# Modelling and Model Predictive Control of Power-Split Hybrid Powertrains for Self-Driving Vehicles

by

Bryce Hosking

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2018

# AUTHOR'S DECLARATION

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

# Statement of Contributions

All modelling and controller development work contained within this thesis is my own. However, this thesis contains some material from the following multi-author paper:

- B.A. Hosking and J. McPhee, "Powertrain Modeling and Model Predictive Longitudinal Dynamics Control for Hybrid Electric Vehicles," in *SAE World Congress*, Detroit, 2018.

In addition, some simulation results and controller development work utilizes the vehicle dynamics model described in the following thesis document:

- M. Van Gennip, "Parameter Identification and Vehicle Dynamic Modelling of an Autonomous Vehicle," University of Waterloo, Waterloo, ON, 2018.

# Abstract

Designing an autonomous vehicle system architecture requires extensive vehicle simulation prior to its implementation on a vehicle. Simulation provides a controlled environment to test the robustness of an autonomous architecture in a variety of driving scenarios. In any autonomous vehicle project, high-fidelity modelling of the vehicle platform is important for accurate simulations. For power-split hybrid electric vehicles, modelling the powertrain for autonomous applications is particularly difficult. The mapping from accelerator and brake pedal positions to torque at the wheels can be a function of many states. Due to this complex powertrain behavior, it is challenging to develop vehicle dynamics control algorithms for autonomous power-split hybrid vehicles.

The 2015 Lincoln MKZ Hybrid is the selected vehicle platform of Autonomoose, the University of Waterloo's autonomous vehicle project. Autonomoose required high-fidelity models of the vehicle's power-split powertrain and braking systems, and a new longitudinal dynamics vehicle controller. In this thesis, a grey-box approach to modelling the Lincoln MKZ's powertrain and braking systems is proposed. The modelling approach utilizes a combination of shallow neural networks and analytical methods to generate a mapping from accelerator and brake pedal positions to the torque at each wheel. Extensive road testing of the vehicle was performed to identify parameters of the powertrain and braking models. Experimental data was measured using a vehicle measurement system and CAN bus diagnostic signals. Model parameters were identified using optimization algorithms. The powertrain and braking models were combined with a vehicle dynamics model to form a complete high-fidelity model of the vehicle that was validated by open-loop simulation.

The high-fidelity models of the powertrain and braking were simplified and combined with a longitudinal vehicle dynamics model to create a control-oriented model of the vehicle. The control-oriented model was used to design an instantaneously linearizing model predictive controller (MPC). The advantages of the MPC over a classical proportional-integral (PI) controller were proven in simulation, and a framework for implementing the MPC on the vehicle was developed. The MPC was implemented on the vehicle for track testing. Early track testing results of the MPC show superior performance to the existing PI that could improve with additional controller parameter tuning.

# Acknowledgements

I would like to thank…

my supervisor, Professor John McPhee for his guidance and support throughout the course my MASc degree. He provided me with an excellent opportunity to explore an exciting and innovative field of engineering, which has been incredible for my professional and personal development.

my thesis readers: Professor Steven Waslander and Professor Baris Fidan, for agreeing to review my thesis and provide valuable feedback on my work.

all members of the Motion Research Group (MoRG), but particularly Mohit Batra, Matthew Van Gennip, and Amer Keblawi for their help in my research pursuits.

Maplesoft, the Natural Sciences and Engineering Research Council of Canada, and the Ontario Government for providing funding for this project.

all the friends I made here at UW that have helped me feel at home in Waterloo.

my parents, Paula and Derek, and sister, Amy, for all their support and encouragement along the way.

And Brea, who has been supportive of me since day one of my MASc degree.

# Dedication

To my Family

# Table of Contents

# List of Figures

## List of Tables

# Chapter 1
# Introduction

## 1.1 Overview of Self-Driving Vehicle Architecture

For the purposes of this thesis a self-driving or autonomous car is defined as any car that meets the SAE standard of level 3 automation, conditional automation, or higher. This is defined as any vehicle where an autonomous system monitors the driving environment, and in at least some driving modes the system is capable of controlling all aspects of dynamic driving, subject to fallback on a human driver for certain interventions [1]. Development of self-driving cars typically involves the design of a complex system architecture composed of five fundamental subsystems: perception, localization, behavior and path planning, vehicle control, and system management [2] [3].

The perception subsystem uses available sensors, such as cameras or Light Detection and Ranging (LIDAR) sensors, to understand and map the vehicle's surrounding environment. This process includes object tracking, road mapping, and interpretation of traffic signage. The localization subsystem uses Global Positioning System (GPS) and Inertial Measurement Unit (IMU) sensors, in addition to the perception sensors, to estimate the pose of the car within its environment. A common problem in autonomous vehicle development is the simultaneous localization and mapping of a vehicle within its environment [4]. To address this problem, perception and localization are frequently combined into a process that fuses data from both sets of sensors.

The planning subsystem uses the vehicle's estimated states and information on the car's environment to determine a desired path of travel for the vehicle. Planning includes both high-level route planning from the road map and local behavior planning at the vehicle level. The behavior planner outputs a local path plan over a known time horizon to the vehicle controller. The vehicle control subsystem attempts to track the local path plan by actuating vehicle control inputs, which include the acceleration command, braking command, and a steering command. System management is a supervisory subsystem that has a variety of functions including subsystem fault detection, sensor monitoring, data logging, and human machine interface [3].

## 1.2 Autonomoose Project

Autonomoose is a project within the University of Waterloo (UW) that is focused on the development of a fully autonomous car for driving on Canadian roads. The vehicle platform selected for the project is a 2015 Lincoln MKZ Hybrid (referred to as the “Moose”) that has been outfitted for drive-by-wire control by the company AutonomousStuff [5] in collaboration with Dataspeed Inc [6]. All subsequent work on hardware selection, software development, integration, and testing has been performed by a multidisciplinary team of UW faculty and students. In September of 2016 the Moose became the first vehicle platform to be approved for autonomous driving on Canadian roads. Figure 1 depicts a photo of the Moose during road testing.

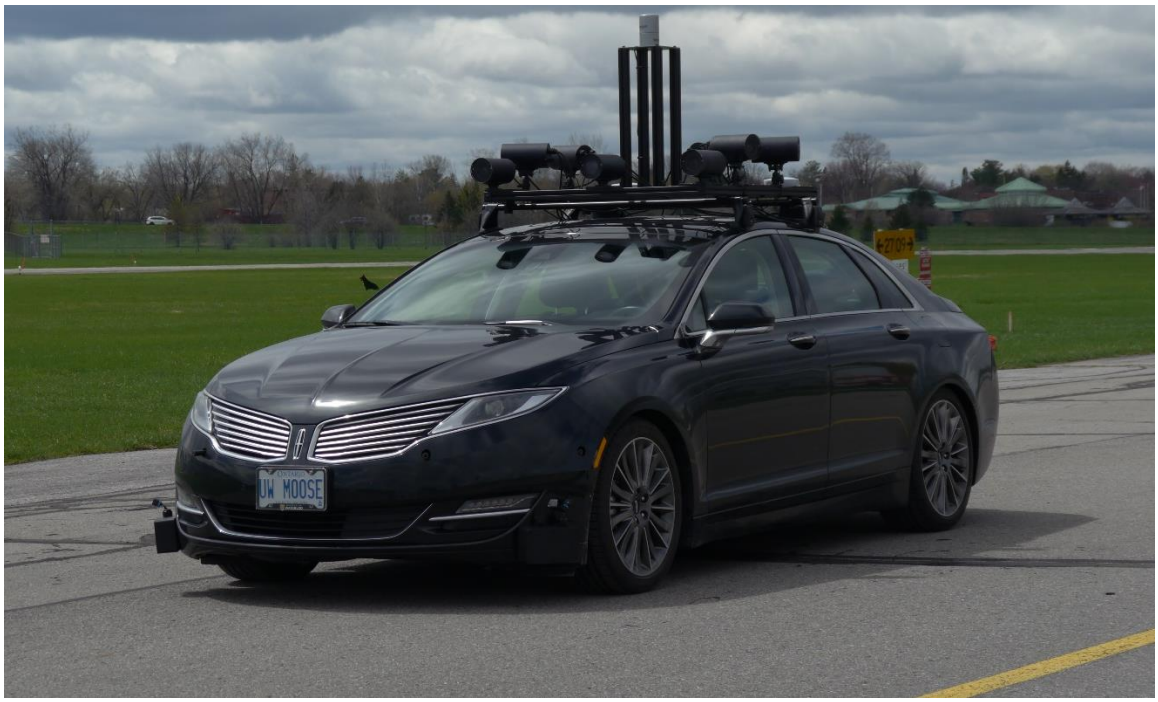


**Figure 1.1: The Moose, UW's autonomous vehicle platform, depicted during road testing.**

The primary research interest of Autonomoose is solving self-driving vehicle development problems that are unique to the conditions of Canada. Unlike more common locales for autonomous vehicle road testing, such as Southern California, Canadian roadways are susceptible to snow, freezing rain, fog,

and other adverse driving conditions. The project also has research interests in autonomous vehicle fuel economy, robust system architecture design, and other topics.

The vehicle modelling and controls team focuses on development of a high-fidelity model of the Moose and path-tracking controller development. The fully integrated vehicle model must include experimentally verified dynamic models of the suspension, steering, tires, and powertrain. The model will be used by the simulation team to test autonomous system architecture on a variety of autonomous driving scenarios prior to real vehicle implementation. The high-fidelity vehicle model will also be used to develop lower fidelity control-oriented models for vehicle path-tracking controller design.

## 1.3 Objectives

The content of this thesis focuses on modelling and controls applications for the Lincoln MKZ's hybrid electric powertrain. The Autonomoose project requires an accurate mapping from the accelerator pedal position ($APP$) and brake pedal position ($BPP$) to the torque applied at the vehicle's wheels. $APP$ and $BPP$ are two of the control inputs, in addition to steering wheel angle, that are used to control the Moose's vehicle dynamics. Due to the complexity of the vehicle's power-split powertrain system, the $APP$ mapping must include modelling of the powertrain control system as well as the drivetrain dynamics. The Autonomoose Project has no association with the Ford Motor Company©, so a priori knowledge of the system is limited. A major contribution of this thesis is the development of a suitable method of modelling the complete powertrain system. Identification of model parameters requires extensive experimental vehicle testing and data acquisition.

Based on performance limitations of the current vehicle dynamics control system for the Lincoln MKZ, the Autonomoose project requires the development of a better longitudinal velocity tracking controller. The second main objective of this thesis work is to utilize the identified APP and BPP mapping models and other vehicle dynamics parameters to develop a longitudinal dynamics controller for the Moose. The controller must initially be tested in simulation on a high-fidelity model of the vehicle before being implemented on the actual vehicle. Completion of this objective solves an important controls problem for the Autonomoose project, but in addition it will create a procedural framework for development and implementation of future model-based vehicle dynamics controllers on the Moose.

## 1.4 Thesis Organization

Chapter 1 of this thesis begins with a brief description of self-driving vehicle architecture. UW's autonomous vehicle project, Autonomoose, is introduced, and the goals of the project are discussed. The objectives of the work contained within this thesis are then explained.

Chapter 2 discusses some required background information for the reader's consideration. Shallow feedforward neural networks are introduced. Two types of neural network that are relevant to this work are explained in detail. The concept of model predictive control is introduced, and the fundamentals of linear model predictive control are explained. Two literature reviews are also discussed. The first is a review of current methods for modelling power-split hybrid powertrains such as the one used in the Lincoln MKZ. The second review is of applications and methods for longitudinal dynamics control of vehicles explored in literature.

Chapter 3 presents the proposed method of modelling the powertrain and brakes of the Lincoln MKZ. A grey-box modelling approach that separates the power-split powertrain into three subsystems is introduced. The modelling methods used in each powertrain subsystem are explained in detail. The model of the front and rear brakes is introduced, and a method of integrating it with the powertrain model is proposed. The interface of the brake and powertrain models with a vehicle dynamics model to form a complete high-fidelity vehicle model is briefly explained.

Chapter 4 discusses the experimentation process used to gather data for model parameter identification. Details of two vehicle testing apparatus, the A&D Technology Vehicle Measurement System and the Vector Canalyzer Tool, are provided. An outline of vehicle tests used for parameter identification was provided.

Chapter 5 discusses the methods and results of parameter identification for the models presented in chapter 3. Neural network training algorithms were used to identify the weighting and offset parameters of neural network subsystem models. The physical parameters of the drivetrain were identified by multiple methods that are described in detail.

Chapter 6 presents the design of an instantaneously linearizing model predictive controller for longitudinal velocity tracking. The powertrain and braking models were adapted to a control-oriented model of the longitudinal vehicle dynamics. The velocity tracking controller was tested in multiple driving simulation scenarios against a benchmark PID controller. Simulation results and the advantages of the model predictive controller over the classical PID controller are discussed.

Chapter 7 outlines the process of how the model predictive controller was implemented on the Moose by integrating it with the existing autonomous stack. The process required the controller to be converted to a Robot Operating System (ROS) node. The procedure for implementing and testing the control is explained, and test results are discussed in detail.

Chapter 8 presents the conclusions of this thesis, and its major contributions to the Autonomoose project are summarized. Recommendations for future work are also discussed. These include possible improvements to the model's level of fidelity and some proposed methods for improving or extending the vehicle dynamics controller.

# Chapter 2
# Background and Literature Review

## 2.1 Feedforward Neural Networks

Artificial neural networks (NNs) were originally conceived as a means of modelling the input-output functionality of complex functions and systems. The structure of NNs was inspired by the behavior of the human brain. Much of the brain is composed of a system of neurons that are connected in a web-like structure by synapses. Decision-making within the brain is performed by the transference of electrical signals along these synapses. When a neuron receives a signal from a connected synapse, it will perform a decision-making process before transmitting an electrical signal along different synapses to other neurons.

Shallow feedforward NNs were some of the first artificial NNs to be devised. A fixed number of input signals are used by the network to calculate a fixed number of output signals. The synapses are modelled as fixed gain terms applied to individual signals, and the neurons are modelled with known nonlinear functions, which are called activation functions. As opposed to the complex web of neurons in the brain, shallow feedforward NNs typically include only one hidden layer with a preselected number of neurons and a unidirectional flow of information. One of the most common forms of NN is the double layer perceptron. An example layout of a three input, four hidden neuron, two output double layer perceptron is depicted in Figure 2.1, but the procedure may be generalized to a network of $n_x$ inputs, $n_n$ hidden neurons, and $n_y$ outputs.

