Design of an Adaptive Cruise Control Model for Hybrid Systems Fault Diagnosis

DESIGN OF AN ADAPTIVE CRUISE CONTROL MODEL FOR HYBRID SYSTEMS FAULT DIAGNOSIS

BY

BENJAMIN BREIMER, B.Eng

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2012) McMaster University

(Computing and Software) Hamilton, Ontario, Canada

TITLE:	Design of an Adaptive Cruise Control Model for Hybrid Systems Fault Diagnosis
AUTHOR:	Benjamin Breimer B.Eng, (Mechatronics Engineering) McMaster University, Hamilton, On
SUPERVISOR:	Dr. Alan Wassyng and Dr. Mark Lawford
NUMBER OF PAGES:	xv, 167

Abstract

Driver Assistance Systems like Adaptive Cruise Control (ACC) can help prevent accidents by reducing the workload on the driver. However, this can only be accomplished if the driver can rely on the system to perform safely even in the presence of faults.

In this thesis we develop an Adaptive Cruise Control model that will be used to investigate Hybrid Systems Fault Diagnosis techniques. System Identification is performed upon an electric motor to obtain its transfer function. This electric motor belongs to a 1/10th scale RC car that is being used as part of a test bench for the Adaptive Cruise Control system. The identified model is then used to design a hybrid controller which will switch between a set of LQR controllers to create an example Adaptive Cruise Controller. The model of the controller is then used to generate fixed point code for implementation on the testbed and validation against the model controller. Finally a detailed hazard analysis of the resulting system is performed using Leveson's STPA.

Notation and abbreviations

Abbreviations

ABC:	Active Brake Control	24
ACC:	Adaptive Cruise Control	1
ARMAX:	Autoregressive-Moving-Average model with eXogenous Input Model	140
ARX:	Autoregressive model with eXogenous Input Model	139
BJ:	Box Jenkins Model	140
CACC:	Coordinated Adaptive Cruise Control	5
CC:	Cruise Control	3
CCC	Cluster of Continuous Controllers	40
CFS:	Car Following System	24
DAS:	Driver Assistance System	4
DC:	Direct Current	79
DES:	Discrete Event System	18
DPU:	Data Processing Unit	40
EDESA:	Extended Discrete Event System Abstraction	18
ETFE:	Empirical Transfer Function Estimate	71

FSA:	Finite State Automata	20
FSM:	Finite State Machine	40
FTA:	Fault Tree Analysis	119
IIR:	Infinite Impulse Response	89
IVS:	Inter-Vehicular Sensor	24
LQR:	Linear Quadratic Regulator	106
LIDAR:	Light Detecting and Ranging	5
MIMO:	Multiple-Input Multiple-Output	48
MPC:	Model Predictive Control	6
MUX:	Multiplexor	58
OE:	Output Error Model	140
PID:	Proportional-Integral-Derivative Controller	42
PD:	Proportional-Derivative Controller	6
PRBS:	Pseudo Random Binary Sequence	72
PSD:	Power Spectrum Density	76
RC:	Remote Control	11
SISO:	Single-Input Single-Output	48
STAMP:	Systems-Theoretic Accident Modelling and Processes	123
STPA:	Systems-Theoretic Process Analysis	123
SVD:	Singular Value Decomposition	86
TCC:	Traditional Cruise Control	24
VSS:	Vehicle Speed Sensor	24

Symbols

$\mathbf{e}_{track}$: The difference between the desired headway, as specified by h_{set}, and the actual headway of the system h^*. Consequently, $e_{track} = h_{set} - h^*$

$\mathbf{e}_{pace}$: The difference between the desired velocity, as specified by V_{set}, and the actual velocity of the subject vehicle V_{self} . Consequently, $e_{pace} = V_{set} - V_{self}$

$\mathbf{e}_{pace_{rel}}$: The difference between the velocity of the leader vehicle V_{lead} and the velocity of subject vehicle V_{self} . Consequently, $e_{pace_{rel}} = V_{lead} - V_{self}$

$\mathbf{h}^*, \mathbf{h}_{self}$: The current headway between the subject and leader vehicles

$\mathbf{h}_{brake}$: The braking threshold headway

$\mathbf{h}_{max}$: The maximum following headway for normal following behaviour

$\mathbf{h}_{min}$: The minimum following headway for normal following behaviour

$\mathbf{h}_{safe}$: The minimum safe following headway

$\mathbf{h}_{set}$: The desired following headway

$\mathbf{h}_{setmax}$: The upper bound on the following headway for which the tracking error need not be reduced further

$\mathbf{h}_{setmin}$: The lower bound on the following headway for which the tracking error need not be reduced further

$\mathbf{V}_{set}$: The driver requested cruising velocity

$\mathbf{tol}_{pace}$: A predefined value that specifies the maximum value e_{pace} for which the system is not obligated to further reduce e_{pace}

$\mathbf{tol}_{set}$: A predefined value that specifies the maximum value e_{track} for which the system is not obligated to further reduce e_{track}

$\mathbf{tol}_{track}$: A predefined value that specifies the maximum value e_{track} for which the system is not obligated to begin reducing e_{track}

Contents

List of Figures

Chapter 1

Background and Literature Review

1.1 Introduction

Adaptive Cruise Control (ACC) is a Driver Assistance System that can help to improve the safety of our roads by reducing the workload on the driver. By extending traditional Cruise Control (CC) with the ability to adjust its velocity based on traffic conditions, ACC can be used in many situations where traditional CC would be impractical.

However, before the advantages of ACC can be leveraged, drivers must be able to trust it to perform safety even in the presence of faults. Fault Diagnosis techniques can be used to detect when a fault has occurred so that the appropriate action can be taken.

The goal of this thesis is to develop a hybrid systems model of Adaptive Cruise Control for use in the development and testing of fault diagnosis techniques on a test

bed involving a Remote Control (RC) car. Additionally, since many of the fault diagnosis and control techniques require a model of the system, a realistic representation of the RC car will be obtained.

1.2 Hybrid systems

Many systems can be described as either evolving continuously or discretely with respect to time. Systems that evolve continuously are often represented by a set of differential equations, while discrete systems are commonly represented as a state machine.

In a continuous system, there are an infinite number of states that the system could be in and the system evolves through these different states as a function of time. In contrast, a discrete system the transition between states occur as the result of an event or by some condition being met.

In a state machine each state is represented by some shape, often an oval, and the available transitions are represented by directed edges between the states. These transitions are labelled with a guard condition and the name of the event(s) that can trigger it. If a transition is triggered, the guard condition must first be evaluated to see if it will be allowed to occur.

While many systems fall into one of these two categories, more complex systems often contain aspects of both continuous and discrete systems. Hybrid systems theory provides a method for describing systems which fall into the latter category. However, understanding a hybrid system as merely one in which both continuous and discrete aspects are present would likely be inadequate. The reason for this claim is that much of the functionality of the hybrid system comes not just from the co-existence of the

continuous and discrete aspects but in their interactions.

A typical hybrid system contains a set of discrete states in addition to the continuous states that are found in a continuous system and which will evolve over time. The key difference is that in a hybrid system the manner in which the continuous states will evolve is dependent on the current discrete state.

Consider for example the longitudinal dynamics of a bicycle. The rider can continuously adjust the speed of the bike by adjusting the amount of force applied through the pedals. In this way the rider has a degree of continuous control over the bike's motion. However, there are also a number of discrete control decisions that can change the way the bike responds to the rider's pedalling. When the rider changes gears, the continuous dynamics of the bike will also change. This can be seen in the fact that, after the gear change, the same amount of force on the pedals will result in a different response from the bike. This is an example of a discrete control decision changing the continuous dynamics of a system. Other discrete control decisions the rider can make include choosing whether or not to choose to apply or release the brakes.

However, not all discrete control decisions are controllable. The bursting of a tire or breaking of a chain are both examples of discrete events that will change the dynamics of the bike even if they were undesired.

1.3 Adaptive Cruise Control systems

Cruise Control (CC) is a driver assistance system that controls the longitudinal dynamics of the host vehicle. By modulating the vehicle's throttle signal it attempts to maintain a user-requested velocity. Adaptive Cruise Control (ACC) extends this system by sensing the presence of a leader vehicle and adjusting the vehicle's speed

accordingly. However, the majority of ACC systems are designed only to operate at highway speeds and will often have a minimum velocity of around 30km/h (Shakouri & Ordys, 2011).

Stop&Go systems were developed to further extend ACC to be able to handle these slower speeds and the style of driving that goes with it. In (Yoshinori Yamamura & Murakami, 2001) the development of a Stop&Go system is discussed along with some of the challenges that arise from low speed driving including smaller inter-vehicular spacing and more frequent changes in velocity.

Adaptive Cruise Control and other Driver Assistance Systems (DAS) have been active areas of research for a number of years and as a result the literature on the topic is rather diverse. However, despite their differences, most ACC systems have at least two things in common.

The first is that any ACC implementation requires a method of gaining information about its environment. Information is needed both about the host vehicle's own continuous state (velocity, acceleration, etc.), as well as information about the presence and behaviour of a leader vehicle. Information about the vehicles own continuous state can be obtained using the sensors that are already a part of the vehicle. An example of this is the wheel speed sensor which provides information to the driver about their velocity via the speedometer.

Information about the lead vehicle's continuous state is more difficult to come by and requires additional hardware. The most common method for gaining this information is by mounting a Radar[1] or Radar-like sensors to the front of the vehicle. Included in this category are Microwave and Doppler radar units as well as the light

[1]RADAR is an acronym for RAdio Detection And Ranging

based LIDAR[2] .

As part of the development process in (Girard et al., 2005), the authors carried out a comparison of three of these types of sensors. Doppler Radar was found to work well at highway speeds with its range of 100 m. It also has the ability to provide information about the range, range rate[3] and azimuth for multiple targets. The azimuth information can be useful in determining whether any of the lead vehicles are in the process of changing lanes. However, the Doppler radar is unable to sense vehicles with zero relative velocity (Girard et al., 2005). Microwave radar, operating in the millimetre wave region, was found to be useful only in Stop&Go situations since it only had a range of 40 m. However, its value in Stop&Go scenarios was shown in its use in (Yoshinori Yamamura & Murakami, 2001). LIDAR had the disadvantage that its output is provided in the form of distance and intensity values for a plane of view. As a result, the data had to undergo processing before the system could gain an understanding of the leader vehicle's behaviour.

Other ACC systems supplement the information from their radar sensor by assuming that the lead vehicle will radio back information about its velocity and acceleration. ACC Systems that rely on this form of information are known as Coordinated Adaptive Cruise Controls (CACC) . Examples of such systems are developed in (Dew, 2002) and (Girard et al., 2005) .

The second aspect that all ACC systems must contain is the ability to take the information provided by the sensors and use it to implement some form of control. While the methods used to implement this control can range greatly from one system

[2]LIDAR stands for LIght Detection And Ranging

[3]The range is the distance between the two vehicles. The range rate is the rate at which this distance changes, it is in effect the relative velocity

to another, any ACC system has to be able to safely adjust its speed in the presence of a leader and maintain a velocity set point in the absence of one.

For this purpose, the ACC system is given access to the vehicle's throttle and brakes and must use each as needed. How the system employs each depends on the control method chosen for the implementation. A wide range of control methods have been used in the literature, from simple Proportional-Derivative (PD)(Girard et al., 2005) feedback control to methods which use Model Predictive Control (MPC)(Kural & B.A., 2010; Shakouri & Ordys, 2011; Li et al., 2011) to obtain their goals. Other types of control that have been implemented for this purpose include Linear Quadratic Regulators (LQR)(Junaid et al., 2005) and Sliding Mode Control (Hedrick & Yip, 2000).

The implementations in (Kural & B.A., 2010; Shakouri & Ordys, 2011; Li et al., 2011) use Model Predictive Control (MPC) in a hierarchical architecture. The top level controllers in these systems were created using Model Predictive Control to determine the trajectory of the vehicle based on a set of goals and constraints while relying on simple linear controllers to control the actuators. The importance of meeting multiple design criteria when developing an ACC system is discussed in (Li et al., 2011). To demonstrate this point, the authors show how MPC can be used to create a design that will have this ability. The example system they design optimizes for fuel economy, tracking accuracy and driver comfort. Also discussed in (Li et al., 2011) is the use of an "Action Point", the idea being that the system will not take action to further reduce the tracking error until it exceeds some threshold.

In (Kural & B.A., 2010), the authors point to MPC's ability to naturally integrate constraints into the optimization process as one of the advantages of this approach to

control. The system in (Kural & B.A., 2010) was developed and tested against a high-fidelity non-linear model of a vehicle under different traffic situations. The authors of (Shakouri & Ordys, 2011) also developed a pair of detailed non-linear models of their vehicle to serve as the basis of their control. The need for two models in their work arose from the fact that the vehicle's continuous dynamics changed depending on if the throttle or brakes were currently engaged.

Although the use of MPC is popular in the literature, it does have several drawbacks including its tendency to be computationally expensive. Alternatives to Model Predictive Control techniques include sliding mode control as discussed in (Hedrick & Yip, 2000) and (Dew, 2002), or the use of LQR controllers as discussed in (Junaid et al., 2005).

The system discussed in (Girard et al., 2005) uses a hybrid control system which switches between Cruise Control mode, Adaptive Cruise Control mode and Coordinated Adaptive Cruise Control (CACC) mode. Each of these modes have their own control law which is used to calculate the desired acceleration. The result is then passed down to a low level controller for implementation. The low level controller is then responsible for deciding whether the brakes or throttle should be used to achieve this goal. The mode that the system is in depends on the information provided by the radar sensor. If there is no lead vehicle then the Cruise Control system will be activated and the system will attempt to achieve the velocity set point using a PD controller. However if there is a lead vehicle present, the system will have to decide between Adaptive Cruise Control and Coordinated Adaptive Cruise Control, based on whether or not the leader is communicating back its continuous state. The purpose of both the ACC and the CACC systems is to maintain a predetermined following

distance; this is done through use of sliding mode control.

Most of the systems that have been discussed use a hierarchical or federated architecture. For example in (Girard et al., 2005) and (Li et al., 2011), the top level controller will determine the desired acceleration and will rely on the lower level controller(s) to implement it. However, the system in (Shakouri & Ordys, 2011) uses a different structure. In this design the low level controller is similar to a traditional Cruise Control system. A simple PI controller is used to achieve whatever velocity is requested by the top-level controller. The top level controller will choose this velocity set point according to the current traffic conditions and ACC objectives.

While the use of a hierarchical architecture is very popular, some other designs like (Junaid et al., 2005) use a more integrated approach. The system presented in this paper uses an LQR controller that takes care of determining both the current ACC objective as well as how to achieve it.

Other areas of research within the field of ACC systems include the use of formal verification techniques (Loos et al., 2011; Jairam et al., 2008), the problem of string stability (Liang & Peng, 1999) and the effect of ACC on drivers (Lees & Lee, n.d.; Han & Yi, 2006). Additionally, studies like the one presented in (Han & Yi, 2006) study driver's habits to better understand how to make the ACC system behave more like a human driver.

1.4 Software Dependability and Fault Tolerance

Since, the goal of this thesis is to produce a model that can be used to develop new fault diagnosis techniques, it would be helpful to briefly cover some of the work that has been done in this field. By looking at what has been done in the past, the need for new techniques can be better understood.

Fault tolerance is the ability of a system to "continue operating properly in the event of the failure of some of its components"(Chandhrasekaran & Choi, 2009). Fault tolerance can take many forms in a system, the most obvious is through the addition of redundancy. An example of such an approach is the N-Version programming method in which N teams of software developers independently work from a set of requirements to develop N versions of the same software. In many cases, each team is required to develop their solution to the problem using a different approach to software development and may even be working from slightly different sets of requirements. The idea is simple, if N-versions of the system are running, then their results can be compared by a voter unit. Then whichever value was chosen by the majority of the subsystems will be applied.

Intrinsic in this method is the assumption that each version of the software will be independent of the others. To test this assumption, John Knight and Nancy Leveson carried out an experiment in (Knight & Leveson, 1986). The findings of this study were that the assumption of independence can fail for a number of reasons including common assumptions, mistakes or oversights in the requirements, and commonalities in the way that programmers are educated. Additionally, by producing multiple versions of each component, the budget for development is spent towards diversity rather than focusing the developer's attention on a single version.

In (Sha, 2001), these problems are cited as proof of the argument that the way to improve software reliability is through simplicity rather than diversity. Their claim is that developing a simple version of the program, that is simple and understandable, will yield more robust results than an N-version solution. This approach is made difficult by the fact that adding features naturally increases the complexity of a system. Their solution to this problem is to create two versions of the system. The first version would be designed with simplicity and verifiability with formal methods in mind. The second could then include all of the required features. At run-time, the simple system could then be used to check the results of the more complicated system.

A similar argument is made in (Wassyng et al., 2012) about developing software for safety critical applications. The claim is that when developing safety-critical software, the control system can become rather complex. Therefore, to ensure the system remains safe, an independent safety system should be developed to monitor the behaviour of the system. The result is once again, a complex system that is difficult to verify, being monitored by a simpler system which has undergone formal verification.

Chapter 2

Introduction and Preliminary information

In this thesis we develop of a Hybrid Systems model for Adaptive Cruise Control. The intended use for this model is to aid in the design and testing of fault diagnostics techniques, especially those that exploit the hybrid nature of the controller.

In a related project, a test bed involving a Remote Control (RC) car was prepared for the testing of automotive safety algorithms. The Adaptive Cruise Control system developed in this thesis was designed to make use of this test bed and its RC car.

Since many of the fault diagnosis and control techniques require a mathematical model of the system, a realistic representation of the RC car was needed. This need was met by using a System Identification process to obtain a model of the car. This model was then used to develop the Hybrid Systems Adaptive Cruise Controller which consists of both a Finite State Machine (FSM) and a bank of LQR controllers.

The resulting model was then used to generate fixed point C code capable of controlling the RC car. The design of the ACC system was also subjected to a

rigorous hazard analysis using STPA.

2.1 Representing a discrete system

Hybrid systems can be modelled in a number of different ways, each having its own strengths and weaknesses. According to (Mohammadi, 2009), the use of Petri nets and Finite State machines is popular among computer scientists, while systems engineers prefer to use a set of differential equations. An example of the latter are switching systems, as explained by (Goebel et al., 2009). A switching system is an equation whose right hand side is determined by a set of conditions. An example switching statement is shown in equation (2.1)

$$f(t) = \begin{cases} \textit{Equation } 1 & \text{if} \quad \textit{condition } 1 \\ \vdots & \vdots \quad \vdots \\ \textit{Equation } n & \text{if} \quad \textit{condition } n \end{cases} \tag{2.1}$$

The representation that will be reflected in this thesis is that of a hybrid automata. The hybrid automata can be thought of as an extension of the state machine discussed previously. Each discrete state in the hybrid automata includes a set of differential equations. These equations represent the continuous dynamics of the system while in that state. A reset map is also included in the hybrid automata to describe the effect of a discrete transition on the continuous dynamics.

2.1.1 Notation of a general hybrid automaton

In (Goebel et al., 2009), the authors state that the hybrid automata usually consists of the following components:

A set of modes, Q

A domain map, $Domain : Q \rightrightarrows \mathbb{R}^n$

A flow map, $f : Q \times \mathbb{R}^n \to \mathbb{R}^n$

A set of edges, $Edges \subset Q \times Q$

A guard map, $Guard : Edges \rightrightarrows \mathbb{R}$

And a reset map, $Reset : Edges \times \mathbb{R}^n \to \mathbb{R}^n$

Note that in this notation $\rightrightarrows$indicates a set value mapping while $\to$ denotes a function.

The flow map in this definition is a state-dependant differential equation that represents the continuous dynamics of the system when that state is active.

As an example of this notation consider the bicycle example from Section 1.2. The different gears would be represented as different discrete modes. The set of discrete modes could also be used to represent whether or not the brakes are being applied. In this case the set of discrete modes Q would include one mode for first gear with brakes applied, and another for first gear with no brakes applied. The continuous variable on the other hand would be the velocity of the bike.

The set of edges would be the transitions between these modes. Their presence or absence would show which transitions are possible. For example an edge would exist between first and second gear, and between second and third gear. However, there

would be no edge between the first and third gear. So to transition from the first gear to the third, the rider would have to transition into second gear temporarily.

The guard map would be a set of logical statements found on individual edges. For a transition to occur over an edge, its guard condition must first evaluate to true. A guard condition from the bike example would be that a transition between gears can only occur if the pedals are moving.

The domain map for each discrete mode would be the range of values which the continuous state could take on while in this mode. The domain map of each gear in the bicycle example would be the range of velocities the bike can achieve while in that gear.

Finally, the flow map would be a set of differential equations that describe the bike's continuous dynamics based on the current mode Q.

An extended notation for fault diagnosis

This definition is sometimes extended, as can be seen in Mohammadi's work on fault diagnosis. In (Mohammadi, 2009), Mohammadi describes his definition of the hybrid system by a 14-tuple.

$$H = (Q, \mathcal{X}, \mathcal{U}, \mathcal{Y}, FT, Init, S, \Sigma, T, G, \rho, D, \lambda, q_0)$$

Where

Q is the set of finite discrete states

$\mathcal{X} \subseteq \mathbb{R}^n$ is the set of vector spaces of continuous state

$\mathcal{U} \subseteq \mathbb{R}^p$ is the set of vector spaces of continuous input

$\mathcal{Y} \subseteq \mathbb{R}^r$ is the set of vector spaces of continuous output

FT is the set of known fault types that are modelled in this notation

$Init \subseteq \mathcal{X}$ is the set of initial continuous states

$S = \{S_q | q \in Q\}$ is the set of dynamic models defining the continuous dynamics of the system

Σ is the set of symbols representing the discrete events labelling the transitions between discrete states

$T \subseteq Q \times \Sigma \times Q$ is the set of discrete transitions

$G : T \times \mathcal{X} \times \mathcal{U} \rightarrow \{True, False\}$is the set of guard conditions

$\rho : T \times \mathcal{X} \rightarrow \mathcal{X}$ is a reset map D is the set of discrete output symbols

$\lambda : Q \rightarrow D$ is the discrete output map

q_0 is the initial discrete state

2.2 Introduction to Hybrid Fault Diagnosis

Hybrid systems can be an effective way to model complex systems which display both continuous dynamics as well as discrete modes. If these complex systems are being developed for a safety critical application then it would be highly desirable to be able to identify when a fault has occurred within the system.

The literature contains numerous papers on the topics of fault tolerance and fault diagnosis for complex systems like Adaptive Cruise Control. However, one of the most interesting techniques we discovered was introduced by Mohammadi in his PhD dissertation (Mohammadi, 2009). The method he introduced utilizes the hybrid systems representation of complex systems to develop a fault diagnosis framework. He

then demonstrated its use on a jet turbine engine. We believe that his work could also be applied to more software intensive systems like Adaptive Cruise Control.

To investigate this possibility, as well as other fault diagnosis techniques, a model of an ACC system was developed. However, before going into the details about the design and implementation of the ACC model, a brief survey of Mohammadi's work is in order. The following discussion will highlight the principles behind Mohammadi's method since it is a good example of how a hybrid system representation can be used to create more diagnosable systems.

2.2.1 The Hybrid Diagnoser

Mohammadi introduced "A hybrid framework for passive on-line fault diagnosis in systems modelled by hybrid automata" (Mohammadi, 2009). The term "passive on-line diagnosis" refers to a system that will, at run-time, monitor the system's outputs to determine its health. As a passive diagnoser, the health of the system will be determined without interjecting test inputs; instead it will rely on the outputs the system gives during its operation.

The advantage of a hybrid diagnoser over one that relies on either purely continuous or purely discrete diagnosis techniques is a greater ability to diagnose the system. By combining the information obtained from the continuous diagnosers with knowledge of the current discrete state, faults that would have otherwise gone undiagnosed can be detected and isolated.

The key to constructing Mohammadi's Hybrid Diagnoser is to obtain an abstraction of the system that will integrate the continuous and discrete sensors while modelling the system as a Finite State Automata. In his thesis, Mohammadi refers to

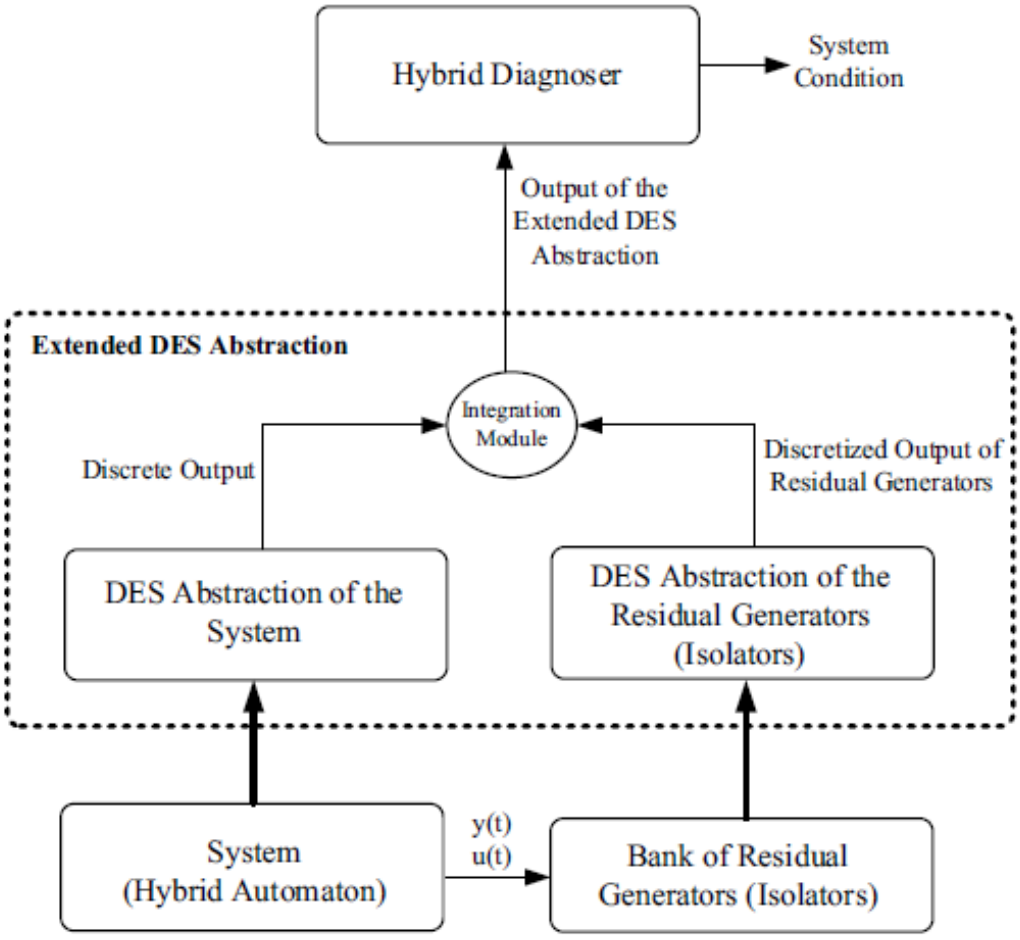


Figure 2.1: Mohammadi's Hybrid Diagnoser (Mohammadi, 2009)

this view of the system as the EDESA or the Extended Discrete Event System Abstraction. Much of the work contained within his thesis focuses on the process of transforming the hybrid state automata into this EDESA.

Obtaining the EDESA

Figure 2.1 shows a schematic of the Hybrid Diagnoser. The system is assumed to have two types of sensors: discrete sensors, whose output changes only at certain

thresholds, and continuous sensors. The strength of the Hybrid Diagnoser is the ability to merge the information that each provides.

The "System" will feed a copy of its inputs $u(t)$ and outputs $y(t)$ to the bank of Residual Generators/Isolators. These Isolators will be designed to monitor the system's continuous dynamics in search of unusual behaviour. Integration of these isolators, with the system and its discrete outputs, is done by obtaining the Discrete Event System (DES) abstraction of each.

The DES abstraction of each isolator will also have its outputs discretized by comparing its residual against a threshold value to see if it has been exceeded. The DES abstraction of the system is essentially the hybrid automata with the continuous dynamics removed from each state. When the DES representation of both the system and the Isolators have been obtained, they are merged together using the synchronous operator to create the Extended DES abstraction or EDESA.

The Residual Generators

The continuous fault diagnosis in (Mohammadi, 2009) is carried out by the bank of Residual Generator/Isolators. A residual generator uses a mathematical model of the system, along with the system's inputs, to calculate an expected value for the system's output. This value will then be compared to the actual output of the system, as measured by the sensors. The result of this comparison will be a residual that should be close to zero under normal conditions. Since faults in a linear system can be modelled as additive[1] (Mohammadi, 2009), the residual will take on a non-zero value when one of these faults has occurred.

[1]In the statespace notation an additive fault is treated as an unknown input to the system. If a fault is active, then the term f_i in $x(k+1) = Ax(k) + Bu(k) + f_i$, would be non zero (Mohammadi, 2009)

As an example, consider a fuel delivery system that is suffering from a fault that is causing a 10% fuel loss on its way to the engine. The amount of fuel that the engine would receive in this situation would be 10% less than what the system controlling it thinks it has received. This discrepancy will manifest itself in the fact that the vehicle will be travelling slower than expected. A residual generator designed for this system would use the model of the vehicle along with the throttle opening to determine the expected velocity of the vehicle. When compared to the sensor value, this expected value will reveal that the vehicle is not responding to the input in the expected manner.

However, there is another use for the residual generators in the Hybrid Diagnoser. Recall that in a hybrid system, the continuous dynamics are dependent on the discrete state and that each state in the hybrid automata will contain its own mathematical model of the system's dynamics. Since the residual generator operates from a specific model of the system's dynamics, a residual generator designed for state A would have a non-zero residual when state B is active. Therefore, in the bank of residual generators, each residual generator will correspond to a specific state and the continuous dynamics that occur when that state is active.

While this does make the hybrid diagnoser more difficult to construct, because of the additional work of designing each isolator, it does yield some advantages as well. The most significant of these advantages is that the bank of residual generators can help determine which state the system is currently operating in. If the system is in state A then only state A's residual generator should have a near-zero residual. If this is not the case either because A's isolator is actually non-zero or if any of the

other states have a near-zero residual, then a problem has been found in the system.

The Hybrid Diagnoser

As previously mentioned, obtaining the EDESA of the system and its isolators is a prerequisite to constructing the Hybrid Diagnoser. However, once it has been obtained the design process can go forward. Since the EDESA abstracts the system as a Finite State Automata (FSA), fault diagnosis techniques from the field of Discrete Event Systems (DES) can be applied at this point.

The method Mohammadi makes use of is a state based method discussed in (Zad et al., 2003). The diagnoser in this method is designed with knowledge of the discrete states in the FSA. Required is the knowledge of what the discrete output will be in each state and the possible states it can transition to. As the system operates the sequence of discrete outputs is recorded. This information is used to identify which state the system is currently in and how it got there.

More importantly, since each state can only transition to a certain subset of the total number of states, only a certain subset of the possible sequences of discrete outputs will correspond to a working system. If the sequence that is observed does not fall into this subset, then the system has transitioned to a fault state. Admittedly this is a rather simplistic explanation of this diagnoser but a more in depth description of this method can be found in (Zad et al., 2003) since such a discussion would be out of the scope of this thesis.

2.3 Thesis Contributions

The contributions of this thesis are as follows

- The development of a Hybrid Model of Adaptive Cruise Control using StateFlow & Simulink and the use of this model to generate fixed point code for an ARM-based implementation on board the RC car
- The derivation of a mathematical model of the RC car for use in simulation, testing and the creation of fault diagnosis techniques
- The application of the STPA hazard analysis technique to the ACC system to derive a list of potential faults and component-level safety constraints

Chapter 3

Introduction to Adaptive Cruise Control and System Design

In this chapter the design of the Hybrid Systems Adaptive Cruise Controller will be discussed. This discussion begins with a list of definitions and acronyms that will be used throughout the thesis. Following that, the notation used to describe the current traffic situation of the ACC controlled vehicle will be introduced.

