U.S.N.A. --- Trident Scholar project; no. 288 (2002)

STEERING CONTROL COMPENSATION OF ACCELERATING VEHICLE MOTION

by

Midshipman Steven Robert Burns, Class of 2002
United States Naval Academy
Annapolis, MD

_____
(signature)

Certification of Advisors Approval

Assistant Professor Richard T. O'Brien, Jr.
Weapons and Systems Engineering Department

_____
(signature)

_____
(date)

Assistant Professor Jenelle Armstrong Piepmeier
Weapons and Systems Engineering Department

_____
(signature)

_____
(date)

Acceptance for the Trident Scholar Committee

Professor Joyce E. Shade
Deputy Director of Research & Scholarship

_____
(signature)

_____
(date)

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (FROM - TO)
07-05-2002		xx-xx-2002 to xx-xx-2002

4. TITLE AND SUBTITLE
Steering Control Compensation of Accelerating Vehicle Motion
Unclassified

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Burns, Steven R. ;

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME AND ADDRESS
US Naval Academy
Annapolis, MD21402

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS
,

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES

14. ABSTRACT
See report.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Public Release	84	email from USNA, (blank) lfenster@dtic.mil
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703767-9007 DSN 427-9007

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATE COVERED
	7 May 2002	

4. TITLE AND SUBTITLE
Steering control compensation of accelerating vehicle motion

5. FUNDING NUMBERS

6. AUTHOR(S)
Burns, Steven Robert

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Naval Academy
Annapolis, MD 21402

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
Trident Scholar project report no. 288 (2002)

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
This document has been approved for public release; its distribution is UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT:
A dangerous and common situation during highway automotive use is the emergency turning/braking (ETB) maneuver. An ETB maneuver is defined by full application of the brakes combined with a significant steering input from the driver. To date, there has been no known development of a dedicated driver assistance controller for this common situation. The proposed ETB controller assists the driver in maintaining control over the vehicle during the ETB maneuver. The ETB controller and similar vehicle subsystem development is limited as a result of the high material cost and space intensiveness of full-size vehicle testing. Normally, physical testing of automotive components is cost and space prohibitive at small research facilities. The use of scale-model vehicle testing in automotive engineering is a recent innovation and, therefore, the development of a scale-model platform is a significant contribution to this field. The scale-model platform components include a treadmill, scale-model vehicle assembled from kit, vision system, desktop computer, and sensor suite. The treadmill and scale-model vehicle simulate the road and vehicle, respectively, while the camera system provides position and orientation data to the vehicle controllers. Utilizing sensor data, the computer maintains the vehicle's position on the treadmill in a closed-loop system, while the treadmill speed controller functions in a parallel open loop. A mathematical model of a vehicle has been developed and a scale-model vehicle platform constructed. The ETB controller prototype has been designed using Linear Quadratic optimal control methods and simulated and will assist the driver in controlling the vehicle during the ETB maneuver.

14. SUBJECT TERMS
Scale-model vehicles, Vehicle dynamics, Advanced vehicle control

15. NUMBER OF PAGES
84

16. PRICE CODE

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

Abstract

A dangerous and common situation during highway automotive use is the emergency turning/braking (ETB) maneuver. An ETB maneuver is defined by full application of the brakes combined with a significant steering input from the driver. To date, there has been no known development of a dedicated driver assistance controller for this common situation. The proposed ETB controller assists the driver in maintaining control over the vehicle during the ETB maneuver.

The ETB controller and similar vehicle subsystem development is limited as a result of the high material cost and space intensiveness of full-size vehicle testing. Normally, physical testing of automotive components is cost and space prohibitive at small research facilities. The use of scale-model vehicle testing in automotive engineering is a recent innovation and, therefore, the development of a scale-model platform is a significant contribution to this field.

The scale-model platform components include a treadmill, scale-model vehicle assembled from kit, vision system, desktop computer, and sensor suite. The treadmill and scale-model vehicle simulate the road and vehicle, respectively, while the camera system provides position and orientation data to the vehicle controllers. Utilizing sensor data, the computer maintains the vehicle's position on the treadmill in a closed-loop system, while the treadmill speed controller functions in a parallel open loop.

A mathematical model of a vehicle has been developed and a scale-model vehicle platform constructed. The ETB controller prototype has been designed using Linear Quadratic optimal control methods and simulated. The effects of various vehicle parameters have been studied using the computer simulation.

The system concept tested in this project will assist the driver in controlling the vehicle during the ETB maneuver, decreasing the likelihood of accident and increasing driver control over the vehicle, by controlling the aspects of the ETB maneuver that are beyond the driver's control, and normally go uncontrolled.

Keywords

Scale-model vehicles, Vehicle dynamics, Advanced vehicle control

Table of Contents

1 Introduction

The development of new automotive technology is difficult for small research facilities because road tests with full-scale vehicles are expensive and require large test tracks and durable sensing equipment. To remedy this situation, scale-model vehicles have been used to bridge the gap between computer simulation and full-scale testing.[1]

A scale-model vehicle platform enables the testing and rapid prototyping of driver assistance controllers.

The Emergency Turning and Braking Maneuver (ETB) maneuver requires the driver to steer the vehicle to move laterally while applying the brakes fully. This maneuver can destabilize vehicle dynamics. The goal of this project is to develop a successful control system that limits the driver's ability to send dangerous inputs to the vehicle control surfaces. This type of control system will increase driver safety without taking control of the vehicle away from the driver.

Previous work at the University of Illinois demonstrated the viability of scale-model vehicles in the development of automotive technologies and design methodology. The apparatus in this project differs from research at University of Illinois in that it is smaller-scale, less expensive, and easier to implement.

This project is divided into several topics: scale-model vehicle platform, three degree of freedom model of a vehicle, controllers for the scale-model vehicle platform, a preliminary design for the driver assistance control system using optimal control state feedback methods.

Modeling the system in three DOF gives a starting point for developing both the scale-model vehicle controllers and driver assistance controllers. By starting with a model of the

[1] Sean N. Brennan, *Modeling and Control Issues Associated with Scaled Vehicles*, Masters Thesis, The University of Illinois, Urbana, Il, 1999, 11.

system, implementing the system on the scale-model vehicle platform becomes a matter of tuning a controller instead of fully developing it with cumbersome physical components.

The scale-model vehicle platform controllers must maintain the vehicle's position in the center of the treadmill, simulating the driver during normal driving conditions. The vehicle will require a longitudinal and lateral controller. Each controller contributes one input to the vehicle's two inputs, throttling and steering.

Using optimal control techniques to implement the ETB controller will allow a more encompassing design procedure while assuring that the best possible controller for the given weights is utilized. Optimal control will also decrease system hardware by allowing some values to be estimated instead of measured.

Developing an ETB controller will demonstrate the viability of using a controller to attenuate driver input, in order to prevent the driver from steering the vehicle beyond its dynamic limits. If the concept test is successful further development of the ETB will occur beyond the scope of this project.

2 Objective

To design and build an Automotive Scale-Model Vehicle (ASMV) platform to test driver assistance control systems. Begin development of a specific controller to assist the driver in controlling the vehicle during the ETB maneuver to concept test the viability of the control system.

3 Background

The results of this project possess significant importance, not only for the automotive industry and the average driver, but also for the Naval Academy. Many drivers have been faced with the need to simultaneously apply the brakes and maneuver the car in order to avoid an accident. The feeling of helplessness in the face of grave danger underscores the need to increase the study of this area of automotive dynamics and control. Models incorporating variable longitudinal velocity have not been investigated, historically, to the same degree as the constant longitudinal velocity case. As a result the vehicle's response during acceleration or braking is not understood as well, and, currently there is no system dedicated to aid the driver during the ETB situation.

This project is a small piece of the momentous undertaking of implementing the Automated Highway System (AHS) to greatly reduce driver fatalities and optimize travel on highway systems.[2] In the AHS, each car's individual automated control system will be mated with a larger overall control system for the entire roadway, leading to smooth coordination of vehicles in transit to their destinations without delays or accidents. Vehicle automation allows a properly designed control system to greatly improve vehicle response by responding much faster and more predictably than a human. Furthermore, the system will have the luxury of sharing information with all other computer "drivers", increasing coordination and lessening the chance of highway

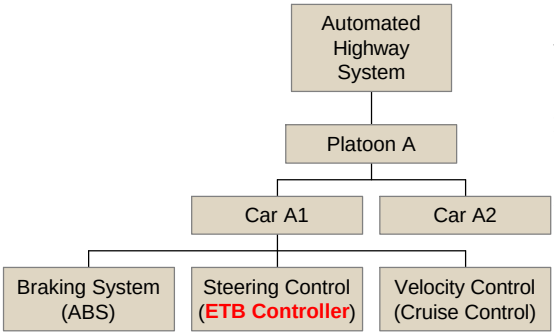


Figure 3.1: AHS Concept

[2] Elizabeth A. Bretz, "Boston Builds a High-IQ Roadway," IEEE Spectrum, August 2000, pg. 47-52.

casualties. The integration of overall AHS structure down to the individual vehicle subsystems including the proposed ETB controller is illustrated in Figure 3.1.

The technological concerns and challenges associated with the implementation of the AHS project are far outweighed by the psychological tribulations associated with convincing human drivers to cede control of their vehicles to a computer. Even though computers already control a number of today's vehicle subsystems.[3] The Anti-lock Braking System (ABS) is a prototypical example of a system that assists the driver by controlling the brakes with a much greater precision than the driver possibly could. ABS prevents the vehicle's wheels from locking up during heavy braking. ABS "pumps" the brakes at a very high frequency, allowing the wheels to continue turning until the vehicle's forward motion stops. When it was first introduced, ABS was criticized because it was believed to lengthen stopping distances in some situations. Over time, it became apparent that ABS's poor performance was due to drivers' improper use of the system. Once drivers were educated on its correct use, ABS became a successful vehicle automation system that greatly increased driver safety. Statistically, the use of ABS has led to lower traffic related fatalities and as a result insurance companies offer lower premiums to drivers who own cars that employ ABS.[4] The same misconceptions and education difficulties follow AHS, only on a much larger scale. This project contributes to the development of AHS by starting the development of a prototypical system that successfully assists the driver in maintaining control of the vehicle during ETB situations.[5]

The Naval Academy directly benefits from results of the project. The vehicle test bed will be used not only in future projects but also in courses on advanced control of vehicle

---

[3] Hal Kassof, "Deployment of ITS – Cornerstones of a Strategy," ITS Quarterly, vol. III, Fall 1995.
[4] J.Y. Wong, *Theory of Ground Vehicles, Second Edition*. New York: John Wiley & Sons, Inc., 1993, p. 239.
[5] Kassof.

dynamics. This test bed and its successors will become the wind tunnels of the Systems Engineering Department, allowing the Systems Department to gain the same real world data regarding automotive dynamics that the Aerospace Engineering Department enjoys regarding flight dynamics.

3.1 Previous Research/Additions to Previous Research

Sean Brennan and the Alleyne Research Group at the University of Illinois at Champagne-Urbana constructed and tested a sophisticated scale-model vehicle platform called the Illinois Roadway Simulator (IRS) from 1997 to 2002. The success they experienced with vehicle designs prompted the construction of the Automotive Scale-Model Vehicle (ASMV) platform at the United States Naval Academy.

The goals governing the design of the ASMV were to build a smaller, less costly scale-model vehicle platform with fundamental design differences from the IRS. IRS was assembled for approximately $30,000 on a horse exercise treadmill four feet in width and eight feet in length. The ASMV platform was constructed at a cost of $1200 on a human exercise treadmill two feet in width and four feet in length. The Alleyne Research Group used a complex and cumbersome mechanical arm connected to the car to determine vehicle position and orientation. The ASMV uses visual data non-intrusively gathered using an external camera. The analysis of the resulting visual data is less complex and the vision system is much less susceptible to mechanical or electrical failure. The IRS uses a large AC motor to turn the treadmill. The AC motor allows the treadmill to turn in only one direction and cannot rapidly bring the treadmill to a stop to simulate braking. Braking tests using the IRS are limited to the rate with which friction slows the treadmill once the AC motor is shut down. The ASMV treadmill uses a DC motor,

which enables testing of braking or even backing the vehicle. The Alleyne Research Group used a closed-loop treadmill motor controller. The vehicle control system used the measured closed-loop treadmill velocity to control the vehicle's speed. The ASMV treadmill motor controller functions open-loop. The car reacts physically to changes in treadmill velocity without direct communication between the treadmill and vehicle control loops. The elimination of internal communication between the vehicle and treadmill control loops dissolves their interdependence and simplifies control of the system.

4 Problem Statement

The interaction between the tires and the road generate the forces acting on a typical road vehicle, illustrated in Figure 4.1.

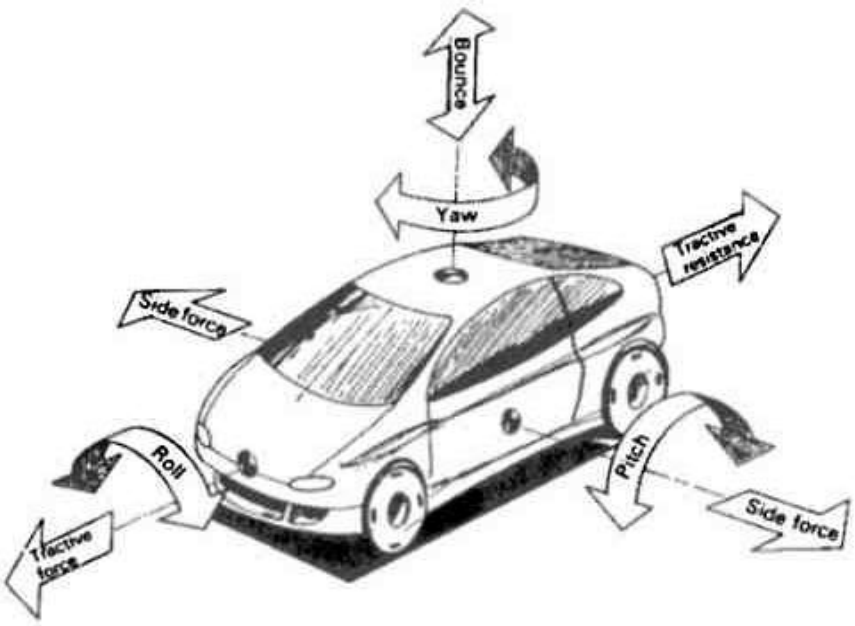


Figure 4.1: Forces on a road vehicle[6]

[6] Brennan, 49.

The friction force developed between the tires and the road is a function of the weight of the car, the condition of the road surface, and the interaction between the tires and the road surface. If the weight above each tire is constant, the friction force between the road and the tires will remain constant. For normal driving operation and road surface conditions, there is sufficient friction force available to accommodate reasonable steering maneuvers. However, during the emergency turning/braking (ETB) maneuver, defined by full application of the brakes combined with an aggressive steering maneuver, a portion of the friction force normally available for turning is used to dissipate the kinetic energy associated with the forward motion of the vehicle. As a result, the vehicle turning response to driver input may be inadequate. Specifically, the vehicle response may be too slow to avoid impact with an obstacle as a result of wheel slippage or the vehicle handling characteristics may degrade to the point that the driver loses control.[7] The handling of the car changes during a braking maneuver as a result of the weight distribution's shift toward the front wheels as the vehicle pitches forward. Additionally, turning the vehicle at high speed induces a moment about its longitudinal axis that transfers the weight distribution toward the side of the vehicle opposite the turn direction. The change in weight distribution makes the vehicle less stable and more difficult to control. Wheel slippage caused by the lack of available friction for turning, further complicates handling. The amalgamation of these factors produce a vehicle response during the ETB maneuver that is unpredictable and dangerous.

Isolated from the design of any single vehicle control subsystem, this project provides refinement to the vehicle design process. The tremendous cost of full-size vehicle testing

[7] Datta N. Godbole, et. al., "Design of Emergency Manuevers for Automated Highway System: Obstacle Avoidance Problem," Proceedings of the 36th Conference on Decision & Control, San Diego, CA, December 1997, pg. 4774-4779.

components provide a motivation to utilize scale-model vehicles as a design step to further refine vehicle concepts before a full-size version is constructed. Improving vehicle design on the scale-model level allows more meaningful full-size testing, where equipment and time is far more expensive. The high cost of full-size testing make it difficult for small research facilities to contribute to the industrial body of knowledge. As a result, the majority of the research is performed by a few facilities that have the resources to perform full-scale tests. Scale-model vehicles bridge the gap between computer simulation and full-size vehicle testing, while enabling more facilities to contribute to the development of safer and more intelligent road vehicles.[8]

5 Hardware Development

The ASMV platform consists of a treadmill, scale-model vehicle, computer control system, power source, and a vision system. The control system has three loops: the driver control loop, treadmill control loop, and the ETB controller loop. The ETB controller loop could be replaced with any driver assistance control system that warrants testing. The treadmill loop controls the treadmill belt velocity, while both the driver control loop and the ETB controller loop are part of the vehicle control system. The velocity of the treadmill is controlled independently from the vehicle control system. The driver loop simulates the driver under normal conditions by correcting for changes in treadmill velocity, vehicle position, and vehicle orientation. The driver loop utilizes image data from the vision system to determine vehicle position and orientation. The computer converts position and orientation deviations from the desired positions into steering and throttle inputs sent to the vehicle via a wire tether. The ETB

[8] Brennan, 12.

controller utilizes the same control path as the driver loop but includes a rate limiter on the vehicle's yaw rate, to prevent the driver loop from turning the vehicle faster than the vehicle's dynamics can tolerate. The entire system is shown in Figure 5.1.

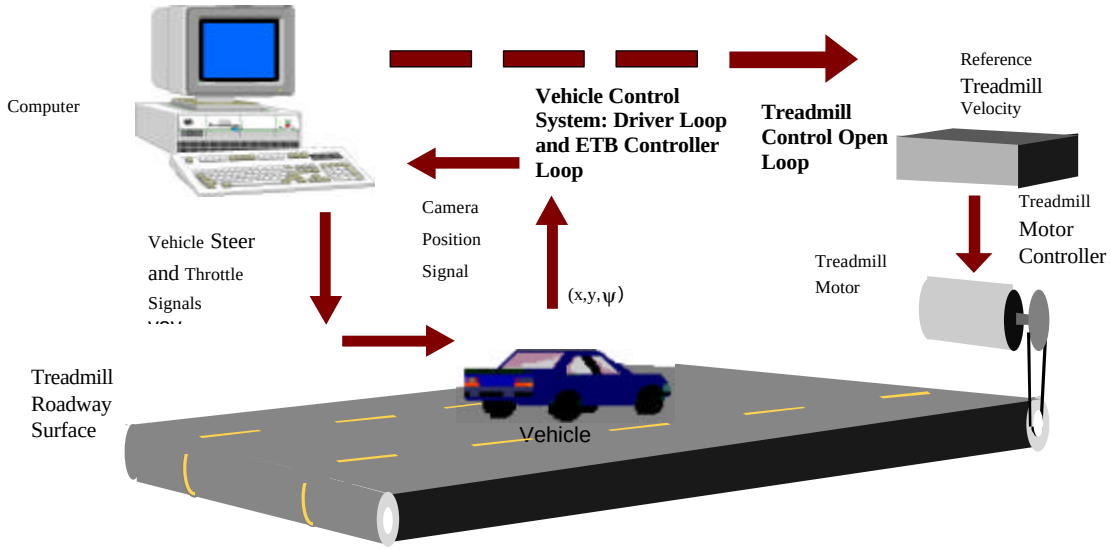


Figure 5.1: ASMV platform with all three control loops illustrated[9]

5. 1 Treadmill Control

A commercial human exercise treadmill was modified to serve as a simulated road surface. Treadmill speed control is necessary to produce different operating conditions for the vehicle. Computer control of the treadmill is preferable to ad hoc manipulation of the treadmill power supply knob to ensure repeatability. The treadmill speed controller utilizes pulse-width modulation (PWM) to achieve repeatable results and high treadmill velocity resolution. A remote PIC microprocessor maintains the PWM signal at 300 Hertz. The duty cycle of the pulse-width is based on an eight-bit value sent to the PIC from the computer via a RS-232 serial connection when a change in the treadmill speed is required. When no communication is present

[9] Brennan, 16.

between the computer and PIC, the PIC maintains the PWM signal at the current duty cycle. The PIC allows power from the four-Amp/ 35-Volt power supply to flow to the treadmill motor in accordance with the rise and fall of the PWM signal. With the assistance of a remote PIC, which requires only event transition information, the computer is able to remotely control treadmill speed while directly controlling the vehicle's position and velocity. Without the PIC, simultaneous control of both systems would prove complicated and might be beyond the capacity of the computer. Treadmill control functions in an open loop independent of the vehicle control system. Control loop independence allowed a fully functional, less complicated test bed design.

5.2 Scale Model Vehicle

The scale model vehicle, manufactured as a kit by HPI Racing, Inc., was constructed utilizing a few included hand tools and small modeling knife. In addition to the contents of the kit, an RC motor, battery pack, speed controller, and a method of vehicle control (normally a RC controller, with a human operator) is required to operate the vehicle. The vehicle's features include four-wheel drive, four-wheel independent suspension, front and rear differentials, realistic tires, and a rugged, rigid structure that resembles a full size vehicle. The vehicle's speed controller has configurable braking and throttling characteristics allowing great flexibility in experimental testing possibilities. The vehicle's steering mechanism is a Futaba standard servomotor. Furthermore, a marker system was installed to enhance vehicle detection by the vision system. The marker system is composed of four red Light Emitting Diodes (LEDs) positioned near the four corners of the vehicle. Power for the vehicle's motor and lighting system is provided by a RC battery pack mounted amidships on the vehicle. The vehicle is shown in Figure 5.2.

