THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF MECHANICAL ENGINEERING

SIMULATION STUDY OF THE ONLINE AERODYNAMIC DRAG COEFFICIENT ESTIMATION OF A HEAVY-DUTY VEHICLE

QIFENG LIU
SPRING 2019

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree in Mechanical Engineering
with honors in Mechanical Engineering

Reviewed and approved* by the following:

Hosam K. Fathy
Professor of Mechanical Engineering
Thesis Supervisor

Stephanie Stockar
Assistant Professor of Mechanical Engineering
Honors Adviser

* Signatures are on file in the Schreyer Honors College.

ABSTRACT

This thesis investigates the estimation of heavy-duty vehicles' aerodynamic drag coefficient onboard during driving. The motivation of this thesis is the importance of accurate aerodynamic drag coefficient estimates for the performance of the close-loop system, e.g., the adaptive controller in the context of heavy-duty vehicle platooning. A platoon is a train-like formation of a group of heavy-duty vehicles at close intervehicular distances. Potential benefits in overall fuel savings coming from platooning are the ultimate result of considerable air drag reduction, the decreasing value of aerodynamic drag coefficient C_D, which can vary to a large extent based on the loading situation and the relative positions with respect to other vehicles.

The thesis demonstrates an estimator that specifically identifies the aerodynamic drag coefficient of one heavy-duty vehicle. The proposed estimation algorithm builds on a longitudinal vehicle dynamics model in a highway-driving scenario in which platooning is beneficial in terms of improved fuel consumption. This model-based estimator has little prior knowledge of the aerodynamic drag coefficient and rolling resistance of the vehicle. Essentially a recursive least squares estimator, the algorithm repeatedly estimates the aerodynamic drag coefficient in order to provide the most updated vehicle dynamics parameters during on-road operation.

The accuracy of the algorithm is quantified by a point cloud of 1,000 independent simulation runs involving with white and Gaussian measurement noise. The estimation quickly converges (within 50 seconds) to ±2 % of the true value of aerodynamic drag coefficient. Simulations with different road profiles demonstrate the robustness of the estimator.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Before doing the research, it was even unimaginable for me to think about what research really is. Through the two-year period of experiencing in the research lab, not just the creativity and learning that are emphasized, perseverance and time management are the more cherished merits. It is Dr. Hosam Fathy, a truly respected mentor, who has been at the balance of challenging and encouraging on my work consistently, led me through the up-and-downs, and lightened the path in front of me. His advice on my professional development and personal takeaways based on his living philosophies has significantly contributed to the foundations of my success as an undergraduate at Penn State, and have profoundly influenced my perceptions on the society and world. Therefore, I want to sincerely thank Dr. Hosam Fathy for his empathetic care to his students and his awareness of creating an international laboratory.

The course ME 497H has provided sufficiently valuable information of composing a well-written thesis. Four incredible instructors of the course, Dr. Jacqueline O'Connor, Dr. Daniel Cortes, Dr. Stephanie Stockar and Dr. Hosam Fathy, had generously given their wisdom on doing research in general and writing techniques, as well as professional development for graduate education, which also have tremendously benefited me on the writing process later.

The Control Optimization Laboratory is an extremely diverse and open-minded working place filled with the brightest minds in the scientific community. Every person in this lab is willing to express him/herself for their represented culture background, and to provide help to each other whenever it is needed. Therefore, I would like to thank the entire COOL lab and recent graduates: Michelle Denlinger, Partha Mishra, Chu Xu, Manuel Rodriguez, Miguel Alvarez, Mahsa Doosthosseini, Adilah Aljunid, Xiaoling Chen, Timothy Cleary, Manasi Nair, Deba Bhattacharjee, and Gift.

Chapter 1

Introduction

1.1 Statement of Objectives

The overarching goal of this project is the assessment of the quickness and accuracy of a simulated real-time onboard estimator of the aerodynamic drag coefficient of a heavy-duty truck. To achieve this goal, the thesis proposes an algorithm that constructs a recursive least squares-based estimator to particularly capture the aerodynamic drag coefficient of the modeled the heavy-duty truck in a high-way driving scenario. This goal translates into the following objectives:

1. Conduct a literature review on the importance and utilization of vehicle chassis parameter estimation in the context of heavy-duty vehicles and intravehicular technologies, specifically vehicle platoon.
2. Examine the concept of recursive least squares estimation and simple linear least squares estimation, especially in the field of vehicle chassis parameter estimation.
3. Construct a longitudinal vehicle dynamics model of a heavy-duty vehicle which is the building block for the aerodynamic drag coefficient estimation, and use the model to generate simulated vehicle's velocity profiles.
4. Apply a recursive least squares estimation to the vehicle dynamics model in order to assess the value of aerodynamic drag coefficient for a designated period of time.
5. Define ±2% accuracy bounds of the theoretical value for the aerodynamic drag coefficient, and employ the linear least squares estimation on a series of simulated drive cycles with

different random noises. The group of the results from the linear least squares estimation quantifies the accuracy represented by a point cloud.

1.2 Motivation

The road transport sector contributes a significant amount of the total energy consumption and emission from the sum of all modes of transportation. According to the European Commission, the percentage of greenhouse gas emissions from road transport compared to all transport sectors was 72.9% in 2015 [1]. From an economical perspective, particularly essential for heavy-duty vehicles (HDVs), these vehicles often bear considerable energy loss because of their burdensome weight and inefficient aerodynamic design mostly from the semi-trailer. As a result, the fuel cost can consist up to 35% of the entire operation costs of HDVs [2]. In the meantime, a typical HDV can consume one fourth of the overall fuel consumption on merely counteracting the aerodynamic force, especially at high speed [3]. Therefore, there is a rising need to reduce the energy consumptions of HDVs by exploiting different methods and technologies.

Among the potential technologies of vehicle connectivity and automation that have positive benefits on fuel savings from HDVs, vehicle platooning contributes substantially to reducing energy consumption during highway driving scenarios [4]. Vehicle platooning is the idea of forming a train-like convoy of vehicles during on-road operation, especially in the situation of high-speed and highway driving. This is appealing to the field of heavy-duty trucks because, on the one hand, a substantial amount of fuel/energy is expended to overcome the air drag force from an HDV’s perspective, and on the other hand, in the United States, as well as true in Europe, most of road freight transport occurs on high-speed freeways [3]. The reason why platooning is effective on HDV’s fuel savings is because the decrease of the air drag force that is acting against the vehicles when traveling at close intravehicular distances. The air drag force associated with road-going vehicles can be modeled as:

$$F_{ad} = \frac{1}{2} C_d \rho A_f v^2 \tag{1}$$

where F_{ad} is the air drag force, C_d is the aerodynamic drag coefficient, ρ is the air density, A_f is the frontal area of the vehicle, and v is the vehicle's velocity. For a typical HDV, the value of the frontal area A_f is very significant, and the effective area is even larger when the semi-trailer is attached. The velocity v here is also at a high value during freeway driving, therefore the square of this number resulting in an extensive impact on the air drag force. Therefore, assuming the air density ρ is almost a constant number, the aerodynamic drag coefficient can dictate the magnitude of the experienced drag force. By its design, the value of a heavy-duty truck's aerodynamic drag coefficient is high, often ranging from 0.5 to 0.9 [5], and is almost twice as that of an average passenger sedan. Consequently, the air drag force F_{ad} can be undesirable when a single heavy-duty truck travels at highway speed, with aerodynamic drag occupying a considerable portion of the engine output [6], thus increasing overall energy consumption. The platooning formation, however, has the benefit of reducing the aerodynamic drag coefficient: when vehicles, in general, are moving at close intervehicular distances, the pressure difference between the front and back of the vehicle diminishes [3], accordingly generating a discounted value of the aerodynamic drag coefficient associated with a more optimal fuel performance. From road experiments, the fuel savings can be as high as 16% from the platooning formation [7]. Therefore, along with the reduction in fuel consumption, the concomitant outcomes from platooning include less greenhouse gas emissions and better cost-effective solutions for the road freight transportation.

In order to enable the platooning formation, vehicle connectivity technologies need to be implemented: a closed-loop controller, for instance, such as a cooperative adaptive cruise control architecture for a heavy-duty vehicles convoy. The performance and viability of a controller depends on the accuracy of the vehicle model used for the controller's design [8]. The underlying model is represented by mathematical formulations including ordinary and partial differential equations that have constant terms which are the vehicle parameters including the vehicle's aerodynamic drag coefficient. In terms of the accuracy of the vehicle model, the parameters are the important factors that subsequently have impacts on the performance of the closed-loop system. Nevertheless, these parameters significantly vary in their values based on the environment/outside conditions: in this case, the aerodynamic drag coefficient of a heavy-duty truck can, in a fleet, fluctuate in a relatively large range dependent on the intervehicle distances and vehicles' own aerodynamic designs. For the automated control of HDVs, in the context of platoon, the acceleration capability (engine power output) of an HDV can be used in three major categories: 1) road loads that include road grade, rolling resistance and air drag; 2) maneuvering that is following the route and trajectory; 3) control authority that is correcting for the spacing or parameter updates/errors [6]. The three areas are mutually related in the sense that when an HDV is tracking the velocity trajectory in a platoon, it, meanwhile, needs to counteract the air drag in order to fulfill the platoon control demand, for example, correcting the intervehicular distances. The utilization and arrangement of engine output, from an HDV's perspective, is critical because these vehicles have comparatively low power to weight ratio [6]. Hence, to achieve the best controllability of an HDV's platoon, there is a need to have an accurate knowledge of the vehicle parameters, such as the aerodynamic drag coefficient, in order to minimize the unnecessary engine effort consumed by inaccurate parameters and to safeguard the platoon not to separate under various external conditions. Therefore, an accurate estimator is required to obtain and provide the knowledge for the aerodynamic drag coefficient, which ultimately has implications on the energy consumptions of heavy-duty vehicles.

1.3 Background

Estimation, either of the system states or the model parameters, is prominent in the area of identification, which consists of four major components: model structure/problem formulation, experiment design, parameter estimation and validation [9]. Especially in the context of adaptive control, parameter estimation is an important part to update the magnitude of the varying parameters during on-line operation. Estimation and adaptive control are used extensively in aerial vehicle's control, but ground vehicles have quickly adopted the techniques as automation gradually penetrates in the ground transportation sector. The next chapter provides a literature review of a brief summary of parameter estimation, its importance in identification and adaptive control, its applications in different fields with emphasis on vehicle chassis parameter estimation, the introduction of simple linear least-squares estimation, and the extension of linear least-squares estimation to recursive least-squares estimation.

Chapter 2

Literature Review

2.1 Parameter Estimation Overview

Identification is a mathematical model-constructing procedure for dynamic systems (i.e., systems with memory of the past) based on observations (measured/collected data) [10]. The first step of the identification process is problem formulation, i.e., using a mathematical representation/model to capture the dynamics of a given system, such as a vehicle's longitudinal motion [9]. The second phase is the experiment design phase that consists of selecting the right/proper input signals to perturb, choosing the outputs to observe, examining actuators, instruments, access and operating constraints, picking the type and parameters of perturbation, and designing sampling schedule [10]. The third component is the parameter estimation as the states of the target system and parameters of a model can vary continuously, or the states and parameters are unknown and have little prior knowledge from the onset of the system. Finally, the last element is validation, i.e., checking the accuracy of the estimation, adequacy of the structure, and the performance of the model. Therefore, in the context of platoon control, parameter estimation, especially in real-time, is necessary when there is little prior knowledge about the aerodynamic drag coefficient and the magnitude of the coefficient alters from time to time.

The control scheme of a vehicle platoon falls into the area of adaptive control and cooperative adaptive control, in which the vehicles need to communicate with each other to adjust their own condition to compensate for external disturbances. In an adaptive control structure, the parameters of the model are updated recursively at each sampling period and the revised parameter values are fed into the controller design. In a graphical illustration, the adaptive controller consists of two loops [9], as shown in Figure 1. The inner loop is the feedback controller, whereas the outer loop is the parameter estimation. The block diagram is one specific type of adaptive systems, a self-tuning regulator that is able to update its own internal parameters in order to optimize the objective output. The component of the parameter estimation in this diagram demonstrated where the parameter estimation lies in a control scheme, and also indicates where an aerodynamic drag coefficient estimator fits in an HDV platoon controller.

