Building a State Space Model of the GEO600 Control Architechture

Alexander Lombardi

August 11, 2010

1 Introduction

In 1916, Albert Einstein's Theory of General Relativity predicted the existence of gravitational waves. Such waves are expected to stretch and contract matter as they move like ripples on a pond through spacetime. In 1993, the Nobel Prize was awarded to Russell Hulse and Joseph Taylor for an indirect detection of gravitational waves from a binary pulsar system. Since then, the focus in the field of gravitational physics has been to directly detect gravitational waves. Although other methods have been tried, and false detections have been claimed, the best shot at actual detections unanimously lies with large scale interferometry. In the past two decades, huge strides have been made in the quest to directly detect gravitational waves using terrestrial laser-interferometers. Such large scale detectors have been built all over the world, creating an international network of detectors searching for gravitational waves from astrophysical sources such as binary systems of black holes or neutron stars, pulsars, and supernovae. There are three detectors in the United States that make up the LIGO experiment. One is located in Livingston, Louisiana with 4 km arms, and two are located in Hanford, Washington: one with 4 km arms and one with 2 km arms. VIRGO is a 3 km interferometer near Pisa, Italy. TAMA is a 300 m detector in Japan. The British-German collaboration, GEO600, is a 600 m detector located near Hannover, Germany. None of these detectors has produced a detection at the present time, but they've all achieved huge improvements in sensitivity, and the continued progression of technology ensures an event rate suitable to claim detection within the next several years. The GEO600 detector was built from 1995 to 2001 based on two prototypes from Glasgow University and the Max-Planck-Institut fr Quantenoptik. The project continues with the addition of the University of Cardiff, and the Albert-Einstein-Institut at Potsdam and Hannover.

2 The GEO600 Interferometer

The GEO600 detector is a dual-recycled folded arm Michelson interferometer. For over a century, variations on the original Michelson interferometer have been used to detect small changes in length. As gravitational waves pass through the interferometer, it changes the optical path length and therefore, the phase of the light in the arms of the detector. The measurement that we actually read from the interferometer then is not actually a length at all. The actual quantity that we measure from the detector is the change in light power on a photodetector in the interferometer output. Control systems (which we will discuss later) keep the interferometer operating on a dark fringe, which means that the light returning to the beam splitter interferes perfectly, and no light reaches the photodetector. Instead, the light returns back in the direction of the laser where it hits the power-recycling mirror and returns into the interferometer arms which allows the laser power to build up in the detector. This space where the beam power

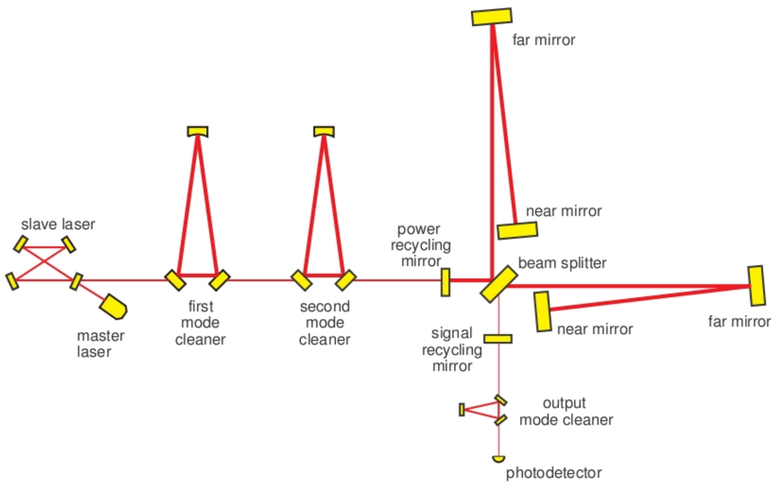


Figure 1: The optical layout of the GEO600 interferometer [5]

builds up is known as the power-recycling cavity. The interferometer uses a 12 W laser that passes through two mode cleaners before entering the beam splitter and power-recycling cavity. Because of this cavity, the power in the GEO600 detector reaches about 10 kW at the beam splitter. When the mirrors of the interferometer are displaced by gravitational waves, or some noise, the interference back at the beam splitter is not perfectly destructive, and some light escapes into the interferometer output. Here, there is another mirror that puts this light back into the interferometer to form the signal-recycling cavity. This amplifies this output signal to reduce the effects of shot noise on sensitivity. This signal is further enhanced by an output modecleaner. A simplified diagram of the beam path in the detector is shown in figure 1.

GEO600 is currently beginning an upgrade program called GEO-HF, the aim of which is to increase the high frequency sensitivity of the detector. GEO-HF will require some new technologies and changes to the current setup. One such upgrade is to use squeezed light as the input carrier beam. Other upgrades include replacing some mirrors, and increasing the power of the laser. Figure 2 shows the sensitivity curve of the current GEO600 detector as it compares to the LIGO and Virgo detectors. It also shows some reasonable predictions for how the GEO-HF upgrades will improve the high frequency sensitivity of the detector. These estimates suggest that GEO may match or exceed the sensitivity of every other gravitational wave interferometer in the world, for frequencies above 900 Hz.

I've already discussed how the limitations of the detector at high frequencies

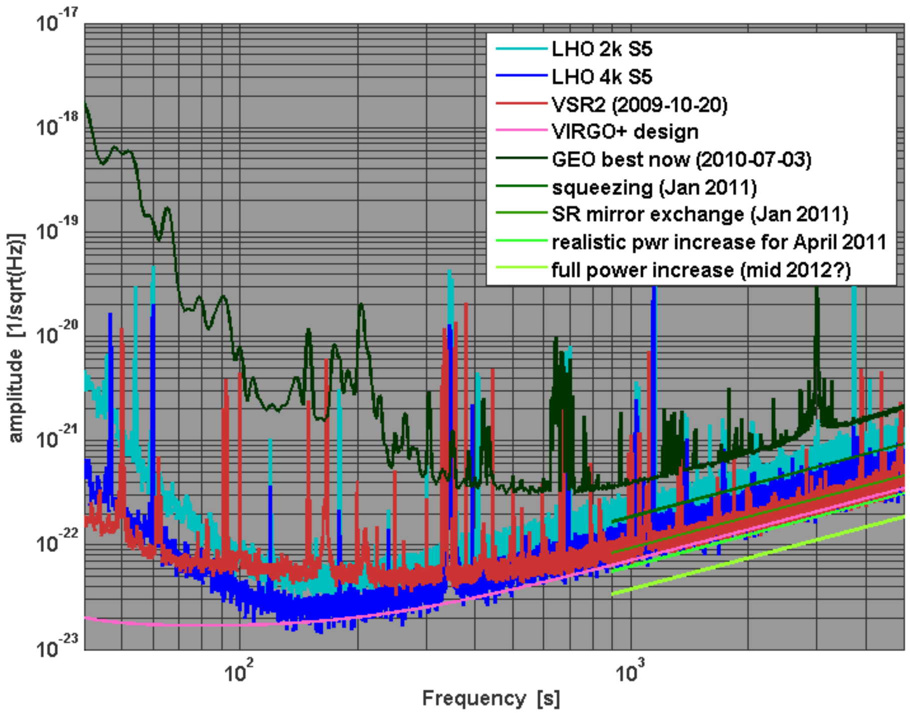


Figure 2: Sensitivity curve for GEO600 and projections for GEO-HF as compared to the current LIGO and VIRGO detectors [4]

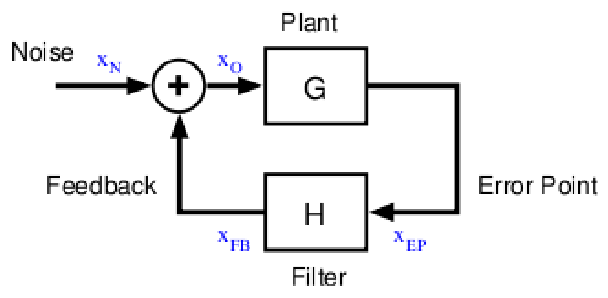


Figure 3: A simplified block diagram for a basic feedback-control loop. Here the two blocks are the plant and the servo. The sensor and actuator are taken to be part of the plant. [3]

are the result of shot noise on the output photodetector. However, these are not the only limiting factors on the detector. The detector sensitivity is limited in the lower frequency range below 40 Hz, due to seismic noise. The region in the middle has limitations that come from thermo-refractive noise, but I can say little else on this subject. For the purposes of this project, seismic noise is the biggest enemy. As with any interferometer, it is important to have arm lengths that are exactly the same. In an ideal situation (in the absence of all noise and a gravitational wave signal) the arm lengths of the detector are, in fact, equal, and the detector operates on a dark fringe. The most difficult task with these interferometers is isolating and controlling the interferometer components from noise. It is extremely important to have a quiet detector, so that real signals stand out. The best way to keep large gravitational wave interferometers at a sensitive operating point is the use of complex control systems.

3 Control Systems

3.1 Control Theory

Control systems are a method of maintaining some physical parameter at a predefined value in the presence of some force that may cause deviation in that parameter. For example, take a free particle moving in three-dimensional space. We want to keep this particle at some exact coordinate (the operating point), but there is some varying force pushing on it. A control loop would allow us to read out the position of the particle, or rather the deviation from the desired coordinate (the error point), and then produce some signal which uses an opposing force to push the particle back where we want it.

Control loops are made up of four basic parts. The physical system that you want to control is called the plant. There is a sensor, which measures the controlled parameter of the plant. It takes in the error point of the physical system, and puts out the error signal. This error signal becomes the input to the servo, which is any system that transforms this error signal into feedback. Finally, there is an actuator, which takes the feedback signal from the servo and uses it to restore the plant to its operating point. Usually, the sensor and actuator are considered as part of the plant when talking about feedback loops. We will see later how this makes sense in a state space model. The block diagram in figure 3 shows a control loop in its simplest form. All of the loops we make will be roughly this same form, but with more detail added to the components of the servo. In the case of the GEO600 interferometer, we wish to control optical systems: lasers, lenses, and mirrors. This work will focus on the end mirrors in the Michelson interferometer. Our error point will be the displacement and rotation of these mirrors due to seismic disturbance. The servo will be a series of filters that turn these errors into forces and torques which are applied to the mirrors by a series of actuators.