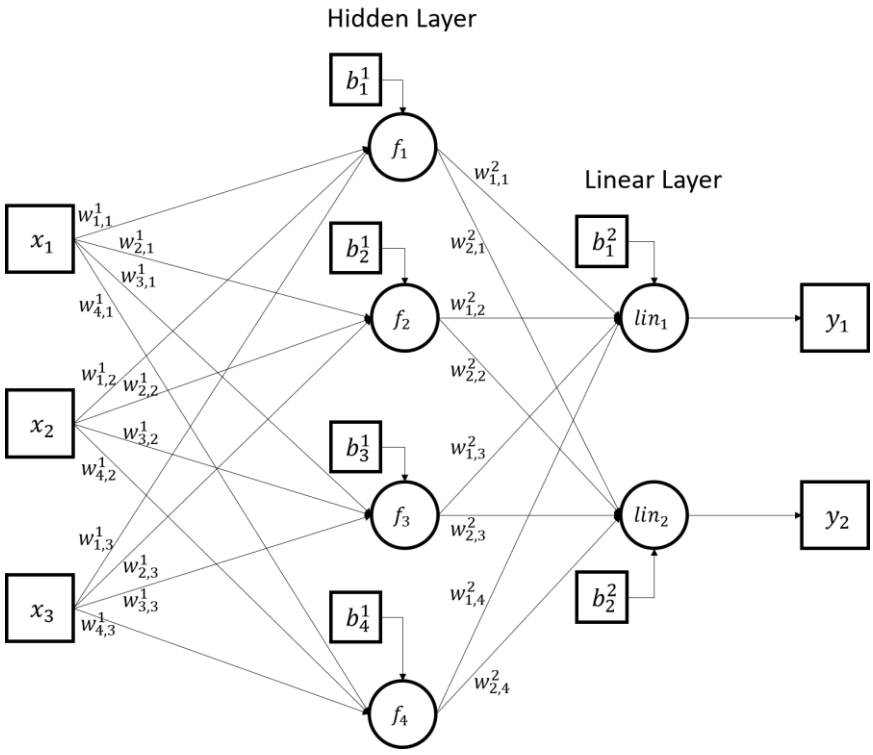


**Figure 2.1: A generic double layer perceptron NN with three inputs, four hidden neurons, and two outputs.**

As depicted in Figure 2.1, at the input to each hidden neuron, $j$, the values of each input signal, $x_i$, are each scaled by an input gain term $w_{j,i}^1$. The scaled input terms and a constant offset term $b_j^1$ are added together, and a nonlinear function $f_j$ is applied to calculate the neuron's output. One of the most common choices for the nonlinear function is a sigmoid. The hidden neuron passes its output to the linear layer of the NN. At each linear neuron, $k$, the output of each hidden layer neuron, $j$, is scaled by an output gain term $w_{k,j}^2$. The scaled hidden neuron outputs and a constant term $b_k^2$ are added together to calculate each NN output $y_k$. The generalized equation of a double layer perceptron NN is represented by Equation 1.1:

$$Y = W2\, Sig(W1\, X + B1) + B2 \tag{1.1}$$

where $X$ is a column vector of $n_x$ inputs, $Y$ is a column vector $n_y$ outputs, $W1$ is a matrix of $n_n \times n_x$ parameter terms, $W2$ is a matrix of $n_y \times n_n$ parameter terms, $B1$ is a column vector of $n_n$ parameter terms, and $B2$ is a column vector of $n_y$ parameter terms. The function $Sig$ is defined by Equation 1.2:

$$Sig(U) = \begin{bmatrix} \frac{1}{1+e^{u_1}} \\ \frac{1}{1+e^{u_2}} \\ \vdots \\ \frac{1}{1+e^{u_n}} \end{bmatrix} \tag{1.2}$$

Early research in the field of NNs mathematically proved that double layer feedforward NN, such as the double layer perceptron, are universal approximators of nonlinear functions [7].

A second type of feedforward NN that is pertinent to the content of this thesis is a double layer classifier NN. These types of NNs attempt to classify the sets of inputs into target categories. For example parameters such as weight, number of seats, and engine size could be used to guess the class of car (sedan, coup, etc.). An example layout of a three input, four hidden neuron, two output double layer classifier is depicted in Figure 2.2, but the procedure may be generalized to a system of $n_x$ inputs, $n_n$ hidden neurons and $n_y$ outputs.

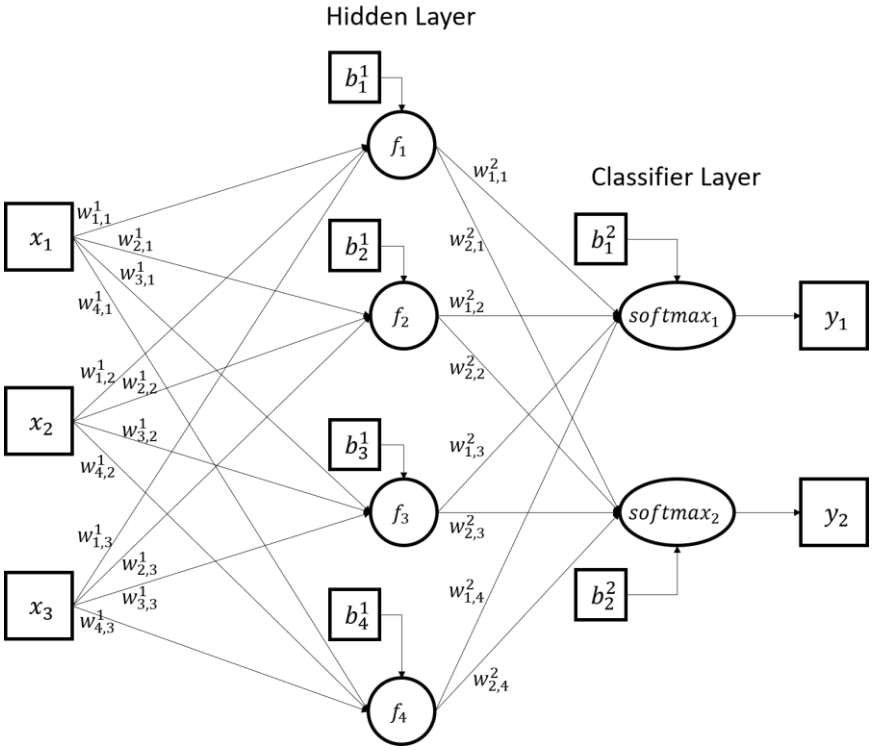


**Figure 2.2: A generic softmax output double layer classifier NN with three inputs, four hidden neurons, and two outputs.**

As depicted in Figure 2.2, The classifier NN has a similar form to the double layer perceptron. The difference is that the linear layer is replaced with a classifier layer. Each neuron of the classifier layer uses a softmax function that calculates the probability of a corresponding classification being true. The output with the highest value corresponds to the classification that the network identifies as most likely to be the true classification. The number of outputs $n_y$ must be equal to the number of possible classifications. The generalized equation of a softmax classification NN is represented by Equation 1.3:

$$Y = softmax(W2\, Sig(W1\, X + B1) + B2) \tag{1.3}$$

where $X$ is a column vector of $n_x$ inputs, $Y$ is a column vector $n_y$ outputs, $W1$ is a matrix of $n_n \times n_x$ parameter terms, $W2$ is a matrix of $n_y \times n_n$ parameter terms, $B1$ is a column vector of $n_n$ parameter

terms, and $B2$ is a column vector of $n_y$ parameter terms. the function $softmax$ is defined by Equation 1.4:

$$softmax(U) = \begin{bmatrix} \frac{e^{u_1}}{\sum_{i=1}^{n} e^{u_n}} \\ \frac{e^{u_2}}{\sum_{i=1}^{n} e^{u_n}} \\ \vdots \\ \frac{e^{u_n}}{\sum_{i=1}^{n} e^{u_n}} \end{bmatrix} \tag{1.4}$$

It is clear from Equation 1.4 that the sum of the output terms of a softmax function is equal to 1, which corresponds to the full range of classification probabilities.

The weighting parameters, $W1$ and $W2$, and the offset parameters, $B1$ and $B2$, of shallow feedforward NNs must be identified from a training data set. Experimental measurements of inputs and their corresponding outputs must be available, and the accuracy of the training is dependent on the size of the data set. $n_y$ must be suitably selected to model the behavior of the subsystem without overfitting the experimental data. Typically, network training is performed using a backpropagation-based algorithm. Backpropagation determines the gradient of the tunable parameters with respect to a predefined error function, and it is combined with a gradient-based optimizer, such as damped least-squares, to determine a local minima of the error function [8].

## 2.2 Model Predictive Control

Model predictive control (MPC) is an advanced controls method that is typically used for complex dynamical systems. MPC uses a discretized control-oriented model of plant dynamics that predicts how changes to the control input will affect the outputs of the system. The controller calculates the control action by solving an online optimization problem at each time step [9]. Figure 2.3 depicts a visualization of an MPC scheme for a single-input single-output (SISO) plant.

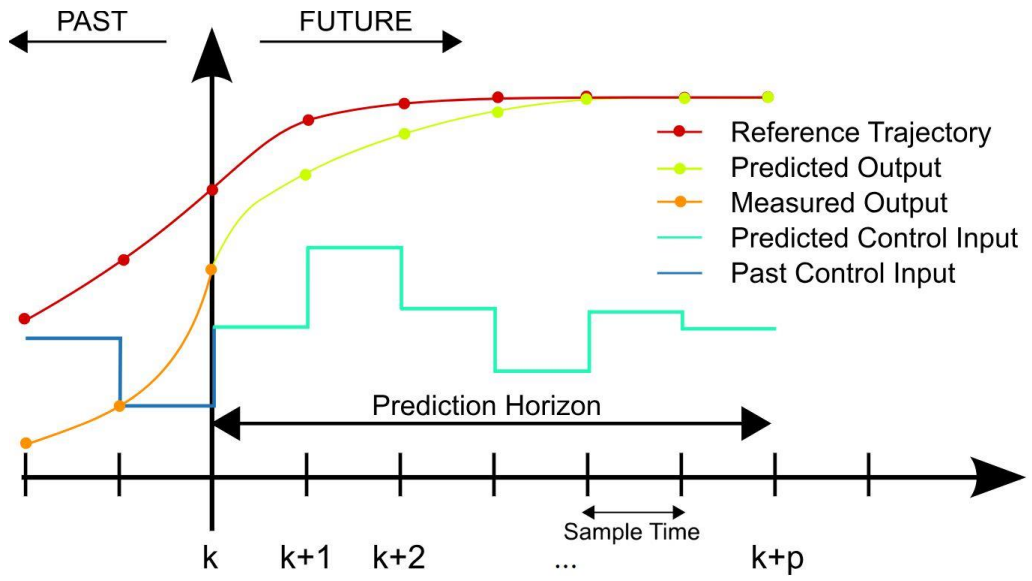


**Figure 2.3: MPC control scheme for a SISO system [10].**

As depicted in Figure 2.3, at each time step, $k$, a desired reference trajectory for the plant output is provided over a finite time horizon of $p$ time steps. The optimization algorithm determines an optimal sequence of control inputs, $u$, for the predicted plant output to track the reference trajectory. At each time step, $k$, only the first control input is applied. The optimization routine is repeated at each time step.

The primary means of tuning an MPC controller is in the selection of function to be optimized, referred to as the objective function. The objective function may include terms for tracking error of the plant output, magnitude of control action applied, rate of change of control action, and other terms related to plant states. Equation 1.5 defines a generic objective function for a multi-input multi-output system:

$$J = \sum_{i=k+1}^{k+p} F\big(e_1(i), e_2(i), \dots e_n(i)\big) + \sum_{j=k}^{k+p-1} G\big(u_1(j), \Delta u_1(j), u_2(j), \Delta u_2(j), \dots u_m(j), \Delta u_m(j)\big) \quad (1.5)$$

where $e$ is the error between reference state, $x_{ref}$, and actual state, $x$. $\Delta u$ represents the change in $u$ between two time steps. Typically the objective function is defined such that its convexity is guaranteed over the feasible region of the system. Constraints can also be applied to the controller such that the boundaries of terms $u$, $\Delta u$, or $x$ are limited to a given solution space. The ability to handle constraints is an important advantage of MPCs because real world dynamics often impose soft or hard constraints on performance. For example, there is a maximum torque that any engine is capable of outputting, and a controller should not demand a torque that is greater than this value.

The primary problem for implementation of an MPC is ensuring that the algorithm can be computed at a faster than real-time speed on control hardware. If the computation of the control input is slower than real-time, then the controller will not be able to keep up with real system dynamics. A common method of attaining real-time performance is linearization of the control-oriented model. If the control-oriented model is approximated as a linear system, the objective function is convex, and the constraints are linear, then the optimization problem may be solved by a quadratic programming (QP) solver algorithm [9]. The ability to implement the MPC optimization as a QP problem does not guarantee real-time implementation, but generally reduces computation time significantly compared to nonlinear optimization problems.

## 2.3 Power-Split Hybrid Powertrain Modelling

The introduction of hybrid electric vehicles (HEVs) to the automobile industry has been a significant part of a larger push for 'green' transportation options. HEVs combine the advantages of internal combustion vehicles and electric vehicles by integrating both power sources into a single powertrain system. One of the most common forms of HEV available for purchase in the modern consumer vehicle market is the power-split HEV. The Toyota Prius, Ford Focus Hybrid, and Lincoln MKZ Hybrid are all examples of power-split HEVs. The powertrain of power-split HEVs are designed such that they can switch powertrain operating modes to behave similarly to a pure electric vehicle, a series hybrid, or a parallel hybrid.

The versatility of power-split powertrains makes them significantly more complex than the powertrains of internal combustion vehicles, electric vehicles, or series and parallel HEVs. As a result,

modelling the dynamics of power-split powertrains for the purposes of vehicle dynamics control or energy management control can be challenging. Previous work in the literature has modelled power-split powertrains primarily using analytical models of the engine, electric motors, battery, and drivetrain. Analytical modelling has proved to be a robust method for simulating power-split powertrains, but such approaches have all required significant a priori knowledge of the powertrain control architecture of the vehicle [11] [12]. In [13] Syed et al. derives an analytical approach to modelling the dynamics of a Ford Escape Hybrid that separately models the power sources, driveline, and braking. Some transient behavior is captured by empirically determined transfer functions. This method generated good simulation results, but the model treated desired driveline torque from each power source and braking torque at each disc brake as inputs to the system. Ford provided Syed et al. with a model of the vehicle's powertrain control module that computes the required $APP$ and $BPP$ commands from desired torques [13]. Liu et al. proposed an analytical modelling approach for a Toyota Prius in [14], but bypassed the problem of modelling the vehicle's powertrain control module by replacing it with a custom rule-based controller. Liu extends this modelling approach for optimal control applications in [15]. No analytical modelling approach in literature has included the powertrain control module as a part of its system identification. Other approaches to modelling of power-split powertrains have relied heavily on experimentally determined maps of individual component performance. In [16] the authors describe a semi-empirical modelling approach for a Toyota Prius that utilized efficiency maps of the engine, motors, and the battery. Empirical approaches result in accurate models of system performance, but they require powertrain disassembly and extensive testing of individual components.

