MODEL BASED CONTROL DESIGN AND INTEGRATION OF AUTOMOTIVE CYBER-PHYSICAL SYSTEMS

By

Di Shang

Thesis
Submitted to the Faculty of the Graduate School of Vanderbilt University in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE in

ELECTRICAL ENGINEERING

May, 2013

Nashville, Tennessee

Approved:
Professor Xenofon Koutsoukos
Professor Alan Peters
ACKNOWLEDGMENTS

I would like to acknowledge the support from General Motors. I would like to thank my research advisor, Dr. Xenofon Koutsoukos, for providing me a lot of constructive advice and the opportunity to do research on the vehicle platform. Also, I would like to be appreciated for the significantly help from two senior PhD work fellow, Emeka Eyisi and Zhenkai Zhang. Finally, I feel very grateful to Dr. Peters for his advice on the thesis writing.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vi</td>
</tr>
<tr>
<td>I INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>I.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>I.2 Challenge</td>
<td>1</td>
</tr>
<tr>
<td>I.3 Problem Statement</td>
<td>2</td>
</tr>
<tr>
<td>I.4 Contribution</td>
<td>3</td>
</tr>
<tr>
<td>I.5 Organization of the thesis</td>
<td>3</td>
</tr>
<tr>
<td>II BACKGROUND AND RELATED WORK</td>
<td>5</td>
</tr>
<tr>
<td>II.1 Lane Keeping Control</td>
<td>5</td>
</tr>
<tr>
<td>II.1.1 Parameter Identification Self Turning Controller</td>
<td>5</td>
</tr>
<tr>
<td>II.1.1.1 Self Tuning Controller Tuned by Dead Beat</td>
<td>5</td>
</tr>
<tr>
<td>II.1.1.2 Self Tuning Controller Tuned by Pole Placement</td>
<td>8</td>
</tr>
<tr>
<td>II.1.2 Self Turning Regulator</td>
<td>9</td>
</tr>
<tr>
<td>II.2 Embedded System Modeling Language</td>
<td>11</td>
</tr>
<tr>
<td>III CONTROL DESIGN</td>
<td>14</td>
</tr>
<tr>
<td>III.1 Vehicle Dynamic Linearized Model</td>
<td>14</td>
</tr>
<tr>
<td>III.2 CCD Camera Model</td>
<td>15</td>
</tr>
<tr>
<td>III.3 Lane Keeping Control Design</td>
<td>16</td>
</tr>
<tr>
<td>III.4 Lane Keeping Control Simulation</td>
<td>17</td>
</tr>
<tr>
<td>III.4.1 Lane Keeping Control Curvy Road Simulation</td>
<td>18</td>
</tr>
<tr>
<td>III.4.2 Lane Keeping Control Circle Road Simulation</td>
<td>18</td>
</tr>
<tr>
<td>IV LANE KEEPING CONTROL SOFTWARE IMPLEMENTATION AND SIMULATION</td>
<td>21</td>
</tr>
<tr>
<td>IV.1 Lane Keeping Control Software Implementation</td>
<td>21</td>
</tr>
<tr>
<td>IV.2 Hardware In the loop Simulation</td>
<td>23</td>
</tr>
<tr>
<td>IV.3 Hardware in the Loop Simulation Platform</td>
<td>24</td>
</tr>
<tr>
<td>IV.4 Lane Keeping Control HIL Simulation</td>
<td>25</td>
</tr>
<tr>
<td>V CPS CONTROL MODEL INTEGRATION</td>
<td>27</td>
</tr>
<tr>
<td>V.1 Adaptive Cruise Controller</td>
<td>27</td>
</tr>
<tr>
<td>V.2 Lane Keeping and Cruise Control Integration</td>
<td>28</td>
</tr>
<tr>
<td>V.2.1 Simulink Simulation</td>
<td>30</td>
</tr>
<tr>
<td>V.2.1.1 Straight-Curve Track Without Supervisor Controller</td>
<td>30</td>
</tr>
<tr>
<td>V.2.1.2 Straight-Curve Track With Supervisor Controller</td>
<td>31</td>
</tr>
<tr>
<td>V.2.1.3 Circle Track Without Supervisor Controller</td>
<td>32</td>
</tr>
<tr>
<td>V.2.1.4 Circle Track With Supervisor Controller</td>
<td>33</td>
</tr>
</tbody>
</table>
VI CONCLUSIONS AND FUTURE WORK ........................................ 41
VI.1 Conclusion ................................................................. 41
VI.2 Future Work .............................................................. 41

BIBLIOGRAPHY ................................................................. 42
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>III.1 Vehicle Dynamic Parameters</td>
<td>14</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.1</td>
<td>CPS Design Layer [1]</td>
<td>2</td>
</tr>
<tr>
<td>II.1</td>
<td>Lane Keeping Control Sketch</td>
<td>5</td>
</tr>
<tr>
<td>II.2</td>
<td>ARMAX Model [2]</td>
<td>6</td>
</tr>
<tr>
<td>III.1</td>
<td>CCD model Sketch</td>
<td>15</td>
</tr>
<tr>
<td>III.2</td>
<td>Lane Keeping Control</td>
<td>16</td>
</tr>
<tr>
<td>III.3</td>
<td>Simulink Model Block</td>
<td>18</td>
</tr>
<tr>
<td>III.4</td>
<td>LKC Curve-Road Simulink Simulation Result</td>
<td>19</td>
</tr>
<tr>
<td>III.5</td>
<td>Lane Keeping Stable Offset</td>
<td>20</td>
</tr>
<tr>
<td>III.6</td>
<td>LKC Circle-Road Simulink Simulation Result</td>
<td>20</td>
</tr>
<tr>
<td>IV.1</td>
<td>Logical Software Architecture of LKC Controller</td>
<td>21</td>
</tr>
<tr>
<td>IV.2</td>
<td>LKC Controller Network Structure</td>
<td>22</td>
</tr>
<tr>
<td>IV.3</td>
<td>LKC Controller Platform Deployment</td>
<td>22</td>
</tr>
<tr>
<td>IV.4</td>
<td>Timing/Execution Model of Lane Keeping Controller</td>
<td>22</td>
</tr>
<tr>
<td>IV.5</td>
<td>HIL General Diagram</td>
<td>24</td>
</tr>
<tr>
<td>IV.6</td>
<td>LKC HIL Simulation Platform</td>
<td>25</td>
</tr>
<tr>
<td>IV.7</td>
<td>Lane Keeping Control HIL Simulation Result</td>
<td>26</td>
</tr>
<tr>
<td>V.1</td>
<td>Adaptive Cruise Controller Diagram</td>
<td>27</td>
</tr>
<tr>
<td>V.2</td>
<td>Adaptive Cruise Control Simulation Result</td>
<td>28</td>
</tr>
<tr>
<td>V.3</td>
<td>Supervisor Controller</td>
<td>29</td>
</tr>
<tr>
<td>V.4</td>
<td>Integrated Control Simulink Model</td>
<td>30</td>
</tr>
<tr>
<td>V.5</td>
<td>Straight-Curve Combined Track</td>
<td>31</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>V.6</td>
<td>Straight-Curve Road Simulation Without Supervisor Control</td>
<td>32</td>
</tr>
<tr>
<td>V.7</td>
<td>Straight-Curve Road Simulation With Supervisor Control</td>
<td>33</td>
</tr>
<tr>
<td>V.8</td>
<td>Circle Road Simulation Without Supervisor Control</td>
<td>34</td>
</tr>
<tr>
<td>V.9</td>
<td>Circle Road Simulation With Supervisor Control</td>
<td>35</td>
</tr>
<tr>
<td>V.10</td>
<td>Integrated Control Platform Structure</td>
<td>36</td>
</tr>
<tr>
<td>V.11</td>
<td>Logical Software Architecture of ACC And LKC Controller</td>
<td>36</td>
</tr>
<tr>
<td>V.12</td>
<td>Network/Platform Representation</td>
<td>37</td>
</tr>
<tr>
<td>V.13</td>
<td>Platform Deployment Aspect of Control Software</td>
<td>38</td>
</tr>
<tr>
<td>V.14</td>
<td>Timing/Execution Model of Integrated Control System</td>
<td>39</td>
</tr>
<tr>
<td>V.15</td>
<td>Integrated Control HIL Simulation Result</td>
<td>40</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

1.1 Motivation

Cyber-Physical System (CPS) are complex systems because of the tight coupling and interactions between the physical dynamic, computational dynamic and communication network. With the interaction between the embedded computers, networks and physical processes, the Cyber and Physical levels of whole system affect each other [4]. As an example of CPS, the recently automotive systems employ nearly 100 electronic control units (ECU) for the computation of physical dynamic analysis and control. And most of these ECUs control the safety-critical subsystem of vehicle, such as steering and breaking system, so a failure of the ECU might lead serious safety problem [5][6]. The complexity of control system in ECUs is constrained by the conflict between the functionality and ECU computation ability, the persisted effort for low production costs, and tight time-to-market [7]. Due to the requirement and constrain of control software system, a reliable and efficient approach for control software design become a dire need.