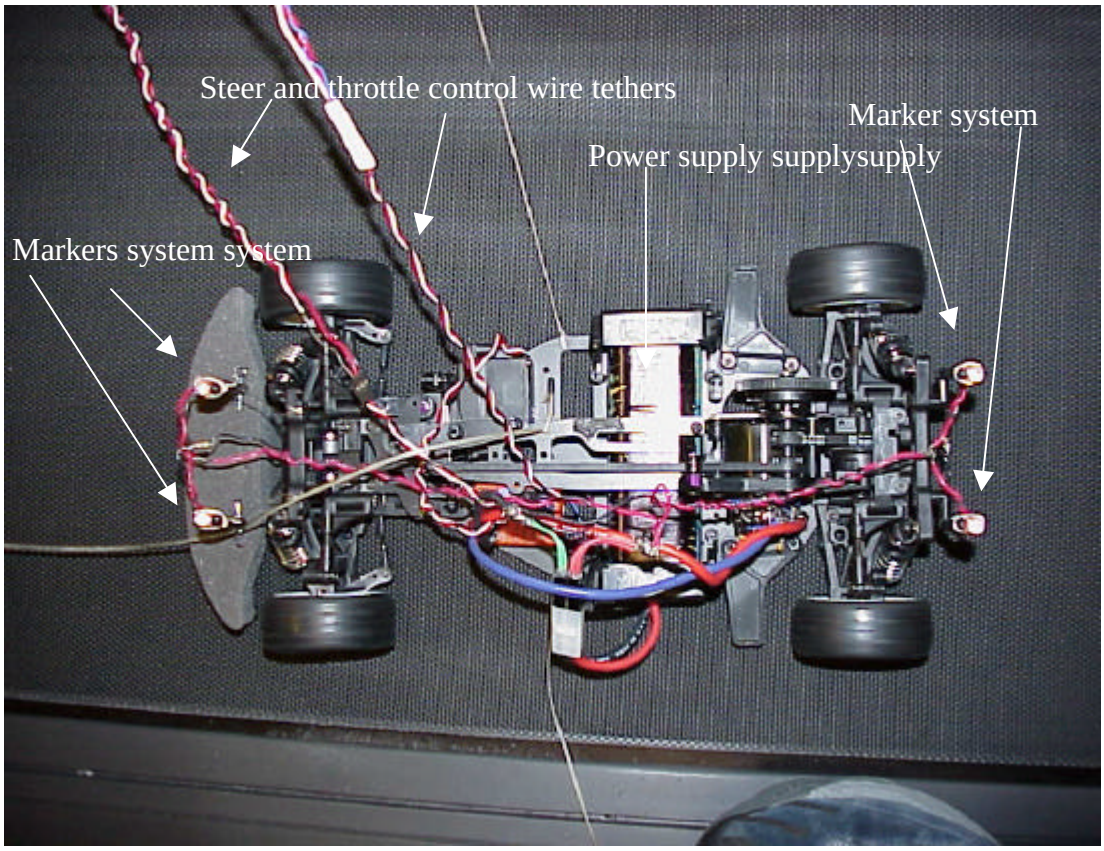


Figure 5.2: The scale-model vehicle: originally designed for remote control, modifications to the scale-model vehicle included the addition of steering and throttle control wires, and an LED marker system that allowed for easy direction by the vision system.

5.3 Sensor Suite

Four ADXL202E accelerometers fitted to ADXL202EB evaluation boards produced by Analog Devices gather data in three dimensions through the proper addition and subtraction of their outputs. The ADXL202E is a tiny (3mm × 3mm × 2mm) lightweight, durable (shock proof to 1000 g's), high resolution from 0 to ±2 g's accelerometer. Its lightweight, small size, and durability allow it to be placed unobtrusively on the model car, while its durability assured its proper function despite the shocks and jolts it received during experimental testing. The sensors will be used to directly measure the yaw rate for the current ETB controller and the longitudinal acceleration in future design iterations of the ETB controller. The sensor suite can

also be used in the future to measure any velocities, accelerations, or angular positions other than yaw that may be needed for the design of future systems on the ASMV platform.

5.4 Vision System

The vision system gathers, analyzes, and sends data to a compiled C++ program within the computer that regulates the vehicle's position and orientation in the center of the treadmill. A camera boom was constructed by the Technical Service Division shop to hold the camera, with a variable height feature (four to six feet above treadmill belt) to image the entire region of interest. The images rapidly gathered by the camera and frame-grabbing software are analyzed using the previously mentioned program to find the vehicle's position and orientation. Specifically, the images are analyzed to determine the location of the four LED markers. The program counts the number of pixels illuminated by each of the LEDs and calculates a weighted average of their positions. Their average value is the centroid, given in terms of the number of pixels below and to the right of the vehicle. The second moment about the vehicle's principal (longitudinal) axis gives the vehicle's orientation. Using the position and orientation data, the computer controller converts the information into the appropriate throttle and steering commands sent to the servo controller board, the Pontech SV203. Commands are of the format "SV**x** I**yyy** \n" where **x** represents the designated servo and **yyy** represents the incremental distance (0-255) the servo is directed to move. A detailed explanation of the computer controller is included in Section 6. The '\n' marks the termination of the communication packet between the computer and SV203. The computer communicates directly with the SV203 servomotor controller board through a RS-232 protocol serial connection. The SV203 delivers specific servo control signals to the steering servomotor and speed controller. These signals are transmitted to the vehicle via

a wire bundle tether. The information from the vision system keeps the car centered on the treadmill and simulates the human driver during normal vehicle operation.

6 Modeling

A three degree-of-freedom (DOF), linear time-invariant (LTI) model of a highway vehicle was developed with constant longitudinal (x) velocity and consideration only for the forces generated by the interaction of the tires and the road. Figure 6.1 shows the vehicle axis orientation. The complete derivation of the model and conversion to state space form is included in the following sections.

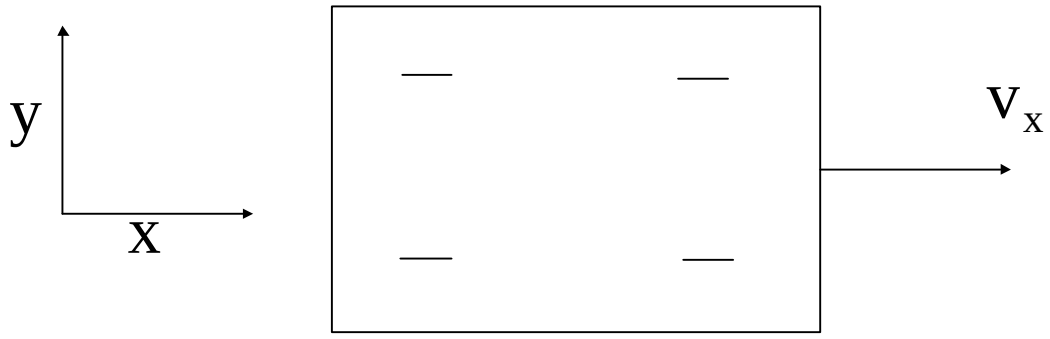


Figure 6.1: Vehicle axis orientation

6.1 Longitudinal Dynamics

The longitudinal dynamics describe the motion of the vehicle when the steering input is zero. The longitudinal equation of motion for the whole vehicle is described by

$$\sum F_x = 2\left(f_t - f_f\right) - 2f_f - C_d v_x^{\,2} = ma_x = m\dot{v}_x \qquad (1)$$

where

$v_x \equiv$ velocity in the x-direction

$f_f \equiv$ force of friction on the tire

$f_t \equiv$ tractive force

The forces acting on the car are a result of the interaction between the tire and road surface. In particular, the tractive force is dependent on the relationship between the longitudinal velocity and the tire angular velocity. Figure 6.2 is the free-body diagram of the tire. Summing the moments about the tire results in the following

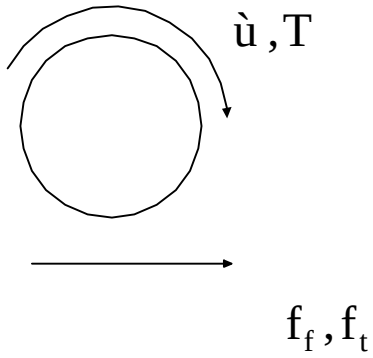


Figure 6.2: Torque acting about the axis of rotation and forces on tire

$$\sum M_y = J\dot{w} = T - r(f_f + f_t) \qquad (2)$$

$r \equiv$ tire radius

$J \equiv$ tire inertia

T torque (from engine through the transmission)

$C_d \equiv$ coefficient of drag

If the engine is assumed to supply only enough torque to the wheels via the transmission to overcome friction forces then

$$\sum F_x = 0 \rightarrow f_t = 2f_f + \frac{1}{2}C_d v^2 \qquad (3)$$

$$w = \frac{v}{r}$$

At constant velocity, this result implies

$$\dot{w} = 0 \qquad (4)$$
$$T = 2\left(f_t + f_f\right)$$

6.2 Lateral Dynamics

Lateral dynamics describe the motion of the vehicle perpendicular to the direction of travel. The lateral motion is specified by the lateral position and yaw angle of the vehicle. Figure 6.3 shows the parameters relevant to lateral motion.

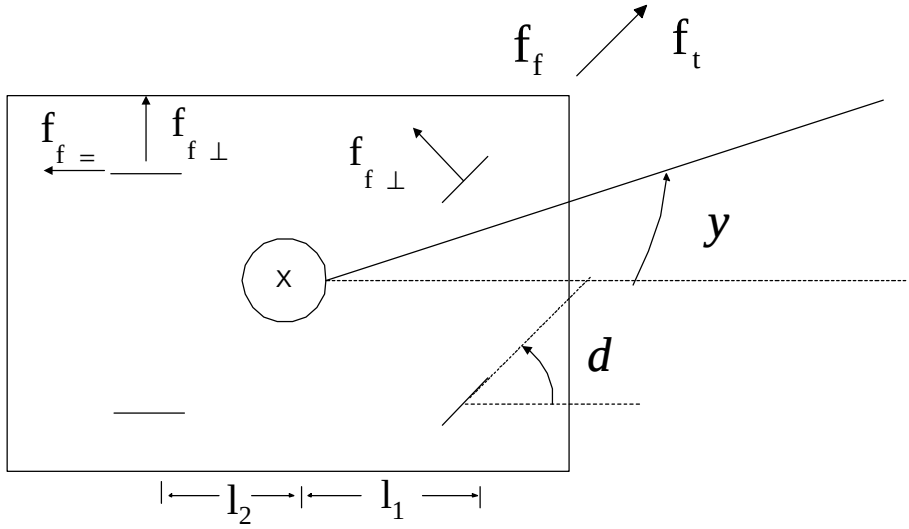


Figure 6.3: Free body diagram of interaction of vehicle tires and road surface

where

$f_{f\perp}$ ≡ friction force perpendicular to motion of tire
d ≡ steering angle
Ψ ≡ yaw angle
$f_{f=}$ ≡ friction force parallel to motion of tire

Assuming the engine/transmission produce only enough torque to maintain constant velocity ($\sum F_x = 0$), the lateral position is described by

$$\sum F_y = 2 f_{f\perp} \cos(d) + 2(f_t - f_f)\sin(d) + 2 f_{f\perp} \qquad (5)$$

Summing the moments about the vehicle's center of gravity results in the moment

$$J_z \ddot{\Psi} = \sum M_z = (2 f_{f\perp} \cos(d) + 2(f_t - f_f)\sin(d))l_1 - 2 f_{f\perp} l_2 \qquad (6)$$

where

$l_1 \equiv$ distance from center of front tire to the center of gravity of car

$l_2 \equiv$ distance from center of rear tire to the center of gravity of car

6.3 Tire Model

The forces acting on the car are a result of the interaction between the tires and road surface. For the case examined, the lateral force, the factors involved are the normal load, tire type and condition, road condition, and velocity.

6.3.1 Slip Angle

Slip angle (a) is angular distance between the direction of motion of the vehicle and the direction of the principal longitudinal axis of the tire. It is caused by brief initial tire slippage as the wheel is turned. The physical meaning of the slip angle is illustrated in Figure 6.4 and 6.5.

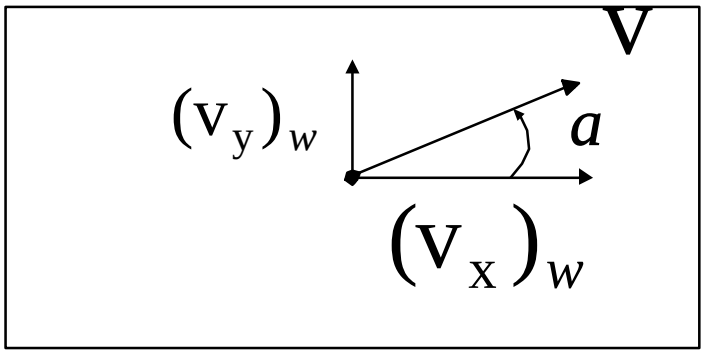


Figure 6.4: Wheel and vehicle velocities as they are affected by tire slippage

where

$(v_x)_w, (v_y)_w \equiv$ x, y velocities of the wheel

Under constant velocity conditions, where no pitch or roll motion occurs, $(v_x)_w \approx v_x$. As shown in Figure 6.6, the force is given by the expression $f_{f\perp} \equiv f_T \sin a$ where

$a \equiv$ slip angle

$f_T \equiv$ total friction force, opposes the direction of vehicle motion

$f_{f=} \equiv$ friction force parallel to tire longitudinal axis

$f_{f\perp} \equiv$ friction force perpendicular to tire longitudinal axis

$v \equiv$ velocity of vehicle

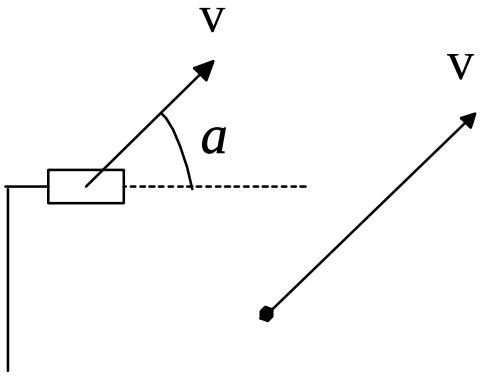


Figure 6.5: Illustration of slip angle on rear tire

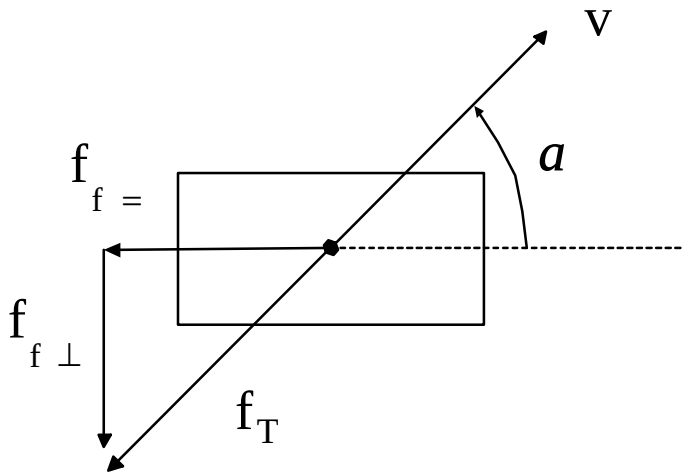


Figure 6.6: Illustration of forces involved in slip angle determination

Linearizing the relationship, using the small angle assumption, between the perpendicular side force and the slip angle, the following expression is obtained

$$f_{f\perp} = -C_s a \tag{7}$$

where the cornering stiffness is defined as

$$C_s = \frac{\partial f_{f\perp}}{\partial a}\Big|_{a=0} \tag{8}$$

The velocity of the rear tire is described by

$$(v_y)_w = v_y - l_2\dot{\Psi} \tag{9}$$

The slip angle of the rear tire (a_r) is

$$\tan a_r = \frac{(v_y)_w}{(v_x)_w} = \frac{v_y - l_2\dot{\Psi}}{v_x} \tag{10}$$

Using the small angle approximation, the expression becomes

$$a_r = \tan^{-1}\frac{v_y - l_2\dot{\Psi}}{v_x} \approx \frac{v_y - l_2\dot{\Psi}}{v_x} \tag{11}$$

Figure 6.7 illustrates the relationship between the steering angle and the slip angle. Using Figure 6.4 and 6.7, an expression for the slip angle of the front tire (a_f) is obtained

$$\tan(a+d) = \frac{(v_y)_w}{(v_x)_w} = \frac{v_y + l_1\dot{\Psi}}{v_x} \tag{12}$$

$$a_f = -d + \tan^{-1}\frac{v_y + l_1\dot{\Psi}}{v_x} \approx -d + \frac{v_y + l_1\dot{\Psi}}{v_x} \tag{13}$$

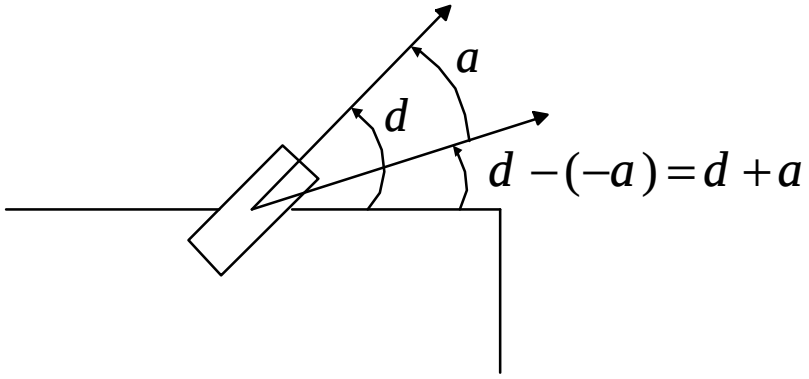


Figure 6.7: Relation between steering angle and slip angle

6.3.2 State Space model of lateral dynamics

State space is a format used for the representation of a system used for control design. The state variables used in this model are

$y_i \equiv$ lateral position in the inertial frame

$v_y \equiv$ lateral velocity

$\Psi \equiv$ yaw angle

$\dot{\Psi} \equiv$ yaw angular velocity

The *i* and *b* subscripts differentiate the inertial and body frame position and velocity. The acceleration in three directions

$$\frac{1}{m}\begin{bmatrix}F_x\\F_y\\F_z\end{bmatrix}=\dot{\bar{v}}=\begin{bmatrix}\dot{v}_{xb}\\\dot{v}_{yb}\\\dot{v}_{zb}\end{bmatrix}+\begin{bmatrix}\dot{\Psi}_x\\\dot{\Psi}_y\\\dot{\Psi}_z\end{bmatrix}\times\begin{bmatrix}v_{xb}\\v_{yb}\\v_{zb}\end{bmatrix}=\begin{bmatrix}\dot{v}_{xb}+v_{zb}\dot{\Psi}_y-v_{yb}\dot{\Psi}_z\\\dot{v}_{yb}+v_{xb}\dot{\Psi}_z-v_{zb}\dot{\Psi}_x\\\dot{v}_{zb}+v_{yb}\dot{\Psi}_x-v_{zb}\dot{\Psi}_y\end{bmatrix} \quad (14)$$

where $v_{zb}=0$, $\dot{\Psi}_x=0$, and $\dot{\Psi}_y=0$. At constant velocity and normal road conditions there will be no velocity in the z direction and no pitch or roll motion. Substituting equation (7) into equation (5) for front and rear tires assuming small angles reveals

$$\begin{aligned}\sum F_y &= 2f_{f\perp}\cos d-2f_r\sin d+2f_{f\perp}\approx 2f_{f\perp}-2f_r d+2f_{f\perp}\\ &=-2C_s a_f-2f_r d-2C_s a_r\end{aligned}$$

Similarly for the moments

$$\begin{aligned}&\sum M_z=(2f_{f\perp}\cos d-2f_r\sin d)l_1+2f_{f\perp}l_2\approx(2f_{f\perp}-2f_r d)l_1-2f_{f\perp}l_2\\&=(-2C_s a_f-2f_r d)l_1+2C_s a_r l_2\end{aligned}$$

Assembling the forces and moments,

$$\frac{d}{dt}\begin{bmatrix}y_i\\v_y\\\Psi\\\dot{\Psi}\end{bmatrix}=\begin{bmatrix}0&1&v_x&0\\0&\frac{-4C_s}{mv_x}&0&\frac{-2C_s(l_1^2-l_2^2)}{J_z v_x}\\0&0&0&1\\0&\frac{-2C_s(l_1-l_{2)}}{J_z}&0&\frac{-2C_s(l_1^2-l_2^2)}{J_z v_x}\end{bmatrix}\begin{bmatrix}y_i\\v_y\\\Psi\\\dot{\Psi}\end{bmatrix}+\begin{bmatrix}0\\\frac{2(C_s-f_r)}{m}\\0\\\frac{2(C_s-f_r)l_1}{J_z}\end{bmatrix} \quad (15)$$

7 Vehicle Control Loops

The vehicle control loops maintain the scale-model vehicle's position in the center of the treadmill during normal vehicle operation. They simulate the driver maintaining the vehicle's position at a constant velocity in a straight line. The controllers will also respond and correct the vehicle's position for minor disturbances.