## 2.4 Longitudinal Vehicle Dynamics Control

Designing a suitable longitudinal vehicle dynamics control algorithm is a common requirement for both autonomous and semi-autonomous vehicles. Many modern production vehicles apply a longitudinal dynamics controller for adaptive cruise control (ACC) systems. ACC systems on production cars typically utilize radar, LIDAR, or cameras to measure the distance and relative velocity to the vehicle in front of the car during highway driving. Depending on the measured distance and velocity, an acceleration or braking command is used to maintain a minimum following distance. Development of

longitudinal vehicle dynamics control for autonomous driving is a more generalized form of the ACC problem where for any given time a desired velocity trajectory over a future time horizon is prescribed by the autonomous stack's local planner. Additional controller design considerations include the effects of vehicle steer angle on vehicle dynamics and the feasibility of the desired trajectory.

In addition to classical methods, numerous controller designs have been explored in literature for both ACC and autonomous longitudinal dynamics control. Ganji et al. proposes an ACC algorithm for a hybrid vehicle based on sliding mode control [17]. Moon et al. proposed a rule-based system that swaps between cruise control modes depending on driving situations [18]. Depending on whether the vehicle is in a normal driving or collision avoidance scenario, the controller will switch between linear quadratic control and a nonlinear method, respectively. Although many successful methods of advanced controls for both ACC and autonomous vehicle control have been developed, MPC has become the most frequently investigated method in literature. Both linear and nonlinear MPC has proven to be a particularly promising method for longitudinal vehicle control for three primary reasons: the desired trajectory of vehicle speed over a finite time horizon is usually known, the fundamental equations governing longitudinal vehicle dynamics are well understood, and there is usually an established set of constraints on control inputs and system states.

Controllers designed for traditional internal combustion (IC) engine vehicles or pure electric vehicles have typically assumed that a simple mapping from accelerator pedal position to wheel torque exists. Batra et al. proposed a non-linear MPC for anti-jerk cruise control of a Toyota Rav4 EV [19]. The paper primarily addresses the issue of half-shaft oscillations in electric vehicles while assuming a linear mapping from an input desired torque to torque at the wheels. In [20] Corona and Schumer propose a piecewise affine system MPC approach for ACC in an IC engine Smart car. The relationship between engine throttle and engine torque is modelled by a simple mapping for each gear. Li et al. presented an ACC design for a heavy-duty truck that highlighted the benefits of MPC by managing multiple objectives: ensuring that tracking error of a reference vehicle following distance converges to zero, ensuring ride comfort by limiting vehicle acceleration and jerk, and preservation of vehicle fuel economy [21]. In [22] the controller was implemented and experimentally validated on a test vehicle. The design employs an approach that separates the controller into two subsystems. A high-level linear

MPC prescribes a desired vehicle acceleration while a low-level controller handles the powertrain nonlinearities to select suitable values for throttle and brake inputs. It is straightforward to implement this approach on ICE or electric powertrain configurations, but handling of the powertrain dynamics is a more complex problem for power-split hybrid vehicles.

Most publications on designing longitudinal dynamics controllers for power-split hybrid vehicles have focused on optimizing fuel efficiency by designing custom powertrain control modules. Vajedi and Azad proposed a nonlinear MPC approach to design an ecological ACC (eco-ACC) for a Toyota Prius [23]. The eco-ACC utilizes an onboard map of upcoming road path and slope information to optimize demanded wheel torque over the prediction horizon, and the demanded torque is inputted directly to a custom-designed energy management system. Borhan et al. proposes a linearizing MPC strategy for tracking reference velocity and minimizing fuel consumption by controlling the speed and output torque of the engine [24]. In all previous literature, an accurate model of the hybrid powertrain control module is either provided by the OEM or is designed by the authors specifically for the relevant application. No methods for retrofitting a longitudinal dynamics controller to a power-split hybrid vehicle's existing powertrain control system have been explored.

# Chapter 3
# Power-Split Hybrid Powertrain and Brake Modelling

## 3.1 Lincoln MKZ Powertrain Architecture

The Lincoln MKZ is equipped with the Ford Motor Company© HF-35 power-split powertrain system that is controlled by a complex and proprietary hybrid energy management system. Like other power-split hybrids, the system includes an IC engine and two sources of electric power conversion, the traction motor and the generator. For modelling purposes, the powertrain system is divided into three primary subsystems. Certain elements of the drivetrain were identified using MATLAB/Simulink toolboxes, but the complete powertrain model was assembled in the acausal modelling environment, MapleSim. Figure 3.1 depicts a diagram of the powertrain control architecture.

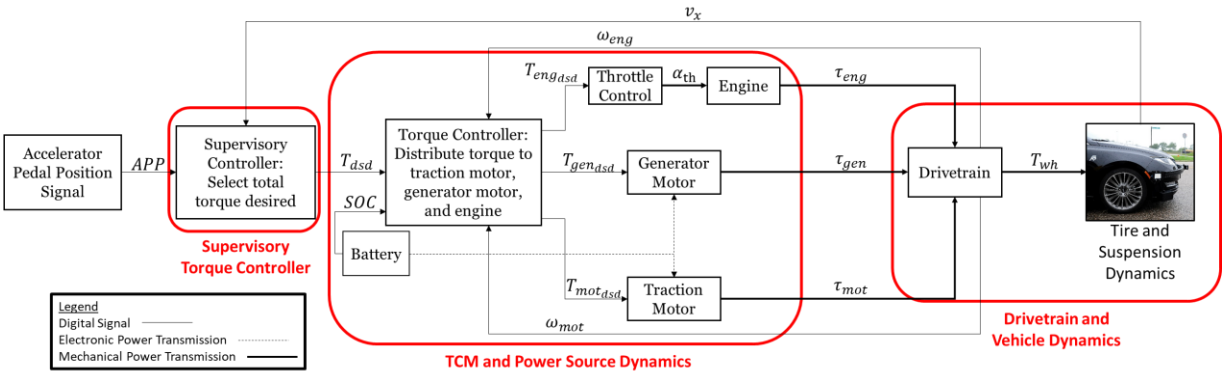


**Figure 3.1: The Lincoln MKZ Hybrid powertrain control architecture.**

The diagram in Figure 3.1 accounts for the effects of only one input to the system, $APP$, on powertrain dynamics. As an aside, the powertrain system also responds to the gear selection (Park, Reverse, Neutral, Drive, or Low) and the brake pedal position ($BPP$) input. For the application discussed in this thesis, the only relevant selection of gear is Drive. For reasons that are discussed later in this chapter, the effects of $BPP$ on powertrain dynamics could not be modelled. A separate braking model that is decoupled from the powertrain dynamics is discussed in Section 3.3.

The supervisory torque controller maps $APP$, expressed as a range of 0 to 100%, to a desired total output torque at the wheels, $T_{dsd}$. Since total available torque typically depends on the speed of the vehicle, $v_x$, it is also an input to the subsystem. Ford did not provide a model of the $T_{dsd}$ mapping.

The torque control module (TCM) and power source dynamics compose the second subsystem. The TCM receives $T_{dsd}$ from the supervisory torque controller and takes measurements of battery state of charge (SOC), and rotational speeds of the traction motor, $\omega_{mot}$, and engine, $\omega_{eng}$. The TCM then uses a proprietary Ford decision-making algorithm to determine desired torques for each of the three power sources. The desired torques, $\tau_{eng_{dsd}}$, $\tau_{mot_{dsd}}$, and $\tau_{gen_{dsd}}$, are inputted to their local control modules, and unknown internal dynamics determine the actual torques, $\tau_{eng}$, $\tau_{mot}$, and $\tau_{gen}$.

The final subsystem is composed of the drivetrain and its interface with vehicle dynamics model. The output shafts of the engine, motor, and generator transfer torques to the drivetrain. Torque flows through the drivetrain and is outputted as torques at the front right wheel, $\tau_{FRwh}$, and front left wheel, $\tau_{FLwh}$. At the start of this project the layout of the drivetrain was known, but none of the kinematic or dynamic parameters were provided by Ford.

## 3.2 Powertrain Subsystem Modelling

As discussed in Section 3.1, at the beginning of this project there was limited a priori knowledge of the Lincoln MKZ's powertrain system. Many of the components, such as the supervisory controller and the TCM, contained a completely opaque control logic. To address these components of the powertrain, a grey-box modelling approach was proposed. Portions of the system where only inputs and outputs can be measured were modelled using NNs, while portions of the system with well-understood components were modelled using analytical approaches.

### 3.2.1 Supervisory Torque Controller

The Supervisory Torque Controller determines $T_{dsd}$ as a function of $APP$ and $v_x$. Observation of experimental data for $T_{dsd}$ also indicated that the supervisory controller implemented some form of transfer function to smooth the controller's performance. This observation is supported by a 2009 patent

owned by Ford outlining a proposed supervisory controller for implementation on future vehicles [25]. The selected approach for modelling this controller is a double layer perceptron NN with time-delayed feedback of the output parameter, $T_{dsd}$, at a time step of 0.05 seconds. The time step was selected based on the available sample rate of experimental data. At any given time step, $k$, $T_{dsd_k}$ is defined as a function of $APP$, $v_x$, and $T_{dsd_{k-1}}$. A modified form of Equation 1.1, written as Equation 3.1, defines the supervisory controller model:

$$T_{dsd_k} = W2\, Sig(W1\, X + B1) + B2 \tag{3.1}$$

where $X$ is a $3 \times 1$ column vector of the inputs $APP$, $v_x$, and $T_{dsd_{k-1}}$. Saturation limits on the range of possible $T_{dsd}$ were applied to the output of the NN to ensure its behavior is always within the physical limits of the real supervisory controller. Section 5.2.1 describes details of parameter identification for the supervisor controller.

### 3.2.2 TCM and Power Source Dynamics

For the purposes of the model, the TCM, traction motor dynamics, generator dynamics, and engine dynamics were lumped into a single subsystem. This choice is due to the interdependent behavior of each of these portions of the control system. The high voltage battery distributes power to the generator and traction motor; however, the high voltage battery also regulates charge of the vehicle's low voltage battery, which in turn powers onboard electronics, air conditioning, and other vehicle systems. Because of these factors, SOC of the high voltage battery depends on several factors that are outside the scope of this work. Battery SOC is hence treated as a random and measurable disturbance variable for the TCM and power source dynamics subsystem.

The TCM of the power-split hybrid powertrain is more complex than traditional internal combustion or fully electric vehicles because there are multiple modes of operation to consider. Each operating mode adjusts how desired torque distributes to the power sources and alters the kinematic constraints of the drivetrain. In the case of the Lincoln MKZ's power-split powertrain, discrete modes of operation were identified by observation of experimental data, referencing the model of a previous generation of

the powertrain described in [13], and referencing Ford's hybrid electric control software documentation. The following modes were identified for forward driving:

1. EV mode: This mode engages when torque demand is sufficiently low and battery SOC is sufficiently high for the engine to remain off. In this mode, the engine shaft is locked in place.

2. Engine cranking: The TCM has determined that power output from the engine is required, and the engine shaft is unlocked. Positive torque is produced by the generator to accelerate the engine up to its ignition speed.

3. Power-split mode: At high speeds or high torque demands, the engine is running and being used to provide torque to the driveshaft. Depending on overall desired torque, the generator and traction motor may be used to either charge the high voltage battery or transmit additional torque to the driveshaft. Two sub-modes of power-split mode exist.

    a. Positive-split: If the high voltage battery is below a threshold SOC, then torque from the engine splits between the path to the driveshaft and the path through the generator to charge the high voltage battery.

    b. Negative-split: This mode is not preferred but is necessary when the high voltage battery is fully charged and the vehicle speed is high. The generator transmits torque through the planetary gear train to drive the vehicle. Due to the kinematics of the planetary gear set, torque from the generator also regulates the engine speed to keep it in the high efficiency operating range. Negative-split mode establishes a power circulation path where some of the power produced by the generator returns to the high-voltage battery through the traction motor (negative motor torque value). Power circulation results in negative-split mode being a less efficient mode of operation than positive-split.

Additional modes of operation exist for when the vehicle is parked or driving in reverse, but modelling them is outside the scope of this thesis. Figure 3.2 depicts a sample window of experimental data where each operating mode is observed.

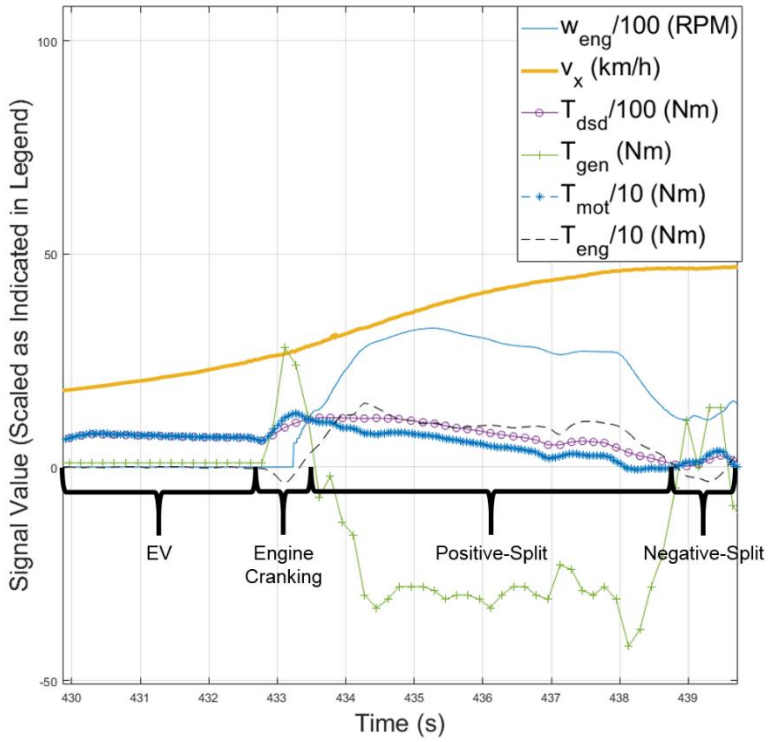


**Figure 3.2: Snapshot of experimental data showing the power-split powertrain switching between operating modes.**

EV, engine cranking, and power-split modes were each denoted by indices 1, 2, and 3, respectively (positive and negative-split were not differentiated by mode). Each experimental data sample was categorized into one of the operating modes based on measured engine speed, generator torque, and operating mode of the previous time step.

