Modal Control with State Estimation for Advanced LIGO Quadruple Suspensions

by

Brett N. Shapiro

B.S. Engineering Science
The Pennsylvania State University, 2005

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 2007

Signature of Author: _____
Department of Mechanical Engineering
August 10, 2007

Certified by: _____
Nergis Mavalvala
Associate Professor of Physics
Thesis Supervisor

Certified by: _____
David Trumper
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by: _____
Lallit Anand
Professor of Mechanical Engineering
Chairman, Committee on Graduate Students

Modal Control with State Estimation for Advanced LIGO Quadruple Suspensions

by

Brett N. Shapiro

Submitted to the Department of Mechanical Engineering
on June 15, 2007 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

Gravitational waves are predicted to exist by Einstein's General Theory of Relativity. These waves are distortions of space-time which until now have remained outside the realm of possible detection, due to their incredibly weak interactions. Now, due to newly built highly sensitive observatories, such as the Laser Interferometer Gravitational Wave Observatory (LIGO), these detections are now believed to be possible and may occur in the very near future.

The research discussed here is on an active control system for the quadruple pendulum, from which the mirrors of the LIGO interferometer are suspended. This pendulum is a chain of four masses used to provide seismic isolation of the mirrors at the level of 10^{-19} m $Hz^{-1/2}$ at 10 Hz. Because the pendulum is so quiet above 10 Hz, the sensor noise used in the active control is not trivial. Thus, the purpose of this research is to optimize a control scheme that has high gain at the resonant frequencies of the pendulum to provide damping while at the same time rolling the gain off to virtually zero at the limit of the gravitational wave detection band less than half a decade away. This requirement is very difficult to achieve with classical control design techniques.

The alternative method explored here is a type of modal control with state estimation where incomplete sensor information is reconstructed and mathematically decomposed into modal responses. The modal responses can be thought of as simple single degree of freedom oscillators that are very easy to control. In this way, a few highly complicated controllers are traded for a larger collection of reasonably simple ones that are easy to design for each mode. Damping vs. noise injection can then be optimized by tailoring the control gain on each mode.

Acknowledgements

This thesis begins my life as a mechanical engineering graduate student and is perhaps my first real introduction into what it means to be an engineer. I would like to thank all those who made this effort possible.

First, I must thank Laurent Ruet. Without his original contribution to this topic this thesis would not be here at all. Next, my supervisor Rich Mittleman for his patience, support, and keeping me focused on the end result. Of course I must mention my advisors as well, Professors Nergis Mavalvala and David Trumper for all their feedback and input into this document.

There are also hosts of other people around LASTI and MIT LIGO that need mentioning. I would like to thank Marie Woods for all her help with travel, shipping, and expenses. Myron MacInnis and Bob Laliberte's support made the extraordinary work of setting up and maintaining the experiments possible. For all those that gave me an excuse to get out of the lab and off campus, thanks, you kept me sane.

The SUS group must receive a special recognition as well. Never have I had access to such an extensive group of professional engineers and scientists. I have learned an immeasurable amount from them all.

The last group of people that needs mentioning was in no way involved with this project but yet still managed to have an enormous impact. First, Jamie. Without her, MIT would just not be the same. Finally, last but not least, a special thanks to my parents and my sister Nina for all their support and encouragement.

Table of Contents

Chapter 1

Introduction: Gravitational Waves and LIGO

This introduction provides a brief look into the nature of gravitational waves (GWs) and how the Laser Interferometer Gravitational Wave Observatory (LIGO) intends to pursue its quest of beginning a new field of GW astronomy. It will then briefly describe the goals of this thesis and how they are important to the work of LIGO.

Beyond the introduction, Chapter 2 describes in detail the quadruple pendulum and its requirements. Chapters 3 and 4 discuss and compare two different control methods for this pendulum. Chapter 5 sets up the experiment and compares actual data using these methods. Chapter 6 provides a detailed description of the current state of this research with the quadruple pendulum and looks briefly at where it may be going in the future. Finally, concluding remarks are provided in Chapter 7.

1.1 Behavior of Gravitational Waves

GWs are a phenomena predicted to exist by Einstein's Theory of General Relativity. In this theory accelerating mass produces GWs in a manner analogous to the way accelerating charges produce electromagnetic waves. Additionally, GW's interaction with surrounding space is incredibly weak. Only the most massive and violent events in the universe such as supernovae and collisions of black holes or neutron stars produce GWs strong enough to even consider detecting.

The effect of GWs is quite different from electromagnetic waves. As they propagate they compress and stretch space. Consider the ring of particles in Figure 1.1

below. If the axis of propagation is perpendicular to the page then the shape of space is altered in the plane of the page, perpendicular to propagation. Two perpendicular axes in this plane simultaneously have opposite effects; one is stretched while the other is compressed.

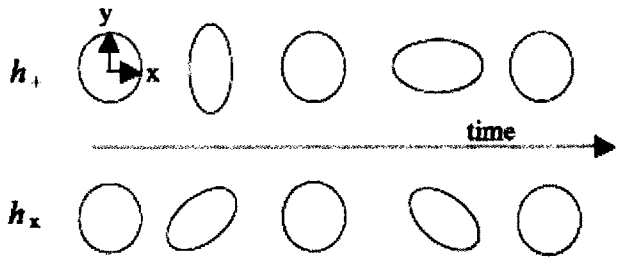


Figure 1.1: Effect of a GW on a ring of particles. The top half shows the effect of a wave polarized with axes, x and y, parallel to the vertical and horizontal axes of the page. The lower half shows the effect of a wave polarized at 45° relative to the upper wave. Space is alternately compressed and stretched along the wave's x and y axes as it propagates through the plane. Taken from [1].

The figure shows two different polarizations of a wave. Polarization is defined as the tilt of the plane wave's x and y axis relative to the observer. The upper half of the figure, h_+, is a case when the polarization is parallel with the vertical and horizontal axes of the page. The lower one is a case when the polarization is tilted 45°. Also, at the observational distances we are considering here, all waves are assumed to be planar.

The stretching of space is related to strain, or in other words a proportional effect. If two objects are far apart, the distance between them will change more than for two objects that are close together. The amount of strain produced by a GW, h, is proportional to the strength of the wave [1]. For a GW of strength h, the specified distance L is altered by $\Delta L = hL$.

1.2 Sources of Gravitational Waves

Sources of GWs that are expected to be strong enough for detection by an observatory such as LIGO include supernovae, coalescing compact binaries, pulsars, and a stochastic background of GWs.

Supernovae are the explosive death of massive stars. If the collapse of the stellar core is perfectly spherically symmetric, then no waves will be produced, however it is

believed this is generally not the case. For example, if the core is spinning its collapse will not be symmetric and strong waves can be produced. The exact evolution of a supernova is not well known, so this source provides an excellent opportunity to expand scientific knowledge of such phenomena.

Coalescing binaries include the merging of a pair of compact, massive objects such as black holes, neutron stars, or a combination of the two. Such events begin with the objects orbiting around their common center of mass. Slowly the orbital distance and period decays due to the slow emission of gravitational radiation. The pair begins spiraling into the center of mass and the GW signal increases both in frequency and amplitude creating a chirp-like signal. Finally they both crash into each other emitting a final burst of gravitational radiation.

Pulsars are rotating neutron stars that emit radiation due to their rotation. This is another mechanism for neutron stars to emit observable gravitational radiation. Again, spherical symmetry is the enemy of wave emission, but if the star is slightly asymmetric and rotating, then waves strong enough to observe from Earth may be produced.

A final source for GW astronomy is a stochastic background of GWs. This could arise from an incoherent superposition of many different signals or a primordial GW background from, for example, the Big Bang. This type of background signal is expected to be broadband and extremely weak. However, if data is integrated over long periods of time and cross-correlated between multiple detectors, coherence may be found indicating, the existence of such a background [1, 2].

1.3 LIGO

For the first time the direct detection of GWs may now be possible due to the work of new observatories such as the Laser Interferometer Gravitational Wave Observatory (LIGO). To date, no direct detections of gravitational radiation have been made due to the incredible weakness of gravitational radiation. A typical wave produced from such powerful sources as described in section 1.2 is expected to induce a strain of only 10^{-21}, about 1000 times smaller than the diameter of a proton for 4km observation paths used in LIGO. Earth-based detectors, such as LIGO, have the added complication of natural and

manmade noise traveling through the ground. Nearly all the work of these detectors up to this point is focused on developing ways to amplify the signal and reduce sources of noise that will wash out these infinitesimally small signals.

To make these detections possible, LIGO comprises two observatories: one is in Hanford, Washington and the other in Livingston, Louisiana. See Figures 1.2 and 1.3 respectively.

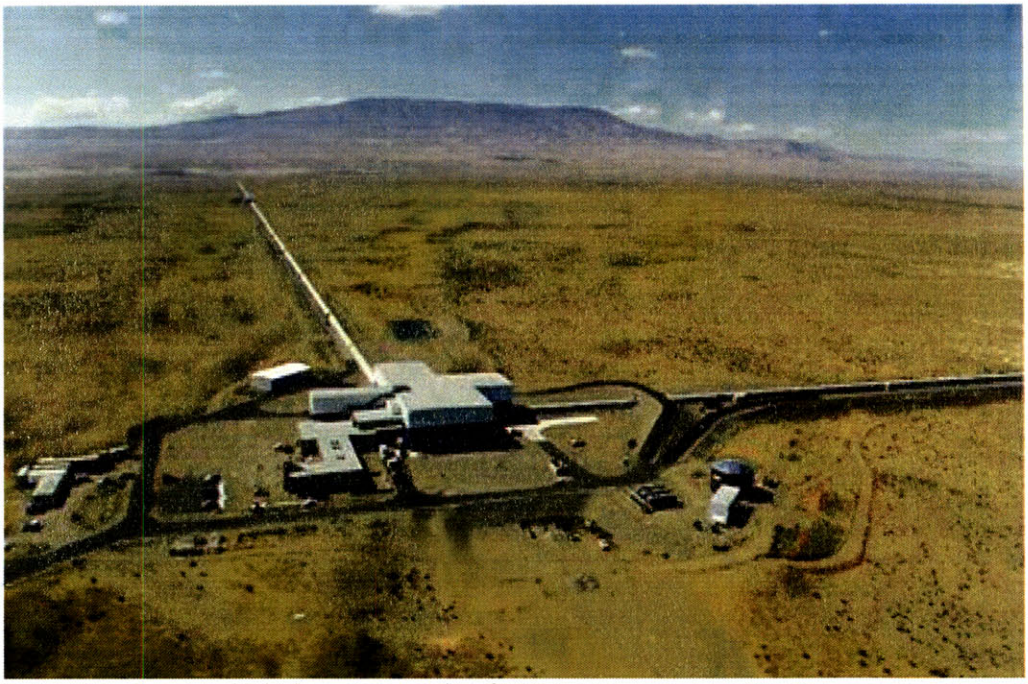


Figure 1.2: LIGO Hanford Observatory. Taken from [3].

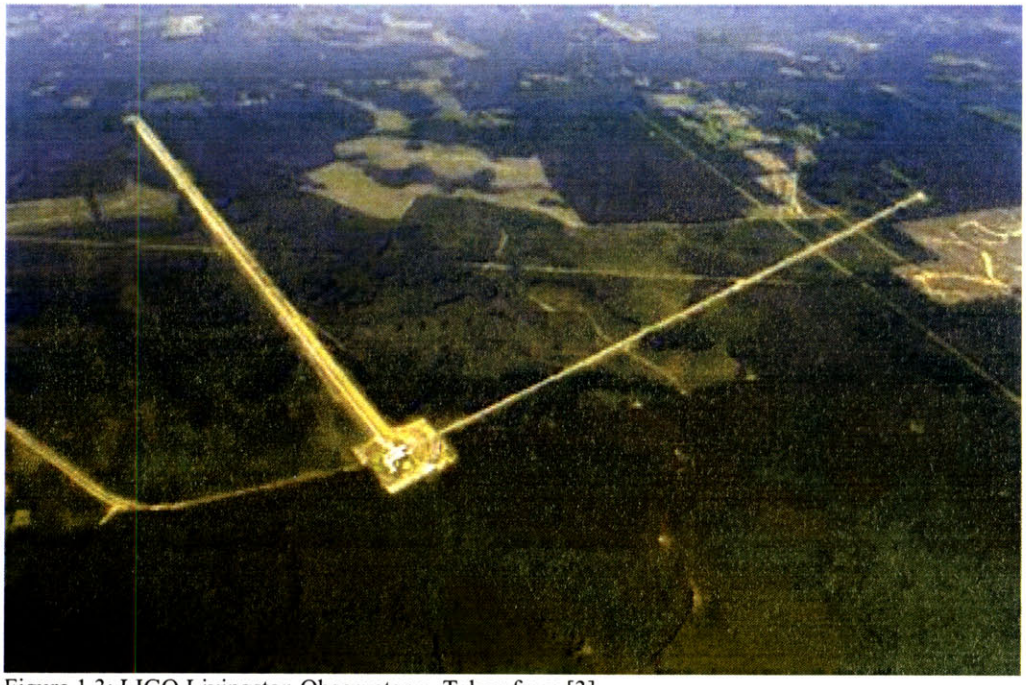


Figure 1.3: LIGO Livingston Observatory. Taken from [3].

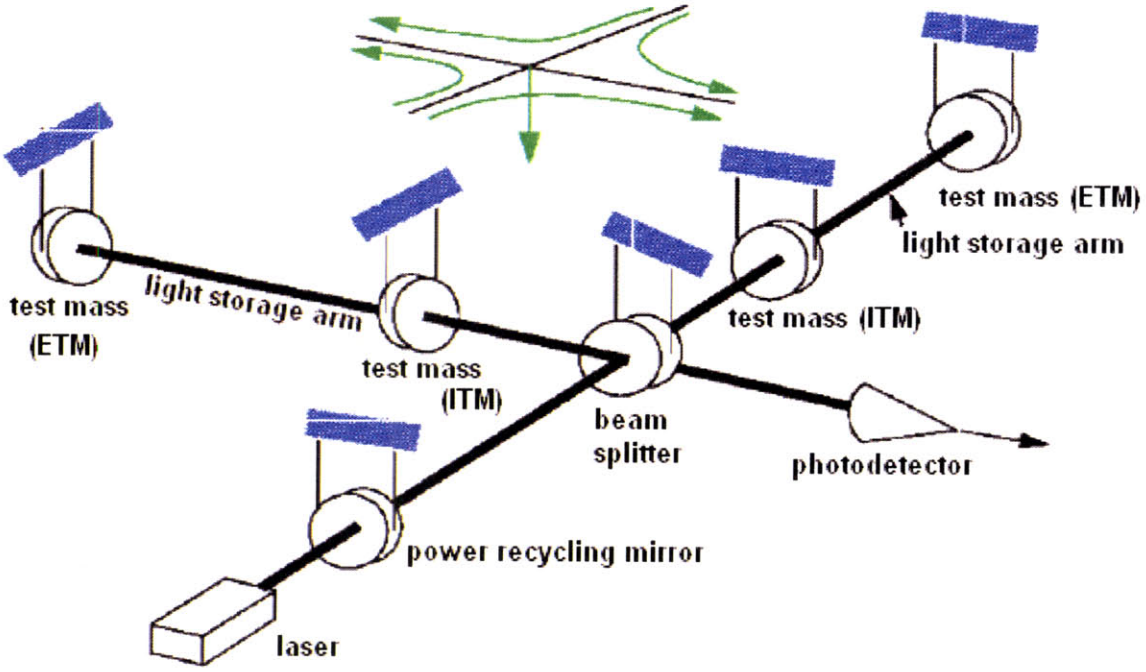


Figure 1.4 Schematic diagram of LIGO's interferometer. Adapted from [1].

Each LIGO Detector consists of a Michelson interferometer with 4 km long Fabry-Perot cavities in each perpendicular arm, used to measure the differential arm length caused by the passing wave. The arms are this long in order to maximize the length change induced by the strain of the wave. The interferometer is housed in a vacuum envelope evacuated to about 10^{-9} torr to prevent interference from gas molecules.

Figure 1.4 shows a basic schematic of these interferometers. The exact details of how these detectors work are rather complicated, but the essential principles can be described as follows. A laser injects 1064 nm (infrared) light into a beam splitter that splits the light into the two orthogonal arms. A Fabry-Perot cavity in each arm of the Michelson, comprising an input test mass (ITM) and an end test mass (ETM), stores the laser light to increase the phase sensitivity of the interferometer. The optics are also suspended as pendula in order to isolate them from ground motion and act as free test particles. A photon will make approximately 100 round trips between these optics in order to amplify the length measurement of each arm. The light then recombines at the beam splitter and continues on to a photodetector. Measuring the intensity of the light at this photodetector provides a measure of the phase difference of the light in each arm, and thus the differential length [1].

The purpose of multiple observatories is motivated by several factors. One is to locate which part of the sky the GW came from and determine its polarization. Another is to create veto scenarios so that GWs can be distinguished from local noise sources such as a heavy truck driving down a nearby road. A GW of cosmic origin should be present coincidentally at both sites whereas the truck will be present at only one. As an extra check on these vetoes the Hanford site actually has a third, 2k interferometer. It is possible that a real GW will only show up at one site if it comes in at an insensitive direction for one interferometer. A collocated half length interferometer still has the benefit that only a GW can produce a simultaneous, strain signal in both with half the displacement in the shorter interferometer. It would be very coincidental for a local event to produce exactly half the displacement in the 2 km detector as the 4 km [4].

In order to make the detectors sensitive enough to see GWs, many different types of noise in the LIGO detection band (~10 to 10000 Hz) have to be eliminated or reduced.

The dominating sources of noise are categorized into seismic, thermal, and shot noise. Each tends to dominate in specific parts of the spectrum.

Seismic noise dominates at frequencies below about 50 Hz and rolls off at about $10^{-7}/f^2$ m $Hz^{-1/2}$. The term seismic noise is used to collectively refer to all sources coming through the ground. These sources include actual seismic motion from tectonic plates and volcanic activity. However, it also includes manmade activity as well as noise from waves crashing onto the shores. In LIGO's current configuration with the detection band starting at about 40Hz, this requires displacement noise reduction of about 10^9 at frequencies above a few Hertz.

Thermal noise occurs directly in the detection band and limits sensitivity around 100 Hz. This noise is associated with the mechanical dissipation or loss factor of the optics and their suspensions. As a result, the optics and suspensions are constructed from very low-loss materials. In low-loss materials most of the noise is concentrated around the resonant frequencies because of high mechanical quality factors (Q). The necessity of high Qs for thermal noise performance is the reason LIGO's suspensions use active damping instead of passive [2].

The high frequency end of the spectrum is dominated by shot noise, or random fluctuations in the number of photons in the laser. Shot noise is white, but it rolls up in the spectrum because of the optical response of the arm cavities. The shot-noise-limited sensitivity is inversely proportional to the square root of the laser power, and can thus be reduced with a higher power laser source. Arbitrarily increasing the power of the laser beam introduces a host of other problems though, such as radiation pressure driven motion and heating of the mirrors. Radiation pressure is the force produced on the mirrors by the transfer of momentum from reflected photons [2].

Figure 1.5 shows the contribution of different noise sources to the design sensitivity of LIGO. In 2005, LIGO finally reached this design sensitivity and began a year long search for GWs. Figure 1.6 shows an actual sensitivity curve of the Livingston observatory taken in 2006 along with target sensitivities for Initial LIGO and Advanced LIGO.

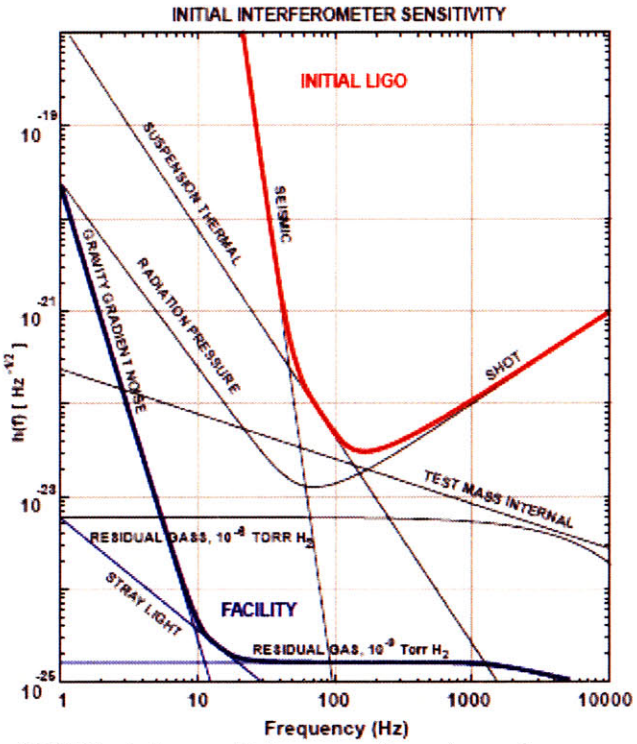


Figure 1.5: Spectrum of LIGO's design sensitivity among the various noise sources. The red curve shows the target sensitivity of the Initial LIGO detectors. The blue curve gives the limits set by the LIGO facilities infrastructure, showing that significantly improved detectors can be housed in the same facility. Taken from [2].

Currently, research and development for the next phase of LIGO, Advanced LIGO, is underway. Its design promises to improve sensitivity by another order of magnitude. Noise will be reduced by upgrading seismic isolation, suspending the optics from low-loss fused silica fibers, and increasing the input power of the laser beam. Figure 1.6 below includes a sample target sensitivity (purple) of this new configuration.

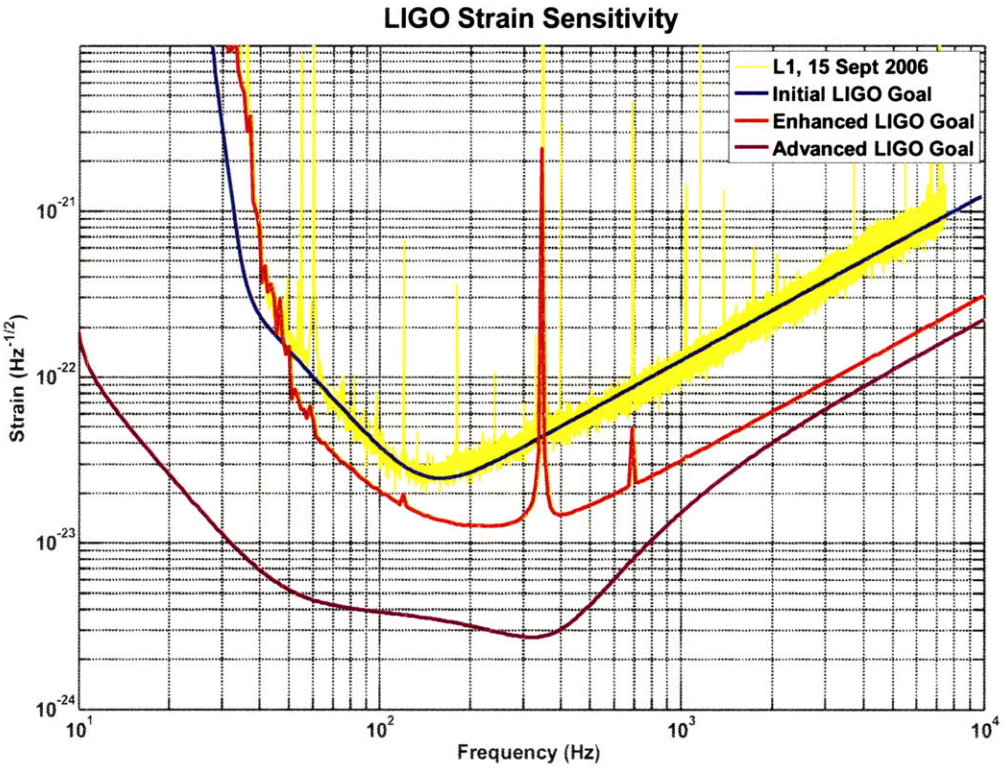


Figure 1.6: Measured strain sensitivity of the Livingston observatory (yellow) with target sensitivities for Initial (blue), Enhanced (red) , and Advanced (purple) LIGO. The maximum improvement in sensitivity for Advanced LIGO is designed to be about 10 times greater than that of Initial LIGO. The Enhanced LIGO configuration is designed to include minor upgrades to initial LIGO during the final years of research and development of Advanced LIGO (Courtesy P. Fritschel and N. Mavalvala).

The upgrade to Advanced LIGO involves the installation of more sophisticated seismic isolation systems to improve sensitivity in the seismic and thermal noise dominated band of the spectrum. The research in this thesis focuses on the upgraded suspension systems to be used as a part of this upgrade. These new suspensions will expand the current single stage ETM and ITM pendula into four stage quadruple pendula.

Each mass of the quadruple pendulum provides f^2 isolation above the resonant frequencies; by using four stages of masses the suspensions can achieve f^8. In such pendula, the motion at the resonant frequencies around 0.5-5 Hz must be sensed and actively damped. It is this damping control that is the specific topic of discussion for this thesis. The quadruple pendulum will be discussed with more detail in Chapter 2.

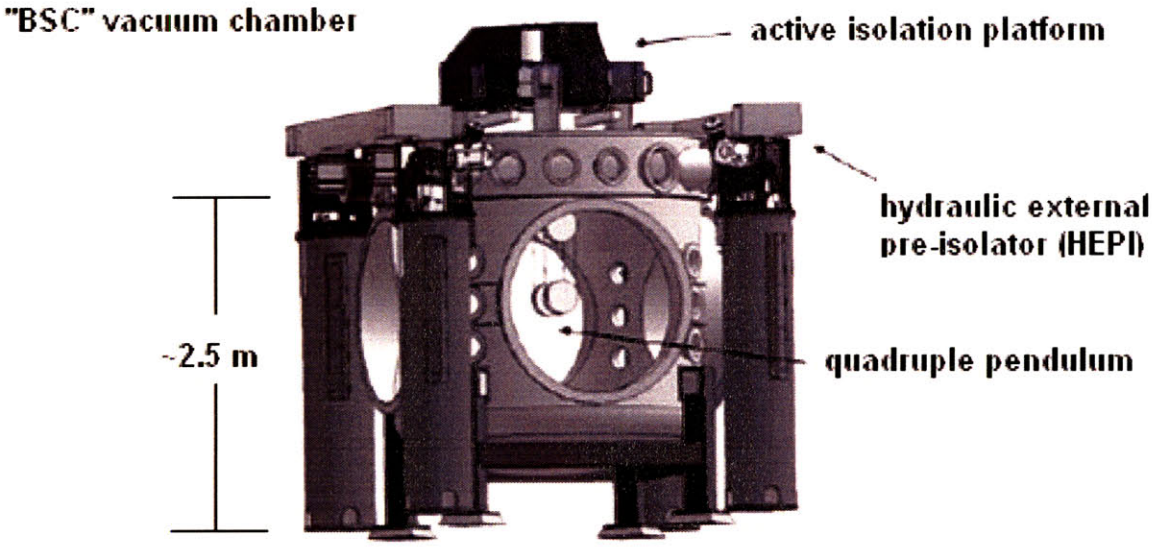


Figure 1.7: Layout of the 3 cascading stages of seismic attenuation in Advanced LIGO: the hydraulic external pre-isolator (HEPI), an active isolation system external to the vacuum chamber; a two stage active isolation system inside the vacuum chamber; and a quadruple pendulum system whose bottom mass is an interferometer mirror. Adapted from [6].

The pendulum chains rest on two prior cascading stages of seismic isolation. Figure 1.7 is an illustration of how these three systems fit together. The first, known as HEPI (Hydraulic External Pre-Isolator), senses and actively removes ground motion at low frequencies with a bandwidth of about 10 Hz. HEPI functions outside the vacuum chambers that house the optics and suspension systems. Its specifications call for actuation in all six DOFs, force generation greater than 2000 N, a range of ±1 mm, and a noise floor no greater than 10^{-9} m/√Hz at 1 Hz. A laminar flow hydraulic actuator meets these requirements and is used in HEPI [6].

The second stage, which supports the suspensions inside the chambers, is an active isolation platform. This platform is actually two stages itself and functions in a similar way to the suspensions. The two stages are suspended by stiff blade springs and provide passive isolation above the resonant frequencies of 2 to 10 Hz. The quadruple pendulum hangs from the lower stage which is equipped with an optics table. The motion in the six degrees of freedom (DOFs) of both stages is sensed and actively damped to reduce the Qs of the resonant frequencies [6].

Chapter 2

The ETM Quadruple Pendulum

The quadruple pendula are an extension of the three stage, triple pendula designed for use in the German-British Gravitational Wave Detector (GEO600). Reference [7] details the design of the triple pendulum and is an excellent source to review the thought process used in extending the mathematics behind a single pendulum to an n-stage pendulum. References [8, 9] outline the Mathematica and Matlab models of the quadruple pendulum.

2.1 End Test Mass (ETM) Requirements

Advanced LIGO specifies various noise limits at 10 Hz and above for the end test masses (ETMs). The limit relevant to the topic of this thesis is technical noise, or noise theoretically reducible given enough time and resources, and accordingly will be the only one considered here.

The technical noise requirements limit the motion of the ETM in all six degrees of freedom (DOFs), with obviously the strictest limit on the DOF along the axis of the interferometer. The requirements at 10 Hz are listed in Table 2.1 below.

DOFs	Noise at 10 Hz (m/√Hz or rad/√Hz)
Longitudinal	10^{-19}
Transverse	10^{-17}
Vertical	10^{-16}
Yaw	10^{-17}
Pitch	10^{-17}
Roll	$2.51*10^{-16}$

Table 2.1: Noise Requirements for the ETM Quadruple Pendulum, relative to inertial space [10].

These limits are set with the requirement that the noise must roll off as f^{-2}. In other words, at 100 Hz the noise must be 100 times lower than at 10 Hz. The optic is also required to settle from an impulse to 1/e of the maximum displacement in 10 s [10]. A description of the DOFs is given in section 2.2.

2.2 Quadruple Pendulum ("Quad") Design

To meet the technical requirements above plus other mechanical isolation requirements the mirrors of the Advanced LIGO interferometers are to be suspended as the last mass of a quadruple pendulum "quad" (see the Figure 2.1). The quad's parameters are defined in the Matlab code of Appendix A.

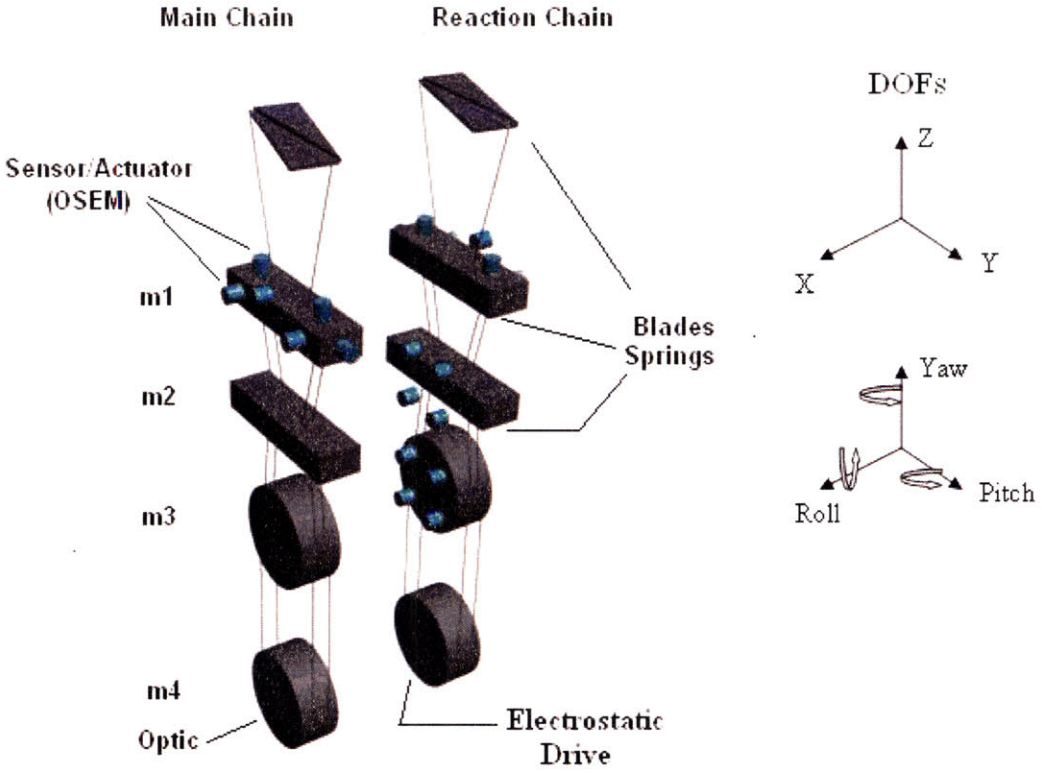


Figure 2.1: Diagram of the quad pendulum with the coordinate system. The quad consists of two chains, a main chain and a reaction chain. The bottom mass of the main chain is the interferometer optic. Three stages of blade springs provide vertical isolation. Sensor actuator devices (OSEMs) provide active damping and control. The reaction chain is used as a quiet actuation surface to reduce noise injected by the OSEMs. The quad parameters are defined in the Matlab code of Appendix A.

The suspension consists of two vertical chains of four masses. The masses on each chain are numbered 1 through 4, where the fourth mass on the main chain functions as the optical component. This optic is the mirror that will reflect the laser light back towards the ITM. The top two masses are approximately 20 kg and the lower two masses are approximately 40 kg. Each stage of the pendulum provides f^2 isolation above the pendulum's resonant frequencies. Thus, by using four masses a performance of f^8 is achieved. The choice of the number of masses is purely a function of the noise isolation requirements and simpler one, two, and three mass suspensions are used when the requirements permit.

To limit the thermal noise contributions to narrow frequency bands, the quality factors (Qs) of the resonances must be very high. In other words, the passive damping

must be very low. Small Qs, or high passive damping, have large thermal dissipation, which produce thermal noise [2]. Because of these high Q resonances, the pendulum will have a large amount of motion at low frequencies and the interferometric cavity will be impossible to lock. However, a way to circumvent this damping/thermal noise problem is active damping. The motion of the pendulum is sensed and actuators respond by pushing on the pendulum in a way such that its motion is reduced enough to lock the cavity [11].

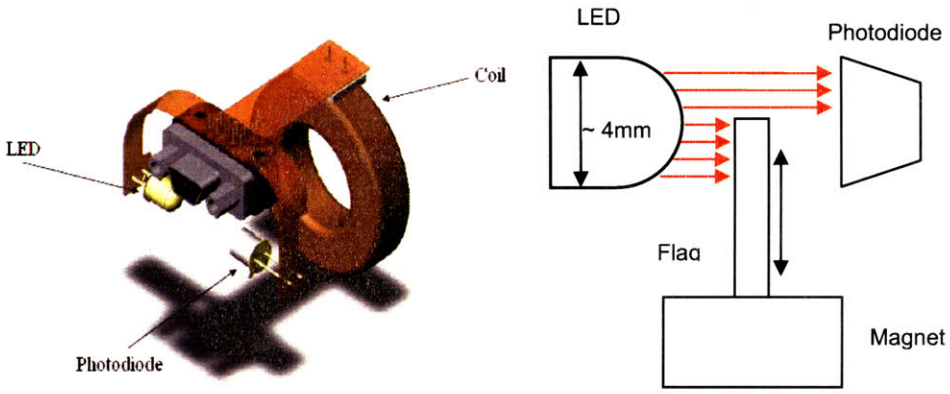


Figure 2.2: A drawing of the working parts of an optical sensor electromagnet (OSEM). The OSEM consists of an LED, photodiode, and coil of wire. A flag mounted to a mass on the quad blocks part of the LED light and produces a position dependent signal from the photodiode. When a current is run through the coil an actuation force is produced on a permanent magnet mounted under the flag. Adapted from [12, 13].

Each chain has sensor/actuator devices called OSEMs (Optical Sensor Electro-Magnet) to control the upper three masses. Figure 2.2 provides a diagram of this device. Motion of the masses is sensed with a shadow sensor. Each OSEM comes equipped with an LED and photodiode. The light emitted from the LED creates a current at the photodiode. A flag and permanent magnet mounted on the mass to be controlled sit between the LED and photodiode, partially blocking the light. The position of the mass can then be determined by the voltage of the photodiode. Running a current through a coil of wire surrounding the magnet (and hence LED and photodiode) gives the actuation. The current through this coil creates a magnetic field that pushes on a permanent magnet mounted behind the flag, and thus the mass [14].

The quad is also the only suspension to use a reaction chain. The reason for this is to provide a seismically isolated surface to actuate against. If the OSEMs were simply mounted to a fixed structure, vibrations of the structure would couple to the pendulum through the OSEMs.

Also benefiting from the reaction chain is an electrostatic drive on the bottom stage. This drive works by placing a high voltage static charge on the surface of the reaction chain bottom mass, also made of fused silica. The charge produces an electrostatic force that acts on the optic. With this method it is possible to actuate the optic without mounting any hardware on it. The idea behind an optic with no physical attachments is yet another noise reduction scheme. The optic is made of fused silica, a very low-loss material used to minimize thermal noise. Any type of glue or mounting scheme would increase the loss of the optic and thereby increase thermal noise.

The quad is controlled with two methods, each designed for a specific purpose. The first, called *local control*, is used to damp out the resonant motion. This is the control scheme that is the focus of this thesis. Each OSEM has a noise floor 10^9 m/$\sqrt{\text{Hz}}$ greater than the requirement of LIGO at frequencies above 100 Hz. In order to damp the resonant motion while minimizing the amount of OSEM sensing noise injected, only the OSEMs on the upper masses are used. In this way, much of the sensor noise is filtered by the four stages of the pendulum.

Eddy current damping is also used on the top mass to aid the local control. This passive damping is also limited to the uppermost mass to limit thermal noise injection. By accepting a small amount of thermal noise, however, sensor noise injection can further be reduced since the control loop gains can be lowered.

The second form of control is called *global control*. This method is used to fix the position of the bottom mass relative to the other optics of the interferometer. Actuation on the three lower masses is available through the remaining OSEMs and electrostatic drive. The interferometer signal itself is used in the feedback, which is much quieter than that of the OSEMs [11].

2.3 Characterization and Parameter Fit

In order to design efficient control loops it is necessary to understand the dynamics of the pendulum. A state space model of the pendulum was created in Matlab® (refer to Appendix A for the code), but it must be verified to confirm that it is within acceptable limits for design purposes. Here we will assume a match of 1% on the resonant frequencies is acceptable.

2.3.1 Characterization

The measurements of the pendulum were done using a swept sine for each DOF at the top mass. Figure 2.3 shows the location of the OSEMs and their names on the top mass which were used for the sensing and actuating of the system identification. The OSEMs are placed such that certain combinations of signals create Cartesian signals. For example, summing the right and left OSEM signals (sensing or actuating) creates a vertical signal. Subtracting face 2 from face 3 creates a yaw signal.

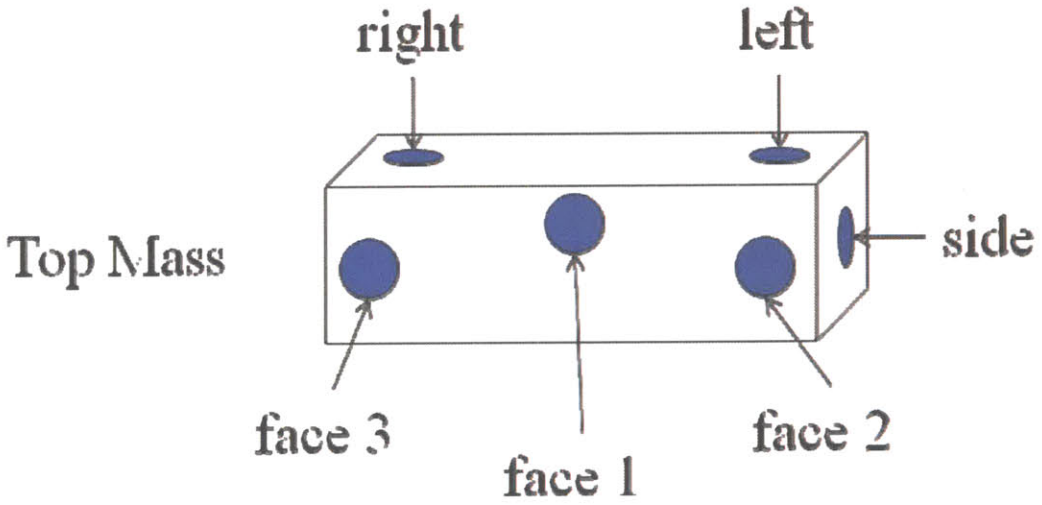


Figure 2.3: The location of the OSEMs around the top mass with their signal names. The OSEMs are placed such that certain combinations create the Cartesian signals, x, y, z (vertical), yaw, pitch, and roll.

Below, Figures 2.4 and 2.5 show respectively the yaw and vertical measured transfer functions, along with the expected transfer function from the Matlab model. The

model curve (red) is artificially shifted upwards for clarity of shape. All measurements are collocated, meaning that the same OSEMs actuate and sense for each measurement.

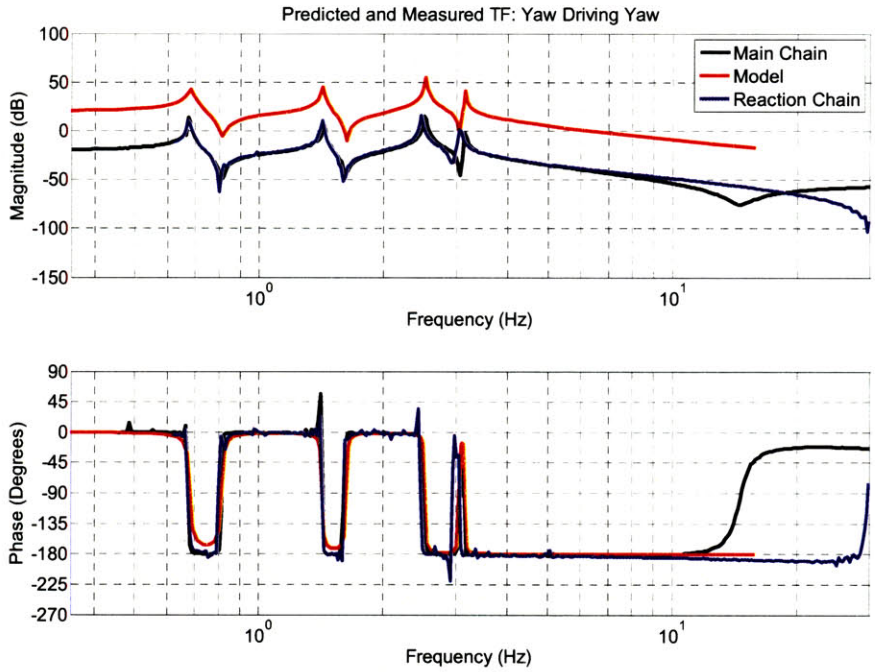


Figure 2.4: Measured and predicted transfer functions of the yaw DOFs using the top stage OSEMs. The peaks are the resonant frequencies of the yaw DOFs. The model (red) is artificially shifted upwards for clarity. These yaw measurements were created by subtracting the face 2 OSEM signals from the face 1 OSEM signals.

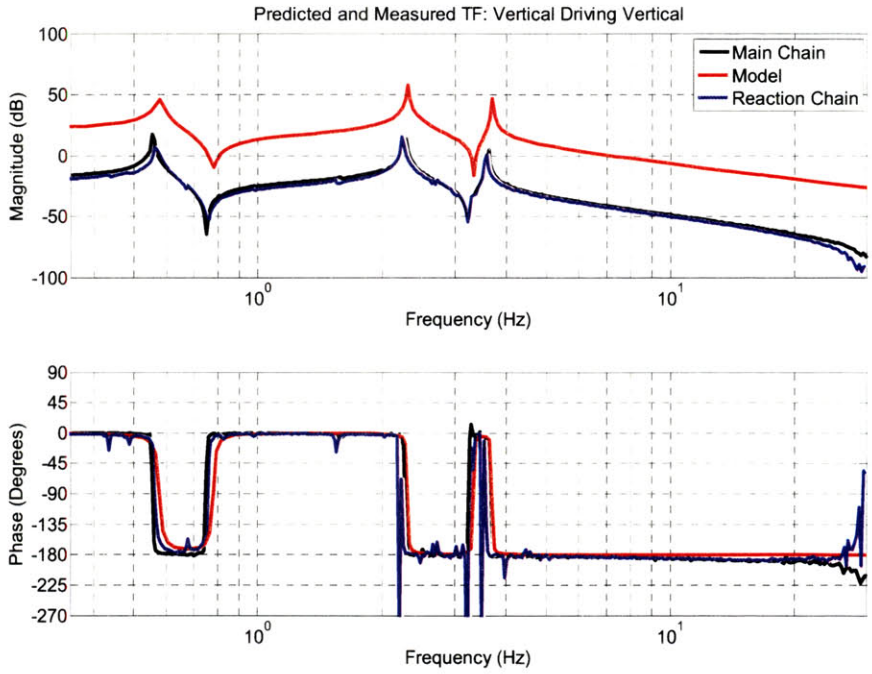


Figure 2.5: Measured and predicted transfer functions of the vertical DOFs using the top stage OSEMs. The peaks are the resonant frequencies of the vertical DOFs. The model (red) is artificially shifted for clarity. These vertical measurements were created by summing the left and right OSEM signals.

The deviations between the measurements and model above 10 Hz are due to electronic pick up due to insufficiently shielded cabling. This coupling is believed to be tolerable because it occurs above the important dynamics of the transfer functions. The prediction is in general a good fit, but for a more accurate look at just how close it is, Table 2.2 compares the measured values of the resonant frequencies to the model results.