7.1 Longitudinal Controller

The longitudinal controller keeps the vehicle centered longitudinally on the treadmill. In conventional road vehicles, the driver uses the throttle input (gas pedal) to maintain a desired longitudinal velocity. On the treadmill, the vehicle is not moving in relation to the Earth, as the full-size vehicle would be, but is only moving in relation to the treadmill belt. The problem of maintaining longitudinal velocity becomes one of maintaining the scale-model vehicle's position in reference to the Earth. By maintaining the vehicle's longitudinal position relative to the Earth, the controller will force the vehicle to match the treadmill's velocity.

7.1.1 Control Strategy

For longitudinal control, a proportional feedback controller based on vehicle position was sufficient. The block diagram of the controller is shown in Figure 7.1. The gain of the proportional controller, K_{long}, is 0.01. The description of the process used to find the gain is included in Section 7.1.2.

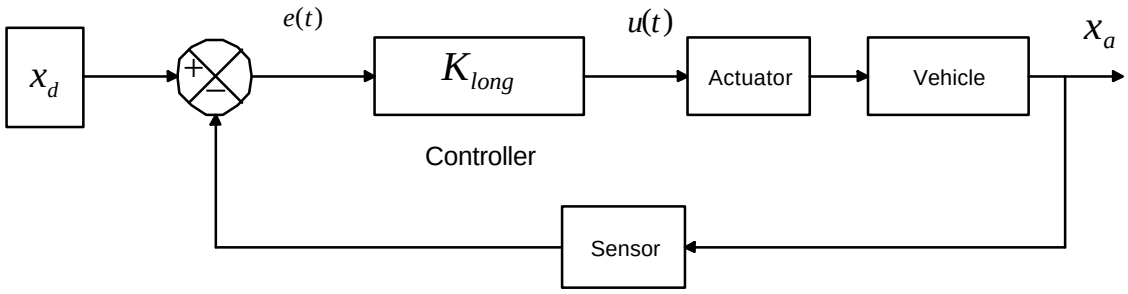


Figure 7.1: Proportional compensator for control of longitudinal position. x_d represents the desired longitudinal position and x_a represents the actual longitudinal position. Both x_d and x_a are measured by the vision system.

7.1.2 Implementation

Implementation of the longitudinal controller begins with the acquisition of the vision data from the vision system. The vision system will return both the longitudinal and lateral coordinates of the vehicle's centroid. The longitudinal controller calculates the difference between the actual pixel location of the longitudinal coordinate of the centroid and the desired pixel value. The error between the two, the position error, is then multiplied by K_{long} and the result is transmitted to the SV203 servo-controller board via a serial connection in the format described in Section 5.4. The gain K_{long} scales the position error into a discrete incremental increase or decrease in speed. There is no limitation upon the output of the controller, but the SV203 will accept only 256 discrete motor speeds from the longitudinal controller. Typical increases or decreases in velocity seldom exceed ten steps. Furthermore, the motor will only overcome internal friction at step 120. The result is a longitudinal controller capable of driving the vehicle at 135 discrete velocities. This proved more than sufficient for maintaining vehicle position on the treadmill at a variety of different treadmill speeds.

The selection of K_{long} was conducted in a trial and error manner without prior simulation. For gains higher than 0.01, the response had excessive oscillations and tended toward instability. For gains lower than 0.01, the response was sluggish. To compromise between a very slow response and a fast response with excessive oscillations, K_{long} is set to 0.01. The treadmill is small and there is limited space for excessive vehicle oscillations, and so a lower gain with a slower response was chosen over higher gains with greater oscillations but a faster response. Lower gains, e.g. $K_{long} = 0.001$, also resulted in a zero control signal dispatched to the vehicle's motor controller for all but very large errors because the discrete nature of the longitudinal controller output resulted in the small error being rounded to zero.

7.2 Lateral Controller

The vehicle's lateral controller must govern both yaw angle (vehicle orientation) and lateral vehicle position. In conventional road vehicles, the driver corrects for errors in lateral position and yaw angle through the steering input. In a full-size vehicle, the driver estimates the necessary steering wheel angle required to simultaneously correct both lateral position and yaw angle based on the driver's past experience in similar situations. In an autonomous vehicle system, such as the ASMV testbed, a controller must computationally combine the errors in yaw angle and lateral position to formulate and generate a single steering control signal transmitted to the vehicle plant.

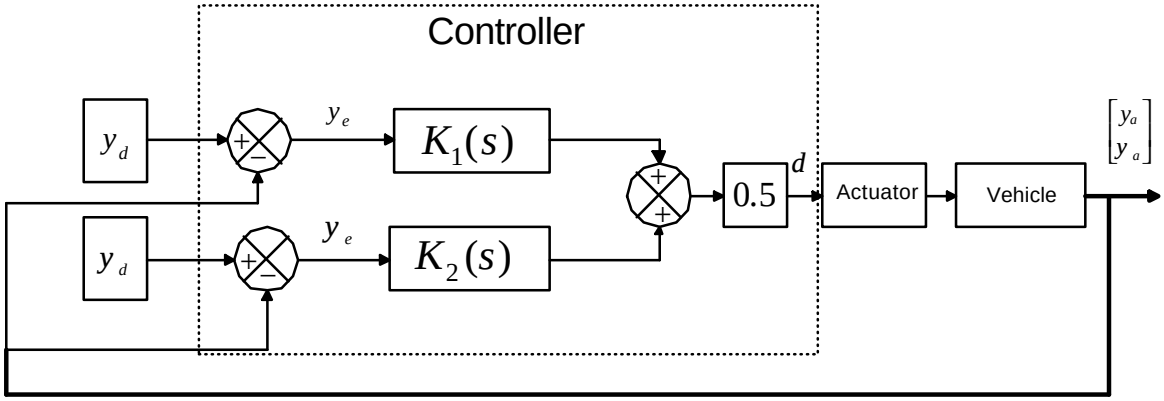


Figure 7.2: Lateral controller design concept with two inputs, y_d and $\dot{y}_d$, and one control output, d

7.2.1 Control Strategy

The challenge in the lateral controller design is to control two vehicle outputs using a single input. The closed-loop system is shown in Figure 7.2.

The following method was employed to combine the necessary yaw change and lateral position change into a single output. Consider any complex number, z, that is the sum of two complex numbers, z_1 and z_2. Written in polar form, z_1 and z_2 become $r_1 e^{jf_1}$ and $r_2 e^{jf_2}$. In general, combining these two numbers would yield the expression.

$$z = \sqrt{r_1 + r_2 + 2r_1 r_2 \cos(f_1 - f_2)}\, e^{j \tan^{-1}\left(\frac{r_1 \sin f_1 + r_2 \sin f_2}{r_1 \cos f_1 + r_2 \cos f_2}\right)}$$

The expression is cumbersome and does not yield an intuitive association between z_1, z_2 and the resultant z. Now, consider the specific case where $f = f_1 = f_2$. The expression for z then becomes $(r_1 + r_2)e^{jf}$. The resulting expression is far simpler and yields a clear relationship between r_1, r_2, f, and the phase and magnitude of z. The simple relationship between the components of the control allows for uncomplicated computation of the control signal.

Using the above result, the controllers $K_1(s)$ and $K_2(s)$ are designed separately, utilizing the concepts of Bode design, to achieve the same gain crossover frequency, w_{gc}, and phase margin, PM. The gain crossover frequency is the frequency where the magnitude of the response is one. On a Bode diagram it is the frequency where the magnitude plot crosses zero. The phase margin is the phase difference between the phase plot at w_{gc}, and –180 degrees. Both concepts are illustrated in Figure 7.3. In this case, the frequency response of the term

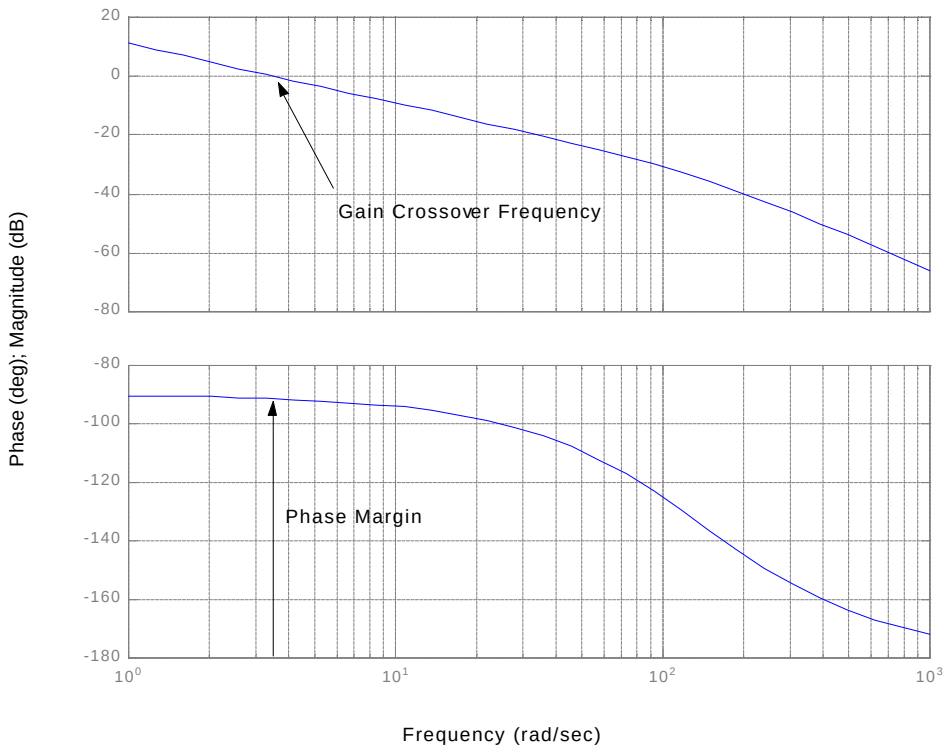


Figure 7.3: Example of Bode Plot with gain crossover frequency and phase margin shown

$G_1(s)K_1(s)+G_2(s)K_2(s)$ at w_{gc} is

$$G_1(jw_{gc})K_1(jw_{gc})+G_2(jw_{gc})K_2(jw_{gc})$$

$$= 1e^{j(-180^\circ + PM)} + 1e^{j(-180^\circ + PM)}$$

$$= 2e^{j(-180^\circ + PM)}$$

Therefore, the w_{gc} and PM for the closed-loop system will be the same as for the individual loops because

$$G_1(jw_{gc})\left(\frac{1}{2}K_1(jw_{gc})\right)K_1(jw_{gc}) + G_2(jw_{gc})\left(\frac{1}{2}K_2(jw_{gc})\right)$$

$$= 1e^{j(-180^\circ + PM)}$$

A design with $K_1(s)$ and $K_2(s)$ as the lateral position and yaw angle controllers, respectively, the gain crossover frequency, w_{gc}, and phase margin, PM, for each compensated system, allows the control signal to be composed of two equally weighted components in the form,

$$\frac{1}{2}\left(\left|G_1(jw_{gc})K_1(jw_{gc})\right| + \left|G_2(jw_{gc})K_2(jw_{gc})\right|\right)e^{j(-180^\circ + PM)}$$

The closed-loop transfer function for the system is

$$\begin{bmatrix} Y_a(s) \\ y_a(s) \end{bmatrix} = \begin{bmatrix} \dfrac{\frac{1}{2}\left(G_1(s)K_1(s) + G_2(s)K_2(s)\right)}{1 + \frac{1}{2}\left(G_1(s)K_1(s) + G_2(s)K_2(s)\right)} \\ \dfrac{\frac{1}{2}\left(G_1(s)K_1(s) + G_2(s)K_2(s)\right)}{1 + \frac{1}{2}\left(G_1(s)K_1(s) + G_2(s)K_2(s)\right)} \end{bmatrix}$$

7.2.2 Implementation

By stepwise testing compensators in order of increasing complexity, a lead compensator was chosen to control the lateral position of the vehicle and a gain compensator was chosen to regulate the yaw angle. The lateral position portion of the lateral controller consists of the lead compensator $K_1(s) = 11.42\dfrac{s+14.99}{s+11.75}$, and the proportional controller $K_2(s) = 0.91$.

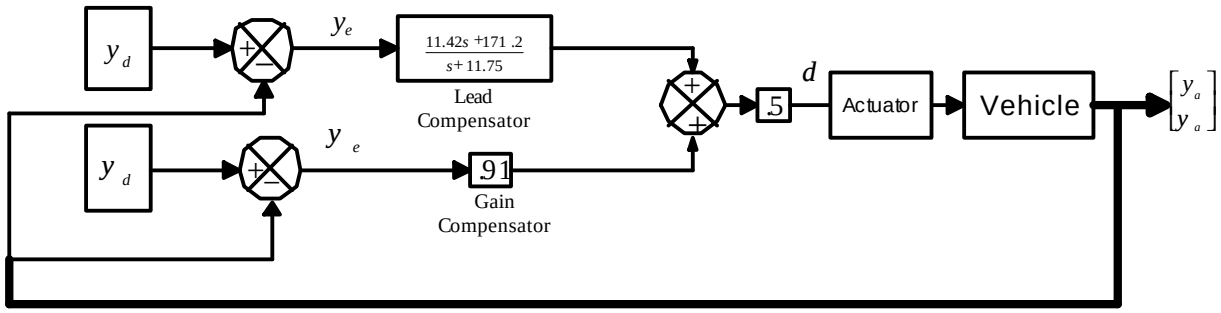


Figure 7.4: Lateral controller with lead and gain compensation components

A block diagram of the resulting lateral controller is shown in Figure 7.4. The full simulation of both controllers is included in the Appendix as Encl(6). Displayed in Figure 7.5 is the results of the simulation for an initial lateral position of 0.05 meters and yaw angle of 5.72 degrees.

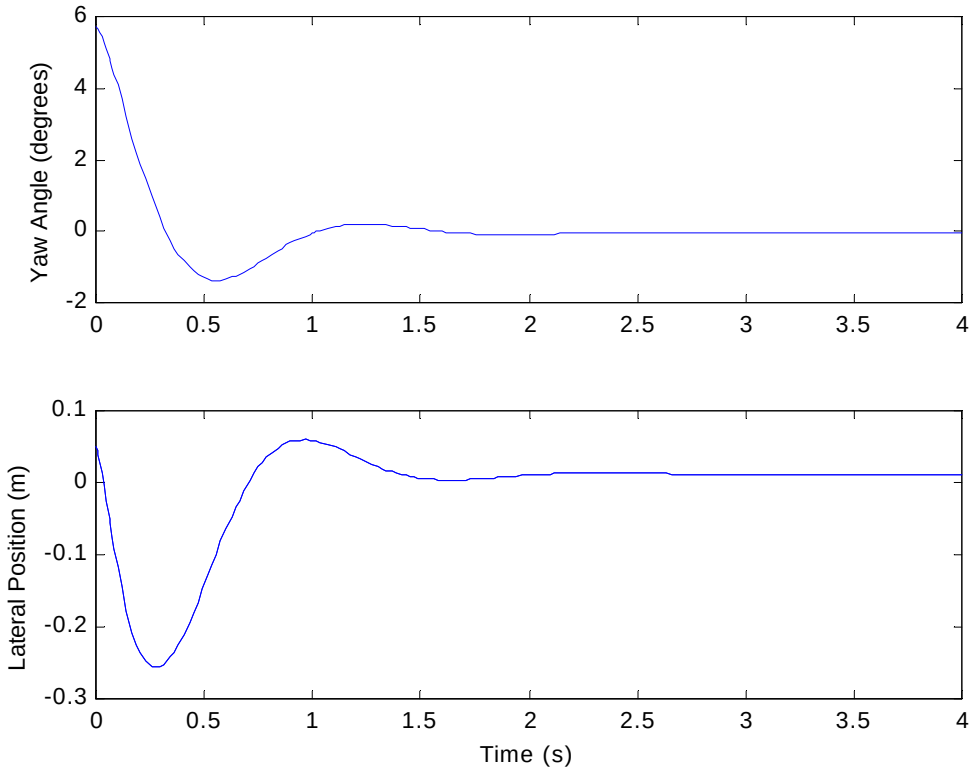


Figure 7.5: The simulation results for lateral controller when initial lateral position is .05 m and initial yaw angle is 5.72 degrees

Once the output signal is determined, it is converted to a number between 0 and 255, using a scaling factor and sent to the SV203 servo controller board. The structure of the vehicle allows the wheels to swing through an angle of only ±30 degrees. The limit on vehicle turning angle eliminates the first and last 15 steps of the servo's turning gradient, allowing 226 discrete angular positions the front wheels could be turned to. The signal is filtered to prevent the wheels from being overturned. The signal is also filtered to assure the wheels are not turned too large an angle resulting in the scale-model leaving the treadmill. Once the signal passes through both filters it is outputted to the SV203, which converts the number into a voltage that is sent to the steering servo.

8 Control Methods

The basis for the controller design using optimal control methods is state feedback control. In order to control a system using state feedback, the controller must have access to all the states in a system. The states of the system are the minimum set of parameters that describe all other pertinent parameters in a system. In state feedback control, the states of the system are measured and combined with a reference signal to generate the control input. The goal is to drive the error between the reference signal and the output to zero. The advantage of state feedback is the dynamics of the system can be modified to meet design specifications. State feedback design will result in a system more closely controlled because all states are used during the control process. The state feedback process consists of two main components shown in Figure 8.1. The controller adjusts the system response to the desired system response. An observer is used to generate an estimate for the states when it is not feasible to measure all the states or when the system is noisy.

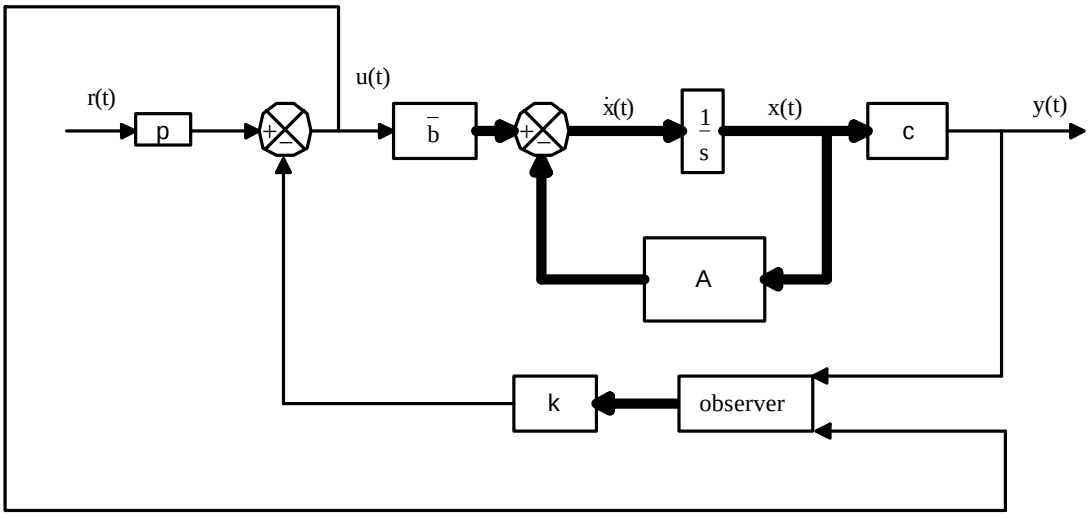


Figure 8.1: State Space Feedback Diagram with Controller and Observer

8.1 The Controller

By converting the realization of the system to controller canonical form, the system can be manipulated mathematically in a simplified manner. A method for conversion to controller canonical form is included in *Control System Design*.[10] When transformed to controller canonical form, the coefficients of the system's characteristic polynomial, whose roots are the system poles, can be easily determined. In the system below d_{11} and d_{12} are the coefficients and together they will form the characteristic equation $s^2 + d_{11}s + d_{12}$.

The generic state space form is given by

$$\dot{x} = Ax + bu$$
$$y = cx$$

where A is the state matrix, b is the input matrix, and c is the output matrix. For two states of a single- input, single-output (SISO) system, the state space model is given by

$$\dot{x}(t) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} x(t) + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} c_1 & c_2 \end{bmatrix}$$

Where $z = T_c^{-1} x$, the same system transformed to controller canonical form[11]

$$\dot{z}(t) = \begin{bmatrix} -d_{11} & -d_{12} \\ 1 & 0 \end{bmatrix} z(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} z(t)$$

[10] Bernard Friedland, *Control System Design: An Introduction to State-Space Methods*. New York: McGraw Hill Book Company, 1986, pg. 192-194.
[11] Friedland, 192-194.

T_c is called the controllability matrix and its function and composition is described in the previous reference.

The state feedback gain $k = [k_1 \quad k_2]$ is chosen such that the characteristic polynomial matches the desired characteristic polynomial. The application of state feedback will result in a closed-loop state matrix in controller canonical form

$$\begin{bmatrix} -d_{11} & -d_{12} \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} [k_1 \quad k_2]$$

The characteristic polynomial of the system is

$$s^2 + (d_{11} + k_1)s + (d_{12} + k_2)$$

Therefore, the gains k_1 and k_2 are chosen to achieve the desired characteristic polynomial. Finally, a state feedback for the system in terms of the original state variables is $k_o = kT_c^{-1}$.