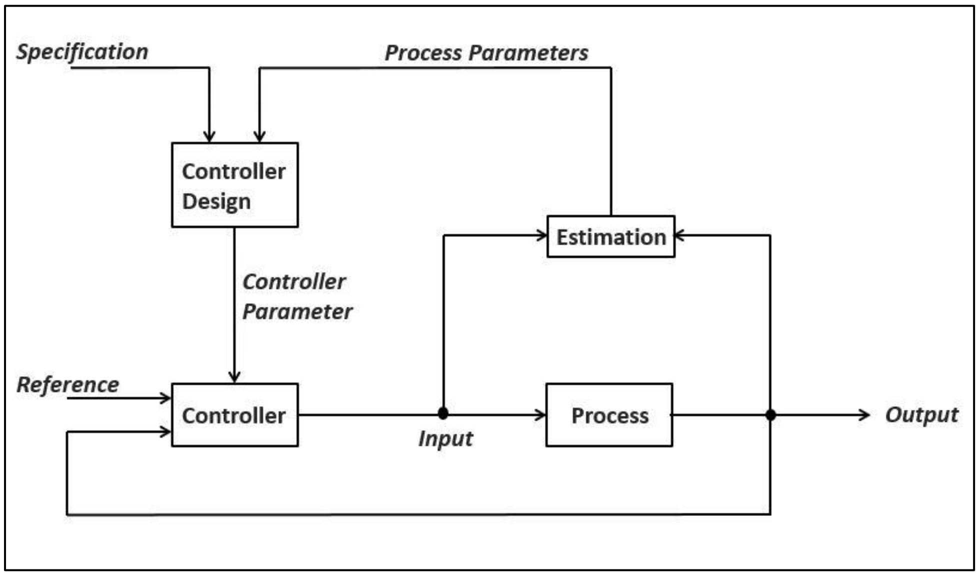

Figure 1. A Block Diagram of A Self-Tuning Regulator with A Parameter Estimation Feature

2.2 Parameter Estimation and Its Applications to Automotive Systems

Parameter estimation is a well-studied subject as different estimating techniques have been applied in many engineering and scientific applications, such as spacecraft dynamics, underwater vehicles and agriculture applications [10]. When control algorithms were initially introduced in the aerospace industry to ensure flight precision and reliability, parameter estimation, especially a maximum-likelihood technique [11], was widely utilized to estimate aircraft coefficients, such as flight roll rate and angle of sideslip. Parameter estimation typically assumes an underlying well-understood mathematical model: as the models for aircraft/spacecraft are already derived based on physics model and experimentally tested [11], McLaughlin *et al.* employ the known atmospheric density models by setting up an established batch weighted least squares estimator to examine the drag coefficient of Earth satellites [12]. As computation power advances, Barati *et al.* are able to use multi-gene Genetic Programing (GP), in the field of fluid mechanics, to develop empirical models for estimating the drag coefficient of flow around a smooth sphere [13].

The applications of parameter estimation in the aerospace industry and fluid mechanics encourage the extension of estimation techniques in automotive engineering: on the one hand, a rich variety of sensors and image processing devices are gradually getting added to vehicles' equipment in order to address multitudinous driving situations; on the other hand, the development of vehicle-to-vehicle and vehicle-to-infrastructure technologies enables communications and cooperative driving strategies. As these advancements present the possibility of obtaining better and more complete knowledge of the vehicle's status and its surroundings, they have leverage in automotive safety and vehicle's controllability: parameter estimation in this sense enables vehicle to know its conditions and status quo better. On example is that Ahlawat *et al.* compare the accuracy of two different experiment methods with the same instrumentations but different inputs, coast-down and force measurement method, using linear regression to estimate the road load from, which includes aerodynamic drag and rolling resistance [14]. The force method generates more accurate results than the coast-down method, however demanding heavy instrumentations with associated high cost. Similar efforts to account for vehicle's road load include Bae *et al.*'s utilization of Global Positioning Systems to calculate the real-time road grade. The resulting road grade measurements are then used to estimate the vehicle's mass, rolling resistance and aerodynamic drag [6]. Vehicle mass has been an early focus in the field of vehicle parameter estimation as the variation in mass has profound influence on vehicle active safety (i.e., vehicle stability control), predominantly for heavy-duty vehicles. Fathy *et al.* investigate an online estimator that seeks on-road vehicles' mass with minimal instrumentation needs, also providing conservative mass error estimates [8]. Likewise, online estimation is an essential attribute for time-varying parameters, such as rolling resistance and tire traction, providing the updates value at each sampling period. Vahidi *et al.* [15] adopt recursive least square estimation with multiple forgetting to address the issues related to different changing rates of different vehicle parameters, vehicle mass and road grade. Vehicle mass estimation is needed for the implementation of electronic stability control unit [8]. Meanwhile, more attention is cast upon real-time aerodynamic drag and rolling resistance estimation as there is an increasing need to optimize vehicle energy management and control vehicle platoon as well [14]. Zhang *et al.* develop a model-based estimator with a supervisory data extraction scheme to specifically assess air drag load and rolling resistance [16]. In terms of the accuracy of the above estimation algorithms, it is dependent on many factors such as the specific experimental design and underlying vehicle model. Kandel *et al.* examine the effect of terrain variability on chassis parameter identifiability for an HDV using Fisher information analysis, concluding that a large road grade variability, a rich dataset in other words, increases estimation accuracy [17].

The overall literature review shows that significant works have been done in automotive industry to improve parameter estimation results to meet different needs. This thesis focuses on the simulation in the recursive least square estimation of a heavy-duty vehicle’s aerodynamic drag coefficient to provide real-time knowledge for the drag value for the purpose of future implementation of adaptive HDV’s platoon control. The following sections demonstrate simple linear least-squares estimation and recursive least squares estimation of aerodynamic vehicle drag, and present the formulation and execution of such algorithms.

2.3 Linear Least-Squares Estimation Formulation

When the system is linear in the parameters, one of the most common approaches for parameter estimation is linear least-squares estimation [9]. The following example of a least-squares estimation is based on [9]. Take the following mathematical model as the starting point:

$$y(i) = \varphi_1(i)\theta_1^0 + \varphi_2(i)\theta_2^0 + \cdots + \varphi_n(i)\theta_n^0 \tag{2}$$

where y is the observed variables, or output measurement, θ_1^0, θ_2^0,…, θ_n^0 are the parameters to be estimated, and $\varphi_1(i)$, $\varphi_2(i)$,…, $\varphi_n(i)$ are the known functions/values that may depend on other known variables, also known as regressors/regression variables in the other term. The overall mathematical model is also referred to as a regression model. If the above equation is written in vector form, the vectors are introduced as:

$$\varphi^T(i) = [\varphi_1(i) \quad \varphi_2(i) \quad \cdots \quad \varphi_n(i)]$$

$$\theta^0 = [\theta_1^0 \quad \theta_2^0 \quad \cdots \quad \theta_n^0]$$

Then, the equation is represented as $y(i) = \varphi^T(i)\theta^0$. The model time index is variable i, which indicates the assumption of discrete set. From an experiment, the pairs of observations $y(i)$ and regressors $\varphi^T(i)$ are recorded in the form of $\{y(i),\ \varphi^T(i), i = 1, 2, \ldots, t\ \}$.

The goal here is to evaluate the parameters such that the outputs computed from (2) concur as much as possible with the observed variable $y(i)$ under the notion of least squares: the parameter θ should be picked in order to minimize the least-squares loss function below:

$$V(\theta, t) = \frac{1}{2}\sum_{i=1}^{t}(y(i) - \varphi^T(i)\theta)^2 \tag{3}$$

Then, the following terms in vector form are introduced:

$$Y(t) = [y(1) \quad y(2) \quad \cdots \quad y(t)]^T$$

$$E(t) = [\varepsilon(1) \quad \varepsilon(2) \quad \cdots \quad \varepsilon(t)]^T$$

$$\Phi(t) = \begin{bmatrix} \varphi^T(1) \\ \vdots \\ \varphi^T(t) \end{bmatrix}$$

$$P(t) = (\sum_{i=1}^{t} \varphi(i)\varphi^T(i))^{-1} = (\Phi^T(i)\Phi(i))^{-1} \quad (4)$$

Among these terms, $\varepsilon(i)$ are the residuals defined as:

$$\varepsilon(i) = y(i) - \hat{y}(i) = y(i) - \varphi^T(i)\theta$$

With previously specified vector forms in (4), the residual vector can also be written as:

$$E = Y - \hat{Y} = Y - \Phi\theta$$

Therefore, the loss function can be reconstructed using the above predefined terms in (4):

$$V(\theta, t) = \frac{1}{2}\sum_{i=1}^{t} \varepsilon^2(i) = \frac{1}{2}E^T E = \frac{1}{2}\|E\|^2 \quad (5)$$

The least-squares estimation theorem states that the loss function of Eq. (3) is minimal for parameter $\hat{\theta}$ such that:

$$\Phi^T\Phi\,\hat{\theta} = \Phi^T Y \quad (6)$$

If the condition that the matrix $\Phi^T\Phi$ being nonsingular is satisfied, then the minimum is unique and given by:

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T \mathrm{Y} \quad (7)$$

In order to prove the least-squares estimation theorem, some further manipulations need to be worked upon Eq, (5), rewritten as:

$$\begin{aligned} 2V(\theta, t) &= E^T E = (Y - \Phi\theta)^T (Y - \Phi\theta) \\ &= \mathrm{Y}^T Y - \mathrm{Y}^T\Phi\theta - \theta^T\Phi^T Y + \theta^T\Phi^T\Phi\theta \quad (8) \end{aligned}$$

The loss function V has a minimum because the matrix $\Phi^T\Phi$ is always nonnegative definite. The minimum can be found by completing the square in the following form:

$$\begin{aligned} 2V(\theta, t) &= \mathrm{Y}^T Y - \mathrm{Y}^T\Phi\theta - \theta^T\Phi^T Y + \theta^T\Phi^T\Phi\theta + \mathrm{Y}^T\Phi(\Phi^T\Phi)^{-1}\Phi^T Y - \mathrm{Y}^T\Phi(\Phi^T\Phi)^{-1}\Phi^T Y \\ &= \mathrm{Y}^T(I - \Phi(\Phi^T\Phi)^{-1}\Phi^T)Y + (\theta - (\Phi^T\Phi)^{-1}\Phi^T Y)^{-1}\Phi^T\Phi(\theta - (\Phi^T\Phi)^{-1}\Phi^T Y) \quad (9) \end{aligned}$$

The vector θ here has no impact on the first term of the right-hand side in Eq. (9), but the second term is dependent on θ. Therefore, in order for the loss function V to be minimized, the second term should be zero, and subsequently the value of θ can be obtained:

$$\theta - (\Phi^T\Phi)^{-1}\Phi^T Y = 0$$

$$\theta = \hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T \mathrm{Y}$$

which is the same equation as Eq. (7). In the remainder of this thesis, least-squares estimaiton is employed for aerodynamic drag coefficient estimation. It also serves as a foundation for the use of recursive least-squares estimation to realize simulated real-time estimation.

2.4 Recursive Least-Squares Estimation Formulation

Linear least-square (LLS) estimation method becomes manageable when the data sets across a specific time span are collected and ready to be used, essentially an offline method as LLS cannot generate while the measurement data is being collected during operation. In this case, the offline method might not be the most efficient way as data collection or observations can be time-consuming. There is a need to obtain the parameter estimation in real time operation, especially in an adaptive control environment. Therefore, the objective becomes that developing a computationally efficient approach to estimate the parameter using the most updated data online. Recursive least-squares estimation is a rearranged form of the traditional least-squares, but recursively using the previous estimate to update the parameter value for the next time step. In another word, the observations and outputs at time $t-1$ are used to estimate the parameters at time t. In this case, $\hat{\theta}(t-1)$ denotes the least-squares estimate based on the obtained data at time $t-1$. The solution to the loss function minimization in Eq. (7) can be rewritten in the recursive form:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1)) \qquad (10)$$

where

$$L(t) = P(t)\varphi(t) = P(t-1)\varphi(t)(I + \varphi^T(t)P(t-1)\varphi(t))^{-1} \tag{11}$$

and

$$P(t) = \left(I - L(t)\varphi^T(t)\right)P(t-1) \tag{12}$$

Here, $P(t)$ is the covariance matrix. In Eq. (10), the parameter value $\hat{\theta}(t)$ is updated at each time step by adding a correction to the error that is between the measured outputs of $y(t)$ and the predicted outputs of $y(t)$ based on the parameter estimate at the previous time step $t-1$. The correction at every time step is proportional to the term $y(t) - \varphi^T(t)\hat{\theta}(t-1)$. The covariance matrix $P(t)$ can be considered the slope of the proportionality, essentially a weighting factor that dictates on how much the correction should be added on the previous estimate. The recursive least-squares estimation (RLS) is the centerpiece for estimating the aerodynamic drag coefficient during simulations in this thesis, and recursively updating the parameter represents an on-line and real-time estimation technique.