The modelled switching behavior of the TCM uses a system of two double layer classifier networks. Both classifiers are feed-forward networks with a hidden sigmoid layer and a softmax output layer, as described in Section 3.1. The first classifier network predicts the conditions for the TCM to begin starting the engine (switch from EV mode to engine cranking mode). The network predicts if the mode

will switch to engine cranking mode based on the following inputs: $v_x$, $SOC$, and $T_{dsd}$. A modified form of Equation 1.3, written as Equation 3.2, defines the engine-start mode selection model:

$$Y_{engoff} = W2\, Sig(W1\, X + B1) + B2 \tag{3.2}$$

where $X$ is a $3 \times 1$ column vector of the inputs $v_x$, $SOC$ and $T_{dsd_k}$, and $Y_{engoff}$ is a $2 \times 1$ column vector of the probabilities that it is currently EV mode, $y_1$, or engine cranking mode, $y_2$. If $y_2 > y_1$ the TCM commands that the operating mode switches to engine cranking.

The second classifier network determines the current TCM mode if the engine shaft is currently rotating. The network predicts if the TCM would select engine cranking mode, power-split mode, or EV mode (meaning the engine shaft begins braking) based on the following inputs: $v_x$, $SOC$, $T_{dsd}$, $\omega_{eng}$, and the operating mode of the TCM at the previous sample time, $k-1$. A modified form of Equation 1.3, written as Equation 3.3, defines the engine-on mode selection model:

$$Y_{engon} = W2\, Sig(W1\, X + B1) + B2 \tag{3.3}$$

where $X$ is a $5 \times 1$ column vector of the inputs $v_x$, $SOC$, $T_{dsd_k}$, $\omega_{eng}$, and $mode_{k-1}$, and $Y_{engon}$ is a $3 \times 1$ column vector of the probabilities that it is currently EV mode, $y_1$, engine cranking mode, $y_2$, or power-split mode, $y_3$. The largest index value of $Y$ is the predicted operating mode. In addition to the two mode selection NNs, constraints on mode switching behavior as a function of current states were applied. The constraints were based on observations of experimental powertrain performance and available Ford documentation, and are used to prevent the model from erroneously entering an impossible set of system states. The additional mode switching constraints are listed below:

1. If $T_{dsd} < 0$ the powertrain will never switch from EV mode to engine cranking mode.

2. If the powertrain is in power-split mode and the engine speed drops below a threshold of operating efficiency, $\omega_{brake}$, the powertrain switches to EV mode to conserve fuel.

3. If the powertrain is in engine cranking mode and the engine speed rises above the threshold ignition speed, $\omega_{ignite}$, then the powertrain switches to power-split mode.

4. Once engine cranking mode has initiated, the powertrain cannot switch directly back into EV mode without reaching engine ignition.

Once the TCM has selected the powertrain operating mode, desired torques $\tau_{eng_{dsd}}$, $\tau_{mot_{dsd}}$, and $\tau_{gen_{dsd}}$ are selected. The low-level dynamics of each power source determines their actual torque outputs $\tau_{eng}$, $\tau_{gen}$, and $\tau_{mot}$, respectively; however, limitations of the data acquisition sample rate prevented desired and actual torque signals from being measured at a sample rate greater than 10Hz. The sample rate was insufficient to determine a transient response between desired torque and actual torques, so the low-level dynamics of each power source were neglected. Instead the mapping from $T_{dsd}$ to $\tau_{eng}$, $\tau_{gen}$, and $\tau_{mot}$ was modeled using an individual double layer perceptron NN. Determining a method of modelling power source transients will be a part of future work. The inputs to the NN are: $T_{dsd}$, $\omega_{eng}$, $\omega_{mot}$, $mode$, and $SOC$. Based on these inputs, the network assumes quasi-static behavior of each power source, so the model does not include transient behavior. A modified form of Equation 1.1, written as Equation 3.4, defines the torque selection model:

$$\{\tau_{eng}; \tau_{mot}; \tau_{gen}\} = W2\, Sig(W1\, X + B1) + B2 \tag{3.4}$$

where $X$ is a $5 \times 1$ column vector of the inputs $T_{dsd_k}$, $\omega_{eng}$, $\omega_{mot}$, $mode$, and $SOC$. Saturation limits on the ranges of $\tau_{eng}$, $\tau_{gen}$, and $\tau_{mot}$ were applied to the output of the NN to ensure its behavior is always within the physical limits of the real powertrain system. Section 5.2.1 describes details of parameter identification for the TCM.

### 3.2.3 Drivetrain Dynamics

Unlike the other two subsystems, sufficient measurable signals were available to identify a physics-based model of the drivetrain. The drivetrain configuration of the Lincoln MKZ is depicted in Figure 3.3.

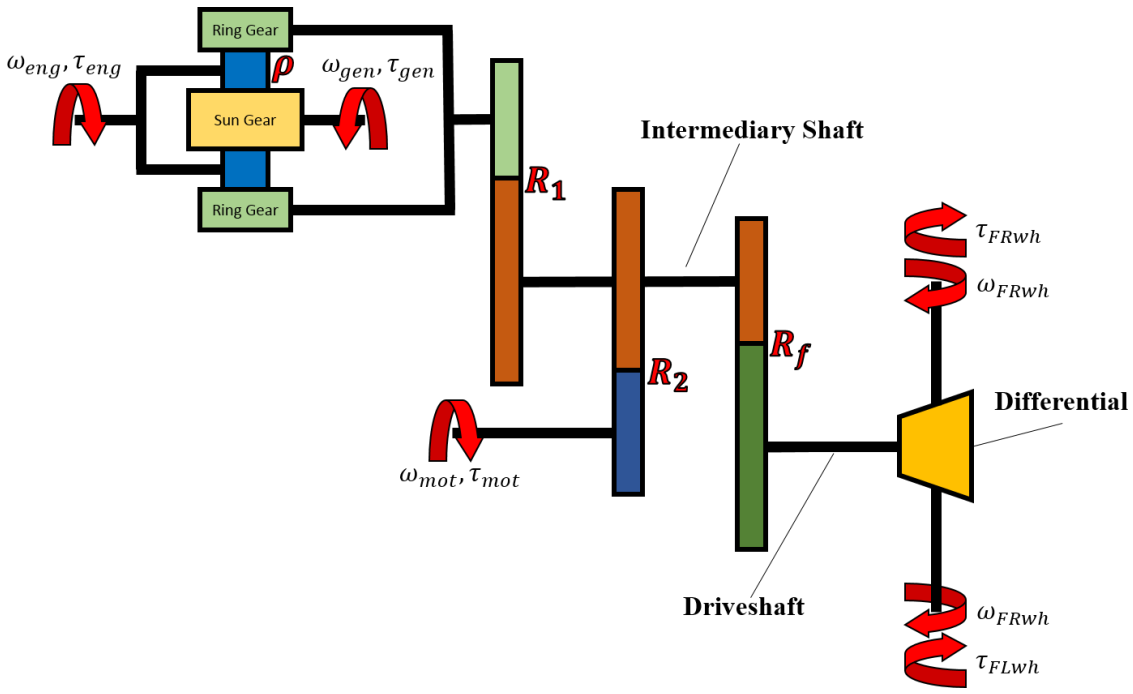


**Figure 3.3: Diagram of the Lincoln MKZ Hybrid's power-split drivetrain configuration.**

Arrows indicate how positive directions of rotation, ω, and torque, τ, are defined. As depicted in Figure 3.3, the system includes a planetary gear set that connects the output shafts of all three power sources. The generator drives the sun gear of the planetary while the engine drives the planet carrier. The planetary gear ratio is defined by Equation 3.5:

$$\rho = \frac{d_{sun}}{d_{ring}} \tag{3.5}$$

where $d_{sun}$ and $d_{ring}$ are the pitch diameters of the sun and ring gears, respectively. The ring gear connects to the output shaft of the traction motor through fixed gear ratios $R_1$ and $R_2$. Similarly, both the ring gear and traction motor shaft connect through the intermediary shaft and final reduction ratio to the driveshaft. The driveshaft connects to the right and left halfshafts through an open differential. For the purposes of longitudinal dynamics modelling and control, it is useful to replace individual wheel speeds and torques with a virtual speed and torque at the driveshaft. In this model the drivetrain reduction that exists in the differential has been lumped with the final drivetrain reduction to form a

single ratio, $R_f$, and the wheel torques are assumed to always be equal. On the real drivetrain system compliance of the halfshafts and high frequency tire dynamics will create small differences in wheel torques, but for simplicity these effects are neglected. The simplified kinematic and dynamic equations relating the halfshafts to the driveshaft are shown in Equations 3.6, 3.7, and 3.8:

$$\tau_{ds} = \tau_{FRwh} + \tau_{FLwh} \tag{3.6}$$

$$\tau_{FRwh} = \tau_{FLwh} \tag{3.7}$$

$$\omega_{ds} = \frac{\omega_{FRwh} + \omega_{FLwh}}{2} \tag{3.8}$$

where $\tau_{ds}$ and $\omega_{ds}$ are the torque and speed of the driveshaft, respectively.

After lumping the components of the right and left wheels into a single driveshaft component, the layout of Figure 3.3 shows that the model has only two degrees of freedom (DOF), which are defined as $\omega_{eng}$ and $\omega_{mot}$. The kinematics and dynamics of the drivetrain are dependent on the powertrain operating mode. When operating in EV mode the engine shaft locks, forcing $\omega_{eng}$ to zero, and the system reduces to have one DOF. The kinematics of the drivetrain in EV mode are defined by Equations 3.9 and 3.10a:

$$\omega_{ds} = \frac{\omega_{mot}}{R_2 R_f} \tag{3.9}$$

$$\omega_{gen} = -\frac{R_1}{R_2 \rho}\omega_{mot} \tag{3.10a}$$

The engine shaft is unlocked when operating in engine cranking or power-split modes. The additional DOF added by the engine shaft changes the kinematics and dynamics of the drivetrain system. For these operating modes Equation 3.10a is replaced by 3.10b:

$$\omega_{gen} = \frac{1+\rho}{\rho}\omega_{eng} - \frac{R_1}{R_2 \rho}\omega_{mot} \tag{3.10b}$$

The gear train efficiency losses are dependent upon direction of torque transmission at each gear meshing, but efficiencies were assumed constant and equal in each direction of torque transmission. This model behavior is well suited for the acausal modelling environment of MapleSim. For forward velocity and acceleration in EV mode, the dynamics are defined by Equations 3.11, 3.12, and 3.13a:

$$I_{gen}\dot{\omega}_{gen} = \tau_{gen} - \tau_s \tag{3.11}$$

$$I_{mot}\dot{\omega}_{mot} = \tau_{mot} - \frac{1}{R_f R_2 \eta_f \eta_2}\tau_{ds} - \frac{R_1 \eta_{r1}}{R_2 \eta_f \eta_s \rho}\tau_s \tag{3.12}$$

$$0 = \tau_{eng} + \frac{1}{\eta_s}\left(1 + \frac{1}{\rho}\right)\tau_s \tag{3.13a}$$

where $\tau_s$ is the torque applied to the sun gear by the planet gears, and $I_{gen}$ and $I_{mot}$ are the lumped rotational inertias of the generator and the motor respectively. $I_{gen}$ is the inertia of the generator and sun gear. $I_{mot}$ combines the inertias of the motor, ring gear, intermediary shaft, gear meshings $R_1$, $R_2$, and $R_f$, the driveshaft, and the halfshafts. The components of $I_{mot}$ are lumped at the motor. For engine cranking and power-split operating modes, Equation 3.13a is replaced by Equation 3.13b:

$$I_{eng}\dot{\omega}_{eng} = \tau_{eng} + \frac{1}{\eta_s}\left(1 + \frac{1}{\rho}\right)\tau_s \tag{3.13b}$$

where $I_{eng}$ is the lumped rotational inertia of the engine, which includes the inertia of the engine, the carrier, and the planets about the axis of the engine shaft. The inertia contributions of rotation of the planet gears about their own axes are typically considered very small in power-split powertrain configurations [13], so they are neglected. The efficiency terms are summarized in table 3.1.

**Table 3.1: Summary of drivetrain lumped efficiency terms.**

| | |
|---|---|
| $\eta_{r1}$ | Lumped efficiency between the ring gear and the planet gears, through gear ratio $R_1$, and through the differential |
| $\eta_2$ | Efficiency through ratio $R_2$ |
| $\eta_f$ | Efficiency through ratio $R_f$ |
| $\eta_s$ | Efficiency between the sun gear and the planet gears |

When switching from engine unlocked to engine locked operating modes, a braking torque is applied to the engine shaft. The applied braking torque is a proportional feedback of the current engine shaft speed, $\omega_{eng}$, that is multiplied by the gain term $K_{engoff}$. $K_{engoff}$ was tuned manually to ensure engine braking behavior matched experimental data.

The complete drivetrain subsystem was modelled using the 1-D mechanical library of MapleSim.

## 3.3 Brake Modelling

Braking behavior of the Lincoln MKZ is more complex than traditional IC engine vehicles. Like many other hybrid vehicles, it increases fuel efficiency by implementing regenerative braking behavior in addition to mechanical braking. During regenerative braking, power flows from the wheels through the drivetrain to apply negative torques to the motor and generator. The motor and generator both absorb the torque and use it to charge the high voltage battery. Similar to APP in the powertrain model, the supervisory controller uses $BPP$ to determine a desired braking torque applied at the wheels. $BPP$ is represented as a unitless range from 0.1385 to 0.5. For any given set of vehicle states, the TCM selects the proportion of desired braking torque that will come from regenerative braking, and a separate braking control module (BCM) allocates how the remaining desired torque will be generated by mechanical braking at each wheel.

Due to limitations of measurable signals during experimentation, it was not possible to separately measure the effects of regenerative braking and mechanical braking in the front wheels; only total braking torque was measurable at each wheel. This introduced a problem for integrating a braking model with the powertrain model used for APP mapping since regenerative and mechanical braking are applied to different parts of the drivetrain subsystem. The proposed solution is a black-box approach that utilizes a cascading system of NNs. The model calculates a front axle torque $\tau_{FB}$ and a rear axle torque $\tau_{RB}$. For the purposes of the model only the longitudinal dynamics of braking were considered, so distribution of braking torque to the right and left wheels was assumed to be equal. Figure 3.4 depicts a diagram of the braking model.