Yaw (Hz)						Vertical (Hz)				
Model	Main	%err±1	React.	%err±1		Model	Main	%err±1	React.	%err±1
0.685	0.674	-1.61	0.674	-1.61		0.581	0.552	-4.99	0.5654	-2.69
1.429	1.424	-0.35	1.425	-0.28		2.325	2.278	-2.02	2.251	-3.18
2.536	2.504	-1.26	2.474	-2.44		3.733	3.613	-3.21	3.613	-3.21
3.165	3.135	-0.95	3.061	-3.29		17.332	17.201/17.418		-0.76/0.50	

Table 2.2: The error between the measured and predicted resonant frequencies. The headings 'Main' and 'React.' refer to the main and reaction chains, respectively. The '%err±1' column is the percent error between the model and measurements at that frequency. The ±1 is the percent uncertainty in the measurement. The 17.332 Hz vertical frequency does not distinguish between the main and reaction chain since the OSEMs at that stage measure only relative displacement.

Not all the predictions are within 1% so the model parameters will have to be refined to reduce this error. A good fit of the model is in fact extremely important to the control scheme discussed in this thesis. The reasons for this are discussed in Chapter 4.

2.3.2 Parameter Fit

As stated in the previous section the model does not predict the dynamics of the pendulum quite as well as we would like. In order to achieve better predictions, a fit of the model is made in Matlab (refer to Appendix B for the code). The physical parameters of the pendulum that affect these modes the most; primarily mass, stiffness, and moment of inertia are altered using a gradient descent method. The error is calculated by comparing the measured peak frequencies to the predicted peak frequencies. For the yaw and vertical DOFs the result is Table 2.3 below.

Yaw (Hz)				Vertical (Hz)		
New Model	Main	%err±1		New Model	Main	%err±1
0.680	0.674	-0.94		0.557	0.552	-0.91
1.424	1.424	0		2.261	2.278	-0.75
2.504	2.504	0		3.613	3.613	0
3.135	3.135	0		17.388	17.201/17.418	-1.09/0.17

Figure 2.3: Error between the measured and best fit predicted resonant frequencies. All the errors are less than the desired 1% margin.

All the frequencies are now within the desired limit of 1%. The one possible exception is the 17.388 Hz vertical frequency. However, this mode is not possible to distinguish between the two chains since its motion is confined to the bottom two masses where the OSEMs only measure relative displacement between the chains. As a result, the frequencies of each chain can be measured, but they are indistinguishable. Moreover, exact predictions for this mode are not necessary, since it is neither measurable nor controllable from the top mass in the first place. Simulations indicate that errors of these magnitudes should not have a significant effect on the performance of the modal control and state estimation presented later.

Chapter 3

Classical Control

A common way for designing control loops is to design a filter by placing poles and zeroes using the Nyquist stability criterion in such a way that the system's performance will meet the specifications. This process can be lengthy and time-consuming depending on the severity of the requirements. The quadruple pendulum is no exception. It must be well controlled up to 5 Hz to damp the resonances while still being allowed to swing freely at 10 Hz for noise isolation. A stable and robust 'optimized' solution is not necessarily obvious here using these Nyquist criterion. However, in this chapter we use this pole-zero placing method in order to create a bench mark for the performance of work presented in this thesis. Figure 3.1 is a block diagram schematic of the classical control simulations used in this chapter.

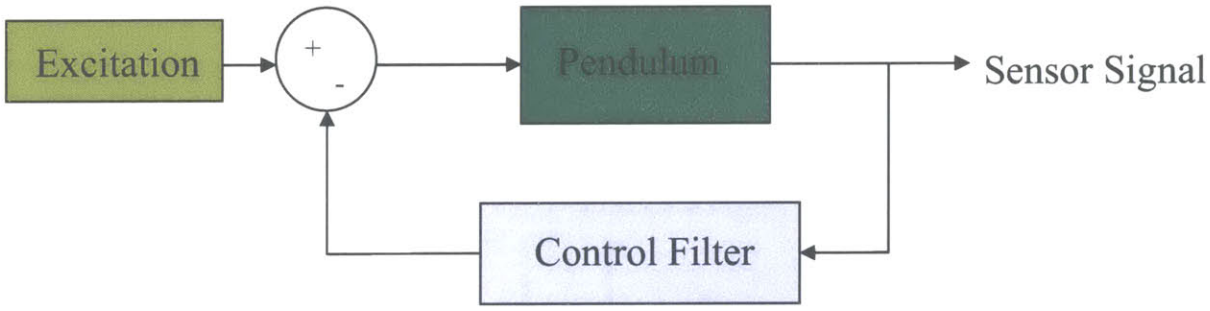


Figure 3.1: A block diagram schematic of the classical control simulations used in this chapter.

We begin the control design with 'optimized' filters that were designed for the triple pendulum [13] and modify them for the quad (Refer to Appendix C for the Matlab code). Figures 3.2 and 3.3 show Bode plots for the yaw and vertical systems

respectively. The filters (green curves) are plotted along with the pendulum (plant, blue curves) and open loop transfer functions (red curves) in Figures 3.1 and 3.2 below. The open loop transfer function is the combination of the plant and controller transfer functions and is used to determine stability margins.

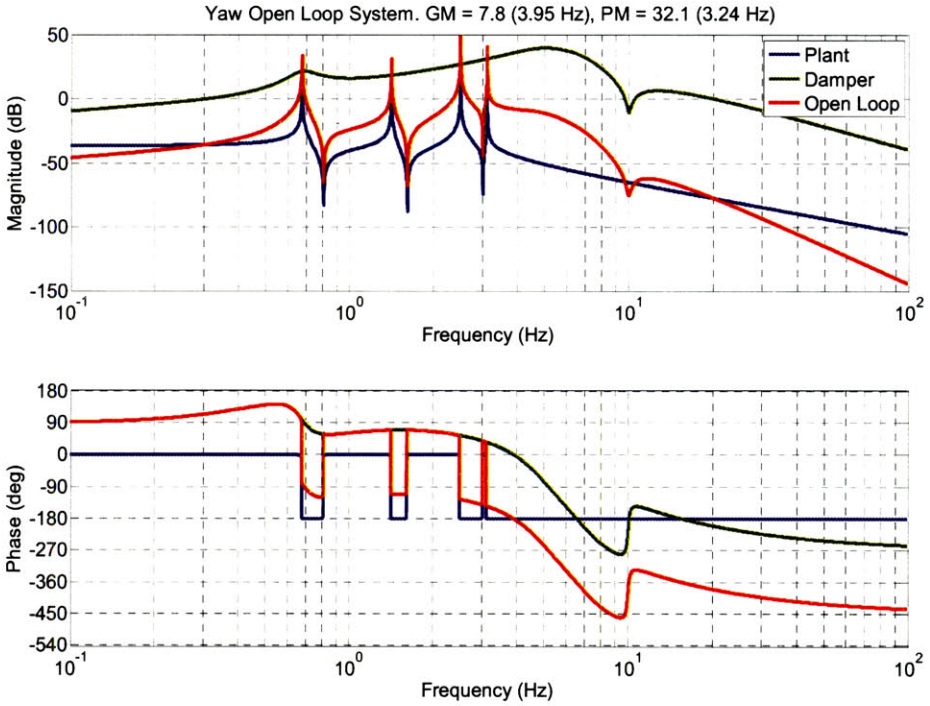


Figure 3.2: Classical control of the yaw DOFs. The yaw transfer function (blue) is plotted with the controller (green) and the open loop system (red).

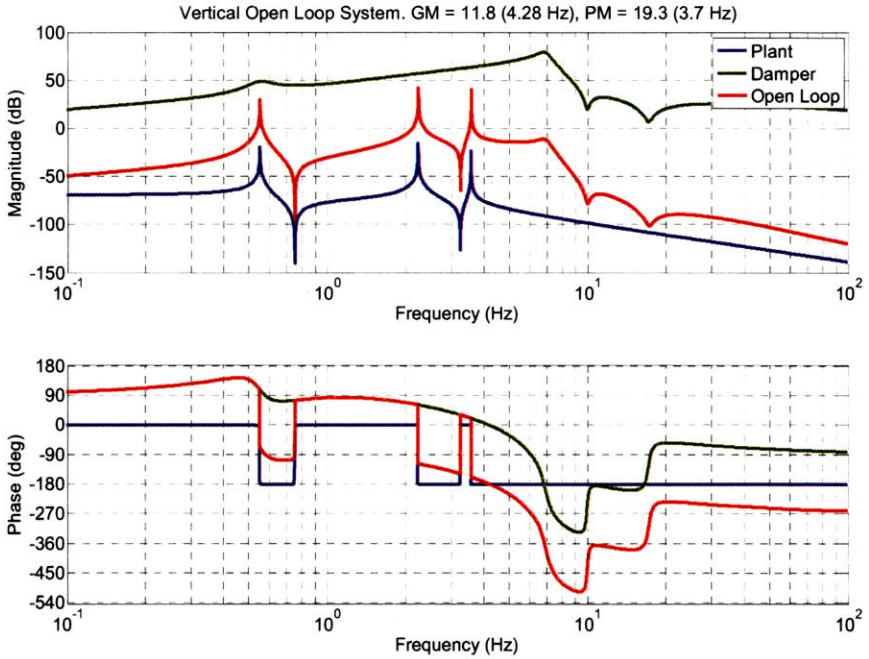


Figure 3.3: Classical control of the vertical DOFs. The vertical transfer function (blue) is plotted with the controller (green) and open loop system (red).

The simulated settling and noise performances are then,

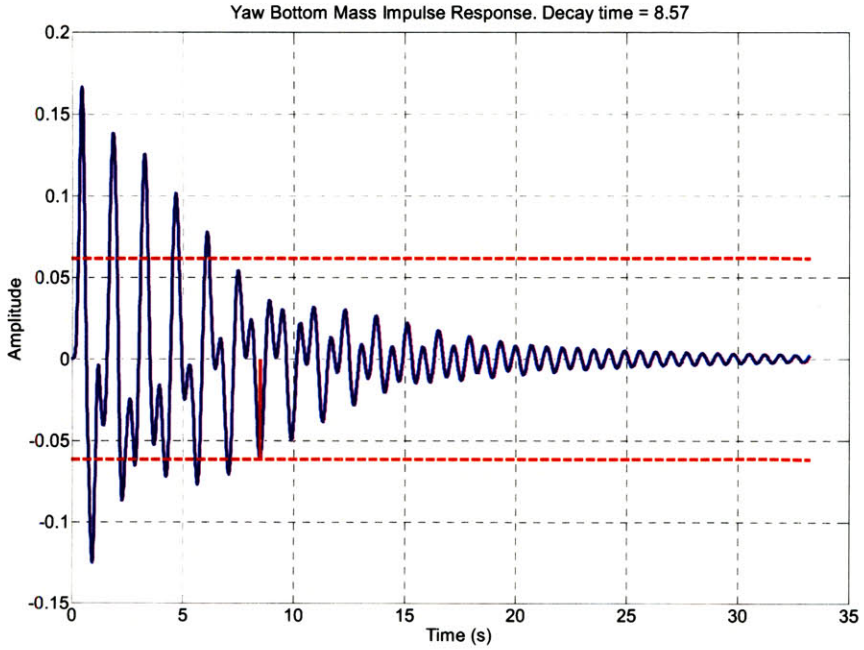


Figure 3.4: The settling performance of the yaw classical control. The red lines are placed at ±1/e of the maximum response. The system settles in less than the required 10 s.

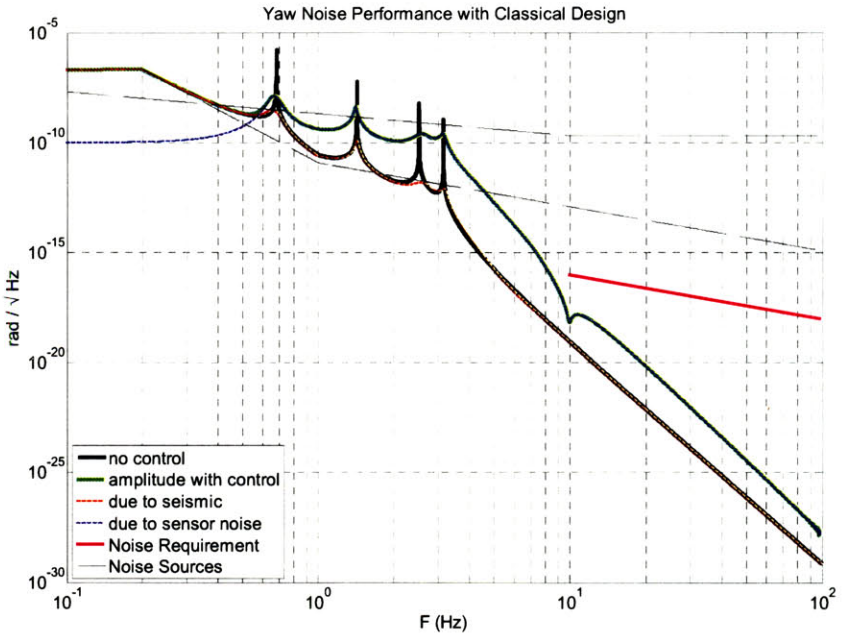


Figure 3.5: Bottom mass noise performance of the yaw classical control. The controller meets the noise requirement. The black curve is the uncontrolled bottom mass yaw response. The green curve is the controlled response and the magenta is the requirement. The dashed lines are the seismic and sensor noise sources.

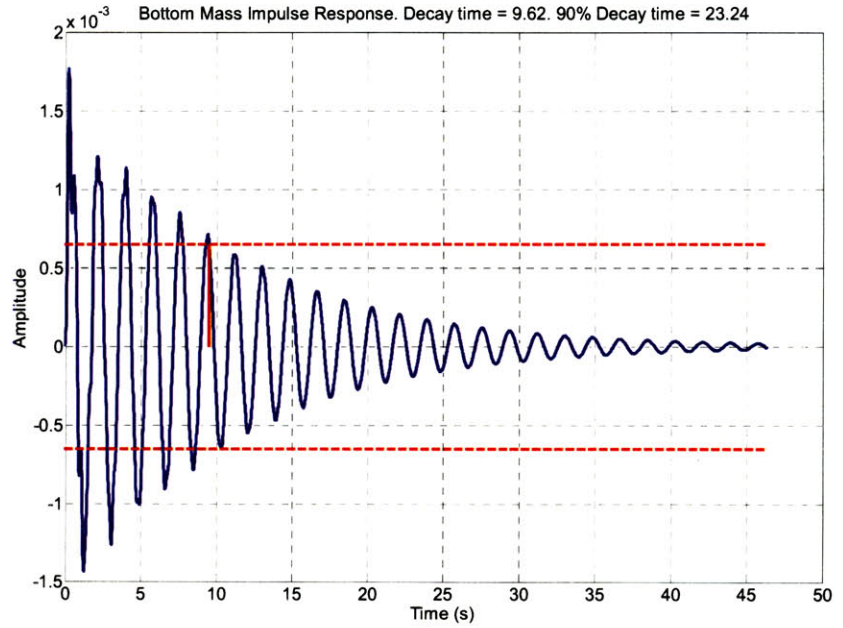


Figure 3.6: Settling performance of the vertical classical control. The red lines are placed at ±1/e of the maximum response. The system settles in less than the required 10 s.

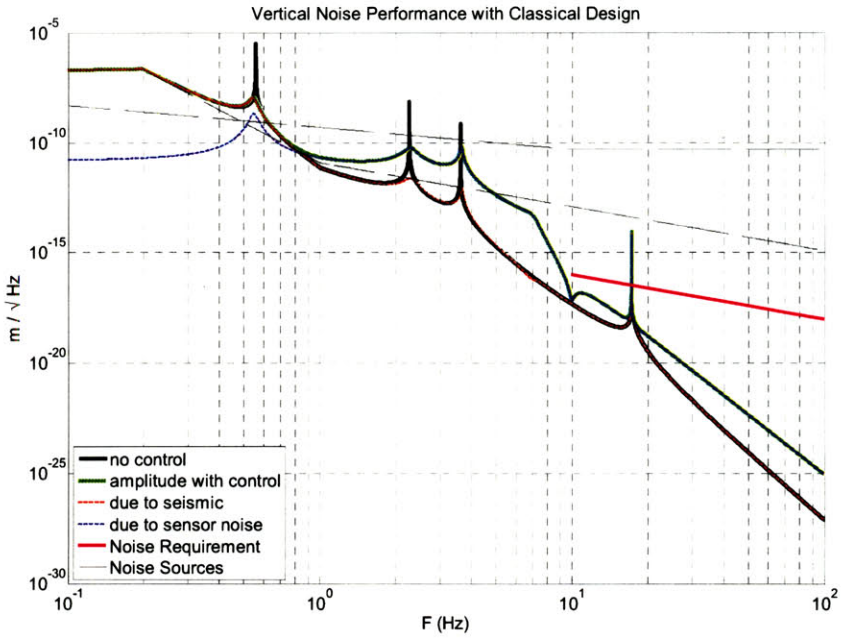


Figure 3.7: Bottom mass noise performance of the vertical classical control. The black curve is the uncontrolled bottom mass vertical response. The green curve is the controlled response and the magenta is the requirement. The dashed lines are the sensor and seismic noise sources. The controller meets the noise requirement except at the 17.3 Hz mode which is not controllable from the top stage. This 17.3 Hz mode is strictly motion of the bottom two masses which is also why it is not visible on the system identification transfer functions.

Figures 3.4 to 3.7 show that these filters do meet the requirements, with the exception of the vertical, which has a mode at 17.3 Hz that exceeds the requirement. This mode is strictly motion of the bottom two masses and can neither be damped nor sensed from the top mass. Since it is well known and defined at a specific frequency this exception can be tolerated. Thus, we could be finished here. However, we want to establish a straight forward and, if possible, better performing method for damping the modes of the suspensions.

Chapter 4

Modal Control with SIMO State Estimation

The modal control with state estimation theory presented in this chapter and chapter 6 is an extension of the work done by Ruet on LIGO's triple suspension system [13]. This chapter discusses the application of the theory to the quad's yaw and vertical DOFs since each of these DOFs is decoupled from the rest of the dynamics of the quad. The remaining DOFs are left for chapter 6 since they are coupled and cannot be separated into smaller systems.

The state estimation here is termed SIMO (Single Input Multi Output) since it estimates four modes from one sensor input. Technically, there are multiple inputs since the control forces are also passed to the estimator, but this terminology allows us to easily distinguish between the estimators here and the estimator for the larger coupled system which receives four sensor signals.

The advantage of modal control is that it allows the pendulum to be mathematically broken down into a series of second order single DOF systems resonant at the pendulum's modes. Then it is sufficient to use one simple design of a control filter and adapt it to each of these individual one DOF systems. The gains on each filter can be tailored to provide just enough damping so that noise filtering is maximized. Since the lower frequency modes inject the least sensor noise and, in general, contribute the most to settling time, the gains can be set to provide sufficient damping while optimizing the noise injection.

4.1 Modal Decomposition

The mathematics of modal decomposition listed here can be found in Kausel [15]. First we include the equations of motion in matrix form. M is the mass matrix, K the stiffness matrix, and x the state vector of the motion of each mass.

$$M\ddot{x} + Kx = 0 \tag{4.1}$$

Assuming a complex solution of x with no damping and ϕ as the mode shape column vector we get

$$x = \phi e^{i\omega t} \tag{4.2}$$

If we plug the solution to x (Eqn. 4.2) into the equation of motion (Eqn. 4.1) and cancel $e^{i\omega t}$

$$\omega^2 M\phi = K\phi \tag{4.3}$$

Then, assuming, valid here, that the mass matrix M is invertible we can rewrite the system as an eigenvector problem where the mode frequency ω is the eigenvalue and the mode shape ϕ is the eigenvector.

$$M^{-1}K\phi = \phi\omega^2 \tag{4.4}$$

For a system with n modes

$$\Phi = \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_n \end{bmatrix} \tag{4.5}$$

$$\Omega = \begin{bmatrix} \omega_1^2 & & & 0 \\ & \omega_2^2 & & \\ & & \cdots & \\ 0 & & & \omega_n^2 \end{bmatrix} \tag{4.6}$$

$$M^{-1}K\Phi = \Phi\Omega \tag{4.7}$$

Φ is then the basis of eigenvectors of the system and Ω is the vector of the square of the mode frequencies. The physical interpretation of the eigenvectors is that they give the mode shapes, which provides information on how each mass moves at each resonant frequency during free vibration.

We can transform the original equation of motion into a decoupled system of modal states by transforming the state vector x into the eigenvector basis q.

$$x = \Phi q \tag{4.8}$$

$$M\Phi\ddot{q} + K\Phi q = P \tag{4.9}$$

$$\Phi^T M\Phi\ddot{q} + \Phi^T K\Phi q = \Phi^T P \tag{4.10}$$

$$M_m\ddot{q} + K_m q = P_m \tag{4.11}$$

Here q is the modal state, M_m the diagonal modal mass matrix, and K_m the diagonal modal stiffness matrix. The introduction of P merely accounts for the control forces used in damping. Since the modal mass and modal stiffness matrices are diagonal, the result is simply a list of n independent equations, where n is the number of degrees of freedom. It is interesting to note, and fundamental to the success of this method, that the 'real' state x is nothing more than a weighted sum of the mode shapes, independent of whether the pendulum is driven or in free vibration.

4.2 Modal Control

For application to a real system there are two important transformations to keep in mind.

$$q = \Phi^{-1}x \tag{4.12}$$

$$P = (\Phi^{\mathrm{T}})^{-1}P_m \tag{4.13}$$

The first, Eqn. (4.12), transforms the sensor signals into the modal state. Control can then be applied on the basis of these new modal signals, which result in the modal actuation forces P_m. The second transformation, Eqn. (4.13), converts the modal forces into real forces that can be applied to the pendulum at the sensor locations.

A simple graphical representation of this process is provided in Figure 4.1.

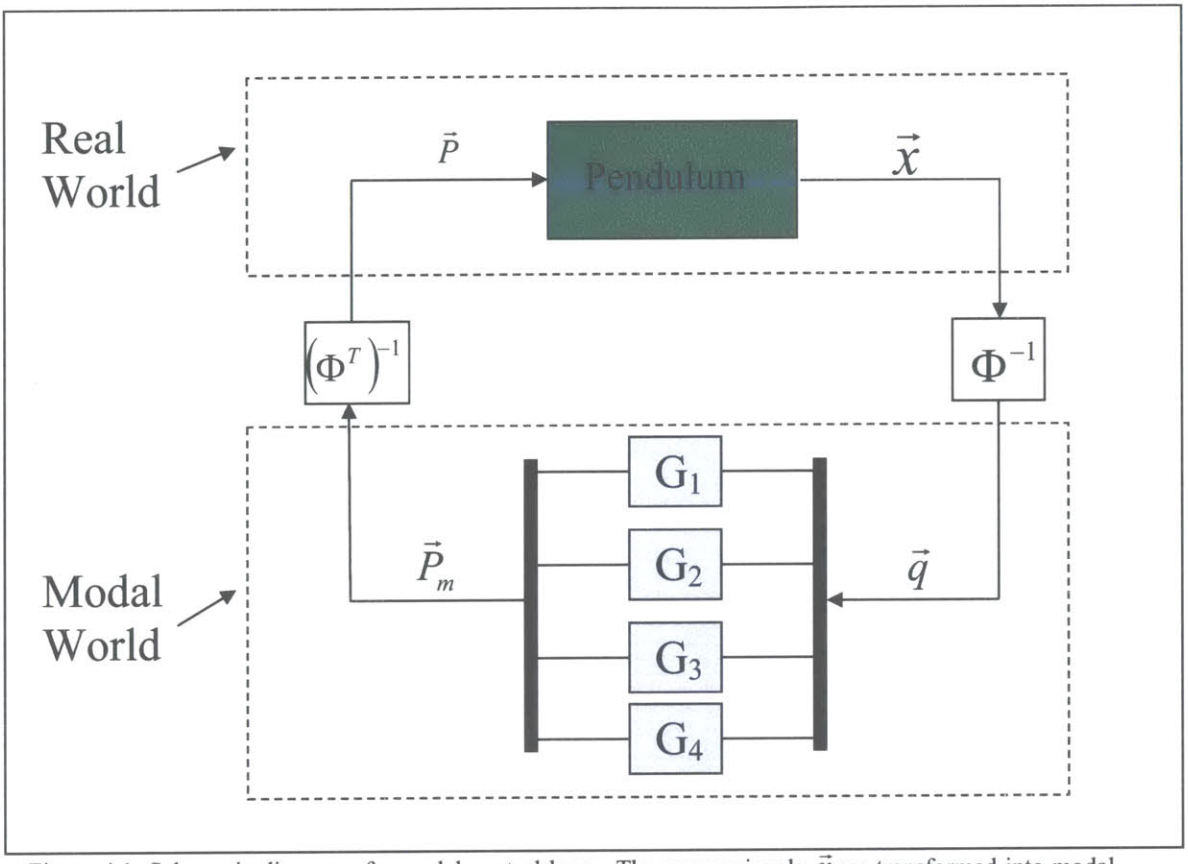


Figure 4.1: Schematic diagram of a modal control loop. The sensor signals $\vec{x}$ are transformed into modal signals $\vec{q}$ and passed through their respective control filters G_i. The resulting modal forces $\vec{P}_m$ are then transformed into real forces $\vec{P}$ that are applied to the pendulum.

This figure has four control filters and thus illustrates the process for a four DOF system. However, it is clear that the process easily extends for an n DOF system. The break out of the control filters is simply to illustrate the independent nature of the controllers on each mode.

The design of the control filters themselves is very straightforward. We first assume perfect decoupling of all modes and consider a single DOF oscillator as the plant (see Eqn. 4.14). The filter design accounts for three factors: (i) they must not impose a DC offset; (ii) they must damp in a reasonable amount of time; and (iii) they must minimize sensor noise injection above 10 Hz.

$$P = \frac{Oscillator}{Excitation} = \frac{\omega_0^2}{k_m\left(s^2 - \omega_0^2\right)} = \frac{1}{m_m(s^2 - \omega_0^2)} \tag{4.14}$$

$$F = \frac{G[\frac{1}{\omega_0^2}s^2 + \frac{10}{3\omega_0}s + 1]s}{[\frac{1}{\omega_0^2}s^2 + \frac{2}{3\omega_0}s + 1][s^2 - 0.684\omega_0 s + 4\omega_0^2]} \tag{4.15}$$

Eqn. (4.14) is the transfer function for a single DOF modal oscillator. ω_0 is the resonant frequency, k_m is the modal stiffness, and m_m is the modal mass. Eqn. (4.15) is the transfer function of the modal controller. G is a variable gain parameter that can be used to adjust the damping performance of the filter. The polynomial in the square brackets of the numerator and the first polynomial in the square brackets of the denominator create a gain bump right at the resonant frequency, with no loss of phase at this frequency, in order to improve damping performance. The second coefficient in these polynomials defines the height and width of the bump. The second polynomial in square brackets in the denominator is centered at twice the resonant frequency in order to roll the gain off to minimize sensor noise injection. The second coefficient of this polynomial defines the angle of the complex pole pair in the complex plain. The single s in the numerator is the zero at 0 Hz that AC couples the filter so that no DC offset exists.

Figure 4.2 has normalized Bode plots of the single DOF oscillator and control filter shown above (Eqns. 4.14 and 4.15). Figure 4.3 shows the open loop Bode and root locus plots.

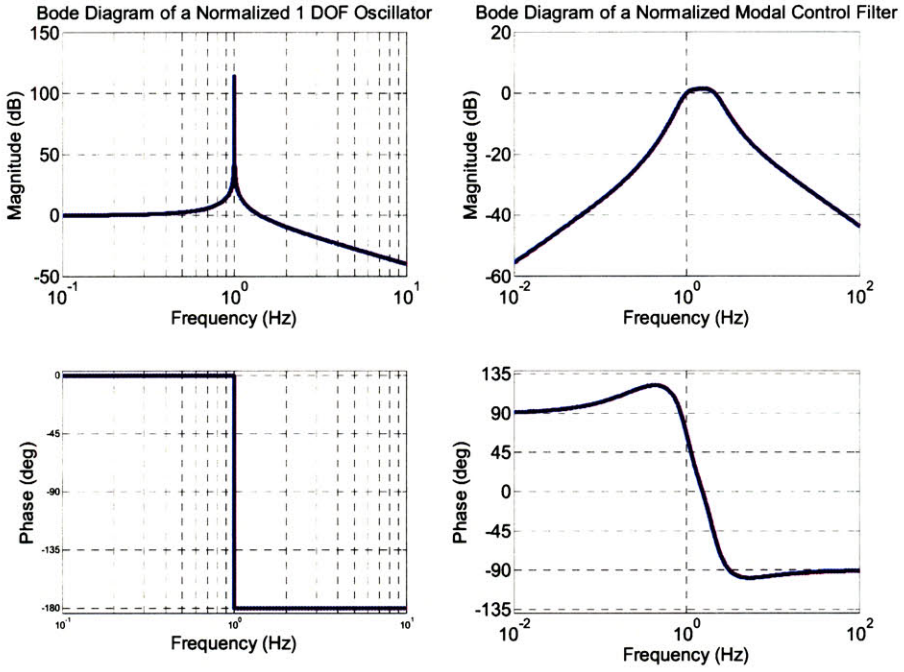


Figure 4.2: Left, a modal plant as a simple oscillator normalized to 1 Hz. Right, a standard modal control filter normalized to damp at 1 Hz.

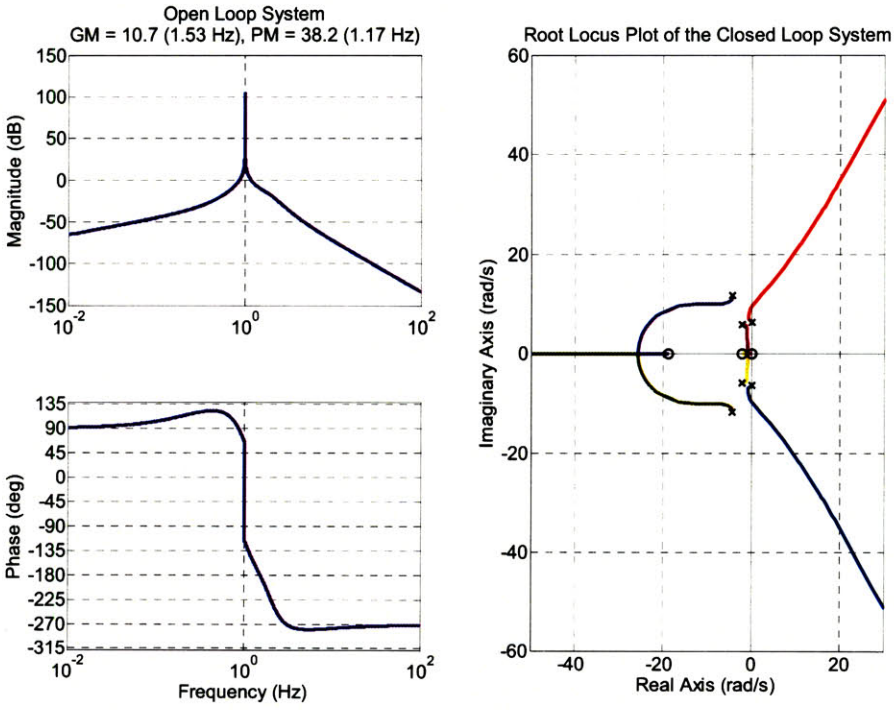


Figure 4.3: Left, the open loop modal control Bode plot with a gain of 1/3. Right, the root locus plot of the modal control. In this case a gain of 1/3 (set by G in Eqn. 4.15) appears to be a good starting place in terms of stability and damping.

The control filter does indeed roll off to zero gain at both extremes of the spectrum. It is parameterized to the frequency of the mode which it damps so that all filters will have identical shapes and merely be shifted left, right, up or down. Figure 4.3 indicates that for such a system an overall gain near 1/3 at the resonant frequency (set by G in Eqn. 4.15) would provide fair margins of stability.

Running a simple impulse response simulation on the quadruple pendulum's yaw system illustrates the feasibility of the modal control idea.

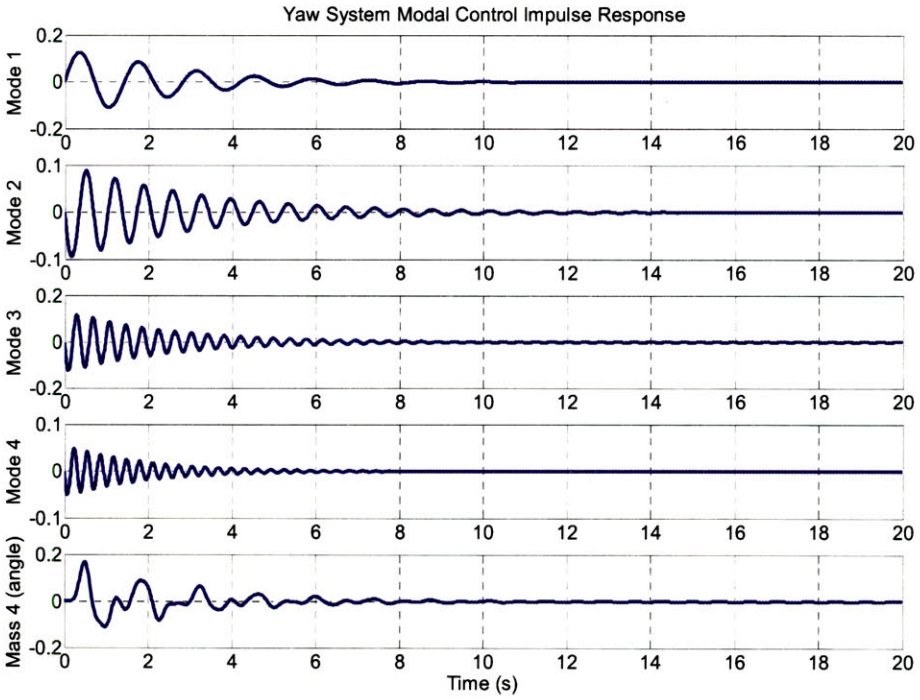


Figure 4.4: Modal control settling response of all 4 modes and the bottom mass to a yaw impulse on the top mass. The bottom mass response is a linear combination of the modal responses. All modes are independently damped systems. All units are arbitrarily normalized

Figure 4.4 shows the response of all four modes excited with an impulse on yaw at the top mass. Also shown is the response of the bottom mass, which is a linear combination of the four modal responses above it. The units are arbitrarily normalized. These responses were easily achieved by adjusting the gains on the four filters.

4.3 State Estimation

Note that the mathematics of modal control require the full state vector, or the positions of all four masses. However, only the top mass is observed. Fortunately, this fact does not rule out the feasibility of modal control. If the dynamics of the pendulum are sufficiently well known, as we already established earlier, the states of all 4 masses can be reconstructed using information from only one of them. This process is done with a state estimator, or observer.

4.3.1 Introduction to the Estimator

If the equations of motion of the pendulum are represented in state space form

$$\dot{x} = Ax + Bu \tag{4.16}$$

$$y = Cx \tag{4.17}$$

where A contains all the dynamic information, B modifies the inputs to the system, and C modifies the state vector x to provide outputs, then the estimator can be represented by the following form

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}) \tag{4.18}$$

The new state vector $\hat{x}$ is now the reconstructed estimated state, which will be used in feedback instead of the real state. The estimator receives two types of inputs. The first is simply the control applied to the real pendulum. The second is the measured part of the real state. The measurement is used to form an error signal between the real state and estimated state. The error signal is then multiplied by the gain L and used as a correction factor in the calculation of the estimated state. It now becomes clear why a mathematical model of the pendulum is necessary. The A and B matrices are required for calculating the estimator [16].

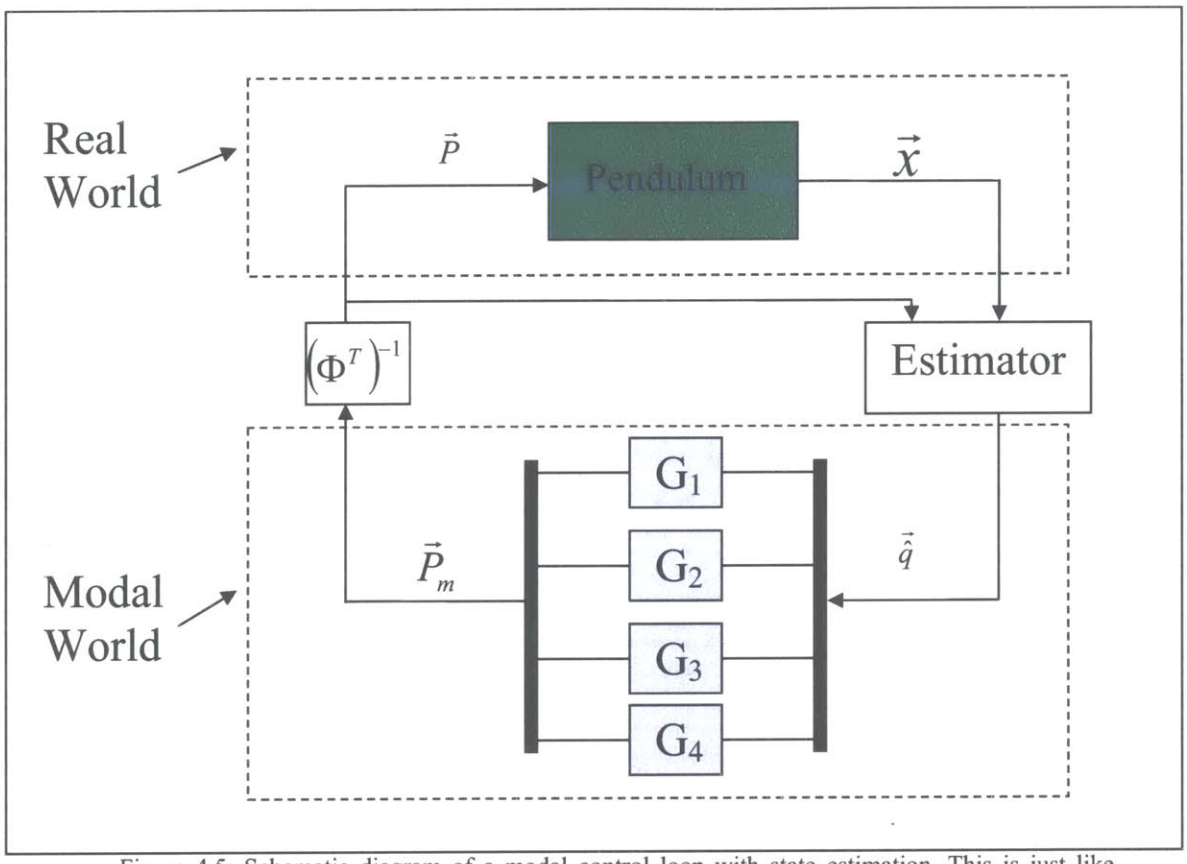


Figure 4.5: Schematic diagram of a modal control loop with state estimation. This is just like Figure 4.1 with the exception that the real to modal conversion is done with an estimator. The estimator also receives feedback from the control forces $\vec{P}$ for a more accurate modal reconstruction.

Figure 4.5 shows a schematic of how the estimator fits with the modal control. It replaces the Φ^{-1} term from Figure 4.1. Both the control forces and sensor signals are included as inputs to the estimator.

A fortunate feature of estimators is that they can be designed completely independently of the control loops. The poles (dynamics) of the control loop are simply the collection of poles of the modal control filters and the estimator. Thus, stability is guaranteed if both the modal control filters and estimator are stable individually, *assuming* the model is an exact representation of the plant. In practice this assumption is never completely true. In that case, a more accurate representation of the estimator might be

$$\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u + L(y - C\hat{x}) \tag{4.19}$$

where $\hat{A}$ and $\hat{B}$ are the model's approximation to the plant. The question of how good the model needs to be to use this stability decoupling assumption will be addressed later.

The model will soon be shown to be good enough, in which case the choice of L is not arbitrary. If L is too large then the estimator will contribute to sensor noise injection. Too small an L and the estimator will be useless in reconstructing the full state. To optimize this term we make use of standard linear-quadratic-regulator (LQR) techniques, which are also conveniently extendable to the more complex MIMO case.

4.3.2 Estimator LQR Optimization Technique

The standard LQR technique is a state space design tool defined for optimal feedback control. The formulation of the technique is readily available in many control theory books, so we briefly introduce it here in order to explain how it is used in relation to estimator design.

By restating Eqn. (4.16) and adding to it Eqn. (4.20)

$$\dot{x} = Ax + Bu \tag{4.16}$$

$$u = -Kx \tag{4.20}$$

we give the standard state space representation of a plant with state feedback control. The state x is to be controlled to zero. Then, we define a cost function

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt \tag{4.21}$$

The matrices Q and R must be positive definite to ensure J is greater than or equal to zero. Considering Q and R as parameters to weight the relative importance of the states and control forces, minimizing J will result in an optimal control law K for the chosen cost function, Eqn. (4.21). Larger values of Q will weight the solution to minimizing the controlled state x whereas larger values of R will weight the solution to minimizing the control forces u. LQR is thus very powerful when designing a system to optimize convergence of the controlled variable in the presence of such limits as actuator force. The mathematics behind the solution to this LQR problem are somewhat complicated but can be found in texts such Ogata [16].

Now we introduce the correlation to estimator design by defining the estimator error $\tilde{x}$.

$$\tilde{x} = x - \hat{x} \tag{4.22}$$

$$\dot{\tilde{x}} = \dot{x} - \dot{\hat{x}} \tag{4.23}$$

Plugging Eqns. (4.16), (4.17), and (4.18) into (4.23)

$$\dot{\tilde{x}} = Ax + Bu - A\hat{x} - Bu - L(Cx - C\hat{x}) \tag{4.24}$$

Canceling like terms and reducing with Eqn. (4.22)

$$\dot{\tilde{x}} = (A - LC)\tilde{x} \tag{4.25}$$

Eqn. (4.25) now states the dynamics of the estimator error. Conveniently this also proves that, under the assumptions of a perfect model A, B, and C, the estimator error dynamics are completely independent of the control inputs, which is why the two can be designed independent of one another. Since the eigenvalues of $A - LC$ are the same as $A^T - C^T L^T$ we can rewrite Eqn. (4.25) as

$$\dot{\tilde{x}} = A^T\tilde{x} - C^T L^T\tilde{x} \tag{4.26}$$

Finally, using a change of variable we obtain

$$\dot{\tilde{x}} = A^T\tilde{x} + C^T z \tag{4.27}$$

$$z = -L^T\tilde{x} \tag{4.28}$$

Now we have the estimator error dynamics in a form that resembles that of the state feedback control. The correlation between equation pair (4.16) to (4.20) and (4.27) to (4.28) illustrates the duality principle between control design and estimator design where

$$A \rightarrow A^T \tag{4.29}$$

$$B \rightarrow C^T \tag{4.30}$$

$$K \rightarrow L^T \tag{4.31}$$

Thus, imagining the estimator as a controller and plugging Eqn. (4.29) into (4.31) appropriately we can solve for an optimum gain L using LQR with the cost function

$$J = \int_0^\infty (\tilde{x}^T Q\tilde{x} + z^T Rz)dt \tag{4.32}$$

This cost function will work. However, since we are damping modes individually it makes more sense to convert the estimator into a modal form and optimize it for individual modes. In this case we go back to the estimator Eqn. (4.18) and perform the modal transformation on the estimated state.

$$\Phi\dot{\hat{q}} = A\Phi\hat{q} + Bu + L(C\Phi q - C\Phi\hat{q}) \tag{4.33}$$

$$\dot{\hat{q}} = \Phi^{-1}A\Phi\hat{q} + \Phi^{-1}Bu + \Phi^{-1}LC\Phi\tilde{q} \tag{4.34}$$

$$A_m = \Phi^{-1}A\Phi \tag{4.35}$$

$$B_m = \Phi^{-1} B \tag{4.36}$$

$$C_m = C\Phi \tag{4.37}$$

$$L_m = \Phi^{-1} L \tag{4.38}$$

$$\dot{\hat{q}} = A_m \hat{q} + B_m u + L_m C_m \tilde{q} \tag{4.39}$$

The subscript m indicates the modal matrices. We can also do the same for the plant state space equations and achieve

$$\dot{q} = A_m q + B_m u \tag{4.40}$$

Performing similar operations on Eqns. (4.27) and (4.28) we get

$$\dot{\tilde{q}} = A_m^T \tilde{q} + C_m^T z_m \tag{4.41}$$

$$z_m = -L_m^T \tilde{q} \tag{4.42}$$

The cost function now becomes

$$J = \int_0^\infty \left(\tilde{q}^T Q \tilde{q} + z_m^T R z_m \right) dt \tag{4.43}$$

This is a cost function that makes sense and is straightforward to use. Since it is the relative proportions of the parameters that matter it follows that large values of Q force $\tilde{q}$ to be small but allow for higher values of z_m. Since z_m is proportional to both L_m and $\tilde{q}$, which is already small, then the solution of L_m must be very large in this case. Realizing that L_m is a direct gain on the sensor readouts implies that there will also be more noise injection. Conversely, large values of R force z_m to be small and allow $\tilde{q}$ to be large. Thus, in this case, L_m is small, limiting sensor noise.

Making Q and R diagonal with positive entries ensures they are positive definite and keeps the individual modes and measurements independent from one another. Individual modes and measurements can also be weighted against one another. In the MIMO case this feature is the most important since we have four measurements with sixteen modes. The cost function then allows the user to quantitatively choose which modes and measurements are most important in terms of settling time and noise injection.

4.4 Applying Modal Control and SIMO Estimation to the Quad

Now that all the tools for the modal control and state estimation are laid out, it is time to implement them on the quadruple pendulum in order to meet the specifications for Advanced LIGO.

4.4.1 Modal Control with Incomplete Actuation

First, however, one more non-ideality must be addressed. The observant reader will have realized a second non-ideality with the modal control overlooked up to this point. Not only do we sense just the top mass, but we actuate on just the top mass. The sensing problem was overcome with an estimator, but how can we properly damp the modes individually if we cannot push on each mass with the right proportions of force?

