NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

MARITIME ADAPTIVE OPTICS BEAM CONTROL

by

Melissa S. Corley

September 2010

Dissertation Supervisor: Brij N. Agrawal

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2010	**3. REPORT TYPE AND DATES COVERED** Dissertation	
4. TITLE AND SUBTITLE: Maritime Adaptive Optics Beam Control		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Melissa S. Corley			
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) N/A		**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. IRB Protocol number _____N/A_____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

The Navy is interested in developing systems for horizontal, near ocean surface, high-energy laser propagation through the atmosphere. Laser propagation in the maritime environment requires adaptive optics control of aberrations caused by atmospheric distortion. In this research, a multichannel transverse adaptive filter is formulated in Matlab's Simulink environment and compared to a complex lattice filter that has previously been implemented in large system simulations. The adaptive filters are used to augment a classical adaptive optics controller and are also compared to a Kalman filter augmenting a classical controller.

Additionally, the Naval Postgraduate School's first laboratory testbed to use adaptive optics for the compensation of atmospheric turbulence is designed and built. The control algorithms are evaluated both in simulation and in the presence of a laboratory-generated disturbance. Finally, effects of horizontal propagation through deep turbulence are created in the lab. Beam control algorithms are tested in this environment to draw initial conclusions about performance in deep turbulence.

For the system implemented in this research, the simple transverse filter in combination with a classical proportional-integral controller performs comparably to the complex lattice filter and the Kalman filter in a standard turbulence scenario and demonstrates more robust performance in the deep turbulence scenario. The adaptive optics testbed itself can be transitioned easily between traditional and deep turbulence scenarios and can support a wide range of atmospheric realizations for further beam control research.

14. SUBJECT TERMS Adaptive Optics, Deep Turbulence, Adaptive Filter, Maritime, Transverse, Lattice, LMS, RLS, Laboratory Testbed			**15. NUMBER OF PAGES** 131
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

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.

MARITIME ADAPTIVE OPTICS BEAM CONTROL

Melissa S. Corley
Captain, United States Air Force
B.S., Mechanical Engineering, Stanford University, 2004
M.S., Aeronautics and Astronautics, Stanford University, 2004
M.B.A., University of New Mexico, 2007

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2010

Author: _____
Melissa S. Corley

Approved by:

_____
Brij Agrawal
Distinguished Professor of
Mechanical and Aerospace
Engineering
Dissertation Supervisor

_____
Marcello Romano
Associate Professor of
Mechanical and Aerospace
Engineering

_____
Oleg Yakimenko
Professor of Mechanical and
Aerospace Engineering

_____
Roberto Cristi
Professor of Electrical and
Computer Engineering

_____
Jae Jun Kim
Research Assistant Professor of Mechanical and Aerospace Engineering

Approved by: _____
Knox Millsaps, Chair, Department of Mechanical and Aerospace Engineering

Approved by: _____
Douglas Moses, Vice Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Navy is interested in developing systems for horizontal, near ocean surface, high-energy laser propagation through the atmosphere. Laser propagation in the maritime environment requires adaptive optics control of aberrations caused by atmospheric distortion. In this research, a multichannel transverse adaptive filter is formulated in Matlab's Simulink environment and compared to a complex lattice filter that has previously been implemented in large system simulations. The adaptive filters are used to augment a classical adaptive optics controller and are also compared to a Kalman filter augmenting a classical controller.

Additionally, the Naval Postgraduate School's first laboratory testbed to use adaptive optics for the compensation of atmospheric turbulence is designed and built. The control algorithms are evaluated both in simulation and in the presence of a laboratory-generated disturbance. Finally, effects of horizontal propagation through deep turbulence are created in the lab. Beam control algorithms are tested in this environment to draw initial conclusions about performance in deep turbulence.

For the system implemented in this research, the simple transverse filter in combination with a classical proportional-integral controller performs comparably to the complex lattice filter and the Kalman filter in a standard turbulence scenario and demonstrates more robust performance in the deep turbulence scenario. The adaptive optics testbed itself can be transitioned easily between traditional and deep turbulence scenarios and can support a wide range of atmospheric realizations for further beam control research.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AF	Adaptive Filter
ALTB	Airborne Laser Test Bed
AO	Adaptive Optics
BILL	Beacon Illuminator Laser
CCD	Charge-Coupled Device
DM	Deformable Mirror
FIR	Finite Impulse Response
FX	Filtered-X
GUI	Graphical User Interface
HEL	High-Energy Laser
IIR	Infinite Impulse Response
KF	Kalman Filter
K-L	Karhunen-Loève
LC	Liquid Crystal
LMS	Least Mean Square
MIMO	Multiple-Input Multiple-Output
MMDM	Micromachined Membrane Deformable Mirror
MSE	Mean Square Error
NPS	Naval Postgraduate School
NRL	Naval Research Laboratory
NSLOT	Navy Surface Layer Optical Turbulence
PI	Proportional-Integral
RLS	Recursive Least Squares
RMS	Root Mean Square
SH	Shack-Hartmann
SLM	Spatial Light Modulator
SPAWAR	Space and Naval Warfare Systems Command
TILL	Track Illuminator Laser
TTM	Tip/Tilt Mirror
WFS	Wavefront Sensor

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

A task of this magnitude is never undertaken alone. I would like to thank the following people who paved my way through this research:

Dr. Brij Agrawal, for chairing my committee and providing academic support and the essential resources I needed in ideas, equipment, research facilities, and people.

The members of my committee: Dr. Roberto Cristi, Dr. Jae Jun Kim, Dr. Marcello Romano, and Dr. Oleg Yakimenko (Спасибо вам за то, что Вы в меня поверили с самого начала), for their wealth of knowledge, advice, and hours of discussions.

Dr. Masaki Nagashima, without whose hard work, patience, and encouragement—both technical and philosophical—I could not have done this.

Dr. Ty Martinez, for helpful conversations, insightful suggestions, and support on hardware that kept me on the right track.

Freddie Santiago, for endless patience and determination, building an optical system with me from scratch, many long days and nights in the lab, and encouraging me the whole way. ¡Mil gracias!

All my friends from the Starfire Optical Range, who gave me the first background knowledge and experience in this area.

Jean Ford, for being the best neighbor, friend, and horse caretaker I could have asked for.

Paul, Lauren, Emily, and Ian, for steadfast friendship and many adventures. Live long and prosper.

Dan Bursch, for long runs, inspiration, and teaching me so much about the meaning of life.

Above all, I thank my parents, grandparents, and my amazing sister (check with her for the English translation), whose advice, patience, love, encouragement, and constant support have made my work possible. Extra thanks to my sister and the mantis shrimp for getting me through the home stretch.

To all my advisors, mentors, friends, and family—you have my eternal gratitude for helping me get here.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Beam control of laser propagation systems is critical in imaging, laser communications, and high-energy laser applications. Atmospheric turbulence causes distortions in the optical phase of a propagating laser beam, leading to blurry images and reduced beam intensity on target. In order to compensate for these distortions, adaptive optics systems containing multiple actuator inputs and multiple sensor outputs must be employed. Control algorithms must therefore be multichannel and able to compensate for the varying nature of atmospheric turbulence.

While adaptive optics systems were first developed for astronomical applications, there has been a growing interest in using such systems in horizontal environments near the surface of the earth. The Navy is particularly interested in adaptive optics for use in high-energy laser (HEL) systems in a maritime environment. The horizontal and slant path turbulence in this environment present a different challenge from the vertical or near-vertical turbulence traditionally simulated and corrected for in astronomical adaptive optics. The turbulence in a horizontal path is often referred to as "deep" or "thick" turbulence as the entire propagation path is contained within the most dense layer of the atmosphere.

The focus of this research is to evaluate different advanced control techniques to simplify the control of a multiple-input multiple-output (MIMO) adaptive optics system, to build an adaptive optics testbed and implement the various control algorithms on the testbed, and to simulate the effects of deep turbulence in the laboratory to compare control algorithm performance in a maritime-like environment. This chapter introduces the background and challenges associated with adaptive optics beam control in a maritime environment, describes state-of-the-art research being performed in this area, and ends with a formulation of the problem investigated in this research.

A. ADAPTIVE OPTICS

The goal of adaptive optics (AO) is to correct for aberrations in imaging, communications, and laser propagation systems caused by turbulence or other

disturbances in the propagation medium. A typical adaptive optics system is shown in Figure 1. The three primary components of an AO system are a wavefront sensor to determine how the beam is distorted, a control computer to calculate the correction to be applied, and a corrective element, usually a deformable mirror (DM), to implement the applied commands (Tyson, 2000).

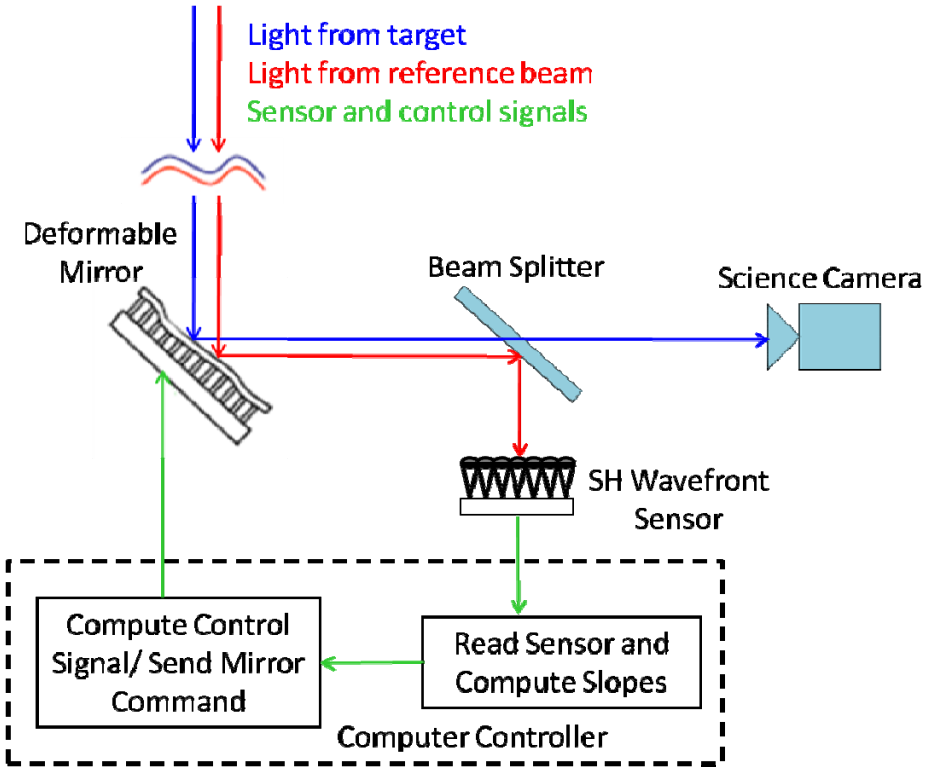

Figure 1. Typical adaptive optics system with Shack-Hartmann wavefront sensor

In this system, a reference beam that sees the same disturbance as the target is sent to the wavefront sensor. A Shack-Hartmann (SH) wavefront sensor (WFS) is an array of lenslets, which produces a grid pattern of spots on a detector such as a CCD camera. An ideal wavefront arriving from a point source in the far field is flat, producing a known grid pattern. An aberrated wavefront will produce some x and y offsets from the reference grid. These offsets can be related to the local slopes of the wavefront at each lenslet according to the equations (Tyson & Frazier, 2004):

$$\theta_x = \frac{\delta_x}{f}, \quad \theta_y = \frac{\delta_y}{f} \tag{1.1}$$

where θ is the slope angle in radians, δ is the centroid location difference in mm, and f is the lenslet focal length in mm.

Thus, the sensor measures *x* and *y* positions on the detector and the computer determines the slopes of the wavefront from the offsets. Using this information, it is possible to reconstruct the wavefront itself, or simply use the slope information directly to determine DM commands. In the latter case, the control computer uses the slope error referenced from zero to determine what commands to send to the deformable mirror to correct for disturbances. Finally, the mirror deforms according to the received commands and the process repeats, actively correcting for the changing turbulence. The goal is to drive the slope error to zero, hence to drive the wavefront to its ideal flat shape. Figure 2 shows a schematic of a SH WFS.

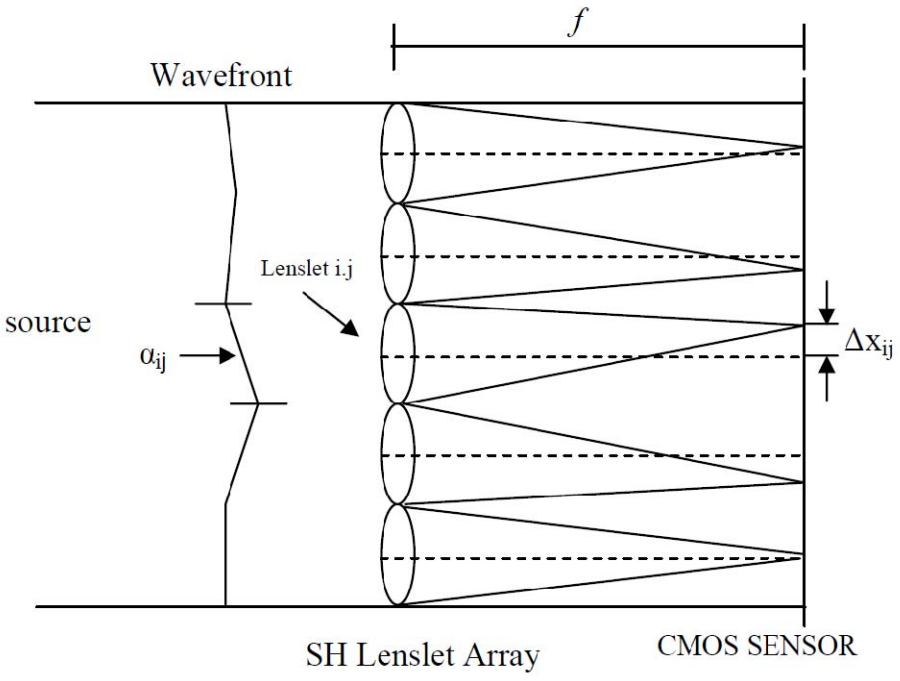

Figure 2. SH WFS schematic (From Allen, 2007)

B. CLASSICAL ADAPTIVE OPTICS CONTROL

The control computer determines mirror commands from Shack-Hartmann wavefront sensor measurements using the influence, or poke matrix. A calibration process prior to the experiment determines this poke matrix. Each individual DM actuator is "poked," or sent a maximum or near-maximum voltage, while all other actuators are held at zero. The slope measurements corresponding to each poked actuator form the columns of the poke matrix. Usually in adaptive optics systems, there are more slope measurements than actuators, forming a tall poke matrix. The structure of a typical poke matrix is:

$$\Gamma = \begin{bmatrix} x_{11} & \cdots & x_{1k} \\ x_{21} & \cdots & x_{2k} \\ \vdots & \cdots & \vdots \\ x_{n1} & \cdots & x_{nk} \\ y_{11} & \cdots & y_{1k} \\ y_{21} & \cdots & y_{2k} \\ \vdots & \cdots & \vdots \\ y_{n1} & \cdots & y_{nk} \end{bmatrix} \tag{1.2}$$

where n is the number of lenslets, the x and y slope measurements are stacked to form a vector of size $2n$ for each actuator, and k is the number of actuators. For a SH WFS using 60 lenslets and a 37-channel DM, the poke matrix is of size 120 x 37.

Once the poke matrix is determined, the system output and input are related as:

$$\mathbf{s} = \mathbf{\Gamma c} \tag{1.3}$$

where $\mathbf{s}$ is the vector of sensor slope outputs (both x and y measurements for each lenslet), $\mathbf{\Gamma}$ is the poke matrix, and $\mathbf{c}$ is the DM command vector. Using this relationship, it is possible to determine the DM commands that will minimize the sensor error at each timestep by using the pseudoinverse of the poke matrix, $\mathbf{\Gamma}^{\dagger}$. Since there are usually more sensor measurements than DM actuators, this forms the least squares solution to the wavefront correction problem.

Often in historical adaptive optics systems, it has been desirable to reconstruct the wavefront phase from the wavefront sensor slope information. However, this research is

primarily concerned with determining the DM commands that will correct for the measured error as quickly and efficiently as possible. Wavefront reconstruction itself is not necessary in this case, since a direct relationship exists between slopes and DM commands. However, the pseudoinverse of the poke matrix may still be referred to as the "reconstructor" in lieu of having an actual wavefront reconstruction step in the process. A basic diagram of a classical integral controller for AO is shown in Figure 3. The integral controller is written in the form:

$$\mathbf{c}_{new} = \mathbf{c}_{old} - g\boldsymbol{\Gamma}^{\dagger}\mathbf{s} \tag{1.4}$$

where g is the integral gain. This controller can also be a proportional-integral, or PI controller, and will be augmented using advanced beam control techniques.

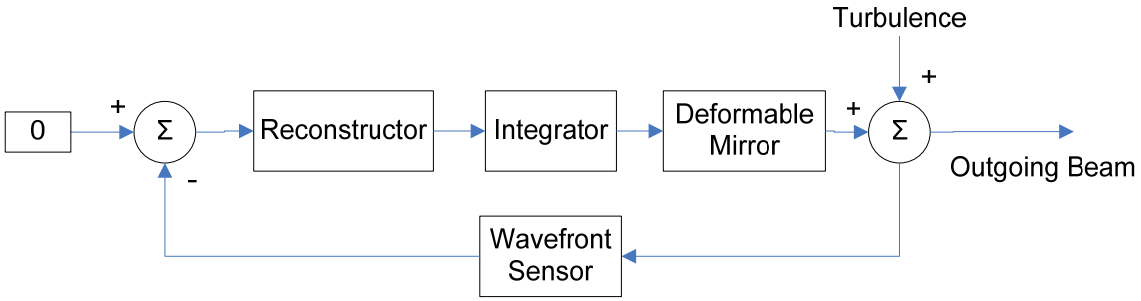

Figure 3. Classical AO control system

C. ADAPTIVE FILTER BACKGROUND

Previous adaptive optics work at the NPS has been applied primarily to vibration control and segment alignment for flexible space telescopes and segmented mirror systems (Allen, 2007; Burtz, 2009). Adaptive filters have been used by the NPS in the control of optical beam jitter (Watkins & Agrawal, 2007; Beerer, 2008), but this is the first time they will be used by the NPS for higher order compensation of atmospheric turbulence. Researchers at UCLA and the Starfire Optical Range have used multichannel filters in astronomical and Airborne Laser Test Bed (ALTB) adaptive optics systems, (Rhoadarmer, et al., 2006; Liu & Gibson, 2007) but this research focuses on the near-surface horizontal environment expected in maritime applications.

Adaptive filters have been used extensively over the past several decades in the field of active noise and vibration control. Unlike passive control, which can employ enclosures, silencers, or mass-spring-damper systems, active noise control employs secondary sources, usually electronic, to produce a canceling signal or wave that reduces the noise disturbance (Kuo, 1996). Two very familiar applications of active noise control include noise canceling headsets and higher end Bluetooth devices for cell phone use. Other common applications of noise control are found in industry, where it is important to reduce noise produced by machinery such as engines, blowers, fans, and compressors, and to reduce noise in ducts (Kuo, 1996). Vibration control also has important applications in industry and manufacturing, home appliances, and satellite platforms, to name only a few areas.

The Naval Postgraduate School began research in adaptive filters for use in the control of optical beam jitter in spacecraft applications. Edwards (1999), Watkins (2007), Yoon (2008), and Beerer (2008), investigated various adaptive control algorithms to attenuate jitter due to narrowband and broadband disturbances. Narrowband disturbances include mechanical vibrations on the optical platform of the spacecraft, and were simulated using mechanical shakers. Broadband disturbances such as the translational effects of atmospheric turbulence were simulated using a fast-steering mirror. Current research efforts in jitter control include improving algorithms for spacecraft platform applications as well as attenuation of jitter in high energy laser systems for maritime applications.

The basic principle of an adaptive filter working in an adaptive algorithm is that controller gains can be varied throughout the control process to adapt to changing parameters and can therefore cancel disturbances more effectively than passive methods. Various adaptive control algorithms have been developed for active noise control and described in detail by Widrow and Stearns (1985, 2002), Elliott and Nelson (1985), Haykin (2002), and Kuo (1996). The Least Mean Square (LMS) algorithm, Recursive Least Squares (RLS) algorithm, and the Filtered-x (FX) equivalents of each are presented in detail in these sources and have been applied to the NPS jitter control testbeds.

Application of adaptive filters to the field of adaptive optics and atmospheric turbulence compensation was proposed by Ellerbroek and Rhoadarmer (1998) and has been investigated primarily by Gibson, et al. (2000, 2007). Adaptive filters can be desirable in adaptive optics as opposed to or in addition to fixed-gain reconstructor algorithms due to the rapidly changing nature of atmospheric turbulence.

D. AIRBORNE LASER TEST BED

Before presenting the ongoing research on adaptive optics control for maritime applications, it will be useful to describe a current application of adaptive optics for horizontal or near horizontal paths for higher altitude applications, namely the Airborne Laser Test Bed (ALTB). The ALTB is under development by the Missile Defense Agency as part of a ballistic missile defense system that will use a High Energy Laser (HEL) to destroy hostile missiles at long range while using adaptive optics to compensate for atmospheric turbulence (Lamberson et al., 2006). The system has been tested in the field and successfully completed a test acquisition-to-engagement demonstration in January of 2010. (MDA, 2010). A concept of operations for the three laser beams in the ALTB system is shown in Figure 4. After the missile has been detected in the infrared, the tracking or Track Illumination Laser (TILL) is used to lock onto the missile nose, actively track the missile, and determine accurate range to the target. Next, the Beacon Illuminator Laser (BILL) is sent to the desired point of destruction on the missile, where its reflected return travels through the same atmospheric path expected for the HEL. As such, the BILL serves as the reference laser for the wavefront sensor to determine the atmospheric turbulence that must be corrected by the AO system. Finally, the HEL is sent out with the proper distortion so that its travel through the atmosphere yields a corrected, focused beam on the target, destroying the missile.

In maritime applications, a similar scenario is a possibility, with the exception that the atmospheric turbulence profile will be different in horizontal and near horizontal cases very close to the water. Maritime scenarios involving ships are likely to occur below approximately 20 meters, with a range up to approximately five kilometers (Hammel, et al., 2004). As in ALTB operation, the use of a reference laser for imaging

and laser propagation in maritime scenarios is important for determining the current state of the atmosphere. To reduce this dependency on a reference beam, research is underway to develop methods of beaconless wavefront sensing, which would reduce several challenges such as the need for an extra laser, extra optics and alignment for the wavefront sensor, and slight differences in the atmospheric path traveled by the reference laser and HEL. However, the research presented here uses a classical Shack-Hartmann wavefront sensor with a reference.

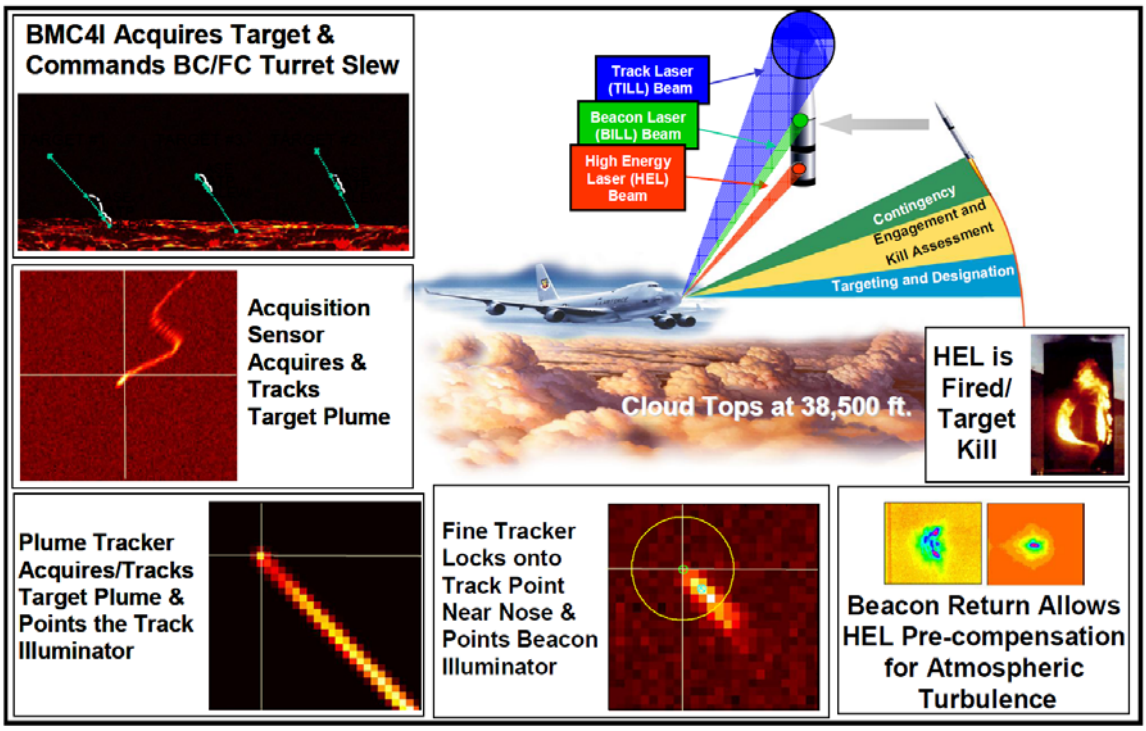

Figure 4. Concept of Operations for ALTB beams (From Lamberson et al., 2006)

E. ATMOSPHERIC TURBULENCE

This section provides some background and commonly used parameters to describe atmospheric turbulence. Turbulence arises from the heating and cooling of the Earth's surface, which cause changes in the index of refraction of the air. These changes alter the path of light propagating through the atmosphere. Atmospheric turbulence in a vertical path has been characterized and simulated successfully for astronomical applications, primarily using Kolmogorov turbulence theory and statistics.

Kolmogorov assumes that large scale turbulent motions are transferred to small scale turbulent motions which are statistically homogeneous and isotropic (Roggemann & Welsh, 1996). This theory mathematically describes the spatial frequency statistics of index of refraction fluctuations in the atmosphere. Kolmogorov theory applies in a range of turbulent eddy sizes called the inertial subrange, which is bounded by an outer scale, L_0, and an inner scale, l_0 (Andrews, 2004). Eddies larger than L_0 are not assumed to be homogeneous and isotropic, and eddies smaller than l_0 dissipate energy as heat rather than transferring it to other eddies. In the inertial subrange, $1/L_0 \ll \kappa \ll 1/l_0$, the spatial power spectral density (PSD) of the index of refraction of air is described as:

$$\Phi_n(\kappa, z) = 0.033 C_n^2(z) \kappa^{-11/3} \tag{1.5}$$

where the wavenumber, κ, is related to the isotropic scale or eddy size, l, by $l = 2\pi/\kappa$, the distance from the aperture is represented by z, and C_n^2 is the structure constant of the index of refraction fluctuations, or a measure of the turbulence strength. Weak turbulence is generally represented by C_n^2 values of about $10^{-17} m^{-2/3}$ or less, while C_n^2 values of about $10^{-13} m^{-2/3}$ or more generally indicate strong turbulence (Andrews, 2004). In vertical turbulence profiles, C_n^2 varies with height above the ground. Several models of C_n^2 have been developed using statistical data collected over the years and often based on specific geographical locations. Collecting and characterizing data on C_n^2 in a deep turbulence, maritime environment is a current effort underway in various organizations.

Another commonly used descriptor of atmospheric turbulence strength is Fried's parameter, or atmospheric coherence length, r_0. This parameter describes the seeing conditions at a particular site and limits a telescope's resolution such that an aperture of dimension r_0 produces a nearly diffraction-limited image; an aperture larger than r_0 will not provide better resolution (Andrews, 2004). Fried's parameter is expressed in centimeters as:

$$r_0 = \left[0.42 \sec(\varsigma) k^2 \int_0^L C_n^2(z) dz \right]^{-3/5} \tag{1.6}$$

where k is the wavenumber $k = 2\pi / \lambda$, λ is the wavelength, ς is the zenith angle, and L is the distance from the source to the telescope aperture along the z axis. Values of r_0 under 5 cm generally represent strong turbulence and poor seeing conditions, while r_0 values over 25 cm represent very good seeing conditions (Andrews, 2004; Wilcox, 2009).

The parameters C_n^2 and r_0 describe the spatial coherence or structure of the atmosphere, but the temporal nature of turbulence is also important. A commonly used method to simulate the temporal transition of atmospheric turbulence is the frozen seeing method or Taylor frozen flow approximation (Roggemann & Welsh, 1996). This approximation assumes that the index of refraction variations in the atmosphere remain constant over a very short time with the exception of a transverse velocity due to wind motion. To simulate this, a phase screen is generated and moved, or "drifted," across the optical aperture to simulate the wind motion. While this assumption simplifies turbulence generation in the laboratory, it neglects other temporal phase variations and can lead to repetitive or short turbulence realizations. The turbulence generated in this research uses a different method of temporal transition, but does make use of the Greenwood time constant, or the time interval over which the atmosphere can be assumed to be essentially the same. This time constant is expressed and approximated by:

$$\tau_0 = \frac{\cos^{3/5}(\varsigma)}{\left[2.91k2\int_{h_0}^{H} C_n^2(h)V^{5/3}(h)dh\right]^{3/5}} \approx \frac{0.314r_0}{V(h)}\left(\frac{D}{r_0}\right)^{1/6} \tag{1.7}$$

where $V(h)$ is the wind velocity as a function of altitude (Andrews, 2004).

To understand the effects of turbulence on a propagating wavefront, the wavefront phase is often expanded to be expressed as a linear combination of orthonormal basis functions. Zernike polynomials have been used extensively in adaptive optics research because they are orthonormal on a unit circle, and most optical components are circular in shape. Furthermore, low order Zernike modes correspond closely with standard low order optical aberrations such as focus, astigmatism, and coma. However, the Zernike modes contain some correlated coefficients, and a more efficient mode set is desirable. The Karhunen-Loève expansion consists of modes that are linear combinations of

Zernike polynomials and have statistically independent coefficients (Roggemann & Welsh, 1996). This mode set has been increasingly implemented in atmospheric simulations. Using the Karhunen-Loève (K-L) modes, a wavefront can be expressed in polar coordinates as:

$$\text{Wavefront}(\rho,\theta) = \sum_{i=1}^{M} a_i K_i(\rho,\theta) \tag{1.8}$$

where *M* is the number of K-L modes, and the a_i coefficients represent the weights given to each mode (Wilcox, 2009). The coefficients are calculated based on telescope and site parameters in addition to Zernike-Kolmogorov residual errors measured experimentally by Fried (1965) and calculated by Noll (1976). The a_i coefficients are used in both Zernike and K-L expansions of the wavefront. Given site parameters, the Zernike-Kolmogorov residual errors, and a chosen mode set, realizations of atmospheric turbulence can be simulated in the laboratory.

F. MARITIME ENVIRONMENT

While statistical data has been collected and analyzed for decades on the vertical turbulence profile, horizontal data collection has begun only relatively recently. No theoretical model currently exists to describe horizontal turbulence that parallels the familiar Kolmogorov statistical model used in vertical AO applications, and investigations are underway to develop such models. Experiments have been performed by SPAWAR in San Diego over a 7.07 km path at Zuniga Shoal to gather data on predicting the atmospheric structure constant, C_n^2, which is integral to the development of a theoretical model of maritime turbulence (Hammel et al., 2007). This work also used the Navy Surface Layer Optical Turbulence (NSLOT) model developed at NPS, which depends primarily on local air and sea temperature measurements. Additionally, the Naval Research Laboratory (NRL) and University of Puerto Rico at Mayagüez have collected horizontal propagation data over water at the Island of Magueyes in Puerto Rico (Santiago, et al, 2005), the University of Florida has taken measurements over maritime paths to study C_n^2 (Vetelino, et al., 2006), and Michigan Tech has begun horizontal path experiments over land and water to develop statistics of the atmospheric coherence length

or Fried parameter, r_0 (Sergeyev & Roggemann, 2010). In the research presented here, the thick aberrator problem is simulated in the lab by applying turbulence to two liquid crystal (LC) spatial light modulators (SLMs) and experimenting with two different optical path lengths between them.

There are many challenges associated with adaptive optics in a maritime environment. Scintillation, or intensity fluctuations due to random index of refraction changes, increases as propagation distance increases and is a primary effect which degrades beam quality in a horizontal, deep turbulence environment. Scintillation effects are studied in astronomical adaptive optics systems for propagation through low elevation angles, and work is underway to develop various systems that can augment classical AO controllers for this purpose (Vorontsov, et al., 2008). In addition to scintillation, branch points, or discontinuities in the optical phase, provide a challenge to AO systems as well (Fried, 1992; Fried, 1998; Sanchez & Oesch, 2009). Branch points are associated with $\pm 2\pi$ jumps or singularities in the phase and occur when the amplitude or intensity in the beam drops to zero. These singularities decrease the effectiveness of many classical wavefront sensors which provide phase and wavefront slope information to the corrector. While the wavefront sensor is designed to detect phase aberrations that can be reconstructed and corrected for, the introduction of phase discontinuities and amplitude variations resulting from scintillation and branch points can corrupt the pure phase measurements, leading to inaccurate wavefront information.

Humidity and temperature fluctuations, aerosols, and wave motion are other marine characteristics that affect turbulence. The research presented here focuses on adaptive optics beam control for thick aberrator or deep turbulence only. Future research will include the effects of jitter control, simulated ship motion, and additional maritime factors. The integration of current AO and HEL jitter control testbeds is shown in the schematic in Figure 5.