3.2 Building a Stable Loop

Any linear system can be fully characterized by its transfer function. The transfer function of a system is the ratio of the system's output to its input as a complex function of the input signal frequency. The transfer function, because it is complex, stores information about the amplitude and phase as a function of frequency. When building a stable control loop, we care about the open loop transfer function of the system. By open loop, we mean that the loop is not connected so that it is actively producing feedback for the system. We take all of the components of our control loop and connect them in series (exactly as in the closed loop but the final output does not return to the beginning to get summed into the input). The open loop transfer function, which we will call open loop gain, is then the ratio of the final output signal to the input signal as a function of signal frequency. The open loop gain of the system is the product of all of the individual component gains. Consequently, the order of our filters in the servo has no impact on the open loop gain.

From the open loop gain, we can tell if the control loop will be stable when it is closed. When closed, system can be in three different states. If the loop is designed correctly, noise in the system will be suppressed, and the loop succeeds in keeping the plant near its operating point. We call this loop stable or locked. If the gain is not set properly, the system may oscillate at some constant frequency, or if the gain is entirely too low, the system will be completely free running and it will behave as if the loop is not connected at all. The phase information about the open loop is also important, because if the phase is wrong, the feedback to the system may have the opposite of the desired effect, amplifying the noise in the system.

To make sure the loop is stable, we can plot the open loop transfer function and examine the amplitude and phase. We look at the phase at the point of

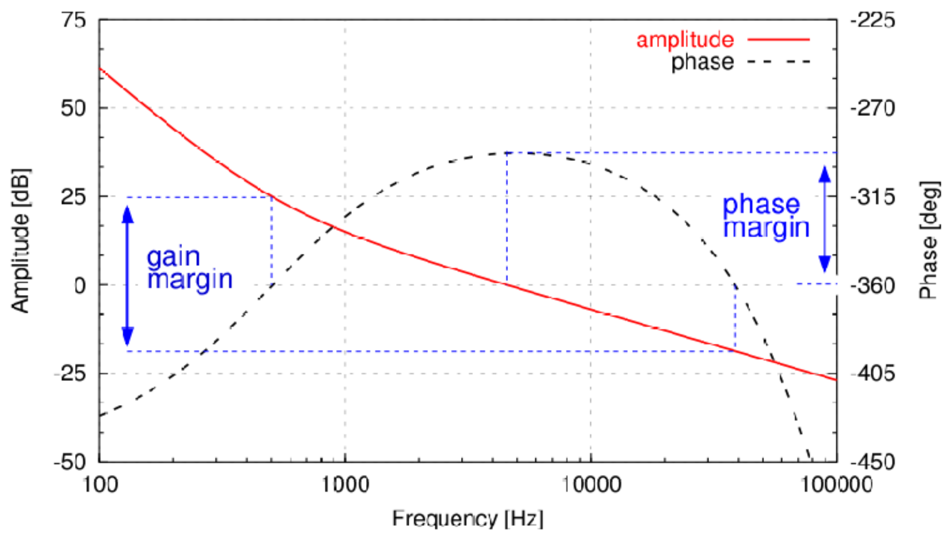


Figure 4: This shows some random open loop transfer function (amplitude and phase as a function of frequency). Notice at the point where the amplitude hits unity gain (0 dB) the phase reaches its peak, which is clearly above -180 degrees relative to DC. This is clearly a stable control loop. [3]

unity gain, where the absolute amplitude is one (0 dB). There is typically only one point in the feedback loop where the transfer function crosses unity gain. For the loop to be stable, the phase must be above -180 degrees (with respect to DC) at the unity gain point. In the case where the transfer function crosses unity gain in multiple places, the previously mentioned criterion must for every unity gain point. Figure 4 shows a conditionally stable loop. Loops are designed such that there is a fast drop off after the unity gain point in order to suppress noise better. The filters are usually chosen so that the phase bumps up just over the -180 degree mark near the unity gain frequency. Depending on the size of this bump, there will be a range of frequencies where the unity gain point can exist. This makes the loop design slightly flexible, though these margins are typically much smaller than that given in figure 4. If we look at the open loop transfer function of our control loop and it obeys the above criterion, we can be sure that the loop will work properly when it is closed.

4 State Space Modeling

A state space model is a method of representing some physical system as a series of first-order differential equations. These equations relate the state variables to the input and output variables of the system in order to create a model that completely describes the systems attributes. The name comes from the axes of the the model space, which represent each state variable of the system. The state space representation consists of the following two equations:

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx + Du \tag{2}$$

where u, y, and x are all vectors containing all of the input, output, and state variables respectively. (1) is called the state equation. (2) is the output equation. A and B are matrices that relate the input and state vectors to the first time-derivative of the state vector. The C and D matrices relate the input and state vectors to the outputs of the system.

A state space model can be created for any linear system, for which you can solve the equations of motion.[1] These equations can be rearranged to meet the form of (1) and (2), and from these equations, it is easy to pick out the state variables to fill our x vector. The redundant state variables are eliminated, if possible, by recognizing them as time-derivatives of the other state variables. Once the state vector is identified and complete, the other elements of the equations are appropriately entered in to the A, B, C, and D matrices. Their position in the appropriate matrix depends on which state variable they scale (if any), and their effect on the inputs and outputs of the system.

The A matrix is called the state matrix. It contains the meat of the model. Most of the information about the system will be contained in the A matrix. B is the input matrix. It contains all information about how the system is affected by the inputs. The C matrix relates the state vector directly to the output vector. This matrix will allow you to choose which variables are read out in the model and in what order they appear in the output vector. Of course, you could make the outputs some linear combination of the state variables, but this was not necessary in any of my models, and no example will be given. The D matrix allows for direct connection of inputs to outputs. For my purposes, it is always full of zeros and will largely be ignored in discussion. The dimensions of each matrix depend on the number of state, input, and output variables. This will become obvious in the following example.

4.1 Example - A Simple Pendulum

A great deal of my time on this project was spent figuring out how to build and use state space models in MATLAB. The following example is the first state space model I created. Because it is simple, and relevant to the rest of this work, we now consider an undamped, perfectly rigid, single pendulum of mass, m. We assume that the pendulum rod of length L has no mass. The equation of motion for this pendulum is

[1]Nonlinear systems may also be modeled, but it is a more complicated procedure that will not be discussed in this paper.

$$\ddot{\theta}L = -g\sin\theta + F(t) \tag{3}$$

$$\ddot{\theta} = -\frac{g}{L}\sin\theta + \frac{F(t)}{L} \tag{4}$$

where θ is the angular displacement of the pendulum from its rest position on the vertical axis, and g is the acceleration due to gravity. We see immediately that this equation does not describe a linear system. However, since we are only interested in a pendulum with a very small range of motion, we can use the small angle approximation, $\sin\theta \approx \theta$. Within this limit, equation (4) becomes

$$\ddot{\theta} = -\frac{g}{l}\theta + \frac{F(t)}{l} \tag{5}$$

From this, we now know our state and input vectors and can build the A and B matrices.

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \qquad u = F(t) \tag{6}$$

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{g}{L} & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} \tag{7}$$

We then choose our C matrix so that we can read out the angular displacement and angular velocity of the pendulum.

$$y = x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{8}$$

Referring back to the our original equations for building a state space model (considering D to be a zero matrix), our model consists of the following two first-order differential matrix equations.

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{L} & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} F(t) \tag{9}$$

$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} F(t) \tag{10}$$

The input $F(t)$ can be considered as some force pushing on the pendulum at its center of mass. Later, we will see how this corresponds to forces on the different masses in the triple pendulum suspention in the GEO600 interferometer.

4.2 Constructing a State Space Model in MATLAB

Once you have compiled all of the necessary information about the system you wish to model, MATLAB makes it very easy to build and connect state space models and test them. To build the model for a physical system such as the pendulum I've just described, you create the A, B, C, and D matrices and put them into your MATLAB code. There is a built in function, which takes these matrices and forms the model. If you intend to connect this model to other models, it is necessary to be diligent about keeping track of your inputs and outputs and the order which they appear in your code. Alternatively, and perhaps a more usefully, you can give names to each input, output and state in your model and use this information to connect models, and better keep track of what is happening in your model. The following is the MATLAB code for the simple pendulum example. It takes in some constant values, builds the matrices and then creates the state space model with labels.

```
g = 9.80665; % units: m/s^2
l = 1.00; % units: m

A = [0 1; (-g/l) 0];
B = [0; 1/l];
C = [1 0];
D = [0];

pendulum_ss = ss(A,B,C,D,'inputname',{'Force'},...
    'outputname',{'Displacement'});
```

In order to create our control loops we will also need to build state space models for the filters in the servo. Here, it is not so intuitive to build the matrices for the model. However, MATLAB still makes things easy for us. There are a series of functions that allow you to build a state space model directly from the transfer function of the system, which can often be measured using a spectrum analyzer. If you know the transfer function, you can enter the coefficients of the polynomials that make up the numerator and denominator, and MATLAB will use this information to build the state space matrices for you. There is another function which allows you to enter the poles, zeros, and gain for your filter, and the matrices are built from there. This is often more convenient as the interferometer documentation tends to specify the pole-zero combinations rather than the transfer function.

