UNIVERSITY OF WASHINGTON
Department of Aeronautics and Astronautics

Multiple Control Systems for the Linear Cart Track Pendulum

June 11, 2003

Luke Dubord
Christopher Lum

AA/EE-449 Design of Automatic Control Systems

LAB EXPERIMENT I

Multiple Control Systems for the Linear Cart Track Pendulum

June 11, 2003

by

Group 3

Luke Dubord June 11, 2003

Christopher Lum June 11, 2003

Table of Contents

Fore Matter

I. List of Figures

II. List of Tables

III. List of Symbols

Table 1: List of symbols, descriptions and units, and numerical value if applicable

Symbol	Description	Value/Units
A	A matrix of LTCP system	
A_i	A matrix of LTCP system with integrator	
β	Angle between vertical and pendulum. Positive clockwise. Also called x_2	rad
D_l	Viscous damping coefficient of pendulum rotation	0.0022 kg m^2/s
D_c	Viscous damping coefficient between cart and track	5 kg/s
D_m	Viscous damping coefficient of motor rotation	0.0021 kg m^2/s
dz/dt	Horizontal cart velocity. Positive to the right. Also called x_3	m/s
$d\beta/dt$	Angular velocity of pendulum. Positive clockwise. Also called x_4	rad/s
f_l	Magnitude of constant force applied to cart. Positive to the right.	N
f_s	Coulomb friction and stiction force b/w cart and track	N
g	Gravitational acceleration	9.807 m/s^2
$G_z(s)$	Transfer function of z(s)/u(s) with inner compensator.	
$G_\beta(s)$	Transfer function of $\beta(s)/u(s)$	
Γ	Disturbance input distribution matrix	
i_a	Current on motor. Also called x_5	amps
J	Cost function	
J_l	Rotation inertia of pendulum about its center of mass	0.008 kg m^2
J_a	Inertia of motor, rotor, and belt	$J_m + J_r$
J_m	Inertia of DC motor	0.00072 kg m^2
J_r	Inertia of rotor and belt	0.00386 kg m^2
K	Motor constant	Nm/amp
K	Full state feedback gain matrix	
K_a	Amplifier gain	-3.6
K_i	Full state feedback with integrator gain matrix	
L	Observer gain matrix	
l_l	Distance from pivot point to center of mass of pendulum	0.445 m
L_a	Inductance of DC motor	0.0033 Henries
λ	Eigenvalue function (ie $\lambda(A)$ means eigenvalues of the matrix A).	
LQR	Linear Quadratic Regulator	
m_l	Mass of pendulum	0.324 kg
m_c	Mass of the cart	2.3 kg
N	Number of encoder lines on sensors	3600
P	Power	Watts
Q	States penalty matrix	

R	Control penalty matrix	
r	Radius of DC motor pulley	0.08 m
R	Resistance of motor and amp ($R_1 + R_A$)	1.3 Ohms
T_s	Settling time	sec
T_s	Sample time of XPCtarget	0.01 s
τ_s	Coulomb friction and stiction torque on DC motor	
V_o	Covariance matrix of sensor noise	
$\overline{v}$	Sensor noises	
W_{min}	Theoretical minimum work to erect pendulum	2.8279 J
W_o	Covariance matrix of process noise	
$\overline{w}$	Process noise	Nm
$\overline{x}$	State vector $\left(z \quad \beta \quad \dot{z} \quad \dot{\beta} \quad i_a\right)^T$	
$\overline{x}_i$	State vector $\left(z \quad \beta \quad \dot{z} \quad \dot{\beta} \quad i_a \quad x_6\right)^T$	
z	Horizontal cart location. Positive to the right. Also called x_1	m

Introductory Matter

IV. Experiment Objectives

The fundamental objective of this series of experiments and simulations is to develop a theoretical and experimental background in controls by using a series of different control methodologies on a common experimental apparatus. This begins by developing the state space model from an analysis of the physical system and developing a mathematical model that accurately describes the dynamics of the system. In order for this to be achieved, a number of physical properties in the system must be determined. While many of these are trivial to determine, dimensions and weights for example, others, such as friction, are much more elusive. The state space model will never be completely accurate due to unmodeled system dynamics and changes in the physical system in time. The later property was particularly noticed during the course of the quarter as the physical system endured a large amount of ware. It is necessary, however, to have a model that is sufficient to use for design analysis. Thus this is the initial step.

Once a model of acceptable accuracy is developed, this model can be used to develop control algorithms. The system, in this case, is an inverted pendulum that is attached to a cart that is able to translate in one dimension. The fundamental goal of the controller is to stabilize the pendulum in the upright position while also controlling the position of the cart. It is strongly desired that the control system have both good performance and good robustness. A series of control methodologies was used to achieve this goal.

During the course of the laboratory, a series of increasingly more applicable control methodologies were used. Each was introduced to help address some of the problems and issues that were observed in the previous control design.

Initially, a classical control design was used. This was implemented by creating an inner loop that used the angular position of the pendulum as feedback for an inner loop that controlled the position of the pendulum and an outer loop that used the position of the cart as feed back for an outer loop that controlled the cart's position. Using this method was tedious and unintuitive; placing the eigenvalues of the system was more of an art and achieving the desired performance was very difficult. Additionally, there was no guarantee of optimality and making quick changes to the control system was difficult.

To address some of these issues, a full state feedback system was implemented. Full state feedback provides excellent performance and is much quicker to implement than the classical control due to the use of functions in *Matlab*. The state of the current was impossible to determine and this pole was left alone and the gain on this state set to zero. This system was unable, however, to reject a constant force applied to the cart, and thus the next step was to add an integrator to the full state feedback to add this robustness.

While full state have very good performance with good robustness, it was based on a system of pole placement. In this method the poles are picked apriori by the designer. The gains that needed to be applied to each state were then determined using *Matlab*. This lacked optimality and thus the next step was to use an LQR design to bring this characteristic.

Both the Full State Feedback and the LQR designs, however, assume that all of the states are available for feedback. In the case of the Linear Cart Inverted Pendulum this is in fact not the case. The only states that are directly available through sensors are the angular position of the pendulum and the position of the cart. Thus all of the other states must be either estimated or ignored. An observed design was then developed to estimate the unobserved states. This was the final design that was implemented.

As a complementary project throughout a pump up control algorithm was developed to bring the pendulum to an upright position from its resting state of straight down. Unlike many of the common algorithms, the one developed uses feedback of the angular velocity to add energy to the pendulum at times when it is most effective. Thus the pump up controller is robust to pendulums of different weights and sizes.

V. Experiment Apparatus

All of the physical experiments that were completed in the course of this experiment were conducted on the Linear Cart Track Pendulum (LCTP) in the Controls Laboratory at the University of Washington. The Controls Laboratory is located in room 211 of Guggenheim Hall.

The LCTP, shown below in Figure 1, consists of a cart that is able to move one dimensionally along a rail. A pillow block is used to reduce the friction between the cart and the rail. The position of the cart can be manipulated by use of a large belt that is connected to the cart and tensioned through two pulleys. One pulley is located at either end of the rail.

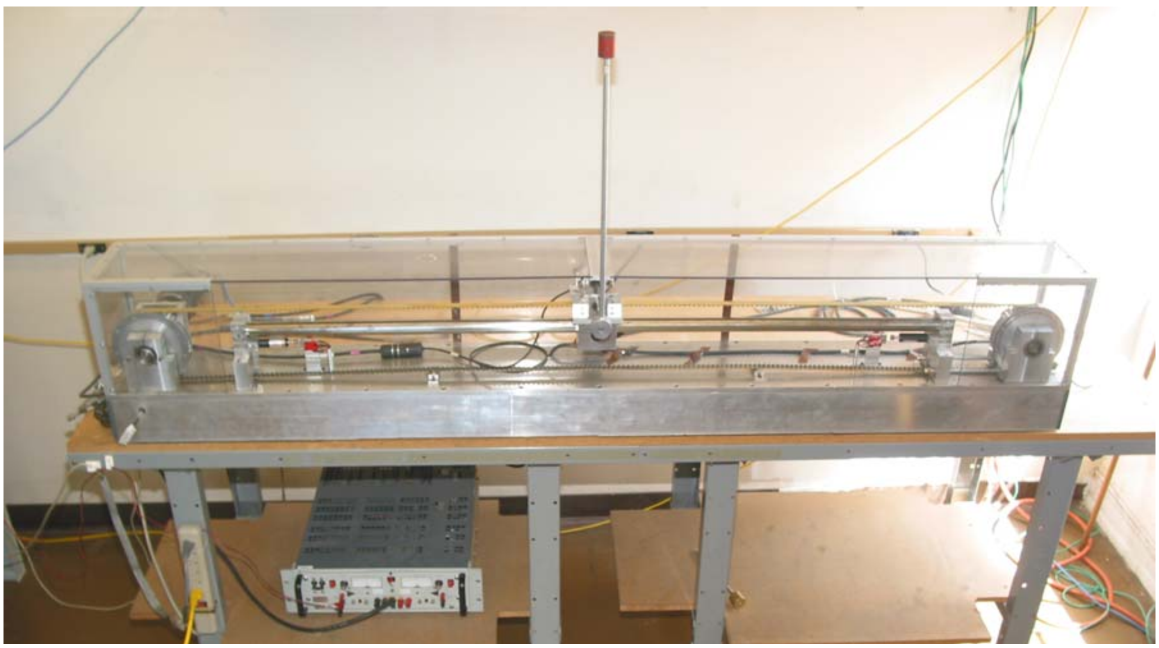


Figure 1: The Linear Cart Track Pendulum

Attached to the pulley on the left in Figure 1 an electric motor that is controlled via the XPC system. The signal from the computer is outputted via a digital to analog board and is imputed to a Kepco inverting voltage amplifier. The Kepco is evident in the bottom left corner of Figure 1. The amplifier had a gain of 3.6 and inverted the incoming signal. A more detailed picture of the electric motor and its connection to the pulley is shown below in Figure 2 and the face of the Kepco is shown below in Figure 3.

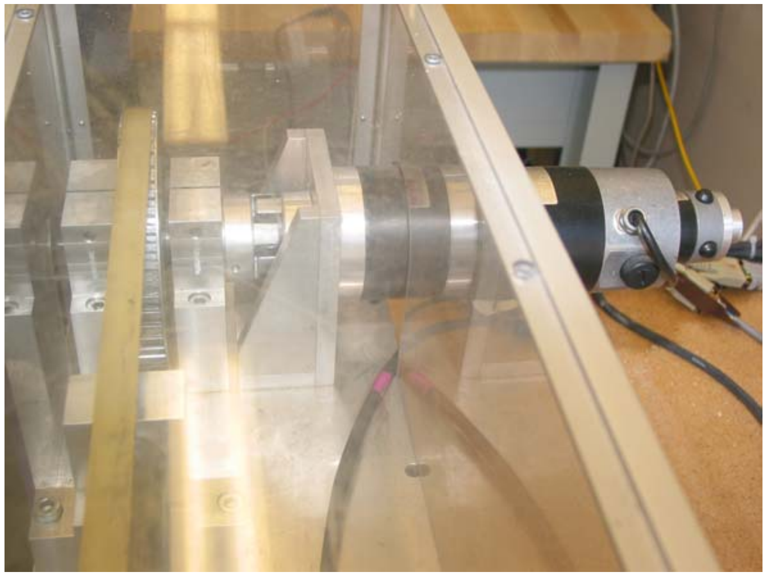


Figure 2: The Alsthom electric motor that changes the Z position of the cart and its connection to the belt.

Figure 3: The Kepco Inverting Voltage Amplifier.

Attached to the cart was a single pendulum. This is shown below in Figure 4. The pendulum was allowed to rotate freely through 360° through the use of a pair of bearings. The position of the pendulum was determined by an Alsthom optical quadrature encoder with a resolution of 3600 steps per revolution. Another Alsthom encoder of the same specifications but of a different type was used to measure the position of the cart. This was done on the pulley opposite the motor. The arrangement of the z measuring encoder is shown below in Figure 5. Because the size of the pulley was known and the pulley-belt mechanism was designed so that the belt was unable to slip with respect to the pulley, the position of the cart in the Z direction could be determined using this encoder.

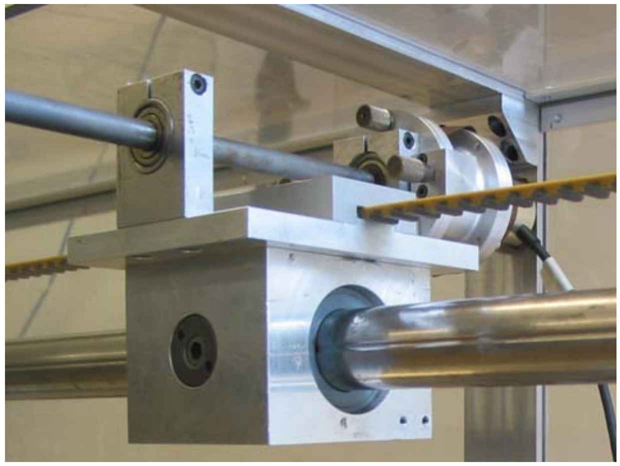


Figure 4: The cart showing the bearings and the pendulum encoder at the rear.

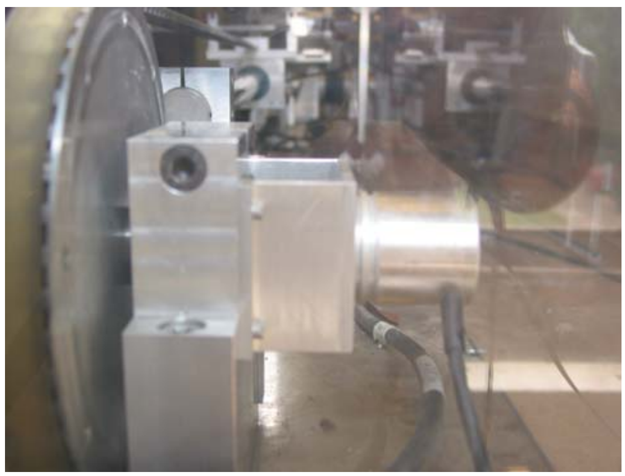


Figure 5: The Z position encoder and mounting.

Thus the LCTP system represents a system that is under-actuated as the angular position of the pendulum can only be controlled by moving the cart and cannot be controller directly. It also represents a system that is nonlinear around the equilibrium point that exists when the pendulum is pointing straight up. This is an equilibrium point that is also unstable.

One final piece of physical apparatus that was used was a push – pull gage that was provided by the University of Washington Aeronautical Laboratory. This push pull device measured the force applied in pounds force and had a resolution of 0.2 lbf. It was used in all of the experimental force rejection experiments and is shown below in Figure 6.

Figure 6: The push - pull gage used to apply an external force to the cart.

System Modeling

VI. Nonlinear Dynamics

In order to begin designing a control system for the inverted pendulum, it is first necessary to derive the mathematical model for the physical system. The resulting equations describing the dynamics of the systems are highly nonlinear and coupled. The appropriate state vector to choose is given by Equation 1. This is the state vector corresponding to the system operating in voltage control mode. This means that the voltage to the amplifier is directly controlled as opposed to controlling the current.

$$\overline{x} = \begin{pmatrix} z & \beta & \dot{z} & \dot{\beta} & i_a \end{pmatrix}^T$$

Equation 1

Using these states, the non-linear equations describing the evolution of the system is given by Equation 2.

$$\dot{x}_1 = x_3$$

$$\dot{x}_2 = x_4$$

$$\dot{x}_3 = \frac{-\left(m_1 l_1^2 + J_1\right)}{\det\left(M\right)}\left(D_C x_3 - m_1 l_1 \sin\left(x_2\right) x_4^2 - \frac{K}{r} x_5 + \frac{D_m}{r^2} x_3 - \left(f_s + \frac{\tau_s}{r}\right) - f_1\right) + \frac{m_1 l_1 \cos\left(x_2\right)}{\det\left(M\right)}\left(D_1 x_4 - m_1 l_1 g \sin\left(x_2\right)\right)$$

$$\dot{x}_4 = \frac{m_1 l_1 \cos\left(x_2\right)}{\det\left(M\right)}\left(D_C x_3 - m_1 l_1 \sin\left(x_2\right) x_4^2 - \frac{K}{r} x_5 + \frac{D_m}{r^2} x_3 - \left(f_s + \frac{\tau_s}{r}\right) - f_1\right) - \frac{\left(m_c + m_1 + \frac{J_a}{r^2}\right)}{\det\left(M\right)}\left(D_1 x_4 - m_1 l_1 g \sin\left(x_2\right)\right)$$

$$\dot{x}_5 = \frac{-\left(R_a + R_1\right)}{L_a} x_5 - \frac{K}{L_a r} x_3 + \frac{K_a}{L_a} u$$

Equation 2[1]

where $\det\left(M\right) = \left(J_1 + m_1 l_1^2\right)\left(m_1 + m_c + \frac{J_a}{r^2}\right) - l_1^2 m_1^2 \cos^2\left(x_2\right)$

$$\left(f_s + \frac{\tau_s}{r}\right) = f_{coulomb} + f_{stiction} = -f_c sign\left(x_3\right) - \frac{K i_{st}}{r} e^{\frac{-|x_3|}{\sigma}} sign\left(x_3\right)$$

[1] For complete derivation of these equations, see Appendix A: Non-Linear State Space Representation

Friction Modeling

In Equation 2, all of the necessary parameters have already been derived and provided in the ltcpsp2k file. However, the term $f_s + \tau_s/r$ which represents the coulomb friction and stiction forces on the system is not accounted for. This is an important factor in the system as it leads to dead zones and limit cycles. The viscous friction is a linear phenomenon (is directly proportional to x_3) and is taken into account by the D_C and D_m terms. Therefore, it becomes necessary to evaluate the effect of the coulomb friction and stiction forces. By fixing the pendulum such that it cannot rotate[2], driving the system with a constant control, and allowing the system to reach steady state, the friction terms can be written as

$$\left[f_s + \frac{\tau_s}{r} - \left(D_c + \frac{D_m}{r} \right) x_3 \right] = -\frac{K\left(K_a u - \frac{K x_3}{r} \right)}{r\left(R_a + R_1 \right)}$$

Equation 3[3]

The only unknowns in Equation 3 is x_3. Therefore, by applying multiple different control voltages, u, and recording the resulting steady state velocity. The friction as a function of x_3 curve can be determined. The response of x_3 vs. time can be plotted for each run and the place where the velocity becomes constant is the steady state value. An example is shown below in Figure 7.

[2] See Experimental Procedure section for more information
[3] See Appendix A: Empirically Evaluating $f_s + \tau_s/r$ for complete derivation

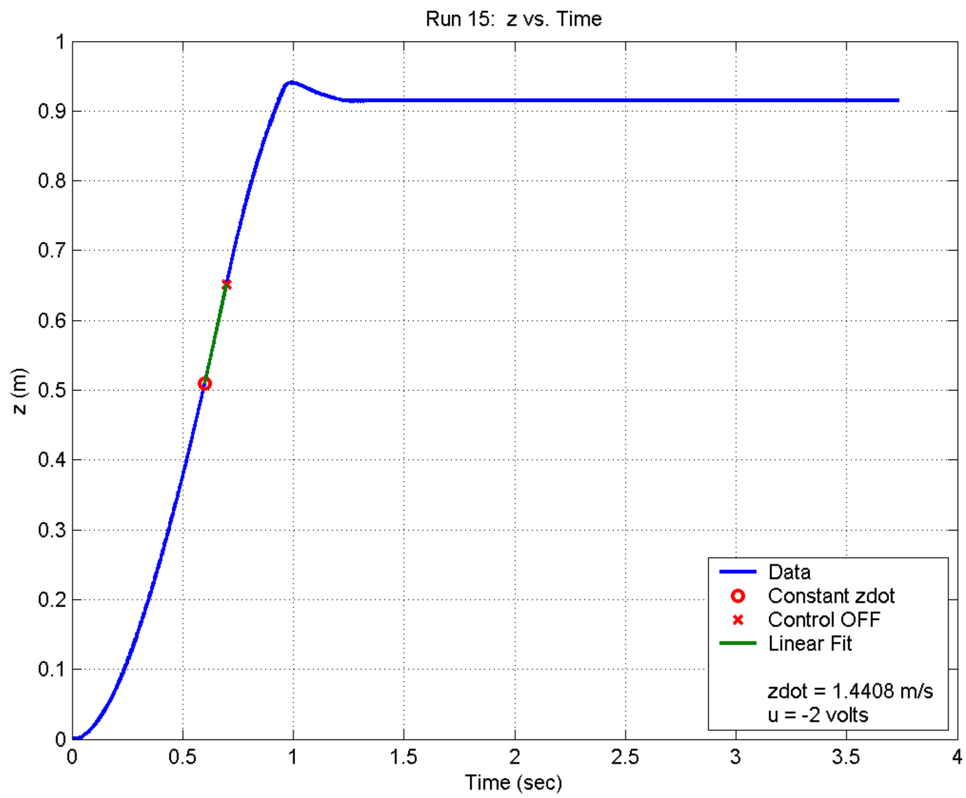


Figure 7: x_3 vs. time for friction evaluate run

As can be seen, this procedure is repeated for many different control voltages. The data is shown below in Table 2.

Table 2: Friction model control and x_3

Run Number	**u (volts)**	**x_3 (m/s)**
1	-0.25	0.00
2	-0.50	0.00
3	-0.60	0.09
4	-0.65	0.13
5	-0.70	0.18
6	-0.75	0.21
7	-0.80	0.27
8	-0.85	0.33
9	-0.90	0.39
10	-0.95	0.44
11	-1.00	0.51
12	-1.25	0.76
13	-1.50	1.02
14	-1.75	1.24
15	-2.00	1.44
16	-2.25	1.65
17	-2.50	1.79
18	-2.75	1.85
19	0.25	0.00
20	0.50	0.00
21	0.55	0.00
22	0.60	0.00
23	0.65	-0.11
24	0.70	-0.16
25	0.75	-0.20
26	0.80	-0.26
27	0.85	-0.33
28	0.90	-0.28
29	0.95	-0.37
30	1.00	-0.45
31	1.25	-0.75
32	1.50	-0.98
33	1.75	-1.17
34	2.00	-1.34
35	2.25	-1.58
36	2.50	-1.72
37	2.75	-1.79

Given this information, the friction force as a function of x_3 can be calculated using Equation 3. The results are shown below in Figure 8.

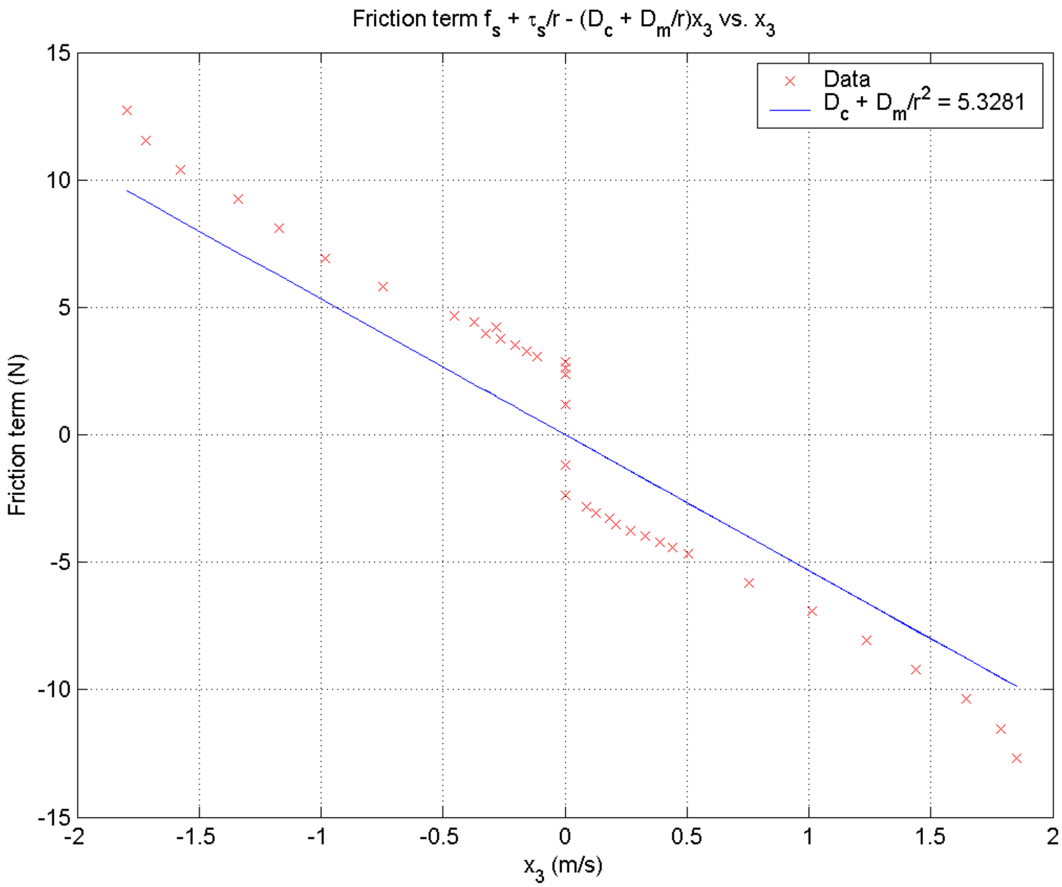


Figure 8: Friction terms as a function of x_3 also showing only viscous friction

Figure 8 shows the collected data along with the predicted friction force if only viscous friction is taken into account using the coefficients published in the ltcp1sp2k file. As can be seen, it appears that the viscous friction coefficients are fairly accurate. The other interesting observation to make is that the stiction force appears to be negligible and that the total friction appears to be the combination of the viscous friction and the constant Coulomb friction. From Figure 8, the magnitude of the coulomb friction is given by

$$f_s + \frac{\tau_s}{r} = -1.1 sign(x_3)$$

VII. Linearization

As can be seen by inspecting Equation 2, the equations describing the evolution of the system are highly non-linear. In order to apply classical design techniques, a linear model is required. Therefore, the system needs to be linearized about an equilibrium point.

From geometry, there are two obvious equilibrium points. The system is at equilibrium when the pendulum is in the up position ($x_2 = 0$), and when the pendulum is in the down position ($x_2 = \pi$). Also note that x_1 does not appear anywhere in the equations of motion shown in Equation 2. Therefore, x_1 can take on any value at the equilibrium point. This makes sense considering that the pendulum should be able to be balanced regardless of the z location. Obviously, the equilibrium control is simply u = 0.

$$\bar{x}_o = \begin{pmatrix} z_o & \beta_o & 0 & 0 & 0 \end{pmatrix}$$
$$u_o = 0$$

Equation 4

where $\beta_o = 0$ or π

The system can now be linearized about these equilibrium points.

$$\frac{d}{dt}\bar{x}\left(\bar{x},u\right) = \frac{d}{dt}\bar{x}\left(\bar{x}_o,u_o\right) + A\left(\bar{x}-\bar{x}_o\right) + B\left(u-u_o\right) + Higher\ Order\ Terms$$

Equation 5

where $$A = \left.\begin{pmatrix} \frac{df_1}{dx_1} & \frac{df_1}{dx_2} & \cdots & \frac{df_1}{dx_n} \\ \frac{df_2}{dx_1} & \cdot & & \vdots \\ \vdots & & \cdot & \\ \frac{df_n}{dx_1} & \cdots & & \frac{df_n}{dx_n} \end{pmatrix}\right|_{\bar{x}=\bar{x}_O}$$

$$B = \left.\begin{pmatrix} \dfrac{df_1}{du} \\ \dfrac{df_2}{du} \\ \vdots \\ \dfrac{df_n}{du} \end{pmatrix}\right|_{u=u_O}$$

However, due to the complex nature of the expressions shown above in Equation 2, taking the derivative df_i/dx_j may be extremely difficult. Therefore, it is easier to make a change of variables so that many of the complex terms will be simplified. Once this occurs df_i/dx_j will be significantly easier to compute.

For linear operation, the states will not vary much from the equilibrium values. Namely

$$\begin{aligned} \overline{x}(t) &= \overline{x}_o + \overline{\delta} x \\ u(t) &= u_o + \delta u(t) \end{aligned}$$

Equation 6

Equation 6 can be substituted into Equation 2, which describes the system near the equilibrium point. Immediately, this expression can be made simpler by neglecting higher order terms ($\delta x_4^2 \approx 0$) and coulomb friction and stiction forces ($f_s \approx 0$) and also applying no constant force ($f_l = 0$). Furthermore, for small angles,

$$\begin{aligned} \cos(\beta_o + \delta x_2) &\approx \cos(\beta_o) - \sin(\beta_o)\delta x_2 \\ \sin(\beta_o + \delta x_2) &\approx \sin(\beta_o) + \cos(\beta_o)\delta x_2 \end{aligned}$$

Equation 7

Furthermore, since β_o can only take on values of 0 or π, then $\sin(\beta_o) = 0$. Therefore,

$$\begin{aligned} \cos(\beta_o + \delta x_2) &\approx \cos(\beta_o) \\ \sin(\beta_o + \delta x_2) &\approx \cos(\beta_o)\delta x_2 \end{aligned}$$

Equation 8

Using these simplifications, the linearized equation of motion are given by

$$\dot{x}_1 = \delta x_3$$

$$\dot{x}_2 = \delta x_4$$

$$\dot{x}_3 \approx \frac{-\left(m_1 l_1^2 + J_1\right)}{\det\left(M_{linear}\right)}\left[\left(D_C + \frac{D_m}{r^2}\right)\delta x_3 - \frac{K}{r}\delta x_5\right] + \frac{m_1 l_1 \cos\left(\beta_o\right)}{\det\left(M_{linear}\right)}\left(D_1 \delta x_4 - m_1 l_1 g \cos\left(\beta_o\right)\delta x_2\right)$$

$$\dot{x}_4 \approx \frac{m_1 l_1 \cos\left(\beta_o\right)}{\det\left(M_{linear}\right)}\left[\left(D_C + \frac{D_m}{r^2}\right)\delta x_3 - \frac{K}{r}\delta x_5\right] - \frac{\left(m_c + m_1 + \frac{J_a}{r^2}\right)}{\det\left(M_{linear}\right)}\left(D_1 \delta x_4 - m_1 l_1 g \cos\left(\beta_o\right)\delta x_2\right)$$

$$\dot{x}_5 = \frac{-\left(R_a + R_1\right)}{L_a}\delta x_5 - \frac{K}{L_a r}\delta x_3 + \frac{K_a}{L_a}\delta u$$

Equation 9

where $\det\left(M_{linear}\right) = \left(J_1 + m_1 l_1^2\right)\left(m_1 + m_c + \frac{J_a}{r^2}\right) - l_1^2 m_1^2 \cos^2\left(\beta_o\right)$

As can be seen, Equation 9, is a significantly simpler set of non-linear equation that can now be linearized using Equation 5. The partial fractions, df_i/dx_j can now be easily calculated. The state space representation of the system is given by

$$\frac{d}{dt}\overline{x} = A\delta\overline{x} + B\delta u$$

$$\overline{y} = C\delta\overline{x} + D\delta u$$

Equation 10

where

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \dfrac{-g l_1^2 m_1^2 \cos^2(\beta_o)}{\det(M_{linear})} & \dfrac{-\left(D_C + \dfrac{D_m}{r^2}\right)\left(J_1 + l_1^2 m_1\right)}{\det(M_{linear})} & \dfrac{D_1 l_1 m_1 \cos(\beta_o)}{\det(M_{linear})} & \dfrac{\dfrac{K}{r}\left(J_1 + l_1^2 m_1\right)}{\det(M_{linear})} \\ 0 & \dfrac{g l_1 m_1 \left(m_1 + m_c + \dfrac{J_a}{r^2}\right)\cos(\beta_o)}{\det(M_{linear})} & \dfrac{\left(D_C + \dfrac{D_m}{r^2}\right) l_1 m_1 \cos(\beta_o)}{\det(M_{linear})} & \dfrac{-D_1 \left(m_1 + m_c + \dfrac{J_a}{r^2}\right)}{\det(M_{linear})} & \dfrac{-\dfrac{K}{r} l_1 m_1 \cos(\beta_o)}{\det(M_{linear})} \\ 0 & 0 & -\dfrac{K}{L_a r} & 0 & \dfrac{-(R_a + R_1)}{L_a} \end{pmatrix}$$

$$B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dfrac{K_a}{L_a} \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Equation 10 gives the completely linear state space representation of the system. The C matrix shown above replicates the actual lab situation where only measurements of x_1 and x_2 are available. However, for design and testing, the C matrix can be changed to be the identity matrix such that all the states are available for monitoring.

The A matrix can be determined for either equilibrium point; up ($\beta_o = 0$) or down ($\beta_o = \pi$). The eigenvalues of the A matrix yields the location of the open loop poles, which can be used to design a control system for stabilizing the system. This can be done in *Matlab* by creating a script file to calculate the A matrix using the numerical values of each parameter as shown in Table 1: List of symbols, descriptions and units, and numerical value if applicable. The result is shown below

```
>> A = CalculateAMatrix(0)    %0 is for up position

A =

         0         0    1.0000         0         0
         0         0         0    1.0000         0
         0   -0.9258   -1.7460    0.0014    0.5612
         0   21.4448    3.4887   -0.0334   -1.1213
         0         0 -518.9394         0 -393.9394
```

To ensure that the linearization by hand was performed correctly, the results can be checked using the linmod feature on the non-linear S-Function which contains the full non-linear equations of motion. The only caveat is that the coulomb friction must be set to zero in order to avoid large gradients near the equilibrium point. The necessary *Simulink* model is shown below in Figure 9.

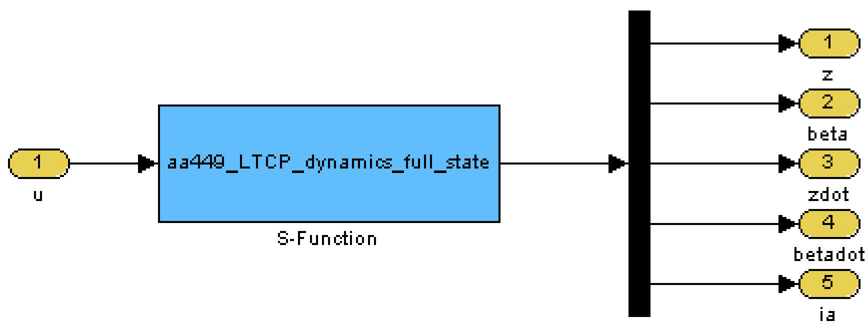


Figure 9: *Simulink* model of plant only used for linearization and deriving A matrix

This model can now be linearized using the linmod function as shown below.

```
>> [A,B,C,D] = linmod('aa449_plant_only')

A =

         0         0    1.0000         0         0
         0         0         0    1.0000         0
         0   -0.9258   -1.7460    0.0014    0.5612
         0   21.4448    3.4887   -0.0334   -1.1213
         0         0 -518.9394         0 -393.9394


B =

  1.0e+003 *

         0
         0
         0
         0
   -1.0909
```

The linmod function also calculates the C and D matrices, but from inspection, it can be seen that the C matrix is simply the identity matrix and the D matrix is zero.

As can be seen, the results from performing the linearization by hand (Equation 9), and using the linmod function yield the same results. The results from the linmod operation can be used to create a state space block of the plant that is completely linear and can be used for design and testing in later stages.

Similarity Transformation

Equation 10 is only one of the many possible state space representations of the system. The state vector can be transformed into the modal state vector using the similarity transformation. The transformation matrix is given by

$$T = \begin{pmatrix} \overline{V}_1 & \overline{V}_2 & \overline{V}_3 & \overline{V}_4 & \overline{V}_5 \end{pmatrix}$$

Equation 11

where $\overline{V}_n$ = eigenvector of eigenvalues λ_n

Using Equation 11, the state space representation given in Equation 10 can be transformed into the modal coordinates using

$$\frac{d}{dt}\overline{\zeta} = A_{\text{modal}}\delta\overline{\zeta} + B_{\text{modal}}\delta u$$

$$\overline{y} = C_{\text{modal}}\delta\overline{\zeta} + D\delta u$$

Equation 12

where $A_{modal} = T^{-1}AT$
$B_{modal} = T^{-1}B$
$C_{modal} = C\ T$

This transformation is useful for checking controllability and observability as will be shown later in the Control System Design section.

Model Reduction

Equation 10 is a 5th order system since the dimension of A is 5x5. However, it is known that the dynamics of the current (x_5) is very fast when compared to the dynamics of the rest of the system.[4] Therefore, it can be considered that the current can actually be directly controlled. In this situation, the system is said to operate in current control mode. If the current is directly controlled, the system reduces to a 4th order system. This can be shown by making the assumption that $\dot{x}_5 \approx 0$. By partitioning the state vector into two categories of slow (x_1, x_2, x_3, and x_4) and fast (x_5) states and using this assumption, the state space representation of the linearized, reduced system is given by

$$\frac{d}{dt}\overline{x} = A_{reduced}\delta\overline{x} + B_{reduced}\delta u$$

$$\overline{x} = C_{reduced}\delta\overline{x} + D_{reduced}\delta u$$

Equation 13

where
$$\begin{aligned}
A_{reduced} &= A_{11} - A_{12}A_{22}^{-1}A_{21} \\
B_{reduced} &= B_s - A_{12}A_{22}^{-1}B_f \\
C_{reduced} &= C_s - C_f A_{22}^{-1}A_{21} \\
D_{reduced} &= D - C_f A_{22}^{-1}B_f
\end{aligned}$$

[4] See Control System Design section for numerical validation of this fact.

VIII. Control System Design

Classical Control Design

The basis of the classical control design is the concept of the root locus. The root locus is defined for the linear model only. Since the goal is to stabilize the pendulum in the up position, the A-matrix in Equation 10 can be linearized with $\beta_o = 0$. The eigenvalues of this linearized matrix yield the open loop poles of the LTCP system with the pendulum in the up position. The linearized A matrix was already calculated in the System Modeling section and is repeated here for convenience.

```
A =

         0         0    1.0000         0         0
         0         0         0    1.0000         0
         0   -0.9258   -1.7460    0.0014    0.5612
         0   21.4448    3.4887   -0.0334   -1.1213
         0         0 -518.9394         0 -393.9394
```

The location of the open loop poles can easily be calculate by finding the eigenvalues of this a matrix using the damp command. This is shown below.

```
>> damp(A)

 Eigenvalue      Damping     Freq. (rad/s)

  4.54e+000    -1.00e+000      4.54e+000
  0.00e+000    -1.00e+000      0.00e+000
 -2.21e+000     1.00e+000      2.21e+000
 -4.86e+000     1.00e+000      4.86e+000
 -3.93e+002     1.00e+000      3.93e+002
```

As can be seen, the poles of the open loop system are unstable since there is a pole in the right half plane. Therefore, a compensator must be designed to stabilize the system.

For the classical control design, two loop compensators will be designed. The first task is to design an inner loop compensator that will stabilize the β loop. This corresponds to the block diagram shown below in Figure 10.

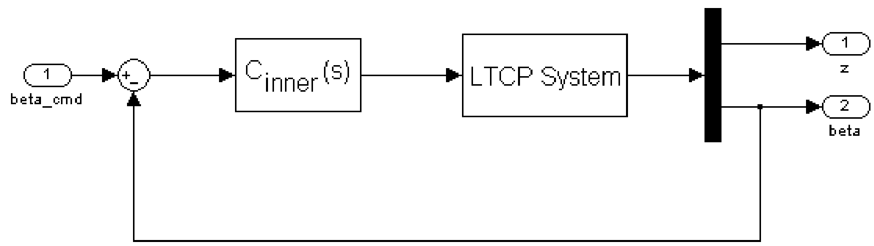


Figure 10: Inner loop for classical control design

In order to design a classical controller, the transfer function between β and u is required. The controllability and observability of this transfer function can be checked using the ctrb and obsv function in *Matlab* as shown below. The rank should be equal to the number of states if the system is fully controllable and observable.

```
>> rank(ctrb(A,B))

ans =

     5

>> rank(obsv(A,C(2,:)))

ans =

     4
```

As can be seen, the system is fully controllable since the rank of the controllability matrix is equal to the number of states. However, the rank of the observability matrix is not 5, therefore, one of the states is unobservable. This is verified by looking at the transfer function between β and u.

$$G(s)=\frac{\beta(s)}{u(s)}=\frac{1223s}{s^4+395.7s^3+970.8s^2-8452s+19180}$$

Equation 14

As can be seen in Equation 14, the characteristic equation is only 4^{th} order whereas there are 5 states. Therefore, one of the states is unobservable due to pole/zero cancellation. In order to determine which of the states is unobservable, the similarity transformation given by Equation 12 must be used. The Matlab results for A_{modal}, B_{modal}, and C_{modal} are shown below.

```
K>> Amodal

Amodal =

         0         0         0         0         0
         0    -393.2         0         0         0
         0         0    4.5437         0         0
         0         0         0   -4.8555         0
         0         0         0         0   -2.2116



K>> Bmodal

Bmodal =

   -0.6253
     -1093
    -1.025
    3.0782
    2.4181



K>> Cmodal

Cmodal =

         1  3.6461e-006   0.0061875    0.016269     0.24216
         0 -7.2867e-006    -0.21471    -0.19993     0.16021
```

As can be seen, the similarity transformation is useful since it diagonalizes the A-matrix, which allows controllability to be checked directly. Since the B_{modal} matrix is fully populated, the single control, δu, is able to affect all of the modal states. Therefore, the system is fully controllable; this confirms the result where the rank of the controllability matrix is equal to the number of states.

From the C_{modal} matrix, it can be seen why the system is not fully observable. The output is given by the product of the C_{modal} matrix and the modal state vector. The β measurement contains no information about the first modal state, ζ_1 since the first entry is zero. Since the definition of the eigenvectors is that $T^{-1}AT$ yields a diagonal matrix with the eigenvalues of A in the diagonals, it can be seen that ζ_1 corresponds with the pole at s = 0. The states that the pole at the origin is associated with can be found by looking at the eigenvectors. The matrix of eigenvectors (T) is shown below.

```
T =

     1  3.6461e-006    0.0061875     0.016269      0.24216
     0 -7.2867e-006     -0.21471     -0.19993      0.16021
     0   -0.0014336     0.028114    -0.078993     -0.53555
     0    0.0028651     -0.97557      0.97078     -0.35431
     0      0.99999    -0.036613      0.10536      0.70947
```

By looking at the A_{modal} matrix, the value $\lambda_1 = 0$ is in the first column, therefore the eigenvector associated with the eigenvalues $\lambda_1 = 0$ is given by $V_1 = (1\ 0\ 0\ 0\ 0)^T$. This shows that this pole is *only* associated with the state $x_1 = z$. This shows that with the measurement of only β, z is not observable.

The other interesting observation to make here is that there is a very fast pole located at s = -393. As can be seen by comparing it with the other poles, this pole is extremely fast and therefore, can be safely ignored. By looking at the A_{modal} matrix, it is seen that the pole at s = -393 is the second column, so the eigenvector that this is associated with it is the second column of the T matrix which is almost entirely dependent on $x_5 = i_a$ with very weak coupling with the other states. Therefore, Equation 13 can be used to reduce the order of the model by assuming that $dx_5/dt \approx 0$. The reduced A matrix is shown below

```
A_reduced =

         0         0    1.0000         0
         0         0         0    1.0000
         0   -0.9258   -2.4853    0.0014
         0   21.4448    4.9658   -0.0334
```

To ensure that only the fast pole is removed, the eigenvalues are easily calculated.

```
>> damp(A_reduced)

 Eigenvalue      Damping     Freq. (rad/s)

  4.54e+000    -1.00e+000      4.54e+000
  0.00e+000    -1.00e+000      0.00e+000
 -2.21e+000     1.00e+000      2.21e+000
 -4.85e+000     1.00e+000      4.85e+000
```

As can be seen, only the fast pole at s = -393 was removed. The resulting transfer functions between the two outputs and the inputs should remain relatively unchanged by this removal of the pole. The bode plots are shown below in Figure 11 and Figure 12.

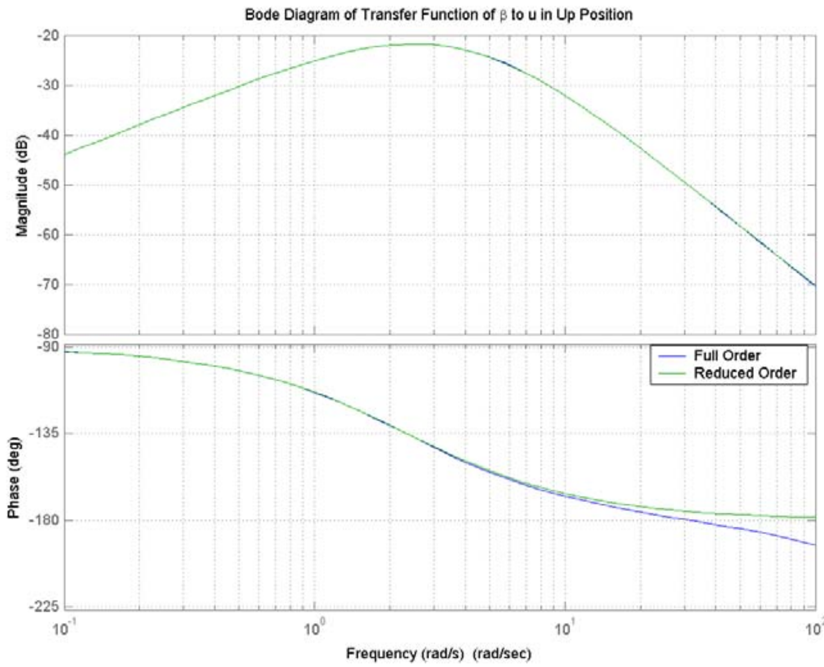


Figure 11: Bode plot between β and u for full order and reduced systems

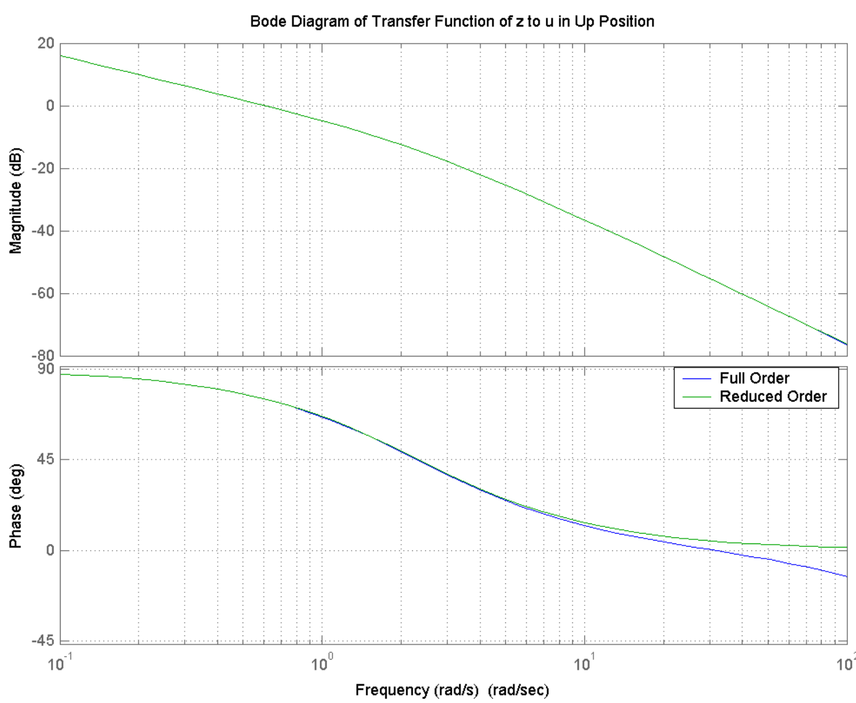


Figure 12: Bode plot between z and u for full order and reduced systems

As can be seen from the two bode plots, at low frequencies, the two systems behave similarly. Only at higher frequencies does the absence of the fast pole cause discrepancies between the two.

Given that the presence of the pole at s = -393 only affects the system at high frequencies, the inner compensator can be designed with the reduced order model. The pole zero map of the transfer function from β to u is shown below.

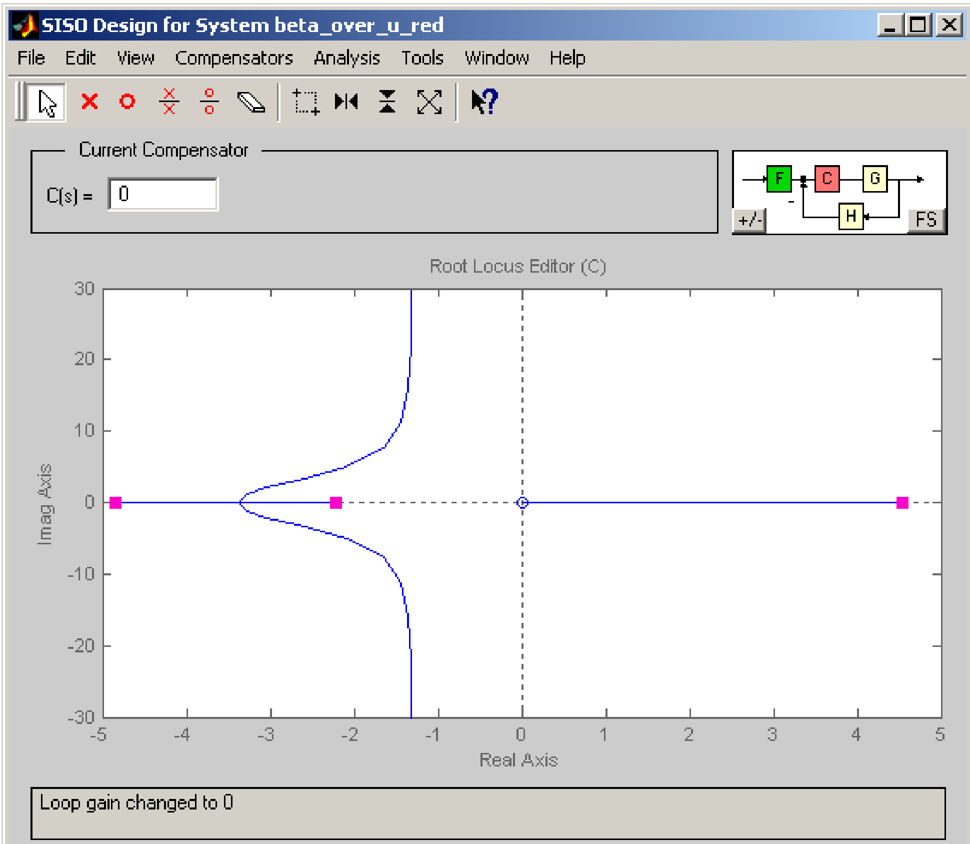


Figure 13: Open loop poles of $G_\beta(s) = \beta(s)/u(s)$ transfer function from reduced order system

Once again, even though the reduced order system is 4th order, there are only three poles because there is pole zero cancellation of the state x_1. As can be seen, there is a zero at the origin. Therefore, the compensator needs to have a pole at the origin to cancel this zero; otherwise the system can only become marginally stable at best with infinite gain. The following inner loop compensator is chosen in order to stabilize the response.

$$C_{inner}(s) = \frac{1083.6426\left(s^2 + 4.553s + 5.191\right)}{s\left(s + 114.9\right)}$$

Equation 15

The resulting closed loop poles with this compensator are shown below in Figure 14.