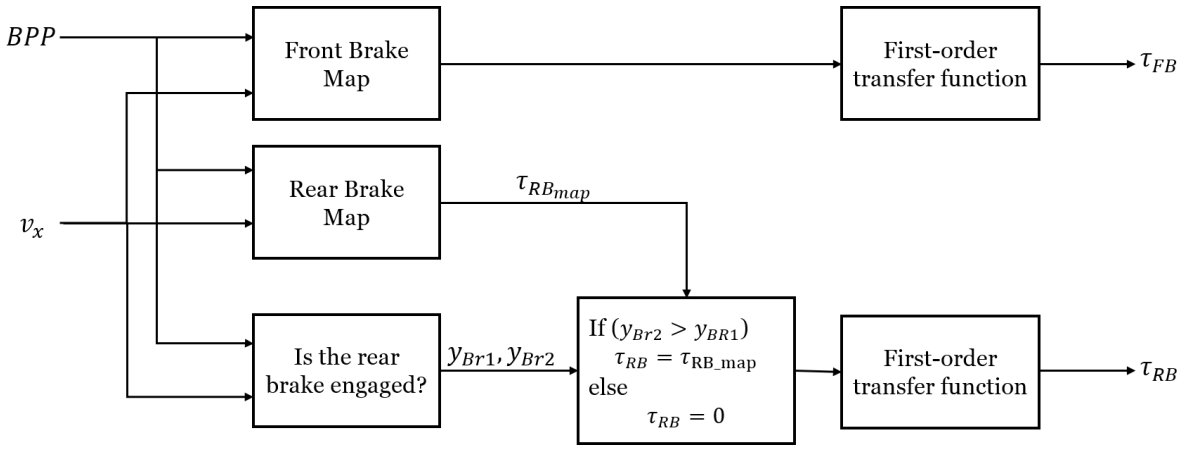


**Figure 3.4: Diagram of the braking model.**

As depicted in Figure 3.4, the braking model is dependent on only two inputs, $BPP$ and $v_x$. $\tau_{FB}$ is calculated directly by a front brake mapping that is modelled with a double layer perceptron NN. A modified form of Equation 1.1, written as Equation 3.14, defines the front brake map model:

$$\tau_{FB} = W2\, Sig(W1\, X + B1) + B2 \tag{3.14}$$

where $X$ is a $2 \times 1$ column vector of the inputs $BPP$ and $v_x$.

Calculation of $\tau_{RB}$ required a more complex mapping. Observation of experimental data showed that for a subset of the input space, $\{BPP, v_x\}$, no rear braking torque is applied. A classifier NN was

implemented to predict if, for any given combination of $BPP$ and $v_x$, the rear brakes will be engaged. A modified form of Equation 1.3, written as Equation 3.15, defines the front rear braking engagement model:

$$Y_{Br} = softmax(W2\ Sig(W1\ X + B1) + B2) \tag{3.15}$$

where $X$ is a $2 \times 1$ column vector of the inputs $BPP$ and $v_x$, and $Y_{Br}$ is a $2 \times 1$ column vector of the probabilities that the rear brakes are not engaged, $y_1$, and that the rear brakes are engaged, $y_2$. If $y_2 > y_1$, then the rear brakes are engaged. $\tau_{RB}$ is set to zero if the brakes are disengaged.

If the rear brakes are engaged, then $\tau_{RB}$ is calculated by the rear braking map. The rear braking map was modelled with another double layer perceptron NN. A modified form of Equation 1.1, written as Equation 3.16, defines the rear brake map model:

$$\tau_{RB_{map}} = W2\ Sig(W1\ X + B1) + B2 \tag{3.16}$$

where $X$ is a $2 \times 1$ column vector of the inputs $BPP$ and $v_x$.

For the purposes of reducing oscillations during simulation, first-order transfer functions with very low time constants are applied to the outputs of the brake model, $\tau_{FB}$ and $\tau_{RB}$. These transfer functions increase the brake model from an index-0 to an index-1 system to increase model stability during simulation.

## 3.4 Powertrain and Brake Model Integration

The limitations of the available braking torque signals, and the subsequently chosen method of modelling the brakes, created a problem for integration of the two models. The output torques of the braking model, $\tau_{FB}$ and $\tau_{RB}$, are applied directly at the wheels of the vehicle model. The model was identified based on the torques measured at each wheel during experiments. The left and right wheel components of $\tau_{FB}$ cannot be applied directly to the halfshafts of the drivetrain model, as it is formulated, because the map of $\tau_{FB}$ implicitly includes the inertial effects of the drivetrain. It is not

sufficient to subtract an inertial torque proportional to $I_{mot}$ from $\tau_{FB}$ because, based on the drivetrain dynamics, there is coupling between $\omega_{eng}$ and $\omega_{mot}$.

The proposed solution to this problem is to introduce a virtual switching clutch component at the drivetrain model's driveshaft that engages and disengages as a function of BPP. The clutch engages when there is a negligible brake pedal input applied to the system. During this mode of operation, the drivetrain model behaves exactly as described in Section 3.2.3. Application of the brake pedal input, defined as a BPP value greater than 0.14, disengages the clutch at the driveshaft to decouple torque at the front wheels from the inertia of the drivetrain. A virtual powertrain braking torque, $\tau_{PBr}$, is then applied on the inboard side of the driveshaft clutch to maintain a very small relative speed, $\omega_{rel}$, between the inboard and outboard sides of the clutch. This limits torque oscillations during clutch reengagement. Future work will re-examine this ad hoc solution. Figure 3.5 depicts a diagram of how the drivetrain behaves during braking maneuvers.

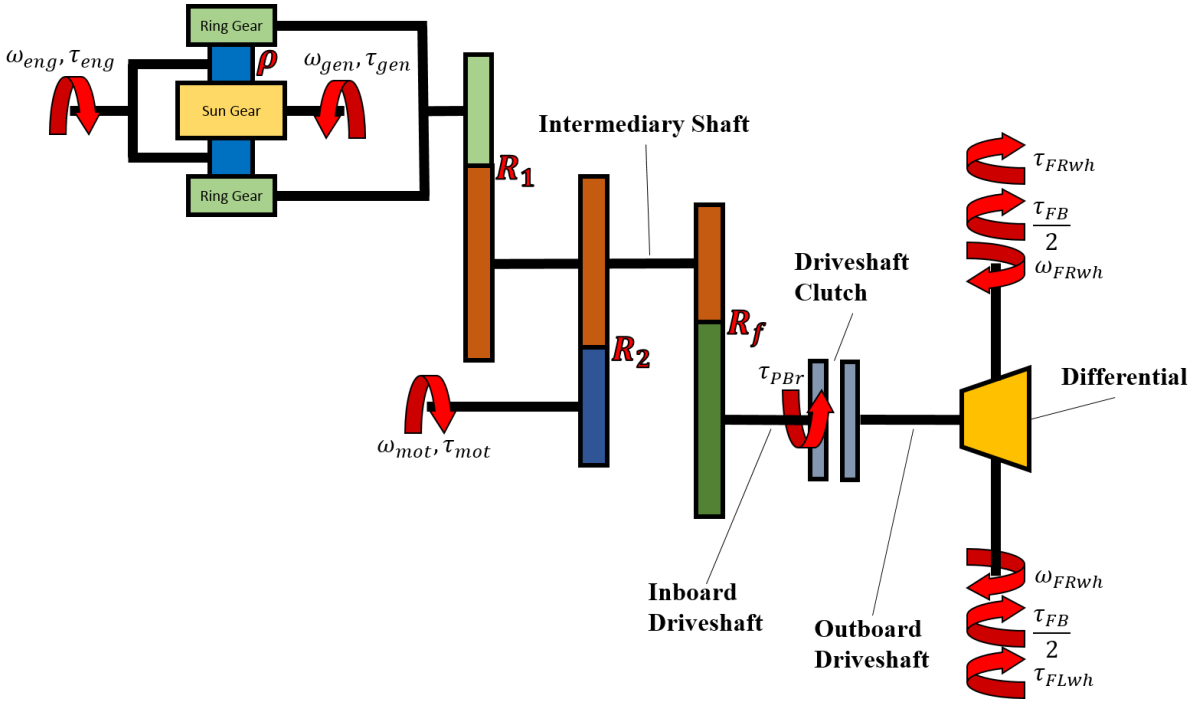


**Figure 3.5: Drivetrain diagram for braking maneuvers.**

$\tau_{PBr}$ is calculated as a function of $\omega_{rel}$ by Equation 3.17:

$$\tau_{PBr} = K_{PBr}\omega_{rel} \tag{3.17}$$

where $K_{PBr}$ is a high gain proportional term that was tuned manually. The clutch is a component of the drivetrain MapleSim model. Parameters of the clutch component were manually tuned to ensure smooth simulation performance and minimal oscillations during engagement and disengagement.

## 3.5 Interaction with Vehicle Dynamics Model

The combined powertrain and braking model is modular such that it can be integrated with any type of vehicle dynamics model in MapleSim. The two drivetrain halfshaft flanges connect with two front wheel components while the braking torques are applied at each wheel using a MapleSim brake component. The details of the vehicle dynamics model, including tire dynamics, suspension configuration, and chassis inertial terms, are independent of the powertrain and braking models. Van Gennip developed a high-fidelity model of the Lincoln MKZ's vehicle dynamics [26] to integrate with the powertrain and braking models. This vehicle dynamics model was integrated with the powertrain and braking models for all simulations discussed in later chapters.

# Chapter 4
# Experimentation

## 4.1 Apparatus

Multiple apparatus were used during vehicle road testing to gather data for model parameter identification. The full suite of data acquisition systems included the A&D Vehicle Measurement System, the CAN bus data acquisition system, and signals read from the Moose's onboard computer. In this chapter each system is described in detail. A summary of measured signals and sampling rates used for powertrain and braking parameter identification is in Appendix A.

### 4.1.1 Vehicle Measurement System

The Vehicle Measurement System (VMS), by A&D Technology, is a complex system of sensors designed for vehicle road testing. The VMS consists of three subsystems of sensors packaged at each wheel. Due to some component failures, the VMS system was not available for the rear right wheel of the vehicle. Figure 4.1 depicts the VMS mounted at the Moose's front left wheel.

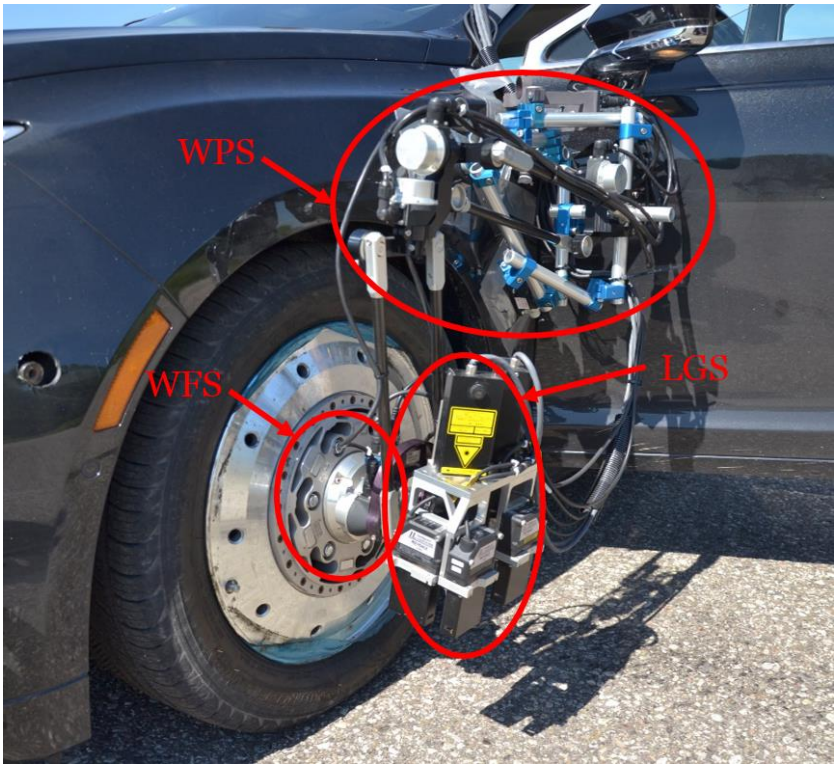


**Figure 4.1: The VMS sensor packaging mounted at the front left wheel of the Moose.**

Wheels with custom VMS compatible hubs replaced the stock vehicle wheels for road testing. The custom hubs were designed such that the first sensor subsystem, the wheel force sensor (WFS), mounts at the center of the hub. The WFS consists of an array of strain gauges that measure force and moment through each principal axis of the hub. The measured moment about the

rotation axis of each wheel corresponds to wheel torques in the powertrain and braking models. In addition, the WFS includes a digital encoder that measures the speed of rotation, $\omega_{wh}$, at each wheel.

The second VMS subsystem is the wheel position sensor (WPS). The WPS connects at the center of the wheel hub, and is mounted to the chassis through a set of rigid linkages with 5 DOF. Each DOF results from a revolute joint with a built-in digital encoder. The VMS monitors the angle of rotation of all five digital encoders to determine the translational and rotational position of the wheel relative to

the chassis. These measurements are not required for powertrain parameter identification, but they can be useful for modelling suspension kinematics and dynamics.

The third VMS subsystem is the laser ground sensor/laser doppler velocimeter (LGS). This subsystem uses an array of laser sensors to detect the ground speed at each wheel, the effective radius of the tire, and other signals. The LGS was not used for any part of the powertrain or braking model parameter identification, but it was important for suspension and tire dynamics parameter identification [27].

### 4.1.2 Vector CANalyzer and Diagnostics

In addition to the signals measured at the wheel by the VMS, parameter identification required measurement of several internal powertrain system states. These states could only be measured through the vehicle's control area network (CAN) bus. The CAN bus signals were measured with a Vector CAN bus measurement Tool and the Vector CANalyzer software. The generic OBD II list of signals were automatically measurable through the CAN bus, and $APP$, $\omega_{eng}$, $v_x$, and $SOC$ could be measured through this protocol. The CANalyzer simultaneously records signals from the VMS, so no CAN bus data synchronization was required during parameter identification pre-processing; however, parameter identification also required measurements of $\omega_{mot}$, $\omega_{gen}$ and all internal powertrain torques.