When we want to examine the transfer function, or check the stability of our loop, we want to plot this information and analyze it graphically. MATLAB allows you to plot the frequency response of any state space model or combination of models with a single function called bode. A Bode plot is a graph of a transfer function. It graphs the amplitude and phase as a function of frequency. MATLAB also makes it very simple to test the ramifications if arbitrary impulses in the system. We can put in any function we want and see how the system responds over time. The function that we use to do this will depend on

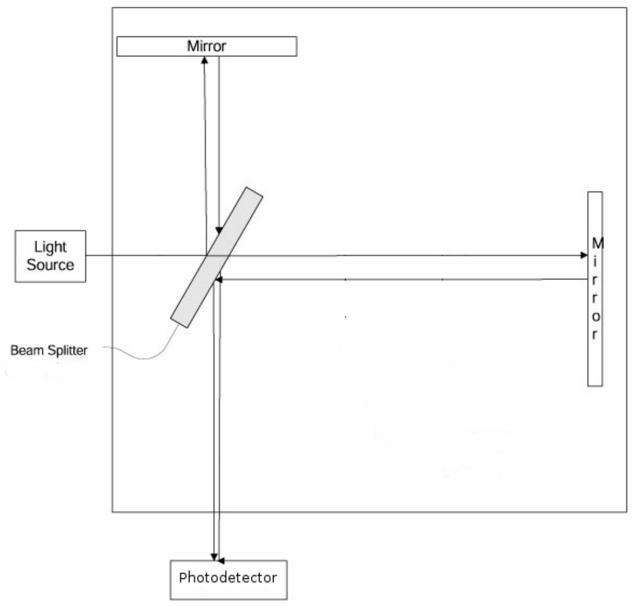


Figure 5: The simplified version of the interferometer as we imagine it for the sake of this model.

the type of input we want to simulate. We will see how this is useful when we build the actual model.

5 Building the Model for the Michelson Loop

Now that we know how to build stable control loops and state space models we can try to model a real system. The GEO600 interferometer requires hundreds of control loops to keep its components stable. For the two month duration of this project, it seemed most sensible to focus on the largest, most important loop so that a more complex model could be built from this later. The main Michelson loop is the control for the arm differential in the interferometer. The way you use an interferometer to take measurements is by finding the difference in the arm lengths of the detector. In a perfectly quiet environment the GEO detector would have perfectly equal arm lengths so that the difference between them is zero and the output to the photodetector is complete darkness. To tackle this project we thought it best to work with a simplified version of the interferometer. In this model, we treat the GEO600 detector as the most basic interferometer as in figure 5. We have a laser, beam splitter, and photodiode that are all treated as completely stationary. We ignore the mode cleaners and recycling cavities so that we are left to model only the suspensions at the end of each arm. In the real detector, the arms are folded, so there is an additional set of mirrors, but we also omit these for simplicity. Not that this is an extreme simplification of the system optics, but once we have a stable model, adding in new components is a rather trivial task. This model will be the foundation of the all inclusive model that can be built over the next several years.

5.1 The GEO600 Triple Pendulum Suspension

To model the Michelson loop, we have two physical systems to model: the triple-pendulum end suspensions. These two suspensions are identical, so we build one set of matrices and build two state space models with different input and output names so that we can differentiate between the two. The triple pendulum suspension is not as nice as the simple model we've already worked out. A diagram of the suspension, with all variables as they appear in the code, is given in Appendix A. Appendix B provides a more general view of the suspension from the two different views [1]. The three masses of the suspension are each attached to the mass above by four wires. In the case of the upper mass, the suspension wires are attached to cantilevers that are connected to the table, which is connected to the ground. For each mass in the pendulum, there are six degrees of freedom: three of dimensions of displacement, and three of rotation. Because these masses are attached by several wires, the masses motion is not only coupled with every other mass, but certain degrees of freedom are coupled together as well. The six degrees of freedom are longitudinal, sideways, vertical, pitch, roll, and yaw. Longitudinal is displacement in the direction of the beam path. This is coupled with the pitch, which is rotation of the mass in the direction of the beam path. Vertical is straight up and down. Yaw is rotation about the vertical axis. Sideways is the side to side swinging of the mass, perpendicular to both the longitudinal and vertical degrees of freedom. It is coupled with roll, which is rotation about the axis of the beam path. These couplings lead to a very complicated set of equations. Fortunately, these equations were solved in 2001 in C. Torrie's PhD thesis.

Because some of these degrees of freedom do not affect the others at all, it makes sense to build four separate models for the two pairs of coupled, and the two uncoupled degrees of freedom. All four models were built in the aforementioned thesis, but they have been updated several times since, to more accurately represent the actual suspensions. I have modified them further for the purposes of modeling the control systems. Since my main goal was to model the main Michelson loop of the interferometer, we really only care about the longitudinal and pitch degrees of freedom. The other models have been set up so that they may be implemented later if necessary, but this is where my work with them stops. From now on, we are dealing only with our simplified version of the interferometer, where the end mirrors only move in two (coupled) degrees of freedom. The actual state space model was built in MATLAB, and the code is given in Appendix C. The equations of motion were worked out by C. Torrie, which describe the coupled longitudinal and pitch motion of the masses in the suspension, but will be omitted here, since they are not part of my work. They can however be easily derived from the A matrix in the code.

For the purpose of implementing local controls, which will be discussed in the next section, it will be useful to know the resonant modes of this suspension system. We can find these resonant modes by taking the A matrix that we've devised, and finding the eigenvalues. I have created a MATLAB program which calculates these values from the A matrix written in the suspension model. This resonant mode calculator is given in Appendix D.

5.2 Building the Local Controls

Before tackling the filters that make up the Michelson loop it is necessary to build the local controls for each suspension system. The local controls only affect the system to which they are attached. The local controls are there to damp the resonant modes that we calculated using the program in Appendix D. Our model has four local control models: one for each degree of freedom on each pendulum. The filters used for the local controls are well documented, so I needed only the transfer function for each filter and to appropriately name the inputs and outputs. The code which creates these filters is not contained in this paper because it is fairly trivial. The block diagram for this model is given in shown in figure 6.

To make sure that this loop is actually working, we open the loop and use the bode function as we discussed before. We examine the open loop transfer function as shown in figure 7. Notice that you can see the resonant frequencies in the amplitude. At these points, the amplitude crosses unity gain several times, but we see that the phase is generally increasing over this range, and our criterion for stable feedback holds.

5.3 Building the Michelson Filters and Connecting the Loop

Until this point, we have largely ignored what the Michelson loop is doing. As with any interferometer, the piece of information we are most interested in is the difference in the arm lengths of the interferometer, which is measured by the laser path distance. This control loop reads out the displacement of the end mirrors, finds the difference between them, and then puts whatever force is necessary back on one or both of the mirrors so that the difference in arm lengths returns to zero. A sketch of this loop is given in figure 8. While the local controls focus on controlling the motion of the mirrors themselves, the Michelson controls are controlling the the difference between the lengths in the arms. This requires many more filters thank the local controls did, but the process is exactly the same. The filters used to control this parameter are well documented, so we may use their transfer function, pole-zero combinations, or

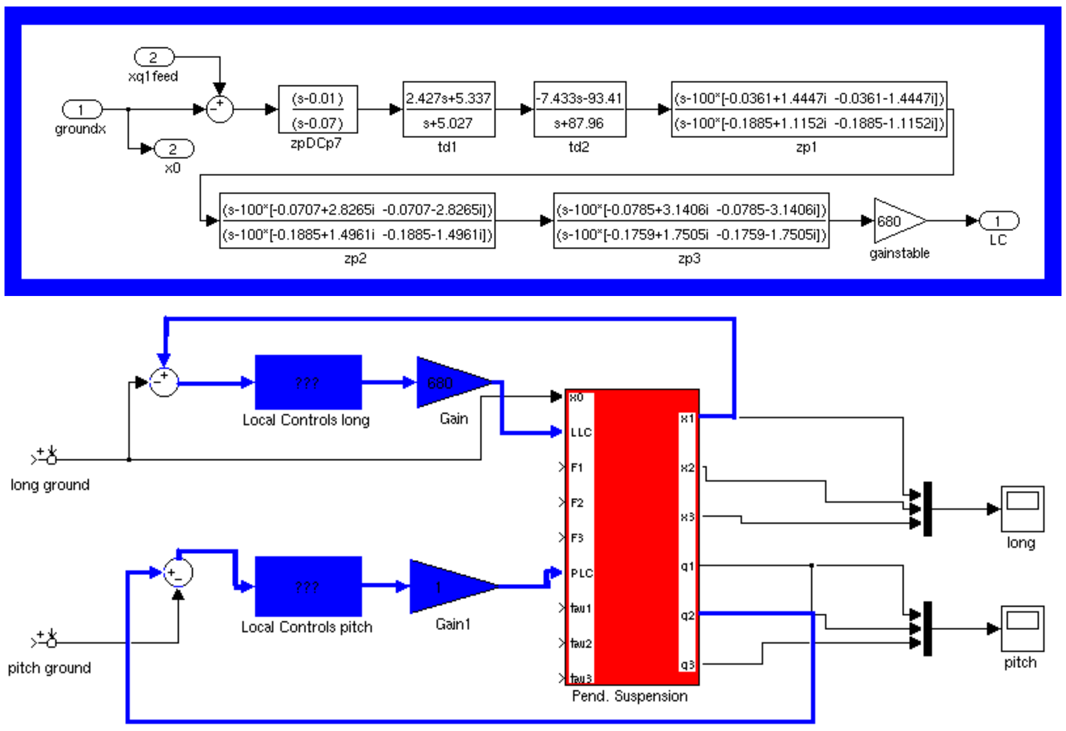


Figure 6: Top: Block model for the filters in the local controls. Bottom: How the local controls connect to the suspension systems.