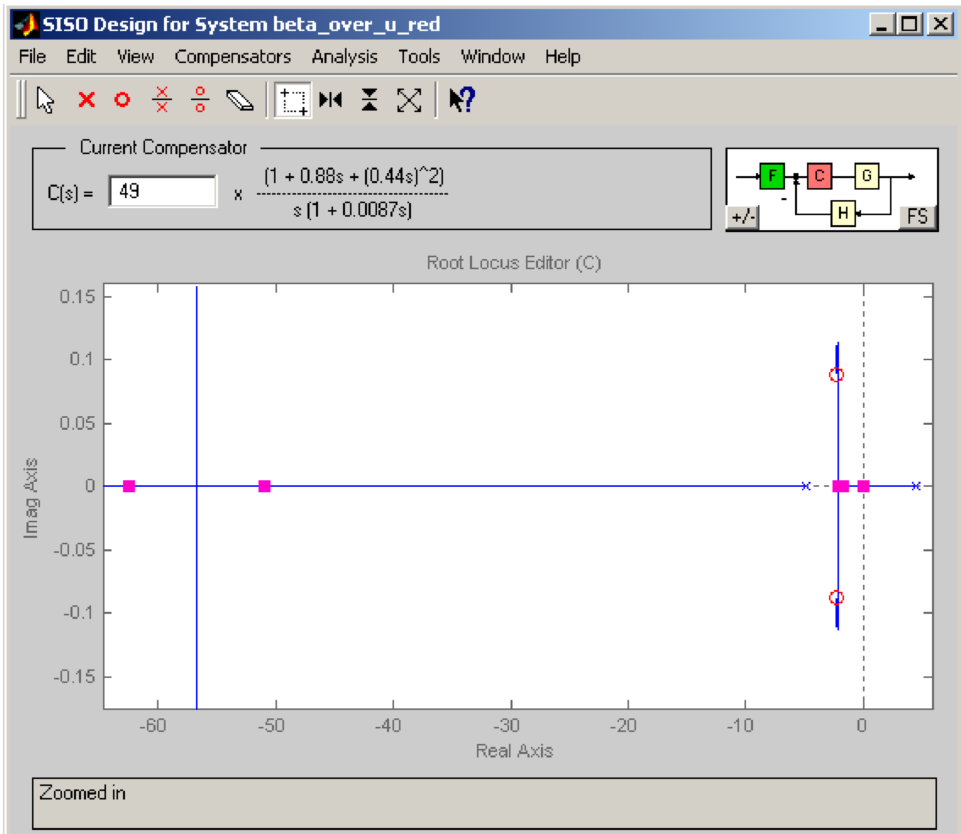


Figure 14: Closed loop poles with inner compensator

As can be seen, the resulting system is stable. This means that the system will be able to maintain $\beta = 0$. The closed loop poles are located at s = -62.5, -51,-2.18, -1.71, and 0. Note that all of these are only the real axis and therefore have a damping of 1. This is to ensure a response that has minimal oscillation.

Now that the inner loop has been designed, the outer loop, which will stabilize z, must be constructed. The transfer function between β_{cmd} and z can now be computed using the linmod function on the block diagram shown above in Figure 10 between the input (β_{cmd}) and the second output (z). This is then imported into SISOtool as the plant, G(s). The root locus of the open loop system is shown below in Figure 15.

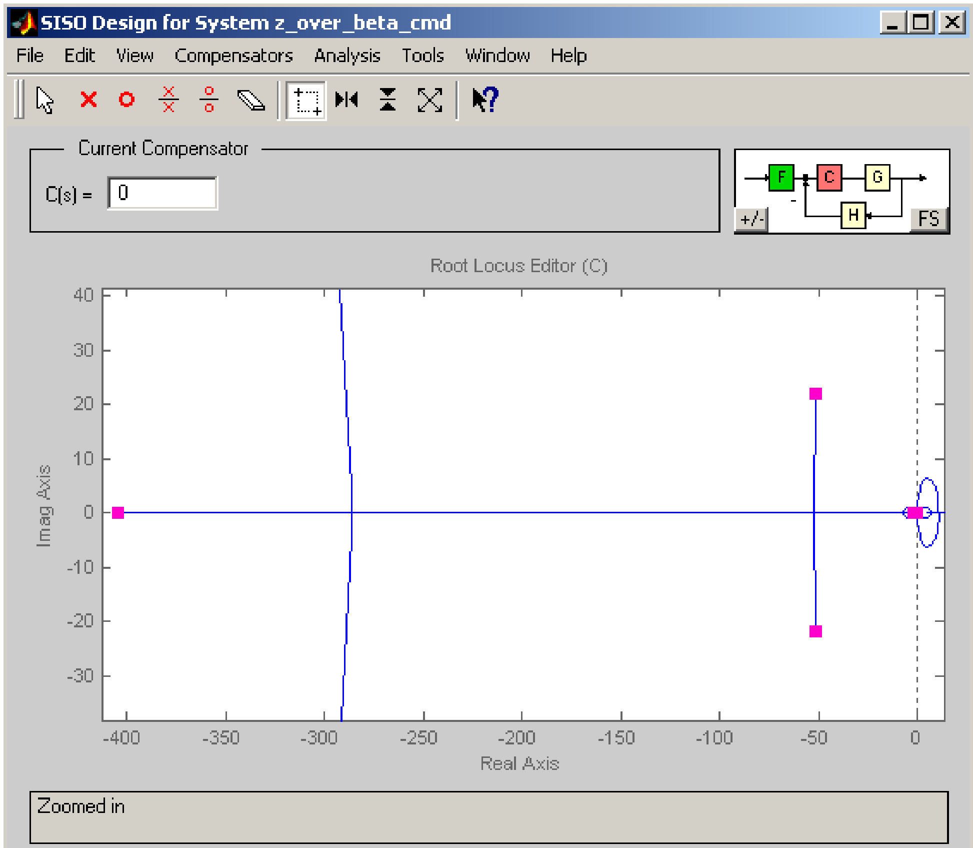
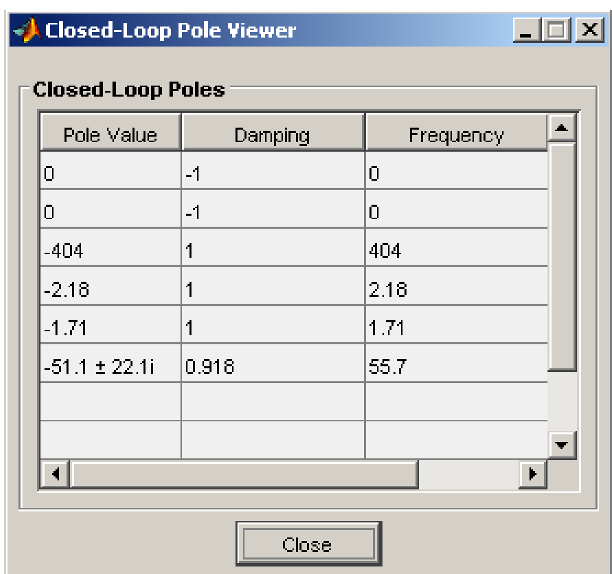


Figure 15: Root locus of $G_z(s) = z(s)/\beta(s)$

The closed loop poles of the system can be seen by using the closed loop pole viewer as shown below.



Pole Value	Damping	Frequency
0	-1	0
0	-1	0
-404	1	404
-2.18	1	2.18
-1.71	1	1.71
-51.1 ± 22.1i	0.918	55.7

Figure 16: Closed loop poles of $G_z(s)$

The interesting observation to make here is that the pole associated with the current at s ≈ -393 has shown back up in the linearization despite reducing the order of the model previously (s = -404). This is because the linmod function does not make the reduction assumption that $dx_5/dt \approx 0$. Therefore, although it was able to be removed manually previously when designing the inner loop, because the linmod function is used, the pole shows up again.

As can be seen in Figure 16, there are two repeated roots at the origin. Therefore, instead of being marginally stable, the system is in fact unstable. Therefore, an outer loop compensator must be designed to stabilize the z location. This corresponds to the following block diagram.

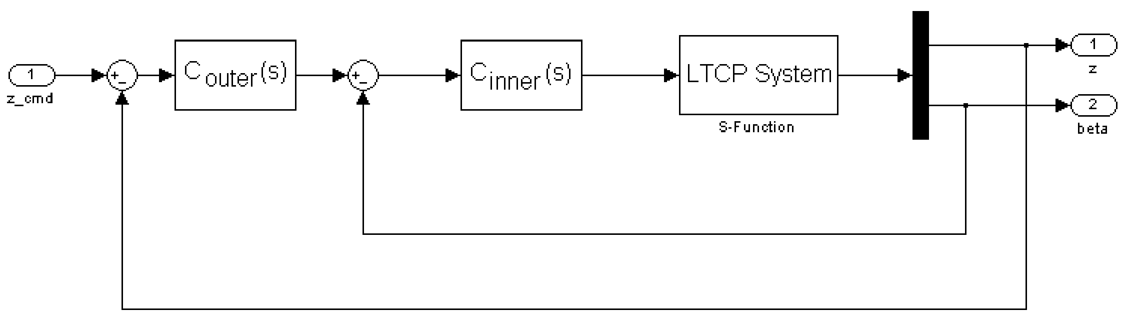


Figure 17: Inner and outer loop for classical control design

The following compensator is chosen to stabilize the z response.

$$C_{outer}(s) = \frac{306.1301\left(s^2 + 4.374s + 5.402\right)}{\left(s + 217.2\right)\left(s^2 + 11.06s + 32.07\right)}$$

Equation 16

The closed loop poles of the system (between z and z_{cmd}) are shown below in Figure 18.

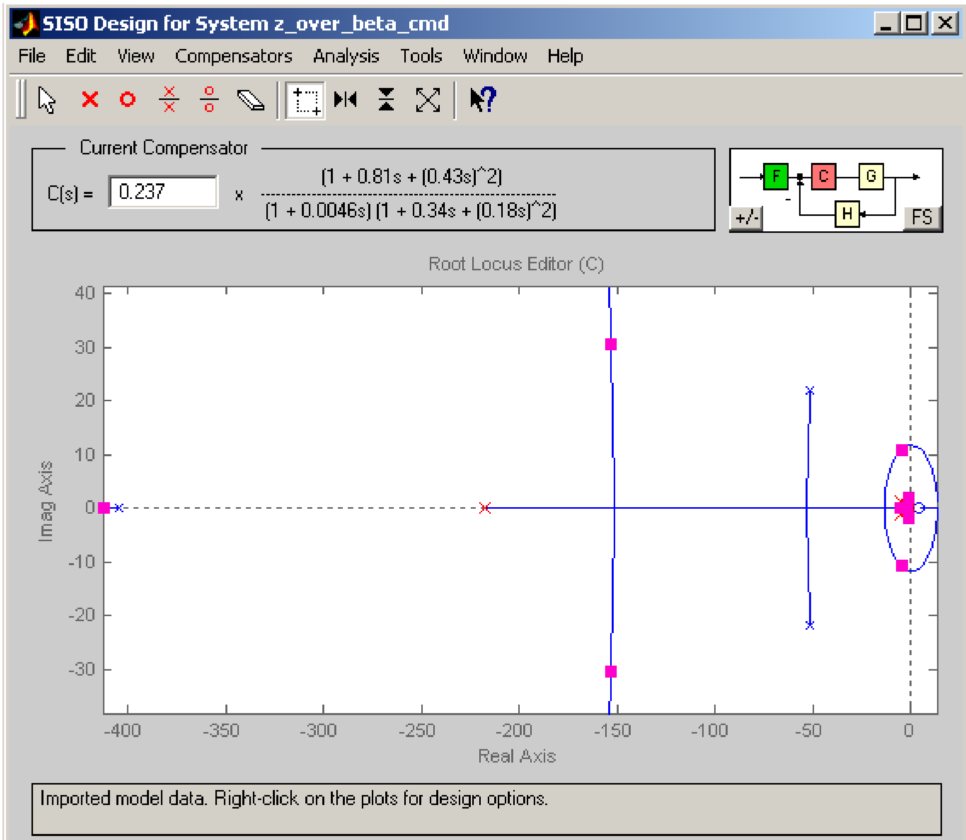
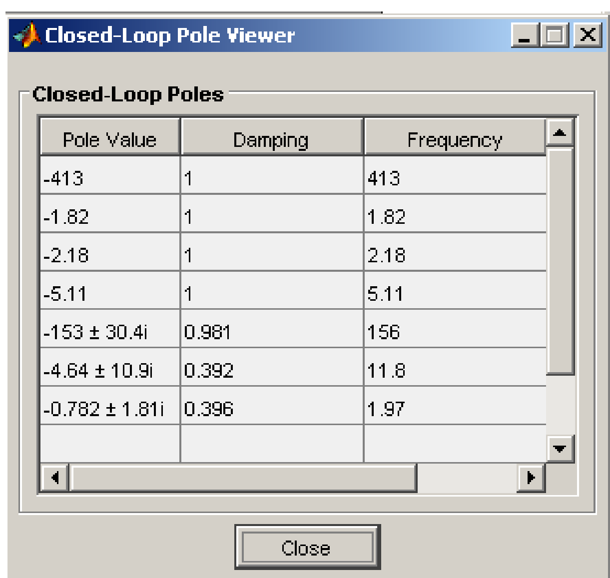


Figure 18: Closed loop roots of $z(s)/z_{cmd}(s)$

The closed loop poles are shown below in Figure 19.



Pole Value	Damping	Frequency
-413	1	413
-1.82	1	1.82
-2.18	1	2.18
-5.11	1	5.11
-153 ± 30.4i	0.981	156
-4.64 ± 10.9i	0.392	11.8
-0.782 ± 1.81i	0.396	1.97

Figure 19: Closed loop poles of $z(s)/z_{cmd}(s)$

As can be seen from the above figure, the closed loop response of the system to a z command is now stable as all the poles are in the left half plane. However, notice that the absolute values of the smallest real part of the closed loop poles is less than 1, therefore, the requirement of $T_s < 1$ sec is not satisfied.

Significant effort was expended in an attempt to make the classical control system comply more exactly with the specifications that were originally given. A classical controller would ideally have the following characteristics:

1. Settling Time to within 2% < 1s
2. Percent Overshoot < 20%
3. Rise Time < 1s
4. A bandwidth of 5 radians per second

These specifications continued goals of the design but it became evident that achieving them all with a classical controller designed via the root locus method would be very difficult and would require a great deal of time as the inner loops would frequently have to be redesigned to respond to the changes in the outer loops. The control system described above served as an initial step.

After a great deal of time, a controller was finally selected that had the best performance. The development of this controller is described below and follows the same general process as the controller described above. In this incarnation the model was not reduced and the fast pole corresponding to the current was maintained throughout the analysis. Because little attention was paid to this fast, stable root during the analysis the root loci diagrams below are shown zoomed into the region of interest.

Once again the linear model of the system was determined using the *Matlab* linmod command. This was then analyzed using the SISOtool. This inner loop was then stabilized using the controller given below.

$$C_{inner} = \frac{176.9\left(s^2 + 6.818s + 11.72\right)}{s(s + 39.94)}$$

Equation 17

The root locus of this controlled inner loop is shown below in Figure 20.

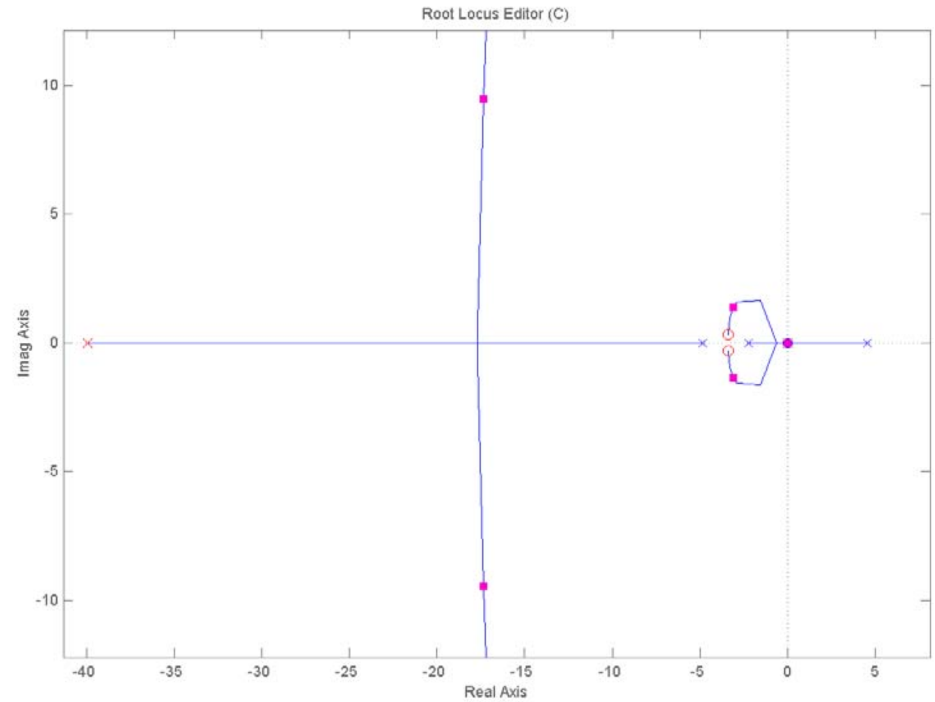


Figure 20: The root locus of the inner loop with the inner loop controller.

The close loop poles of this inner control system are shown below in Figure 21

Eigenvalue	Damping	Freq. (rad/s)
0.00e+000	-1.00e+000	0.00e+000
6.88e-014	-1.00e+000	6.88e-014
-3.10e+000 + 1.37e+000i	9.15e-001	3.39e+000
-3.10e+000 - 1.37e+000i	9.15e-001	3.39e+000
-1.74e+001 + 9.46e+000i	8.78e-001	1.98e+001
-1.74e+001 - 9.46e+000i	8.78e-001	1.98e+001
-3.95e+002	1.00e+000	3.95e+002

Figure 21: Closed Loop Damping and Poles of the Inner Loop

Once again it is observed that there is a pair of zeros at the origin. Thus the system is once again unstable in the z state. This is controlled by wrapping a second controller around the inner loop. This outer loop went through many variations as the desired performance was sought. There were little requirements for the construction and performance of the inner loop but the requirements for the outer loop were very stringent. It was eventually determined that the controller shown below was an admirable balance of the requirements.

$$C_{outer} = \frac{11.85(s+0.5)}{(s+28.74)(s+5.26)}$$

Equation 18

The root locus of this design is shown below in Figure 22.

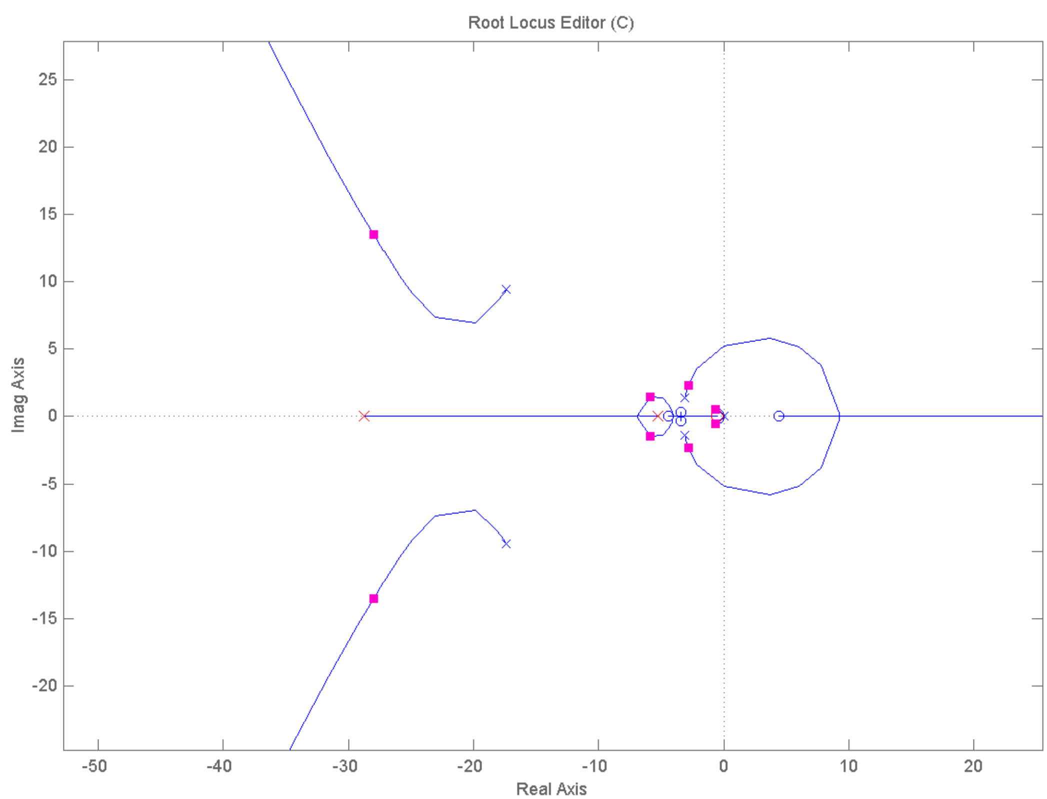


Figure 22: The Root Locus Diagram of the Outer Loop

In Figure 22 above there is a significant amount of activity near the origin and thus Figure 23 below shows a close up of this region. It is important not to forget that there is still the fast pole far to the left that corresponds to the current and is not shown in either Figure 22 or Figure 23 because of the scales.

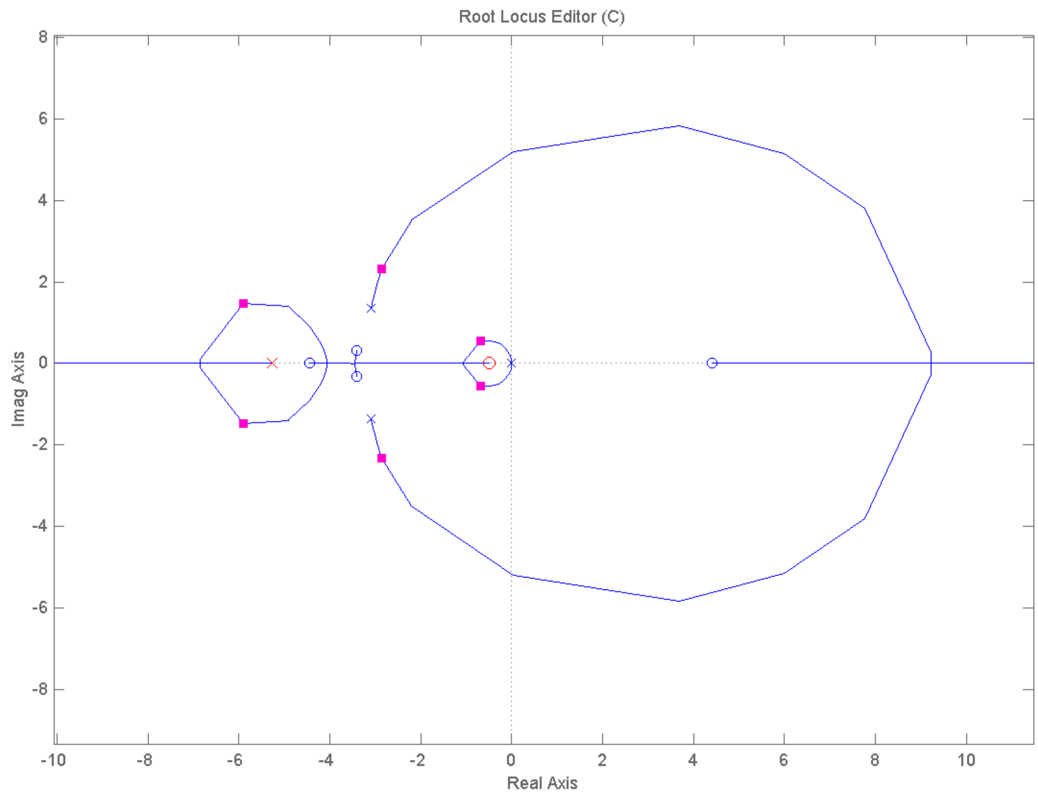


Figure 23: The Outer Loop Root Locus showing the region surrounding the origin in detail.

The closed loop poles of the system are shown below in Figure 24.

Eigenvalue	Damping	Freq. (rad/s)
-6.82e-001 + 5.48e-001i	7.79e-001	8.75e-001
-6.82e-001 - 5.48e-001i	7.79e-001	8.75e-001
-2.85e+000 + 2.33e+000i	7.75e-001	3.69e+000
-2.85e+000 - 2.33e+000i	7.75e-001	3.69e+000
-5.91e+000 + 1.48e+000i	9.70e-001	6.09e+000
-5.91e+000 - 1.48e+000i	9.70e-001	6.09e+000
-2.80e+001 + 1.35e+001i	9.00e-001	3.11e+001
-2.80e+001 - 1.35e+001i	9.00e-001	3.11e+001
-3.95e+002	1.00e+000	3.95e+002

Figure 24: The closed loop poles and damping of the combined control system.

This was the classical control system that was selected for further analysis and the one that received the most time on the actual equipment. A complete robustness analysis of this controller was performed and can be found below in the Classical Design Performance section.

Full State Feedback Design

In addition to classical control design, the dynamics of the system can be modified using full state feedback. The block diagram corresponding to the full state feedback system is shown below

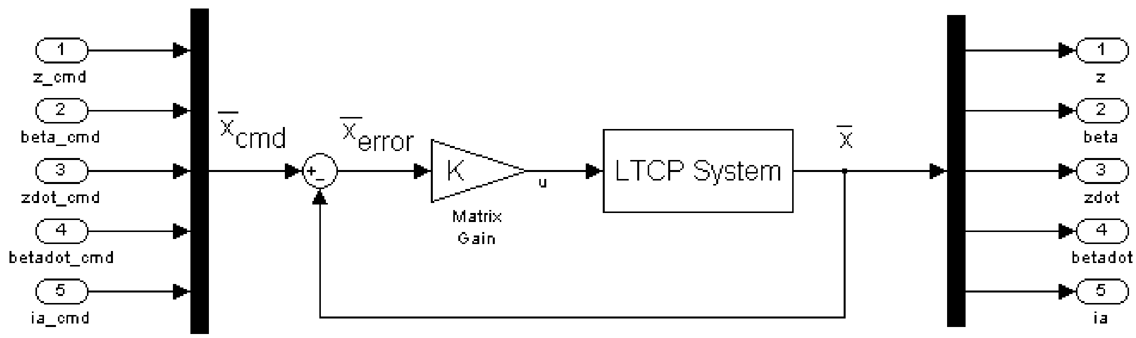


Figure 25: Block diagram of full state feedback architecture

Assuming that the system is fully controllable, by using a simply matrix gain, the state space representation of this system is given by

$$\frac{d}{dt}\overline{x} = \left(A - BK\right)\delta\overline{x} + BK\delta\overline{x}_{cmd}$$

Equation 19

As can be seen from Equation 19, the closed loop A matrix is given by A-BK. Since the gains K can be chosen to be theoretically anything, the eigenvalues of the closed loop system can be placed anywhere on the real/imaginary plane. This method of pole-placement (also known as eigen-assignment) can be used to design a full state compensator that will locate the closed loop poles in a desirable location.

The relationship between the gains and the location of the poles can be realized by writing out the desired characteristic equation.

$$CharEq_{desired} = \left(s + p_1\right)\left(s + p_2\right)\ldots\left(s + p_n\right)$$

Equation 20

Expanding this expression yields a polynomial of order n with coefficients that are functions of the desired poles. The characteristic equation can also be calculated using

$$CharEq = \det\left(sI - \left(A - BK\right)\right)$$

Equation 21

Expanding this expression yields and expression of order n with coefficients that are functions of only the gains k_1, k_2, …, k_n. Comparing the coefficients from the desired characteristic equation (from expanding the desired poles) and the coefficients from the actual characteristic equation (from taking the determinant) yields n equations with n unknowns (k_1 through k_n).

As shown in the Classical Control Design section, the system is fully controllable. Therefore, all of the poles can be moved to anywhere on the real/imaginary plane. However, as the name implies, full-state feedback requires knowledge of all the states. The states x_3 and x_4 can be obtained by differentiating x_1 and x_2, respectively. However, since there is no current measuring resistor, the state x_5 is not measured and therefore, with the current setup, it is impossible to obtain the state x_5. This effectively limits which poles can be moved. Recall that the pole λ_2 = -393.2 has the corresponding eigenvector

$$V_2 = (3.64\text{e-}6 \quad -7.28\text{e-}6 \quad -0.001 \quad 0.0028 \quad 0.9999)^T$$

This shows that this pole is almost entirely associated with the state x_5. Therefore, if the pole λ_2 = -393.2 is to be moved, it will require a gain on the state x_5. Since this gain cannot be provided, it is best to leave the pole λ_2 = -393.2 where it is in order to keep the gain on the state x_5 small.

Since there is no optimality associated with the eigen-assignment method, the desired poles are chosen in a semi-random fashion. The poles cannot be pushed too far to the left of the real/imaginary plane or else saturation of the actuator may occur. Furthermore, the pole at s = -393.2 cannot be moved for reasons stated previously. The desired poles are given by

```
desired_poles =

     -4
     -6
     -4 +          4i
     -4 -          4i
 -393.2
```

The gains required to place the system poles at these locations is calculated using the place command and is given by

```
K_matrix =

       25.173       64.197       18.528       14.354    -0.014191
```

As can be seen, by not moving the pole at s = -393.2, the gain k_5 is small and can be safely neglected.

Full State Feedback with Integrator Design

It will be shown later in the Full-State Feedback Performance section that the full state controller by itself is inadequate. It is unable to track a z command in the presence of a constant disturbance. Therefore, it becomes necessary to add an integrator on the z_{error} signal in order to reduce the steady state error to zero. The *Simulink* model used to simulate the system with the integrator and full state controller is shown in the Full State Feedback with Integrator Performance section as Figure 73.

Now that the integrator is added, the order of the model is increased to 6^{th} order. It now becomes necessary to obtain the state space model of the plant that is augmented with the integrator. Only the LTCP system and the integrator are shown below in Figure 26.

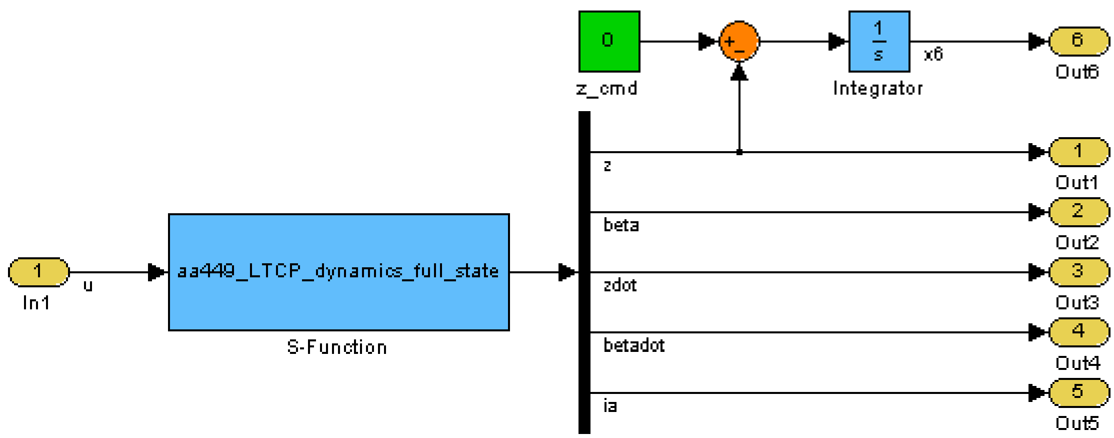


Figure 26: LTCP plant and integrator

The state space representation of Figure 26 now can be written. Recall that the state space representation of the LTCP plant alone is given by Equation 10, which is repeated here for convenience.

$$\frac{d}{dt}\overline{x} = A\delta\overline{x} + B\delta u$$

$$\overline{y} = C\delta\overline{x} + D\delta u$$

The state space representation of the integrator can easily be computed using control canonical form (assuming that $z_{cmd} = 0$).

$$\dot{x}_6 = -\delta x_1$$

$$y_6 = \delta x_6$$

Equation 22

Therefore the overall state space representation of the system is given by

$$\frac{d}{dt}\overline{x}_i = A_i + B_i \delta u$$

$$\overline{y} = I\overline{x}_i$$

Equation 23

where $\overline{x}_i = \begin{pmatrix} z & \beta & \dot{z} & \dot{\beta} & i_a & x_6 \end{pmatrix}^T$

$$A_i = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & 0 \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & 0 \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} & 0 \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} & 0 \\ A_{51} & A_{52} & A_{53} & A_{54} & A_{55} & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$B_i = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \dfrac{K_a}{L_a} \\ 0 \end{pmatrix}$$

As can be seen from Figure 26, the outputs are actually the states of the model. This result can be verified by using the linmod function

```
>> [Ai,Bi,Ci,Di]=linmod('aa449_integrator_plant_only')

Ai =

         0         0         1         0         0         0
         0         0         0         1         0         0
         0  -0.92583    -1.746 0.0014405   0.56119         0
         0    21.445    3.4887 -0.033366   -1.1213         0
         0         0   -518.94         0   -393.94         0
        -1         0         0         0         0         0


Bi =

         0
         0
         0
         0
   -1090.9
         0


Ci =

         1         0         0         0         0         0
         0         1         0         0         0         0
         0         0         1         0         0         0
         0         0         0         1         0         0
         0         0         0         0         1         0
         0         0         0         0         0         1
```

In standard fashion, a full state controller can now be designed such that $\delta u = -K_i \bar{x}_i$ and the closed loop poles of the entire system (controller and plant) are located at $\lambda(A_i - B_iK_i)$.

As a final check, the state space representation of Figure 73 (plant and controller block diagram) can be computed to show that the closed loop poles are given by $\lambda(A_i - B_iK_i)$. From the block diagram in Figure 73, the expression for δu is given by

$$\delta u = K\overline{x}_{cmd} - K_i\delta\overline{x}$$

Equation 24

where $K = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 & k_5 \end{pmatrix}$

$K_i = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & -k_6 \end{pmatrix}$

$\overline{x}_{cmd} = \begin{pmatrix} x_{1cmd} & x_{2cmd} & x_{3cmd} & x_{4cmd} & x_{5cmd} \end{pmatrix}^T$

$\delta\overline{x} = \begin{pmatrix} \delta x_1 & \delta x_2 & \delta x_3 & \delta x_4 & \delta x_5 & \delta x_6 \end{pmatrix}^T$

The state space representation of the augmented system is now given by

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = A_i + B_i\delta u + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ x_{1cmd} \end{pmatrix}$$

Equation 25

Substituting in the expression for δu yields the state space representation of the closed loop system.

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \left(A_i - B_iK_i\right)\delta\overline{x} + \left[B_iK + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \right]\overline{x}_{cmd}$$

Equation 26[5]

[5] See Appendix A for complete derivation

Therefore, this shows that the closed loop poles of the system are given by the eigenvalues of $A_i - B_iK_i$. Therefore, it shows that the model of the plant (LTCP and integrator) should be given by A_i and B_i as the A and B matrix. This can be verified using the linmod function

Given the matrix A_i and B_i, the place command can calculate the K_i matrix required to place the eigenvalues of $A_i - B_iK_i$. The values of K_i can be chosen such that the poles a placed at desirable locations.

Before attempting to place these poles, the system controllability must be checked. The system before the integrator was added was already confirmed to have full rank (see Classical Control Design section), but this does not guarantee that the system is fully controllable when the integrator is added.

```
>> rank(ctrb(Ai,Bi))

ans =

     3
```

As can be seen, this is not the correct response. The system rank cannot decrease from 5 to 3 simply by adding an integrator. The singular decomposition values can be checked to see if the controllability matrix is truly of rank 3.

```
>> svd(ctrb(Ai,Bi))

ans =

  1.0272e+016
       2088.6
       62.112
       6.6403
      0.66208
      0.48099
```

As can be seen from the svd command, the controllability matrix is actually full rank. The problem is that there is such a large range of values that the rank function incorrectly assumes that the numbers are small. Since the matrix is full rank, all of the poles can be moved using full state feedback. The desired poles are shown below

```
desired_poles =

          -4
          -6
          -4 +          4i
          -4 -          4i
      -393.2
          -2
```

These are the same poles as the full state feedback design with an extra pole at s = -2 for the integrator state. The gains needed to place the poles at the desired locations can now be calculated using the place command.

```
Ki_matrix =

       58.943       93.105       30.157       20.843    -0.016024      -50.346
```

The closed loop poles can be confirmed by using the linmod function on the *Simulink* model shown in Figure 73 between z and z_{cmd}. The A matrix will be the same between any input and any output. Notice that the last element of the Ki_matrix is –50.346. Due to the control architecture, this is actually $-k_6$. Therefore, $k_6 = 50.346$. We can now use Equation 26 to calculate the close loop state space representation of the entire system.

```
>> Ai_closed_loop = Ai-(Bi*Ki_matrix)

Ai_closed_loop =

           0           0           1           0           0           0
           0           0           0           1           0           0
           0    -0.92583      -1.746   0.0014405     0.56119           0
           0      21.445      3.4887   -0.033366     -1.1213           0
       64301  1.0157e+005      32379       22738     -411.42      -54922
          -1           0           0           0           0           0

>> Bi_closed_loop = Bi*[k1 k2 k3 k4 k5] + [0 0 0 0 0;
                                           0 0 0 0 0;
                                           0 0 0 0 0;
                                           0 0 0 0 0;
                                           0 0 0 0 0;
                                           1 0 0 0 0;]

Bi_closed_loop =

           0           0           0           0           0
           0           0           0           0           0
           0           0           0           0           0
           0           0           0           0           0
      -64301 -1.0157e+005     -32898      -22738      17.481
           1           0           0           0           0
```

We can verify that this is the correct state space representation using the linmod function on the closed loop system (Figure 73). Once again, the Ki_matrix returned from the place command cannot be directly used in the model. In the *Simulink* model, the last gain k_6 sign must be changed. Namely k_6 = -K_matrix(6). After this is changed, the linmod function can be used.

```
>> [ai,bi,ci,di] = linmod('aa449_full_state_integrator');
>> ai

ai =

           0            0            1            0            0            0
           0            0            0            1            0            0
           0     -0.92583       -1.746    0.0014405      0.56119            0
           0       21.445       3.4887    -0.033366      -1.1213            0
       64301  1.0157e+005        32379        22738      -411.42       -54922
          -1            0            0            0            0            0

>> bi

bi =

           0            0            0            0            0
           0            0            0            0            0
           0            0            0            0            0
           0            0            0            0            0
      -64301 -1.0157e+005       -32898       -22738       17.481
           1            0            0            0            0

>> eig(ai)

ans =

      -393.2
          -4 +          4i
          -4 -          4i
          -6
          -4
          -2
```

As can be seen, ai = Ai_closed_loop and bi = Bi_closed_loop.

LQR with Integrator Design

As shown above in the Full State Feedback section, the poles can be placed at any location on the real-imaginary plane. Although the placement of this appears to be arbitrary, it is possible to find an optimal location of these poles using linear quadratic regulator (LQR) theory. In order to do this, a cost function needs to be developed. The cost function J is defined as

$$J\left(\delta\overline{x}_i,\delta\overline{u},\overline{x}_o\right)=\int_0^{t_f}\left(\delta\overline{x}_i^{\,T}Q\delta\overline{x}+\delta u^T R\delta u\right)dt$$

Equation 27

As can be seen from inspection of Equation 27, J is a function of the response, control, and initial conditions. As can be seen, the cost function increases if there is a large response of the states from the equilibrium position and if there is a large amount of control applied. The LQR analysis goal is to find the optimal value of the gain matrix K_i that will minimize the cost function J. For full state feedback control architecture, the gain matrix that will minimize the cost function is given by

$$K_i = R^{-1}B_i^T S$$

Equation 28

where S is the solution to the Riccati equation shown below

$$A_i^T S + SA_i - SB_iR^{-1}B_i^T S + Q = 0$$

Equation 29

As can be seen, the gains are directly related to the values of the Q and R matrix.

The gain matrix K_i can be calculated using the lqr function in *Matlab*. In order to use this, the Q and R matrices must be defined. Given that A_i is a 6x6 matrix, Q must be a 6x6 matrix. Likewise, since u is simply a scalar (single control), then R is also a scalar. In order to tailor the gains, the values in the Q and R matrices can be chosen such that certain states or controls are penalized more or less than others. By inspection of Equation 27 (cost function), it can be seen that the diagonal elements of Q are the direct penalties of the states. Therefore, Q can be chosen as a diagonal matrix where the different diagonal elements penalize the different states. Since it is desired that the system track a z command well, the penalty on x_1 and x_6 should be high. Also, since dz/dt, dβ/dt, and i_a are not of concern, their penalties will be set to zero. This effectively means that the states x_3, x_5, and x_6 will be allowed as large a response as needed. This yields the following Q matrix

```
Q =

    10     0     0     0     0     0
     0     1     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0    10
```

As can be seen, only the diagonal elements have values that correspond to the penalties on these states. Off diagonal elements could also be added to signify penalties on states due to the coupling with other states, but since this adds unnecessary complexity, it is not included here.

In order to accurately track a z command, the gains must be relatively high. This also means that a lot of control must be used in order to track z with a small error. Therefore, the penalty on u should be small. This leads to the following R matrix

```
R =

    0.0075
```

Now that the Q and R matrix are defined, the optimal gains that satisfy the Riccati solution using the lqr function

```
>> [Ki,S,E] = lqr(Ai,Bi,Q,R);
>> Ki

Ki =

   60.4604  102.9649   33.4495   22.8921   -0.0171  -36.5148
```

Notice that the last element of the K_i matrix is actually $-k_6$ since the lqr function calculates K_i such that $u = -K_i x_i$ yields the desired closed loop poles. As stated above in the Full State Feedback with Integrator Design section, the K_i matrix is actually $K_i = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & -k_6 \end{pmatrix}$.

Observer Design

As can be seen in the previous two sections, the concept of full state feedback is dependent on the assumption that all of the states are available for to use for feedback. However, in reality, only the states x_1 and x_2 are actually available for feedback. The other states must be estimated. One method is use an observer design to reconstruct the states of the system using knowledge of the dynamics. The linear model of the real system can be written as

$$\frac{d}{dt}\overline{x} = A\overline{x} + Bu + \Gamma\overline{w}$$

$$\overline{y}_{sensor} = C_{sensor}\overline{x} + D_{sensor}u + \overline{v}$$

Equation 30

where Γ = disturbance input distribution matrix
$\overline{w}$ = process noise
$\overline{v}$ = sensor noises

In this application, the disturbances to the plant can be modeled as entering the plant through the control signal (ie a noisy control signal), therefore, Γ = B. Therefore, the process noise vector, $\overline{w}$, is simply a scalar which describes how the input signal fluctuates. Similarly, the sensor noise vector, $\overline{v}$, describes how the noise in the sensor signal fluctuates. The expectation of the process noise is defined as.

$$E\left[\overline{w}(t)\right] = \overline{\overline{w}}$$

Usually, $\overline{\overline{w}}$ = 0, which states that the mean value of the noise is zero and is considered random.

The covariance matrices of these processes can be evaluated. These are defined as

$$W_o = E\left[\left(\overline{w}(t) - \overline{\overline{w}}\right)\left(\overline{w}(t) - \overline{\overline{w}}\right)^T\right]$$

Since $\overline{w}$ is simply a scalar, the covariance matrix W_o is also a scalar that describes how much the signal varies. A large value of W_o implies that the process noise is very noisy and the dynamics derived using the linear model may be inaccurate.

In a similar fashion, the covariance matrix for sensor noise, V_o can be defined. However, this will be a 2x2 matrix since the sensor noise matrix is a 2x1 vector.

$$V_o = \begin{pmatrix} \sigma^2_{v_1} & \mu_{v_1,v_2} \\ \mu_{v_1,v_2} & \sigma^2_{v_2} \end{pmatrix}$$

where $\sigma^2_{v_1}$ and $\sigma^2_{v_2}$ = variance of the sensor noise
μ_{v_1,v_2} and μ_{v_2,v_1} = correlation noise elements

Since the two sensor noise elements are not correlated (ie noise in the β sensor does not affect the noise in the z sensor), the $\mu_{v_2,v_1} = \mu_{v_1,v_2} = 0$. Large values of $\sigma^2_{v_1}$ and $\sigma^2_{v_2}$ imply that the sensor are very noisy and should not be used for feedback.

Due to the design of the system having only two sensors, the $\overline{y}_{sensor}$ output does not include all states. Therefore, using knowledge of the linear plant model (ie the A and B matrices), the state space equation of the estimator can be written as

$$\frac{d}{dt}\hat{x} = A\hat{x} + Bu + L\left(\overline{y}_{sensor} - \hat{y}\right)$$

$$\hat{y}_{observer} = I\hat{x} + D_{observer}u$$

Equation 31

where $\hat{y} = C_{sensor}\hat{x} + D_{sensor}u$
$\hat{x}(0) = \overline{0}$

Equation 30 describes how the estimated states progress with time. The *Simulink* model used to implement this observer is shown below in Figure 27.

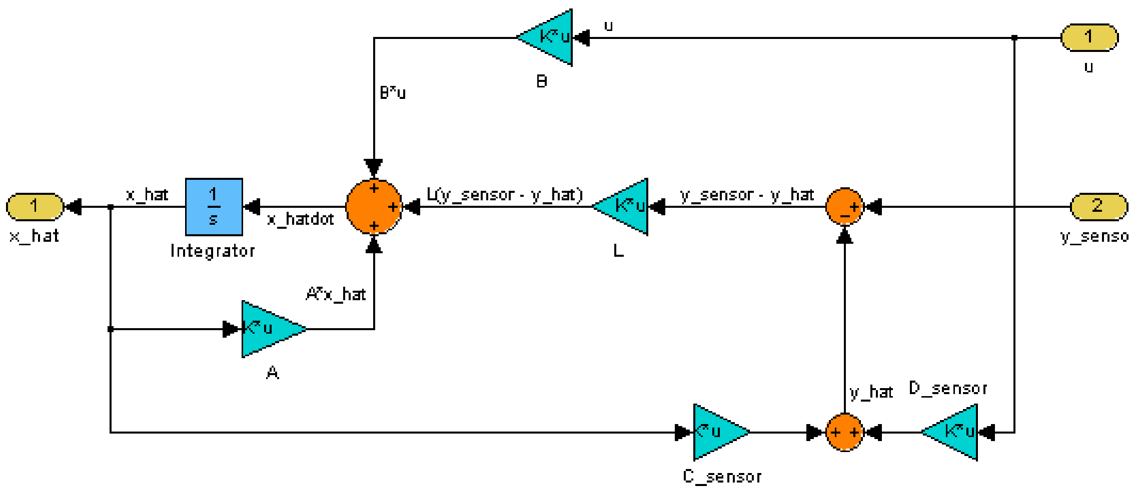


Figure 27: Observer Filter

Given a stable observer, the estimated states will converge to the actual states as time goes to infinity. Generally, the initial conditions of the system are not known, so the estimated states are initialized at zero. The dynamics of the error, $e = (\hat{x} - \overline{x})$, is evident by subtracting Equation 30 from Equation 31.

$$\frac{d}{dt}\overline{e} = \left(A - LC_{sensor}\right)\overline{e} + L\overline{v} - \Gamma\overline{w}$$

$$\overline{e}(0) = -\overline{x}(0)$$

The observer gain matrix, L can be chosen such that the eigenvalues of A-LC_{sensor} are stable. This guarantees that the error between the estimated states and the actual states will go to zero as time goes to infinity. The poles governing the error dynamics can be place at any location. One method to place the gains is the eigen-assignment method that is similar to the method discussed in the Full State Feedback Design section. The desired poles of the error using this method are shown below.

```
desired_error_poles =

        -390
          -5
          -6
          -2 +          1i
          -2 -          1i
```

The *place* command can be used after some matrix manipulation, namely using the transpose operation. The *place* command returns the gains K used to place λ(A-BK) at the desired locations. However, in this situation, the matrix L is to be found which places $\lambda(A-LC_{sensor})$ at the desired locations. The eigenvalues of a matrix do not change with the transpose command.

$$\lambda\left(A-LC_{sensor}\right)=\lambda\left(A-LC_{sensor}\right)^{T}=\lambda\left(A^{T}-C_{sensor}{}^{T}L^{T}\right)$$

Notice how the last term has the correct order and arguments for the place command. *place*(A^T,$C_{sensor}{}^T$,desired_error_poles) will return the L^T which places the poles at the desired location.

```
>> K = place(A',C_sensor',desired_error_poles);
>> L = K'

L =

       4.1216        1.674
        2.285       5.1596
        249.3      -493.23
      -495.05       1018.7
  -1.743e+005  3.4406e+005
```

This observer gain matrix can be used with the model shown above in Figure 27 to estimate the states of the system.