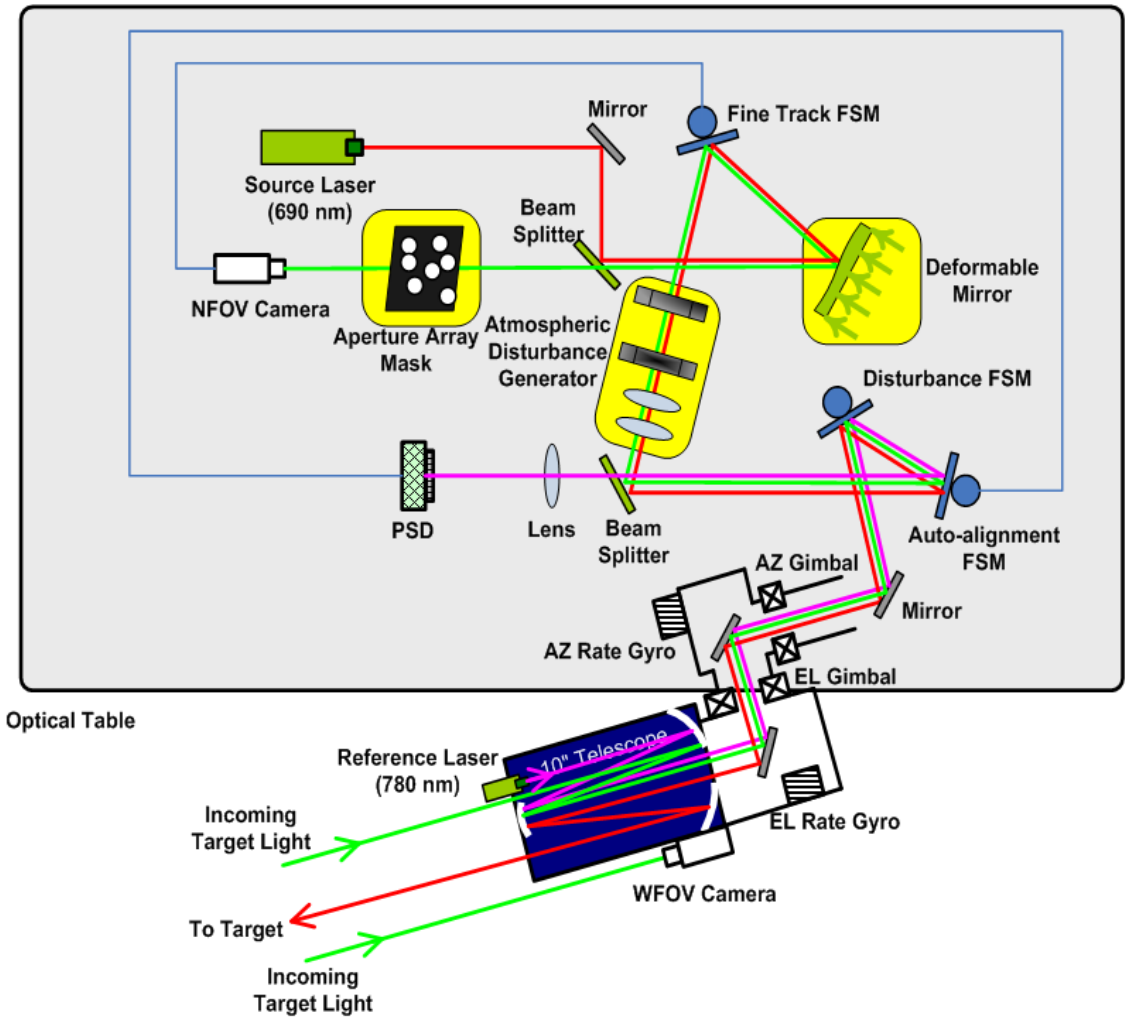

Figure 5. Schematic of HEL with adaptive optics (From HEL Testbed Poster, 2008)

G. PROBLEM FORMULATION

The goal of this dissertation is to lay a foundation for beam control research in a maritime environment. This will provide insight into the challenge of simplifying adaptive optics control for Navy systems in shipboard applications. To undertake this challenge, both analytical and experimental work will be accomplished. First, three advanced, multichannel control algorithms will be evaluated in augmenting a classical adaptive optics controller. They will be evaluated in the performance and speed of minimizing wavefront slope error across the aperture as well as for control effort expended and simplicity of implementation. The three algorithms tested will be a

transverse Least Mean Square adaptive filter, a lattice Recursive Least Squares adaptive filter, and a Kalman filter. They will augment a classical Proportional-Integral controller in an adaptive optics system. Second, a new adaptive optics testbed for the compensation of atmospheric turbulence will be designed and built. After being evaluated in simulation, the algorithms above will be tested in the laboratory system. Third, the laboratory testbed will be modified to generate a deep turbulence scenario that produces intensity fluctuations and dropouts in the laser beam profile. In this way, the deep turbulence characteristics of a maritime environment will be simulated in the laboratory. Initial beam control comparisons will then be performed in this environment to draw initial conclusions on how well the algorithms can minimize wavefront slope error in the challenging deep turbulence scenario.

This research is presented in the following way: Chapter II describes the advanced beam control algorithms investigated in this research. Chapter III presents the laboratory system developed, and Chapter IV presents simulation and testbed results comparing the various control methods. Chapter V describes the deep turbulence simulation and presents associated results. Finally, Chapter VI provides conclusions and recommendations for further study.

II. ADVANCED BEAM CONTROL FOR ADAPTIVE OPTICS

This chapter presents the theoretical development of the adaptive and Kalman filters used in this research. The Least Mean Square (LMS) and Filtered-x Least Mean Square (FXLMS) algorithms will be described first, followed by an explanation of how they are used to augment the classical Proportional-Integral (PI) controller. The extension of these algorithms to their multichannel equivalents is presented next, followed by a description of the Recursive Least Squares (RLS) algorithm and its multichannel equivalent, as well as the implementation of the algorithms in Matlab's Simulink environment. Finally, the Kalman filter and its Simulink implementation are described.

A. LMS/FXLMS ADAPTIVE FILTER

Adaptive filters can be infinite impulse response (IIR) or finite impulse response (FIR). IIR filters contain feedback paths in their structure and respond indefinitely, though this leads to potential instability (Haykin, 2002). On the other hand, an FIR filter contains only feedforward paths and its response dies off after a finite duration, making the filter inherently stable. FIR filters are more popular in real applications, and the filter used in this research is FIR. Two commonly used implementations of an FIR filter include transverse and lattice structures. Though the general formulation of the adaptive algorithms to be presented can be found in several sources, the developments here primarily follow those of Kuo (1996).

An L^{th} order transverse FIR filter has the structure shown in Figure 6. Each of the L stages, or taps, delays the input signal by one sample, which leads many to call this filter a tapped-delay line. The filter output is expressed as follows:

$$y(n) = \sum_{i=0}^{L} w_i(n)x(n-i) = \boldsymbol{w}^T(n)\boldsymbol{x}(n) \tag{2.1}$$

where $\boldsymbol{w}(n)$ is the filter weight vector of length L whose i^{th} component is $w_l(n)$, $\boldsymbol{x}(n)$ is the vector of delayed inputs $x(n-i)$, and $y(n)$ is the filter output.

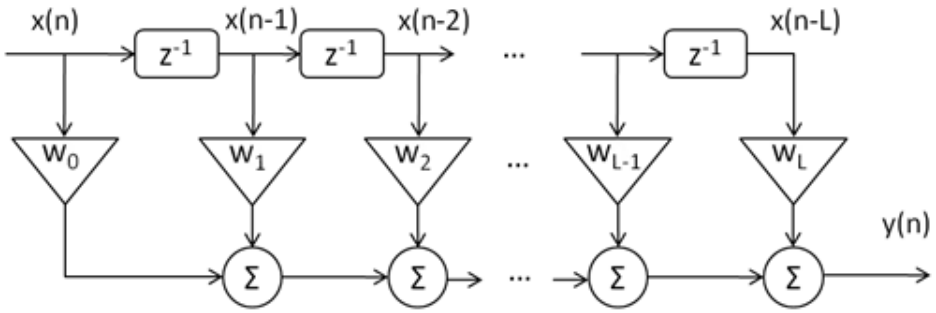

Figure 6. Transverse FIR filter structure

An N^{th} order lattice FIR filter has the general structure shown in Figure 7. Each stage of the lattice filter is used to calculate an orthogonal forward and backward prediction error. Given a stationary process with correlated samples, the backward prediction errors consist of uncorrelated random variables, indicating that the lattice structure has performed the maximum prediction it can. These prediction errors can be used to estimate the disturbance or desired response (Haykin, 2002). The lattice filter stages are modular and independent, so additional stages can be added if necessary without recalculating earlier coefficients. While the lattice structure is complex, it provides efficient implementation of the RLS algorithm. Jiang and Gibson developed such an RLS lattice filter with channel orthogonalization for use in large multichannel systems (1995). Over the past decade they have used this filter to augment classical control methods in simulating efficient compensation of phase distortions due to atmospheric turbulence in adaptive optics systems for airborne laser applications (Gibson, et al., 2000; Liu & Gibson, 2007).

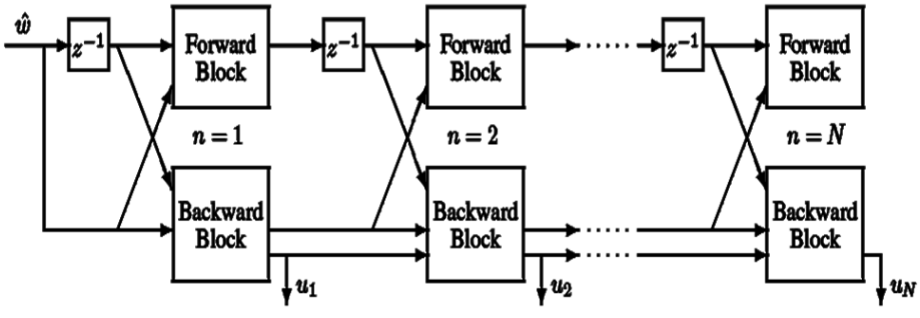

Figure 7. Lattice filter structure (From Yoon, 2008)

Figure 8 shows the simplest implementation of a feedforward adaptive algorithm. The function of the adaptive filter is to modify an incoming or reference signal, $x(n)$, to cancel a disturbance applied to the system or to produce a desired signal, $d(n)$. In adaptive optics, $d(n)$ is a disturbance to be canceled. Using a transverse filter and a reference that is correlated with the disturbance, the filter delays the incoming signal $L-1$ times and multiplies the resulting vector by a set of L weights, as shown in Figure 6. The error, $e(n)$, is measured at an error sensor and is the difference between the applied disturbance and the filter output, $y(n)$, which is the canceling signal from the adaptive filter.

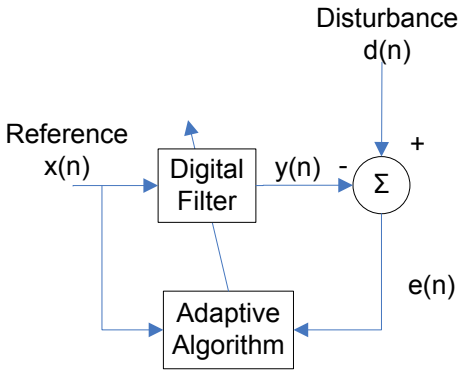

Figure 8. Simple implementation of adaptive algorithm

The adaptive weights are computed by an algorithm that uses the reference and error signals to minimize a cost function. In the LMS algorithm, the cost function is the mean square error (MSE), which is the expectation of $e(n)^2$ and is denoted by $\xi(n)$. When the statistics of the disturbance and the reference signal are available, the weights that minimize the MSE can be computed. In practice, however, such a priori information is often unavailable. As a result, the MSE is approximated by the instantaneous squared error and minimized using iterative steepest-gradient descent to update the weights in the direction of lowest error. The resulting form of the LMS algorithm is expressed as:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \alpha \boldsymbol{x}(n)e(n) \tag{2.2}$$

where α is the convergence coefficient that controls the speed at which the algorithm converges to steady-state weight values.

In reality, the basic adaptive algorithm must be modified because the control signal passes through a physical actuator before its effect is sensed at the error sensor. A secondary path or plant transfer function, $S(z)$, contains information on the interaction between sensor and actuator and is denoted by the S block in the diagram in Figure 9. Its effect on the control action must be taken into account to prevent instability and ensure that the filter, W, cancels the disturbance *after* the secondary plant, not before.

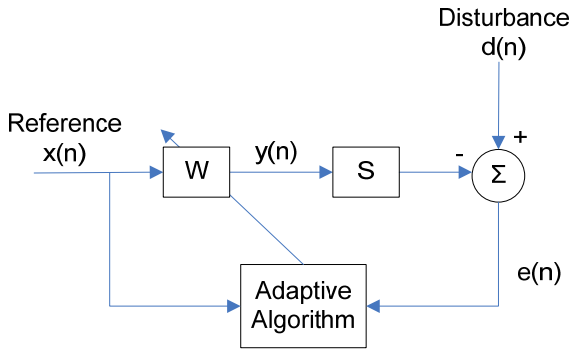

Figure 9. Adaptive algorithm showing secondary plant after filter output

To account for the secondary plant dynamics, the reference signal is passed through a copy or estimate of the secondary plant, $\hat{S}(z)$, before being used in the adaptive algorithm. The LMS algorithm is updated as:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \alpha \boldsymbol{r}(n) e(n) \tag{2.3}$$

where $\boldsymbol{r}(n)$ represents the filtered reference, $\boldsymbol{r}(n) = \hat{s}(n) * \boldsymbol{x}(n)$. As such, this method is called Filtered-x LMS, or FXLMS. The complete FXLMS model is shown in Figure 10.

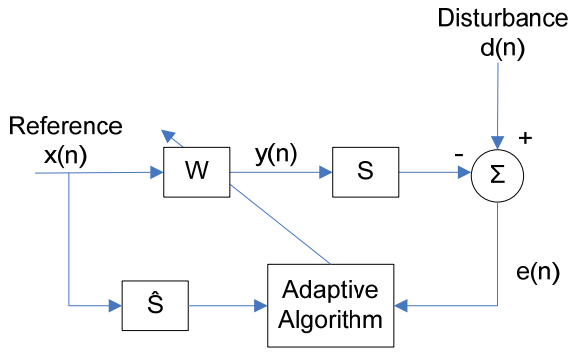

Figure 10. FXLMS structure including secondary plant estimate prior to adaptation

Given exact knowledge of the secondary plant, the weights calculated by the adaptive algorithm should converge to optimal values. In practice, this plant can include system interactions that are difficult to model precisely, though the adaptive algorithm can compensate for these uncertainties if the modeling error is not too large. In the adaptive optics problem, the secondary plant is represented by the dynamics of any control loop in the system, the influence function or poke matrix of the deformable mirror, and the dynamics of the mirror itself. Mirror dynamics must be included for large, flexible mirrors such as the James Webb Space Telescope and those for other lightweight spacecraft applications. However, the small mirror for laboratory and terrestrial applications that is used in this research is assumed to be rigid or static, and the secondary plant reduces to the dynamics of the control loop and mirror's poke matrix.

The practical challenge with the adaptive filters described so far is that they require a reference signal that is correlated with the disturbance in order to provide feedforward correction and canceling. This is possible in acoustic applications where a reference sensor can be placed upstream of the corrective secondary sources to sample the disturbance and still have time to correct downstream (Kuo, 1996). In vibration control for applications involving rotating machinery or other periodic disturbances, the adaptive filter again performs well as a feedforward controller. However, in the presence of atmospheric turbulence in optical systems, an external reference correlated to the disturbance is not available. The wavefront sensor provides the error signal in terms of the slope of the wavefront phase, which contains the effects of both the secondary plant (DM controller) and the distortion due to turbulence. These cannot be separated in the real system. Furthermore, while atmospheric turbulence contains some structure, it is not precisely periodic in nature.

In lieu of an external reference, it is possible to generate an internal reference that is an estimate of the disturbance (Kuo, 1996). This reference is produced by passing the adaptive filter output through another estimate of the secondary plant and adding this output to the error signal. This effectively removes the adaptive filter output from the

error, leaving only an estimate of the disturbance. If the secondary plant model is precise, the reference signal becomes the disturbance itself. The reference signal is expressed as:

$$x(n) = e(n) + \hat{s}(n) * y(n) = \hat{d}(n) \tag{2.4}$$

with "*" denoting the convolution operation. Figure 11 shows the modified controller diagram in which the reference signal is generated.

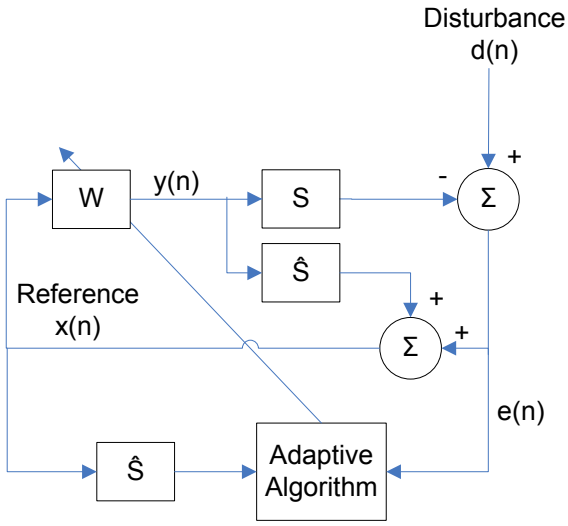

Figure 11. Adaptive controller diagram with internally generated reference

The need to estimate the disturbance from the error adds a delay in the system and turns a previously feedforward controller into a feedback controller. As such, the ability of a true feedforward controller to address broadband disturbances becomes limited. It is expected that when the adaptive controller augments a classical controller, the performance of both will be improved.

B. ADAPTIVE FILTER WITH CLASSICAL CONTROL LOOP

Classical integral control for adaptive optics was described in a general form in Chapter I, Section C. The classical controller implemented in this research is a Proportional-Integral (PI) controller, denoted by $C(z)$ in discrete form and expressed as:

$$C(z) = \frac{K_i z}{z-1} + K_p \tag{2.5}$$

where the integral gain, K_i, and the proportional gain, K_p, are designed to meet reasonable specifications of the PI control loop, and the discrete sample time, Ts, is included in the integral gain. In simulation and testbed results, the controller parameters used will be provided. The adaptive filter is expected to improve the overall controller performance for disturbances which are outside the bandwidth of the PI controller.

The combination of the adaptive controller with internally generated reference and the PI controller is shown in Figure 12. The system delay, z^{-q}, is included and can represent a delay of any number of time steps, q. In simulations and laboratory experiments, q is assumed to be one. All of the components considered to be part of the adaptive controller are grouped inside the dashed box on the left and will hereafter be denoted in diagrams by A. The secondary path is from the point the adaptive filter output is applied to the point before the disturbance is added to the system. When the adaptive filter output is injected before the PI controller, the secondary path model at steady state becomes the transfer function of the closed PI loop, given by:

$$S(z) = \frac{C(z)}{1 + z^{-q}C(z)} \tag{2.6}$$

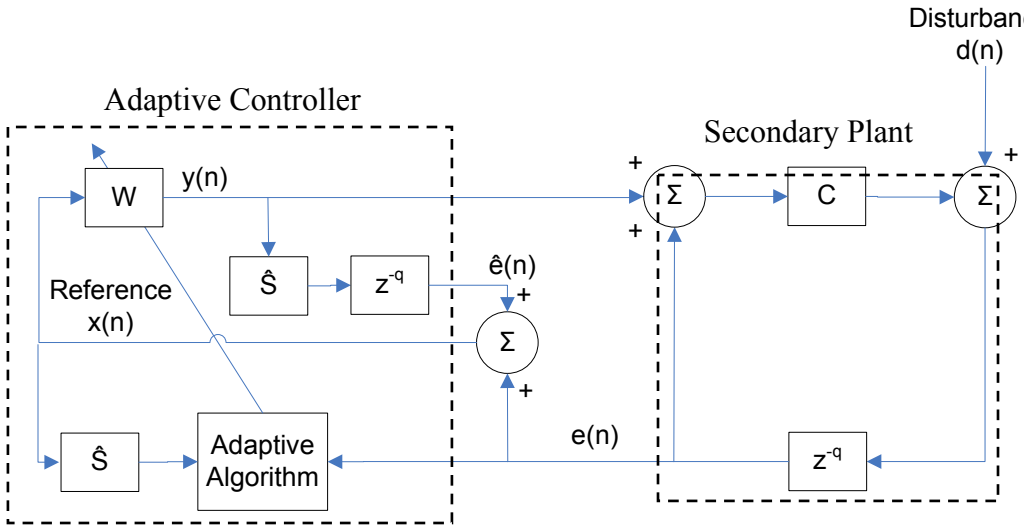

Figure 12. Final configuration of adaptive and PI controllers together

C. MULTICHANNEL LMS/FXLMS ADAPTIVE FILTER

Since an adaptive optics system is a Multiple Input Multiple Output (MIMO) system, the LMS and FXLMS adaptive algorithms must be extended for use with multiple channels. Multiple error LMS was proposed by Elliott, et al. (1985, 1987), for multichannel active noise control applications. The summation notation here draws from Elliott, while for consistency most of the variables again follow Kuo's summary. In a multichannel adaptive filter algorithm, the number of reference signals can be independent of the number of sensors or actuators. For M error sensors, K control actuators, and J reference signals, there are MxK secondary path models and KxJ weight vectors. Each of the weight vectors is length L, so that the actual size of the weight matrix or filter W is $KxJL$. Each secondary path model, s_{mk} represents the relationship between the control action by the k^{th} actuator and the error observed by the m^{th} sensor. For the rigid deformable mirror whose dynamics are ignored, the mirror's contribution to the secondary path model consists of the elements of the poke matrix, each of which represents the interaction between a particular mirror actuator and Shack-Hartmann wavefront sensor lenslet. The command for the k^{th} actuator is generated by multiplying each of the J reference signals by its corresponding weight vector and is expressed as:

$$y_k(n) = \sum_{j=1}^{J} \boldsymbol{w}_{kj}^{T}(n)\boldsymbol{x}_j(n) \tag{2.7}$$

where $\boldsymbol{w}_{kj}(n) = \begin{bmatrix} w_{kj0}(n) & w_{kj1}(n) & \cdots & w_{kj(L-1)}(n) \end{bmatrix}^T$, the complete command vector is given by $\boldsymbol{y}(n) = \begin{bmatrix} y_1(n) & y_2(n) & \cdots & y_K(n) \end{bmatrix}^T$, and the complete reference vector is given by $\boldsymbol{x}(n) = \begin{bmatrix} \boldsymbol{x}_1^T(n) & \boldsymbol{x}_2^T(n) & \cdots & \boldsymbol{x}_J^T(n) \end{bmatrix}^T$. The weight update equation is simply the multichannel expression of the FXLMS algorithm shown before and is given by:

$$\boldsymbol{w}_{kj}(n+1) = \boldsymbol{w}_{kj}(n) + \alpha \sum_{m=1}^{M} \boldsymbol{r}_{kjm}(n) e_m(n) \tag{2.8}$$

where $\boldsymbol{e}(n) = \begin{bmatrix} e_1(n) & e_2(n) & \cdots & e_M(n) \end{bmatrix}^T$, and the filtered reference is $\boldsymbol{r}_{kjm}(n) = s_{mk}(n) * \boldsymbol{x}_j(n)$.

D. RLS ADAPTIVE FILTER

The Recursive Least Squares algorithm follows much of the development shown for LMS, with the important exception that it includes past data in its cost function. This accommodates nonstationary signals and usually provides faster convergence and smaller steady-state error than the LMS algorithm, though it is more computationally complex (Kuo, 1996). Instead of expressing the MSE as the instantaneous squared error signal only, the cost function becomes:

$$\xi(n) = \sum_{i=1}^{n} \lambda^{n-i} e^2(i) \tag{2.9}$$

where the forgetting factor, $0 < \lambda \leq 1$, allows more recent data to be weighted more heavily and data long past to be forgotten. A value of $\lambda = 1$ implies that nothing is forgotten, while smaller values allow more forgetting. As it is desirable to use as much information as possible, values of λ used in the NPS jitter control testbeds range from 0.9–0.99999. In this research, some experimentation with λ will take place in the deep turbulence scenario.

While the error and control signal expressions in RLS are identical to those of LMS, the weight update process is different. Optimal weights could be calculated from the statistics of the reference and disturbance signals if they are available, but such computation is extremely difficult for large sample times. Instead of calculating and inverting the correlation matrix of the reference input, $\boldsymbol{R}(n)$, the inverse correlation matrix, $\boldsymbol{Q}(n) = \boldsymbol{R}^{-1}(n)$ is calculated recursively. This eliminates the need to compute or invert $\boldsymbol{R}(n)$, greatly reducing the complexity of the RLS algorithm. The recursive equations for weight updates using the filtered reference for FXRLS formulation are:

$$\boldsymbol{z}(n) = \lambda^{-1}\boldsymbol{Q}(n-1)\boldsymbol{r}(n) \tag{2.10}$$

$$\boldsymbol{k}(n) = \frac{\boldsymbol{z}(n)}{\boldsymbol{r}^T(n)\boldsymbol{z}(n)+1} \tag{2.11}$$

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \boldsymbol{k}(n)e(n) \tag{2.12}$$

where $\boldsymbol{z}(n)$ is an intermediate calculation and $\boldsymbol{k}(n)$ is the current gain vector. Finally, the inverse sample correlation matrix is updated as :

$$\boldsymbol{Q}(n) = \lambda^{-1}\boldsymbol{Q}(n-1) - \boldsymbol{k}(n)\boldsymbol{z}^T(n) \tag{2.13}$$

In the multichannel case, each length L reference signal, $\boldsymbol{r}_{jkm}(n)$, will produce a corresponding $\boldsymbol{z}_{jkm}(n)$ and $\boldsymbol{k}_{jkm}(n)$. The control input to the k^{th} actuator will still be expressed as in Equation 3.7, but the multichannel weight update will be expressed as:

$$\boldsymbol{w}_{kj}(n+1) = \boldsymbol{w}_{kj}(n) + \sum_{m=1}^{M} \boldsymbol{k}_{kjm}(n) e_m(n) \tag{2.14}$$

The RLS algorithm used for comparison with LMS in this research is implemented using the lattice filter structure developed by Jiang and Gibson (1995).

E. MULTICHANNEL FORMULATION FOR SIMULINK

The previous sections have presented the theoretical development of commonly used adaptive filter algorithms. To implement the transverse adaptive filter algorithms in Matlab's Simulink environment for use with the adaptive optics testbed in this research, some modifications are made. Two classes of models are tested. The first uses the full error dimension, M, number of adaptive filters, and the second uses the actuator dimension, , K number of adaptive filters. For the DM and WFS used, $K < M$.

For a general multichannel adaptive filtering problem, the number of error and reference signals can vary depending on the application. For adaptive optics, the upper limit on the number of control channels and error sensors is determined by the number of DM actuators and WFS lenslets in hardware. Fewer control channels can be used if actuators are grouped, or slaved, or if some other form of model reduction is desired. Given a number of control channels, the placement of the reconstructor matrix, which is the pseudoinverse of the poke matrix, determines the number of error and reference channels used by the adaptive algorithm. The reconstructor can be placed in the path of the classical control loop only, or in the paths of both the adaptive and classical control loops.

If the reconstructor is used in the classical control loop only, the full M number of error measurements is used in the adaptive filter, and M reference signals are generated as disturbance estimates. Figure 13 shows this configuration, with the poke

matrix and pseudoinverse denoted by Γ and $\Gamma^{\dagger}$, respectively. This configuration is referred to as the sensor space model since it uses the full sensor dimension in the adaptive filter.

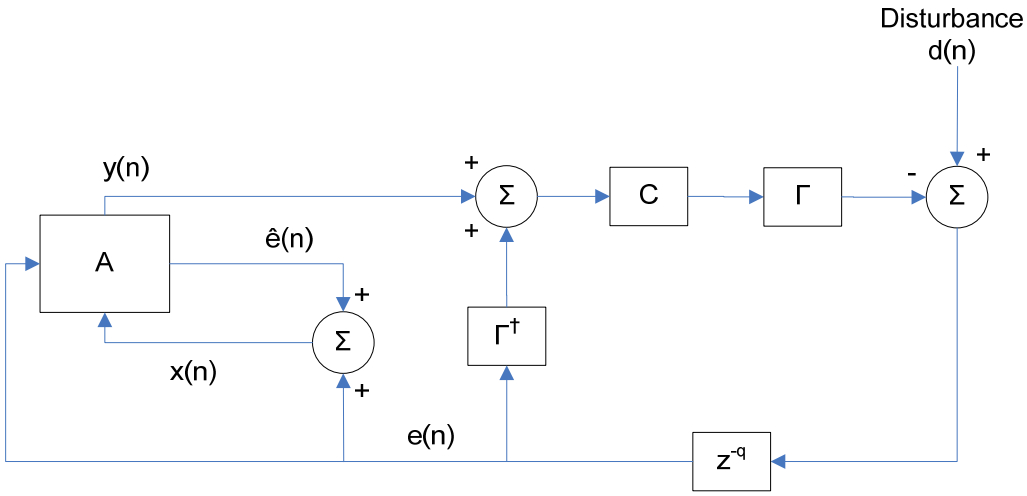

Figure 13. Sensor space model; reconstructor in classical loop only

In the sensor space model, the poke matrix is included in the secondary plant. Discrete transfer functions are represented by capital letters and the variable z, while impulse responses are indicated by lower case letters with the variable n. The sensor space secondary plant is given by:

$$S_{mk}(z) = \frac{C(z)}{1 + z^{-q}C(z)}\Gamma_{mk} \tag{2.15}$$

and the filtered reference is given by:

$$\boldsymbol{r}_{jkm}(n) = S_{mk}(z)\boldsymbol{x}_j(n) = \frac{C(z)}{1 + z^{-q}C(z)}\Gamma_{mk}\boldsymbol{x}_j(n) \tag{2.16}$$

Since the poke matrix elements are constant, the filtered reference can be modified as:

$$\boldsymbol{r}'_j(n) = \frac{C(z)}{1 + z^{-q}C(z)}\boldsymbol{x}_j(n) \tag{2.17}$$

so that $\boldsymbol{r}_{kjm}(n) = \Gamma_{mk}\boldsymbol{r}'_j(n)$. It is then possible to modify the summation term on the right hand side of Equation 3.8 to be:

$$\sum_{m=1}^{M}\boldsymbol{r}_{kjm}(n)e_m(n) = \sum_{m=1}^{M}\Gamma_{mk}\boldsymbol{r}'_j(n)e_m(n) = \boldsymbol{r}'_j(n)\sum_{m=1}^{M}\Gamma_{km}^{T}e_m(n) \tag{2.18}$$

where the modified filtered reference takes the place of $\boldsymbol{r}_{kjm}$, the j^{th} reference is then brought outside of the summation, and the poke matrix is transposed to preserve the correct dimensionality. This creates a converted error vector of length K that can be expressed as $\boldsymbol{e}'(n) = \Gamma^T \boldsymbol{e}(n)$, which allows a final weight update equation to be formulated as:

$$\boldsymbol{w}_{kj}(n+1) = \boldsymbol{w}_{kj}(n) + \alpha \boldsymbol{r}'_j(n) e'_k(n) \tag{2.19}$$

This final form of the weight updates lends itself to more efficient and straightforward implementation in Simulink's block diagram environment, where the matrix of $KxJL$ update terms is formed by $\alpha \boldsymbol{e}(n) \boldsymbol{r}(n)^T$.

The alternative configuration is when the reconstructor is placed in the error path before being sent into the adaptive filter, so that it is present in both control loops. This means that prior to being seen by the adaptive filter, the error is projected into the actuator dimension, K, and K reference signals are produced. In the present case where there are fewer actuators than sensors, this reduces the computation and convergence time for the adaptive filter, while the pseudoinverse projection preserves the useable error information for weight adaptation. This configuration is shown in Figure 14 and is referred to as the actuator space model since the adaptive filter takes the actuator dimension. While both sensor and actuator space models should have comparable performance, the actuator space model is favored for its faster convergence.