It turns out that the modes only need to be coupled strongly enough to the top mass for this method to work. For example, assume the pendulum is vibrating at the fundamental vertical mode, so that all the masses are oscillating up and down together. If a damping force is applied at only the top mass, the energy of this motion can still be removed even without actuating on the others. If the top mass were stationary while all the others were oscillating clearly this would not be the case. Also, the damping filters are designed to have high gain specifically at the frequency of the modes they are damping so cross coupling to other modes will be minimized.

Plots are included below for yaw that prove that damping still works and that cross coupling to other modes is only a second order effect. Here the estimator is ignored, for the moment, to focus just on the aspects of modal control. Both cases include an impulse on just the first mode. Modes two, three, and four are also zoomed in an order of magnitude relative to the first to make the point visible.

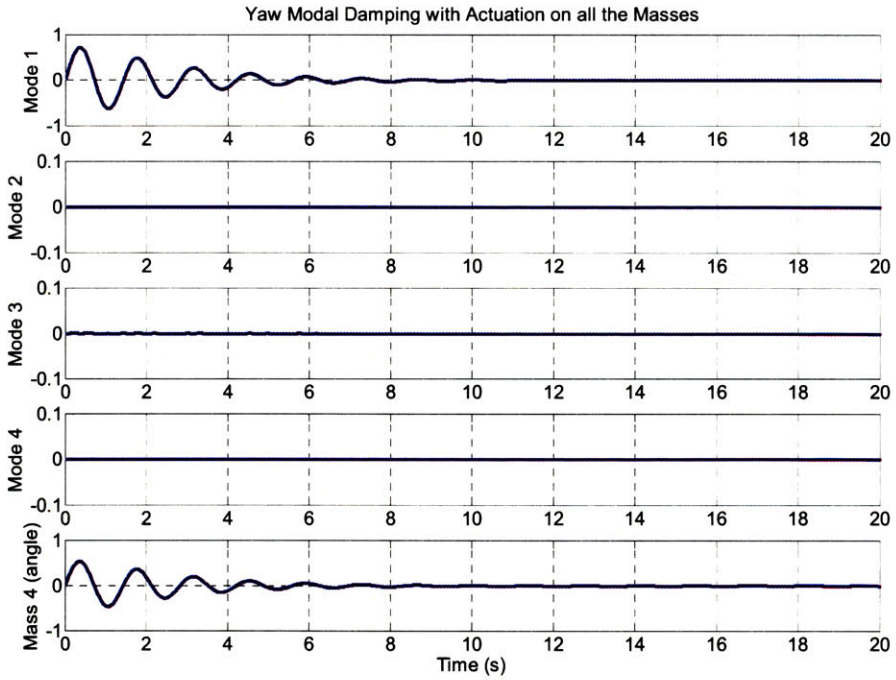


Figure 4.6: Impulse to mode 1 with full observation of each mode and actuation on each mass. Only mode 1 has a nonzero response. All units are arbitrarily normalized.

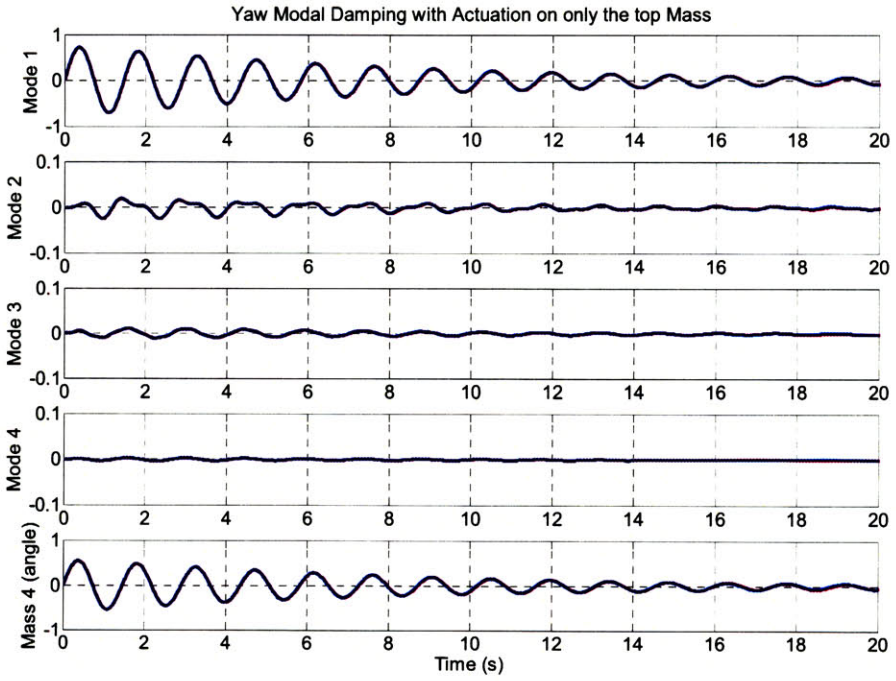


Figure 4.7: Impulse on mode 1 with full observation of each mode but actuation on only the top mass. The other modes respond with at most 5% of the amplitude of mode 1. Modes 2 through 4 are zoomed in 10 times on the y-axis to make the small response visible. All units are arbitrarily normalized.

In the first case there is no coupling to the other modes at all, certainly the best case scenario. The second case is not too far from the ideal though. The 'leaking' response of the first mode to the others is less than an order of magnitude and virtually invisible in the response of the bottom mass. The damping ability of the mode seems to be hardly affected either. For completeness we now inject an impulse into the top mass to excite all the modes and observe that they all can be damped with incomplete actuation in the time required. Though this is not a proof of robustness it gives us an intuition that the 'leaking' between modes is tolerable in this case.

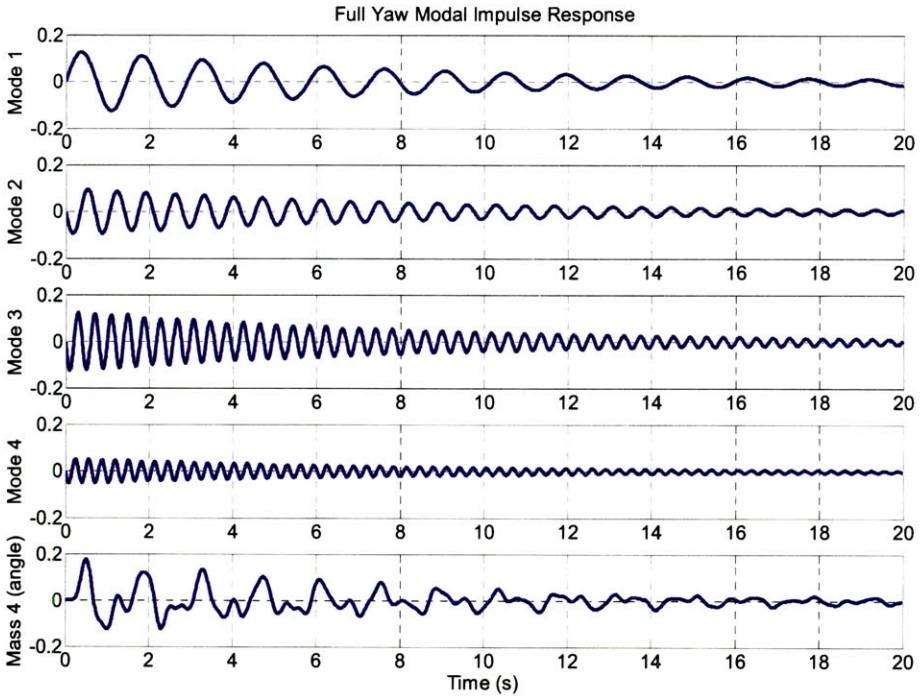


Figure 4.8: Damping response of all 4 modes with only top mass actuation to an impulse on yaw. All the modes are dominated by their own response and still damp within the required limit. In this case the gains are set to reduce each mode by 1/e in 9 seconds. All units are arbitrarily normalized

In Figure 4.8 all the modes are dominated by their own responses and actually damp in less time than the requirement. Here the controller gains are set to damp each mode to 1/e of the maximum value in 9 seconds. Specifically the values are

$$G1 = 7.7$$

$$G2 = 6.6$$

$$G3 = 2.2$$

$$G4 = 10.6$$

We thus conclude that as long as enough of a mode couples to the actuated mass, it is stable and can be damped. A more complete analysis of stability is given along with estimation model error in section 4.5.

4.4.2 SIMO State Estimator Design

The feasibility of modal control with incomplete actuation has been established so now it is time to include the state estimation. First we return to Eqn. (4.43)

$$J = \int_0^\infty \left(\tilde{q}^T Q \tilde{q} + z_m^T R z_m\right) dt \tag{4.43}$$

The design of the estimator involves choosing the values of the Q and R matrices. Large values of Q and small values of R mean better state estimation with large sensor noise injection. Small values of Q with large values of R mean low noise injection but poor state estimation. Earlier we said that the higher frequency modes contribute the most to noise injection. As a result Q is chosen to be of the form

$$Q = \begin{bmatrix} 1/freq(1) & & & 0 \\ & 1/freq(2) & & \\ & & \cdots & \\ 0 & & & 1/freq(n) \end{bmatrix} \tag{4.44}$$

In other words, the diagonal of Q is the reciprocal of all the resonant frequencies. This favors damping at the lower, more energetic end of the spectrum and noise rejection at the higher, most contributing end. Then all that is left is to do is choose the values for R. For the uncoupled degrees of freedom, yaw and vertical, choosing R is rather straightforward since it is a scalar in this case. The method extends without too much difficulty to the more complicated degrees of freedom as well.

We simulate the closed loop system with many values of R and plot the results with settling time and noise rejection at the bottom mass. Starting with yaw, the results are the following.

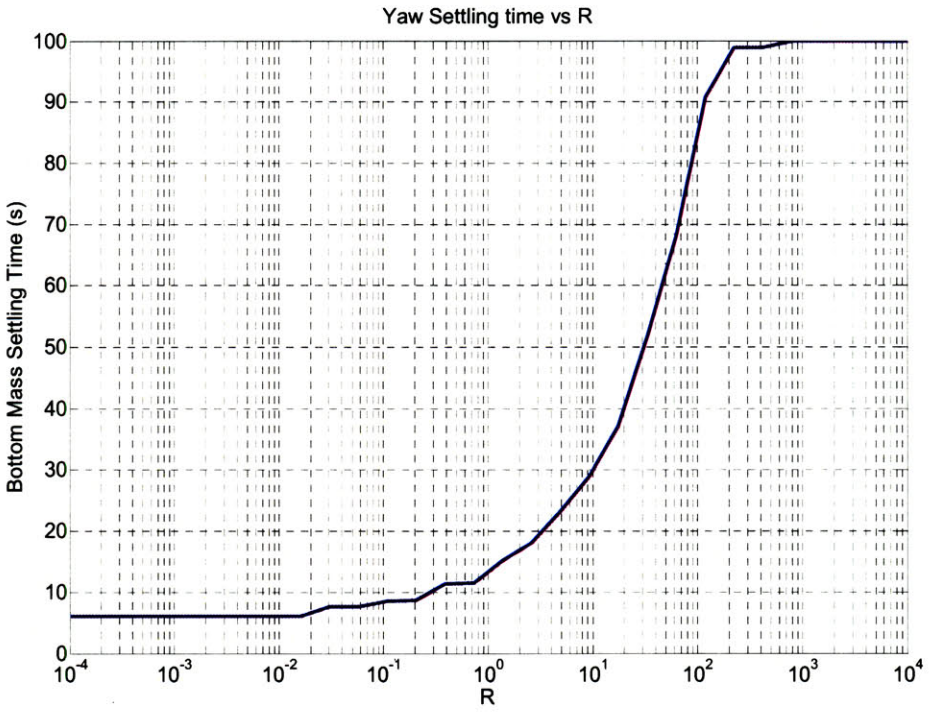


Figure 4.9: The settling time of the bottom mass in yaw versus the cost function parameter R. Small values of R approach the limit where the full state is completely observed. Large values of R approach the limit where there is no state estimation. The settling time has been artificially limited to 100 seconds since we do not care about values greater than that.

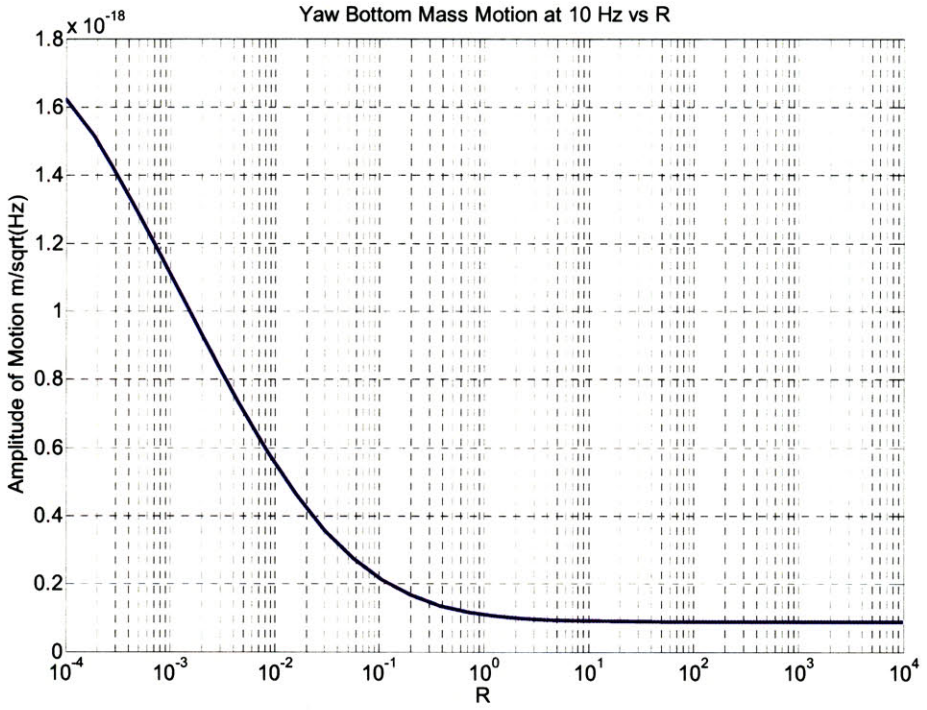


Figure 4.10: Motion of the bottom mass in yaw at 10 Hz as a function of R. As R gets larger the limit of zero sensor noise injection is achieved.

Decreasing values of R indicate a minimum settling time of 7.41 s which is the limit with complete state observation. Increasing values of R indicate a minimum noise level of $9.4 \times 10^{-20} m \cdot Hz^{-1/2}$, which is the seismic noise limit with zero sensor noise. The optimal point is someplace in the middle. A value of 0.1 seems to keep settling time at about the requirement of ten seconds while choosing a noise level nearly at the minimum. If the settling time turns out to be not quite under 10 seconds this is ok, because we will adjust the controller gains later. Performing the same process on the vertical DOFs results in an R of 0.1 as well.

Now we have an estimator for both yaw and vertical motion. Finally, we simulate the full system in yaw to show settling performances and make final adjustments on the gains.

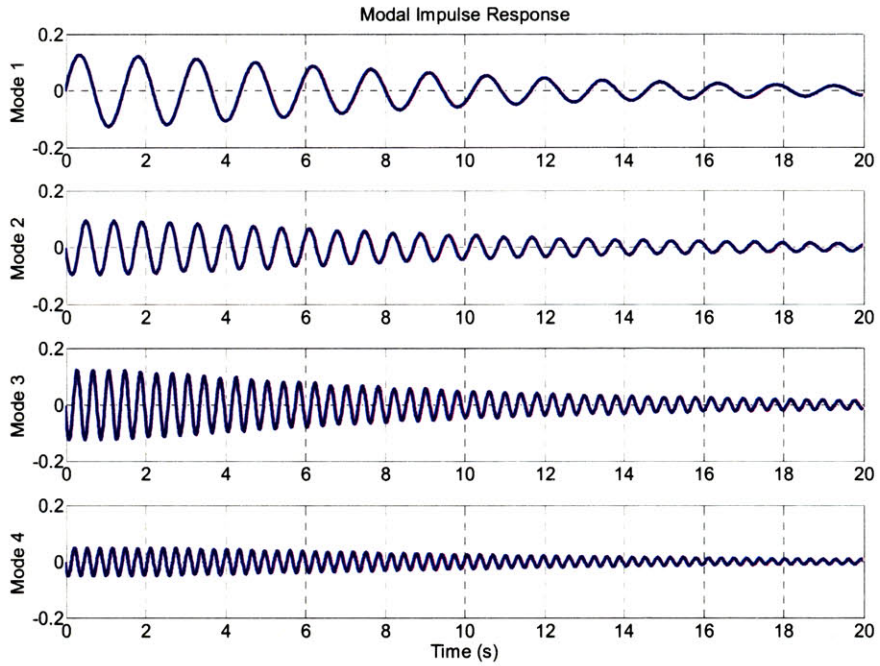


Figure 4.11: Modal settling response to a yaw impulse with state estimation. This response is very similar to the case with no state estimation.

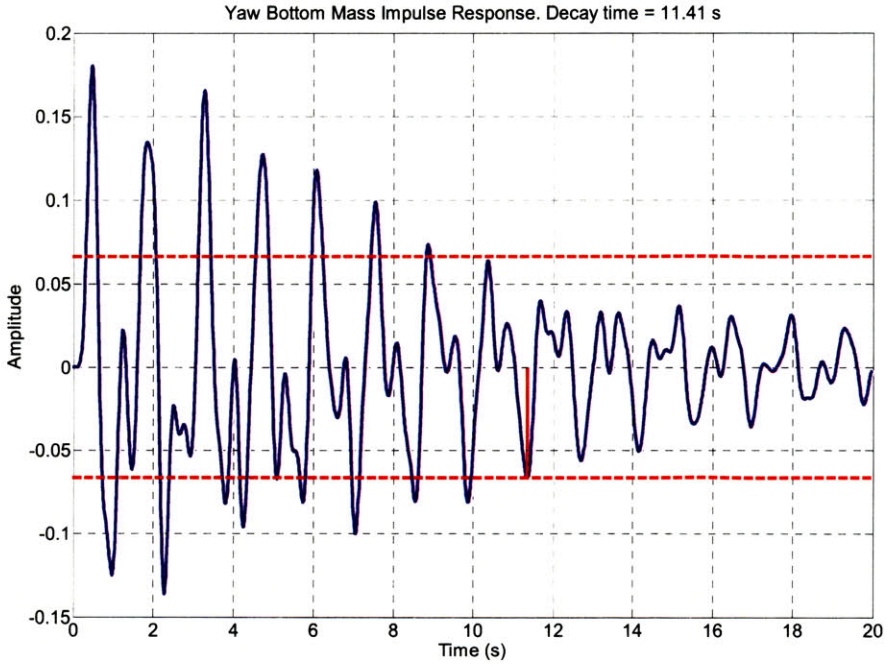


Figure 4.12: Settling response of the bottom mass in yaw to a yaw impulse. The red lines indicate 1/e of the maximum. The estimator is used with the original set of control gains which does not provide quite enough settling.

The modal response is largely unchanged from the case with no estimation. The bottom mass, however, does not quite meet the settling time requirement. We take this opportunity to illustrate that much of the bottom mass's response is dominated by the lower frequency modes. So, by increasing the gain on these modes while decreasing the gain on the higher modes we will improve both settling and noise rejection. The following new gains are chosen:

$$G1 = 12$$

$$G2 = 10$$

$$G3 = 1.7$$

$$G4 = 8$$

The new settling performance is

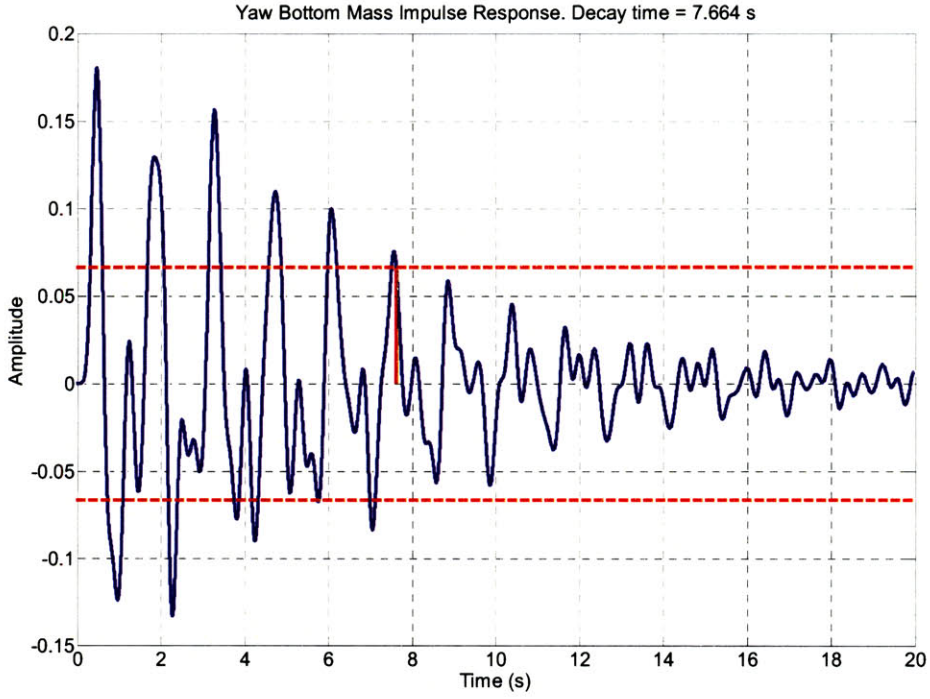


Figure 4.13: Settling response of the bottom mass in yaw with state estimation and a newly adjusted set of control gains. The red lines indicate 1/e of the maximum. This settling time is a few seconds faster with higher gains on the lower modes even though the gains on the higher modes were decreased.

Settling time decreases from over 11 s to 7.7 s. The total noise at 10 Hz, from seismic and sensor, decreases from $2.5\times10^{-19} m\cdot Hz^{-1/2}$ to $2.3\times10^{-19} m\cdot Hz^{-1/2}$. The final predicted noise spectrum is shown in Figure 4.14.

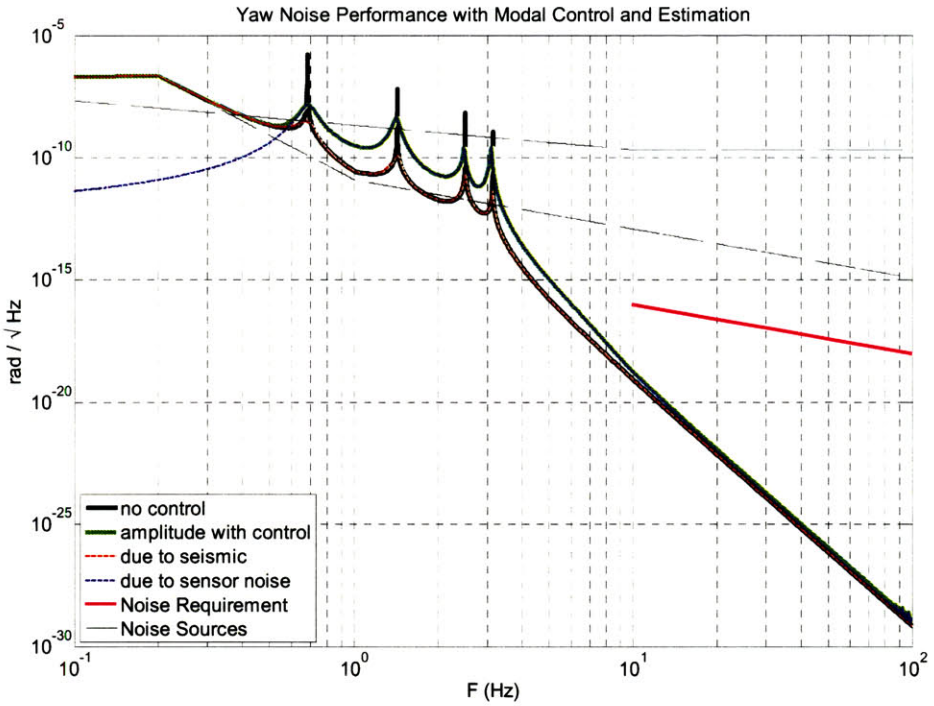


Figure 4.14: Yaw bottom mass noise performance with modal control and state estimation. The black curve is the response with no control, the green is the response with control, and the magenta is the requirement. The dashed lines are seismic and sensor noise. This method nearly meets the limiting factor of seismic noise and is far below the requirement.

The next figure, 4.14, compares this noise performance with the classical control method introduced in Chapter 3.

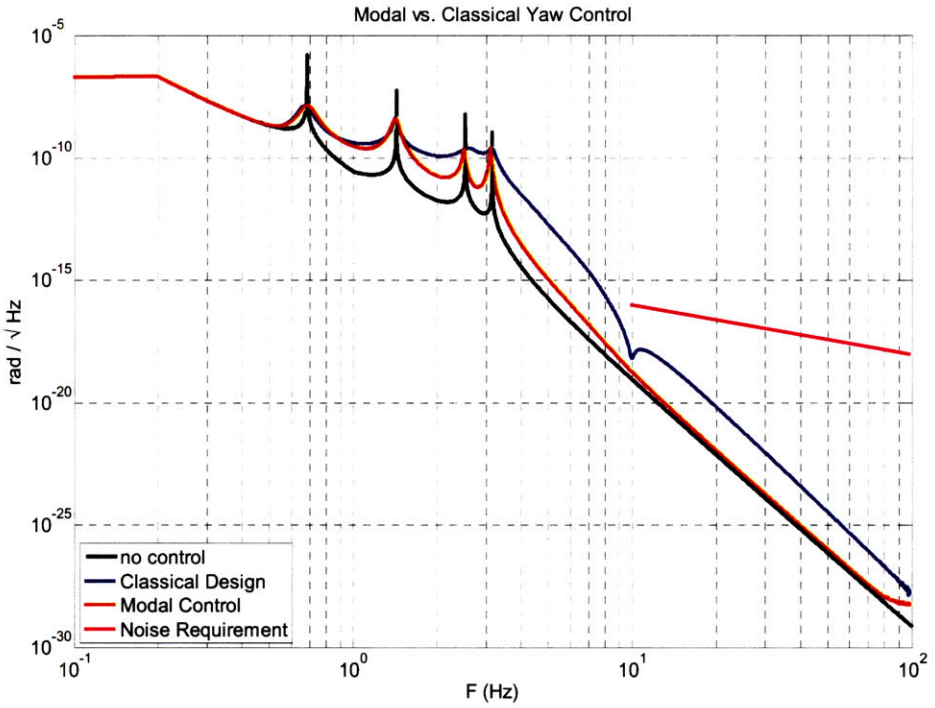


Figure 4.15: Comparison of classical control to modal control noise rejection on yaw. The blue curve is the performance with classical control, the red is with modal control and estimation, and the black is with no control. Modal control provides about 2 orders of magnitude more rejection above 10 Hz.

The noise performance of the modal control averages almost two orders of magnitude less than the classical control with similar settling times.

Settling time and noise performance for the vertical are very similar, shown in Figures 4.16 and 4.17 respectively.

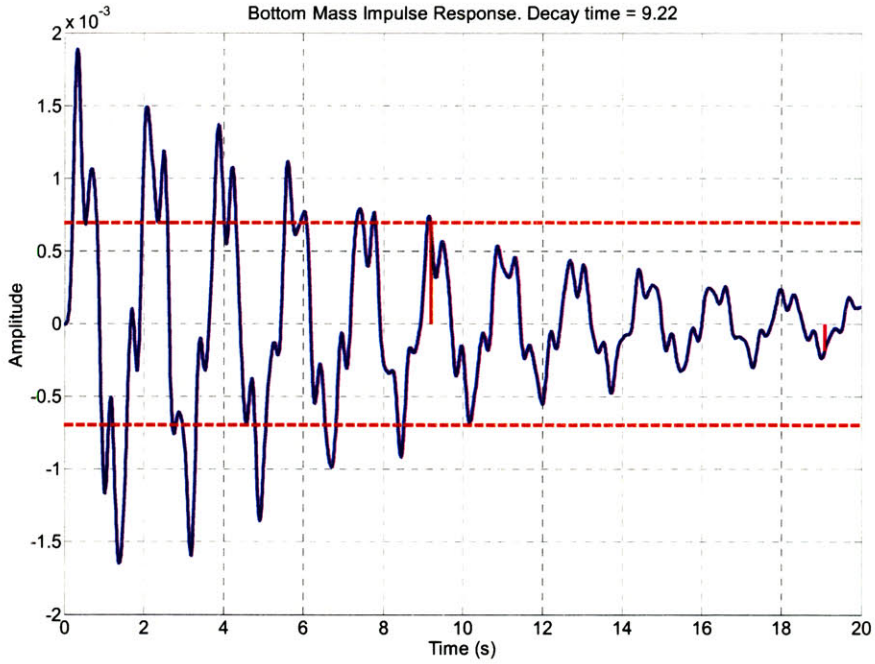


Figure 4.16: Vertical bottom mass settling time with modal control and state estimation. The red lines indicate 1/e of the maximum. This system settles by 1/e in 9.22 seconds.

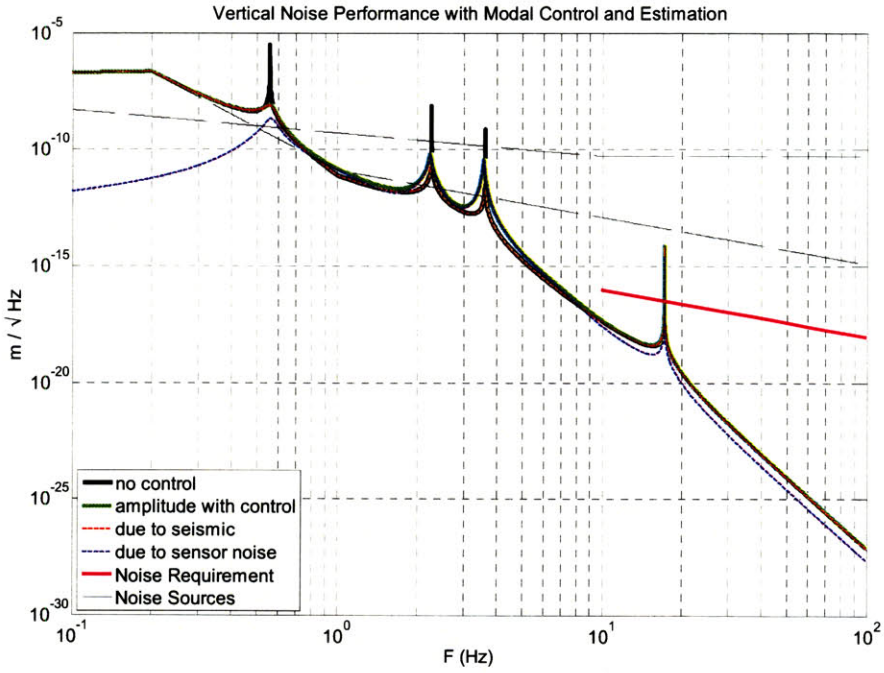


Figure 4.17: Vertical bottom mass noise performance with modal control and state estimation. The black curve is the response with no control, the green is the response with control, and the magenta is the requirement. The dashed lines are the noise sources. The noise level is at the seismic limit above 10 Hz except for the uncontrollable 17 Hz mode of the lower 2 masses.

The comparison of noise performance between modal and classical control for vertical motion is shown in Figure 4.18.

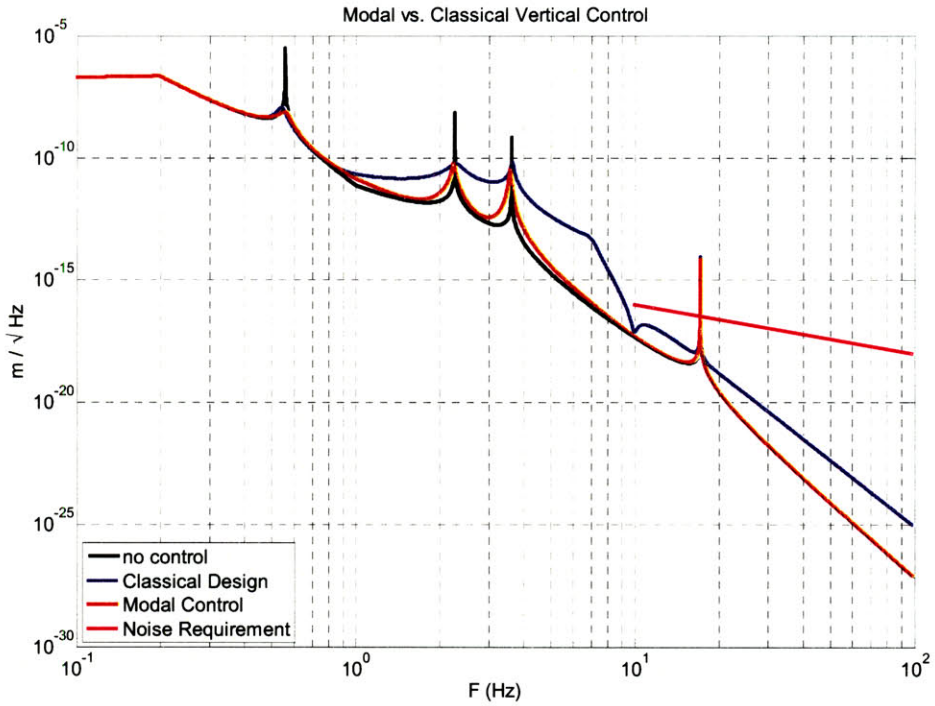


Figure 4.18: Comparison of classical control to modal control noise rejection on vertical. The black curve is the response with no control, the red is with modal control and estimation, and the blue is with classical control. Modal control provides an order of magnitude more noise rejection above 10 Hz.

Again, the results for vertical are similar and, in fact, show that the total motion above 10 Hz is dominated by seismic noise, the lowest possible limit.

4.5 Numerical Stability Analysis

As stated before, assuming a perfect model, stability is guaranteed if both the estimator and controller are stable individually. In the real world where perfect models do not exist, however, we must check that our understanding of the suspension's dynamics is good enough. In this section we simulate the whole closed loop system with the plant, estimator, and controllers. All the plant parameters are randomized by a certain percentage 100 times while holding the estimator fixed. Then the closed loop poles of all these iterations are plotted to see if any are unstable.

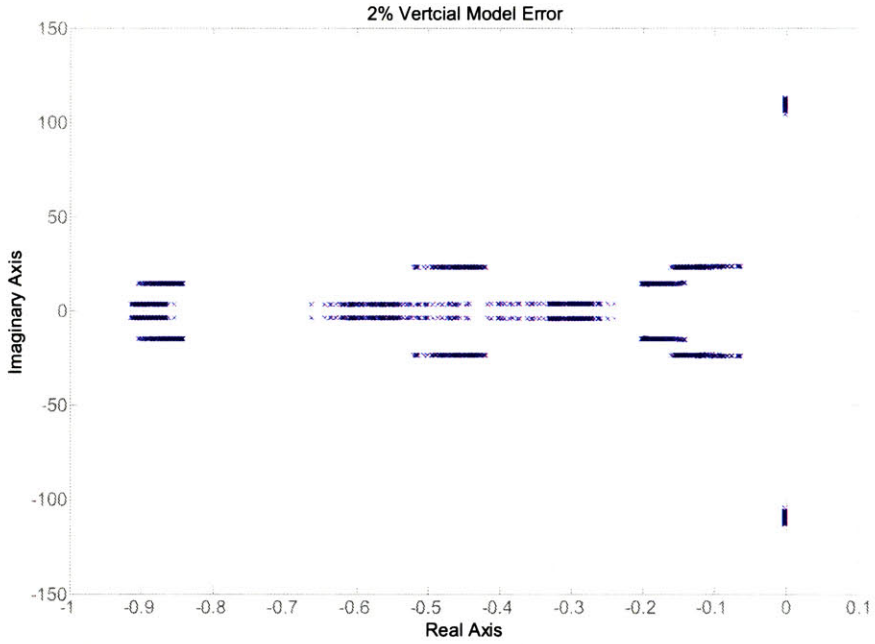


Figure 4.19: 100 iterations of the closed loop vertical poles with 2% randomized plant parameter error. Only the poles closest to the imaginary axis are shown. All poles are in the stable left hand plane. This amount of error is acceptable.

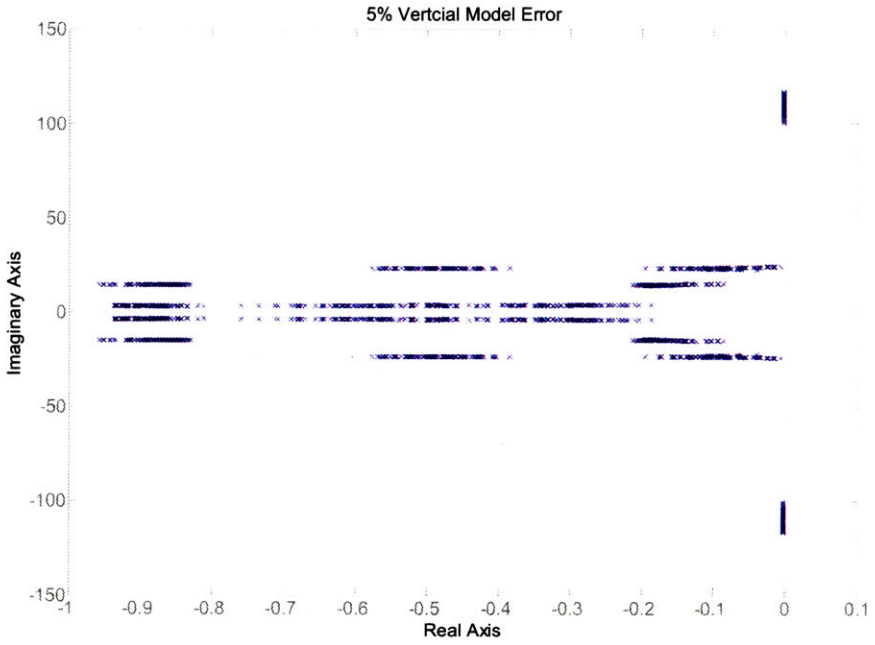


Figure 4.20: 100 iterations of the closed loop vertical poles with 5% randomized plant parameter error. Only the poles closest to the imaginary axis are shown. All poles are stable, though some may be poorly damped.

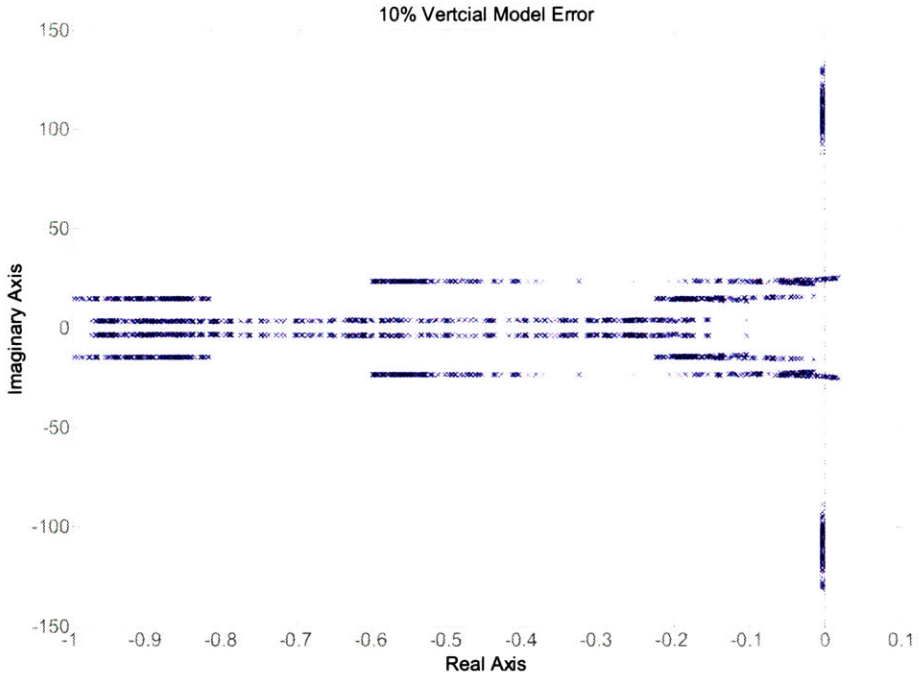


Figure 4.21: 100 iterations of the closed loop vertical poles with 10% randomized plant parameter error. Only the poles closest to the imaginary axis are shown. Possible instability may exist. This amount of error is unacceptable.

Figures 4.19 to 4.21 show 2%, 5%, and 10% error between plant and model, respectively, for the vertical DOFs. The figures zoom in on specifically the region that is most unstable to clearly indicate the crossing of the imaginary axis. The 2% and 5% cases show that the system is very unlikely to go unstable if the model error is less than 5%. However, if the error is greater than that, some poles may end up in the right half plane. The poles that appear on the imaginary axis are the undamped vertical fourth mode between the bottom two masses. They indicate that the position of the modes can move by a greater percentage than the parameter error, almost double in this case. Similar simulations on the yaw DOF show similar results. This information supports our original assumption that if we can keep the modes within 1% error of the model than all modes should be stable and well damped.

Chapter 5

Experiment

5.1 Setup

Due to the state of hardware and software installation, the experiment focuses on acquiring results from the yaw and vertical DOFs. The setup, shown in Figure 5.1 for the vertical system, is very much like what was shown previously in Figure 4.5. The OSEM signals are combined to produce vertical and yaw signals of the top mass. These signals are fed into a computer which then passes them through an estimator that reconstructs the modal motion of the pendulum in real time. The modal signals then go through control filters to create modal damping forces. These modal damping forces are mathematically converted to real damping forces by multiplying them with a transform of the eigenvector basis. The damping signals then go to both the DAC and the estimator. The DAC drives the actuators in the OSEMs that damp the pendulum. The model of the pendulum in the estimator is driven as well to increase the accuracy of modal state reconstruction.

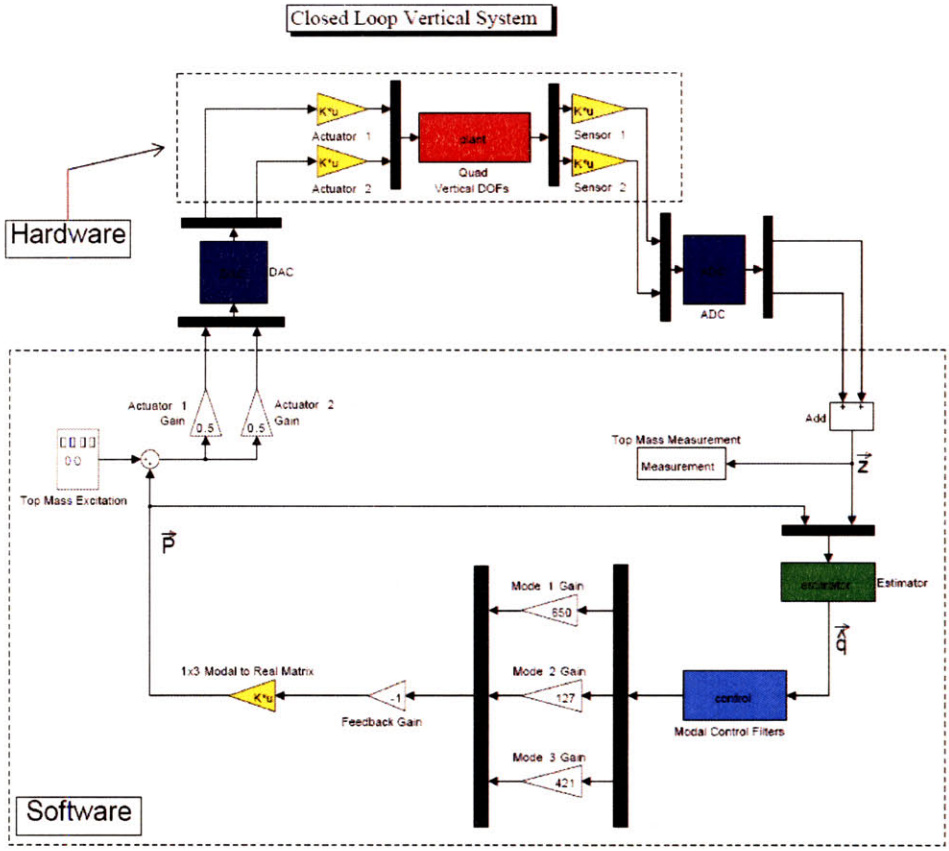


Figure 5.1: Experimentation schematic of the vertical system of motion. The two collocated sensors and actuators (OSEMs) are placed on the top mass such that each one contributes exactly half to the position measurement and actuation respectively. The sensor signals are passed through an ADC (sampling time 16384 Hz) and combined to produce the vertical position measurement. This measurement is used by the estimator to approximate the modal state of the system. The estimated modal state goes through the control filters and gains to generate the modal damping forces. These damping forces are converted, using a transform of the modal basis, into a vertical damping force. This damping force is split into two identical half strength signals and sent through a DAC. Each of these signals is given to the actuators to apply at the top mass of the quad.

The layout of the yaw system is virtually identical to that of the vertical and consequently it is not shown here. The main difference is that 4 DOFs are controlled in yaw while only 3 are controlled in vertical. Thus, there would be an extra gain box after the control filters.

For both systems two OSEMs are used to provide the sensing and actuation of the top mass position. Each OSEM contributes exactly one half to each system's position and actuation signal. The measured position is sampled by an ADC at 16384 Hz. This position signal, combined with the actuation signal, is passed through the discrete estimator to predict the modal responses. These estimated responses are sent through the

modal control filters to produce modal damping forces. Conversion of these modal damping forces to an applicable vertical or yaw damping force is done with the matrix transform stated in Chapter 4 as Eqn. (4.13), $P = (\Phi^T)^{-1}P_m$. Φ is the quad's modal basis. Since the top mass only contributes one DOF, only the top row of the $(\Phi^T)^{-1}$ matrix is needed. In Figure 5.1 this is the $1x3$ modal to real conversion. In the case of yaw this is a $1x4$ matrix. The resulting control signal is then returned to the estimator for estimation of the next sampled data point. Finally, of course, the control signal is also sent to the DAC and then the actuators to apply the vertical or yaw damping force.