On the other hand, to achieve a global system objective, such as vehicle self-driving, individual control systems are designed isolate then integrated with each other. During the control system integration, interactions from both the Cyber and physical domains, which may not be accounted for during design can manifest as a result of the composition of these components. Additionally, the independently design control application might have objective with result in conflicts during operation. Due to the tight coupling in CPS, the complex interaction within and between the Cyber and physical domains of CPS affects the overall behavior of the integrated system and can result in unintended behaviors if not properly handled. Most integration control system problems are usually found in the final phase of the development cycle. And system correcting costs effort highly because it involves the modification of specifications, requirements and design. Hence, a systematic design, analysis and realistic testing of such system.

1.2 Challenge

The widely-known technique, model-based design approach (MBD)[8], is advantageous to solve the control system integration problem. But a reasonable approach used to integrate control system components which are generated from software and deployed on platform is still not clear. That makes the model integrating by MBD challenging for the model interaction during the design often manifest when integration. Currently, to make the system work, ad-hoc methods are usually used but this is not realistic especially when the system complexity increase.
I.3 Problem Statement

In [1], it is presented the Cyber-Physical System is composed by three fundamental layers (Fig I.1). The three layers are physical layer, the platform layer and the computation/communication layer. The physical layer represents system components related to the physical dynamics which are usually described in continues time by physical equations. Then the platform layer represents CPS components for communication network and computation which interact with the physical layer through sensors and actuators. This thesis is based on a case of three CPS design layers. An automotive Cyber-Physical system is explored. Specifically, the physical layer represents the vehicle physical components such as engine, steering, break or tires. The physical objects are interconnected by physical components (e.g steering wheel) or Cyber-Physical objects (e.g. steer by wire). The platform layer is composed by electrical control units as well as the network components on which ECUs communicate with each other. The computation/communication layer includes software codes for control application for vehicle, for example lane keeping control (LKC) and adaptive cruise control (ACC).

On the other hand, considering the model integration on CPS, the behavior emerging of vehicle control include the coupling and interactions within and across the components in all three design layers. These interactions and the effects of overall system behaviors are typically divided into two main categories:

1. **Physical Interactions** represent interactions that manifest as a result of composition of physical objects as well as changes in their dynamics and environment. Examples of such interactions are effects of changes in physical structure such as mass, suspension type, engine type etc. These interactions also include changes in the environment such as changes in road geometry, curves, road grade, banking, frictional surfaces etc.

![Figure I.1: CPS Design Layer [1]](image-url)
2. **Cyber Interactions** represent interactions led by composition of Cyber components and variation in both of network/platform and computation/communication layers. The changes might include variation in network and computation component capacities and speed, deployment model, shared resources, task allocation as well as timing specifications and scheduling etc.

In this thesis, it is assumed the physical layer components are defined by the CarSim vehicle dynamic model. It is also assumed the network/platform is specified base on a defined set of computational nodes and communication network. The research problem address in this thesis is the design and integration of components in the computation/communication layer of the vehicle, an automotive Cyber-Physical System. In detail, a lane keeping controller for vehicle lateral dynamic is designed then integrated with the adaptive cruise controller which is for vehicle longitudinal dynamic. Also, the integrated system behavior which is emerged from the Cyber and Physical interactions is discussed.

**I.4 Contribution**

In this thesis, a model-based design and integration of control system is performed on an automotive Cyber-Physical System. In detail, following the top-down model based design and integration method, a lane keeping control (LKC) application which is used for vehicle lateral dynamic stability control is designed. Then the LKC is integrated with an existing adaptive cruise controller (ACC) to form a vehicle integration control system for the overall objective towards to autonomous driving. The approach, using well-defined models, aims to evaluate and address the interaction from the Cyber and Physical domains that manifest as a result of the integration. The Embedded System Modeling Language (ESMoL)[3] is chosen as the model based design method. It streamlines control design with software modeling, code generation and deployment on platform/network, providing detailed models for the various design layers in order to constrain the resulting interactions. CarSim, a commercial vehicle dynamic simulation environment, is included for both for system design and simulation which increases the efficient of system development. The evaluating results of control design and integration are provided. For both the LKC system and integrated control system, Simulink simulation and hardware in the loop simulation on a time-triggered experiment platform are performed. From the evaluation result, it is shown the designed lane keeping controller can perform the desired control which is preventing the vehicle drive off the road. Moreover, integrated control system coordinates the LKC and ACC well in the situation when there is conflict between them.

**I.5 Organization of the thesis**

The chapters of thesis are organized as following. In Chapter 2, the background and related work about the lane keeping control is discussed. And there is also a brief introduction for the ESMoL, the design flow of
model based control design. In Chapter 3, the theoretical design of the lane keeping controller is described and the Simulink simulation result is shown and discussed. In Chapter 4, the software implementation of lane keeping controller is presented as well as the LKC hardware in the loop simulation result. In Chapter 5, the integration research of LKC and adaptive cruise controller (ACC) is discussed. The ACC is briefly reviewed then a supervisor controller is designed. Also, simulation result of integrated control system is included. At last, Chapter 6 summarizes the total result and provides further research direction.
CHAPTER II

BACKGROUND AND RELATED WORK

II.1 Lane Keeping Control
Motivated by the need to overcome dynamic traffic congestion problems and driving safety issues, the lane keeping control of vehicles has become a very active research area. The lane keeping control is a driver-assistance vehicle feature that automatically controls a vehicle’s lateral distance at look ahead distance (Fig II.1) in order to keep the vehicle between lane markings while keeping other parameters such as lateral acceleration within a comfortable driving range. The lane keeping controller (LKC) executes this objective with the aid of magnetic trackers for detecting magnetic markers on the road or through an integrated vision system. Over the years, there have been quite a few developed techniques for lane keeping control. The authors in [9] presented a performance evaluation of several lane keeping control techniques such as H-infinity control, fuzzy control and self-tuning regulator and discussed the relative trade-off with each approach. The nested PID structure introduced in [10] is deployed as lane keeping controller. The control strategy of the nest-PID control is to force the lateral displacement of the vehicle at a look ahead distance to zero. Two other control methods are introduced as following and the nest-PID structure is illustrated in Chapter 3.

II.1.1 Parameter Identification Self Turning Controller
In [11], self-tuning controllers based on system parameter identification and tuned by dead beat or pole placement control methods are introduced.

II.1.1.1 Self Tuning Controller Tuned by Dead Beat
The dead beat control method is an algebraic method. The whole system which includes the controller and plant is analyzed then the controller’s parameters are chosen according to quality criteria. As the FigureII.2,
the whole system is represented with several discrete transfer function blocks. The right part is the transfer function for plant and error signal. And the left part is the function for controller which is composed by functions for feedback signal and reference input signal. Several assumptions are made before the controller design, they are: the plant can be represent as an ARMAX form and the plant has zero initial conditions. From the ARMAX plant model, we have the equation:

\[ A(z^{-1})y(k) = z^{-d}B(z^{-1})u(k) + C(z^{-1})e_s(k) \]  \hspace{1cm} (II.1)

When \( e_s = v(k) = 0 \), and \( d = 0 \) from the controller a equation can be derived from the transfer function:

\[ P(z^{-1})K(z^{-1})u(k) = R(z^{-1})w(k) - Q(z^{-1})y(k) \]  \hspace{1cm} (II.2)

Combine equation II.1 and II.2, the whole transfer function between input reference \( w(k) \) and plant output \( y(k) \) is:

\[ G_w(z) = \frac{Y(z)}{W(z)} = \frac{B(z^{-1})R(z^{-1})}{A(z^{-1})K(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1})} \]  \hspace{1cm} (II.3)

From the transfer function, the tracking error can be derived as:

\[ E(z^{-1}) = W(z) - Y(z) = \left[ 1 - \frac{B(z^{-1})R(z^{-1})}{A(z^{-1})K(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1})} \right]W(z^{-1}) \]  \hspace{1cm} (II.4)

In order to make sure small and zero order level tracking error in a finite number of control steps, the polynomial of \( E(z^{-1}) \) should be as simple as possible, such as not in the form of a fraction. If assume \( K(z^{-1}) = 1 \), then we have:

\[ A(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1}) = 1 \]

\[ E(z^{-1}) = [1 - B(z^{-1})R(z^{-1})]W(z^{-1}) \]  \hspace{1cm} (II.5)

Figure II.2: ARMAX Model [2]
And $W(z^{-1})$ can be represented as a polynomial ratio form as $\frac{N_w(z^{-1})}{D_w(z^{-1})}$, $E(z^{-1})$ can be simplified further if $D_w(z^{-1})$ decides $1 - B(z^{-1})R(z^{-1})$ and the ratio between them is defined as

$$S(z^{-1}) = \frac{1 - B(z^{-1})R(z^{-1})}{D_w(z^{-1})}$$  \hspace{1cm} (II.6)