The Ford VCM II is a diagnostics tool that is capable of sampling all required powertrain signals through a proprietary set of CAN bus commands, but it cannot log the diagnostic signals in a usable format. By using a CAN bus cable splitter, both the VCM II and the Vector tool were simultaneously connected to the Moose's CAN bus. The VCM II sent command messages requesting packets of diagnostics powertrain data, and the return messages that contained the data were recorded by the CANalyzer. The limitation of this approach is that all diagnostic signals were restricted to a sample rate of approximately 10 Hz while all other signals were available at either 100Hz or 50Hz.

### 4.1.3 Brake Pedal Position Data Acquisition

According to Ford diagnostic signal documentation, $BPP$ was not available for measurement through the VCM II. $BPP$ had to be measured and recorded using the Moose's onboard Linux computer system. The Moose's computer is capable of reading $BPP$ using a Dataspeed hardware component, the Throttle-Brake Combination By-Wire Interface. The Moose also uses this hardware to send $APP$ and $BPP$ commands to the vehicle when it is operating in autonomous driving mode. The module communicates with the vehicle using a custom CAN bus command protocol summarized in Dataspeed Incorporated's documentation [28]. Figure 4.2 depicts the computer's data acquisition user interface during road testing.

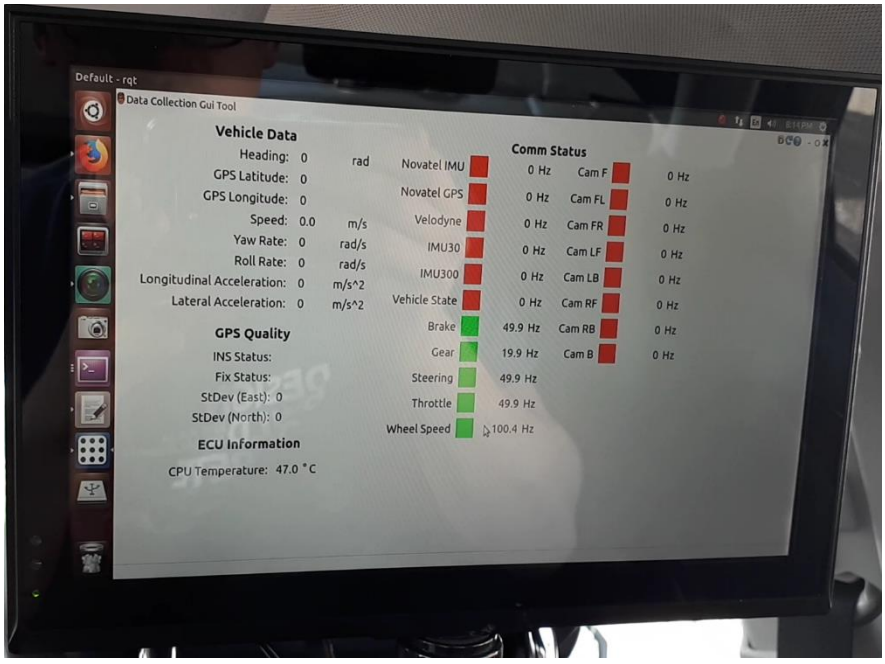


**Figure 4.2: Onboard display of the Moose's computer during road testing.**

$BPP$ data was recorded at a rate of 50Hz. During parameter identification, pre-processing the $BPP$ data's timestamp was synchronized with CANalyzer data using the MATLAB signal processing toolbox.

## 4.2 Vehicle Testing Procedure

Selecting an appropriate test procedure for vehicle dynamics and powertrain parameter identification usually depends on the unknown parameters. For any parameters that need to be identified, it is a necessary but not sufficient condition that the inputs to the system, such as acceleration command or steering angle, create a system behavior that is persistently exciting. For linear system models, the required conditions for persistence of excitation are well established [29], but high-fidelity models of vehicle systems are highly nonlinear. In addition, identification of NN-based models require more than a persistently exciting input. Shallow NN training requires large data sets that cover the full input space. To meet this requirement, a large set of vehicle tests were performed, but specific drive cycle test plans were not generally followed. Unlike some types of vehicle testing, such as vehicle fuel-economy benchmark tests, neural network modelling benefits from randomness in driver behavior. Most vehicle testing occurred at the Region of Waterloo's Emergency Services Training Centre, which has a vehicle maneuvering test track. However, some public road testing data was used to augment the data sets for supervisory controller and TCM neural network training. The speed profiles of some sample test maneuvers are depicted in Figure 4.3.

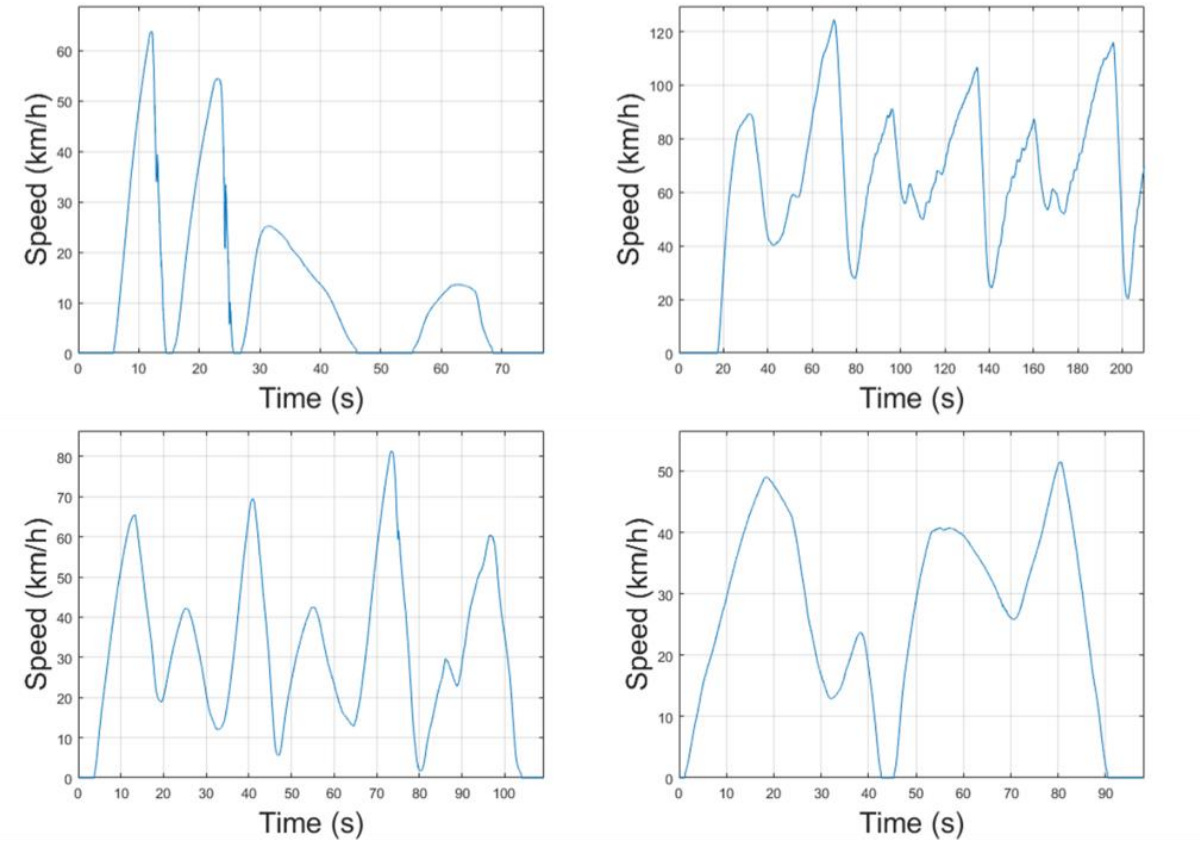


**Figure 4.3: Speed profile data for braking maneuvers with various BPP values (top left), laps of the Waterloo regional test track (top right), Hard accelerations from various starting speeds (bottom left), and public driving on a road with traffic lights (bottom right).**

# Chapter 5
# Parameter Identification

## 5.1 Drivetrain Parameters

Due to limited a priori knowledge of the drivetrain, all gear ratios, gear train efficiencies, and lumped inertia parameters needed to be identified from experimental data. Gear train ratios were identified from experimental measurements of $\omega_{eng}$, $\omega_{gen}$, $\omega_{mot}$, $\omega_{FRwh}$, $\omega_{FLwh}$ and Equations 3.8, 3.9, 3.10a, and 3.10b. However, without a measurement of speed for the intermediary shaft, not all gear ratios could be identified individually. Instead of identifying $R_1$, $R_2$, and $R_f$ directly, lumped gear ratios $R_f R_1$ and $R_f R_2$ were identified in addition to the planetary ratio $\rho$. Identification of gear ratios was performed by determining the mean ratios between measurements of $\omega_{eng}$, $\omega_{gen}$, $\omega_{mot}$, and $\omega_{ds}$ from multiple driving test runs. Each ratio was determined with a standard deviation of less than 1%. Table 5.1 summarizes the mean and standard deviation of each effective gear ratio.

**Table 5.1: Effective drivetrain gear ratios.**

| Symbol | Mean Value | Standard Deviation |
|---|---|---|
| $\rho$ | 0.395 | 0.001 |
| $R_f R_1$ | 4.08 | 0.03 |
| $R_f R_2$ | 10.4 | 0.05 |

For identification of drivetrain dynamics parameters, the MapleSim drivetrain model was converted to an S-function. Parameter identification was performed using the Simulink Parameter Identification toolbox. For each iteration of parameter identification, a test's experimental measurements of $\tau_{eng}$, $\tau_{mot}$, $\tau_{gen}$, $\tau_{FLwh}$, $\tau_{FRwh}$, and powertrain mode were applied to the drivetrain model, and experimental measurements of $\omega_{eng}$ and $\omega_{mot}$ were set as references for the drivetrain model to track. Using a

nonlinear least squares trust-region reflective algorithm, the parameter estimator attempted to identify the inertial and efficiency parameters that resulted in the minimum mean-squared error between experimental and simulated measurements of $\omega_{eng}$ and $\omega_{mot}$. Identification of all dynamic parameters, which includes the three lumped inertias and four gear meshing efficiencies, was challenging because all seven parameters needed to be identified for a system of only two DOFs. It was not possible to identify a unique set of parameters from experimental data.

A sensitivity analysis was performed on the drivetrain model to observe how model performance was affected by a random variance in each parameter within their range of possible values. The influence of each parameter was measured by observing how the simulated outputs for $\omega_{eng}$ and $\omega_{mot}$ were affected by parameter variance. Figure 5.1 is a tornado plot summarizing the parameter influence results of the sensitivity analysis.

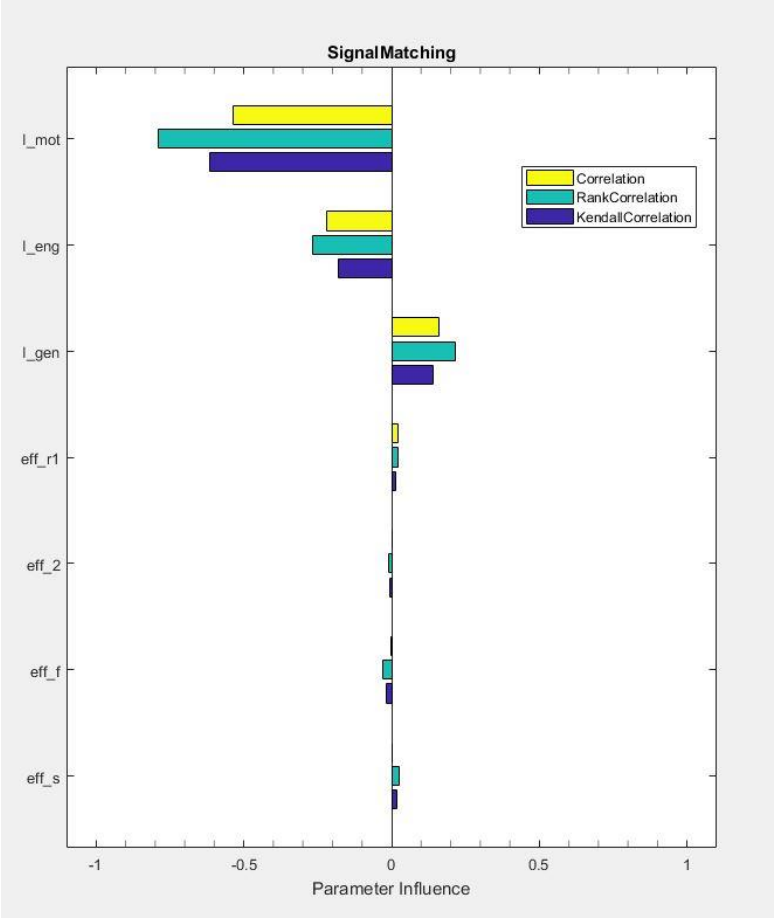


**Figure 5.1: Tornado plot showing correlations between parameters and model performance.**

It is clear from Figure 5.1 that the Kendal, Spearman (rank), and linear correlations for the three lumped inertial parameters is much greater than the four efficiency terms. Based on the results of sensitivity analysis, each efficiency term was set to a generic reference value of 0.98. This simplification of the model made identifying unique values for $I_{eng}$, $I_{mot}$, and $I_{gen}$ possible. Depending on the road test used, each iteration of parameter identification converged to marginally different values for the parameters. The final combination of selected parameters is an approximated mean of several parameter identification results. Approximated inertial parameter values are depicted in table 5.2.

**Table 5.2: Approximated lumped rotational inertia parameters of the drivetrain model.**

| Symbol | Approximated Rotational Inertia ($kgm^2$) |
|---|---|
| $I_{mot}$ | 0.15 |
| $I_{gen}$ | 0.015 |
| $I_{eng}$ | 0.14 |

## 5.2 Neural Network Training

NN training required data to be pre-processed and compiled into a specific format. For each NN, experimental measurements were organized into a matrix of experimental inputs with a width equal to the number of inputs, $n_x$, and a length equal to the number of data samples, $n_s$. A corresponding matrix of experimental outputs was organized to have a width equal to the number of outputs, $n_y$, and a length equal to $n_s$.

NN training was performed using the MATLAB Neural Net Fitting and Neural Net Pattern Recognition toolboxes. At the start of NN training, these toolboxes randomly divide the experimental samples into three sets: training data, validation data, and testing data. Training data is used by the NN training algorithm to iteratively optimize weight and offset terms $W1$, $W2$, $B1$, and $B2$ such that the error between the mapping of NN outputs and training data outputs is minimized. At the end of each iteration of optimization the NN performance is checked against the validation data outputs. When the validation error stops improving, the optimization procedure halts. Once optimization is completed, the NN performance is checked against the testing data outputs. The testing data is not used at any point in the optimization procedure, so it is a fully independent measure of the network's fit. The default ratios of 70% training data, 15% validation data, and 15% testing data were used for all NNs.