With this configuration we cannot directly measure the motion of bottom mass, but we can make measurements on the top stage. Comparing the response of the top stage to those in simulation checks that our understanding of the complete closed loop system including the estimator, modal control, and mathematical transformations is correct. The confidence achieved from these results justifies the resources needed to extend this method onto the more complicated DOFs and completely control the pendulum.

5.2 Results

This section has four figures (Figures 5.2 to 5.5) summarizing the results of the modal control with state estimation on yaw and vertical. In all cases the control loops are turned on with the gains chosen in Chapter 4 so that the settling time is no more than ten seconds. Each set of DOFs has a damped transfer function and step response plot that includes curves from simulation and experimentation showing the amount of agreement of the results to the system design.

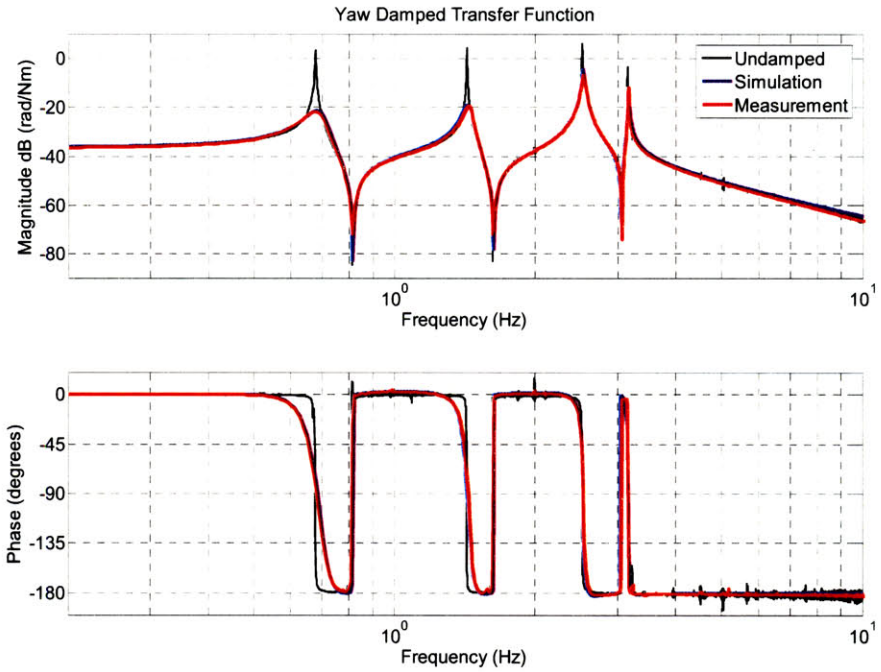


Figure 5.2: The red curve is the measured yaw TF of the top mass with modal control damping on. The damping gains are set to achieve a settling time of less than 10 s. The blue curve is a simulation of the same test. The black curve, the measured undamped response, is a reference to show peak reduction.

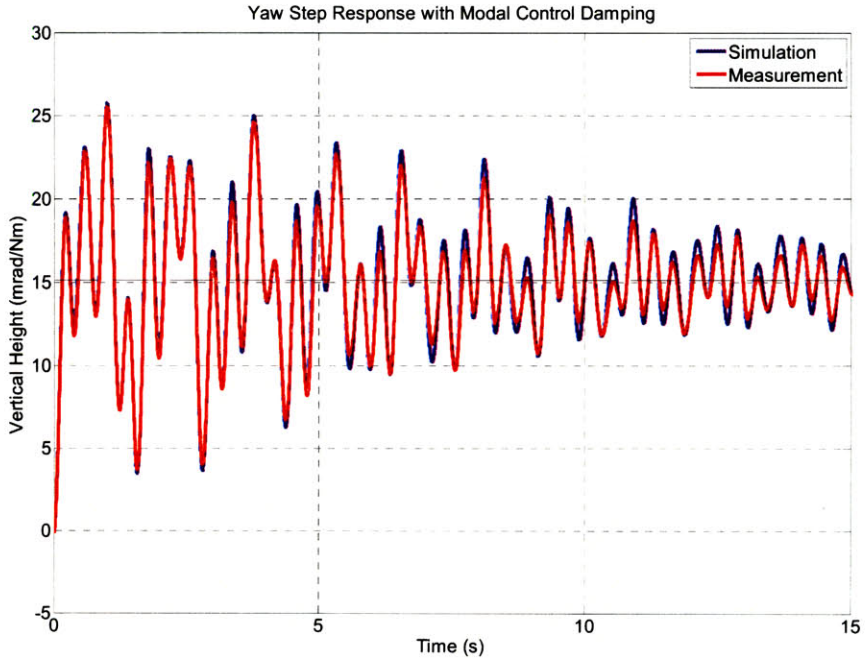


Figure 5.3: The red curve is the measured yaw step response of the top mass with the modal damping on. The damping gains are set to achieve a settling time of less than 10 s. The blue curve is a simulation of the same test. In this case the settling time is about 8.15 s.

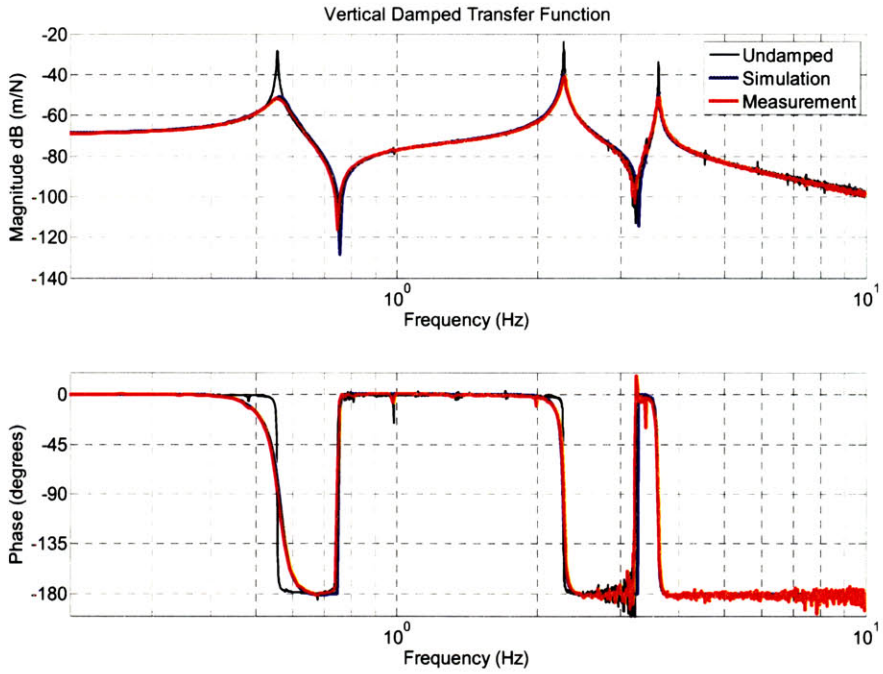
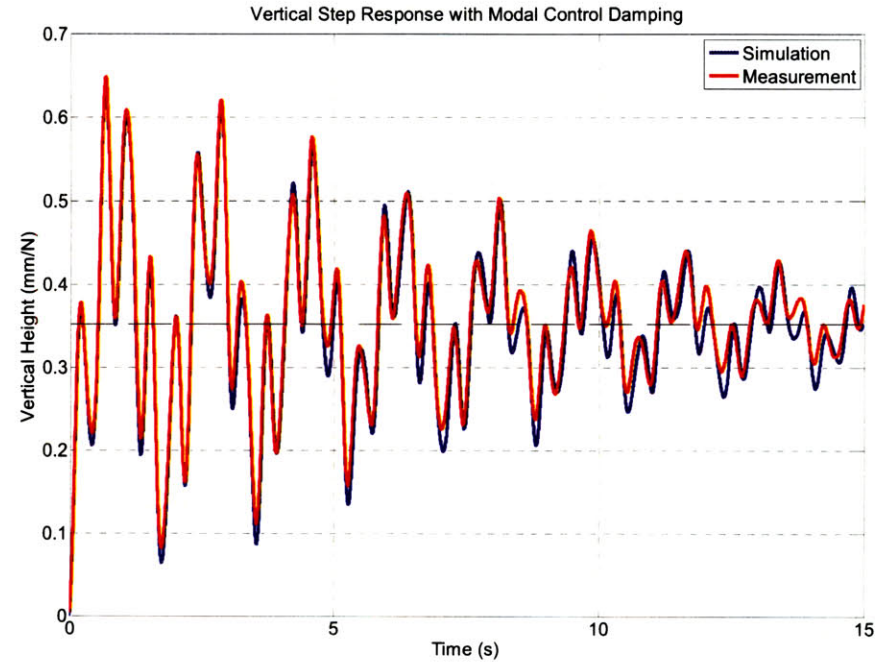


Figure 5.4: The red curve is the measured vertical TF of the top mass with modal control damping on. The damping gains are set to achieve a settling time of less than 10 s. The blue curve is a simulation of the same test. The black curve, the measured undamped response, is a reference to show peak reduction.



5.5: The red curve is the measured vertical step response of the top mass with the modal damping on. The damping gains are set to achieve a settling time of less than 10 s. The blue curve is a simulation of the same test. In this case the settling time is about 8.5 s.

The measured results match simulation well indicating that our understanding of the system is good. The settling times for yaw are 8.16 s for simulation and 8.14 s for experimentation. For vertical they are 8.87 s for simulation and 8.19 s for experimentation. The design requirements do not actually specify settling times for the top stage. However, they show agreement between simulation and measurement and are likely to be similar to the settling time for the bottom mass. It is also possible to adjust parameters in the model fit if better fits are needed or weight the fit to approximate certain modes better than others.

The high agreement between these results indicates that the likelihood of achieving similar results with noise injection and on the remaining DOFs is good as well.

Chapter 6

MIMO Estimation and Modal Control

In this chapter we simulate the application of modal control with MIMO state estimation to the more complex *x*-*y*-pitch-roll system. Here we develop the entire design process for the estimator and simulate the results. This process is not yet extended to experimentation since the fit of the estimator's model is not yet good enough to guarantee stability due to unexpected parameter uncertainty. An analysis of the robustness of the stability to estimator error is included to provide an intuition for where the model error needs to be to use this process.

6.1 Modal Control Simulation

First, to prove that modal control with partial actuation works by itself (i.e. without an estimator) on the quadruple coupled system a simulation is created assuming that all states of the pendulum are sensed. The gains of the control filters are set so that each mode damps in roughly nine seconds when an impulse is injected to all four DOFs of the top mass in the system. Instead of plotting the damping of all sixteen modes individually we prove damping by plotting the damping time of the bottom mass in each DOF when an impulse is injected to the same DOF on the top mass.

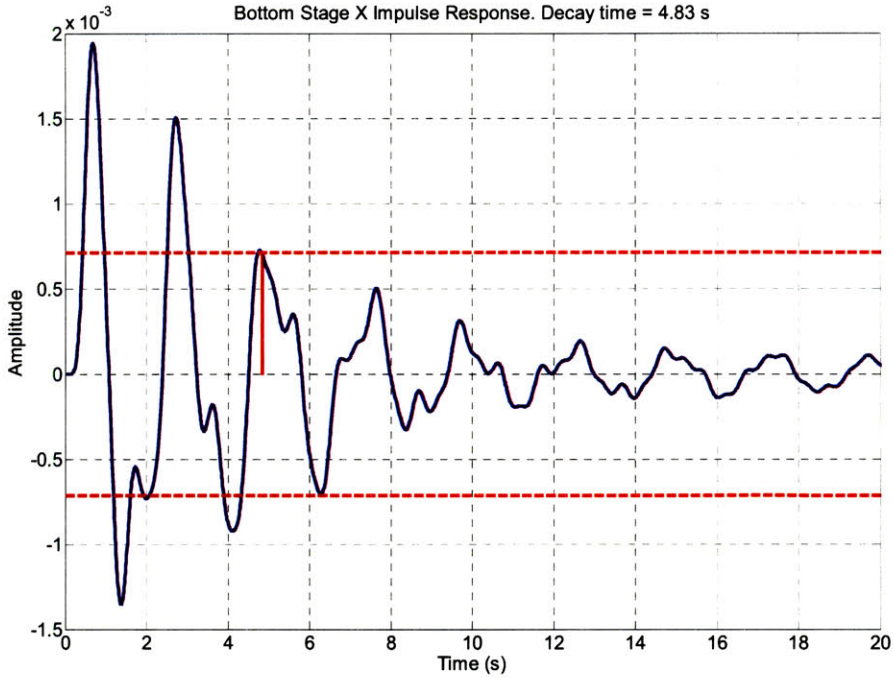


Figure 6.1: The pure modal control settling time of the bottom stage x DOF with an x impulse on the top stage.

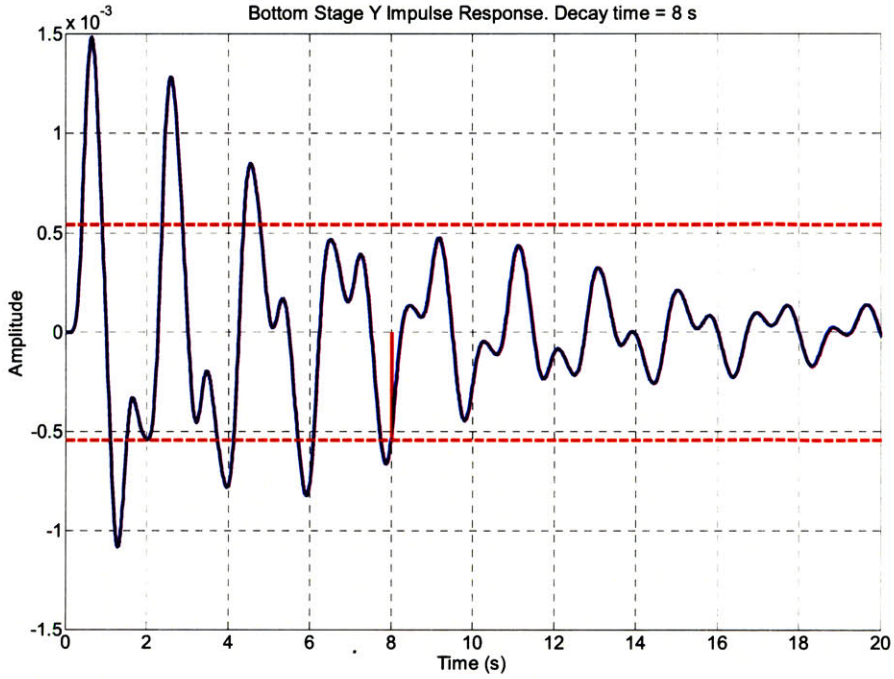


Figure 6.2: The pure modal control settling time of the bottom stage y DOF with a y impulse on the top stage.

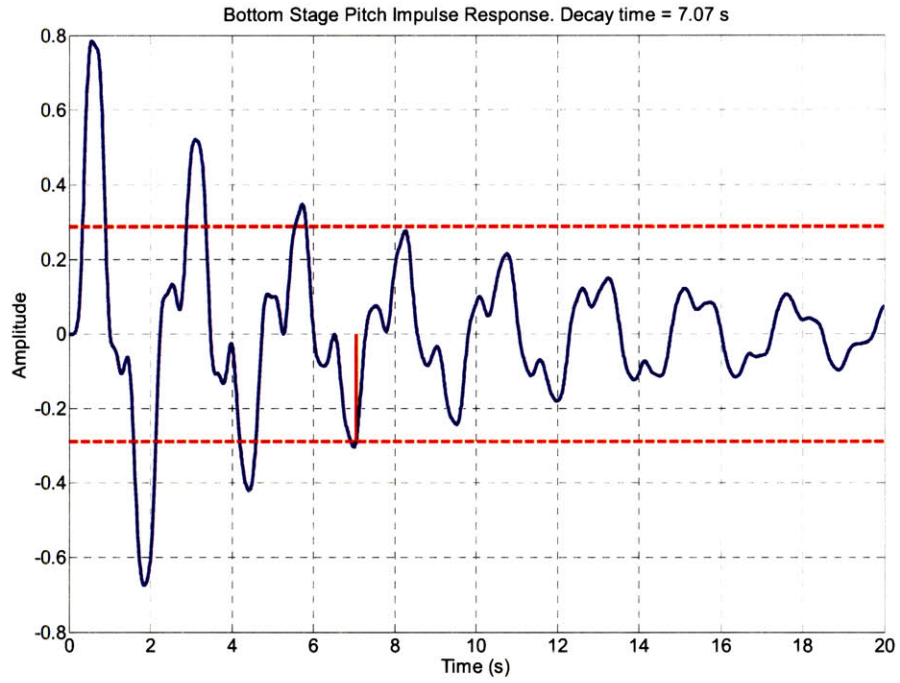


Figure 6.3: The pure modal control settling time of the bottom stage pitch DOF with a pitch impulse on the top stage.

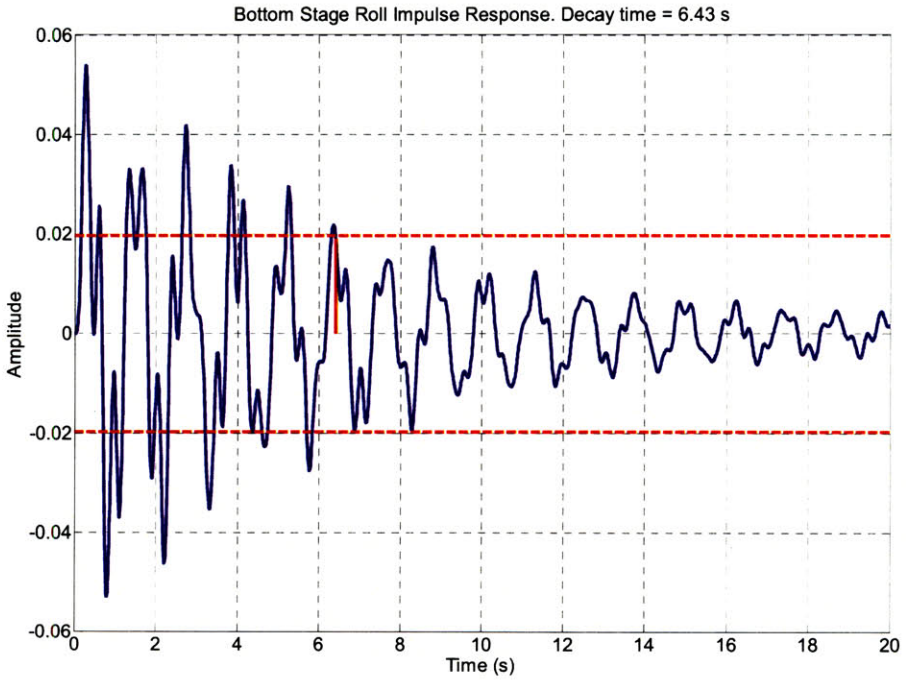


Figure 6.4: The pure modal control settling time of the bottom stage roll DOF with a roll impulse on the top stage.

These plots indicate that damping time is achievable well below the required limit. It is noteworthy that since this system is more complex, both with four times the number of modes and four times the number of DOFs, the damping is not quite as clean as the completely decoupled systems. Some cross coupling between modes does occur while restricted to control at the top mass. Despite this fact, the plots above confirm that damping can still occur if each mode is individually damped.

6.2 Design of a MIMO Modal Control Estimator

The design of the MIMO estimator largely follows the same procedure as that of the SIMO and still makes use of Eqn. (4.43).

$$J = \int_0^\infty \left(\tilde{q}^T Q \tilde{q} + z_m^T R z_m\right) dt \tag{4.43}$$

The structure of Q will remain the same where the diagonal is the inverse of the mode frequencies. The added complexity is that R is now a 4x4 matrix so the equation has more degrees of freedom to optimize. R will be kept diagonal like Q and will have the form

$$R = \begin{bmatrix} R1 & & & 0 \\ & R2 & & \\ & & R3 & \\ 0 & & & R4 \end{bmatrix} \tag{6.1}$$

Each of the four elements corresponds to one of the four sensor signals. In the SISO case it was easy to choose the single value of R because we could just plot it against settling time and noise injection. With four values of R there are too many dimensions to make clear visual plots. To get around this problem we consolidate all the information into a single scatter plot of noise versus settling time, shown in Figure 6.5. Each plotted value is one of many trials of different combinations of R values.

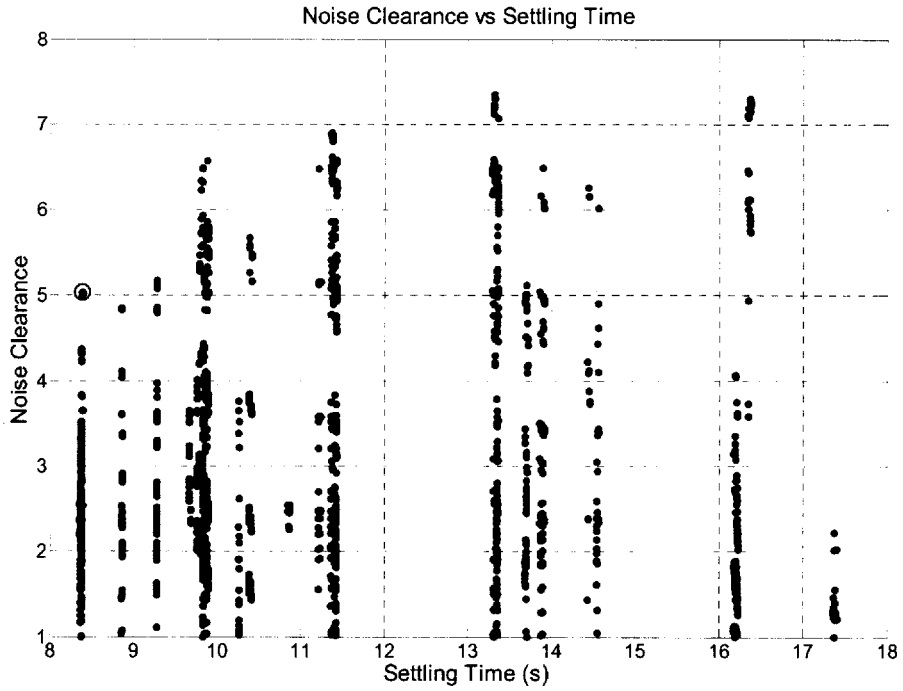


Figure 6.5: This scatter plot summarizes the performance of the estimator for many different combinations of the R matrix. Each point on the plot refers to an estimator with a unique *R* matrix. Settling time is the longest settling time of x, y, pitch, and roll for each value of *R*. Noise clearance is the factor below the noise limit at 10 Hz of the DOF closest to its requirement. For example, if the roll noise at 10 Hz is 2 times below its design requirement and all the other DOFs are farther below their respective requirements, then the plotted point will have a noise clearance of 2. The red circle encloses the 'optimal' *R* matrix.

In Figure 6.5 the noise clearance refers to how many times below the noise limit a trial is. The values plotted are the worst of all four types of motion. For example, if x, y, and pitch are 10 times below their noise limit, but roll is only 2 times below, then a noise clearance value of 2 will be reported. Settling time is chosen in the same way, except the largest value is reported. Thus, the optimal point is in the top left corner of the plot because it will have the highest noise rejection for the lowest settling time. The combination of *R* values chosen here is circled on the figure above. Any value between 1 and 10 seconds meets the requirements of the system so we could simply chose the one with the best noise reduction. However, settling time is a rough estimate of state estimation. Thus, choosing this value with a lower settling time assumes a more accurate estimator which should provide more flexibility in adjusting damping gains for whatever the situation may call for. Also, in this example, the worst case of the noise is still 5 times below the required limit. The *R* matrix corresponding to this choice is

$$R = \begin{bmatrix} 0.01 & & & 0 \\ & 0.01 & & \\ & & 1 & \\ 0 & & & 0.0362 \end{bmatrix} \tag{6.2}$$

The third value, corresponding to pitch, is two orders of magnitude larger than the other values. This means that pitch does not need highly accurate state estimation to provide sufficient damping, implying that estimation of the other degrees of freedom captures enough of the mode shapes to damp pitch as well.

6.3 Simulated Results with MIMO State Estimation

This section analyzes the settling time and noise performance of this control scheme on the quadruple coupled system. Figures 6.6 to 6.9 show settling time plots like those in the previous section, but with state estimation in place.

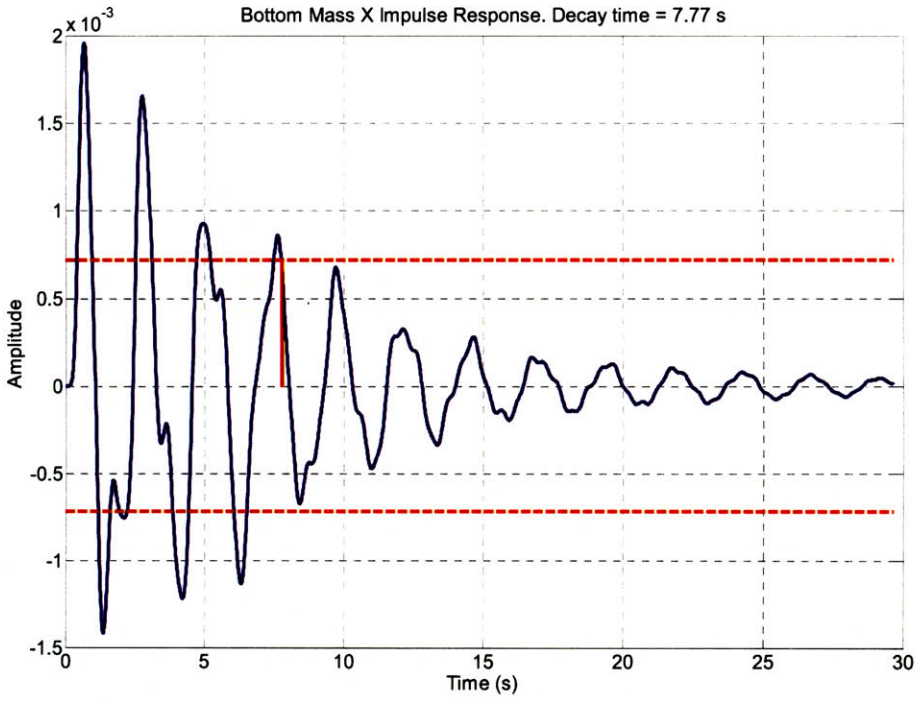


Figure 6.6: Bottom stage x response with modal control and state estimation to an x impulse on the top stage. Decay time is less than 10 s.

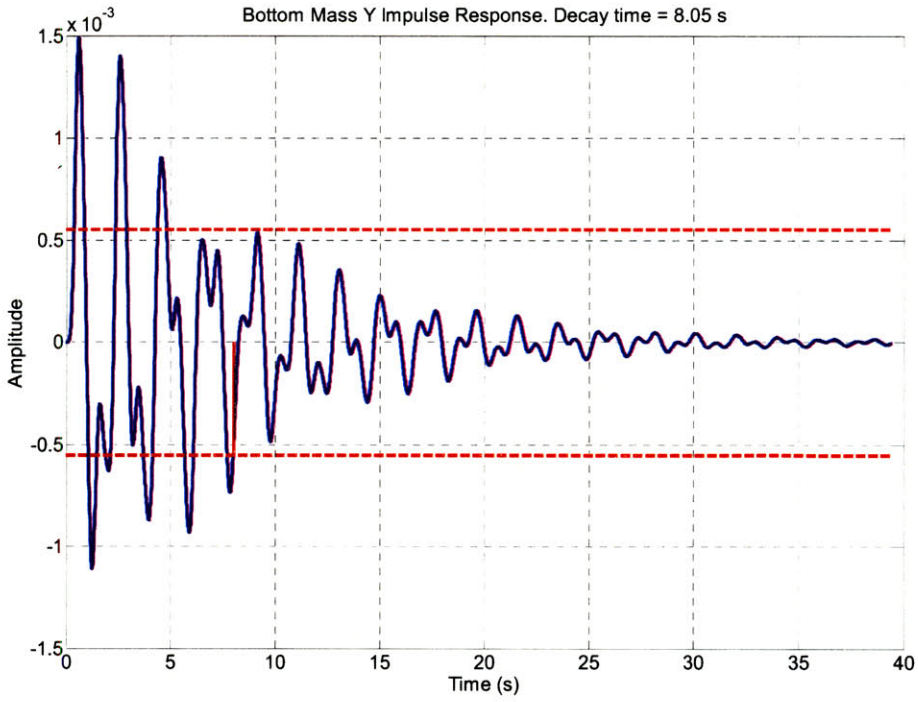


Figure 6.7: Bottom stage y response with modal control and state estimation to a y impulse on the top stage. Decay time is less than 10 s.

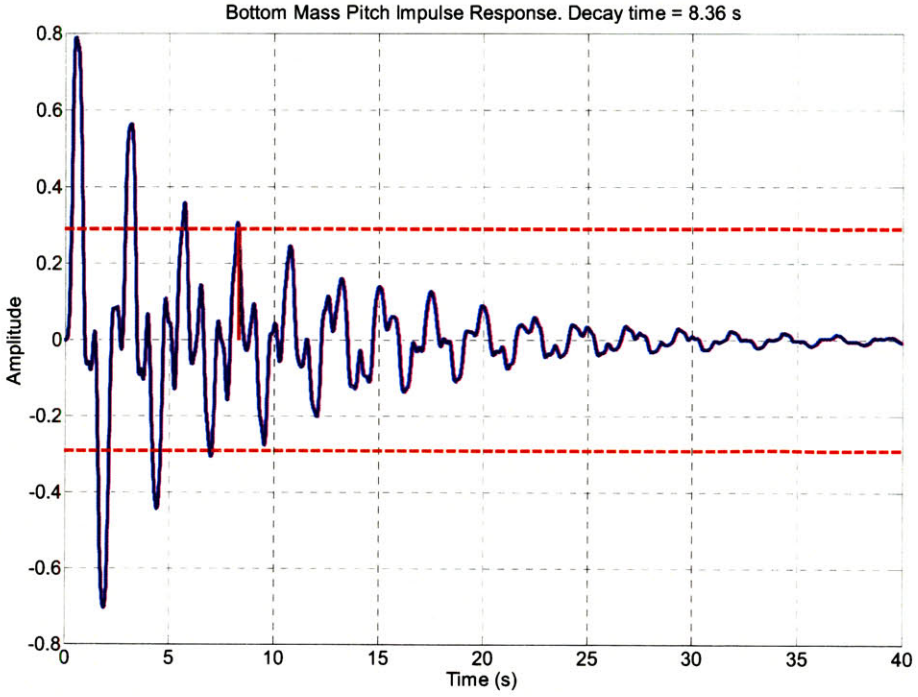


Figure 6.8: Bottom stage pitch response with modal control and state estimation to a pitch impulse on the top stage. Decay time is less than 10 s.

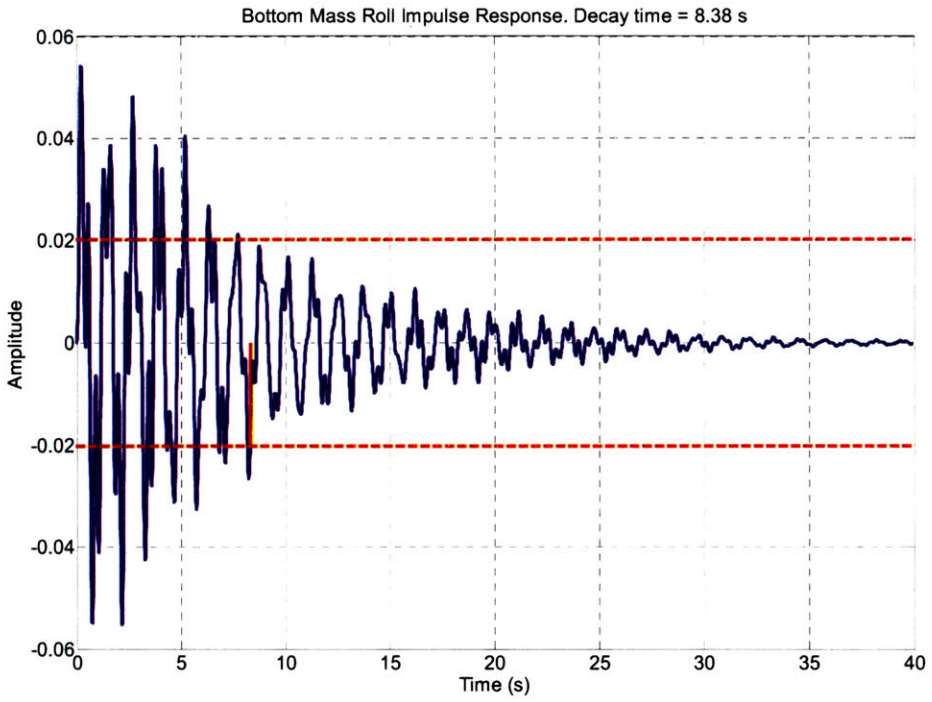


Figure 6.9: Bottom stage roll response with modal control and state estimation to a roll impulse on the top stage. Decay time is less than 10 s.

The settling time is higher than that without state estimation, so certainly the accuracy of the sensors has been slightly reduced. However, damping still occurs in less than the canonical 10 seconds. In fact some gains could even be reduced to further improve sensor noise rejection.

For now we keep the gains where they are to show that both the noise rejection and settling time can be set well below the limits and performance of classical control. Figures 6.10 to 6.13 are noise plots for each DOF on the bottom mass. Each plot has results of the modal control and estimation scheme along with a classical control scheme in order to show the difference in performance. The coupling of noises from all DOFs is taken into account. For example, the total noise in x is a sum of all noise sources coupling from x, y, pitch, and roll.

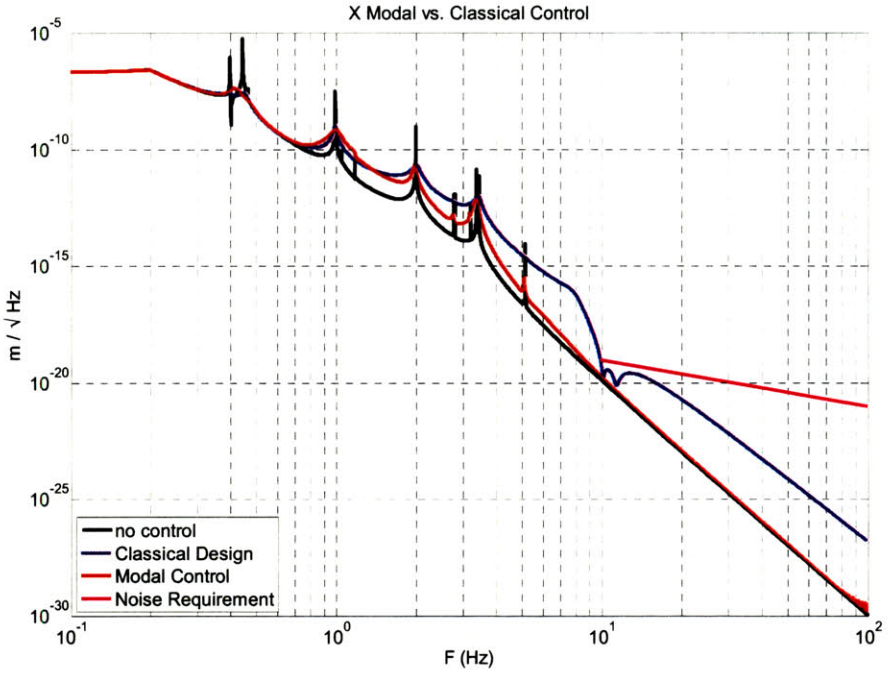


Figure 6.10: Bottom stage x noise response. The black curve represents the seismic noise limit with no control. The red curve is the total noise performance with modal control and state estimation. The blue curve is the total noise with classical control. The pink curve is the design requirement.

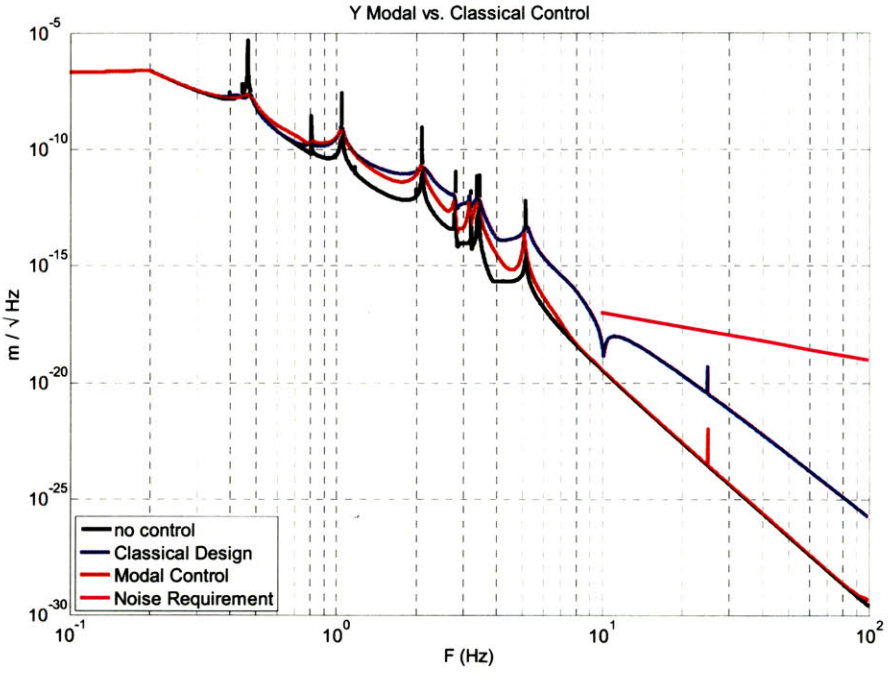


Figure 6.11: Bottom stage y noise response. The black curve represents the seismic noise limit with no control. The red curve is the total noise performance with modal control and state estimation. The blue curve is the total noise with classical control. The pink curve is the design requirement.

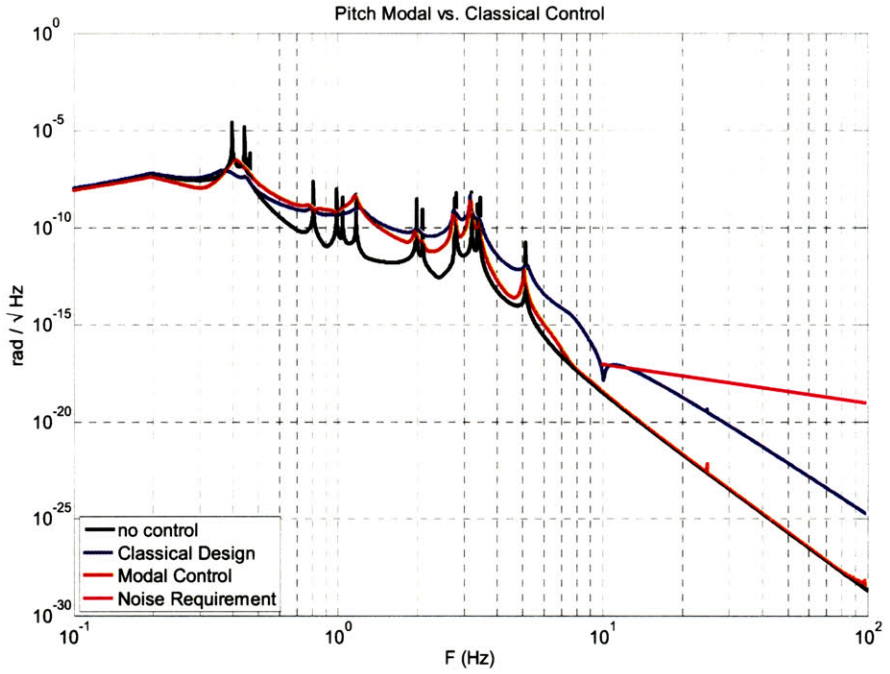


Figure 6.12: Bottom stage pitch noise response. The black curve represents the seismic noise limit with no control. The red curve is the total noise performance with modal control and state estimation. The blue curve is the total noise with classical control. The pink curve is the design requirement.

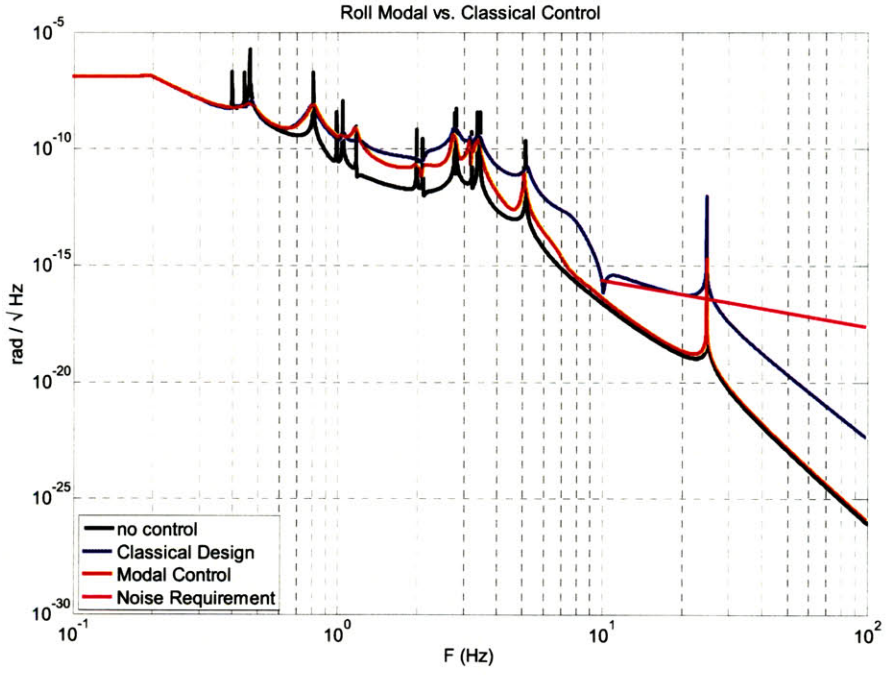


Figure 6.13: Bottom stage roll noise response. The black curve represents the seismic noise limit with no control. The red curve is the total noise performance with modal control and state estimation. The blue curve is the total noise with classical control. The pink curve is the design requirement.

Although not shown, the classical control filters have settling time under ten seconds as well. However, the modal control with state estimation produces a noise response one to two orders of magnitude less than classical control with a similar damping performance.

Another interesting change applied for this system is that the highest damped mode at 5.1 Hz has an altered control filter. This mode has a considerable noise contribution to roll, and although not necessary, the sensor noise injection can be significantly reduced by rolling off the gain of the control faster. The filter used is shown below. The stability margins for this filter are similar to the standard modal controller, however, due to the sharp changes in gain and phase the filter is sensitive to couplings from other nearby modes that may 'leak' through. This sensitivity is part of the reason the filter is used only at the 5.1 Hz mode, which is the most isolated from any other.

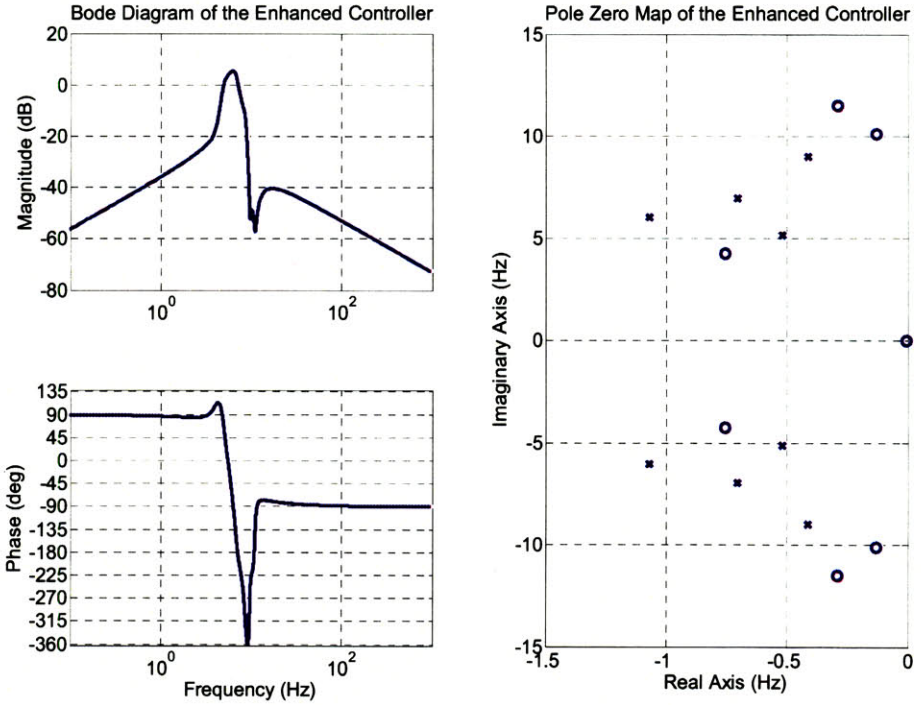


Figure 6.14: An alternative enhanced modal control filter. This filter has the advantage of rapidly rolling off the noise injection at 10 Hz. Implementing this filter on the 5.1 Hz mode alone reduces the noise injection above 10 Hz by more than 3 dB with negligible impact on settling time.

Being within a factor of 2 of the 10 Hz lower edge of the GW band, this mode has a larger contribution to sensor noise than any other. The sudden drop in gain at 10 Hz

makes this contribution much less than it otherwise would be. Another reason this alteration helps is the fact that the roll DOFs are the noisiest already because they only act as a triple pendulum up to the highest 25.1 Hz mode, which is inside the GW band. At 10 Hz the isolation is about an order of magnitude worse than the other DOFs. This is part of the reason that the classical control filter was not able to meet the limit at all. Thus, minimizing sensor noise even further is an added benefit. However, even without this enhanced filter, the control scheme still meets all the requirements since settling time is unaltered and the total noise still has some margin below the requirement. The main point here is to show that simple alterations on even one mode can improve performance if such changes are deemed necessary at a later time. To illustrate that this change is purely optional, the roll noise performance without it is plotted in Figure 6.15. The noise at 10 Hz is 1.44 times higher in this plot.

