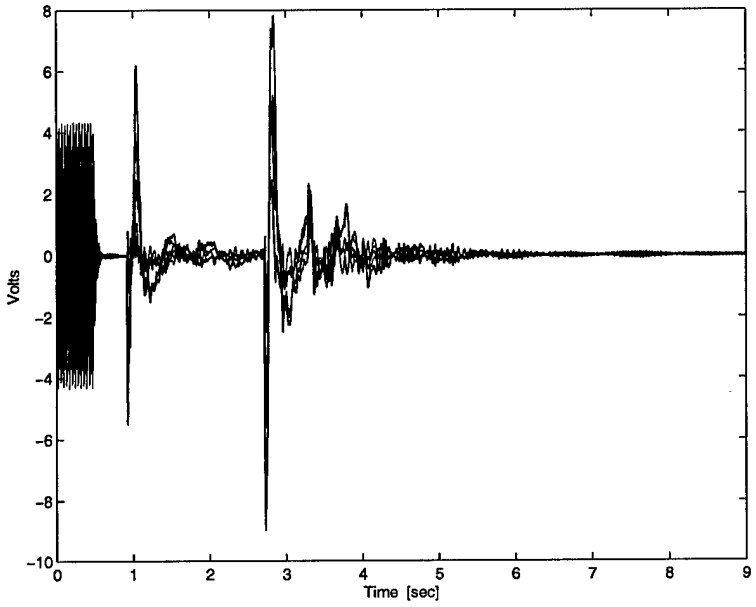

Figure V-13: Multiple Error LMS, Sensor Outputs - Impulsive Disturbance

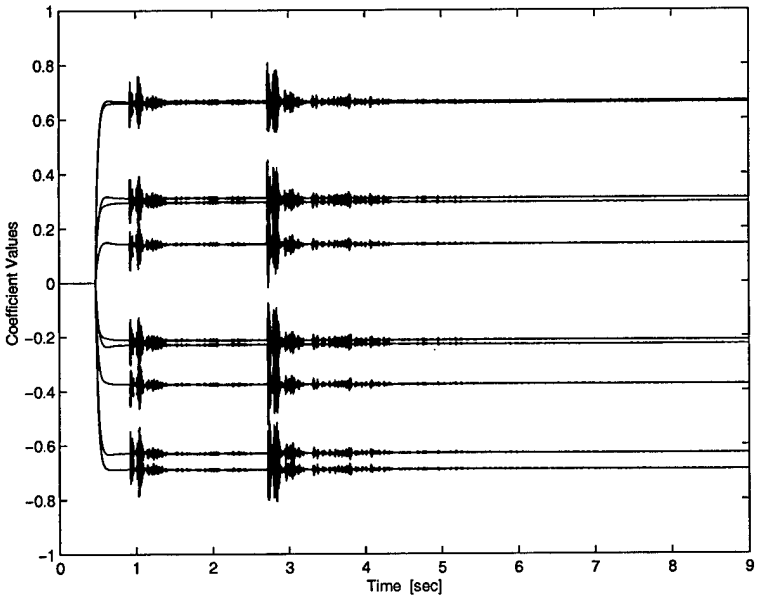

Figure V-14: Multiple Error LMS, Control Coefficients - Impulsive Disturbance

Also tested was the impact of plant modification on the controller's performance by adding a significant amount of mass to the top of the platform. This had very little impact on the performance of the Multiple Error LMS controller, and the filter weights were able to adapt to new optimal values. Similar experiments to those performed above will be performed on the UQP in the next two sections using the Clear Box Algorithm (both methods), and comparisons between controllers will be made in Chapter VI.

b) Clear Box, Sine/Cosine Method

The implementation of the Clear Box algorithm was presented briefly in Section B.2. Recall from Section II.C.6.b) starting on page 44 that the Sine/Cosine Method forms a control signal by recursively estimating the coefficients of sine/cosine pairs (one pair for each disturbance frequency). The steady state values of the coefficients should result in a minimization of the error signals at the sensor outputs.

The first experiment using the Sine/Cosine Method uses the same 120 Hz constant disturbance frequency as that used earlier with the Multiple Error LMS algorithm. The recursive estimation process uses a "forgetting factor" λ described in Eq. (2-74), and an initial covariance associated with the coefficients. Also a "threshold level" for the damping ratio must be selected, below which the identified frequency component of the disturbance effect signal is considered a disturbance.

The results of the first experiment (controlling a static 120 Hz disturbance) can be seen in Figure V-15, where the control action is initiated at $t = 0.5$ seconds. Similar results to the Multiple Error LMS algorithm are achieved, including a rapid

convergence and reduction of the disturbance by, in this case, 40 dB. In the frequency domain (Figure V-16) it is seen that a similar (perhaps more narrow) notch is formed at the disturbance frequency as was seen in Figure V-4. However, there is less of an adverse impact of the controller on frequencies immediately surrounding the disturbance.

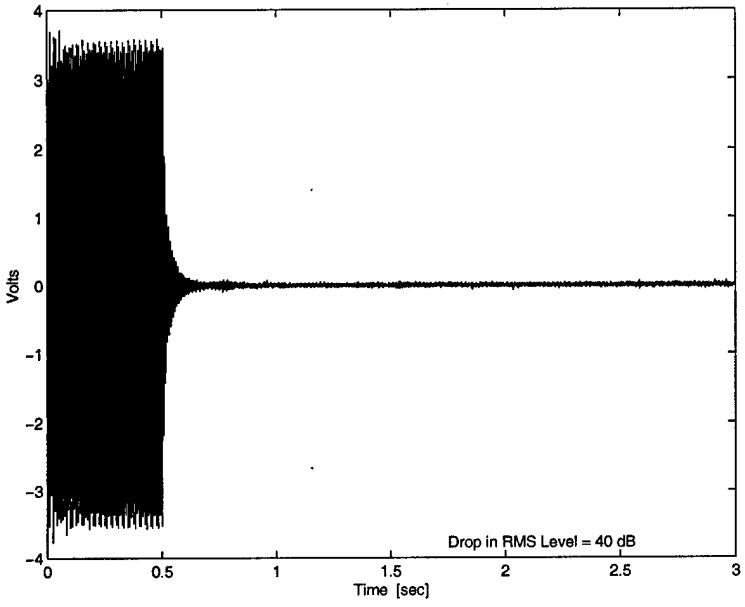

Figure V-15: Clear Box Sine/Cosine Sensor Outputs, 120 Hz Disturbance

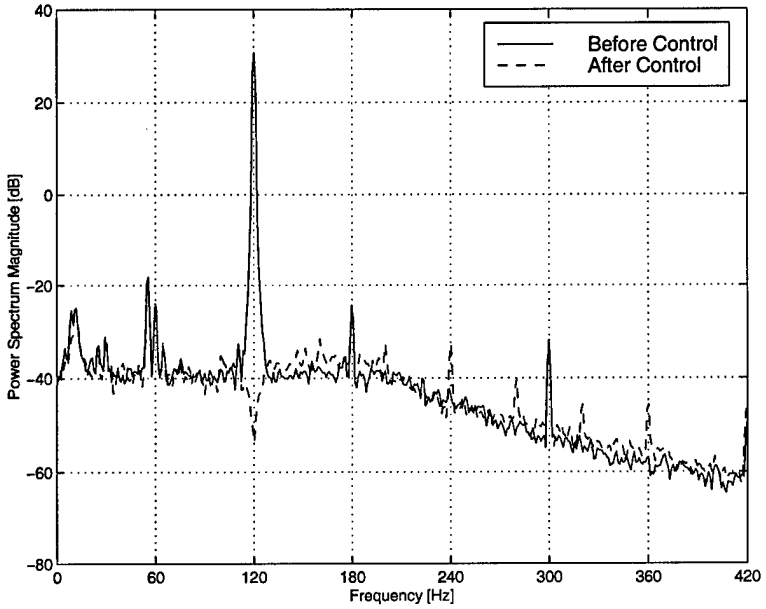

Figure V-16: Sine/Cosine Spectral Comparison, 120 Hz Disturbance

The control signals used to drive the six actuators are shown together in Figure V-17. Each strut's control signal is generated by a sine/cosine pair for each disturbance frequency, so in this case there are twelve coefficients (two per strut) that must be recursively estimated. For the single, 120 Hz disturbance the time histories of the two recursively calculated coefficients for strut #1 are shown in Figure V-18. In this case λ (the forgetting factor) was set to 0.98. A value closer to 1 gives a "smoother" history that responds less quickly to changes in the calculated disturbance effect, and a smaller value gives a more "noisy" history. For a static disturbance, the choice of forgetting factor does not have a great deal of impact on the performance, unless the disturbance frequency estimate is incorrect (demonstrated later in this section).

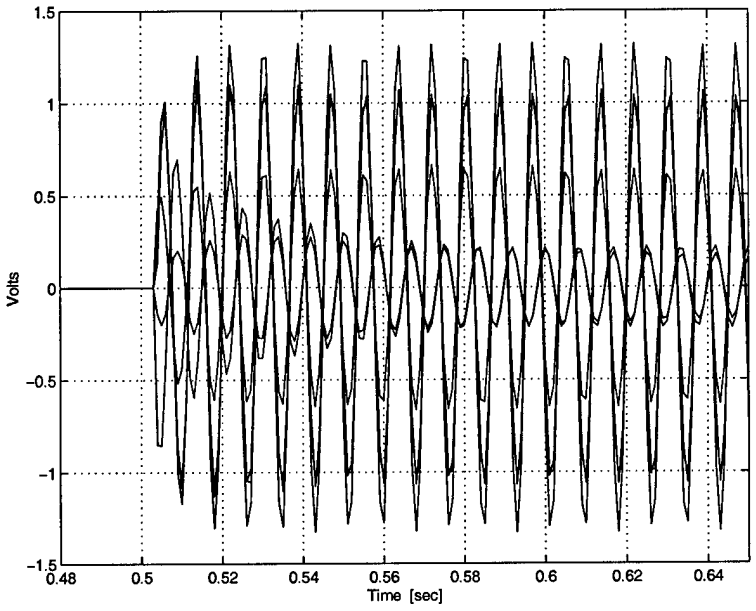

Figure V-17: Sine/Cosine Control Signals, 120 Hz Disturbance

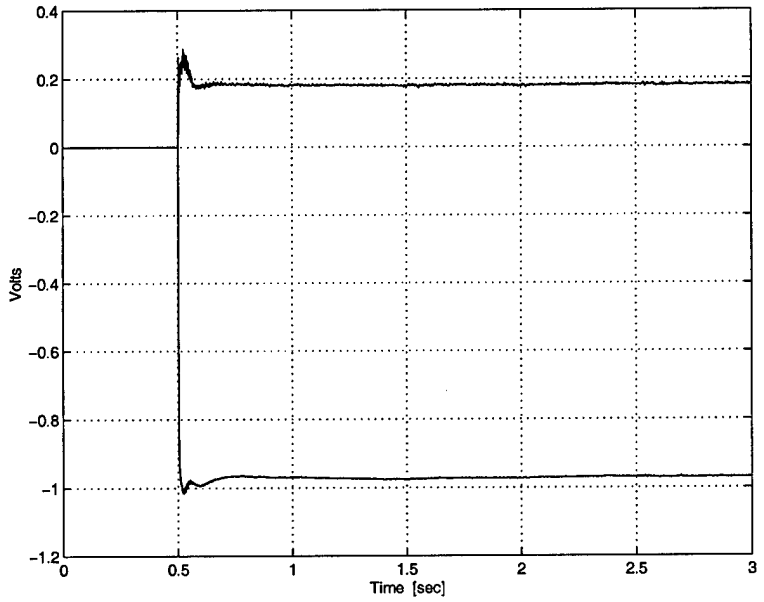

Figure V-18: Sine/Cosine Control Coefficients, Strut #1, 120 Hz Disturbance

In a manner similar to that used with the Multiple Error LMS experiments, the RMS level of the steady state sensor outputs is calculated for each strut, and overall. As is seen in Figure V-19 the results are similar to those obtained using Multiple Error LMS. The overall RMS level is only slightly higher than the level of the noise.

Similar experiments were performed over a range of frequencies, and the results are plotted in Figure V-20. The performance is very similar to that of the Multiple Error LMS controller, controlling the disturbance close to the noise level in all frequencies tested. Again, the dB drop is seen to correspond roughly to the shaker frequency response. In general, there are no parameters that need to be "tuned" to achieve good performance, in contrast to the Multiple Error LMS algorithm where the adaptation rate must be selected based on the frequency of the disturbance (see previous section).

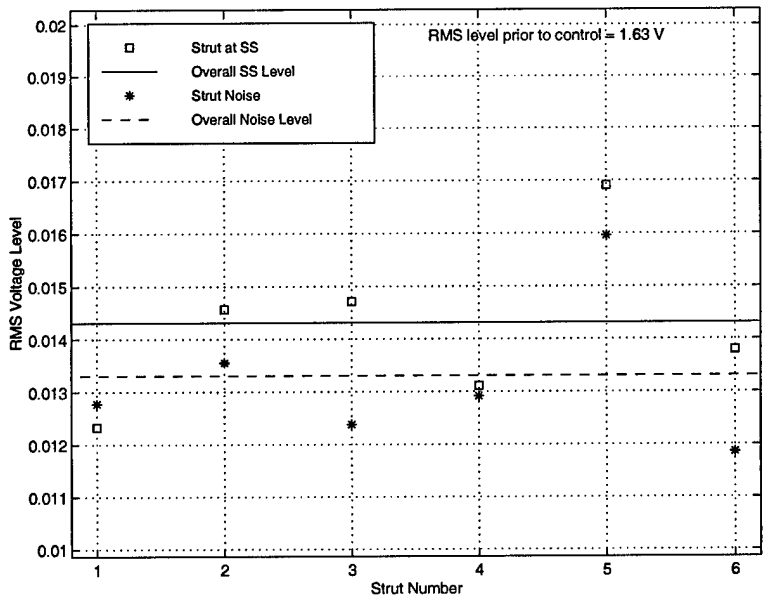

Figure V-19: Steady State and Sensor Noise Levels, 120 Hz Disturbance

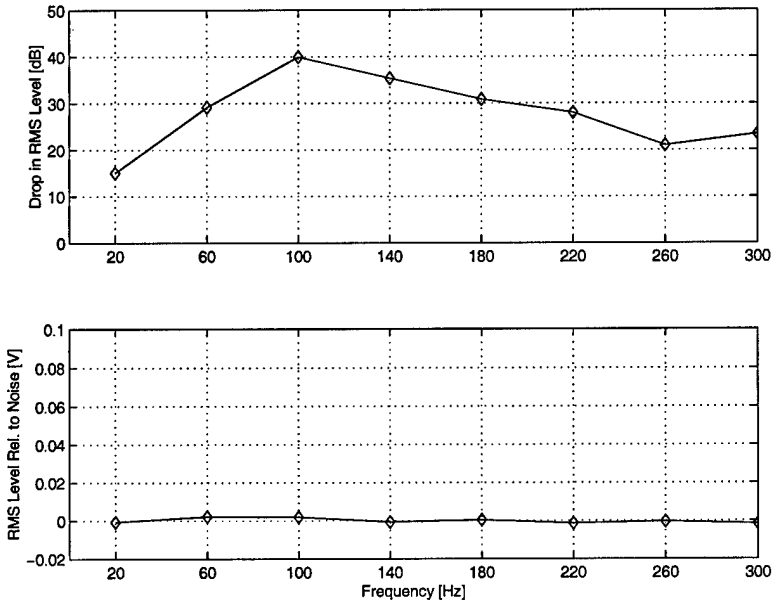

Figure V-20: Clear Box Sine/Cosine Method, Performance vs. Frequency

The stability of the Sine/Cosine Method is tested by imparting disturbances to the platform. The controller is very robust with respect to disturbances, and even heavy shaking of the platform does not cause instability. A sample of the six sensor outputs and the strut #1 control coefficients are shown in Figure V-21 and Figure V-22, respectively.

Also tested was the impact of plant modification on the controller performance. Addition of significant mass to the top of the platform had very little impact on the performance of the Sine/Cosine Method controller, and the coefficients were able to adapt to new optimal values.

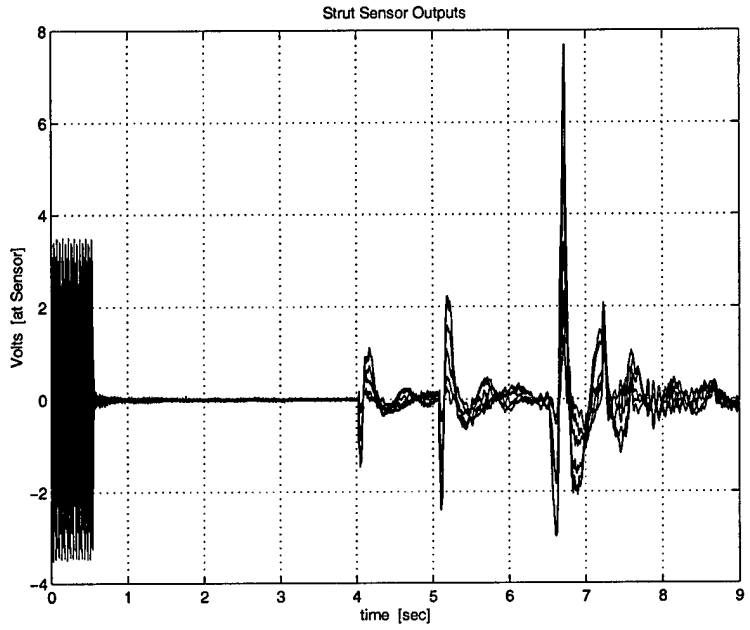

Figure V-21: Sine/Cosine Method, Sensor Outputs - Impulsive Disturbances

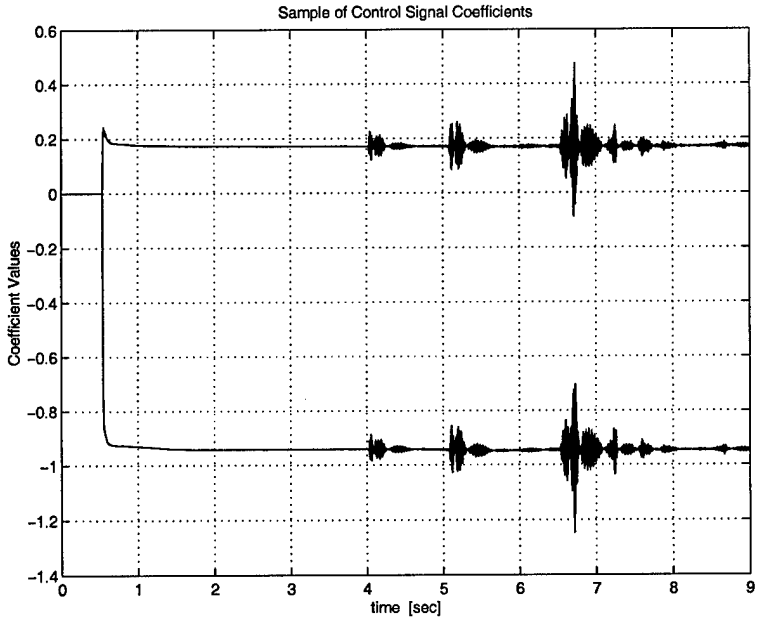

Figure V-22: Sine/Cosine Method, Control Coefficients - Impulsive Disturbances

The role that the forgetting factor plays in the performance of the Sine/Cosine controller depends on the accuracy of the disturbance frequency estimate. To demonstrate this, a static disturbance frequency of 150 Hz is used, and the estimate of the frequency is deliberately set at various values (identified by percent error). The result is seen in Figure V-23, which shows that more accurate disturbance frequency estimates help improve performance.

The forgetting factor becomes an important tool for improving performance of the Sine/Cosine controller when the frequency estimate is incorrect. "Forgetting" more of the past data allows the control signal coefficients to "cycle" and minimize the sensor error (see discussion in Section II.C.6.b). A demonstration of the effect is accomplished with a static disturbance by keeping the error of the frequency estimate constant, and then using a variety of forgetting factors to see how performance is effected. Figure V-24 uses data for four different error levels to show that improved performance is obtained by using a smaller forgetting factor (recall that a *smaller* forgetting factor provides *more* "data forgetting"). Use of a forgetting factor lower than 0.65 has little additional impact, however. The coefficient "cycling", discussed earlier in Section II.C.4, is shown in Figure V-25 demonstrating that the coefficients oscillate at a frequency equal to the difference between the actual and the estimated frequency (in this case the frequency estimate was off by 1% of 100 Hz, so the cycling was at 1 Hz).

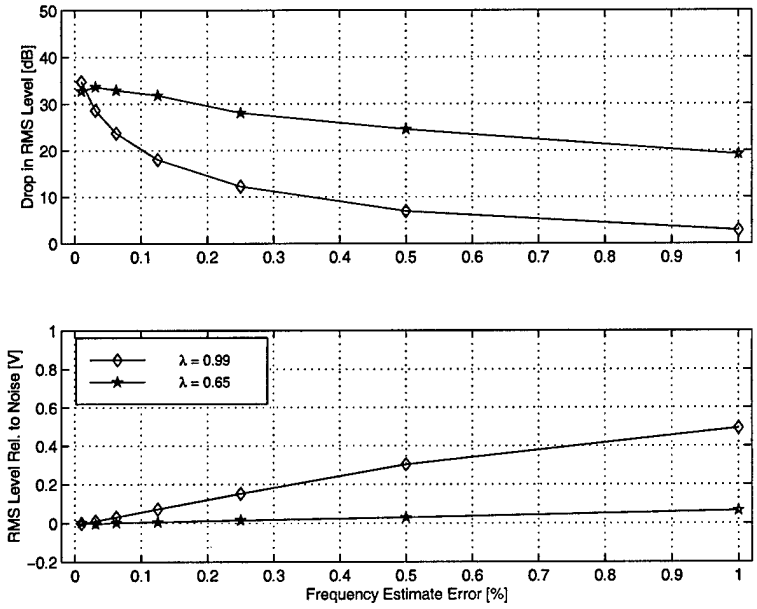

Figure V-23: Effect of Frequency Error on Sine/Cosine Method Performance

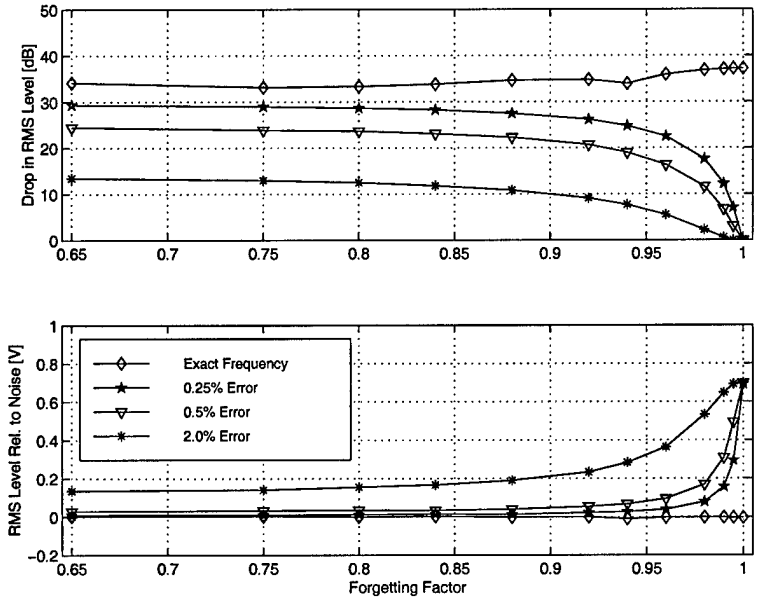

Figure V-24: Effect of Forgetting Factor on Sine/Cosine Method Performance

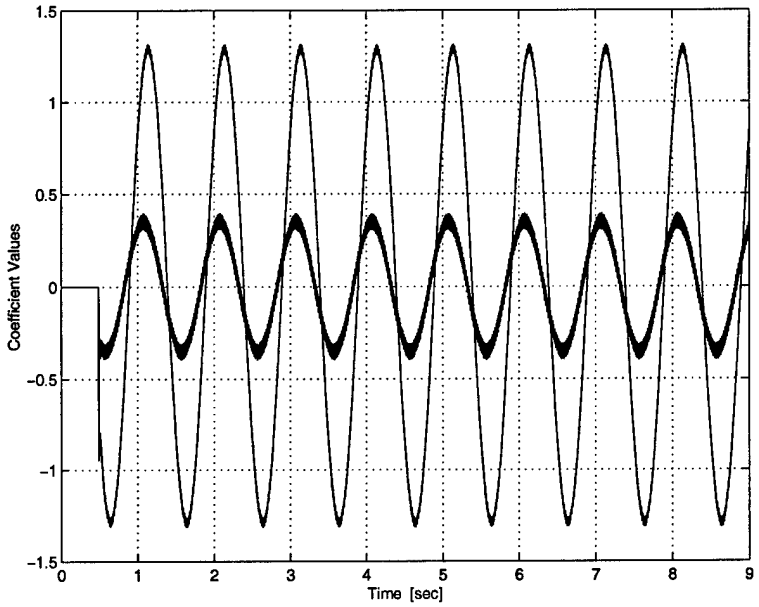

Figure V-25: Demonstration of Coefficient Cycling

Note that during the course of the single static disturbance frequency experiments using the Sine/Cosine controller, a typical error in the estimate of the disturbance frequency was 0.001% and was always less than 0.02%. Thus for static frequencies the issue is not critical. However, if the disturbance frequency is changing rapidly, then the forgetting factor plays an important role in achieving good performance, as will be demonstrated in Section D.1.b) starting on page 134.

c) Clear Box, Adaptive Basis Method

The Adaptive Basis Method controller is implemented in a fashion identical to that of the Sine/Cosine Method. The difference between the two is that the Adaptive Basis Method's recursive algorithm estimates control signal coefficients for a

basis that consists of the time-shifted disturbance effect signal, instead of pairs of sine and cosine functions. Estimation of the disturbance frequency(ies) is not required since this information is included in the disturbance effect signal.

The first experiment with the Adaptive Basis controller is identical to that conducted in the previous sections. A constant 120 Hz disturbance is imparted to the UQP via the bass shaker. Recursive estimation of the control coefficients results in minimization of the error signal at the sensor outputs, as shown by the time history of the six sensor outputs in Figure V-26. Inspection of the spectral content of the strut #1 error signal before and after the controller is activated (Figure V-27) reveals that the 120 Hz signal is completely eliminated. A small amount of amplification is present in frequency bands centered around AC power frequency multiples (i.e., 60 Hz $\times k$, $k = 1, 2, \ldots$).

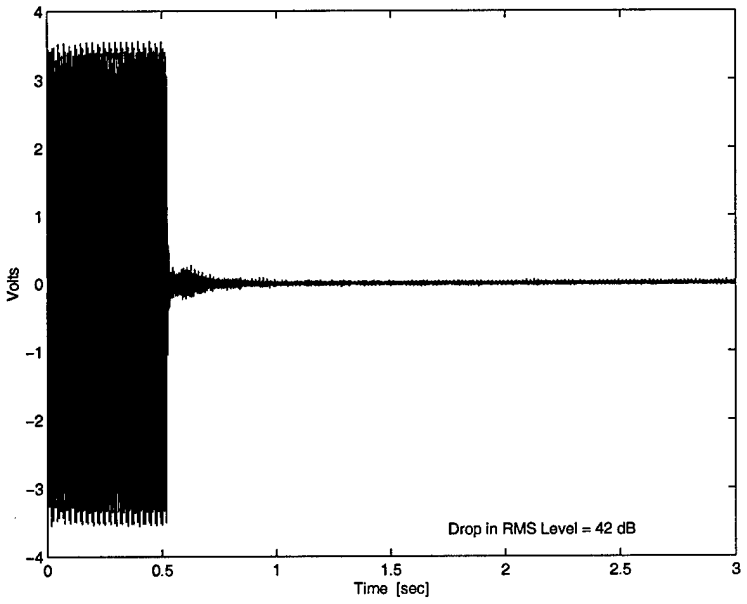

Figure V-26: Clear Box Adaptive Basis Controller Results for a 120 Hz Disturbance

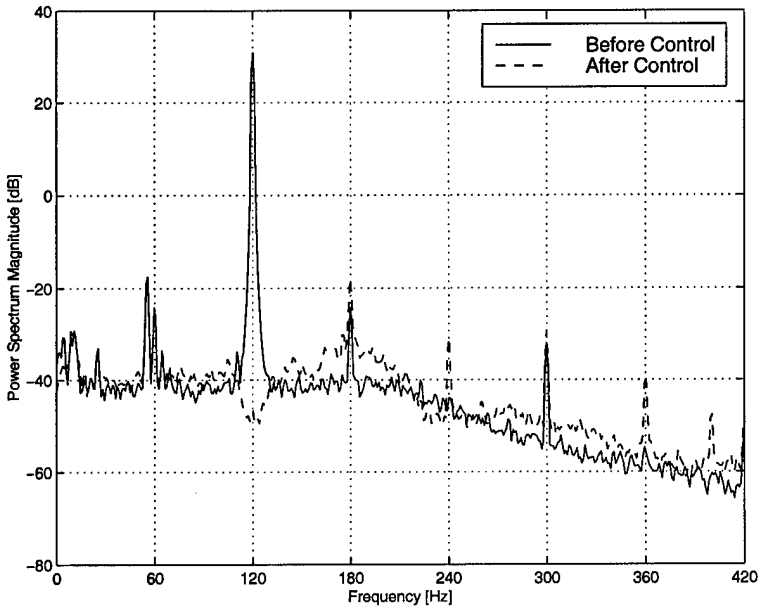

Figure V-27: Power Spectrum Comparison, 120 Hz Disturbance

The sinusoidal control signals for the six actuators are shown in Figure V-28. The control signals represent a linear combination of the η shifted time histories. As discussed in II.C.6.c), a single strut's disturbance effect time history can serve as the basis for all six actuators' control signals. The coefficients for each strut adapt as needed to account for unique gain and phase requirements. In this experiment six time-shifted η signals are employed ($N = 6$), and the convergence of their associated coefficients is shown in Figure V-29. If the sensor signals and the calculated η signal are noise-free and zero-mean, two time-shifted signals are sufficient to completely cancel the disturbance. In this case performance is improved by using redundant time shifted signals (demonstrated later).