So at last the expression of $E(z^{-1})$ can be simplified as $S(z^{-1})N_w(z^{-1})$ if the following equations hold:

$$\left\{
\begin{array}{l}
A(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1}) = 1 \\
S(z^{-1})D_w(z^{-1}) + B(z^{-1})R(z^{-1}) = 1
\end{array}\right.$$  \hspace{1cm} (II.7)

In the situation of vehicle lateral control system, the plat polynomial $A(z^{-1})$ and $B(z^{-1})$ are usually fourth order polynomial:

$$A(z^{-1}) = 1 + a_1z^{-1} + a_2z^{-2} + a_3z^{-3} + a_4z^{-4}$$

$$B(z^{-1}) = b_1z^{-1} + b_2z^{-2} + b_3z^{-3} + b_4z^{-4}$$  \hspace{1cm} (II.7)

where the coefficients $a_n$ and $b_n$ are gotten from system parameter identification. Reference signal is usually a constant or step function which can be represented as $\frac{W}{1-z^{-1}}$. If choose a third order controller to control this system, the representation of $P(z^{-1})$ and $Q(z^{-1})$ are

$$P(z^{-1}) = p_1z^{-1} + p_2z^{-2} + p_3z^{-3}$$

$$Q(z^{-1}) = q_0 + q_1z^{-1} + q_2z^{-2} + q_3z^{-3}$$  \hspace{1cm} (II.8)

To calculate the controller parameters $p_n$ and $q_n$, the representation of $P(z^{-1})$, $Q(z^{-1})$ and $W(z^{-1})$ need to be substituted into equation II.7 and finally calculated by:

$$R(z^{-1}) = \frac{1}{b_1+b_2+b_3+b_4}$$  \hspace{1cm} (II.10)
II.1.1.2 Self Tuning Controller Tuned by Pole Placement

The pole placement control tuning is another algebraic method. By set up suitable poles, the performance of controller can be tuned for different plant model. The basic idea of pole placement tuning is introduced as following.

A system with form shown by Figure II.2 is considered for the control design. Recall the equation II.4, instead of setting the denominator as 1, it is set as a polynomial \( D(z^{-1}) \) which is related to the desired poles:

\[
A(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1}) = D(z^{-1}) \tag{II.11}
\]

Substitute \( A(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1}) \) by \( D(z^{-1}) \) in equation II.4 and still modify the reference signal \( W(z) \) to the ratio form, the equation is changed to:

\[
E(z^{-1}) = \frac{D(z^{-1}) - B(z^{-1})R(z^{-1})}{(A(z^{-1})K(z^{-1})P(z^{-1}) + B(z^{-1})Q(z^{-1}))} \frac{N_w(z^{-1})}{D_w(z^{-1})} \tag{II.12}
\]

In order to make the expression of \( E(z^{-1}) \) simple, same as the dead-beat situation, it is still assumed \( D_w(z^{-1}) \) divides \( D(z^{-1}) - B(z^{-1})R(z^{-1}) \) and the ratio is \( S(z^{-1}) \).

\[
B(z^{-1})R(z^{-1}) + D_w(z^{-1})S(z^{-1}) = D(z^{-1}) \tag{II.13}
\]

For a constant input, \( D_w(z^{-1}) = 1 - z^{-1} \). So to calculate the controller parameters, equation II.11 and II.13 show be solved together. For a four order vehicle plant, the finial equation for the controller parameter is:

\[
\begin{bmatrix}
q_0 \\
q_1 \\
q_2 \\
q_3 \\
q_4 \\
p_1 \\
p_2 \\
p_3
\end{bmatrix} =
\begin{bmatrix}
b_1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
b_2 & b_1 & 0 & 0 & 0 & a_1 - 1 & 1 & 0 \\
b_3 & b_2 & b_1 & 0 & 0 & a_2 - a_1 & a_1 - 1 & 1 \\
b_4 & b_3 & b_2 & b_1 & 0 & a_3 - a - 2 & a_2 - a_1 & a_1 - 1 \\
0 & b_4 & b_3 & b_2 & b_1 & a_4 - a - 3 & a_3 - a_2 & a_2 - a_1 \\
0 & 0 & b_4 & b_3 & b_2 & a_4 - a - 3 & a_3 - a_2 & a_3 - a_2 \\
0 & 0 & 0 & b_4 & b_3 & 0 & -a_4 & a_4 - a - 3 \\
0 & 0 & 0 & 0 & b_4 & 0 & 0 & -a_4 \\
\end{bmatrix}^{-1}
\begin{bmatrix}
d_1 + 1 - a_1 \\
d_2 + a_1 - a_2 \\
d_3 + a_2 - a_3 \\
d_4 + a_3 - a_4 \\
d_5 + a_4 \\
d_6 \\
d_7 \\
d_8
\end{bmatrix} \tag{II.14}
\]

Usually, \( D(z^{-1}) \) is set with the poles of form \( s^2 + 2\xi\omega_n s + \omega_n^2 \) which we can define the dynamic by the damping ratio (\( \xi \)) and the natural frequency (\( \omega_n \)). After a group of \( \xi \) and \( \omega_n \) are defined, the continues poles are calculated by \( s_{1,2} = -\xi\omega_n \pm \omega_n \sqrt{\xi^2 - 1} \). According to the relationship between poles of continuous
system and related discrete system \( z_i = e^{s_i T_0} \), the two poles of discrete system are:

\[
\begin{align*}
    z_1 &= e^{T_0(-\xi \omega_n + \omega_n \sqrt{\xi^2 - 1})}, \\
    z_2 &= e^{T_0(-\xi \omega_n - \omega_n \sqrt{\xi^2 - 1})}
\end{align*}
\] (II.15)

Solve the parameters of \( D \) by

\[
D(z^{-1}) = (z - z_1)(z - z_2) = 1 + d_1 z^{-1} + d_2 z^{-2}
\]

\[
d_1 = -2 e^{-2 \xi \omega_n T_0} \cos(\omega_n T_0 \sqrt{1 - \xi^2}) \quad \xi \leq 1
\] (II.16)

\[
d_1 = -2 e^{-2 \xi \omega_n T_0} \cosh(\omega_n T_0 \sqrt{1 + \xi^2}) \quad \xi > 1
\] (II.17)

\[
d_2 = e^{-2 \xi \omega_n T_0}
\] (II.18)

### II.1.2 Self Turning Regulator

An adaptive self-tuning regulator for lane keeping control is presented in [12]. The approach provides lane keeping with robustness to unknown system parameters but the complexity of the approach makes it quite challenging for an actual implementation on a digital platform.

In detail, before introduce the controller, several system coordinates transfer is introduced to make the system controllable. From the system equation with uncertain systems:

\[
\begin{align*}
\dot{x} &= f(x, u) + g(x, \theta)u \\
y &= h(x, \theta)
\end{align*}
\]

where \( \theta \) is the unknown parameter vector, \( u \) and \( y \) are the plant input and output respectively. It can be transferred to the minimum phase and in observer canonical form:

\[
\begin{align*}
    \dot{\xi} &= A_c \xi + b(\theta)\mu + \psi(y, \theta) \\
y &= C_c \xi
\end{align*}
\]

\[
A_c = \begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 1 \\
0 & 0 & 0 & \cdots & 0
\end{bmatrix},
\quad b = \begin{bmatrix}
b_p \\
b_n
\end{bmatrix},
\quad c = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0
\end{bmatrix}
\]

Then a filter of following form is introduced:

\[
\begin{bmatrix}
\dot{\xi}_1 \\
\dot{\xi}_2 \\
\vdots \\
x_{i \rho - 1}
\end{bmatrix} = \begin{bmatrix}
-\lambda_1 & 1 & 0 & \cdots & 0 \\
0 & -\lambda_2 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & -\lambda_{\rho - 1}
\end{bmatrix} \begin{bmatrix}
\xi_1 \\
\xi_2 \\
\vdots \\
x_{i \rho - 1}
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
\vdots \\
1
\end{bmatrix}
\] (II.19)
where \( \rho \) is the relative degree of the system. And also define a vector denoted by \( d[i] \theta \) as \( d[\rho] \theta = b(\theta), \)
\( d[i - 1] \theta = A_i d[i] \theta + i - 1 d[i] \theta, \rho \geq i \geq 2. \) Change the coordinates of system II.19 by define \( z = \xi - \sum_{i=2}^{\rho} d[i] \xi_{i-1} \). System II.19 is changed to:

\[
\left\{
\begin{array}{l}
\dot{z} = A_z z + d[1] \xi_1 + \psi(y, \theta) \\
y = C_z z
\end{array}
\right.
\]