The three DOF vehicle model was simulated using various state feedback gain vectors. The goal of the simulation process was to explore the effects of different gains, not to design a truly useful controller or improve system response at this stage of the design. Eventually design of the ETB controller will include a similar analysis where the goal will be to improve system performance. The simulation is included in the Appendix as Encl(3). For a further explanation of ETB controller testing, see Section 9. A state feedback diagram with the controller operating directly from the states of the system is shown in Figure 8.2.

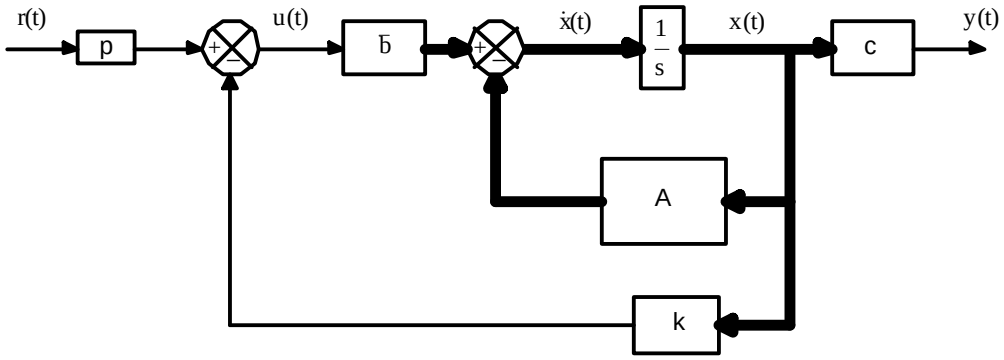


Figure 8.2: State Feedback Diagram with Controller

From Figure 8.2

$$u(t) = -kx(t) + pr(t)$$

$$\dot{x}(t) = Ax(t) + b(-kx(t) + pr(t))$$
$$\dot{x}(t) = (A - bk)x(t) + (bp)r(t)$$
$$y(t) = cx(t)$$

The state equation takes on a new form with the new state matrix in the form $(A - bk)$. For an LTI system such as the one proposed for the three DOF model, A and b will not change. By changing the state feedback gain, the closed-loop poles of the system will change and, in turn, change the system's response to a reference signal.

8.2 The Observer

An observer is used in state-space control system design to estimate the states given the control signal and the system output. The system must also be observable; all states must share interdependence. The observer is shown in a state feedback system in Figure 8.1. Observers are used in applications where it's not feasible to measure all the states in the system or noise obscures the measurement of the states. Design of the observer gain is analogous to the design

of the state feedback gain. Design begins with conversion of the system to observer canonical form. The conversion of a generic state-space model to observer canonical form simplifies the computations required.

For two states of a SISO system, the state space model is given by

$$\dot{x}(t) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} x(t) + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} c_1 & c_2 \end{bmatrix}$$

Where $z = T_o^{-1} x$, the same system transformed to observer canonical form[12]

$$\dot{z}(t) = \begin{bmatrix} -d_{11} & 1 \\ -d_{21} & 0 \end{bmatrix} z(t) + \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

The goal of the observer is to estimate the states quickly and use the state estimates in the state feedback controller. However, if the observer is too fast it will respond to noise in the system output measurement. In this case, the observer response (the estimate of the state) is noisy and does not accurately represent the actual state. Figures 8.3 and 8.4 illustrate this effect. In Figure 8.3, the state estimates oscillate around the actual state values due to the noise in the output measurement. The error in the state estimate will be transferred to the controller and eventually the system. Furthermore, the noisy control signal may cause unnecessary wear on actuators that produce the control signal. Figure 8.4 demonstrates the behavior of the estimation error as time progresses.

[12] Friedland, 210-212.

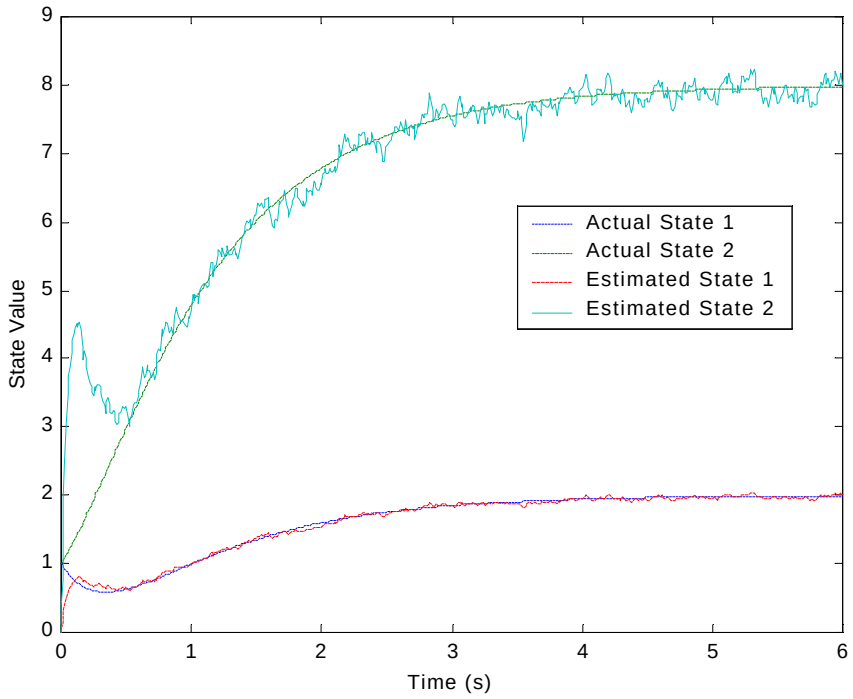


Figure 8.3: Comparison of actual and estimated states observer poles that are too large

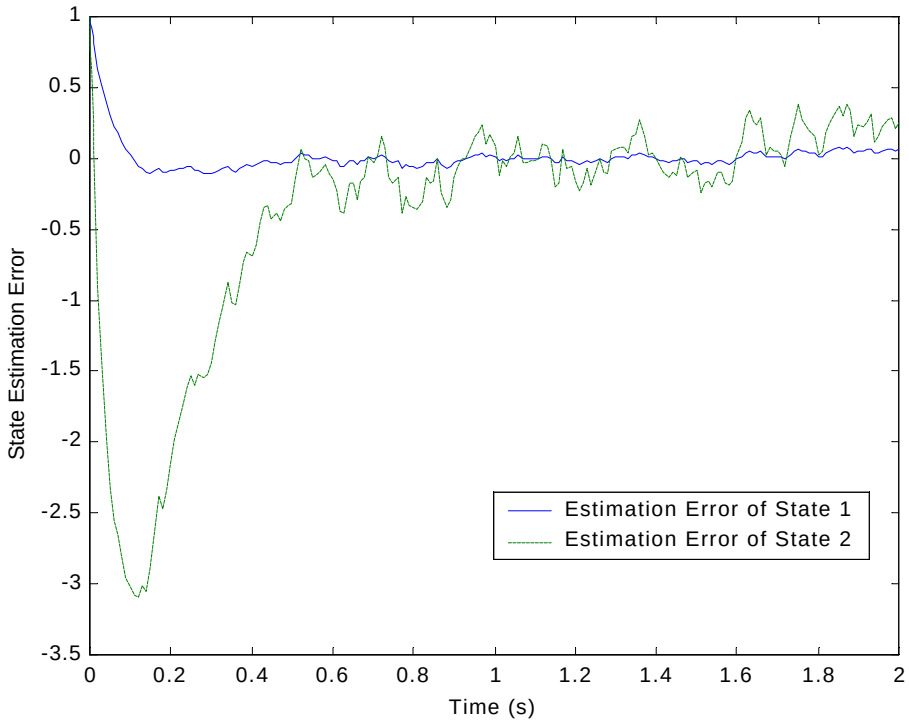


Figure 8.4: State estimation error observer poles that are too large

The observer must be faster than the closed-loop systems' transient response so an accurate state estimate can be used in the state feedback control process. If the observer response is too slow, the observer output will also not accurately represent the state. A poor estimate of the state will force the controller to act on substandard information, producing a poor system response. Figures 8.5 and 8.6 demonstrate the effects of an observer with poles that are too slow. Figure 8.5 shows the difference between the actual and estimated state. The estimate cannot accurately track the actual state, failing to serve as a reasonable estimate even after ten seconds. The estimate is very slow in responding to changes in the actual state. The discrepancy between the estimate and the actual state will generate inaccurate information that will be passed to the controller and will cause poor system response similar to the observer with poles that are too fast. The choosing of

observer poles is not a matter of extremes but of finding a balance between an observer that reacts too slow or too fast, each extreme generating bad estimates that will cause system response performance to suffer. The simulation is included as Encl(2) in the Appendix. The three DOF model with an observer is included as Encl(4).

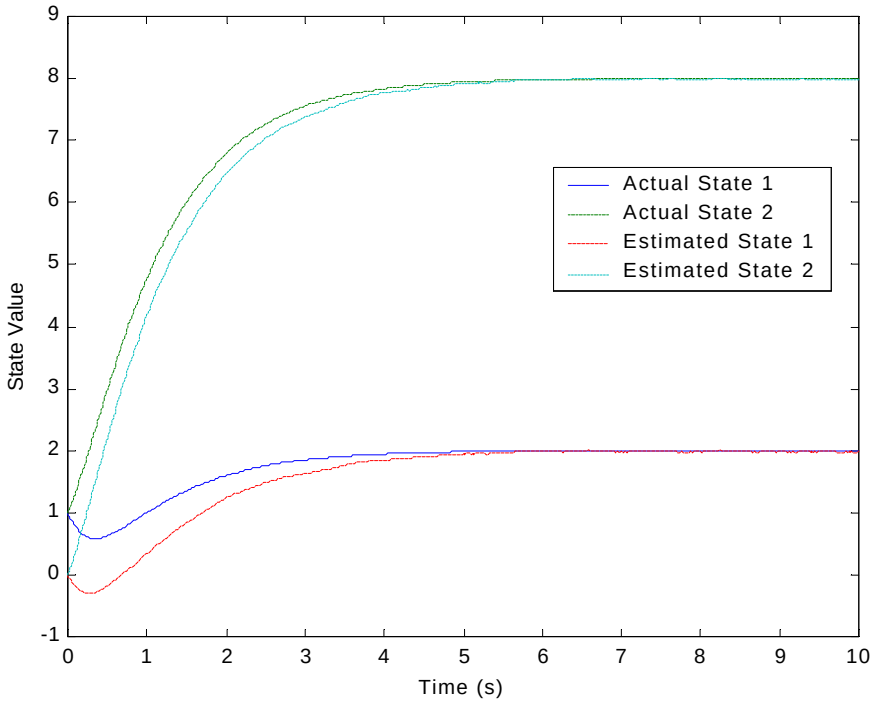


Figure 8.5: Comparison of actual and estimated states observer poles too small

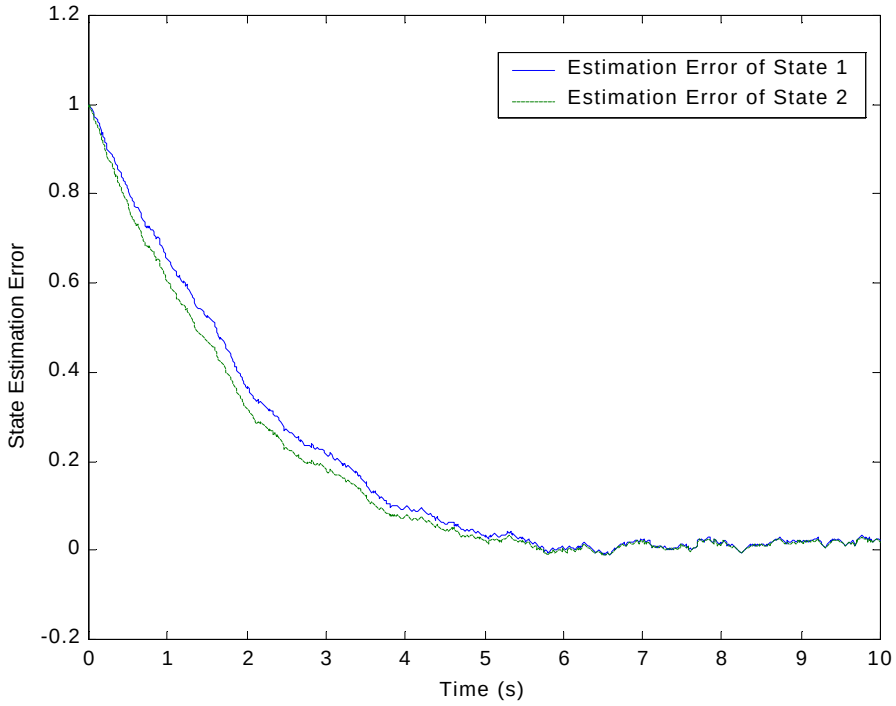


Figure 8.6: State estimation error observer poles too small

8.3 Optimal Control

Ad hoc pole placement design techniques work well for 1st or 2nd degree systems, but become difficult to use on higher order systems like the model developed in this project. In higher order systems the connections between closed-loop poles and zeroes and the closed-loop response is no longer intuitive. Optimal control allows all the poles to be placed by varying external design parameters until the desired response is achieved. Using these design parameters, the iterative design becomes systematic, easing the burden of designing a control system that achieves the design objectives. The quadratic cost function has proven its worth as a tool and is easy to solve, as the integral of a quadratic is a scalar function.[13] It is possible to use

[13] Raymond T. Stefani, et. al., *Design of Feedback Control Systems*. Boston: Saunders College Publishing, 1994.

other cost functions but the solutions can be more cumbersome and will not necessarily yield a better solution.

The optimal control scheme used in this project is know as Linear Quadratic Optimal Control. To explain this scheme, a scalar case is considered where the cost function is $J = \int_0^\infty \left(qx^2(t) + ru^2(t)\right)dt$ and the plant is described by the scalar differential equation:

$$\dot{x}(t) = ax(t) + bu(t)$$

$$y(t) = x(t)$$

$$x(0) = x_0 \quad \text{and} \quad u(t) = -kx$$

The cost function, J, can be interpreted as the total energy of the system contributed by the state, $x(t)$, and control input, $u(t)$. The design parameters (weights) q and r represent the penalties on the state and control. By finding a relationship between q, r, and k (in this case a scalar controller gain), one can find a k gain using a Riccati equation as derived below.

Substituting $u(t) = -kx(t)$ into the cost function yields:

$$J = \int_0^\infty \left((q + rk^2)x^2(t)\right)dt$$

Where $x(0) = x_0$ and $x(t) = x_0 e^{(a-bk)t}$ found by using Laplace transform techniques. Integrating with respect to time:

$$J = x_0{}^2 \frac{q + rk^2}{2(bk - a)} = px_0{}^2 \quad \text{where} \quad f(k) = p = \frac{q + rk^2}{2(bk - a)}$$

Rewriting expression for p yields the equation:

$$2ap + q + rk^2 - 2bkp = 0 \qquad (1)$$

Minimizing the value of p will minimize the value of cost, J, since J is proportional to x_0^2. To minimize p differentiate $f(k)$ with respect to time and set the derivative to zero. The denominator of the resulting expression is always positive and, therefore, the derivative is zero if and only if the numerator equals zero. It follows that k must satisfy

$$rbk^2 - 2ark - bq = 0 \qquad (2)$$

From standard optimal control theory, the value of k that minimizes the cost is $k = r^{-1}bp$ where p satisfies the Riccati equation,[14]

$$2ap + q - b^2 p^2 r^{-1} = 0 \qquad (3)$$

To verify this result, substitute $k = r^{-1}bp$ into (1) and note that the resulting equation is the Riccati equation

$$2ap + q - b^2 p^2 r^{-1} = 0$$

The expression for k comes from a Hamilton-Jacobi minimization of the cost function.[15] In practice the Riccati equation can be solved directly to find the value of p that minimizes the cost. Once p is found, the controller gain k is easily computed using $k = r^{-1}bp$.

[14] Donald E. Kirk, *Optimal Control Theory: An Introduction*. New York: Prentice Hall Inc., 1970.

Using this process to minimize the cost of the system is a valid way to assure proper system response using only the minimum amount of control necessary. Minimum control is economically desirable because the less control used on a system, the lower the power control system components will require during operation. System control that requires only small amounts of power can be accomplished using smaller, less expensive motors actuators, and other components.

In the design process, "penalties" are assigned to the state and control by selecting q and r. If $\frac{q}{r}$ is small, the control is penalized heavily, allowing the state to grow quickly with little control used to limit the state. Figure 8.7 illustrates the response and input to a system with little control. The control input remains small and the states are sluggish in moving to their steady state values. Small control will yield a poor transient system response with a high settling time.

[15] Kirk.

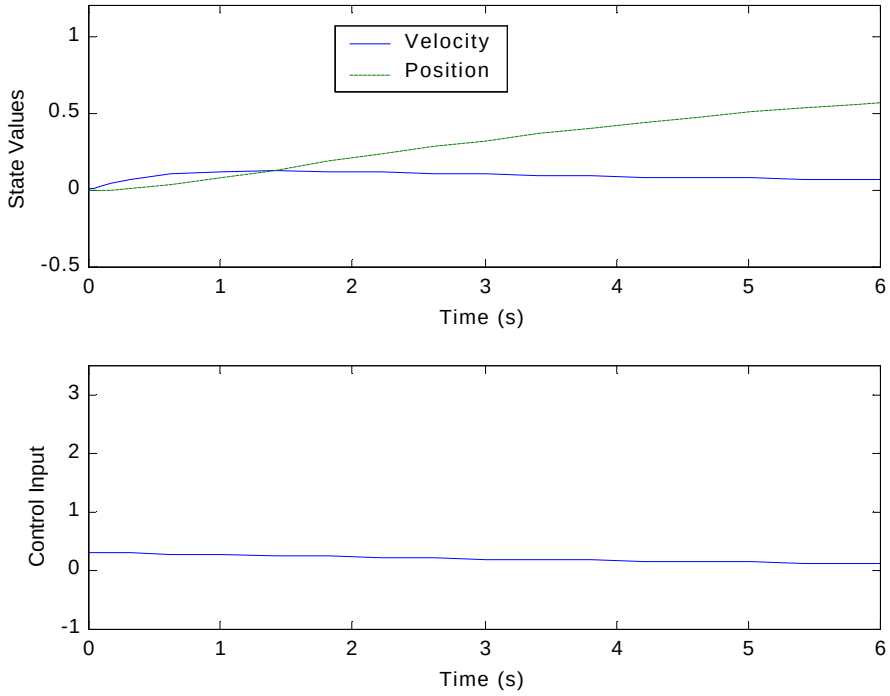


Figure 8.7: Control heavily penalized

If $\frac{q}{r}$ is large, the state is penalized heavily, resulting in a system with a small state that has a faster response. Large control effort requires higher energy components that are more expensive and cause more difficulty to implement. The larger the control effort, the more emphasis is placed on pushing the state to its steady state value. Figure 8.8 shows the effect of $\frac{q}{r}=5$ for q in $Q=\begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix}$.

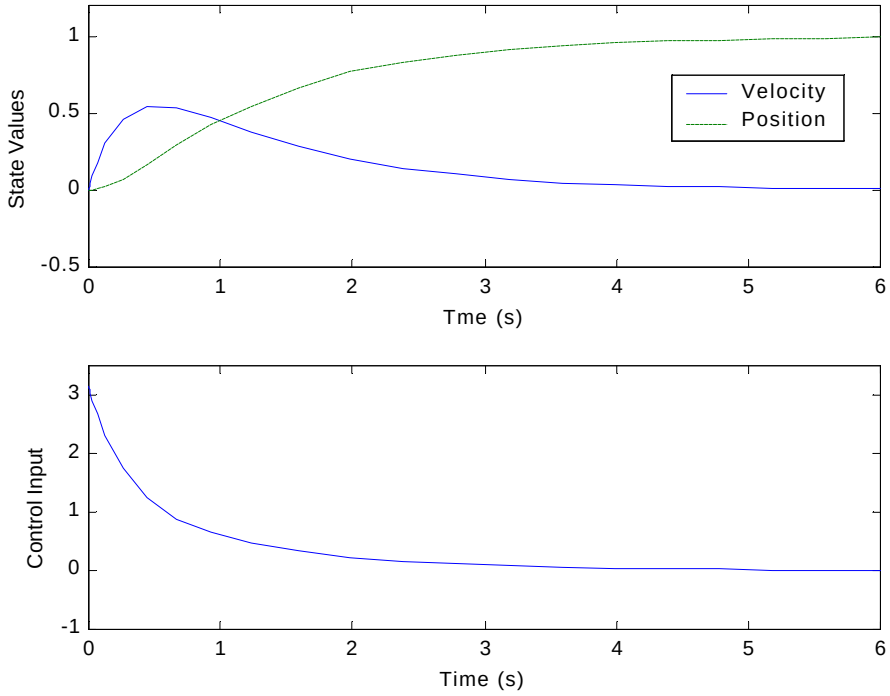


Figure 8.8: Control moderately penalized

Figure 8.9 illustrates a system where $\frac{q}{r}=100$. The control input is high and the system quickly approaches its settling time. The simulation is included as Encl(5) in the Appendix.