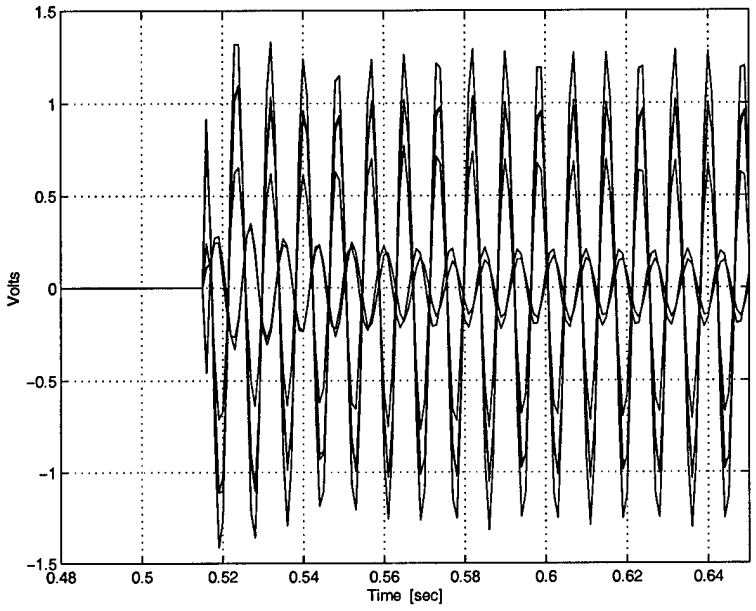

Figure V-28: Adaptive Basis Method Control Signals, 120 Hz Disturbance

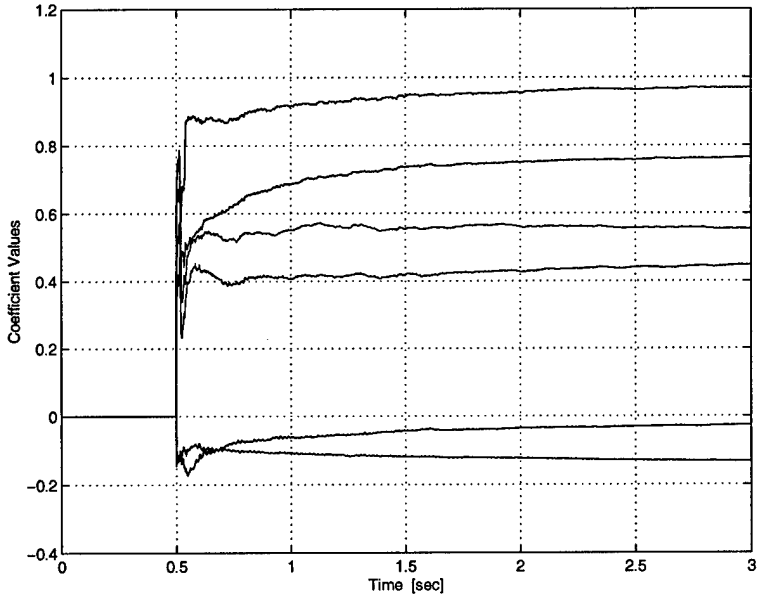

Figure V-29: Adaptive Basis Method Control Coefficients, 120 Hz Disturbance

In a manner similar to the experiments with the other controllers, the steady state RMS level of the sensor signals can be compared with the sensor noise level, which can vary as the room environment changes. Thus, the sensor noise level may be at different levels for different experiments. Figure V-30 reveals that for this experiment, the steady state sensor RMS level was actually below the sensor noise level (discussed in Chapter VI).

One parameter that the user can choose is the number of time shifted disturbance effect signals N to use in forming the control signal. Since the coefficients must be estimated for each time shifted signal, increasing the number of shifts also increases the computational burden on the processor. Using a model order p of 15 the maximum number of time shifts that can be handled by the alpha processor operating at 1 kHz is 6. Performing experiments at various disturbance frequencies using $N = 2, 4, \text{and } 6$ allows the overall performance of the Adaptive Basis controller to be characterized, and is shown in Figure V-31. Clearly, using more time shifts improves the performance and extends the frequency range over which disturbances can be effectively controlled.

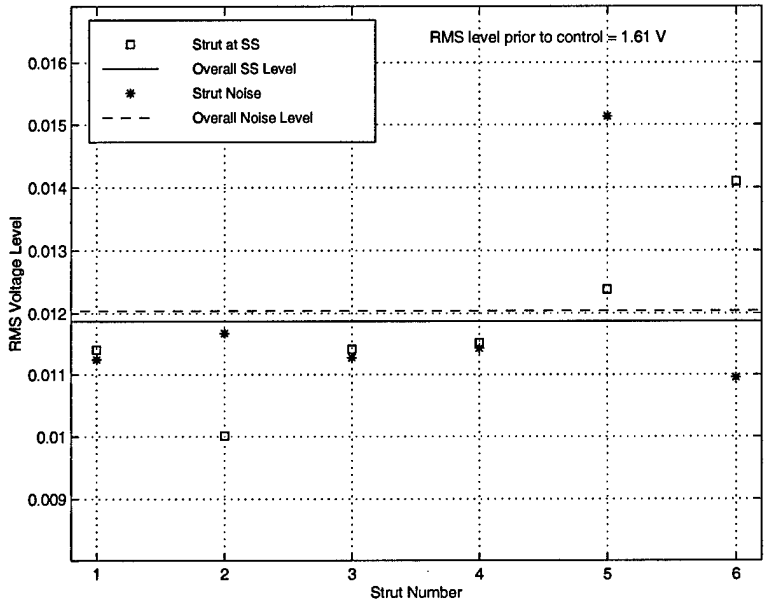

Figure V-30: Steady State and Sensor Noise Level, 120 Hz Disturbance

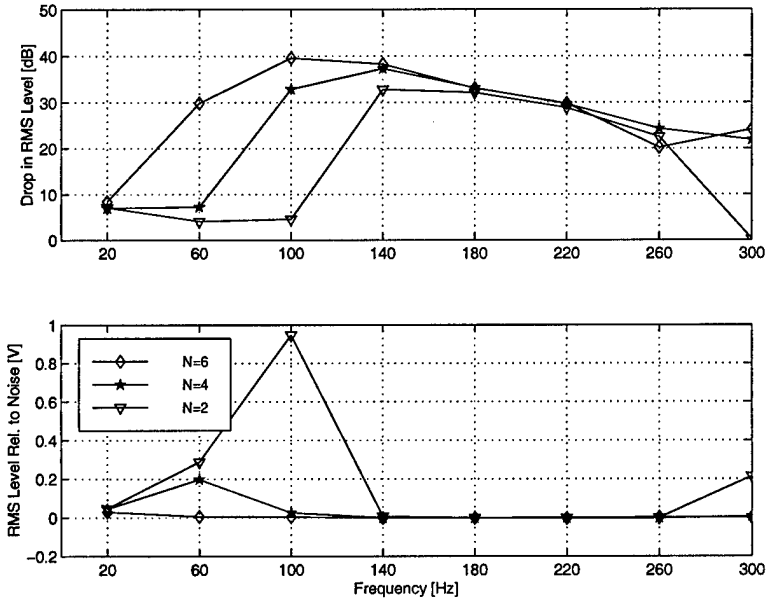

Figure V-31: Adaptive Basis Method Performance vs. Frequency

To assess the stability of the controller while in operation, an impulsive disturbance is applied to the top of the UQP platform while controlling a 120 Hz disturbance. Figure V-32 and Figure V-33 show the sensor output and control coefficient time histories, respectively, and reveal that pre-impulse conditions are re-established after a short adjustment period. Using a forgetting factor less than one reduces the time required to recover, but may adversely effect steady-state performance. Note that very large impulsive disturbances were observed to cause instability of the controller. The effects of plant modification (addition of mass to the platform) were also tested on the Adaptive Basis Method controller, with very little impact on performance.

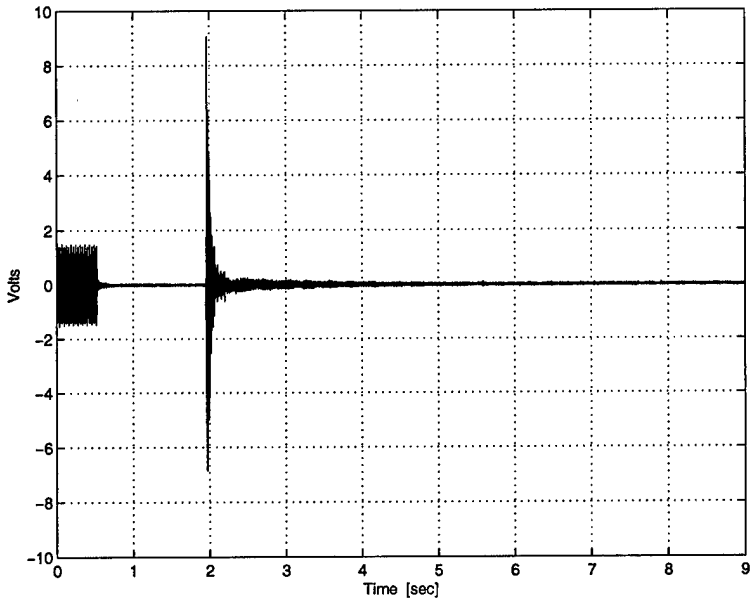

Figure V-32: Adaptive Basis Method, Sensor Outputs – Impulsive Disturbance

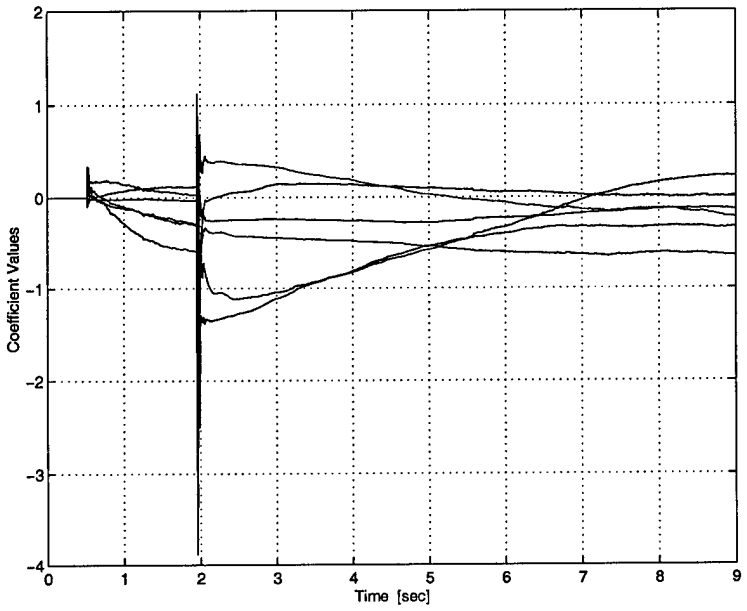

Figure V-33: Adaptive Basis Method, Control Coefficients – Impulsive Disturbance

Recall that for the Sine/Cosine Controller the recursive algorithm's forgetting factor greatly effects performance when the disturbance frequency estimate is in error, but not when the estimate is exact (Figure V-24 on page 114). In the case of the Adaptive Basis controller there are no frequency estimates required, but the recursive algorithm can still use a forgetting factor, allowing recent data to be more heavily weighted than past data. Using the same dB drop and RMS level comparisons used earlier, Figure V-34 shows the performance of the Adaptive Basis Method against a static 150 Hz disturbance using various forgetting factors. The results indicate that using all past data improves performance in the case of a disturbance with constant frequency (to be discussed in Chapter VI).

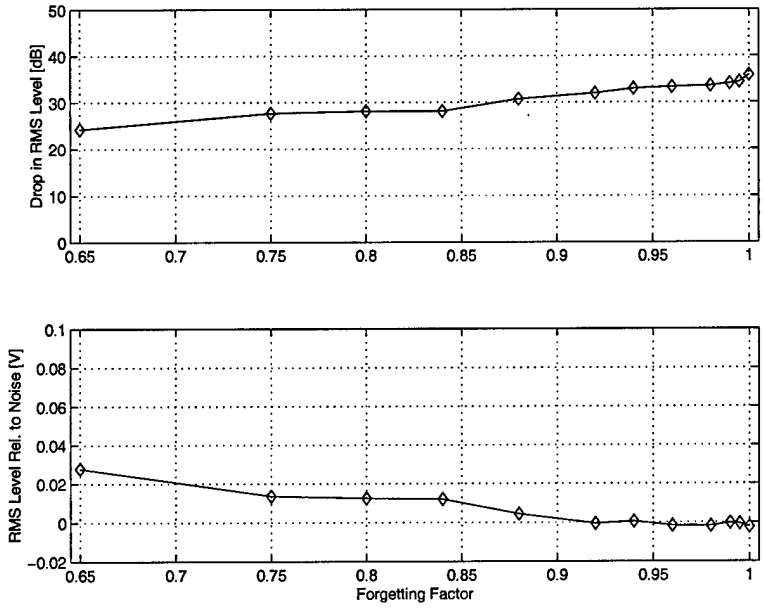

Figure V-34: Effect of Forgetting Factor, 150 Hz Disturbance

Another factor that can affect the Adaptive Basis controller's performance is the accuracy of the identified system model, since the model is used in the calculation of the disturbance effect and the control signal coefficients. The accuracy of the model is improved by using a larger model order p. The effect on controller performance can be seen from the data in Figure V-35 which was compiled from two sets of experiments against disturbances at 90 Hz and 140 Hz. The disturbance amplitudes were adjusted so that the magnitude of the sensor response (without control) was the same for both disturbances. In both sets of data the model order is increased from one experiment to the next. The general trend is toward improved performance with larger p, especially at the lower frequency (90 Hz). In general, performance improves with increasing p and N.

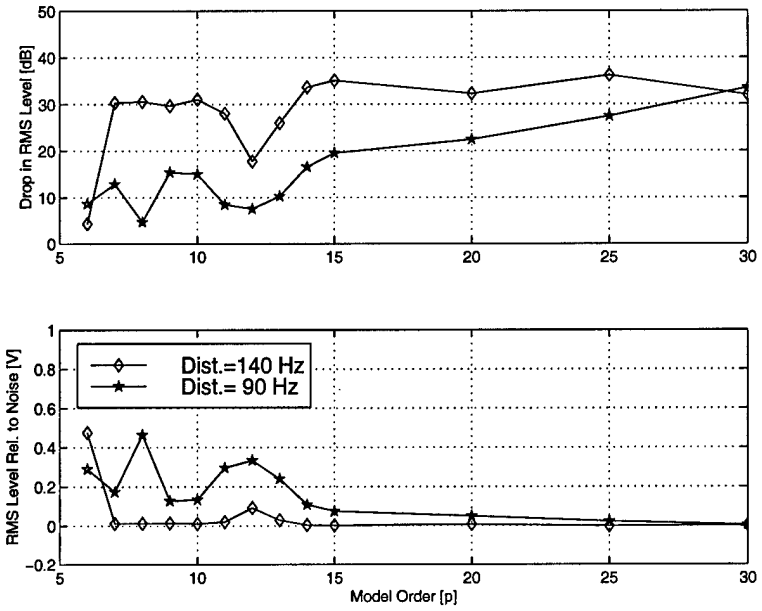

Figure V-35: Adaptive Basis Method, Performance vs. Model Order

2. Multiple Static Disturbance Frequencies

There are two cases of "multiple static disturbances" that are explored in this section. The first demonstrates how each controller can handle the case of two distinct disturbance frequencies. The second is a case where there is a single disturbance frequency that also generates harmonic disturbances.

a) Multiple Error LMS

Two disturbance frequencies, at 95 and 160 Hz, are present for the first experiment. For the Multiple Error LMS algorithm the adaptation rate must be chosen based on the highest frequency (so that the system will remain stable). Figure V-36 and Figure V-37 below show the output history and spectral content of the sensor signals,

respectively. The conservative choice of adaptation rate results in a slightly lengthened period of convergence, but it is still less than 0.5 seconds. Once again the disturbances are reduced to a level comparable to the sensor noise level. The spectral comparisons show complete elimination of the disturbance energy at the two disturbance frequencies of 95 Hz and 160 Hz.

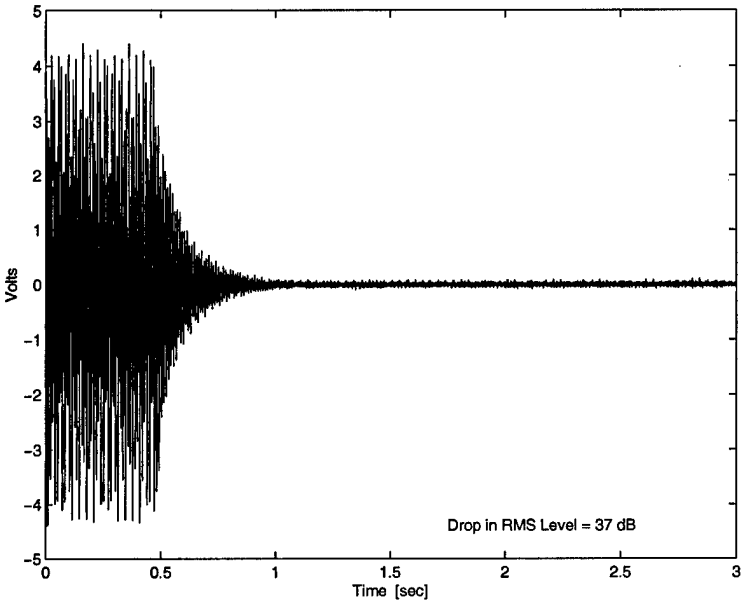

Figure V-36: Multiple Error LMS Sensor Outputs, 2 Static Disturbances

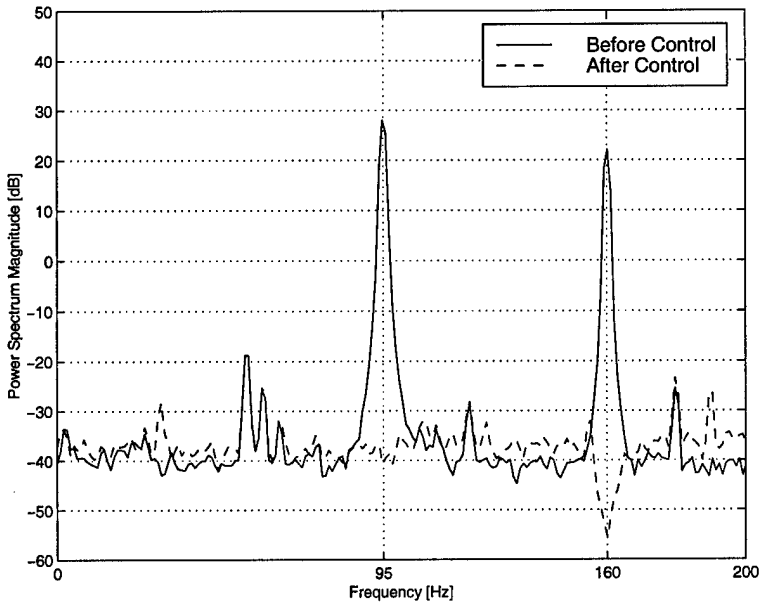

Figure V-37: Multiple Error LMS Spectral Comparison, 2 Static Disturbances

The second experiment with multiple disturbances is a case where the fundamental disturbance frequency is 50 Hz, and two harmonics are present at 100 Hz and 150 Hz. In such a case the 50 Hz signal is sent to the shaker and also acts as the reference signal $x(k)$, after passing through the unit delay which represents the sensor transfer function (Figure V-1 on page 94). The dynamics of the shaker-to-sensor system at 50 Hz are such that harmonic disturbances are also generated. These harmonic frequencies are not present in the $x(k)$ signal. Figure V-38 below shows the before/after comparison of the spectral content of the sensor signal for strut #1. Clearly the fundamental frequency is controlled, but the harmonics at 100 Hz and 150 Hz are not.

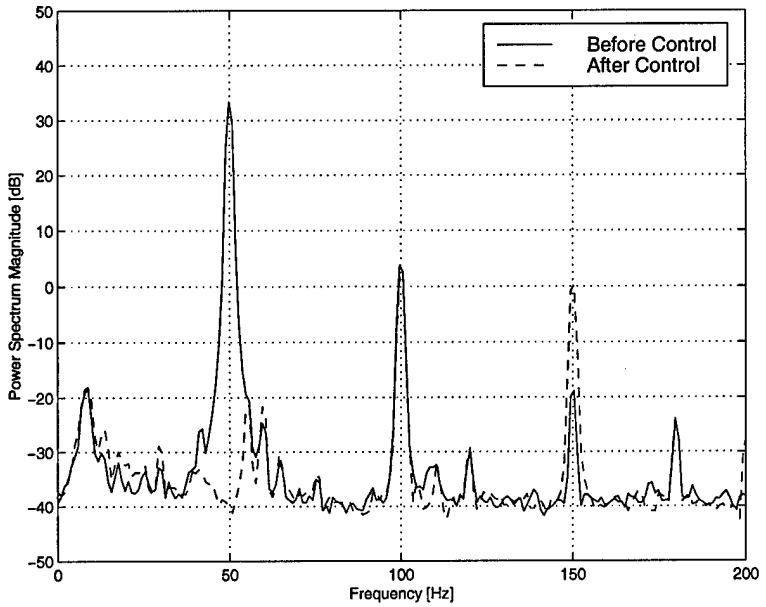

Figure V-38: Spectral Comparison, Multiple Error LMS Controlling Harmonics

b) Clear Box, Sine/Cosine Method

The same two cases used above are tested using the Sine/Cosine Method of the Clear Box algorithm. The first experiment is with two distinct frequencies at 95 Hz and 160 Hz. The sensor outputs are shown in Figure V-39, and the before/after spectral content of the strut #1 sensor is shown in Figure V-40. Use of a forgetting factor less than one helps improve performance, and in this case the value was set to 0.99. The fact that the control signal is formed from sine and cosine functions results in no impacts at frequencies other than the disturbance frequencies.

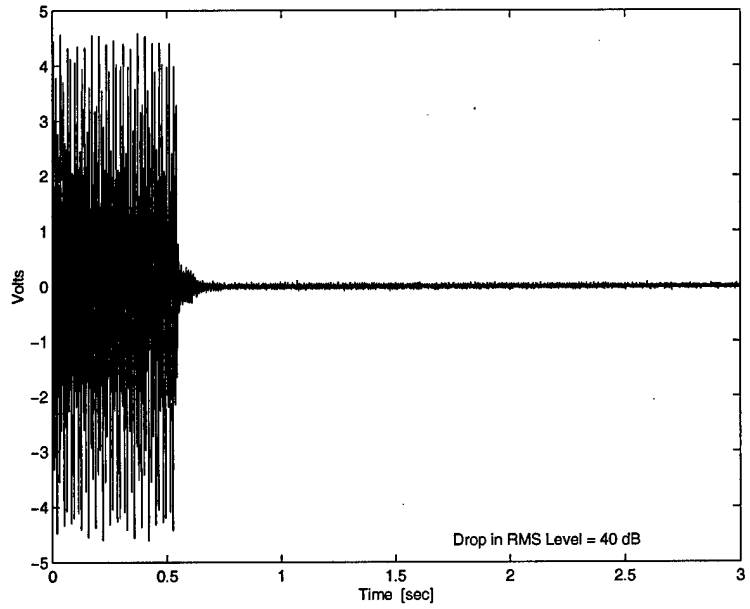

Figure V-39: Sine/Cosine Method Sensor Outputs, 2 Static Disturbances

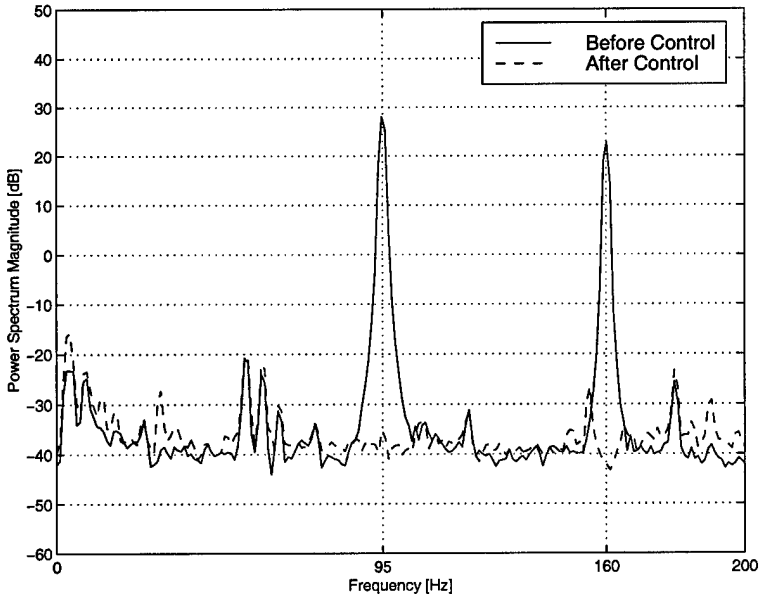

Figure V-40: Sine/Cosine Method Spectral Comparison, 2 Static Disturbances

For the case of harmonic disturbances, the Clear Box Algorithm (either method) has an advantage over the Multiple Error LMS algorithm since it uses the calculated disturbance effect signal. In calculating the disturbance effect signal, the controller essentially estimates the outputs that are not caused by the known actuator inputs. As such, the fundamental disturbance *and* its harmonics are present in the disturbance effect signal $\eta(k)$. The Sine/Cosine Method estimates the disturbance frequencies based on the $\eta(k)$ signal, and controls each one as if it were a separate disturbance. The before/after spectral comparison (again using the strut #1 sensor) in Figure V-41 clearly indicates complete control of all three disturbances.

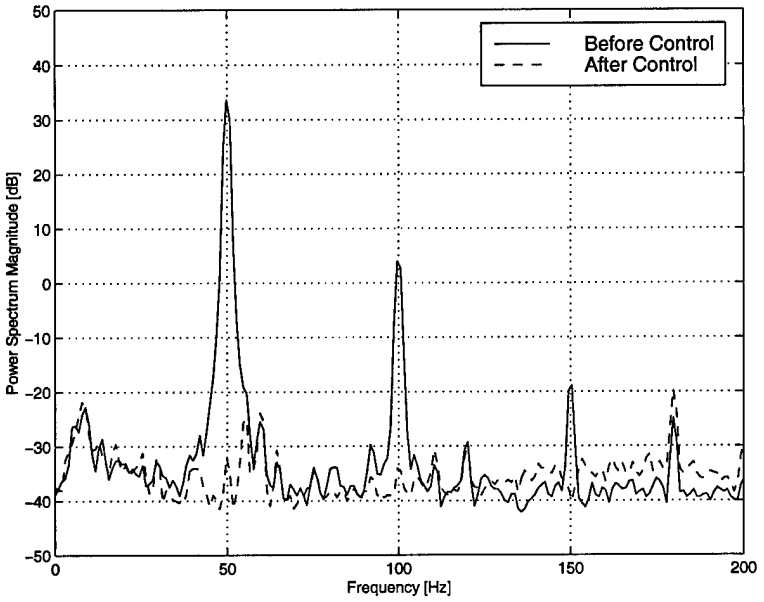

Figure V-41: Spectral Comparison, Sine/Cosine Method Controlling Harmonics

c) Clear Box, Adaptive Basis Method

The same two cases of multiple frequency disturbances are tested using the Adaptive Basis Method of the Clear Box algorithm. For the case of two distinct disturbances the six sensor outputs are shown in Figure V-42, and the before/after spectral content of the strut #1 sensor is shown in Figure V-43. Note there is still a small amount of energy at the two disturbance frequencies. This is likely due to the fact that there are not as many "redundant shifts" available since there are now two disturbances present (discussed further in Chapter VI). As in the single, static frequency case the use of a forgetting factor equal to 1.0 helps improve performance when the disturbance frequencies are not varying with time.