The overall quality of fit for each double layer perceptron NN was assessed by observing the trained network's linear regression for each combined data set. The fit of each regression is represented by the correlation coefficient R, which will always have a value between 0 and 1. A perfect NN fit is

represented by an R-value of 1 for the combined data set. Similarly, the quality of fit of each classifier NN was assessed by observing the network's confusion matrix compared to experimental data. The confusion matrix indicates how frequently a classifier NN makes a misclassification, and expresses performance as a percent accuracy (or percent confusion). A confusion matrix that depicts 100% accuracy (0% confusion) means that the network correctly classified all experimental data samples. In general, as the selected number of hidden neurons, $n_n$, increases, the training algorithm will identify NNs with higher R-value or percent accuracy on the combined data set. This is because increasing $n_n$ increases the DOFs in the model by increasing the total number of terms in $W1$, $W2$, and $B1$. However, increasing $n_n$ also increases the likelihood that the NN will overfit the data. An overfitted NN is a poor generalization of the actual function it is trying to model, and will likely perform poorly for a set of inputs that are not contained in the training dataset. Overfitting was identified by comparing the NN's training and validation performance with its testing performance. When the testing performance is significantly worse, then the NN is likely overfitted to the training data. An accepted rule-of-thumb is that the number of neurons in the hidden layer should be between the number of network inputs and the number of network outputs [30]. Details of training and results for each NN are provided below.

### 5.2.1 Supervisory Torque Controller

For identification of the supervisory torque controller NN model, represented by Equation 3.1, the supervisory controller NN model was trained using the Levenberg-Marquardt algorithm. The best training performance without overfitting the data was observed for a network with two hidden neurons. For the case where the transient behavior has decayed ($T_{dsd_k} = T_{dsd_{k-1}}$) the supervisory torque controller NN model is depicted in Figure 5.2.

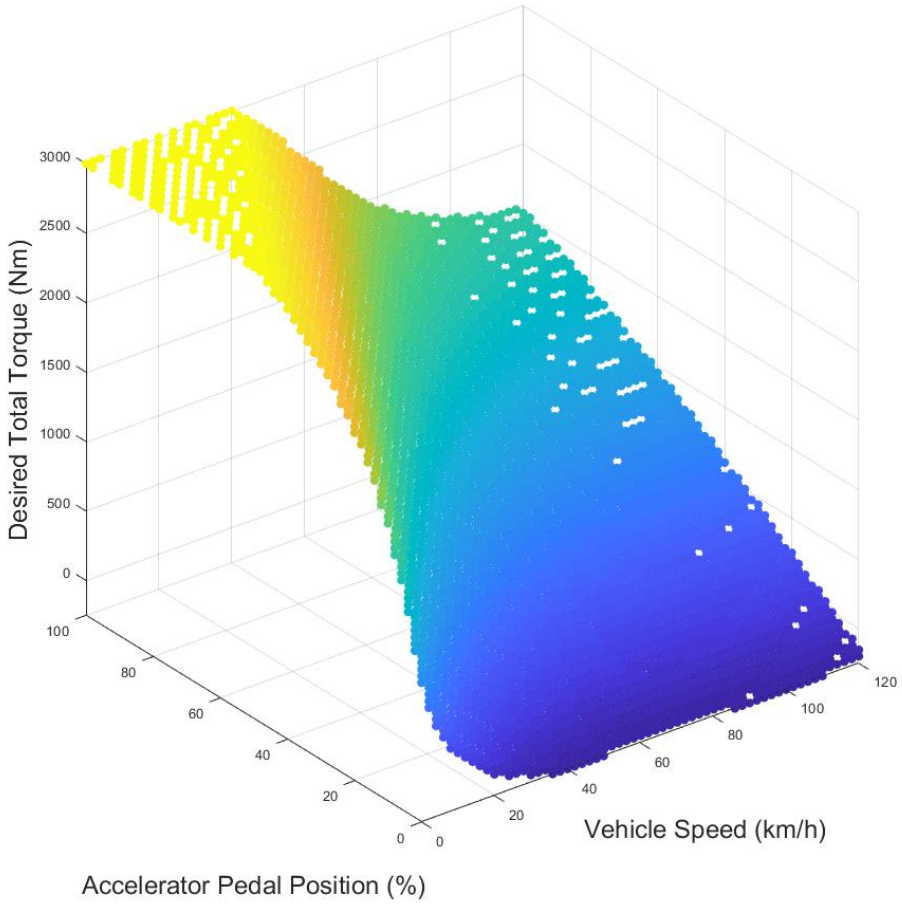


**Figure 5.2: Equilibrium points of the supervisory torque controller NN.**

The combined data R-value of the supervisory controller model is 0.999, which indicates an excellent fit to the data. Refer to Appendix B for the full regression plot.

### 5.2.2 TCM Mode Selection Mapping

The two classifier NNs used for the TCM mode selection map were trained using the scaled conjugate gradient backpropagation algorithm. The engine-start mode selection model, represented by Equation 3.2, showed the best performance without overfitting the data when three hidden neurons were used. The combined data confusion plot for the engine-start mode selection model is depicted in Figure 5.3, where class 1 is EV mode and class 2 is engine cranking mode. Boxes along the diagonal, which are

colored green, indicate instances where classification is correct, while red boxes indicate instances where classification is incorrect. Each light grey box summarize the percent accuracy and percent confusion of all data points in its respective row or column. Green text indicates percent accuracy, and red text indicates percent confusion. The percentages in the dark grey box in the bottom right-hand corner of the matrix give the accuracy and confusion for the entire dataset.


**Figure 5.3: TCM engine-start mode selection model confusion matrix.**

As depicted in Figure 5.3, the model correctly predicted whether or not the engine will start at the next time step 98.6% of the time. The largest source of error in the model is that in 4% of samples, it falsely predicts EV mode when experimental data indicates engine cranking mode.

The engine-on mode selection model, represented by Equation 3.3, showed best performance without overfitting the data when five hidden neurons were used. The combined data confusion plot for the engine-on mode selection model is depicted in Figure 5.4, where class 1 is EV mode, class 2 is engine cranking mode, and class 3 is power-split mode.

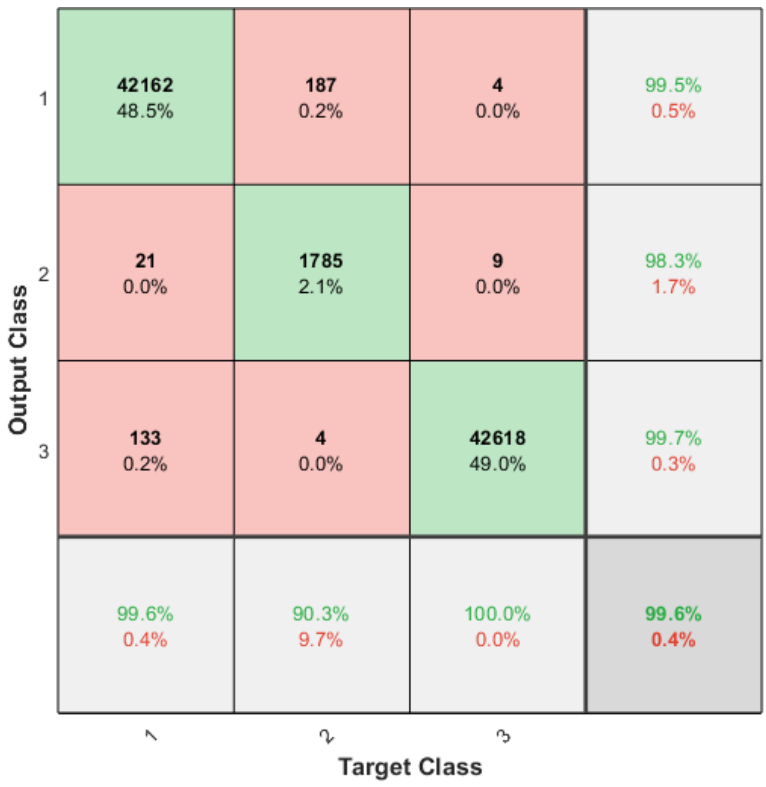


**Figure 5.4: TCM engine-on mode selection model confusion matrix.**

As depicted in Figure 5.4, when the engine shaft is unlocked, the model correctly identified the operating mode at the next time step in 99.6% of samples. This indicates an excellent fit to the data. The only significant source of error in the map is that the model falsely classified 9.7% of experimental engine cranking mode classifications as EV mode. Most of these errors occurred when the powertrain was in engine cranking mode at the previous time step. This error does not factor into model performance because this switching behavior would violate powertrain mode constraint 4 (see Section 3.2.2).

### 5.2.3 Output Torque Model

The double layer perceptron NN used for the output torque model, represented by Equation 3.4, was identified using the Levenberg-Marquardt algorithm. The best neural network performance without

overfitting the data occurred when five hidden neurons were used. The identified NN has an R-value of 0.979 compared to the combined experimental data. Refer to Appendix B for the full regression plot. Since the NN does not model transient behavior in the power sources, the accuracy of this model was limited.

### 5.2.4 Braking Torque Model

The double layer perceptron NN used for the front axle braking model, represented by Equation 3.14, was identified using the Levenberg-Marquardt algorithm. The best neural network performance without overfitting the data occurred when two hidden neurons were used. The identified NN has an R-value of 0.976 compared to the combined experimental data. Refer to Appendix B for the full regression plot. The front axle braking model does not consider transient behavior, which may have affected the accuracy of the model.

Braking torque was not available for measurement at the rear right wheel, so the rear axle braking torque used for model identification was assumed as double the torque measured at the rear left wheel. The classifier NN used to model the rear brake engagement, represented by Equation 3.15, was identified using the scaled conjugate gradient backpropagation method. The combined data confusion plot for the rear brake engagement model is represented by Figure 5.5.

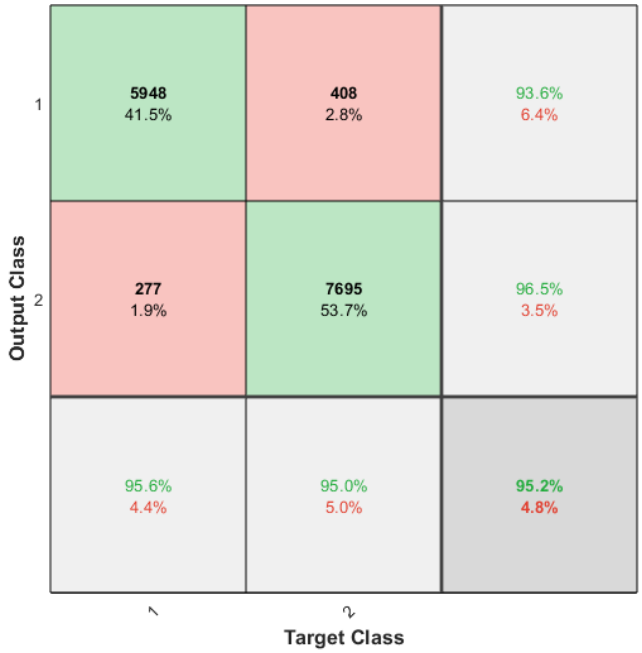


**Figure 5.5: Rear brake switch model confusion matrix.**

As depicted in Figure 5.5, the rear brake switch model correctly guesses whether or not the rear brakes will engage in 95.2% of the experimental data. Since experimental torque was only available at one rear wheel, it is possible that the rear brake switching behavior responds to unmodeled vehicle lateral dynamics effects. The rear brake switch model also does not consider transient brake behavior. Both of these factors may have influenced the accuracy of the model.

The double layer perceptron NN used for the rear axle braking model, represented by Equation 3.16, was identified using the Levenberg-Marquardt algorithm. The best neural network performance without overfitting the data occurred when two hidden neurons were used. The identified NN has an R-value of 0.920 compared to the combined experimental data. Refer to Appendix B for the full regression plot. The same factors that affected the rear brake switching model likely influenced the accuracy of the rear axle braking model.

The combined braking torque map, which combines the front and rear axle braking torque models, is depicted in Figure 5.6.

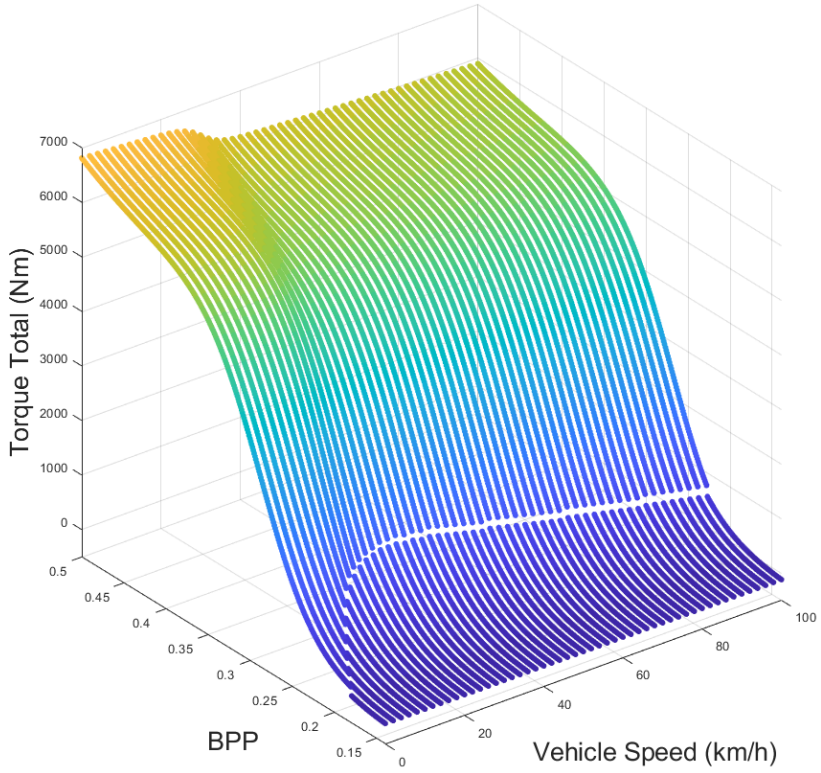


**Figure 5.6: Braking map of the combined front and rear axle torques.**

The discontinuity of the surface, indicated by the white line depicted in Figure 5.6, shows the boundary of rear braking torque engagement. The accuracy of the complete braking model was tested by comparing the output of the combined map, $\tau_{FBr} + \tau_{TBr}$, to the combined experimentally measured torques, $\tau_{FRwh} + \tau_{FLwh} + 2\tau_{RLwh}$. Figure 5.7 depicts the error histogram of total brake torque map.