As will be shown in the Observer Performance section, the eigen-assignment method of selecting the observer gain matrix is somewhat random and does not perform as well as desired. The second method to place the error poles is to use the *lqe* function in *MatLab*, which adds optimally to the placement, resulting in a Kalman filter. In order to do this, the covariance matrix of process noise (W_o) and the covariance matrix of sensor noise (V_o) must be defined. The overall goal of the Kalman filter is to provide accurate estimates of the system states. The magnitude of the Kalman gain (L) can be adjusted using W_o and V_o. A large value of W_o and small V_o implies that the plant is very noise compared to the sensors, and therefore, the sensors should be used and the estimated states should be assumed to be accurate. Conversely, a large V_o and small W_o implies that the sensors are noisy, and therefore, any states estimated from these noisy signals may not be accurate. In this situation, a large value of W_o and relatively small V_o is used. Furthermore, as stated above, it is assumed that the sensor noise is uncorrelated, so the off diagonal elements of V_o are zero. Also, since the plant process noise is assumed to enter through the amplifier, $\Gamma = B$. The *lqe* function is now used to calculate the Kalman gain.

```
>> Wo = 1e3;
>> Vo = diag([0.1 0.1]);
>> [L,Q,E] = lqe(A,B,C_sensor,Wo,Vo);
>> L

L =

    8.6202      -7.6327
   -7.6327       21.042
    66.283      -110.31
   -116.09       250.51
    -36.47       43.783
```

This set of Kalman gains can be used to estimate all of the system states from the two measurements made. The poles of the error can also be calculated in order to ensure that they are stable.

```
E =

     -393.2
   -13.575 +     12.808i
   -13.575 -     12.808i
   -2.5171 +     1.5567i
   -2.5171 -     1.5567i
```

As can be seen, these are all stable and therefore, the error between the estimated states and the actual states should go to zero in the linear case.

IX. Experimental Procedure

Building the model for *XPCtarget*

The main difference between the simulation and actual *XPCtarget* model is the plant. In simulation, the plant is modeled using an s-Function or a linear state space block. However, in the actual system, the actual hardware is the plant. This is called a hardware in the loop simulation. Special *XPCtarget* blocks must be used to interface the *Simulink* model with the actual hardware. This is shown below in Figure 28.

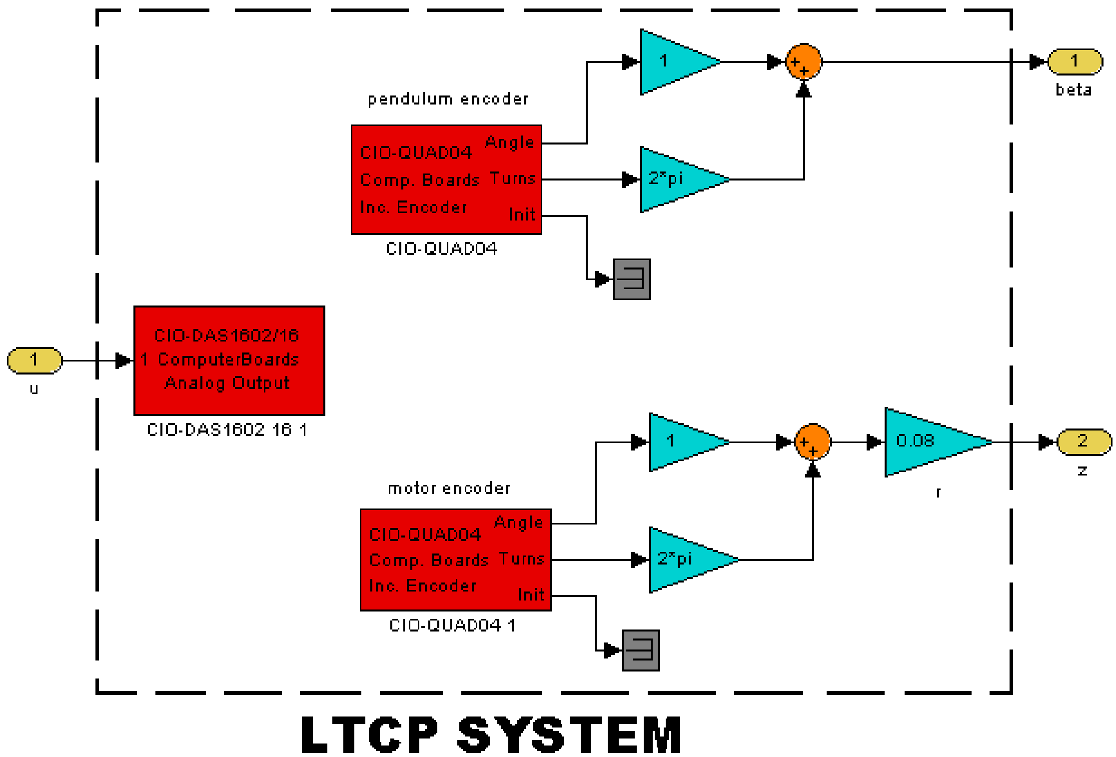


Figure 28: *Simulink* blocks used to interface with *XPCtarget*

Find the three red blocks shown in Figure 28 from the *Simulink* library under the *XPCtarget* subsection. Ensure that the D/A converter (CIO-DAS 1602 16 1) block settings are the same as shown below in Figure 29.

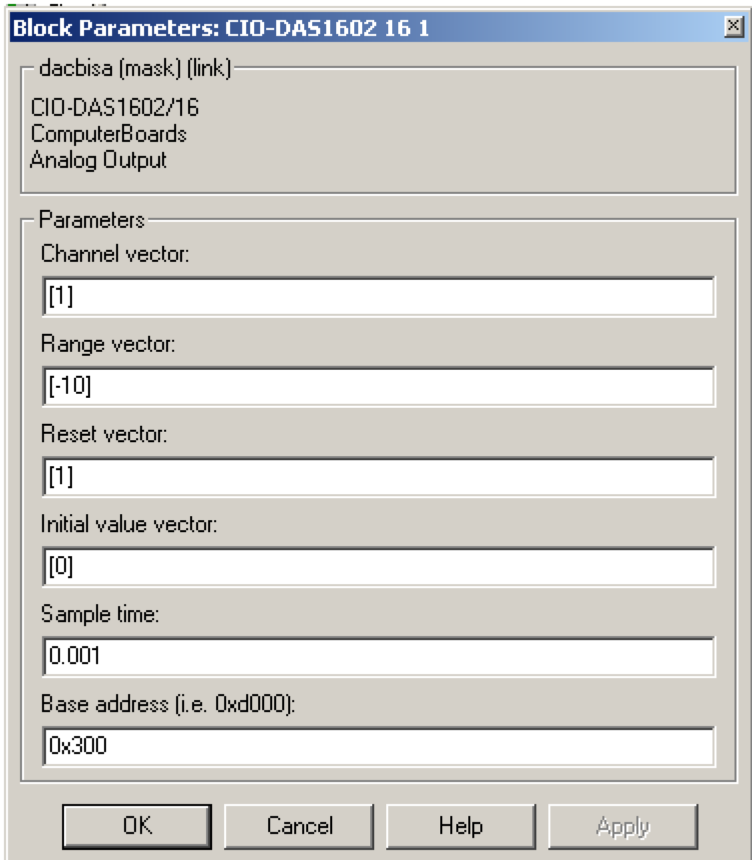


Figure 29: D/A converter settings

Ensure that the pendulum encoder block (CIO-QUAD04) settings are the same as those shown below in Figure 30.

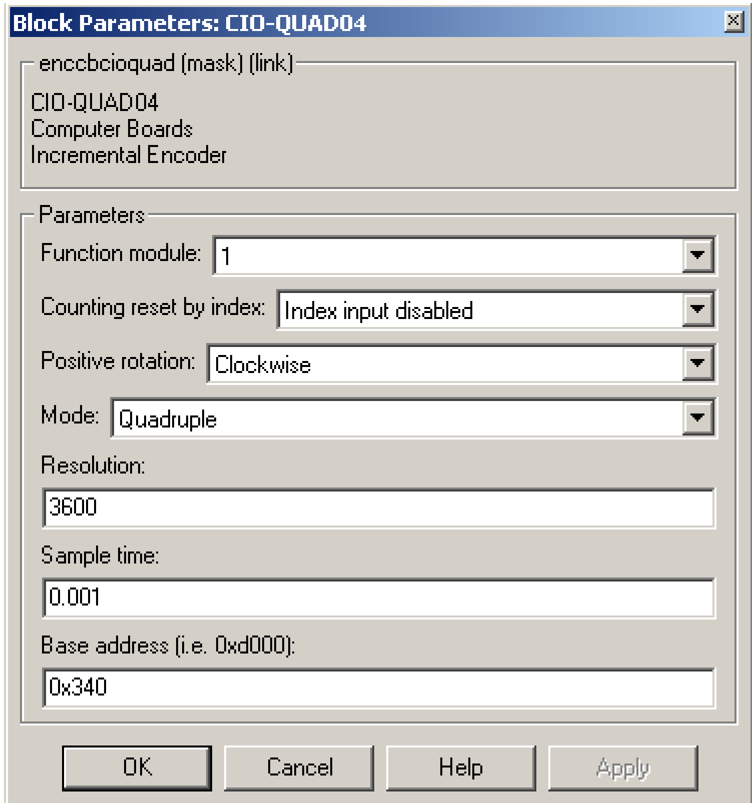


Figure 30: Pendulum encoder settings

Ensure that the motor encoder (CIO-QUAD04 1) settings are the same as the pendulum encoder settings except for the "Function module:" setting, which should be set to 2.

The z and β encoders should already be hooked up and interface directly to the computer without being routed through the green *LabVIEW* data card. Ensure that the ground line for the Kepco amplifier is hooked into slot 19 on the green *LabVIEW* data card. Also, make sure that the signal to the Kepco amplifier is hooked into slot 9 on the green LabVIEW data card.

Turn on the Kepco amplifier.

Build the controller and the LTCP system models in *Simulink* and select Tools > Realtime Workshop > Build Model. This translates the *Simulink* into C code and uploads it to the *XPCtarget* computer.

Click the "Connect to Target" button on the toolbar and press the play button to begin the realtime code.

Applying Disturbance Force

A constant force can be placed on the cart using the linear push/pull gauge. First, start the real time code with the pendulum in the vertical position.

Once the pendulum stabilizes, apply a constant force on the cart as shown below in Figure 31.

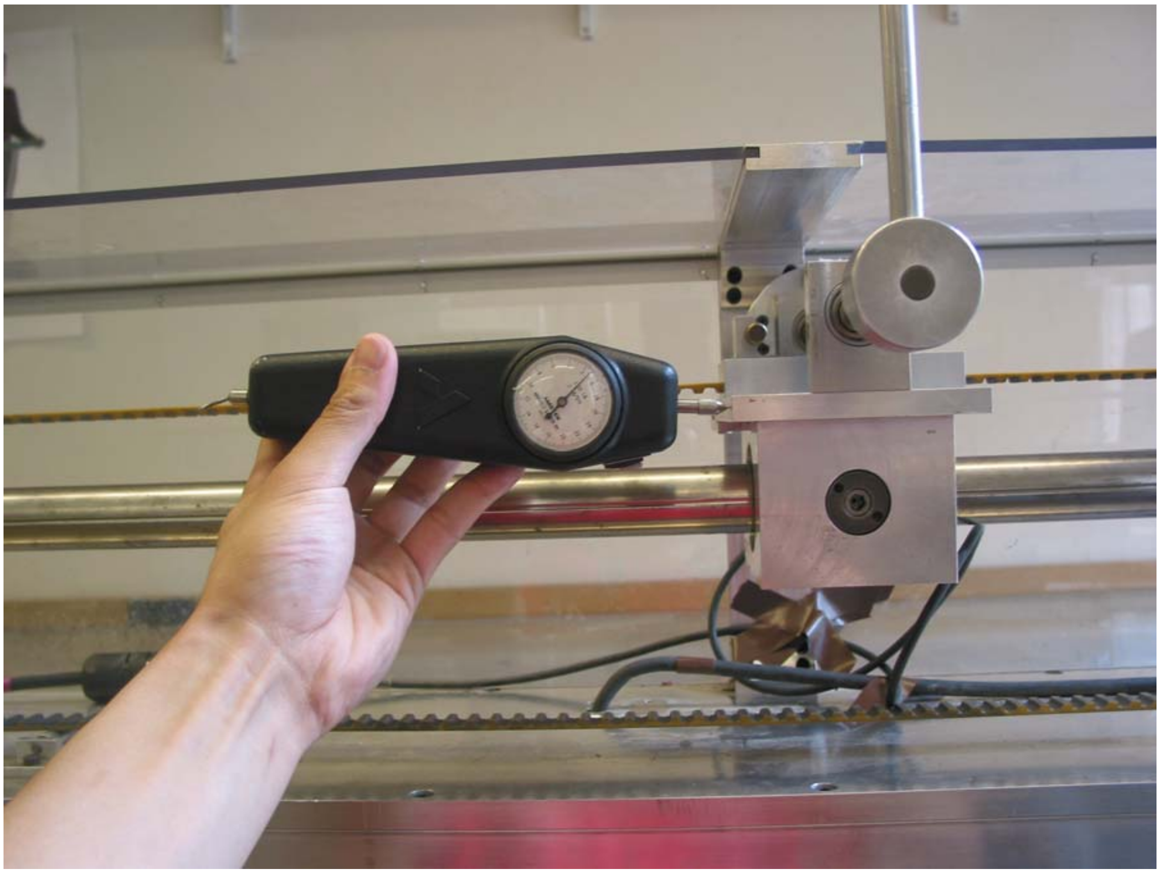
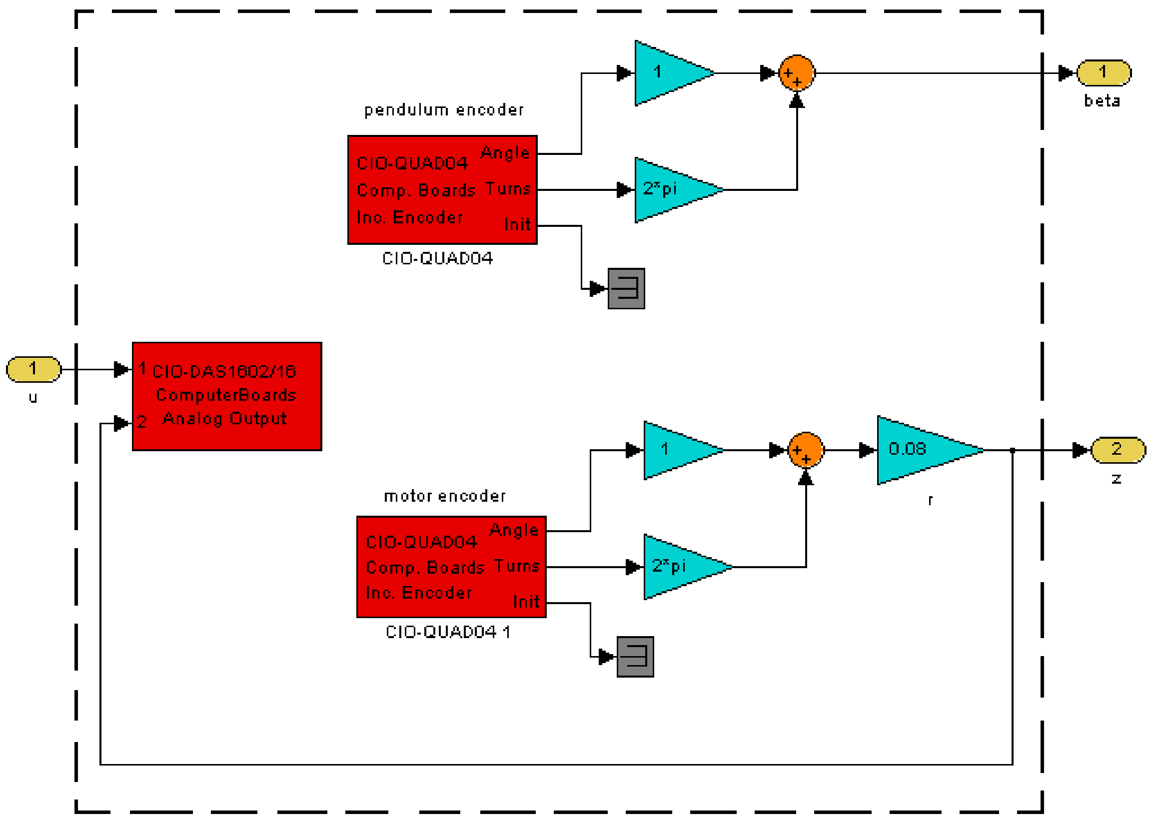


Figure 31: Applying a positive f_1 force

Connecting the DSA

The Dynamic Signal Analyzer can be used to find the frequency response of the system between z and z_{cmd}. In order to do this, there must be several modifications made to the LTCP System block. The DSA will output a signal, which will be converted from the analog signal to a digital signal to be used as the z_{cmd}. The z-encoder reading can then be converted from a digital signal to an analog signal and used as the input into the DSA.

Modify the LTCP *Simulink* block to be the same as that shown below in Figure 32.



LTCP SYSTEM WITH DSA

Figure 32: Modified LTCP system block to include digital to analog conversion of z signal

Change the digital to analog converter block to have the same parameters as that shown below in Figure 33.

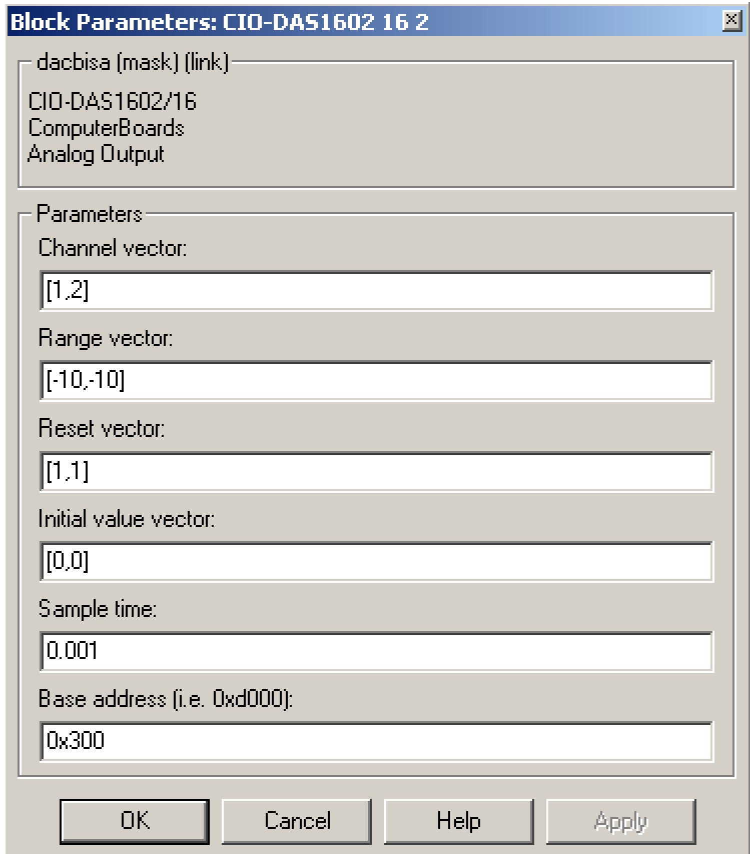
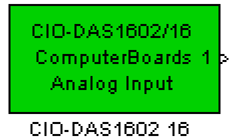


Figure 33: Modified parameters for digital to analog converter

Now, obtain the analog to digital block that will convert the analog signal from the DSA to a digital z_{cmd} signal. The block (CIO-DSA 1602 16) and its parameters are shown below in Figure 34.



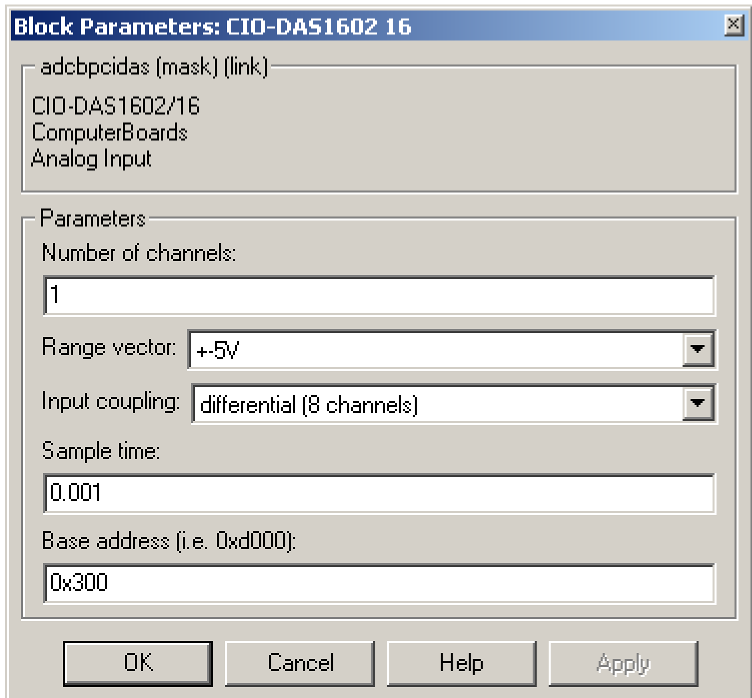


Figure 34: z_{cmd} block and parameters

Now that all of the *Simulink* blocks are in place, the actual DSA input and output must be hooked up to the *LabVIEW* data card.

Connect the DSA source ground to slot 18 on the *LabVIEW* data card
Connect the DSA source signal to slot 37 on the *LabVIEW* data card
Connect the DSA channel 2 ground to slot 19 on the *LabVIEW* data card
Connect the DSA channel 2 signal to slot 27 on the *LabVIEW* data card

The fully connected LabVIEW data card should look like Figure 35.

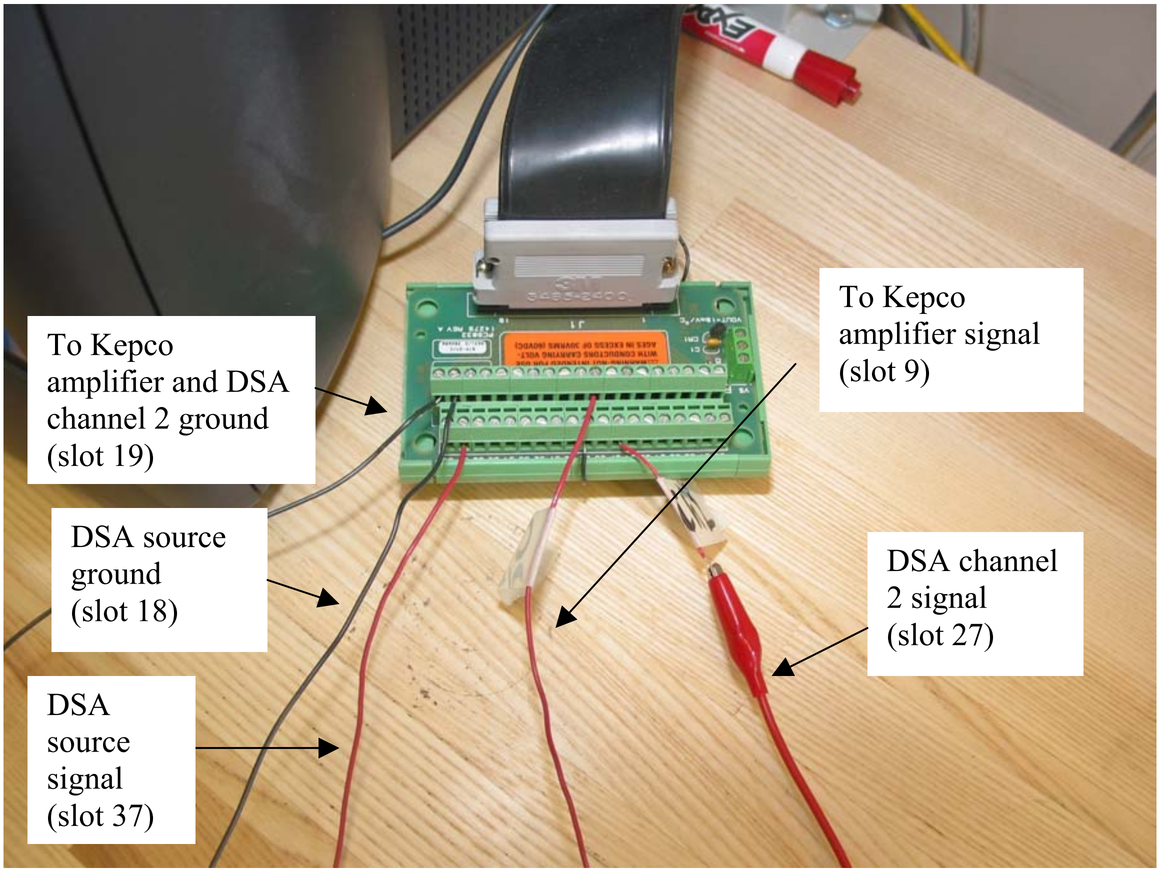


Figure 35: LabVIEW data card connections for DSA analysis

The DSA is now connected to the model and can be used to analyze the frequency response of the z over z_{cmd} transfer function. For information regarding programming the DSA, see Appendix C.

X. Modeling Validation

In order to simulate the response of the system with the various controllers, it becomes necessary to develop an accurate model of the plant dynamics. The equations of motion have already been derived and the numerical values of the parameters are given in Table 1. Given the time constraint on this project, it would not be efficient to derive experiments to test the accuracy of these constants so the published values will be used. However, since the effect of the coulomb friction (stiction was determined negligible as shown in the System Modeling > Friction Modeling section) was not published, it is necessary to derive its effect. After this is done, the parameters must be compared with experimental data in order to validate the accuracy of the plant model. Using a coulomb friction model of

$$f_s + \frac{\tau_s}{r} = -1.1 sign\left(x_3\right)$$

Equation 32

The simplest way to evaluate the accuracy of the friction model is to allow the pendulum to swing freely and to apply a constant control voltage to the system. The coulomb friction is a necessary part of the system and cannot be neglected. A constant control of u = -1 volts is applied until the cart reaches 0.5 m and then the control is turned off. The response of the model without coulomb friction is shown below in Figure 36.

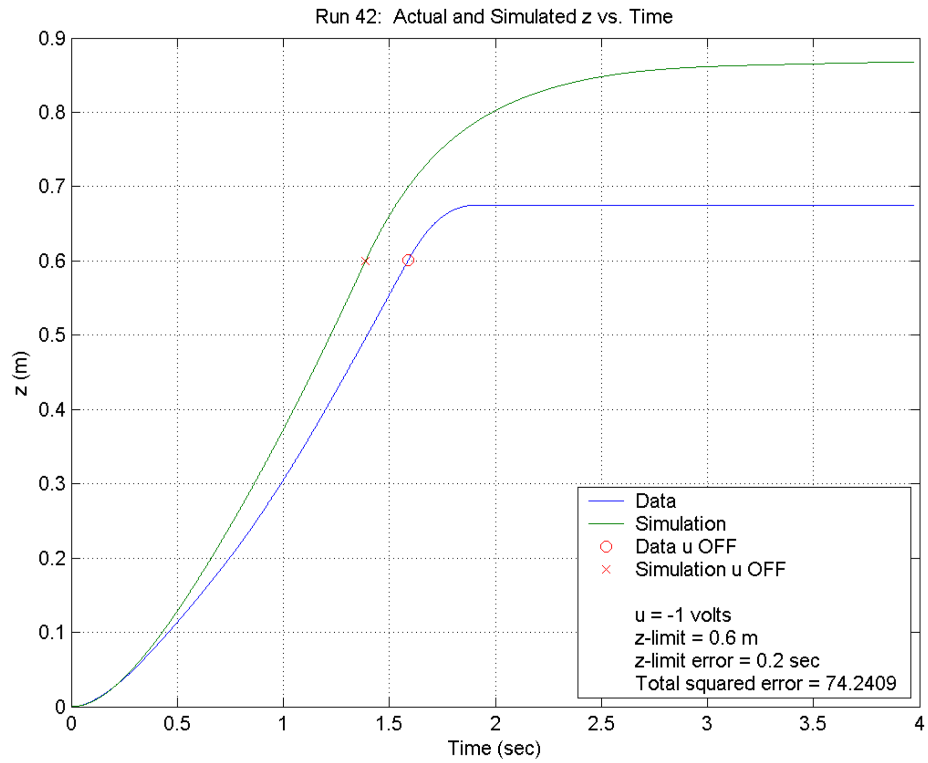


Figure 36: Response of system to constants u = -1 volts with no coulomb friction

As can be seen, the model matches fairly well in the section where there is control applied, but as soon as the control is off and the pendulum is "coasting", the simulation deviates from the actual response (Total squared error of 74.24). The response to the same control sequence with the friction term added to the non-linear S-Function is shown below in Figure 37.

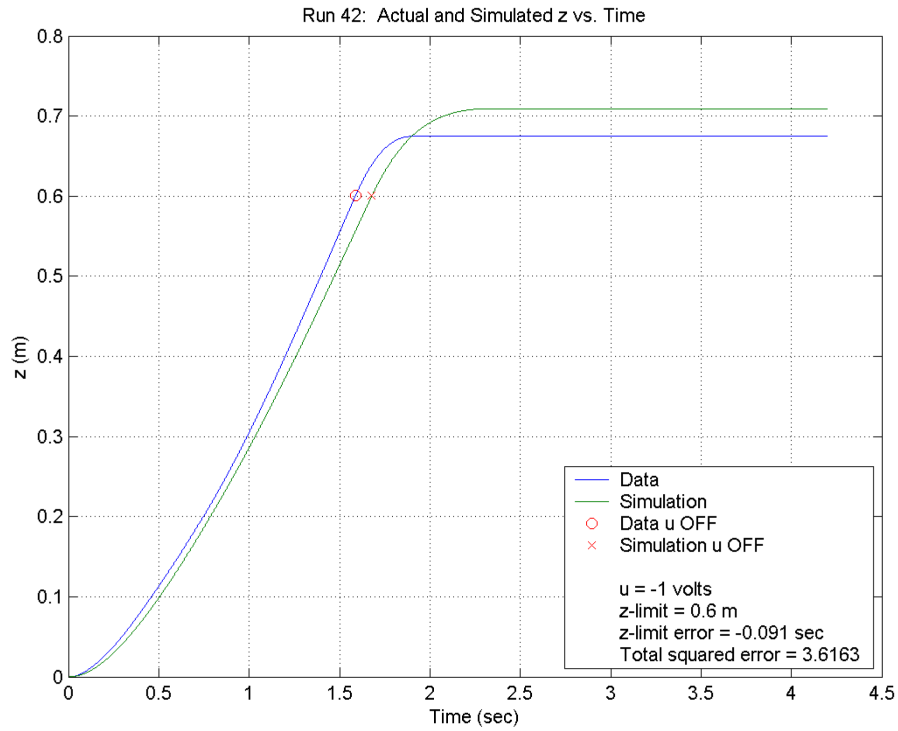


Figure 37: Response of system to constants u = -1 volts with coulomb friction

As can be seen in Figure 37, the simulated response matches the actual response significantly better when the coulomb friction is included in the nonlinear S-Function. The total squared error is now reduced to only 3.61, a significant improvement from 74.24 when the friction is neglected.

The model accuracy can also be checked when the control is a dynamic signal instead of a simple step. Also recall that there are 5 states associated with the non-linear model, therefore, the other simulated states (β, dz/dt, etc.) can also be checked against experimental data.

The simulated and actual response of the system to a ramp of slope = +1 volt/sec is shown below in Figure 38.

The simulated and actual response of the system to a sin wave of amplitude = 1.5 volts and a frequency of 5 rads/sec is shown below in Figure 39.

The simulated and actual response of the system to a series of steps, the first of magnitude = +2 volts @ t = 1 sec, the second of magnitude = -4 volts @ t = 1.6 sec, and the last of magnitude = +2 volts @ t = 2.2 sec, is shown below in Figure 40.

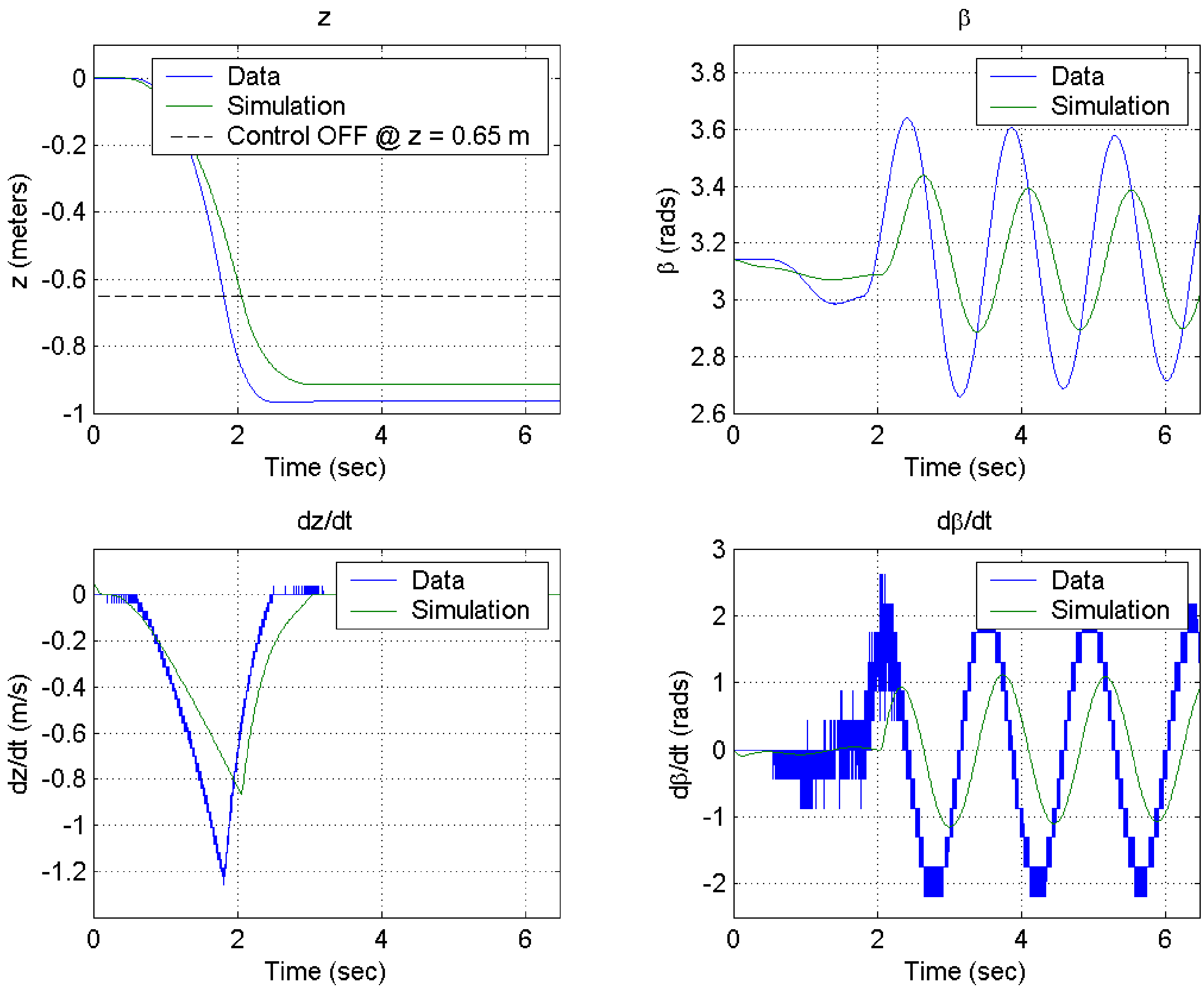


Figure 38: Actual and simulated response of system to ramp input with slope = +1 volt/sec

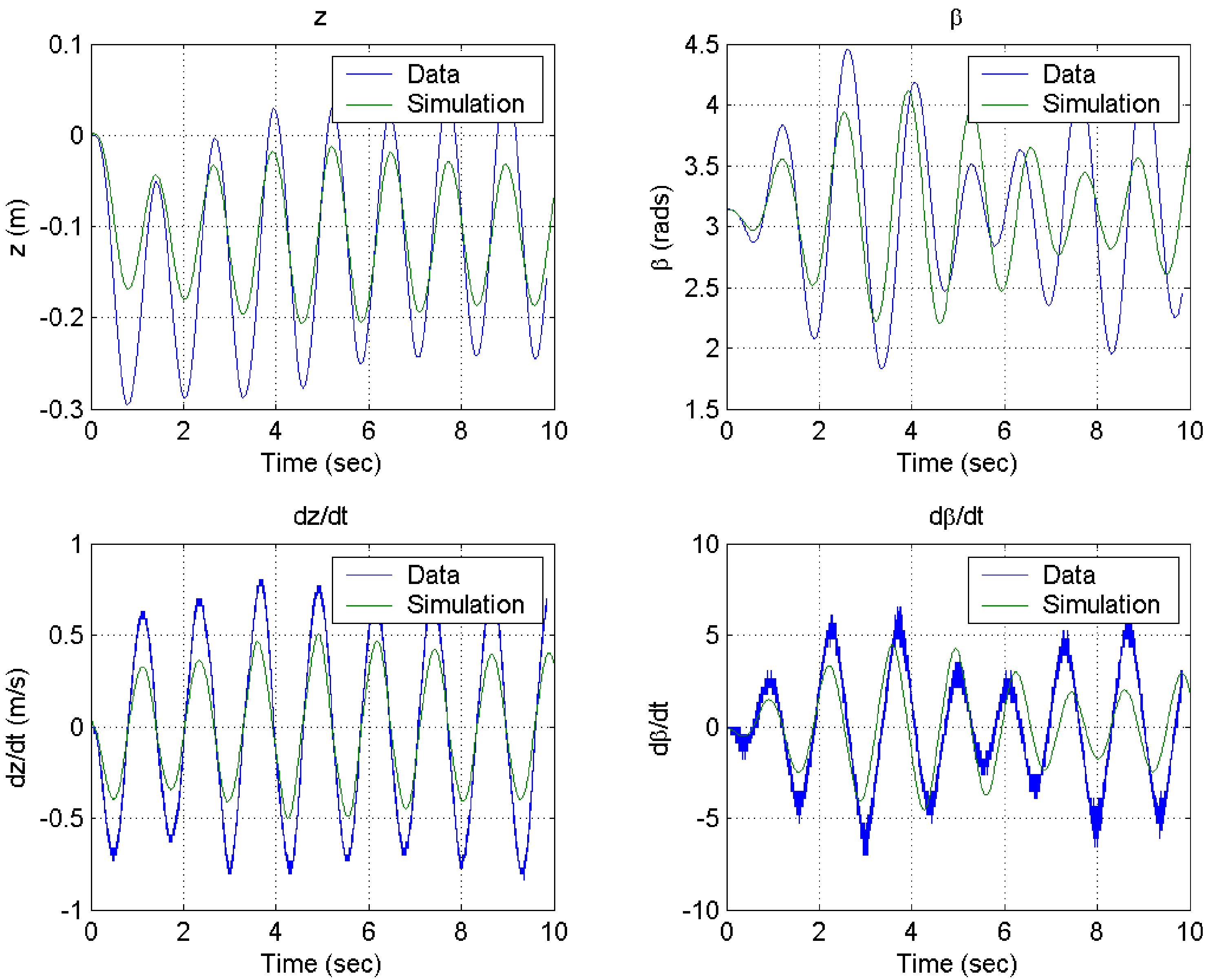


Figure 39: Actual and simulated response of system to sin wave of amplitude = 1.5 volts and frequency = 5 rad/s

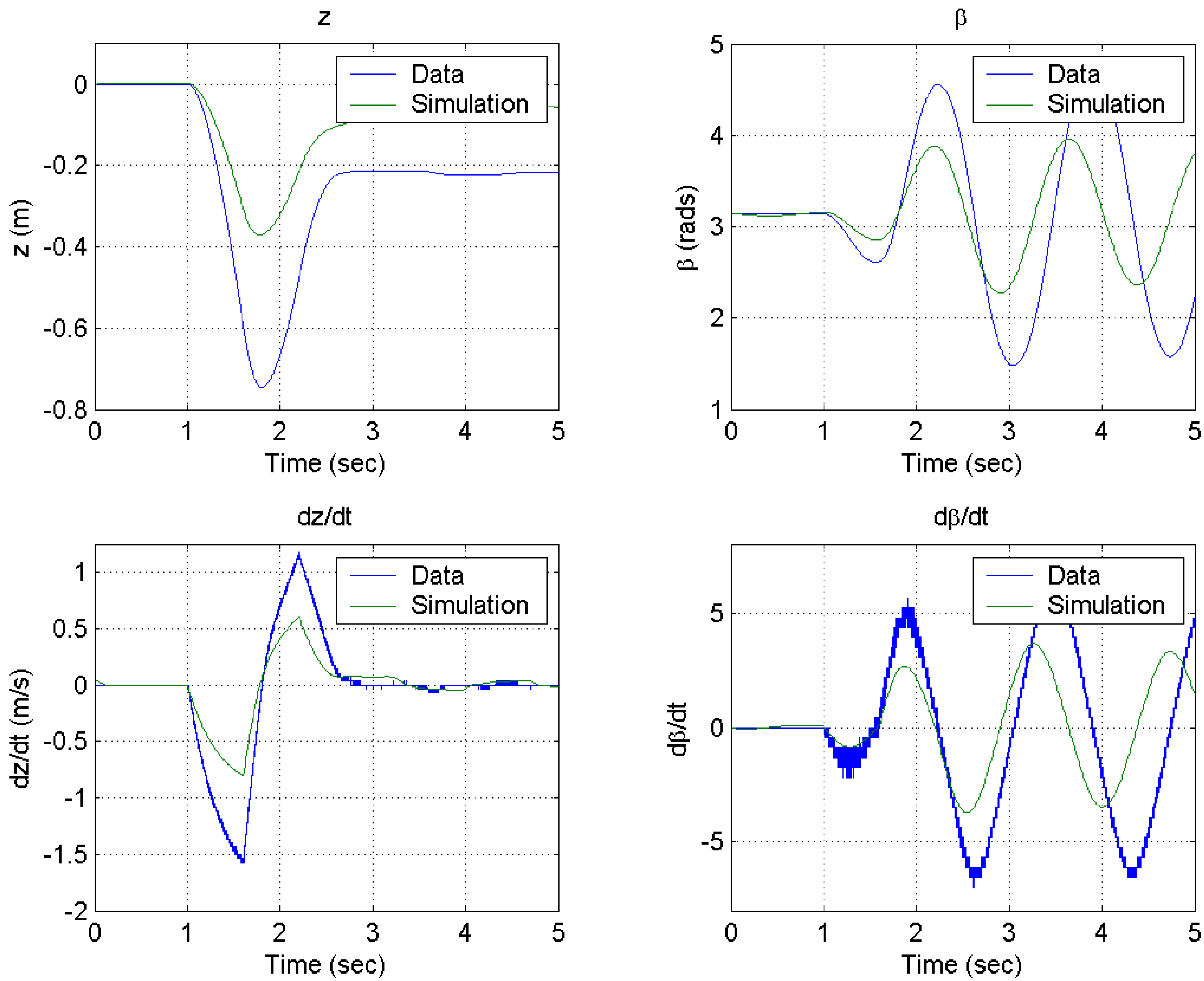


Figure 40: Actual and simulated response of system to series of steps

As can be seen, the fully non-linear model seems to model that actual system fairly well. It appears that the simulation is actually slightly stiffer than the actual model. This implies that the magnitude of the modeled friction is actually higher than reality. Also, the signal of dz/dt and dβ/dt gathered in lab appear to be much more noisy than the simulated results. This is due to the fact that the experimental values are actually estimated states. They are calculated using simple differentiation, which appears to be very noisy. However, the full non-linear simulation has these states without estimation, so the signal is smooth.

Keep in mind that the accuracy of the fully non-linear model is evaluated here. As shown above, these complex equations can be simplified in several ways. One way is to neglect the non-linear coulomb friction term. The other way is to linearize these equations about an equilibrium point. Note that the accuracy of the model should go down with each approximation made. The effects of these approximations will be evaluated in the Control Design Performance section.

XI. Control Design Performance

Classical Design Performance

The classical design incorporated the following controllers for the inner and outer loops.

Inner Loop:

$$C_{inner} = \frac{176.9\left(s^2 + 6.818s + 11.72\right)}{s(s + 39.94)}$$

Equation 33

Outer Loop:

$$C_{outer} = \frac{11.85(s + 0.5)}{(s + 28.74)(s + 5.26)}$$

Equation 34

These controllers were implemented as shown below in Figure 41. For ease of analysis each of the controllers was saved as a LTI system. This allowed the easy interchange of different control designs and allowed for a single model to be very flexible.

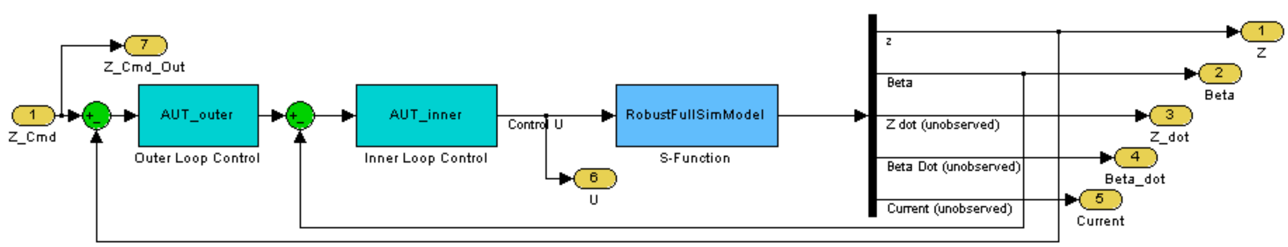


Figure 41: The Classical Control Model using LTI systems.

Because of the nature of the robustness analysis performed it was often more convenient to create specific, purpose built models that would be used for particular robustness calculations rather than create a single large, complicated system that would be able to handle all of the tasks that were needed. All of these were designed to accept the same LTI systems with common inner and outer loop controllers. Thus the inner and outer loop controllers needed to be specified only once.

The first piece of analysis was performed on the inner loop. The damping of the inner loop poles can be easily found by using the *damp* command in *Matlab*. These were presented earlier as Figure 21 and are shown again below for convenience.

Eigenvalue	Damping	Freq. (rad/s)
0.00e+000	-1.00e+000	0.00e+000
6.88e-014	-1.00e+000	6.88e-014
-3.10e+000 + 1.37e+000i	9.15e-001	3.39e+000
-3.10e+000 - 1.37e+000i	9.15e-001	3.39e+000
-1.74e+001 + 9.46e+000i	8.78e-001	1.98e+001
-1.74e+001 - 9.46e+000i	8.78e-001	1.98e+001
-3.95e+002	1.00e+000	3.95e+002

Figure 42: The Closed Loop Poles of the inner control loop. (Figure 15 Repeated)

By taking the transfer function of the inner loop it is also possible to determine the gain and phase margins for the inner loop. The results of the combined inner and outer loops will not be better than that of the inner loop and thus it is necessary that the inner loop have a very good gain and phase margin. The bode plot of the control signal for the system is shown below in Figure 43.

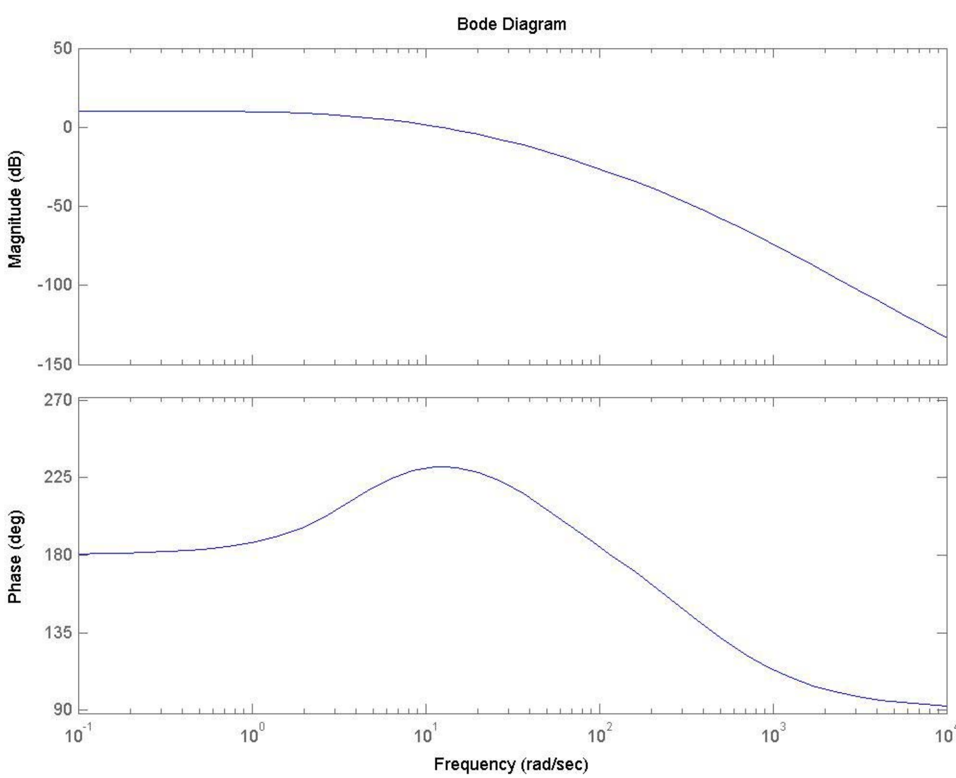


Figure 43: A Bode Diagram of the Control Loop of the Inner Loop Beta Control.

The gain and phase margins of the control loop and the β sensor feedback loop can also be calculated and these are shown below in Table 3.

Table 3: Robustness of the Classical Design Inner Loop

Loop	Gain Margin (dB)	Phase Margin (deg)	KΔ Range	Time Delay (sec)
Control	28.86	51.5	27.74	0.074
β	-10.4 & 28.8	51.5	27.74	0.074

It can be seen from the above figure that the gain and phase margins of the inner loop are very good and thus provide an excellent basis for developing the control of the z loop.

It can be seen from the eigenvalues in Figure 42 that there are two eigenvalues at zero indicating that the system is unstable. It was determined earlier using a transformation into modal form that this corresponded to an instability in the z state as the z state is unobservable from the β state. Thus the system is unstable in z. This result was confirmed in simulation as can be seen below in Figure 44 and Figure 45. Figure 44 shows the beta response using only the inner loop and Figure 45 shows the z response for this same loop. It is obvious that the instability in z that is observed in Figure 45 necessitated the implementation of an outer loop controller.