The second half of this chapter begins by listing the requirements and safety constraints that were used to build the system. The design of the system is then discussed in two stages; the high level design is presented along with a description of the architecture used to create the controller. Finally, two designs of Hybrid ACC are presented. The first design uses a Finite State Machine to switch between a set of PID controllers. The second design improves upon the first by introducing more sophisticated LQR-based MIMO controllers which can be used to simplify the design.

3.1 Definitions

Progress $P_i(t)$: The progress of each vehicle is defined as the total distance it has travelled from some arbitrary but common starting point.
The progress of the i^{th} vehicle can be represented by $P_i(t)$.
Note: If i is not explicitly denoted, then it is referring to the subject vehicle.

Headway $h(t)$: The headway of two vehicles A and B is defined as the time it would take for A to reach B based on their current velocities. Put another way, the headway of A and B is the period of time h it would take so that $P_A(t+h) = P_B(t)$, assuming that the velocity of each vehicle is held constant through the interval h.
The headway of the i^{th} vehicle and its leader is denoted $h_i(t)$.
Note:*If i is not explicitly denoted, then it is referring to the subject vehicle.*

Subject Vehicle: Also referred to as **Self** or the **Host Vehicle**.
The Subject vehicle is the vehicle that is equipped with the ACC system along with its supporting hardware. This vehicle's longitudinal dynamics will be controlled by the ACC system for as long as the ACC system is activated.

Lead Vehicle: Also referred to as the **Leader**.
The vehicle that is currently ahead of the subject vehicle in the lane that the subject vehicle is operating in.

$$LV \coloneqq vehicle(i) | \min_{i=1:n} (P_i(t) - P_s(t)) \quad \wedge \; P_i(t) > P_s(t))$$

Where $P_s(t)$ represents the progress of the Subject Vehicle

IVS: **Inter-Vehicular Sensor**

The term IVS shall be used to refer to the sensor that provides information on the inter-vehicular states *range* and *range-rate* also known as d_{rel} and v_{rel} respectively. This category includes sensors like radar, lidar, ladar and ultra-sonic sensors

VSS: **Vehicle Speed Sensor**

The term VSS shall be used to refer to the sensor that provides the information about the host vehicle's velocity. This category includes Hall Effect sensors and optical encoders.

ABC: **Active Brake Control**

The subsystem within ACC that when activated is responsible for controlling the subject vehicle's braking behaviour.

CFS: **Car Following System**

The subsystem within ACC that when activated is responsible for controlling the subject vehicle's dynamic following behaviour.

TCC: **Traditional Cruise Control**

The subsystem within ACC that when activated is responsible for controlling the subject vehicle's cruising behaviour.

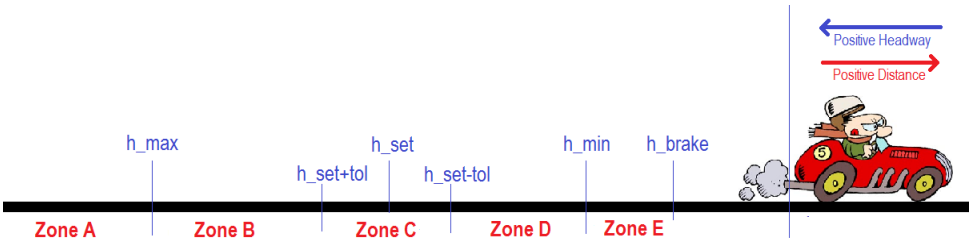


Figure 3.1: Diagram of Headway Zones

3.2 Problem Space and Notation

Adaptive Cruise Control is a driver assistance system responsible for controlling the vehicle's longitudinal behaviour on behalf of the driver. The functional requirements for ACC specifies two operating modes. The first operating mode is similar to what would be found in a more traditional cruise control system. In this operating mode the ACC system will control the vehicle's throttle so as to maintain a cruising velocity requested by the driver. However, if the ACC controlled vehicle is approaching a slower moving vehicle then the second operating mode will be used. In this mode the ACC controlled vehicle will slow down and begin following the lead vehicle by maintaining a gap whose size is chosen by the driver.

Since the behaviour of the vehicle will be influenced by its proximity to the leader vehicle, a system was devised for categorizing its current following distance. Figure 3.1 gives a visual representation of this system while a more in-depth explanation of each "Zone" can be found in section 3.2.2. The legend in the top right corner of the diagram indicates that the vehicles are moving in the positive direction on the x-axis. It also shows that the value of "Headway" will get smaller as the host vehicle gets closer to the lead vehicle. So a vehicle that is operating in Zone A will have a much larger headway than one operating in Zone E.

3.2.1 Symbols

$\mathbf{h}^*, \mathbf{h}_{self}$ (s): The current headway between the subject and leader vehicles

$\mathbf{h}_{brake}$ (s): The threshold which specifies the minimum following time needed to brake if both vehicles were to apply max braking power at the exact same time

$\mathbf{h}_{safe}$ (s): The threshold headway which specifies the minimum following time needed to:

1. Recognize the lead vehicle is decelerating (Sensing)
2. Make a decision on what to do and do it (Classification and Communication Delays)
3. Come to a stop without colliding with the lead vehicle (Actuation)

As a result $h_{safe} > h_{brake}$

Note: The radar system's sensing range must be greater than both $h_{safe} \times V_{self}$ and $h_{max} \times V_{self}$

$\mathbf{h}_{min}$ (s): The threshold that specifies the smallest value for headway in which the system should attempt to maintain normal operation of the vehicle without utilizing the Active Braking Control. For the system to be able to operate safely the condition $h_{min} \leq h_{safe}$ must remain true.

$\mathbf{h}_{max}$ (s): The threshold beyond which the system should behave as a traditional cruise control system and no longer attempt to meet the objective of keeping $e_{track} \leq tol_{track}$

The factors that must be considered when choosing this value include, but are not limited to: the upper bound of the range capabilities

of the radar system to provide reliable sensor data and driver car following behaviour characteristics

$\mathbf{h}_{set}$ (s): A value set by the driver that specifies the desired following time. This value may be chosen from a range of values set by the manufacturer to ensure that $h_{setmin} > h_{min}$ and $h_{max} > h_{setmax}$. When in "following mode" this value will become the desired value for h^*

$\mathbf{h}_{setmax}$ (s): The upper bound on the headway region. It is the maximum headway value for which the system's tracking error will be considered acceptable.

- Acceptable here means that $e_{track} \leq tol_{track}$, or equivalently $h_{setmax} \leq h_{set} + tol_{tracking}$

$\mathbf{h}_{setmin}$ (s): The lower bound on the headway region explained above. However, here acceptable means that $e_{track} \leq tol_{track}$, or equivalently $h_{setmax} \leq h_{set} - tol_{tracking}$

$\mathbf{V}_{self}$ $(^{km}/_{h})$: The current velocity of the Subject vehicle

$\mathbf{V}_{lead}$ $(^{km}/_{h})$: The current velocity of the Leader vehicle

Note: This value cannot be directly obtained since there is no direct communication between the vehicles with regard to vehicle states. Instead it must be estimated using the Radar sensor while taking into account the velocity of the vehicle with the Radar sensor.

$\mathbf{V}_{set}$ $(^{km}/_{h})$: A value set by the driver that specifies the desired cruising velocity. The system will attempt to minimize the difference between V_{self} and

V_{set} unless there is a slower moving leader vehicle less than h_{max} seconds ahead

e$_{pace}$ ($^{km}/_{h}$): The difference between the desired velocity, as specified by V_{set}, and the actual velocity of the subject vehicle V_{self} . Consequently, $e_{pace} = V_{set} - V_{self}$.

Note: This value is only relevant when "cruising".

e$_{pace_{rel}}$ ($^{km}/_{h}$): The difference between the velocity of the leader vehicle V_{lead} and that of the subject vehicle V_{self} . Consequently, $e_{pace_{rel}} = V_{lead} - V_{self}$.

Note: This value is only relevant when "following".

e$_{track}$ (s): The difference between the desired headway, as specified by h_{set}, and the actual headway of the system h^*. Consequently, $e_{track} = h_{set} - h^*$.

Note: This value is only relevant when "following".

tol$_{pace}$ ($^{km}/_{h}$): A predefined value that specifies the maximum value for e_{pace} beyond which the system is not obligated to further reduce e_{pace}.

tol$_{set}$ (s): A predefined value that specifies the maximum value for e_{track} beyond which the system is not obligated to <u>further</u> reduce e_{track}

tol$_{track}$ (s): A predefined value that specifies the maximum value for e_{track} beyond which the system is not obligated to <u>begin</u>[1] reducing e_{track}

Note that $h_{set} + tol_{track} = h_{setmax} \leq h_{max}$

and $h_{set} - tol_{track} = h_{setmin} \leq h_{min} \leq h_{safe}$

[1]The difference between tol_{track} and tol_{set} will become more apparent in the section about defining the state transitions, but in general tol_{set} is concerned with specifying when the activity of reducing the error can be ceased, while tol_{track} specifies when this activity should be resumed. So the difference between the two is a hysteresis region.

3.2.2 Explanation of Headway Zones

The behaviour of the ACC controlled vehicle depends on how close it is to the leader vehicle. While the system is running it will use its Radar sensor to measure the distance to the leader. It will then use this information to categorize its current situation into one of 5 "Zones". Table 3.1 gives a mathematical definition of each zone.

Definition of Zones	
Zone A	$h^* \geq h_{max}$
Zone B	$h_{max} < h^* < (h_{set} + tol_{set})$
Zone C	$(h_{set} + tol_{set}) \leq h^* \leq (h_{set} - tol_{set})$
Zone D	$(h_{set} - tol_{set}) < h^* < h_{min}$
Zone E	$h^* < h_{brake}$
Note:	$h_{brake} \leq h_{min}$

Table 3.1: Definition of Headway Zones

Zone A Subject's leader is either beyond the reliable range of the IVS, or it beyond the range at which following behaviour would be considered desirable

Zone B Subject vehicle is within the range at which following behaviour should be considered but still has a larger headway than the user requested

Zone C Subject vehicle is within the desired range of following times

Zone D Subject vehicle's headway is less than desired but still within the safe range or it less than the safe range but in a situation where Active Brake Control should not be utilized

Zone E: Gentle Braking Region. Subject vehicle's headway is less than the safe range and Active Brake Control should be utilized but its power limited to some predefined value

3.3 Assumptions

1. It is assumed that the Subject Vehicle contains the following sensors

   (a) **VSS**: A vehicle speed sensor (eg, encoder) that can detect the current speed of the vehicle.

   (b) **Radar/IVS**: A radar system or other range detecting sensor that is capable of sensing both range and range rate.

2. It is assumed that there is no direct communication of state values back from the lead vehicle (ie this is not a CACC[2] system).

3. It is assumed that a system has been worked out to provide reliable range and range-rate information around bends and for vehicles that are only partially in the lane (provided the Radar sensor is currently operating correctly).

4. It is assumed that the velocity of all vehicles is no less than 0 (ie, they are not reversing).

5. It is assumed that the driver or some other system is responsible for steering the vehicle.

6. It is assumed that the system will never have the need to apply both the brakes and throttle at the same instance.

[2]A description of Coordinated Adaptive Cruise Control is included in the literature review

3.4 System Requirements

To be able to effectively design the Adaptive Cruise Control system, a proper understanding of the desired functionality should be obtained and recorded as a set of requirements.

This section contains the Functional Requirements that were used to design the system. Additionally, a list of safety constraints which were obtained using the STPA hazards analysis has been included. More information about STPA and its use on Hybrid ACC can be found in Chapter 7.

3.4.1 Functional Requirements

1. Safety
    (a) The driver must be able to regain control of the vehicle or override the system at any point in its operation
    (b) The space between the subject and lead vehicle must remain positive so long as they are in the same lane,
        i. The subject vehicle must leave enough space to be able to come to a complete stop if the lead vehicle were to suddenly apply full braking power
        ii. The system must be able to respond in a timely manner to rapid deceleration of the lead vehicle or to a vehicle cutting in
    (c) The system must never actively move into an unsafe following distance
        i. If a vehicle cuts in, leading to a potentially unsafe situation, the system:

A. must not take any action that will further reduce the following distance

B. must work to return the system to a safe following distance

(d) Avoid sudden braking or rapid deceleration

(e) String stability of the traffic

i. A string of at least N vehicles should be able to drive in series without the string becoming unstable

(f) The system must have a framework in place to attempt to detect when an error has occurred within the ACC system, its actuators or with one of its sensors

(g) The system must be shut down immediately and control returned to the driver if any of the following occur

i. The driver presses the brake pedal, clutch, or chooses to disable ACC from the instrumentation panel

ii. A sensor, for which there is no reliable alternative, fails

iii. A sensor fails and due to its failure it becomes impossible to verify the correctness of the other sensor(s) in the system

iv. The fault diagnosis unit detects the presence of a unrecoverable fault Any one of the following fails: CPU, Actuator, communication bus

2. Tracking accuracy

(a) When the subject vehicle approaches a slower moving lead vehicle, it must slow down to the leader's velocity and follow at some predefined time headway

(b) When the lead vehicle is not accelerating and is travelling at a velocity less than the user's desired velocity, as expressed by the velocity set point, then:

i. At every point in time the system's tracking error must be less than some predefined value tol_{track} or approaching it[3]

ii. The system must achieve a tracking error at least as small as tol_{track} within some predefined time period so long as the leader's velocity doesn't change by more than $X_1\%$ during that time period.

(c) If either of the following are true:

- there is no lead vehicle present
- the lead vehicle is moving at least as fast as the subject vehicle and the current headway exceeds h_{min}

then the subject vehicle will act as a traditional cruise control system and will attempt to minimize the difference between the velocity set point and the current velocity of the vehicle

i. The tracking error in this situation will be defined as $e_{pace} = V_{set} - V_{self}$

3. Driver comfort/Fuel economy

(a) Restrict maximum acceleration and braking power

(b) Transitions between different modes of operation should be smooth enough so that acceleration will be less than $X_{max_{accel}}$

[3]Some designs in the literature have a requirement that goes something like this: "Tracking errors should approach 0 as time approaches infinity", however this won't be included in this set of requirements due to conflict with the string stability requirement

(c) Avoid braking when it is safe to do so

(d) The driver should be able to specify the following from the instrumentation panel[4]

i. Desired velocity of the vehicle : v_{set}

ii. Desired following time : h_{set} (from a range of acceptable values)

iii. Desired following accuracy which is inversely proportional to $tol_{tracking}$ this value will be chosen from a range of acceptable values provided by the system

3.4.2 Safety Constraints

In addition to the high level functional requirements, which deal with the system at a fairly high level of abstraction, a set of safety constraints were added. These safety constraints are the result of using the STPA process on the high level design of Hybrid ACC. One of the strengths of STPA is its ability to assign responsibility for enforcing the safety requirements to individual components. As a result, these constraints deal with the system at a much lower level, mostly dealing with individual subsystems.

Full system

S0.1 While the system is in control of the vehicle, the velocity must not exceed the user-defined velocity set point by more than some pre-defined tolerance. The value of this tolerance is dependant on the mode in which the system is currently operating. If it is in 'following' mode then the tolerance will be tol_{pace}

[4]A proper HMI design would be required in the actual implementation to determine the best way to provide the driver with an interface that won't distract them from their task.

and if it is in 'cruise' mode then it will be $tol_{pace_{rel}}$

S0.2 If approaching a vehicle that is moving slower than the user defined velocity then the system must slow down and begin to maintain the user-specified following distance

S0.3 The system must reduce throttle in the presence of a leader vehicle that is less than h_{setMax} away if the leader vehicle's velocity is greater than the set point velocity

S0.4 Hysteresis regions must be added to the switching conditions to avoid rapid switching and system instability due to delays

Braking System

To utilize the system's brakes two things must happen: the Active Brake Controller must first be activated then it must be used to activate the brakes.

Brakes

S1.1 The system must activate the brakes when the gap between the vehicle and its leader is less than the safe following distance

S1.2 The vehicle must not brake unless the gap is less than the safe following distance

S1.3 Brake power must be limited to $X_{max_{brake}}\%$

S1.4 The brake lights must be on when the vehicle is braking

Active Brake Controller Subsystem

S2.1 The ABC Controller must be activated if the gap is less than the safe following distance

S2.2 The ABC Controller must not be activated unless the gap is less than the safe following distance

Car Following Subsystem

S3.1 Once activated the ABC controller must remain active until the system has restored a safe following distance or the user deactivates or suspends the system

S3.2 The system must not attempt to follow a vehicle whose velocity is greater than the user defined set point velocity

S3.3 The system must not attempt to follow a vehicle whose headway from the host vehicle is greater than h_{max} aka Zone A

S3.4 Once activated the CFS must remain active unless

- the gap between the host vehicle and its leader is less than the safe following distance
- the leader vehicle's velocity becomes greater than the set point velocity
- the distance between the velocity of the host vehicle and its leader becomes greater than h_{max} or the user deactivates/suspends the system

Traditional Cruise Control Subsystem

S4.1 In the absence of a suitable leader vehicle to follow, the system must attempt to maintain the user-defined velocity without exceeding it by more than tol_{pace}

S4.2 Once activated, the TCC must remain active unless the system approaches a slower moving vehicle (Zone C or closer) or the user deactivates or suspends the system

Interaction with Driver

S5.1 The system must not activate or resume unless commanded to by the driver

S5.2 The system must return control to the driver upon receiving the deactivation signal/suspend signal or after sensing the user's brakes

a. This must occur within X milliseconds from when the signal is sent

S5.3 The system must indicate when it is active

S5.4 The system must indicate if CC_{only} is active

S5.5 The system must indicate when "Initiation" is complete and the vehicle is under ACC's control

S5.6 Once activated the system must remain active unless commanded by the driver to "deactivate" or "suspend" or if any of the events identified in requirement 1(g) occurs

S5.7 The system MUST respond to the user's acceleration and braking commands (User command actions override the system's commands)

3.5 High Level Design of a Car Following System

If the ACC controlled vehicle doesn't currently have a slower moving leader vehicle then its desired velocity is simply V_{set}.

However, if the vehicle is approaching a slower moving vehicle, then determining the desired velocity of the system becomes more complicated. Figure 3.2 shows a plot of a function $f : e_{gap} \rightarrow V_{rel}$ which describes the desired velocity of the vehicle in a "following" situation.

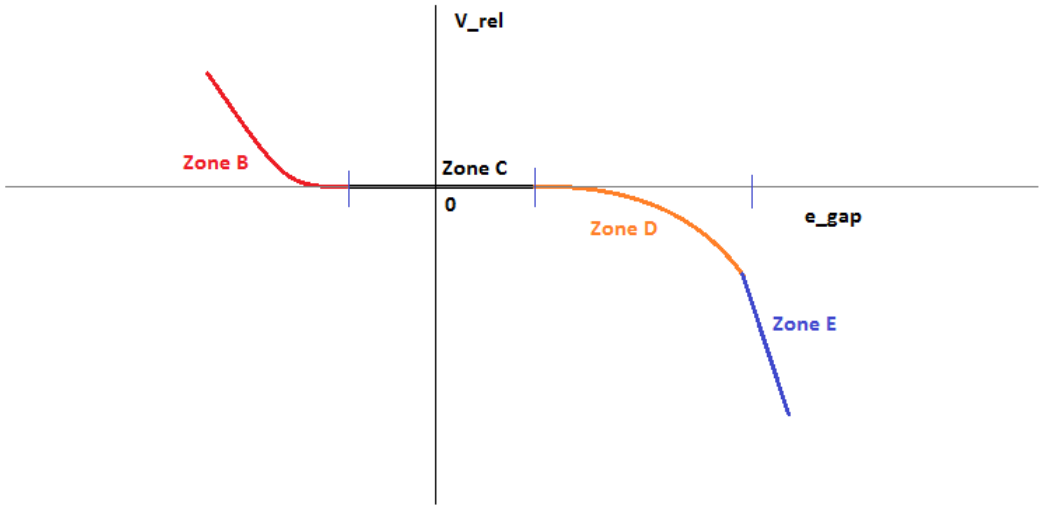


Figure 3.2: Desired V_{rel} as a function of e_{gap}

When following a leader vehicle the system's continuous dynamics can be described by a function $f : e_{gap} \rightarrow V_{rel}$ which has the following properties:

1. Piecewise continuous (smooth transitions)
2. $|\frac{\partial f}{\partial e_{gap}}| < A$, where A = max allowable acceleration,
    - i.e. Magnitude of the Slope in of each segment must be less than A

3. Relative velocity should be a monotonically decreasing function of the quantity $e_{gap} = h_{set} - h^*$. As a result the following statements must hold true so long as the system is in following mode

$$\left(e_{gap} > tol_{tracking} \rightarrow \frac{\partial f}{\partial e_{gap}} < 0\right) \tag{3.1}$$

$$\left(e_{\text{gap}} < -tol_{tracking} \rightarrow \frac{\partial f}{\partial e_{gap}} < 0\right) \tag{3.2}$$

   i.e. if $e_{gap} \geq tol_{tracking}$ the slope must be negative and such that e_{gap} will decrease over time

Notes:

1. The slope of the curve at the end of Zone D represents the deceleration the system would undergo when the throttle is set to 0 and the brakes are not applied

2. The slope of the curve in Zone E represents the braking power (assuming a linear model for the brakes)

3.6 Design of an Adaptive Cruise Controller

This section will highlight the workings of the ACC system that was designed in response to the requirements listed above. Implicit in the functional requirements was the need for the system's dynamics to adapt to the current situation. If no lead vehicle is present, or if the leader is travelling faster than the set point velocity, then the system must act as a traditional cruise control system. However, if approaching

a slower moving vehicle then the system must switch to following at a safe distance while matching the lead vehicle's velocity. Finally, the ACC system must issue the brake command if the lead vehicle gets too close.

The use of a hybrid automata will allow for a controller to be designed that will be able to handle the need for different operating modes without neglecting the need for continuous control over the vehicle's dynamics.

The resulting system consists of three subsystems and a data flow between them as depicted in the block diagram of Figure 3.3.

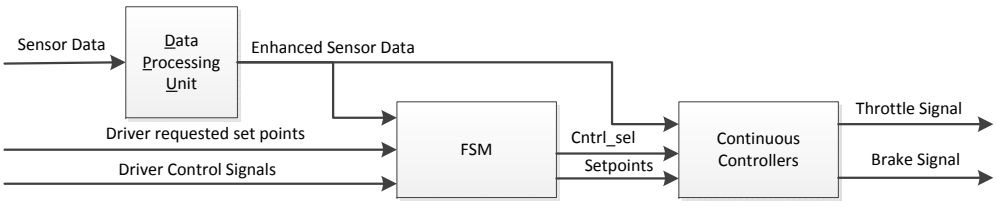


Figure 3.3: The Architecture of the Adaptive Cruise Control System

The first of these subsystems is the Data Processing Unit or DPU. This subsystem plays a supporting role to the other subsystems by providing each with sensor data that has undergone the necessary filtering and conditioning.

Together, the other two components make up the hybrid automata. The Finite State Machine (FSM) makes the discrete control decisions using the sensor data and the set points chosen by the driver. The FSM itself is a (discrete) automata consisting of several states, each of which corresponds to one of the system's operating modes.

The continuous portion of the hybrid automata is implemented by a cluster of continuous controllers, which shall be called the CCC. Each controller within this

bank has been designed to implement the desired dynamics for a specific mode. The goals and design of each of these controllers will be discussed in more detail later.

The decision as to which of the continuous controllers should be activated is taken care of by the FSM. The FSM will also supply the active continuous controller with the set points it is responsible for achieving.

Driver Interface

The driver interacts with the system through the instrumentation panel. The instrumentation panel provides the driver with feedback about the system's current state as well as the opportunity to issue commands like "Activate ACC" or "Suspend".

The driver is also able to issue a command to activate only the subset of ACC which implements traditional cruise control. If this is chosen then the system will attempt to match the vehicle's velocity with the user's requested velocity *even if it is approaching a slower moving vehicle*. It is important to note that in this situation it is up to the driver to manage the gap with the other vehicle and to operate the brakes if necessary[5].

The user interface also allows the user to increase or decrease the velocity and headway set point values. By adjusting these values, the driver can control what cruising velocity the system will attempt to achieve and how much of a gap it should leave if a slower moving vehicle prevents this goal from being realized.

If the driver chooses to manually operate the throttle through the gas pedal, the

[5]It would be beneficial to consider different ways the system could alert the driver to the need for extra braking power. Alternatively, ACC could be paired with a pre-crash braking system which could be given full control of the brakes when the vehicles get too close

system will be temporarily suspended until the driver removes their foot from the gas pedal. If, on the other hand, the driver presses the brake pedal, then the system will be suspended until the driver chooses to manually re-activate the system by pressing the corresponding button on the user interface.

3.6.1 An Initial Design for the Finite State Machine

As a first attempt the FSM was designed to work with a bank of PID[6] controllers. In this scheme each controller would have a single "goal" and would map to exactly one state. An explanation of the FSM that was designed to control this system in included below.

System States

The original FSM consisted of four states to represent the system when it was active and two states for when ACC was inactive.

A description of these states is included below and a graphical representation of this state machine can be seen in Figure 3.4.

1. **ACC Off:** The ACC system is off and will not attempt to control the vehicle or to gain information about current conditions via its sensors.

2. **ACC Initialize:** The user has initiated the ACC system. The system will begin to use its sensors to gather information about the current scenario. This information will be used to choose the appropriate state to transition into. Control of the actuators has not yet commenced.

[6]PID stands for Proportional-Integral-Derivative and is a popular form for simple single-input single-output controllers

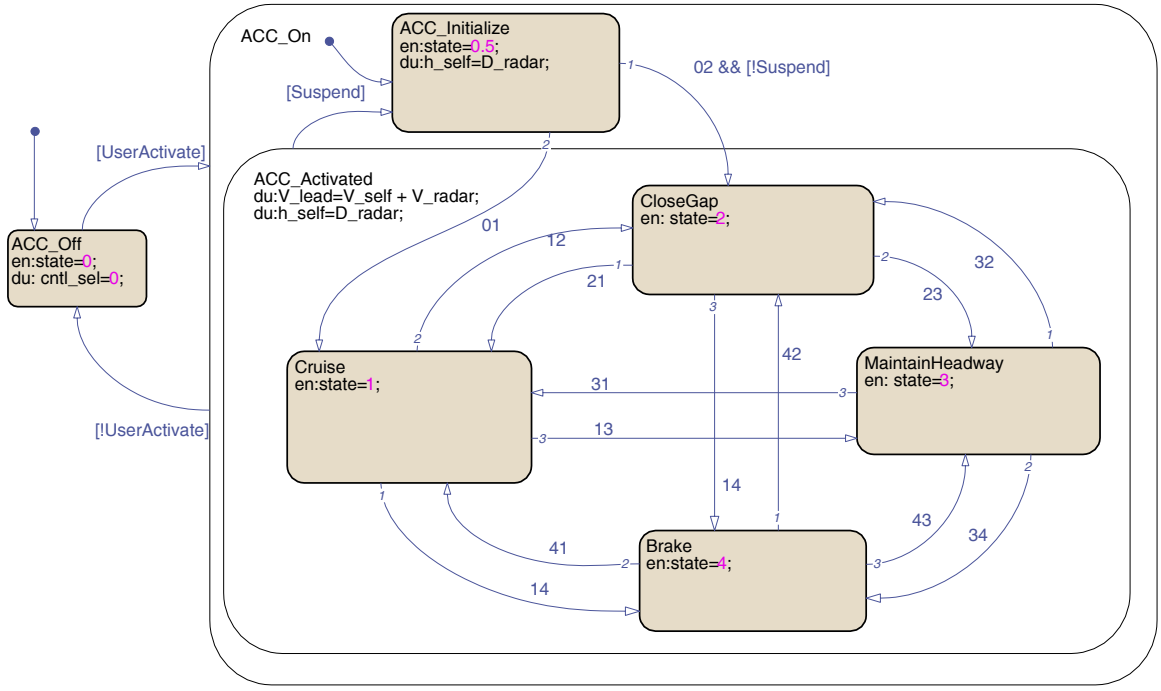


Figure 3.4: StateFlow Diagram of State Automata

3. **Cruise:** One of the four active states. While this state is active the system will attempt to minimize the difference between the velocity requested by the user and the actual velocity of the system.

4. **Close Gap**: While this state is active the system will attempt to minimize the difference between the current headway of the system and that requested by the driver.

5. **Maintain Headway**: While this state is active the system will attempt to match the velocity of the leading vehicle so as to maintain a constant headway.

6. **Apply Brakes:** While this state is active the system will apply the brakes until a safe gap has been restored. While in this state the system will attempt to minimize the difference between the desired headway and actual headway.

It should be noted that only one of the active states (Apply Brakes) will result in any brake action at all. The other three active states rely on manipulating the throttle signal to achieve their goals. A summary of each state is included in Table 3.2. The table shows each state's name along with the control variable it will attempt to minimize, the sensors that will provide it with data and the actuators that will carry out its commands.

State Name	Control Variable	Sensor Used	Actuator
ACC Off	-	-	-
ACC Initialize	-	VSS, Radar	-
Cruise	$e_{pace} = V_{set} - V_{self}$	VSS	Engine
Close Gap	$e_{track} = h_{set} - h^*$	VSS, Radar	Engine
Maintain Headway	$e_{pace_{rel}} = V_{lead} - V_{self}$	Radar	Engine
Active Brake Control	$e_{track} = h_{set} - h^*$	VSS, Radar	Brakes

Table 3.2: State Information

State Transitions

Equally important to the design of the individual states are the transitions between them and the associated guard conditions. The diagram in Figure 3.4 depicts these transitions as directed edges which have been labelled with a two digit number.

In the actual implementation, this number would be replaced with the guard

	Zone A	Zone B	Zone C	Zone D	Zone E
$V_{Lead} \geq V_{Set} + tol_{hyst}$	Cruise	Cruise	Cruise	CloseGap	Brake
$\|V_{Lead} - V_{Set}\| \leq tol_{hyst}$	Cruise	Cruise/CloseGap	Cruise/Headway	CloseGap	Brake
$V_{Lead} < V_{Set} - tol_{hyst}$	Cruise	CloseGap	Headway	CloseGap	Brake

Table 3.3: Transition Criteria for each for ACC with PID

condition for that transition. For now these labels will provide a way of referencing their corresponding transition.

Using the functional requirements, Table 3.3 was created to indicate what the system's next state should be based on current conditions. The flow chart in Figure 3.5 depicts the reasoning which led to Table 3.3. This flow chart depiction is useful here since it shows the hierarchy of decision criteria.

The transition criteria that result from this analysis can be summarized as follows.

1. Zone A[7] always leads to "Cruise"
2. Zone E[8] always leads to the state "CloseGap" to slow the vehicle down.
3. So long as the velocity of the leader is *at least as fast* as the set point velocity and the vehicle is not in Zone E, then the next state will be Cruise
4. So long as the velocity of the leader is *slower* than our set point velocity
   (a) Zone D [9] will lead to CloseGap
   (b) Zone B[10] will lead to CloseGap, unless it is transitioning from the state Cruise and is currently in the hysteresis region

[7] $h^* \geq h_{max}$
[8] $h^* < h_{max}$
[9] $(h_{set} - tol_{set}) < h^* < h_{min}$
[10] $h_{max} < h^* < (h_{set} + tol_{set})$

(c) Zone C [11] will lead to Headway, unless it is transitioning from the state Initialize

The transitions will be driven by a clock and guarded by the conditions expressed by Table 3.3.