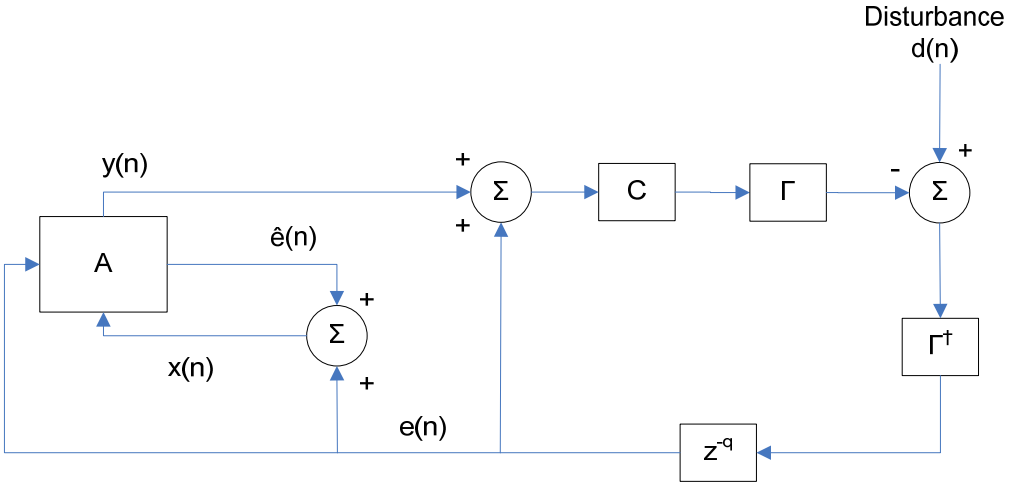

Figure 14. Actuator space model; reconstructor in both loops

In the actuator space model, the secondary plant includes the reconstructor matrix, which allows it to reduce to the transfer function of the classical loop only by:

$$S_{mk}(z) = \frac{C(z)}{1+z^{-q}C(z)}\Gamma^{\dagger}{}_{km}\Gamma_{mk} = S_{mk}(z) = \frac{C(z)}{1+z^{-q}C(z)} \tag{2.20}$$

The poke matrix is then eliminated from the formulations developed for the sensor space model, and the error vector used in the adaptive filter is already of dimension K and can be used directly in the weight update equation as:

$$\boldsymbol{w}_{kj}(n+1) = \boldsymbol{w}_{kj}(n) + \alpha \boldsymbol{r}'_{j}(n)e_{k}(n) \tag{2.21}$$

As in the sensor space model, this can be implemented in Simulink with the update terms computed by $\alpha \boldsymbol{e}(n)\boldsymbol{r}(n)^{T}$. However, because $J = K$ reference signals are generated in actuator space, the computation inside the adaptive algorithm can be reduced by decoupling the channels and using only $\boldsymbol{r}_{k}(n)$ and $e_{k}(n)$ to generate the k^{th} actuator command. In this case, $\boldsymbol{w}_{kj}(n) = 0$ for $j \neq k$, and the size of the decoupled weight update matrix is KxL as opposed to $KxJL$, greatly reducing the number of computations needed at each step of the process. This assumes that the actuator action of one channel does not affect the error in the other channels and the reference signal or disturbance estimate for a particular channel can be obtained from the error signal of that channel alone. For a continuous surface deformable mirror, the actuator action is not necessarily decoupled. While it is possible to determine a combination of mirror modes for which decoupling can be assumed, the goal of this research is to simplify the control implementation as much as possible, and no additional modal analysis is done. Both the coupled and decoupled algorithms are tested in simulation. For this system, the coupled model converges faster at the same convergence rate, making it the preferred option for control.

F. KALMAN FILTER

As an alternative to the adaptive filter, a Kalman filter is introduced to augment the classical PI controller to compare its performance. A Kalman filter is a classical observer that produces an estimate or prediction based on a state space system model in the presence of process noise and measurement noise (Haykin, 2002). As formulated in this research, it is used to estimate the disturbance, like the adaptive filter, so that it

cancels the disturbance to minimize the wavefront slope error. The state space process and measurement models are expressed as:

$$\boldsymbol{x}_{k+1} = \Phi_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{w}_k \tag{2.22}$$

$$\boldsymbol{y}_k = H_k \boldsymbol{x}_k + \boldsymbol{v}_k \tag{2.23}$$

where $\boldsymbol{x}$ is the process, Φ is the state transition matrix, B is the input or influence matrix, $\boldsymbol{w}$ is the process noise, $\boldsymbol{y}$ is the measurement, H is the measurement matrix, and $\boldsymbol{v}$ is the measurement noise. The standard predictor Kalman filter equations for projecting the current state ahead are expressed as:

$$\hat{\boldsymbol{x}}_{k+1}^{-} = \Phi_k \hat{\boldsymbol{x}}_k^{+} + B_k \boldsymbol{u}_k \tag{2.24}$$

$$P_{k+1}^{-} = \Phi_k P_k^{+} \Phi_k^T + Q_k \tag{2.25}$$

where P is the estimated error covariance matrix and Q is the process noise covariance matrix. The predicted quantities $\hat{\boldsymbol{x}}_{k+1}^{-}$ and P_{k+1}^{-} for the timestep k+1 become the current quantities used at timestep k, or $\hat{\boldsymbol{x}}_k^{-}$ and P_k^{-}, respectively. They are used in the corrector equations for updating the prediction with a new measurement, which are expressed as:

$$K_k = P_k^{-} H_k^T \left(H_k P_k^{-} H_k^T + R_k \right)^{-1} \tag{2.26}$$

$$\hat{\boldsymbol{x}}_k^{+} = \hat{\boldsymbol{x}}_k^{-} + K_k \left(\boldsymbol{y}_k - H_k \hat{\boldsymbol{x}}_k^{-} \right) \tag{2.27}$$

$$P_k^{+} = \left(I - K_k H_k \right) P_k^{-} \tag{2.28}$$

where K is the Kalman gain and R is the measurement noise covariance matrix. The process and error covariance matrix estimates are then used again in Equations 2.24 and 2.25, and the recursive calculations continue.

Figure 15 shows the Kalman filter model implemented in Simulink. The Kalman filter formulation replaces the *A* block representing the adaptive filter in Figure 14. The filter outputs the estimated measurement of the disturbance, which is in turn filtered by the inverse of *G*, the closed loop transfer function of the classical control loop. It is then injected into that control loop to cancel the measured disturbance and minimize the wavefront error. In order for the estimated measurement to be corrected by the actual measurement, it is delayed and passed through the negative of the closed loop transfer function before the comparison.

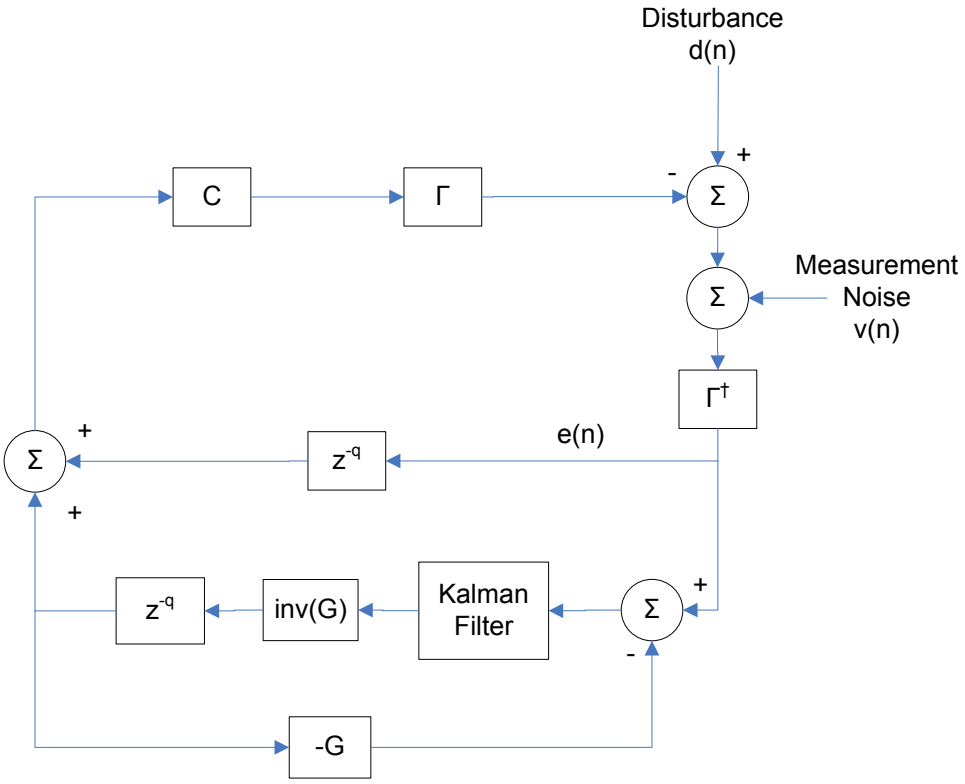

Figure 15. Kalman filter augmenting classical PI loop

Figure 16 shows an alternate implementation to save computation time. Since the Kalman filter output passes through both the inverse of *G* and *G* itself before being compared to the actual measurement, an alternate model can be used where the output passes through the delay only, and the sign in the summation block is changed to account for the negative of *G* from before. As a result, the estimated measurement is compared to the actual measurement as shown in the standard Kalman filter equations.

In simulation, a disturbance process is easily created from a white noise input and its model used as the Kalman filter's state space model. With a disturbance generated from the laboratory testbed, system identification of the disturbance must be performed in order to determine a model. This is done using Matlab's system identification toolbox. System identification is performed on each disturbance channel using the *iddata* command. Each channel is then converted to a 4^{th} order autoregressive model using the *ar* command. Finally, the transfer functions of each channel are compiled into a diagonal

matrix and converted to a state space model using the *ss* command. This generates the state transition matrix and measurement matrix that are used by the Kalman filter.

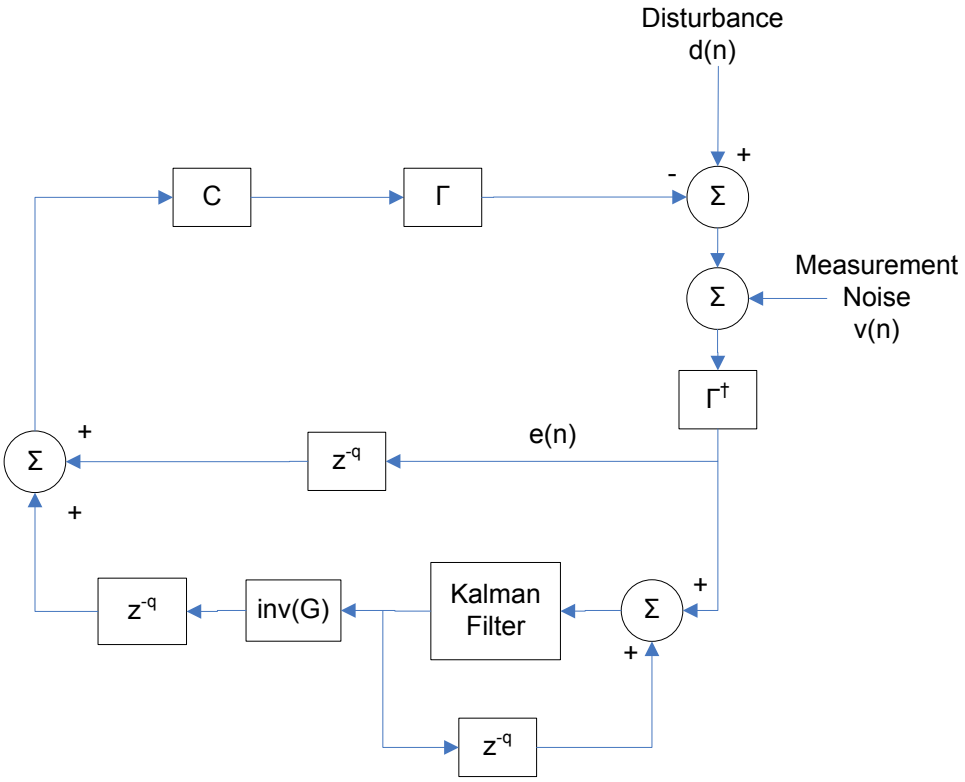

Figure 16. Simpler Kalman filter model bypassing *G* blocks in feedback loop

With the formulations described in this chapter, the classical PI, transverse LMS adaptive filter, RLS lattice filter, and Kalman filter have been implemented in Simulink. Chapter IV presents the results of applying the various control algorithms in simulation and on the AO testbed.

III. LABORATORY TESTBED

This chapter describes the laboratory testbed developed for this research. An overview of the adaptive optics system is presented first, followed by more detailed information regarding the deformable mirror, wavefront sensor, spatial light modulators, and other optical components used in the system.

A. SYSTEM OVERVIEW

Figure 17 shows the adaptive optics testbed with primary components labeled, and Figure 18 shows a schematic of the AO system with four available beam paths. Initial control algorithm testing was performed using this initial configuration, which will be referred to as the short path. The deep turbulence scenario was created by extending the beam path in a configuration referred to as the long path.

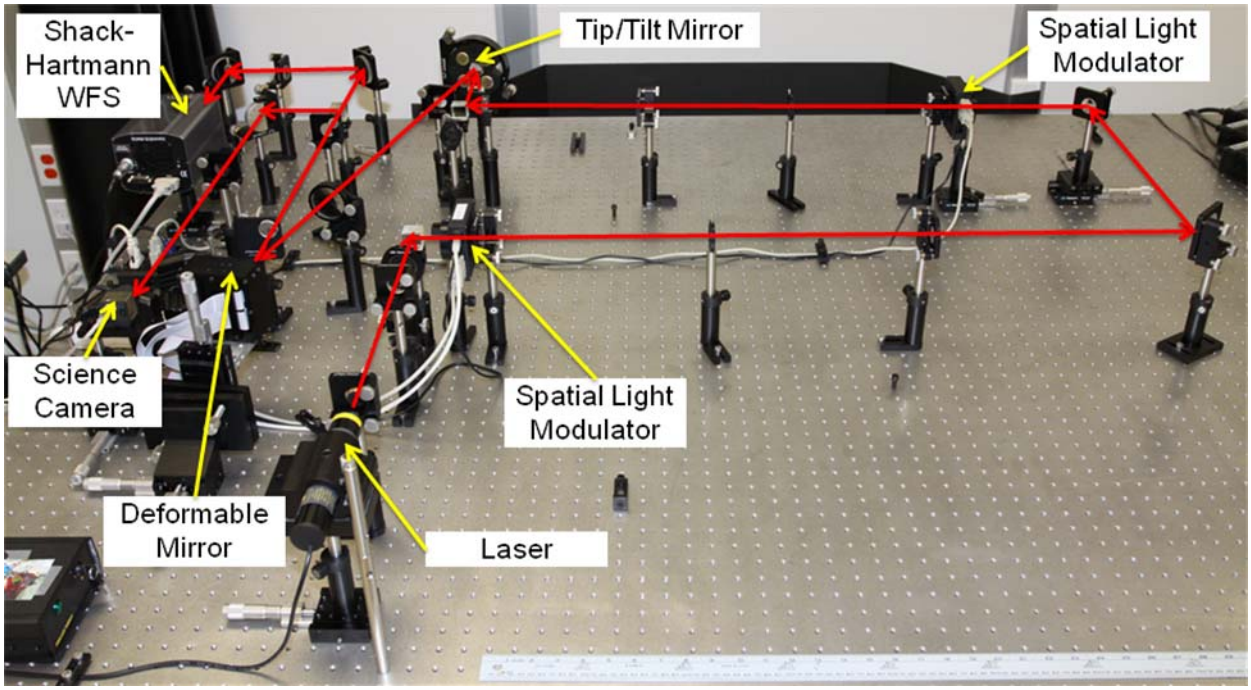

Figure 17. Laboratory testbed showing primary components

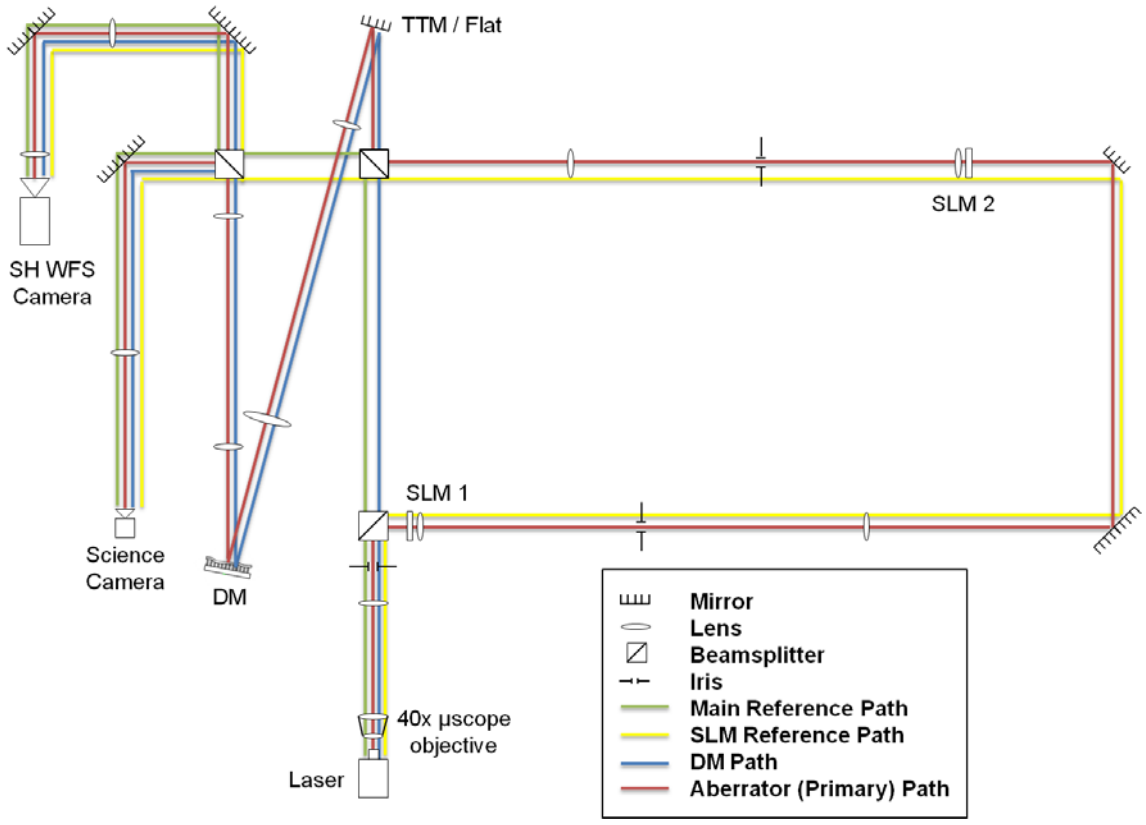

Figure 18. Schematic of laboratory system in short path

The main reference and DM paths are used for basic system alignment so that all other paths can be aligned to them. The main reference path passes from the laser through collimating optics and beamsplitters to arrive at the science camera and wavefront sensor camera. The DM path follows the reference path with the exception that instead of going straight to the cameras, it travels to the tip/tilt mirror (TTM), which currently serves as a flat mirror only, as well as the DM. This path is aligned to the reference path by using interferometric techniques. To apply atmosphere to the system, the beam must pass through an aberrator provided by two liquid crystal (LC) spatial light modulators (SLMs). The SLM reference path passes through the SLMs but, like the main reference, bypasses the DM. With zero aberrations applied to the SLMs, this beam forms the reference image used to build the poke matrix and drive the AO system. The final beam path is called the aberrator or primary path, as it is the one used for data collection

and control through applied atmosphere. The primary path passes through the SLMs and accompanying optics and is relayed to the DM and cameras.

While Chapter V will provide more detail on the deep turbulence scenario created in the lab during this research, an overview of the system modification is provided here. Using additional mirrors and reflecting the beam to and from an optical table across the room, the aberrator path was extended by approximately 22 m. The setup can be changed quickly and easily to support either the short or long path by using a translation stage to move only two mirrors. These mirrors break and return the beam to its original path, and can be moved in or out of the original path as desired. Figure 19 shows the updated configuration, and Figure 20 shows a schematic with the long path extension in blue. The primary components used in the testbed are described next.

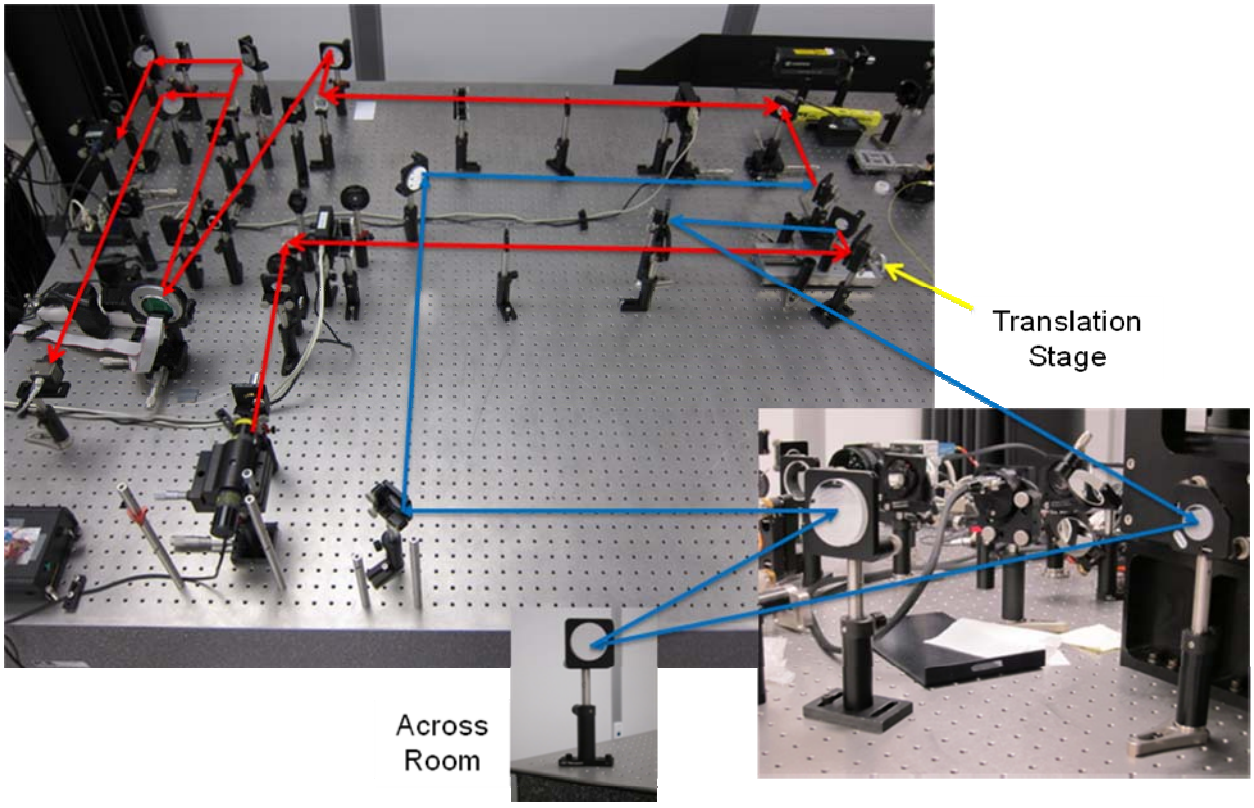

Figure 19. Modified testbed showing long path extension in blue

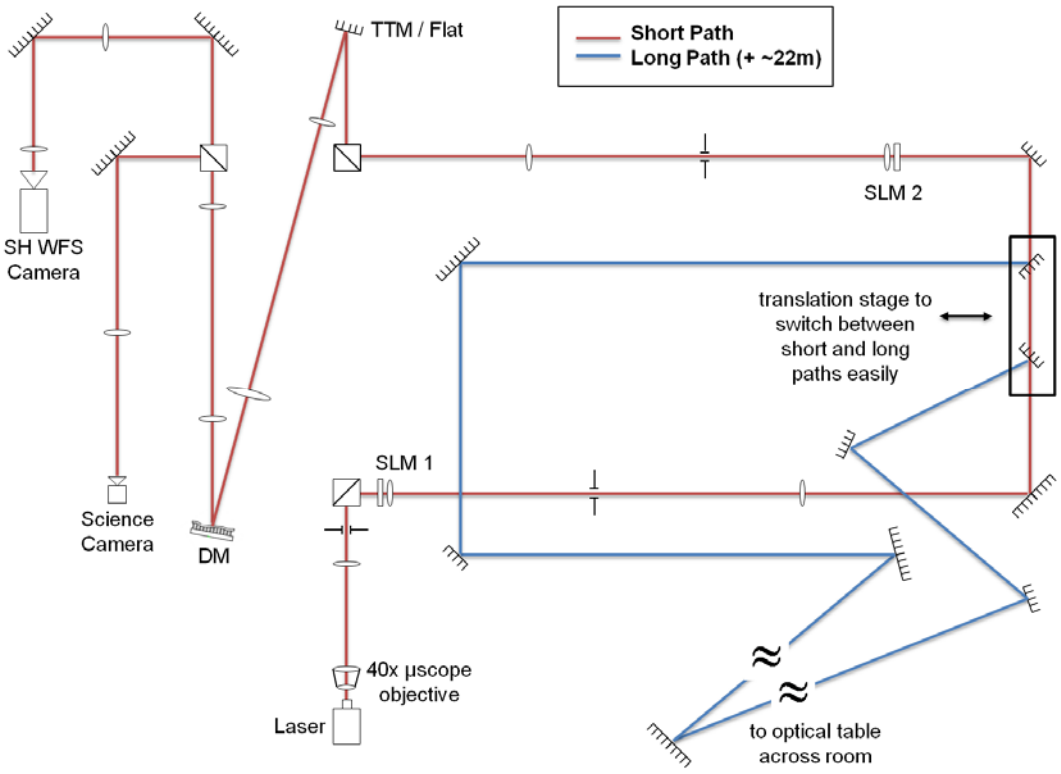

Figure 20. Schematic showing beam path extension and translation stage location

B. DEFORMABLE MIRROR

The deformable mirror used in this testbed is an OKO 37-channel micromachined membrane deformable mirror (MMDM). It is controlled by applying an array of voltages to electrodes on the back surface of the mirror. Figure 21 shows the actuator structure of the mirror.

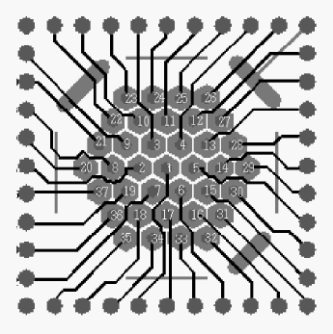

Figure 21. 37-channel MMDM (From OKO Technologies, 2008)

The membrane mirror is fixed on the outside rim while the surface deflects, yielding a quadratic relationship between applied voltage and mirror deflection. Because the electrostatic force is attractive, the mirror can only move in one direction from a flat reference, producing only concave shapes. In order to allow bidirectional control, the mirror is initially set at a biased position in the middle of its range of motion. Figure 22 depicts biased DM operation.

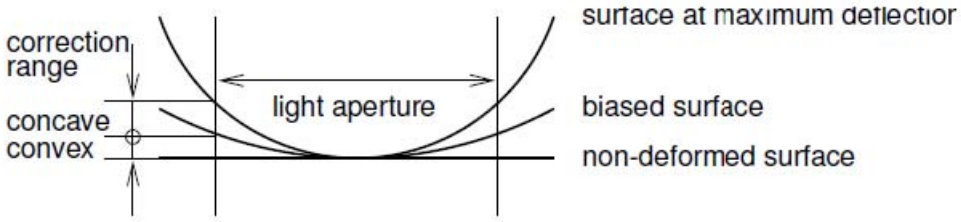

Figure 22. Biased DM operation (From OKO Technologies, 2008)

The AO system control algorithm uses Shack-Hartmann wavefront sensor slopes in determining the control signals to be applied to the DM actuators. These signals are scaled from -1 to 1, where -1 represents the maximum deflection in one direction, 1 represents the maximum deflection in the opposite direction, and 0 represents the biased position. The control signal is converted to a voltage signal between 0 and 255, which is converted in DM hardware to an actual voltage between 0 and 230 V. The relationship between the control signal (-1 to 1) and the voltage signal (0 to 255) is as follows:

$$V = \sqrt{0.5*(c+1)}*V_{\max} \tag{3.1}$$

To ensure that the maximum available throw of the mirror can be used effectively, the DM is tested to make sure the entire linear range of the mirror is in use. This is done by monitoring the slope response of a single sensor element to the action of a single actuator. Recording the wavefront slope produced by one Shack-Hartmann lenslet in the *x* direction, the voltage applied to one actuator was increased incrementally to determine if the mirror response saturated. Figure 23 shows that the sensor response was linear to the maximum control signal of 255. With this information, the correct bias signal

according to the relationship in Equation 3.1 for a control signal of zero is approximately 180.

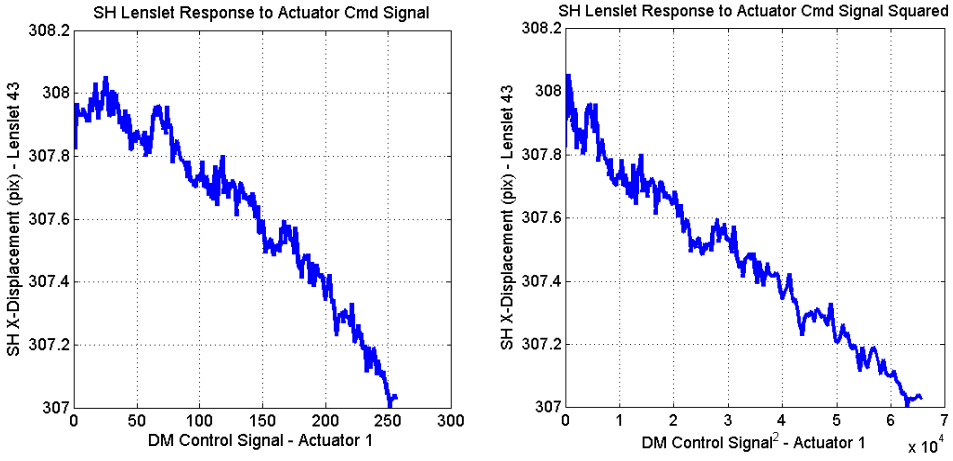

Figure 23. SH slope response vs. DM control signal

(Left) Slope response vs. DM control signal (Right) Slope response vs. DM control signal squared, showing linear response for entire voltage range

C. SHACK-HARTMANN WAVEFRONT SENSOR

The Shack-Hartmann (SH) wavefront sensor (WFS) is an OKO device with an array of 127 lenslets arranged in a hexagonal pattern. The array is attached directly to a Basler A601f camera with a resolution of 640 x 480 pixels and an 8-bit frame rate of approximately 20 fps. While it is desirable to use as many lenslets as possible to provide the best slope information of the wavefront, the beam size is decreased in the system to accommodate the size of the DM, and the full complement of lenslets is not illuminated by the beam. In the short path with higher intensity, 60 lenslets are used. In the long path, more light is lost due to diffraction and reflections, and 46 lenslets are used.

For wavefront correction, an initial reference image is taken before running an experiment. A center of mass centroiding algorithm is used to find the locations of the reference grid to be used in locating centroids for the duration of the experiment. The centroids are calculated from a 20x20 pixel box centered on the reference grid locations. This box is called a wavefront sensor subaperture. Once the reference centroids are calculated, they are used throughout the experiment as the reference centroids to which to

compare current centroids. The local wavefront slopes are calculated at each timestep by taking the difference of current centroid locations and reference centroid locations in pixels, and multiplying by the pixel width over the focal length to get slope angles in radians, as per Equation 1.1. Figure 24 shows a reference image with grid and centroid locations marked by asterisks.

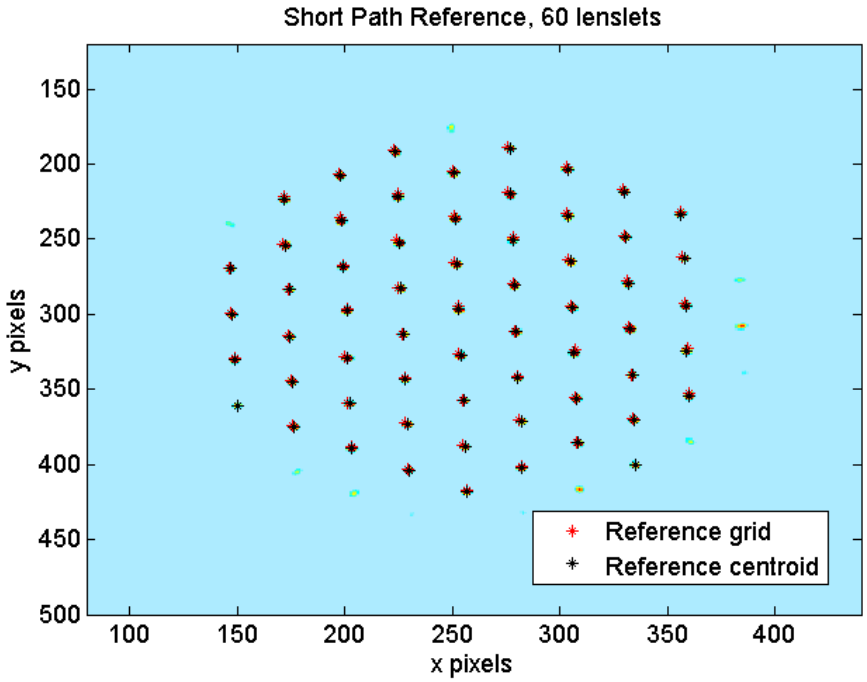

Figure 24. Image of SH array on CCD showing reference grid and centroid locations

D. SPATIAL LIGHT MODULATORS

The liquid crystal (LC) spatial light modulator (SLM) used in the testbed is a Holoeye LC2002 device with 800x600 pixels of resolution and an operational rate of 33Hz. It consists of a diffraction grating that modulates the incoming wavefront by π radians. To increase the modulation range to a full 2π, a Fourier filter in the form of an iris or aperture stop is placed in the beam to select either the +1 or -1 diffractive order to propagate through the system. Alignment biases are applied in software to separate the diffractive orders enough to pass through the desired order. Figure 25 shows the LC2002 on the left and its mounting in the testbed on the right.

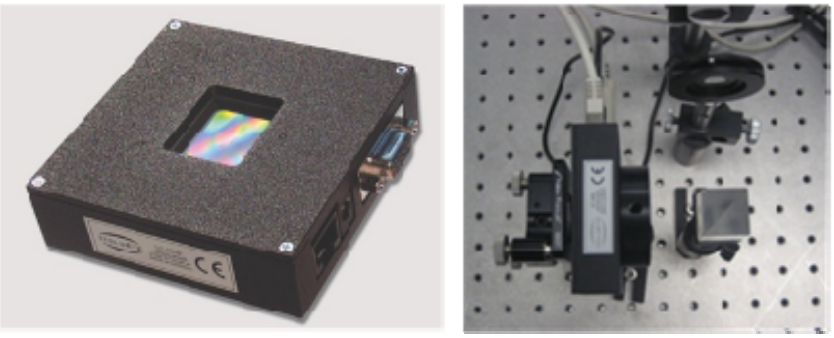

Figure 25. Holoeye LC2002 SLM

While the diffractive nature of the SLM decreases the available light and provides additional alignment challenges, the device provides a great deal of flexibility in generating and applying aberrations in the optical path. Atmospheric scenarios can be changed quickly in software without having to change hardware. This provides a significant advantage over other popular hardware such as rotating plates imprinted with specific atmospheric statistics, which cannot be changed without constructing new plates.