Then define \( \eta_i = z_{i+1} - \frac{d_{i+1}[1] \xi_1}{d_{i}[1] \xi_1} z_1. \) The coordinates of system II.20 can be changed to:

\[
\left\{
\begin{array}{l}
\dot{\eta} = \Gamma(\theta) \eta + \beta(\theta) y + \tilde{\psi}(y, \theta) \\
\dot{y} = \eta_1 + \frac{d_2[1]}{d_1[1]} \xi_1 y + \psi_1(y, \theta) + d_1[1] \xi_1
\end{array}
\right.
\]

\[
\Gamma(\theta) = \begin{bmatrix}
\frac{d_2[1]}{d_1[1]} & 1 & 0 & \cdots & 0 \\
\frac{d_2[1]}{d_1[1]} & 0 & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{d_2[1]}{d_1[1]} & 0 & 0 & \cdots & 1 \\
\frac{d_2[1]}{d_1[1]} & 0 & 0 & \cdots & 0
\end{bmatrix}, \quad \beta(\theta) = \begin{bmatrix}
\frac{d_2[1]}{d_1[1]} - \frac{(d_1[1])^2}{(d_1[1])^2} \\
\frac{d_2[1]}{d_1[1]} - \frac{(d_1[1])^2}{(d_1[1])^2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{d_2[1]}{d_1[1]} - \frac{(d_1[1])^2}{(d_1[1])^2} \\
\frac{d_2[1]}{d_1[1]} - \frac{(d_1[1])^2}{(d_1[1])^2}
\end{bmatrix}
\]

\[
\tilde{\psi}(y, \theta) = \begin{bmatrix}
\psi_2 - \frac{d_2[1]}{d_1[1]} \psi_1 \\
\psi_3 - \frac{d_2[1]}{d_1[1]} \psi_1 \\
\vdots \\
\psi_{n-1} - \frac{d_{n-1}[1]}{d_{n-1}[1]} \psi_1 \\
\psi_n - \frac{d_n[1]}{d_n[1]} \psi_1
\end{bmatrix}
\]

Because the desired lateral displacement is zero, \( e = y - y_d = y \). So just change \( y \) in equation II.20 to \( e \):

\[
\left\{
\begin{array}{l}
\dot{\eta} = \Gamma(\theta) \eta + e \beta(\theta) + e \psi(e, y, \theta) \\
\dot{e} = \eta_1 + \frac{d_2[1]}{d_1[1]} e + e \psi_1(e, y, \theta) + d_1[1] \xi_1 (\xi_1 + p(\theta))
\end{array}
\right.
\]

Where

\[
e \phi(e, y, \theta) = \psi(e, y, \theta), \quad \gamma(r, y, \theta) = \begin{bmatrix}
\phi_2(e, y, \theta) - \frac{d_2[1]}{d_1[1]} \psi_1(e, y, \theta) \\
\phi_3(e, y, \theta) - \frac{d_2[1]}{d_1[1]} \psi_1(e, y, \theta) \\
\vdots \\
\phi_{n-1}(e, y, \theta) - \frac{d_{n-1}[1]}{d_{n-1}[1]} \psi_1(e, y, \theta) \\
\phi_n(e, y, \theta) - \frac{d_n[1]}{d_n[1]} \psi_1(e, y, \theta)
\end{bmatrix}
\]

Through the Lyapunov function [2] to design the control low, the controller equation include the control
output $\xi_2^*$ is:

$$
\xi_2^* = \lambda_1 \xi_1^* + \frac{\partial \xi_1^*}{\partial t} - \frac{1}{2} \left( \frac{\partial \xi_1^*}{\partial e} \right)^2 \xi_1(\hat{k}_2[1] + \hat{k}_3[1] + \phi_1^2) - \hat{k}_1[1] \xi_1
$$

$$
\xi_1^* = -e \hat{k}_1 + \hat{k}_2 \alpha_1(e,y_d) + \hat{k}_2 \alpha_2(e,y_d) - \dot{\hat{\rho}}
$$

$$
\dot{\hat{k}}_1[1] = \xi_2^2, \hat{k}_2[1] = \xi_1^2, \hat{k}_3[1] = \xi_1^2, \hat{\xi}_1[1] = \xi_1^2, \dot{\hat{k}}_4[1] = \xi_1^2, \phi_2^2
$$

$$
\rho_1[1] = -2 \xi_1 \frac{\partial \xi_1^*}{\partial e}, \rho_2[1] = -2 \xi_1 \frac{\partial \xi_1^*}{\partial e} \dot{\hat{\rho}}
$$

$$
\dot{\hat{k}}_1 = e^2, \dot{\hat{k}}_2 = e^2 \alpha_1(e,y_d), \dot{\hat{k}}_3 = e^2 \alpha_2(e,y_d)
$$

$$
\dot{\hat{\rho}} = e
$$

As [13], the system will be stable when $\alpha_1(e,y_d) \geq |\phi_1(e,y,\theta)|, \alpha_2(e,y_d) \geq ||\gamma(e,y,\theta)||$. When implemented, $\hat{k}$ and $\hat{k}_1$ are used to estimate $k$ (max of $k_i|i = 1,2,3$) and $k_1$ (max of $k_j[j = 1,2,3,4,$ $\xi_2^*$ is the controller output. Then the equations were as following:

$$
\xi_2^* = \lambda_1 \xi_1^* + \frac{\partial \xi_1^*}{\partial t} - \hat{k}_1 \left( \frac{\xi_1^2}{2} \frac{\partial \xi_1^*}{\partial e} \right)^2 (1 + \alpha_1^2 + \phi_2^2) + \frac{\partial \xi_1^*}{\partial e} (\dot{\rho}_1 - \dot{\rho}_2) - \dot{\hat{\rho}}
$$

$$
\xi_1^* = -\hat{k} e (1 + \alpha_1 + \alpha_2^2) - \dot{\hat{\rho}}
$$

$$
\phi_1^2 = -\hat{k} (1 + \alpha_1 + \alpha_2^2)
$$

$$
\dot{\hat{k}}_1 = e^2 (1 + \alpha_1 + \alpha_2^2), \dot{\hat{k}}_2 = e^2 \alpha_1(e,y_d), \dot{\hat{k}}_3 = e^2 \alpha_2(e,y_d)
$$

$$
\rho_1[1] = -2 \xi_1 \frac{\partial \xi_1^*}{\partial e}, \rho_2[1] = -2 \xi_1 \frac{\partial \xi_1^*}{\partial e} \dot{\hat{\rho}}, \dot{\hat{\rho}} = e
$$

### II.2 Embedded System Modeling Language

Embedded System Modeling Language (ESMoL) is a suite of domain-specific modeling language to integrate the disparate aspects of a safety-critical embedded systems design and maintain proper separation of concerns between control engineering, hardware specification, and software development teams [3]. The ESMoL sets up the relationship between controller model defined in Simulink, software implementation of the controller model and hardware on which control software will be deployed.

Structurally, ESMoL is a two-stages interpret architecture which integrates analysis tools and code generators. The first stage explicitly represents any inferred model relationships to the ESMoL Abstract model then the second stage analyses the abstract model and generate the software code. The ESMoL provides a single multi-aspect embedded software design environment so all development processing are related to a single model. By the language-specified relations, ESMoL links Simulink control model with software and
hardware design concepts. Then in every ESMoL model, objects and parameters are included to describe
deployment of software components to hardware platforms. Moreover, ESMoL can shorten design cycles
because the analysis, simulation and deployment capabilities are integrated.

There are other tools have the similar partial functionality with ESMoL. For the code generation, Real-
Time Workshop in MATLAB Simulink, for example, is another automatic code generation tool. But it can
only transfer Simulink block to C function code and further development is desired to implement code on real
time platform. The TTPPlan and similar toolbox developed by group from University of California, Berkeley
are known because of the time triggered scheduling functionality. For the platform specific simulation, there
are simulation tools such as Truetime, for network environment simulation, and System C. Compared with
the tools mentioned above, ESMoL is the first tool integrated all necessary capacities from model design to
real time software implementation.

Figure II.3 illustrates the ESMoL design flow which can be divided into 10 steps. A designed Simulink
controller model is imported automatically into the Generic Modeling Environment (GME) then changed to a
functional specification (Step 1). Software functions for controller model implementation are specified by the
imported model from Simulink (Step 2). Then, the hardware topology structure is defined as distributed ports
interconnected time triggered networks (Step 3). For Step 4, the deployment model is specified by generating
nodes for software components and communication ports for data message. By Step 5, timing model is set
by include timing parameter blocks to related software components and messages. Step 6 indicates a model
transformation from ESMoL model to the ESMoL Abstract model. All implied relationship are represented
by explicit relation objective in ESMoL. By Step 7, model interpreters transfer ESMoL Abstract model to
analysis specification then the scheduling problem is solved by the SchdeTool tool. For Step 8 and 9, another
interpreter transfers the analysis result from Step 7 back to ESMoL Abstract and ESMoL model. At last,
shown as Step 10, users can generate simulation code which can be deployed on desired platform.
Figure II.3: ESMOL Design Flow [3]
CHAPTER III

CONTROL DESIGN

In this Chapter, a lane keeping controller model is design in MATLAB/Simulink. A vehicle dynamic linear model is introduced first. Then nested PID controller structure as well as the controller parameter are illustrated. The Simulink simulation results are presented at last.