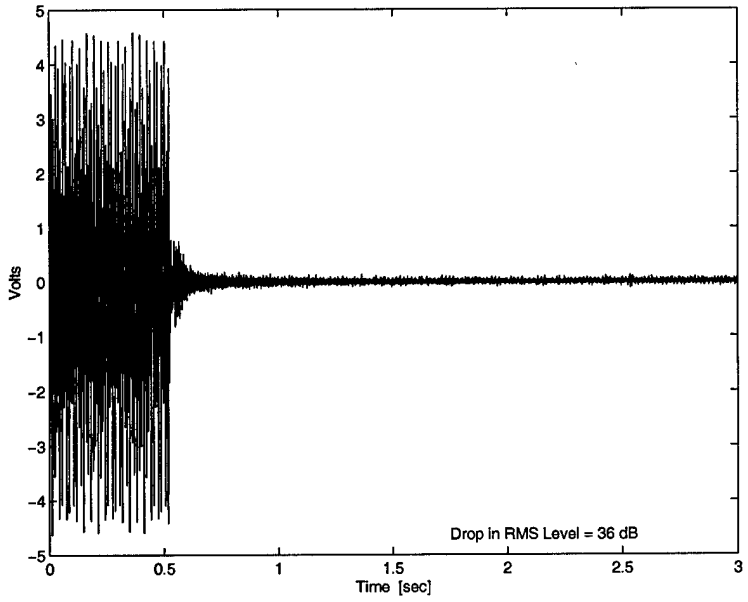

Figure V-42: Adaptive Basis Method Sensor Outputs, 2 Static Disturbances

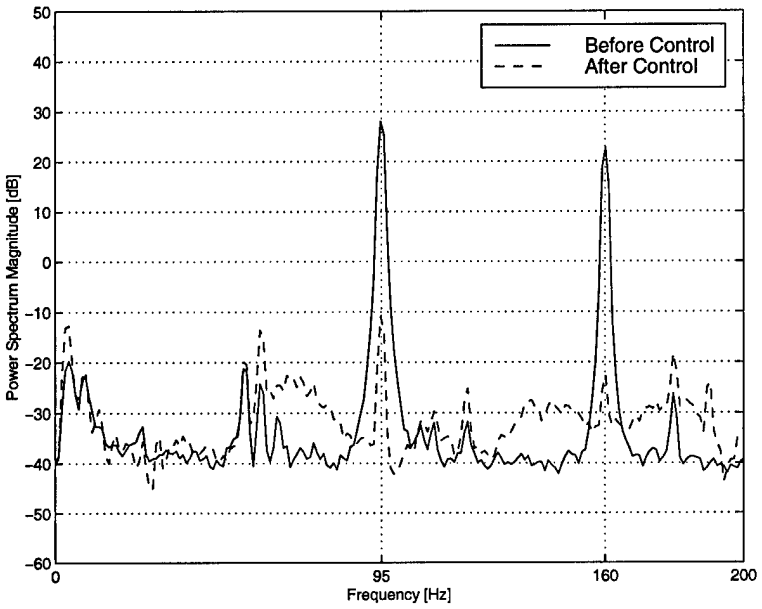

Figure V-43: Adaptive Basis Method Spectral Comparison, 2 Static Disturbances

The case of harmonic disturbances was also tested using the Adaptive Basis Method. In this case a fundamental frequency of 105 Hz is chosen, which generates harmonics at 210 Hz and 315 Hz. The 210 Hz disturbance is not very strong, and in fact the controller causes slight amplification at this frequency, but there is clear reduction of the energy present in the fundamental disturbance and the harmonic at 315 Hz, as seen in Figure V-44.

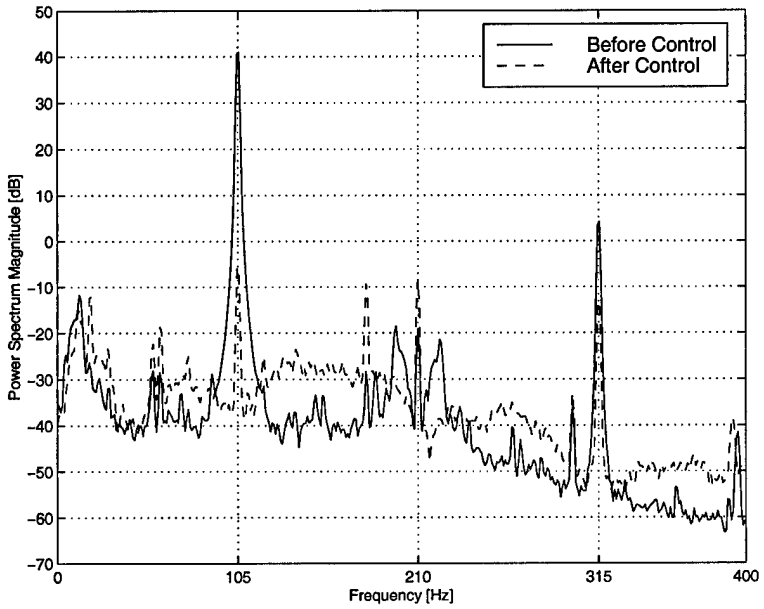

Figure V-44: Spectral Comparison, Adaptive Basis Method Controlling Harmonics

D. REJECTING TIME-VARYING DISTURBANCES

Now that all three controllers have been tested against disturbances that do not vary with time, we move our attention to the case of time-varying disturbances.

1. Single, Time-Varying Disturbance Frequency

To test the controllers against a single, time-varying disturbance frequency a standard frequency variation profile is used. This profile, shown in Figure V-45, starts at 120 Hz and is held constant for 1 second. Then the frequency ramps up at a "rapid" rate of 2 Hz/sec for four seconds, and is held constant at 128 Hz for one second. Then the frequency ramps down at a "slow" 0.1 Hz/sec for four more seconds, and is finally held

constant at 127.6 Hz for two seconds to complete the experiment. In all cases the controller is activated at 0.5 seconds.

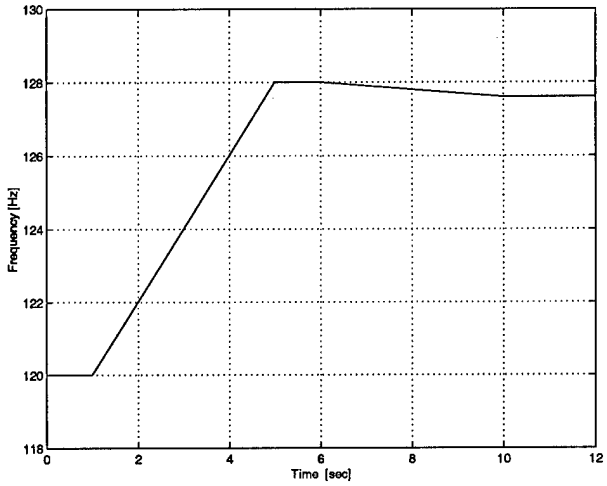

Figure V-45: Frequency Variation Profile, Single Disturbance

a) Multiple Error LMS

With knowledge of the exact disturbance frequency, provided by the $x(k)$ signal, the Multiple Error LMS algorithm performs quite well over the course of the frequency variation profile, as shown in Figure V-46. Only a slight drop in performance is noted during the rapid frequency variation phase.

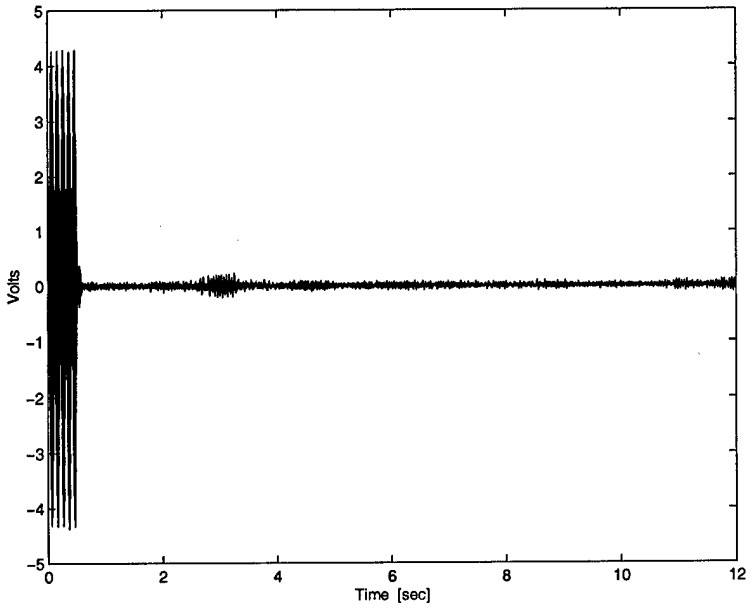

Figure V-46: Multiple Error LMS Sensor Outputs, 1 Varying Frequency

b) Clear Box, Sine/Cosine Method

The Sine/Cosine controller's performance during the frequency variation profile is shown in Figure V-47. In the UQP implementation of the Sine/Cosine Method the disturbance frequencies are updated once per second using batch processing of the most recent $\eta(k)$ time history. Thus, as the true disturbance frequency strays from the estimated frequency the performance begins to degrade. The coefficient cycling, shown in Figure V-48 for the strut #1 coefficients, is unable to compensate for the frequency variation during the rapid ramp-up phase. In Section C.1.b) starting on page 106, it was shown how the use of a smaller forgetting factor improves performance when the frequency estimate is incorrect. Unfortunately, use of a forgetting factor below 0.9

during the frequency variation profile resulted in excessive noise in the control coefficients and led to instability. The results in the two figures below are obtained using a forgetting factor of 0.95.

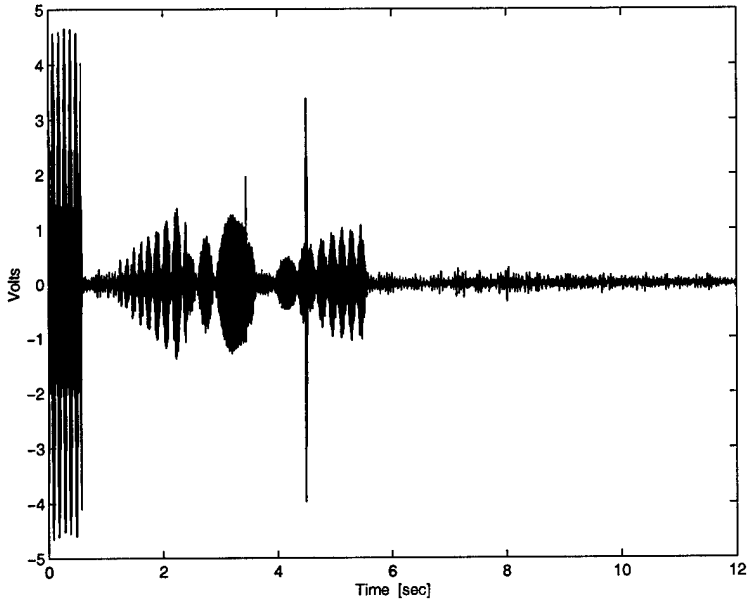

Figure V-47: Sine/Cosine Method Sensor Outputs, 1 Varying Frequency

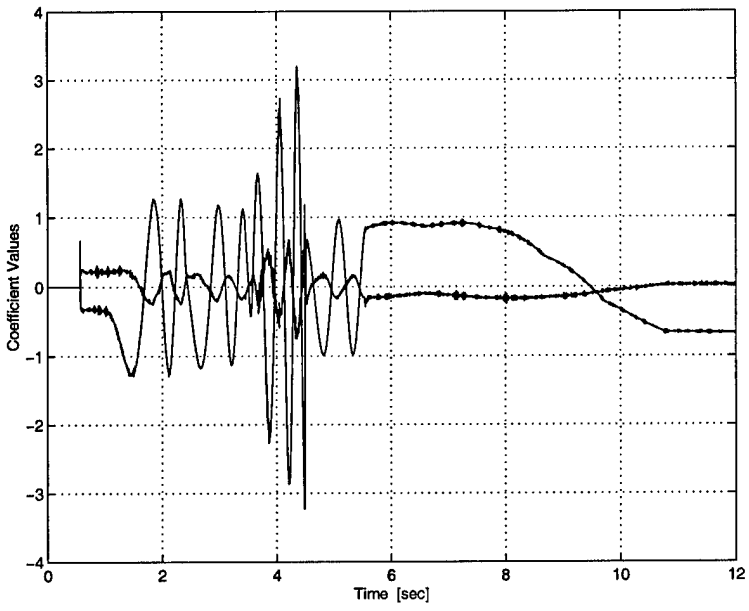

Figure V-48: Sine/Cosine Method Strut #1 Coefficients, 1 Varying Frequency

c) Clear Box, Adaptive Basis Method

Finally, the frequency variation profile is used while controlling the UQP with the Adaptive Basis Method. Again, this controller does not need to estimate the disturbance frequencies since the information is present in the $\eta(k)$ signal. The performance is very good during the entire profile, as shown in Figure V-49. Note that this performance is obtained without the use of a sensor-provided reference signal.

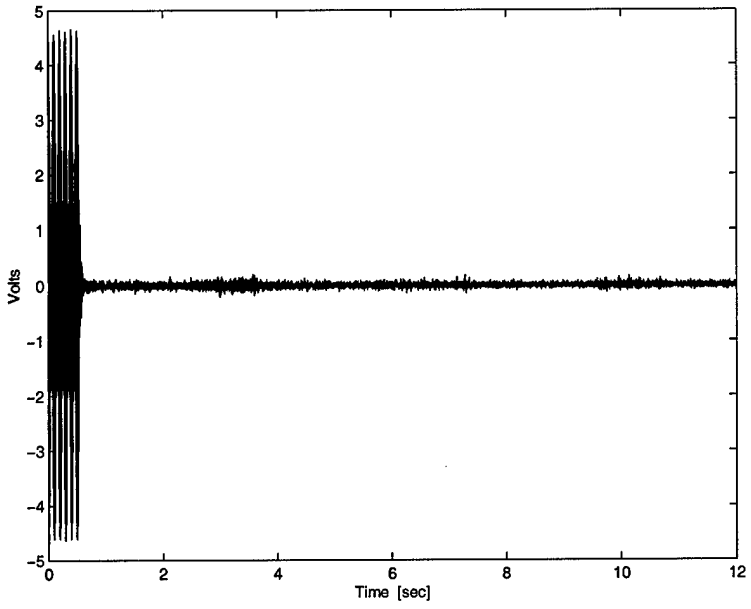

Figure V-49: Adaptive Basis Method Sensor Outputs, 1 Varying Frequency

2. Multiple Time-Varying Disturbance Frequencies

The next set of experiments involves the use of two time-varying frequencies. The variation profile is shown in Figure V-50. The first frequency starts at 140 Hz and ramps up at a rate of 0.5 Hz/sec. The second starts at 144 Hz and ramps down at 0.5 Hz/sec. The two frequencies "converge" at 142 Hz at four seconds into the experiment (which lasts for a total of nine seconds). The two frequencies combine to form a disturbance that pulsates; more slowly as the frequencies converge, and more rapidly as they diverge. When they cross at 142 Hz they appear to be a single disturbance.

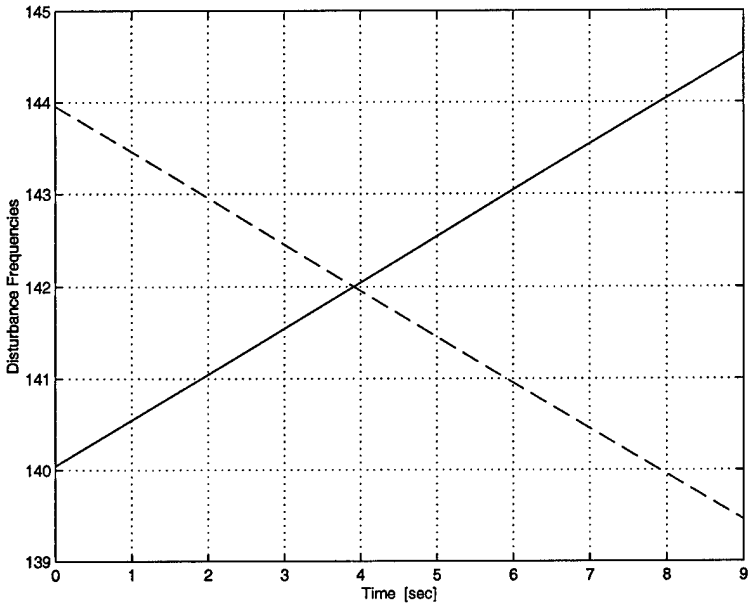

Figure V-50: Frequency Variation Profile, 2 Varying Disturbances

a) Multiple Error LMS

The Multiple Error LMS algorithm is tested first against the two time-varying disturbances, and the six sensor outputs and the strut #1 control coefficients are shown in Figure V-51 and Figure V-52, respectively. The pulsating nature of the disturbances results in short periods of degraded performance where the coefficients cannot adapt quickly enough to keep performance at an optimum. The adaptation rate used for this experiment is 0.01, which can be tuned to a more aggressive value for better performance, but at the risk of instability. Recall from earlier experiments that higher frequency disturbances cause larger $\bar{r}^2$ levels, resulting in instability at lower values of the adaptation rate.

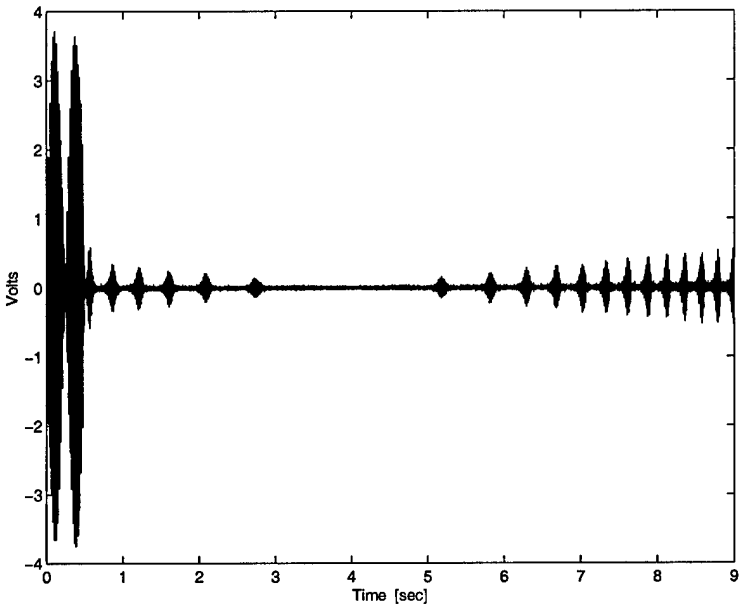

Figure V-51: Multiple Error LMS Sensor Outputs, 2 Varying Disturbances

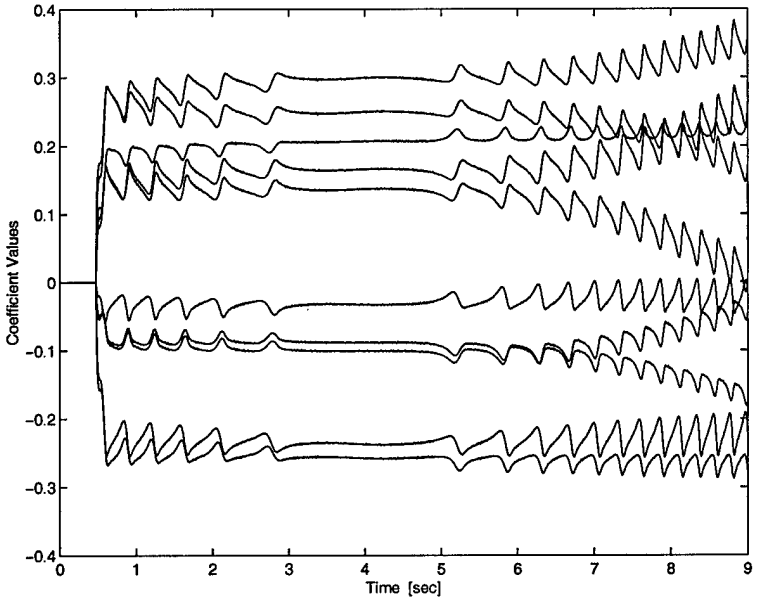

Figure V-52: Multiple Error LMS Control Coefficients, 2 Varying Disturbances

b) Clear Box, Sine/Cosine Method

Next, the Sine/Cosine Method is tested against the two time-varying disturbances. The resulting sensor outputs and control coefficient histories are shown in Figure V-53 and Figure V-54, respectively, and show that the controller has difficulty maintaining optimal performance. The forgetting factor is again key in improving performance since the true frequencies are constantly drifting from the frequency estimates (updated once per second). In this experiment a forgetting factor of 0.99 is used. A lower value improves performance but instability is observed for values below 0.95. During the period from 5.5 to 6.5 seconds the Host PC frequency estimation code is unable to detect a mode in the disturbance effect signal that has a damping ratio below the pre-set threshold, resulting in all control coefficients being set to zero, and performance returning to an uncontrolled state.

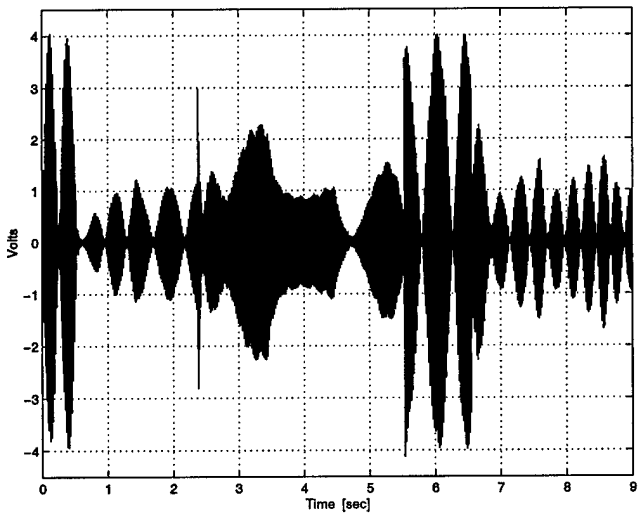

Figure V-53: Sine/Cosine Method Sensor Outputs, 2 Varying Disturbances

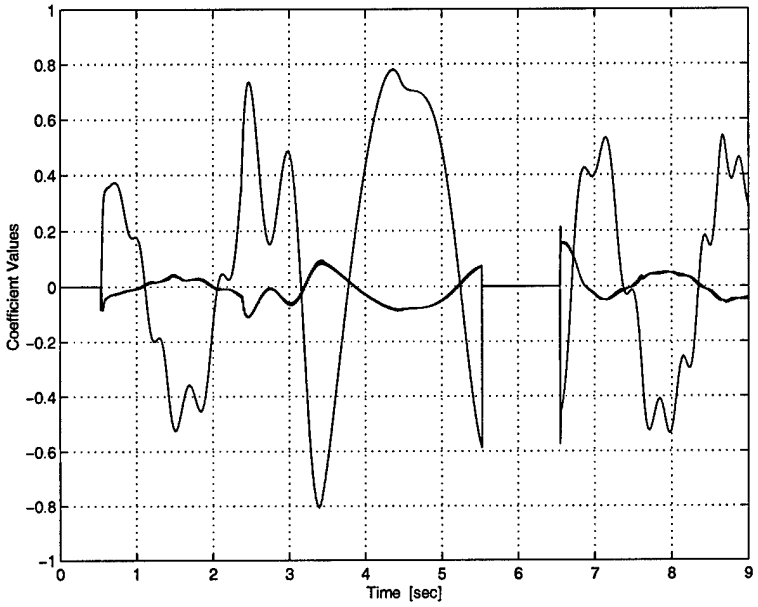

Figure V-54: Sine/Cosine Method Control Coefficients, 2 Varying Disturbances

c) Clear Box, Adaptive Basis Method

Finally, the Adaptive Basis Method is tested against the two time-varying disturbances. The six sensor outputs and the strut #1 control coefficients are shown in Figure V-55 and Figure V-56, respectively, and show relatively good performance with a pulsating nature similar to that observed with the Multiple Error LMS controller. The forgetting factor used in this experiment is 0.99 (the same as the Sine/Cosine Method's experiment). Performance is not affected greatly by using a lower value, and stability is also not affected by a lower forgetting factor, in contrast to the Sine/Cosine Method which is sporadically unstable for values below 0.95.

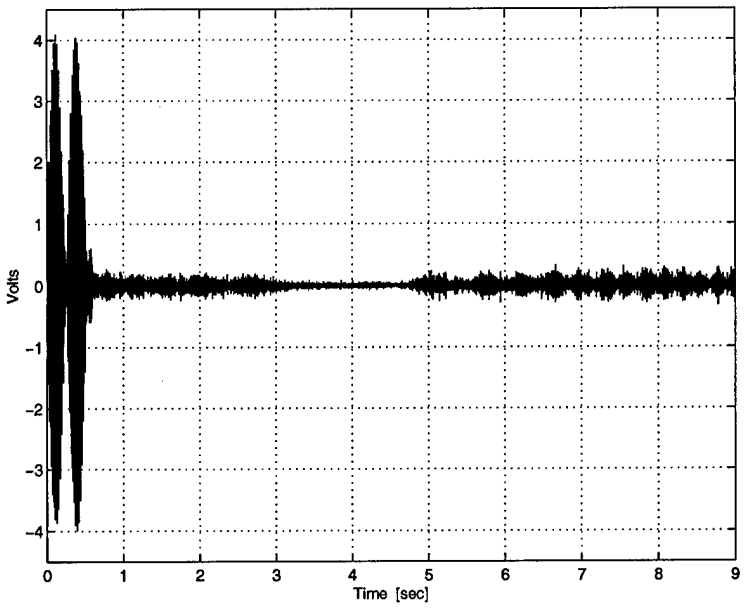

Figure V-55: Adaptive Basis Method Sensor Outputs, 2 Varying Disturbances

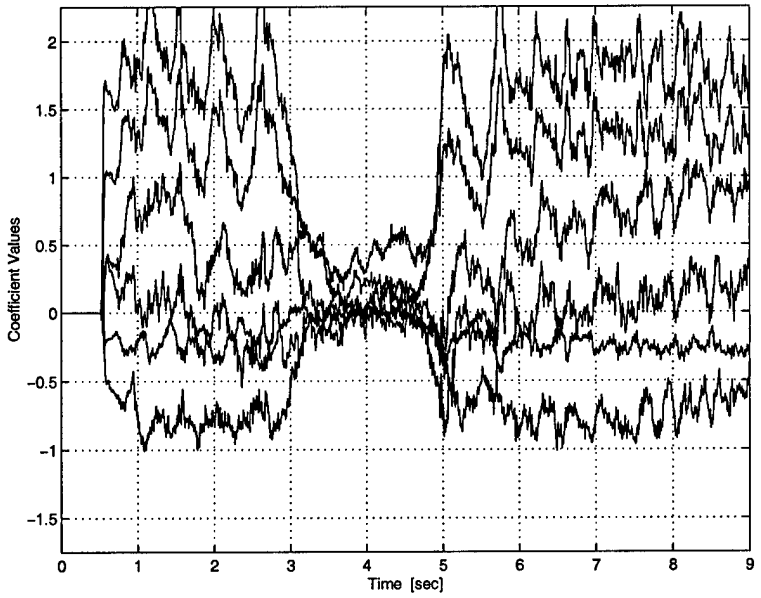

Figure V-56: Adaptive Basis Method Control Coefficients, 2 Varying Disturbances

E. THE CASE OF AN UNCONTROLLABLE MODE

One of the primary advantages of the Clear Box algorithm is the ability to selectively control disturbances. This is important in the case where it is undesirable to attempt control of an uncontrollable or weakly controllable mode, since actuator saturation and instability may result. The Multiple Error LMS algorithm is, in general, not capable of selective disturbance control.

The experiments in this section show that both methods of the Clear Box algorithm allow selective disturbance control. The capability is inherent in the Sine/Cosine Method, but is also shown with the Adaptive Basis Method through the use of filtering of the disturbance effect signal.

1. Clear Box, Sine/Cosine Method

The experiment used to demonstrate selective disturbance control employs two static disturbances at 110 Hz and 140 Hz. Since the highly damped UQP system does not have any modes that are "uncontrollable" in the frequency range of interest, it is necessary to artificially designate the 140 Hz disturbance as being uncontrollable.

The results are shown in the three figures that follow. Figure V-57 shows The sensor outputs for all six struts. Before the controller is initiated the outputs contain the effects of both disturbances. In the disturbance identification portion of the code, logic is implemented that "de-selects" any identified disturbance that is close to 140 Hz. This approach is valid for a system with time-invariant dynamics, but in general (for systems with time-varying dynamics) the magnitude of the required control action can be

analyzed to determine if a particular disturbance is, in fact, "uncontrollable". Figure V-58 simply shows a close up view of the sensor outputs, showing that the signal that remains is a 140 Hz signal. The before/after spectral comparison is shown in Figure V-59, indicating control action effectively removes the disturbance at 110 Hz, but does nothing at 140 Hz. There are no significant adverse effects at any other frequencies.