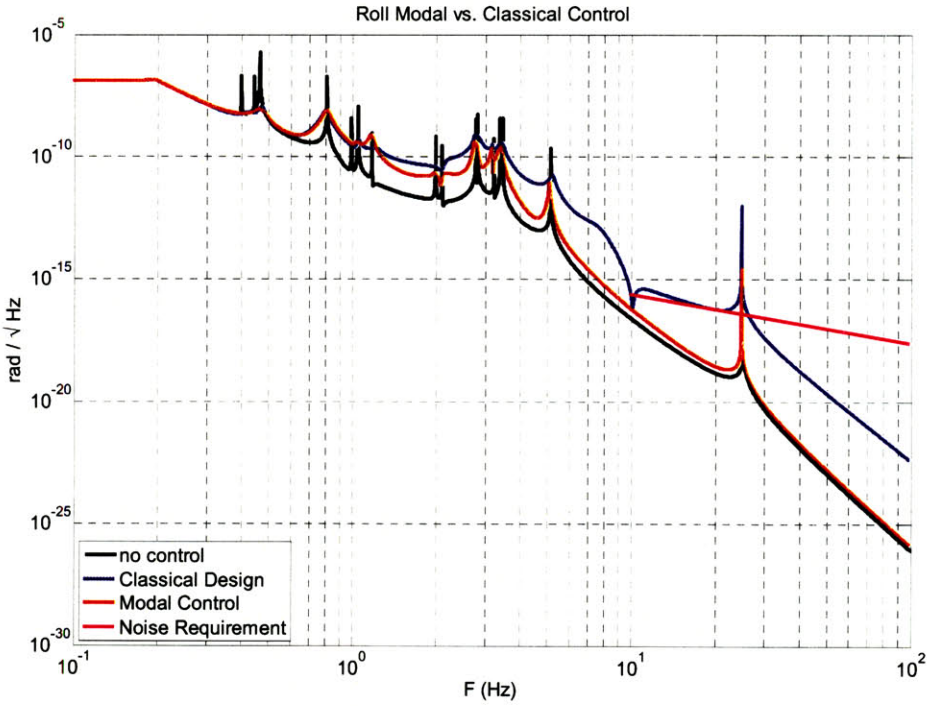


Figure 6.15: This figure is for comparison with Figure 6.13. It shows the noise performance without the enhanced filter on the 5.1 Hz mode shown in Figure 6.14. The noise above 10 Hz is over 3 dB higher here.

6.4 Numerical Stability Analysis

Due to the added complexity of this system the stability analysis is repeated here as well. We expect that, with the closer proximity of an increased number of modes and increased complexity of mode shapes relative to yaw and vertical, this system will be highly sensitive to model error. Figures 6.16 and 6.17 are plots of the most unstable region of closed loop poles when the parameters of the plant are randomized within the stated error 100 times. Each combination is plotted together to provide an idea of how likely a pole is to cross the imaginary access.

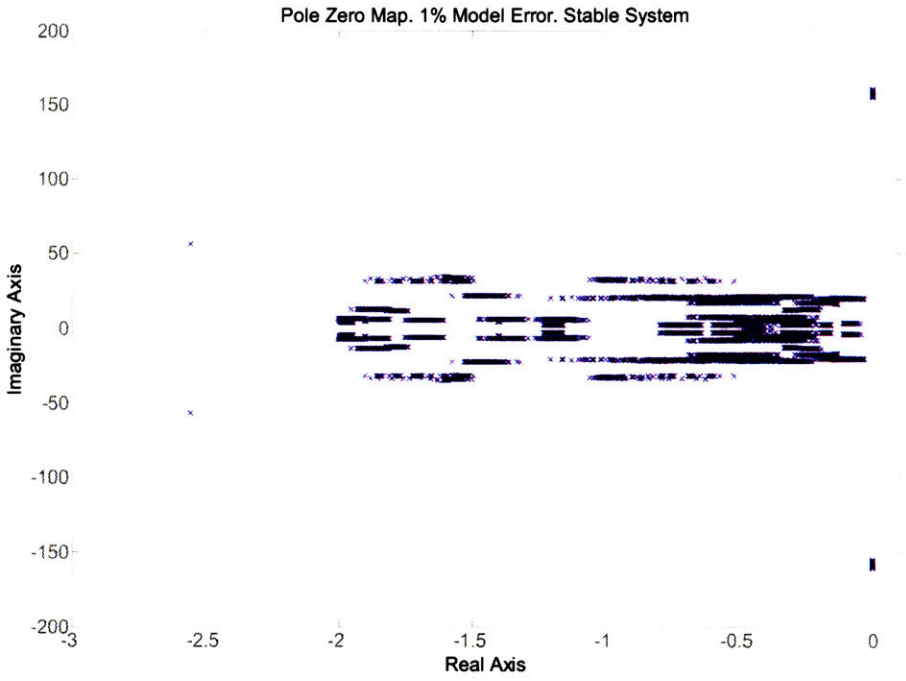


Figure 6.16: This is a pole zero map of the complete closed loop *x*-*y*-pitch-roll system with 100 trials of randomized plant error. The plant parameters are randomized to have a ±1% error with respect to the state estimator. The figure is zoomed in over the most unstable region of the system for clarity. All the poles are in the stable region indicating this amount of error is tolerable.

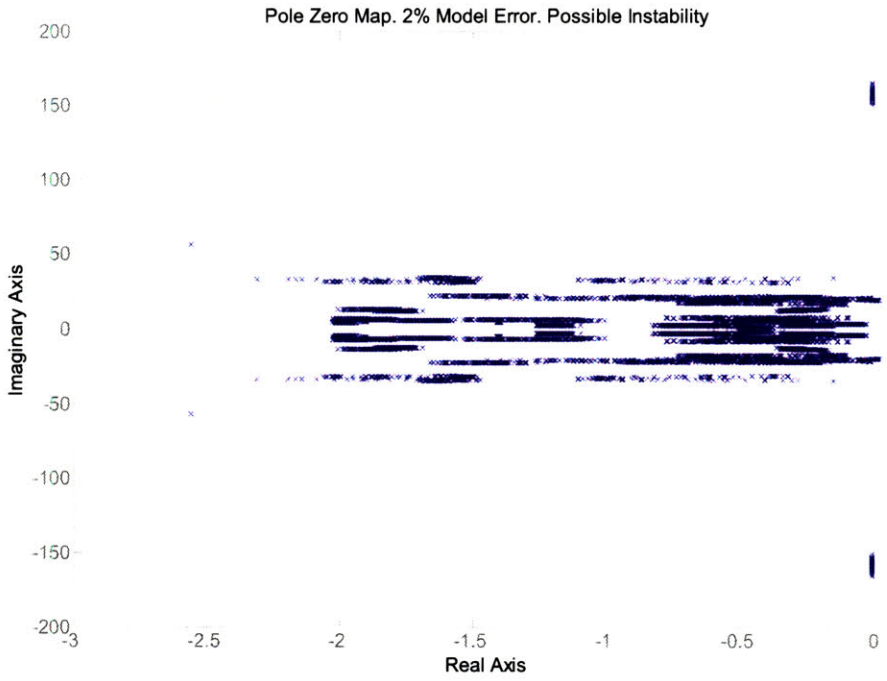


Figure 6.17: This is a pole zero map of the complete closed loop x-y-pitch-roll system with 100 trials of randomized plant error. The plant parameters are randomized to have a ±2% error with respect to the state estimator. The figure is zoomed in over the most unstable region of the system for clarity. Some of the poles are in the unstable region indicating this amount of error is not reasonable.

A system with 1% model error appears to have guaranteed stability since all poles are in the left hand plane. However, a system with just 2% error can have unstable poles in the right half plane. As expected, this is more sensitive than the decoupled systems which had guaranteed stability for at least 5% error. Thus, this model must be more accurate than the yaw and vertical models.

We know for sure that the current model is not this good since some modes are 20% to 30% off. Attempts to make a gradient descent fit prove that this method alone is not good enough. The algorithm lacks robustness and tends to get caught in local minima which cause it to output 'best fit' results that clearly do not match observations. More work needs to be done in order to get a proper fit of the model. Initial trials with alternate methods show good promise however. One method uses a Monte Carlo like algorithm. The model parameters are randomized within a certain uncertainty range many times. The error is calculated for each combination. After enough iterations there will be a combination of parameters with dynamics very similar to the actual quad. Since this

method simply makes a set of random guesses it is not possible for it to tend towards a false result. Then, using this result as an initial parameter set, gradient descent can reduce the error even further. The odds of the gradient descent outputting a false fit are highly reduced since the parameters should be very accurate already. The addition of more parameters to the error calculation can also help this method work faster. Since this system is rather complex there may be many combinations of parameters that put the modes in all the right places in the spectrum while still proving false mode shapes. As a result, including the measurable parts of the mode shapes into the error calculation should help these methods output the correct best fit solution.

This problem of model fitting is a major source of ongoing research. It may turn out that an entirely different method of model fitting is necessary or that it is not even possible within the given resources to solve this problem with this pendulum. This pendulum is scheduled to be replaced in the fall of 2007 with another, the 'Noise Prototype Quadruple Pendulum,' which is expected to be more accurately characterized. Thus, in a short time this issue of model fitting may become obsolete before it is solved. More about the parameter uncertainty and Noise Prototype are discussed in Chapter 7.

Chapter 7

Conclusions and Future Work

In this chapter we look at what work still remains to be done. This work includes the status of the interferometric cavity test that will sense motion of the x DOF at the bottom mass as well fitting the x-y-pitch-roll model to the measured data of the system. The results of the cavity test are expected to show conclusively the potential of this method of control for a quadruple pendulum. Final conclusions are also drawn in section 7.3.

7.1 Logistics of Control

The application of modal control with state estimation to the vertical and yaw DOFs is relatively easy since they are uncoupled from the other DOFs and can be treated as independent systems. The logistics of studying this method on the coupled DOFs of the quad are significantly different for various reasons. First, the cavity locking scheme must be known. For example, it was discovered with the triple pendulum that when a cavity is locked by actuating on a suspension's bottom mass, its dynamics change. As a result, the modes of the suspension will change and the modal control would no longer be valid. There are several ways of solving this issue. Two ideas include locking the cavity with another suspension or running the local control with a model that takes into account the effect of the cavity locking. Each method has advantages and disadvantages. Locking with another suspension ignores the purpose of the electrostatic drive and the fact that the other suspension may use modal control as well. Using an altered model of the quad

does not address the transition that will take place from an unlocked state to a locked state.

Another logistical issue is sufficiently understanding the dynamics of the quad. The problem here is that some of the pitch modes behave differently than the model. For example, one mode is measured 35% higher than what the model predicts. The reason for the discrepancy has to do with the equilibrium position (relative to the center of mass of the relevant stage of the quad) of the tips of the lower two stages of blade springs. These positions are important because they determine where the wires to the next lower stage break off and thus contribute largely to the stiffness in pitch. However, the exact location of the effective break off is difficult to measure since the wire does not bend exactly at the blade spring tip, but rather some millimeters lower [18]. Since the point is designed to be less than 1 mm from the center of mass, a discrepancy of a few millimeters makes a large difference. Because of this issue, it is not clear whether the model fit of the data is good enough to use in modal control and state estimation.

However, the model fit is expected to be resolved shortly with more study and various fitting techniques. In the mean time, the theory behind modal control works for any number of coupled DOFs and is extended to the rest of the quad here.

7.2 Quadruple Pendulum Noise Prototype

The current quadruple pendulum is known as the controls prototype, which is the second in a series of three prototyping stages to the final suspension configuration of Advanced LIGO. The third and final, known as the noise prototype will be assembled this summer for testing. The primary differences include new OSEMs and a change of three of the lower stages to fused silica masses to accommodate a realistic optic. Thus, the behavior of the dynamics should not be greatly altered with the exception of different mode shapes and frequencies.

Once assembled, this modal control with state estimation technique will be applied again. With this suspension we should be able to explore cavity locking techniques in more depth. This exploration into cavity locking should allow us to

understand exactly how modal control will be impacted by a locked cavity and how best to pursue it.

Another added complexity is that the suspension will be mounted to a two-stage active isolation platform with its own dynamics. The interaction between this isolation platform and the quadruple pendulum may not be trivial and will likely require further study as well.

7.3 Conclusion

Modal control with state estimation offers a high performing and easy to design alternative to classical control techniques. It allows the user the ability to decouple each mode and decompose a highly complex MIMO system into a collection of simpler SISO, or nearly SISO, systems. The user can then weight the importance of each mode in terms of noise and settling time. Generally more damping can be applied to the lower modes since they contribute the least towards sensor noise injection and the most towards overall settling time. With this technique, sensor noise injection can be pushed down to limits comparable to seismic noise, and in some cases even lower. Considering that classical control techniques still inject orders of magnitude more noise, modal control with state estimation promises to be an excellent tool for the control of Advanced LIGO suspensions.