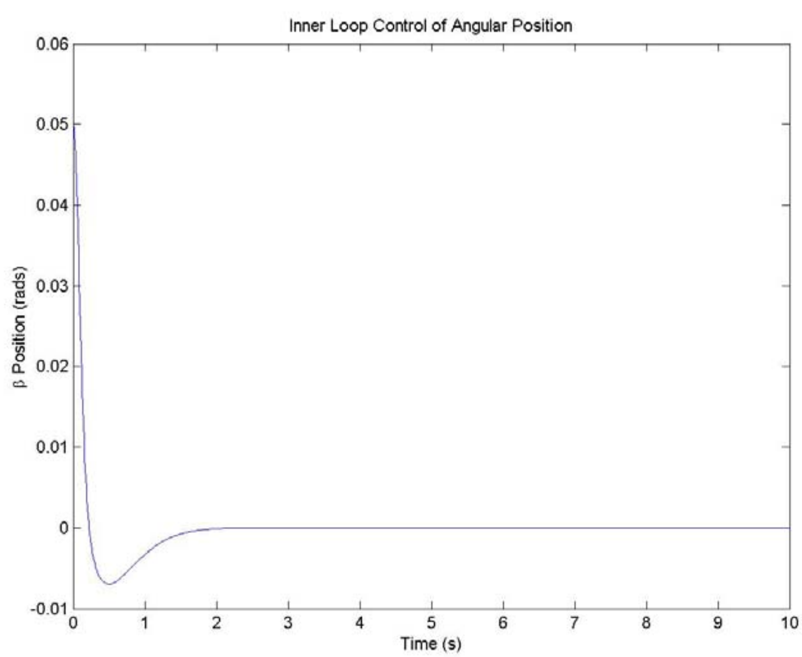


Figure 44: The Angular Position Response of the Inner Loop to an initial disturbance in β

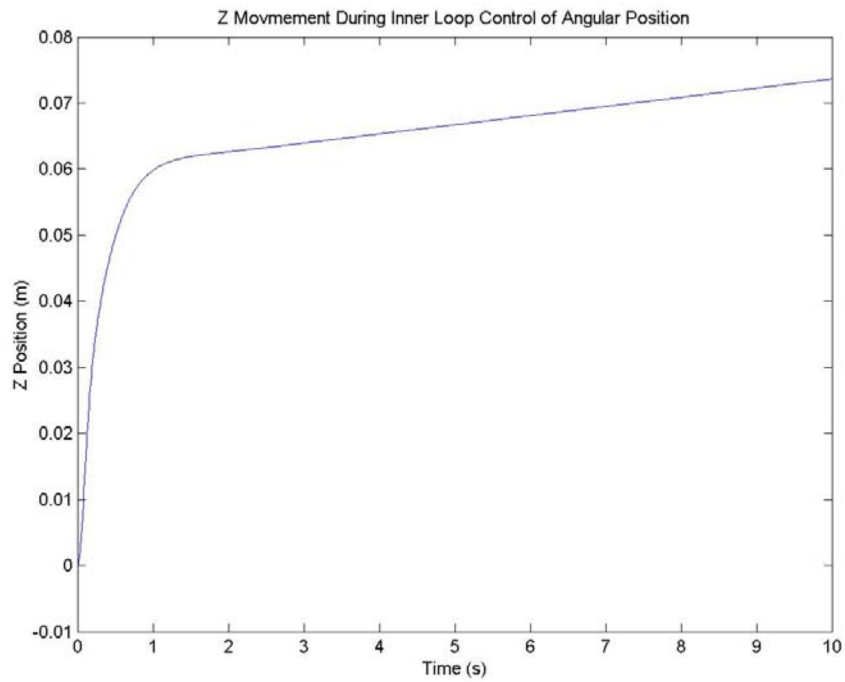


Figure 45: The z response of the Inner Loop to an initial disturbance in β.

Thus the outer control loop was designed to feed back the z position to the system and control this in conjunction with keeping the pendulum in the fully upright position. As was described earlier the *SISOtool* in *Matlab* was used to construct the outer loop controller. The controller is provided below for reference.

$$C_{outer} = \frac{11.85(s+0.5)}{(s+28.74)(s+5.26)}$$

Equation 35

The root locus of this design was presented previously in Figure 22 and Figure 23 on page 41. The closed loop poles of this design are shown below in Figure 46.

Eigenvalue	Damping	Freq. (rad/s)
-6.82e-001 + 5.48e-001i	7.79e-001	8.75e-001
-6.82e-001 - 5.48e-001i	7.79e-001	8.75e-001
-2.85e+000 + 2.33e+000i	7.75e-001	3.69e+000
-2.85e+000 - 2.33e+000i	7.75e-001	3.69e+000
-5.91e+000 + 1.48e+000i	9.70e-001	6.09e+000
-5.91e+000 - 1.48e+000i	9.70e-001	6.09e+000
-2.80e+001 + 1.35e+001i	9.00e-001	3.11e+001
-2.80e+001 - 1.35e+001i	9.00e-001	3.11e+001
-3.95e+002	1.00e+000	3.95e+002

Figure 46: The closed loop Eigenvalues and Damping of the Outer Loop (Figure 17 Repeated)

It can easily be seen that in this design all of the poles have a damping greater than 0.707 with the smallest damping being 0.779. Additionally, the largest real part of any of the eigenvalues occurs at the complex roots –0.682 +/- 0.548i which occur at 0.875 radians per second. It is obvious from this small value of the real section that the settling time of 1s was not obtained using this design. It does, however, have very strong damping.

Figure 47 below shows the response of the system to a typical step response and shows a great deal of the relevant information. This analysis was performed on the linear system.

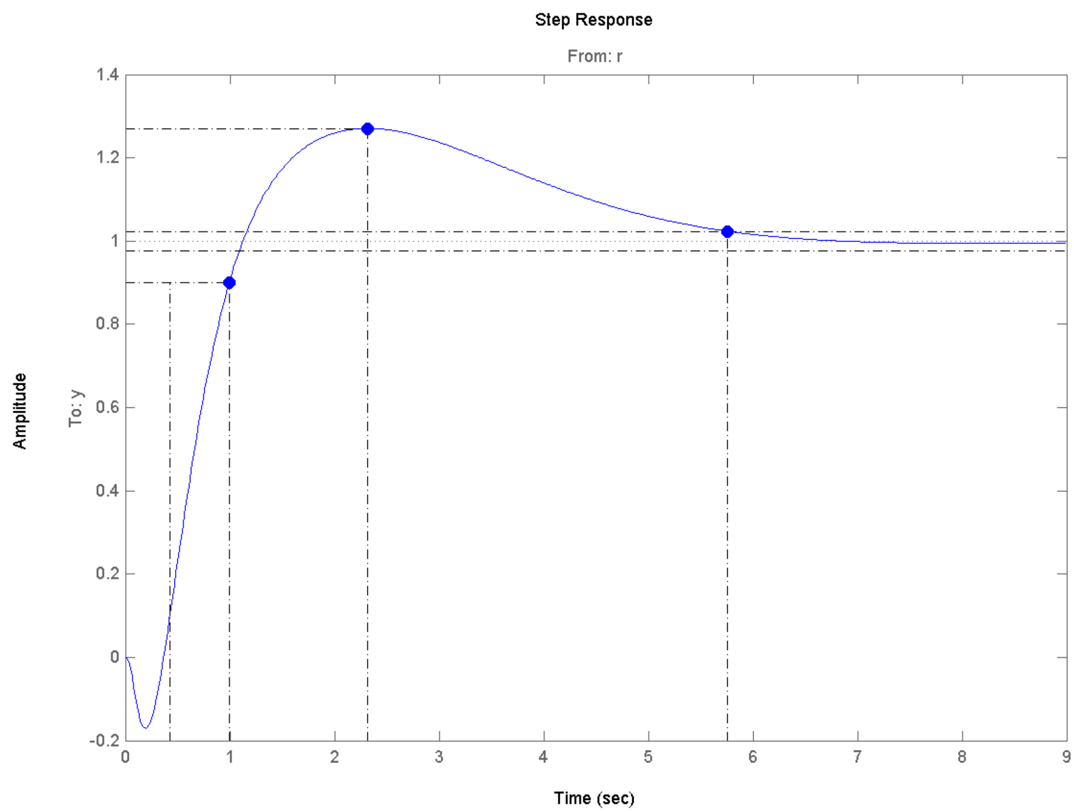


Figure 47: The step response of the Outer Loop in the linear simulation.

The performance characteristics of this design are shown below in Table 4.

Table 4: The step response performance of the Classical Design

Characteristic	Value
Overshoot	27 %
Rise Time	0.570 s
Settling Time	4.86 s
Bandwidth	3.8 radians / s

The performance characteristics presented above in Table 4 are all for the completely linear system. It is a legitimate question to ask how these performance results change with the addition of a saturation block on the control and with the addition of friction into the system. Both of these are present in the physical system on which the control will be implemented and thus must be taken into account when evaluating the control system. The response of the system to a step command of 0.2 meters is shown below for all three cases. The controller and system for each of these tests is the exact same but for the addition of a saturation block in the linear with saturation case and the addition of friction and a saturation block in the non-linear case. The resulting plots are shown below in Figure 48 and Figure 49.

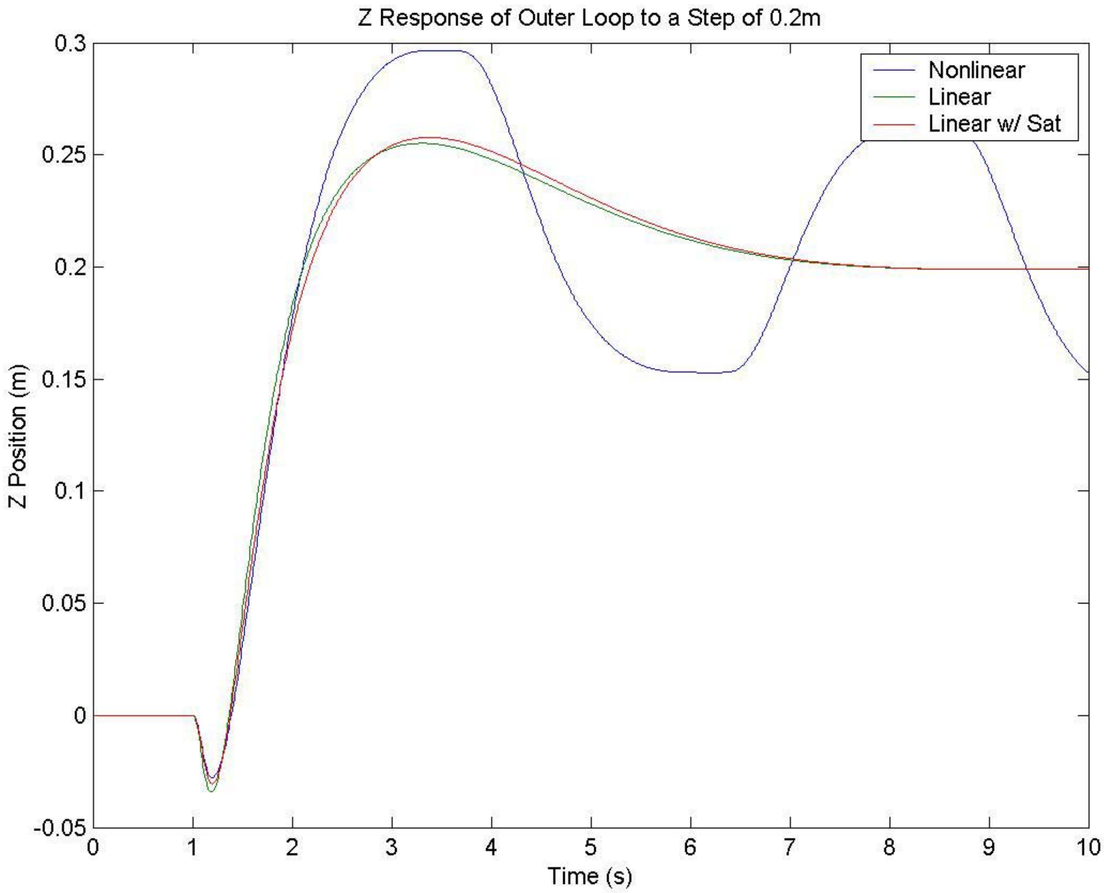


Figure 48: Z response of the Classical Control System to a step input of 0.2 m

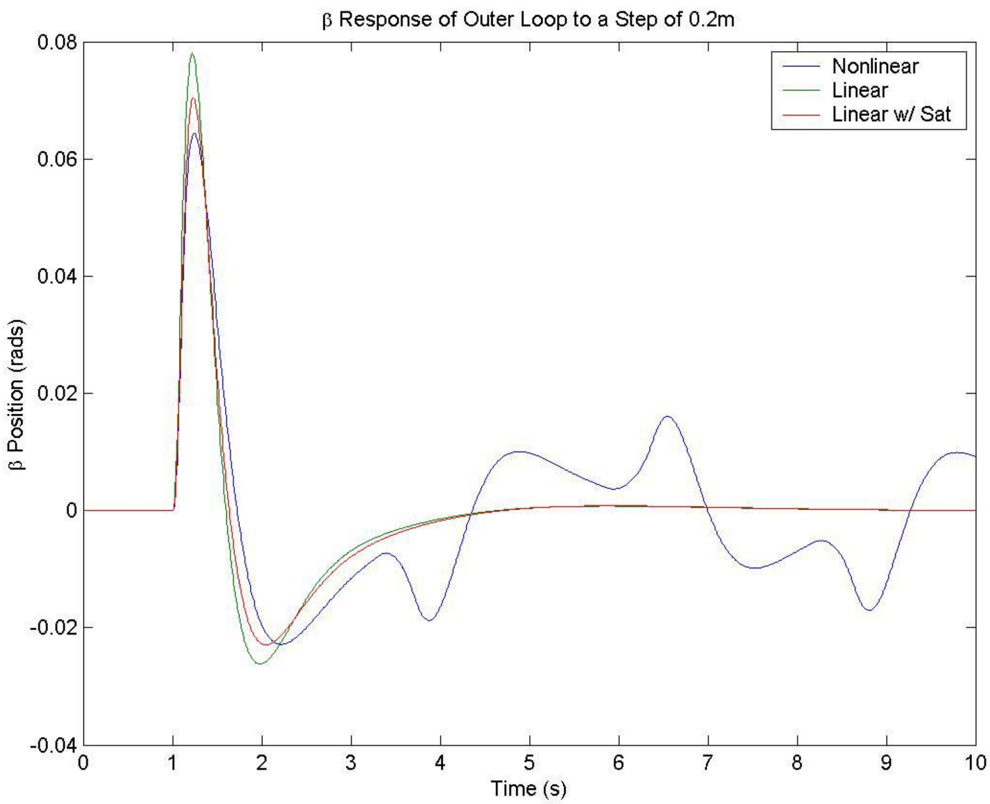


Figure 49: Beta response of the Classical Control System to a step input of 0.2 m

As can be seen in Figure 48 and Figure 49 above the addition of a saturation block on the control does little to affect the response of the systems and the performance characteristics are almost identical. The reason for this can be seen below in Figure 50 where it is observed that the control signal spends the vast majority of its time within the saturation limits and thus the control signal is not greatly affected by its addition.

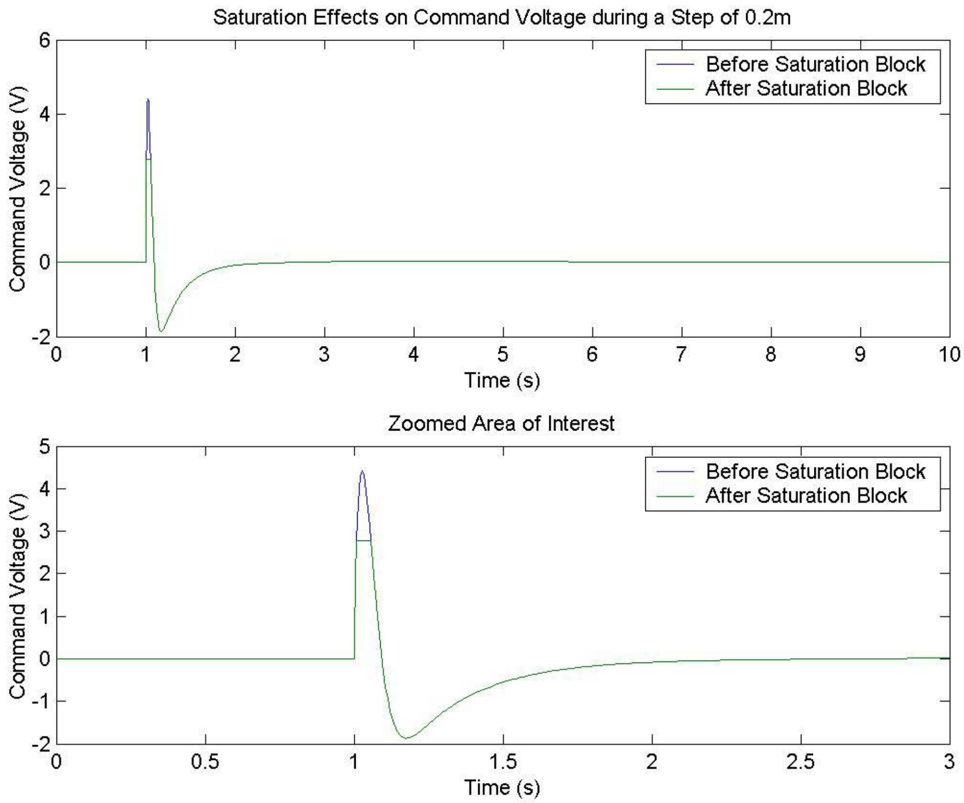


Figure 50: Effects of Saturation Block on the control Signal during a 0.2 m step.

By looking at Figure 48 it is also evident that when the full non-linear model is used the system quickly goes into a steady state oscillation. This oscillation makes the calculation of settling time very difficult. While it could be argued that the settling time is that amount of time that it takes the system to reach the steady state oscillation this would be artificially making the settling time much smaller as the system begins to oscillate almost immediately after rising. Thus it is better said that the settling time is undermined in this case. It is evident, however, that in the full non-linear simulation the overshoot of the system doubles to nearly 50 % while the rise time remains largely the same.

These experimental results can be compared to actual data that was obtained from implementing this control system on the Linear Track Cart Pendulum. Once again as step of 0.2 m was used as a reference point. Figure 51 below shows the position response of the cart to this step.

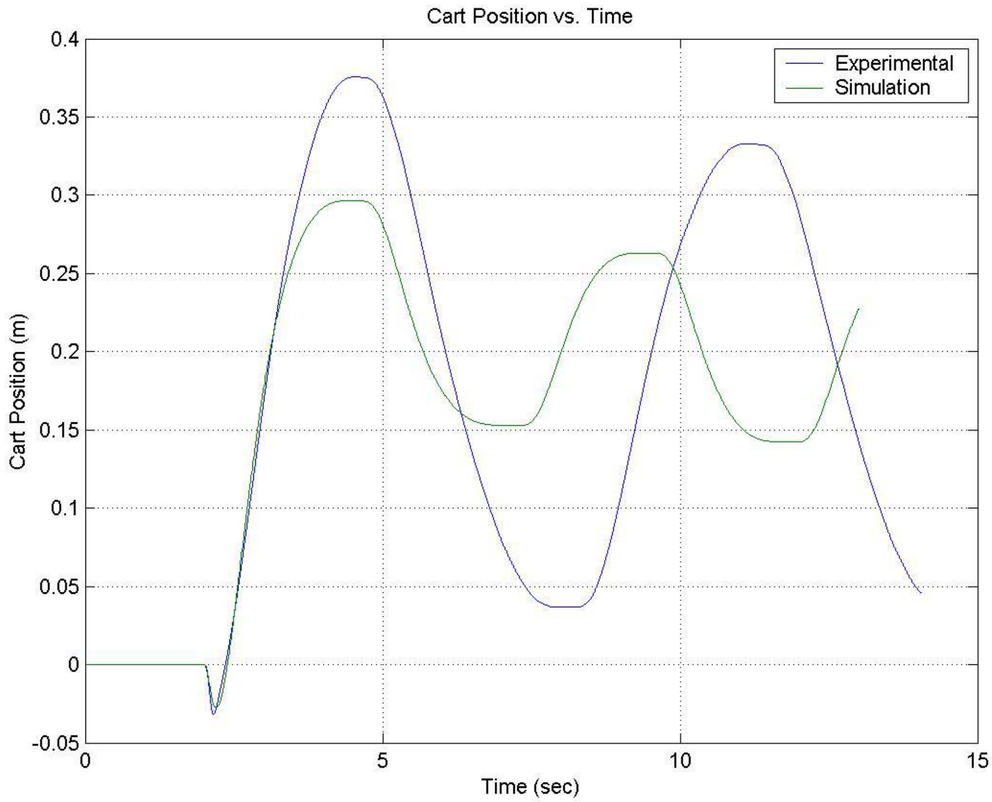


Figure 51: Simulated Position Response and Actual Position Response to a 0.2 m step.

By examining Figure 51 two things become immediately apparent. The first is that the percent overshoot in the real system is much greater than was predicted in the simulation. This has a great deal to do with the manner in which the system was modeled. It was also noticed that while results on the system were the same if the experiments were conducted on the same day there was significant variation between trails taken on different days.

It is also evident in Figure 51 that the frequency of oscillation of the system is also significantly different in the simulation than it is in the real system. For further insight into why this might be the case it is worthwhile to look at the response of β during this same test. This data is shown below in Figure 52.

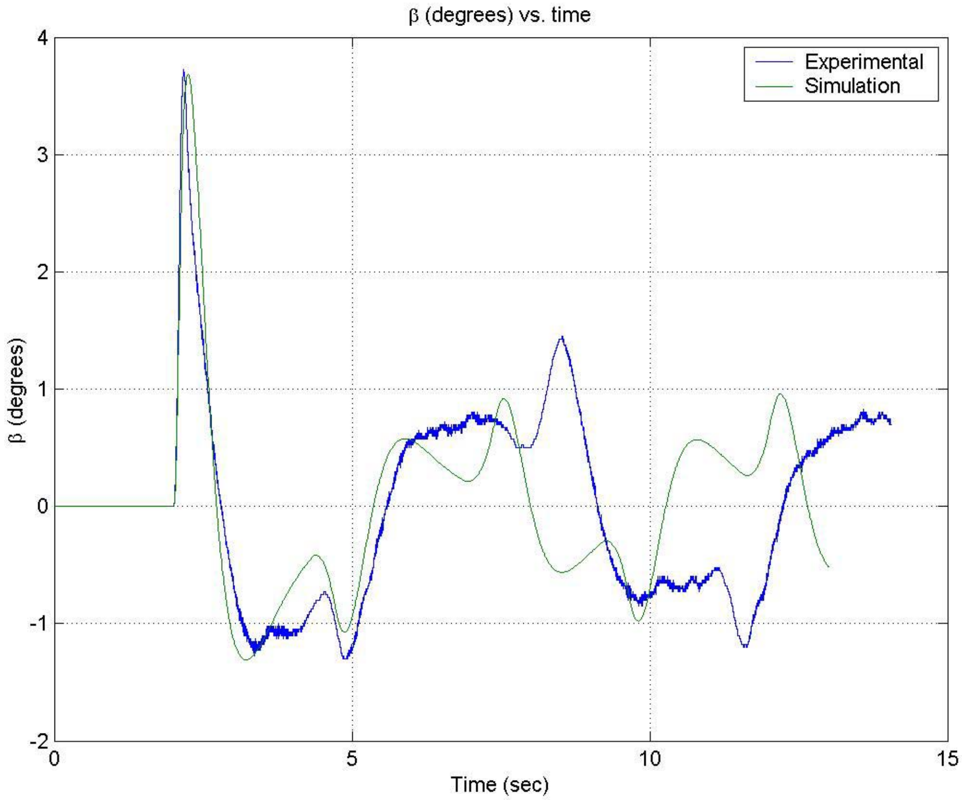


Figure 52: Simulated Angular Response and Actual Angular Position Response to a 0.2 m step.

In Figure 52 above it is evident that the same general pattern is observed for both the simulated and the actual case. This graph also gives clues as to why the frequency of oscillation of the two cases is different as was observed in Figure 51. In the experimental data it is observed that the value of β slowly increases and almost plateaus at the beginning of each cycle. In Figure 52 above this occurs between 6s and 7s, between 9s and 12s. These regions of interest are shown explicitly in Figure 53 below.

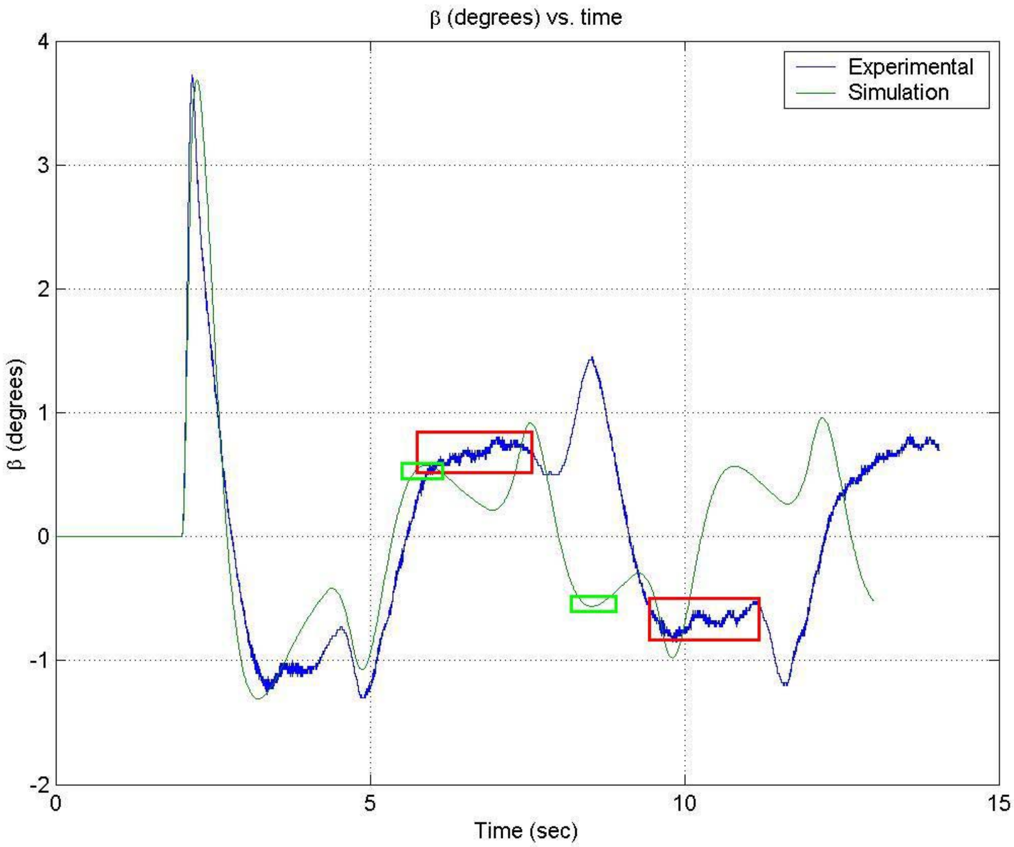


Figure 53: Simulated Angular Response and Actual Angular Position Response to a 0.2 m step with highlighted areas of interest.

In Figure 53 above the green areas represent the times when the cart is accelerating in an attempt to prevent the pendulum from falling. They also represent times when the velocity of the pendulum is at or near zero radians per second. In the simulation, shown above with the dark green line and the green boxes, these periods are relatively short and change quickly. In the actual response, in blue with red boxes, these periods are extended. This leads to the conclusion that while the stiction on the cart was negligible and could be ignored, the stiction on the pendulum is in fact significant.

The increase in pendulum stiction in the actual system verses the simulated system means that the pendulum is spends a greater amount of time away from the perfectly vertical position. It also means that the cart is forced to move a greater distance during the limit cycle. This is caused by the fact that the pendulum is largely unaffected until the force on the pendulum exceeds that posed by stiction. By extending the testing that was done on the cart to the pendulum and accurately characterizing its stiction component a more accurate simulation could be determined.

Largely because of these effects it was observed that the limit cycle of the system increased from 11 cm to 29 cm. This can easily be seen in Figure 51 on page 85. This sort of limit cycle oscillation was considered to be unacceptable. It was thus decided to make use of some of the gain margin and increase the gain on the Z tracking outer loop in the laboratory. Simultaneously, it was decided to prove that the system could take a –6dB variation in the gain of this loop. The change in performance in the system with the variations in gain is shown below in Figure 54.

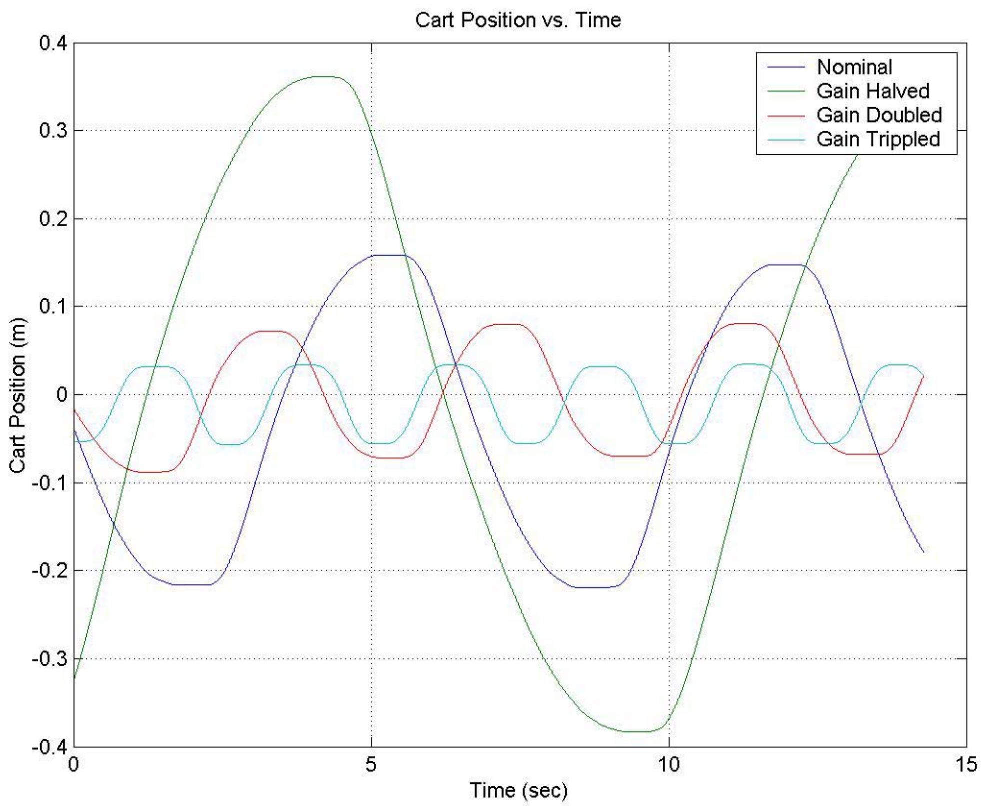


Figure 54: The variation of peak - peak distance with increasing gain.

It can be seen in experimental data presented in Figure 54 above, the peak to peak distance is can be significantly reduced if the gain is increased. Doubling the gain on the z control loop decreases the peak-to-peak variation from 29 cm to 15 cm. This can be further decreased to 8.8 cm if the gain is tripled from the nominal. It was observed in the laboratory that the robustness of the design suffered significantly when the gain was increased though the performance was increased. Thus it was decided to maintain the original gains and have a very large amount of robustness rather than sacrificing this robustness for performance. This proved prudent as the physical system changed significantly over the course of the quarter and many of the high performance, low robustness designs that worked early in the quarter failed later in the quarter due to changes in the plant due to wear and tear on the system.

A more intensive analysis of the gain and phase sensitivities of the complete classical control system was also performed in order to determine the robustness of the system. The robustness of the control loop was determined by cutting the control after it has emanated from the controller and before it had entered the plant. The model that was used is shown below in Figure 55.

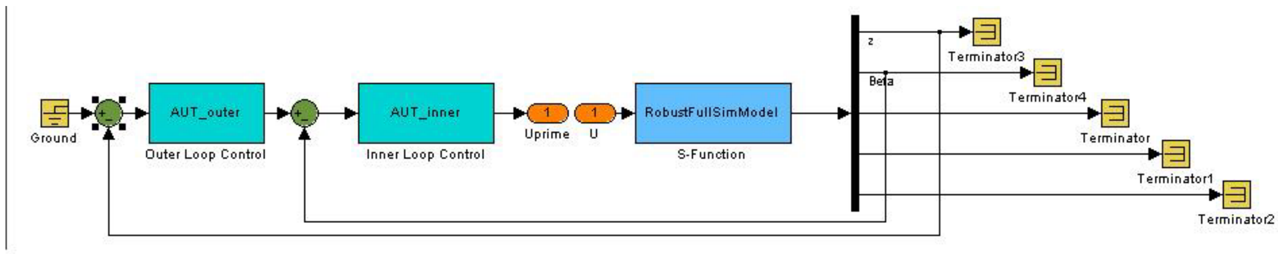


Figure 55: The Model used to evaluate the robustness of the classical control loop.

The linear model of this system was obtained using the *Matlab* linmod function. From this state space model the transfer function from U to U_{prime} was determined. From the transfer function the robustness of the loop can be determined. The results of the analysis are shown below in Table 5.

Table 5: Robustness of the Control Loop for the Classical Design

Loop	Gain Margin (dB)	Phase Margin (deg)	KΔ Range	Time Delay (sec)
Control	-6.45 & 30.36	48.06	0.4757 to 32.99	0.0897

It can be seen from the table above that this system is very robust with respect to the control signal with significant gain and phase margins. A similar analysis can be performed on each of the sensor loops. By cutting the sensor wire in *Simulink* it is possible to determine variations in the sensor signals that are possible before the system goes unstable. Figure 56 shows the model that was used to determine the allowable variation in β. A similar model was created to determine the allowable variation in the position feedback.

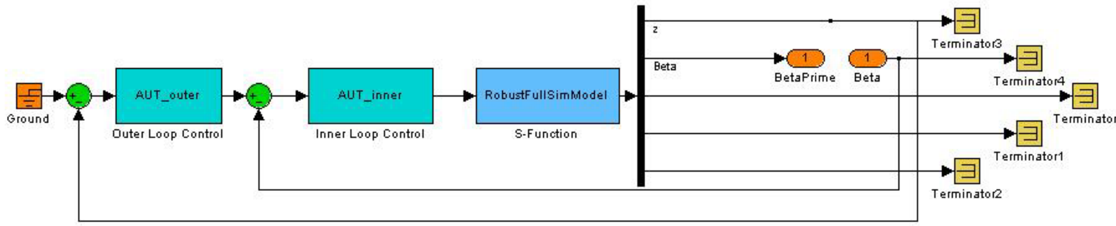


Figure 56: The model used to determine the robustness in the Beta sensors signal.

The robustness for both of the sensor loops is shown below in Table 6.

Table 6: Robustness of the Sensor Loops for the Classical Design

Loop	**Gain Margin (dB)**	**Phase Margin (deg)**	**KΔ Range**	**Time Delay (sec)**
β sensor loop	-4.43 & 28.92	-35.27 (lead) & 41.3 (lag)	0.6003 to 27.93	0.0693 (lag)
Z sensor loop	-∞ to 8.86	47.6	0 to 2.774	0.6482

It is observed in Table 6 that if the gain in the Z sensor loop is decreased to zero the system is still stable. This is valid since if the z loop is eliminated it is effectively the same as having only the inner β loop working. This will maintain the pendulum in a stable upright position. The position, however, will be an uncontrolled state.

Another measure of the robustness of the system is the ability to reject a constant force applied. In this case such a force was applied by using a push pull gage on the actual system. First, this was done in simulation and it was discovered that the maximum force that the system could withstand was 10.9 N or 2.45 lbf. It was also observed that the manner in which the force was applied also affected the ability of the system to resist it. If a step force was applied then the results were highly dependent on the location of the pendulum at the time of application. To truly test the steady state ability of the system to resist an external force the force was gradually applied using a prefilter. This reflected the manner in which the force was allied in the laboratory. Figure 57 below shows the model that was used to determine the maximum force that could be applied.

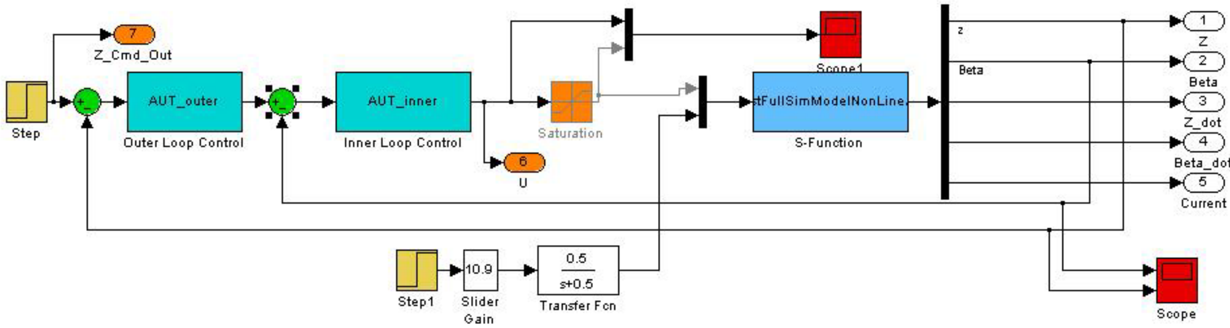


Figure 57: The model used to apply a constant force to the classical system.

Note that in Figure 57 above the disturbance force step command is sent through a rather slow filter before entering the model. Figure 58 below shows the simulated and experimental control voltage results for a disturbance force of 2 lbf or 8.9 N. These results show that the predicted command voltage is much higher than that which was observed in the laboratory. The consequence of this is that the laboratory system can withstand a much greater force than the 10.9 N predicted by the simulation.

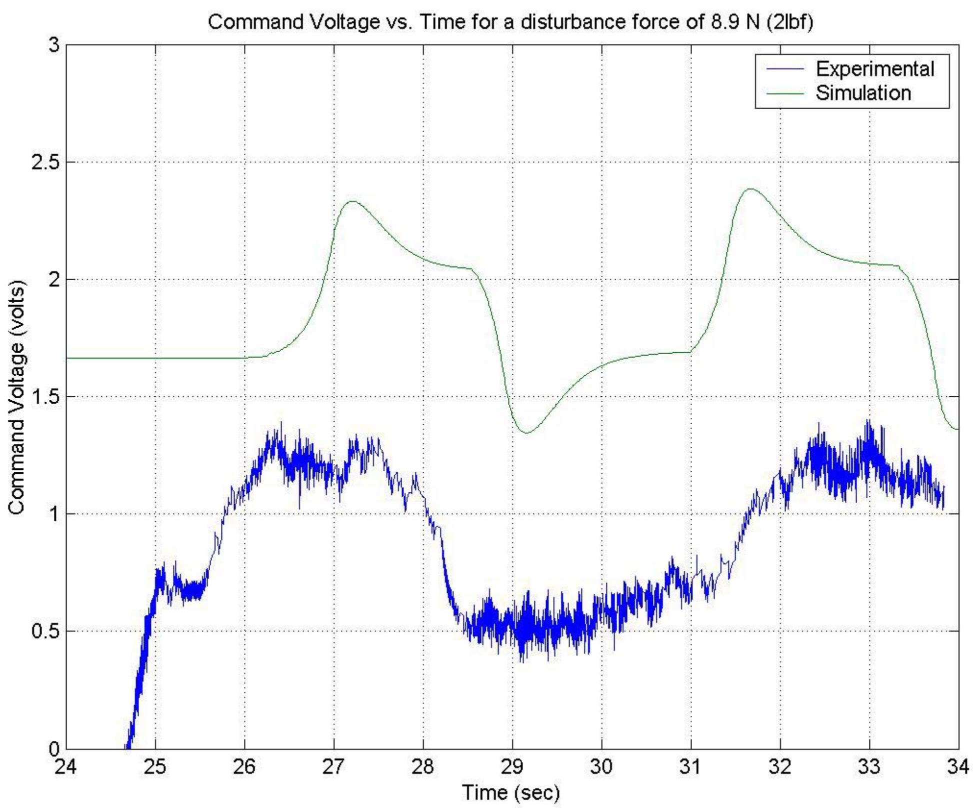


Figure 58: Control Voltage for the classical controller during the rejection of a 8.9 N (2lbf) force.

The final piece of analysis that was done on the Classical controller was the mapping of the area of attraction of the controller in terms of both the position and the velocity domains. An automated code was developed to run the simulation over a grid of possible initial conditions. A each step it was determined if the simulation was able to recover from the initial condition and move into a stable orientation or if it was unable to recover and allowed the pendulum to fall. For the position domain, analysis was done with both the fully nonlinear simulation and with the linear simulation with a saturation block. Performing this analysis on the linear model without the saturation block would be trivial as the system is able to recover from any set of initial conditions in position space in the purely linear case. The results of the mapping of position space are shown below in Figure 59. Likewise, the results of the mapping of velocity space are also shown below in Figure 60.

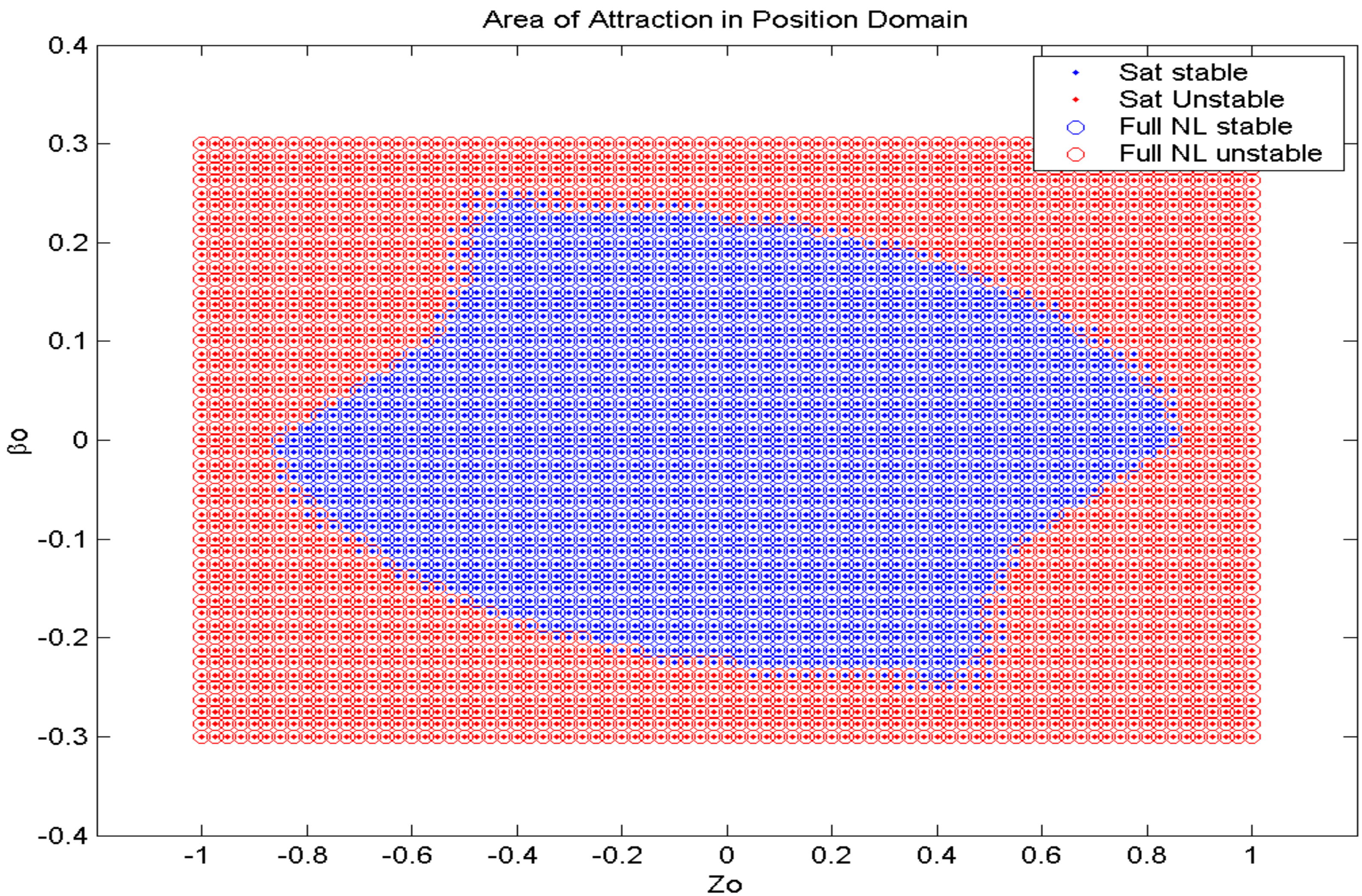


Figure 59: The Area of Attraction of the Classical Controller in the Position Domain.

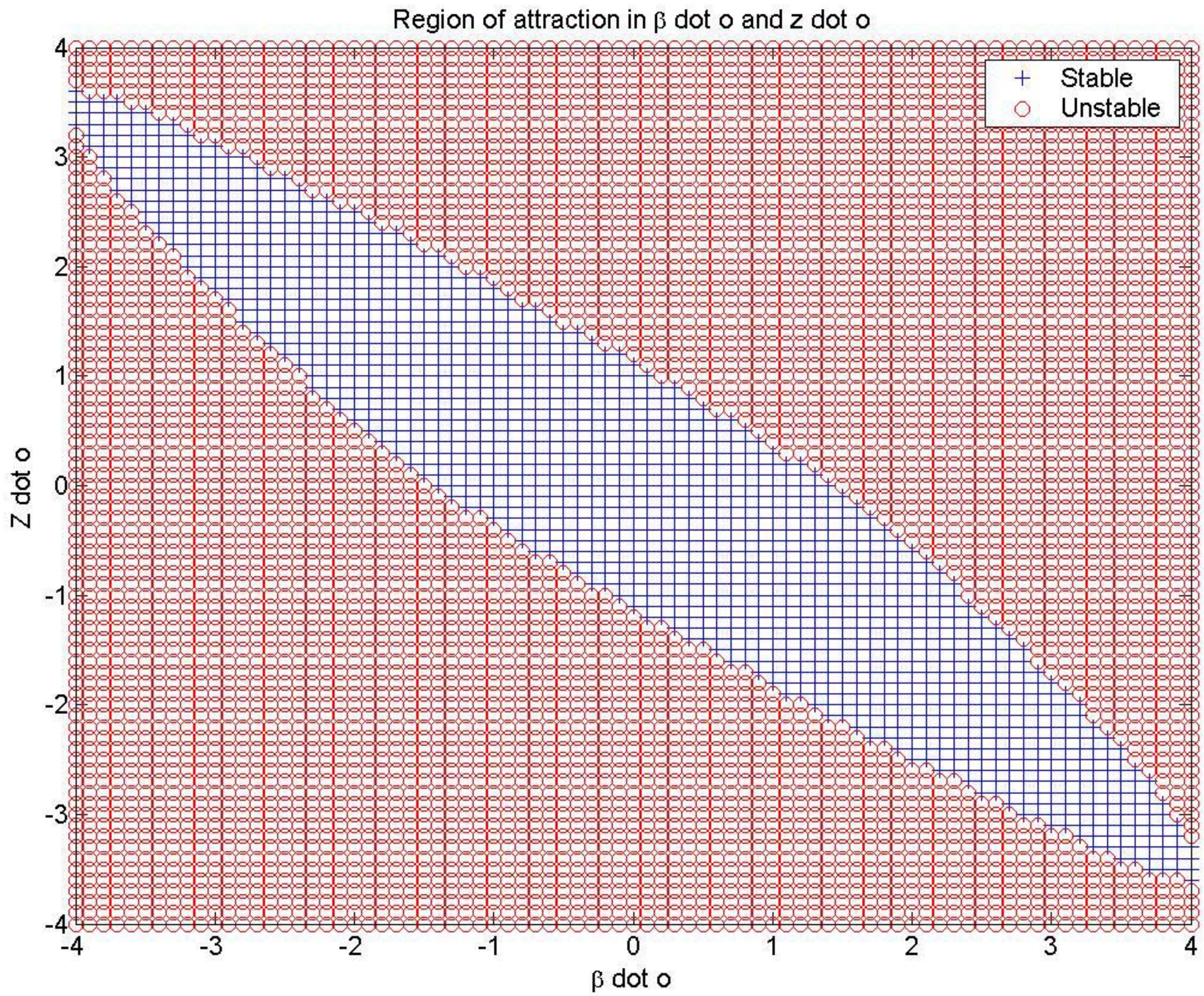


Figure 60: The Area of Attraction of the Classical Controller in the Velocity Domain.

As can be seen in Figure 59 and Figure 60 the area of attractions reflect the physics of the system. In Figure 59 it is evident that the space of solutions is symmetric along the diagonal axis. This reflects the symmetry evident in the system. A positive z displacement with a positive β will behave in a similar manner to a negative z and a negative β. It is also interesting to note in this graph that the maximum initial β displacement that can be endured occurs when the z displacement is non-zero. This makes intuitive sense as an error in z in the correct direction can actually help the system in the recovery of the pendulum. Conversely, in the opposite fashion, a z displacement in the wrong direction can limit the ability of the system to recover the pendulum from an initial β offset.

In Figure 59 both the full non-linear and the linear model with a saturation block are plotted. The linear system with saturation is plotted as dots and the full nonlinear system is plotted as circles. It is evident from the two sets of data that the addition of the friction terms does not significantly affect the catch basin of the controller. From this information it ca be deduced that if the size of the area of attraction could be increased by increasing the saturation level of the motor.

Looking at Figure 60 it is evident that the velocity domain has a very distinct pattern wherein the system is able to recover from a significant angular velocity only if this is combined with a significant negative cart velocity. Once again this is reflected along the diagonal axis. Table 7 below shows the maximum initial displacements that the controller can recover from.

Table 7: The Range of Stable Initial Conditions for the Classical Controller

Model	**Maximum Z(0) If β = 0**	**Maximum β(0) If Z = 0**	**Largest Possible β(0) if Z(0) ≠0**	**Largest Possible Z(0) if β(0) ≠0**
Linear with Saturation	+/- 0.825 m	+/- 0.225 rad	0.25 rad where $-0.475 \leq Z(0) \leq -0.325$	0.85 m where $0.0125 \leq \beta(0) \leq 0.05$
Fully Nonlinear	+/- 0.85 m	+/- 0.2125 rad	0.2375 rad where $-0.425 \leq Z(0) \leq -0.375$	0.85 m where $0 \leq \beta(0) \leq 0.0125$

Full-State Feedback Performance

As stated in the Control System Design section under Full State Feedback Design, the gains designed are given by

```
K_matrix =

      25.173       64.197       18.528       14.354    -0.014191
```

The *Simulink* model used to simulate the response of the system with the full-state controller and to evaluate the performance and robustness is shown on the next page as Figure 61.

Equation 2 (non-linear system dynamics including coulomb friction) is programmed into the blue S-Function block. The gains on the states are contained in the matrix gain block with the last element (corresponding to k_5) set to zero. This models the fact that in the lab, x_5 is not measurable. The switches and delta blocks elsewhere are used for robustness analysis and will be discussed shortly.