The Naval Research Laboratory (NRL) has developed software to apply atmospheric aberrations on the SLMs using a Matlab graphical user interface (GUI). The SLM control GUI is shown in Figure 26. Currently, the atmosphere generated in software is based on traditional Kolmogorov statistics used for astronomical applications as described in Chapter I. As analytical models are developed to describe horizontal turbulence and include effects particular to the maritime environment, the software can be modified to accommodate these model changes. In the meantime, thick aberrator effects are created in the lab using the long path extension and will be described in Chapter V.

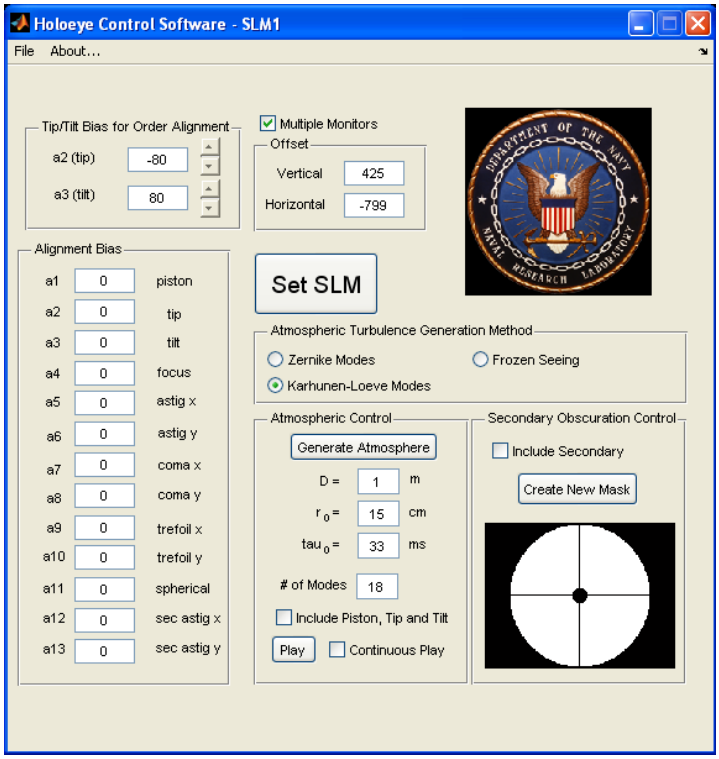

Figure 26. Screen capture of SLM control GUI

The NRL software GUI allows the user to specify telescope and site parameters such as the telescope diameter, D, and the atmospheric coherence length or Fried parameter, r_0. Turbulence generation can be performed with the Zernike polynomial expansion or the Karhunen-Loève expansion. The Karhunen-Loève modes are used in this research as they contain a statistically independent set of coefficients based on Zernike modes, and are often used in turbulence simulation. The K-L modes are used with a new method developed by Wilcox for simulating smoothly transitioning phase screens.

Once site parameters, a polynomial mode set, and any initial alignment biases or static aberrations are selected, the controller generates a user-specified number of phase screens to represent the atmospheric simulation. To include the random nature of the atmosphere, Wilcox augments the K-L polynomial expansion shown in Equation 1.7 to include Tatarskii's assumption of a Gaussian random distribution in phase variances due to turbulence. The wavefront is then described by:

$$\text{Wavefront}(\rho,\theta) = \sum_{i=1}^{M}(1+X_i)a_iK_i(\rho,\theta) \tag{3.2}$$

where X_i is the Gaussian noise for each mode. These X_i values can be generated in software by using a random number generator with a Gaussian distribution (Wilcox, 2009).

To provide a smooth transition between frames of atmosphere, the random numbers must become a continuous function of time, expressed as:

$$\text{Wavefront}(\rho,\theta) = \sum_{i=1}^{M}(1+X_i(t))a_iK_i(\rho,\theta) \tag{3.3}$$

Wilcox produces this continuous function by generating a vector of random numbers and then fitting a spline curve to the vector to represent the temporal progression of atmospheric turbulence. A sample spline curve is shown in Figure 27. The random number generation and spline curve fitting are repeated for each mode. When combined with the K-L modes, a realistic realization of smoothly transitioning atmospheric turbulence is created. The simulation is then applied on the SLM and run at a user-specified rate up to the device's operation rate of 33 Hz. The simulation can be changed easily and quickly as new atmospheric and site parameters are desired for testing, and the underlying code can be changed as new models of turbulence become available.

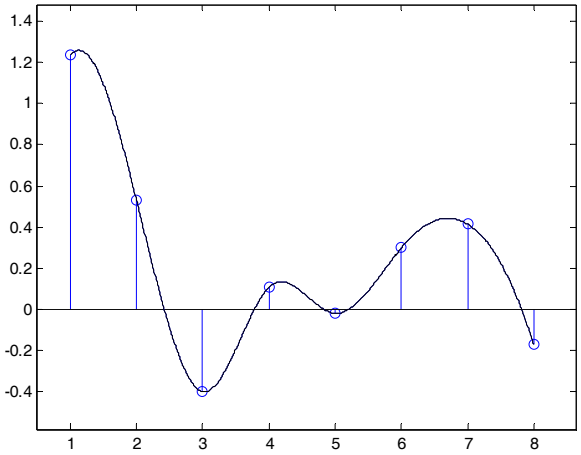

Figure 27. Sample $X_i(t)$ function for smooth transitions (From Wilcox, 2009)

E. OTHER COMPONENTS

The laser used is a continuous wave CVI Melles Griot Helium Neon Class II laser with output power of 0.5 mW, operating at a wavelength of 633 nm. The science camera is an IDS uEye-2210SE CCD camera with a resolution of 640 x 480 pixels and an 8-bit frame rate of 75 fps. It is used to capture images of the corrected and uncorrected beam. Other optical components on the table include lenses, mirrors, aperture stops, beamsplitters, and filters which reimage the system pupil plane and collimate, expand, and shrink the laser beam as needed to propagate to the sensing and correcting elements. The optical components used are primarily produced by Edmund Optics, Newport/New Focus, Thor Labs, and CVI Melles Griot.

Two computer controllers are used for the full experimental system. The deformable mirror and Shack Hartmann WFS are driven by one computer in Matlab's Simulink environment. Various control algorithms are implemented in Simulink for testing and comparison, and the AO elements are driven directly from the program using Simulink's S-function capability. The SLMs and uEye science camera are controlled by a separate computer, with the atmosphere running independently of the AO control system. Figure 28 shows the displays for each control setup. The AO control monitor is on the right. In the four-monitor system on the left, each SLM has its own monitor showing the applied phase screen, the science image is shown in the bottom right, and the SLM control GUIs are shown in the bottom left.

Figure 28. Computer control monitors

THIS PAGE INTENTIONALLY LEFT BLANK

IV. BEAM CONTROL RESULTS

This chapter presents simulation and testbed control results for the algorithms evaluated in this research. The adaptive filters are tested in simulation with a generic sinusoidal disturbance, and the adaptive and Kalman filters are implemented in the testbed. Results are shown from the testbed, followed by simulations performed with a testbed–generated disturbance.

A. SIMULATION WITH GENERIC DISTURBANCE

In order to understand and develop confidence in the implementation of adaptive filters in Matlab's Simulink environment, simulations with a generic disturbance are performed prior to using an atmospheric testbed disturbance. The simulations also give insight into which of the LMS adaptive filter implementations should be transitioned to the testbed. The only testbed component used is the system poke matrix.

The simulated disturbance consists of three sinusoids and band-limited white noise. The sinusoids have randomly chosen frequencies of 12.5 Hz, 8.2 Hz, and 1.0 Hz, with randomly chosen phases of 2.98 rad, 1.37 rad, and 0 rad, respectively. The white noise power is 1.0^{-10}. The disturbance is copied into all of the sensor channels so that each channel sees the same disturbance. The LMS algorithm uses 20 tap weights in each channel. It is expected that the adaptive filters will perform better than the classical PI loop in the presence of the sinusoidal disturbance. In all simulations, the PI controller is turned on immediately and the adaptive filter is turned on at 1 second.

Figure 29 shows the RMS error over all the channels for 20 second simulations using the three filtered-x (FX) LMS adaptive filter implementations described in Chapter II: sensor space, actuator space, and decoupled actuator space. All three models are tested with a convergence coefficient of $\alpha = 0.01$. The coupled actuator space model converges almost immediately, while the decoupled model has a longer transient period. The sensor space model converges negligibly in the course of the simulation and is overlaid on the disturbance itself.

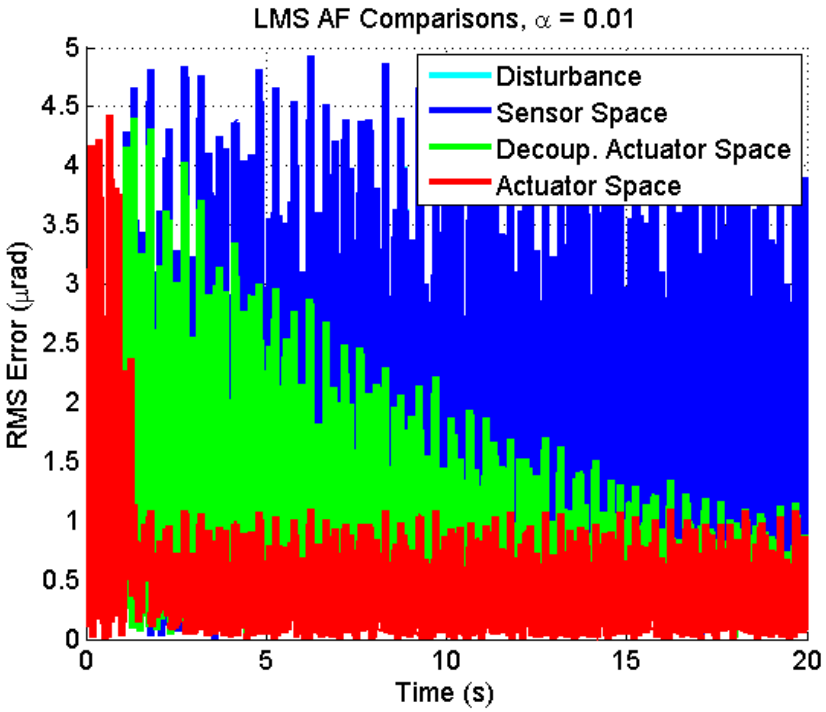

Figure 29. LMS AF configurations

Results show RMS wavefront error over the aperture versus time.

The weights for one channel for each LMS implementation are shown in Figure 30. As shown by the RMS error results, the coupled actuator space model is the only one whose weights converge in the 20-second time period. This indicates that indeed there is some interaction between the sensor and actuator channels that prevents them from being treated as completely decoupled. The weights for the coupled model converge at about four seconds, or three seconds after being turned on. However, the error is reduced to approximately steady state almost immediately after being turned on, indicating that the LMS algorithm can control successfully while its weights are converging. From these results, the coupled actuator space model appears to be the best candidate for testbed implementation.

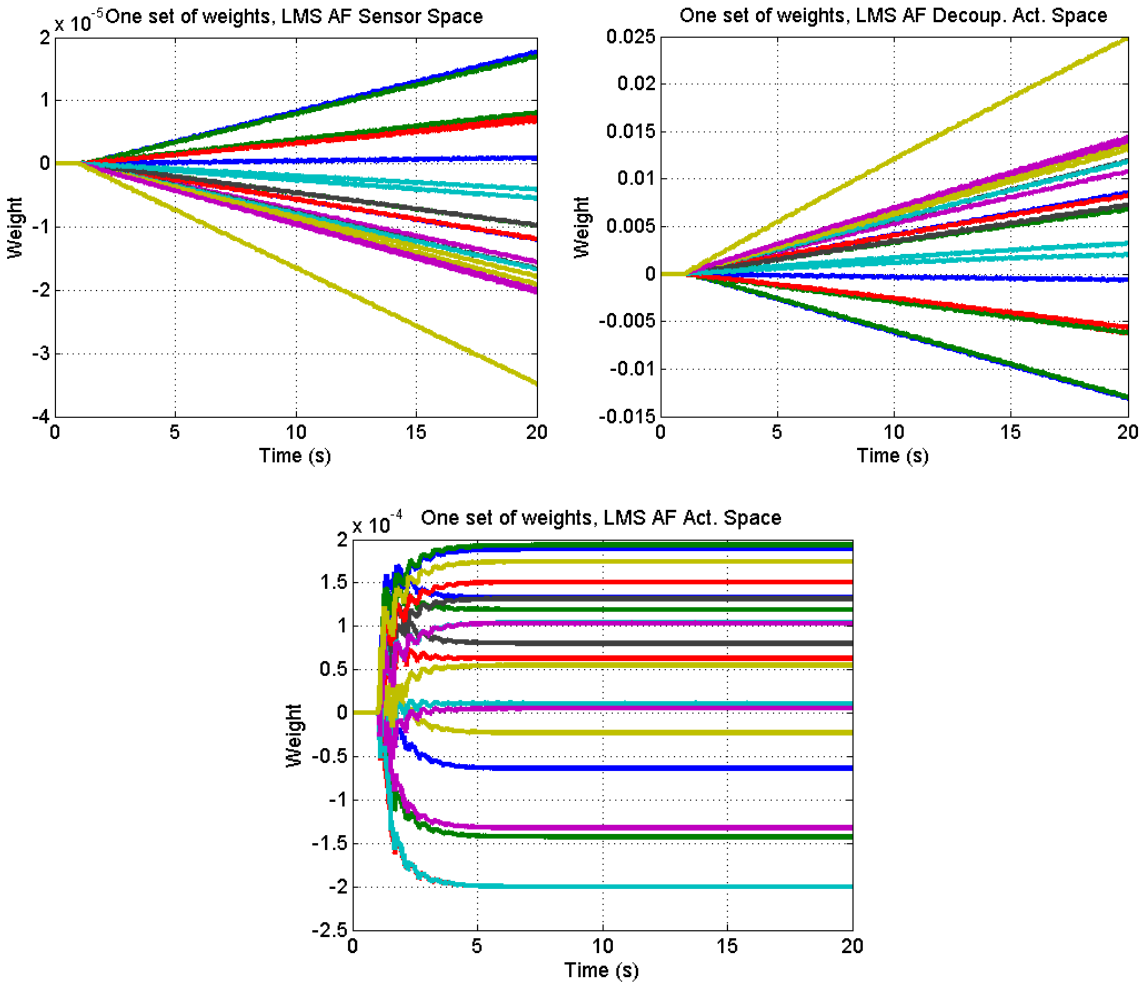

Figure 30. One channel of weights for each LMS AF implementation

The sensor space model is expected to converge in time, though its larger number of channels and weights makes it very slow. Figure 31 shows the results of running the sensor space model for 100 seconds with a convergence coefficient of $\alpha = 10$, to ensure that the model does perform as expected. However, the long convergence time eliminates this model from efficient testbed implementation.

Figure 32 shows the RMS error results of simulating both the PI and LMS AF controllers. In adding the PI controller, the adaptive filter convergence coefficient was reduced to $\alpha = 0.001$ for better performance. This leads to the slower convergence of the adaptive filter algorithms seen in Figure 32. The PI improves sensor space model

performance, though the PI controller alone performs better than this combination. The coupled actuator space model still converges quickly to the lowest steady state error.

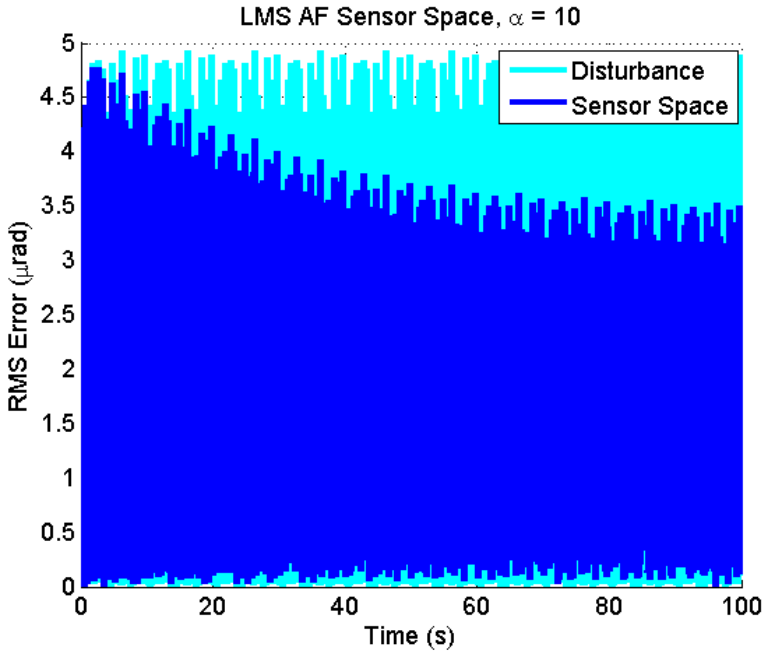

Figure 31. LMS AF sensor space model convergence

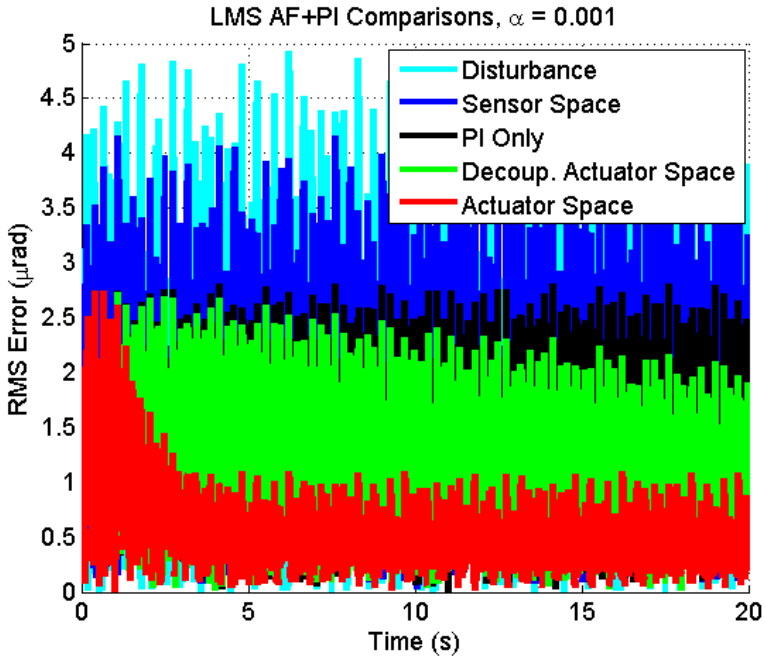

Figure 32. LMS AF implementations with PI

The weights for the LMS controller with PI are shown in Figure 33. As before, the sensor space and decoupled actuator weights do not converge in the given time. The coupled actuator weights show slightly more wandering than in the adaptive filter working alone. Since the PI does some of the work, the adaptive filter weights can take on multiple values and still lead to good control performance. Again, it appears that the coupled actuator space model will perform the best and should be implemented on the testbed.

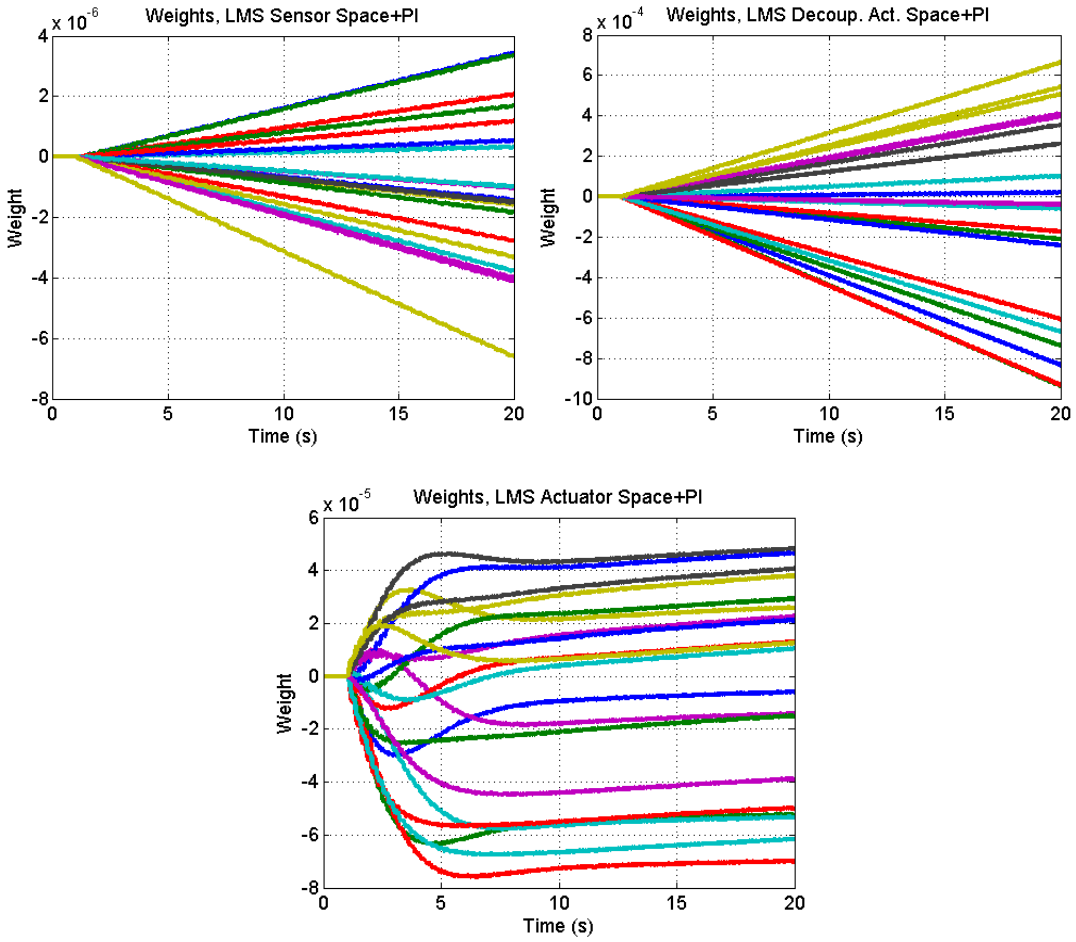

Figure 33. One channel of weights for each LMS AF + PI implementation

Results showing the LMS transverse filter performance compared to RLS lattice filter performance are shown in Figure 34. The RLS lattice converges slightly faster than the LMS transverse, which is expected especially for a generic sinusoidal disturbance. The models converge to approximately the same steady state error. An inset showing

LMS overlaid on RLS shows the steady state reached slightly faster in RLS than LMS. The RLS lattice filter weights are not available for comparison in the model output.

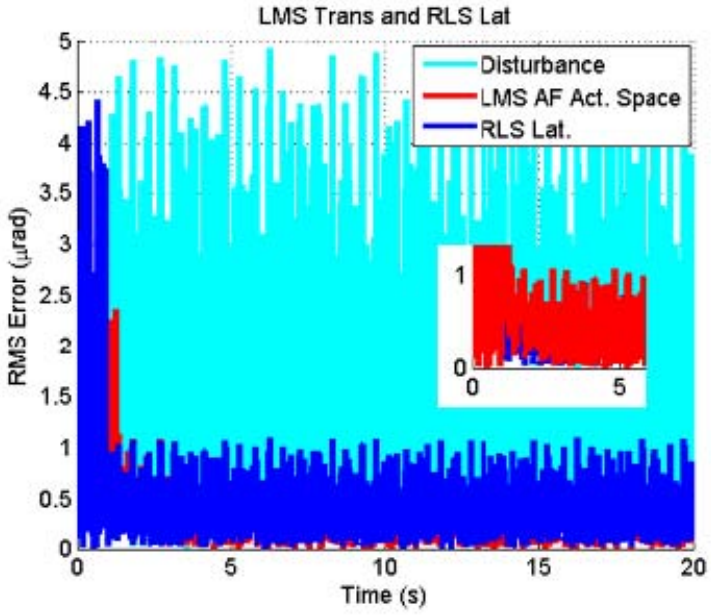

Figure 34. LMS transverse and RLS lattice comparison

Figure 35 shows LMS and RLS compared with the addition of the PI controller. The inset shows that once again the RLS algorithm converges slightly faster. However, the RLS steady state error is higher in the presence of the PI. While the RLS filter provided by Liu and Gibson does not follow the Filtered-x formulation, it does account for the transfer function of the classical PI loop in providing an estimated disturbance. Further exploration of the interaction between the RLS lattice and PI loops in a sinusoidal disturbance can be performed in collaboration with Liu and Gibson. The results here indicate that in some cases the adaptive filters working alone can perform better than the AF + PI implementations.

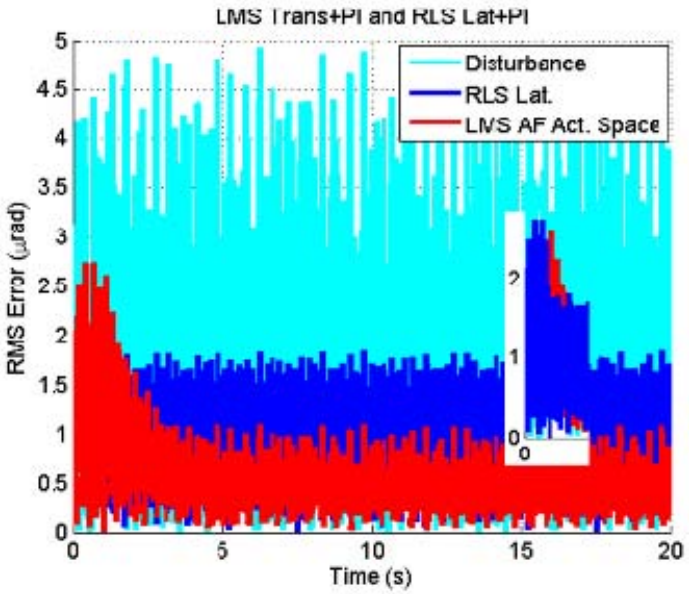

Figure 35. LMS transverse + PI and RLS lattice + PI

B. TESTBED RESULTS

1. Testbed Disturbance

All testbed results presented in this chapter are obtained from applying an atmospheric profile on SLM 2 only, the second SLM encountered in the beam path. This SLM is located in the system pupil plane, meaning it is conjugate to the wavefront sensor and the deformable mirror, or that the image of the aberrations is seen in the same way at all these planes. This is done so that pure phase aberrations would be introduced into the system, allowing comparisons of algorithm performance in an atmospheric environment giving the best chance of success before introducing the more challenging deep turbulence scenario.

The testbed disturbance consists of the SH WFS data obtained from applying atmosphere on SLM 2 with no controller in the loop. The atmospheric profile generated is for a telescope aperture of 1 m diameter and an atmospheric coherence length of $r_0 = 15$ cm, representing an atmosphere of medium strength. The atmosphere is run at 7.5 Hz on the SLMs, as the AO loop using a Simulink hardware interface can run currently at a maximum rate of 15 Hz. This rate is limited by the camera and can be

improved with the introduction of a camera with a faster frame rate. In reality, the atmosphere changes more on the order of 100 Hz. If desired, the testbed disturbance can be artificially sped up in simulation by decreasing the sample time of the controller.

2. Hardware Control Results

The PI, adaptive filter, and Kalman filter algorithms have been implemented on the testbed. Figure 36 shows science camera images of the SLM reference beam, the primary SLM beam passing through the DM as well, and a frame of uncorrected atmospheric aberration. The beam shape in the DM image is due to aberrations in the mirror itself, which can limit the throw and control available in the DM. Figure 37 shows the science camera images with correction algorithms applied to the frame of atmosphere. The PI, LMS AF alone, and LMS AF + PI all perform comparably in driving the aberration towards the shape of the reference. The Kalman + PI drives slightly better, though its performance decreases after a few seconds because it encounters command saturation problems. The RLS lattice algorithms encounter command saturation as well and do not work sufficiently well with testbed hardware to produce a a good science camera image. This remaining challenge with saturation of DM commands in hardware makes algorithm performance more effectively compared without hardware limitations. As a result, for faster computation and comparison, the RMS error results presented next are generated using the testbed Simulink models and testbed disturbance, but using the system poke matrix instead of the hardware interface. Further investigation into the causes of and differences between simulated and hardware control of the testbed disturbance must be performed.

Figure 36. Science camera images with no applied control

(a) Reference beam (b) Primary path with DM, no atmospheric aberration (c) Frame of atmosphere applied on SLM 2

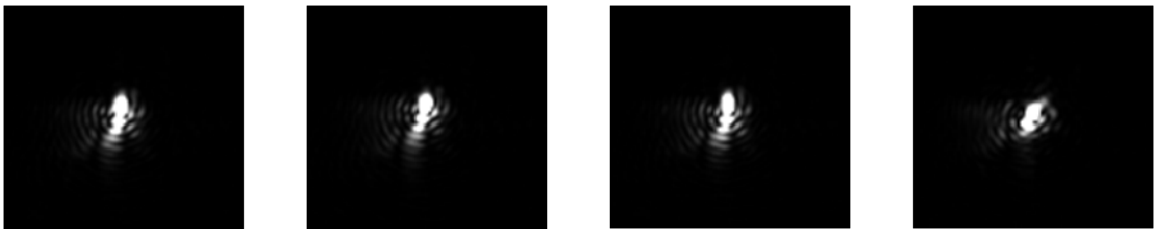

Figure 37. Control applied to frame of atmosphere
Driving atmosphere to reference with (a) PI (b) LMS AF (c) LMS AF+PI (d) Kalman+PI

3. PI Controller Gain Selection

Some iteration is performed to determine the proportional and integral gains to be used in the classical PI controller loop. Since algorithm implementation using the testbed disturbance produces controller commands that saturate the +/- 1 limits of the DM, tradeoffs are made between algorithm performance and the level of control effort required for each gain combination. The control effort required by the LMS AF alone is independent of the PI gain selection and is used for comparison. The PI controller is designed using the zero-pole-gain form by selecting gain, k_1, and zero, z_1, values and then converting to the PI transfer function form:

$$\frac{k_1(z-z_1)}{z-1} = \frac{(K_i + K_p)z - K_p}{z-1} \tag{3.4}$$

where $K_p = k_1 z_1$ and $K_i = (k_1 - K_p)$. The K_i values shown include the discrete sample time, Ts. Table 1 shows three cases, the first with a much larger gain than zero value, the second with equal values, and the third with a smaller gain than zero value. RMS error results for each case are shown in section 4. The first case with the highest gain shows the best RMS error performance, but as a result has the lowest gain margin and highest control effort. The control effort in the third case is the lowest and most comparable to the control effort required by the LMS adaptive filter alone. For the slight reduction in RMS performance over Case 2, Case 3 uses slightly less control effort and doubles the gain margin. For the combination of sufficient performance, highest gain margin, and lowest control effort of the cases presented, the gains in Case 3 are used for the testbed disturbance results in this research.

Table 1. PI Gain Selection

	z1	k1	Ki	Kp	GM (dB)	PM (deg)	uMin	uMax
Case 1	0.1	0.7	0.63	0.07	2.60	75.0	-2.31	1.79
Case 2	0.4	0.4	0.24	0.16	3.57	92.0	-1.81	1.74
Case 3	0.3	0.2	0.14	0.06	7.69	89.4	-1.66	1.71
	LMS AF						-1.67	1.51

4. RMS Error Results

RMS error results are shown as in simulation for the following algorithms: PI, LMS AF, LMS AF + PI, RLS Lattice, RLS Lattice + PI. The PI controller is turned on immediately, and the adaptive controllers are turned on at 1 second. The atmosphere is recorded from the testbed running at 7.5 Hz. For the slow atmospheric rates achievable with current hardware, it is expected that the PI controller can perform well even without the adaptive filter. However, the predictive ability of the adaptive filter should improve PI performance. The RMS error results show the disturbance without control as well as the reference for a clean beam bypassing the DM and passing through the SLMs with no atmosphere applied. This represents the best control that could be achieved in the system.

Figures 38–40 show the RMS error results for the PI controller gain cases specified in Table 1. As described, Case 1 has the highest gains and performs well enough alone that the addition of the LMS AF yields no further performance improvement. Both algorithms perform better than the LMS AF alone in this case. However, the performance is achieved at the cost of a large control effort and low gain margin. Figure 39 shows the PI performance slightly reduced as expected for the lower gains of Case 2. In this case, the PI controller performs better than the LMS AF alone, though the combination of LMS Af and PI works better than either alone. Case 3 showed the most comparable control effort between the PI and LMS AF controllers. Figure 40 shows that for approximately the same control effort by each working alone, the LMS AF is able to achieve lower error than the PI. The controllers working in combination increase the control effort slightly but reduce the error even further, so that again the

combination performs better than either algorithm working alone. When the error reduction performance is balanced with control effort and gain margin, Case 3 is chosen as the best PI controller.