III.1 Vehicle Dynamic Linearized Model

The widely used linearized vehicle model is considered for the controller design at the first stage. As described in [14], the linearized single track vehicle model which can capture the necessary car lateral steering dynamic is:

\[
\begin{align*}
    m(\dot{v}_x - rv_y) &= f_{lf} \cos \delta_f + f_{sf} \sin \delta_f + f_{lr} \\
    m(\dot{v}_y + rv_x) &= f_{lf} \sin \delta_f - f_{sf} \cos \delta_f - f_{sr} \\
    J \dot{\gamma} &= l_f (f_{lf} \sin \delta_f - f_{sf} \cos \delta_f) + l_r f_{sr}
\end{align*}
\]

(III.1)

\[
f_{hl}(\alpha_i) = D \sin \{C \arctan [(1 - E)B \alpha_i + E \arctan (B \alpha_i)]\}
\]

(III.2)

\[
\alpha_f = \frac{v_y + l_f r}{v_x} - \delta_f, \quad \alpha_r = \frac{v_y - l_r r}{v_x}
\]

(III.3)

\[
v_y = v \sin \beta
\]

(III.4)

where \( f_{sf} \) and \( f_{sr} \) are the front and rear lateral forces, \( f_{lf} \) and \( f_{lr} \) are the front and rear longitudinal forces.

The definitions of other parameters are described by table III.1.

| \( v \) | vehicle velocity |
| \( \tau \) | vehicle yaw rate |
| \( a_l \) | lateral acceleration |
| \( m \) | vehicle mass |
| \( l_f \) | front axle-CG distance |
| \( \alpha_f \) | tire sideslip angle |
| \( \alpha_r \) | tire sideslip angle |
| \( \delta_f \) | front steering angle |
| \( \delta_r \) | front steering angle |
| \( J \) | vertical axle inertia |
| \( l_c \) | rear axle-CG distance |
| \( \beta \) | vehicle sideslip angle |
| \( C \) | Pacejka parameter |
| \( E \) | Pacejka parameter |
| \( c_{f,r} \) | cornering stiffness |
III.2 CCD Camera Model

In reality situation, the lateral displacement $y_l$ at a look head distance $l_s$ is measured by a CCD camera. During the design process, the camera is modeled by the equation:

$$\dot{y} = \beta v + l_s r + v \psi$$

(III.5)

Where $\beta$, $\rho$, $\psi$, $l_s$ are the side slip angle, road curvature, yaw angle and look ahead distance respectively.

As showed by Figure III.1, the lateral displacement at ahead is calculated base on current displacement with the effect of vehicle movement. The integration of $\beta v + v \psi$ is the current displacement while the difference between it with the displacement at head relates to the $l_s$ and the angle $\Psi_{error}$. The $\Psi_{error}$ is an angle difference between the vehicle yaw angle and the tangent direction of the curve. And it is calculated by $r - \nu \rho$ which means the difference between the yaw rate the vehicle turn and yaw rate it is required to keep on the tangent direction of the curve. To perform the lateral displacement calculation, an assumption is made that the curve is a piece of straight line between the current displacement and $y_l$ measurement point. So it is required that the radius of the curve should be much larger than the chosen look ahead distance value.

![Figure III.1: CCD model Sketch](image)

The system III.2 is linearized by decoupling the longitudinal dynamics from lateral dynamics. The reduced linear system which contains the dynamic of system III.2 and CCD camera measurements (equation III.5) is represented by the following linear vector system:

$$
\begin{bmatrix}
\dot{\beta} \\
\dot{r} \\
\dot{\psi} \\
\dot{y}_l
\end{bmatrix}
= 
\begin{bmatrix}
a_{11} & a_{12} & 0 & 0 \\
a_{21} & a_{22} & 0 & 0 \\
0 & 1 & 0 & 0 \\
v & l_s & v & 0
\end{bmatrix}
\begin{bmatrix}
\beta \\
r \\
\psi \\
y_l
\end{bmatrix}
+ 
\begin{bmatrix}
b_1 \\
b_2 \\
0 \\
0
\end{bmatrix}
\delta_f
+ 
\begin{bmatrix}
0 \\
0 \\
-\nu \\
-v l_s
\end{bmatrix}
\rho
$$
The coefficients in equations are depend on \( v \) and uncertain physical parameters:

\[
\begin{align*}
    a_{11} &= \frac{c_f + c_r}{m v}, & a_{12} &= -1 - \frac{c_f l_f - c_r l_r}{m v^2}, \\
    a_{21} &= -\frac{c_f l_f - c_r l_r}{J}, & a_{22} &= -\frac{c_f l_f^2 + c_r l_r^2}{J v}, \\
    b_1 &= \frac{c_f}{m v}, & b_2 &= \frac{c_f l_f}{J v},
\end{align*}
\]  

(III.6)

where \( c_f \) and \( c_r \) are the front and rear tire cornering stiffness after the linear approximation of equation III.2. The value of parameters are refereed from paper [10].

III.3 Lane Keeping Control Design

For lane keeping control system, a nested PID controller is deployed as shown in figure III.2.

![Diagram of Lane Keeping Control](image)

Figure III.2: Lane Keeping Control

The outer loop controller, denoted as Controller-1 in the figure, is a PID type controller with an additive integral action on the lateral offset to reject the disturbances on the curvature which is increase linearly with respect to time. The Controller-1 computes a desired reference yaw rate base on the vehicle’s lateral displacement. And the control law is illustrated by the following equation:

\[
r_d = -K_{P2} y_l - K_{I2} \int_0^t y_l dt - K_{I3} \int_0^t \int_0^t y_l dt - K_d y_{ld}
\]

(III.7)

Then the \( y_{ld} \) is given by:

\[
y_{ld} = -\frac{1}{\tau^2} \alpha + \frac{1}{\tau} y_l \quad (III.8)
\]

\[
\dot{\alpha} = -\frac{1}{\tau} \alpha + y_l \quad (III.9)
\]

Where \( \tau \) is the filter time constant and is set to a value of 0.01. \( K_{P2}, K_{I2}, K_{I3} \) and \( K_d \) are the controller gain values.
The inner loop controller, denoted as Controller-2 in figure III.2, is a PI-type controller. It calculates the steering angle value based on the error between the yaw rate and desired value output from Controller-1. The control law is described as follows:

\[
\delta_f = -K_{P1}(r - r_d) - K_{I1}\int_0^t (r - r_d) \, dt
\]

(III.10)

On the aspect of parameters’ value, the values are first tuned with the vehicle linear model by the tuner toolbox, then tuned to suit for the particular CarSim vehicle dynamic model. At last, the parameters are set as shown:

\[
\begin{align*}
K_{P1} &= 12; \\
K_{I1} &= 10; \\
K_{P2} &= 1.6; \\
K_{I2} &= 0.12; \\
K_{I3} &= 0.01; \\
K_d &= 0.0005; \\
\tau &= 0.01
\end{align*}
\]

(III.11)

### III.4 Lane Keeping Control Simulation

Simulations of the lane keeping control are performed in the MATLAB Simulink to insure the correct of control law.

The CarSim vehicle dynamic model is included as a represent of vehicle dynamic. CarSim is dynamic simulation environment used to predict vehicle behavior. It provides an accurate, extensible, fast, stable and cost effective solution for vehicle dynamic prediction [15]. With the extensibility, the CarSim model can be included in MATLAB Simulink and simulated with the Simulink controller model. And it also supports the real-time simulation with hardware in the loop using systems from most RT suppliers. A standard E-class Sedan vehicle is used as the simulation car model. As presented by Figure III.3, there are a data type conversion block and a rate transition block between the connection ports of CarSim model and controller block. The reason of including the data type conversion is because the data type of controller and CarSim model is not same. For the purpose of latter software implementation, all numeral data in the controller is set to the fixed-point type (32 bits for word length and 16 bits for fraction length) while data type CarSim model is floating number. On the other hand, the rate transfer block is include to perform up or down sample between blocks for the sample rates of CarSim model and controller are 1ms and 10ms respectively. Inside the controller block there are two subsystem for the CCD camera and the nested PID controller structure.