Chapter 3

System Model and Drive Cycle Formulation

In this Chapter, a vehicle longitudinal dynamics model is constructed to build the foundation for the aerodynamic drag coefficient estimation. In order to perform parameter estimation, inputs for the model need to be collected from a simulated drive cycle in which the data is produced. Hence, the following sections are organized as two components: the longitudinal vehicle dynamics model formulation and the sample drive cycle generation for the parameter estimation simulation.

3.1 Longitudinal Vehicle Dynamics Model Formulation

The aerodynamic drag coefficient estimation algorithm is constructed based on the longitudinal vehicle dynamics model. Table 1 provides a nomenclature for the model in this study.

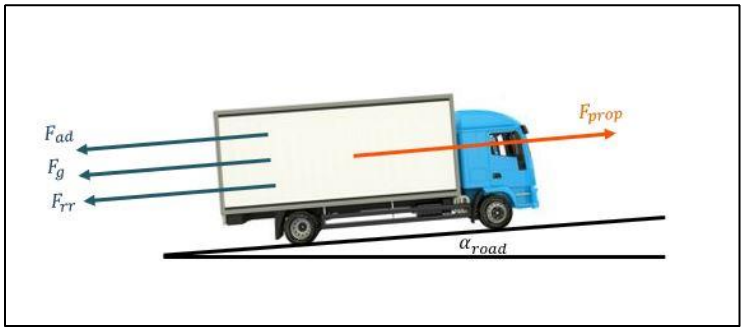

Figure 2. An HDV’s Experienced Forces in Longitudinal Dynamics Model

A vehicle longitudinal dynamics model is a Newtonian equation of motion describing a vehicle’s longitudinal motion, consisting of the vehicle’s propulsion force, braking force and vehicle road load that includes the aerodynamic drag, rolling resistance and force resulting from the road grade [16]. The vehicle’s braking events are not considered in this thesis, thus assuming the braking force to be zero throughout the simulation study as the significant change in tire dynamics during braking that is not in the scope of the thesis. The free-body diagram of the system and the longitudinal forces are shown in Figure 2. Then, the vehicle longitudinal dynamics model is described in the following equation:

Table 1. Vehicle Parameters with Nominal Values

Parameter	***Description***
M	vehicle mass with no trailer [8800 kg]
g	gravitational acceleration constant [9.81 m/s2]
$\boldsymbol{\rho_{air}}$	ambient air density [1.275 kg/m3]
$\boldsymbol{C_d}$	aerodynamic drag coefficient [0.65]
$\boldsymbol{A_f}$	frontal area of the vehicle [5 m2]
$\boldsymbol{C_r}$	rolling resistance coefficient [0.0006]
$\boldsymbol{\alpha_{road}}$	road grade [rad]
$\boldsymbol{U_{veh}}$	vehicle longitudinal velocity [m/s]
$\boldsymbol{F_{prop}}$	vehicle traction/propulsion force [N]

$$M\dot{U}_{veh} = F_{prop} - F_{ad} - F_{rr} - F_g \tag{13}$$

where F_{prop} is a simulated lumped term that represents the traction force generating on the vehicle wheels. In Eq. (13), F_{ad}, F_{rr} and F_g are the aerodynamic drag, rolling resistance and force caused by road grade and are given by:

$$F_{ad} = \frac{1}{2} C_d \rho_{air} A_f U_{veh}^2 \tag{14}$$

$$F_{rr} = C_r M g cos(\alpha_{road}) \tag{15}$$

$$F_g = M g sin(\alpha_{road}) \tag{16}$$

Including these force terms, Eq. (13) can be rearranged as the format below:

$$\dot{U}_{veh} = \frac{1}{M}[F_{prop} - \frac{1}{2} C_d \rho_{air} A_f U_{veh}^2 - C_r M g cos(\alpha_{road}) - M g sin(\alpha_{road})] \tag{17}$$

This longitudinal dynamics model, a state space representation fundamentally, is then implemented to generate the simulated velocity trajectory, and solved using MATLAB ode45 differential equation solver, provided with time histories of the traction/propulsion force F_{ad} and road grade α_{road}. With further arrangements to Eq. (17) in Chapter 4, least squares and recursive least squares formulations are formulated to conduct the parameter estimation on vehicle's aerodynamic drag coefficient.

3.2 Generating Simulated Data for Propulsion Force and Road Grade

The generation of the time histories data for traction force and road grade are designed and simulated with an intention to provide a rich set of simulation studies. Although different from real driving scenarios, while the simulated propulsion force exhibits abrupt step changes over time, the simulated road grade consists of a mixture of sinusoidal, linear and step variations. Figure 3 and Figure 4 show examples of the simulated propulsion force and road grade respectively.

The above traction/propulsion force and road profile time histories are corrupted with independent random noise signals, by using *randn* command in MATLAB, to represent sensor noise effects. This helps with assessing estimator robustness and accuracy.

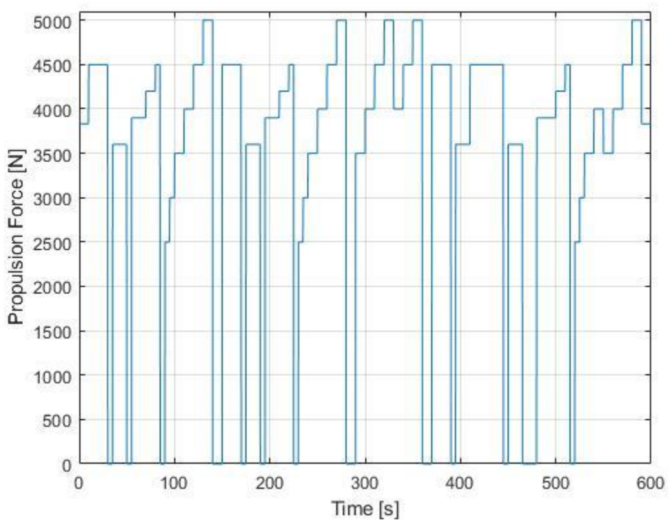

Figure 3. Sample Vehicle Propulsion Force

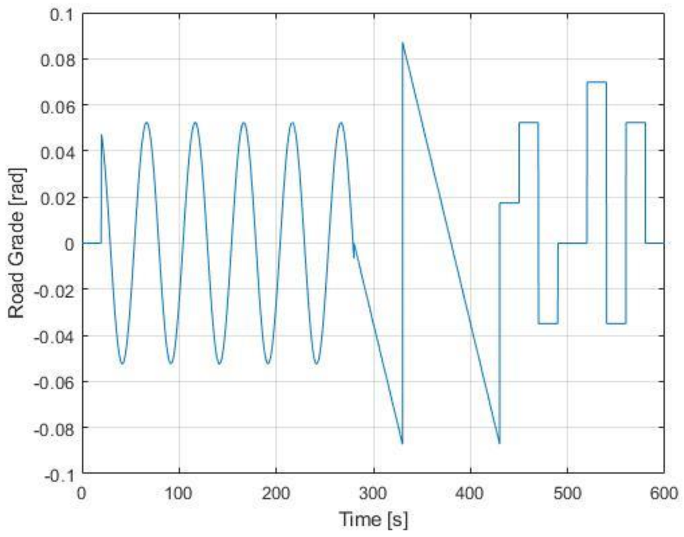

Figure 4. Sample Road Grade Profile

3.2 Producing Simulated Velocity Trajectory/Drive Cycle

The simulated propulsion force history and terrain profile are generated as inputs into the vehicle longitudinal dynamics model Eq. (17) to obtain time histories of the vehicle velocity trajectory. Eq. (17) is solved using MATLAB ode 45 function to generate the data for the vehicle velocity U_{veh} as shown in Figure 5.

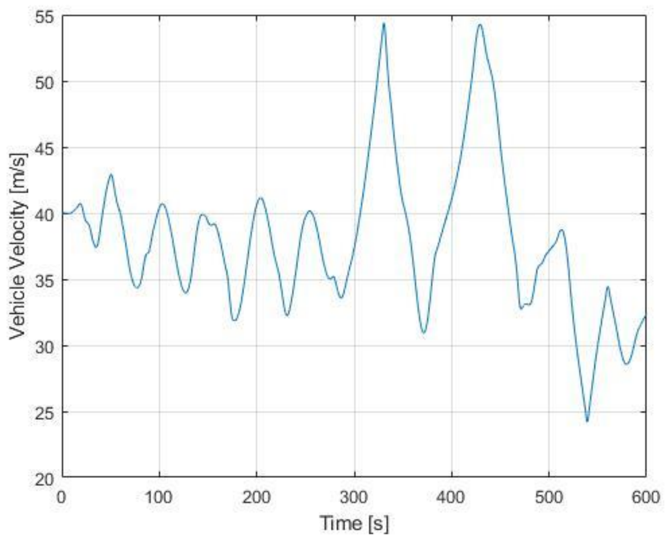

Figure 5. Simulated Vehicle Longitudinal Velocity Trajectory

After producing the simulated propulsion force and road grade profile along with the resulting vehicle velocity trajectory, one last observation term, the vehicle acceleration $\dot{U}_{veh}$, is needed for aerodynamic drag coefficient estimation. In an experimental setting, the acceleration $\dot{U}_{veh}$ is obtained from the sensors, but for the purpose for this simulation study $\dot{U}_{veh}$ can also be obtained directly from calculating Eq. (17) when all vehicle parameters are assumed to be known, and the propulsion force, terrain profile and velocity trajectory are fed in the equation:

$$\dot{U}_{veh} = \frac{1}{M}[F_{prop} - \frac{1}{2}C_d\rho_{air}A_fU_{veh}^2 - C_rMgcos(\alpha_{road}) - Mgsin(\alpha_{road})]$$

The propulsion force, road grade, vehicle velocity and acceleration are the observations or measurements to formulate the measurement outputs and regressors in the parameter estimation scheme. Similar to introducing sensor noise to the traction force and road grade measurements, independent and random noises are also considered in the vehicle's velocity profile and acceleration in Chapter 4.

Chapter 4

Least-Squares Estimation Problem Formulation

In this chapter, both linear least-squares and recursive least-squares estimation techniques are utilized to perform parameter estimation for two vehicle parameters: the aerodynamic drag coefficient C_d and rolling resistance coefficient C_r. The formulation of the estimation algorithm follows the established least-squares estimation procedures in Section 2.3. The conditions and assumptions for the linear least-squares estimation of longitudinal vehicle dynamics model parameters are discussed below.

1. The vehicle longitudinal dynamics model represented by Eq. (17) is assumed to be an accurate enough model to largely capture the dynamics of a longitudinally moving vehicle. In addition, for this specific simulation study, the drag coefficient C_d is assumed to be constant or slowly time varying.
2. The measurement noise signals added to vehicle propulsion force, road grade, velocity and acceleration are assumed to be independent and identically distributed with zero means—i.e., white noise. In addition, different magnitudes of the noise correspond to the different measured signals. For instance, the scale of the noise existing in propulsion force measurement ought to be larger than that of the noise related to the road grade observations, since the typical numerical value of the propulsion force is larger. As this assumption stands, the assumption of which all relevant inputs and outputs have available data is also met.
3. As the name of linear least-squares estimation indicates, the estimation model is linear in the parameters C_d and C_r.

These assumptions and conditions can hold true because of the design of this simulation study. However, many of them can also fail to be satisfied when estimating the parameters from actual experimental data. A considerably contributing factor is that, although the vehicle longitudinal dynamics model is simple and well-studied, the model does not capture the full representation of the rolling resistance when the vehicle's tires are either over or under inflation, and of vehicle's aerodynamics when the wind disturbance occurs. In terms of the measurement noise, white noise that is independent and identically distributed with zero mean is ultimately an ideal world condition, but the sensor and instrumentation noise signals from actual data acquisition systems could be biased and correlated. Nevertheless, these assumptions and conditions are met in this specific simulation study.

4.1 Linear Least-Squares Estimation Problem Formulation

The parameter estimation algorithm for C_d and C_r is based on the formulation presented in Section 2.3 and implemented in MATLAB. First, the assumption of having white noise corrupting the sensor measurements is presented below:

$$F_{prop,n} = F_{prop} + noise1$$

$$\alpha_{road,n} = \alpha_{road} + noise2$$

$$U_{veh,n} = U_{veh} + noise3$$

$$\dot{U}_{veh,n} = \dot{U}_{veh} + noise4$$

where $F_{prop,n}$, $\alpha_{road,n}$, $U_{veh,n}$ and $\dot{U}_{veh,n}$ are the representations for noisy measurements of the propulsion force, road grade, velocity and acceleration respectively.