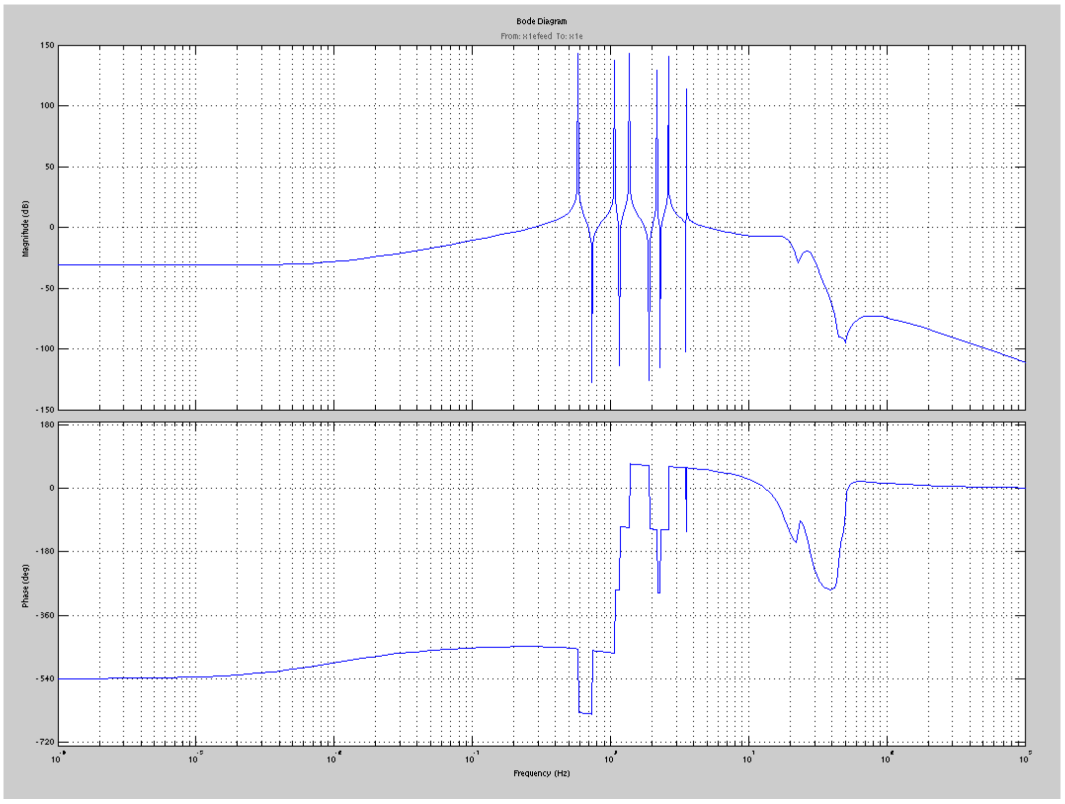


Figure 7: Open loop gain for the suspension with local controls attached.

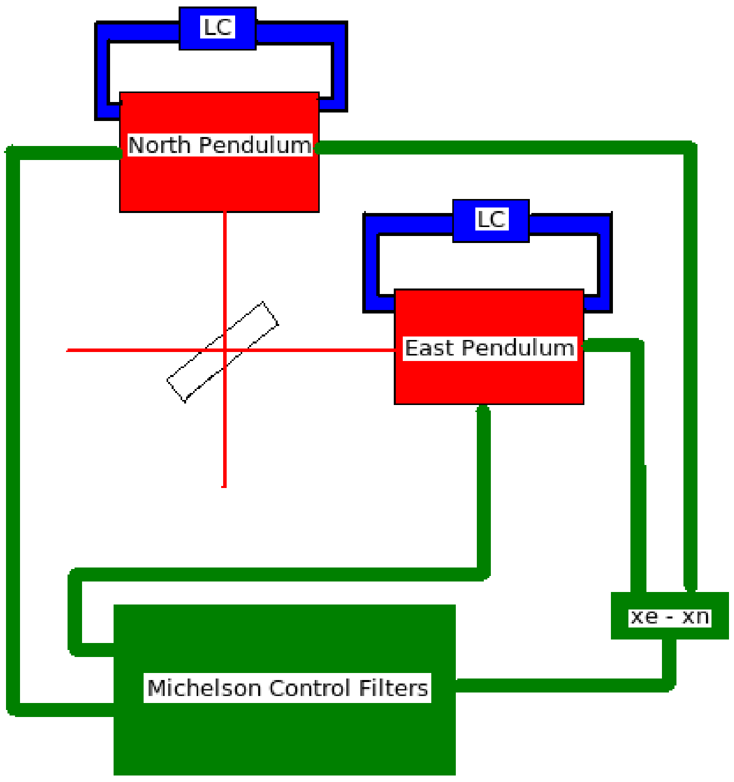


Figure 8: Block model diagram for the Michelson control loop

notch frequency to build each filter and then connect them by name. A block diagram containing each of these filters is shown in figure 9.

Now that we have built the pendulum model, the local controls and the Michelson controls, we must connect everything and test it out. Figure 8 shows the general picture of how the Michelson loop is connected. Figure 10 is the all inclusive diagram of the state space model for the Michelson loop. Every block, including the summations and the split-path points has its own state space model. The original diagram (without the fancy color coding) can be found in the Appendix.

5.4 Testing the Model

The model is now completely constructed, but we must remember that every feedback loop must satisfy a specific phase criterion. As before, we look at the open loop transfer function of the system (figure 11). In this plot we see that the unity gain point is right around 150 Hz. In the filters for the Michelson loop, we

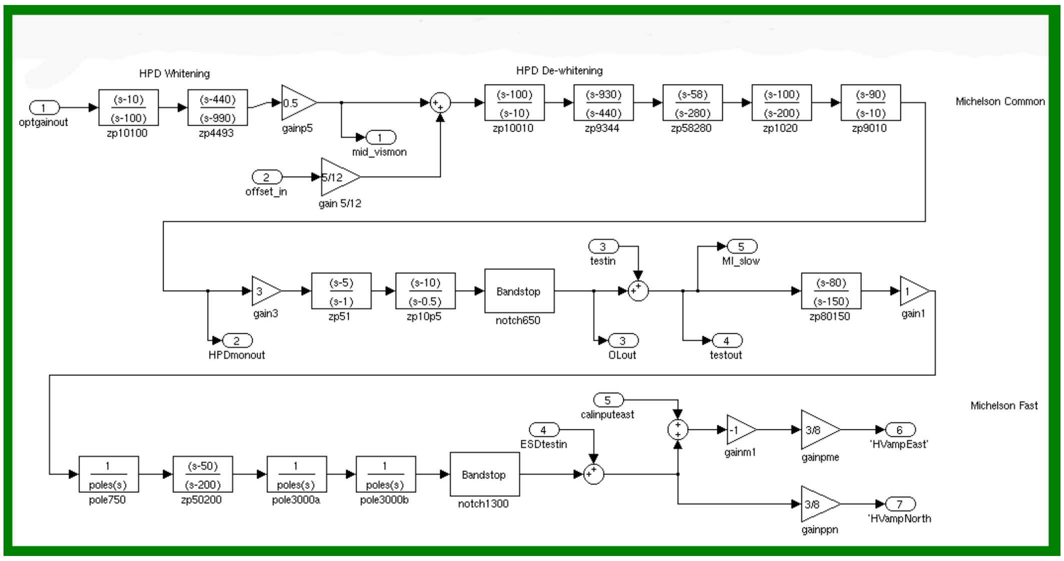


Figure 9: Block Diagram of all filters in the Michelson controls

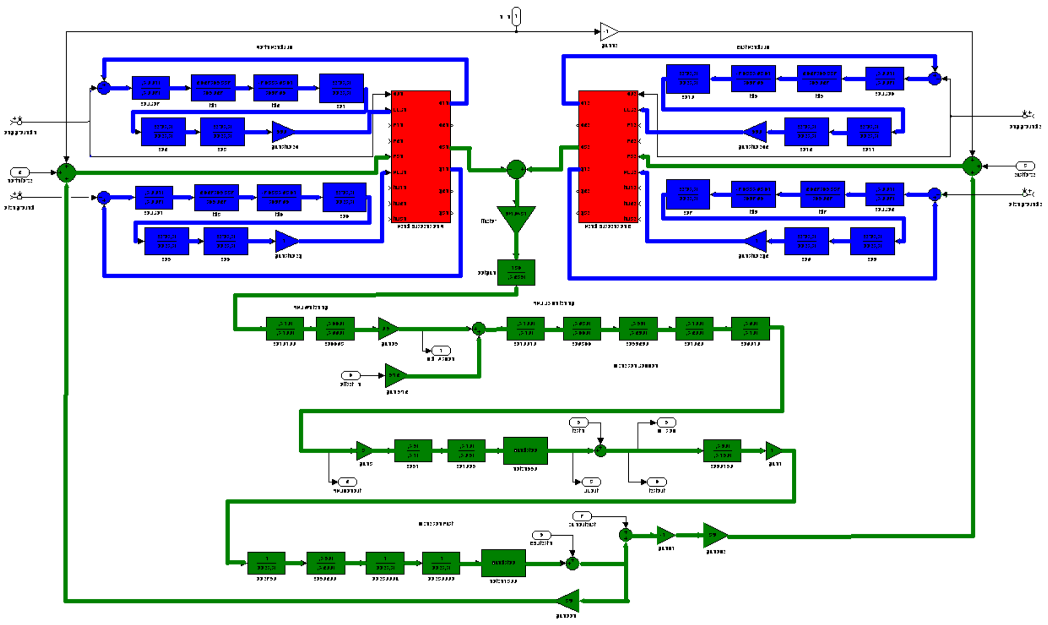


Figure 10: Block Diagram for the entire Michelson control loop. This includes the North and East pendula, each with 2 local controls, the summing block which finds the arm difference, the optical gain, and all of the michelson filters.