Appendix A

Quadruple Pendulum Matlab Model

```
%adapted from Calum Torrie's ssmake4p by Ken Strain 3/01
%yaw and roll bugs fixed?
%
% 2/14/02, Mark Barton:
%   corrected quad matrix elements derived by generalizing
%   Calum's triple matrix elements using Mathematica
%   rename to ssmake4pv2eMB
% 2/24/05, Mark Barton:
%   new matrix elements with more realistic blades
%   generated from scratch in Mathematica
%   rename to ssmake4pv2eMB2
% 3/30/05, Mark Barton:
%   adjust sign of certain elements to match
%   conventions in Calum's original code

% 12/12/05, Mark Barton:
%   allow for lateral model with backward
%   compatibility to older elements
%   rename to ssmake4pv2eMB3

% 3/30/06, Mark Barton
% eliminate use of && operator for backwards compatibility with R6

% 4/6/06, Mark Barton
% rename ssmake4pv2eMB4
% allow for ribbons

% 4/26/06, Mark Barton
% include calculation of useful stuff formerly in quadopt.m: pend.tln, pend.tl1, pend.tl2,
%  pend.tl3, pend.l_suspoint_to_centreofoptic and pend.l_suspoint_to_bottomofoptic
% force calculation of flexn,... flex3 even if stage2 not set; export to pend

% 5/9/06, Mark Barton
% If pend.kcn is defined, use directly as kcn, rather than calculating kcn from
% ufcn. Similarly for pend.kc1 and pend.kc2.

% 9/15/06, Mark Barton
% Removed erroneous factor of 1/4 in M31 and M32 for ribbons

% 12/24/06, Mark Barton
% Added feature: if there is a string paramfile defined, it is used as the
% parameter file name instead of quadopt.

% 12/31/06, Mark Barton
% Cleaned up input order, renamed ssmake4pv2eMB4c.m

% 2/6/07, Mark Barton
% Reworked for full matrices, renamed ssmake4pv2eMB4f.m

global pend
pff = 0;
if exist('paramfile')
```

```
    if ischar(paramfile)
        if length(paramfile)>=1
            pff = 1;
            eval(paramfile);
        end
    end
end
if not(pff)
    quadopt;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%for model fit
if exist('modfit') == 0
    modfit = 0;
end

%parameter modfit is for model fitting
%modfit = 0 uses the default quad parameters
if modfit == 1              %LPTR
    if length(fitparam) == 32
    pend.dm =   fitparam(1);
    pend.dn =   fitparam(2);
    pend.d0 =   fitparam(3);
    pend.d1 =   fitparam(4);
    pend.d2 =   fitparam(5);
    pend.d3 =   fitparam(6);
    pend.d4 =   fitparam(7);
    pend.Iny =  fitparam(8);
    pend.I1y =  fitparam(9);
    pend.I2y =  fitparam(10);
    pend.I3y =  fitparam(11);
    pend.Inxy = fitparam(12);
    pend.I1xy = fitparam(13);
    pend.Inx =  fitparam(14);
    pend.I1x =  fitparam(15);
    pend.I2x =  fitparam(16);
    pend.I3x =  fitparam(17);
    pend.su =   fitparam(18);
    pend.si =   fitparam(19);
    pend.sl =   fitparam(20);
    pend.kx1 =  fitparam(21);
    pend.mn =   fitparam(22);
    pend.m1 =   fitparam(23);
    pend.m2 =   fitparam(24);
    pend.m3 =   fitparam(25);
    pend.Yn =   fitparam(26);
    pend.Y1 =   fitparam(27);
    pend.Y2 =   fitparam(28);
    pend.Y3 =   fitparam(29);
    pend.kcn =  fitparam(30);
    pend.kc1 =  fitparam(31);
    pend.kc2 =  fitparam(32);
```

```
    elseif length(fitparam) == 30
    pend.dm =   fitparam(1);
    pend.dn =   fitparam(2);
    pend.d0 =   fitparam(3);
    pend.d1 =   fitparam(4);
    pend.d2 =   fitparam(5);
    pend.d3 =   fitparam(6);
    pend.d4 =   fitparam(7);
    pend.Iny =  fitparam(8);
    pend.I1y =  fitparam(9);
    pend.I2y =  fitparam(10);
    pend.Inxy = fitparam(11);
    pend.I1xy = fitparam(12);
    pend.Inx =  fitparam(13);
    pend.I1x =  fitparam(14);
    pend.I2x =  fitparam(15);
    pend.su =   fitparam(16);
    pend.si =   fitparam(17);
    pend.sl =   fitparam(18);
    pend.kx1 =  fitparam(19);
    pend.mn =   fitparam(20);
    pend.m1 =   fitparam(21);
    pend.m2 =   fitparam(22);
    pend.m3 =   fitparam(23);
    pend.Yn =   fitparam(24);
    pend.Y1 =   fitparam(25);
    pend.Y2 =   fitparam(26);
    pend.Y3 =   fitparam(27);
    pend.kcn =  fitparam(28);
    pend.kc1 =  fitparam(29);
    pend.kc2 =  fitparam(30);
    end
elseif modfit == 2          %Yaw
    pend.Inz =  fitparam(1);
    pend.I1z =  fitparam(2);
    pend.I2z =  fitparam(3);
elseif modfit == 3          %Vertical
    pend.kcn = fitparam(1);
    pend.kc1 = fitparam(2);
    pend.kc2 = fitparam(3);
    pend.mn =  fitparam(4);
    pend.m1 =  fitparam(5);
    pend.m2 =  fitparam(6);
elseif modfit == 23         %Vertical-Yaw
    pend.Inz =  fitparam(1);
    pend.I1z =  fitparam(2);
    pend.I2z =  fitparam(3);
    pend.I3z =  fitparam(4);
    pend.kcn =  fitparam(5);
    pend.kc1 =  fitparam(6);
    pend.kc2 =  fitparam(7);
    pend.mn  =  fitparam(8);
```

```
    pend.m1  =  fitparam(9);
    pend.m2  =  fitparam(10);
elseif modfit == 4          %Transverse-Roll
    pend.nn0 = fitparam(1);
    pend.nn1 = fitparam(2);
    pend.n0 = fitparam(3);
    pend.n1 = fitparam(4);
    pend.n2 = fitparam(5);
    pend.n3 = fitparam(6);
    pend.n4 = fitparam(7);
    pend.n5 = fitparam(8);
    pend.kcn = fitparam(9);
    pend.kc1 = fitparam(10);
    pend.kc2 = fitparam(11);
elseif modfit == 10
    pend = pend_mm;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%for model fit



% general redefinition of variables to simplify notation
% and cope with different pendulum designs

% Make sure all flags are defined
if not(isfield(pend,'stage2'))
    pend.stage2=0; % define, default to false
end
if not(isfield(pend,'ribbon') )
    pend.ribbon=0; % define, default to false
end

g = pend.g;
Yn   = pend.Yn;
Y1   = pend.Y1;
Y2   = pend.Y2;
Y3   = pend.Y3;
if isfield(pend, 'kcn')
    kcn = pend.kcn;
    pend.ufcn = sqrt(2*kcn/pend.mn)/2/pi;
else
    kcn =  1/2 * (2*pi*pend.ufcn)^2*pend.mn;
    pend.kcn = kcn;
end
if isfield(pend, 'kc1')
    kc1 = pend.kc1;
    pend.ufc1 = sqrt(2*kc1/pend.m1)/2/pi;
else
    kc1 =  1/2 * (2*pi*pend.ufc1)^2*pend.m1;
    pend.kc1 = kc1;
end
if isfield(pend, 'kc2')
```

```
    kc2 = pend.kc2;
    pend.ufc2 = sqrt(2*kc2/pend.m2)/2/pi;
else
    kc2 =  1/2 * (2*pi*pend.ufc2)^2*pend.m2;
    pend.kc2 = kc2;
end

% for backward compatibility, default off-axis MOI terms to zero
if isfield(pend, 'Inxy')
else
    pend.Inxy = 0;
end
if isfield(pend, 'Inyz')
else
    pend.Inyz = 0;
end
if isfield(pend, 'Inzx')
else
    pend.Inzx = 0;
end
if isfield(pend, 'I1xy')
else
    pend.I1xy = 0;
end
if isfield(pend, 'I1yz')
else
    pend.I1yz = 0;
end
if isfield(pend, 'I1zx')
else
    pend.I1zx = 0;
end
if isfield(pend, 'I2xy')
else
    pend.I2xy = 0;
end
if isfield(pend, 'I2yz')
else
    pend.I2yz = 0;
end
if isfield(pend, 'I2zx')
else
    pend.I2zx = 0;
end
if isfield(pend, 'I3xy')
else
    pend.I3xy = 0;
end
if isfield(pend, 'I3yz')
else
    pend.I3yz = 0;
end
```

```
if isfield(pend, 'I3zx')
else
    pend.I3zx = 0;
end

Inxy = pend.Inxy;
Inyz = pend.Inyz;
Inzx = pend.Inzx;
I1xy = pend.I1xy;
I1yz = pend.I1yz;
I1zx = pend.I1zx;
I2xy = pend.I2xy;
I2yz = pend.I2yz;
I2zx = pend.I2zx;
I3xy = pend.I3xy;
I3yz = pend.I3yz;
I3zx = pend.I3zx;


ln   = pend.ln;
l1   = pend.l1;
l2   = pend.l2;
l3   = pend.l3;
kwn =pend.Yn*pi*pend.rn^2/ln*pend.nwn/2;
kw1 =pend.Y1*pi*pend.r1^2/l1*pend.nw1/2;
kw2 =pend.Y2*pi*pend.r2^2/l2*pend.nw2/2;
if pend.ribbon
    kw3 =pend.Y3*pend.W3*pend.t3/l3*pend.nw3/2;
else
    kw3 =pend.Y3*pi*pend.r3^2/l3*pend.nw3/2;
end

%**********************************************************************
% allows choice of 2 wires to set separation to zero
sm = 0; % separation of top wires at structure
sn = 0; % separation of top wires at mass N

s0=pend.su;
s1=pend.su;
s2=pend.si;
s3=pend.si;
s4=pend.sl;
s5=pend.sl;

dm=pend.dm;
dn=pend.dn;
d0=pend.d0;
d1=pend.d1;
d2=pend.d2;
d3=pend.d3;
d4=pend.d4;
```

```
%*********************************************************************
mn = pend.mn;
m1 = pend.m1;
m2 = pend.m2;
m3 = pend.m3;
Inx = pend.Inx;
I1x = pend.I1x;
I2x = pend.I2x;
I3x = pend.I3x;
Iny = pend.Iny;
I1y = pend.I1y;
I2y = pend.I2y;
I3y = pend.I3y;
Inz = pend.Inz;
I1z = pend.I1z;
I2z = pend.I2z;
I3z = pend.I3z;

mn3  = mn+m1+m2+m3;
m13 = m1+m2+m3;
m23 = m2+m3;

nn0 = pend.nn0;
nn1 = pend.nn1;
n0  = pend.n0;
n1  = pend.n1;
n2  = pend.n2;
n3  = pend.n3;
n4  = pend.n4;
n5  = pend.n5;

%*************************************************************************************
% cosine and sine of the angle the wire makes with the vertical (z)

sin=(nn1-nn0)/ln;           %sin(omegan)
si1=(n1-n0)/l1;         %sin(omega1)
si2=(n3-n2)/l2;         %sin(omega2)
si3=(n5-n4)/l3;         %sin(omega3)

cn=(ln^2-(nn1-nn0)^2)^0.5/ln;   %cos(omegan)
c1=(l1^2-(n1-n0)^2)^0.5/l1; %cos(omega1)
c2=(l2^2-(n3-n2)^2)^0.5/l2; %cos(omega2)
c3=(l3^2-(n5-n4)^2)^0.5/l3; %cos(omega3)

%*********************************************************
bd = pend.bd;
%*********************************************************************

kn = kwn;
k1 = kw1;
k2 = kw2;
```

```
k3 = kw3;
su=pend.su;
si=pend.si;
sl=pend.sl;

% Vertical distances between centres of mass
pend.tln = sqrt(pend.ln^2 - (pend.nn0-pend.nn1)^2) + pend.dm;
pend.tl1 = sqrt(pend.l1^2 - (pend.n0-pend.n1)^2) + pend.dn + pend.d0;
pend.tl2 = sqrt(pend.l2^2 - (pend.n2-pend.n3)^2) + pend.d1 + pend.d2;
pend.tl3 = sqrt(pend.l3^2 - (pend.n4-pend.n5)^2) + pend.d3 + pend.d4;

% Distance to the centre of mass from suspension point
pend.l_suspoint_to_centreofoptic  =  pend.tln +pend.tl1+pend.tl2+pend.tl3;

% Distance to the bottom of the test mass from suspension point
pend.l_suspoint_to_bottomofoptic  =  pend.tln +pend.tl1+pend.tl2+pend.tl3+pend.tr;

% Stiffness of fibres/ribbons
pi = 4*atan(1);
Mn1 = (1/4)*pi*pend.rn^4;
M11 = (1/4)*pi*pend.r1^4;
M21 = (1/4)*pi*pend.r2^4;
if pend.ribbon
    M31 = pend.W3*pend.t3^3/12;
    M32 = pend.W3^3*pend.t3/12;
else
    M31 = (1/4)*pi*pend.r3^4;
    M32 = (1/4)*pi*pend.r3^4;
end
pend.flexn = sqrt(2*Mn1*Yn/mn3/g)*cn^(3/2);
pend.flex1 = sqrt(4*M11*Y1/m13/g)*c1^(3/2);
pend.flex2 = sqrt(4*M21*Y2/m23/g)*c2^(3/2);
pend.flex3 = sqrt(4*M31*Y3/m3/g)*c3^(3/2); % for long/pitch, yaw and vertical
pend.flex3tr = sqrt(4*M32*Y3/m3/g)*c3^(3/2); % for trans/roll, with M32 instead of M31

flexn = pend.flexn;
flex1 = pend.flex1;
flex2 = pend.flex2;
flex3 = pend.flex3; % for long/pitch, yaw and vertical; see below for trans/roll

if pend.stage2
    % apply fudges to approximate Mathematica Stage2 results
    dm = dm + flexn;
    dn = dn + flex1;
    d0 = d0 + flex1;
    d1 = d1 + flex2;
    d2 = d2 + flex2;
    d3 = d3 + flex3;
    d4 = d4 + flex3;
    ln = ln - 2*flexn/cn;
    l1 = l1 - 2*flex1/c1;
    l2 = l2 - 2*flex2/c2;
```

```
    l3 = l3 - 2*flex3/c3;
    nn0 = nn0 + sin*flexn/cn;
    nn1 = nn1 - sin*flexn/cn;
    n0 = n0 + si1*flex1/c1;
    n1 = n1 - si1*flex1/c1;
    n2 = n2 + si2*flex2/c2;
    n3 = n3 - si2*flex2/c2;
    n4 = n4 + si3*flex3/c3;
    n5 = n5 - si3*flex3/c3;
end

newelements=0;
if isfield(pend,'kxn')
    if isfield(pend,'kx1')
        if isfield(pend,'kx2')
            % lateral compliances have been defined - use lateral model matrix
            % elements

            kxn = pend.kxn;
            kx1 = pend.kx1;
            kx2 = pend.kx2;

            symbexport4latfull
            newelements = 1;
        end
    end
end

if not(newelements)
    symbexport4full
end

mbquadA = [...
    zeros(24) eye(24)
    (-km)\(xm-cqxm'/qm*cqxm) -bd*eye(24)
];

bm = (-km)\(cxsm-cqxm'/qm*cqsm); % ground displacement inputs: x,y,z,yaw,pitch,roll
bm2 = km\eye(24); % coordinate force inputs: pitchn,xn,pitch1,x1,...

mbquadB = [...
    zeros(24,30)
    bm bm2
];

mbquadC = [...
    eye(24) zeros(24,24)
];

 mbquadD = [...
    zeros(24,30)
 ];
```

```
mbquad = ss(mbquadA,mbquadB,mbquadC,mbquadD);
```

```
% This file just redirects to the file that does the work...

clear pend;

quadopt20060509controlsmainasbuiltoffdiag;
%quadopt20060509controlsmainasbuiltsym;
```

```
%function[pend] = quadopt
global pend

% Corresponds to case 20060509controlsmainasbuiltsym of model mbquadlite2lateral
% For use with matrix elements ssmake4pv2eMB4.m/symbexport4lat.m of 5/9/06 or later.
% An approximation to the as-built controls prototype (10/11/05 build, main chain).
% Same as 20060331asbuiltsym but with measured masses from T040229-12
% and blade spring constants recomputed directly from bounce mode frequencies.
% All as-built numbers except off-diagonal MOI components are zeroed.

                        pend.version = '20060509controlsmainasbuiltsym';
                              pend.g = 9.81;
                             pend.nx = 0.1300; % T040214-01
                             pend.ny = 0.5000; % T040214-01
                             pend.nz = 0.0840; % T040214-01
                           pend.denn = 4000; % T040214-01
                             pend.mn = 22.285; % measured, T040229-12
                            pend.Inx = 0.4557; % MPL; 9/1/05
                            pend.Iny = 0.0712; % MPL; 9/1/05
                            pend.Inz = 0.4546; % MPL; 9/1/05
                             pend.ux = 0.1300; % T040214-01
                             pend.uy = 0.5000; % T040214-01
                             pend.uz = 0.0840; % T040214-01
                           pend.den1 = 4000; % T040214-01
                             pend.m1 = 21.372; % measured, T040229-12
                            pend.I1x = 0.5106; % MPL; 9/1/05
                            pend.I1y = 0.0598; % MPL; 9/1/05
                            pend.I1z = 0.5136; % MPL; 9/1/05
                             pend.ix = 0.1300; % T040214-01
                             pend.ir = 0.1570; % T040214-01
                           pend.den2 = 3860; % T040214-01
                             pend.m2 = 39.2; %39.264; % measured, T040229-12
                            pend.I2x = 0.4708; % CAD; CT 11/17/05
                            pend.I2y = 0.2772; % CAD; CT 11/17/05
                            pend.I2z = 0.2762; % CAD; CT 11/17/05
                             pend.tx = 0.1300; % T040214-01
                             pend.tr = 0.1570; % T040214-01
                           pend.den3 = 3980; % T040214-01
                             pend.m3 = 39.4; %39.408; % measured, T040229-12
                            pend.I3x = 0.4640; % CAD; CT 11/17/05
                            pend.I3y = 0.2737; % CAD; CT 11/17/05
                            pend.I3z = 0.2722; % CAD; CT 11/17/05
                             pend.ln = 0.4465; % measured, BS, 03/16/06
                             pend.l1 = 0.3085; % T040214-01
                             pend.l2 = 0.3400; % T040214-01
                             pend.l3 = 0.6000; % T040214-01 (T050172 was misleading - this
didn't change)
                            pend.nwn = 2;
                            pend.nw1 = 4;
                            pend.nw2 = 4;
                            pend.nw3 = 4;
                             pend.bd = 0.00;
```

```
                        pend.rn = 5.200*10^-04; % MB; 9/29/05
                        pend.r1 = 3.5000*10^-04; % T040214-01
                        pend.r2 = 3.1000*10^-04; % T040214-01
                        pend.r3 = 0.018*0.0254/2; % 0.018" - spool; 11/18/05
                        pend.Yn = 2.2000*10^+11; % T040214-01
                        pend.Y1 = 2.2000*10^+11; % T040214-01
                        pend.Y2 = 2.2000*10^+11; % T040214-01
                        pend.Y3 = 2.119*10^+11; % measured - MB; 11/18/05
                        pend.dm = 1.0000*10^-03 - 0.00507 + 0.0023; % T050172
                        pend.dn = 1.0000*10^-03 - 0.00335 + 0.002;%         +0.00741; ↙
% T050172
                        pend.d0 = 1.0000*10^-03 - 0.00335 + 0.0017; % T050172
                        pend.d1 = 1.0000*10^-03 - 0.00302 + 0.002;%         +0. ↙
0036576; % T050172
                        pend.d2 = 1.0000*10^-03 - 0.00302 + 0.0015; % T050172
                        pend.d3 = 1.0000*10^-03 - 0.00204 + 0.001; % T050172
                        pend.d4 = 1.0000*10^-03 - 0.00204 + 0.001; % T050172
                pend.twistlength = 0;                % T040214-01
                       pend.d3tr = 1.0000*10^-03;    % T040214-01
                       pend.d4tr = 1.0000*10^-03;    % T040214-01
                         pend.sn = 0;                % T040214-01
                         pend.su = 0.003;  % T040214-01
                         pend.si = 0.003;  % T040214-01
                         pend.sl = 0.015;  % T040214-01
                        pend.nn0 = 0.250;  % T040214-01
                        pend.nn1 = 0.090;  % T040214-01
                         pend.n0 = 0.200;  % T040214-01
                         pend.n1 = 0.060;  % T040214-01
                         pend.n2 = 0.140;  % T040214-01
                         pend.n3 = 0.164; % CT, email to NR, 9/22/04
                         pend.n4 = 0.158; % CT, email to NR, 9/22/04
                         pend.n5 = 0.158; % CT, email to NR, 9/22/04
                             sin = (pend.nn1-pend.nn0)/pend.ln;
                             si1 = (pend.n1-pend.n0)/pend.l1;
                             si2 = (pend.n3-pend.n2)/pend.l2;
                             si3 = (pend.n5-pend.n4)/pend.l3;
                             ffn = 0.807; % from Ian's data, 11/30/05
                             ff1 = 0.641; % from Ian's data, 11/30/05
                             ff2 = 0.608; % from Ian's data, 11/30/05
                            kffn = 1 + ffn*tan(asin(sin));
                            kff1 = 1 + ff1*tan(asin(si1));
                            kff2 = 1 + ff2*tan(asin(si2));
                        pend.kcn = 4*pi^2*(59/60)^2*61*kffn; % measured, T040229-12
                        pend.kc1 = 4*pi^2*(70/60)^2*50*kff1; % measured, T040229-12
                        pend.kc2 = 4*pi^2*(76/60)^2*39*kff2; % measured, T040229-12
                        pend.kxn = 1.0*10^5;  % as for middle
                        pend.kx1 = 1.0*10^5;  % Justin 11/29/05
                        pend.kx2 = 0.8*10^5;%6.5630e+004;  % Ian 12/09/05
                    pend.stage2 = 1;
                       pend.Inxy = 0.04157;
                       pend.Inyz = 0.000018;
                       pend.Inzx = 0.000104;
```

```
                            pend.I1xy = 0.02718;
                            pend.I1yz = 0.000109;
                            pend.I1zx = 0.000005;
                            pend.I2xy = 0;
                            pend.I2yz = 0;
                            pend.I2zx = 0;
                            pend.I3xy = 0;
                            pend.I3yz = 0;
                            pend.I3zx = 0;

% Output of quad_ref.m
%   longpitch1: [0.3976 0.4434 0.9894 1.1708]
%   longpitch2: [1.9828 2.8193 3.2322 3.4046]
%          yaw: [0.6846 1.4288 2.5359 3.1652]
%   transroll1: [0.4640 0.8079 1.0443 2.0957]
%   transroll2: [2.7519 3.3251 5.0459 25.1226]
%     vertical: [0.5814 2.3255 3.7326 17.3318]


% Stage2 output of Mathematica for comparison
%   N   f   type
%   1   0.3975644352339722  pitch3  pitch2
%   2   0.4433775561541977  x3  x2
%   3   0.4639809669357563  y3
%   4   0.5814249875070461  z3  z2
%   5   0.6846092071853637  yaw3    yaw1
%   6   0.8079415294919186  roll1   roll2   roll3
%   7   0.9894144499624277  x2  x3
%   8   1.044307242552718   y2  y3
%   9   1.170812753787754   pitch0  pitch1
%   10  1.428841235069959   yaw3    yaw1
%   11  1.982827019933515   x0  x2
%   12  2.095734967343347   y0  y1
%   13  2.325469494217848   z0  z1
%   14  2.535886058824618   yaw0    yaw2
%   15  2.751930679786737   roll1   roll0   roll3   roll2
%   16  2.819287432755158   pitch1  pitch0
%   17  3.165249402433968   yaw2    yaw1
%   18  3.232175696418479   pitch2  pitch3
%   19  3.325113056347424   roll1   y1  y0
%   20  3.404592789682972   x1  x0
%   21  3.732607589631745   z1  z0
%   22  5.045900440685648   roll0   y0
%   23  17.33177412817714   z2  z3
%   24  25.12255788363513   roll2   roll3
```

```
qm = [...
kcn+cn^2*kn+(g*mn3)/(2*cn*ln)-(cn*g*mn3)/(2*ln) 0 0 0 0 0
 0 kcn+cn^2*kn+(g*mn3)/(2*cn*ln)-(cn*g*mn3)/(2*ln) 0 0 0 0
 0 0 c1^2*k1+kc1+(g*m13)/(2*c1*l1)-(c1*g*m13)/(2*l1) 0 0 0
 0 0 0 c1^2*k1+kc1+(g*m13)/(2*c1*l1)-(c1*g*m13)/(2*l1) 0 0
 0 0 0 0 c2^2*k2+kc2+(g*m23)/(2*c2*l2)-(c2*g*m23)/(2*l2) 0
 0 0 0 0 0 c2^2*k2+kc2+(g*m23)/(2*c2*l2)-(c2*g*m23)/(2*l2)
];
cqsm = [...
0 -((2*cn*kn*ln-g*mn3)*(nn0-nn1))/(2*ln^2) (2*cn^3*kn*ln+g*mn3-cn^2*g*mn3)/(2*cn*ln) 0 0↙
-(2*cn^3*kn*ln*nn0-cn^2*g*mn3*nn0+g*mn3*nn1)/(2*cn*ln)
 0 ((2*cn*kn*ln-g*mn3)*(nn0-nn1))/(2*ln^2) (2*cn^3*kn*ln+g*mn3-cn^2*g*mn3)/(2*cn*ln) 0 0↙
(2*cn^3*kn*ln*nn0-cn^2*g*mn3*nn0+g*mn3*nn1)/(2*cn*ln)
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
];
xm = [...
(g*m13)/(c1*l1)+(g*mn3)/(cn*ln) 0 0 0 -((dn*g*m13)/(c1*l1))+(dm*g*mn3)/(cn*ln) 0 -((g*m13)↙
/(c1*l1)) 0 0 0 -((d0*g*m13)/(c1*l1)) 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 (g*m13)/(c1*l1)+(g*mn3)/(cn*ln)+(k1*(n0-n1)^2)/l1^2-(g*m13*(n0-n1)^2)/(2*c1*l1^3)+(k1*↙
(-n0+n1)^2)/l1^2-(g*m13*(-n0+n1)^2)/(2*c1*l1^3)+(kn*(nn0-nn1)^2)/ln^2-(g*mn3*(nn0-nn1)^2)/↙
(2*cn*ln^3)+(kn*(-nn0+nn1)^2)/ln^2-(g*mn3*(-nn0+nn1)^2)/(2*cn*ln^3) -((c1*k1*(n0-n1))/l1)+↙
(g*m13*(n0-n1))/(2*l1^2)-(c1*k1*(-n0+n1))/l1+(g*m13*(-n0+n1))/(2*l1^2)-(cn*kn*(nn0-nn1))↙
/ln+(g*mn3*(nn0-nn1))/(2*ln^2)-(cn*kn*(-nn0+nn1))/ln+(g*mn3*(-nn0+nn1))/(2*ln^2) 0 0↙
(dn*g*m13)/(c1*l1)-(dm*g*mn3)/(cn*ln)-(k1*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(2*l1^2)+↙
(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(4*c1*l1^3)-(k1*(-n0+n1)*(2*c1*l1*n0-2*dn*(-↙
n0+n1)))/(2*l1^2)+(g*m13*(-n0+n1)*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*c1*l1^3)+(kn*(nn0-nn1)*↙
(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln^2)-(g*mn3*(nn0-nn1)*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/↙
(4*cn*ln^3)+(kn*(-nn0+nn1)*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln^2)-(g*mn3*(-nn0+nn1)*(-2↙
*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*cn*ln^3) 0 -((g*m13)/(c1*l1))-(k1*(n0-n1)^2)/l1^2+(g*m13*↙
(n0-n1)^2)/(2*c1*l1^3)-(k1*(-n0+n1)^2)/l1^2+(g*m13*(-n0+n1)^2)/(2*c1*l1^3) (c1*k1*(n0-n1))↙
/l1-(g*m13*(n0-n1))/(2*l1^2)+(c1*k1*(-n0+n1))/l1-(g*m13*(-n0+n1))/(2*l1^2) 0 0 (d0*g*m13)/↙
(c1*l1)-(k1*(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/(2*l1^2)+(g*m13*(n0-n1)*(-2*d0*(n0-n1)↙
+2*c1*l1*n1))/(4*c1*l1^3)-(k1*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1^2)+(g*m13*(-↙
n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*c1*l1^3) 0 0 0 0 0 0 0 0 0 0 0 0
 0 -((c1*k1*(n0-n1))/l1)+(g*m13*(n0-n1))/(2*l1^2)-(c1*k1*(-n0+n1))/l1+(g*m13*(-n0+n1))/↙
(2*l1^2)-(cn*kn*(nn0-nn1))/ln+(g*mn3*(nn0-nn1))/(2*ln^2)-(cn*kn*(-nn0+nn1))/ln+(g*mn3*(-↙
nn0+nn1))/(2*ln^2) 2*c1^2*k1+2*cn^2*kn+(g*m13)/(c1*l1)-(c1*g*m13)/l1+(g*mn3)/(cn*ln)-↙
(cn*g*mn3)/ln 0 0 (c1*k1*(-2*c1*l1*n0-2*dn*(n0-n1)))/(2*l1)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-↙
n1)))/(4*l1^2)+(c1*k1*(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1)-(g*m13*(2*c1*l1*n0-2*dn*(-↙
n0+n1)))/(4*l1^2)-(cn*kn*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln)+(g*mn3*(-2*dm*(nn0-nn1)↙
+2*cn*ln*nn1))/(4*ln^2)-(cn*kn*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln)+(g*mn3*(-2↙
*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*ln^2) 0 (c1*k1*(n0-n1))/l1-(g*m13*(n0-n1))/(2*l1^2)+↙
(c1*k1*(-n0+n1))/l1-(g*m13*(-n0+n1))/(2*l1^2) -2*c1^2*k1-(g*m13)/(c1*l1)+(c1*g*m13)/l1 0 0↙
(c1*k1*(-2*d0*(n0-n1)+2*c1*l1*n1))/(2*l1)-(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)+↙
(c1*k1*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2) 0↙
0 0 0 0 0 0 0 0 0 0 0
 0 0 0 (g*mn3*(2*(nn0-nn1)*nn1+2*nn1^2))/(4*cn*ln)+(g*mn3*(2*nn1^2-2*nn1*(-nn0+nn1)))/↙
(4*cn*ln)+(k1*(n0-n1)^2*su^2)/l1^2-(g*m13*(n0-n1)^2*su^2)/(2*c1*l1^3)+(k1*(-n0+n1)^2*su^2)↙
/l1^2-(g*m13*(-n0+n1)^2*su^2)/(2*c1*l1^3)+(g*m13*(2*n0^2-2*n0*(n0-n1)+2*su^2))/(4*c1*l1)+↙
```

```
(g*m13*(2*n0^2+2*n0*(-n0+n1)+2*su^2))/(4*c1*l1) (g*m13*(-2*dn*n0+2*dn*(n0-n1)))/(4*c1*l1)+↙
(g*m13*(2*dn*n0+2*dn*(-n0+n1)))/(4*c1*l1)+(g*mn3*(2*dm*(nn0-nn1)+2*dm*nn1))/(4*cn*ln)+↙
(g*mn3*(-2*dm*nn1+2*dm*(-nn0+nn1)))/(4*cn*ln)+(c1*k1*(n0-n1)*su^2)/l1-(g*m13*(n0-n1)*su^2) ↙
/(2*l1^2)+(c1*k1*(-n0+n1)*su^2)/l1-(g*m13*(-n0+n1)*su^2)/(2*l1^2) 0 0 0 0 -((k1*(n0-n1) ↙
^2*su^2)/l1^2)+(g*m13*(n0-n1)^2*su^2)/(2*c1*l1^3)-(k1*(-n0+n1)^2*su^2)/l1^2+(g*m13*(- ↙
n0+n1)^2*su^2)/(2*c1*l1^3)+(g*m13*(-2*n0*n1-2*su^2))/(2*c1*l1) -((c1*k1*(n0-n1)*su^2)/l1)+↙
(g*m13*(n0-n1)*su^2)/(2*l1^2)-(c1*k1*(-n0+n1)*su^2)/l1+(g*m13*(-n0+n1)*su^2)/(2*l1^2) 0 0↙
0 0 0 0 0 0 0 0 0 0 0
 -((dn*g*m13)/(c1*l1))+(dm*g*mn3)/(cn*ln) 0 0 (g*m13*(-2*dn*n0+2*dn*(n0-n1)))/(4*c1*l1)+↙
(g*m13*(2*dn*n0+2*dn*(-n0+n1)))/(4*c1*l1)+(g*mn3*(2*dm*(nn0-nn1)+2*dm*nn1))/(4*cn*ln)+↙
(g*mn3*(-2*dm*nn1+2*dm*(-nn0+nn1)))/(4*cn*ln)+(c1*k1*(n0-n1)*su^2)/l1-(g*m13*(n0-n1)*su^2) ↙
/(2*l1^2)+(c1*k1*(-n0+n1)*su^2)/l1-(g*m13*(-n0+n1)*su^2)/(2*l1^2) (g*(2*dm^2+2*cn*dm*ln) ↙
*mn3)/(2*cn*ln)+2*c1^2*k1*su^2-(c1*g*m13*su^2)/l1+(g*m13*(2*dn^2+2*c1*dn*l1+2*su^2))/↙
(2*c1*l1) 0 (dn*g*m13)/(c1*l1) 0 0 -((c1*k1*(n0-n1)*su^2)/l1)+(g*m13*(n0-n1)*su^2)/↙
(2*l1^2)-(c1*k1*(-n0+n1)*su^2)/l1+(g*m13*(-n0+n1)*su^2)/(2*l1^2) -2*c1^2*k1*su^2+↙
(c1*g*m13*su^2)/l1+(g*m13*(2*d0*dn-2*su^2))/(2*c1*l1) 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 (dn*g*m13)/(c1*l1)-(dm*g*mn3)/(cn*ln)-(k1*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(2*l1^2)+↙
(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(4*c1*l1^3)-(k1*(-n0+n1)*(2*c1*l1*n0-2*dn*(- ↙
n0+n1)))/(2*l1^2)+(g*m13*(-n0+n1)*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*c1*l1^3)+(kn*(nn0-nn1)*↙
(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln^2)-(g*mn3*(nn0-nn1)*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/↙
(4*cn*ln^3)+(kn*(-nn0+nn1)*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln^2)-(g*mn3*(-nn0+nn1)*(-2↙
*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*cn*ln^3) (c1*k1*(-2*c1*l1*n0-2*dn*(n0-n1)))/(2*l1)-(g*m13*↙
(-2*c1*l1*n0-2*dn*(n0-n1)))/(4*l1^2)+(c1*k1*(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1)-(g*m13*↙
(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2)-(cn*kn*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln)+(g*mn3*↙
(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(4*ln^2)-(cn*kn*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln)+↙
(g*mn3*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*ln^2) 0 0 (k1*(-2*c1*l1*n0-2*dn*(n0-n1))^2)/↙
(4*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))^2)/(8*c1*l1^3)+(g*m13*↙
(2*dn^2+2*c1*dn*l1+2*n0^2-2*n0*(n0-n1)))/(4*c1*l1)+(k1*(2*c1*l1*n0-2*dn*(-n0+n1))^2)/↙
(4*l1^2)-(g*m13*(2*c1*l1*n0-2*dn*(-n0+n1))^2)/(8*c1*l1^3)+(g*m13*↙
(2*dn^2+2*c1*dn*l1+2*n0^2+2*n0*(-n0+n1)))/(4*c1*l1)+(kn*(-2*dm*(nn0-nn1)+2*cn*ln*nn1)^2)/↙
(4*ln^2)-(g*mn3*(-2*dm*(nn0-nn1)+2*cn*ln*nn1)^2)/(8*cn*ln^3)+(g*mn3*(2*dm^2+2*cn*dm*ln+2*↙
(nn0-nn1)*nn1+2*nn1^2))/(4*cn*ln)+(kn*(-2*cn*ln*nn1-2*dm*(-nn0+nn1))^2)/(4*ln^2)-(g*mn3*↙
(-2*cn*ln*nn1-2*dm*(-nn0+nn1))^2)/(8*cn*ln^3)+(g*mn3*(2*dm^2+2*cn*dm*ln+2*nn1^2-2*nn1*(-↙
nn0+nn1)))/(4*cn*ln) 0 -((dn*g*m13)/(c1*l1))+(k1*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/↙
(2*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(4*c1*l1^3)+(k1*(-n0+n1)*(2*c1*l1*n0-↙
2*dn*(-n0+n1)))/(2*l1^2)-(g*m13*(-n0+n1)*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*c1*l1^3) -(c1*k1*↙
(-2*c1*l1*n0-2*dn*(n0-n1)))/(2*l1)+(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1)))/(4*l1^2)-(c1*k1*↙
(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1)+(g*m13*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2) 0 0 (k1*(-2↙
*c1*l1*n0-2*dn*(n0-n1))*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-↙
n1))*(-2*d0*(n0-n1)+2*c1*l1*n1))/(8*c1*l1^3)+(g*m13*(2*d0*dn-2*n0*n1))/(2*c1*l1)+(k1*(-2↙
*c1*l1*n1-2*d0*(-n0+n1))*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2)-(g*m13*(-2*c1*l1*n1-2*d0*(-↙
n0+n1))*(2*c1*l1*n0-2*dn*(-n0+n1)))/(8*c1*l1^3) 0 0 0 0 0 0 0 0 0 0 0 0
 -((g*m13)/(c1*l1)) 0 0 0 (dn*g*m13)/(c1*l1) 0 (g*m13)/(c1*l1)+(g*m23)/(c2*l2) 0 0 0↙
(d0*g*m13)/(c1*l1)-(d1*g*m23)/(c2*l2) 0 -((g*m23)/(c2*l2)) 0 0 0 -((d2*g*m23)/(c2*l2)) 0 0↙
0 0 0 0 0
 0 -((g*m13)/(c1*l1))-(k1*(n0-n1)^2)/l1^2+(g*m13*(n0-n1)^2)/(2*c1*l1^3)-(k1*(-n0+n1)^2)↙
/l1^2+(g*m13*(-n0+n1)^2)/(2*c1*l1^3) (c1*k1*(n0-n1))/l1-(g*m13*(n0-n1))/(2*l1^2)+(c1*k1*(-↙
n0+n1))/l1-(g*m13*(-n0+n1))/(2*l1^2) 0 0 -((dn*g*m13)/(c1*l1))+(k1*(-2*c1*l1*n0-2*dn*(n0-↙
n1))*(n0-n1))/(2*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(4*c1*l1^3)+(k1*(-n0+n1)↙
*(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1^2)-(g*m13*(-n0+n1)*(2*c1*l1*n0-2*dn*(-n0+n1)))/↙
(4*c1*l1^3) 0 (g*m13)/(c1*l1)+(g*m23)/(c2*l2)+(k1*(n0-n1)^2)/l1^2-(g*m13*(n0-n1)^2)/↙
(2*c1*l1^3)+(k1*(-n0+n1)^2)/l1^2-(g*m13*(-n0+n1)^2)/(2*c1*l1^3)+(k2*(n2-n3)^2)/l2^2-↙
```

```
(g*m23*(n2-n3)^2)/(2*c2*l2^3)+(k2*(-n2+n3)^2)/l2^2-(g*m23*(-n2+n3)^2)/(2*c2*l2^3) -
((c1*k1*(n0-n1))/l1)+(g*m13*(n0-n1))/(2*l1^2)-(c1*k1*(-n0+n1))/l1+(g*m13*(-n0+n1))/
(2*l1^2)-(c2*k2*(n2-n3))/l2+(g*m23*(n2-n3))/(2*l2^2)-(c2*k2*(-n2+n3))/l2+(g*m23*(-n2+n3))/
(2*l2^2) 0 0 -((d0*g*m13)/(c1*l1))+(d1*g*m23)/(c2*l2)+(k1*(n0-n1)*(-2*d0*(n0-n1)
+2*c1*l1*n1))/(2*l1^2)-(g*m13*(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*c1*l1^3)+(k1*(-n0+n1)
*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1^2)-(g*m13*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/
(4*c1*l1^3)-(k2*(-2*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/(2*l2^2)+(g*m23*(-2*c2*l2*n2-2*d1*(n2-
n3))*(n2-n3))/(4*c2*l2^3)-(k2*(-n2+n3)*(2*c2*l2*n2-2*d1*(-n2+n3)))/(2*l2^2)+(g*m23*(-
n2+n3)*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*c2*l2^3) 0 -((g*m23)/(c2*l2))-(k2*(n2-n3)^2)/l2^2+
(g*m23*(n2-n3)^2)/(2*c2*l2^3)-(k2*(-n2+n3)^2)/l2^2+(g*m23*(-n2+n3)^2)/(2*c2*l2^3) (c2*k2*
(n2-n3))/l2-(g*m23*(n2-n3))/(2*l2^2)+(c2*k2*(-n2+n3))/l2-(g*m23*(-n2+n3))/(2*l2^2) 0 0
(d2*g*m23)/(c2*l2)-(k2*(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2^2)+(g*m23*(n2-n3)*(-2*d2*
(n2-n3)+2*c2*l2*n3))/(4*c2*l2^3)-(k2*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2^2)+
(g*m23*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*c2*l2^3) 0 0 0 0 0 0
 0 (c1*k1*(n0-n1))/l1-(g*m13*(n0-n1))/(2*l1^2)+(c1*k1*(-n0+n1))/l1-(g*m13*(-n0+n1))/
(2*l1^2) -2*c1^2*k1-(g*m13)/(c1*l1)+(c1*g*m13)/l1 0 0 -(c1*k1*(-2*c1*l1*n0-2*dn*(n0-n1)))/
(2*l1)+(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1)))/(4*l1^2)-(c1*k1*(2*c1*l1*n0-2*dn*(-n0+n1)))/
(2*l1)+(g*m13*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2) 0 -((c1*k1*(n0-n1))/l1)+(g*m13*(n0-n1))
/(2*l1^2)-(c1*k1*(-n0+n1))/l1+(g*m13*(-n0+n1))/(2*l1^2)-(c2*k2*(n2-n3))/l2+(g*m23*(n2-n3))
/(2*l2^2)-(c2*k2*(-n2+n3))/l2+(g*m23*(-n2+n3))/(2*l2^2) 2*c1^2*k1+2*c2^2*k2+(g*m13)/
(c1*l1)-(c1*g*m13)/l1+(g*m23)/(c2*l2)-(c2*g*m23)/l2 0 0 -(c1*k1*(-2*d0*(n0-n1)
+2*c1*l1*n1))/(2*l1)+(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)-(c1*k1*(-2*c1*l1*n1-2*d0*
(-n0+n1)))/(2*l1)+(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2)+(c2*k2*(-2*c2*l2*n2-2*d1*
(n2-n3)))/(2*l2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3)))/(4*l2^2)+(c2*k2*(2*c2*l2*n2-2*d1*(-
n2+n3)))/(2*l2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*l2^2) 0 (c2*k2*(n2-n3))/l2-(g*m23*
(n2-n3))/(2*l2^2)+(c2*k2*(-n2+n3))/l2-(g*m23*(-n2+n3))/(2*l2^2) -2*c2^2*k2-(g*m23)/(c2*l2)
+(c2*g*m23)/l2 0 0 (c2*k2*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2)-(g*m23*(-2*d2*(n2-n3)
+2*c2*l2*n3))/(4*l2^2)+(c2*k2*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2)-(g*m23*(-2*c2*l2*n3-
2*d2*(-n2+n3)))/(4*l2^2) 0 0 0 0 0 0
 0 0 0 -((k1*(n0-n1)^2*su^2)/l1^2)+(g*m13*(n0-n1)^2*su^2)/(2*c1*l1^3)-(k1*(-n0+n1)^2*su^2)
/l1^2+(g*m13*(-n0+n1)^2*su^2)/(2*c1*l1^3)+(g*m13*(-2*n0*n1-2*su^2))/(2*c1*l1) -((c1*k1*
(n0-n1)*su^2)/l1)+(g*m13*(n0-n1)*su^2)/(2*l1^2)-(c1*k1*(-n0+n1)*su^2)/l1+(g*m13*(-n0+n1)
*su^2)/(2*l1^2) 0 0 0 0 (k2*(n2-n3)^2*si^2)/l2^2-(g*m23*(n2-n3)^2*si^2)/(2*c2*l2^3)+(k2*(-
n2+n3)^2*si^2)/l2^2-(g*m23*(-n2+n3)^2*si^2)/(2*c2*l2^3)+(g*m23*(2*n2^2-2*n2*(n2-n3)
+2*si^2))/(4*c2*l2)+(g*m23*(2*n2^2+2*n2*(-n2+n3)+2*si^2))/(4*c2*l2)+(k1*(n0-n1)^2*su^2)
/l1^2-(g*m13*(n0-n1)^2*su^2)/(2*c1*l1^3)+(k1*(-n0+n1)^2*su^2)/l1^2-(g*m13*(-n0+n1)^2*su^2)
/(2*c1*l1^3)+(g*m13*(2*(n0-n1)*n1+2*n1^2+2*su^2))/(4*c1*l1)+(g*m13*(2*n1^2-2*n1*(-n0+n1)
+2*su^2))/(4*c1*l1) (g*m13*(2*d0*(n0-n1)+2*d0*n1))/(4*c1*l1)+(g*m13*(-2*d0*n1+2*d0*(-
n0+n1)))/(4*c1*l1)+(g*m23*(-2*d1*n2+2*d1*(n2-n3)))/(4*c2*l2)+(g*m23*(2*d1*n2+2*d1*(-
n2+n3)))/(4*c2*l2)+(c2*k2*(n2-n3)*si^2)/l2-(g*m23*(n2-n3)*si^2)/(2*l2^2)+(c2*k2*(-n2+n3)
*si^2)/l2-(g*m23*(-n2+n3)*si^2)/(2*l2^2)+(c1*k1*(n0-n1)*su^2)/l1-(g*m13*(n0-n1)*su^2)/
(2*l1^2)+(c1*k1*(-n0+n1)*su^2)/l1-(g*m13*(-n0+n1)*su^2)/(2*l1^2) 0 0 0 0 -((k2*(n2-n3)
^2*si^2)/l2^2)+(g*m23*(n2-n3)^2*si^2)/(2*c2*l2^3)-(k2*(-n2+n3)^2*si^2)/l2^2+(g*m23*(-
n2+n3)^2*si^2)/(2*c2*l2^3)+(g*m23*(-2*n2*n3-2*si^2))/(2*c2*l2) -((c2*k2*(n2-n3)*si^2)/l2)+
(g*m23*(n2-n3)*si^2)/(2*l2^2)-(c2*k2*(-n2+n3)*si^2)/l2+(g*m23*(-n2+n3)*si^2)/(2*l2^2) 0 0
0 0 0 0 0
 -((d0*g*m13)/(c1*l1)) 0 0 -((c1*k1*(n0-n1)*su^2)/l1)+(g*m13*(n0-n1)*su^2)/(2*l1^2)-
(c1*k1*(-n0+n1)*su^2)/l1+(g*m13*(-n0+n1)*su^2)/(2*l1^2) -2*c1^2*k1*su^2+(c1*g*m13*su^2)
/l1+(g*m13*(2*d0*dn-2*su^2))/(2*c1*l1) 0 (d0*g*m13)/(c1*l1)-(d1*g*m23)/(c2*l2) 0 0 (g*m13*
(2*d0*(n0-n1)+2*d0*n1))/(4*c1*l1)+(g*m13*(-2*d0*n1+2*d0*(-n0+n1)))/(4*c1*l1)+(g*m23*(-2
*d1*n2+2*d1*(n2-n3)))/(4*c2*l2)+(g*m23*(2*d1*n2+2*d1*(-n2+n3)))/(4*c2*l2)+(c2*k2*(n2-n3)
*si^2)/l2-(g*m23*(n2-n3)*si^2)/(2*l2^2)+(c2*k2*(-n2+n3)*si^2)/l2-(g*m23*(-n2+n3)*si^2)/
```

```
(2*l2^2)+(c1*k1*(n0-n1)*su^2)/l1-(g*m13*(n0-n1)*su^2)/(2*l1^2)+(c1*k1*(-n0+n1)*su^2)/l1-
(g*m13*(-n0+n1)*su^2)/(2*l1^2) 2*c2^2*k2*si^2-(c2*g*m23*si^2)/l2+(g*m23*
(2*d1^2+2*c2*d1*l2+2*si^2))/(2*c2*l2)+2*c1^2*k1*su^2-(c1*g*m13*su^2)/l1+(g*m13*
(2*d0^2+2*c1*d0*l1+2*su^2))/(2*c1*l1) 0 (d1*g*m23)/(c2*l2) 0 0 -((c2*k2*(n2-n3)*si^2)/l2)+
(g*m23*(n2-n3)*si^2)/(2*l2^2)-(c2*k2*(-n2+n3)*si^2)/l2+(g*m23*(-n2+n3)*si^2)/(2*l2^2) -2
*c2^2*k2*si^2+(c2*g*m23*si^2)/l2+(g*m23*(2*d1*d2-2*si^2))/(2*c2*l2) 0 0 0 0 0 0 0
 0 (d0*g*m13)/(c1*l1)-(k1*(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/(2*l1^2)+(g*m13*(n0-n1)*(-2
*d0*(n0-n1)+2*c1*l1*n1))/(4*c1*l1^3)-(k1*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1^2)+
(g*m13*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*c1*l1^3) (c1*k1*(-2*d0*(n0-n1)
+2*c1*l1*n1))/(2*l1)-(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)+(c1*k1*(-2*c1*l1*n1-2*d0*
(-n0+n1)))/(2*l1)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2) 0 0 (k1*(-2*c1*l1*n0-2*dn*
(n0-n1))*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(-2*d0*
(n0-n1)+2*c1*l1*n1))/(8*c1*l1^3)+(g*m13*(2*d0*dn-2*n0*n1))/(2*c1*l1)+(k1*(-2*c1*l1*n1-
2*d0*(-n0+n1))*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1))*
(2*c1*l1*n0-2*dn*(-n0+n1)))/(8*c1*l1^3) 0 -((d0*g*m13)/(c1*l1))+(d1*g*m23)/(c2*l2)+(k1*
(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/(2*l1^2)-(g*m13*(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/
(4*c1*l1^3)+(k1*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1^2)-(g*m13*(-n0+n1)*(-2
*c1*l1*n1-2*d0*(-n0+n1)))/(4*c1*l1^3)-(k2*(-2*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/(2*l2^2)+
(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/(4*c2*l2^3)-(k2*(-n2+n3)*(2*c2*l2*n2-2*d1*(-
n2+n3)))/(2*l2^2)+(g*m23*(-n2+n3)*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*c2*l2^3) -(c1*k1*(-2*d0*
(n0-n1)+2*c1*l1*n1))/(2*l1)+(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)-(c1*k1*(-2
*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1)+(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2)+(c2*k2*(-2
*c2*l2*n2-2*d1*(n2-n3)))/(2*l2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3)))/(4*l2^2)+(c2*k2*
(2*c2*l2*n2-2*d1*(-n2+n3)))/(2*l2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*l2^2) 0 0 (k1*(-2
*d0*(n0-n1)+2*c1*l1*n1)^2)/(4*l1^2)-(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1)^2)/(8*c1*l1^3)+
(g*m13*(2*d0^2+2*c1*d0*l1+2*(n0-n1)*n1+2*n1^2))/(4*c1*l1)+(k1*(-2*c1*l1*n1-2*d0*(-n0+n1))
^2)/(4*l1^2)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1))^2)/(8*c1*l1^3)+(g*m13*
(2*d0^2+2*c1*d0*l1+2*n1^2-2*n1*(-n0+n1)))/(4*c1*l1)+(k2*(-2*c2*l2*n2-2*d1*(n2-n3))^2)/
(4*l2^2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))^2)/(8*c2*l2^3)+(g*m23*
(2*d1^2+2*c2*d1*l2+2*n2^2-2*n2*(n2-n3)))/(4*c2*l2)+(k2*(2*c2*l2*n2-2*d1*(-n2+n3))^2)/
(4*l2^2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3))^2)/(8*c2*l2^3)+(g*m23*
(2*d1^2+2*c2*d1*l2+2*n2^2+2*n2*(-n2+n3)))/(4*c2*l2) 0 -((d1*g*m23)/(c2*l2))+(k2*(-2
*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/(2*l2^2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/
(4*c2*l2^3)+(k2*(-n2+n3)*(2*c2*l2*n2-2*d1*(-n2+n3)))/(2*l2^2)-(g*m23*(-n2+n3)*(2*c2*l2*n2-
2*d1*(-n2+n3)))/(4*c2*l2^3) -(c2*k2*(-2*c2*l2*n2-2*d1*(n2-n3)))/(2*l2)+(g*m23*(-2
*c2*l2*n2-2*d1*(n2-n3)))/(4*l2^2)-(c2*k2*(2*c2*l2*n2-2*d1*(-n2+n3)))/(2*l2)+(g*m23*
(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*l2^2) 0 0 (k2*(-2*c2*l2*n2-2*d1*(n2-n3))*(-2*d2*(n2-n3)
+2*c2*l2*n3))/(4*l2^2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))*(-2*d2*(n2-n3)+2*c2*l2*n3))/
(8*c2*l2^3)+(g*m23*(2*d1*d2-2*n2*n3))/(2*c2*l2)+(k2*(2*c2*l2*n2-2*d1*(-n2+n3))*(-2
*c2*l2*n3-2*d2*(-n2+n3)))/(4*l2^2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3))*(-2*c2*l2*n3-2*d2*(-
n2+n3)))/(8*c2*l2^3) 0 0 0 0 0 0
 0 0 0 0 0 0 -((g*m23)/(c2*l2)) 0 0 0 (d1*g*m23)/(c2*l2) 0 (g*m23)/(c2*l2)+(g*m3)/(c3*l3)
0 0 0 (d2*g*m23)/(c2*l2)-(d3*g*m3)/(c3*l3) 0 -((g*m3)/(c3*l3)) 0 0 0 -((d4*g*m3)/(c3*l3))
0
 0 0 0 0 0 0 0 -((g*m23)/(c2*l2))-(k2*(n2-n3)^2)/l2^2+(g*m23*(n2-n3)^2)/(2*c2*l2^3)-(k2*(-
n2+n3)^2)/l2^2+(g*m23*(-n2+n3)^2)/(2*c2*l2^3) (c2*k2*(n2-n3))/l2-(g*m23*(n2-n3))/(2*l2^2)+
(c2*k2*(-n2+n3))/l2-(g*m23*(-n2+n3))/(2*l2^2) 0 0 -((d1*g*m23)/(c2*l2))+(k2*(-2*c2*l2*n2-
2*d1*(n2-n3))*(n2-n3))/(2*l2^2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/(4*c2*l2^3)+
(k2*(-n2+n3)*(2*c2*l2*n2-2*d1*(-n2+n3)))/(2*l2^2)-(g*m23*(-n2+n3)*(2*c2*l2*n2-2*d1*(-
n2+n3)))/(4*c2*l2^3) 0 (g*m23)/(c2*l2)+(g*m3)/(c3*l3)+(k2*(n2-n3)^2)/l2^2-(g*m23*(n2-n3)
^2)/(2*c2*l2^3)+(k2*(-n2+n3)^2)/l2^2-(g*m23*(-n2+n3)^2)/(2*c2*l2^3)+(k3*(n4-n5)^2)/l3^2-
(g*m3*(n4-n5)^2)/(2*c3*l3^3)+(k3*(-n4+n5)^2)/l3^2-(g*m3*(-n4+n5)^2)/(2*c3*l3^3) -((c2*k2*
```

```
(n2-n3))/l2)+(g*m23*(n2-n3))/(2*l2^2)-(c2*k2*(-n2+n3))/l2+(g*m23*(-n2+n3))/(2*l2^2)-
(c3*k3*(n4-n5))/l3+(g*m3*(n4-n5))/(2*l3^2)-(c3*k3*(-n4+n5))/l3+(g*m3*(-n4+n5))/(2*l3^2) 0
0 -((d2*g*m23)/(c2*l2))+(d3*g*m3)/(c3*l3)+(k2*(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2^2)
-(g*m23*(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*c2*l2^3)+(k2*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-
n2+n3)))/(2*l2^2)-(g*m23*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*c2*l2^3)-(k3*(-2
*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/(2*l3^2)+(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/
(4*c3*l3^3)-(k3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-n4+n5)))/(2*l3^2)+(g*m3*(-n4+n5)*(2*c3*l3*n4-
2*d3*(-n4+n5)))/(4*c3*l3^3) 0 -((g*m3)/(c3*l3))-(k3*(n4-n5)^2)/l3^2+(g*m3*(n4-n5)^2)/
(2*c3*l3^3)-(k3*(-n4+n5)^2)/l3^2+(g*m3*(-n4+n5)^2)/(2*c3*l3^3) (c3*k3*(n4-n5))/l3-(g*m3*
(n4-n5))/(2*l3^2)+(c3*k3*(-n4+n5))/l3-(g*m3*(-n4+n5))/(2*l3^2) 0 0 (d4*g*m3)/(c3*l3)-(k3*
(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3^2)+(g*m3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/
(4*c3*l3^3)-(k3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3^2)+(g*m3*(-n4+n5)*(-2
*c3*l3*n5-2*d4*(-n4+n5)))/(4*c3*l3^3)
 0 0 0 0 0 0 0 (c2*k2*(n2-n3))/l2-(g*m23*(n2-n3))/(2*l2^2)+(c2*k2*(-n2+n3))/l2-(g*m23*(-
n2+n3))/(2*l2^2) -2*c2^2*k2-(g*m23)/(c2*l2)+(c2*g*m23)/l2 0 0 -(c2*k2*(-2*c2*l2*n2-2*d1*
(n2-n3)))/(2*l2)+(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3)))/(4*l2^2)-(c2*k2*(2*c2*l2*n2-2*d1*(-
n2+n3)))/(2*l2)+(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*l2^2) 0 -((c2*k2*(n2-n3))/l2)+
(g*m23*(n2-n3))/(2*l2^2)-(c2*k2*(-n2+n3))/l2+(g*m23*(-n2+n3))/(2*l2^2)-(c3*k3*(n4-n5))/l3+
(g*m3*(n4-n5))/(2*l3^2)-(c3*k3*(-n4+n5))/l3+(g*m3*(-n4+n5))/(2*l3^2) 2*c2^2*k2+2*c3^2*k3+
(g*m23)/(c2*l2)-(c2*g*m23)/l2+(g*m3)/(c3*l3)-(c3*g*m3)/l3 0 0 -(c2*k2*(-2*d2*(n2-n3)
+2*c2*l2*n3))/(2*l2)+(g*m23*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*l2^2)-(c2*k2*(-2*c2*l2*n3-2*d2*
(-n2+n3)))/(2*l2)+(g*m23*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*l2^2)+(c3*k3*(-2*c3*l3*n4-2*d3*
(n4-n5)))/(2*l3)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5)))/(4*l3^2)+(c3*k3*(2*c3*l3*n4-2*d3*(-
n4+n5)))/(2*l3)-(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*l3^2) 0 (c3*k3*(n4-n5))/l3-(g*m3*(n4-
n5))/(2*l3^2)+(c3*k3*(-n4+n5))/l3-(g*m3*(-n4+n5))/(2*l3^2) -2*c3^2*k3-(g*m3)/(c3*l3)+
(c3*g*m3)/l3 0 0 (c3*k3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3)-(g*m3*(-2*d4*(n4-n5)
+2*c3*l3*n5))/(4*l3^2)+(c3*k3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3)-(g*m3*(-2*c3*l3*n5-2*d4*
(-n4+n5)))/(4*l3^2)
 0 0 0 0 0 0 0 0 0 -((k2*(n2-n3)^2*si^2)/l2^2)+(g*m23*(n2-n3)^2*si^2)/(2*c2*l2^3)-(k2*(-
n2+n3)^2*si^2)/l2^2+(g*m23*(-n2+n3)^2*si^2)/(2*c2*l2^3)+(g*m23*(-2*n2*n3-2*si^2))/
(2*c2*l2) -((c2*k2*(n2-n3)*si^2)/l2)+(g*m23*(n2-n3)*si^2)/(2*l2^2)-(c2*k2*(-n2+n3)*si^2)
/l2+(g*m23*(-n2+n3)*si^2)/(2*l2^2) 0 0 0 0 (k2*(n2-n3)^2*si^2)/l2^2-(g*m23*(n2-n3)^2*si^2)
/(2*c2*l2^3)+(k2*(-n2+n3)^2*si^2)/l2^2-(g*m23*(-n2+n3)^2*si^2)/(2*c2*l2^3)+(g*m23*(2*(n2-
n3)*n3+2*n3^2+2*si^2))/(4*c2*l2)+(g*m23*(2*n3^2-2*n3*(-n2+n3)+2*si^2))/(4*c2*l2)+(k3*(n4-
n5)^2*sl^2)/l3^2-(g*m3*(n4-n5)^2*sl^2)/(2*c3*l3^3)+(k3*(-n4+n5)^2*sl^2)/l3^2-(g*m3*(-
n4+n5)^2*sl^2)/(2*c3*l3^3)+(g*m3*(2*n4^2-2*n4*(n4-n5)+2*sl^2))/(4*c3*l3)+(g*m3*
(2*n4^2+2*n4*(-n4+n5)+2*sl^2))/(4*c3*l3) (g*m23*(2*d2*(n2-n3)+2*d2*n3))/(4*c2*l2)+(g*m23*
(-2*d2*n3+2*d2*(-n2+n3)))/(4*c2*l2)+(g*m3*(-2*d3*n4+2*d3*(n4-n5)))/(4*c3*l3)+(g*m3*
(2*d3*n4+2*d3*(-n4+n5)))/(4*c3*l3)+(c2*k2*(n2-n3)*si^2)/l2-(g*m23*(n2-n3)*si^2)/(2*l2^2)+
(c2*k2*(-n2+n3)*si^2)/l2-(g*m23*(-n2+n3)*si^2)/(2*l2^2)+(c3*k3*(n4-n5)*sl^2)/l3-(g*m3*(n4-
n5)*sl^2)/(2*l3^2)+(c3*k3*(-n4+n5)*sl^2)/l3-(g*m3*(-n4+n5)*sl^2)/(2*l3^2) 0 0 0 0 -((k3*
(n4-n5)^2*sl^2)/l3^2)+(g*m3*(n4-n5)^2*sl^2)/(2*c3*l3^3)-(k3*(-n4+n5)^2*sl^2)/l3^2+(g*m3*(-
n4+n5)^2*sl^2)/(2*c3*l3^3)+(g*m3*(-2*n4*n5-2*sl^2))/(2*c3*l3) -((c3*k3*(n4-n5)*sl^2)/l3)+
(g*m3*(n4-n5)*sl^2)/(2*l3^2)-(c3*k3*(-n4+n5)*sl^2)/l3+(g*m3*(-n4+n5)*sl^2)/(2*l3^2) 0
 0 0 0 0 0 -((d2*g*m23)/(c2*l2)) 0 0 -((c2*k2*(n2-n3)*si^2)/l2)+(g*m23*(n2-n3)*si^2)/
(2*l2^2)-(c2*k2*(-n2+n3)*si^2)/l2+(g*m23*(-n2+n3)*si^2)/(2*l2^2) -2*c2^2*k2*si^2+
(c2*g*m23*si^2)/l2+(g*m23*(2*d1*d2-2*si^2))/(2*c2*l2) 0 (d2*g*m23)/(c2*l2)-(d3*g*m3)/
(c3*l3) 0 0 (g*m23*(2*d2*(n2-n3)+2*d2*n3))/(4*c2*l2)+(g*m23*(-2*d2*n3+2*d2*(-n2+n3)))/
(4*c2*l2)+(g*m3*(-2*d3*n4+2*d3*(n4-n5)))/(4*c3*l3)+(g*m3*(2*d3*n4+2*d3*(-n4+n5)))/
(4*c3*l3)+(c2*k2*(n2-n3)*si^2)/l2-(g*m23*(n2-n3)*si^2)/(2*l2^2)+(c2*k2*(-n2+n3)*si^2)/l2-
(g*m23*(-n2+n3)*si^2)/(2*l2^2)+(c3*k3*(n4-n5)*sl^2)/l3-(g*m3*(n4-n5)*sl^2)/(2*l3^2)+
(c3*k3*(-n4+n5)*sl^2)/l3-(g*m3*(-n4+n5)*sl^2)/(2*l3^2) 2*c2^2*k2*si^2-(c2*g*m23*si^2)/l2+
```

```
(g*m23*(2*d2^2+2*c2*d2*l2+2*si^2))/(2*c2*l2)+2*c3^2*k3*sl^2-(c3*g*m3*sl^2)/l3+(g*m3*
(2*d3^2+2*c3*d3*l3+2*sl^2))/(2*c3*l3) 0 (d3*g*m3)/(c3*l3) 0 0 -((c3*k3*(n4-n5)*sl^2)/l3)+
(g*m3*(n4-n5)*sl^2)/(2*l3^2)-(c3*k3*(-n4+n5)*sl^2)/l3+(g*m3*(-n4+n5)*sl^2)/(2*l3^2) -2
*c3^2*k3*sl^2+(c3*g*m3*sl^2)/l3+(g*m3*(2*d3*d4-2*sl^2))/(2*c3*l3) 0
 0 0 0 0 0 0 0 (d2*g*m23)/(c2*l2)-(k2*(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2^2)+(g*m23*
(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*c2*l2^3)-(k2*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/
(2*l2^2)+(g*m23*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*c2*l2^3) (c2*k2*(-2*d2*(n2-n3)
+2*c2*l2*n3))/(2*l2)-(g*m23*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*l2^2)+(c2*k2*(-2*c2*l2*n3-2*d2*
(-n2+n3)))/(2*l2)-(g*m23*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*l2^2) 0 0 (k2*(-2*c2*l2*n2-2*d1*
(n2-n3))*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*l2^2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))*(-2*d2*
(n2-n3)+2*c2*l2*n3))/(8*c2*l2^3)+(g*m23*(2*d1*d2-2*n2*n3))/(2*c2*l2)+(k2*(2*c2*l2*n2-2*d1*
(-n2+n3))*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*l2^2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3))*(-2
*c2*l2*n3-2*d2*(-n2+n3)))/(8*c2*l2^3) 0 -((d2*g*m23)/(c2*l2))+(d3*g*m3)/(c3*l3)+(k2*(n2-
n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2^2)-(g*m23*(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/
(4*c2*l2^3)+(k2*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2^2)-(g*m23*(-n2+n3)*(-2
*c2*l2*n3-2*d2*(-n2+n3)))/(4*c2*l2^3)-(k3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/(2*l3^2)+
(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/(4*c3*l3^3)-(k3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-
n4+n5)))/(2*l3^2)+(g*m3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*c3*l3^3) -(c2*k2*(-2*d2*
(n2-n3)+2*c2*l2*n3))/(2*l2)+(g*m23*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*l2^2)-(c2*k2*(-2
*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2)+(g*m23*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*l2^2)+(c3*k3*(-2
*c3*l3*n4-2*d3*(n4-n5)))/(2*l3)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5)))/(4*l3^2)+(c3*k3*
(2*c3*l3*n4-2*d3*(-n4+n5)))/(2*l3)-(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*l3^2) 0 0 (k2*(-2
*d2*(n2-n3)+2*c2*l2*n3)^2)/(4*l2^2)-(g*m23*(-2*d2*(n2-n3)+2*c2*l2*n3)^2)/(8*c2*l2^3)+
(g*m23*(2*d2^2+2*c2*d2*l2+2*(n2-n3)*n3+2*n3^2))/(4*c2*l2)+(k2*(-2*c2*l2*n3-2*d2*(-n2+n3))
^2)/(4*l2^2)-(g*m23*(-2*c2*l2*n3-2*d2*(-n2+n3))^2)/(8*c2*l2^3)+(g*m23*
(2*d2^2+2*c2*d2*l2+2*n3^2-2*n3*(-n2+n3)))/(4*c2*l2)+(k3*(-2*c3*l3*n4-2*d3*(n4-n5))^2)/
(4*l3^2)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))^2)/(8*c3*l3^3)+(g*m3*(2*d3^2+2*c3*d3*l3+2*n4^2-
2*n4*(n4-n5)))/(4*c3*l3)+(k3*(2*c3*l3*n4-2*d3*(-n4+n5))^2)/(4*l3^2)-(g*m3*(2*c3*l3*n4-
2*d3*(-n4+n5))^2)/(8*c3*l3^3)+(g*m3*(2*d3^2+2*c3*d3*l3+2*n4^2+2*n4*(-n4+n5)))/(4*c3*l3) 0
-((d3*g*m3)/(c3*l3))+(k3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/(2*l3^2)-(g*m3*(-2*c3*l3*n4-
2*d3*(n4-n5))*(n4-n5))/(4*c3*l3^3)+(k3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-n4+n5)))/(2*l3^2)-
(g*m3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*c3*l3^3) -(c3*k3*(-2*c3*l3*n4-2*d3*(n4-n5)))
/(2*l3)+(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5)))/(4*l3^2)-(c3*k3*(2*c3*l3*n4-2*d3*(-n4+n5)))/
(2*l3)+(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*l3^2) 0 0 (k3*(-2*c3*l3*n4-2*d3*(n4-n5))*(-2
*d4*(n4-n5)+2*c3*l3*n5))/(4*l3^2)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))*(-2*d4*(n4-n5)
+2*c3*l3*n5))/(8*c3*l3^3)+(g*m3*(2*d3*d4-2*n4*n5))/(2*c3*l3)+(k3*(2*c3*l3*n4-2*d3*(-
n4+n5))*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*l3^2)-(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5))*(-2
*c3*l3*n5-2*d4*(-n4+n5)))/(8*c3*l3^3)
 0 0 0 0 0 0 0 0 0 0 0 0 -((g*m3)/(c3*l3)) 0 0 0 (d3*g*m3)/(c3*l3) 0 (g*m3)/(c3*l3) 0 0 0
(d4*g*m3)/(c3*l3) 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 -((g*m3)/(c3*l3))-(k3*(n4-n5)^2)/l3^2+(g*m3*(n4-n5)^2)/