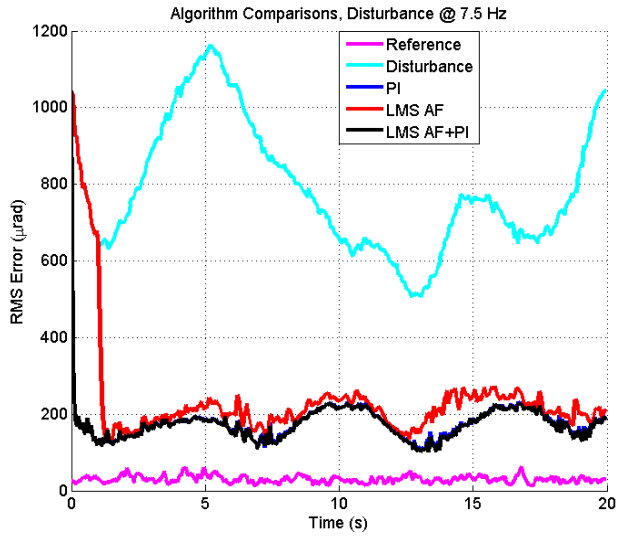

Figure 38. PI and LMS algorithms compared for Case 1 gains

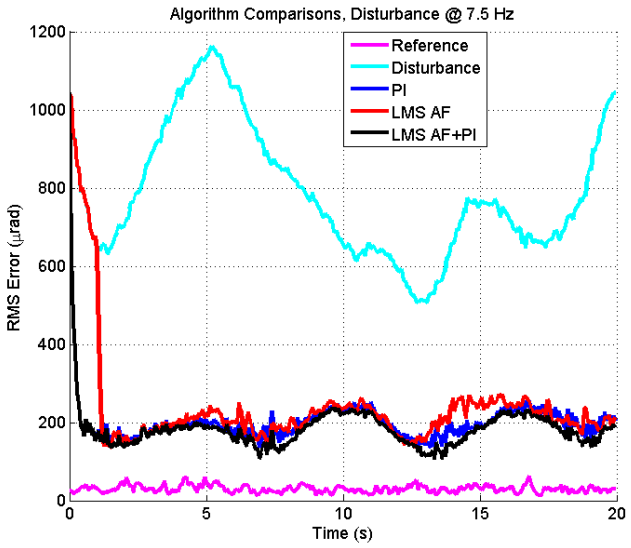

Figure 39. PI and LMS algorithms compared for Case 2 gains

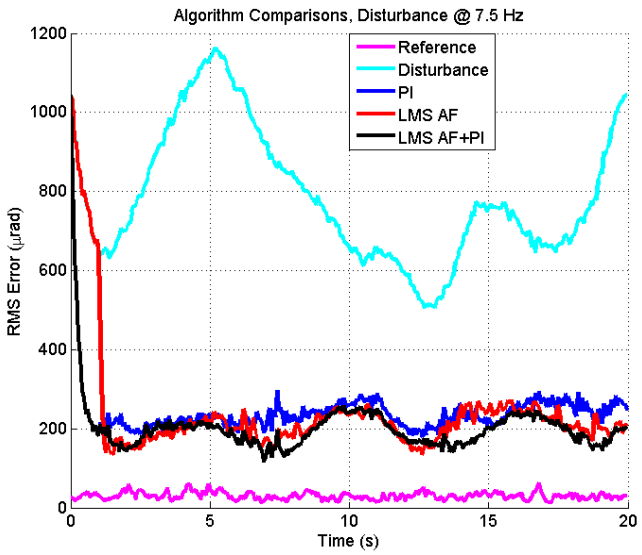

Figure 40. PI and LMS algorithms compared for Case 3 gains

Table 2 shows the average RMS error in comparing the PI, LMS AF, and LMS AF + PI algorithms. As reflected in Figure 40, the LMS AF works slightly better than the PI alone, while the combination of LMS AF + PI works the best overall.

Table 2. Average RMS error for PI, LMS AF, LMS AF + PI

Algorithm	Avg RMS Error (μrad)
LMS AF + PI	189.3
LMS AF	214.0
PI	238.8

Figure 41 shows the LMS AF and LMS AF + PI weights for the Case 3 PI controller gains. The weights did not converge as easily in the testbed cases as in simulation. Since the disturbance is non-periodic in nature, it appears that the adaptive filter has more difficulty converging to steady state values, even when error performance is good. Once again, the LMS algorithm can demonstrate control performance during the convergence period of its weights. With and without the PI controller, the weights follow similar trends, with several remaining near zero and most following similar trajectories with different values.

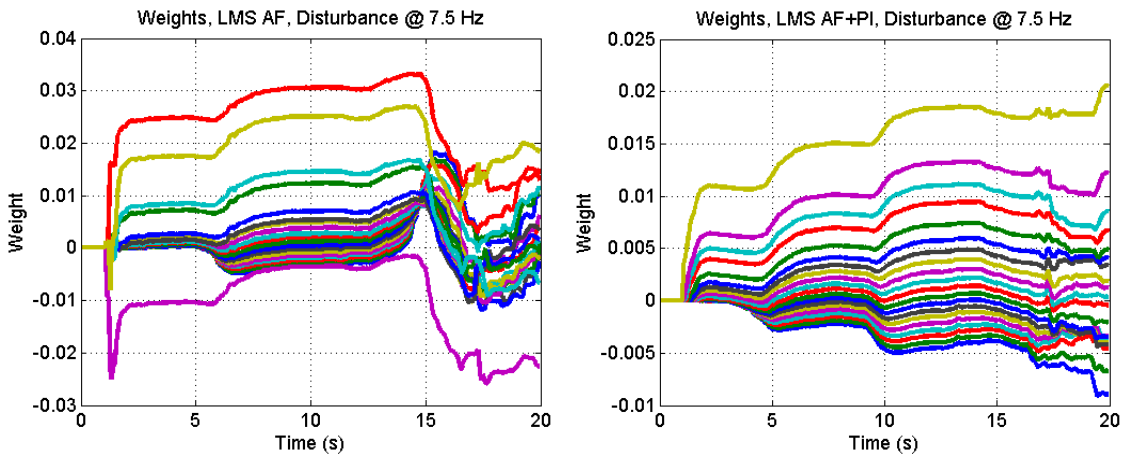

Figure 41. Weights, (Left) LMS AF, 7.5 Hz (Right) LMS AF + PI, 7.5 Hz

The RLS lattice filter shows interesting results when implemented in the testbed model. The model provided by UCLA is designed so that it does not control during the convergence period of its weights. Any control signal injected into the system during convergence causes a high peak in the RMS error. To account for this, the filter uses a learning time during which it outputs a zero control signal but accepts reference information to estimate the necessary gains. Figure 42 shows the RMS error results for the PI, LMS AF, and RLS Lattice cases with the RLS learning time set so that RLS turns on at 1 second, just as for LMS. It appears that for this atmospheric disturbance, the filter has difficulty determining the appropriate gains in a reasonable time, taking just over half the simulation time to converge. The RLS weights are not available as output from the model, but the results in Figure 42 indicate that the algorithm does converge after approximately 11 seconds. While the LMS weights in Figure 41 oscillate somewhat themselves and do not reach a clear convergence in the simulation time, the LMS algorithm still manages to reduce the RMS error without taking a significant amount of time for the initial error reduction. As expected from analytical and simulation results, it is possible that the RLS algorithm does in fact converge faster than the LMS algorithm, yet at the same time the LMS algorithm can control significantly better during its convergence time than the RLS model used in this research.

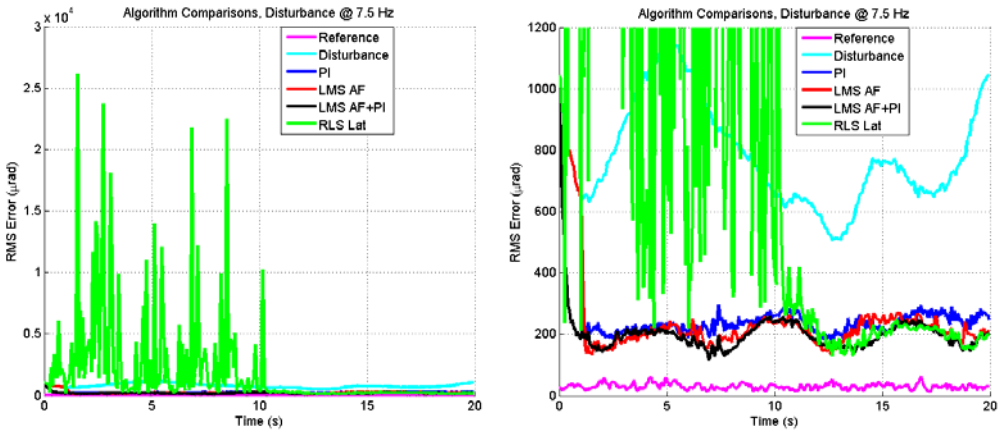

Figure 42. RLS lattice convergence time

Figure 43 shows the LMS and RLS results while increasing the learning time for the RLS so that it is turned off until convergence. The RLS Lattice + PI is also included. The addition of the PI controller to the RLS allows at least the PI level of control during the time the RLS converges. Furthermore, the maximum control effort required by the RLS Lattice + PI algorithm is lower than that of the RLS Lattice alone. These results indicate that it is very beneficial for the PI and RLS Lattice to work together not just for performance, but also for control. The transverse LMS AF, however, can perform almost as well as its counterparts without needing the PI controller as well.

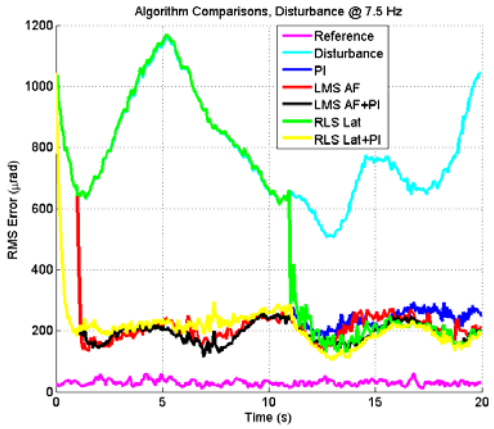

Figure 43. LMS, RLS, and PI algorithms compared at 7.5 Hz
PI slightly improves RLS performance, LMS AF + PI performs comparably.

Finally, the Kalman filter is added and its performance compared to that of the LMS and RLS adaptive algorithms. For the Kalman filter, the process and measurement noise were estimated iteratively to determine noise powers that reasonably represented the disturbance. Since the disturbance from the testbed contained its own measurement noise, a low level of additional measurement noise was added with a power of 1.0^{-9}, which was also used as the measurement noise covariance. Various levels of process noise were tried, with the process modeled successfully using a noise power of 1.0^{-6} for the process noise covariance.

Figure 44 shows the RMS error results comparing the PI alone and each of the filters augmenting the PI. Adaptive filters working alone are not included here. The Kalman + PI performs as well as or better than the RLS Lattice + PI, with both of these algorithms performing slightly better than the LMS AF + PI. However, it is important to remember that the Kalman filter uses a disturbance model that is generated from the exact disturbance being injected into the system. In a real scenario, this disturbance model must be estimated online, requiring additional sophistication and complexity.

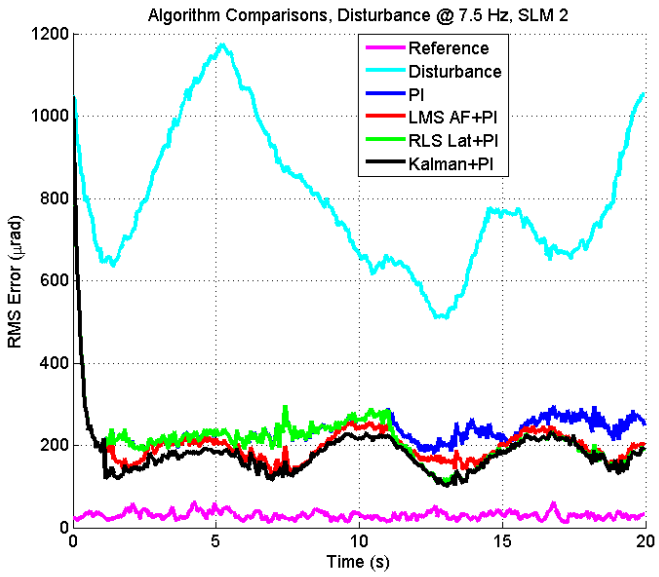

Figure 44. PI and augmentations

LMS AF + PI performs almost as well as RLS Lattice + PI and Kalman + PI; all perform comparably in reducing error.

Table 3 shows the average RMS error of the PI and PI augmentation algorithms. As reflected in Figure 44, the Kalman + PI performs the best, slightly better than the RLS Lattice + PI. The LMS AF + PI performs almost as well as these algorithms, without requiring the longer time to begin controlling that the RLS requires, and without requiring the disturbance model that the Kalman filter requires. All advanced filters show improved performance over the classical PI controller working alone.

Table 3. Average RMS Error for PI and Augmented PI Algorithms

Algorithm	Avg RMS Error (μrad)
Kalman + PI	169.3
RLS Lat + PI	171.1
LMS AF + PI	189.3
PI	238.8

The final minimum and maximum control efforts for each of the six algorithms evaluated are shown in Table 4. All show similar control efforts, with the exception of the higher effort of the RLS Lattice working alone. The PI reduces this control effort to be more comparable to the control efforts of the other algorithms. Of the augmented PI algorithms, the Kalman + PI shows the highest control effort range.

Table 4. Algorithm Control Efforts

Algorithm	uMin	uMax
PI	-1.66	1.71
LMS AF	-1.67	1.51
LMS AF + PI	-1.77	1.73
RLS Lat	-2.79	1.90
RLS Lat + PI	-1.66	1.88
Kalman + PI	-1.80	1.84

Testbed results indicate that for the system developed in this research, the LMS adaptive filter can perform sufficiently well for demonstration and research purposes in the compensation of atmospheric turbulence. The LMS AF alone performs comparably to the converged RLS lattice and decreases sensor slope error significantly faster during

convergence. Compared to the classical PI controller tested in this research, both LMS and RLS adaptive filters perform better. However, combining the PI and adaptive controllers improves the performance over each working alone. The Kalman + PI algorithm performs the best overall in reducing the wavefront slope error, though its control effort range is the highest of the augmented PI algorithms. The Kalman filter requires a disturbance model that must be determined online in a real scenario, and the RLS lattice requires a long learning time before it can effectively control or augment the PI in the current testbed. The transverse LMS adaptive filter has the simplest structure, requires neither a disturbance model nor a long learning time, and performs almost as well as the more complex algorithms in reducing the wavefront slope error. For these reasons, the LMS adaptive filter + PI combination is the best controller candidate in the current testbed and successfully simplifies the current adaptive optics control.

THIS PAGE INTENTIONALLY LEFT BLANK

V. DEEP TURBULENCE SIMULATION

This chapter describes the testbed modifications performed to create a deep turbulence atmospheric scenario in the laboratory. First, the effects of deep turbulence are reviewed, followed by the rationale for extending the propagation path between the SLM–generated atmospheric profiles. Next, intensity fluctuations observed in the laboratory are described, as well as a method of detecting intensity dropouts. Atmospheric profiles with and without intensity dropouts are presented for four different rates of atmosphere applied on the SLMs. The chapter concludes by presenting initial beam control results in the presence of laboratory—generated deep turbulence effects.

A. DEEP TURBULENCE EFFECTS

Deep turbulence is an integral part of the atmosphere in a maritime environment. Scintillation and branch point effects increase as propagation distance increases, because phase variations in the wavefront propagating over a sufficiently long path will begin to interfere with each other and develop amplitude variations. Locations where the intensity of the beam is zero cause discontinuities or singularities in the phase, making it more difficult for classical least squares reconstructors to determine appropriate control commands. These amplitude variations and phase discontinuities do not correspond to the phase aberrations classical wavefront sensors can detect, introducing a challenging scenario for beam control algorithms to handle.

Since it is known that branch points and scintillation are characteristic of the deep turbulence problem, the testbed developed for this research is modified to simulate the effects of intensity fluctuations and intensity dropouts on the Shack-Hartmann WFS. This is accomplished by applying atmosphere on two separate SLMs, both individually and simultaneously, and extending the beam path between them to observe the atmospheric disturbances produced. The success of this experiment lays an important foundation for simulating maritime-like horizontal atmosphere in the laboratory for beam control in HEL ship systems.

B. SLM PLANE SEPARATION

The laboratory testbed was originally designed to incorporate two SLMs so the effects of applying atmosphere at two different planes in the system could be studied. SLM 2 is placed in the system pupil as described in the testbed results so that atmosphere applied there can serve as the control with which to compare atmosphere applied at other locations in the system. SLM 1 is placed 18.5 inches upstream of SLM 2. However, it is expected that this separation is not long enough to produce true effects of deep turbulence. The beam path extension described in Chapter III extends the separation distance between SLM 1 and SLM 2 to approximately 22 m.

Figure 45 shows a conceptual diagram of the configurations available. In the short path, the image of aberrations applied at SLM 2 will be sensed accurately at the wavefront sensor since they are in conjugate planes. When the atmosphere is applied at SLM 1, some additional propagation distance will be included. However, it is expected that the propagation distance is not long enough to cause deep turbulence effects, and that the wavefront sensor reconstruction of the phase aberrations will still be fairly comparable to those of SLM 2. If the error is similar to that of SLM 2, it can be concluded that some uncertainty in determining the location of the aberration plane in a real scenario can be accepted.

In the long path shown in Figure 45, it is expected that the image of aberrations applied at SLM 2 will include some contribution from atmospheric effects in the room, leading to higher variations in the wavefront slope error. However, since the SLM atmospheric phase profile is still applied in the system pupil plane, the phase aberrations should still be sensed fairly accurately at the wavefront sensor. On the other hand, when phase aberrations are applied at SLM 1 in the long path, upstream of the beam path extension, they propagate through a much longer distance before reaching the wavefront sensor. It is at this point that intensity fluctuations and dropouts are expected to contribute significantly to the disturbance profile sensed at the wavefront sensor.

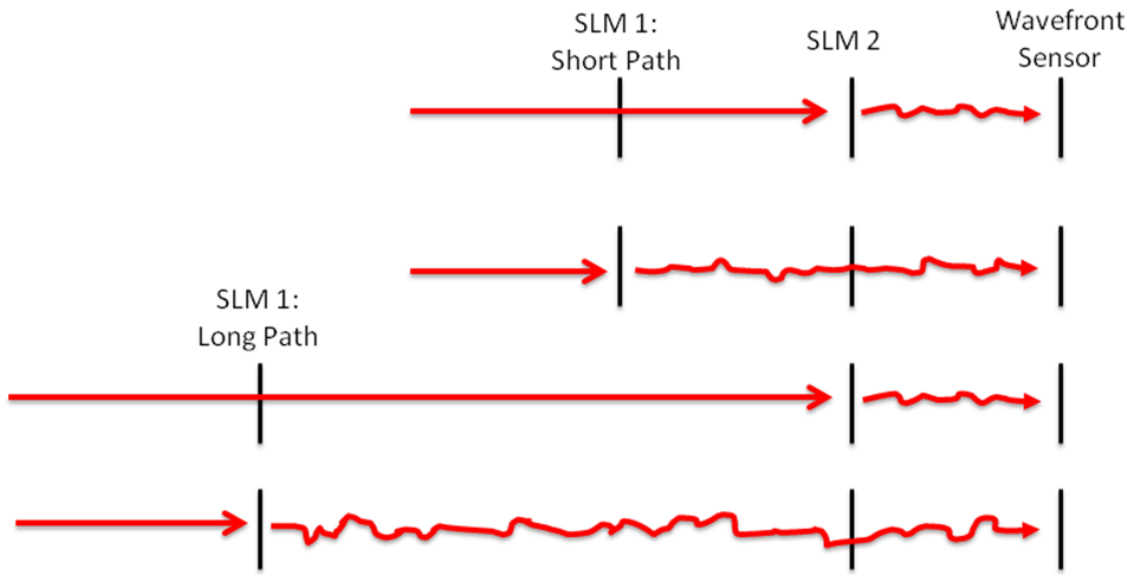

Figure 45. Visualization of SLM and beam path combinations

C. INTENSITY FLUCTUATIONS

Before applying atmosphere to the SLMs in the long path, wavefront sensor images of the SLM reference beam are recorded in the long path to determine whether the path extension through ambient atmosphere alone produces noticeable changes in the intensity profile of the laser beam. The wavefront sensor images show that there is indeed a significant difference in the behavior of WFS lenslet spots on the camera. The long path beam visibly fluctuates across the lenslets more than the short path beam. While the fluctuations are more apparent in video, Figure 46 and Figure 47 show a series of WFS images in the short and long paths, respectively. The images show every other frame of 12 frames for each path. The overall intensity level is lower in the long path due to diffraction effects and the increased number of mirror reflections. The fluctuations rather than the intensity represent the differenc between the image sequences. The six images shown for the short path look identical, whereas the six frames shown for the long path have some spots that differ from frame to frame. The simplest variation to see is the changing intensity of brighter hot spots in some of the central lenslets throughout the long path frames. Video data collected by the NRL in Puerto Rico over a 110 m propagation path over water in 2003 shows very similar intensity fluctuation behavior.

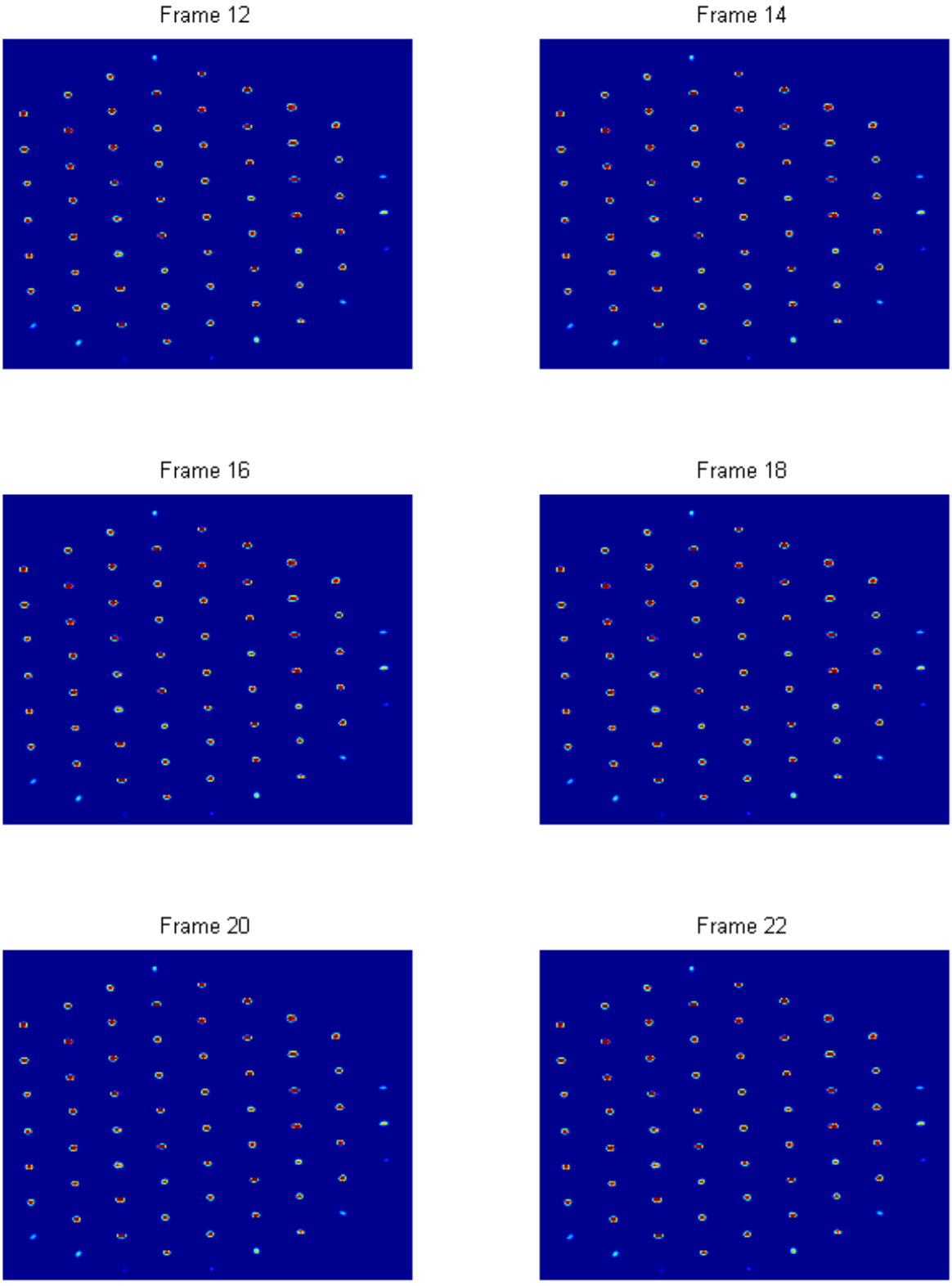

Figure 46. WFS images in short path, no SLM aberrations

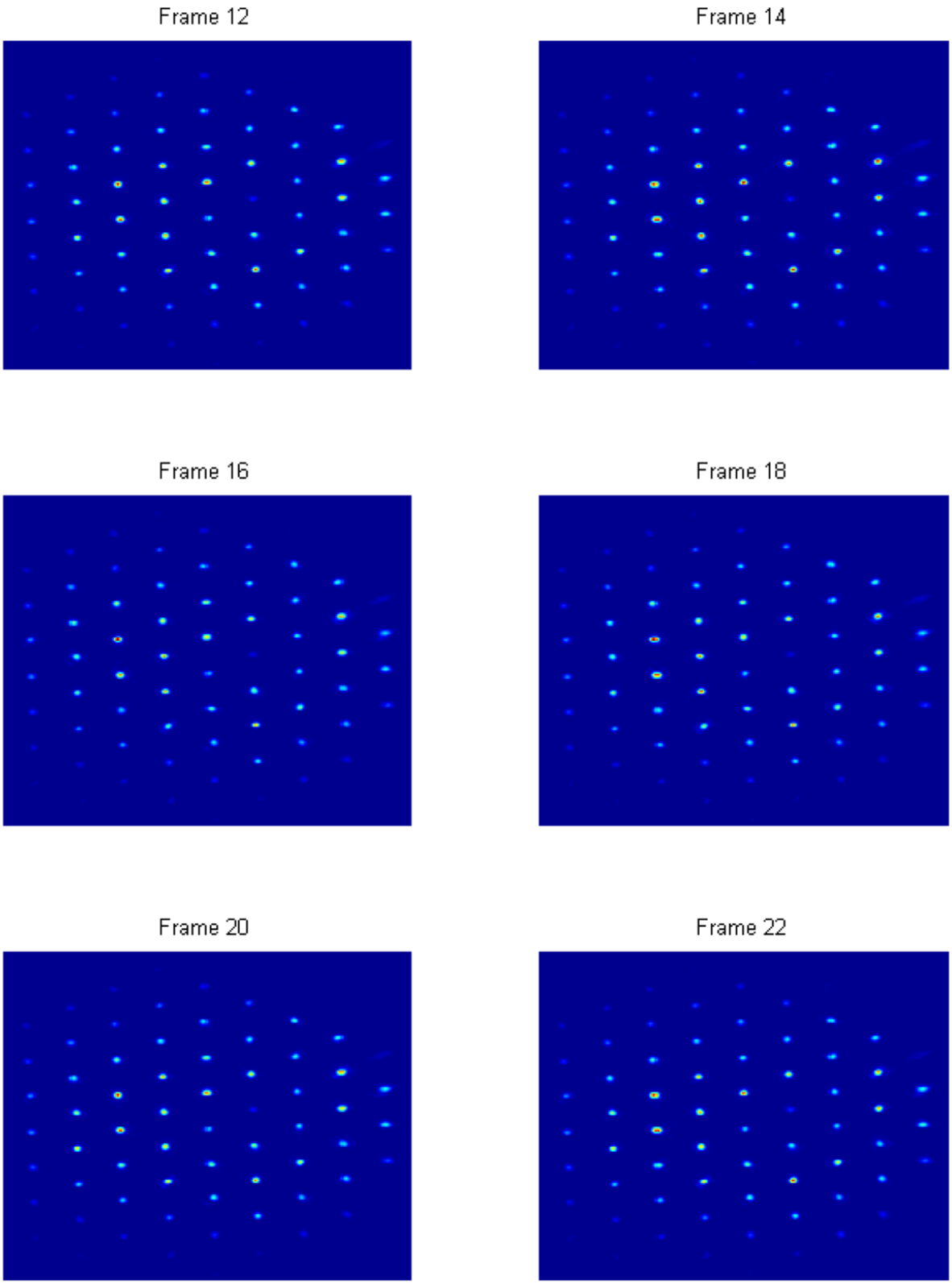

Figure 47. WFS images in long path, no SLM aberrations

With intensity fluctuations demonstrated in the long path due to ambient atmospheric conditions only, it is expected that propagating atmosphere from SLM 1 through the long path will yield the desired intensity fluctuation and dropout behavior for simulating the effects of deep turbulence. While the long path effects are expected to slightly increase the variation of wavefront sensor error in the presence of atmosphere applied on SLM 2, it is still expected that the wavefront sensor will be able to detect the phase aberrations fairly well, since the stronger applied atmosphere will not have propagated the long distance.

D. INTENSITY DROPOUT DETECTION

Once the effects of intensity fluctuations and dropouts in the long path are observed visually, a rudimentary detection method is developed in software and implemented in the WFS centroiding algorithm. Using the 8-bit Basler camera with low noise, the Shack-Hartmann WFS subapertures sense only the lenslet spots themselves above a zero threshold. If a spot drops out completely, the subaperture contains only zero-valued pixels. This work found that it is possible to use this information both as an indication of the presence of an intensity dropout, and to assign a placeholder value for the missing centroid.

1. Dropout Detection with Beam Blocking

The centroiding algorithm, which determines the lenslet spot offsets from the reference centroids, is modified to detect dropouts as follows. If a WFS subaperture contains only zero-valued pixels, the algorithm returns an indication of an intensity dropout, a (1). If there is no dropout, the algorithm returns a (0). In the former case in the absence of a centroid reading, the centroid is artificially set to return the centroid position as the bottom right corner of the subaperture. This is done to induce high slope error, since the reference centroids from which the offsets are determined are close to the center of the subaperture. While this method of artificially assigning a centroid location is understood to produce some inaccurate error results, it is implemented for simplicity and the visualization of error trends in the disturbance profiles. Once it has been determined that dropouts occur, and to what extent they occur in the short and long paths,

more appropriate methods of estimating centroid locations or trajectories in case of a dropout can be developed. To demonstrate the outcome of this simple detection method, the beam is blocked manually by an index card. Figure 48 shows the long path reference image overlaid with reference grid locations. The subapertures containing intensity dropouts during partial obscuration of the beam are marked by blue stars. The dropouts are detected as expected.

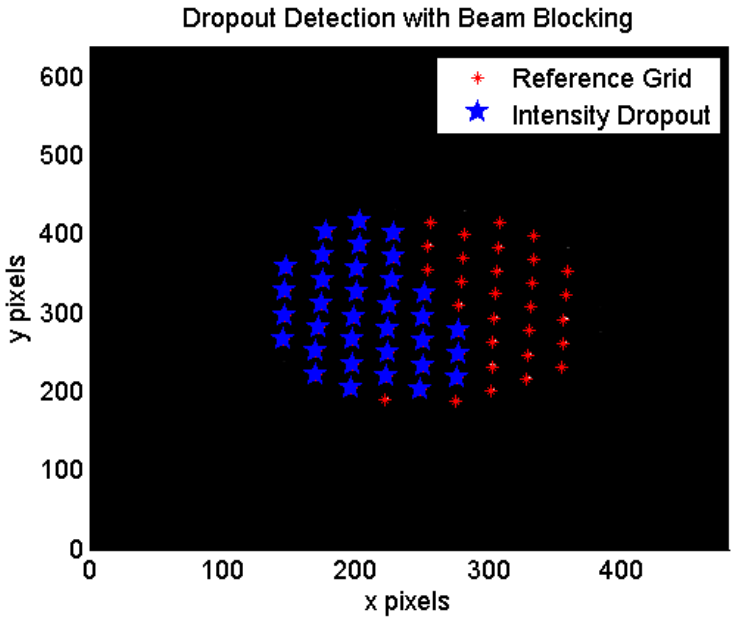

Figure 48. Intensity dropouts caused by manually blocking laser beam

2. Dropout Detection with Atmosphere

With dropout detection in place, atmospheric data is collected in 20-second intervals in both the short and long paths to determine the number of intensity dropouts in each case and the trends followed by the disturbance data. Four rates of the same atmospheric realization are recorded. The atmospheric realization is run at 1 Hz, 3 Hz, 5 Hz, and 7.5 Hz on SLM 2 alone, SLM 1 alone, and on both SLMs together. Table 5 shows the number of dropouts in each case, and the percentage of total possible dropouts, which is the number of lenslets (60 in short path, 46 in long path) times the number of frames (300). While the data could be improved by running several more iterations and averaging the results, the overall trends in the dropouts followed expected patterns. No

dropouts were detected with atmosphere applied on SLM 2 in either the short or long path. While the WFS images showed fluctuation in the long path SLM 2 case even without atmosphere, these fluctuations are expected to increase variation in the error data if not to induce actual dropouts in the SLM 2 long path case. As expected, the error from applying atmosphere upstream at SLM 1 increases somewhat in the short path and dramatically in the long path compared to atmosphere applied in the pupil plane at SLM 2. Dropouts due to atmosphere on SLM 1 and on both SLMs represent about 0.5–1% in the short path and 5–10% in the long path.