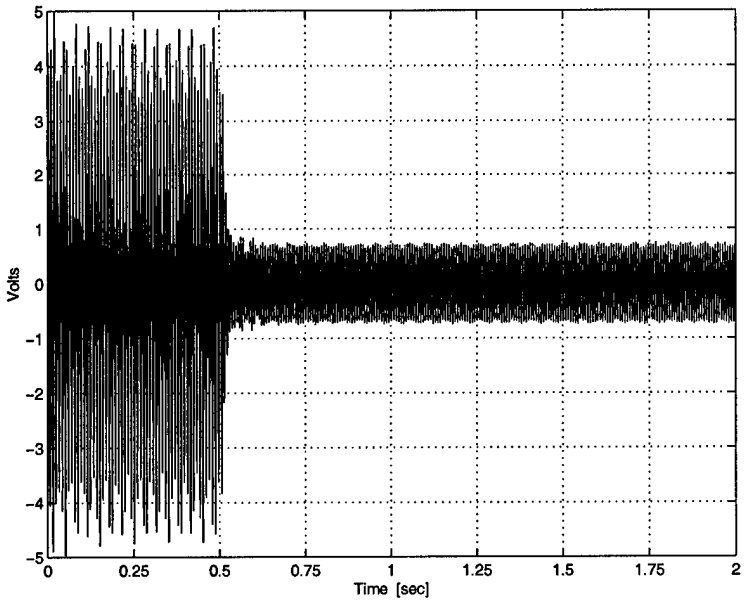

Figure V-57: Sine/Cosine Method Sensor Outputs, Selective Disturbance Control

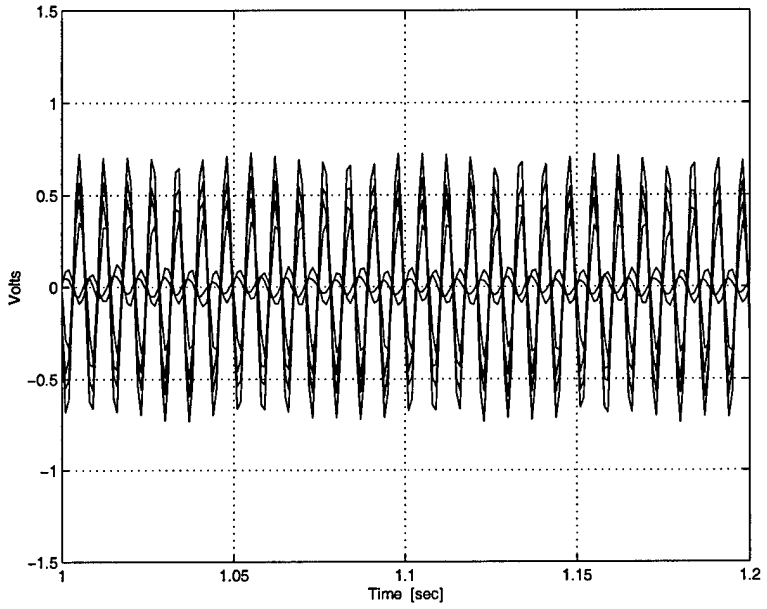

Figure V-58: Sine/Cosine Method Sensor Outputs (Zoom in)

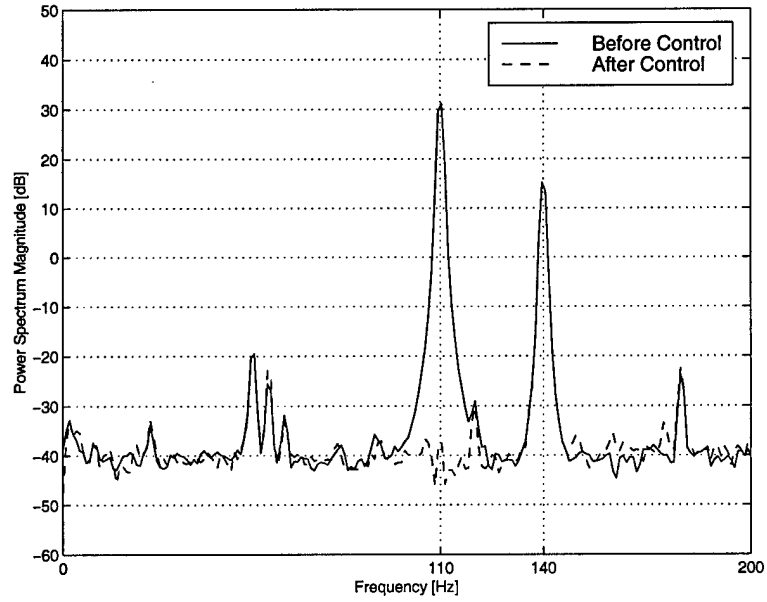

Figure V-59: Sine/Cosine Method Spectral Comparison, Selective Control

2. Clear Box, Adaptive Basis Method

The same experiment described above for the Sine/Cosine Method is tested using the Adaptive Basis Method. A digital filter, with frequency response shown in Figure V-60, is employed to filter the disturbance effect signal, and effectively eliminate the information at the 140 Hz frequency. Thus, the time shifted disturbance effect basis functions are unable to control the disturbance at 140 Hz. The results are shown in the three figures that follow. The before/after spectral comparison in Figure V-63 shows effective control of the 110 Hz disturbance, but no control of the 140 Hz disturbance. Some minor effects at other frequencies are noted.

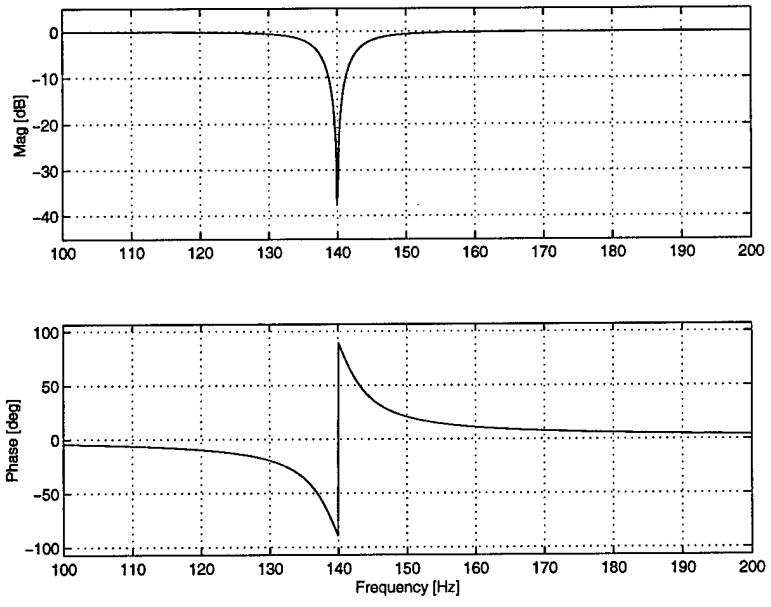

Figure V-60: 2nd Order Butterworth Notch Filter for Selective Control

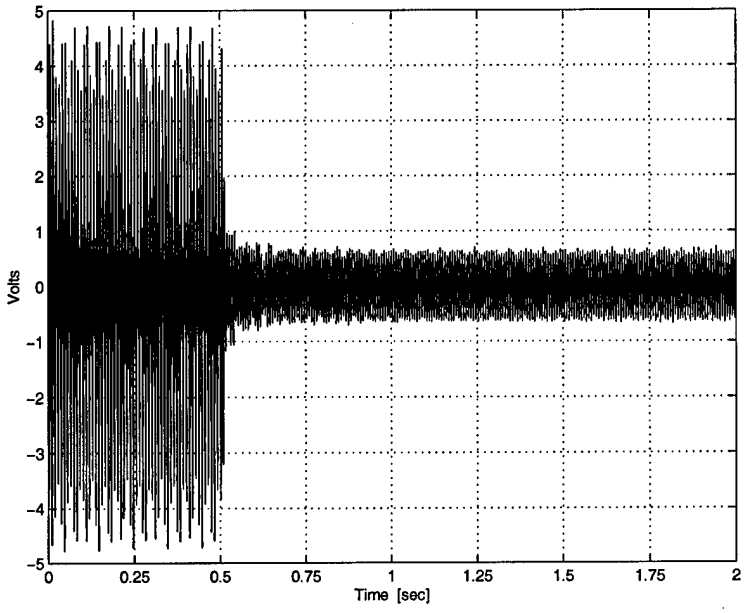

Figure V-61: Adaptive Basis Method Sensor Outputs, Selective Disturbance Control

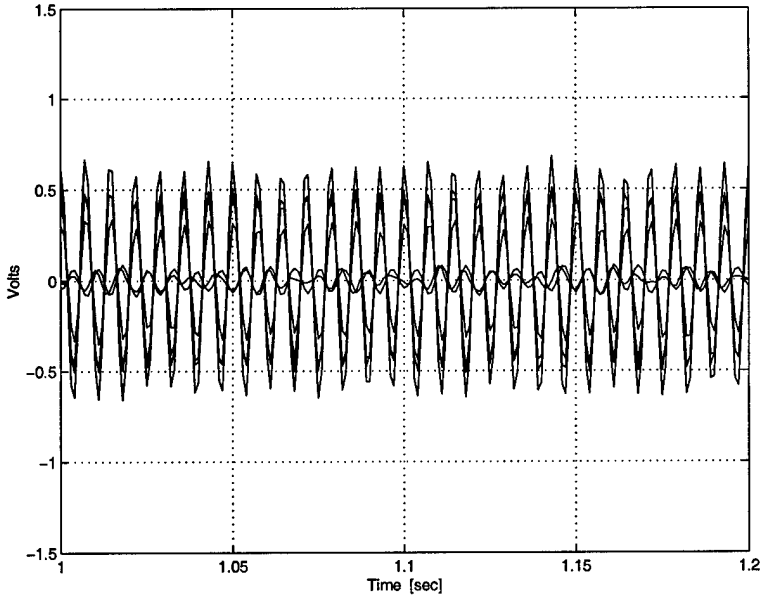

Figure V-62: Adaptive Basis Method Sensor Outputs (Zoom in)

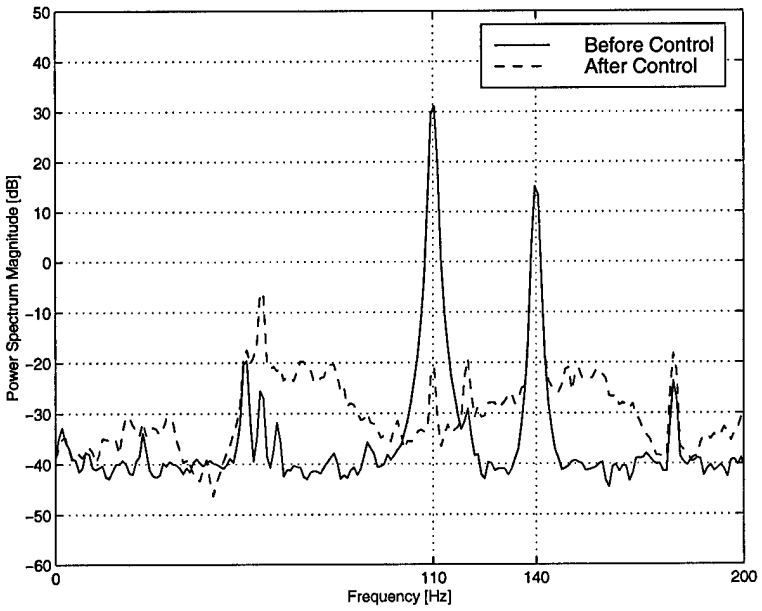

Figure V-63: Adaptive Basis Method Spectral Comparison, Selective Control

VI. DISCUSSION OF RESULTS

In this chapter the performance characteristics of the Multiple Error LMS, Clear Box Sine/Cosine, and Clear Box Adaptive Basis controllers are compared across the various experiments performed in Chapter V. The comparisons are divided into the two broad categories of "performance" and "implementation issues".

A. PERFORMANCE

1. General

The experiments that demonstrate control of a single static disturbance (conducted in Chapter V) reveal several key characteristics of the three controllers. All three are able to control the disturbance to the noise level across the entire selected control bandwidth (20-300 Hz). The three controllers also demonstrate that they impart very little energy to the system at frequencies other than the disturbance frequency. This is especially true in the case of the Sine/Cosine Method since the control signal consists of pure sine & cosine functions at the estimated disturbance frequencies.

The experiments with two static disturbances reveal that the Multiple Error LMS Algorithm and the Clear Box Sine/Cosine Method achieve slightly better performance than the Clear Box Adaptive Basis Method. The Adaptive Basis Method increases the energy at frequencies other than the disturbance frequency (Figure V-43), indicating that these unintended frequency components are present in the disturbance effect signal and

thus in the basis functions. The available processing power only allows a maximum of $N = 6$ when using the Adaptive Basis Method, which limits control to three frequencies (or two frequencies and an offset bias). The existence of additional low-level frequency components is the most likely reason the controller is unable to completely cancel the two static disturbances. A 36 dB drop is achieved, in comparison with the 40 dB drop of the Sine/Cosine Method.

An advantage of the Clear Box Algorithm is the inherent ability to control unexpected disturbances and harmonics. Again, the disturbance effect signal reveals all system outputs that are not predicted by the past input/output data as applied to the disturbance-free system model (Eq.s (2-60) and (2-62)). These disturbances are controlled without difficulty since the two Clear Box methods use the disturbance effect signal as either a source for the disturbance frequency estimates (Sine/Cosine Method) or as a basis function source (Adaptive Basis Method). In contrast, controlling harmonics with the Multiple Error LMS Algorithm requires prior knowledge of the number of harmonics that need control. Periodic signals at the harmonic frequencies can then be artificially generated and added to the original reference signal (from which the frequency information is obtained). This process is computationally inefficient since not all disturbance frequencies have harmonics that significantly affect the system output. In addition, unexpected disturbances may not be controllable with the Multiple Error LMS Algorithm since the pre-selected sensor locations may yield a very weak signal.

Experiments controlling time-varying disturbance frequencies reveal that the Clear Box Sine/Cosine Method is unable to control rapidly changing disturbances.

Although coefficient cycling allows some improvement from the uncontrolled condition, excessive rates of change cannot be compensated for. Typically the disturbances present on spacecraft are not varying rapidly with time, with the exception of momentum wheels that often change rates rapidly during slew or momentum dumping maneuvers. However, the general applicability of the Clear Box Algorithm is improved if it can control rapidly changing disturbances. The Adaptive Basis Method addresses this need, as shown in the time-varying disturbance experiments in Chapter V. An alternative approach is to improve the Sine/Cosine Method's ability to estimate frequencies accurately (discussed in Section II.D) or increase the rate with which the updated frequencies are provided, or a combination of both.

Selective disturbance control is a feature unique to the Clear Box Algorithm, and is demonstrated on a hypothetical "uncontrollable mode" in Chapter V using both Clear Box methods. The Multiple Error LMS Algorithm does not provide for selective mode control. The only possible modification that would allow a degree of selective control would be to implement a comb filter for the reference signal $x(k)$ (frequencies that are de-selected for control are filtered out of $x(k)$). This implies adding some form of system identification capability to the LMS control implementation. Even so, the only way to determine which frequencies are weakly controllable is by finding the system transmission zeros. There is no capability to determine what portion of the control signal is due to a particular frequency, or how much of the system output is due to a particular disturbance. Without this information the disturbance control selections cannot be made as intelligently as with Clear Box.

2. Tuning Requirements

A drawback of the Multiple Error LMS Algorithm is that tuning of the adaptation rate is required to achieve optimal convergence to the steady-state filter weight solution. For a given adaptation rate, instability is more likely as the disturbance frequency increases, and convergence is slower as the disturbance frequency decreases. The adaptation rate must be chosen conservatively (i.e., a low rate) to guarantee that stability is maintained, and thus rapid convergence is not possible for every disturbance.

By comparison, the Clear Box methods require relatively little tuning. The forgetting factor's effect on performance has no dependence on the disturbance frequency. For the Sine/Cosine Method a forgetting factor of 0.95 yields stable performance (even with rapidly varying disturbances) and reduction to the noise level when typical frequency estimation errors are present. The Adaptive Basis Method performs well in all cases (static or varying disturbances) when a forgetting factor of 0.99 is used.

3. Processing Requirements

A quantitative comparison of the processing required for the investigated algorithms reveals the relative efficiency of each. There is no way to directly compare the Multiple Error LMS Algorithm to the Clear Box Algorithm since one uses an FIR filter model and the other uses an ARX model. Although there is no way to use the same "model order", it is revealing to note that using $p = 10$ for the Clear Box Algorithm, and $J = 20$ for the Multiple Error LMS Algorithm both yield system models with 720

coefficients (for 6 inputs and 6 outputs). This may be misleading for lightly-damped systems, however, since the FIR model order would need to be much larger to match the accuracy that is possible with a relatively low order ARX model.

The tables below indicate the percentage of available processing power used by each algorithm for various model orders, and employing different numbers of filter weights or control coefficients. The cases that offer an approximate comparison (assuming a well-damped system) are circled.

		Number of Filter Weights per strut, *I*		
		5	10	15
FIR Model Order, *J*	20	(17.1)	31.7	43.4
	30	23.6	44.2	61.3
	40	30.5	57.4	80.0

Table VI-1: Multiple Error LMS Algorithm, Processing Required (%)

		Number of Frequencies Controlled, f		
		1	2	3
Model Order, p	10	19.3	41.1	(72.3)
	15	22.5	45.8	79.0
	20	25.3	50.5	85.0

Table VI-2: Clear Box Sine/Cosine Method, Processing Required (%)

		Number of Basis Functions, N		
		2	4	6
Model Order, p	10	21.2	44.2	(77.1)
	15	24.6	51.5	84.3
	20	27.3	55.8	92.9

Table VI-3: Clear Box Adaptive Basis Method, Processing Required (%)

4. Stability

The experiments performed in Chapter V demonstrate that the three algorithms are sufficiently stable while operating under normal conditions, and adapt to situations where the plant model is altered by the addition of mass to the top of the platform. Impulsive disturbances to the platform are handled very well by the Multiple Error LMS

Algorithm, and also by the Clear Box Sine/Cosine Method, with coefficients rapidly returning to pre-impulse values. While the Clear Box Adaptive Basis Method can handle small impulsive disturbances, very large disturbances tend to cause instability. The immediate presence of the disturbance information in the basis functions of the controller results in an attempt by the control coefficients to cancel the disturbance, and the PZT actuators are quickly saturated due to their small displacement capability (50 microns).

The extremely nonlinear nature of the Adaptive Basis Method hinders analytical proof of the algorithm's stability for cases other than single disturbance frequencies with slow adaptation on SISO systems. The Clear Box Sine/Cosine Method was shown to be stable under such assumptions in the presence of small perturbations in the system model parameters [Ref. 108], and has an effective phase margin of ±90 degrees. The single channel Filtered-x LMS Algorithm has been shown to be stable theoretically (also with a ±90 degree phase margin) [Ref.s 109, 110], while the Multiple Error LMS Algorithm has only been theoretically proven stable for restricted disturbance cases [Ref.s 111,112]. Despite this lack of proven stability (for the general case) adaptive feedforward algorithms have been successfully applied to many applications with good observed stability, even using rapid adaptation rates.

B. IMPLEMENTATION ISSUES

The UQP system is a highly damped system with dynamics that do not vary significantly with time. Implementing adaptive feedforward control for a different application first requires a determination of several characteristics of the system and the

expected disturbances. The factors affecting controller selection for a given application include; 1) the nature of the system dynamics, 2) the nature of the disturbances, 3) the availability of processing resources, 4) the cost associated with additional sensors, 5) the control authority of the selected actuators, and 6) the importance of reliability and fault tolerance.

1. System Dynamics Considerations

If the system dynamics (from actuator input to sensor output) vary with time, then periodic re-identification is required to maintain optimal performance. The system identification approach inherent in the Clear Box Algorithm is well-suited to such systems, allowing operation without any prior knowledge of the nature of the physical plant.

If the system is lightly damped then the required model order is much less when using an ARX model (as with Clear Box) than with an FIR filter model (as with LMS algorithms). A lower model order, in general, translates into reduced processing requirements. Typically, lightly damped systems have transmission zeros that translate into weakly controllable modes. The required control signal at these frequencies is very large since the low gain of the system must be compensated for. Since it is desirable to use light weight (generally less capable) actuators for space applications, a large control signal will likely cause actuator saturation, leading to possible system instability. The selective control capability unique to the Clear Box Algorithm is indispensable in such cases. Equations (2-66) and (2-85) are used to determine the magnitude of the required

control signal, and the effect of each disturbance on the system output, respectively. If control authority is limited, logic statements are used to de-select disturbances for control if the magnitude of the required control signal is too large (thus preventing actuator saturation) or if the impact on the system output is very small (preventing inefficient use of processor resources on insignificant disturbances).

2. Disturbance Considerations

If it is impossible (or undesirable) to turn off disturbance sources when performing system identification the resulting model will be disturbance-corrupted. In such cases the Clear Box Algorithm is the only alternative for obtaining a disturbance-free system model. In fact, in 50% of all cases, finite length data records result in identified disturbance modes having slightly negative damping ratios. This results in the identified model being unstable, and unusable for control.

For the case of a system with time-invariant dynamics (such as the UQP), periodic disturbances that have slowly varying frequencies (i.e., change less than 0.1 Hz per second) can be controlled equally well by the three controllers implemented in this research. A slight edge is given to the Clear Box Sine/Cosine Method due to its complete lack of negative impact at frequencies other than those of the controlled disturbances.

Rapidly varying disturbances are more effectively controlled by the methods that do not require estimates of the disturbance frequencies (the Multiple Error LMS Algorithm and the new Clear Box Adaptive Basis Method). However, the performance of the Sine/Cosine Method against rapidly time-varying disturbances could be greatly

improved by implementing a polynomial curve fit for the time-history of each disturbance's frequency. This would allow a more accurate estimate of the frequency at the time steps between frequency updates. More efficient processing techniques would also shorten the time required between updates.

3. Coding and Processing

The Multiple Error LMS Algorithm requires the least amount of coding, mainly due to the fact that it does not require recursive estimation of its control parameters. Instead it uses a simple filter weight update consisting of an adaptation rate "gain", and the product of the error signals and the filtered reference signal, as shown in Eq. (2-22). The result is that the Multiple Error LMS Algorithm requires the least amount of processing power.

The computational requirements of the two Clear Box methods are similar, but there are some differences worthy of discussion. The number of basis functions used in the Sine/Cosine Method is always twice the number of disturbance frequencies that have been identified and selected for control, since a sine/cosine pair is assigned to each estimated disturbance frequency. Thus, if new disturbances emerge the required processing power will increase as sine/cosine pairs are added. For the Adaptive Basis Method, the number of basis functions N is selected ahead of time, so an upper bound on the number of disturbance frequencies f is required to assess the correct value of N according to Eq. (2-87). For this reason, the processing requirements of the Adaptive Basis Method are constant, and do not depend on the number of disturbances present.

The Sine/Cosine Method is the only controller (of the three) that requires estimates of the disturbance frequencies (a process that must be done in batch mode using modal decomposition and logic statements). This process, performed by the host PC during the UQP experiments, represents additional computation required for this method.

For systems with time-varying dynamics, or to add fault tolerance to the system, re-identification of the system model is required periodically. To perform disturbance-free system identification using disturbance-corrupted data (via the Clear Box Algorithm) requires selective elimination of disturbance modes using logic statements, which must also be done in batch mode using an additional processor. Again, this function is performed by the host PC for the UQP experiment.

As space qualified processors become more capable (and require less power, weight, and volume) the additional processing burden required by the Clear Box Algorithm will be less of a consideration. The added capabilities (identification in the presence of disturbances, control of time-varying systems, and selective mode control) represent features not included in the Multiple Error LMS Algorithm. Thus the added processing required by the Clear Box Algorithm is not without added benefit.

4. Cost Associated with Additional Sensors

A primary advantage of the Clear Box Algorithm is the ability to control any periodic disturbance (including unexpected disturbances and harmonics) without requiring an additional sensor to provide a disturbance-correlated signal. Addition of sensors to provide this signal for the Multiple Error LMS Algorithm is accompanied by

the addition of power supplies, signal conditioning, wiring, a signal combiner (to provide a single disturbance signal from all of the sensors) and mounting hardware. Also to be considered are the additional weight, volume, and integration & test costs associated with these components.

5. Reliability and Fault Tolerance

The ability to rapidly re-accomplish system identification is important for handling unexpected situations such as a mechanical failure of the system. For example, if the UQP experienced a strut actuator failure it is possible the system model for the remaining struts would be affected. Since the Clear Box Algorithm requires no knowledge of the system dynamics, the next system identification performed on the system would account for any changes induced by the failure. For the case of the UQP the strut with the failed actuator would still offer passive isolation capability. In this manner, the system identification approach to the control problem adds a degree of fault tolerance to the controller that is not present with the LMS Algorithm.

VII. CONCLUSIONS

The Multiple Error LMS Algorithm is widely accepted for use with MIMO systems that require vibration control/disturbance rejection. However, the need to supply the algorithm with a disturbance correlated signal is undesirable for space applications since additional sensor hardware and wiring is required. Also, traditional methods allow no capability to selectively assign control action to specific disturbances, and thus the actuators must be sized to handle control of all expected disturbances simultaneously.

The Clear Box Algorithm approaches active disturbance rejection from a system identification perspective, allowing intelligent operation in an information-rich environment. The algorithm can handle systems with time-varying dynamics, does not require a measured disturbance-correlated signal, and can handle unanticipated disturbances and harmonics. The selective disturbance control feature of the Clear Box Algorithm allows intelligent assignment of actuator resources based on available information about the size of the required control signal and the impact of each disturbance on the system output. Thus, smaller light-weight actuators can be used since selective control can prevent actuator saturation. Fault tolerance and adaptability are enhanced by the Clear Box Algorithm through the use of frequent system identification.

The Clear Box Sine/Cosine Method requires estimates of the disturbance frequencies, and performance suffers if the frequencies drift significantly before the next update is available. Efficient processing techniques can reduce the time between updates,

but eliminating the need for frequency estimation is preferable. A new Adaptive Basis Method is presented that uses the disturbance effect signal directly as a computed basis for the control signal. This new method addresses the problem of controlling rapidly varying frequencies (since frequency estimates are not required), and maintains the attractive features associated with the Clear Box Algorithm.

Extensive experiments employing the three control techniques on the Ultra Quiet Platform demonstrate the attractiveness of Clear Box methods for active vibration isolation and disturbance rejection. Performance meets or exceeds that of the Multiple Error LMS Algorithm while requiring neither prior knowledge of the system dynamics nor a measured disturbance-correlated signal.

A. CONTRIBUTIONS

The Multiple Error LMS algorithm has been demonstrated in many different applications of noise and vibration control. The Clear Box algorithm, however, has only been implemented using the Sine/Cosine Method for purposes of controlling structural vibrations on a truss [Ref. 113]. Thus, this is the first application of Clear Box on a vibration isolation platform, and the first experimental implementation using greater than two inputs and two outputs. The utility of Clear Box for spacecraft vibration isolation comes from eliminating the requirement for additional sensors, allowing the use of lighter weight actuators, and adding a greater degree of fault tolerance and adaptability by employing frequent system identification.

The Sine/Cosine Method's coefficient cycling (when frequency estimates are incorrect) had been observed, but not explained. The theoretical prediction of this cycling is offered for the first time in this dissertation.

The Adaptive Basis Method was developed during the course of this research. It has the capability to control rapidly varying disturbances to a degree not currently possible with the Sine/Cosine Method, and is still capable of selective disturbance control. Thus, all of the attractive features of the Clear Box Algorithm are retained while adding an improved capability to control rapidly varying frequencies.

The data acquisition and control system for the UQP experiment was also implemented during the course of this research. All coding was developed so that the user can easily change the number of inputs, outputs, and sample rate. Thus, either the LMS or Clear Box controllers could be easily adapted and applied to other experiments. The equipment remains in the Spacecraft R&D Center's Smart Structures Laboratory at the Naval Postgraduate School for follow-on work.

B. RECOMMENDATIONS FOR FURTHER STUDY

The effectiveness of the Adaptive Basis Method could be improved by adding more shifted basis functions (i.e. $N > 6$). This could be verified by adding additional processing to the UQP control hardware, or by implementing the controller on a system with fewer inputs & outputs. It is quite possible that any periodicity to the sensor "noise" would be controlled by using more shifted basis functions, allowing consistent control to a level below the typical RMS noise levels encountered in this research.

The instability induced by large impulsive disturbances while using the Adaptive Basis Method could be controlled through logic statements that effectively turn off the adaptation of the basis functions if a large disturbance is sensed. Typically such disturbances are of very short duration, and the impact to performance would be minimal. Implementation of such a "smart switch" would improve the overall stability properties of the controller.

An alternative solution to the rapidly varying frequency problem is to significantly improve the frequency estimation accuracy of the Sine/Cosine Method. Implementing polynomial curves fit to past frequency estimates would allow more accurate extrapolation of the disturbance frequencies to the time steps prior to the next update. This would allow more precise control of rapidly varying disturbances, while maintaining the desirable performance of the Sine/Cosine Method.