(2*c3*l3^3)-(k3*(-n4+n5)^2)/l3^2+(g*m3*(-n4+n5)^2)/(2*c3*l3^3) (c3*k3*(n4-n5))/l3-(g*m3*
(n4-n5))/(2*l3^2)+(c3*k3*(-n4+n5))/l3-(g*m3*(-n4+n5))/(2*l3^2) 0 0 -((d3*g*m3)/(c3*l3))+
(k3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/(2*l3^2)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))
/(4*c3*l3^3)+(k3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-n4+n5)))/(2*l3^2)-(g*m3*(-n4+n5)*(2*c3*l3*n4-
2*d3*(-n4+n5)))/(4*c3*l3^3) 0 (g*m3)/(c3*l3)+(k3*(n4-n5)^2)/l3^2-(g*m3*(n4-n5)^2)/
(2*c3*l3^3)+(k3*(-n4+n5)^2)/l3^2-(g*m3*(-n4+n5)^2)/(2*c3*l3^3) -((c3*k3*(n4-n5))/l3)+
(g*m3*(n4-n5))/(2*l3^2)-(c3*k3*(-n4+n5))/l3+(g*m3*(-n4+n5))/(2*l3^2) 0 0 -((d4*g*m3)/
(c3*l3))+(k3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3^2)-(g*m3*(n4-n5)*(-2*d4*(n4-n5)
+2*c3*l3*n5))/(4*c3*l3^3)+(k3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3^2)-(g*m3*(-
n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*c3*l3^3)
 0 0 0 0 0 0 0 0 0 0 0 0 0 (c3*k3*(n4-n5))/l3-(g*m3*(n4-n5))/(2*l3^2)+(c3*k3*(-n4+n5))/l3-
```

```
(g*m3*(-n4+n5))/(2*l3^2) -2*c3^2*k3-(g*m3)/(c3*l3)+(c3*g*m3)/l3 0 0 -(c3*k3*(-2*c3*l3*n4-2*d3*(n4-n5)))/(2*l3)+(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5)))/(4*l3^2)-(c3*k3*(2*c3*l3*n4-2*d3*(-n4+n5)))/(2*l3)+(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*l3^2) 0 -((c3*k3*(n4-n5))/l3)+(g*m3*(n4-n5))/(2*l3^2)-(c3*k3*(-n4+n5))/l3+(g*m3*(-n4+n5))/(2*l3^2) 2*c3^2*k3+(g*m3)/(c3*l3)-(c3*g*m3)/l3 0 0 -(c3*k3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3)+(g*m3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*l3^2)-(c3*k3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3)+(g*m3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*l3^2)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -((k3*(n4-n5)^2*sl^2)/l3^2)+(g*m3*(n4-n5)^2*sl^2)/(2*c3*l3^3)-(k3*(-n4+n5)^2*sl^2)/l3^2+(g*m3*(-n4+n5)^2*sl^2)/(2*c3*l3^3)+(g*m3*(-2*n4*n5-2*sl^2))/(2*c3*l3) -((c3*k3*(n4-n5)*sl^2)/l3)+(g*m3*(n4-n5)*sl^2)/(2*l3^2)-(c3*k3*(-n4+n5)*sl^2)/l3+(g*m3*(-n4+n5)*sl^2)/(2*l3^2) 0 0 0 0 (k3*(n4-n5)^2*sl^2)/l3^2-(g*m3*(n4-n5)^2*sl^2)/(2*c3*l3^3)+(k3*(-n4+n5)^2*sl^2)/l3^2-(g*m3*(-n4+n5)^2*sl^2)/(2*c3*l3^3)+(g*m3*(2*(n4-n5)*n5+2*n5^2+2*sl^2))/(4*c3*l3)+(g*m3*(2*n5^2-2*n5*(-n4+n5)+2*sl^2))/(4*c3*l3)+(g*m3*(2*d4*(n4-n5)+2*d4*n5))/(4*c3*l3)+(g*m3*(-2*d4*n5+2*d4*(-n4+n5)))/(4*c3*l3)+(c3*k3*(n4-n5)*sl^2)/l3-(g*m3*(n4-n5)*sl^2)/(2*l3^2)+(c3*k3*(-n4+n5)*sl^2)/l3-(g*m3*(-n4+n5)*sl^2)/(2*l3^2) 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 -((d4*g*m3)/(c3*l3)) 0 0 -((c3*k3*(n4-n5)*sl^2)/l3)+(g*m3*(n4-n5)*sl^2)/(2*l3^2)-(c3*k3*(-n4+n5)*sl^2)/l3+(g*m3*(-n4+n5)*sl^2)/(2*l3^2) -2*c3^2*k3*sl^2+(c3*g*m3*sl^2)/l3+(g*m3*(2*d3*d4-2*sl^2))/(2*c3*l3) 0 (d4*g*m3)/(c3*l3) 0 0 (g*m3*(2*d4*(n4-n5)+2*d4*n5))/(4*c3*l3)+(g*m3*(-2*d4*n5+2*d4*(-n4+n5)))/(4*c3*l3)+(c3*k3*(n4-n5)*sl^2)/l3-(g*m3*(n4-n5)*sl^2)/(2*l3^2)+(c3*k3*(-n4+n5)*sl^2)/l3-(g*m3*(-n4+n5)*sl^2)/(2*l3^2) 2*c3^2*k3*sl^2-(c3*g*m3*sl^2)/l3+(g*m3*(2*d4^2+2*c3*d4*l3+2*sl^2))/(2*c3*l3) 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 (d4*g*m3)/(c3*l3)-(k3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3^2)+(g*m3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*c3*l3^3)-(k3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3^2)+(g*m3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*c3*l3^3) (c3*k3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3)-(g*m3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*l3^2)+(c3*k3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3)-(g*m3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*l3^2) 0 0 (k3*(-2*c3*l3*n4-2*d3*(n4-n5))*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*l3^2)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))*(-2*d4*(n4-n5)+2*c3*l3*n5))/(8*c3*l3^3)+(g*m3*(2*d3*d4-2*n4*n5))/(2*c3*l3)+(k3*(2*c3*l3*n4-2*d3*(-n4+n5))*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*l3^2)-(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5))*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(8*c3*l3^3) 0 -((d4*g*m3)/(c3*l3))+(k3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3^2)-(g*m3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*c3*l3^3)+(k3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3^2)-(g*m3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*c3*l3^3) -(c3*k3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3)+(g*m3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*l3^2)-(c3*k3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3)+(g*m3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*l3^2) 0 0 (k3*(-2*d4*(n4-n5)+2*c3*l3*n5)^2)/(4*l3^2)-(g*m3*(-2*d4*(n4-n5)+2*c3*l3*n5)^2)/(8*c3*l3^3)+(g*m3*(2*d4^2+2*c3*d4*l3+2*(n4-n5)*n5+2*n5^2))/(4*c3*l3)+(k3*(-2*c3*l3*n5-2*d4*(-n4+n5))^2)/(4*l3^2)-(g*m3*(-2*c3*l3*n5-2*d4*(-n4+n5))^2)/(8*c3*l3^3)+(g*m3*(2*d4^2+2*c3*d4*l3+2*n5^2-2*n5*(-n4+n5)))/(4*c3*l3)
];
km = [...
mn 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 mn 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 mn 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 Inz Inyz Inzx 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 Inyz Iny Inxy 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 Inzx Inxy Inx 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 m1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 m1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 m1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 I1z I1yz I1zx 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 I1yz I1y I1xy 0 0 0 0 0 0 0 0 0 0 0
```

```
 0 0 0 0 0 0 0 0 0 I1zx I1xy I1x 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 m2 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 m2 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 m2 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I2z I2yz I2zx 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I2yz I2y I2xy 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I2zx I2xy I2x 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 m3 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 m3 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 m3 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I3z I3yz I3zx
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I3yz I3y I3xy
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I3zx I3xy I3x
];
cqxm = [...
0 (cn*kn*(nn0-nn1))/ln-(g*mn3*(nn0-nn1))/(2*ln^2) -(cn^2*kn)-(g*mn3)/(2*cn*ln)+(cn*g*mn3)/↙
(2*ln) 0 0 (g*mn3*nn1)/(2*cn*ln)+(cn*kn*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln)-(g*mn3*(-2↙
*dm*(nn0-nn1)+2*cn*ln*nn1))/(4*ln^2) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 (cn*kn*(-nn0+nn1))/ln-(g*mn3*(-nn0+nn1))/(2*ln^2) -(cn^2*kn)-(g*mn3)/(2*cn*ln)+↙
(cn*g*mn3)/(2*ln) 0 0 -(g*mn3*nn1)/(2*cn*ln)+(cn*kn*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln)↙
-(g*mn3*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*ln^2) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 -((c1*k1*(n0-n1))/l1)+(g*m13*(n0-n1))/(2*l1^2) c1^2*k1+(g*m13)/(2*c1*l1)-(c1*g*m13)/↙
(2*l1) 0 0 (g*m13*(-2*n0+2*(n0-n1)))/(4*c1*l1)+(c1*k1*(-2*c1*l1*n0-2*dn*(n0-n1)))/(2*l1)-↙
(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1)))/(4*l1^2) 0 (c1*k1*(n0-n1))/l1-(g*m13*(n0-n1))/(2*l1^2)↙
-(c1^2*k1)-(g*m13)/(2*c1*l1)+(c1*g*m13)/(2*l1) 0 0 (g*m13*n1)/(2*c1*l1)+(c1*k1*(-2*d0*(n0-↙
n1)+2*c1*l1*n1))/(2*l1)-(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2) 0 0 0 0 0 0 0 0 0 0↙
0
 0 -((c1*k1*(-n0+n1))/l1)+(g*m13*(-n0+n1))/(2*l1^2) c1^2*k1+(g*m13)/(2*c1*l1)-(c1*g*m13)/↙
(2*l1) 0 0 (g*m13*(2*n0+2*(-n0+n1)))/(4*c1*l1)+(c1*k1*(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1)-↙
(g*m13*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2) 0 (c1*k1*(-n0+n1))/l1-(g*m13*(-n0+n1))/↙
(2*l1^2) -(c1^2*k1)-(g*m13)/(2*c1*l1)+(c1*g*m13)/(2*l1) 0 0 -(g*m13*n1)/(2*c1*l1)+(c1*k1*↙
(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2) 0 0 0 0 0↙
0 0 0 0 0 0 0
 0 0 0 0 0 0 0 -((c2*k2*(n2-n3))/l2)+(g*m23*(n2-n3))/(2*l2^2) c2^2*k2+(g*m23)/(2*c2*l2)-↙
(c2*g*m23)/(2*l2) 0 0 (g*m23*(-2*n2+2*(n2-n3)))/(4*c2*l2)+(c2*k2*(-2*c2*l2*n2-2*d1*(n2-↙
n3)))/(2*l2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3)))/(4*l2^2) 0 (c2*k2*(n2-n3))/l2-(g*m23*(n2-↙
n3))/(2*l2^2) -(c2^2*k2)-(g*m23)/(2*c2*l2)+(c2*g*m23)/(2*l2) 0 0 (g*m23*n3)/(2*c2*l2)+↙
(c2*k2*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2)-(g*m23*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*l2^2) 0 0↙
0 0 0 0
 0 0 0 0 0 0 0 -((c2*k2*(-n2+n3))/l2)+(g*m23*(-n2+n3))/(2*l2^2) c2^2*k2+(g*m23)/(2*c2*l2)-↙
(c2*g*m23)/(2*l2) 0 0 (g*m23*(2*n2+2*(-n2+n3)))/(4*c2*l2)+(c2*k2*(2*c2*l2*n2-2*d1*(-↙
n2+n3)))/(2*l2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*l2^2) 0 (c2*k2*(-n2+n3))/l2-(g*m23*↙
(-n2+n3))/(2*l2^2) -(c2^2*k2)-(g*m23)/(2*c2*l2)+(c2*g*m23)/(2*l2) 0 0 -(g*m23*n3)/↙
(2*c2*l2)+(c2*k2*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2)-(g*m23*(-2*c2*l2*n3-2*d2*(-n2+n3)))/↙
(4*l2^2) 0 0 0 0 0 0
];
cxsm = [...
-((g*mn3)/(cn*ln)) 0 0 0 0 0
 0 (g*mn3*(-ln^2+(nn0-nn1)^2)-2*cn*kn*ln*(nn0-nn1)^2)/(cn*ln^3) 0 0 0 -(((2*cn*kn*ln-↙
g*mn3)*nn0*(nn0-nn1))/ln^2)
 0 0 (-2*cn^3*kn*ln-g*mn3+cn^2*g*mn3)/(cn*ln) 0 0 0
 0 0 0 -((g*mn3*nn0*nn1)/(cn*ln)) 0 0
```

```
 -((dm*g*mn3)/(cn*ln)) 0 0 0 0 0
 0 (dm*(g*mn3*(ln^2-(nn0-nn1)^2)+2*cn*kn*ln*(nn0-nn1)^2)-cn*ln*(2*cn*kn*ln-g*mn3)*(nn0-↙
nn1)*nn1)/(cn*ln^3) 0 0 0 (nn0*(-2*cn^3*kn*ln^2*nn1-g*ln*mn3*nn1+cn*dm*g*mn3*(-nn0+nn1)↙
+cn^2*ln*(2*dm*kn*(nn0-nn1)+g*mn3*nn1)))/(cn*ln^2)
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
];
```

```
qm = [...
kcn+cn^2*kn+(g*mn3)/(2*cn*ln)-(cn*g*mn3)/(2*ln) 0 0 0 0 0 0 0 0 0 0 0
 0 kxn+(g*mn3)/(2*cn*ln) 0 0 0 0 0 0 0 0 0 0
 0 0 kcn+cn^2*kn+(g*mn3)/(2*cn*ln)-(cn*g*mn3)/(2*ln) 0 0 0 0 0 0 0 0 0
 0 0 0 kxn+(g*mn3)/(2*cn*ln) 0 0 0 0 0 0 0 0
 0 0 0 0 c1^2*k1+kc1+(g*m13)/(2*c1*l1)-(c1*g*m13)/(2*l1) 0 0 0 0 0 0 0
 0 0 0 0 0 kx1+(g*m13)/(2*c1*l1) 0 0 0 0 0 0
 0 0 0 0 0 0 c1^2*k1+kc1+(g*m13)/(2*c1*l1)-(c1*g*m13)/(2*l1) 0 0 0 0 0
 0 0 0 0 0 0 0 kx1+(g*m13)/(2*c1*l1) 0 0 0 0
 0 0 0 0 0 0 0 0 c2^2*k2+kc2+(g*m23)/(2*c2*l2)-(c2*g*m23)/(2*l2) 0 0 0
 0 0 0 0 0 0 0 0 0 kx2+(g*m23)/(2*c2*l2) 0 0
 0 0 0 0 0 0 0 0 0 0 c2^2*k2+kc2+(g*m23)/(2*c2*l2)-(c2*g*m23)/(2*l2) 0
 0 0 0 0 0 0 0 0 0 0 0 kx2+(g*m23)/(2*c2*l2)
];
cqsm = [...
0 -((2*cn*kn*ln-g*mn3)*(nn0-nn1))/(2*ln^2) (2*cn^3*kn*ln+g*mn3-cn^2*g*mn3)/(2*cn*ln) 0 0↙
-(2*cn^3*kn*ln*nn0-cn^2*g*mn3*nn0+g*mn3*nn1)/(2*cn*ln)
 (g*mn3)/(2*cn*ln) 0 0 (g*mn3*nn1)/(2*cn*ln) -(g*mn3)/2 0
 0 ((2*cn*kn*ln-g*mn3)*(nn0-nn1))/(2*ln^2) (2*cn^3*kn*ln+g*mn3-cn^2*g*mn3)/(2*cn*ln) 0 0↙
(2*cn^3*kn*ln*nn0-cn^2*g*mn3*nn0+g*mn3*nn1)/(2*cn*ln)
 (g*mn3)/(2*cn*ln) 0 0 -(g*mn3*nn1)/(2*cn*ln) -(g*mn3)/2 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
];
xm = [...
(g*m13)/(c1*l1)+(g*mn3)/(cn*ln) 0 0 0 -((dn*g*m13)/(c1*l1))+(dm*g*mn3)/(cn*ln) 0 -((g*m13)↙
/(c1*l1)) 0 0 0 -((d0*g*m13)/(c1*l1)) 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 (g*m13)/(c1*l1)+(g*mn3)/(cn*ln)+(k1*(n0-n1)^2)/l1^2-(g*m13*(n0-n1)^2)/(2*c1*l1^3)+(k1*↙
(-n0+n1)^2)/l1^2-(g*m13*(-n0+n1)^2)/(2*c1*l1^3)+(kn*(nn0-nn1)^2)/ln^2-(g*mn3*(nn0-nn1)^2)/↙
(2*cn*ln^3)+(kn*(-nn0+nn1)^2)/ln^2-(g*mn3*(-nn0+nn1)^2)/(2*cn*ln^3) -((c1*k1*(n0-n1))/l1)+↙
(g*m13*(n0-n1))/(2*l1^2)-(c1*k1*(-n0+n1))/l1+(g*m13*(-n0+n1))/(2*l1^2)-(cn*kn*(nn0-nn1))↙
/ln+(g*mn3*(nn0-nn1))/(2*ln^2)-(cn*kn*(-nn0+nn1))/ln+(g*mn3*(-nn0+nn1))/(2*ln^2) 0 0↙
(dn*g*m13)/(c1*l1)-(dm*g*mn3)/(cn*ln)-(k1*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(2*l1^2)+↙
(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(4*c1*l1^3)-(k1*(-n0+n1)*(2*c1*l1*n0-2*dn*(-↙
n0+n1)))/(2*l1^2)+(g*m13*(-n0+n1)*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*c1*l1^3)+(kn*(nn0-nn1)*↙
(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln^2)-(g*mn3*(nn0-nn1)*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/↙
(4*cn*ln^3)+(kn*(-nn0+nn1)*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln^2)-(g*mn3*(-nn0+nn1)*(-2↙
*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*cn*ln^3) 0 -((g*m13)/(c1*l1))-(k1*(n0-n1)^2)/l1^2+(g*m13*↙
(n0-n1)^2)/(2*c1*l1^3)-(k1*(-n0+n1)^2)/l1^2+(g*m13*(-n0+n1)^2)/(2*c1*l1^3) (c1*k1*(n0-n1))↙
/l1-(g*m13*(n0-n1))/(2*l1^2)+(c1*k1*(-n0+n1))/l1-(g*m13*(-n0+n1))/(2*l1^2) 0 0 (d0*g*m13)/↙
(c1*l1)-(k1*(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/(2*l1^2)+(g*m13*(n0-n1)*(-2*d0*(n0-n1)↙
+2*c1*l1*n1))/(4*c1*l1^3)-(k1*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1^2)+(g*m13*(-↙
n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*c1*l1^3) 0 0 0 0 0 0 0 0 0 0 0
 0 -((c1*k1*(n0-n1))/l1)+(g*m13*(n0-n1))/(2*l1^2)-(c1*k1*(-n0+n1))/l1+(g*m13*(-n0+n1))/↙
(2*l1^2)-(cn*kn*(nn0-nn1))/ln+(g*mn3*(nn0-nn1))/(2*ln^2)-(cn*kn*(-nn0+nn1))/ln+(g*mn3*(-↙
nn0+nn1))/(2*ln^2) 2*c1^2*k1+2*cn^2*kn+(g*m13)/(c1*l1)-(c1*g*m13)/l1+(g*mn3)/(cn*ln)-↙
```

```
(cn*g*mn3)/ln 0 0 (c1*k1*(-2*c1*l1*n0-2*dn*(n0-n1)))/(2*l1)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-
n1)))/(4*l1^2)+(c1*k1*(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1)-(g*m13*(2*c1*l1*n0-2*dn*(-
n0+n1)))/(4*l1^2)-(cn*kn*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln)+(g*mn3*(-2*dm*(nn0-nn1)
+2*cn*ln*nn1))/(4*ln^2)-(cn*kn*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln)+(g*mn3*(-2
*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*ln^2) 0 (c1*k1*(n0-n1))/l1-(g*m13*(n0-n1))/(2*l1^2)+
(c1*k1*(-n0+n1))/l1-(g*m13*(-n0+n1))/(2*l1^2) -2*c1^2*k1-(g*m13)/(c1*l1)+(c1*g*m13)/l1 0 0
(c1*k1*(-2*d0*(n0-n1)+2*c1*l1*n1))/(2*l1)-(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)+
(c1*k1*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2) 0
0 0 0 0 0 0 0 0 0 0 0
 0 0 0 (g*mn3*(2*(nn0-nn1)*nn1+2*nn1^2))/(4*cn*ln)+(g*mn3*(2*nn1^2-2*nn1*(-nn0+nn1)))/
(4*cn*ln)+(k1*(n0-n1)^2*su^2)/l1^2-(g*m13*(n0-n1)^2*su^2)/(2*c1*l1^3)+(k1*(-n0+n1)^2*su^2)
/l1^2-(g*m13*(-n0+n1)^2*su^2)/(2*c1*l1^3)+(g*m13*(2*n0^2-2*n0*(n0-n1)+2*su^2))/(4*c1*l1)+
(g*m13*(2*n0^2+2*n0*(-n0+n1)+2*su^2))/(4*c1*l1) (g*m13*(-2*dn*n0+2*dn*(n0-n1)))/(4*c1*l1)+
(g*m13*(2*dn*n0+2*dn*(-n0+n1)))/(4*c1*l1)+(g*mn3*(2*dm*(nn0-nn1)+2*dm*nn1))/(4*cn*ln)+
(g*mn3*(-2*dm*nn1+2*dm*(-nn0+nn1)))/(4*cn*ln)+(c1*k1*(n0-n1)*su^2)/l1-(g*m13*(n0-n1)*su^2)
/(2*l1^2)+(c1*k1*(-n0+n1)*su^2)/l1-(g*m13*(-n0+n1)*su^2)/(2*l1^2) 0 0 0 0 -((k1*(n0-n1)
^2*su^2)/l1^2)+(g*m13*(n0-n1)^2*su^2)/(2*c1*l1^3)-(k1*(-n0+n1)^2*su^2)/l1^2+(g*m13*(-
n0+n1)^2*su^2)/(2*c1*l1^3)+(g*m13*(-2*n0*n1-2*su^2))/(2*c1*l1) -((c1*k1*(n0-n1)*su^2)/l1)+
(g*m13*(n0-n1)*su^2)/(2*l1^2)-(c1*k1*(-n0+n1)*su^2)/l1+(g*m13*(-n0+n1)*su^2)/(2*l1^2) 0 0
0 0 0 0 0 0 0 0 0 0 0
 -((dn*g*m13)/(c1*l1))+(dm*g*mn3)/(cn*ln) 0 0 (g*m13*(-2*dn*n0+2*dn*(n0-n1)))/(4*c1*l1)+
(g*m13*(2*dn*n0+2*dn*(-n0+n1)))/(4*c1*l1)+(g*mn3*(2*dm*(nn0-nn1)+2*dm*nn1))/(4*cn*ln)+
(g*mn3*(-2*dm*nn1+2*dm*(-nn0+nn1)))/(4*cn*ln)+(c1*k1*(n0-n1)*su^2)/l1-(g*m13*(n0-n1)*su^2)
/(2*l1^2)+(c1*k1*(-n0+n1)*su^2)/l1-(g*m13*(-n0+n1)*su^2)/(2*l1^2) (g*(2*dm^2+2*cn*dm*ln)
*mn3)/(2*cn*ln)+2*c1^2*k1*su^2-(c1*g*m13*su^2)/l1+(g*m13*(2*dn^2+2*c1*dn*l1+2*su^2))/
(2*c1*l1) 0 (dn*g*m13)/(c1*l1) 0 0 -((c1*k1*(n0-n1)*su^2)/l1)+(g*m13*(n0-n1)*su^2)/
(2*l1^2)-(c1*k1*(-n0+n1)*su^2)/l1+(g*m13*(-n0+n1)*su^2)/(2*l1^2) -2*c1^2*k1*su^2+
(c1*g*m13*su^2)/l1+(g*m13*(2*d0*dn-2*su^2))/(2*c1*l1) 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 (dn*g*m13)/(c1*l1)-(dm*g*mn3)/(cn*ln)-(k1*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(2*l1^2)+
(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(4*c1*l1^3)-(k1*(-n0+n1)*(2*c1*l1*n0-2*dn*(-
n0+n1)))/(2*l1^2)+(g*m13*(-n0+n1)*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*c1*l1^3)+(kn*(nn0-nn1)*
(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln^2)-(g*mn3*(nn0-nn1)*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/
(4*cn*ln^3)+(kn*(-nn0+nn1)*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln^2)-(g*mn3*(-nn0+nn1)*(-2
*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*cn*ln^3) (c1*k1*(-2*c1*l1*n0-2*dn*(n0-n1)))/(2*l1)-(g*m13*
(-2*c1*l1*n0-2*dn*(n0-n1)))/(4*l1^2)+(c1*k1*(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1)-(g*m13*
(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2)-(cn*kn*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln)+(g*mn3*
(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(4*ln^2)-(cn*kn*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln)+
(g*mn3*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*ln^2) 0 0 (k1*(-2*c1*l1*n0-2*dn*(n0-n1))^2)/
(4*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))^2)/(8*c1*l1^3)+(g*m13*
(2*dn^2+2*c1*dn*l1+2*n0^2-2*n0*(n0-n1)))/(4*c1*l1)+(k1*(2*c1*l1*n0-2*dn*(-n0+n1))^2)/
(4*l1^2)-(g*m13*(2*c1*l1*n0-2*dn*(-n0+n1))^2)/(8*c1*l1^3)+(g*m13*
(2*dn^2+2*c1*dn*l1+2*n0^2+2*n0*(-n0+n1)))/(4*c1*l1)+(kn*(-2*dm*(nn0-nn1)+2*cn*ln*nn1)^2)/
(4*ln^2)-(g*mn3*(-2*dm*(nn0-nn1)+2*cn*ln*nn1)^2)/(8*cn*ln^3)+(g*mn3*(2*dm^2+2*cn*dm*ln+2*
(nn0-nn1)*nn1+2*nn1^2))/(4*cn*ln)+(kn*(-2*cn*ln*nn1-2*dm*(-nn0+nn1))^2)/(4*ln^2)-(g*mn3*
(-2*cn*ln*nn1-2*dm*(-nn0+nn1))^2)/(8*cn*ln^3)+(g*mn3*(2*dm^2+2*cn*dm*ln+2*nn1^2-2*nn1*(-
nn0+nn1)))/(4*cn*ln) 0 -((dn*g*m13)/(c1*l1))+(k1*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/
(2*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(4*c1*l1^3)+(k1*(-n0+n1)*(2*c1*l1*n0-
2*dn*(-n0+n1)))/(2*l1^2)-(g*m13*(-n0+n1)*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*c1*l1^3) -(c1*k1*
(-2*c1*l1*n0-2*dn*(n0-n1)))/(2*l1)+(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1)))/(4*l1^2)-(c1*k1*
(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1)+(g*m13*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2) 0 0 (k1*(-2
*c1*l1*n0-2*dn*(n0-n1))*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-
n1))*(-2*d0*(n0-n1)+2*c1*l1*n1))/(8*c1*l1^3)+(g*m13*(2*d0*dn-2*n0*n1))/(2*c1*l1)+(k1*(-2
```

```
*c1*l1*n1-2*d0*(-n0+n1))*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2)-(g*m13*(-2*c1*l1*n1-2*d0*(-
n0+n1))*(2*c1*l1*n0-2*dn*(-n0+n1)))/(8*c1*l1^3) 0 0 0 0 0 0 0 0 0 0 0 0
 -((g*m13)/(c1*l1)) 0 0 0 (dn*g*m13)/(c1*l1) 0 (g*m13)/(c1*l1)+(g*m23)/(c2*l2) 0 0 0
(d0*g*m13)/(c1*l1)-(d1*g*m23)/(c2*l2) 0 -((g*m23)/(c2*l2)) 0 0 0 -((d2*g*m23)/(c2*l2)) 0 0
0 0 0 0 0
 0 -((g*m13)/(c1*l1))-(k1*(n0-n1)^2)/l1^2+(g*m13*(n0-n1)^2)/(2*c1*l1^3)-(k1*(-n0+n1)^2)
/l1^2+(g*m13*(-n0+n1)^2)/(2*c1*l1^3) (c1*k1*(n0-n1))/l1-(g*m13*(n0-n1))/(2*l1^2)+(c1*k1*(-
n0+n1))/l1-(g*m13*(-n0+n1))/(2*l1^2) 0 0 -((dn*g*m13)/(c1*l1))+(k1*(-2*c1*l1*n0-2*dn*(n0-
n1))*(n0-n1))/(2*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(n0-n1))/(4*c1*l1^3)+(k1*(-n0+n1)
*(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1^2)-(g*m13*(-n0+n1)*(2*c1*l1*n0-2*dn*(-n0+n1)))/
(4*c1*l1^3) 0 (g*m13)/(c1*l1)+(g*m23)/(c2*l2)+(k1*(n0-n1)^2)/l1^2-(g*m13*(n0-n1)^2)/
(2*c1*l1^3)+(k1*(-n0+n1)^2)/l1^2-(g*m13*(-n0+n1)^2)/(2*c1*l1^3)+(k2*(n2-n3)^2)/l2^2-
(g*m23*(n2-n3)^2)/(2*c2*l2^3)+(k2*(-n2+n3)^2)/l2^2-(g*m23*(-n2+n3)^2)/(2*c2*l2^3) -
((c1*k1*(n0-n1))/l1)+(g*m13*(n0-n1))/(2*l1^2)-(c1*k1*(-n0+n1))/l1+(g*m13*(-n0+n1))/
(2*l1^2)-(c2*k2*(n2-n3))/l2+(g*m23*(n2-n3))/(2*l2^2)-(c2*k2*(-n2+n3))/l2+(g*m23*(-n2+n3))/
(2*l2^2) 0 0 -((d0*g*m13)/(c1*l1))+(d1*g*m23)/(c2*l2)+(k1*(n0-n1)*(-2*d0*(n0-n1)
+2*c1*l1*n1))/(2*l1^2)-(g*m13*(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*c1*l1^3)+(k1*(-n0+n1)
*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1^2)-(g*m13*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/
(4*c1*l1^3)-(k2*(-2*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/(2*l2^2)+(g*m23*(-2*c2*l2*n2-2*d1*(n2-
n3))*(n2-n3))/(4*c2*l2^3)-(k2*(-n2+n3)*(2*c2*l2*n2-2*d1*(-n2+n3)))/(2*l2^2)+(g*m23*(-
n2+n3)*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*c2*l2^3) 0 -((g*m23)/(c2*l2))-(k2*(n2-n3)^2)/l2^2+
(g*m23*(n2-n3)^2)/(2*c2*l2^3)-(k2*(-n2+n3)^2)/l2^2+(g*m23*(-n2+n3)^2)/(2*c2*l2^3) (c2*k2*
(n2-n3))/l2-(g*m23*(n2-n3))/(2*l2^2)+(c2*k2*(-n2+n3))/l2-(g*m23*(-n2+n3))/(2*l2^2) 0 0
(d2*g*m23)/(c2*l2)-(k2*(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2^2)+(g*m23*(n2-n3)*(-2*d2*
(n2-n3)+2*c2*l2*n3))/(4*c2*l2^3)-(k2*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2^2)+
(g*m23*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*c2*l2^3) 0 0 0 0 0 0
 0 (c1*k1*(n0-n1))/l1-(g*m13*(n0-n1))/(2*l1^2)+(c1*k1*(-n0+n1))/l1-(g*m13*(-n0+n1))/
(2*l1^2) -2*c1^2*k1-(g*m13)/(c1*l1)+(c1*g*m13)/l1 0 0 -(c1*k1*(-2*c1*l1*n0-2*dn*(n0-n1)))/
(2*l1)+(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1)))/(4*l1^2)-(c1*k1*(2*c1*l1*n0-2*dn*(-n0+n1)))/
(2*l1)+(g*m13*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2) 0 -((c1*k1*(n0-n1))/l1)+(g*m13*(n0-n1))
/(2*l1^2)-(c1*k1*(-n0+n1))/l1+(g*m13*(-n0+n1))/(2*l1^2)-(c2*k2*(n2-n3))/l2+(g*m23*(n2-n3))
/(2*l2^2)-(c2*k2*(-n2+n3))/l2+(g*m23*(-n2+n3))/(2*l2^2) 2*c1^2*k1+2*c2^2*k2+(g*m13)/
(c1*l1)-(c1*g*m13)/l1+(g*m23)/(c2*l2)-(c2*g*m23)/l2 0 0 -(c1*k1*(-2*d0*(n0-n1)
+2*c1*l1*n1))/(2*l1)+(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)-(c1*k1*(-2*c1*l1*n1-2*d0*
(-n0+n1)))/(2*l1)+(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2)+(c2*k2*(-2*c2*l2*n2-2*d1*
(n2-n3)))/(2*l2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3)))/(4*l2^2)+(c2*k2*(2*c2*l2*n2-2*d1*(-
n2+n3)))/(2*l2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*l2^2) 0 (c2*k2*(n2-n3))/l2-(g*m23*
(n2-n3))/(2*l2^2)+(c2*k2*(-n2+n3))/l2-(g*m23*(-n2+n3))/(2*l2^2) -2*c2^2*k2-(g*m23)/(c2*l2)
+(c2*g*m23)/l2 0 0 (c2*k2*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2)-(g*m23*(-2*d2*(n2-n3)
+2*c2*l2*n3))/(4*l2^2)+(c2*k2*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2)-(g*m23*(-2*c2*l2*n3-
2*d2*(-n2+n3)))/(4*l2^2) 0 0 0 0 0 0
 0 0 0 -((k1*(n0-n1)^2*su^2)/l1^2)+(g*m13*(n0-n1)^2*su^2)/(2*c1*l1^3)-(k1*(-n0+n1)^2*su^2)
/l1^2+(g*m13*(-n0+n1)^2*su^2)/(2*c1*l1^3)+(g*m13*(-2*n0*n1-2*su^2))/(2*c1*l1) -((c1*k1*
(n0-n1)*su^2)/l1)+(g*m13*(n0-n1)*su^2)/(2*l1^2)-(c1*k1*(-n0+n1)*su^2)/l1+(g*m13*(-n0+n1)
*su^2)/(2*l1^2) 0 0 0 0 (k2*(n2-n3)^2*si^2)/l2^2-(g*m23*(n2-n3)^2*si^2)/(2*c2*l2^3)+(k2*(-
n2+n3)^2*si^2)/l2^2-(g*m23*(-n2+n3)^2*si^2)/(2*c2*l2^3)+(g*m23*(2*n2^2-2*n2*(n2-n3)
+2*si^2))/(4*c2*l2)+(g*m23*(2*n2^2+2*n2*(-n2+n3)+2*si^2))/(4*c2*l2)+(k1*(n0-n1)^2*su^2)
/l1^2-(g*m13*(n0-n1)^2*su^2)/(2*c1*l1^3)+(k1*(-n0+n1)^2*su^2)/l1^2-(g*m13*(-n0+n1)^2*su^2)
/(2*c1*l1^3)+(g*m13*(2*(n0-n1)*n1+2*n1^2+2*su^2))/(4*c1*l1)+(g*m13*(2*n1^2-2*n1*(-n0+n1)
+2*su^2))/(4*c1*l1) (g*m13*(2*d0*(n0-n1)+2*d0*n1))/(4*c1*l1)+(g*m13*(-2*d0*n1+2*d0*(-
n0+n1)))/(4*c1*l1)+(g*m23*(-2*d1*n2+2*d1*(n2-n3)))/(4*c2*l2)+(g*m23*(2*d1*n2+2*d1*(-
n2+n3)))/(4*c2*l2)+(c2*k2*(n2-n3)*si^2)/l2-(g*m23*(n2-n3)*si^2)/(2*l2^2)+(c2*k2*(-n2+n3)
```

```
*si^2)/l2-(g*m23*(-n2+n3)*si^2)/(2*l2^2)+(c1*k1*(n0-n1)*su^2)/l1-(g*m13*(n0-n1)*su^2)/
(2*l1^2)+(c1*k1*(-n0+n1)*su^2)/l1-(g*m13*(-n0+n1)*su^2)/(2*l1^2) 0 0 0 0 -((k2*(n2-n3)
^2*si^2)/l2^2)+(g*m23*(n2-n3)^2*si^2)/(2*c2*l2^3)-(k2*(-n2+n3)^2*si^2)/l2^2+(g*m23*(-
n2+n3)^2*si^2)/(2*c2*l2^3)+(g*m23*(-2*n2*n3-2*si^2))/(2*c2*l2) -((c2*k2*(n2-n3)*si^2)/l2)+
(g*m23*(n2-n3)*si^2)/(2*l2^2)-(c2*k2*(-n2+n3)*si^2)/l2+(g*m23*(-n2+n3)*si^2)/(2*l2^2) 0 0
0 0 0 0 0
 -((d0*g*m13)/(c1*l1)) 0 0 -((c1*k1*(n0-n1)*su^2)/l1)+(g*m13*(n0-n1)*su^2)/(2*l1^2)-
(c1*k1*(-n0+n1)*su^2)/l1+(g*m13*(-n0+n1)*su^2)/(2*l1^2) -2*c1^2*k1*su^2+(c1*g*m13*su^2)
/l1+(g*m13*(2*d0*dn-2*su^2))/(2*c1*l1) 0 (d0*g*m13)/(c1*l1)-(d1*g*m23)/(c2*l2) 0 0 (g*m13*
(2*d0*(n0-n1)+2*d0*n1))/(4*c1*l1)+(g*m13*(-2*d0*n1+2*d0*(-n0+n1)))/(4*c1*l1)+(g*m23*(-2
*d1*n2+2*d1*(n2-n3)))/(4*c2*l2)+(g*m23*(2*d1*n2+2*d1*(-n2+n3)))/(4*c2*l2)+(c2*k2*(n2-n3)
*si^2)/l2-(g*m23*(n2-n3)*si^2)/(2*l2^2)+(c2*k2*(-n2+n3)*si^2)/l2-(g*m23*(-n2+n3)*si^2)/
(2*l2^2)+(c1*k1*(n0-n1)*su^2)/l1-(g*m13*(n0-n1)*su^2)/(2*l1^2)+(c1*k1*(-n0+n1)*su^2)/l1-
(g*m13*(-n0+n1)*su^2)/(2*l1^2) 2*c2^2*k2*si^2-(c2*g*m23*si^2)/l2+(g*m23*
(2*d1^2+2*c2*d1*l2+2*si^2))/(2*c2*l2)+2*c1^2*k1*su^2-(c1*g*m13*su^2)/l1+(g*m13*
(2*d0^2+2*c1*d0*l1+2*su^2))/(2*c1*l1) 0 (d1*g*m23)/(c2*l2) 0 0 -((c2*k2*(n2-n3)*si^2)/l2)+
(g*m23*(n2-n3)*si^2)/(2*l2^2)-(c2*k2*(-n2+n3)*si^2)/l2+(g*m23*(-n2+n3)*si^2)/(2*l2^2) -2
*c2^2*k2*si^2+(c2*g*m23*si^2)/l2+(g*m23*(2*d1*d2-2*si^2))/(2*c2*l2) 0 0 0 0 0 0 0
 0 (d0*g*m13)/(c1*l1)-(k1*(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/(2*l1^2)+(g*m13*(n0-n1)*(-2
*d0*(n0-n1)+2*c1*l1*n1))/(4*c1*l1^3)-(k1*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1^2)+
(g*m13*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*c1*l1^3) (c1*k1*(-2*d0*(n0-n1)
+2*c1*l1*n1))/(2*l1)-(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)+(c1*k1*(-2*c1*l1*n1-2*d0*
(-n0+n1)))/(2*l1)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2) 0 0 (k1*(-2*c1*l1*n0-2*dn*
(n0-n1))*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)-(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1))*(-2*d0*
(n0-n1)+2*c1*l1*n1))/(8*c1*l1^3)+(g*m13*(2*d0*dn-2*n0*n1))/(2*c1*l1)+(k1*(-2*c1*l1*n1-
2*d0*(-n0+n1))*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1))*
(2*c1*l1*n0-2*dn*(-n0+n1)))/(8*c1*l1^3) 0 -((d0*g*m13)/(c1*l1))+(d1*g*m23)/(c2*l2)+(k1*
(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/(2*l1^2)-(g*m13*(n0-n1)*(-2*d0*(n0-n1)+2*c1*l1*n1))/
(4*c1*l1^3)+(k1*(-n0+n1)*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1^2)-(g*m13*(-n0+n1)*(-2
*c1*l1*n1-2*d0*(-n0+n1)))/(4*c1*l1^3)-(k2*(-2*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/(2*l2^2)+
(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/(4*c2*l2^3)-(k2*(-n2+n3)*(2*c2*l2*n2-2*d1*(-
n2+n3)))/(2*l2^2)+(g*m23*(-n2+n3)*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*c2*l2^3) -(c1*k1*(-2*d0*
(n0-n1)+2*c1*l1*n1))/(2*l1)+(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2)-(c1*k1*(-2
*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1)+(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2)+(c2*k2*(-2
*c2*l2*n2-2*d1*(n2-n3)))/(2*l2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3)))/(4*l2^2)+(c2*k2*
(2*c2*l2*n2-2*d1*(-n2+n3)))/(2*l2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*l2^2) 0 0 (k1*(-2
*d0*(n0-n1)+2*c1*l1*n1)^2)/(4*l1^2)-(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1)^2)/(8*c1*l1^3)+
(g*m13*(2*d0^2+2*c1*d0*l1+2*(n0-n1)*n1+2*n1^2))/(4*c1*l1)+(k1*(-2*c1*l1*n1-2*d0*(-n0+n1))
^2)/(4*l1^2)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1))^2)/(8*c1*l1^3)+(g*m13*
(2*d0^2+2*c1*d0*l1+2*n1^2-2*n1*(-n0+n1)))/(4*c1*l1)+(k2*(-2*c2*l2*n2-2*d1*(n2-n3))^2)/
(4*l2^2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))^2)/(8*c2*l2^3)+(g*m23*
(2*d1^2+2*c2*d1*l2+2*n2^2-2*n2*(n2-n3)))/(4*c2*l2)+(k2*(2*c2*l2*n2-2*d1*(-n2+n3))^2)/
(4*l2^2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3))^2)/(8*c2*l2^3)+(g*m23*
(2*d1^2+2*c2*d1*l2+2*n2^2+2*n2*(-n2+n3)))/(4*c2*l2) 0 -((d1*g*m23)/(c2*l2))+(k2*(-2
*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/(2*l2^2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))*(n2-n3))/
(4*c2*l2^3)+(k2*(-n2+n3)*(2*c2*l2*n2-2*d1*(-n2+n3)))/(2*l2^2)-(g*m23*(-n2+n3)*(2*c2*l2*n2-
2*d1*(-n2+n3)))/(4*c2*l2^3) -(c2*k2*(-2*c2*l2*n2-2*d1*(n2-n3)))/(2*l2)+(g*m23*(-2
*c2*l2*n2-2*d1*(n2-n3)))/(4*l2^2)-(c2*k2*(2*c2*l2*n2-2*d1*(-n2+n3)))/(2*l2)+(g*m23*
(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*l2^2) 0 0 (k2*(-2*c2*l2*n2-2*d1*(n2-n3))*(-2*d2*(n2-n3)
+2*c2*l2*n3))/(4*l2^2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))*(-2*d2*(n2-n3)+2*c2*l2*n3))/
(8*c2*l2^3)+(g*m23*(2*d1*d2-2*n2*n3))/(2*c2*l2)+(k2*(2*c2*l2*n2-2*d1*(-n2+n3))*(-2
*c2*l2*n3-2*d2*(-n2+n3)))/(4*l2^2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3))*(-2*c2*l2*n3-2*d2*(-
```

```
n2+n3)))/(8*c2*12^3) 0 0 0 0 0 0
 0 0 0 0 0 0 -((g*m23)/(c2*12)) 0 0 0 (d1*g*m23)/(c2*12) 0 (g*m23)/(c2*12)+(g*m3)/(c3*13) ↙
0 0 0 (d2*g*m23)/(c2*12)-(d3*g*m3)/(c3*13) 0 -((g*m3)/(c3*13)) 0 0 0 -((d4*g*m3)/(c3*13)) ↙
0
 0 0 0 0 0 0 0 -((g*m23)/(c2*12))-(k2*(n2-n3)^2)/12^2+(g*m23*(n2-n3)^2)/(2*c2*12^3)-(k2*(- ↙
n2+n3)^2)/12^2+(g*m23*(-n2+n3)^2)/(2*c2*12^3) (c2*k2*(n2-n3))/12-(g*m23*(n2-n3))/(2*12^2)+ ↙
(c2*k2*(-n2+n3))/12-(g*m23*(-n2+n3))/(2*12^2) 0 0 -((d1*g*m23)/(c2*12))+(k2*(-2*c2*12*n2- ↙
2*d1*(n2-n3))*(n2-n3))/(2*12^2)-(g*m23*(-2*c2*12*n2-2*d1*(n2-n3))*(n2-n3))/(4*c2*12^3)+ ↙
(k2*(-n2+n3)*(2*c2*12*n2-2*d1*(-n2+n3)))/(2*12^2)-(g*m23*(-n2+n3)*(2*c2*12*n2-2*d1*(- ↙
n2+n3)))/(4*c2*12^3) 0 (g*m23)/(c2*12)+(g*m3)/(c3*13)+(k2*(n2-n3)^2)/12^2-(g*m23*(n2-n3) ↙
^2)/(2*c2*12^3)+(k2*(-n2+n3)^2)/12^2-(g*m23*(-n2+n3)^2)/(2*c2*12^3)+(k3*(n4-n5)^2)/13^2- ↙
(g*m3*(n4-n5)^2)/(2*c3*13^3)+(k3*(-n4+n5)^2)/13^2-(g*m3*(-n4+n5)^2)/(2*c3*13^3) -((c2*k2* ↙
(n2-n3))/12)+(g*m23*(n2-n3))/(2*12^2)-(c2*k2*(-n2+n3))/12+(g*m23*(-n2+n3))/(2*12^2)- ↙
(c3*k3*(n4-n5))/13+(g*m3*(n4-n5))/(2*13^2)-(c3*k3*(-n4+n5))/13+(g*m3*(-n4+n5))/(2*13^2) 0 ↙
0 -((d2*g*m23)/(c2*12))+(d3*g*m3)/(c3*13)+(k2*(n2-n3)*(-2*d2*(n2-n3)+2*c2*12*n3))/(2*12^2) ↙
-(g*m23*(n2-n3)*(-2*d2*(n2-n3)+2*c2*12*n3))/(4*c2*12^3)+(k2*(-n2+n3)*(-2*c2*12*n3-2*d2*(- ↙
n2+n3)))/(2*12^2)-(g*m23*(-n2+n3)*(-2*c2*12*n3-2*d2*(-n2+n3)))/(4*c2*12^3)-(k3*(-2 ↙
*c3*13*n4-2*d3*(n4-n5))*(n4-n5))/(2*13^2)+(g*m3*(-2*c3*13*n4-2*d3*(n4-n5))*(n4-n5))/ ↙
(4*c3*13^3)-(k3*(-n4+n5)*(2*c3*13*n4-2*d3*(-n4+n5)))/(2*13^2)+(g*m3*(-n4+n5)*(2*c3*13*n4- ↙
2*d3*(-n4+n5)))/(4*c3*13^3) 0 -((g*m3)/(c3*13))-(k3*(n4-n5)^2)/13^2+(g*m3*(n4-n5)^2)/ ↙
(2*c3*13^3)-(k3*(-n4+n5)^2)/13^2+(g*m3*(-n4+n5)^2)/(2*c3*13^3) (c3*k3*(n4-n5))/13-(g*m3* ↙
(n4-n5))/(2*13^2)+(c3*k3*(-n4+n5))/13-(g*m3*(-n4+n5))/(2*13^2) 0 0 (d4*g*m3)/(c3*13)-(k3* ↙
(n4-n5)*(-2*d4*(n4-n5)+2*c3*13*n5))/(2*13^2)+(g*m3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*13*n5))/ ↙
(4*c3*13^3)-(k3*(-n4+n5)*(-2*c3*13*n5-2*d4*(-n4+n5)))/(2*13^2)+(g*m3*(-n4+n5)*(-2 ↙
*c3*13*n5-2*d4*(-n4+n5)))/(4*c3*13^3)
 0 0 0 0 0 0 0 (c2*k2*(n2-n3))/12-(g*m23*(n2-n3))/(2*12^2)+(c2*k2*(-n2+n3))/12-(g*m23*(- ↙
n2+n3))/(2*12^2) -2*c2^2*k2-(g*m23)/(c2*12)+(c2*g*m23)/12 0 0 -(c2*k2*(-2*c2*12*n2-2*d1* ↙
(n2-n3)))/(2*12)+(g*m23*(-2*c2*12*n2-2*d1*(n2-n3)))/(4*12^2)-(c2*k2*(2*c2*12*n2-2*d1*(- ↙
n2+n3)))/(2*12)+(g*m23*(2*c2*12*n2-2*d1*(-n2+n3)))/(4*12^2) 0 -((c2*k2*(n2-n3))/12)+ ↙
(g*m23*(n2-n3))/(2*12^2)-(c2*k2*(-n2+n3))/12+(g*m23*(-n2+n3))/(2*12^2)-(c3*k3*(n4-n5))/13+ ↙
(g*m3*(n4-n5))/(2*13^2)-(c3*k3*(-n4+n5))/13+(g*m3*(-n4+n5))/(2*13^2) 2*c2^2*k2+2*c3^2*k3+ ↙
(g*m23)/(c2*12)-(c2*g*m23)/12+(g*m3)/(c3*13)-(c3*g*m3)/13 0 0 -(c2*k2*(-2*d2*(n2-n3) ↙
+2*c2*12*n3))/(2*12)+(g*m23*(-2*d2*(n2-n3)+2*c2*12*n3))/(4*12^2)-(c2*k2*(-2*c2*12*n3-2*d2* ↙
(-n2+n3)))/(2*12)+(g*m23*(-2*c2*12*n3-2*d2*(-n2+n3)))/(4*12^2)+(c3*k3*(-2*c3*13*n4-2*d3* ↙
(n4-n5)))/(2*13)-(g*m3*(-2*c3*13*n4-2*d3*(n4-n5)))/(4*13^2)+(c3*k3*(2*c3*13*n4-2*d3*(- ↙
n4+n5)))/(2*13)-(g*m3*(2*c3*13*n4-2*d3*(-n4+n5)))/(4*13^2) 0 (c3*k3*(n4-n5))/13-(g*m3*(n4- ↙
n5))/(2*13^2)+(c3*k3*(-n4+n5))/13-(g*m3*(-n4+n5))/(2*13^2) -2*c3^2*k3-(g*m3)/(c3*13)+ ↙
(c3*g*m3)/13 0 0 (c3*k3*(-2*d4*(n4-n5)+2*c3*13*n5))/(2*13)-(g*m3*(-2*d4*(n4-n5) ↙
+2*c3*13*n5))/(4*13^2)+(c3*k3*(-2*c3*13*n5-2*d4*(-n4+n5)))/(2*13)-(g*m3*(-2*c3*13*n5-2*d4* ↙
(-n4+n5)))/(4*13^2)
 0 0 0 0 0 0 0 0 0 -((k2*(n2-n3)^2*si^2)/12^2)+(g*m23*(n2-n3)^2*si^2)/(2*c2*12^3)-(k2*(- ↙
n2+n3)^2*si^2)/12^2+(g*m23*(-n2+n3)^2*si^2)/(2*c2*12^3)+(g*m23*(-2*n2*n3-2*si^2))/ ↙
(2*c2*12) -((c2*k2*(n2-n3)*si^2)/12)+(g*m23*(n2-n3)*si^2)/(2*12^2)-(c2*k2*(-n2+n3)*si^2) ↙
/12+(g*m23*(-n2+n3)*si^2)/(2*12^2) 0 0 0 0 (k2*(n2-n3)^2*si^2)/12^2-(g*m23*(n2-n3)^2*si^2) ↙
/(2*c2*12^3)+(k2*(-n2+n3)^2*si^2)/12^2-(g*m23*(-n2+n3)^2*si^2)/(2*c2*12^3)+(g*m23*(2*(n2- ↙
n3)*n3+2*n3^2+2*si^2))/(4*c2*12)+(g*m23*(2*n3^2-2*n3*(-n2+n3)+2*si^2))/(4*c2*12)+(k3*(n4- ↙
n5)^2*s1^2)/13^2-(g*m3*(n4-n5)^2*s1^2)/(2*c3*13^3)+(k3*(-n4+n5)^2*s1^2)/13^2-(g*m3*(- ↙
n4+n5)^2*s1^2)/(2*c3*13^3)+(g*m3*(2*n4^2-2*n4*(n4-n5)+2*s1^2))/(4*c3*13)+(g*m3* ↙
(2*n4^2+2*n4*(-n4+n5)+2*s1^2))/(4*c3*13) (g*m23*(2*d2*(n2-n3)+2*d2*n3))/(4*c2*12)+(g*m23* ↙
(-2*d2*n3+2*d2*(-n2+n3)))/(4*c2*12)+(g*m3*(-2*d3*n4+2*d3*(n4-n5)))/(4*c3*13)+(g*m3* ↙
(2*d3*n4+2*d3*(-n4+n5)))/(4*c3*13)+(c2*k2*(n2-n3)*si^2)/12-(g*m23*(n2-n3)*si^2)/(2*12^2)+ ↙
(c2*k2*(-n2+n3)*si^2)/12-(g*m23*(-n2+n3)*si^2)/(2*12^2)+(c3*k3*(n4-n5)*s1^2)/13-(g*m3*(n4- ↙
```

```
n5)*sl^2)/(2*l3^2)+(c3*k3*(-n4+n5)*sl^2)/l3-(g*m3*(-n4+n5)*sl^2)/(2*l3^2) 0 0 0 0 -((k3*↙
(n4-n5)^2*sl^2)/l3^2)+(g*m3*(n4-n5)^2*sl^2)/(2*c3*l3^3)-(k3*(-n4+n5)^2*sl^2)/l3^2+(g*m3*(-↙
n4+n5)^2*sl^2)/(2*c3*l3^3)+(g*m3*(-2*n4*n5-2*sl^2))/(2*c3*l3) -((c3*k3*(n4-n5)*sl^2)/l3)+↙
(g*m3*(n4-n5)*sl^2)/(2*l3^2)-(c3*k3*(-n4+n5)*sl^2)/l3+(g*m3*(-n4+n5)*sl^2)/(2*l3^2) 0
 0 0 0 0 0 0 -((d2*g*m23)/(c2*l2)) 0 0 -((c2*k2*(n2-n3)*si^2)/l2)+(g*m23*(n2-n3)*si^2)/↙
(2*l2^2)-(c2*k2*(-n2+n3)*si^2)/l2+(g*m23*(-n2+n3)*si^2)/(2*l2^2) -2*c2^2*k2*si^2+↙
(c2*g*m23*si^2)/l2+(g*m23*(2*d1*d2-2*si^2))/(2*c2*l2) 0 (d2*g*m23)/(c2*l2)-(d3*g*m3)/↙
(c3*l3) 0 0 (g*m23*(2*d2*(n2-n3)+2*d2*n3))/(4*c2*l2)+(g*m23*(-2*d2*n3+2*d2*(-n2+n3)))/↙
(4*c2*l2)+(g*m3*(-2*d3*n4+2*d3*(n4-n5)))/(4*c3*l3)+(g*m3*(2*d3*n4+2*d3*(-n4+n5)))/↙
(4*c3*l3)+(c2*k2*(n2-n3)*si^2)/l2-(g*m23*(n2-n3)*si^2)/(2*l2^2)+(c2*k2*(-n2+n3)*si^2)/l2-↙
(g*m23*(-n2+n3)*si^2)/(2*l2^2)+(c3*k3*(n4-n5)*sl^2)/l3-(g*m3*(n4-n5)*sl^2)/(2*l3^2)+↙
(c3*k3*(-n4+n5)*sl^2)/l3-(g*m3*(-n4+n5)*sl^2)/(2*l3^2) 2*c2^2*k2*si^2-(c2*g*m23*si^2)/l2+↙
(g*m23*(2*d2^2+2*c2*d2*l2+2*si^2))/(2*c2*l2)+2*c3^2*k3*sl^2-(c3*g*m3*sl^2)/l3+(g*m3*↙
(2*d3^2+2*c3*d3*l3+2*sl^2))/(2*c3*l3) 0 (d3*g*m3)/(c3*l3) 0 0 -((c3*k3*(n4-n5)*sl^2)/l3)+↙
(g*m3*(n4-n5)*sl^2)/(2*l3^2)-(c3*k3*(-n4+n5)*sl^2)/l3+(g*m3*(-n4+n5)*sl^2)/(2*l3^2) -2↙
*c3^2*k3*sl^2+(c3*g*m3*sl^2)/l3+(g*m3*(2*d3*d4-2*sl^2))/(2*c3*l3) 0
 0 0 0 0 0 0 0 (d2*g*m23)/(c2*l2)-(k2*(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2^2)+(g*m23*↙
(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*c2*l2^3)-(k2*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/↙
(2*l2^2)+(g*m23*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*c2*l2^3) (c2*k2*(-2*d2*(n2-n3)↙
+2*c2*l2*n3))/(2*l2)-(g*m23*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*l2^2)+(c2*k2*(-2*c2*l2*n3-2*d2*↙
(-n2+n3)))/(2*l2)-(g*m23*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*l2^2) 0 0 (k2*(-2*c2*l2*n2-2*d1*↙
(n2-n3))*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*l2^2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3))*(-2*d2*↙
(n2-n3)+2*c2*l2*n3))/(8*c2*l2^3)+(g*m23*(2*d1*d2-2*n2*n3))/(2*c2*l2)+(k2*(2*c2*l2*n2-2*d1*↙
(-n2+n3))*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*l2^2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3))*(-2↙
*c2*l2*n3-2*d2*(-n2+n3)))/(8*c2*l2^3) 0 -((d2*g*m23)/(c2*l2))+(d3*g*m3)/(c3*l3)+(k2*(n2-↙
n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2^2)-(g*m23*(n2-n3)*(-2*d2*(n2-n3)+2*c2*l2*n3))/↙
(4*c2*l2^3)+(k2*(-n2+n3)*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2^2)-(g*m23*(-n2+n3)*(-2↙
*c2*l2*n3-2*d2*(-n2+n3)))/(4*c2*l2^3)-(k3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/(2*l3^2)+↙
(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/(4*c3*l3^3)-(k3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-↙
n4+n5)))/(2*l3^2)+(g*m3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*c3*l3^3) -(c2*k2*(-2*d2*↙
(n2-n3)+2*c2*l2*n3))/(2*l2)+(g*m23*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*l2^2)-(c2*k2*(-2↙
*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2)+(g*m23*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(4*l2^2)+(c3*k3*(-2↙
*c3*l3*n4-2*d3*(n4-n5)))/(2*l3)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5)))/(4*l3^2)+(c3*k3*↙
(2*c3*l3*n4-2*d3*(-n4+n5)))/(2*l3)-(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*l3^2) 0 0 (k2*(-2↙
*d2*(n2-n3)+2*c2*l2*n3)^2)/(4*l2^2)-(g*m23*(-2*d2*(n2-n3)+2*c2*l2*n3)^2)/(8*c2*l2^3)+↙
(g*m23*(2*d2^2+2*c2*d2*l2+2*(n2-n3)*n3+2*n3^2))/(4*c2*l2)+(k2*(-2*c2*l2*n3-2*d2*(-n2+n3))↙
^2)/(4*l2^2)-(g*m23*(-2*c2*l2*n3-2*d2*(-n2+n3))^2)/(8*c2*l2^3)+(g*m23*↙
(2*d2^2+2*c2*d2*l2+2*n3^2-2*n3*(-n2+n3)))/(4*c2*l2)+(k3*(-2*c3*l3*n4-2*d3*(n4-n5))^2)/↙
(4*l3^2)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))^2)/(8*c3*l3^3)+(g*m3*(2*d3^2+2*c3*d3*l3+2*n4^2-↙
2*n4*(n4-n5)))/(4*c3*l3)+(k3*(2*c3*l3*n4-2*d3*(-n4+n5))^2)/(4*l3^2)-(g*m3*(2*c3*l3*n4-↙
2*d3*(-n4+n5))^2)/(8*c3*l3^3)+(g*m3*(2*d3^2+2*c3*d3*l3+2*n4^2+2*n4*(-n4+n5)))/(4*c3*l3) 0↙
-((d3*g*m3)/(c3*l3))+(k3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/(2*l3^2)-(g*m3*(-2*c3*l3*n4-↙
2*d3*(n4-n5))*(n4-n5))/(4*c3*l3^3)+(k3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-n4+n5)))/(2*l3^2)-↙
(g*m3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*c3*l3^3) -(c3*k3*(-2*c3*l3*n4-2*d3*(n4-n5)))↙
/(2*l3)+(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5)))/(4*l3^2)-(c3*k3*(2*c3*l3*n4-2*d3*(-n4+n5)))/↙
(2*l3)+(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*l3^2) 0 0 (k3*(-2*c3*l3*n4-2*d3*(n4-n5))*(-2↙
*d4*(n4-n5)+2*c3*l3*n5))/(4*l3^2)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))*(-2*d4*(n4-n5)↙
+2*c3*l3*n5))/(8*c3*l3^3)+(g*m3*(2*d3*d4-2*n4*n5))/(2*c3*l3)+(k3*(2*c3*l3*n4-2*d3*(-↙
n4+n5))*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*l3^2)-(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5))*(-2↙
*c3*l3*n5-2*d4*(-n4+n5)))/(8*c3*l3^3)
 0 0 0 0 0 0 0 0 0 0 0 0 -((g*m3)/(c3*l3)) 0 0 0 (d3*g*m3)/(c3*l3) 0 (g*m3)/(c3*l3) 0 0 0↙
(d4*g*m3)/(c3*l3) 0
```

```
 0 0 0 0 0 0 0 0 0 0 0 0 0 -((g*m3)/(c3*l3))-(k3*(n4-n5)^2)/l3^2+(g*m3*(n4-n5)^2)/
(2*c3*l3^3)-(k3*(-n4+n5)^2)/l3^2+(g*m3*(-n4+n5)^2)/(2*c3*l3^3) (c3*k3*(n4-n5))/l3-(g*m3*
(n4-n5))/(2*l3^2)+(c3*k3*(-n4+n5))/l3-(g*m3*(-n4+n5))/(2*l3^2) 0 0 -((d3*g*m3)/(c3*l3))+
(k3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))/(2*l3^2)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5))*(n4-n5))
/(4*c3*l3^3)+(k3*(-n4+n5)*(2*c3*l3*n4-2*d3*(-n4+n5)))/(2*l3^2)-(g*m3*(-n4+n5)*(2*c3*l3*n4-
2*d3*(-n4+n5)))/(4*c3*l3^3) 0 (g*m3)/(c3*l3)+(k3*(n4-n5)^2)/l3^2-(g*m3*(n4-n5)^2)/
(2*c3*l3^3)+(k3*(-n4+n5)^2)/l3^2-(g*m3*(-n4+n5)^2)/(2*c3*l3^3) -((c3*k3*(n4-n5))/l3)+
(g*m3*(n4-n5))/(2*l3^2)-(c3*k3*(-n4+n5))/l3+(g*m3*(-n4+n5))/(2*l3^2) 0 0 -((d4*g*m3)/
(c3*l3))+(k3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3^2)-(g*m3*(n4-n5)*(-2*d4*(n4-n5)
+2*c3*l3*n5))/(4*c3*l3^3)+(k3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3^2)-(g*m3*(-
n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*c3*l3^3)
 0 0 0 0 0 0 0 0 0 0 0 0 0 (c3*k3*(n4-n5))/l3-(g*m3*(n4-n5))/(2*l3^2)+(c3*k3*(-n4+n5))/l3-
(g*m3*(-n4+n5))/(2*l3^2) -2*c3^2*k3-(g*m3)/(c3*l3)+(c3*g*m3)/l3 0 0 -(c3*k3*(-2*c3*l3*n4-
2*d3*(n4-n5)))/(2*l3)+(g*m3*(-2*c3*l3*n4-2*d3*(n4-n5)))/(4*l3^2)-(c3*k3*(2*c3*l3*n4-2*d3*
(-n4+n5)))/(2*l3)+(g*m3*(2*c3*l3*n4-2*d3*(-n4+n5)))/(4*l3^2) 0 -((c3*k3*(n4-n5))/l3)+
(g*m3*(n4-n5))/(2*l3^2)-(c3*k3*(-n4+n5))/l3+(g*m3*(-n4+n5))/(2*l3^2) 2*c3^2*k3+(g*m3)/
(c3*l3)-(c3*g*m3)/l3 0 0 -(c3*k3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3)+(g*m3*(-2*d4*(n4-n5)
+2*c3*l3*n5))/(4*l3^2)-(c3*k3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3)+(g*m3*(-2*c3*l3*n5-2*d4*
(-n4+n5)))/(4*l3^2)
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -((k3*(n4-n5)^2*s1^2)/l3^2)+(g*m3*(n4-n5)^2*s1^2)/
(2*c3*l3^3)-(k3*(-n4+n5)^2*s1^2)/l3^2+(g*m3*(-n4+n5)^2*s1^2)/(2*c3*l3^3)+(g*m3*(-2*n4*n5-
2*s1^2))/(2*c3*l3) -((c3*k3*(n4-n5)*s1^2)/l3)+(g*m3*(n4-n5)*s1^2)/(2*l3^2)-(c3*k3*(-n4+n5)
*s1^2)/l3+(g*m3*(-n4+n5)*s1^2)/(2*l3^2) 0 0 0 0 (k3*(n4-n5)^2*s1^2)/l3^2-(g*m3*(n4-n5)
^2*s1^2)/(2*c3*l3^3)+(k3*(-n4+n5)^2*s1^2)/l3^2-(g*m3*(-n4+n5)^2*s1^2)/(2*c3*l3^3)+(g*m3*
(2*(n4-n5)*n5+2*n5^2+2*s1^2))/(4*c3*l3)+(g*m3*(2*n5^2-2*n5*(-n4+n5)+2*s1^2))/(4*c3*l3)
(g*m3*(2*d4*(n4-n5)+2*d4*n5))/(4*c3*l3)+(g*m3*(-2*d4*n5+2*d4*(-n4+n5)))/(4*c3*l3)+(c3*k3*
(n4-n5)*s1^2)/l3-(g*m3*(n4-n5)*s1^2)/(2*l3^2)+(c3*k3*(-n4+n5)*s1^2)/l3-(g*m3*(-n4+n5)
*s1^2)/(2*l3^2) 0
 0 0 0 0 0 0 0 0 0 0 0 0 -((d4*g*m3)/(c3*l3)) 0 0 -((c3*k3*(n4-n5)*s1^2)/l3)+(g*m3*(n4-n5)
*s1^2)/(2*l3^2)-(c3*k3*(-n4+n5)*s1^2)/l3+(g*m3*(-n4+n5)*s1^2)/(2*l3^2) -2*c3^2*k3*s1^2+
(c3*g*m3*s1^2)/l3+(g*m3*(2*d3*d4-2*s1^2))/(2*c3*l3) 0 (d4*g*m3)/(c3*l3) 0 0 (g*m3*(2*d4*
(n4-n5)+2*d4*n5))/(4*c3*l3)+(g*m3*(-2*d4*n5+2*d4*(-n4+n5)))/(4*c3*l3)+(c3*k3*(n4-n5)*s1^2)
/l3-(g*m3*(n4-n5)*s1^2)/(2*l3^2)+(c3*k3*(-n4+n5)*s1^2)/l3-(g*m3*(-n4+n5)*s1^2)/(2*l3^2)
2*c3^2*k3*s1^2-(c3*g*m3*s1^2)/l3+(g*m3*(2*d4^2+2*c3*d4*l3+2*s1^2))/(2*c3*l3) 0
 0 0 0 0 0 0 0 0 0 0 0 0 (d4*g*m3)/(c3*l3)-(k3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/
(2*l3^2)+(g*m3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*c3*l3^3)-(k3*(-n4+n5)*(-2*c3*l3*n5-
2*d4*(-n4+n5)))/(2*l3^2)+(g*m3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*c3*l3^3) (c3*k3*
(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3)-(g*m3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*l3^2)+(c3*k3*(-2
*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3)-(g*m3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*l3^2) 0 0 (k3*(-2
*c3*l3*n4-2*d3*(n4-n5))*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*l3^2)-(g*m3*(-2*c3*l3*n4-2*d3*(n4-
n5))*(-2*d4*(n4-n5)+2*c3*l3*n5))/(8*c3*l3^3)+(g*m3*(2*d3*d4-2*n4*n5))/(2*c3*l3)+(k3*
(2*c3*l3*n4-2*d3*(-n4+n5))*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(4*l3^2)-(g*m3*(2*c3*l3*n4-2*d3*(-
n4+n5))*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(8*c3*l3^3) 0 -((d4*g*m3)/(c3*l3))+(k3*(n4-n5)*(-2
*d4*(n4-n5)+2*c3*l3*n5))/(2*l3^2)-(g*m3*(n4-n5)*(-2*d4*(n4-n5)+2*c3*l3*n5))/(4*c3*l3^3)+
(k3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3^2)-(g*m3*(-n4+n5)*(-2*c3*l3*n5-2*d4*(-
n4+n5)))/(4*c3*l3^3) -(c3*k3*(-2*d4*(n4-n5)+2*c3*l3*n5))/(2*l3)+(g*m3*(-2*d4*(n4-n5)
+2*c3*l3*n5))/(4*l3^2)-(c3*k3*(-2*c3*l3*n5-2*d4*(-n4+n5)))/(2*l3)+(g*m3*(-2*c3*l3*n5-2*d4*
(-n4+n5)))/(4*l3^2) 0 0 (k3*(-2*d4*(n4-n5)+2*c3*l3*n5)^2)/(4*l3^2)-(g*m3*(-2*d4*(n4-n5)
+2*c3*l3*n5)^2)/(8*c3*l3^3)+(g*m3*(2*d4^2+2*c3*d4*l3+2*(n4-n5)*n5+2*n5^2))/(4*c3*l3)+(k3*
(-2*c3*l3*n5-2*d4*(-n4+n5))^2)/(4*l3^2)-(g*m3*(-2*c3*l3*n5-2*d4*(-n4+n5))^2)/(8*c3*l3^3)+
(g*m3*(2*d4^2+2*c3*d4*l3+2*n5^2-2*n5*(-n4+n5)))/(4*c3*l3)
];
```

```
km = [...
mn 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 mn 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 mn 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 Inz Inyz Inzx 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 Inyz Iny Inxy 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 Inzx Inxy Inx 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 m1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 m1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 m1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 I1z I1yz I1zx 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 I1yz I1y I1xy 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 I1zx I1xy I1x 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 m2 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 m2 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 m2 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I2z I2yz I2zx 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I2yz I2y I2xy 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I2zx I2xy I2x 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 m3 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 m3 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 m3 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I3z I3yz I3zx
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I3yz I3y I3xy
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 I3zx I3xy I3x
];
cqxm = [...
0 (cn*kn*(nn0-nn1))/ln-(g*mn3*(nn0-nn1))/(2*ln^2) -(cn^2*kn)-(g*mn3)/(2*cn*ln)+(cn*g*mn3)/↙
(2*ln) 0 0 (g*mn3*nn1)/(2*cn*ln)+(cn*kn*(-2*dm*(nn0-nn1)+2*cn*ln*nn1))/(2*ln)-(g*mn3*(-2↙
*dm*(nn0-nn1)+2*cn*ln*nn1))/(4*ln^2) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 -(g*mn3)/(2*cn*ln) 0 0 -(g*mn3*nn1)/(2*cn*ln) -(dm*g*mn3)/(2*cn*ln) 0 0 0 0 0 0 0 0 0 0 0↙
0 0 0 0 0 0 0
 0 (cn*kn*(-nn0+nn1))/ln-(g*mn3*(-nn0+nn1))/(2*ln^2) -(cn^2*kn)-(g*mn3)/(2*cn*ln)+↙
(cn*g*mn3)/(2*ln) 0 0 -(g*mn3*nn1)/(2*cn*ln)+(cn*kn*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(2*ln)↙
-(g*mn3*(-2*cn*ln*nn1-2*dm*(-nn0+nn1)))/(4*ln^2) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 -(g*mn3)/(2*cn*ln) 0 0 (g*mn3*nn1)/(2*cn*ln) -(dm*g*mn3)/(2*cn*ln) 0 0 0 0 0 0 0 0 0 0 0↙
0 0 0 0 0 0 0
 0 -((c1*k1*(n0-n1))/l1)+(g*m13*(n0-n1))/(2*l1^2) c1^2*k1+(g*m13)/(2*c1*l1)-(c1*g*m13)/↙
(2*l1) 0 0 (g*m13*(-2*n0+2*(n0-n1)))/(4*c1*l1)+(c1*k1*(-2*c1*l1*n0-2*dn*(n0-n1)))/(2*l1)-↙
(g*m13*(-2*c1*l1*n0-2*dn*(n0-n1)))/(4*l1^2) 0 (c1*k1*(n0-n1))/l1-(g*m13*(n0-n1))/(2*l1^2)↙
-(c1^2*k1)-(g*m13)/(2*c1*l1)+(c1*g*m13)/(2*l1) 0 0 (g*m13*n1)/(2*c1*l1)+(c1*k1*(-2*d0*(n0-↙
n1)+2*c1*l1*n1))/(2*l1)-(g*m13*(-2*d0*(n0-n1)+2*c1*l1*n1))/(4*l1^2) 0 0 0 0 0 0 0 0 0 0↙
0
 (g*m13)/(2*c1*l1) 0 0 (g*m13*(2*n0-2*(n0-n1)))/(4*c1*l1) (g*(-2*dn-2*c1*l1)*m13)/↙
(4*c1*l1) 0 -(g*m13)/(2*c1*l1) 0 0 -(g*m13*n1)/(2*c1*l1) -(d0*g*m13)/(2*c1*l1) 0 0 0 0 0 0↙
0 0 0 0 0 0 0
 0 -((c1*k1*(-n0+n1))/l1)+(g*m13*(-n0+n1))/(2*l1^2) c1^2*k1+(g*m13)/(2*c1*l1)-(c1*g*m13)/↙
(2*l1) 0 0 (g*m13*(2*n0+2*(-n0+n1)))/(4*c1*l1)+(c1*k1*(2*c1*l1*n0-2*dn*(-n0+n1)))/(2*l1)-↙
(g*m13*(2*c1*l1*n0-2*dn*(-n0+n1)))/(4*l1^2) 0 (c1*k1*(-n0+n1))/l1-(g*m13*(-n0+n1))/↙
(2*l1^2) -(c1^2*k1)-(g*m13)/(2*c1*l1)+(c1*g*m13)/(2*l1) 0 0 -(g*m13*n1)/(2*c1*l1)+(c1*k1*↙
(-2*c1*l1*n1-2*d0*(-n0+n1)))/(2*l1)-(g*m13*(-2*c1*l1*n1-2*d0*(-n0+n1)))/(4*l1^2) 0 0 0 0 0↙
0 0 0 0 0 0 0
```

```
 (g*m13)/(2*c1*l1) 0 0 (g*m13*(-2*n0-2*(-n0+n1)))/(4*c1*l1) (g*(-2*dn-2*c1*l1)*m13)/↙
(4*c1*l1) 0 -(g*m13)/(2*c1*l1) 0 0 (g*m13*n1)/(2*c1*l1) -(d0*g*m13)/(2*c1*l1) 0 0 0 0 0 0↙
0 0 0 0 0 0 0
 0 0 0 0 0 0 0 -((c2*k2*(n2-n3))/l2)+(g*m23*(n2-n3))/(2*l2^2) c2^2*k2+(g*m23)/(2*c2*l2)-↙
(c2*g*m23)/(2*l2) 0 0 (g*m23*(-2*n2+2*(n2-n3)))/(4*c2*l2)+(c2*k2*(-2*c2*l2*n2-2*d1*(n2-↙
n3)))/(2*l2)-(g*m23*(-2*c2*l2*n2-2*d1*(n2-n3)))/(4*l2^2) 0 (c2*k2*(n2-n3))/l2-(g*m23*(n2-↙
n3))/(2*l2^2) -(c2^2*k2)-(g*m23)/(2*c2*l2)+(c2*g*m23)/(2*l2) 0 0 (g*m23*n3)/(2*c2*l2)+↙
(c2*k2*(-2*d2*(n2-n3)+2*c2*l2*n3))/(2*l2)-(g*m23*(-2*d2*(n2-n3)+2*c2*l2*n3))/(4*l2^2) 0 0↙
0 0 0 0
 0 0 0 0 0 0 (g*m23)/(2*c2*l2) 0 0 (g*m23*(2*n2-2*(n2-n3)))/(4*c2*l2) (g*(-2*d1-2*c2*l2)↙
*m23)/(4*c2*l2) 0 -(g*m23)/(2*c2*l2) 0 0 -(g*m23*n3)/(2*c2*l2) -(d2*g*m23)/(2*c2*l2) 0 0 0↙
0 0 0 0
 0 0 0 0 0 0 0 -((c2*k2*(-n2+n3))/l2)+(g*m23*(-n2+n3))/(2*l2^2) c2^2*k2+(g*m23)/(2*c2*l2)-↙
(c2*g*m23)/(2*l2) 0 0 (g*m23*(2*n2+2*(-n2+n3)))/(4*c2*l2)+(c2*k2*(2*c2*l2*n2-2*d1*(-↙
n2+n3)))/(2*l2)-(g*m23*(2*c2*l2*n2-2*d1*(-n2+n3)))/(4*l2^2) 0 (c2*k2*(-n2+n3))/l2-(g*m23*↙
(-n2+n3))/(2*l2^2) -(c2^2*k2)-(g*m23)/(2*c2*l2)+(c2*g*m23)/(2*l2) 0 0 -(g*m23*n3)/↙
(2*c2*l2)+(c2*k2*(-2*c2*l2*n3-2*d2*(-n2+n3)))/(2*l2)-(g*m23*(-2*c2*l2*n3-2*d2*(-n2+n3)))/↙
(4*l2^2) 0 0 0 0 0 0
 0 0 0 0 0 0 (g*m23)/(2*c2*l2) 0 0 (g*m23*(-2*n2-2*(-n2+n3)))/(4*c2*l2) (g*(-2*d1-2*c2*l2)↙
*m23)/(4*c2*l2) 0 -(g*m23)/(2*c2*l2) 0 0 (g*m23*n3)/(2*c2*l2) -(d2*g*m23)/(2*c2*l2) 0 0 0↙
0 0 0 0
];
cxsm = [...
-((g*mn3)/(cn*ln)) 0 0 0 0 0
 0 (g*mn3*(-ln^2+(nn0-nn1)^2)-2*cn*kn*ln*(nn0-nn1)^2)/(cn*ln^3) 0 0 0 -(((2*cn*kn*ln-↙
g*mn3)*nn0*(nn0-nn1))/ln^2)
 0 0 (-2*cn^3*kn*ln-g*mn3+cn^2*g*mn3)/(cn*ln) 0 0 0
 0 0 0 -((g*mn3*nn0*nn1)/(cn*ln)) 0 0
 -((dm*g*mn3)/(cn*ln)) 0 0 0 0 0
 0 (dm*(g*mn3*(ln^2-(nn0-nn1)^2)+2*cn*kn*ln*(nn0-nn1)^2)-cn*ln*(2*cn*kn*ln-g*mn3)*(nn0-↙
nn1)*nn1)/(cn*ln^3) 0 0 0 (nn0*(-2*cn^3*kn*ln^2*nn1-g*ln*mn3*nn1+cn*dm*g*mn3*(-nn0+nn1)↙
+cn^2*ln*(2*dm*kn*(nn0-nn1)+g*mn3*nn1)))/(cn*ln^2)
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
 0 0 0 0 0 0
];
```