Note: This is a slight variation of the hybrid automata that was used by Mohammadi in his thesis (Mohammadi, 2009). The main difference being that in Mohammadi's version, the guard conditions were evaluated to decide if a transition was valid. However, even if the transition was found to be valid, it would only be taken when the state invariant became false

[11] $(h_{set} + tol_{set}) \leq h^* \leq (h_{set} - tol_{set})$

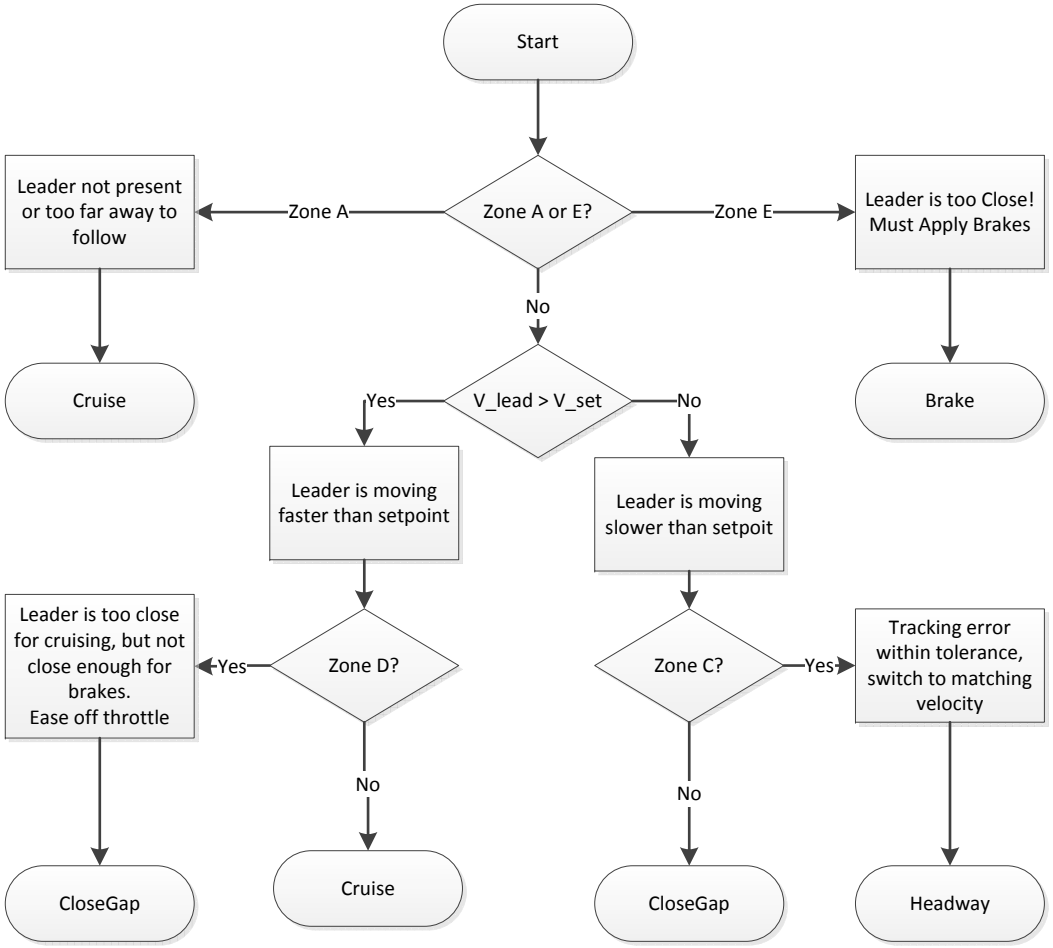


Figure 3.5: Decision process for choosing state transitions

3.6.2 An Improved Finite State Machine

The FSM discussed in Section 3.6.1 was designed for a system which had the limitation of using Single-Input-Single-Output (SISO) controllers. This limitation resulted in a more complicated following mode since the system would have to switch between tracking the distance to the leader and their velocity.

However, Multiple-Input-Multiple-Output (MIMO) controllers like the LQR can be designed to track both of these quantities at the same time. The benefit of this can be seen when one considers the FSM that would be needed to control such a system. While the original FSM had to switch back and forth between two states to ensure accurate following, the FSM for this MIMO-enabled system would be in the same state for as long as the following continues.

System States

The FSM for the LQR-controlled system shares most of its states with the PID-controlled system. In fact the only difference between these two FSMs is that the LQR-FSM merges the states **Close Gap** and **Cruise** into a single state **Follow**. The description of follow can be seen below.

Follow: While this state is active the system will attempt to minimize the difference between the current headway and that requested by the driver *while* concurrently attempting to match the leader's velocity

State transitions

The move to LQR controllers also has an effect on the transitions between states. For starters, by reducing the number of states a reduction in the number of transitions is

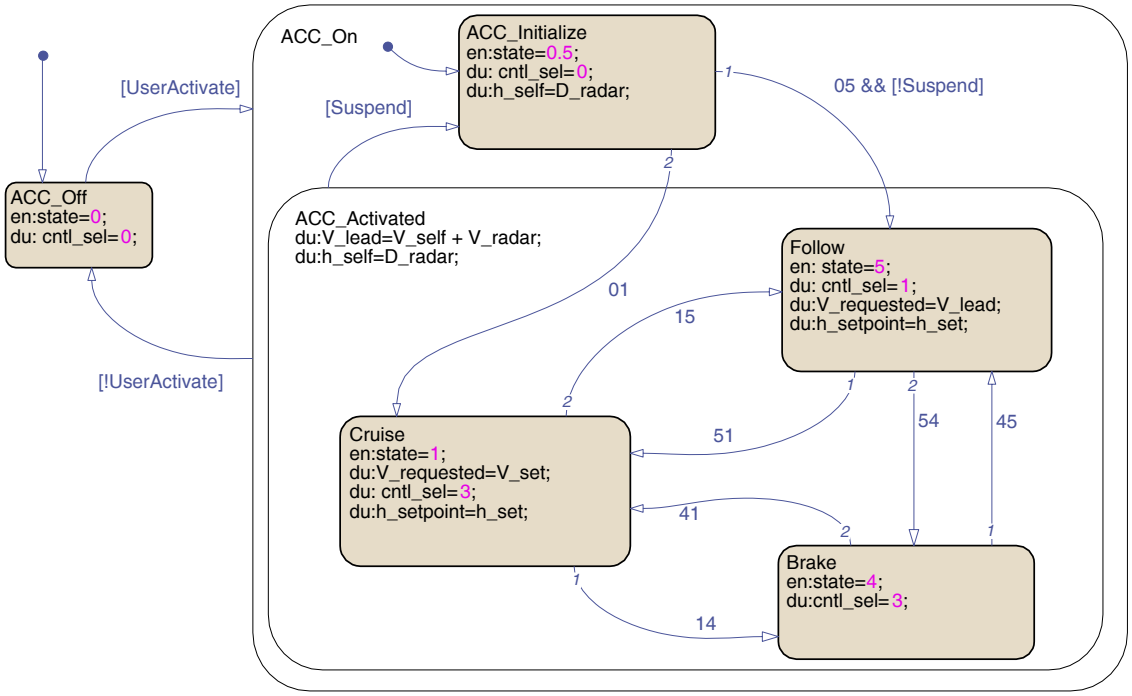


Figure 3.6: The FSM designed to control the LQR controllers

	Zone A	**Zone B**	**Zone C**	**Zone D**	**Zone E**
$V_{Lead} \geq V_{Set} + tol_{hyst}$	Cruise	Cruise	Cruise	Follow	Brake
$\lvert V_{Lead} - V_{Set} \rvert \leq tol_{hyst}$	Cruise	Cruise/Follow	Cruise/Follow	Follow	Brake
$V_{Lead} < V_{Set} - tol_{hyst}$	Cruise	Follow	Follow	Follow	Brake

Table 3.4: Transition conditions for ACC with LQR

also realized. Secondly, since this reduction in the number of states was obtained by merging two existing states, the guard conditions on the transitions to the new states will be simpler. Before the FSM had to determine not only if the system should be following but also which quantity it should begin tracking first. In the new system, the continuous LQR controller will work to achieve both goals simultaneously.

Table 3.4 shows the guard conditions for the new system which is depicted in Figure 3.6

The LQR controllers

Information on the LQR controller and its use in this Adaptive Cruise Control System can be found in Chapter 6.

3.7 Summary

In this chapter notation was given to make use of different 'Zones' to categorize the following situation of the ACC vehicle. This notation made it easier to communicate the design and requirements.

The system requirements were also given along with the safety constraints that are derived using STPA in Chapter 7. Finally, the design of the system was discussed. Included in this discussion was the use of LQR controllers from Chapter 6, to simplify the design. Chapter 4 will describe the realization of this design in Simulink and

Stateflow.

Chapter 4

Simulink Model

4.1 Introduction

To be able to simulate the operation of the ACC system a model was created in Matlab's Simulink. Included in this model is an implementation of the ACC system as well as the mathematical model of the RC car that will be developed in Chapter 5. The top level of this Simulink model consists of 6 blocks or subsystems and the signals that flow between them. Four of these blocks represent components of the ACC system and the other two represent the host vehicle and the leader vehicle respectively.

4.2 The Signals

Before diving into each of the subsystems and the role they fulfil, it is helpful to first gain an understanding of the signals that are passed between them as inputs and outputs.

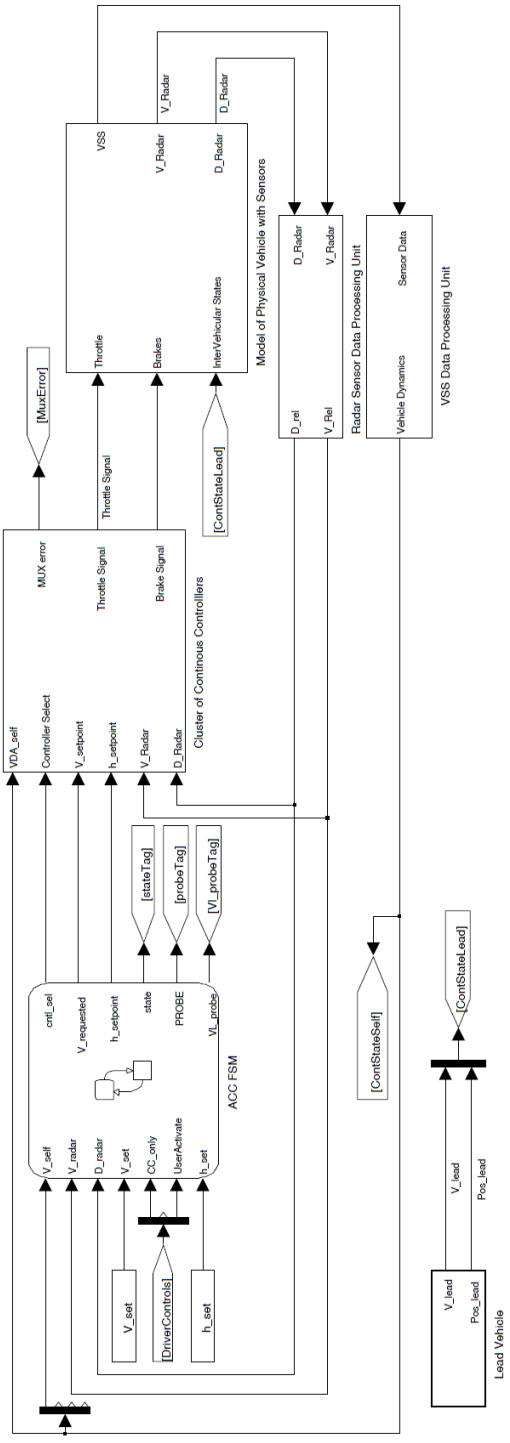


Figure 4.1: Top level view of ACC Simulink Model

V_{set} ($^{km}/_{h}$): This signal represents the driver's desired cruising velocity. The system will attempt to achieve this velocity so long as there is not a slower moving vehicle in its path.

h_{set} (s): This signal represents the driver's desired following distance for situations in which a slower moving leader vehicle is preventing the ACC system from being able to achieve V_{set}

$V_{requested}$ ($^{km}/_{h}$): While V_{set} represents the velocity that the driver wants the system to cruise at, $V_{requested}$ is the velocity that the FSM decides the system should attempt to achieve based on the current conditions.

$h_{requested}(s)$: Similarly $h_{requested}$ is the following distance that the FSM system decides the system should attain based on the current conditions.

Throttle Signal (% $\times$ 10): This signal is passed from the ACC controller to the vehicle to operate the throttle. The value is a number in the range 0-1000 where 1000 is full throttle. Effectively then the units for this signal become %Throttle $\times$ 10.

VSS ($^{km}/_{h}$): This signal represents the output of the Vehicle Speed Sensor (VSS) and gives a reading of the vehicle's velocity. *Alternatively the units on this could be converted to $^{m}/_{s}$ by the DPU*

V_{self} ($^{km}/_{h}$): This signal represents the system's measurement of the

	host vehicle's velocity after it has undergone signal processing.
V_{Radar} ($^{km}/_{h}$):	This signal represents the velocity reading taken from the radar mounted on the host vehicle. As a result the value contained in this signal will be the relative velocity between the lead vehicle and the Radar's own velocity. Thus the leader's velocity can be estimated as $V_{self}+V_{Radar}$
V_{rel} ($^{km}/_{h}$):	This signal represents V_{Radar} after it has undergone signal processing (See Section 4.3) This value is used by the system to estimate the leader's velocity by the relation $V_{leader} = V_{self} + V_{rel}$.
D_{Radar} (s):	This signal represents the Radar sensor's measurement of the distance between the host vehicle and the leader vehicle.
D_{rel} (s):	This signal represents D_{Radar} after it has undergone signal processing.
VDA_{self}:	This signal is very important as it represents the system's estimate of the vehicle's continuous state. This signal is a 4-tuple with the units of ($^{km}/_{h}$, -,s,$^{km}/_{h^2}$).(See Section 4.3)
Inter-vehicle states:	This signal is used only in simulation as a way to provide the simulated leader vehicle's continuous state (For more information see the description of the lead vehicle in Section 4.3.1)

4.3 The Subsystems

As previously mentioned, the blocks included in the top-level view of the Simulink Model can be partitioned into three categories:

1. Components of the ACC controller
2. Blocks that represent physical systems that are being simulated
3. Scopes and tags that are used to study the behaviour of the system

In the following section the blocks in the first two categories will be explored.

1. Components of the ACC Controller

The architecture of the ACC Controller can be viewed as consisting of four components. Sensor data that is fed into the ACC system is first passed to one of the two Data Processing Units. From there it is passed on to the FSM and the Cluster of Continuous Controllers (CCC). The FSM also provides instructions to the CCC with regard to which controller to use and what set points are to be achieved.

Block: *ACC Discrete*

The block used to implement *ACC Discrete* is designed using the Stateflow toolbox which allows for the creation of FSMs in the Simulink environment. The *ACC Discrete* block implements the discrete control decisions that are made by the controller and the discrete states or modes that arise out of this.

Inputs Each of the inputs to this block can be classified as either feedback data or as a driver control signal. In the first category are the signals V_{self}, V_{Radar}, and D_{Radar}, which provide

the block with information about the current state of the system. These signals are provided by the Data Processing Units.

The second category of inputs (the driver control signals) consist of set points like V_{set} and h_{set} as well as boolean signals which allow the driver to make decisions about the behaviour of the system. Included in this is the ability of the driver to turn the system on or off, to put it in *Cruise Control Only* mode or to temporarily suspend operation.

Outputs There are 3 outputs from this block all of which are fed to the CCC. The first of these outputs is *cntl_sel* which tells the CCC which of its internal controllers should currently be active. The other two outputs are the signals $V_{requested}$ and $h_{requested}$, which are the set points that the CCC is responsible for achieving.

Internal Operation The internal operation of this subsystem is driven by a discrete automata. This automata is made up of discrete states/modes and the transitions between these modes which are guarded by logical statements. The transitions and their guard conditions are used to implement the discrete control decisions, e.g. when to switch from cruising to following. The discrete states are used to realize the hybrid

modes that were identified during design of the ACC algorithm (Following, Cruising, ACC Off, etc). Each discrete state influences the behaviour of the overall system by its choice of controllers and set points (both of which are passed to the CCC).

Example of operation If the system is too far from any lead vehicle then it will begin to act like a traditional cruise control system (ie it will try to minimize the error between the vehicle's velocity and the set point velocity.) So in this case *cntl_sel* will be set so that the Cruise System controller (inside the CCC) will be allowed to pass its output through the MUX and therefore to be the controller that is currently active.

Block: *Cluster of Continuous Controllers*

This block/subsystem implements the continuous control portion of the hybrid controller. This is done by providing three Linear Quadratic Regulators (LQR) and the ability of the *ACC Discrete* subsystem to decide which of them will generate the output. Each of these LQR controllers has been designed to meet the control goals of its respective state. More information about the LQR controllers and their design can be found in Chapter 6.

As mentioned there are three LQR controllers present in the CCC

Following System: This LQR controller's goal is to attain the following distance specified by the driver and then to match its velocity. Additionally,

it also has the goal of minimizing the acceleration since large accelerations can introduce instability into the system, shake up the driver and are bad for fuel economy.

Cruise System: This LQR controller's goal is to minimize acceleration while minimizing the difference between the vehicle's velocity and that requested by the user. In other words this is a traditional Cruise Control system.

Braking System: At present, this system has yet to be implemented. The reason for this is that the RC does not have a proper set of brakes. Furthermore, once the brakes are added to the RC car, an estimate of its transfer function is needed for the LQR design process.

However this controller's goal will be to use the brakes to slow the car down until it is following at a safe distance and velocity.

The outputs of this block are the Throttle and Brake Signals, both of which are represented in the format of a % x10.

Block: *VSS Data Processing Unit*

This block represents the subsystem responsible for carrying out the signal processing on the sensor data provided by the Vehicle Speed Sensor (VSS).

The first stage in this unit is a low pass filter responsible for removing the high-frequency noise which, if left unfiltered, would greatly degrade the control effort. Currently the implementation of this filter has not yet taken place. However, the chapter on System Identification covers many of the considerations that would need to be considered here. The main difference between this situation and the one presented

in Chapter 5, is that here the processing has to happen real-time so the approach used there using *filtfilt* will not be applicable here.
Next, the filtered signal is passed on to the full-state observer which will derive information about the other continuous states (eg acceleration) from the velocity information provided by the VSS. This data is important since LQR controllers require knowledge of the full system state. In this case the Observer that has been implemented is a Kalman filter in 'Triple Integrator form' (Canet, 1994) .

Output: The output of this block is an array of 4 signals, collectively called VDA_{self}, the signals are in order: V_{self}, SS_{state}, d_{self}, a_{self}. Note that the SS_{state} is a by-product of the State Space notation and while it has no physical significance on its own, it is necessary for the calculations of V_{self}.

Block: *Radar Data Processing Unit*

This block is very similar to the *VSS Data Processing Unit* block except it operates on the data from the radar sensor instead of the VSS. Also, its Kalman observer only keeps track of two states (d_{rel} and V_{rel}) both of which the system is able to measure directly. So while there are not any unobserved states that need to be estimated, the Kalman filter can still help improve the quality of d_{rel} and V_{rel} since the relationship between these signals is known.

Example: Since V_{rel} is a derivative of d_{rel}, then if d_{radar} suddenly jumps sharply and V_{radar} does not, then there is possibly something wrong

4.3.1 Blocks representing physical systems

Block: *Model of Physical Vehicle with Sensors*

This block represents the physical system being controlled, in this case the RC car. The main component within this subsystem is the Discrete Vehicle Model which contains the transfer function for the RC car. This transfer function was obtained using the System Identification process. In addition to this there are also blocks that simulate the functionality of the Radar and Vehicle Speed Sensors (VSS) and allow for noise to be artificially introduced for simulation purposes.

Block: *Lead Vehicle*

The final block on the top-level diagram is a simulated leader vehicle. Like the *Model of Physical Vehicle with Sensors* this block is only used for simulation purposes and is not included in the code generation. Within this block the developer can draw out velocity trajectories for the leader vehicle that can be used as test cases when simulating the system.

4.4 Code Generation process

One of the goals of this project was to implement the ACC system on the physical RC car. This set-up would allow for faults to not only be simulated in the Simulink model but also using the physical test bench.

A test bed using an RC car to test active safety systems like ACC is being developed in a related project (Sullivan, 2012). As part of that project, the RC car, whose model is obtained in Chapter 5 of this thesis, was equipped with an ARM

board which provides an API for interfacing with the hardware.

To be able to use the Hybrid ACC model on this test bench, the code was first converted into a fixed-point representations and then into executable C-code. Matlab provides a suite of tools that enabled this process.

- The Model Advisor tool examined the model to ensure that the system was compliant with good coding practices and warns of aspects that may cause problems.
- The Fixed Point tool was used to analyse the range of data each signal would be expected to represent. Following this, it was used to convert the signals into the appropriate fixed point representation.
- The Matlab Embedded Coder toolbox was used to ensure the code was compliant with the needs of the code generator. Once this was complete the fixed point code was generated and then installed on the ARM board.

4.4.1 Porting to the Test Bench

In porting the code to the RC car several changes had to be made to the model to reflect changes in the test bench. The first of these changes is the units of measurement that were used for both velocity and distance. In the model these measurements were done in km/h and km respectively. However, since the test bench used cm and only allowed for 16 bit data types over the CAN bus, attempts to interface the two resulted in large numerical errors. To solve this problem a copy of the model was altered to use cm as its unit of measurement.

The second change was to the sampling rate. The model was originally developed to run at 300 Hz, the frequency used during the system identification process. However, the CAN bus was found to be unable to keep up with the model when it operated at this rate. As a result, the model was reconfigured to run at 100 Hz before it was converted into C code.

Finally, the LQR controllers had to be adjusted using the methods described in Chapter 6 to get the desired response from the system. During this tuning it was found that the trade-off between reducing the steady state error and keeping the LQR smooth made it difficult to find the ideal tuning. In response to this, a small integral signal was introduced into the following controller.

4.4.2 Test Bench Results

Testing of the ACC system on the test bench was done in an incremental fashion, it started with simple tests that were easy to verify. Then once they had been completed and any necessary corrective action taken, the next stage of tests was applied. It is important to note that the test that appear at the beginning of this process are rather simplistic and would not represent actual driving conditions (eg. A leader whose velocity changes as a step function). However, the final two tests capture a more realistic operating environment.

The first step was to verify that the modified model still matched the physical car. To do this, the system was run at various throttle levels and the resulting speed compared to what the model predicted. The results of this test proved that the steady state velocity of the car still matched the results of Chapter 5.

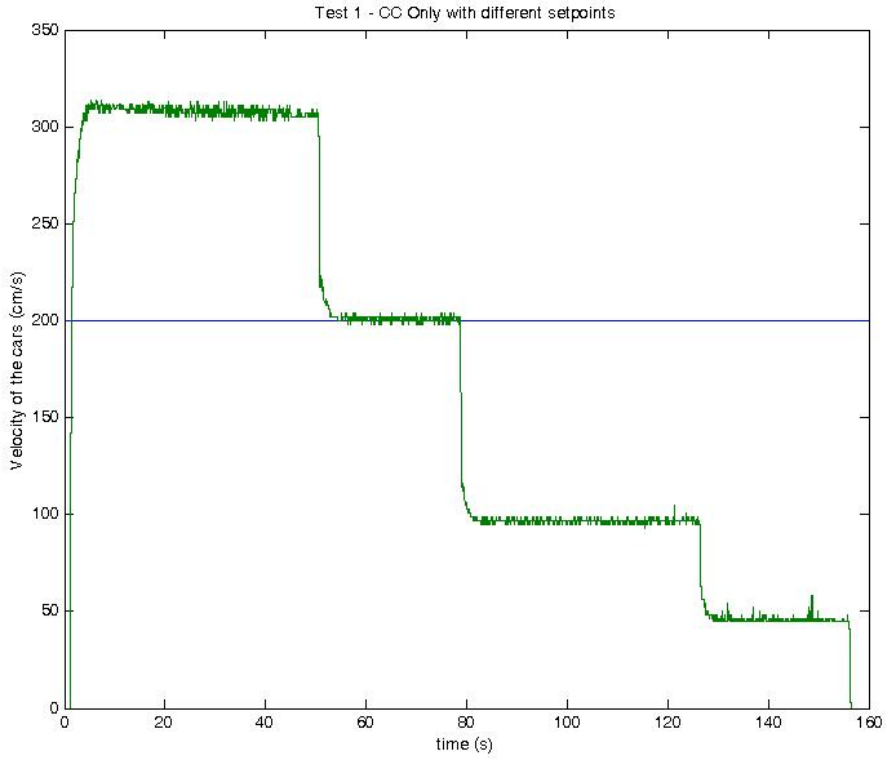


Figure 4.2: The RC Car cruising at 300cm/s, 200cm/s, 100cm/s and 50cm/s

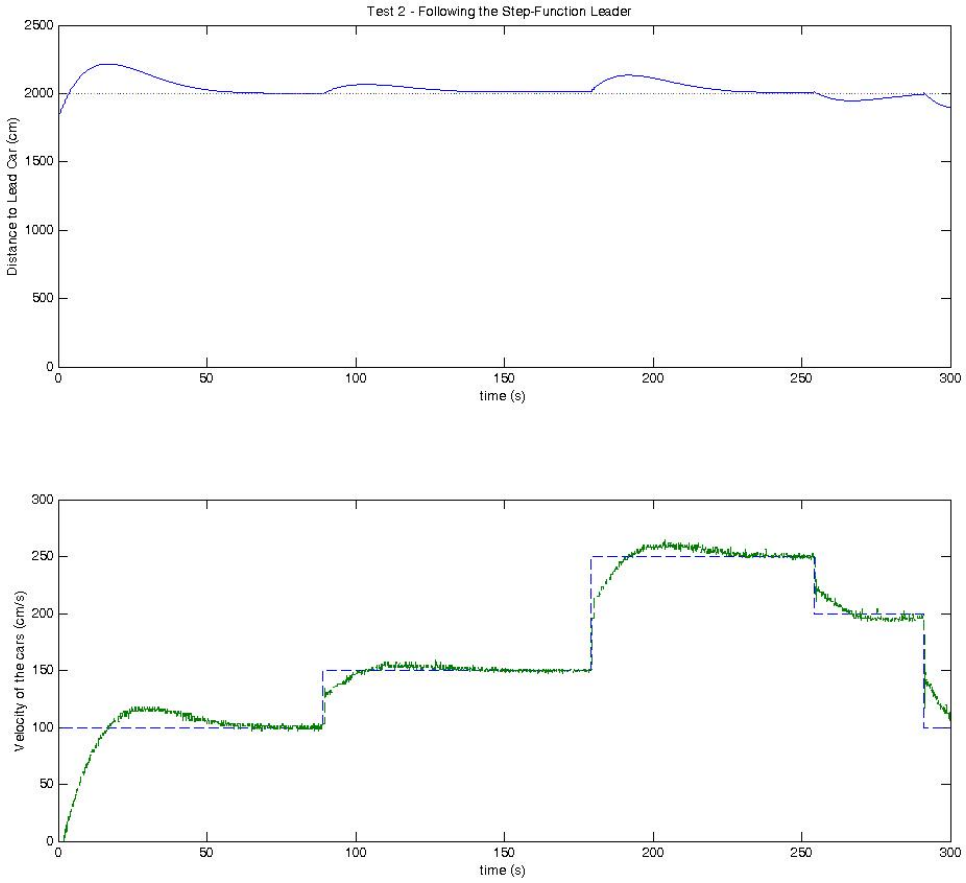


Figure 4.3: The ACC car following a leader travelling at different velocities

The second test was concerned with the behaviour of the car using only traditional cruise control. This was achieved by giving the system different velocity set points while the "Cruise Control Only" option was selected. Then the resulting velocities were compared with the requested velocities. The initial results of this test allowed for the "Cruising" LQR to be tuned to the test bench. The later results were then used to verify the effectiveness of this tuning. An example of the system's performance under this test can be seen in Figure 4.2.

The third test was to place the car in a "Following" situation on the test bench and to adjust the leader's velocity to see how the ACC vehicle would respond. As in the previous case, the test results were used to both tune and verify the corresponding LQR controller. In Figure 4.3 the system is following a leader whose velocity is changing to different velocities according to a step function. Then in the Figure 4.5 the system is following a vehicle whose velocity is changing continuously with respect to time.

Finally, a set of tests was created to test the switching from "Cruising" to "Following" and vice versa. These tests allowed for the verification of the transitions on the physical car. Although these transitions had already been verified on the Simulink model these tests showed that the hysteresis zones had to be adjusted. These adjustments were necessary due to some of the non-linear effects that the model couldn't capture. An example of the system switching from cruising to following can be found in Figure 4.4 while Figure 4.5 shows the opposite.

4.5 Summary

The topic of this chapter was the Simulink model which implements the design of Chapter 3. The ACC system consists of three components: the data processing units, the finite state machine and the continuous controllers. Two additional subsystems represent the physical system being controlled (the car) and the lead vehicle, respectively. For each of these systems the inputs and outputs were presented along with a description of how each subsystem contributes to the whole.

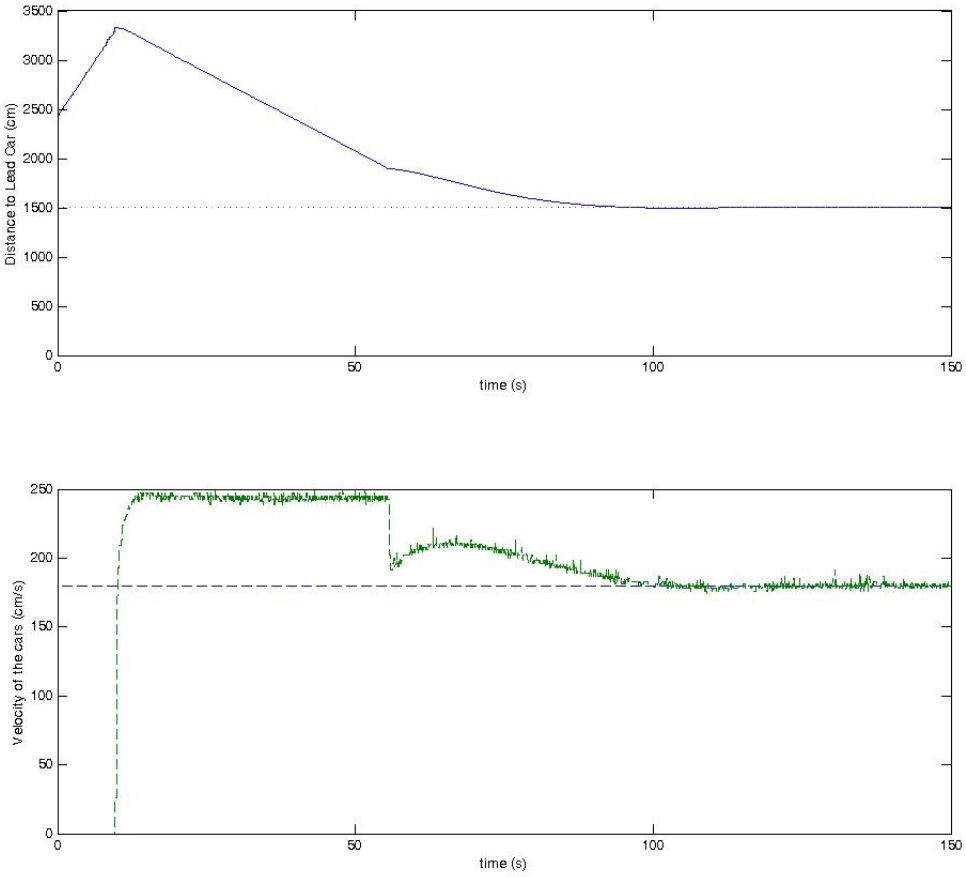


Figure 4.4: ACC is Cruising until it gets within range

This Simulink model was used to simulate the ACC system functioning in response to the different trajectories given to the leader vehicle. The model was also used to generate code that could be used to control the physical car on the test bed. To meet the needs of the ARM-board on the vehicle the model was converted to fixed point code and verified using the Simulink model checker.