Table 5. Intensity Dropouts in Short and Long Paths

		Short Path		Long Path	
		# Dropouts	% (of 18000)	# Dropouts	% (of 13800)
SLM 2	1 Hz	0	0	0	0
	3 Hz	0	0	0	0
	5 Hz	0	0	0	0
	7.5 Hz	0	0	0	0
SLM 1	1 Hz	208	1.16	804	5.83
	3 Hz	20	0.11	761	5.51
	5 Hz	27	0.15	1480	10.7
	7.5 Hz	15	0.08	1238	8.97
Both	1 Hz	317	1.76	865	6.27
	3 Hz	97	0.54	758	5.49
	5 Hz	42	0.23	1498	10.9
	7.5 Hz	299	1.66	1244	9.01

Figure 49 shows the disturbance profiles on SLM 2 in the short path, while Figure 50 shows them in the long path. The different profiles represent the different rates at which atmosphere was applied on the SLMs. No control was applied, so error minimization is not expected. This data is shown to compare disturbance profile trends only. The error profiles between SLM 2 in the short path and SLM 2 in the long path are comparable as expected, with slightly more variation in the long path disturbance data, as well as more noise on the disturbance profiles. Some differences in the timing or the amount of atmospheric realization that passes in the same 20 seconds of data collection in various cases are due to imperfect timing control using Simulink's hardware interface.

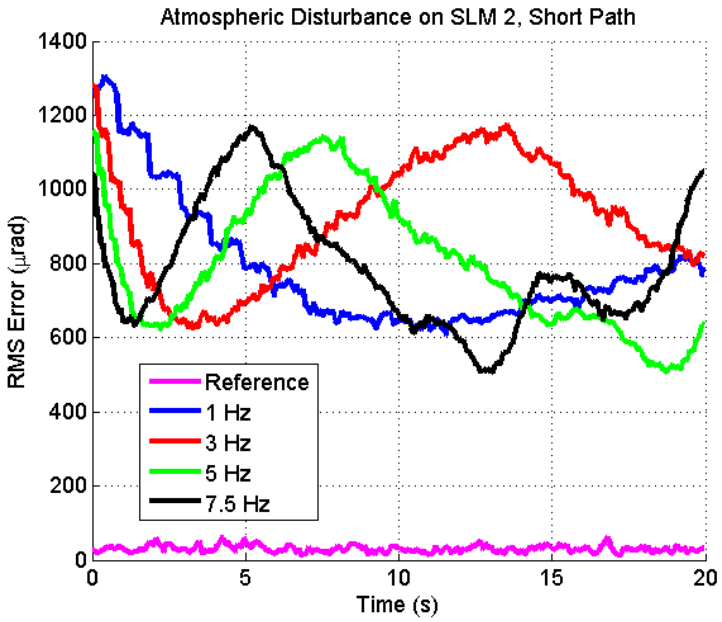

Figure 49. Atmospheric disturbance at different rates on SLM2 in short path

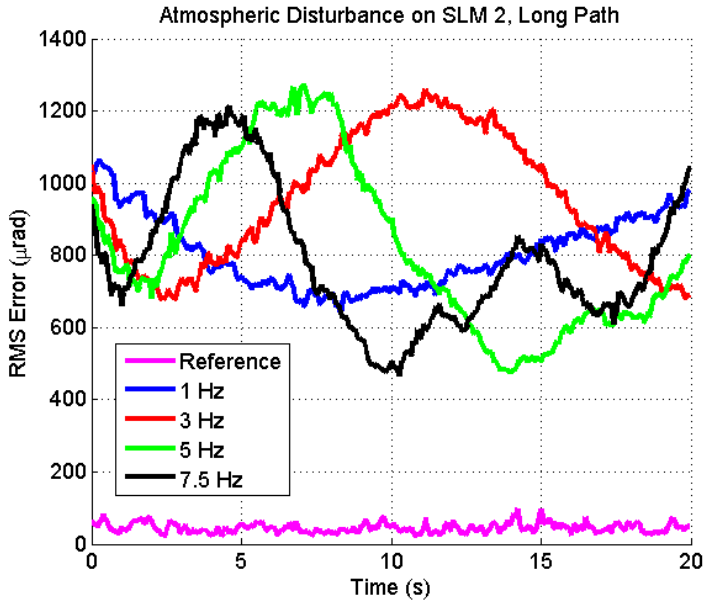

Figure 50. Atmospheric disturbance at different rates on SLM2 in long path

Figure 51 shows the results of atmosphere applied on SLM 1 in the short path. While some dropouts are present in the first few seconds, the disturbance profiles are very similar to those on SLM 2; however, the overall error is slightly higher as expected for the out-of-pupil location.

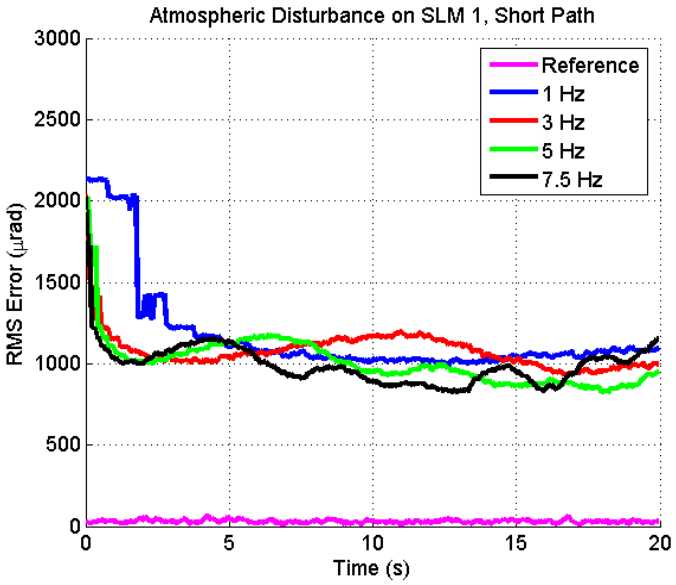

Figure 51. Atmospheric disturbance at different rates on SLM1 in short path

Figure 52 shows the atmosphere applied on SLM 1 in the long path. As expected in the significantly more challenging scenario, the disturbance profiles show much higher error than in the short path. The discontinuities are expected due to the simple dropout detection method. However, in the presence of so many dropouts, it is expected that even given more accurate centroid and slope data, the error will follow similar trends.

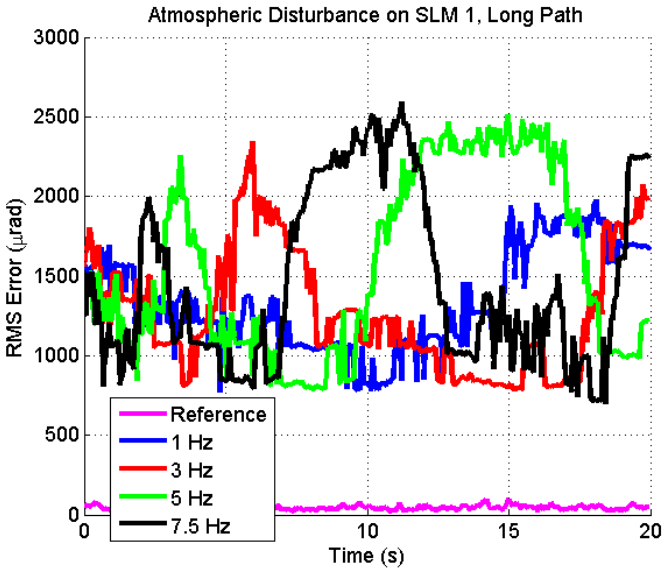

Figure 52. Atmospheric disturbance at different rates on SLM1 in long path

Figure 53 and Figure 54 show the disturbance profiles for atmosphere applied on both SLMs together in the short and long paths, respectively. In the short path, the error increases with the combination of atmospheres applied, with some dropouts in the 1 Hz and 7.5 Hz atmospheres beginning to demonstrate the effect of discontinuities with dropout detection. However, even in experiencing double the phase aberrations by traveling through both SLMs, the wavefront error does not reach the level of that shown in the long path. The error data in the long path for atmosphere on both SLMs appears to be very similar to that shown in Figure 52 for atmosphere only on SLM 1, indicating that the error is dominated by the effects from SLM 1 propagation. In the short path, the difference between SLM 1 error and error from both SLMs is more noticeable since the errors from each SLM are of similar magnitudes. The overall trends demonstrated by the disturbance profiles agree with the wavefront sensor image information in indicating that the effects of deep turbulence are indeed present in the long path testbed configuration.

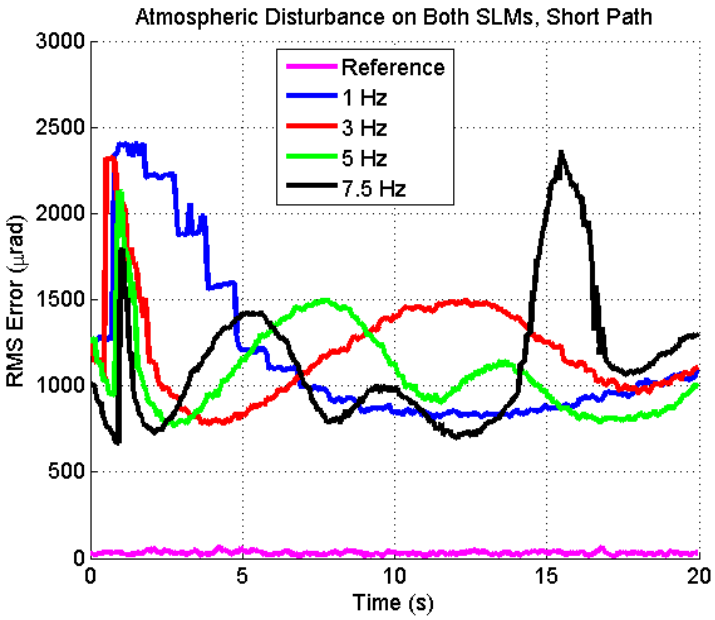

Figure 53. Atmospheric disturbance at different rates on both SLMs in short path Discontinuities from dropouts start to appear and affect slope error.

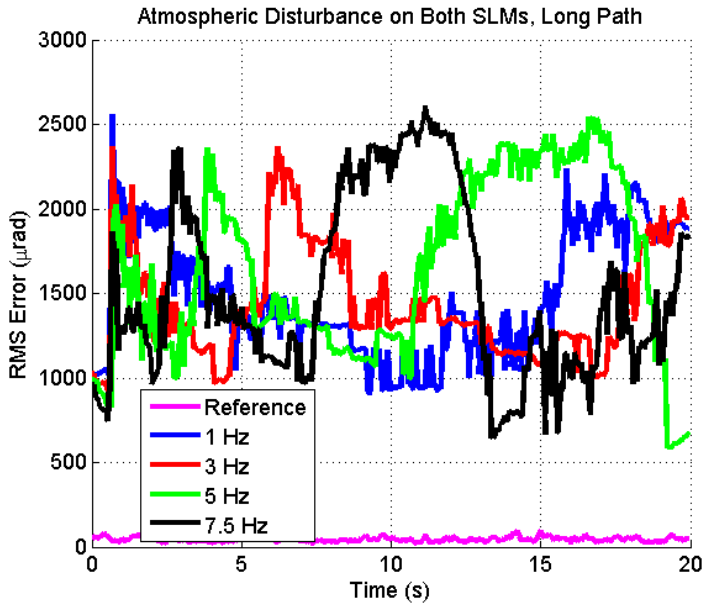

Figure 54. Atmospheric disturbance at different rates on both SLMs in long path Error is similar to SLM 1 alone as its effects dominate.

E. BEAM CONTROL IN DEEP TURBULENCE

Chapter IV results from short path with atmosphere applid on SLM 2 showed that the LMS transverse, RLS lattice, and Kalman filters performed comparably in augmenting the PI controller except for the lattice filter's long learning period the Kalman filter's use of a disturbance model. To draw some preliminary conclusions on control algorithm performance in deep turbulence scenarios, the algorithms evaluated in this research are applied to the atmospheric disturbance profiles generated from SLM 1 in the long path. It is important to note that the artificial assignment of missing centroids to the bottom right subaperture corner leads to poor slope calculations in the presence of dropouts, creating jumps or discontinuities in centroid motion. While the goal of simulating deep turbulence is indeed to introduce the challenge of discontinuities, the actual trajectories of fading centroids are not estimated here. Extrapolation or other fitting methods to project or estimate centroid locations in dropout cases would yield more accurate error data. However, with the understanding that control performance, especially in the long path disturbance profiles presented here, will not be as realistic as possible, testing the algorithms in this scenario should give insight into the trends expected in the presence of more realistic disturbance data.

In the extremely challenging disturbance profiles generated from SLM 1 in the long path, the RLS Lattice filter is tested individually to determine if its performance can be improved. The discontinuous nature of the disturbance provides a particular challenge to RLS convergence, since the algorithm uses the past data in its weight estimation process. The algorithm effectively starts its calculations over when it encounters a discontinuity, yielding a longer convergence time than normal. Figure 55 shows the variation of RLS forgetting factor, λ, in the presence of deep turbulence. None of the settings allows the RLS lattice to learn in the course of the simulation time, but the lowest forgetting factor of $\lambda = 0.1$ leads to the lowest error peaks. The lower forgetting factor allows older data to be weighted less in parameter estimation, so that the most recent data dominates the weight update. In the deep turbulence disturbance, this means that fewer discontinuities are contained in the past data, and the algorithm can attempt to determine optimal weights with a shorter history and greater chance of success.

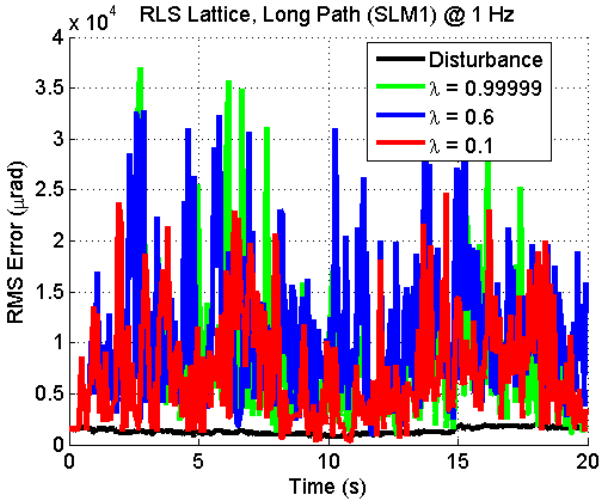

Figure 55. Parameter selection for RLS lattice in deep turbulence

Figure 56 shows LMS and RLS results for each rate of the atmospheric realization. All provide a challenge to the RLS convergence as described.

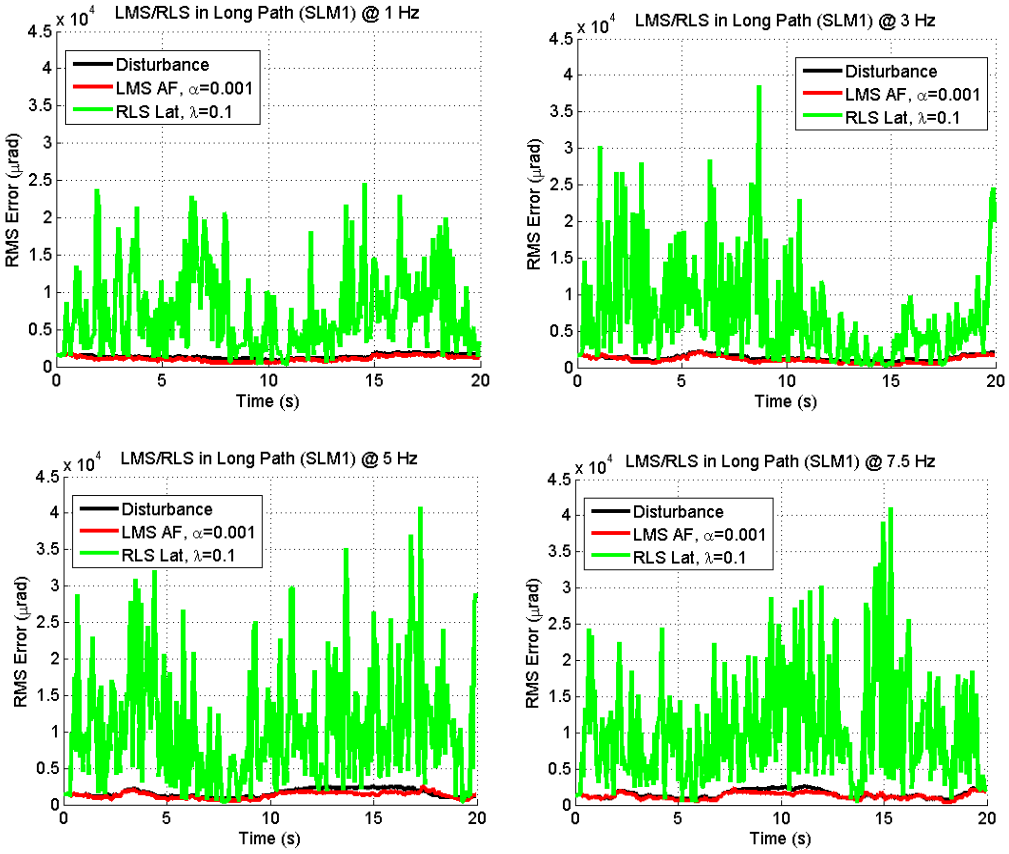

Figure 56. LMS and RLS performance for 1 Hz, 3 Hz, 5 Hz, 7.5 Hz atmospheres

Figure 57 shows PI, LMS AF, and Kalman filter performance in the long path with atmosphere on SLM 1 at each atmospheric rate. While none of the algorithms performs very well, they are all able to keep the error below the level of the disturbance. As before, it appears that the LMS AF has some control ability while it converges, unlike the RLS algorithm.

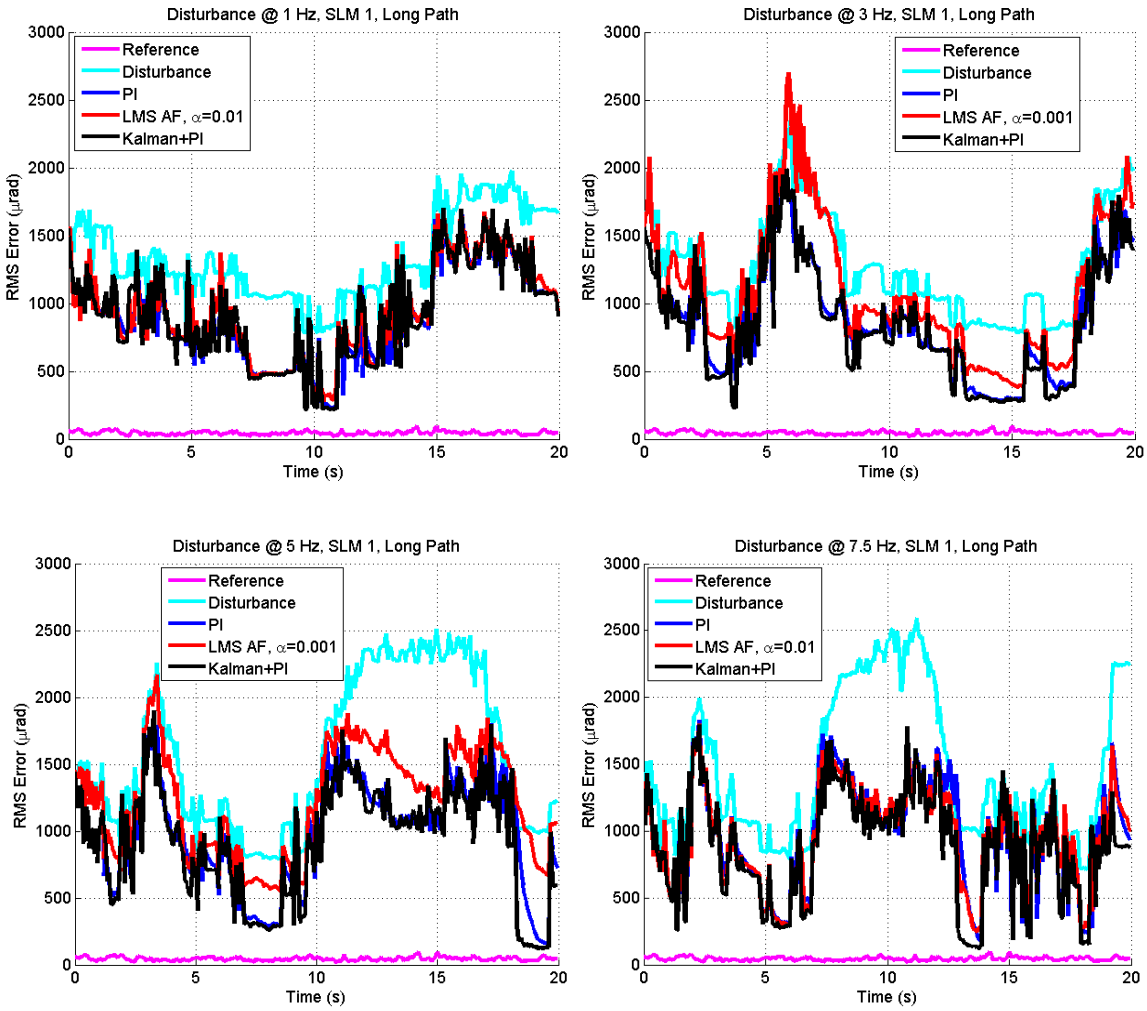

Figure 57. PI, LMS AF, and Kalman filter performance in long path disturbances (Top Left) 1 Hz with LMS AF α=0.01, (Top Right) 3 Hz with LMS AF α=0.001, (Bottom Left) 5 Hz with LMS AF α=0.001, (Bottom Right) 7.5 Hz with LMS AF α=0.01

It was necessary to modify the LMS AF convergence coefficient, α, to improve performance in two of the atmospheric cases. The value of $\alpha = 0.01$, which was used in all previous control results for the LMS AF working alone, proved too large for the 3 Hz

and 5 Hz long path atmosphere cases. This convergence rate showed some unstable performance with high RMS error and control effort. The value was reduced to $\alpha = 0.001$ for these cases, yielding stable yet somewhat degraded performance in error rejection. However, for the 1 Hz and 7.5 Hz atmosphere rates, the standard value of $\alpha = 0.01$ was able to provide stable control. This indicates that care must be taken in a real scenario to determine the appropriate convergence coefficient.

In the cases when the LMS AF can control with the convergence rate used in all other comparisons, it performs very similarly to the PI and Kalman + PI algorithms. The predictive algorithms do not provide additional benefit over the PI in this case since the poor slope reconstruction with discontinuities makes accurate prediction difficult. Results for combining the LMS AF and PI are not shown, as the addition of the LMS AF provided no additional benefit over the PI alone. The Kalman filter, while it also makes recursive use of previous data, does not suffer as much as the RLS algorithm since it is once again using a disturbance model based on the input disturbance itself in its prediction.

Performance comparisons indicate that while correction is not idea, the LMS AF algorithm can tolerate large errors from poor wavefront estimation better than the RLS algorithm. The complexity of the RLS algorithm, which can improve its performance in some cases, prevents it from tackling the deep turbulence problem as well as the simple, robust transverse LMS filter.

The results shown in this chapter indicate that the current flexible laboratory setup can create the effects of scintillation and branch points that are commonly associated with deep turbulence. When models become available that include the additional effects of maritime temperature and humidity factors, as well as aerosol effects, these models can be implemented on the SLMs to produce an even more realistic maritime environment. Additionally, with more sophisticated intensity dropout compensation, beam control performance in this challenging environment can be further explored. For the simple system and algorithms developed in this research, the LMS AF shows more robust performance in deep turbulence effects than the complex RLS algorithm.

VI. CONCLUSION

This chapter summarizes the results of this research and draws conclusions regarding the control algorithms tested. Areas for further research are described, and a final summary of contributions is provided.

A. RESULTS AND CONCLUSIONS

As stated in Chapter I.G, the goal of this research was to simplify adaptive optics control for Navy systems in shipboard applications. In order to do this, beam control algorithms were implemented and compared in simulation, and an adptive optics testbed was developed to compare the algorithms in a turbulence-like scenario. The testbed was also built with the goal of simulating a maritime atmospheric environment.

A classical adaptive optics control algorithm was augmented in three ways: with a transverse Least Mean Square adaptive filter, a lattice Recursive Least Squares adaptive filter, and a Kalman filter. In simulation with a generic sinusoidal disturbance, the LMS and RLS adaptive filters performed with comparable steady state error, while the RLS parameters converged faster. This was expected due to the RLS filter's use of past data in its weight estimation. The RLS algorithm converged in less than one second, while the LMS took approximately three seconds. However, the LMS algorithm reduced the error to steady state almost as quickly as the RLS algorithm. An important factor in analyzing RLS success is that the model provided does not control at all until it has converged. The LMS algorithm, on the other hand, is designed to control as it learns and converges. As a result, convergence comparisons and control performance comparisons between these two algorithms do not always yield the same conclusions.

With the testbed disturbance, the difference in convergence and control performance became more apparent than in the sinusoidal disturbance case. The RLS algorithm took over half the simulation time, approximately 11 seconds, to converge and begin controlling in the presence of the testbed disturbance. This longer convergence time was likely due to the non-periodic nature of the atmospheric profile. It is also possible that the algorithm requires a higher sample rate as compared to the rate of the

disturbance. This can be solved in the future with faster hardware and improved timing control. Once the algorithm did converge, its control performance was very good. The weights for the LMS algorithm showed some plateaus indicating intermediate levels of convergence, but did not reach final steady state values during the simulation time. This indicates that as expected, the RLS algorithm converged faster than the LMS algorithm. However, despite a longer learning period, the LMS algorithm reached a low level of RMS error almost immediately after being turned on. This performance trend was very similar to that in the presence of the sinusoidal disturbance, with more fluctuation in the steady state error due to the non-periodic disturbance. As a result, while the algorithm may have taken longer to converge and achieved a slightly higher average wavefront slope error , the immediate control ability and comparable error performance of the LMS algorithm make it the better controller in this testbed disturbance.

While a Kalman filter is structurally very similarly to an RLS adaptive filter, the Kalman filter did not suffer from the same performance issues because it estimated its gain based on both the disturbance measurement and a model of the disturbance, which the RLS did not have. However, the additional algorithm complexity and the necessity of prior or online model estimation for a non-periodic atmospheric disturbance again make the simplicity of the LMS algorithm more attractive.

While using hardware control in the testbed itself, the LMS adaptive filter was the only PI augmentation that consistently and successfully worked to minimize slope error within the current hardware and software limitations. As described above for testbed simulations with the testbed-generated disturbance, only a slight improvement in wavefront slope control was achieved using either the RLS or Kalman filters in place of the LMS filter in augmenting the PI loop. As a result, the simple, robust LMS adaptive filter is preferred for controlling the atmospheric disturbance generated in the laboratory.

The effects of deep turbulence existing in a maritime environment were generated in the testbed by extending the beam path between the spatial light modulator phase screens where atmospheric turbulence profiles were applied. This provided a foundation for the first critical elements of a maritime environment to be simulated. Intensity

dropouts were accounted for by assigning missing centroids to an artificial subaperture corner location. This provided discontinuities in the wavefront slope reconstruction and challenged the control performance of each algorithm.

All algorithms showed degraded control in the deep turbulence environment, but the LMS adaptive filter algorithm was robust enough to control to some degree, unlike the more complex RLS algorithm. Once again this was likely related to a difference between convergence and control performance. The standard, short path testbed results showed that the RLS algorithm took longer to reach optimal weights for a non-periodic but smoothly transitioning disturbance than for a sinusoidal and smoothly transitioning disturbance. However, it was still able to determine weights and produce satisfactory control performance for the non-periodic disturbance. In the deep turbulence scenario with intensity dropouts, the disturbance no longer transitioned smoothly. The significant number of discontinuities in the disturbance profile made it extremely difficult for the lattice RLS algorithm to estimate optimal weights from trends in historical data. It was shown that decreasing the RLS forgetting factor from 0.99999 to 0.1 slightly improved its performance, though the algorithm still did not converge in the time available. This indicates that simply including fewer historical discontinuities can help the algorithm as it estimates the weights, but the presence of the remaining discontinuities still interrupts those weight estimates and prolongs the convergence time of the algorithm.

In the deep turbulence scenario, the Kalman filter again used a disturbance model so that it did not encounter the same performance challenge as the lattice RLS algorithm. However, like the LMS adaptive filter and the classical PI controller, the Kalman filter demonstrated reduced performance in the presence of the discontinuous deep turbulence environment. All non-RLS algorithms performed very similarly in deep turbulence, barely controlling the error but able to keep it lower than the level of the disturbance. The Kalman filter is made robust from its use of a disturbance model in the estimation process, while the PI is robust in its feedback that is only based on the slope error and reconstructor matrix. The transverse LMS adaptive filter has a very simple structure and only attempts to predict one step ahead of the input error. These elements contribute to

more robust performance from the simpler algorithms than the lattice RLS filter in the current deep turbulence environment with poor wavefront slope reconstruction.

The conclusions presented here are drawn from the algorithms and adaptive optics testbed currently in use at the Naval Postgraduate School. Further developments, comparisons, and analysis can be done to draw more general conclusions on beam control performance in the maritime environment. Some of these recommendations are provided in section B. For the testbed and atmospheric disturbance profiles generated in this research, the simple, transverse LMS adaptive filter is chosen as the best overall controller to augment and improve the performance of a classical adaptive optics controller.

B. RECOMMENDATIONS FOR FUTURE WORK

A great deal of work remains to be done in the areas researched here. The simple LMS transverse filter has proven to perform very well for the small system developed in this research, especially as compared to the RLS filter in the deep turbulence scenario created in the laboratory. However, the RLS lattice filter has also been shown previously to perform well in large systems for simulations of Airborne Laser Test Bed engagements. Collaboration with Liu and Gibson will be useful to further investigate the factors surrounding LMS and RLS performance.

To improve timing control of hardware in the testbed, the control algorithms developed here can be implemented in Simulink's Real Time environment or in Labview. Additionally, the speed of the closed loop system could be improved by integrating a Roper Scientific camera to serve as the SH WFS. With a faster frame rate, faster disturbance data can be collected and tested for algorithm performance.

To produce more realistic atmospheric turbulence in the lab, SLMs with faster update rates can be investigated. SLMs running at 100 Hz or more are desirable. A fast enough WFS camera would become critical in this case.

To further investigate beam control performance in the deep turbulence scenario, a method of tracking centroid trajectories or extrapolating likely positions in the case of

dropouts should be developed. This will provide more accurate deep turbulence atmospheric profiles for more realistic control. Furthermore, investigation of the limits of aberrator plane separation for sufficient beam control should be performed.

With the many data collection efforts underway to improve statistical knowledge of turbulence in a maritime environment, new models will be developed to describe maritime turbulence. These models should be implemented on the current SLMs or future faster SLMs to combine near-surface effects with deep turbulence effects for a complete laboratory simulation of the maritime environment.

Finally, the current adaptive optics testbed will be integrated with the HEL jitter control testbed. The optics table containing both systems will be configured as a ship motion simulator. In this way, a comprehensive laboratory system will be available for testing beam control algorithms in a maritime environment.

C. SUMMARY OF CONTRIBUTIONS

This work constitutes three main contributions to the investigation of maritime adaptive optics beam control. First, this research implemented three advanced, multichannel control algorithms in Matlab's Simulink environment for use in the multiple-input multiple-output adaptive optics control problem. The control algorithms were compared in simulation, with a laboratory-generated disturbance, and in direct hardware control of an adaptive optics system.

Second, the Naval Postgraduate School's first laboratory system to use adaptive optics for the compensation of atmospheric turbulence was designed and built. The testbed was designed for flexibility in implementing a wide range of atmospheric realizations.

Third, the laboratory testbed was modified to create the deep turbulence effects of intensity fluctuations and dropouts. An intensity dropout detection and compensation method was developed and implemented, and it was discovered that deep turbulence effects were indeed present in the beam path extension in the laboratory. Algorithms were compared once more for performance in this environment.

Control and testbed results indicate that for the current system, adaptive optics beam control can be simplified successfully by the implementation of a simple, robust, transverse LMS adaptive filter to augment a classical controller. With the demonstrated capability of creating a deep turbulence scenario in the lab and the future development of atmospheric turbulence models that include additional maritime effects, the current testbed is ready for the next generation of adaptive optics beam control research and development.

APPENDIX A. SIMULINK MODELS

Appendix A shows the Simulink models developed for this research. Top level models for each algorithm are presented. Subsystems are shown in blue. For subsystems that are used in multiple models, only one instance of the subsystem is provided.

Figure 58 shows the top level model for the LMS adaptive filter as it is used with direct hardware control. It is implemented in actuator space so the error into the adaptive filter is of the actuator dimension. The switches with clock input are used to control the start times of the algorithms. The saturation block limits the hardware control signal to +/-1.