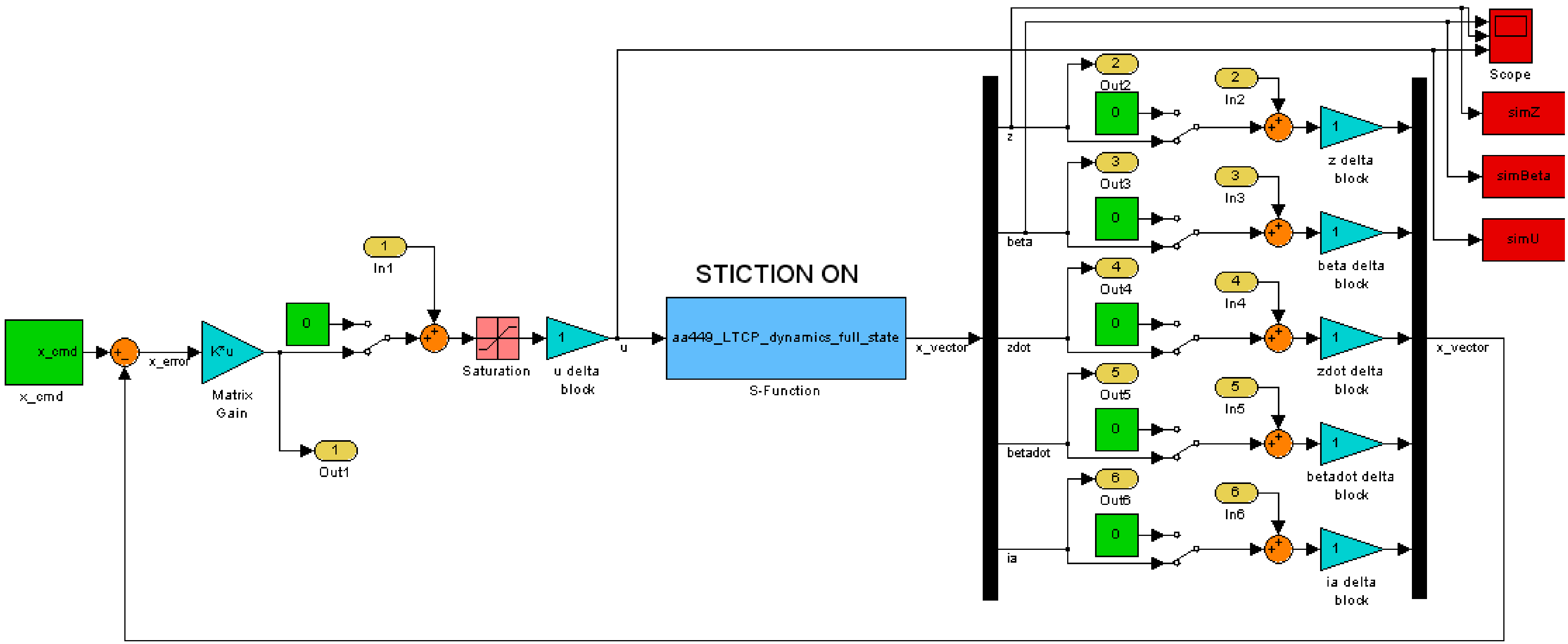


Figure 61: *Simulink* model with full-state feedback controller

The first thing to evaluate with this control design is the stability of the system. Keep in mind that the set of full-state gains were designed using the linearized model (coulomb friction and higher order terms such as $x_4^2 = 0$), therefore, these set of gains may not guarantee stability with the non-linear model that is shown above in Figure 61. However, in order to evaluate the damping and max real part of the eigenvalues, a linear system is required. As stated in Equation 19, the closed loop poles of the system are given by λ(A – BK). The damping of the system can also be evaluated using the simple damp command.

```
>> damp(A - B*K_matrix)

        Eigenvalue              Damping     Freq. (rad/s)

 -4.08e+000                    1.00e+000      4.08e+000
 -4.09e+000 + 3.72e+000i       7.39e-001      5.53e+000
 -4.09e+000 - 3.72e+000i       7.39e-001      5.53e+000
 -6.43e+000                    1.00e+000      6.43e+000
 -3.77e+002                    1.00e+000      3.77e+002
```

As can be seen, the actual closed loop poles are not exactly in the desired locations. This discrepancy exists due to the fact that k_5 was set to zero. However, the closed loop poles did not vary far from the desired poles so the design is deemed acceptable.

Furthermore, the largest real parts of the poles are at –4.09 and the minimum damping is 0.739, so the design should meet the settling time requirements.

The model can be analyzed using three different models of the plant: the full non-linear model with coulomb friction on, the non-linear model with coulomb friction off, and the linearized model.

Before evaluating the performance of the system in response to command inputs, the response of the system to initial conditions other than the equilibrium point can be evaluated. The results are shown below in Table 8.

Table 8: Response of system to initial conditions using full state feedback controller

	Actual System	**Full Non-Linear**	**Non-Linear w/o Coulomb Friction**	**Linear**
z(0) (m)	N/A	0.368	0.371	N/A
β(0) (deg)	N/A	9.65	9.79	N/A
Limit cycle (peak to peak)	+0.05/-0.02	+/- 0.01	None	None

As can be seen, the range of initial conditions that the system can withstand increases as more approximations are made. The most sensitive is the actual physical system whereas the most insensitive is the fully linear model. The linear model has no range of initial conditions because it is a linear system with no saturation. Therefore, it will be able to recover from any initial condition.

Now that the response of the system to initial conditions is known, the initial conditions can be set to roughly zero and the response of the system to a step input can be evaluated. The response of the system to a step in z of magnitude 0.2 meters using the fully nonlinear model is shown below Figure 62.

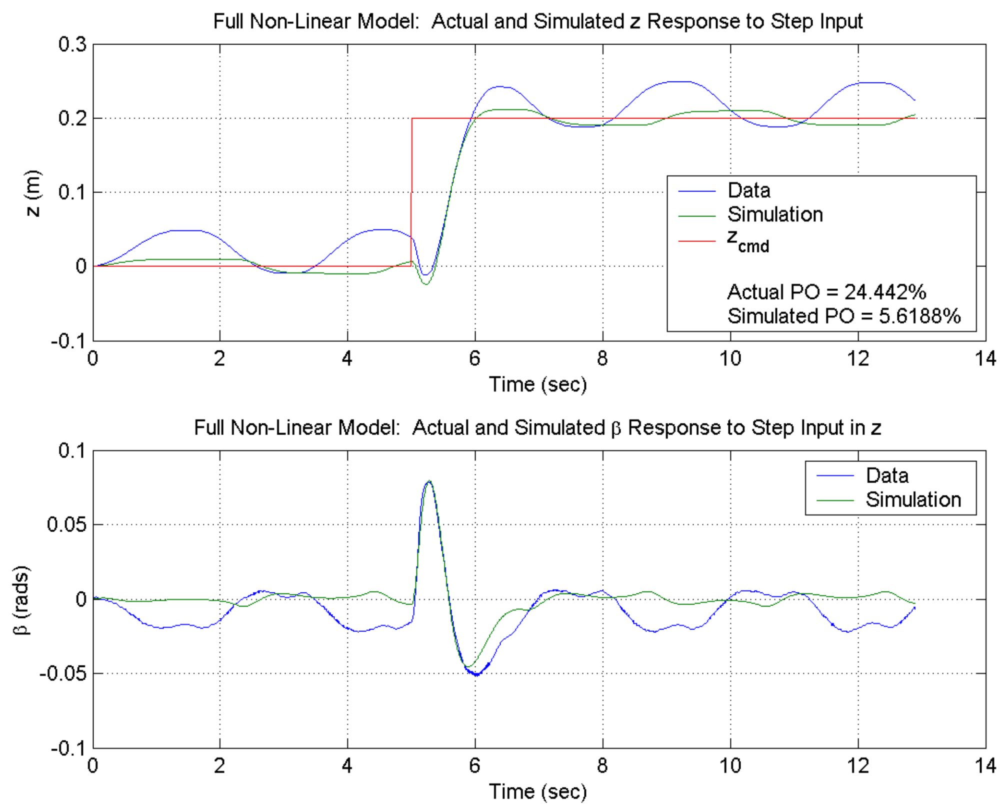


Figure 62: Actual and simulated z and β responses to a step input of z = 0.2 using full non-linear model

As can be seen, the fully non-linear model with coulomb friction turned on models the actual system with a fair degree of accuracy. The model exhibits the same limit cycle behavior where it is unable to obtain a zero steady state error. This is due to the fact that the control cannot overcome the coulomb friction with a small error. The gain can be increased to attempt to solve this problem.

Also note that the model is not exactly the same as the data, this is due to several reasons. In simulation, the model is required to have an initial condition of dz/dt = 0.01 m/s. If the initial condition is dz/dt = 0, *Matlab* has a problem attempting to derive the coulomb friction force. Setting dz/dt at t = 0 to 0.01 m/s solves this problem. However, in lab, it is very difficult to give the system these same initial conditions. Therefore, the simulation and the data may not match well initially. This is most evident in the response of β.

The response of the system to a step in z of magnitude 0.2 meters using the nonlinear model with coulomb friction off is shown below in Figure 63.

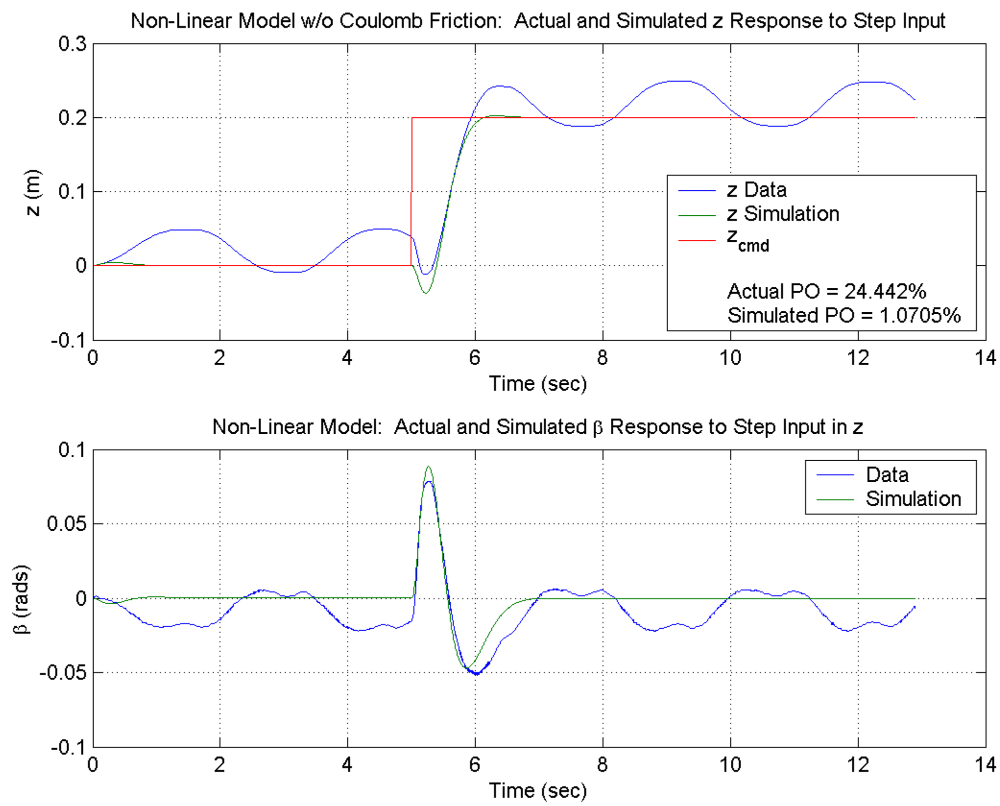


Figure 63: Actual and simulated z and β responses to a step input of z = 0.2 using non-linear model w/o coulomb friction

Once again, the model without the coulomb friction appears to model the system fairly well. The dynamics while the system is in transition is almost identical. However, the most significant change is that the simulation no longer predicts the limit cycles. Instead, there is a simulated zero steady state error and a much smaller simulated percent overshoot. As stated above, the reasons the limit cycles appear is due to the non-linear coulomb friction, and since this effect was removed, then the limit cycles disappear.

Lastly, the response of the system to a step in z of magnitude 0.2 meters using the linearized model is shown below in Figure 64.

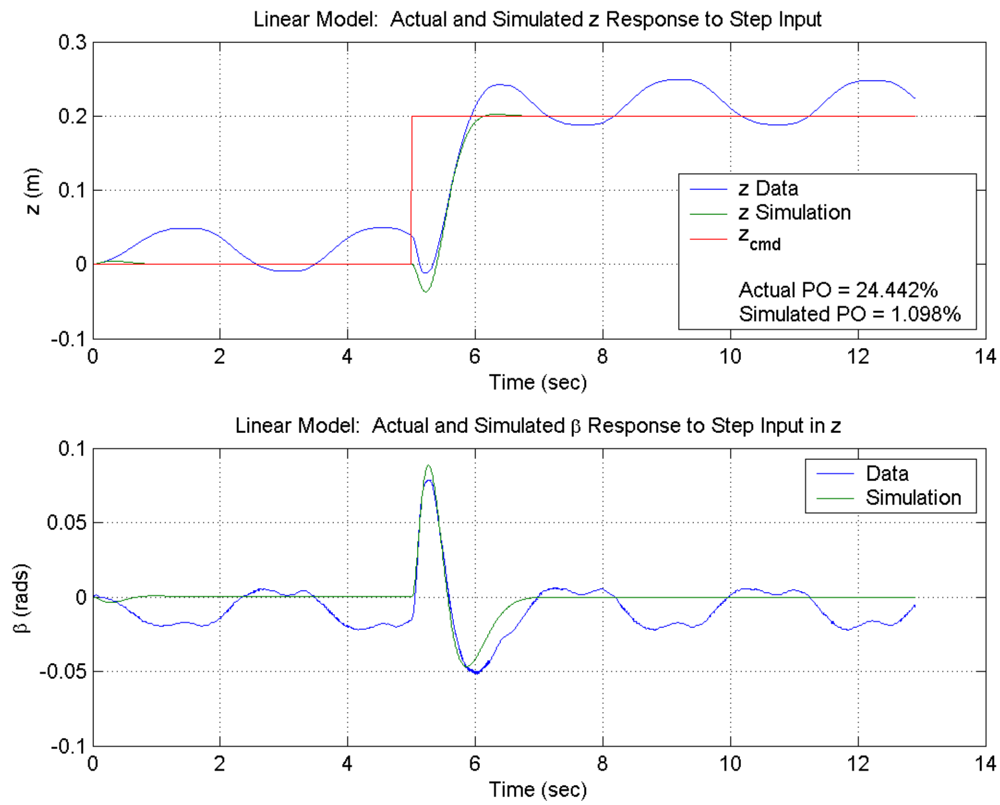


Figure 64: Actual and simulated z and β responses to a step input of z = 0.2 using fully linearized model.

As can be seen above by comparing Figure 63 and Figure 64, the non-linear system without coulomb friction and the fully linear model behave in a very similar fashion. This is because for small perturbations, the linearized model should model the actual system very well. However, the two systems can deviate if there are large perturbations. In that situation, the non-linear model without coulomb friction would provide a more accurate representation since it would take things like centripetal acceleration into account.

In addition to the percent overshoot, the settling time and rise time of the three different models can be evaluated using the 2% criterion for the settling time and the 10% to 90% criterion for the rise time. A sample plot showing how the settling time is calculated is shown below in Figure 65.

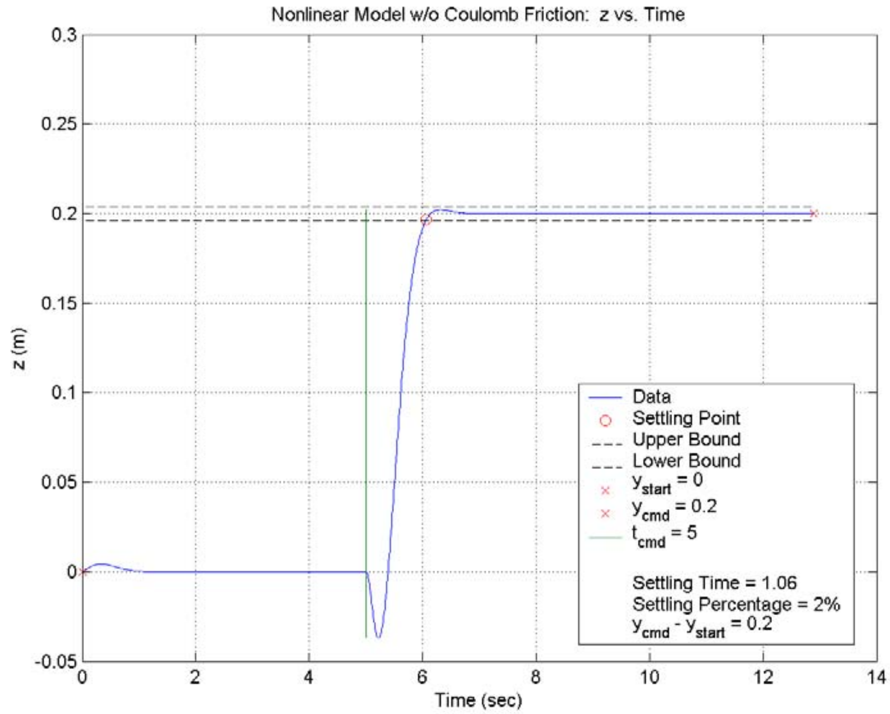


Figure 65: Settling time analysis for non-linear model w/o coulomb friction

The performance characteristics of the full state controller with the different types of models are shown below in Table 9.

Table 9: Performance characteristics of full state controller with various simulation models

	Actual System	**Full Non-Linear**	**Non-Linear w/o Coulomb Friction**	**Linear**
Percent Overshoot (%)	24.442	5.618	1.071	1.098
Settling Time (sec)	INF	INF	1.06	1.06
Rise Time (sec)	0.849	0.45	0.46	0.47
Bandwidth (rad/s)	N/A	N/A	N/A	6.1

As can be seen from Table 9, the different models predict different responses. Obviously, the full non-linear model with coulomb friction appears to be the most accurate. The settling time is undefined due to the limit cycles, which do not decay, and therefore the system never becomes settled. It is also notable that for the non-linear models, the percent overshoot and settling time will change depending on the magnitude of the step, but the response of the linear model is the same for all magnitudes of step inputs (as long as saturation does not occur).

Also, notice that the bandwidth of the system can only be evaluated in simulation using the linear model. The actual system is highly non-linear and requires the use of the DSA in order to evaluate its frequency response. This is done below in the LQR with Integrator Performance section.

It is possible to evaluate the bandwidth in simulation using the non-linear models by creating a program that will input a sin wave of a certain frequency and then measuring the output sin wave and calculating the attenuation and phase shift. This code is currently in development and is not completely tested and dependable, so its analysis is not included.[6]

In addition to performance, the disturbance rejection capabilities of the system must also be evaluated. In the lab, a constant force can be applied to the cart using the push pull gauge as explain in the Experimental Procedure section. The actual response of the system with a constant force of f_1 = 13 N is shown below

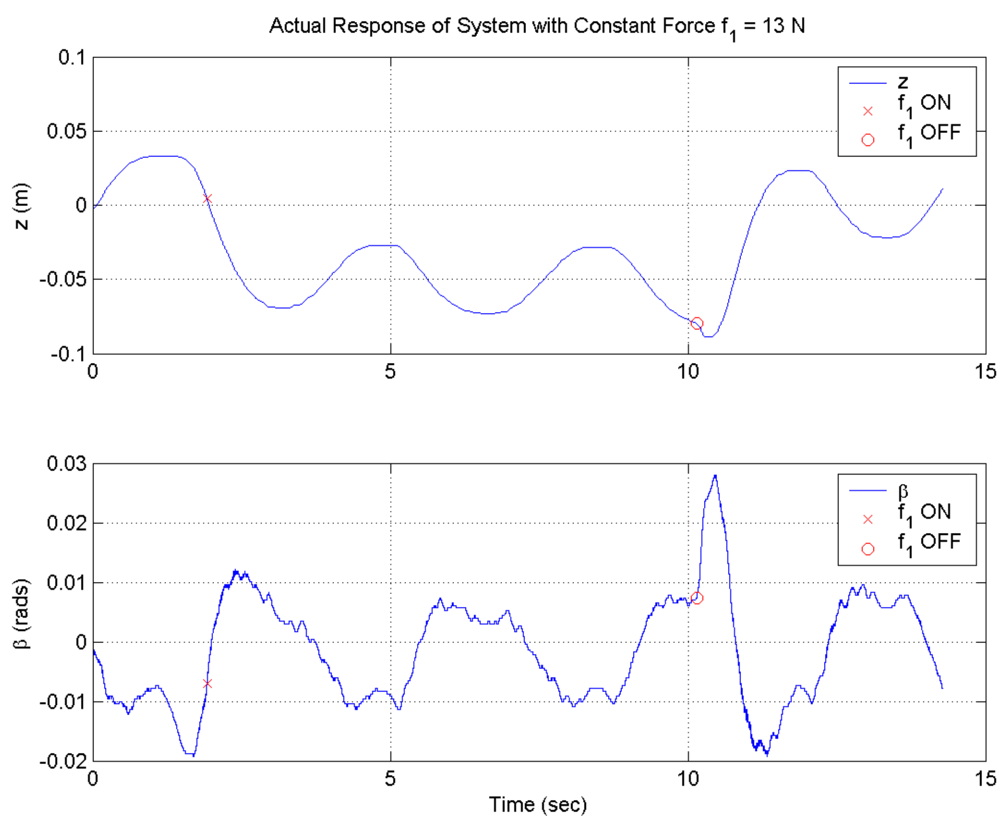


Figure 66: Actual response of system to when constant force f_1 = 13 N is applied

[6] See Lum et all, Modeling and Design of a DC Motor Control System.

This result can also be simulated by setting f_1 to a value other than zero in the non-linear S-Function. This will simulate a force on the cart. The response of the fully non-linear model with $\bar{x}_{cmd} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T$ and a constant force of f_1 = +8 N is shown below in Figure 67.

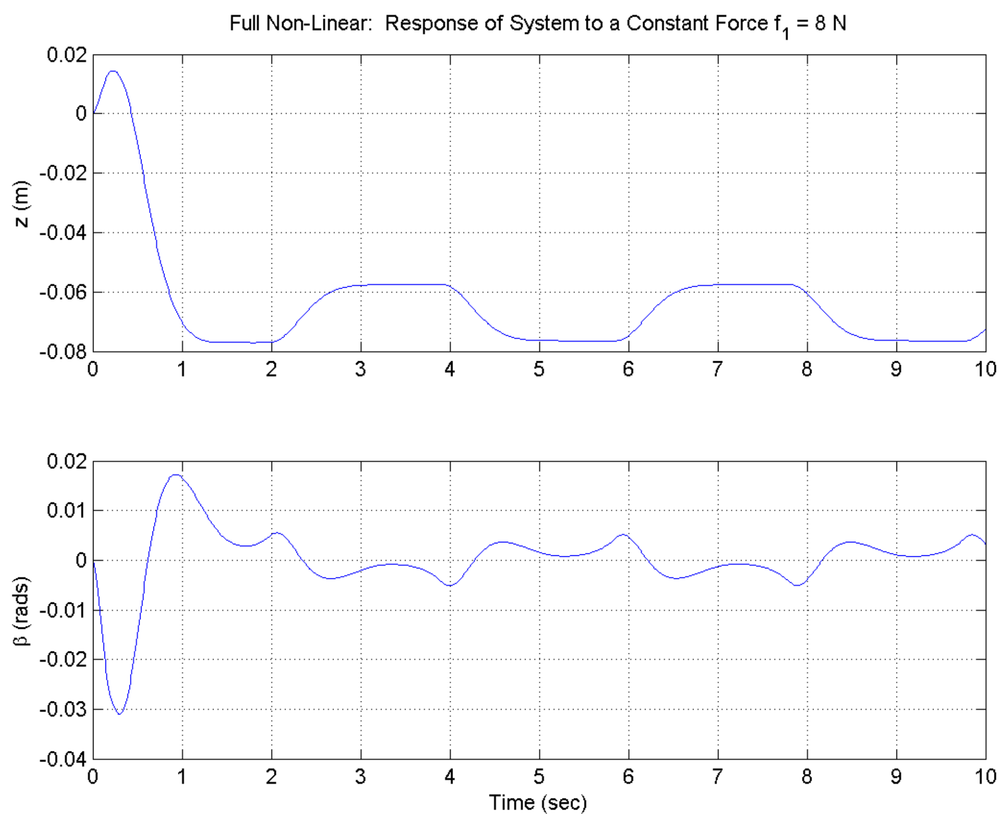


Figure 67: Simulated response of system to a constant force of 8 N with full non-linear model

As can be seen, there is a slight problem with the current controller design. Namely, it is unable to reject a constant force. The response of the system is as expected. It is still stable and still exhibits the limit cycles. However, notice that the system is not able to track a z_{cmd} of 0 in the presence of the constant force and there is some steady state error. The reason why this occurs is evident by looking at the transfer function between z and z_{cmd}

```
Transfer function:
                -1.541e004 s^2 - 469.8 s + 3.02e005
-------------------------------------------------------------------
s^5 + 395.7 s^4 + 7186 s^3 + 5.432e004 s^2 + 2.026e005 s + 3.02e005
```

As can be seen, the transfer function has a DC gain of 1. Therefore, under normal circumstances, there should be zero steady state error in the presence of a step function. However, also notice that there is no integrator. Therefore, the system is unable to reject a constant disturbance. Therefore, an integrator must be added to fix this problem. The design of this controller is discussed above in the Full State Feedback with Integrator Design section.

The other interesting observation to make here is the fact that even though a positive force was applied (to the right), the system settled out to the left of the starting point (negative z). This is possibly due to the fact that the system has non-minimum phase characteristics. The force from the motor on the cart can be evaluated by multiplying the current (x_5) by the motor constant (K) to obtain the torque produced by the motor. This can then be divided by the pulley radius (r) to obtain the force on the cart by the motor. The response of the non-linear system with the coulomb friction turned off is shown below in Figure 68.

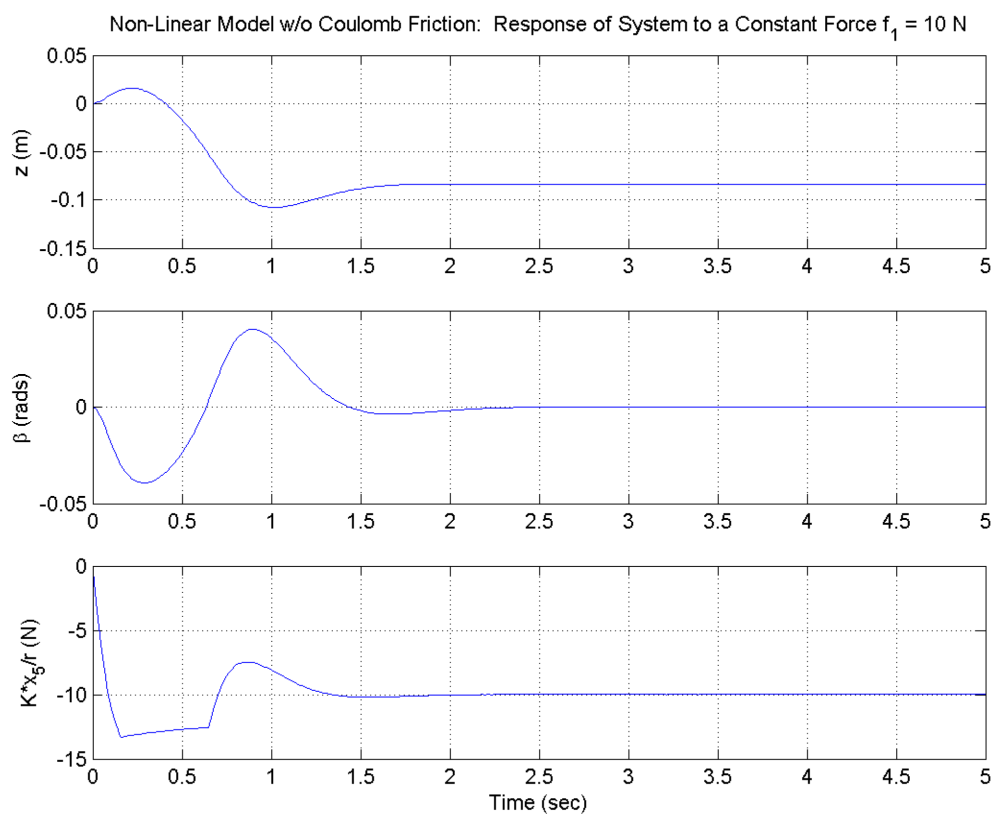


Figure 68: Non-linear model w/o coulomb friction. Showing z, beta, and Kx_5/r vs. time

As can be seen from Figure 68, the force on the cart by the motor is negative and of higher magnitude that the constant +10 N force on the cart for the majority of the time. This creates a net force to the left which makes the cart move to the left before the net force is reduced to zero around t = 1.5 seconds.

This explains why even though a positive force was applied, the cart steady state error is to the left of the starting position.

Since the system does not contain an integrator, it will have a steady state error with any constant force on the cart. The amount of force required for the system to become unstable (state x_2 to leave the equilibrium point) is tabulated below in Table 10. Since the constant force is by definition a non-linear phenomena, the linear model cannot be used to predict the response of system in the presence of a constant force.

Table 10: Disturbance rejection of full state feedback controller

	Actual System	**Full Non-Linear**	**Non-Linear w/o Coulomb Friction**	**Linear**
f_1 max (N)	17.79	9.71	10.31	N/A

As can be seen from Table 10, the maximum force that can be rejected in simulation and in the actual system is of the same order of magnitude. There are many possible reasons why the two are not exactly the same. Namely, it is extremely difficult to apply a constant force in the lab given the limit cycles cause constant motion in the cart. Also notice that these are the maximum forces that can be applied before the system saturates and becomes unstable. However any amount of force on the cart will cause a steady state error. Also note that the linear system has infinite disturbance rejection capabilities since the control voltage is unbounded.

The disturbance rejection problem is addressed in the next section, Full State Feedback with Integrator Performance.

In addition to disturbance rejection, the robustness of the control design must be taken into account. This evaluates how much unmodeled dynamics the system is able to tolerate before becoming unstable. In order to do this, the gain margin and phase margin must be evaluated. The switches shown in Figure 61 can be turned on or off to open/close certain loops. The linearized model of the system can then be obtained and the gain and phase margin can be calculated by obtaining the appropriate transfer function. Due to the fact that the system must be linearized in order to obtain the transfer function, the fully non-linear model cannot be used since the coulomb friction causes large gradients in near the equilibrium point. Furthermore, the saturation block in Figure 61 must be removed since *Simulink* will treat this as a gain, which is inaccurate. The system should not vary much from the equilibrium point so the saturation should not be needed.

For example, in order to evaluate the robustness of the system in light of an uncertainty in the plant model, the switch on the control signal is opened and the system is linearized using the linmod function. A continuous time system is then created using the ss function. The results is shown below

```
a =
              x1         x2         x3         x4         x5
   x1          0          0          1          0          0
   x2          0          0          0          1          0
   x3          0    -0.9258     -1.746    0.00144     0.5612
   x4          0      21.44      3.489   -0.03337     -1.121
   x5          0          0     -518.9          0     -393.9


b =
           u1     u2     u3     u4     u5     u6
   x1       0      0      0      0      0      0
   x2       0      0      0      0      0      0
   x3       0      0      0      0      0      0
   x4       0      0      0      0      0      0
   x5   -1091      0      0      0      0      0


c =
            x1       x2       x3       x4       x5
   y1   -25.17    -64.2   -18.53   -14.35        0
   y2        1        0        0        0        0
   y3        0        1        0        0        0
   y4        0        0        1        0        0
   y5        0        0        0        1        0
   y6        0        0        0        0        1


d =
            u1       u2       u3       u4       u5       u6
   y1        0   -25.17    -64.2   -18.53   -14.35        0
   y2        0        0        0        0        0        0
   y3        0        0        0        0        0        0
   y4        0        0        0        0        0        0
   y5        0        0        0        0        0        0
   y6        0        0        0        0        0        0
```

As can be seen, the number of inputs (columns of the b matrix) corresponds with the number of input points in Figure 61. Likewise the number of output (rows of the c matrix) matches with the number of output points in Figure 61. The other interesting thing to notice is that the A matrix is the same as the A matrix of the plant alone. This is because the controller is simply a set of gains with no states of its own. Furthermore, since the loop has been opened, the open loop poles will be obtained from the A matrix. The transfer function between input 1 and output 1 can be calculated. This gives the transfer function $u/u_{feedback}$. The result is shown below.

```
Transfer function:
 -6215 s^3 - 6.277e004 s^2 - 2.218e005 s - 3.02e005
---------------------------------------------------
s^5 + 395.7 s^4 + 970.8 s^3 - 8452 s^2 - 1.918e004 s
```

Equation 36

This transfer function gives the response of the fed-back control voltage, $u_{feedback}$, to a perturbation in the control voltage. Naturally, when this signal becomes in-phase with itself, the system will become unstable since the signals will add to create an unbounded response. Therefore, instead of –180 degree crossings, the critical crossings are 0 or –360 degrees. The bode plot of this transfer function using 0 degree crossings is shown below in Figure 69.

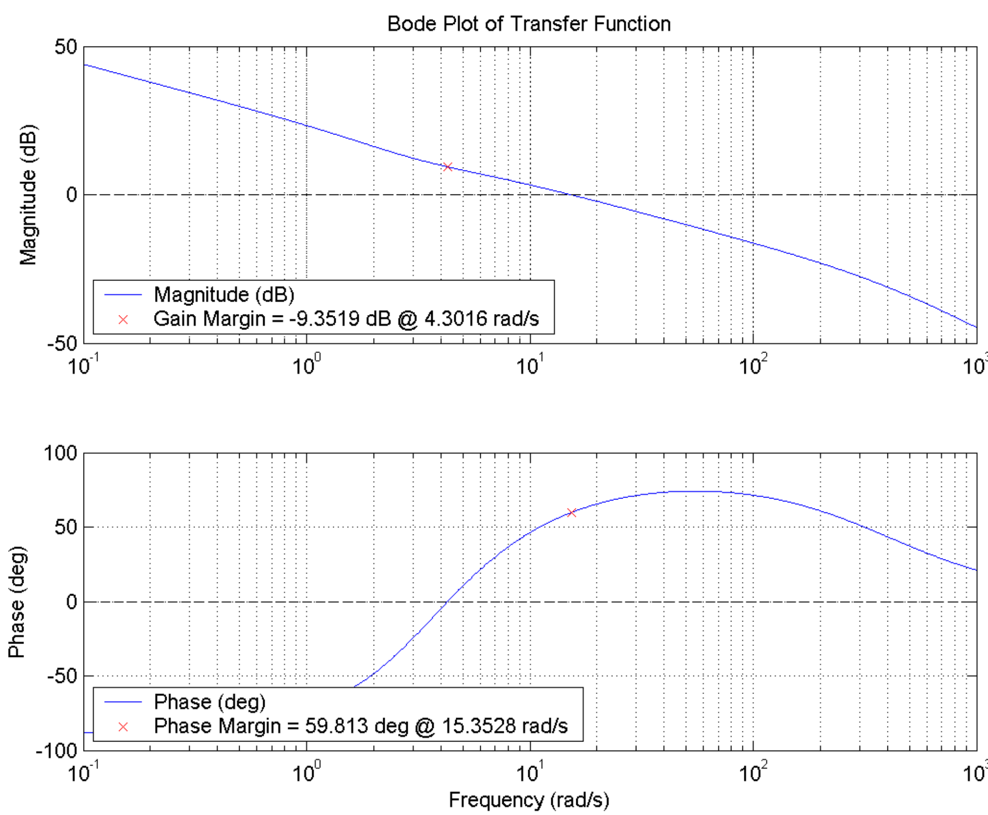


Figure 69: Bode plot of control loop with 0 degree crossing

As can be seen from Figure 69, the bode plot of the control loop shows the gain margin to be –9.3519 dB and the phase margin to be 59.813 degrees. There are several interesting observations to make here. The first is that there is a negative gain margin. This does not mean that the system is unstable, but it does set a lower limit on the amount of variation that is tolerable in the negative direction. Namely, this means that if the gain is lowered, the system will go unstable. This is characteristic of an open loop unstable system. If the gain is lowered, the closed loop poles will migrate back to their open loop locations in the right half plane. Obviously, this will make the system unstable. Equally interesting is the fact that there is no upper limit on the gain margin. This means that the gain can be increased to infinity and the system will not go unstable. Obviously, this is opposite of an open loop stable system. In an open loop stable system, if the gain is increased to infinity, the poles migrate to the right half plane, making the system unstable.

The gain margin of –9.3519 dB means that the KΔ block can be set to as low as 20 $\log_{10}(-9.3519) = 0.3407$ before the system goes unstable. The response of the system with the control loop KΔ block set to 0.3407 is shown below in Figure 70.

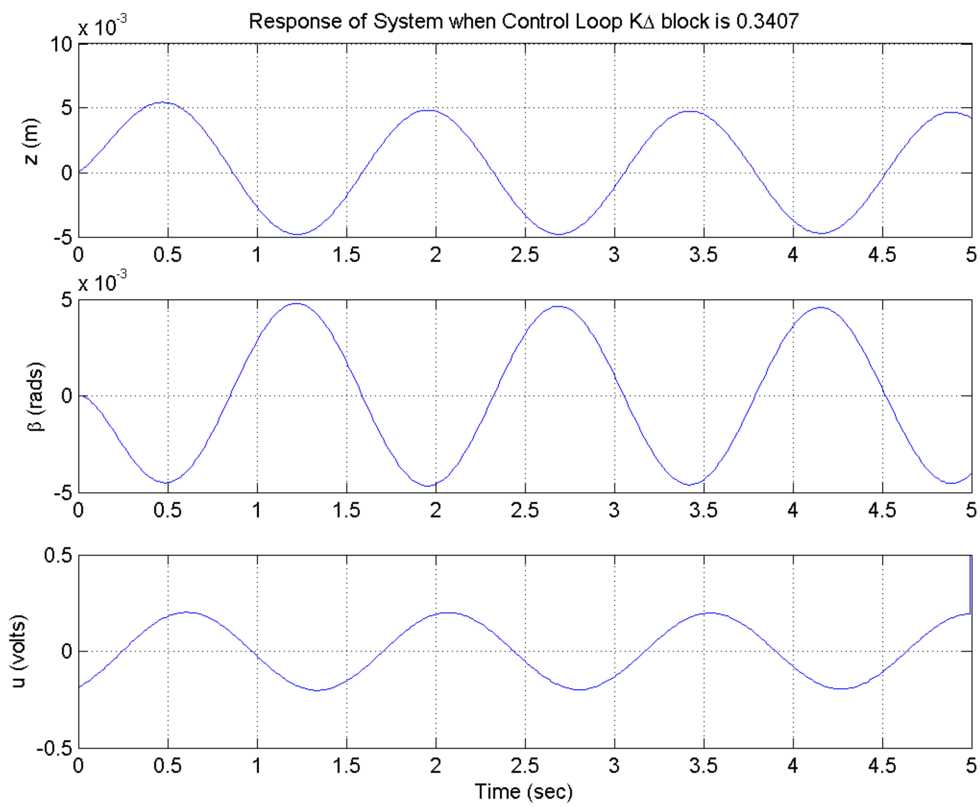


Figure 70: Response of system when control loop KΔ block set to 0.3407

As can be seen from Figure 70, the system is marginally stable. This analysis shows how much the variation in the plant model that system can sustain.

Similarly, the amount of time delay that the system can sustain can be obtained from the phase margin. The phase margin directly shows how much phase delay the system can sustain. Recall that the phase delay can be implemented by replacing the KΔ with

$$K\Delta = e^{-j\theta}$$

Equation 37

However, the time delay is represented by replacing the KΔ block with

$$K\Delta = e^{-j\omega\tau}$$

Equation 38

Therefore, the amount time delay that the system can tolerate is given by

$$\tau = \frac{\theta}{\omega}$$

Equation 39

From the bode plot of the control loop shown above in Figure 69, the phase margin of this system is 59.813 degrees which occurs at $\omega = 15.3528$ rad/s. Therefore, the maximum time delay in the control loop that the system can handle is $\tau = 0.068$ seconds. The response of the system with a time delay of this magnitude replacing the KΔ block in the plant is shown below in Figure 71.

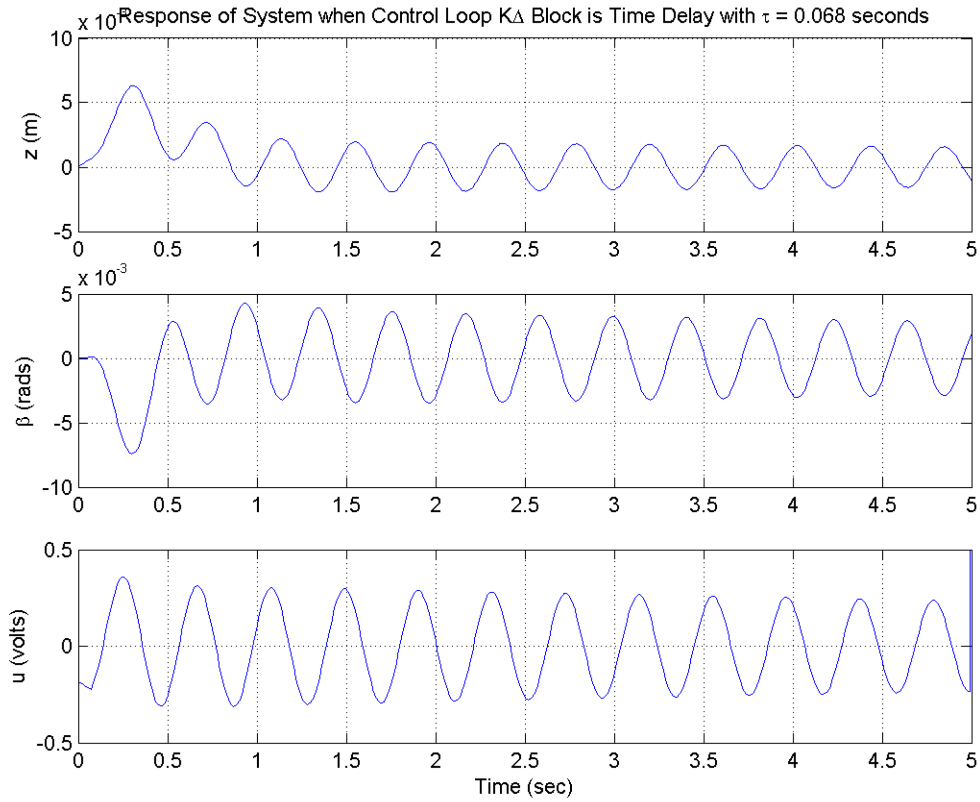


Figure 71: Response of system when control loop KΔ block is time delay with τ = 0.068 seconds

As can be seen, the system becomes marginally stable with this time delay. A similar analysis can be performed for the other 5 loops (z-loop, β-loop, dz/dt-loop, dβ/dt-loop, and i_a-loop). The interesting thing to notice is that for some of the loops, there are two crossings at 0 degrees. This implies two gain margins. One such example is the dz/dt loop as shown below in Figure 72.

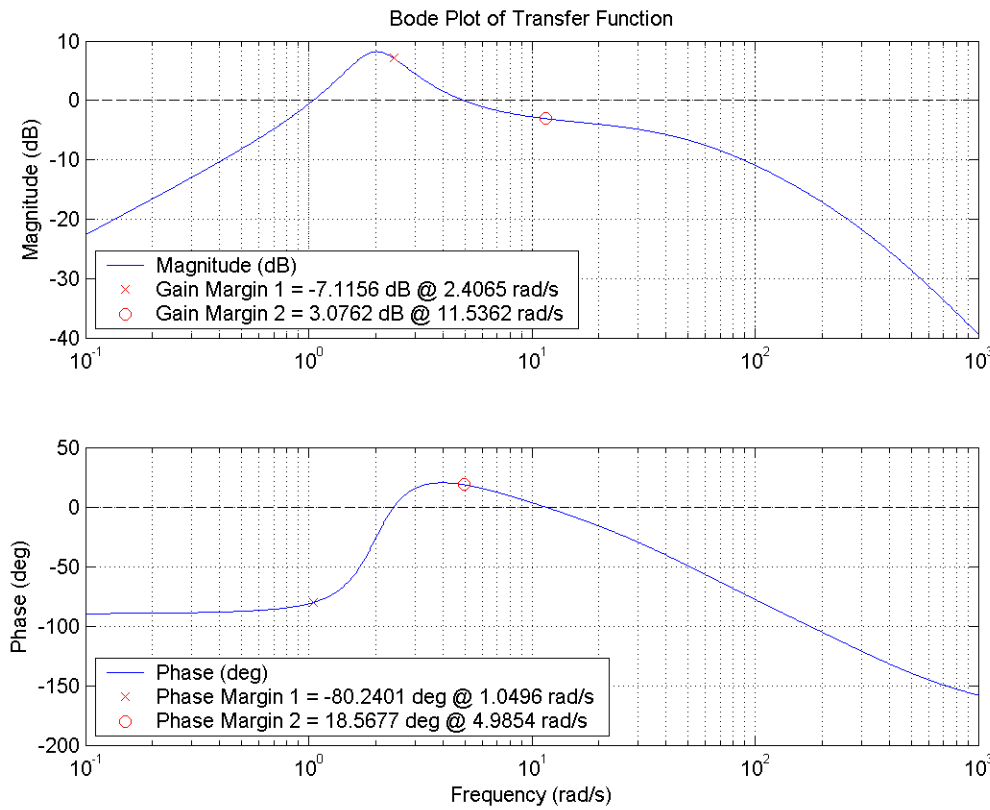


Figure 72: Bode plot of dz/dt loop

As can be seen, there are two gain margins and two phase margins. This shows that the amount that the KΔ block can be varied is bounded. The gain cannot be increased more than 3.08 dB and cannot be reduced by more than –7.12 dB.

The tabulated results for all of the different loops are shown below in Table 11.

Table 11: Robustness of different loop for full state feedback controller

Loop	**Gain Margin (dB)**	**Phase Margin (deg)**	**KΔ Range**	**Time Delay (sec)**
Control	-9.35	59.81	0.34 to ∞	0.068
z	7.02	59.31	0 to 2.25	0.651
β	-4.62 & 31.35	-37.33 & 26.29	0.59 to 26.97	0.066
dz/dt	-7.12 & 3.08	-80.24 & 18.57	0.44 to 1.43	0.065
dβ/dt	8.13 & -3.29	-18.70 & 47.57	0.69 to 2.55	0.023
i_a	N/A	N/A	N/A	N/A

As can be seen, the i_a loop is not applicable because the gain k_5 has been set to zero to simulate that this state is not available in the lab. Therefore, the gain or phase margin on this loop is not applicable since it is not being used.

Keep in mind that these gain and phase margins are evaluated for each loop individually. This means that they are valid only if every other loop that is not being analyzed has a KΔ block with a gain of 1 and a time delay of 0 seconds. In other words, the analysis is performed on each loop assuming that all the other loops are perfectly modeled. Obviously this is not the case in reality and may lead to discrepancies between the simulated robustness presented here and the actual robustness of the real system where many loops may deviate from the normal settings simultaneously.

As a final note, this control scheme assumes that all states except for current ($k_5 = 0$) are available for feedback. However, dz/dt and $d\beta/dt$ are not actually measured in lab. Instead, these are estimated using a pseudo-derivative. Therefore, it becomes necessary to develop a controller that only requires z, β, z_{cmd}, and β_{cmd} as inputs. However, since this controller cannot reject a constant disturbance, it will not be used as the final full state controller. The final controller will be analyzed using the situation where only z, β, z_{cmd}, and β_{cmd} are available as inputs. This is presented in the LQR with Integrator Performance section.

Full State Feedback with Integrator Performance

As shown previously in the Full State Feedback Design section, the current full state feedback system is somewhat inadequate. It is not able to handle a constant force on the cart and has a steady state error in z. The gain matrix is repeated here for convenience

```
Ki_matrix =

      58.943       93.105       30.157       20.843    -0.016024      -50.346
```

Once again, note that the sixth element of this matrix is actually $-k_6$ due to the architecture. Once again, the state x_5 is not available in the lab, so this gain is set to zero. Naturally, this will change the location of the closed loop poles so they are not exactly at the desired locations anymore. Hopefully, since k_5 is small, this will not drastically change the locations or affect the stability.

The first thing to evaluate is the stability of the system. As demonstrated above in the Full State Feedback with Integrator Design section, the closed loop poles are stable. The damping and maximum real part of the closed loop poles can be evaluated using the damp function

```
>> damp(ai)

        Eigenvalue             Damping      Freq. (rad/s)

 -1.99e+000                   1.00e+000      1.99e+000
 -3.96e+000 + 3.59e+000i      7.41e-001      5.35e+000
 -3.96e+000 - 3.59e+000i      7.41e-001      5.35e+000
 -4.17e+000                   1.00e+000      4.17e+000
 -6.79e+000                   1.00e+000      6.79e+000
 -3.75e+002                   1.00e+000      3.75e+002
```

As can be seen, by setting k_5 to zero, this slightly changes the location of the closed loop poles. The minimum damping is 0.741 and the maximum real part of any of the eigenvalues is -1.99. Since this system is still stable, there is no need for concern.