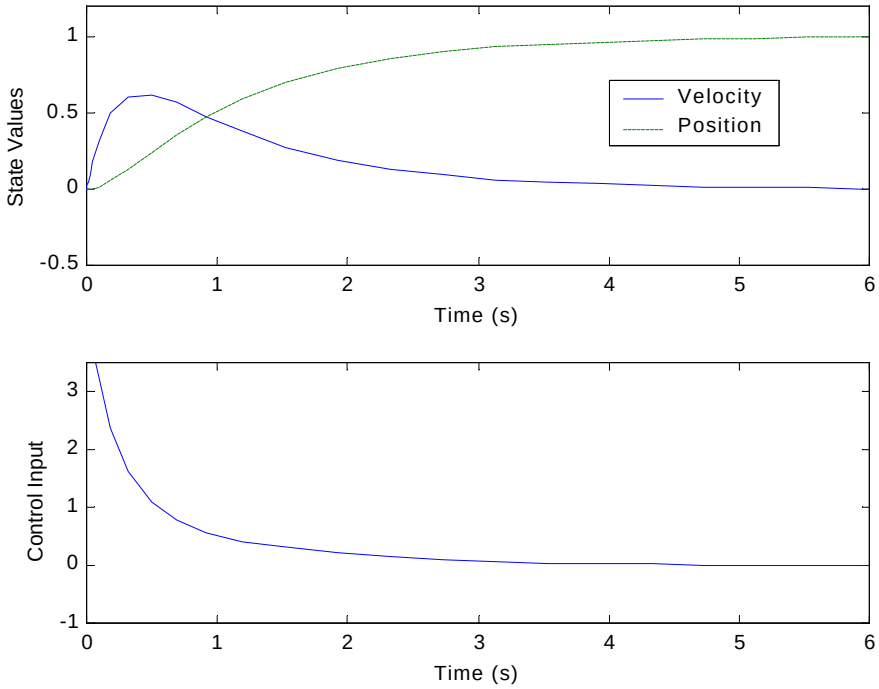


Figure 8.9: Control is lightly penalized

The case where $q = 0$ is useful in illustrating properties of the Riccati equation solution. If $q = 0$ the control is infinitely penalized relative to the state, allowing the system to use only the minimum possible control to maintain the stability of the system. If the plant is unstable ($a > 0$), the LQ optimal controller will reflect the pole at a about the imaginary axis to $-a$. If the system is already stable, no control effort will be used to change the system's response. The effect of $q = 0$ will be illustrated by the following solution of both an initially stable and unstable solution.

Riccati equation:

$$2ap - r^{-1}b^2 p^2 + q = 0$$

If $q = 0$ and $r > 0$:

$$2ap - r^{-1}b^2 p^2 = 0$$

$$p(2a - r^{-1}b^2 p) = 0$$

$$p = 0 \quad , \quad p = 2arb^{-2}$$

If $a < 0$, it follows that $p = 0$ and $k = 0$. The plant is stable and, therefore, the closed-loop state matrix is equal to the open-loop state matrix, a. If $a > 0$, the open-loop system is unstable and $p \neq 0$ for closed-loop system stability. Using $p = 2arb^{-2}$, the expression for k:

$$k = r^{-1}bp = 2ab^{-1}$$

The closed-loop state becomes, $a - bk = a - 2a = -a$. The unstable pole in this system is reflected through the imaginary axis, making the system stable while using the minimum control effort.

Optimizing the observer is also possible through the use of a Kalman Filter. A Kalman Filter will estimate the state of a plant given a set of known inputs and a set of measurements. Like the controller optimization process, the observer's Kalman Filter solution is obtained by solving a Riccati equation. The Kalman Filter is designed to minimize the variance of the state estimation error when noise corrupts the state and output of the system. The Kalman Filter will produce the optimal observer given that the noise sources are white and Gaussian.[16]

9 ETB Controller

The ETB controller limits the driver's steering input rate, effectively restricting the rate the vehicle tires can turn in order to prevent the driver from turning the vehicle beyond its dynamic limits. The ETB controller does not, however, affect the driver's control over vehicle

[16] Friedland.

position or orientation. The driver maintains similar control over the vehicle's position and orientation available in a vehicle unequipped with the ETB controller. The ETB controller is truly a driver assistance control system with a strong dependence on the driver.

9.1 Control Strategy

In terms of the ASMV platform, the ETB controller reacts to driver input produced by the driver loop, simulating the human driver and maintaining vehicle lateral position and orientation on the treadmill. As discussed in the Section 5, the driver loop performs the same functions as a human driver in normal driving situations. The ETB controller responds to driver input when the driver loop controllers are modified to produce a "bad" response, one that produces too great a yaw rate. When the yaw rate is too great, the vehicle is in danger of a catastrophic occurrence such as a rollover or skid. The bad response simulates an overanxious driver who has turned the wheel at a rate greater than the vehicle dynamics will safely allow. Quick lateral movements, requiring a large steering input over a short time, to avoid a stationary obstacle or another vehicle on the road are a common driving experience.[17] Once activated the ETB controller will maintain the vehicle yaw rate about the desired maximum safe yaw rate. Any attempt by the driver to turn the steering wheel faster than the maximum safe yaw rate will be met with increased steering input attenuation, physically preventing the driver from turning the steering wheel faster than the vehicle's dynamics will tolerate. The block diagram of the driver loop interrupted by the ETB controller is shown in Figure 9.1. The complete system model of the ETB controller as modeled in the software package SIMULINK is included in the Appendix as Encl(7).

[17] Brennan, 3.

State feedback-based optimal control of the yaw rate and lateral velocity does not require consideration of the vehicle's lateral position or steering angle states during control. Furthermore, the model the ETB controller governs must be modified because the vehicle model developed in Section 6 that includes all four states is not observable. Assuming only the yaw rate is measured, the other three states cannot be known through the use of a Kalman Filter because they have no dependence on the yaw rate. Without knowledge of all four states, state feedback control of the original model is not possible, and the advantages of state feedback control described in Section 8 are lost. If we choose to implement state feedback control without a Kalman Filter, all states must be measured independently. Measuring the lateral position, the most difficult state to measure, would require an external measuring system akin to the Global Positioning System. As a result, the ETB controller is based on a modified vehicle model that uses two states, yaw rate and lateral velocity, for control. The simplification of the model makes the system observable, allowing the lateral velocity of the vehicle to be estimated from the yaw rate using a Kalman Filter. The yaw rate is easily measured using onboard accelerometers or a gyro.

9.2 Implementation

In order to better understand the function of the ETB controller, a signal is followed through the controller starting with the comparison between the maximum safe yaw rate and the absolute value of the actual yaw rate. The difference is then passed through a switch connected to the *Toggle ON/OFF* block in the forward path of the ETB controller. If the actual yaw rate is less than the maximum safe yaw rate, a zero is sent to the *Toggle ON/OFF* block, essentially turning off the ETB controller. Otherwise, a one is sent to the *Toggle ON/OFF* block, enabling

the ETB controller. There is one other product block involved in the determination of the ETB controller output. The *Tracking* block takes one input from the maximum safe yaw rate and the other from the sign of the actual yaw rate. If the sign of the actual yaw rate is positive, the product block references the ETB controller to the maximum safe yaw rate. If the output of the *Tracking* block is negative, the ETB controller is referenced to the negative of the maximum safe yaw rate. The other signal entering the *Toggle ON/OFF* block is the ETB controller input. When the ETB controller is toggled on, the ETB controller input passes through *Toggle ON/*OFF and is added to the driver's steering input. The actuator block simulates the transmission of steering input to wheel angle. The steering input is the driver's input augmented by the ETB controller. The saturation block accounts for physical limitations of the steering system. The driver steering input is provided by the driver loop monitoring all four vehicle states, simulating the driver sloppily regulating the vehicle's position while making a quick lateral movement. The addition of the driver steering input and the ETB controller input enters both the simplified and the full vehicle model. The driver loop uses the full vehicle controller to maintain control of the vehicle, while the ETB controller uses the simplified vehicle model to control the yaw rate and lateral velocity.

Despite the ETB controller's limitations on the driver's input, the driver's ability to position the vehicle will not be affected. The ETB controller does not monitor or control the vehicle's position states.

9.3 ETB Controller Simulation Results

For all tests the maximum safe yaw rate was arbitrarily set at eight degrees per second. Examining the performance of a vehicle with the ETB controller engaged versus a vehicle with the ETB controller disengaged realized a tremendous improvement in the vehicle's performance during the ETB maneuver. For the uncompensated system, shown in Figure 9.2, the maximum yaw rate reaches 80 degrees per second for a sustained

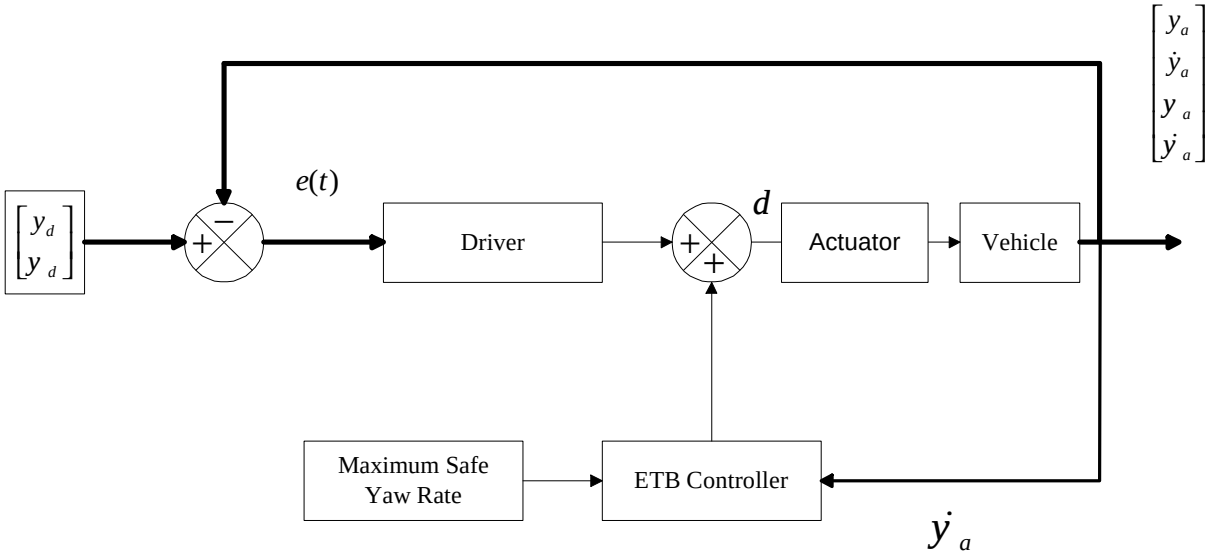


Figure 9.1: The ETB Controller

driver steering input of 30 degrees. The vehicle employing either version of the ETB controller, results in Figures 9.3 and 9.4, tested never exceeded twelve degrees per second of yaw rate. The implementation of the ETB controller drastically reduced the yaw rate during a quick lateral movement allowing the driver to maintain control of the vehicle even after an initially poor human reaction.

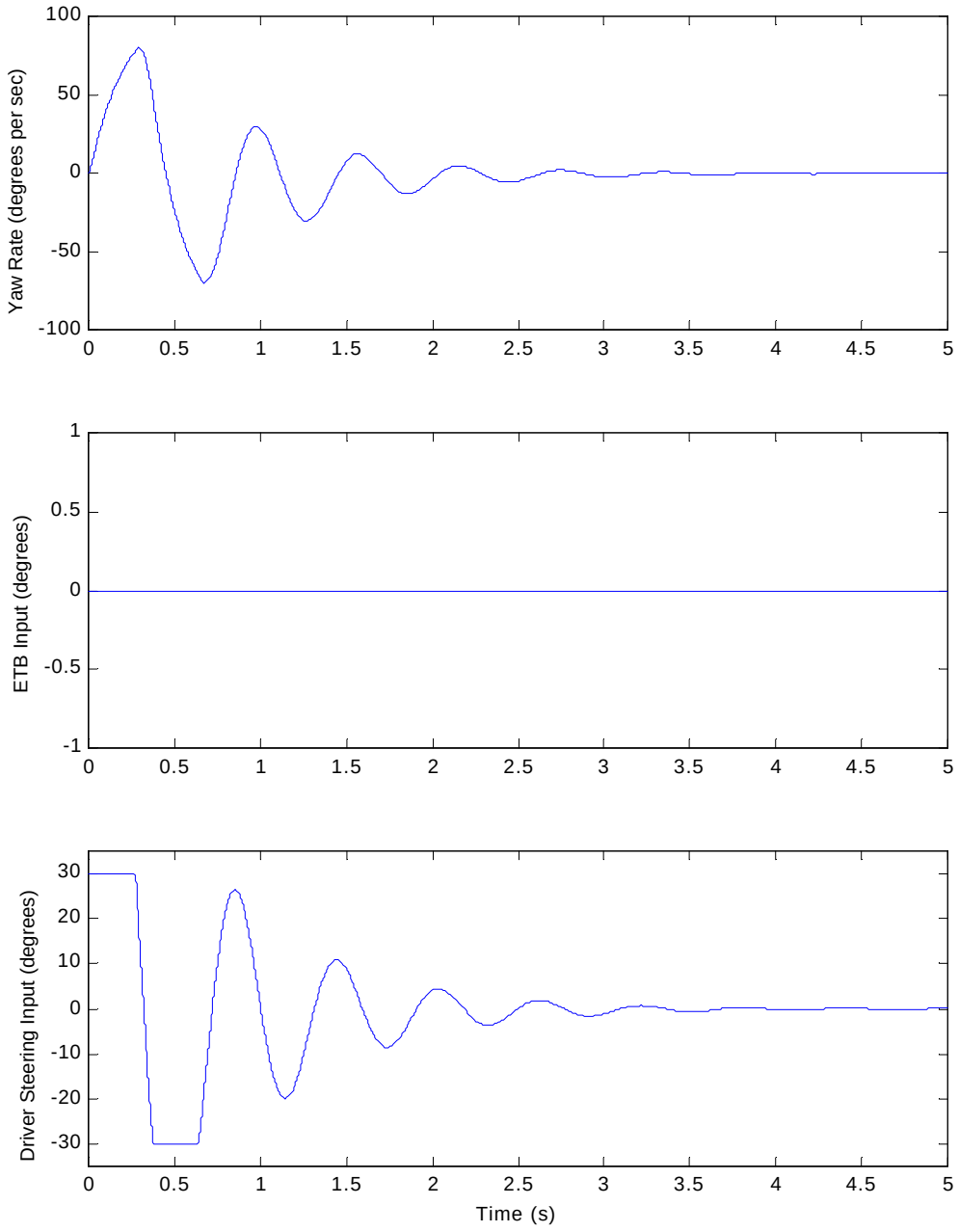


Figure 9.2: Yaw rate, ETB input, and driver steering input for a 0.3 m quick lateral movement with the ETB controller disengaged

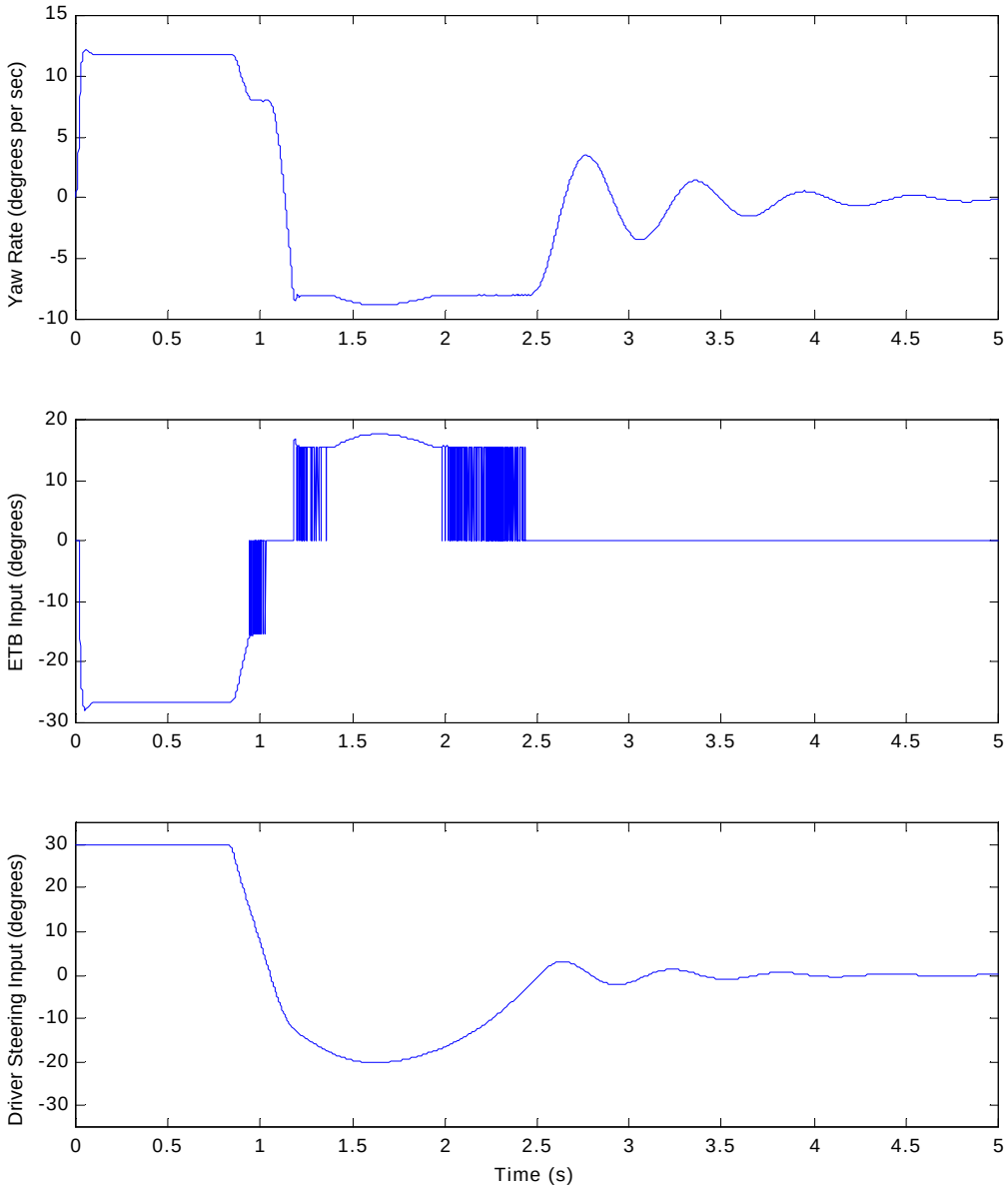


Figure 9.3: The driver input, ETB controller input, and the resulting yaw rate of the vehicle for quick lateral movement of 0.3 m with high control cost

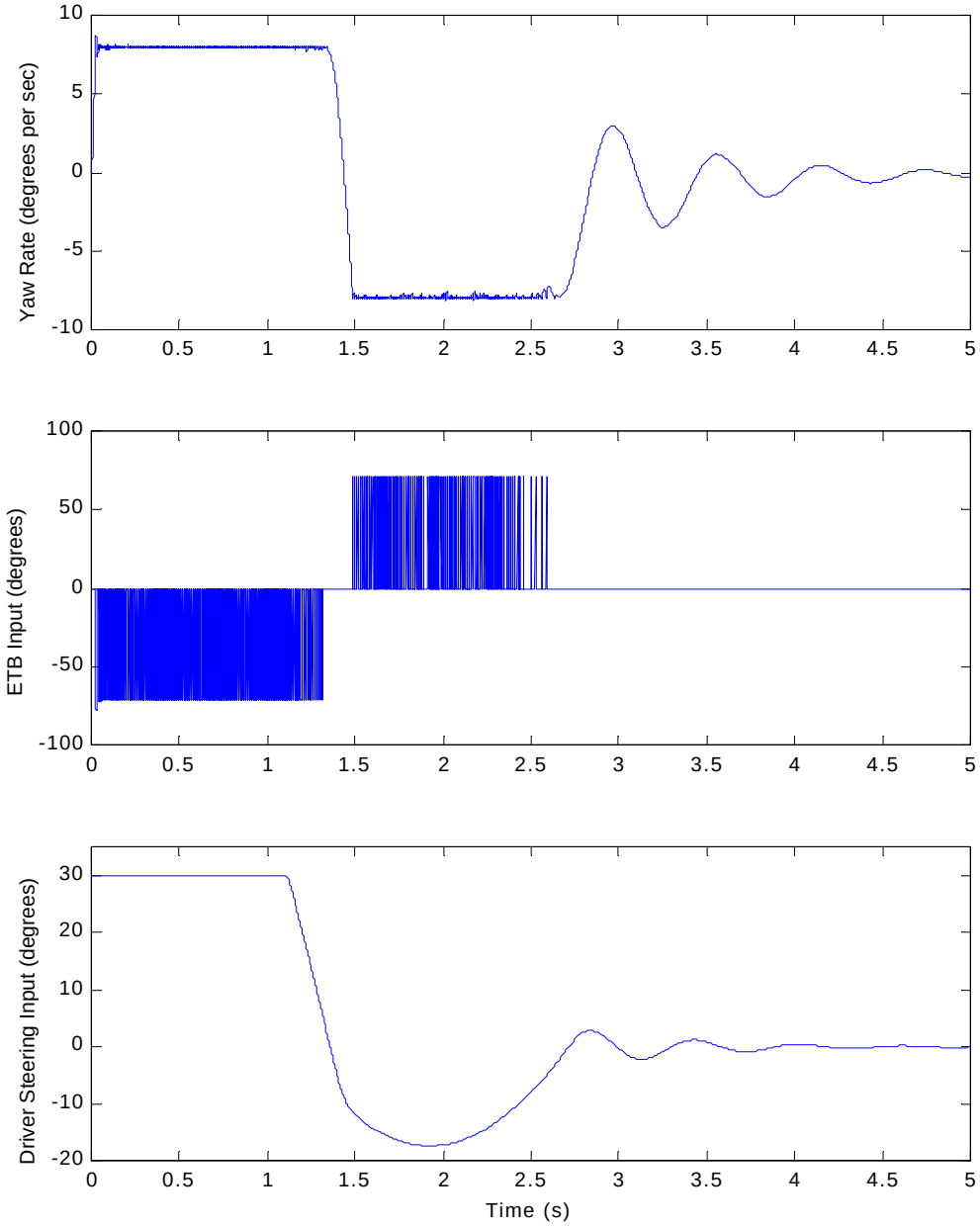


Figure 9.4: The driver input, ETB controller input, and the resulting yaw rate of the vehicle for quick lateral movement of 0.3 m with low control cost

For a high control cost where $Q = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix}$ with $a = 10$ and $r = 1$ the ETB controller limits the vehicle yaw rate to a maximum rate of 12 degrees per second. For the majority of the simulation the yaw rate is held below the maximum safe yaw rate of eight degrees per second. Setting a high cost for control yields a comparably weak controller overwhelmed by the driver when the wheels are turned as far as possible (30 degrees for the scale-model vehicle). During the first second of the simulation the driver has the wheels hard over and despite the ETB controller's input, the yaw rate remains approximately 12 degrees per second, which exceeds the maximum safe yaw rate for the simulation of eight degrees per second. As the vehicle moves past the desired position and the driver turns the wheels hard in the other direction, the ETB controller once again is unable to keep the yaw rate below the maximum safe yaw rate. The driver does not turn the wheels as far on this oscillation and thus the maximum safe yaw rate is exceeded by less than a degree per second. The yaw rate, ETB controller input, and driver steering input for the high cost controller are included in Figure 9.3.