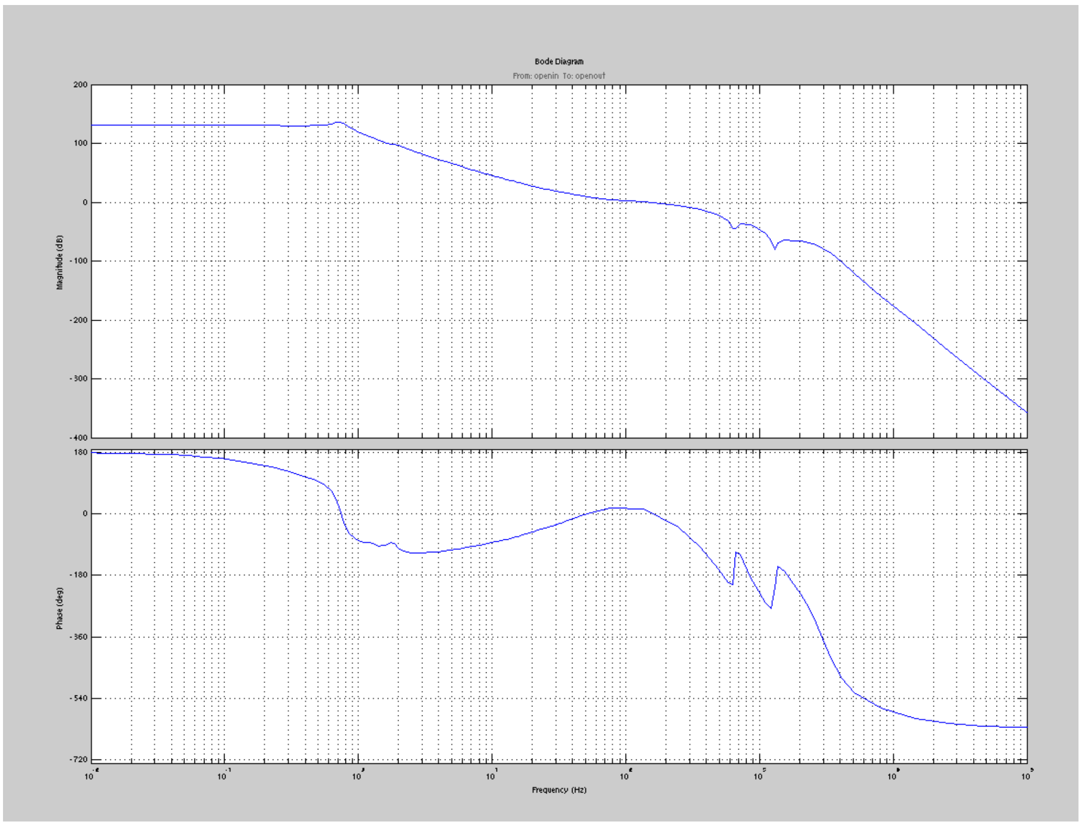


Figure 11: Open loop transfer function for the entire michelson loop.

provide an appropriate amount of gain for this to occur. We want this because the filter design gives us a bump in phase with a small margin right around 150 Hz. We see then that the phase is above 0 degrees (-180 with respect to DC) for a small frequency range surrounding the unity gain frequency. This phase margin creates the stable loop that we desire and we are now ready to test the model.

We must remember that the Michelson loop's job is to make the difference in the interferometer arm lengths zero. So we test to see what happens if we push on one of the mirrors with some arbitrary unit force impulse. The results of this simulation are shown in figure 12. We see that if you push on one mirror, the other mirror responds by moving with the same direction an amplitude. Pushing on the other mirror works the same way. If both mirrors are being displaced by the same amount, the arm difference is zero as we want it to be. In this case, the loop is working to keep one pendulum moving with the other, and takes about 12 seconds to ctop oscillating. Now if we apply a differential impulse to both mirrors simultaneously (figure 13), the mirrors move opposite

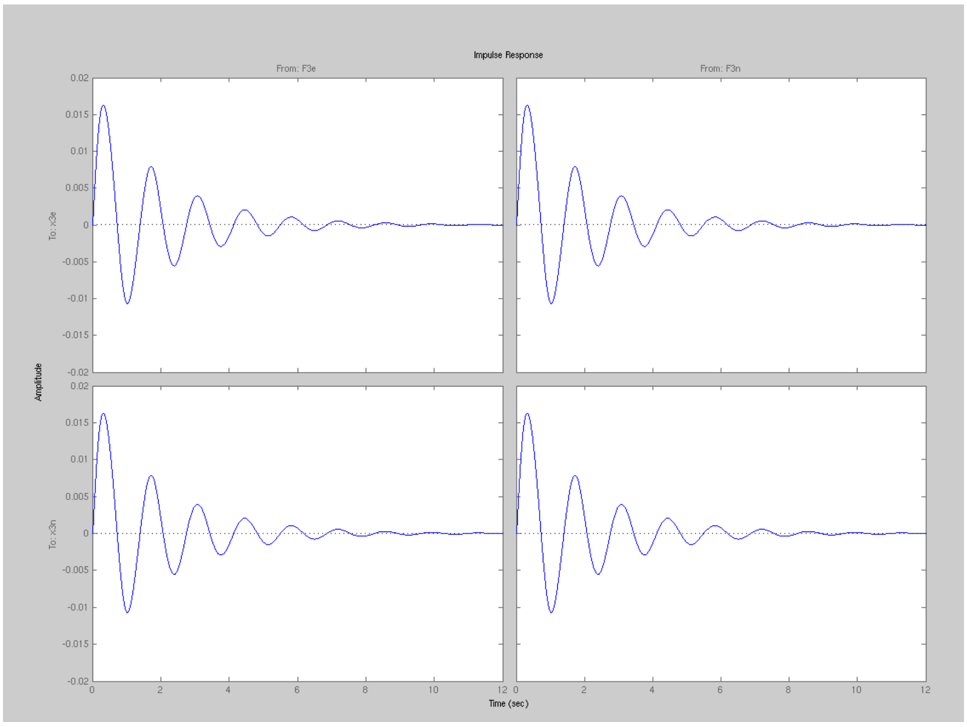


Figure 12: Test impulse on east and north pendula, separately, to view how the Michelson loop responds on the other pendulum. Mirrors oscillate sympathetically.

each other with the same amplitude, as they should. We notice in this case that it takes the masses about 0.1 seconds to come to a complete stop, much shorter than in the impulse (of equal magnitude) in figure 12. This means that our loop is also doing a good job of damping the oscillations.

Now that we've checked that the model reacts the way its supposed to, the last thing I wanted to do while working at GEO600, was to see how accurately this model reflects the real detector. To do this we plot the open loop transfer function for the model on top of the open loop transfer function of the real interferometer. To find the open loop transfer function in the GEO600 interferometer, we had to create a new test input and two outputs in the loop. The only way to get the transfer function was to take a measurement while the detector was in lock. The measurement was taken by connecting the new input and outputs to a spectrum analyzer. Signals over a large range of frequencies were put into the input and read out immediately after they were put into the loop. The other output was put just before the new test input, so we could take the ratio of outputs at the end of the loop to those just after the input, giving us a large set of complex points, which collectively form the open loop transfer function

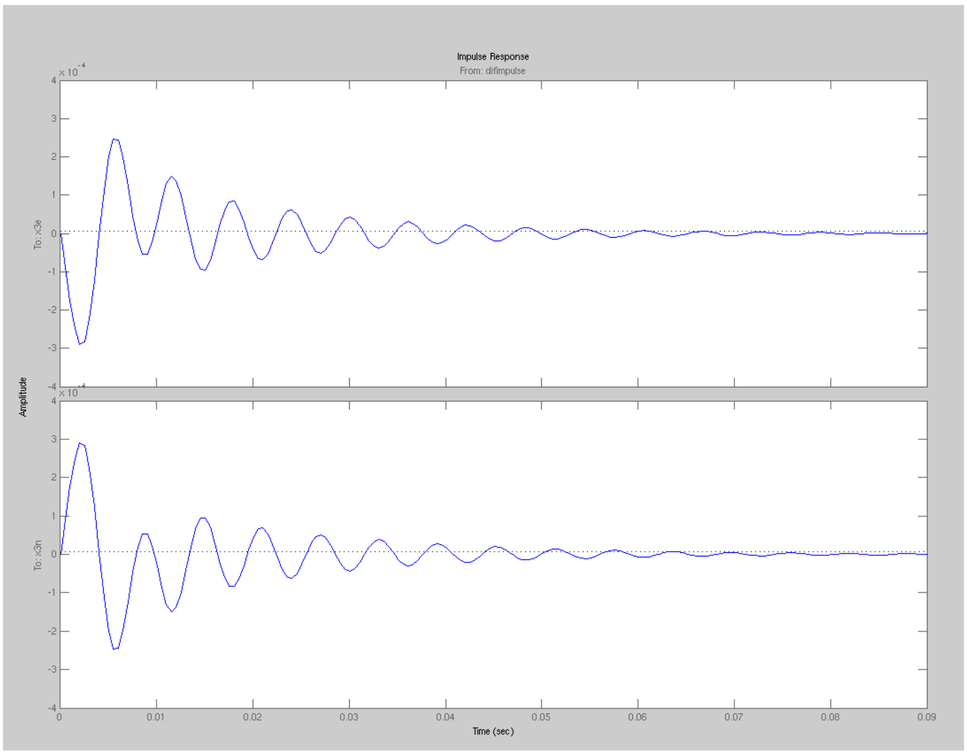


Figure 13: Response to differential impulse on end mirrors. Mirrors oscillate in opposite directions with equal magnitude.

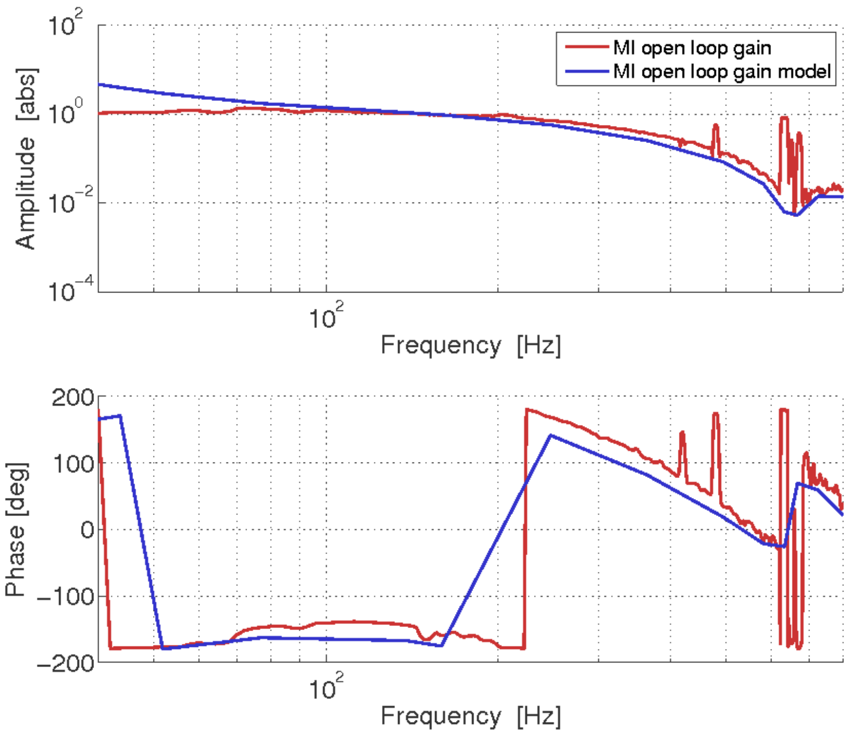


Figure 14: Comparison of the state space model to the actual GEO600 Michelson loop. The model is in blue, while the actual detector is in red. Data points for the red curve were taken around the unity gain frequency

of the real system. The model easily generates this transfer function with a few minor adjustments to the input and output names and the bode function. This comparison plot is given in figure 13. Data points for the detector transfer function were taken around the unity gain frequency because it is relatively well behaved. We see that in this range, the model matches with the real detector quite well. Out side of this range, the transfer functions diverge quite a bit, but since we are modeling the control system, we mostly care about the frequencies around unity gain.