The simulation in two kinds of road environment is performed. The simulation results are illustrated and discussed in the following section.
III.4.1 Lane Keeping Control Curvy Road Simulation

The first simulation is on a standard curvy road provided by CarSim. The longitudinal velocity of the car is kept constantly at 108 km/h during the whole simulation processing. As shown in the figure, the initial position of vehicle is (-800, 0) which is in meter. From the lateral displacement and lateral acceleration figure, it seems, under the LKC control, the vehicle pass the curve safely. The lateral displacement is kept in the range \([-0.15, 0.05]\) which is small enough to believe the vehicle is tracking the path. Although there is slight offset, about 0.08 m, at last, the displacement is stable. By observing the lateral acceleration plotting, it is believed the driving is comfortable because the value falls into the range \([-1.5, 1.6]\) which is less than the lateral acceleration limitation \(2.4 m/s^2\) [16]. Moreover, the steering angle which is in the range \([-15, 15]\) is acceptable because it is physically implementable.

One phenomenon might need to be explained more is the offset (about 0.08 m) at last on the lateral displacement figure. The reason of that is, to make the experimental vehicle to follow a straight line stably, a slightly steering angle is required instead of a zero value. The following figure illustrates the reason graphically. To generate a small positive steering angle control signal, there should be a negative offset. As shown in FigureIII.5, the dash line in the middle is the tracking path. And the arrow indicates the direction of vehicle movement. The car is on the right of tracking path then a steering angle to the left is output by controller finally it leads the vehicle go straight.

III.4.2 Lane Keeping Control Circle Road Simulation

For the second simulation, the vehicle is simulated on a circle road with 500 feet radius. A standard Class-E Sedan is still used as the car model. The vehicle starts at an initial point of (0, 0) then accelerates to 80 km/h. From the FigureIII.6, it can be seen that the LKC performs the desired objective effectively by dynamically
set the steering angle value. From the figure for lateral displacement, it is shown the displacement value is stable and falls into the range [-0.1 0.08] so it means the vehicle tracks the circle path under the control. Then it is shown the steering angle stable at 40 degree after a peak value of 48 degree which is not exceed the physical limitation. Although the lateral acceleration value (stable at $3.5 m/s^2$ after a peak with $4 m/s^2$) is slightly high for a comfortable drive, it is caused by the high longitudinal speed and sharp curve. This issue will be solved in the chapter considers the integrated control system.
Figure III.5: Lane Keeping Stable Offset

Figure III.6: LKC Circle-Road Simulink Simulation Result
In this chapter, the software implementation of the lane keeping control system is introduced. A hardware in the loop simulation is performed then the result is discussed and compared with the Simulink result to show the control effect introduced by the real time platform.

**IV.1 Lane Keeping Control Software Implementation**

In this section, the lane keeping controller software system is built following the ESMoL design flow which is introduced in Chapter 2.

Firstly, the LKC Simulink model is imported into GME. As Figure IV.1, the software function of controller component is presented. Each GME block represents a task so there are three task in system in total. The InputHandler and OutputHandler are used to represent the function of CarSim model sensor and actuator. And the LKC task block stands for the lane keeping controller.

![Figure IV.1: Logical Software Architecture of LKC Controller](image)

Then, in Figure IV.2, the network/platform topology structure are defined in ESMoL base on the control system structure. One ECU for LKC is included and connected to the TTEthernet. The CarSim simulator runs on the RT-Target node with two virtual I/O devices which indicate the sensors and actuators of vehicle. Also, the RT-Target node connects to the TTEthernet block and the communication between RT-Target and ECU is set up.

In Figure IV.3, the deployment model of control software is presented. The dashed arrows indicate assignment of components to their respective processors, and solid lines represent assignment of message instances (component ports) to communication channels (port objects) on the processor. As a part of CarSim, tasks, InputHandler and OutputHandler, are assigned to the RT-Target processor. And the LKC is deployed on the ECU.

Figure IV.4 indicates the timing and execution model for tasks and message transfers of the control system. In detail, the clock block is used to specify the timing of related operation. For example, the I2LExec
block specifies the timing for message transfer from the InputHandler to the LKC. By setting the execution period, desired offset and the worst case duration for every communication and the scheduler with TTEthernet driver to take advantage of the synchronized time base all tasks can be executed according to the time-triggered paradigm. As other example, the LKCExec block defines the execution period of LKC which is 10ms.

For the task scheduling, a heuristic algorithm base on the bottom-level of system task graph is used. The
longest path from any task to the end in the graph is represented as the bottom-level. Follow the technology described in [17], the message order is determined while the allocation of the tasks to the processors is pre-defined. The algorithm orders the tasks as the bottom-level of its corresponding task graph vertex (in descending order). By this method, all dependencies between tasks are ensured before the task scheduling. For the LKC control system, the critical path is InputHandler $\rightarrow$ LKC $\rightarrow$ OutputHandler.

After network/task scheduling, the schedule information is updated into the ESMoL and ESMoL_Abstract models automatically. The interpreter uses the updated ESMoL_Abstract model to assemble all the codes for compilation. Using the network scheduling result, a tool named as TTEBuild from TTEtech generates the binary configuration files for TTEthernet switch and C code configuration files for ECUs. C code files, generated by Real Time Workshop, and TTEBuild with glue code files are assembled by ESMoL Stage 2 interpreter then makefiles are generated automatically. In order to take advantage of the synchronized time base of TTEthernet, all tasks are executed in RT-Linux kernel space. The compiled kernel modules are deployed on ECUs.

### IV.2 Hardware In the loop Simulation

The hardware in the loop simulation is a widely used technique during the real-time system development and verification. By substituting the actually plant by dynamic mathematical representation, the embedded software plant can be tested effectively. The electrical emulation of sensors and actuators are included in the plant mathematical representation. The embedded system reads the value from each electrically emulated sensor, then calculates the control output by the programmed control law code and implements the control through the electrically emulated actuators. So a feedback loop is formed between the embedded software and plant mathematical representation model. As a example of HIL simulation, the whole experimental platform for this experiment includes the CarSim model to represent the dynamic mathematically. In detail, there are mathematically representations for vehicle dynamics (such as steering, engine etc), sensors and actuators and road dynamic which shown as Figure IV.5.

Compared to the traditional embedded software system development process, which is developing embedded software with real plant directly, the hardware-in-the-loop simulation will be more efficient in many situations. Usually, four factors, cost, duration, safety and feasibility will be considered when judge the development and test efficiency.

The cost of embedded software developing is depend on cost of efforts and materials. For a complex plant, for example vehicle or aircraft, a mathematical representation software will be much cheaper than a real plant. And using the mathematical model for software development does not require efforts for maintaining for the actually plant. On the aspect of development duration, although developing software on the actually plant is
a more direct approach, but the problem caused by the plant might make the design lasts longer. Take the car embedded software as an example, the designer cannot absolutely insure all components on the actual vehicle perform the correctly mechanical function so a unknown component mis-function will caused problem which cost long time to locate and fix. Considering the safety factor, a mathematically model is preferred than an actually plant under some circumstances. A good example is an experiment to test the operation range of plant during which a really plant might crash, be destroyed and then lead catastrophic consequence [18]. For the feasibility, it is normal to explore the platform under situations which is not easy to achieve with real plant. For example, to explore critical timings which means user actions are given with a short time interval.

IV.3 Hardware in the Loop Simulation Platform

A system structure for the TTEthernet-based hardware in the loop simulator is described by Figure IV.6. The whole platform are divided into two layers: physical layer and platform layer.

The physical layer modeling the physical dynamic of the CPS system includes two components. The design/visualization PC is used to compute CarSim vehicle dynamic during HIL simulation and the initial control design and verification. On the other hand, the Target PC is a NI LabView Real-Time Target running NIs Real-Time Module which provides a complete solution for creating reliable, stand-alone real-time systems [20]. During experiments, the vehicle dynamics modeled by CarSim is run on the Target PC. Moreover, the Target PC is integrated with a TTEtech PCIe-XMC card which enables the seamless integration and communication with ECUs on the Time-Triggered network supported by the TTEthernet switch [19].

The platform layer is composed by the TTEthernet switch and ECU. The TTEthernet Development Switch has 8 ports with 100Mbps bandwidth. It supports 100 Base-TX Ethernet and also enables hard real-time communication based on the TTEthernet protocol. The switch set up the communication between the ECU and XMC card of Target PC and operates on the configuration defined by users.
The ECU is an IBX-530W-ATOM box with Intel Atom CPU and operated by Real-Time Linus system. To enable the communication with other components in the TTEthernet network, each ECU is equipped a TTEthernet Linux driver with related protocol. The controller software C code, generated from the design process in ESMoL, is deployed and run on the ECU. Each software system component is executed in the kernel space of running RT-Linux and follows the TTEthernet synchronized time.

**IV.4 Lane Keeping Control HIL Simulation**

A HIL simulation of the lane keeping controller is performed on the platform described before. A CarSim curvy road is used as the road environment and Class-E Sedan vehicle model is chosen as the vehicle dynamic model. The vehicle start at the position (-800, 0) then accelerate to 108 km/h and always tracks the path. The simulation result is shown by Figure IV.7. Two groups of data are illustrated on the figure. The solid black line indicates the HIL simulation result. For comparison, the Simulink simulation result on the same curvy road (Figure III.4) is referred and plotted by the dot red line.