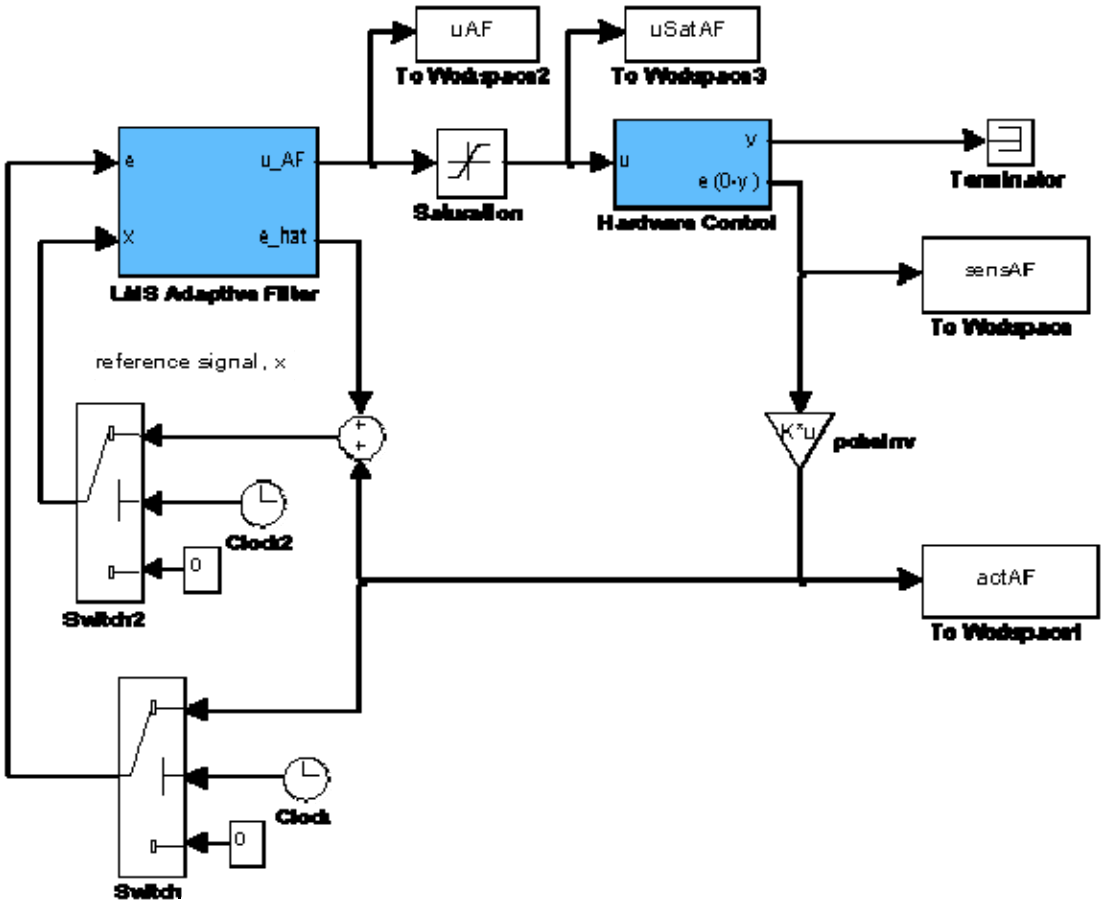

Figure 58. Hardware control model of LMS adaptive filter only

Figure 59 shows the LMS Adaptive Filter subsystem. The filter output passes through the secondary plant estimate to be used in generating the reference signal or disturbance estimate. In the actuator space model in the absence of the PI controller, a computational delay is the only element in the secondary path. One set of weights is selected for viewing the results.

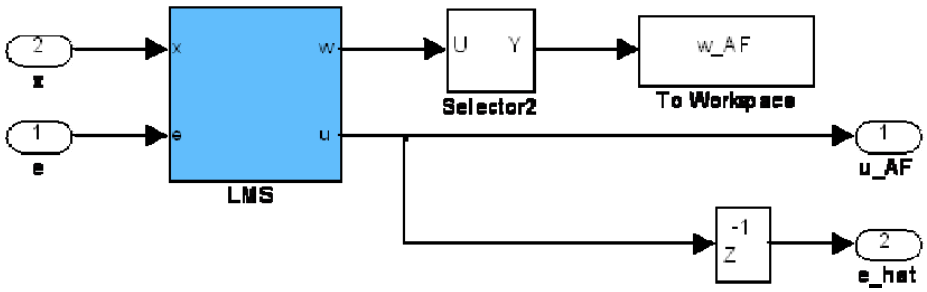

Figure 59. LMS Adaptive Filter subsystem with weight output and secondary path

Figure 60 shows the LMS subsystem with the coupled actuator space model of the adaptive filter. The filtered reference signal is used in the weight update, and the unfiltered reference signal is used in calculating the filter output.

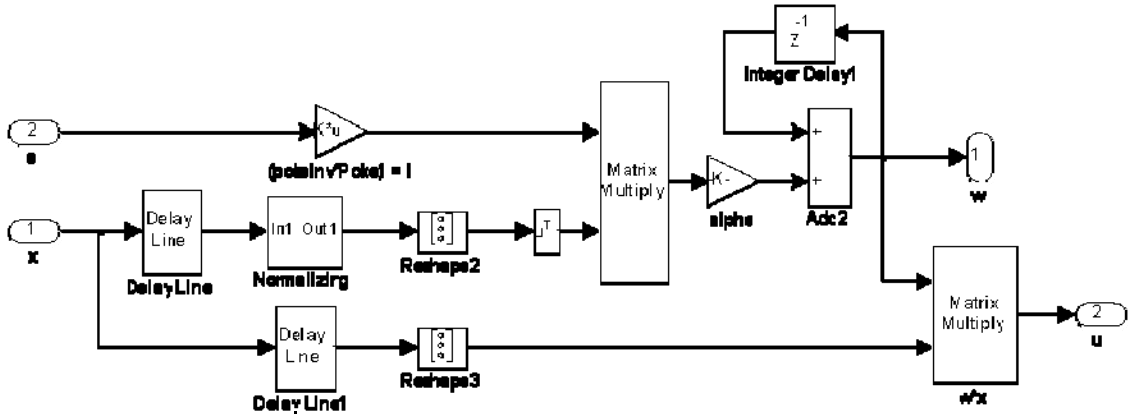

Figure 60. LMS subsystem showing weight updates and filter output calculations

Figure 61 shows the Hardware Control subsystem. The Shack-Hartmann WFS camera and the DM are controlled through a Level-2 M-file S-Function. The code for this function is shown in Appendix B. The outputs of the S-Function are the current

lenslet centroids and a vector of the current dropouts, if any. The reference centroids are subtracted from the current centroids and multiplied by the quanity $\frac{\text{pixel width}}{\text{focal length}}$ to generate the wavefront sensor slopes in radians. These quantities represent the pixel width of the WFS camera and the focal length of the SH lenslet array. The slopes are compared to a zero reference or desired signal to form the error to be used in the control algorithms.

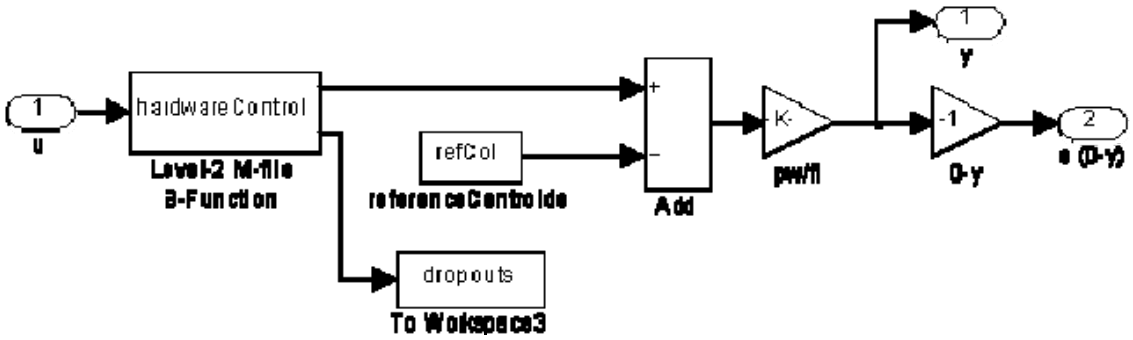

Figure 61. Hardware Control subsystem

Figure 62 shows the testbed model with the poke matrix simulation replacing the hardware control block in Figure 58. While a computational delay is automatically present in hardware control, that delay must be included in the simulation model as shown in the controller feedback path. Figure 63 shows the Simulation subsystem. In this subsystem, the user can select the disturbance source. The disturbance is either a generic sinusoidal disturbance or a testbed-generated disturbance. The testbed disturbance is previously recorded from applying atmospheric turbulence on the SLM(s) and recording the WFS response from a hardware control model with control turned off. This WFS response and its associated time vector are used as the disturbance input when a testbed-generated disturbance is desired.

Figure 64 shows the Simulated Disturbance subsystem used for the generic disturbance. It contains three sinusoidal components with different frequencies and phase values. It also contains white noise and the option to include a bias offset. For the results presented in this research, a bias offset is not used.

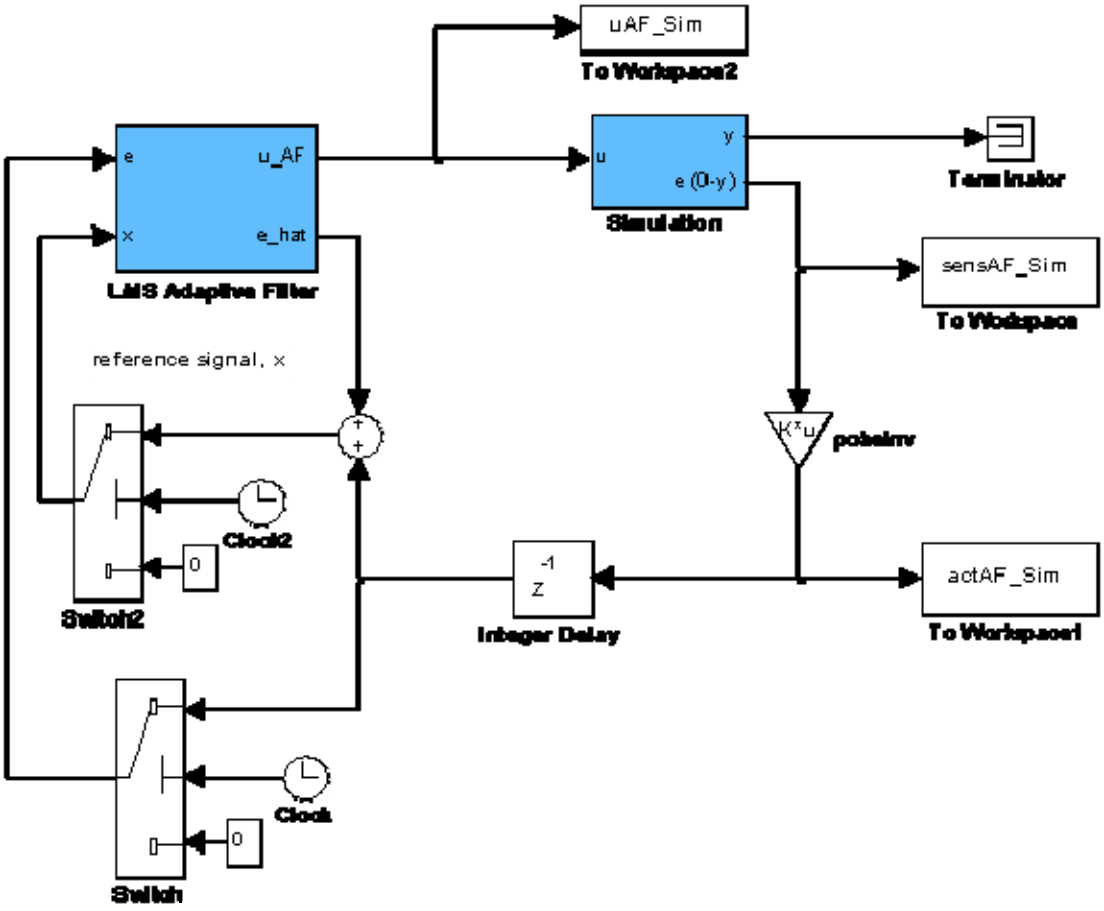

Figure 62. LMS AF model with poke simulation replacing hardware control

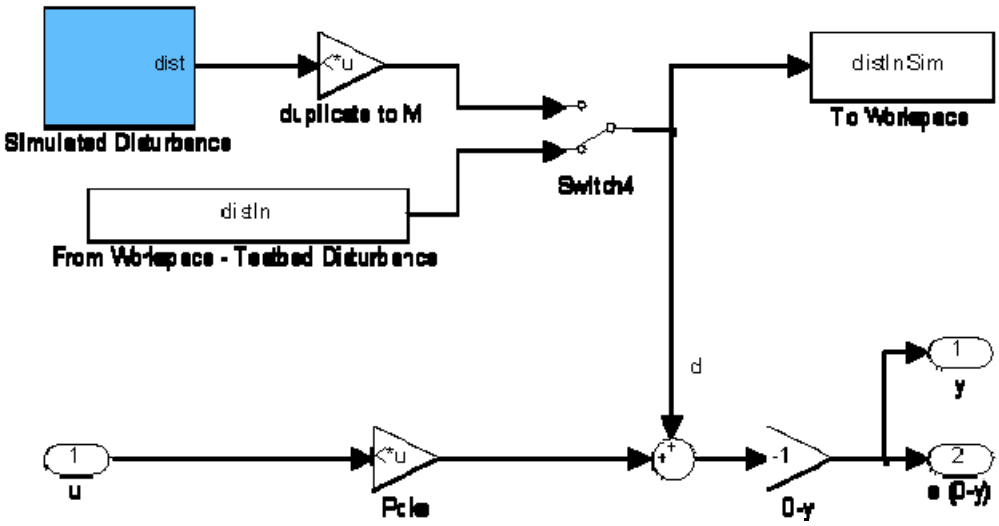

Figure 63. Simulation subsystem with simulated or testbed disturbance

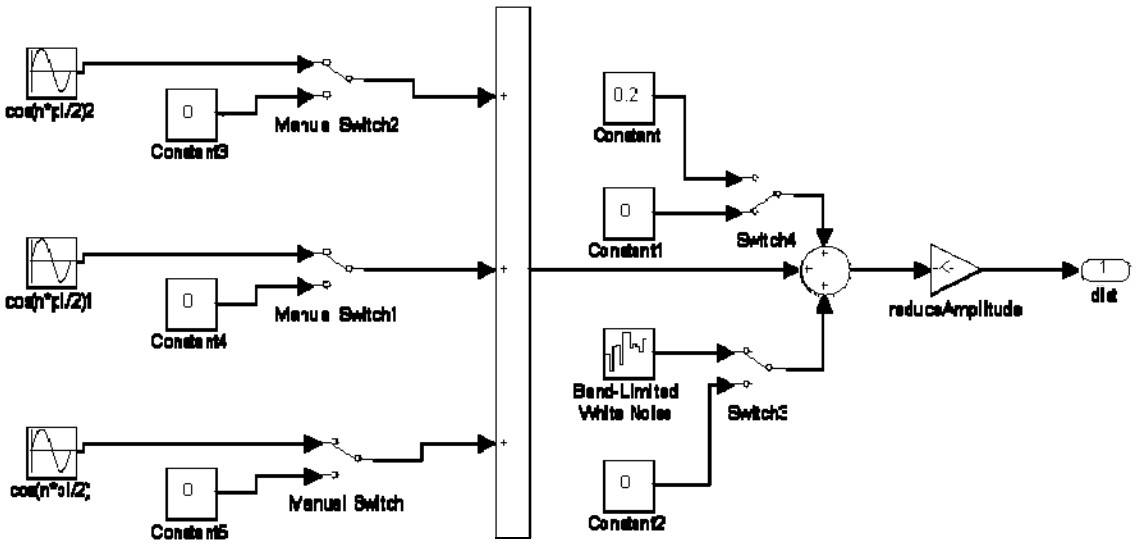

Figure 64. Simulated Disturbance subsystem with sinusoids and white noise

For the LMS adaptive filter working alone, a complete system with its associated subsystems has been shown for both the hardware control and simulation configurations. The following figures will present the remaining algorithms in the hardware control configuration, along with any new subsystems.

Figure 65 shows the hardware control model with the PI controller only. The PI subsystem is shown in Figure 66 and consists of a discrete integral controller and a proportional gain.

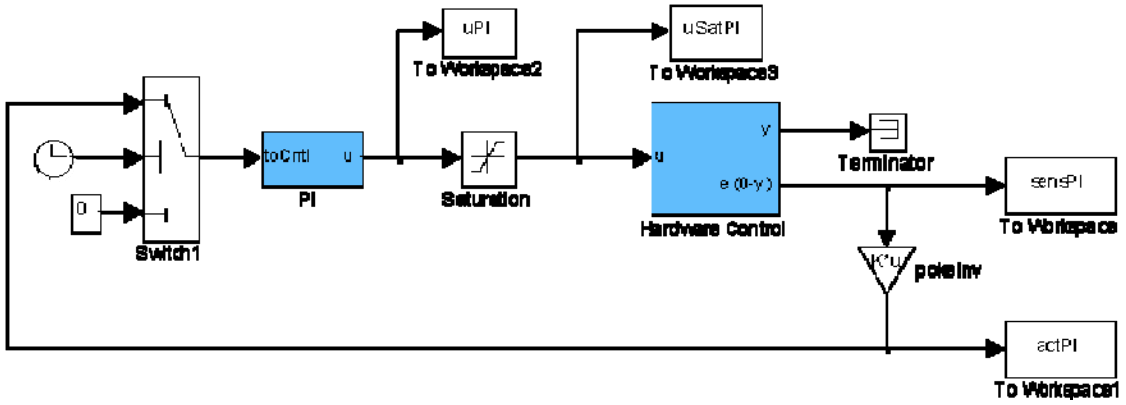

Figure 65. Hardware control model with PI controller only

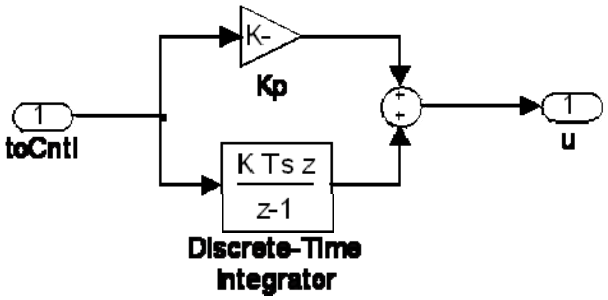

Figure 66. PI subsystem

Figure 67 shows the hardware control model of the LMS AF + PI controller. Figure 68 shows the LMS Adaptive Filter subsystem. Unlike in Figure 59 for the LMS AF alone, the closed PI loop is included in the secondary path estimate for the filter output to generate the internal reference. Figure 69 shows the LMS subsystem and the placement of the closed PI loop in the filtered reference path to be used in updating the adaptive weights. Figure 70 shows the PI Closed subsystem.

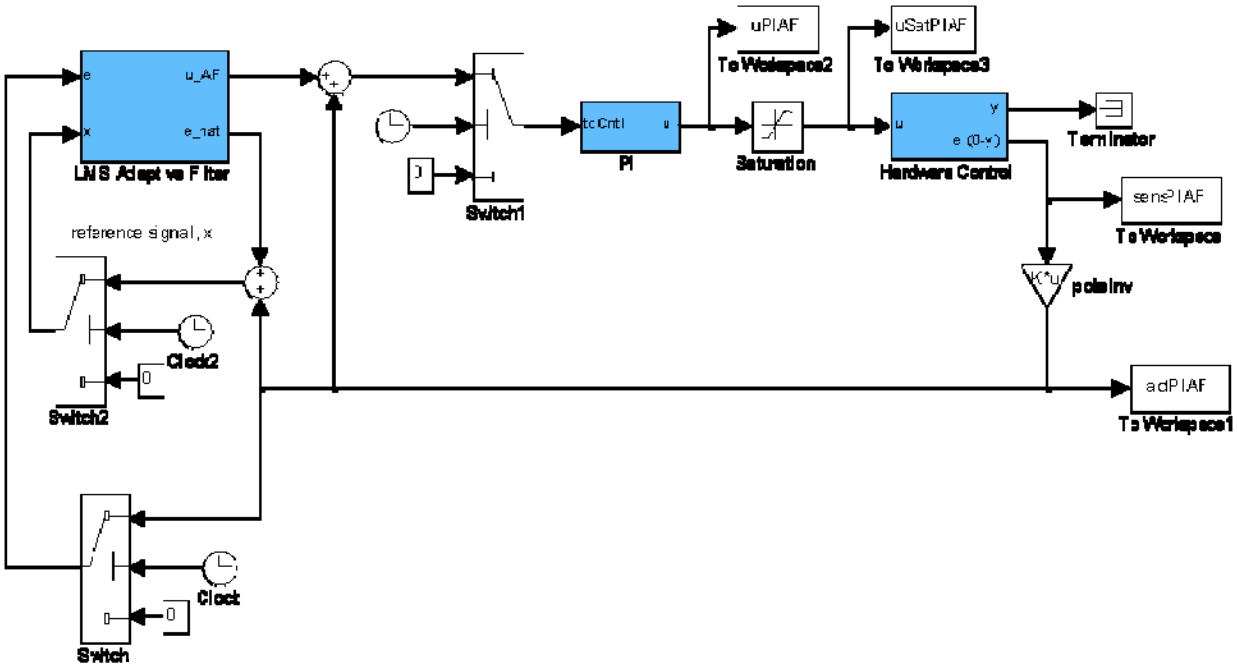

Figure 67. Hardware control model of LMS adaptive filter + PI

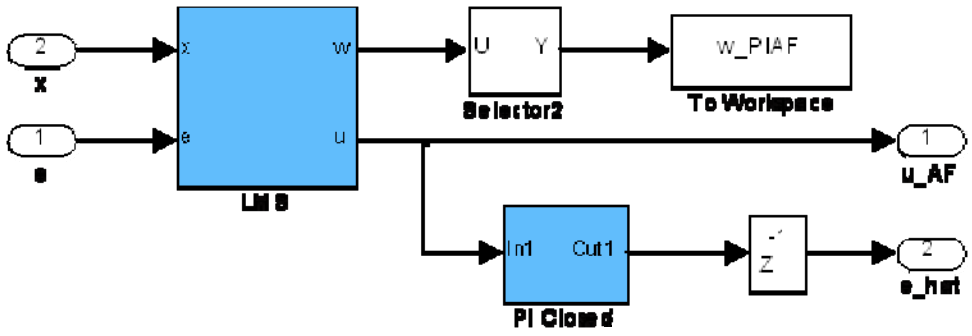

Figure 68. LMS Adaptive Filter subsystem showing closed loop PI in secondary path

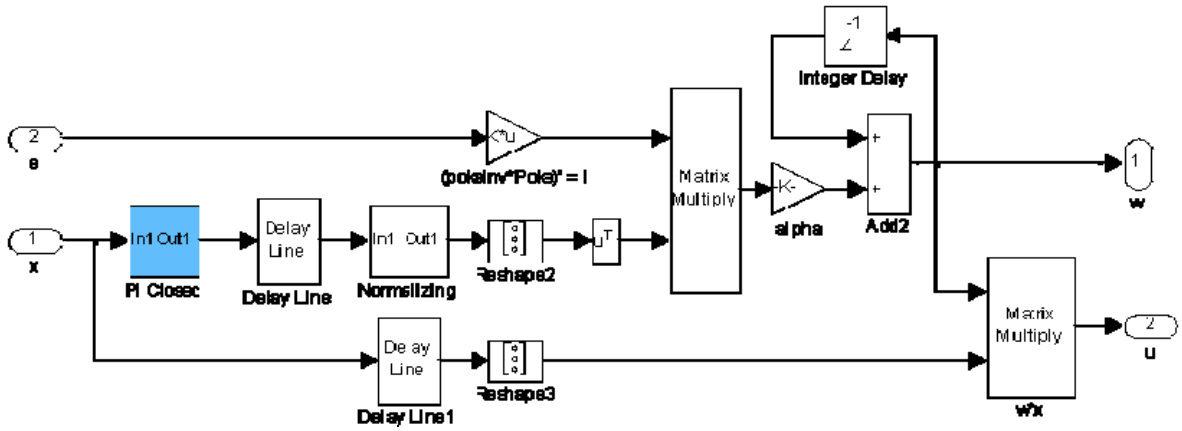

Figure 69. LMS subsystem showing closed loop PI filtering reference signal

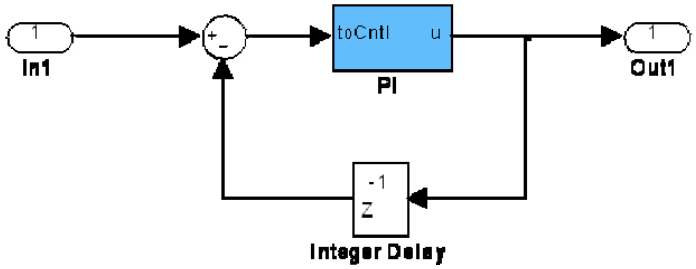

Figure 70. PI Closed subsystem

Figure 71 shows the hardware control model of the RLS lattice adaptive filter. Figure 72 shows the RLS Lattice AF subsystem on the left, where the green AOLat(z) component has been provided as a masked block by Gibson's UCLA research group. The right image in Figure 72 is the underlying structure of AOLat(z), where Aolat6 is an S-Function developed at UCLA. Figure 73 shows the hardware control model of the

RLS lattice adaptive filter with the PI controller. As shown in the LMS case, Figure 74 shows the RLS Lattice AF subsystem with the closed loop PI model in the secondary path for internal reference signal generation.

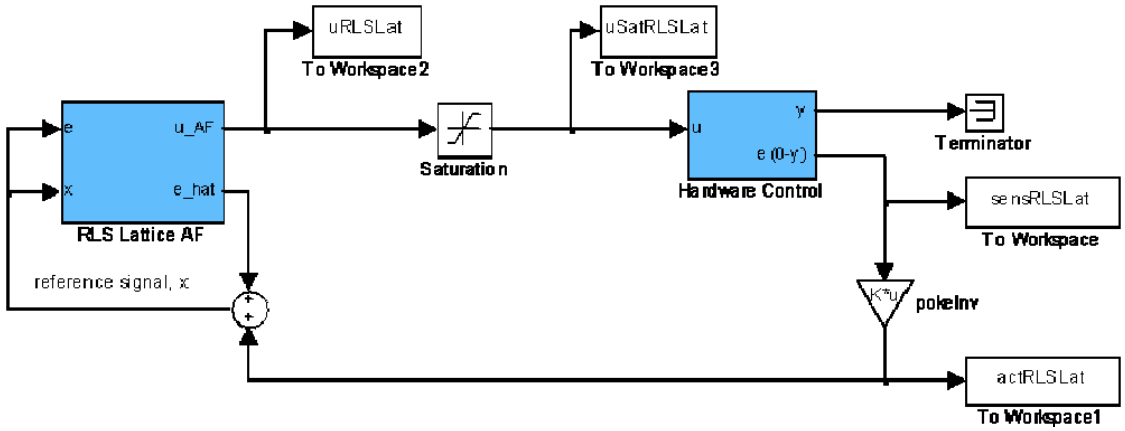

Figure 71. Hardware control model of RLS lattice adaptive filter

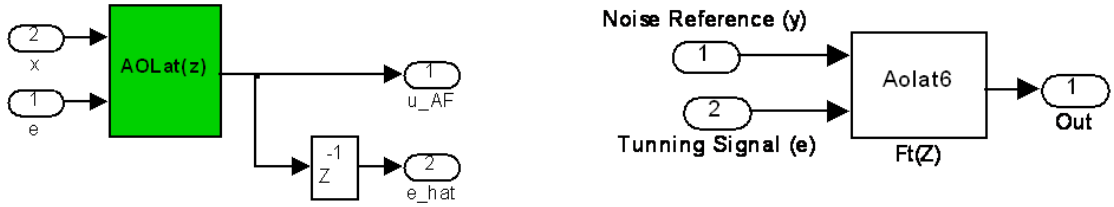

Figure 72. (Left) RLS Lattice AF subsystem (Right) Under AOLat(z) mask

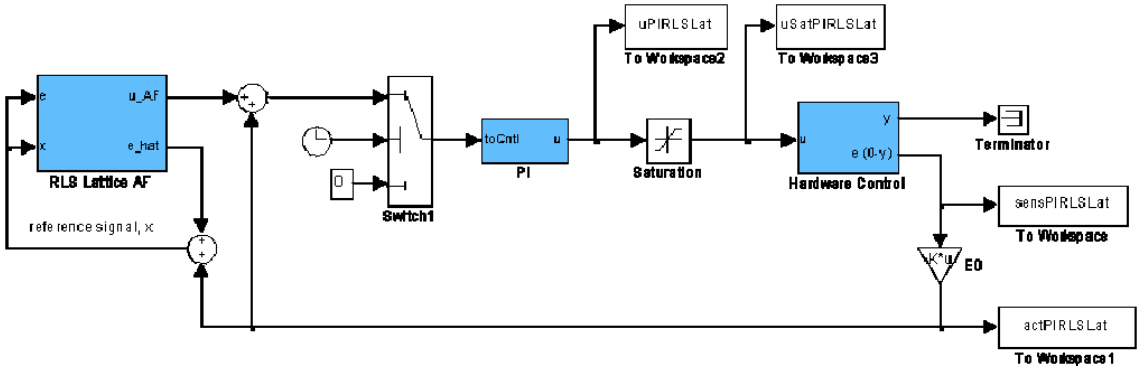

Figure 73. Hardware control model of RLS lattice adaptive filter + PI

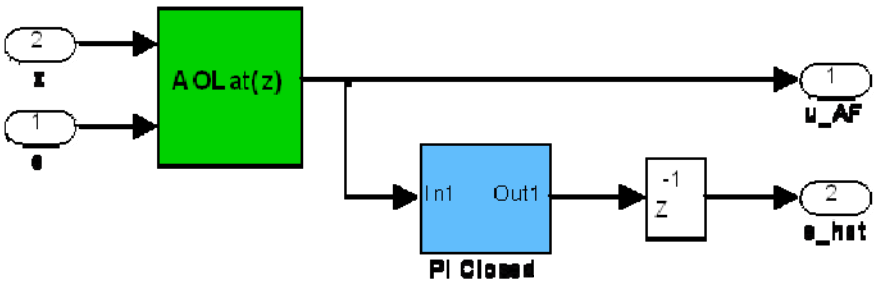

Figure 74. RLS Lattice AF subsystem showing closed loop PI in secondary path

The user-specified function parameters for the AOLat(z) block are shown in Figure 75. The parameters on the left are used for the RLS lattice working alone, while the parameters on the right are used for the RLS lattice + PI model.

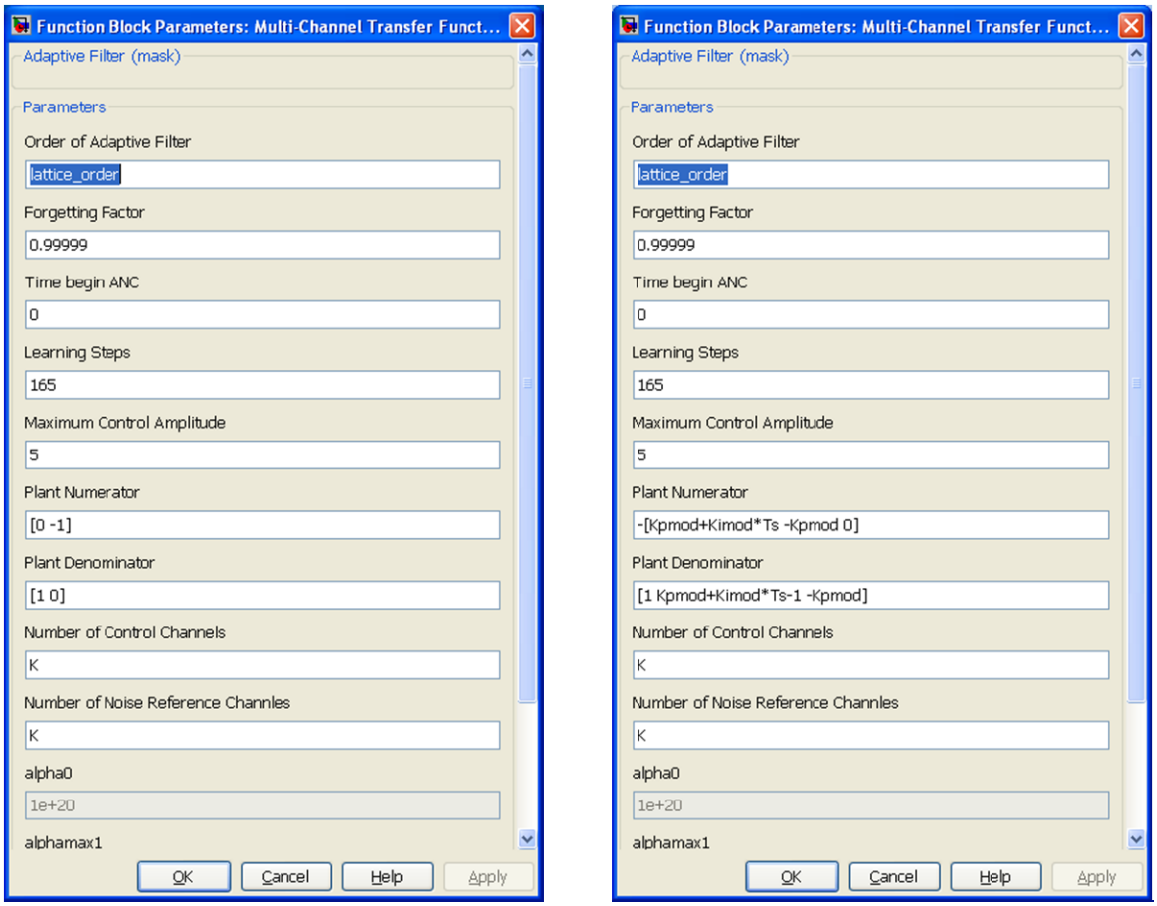

Figure 75. Dialog parameters used in AOLat(z) block
(Left) RLS lattice working alone (Right) RLS lattice + PI

Figure 76 shows the hardware control model of the Kalman filter + PI algorithm. The Kalman filter elements replace the adaptive filter elements from the previous models. Figure 77 shows the Kalman Filter subsystem, which uses the Kalman Filter block from Simulink's library. Figure 78 shows the Noise subsystem, which injects measurement noise into the wavefront sensor output.