5.5 The Future of the State Space Model

For the limited time that I spent at GEO600, it seems that this state space model is a perfectly good representation of the real detector. As mentioned earlier, this is a heavily simplified version of the Michelson interferometer. The hope is that in the near future, someone can pick up this work and add in many of the components that I've neglected for the sake of time. This will lead to a much more accurate model that could be extremely useful in creating a quieter detector. The model can be used to verify the calibration of the detector. It will also be a powerful tool for investigating glitches in the detector. If there is a recurring glitch somewhere in the detector, the state space model can be useful in hunting the source, because you can easily inject arbitrary signals into the model at any point you wish, and examine the ramifications. This state space model has the potential to be an extremely important element in improving the data quality from the GEO600 detector.

Acknowledgements

I would like to thank Prof. Bernard Whiting, Prof. Guido Meuller, and the University of Florida Physics Department for accepting me into this program. Antonis Mytidis and Kristin Nichola for arranging all of my travel and finances, checking in on me during my travels, and especially for picking me up at the Jacksonville airport in the middle of the night. Prof. Laura Cadonati for mentoring me, assisting me in becoming a part of this program, and being an all-around excellent advisor. Prof. Bruce Allen and all of the people at the Albert Einstein Institute in Hannover for accepting me into their group and making me feel important. Also, for inviting me out with them during and after work for lunch or drinks. Hartmut Grote and the rest of the the GEO family for letting me take up space and occasionally kick the detector out of lock, and for making me feel like a part of their family. A special thanks to Marc, for cooking nearly all of my lunches. They were delicious. Most of all, thank you to Dr. Jonathan Leong, for providing all the knowledge and encouragement I could ask for in this project, being an excellent mentor, and also showing me an excellent time in Germany. Lastly, I would like to thank the National Science Foundation for funding this program. Without their support, none of this would have been possible.