Appendix B

Gradient Descent Model Fit

```
clear,clc
close all

%Original Values
quadopt20060509controlsmainasbuiltoffdiag; %from model definition code

fitparam0(1) = pend.Inz;
fitparam0(2) = pend.I1z;
fitparam0(3) = pend.I2z;
fitparam0(4) = pend.I3z;
fitparam0(5) = pend.kcn;
fitparam0(6) = pend.kc1;
fitparam0(7) = pend.kc2;
fitparam0(8) =pend.mn;
fitparam0(9) =pend.m1;
fitparam0(10) =pend.m2;
%fitparam0(11) =pend.m3;

original_values = fitparam0;% - iparamos;

mm = [0.48 0.427 0.98 1.583 2.0 2.688 3.437 3.407 ...
      0.458 0.803 1.039 2.097 2.752 3.21 5.151 ...
      0.6768 1.4297 2.5195 3.1455 ...
      0.5550 2.2686 3.6157];%measured modes
% mm = [0.674 1.424 2.504 3.135 0.552 2.278 3.613];%measured modes
mm = sort(mm);

f_Inz =  fitparam0(1);
f_I1z =  fitparam0(2);
f_I2z =  fitparam0(3);
f_I3z =  fitparam0(4);
f_kcn = fitparam0(5); %pend.kcn;   %30
f_kc1 = fitparam0(6); %pend.kc1;   %31
f_kc2 = fitparam0(7); %pend.kc2;   %32
f_mn = fitparam0(8); %pend.mn;     %22
f_m1 = fitparam0(9); %pend.m1;     %23
f_m2 = fitparam0(10); %pend.m2;     %24
% f_m3 = fitparam0(11); %pend.m3;     %25

z_massmomerr = 0.005;
masserr = 0.02; %mass percent change limit
kc_err = 0.1;
lb = [(1-z_massmomerr)*f_Inz (1-z_massmomerr)*f_I1z (1-2*z_massmomerr)*f_I2z (1-
z_massmomerr)*f_I3z (1-kc_err)*f_kcn (1-kc_err)*f_kc1 (1-kc_err)*f_kc2 (1-masserr)*f_mn
(1-masserr)*f_m1 (1-masserr)*f_m2];
ub = [(1+z_massmomerr)*f_Inz (1+z_massmomerr)*f_I1z (1+2*z_massmomerr)*f_I2z
(1+z_massmomerr)*f_I3z (1+kc_err)*f_kcn (1+kc_err)*f_kc1 (1+kc_err)*f_kc2 (1+masserr)*f_mn
(1+masserr)*f_m1 (1+masserr)*f_m2];

model_damping = -0.03;
fitparam = fitparam0;
```

```
%original error
modfit = 0; %#ok<NASGU>
ssmake4pv2eMB4f;
invMK = -mbquadA(25:48,1:24);
[phi_temp modes] = eig(invMK);
modes(1,:) = sqrt(diag(modes))/(2*pi);
modes(2:end,:) = [];
[sortmodes ind] = sort(modes);
sortmodes(23:24) = [];
err0 = scatterfit_err(fitparam);

%plotting original vertical model
mbquad.a(25:48,25:48) = model_damping*eye(24);
load M0TFMat
f = M0TFMat(2,2).freq;
measmag = 20*log10(abs(M0TFMat(4,4).resp));
measph = unwrap_pend(M0TFMat(4,4).resp,f);
[mag ph]=bode(mbquad(3,9),2*pi*f);
mag = 20*log10(squeeze(mag)) + 53;
ph = squeeze(ph);
figure,subplot(211)
semilogx(f,mag,f,measmag)
set(gca,'XLim',[f(1) f(end)])
title('Bode Plot. Original Vertical Model')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
legend('Model','Meas')
grid on
subplot(212),semilogx(f,ph,f,measph)
set(gca,'XLim',[f(1) f(end)],'YLim',[-190 10],'Ytick',-180:45:0)
xlabel('Frequency (Hz)')
ylabel('Phase (degrees)')
grid on

%plotting original yaw model
f = M0TFMat(2,2).freq;
measmag_yaw = 20*log10(abs(M0TFMat(3,3).resp));
measph_yaw = unwrap_pend(M0TFMat(3,3).resp,f);
[mag ph]=bode(mbquad(4,10),2*pi*f);
mag = 20*log10(squeeze(mag)) + 16;
ph = squeeze(ph);
figure,subplot(211)
semilogx(f,mag,f,measmag_yaw)
set(gca,'XLim',[f(1) f(end)])
title('Bode Plot. Original Yaw Model')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
legend('Model','Meas')
grid on
subplot(212),semilogx(f,ph,f,measph_yaw)
set(gca,'XLim',[f(1) f(end)],'YLim',[-190 10],'Ytick',-180:45:0)
xlabel('Frequency (Hz)')
```

```
ylabel('Phase (degrees)')
grid on

%Performing Optimization
options = optimset('MaxFunEvals',5000,'MaxSQPIter',50000,'TolFun',1e-10,'TolX',1e-10);
[finalvalue,err] = fminimax(@vertyawfit_err,fitparam,[],[],[],[],lb,ub,[],options);

% Amount of change in parameters. Useful to see what maxes out.
change = 100*(finalvalue-original_values)./original_values;

fitparam = finalvalue;
modfit = 23; %#ok<NASGU>
ssmake4pv2eMB4f;
invMK = -mbquadA(25:48,1:24);
[phi_temp modes] = eig(invMK);
modes(1,:) = sqrt(diag(modes))/(2*pi);
modes(2:end,:) = [];
[sortmodes_final ind_final] = sort(modes);
sortmodes_final(23:24) = [];
phi = phi_temp(:,ind_final);

%plotting new vertical model
mbquad.a(25:48,25:48) = model_damping*eye(24);
[mag_f ph_f]=bode(mbquad(3,9),2*pi*f);
mag_f = 20*log10(squeeze(mag_f)) + 53;
ph_f = squeeze(ph_f);
figure,subplot(211)
semilogx(f,mag_f,f,measmag)
set(gca,'XLim',[f(1) f(end)])
title('Bode Plot. Best Fit Vertical Model')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
legend('Model','Meas')
grid on
subplot(212),semilogx(f,ph_f,f,measph)
set(gca,'XLim',[f(1) f(end)],'YLim',[-190 10],'Ytick',-180:45:0)
xlabel('Frequency (Hz)')
ylabel('Phase (degrees)')
grid on

%plotting new yaw model
[mag_f ph_f]=bode(mbquad(4,10),2*pi*f);
mag_f = 20*log10(squeeze(mag_f)) + 16;
ph_f = squeeze(ph_f);
figure,subplot(211)
semilogx(f,mag_f,f,measmag_yaw)
set(gca,'XLim',[f(1) f(end)])
title('Bode Plot. Best Fit Yaw Model')
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
legend('Model','Meas')
grid on
```

```
subplot(212),semilogx(f,ph_f,f,measph_yaw)
set(gca,'XLim',[f(1) f(end)],'YLim',[-190 10],'Ytick',-180:45:0)
xlabel('Frequency (Hz)')
ylabel('Phase (degrees)')
grid on


%Error Bar Graph Data
% [phi_pitch ind_pitch] = filter_pitchmodes(phi,sortmodes_final);
mode_num = [4 5 10 13 14 17 21];
mm_num = [4 5 9 13 14 17 21];
for i = 1:length(mode_num)
    errpercent(i,1) = 100*(mm(mm_num(i)) - sortmodes(mode_num(i)))/sortmodes(mode_num(i));
%#ok<AGROW> %Before Fit
    errpercent(i,2) = 100*(mm(mm_num(i)) - sortmodes_final(mode_num(i)))/sortmodes_final
(mode_num(i)); %#ok<AGROW> %After Fit
end
max_mode_err = max(abs(errpercent(:,2))) %#ok<NOPRT>

%Plotting Mode Error Bar Graph
figure,bar(abs(errpercent))
% set(gca,'XLim',[0 23])
title(['Reduction in the Error of the Modes After Model Fit. Total Error reduced from ',
num2str(err0),' to ',num2str(err)])
ylabel('Percent error')
xlabel('Z1                                Yaw1                                Yaw2
Z2                          Yaw3                          Yaw4
Z3')
legend('Before Fit','After Fit')
grid on
modfit = 0;

%Summarizing and outputting the parameter changes
Results = {'Parameter','Original Value','Final Value','Change'
            'Inz',original_values(1),finalvalue(1),[num2str(change(1)),'%']
            'I1z',original_values(2),finalvalue(2),[num2str(change(2)),'%']
            'I2z',original_values(3),finalvalue(3),[num2str(change(3)),'%']
            'I3z',original_values(4),finalvalue(4),[num2str(change(4)),'%']
            'kcn',original_values(5),finalvalue(5),[num2str(change(5)),'%']
            'kc1',original_values(6),finalvalue(6),[num2str(change(6)),'%']
            'kc2',original_values(7),finalvalue(7),[num2str(change(7)),'%']
            'mn',original_values(8),finalvalue(8),[num2str(change(8)),'%']
            'm1',original_values(9),finalvalue(9),[num2str(change(9)),'%']
            'm2',original_values(10),finalvalue(10),[num2str(change(10)),'%']
%             'm3',original_values(11),finalvalue(11),[num2str(change(11)),'%']
            'Percent Error',   err0,              err,     [num2str(100*(err-err0)
/err0),'%']} %#ok<NOPRT>
```

```
function err = vertyawfit_err(fitparam) %#ok<INUSD>
mm = [0.4270 0.4580 0.4800 0.5550 0.6768 0.8030 0.9800 1.0390 ...
      1.4297 1.5830 2.0000 2.0970 2.2686 2.5195 2.6880 ...
      2.7520 3.1455 3.2100 3.4070 3.4370 3.6157 5.1510 0.5333 2.278];

modfit = 23; %#ok<NASGU>
ssmake4pv2eMB4f;
invMK = -mbquadA(25:48,1:24);
[phi_temp modes] = eig(invMK);
modes(1,:) = sqrt(diag(modes))/(2*pi);
modes(2:end,:) = [];
sortmodes = sort(modes);
sortmodes(23:24) = [];

mode_num = [4 5 10 13 14 17 21];
mm_num = [4 5 9 13 14 17 21];
%proportional error
err = 0;
for i = 1:length(mm_num)
    err = abs(sortmodes(mode_num(i))-mm(mm_num(i)))/mm(mm_num(i)) + err;
end
err = 100*err/length(mm_num);
```

Appendix C

Classical Control Design

```
function damper = design_classic_damper(choice)

load seismic_noise
load sensor_noise
f=logspace(log10(0.1),log10(100),5000);
Aw(1,:)=interp1(seismic_noise.freq,seismic_noise.amplitude(1,:),f,'nearest'); %#ok<AGROW>
Av(1,:)=interp1(sensor_noise.freq,sensor_noise.amplitude(1,:),f,'nearest'); %#ok<AGROW>

if strcmp(choice,'vert') || strcmp(choice,'all')
sys = generate_system('vert');
plant1=sys.plant(1,2);
freq=damp(plant1)/2/pi;
freq=freq(1:2:8);
freqmax=max(freq);

[al1,bl1,cl1,dl1] = lowpass(5,1);%lowpass
[ad1,bd1,cd1,dd1] = transdif(0.75,4,1); %phase lead
[ad2,bd2,cd2,dd2] = transdif(0.75,4,1); %more phase lead
[alp,blp,clp,dlp] = sculte(6.5,2,10.1,40,1); %low pass first part
[alp2,blp2,clp2,dlp2] = sculte(7,10,freq(4),20,1); %low pass second part
bum=bump(freq(1),10,4);
[al,bl,cl,dl] = series(ad1,bd1,cd1,dd1,al1,bl1,cl1,dl1);
[al,bl,cl,dl] = series(al,bl,cl,dl,ad2,bd2,cd2,dd2);
[al,bl,cl,dl] = series(al,bl,cl,dl,alp,blp,clp,dlp);
[al,bl,cl,dl] = series(al,bl,cl,dl,alp2,blp2,clp2,dlp2);
damper = ss(al,bl,cl,dl);
velocity_damper=zpk([0],[-2*pi*1],1);
damper=3000*damper*bum*velocity_damper;
end

if strcmp(choice,'yaw') || strcmp(choice,'all')
sys = generate_system('yaw');
plant1=sys.plant(1,2);
freq=damp(plant1)/2/pi;
freq=freq(1:2:8);
freqmax=max(freq);

[al1,bl1,cl1,dl1] = lowpass(9,1);%lowpass
[ad1,bd1,cd1,dd1] = transdif(0.6,6,1); %phase lead
[ad2,bd2,cd2,dd2] = transdif(0.4,6,1); %more phase lead
[alp,blp,clp,dlp] = sculte(7,2,10.1,40,1); %low pass first part
[alp2,blp2,clp2,dlp2] = sculte(8,10,11.5,20,1); %low pass second part
peak = 5;qp = 2;
p   = pi*peak*(-1/qp + i*sqrt(4 - 1/(qp^2)));
p   = [conj(p) p]';
lp2 = zpk([],p,1);
lp2 = lp2/abs(freqresp(lp2,0));
bum=bump(freq(1),10,6);
%start with just gain and then add on other stages
[al,bl,cl,dl] = series(ad1,bd1,cd1,dd1,al1,bl1,cl1,dl1);
[al,bl,cl,dl] = series(al,bl,cl,dl,ad2,bd2,cd2,dd2);
[al,bl,cl,dl] = series(al,bl,cl,dl,alp,blp,clp,dlp);
```

```
damper = ss(a1,b1,c1,d1);
velocity_damper=zpk([0],[-2*pi*0.2],1);
damper=40*damper*bum*velocity_damper*lp2;
end


function [a,b,c,d]=lowpass(f,k)
z=[];
p=-f*2*pi;

[a,b,c,d] = zp2ss(z,p,k);


function [a,b,c,d] = transdif(lf,hf,dcGain)

%transistional differentiator in state space representation
%
%[a,b,c,d] = transdif(lf,hf,dcGain);
%
%lf         start differentaition (Hz)
%hf         stop differentaition (Hz)
%dcGain         dc gain
%
%Stuart Killbourn (October 95)

z   = -2*pi*lf;
p   = -2*pi*hf;
k   = dcGain*(hf/lf);

[a,b,c,d] = zp2ss(z,p,k);


function [a,b,c,d] = sculte(peak,qp,notch,qn,dcGain)

%resonant 2-pole low pass filter in state space representaion
%
%[a,b,c,d] = sculte(peak,qp,notch,qn,dcGain);
%
%peak           frequency cut (Hz)
%qp         Q factor of resonance
%notch          notch frequency (above peak)
%qn         Q factor of notch
%dcGain         dc gain
%
%Stuart Killbourn (October 95)

z   = pi*notch*(-1/qn + i*sqrt(4 - 1/(qn^2)));
z   = [conj(z) z]';
p   = pi*peak*(-1/qp + i*sqrt(4 - 1/(qp^2)));
p   = [conj(p) p]';
k   = dcGain*(peak/notch)^2;
```

```
[a,b,c,d] = zp2ss(z,p,k);


function [sys]=bump(f0,Q,atten)
%NOTCH builds a bogus notch filter
%[sys]=notch(f0,Q,attenuation);
%f0 is the center freq, Q and atten control the width and depth
%
% see notch2 for the old version
%
% Brian Lantz Jan 23, 2003

top=1/Q;
bottom=atten/Q;

dhf=[1/(2*pi*f0)^2 2*top/(2*pi*f0) 1];
nhf=[1/(2*pi*f0)^2 2*bottom/(2*pi*f0) 1];
num=nhf;
den=dhf;
sys = tf(num,den);
```

```
function sys = generate_system(dof)

natural_damp = -0.001;

if strcmp(dof,'yaw')
load('Model/quad_yaw_bestfit.mat')
load('Model/seismic_noise.mat')
load('Model/sensor_noise.mat')
% natural_damp = -0.015; %from comparison of measurements to simulation
sys.plant=ss_model;
sys.plant.a(5:8,5:8) = natural_damp*eye(4);
sys.ndof=4;
% sys.plant.a(sys.ndof+1:end,sys.ndof+1:end) = -0.001*eye(sys.ndof);
sys.w_size=1;
sys.v_size=1;
sys.u_size=4;
sys.A=[1 0 0 0;0 0 0 0;0 0 0 0; 0 0 0 0];
sys.S=[1 0 0 0];
sys.leverarm = 0.240;
% sys.modal2realchange=ones(4);
sys.Awfreq=seismic_noise.freq;
sys.Awnoise=seismic_noise.amplitude;
sys.Avfreq=sensor_noise.freq;
sys.Avnoise=sensor_noise.amplitude;
end


if strcmp(dof,'vert')
load('Model/quad_vert_bestfit_a.mat')
% load('Model/quad_vert.mat')
load('Model/seismic_noise.mat')
load('Model/sensor_noise.mat')
sys.plant=ss_model;
sys.plant.a(5:8,5:8) = natural_damp*eye(4);
sys.ndof=4;
% sys.plant.a(sys.ndof+1:end,sys.ndof+1:end) = -0.001*eye(sys.ndof);
sys.w_size=1; %seismic
sys.v_size=1; %noise
sys.u_size=4; %inputs
% sys.A=[1 0 0 0;0 0 0 0;0 0 0 0]; %actuator
sys.A=[1 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0]; %actuator
sys.S=[1 0 0 0]; %sensor
% sys.modal2realchange=ones(4);
sys.Awfreq=seismic_noise.freq;
sys.Awnoise=seismic_noise.amplitude;
sys.Avfreq=sensor_noise.freq;
sys.Avnoise=sensor_noise.amplitude;
sys.leverarm = 1;
end
```

Bibliography

[1] Jim Hough, Sheila Rowan, B S Sathyaprakash. "The Search for Gravitational Waves." J. Phys. B: At. Mol. Opt. Phys. 38, S497–S519 (2005)

[2] Daniel Sigg. "Gravitational Waves." (1998). LIGO Publication Number P980007-00

[3] Peter R. Saulson. "Gravitational Waves - From Astrophysics to Optics." (2005). LIGO Graphics/Presentation Number G050252

[4] Peter R. Saulson. "Fundamentals of Interferometric Gravitational Wave Detectors." World Scientific (1994).

[5] Peter Fritschel. "Advanced LIGO Systems Design." (2001). LIGO Technical Document Number T010075-00

[6] N.A. Robertson, et al. "Seismic Isolation and Suspension Systems for Advanced LIGO." Gravitational Wave and Particle Astrophysics Detectors, Proceedings of SPIE, 5500, 81-91 (2004)

[7] Calum Iain Eachan Torrie. *Develepment of Suspensions for the Geo 600 Gravitational Wave Detector* (2000). PhD Thesis. Department of Physics and Astronomy. University of Glasgow.

[8] Mark Barton. "Models of the Advanced LIGO Suspensions in Mathematica." (2003). LIGO Technical Document Number T020205-00-D

[9] Mark Barton. "Pendulum Modeling in Mathematica and Matlab." (2007). LIGO Graphics/Presentation Number G070283-00

[10] Mark Barton, Phil Willems, Peter Fritschel, David Shoemaker, Norna Robertson. "Cavity Optic Optics Suspension Subsystem." (2001). LIGO Technical Document Number T010007-03

[11] N.A. Robertson, et al. "Quadruple Suspension Design for Advanced LIGO." Classical and Quantum Gravity. 19, 4043-4058 (2002). LIGO Publication Number P020001-A-R

[12] Stuart Aston. "Noise Prototype OSEM Development." (2006). LIGO Graphics/Presentation Number G060116-00

[13] Laurent Ruet. *Active Control and Sensor Noise Filtering Duality Application to Advanced LIGO Suspensions* (2002). PhD Thesis, Doctoral School of Mechanics, Energy, Civil Engineering, and Acoustics (MEGA). Institut National des Sciences Appliquées de Lyon (INSA Lyon)

[14] Stuart Aston, Clive Speake. "Geometric OSEM Sensor Development." (2004). LIGO Technical Document Number T040043-01

[15] Eduardo Kausel. *Advanced Structural Dynamics*. (2006). MIT 2.060 Course Notes

[16] Katsuhiko Ogata. *Modern Control Engineering, 4th Ed.* Prentice Hall (2002)

[17] Barry C. Barish, Rainer Weiss. "LIGO and the Detection of Gravitational Waves." Physics Today, 52, 44-50 (1999)

[18] Mark Barton. "Modeling of the AdvLIGO Quad Pendulum Controls Prototype." (2006). LIGO Graphics/Presentation Number G060086-00