Since the Clear Box Algorithm provides an accurate system dynamics model, this model could be used to implement a feedback controller for broadband disturbances. The two controllers working together would provide a total solution for disturbance rejection, and could be implemented on the UQP or another experimental apparatus.

An ideal adaptive feedforward controller would have the intelligence of the Clear Box Algorithm and the computational efficiency of the LMS Algorithm. Creating a "hybrid" controller is possible by using the system identification information from Clear Box to improve the intelligence of the LMS Algorithm, while also eliminating the need for a separate sensor to provide the disturbance-correlated signal. This would be accomplished by using the calculated disturbance effect signal $\eta(k)$ as the reference

signal $x(k)$. Selective disturbance control would be accomplished through filtering of the disturbance effect signal, as with the Adaptive Basis Method. Preliminary efforts to accomplish this met with some success, but more work is required to refine the technique.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: COMPUTER CODE

The following code listings are provided for the Clear Box Algorithm's implementation on the Ultra Quiet Platform. Some of the more simple subroutines have been omitted for brevity. A description of each file and its function are provided below.

File Name	Description	Type	Page #
init_uqp.m	Initializes variables needed by the Clear Box Algorithm	M file (script)	168
start_uqp.m	Host PC supervisory code that interfaces with the user, controls the order of tasks accomplished by the DSP, and initiates system and disturbance identification.	M file (script)	171
cbox_sys_id.m	System identification code	M file (function)	178
eta2freq_id.m	Frequency identification from disturbance effect data	M file (function)	181
arx2ss.m	Conversion from ARX form to state-space form	M file (function)	183
ss2modal.m	Diagonalization of state-space model	M file (function)	184
ss2arx.m	Conversion from state-space form to ARX form	M file (function)	186
elim_dist.m	Elimination of disturbances from corrupted model	M file (function)	188
analyze_modes.m	System ID mode analysis to determine which modes are disturbances (used for subsequent elimination of disturbance modes from the system model)	M file (function)	189
analyze_eta_model.m	Frequency ID mode analysis to determine which modes are disturbances (used for controlling the disturbances using the Sine/Cosine Method)	M file (function)	191
dist_gen.c	Disturbance generator	C code (S-function)	192
eta.c	Disturbance Effect Calculation	C code (S-function)	198
u_ff.c	Feedforward Control Calculation – Sine/Cosine Method	C code (S-function)	202
u_ff_eta.c	Feedforward Control Calculation – Adaptive Basis Method	C code (S-function)	214

Table A-1: Index of Computer Code Listings

VARIABLE INITIALIZATION

```
% init_uqp.m                Stephen Edwards                10 May 99
%
% This script file initializes the variables and parameters needed
% for the UQP control experiment "uqp.mdl"

clear
close all          % Closes all open figure windows

% Initialize the parameters & parameter sizes needed for the S-
Functions that calculate
% the disturbance effect and the feedforward control signal(s).
%
%  p               = Order of ARX model
%  m               = Number of inputs to UQP (actuators)
%  q               = Number of outputs from UQP (sensors)
%  theta_bar       = Disturbance-free ARX model coefficients
%                  = [alpha_bar' ; beta_bar']
%                     - each alpha_bar "coefficient" has dimension qxq
%                     - each beta_bar "coefficient" has dimension qxm
%                     - there are p of each, so the dimensions of theta
%                        are [(p*q + p*m) x q]
%  nfreq           = The number of freq's to be controlled currently
%  max_freq        = The max possible number of freq's to be controlled
%  control_freqs   = Identified frequencies to control

fprintf( '\n \n \n Welcome to the UQP Control Experiment... \n \a')
m         =input( '\n How many input channels? [eg. 6]:              ');
q         =input( '\n How many output channels? [eg. 6]:             ');
p         =input(['\n What value for p (options below)?', ...
                  '\n [5,6,7,8,9,10,15,20,30 or 40]:                '']);

% Initialize variables for Recursive Least Squares
lambda          =0.999;              % Forgetting Factor
p_init          =1e1;                % Initial RLS covariance

% Initialize variables for u_ff.c
nfreq           =0;
max_freq        =3;
control_freqs   =zeros(1,max_freq);  % Initialize to zero

% Initialize variables for u_ff_eta_Nx.c
basis           =3;                  % Strut number chosen as basis
filt_order      =8;
filt_alpha      =zeros(1,filt_order+1);
filt_beta       =zeros(1,filt_order+1);
```

```
% Initialize the ARX model to reference values
switch p
case 5
   load models\cbox_p05_10sec   theta_bar   alpha_bar   beta_bar
case 10
   load models\cbox_p10_10sec   theta_bar   alpha_bar   beta_bar
case 15
   load models\cbox_p15_10sec   theta_bar   alpha_bar   beta_bar
case 20
   load models\cbox_p20_10sec   theta_bar   alpha_bar   beta_bar
case 25
   load models\cbox_p25_10sec   theta_bar   alpha_bar   beta_bar
case 30
   load models\cbox_p30_10sec   theta_bar   alpha_bar   beta_bar
case 40
   load models\cbox_p40_10sec   theta_bar   alpha_bar   beta_bar
end

%  Initialize the enable/disable flags (on=1, off=0)
flag_update    =0;   % Feedforward coefficient update
flag_control   =0;   % Feedforward control
flag_noise     =0;   % Excitation white noise
flag_disturb   =0;   % Sinusoidal disturbance(s)
flag_dist_var  =0;   % Time variation of disturbance(s)
flag_reset     =0;   % Reset for controller RLS algorithm
flag_wrong_freq=0;   % For experiments where the ID'd freq. Is
                       deliberately set incorrectly
flag_eta_filter=0;   % For eta-controller experiments where not all
                       frequencies are controlled
flag_archive   =0;   % Archive data from the experiment run

% Initialize the DC offset and the AC limit.  The UQP actuators should
% be operated in the 0-100 Volt region, so typically an offset of 50
% Volts is used, and the limit on the AC signal should be set to +/- 20
% Volts at most. Note:  Levels are voltage at actuator (after
% amplifier).  The gain associated with the D/A converter and the
% amplifier are accounted for using a gain block in the "Control Signal
% Processing" Subsystem in uqp.mdl (Simulink Diagram).

DC_offset   =50.0;
AC_limit    =10.0;

% Initialize the limit of the Disturbance generator voltage.  This
% protects the bass shaker from exposure to excessive voltage levels as
% amplitudes are sometimes varying with time, and if not monitored
% closely could get too high.  Limit expressed in +/- volts at the
% amplifier input.

Dist_limit  =4.0;

% Initialize the sizes of the disturbance related parameter vectors.
% This way, the set_dist.m file can be used while the DSP is running to
% change the number of disturbances frequencies that are present (a
% maximum of 5 unless these vectors are made bigger, and the
```

```
% build/download process is repeated).
%
%  dist_amp0    = initial amplitude (volts at Kepco amplifier input)
%  amp_rate     = rate of amplitude change (volts/sec)
%  dist_freq0   = initial frequencies conv. to rad/sec
%  freq_rate    = rate of frequency change (Hz/sec) conv. to rad/sec/sec
%  phase        = Initial phase angle of sinusoid conv. to radians

dist_amp0 =[  0.0      0.0      0.0      0.0      0.0    ];
amp_rate  =[  0.0      0.0      0.0      0.0      0.0    ];
dist_freq0=[  0.0      0.0      0.0      0.0      0.0    ]*2*pi;
freq_rate =[  0.0      0.0      0.0      0.0      0.0    ]*2*pi;
phase     =[  0.0      0.0      0.0      0.0      0.0    ]*pi/180;


% Initialize the sample rate of the controller and I/O functions
sample_rate =1000;                % Controller sample rate (Hz)
dt=1/sample_rate;                 % Sample period

io_rate=10*sample_rate;           % Sample rate of A/D & D/A
dt_io=1/io_rate;
```

HOST PC SUPERVISORY CODE

```
% start_uqp.m                Stephen Edwards                    10 May 99
%
% This script file is the Matlab code that oversees execution of the
% control experiments on the Ultra Quiet Platform using the clear box
% method.
%
% To use this file you must:
%     1) At the matlab prompt type "uqp" (in directory c:\sge\cbox)
%           - this opens the Simulink diagram that represents the DSP
%             code
%           - it also automatically initializes the variables defined
%             in the file "init_uqp.m"
%     2) "Download" the DSP code from the Simulink model "uqp.mdl"
%        using the Multiprocessor Setup window in the Simulink Diagram
%        (not RTW build!)
%     3) At the Matlab prompt type "start_uqp", and follow the prompts


% First initialize the MLIB and MTRACE environments.  This gets the
% trace file variable descriptions (addresses) for the various
% parameters used in the DSP code (flags, disturbance levels, etc).
% Once this routine is run, the various Matlab prompt commands can be
% used (i.e. "noise_on", "control_off", etc.)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Prompt for ID/Control Selections and Initialize DSP            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('\n\n\n WELCOME TO THE UQP CONTROLLER - GOOD LUCK! \n\n')

data_duration           = 600;                  % Duration of exp data
experiment_duration     = data_duration + 1;    % Duration in seconds

% Ask the user what type of experiment to perform
exp_type       =input(['\n',...
      '\n What Type of Experiment?\n',...
      '\n 1 = Perform system ID before initiating controller',...
      '\n 2 = Initiate controller using the stored reference model',...
      '\n 3 = Initiate controller using model from last experiment',...
      '\n Choice = ']);

control_type   =input(['\n',...
      '\n What Type of Controller are you set up for?\n',...
      '\n 1 = Sine / Cosine Method',...
      '\n 2 = Eta Method',...
      '\n Choice = ']);
```

```
switch control_type

case 1   % sine/cosine-controller
   freq_id_meth   =input(['\n',...
       '\n What Type of Frequency ID? \n',...
       '\n 1 = Perform disturbance freq ID using input/output data',...
       '\n 2 = Perform disturbance freq ID using disturb. effect',...
              'data [recommended]',...
       '\n 3 = Deliberately assign incorrect frequency(ies)',...
       '\n Choice = ']);

   switch freq_id_meth
   case 1
      dist_loop_duration   = 10;     % seconds (choose larger value
                                     % since the controller must be
                                     % turned off to identify the dist.
                                     % frequencies, so you don't want to
                                     % do it as often)

   case 2
      dist_loop_duration   = 1.0;    % seconds (choose smaller value
                                     % since the controller can stay on
                                     % while identifying the disturbance
                                     % frequencies)
   case 3
      dist_loop_duration = experiment_duration; % Prevents looping
      freq_error  =input(['\n',...
         '\n Enter the percent error to use for the disturbance',...
           ' frequency:',...
         '\n [i.e. "1" equals 1 percent, negative values OK]',...
         '\n Choice = ']);

   end

case 2   % eta-controller
   % Set variables used in eta controller
   N=input('\n What is Value of N you are using? :  ');
   basis_set_code=1;     % set #1 = {2, 7, 9, 12, 16, 22, 23, 30}
                         % this code stored in archive data file.
   flag_eta_filter   =input(['\n',...
         '\n Do you want to filter the eta signal?\n',...
         '\n 1 = yes',...
         '\n 0 = no',...
         '\n Choice = ']);
   switch flag_eta_filter
   case 0
      % do nothing - filter not used in eta controller S-Function
   case 1
      [filt_order,filt_alpha,filt_beta]=filter_design_butter(dt);
      % This is a simple function to design a Butterworth filter
      % using the Matlab Signal Processing Toolbox commands
      % (not provided here)
   end
end
```

```
% Initialize the environment for dSPACE MLIB and MTRACE commands
init_mlib
      % This is a Matlab script file to assign variable names to the
      % addresses of required DSP variables, allowing the passing of
      % variables to and from the DSP

% Make sure configuration is initialized (simple Matlab script file
% files not included here).
reset_uqp
control_off
update_off
noise_off

% Load filter if appropriate
if flag_eta_filter == 1
  load_filter
end

% Select RLS parameters best suited to control method
% Future option - can use set_lambda (forgetting factor) and set_p_init
%               - (covariance) to modify variable in DSP

% Set up the disturbances
fprintf('\n\n\n DISTURBANCE SET-UP FOR THIS EXPERIMENT: \n\n')
set_dist

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Perform System Identification, if Selected Above                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

switch exp_type
case 1   %%% case = Do ID First, then Control %%%

   fprintf('\n >>> Starting System ID \n')
   % Initialize and turn on the excitation noise (all 6 struts at once)
   id_type='system     ';  % field must have the same number of spaces
                           % as the word "disturbance"
   dist_on
   set_noise_auto          % Set levels of white noise
   pause(1);               % Let initial turn-on transients die out

   % Get the input & output data and store in the variable name
   % 'trace_data'
   num_sec=10.0;           % The duration of the data capture
   get_sys_id_data         % A Matlab script file (not provided)
   noise_off               % Turn off the 6-strut excitation

   % Use a batch version of the Clear Box method to get the system
   % disturbance-free model, the disturbance frequency(ies) and damping
   % ratio(s), and the chosen model order, p.
   [alpha_bar,beta_bar,theta_bar,d_freqs,d_damps,p]=...
                                          cbox_sys_id(trace_data,p,dt);
```

```
   % Download Disturbance-Free Model to DSP
   load_model

   % Note: Model data is saved during cbox_batch function to:
   %       c:\sge\cbox\models\current_cbox_model

case 2   % Start controller using reference model loaded earlier %

   id_type='none        ';  % field must have the same number of spaces
                            % as the word "disturbance"

   switch p
   case 5
      load models\cbox_p05_10sec   theta_bar   alpha_bar   beta_bar
   case 10
      load models\cbox_p10_10sec   theta_bar   alpha_bar   beta_bar
   case 15
      load models\cbox_p15_10sec   theta_bar   alpha_bar   beta_bar
   case 20
      load models\cbox_p20_10sec   theta_bar   alpha_bar   beta_bar
   case 25
      load models\cbox_p25_10sec   theta_bar   alpha_bar   beta_bar
   case 30
      load models\cbox_p30_10sec   theta_bar   alpha_bar   beta_bar
   case 40
      load models\cbox_p40_10sec   theta_bar   alpha_bar   beta_bar
   end

   load_model     % Download Disturbance-Free Model to the DSP

case 3   % Start controller using model identified in last experiment %

   id_type='none        ';  % field must have the same number of spaces
                            % as the word "disturbance"

   load models\current_cbox_model  theta_bar  beta_bar
   % Download Disturbance-Free Model (loaded in init_uqp.m or done in
   % prior experiment) to the DSP
   load_model

end   % switch for experiment type


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Start Selected Controller                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

start_time=clock;       % Mark this time as the start of the experiment
first_time = 1;         % Used to control output to screen using eta
                        % freq ID method
dist_on                 % Turn on the disturbance, if not already on
```

```
while (etime(clock,start_time) < experiment_duration)

   begin_loop=clock;

   switch control_type
   case 1   % Controller using sine/cosine basis
      switch freq_id_meth
      case 1                    % Sine/cosine - Disturbance freq ID via
                                % input/output data

         id_type='disturbance';

         update_off             % Disable control coefficient updating
         control_off            % Turn off control signal
         set_noise_auto         % Set white noise levels
         pause(1);              % Let initial turn-on transients die out

         num_sec=3.0;           % Duration of data capture
         get_in_out_data        % Get the input & output data and store
                                % in the variable name 'trace_data'
         noise_off              % Turn off the 6-strut excitation

         % Do disturbance ID
         p_dist=10;
         [d_freqs,d_damps]=...
                  cbox_freq_id(trace_data,p_dist,dt_data,max_freq);
            % Note: the above function is essentially the same as
            % "cbox_sys_id", except the routine stops as soon as the
            % frequencies are identified, thus saving computational
            % resources. (thus this routine is not provided)

         % Determine disturbance info from cbox_freq_id output
         nfreq                  =length(d_freqs);
         control_freqs          =zeros(1,max_freq);
         control_freqs(1:nfreq) =d_freqs;

         load_freqs             % Download frequencies to DSP

         % If first time through start data acquisition
         if first_time == 1
            dist_reset          % If disturbances are using a variation
                                % profile, this will reset them back to
                                % their initial conditions.
            dist_on             % Turn on the disturbance

            num_sec=data_duration;
            dt_exper_data=ceil(num_sec/3)*dt;   % Prevents overflow of
                                                % memory buffer.
            get_exper_data      % Capture data
         end

         control_on             % Turn on controller
         update_on
```

```
   % Pause until time for the next frequency update
   %(dist_loop_duration set at beginning of "start_uqp.m" )
   while(etime(clock,begin_loop) < dist_loop_duration), end
   first_time = 0;

case 2    % Sine/Cosine Method -
          % Disturbance freq ID via disturbance effect data


   num_sec=dist_loop_duration-0.5;
   dt_eta_data=ceil(num_sec/3)*dt;  % Prevents memory overflow
   get_eta_data                     % Get the disturbance effect
                                    % data and store in the
                                    % variable name 'eta_hist'

   % Do disturbance frequency ID (AR model order = tau)
   tau=10;
   [d_freqs,d_damps]=...
      eta2freq_id(eta_hist,tau,dt_eta_data,max_freq,first_time);

   % Determine disturbance info from eta2freq_id output
   nfreq=length(d_freqs);
   control_freqs=zeros(1,max_freq);
   control_freqs(1:nfreq)=d_freqs;

   load_freqs                       % Download frequencies to DSP

   % If first time through start data acquisition, and turn on
   % controller/update
   if first_time == 1
      dist_reset_no_echo            % These versions prevent
      dist_on_no_echo               % output to the screen
      num_sec=data_duration;
      dt_exper_data=ceil(num_sec/3)*dt;    % Prevents overflow of
                                           % memory buffer
      get_exper_data
      control_on_no_echo
      update_on_no_echo
   end

   % Pause until time for the next frequency update
   while(etime(clock,begin_loop) < dist_loop_duration), end
   first_time = 0;

case 3    % Sine/Cosine Method -
          % Deliberately load the incorrect frequency

   d_freqs        =dist_freq0(1:num_dist)*(1 + freq_error/100);
   nfreq          =length(d_freqs);
   control_freqs  =zeros(1,max_freq);
   control_freqs(1:nfreq)  =d_freqs;

   load_freqs        % Download frequencies to DSP
```

```
        % If first time through start data acquisition
        if first_time == 1
           num_sec=data_duration;
           dt_exper_data=ceil(num_sec/3)*dt;    % Prevents overflow of
                                                % memory buffer
           get_exper_data
        end

        % Turn on controller
        control_on
        update_on

        % Pause until time for the next frequency update
        while(etime(clock,begin_loop) < dist_loop_duration), end
        first_time = 0;

     end

  case 2       % Adaptive Basis Method

     num_sec=data_duration;
     dt_exper_data=ceil(num_sec/3)*dt;    % Prevents overflow of
                                          % memory buffer
     get_exper_data
     control_on
     update_on

     while(etime(clock,begin_loop) < experiment_duration), end

     % Do nothing - S-Function uses eta time history as the basis for
     % the control signal, so the frequencies do not need to be
     % estimated.

  end

end      % Experiment Complete

% Get and save the experiment data (script file not provided)
fetch_exper_data

% Turn off disturbance, noise (if present), controller, and updates
stop

% Let transients die out, then take noise level readings
pause(2);
num_sec=3.0;
dt_noise_data=ceil(num_sec/3)*dt;
get_noise_data                          % saved in variable 'noise_data'
save models\current_noise_data   noise_data   dt_noise_data

plot_experiment_results                 % Plot results

fprintf('\n\n\n STOP TIME REACHED - EXPERIMENT COMPLETE \n\n')
```

SYSTEM IDENTIFICATION

```
function [alpha_bar,beta_bar,theta_bar,d_freqs,d_damps,p]=...
                                          cbox_sys_id(trace_data,p,dt)

% Do system id based on "noisy" input / output data using the
% "clear box" method of Phan & Goodzeit.

%  alpha, beta:   ARX model coefficients
%  theta:         [alpha beta] coefficients combined into one matrix
%  p:             order of ARX model
%  f:             number of disturbance frequencies to eliminate
%  u, y, dt:      Input / output data, sample time
%  Ap, Bp, Cp:    State space model in observer canonical form
%  T:             Eigenvector Matrix of Ap
%  lambda:        Eigenvalue matrix (block diagonals = eigenvalues)
%  Lambda:        Ap transformed to modal form
%  Gamma:         Bp transformed to modal form
%  Omega:         Cp transformed to modal form


% Determine DARMA & State Space Models from the data, diagonalize, and
% find the disturbance frequency(ies).  The wn vector is sorted by
% lowest to highest damping ratio, so the first wn is the most likely
% to be a disturbance.


% Calculate the DARMA model coefficients associated with the input/
% output data supplied:
%
%   y(k) = alpha(1)*y(k-1) + alpha(2)*y(k-2) + ... + alpha(p)*y(k-p)
%        + beta(1)*u(k-1) + beta(2)*u(k-2) + ... + beta(p)*u(k-p)
%

% Assume number of inputs and outputs is 6 each (Full MIMO for UQP).
% Routine is coded for general MIMO with m inputs and q outputs.

in_struts   =[1 2 3 4 5 6];
out_struts  =[1 2 3 4 5 6];

m=length( in_struts);            % Number of Inputs
q=length(out_struts);            % Number of Outputs

u=trace_data(in_struts,:);       % Each row is time history of inputs
y=trace_data(out_struts+6,:);    % Each row is time history of outputs

L=length(u)-1;                   % number of data points in
                                 % input/output data (k=0,1,2,..L)

n_imp=100;                       % The number of impulse response data
                                 % points to calculate
```

```
% Set the damping threshold (below which a mode is considered a
% disturbance) - the threshold should usually be set lower for higher
% model order.

switch p
case 5
   damp_thresh=1e-1;
case 10
   damp_thresh=1e-1;
case 15
   damp_thresh=5e-3;
case 20
   damp_thresh=5e-3;
case 25
   damp_thresh=1e-4;
case 30
   damp_thresh=1e-4;
case 40
   damp_thresh=1e-4;
otherwise
   damp_thresh=5e-2;
end

% Build Y & V (See Goodzeit & Phan MAE Tech Report #2096 -
%              Princeton University)

fprintf('\n\n >>> Organizing input/output data \n')
Y=y(:,p+1:L+1);
V=zeros(p*q+p*m,L-p+1);
for j=1:p
   V(      (j-1)*m+1 :     j*m,:) = u(:,j:L-p+j);
   V(p*m + (j-1)*q+1 : p*m+j*q,:) = y(:,j:L-p+j);
end

% Use the pseudoinverse to determine the ARX model coefficients, alpha
% and beta

fprintf('\n >>> Calculating ARX model coefficients \n')
K        =Y*V'*pinv(V*V');
CC_MT    =K(:,    1:p*m);
Minus_CM =K(:,p*m+1:p*m+p*q);
clear K Y V                    % save memory space

alpha =[];
beta  =[];
for j=1:p
   alpha =[alpha Minus_CM(:,p*q-j*q+1:p*q-(j-1)*q)];
   beta  =[beta     CC_MT(:,p*m-j*m+1:p*m-(j-1)*m)];
end
clear Minus_CM CC_MT
theta=[alpha';beta'];
fprintf('\n >>> ARX model complete \n ')
```

```
% Transform to State Space Observable Canonical form
[Ap,Bp,Cp,Dp]=arx2ss(theta,p);
fprintf('\n >>> Converted to State Space form \n ')

% Transform to Modal (Block Diagonal) Form and calculate
% wn & zeta for all modes
[Am,Bm,Cm,Dm,num_modes,wn,zeta,m_idx] = ss2modal(Ap,Bp,Cp,Dp,dt);

% Determine which modes are disturbance modes (by index), and how much
% each mode impacts the total pulse response of the system
[dist_modes,mode_norms]=...
      analyze_modes(Am,Bm,Cm,Dm,num_modes,wn,zeta,m_idx,...
                    damp_thresh,n_imp,dt);

% Output frequency information to the screen
fprintf('\n >>> Frequencies identified as disturbances are: \n')
for i=1:length(dist_modes)
   fprintf('%10d - %9.4f  Hz \n',i,wn(dist_modes(i))/2/pi)
end

% Eliminate the disturbance modes from the state space model
[A_bar,B_bar,C_bar,D_bar]=elim_dist(Am,Bm,Cm,Dm,wn,dist_modes,m_idx);
fprintf('\n >>> Disturbances Eliminated from State Space Model \n ')

% Return model to ARX form (keep its order = p)
fprintf('\n >>> Computing disturbance-free ARX model \n ')
[alpha_bar,beta_bar]=ss2arx(A_bar,B_bar,C_bar,p);

theta_bar=[alpha_bar';beta_bar'];

% Save the model and data
save models\current_cbox_model alpha_bar beta_bar theta_bar p dt A_bar
B_bar C_bar D_bar trace_data
save models\current_model_data trace_data dt
save models\thesis_dist_corrupt_model Ap Bp Cp Dp dt

d_freqs=wn(dist_modes);
d_damps=zeta(dist_modes);

plot_damp_and_norm          % Plots damping ratio and pulse response
                            % norms for each mode of the system
                            % (not provided)
```

FREQUENCY IDENTIFICATION FROM DISTURBANCE EFFECT DATA

```
function [d_freqs,d_damps]=...
                    eta2freq_id(eta_hist,p,dt,max_freq,first_time)

% Stephen Edwards                                          May 1999

% Identify an AR model for the disturbance effect data

%  p:              order of ARX model
%  f:              number of disturbance frequencies to eliminate
%  y, dt:          eta data, sample time
%  max_freq        The maximum number of disturbance frequencies that
%                  can be controlled



% Assume number of outputs is 6 (Full MIMO for UQP).  The routine is
% coded for general MIMO with m inputs and q outputs.

out_struts  =[1 2 3 4 5 6];
q=length(out_struts);              % Number of Outputs
y=eta_hist(out_struts,:);          % Each row is time histoy of outputs
L=length(y)-1;                     % number of data points in eta data
                                   % (k=0,1,2,..L)

damp_thresh=1e-3;                  % Set the damping threshold (below
                                   % which a mode is considered a
                                   % disturbance)

if first_time == 1
   fprintf('\n >>> Estimating disturbance frequencies.... \n')
   fprintf('\n >>> Identified disturbance Frequencies are [Hz]: \n')
end

% Build Y & V
% (See Goodzeit & Phan MAE Tech Report #2096 - Princeton University)
Y=y(:,p+1:L+1);
V=zeros(p*q,L-p+1);
for j=1:p
   rows = [(j-1)*q+1 : j*q];
   cols = [j:L-p+j];
   V( rows , : ) = y( : , cols );
end

clear rows cols

% Use the pseudoinverse to determine the AR model coefficients, gamma
K     =Y*V'*pinv(V*V');
gamma =[];
```

```
for j=1:p
   gamma =[gamma K(:,p*q-j*q+1:p*q-(j-1)*q)];
end

clear K Y V

% Transform to State Space Observable Canonical form
[Am,num_modes,wn,zeta,m_idx] = ar2modal(gamma,p,dt);

% Determine which modes are disturbance modes (by index)
[dist_modes,not_controlled]=...
                     analyze_eta_model(wn,zeta,damp_thresh,max_freq);

% Print out info to the screen
if isempty(dist_modes)
   fprintf('\n   None')
else
   fprintf('\n   ')
   for i=1:length(dist_modes)
      fprintf('  %11.4f',wn(dist_modes(i))/2/pi)
   end
   if ~isempty(not_controlled)
      fprintf('     *** Dist Freqs Not Controlled =  ')
      for i=1:length(not_controlled)
         fprintf('  %11.4f',wn(not_controlled(i))/2/pi)
      end
   end
end

d_freqs=wn(dist_modes);
d_damps=zeta(dist_modes);

% Plot mode damping ratios
figure(2)
semilogy(wn/2/pi,abs(zeta),'bx',...
         wn(dist_modes)/2/pi,abs(zeta(dist_modes)),'ro',...
         [0 500],[damp_thresh damp_thresh],'r--')
title(['Disturbance ID Mode: Damping Ratio vs. Frequency, tau = ',...
       num2str(p)])
text(300,damp_thresh*1.5,'Damping Ratio Threshold')
axis([0 500 1e-7 max(abs(zeta))])
grid on
```

CONVERSION FROM ARX TO STATE SPACE FORM

```
function [A,B,C,D]=arx2ss(theta,p)

% File provided courtesy of Neil Goodzeit, Lockheed Martin Missiles &
% Space (© Copyright by Neil E. Goodzeit, 1998, All Rights Reserved)

% .... Calculate the number of inputs and outputs
       n_out = min(size(theta)) ;
       n_in  = (length(theta) - p*n_out) / p ;

% .... The ARX model response coefficients
       idx    = p*n_out ;
       alpha = theta(1:idx,1:n_out)' ;

% .... The ARX model input coefficients
       beta = theta(idx+1:idx+p*n_in,1:n_out)' ;

% .... Construct the state space model in observable canonical form
       A  = [ ] ;
       B  = [ ] ;

       for j = 1:p

         j1  = j-1 ;
         idx = j1 * n_out + 1 ;

        if j<p
          A   = [ A ; [ alpha(:,idx:idx+n_out-1), ...
                  zeros(n_out,j1*n_out), eye(n_out,n_out), ...
                  zeros(n_out,(p-j-1)*n_out) ] ] ;
        else
          A   = [ A ; [ alpha(:,idx:idx+n_out-1), ...
                  zeros(n_out,j1*n_out)      ] ] ;
        end

         B  = [ B  ; beta(:,idx:idx+n_out-1) ] ;

      end

      C  = [ eye(n_out,n_out), zeros(n_out,n_out*(p-1)) ] ;
      D  = zeros(n_out,n_in) ;
```