For a low control cost where Q remains the same, $a = 100$ and $r = 1$, the ETB controller maintains the yaw rate below the maximum safe yaw rate even when the driver has the wheels turned hard over. The low control cost allows the controller to exert more effort in restraining the yaw rate. The resulting maximum ETB controller input of 80 degrees was nearly a six-fold increase over the controller input for the high cost controller. The yaw rate, ETB input, and driver steering input for the low cost controller simulation are included in Figure 9.4.

Comparing the three remaining states, lateral position, yaw angle, and lateral velocity between the simulations for high cost, shown in Figure 9.5, and low cost,

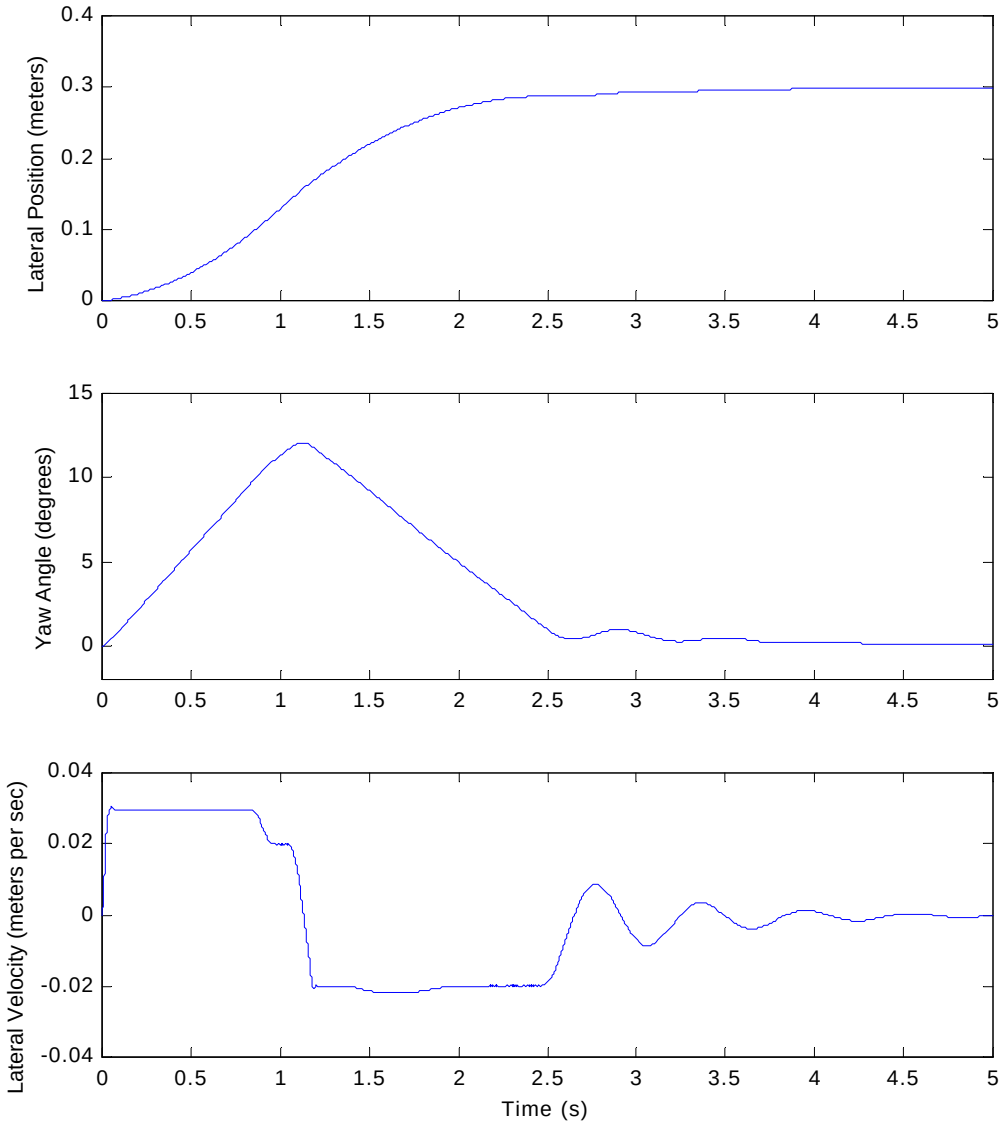


Figure 9.5: The lateral position, yaw angle, and lateral velocity for quick lateral movement of 0.3 m with high control cost

shown in Figure 9.6, controllers yielded some significant results. The low cost controller slightly extended the settling time of the system. The peak response of the yaw angle remained the same for both systems at 12 degrees, but the low cost controller resulted in a longer settling time. The

lateral velocity was more limited by the low cost controller as result of the relationship between lateral velocity and yaw rate. By

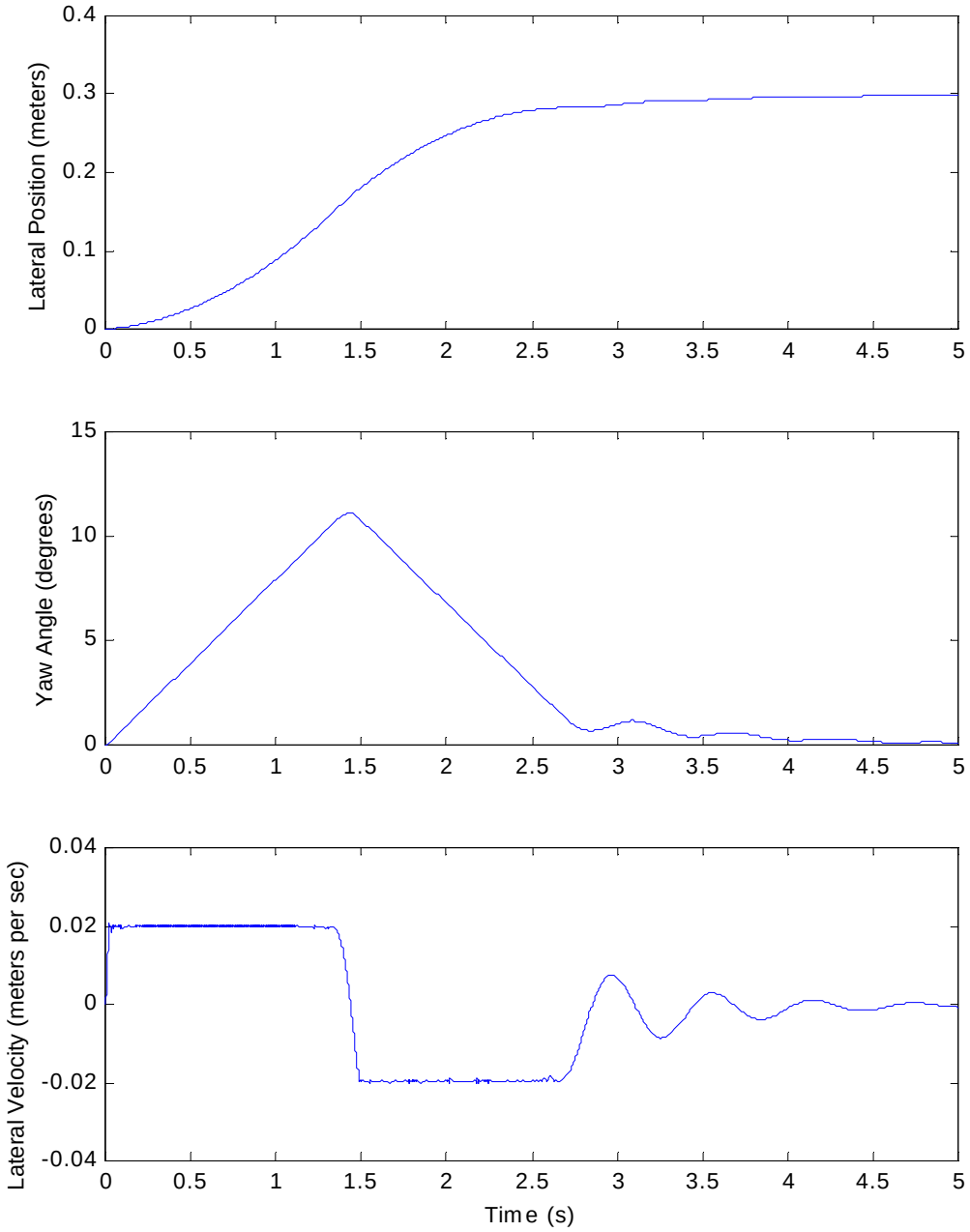


Figure 9.6: The lateral position, yaw angle, and lateral velocity for quick lateral movement of 0.3 m with low control cost

limiting the yaw rate, the lateral velocity is indirectly limited. For the high cost controller, the lateral velocity was limited to 0.03 meters per second, while the low cost controller limited the lateral velocity to 0.02 meters per second. The lateral velocity state experienced the same increased oscillatory nature the other states demonstrated.

For the uncompensated system, the settling times for all states were slightly lower than for either vehicle with the ETB controller engaged. Although the settling time is lower the yaw rate and lateral velocity required to generate the lower settling time are unreasonable and would likely result in the driver losing control of the vehicle. The lateral position, yaw angle, and lateral velocity states for the uncompensated system are shown in Figure 9.6.

In summary, the low cost controller more effectively limited the rate states of the system creating a safer situation with the tradeoff of a marginally extended settling time and state oscillation. Both ETB controllers restrained the vehicle's yaw rate, adding to the driver's control over the vehicle and limiting dynamic instability of the system.

Figure 9.7 illustrates the resultant chatter of the ETB controller as the yaw rate transitions above and below the maximum safe yaw rate. The figure specifically shows the chatter for the high cost controller, but there is little difference in the chatter for high or low cost, except for the rate with which the chatter occurs. The chatter poses a significant problem for implementation of the ETB controller. If the ETB input was to be a mechanical input into the steering system the mechanical system would not be able to respond with the speed necessary to make the controller useful and the continuous rapid movement would cause mechanical fatigue and eventual system failure. The swift rate of operation of the ETB controller requires the driver input and ETB controller input to be

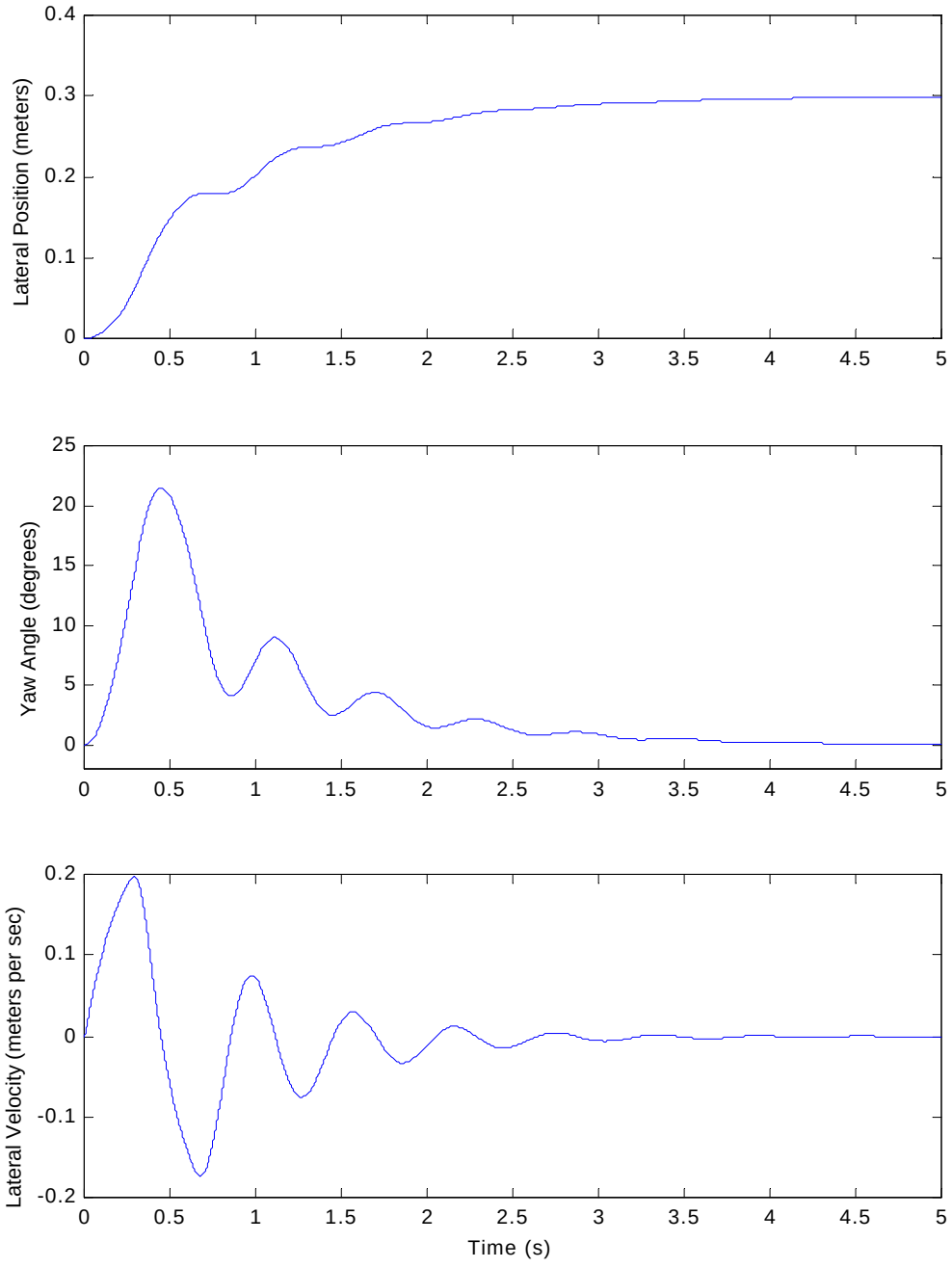


Figure 9.6: The lateral position, yaw angle, and lateral velocity for quick lateral movement of .3 m with ETB controller disengaged

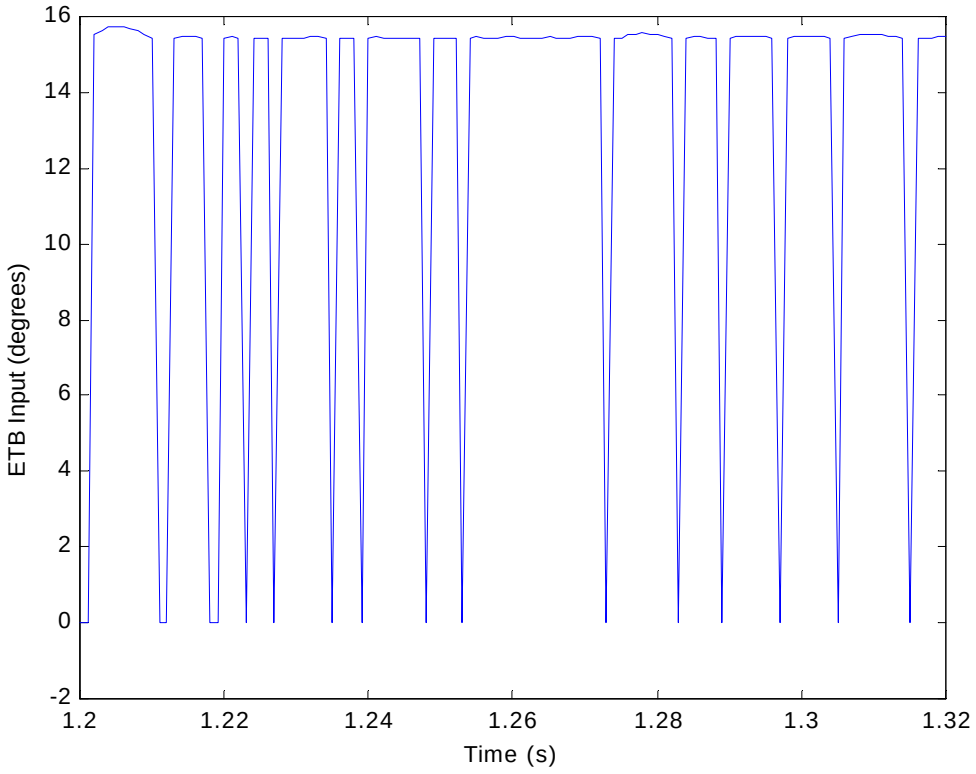


Figure 9.7: Illustration of ETB controller chatter as yaw rate transitions above and below maximum safe yaw rate

added electrically before the combined input enters a steering actuator. The chatter shown in Figure 9.7 will have no ill effects on an electrical system. The concept discussed here is a steer-by-wire vehicle, similar to fly-by-wire aircraft of today like the F-16 and F/A-18. In fly-by-wire aircraft, pilots do not directly manipulate the control surfaces of the aircraft. The pilot's input is sent to a controller that converts the input into actual control effort. Steer-by-wire vehicles are on the road today and this system, with extensive further development could be implemented on such a vehicle. In the simulation of this system, included in the Appendix as Encl (7), the steer-by-wire concept is implemented by summing the driver and ETB controller input before they enter the actuator block.

10 Ongoing Research

One of the major project goals from the outset of this project was to deliver the ASMV as a permanent addition to the resources of USNA. The ASMV will continue to be used next year as development of the ETB controller continues. Beyond next year, the development of additional control systems on the ASMV requires only the imagination and foresight to consider and test such systems. The hardware for doing so is already in place. The Alleyne Research Group has demonstrated the validity of the scale-model vehicle as a viable research medium for automotive study. With the ASMV, it is now the Naval Academy's turn to plunge into that endeavor.

11 Conclusion

The main objectives of the project have been satisfied. The Automotive Scale-Model Vehicle (ASMV) platform was designed and constructed in smaller scale, for a far smaller price, and with some significant improvements over the Illinois Roadway Simulator.[18] The ASMV can effectively regulate a scale-model vehicle moving at a constant velocity in a straight line subject to minor disturbances. The ASMV lateral and longitudinal controllers were successfully designed and implemented to maintain vehicle position in the center of the treadmill. The ASMV controllers can be easily modified to accommodate a driver assistance control system by altering the computer code that controls the vehicle and treadmill. During any driver assistance control system test, the driver loop continues to simulate the driver.

A three DOF model was developed and incorporated into the design of both the driver loop and the ETB controller. The three DOF model assumed a constant longitudinal velocity and

[18] Brennan.

the only forces present are the interaction of the road and tires. This model does not incorporate the effects of braking, but is a valid model for examining driver assistance control for sudden lateral motion.

Optimal control methods were researched and the knowledge gleaned from that endeavor was employed in the development of the ETB controller. The ETB controller is an optimal state feedback controller that utilizes a Kalman Filter to estimate the lateral velocity from the yaw rate measurement.

The ETB controller was designed and implemented in simulation for a quick lateral movement. The simulation results of the ETB simulation illustrated the viability of using a controller to limit the vehicle's yaw rate. The ETB controller successfully attenuated the driver steering input to keep the vehicle's yaw rate within the tolerable dynamics of the vehicle. The ETB controller increased driver safety during a quick lateral movement by decreasing the vehicle's response to exaggerated driver inputs, while continuing to move in the desired trajectory.

Bibliography

Brennan, Sean N. *Modeling and Control Issues Associated with Scaled Vehicles*. Masters Thesis, The University of Illinois, Urbana, IL, 1999.