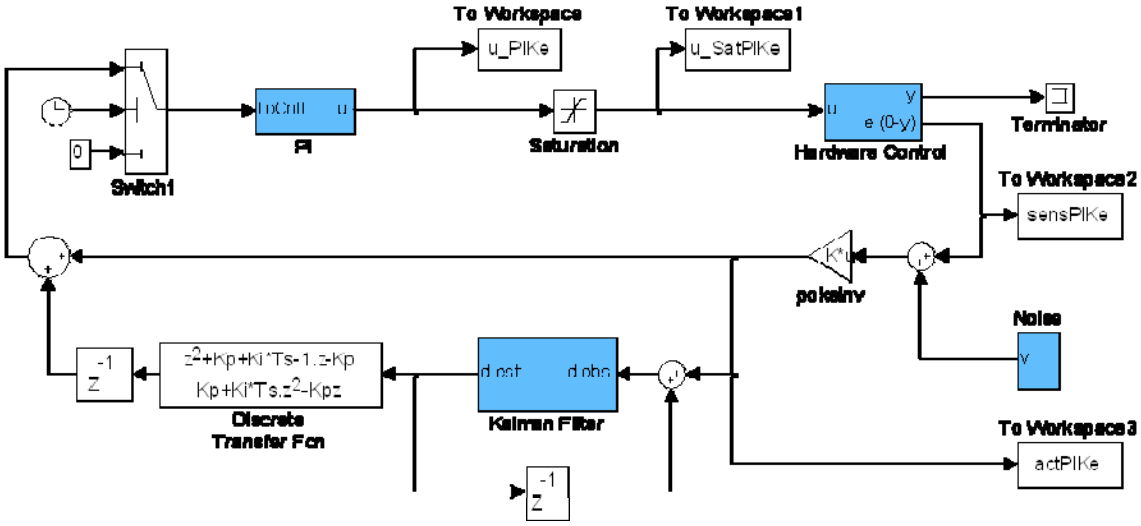

Figure 76. Hardware control model of Kalman Filter + PI

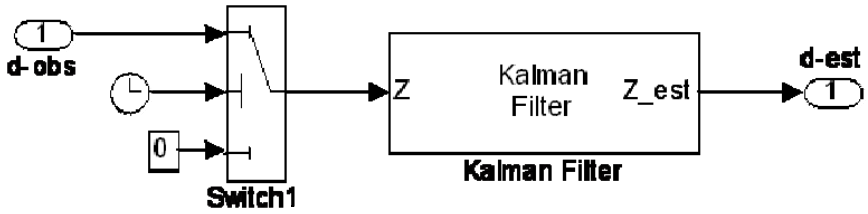

Figure 77. Kalman Filter subsystem using Simulink's KF block

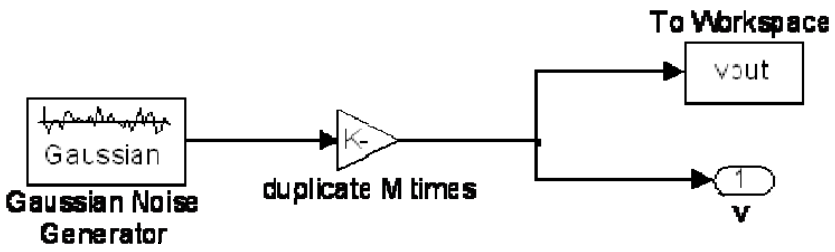

Figure 78. Noise subsystem showing addition of measurement noise

Figure 79 shows the function block parameters used for the Kalman Filter block. The state space model, *sysAll*, is generated as described in Chapter II.F. Matlab code for generating this model from a measured disturbance is shown in Appendix B.

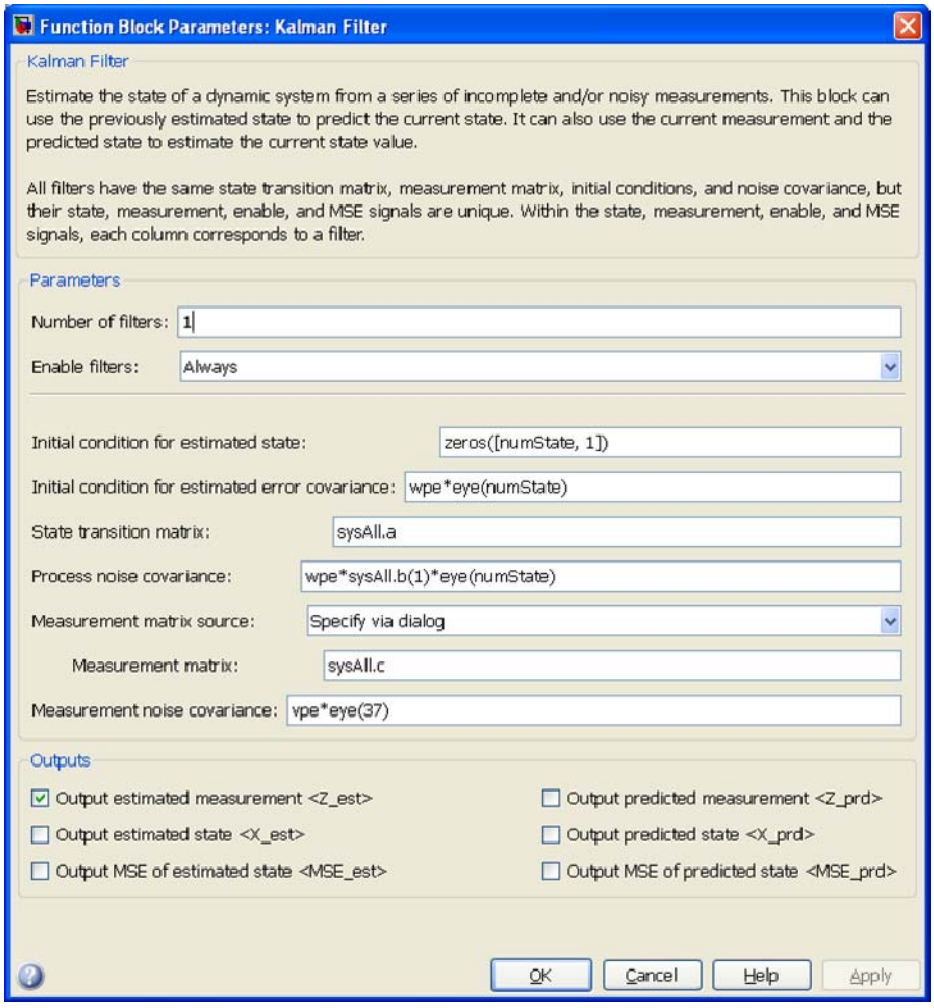

Figure 79. Dialog parameters used in Kalman Filter block

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MATLAB CODE

Appendix B shows selected Matlab code developed for this research. Code specific to the hardware interface is presented first, followed by code developed for the Simulink models presented in Appendix A.

A. POKE MATRIX

The code in this section is used to generate the system poke matrix prior to running the experiment. First, the workspace is initialized with several parameter values associated with the hardware.

```
% Constants, 37-ch DM and Basler camera
pw = 0.009630;          % pixel width in mm (Basler camera)
fl = 17.361408;         % focal length in mm (Hartmann mask, AO table)
n = 37;                 % number of DM actuators
ns = n;                 % Provides possibility of slaving in Simulink
cal = 50;               % threshold for Basler camera
Ts = 0.0667;            % sample time for discrete system
Vbias = sqrt(0.5)*255;  % bias voltage for 37-ch DM (quadratic law)
pwfl = pw/fl;
```

Using the SLM reference path, a reference image is taken. The reference centroids calculated from this image will be used in generating the poke matrix and in the control algorithms for calculating wavefront slopes. If a significant amount of ambient noise in the room is present, the reference image can be averaged from multiple frames. A single image is used here. The *BAOGrab* function is a .dll designed by Baker Adaptive Optics to work with the Basler camera. It has been used historically at the NPS for Matlab interfacing with the camera and is not shown here. The *findRefCent* function is modified from previously developed NPS code. The *sortRef* function is developed for this research and ensures proper ordering of the lenslets so that they are sequenced from left to right, top to bottom. It is not shown here. The *findRefCent* function and the *findCent* function, which is used for all subsequent image centroids, are shown in Section B.

```
% Capture reference image
BAOGrab(0);          % Initialize camera
a = (BAOGrab(1));    % Grab image
BAOGrab(2);          % De-initialize camera
```

```
% Find centroids and reference grid
[cent grid1] = findRefCent(a,cal);

% Sort so ordering is always left to right, top to bottom, same hexagon
[refGrid refCent numLens] = sortRef(grid1,cent);
refCol = reshape(refCent,2*numLens,1);
```

Once the reference image is captured, the reference beam is blocked and the primary path used. This path travels through the SLMs and the DM. As in the reference case, no atmosphere is applied on the SLMs at this point. The DM shape is initialized by applying the bias voltage to all actuators. The *BAODMirror* function, like the *BAOGrab* function, is a .dll developed by Baker Adaptive Optics for use with the OKO Technologies deformable mirrors. After initializing the poke matrix with the appropriate size, each actuator is poked in a loop that generates the full matrix.

```
% Set bias on DM for generating poke matrix
V = ones(1,n)*Vbias;
BAODMirror(hex2dec('6800'),hex2dec('6400'),V);

% Initialize poke matrix
poke = zeros(2*numLens,ns);
pokeSlopes = zeros(2*numLens,ns);

% Create poke matrix
for i = 1:ns
    V = ones(ns,1)*Vbias;       % set bias value on all actuators
    V(i) = 255;                 % apply max voltage to current actuator
    BAODMirror(hex2dec('6800'),hex2dec('6400'),V);
    pause(0.2)

    % Grab frame from current actuator response
    BAOGrab(0);
    currentAct = (BAOGrab(1));
    BAOGrab(2);
    SH = findCent(currentAct,refGrid,numLens);

    % Populate poke matrix for current actuator
    poke(:,i) = reshape(SH,2*numLens,1);          % in terms of
centroids
    pokeSlopes(:,i) = (poke(:,i)-refCol)*pw/fl;   % in terms of slopes
end

% Set DM values back to bias
V = ones(1,n)*Vbias;
BAODMirror(hex2dec('6800'),hex2dec('6400'),V);
```

B. CENTROID CALCULATIONS

This section presents the functions developed to calculate centroids in the reference and all subsequent images. The functions presented are: *findRefCent* and *findCent*, both of which are modified from previously developed NPS code.

1. findRefCent.m

```
function [refCent refGrid numLens] = findRefCent(a,cal)
% This function takes a Shack-Hartmann (SH) wavefront sensor (WFS)
image
% and calculates the initial centroid locations.
% INPUTS:
% a, the image (640 x 480)
% cal, the camera threshold value (for Basler, using 50 (of 255))
% OUTPUTS:
% refCent, the x and y reference centroid locations
% refGrid, the x and y reference grid locations (used as starting grid
for
% later centroid calculations)
% numLens, number of lenslets found above threshold (changes with beam
% size)
% ------------------------------------------------------------------
----

% Initialize centroid storage
refGrid = zeros(127,2);
refCent = zeros(127,2);

% Find size of image
size_a = size(a);
x = size_a(2);      % num columns = x
y = size_a(1);      % num rows = y

k = 1;              % to increment loop
n = 10;             % number of pixels before/after to black out for
Basler

% Search image for centroids
for i = 1:y
    for j = 1:x
         if a(i,j)>cal
            P = double(a(i-n:i+n,j-n:j+n));
            xi = j-n;                       % TOP left corner of
lenslet box
            yi = i-n;
            S = size(P); Sy = S(1); Sx = S(2);

            % Compute centroid
            M = sum(sum(P));                % find total "weight"
            X = (linspace(1,Sx,Sx))';       % vector of positions
```

```
            Y = linspace(1,Sy,Sy);
            Rx = P*X;                       % weighted x's (rows)
            Ry = Y*P;                       % weighted y's (columns)
            Rx = sum(Rx)*1/M;               % x centroid in box
            Ry = sum(Ry)*1/M;               % y centroid in box
            x_cent = xi+Rx-1;               % x centroid in original
image
            y_cent = yi+Ry-1;               % y centroid in original
image

            % Store location of centroid
            refCent(k,1) = x_cent;          % x location of centroid
overall
            refCent(k,2) = y_cent;          % y location of centroid
overall
            refGrid(k,1) = (j);             % x grid position (center
of box)
            refGrid(k,2) = (i);             % y grid position (center
of box)

            % Black out lenslet
            a(i-n:i+n,j-n:j+n) = zeros(Sy,Sx);
            k = k+1;
        end
    end
end

% Return only nonzero entries - account for only the lenslets which are
% selected
refCent = refCent(1:k-1,:);
refGrid = refGrid(1:k-1,:);
numLens = k-1;
```

2. findCent.m

```
function cent = findCent(a,refGrid,len)
% This function takes a Shack-Hartmann (SH) wavefront sensor (WFS)
image
% and calculates the centroid locations based on the reference grid
% produced from findRefCent.m
% INPUTS:
% a, the image (640 x 480)
% refGrid, the x and y locations of the CENTERS of the centroiding
% boxes (subapertures) found from findRefCent.m
% len, the number of lenslets used
% OUTPUT:
% cent, the current centroid locations in the WFS subapertures
% ------------------------------------------------------------------
----

% Initialize centroid storage
cent = zeros(len,2);
```

```
n = 10;    % number of pixels before/after to black out for Basler
(same as findRefCent.m)

% Find centroids
for i = 1:len
    % Set the same centroid subaperture as found in refGrid
    % (y = rows, x =s columns, origin of this box is the top left
corner)

    P = double(a(refGrid(i,2)-n:refGrid(i,2)+n,refGrid(i,1)-
n:refGrid(i,1)+n));
    S = size(P); Sy = S(1); Sx = S(2);

    % Compute centroid
    M = sum(sum(P));            % find total "weight"
    if M == 0                   % account for intensity dropout
        M = 1;                  % avoids dividing by zero
    end
    X = (linspace(1,Sx,Sx))';   % vector of positions
    Y = linspace(1,Sy,Sy);
    Rx = P*X;                   % weighted x's
    Ry = Y*P;                   % weighted y's
    Rx = sum(Rx)*1/M;           % x centroid in box
    Ry = sum(Ry)*1/M;           % y centroid in box

    % Return centroid x and y locations
    if Rx == 0 && Ry == 0       % account for intensity dropout
%         cent(i,1) = Rx-1 + refGrid(i,1);    % centroid in center
%         cent(i,2) = Ry-1 + refGrid(i,2);    % centroid in center
        cent(i,1) = Rx+n-1 + refGrid(i,1);  % centroid in bottom right
        cent(i,2) = Ry+n-1 + refGrid(i,2);  % centroid in bottom right
    % No intensity dropout - assign centroid normally
    else
        cent(i,1) = Rx-n-1 + refGrid(i,1);
        cent(i,2) = Ry-n-1 + refGrid(i,2);
    end
end
```

C. SIMULINK CODE

This section presents the code developed in this research to run the Simulink models shown in Appendix A. All models with the exception of the Kalman filter are initialized with the actuator space parameters in the file, *AOTestbed_params_act*. The Kalman filter is initialized with parameters of its own in the file, *Kalman_params*. The Level-2 M-file S-Function, *hardwareControl*, was shown in Figure 61 in the Hardware Control subsystem and is presented in this section as well. Most of the code in this S-Function will be familiar, as it performs the same centroid calculations as the *findCent*

function and uses the *BAOGrab* and *BAODMirror* hardware interface functions used previously. The S-Function is built from Matlab's template. The dialog parameters used in the S-Function are shown in Figure 80. All other parameters used are obtained within the function.

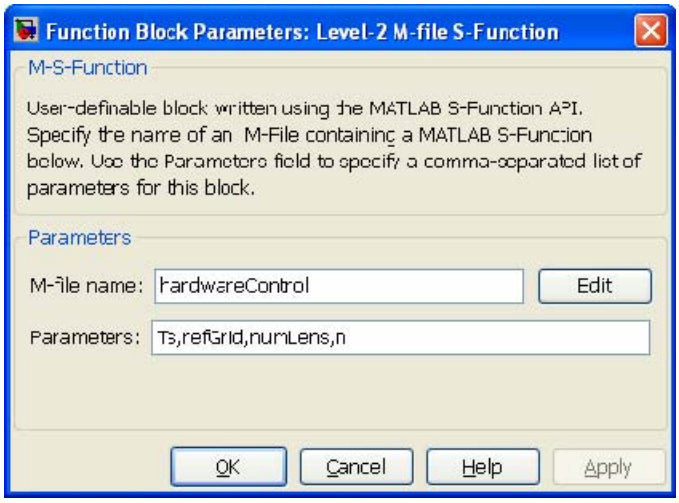

Figure 80. Parameters passed to *hardwareControl* S-Function

1. AOTestbed_params_act.m

```
% Parameters for actuator space adaptive filter models
% ------------------------------------------------------------------------
----
Ts = 0.0667;          % sample time for discrete system

% Initial condition for discrete integrator
u1 = 0;

% Take SVD of poke matrix
C = pokeSlopes;
[U,Sig,V] = svd(C,0);

% Find pseudoinverse of poke matrix from SVD
invSig = inv(Sig);
E0 = V*invSig*U.';
[u,s,v] = svd(C);    % complete set of basis vectors

K = size(C,2);       % number of DM actuators
M = size(C,1);       % number of sensor measurements

% PI controller (with zpk model)
% ------------------------------------------------------------------------
----
z1=0.3;
k1=0.2;
```

```
% Convert to Kp+Ki*Ts*z/(z-1) form
% 0.45 (z-0.3)     (Kp+KiTS)z-Kp
% ------------  =  ------------
%    (z-1)            (z-1)

Kp=z1*k1;
Ki = (k1-Kp)/Ts;
Kpmod = Kp;
Kimod = Ki;

% Adaptive filter design
% ------------------------------------------------------------------------
----
% number of reference signals for adaptive filter
Ja = K;   %--- disturbance in actuator space
Js = M;   %--- disturbance in sensor space
J = M;

% Small number for normalization
e0 = 1.192092896e-07;

% Number of weights for each Wkj
L = 20;

% Set adaptation rate
a_act = .01;    % For coupled AF
% a_act = .001;   % For coupled AF+PI
```

2. Kalman_params.m

```
% Kalman Filter parameters
% ------------------------------------------------------------------------
----
% Kp and Ki chosen from adaptive filter algorithms
Ts = .0667;
Kp = 0.06;
Ki = 2.099;
zinv = tf([1],[1 0],Ts);
sysP = tf([Kp],[1],Ts);
sysI = tf([Ki*Ts,0],[1 -1],Ts);

% Sensitivity transfer function calculated by hand
% CL TF and inv(CL TF) calculated by hand and implemented in Simulink
as
% discrete transfer functions vs. LTI objects for multichannel control
sf = tf([1 -1 0],[1 Kp+Ki*Ts-1 -Kp],Ts);

start_time_kalman = 1;

% System identification of disturbance
```

```
% ------------------------------------------------------------------------
----
% Run distInToAct.mdl each time new disturbance is loaded; runs sensor
data
% through pokeInv (reconstructor) to generated actuator space
disturbance
% for 37-channel Kalman filter.  "distInA" is output of this model.
d = distInA;

% Identify disturbance for all channels
distOrder = 4;
sys1 = ar(iddata(d(:,1),[],Ts),distOrder);
sys1tf = tf(sys1);
sys1S = sys1tf*sf;  % Disturbance as seen by KF
sys_all = sys1S;
for i = 2:37
    sysi = ar(iddata(d(:,i),[],Ts),distOrder);
    sysitf = tf(sysi);
    sysiS = sysitf*sf;
    sys_all = append(sys_all,sysiS);
    sys_all = minreal(sys_all);
end
% Create state space model from transfer function model
sysAll = ss(sys_all);
numState = size(sysAll.a,1);

% Process noise
wp = 1.0e-6*Ts;
vp = 1.0e-9*Ts;
% Measurement noise
wpe = wp/Ts;
vpe = vp/Ts;
```

3. hardwareControl.m

```
function hardwareControl(block)
%MSFUNTMPL_BASIC A template for a Leve-2 M-file S-function
%   The M-file S-function is written as a MATLAB function with the
%   same name as the S-function. Replace 'msfuntmpl_basic' with the
%   name of your S-function.
%
%   It should be noted that the M-file S-function is very similar
%   to Level-2 C-Mex S-functions. You should be able to get more
%   information for each of the block methods by referring to the
%   documentation for C-Mex S-functions.
%
%   Copyright 2003-2007 The MathWorks, Inc.

%%
%% The setup method is used to setup the basic attributes of the
%% S-function such as ports, parameters, etc. Do not add any other
%% calls to the main body of the function.
```

```
%%
setup(block);

%endfunction

%% Function: setup ===================================================
%% Abstract:
%%   Set up the S-function block's basic characteristics such as:
%%   - Input ports
%%   - Output ports
%%   - Dialog parameters
%%   - Options
%%
%%   Required         : Yes
%%   C-Mex counterpart: mdlInitializeSizes
%%
function setup(block)

% Register number of ports
block.NumInputPorts  = 1;
block.NumOutputPorts = 2;

block.AllowSignalsWithMoreThan2D;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

% Register parameters
block.NumDialogPrms     = 4;

NUM_ACTUATOR_CH     = block.DialogPrm(4).Data;
% Override input port properties
block.InputPort(1).Dimensions        = NUM_ACTUATOR_CH;
block.InputPort(1).DatatypeID  = 0;  % double
block.InputPort(1).Complexity  = 'Real';
block.InputPort(1).DirectFeedthrough = false;
block.InputPort(1).SamplingMode = 0;

% Override output port properties
block.OutputPort(1).Dimensions       = block.DialogPrm(3).Data*2;
block.OutputPort(1).DatatypeID  = 0; % 0 = double, uint8 = 3, uint16 =
5
block.OutputPort(1).Complexity  = 'Real';
block.OutputPort(1).SamplingMode = 0;

block.OutputPort(2).Dimensions       = block.DialogPrm(3).Data;
block.OutputPort(2).DatatypeID  = 0; % 0 = double, uint8 = 3, uint16 =
5
block.OutputPort(2).Complexity  = 'Real';
block.OutputPort(2).SamplingMode = 0;

% Register sample times
```

```
%  [0 offset]            : Continuous sample time
%  [positive_num offset] : Discrete sample time
%
%  [-1, 0]               : Inherited sample time
%  [-2, 0]               : Variable sample time
%block.SampleTimes = [-1 0];
block.SampleTimes = [block.DialogPrm(1).Data 0];

%% -----------------------------------------------------------------
%% The M-file S-function uses an internal registry for all
%% block methods. You should register all relevant methods
%% (optional and required) as illustrated below. You may choose
%% any suitable name for the methods and implement these methods
%% as local functions within the same file. See comments
%% provided for each function for more information.
%% -----------------------------------------------------------------

%block.RegBlockMethod('PostPropagationSetup',    @DoPostPropSetup);
%block.RegBlockMethod('InitializeConditions', @InitializeConditions);
block.RegBlockMethod('SetInputPortSamplingMode',@SetInputPortSamplingMo
de);
%block.RegBlockMethod('SetInputPortDimensions', @SetInpPortDims);
%block.RegBlockMethod('Start', @Start);
block.RegBlockMethod('Outputs', @Outputs);     % Required
%block.RegBlockMethod('Update', @Update);
%block.RegBlockMethod('Derivatives', @Derivatives);
block.RegBlockMethod('Terminate', @Terminate); % Required

%end setup


%%
%% PostPropagationSetup:
%%   Functionality    : Setup work areas and state variables. Can
%%                      also register run-time methods here
%%   Required         : No
%%   C-Mex counterpart: mdlSetWorkWidths
%%
function DoPostPropSetup(block)

% Register all tunable parameters as runtime parameters.
%block.AutoRegRuntimePrms;

%endfunction

%%
%% InitializeConditions:
%%   Functionality    : Called at the start of simulation and if it is
%%                      present in an enabled subsystem configured to
reset
%%                      states, it will be called when the enabled
subsystem
%%                      restarts execution to reset the states.
%%   Required         : No
```

```
%%   C-MEX counterpart: mdlInitializeConditions
%%
function InitializeConditions(block)

%end InitializeConditions

%%
%% Start:
%%   Functionality    : Called once at start of model execution. If you
%%                      have states that should be initialized once,
this
%%                      is the place to do it.
%%   Required         : No
%%   C-MEX counterpart: mdlStart
%%
function Start(block)

%endfunction

%%
%% Outputs:
%%   Functionality    : Called to generate block outputs in
%%                      simulation step
%%   Required         : Yes
%%   C-MEX counterpart: mdlOutputs
%%
function Outputs(block)

% Send current command to DM: convert control signal to voltages
V = sqrt((block.InputPort(1).data+1)*.5)*255;
BAODMirror(hex2dec('6800'),hex2dec('6400'),V);

% Get WFS camera image
BAOGrab(0);
a = BAOGrab(1);
BAOGrab(2);

refGrid = block.DialogPrm(2).Data;
len = block.DialogPrm(3).Data;

% Initialize centroid storage
cent = zeros(len,2);
dropouts = zeros(len,1);

n = 10;    % number of pixels before/after to black out for Basler
(same as findRefCent.m)

% Find centroids
for i = 1:len
    % Set the same centroid subaperture as found in refGrid
    % (y = rows, x =s columns, origin of this box is the top left
corner)
```

```
    P = double(a(refGrid(i,2)-n:refGrid(i,2)+n,refGrid(i,1)-
n:refGrid(i,1)+n));
    S = size(P); Sy = S(1); Sx = S(2);

    % Compute centroid
    M = sum(sum(P));            % find total "weight"
    if M == 0                   % account for intensity dropout
        M = 1;                  % avoids dividing by zero
    end
    X = (linspace(1,Sx,Sx))';   % vector of positions
    Y = linspace(1,Sy,Sy);
    Rx = P*X;                   % weighted x's
    Ry = Y*P;                   % weighted y's
    Rx = sum(Rx)*1/M;           % x centroid in box
    Ry = sum(Ry)*1/M;           % y centroid in box

    % Return centroid x and y locations
    if Rx == 0 && Ry == 0       % account for intensity dropout
%         cent(i,1) = Rx-1 + refGrid(i,1);    % centroid in center
%         cent(i,2) = Ry-1 + refGrid(i,2);    % centroid in center
        cent(i,1) = Rx+n-1 + refGrid(i,1);  % centroid in bottom right
        cent(i,2) = Ry+n-1 + refGrid(i,2);  % centroid in bottom right
        dropouts(i) = 1;                    % indicate dropout
    % No intensity dropout - assign centroid normally
    else
        cent(i,1) = Rx-n-1 + refGrid(i,1);
        cent(i,2) = Ry-n-1 + refGrid(i,2);
        dropouts(i) = 0;                    % indicate no dropout
    end
end

% Output centroids and dropouts
block.OutputPort(1).Data = reshape(cent,1,len*2);
block.OutputPort(2).Data = dropouts;

%end Outputs

%%
%% Update:
%%   Functionality    : Called to update discrete states
%%                      during simulation step
%%   Required         : No
%%   C-MEX counterpart: mdlUpdate
%%
function Update(block)

%end Update

%%
%% Derivatives:
%%   Functionality    : Called to update derivatives of
%%                      continuous states during simulation step
%%   Required         : No
%%   C-MEX counterpart: mdlDerivatives
```

```
%%
% function Derivatives(block)

%end Derivatives

%%
%% Terminate:
%%   Functionality    : Called at the end of simulation for cleanup
%%   Required         : Yes
%%   C-MEX counterpart: mdlTerminate
%%
function Terminate(block)

%end Terminate
```

The Simulink models and Matlab code provided in Appendices A and B are the foundation for the research and results presented in this dissertation. They can be modified and improved for future research with the AO testbed developed in this work.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Adaptive Optics Guide. (2008). Delft: OKO Technologies.

Allen, M. R. (2007). Wavefront control for space telescope applications using adaptive optics. Monterey: NPS Master's Thesis.

Andrews, L. C. (2004). Field guide to atmospheric optics. Bellingham: SPIE Press.

Beerer, M. J. (2008). Adaptive filter techniques for optical beam jitter control and target tracking. Monterey: NPS Master's Thesis.

Burtz, D. C. (2009). Fine surface control of flexible space mirrors using adaptive optics and robust control. Monterey: NPS PhD Dissertation.

Edwards, S. G. (1999). Active narrowband disturbance rejection on an ultra quiet platform. Monterey: NPS PhD Dissertation.

Ellerbroek, B. L., & Rhoadarmer, T. A. (1998). Real-time adaptive optimization of wave-front reconstruction algorithms for closed-loop adaptive-optical systems. SPIE Adaptive Optical System Technologies, 3353.

Elliott, S. J., & Nelson, P. A. (1985). The application of adaptive filtering to the active control of sound and vibration. ISVR: Technical Report 136.

Elliott, S. J., et al. (1987). A multiple error LMS algorithm and its application to the active control of sound and vibration. IEEE Transactions on Acoustics, Speech, and Signal Processing, 35(10).

Fried, D. L. (1965). Statistics of a geometric representation of wavefront distortion. J. Opt. Soc. Am., 55(11).

Fried, D. L. (1998). Branch point problem in adaptive optics. J. Opt. Soc. Am., 15 (10).

Fried, D. L., & Vaughn, J. L. (1992). Branch cuts in the phase function. J. Opt. Soc. Am., 31(15).

Gibson, J. S., et al. (2000). Adaptive optics: wave-front correction by use of adaptive filtering and control. Applied Optics, 39(16).

Hammel, S. (2007). Turbulence effects on laser propagation in a marine environment. SPIE Atmospheric Optics: Models, Measurements, and Target-In-The-Loop Propagation, 6708.

Hammel, S., et al. (2004). Atmospheric characterization for high energy laser beam propagation in the maritime environment. SPIE Target-In-The-Loop: Atmospheric Tracking, Imaging, and Compensation, 5552.

Haykin, S. (2002). Adaptive filter theory. Upper Saddle River: Prentice-Hall, Inc.

Jiang, S.-B., & Gibson, J. S.. (1995). An unwindowed multichannel lattice filter with orthogonal channels. IEEE Transactions on Signal Processing, 43(12).

Kuo, S. M., & Morgan, D. R. (1996). Active noise control systems. New York: John Wiley & Sons, Inc.

Lamberson, S., et al. (2007). The Airborne Laser. SPIE International Symposium on Gas Flow, Chemical Lasers, and High-Power Lasers, 6346.

Liu, Y.-T., & Gibson, J. S. (2007). Adaptive control in adaptive optics for directed-energy systems. Optical Engineering, 46(4).

Mantravadi, S. V., et al. (2004). Simple laboratory system for generating well-controlled atmospheric-like turbulence. SPIE Advanced Wavefront Control: Methods, Devices, and Applications II, 5553.

Missile Defense Agency. (2010). *Ballistic Missile Defense System/Supporting Efforts/Airborne Laser Test Bed*. Retrieved on 3 Sep 2010, from http://www.mda.mil/system/altb.html.

Noll, R. J. (1976). Zernike polynomials and atmospheric turbulence. J. Opt. Soc. Am., 66(3).

Rhoadarmer, T. A., et al. (2006). Adaptive control and filtering for closed-loop adaptive-optical wavefront reconstruction. SPIE Advanced Wavefront Control: Methods, Devices, and Applications IV, 6306(1).

Roggemann, M. C., & Welsh, B. (1996). Imaging through turbulence. Boca Raton: CRC Press, Inc.

Sanchez, D. J., & Oesch, D. W. (2009). The aggragate behavior of branch points I – the creation and evolution of branch points. Albuquerque: Air Force Research Laboratory Technichal Paper 1012.

Santiago, F., et al. (2005). Low altitude horizontal scintillation measurements of atmospheric turbulence over the sea: experimental results. SPIE Active and Passive Optical Components for WDM Communications V, 6014, 2005.

Sergeyev, A., & Roggemann, M. C. (2010). Near the ground laser communications system: Fried parameter estimation from the WFS measurements. IEEE Aerospace Conference.

Spacecraft Research and Design Center. (2008). HEL Testbed Poster. Monterey: NPS.

Tyson, R. K. (2000). Introduction to adaptive optics. Bellingham: SPIE Press.

Tyson, R. K., & Frazier, B. W. (2004). Field guide to adaptive optics. Bellingham: SPIE Press.

Vetelino, F. S., et al. (2006). Initial measurements of atmospheric parameters in a marine environment. SPIE Atmospheric Propagation III, 6215.

Vorontsov, M., et al. (2009). Deep turbulence effects compensation experiments with a cascaded adaptive optics system using a 3.63m telescope. J. Opt. Soc. Am., 48(1).

Watkins, J. R., & Agrawal, B. N. (2007). Use of least mean squares filter in control of optical beam jitter. AIAA Journal of Guidance, Control, and Dynamics, 30(4).

Widrow, B., & Stearns, S. D. (2002). Adaptive signal processing. Upper Saddle River: Prentice-Hall, Inc.

Wilcox, C. C. (2009). Optical phase aberration generation using a liquid crystal spatial light modulator. Albuquerque: University of New Mexico PhD Dissertation.

Yoon, H., et al. (2008). Laser beam jitter control using recursive-least-squares adaptive filters. DEPS Beam Control Conference.

Yoon, H. (2008). Optical beam jitter control with adaptive filtering techniques. NPS Spacecraft Research and Design Center Slide Presentation.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. Brij N. Agrawal
   Naval Postgraduate School
   Monterey, California

4. Dr. Roberto Cristi
   Naval Postgraduate School
   Monterey, California

5. Dr. Jae Jun Kim
   Naval Postgraduate School
   Monterey, California

6. Dr. Marcello Romano
   Naval Postgraduate School
   Monterey, California

7. Dr. Oleg Yakimenko
   Naval Postgraduate School
   Monterey, California

8. Dr. Masaki Nagashima
   Naval Postgraduate School
   Monterey, California

9. Dr. Italo Toselli
   Naval Postgraduate School
   Monterey, California

10. Freddie Santiago
    Naval Research Laboratory
    Albuquerque, New Mexico