DIAGONALIZATION OF STATE SPACE MODEL

```
function [Am,Bm,Cm,Dm,n_mode,w_mode,z_mode,m_idx] =
                                                    ss2modal(Ap,Bp,Cp,Dp,dt)
% ss2modal.m
%--------------------------------------------------------------------------
% ... File provided courtesy of Neil Goodzeit, Lockheed Martin Missiles
%     & Space  (© Copyright by Neil E. Goodzeit, 1998, All Rights
%     Reserved)
%
% ... Modified by Stephen Edwards 4 May 99
%
% ... Convert the input state-space model to modal form
%
% ... Input parameters:
%
% ... Ap, Bp, Cp, Dp    the input state-space model
% ... dt                the sampling interval
%
% ... Output parameters:
%
% ... Am, Bm, Cm, Dm    the state-space model matrices in modal form
% ... n_mode            the number of modes
% ... w_mode            the mode frequency (rad/sec)
% ... z_mode            the mode damping ratio
% ... m_idx             the mode index matrix (n_mode,2)
%
%--------------------------------------------------------------------------

% ... Transform the model to modal coordinates
      [V,Am] = eig(Ap);

      i_eig  = 1;
      n_mode = 0;

      for j = 1:length(V)        % Iterate on each eigenvector column

         if imag(V(1,j)) == 0    % Real Eigenvector (1st order mode)

             n_mode          = n_mode + 1;
             s               = log(Am(j,j))/dt;  % See annotations below
             w_mode(n_mode)  = abs(s);
             z_mode(n_mode)  = -real(s)/w_mode(n_mode);
             m_idx (n_mode, 1:2) = [ j , j ];


% continued on next page
```

```
      else                        % Complex Eigenvector (2nd order mode)
         if  i_eig == 1           % 1st column of complex conjugate

            V(:,j) = real(V(:,j));     % Take real part
            i_eig  = 2;
            n_mode = n_mode + 1;
            s      = log(Am(j,j))/dt;  % z=e^(s*T) --> s=(ln(z))/T
                                       % s=-sigma+j*wd
            w_mode(n_mode) = abs(s);    % wn=sqrt(sigma^2+wd^2)

            z_mode(n_mode) = -real(s)/w_mode(n_mode);
                                       % zeta=sigma/wn
            m_idx (n_mode, 1:2) = [ j , j+1 ];

         elseif i_eig == 2             % 2nd column of complex conj.

            V(:,j) = imag(V(:,j));     % Take imaginary part
            i_eig  = 1;

         end

      end

   end

% ... Calculate the poles and the modal coordinate state-space matrices

   Am = inv(V)*Ap*V                  ;
   Bm = inv(V)*Bp                    ;
   Cm = Cp*V                         ;
   Dm = zeros(size(Cm,1),size(Bm,2)) ;
```

CONVERSION FROM STATE SPACE TO ARX FORM

```
function [alpha,beta]=ss2arx(A,B,C,p)

% ss2arx.m                    Stephen Edwards                    May 1999

% Some elements of the code provided by Neil Goodzeit, Lockheed
% Martin Missiles & Space
% (© Copyright by Neil E. Goodzeit, 1998, All Rights Reserved)

% Calculate the order p ARX model coefficients
% from the state-space model matrices A, B, C

% alpha is the coefficient matrix of the past response
% beta is the coefficient matrix of the past input

   n_in  = min(size(B)) ;
   n_out = min(size(C)) ;
   n     = length(A) ;

% Initialize matrices
   Ap = eye(size(A)) ;
   Op = C ;
   Cp = B ;
   Tp = [ zeros(n_out,n_in) ; C*B ] ;

% Raise A to the p power
   for i = 1:p

      if i ~= 1

         % The observability and controlability matrices
         Op = [ Op ;  C*Ap ] ;
         Cp = [ Ap*B, Cp   ] ;

         % The Toeplitz Matrix
         if i~= p
            Tp = [ Tp ;  C*Ap*B ] ;
         end

      end

      Ap = Ap * A ;

   end

   % Construct the toeplitz matrix from the first column
   for i = 2:p

      % The row and column indices
      col = (i-2)*n_in + 1 ;
      row = (i-1)*n_out + 1 ;
```

```
      n_rows   = (p - i)*n_out ;

      if i == p
         Tp = [ Tp, zeros(p*n_out,n_in) ] ;
      else
         Tp =  [ Tp, [zeros(i*n_out,n_in) ; Tp(row:row+n_rows-
1,col:col+n_in-1) ] ] ;
      end

   end

   % Calculate the observer gain matrix
   M = -Ap*pinv(Op) ;

   % Calculate the ARX model coefficients ( ordered from k-p to k-1)
   % a's are coefficients of y, b's are coefficients of u
   a  = -C*M ;
   b  =  C*(Cp + M*Tp) ;

   alpha = [] ;
   beta  = [] ;

   % Reorder the coefficients
   for i = 1:p

      id_a = (p-i)*n_out+1 ;
      id_b = (p-i)*n_in+1 ;

      alpha = [alpha, a(:,id_a:id_a+n_out-1) ] ;
      beta  = [beta,  b(:,id_b:id_b+n_in-1) ] ;

   end
```

ELIMINATION OF DISTURBANCE MODES

```
function
[A_bar,B_bar,C_bar,D_bar]=elim_dist(Am,Bm,Cm,Dm,wn,dist_modes,m_idx)

% elim_dist.m              Stephen Edwards              5 May 99
%
% A function to eliminate the disturbance modes from the modal state
% space model.
%
% Input Parameters:
%
%  Am,Bm,Cm,Dm ..... State Space model in modal form
%  wn ............... Natural frequencies of the modes
%  dist_modes ...... A vector of the indeces for the disturbance modes
%                    (each 2nd order mode has one index value)
%  m_idx ........... Identified the states associated with each mode
%                    (one row for each mode)
%
% Output Parameters:
%
%  A_bar,B_bar,
%  C_bar,D_bar ..... Disturbance-free State Space model
%  p_bar ........... p value of disturbance free model

A_bar=Am;
B_bar=Bm;
C_bar=Cm;
D_bar=Dm;

num_dist=length(dist_modes);
states_to_elim=[];

for i=1:num_dist
   % Identify disturbance mode rows and columns
   states_to_elim=[states_to_elim,  m_idx(dist_modes(i),:)];
   fprintf('\n >>> %9.4f Hz mode eliminated \n',wn(dist_modes(i))/2/pi)
end

A_bar(states_to_elim,:)=[];      % Remove rows
B_bar(states_to_elim,:)=[];

A_bar(:,states_to_elim)=[];      % Remove columns
C_bar(:,states_to_elim)=[];
```

SYSTEM MODE ANALYSIS

```
function [dist_modes,mode_norms]=...
analyze_modes(Am,Bm,Cm,Dm,num_modes,wn,zeta,m_idx,damp_thresh,n_imp,dt)

% analyze_modes.m
%-----------------------------------------------------------------------
% ... Original file provided courtesy of Neil Goodzeit, Lockheed Martin
%     Missiles & Space  (© Copyright by Neil E. Goodzeit, 1998,
%     All Rights Reserved)
%
% ... Modified by Stephen Edwards, May 99
%
% ... Input parameters:

% ... Am, Bm, Cm, Dm    the state-space model matrices in modal form
% ... num_modes         the number of modes
% ... wn, zeta          the modal frequencies (rad/sec), damping ratios
% ... m_idx             the mode index matrix (num_modes,2)
%                       the starting and ending state for each mode
% ... damp_thresh       the damping ratio threshold (below threshold
%                       means it's a disturbance)
% ... n_imp             the number of impulse response samples
% ... dt                the sampling interval (seconds)
%
%
% ... Output parameters:

% ... dist_modes        Indeces of modes that are disturbances (have
%                       damping ratios below the threshold passed into
%                       the function (damp_thresh) ).
% ... mode_norms        Inner product of total pulse response and modal
%                       pulse response - gives the correlation/
%                       contribution of each mode to the total output.
%-----------------------------------------------------------------------


% ... The number of inputs, outputs, and states
      n_out   = size(Cm, 1) ;       % The number of rows of C
      n_in    = size(Bm, 2) ;       % The number of columns of B
      n_state = length(Am)  ;

% ... Set the direct transmission matrix to zero
      D_mode = zeros(num_modes*n_out, n_in   ) ;

% ... Calculate the modal output matrix
      C_mode = zeros(num_modes*n_out, n_state) ;

      for i = 1:num_modes
          idx = n_out*(i-1) + 1 ;
          C_mode(idx:n_out*i , m_idx(i,1):m_idx(i,2)) = Cm(1:n_out,
m_idx(i,1):m_idx(i,2)) ;
```

```
      end

      mode_norms = zeros(num_modes,1) ;

      for i = 1:n_in

% .....  The modal impulse responses
         y  = dimpulse(Am,Bm,C_mode,D_mode,i,n_imp) ;

% .....  The total impulse response
         y_t= dimpulse(Am,Bm,Cm,    Dm,    i,n_imp) ;

% .....  Sum the impulse responses for each mode
         for j = 1:num_modes

             idx = n_out*(j-1) + 1 ;
             for jj = 1:n_out
                  y2(idx+jj-1) = y_t(:,jj)' * y(:,idx+jj-1)  ;
             end

             mode_norms(j) = mode_norms(j) + sum(y2(idx:idx+n_out-1)) ;
%             mode_norms(j) = mode_norms(j) + max(y2(idx:idx+n_out-1))
;


         end

      end

      mode_norms=mode_norms';
      clear idx

% ... Sort modes from min to max damping ratio
      [sorted_zeta,idx]=sort(abs(zeta));
      sorted_wn=wn(idx);

% ... Find the indeces of those below the damping threshold - assume
% ... no first order modes will meet the criteria.
      dist_modes=[];
      i=1;
      current_zeta=sorted_zeta(i);

      while (current_zeta < damp_thresh)
         dist_modes=[dist_modes idx(i)];
         i=i+1;
         current_zeta=sorted_zeta(i);
      end
```

DISTURBANCE MODE ANALYSIS

```
function [dist_modes,not_controlled]=...
                        analyze_eta_model(wn,zeta,damp_thresh,max_freq)

% analyze_eta_model.m          Stephen Edwards                Jul 1999
%-----------------------------------------------------------------------
% ... Input parameters:

% ... wn                 The natural frequencies
% ... zeta               The modal damping ratios
% ... damp_thresh        The damping ratio threshold (below threshold
%                        means it's a disturbance)
% ... max_freq           The maximum number of frequencies that can be
%                        handled by the processor
%
% ... Output parameters:

% ... dist_modes         Indeces of modes that are disturbances (have
%                        damping ratios below the threshold)
% ... not_controlled     Modes that qualify as disturbances but are not
%                        controlled (due to excessive number of them)
%-----------------------------------------------------------------------

% ... Sort modes from min to max damping ratio
      [sorted_zeta,idx]=sort(abs(zeta));

% ... Find the indeces of those below the damping threshold - assume
% ... no first order modes will meet the criteria.
      dist_modes=[];
      i=1;
      current_zeta=sorted_zeta(i);

      while (current_zeta < damp_thresh)
         dist_modes=[dist_modes idx(i)];
         i=i+1;
         current_zeta=sorted_zeta(i);
      end

% ... Sort identified modes from lowest to highest frequency
%     (to prevent flip-flopping)
      [sorted_wn,idx]=sort(wn(dist_modes));
      dist_modes=dist_modes(idx);

% ... Only take the number of modes that the processor can handle
      if length(dist_modes) > max_freq
         dist_modes     =dist_modes(1:max_freq);
         not_controlled =dist_modes(max_freq+1:length(dist_modes));
      else
         not_controlled =[];
      end
```

DISTURBANCE GENERATION

```
/*
 * dist_gen.c      Generates time varying disturbances.
 *                 The parameters are initialized in the Simulink
 *                 initialization file (i.e. "init_id_mimo", etc.)
 *
 *
 * Passed Parameters:
 *    FLAG_DISTURB  = Logic flag (1 = enable disturbances)
 *    FLAG_DIST_VAR = Logic flag (1 = enable disturbance time-variaton)
 *    DIST_AMP      = The initial amplitude(s) of the disturbance(s)
 *    AMP_RATE      = Rate of change of the amplitude(s)
 *    DIST_FREQ     = The initial frequency(ies) of the disturbance(s)
 *    FREQ_RATE     = Rate of change of the frequency(ies)
 *    PHASE         = The phase angles of the dist. sinusiod(s)
 *    DT            = Sample time (step size)
 *
 * Other Variables:
 *    NUM_DIST      = The number of disturbance frequencies
 *
 */

#define S_FUNCTION_NAME  dist_gen
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

#define FLAG_DISTURB(S)        ssGetSFcnParam(S,0)
#define FLAG_DIST_VAR(S)       ssGetSFcnParam(S,1)
#define DIST_AMP(S)            ssGetSFcnParam(S,2)
#define AMP_RATE(S)            ssGetSFcnParam(S,3)
#define DIST_FREQ(S)           ssGetSFcnParam(S,4)
#define FREQ_RATE(S)           ssGetSFcnParam(S,5)
#define PHASE(S)               ssGetSFcnParam(S,6)
#define DT(S)                  ssGetSFcnParam(S,7)

#define N_PARAM                8

#define TWO_PI                 6.28318530717959
#define NUM_DIST               mxGetN(DIST_FREQ(S))

#define NUM_SECTIONS           5
#define WK_TIMER               3*NUM_DIST


static void mdlInitializeSizes(SimStruct *S)
{
    int_T  num_rwork;

    num_rwork=3*NUM_DIST+1;          /* See ssSetNumRWork below */

    ssSetNumSFcnParams(S, N_PARAM);
```

```
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
    {
        /* Return if expected number != actual number */
        return;
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 0)) return;
    if (!ssSetNumOutputPorts(S, 2)) return;
    ssSetOutputPortWidth(S, 0, 1);
                /* The first output is a sum of all the
                   disturbances generated -- it contains
                   all of the diturbance frequencies.    */
    ssSetOutputPortWidth(S, 1, NUM_DIST);
                /* The second output is a vector of the
                   actual disturbance frequencies.      */
    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, num_rwork);
                /* RWork[0..n_dist-1]=Current dist. amplitudes.
                   RWork[n_dist..2*n_dist-1]=Current dist. freqs.
                   RWork[2*n_dist..3*n_dist-1]=Current dist. angles
                   RWork[3*n_dist]=Amount of time in profile section */
    ssSetNumIWork(S, 1);
                /* IWork[0]=Current Section Number */
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* Function: mdlInitializeSampleTimes
=========================================
 * Abstract:
 *    This function is used to specify the sample time(s) for your
 *    S-function.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
      real_T      dt;
      dt          =mxGetPr(DT(S))[0];
      ssSetSampleTime(S, 0, dt);
      ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
  /* Note, this routine will be called at the start of simulation and
   * if it is present in an enabled subsystem configured to reset
   * states, it will be call when the enabled subsystem restarts
   * execution to reset the states.
   */
```

```
static void mdlInitializeConditions(SimStruct *S)
{
      int_T       i;
      real_T      *amp        =mxGetPr(DIST_AMP(S));
      real_T      *freq       =mxGetPr(DIST_FREQ(S));
      real_T      *phase      =mxGetPr(PHASE(S));
      real_T      *RWork      =ssGetRWork(S);
      int_T       *IWork      =ssGetIWork(S);


      for (i=0 ; i<NUM_DIST ; i++)
            {                 /* Initialize the amplitudes of
                                 the disturbance frequencies  */

                  RWork[i]=amp[i];

                              /* Initialize the angles of
                                 the disturbance frequencies
                                 to their corresponding phase
                                 angles                */

                  RWork[i+NUM_DIST]=freq[i];

                              /* Initialize the angles of
                                 the disturbance frequencies
                                 to their corresponding phase
                                 angles                */

                  RWork[i+2*NUM_DIST]=phase[i];
            }

      RWork[WK_TIMER]   =0.0; /* Set timer to zero          */

      IWork[0]    =0;         /* Counter for time-varying profile
                                 that keeps track of the time
                                 so that different sections of
                                 the profile can have different
                                 frequency variation rates    */
}
#endif /* MDL_INITIALIZE_CONDITIONS */

#undef MDL_START  /* Change to #define to add function */
#if defined(MDL_START)
  static void mdlStart(SimStruct *S)
  {
  }
#endif /*  MDL_START */


/*
 * Function: mdlOutputs
 */
```

```
static void mdlOutputs(SimStruct *S,  int_T tid)
{
      int_T       i;
      int_T       flag_disturb      =mxGetPr(FLAG_DISTURB(S))[0];
      int_T       flag_dist_var     =mxGetPr(FLAG_DIST_VAR(S))[0];
      int_T       *IWork            =ssGetIWork(S);
      int_T       current_section;

      real_T      section_duration[NUM_SECTIONS]=
                                    {1.075,4.0,1.0,4.0,100.0};
      real_T      section_freq_rate[NUM_SECTIONS]=
                                    {0.0,12.5664,0.0,-0.6283,0.0};
      real_T      time_in_section;
      real_T      dt                =mxGetPr(DT(S))[0];
      real_T      *amp_rate         =mxGetPr(AMP_RATE(S));
      real_T      *freq_rate        =mxGetPr(FREQ_RATE(S));
      real_T      *RWork            =ssGetRWork(S);
      real_T      *y1               =ssGetOutputPortRealSignal(S,0);
      real_T      *y2               =ssGetOutputPortRealSignal(S,1);

/*
 * If the disturbance is enabled, update and output disturbance signal
 */
      if (flag_disturb == 1)
      {
            switch(flag_dist_var)
                  /*  Update the amplitude, frequency, and angle if
                      variation enabled or if the time-varying
                      profile is being used */
            {
                  case(0):
                  {           /*  Update only the angle if time
                                  variation is disabled */

                  for (i=0 ; i<NUM_DIST ; i++)
                        RWork[i+2*NUM_DIST] += RWork[i+NUM_DIST]*dt;
                        break;
                  }

                  case(1):
                  {           /*  Use linear variation amounts in
                                  set_dist.m to adjust the
                                  amplitude, frequency, and angle */

                     for (i=0 ; i<NUM_DIST ; i++)
                     {

                        RWork[i]                +=amp_rate[i]*dt;
                        RWork[i+NUM_DIST]       +=freq_rate[i]*dt;
                        RWork[i+2*NUM_DIST]     +=RWork[i+NUM_DIST]*dt;
                     }
                     break;
                  }
```

```
            case(2):
            {           /*  Keep amplitude constant, but update
                            frequency using the pre-set rates
                            (by profile section) defined at the
                            beginning of this file, and also
                            update angle.  This is only valid
                            for single disturbance frequencies
                            (i.e., i=0)    */

               current_section      =IWork[0];

               RWork[0+NUM_DIST]    +=
                         section_freq_rate[current_section]*dt;
               RWork[0+2*NUM_DIST]  +=RWork[0+NUM_DIST]*dt;


               time_in_section      =RWork[WK_TIMER];
               time_in_section+=dt;

               if (time_in_section >
                              section_duration[current_section])
                  {
                  current_section   += 1;
                  time_in_section    = 0.0;
                  }

               IWork[0]             =current_section;
               RWork[WK_TIMER]      =time_in_section;

            }

      }     /* End of amplitude, frequency, and angle updates */


      /* Calculate disturbance (one signal includes all freqs) */
      y1[0]=0.0;
      for (i=0 ; i<NUM_DIST ; i++)
      {
            /* Keep the angle between -2*pi and 2*pi radians */
            if ( RWork[i+2*NUM_DIST] > TWO_PI )
            {
                  RWork[i+2*NUM_DIST]-=TWO_PI;
            }
            if ( RWork[i+2*NUM_DIST] < -TWO_PI )
            {
                  RWork[i+2*NUM_DIST]+=TWO_PI;
            }
      /* Output disturbance signal */
            y1[0]       += RWork[i]*sin(RWork[i+2*NUM_DIST]);

      /* Output true disturbance frequencies */
            y2[i]        = RWork[i+NUM_DIST];
      }
}
```

```
        else        /* Set output to zero if disturbance disabled */
        {
              y1[0]=0.0;              /* Output no disturbance */

              for (i=0 ; i<NUM_DIST ; i++)
                    y2[i]=0.0;        /* Output freqs = 0 */
        }
}

/* Disturbance calculations complete */



#define MDL_UPDATE  /* Change to #undef to remove function */
#if defined(MDL_UPDATE)
  static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */

#define MDL_DERIVATIVES  /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
  static void mdlDerivatives(SimStruct *S)
  {
  }
#endif /* MDL_DERIVATIVES */

static void mdlTerminate(SimStruct *S)
{
}

/*=============================*
 * Required S-function trailer *
 *=============================*/

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c"  /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"   /* Code generation registration function */
#endif
```

DISTURBANCE EFFECT CALCULATION

```
/*
 * eta.c    S-function to calculate the disturbance effect
 *          from input-output data and user supplied model coefficients
 *
 *
 */

#define S_FUNCTION_NAME eta
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

#define THETA_MATRIX(S)       ssGetSFcnParam(S,0)
#define P_MOD(S)              ssGetSFcnParam(S,1)
#define DT(S)                 ssGetSFcnParam(S,2)
#define N_PARAM               3

static void mdlInitializeSizes(SimStruct *S)
{
      int_T      p      =mxGetPr(P_MOD(S))[0];
                  /* order of ARX model */

      int_T    pq_pm    =mxGetM(THETA_MATRIX(S));
                  /* coefficient matrix row dimension = p*q + p*m */

      int_T    q  =mxGetN(THETA_MATRIX(S));
                  /* coefficient matrix column dimension =q */

      int_T    m;

      m=(pq_pm/p)-q;
      ssSetNumSFcnParams(S, N_PARAM);
      if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
      {
          /* Return if expected number != actual number */
          return;
      }

      ssSetNumContStates(S, 0);
      ssSetNumDiscStates(S, 0);

      if (!ssSetNumInputPorts(S, 2)) return;
      ssSetInputPortWidth(S, 0, m);
                  /* 1st Input = Inputs  (to Actuators) */

      ssSetInputPortWidth(S, 1, q);
                  /* 2nd Input = Outputs (from Sensors) */

      ssSetInputPortDirectFeedThrough(S, 0, 1);
      ssSetInputPortDirectFeedThrough(S, 1, 1);
```

```
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, q);      /* Dist. Effect */

    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, pq_pm);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    real_T   dt =mxGetPr(DT(S))[0];

    ssSetSampleTime(S, 0, dt);
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
static void mdlInitializeConditions(SimStruct *S)
{
    int         i;
    real_T      *RWork  =ssGetRWork(S);

    for ( i = 0 ; i < ssGetNumRWork(S) ; i++ )
    {
        RWork[i] = 0.0;
    }
}
#endif /* MDL_INITIALIZE_CONDITIONS */


/* Function: mdlOutputs
 * In this function, you compute the outputs of your S-function
 * block. Generally outputs are placed in the output vector, ssGetY(S).
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{

    real_T      *theta      =mxGetPr(THETA_MATRIX(S));
        /* pointer to the model coefficients matrix structure */

    int_T       p           =mxGetPr(P_MOD(S))[0];
        /* order of ARX model */

    int_T       pq_pm       =mxGetM(THETA_MATRIX(S));
        /* coefficient matrix row dimension = p*q + p*m */

    int_T       q           =mxGetN(THETA_MATRIX(S));
        /* coefficient matrix column dimension =q */
```

```
    int_T             i, j, m;

    InputRealPtrsType  u1   =ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType  u2   =ssGetInputPortRealSignalPtrs(S,1);
    real_T             *y   =ssGetOutputPortRealSignal(S,0);
    real_T           *Rwork =ssGetRWork(S);

/*
 *    The RWork vector indeces are set up as follows:
 *
 *    [0]    = y1(k-1)   1st sensor output, delayed by one time step
 *    [1]    = y2(k-1)   2nd sensor output,    "       "       "
 *     |
 *    [q-1]  = yq(k-1)   qth sensor output,    "       "       "
 *
 *    [q]    = y1(k-2)   1st sensor output, delayed by two time steps
 *     |
 *    [2*q-1]= yq(k-2)   qth sensor output,    "       "       "
 *     |
 *     |
 *    [p*q-1]= yq(k-p)   qth sensor output, delayed by p time steps
 *-------------------------------------------------------------------
 *
 *    [p*q]   = u1(k-1)  1st actuator input, delayed by one time step
 *    [p*q+1]= u2(k-1)   2nd actuator input,    "       "       "
 *     |
 *    [p*q+m-1] = um(k-1) mth actuator input,   "       "       "
 *
 *    [p*q+m]   = u1(k-2) 1st actuator input, delayed by two time steps
 *     |
 *    [p*q+2*m-1] = um(k-2) mth actuator input,   "       "       "
 *     |
 *     |
 *    [p*q+p*m-1] = um(k-p) mth actuator input, delayed by p time steps
 */

    m=(pq_pm/p)-q;

/*  Shift the memory */
    for (i = pq_pm-1 ; i > p*q+m-1 ; i-- )
        RWork[i] = RWork[i-m];

    for (i = p*q-1 ; i > q-1 ; i-- )
        RWork[i] = RWork[i-q];

/*  Save the current input */
    for (i = 0 ; i < q ; i++ )
        RWork[i]    = *u2[i];

    for (i = 0 ; i < m ; i++ )
        RWork[p*q+i]      = *u1[i];
```

```
/*  Calculate the disturbance effect - eta */

      for ( i = 0 ; i < q ; i++ )
      {
            y[i] = 0.0;
            for ( j = 0 ; j < pq_pm ; j++ )
            {
                  y[i] += theta[i*pq_pm+j] * RWork[j];
            }
      }
}


#define MDL_UPDATE  /* Change to #undef to remove function */
#if defined(MDL_UPDATE)
  static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */

#define MDL_DERIVATIVES  /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
  static void mdlDerivatives(SimStruct *S)
  {
  }
#endif /* MDL_DERIVATIVES */

static void mdlTerminate(SimStruct *S)
{
}
/*=============================*
 * Required S-function trailer *
 *=============================*/

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c"  /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"   /* Code generation registration function */
#endif
```