Appendix

Appendix C

```
% pendlocal.m - Alexander Lombardi - July 6, 2010
% Generates state space model for triple pendulum in longitudinal
    and pitch
% degrees of freedom and calculates the transfer function of the
    system
% credit: Calum Torrie and Ken Strain for all input values and
    equations of
% motion

global pend       %% declares global variable "pend"
extra =0;  g =9.8; bd=0;
pend.title = 'Pendulum parameters and derived properties';

%*******************************************************************

% UPPER MASS CALCULATED FROM MOFI2.m (C:/accounts/calum/3pend/jif)

%% dimensions of UPPER MASS (square)
ux                 = 0.1;
uy                 = 0.3;
uz                 = 0.07;
I1x                = 0.0459+(uz/2+0.02)^2*extra;  %KAS %% moment of
    inertia (roll)
I1y                = 0.0072+(uz/2+0.02)^2*extra;  %KAS %% moment of
    inertia (pitch/longitudinal)
I1z                = 0.0583;                      %ignore extra KAS
    %% moment of inertia (yaw)
m1                 = 5.7+0.1+extra;   % +0.1 for clamps and blades
    etc.. +0.4 for MCE  KAS

%*******************************************************************

%dimension of INTERMEDIATE MASS (cylinder)
ix                 = 0.1;             %% intermediate mass thickness
ir                 = 0.09;            %% intermediate mass radius
den2               = 2202;            % density (fused silica)
I2x                = 0.0227;          %% moment of inertia (roll)
I2y                = 0.01601;         %% moment of inertia (pitch/
    longitudinal)
I2z                = 0.01601;         %% moment of inertia (yaw)
m2                 = 5.6;             % calculated from bladejif2.
    xls
%********************************************************************

%dimensions of TEST MASS (cylinder)
tx                 = 0.1;             %% test mass thickness
tr                 = 0.09;            %% test mass radius
den3               = 2202;            % density (fused silica)
I3x                = 0.0227;          %% moment of inertia (roll)
I3y                = 0.01601;         %% moment of inertia (pitch/
    longitudinal)
I3z                = 0.01601;         %% moment of inertia (yaw)
m3                 = 5.6;             % calculated from bladejif2.
    xls
%*******************************************************************


l1                 = 0.42;            % upper wire length
l2                 = 0.187;           % intermediate wire length
l3                 = 0.28;            % lower wire length

%*********************************************************************************************


nw1                = 2;               % number of wires per stage (2
     or 4)
```

```
nw2                 = 4;                % = number of cantilevers (if
    fitted)
nw3                 = 4;

%**************************************************************************************


r1                  = 250e-6;           % radius of upper wire
r2                  = 175e-6;           % radius of intermediate wire
r3                  = 135e-6;           % radius of lower wire
                                        % Geppo measurements of bounce
                                             frequency uncoupled

%**************************************************************************************


Y1                  = 1.65e11;          % Youngs Modulus of upper wire
            (s/steel 302)
Y2                  = 1.65e11;          % Youngs Modulus of
    intermediate wire (s/steel 302)
Y3                  = 7e10;             % Youngs Modulus of lower wire
            (fused silica)

%*********************************************************************


ufc1                = 2;                % uncoupled mode frequency of
    cantilever stage(=0 for no cantilevers)
ufc2                = 3;
ufc3                = 0;

% NB:- uncoupled mode frequency- the frequency observed for a
    cantilever in a particular
%      stage supporting only the mass of that stage

%******************************************************************************************

%these parameters are least certain - likely to have major errors in
     d0 -
%d2 due to blade deflection and added masses (added masses NOT in
    model for
%MCe/n mirrors

d0                  = -0.004;           % height of upper wire break-
    off (above c.of m. upper mass)
d1                  = 0.014;            % height of intermediate wire
    break-off (below c.of m. upper mass)
d2                  = 0.001;            % height of intermediate wire
    break-off (above c.of m. of int. mass)
d3                  = 0.009;            % height of lower wire break-
    off (below c.of m. intermediate mass)  silica fibre flex from BS
d4                  = 0.009;            % height of lower wire break-
    off (above c.of m.test mass)  silica fibre flex from BS

%******************************************************************************************


% X direction separation

su                  = 0.00;             % 1/2 separation of upper
    wires
si                  = 0.03;             % 1/2 separation of
    intermediate wires
sl                  = 0.005;            % 1/2 separation of lower
    wires

%******************************************************************************************


% Y direction separation
```

```
n0                 = 0.03;            % 1/2 separation of upper
    wires at suspension point
n1                 = 0.04;            % 1/2 separation of upper
    wires at upper mass


n2                 = 0.045;           % 1/2 separation of
    intermediate wires at upper mass
n3                 = ir-0.0035+0.01;  % 1/2 separation of
    intermediate wires at intermediate mass


n4                 = ir-0.0035+0.005; % 1/2 separation of lower
    wires at intermediate mass
n5                 = tr-0.0035+0.005; % 1/2 separation of lower
    wires at test mass

% NB:- i.e. n4 = (radius) - (flat) + (break-off bar)

%*********************************************************************************


% local control units mechanical details

leverarmrt         = 0.03;            % half spacing of coils acting
     on tilt, rt
leverarmrz         = 0.08;            % half spacing of coils acting
     on rotation, rz
leverarmrl         = 0.08;            % half spacing of coils acting
     on roll, rl

lever_pitch        = leverarmrt;      % notation change
lever_roll         = leverarmrl;      % notation change
lever_yaw          = leverarmrz;      % notation change

%local control gains
gain = 0.4;
gainzrtrl = gain; % vertical, z, tilt, rt, roll rl (coils on top of
    upper mass)
gaint = gain.*2; % sideways, t (coil on one end of upper mass)
gainlrz = gain; % longitudinal, l, rotation, rz (coils on long rear
    side of mass)

kc1 =  1/2*(2*pi*ufc1)^2*m1;  %% spring constant of cantilever stage
     1
kc2 =  1/2*(2*pi*ufc2)^2*m2;    %% spring constant of cantilever
    stage 2

kw1 =Y1*pi*r1^2/l1*nw1/2;     %% spring constant of upper wire (s/
    steel 302)
kw2 =Y2*pi*r2^2/l2*nw2/2;     %% spring constant of intermediate
    wire (s/steel 302)
kw3 =Y3*pi*r3^2/l3*nw3/2;     %% spring constant of lower wire (
    fused silica)

if (kc1==0)
  k1=kw1;
else
  k1=kc1*kw1/(kc1+kw1);       %% combining spring constants for
      upper suspension total
end
if (kc2==0)
  k2=kw2;
else
  k2=kc2*kw2/(kc2+kw2);       %% combining spring constants for
      lower suspension total
end
  k3=kw3;

%*******************************************************************
```

```
% allows choice of 2 wires to set separation
% in X-direction wires must be vertical
s0=su;                          %% 1/2 separation of upper wires
s1=su;                          %% -->
s2=si;                          %% 1/2 separation of intermediate
    wires
s3=si;                          %% -->
s4=sl;                          %% 1/2 separation of lower wires
s5=sl;                          %% -->

%*********************************************************************


m13 = m1+m2+m3;                 %% combining U,I, and T masses
m23 = m2+m3;                    %% combining I and T masses

%************************************************************************************


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

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

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)

%*********************************************************

% longitudinal and pitch equations of motion
% errors in some terms corrected 6/99

k11 = -m13*g*d0/I1y-2*k1*s0^2*c1^2/I1y-m23*g*d1/I1y-2*k2*s2^2*c2/I1y
    -m23*g*d1^2/I1y/l2/c2 ...
        -m13*g*d0^2/I1y/l1/c1-m13*g*s0^2*si1^2/I1y/l1/c1-m23*g*s2^2*
            si2^2/I1y/l2/c2;
k12 = -m23*g*d1/I1y/l2/c2+m13*g*d0/I1y/l1/c1;
k13 = -m23*g*d1*d2/I1y/l2/c2+2*k2*s2^2*c2^2/I1y+m23*g*s2^2*si2^2/l2/
    c2/I1y;
k14 = +m23*g*d1/I1y/l2/c2;
k15 = 0;
k16 = 0;

k21 = +m13*g*d0/m1/l1/c1-m23*g*d1/m1/l2/c2;
k22 = -m13*g/m1/l1/c1-m23*g/m1/l2/c2;
k23 = -m23*g*d2/m1/l2/c2;
k24 = +m23*g/m1/l2/c2;
k25 = 0;
k26 = 0;

k31 = +2*k2*s2^2*c2^2/I2y-m23*g*d2*d1/I2y/l2/c2+m23*g*s2^2*si2^2/I2y
    /l2/c2;
k32 = -m23*g*d2/I2y/l2/c2 ;
k33 = -m23*g*d2/I2y-2*k2*s2^2*c2^2/I2y-m3*g*d3/I2y-2*k3*s4^2*c3^2/
    I2y-m23*g*d2^2/I2y/l2/c2 ...
        -m3*g*d3^2/I2y/l3/c3-m23*g*s2^2*si2^2/l2/I2y/c2-m3*g*s4^2*
            si3^2/l3/I2y/c3;
k34 = +m23*g*d2/I2y/l2/c2-m3*g*d3/I2y/l3/c3;
k35 = -m3*g*d3*d4/I2y/l3/c3+2*k3*s4^2*c3^2/I2y+m3*g*s4^2*si3^2/l3/
    I2y/c3;
k36 = +m3*g*d3/I2y/l3/c3;

k41 = +m23*g*d1/m2/l2/c2;
k42 = +m23*g/m2/l2/c2;
k43 = +m23*g*d2/m2/l2/c2-d3*m3*g/m2/l3/c3;
k44 = -m23*g/m2/l2/c2-m3*g/m2/l3/c3;
k45 = -m3*g*d4/m2/l3/c3;
k46 = +m3*g/m2/l3/c3;
```

```
k51 = 0;
k52 = 0;
k53 = +2*k3*s4^2*c3^2/I3y-m3*g*d4*d3/l3/I3y/c3+m3*g*s4^2*si3^2/I3y/
    l3/c3;
k54 = -m3*g*d4/I3y/l3/c3;
k55 = -m3*g*d4/I3y-2*k3*s4^2*c3^2/I3y-m3*g*d4^2/I3y/l3/c3-m3*g*s4^2*
    si3^2/I3y/l3/c3;
k56 = +m3*g*d4/I3y/l3/c3;

k61 = 0;
k62 = 0;
k63 = +g*d3/l3/c3;
k64 = +g/l3/c3;
k65 = +g*d4/l3/c3;
k66 = -g/l3/c3;

%*******************************************************
% masses in descending order (top = 1, intermediate = 2, mirror = 3)
% angle/displacement angle/displacement angle/displacement
% unphysical damping used to make open loop phase plots readable
% set b = 0 for closed loop plots

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% state space matrices pitch and longitudinal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A=[0  0 0 0 0 0 1 0 0 0 0 0
   0  0 0 0 0 0 0 1 0 0 0 0
   0  0 0 0 0 0 0 0 1 0 0 0
   0  0 0 0 0 0 0 0 0 1 0 0
   0  0 0 0 0 0 0 0 0 0 1 0
   0  0 0 0 0 0 0 0 0 0 0 1
   k11  k12 k13 k14 k15 k16 -bd 0 0 0 0 0
   k21  k22 k23 k24 k25 k26 0 -bd 0 0 0 0
   k31  k32 k33 k34 k35 k36 0 0 -bd 0 0 0
   k41  k42 k43 k44 k45 k46 0 0 0 -bd 0 0
   k51  k52 k53 k54 k55 k56 0 0 0 0 -bd 0
   k61  k62 k63 k64 k65 k66 0 0 0 0 0 -bd];

B=[0  0 0 0  0  0  -m13*g*d0/I1y/l1/c1 +m13*g/m1/l1/c1   0  0 0 0
   0  0 0 0  0  0    0  1/m1  0     0 0 0
   0  0 0 0  0  0    0  1/m1  0     0 0 0
   0    0   0   0  0  0    0     0    0    1/m2  0 0
   0  0 0 0  0  0    0     0     0 0 1/m3
   0    0   0   0  0  0  1/I1y   0    0   0 0 0
   0    0   0   0  0  0  1/I1y   0    0   0 0 0
   0    0   0   0  0  0    0     0  1/I2y 0 0 0
   0    0   0   0  0  0    0     0    0     0 1/I3y 0
]';

C=[0  1 0 0 0 0 0 0 0 0 0 0
   0  0 0 1 0 0 0 0 0 0 0 0
   0  0 0 0 0 1 0 0 0 0 0 0
   1  0 0 0 0 0 0 0 0 0 0 0
   0  0 1 0 0 0 0 0 0 0 0 0
   0  0 0 0 1 0 0 0 0 0 0 0];

D=[0  0  0  0  0  0  0  0 0
   0  0  0  0  0  0  0  0 0
   0  0  0  0  0  0  0  0 0
   0  0  0  0  0  0  0  0 0
   0  0  0  0  0  0  0  0 0
   0  0  0  0  0  0  0  0 0];

% Generate ss model for east pendulum
lpneast = ss(A,B,C,D,'statename',{'sq1e' 'sx1e' 'sq2e' 'sx2e' 'sq3e'
     'sx3e' 'sQ1e' 'sv1e' 'sQ2e' ...
                    'sv2e' 'sQ3e' 'sv3e'}, 'inputname',{'x0e' 'LLCe'
                         'F1e' 'F2e' 'F3e' ...
                    'PLCe' 'tau1e' 'tau2e' 'tau3e'},'outputname',{'
                        x1e' 'x2e' 'x3e'  ...
                    'q1e' 'q2e' 'q3e'});


% Generate ss model for north pendulum
lpnnorth = ss(A,B,C,D,'statename',{'sq1n' 'sx1n' 'sq2n' 'sx2n' 'sq3n
    ' 'sx3n' 'sQ1n' 'sv1n' 'sQ2n' ...
                    'sv2n' 'sQ3n' 'sv3n'}, 'inputname',{'x0n' 'LLCn'
                         'F1n' 'F2n' 'F3n' ...
                    'PLCn' 'tau1n' 'tau2n' 'tau3n'},'outputname',{'
                        x1n' 'x2n' 'x3n'  ...
                    'q1n' 'q2n' 'q3n'});



% Generate split after output to feed back to local control
A=[];B=[];C=[];D=[1;1];
splitbackxe = ss(A,B,C,D,'inputname',{'x1e'},'outputname',{'x1eout'
    'x1efeed'});
A=[];B=[];C=[];D=[1;1];
splitbackqe = ss(A,B,C,D,'inputname',{'q1e'},'outputname',{'q1eout'
    'q1efeed'});
A=[];B=[];C=[];D=[1;1];
splitbackxn = ss(A,B,C,D,'inputname',{'x1n'},'outputname',{'x1nout'
    'x1nfeed'});
A=[];B=[];C=[];D=[1;1];
splitbackqn = ss(A,B,C,D,'inputname',{'q1n'},'outputname',{'q1nout'
    'q1nfeed'});

% Generate local controls
localctrl;

% create single model of east pendulum connected to local controls
inputs = {'groundxe' 'F3e'};% 'groundqe'};
outputs = {'x1eout' 'x2e' 'x3e' 'q1eout' 'q2e' 'q3e'};
pendlocale = connect(lpneast,localeast,splitbackxe,splitbackqe,
    inputs,outputs);

% create single model of north pendulum connected to local controls
inputs = {'groundqn'};
outputs = {'x1nout' 'x2n' 'x3n' 'q1nout' 'q2n' 'q3n'};
pendlocaln = connect(lpnnorth,localnorth,splitbackxn,splitbackqn,
    inputs,outputs);

% create a model of the pendulum with controls and every possible
% input/output - east arm
inputs = {'groundxe' 'groundqe' 'F1e' 'F2e' 'F3e' 'tau1e' 'tau2e' '
    tau3e'};
outputs = {'x1eout' 'x2e' 'x3e' 'q1eout' 'q2e' 'q3e'};
totalpende = connect(lpneast,localeast,splitbackxe,splitbackqe,
    inputs,outputs);

% create a model of the pendulum with controls and every possible
% input/output - north arm
inputs = {'groundxn' 'groundqn' 'F1n' 'F2n' 'F3n' 'tau1n' 'tau2n' '
    tau3n'};
outputs = {'x1nout' 'x2n' 'x3n' 'q1nout' 'q2n' 'q3n'};
totalpendn = connect(lpnnorth,localnorth,splitbackxn,splitbackqn,
    inputs,outputs);
```

Appendix D

```
% modecalc.m calulates the mode frequencies of a triple pendulum in
    the
% longitudinal and tilt degrees of freedom
% written by C. Torrie, Updated and modified by A. Lombardi - July
    22, 2010

% call in pendulum values
trippendlp;

% longitudinal and tilt matrx
A_LRT       = [ k11 k12 k13 k14 k15 k16
                k21 k22 k23 k24 k25 k26
                k31 k32 k33 k34 k35 k36
                k41 k42 k43 k44 k45 k46
                k51 k52 k53 k54 k55 k56
                k61 k62 k63 k64 k65 k66];

% calculation of the mode frequencies (Hz)

p = sqrt(abs(eig(A_LRT)));
longtilt  = p/2/pi
```

References

[1] Torrie, Calum. “Development of Suspensions for the GEO600 Gravitational Wave Detector”, Ph.D. thesis, University of Glasgow, 2001.

[2] Grote, Hartmut, “Making it Work: Second Generation Interferometry in GEO600”, Ph.D. thesis, Vom Fachbereich Physik der Universitat Hannover, 2003.