These noisy measurement terms are used in the following algorithm construction. Following Eq. (2), the measurement outputs are placed on the left-hand side, and the lumped terms of the regressors and desired parameters are grouped on the right-hand side, shown below:

$$F_{prop,n} - M\dot{U}_{veh,n} - Mgsin\left(\alpha_{road,n}\right) =$$

$$\left(\frac{1}{2}\rho_{air}A_f U_{veh,n}^2\right)C_d + (Mgcos\left(\alpha_{road,n}\right))C_r \tag{18}$$

where the estimation parameters C_d and C_r are factored out of their respective regressor terms. The arrangement of Eq. (18) is guided by the form $y(i) = \varphi^T(i)\theta^0$, as well as $Y = \Phi\theta$ in vector form. Therefore, the measurement output vector Y can be defined as the following:

$$Y = \begin{bmatrix} F_{prop,n;1} - M\dot{U}_{veh,n;1} - Mgsin\left(\alpha_{road,n;1}\right) \\ F_{prop,n;2} - M\dot{U}_{veh,n;2} - Mgsin\left(\alpha_{road,n;2}\right) \\ \vdots \\ F_{prop,n;t} - M\dot{U}_{veh,n;t} - Mgsin\left(\alpha_{road,n;t}\right) \end{bmatrix} \tag{19}$$

which is the entire time history of the measurement outputs at every instance of time for a designed time span in a vector form. In addition, the vector forms for regressors Φ and parameters θ in their time series are below:

$$\Phi = \begin{bmatrix} \frac{1}{2}\rho_{air}A_f U_{veh,n;1}^2 & Mgcos\left(\alpha_{road,n;1}\right) \\ \frac{1}{2}\rho_{air}A_f U_{veh,n;2}^2 & Mgcos\left(\alpha_{road,n;2}\right) \\ \vdots & \vdots \\ \frac{1}{2}\rho_{air}A_f U_{veh,n;t}^2 & Mgcos\left(\alpha_{road,n;t}\right) \end{bmatrix} \tag{20}$$

$$\vec{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} C_d \\ C_r \end{bmatrix} \tag{21}$$

Organizing Eq. (19), (20), (21) in an expanded form of Eq. (18) is shown as:

$$\begin{bmatrix} F_{prop,n;1} - M\dot{U}_{veh,n;1} - Mgsin\left(\alpha_{road,n;1}\right) \\ F_{prop,n;2} - M\dot{U}_{veh,n;2} - Mgsin\left(\alpha_{road,n;2}\right) \\ \vdots \\ F_{prop,n;t} - M\dot{U}_{veh,n;t} - Mgsin\left(\alpha_{road,n;t}\right) \end{bmatrix} =$$

$$\begin{bmatrix} \frac{1}{2}\rho_{air}A_f U_{veh,n;1}^2 & Mgcos\left(\alpha_{road,n;1}\right) \\ \frac{1}{2}\rho_{air}A_f U_{veh,n;2}^2 & Mgcos\left(\alpha_{road,n;2}\right) \\ \vdots & \vdots \\ \frac{1}{2}\rho_{air}A_f U_{veh,n;t}^2 & Mgcos\left(\alpha_{road,n;t}\right) \end{bmatrix} \begin{bmatrix} C_d \\ C_r \end{bmatrix} \tag{22}$$

The estimation on $\vec{\theta}$ is calculated by Eq. (7) as $\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T\mathrm{Y}$, where $\hat{\vec{\theta}}$ is denoted for the estimates of $\hat{C}_d$ and $\hat{C}_r$. The procedure discussed in this section is how linear least-squares estimation is employed in this particular simulation study. The extension of the least-squares estimation is the recursive least-squares estimation in the next Section 4.2. Furthermore, the formulation of this linear least-squares estimation is the foundation for generating the point-cloud that accounts for the overall accuracy of the estimates in the next Chapter 5.

4.2 Recursive Least-Squares Estimation Problem Formulation and Results

Recursive least-squares estimation is based on simple linear least-squares estimation, but is more computationally efficient and faster than the latter one because RLS does not need to store the measurement data for the time span to estimate the parameters. Instead, the idea of RLS estimation is obtaining the current estimate just based on the previous estimate, which is the measurement information at the preceding time step. The formulation of this RLS estimation for parameters C_d and C_r follows the procedures from Eq. (10), (11), (12). In Eq. (11) and (12), the regression gain $L(t)$ is a function of the covariance matrix $P(t)$, which has influence on the parameter estimation. The covariance matrix needs to be initialized to some user-selected value for starting the RLS estimation process. The magnitude of this initial value determines the progression of future parameter estimates: a small initial covariance matrix corresponds to a higher likelihood of rapid departure from initial parameter estimates, whereas a large initial covariance matrix reduces that likelihood.

In order to visualize the influence on estimates by changing covariance values, an example study focuses on estimating just a numerical value of 0.006, the value of rolling resistance coefficient, with different initial covariance values. The example study actually uses the recursive least-squares estimation scheme formulated later in this chapter (i.e., RLS estimation algorithm steps (1) to (3)). The purpose here is to demonstrate the impact of covariance on the RLS estimation progress. Here, Figures 6 and 7 demonstrate a comparison of the estimation process for the same parameter using a small and large initial covariance matrix. On the one hand, the estimate quickly converges when having a small initial covariance matrix, although with larger initial fluctuations; on the other hand, the estimate takes longer to reach a steady value.

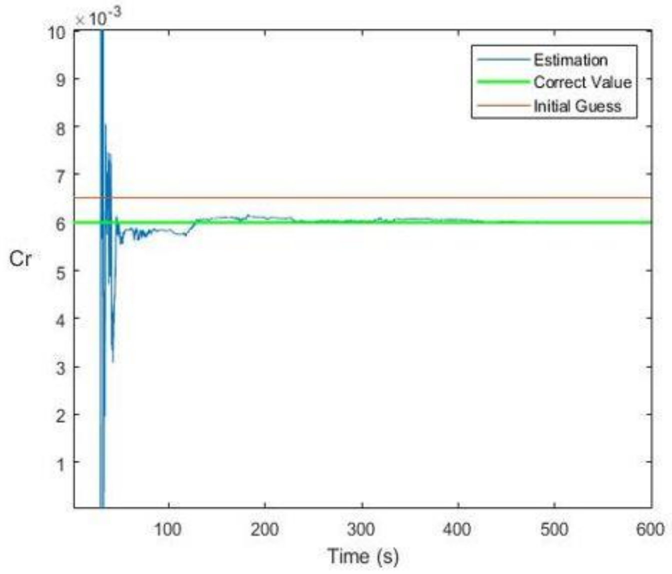

Figure 6. Recursive Least-Squares Estimation of 0.006 with A Small Initial Covariance Value

In this Figure 6, for the purpose of demonstration, the covariance value is set to be 0.1. The estimates fluctuate considerably for a period of time.

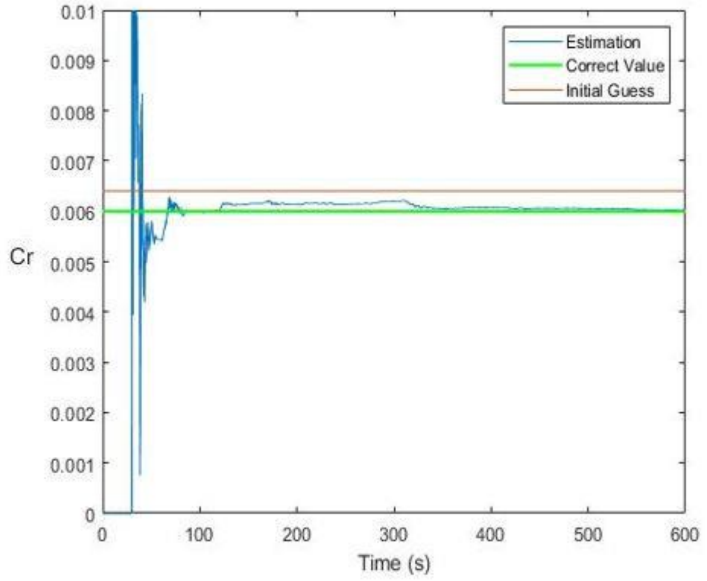

Figure 7. Recursive Least-Squares Estimation of 0.006 with A Large Initial Covariance Value

In Figure 7, in contrast, the covariance value is 10,000, which causes the estimates to have the tendency of remaining close to the initial guess of desired parameter. For quite a long period of time after the early convergence, the estimates are influenced by the initial parameter guess and slowly approach the correct estimate.

The size of covariance matrix, a diagonal matrix, is determined by the number of target parameters in the estimation simulation: for instance, if only one parameter θ is to be estimated, the covariance matrix in the RLS estimation algorithm is just $[P_1]$, where P_1 is the covariance value in this case; whereas if two parameters θ_1 and θ_2 are desired in the estimation, then the covariance matrix can be set to $\begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}$, where P_1 and P_2 are now the covariance values corresponding to θ_1 and θ_2 respectively. Because the desired parameters are C_d and C_r in the simulation study, the covariance matrix is pre-allocated as $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$,

In addition, RLS estimation also requires initial guesses of the interested parameters. To give values to the parameter guesses, depending on the scenarios, on the one hand, it is possible to borrow from commonsense and day-to-day observations: for instance, if the objective is to estimate the air density, it might be logical to have the initial guess of 1.2 kg/m^3 to start the RLS estimation. On the other hand, linear least-squares estimation can be applied to pre-existing datasets, producing a rough idea/knowledge for the target parameter. In this simulation study of estimating the vehicle's aerodynamic drag coefficient, the data in the first 30 seconds of the 600-second simulation are used to conduct linear least-squares estimation of parameters C_d and C_r. The resulting estimates are the parameter initializations for the subsequent RLS estimation. Congruently, the measurement output, regressor matrices Y and Φ, Eq. (19) and (20), are slightly modified as follows:

$$Y_{initial\ 30} = \begin{bmatrix} F_{prop,n;1} - M\dot{U}_{veh,n;1} - Mgsin\left(\alpha_{road,n;1}\right) \\ F_{prop,n;2} - M\dot{U}_{veh,n;2} - Mgsin\left(\alpha_{road,n;2}\right) \\ \vdots \\ F_{prop,n;30} - M\dot{U}_{veh,n;30} - Mgsin\left(\alpha_{road,n30}\right) \end{bmatrix} \tag{23}$$

$$\Phi_{initial\ 30} = \begin{bmatrix} \frac{1}{2}\rho_{air}A_f U_{veh,n;1}^2 & Mgcos\left(\alpha_{road,n;1}\right) \\ \frac{1}{2}\rho_{air}A_f U_{veh,n;2}^2 & Mgcos\left(\alpha_{road,n;2}\right) \\ \vdots & \vdots \\ \frac{1}{2}\rho_{air}A_f U_{veh,n;30}^2 & Mgcos\left(\alpha_{road,n;30}\right) \end{bmatrix} \tag{24}$$

The outputs of the initializing estimates can be different in every independent simulation run due to the random noise added to the measurement data.

After the initializations of the covariance matrix and parameter guesstimates, the terms $P(t-1)$ and $\hat{\theta}(t-1)$ from Eq. (10), (11) and (12) are implemented for the onset of RLS estimation. When $P(t-1)$ and $\hat{\theta}(t-1)$ become available, the RLS estimation algorithm is executed using the following logic from step a) to c):

a) $L(t)$, the updating gain for the parameter estimates at the next time instance, is firstly calculated as Eq. (11) is indicated:

$$L(t) = P(t-1)\varphi(t)(I + \varphi^T(t)P(t-1)\varphi(t))^{-1}$$

where I is the identity matrix and $\varphi^T(t)$, defined in Section 2.3, in is the regressor matrix at one time instance. $\varphi^T(t)$ can rewritten as:

$$\varphi^T(t) = \left[\frac{1}{2}\rho_{air}A_f U_{veh,n}^2(t) \quad Mgcos\big(\alpha_{road,n}(t)\big)\right]$$

$\varphi^T(t)$ for the next time instance is fed in the algorithm once the data becomes available.

b) When the updating gain $L(t)$ and measurement output term $y(t)$ become available online, by utilizing Eq. (10), the parameter estimates for next time step are recalculated as the following:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)(y(t) - \varphi^T(t)\hat{\theta}(t-1))$$

where $y(t)$ is the measurement output term at the next time step and can be written as the following:

$$y(t) = F_{prop,n}(t) - M\dot{U}_{veh,n}(t) - Mgsin\big(\alpha_{road,n}(t)\big)$$

Also, $\hat{\theta}(t-1)$ is obtained previously from linear least-squares estimation on the first 30 seconds. The new $\hat{\theta}(t)$ is utilized again for next cycle of estimation in a recursive manner.

c) In terms of updating covariance matrix, $P(t)$ for the next time step is a function of the current updating matrix $L(t)$ and the previous covariance matrix $P(t-1)$, Eq. (12):

$$P(t) = \big(I - L(t)\varphi^T(t)\big)P(t-1)$$

When the new covariance matrix $P(t)$ is computed and accessible, it is utilized again back in the step 1). When the connection and transition between the final step 3) and first step 1) are accomplished, the algorithm logic loop of RLS estimation completes and proceeds under an online scheme accordingly. The specific MATLAB code is included in the Appendix A.