CONTROL CALCULATION – SINE/COSINE METHOD

```
/* u_ff.c
 *
 * S-function to calculate the feedforward control signals
 * using the disturbance effect and the disturbance-free model.
 * Recursive Least Squares is used to converge on the feedforward
 * control signals' sine & cosine coefficients.
 *
 * Some of the elements of this S-Function were modeled after
 * or taken from code developed by Neil Goodzeit (formerly at Princeton
 * University) in his file named "ff_fast.c", but were enhanced
 * to allow more than two control inputs.
 *
 * David Marco (NPS) supplied a C-code routine to do matrix inversion
 * using the LU decompostion method.
 */

#define S_FUNCTION_NAME u_ff
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

#define FLAG_CONTROL(S) ssGetSFcnParam(S,0)
#define FLAG_UPDATE(S)  ssGetSFcnParam(S,1)
#define FLAG_FILTER(S)  ssGetSFcnParam(S,2)
#define BETA(S)         ssGetSFcnParam(S,3)
#define P_INIT(S)       ssGetSFcnParam(S,4) /* Covariance */
#define LAMBDA(S)       ssGetSFcnParam(S,5) /* Forgetting Factor */
#define DT(S)           ssGetSFcnParam(S,6)
#define M(S)            ssGetSFcnParam(S,7) /* Number of Inputs */
#define NFREQ(S)        ssGetSFcnParam(S,8)
#define FREQ(S)         ssGetSFcnParam(S,9)

#define N_PARAM         10    /* Number of S-Function parameters */
#define MAX_COEF        36    /* Maximum number of feedforward
                                 coefficients = 2*m*MAX_FREQ */
#define MAX_FREQ        3     /* Maximum number of controlled
                                 frequencies */
#define MAX_2_FR        6     /* Two times the max number of
                                 frequencies */
#define MAX_CNTL        6     /* Maximum number of feedforward
                                 controls */
#define WK_COEF         1296  /* The work vector location of the
                                 feedforward coefficients
                                 (MAX_COEF^2 elements) */
#define WK_ANGL         1332  /* The work vector location of the
                                 feedforward angles
                                 (MAX_FREQ elements) */
#define WK_SC           1335  /* The work vector location of the sines
                                 and cosines (2*p*NFREQ elements) */
```

```
#define N_IWORK         2     /* Number of integer work vector
                                 elements */
#define TWO_PI          6.283185307


static void mdlInitializeSizes(SimStruct *S)
{
      int_T p, n_in, n_out1, n_out2, n_out3, n_work;
      int_T nctl  =mxGetPr(M(S))[0]; /* The number of ff controls */
      int_T q     =mxGetM(BETA(S));  /* Number of rows of Beta = q */
      int_T n_col =mxGetN(BETA(S));  /* Number of columns of Beta */

      p           = n_col/nctl;      /* ARX model order */
      n_in        = q;
      n_out1      = nctl;            /* feeforward signals */
      n_out2      = q;               /* residuals */
      n_out3      = MAX_COEF;        /* feedforward coefficients */
      n_work      =
            MAX_COEF*MAX_COEF + MAX_COEF + MAX_FREQ + 2*p*MAX_FREQ;

      ssSetNumSFcnParams(S, N_PARAM);
      if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
      {
          /* Return if expected number != actual number */
          return;
      }

      ssSetNumContStates(S, 0);
      ssSetNumDiscStates(S, 0);

      if (!ssSetNumInputPorts(S, 1)) return;
      ssSetInputPortWidth(S, 0, n_in);
                        /* Input = Disturbance Effect (q channels) */
      ssSetInputPortDirectFeedThrough(S, 0, 1);

      if (!ssSetNumOutputPorts(S, 3)) return;
      ssSetOutputPortWidth(S, 0, n_out1);
      ssSetOutputPortWidth(S, 1, n_out2);
      ssSetOutputPortWidth(S, 2, n_out3);
      ssSetNumSampleTimes(S, 1);
      ssSetNumRWork(S, n_work);
      ssSetNumIWork(S, N_IWORK);
      ssSetNumPWork(S, 0);
      ssSetNumModes(S, 0);
      ssSetNumNonsampledZCs(S, 0);

      ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* Function: mdlInitializeSampleTimes
 *    This function is used to specify the sample time(s) for your
 *    S-function. You must register the same number of sample times as
 *    specified in ssSetNumSampleTimes.
 */
```

```
static void mdlInitializeSampleTimes(SimStruct *S)
{
      real_T   tstep    =mxGetPr(DT(S))[0];

      ssSetSampleTime(S, 0, tstep);
      ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)

static void mdlInitializeConditions(SimStruct *S)
{
      int_T       i;
      int_T       *IWork      =ssGetIWork(S);
      real_T      *RWork      =ssGetRWork(S);
      real_T      p_init      =mxGetPr(P_INIT(S))[0];

/*
 * Initilize covariance matrix, feedforward coef,
 * and feedforward angles
 */

/* Zero the real work vector elements */
    for ( i = 0 ; i < ssGetNumRWork(S) ; i++ )
         RWork[i] = 0.0;

/* Initialize the cov matrix diagonal elements */
    for ( i = 0 ; i < MAX_COEF ; i++ )
         RWork[i+MAX_COEF*i] = p_init;

/* Zero the integer work vector elements */
    for ( i = 0 ; i < ssGetNumIWork(S) ; i++ )
         IWork[i] = 0;

}
#endif /* MDL_INITIALIZE_CONDITIONS */



/* Function: mdlOutputs
 *    In this function, you compute the outputs of your S-function
 *    block.
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{

/*************** Type declarations **********************************/

      InputRealPtrsType  u          =ssGetInputPortRealSignalPtrs(S,0);
      real_T             *y_cont    =ssGetOutputPortRealSignal(S,0);
      real_T             *y_res     =ssGetOutputPortRealSignal(S,1);
      real_T             *y_coef    =ssGetOutputPortRealSignal(S,2);
```

```
    real_T      *RWork      =ssGetRWork(S)                ;
    real_T      *freq       =mxGetPr(FREQ(S))             ;
    real_T      *beta       =mxGetPr(BETA(S))             ;
    real_T      lambda      =mxGetPr(LAMBDA(S))[0]        ;
    real_T      tstep       =mxGetPr(DT(S))[0]            ;

    int_T       *IWork      =ssGetIWork(S)                ;
    int_T       n_row       =mxGetM(BETA(S))              ;
    int_T       n_col       =mxGetN(BETA(S))              ;
    int_T       nctl        =mxGetPr(M(S))[0]             ;
    int_T       nfreq       =mxGetPr(NFREQ(S))[0]         ;
    int_T       on_off      =mxGetPr(FLAG_CONTROL(S))[0]  ;
    int_T       enable      =mxGetPr(FLAG_UPDATE(S))[0]   ;

    int_T       enable_upd                  ;
    int_T       counter                     ;  /* The enable counter */
    int_T       i, j, k, l, m               ;  /* Loop indices */
    int_T       idx, idx_c, idx_s, idx_f;  /* Work indexes */
    int_T       p                           ;  /* The ARX model order */

/* Temporary variables */

    real_T      angle[MAX_2_FR]             ;  /* ff sines/cosines */
    real_T      coef[MAX_CNTL][MAX_COEF];  /* The coefficient matrix
                                              for the feedforward
                                              recursion */
    real_T      ff_angle                    ;  /* The ff angle */
    real_T      c_angle_k                   ;  /* The cosine of shifted
                                              feedforward angle */
    real_T      s_angle_k                   ;  /* The sine of shifted
                                              feedforward angle */
    real_T      c_temp                      ;  /* Temporary storage for
                                              cosine */
    real_T      s_temp                      ;  /* Temporary storage for
                                              sine   */
    real_T      y_k[MAX_CNTL]               ;
    real_T      res[MAX_CNTL]               ;
    real_T      gain[MAX_COEF][MAX_CNTL];  /* The update gains */
    real_T      temp[MAX_COEF][MAX_CNTL];  /* P*coef' */
    real_T      det                         ;
    real_T      p_del                       ;  /* The update to cov
                                              matrix elements */
    int_T       ncoef                       ;

/* Variables for matrix inversion */

    int_T       n, ki, sing_flag            ;
    real_T      b, b1, b2                   ;
    real_T      a_local[MAX_CNTL+1][MAX_CNTL+1];    /* [7][7] */
    real_T      ainv[MAX_CNTL+1][MAX_CNTL+1];       /* [7][7] */
```

```
/*  Determine the ARX model order */
    p = n_col / nctl ;


/*********** Evaluate the enable logic ***************************/


    IWork      = ssGetIWork(S);    /* pointer to integer work vector */
    enable_upd = IWork[0]     ;
    counter    = IWork[1]     ;

    if ( enable != enable_upd )
    {
      if (enable == 0 )
      {
            enable_upd = 0 ;         /* Disable the coefficient updates
                                        immediately when commanded */
            counter    = 0 ;         /* Reset enable delay counter */
      }

      if (enable == 1 )
      {
            if ( counter > p )
                  enable_upd = 1 ;   /* Enable logic after a
                                        p step delay */
            else
                  counter += 1 ;     /* Otherwise increment
                                        the counter */
      }
    }                                /* End of enable logic */

    IWork[0] = enable_upd ;
    IWork[1] = counter    ;

/*********** Zero the feedforward equation coefficients ************/
    for ( i = 0 ; i < MAX_CNTL ; i++ )
    {
      for ( j = 0 ; j < MAX_COEF ; j++ )
            coef[i][j] = 0.0 ;       /* Zero the feedforward
                                        equation coefficients */
    }

/*******  Calculate the feedforward angles ************************/
    for ( i = 0 ; i < nfreq ; i++ )
    {
      idx_f              = 2 * i ;
      ff_angle           = RWork[WK_ANGL + i] ;
      ff_angle          += freq[i] * tstep;
      if ( ff_angle > TWO_PI )
               ff_angle -= TWO_PI;
      angle[idx_f    ]   = cos(ff_angle);
      angle[idx_f + 1]   = sin(ff_angle);
      RWork[WK_ANGL + i] = ff_angle;
```

```
/*******  Calculate the linear equation coefficients ***************/

      idx_c = 2 * i * nctl;          /* starting column index in coef
                                        matrix for cosine terms */
      idx_s = idx_c + nctl;          /* starting column index in coef
                                        matrix for sine terms */

      for ( k = 0 ; k < p ; k++ )
      {

          if ( k != 0 )
          {
            c_temp = c_angle_k;
            s_temp = s_angle_k;
          }

          if ( k == 0 )
          {
            c_temp = angle[idx_f  ];
            s_temp = angle[idx_f+1];
          }

          c_angle_k = RWork[WK_SC + k + idx_f*p    ]; /* Retrieve
                                                         cosines from
                                                         the table */

          s_angle_k = RWork[WK_SC + k + (idx_f+1)*p]; /* Retrieve
                                                         sines from
                                                         the table */

          RWork[WK_SC + k + idx_f*p    ] = c_temp;    /* Shift the
                                                         cos table */

          RWork[WK_SC + k + (idx_f+1)*p] = s_temp;    /* Shift the
                                                         sine table */

          idx = k * nctl ; /* starting column index for beta matrix */

          for ( j = 0 ; j < n_row ; j++ )
          {
            for ( l = 0 ; l < nctl ; l++ )
            {
              coef[j][idx_c+l] += beta[j+(idx+l)*n_row] * c_angle_k;
              coef[j][idx_s+l] += beta[j+(idx+l)*n_row] * s_angle_k;
            }
          }

        }        /* End loop for p */

      }    /* End loop for nfreq */
```

```
/************ Update the feedforward control coefficients **********/

/***************** Calculate the eta residual *********************/
    ncoef = nctl * 2 * nfreq ;    /*  Number of feedforward
                                      coefficients to solve for */

    for ( i = 0 ; i < n_row ; i++ )
    {
      y_k[i] = 0.0;
      for ( j = 0 ; j < ncoef ; j++ )
            y_k[i] += coef[i][j] * RWork[WK_COEF + j];
                  /* The expected value of the disturbance effect */

      res[i] = *u[i] - y_k[i];              /* Calculate the residuals */
      y_res[i] = res[i];                    /* Output the residuals */
    }

/************ Calculate the update gain matrix *********************/
    for ( i = 0 ; i < ncoef ; i++ )
    {
      for ( j = 0 ; j < n_row ; j++ )
      {
             temp[i][j] = 0.0;
             for ( k = 0 ; k < ncoef ; k++ )
                 temp[i][j] += RWork[i + MAX_COEF*k] * coef[j][k] ;
                                  /* P*fi = P*coef' (ncoef x n_row) */
      }
    }

/* Prepare matrix "a_local" (fi'*P*fi + lambda*eye(n_row, n_row)) for
 * inversion.  This routine uses indeces from 1..n (as opposed to
 * 0..n-1), so the first row and column will be zeroed out before
 * proceding. */

    for ( i = 0 ; i <= n_row ; i++ )
      a_local[0][i]     = 0.0;

    for ( i = 1 ; i <= n_row ; i++ )
      a_local[i][0]     = 0.0;

    for ( i = 1 ; i <= n_row ; i++ )
    {
      for ( j = 1 ; j <= n_row ; j++ )
      {
             a_local[i][j] = 0.0;
             if (i == j)
                  a_local[i][j] = lambda;

             for ( k = 0 ; k < ncoef ; k++ )
                  a_local[i][j] += coef[i-1][k] * temp[k][j-1];
                        /*  fi'*P*fi + lambda*eye(n_row, n_row)     */
                        /*  = coef*P*coef'+lambda*eye(n_row, n_row) */
      }
    }
```

```
/*  Calculate matrix inverse and update gain only if logic is enabled.
 *  The matrix being inverted is equivalent to:
 *              inv(fi'*P*fi + lambda*eye(n_row, n_row))
 */

    if ( enable_upd == 1 )
    {
      sing_flag = 0;
      n = n_row;

      for (i=1;i<=n;++i)
      {
         for (j=1;j<=n;++j)
         {
            ainv[i][j] = 0.0;
         }
      }

      for (i=1;i<=n;++i)
      {
         ainv[i][i] = 1.0;
      }

      for (k=1;k<=n-1;++k)
      {
         b = a_local[k][k];
         ki = k;

         for (i=k+1;i<=n;++i)
         {
            if( (fabs(b) - fabs(a_local[i][k])) >= 0.0 )
            {
            }
            else
            {
               b = a_local[i][k];
               ki = i;
            }
         }

         if( fabs(b) < 0.0001)
         {
            sing_flag = 1;
            break;
         }

         if( (ki-k) == 0)
         {
         }
         else
         {
            for (j=k;j<=n;++j)
```

```
            {
               b1 = a_local[k][j];
               a_local[k][j] = a_local[ki][j];
               a_local[ki][j] = b1;
            }
            for (j=1;j<=n;++j)
            {
               b2 = ainv[k][j];
               ainv[k][j] = ainv[ki][j];
               ainv[ki][j] = b2;
            }
         }

         for (j=k+1;j<=n;++j)
         {
            a_local[k][j] = a_local[k][j]/b;
         }

         for (j=1;j<=n;++j)
         {
            ainv[k][j] = ainv[k][j]/b;
         }

         for (i=k+1;i<=n;++i)
         {
            for (j=k+1;j<=n;++j)
            {
               a_local[i][j] = a_local[i][j] -
                                    a_local[i][k]*a_local[k][j];
            }

            for (j=1;j<=n;++j)
            {
               ainv[i][j] = ainv[i][j] - a_local[i][k]*ainv[k][j];
            }
         }
      }

      if(sing_flag == 0)
      {
         for (j=1;j<=n;++j)
         {
            ainv[n][j] = ainv[n][j]/a_local[n][n];
         }

         for (k=n-1;k>=1;--k)
         {
            for (j=1;j<=n;++j)
            {
               for (i=k+1;i<=n;++i)
               {
                  ainv[k][j] = ainv[k][j] - a_local[k][i]*ainv[i][j];
               }
            }
```

```
         }
      }
      else
      {
/*          mexPrintf("Singular or Ill-Conditioned Matrix\n");      */
      }



/*
 *  MATRIX INVERSION COMPLETE *
 *  Now that Inversion is done, continue with Gain update
 * (Remember that *ai pointer is 7x7, not 6x6 (when m=6))
 */

      for ( i = 0 ; i < ncoef ; i++ )
      {
            for ( j = 0 ; j < n_row ; j++ )
            {
               gain[i][j] = 0.0;

               for ( k = 0 ; k < n_row ; k++ )
                  gain[i][j] += temp[i][k] * ainv[k+1][j+1];
            /* L=P*fi'*inv(fi'*P*fi + lambda*eye(n_row, n_row)) */

            }
      }
  }
/* End coefficient update enable (starts prior to matrix inversion) */


/* Update the parameter estimates */
    for ( i = 0 ; i < ncoef ; i++ )
    {
      if ( enable_upd == 1 )
      {
            for ( j = 0 ; j < n_row ; j++ )
                  RWork[WK_COEF + i] += gain[i][j] * res[j];

      }     /* End coefficient update enable */

      y_coef[i] = RWork[WK_COEF + i];
            /* Put the coefficients in the output vector */
    }

    for ( i = ncoef ; i < MAX_COEF ; i++ )
      y_coef[i] = 0.0;

/************ Update the covariance matrix ************************/
    if ( enable_upd == 1 )
    {
      for ( i = 0 ; i < ncoef ; i++ )
      {
```

```
            for ( j = i ; j < ncoef ; j++ )
            {
                  p_del = 0.0;

                  for ( k = 0 ; k < n_row ; k++ )
                        p_del += gain[i][k] * temp[j][k];

                  RWork[i + MAX_COEF*j] -= p_del;
                  RWork[i + MAX_COEF*j] /= lambda;

                  if ( i != j )
                        RWork[j + MAX_COEF*i] = RWork[i + MAX_COEF*j];
            }
      }
    }     /*End coefficient update enable */

/**** Calculate the feedforward control, if enabled (else = 0) ****/
    if ( on_off == 1 )
    {
      for ( i = 0 ; i < nctl ; i++ )
      {
            y_cont[i] = 0.0;

            for ( j = 0 ; j < 2*nfreq ; j++ )
            {
                  idx = j * nctl + i;
                  y_cont[i] += RWork[WK_COEF + idx] * angle[j];
            }
      }
    }
    else
    {
      for ( i = 0 ; i < nctl ; i++ )
      {
            y_cont[i] = 0.0;
      }
    }

} /******* End of mdlOutputs *******/

#define MDL_UPDATE
#if defined(MDL_UPDATE)
  static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */

#define MDL_DERIVATIVES  /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
  static void mdlDerivatives(SimStruct *S)
  {
  }
#endif /* MDL_DERIVATIVES */
```

```
static void mdlTerminate(SimStruct *S)
{
}

/*=============================*
 * Required S-function trailer *
 *=============================*/

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c"  /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"   /* Code generation registration function */
#endif
```

CONTROL CALCULATION – ADAPTIVE BASIS METHOD

```
/* u_ff_eta.c
 *
 * S-function to calculate the feedforward control signals
 * using the Clear Box Adaptive Basis Method.  Recursive
 * Least Squares is used to converge on the feedforward
 * control signals' coefficients.
 *
 * This version employs filtering of the eta signal to remove
 * frequencies associated with modes de-selected for control.
 *
 * David Marco (NPS) supplied a C-code routine to do matrix inversion
 * using the LU decompostion method.
 */

#define S_FUNCTION_NAME u_ff_eta_N6
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

#define FLAG_CONTROL(S) ssGetSFcnParam(S,0)
#define FLAG_UPDATE(S)  ssGetSFcnParam(S,1)
#define FLAG_FILTER(S)  ssGetSFcnParam(S,2)
#define BETA(S)         ssGetSFcnParam(S,3)
#define P_INIT(S)       ssGetSFcnParam(S,4)
#define LAMBDA(S)       ssGetSFcnParam(S,5)
#define DT(S)           ssGetSFcnParam(S,6)
#define BASIS(S)        ssGetSFcnParam(S,7)
#define FILT_ORDER(S)   ssGetSFcnParam(S,8)
#define FILT_ALPHA(S)   ssGetSFcnParam(S,9)
#define FILT_BETA(S)    ssGetSFcnParam(S,10)
#define N_PARAM         11    /* Number of S-Function parameters */
#define M               6     /* m = Number of inputs */
#define Q               6     /* q = Number of outputs */
#define N               6     /* N = Number of basis sets needed =
                                 2*Number of controlled frequencies */
#define MAX_DEL         72    /* Maximum amout of delay used in eta
                                 history =
                                 L[N] + p_max + 2 = 30 + 40 + 2  */
#define N_COEF          36    /* Number of ff coefficients=(m*N) */
#define WK_COEF         1296  /* The work vector location of the ff
                                 coefficients (N_COEF^2) */
#define WK_ETA          1332  /* Start of the time history of the
                                 disturbance effect, eta
                                 (N_COEF^2 + N_COEF) */
#define WK_FILT_ETA     1764  /* Start of the time history of the
                                 filtered eta signal */
#define N_IWORK         2     /* Number of integer work vector elems */

/* Function: mdlInitializeSizes */
static void mdlInitializeSizes(SimStruct *S)
{
```

```
    int_T p, n_in, n_out1, n_out2, n_out3, n_work   ;
    int_T pm          =mxGetN(BETA(S))              ;
    int_T delay[N]    ={2, 7, 9, 12, 16, 22}        ;
    int_T l_max       =delay[N-1]                   ;

    p     = pm/M                                    ;
    n_in  = Q                                       ;
    n_out1      = M                                 ;
    n_out2      = Q                                 ;
    n_out3      = N_COEF                            ;
    n_work      = WK_FILT_ETA + MAX_DEL             ;     /* */

/* Variable Definitions
 *
 *    delay       Basis time shifts - defined here and in mdlOutputs
 *    l_max       Maximum amount of time history delay
 *    p           ARX model order
 *    n_in        Number of inputs to S-function
 *    n_out1      Number of elements in first output vector
 *    n_out2      Number of elements in second output vector
 *    n_out3      Number of elements in third output vector
 *    n_work      Number of work vector elements; covariance,
 *                coefficients, unfiltered eta time histories for all
 *                six struts, and filtered scalar eta history for basis
 *                strut - see comments at end of declarations in
 *                mdlOutputs function.
 */

    ssSetNumSFcnParams(S, N_PARAM);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
    {
        return; /* Return if expected number != actual number */
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, n_in);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 3)) return;
    ssSetOutputPortWidth(S, 0, n_out1);
    ssSetOutputPortWidth(S, 1, n_out2);
    ssSetOutputPortWidth(S, 2, n_out3);
    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, n_work);
    ssSetNumIWork(S, N_IWORK);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}
```

```
/* Function: mdlInitializeSampleTimes */
static void mdlInitializeSampleTimes(SimStruct *S)
{
      real_T      dt    =mxGetPr(DT(S))[0];

      ssSetSampleTime(S, 0, dt);
      ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
#if defined(MDL_INITIALIZE_CONDITIONS)
static void mdlInitializeConditions(SimStruct *S)
{
      int_T       i;
      int_T       *IWork            =ssGetIWork(S);
      real_T      *RWork            =ssGetRWork(S);
      real_T      p_init            =mxGetPr(P_INIT(S))[0];

/*
 * Initilize the cov matrix, feedforward coef, and feedforward angles
 */

/* Zero all real work vector elements */
    for ( i = 0 ; i < ssGetNumRWork(S) ; i++ )
              RWork[i] = 0.0;

/* Initialize the cov matrix diagonal elements */
    for ( i = 0 ; i < N_COEF ; i++ )
         RWork[i+N_COEF*i] = p_init;

/* Zero the integer work vector elements */
    for ( i = 0 ; i < ssGetNumIWork(S) ; i++ )
         IWork[i] = 0;
}
#endif /* MDL_INITIALIZE_CONDITIONS */

/* Function: mdlOutputs */
static void mdlOutputs(SimStruct *S, int_T tid)
{
/*************** Type declarations *********************************/

      InputRealPtrsType  u          =ssGetInputPortRealSignalPtrs(S,0);
      real_T             *y_cont    =ssGetOutputPortRealSignal(S,0);
      real_T             *y_res     =ssGetOutputPortRealSignal(S,1);
      real_T             *y_coef    =ssGetOutputPortRealSignal(S,2);

      real_T      *RWork      =ssGetRWork(S)          ;
      real_T      *beta       =mxGetPr(BETA(S))       ;
      real_T      lambda      =mxGetPr(LAMBDA(S))[0]  ;
      real_T      max_del;    /* Maximum delay of eta history */
      real_T      dt          =mxGetPr(DT(S))[0]      ;

      int_T         *IWork    =ssGetIWork(S)          ;
      int_T basis   =mxGetPr(BASIS(S))[0]       ;  /* Basis strut # */
```

```
      int_T pm      =mxGetN(BETA(S))            ;
      int_T delay[N]={2, 7, 9, 12, 16, 22}      ;  /* Time shifts */
      int_T l_max   =delay[N-1]                 ;  /* Max delay */
      int_T on_off  =mxGetPr(FLAG_CONTROL(S))[0];
      int_T enable  =mxGetPr(FLAG_UPDATE(S))[0] ;
             /* The externally commanded coefficient update enable */

      int_T enable_upd;
            /* The internal feedforward coefficient update enable */

      int_T counter;
      int_T       i, j, k, l  ;  /* Loop indices */
      int_T       p           ;  /* ARX model order */

/* Variables for eta filtering */
      int_T       filt_enable=mxGetPr(FLAG_FILTER(S))[0];
                        /* The enable for the filtering of eta */

      int_T       filt_ord   =mxGetPr(FILT_ORDER(S))[0];
                        /* Order of ARX model for eta filter */

      real_T      *filt_beta =mxGetPr(FILT_BETA(S));
      real_T      *filt_alpha=mxGetPr(FILT_ALPHA(S));
                        /* Pointers to filter's ARX model coeff's */

/* Temporary variables */

      real_T      Phi[N_COEF][Q]          ;  /* Phi matrix */
      real_T      eta_star[MAX_DEL+1]     ;  /* Scalar eta history */
      real_T      eta_hat[M]              ;  /* Estimated eta */
      real_T      res[M]                  ;
            /* residual = -eta + eta_hat  = y - y_hat  ( y = -eta ) */

      real_T      gain[N_COEF][Q]         ;
            /* The update gain matrix = P*Phi*inv[Phi'*P*Phi+R] */

      real_T      temp[N_COEF][Q]         ;  /* P*Phi */
      real_T      det                     ;
      real_T      p_del                   ;
            /* The update to each covariance matrix element */

/* Variables for matrix inversion */
      int_T       n, ki, sing_flag            ;
      real_T      b, b1, b2                   ;
      real_T      a_local[Q+1][Q+1]           ;     /* [7][7] */
      real_T      ainv[Q+1][Q+1]              ;     /* [7][7] */

/*The RWork vector indeces are set up as follows:
 * RWork[0 to N_COEF^2-1] -- The feedforward coefficient cov matrix
 *                           (stored following Matlab convention)
 * RWork[last +1  to  last + N_COEF]      -- The feedforward coeff's
 * RWork[last +1  to  last + MAX_DEL*q]   -- The eta time history
 * RWork[last +1  to  last + MAX_DEL]     -- History of filtered eta
 */
```

```
/*  Determine the ARX model order */
    p = pm / M ;

/*********** Evaluate the enable logic ************/
    IWork      = ssGetIWork(S)  ;
    enable_upd = IWork[0]       ;
    counter    = IWork[1]       ;

    if ( enable != enable_upd )
    {
      if (enable == 0 )
      {
            enable_upd = 0 ;
            counter    = 0 ;
      }

      if (enable == 1 )
      {
      /* Enable control updates after a delay of (l_max+p) steps */
            if ( counter > l_max + p )
                  enable_upd = 1 ;

      /* Otherwise increment the counter */
            else
                  counter += 1 ;
      }

    }     /* End of enable logic */

    IWork[0] = enable_upd ;
    IWork[1] = counter    ;

    /********** Shift the eta time history ***********/

    max_del = l_max+p+2;
    for ( i = (max_del)*Q-1 ; i > Q-1 ; i-- )
      RWork[WK_ETA+i] = RWork[WK_ETA+i-Q];

    /************ Store the new eta data *************/

    for ( i = 0 ; i < Q ; i++ )
      RWork[WK_ETA+i] = *u[i];

    /********** Filter eta if enabled ***************/

    if ( filt_enable == 1 )
    {
    /* Shift filtered eta history, then calc. new filtered value */

      for ( i = max_del ; i > 0 ; i-- )
            RWork[WK_FILT_ETA+i] = RWork[WK_FILT_ETA+i-1];

      RWork[WK_FILT_ETA] = 0.0;
```

```
      for ( i = 1 ; i <= filt_ord ; i++ )
            RWork[WK_FILT_ETA] -= filt_alpha[i] * RWork[WK_FILT_ETA+i];

      for ( i = 0 ; i <= filt_ord ; i++ )
            RWork[WK_FILT_ETA] += filt_beta[i] *
                                    RWork[WK_ETA+Q*i+basis-1];

      /* Form Basis vector from filtered scalar eta basis */

      for ( i = 0 ; i < max_del ; i++ )
            eta_star[i] = RWork[WK_FILT_ETA+i];
    }

    else
    {
    /* Form scalar basis vector directly from unfiltered eta basis */

      for ( i = 0 ; i < max_del ; i++ )
            eta_star[i] = RWork[WK_ETA+Q*i+basis-1];
    }

/********* Calculate the phi matrix ***********/

    for ( l = 1 ; l <= N ; l++ )
    {
      for ( i = 0 ; i < Q ; i++ )               /* rows */
      {
            for ( j = 0 ; j < M ; j++ )         /* columns */
            {
                  Phi[(l-1)*M+j][i] = 0.0;

                  for ( k = 0 ; k < p ; k++ )
                  {
                      Phi[(l-1)*M+j][i] += beta[k*Q*M + j*Q +i]
                                           *eta_star[delay[l-1]+k+1];
                  }
            }
      }
    }

/*********** Calculate the eta residual ************/

    for ( i = 0 ; i < Q ; i++ )
    {
      eta_hat[i] = 0.0;

      for ( j = 0 ; j < N_COEF ; j++ )
            eta_hat[i] += Phi[j][i] * RWork[WK_COEF + j];
                  /* The expected value of the disturbance effect */

      res[i] = -(*u[i]) - eta_hat[i];
                  /* Calculate the residuals */

      y_res[i] = res[i];
```

```
                /* Output the residuals */
    }

/************ Calculate the update gain matrix *********************/

    for ( i = 0 ; i < N_COEF ; i++ )
    {
      for ( j = 0 ; j < Q ; j++ )
      {
             temp[i][j] = 0.0;

             for ( k = 0 ; k < N_COEF ; k++ )
                 temp[i][j] += RWork[i + N_COEF*k] * Phi[k][j];
                                        /* = P*Phi (N_COEF x Q) */
      }
    }

/* Prepare "a_local" (fi'*P*fi + lambda*eye(q, q)) for inversion.
 * This routine uses indeces from 1..n (as opposed to 0..n-1),
 * so the first row and column will be zeroed out before proceding.
 */

    for ( i = 0 ; i <= Q ; i++ )
      a_local[0][i]     = 0.0;

    for ( i = 1 ; i <= Q ; i++ )
      a_local[i][0]     = 0.0;

    for ( i = 1 ; i <= Q ; i++ )
    {
      for ( j = 1 ; j <= Q ; j++ )
      {
             a_local[i][j] = 0.0;
             if (i == j)
                  a_local[i][j] = lambda;

             for ( k = 0 ; k < N_COEF ; k++ )
                  a_local[i][j] += Phi[k][i-1] * temp[k][j-1];
                              /*  Phi'*P*Phi + lambda*eye(q, q)  */
      }
    }

/* Matrix Inversion code omitted - it is exactly the same as
 * that in the Sine/Cosine Method's code.
 */

/* Now that Inversion is done, continue with Gain update
 * (Remember that *ai pointer is 7x7, not 6x6 (when m=6)
 */
      for ( i = 0 ; i < N_COEF ; i++ )
      {
            for ( j = 0 ; j < Q ; j++ )
            {
            gain[i][j] = 0.0;
```

```
            for ( k = 0 ; k < Q ; k++ )
                  gain[i][j] += temp[i][k] * ainv[k+1][j+1];
                        /* L=P*fi'*inv(fi'*P*fi + lambda*eye(q, q)) */
                  }
         }
      }  /*  End coeff update enable (starts prior to matrix inv) */

/********* Update the parameter estimates ************/
    for ( i = 0 ; i < N_COEF ; i++ )
    {
      if ( enable_upd == 1 )
      {
            for ( j = 0 ; j < Q ; j++ )
                  RWork[WK_COEF + i] += gain[i][j] * res[j];

      }     /* End coefficient update enable */

      y_coef[i] = RWork[WK_COEF + i];
            /* Put the coefficients in the output vector */
    }

/********** Update the covariance matrix ************/
    if ( enable_upd == 1 )
    {
      for ( i = 0 ; i < N_COEF ; i++ )
      {
            for ( j = i ; j < N_COEF ; j++ )
            {
                  p_del = 0.0;
                  for ( k = 0 ; k < Q ; k++ )
                        p_del += gain[i][k] * temp[j][k];
                              /* Assumes cov matrix is symmetric   */
                              /* ie, temp[j][k] represents Phi'*P  */

                  RWork[i + N_COEF*j] -= p_del;
                  RWork[i + N_COEF*j] /= lambda;
                              /* Employ forgetting factor */

                  if ( i != j )
                        RWork[j + N_COEF*i] = RWork[i + N_COEF*j];
                                          /* Force Symmetry */
            }
      }
    }             /*End coefficient update enable */

/* Calculate the feedforward control, if enabled (else = 0) */
    if ( on_off == 1 )
    {
      for ( j = 0 ; j < M ; j++ )
      {
            y_cont[j] = 0.0;
            for ( i = 0 ; i < N ; i++ )
```

```
                    y_cont[j] -= RWork[WK_COEF + (i)*M +j] *
                                                eta_star[delay[i]];

              /* Note: Minus sign (-=) above is to counteract     */
              /* minus sign in summing block in Simulink diagram  */
        }
      }

      else
      {
        for ( j = 0 ; j < M ; j++ )
        {
              y_cont[j] = 0.0;
        }
      }
} /********** End of mdlOutputs ***************/

#define MDL_UPDATE
#if defined(MDL_UPDATE)
  static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */

#define MDL_DERIVATIVES
#if defined(MDL_DERIVATIVES)
  static void mdlDerivatives(SimStruct *S)
  {
  }
#endif /* MDL_DERIVATIVES */

static void mdlTerminate(SimStruct *S)
{
}


/*==============================*
 * Required S-function trailer *
 *==============================*/

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c"  /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"   /* Code generation registration function */
#endif
```

LIST OF REFERENCES

1. Goodzeit, N.E., *System and Disturbance Identification for Adaptive Disturbance-Rejection Control*, Ph.D. Dissertation #3019T, pp. 5-16, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, June 1998.