By observing the plotting in black, it seems the embedded controller (control system on the ECU) can still perform the acceptable control to achieve the lane keeping objective. It is shown the lateral displacement value is stable and falls into the range [-0.12 0.08]. Also the steering angle value is between -15 deg and 15 deg which is physical implementable and lateral acceleration value is smaller than the limitation of comfortable driving. But by comparing the red line and black line, some conclusions can be made. Although the steering angle and lateral acceleration values are almost same during the whole simulation, there is an observable
difference between the lateral displacement plotting of two experiments. Generally, the whole displacement figure of platform (solid black line) is almost in the same shape with the Simulink figure (dot red line) expect a positive offset. But in detail, comparing two figures at the time 27s, the platform figure keep same while Simulink figure begins to decrease. Then this difference integrals and it causes the offset at last. This phenomenon reveals the fact that the platform vehicle model reacts to the control slower than the Simulink model. By reviewing the platform structure, it can be found that control delay and synchronous issues are inevitable due to the switch network connection set up between controller and car model. From this result, a conclusion can be made. It is that, addition to the traditional criteria such as stability, controllability and robust ability to noise or vary input, the time delay robust of controllers will be an important factor for CPS controller.
CHAPTER V

CPS CONTROL MODEL INTEGRATION

In this chapter, we present the integration of two control models on CPS system. The two controllers are the lane keeping controller described in prior chapters and an adaptive cruise controller illustrated in [7]. Before the introduction of controls integration issue, a briefly review of adaptive cruise controller is given. Then controller integration theory is explored beginning with the controller conflict under the particular situation and finishing at experimental result discussion and comparison. Hardware in the loop simulation is also performed to discover the control effect introduce by the experimental platform.

V.1 Adaptive Cruise Controller

The adaptive cruise control (ACC) system is an active safety and driver-assistance vehicle feature that automatically controls a vehicle longitudinal velocity in a dynamic traffic environment. It controls the ACC-equipped vehicle to maintain a certain distance, which is defined by desired time gap and current velocity, after the leading vehicle. Structurally, the ACC is a hierarchical control with two levels as shown in Figure V.1.

![Figure V.1: Adaptive Cruise Controller Diagram](image)

The upper level controller uses the driver inputs, the radar measurements and current distance and velocity of ACC-host car related to the leading vehicle, to compute the desired acceleration that is required to achieve the desired spacing or velocity. Besides, the low level controller is defined as two functions. First, to determine whether a braking or throttle control is required base on the desired acceleration form upper level controller. Second, to apply a braking or throttle control to vehicle in order to achieve the desire acceleration.

On the aspect of control function, there are two states of control which are the essential to behave like the
conventional cruise control system. The first one is when the leading car is detected by the radar, the ACC control the velocity to maintain a pre-defined time gap between two cars. The second one is when the leading car is absent, the ACC system control the velocity to maintain a driver set value. To explain the function more clearly, an experiment scenario of past work is shown by Figure V.2 to explain the speed control of ACC.

![Figure V.2: Adaptive Cruise Control Simulation Result](image)

In this experiment, two cars are set on a straight and flat road with an initial distance of 65 meters. The initial speed values of the leading and following vehicle (ACC host vehicle) are 60 km/h and 65 km/h respectively. The driver set cruise speed value is 80 km/h. At the beginning of the experiment, the following car does not detect the leading car so it accelerates to the driver set velocity value (80 km/h). The controller is in stage two mentioned above. Then, the leading car is detected and the following car breaks to reach the same speed with the leading car in order to maintain the time gap. That is the stage one as described before. After that, from about 40 second, the lead car began to accelerate so following car speeds up also until its speed reach the driver set value again. At last, the leading car begin to break and following car then keep up with the leading car.

### V.2 Lane Keeping and Cruise Control Integration

In this section, we consider the integration of the lane keeping controller design in Chapter III and previously described adaptive cruise controller. Before the integrated system design, the interaction and confliction between integrated control components should be considered first. Although the two controllers modify the behavior of two seemingly different dynamics of the vehicle, with the ACC controlling the longitudinal dynamics while the LKC controls the lateral dynamics, there exists physical interactions between lateral and longitudinal dynamics of the vehicle. Not only that, changes in the physical environment such as geometry of
vehicle path or road curvature highlights certain conflicts in the operation of the two distinctive controllers. For example, the ACC on detecting a leading vehicle dynamically adjusts the speed of the ACC-equipped vehicle to the lead vehicle’s speed. On a curved road, the ACC in an effort to track the leading vehicle, can attain a vehicle speed that might be too fast, such that it can potentially obstruct the LKC’s ability to maintain the desired lateral distance resulting in a conflict. This type of conflict can potentially result in undesired and unintended behavior of the overall system. In order to address, these types of conflicts, we integrate a supervisory controller whose main objective is to restrict the regions of operation of the integrated system in a safe desirable manner.

![Supervisor Controller Diagram](image)

Figure V.3: Supervisor Controller

As the Figure V.3, the supervisory controller operates by dynamically determining the desired longitudinal set speed of the ACC base on the perception of the current road geometry, specifically the road curvature. The idea is that by restricting the allowable speed for the ACC base on the road curvature, the LKC can equally be able to achieve its desired objective of maintaining a desired lateral distance. Hence, on a relatively straight road, the ACC operates in its normal mode with the user set-speed and radar inputs but on a curvy road, the supervisory controller modifies the user-set speed to a desired speed as the radius of curvature. The underlying relationship between desired speed and road curvature is described by equation:

\[ v_x = \sqrt{a_{yMax} \cdot \rho} \]  \hspace{1cm} (V.1)

Where \( v_x \), \( a_{yMax} \) and \( \rho \) are the driver set velocity, maximum lateral acceleration and radius of the curvature respectively.

Several groups of experiment simulation result are provided in the following part. Firstly, there are experiments in Simulink. The effect of supervisor controller is discussed by comparing the results and it is proved the including of supervisor controller is necessary to improve system performance. After that, the second group of experiment involves the model-based software development of the integrated system with the three controllers based on the controller software implementation and the deployment on the experimental...
platform for a hardware-in-the-loop simulation.

### V.2.1 Simulink Simulation

The simulation for the integrated control system in Simulink (Figure V.4) is performed first. Two CarSim models are included to represent the leading and following vehicles. For controllers, all components for one controller are included in the same Simulink block for code generation in the latter section.

![Integrated Control Simulink Model](image-url)

**Figure V.4: Integrated Control Simulink Model**

#### V.2.1.1 Straight-Curve Track Without Supervisor Controller

For this simulation, the ACC and LKC are just combined together without the coordination of supervisor controller. The selected test track is a dynamic path with a combination of straight paths and curved roads with radii of 160m, 200m and 160m for the three curves as seen in Figure V.5. The desired time gap of the ACC is set to 1.5s. The leading vehicle starts at an initial position of (0, 0) with an initial speed of 30 km/h while the host vehicle, equipped with the integrated control system, starts at an initial position of (-800, 0) with an initial speed of 80 km/h.

From Figure V.6, it can be seen that the ACC performs its desired objectives effectively by dynamically tracking the leading vehicle’s speed as shown in Figure V.6a. While at same time maintaining a safe vehicle distance as shown in Figure V.6b. On the other hand, the performance of the LKC is deteriorated due to the resulting conflicts and interactions with the ACC. The lateral displacement, as shown in Figure V.6c, deviates from the desired lane of the vehicle with a peak value of about -0.7 m. This amount of deviation can result in potentially catastrophic consequence such as a collision with vehicle in other lane. Although, the curves in the paths are very aggressive, the lateral acceleration exceeds $4 \text{m/s}^2$ for the most part in the curved roads which will lead the passenger feels uncomfortable. So it is necessary to improve the performance of lane...
keeping control when cooperates with the adaptive cruise controller.

V.2.1.2 Straight-Curve Track With Supervisor Controller

For this simulation, the ACC and LKC are integrated together with the coordination of supervisor controller. The simulation result of the second experiment is shown by Figure V.7. Moreover, the lateral displacement and lateral acceleration of first experiment are also plotted by the dot red line in order to compare. In the second experiment, a supervisor controller is included to suit the longitudinal speed through the adaptive controller base on the perceived road geometry/curvature. The specified controller, system parameters, chosen CarSim vehicle model and road model are the same as in the previous scenario.