An important part of this model was the representation of the physical vehicle found in the block "Model of the Physical Vehicle with Sensors". The next chapter

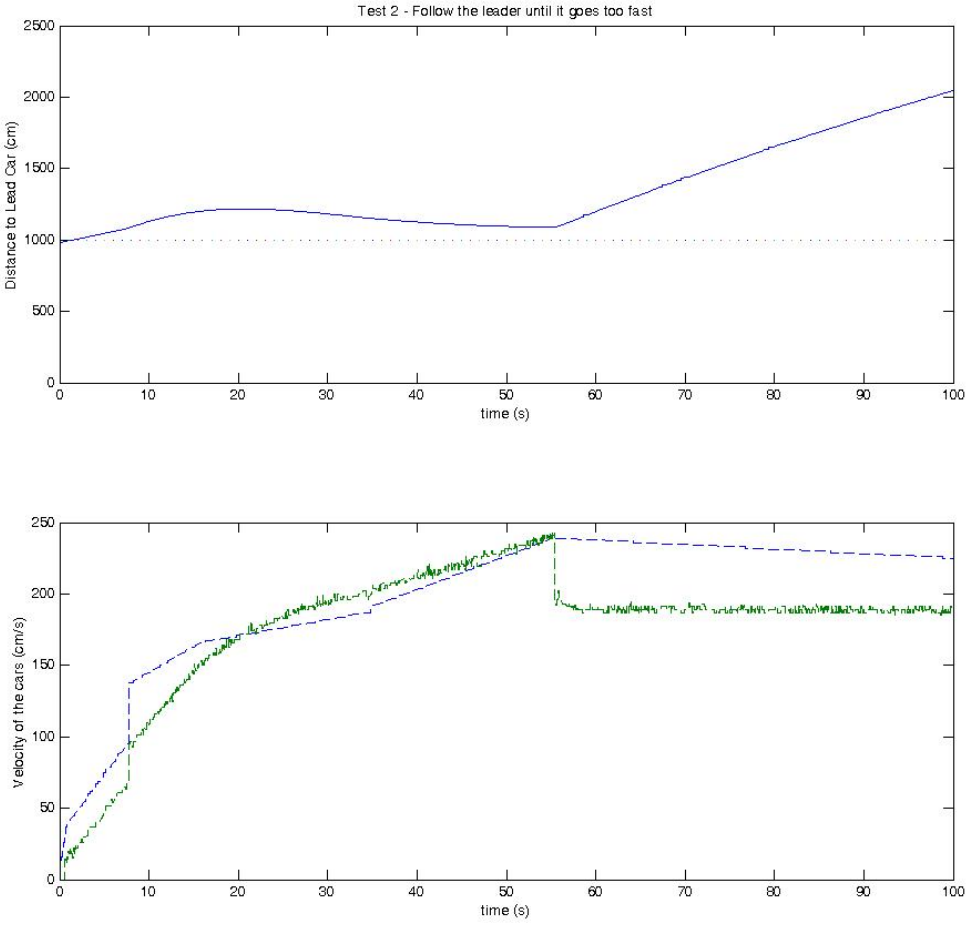


Figure 4.5: Following a car then switching to Cruise

explains how this model was obtained using the System Identification process.

Chapter 5

System Identification of the Adaptive Cruise Control Testbed

System Identification is a technique for obtaining a black box model of a physical system. In this chapter the system that is being identified is the electric motor of an RC car. The resulting model will be used in creating a simulation of the system, designing the continuous controllers and in developing the fault diagnosis.

The process involves providing the system with a known input signal and using sensors to measure and record the output. An assortment of statistical tools will then be applied to the results to derive a model that is able to predict the output of the system. Important aspects to consider in this process include stability, accuracy and computational costs.

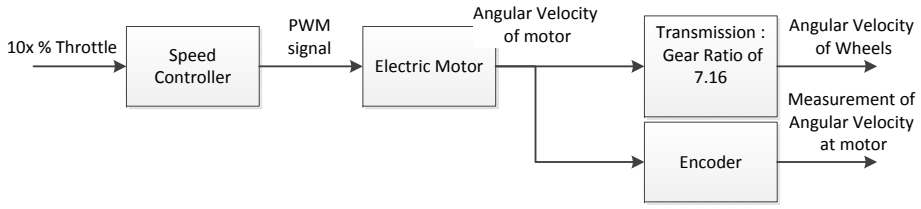


Figure 5.1: System Setup

5.1 The System

The target of this process is a $1/10^{th}$ scale model Remote Control (RC) car. The resulting model will relate the input, provided to the speed controller, to the output, measured at the wheels. The input signal is a number between 0 and 1000 and the units can be thought of as $(\%\ of full throttle) \times 10$. This signal is provided to the speed controller which will convert it into a Pulse-Width Modulated (PWM) signal to drive the motor.

The output is measured by an optical encoder with 256 pulses per revolution. The encoder has been attached to the output gear of the electric motor which drives a simple transmission in order to power the wheels. The gear ratio between the output gear of the motor and the wheels is 7.16, therefore the angular velocity of the wheels can be found by multiplying the measured angular velocity by 7.16.

A limitation of this system, at the time of the System Identification process, was a maximum update period of 3 ms. As a result the maximum rate at which the input could be fed was 333 Hz. For simplicity the sensor's sampling period was set to this same frequency.

5.2 Initial Tests

The system identification process requires that certain assumptions be met about the system. In particular, the model structures that will be used in this report assume that the underlying model is linear and that the noise is Gaussian in nature. Ensuring these structures can be used and that their results will be accurate requires the use of preliminary tests to both verify these assumptions and to gain other valuable information about the system.

5.2.1 Frequency Range of Interest and the Bandwidth of the System

One of the most important parameters to consider when designing the experiment is the range of frequencies that will be excited and measured. When designating this range it is important to capture both the bandwidth of the system and a large enough region around it in order to capture the transient effects. The bandwidth of the system can be found by looking at the ETFEs[1] of the system and locating the -3dB point or the point where the signal is at 70.7% of its original amplitude. The bandwidth of the RC car was found to be around 60 rad/sec or 9.5 Hz.

Ideally the frequency range of interest would be 10× the bandwidth. However, limitations like the signal to noise ratio, the Nyquist rate and the power spectrum of the input can prevent this from being realistic. An examination of the ETFE plots show that at higher frequencies the signal to noise ratio becomes very small making

[1] An ETFE or Empirical Transfer Function Estimate can be found by $G_N(w) = \frac{Y_N(w)}{X_N(w)}$ where $U_N(w)$ and $Y_N(w)$ are the FFT of the input and output signals respectively Habibi (2011b). The ETFE is often represented graphically in the form of a bode plot.

data in that region unreliable.

Additionally, to prevent aliasing in the data, anything above the Nyquist frequency (166 Hz) will be removed via filtering. It will also be important to have a flat power spectrum for the input signal. This is an important point to take into consideration when using PRBS[2] signals for inputs since the flat portion of the PRBS's power spectrum is limited to 1/3rd of the switching rate (Habibi, 2011b). In this case the switching rate was set equal to the sampling rate of 333 Hz.

Taking all these limitations into consideration, the frequency range of interest was decided to be between 0-45 Hz

5.2.2 Delay

One of the most important characteristics to identify, when building a mathematical model of a system, is the time delay from when an input signal is applied to when the system begins to respond to it. Determining the delay of the system, before beginning the main identification process, can greatly simplify the process by reducing the number of models that need to be compared in the identification stage.

Step Input

To determine the delay of the system, a step input was applied to the motor. Figure 5.3 shows the system's response to this step from 30% (2.36V) to 40% (2.49V).

From this plot it can be seen that there is a 2 sample delay from when the input signal is applied, to when the system begins to respond to it. Since the test was done at a sampling rate of 333 Hz that 2 sample delay translates to 0.006 seconds.

[2]PRBS is an acronym for Pseudo-Random Binary Sequence

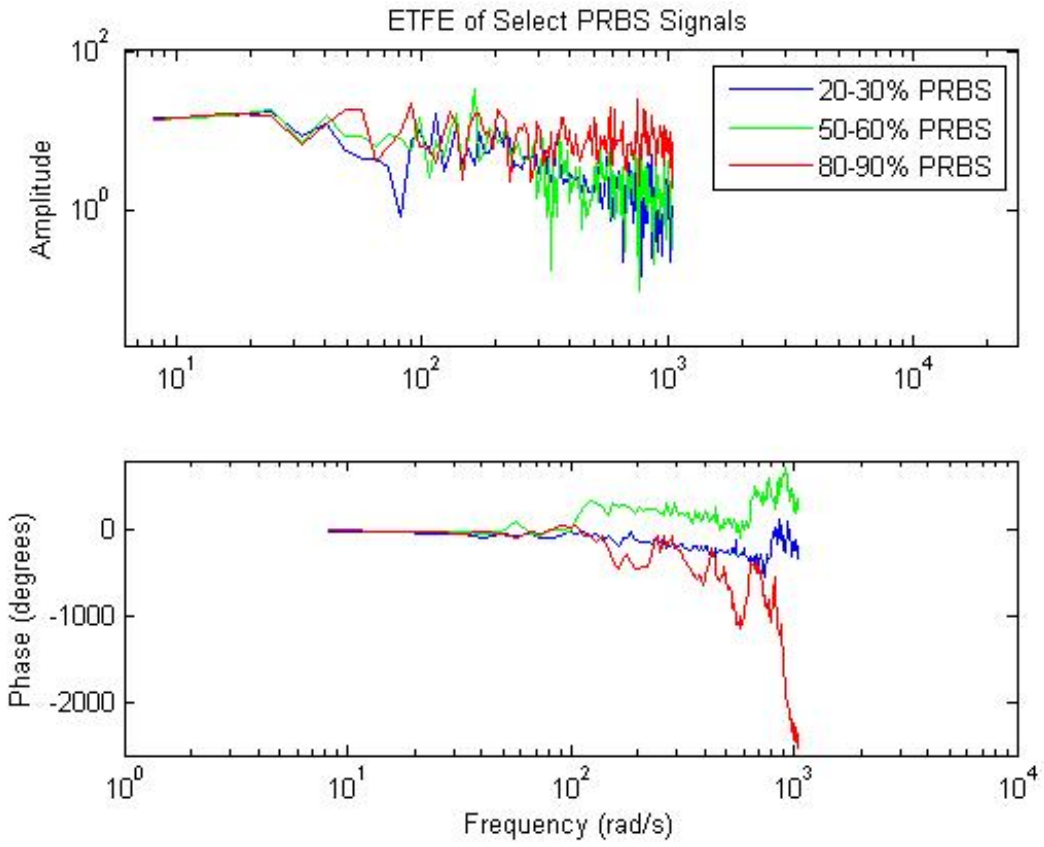


Figure 5.2: ETFE of PRBS with no windowing

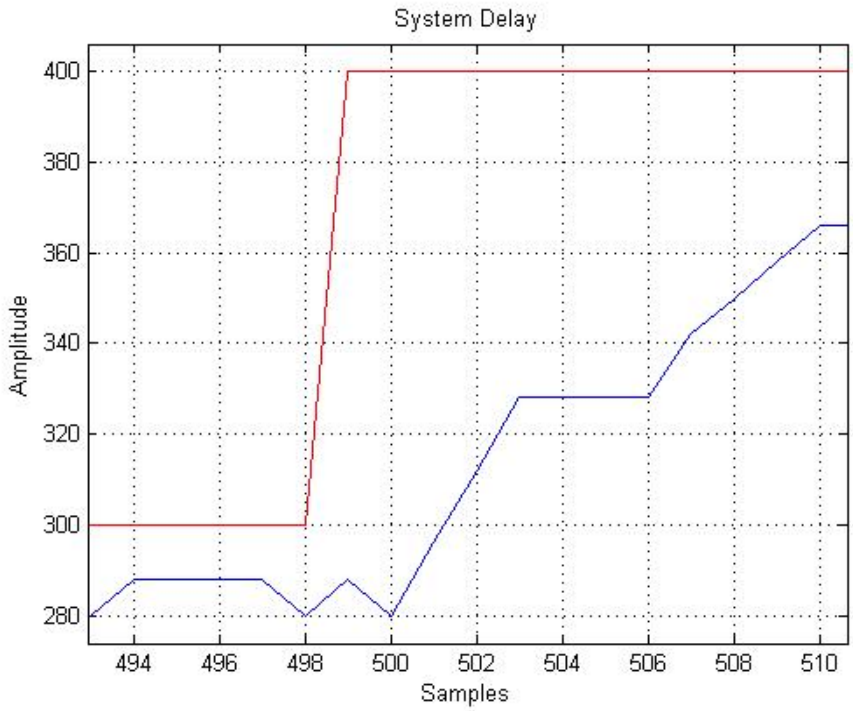


Figure 5.3: Delay in Step from 30% to 40% input. Red denotes the input signal and blue the system's response

Impulse Response

Figure 5.4 shows the system responding to an impulse from 50%(2.63V) to 60% (-2.76V). Since it is very difficult to generate an impulse input due to electrical limitations, a step input from 2.63V-2.76V is applied and the result differentiated. This process will be explained in more detail when discussing the methods used to estimate the model order. However, examination of this plot also shows that the delay can be estimated at 2 samples or 4 milliseconds.

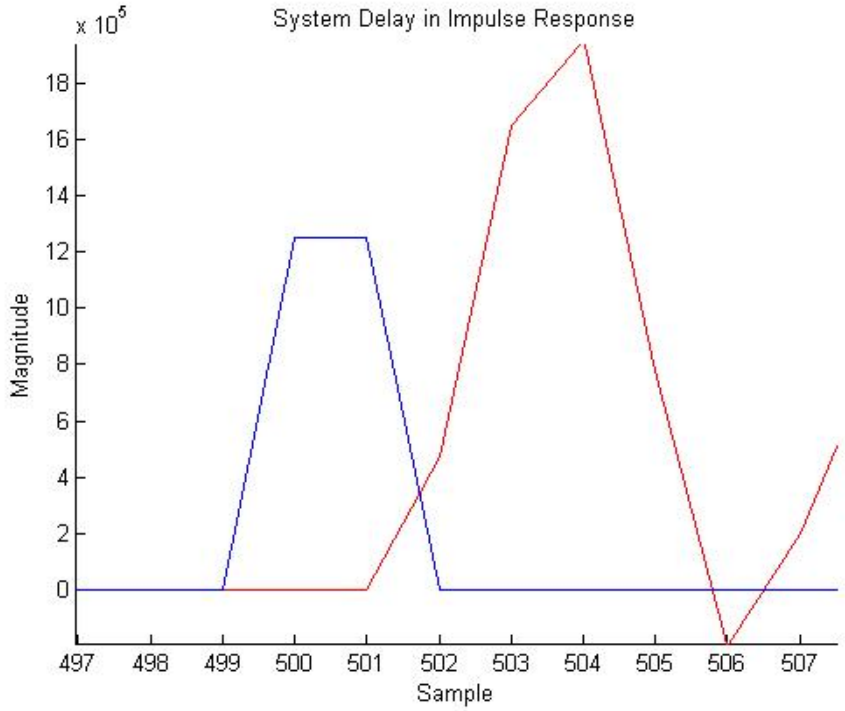


Figure 5.4: Delay in Impulse Signal

5.2.3 Dead Band

A common source of non-linearities within systems is the region known as the Dead Band. In this region the non-linear effects of static friction and the stick-slip phenomenon can be observed, both of which can cause large errors when trying to derive our model.

Identification of this region allows for steps to be taken to reduce its effect. The first step that can be taken is to avoid this region when collecting the data that will be used to identify the model. Secondly, once the model is obtained its use can be restricted to exclude this band or at the very least to apply some appropriate compensation.

To locate this region a sequence of step inputs, with enough time between steps to

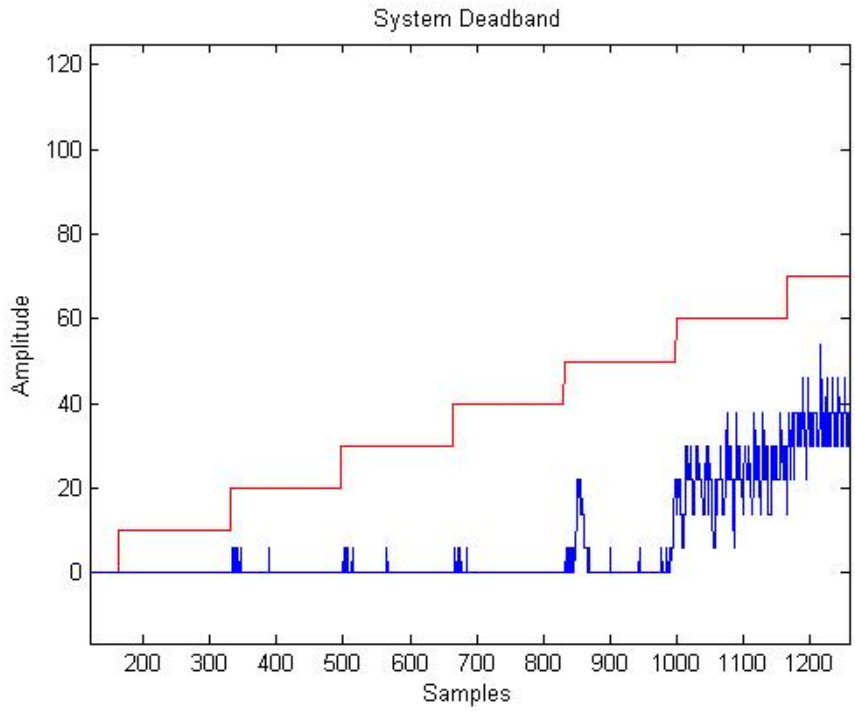


Figure 5.5: Dead band as shown by sequential steps

compensate for the delay, was provided. Figure 5.5 shows the result of this process. In this plot each step has a magnitude of 0.135 mV and the time between steps is 3 seconds. Examination of this plot reveals that the motor enters the linear region at the 6^{th} step or at approximately 2.03 V or 6% full throttle. The effects of the slip-stick phenomenon can be seen in the region preceding that point.

5.2.4 Noise Characteristics

Another assumption that had to be investigated is whether or not the noise in the system could be considered uncorrelated. An estimate of the Welch power spectrum density (PSD) was plotted for a series of step inputs. The resulting plots can be seen in Figures 5.6 and 5.7.

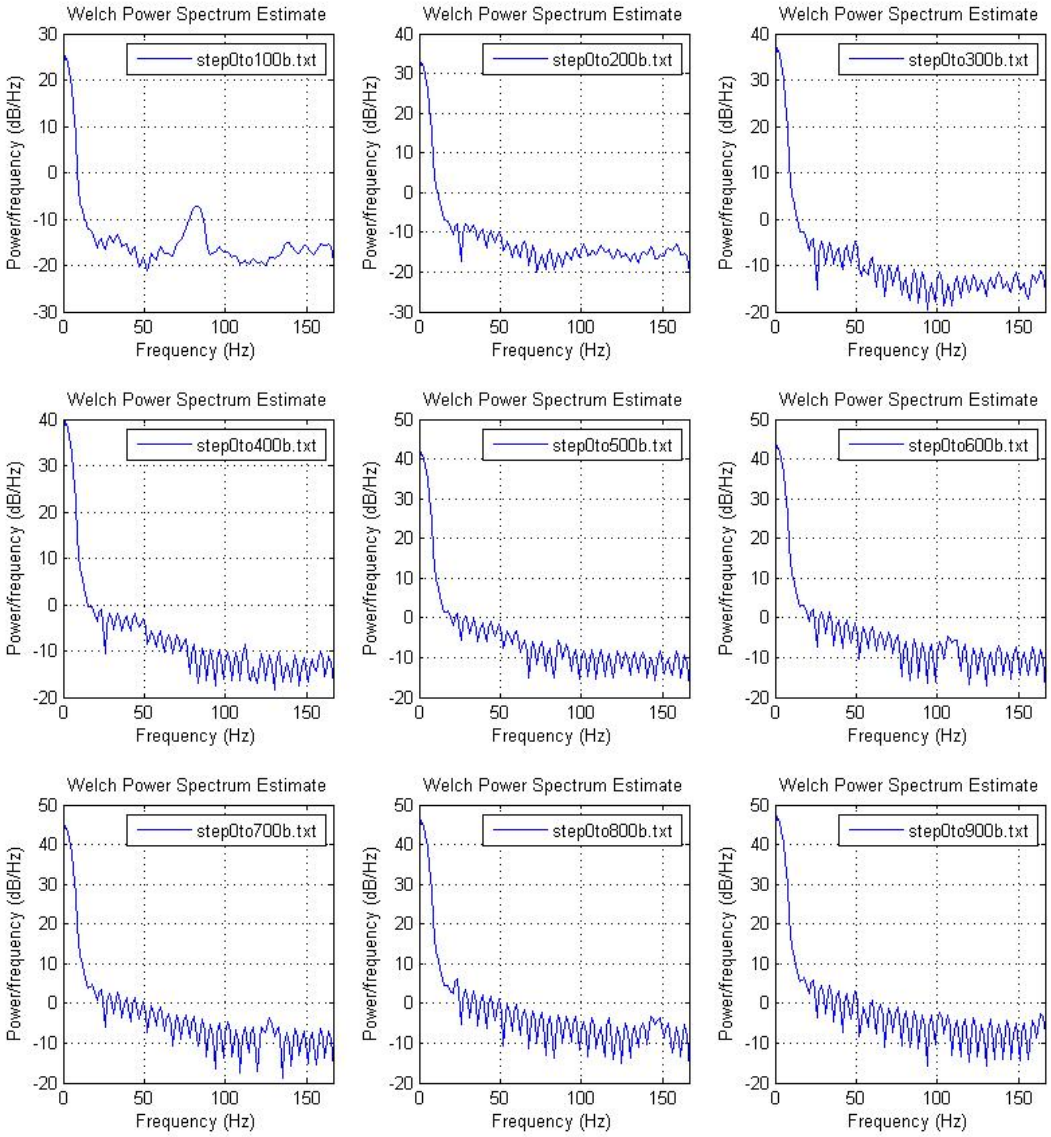


Figure 5.6: Welch Power Spectrum of Steps with Varying Amplitudes

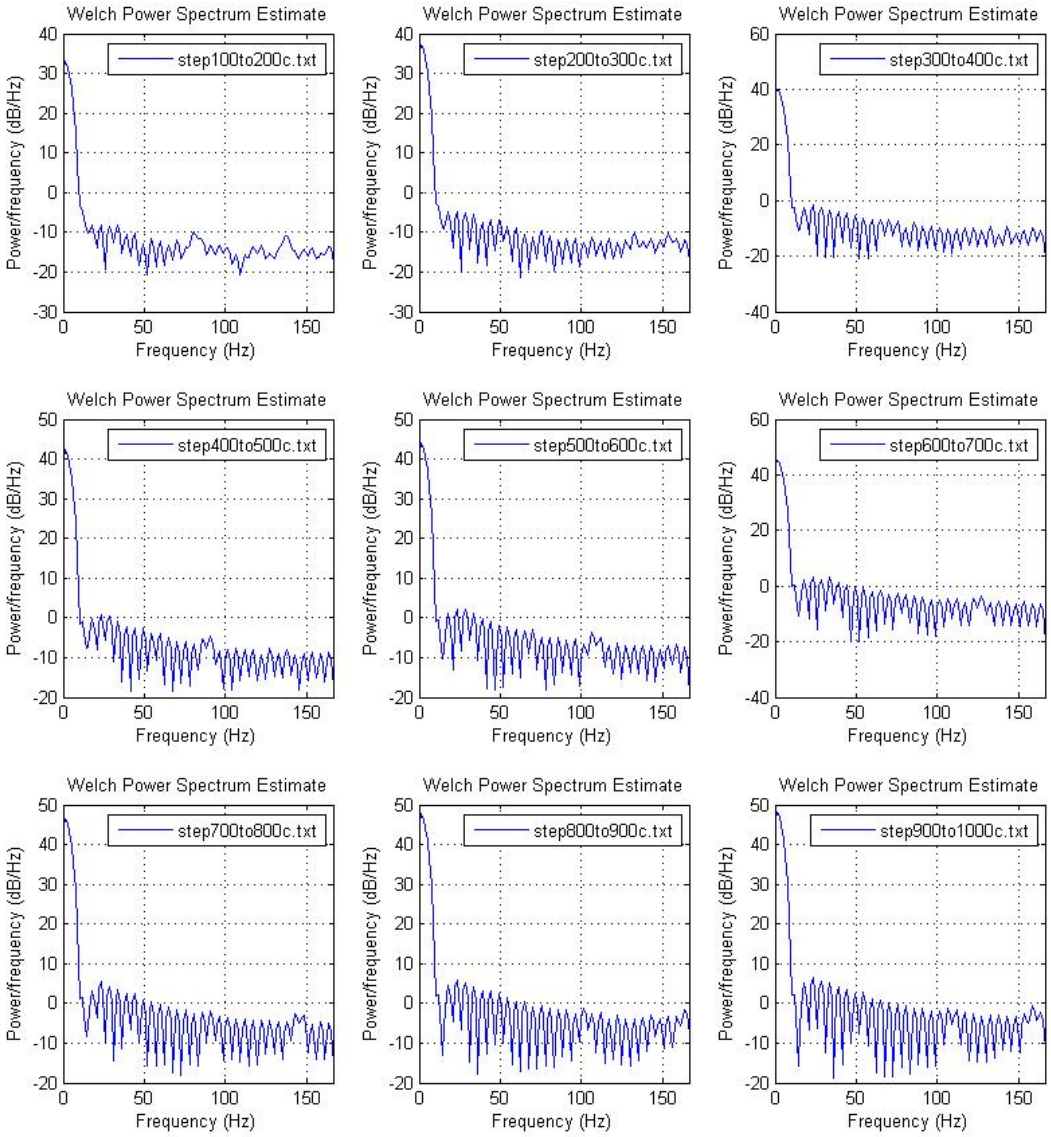


Figure 5.7: Welch Power Spectrum of Steps with Varying Operating Points

An examination of the plots in Figures 5.6 and 5.7 shows an initial spike followed by a period of rapid attenuation. The section following that, from 10-150 Hz, is mostly flat however a slight downward trend visible in Figure 5.6.

The initial spike reveals that a large magnitude for signals with a frequency of >10Hz. This is caused by our step signal which represents a DC signal before and after the change in amplitude takes place.

The real area of interest here is the flatness of the band that follows the initial spike which is mostly noise. The PSD shows the "power" of each frequency present. If this section is flat then the power of the noise is independent of the frequency. The noise in these two plots is fairly flat after the initial spike. However, the slight downwards trend in Figure 5.6 may limit the accuracy of the final model.

5.2.5 Linear Piece-Wise Region

As previously stated, the method being used in this report assumes an underlying linear system. However, it is not uncommon for a system to meet this criteria only within a certain subset of its range. Also possible is the situation where the operating range could contain more than one linear region. In the situation with multiple linear regions, a separate model would need to be obtained for each of the regions. Consequently this section investigates whether the RC car has zero, one or multiple linear regions.

DC Gain Test

The first method used to investigate these questions was the DC gain test. In this test, the system was fed a series of step inputs and the average of their *steady state*

values recorded. This was done 3 times for each voltage level to improve reliability.

Input	**Input (V)**	**RPM of Motor**			**Predicted Value**	**Residual**		
		Trial 1	**Trial 2**	**Trial 3**		**Trial 1**	**Trial 2**	**Trial 3**
100	2.085	36.931	40.660	40.829	*44.019*	7.088	3.350	3.190
200	2.220	96.228	96.769	96.867	*99.638*	3.409	2.868	2.771
300	2.355	154.075	155.024	155.042	*155.256*	1.181	0.232	0.214
400	2.490	210.960	211.357	211.874	*210.875*	-0.085	-0.482	-0.998
500	2.625	269.816	270.233	270.277	*266.494*	-3.322	-3.739	-3.783
600	2.760	326.102	326.403	326.661	*322.112*	-3.990	-4.291	-4.548
700	2.895	380.949	381.350	381.401	*377.731*	-3.218	-3.619	-3.670
800	3.030	439.475	439.554	439.670	*433.350*	-6.125	-6.205	-6.320
900	3.165	496.197	496.235	496.155	*488.968*	-7.229	-7.267	-7.187
1000	3.300	527.355	527.115	527.284	*544.587*	17.232	17.472	17.303

Table 5.1: Results of the DC Gain Test

Linear regression was performed on this data and the results plotted in Figure 5.8. The linear regression found the relationship to be

$$RPM_{Output} = 412 * (V_{Input}) - 815 \tag{5.1}$$

The corresponding correlation coefficient of 0.9982 suggests that this is a very good fit and that the system is indeed linear.

Delving a little deeper into the data, the residuals at each voltage level were also produced and plotted in Figures 5.9 and 5.10. These residuals were found by using equation (5.1) to predict what the RPM of the motor would be at that voltage level. The difference between the predicted value and the actual value then became the residual. The relative residual is defined here as the residual divided by the magnitude of the predicted value.

The results from the residual analysis also support the claim that system has one linear region in the defined operating region of 0 - 1000. However, the residuals also

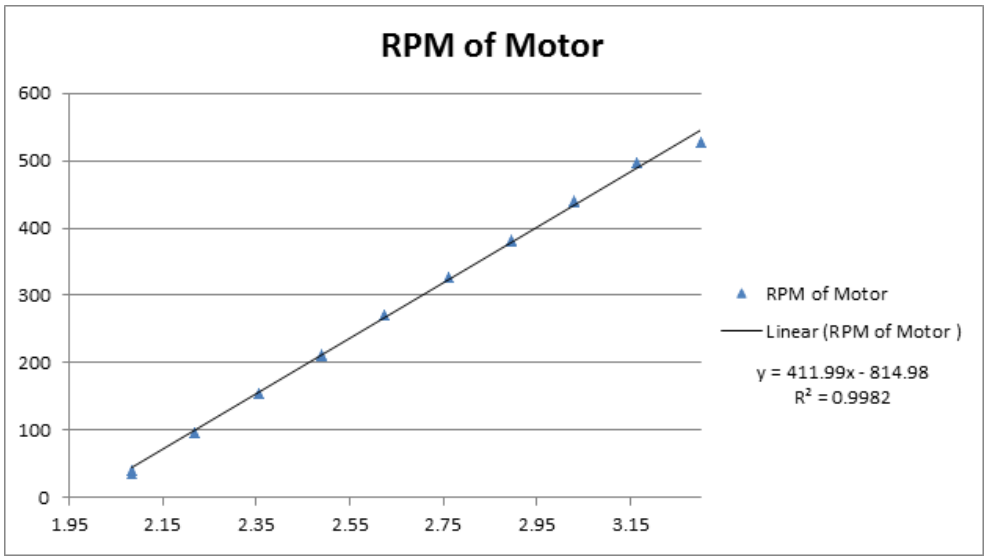
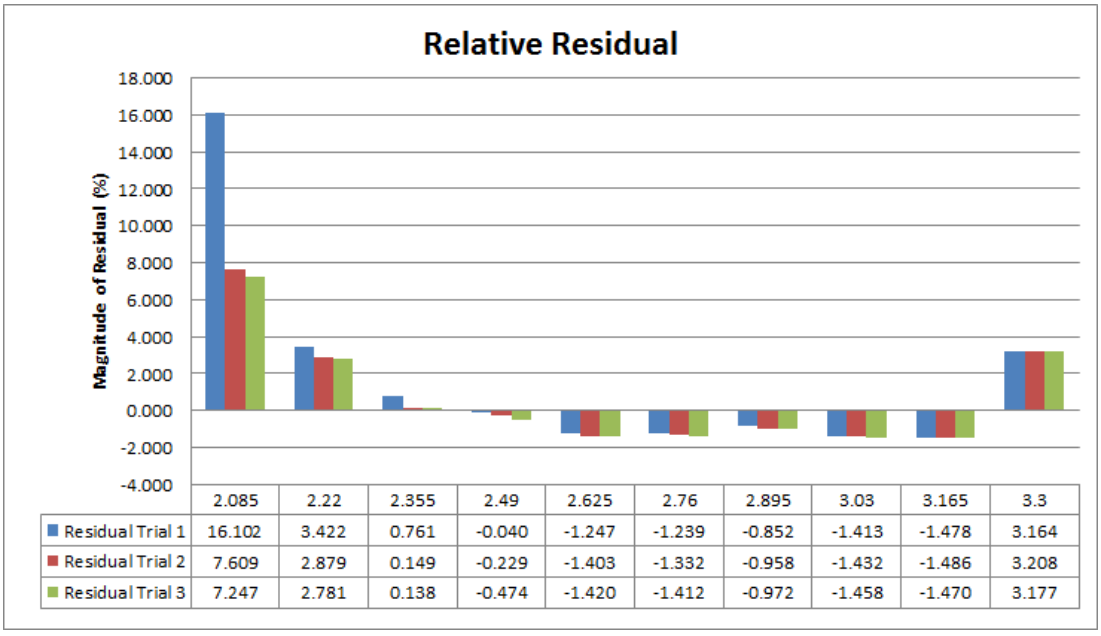


Figure 5.8: Plot of the motors steady state gain across the operating region

reveal that at both ends of this range, the values don't fit as well. As a result it can be expected that the model won't be as accurate in predicting values that appear near the boundaries. Another conclusion that can be drawn from this data is that the data collection from the system identification should come from the middle of the operating range.