Bretz, Elizabeth A. "Boston Builds a High-IQ Highway," *IEEE Spectrum*. August 2000, pp. 47-52.

Friedland, Bernard. *Control System Design: An Introduction to State-Space Methods.* New York: McGraw Hill Book Company, 1986.

Godbole, Datta N., et. al. "Design of Emergency Maneuvers for Automated Highway System: Obstacle Avoidance Problem." *Proceedings of the 36th Conference on Decision & Control*, San Diego, CA, December 1997.

Kassoff, Hal. "Deployment of ITS – Cornerstones of a Strategy." *ITS Quarterly*, vol. III, Fall 1995.

Kirk, Donald E. *Optimal Control Theory: An Introduction*. New York: Prentice Hall Inc., 1970.

Stefani, Raymond T., et. al. *Design of Feedback Control Systems*. Boston: Saunders College Publishing, 1994.

Wong, J.Y. *Theory of Ground Vehicles, Second Edition*. New York: John Wiley & Sons, Inc., 1993.

Appendix

Encl(1): Experimental Apparatus Control Program

Encl(2): Observer Example

Encl(3): Three DOF Model Simulation

Encl(4): Three DOF Model Simulation with Observer

Encl(5): Optimal State Feedback Gains Through Riccati Equation

Encl(6): Lateral Controller Simulation

Encl(7): ETB Controller Simulation

```
//Experimental Apparatus Control Program
//August 2001 - April 2002

#include <conio.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <string.h>
#include <time.h>
#include <bios.h>
#include <conio.h>
#include <iostream>


/*defined global constants*/
#define         ZERO            0       //null value
#define         SWEEPTIME       33      //single sweep time (ms) of full scan (not used)
#define         MSGDELAY        2       //delay (sec) to assure message is read
#define         BASEPORT        0x240   //base port address of DT55 I/O registers
#define         DISPLAY         0       //display command port address offset
#define         LIVE            0x0C050 //turn live display on command
#define         FREEZE          0x08050 //store and display command (col autoincrement enabled)
#define         COLINCR         0x050   //enable col autoincrement command
#define         ROWINCR         0x054   //enable row autoincrement command
#define         BLANK           0x010   //turn vision display off command
#define         PIXROW          0x0A    //pixel row select command port address offset
#define         PIXCOL          0x08    //pixel column select command port address offset
#define         PIXOFFSET       0x0C    //selected pixel port address offset
#define         AUTOPIXOFFSET 0x0E      //selected pixel port address offset with autoincrement
#define         MINROW          75      /minimum selectable image row (0-479)
#define         MAXROW          445     /maximum selectable image row (0-479)
#define         MINCOL          0       //minimum selectable image column (0-639)
#define         MAXCOL          639     //maximum selectable image column (0-639)
#define         WHITE           0x0FF   //maximum possible gray level (0-255)
#define         BLACK           0       //minimum possiblegray level (0-255)
#define         threshold 253

#define         end_str_thr     "SV7 M0"
#define         init_steer      "SV1 M128"
#define         init_thr        "SV7 M100"
#define         init_tread_speed 100

#define COM1    0x03F8
#define COM2    0x02F8
#define COM5    0xFC68
#define COM6    0xFC70

#define B2400                   0x30
#define B9600                   0x0C    //Defines baud rates
#define B19200                  0x06
#define B38400                  0x03

#define TRUE    1
#define FALSE   0
```

```
//int pic_Port = COM1;                      // Transmit and Receive Data Register
//int Baud = B19200;                        //Baud rate to talk to PIC
double k_thr = .01;
double k_only = 5.0;
float yc_d = 350.0;
float xc_d = 260.0;
int pic_Port = COM1;
int servo_Port = COM2;
float m_past;

/*function prototypes*/
void fg_send(int addr_offset, int value);
int fg_get(int addr_offset);
void InitPort(int baud, int pic_port);          //Initialize Port
void SendCharToPort(int chr, int pic_port);     //Send character to port
int ReadTxStatus(int pic_port);                 //Read transmitter tatus register
void specs(void);                               //Finds vehicle orientation and position
int correct(int xc, int yc, float phi);         //Contains vehicle controllers
void cursor(int row,int col);                   //Places cursor at centroid of vehicle



int main(void)
{
        int i, DONE = FALSE, speed=0;
        char char_t[3]="0x0", char_k_thr[5]="0x0", char_k_steer[5] = "0x0", end_str;
        int counter=0, flag=0, char_ind=0;

        clrscr();

         InitPort(B19200, pic_Port); // Initialize port communicating with treadmill control PIC
         InitPort(B9600, servo_Port); //Initialize port communicating with SV203

         printf("... Use ESC key to quit ... \n\n\n");

         for (i=0; i<=strlen(init_steer); i++){
                if (ReadTxStatus(servo_Port)){          //Makes steering angle zero
                        SendCharToPort(init_steer[i],servo_Port);
                        delay(1);
                } //end if
        }  //end for i

        if(ReadTxStatus(servo_Port)) SendCharToPort(13,servo_Port);
        if (ReadTxStatus(pic_Port)) SendCharToPort(init_tread_speed,pic_Port);

        delay(1);


                 //Initial speed
        for (i=0; i<=strlen(init_thr); i++){
                if (ReadTxStatus(servo_Port)){
                        SendCharToPort(init_thr[i],servo_Port);
                        delay(1);
                }  //end if
        }  //end for i
        if (ReadTxStatus(servo_Port)) SendCharToPort(13,servo_Port);


         while(DONE!=1){
```

```
                    if (( kbhit() ) && ( flag==0 )){

                    char_ind = getch();
                    switch(char_ind){
                              case '\x1B':
                              if (char_ind == '\x1B'){
                                        DONE = TRUE;
//Decreases vehicle and treadmill speed to zero when user terminates program
                                        if (ReadTxStatus(pic_Port)) SendCharToPort(0,pic_Port);
                                        delay(1);
                                        for (i=0; i<=strlen(end_str_thr); i++){
                                                  if (ReadTxStatus(servo_Port)){
                                                            SendCharToPort(end_str_thr[i],servo_Port);
                                                            delay(1);
                                                  }  //end if
                                        }  //end for i
                                        if (ReadTxStatus(servo_Port)) SendCharToPort(13,servo_Port);
                              }
                              break;
                              case 'K':
                              case 'k':
                                        cout << "Enter new throttle gain, k_thr" << "\n";
                                        flag = 1;
                                        break;
                               case 'T':
                               case 't':
                                        cout << "Enter treadmill speed (000-255)" << "\n";
                                        flag = 2;
                                         break;
                              }  //end switch(char_ind)
                    }  //end if
                    switch (flag){
                              case 1:       //Gain change
                                        if (( kbhit() ) && ( counter!=5 )){
                                                  char_k_thr[counter] = getch();
                                                  counter++;
                                        }  //end if
                                        else if (( kbhit() )&& ( counter==5 )){

                                        strcat(char_k_thr,".");

                                        k_thr = atof(char_k_thr);
                                                  counter=0;
                                                  flag=0;
                                                  cout << k_thr << "\n";
                                        }  //end else if
                              break;
                              case 2:      //Treadmill speed change
                                        if (( kbhit() ) && ( counter!=3 )){
                                                  char_t[counter]=getch();
                                                  counter++;
                                        }  //end if
                                        else if (( kbhit() ) && ( counter==3 )){

                                        speed=atoi(char_t);
                                                  cout << speed << "\n";

                                        ReadTxStatus(pic_Port);

                                        SendCharToPort(speed,pic_Port);
                                                  counter = 0;
```

```
                                        flag = 0;
                                    }  //end else if
                                break;
                               default:
                                        flag=0;
                        }  //end switch(flag)


                        fg_send(DISPLAY, COLINCR);//col autoincrement across rows
                        specs();


                } //ends while
                return 0;

}  //end main


/*****************************************************/
/*Function to send commands or data to the DT55    */
/*Data Translation frame grabber. The value of     */
/*'addr_offset' determines which frame grabber     */
/*function receives the command or data of 'value'.*/
/*****************************************************/
void fg_send(int addr_offset, int value){
        outport(BASEPORT+addr_offset, value);
}  //end fg_send


/*****************************************************/
/*Function whose value will be returned as the    */
/*integer data read from the DT55 Data Translation*/
/*frame grabber.  The value of 'addr_offset'      */
/*determines from where the data is read.         */
/*****************************************************/
int fg_get(int addr_offset){
        return inport(BASEPORT+addr_offset);
}  //end fg_get


void specs(void) {

        int i,j,pix,xc,yc;
        float phi,m[9]={0.0};


        fg_send(DISPLAY, LIVE);
        fg_send(DISPLAY, FREEZE);
        delay(30);


                        for (i=MINROW; i<=MAXROW; i+=3){
                fg_send(PIXROW, i);//x (row) address of pixel


                        for (j=MINCOL; j<=MAXCOL; j+=3){
                        fg_send(PIXCOL, j);//x (row) address of pixel
                                pix = fg_get(PIXOFFSET);

                        if (pix>threshold){
```

```
                    fg_send(PIXOFFSET,WHITE);
                    m[0] = m[0] + 1.0;
                    m[1] = m[1] + i;
                    m[2] = m[2] + j;
                    m[3] = m[3] + (1.0*i*j);
                    m[4] = m[4] + (1.0*i*i);
                    m[5] = m[5] + (1.0*j*j);

                }   //end if
                else
                    fg_send(PIXOFFSET,BLACK);

                                    }           //end for j

                    }           //end for i

    if(m[0]==0) m[0]=m_past;

                     xc = (int)(m[1]/m[0]);     //Computes vehicle centroid
    yc = (int)(m[2]/m[0]);

    m[6] = m[3] - (m[1]*m[2])/m[0];
    m[7] = m[4] - (m[1]*m[1])/m[0];
    m[8] = m[5] - (m[2]*m[2])/m[0];

                     phi = (180.0/3.14159)*(0.5*atan2(2*m[6],m[7]-m[8]));   //Computes vehicle orientation

                     cout << "Centroid = " << xc << ", " << yc << "\n";
                     cout << "Heading = " << phi << "\n";

    if(phi>0)
                    phi = 90.0-phi;
    else if(phi<90)
                    phi = -(90.0+phi);


    m_past=m[0];
    cout << phi << "\n";
    correct(xc, yc, phi);
                    cursor(xc,yc);
}           //end specs



int correct(int xc, int yc, float phi){
int port, change_thr_int, change_steer_int, i, steer_int;
float change_thr, change_steer, change_orien, change_lat_pos=0.0;
float change_lat_pos_last=0.0, lat_pos_error_last=0.0, yc_old=0.0;
char *change_thr_str, *thr_str, *base_thr_str;
char *change_steer_str, *steer_str, *base_steer_str;
time_t t, t_old, t_new;


thr_str = (char*)malloc(256);
change_thr_str = (char*)malloc(256);
base_thr_str = (char*)malloc(256);

steer_str = (char*)malloc(256);
change_steer_str = (char*)malloc(256);
base_steer_str = (char*)malloc(256);
```

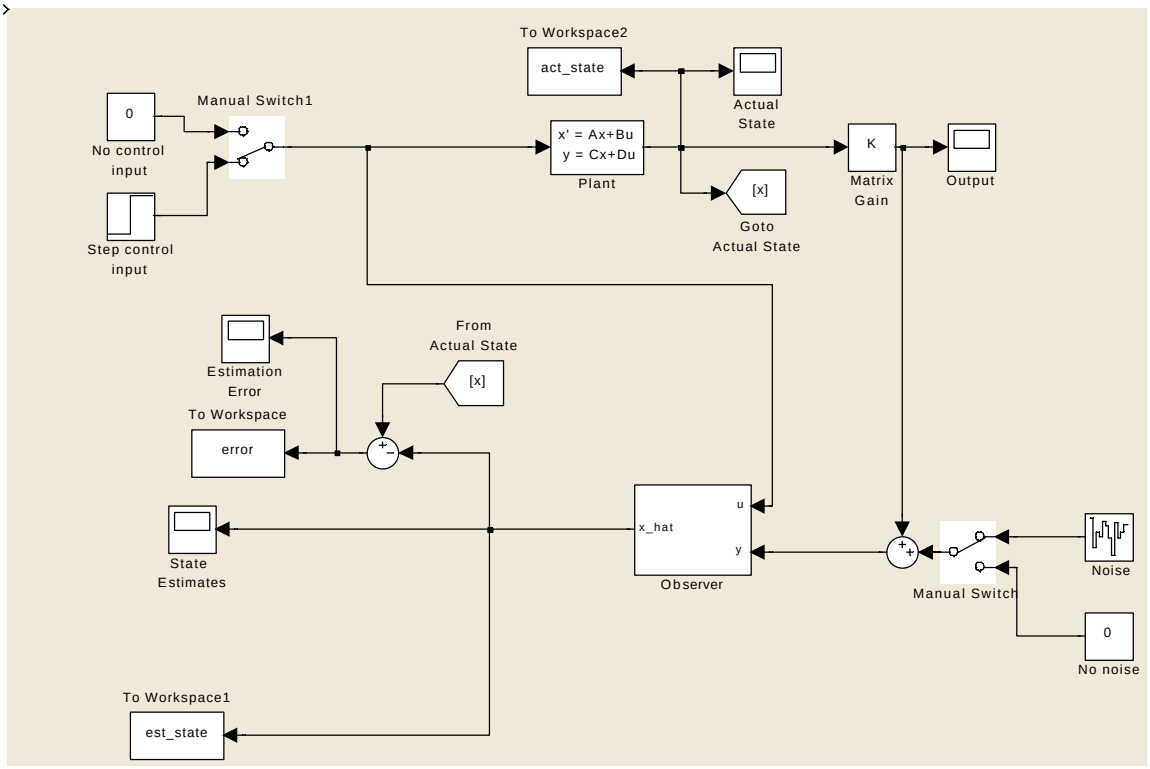
```
base_thr_str = "SV7 I";
base_steer_str = "SV1 I";

strcpy(change_thr_str,"\0");
strcpy(thr_str,"\0");

strcpy(change_steer_str,"\0");
     strcpy(steer_str,"\0");

                    t=time(NULL);




                    /*Throttle control*//////////////////////////////////////////////////

                    change_thr = k_thr*((double)yc_d-(double)yc);//*((yc-yc_old)/(t-t_old));

     change_thr_int = (int)change_thr;
     itoa(change_thr_int,change_thr_str,10);

     strcat(thr_str,base_thr_str) ;
     strcat(thr_str, change_thr_str);

     cout << thr_str << "\n";


                    //Sends throttle command to vehicle
                    for (i=0; i<=strlen(thr_str); i++){
                                        if (ReadTxStatus(servo_Port)){
          SendCharToPort(thr_str[i],servo_Port);
          //cout << i;
          delay(1);
        }
                    }  //end for i

     if(ReadTxStatus(servo_Port));
                    SendCharToPort(13,servo_Port);


                         //Steering control//////////////////////////////////////////////////////

     change_orien = k_only*( (double)phi*3.1459/180);

     //cout << change_orien << "\n\n";

     change_lat_pos = -.2762*change_lat_pos_last + 13.43*((double)xc_d-(double)xc)+5.165*lat_pos_error_last;

     change_lat_pos_last = change_lat_pos;
     lat_pos_error_last = (double)xc_d-(double)xc;

     change_steer = (.5*change_lat_pos + .5*change_orien)/5.0;

     if(change_steer >= 25.0)
          change_steer = 25.0;          //Filter to prevent too great a change in wheel angle
     else if(change_steer <= -25.0)
          change_steer = -25.0;
```

```
	steer_int = (int)change_steer;
	itoa(steer_int, change_steer_str, 10);

	strcat(steer_str, base_steer_str);
	strcat(steer_str, change_steer_str);

	cout << steer_str << "\n";

	//Sends steering commands to car
			for (i=0; i<=strlen(steer_str); i++){
					if (ReadTxStatus(servo_Port)){
		SendCharToPort(steer_str[i],servo_Port);
		//cout << i;
		delay(7);
	    }  //end if
			}  //end for i

				if(ReadTxStatus(servo_Port));
			SendCharToPort(13,servo_Port);


		free(change_thr_str);
	free(base_thr_str);
	free(thr_str);

	free(change_steer_str);
	free(base_steer_str);
	free(steer_str);

       //delay(300);
		return 1;
}  //end correct


//=================================================================
//=================================================================

   //The following port addresses have to be recomputed
   //in your program for each port change
   //int TxDreg = Port;                                   // Transmit Data Register (WO)
   //int RxDreg = Port;                                   // Receive Data Register (RO)
   //int BaudLreg = Port;       // Baud Rate Devisor Lower Half (WO)
   //int BaudUreg = Port+1;     // Baud Rate Divisor Upper Half (WO)
   //int IEreg = Port+1;                                  // Interrupt Enable Register
   //int IIreg = Port+2;                                  // Interrupt Identification Register (RO)
   //int LCreg = Port+3;                                  // Line Control Register
   //int MCreg = Port+4;                                  // Modem Control Register
   //int LSreg = Port+5;                                  // Line Status Register (RO)
   //int MSreg = Port+6;                                  // Modem Status Register (RO)
   //int SRreg = Port+7;                                  // Shadow Receive Register (RO) Only newer PCs

void InitPort(int baud, int pic_Port){   // Init Port

		char Ubaud, Lbaud;
		int IEreg, LCreg, BaudLreg, BaudUreg, MCreg;

		BaudLreg = pic_Port;       // Baud Rate Devisor Lower Half (WO)
		BaudUreg = pic_Port+1;     // Baud Rate Divisor Upper Half (WO)
		IEreg = pic_Port+1;                                  // Interrupt Enable Register
		LCreg = pic_Port+3;                                  // Line Control Register
```

```
        MCreg = pic_Port+4;                     // Modem Control Register

        // Turn off all interrupts in the IEreg  (00H)
        outportb(IEreg, 0x00);

        // Set LCreg to point to latch baud devisor (80H)
        outportb(LCreg, 0x80);

        // Latch the baud rate
        Ubaud = baud & 0xFF00;
        Lbaud = baud & 0x00FF;
  outportb(BaudUreg, Ubaud);
  outportb(BaudLreg, Lbaud);

  // Set LCreg to turn off the baud devisor latch and
  // setup b bits, 1 stop, no parity, no break ( 03H)
  outportb(LCreg, 0x03);

  // Set DTR and RTS and reset IntEnab in the MCreg (03H)
  outportb(MCreg, 0x03);
}  //end InitPort

//=====================================================================

void SendCharToPort(int chr, int pic_Port){             // Send Character to port

    int TxDreg;

    TxDreg = pic_Port;                                  // Transmit Data Register (WO)
    outportb(TxDreg, chr);
}  //end SendCharToPort


//=====================================================================

int ReadTxStatus(int pic_Port){  // Read Transmitter Status Reg

    int LSreg;

    LSreg = pic_Port+5;                         // Line Status Register (RO)
    if(inportb(LSreg) & 0x20) return 1;
    else return 0;
}  //end ReadTxStatus




/*void cursor(int row, int col){
        int i, pix, leg = 5;//cruciform height (width = (2*leg + 1) pixels)

        delay(SWEEPTIME);//delay for image stability

        //display horizontal segment
        fg_send(DISPLAY, COLINCR);
        fg_send(PIXROW, row);
        fg_send(PIXCOL, col-leg);
        for(i = col-leg; i <= col+leg; i++)
        {
                pix = fg_get(PIXOFFSET);
                fg_send(AUTOPIXOFFSET, pix+128);
```

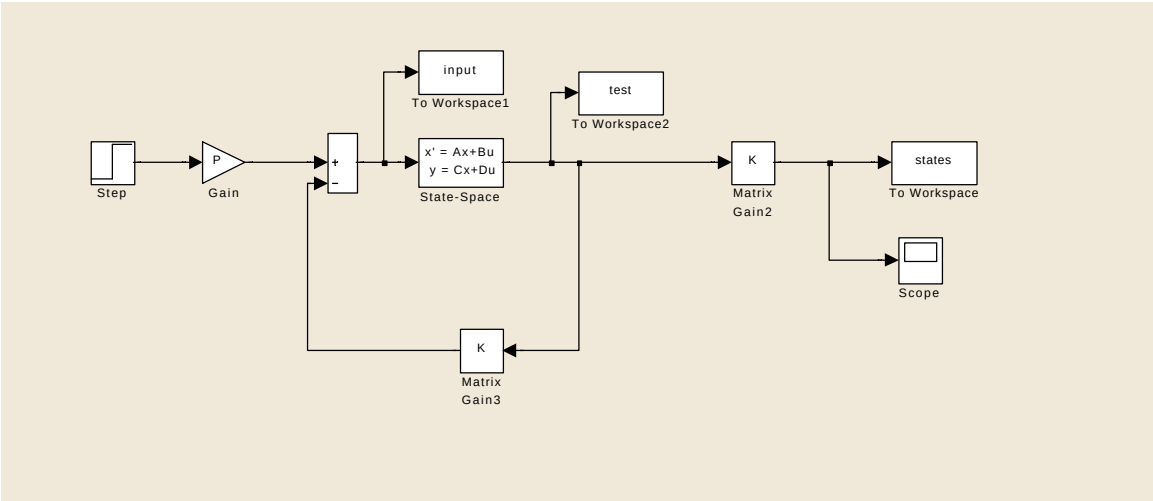
```
        }

        //display vertical segment
        fg_send(DISPLAY, ROWINCR);
        fg_send(PIXROW, row-leg);
        fg_send(PIXCOL, col);
        for(i = row-leg; i <= row+leg; i++)
        {
                pix = fg_get(PIXOFFSET);
                fg_send(AUTOPIXOFFSET, pix+128);
        }

}  //end cursor */
```