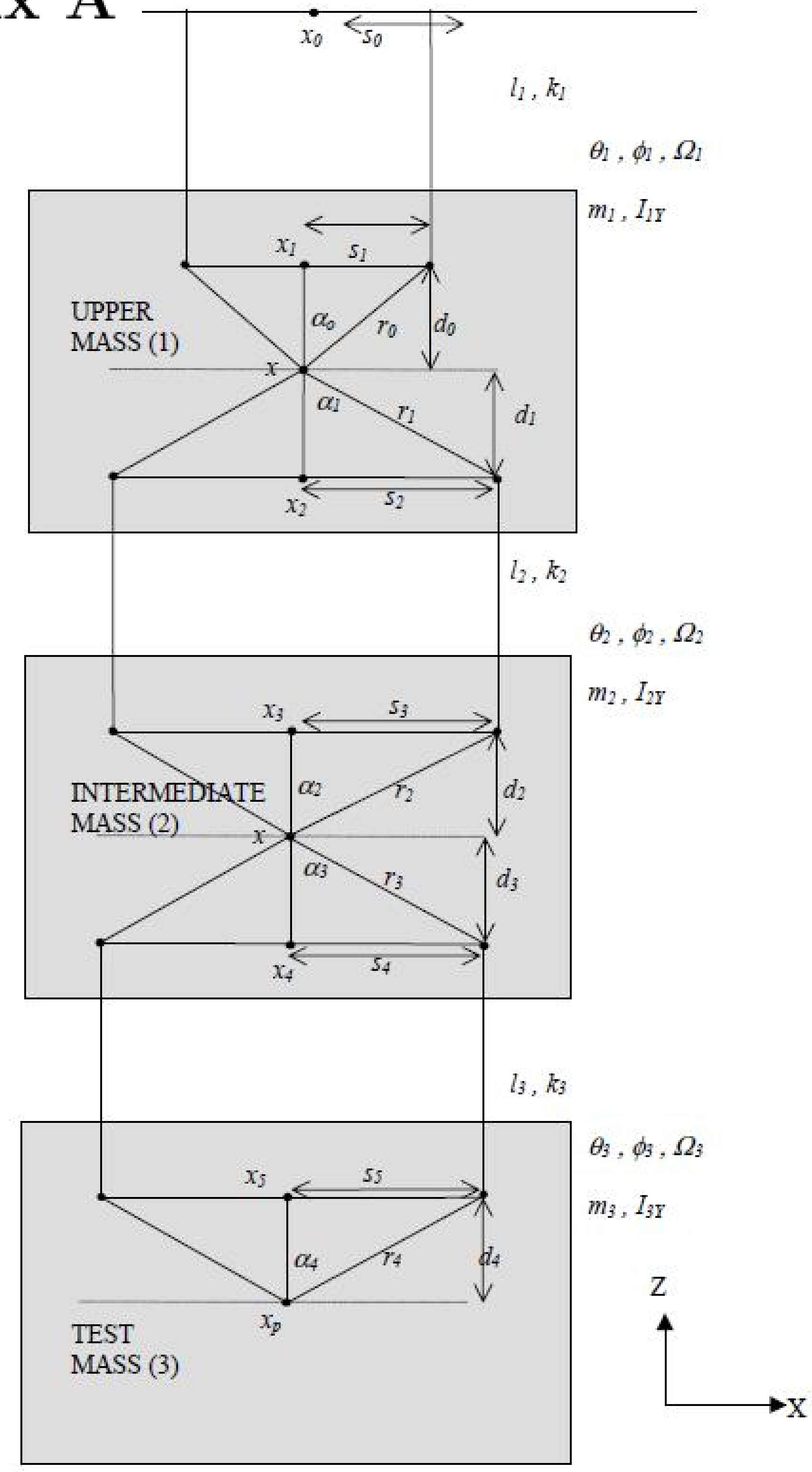
[3] Freise, Andreas, “The Next Generation of Interferometry: Multi-Frequency Optical Modelling, Control Concepts and Implementation”, Ph.D. thesis, Vom Fachbereich Physik der Universitat Hannover, 2003.

[4] Leong, Jonathan, “GEO600 Detector Characterization”, GEO Collaboration, 2010.

[5] Willke, B. *et al*, “The GEO600 Gravitational Wave Detector”, Amaldi, 2001.

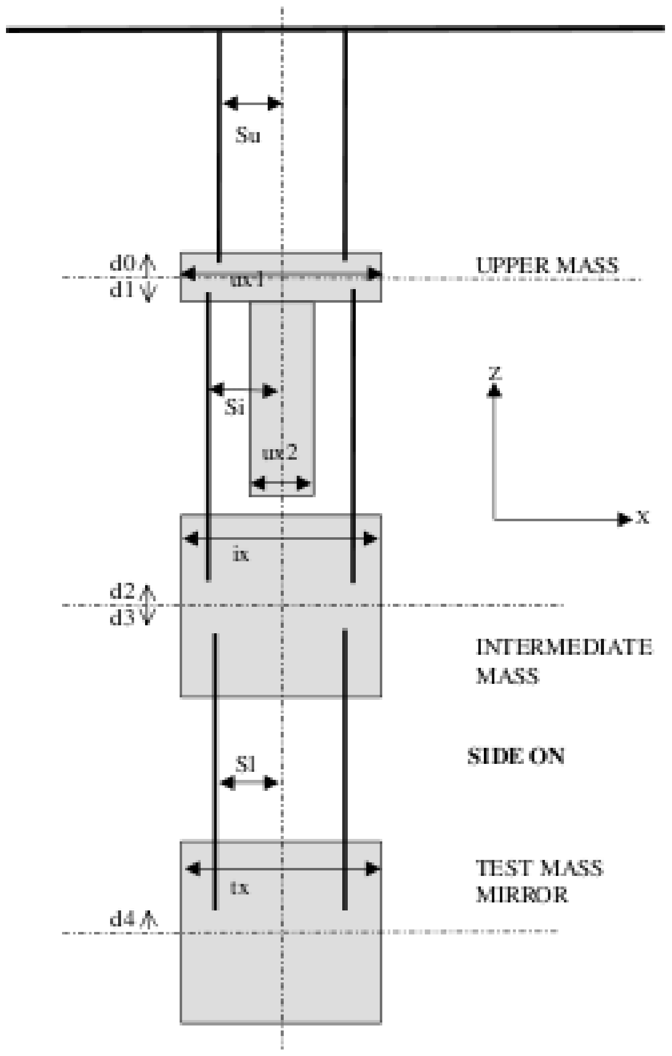
[6] Horowitz, Paul and Winfield Hill. “The Art of Electronics”. 2 edition. Cambridge University Press, July 1989.

Appendix A

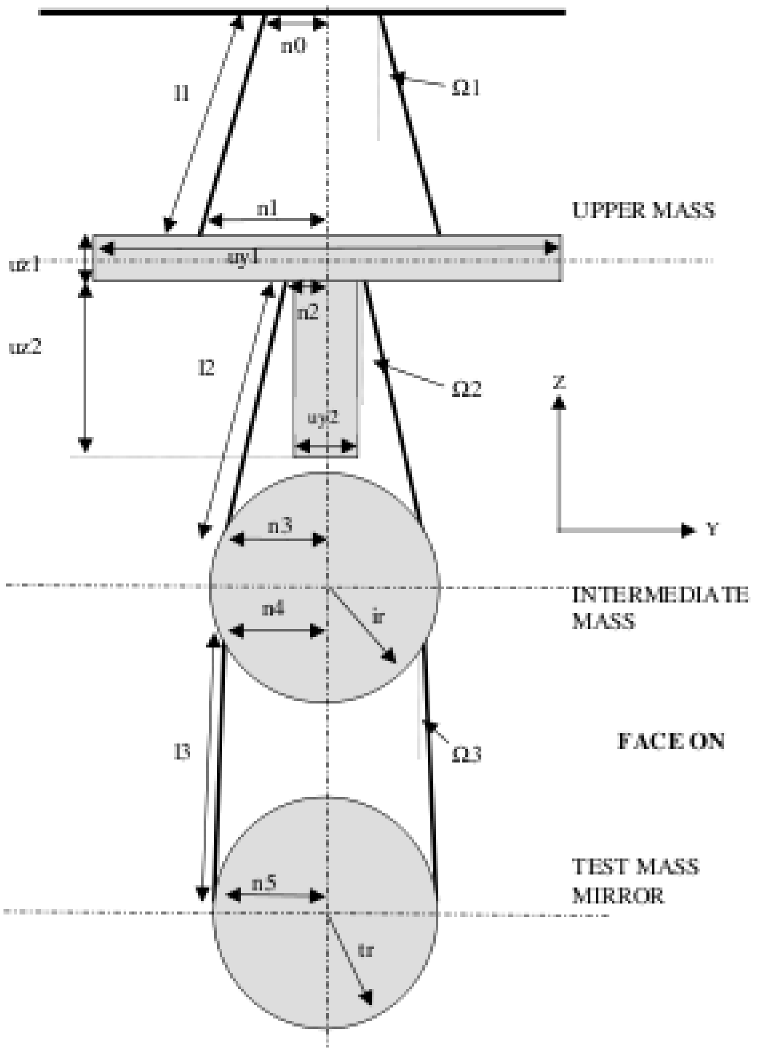

x_0
s_0
l_1 , k_1
θ_1 , ϕ_1 , Ω_1
m_1 , I_{1Y}
x_1
s_1
UPPER MASS (1)
α_o
r_0
d_0
x
α_1
r_1
d_1
x_2
s_2
l_2 , k_2
θ_2 , ϕ_2 , Ω_2
m_2 , I_{2Y}
x_3
s_3
INTERMEDIATE MASS (2)
α_2
r_2
d_2
x
α_3
r_3
d_3
x_4
s_4
l_3 , k_3
θ_3 , ϕ_3 , Ω_3
m_3 , I_{3Y}
x_5
s_5
α_4
r_4
d_4
x_p
TEST MASS (3)
Z
X

Appendix B

B.1 The parameters of a triple pendulum (side view).



B.2 The parameters for a triple pendulun (face on view).


n0
l1
Ω1
n1
UPPER MASS
uz1
uy1
n2
uz2
l2
Ω2
Z
uy2
Y
n3
INTERMEDIATE MASS
n4
ir
l3
Ω3
FACE ON
TEST MASS MIRROR
n5
tr