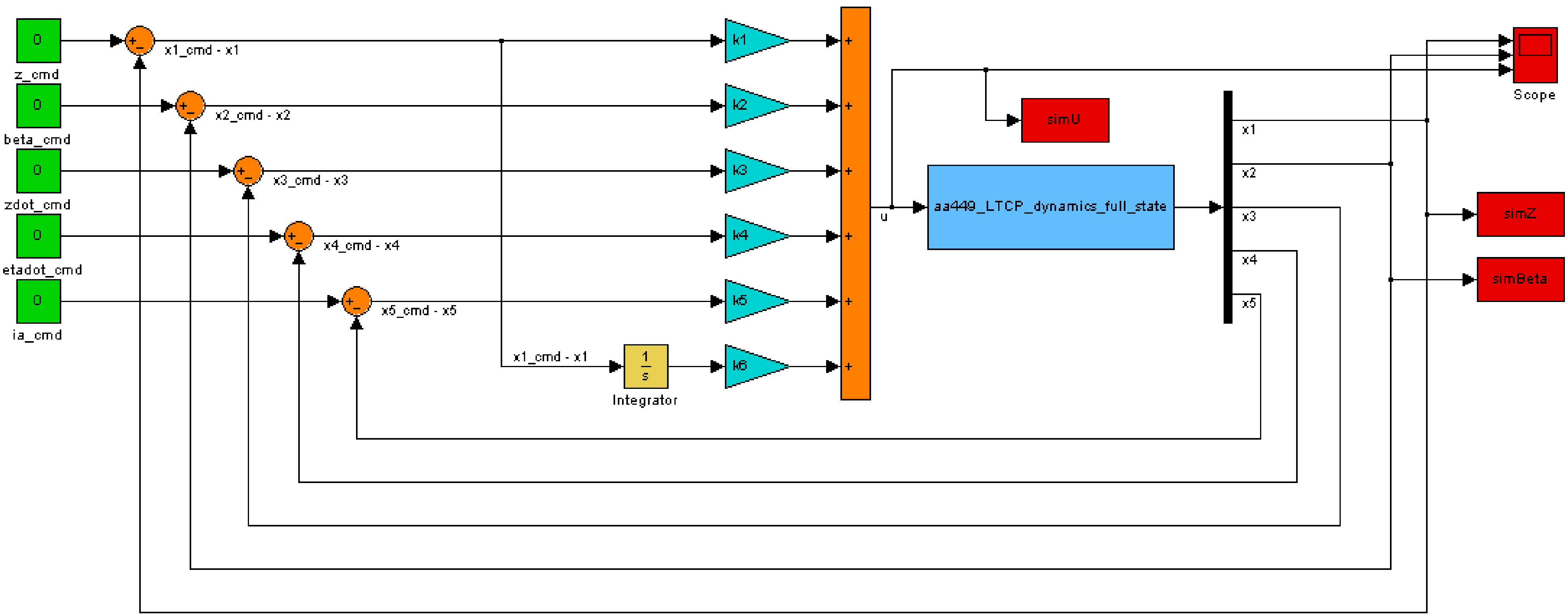


Figure 73: *Simulink* model used to simulate full state controller with integrator

Next, the response of the system to initial conditions can be evaluated. The response of the full non-linear model to an initial condition of z(0) = 0.153 m is shown below in Figure 74.

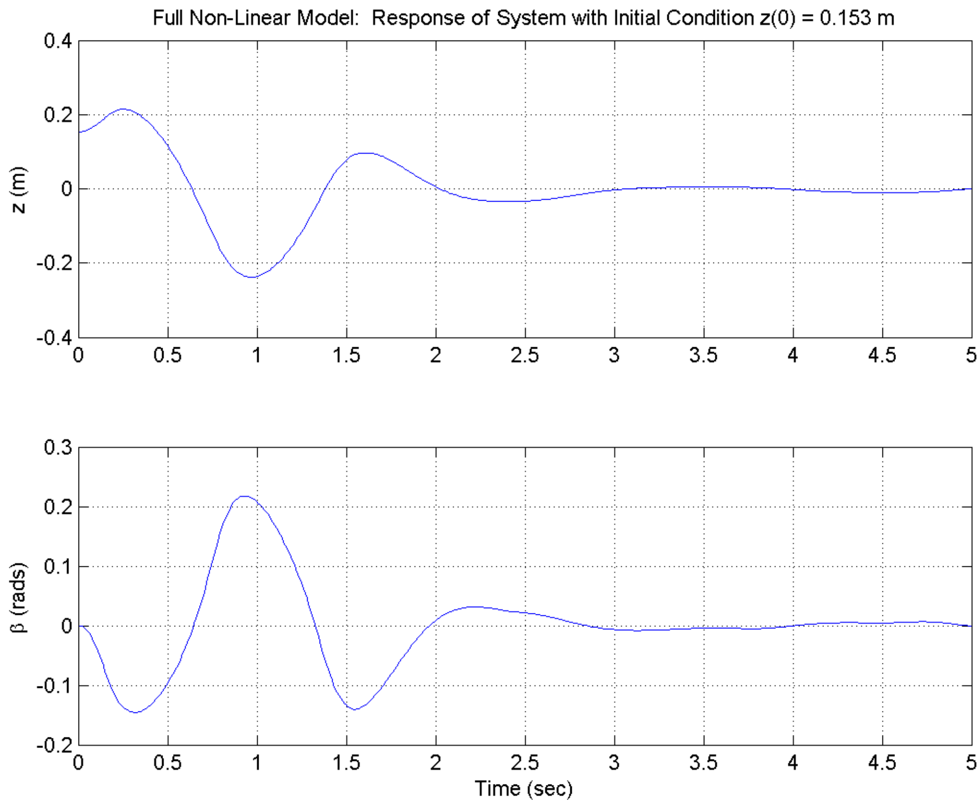


Figure 74: Response of full non-linear model to initial condition of z(0) = 0.153 m

As can be seen, the system recovers from this initial condition. A similar analysis can be performed for the response to an initial condition, β(0) and the process is repeated for the other two models (the non-linear w/o coulomb friction and the fully non-linear model). The results are tabulated below in Table 12.

Table 12: Response of system to initial conditions using full state feedback with integrator controller

	Actual System	**Full Non-Linear**	**Non-Linear w/o Coulomb Friction**	**Linear**
z(0) (m)	N/A	0.153	0.154	N/A
β(0) (deg)	N/A	6.42	6.47	N/A
Limit cycle (peak to peak)	+0.05/-0.0087	+/-0.084	None	None

It appears that this controller is not able to handle a perturbation in initial conditions as well as the full state feedback controller without the integrator.

Granted, these are two completely different controllers with two different sets of gains. Once again, it appears that the full non-linear model represents the actual system best.

Now that the response of the system to initial conditions is known, the initial conditions can be set to roughly zero and the response of the system to a step input can be evaluated. The actual and simulated responses of the system to a step in z of magnitude 0.1 meters using the different models is shown below in Figure 75 through Figure 77.

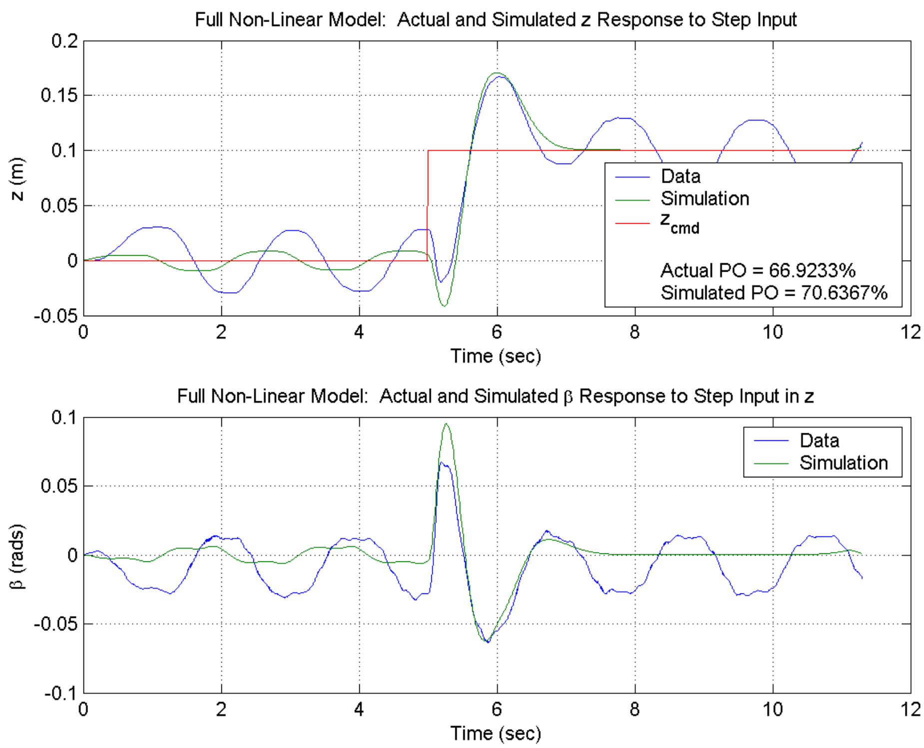


Figure 75: Response of fully non-linear model with full state feedback with integrator controller

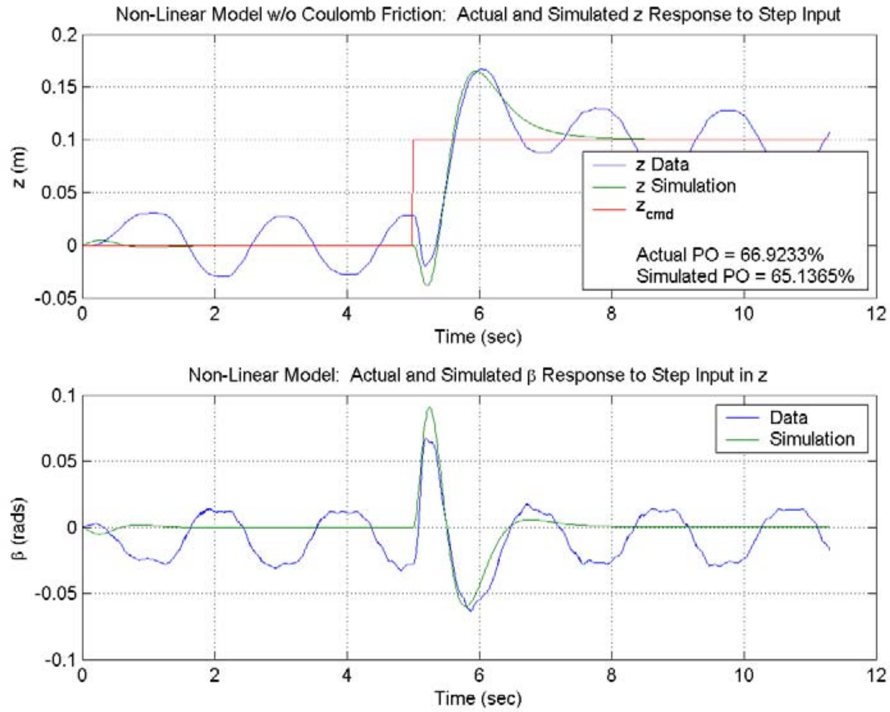


Figure 76: Response of non-linear model w/o coulomb friction w/ full state feedback w/ integrator

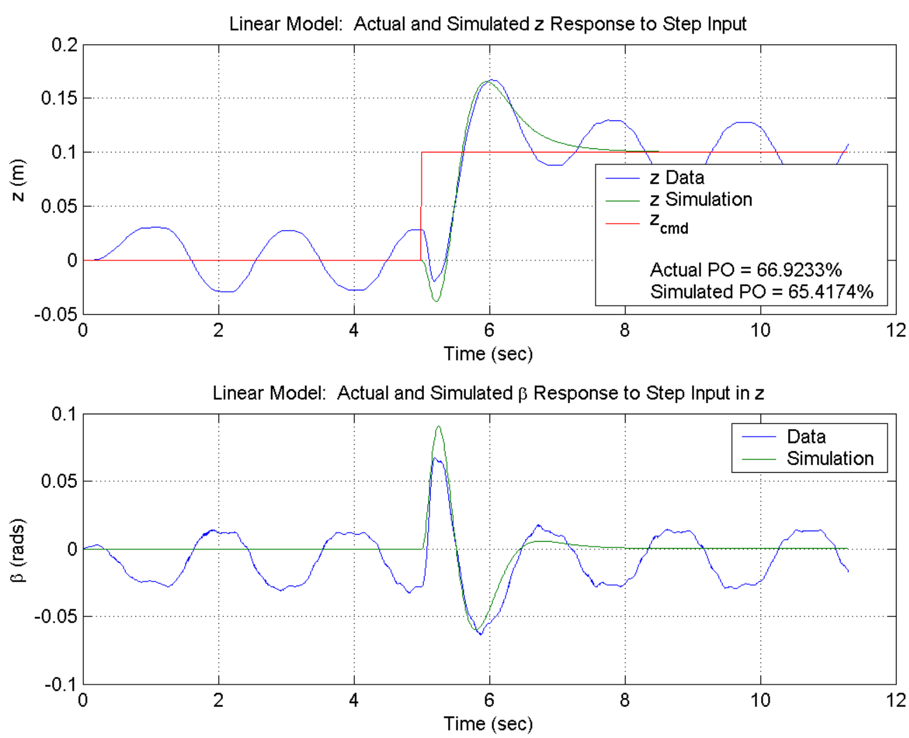


Figure 77: Response of linear model w/o coulomb friction w/ full state feedback w/ integrator

The responses of the different models with the full state feedback with integrator controller are tabulated below in Table 13.

Table 13: Performance characteristics of full state feedback with integrator controller

	Actual System	Full Non-Linear	Non-Linear w/o Coulomb Friction	Linear
Percent Overshoot (%)	66.92	70.64	65.14	65.41
Settling Time (sec)	INF	INF	2.8	2.8
Rise Time (sec)	0.595	0.17	0.17	0.18
Bandwidth (rad/s)	N/A	N/A	N/A	10.7

As can be seen, it appears that the simulation models match the actual percent overshoot extremely well. There appears to be a large discrepancy in the rise time however. This can be explained by looking at how the rise time is calculated. The response of the actual system and the parameters used to calculate rise time are shown below in Figure 78.

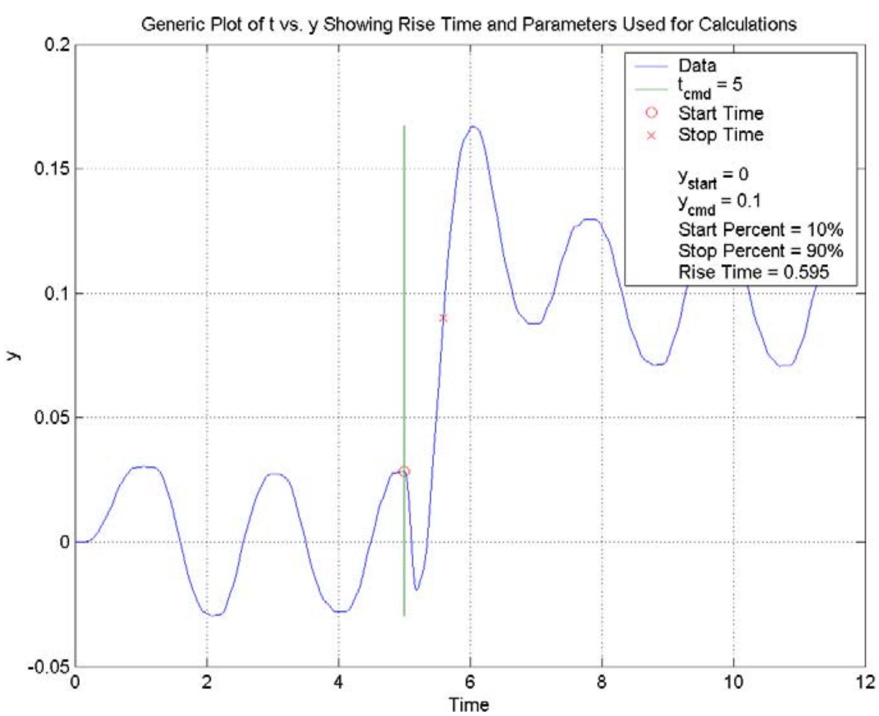


Figure 78: Actual response of system and parameters used to calculate rise time

The response and the parameters used to calculate rise time for the fully non-linear simulation is shown below in Figure 79.

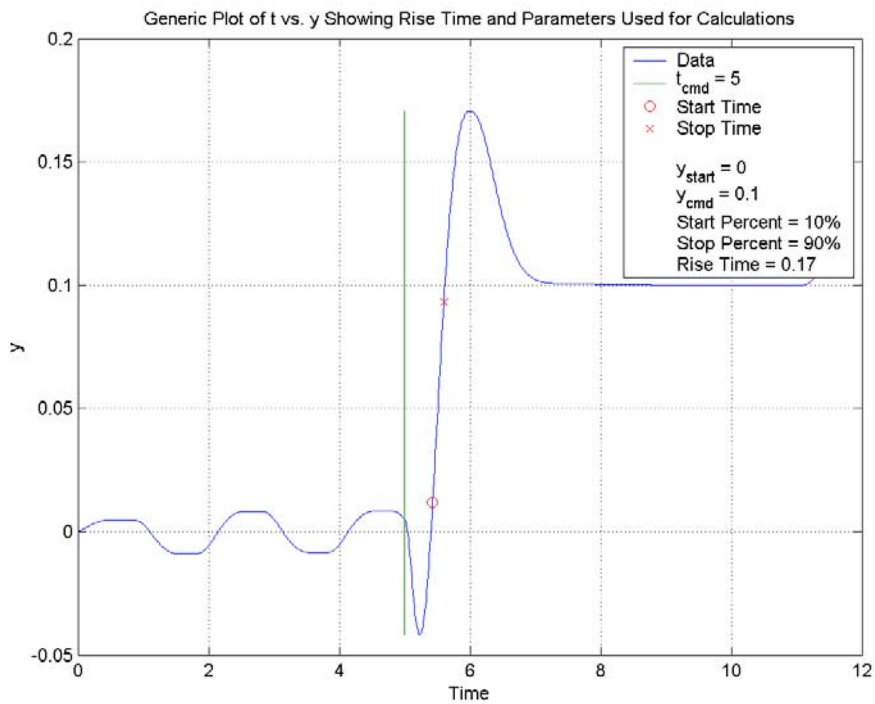


Figure 79: Simulated response of system and parameters used to calculate rise time

As can be seen, the rise time is defined as the time it takes the system to reach 90% of the final value from 10% of the final value. As can be seen from Figure 78, the limit cycle causes the rise time calculate to start exactly at t = 5 seconds since the response of the system at t = 5 is greater than 0.01 (10% of 0.1 m). This shows that the actual rise time is a function of the initial conditions. Conversely, in the simulation, the limit cycle is smaller than 0.01 so the calculation of rise time does not start until the system is already moving in the correct direction. This is why the simulated rise time is so much smaller than the actual rise time. In actuality, the simulated rise time is probably the more accurate number and better represents the time it takes the system to move to its commanded values. This explains why there is such a large discrepancy in the simulated and the actual rise time.

Now that the performance of the full state feedback with integral controller has been evaluated, the disturbance rejection capabilities of the system can be evaluated. A constant force of f_1 = 13 N is applied to the system. The response with the integrator and without the integrator is shown below.

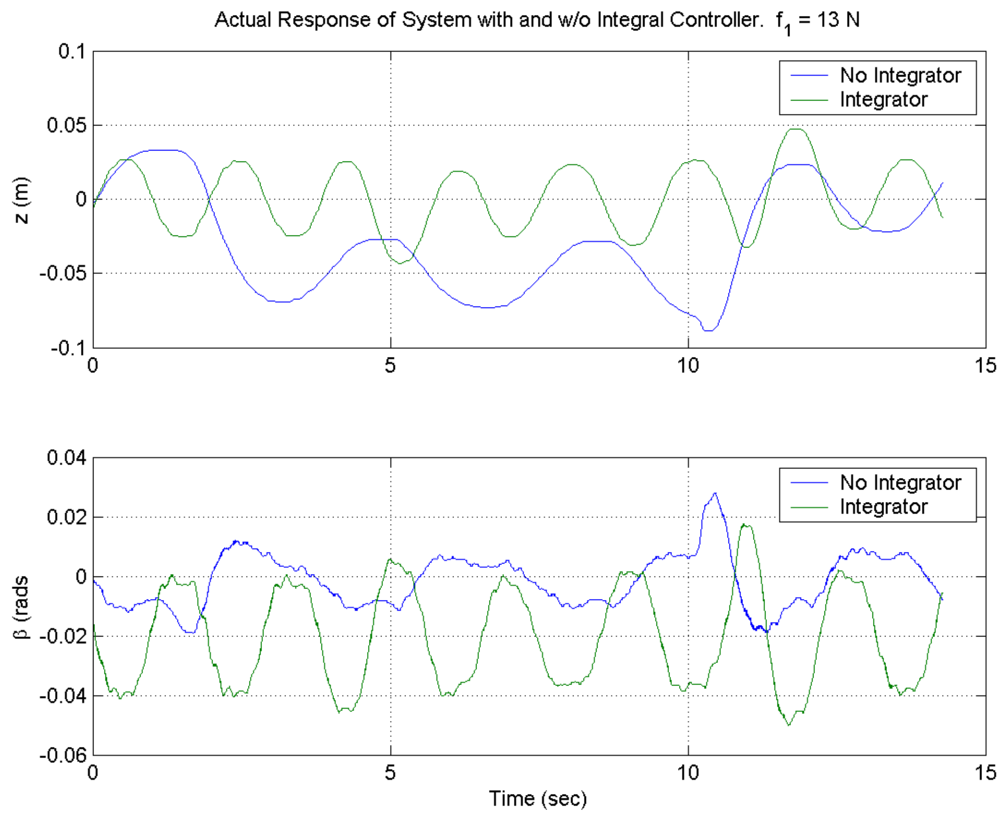


Figure 80: Actual response of system to a constant force f1 = 13N with and without integral control

As can be seen, the integral control does indeed assure that the system oscillates about 0 even in the presence of a constant force. The interesting thing to notice is that from the z vs. time response, it is difficult to tell when the force was applied and when it was removed in lab. However, this becomes more evident when looking at the u vs. time response as shown below in

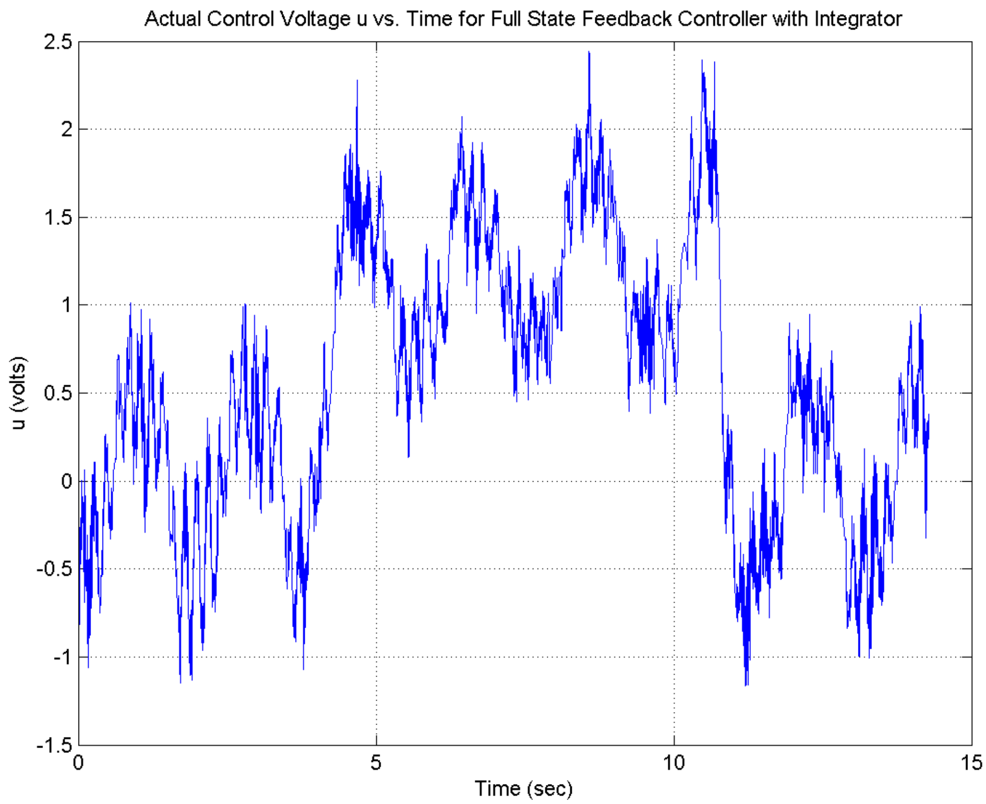


Figure 81: Actual u vs time for full state feedback controller with integrator

As can be seen from Figure 81, it appears that the force was applied at roughly 4 seconds and was removed at roughly 11 seconds. This shows that the voltage has an offset in order to oppose the constant force applied to the cart. Also notice that the peaks of the control voltage when the force is applied ($4 < t < 11$ sec) are nearing the saturation limit of 2.78 volts. This is the limiting factor that dictates how much force the system can reject. As the force increases, so does the control voltage necessary to counter this force. If the force is too high, the DC motor will saturate and the system will become unstable.

The actual response of the system can be compared with the simulation. Note that the simulation is not able to handle 13 Newtons, so a constant force of 8 Newtons is used instead.

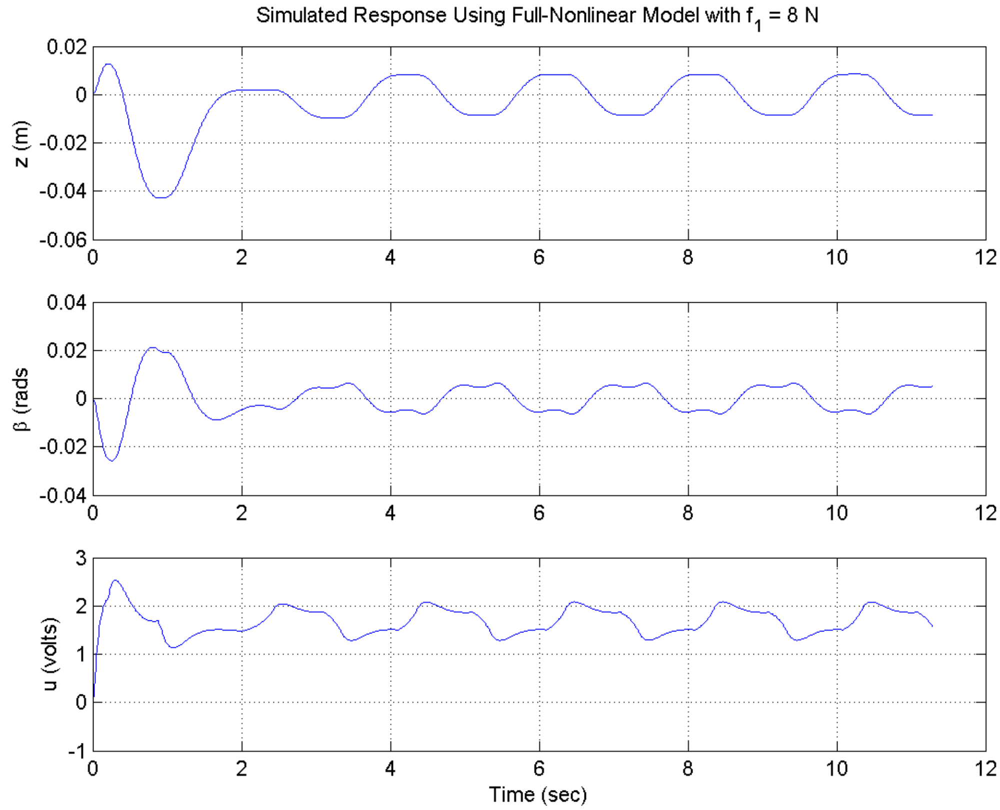


Figure 82: Simulated response of system subject to $f_l = 8$ N.

As can be seen, the simulated response matches the actual response quite well. It predicts that even under a constant load, the system will oscillate about the command signal. It also predicts that the control voltage will have a positive offset in order to counter the constant force. The disturbance rejection capabilities of the various different models are tabulated below in Table 14.

Table 14: Disturbance rejection of full state feedback with integrator controller

	Actual System	**Full Non-Linear**	**Non-Linear w/o Coulomb Friction**	**Linear**
f_l max (N)	17.8	10.03	10.60	N/A

Notice that the disturbance rejection capabilities look almost the same as the full state controller with no integrator. However the main difference is that with the integral controller, the system is able to obtain zero steady state error (with in the bounds of the limit cycles) for any value of f_l that is less that the values in Table 14. Previously, any value of f_l would have resulted in steady state error.

Now that the disturbance rejection capabilities of this controller have been evaluated, the robustness of the controller can be analyzed. The bode plots of the various loops are shown below in

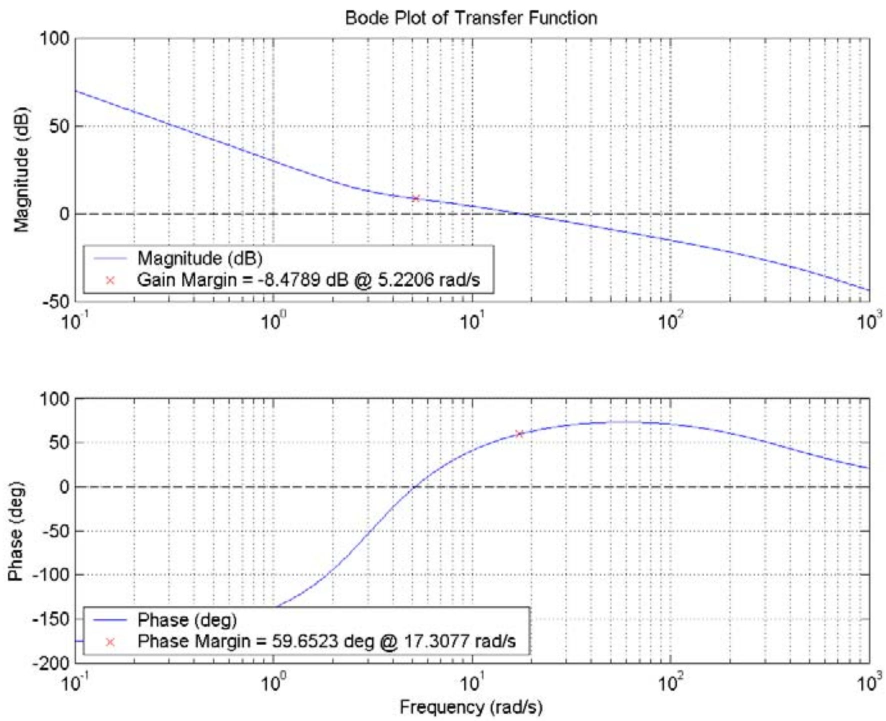


Figure 83: Bode plot of control loop using full state feedback with integrator controller

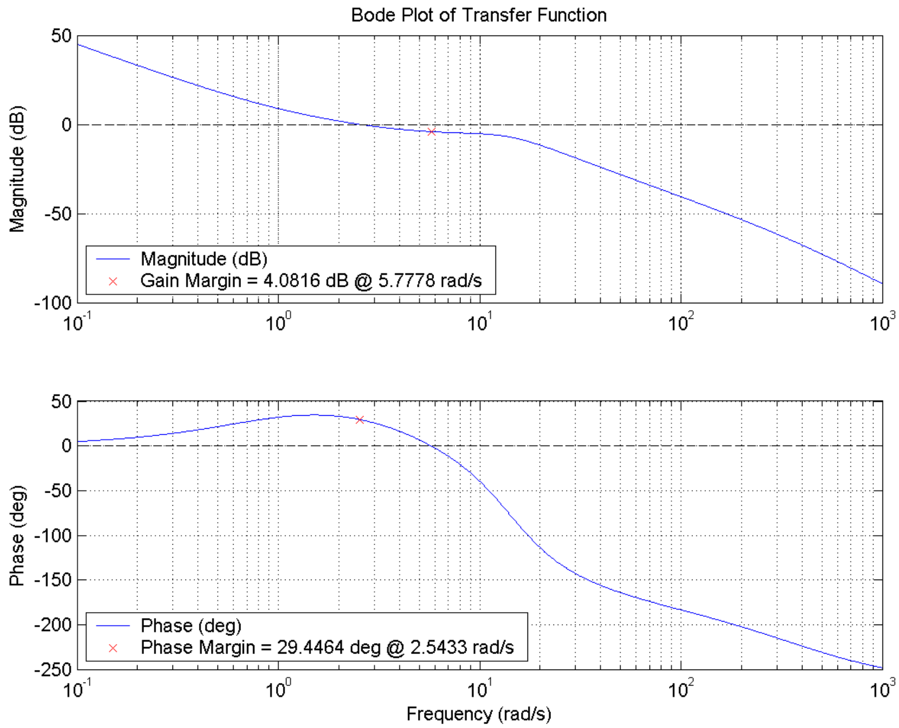


Figure 84: Bode plot of z loop using full state feedback with integrator controller

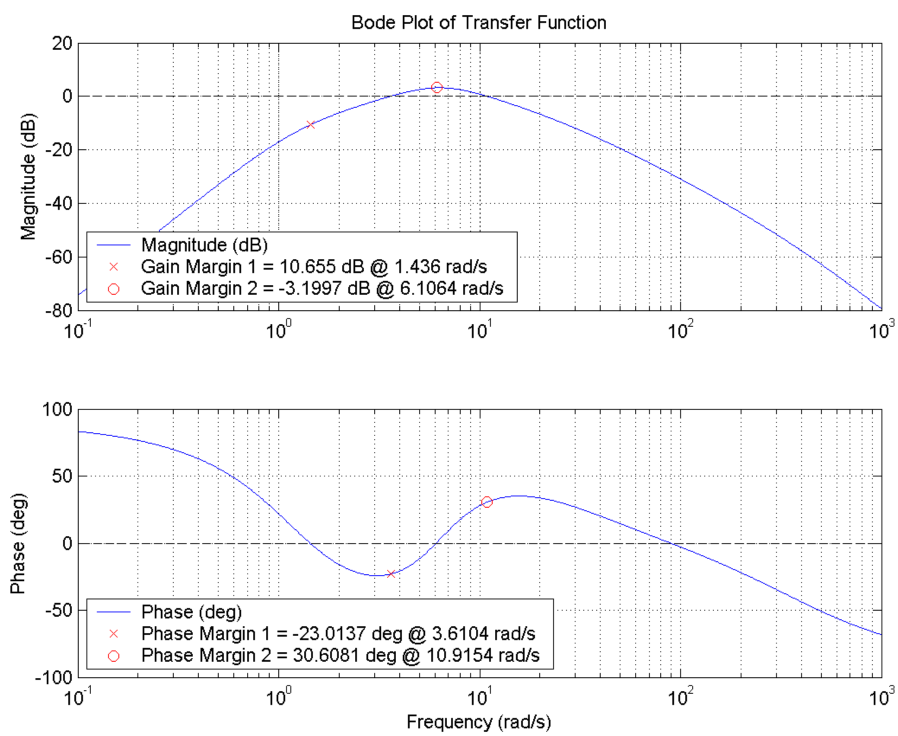


Figure 85: Bode plot of β loop using full state feedback with integrator controller

The results of the gain margin and variation along with the phase margin and variation are tabulated below in Table 15.

Table 15: Robustness of different loop for full state with integrator controller

Loop	**Gain Margin (dB)**	**Phase Margin (deg)**	**KΔ Range**	**Time Delay (sec)**
Control	-8.48	59.65	0.38	0.060
z	4.08	29.45	1.60	0.202
β	-3.20 & 10.65	-23.01 & 30.61	0.69 & 3.41	0.049

LQR with Integrator Performance

Since this controller is considered the final controller, it should be modeled the same way that is implemented in the laboratory environment. In the previous control designed and performance evaluations, it was designed such that the state x_5 was not needed for feedback by designing such that k_5 was small and could be assumed zero. However, these designs assumed that the states dz/dt and dβ/dt were readily available. In actuality, these states must be estimated. Therefore, the controller shown below in
Figure **86** is used to estimate the states that are not measured and implement the quasi-full state feedback using these estimates.

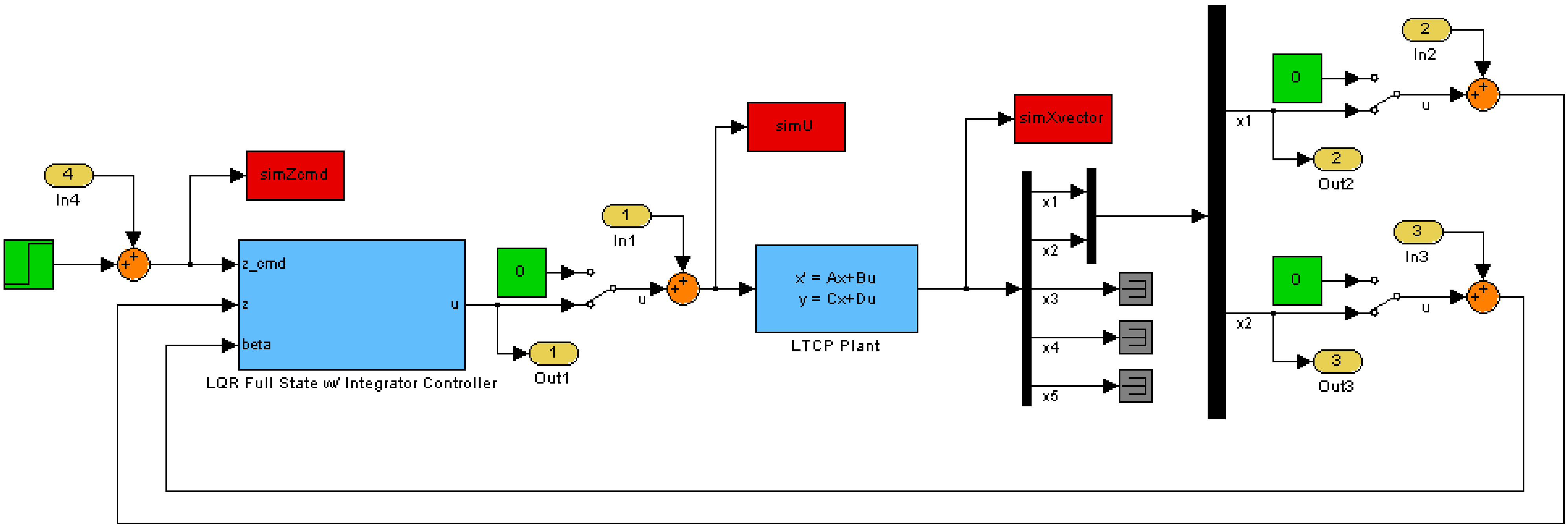


Figure 86: Full state LQR with Integrator *Simulink* Model

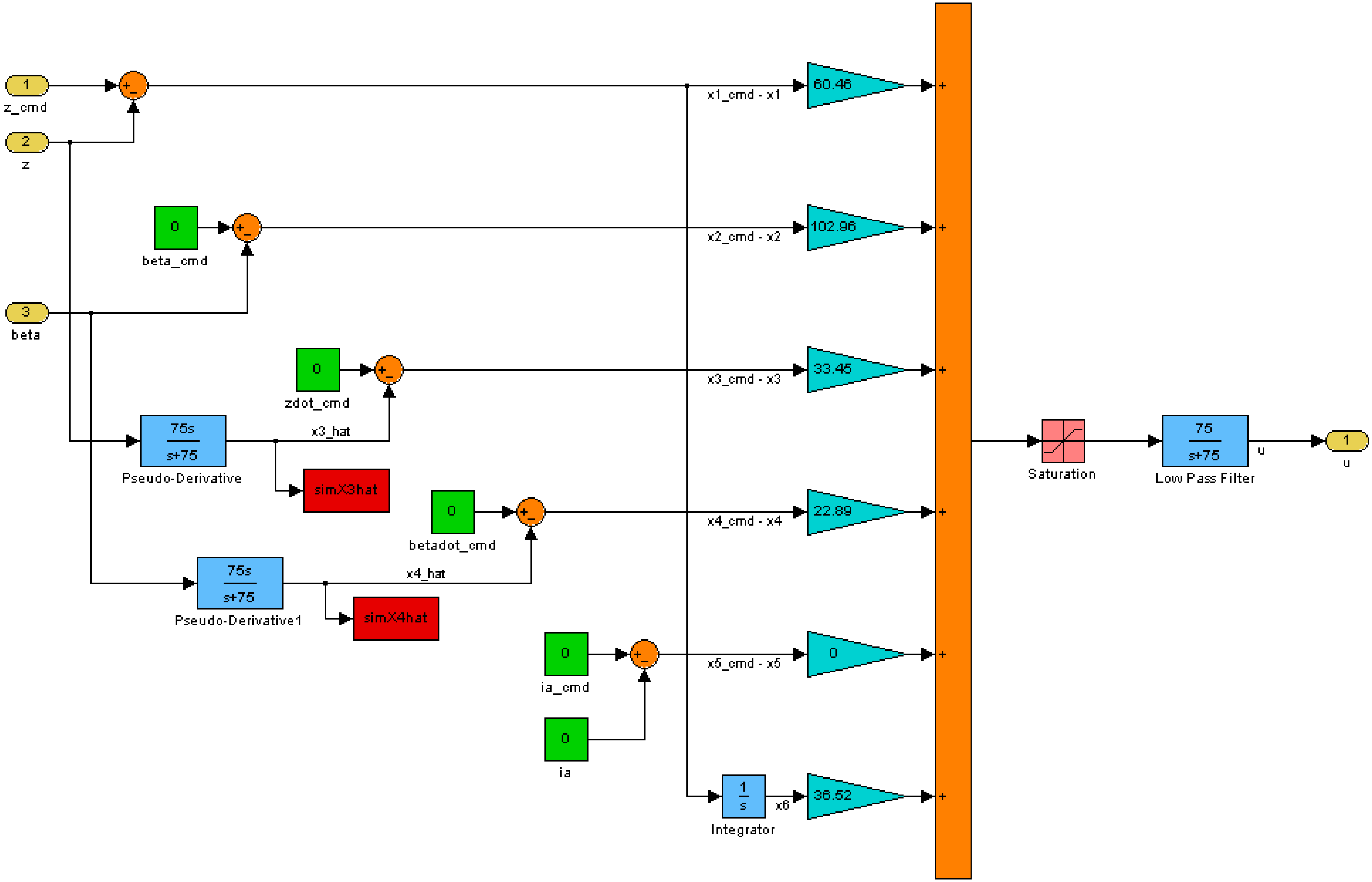


Figure 87: LQR Full-state with integral controller

The closed loop eigenvalues of the system and their damping can easily be calculated using the linmod function. Note that it would not be correct to simply calculate the closed loop eigenvalues by simply using $\lambda(A_i - B_iK_i)$. This is because this gives the eigenvalues if full state feedback were available without estimation. These eigenvalues are shown below.

```
>> damp(Ai-Bi*Ki)

        Eigenvalue              Damping     Freq. (rad/s)

 -1.00e+000                    1.00e+000      1.00e+000
 -3.93e+000 + 1.21e+000i       9.56e-001      4.11e+000
 -3.93e+000 - 1.21e+000i       9.56e-001      4.11e+000
 -6.16e+000 + 5.28e+000i       7.59e-001      8.11e+000
 -6.16e+000 - 5.28e+000i       7.59e-001      8.11e+000
 -3.93e+002                    1.00e+000      3.93e+002
```

However, in this situation, the states x_3 and x_4 are estimated using pseudo derivatives. Therefore, it becomes necessary to use linmod to find the actual eigenvalues of the system including the estimators and filters. This is shown below.

```
>> [ai,bi,ci,di]=linmod('aa449_full_state_integrator_lqr_linear');
>> damp(ai)

        Eigenvalue              Damping     Freq. (rad/s)

 -1.03e+000                    1.00e+000      1.03e+000
 -2.43e+000                    1.00e+000      2.43e+000
 -3.64e+000 + 4.25e+000i       6.50e-001      5.59e+000
 -3.64e+000 - 4.25e+000i       6.50e-001      5.59e+000
 -1.50e+001 + 2.20e+001i       5.62e-001      2.66e+001
 -1.50e+001 - 2.20e+001i       5.62e-001      2.66e+001
 -7.50e+001                    1.00e+000      7.50e+001
 -1.13e+002                    1.00e+000      1.13e+002
 -3.92e+002                    1.00e+000      3.92e+002
```

As can be seen, the closed loop poles of the actual system are slightly different than those given by $\lambda(A_i - B_iK_i)$. The main reason for this is that the states x_3 and x_4 are estimated. Also notice that the system is now a 9th order system instead of a 6th order. This is because there are two extra states associated with the two pseudo-derivatives and a single state associated with the control filter. The largest real part of the eigenvalues is –1.03 and the smallest damping ratio is 0.562.

Now that the system is proven to be stable, the region of attraction can be calculated. The regions of attraction for the two non-linear models are shown below in Figure 88 and Figure 89.

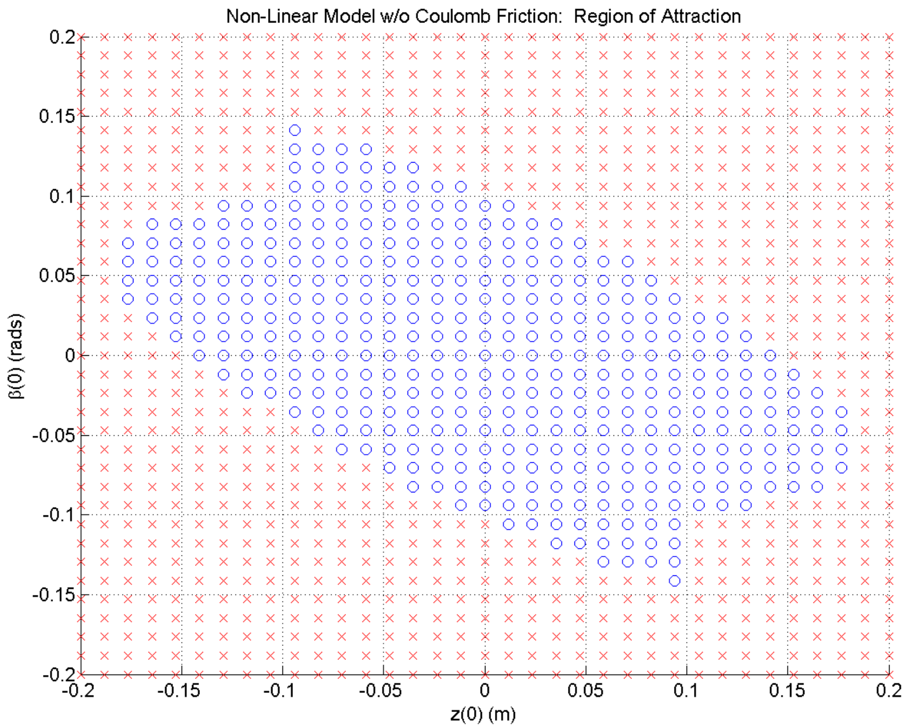


Figure 88: Non-linear model w/o coulomb friction region of attraction

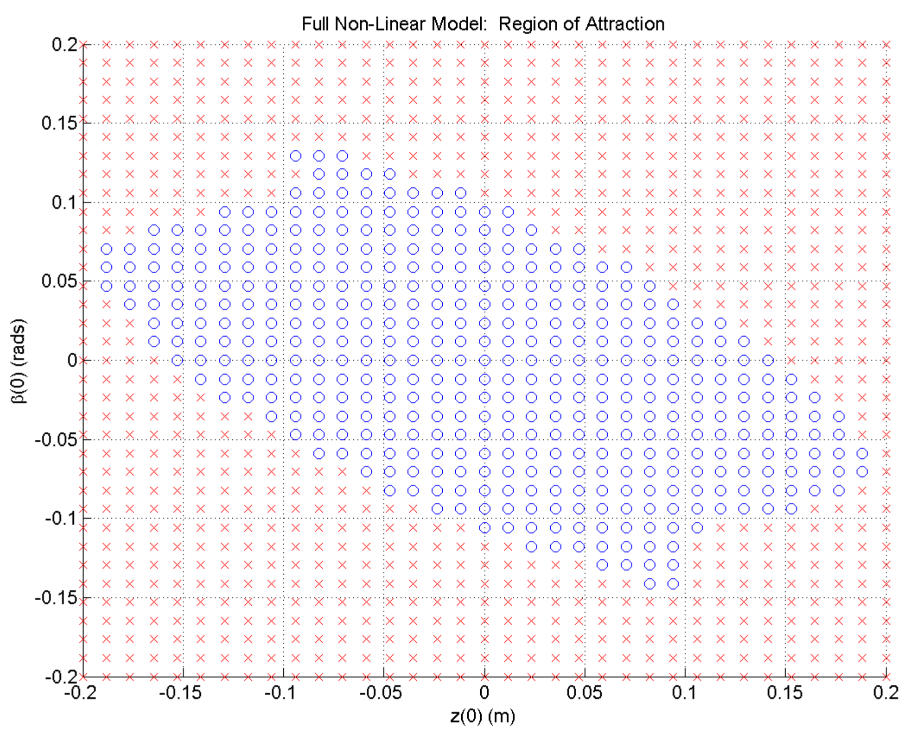


Figure 89: Full non-linear model region of attraction

As can be been from Figure 88 and Figure 89, the responses of the systems to initial conditions are fairly similar. Note that the region of attraction for the non-linear model w/o coulomb friction is symmetric about the z(0) = -β(0) line. This makes sense considering that the model is symmetric with no offsets. A point reflected about this line corresponds to two situations shown below in

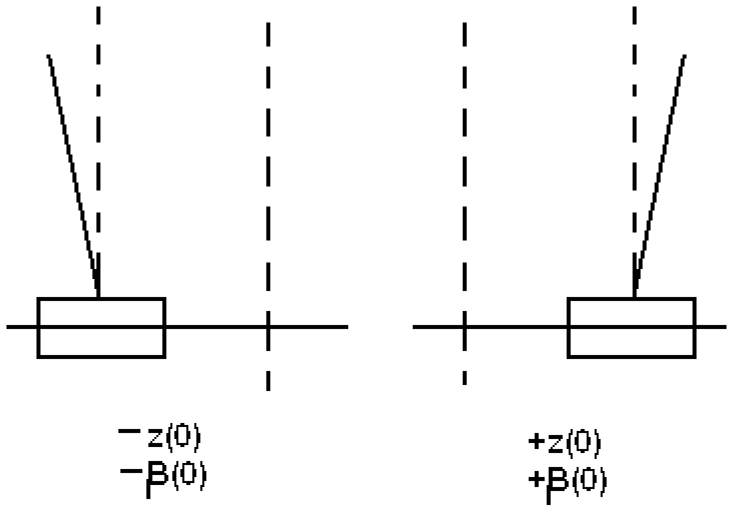


Figure 90: Two situations showing conditions reflected about z(0) = -β(0) line

This shows why if there are no biased signals or offsets, the region of attraction should be symmetric about the z(0) = -β(0) line. Notice that this is not the case with the Full Non-linear model with coulomb friction. This is due to the fact that the simulation needs to be run with a finite dz/dt(0) in order to have the coulomb friction work correctly (there are discontinuities with the f_s term at dz/dt = 0).