From Figure V.7a and b, it can be observed that the longitudinal speed of the following car is modified base on the consideration on the road curve and the distance between two vehicles. For example at the time 30s, 85s and 160s, the following car breaks down because of the curve regardless that the distance it falls behind the leading car is already larger than the desire value. After finish the turn, supervisor controller accelerates the following vehicle to catch up the leading car. That is the situation occurs at 70s, 120s and 200s. From 40s to 150s, the distance figure keep constantly at 100m which means the leading car is too far way to detect by the radar. Figure V.7c and d compares the lateral performance of the system with and without a supervisory controller. It can be seen in Figure V.7c that the lateral displacement for the case with a supervisory controller is limited to a peak value about -0.34m as compared to -0.7m in the case without a supervisory controller. Likewise, the lateral acceleration is also reduced in the aggressive curves as compared to the case without the supervisory controller. As shown in the figure, the acceleration value decrease to $2\text{m/}s^2$ for most time.
when the vehicle is turning. These two scenarios highlights the importance of the supervisory controller in the integration of the two independently designed controllers specifically in handling interactions emerging from the physical layer of the CPS.

V.2.1.3 Circle Track Without Supervisor Controller

In order to prove the performance improvement introduced by the supervisor controller, an additional group of Simulink simulation is performed. For these simulations, a 500-feet-radius circle is set as the road environment. Both of the leading and following cars are chosen to the same CarSim model as the prior simulation. The initial position of the following car is (0, 0) and the leading car is start at 60m ahead along the road. The set speed of the ACC is increased to 120km/h.

Firstly, the simulation result of system without the supervisor control is shown as Figure V.8. From the vehicle path figure, it can be clearly seen that the control objective is not achieved. The vehicle slips out of the road which will lead a really dangerous accident in actually situation. The car velocity figure shows that the following vehicle (controller equipped vehicle) accelerates first to catch up with the leading car then modifies...
its speed as the leading car until slips off the road. Then, the vehicle steering angle plotting also reveals the problem. At about 90s, the steering angle value begins to significantly oscillate between the positive and negative limitation of the controller output. And similar with the steering angle figure, the acceleration value is unacceptable that the value varies rapidly and is not stable. In conclusion, the controller totally crashes at 90s and improvement must be introduced to prevent serious consequence.

V.2.1.4 Circle Track With Supervisor Controller

Secondly, a simulation result of situation when the supervisor controller is included is presented by Figure V.9. Different with the last simulation, the vehicle path figure indicates the vehicle follows the desired circle road path. The main reason of this improvement is the introduction of vehicle longitudinal speed control from supervisor controller. As shown by the car speed figure, the following car accelerates first and catches up the leading car at 20s. Then, between 30s and 90s when the leading car speed up to 110km/h, the following car does not only accelerate to follow the leading car. Because the supervisor controller dynamical sets the speed
Figure V.8: Circle Road Simulation Without Supervisor Control

base on the road curvature, the longitudinal velocity is limited at 66km/h on the circle. So instead of a same
accelerating as the leading car, the following vehicle stops to drive faster after reaches the speed limitation
calculated by the supervisor controller. The performance improvement can also observed on the steering angle
and lateral acceleration figure. From Figure V.9, the steering angle dose not significantly oscillate during the
simulation and stable at about 31deg. The peak value is 32deg so the steering angle value is acceptable for all
values are physical implementable. On the other hand, the lateral acceleration is also stable and fall into the
range \([0m/s^2, 2m/s^2]\) which means the driving can be considered as a comfortable driving by passengers. The
compare of these two simulation results also indicates the supervisor controller is necessary. The supervisor
controller can not only improve the system performance but also will prevent the whole system crushes or to
be unstable under particular circumstance.
V.2.2 HIL Simulation

V.2.2.1 System Architecture

A hardware in the loop simulation is also performed for the integrated control system. The simulation is run on the same time triggered real time platform with the one used in Chapter 2 for the lane keeping control HIL simulation. But because there are more control software application in the integrated control system, the structure of simulation platform is more complex in this experiment. As shown in figure V.10, there are three ECUs for the three controllers in total. Instead of deploying the controller software into a single ECU the controller code is deployed distributively. The fact that communications between controllers must through the TTEthernet switch network leads the control effect introduced by the network connection can be observed in this experiment.
V.2.2.2 Controller Software Implementation

Follow the ESMoL design flow, the integrated LKC and ACC controller Simulink models are imported into GME then the embedded software code is generated.

Figure V.11 indicates the software functions of the controller components in GME. In the figure, each GME block represents a task and, for the integrated controller, there are four tasks, which are Supervisor, ACC, LKC, and Collection respectively. Two additional tasks, InputHandler and OutputHandler, are used to represent the sensing and actuation of the CarSim.

Figure V.11: Logical Software Architecture of ACC And LKC Controller.
In Figure V.12, the network/platform topology structure are modeled in the ESMoL. Three ECUs for ACC, LKC and supervisor controller are specified as ECU1, ECU2 and ECU3. The RT-Target node is where the CarSim simulator runs. Then two virtual I/O devices are included to represent the sensors and actuators of vehicle. Specific parameters for TTEthernet need to be defined, such as hyperperiod, bandwidth, time slot size, clock synchronization cycle, and synchronization precision. These specified parameters can be used to generate the TTEthernet configuration script using an ESMoL interpreter.

![Network/Platform Representation](image)

In Figure V.13, the deployment model of control software is illustrated. Dashed arrows indicate assignment of components to their respective processors, and solid lines represent assignment of message instances (component ports) to communication channels (port objects) on the processor. Manually, the Supervisor and Collection are deployed on ECU1, the ACC is deployed on ECU2 and LowLevelController and LKC are set up on ECU3.

Figure V.14 illustrates the timing and execution model for tasks and message transfers of the control system. The control system runs at a period of 10ms. Since a synchronized time base for communication is provided by the TTEthernet, all the message transfers which are indicated by the timing block between tasks are attached with TTExecInfo to indicate time-triggered communication. For instance, S2LExec is used to specify the timing for the message transferring from Supervisor to LKC. The execution period, desired offset and the worst case duration for every communication are set. Execution period is the hyper-period of the TTEthernet configuration. And the desired offset is used to represent the initstart_ns field of TT message in the generated TTEthernet configuration script. Moreover, the worst case duration indicates the worst case communication time of TTEthernet. The TTEthernet driver on each ECU has a scheduler to take advantage of the synchronized time base, which can invoke the tasks according to a static schedule. Thus, all the tasks
can be executed according to the time-triggered paradigm. The \textit{TTExecInfo} is defined for each task by the timing block connect to the particular task. For example, \textit{ACCExec} block defines the execution period of \textit{ACC} is 10ms. Before scheduling, only the execution period and the task’s worst case execution time are necessary to provide. All the period time is measured empirically.

For the task scheduling, the same heuristic algorithm is used as the one in Chapter 3. After network/task scheduling, the same method is applied to generate codes then deploy on ECUs.

\textbf{V.2.2.3 Simulation Result}

The results of integrated control system hardware in the loop simulation is shown in Figure V.15. Observed from Figure V.15a and b, the vehicle speed and distance are quite same compared to the simulation result in Simulink (Figure V.7a and b). The vehicle still adjusts the speed base on the road curve and distance after the leading car. But from Figure V.15c and d, difference between Simulink and Platform result can be observed clearly. Taking a closer look at Figure V.15c, the black line enjoys the similar shape but a positive offset from the red line. The offset is caused by the not synchronizing between components and network communication delay introduced by platform. The same phenomenon is also shown in Figure IV.7b. Then, from Figure V.15d, some oscillations on the black figure is illustrated which is also leaded by the synchronization and delay issue.
Figure V.14: Timing/Execution Model of Integrated Control System.
Figure V.15: Integrated Control HIL Simulation Result

(a) Vehicle Velocity. Solid Line: Following Car, Dot Line: Leading Car
(b) Distance. Solid Line: Current Value, Dot Line: Desired Value
(c) Lateral Displacement
(d) Lateral Acceleration
CHAPTER VI

CONCLUSIONS AND FUTURE WORK

VI.1 Conclusion
This thesis presents model based design approach that facilitates the design and integration of time-triggered automotive control system. The model based design in apply on the lane keeping controller design and the integration between lane keeping control and adaptive cruise control. For the integration, the supervisor controller is introduced to handle certain physical interactions such as the impact of road curvature on the behavior of the two controller. The model design and integration of LKC and ACC is verified by the Simulink and hardware in to loop simulation. And the simulation results are discussed and compared then the necessity of supervisor controller is proved and the control effect introduced by simulation platform is illustrated.

VI.2 Future Work
In Chapter 5, the control model integration based on the model based design is illustrated. As an important step of model integration, the analysis of interaction and conflict between controller models should be performed at first but it is not included in the ESMoL. In order to achieve the automatically model based controller integration, the automatic algorithm base on model conflict analysis and resolve should be considered in the future. To get the solution, the understanding of relationship between plants dynamic for each control model is required which means the plant physical dynamic recognition technology might be worth to consider.
BIBLIOGRAPHY


[16] Ahmet Sami KILIN and Tamer BAYBURA. Determination of minimum horizontal curve radius used in the design of transportation structures, depending on the limit value of comfort criterion lateral jerk.