In this simulation study, from time 30 seconds to 600 seconds, RLS estimation of parameters C_d and C_r is executed. The parameter estimation results are exhibited in Figure 8 and 9. Especially for the aerodynamic drag coefficient estimation, recursive least-squares technique produces desirable estimates with an accuracy of 2% in 50 seconds. The analysis on the quality of the estimation is discussed in the next Chapter 5.

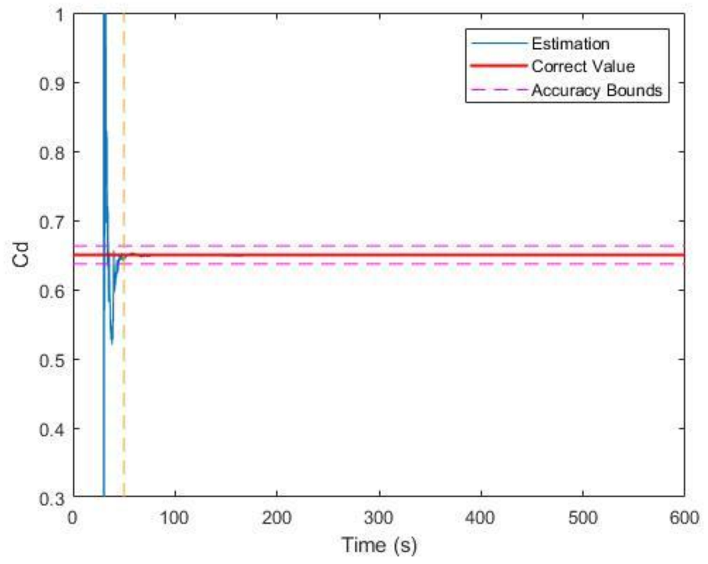

Figure 8. Estimation of Vehicle's Aerodynamic Drag Coefficient C_d Using RLS

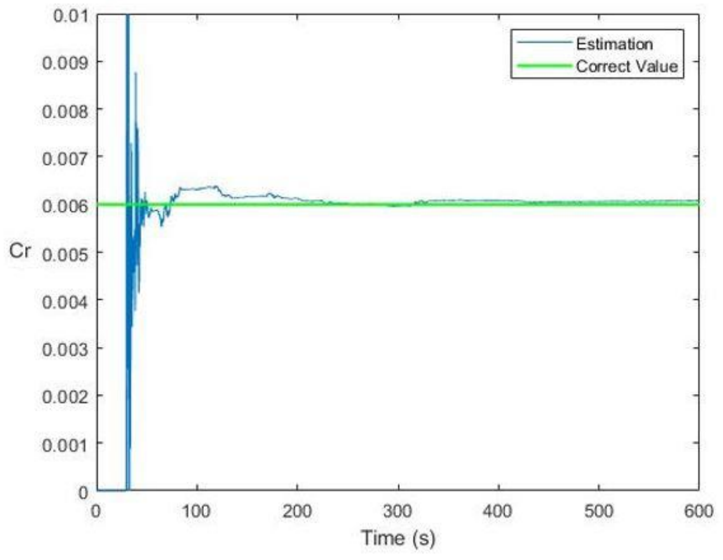

Figure 9. Estimation of Rolling Resistance Coefficient C_r Using RLS

Chapter 5

Estimation Results Quality Assessment

The ultimate goal of this simulation study is to develop an estimator algorithm that can estimate the vehicle's aerodynamic drag coefficient online within a ±2 % bound of its correct value and can in 50 seconds converge inside this desired bound. The specific choice of these bounds is application-dependent. In this work, this choice reflects the need for fast and accurate aerodynamic drag coefficient parameter estimation as a precursor to adaptive platooning. The following sections analyze the results produced in Section 4.2 and validate the estimator from the perspectives of Monte Carlo Simulations.

5.1 Accuracy Bound and Time Threshold Definition

In order to accomplish the established goal of having ±2% accuracy in the estimation process, a bound is identified first based on the true/proposed value of the vehicle's aerodynamic drag coefficient, of which C_d is designed to be 0.65 when constructing the longitudinal dynamics model in Chapter 3. Then, the lower and higher bounds of the RLS estimation process are simply defined as the followings:

$$Lower\ Bound = (1 - 2\%) * C_d \tag{25}$$

$$Higher\ Bound = (1 + 2\%) * C_d \tag{26}$$

where C_d here is the defined vehicle parameter, aerodynamic drag coefficient 0.65. The lower and higher bounds are graphed along with the RLS estimation in Figure 8, where the pink-dashed lines are ±2 % limits of the true/proposed C_d magnitude.

One advantage of recursive least-squares estimation when introducing the technique in Section 2.4 is its computational efficiency, which does not need the complete collection of data to initiate the estimation like linear least-squares estimation does. The RLS algorithm yet estimates the parameter at every time step, using the most updated data set. Hence, in order to validate and demonstrate the advantageous computation efficiency, RLS estimation needs to produce the desirable results within a necessary time limit: the time threshold for RLS estimation reaching inside the ±2 % accuracy bound is established to be 50-second time mark from the onset of RLS estimation. The 50-second time mark is represented by the orange-dashed line in Figure 8.

The ±2 % accuracy bound and 50-second target-time together substantiate the claim of developing an estimator that is accurate. In Figure 8, the estimates converge inside the ±2 % accuracy bound, spending less than 50 seconds. The result is generated by single simulation run, however the outcomes of other test runs showing a similar pattern. The point cloud in the next Section 4.2 is an alternative representation justifying the credibility of the estimator's accuracy.

5.2 Point-Cloud of Estimation Results Generation

A point-cloud is a graphic demonstration of a given estimator's accuracy. The point cloud in Figure 10 shows the results of linear least-squares parameter estimation for 1,000 simulations, i.e., the 600-second simulation study is repeated for 1,000 times. Similarly, the point cloud in Figure 11 is the group of estimation results from 1,000 recursive least-squares estimates. Every new simulation from both methods is unique because the white noise is recreated every time. Therefore, the purposes of producing the point-cloud are to test the estimator's robustness under noisy environment and to validate the accuracy of the estimator, especially the results in Section 4.1 and 4.2. The algorithm for generating the linear least-squares point-cloud is essentially the procedure composed in Section 4.1 as formulating the linear least-squares estimation, yet reiterating the simulation for 1,000 times: $\hat{\vec{\theta}} = (\Phi^T\Phi)^{-1}\Phi^T\mathrm{Y}$ for 1,000 iterations. In the meantime, the algorithm for producing the recursive least-squares point-cloud is guided by the estimator's design criterion of obtaining the aerodynamic drag coefficient estimate with ±2% accuracy in 50 seconds: the estimate at the 50th second is stored, and 1,000 estimates at this time point are collected from repetitive simulations. Therefore, in other words, these point-clouds represent Monte Carlo simulations of both least-squares and RLS estimator. While the simulated propulsion force F_{prop}, road terrain profile α_{road}, velocity trajectory U_{veh} and acceleration $\dot{U}_{veh}$ are kept unchanged, the noise for every set of measurement data is ultimately distinct in every simulation repetition. When the plotting the point-cloud, the result of the parameter estimates is placed on the graph after every completion of the linear least-squares and RLS estimation; until the 1,000th simulation iteration, 1,000 estimates of parameters C_d and C_r are all positioned on the same graph. Then, the point-clouds of parameters C_d and C_r is generated, and a scatter plot result is displayed in both Figures 10 and 11.

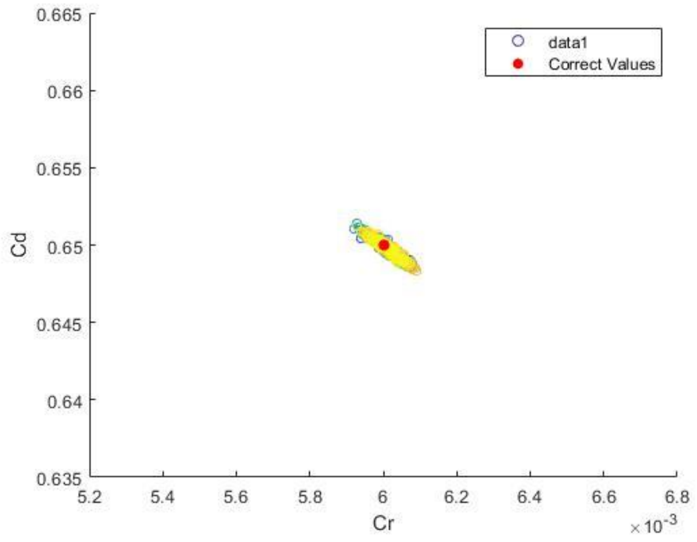

Figure 10. Point-Cloud of 1,000 Linear Least-Squares Estimation Results

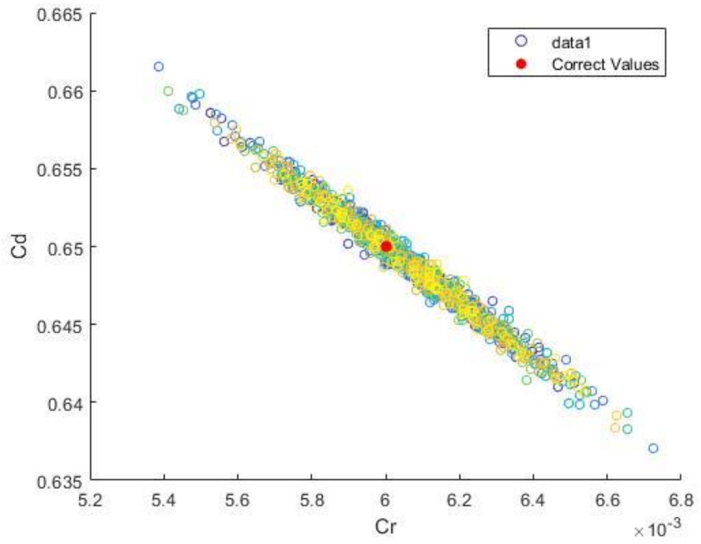

Figure 11. Point-Cloud of 1,000 Recursive Least-Squares Estimation Results

In both figures, the correct values of C_d and C_r are shown as a red dot, whereas the other circled points are estimates from 1,000 simulations. For the purpose of this simulation study, the focus is on the aerodynamic drag coefficient: from the y axis where all the possible values of 1,000 C_d estimates could locate, the extreme ends of the axis in Figure 10 are 0.648 and 0.652, which is ±0.002 deviation from the correct parameter value. ±0.002 deviation is well within the ±2 % accuracy bound proposed in the previous Section 5.1. The range in Figure 11 also shows a promising accuracy for RLS estimation as majority of the estimation results fall in the designed accuracy bound. Thus, the point-cloud again indicates and justifies both the robustness and accuracy of the proposed RLS estimator under noisy scenarios.