	2.085	2.22	2.355	2.49	2.625	2.76	2.895	3.03	3.165	3.3
Residual Trial 1	7.088	3.409	1.181	-0.085	-3.322	-3.990	-3.218	-6.125	-7.229	17.232
Residual Trial 2	3.350	2.868	0.232	-0.482	-3.739	-4.291	-3.619	-6.205	-7.267	17.472
Residual Trial 3	3.190	2.771	0.214	-0.998	-3.783	-4.548	-3.670	-6.320	-7.187	17.303

Figure 5.9: Plot of the residuals from the DC Gain test



	2.085	2.22	2.355	2.49	2.625	2.76	2.895	3.03	3.165	3.3
Residual Trial 1	16.102	3.422	0.761	-0.040	-1.247	-1.239	-0.852	-1.413	-1.478	3.164
Residual Trial 2	7.609	2.879	0.149	-0.229	-1.403	-1.332	-0.958	-1.432	-1.486	3.208
Residual Trial 3	7.247	2.781	0.138	-0.474	-1.420	-1.412	-0.972	-1.458	-1.470	3.177

Figure 5.10: Plot of the relative residuals from the DC Gain test

5.3 Model Order Estimation

When using linear regression to fit a model to data a common pitfall is over-parameterization. Higher-order model estimations can often appear to give a better fit for a particular set of data. However, these results are misleading and the effect will be noticed when the model is used to estimate a different type of input. To counter this effect two steps are taken: The first is that the data used to create the model will not be the data used to verify it, though this will be explained more fully later on. The second step is to determine the model's order prior to the main estimation.

5.3.1 ARX

The first method used to approximate the order of the system involves generating a series of models within the expected range of system order. Since the delay of the system was already known to be 2 samples, that value was held constant while the orders of the numerator and denominator were varied from one to ten. The models were created from a PRBS signal switching between 50-60% of full throttle.

Prior to generating the models, the data was split into two intervals. The first interval was then used to generate each model and the second interval was used to assess the level of fit the model could achieve. The results of this process are included in the box plots below. The first boxplot shows the ARX estimate of the system using unfiltered data to generate the models. The data used to generate the second plot was filtered with a ChebyChev type 2 filter as explained in Section 5.4.

- Figure 5.11 shows the fit level of the different models when unfiltered data is used to create and verify the ARX models.

- Figure 5.12 shows the fit level of the different models when filtered data is used to create and verify the ARX models.
- Figure 5.13 shows the fit level of the different models when filtered data is used to create the ARX models and filtered data is used to verify them.

According to (McCullough, 2011), the model which shows the greatest improvement over the previous model can be regarded as the best fit. In figures 5.11 and 5.12 the bar that corresponds to the greatest drop is the second bar. In both of these diagrams that second bar represents the output of the ARX 222 model[3][4].

[3]The identity of each bar can be found in Matlab's ident tool by selecting the bar of interest
[4]The ARX model is discussed in more details in Section A.2 of the Appendix

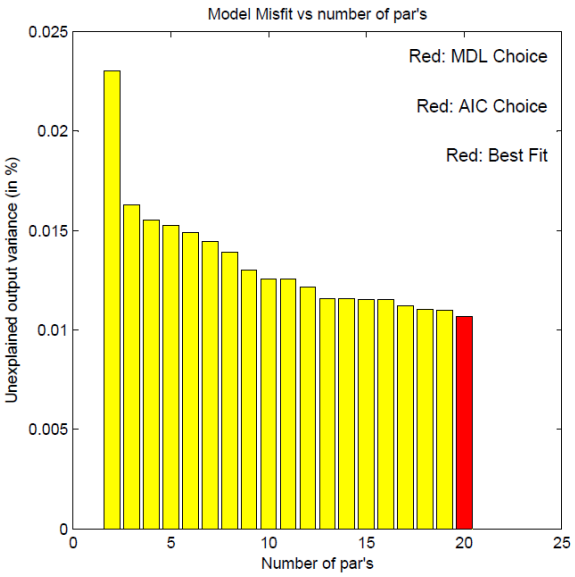


Figure 5.11: ARX estimate on unfiltered data

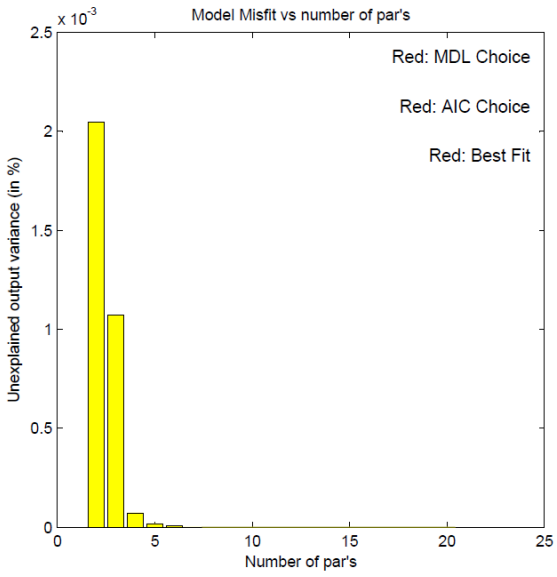


Figure 5.12: ARX order estimate on filtered data

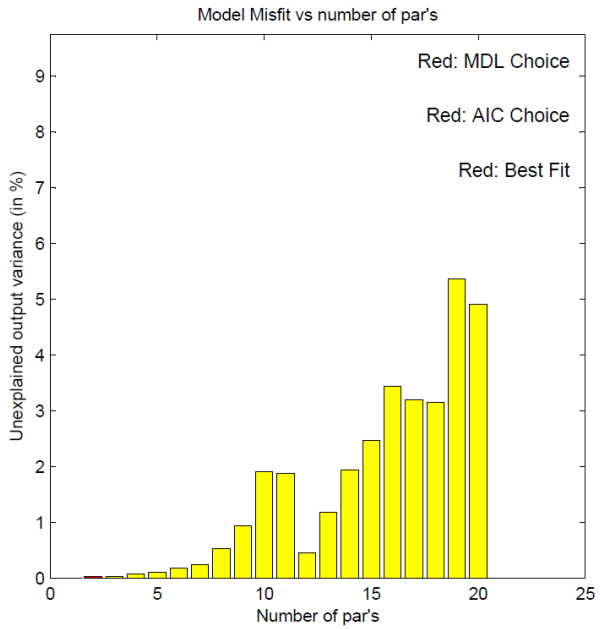


Figure 5.13: ARX order estimate using filtered data to estimate unfiltered data

5.3.2 Impulse Response

The second method used to estimate the system order involves analysing the system's response to an impulse input. If a system's response to an impulse input is put into a Hankel matrix, then the rank of this matrix will be the order of the system (Habibi, 2011b). However, there are a number of physical limitations that will make the process a little more challenging.

The first of these challenges is that it is impossible to generate a true impulse input due to limitations in the electrical control circuitry. Fortunately, the effect of an impulse can be approximated by taking the derivative of the system's response to a step input.

Once the data is collected, it needs to be filtered to remove the noise from the sensors or any frequency about 45 Hz, the upper bound on the frequency range of interest. A low pass ChebyChev II filter was used for this purpose. Section 5.4 explains how this filter was chosen.

After filtering the data, its derivative was taken and placed into a Hankel matrix. However, filtering is unable to remove all the noise from the system, a situation which is only made worse by taking the derivative of the signal. As a result, determining the system's rank is a non-trivial task.

As an alternative to finding the rank of the matrix directly, Singular Value Decomposition or SVD was employed to factor the system and to find the matrix S. S is a diagonal matrix containing the singular values of the Hankel matrix on which SVD was used.

The results of this process are shown in Figure 5.14 which shows the system to be second order. It is easy to see that the noise still has an effect on the system, but

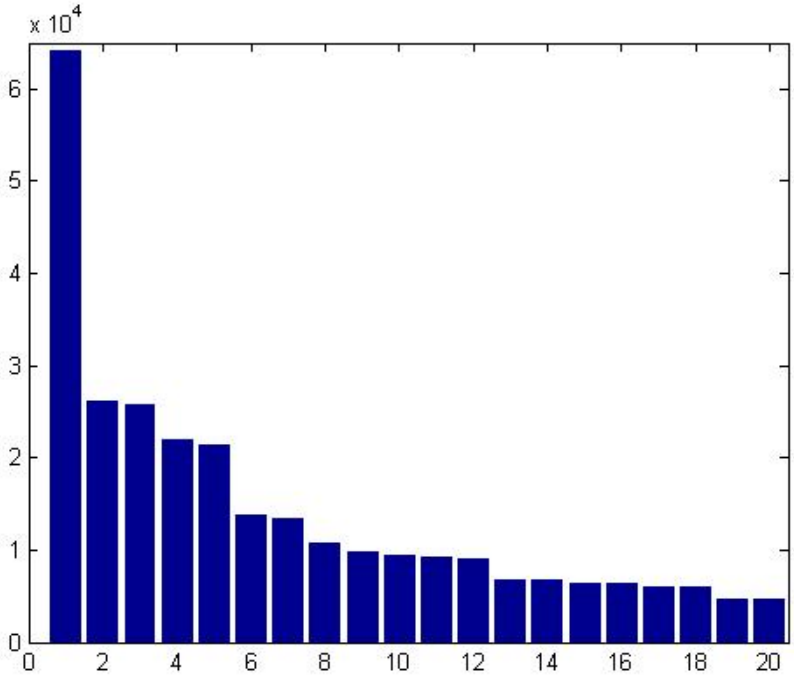


Figure 5.14: Singular Values Plot (Zoomed in)

this method allows for the number of dominant singular values to be determined.

Since the second bar shows the most significant drop in Figure 5.14, the system is once again found to be a second order system (McCullough, 2011).

The Matlab code used to carry out this process can be seen below

```
%Choose data and filter the signal
    [b,a]=cheby2(10,40,45/(333/2),'low');
    data=filtfilt(b,a,y_stepAmp(:,6));

    %select the impulse region
    data=data(100:400);
```

```
%Differentiate the step into an impulse
for t=2:length(data)-1
    dx(t)=(data(t+1)-data(t-1))/(2*0.003);
end

%Create Hankel Matrix and find its singular values
han=hankel(dx);
[U,S,V]=svd(han);

%Plot the result
figure(92);bar(diag(S));axis([0, 20.5, 0, 6.5*10^4]);
```

5.3.3 Resulting Model Order

In both tests, the system was found to be 2^{nd} order, though there is strong evidence to suggest that while there are second order dynamics present, the first order dynamics will be dominant. These results are useful for reducing the number of different models that need to be estimated in the main estimation process. However, since the system's first order dynamics are so much stronger than the second order dynamics, both first and second order models will be investigated.

5.4 Data Processing

The model produced by the system identification is highly dependent on the quality of the data used to create it. Consequently, the data processing stage is very important in ensuring the quality of the final result. In this section two different Infinite Impulse Response (IIR) filters will be investigated.

The disadvantage of using IIR filters is that they don't have a linear phase which will result in non-equal time delays for different frequencies. However, since the calculations are being completed off-line, and therefore all the data is available before hand, zero-phase filtering can be carried out using Matlab's *filtfilt* command. The *filtfit* command will filter the data, flip it, filter it again and then flip it back. Doing so will ensure the phase of the signal is kept intact.

5.4.1 Filter Design

The choice of filters relies heavily on what the data will be used for. The four most common types of IIR filters are the Butterworth, ChebyChev type I, ChebyChev type II and Elliptical filters. Of these only the Butterworth and ChebChev type II filters are classified as maximally flat in the pass band.

A comparison was made of a 10^{th} order Butterworth filter with a 10^{th} order Cheby-Chev II filter. Both of these filters were designed to have a cut-off frequency of 45 Hz. The advantage of the Butterworth filter is that it is maximally flat in both the pass band and the stop band. The ChebyChev II filter on the other hand, is only flat in the pass band but can achieve a much sharper drop-off than the Butterworth can for a given number of poles.

The flatness of the stop band is not terribly important for system identification so

long as the attenuation in this range is sufficiently large. It can be seen from Figure 5.16 that the attenuation in ChebyChev II's stop band oscillates between 40 and 70 dB. It can also be seen that the drop off is much sharper than what the Butterworth filter was able to achieve in Figure 5.15.

The ChebyChev type II filter was chosen for all data processing carried out in this chapter[5]. This decision was made because it had better drop-off than the Butterworth and maintained the flatness of the pass band.

[5]It should also be noted that the experiments were also carried out using the Butterworth filter for comparison and the difference was quite noticeable

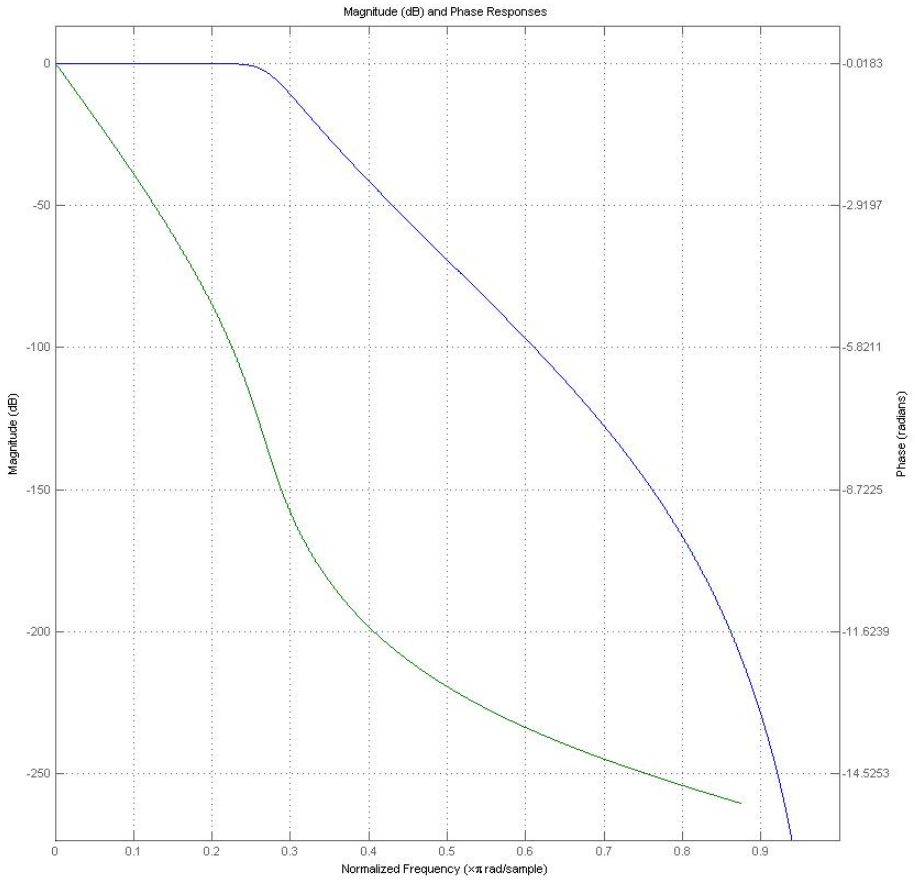


Figure 5.15: Butterworth Response

```
%Butterworth Filter
  clear filt
  [filt(:,1) filt(:,2)]= butter(10,45/(333/2),'low');
  fvtool(filt(:,1), filt(:,2) );
  %filtfilt() will be used to carry out zero-phase filtering
```

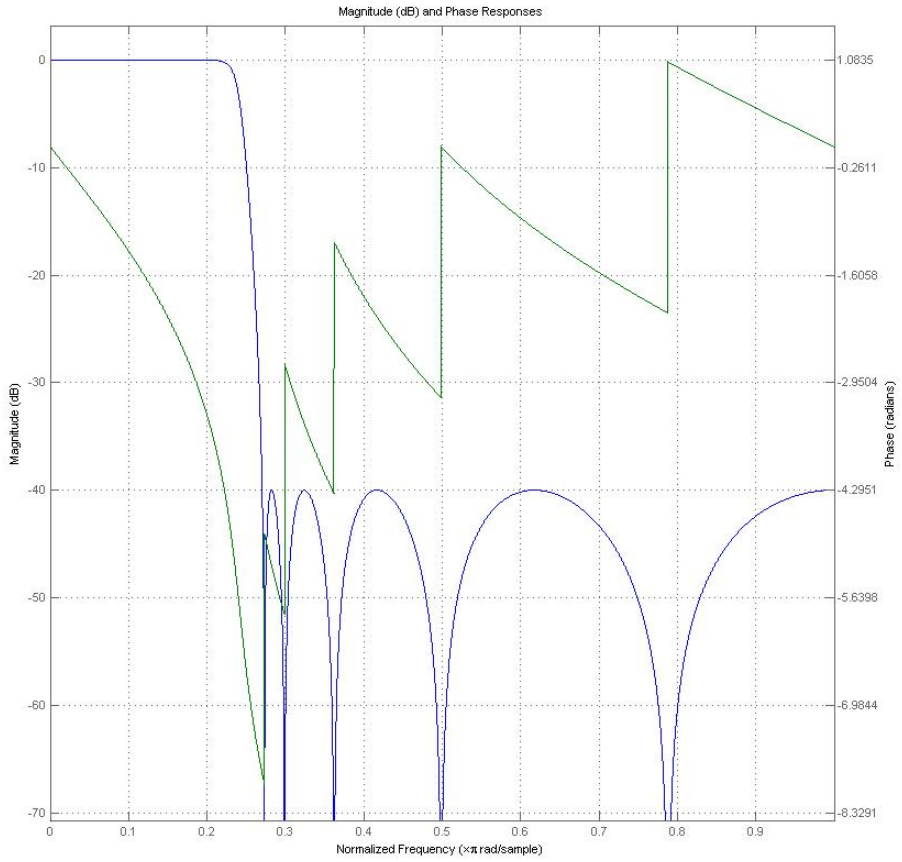


Figure 5.16: ChebyChev II Response

```
%Chebychev2 Filter
   clear filt
   [filt(:,1) filt(:,2)]= cheby2(10,40,45/(333/2),'low');
   fvtool(filt(:,1), filt(:,2) );
   %filtfilt() will be used to carry out zero-phase filtering
```

5.4.2 The Difference Filtering Makes

Figure 5.17 demonstrates the effect that our 10^{th} order ChebyChev II filter has on the data. The plot shown here is an ETFE of the system's behaviour before (blue) and after (green) filtering. The plot shows how the data up to 45 Hz (approximately 280 rad/s) remains mostly untouched and how shortly after the 45 Hz point, the signal has been greatly attenuated.

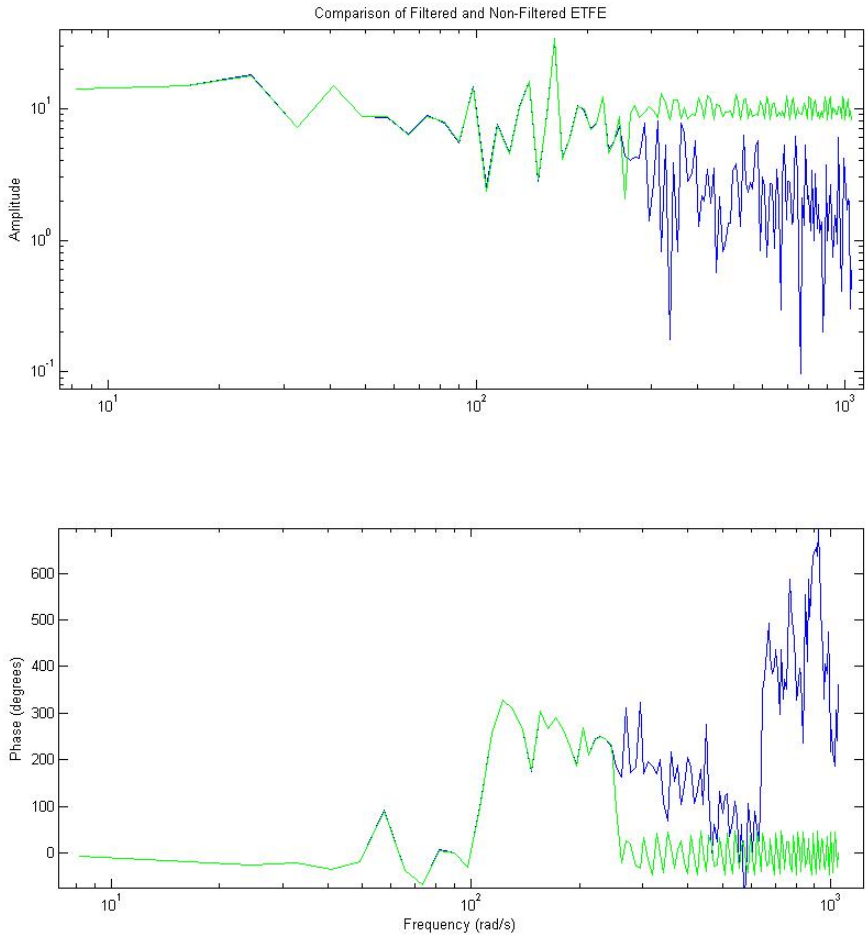


Figure 5.17: The effect of the ChebyChev II filter on a PRBS signal. Blue is the original signal and green the filtered signal

5.5 Model Estimation

The actual model estimation portion of this project was done using a process known as "Parameter Identification" (Habibi, 2011a). In this process the data is fit to a predefined model structure with a number of parameters. The choice of these parameters will define the model's behaviour. The models that were used for this process are ARX, ARMAX, Output Error (OE) and the Box Jenkins (BJ) model. A brief description of each of these can be found in the Appendix in Section A.2.

The estimation process was done in an iterative fashion. The process was broken down into a series of phases and the estimation process began with the simplest types of models and progressed towards the more complex models. This progression allowed for the simpler models with less parameters (ARX and OE) to identify the parameters which were also shared by the more complex models (ARMAX and Box Jenkins). As a result only 2-4 different Box Jenkins models needed to be compared rather than the 16 that would have had to have been compared had the process gone straight to Box-Jenkins.

The complete list of phases is listed below

1. Phase 1: Compare the Fit and Residuals of various ARX models
2. Phase 2: Compare the Fit and Residuals of various OE models
3. Phase 3: Compare the Fit and Residuals of various ARMAX models
4. Phase 4: Compare the Fit and Residuals of various Box Jenkins models
5. Phase 5: Compare the Fit and Residuals of the top model from each of phases 1-4

6. Phase 6: Compare the Fit and Residuals of the top model from each of phases 1-4 on other types of input data like steps and chirp signals

Phases 1-4 were used to choose the best ARX, OE, ARMAX and Box Jenkins models in an elimination style. The "winner" of each of these phases was then compared in phases 5 and 6. For the sake of brevity, only the results of phases 5 and 6 are reported on here. Phase 5 is reported on in this section of the thesis and Phase 6's results are shown in Section 5.7 *Model Evaluation and Validation.*

ARX 222

$$y(z) = \frac{2.225z^{-2} - 2.067z^{-3}}{1 - 1.744z^{-1} + 0.755z^{-2}}u(z) + \frac{1}{1 - 1.744z^{-1} + 0.755z^{-2}}e(z) \tag{5.2}$$

OE 222

$$y(z) = \frac{2.22z^{-2} - 1.698z^{-3}}{1 - 1.41z^{-1} + 0.444z^{-2}}u(z) + e(z) \tag{5.3}$$

ARMAX 2222

$$y(z) = \frac{2.22z^{-2} - 2.046z^{-3}}{1 - 1.744z^{-1} + 0.756z^{-2}}u(z) + \frac{1 + 1.952z^{-1} + 0.967z^{-2}}{1 - 1.744z^{-1} + 0.756z^{-2}}e(z) \tag{5.4}$$

BJ 22222

$$y(z) = \frac{1.725z^{-2} - 1.357z^{-3}}{1 - 1.568z^{-1} + 0.5924z^{-2}}u(z) + \frac{1 + 1.942z^{-1} + 0.963z^{-2}}{1 - 1.764z^{-1} + 0.9439z^{-2}}e(z) \tag{5.5}$$

5.6 Model Fit

The fit of the model can be assessed by applying an input signal to the model and examining how closely its output matches the output of the actual system. Here the model's fit is defined as "the percentage of the measured output that was explained by the model" (MATLAB, 2012) and is calculated as

$$Fit = 100 * \left(1 - \frac{\left\|Y - \hat{Y}\right\|}{\|Y - mean\,(Y)\|}\right) \tag{5.6}$$

In Table 5.2 the fit of each model is shown with different prediction horizons. The Matlab documentation states that "Measured output values in data up to time t-prediction_horizon are used to predict the output of *sys* at time t (MATLAB, 2012)" and that "For time-series models, use a finite value for Prediction_horizon. (MATLAB, 2012)". Since the discrete models we are using are second order with a time delay we are most concerned with the results for a prediction horizon of length 3. The data in Table 5.2 shows that the ARMAX and BJ models display the best fit for the given validation data.

Prediction Horizon	**Model Fit**			
	ARX	**OE**	**ARMAX**	**Box Jenkins**
1	93.43%	64.24%	98.15%	98.72%
2	82.50%	64.24%	91.65%	94.19%
3	69.43%	64.24%	80.62%	86.65%
5	33.07%	64.24%	53.73%	71.01%
Simulation	49.74%	64.24%	48.44%	62.72%

Table 5.2: Comparing Model fit over different prediction horizons

5.6.1 Model Residue

The model residuals are the differences between the model predicted output and what was measured. In effect, the residuals are the part of the output signal that the model failed to reproduce.

Additionally, the correlation between the current and past residuals (the auto-correlations) and the correlation between the input and residuals (the cross-correlation) can provide insight on the maturity of the model. That is, by studying where the model fails to predict the output of the system accurately, a better understanding of the model's effectiveness can be obtained.

The plots in Figures 5.18, 5.19, 5.20 and 5.21 each consist of two sub plots. The top sub plot shows the auto-correlation of output residuals while the bottom sub plot shows the cross correlation.

The autocorrelation plots for the ARX ,ARMAX and OE models eventually settle into the 95% confidence interval, but the BJ model continues to display oscillations even 40 samples after the input. This would suggest that there is information present in past outputs that could have been used to more accurately predict the current output.

The cross correlation plots of the OE and BJ models reveal the presence of correlation between the inputs and the residuals that doesn't attenuate over time. On the other hand the ARX and ARMAX models display rather a strong correlation between the inputs and the residuals during the first 15 samples, however after that the strength of the correlation attenuates into the 99% confidence interval.

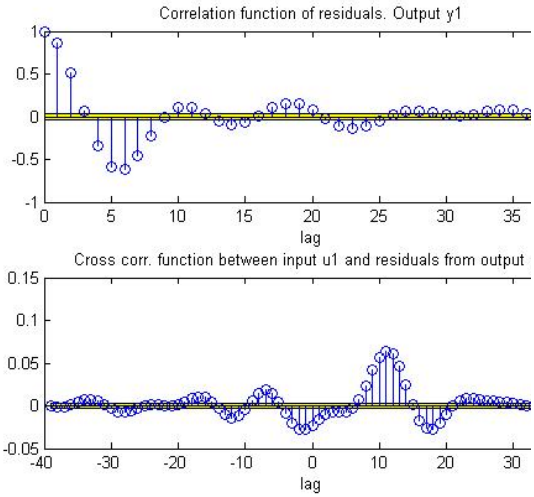


Figure 5.18: Residue plot of ARX 222

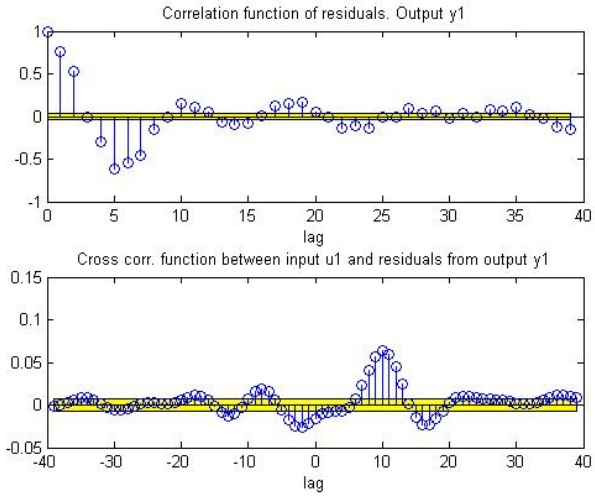


Figure 5.19: Residue plot of ARMAX 2222

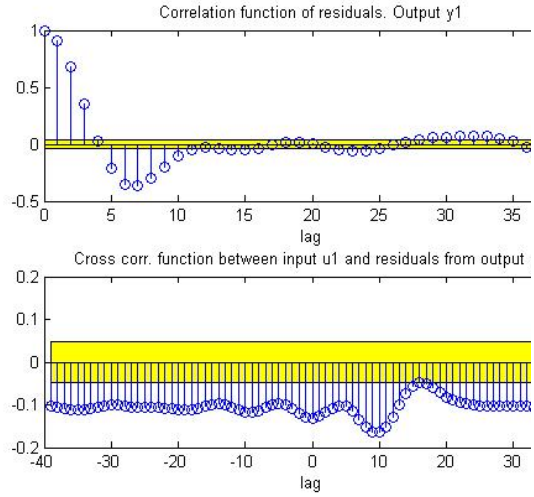


Figure 5.20: Residue plot of OE 222

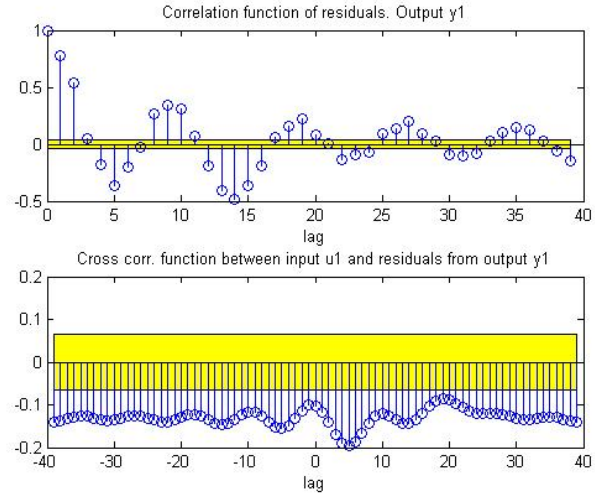


Figure 5.21: Residue plot of BJ 22222

5.6.2 Poles Analysis

Stability is an important property to consider when evaluating a model. To investigate the stability of equations (5.2), (5.3), (5.4) and (5.5), an analysis of their poles was carried out. Figure 5.22 show the poles and zeroes of each of these models plotted on the unit circle. Figure 5.23 is also included and focuses on the area of interest.

A system is considered to be stable if all of its poles, which are represented as $\times$ in the diagram, lie within the unit circle. The presence of poles on the unit circle itself would indicate that the system is marginally stable while a system with poles outside the circle are considered unstable.

Although it is difficult to make out the individual poles in Figure 5.23 it can be seen that they all lie within the permitted range. Therefore, it can be safely assumed that the systems are stable.