The results of the region of attraction analysis are tabulated below in Table 16.

Table 16: Response of system to initial conditions using full state LQR feedback with integrator controller

	Actual System	**Full Non-Linear**	**Non-Linear w/o Coulomb Friction**	**Linear**
z(0) (m)	N/A	0.14	0.14	N/A
β(0) (deg)	N/A	5.17	5.17	N/A
Limit cycle (peak to peak)	+0.018/-0.016	+/-0.0057	None	None

By comparing the peak to peak limit cycle size for the LQR controller with the peak to peak limit cycle size for the non-optimal full-state feedback controller, it can be seen that the LQR controller cuts the size of the limit cycle by more than half. This is a product of choosing a low R value and thus, making the overall gains higher. However, the consequence of this is that the region of attraction for the LQR controller is smaller than the region of attraction for the non-optimal controller.

Now that the region of attraction has been defined, the transient response of the system can be evaluated by subjecting the system to a step input. The actual and simulated responses of the system to a step in z of magnitude 0.1 meters using the different models is shown below in Figure 91 through Figure 93.

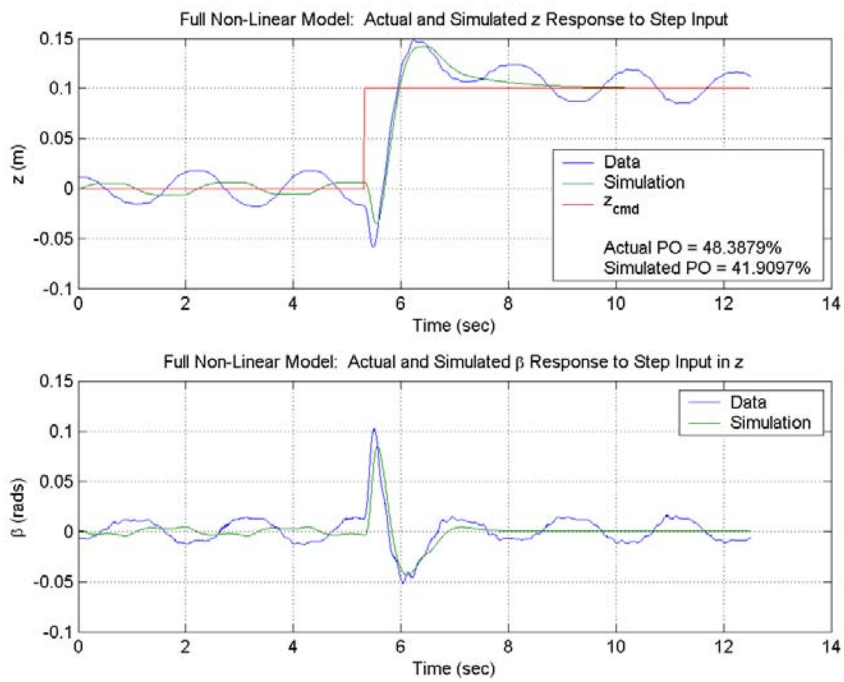


Figure 91: Actual and simulated response to a z_{cmd} step of 0.1m using fully non-linear model

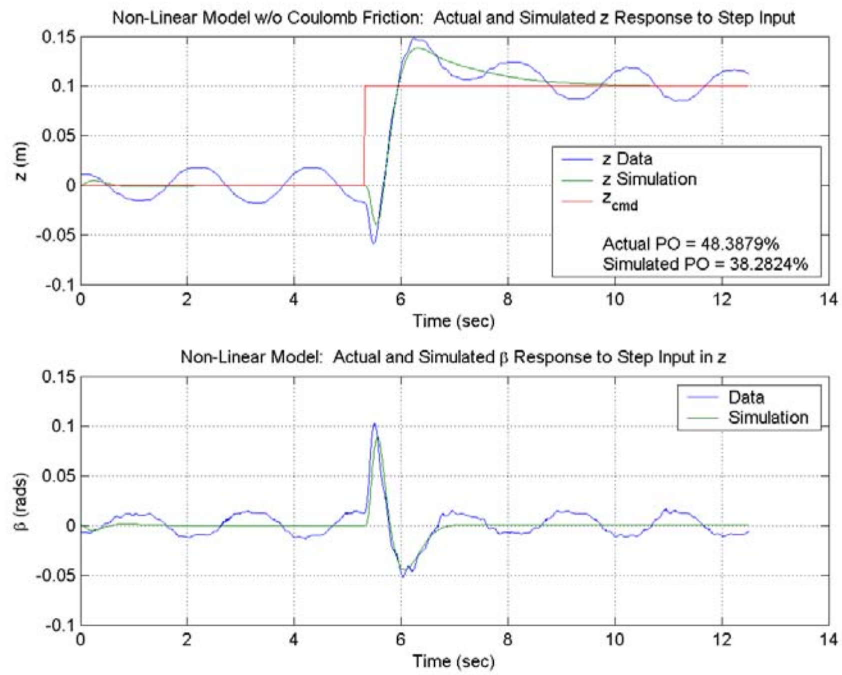


Figure 92: Actual and simulated response to a z_{cmd} step of 0.1m using model w/o coulomb friction

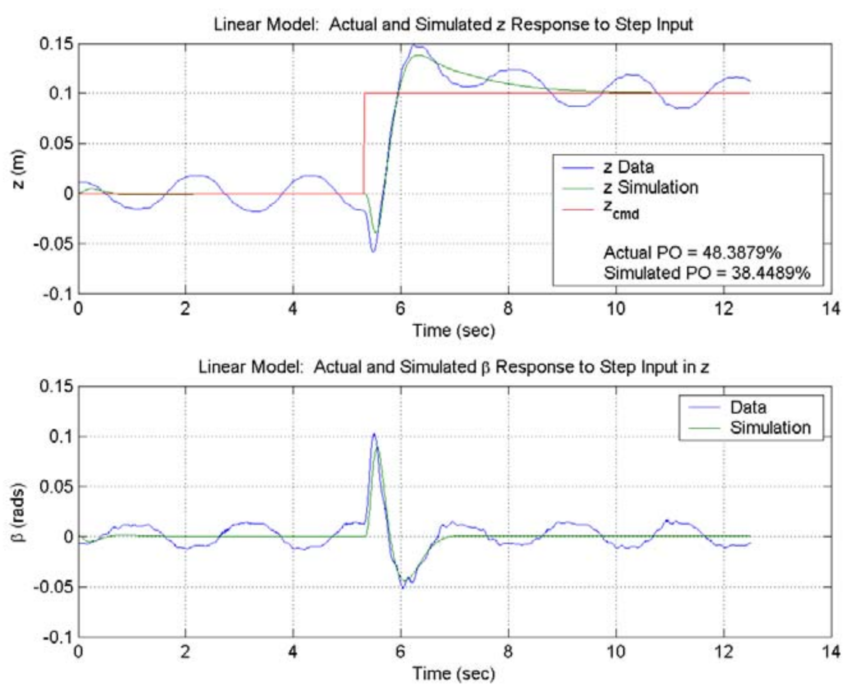


Figure 93: Actual and simulated response to a z_{cmd} step of 0.1m using fully linear model

In the previous two sections (Full-State Feedback Performance and Full State Feedback with Integrator Design), the bandwidth of the system between z and z_{cmd} was calculated from the bode plot of the linear model. However, since this LQR controller is the final controller, its real world bandwidth can be calculated using the DSA. The actual bode plot of the system is shown below in Figure 94.

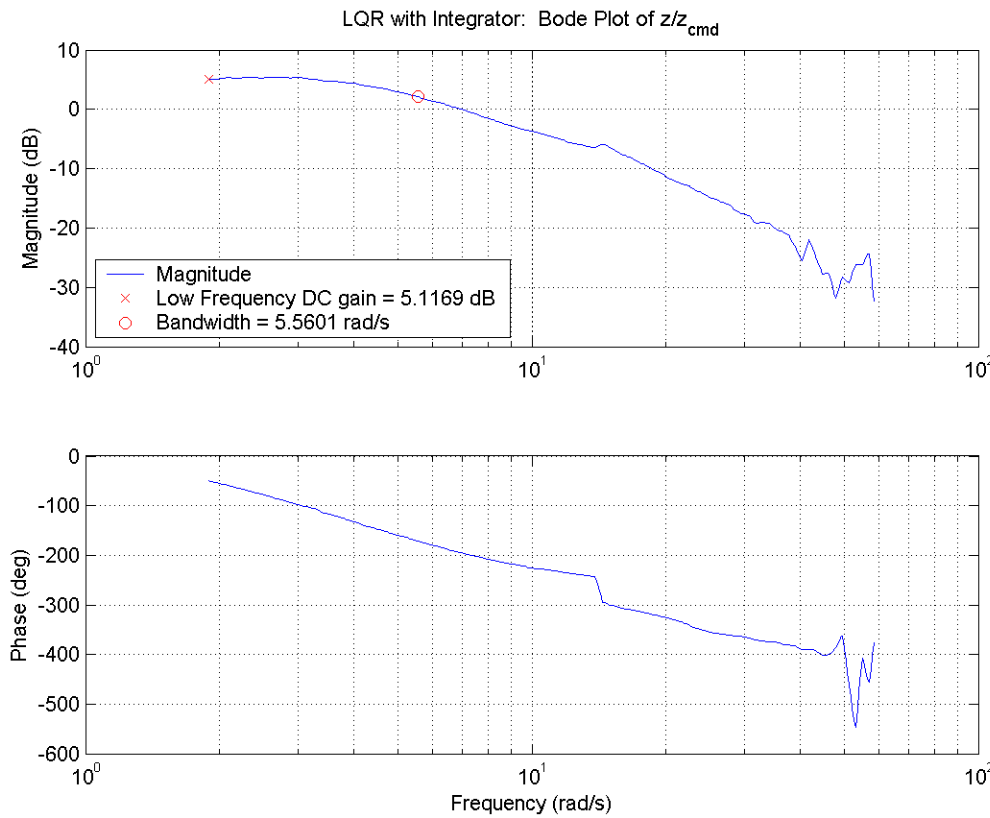


Figure 94: Actual Bode plot of z/z_{cmd} using LQR w/ integrator controller

As can be seen, it appears that the actual bandwidth is about 5.56 rad/s. Another interesting observation to make is that the low frequency DC gain is not zero. This means that at low frequencies, the system is actually amplifying the signals. This can be explained by looking at the bode response of the linear model which is shown below.

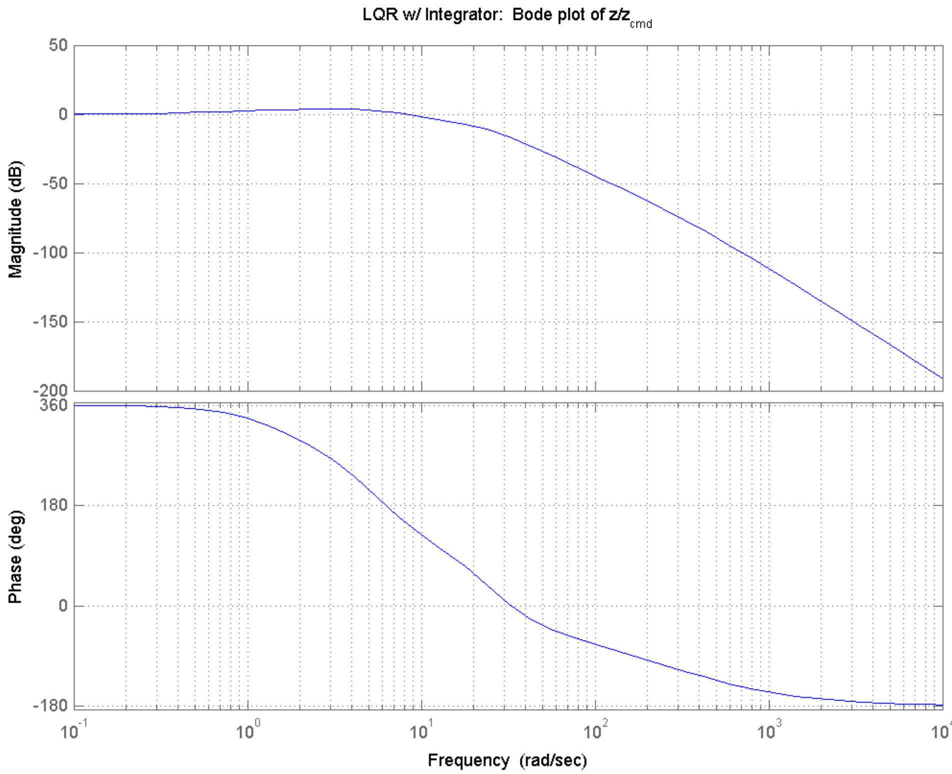


Figure 95: Linear model Bode plot of z/z_{cmd} using LQR w/ integrator controller

As can be seen from the bode plot of the linear system, it can be seen that at extremely low frequencies, the system does have a DC gain of 0. However, as the frequency increases, the magnitude increases. This makes sense considering that this means that the system is amplifying the signal, which corresponds to the system overshooting the desired tracking system. However, in the actual system, the DSA is unable to analyze the system at frequencies lower than 0.3 Hz because at this low frequency, the stiction in the model causes the pendulum to oscillate and therefore, z also oscillates. This generates a signal that is made up of two sin waves. The low frequency carrier wave is the input signal and there is a slightly higher frequency sin wave that corresponds to the z oscillation due to stiction. The DSA is unable to handle this superposition of sin waves since it is inputting a pure sin wave and is expecting a sin wave in return. Therefore, the DSA sin sweep is forced to start at 0.3 Hz. However, this is in the region where the system is amplifying the signal. At low frequencies, the actual DC gain is zero, so the bandwidth of the system using the DSA can be recalculated by looking for where the magnitude reaches –3 dB since the low frequency DC gain is actually zero. This is shown below.

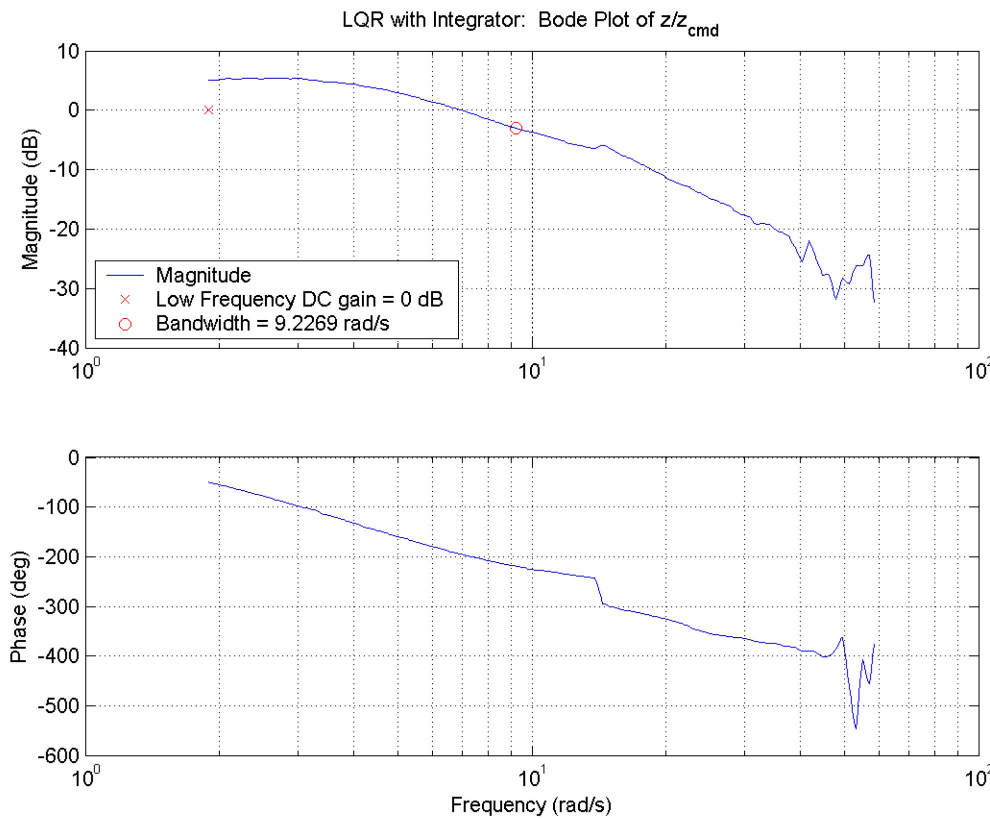


Figure 96: Actual Bode plot of z/z_{cmd} using LQR w/ integrator controller using correct DC gain

As can be seen, in this situation, the simulated bandwidth matches the actual bandwidth very well. The other interesting thing to notice is that at higher frequencies, the actual bode plot deviates significantly from the simulated bode plot. Obviously, the actual bode plot takes non-linear effects into account and the simulated bode plot does not, but the interesting question is which non-linear effects cause this discrepancy. Since the system is moving constantly, the coulomb friction has some effect, but not much. Furthermore, since the pendulum is not rotating much, the centripetal acceleration forces are also not a factor. From the current model, the actual bode plot should match up fairly well with the linear system. The problem is in the equations of motion. In the derivation of the mathematical representation of the system, the amplifier is modeled as a simple gain with a voltage limit to model the max voltage on the motor. However, in actuality, the amplifier has both a voltage limit and a current limit. By watching the amplifier during the DSA sin sweep, it can be seen that at higher frequencies, the current limit light would light up, signaling that the current was saturating. This non-linear phenomenon explains why at higher frequencies, the actual bode plot begins to look so strange and different from the linear model.

The results from the different models are tabulated below in Table 17.

Table 17: Performance characteristics of full state LQR feedback with integrator controller

	Actual System	**Full Non-Linear**	**Non-Linear w/o Coulomb Friction**	**Linear**
Percent Overshoot (%)	48.3879	41.91	38.28	38.45
Settling Time (sec)	INF	3.76	4.24	4.25
Rise Time (sec)	0.216	0.21	0.21	0.21
Bandwidth (rad/s)	9.22	N/A	N/A	11.2

Once again, the simulation seems to accurately predict both the percent overshoot and settling time. Also, the performance in almost every respect increased. It appears that the simulated settling time actually increased, but since neither system actual ever settles to within 2% of the final value, the meaning of this is minimal.

The disturbance rejection capabilities of the LQR with integrator controller can now be evaluated. The actual response of the system when placed under a load of f_1 = 15.56 N is shown below

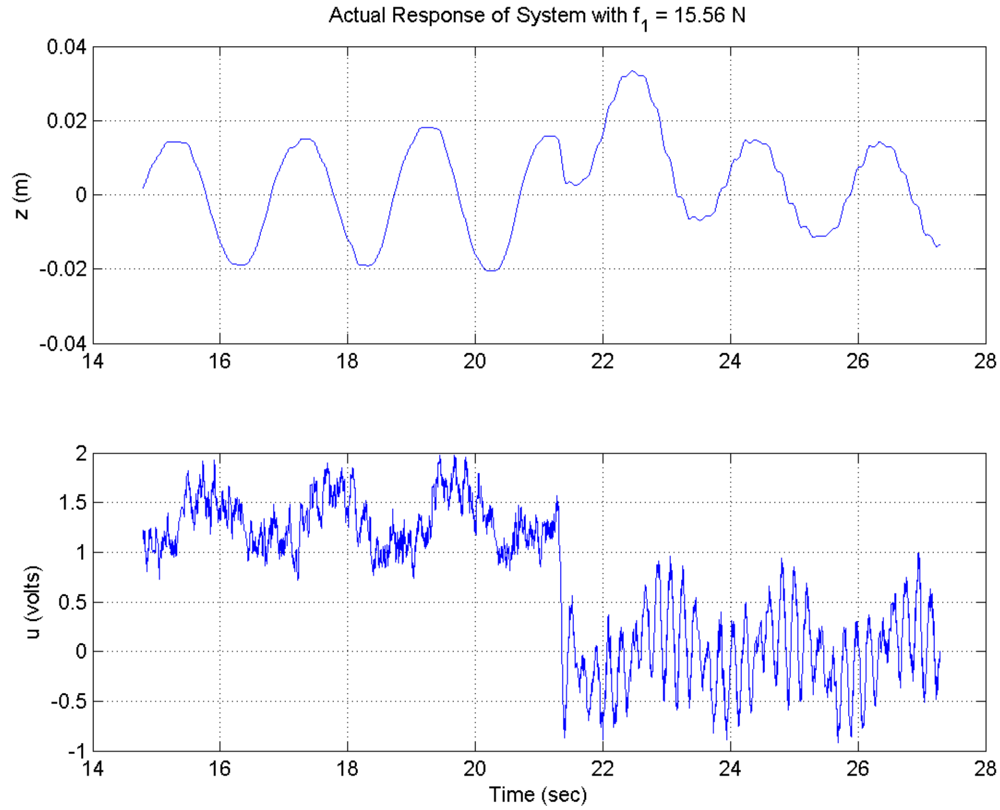


Figure 97: Actual response of system to constant force f_1 = 15.56 N

As can be seen, the load is removed at roughly t = 21 seconds. The integrator appears to be working as desired since the system is still able to track a $z_{cmd} = 0$ signal. The simulated results for the maximum force that can be tolerated by the system is shown below in Table 18.

Table 18: Disturbance rejection of LQR full state feedback with integrator controller

	Actual System	**Full Non-Linear**	**Non-Linear w/o Coulomb Friction**	**Linear**
f_1 max (N)	22.24	10.15	10.23	N/A

There appears to be a slight discrepancy between the actual maximum force tolerable and the simulated maximum force. One possible explanation to this is in the way the force was applied. As stated in the Experimental Procedure section, the force was gradually applied once the pendulum stabilized. However, in simulation, the force is applied as a step function. This might create a large disturbance that could cause instability at a lower value of f_1. If the force f_1 were applied as a ramp in the simulation, it might match the experimental value better.

Lastly, the robustness of the system can be evaluated. The bode plots of the three different loops are shown below in Figure 98 through Figure 100.

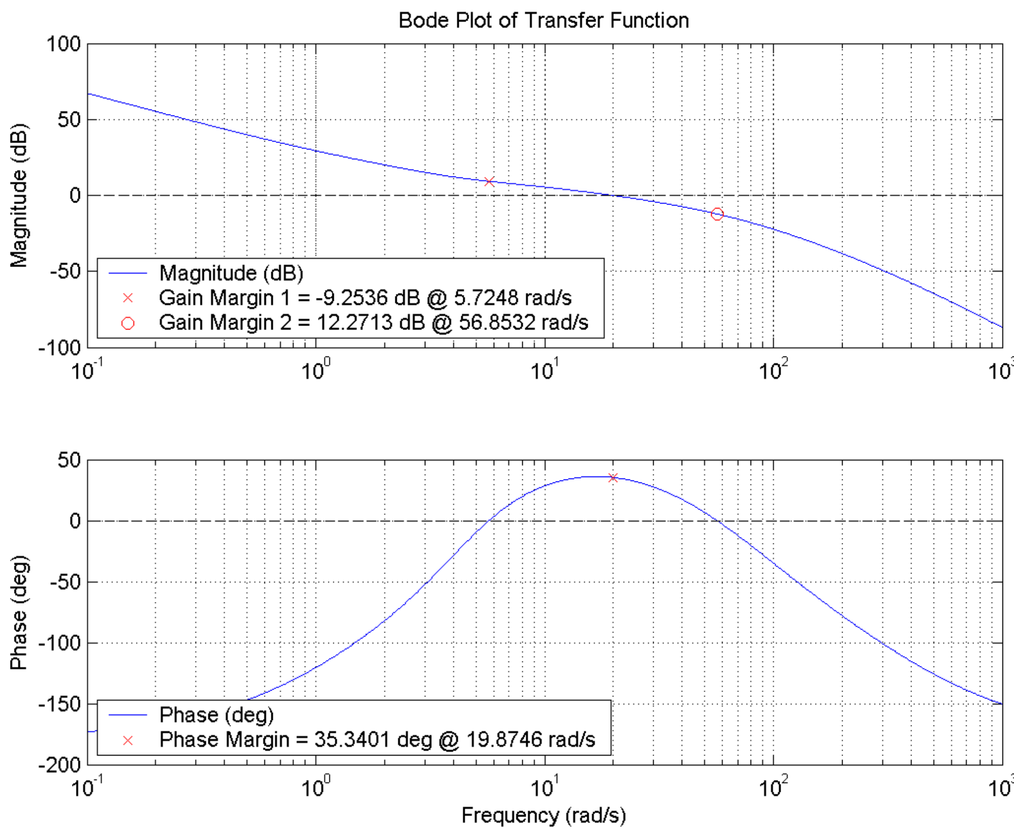


Figure 98: Bode plot of control loop using full state LQR with integrator controller

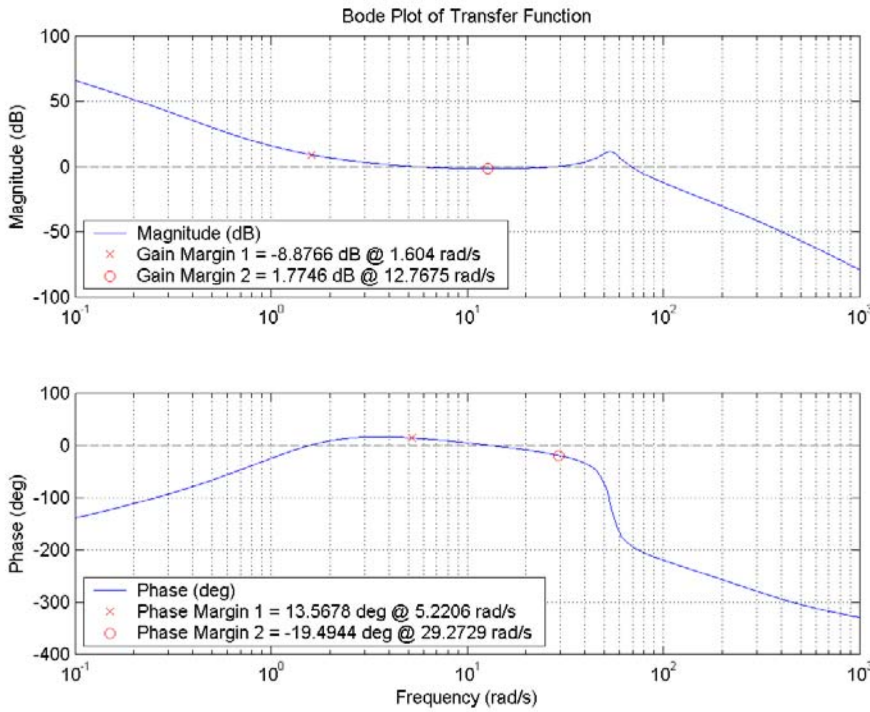


Figure 99: Bode plot of z loop using full state LQR with integrator controller

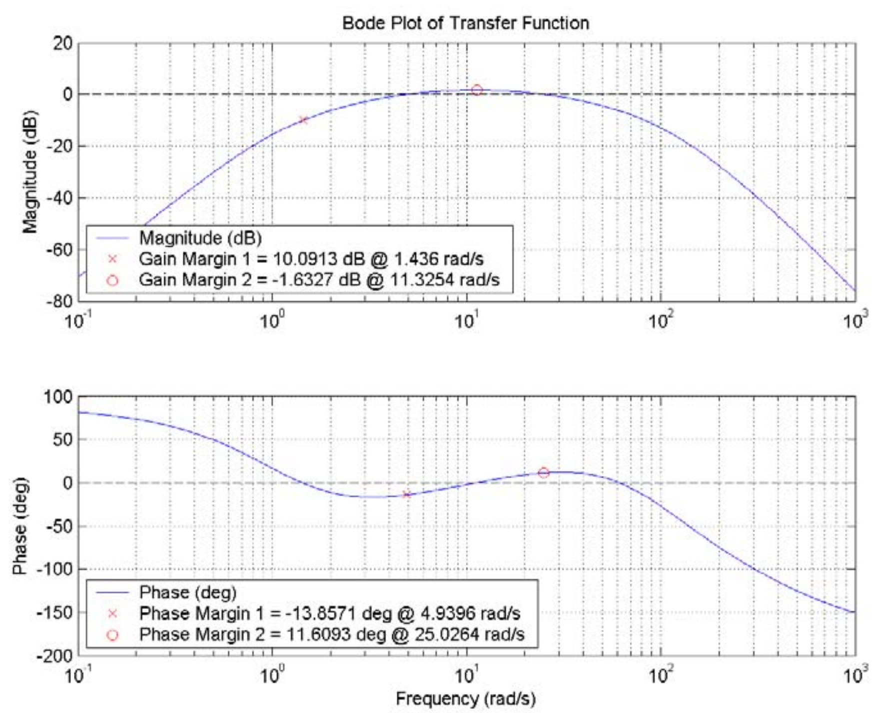


Figure 100: Bode plot of β loop using full state LQR with integrator controller

The gain margin and variation along with the phase margin and variations are tabulated below in Table 19.

Table 19: Robustness of different loop for full state LQR with integrator controller

Loop	Gain Margin (dB)	Phase Margin (deg)	KΔ Range	Time Delay (sec)
Control	-9.25 & 12.21	35.34	0.34 & 4.11	0.031
z	-8.88 & 1.78	-19.49 & 13.57	0.35 & 1.23	0.045
β	-1.63 & 10.09	-13.86 & 11.61	0.83 & 3.20	0.008

There are several interesting observations to make here concerning the system robustness. First of all, by looking at Figure 98 (Bode plot of control loop), it can be seen that there are two gain margins. This is curious due to the fact that in the previous Full State Feedback with Integrator Performance section, the same analysis and the bode plot of the control loop only had one gain margin (Figure 83). In effect, the two controllers are the same (with slightly different gains). However, in Full State Feedback with Integrator Performance section, the analysis was performed assuming that states x_3 and x_4 were measured and directly available for feedback. However, in the LQR with Integrator Performance section, the analysis was performed using pseudo-derivatives to estimate x_3 and x_4 from the states x_1 and x_2, respectively. At high frequencies, these pseudo-derivatives add phase lag and attenuate the signal. Therefore, the phase decreases at higher frequencies, which causes a second gain margin.

Observer Performance

The *Simulink* model used to simulate the system with the LQR full state with integrator controller and the Kalman filter to reconstruct the system states is shown below in Figure 101. The LQR full state with integrator controller is the same controller shown in Figure 87 in the LQR with Integrator Performance section with some modifications. Namely, the states x_3 and x_4 are not estimated using pseudo-derivatives. Instead, the estimated states are used from the Kalman filter. Another difference is that the gain k_5 is not set to zero since the state $\hat{x}_5$ is available for feedback. Instead, gain designed using the LQR method of k_5 = -0.0171 is used.

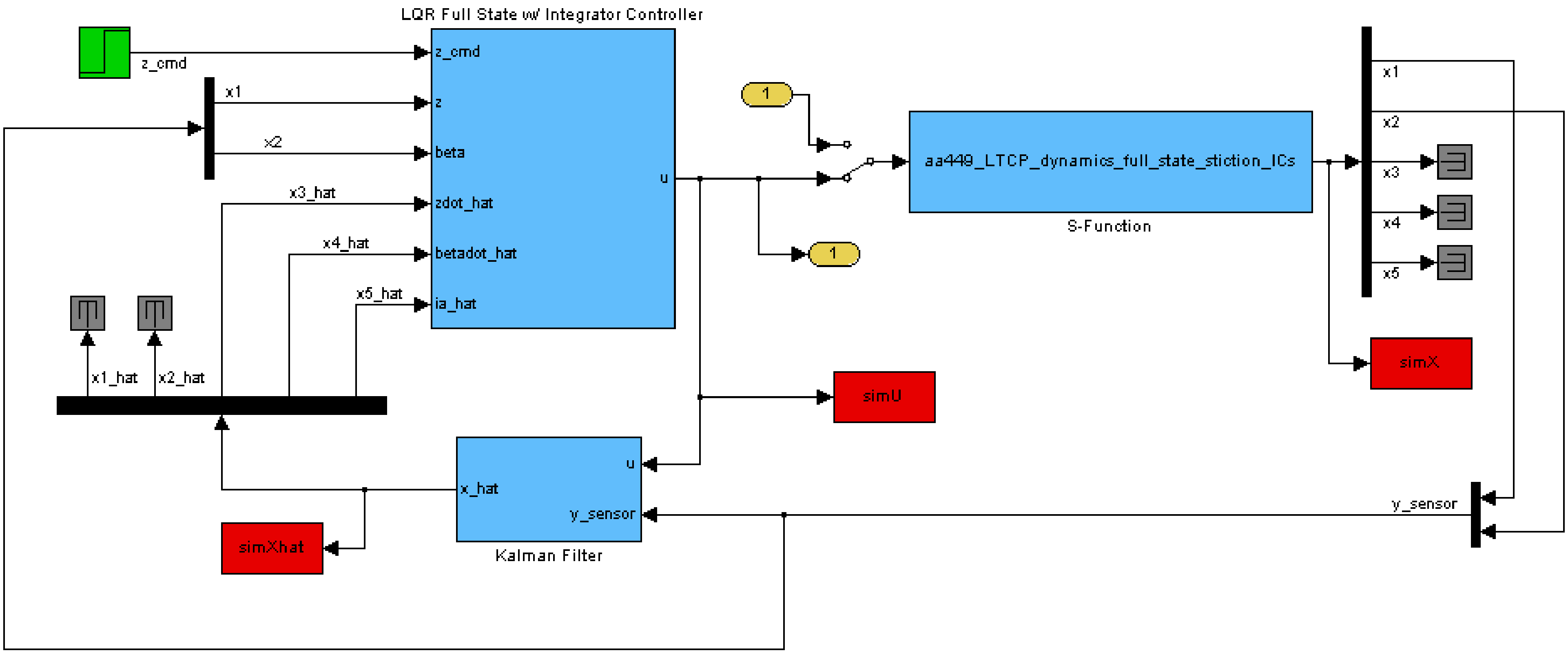


Figure 101: LQR full state feedback with integrator control system with Kalman filter

Notice in Figure 101 that not all of the estimated states are used for feedback. Namely, $\hat{x}_1$ and $\hat{x}_2$ are not used because the actual measured x_1 and x_2 are available for feedback. The other interesting thing to notice is that the Kalman filter does not estimate the integrator state. This is due to the fact that x_6 is directly linked to x_1, which is directly measured. Therefore, it x_1 is directly available for feedback, so is x_6 and estimating this state would be counter productive.

First, the eigen-assignment method can be evaluated. For convenience, the observer gain matrix calculated using the eigen-assignment method is repeated here

```
L =

       4.1216        1.674
        2.285       5.1596
        249.3      -493.23
      -495.05       1018.7
  -1.743e+005  3.4406e+005
```

The system's response to initial conditions can be evaluated. The system can be given some initial conditions other than zero, and using the observed states as feedback, the controller can try and stabilize the system. The initial conditions that the actual system is subjected to is shown below,

$$\overline{x}(0) = \begin{pmatrix} 0.1 & -0.05 & 0.1 & 0.1 & 1 \end{pmatrix}^T$$

The simulated response using the eigen-assigned L matrix is shown below in Figure 102.

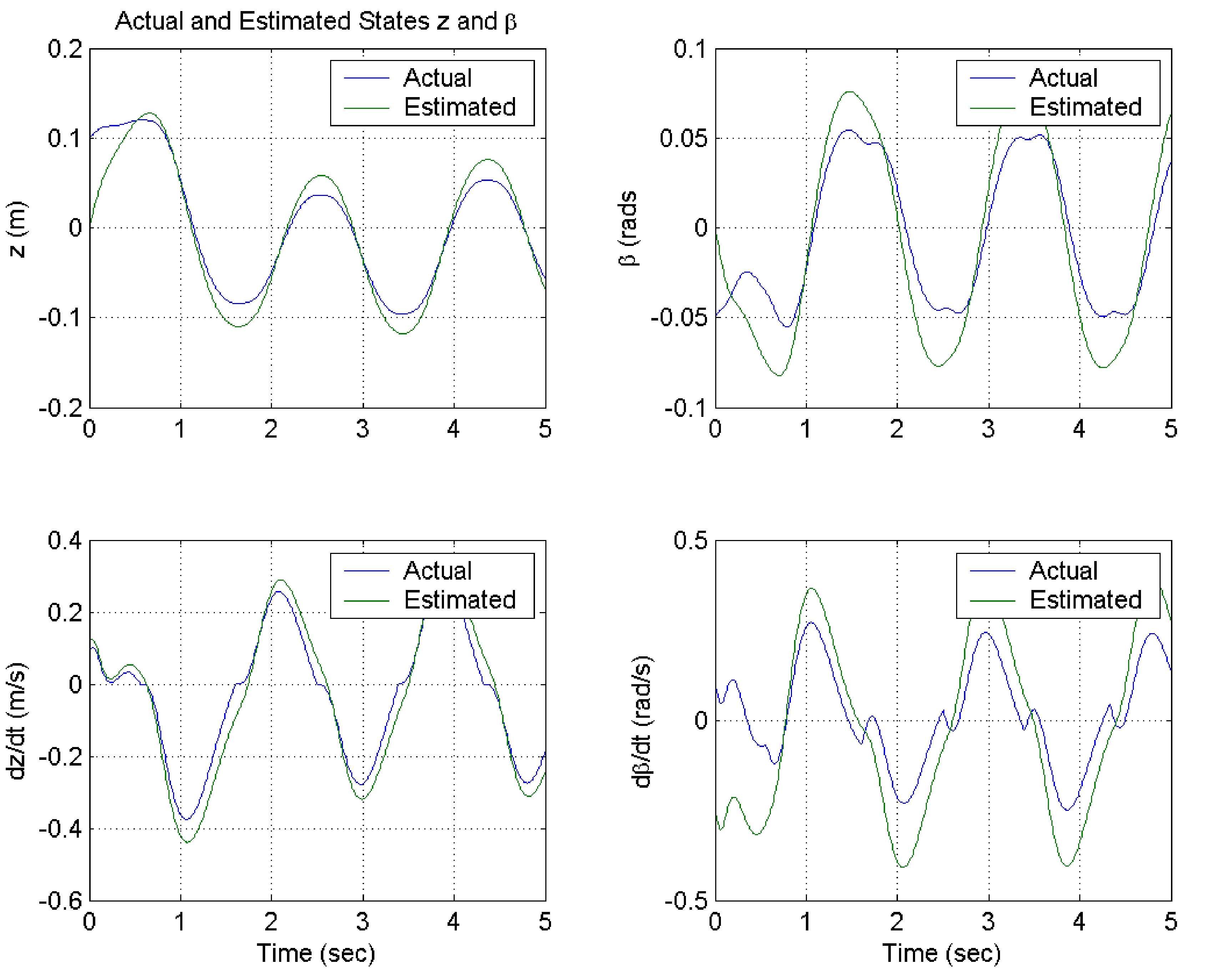


Figure 102: Simulated actual and estimated states using eigen-assigment method for deriving L

As can be seen from Figure 102, it appears that the observer estimates the actual states $x_1 - x_4$ fairly well. The estimated states start at zero, which is incorrect, but they quickly converge with the actual states as the error attempts to go to zero. As can be seen, it does not perfectly estimate the system states. This is because the estimator is assuming a linear model of the plant. However, the simulation is using the full non-linear model with coulomb friction. This explains some of the discrepancies.

Although it appears that states $x_1 - x_4$ are estimated quite accurately using the eigen-assignment method to find L, but x_5 is not accurate at all. The estimated and actual x_5 state is shown below in Figure 103.

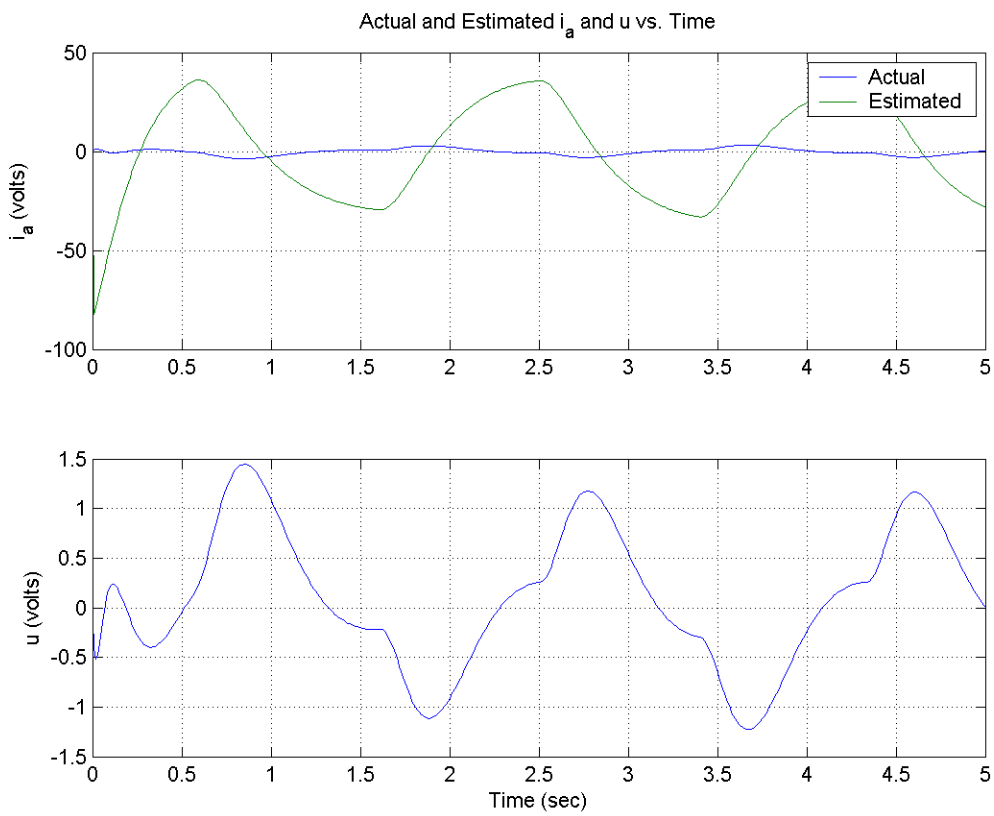


Figure 103: Estimated and actual x_5 using eigen-assignment method to solve for L

As can be seen, the estimator provides an extremely poor approximation of x_5. This is unacceptable considering the entire goal of the estimator is to obtain this state (x_3 and x_4 can be acceptably estimated using pseudo-derivatives). Despite this inaccuracy, the system is able to maintain stability since the gain k_5 is extremely small (k_5 = -0.0171). In the event that this gain was higher, this estimator would not work since it would probably make the system unstable. Therefore, another more accurate method is required.

The Kalman gain matrix derived using the *lqe* function is repeated here for convenience.

```
L =

    8.6202     -7.6327
   -7.6327      21.042
    66.283     -110.31
   -116.09      250.51
    -36.47      43.783
```

The simulated response of the actual and simulated x_5 state subject to the same initial conditions is shown below in Figure 104.

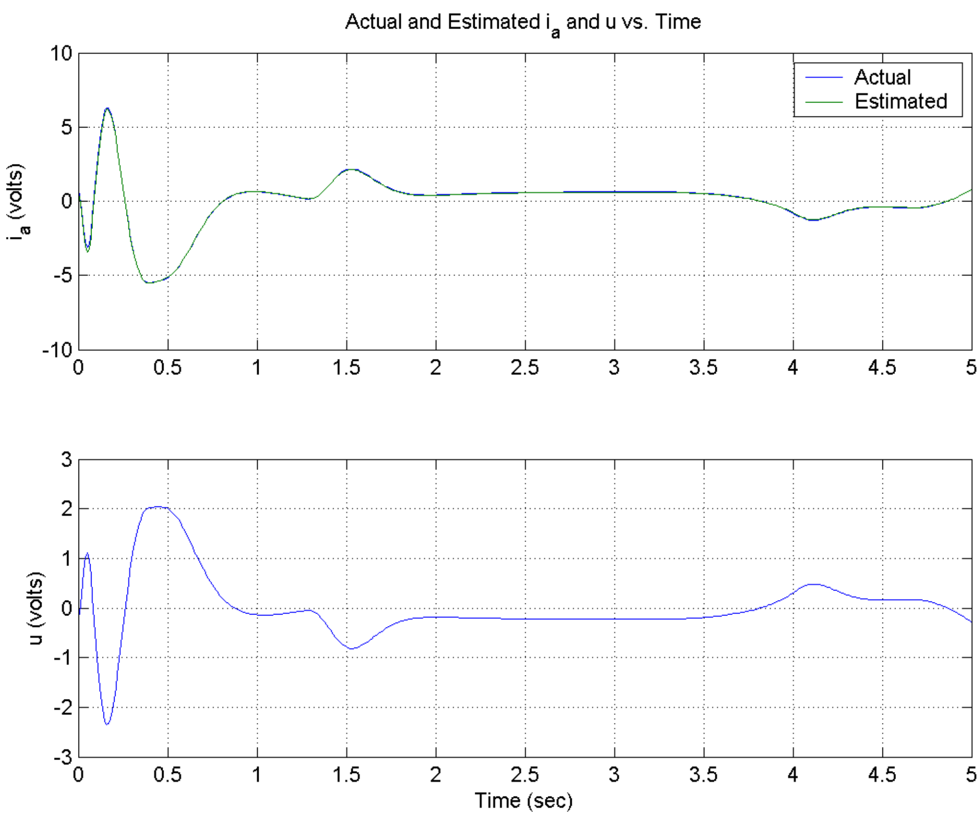


Figure 104: Estimated and actual x_5 using lqe method to solve for L

As can be seen, the estimated and actual x_5 match much better, the two lines are almost only top one another. The actual and estimated states $x_1 - x_4$ are shown below in Figure 105.

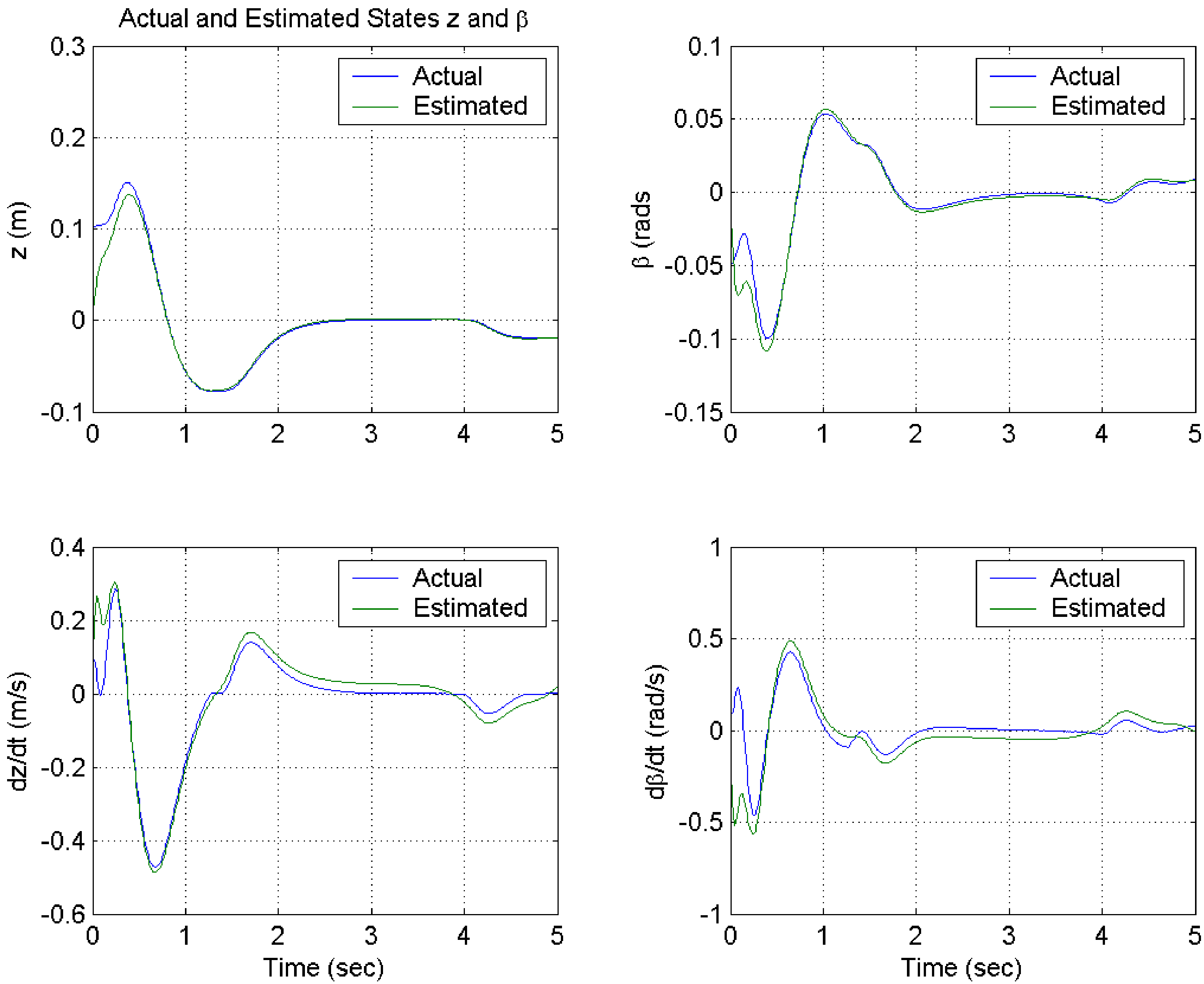


Figure 105: Simulated actual and estimated states using lqe method for deriving L

Recall that the goal of the observer is to reconstruct the state x_5. Notice from Figure 104 that it appears that x_5 is directly related to the control voltage u. This makes sense considering the fast dynamics associated with x_5. At steady state, the current on the motor, x_5, can be directly related to the control voltage u using Ohm's law. Namely,

$$x_5 = \frac{K_a u}{R_1 + R_A}$$

Equation 40

Equation 40 is only valid at steady state when the dynamics of the current have died away. However, it was already shown that the pole that is mainly associated with x_5 is located at s = -393.15. Therefore, the dynamics are very fast and it may be possible to reconstruct a fair approximation of x_5 using Equation 40, which states that x_5 is simply u multiplied by a static gain. The system's response of x_5 to the previously stated initial conditions is shown below in Figure 106.