Chapter 6

Limitations of the Proposed RLS Estimator

Although the proposed RLS estimator has satisfied the design criteria in this specific simulation study, many limitations still exist and are discussed in the following perspectives:

1. A truly well-designed estimator can produce reliable results under various conditions and even when the underlying model can be falsely-assumed in extreme scenarios. In this case, the designed estimator by the simulation study has not yet proven its viability and functionality in real highway and platooning environments. Hence, at least experimental data needs to be implemented into the estimator in order to validate its performance.
2. In terms of the simulated measurement data, although intended to represent the reality, the generated velocity profile has impractical values of 50 m/s for a heavy-duty vehicle. In addition, the acceleration measurements in the study is directly calculated from Eq. (17), yet can be obtained from taking the time derivative of the velocity profile, which is a more representative method to collect acceleration data.
3. The aerodynamic drag coefficient in the simulation study is essentially a constant parameter, simplifying the overall estimation formulation. In realistic setting of a platoon consisting of heavy-duty vehicles, the drag coefficient C_d can vary in a great extent in terms of its own magnitude, based on the intravehicular distances. The frequency of C_d value changes would possibly be faster than the estimation time (50-second) from the estimator. Therefore, the proposed estimator might experience significant lags if the changes occur frequently. The limitation might be overcome by introducing a more sophisticated algorithm that incorporates forgetting factors to the recursive least-squares technique, similar to the work done by Vahidi

et al. [15]. Ultimately, the estimator should rapidly converge to the correct value under a constantly changing environment, e.g., on-road experiments.

4. The point-cloud introduced in Section 5.2 validates the performance of the estimator. However, the correct parameter values are not truly centered inside the scatter plot: a slight shift of the point-cloud might suggest the possibility of having biased estimates. The cause of the potential bias needs to be carefully examined and researched in the future work.

Chapter 7

Conclusion

This thesis proposes a recursive least-squares estimation algorithm that is accurate and fast enough to assess the vehicle's aerodynamic drag coefficient in a simulated online environment. Parameter estimation is a well-researched topic in broad areas including especially automotive industry. The ultimate goal of developing an online estimator for the drag coefficient is to employ such an estimator in heavy-duty vehicle platoon control, in which the drag coefficient might have profound impact on the optimal control demands. Therefore, the thesis is motived to construct a simple simulation study on the recursive least-squares estimation of vehicle load parameter. The outcome of the estimator satisfies the objective of reaching the correct value of drag coefficient in 50 seconds with ±2 % accuracy. The justification on the quality of the designed estimator is done by the generation of a Monte Carlo point-cloud of estimates. Some limitations of the proposed recursive least-squares estimator are discussed, which are the motivations and guidelines for the future work following the simulation study.

Appendix A

MATLAB Code for RLS Estimation with Initialization by Linear Least-Squares

```
% Qifeng Liu
% Schreyer Honor Thesis
% Recursive Least-Squares Estimation with Initialization by Linear
% Least-Squares Estimation

clc;
clear all;
close all;

%% Defined Known Variables
m = 8800; % vehicle mass [kg]
Af = 5; % frontal area [m^2]
rho = 1.275; % density of the air[kg/m^3]
g = 9.81; % gravity [m/s^2]

%% True (i.e., to-be-estimated) Air Drag Coefficient
Cd = 0.65; % air drag coefficient
mu = 0.006; % rolling resistance

%% ODE45 Formulation
v_0 = 40; % initial velocity
t_tot = 600; % simulation time [s]
time_step = 0.02;
tspan = 0:time_step:t_tot; % time range and step in ODE45
[t, v] = ode45(@(t,v) model_x(t,v,m,Af,Cd,mu,rho,g,v_0), tspan, v_0); % ODE45
that solve state-space equation at predetermined time step

%% Propulsion Force and Road Grade Extracted from Ode Functions
F_er = zeros(length(tspan),1); % pre-locate engine force
grade_r = zeros(length(tspan),1); % pre-locate road grade measurement
for jj = 1:length(tspan)
    t_in = t(jj); % time index
    F_er(jj) = engine(t_in,m,Af,Cd,mu,rho,g,v_0); % engine force points
extraction
    grade_r(jj) = grade_profile(t_in); % road grade points extraction
end

%% Acceleration
acc = (1/m)*(F_er -0.5*rho*Cd*Af*v.^2 - mu*m*g*cos(grade_r)-
m*g*sin(grade_r)); % acceleration directly from ODE function

%% Noisy Acquisition
s = length(tspan); % number of steps
F_n = F_er + 30*randn(s,1); % noisy engine force
ac_n = acc + 0.01*randn(s,1); % noisy acceleration
grade_n = grade_r + 0.001*randn(s,1); % noisy road grade
v_n = v + 0.1*randn(s,1); % noisy velocity
```

```
%% Prior knowledge on Cd and Mu before RLS (first 30 seconds linear least-
squares estimation)
ss = 30*(1/time_step)+1; % first 30 seconds of data
Phil_n = [0.5*rho*Af*v_n(1:ss).^2 m*g*cos(grade_n(1:ss))]; % Regressor
Y_n = F_n(1:ss) - m*ac_n(1:ss) - m*g*sin(grade_n(1:ss));
theta_n = (Phil_n'*Phil_n)\Phil_n'*Y_n;

%% RLS
Y = zeros(length(tspan),1); % pre-locate Y measured
for k = 1:length(tspan)
    Y(k) = F_n(k)-m*ac_n(k)-m*g*sin(grade_n(k)); % Y measured
end

regr = [(0.5*rho*Af*v_n.^2).'; % regressors for Cd
        (m*g*cos(grade_n)).']; % regressors for rolling resistance

theta = zeros(2,s); % pre-locate to-be-estimated parameters, Cd and Mu
L = zeros(2,1,s); % pre-locate updating gain
P = zeros(2,2,s); % pre-locate covariance matrix
for kk = ss:s-1
    P(:,:,ss) = [0.005 0;
                0 0.00005]; % initial covariance matrix
    theta(:,ss) = [theta_n(1,1); theta_n(2,1)]; % initial guesses for Cd and
Cr from Linear Least-Squares Estimation

    L(:,:,kk+1) = P(:,:,kk)*regr(:,kk+1)/(1 +
regr(:,kk+1).'*P(:,:,kk)*regr(:,kk+1));
    P(:,:,kk+1) = (eye(2) - L(:,:,kk+1)*regr(:,kk+1).')*P(:,:,kk);
    theta(:,kk+1) = theta(:,kk) + L(:,:,kk+1)*(Y(kk+1) -
regr(:,kk+1).'*theta(:,kk)); % RLS updating scheme
end

figure(1) % Plotting for Cd estimation process
plot(t,theta(1,:)) % plot Cd estimation vs. time
hold on
plot([0 t_tot],[Cd Cd],'r','linewidth',1.5) % correct/reference Cd value
hold on
plot([0 t_tot],[1.02*Cd 1.02*Cd],'--m')
hold on
plot([0 t_tot],[0.98*Cd 0.98*Cd],'--m')
ylim([0.3 1])
xlabel('Time (s)')
ylabel('Cd')
legend('Estimation','Correct Value')

figure(2) % Plotting for Cr estimation process
plot(t,theta(2,:)) % plot Cr estimation vs. time
hold on
plot([0 t_tot],[mu mu],'g','linewidth',1.5) % correct/reference Mu value
ylim([0 0.01])
xlabel('Time (s)')
ylabel('Mu')
```

```
legend('Estimation','Correct Value')

%% Function for Ode45
function vdot = model_x(t,v,m,Af,Cd,mu,rho,g,v_0) % function inside ODE45

F_e = engine(t,m,Af,Cd,mu,rho,g,v_0); % engine force function

phi = grade_profile(t); % road grade function

vdot = (1/m)*(F_e -0.5*rho*Cd*Af*v.^2 - mu*m*g*cos(phi)-m*g*sin(phi)); %
state space model that describes longitudinal motion of the vehicle

end

%% Propulsion Force and Road Grade Generation
function road_g = grade_profile(t) % road grade function based on time
if t <= 20
    road_g = deg2rad(0);
elseif 20 < t && t <= 280
    road_g = deg2rad(3*sin(2*pi()*0.02*(t-4)));
elseif 280 < t && t <= 330
    road_g = deg2rad(-0.1*(t-280));
elseif 330 < t && t <= 430
    road_g = deg2rad(-0.1*(t-330)+5);
elseif 430 < t && t <= 450
    road_g = deg2rad(1);
elseif 450 < t && t <= 470
    road_g = deg2rad(3);
elseif 470 < t && t <= 490
    road_g = deg2rad(-2);
elseif 490 < t && t <= 520
    road_g = deg2rad(0);
elseif 520 < t && t <= 540
    road_g = deg2rad(4);
elseif 540 < t && t <= 560
    road_g = deg2rad(-2);
elseif 560 < t && t <= 580
    road_g = deg2rad(3);
else
    road_g = deg2rad(0);
end
end

function F_engine = engine(t,m,Af,Cd,mu,rho,g,v_0) % propulsion force
function based on time

if t <= 10
    F_engine = (0.5*rho*Cd*Af*v_0^2)+(mu*m*g);
elseif 10 < t && t <= 30
    F_engine = 4500;
elseif 30 < t && t <= 35
    F_engine = 0;
elseif 35 < t && t <= 50
```

```
    F_engine = 3600;
elseif 50 < t && t <= 55
    F_engine = 0;
elseif 55 < t && t <= 70
    F_engine = 3900;
elseif 70 < t && t <= 80
    F_engine = 4200;
elseif 80 < t && t <= 85
    F_engine = 4500;
elseif 85 < t && t <= 90
    F_engine = 0;
elseif 90 < t && t <= 95
    F_engine = 2500;
elseif 95 < t && t <= 100
    F_engine = 3000;
elseif 100 < t && t <= 110
    F_engine = 3500;
elseif 110 < t && t <= 120
    F_engine = 4000;
elseif 120 < t && t <= 130
    F_engine = 4500;
elseif 130 < t && t <= 140
    F_engine = 5000;
elseif 140 < t && t <= 150
    F_engine = 0;
elseif 150 < t && t <= 170
    F_engine = 4500;
elseif 170 < t && t <= 175
    F_engine = 0;
elseif 175 < t && t <= 190
    F_engine = 3600;
elseif 190 < t && t <= 195
    F_engine = 0;
elseif 195 < t && t <= 210
    F_engine = 3900;
elseif 210 < t && t <= 220
    F_engine = 4200;
elseif 220 < t && t <= 225
    F_engine = 4500;
elseif 225 < t && t <= 230
    F_engine = 0;
elseif 230 < t && t <= 235
    F_engine = 2500;
elseif 235 < t && t <= 240
    F_engine = 3000;
elseif 240 < t && t <= 250
    F_engine = 3500;
elseif 250 < t && t <= 260
    F_engine = 4000;
elseif 260 < t && t <= 270
    F_engine = 4500;
elseif 270 < t && t <= 280
    F_engine = 5000;
elseif 280 < t && t <= 290
```

```
    F_engine = 0;
elseif 290 < t && t <= 300
    F_engine = 3500;
elseif 300 < t && t <= 310
    F_engine = 4000;
elseif 310 < t && t <= 320
    F_engine = 4500;
elseif 320 < t && t <= 330
    F_engine = 5000;
elseif 330 < t && t <= 340
    F_engine = 4000;
elseif 340 < t && t <= 350
    F_engine = 4500;
elseif 350 < t && t <= 360
    F_engine = 5000;
elseif 360 < t && t <= 370
    F_engine = 0;
elseif 370 < t && t <= 390
    F_engine = 4500;
elseif 390 < t && t <= 395
    F_engine = 0;
elseif 395 < t && t <= 410
    F_engine = 3600;
elseif 410 < t && t <= 425
    F_engine = 4500;
elseif 425 < t && t <= 445
    F_engine = 4500;
elseif 445 < t && t <= 450
    F_engine = 0;
elseif 450 < t && t <= 465
    F_engine = 3600;
elseif 465 < t && t <= 480
    F_engine = 0;
elseif 480 < t && t <= 500
    F_engine = 3900;
elseif 500 < t && t <= 510
    F_engine = 4200;
elseif 510 < t && t <= 515
    F_engine = 4500;
elseif 515 < t && t <= 520
    F_engine = 0;
elseif 520 < t && t <= 525
    F_engine = 2500;
elseif 525 < t && t <= 530
    F_engine = 3000;
elseif 530 < t && t <= 540
    F_engine = 3500;
elseif 540 < t && t <= 550
    F_engine = 4000;
elseif 550 < t && t <= 560
    F_engine = 3500;
elseif 560 < t && t <= 570
    F_engine = 4000;
elseif 570 < t && t <= 580
```

```
    F_engine = 4500;
elseif 580 < t && t <= 590
    F_engine = 5000;
else
    F_engine = (0.5*rho*Cd*Af*v_0^2)+(mu*m*g);
end
end
```