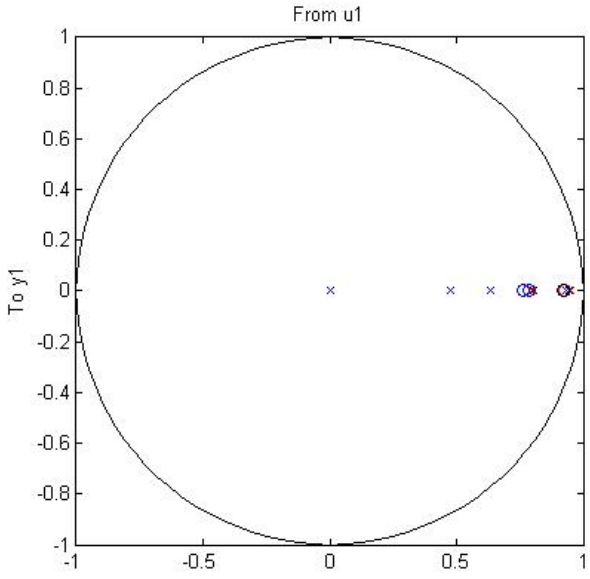


Figure 5.22: Map of the Poles and Zeroes

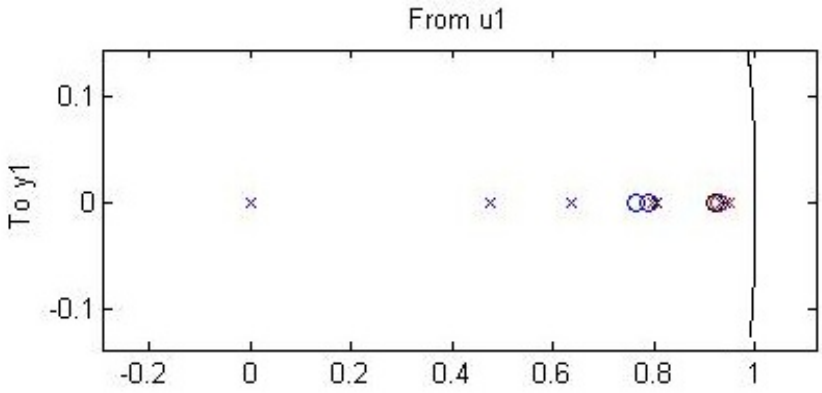


Figure 5.23: Map of the Poles and Zeroes (Zoomed in)

5.7 Model Evaluation and Validation

In the previous section, the ARX, ARMAX, OE and BJ candidates were compared based on their performance on a specific set of validation data. That data was very similar to the data that was used to create those models. In this section the models are again compared but this time it will be against a range of validation data.

5.7.1 Model Response to steps at different levels

Table 5.3 shows the model fit of each model for a range of Step inputs. In each of these tests the input was a step which would start at 0% throttle and go up to the value indicated in the table. These tests were also repeated with fixed step size of 10% and the starting point ranging from 0-90% throttle with similar results.

The data in Table 5.3 shows that the ARX and ARMAX models have the best performance across the operating envelope. It also shows that the model performs much better when operating at above 20% throttle. This is especially noticeable in

the BJ and OE models which have very poor performance below this point.

The similarity in the ARX and ARMAX results can be attributed to the fact that the only difference in their structure is the noise model. The similarity supports the earlier findings that the noise in the system has very little 'colour'. However, it also should be noted that ARMAX with its more complex noise model does consistently predict the output more effectively, even if the difference is very small at times.

Step Amplitude	**Model Fit**			
	ARX	**OE**	**ARMAX**	**Box Jenkins**
10% Throttle	85.78%	6.19%	90.46%	51.47%
20% Throttle	95.65%	87.79%	97.15%	93.64%
30% Throttle	96.31%	92.61%	97.54%	96%
40% Throttle	97.6%	87.81%	98.42%	93.63%
50% Throttle	97.81%	83.38%	98.59%	91.43%
60% Throttle	97.52%	82.23%	98.4%	90.78%
70% Throttle	97.72%	82.08%	98.52%	90.71%
80% Throttle	98.29%	80.33%	98.92%	89.8%
90% Throttle	98.19%	79.79%	98.85%	89.52%
100% Throttle	98.41%	89.75%	98.99%	94.69%

Table 5.3: Model Fit for step inputs

5.7.2 Model response to a chirp signal

The final set of validation data that was used to evaluate each model was a chirp signal. A chirp signal is a sinusoid in which the frequency of the oscillation increases over time. The first 3 seconds of this test can be seen in Figure 5.24. The fit level of each signal is as follows: ARX222 57.53%, OE222 -80.5%, ARMAX2222 72.3% and BJ22222 7.087%.

This final test shows that in the rapidly changing chirp signal, the performance of the BJ and OE models deteriorates rapidly. It also is the one test case in which the ARMAX model pulls ahead of the ARX model in a significant manner.

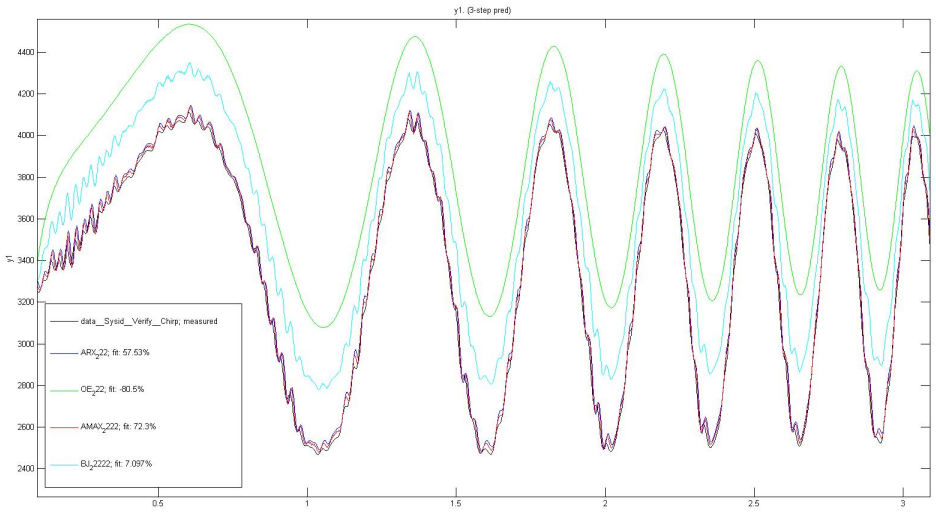


Figure 5.24: Results of the models on the first 3 seconds of a chirp signal

5.8 The Resulting Model

Based on the results of the original validation data, the chirp data and the step inputs data, the ARMAX model was chosen as the model to represent the system. The resulting transfer function is

$$y(z) = \frac{[2.22z^{-2} - 2.046z^{-3}]/30}{1 - 1.744z^{-1} + 0.756z^{-2}}u(z) + \frac{[1 + 1.952z^{-1} + 0.967z^{-2}]}{1 - 1.744z^{-1} + 0.756z^{-2}}e(z) \tag{5.7}$$

This transfer function will relate the throttle input to the motor speed at the output gear. To make the transfer function more meaningful, it can be multiplied by the scaling factor of 1/29.92 [6] to get the output in km/h. This scaling factor will take into account the gears in the transmission and the conversion from angular to linear velocity.

A factor of 1/30 can be seen in the numerator of the system model. Its presence is a result of the data being scaled up during the system identification process to make the results easier to plot along side the inputs.

5.9 Conclusions

In this chapter the RC car was put through a batch of tests to gain a better understanding of the system, its noise level and linear regions. This information was then used in preparing the system for the main identification process. Four candidate models were compared for a variety of different input signals including PRBS, Chirp and Step signals at different voltage levels. The result of this comparison was the decision to accept the ARMAX 2222 model to represent the system throughout the

[6] This scaling factor is calculated in Section A.1 of the appendix

rest of the project. This model will be used to create a Simulink model for the system, the LQR controllers that will control the system and the observers that will assist in monitoring the system's health.

Chapter 6

LQR Controller Design and Kalman Estimator Implementation

6.1 Introduction

This chapter covers the design and implementation of a Linear Quadratic Regulator (LQR) for use in the ACC implementation. The LQR is an optimal controller and can be obtained by solving the Riccati equation to determine the optimal feedback gains to be used. This process requires knowledge of the system's mathematical model and full state feedback. It is on these issues that this chapter will focus.

It would also be useful to note, that in this chapter a lower case variable represents a vector, while an upper case variable represents a matrix.

6.1.1 LQR basics

The LQR is an optimal controller with regards to energy and is capable of handling systems with multiple inputs and outputs. Advantages of optimizing for energy include inherent closed loop stability and a degree of robustness (Hesphanha, 2007).

The control input in an LQR controller is calculated according to the state feedback law $u = -Kx$. The gain matrix K, is the solution to the least-squares optimization problem of minimizing the cost of one of the functions shown in (6.1). The values R, Q and N are positive definite matrices whose value is chosen by the user to tune the system.

$$J = Sum(x^T Qx + u^T Ru + 2 * x^T Nu) \tag{6.1a}$$

$$J = Sum(y^T Qy + u^T Ru + 2 * y^T Nu) \tag{6.1b}$$

Note that (6.1a) is the cost function for a discrete LQR controller with state weighting and (6.1b) is the cost function for a discrete LQR with output weighting (MATLAB, 2012).

6.1.2 Designing LQR Controllers

Without going into the details of the derivation of the LQR controller, it should be stated that the solution to the optimization problem can be found by solving the Riccati equation (6.2a). The solution of this equation S, is used to calculate the LQR matrix gain K according to equation (6.2b).

$$0 = SA + A^T S + C^T C - SBR^{-1}B^T S + Q \tag{6.2a}$$

$$K = R^{-1}B^T S \tag{6.2b}$$

Inherent in this controller is a trade-off between minimizing the control input $u(k)$ and either the internal states $x(k)$ or the system output $y(k)$ depending on whether equation (6.1a) or (6.1b) is used . The input and output penalty matrices, R and Q respectively, provide the designer with a way of managing these trade-offs. The matrices A, B and C are provided by the state space model of the system being controlled. They will be discussed in more detail later.

6.2 Obtaining the System Model

6.2.1 Converting to State Space

Chapter 5 explained how the Transfer Function of the RC car was obtained. At the end of that chapter the following Transfer Function was given.

$$G(z) = \frac{Y(z)}{U(z)} = z^{-2} * \frac{[2.2202 - 2.046z^{-1}]/30}{1 - 1.7445z^{-1} + 0.756z^{-2}} * scale \tag{6.3}$$

This Transfer Function gives the relationship between the input (a number between 0-1000) and the output velocity, which is measured at the wheels. The scaling factor *scale* takes into consideration the gear ratio between the motor and the wheels as well as the unit conversion. Finally, the factor z^{-2} indicates that the system contains a 2 sample delay between the input and output.

The LQR design process calls for the system to be represented in a State Space (SS) representation. To convert the Transfer Function (TF) given in (6.3) into a SS equation, the canonical forms were employed. The process for carrying out this conversion is described in (Ogata, 1987) and the results can be seen in (6.4) and (6.5).

The resulting SS equation in controllable canonical form is

$$\begin{bmatrix} x_{1_{cntl}}(k+1) \\ x_{2_{cntl}}(k+1) \end{bmatrix} = \begin{bmatrix} 1.745 & -0.756 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_{1_{cntl}}(k) \\ x_{2_{cntl}}(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k) \tag{6.4a}$$

$$y_{cntl}(k) = \begin{bmatrix} 0.0740 & -0.0682 \end{bmatrix} \begin{bmatrix} x_{1_{cntl}}(k) \\ x_{2_{cntl}}(k) \end{bmatrix} \tag{6.4b}$$

and in observable canonical form

$$\begin{bmatrix} x_{1_{obs}}(k+1) \\ x_{2_{obs}}(k+1) \end{bmatrix} = \begin{bmatrix} 1.745 & 1 \\ -0.756 & 0 \end{bmatrix} \begin{bmatrix} x_{1_{obs}}(k) \\ x_{2_{obs}}(k) \end{bmatrix} + \begin{bmatrix} 0.0740 \\ -0.0682 \end{bmatrix} u(k) \tag{6.5a}$$

$$y_{obs}(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_{1_{obs}}(k) \\ x_{2_{obs}}(k) \end{bmatrix} \tag{6.5b}$$

Representing the 2 sample delay shown in (6.3) required special attention. The first of these delays was absorbed into the transfer function before converting to state space. This allowed the resulting state-space equations (6.4) and (6.5) to have results in which $y(k)$ doesn't directly rely on the input. This observation will be important to the calculation of the extended system model in the next section.

The other delay z^{-1} must be handled by cascading the SS model with a one sample

unit delay.

6.2.2 Expanding the model

The structure of the 2^{nd} order SS equation will be

$$X_{motor}(k+1) = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x_{1_{motor}}(k) \\ x_{2_{motor}}(k) \end{bmatrix} + \begin{bmatrix} b_1(k) \\ b_2(k) \end{bmatrix} u(k) \tag{6.6a}$$

$$y_{motor}(k) = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} x_{1_{motor}}(k) \\ x_{2_{motor}}(k) \end{bmatrix} + d_1 u(k) \tag{6.6b}$$

The SS equations in (6.4)- (6.5) each contain one output $y(k)$ which will correspond to the velocity of the system measured at the motor.

$$V_{motor}(k) = y_1(k) \tag{6.7}$$

Both the design and operation of the LQR controller require knowledge of the full state of the system. When the system is running, the VSS will provide measurements of the velocity of the system. This measurement will also be passed to a Kalman Observer which will use the value to provide an estimate of the vehicle's position and acceleration.

However, the system model used to design the LQR must contain information about these states and more importantly how they are related. To facilitate this need, two new states were added to the model of the system, both as internal states and as outputs.

The first of these new states is the current position of the system. This value is simple to calculate in SS form since it is the integral of the vehicle's velocity.

$$Velocity: \quad v(k) = c_1 x_1(k) + c_2 x_2(k) + d_1 u(k) \tag{6.8a}$$

$$Position: p(k+1) = p(k) + v(k) * T_s + a(k) T_s^2/2 \tag{6.8b}$$

$$= p(k) + [c_1 x_1(k) + c_2 x_2(k) + d_1 u(k)] T_s + a(k) T_s^2/2 \tag{6.8c}$$

Note that T_s is the sampling time of the system.

The second new state is the acceleration of the system. Calculating this value is a little trickier since it requires knowledge of the past. To handle this, a third extra state could be added to keep track of the velocity of the system at time $k-1$. However, when this approach was attempted, the solver for the Riccati equation ran into problems with the matrix not being full rank.

Another approach is to consider the nature of the SS equation. At each time-step two things happen, the output is calculated and the value of the internal states updated. $v(k)$ will be calculated by the output equation using the values $x_1(k)$ and $x_2(k)$. Therefore, calculation of $v(k+1)$ will require knowledge of $x_1(k+1)$ and $x_2(k+1)$, values which will be calculated by the state-update equation. Equation (6.9) shows how this was done.

Acceleration :

$$a(k+1) = \frac{v(k+1) - v(k)}{T_s} \tag{6.9a}$$

$$v(k) = c_1 x_1(k) + c_2 x_2(k) + d_1 u(k) \tag{6.9b}$$

$$v(k+1) = c_1 x_1(k+1) + c_2 x_2(k+1) + d_1 u(k+1) \tag{6.9c}$$

$$\begin{aligned} v(k+1) = {} & c_1[a_1 x_1(k) + a_2 x_2(k) + b_1 u(k)] \\ & + c_2[a_3 x_1(k) + a_4 x_2(k) + b_2 u(k)] + d_1 u(k+1) \end{aligned} \tag{6.9d}$$

Now that $v(k+1)$ and $v(k)$ are known, the acceleration of the system can be approximated as follows

$$\begin{aligned} v(k+1) = {} & (c_1 a_1 + c_2 a_3) x_1(k) + (c_1 a_2 + c_2 a_4) x_2(k) + \\ & (c_1 b_1 + c_2 b_2) u(k) + d_1 u(k+1) \\ -v(k) = {} & c_1 x_1(k) + c_2 x_2(k) + d_1 u(k) \end{aligned} \tag{6.10a}$$

$$\begin{aligned} a(k+1) = 1/T_s((c_1 a_1 + c_2 a_3 - c_1) x_1(k) + (c_1 a_2 + c_2 a_4 - c_2) x_2(k) + \\ (c_1 b_1 + c_2 b_2 - d_1) u(k) + d_1 u(k+1)) \end{aligned} \tag{6.10b}$$

Taking the results of (6.8c) and (6.10b), the SS equation of (6.6) can be updated to include the new states.

$$x_{LQR}(k) = \begin{bmatrix} a_1 & a_2 & 0 & 0 \\ a_3 & a_4 & 0 & 0 \\ c_1 T_s & c_2 T_s & 1 & T_s^2/2 \\ \frac{c_1 a_1 + c_2 a_3 - c_1}{T_s} & \frac{(c_1 a_2 + c_2 a_4 - c_2)}{T_s} & 0 & 0 \end{bmatrix} x_{LQR}(k) + \begin{bmatrix} b_1 \\ b_2 \\ 0 \\ \frac{c_1 b_1 + c_2 b_2 + d_1}{T_s} \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ d_1 \end{bmatrix} u(k+1) \tag{6.11a}$$

$$y_{LQR}(k) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ c_1 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_{LQR}(k) \tag{6.11b}$$

Substituting in the parameters from (6.4) and (6.5), the solution then becomes.

Controllable Canonical Form

$$x_{LQR_{cntl}(k)} = \begin{bmatrix} 1.744 & -0.756 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0.0740 T_s & -0.0682 T_s & 1 & T_s^2/2 \\ -0.0131/T_s & 0.0123/T_s & 0 & 0 \end{bmatrix} x_{LQR_{cntl}}(k) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0.0740/T_s \end{bmatrix} u(k) \tag{6.12a}$$

$$z_{LQR_{cntl}}(k) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0.0740 & -0.0682 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_{LQR_{cntl}}(k) \tag{6.12b}$$

Observable Canonical Form

$$x_{LQR_{obs}}(k+1) = \begin{bmatrix} 1.744 & 1 & 0 & 0 \\ -0.756 & 0 & 0 & 0 \\ T_s & 0 & 1 & T_s^2/2 \\ -0.744/T_s & 1/T_s & 0 & 0 \end{bmatrix} x_{LQR_{obs}}(k) + \begin{bmatrix} 0.0740 \\ -0.682 \\ 0 \\ 0.0740/T_s \end{bmatrix} u(k) \tag{6.13a}$$

$$y_{LQR_{obs}}(k) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_{LQR_{obs}}(k) \tag{6.13b}$$

In both these equations the output vector $y(k)$ will be $\begin{bmatrix} p(k) & v(k) & a(k) \end{bmatrix}^T$ and the state vector will be $\begin{bmatrix} v(k) & \sim(k) & p(k) & a(k) \end{bmatrix}^T$. Where $\sim(k)$ is a state without physical meaning but is still needed for the calculation of $V(k)$.

At this point a decision was made to use the Observable Canonical Form since its output equation will allow for easier tuning of the controller.

6.3 Designing LQR Controllers for ACC

The LQR controllers for the ACC system were designed in Matlab using equation (6.13) as the model of the system. Matlab provides a command *lqry* which will compute LQR gains using the method described earlier in this chapter.

Since Matlab will take care of solving the Ricatti equation shown in equation (6.2a), the designer has only to tune the parameters to obtain the desired behaviour. Firstly, the values of the input and output penalty matrices R and Q need to be

chosen. Since there isn't a set way to choose these values, they will need to be tuned manually.

The second method the designer has of altering the system is to tune the measurement matrix. The measurement matrix in (6.13) can be modified to reflect this ability as such

$$z(k) = \begin{bmatrix} 0 & 0 & w_1 & 0 \\ w_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_3 \end{bmatrix} x(k) \tag{6.14}$$

By changing the value of any of these weights w relative to the others, the designer can choose to put more importance on a particular state's goal. For example, the LQR controller that will be used for cruising doesn't care what the error in the following distance is but the difference between the cruising velocity and set point velocity will be very important.

The ability to tune the measurement matrix in this way allowed for the design of two different LQR controllers using the same system model. A brief description of each is included below.

Following Subsystem

The LQR controller designed for following behaviour has three states it attempts to minimize. It will attempt to match the lead vehicle's velocity, achieve the driver's desired inter-vehicle spacing, and it will attempt to minimize acceleration.

In this controller the velocity and position data that matter are relative to the lead vehicle. As such this controller makes heavy use of the radar sensor. Matching the lead vehicle's velocity involves minimizing the *relative velocity* which is measured by

the radar unit. On the other hand, achieving the desired following distance requires minimizing the difference between the set-point and the inter-vehicle distance, which is also measured by the radar unit.

The acceleration that is to be minimized is not relative to the lead vehicle and as such its measurement comes from the observer tied to the VSS sensor and not the radar.

Cruising Subsystem

The LQR Controller designed for implementing the cruising behaviour has only two goals that it will attempt to achieve. The first of these goals is to minimize the difference between he vehicle's current velocity and the driver requested cruising velocity. The second goal is once again to minimize the acceleration of the vehicle. Both of these goals require only the sensor data provided by the VSS and its associated observer. As a result, the behaviour of this controller is independent of the radar unit.

Under actuated systems

Note that this system is considered to be "under-actuated". An under-actuated system is one that has more controlled outputs than process inputs (Hesphanha, 2007). A result of this is that the system will have to make temporary trade-offs in its goals. For example when following a vehicle the system may need to temporarily increase its relative velocity so that it can move closer to the lead vehicle and in doing so achieve the desired inter-vehicle spacing. However, once this is accomplished both the velocity and distance criteria will dictate that the system should once again minimize its

relative velocity. These trade-offs can be managed by tuning the measurement matrix and the output penalty matrix Q.

6.4 Obtaining Full State Feedback

Implementing the LQR controller with the calculated gain requires that the full state vector be available for feedback. However, the VSS only directly provides information about the vehicle's velocity. To obtain the acceleration and position data, either additional sensors need to be added or an observer introduced.

For this project the decision was made to use an observer in the interest of reducing the hardware costs. In designing the observer there are a number of options available. The first option is to use the enlarged system model found in (6.13) as the model of the system. The problem with this approach is that (6.13) relies on numerical differentiation, a property which will cause large errors when noise is present in the system. The reason (6.13) was able to be used when designing the system was that those calculations were being done off-line and not with noisy sensor data.

A method for dealing with the unreliable sensor data and the need to observe the full state of the system can be handled together through the use of a Kalman observer. Kalman observers are commonly used in navigation applications. The Kalman filter combines the measurement data with knowledge about the model of the system to determine an estimate of the current state of the system using Bayesian statistics.

In the case of the ACC system, the Kalman filter was used to combine the knowledge of how the states are related (by the kinematic equations) with the sensor data. It was used to increase the reliability of the radar data which estimates the relative distance and velocity. Since the relative velocity is a function of the change in the

relative distance, these two values can not change independently. Sensor readings that would suggest otherwise are likely unreliable.

However, the main use of the Kalman observer was to extract information about the vehicle's position and acceleration from the data provided by the VSS. A variation of the Triple Integrator Form discussed in (Canet, 1994) was used and can be seen in (6.15). The difference between the version shown below and the one appearing in (Canet, 1994) is that (Canet, 1994)'s uses measurement data of the system's position while the one below uses measured velocity data.

$$X_{Kalman}(kT+T) = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} X_{Kalman}(kT) + w(kT) \tag{6.15a}$$

$$Y_{Kalman}(kT) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} X_{Kalman}(kT) + e(kT) \tag{6.15b}$$

6.5 Summary

In this chapter a set of Linear Quadratic Regulators was designed to implement the continuous control aspect of ACC. The problem of providing full-state feedback was handled for both the design and implementation of the LQR controllers. By tuning the output and penalty matrices the desired behaviour for continuous control was obtained for both the cruising controller and the following controller.

These controllers make up an important part of the design presented in Chapter 3. In the next chapter the resulting system will be examined using Hazard Analysis techniques.

Chapter 7

Identifying the Faults

7.1 Introduction

Hazard Analysis techniques are used to gain information about the potential hazards and faults that can occur within a system. Their use is critical to the effort of building safety critical systems like Adaptive Cruise Control. Of special importance for a safety critical system is the need to identify the conditions under which the system's operation can result in harm to humans or damage to the environment in which it operates.

7.2 Fault Tree Analysis

7.2.1 Intro

One of the techniques that was used to analyse the ACC system is known as Fault Tree Analysis or FTA . This technique was applied at the beginning of the project

to gain a better understanding of the potential faults that needed to be considered during the design process.

Fault Tree Analysis is based upon the "Chain-of-Events" accident causation model. The "Chain-of-Events" model holds that accidents can be explained by a sequential chain of events initiated by component failure or human error (Leveson, 2009).

A very simple FTA notation was used for the analysis in this thesis. Figure 7.1 shows the legend for each of the four symbols and their purpose. The transfer block, represented by a pair of triangles, can be used to continue a branch of the tree in another location. This functionality can be quite useful when creating large fault trees or fault trees where a branch is repeated in the tree.

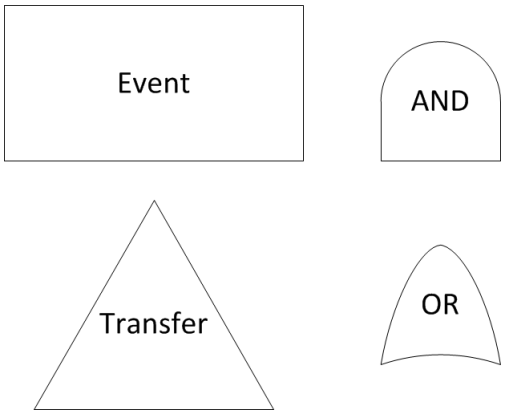


Figure 7.1: Legend for FTA shapes

Analysis in FTA is an iterative, top-down process. To begin, the hazard/scenario whose causes are being investigated, is placed at the top of the tree. Then the events/scenarios that are immediate causes for that event are placed in the second row and connected to the root via the boolean gates. This process is then repeated for each of the situations that are now at the bottom of the tree.

The top level hazard used to start the analysis for the ACC system is "Gap

between vehicles is incorrect" which immediately led to two situations "Gap between vehicles is too small" and "Gap between vehicles is too large". To be able to represent this tree, it was broken into two sub trees as seen in Figures 7.2 and 7.3.

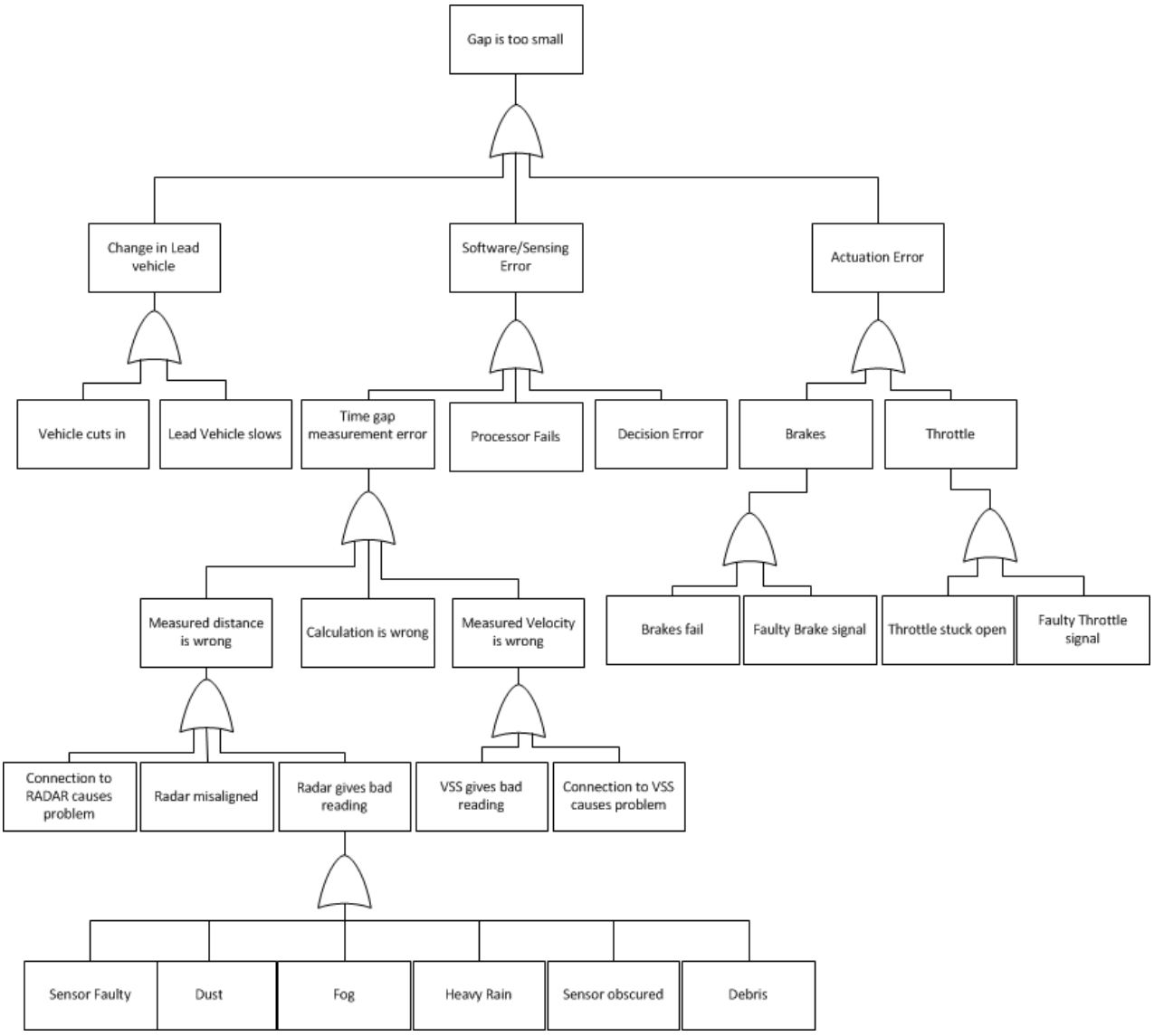


Figure 7.2: Fault tree analysis on case "Gap too Small"

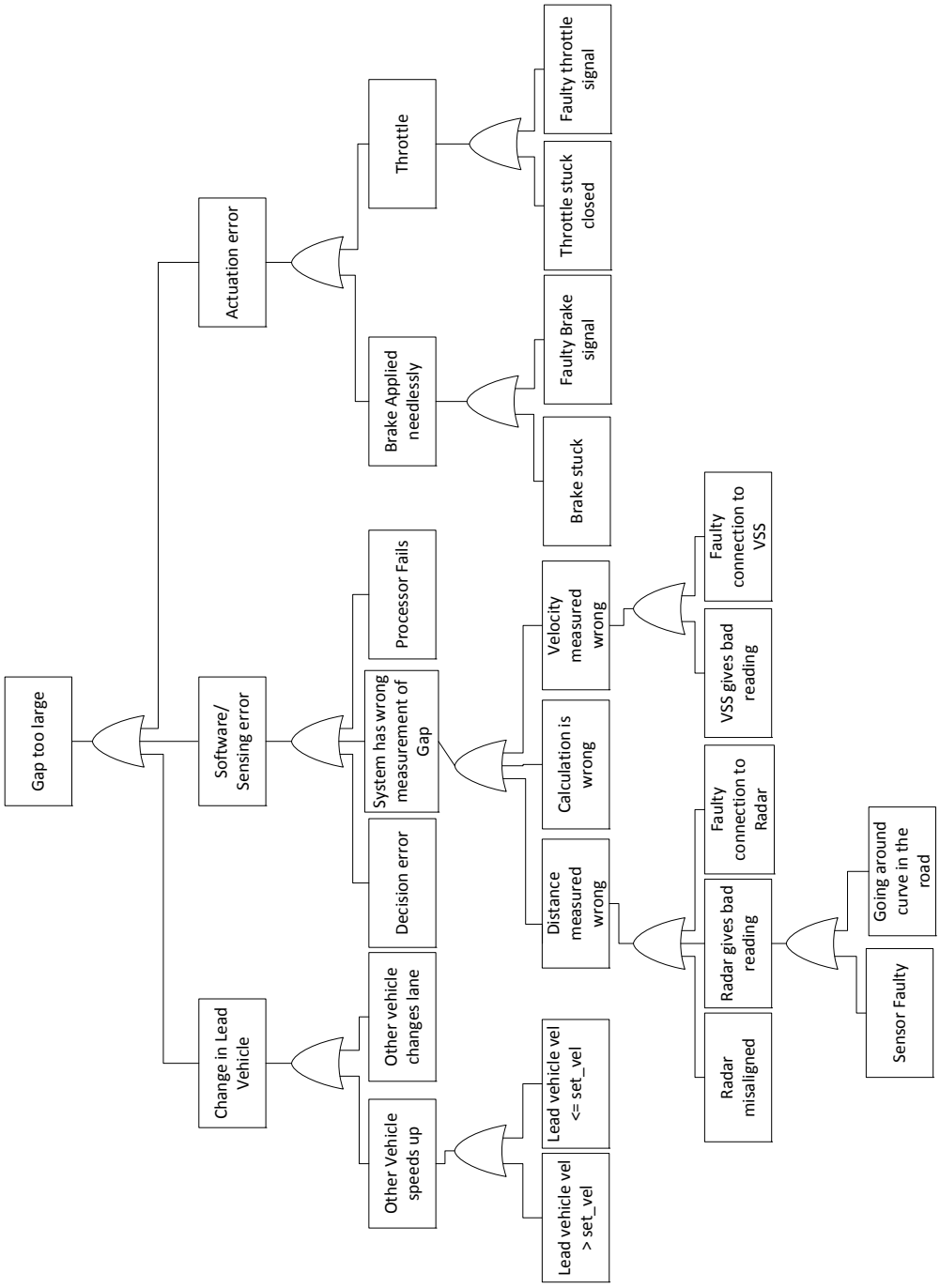


Figure 7.3: Fault Tree Analysis on case "Gap too Large"

7.3 System-Theoretic Process Analysis

7.3.1 Intro

After the preliminary system design had been developed, an alternative to FTA was applied to the system. This new technique is known as the System-Theoretic Process Analysis or STPA. Although, this technique was was only applied to the design at the end of the project it should be noted that STPA contains provisions for examining a design throughout its life cycle. A full comparison between FTA and STPA is beyond the scope of this thesis but it is beneficial to highlight a few key areas before going ahead with its application.