Encl(2): Observer Example



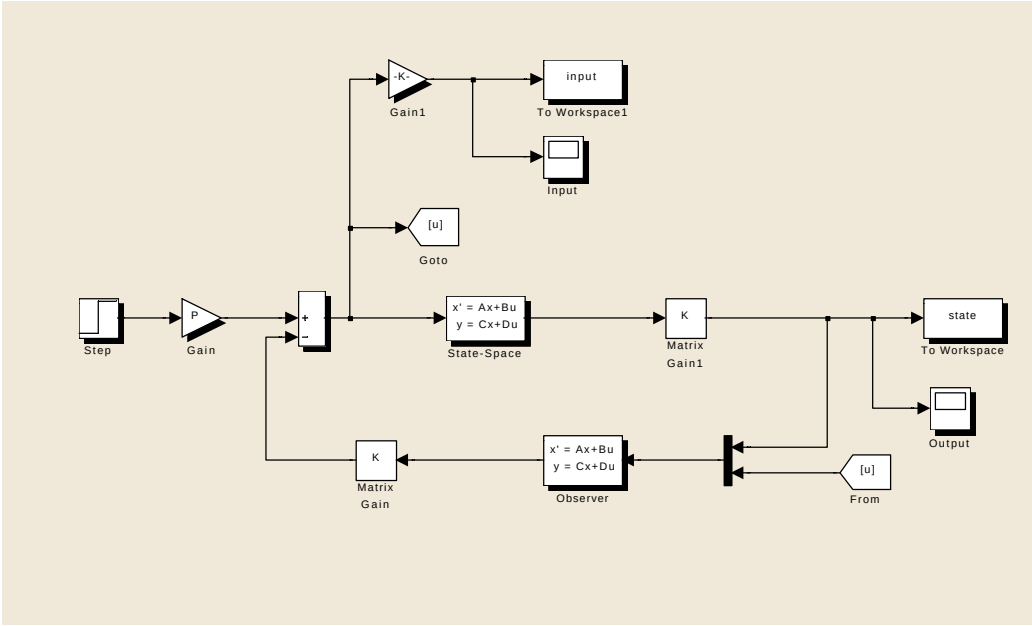

```
%Observer Example

format compact
clear
A = [-4 1;-3 0];
b = [0;6];                      %system definition
c = [1 0];
x0 = [1;1];
f = [-2.6;-2.5]                 %observer gains
sim('obsv_ex')

plot(tout,error)
hold on;
title('State Estimation Error')
xlabel('Time (s)')
ylabel('State Estimation Error')
hold off;
figure
plot(tout,act_state,tout,est_state)
hold on;
title('Comparison of Actual and Estimated States')
xlabel('Time (s)')
ylabel('State Value')
```

Encl(3): Three DOF Model Simulation




```
%Two Dimensional Model Simulation

format compact
clear two_dim_model_cont_obs
m = 1544+89+144;   %mass of car
l1 = 1.28;              %distance from front axle to CG
l2 = 1.72;              %distance from rear axle to CG
w = 1.519;              %width of car
cdt = .468;             %drag coefficient
Jx = 432.842;           %inertia about roll axis
Jy = 2711.2;            %intertia about pitch axis
Jz = 2746.04;           %intertia about yaw axis
Cs = 40000;             %cornering stiffness
vx = 25;                %velocity of car
fr = 0;                 %friction on the road

P = 1;                          %scales output so it matches desired output

A = [0 1 vx 0; 0 (-4*Cs/(m*vx)) 0 (-2*Cs*(l1-l2)/(m*vx) - vx); 0 0 0 1; 0 (-2*Cs*(l1-l2)/(Jz*vx)) 0 (-2*Cs*(l1^2+l2^2)/(Jz*vx))];
B = [0; 2*(Cs-fr)/m; 0; 2*(Cs-fr)*l1/Jz];
C = [1 0 0 0];%eye(4)
D = [0;0;0;0];

Mc=ctrb(A,B);                   %controllability matrix
rank(Mc);                               %if rank(Mc)!= 0, then the system can be transformed into controller form
t4 = [0 0 0 1]*inv(Mc);       %computes last row of Tc^-1
Tcinv = [t4*A*A*A;t4*A*A;t4*A;t4];
Tc=inv(Tcinv);

Ac = Tcinv*A*Tc
Bc = Tcinv*B
Cc = C*Tc
Dc = D;

pol_es=eig(Ac)

poles_desired=[-4+i*4 -4-i*4 -3.00778+i*8.85387 -3.00778-i*8.85387];

cond_eye = [1 0 0 0; 0 1 0 0; 0 0 1 0];
Acd_2 = poly(poles_desired);
Acd_1 = -Acd_2(2:5);
Acd = [Acd_1;cond_eye];
K = Ac(1,:)-Acd(1,:)
```

Encl(4): Three DOF Model Simulation with Observer

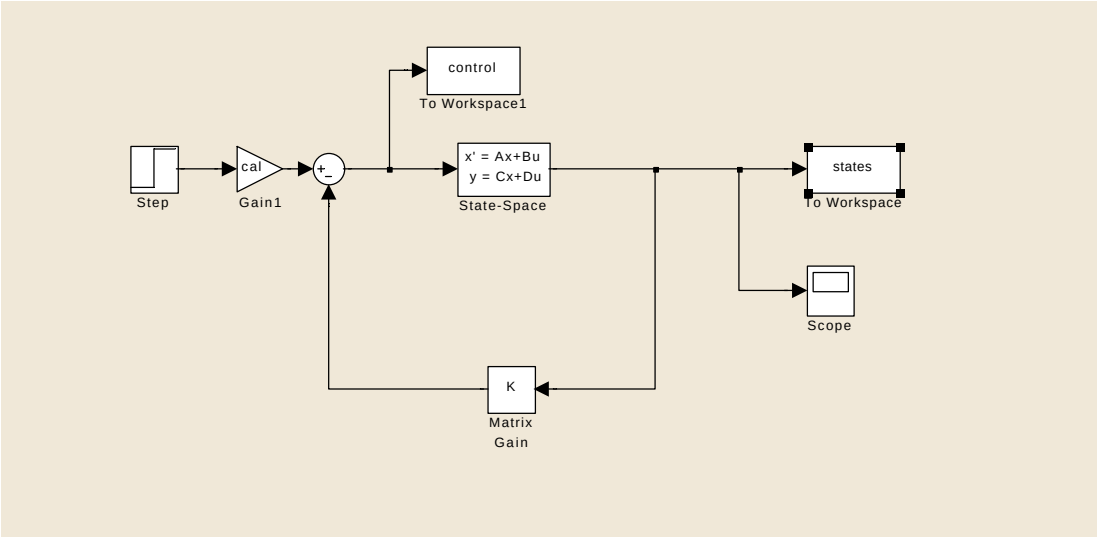


%Two Dimensional Model Simulation

```
format compact
clear two_dim_model_cont_obs
m = 1544+89+144;   %mass of car
l1 = 1.28;         %distance from front axle to CG
l2 = 1.72;         %distance from rear axle to CG
w = 1.519;         %width of car
cdt = .468;        %drag coefficient
Jx = 432.842;      %inertia about roll axis
Jy = 2711.2;       %intertia about pitch axis
Jz = 2746.04;      %intertia about yaw axis
Cs = 40000;        %cornering stiffness
vx = 25;           %velocity of car
fr = 0;            %friction on the road

P = 1;             %scales output so it matches desired output

A = [0 1 vx 0; 0 (-4*Cs/(m*vx)) 0 (-2*Cs*(l1-l2)/(m*vx) - vx); 0 0 0 1; 0 (-2*Cs*(l1-l2)/(Jz*vx)) 0 (-2*Cs*(l1^2+l2^2)/(Jz*vx))];
B = [0; 2*(Cs-fr)/m; 0; 2*(Cs-fr)*l1/Jz];
C = [1 0 0 0];%eye(4)
D = [0;0;0;0];

Mc=ctrb(A,B);                  %controllability matrix
rank(Mc);                      %if rank(Mc)!= 0, then the system can be transformed into controller form
t4 = [0 0 0 1]*inv(Mc);        %computes last row of Tc^-1
Tcinv = [t4*A*A*A;t4*A*A;t4*A;t4];
Tc=inv(Tcinv);

Ac = Tcinv*A*Tc
Bc = Tcinv*B
Cc = C*Tc
Dc = D;

pol_es=eig(Ac)

poles_desired=[-4+i*4 -4-i*4 -3.00778+i*8.85387 -3.00778-i*8.85387];

cond_eye = [1 0 0 0; 0 1 0 0; 0 0 1 0];
Acd_2 = poly(poles_desired);
```

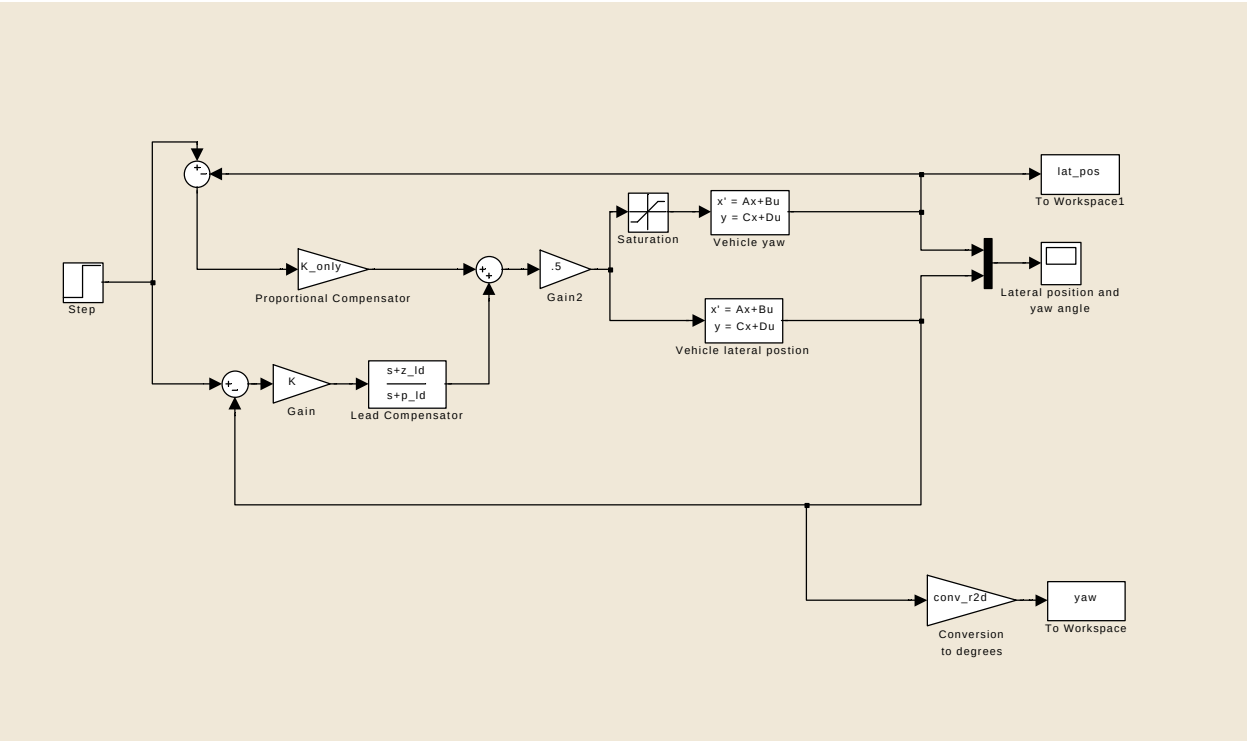
```
Acd_1 = -Acd_2(2:5);
Acd = [Acd_1;cond_eye];
K = Ac(1,:)-Acd(1,:)


Mo = [C; C*A; C*A*A; C*A*A*A];                  %forms Observability matrix
to = inv(Mo)*[0;0;0;1];                             %computes last column of To
To = [A*A*A*to A*A*to A*to to];                 %forms To

Ao = inv(To)*A*To
Bo = inv(To)*B                                      %computes coefficients of Observability form
Co = C*To
Do = D

hpoles_desired=[-50+i -50-i -100+i -100-i];
Aod_2 = poly(hpoles_desired);
Aod_1 = -Aod_2(2:5);
Aod = [Aod_1;cond_eye]';
h = Ao(:,1)-Aod(:,1)


observer = A-To*h*C
observer_b = [To*h B]
P = -1/(C*inv(A-B*K*Tcinv)*B);
sim('two_dim_model_cont_obs')
```

Encl(5): Optimal State Feedback Gains Through Riccati Equation



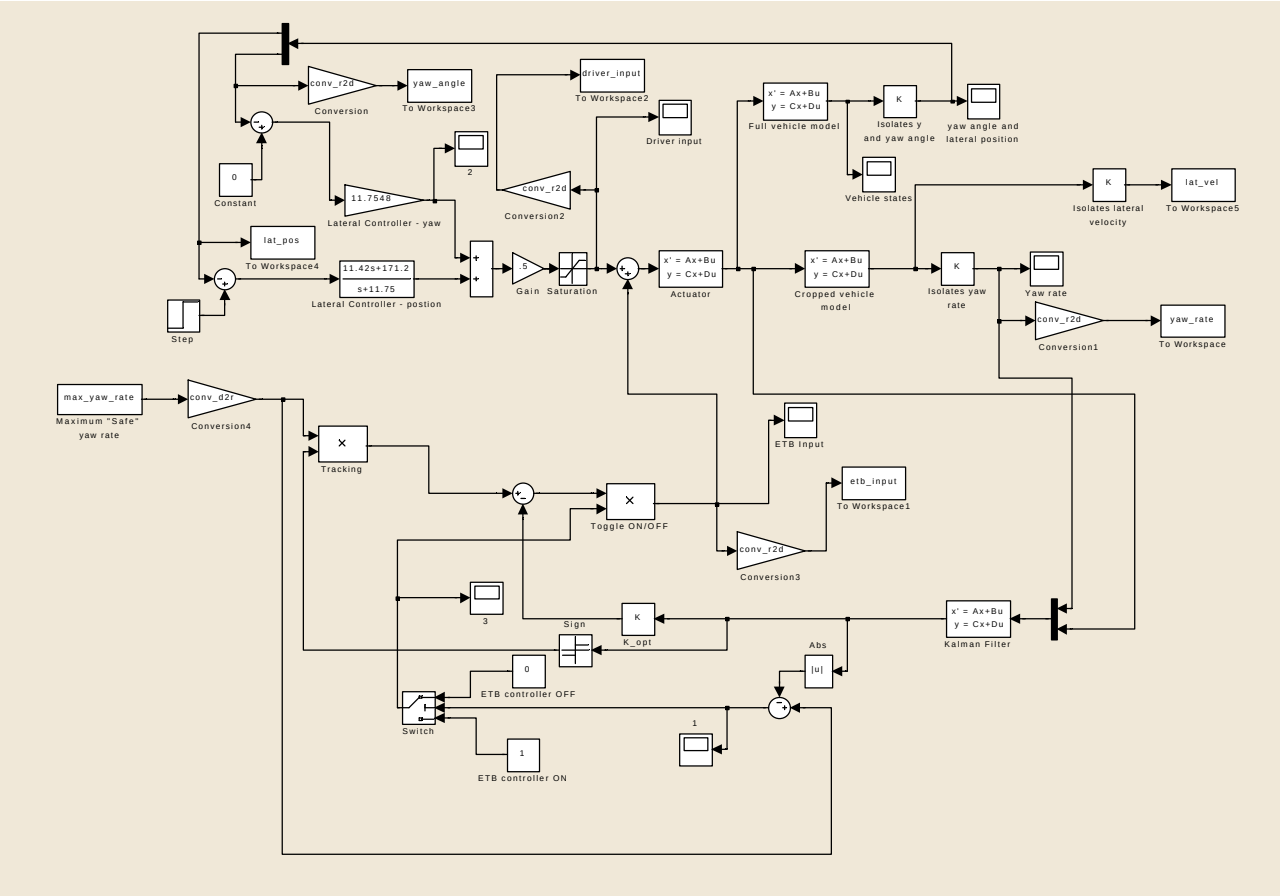

```
%Matrix Riccati Controller

format compact
A = [-2 0;1 0];
B = [1;0];                          %system defintion
C = eye(2)

q1 = 20
q2 =20
Q = [q1 0;0 q2]                     %costs
r = 1;

P = are(A,B*inv(r)*B',Q);           %Riccati equation

K = inv(r)*B'*P                     %state feedback gain

cal = -1/([0 1]*inv(A-B*K)*B);
sim('matrix_Ricc_sim')
figure
subplot(211), plot(tout,states)
subplot(212), plot(tout,contol)
```

Encl(6): Lateral Controller Simulation




```
%Lateral Controller Simulation

conv_d2r =pi/180;
conv_r2d = 180/pi;
m = 1.47;
Jz = .0236;
l1 = .13;
l2 = .15;
Cs = 46;
fr = 0;
vx = 1;

A = [0 1 vx 0; 0 (-4*Cs/(m*vx)) 0 (-2*Cs*(l1-l2)/(m*vx) - vx); 0 0 0 1; 0 (-2*Cs*(l1-l2)/(Jz*vx)) 0 (-2*Cs*(l1^2+l2^2)/(Jz*vx))];
B = [0; 2*(Cs-fr)/m; 0; 2*(Cs-fr)*l1/Jz];
lat_pos_isolation = [1 0 0 0];
yaw_isolation = [0 0 1 0];

D = zeros(size(lat_pos_isolation,1),size(B,2));

G_ss = ss(A,B,yaw_isolation,D);

z_ld = 15;
wgc = 40;
PM = 70;

[mag_pro,phase_pro] = bode(G_ss, 40);
```

```
K_only = 1/10^(mag_pro/20);


[mag,phase] = bode(G_ss, wgc);

c_angle = (-180 + PM - phase)*conv;
M_zld = sqrt(wgc^2 + z_ld^2);
zld_angle = atan(wgc/z_ld);
pld_angle = zld_angle - c_angle;
p_ld = wgc/tan(pld_angle)
M_pld = sqrt(wgc^2 + p_ld^2);
M_cld = M_zld/M_pld;
K = 1/(M_cld*mag);
G_c = K*tf([1 z_ld],[1 p_ld])

bode(G_ss*G_c)
figure
bode(zpk(G_ss)*K_only)

sim('lateral_control_sim');

subplot(2,1,1),plot(tout,yaw),ylabel('Yaw Angle (degrees)');
subplot(2,1,2),plot(tout,lat_pos),ylabel('Lateral Position (m)');
hold on;
xlabel('time (s)')
```



%ETB Controller Simulation

format compact

conv_r2d = 180/pi;
conv_d2r = pi/180;

m = 1.47;
Jz = .0236;
l1 = .13;
l2 = .15;
Cs = 46;
fr = 0;
vx = 1;

max_yaw_rate = 8;
max_lat_vel = 4;

A = [0 1 vx 0; 0 (-4*Cs/(m*vx)) 0 (-2*Cs*(l1-l2)/(m*vx) - vx); 0 0 0 1; 0 (-2*Cs*(l1-l2)/(Jz*vx)) 0 (-2*Cs*(l1^2+l2^2)/(Jz*vx))];
B = [0; 2*(Cs-fr)/m; 0; 2*(Cs-fr)*l1/Jz];
C = eye(4);
D = zeros(size(C,1),size(B,2));

```
A_crop = [(-4*Cs/(m*vx)) (-2*Cs*(l1-l2)/(m*vx) - vx);(-2*Cs*(l1-l2)/(Jz*vx)) (-2*Cs*(l1^2+l2^2)/(Jz*vx))];
B_crop = [ 2*(Cs-fr)/m; 2*(Cs-fr)*l1/Jz];
C_crop = eye(2);
D_crop = zeros(size(C_crop,1),size(B_crop,2));

Ga = tf(1,[.2 1]);
[f,g,h,j]=ssdata(Ga);

position_isolation = [1 0 0 0; 0 0 1 0];
yaw_rate_isolation = [0 1];
lat_vel_isolation = [1 0];

alpha =100;
Q = alpha*eye(2);
r = 1;

P = are(A_crop,B_crop*inv(r)*B_crop',Q);

K_opt = inv(r)*B_crop'*P;


sim('ETB_controller_sim');

subplot(3,1,1),plot(tout,yaw_rate),ylabel('Yaw Rate (degrees per sec)');
subplot(3,1,2),plot(tout,etb_input),ylabel('ETB Input (degrees)');
subplot(3,1,3),plot(tout,driver_input),ylabel('Driver Steering Input (degrees)');

hold on;
xlabel('Time (s)')
hold off;

figure
plot(tout,etb_input)
hold on;
xlabel('Time (s)');
ylabel('ETB Input (degrees)');
hold off;

figure
subplot(3,1,1),plot(tout,lat_pos),ylabel('Lateral Position (meters)');
subplot(3,1,2),plot(tout,yaw_angle),ylabel('Yaw Angle (degrees)');
subplot(3,1,3),plot(tout,lat_vel),ylabel('Lateral Velocity (meters per sec)');
hold on;
xlabel('Time (s)')
hold off;
```