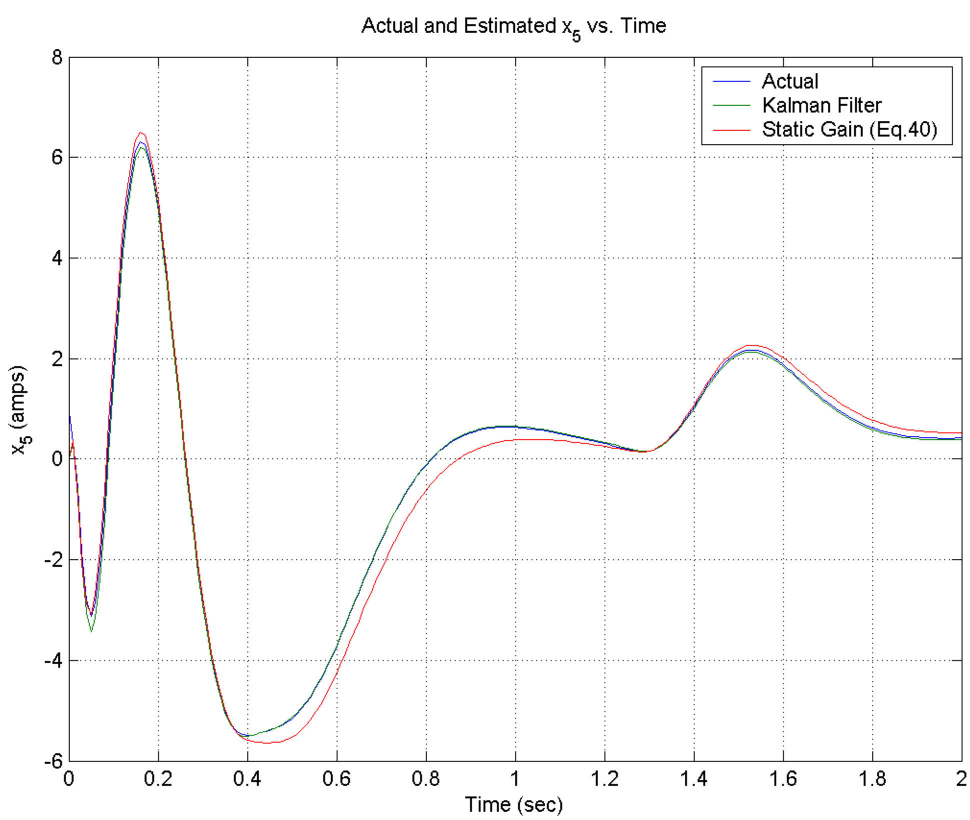


Figure 106: Actual and estimated x_5 using Kalman filter and estimated x_5 using static gain

Zooming in on the section where the error appear the most apparent yields Figure 107.

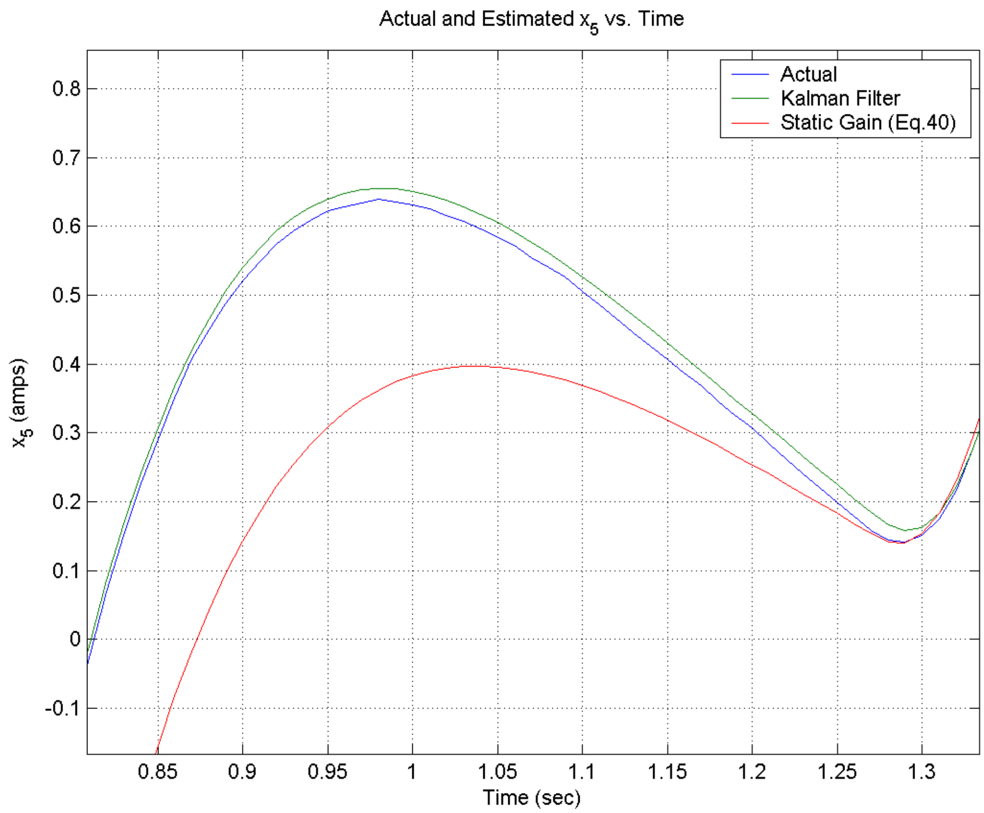


Figure 107: Zoomed in on section where discrepancy exists

As can be seen, it appears that x_5 can be accurately estimated using a static gain on the control voltage. In the worst case, it appears that the discrepancy is only 0.22 amps. The Kalman filter still appears to be more accurate, but not by much.

It should be noted that the Kalman filter is still very useful considering that it provides built in filtering on the states x_3 and x_4, but from the above analysis, it appears that x_3 and x_4 could be obtained using pseudo-derivatives and x_5 can be estimated very accurately using a static gain on the control voltage.

The robustness of the system with regard to its tolerance to uncertainties in the plant can be evaluated. The bode plot of the system is shown below

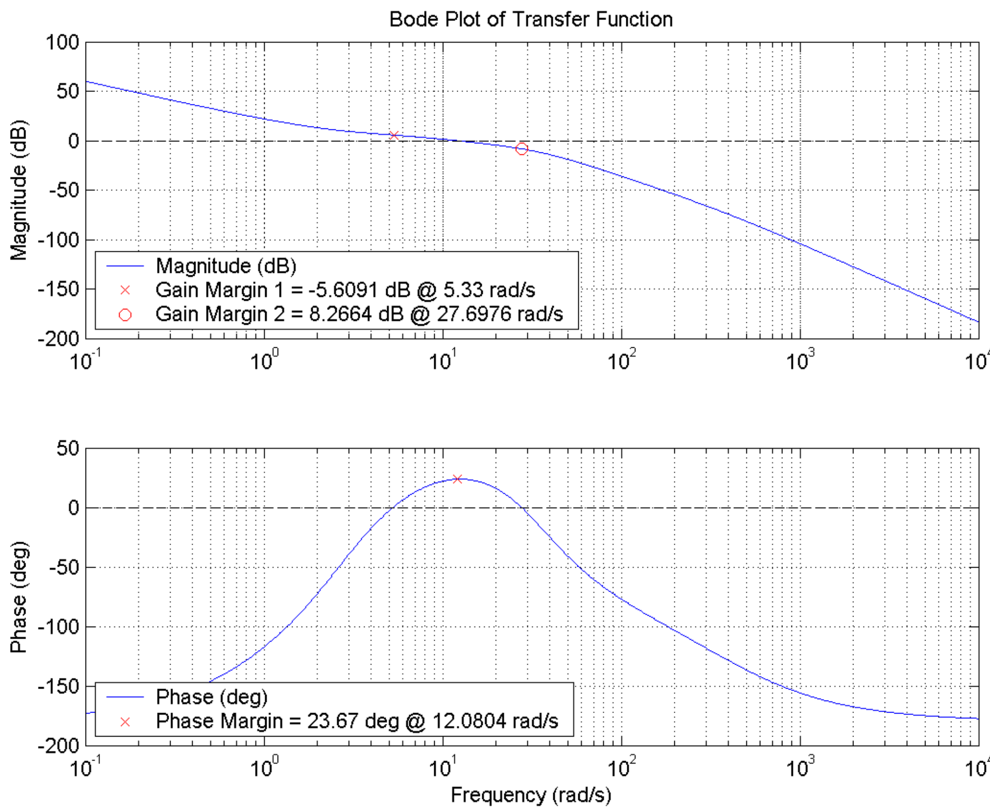


Figure 108: Bode plot of control loop for observer system

The interesting observation to make here is that the gain and phase margin using the observer is significantly less than the gain and phase margin for the same set of LQR gains, but using the pseudo-derivatives to estimate x_3 and x_4 (Figure 98). This is due to the fact that the observer is very sensitive in terms of robustness. It can be shown that the gain and phase margin can be recovered using a method called Loop Transfer Recovery. This involves changing the W_o matrix (or scalar in this case). As W_o approaches infinity, the original gain and phase margin should be recovered. The bode plot of the control loop with increasing values of W_o and the bode plot of the control loop without the observer (assuming that all states are directly available for measurement) is shown below in Figure 109.

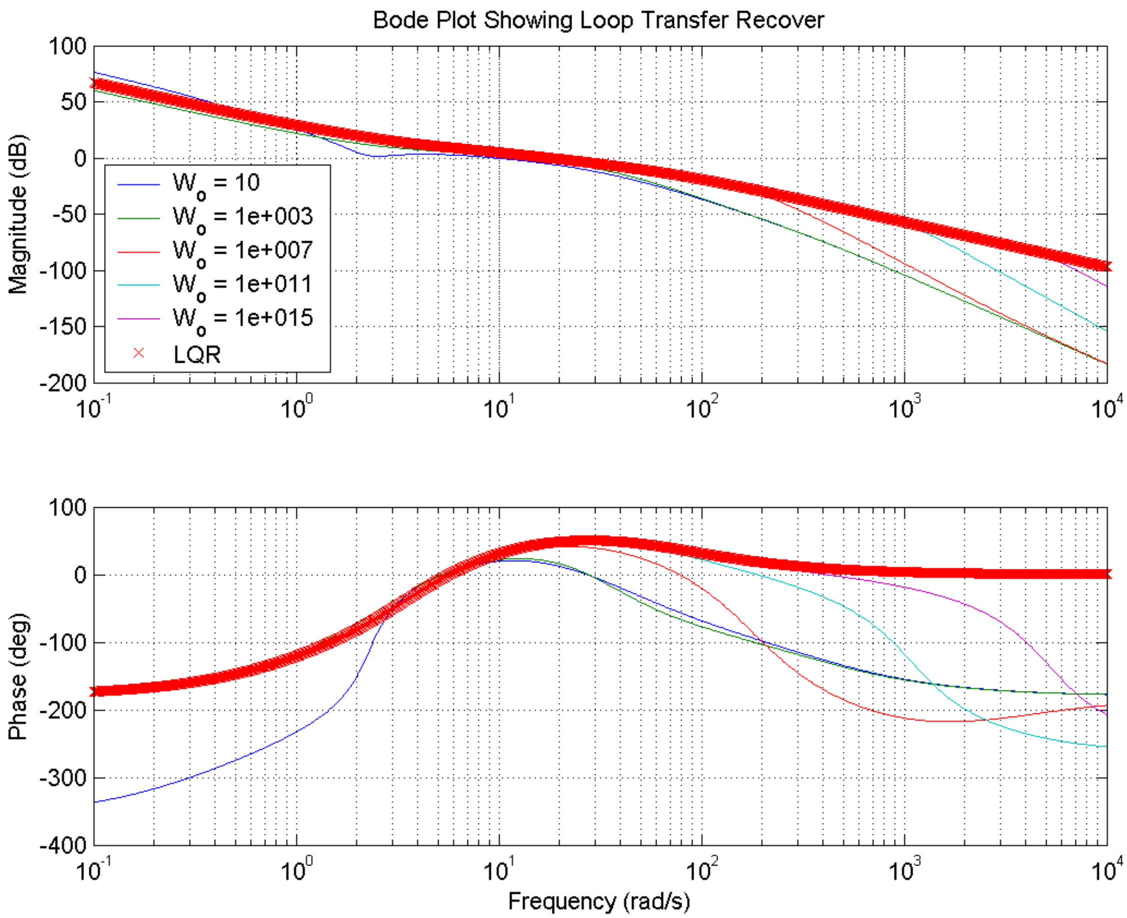


Figure 109: Observer loop transfer recovery

The gain and phase margin of the system assuming that all the states are directly available for measurement (thick red line) are given below

```
Gm_plant =

   -9.3950

>> Pm_plant

Pm_plant =

   47.0324
```

The corresponding gain and phase margins for the increasing values of W_o are shown below.

```
>> Gm

Gm =

   -2.9690   10.7471
   -5.6091    8.2664
   -8.5481   15.4626
   -9.2042   28.7238
   -9.3538   41.7408

>> Pm

Pm =

   19.1916
   23.6700
   40.5323
   45.6744
   46.7753
```

As can be seen, the phase margin recovers and begins to approach the gain margins predicted. Also notice that the Kalman filter adds lag to the system, which results in a second gain margin.

Pump-up Control Performance

All of the previous controllers function on the premise that the pendulum is in the vertical position. It has been shown that these controllers are designed using the linearized model about the up position. If it were to deviate too far from this location, the controller would not function. However, the vertical position is unstable, and without control, the stable equilibrium point is in the down position. Clearly, the pendulum must be moved from the stable down position to the up position so the controller can function. One method is to simply pick the pendulum up using a human. This turns out to be the most time and energy efficient method. However, in the event that this is not available, the pendulum can be inverted using a pump-up controller. The controller used to pump up the pendulum is shown below in Figure 110.

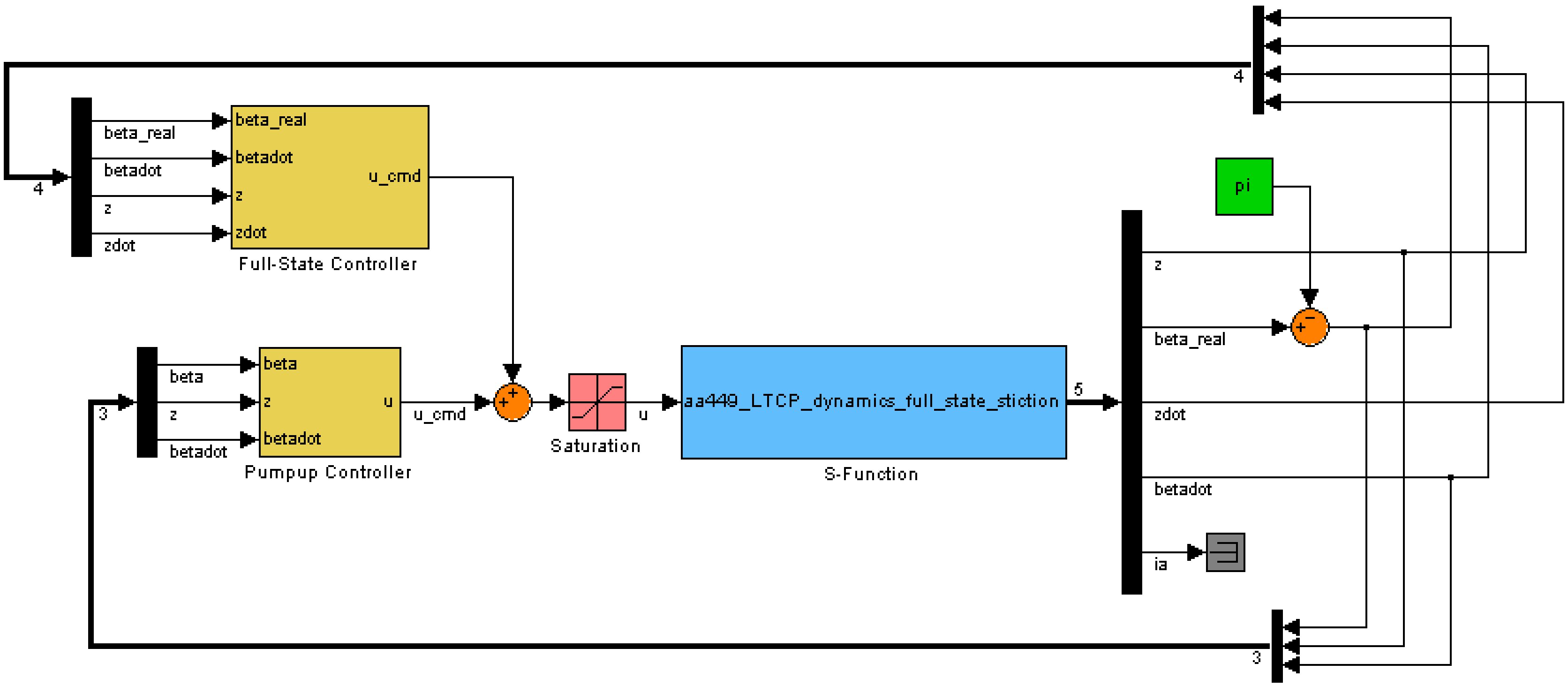


Figure 110: *Simulink* model used to pump-up and catch pendulum

When the simulation is started, the pump up controller provides the control sequence to move the pendulum from the down position to the up position. The Full-State controller does not output any voltage until the angle β travels through +/-180 degrees. The pump up controller subsystem is shown below in Figure 111.

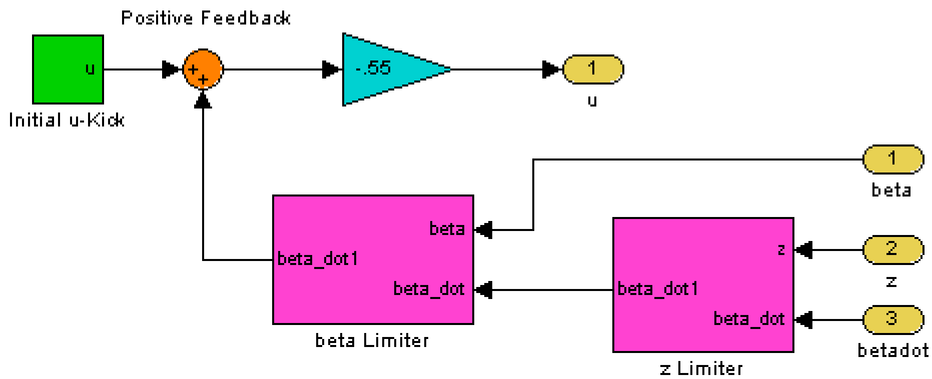


Figure 111: Pump-up controller

The pump-up controller is modeled after a common swing. It is a positive feedback system, which feeds back dβ/dt, which makes the system unstable. The β-limiter only allows the signal to go through if the pendulum angle β is between the specified angle[7]. For example, if a cone angle of 90 degrees were specified, the positive feedback would only be in effect when the pendulum is below the horizontal line. The cone angle definition is shown below in Figure 112.

[7] For detailed schematics and descriptions of special function blocks, see Appendix B.

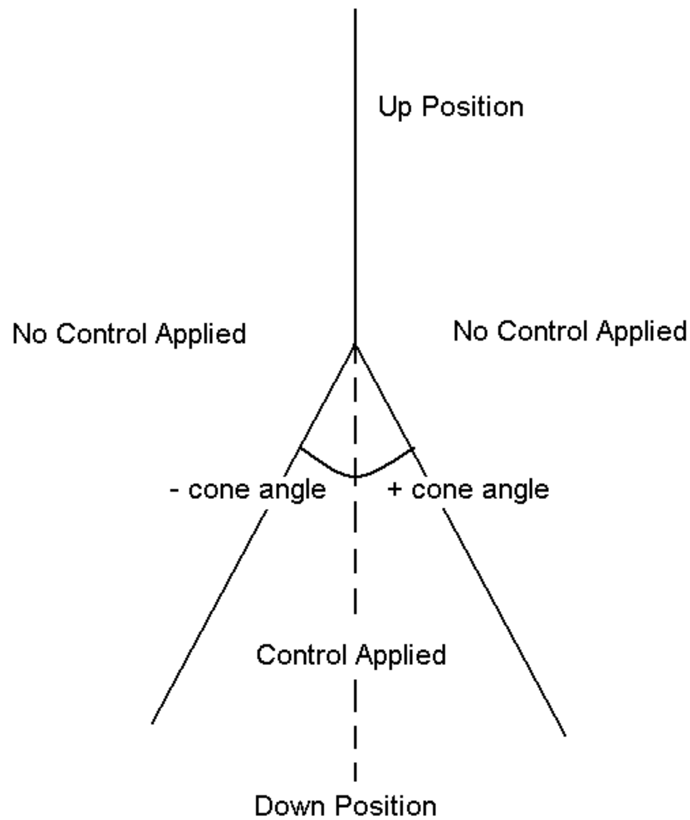


Figure 112: Cone angle diagram

As can be seen, the input only occurs when the pendulum is near the bottom.

The z-limiter only allows the signal to feed through if the absolute value of z is between the specified value. This is simply to model the effect of the hard stops on the actual system.

Once the pendulum is in the vertical, the full state controller can attempt to catch the pendulum. The full state controller subsystem is shown below in Figure 113.

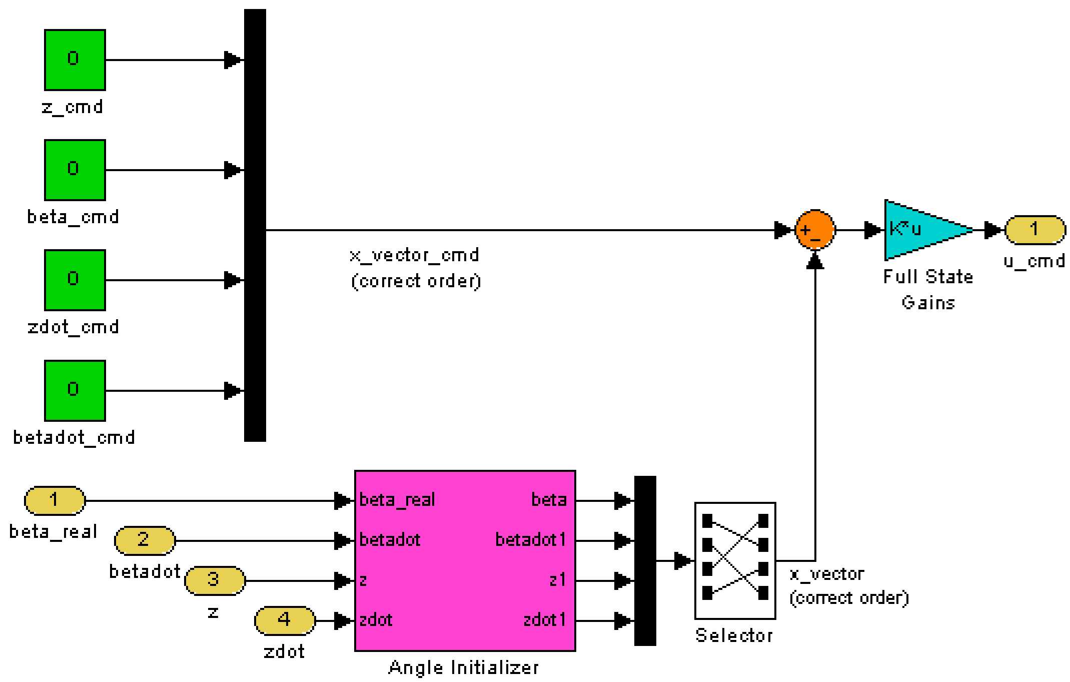


Figure 113: Full State controller used to catch pendulum once it is vertical

In order to make it so the full state controller does not attempt to balance the pendulum until it is vertical, the angle initializer is used. The angle initializer outputs all zeros until the pendulum reaches vertical. At this point, it feeds through all of the system states and sets β to zero at the vertical position[8].

The system can be simulated using a cone angle of 41.25 degrees is shown below in Figure 114.

[8] For detailed schematics and descriptions of special function blocks, see Appendix B.

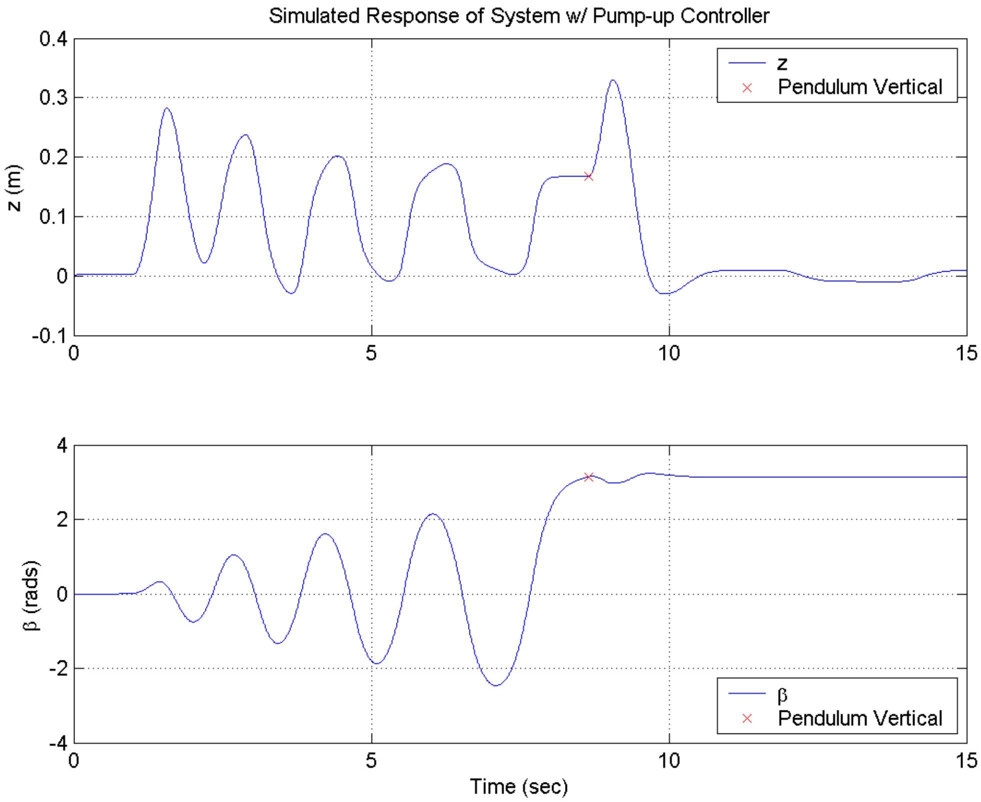


Figure 114: Simulated response of system using pump-up controller

As can be seen, the pump up controller is able to raise the pendulum to vertical after about 8.5 seconds. Notice how the system is unstable initially when the unstable pump up controller is functioning and then becomes stable after the pendulum reaches vertical and the stable full state controller takes over. The control voltage applied during this time can also be plotted as shown below in Figure 115.

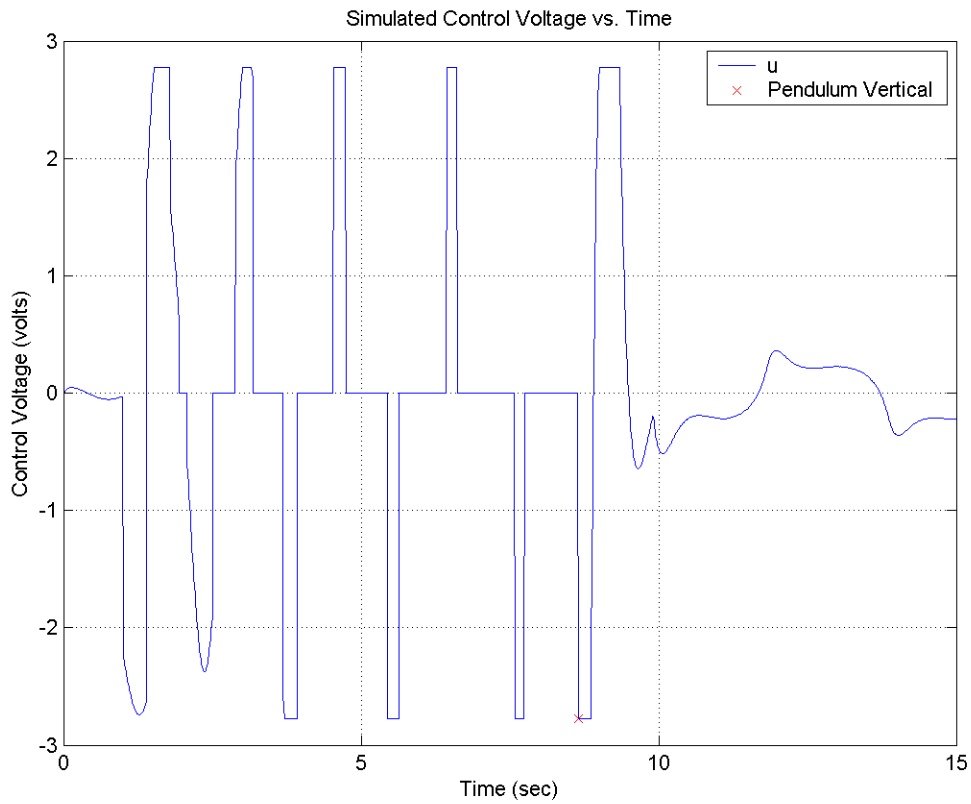


Figure 115: Simulated control voltage vs. time during pump up and catch

Notice that before the pendulum reaches the vertical position, the pump up controller is applying small voltage commands at certain intervals when the pendulum is within the specified cone angle. Then, as soon as the pendulum is vertical (denoted by red x), the full state controller takes over and stabilizes the system. The total energy of the system (neglecting translational kinetic energy) during the maneuver can be plotted in 3D as shown below in Figure 116.

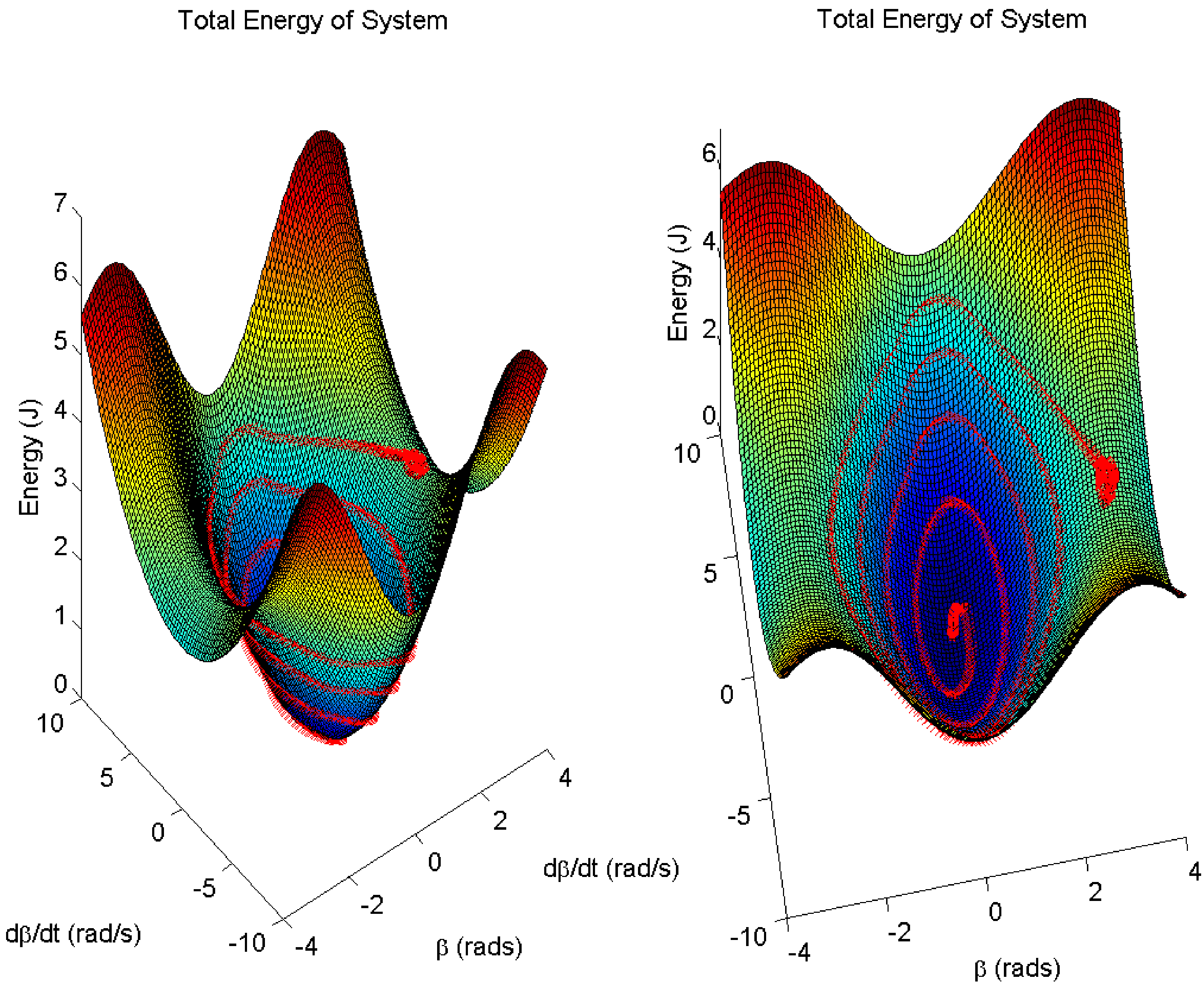


Figure 116: Total energy of system during pump-up sequence

As can be seen, the system is initially stable from an energy point of view since it starts in the local minimum point. As the energy is added to the system, the system moves to the saddle point, which is unstable.

Theoretically, since gravity is a conservative force field, the minimum work that needs to be done against gravity is simply the change in potential energy from the down position to the up position.

$$W_{\min} = 2l_1 m_1 g = 2.82J$$

Equation 41

Another interesting observation to make here is that the amount of energy expended by the motor to get the pendulum to the vertical position can be calculated from the control voltage on the motor. Since the resistance of the system remains constant, the power at every time can be calculated using

$$P = \frac{(K_A u)^2}{R_A + R_1}$$

Equation 42

The power can then be integrated with respect to time to obtain the amount of work used to move the pendulum to the vertical position. The minimum is equal to 143.7 J. Notice that this is significantly more than the minimum energy required shown in Figure 116.

The efficiency of this pump up controller can be measured by dividing the minimum work required by the work expended. In this situation, this controller is 1.96% efficient. This leads to two important conclusions. One conclusion is that the friction takes out energy from the system as the pendulum rotates and the cart translates. This guarantees that the efficiency can never be 100% since the viscous and coulomb friction forces from the rotation of the pendulum will always take energy out of the system. The other conclusion is that energy transferred from the motor to erecting the pendulum is a function of the pendulum position. For example, the transfer of energy from the motor to the pendulum is 0% efficient of the control is applied when the pendulum is in the horizontal position. In this situation, all of the energy from the motor goes into the translation of the cart/pendulum system, but no energy goes into rotation and therefore, no energy goes into moving the pendulum against gravity.

Clearly, in order to achieve the most efficient energy transfer from the motor to the pendulum rotation, the energy should be an impulse to the motor when the pendulum is directly in the down position. However, an impulse cannot be applied, and therefore, the best pump-up controller scheme can be determined by varying the cone angle and computing the work done by the motor. The cone angle is varied between 20 and 90 degrees and the work expended by the motor to erect the motor can be calculated.

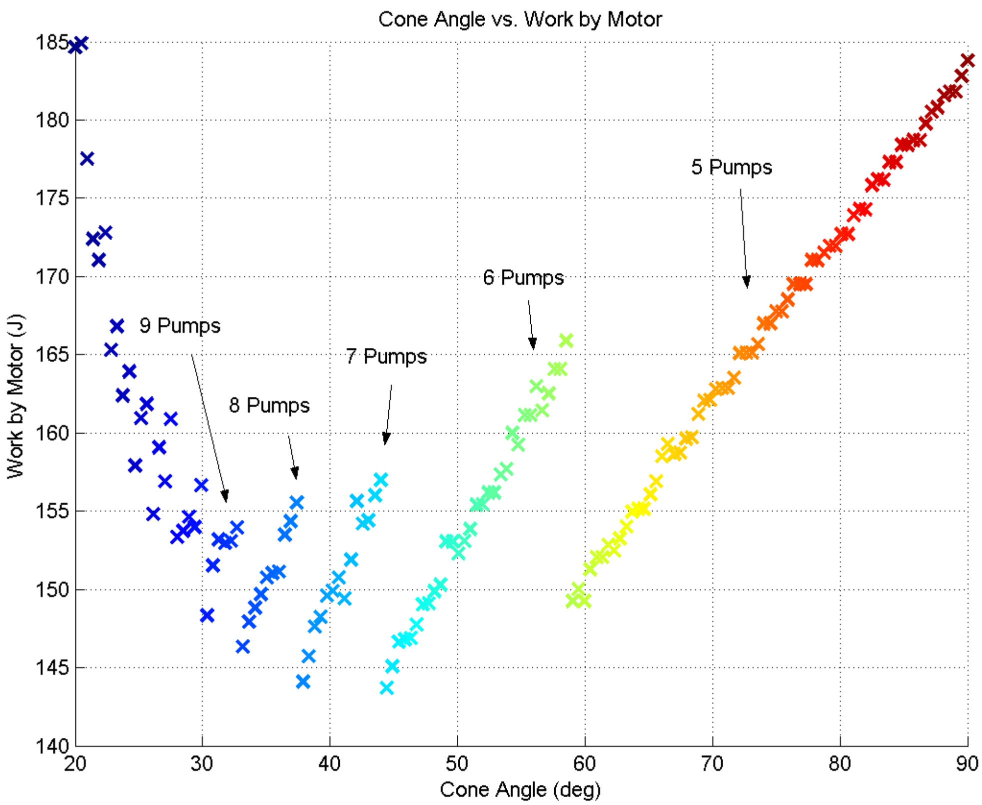


Figure 117: Simulated work done by motor vs. cone angle

As can be seen, this is an interesting plot. As the cone angle increases, there are distinct bands that are generated. This shows that the work done by the motor to erect the pendulum is non-linear. This makes sense, each of these bands represent a discrete number of "pumps" required to erect the pendulum. A pump is defined as how many times the pendulum passes through the down position. For example for a cone angle between 44.5 and 58.4 degrees, the system requires 6 pumps to get the pendulum vertical. The interesting observation to make is that even though the number of pumps is the same, there is a wide range of work required for the same numbers of pumps. At a minimum, it requires ~143.7 J of work to erect the pendulum with a cone angle of 44.5 degrees. However, if the cone angle is increases, the work increases, but the numbers of pumps remain the same. The extra energy done by the motor is converted into an initial condition (dß/dt) when the pendulum reaches vertical. Since these initial conditions are not desirable, there are several discrete cone angles where the initial conditions are minimized. The lowest point on each of the bands correspond to minimal initial conditions where the full state controller would have the best chance to catch the pendulum. The time to vertical vs. the work done by the motor can also be plotted which is shown below in Figure 118.

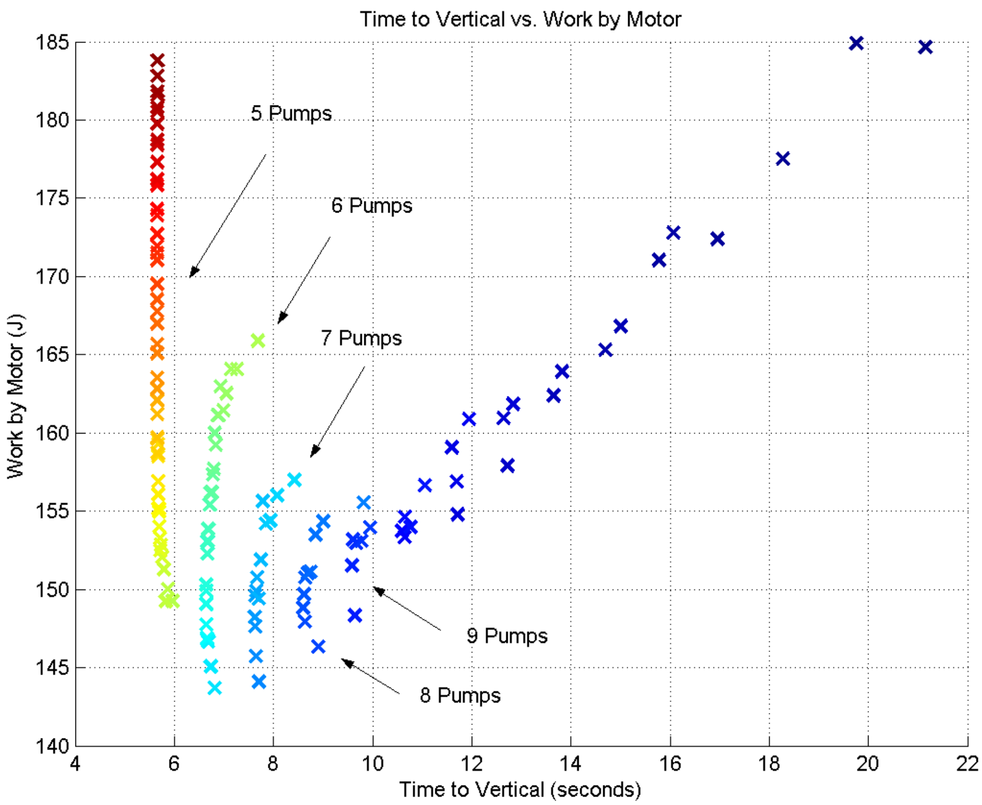


Figure 118: Simulated time to vertical vs. work done by motor

As can be seen here, there is an interesting relationship between time to vertical and work done by motor. Clearly, a minimum time maneuver does not imply minimum work expenditure and vice versa. Using this type of controller (positive feedback), it appears that the minimum energy required is ~143.7 J. This leads to a 1.96% efficiency. However, this is clearly not a minimum time maneuver, as it requires 6.81 seconds. The minimum time maneuver has been solved by Bryson et al. and requires only 2 pumps and less than 2 seconds. Although this optimized pump up sequence was not realized, another lab group used the guess and check method to derive a set of step functions to erect the pendulum in 2 pumps.[9] The pump up has an estimated time to vertical of 2.5 seconds. However, it requires that the motor be completely saturated for the entire duration. This yields a work done by the motor of 192.3 J with an efficiency of 1.46%. As can be seen, even though this sequence takes less time, it takes more work.

[9] See Holzinger, Marcus; Carter, John; and Taplin, Lisa report.

This analysis is helpful since it illustrates the differences in cost function definition. In the laboratory situation, energy is abundant and therefore, a more applicable pump up controller would be similar to the Bryson controller that minimized time. However, this is not an energy efficient system. If the DC motor was operated by a battery of other finite energy source, a controller that minimized energy expenditure would be more appropriate.

XII. Conclusions

As described in the introduction, the purpose of this laboratory was to explore a number of different control approaches to the problem of controlling a pendulum in the inverted position. The initial approach was to use a classically designed controller that was developed using root locus. This then evolved into a full state design. The disturbance rejection of the full state design was then improved by implementing an integrator on the z state. To add some degree of optimality to the controller design the full state control gains, which had previously been determined through pole placement, were then developed using a LQR approach. In all of these full state implementations, the velocity states were assumed to be known while in reality they were developed by using pseudo-derivatives. This was then taken into account in the final implement design by using an observer-based design.

Each of these approaches represented an evolution of the previous approach with the classical design serving as an initial starting block. By building upon the knowledge that was learned in each of the previous designs the controllers evolved such that they had not only better performance but also were based upon a more accurate estimation of what the states of the system were actually doing.

In all, the project was successful and a number of viable controls with good performance and robustness were developed.

Appendix A
Derivation of State Equations

Appendix B
Custom Blocks of Interest

The State Initializer Logic Block

The purpose of this block is to assist the implementation of control by setting all of the states to zero until the pendulum is placed in the full upright position by the operator. By triggering off of the β value, it uses a series of switches to set the all values to zero until β reaches either π or $-\pi$. It then passes through Beta dot, Z and Z dot unaffected. However, because β is initialized in xPC by the initial position of the pendulum, down in this case, and the control laws are developed with $\beta = 0$ being the upright position, the state initializer logic block either adds or subtracts π from the value of β depending on the direction that the pendulum was raised. Once the pendulum has reached the upright position for the first time then State Initializer Logic Block will never again set the states to zero until the simulation is reset. This block is shown below in Figure 119.

Positive Trigger Block

This block is an integral part of the State Initializer Logic Block described above. Essentially, once the trigger point on the input signal is reached, the block continuously sends out a non-zero signal even if the input signal subsequently drops below the trigger point. This block is shown below in Figure 120. The Negative Trigger Block is functionally the same except for the fact that it has a different trigger point.

Z Limiting Virtual Stop

The Z Limiting Virtual Stop takes as inputs both the control voltage and the z position. While the z position is inside the limits set by the block the control voltage signal is allowed to pass through unaffected. When the z signal exceeds the limits set within the block, however, the control voltage is set to zero. In this manner the block acts as an electronic stop cutting power to the motor before the cart slams into the physical stops at the end of the track.

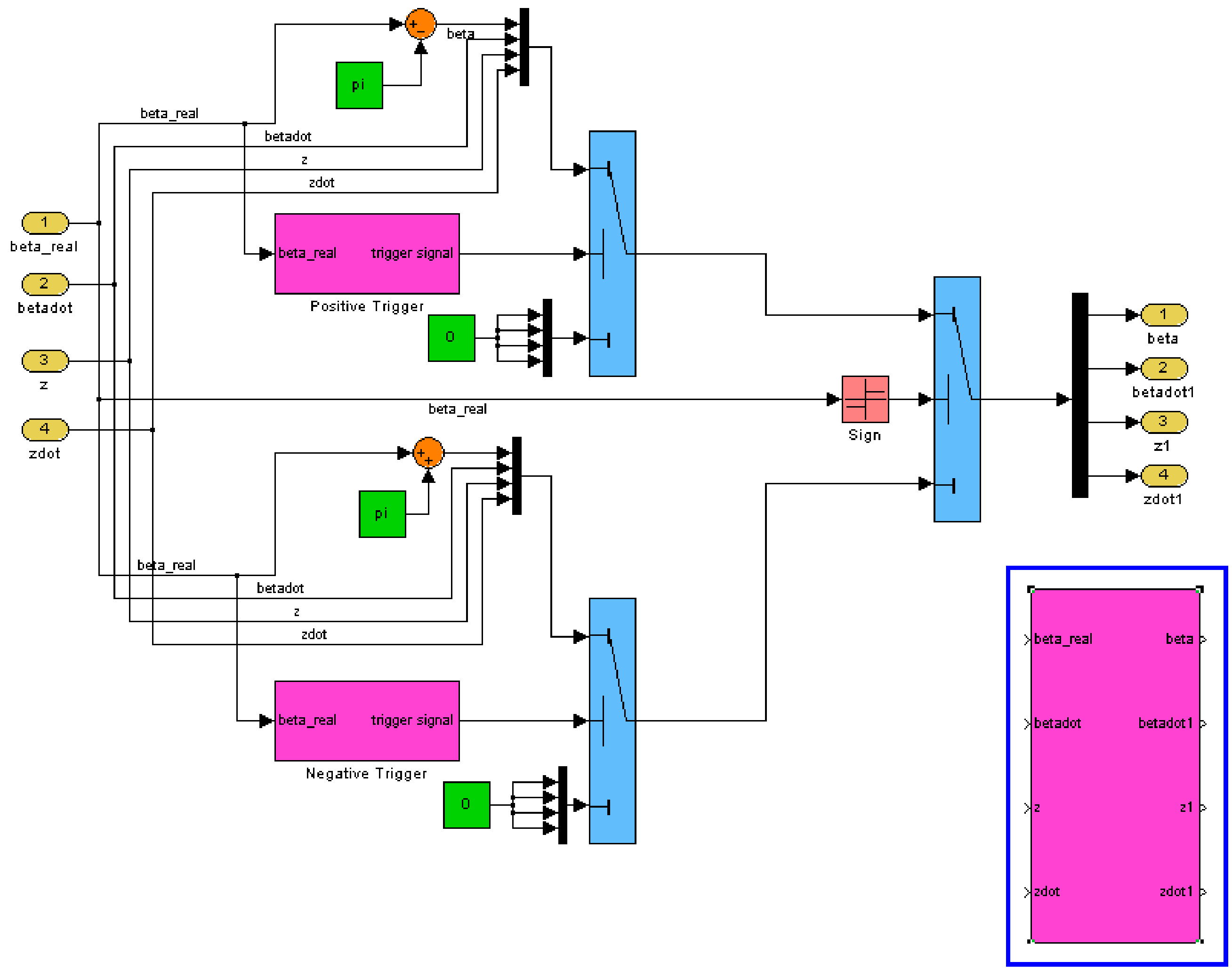


Figure 119: The State Initializer Logic Block

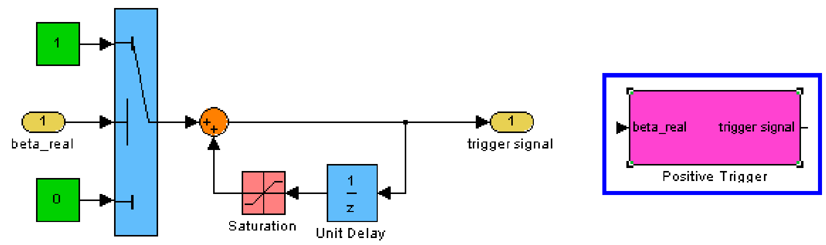


Figure 120: The Positive Trigger Block

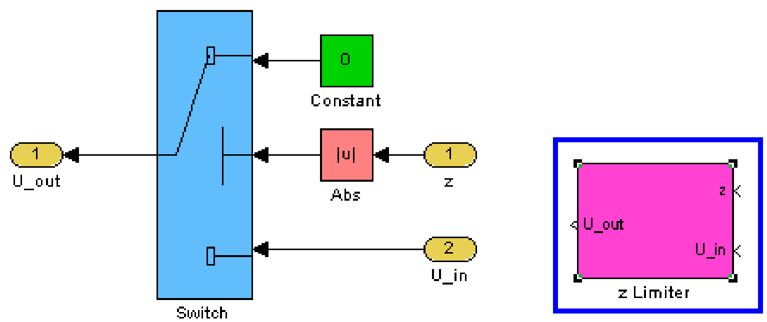


Figure 121: The Z limiting virtual end stop.

Appendix C

DSA Programming

Bibliography

- Hass, Brian; Lum, Christopher; Stephens, Amanda; Reisner, Travis. Modeling and Design of a DC Motor Control System. Feb. 21, 2003.

- Bryson, Arthur E. Jr. Dynamic Optimization. Addison Wesley Longman, Inc. 1999.

- Ly, Uy-Loi. AA/EE-449 Design of Automatic Control Systems Class Notes. April 17, 2003.

- 2 & 4 Channel Quadrature Encoder Input Boards User's Manual