STPA is based off of the System-Theoretic Accident Modelling and Processes also known as STAMP (Leveson, 2009) . Of special importance to this project is the way that STAMP views safety as both

A) A System/Emergent property

B) A control problem.

The first point refers to the idea that the safety of a system can't be inferred by proving the safety of its individual components. Put another way, safety is an emergent property, one that depends on the interaction of the components involved and not just the behaviour of the individual components.

The second point involves a bit of a paradigm shift from the classic "Chain-of-events" model. STAMP claims that accidents should not be viewed as the result of a failure. Instead an accident in STAMP is regarded as the result of inadequate control

actions. The effects of this view will become quite evident during Step 1 of the STPA process.

For more information about the STAMP accident model and STPA, readers are encouraged to consult either (Leveson, 2009) or (Song, 2012).

7.3.2 Getting started

The STPA process is broken down into a two step process. The first step is to identify the inadequate control actions that could lead to the system being in a hazardous state. The second step is then to examine the system to determine how these control actions could arise. However, there are several things that need to be taken care of before the process can begin. This includes identifying the accidents, the system level hazards and system level safety requirements. It also requires developing a model of the system's control structure that will be used to analyse where the responsibility and risk lie within the system.

Defining the Accidents and Hazards

The process begins by identifying the "accidents" or loss events that must be avoided. In STAMP an accident is defined as

> An undesired or unplanned event that results in a loss, including loss of human life or human injury, property damage, environmental pollution, mission loss, etc. (Leveson, 2009)

The list of possible accidents that were identified for ACC can be found in Table 7.1

Once the stakeholders have agreed upon the list of accidents that must be avoided, the next step is to determine the associated hazards. In STAMP a hazard is defined

AC 1: Collision with lead car causing harm/damage to occupants/vehicles
AC 2: Vehicle is hit from behind resulting in harm/damage to the occupants/vehicles
AC 3: Occupants/Vehicle harmed/damaged but not because of a collision
AC 4: Damage/Harm inflicted on operating environment

Table 7.1: Accidents/ Unacceptable Losses for Adaptive Cruise Control

as

> "A system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (Leveson, 2009)"

In short, the term "accident" refers to the undesirable events that must be avoided and "hazard" refers to the set of conditions under which the system is susceptible to such accidents. Note that implicit in the definition of a hazard, is the assumption that they lie within the system's boundaries. This is necessary since as (Leveson, 2009) points out, designers can only be responsible for eliminating/controlling hazards that are within their design space. Table 7.3 shows the list of hazards identified for the ACC system. The second column of this table shows which accidents from Table 7.1 each hazard is considered to be in danger of contributing towards.

Hazard	**Corresponding Accident**
H1. Vehicle fails to leave enough of a gap	AC 1
H2. Vehicle accelerates/decelerates unexpectedly	AC 2
H3. Vehicle accelerates/decelerates rapidly	AC 2, AC 3
H4. Vehicle obtains unsafe velocity	AC 1, AC 4
H5. Driver confusion about systems operating mode	AC 1, AC 2 AC 3

Table 7.2: System Level Hazards for Adaptive Cruise Control

Hazard	Safety Constraints
H1. Vehicle fails to leave enough of a gap	System must leave a headway of at least X_1 seconds when active
H2. Unexpected deceleration	System must signal when undergoing deceleration of more than X_2 m/s^2
	System must activate brake lights when brakes are being applied
H3. Rapid acceleration/deceleration	Acceleration of system must not exceed X_3 m/s^2
	Deceleration of system must not exceed X_4 m/s^2
	Braking must be limited to X_5% of total braking power
H4. Vehicle obtains unsafe velocity	The vehicle must not exceed the velocity set point given by the driver by a tolerance of more than X_7%
H5. Driver confusion about systems operating mode	System must keep the driver informed about its current mode of operation (Active, Suspended, OFF) as well as current velocity

Table 7.3: High-Level Safety Constraints

Now that the high level/system hazards have been identified, they can be used to generate safety constraints that will be included as part of the design requirements. Note that just as the hazards that are being used here are system-level hazards, so also the constraints that are produced from them will be system-level. The later steps of STPA will help to refine these constraints and to assign the responsibility of enforcing them to individual components. Table 7.2 shows the list of System-Level Safety Constraints developed in response to the system hazards identified in the ACC system.

Defining the control structure

The control structure of the system is a functional decomposition of the system and can be represented by a component diagram. The boxes represent the components/-subsystems and the directed edges represent the control instructions and feedback information that flows between them.

Starting with the very simple and highly abstracted view of the system shown in Figure 7.4, the subsystems can be expanded to include the desired amount of detail about the system. In the case of the ACC system, emphasis is given to the controller and its internal structure as shown in Figure 7.5. These diagrams are important because they will serve as the working models for the STPA process.

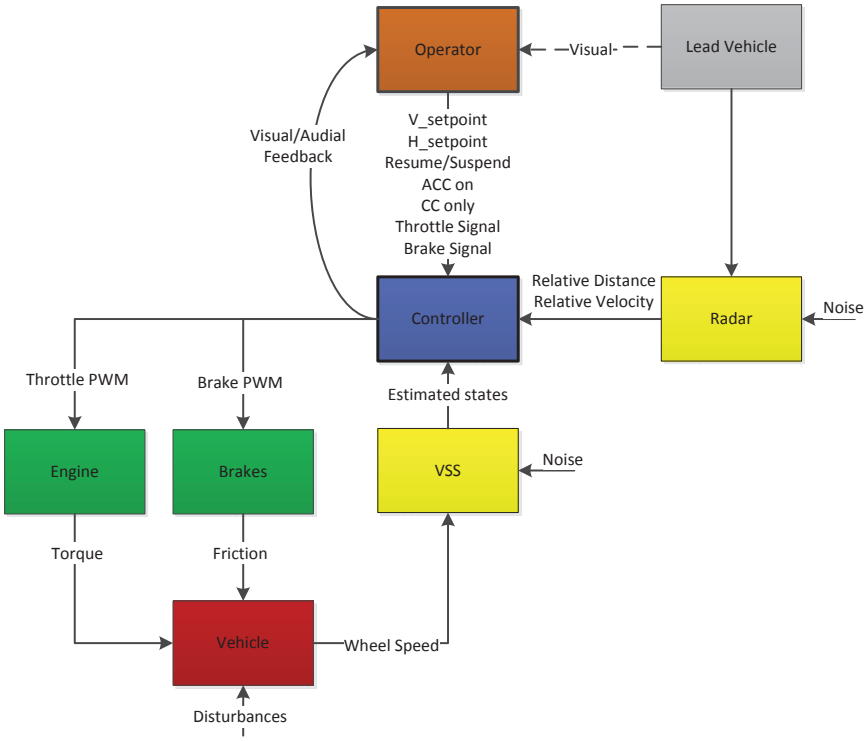


Figure 7.4: Top Level Control Structure

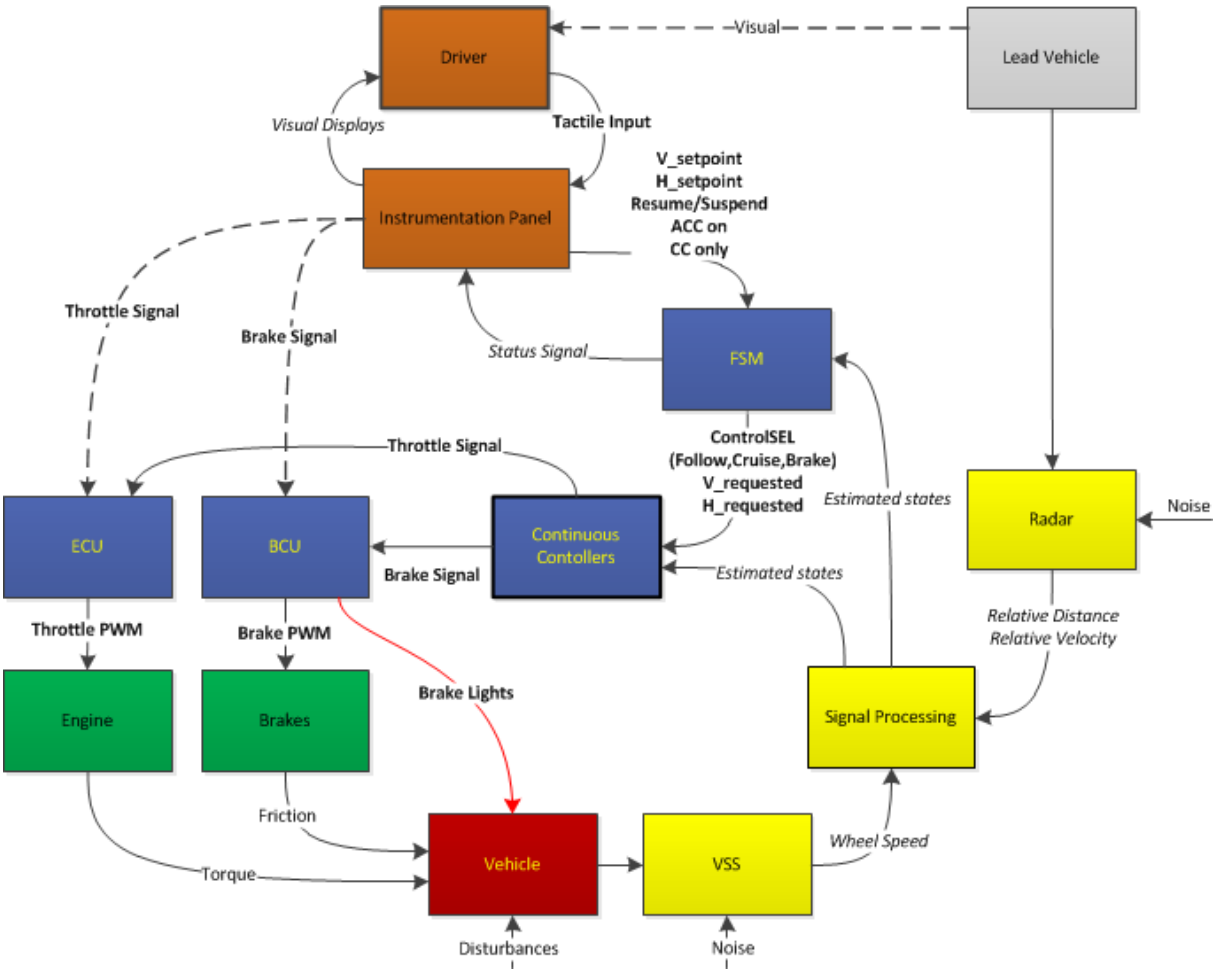


Figure 7.5: Full/Expanded Control Structure

7.3.3 STPA Step 1

As previously mentioned, STPA views safety as a control problem and analysis on the system is carried out in a top-down manner. The process begins with a high-level model of the system (the control structure), the list of system hazards and system safety requirements/constraints. Once these have been prepared, the list of hazards is refined by examining the control structure to determine what inadequate control signals could result in the hazardous states identified earlier.
STPA provides four categories of inadequate control, these are (Leveson, 2009):

a. "A required control action not provided or not followed"

b. "An incorrect or unsafe control action is provided"

c. "A potentially safe control action is provided too early or too late, that is, at the wrong time or in the wrong sequence"

d. "A correct control action is stopped too soon"

Analysis begins by taking each command action in the system and considering the effect of it being inadequate in any of the four ways listed above. The goal of this process is to identify the scenarios that result in inadequate control so that the next step of STPA can investigate how they occur.

This process was carried out on the ACC system and the results stored in a table. An excerpt of these results can be found in Figure 7.6 below. The full table can be found in the Appendix .

Results of the STPA Step 1

Control Action	Missing or Not Provided	Provided Incorrectly	Wrong Timing/Order	Stopped too soon
Apply Brakes	The vehicle will fail to activate the brakes. **(H1)** As a result the vehicle will be moving too quickly for current conditions **(H4)**	The vehicle will activate the brakes unexpectedly **(H1)**	Late: Brakes won't be activated on time **(H1)**	Braking will subside prematurely. The vehicle will still be travelling too fast for the current conditions. **(H1)** **(H4)**
Activate the controller "*Active Brake Controller*" (ABC)	The system will fail to activate the braking system **(H1)**	The system will activate the brake controller.	Late: Vehicle wont activate the brakes on time **(H1)**	Brakes will be deactivated prematurely. The velocity will still be too large/gap too small. **(H1)** **(H4)**
		The brake controller *may in turn activate the brakes, resulting in unexpected braking **(H2)** **(H3)**(H5)	Late: Rapid deceleration may be required by to prevent colliding with LV. (Risk being rear-ended) **(H3)**	

Figure 7.6: Excerpt from STPA step 1

Identification of the potentially inadequate control actions within the system allows for the development of safety constraints. The list of safety constraints that were developed from this stage of STPA are listed in Table A.1 in the Appendix.

7.3.4 STPA Step 2

Once each command action has been analysed, the ones that are determined to be a potential safety risk undergo another stage of analysis. In this stage the control structure is again examined, but this time the goal is to determine the causal factors of such command actions being issued.

This process itself has two steps: firstly each controller's process models must be identified; secondly these process models are used in conjunction with the control structure to identify the causal factors of inadequate control.

A controller's process model is a representation of how the controller views the system. If this process model does not match the physical system, then the controller may end up issuing command actions that do more harm than good. Take for example the case where a faulty sensor results in the controller's estimate of the vehicle's velocity being slower than it really is. If the vehicle is currently "cruising", then the result of this discrepancy would be the controller causing the vehicle to exceed the set point velocity.

It is also important to note that both technical and human controllers have an internal process model from which they make control decisions. A human driver that doesn't properly understand the system, or operates it under false information, is at risk of causing an accident. This is the reasoning behind the inclusion of hazard (**H5**). The process models for the controllers in the ACC system are shown in Table

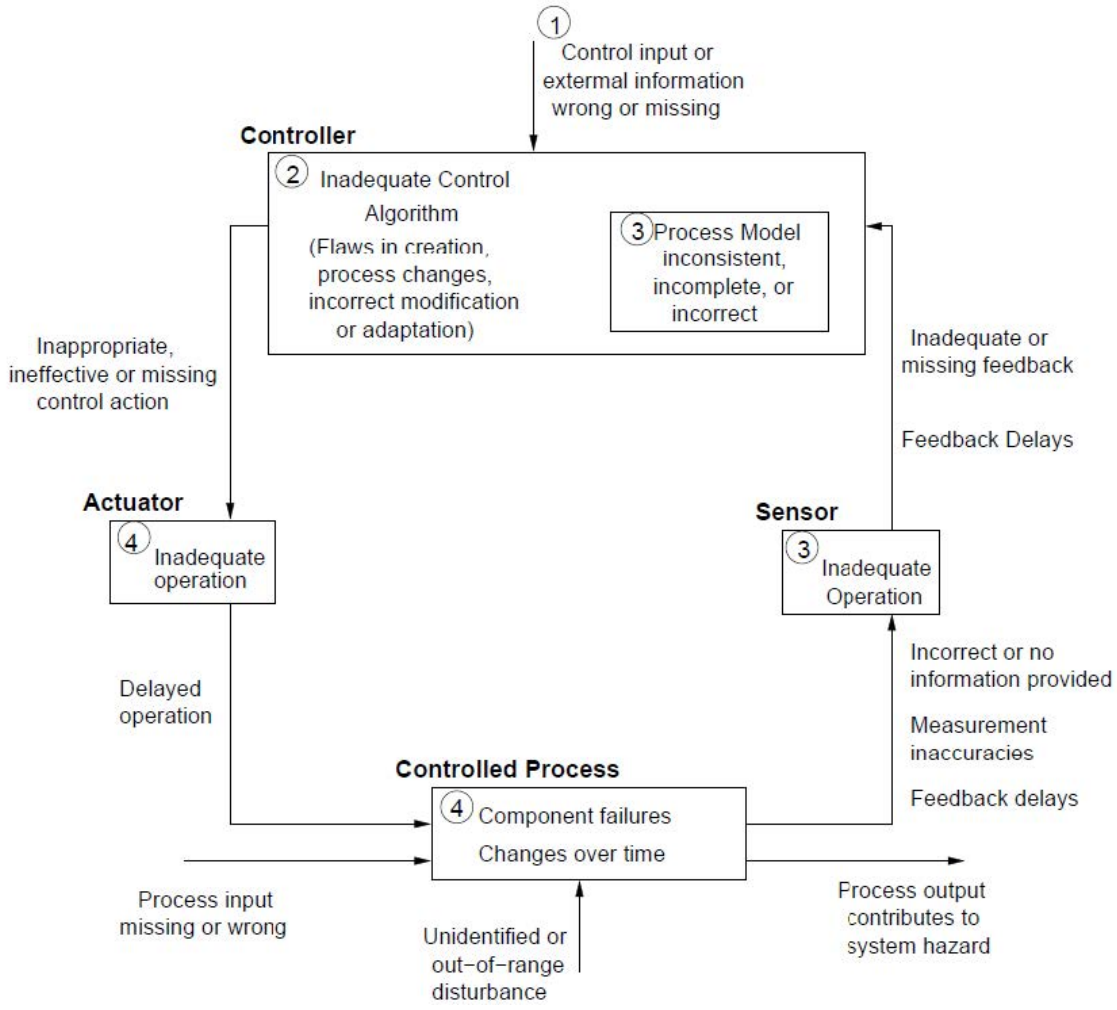


Figure 7.7: Causal factors for inadequate control (Leveson, 2009)

7.4.

Controller	Property	Possible Values				
FSM	Velocity	Too fast	Acceptable	Too slow	Unknown	
	Leader Velocity	Faster	w/in Tol	Slower	Unknown	
	Following distance	Zone A	Zone B	...	Zone E	Unknown
	ACC Status	ACC_{Active}	CC_{Active}	Suspended	Off	
	Internal Mode	Cruise	Follow	Brake	Off	
Driver	Velocity	Too fast	Acceptable	Too slow	Unknown	
	Leader Velocity	Faster	w/in Tol	Slower	Unknown	
	Following distance	Too close	Acceptable	Too far	Unknown	
	ACC Status	ACC_{Active}	CC_{Active}	Suspended	Off	Unknown
	Mode	Following	Traditional	Inactive	Unknown	
CCC	Following Distance	Too close	Acceptable	Too far	Unknown	
	Velocity	Too fast	Acceptable	Too slow	Unknown	
	Leader Velocity	Faster	w/in Tol	Slower	Unknown	
	Active Controller	Brake	Follow	Cruise		
Brake	Following Distance	Too close	Acceptable	Too far	Unknown	
	Velocity	Too fast	Slow enough	Unknown		
Follow	Following Distance	Too close	Acceptable	Too far	Unknown	
	Velocity	Too fast	Acceptable	Too slow	Unknown	
	Leader Velocity	Faster	w/in Tol	Slower	Unknown	
Cruise	Velocity	Too fast	Acceptable	Too slow	Unknown	

Table 7.4: Process Models For ACC system

For the ACC brake system, the causal analysis was first attempted directly on the full control structure. For this process Figure 7.7 was used as a guideline of what types of causal factors might arise in each part of the control structure. However, this was done without lumping together the different components as either an “actuator”, “controller”, “controlled process” or “sensor”. The result of this process can be seen in Figure A.5 and then again in the form of a chart. Both are located in the Appendix.

It quickly became clear that this method of analysis would result in diagrams that were burdensome to create due to the resulting size and clutter. However, the resulting diagram did allow for an intuitive mapping of faults to individual components.

Following this, the causal analysis for the brake system was attempted a second time. This time the only components that were involved in the control action's control loop were included. Additionally, each component's role in that specific control loop was classified as one of the following 4 categories: the controller that issues the control action, the actuator that carries out the control action, the controlled process that is being controlled via the control action, and the sensor that monitors the system to provide feedback about the effectiveness of the control action.

Once this was done Figure 7.7 was used as a guide to identify each of the potential sources of this inadequate control action. Representing the system in a similar manner to Figure 7.7 made it much easier to use Figure 7.7 as a guide. The resulting diagrams were much cleaner and the analysis was simpler to carry out. The disadvantage is that extra care had to be taken to consider which components were represented by each block. For example, the "Sensor" block included the Encoder, the Radar, the filters and the observer.

Figure A.6 and Figure A.7 show the two diagrams that carry out the equivalent analysis to Figure A.5. Two diagrams were needed here since there are two stages of control (The FSM, and the CCC). In Figure A.6 the CCC is the controller while in Figure A.7 it is considered to be part of the actuator system.

By looking at the result of these two methods, it was eventually determined that the analysis for the remaining control actions would be done according to the second method. This decision was heavily influenced by the fact that the second method resulted in diagrams that were much cleaner, easier to read and as a result easier to create.

This process was repeated for each inadequate control signal that could lead to a safety concern. During this process it became apparent that many of the fault that were identified were common to multiple diagrams. To be able to present a clear, unified list of faults out of the result of this analysis Table A.2 was created. The disadvantage this representation had over the diagrams is that it removes the context. For example it does not show what inadequate control action "Radar Blocked" can lead to. The Table can be found in the Appendix on page 155.

This information about potential faults is useful when preparing maintenance schedules, choosing sensors and identifying faults that should be identifiable by the fault diagnosis system.

7.4 Conclusion

Hazard Analysis techniques are essential tools in the design and construction of safety critical systems. In this chapter the Hybrid Systems Adaptive Cruise Control was initially subjected to Fault Tree Analysis. Following that a more comprehensive analysis of the system was carried out using STPA. The result of this process is a better understanding of the system and its vulnerability as well as a better idea how to mitigate these vulnerabilities. Special attention was given to the possibility of inadequate control actions and the faults that could cause them. This information was used to create safety constraints that the system must enforce to remain in a safe state along with a list of faults that should prove useful when designing the fault diagnosis system.

Chapter 8

Conclusion

In this thesis a model of a Hybrid Systems Adaptive Cruise Controller was developed and implemented via Matlab's Simulink. This model contained both a mathematical representation of the physical system being controlled as well as a realization of the ACC controller. The controller consists of a set of LQR controllers and a Finite State Automata. The individual LQR controllers implement different aspects of the ACC system's behaviour like cruising or following. The use of a Finite State Automata allows for switching between these LQR controllers according to the current traffic situation.

STPA was also used to perform a detailed hazards analysis of the resulting controller. Its results can be used to gain a better understanding of what faults a diagnosis system would need to be able to identify. Additionally, the mathematical representation of the system that was used to develop the LQR controllers is provided. It can be used to verify the system's results through simulation or incorporated into the design of fault diagnosis tools like Residual Generators.

Appendix A

Appendix

A.1 Calculation of the Scaling Factor

The following discussion will explain how the scaling factor was obtained for Section 5.8

$$y(z) = \frac{2.22z^{-2} - 2.046z^{-3}}{1 - 1.744z^{-1} + 0.756z^{-2}}u(z) + \frac{1 + 1.952z^{-1} + 0.967z^{-2}}{1 - 1.744z^{-1} + 0.756z^{-2}}e(z) \qquad \text{(A.1)}$$

The output of the system from the original transfer function can be converted into RPMs by the following formula

$$Output \times \frac{SamplingRate \times 60}{1000} = Output \times \frac{333 \times 60}{1000} = Output\ in\ RPM \qquad \text{(A.2)}$$

The next thing to consider is that the angular velocity of the car is measured at the output gear of the motor. To obtain the angular velocity of the wheels the gear ratio

of 7.16 must be taken into account.

$$Output\ at\ Wheels = \frac{1}{7.16} \times Output\ at\ Motor \tag{A.3}$$

The next step is to convert the angular velocity given in RPM into m/s

$$RPM = \frac{Revolutions}{Minute} \times \frac{\pi \times diameter}{60} = RPM \times \frac{\pi \times 63.55 * 10^{-3}}{60} = m/s \tag{A.4}$$

So to convert angular velocity measured at the motor's output gear into the velocity of the car measured in m/s a scaling factor of $\frac{1}{107.7}$ can be used. Alternatively if the desired output is to be measured in km/h then the scaling factor would be $\frac{1}{29.92}$

A.2 Model Structures in System Identification

Chapter 5 discussed the identification of an electric motor using the system identification technique. The form of system identification that was used in Chapter 5 is known as Parametric Identification (Habibi, 2011a). In this approach a set of general models are used and adapted to the particular system by identifying the parameters.

To aid in this process four transfer function models were used and their parameters identified. These are the ARX, the ARMAX, the Output Error and the Box Jenkins model. A short description of each is included below.

A.2.1 Auto-Regressive Model with an eXogenous Input Model (ARX)

The ARX Model takes the form of a simple difference equation shown in equation (A.5) (Johansson, 1993). In this equation A and B are polynomials which are determined using "prediction error methods" (Johansson, 1993).

$$A(z^{-1})y_k = z^{-d}B(z^{-1})u_k + w_k \tag{A.5}$$

This structure of this model can be seen in Figure A.1 (Habibi, 2011a). The two components to this model are the system model $\frac{B(z)}{A(z)}$ and the noise model $\frac{1}{A(z)}$.

According to (Habibi, 2011a), the main advantage of ARX is that "it can be found analytically without numerical optimization and iterations". Its disadvantage is a highly restricted noise model.

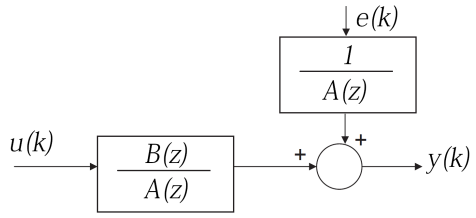


Figure A.1: Structure of an ARX model

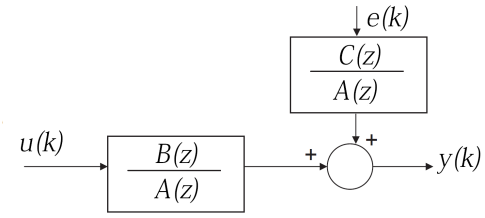


Figure A.2: Structure of an ARMAX model

A.2.2 Auto-Regressive-Moving-Average Model with an eXogenous Input Model (ARMAX)

The ARMAX model expands the ARX model by increasing the flexibility of the noise model. Like the ARX model, the ARMAX model takes the form of a difference equation as can be seen in equation (A.6)

$$A(z^{-1})y_k = z^{-d}B(z^{-1})u_k + C(z^{-1})w_k \tag{A.6}$$

Once again A, B and C are polynomials that are determined by the "prediction error method" of data fitting (Johansson, 1993). The structure of the ARMAX model can be seen in Figure A.2 (Habibi, 2011a).

A.2.3 Box Jenkins Model (BJ) and Output Error (OE)

The Box Jenkins model and the Output Error model are transfer function models (Johansson, 1993). Unlike the difference equations used to describe the ARX and ARMAX, the noise models in these systems are independent of the system model.

The Box Jenkins models the disturbances in the system as "a white noise sequence w_k filtered through the transfer function C/D (Johansson, 1993)". It also contains the most flexible noise model of any of the model structures discussed in this thesis.

$$y_k = \frac{B(z^{-1})}{F(z^{-1})}u_k + \frac{C(z^{-1})}{D(z^{-1})}w_k \tag{A.7}$$

The Output-Error Model on the other hand makes no assumptions about the disturbances in the system (Johansson, 1993). It can be described by the transfer

function

$$y_k = \frac{B(z^{-1})}{F(z^{-1})}u_k + v_k \tag{A.8}$$

The structures of the Box Jenkins and Output Error models can be seen in Figures A.3 (Habibi, 2011a) and A.4 (Habibi, 2011a) respectively.