Appendix B

MATLAB Code for Generating Linear Least-Squares Estimates Point-Cloud

```
% Qifeng Liu
% Schreyer Honor Thesis
% Linear Least-Squares Estimates Point-Cloud

clc;
clear all;
close all;

%% Defined Known Variables
m = 8800; % vehicle mass [kg]
Af = 5; % frontal area [m^2]
rho = 1.275; % density of the air[kg/m^3]
g = 9.81; % gravity [m/s^2]

%% True (i.e., to-be-estimated) Air Drag Coefficient
Cd = 0.65; % air drag coefficient
mu = 0.006; % rolling resistance

%% ODE45 Formulation
v_0 = 40; % initial velocity
t_tot = 600; % simulation time [s]
tspan = 0:0.02:t_tot; % time range and step in ODE45
[t, v] = ode45(@(t,v) model_x(t,v,m,Af,Cd,mu,rho,g,v_0), tspan, v_0); % ODE45
that solve state-space equation

%% Propulsion Force and Road Grade Extracted from Ode Functions
F_er = zeros(length(tspan),1); % pre-locate engine force
grade_r = zeros(length(tspan),1); % pre-locate road grade measurement
for jj = 1:length(tspan)
    t_in = t(jj); % time index
    F_er(jj) = engine(t_in,m,Af,Cd,mu,rho,g,v_0); % engine force points
extraction
    grade_r(jj) = grade_profile(t_in); % road grade points extraction
end

%% Acceleration
acc = (1/m)*(F_er -0.5*rho*Cd*Af*v.^2 - mu*m*g*cos(grade_r)-
m*g*sin(grade_r)); % acceleration directly from ODE function

%% Linear Least-Squares Estimation without Noise Signals
Phil = [0.5*rho*Af*v.^2 m*g*cos(grade_r)];
Y = F_er - m*acc - m*g*sin(grade_r);
theta = (Phil'*Phil)\Phil'*Y;

%% Point-Cloud Generation, 1000 runs with noise in velocity, acceleration,
road grade, engine force
s = length(tspan); % size of the noise
```

```
theta_n = zeros(2,1000);

for ii = 1:1000
    F_n = F_er + 30*randn(s,1); % noisy engine force
    ac_n = acc + 0.01*randn(s,1); % noisy acceleration
    grade_n = grade_r + 0.001*randn(s,1); % noisy road grade on y measured
(output variable)
    grade_2n = grade_r + 0.001*randn(s,1); % noisy road grade on regressor
(input variable)
    v_n = v + 0.1*randn(s,1); % noisy velocity

    Phil_n = [0.5*rho*Af*v_n.^2 m*g*cos(grade_n)]; % Regressor Matrix
    Y_n = F_n - m*ac_n - m*g*sin(grade_n); % Measurement Output
    theta_n(:,ii) = (Phil_n'*Phil_n)\Phil_n'*Y_n; % Estimates
end

figure(1) % Point-Cloud Graph
c = linspace(1,10,length(theta_n(1,:)));
scatter(theta_n(2,:),theta_n(1,:),25,c)
hold on
sz = 40;
scatter(theta(2,1),theta(1,1),sz,'r','filled','DisplayName','Correct Values')
legend
xlabel('Cr')
ylabel('Cd')
xlim([0.0052 0.0068]);
ylim([0.635 0.665]);


%% Function for Ode45
function vdot = model_x(t,v,m,Af,Cd,mu,rho,g,v_0) % function inside ODE45

F_e = engine(t,m,Af,Cd,mu,rho,g,v_0); % engine force function

phi = grade_profile(t); % road grade function

vdot = (1/m)*(F_e -0.5*rho*Cd*Af*v.^2 - mu*m*g*cos(phi)-m*g*sin(phi)); %
state space model that describes longitudinal motion of the vehicle

end

%% Propulsion Force and Road Grade Generation
function road_g = grade_profile(t) % road grade function based on time
if t <= 20
    road_g = deg2rad(0);
elseif 20 < t && t <= 280
    road_g = deg2rad(3*sin(2*pi()*0.02*(t-4)));
elseif 280 < t && t <= 330
    road_g = deg2rad(-0.1*(t-280));
elseif 330 < t && t <= 430
    road_g = deg2rad(-0.1*(t-330)+5);
elseif 430 < t && t <= 450
```

```
    road_g = deg2rad(1);
elseif 450 < t && t <= 470
    road_g = deg2rad(3);
elseif 470 < t && t <= 490
    road_g = deg2rad(-2);
elseif 490 < t && t <= 520
    road_g = deg2rad(0);
elseif 520 < t && t <= 540
    road_g = deg2rad(4);
elseif 540 < t && t <= 560
    road_g = deg2rad(-2);
elseif 560 < t && t <= 580
    road_g = deg2rad(3);
else
    road_g = deg2rad(0);
end
end

function F_engine = engine(t,m,Af,Cd,mu,rho,g,v_0) % propulsion force
function based on time

if t <= 10
    F_engine = (0.5*rho*Cd*Af*v_0^2)+(mu*m*g);
elseif 10 < t && t <= 30
    F_engine = 4500;
elseif 30 < t && t <= 35
    F_engine = 0;
elseif 35 < t && t <= 50
    F_engine = 3600;
elseif 50 < t && t <= 55
    F_engine = 0;
elseif 55 < t && t <= 70
    F_engine = 3900;
elseif 70 < t && t <= 80
    F_engine = 4200;
elseif 80 < t && t <= 85
    F_engine = 4500;
elseif 85 < t && t <= 90
    F_engine = 0;
elseif 90 < t && t <= 95
    F_engine = 2500;
elseif 95 < t && t <= 100
    F_engine = 3000;
elseif 100 < t && t <= 110
    F_engine = 3500;
elseif 110 < t && t <= 120
    F_engine = 4000;
elseif 120 < t && t <= 130
    F_engine = 4500;
elseif 130 < t && t <= 140
    F_engine = 5000;
elseif 140 < t && t <= 150
    F_engine = 0;
elseif 150 < t && t <= 170
```

```
    F_engine = 4500;
elseif 170 < t && t <= 175
    F_engine = 0;
elseif 175 < t && t <= 190
    F_engine = 3600;
elseif 190 < t && t <= 195
    F_engine = 0;
elseif 195 < t && t <= 210
    F_engine = 3900;
elseif 210 < t && t <= 220
    F_engine = 4200;
elseif 220 < t && t <= 225
    F_engine = 4500;
elseif 225 < t && t <= 230
    F_engine = 0;
elseif 230 < t && t <= 235
    F_engine = 2500;
elseif 235 < t && t <= 240
    F_engine = 3000;
elseif 240 < t && t <= 250
    F_engine = 3500;
elseif 250 < t && t <= 260
    F_engine = 4000;
elseif 260 < t && t <= 270
    F_engine = 4500;
elseif 270 < t && t <= 280
    F_engine = 5000;
elseif 280 < t && t <= 290
    F_engine = 0;
elseif 290 < t && t <= 300
    F_engine = 3500;
elseif 300 < t && t <= 310
    F_engine = 4000;
elseif 310 < t && t <= 320
    F_engine = 4500;
elseif 320 < t && t <= 330
    F_engine = 5000;
elseif 330 < t && t <= 340
    F_engine = 4000;
elseif 340 < t && t <= 350
    F_engine = 4500;
elseif 350 < t && t <= 360
    F_engine = 5000;
elseif 360 < t && t <= 370
    F_engine = 0;
elseif 370 < t && t <= 390
    F_engine = 4500;
elseif 390 < t && t <= 395
    F_engine = 0;
elseif 395 < t && t <= 410
    F_engine = 3600;
elseif 410 < t && t <= 425
    F_engine = 4500;
elseif 425 < t && t <= 445
```

```
    F_engine = 4500;
elseif 445 < t && t <= 450
    F_engine = 0;
elseif 450 < t && t <= 465
    F_engine = 3600;
elseif 465 < t && t <= 480
    F_engine = 0;
elseif 480 < t && t <= 500
    F_engine = 3900;
elseif 500 < t && t <= 510
    F_engine = 4200;
elseif 510 < t && t <= 515
    F_engine = 4500;
elseif 515 < t && t <= 520
    F_engine = 0;
elseif 520 < t && t <= 525
    F_engine = 2500;
elseif 525 < t && t <= 530
    F_engine = 3000;
elseif 530 < t && t <= 540
    F_engine = 3500;
elseif 540 < t && t <= 550
    F_engine = 4000;
elseif 550 < t && t <= 560
    F_engine = 3500;
elseif 560 < t && t <= 570
    F_engine = 4000;
elseif 570 < t && t <= 580
    F_engine = 4500;
elseif 580 < t && t <= 590
    F_engine = 5000;
else
    F_engine = (0.5*rho*Cd*Af*v_0^2)+(mu*m*g);
end
end
```