2. Widrow, B. and Stearns, S.D., *Adaptive Signal Processing*, pp. 288-294, Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

3. Elliott, S.J., Stothers, I.M., and Nelson, P.A., "A Multiple Error LMS Algorithm and Its Application to the Active Control of Sound and Vibration," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 10, pp. 1423-1434, October 1987.

4. Goodzeit, N.E., and Phan, M.Q., "Exact System Identification in the Presence of Completely Unknown Periodic Disturbances," Department of Mechanical and Aerospace Engineering Technical Report No. 2096, Princeton University, January 1997, Princeton, NJ. Also, *Journal of Guidance, Control, and Dynamics*, to appear.

5. Goodzeit, N.E., and Phan, M.Q., "A Clear-Box Adaptive System for Flexible Spacecraft Identification and Disturbance Rejection," *Journal of Vibration and Control*, in press.

6. Lueg, P., *Verfahren zur Dampfung von Schallschwingungen*, German Patent, DRP No. 655,508, Filed January 1933, Issued December 1937.

7. Olson, H.F. and May, E.G., "Electronic Sound Absorber," *Journal of the Acoustical Society of America*, Vol. 25, No. 6, pp. 1130-1136, 1953.

8. Olson, H.F., "Electronic Control of Noise, Vibration, and Reverberation," *Journal of the Acoustical Society of America*, Vol. 28, No. 5, pp. 966-972, 1956.

9. Jessel, M.J.M. and Mangiante, G.A., "Active Sound Absorbers in an Air Duct," *Journal of Sound and Vibration*, Vol. 23, No. 3, pp.383-390, 1972.

10. Swinbanks, M.A., "The Active Control of Sound Propogation in Long Ducts," *Journal of Sound and Vibration*, Vol. 27, No. 3, pp. 411-436, 1973.

11. Poole, J.H.B. and Leventhall, H.G. "An Experimental Study of Swinbanks' Method of Active Attenuation of Sound in Ducts," *Journal of Sound and Vibration*, Vol. 49, No. 2, pp. 257-266, 1976.

12. Poole, J.H.B. and Leventhall, H.G. "Active Attenuation of Noise in Ducts," *Journal of Sound and Vibration*, Vol. 57, No. 2, pp. 308-309, 1978.

13. Chaplin, G.G.B. and Smith, R.A., "The Sound of Silence – The Silencing of Diesel Exhausts by Out-of-Phase Cancellation Using a Microprocessor Has Now Been Achieved," *Engineering*, No. 218, pp. 672-673, 1978.

14. Meirovitch, L. and Oz, H., "Modal-Space Control of Large Flexible Spacecraft Possessing Ignorable Coordinates," *Journal of Guidance and Control*, Vol. 3, No. 6, pp.569-577, 1980.

15. Burdess, J.S. and Metcalfe, A.V., "The Active Control of Forced Vibration Produced by Arbitrary Disturbances," *Journal of Vibration, Acoustics, Stress, and Reliability in Design*, Vol. 107, pp. 33-37, 1985.

16. Lee, S. and Sinha, A., "Design of an Active Vibration Absorber," Journal of Sound and Vibration, Vol. 109, No. 2, pp. 347-352, 1986.

17. Takagami, T. and Jimbo, Y., "Study of an Active Vibration Isolation System – A Learning Control Method," *Journal of Low Frequency Noise and Vibration*, Vol. 4, No. 3, pp 104-119, 1985.

18. Owen, R.G. and Jones, D.I., "Multivariable Control of an Active Anti-Vibration Platform," IEEE Transactions on Magnetics, Vol. 22, No. 5, pp.523-525, 1986.

19. Anton, E. and Ulbrich, H., "Active Control of Vibrations in the Case of Asymmetrical High-Speed Rotors by Using Magnetic Bearings," *Journal of Vibration, Acoustics, Stress, and Reliability in Design*, Vol. 107, pp. 410-415, 1985.

20. Burrows, C.R. and Sahinkaya, M.N., "Vibration Control of Multi-Mode Rotor-Bearing Systems," *Proceedings of the Royal Society of London*, Vol. A 386, pp. 77-94, 1983.

21. Schafer, B.E. and Holzach, H., "Experimental Research on Flexible Beam Modal Control," *Journal of Guidance and Control*, Vol. 8, No. 5, pp. 597-604, 1985.

22. Bailey, T. and Hubbard, J.E. (Jr.), "Distributed Piezoelectric-Polymer Active Vibration Control of a Cantilevel Beam," Journal of Guidance and Control, Vol. 8, No. 5, pp.605-611, 1985.

23. Tanaka, N. and Kikushima, Y., "On the Supression of Ground Vibration by Active Force Controller (Supression of Exciting Force by Feedforward Control)," Bulletin of the JSME, Vol. 26, No. 215, pp. 839-847, 1983.

24. Goodwin, G.C. and Sin, K.S., *Adaptive Filtering, Prediction, and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

25. Widrow, B., Glover, J.R., Jr., McCool, J.M., Kaunitz, J., Williams, C.S., Hearn, R.J., Zeidler, J.R., Dong, E., Jr. and Goodlin, R.C., "Adaptive Noise Cancelling: Principles and Applications," *Proceedings of the IEEE*, Vol. 63, No. 12, pp. 1692-1716, 1975.

26. Widrow, B., McCool, J.M., Larimore, M.G. and Johnson, C.R., Jr., "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," *Proceedings of the IEEE*, Vol. 64, No. 8, pp. 1151-1162, 1976.

27. Widrow, B., and Stearns, S.D., *Adaptive Signal Processing*, pp. 99-116, Prentice-Hall, Englewood Cliffs, NJ, 1985.

28. Feintuch, P.L., "An Adaptive Recursive LMS Filter'" *Proceedings of the IEEE*, Vol. 64, No. 11, pp. 1622-1624, 1976.

29. Sundararajan, N., Williams, J.P. and Montgomery, R.C., "Adaptive Modal Control of Structural Dynamic Systems Using Recursive Lattice Filters," Journal of Guidance and Control, Vol. 8, No. 2, pp. 223-229, 1985.

30. Swanson, D.C., *Active Attenuation of Acoustic Noise Using Adaptive ARMAX Control*, Ph.D. Dissertation, Pennsylvania State University, University Park, PA, 1986.

31. Johnson, C.R. and Larimore, M.G., "Comments on and Additions to 'An Adaptive Recursive LMS Filter'," *Proceedings of the IEEE*, Vol. 65, No. 9, pp. 1399-1404, 1977.

32. Sommerfeldt, S.D. and Tichy, J., "Adaptive Vibration Control of Vibration Isolation Mounts, Using and LMS-Based Control Algorithm," Technical Report No. TR 98-007, p. 6, Applied Research Laboratory, Pennsylvania State University, State College, PA, November 1989.

33. Clarkson, P.M. and White, P.R., "Simplified Analysis of the LMS Adaptive Filter Using a Transfer Function Approximation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 7, pp. 987-993, 1987.

34. Elliott, S.J., Stothers, I.M., and Nelson, P.A., "A Multiple Error LMS Algorithm and Its Application to the Active Control of Sound and Vibration," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 10, pp. 1423-1434, October 1987.

35. Sievers, L.A. and von Flotow, A.H., "Comparison and Extensions of Control Methods for Narrow-Band Disturbance Rejection," *IEEE Transactions on Signal Processing*, Vol. 40, No. 10, pp. 2377-2391, October 1992.

36. Wang, A.K., and Ren, W., "Convergence Analysis of the Multi-Variable Filtered-x LMS Algorithm with Application to Active Noise Control," IEEE Transactions on Signal Processing, Vol. 47, No. 4, pp. 1166-1169, April 1999.

37. Phan, M., Horta, L.G., Juang, J.-N. and Longman, R.W., "Linear System Identification Via an Asymptotically Stable Observer," *Journal of Optimization Theory and Applications*, Vol. 79, No. 1, pp. 59-86, October 1993.

38. Phan, M., Horta, L.G., Juang, J.-N. and Longman, R.W., "Improvement of Observer/Kalman Filter Identification by Residual Whitening," *Journal of Vibrations and Acoustics*, Vol. 117, pp.232-239, 1995.

39. Juang, J.-N. *Applied System Identification*, Prentice-Hall, Englewood Cliffs, NJ, pp. 275-327, 1994.

40. Lim, T.W., Bosse, A. and Fisher, S., "Adaptive Filters for Real-Time System Identification and Control," Journal of Guidance, Control, and Dynamics, Vol. 20, No. 1, pp. 61-66, 1997.

41. Goodzeit, N.E., *System and Disturbance Identification for Adaptive Disturbance-Rejection Control*, Ph.D. Dissertation #3019T, p. 4, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, June 1998.

42. Collins, S.A. and von Flotow, A.H., "Active Vibration Isolation for Spacecraft," *42nd Congress of the International Astronautical Federation*, Montreal, October 1991.

43. Laskin, R.A. and Sirlin, S.W., "Future Payload Isolation and Pointing System Technology," *Journal of Guidance and Control*, Vol. 9, No. 4, pp. 469-477, July-August 1986.

44. Sirlin, S.W. and Laskin, R.A., "Payload Isolation and Precision Pointing for the 1990's," *Advances in the Astronautical Sciences*, Vol. 57, pp. 39-60, 1985.

45. Wilke, P.S., Johnson, C.D. and Grosserde, P.J., "GFO/Taurus Whole-Spacecraft Vibration Isolation System," Proceedings of the 12th Annual AIAA/USU Conference on Small Satellites, Logan, Utah September 1998.

46. Cunningham, D.C., "Performance/Sizing Tradeoffs in Active and Passive Launch Isolation," *Damping 1993*, San Francisco, CA, February 1993.

47. Lee-Glauser, G.J., *Vibration Control of Spacecraft and Space Structures from Lift-Off to On-Orbit Environments*, Ph.D. Dissertation, Clarkson University, 1994.

48. Agrawal, B.N., Bang, H. and Jones, E., "Application of Piezoelectric Actuators and Sensors in Vibration Control of Flexible Spacecraft Structures," *43rd Congress of the International Astronautical Federation*, Washington, DC, 28 August – 5 September 1992.

49. Kaplow, C.F. and Velman, J.R., "Application of an Active Local Vibration Isolation Concept to a Flexible Space Telescope," *Journal of Guidance and Control*, Vol. 3, No. 3, pp. 227-233, May-June 1980.

50. Neat, G.W., Laskin, R.A., Regner, J.W. and von Flotow, A.H., "Advanced Isolation/Precision Pointing Platform Testbed for Future Spacecraft Missions," *Advances in the Astronautical Sciences*, Vol. 86, pp. 37-55, 1994.

51. Rodden, J.J., Dougherty, H.J., Reschke, L.F., Hasha, M.D. and Davis, L.P., "Line-of-Sight Performance Improvement with Reaction-Wheel Isolation," Advances in the Astronautical Sciences, Vol. 61, pp. 71-84, 1986.

52. Spanos, J., Rahman, Z., and von Flotow, A.H., "Active Vibration Isolation on an Experimental Flexible Structure," *Proceedings of the SPIE Conference on Smart Structures and Intelligent Systems*, Paper #1917-60, Albuquerque, NM, February 1993.

53. Wilson, J.F. and Davis, L.P., "Viscous Damped Space Structure for Reduced Jitter," *Proceedings, 59th Shock and Vibration Symposium*, Huntsville, AL, October 1987.

54. Blackwood, G.H., Hyde, T.T., Miller, D.W., Crawley, E.F., Shao, M. and Laskin, R.A., "Stellar Interferometer Tracking Experiment (SITE): A Proposed Technology Demonstration Experiment," *44th Congress of the International Astronautical Federation*, Graz, Austria, October 1993.

55. Germann, L. and Gupta, A.A., "The Six-DOF, Magnetic Suspended Fine Steering Mirror," Proceedings of the Annual Rocky Mountain Guidance and Control Conference, pp. 155-167, Keystone, CO, February 1990.

56. Gupta, A.A. and Germann, L., "Precision Pointing and Inertial Line-of-Sight Stabilization Using Fine-Steering Mirror and Strap-Down Inertial Sensors," *Advances in the Astronautical Sciences*, AAS Paper 89-036, 1989.

57. Hain. H.L. and Miller, R., "Isolation Mounts for the HEAO-B X-Ray Telescope," *The Shock and Vibration Bulletin*, Vol. 48, Part 2, pp. 97-113, 1978.

58. Hamilton, B.J., Andrus, J.H. and Carter, D.R., "Pointing Mount with Active Vibration Isolation for Large Payloads," *Advances in the Astronautical Sciences*, Vol. 63, pp. 299-318, 1987.

59. Kekler, C.R., "ASPS Performance with Large Payloads Onboard the Shuttle Orbiter," *Journal of Guidance, Control and Dynamics*, Vol. 5, No. 1, pp. 32-36, 1980.

60. Laskin, R.A., Koph, E.H., Sirlin, S.W., Spanos, J.T. and Wiktor, P.J., "Reactionless Gimbal Actuation for Precision Pointing of Large Payloads," AAS/AIAA Astrodynamics Specialist Conference, Kalispel, MT, August 1987.

61. Sevaston, G.E., Socha, M.M. and Eisenman, A., "The Circumstellar Imaging Telescope Image Motion Compensation System: Ultra-Precise Control on the Space Station Platform," *Advances in the Astronautical Sciences*, Vol. 68, pp. 291-310, 1989.

62. Allen, T.S., Havenhill, D.D. and Kral, K.D., "FEAMIS: A Magnetically Suspended Isolation System for Space-Based Materials Processing," *Annual AAS Guidance and Controls Conference*, Keystone, CO, February 1986.

63. Fenn, R.C., Downer, J.R., Gondhalekar, V. and Johnson, B.G., "An Active Magnetic Suspension for Space-Based Microgravity Vibration Isolation," *ASME Winter Annual Meeting*, pp. 49-56, 1990.

64. Grodsinsky, C.M. and Brown, G.V., "Low Frequency Vibration Isolation Technology for Microgravity Space Experiments," *ASME Conference on Mechanical Vibration and Noise*, pp. 295-302, Montreal, September 1989.

65. Grodsinsky, C.M., "Development and Approach to Low-Frequency Microgravity Isolation Systems," NASA Technical Paper 2984, August 1990.

66. Grodsinsky, C.M., "Development to Demonstration of Active Magnetic Microgravity Isolation Systems," *24th International Conference on Environmental Systems*, SAE Paper 941418, Friedrichshafen, Germany, June 1994.

67. Hampton, R.D., Knospe, C.R. and Grodsinsky, C.M., "Controller Design for Microgravity Vibration Isolation Systems," *43rd Congress of the International Astronautical Federation*, Washington, DC, 28 August – 5 September 1992.

68. Jones, D.I., Owens, A.R. and Owen, R.G., "A Microgravity Facility for In-Orbit Experiments," *ASME Winter Annual Meeting*, pp. 76-73, Dallas, TX, 1990.

69. Lee-Glauser, G.J., *Vibration Control of Spacecraft and Space Structures from Lift-Off to On-Orbit Environments*, Ph.D. Dissertation, Clarkson University, 1994.

70. Mercadal, M., Blaurock, C.A., von Flotow, A.H. and Werely, N.M., "Spacecraft Micro-Gee Vibration Isolation with Enhanced Performance via Feedback Control," *42nd Congress of the International Astronautical Federation*, Montreal, 1991.

71. Sinha, A., Kao, C.K. and Grodsinsky, C.M., "A New Approach to Controller Design for Microgravity Isolation Systems," *Acta Astronautica*, Vol. 21, No. 11, pp. 771-775, 1990.

72. Sinha, A. and Wang, Y.-P., "Digital Control Algorithms for Microgravity Isolation Systems," *ASME Conference on Mechanical Vibration and Noise*, pp. 247-256, 1991.

73. Stampleman, D.S. and von Flotow, A.H., "Microgravity Isolation Mounts Based Upon Piezoelectric Film," *ASME Winter Annual Meeting*, pp. 57-65, Dallas, TX, 1990.

74. Stewart, D., "A Platform with Six Degrees of Freedom," *Proceedings of the Institute of Mechanical Engineering, London*, Vol. 180, Pt. I, pp. 371-386, 1965.

75. Anderson, E.H. and How, J.P., "Active Vibration Isolation Using Adaptive Feedforward Control," *Proceedings, 1997 American Control Conference*, Vol. 3, pp. 1783-1788, Albuquerque, NM, June 4-6 1997.

76. Anderson, E.H., Leo, D.J. and Holcomb, M.D., "Active System for Vibration Isolation of Spacecraft Instruments," *Advances in the Astronautical Sciences*, Vol. 92, pp. 465-479, 7-11 February 1996.

77. Geng, Z.J. and Haynes, L.S., "Six Degree-of-Freedom Vibration Control Using the Stewart Platforms," *IEEE Transactions on Control Systems Technology*, vol. 2, no. 1, pp. 45-53, March 1994.

78. Spanos, J. and Rahman, Z., "Control Concepts for Active Multi-axis Vibration Isolation," *Proceedings, 6th International Conference on Adaptive Strucures*, pp. 348-352, Key West, FL, November, 1995.

79. Spanos, J., Rahman, Z. and Blackwood, G., "A Soft 6-axis Active Vibration Isolator," Proceedings, American Control Conference, Vol. 1, pp. 412-416, Seattle, WA, June 1995.

80. Sullivan, J., Rahman, Z., Cobb, R. and Spanos, J., "Closed-Loop Performance of a Vibration Isolation and Supression System," Proceedings, American Control Conference, Vol. 6, pp. 3974-3978, Albuquerque, NM, June 1997.

81. Widrow, B. and Stearns, S.D., *Adaptive Signal Processing*, p. 323, Prentice-Hall, Englewood Cliffs, NJ, 1985.

82. Glover. J., *Adaptive Noise Cancelling of Sinusoidal Interferences*, Ph.D. Dissertation, Stanford University, Stanford, CA, 1975.

83. Widrow, B. and Stearns, S.D., *Adaptive Signal Processing*, p. 22, Prentice-Hall, Englewood Cliffs, NJ, 1985.

84. Widrow, B. and Stearns, S.D., *Adaptive Signal Processing*, pp. 59, Prentice-Hall, Englewood Cliffs, NJ, 1985.

85. Widrow, B. and Stearns, S.D., *Adaptive Signal Processing*, pp. 103, Prentice-Hall, Englewood Cliffs, NJ, 1985.

86. Stewart, D., "A Platform with Six Degrees of Freedom," *Proceedings of the Institute of Mechanical Engineering, London*, Vol. 180, Pt. I, pp. 371-386, 1965.

87. Elliott, S.J., Stothers, I.M., and Nelson, P.A., "A Multiple Error LMS Algorithm and Its Application to the Active Control of Sound and Vibration," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 10, pp. 1423-1434, October 1987.

88. Elliott, S.J., Stothers, I.M., and Nelson, P.A., "A Multiple Error LMS Algorithm and Its Application to the Active Control of Sound and Vibration," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 10, p. 1424, October 1987.

89. Clarkson, P.M. and White, P.R., "Simplified Analysis of the LMS Adaptive Filter Using a Transfer Function Approximation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 7, pp. 987-993, 1987.

90. Elliott, S.J., Stothers, I.M., and Nelson, P.A., "A Multiple Error LMS Algorithm and Its Application to the Active Control of Sound and Vibration," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 10, pp. 1423-1434, October 1987.

91. Sievers, L.A. and von Flotow, A.H., "Comparison and Extensions of Control Methods for Narrow-Band Disturbance Rejection," *IEEE Transactions on Signal Processing*, Vol. 40, No. 10, pp. 2377-2391, October 1992.

92. Wang, A.K., and Ren, W., "Convergence Analysis of the Multi-Variable Filtered-x LMS Algorithm with Application to Active Noise Control," IEEE Transactions on Signal Processing, Vol. 47, No. 4, pp. 1166-1169, April 1999.

93. Morgan, D.R. and Sanford, C., "A Control Theory Approach to the Stability and Transient Analysis of the Filtered-x LMS adaptive Notch Filter," *IEEE Trans. Signal Processing*, Vol. 40, September 1992.

94. Wang, A.K. and Ren, W., "Convergence Analysis of the Multi-Variable Filtered-X LMS Algorithm with Application to Active Noise Control," IEEE Transactions on Signal Processing, Vol. 47. No. 4, pp. 1166-1169, April 1999.

95. Douglas, S.C., "Analysis of the Multiple-Error and Block Least-Mean_Square Adaptive Algorithms," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 42, No. 2, pp. 92-101, February 1995.

96. Fuller, C.R., Elliott, S.J. and Nelson, P.A., *Active Control of Vibration*, p. 100, Academic Press Limited, London, 1996.

97. Goodzeit, N.E., *System and Disturbance Identification for Adaptive Disturbance-Rejection Control*, Ph.D. Dissertation #3019T, pp. 50-63, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, June 1998.

98 . Goodzeit, N.E., *System and Disturbance Identification for Adaptive Disturbance-Rejection Control*, Ph.D. Dissertation #3019T, p. 88, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, June 1998.

99 . Goodwin, G.C. and Sin, K.S., *Adaptive Filtering, Prediction, and Control*, p. 64, Prentice-Hall, Englewood Cliffs, NJ, 1984.

100 .Goodzeit, N.E., *System and Disturbance Identification for Adaptive Disturbance-Rejection Control*, Ph.D. Dissertation #3019T, pp. 205-211, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, June 1998.

101. Elliott, S.J., Stothers, I.M., and Nelson, P.A., "A Multiple Error LMS Algorithm and Its Application to the Active Control of Sound and Vibration," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 10, p. 1426, October 1987.

102. Geng, Z.J. and Haynes, L.S., "Six Degree-of-Freedom Vibration Control Using the Stewart Platforms," *IEEE Transactions on Control Systems Technology*, vol. 2, no. 1, pp. 45-53, March 1994.

103. Beavers, G.D., *System Identification of an Ultra Quiet Vibration Isolation Platform*, Master's Thesis, Naval Postgraduate School, Monterey, CA, June 1997.

104. Phan, M., Horta, L.G., Juang, J.-N., and Longman, R.W., "Linear System Identification Via an Asymptotically Stable Observer," *Journal of Optimization Theory and Applications*, Vol. 79, No. 1, pp. 59-86, October 1993.

105. Juang, J.-N., Phan, M., Horta, L.G., and Longman, R.W., "Identification of Observer/Kalman Filter Markov Parameters: Theory and Experiments," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 2, March-April 1993, pp. 320-329.

106. Phan, M., Horta, L.G., Juang, J.-N., and Longman, R.W., "Improvement of Observer/Kalman Filter Identification (OKID) by Residual Whitening," *Journal of Vibrations and Acoustics*, Vol. 117, April 1995, pp. 232-238.

107. Goodzeit, N.E., *System and Disturbance Identification for Adaptive Disturbance-Rejection Control*, Ph.D. Dissertation #3019T, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, June 1998.

108. Goodzeit, N.E., *System and Disturbance Identification for Adaptive Disturbance-Rejection Control*, Ph.D. Dissertation #3019T, pp. 205-211, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, June 1998.

109. Elliott, S.J., Stothers, I.M., and Nelson, P.A., "A Multiple Error LMS Algorithm and Its Application to the Active Control of Sound and Vibration," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-35, No. 10, p. 1424, October 1987.

110. Feintuch, P., Bershad, N.J., and Lo, A.K., "A frequency Domain Model for Filtered LMS Algorithm Stability Analysis, Design, and Elimination of the Training Mode," IEEE Transactions on Signal Processing, Vol. 41, pp. 1518-1531, April 1993.

111 .Douglas, S.C., "Analysis of the Multiple-Error and Block Least-Mean_Square Adaptive Algorithms," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 42, No. 2, pp. 92-101, February 1995.

112 .Sievers, L.A. and von Flotow, A.H., "Comparison and Extensions of Control Methods for Narrow-Band Disturbance Rejection," IEEE Transactions on Signal Processing, Vol. 40, No. 10, pp. 2377-2391, October 1992.

113 .Goodzeit, N.E., *System and Disturbance Identification for Adaptive Disturbance-Rejection Control*, Ph.D. Dissertation #3019T, Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, June 1998.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Prof. Brij Agrawal 2
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, CA 93943

4. Maj. Stephen Edwards 3
   21743 Pinewood Ct.
   Sterling, VA 20164

5. Prof. Gerald Lindsey 1
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, CA 93943

6. Prof. Isaac Kaminer 1
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, CA 93943

7. Prof. Roberto Cristi 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943

8. Prof. Joshua Gordis 1
   Department of Mechanical Engineering
   Naval Postgraduate School
   Monterey, CA 93943

9. Prof. Minh Phan 1
Dept. of Mechanical and Aerospace Engineering
Engineering Quadrangle
Princeton University
Princeton, NJ 08544

10. Prof. Richard Longman 1
220 Mudd Building
Dept. of Mechanical Engineering
Columbia University
New York, NY 10027

11. Dr. Neil Goodzeit 1
1163 Lincoln Ave
Palo Alto, CA 94301

12. Dr. Eric Anderson 1
CSA Engineering, Inc.
2565 Leghorn St.
Mountain View, CA 94043-1613

13. CAPT Tom McKannon 1
Room 44G06
National Reconnaissance Office
14675 Lee Road
Chantilly, VA 20151

14. Bradley Smith 1
Room 14A00L
National Reconnaissance Office
14675 Lee Road
Chantilly, VA 20151

15. Dr. R. Scott Erwin 1
AFRL/Space Vehicles Directorate
Bldg. 472, Room 266
3550 Aberdeen Ave. SE
Kirtland AFB, NM 87117-5776

16. Dr. Zahidul Rahman 1
Jet Propulsion Laboratory
Mail Stop 198-326
4800 Oak Grove Drive
Pasadena, CA 91109