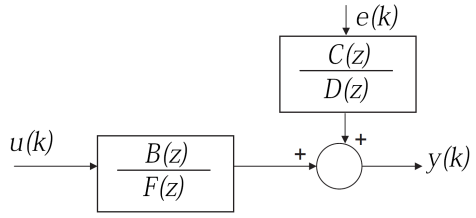


Figure A.3: Structure of a Box Jenkin model

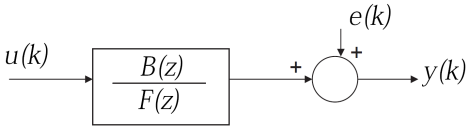


Figure A.4: Structure of an OE model

A.3 Results from STPA Step 1

Results of Step 1 in STPA				
Control Action	**Missing or Not Provided Causes Hazard**	**Provided Incorrectly Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopped too soon Causes Hazard**
Apply Brakes	The vehicle fails to activate brakes when too close to a leading vehicle**(H1)(S1)**	The vehicle brakes unexpectedly **(H2) (S2)(S3)**	The vehicle fails to start braking until it is too close to the leading vehicle **(H1).** System may attempt to use heavy braking to make up for lost time **(H3)(S1)(S3)**	The vehicle stops braking when it is still too close to the leading vehicle **(H1)(S1)**
	The vehicle fails to brake when the velocity is greater than the set point**(H4)**		Vehicle is rear ended because braking started before the brake lights come on. **(H2)(S4)**	The Vehicle stops braking when it is still travelling faster than the set point **(H4)**

Activate the controller "*Active Brake Controller*" (ABC)	The brake system is not activated when following too close to the lead vehicle. Leads to brakes not being applied **(H1)** (cf "Apply Brakes") **(S5)(S1)**	Accidental activation of ABC leads to incorrect & unexpected braking (cf "Apply Brakes") **(S6)(S2)**	Activating the brake controller too soon results in unexpected braking **(H2)(S6)(S2)** (cf "Apply Brakes")	Brake controller will be deactivated prematurely leading to the brakes being deactivated prematurely. **(H1) (H4)** (cf "Apply Brakes") **(S6)(S2)**
		The system leaves the mode it was supposed to be operating in (following or cruise) without notifying the driver**(H5)**	Activating the brake controller too late results in brakes not being applied when needed. (cf "Apply Brakes") **(S6) (S2)**	The system deactivates *ABC* without activating a different controller and without alerting the user that the system is not currently controlling the vehicle**(H5)**

Activate the controller "*Cruise*" /TCC	Vehicle reaches an unsafe velocity because it is attempting to follow a vehicle that is moving too fast **(H4)(S8)(S10)(S12)**	Cruise activated when slower moving vehicle is in its path **(H1)(H5)(S11)**		The system deactivates "cruise" without activating a different controller and without alerting the user that the system is not currently controlling the vehicle**(H5)**
Activate the controller "*Follow*"/CFS	Vehicle continues to use "Cruise" when approaching a slower moving vehicle**(H1)(S11)**	System attempts to follow a vehicle moving faster than the set point and achieves an unsafe velocity **(H4)(S8)(S10)(S12)**	The system gets too close to the leader before it attempt to follow**(H1)(S11)**	The CFS is deactivated when following another car and as a result gets too close to the leader **(H1)(S11)(S14)**

		The system attempts to follow a vehicle that is too far away. Causing the LQR controller to attempt to minimize a very large error. Could result in large velocities/accelerations **(H3)(H4)(S9)(S10)(S12)**	Late activation of controller results in system instability as it attempts to make up for lost time. For example by the time it realizes it should have been following it is now time to brake **(S15)**	The system deactivates "*follow*" without activating a different controller and without alerting the user that the system is not currently controlling the vehicle **(H5)**

Open Throttle	The vehicle will decelerate unexpectedly (**H2b**) and will fall below its velocity set point	NA	Delay between when the throttle is requested and activated results in system instability	
Maintain				
increase	The vehicle does not accelerate when commanded by the user to avoid a dangerous situation **(S36)**	The vehicle moves too close to the leader (**H1**)**(S16)**		
–		Vehicle accelerates from the desired speed to an unsafe velocity **(H4)**		

–		Vehicle experiences runaway acceleration **(H2a)**		
decrease	The vehicle fails to slow down when following approaching a slower moving vehicle (**H1**) **(H4)(S16)**	The vehicle will decelerate unexpectedly (**H2b**)		The vehicle fails to slow down enough when approaching a slower moving vehicle **(H1) (H4)(S16)**

Driver *"Activates"* ACC	The driver thinks the system is in control and removes foot from accelerator leading to unexpected but gradual vehicle deceleration **(H5)**	System "Activates" and begins to control the system at the same time as the driver **(H5)** (Controller coordination problem) **(S30)(S32)**		N/A
	The driver thinks the system is in control and does not apply the brakes its needed **(H1) (H5) (S32)**		The driver thinks the system is in control and does not apply the brakes its needed **(H1) (H5) (S32) (S34)**	

Driver "*De-activates*" ACC	The ACC system does not return control to the driver **(H5)(S31)**	The ACC system deactivates unexpectedly and does not brake when needed (**H1**)(**H5**)(**S32**) (**S34**)	ACC delays in returning control to the driver **(H5)(S31a)**	N/A
Driver "*Resumes*" ACC	The driver thinks the system is in control and removes foot from accelerator leading to unexpected but gradual vehicle deceleration **(H5)**	System "Resumes" and begins to control the system at the same time as the driver **(H5)** (Controller coordination prob-lem)**(S30)(S32)**		N/A

	The driver thinks the system is in control and does not apply the brakes when the leader applies its brakes **(H1)(H5)(S32)(S34)**			
Driver "*Suspends*" ACC	The ACC system does not return control to the driver **(H5)(S31)**	The ACC system deactivates unexpectedly and does not brake when needed **(H1)(H5)(S32)(S35)**	ACC delays in returning control to the driver **(H5)(S31a)**	N/A

"*Cruise Control Only*" selected by driver	System use full ACC functionality, including brakes and following procedures while driver expects a traditional CC **(H2)(H5)(S33)**	When approaching a lead vehicle it does not use brakes or following procedures **(H1)(H5)(S33)**		N/A
Brake lights activated	The vehicle's brake lights do not turn on to indicate to follower that it is braking **(H2)(S4)**	No hazard	The brake lights do not turn on with the brakes **(H2)**	The brake lights will turn off while the vehicle is still slowing

A.4 Results from STPA Step 2

A.4.1 Diagrams

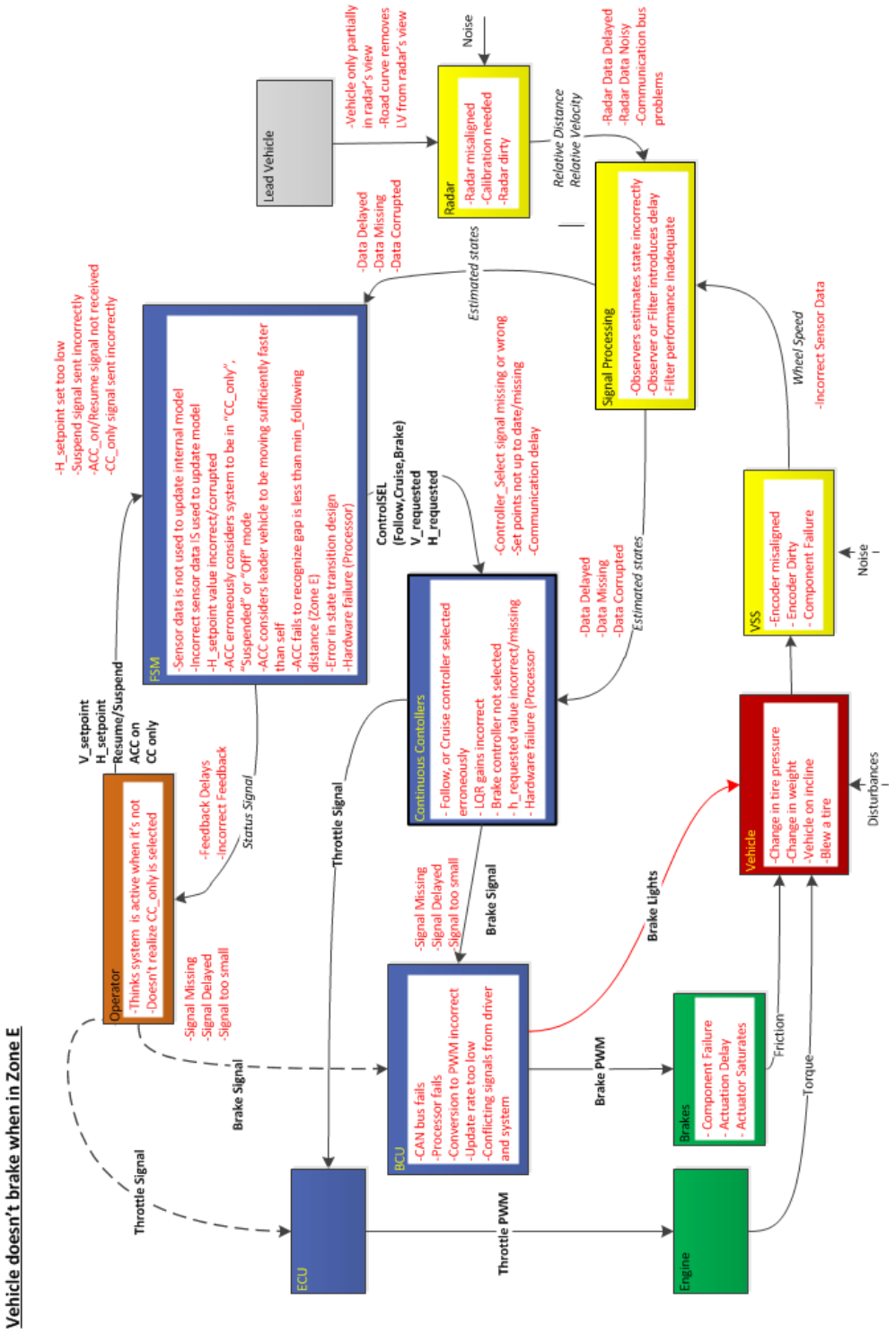


Figure A.5: Results of STPA Step 2 - Method 1

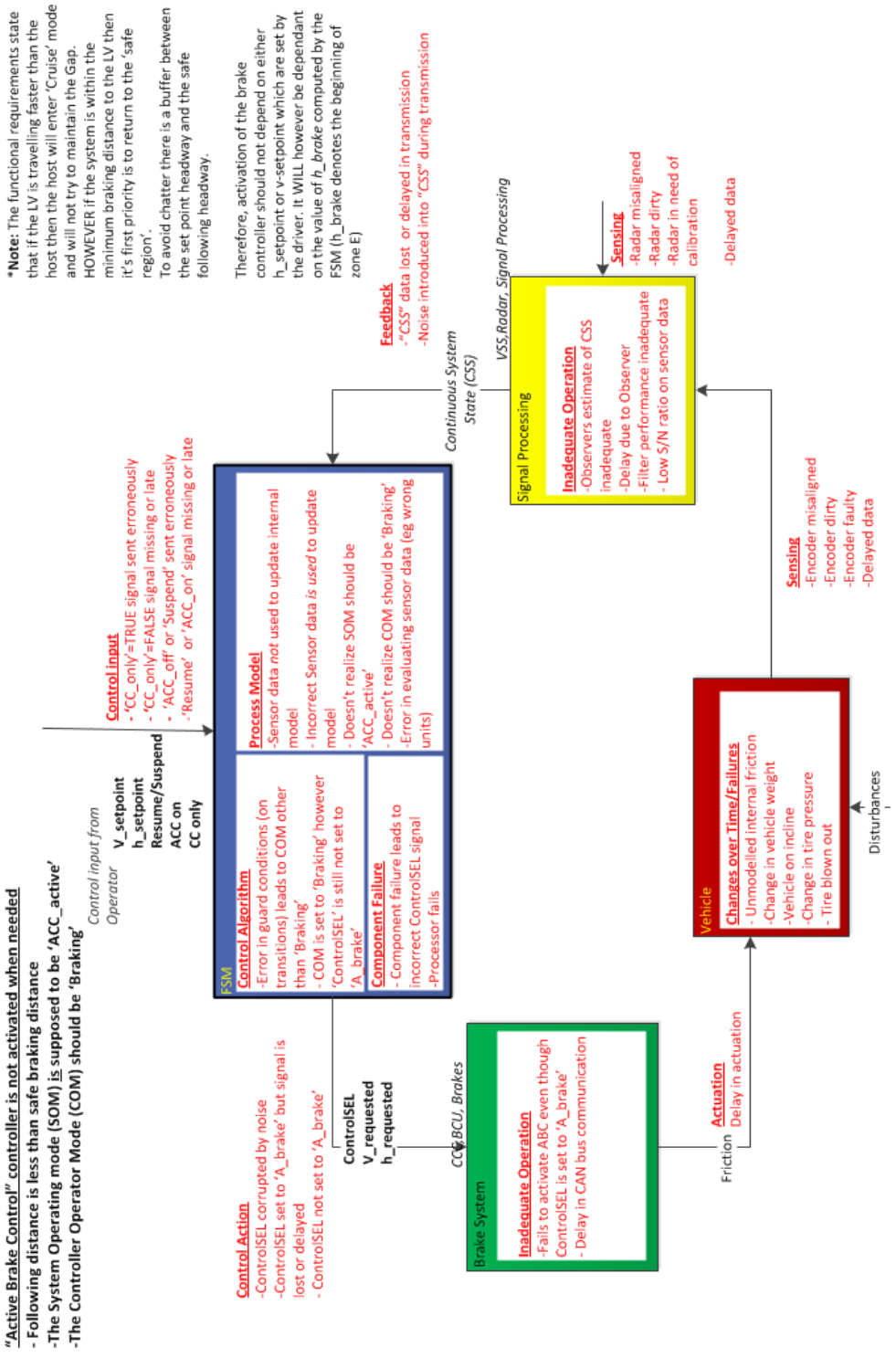


Figure A.6: Results of STPA Step 2 - Method 2

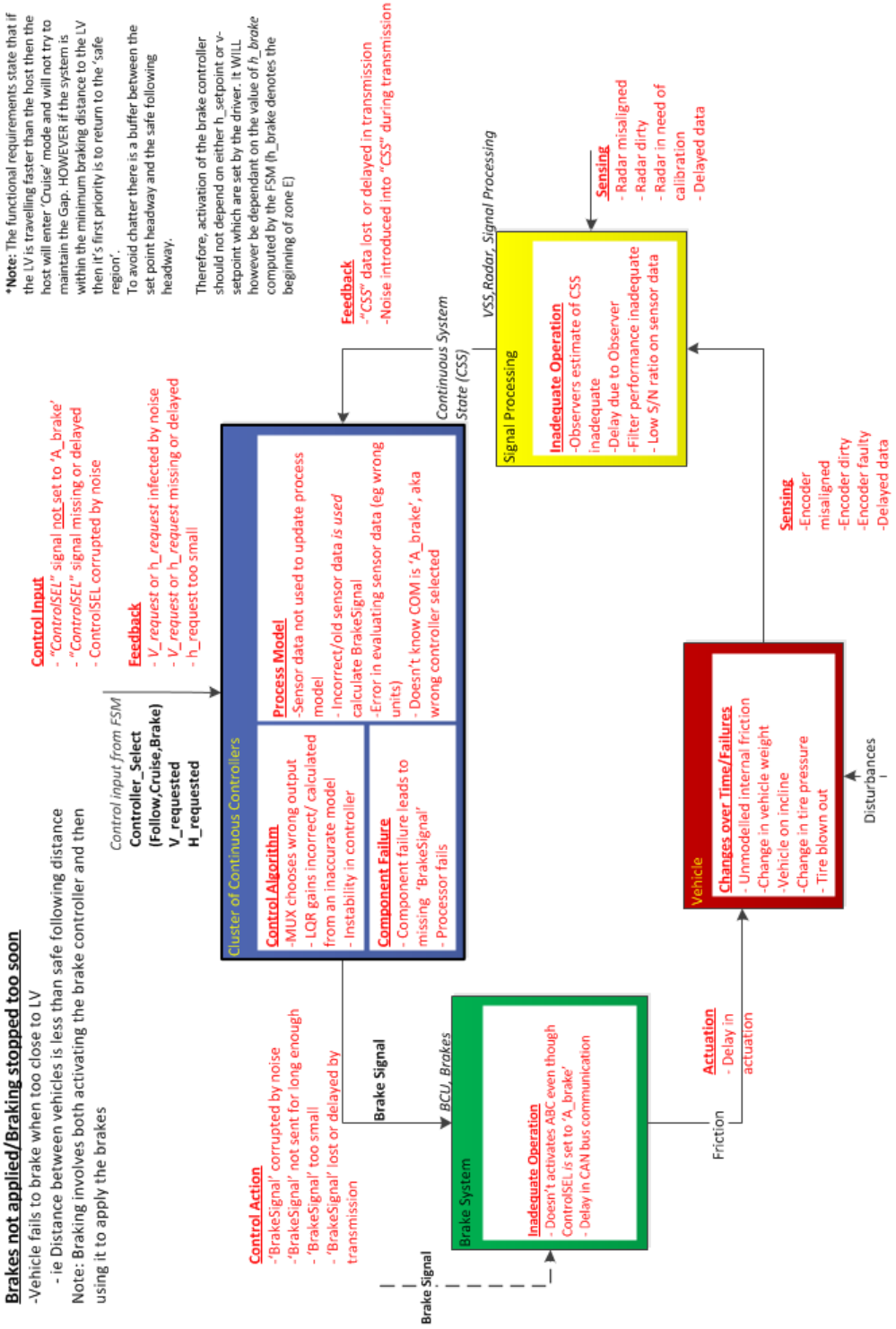


Figure A.7: Results of STPA Step 2 - Method 2

A.4.2 Table of Results

Results of STPA Step 2		
Component	Type of failure	Specific
Radar	Feedback Inadequacies	Radar Blocked
		Radar Dirty
		Calibration Needed
		Misaligned/Mounting issues
		Unable to operate in current environment (eg weather)
	Feedback delays	Update rate too low
	Component failure	Internal fault on Radar sensor
Encoder	Feedback Inadequacies	Encoder Dirty
		Misaligned/Mounting issues
		Resolution too low
	Feedback delays	Update rate too low
	Component failure	Component failure
Signal Processing Unit	Inaccuracy	Inadequate filtering to remove noise
		Filtering corrupts data
		Observer in need of tuning
	Feedback delays	Observer's operation too slow
	Inaccurate Process Model/ Inadequate control	Incorrect scaling of sensor data for input to system

Component	Type of failure	Specific
		Observer's model of system incorrect
Brakes	Component Failure	Fault in physical braking system (eg brake pads, master cylinder etc.)
	Delay	Delay in actuation
Engine	Component Failure	Fault in physical engine
	Delay	Delay in actuation
Brake and Throttle Control Units	Component Failure	Delay in CAN bus communication
		CAN bus loses signal
		CAN bus corrupts data
		Processor Failure
		Power loss
	Inadequate operation	Error in calculating PWM signal
	Controller Interaction	Conflicting commands from system and driver
CCC	Process Model Incorrect	Sensor feedback missing or delayed
		Incorrect/Old sensor data used to calculate Brake and Throttle signals

Component	Type of failure	Specific
	Inadequate Control Algorithm	LQR gains incorrect/ calculated from an inaccurate/outdated model
		Instability in controller
		Sensor data incorrectly scaled/interfaced
	Inadequate Control Action	Outputs wrong controller's result
	Inadequate Control Input	Incorrect controller selected
		Set points ($V_{request}$ & $h_{request}$ missing, delayed or corrupt)
	Component Failure	Processor Failure
		Power loss
		MUX failure
FSM	Process Model Incorrect	State transitions incorrect
	Inadequate Control Algorithm	Guard conditions incorrect
		State contents incorrect
		Sensor Data not used to update internal model
		Incorrect Sensor data is used to update internal model

Component	Type of failure	Specific
		Error in evaluating sensor data (eg units mismatch)
	Mode Confusion	Wrong system operating mode
		Wrong Controller operating mode (state)
	Bad Control Input	Incorrect, Missing or delayed Driver Signal*
	Component Failure	Processor Failure
		Power Loss
Vehicle	Changes over time	Vehicle on incline
		Low or flat tires
		Un-modelled frictions
		Change in vehicle weight
	Component Failures	Component failure (eg axle brakes)
Inter-component communication	Feedback/Control Delay	Signal delayed
	Feedback/Control Incorrect	Signal corrupted by noise
	Feedback/Control missing	Signal lost in transmission

Table A.2: STPA Step 2 Results

Obtaining the Systems as a Differential Equation

Starting with the transfer function from Chapter 5

$$G(z) = \frac{Y(z)}{U(z)} = z^{-2}\frac{[2.2202 - 2.046z^{-1}]/30}{1 - 1.7445z^{-1} + 0.756z^{-2}} \times scale \tag{A.9}$$

Convert to s-domain using Zero Order Hold

$$G(s) = \frac{Y(s)}{U(s)} = e^{-2Ts}\frac{27.17s + 739.5}{s^2 + 93.245s + 1403} \times scale \tag{A.10}$$

To make things easier

$$G(s) = \frac{Y(s)}{U(s)} = e^{-2T_s}\frac{a_1 s + a_2}{s^2 + b_1 s + b_2} \tag{A.11a}$$

$$a_1 = 21.17 \times scale \tag{A.11b}$$

$$a_2 = 739.5 \times scale \tag{A.11c}$$

$$b_1 = 93.245 \tag{A.11d}$$

$$b_2 = 1043 \tag{A.11e}$$

Next convert to Time Domain

$$Y(s)\left(s^2 + b_1 s + b_2\right) = U(s)\left[e^{-2Ts}(a_1 s + a_2)\right] \tag{A.12}$$

$$\ddot{y}(t) + b_1\dot{y}(t) + b_2 y(t) = a_1\dot{u}(t + 2T_s) + a_2 u(t + 2T_s) \tag{A.13}$$

$$y(t) = \frac{a_1\dot{u}(t + 2T_s) + a_2 u(t + 2T_s) - \ddot{y}(t) - b_1\dot{y}(t)}{b_2} \tag{A.14}$$

According to the LQR feedback law for Output weighted LQR

$$u(t) = -K_{LQR}y(t) \tag{A.15}$$

Therefore,

$$u(t) = -\begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{bmatrix} \begin{bmatrix} V(t) & \sim(t) & d(t) & a(t) \end{bmatrix} \tag{A.16}$$

$$u(t) = -k_1 V(t) - k_2 \sim(t) - k_3 d(t) - k_4 a(t) \tag{A.17}$$

$$d(t) = \int y(t)\mathrm{d}t \tag{A.18}$$

$$\sim(t) = -0.756V(t - T_s) - 0.0559u(t) \tag{A.19}$$

$$= c_1 V(t - T_s) + c_2 u(t) \tag{A.20}$$

Since the output of the system is velocity, $y(t) = V(t)$

$$u(t) = -k_1 y(t) - k_2 c_1 y(t - T_s) - k_2 c_2 u(t) - k_3 \int y(t)\mathrm{d}t - k_4 \dot{y}(t) \tag{A.21}$$

$$u(t) = \frac{-k_1 y(t) - k_2 c_1 y(t - T_s) - k_3 \int y(t)\mathrm{d}t - k_4 \dot{y}(t)}{[1 + k_2 c_2]} \tag{A.22}$$

$$\dot{u}(t) = \frac{-k_1 \dot{y}(t) - k_2 c_1 \dot{y}(t - T_s) - k_3 \cdot y(t) - k_4 \ddot{y}(t)}{[1 + k_2 c_2]} \tag{A.23}$$

Subbing into (A.14)

$$y(t) = \frac{a_1\dot{u}(t+2T_s) + a_2u(t+2T_s) - \ddot{y}(t) - b_1\dot{y}(t)}{b_2} \tag{A.24}$$

$$y(t-2T_s) = \frac{a_1\dot{u}(t) + a_2u(t) - \ddot{y}(t-2T_s) - b_1\dot{y}(t-2T_s)}{b_2} \tag{A.25}$$

$$\begin{aligned} b_2y(t-2Ts) + \ddot{y}(t-2T_s) + b_1\dot{y}(t-2T_s) = \\ \frac{a_1}{[1+k_2c_2]}[-k_1\dot{y}(t) - k_2c_1\dot{y}(t-T_s) - k_3\cdot y(t) - k_4\ddot{y}(t)] \\ + \frac{a_2}{[1+k_2c_2]}[-k_1y(t) - k_2c_1y(t-T_s) - k_3\int y(t)\mathrm{d}t - k_4\dot{y}(t)] \end{aligned} \tag{A.26}$$

Multiply in

$$\begin{aligned} (1+k_2c_2)[b_2y(t-2T_s) + \ddot{y}(t-2T_s) + b_1\dot{y}(t-2T_s)] = \\ - a_1k_1\dot{y}(t) - a_1k_2c_1\dot{y}(t-T_s) - a_1k_2c_2\dot{u}(t) - a_1k_3\cdot y(t) - a_1k_4\ddot{y}(t) \\ - a_2k_1y(t) - a_2k_2c_1y(t-T_s) - a_2k_2c_2U(t) - a_2k_3\int y(t)\mathrm{d}t - a_2k_4\dot{y}(t) \end{aligned} \tag{A.27}$$

Organize terms by derivative

$$\begin{aligned} [b_2 + b_2k_2c_2]y(t-2T_s) + [1+k_2c_2]\ddot{y}(t-2T_s) + [b_1 + b_1k_2c_2]\dot{y}(t-2T_s) = \\ - [a_2k_1 + a_1k_3]y(t) - [a_1k_1 + a_2k_4]\dot{y}(t) - [a_1k_4]\ddot{y}(t) \\ - a_2k_3\int y(t)\mathrm{d}t - [a_2k_2c_1]y(t-T_s) - [a_1k_2c_1]\dot{y}(t-T_s) \end{aligned} \tag{A.28}$$

Isolate the $y(t)$ term

$$\begin{aligned}[a_2k_1 + a_1k_3]y(t) = &-[a_1k_1 + a_2k_4]\dot{y}(t) - [a_1k_4]\ddot{y}(t) - a_2k_3\int y(t)\mathrm{d}t \\ &- [a_2k_2c_1]y(t-T_s) - [a_1k_2c_1]\dot{y}(t-T_s) - [b_2 + b_2k_2c_2]y(t-2Ts) \\ &- [1 + k_2c_2]\ddot{y}(t-2T_s) - [b_1 + b_1k_2c_2]\dot{y}(t-2Ts) \quad \text{(A.29)}\end{aligned}$$

Finally

$$\begin{aligned}y(t) = &-\frac{a_1k_1 + a_2k_4}{a_2k_1 + a_1k_3}\dot{y}(t) - \frac{a_1k_4}{a_2k_1 + a_1k_3}\ddot{y}(t) - \frac{a_2k_3}{a_2k_1 + a_1k_3}\int y(t)\mathrm{d}t \\ &- \frac{a_2k_2c_1}{a_2k_1 + a_1k_3}y(t-T_s) - \frac{a_1k_2c_1}{a_2k_1 + a_1k_3}\dot{y}(t-T_s) - \frac{b_2 + b_2k_2c_2}{a_2k_1 + a_1k_3}y(t-2T_s) \\ &- \frac{1 + k_2c_2}{a_2k_1 + a_1k_3}\ddot{y}(t-2T_s) - \frac{b_1 + b_1k_2c_2}{a_2k_1 + a_1k_3}\dot{y}(t-2T_s) \quad \text{(A.30)}\end{aligned}$$

If we use km/h then $scaling = 1/29.92$. So the numerical values to sub back into the equations are

$$\begin{aligned}a_1 &= 21.17/29.92 = 0.708 & b_1 &= 93.25 & c_1 &= -0.756 \\ a_2 &= 739.5/29.92 = 24.72 & b_2 &= 1043 & c_2 &= -0.056\end{aligned}$$

$$\begin{aligned}y(t) = &-\tfrac{0.708k_1+24.72k_4}{24.72k_1+0.708k_3}\dot{y}(t) - \tfrac{0.708k_4}{24.72k_1+0.708k_3}\ddot{y}(t) \quad \text{(A.31)} \\ &-\tfrac{24.72k_3}{24.72k_1+0.708k_3}\textstyle\int y(t)\mathrm{d}t + \tfrac{18.69k_2}{24.72k_1+0.708k_3}y(t-T_s) \\ &+\tfrac{0.535k_2}{24.72k_1+0.708k_3}\dot{y}(t-T_s) - \tfrac{1043-85.41k_2}{24.72k_1+0.708k_3}y(t-2T_s) \\ &-\tfrac{1+k_2(-0.056)}{24.72k_1+0.708k_3}\ddot{y}(t-2T_s) - \tfrac{93.25-5.22k_2}{24.72k_1+0.708k_3}\dot{y}(t-2T_s)\end{aligned}$$

So the resulting differential equation is dependent on the values of $\begin{bmatrix} k_1 & k_2 & k_3 & k_4 \end{bmatrix}$

References

Canet, P. (1994). *Kalman filter estimation of angular velocity and acceleration on-line implementation.* Unpublished doctoral dissertation, McGill University, Montreal, Qc.

Chandhrasekaran, V., & Choi, E. (2009, Sept.). Fault tolerance for embedded control system. In *Iscit 2009. 9th international symposium on communications and information technology* (p. 1316 -1320).

Dew, M. (2002). *Coordinated adaptive cruise control: Design and simulation.* Unpublished doctoral dissertation, University of California, Berkeley.

Girard, A. R., Spry, S., & Hendrick, K. (2005, March). Intelligent cruise control applications: Real-time embedded hybrid control software. *Robotics Automation Magazine, IEEE*, *12*(1), 22 - 28. doi: 10.1109/MRA.2005.1411415

Goebel, R., Sanfelice, R., & Teel, A. (2009, April). Hybrid dynamical systems. *IEEE Control Systems*, *29*(2), 28 -93.

Habibi, S. R. (2011a). *Lecture notes on system identification - part 2.* University Lecture.

Habibi, S. R. (2011b). *System identification lecture 6.* University Lecture.

Han, D., & Yi, K. (2006, June). Evaluation of adaptive cruise control algorithms on a virtual test track. In *American control conference, 2006.*

Hedrick, J. K., & Yip, P. (2000). Multiple sliding surface control: Theory and application. *Journal of Dynamic Systems, Measurement, and Control*, *122*(4), 586-593.

Hesphanha, J. (2007). *Undergraduate lecture notes on lqg/lqr controller design.* University Lecture.

Jairam, S., Lata, K., Roy, S., & Bhat, N. (2008). Verification of a mems-based adaptive cruise control system using simulation and semi-formal approaches. In *Electronics, circuits and systems, 2008. icecs 2008. 15th ieee international conference on* (p. 910 -913).

Johansson, R. (1993). Englewood Cliff, New Jeresy, 07632: Prentice Hall Inc.

Junaid, K. M., Shuning, W., Usman, K., & Naveed, R. (2005). Lqr autonomous longitudinal cruise control with a minimum state observer. In *Proceedings of the eighth iasted international conference: Intelligent systems and control.*

Knight, J. C., & Leveson, N. G. (1986, January). An experimental evaluation of the assumption of independence in multi-version programming. *IEEE Trans. Softw. Eng.*, *12*(1), 96–109.

Kural, E., & B.A., G. (2010, Oct.). Model predictive adaptive cruise control. In *2010 ieee international conference on systems man and cybernetics (smc)* (p. 1455 -1461).

Lees, M. N., & Lee, J. D. (n.d.). Driver distraction and reliance adaptive cruise control in the context of sensor reliability and algorithm limits. In *Proceedings of the third international driving symposium on human factors in driver assessment training and vehicle design.*

Leveson, N. (2009). *Engineering a safer world: Draft.*

Li, S., Li, K., Rajamani, R., & Wang, J. (2011, May). Model predictive multi-objective vehicular adaptive cruise control. *IEEE Transactions on Control Systems Technology*, *19*(3), 556 -566.

Liang, C.-Y., & Peng, H. (1999). Optimal adaptive cruise control with guaranteed string stability. In *Proceedings of the 1998 avec conference* (p. 717-722).

Loos, S., Platzer, A., & Nistor, L. (2011). Adaptive cruise control: Hybrid, distributed, and now formally verified. In M. Butler & W. Schulte (Eds.), *Fm 2011: Formal methods* (Vol. 6664, p. 42-56). Springer Berlin Heidelberg.

MATLAB. (2012). *Matlab documentation.* The MathWorks Inc.

McCullough, K. (2011). *Design and characterization of a dual electro-hydrostatic actuator.* Unpublished doctoral dissertation, McMaster University, Hamilton, ON.

Mohammadi, R. (2009). *Fault diagnosis of hybrid systems with applications to gas turbine engines.* Unpublished doctoral dissertation, Concordia University, Montreal, Qc.

Ogata, K. (1987). *Discrete-time control systems* (M. Rizzi, Ed.). Prentice Hall Inc.

Sha, L. (2001, Jul/Aug). Using simplicity to control complexity. *Software, IEEE*, *18*(4), 20 -28. doi: 10.1109/MS.2001.936213

Shakouri, P., & Ordys, A. (2011, Oct.). Application of the state-dependent nonlinear model predictive control in adaptive cruise control system. In *14th international ieee conference on intelligent transportation systems (itsc)* (p. 686 -691).

Song, Y. (2012). *Applying system-theoretic accident model and processes (stamp) to hazard analysis*. Unpublished master's thesis, McMaster University, Hamilton, ON.

Sullivan, J. (2012). *A development and testing platform for automotive active safety systems using a small-scale vehicle*. Unpublished master's thesis, McMaster University, Hamilton, ON.

Wassyng, A., Lawford, M., & Maibaum, T. (2012). Separating safety and control systems to reduce complexity. In M. Hinchey & L. Coyle (Eds.), *Conquering complexity* (p. 85-102). Springer London.

Yoshinori Yamamura, M. K., Masahiko Tabe, & Murakami, T. (2001). Development of an adaptive cruise control system with stop-and-go capability. *SAE Technical Paper 2001-01-0798*.

Zad, S. H., Kwong, R., & W.M.Wonham. (2003). Fault diagnosis in discrete event systems: Framework and model reduction. *IEEE Transactions On Automatic Control*, *48*(E2), 1199-1212.