Appendix C

MATLAB Code for Generating RLS Estimates Point-Cloud

```
% Qifeng Liu
% Schreyer Honor Thesis
% Recursive Least-Squares Estimates Point-Cloud

clc;
clear all;
close all;

%% Defined Known Variables
m = 8800; % vehicle mass [kg]
Af = 5; % frontal area [m^2]
rho = 1.275; % density of the air[kg/m^3]
g = 9.81; % gravity [m/s^2]

%% True (i.e., to-be-estimated) Air Drag Coefficient
Cd = 0.65; % air drag coefficient
mu = 0.006; % rolling resistance

%% ODE45 Formulation
v_0 = 40; % initial velocity
t_tot = 600; % simulation time [s]
time_step = 0.02;
tspan = 0:time_step:t_tot; % time range and step in ODE45
[t, v] = ode45(@(t,v) model_x(t,v,m,Af,Cd,mu,rho,g,v_0), tspan, v_0); % ODE45
that solve state-space equation at predetermined time step

%% Propulsion Force and Road Grade Extracted from Ode Functions
F_er = zeros(length(tspan),1); % pre-locate engine force
grade_r = zeros(length(tspan),1); % pre-locate road grade measurement
for jj = 1:length(tspan)
    t_in = t(jj); % time index
    F_er(jj) = engine(t_in,m,Af,Cd,mu,rho,g,v_0); % engine force points
extraction
    grade_r(jj) = grade_profile(t_in); % road grade points extraction
end

%% Acceleration
```

```
acc = (1/m)*(F_er -0.5*rho*Cd*Af*v.^2 - mu*m*g*cos(grade_r)-
m*g*sin(grade_r)); % acceleration directly from ODE function
%% 1,000 RLS Simulations
rls_pointclou = zeros(2,1000);

for gg = 1:1000; % for-loop for 1,000 repetitions

    tatime = 80*(1/time_step)+1; % the 80th time step (80 seconds mark
because of first 30 seconds and 50 seconds target)
    s = length(tspan); % number of steps
    F_n = F_er + 30*randn(s,1); % noisy engine force
    ac_n = acc + 0.01*randn(s,1); % noisy acceleration
    grade_n = grade_r + 0.001*randn(s,1); % noisy road grade
    v_n = v + 0.1*randn(s,1); % noisy velocity

    ss = 30*(1/time_step)+1; % first 30 seconds of data

    Phil_n = [0.5*rho*Af*v_n(1:ss).^2 m*g*cos(grade_n(1:ss))];
    Y_n = F_n(1:ss) - m*ac_n(1:ss) - m*g*sin(grade_n(1:ss));
    theta_n = (Phil_n'*Phil_n)\Phil_n'*Y_n;


    Y = zeros(length(tspan),1); % prelocate Y measured

    for k = 1:length(tspan)
        Y(k) = F_n(k)-m*ac_n(k)-m*g*sin(grade_n(k)); % Y measured
    end

    regr = [(0.5*rho*Af*v_n.^2).'; % regressors for Cd
            (m*g*cos(grade_n)).']; % regressors for rolling resistance

    theta = zeros(2,s); % pre-locate to-be-estimated parameters, Cd and Mu
    L = zeros(2,1,s); % preo-locate updating gain
    P = zeros(2,2,s); % prel-ocate covariance matrix
    for kk = ss:s-1
        P(:,:,ss) = [0.005 0;
                     0 0.00005]; % initial covariance matrix
        theta(:,ss) = [theta_n(1,1); theta_n(2,1)]; % initial guesses for Cd
and Mu

        L(:,:,kk+1) = P(:,:,kk)*regr(:,kk+1)/(1 +
regr(:,kk+1).'*P(:,:,kk)*regr(:,kk+1));
        P(:,:,kk+1) = (eye(2) - L(:,:,kk+1)*regr(:,kk+1).')*P(:,:,kk);
        theta(:,kk+1) = theta(:,kk) + L(:,:,kk+1)*(Y(kk+1) -
regr(:,kk+1).'*theta(:,kk)); % RLS updating scheme
    end

    rls_pointclou(:,gg) = theta(:,tatime);
end

figure(1) % Point-Cloud Graph
c = linspace(1,10,length(rls_pointclou(1,:)));
```

```
scatter(rls_pointclou(2,:),rls_pointclou(1,:),25,c)
hold on
sz = 40;
scatter(mu,Cd,sz,'r','filled','DisplayName','Correct Values')
legend
xlabel('Cr')
ylabel('Cd')
xlim([0.0052 0.0068]);
ylim([0.635 0.665]);




%% Function for Ode45
function vdot = model_x(t,v,m,Af,Cd,mu,rho,g,v_0) % function inside ODE45

F_e = engine(t,m,Af,Cd,mu,rho,g,v_0); % engine force function

phi = grade_profile(t); % road grade function

vdot = (1/m)*(F_e -0.5*rho*Cd*Af*v.^2 - mu*m*g*cos(phi)-m*g*sin(phi)); %
state space model that describes longitudinal motion of the vehicle

end

%% engine force and road grade function

function road_g = grade_profile(t) % road grade function based on time
if t <= 20
    road_g = deg2rad(0);
elseif 20 < t && t <= 280
    road_g = deg2rad(3*sin(2*pi()*0.02*(t-4)));
elseif 280 < t && t <= 330
    road_g = deg2rad(-0.1*(t-280));
elseif 330 < t && t <= 430
    road_g = deg2rad(-0.1*(t-330)+5);
elseif 430 < t && t <= 450
    road_g = deg2rad(1);
elseif 450 < t && t <= 470
    road_g = deg2rad(3);
elseif 470 < t && t <= 490
    road_g = deg2rad(-2);
elseif 490 < t && t <= 520
    road_g = deg2rad(0);
elseif 520 < t && t <= 540
    road_g = deg2rad(4);
elseif 540 < t && t <= 560
    road_g = deg2rad(-2);
elseif 560 < t && t <= 580
    road_g = deg2rad(3);
else
    road_g = deg2rad(0);
```

```
end
end

function F_engine = engine(t,m,Af,Cd,mu,rho,g,v_0) % propulsion force
function based on time

if t <= 10
    F_engine = (0.5*rho*Cd*Af*v_0^2)+(mu*m*g);
elseif 10 < t && t <= 30
    F_engine = 4500;
elseif 30 < t && t <= 35
    F_engine = 0;
elseif 35 < t && t <= 50
    F_engine = 3600;
elseif 50 < t && t <= 55
    F_engine = 0;
elseif 55 < t && t <= 70
    F_engine = 3900;
elseif 70 < t && t <= 80
    F_engine = 4200;
elseif 80 < t && t <= 85
    F_engine = 4500;
elseif 85 < t && t <= 90
    F_engine = 0;
elseif 90 < t && t <= 95
    F_engine = 2500;
elseif 95 < t && t <= 100
    F_engine = 3000;
elseif 100 < t && t <= 110
    F_engine = 3500;
elseif 110 < t && t <= 120
    F_engine = 4000;
elseif 120 < t && t <= 130
    F_engine = 4500;
elseif 130 < t && t <= 140
    F_engine = 5000;
elseif 140 < t && t <= 150
    F_engine = 0;
elseif 150 < t && t <= 170
    F_engine = 4500;
elseif 170 < t && t <= 175
    F_engine = 0;
elseif 175 < t && t <= 190
    F_engine = 3600;
elseif 190 < t && t <= 195
    F_engine = 0;
elseif 195 < t && t <= 210
    F_engine = 3900;
elseif 210 < t && t <= 220
    F_engine = 4200;
elseif 220 < t && t <= 225
    F_engine = 4500;
elseif 225 < t && t <= 230
    F_engine = 0;
```

```
elseif 230 < t && t <= 235
    F_engine = 2500;
elseif 235 < t && t <= 240
    F_engine = 3000;
elseif 240 < t && t <= 250
    F_engine = 3500;
elseif 250 < t && t <= 260
    F_engine = 4000;
elseif 260 < t && t <= 270
    F_engine = 4500;
elseif 270 < t && t <= 280
    F_engine = 5000;
elseif 280 < t && t <= 290
    F_engine = 0;
elseif 290 < t && t <= 300
    F_engine = 3500;
elseif 300 < t && t <= 310
    F_engine = 4000;
elseif 310 < t && t <= 320
    F_engine = 4500;
elseif 320 < t && t <= 330
    F_engine = 5000;
elseif 330 < t && t <= 340
    F_engine = 4000;
elseif 340 < t && t <= 350
    F_engine = 4500;
elseif 350 < t && t <= 360
    F_engine = 5000;
elseif 360 < t && t <= 370
    F_engine = 0;
elseif 370 < t && t <= 390
    F_engine = 4500;
elseif 390 < t && t <= 395
    F_engine = 0;
elseif 395 < t && t <= 410
    F_engine = 3600;
elseif 410 < t && t <= 425
    F_engine = 4500;
elseif 425 < t && t <= 445
    F_engine = 4500;
elseif 445 < t && t <= 450
    F_engine = 0;
elseif 450 < t && t <= 465
    F_engine = 3600;
elseif 465 < t && t <= 480
    F_engine = 0;
elseif 480 < t && t <= 500
    F_engine = 3900;
elseif 500 < t && t <= 510
    F_engine = 4200;
elseif 510 < t && t <= 515
    F_engine = 4500;
elseif 515 < t && t <= 520
    F_engine = 0;
```

```
elseif 520 < t && t <= 525
    F_engine = 2500;
elseif 525 < t && t <= 530
    F_engine = 3000;
elseif 530 < t && t <= 540
    F_engine = 3500;
elseif 540 < t && t <= 550
    F_engine = 4000;
elseif 550 < t && t <= 560
    F_engine = 3500;
elseif 560 < t && t <= 570
    F_engine = 4000;
elseif 570 < t && t <= 580
    F_engine = 4500;
elseif 580 < t && t <= 590
    F_engine = 5000;
else
    F_engine = (0.5*rho*Cd*Af*v_0^2)+(mu*m*g);
end
end
```

BIBLIOGRAPHY

[1] European Commission, *EU Transport in Figures—Statistical Pocketbook*. Luxembourg: Publications Office of the European Union, 2015.

[2] Scania AB,.(2014), Annual report. [Online]. Available: http://scania.com/investor-relations/financial-reports/2014/scania-annual-report-2014.aspx

[3] Alam, A., Besselink, B., Turri, V., Martensson, J., & Johansson, K. H. (2015). Heavy-duty vehicle platooning for sustainable freight transportation: A cooperative method to enhance safety and efficiency. *IEEE Control Systems Magazine*, 35(6), 34-56.

[4] Alvarez, M., Xu, C., Rodriguez, M. A., Al-Mamun, A., Wahba, M., Brennan, S., & Fathy, H. K. (2018). Reducing Road Vehicle Fuel Consumption by Exploiting Connectivity and Automation: A Literature Survey. Beijing: *AVEC*. Retrieved March 25, 2019.

[5] Chowdhury, H., Moria, H., Ali, A., Khan, I., Alam, F., & Watkins, S. (2013). A study on aerodynamic drag of a semi-trailer truck. *Procedia Engineering*, 56, 201-205.

[6] Bae, H. S., Ryu, J., & Gerdes, J. C. (2001, August). Road grade and vehicle parameter estimation for longitudinal control using GPS. In *Proceedings of the IEEE Conference on Intelligent Transportation Systems* (pp. 25-29).

[7] Davila, A., & Ferrer, A. (2014). Tackling Three Critical Issues of Transportation: Environment, Safety and Congestion Via Semi-autonomous Platooning (No. 2014-01-2007). *SAE Technical Paper*.

[8] Fathy, H. K., Kang, D., & Stein, J. L. (2008, June). Online vehicle mass estimation using recursive least squares and supervisory data extraction. In *2008 American control conference* (pp. 1842-1848). IEEE.

[9] Åström, K. J., & Wittenmark, B. (1995). *Adaptative control*. Reading (Mass.): Addison Wesley.

[10] Norton, J. P. (2009). *An introduction to identification*. Mineola, NY: Dover Publications.

[11] Iliff, K. W. (1989). Parameter estimation for flight vehicles. *Journal of Guidance, Control, and Dynamics*, 12(5), 609-622.

[12] McLaughlin, C. A., Mance, S., & Lichtenberg, T. (2011). Drag coefficient estimation in orbit determination. *The Journal of the Astronautical Sciences*, 58(3), 513-530.

[13] Barati, R., Neyshabouri, S. A. A. S., & Ahmadi, G. (2014). Development of empirical models with high accuracy for estimation of drag coefficient of flow around a smooth sphere: An evolutionary approach. *Powder Technology*, 257, 11-19.

[14] Ahlawat, R., Bredenbeck, J., & Ichige, T. (2013). Estimation of road load parameters via on-road vehicle testing. *Tire Technology Expo*.

[15] Vahidi, A., Stefanopoulou, A., & Peng, H. (2005). Recursive least squares with forgetting for online estimation of vehicle mass and road grade: theory and experiments. *Vehicle System Dynamics*, 43(1), 31-55.

[16] Zhang, D., Ivanco, A., & Filipi, Z. (2015). Model-Based Estimation of Vehicle Aerodynamic Drag and Rolling Resistance. *SAE International Journal of Commercial Vehicles*, 8(2015-01-2776), 433-439.

[17] Kandel, A. I., Wahba, M., Geyer, S., & Fathy, H. K. (2018). Impact of Terrain Variability on Chassis Parameter Identifiability for a Heavy-Duty Vehicle. *2018 European Control Conference (ECC)*. doi:10.23919/ecc.2018.8550040

ACADEMIC VITA OF Qifeng Liu

qjl5003@psu.edu

EDUCATION **The Pennsylvania State University, University Park, PA** May 2019

B.S., Mechanical Engineering, May 2019

Schreyer Honors College (Undergraduate Research Thesis Component)

Minors in Engineering Leadership Development and Economics

Teaching Assistant, ENGR 118 Impact of Culture on Engineering in China (Summer 2016)

Lean-Sigma Process Improvement Yellow Belt | GD&T Micro-credential

Karlsruhe Institute of Technology, Karlsruhe, Germany (Exchange-semester) 03/2017 – 08/2017

Relevant Coursework: Vehicle Ride Comfort and Acoustics | Safe Mechatronics | Machine dynamics

EXPERIENCE **NEXTCAR Research Project: Aerodynamic Drag Coefficient Estimation, Penn State** 01/18 – Present

- Accurately estimate a heavy-duty truck's aerodynamic drag coefficient in less than 50 seconds by constructing vehicle longitudinal dynamics model and using recursive least squares estimation
- Use linear regression to conduct parameter identification through 1,000 simulations having independent, random, and white noise in order to quantify a 2% estimation accuracy
- Create parameter estimation for NEXTCAR platoon control to increase heavy-duty vehicle's fuel efficiency

Collaborative Hazard Detection for Connected Vehicles - Research Project, **Penn State** 07/18 – Present

- Plan to construct a novel descriptor using optical flow at early vision level to address hazard and collision warning under the presence of sensor occlusions due to close vehicle distance during platoon
- Generate warning messages to other vehicles at the occurrence of hazard situations

General Motors Subsystem Safety Lab, Test, and Validation Intern, Shanghai, China 05/15 – 08/15

- Performed safety tests for occupant protection, interior impact & upper interior head impact, safety belt assembly anchorage, anchorage of seats, and lower anchors and tethers for children
- Configured hydraulic impact machine and laser scanner, vehicles and test-dummies locations/impact points
- Collaborated with cross-functional teams and met safety validation goals for the Cadillac CT6 and Buick GL8

PROJECTS **Team Lead, ArcelorMittal Design Project** 01/15 – 05/15

- Won the most sustainable design in the ArcelorMittal-sponsored project competition
- Created and optimized a plan for ArcelorMittal Steelton to recycle and reuse transportation pallets, fluid delivery drums/totes, and refractory bricks

Team Lead, ENGR407 Entrepreneurship Group Projects 08/16 – 12/16

- Designed and sold Penn State rally towels at football games, generating $5,000 profit (ranked 2nd in class)
- Modeled a start-up company, *Simppliances*— offered a portable cup and silverware washer as first products, as the final presentation showcase reached 2nd place out of 15 teams
- Crafted marketing strategies and simulated realistic company financial situations to assess the prospects

SKILLS Computer-based: AutoCAD, CATIA V5, DaqView, MATLAB, OpenModelica, SolidWorks, SignalCalc ACE

Languages: English (Fluent Proficiency), Chinese (Native Proficiency), Deutsch (Elementary Proficiency)