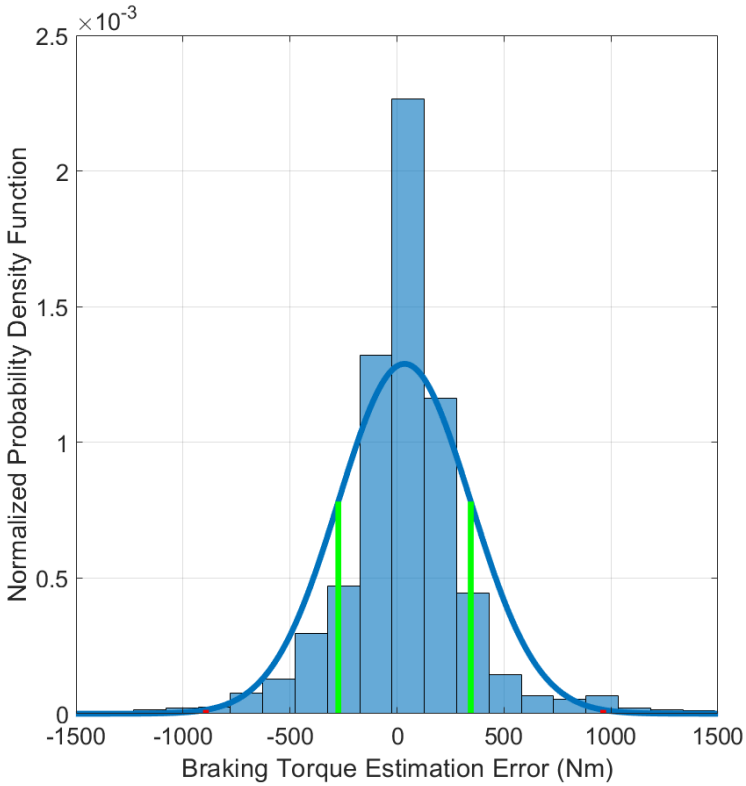


**Figure 5.7: Error Histogram of the total braking torque model.**

The mean error of the total torque model is $\mu = 35.7Nm$, and the standard deviation is $\sigma = 309Nm$. The boundaries for one standard deviation and three standard deviations are depicted in Figure 5.7 as the green and red bars, respectively. Relative to the total available braking torque, which exceeds $6000Nm$, this error in the model is approximately 5%.

Some restrictions of the braking torque model are worth noting. Due to the small size of the test track, limited braking maneuver data was available at speeds greater than 90 km/h. In addition, values of $BPP$ greater than 0.37 generally resulted in the vehicle's tire entering the nonlinear region of high longitudinal slip, causing considerable torque oscillations in the experimental data. For both of these regions, the accuracy of the braking model is uncertain.

## 5.3 Full Model Validation

The full high-fidelity vehicle model, which includes the powertrain, braking, and vehicle dynamics models, was validated using an experimental driving dataset. The selected dataset was used as training data for the braking model map, but otherwise it was unused during parameter identification. Experimental measurements of vehicle inputs $APP$, $BPP$, and steer angle were inputted to the high-fidelity model, and open-loop simulation output states $v_x$ and $\omega_{eng}$ were compared to experimental measurements. Figures 5.8, 5.9, and 5.10 depict some snapshots of full model validation.

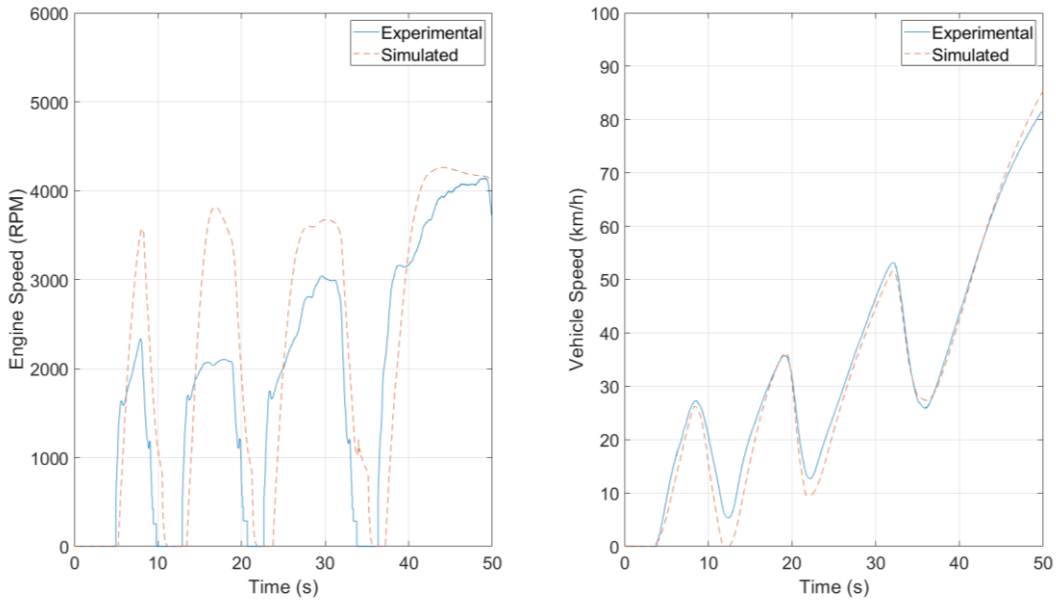


**Figure 5.8: Snapshot 1 of Full Vehicle Model Validation.**

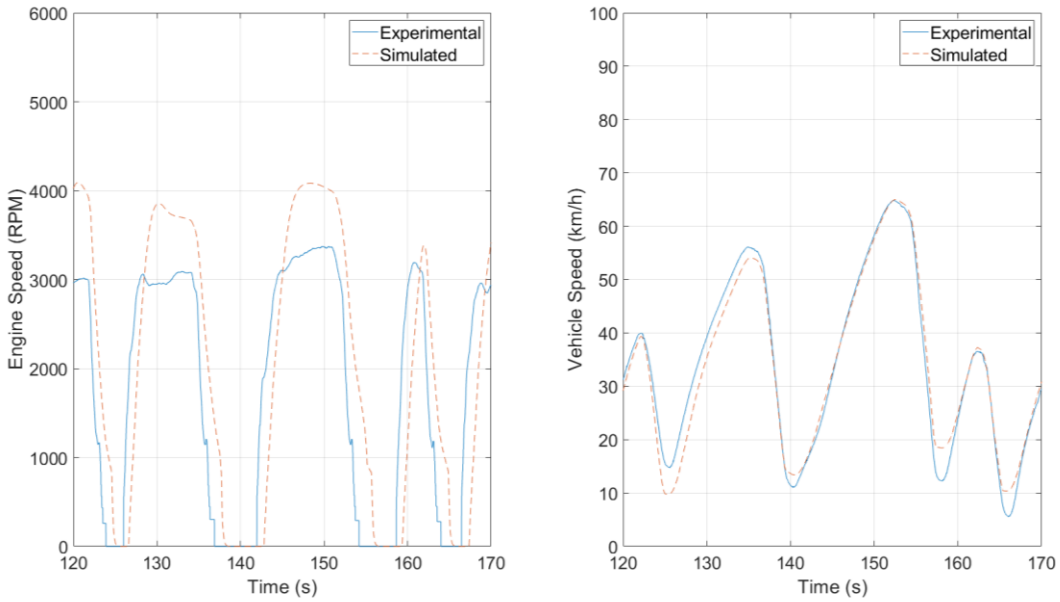


**Figure 5.9: Snapshot 2 of Full Vehicle Model Validation.**

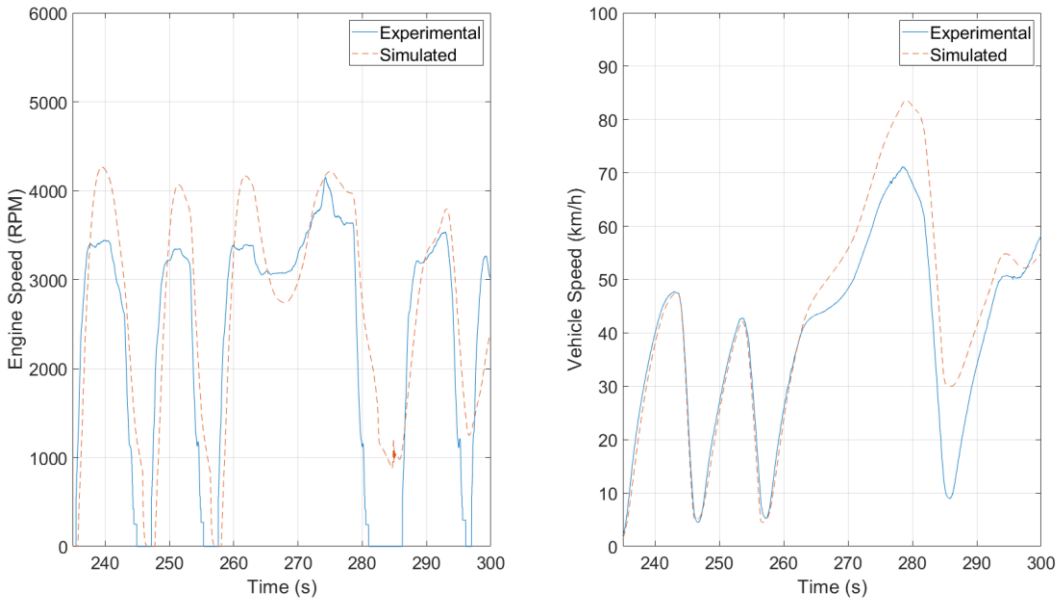


**Figure 5.10: Snapshot 3 of Full Vehicle Model Validation.**

Observation of the engine speed plots in Figures 5.8 to 5.10 show that the model's powertrain operating mode tracks the general trend of experimental data well. The model tends to initiate engine cranking at the same times as the actual vehicle, but during braking maneuvers the model will switch back to EV mode (engine speed returns to zero) more slowly than the actual vehicle. This effect is due to the virtual drivetrain clutch component described in Section 3.4. However, the average engine speeds predicted by the model are higher than the speeds observed in experimental data. It is possible that some unobserved powertrain controller dynamics serve to regulate engine speed. Despite these engine speed tracking errors in the powertrain model, the output vehicle speed tracking is not affected.

Qualitative assessment of Figures 5.8 to 5.10 indicates that the full vehicle model tracks experimental data well. The model's vehicle speed tracks the experimental data within 3km/h for the majority of simulation time. The high-fidelity model's speed tracking performance has also proven to be robust. Even after the model's speed diverges from experimental data, such as at 10 seconds for Figure 5.8 or at 262 seconds for Figure 5.10, the model tends to converge again with experimental data over time. This indicates that at some operating points there is mismatch between the high-fidelity model and the actual vehicle performance, but the average speed tracking performance over the vehicle's entire operating range is accurate. Over the entire open-loop simulation, which ran continuously for 580 seconds, The root-mean-square (RMS) error of the simulated vehicle velocity was 7.2 km/h.

There were several unmeasured disturbances acting on the actual car that were not used in the high-fidelity vehicle model simulation. Time-dependent factors such as road slope, road surface, and wind could have caused the model to diverge from experimental data. In addition, some lateral dynamics effects on the powertrain and braking dynamics have yet to be modelled, so divergence of the model from experimental data occurred at high speeds and high steer angles.

# Chapter 6
# MPC for Longitudinal Vehicle Dynamics

## 6.1 Control-Oriented Model

To design an MPC for longitudinal dynamics of the Moose, a control-oriented model of combined powertrain, braking, and vehicle dynamics was required. Using the high-fidelity model described in Chapter 3 as a baseline, the control-oriented model needed to be simplified to allow real-time controller implementation. The control-oriented model equations needed to be smooth and differentiable in their entire operating space, so that linearization and linear MPC techniques could be applied.

Converting the supervisory controller and TCM models to a control-oriented form was a challenge during controller development. Using Maple, the supervisory controller model was converted to an optimized symbolic function that estimates the desired torque $T^*_{dsd_k}$, represented by Equation 6.1:

$$T^*_{dsd_k} = f_{T^*_{dsd}}(APP, v_x, T^*_{dsd_{k-1}}) \tag{6.1}$$

where * is used to denote any variables that are predicted by the model instead of measured. By definition, the NN function is guaranteed to be smooth and differentiable. The time-delayed estimated desired torque term $T^*_{dsd_k}$ was introduced to the state space equations as an augmented vehicle state.

Attempting to predict the switching of powertrain mode over the MPC prediction horizon is logistically challenging since the drivetrain equations of motion (EOMs) change with operating mode. As such, the control-oriented model was designed to make a rule-based estimate of powertrain mode based only on the previous time step's powertrain mode, previous estimated desired torque $T^*_{dsd_k}$, current speed $v_{x_k}$, and current and previous engine speeds $\omega_{eng_k}$ and $\omega_{eng_{k-1}}$. The rule-based behavior was modelled from experimental powertrain performance. The ruled-based system has a slower mode switching response than the high-fidelity NN models, but it will be more robust to any unmodelled behavior of the powertrain system. The torque selection model, and its interaction with the drivetrain dynamics, was too complex for direct implementation into the control-oriented model. Adding the NN

directly into the control-oriented model caused the resulting symbolic linearization to be too large for MATLAB implementation. Instead the torque selection NN model is numerically linearized at each time step to calculate the partial derivatives $\frac{\partial \tau^*_{eng}}{\partial T^*_{dsd}}$, $\frac{\partial \tau^*_{gen}}{\partial T^*_{dsd}}$, and $\frac{\partial \tau^*_{mot}}{\partial T^*_{dsd}}$. The partial derivatives are determined by the finite difference method. At each time step the approximated linearized equations for torque at each power source are represented by Equations 6.2, 6.3, and 6.4:

$$\tau^*_{eng} \cong \tau^*_{eng\,k-1} + \frac{\partial \tau^*_{eng}}{\partial T_{dsd}}\left(T^*_{dsd_k} - T^*_{dsd\,k-1}\right) \tag{6.2}$$

$$\tau^*_{gen} \cong \tau^*_{gen\,k-1} + \frac{\partial \tau^*_{gen}}{\partial T_{dsd}}\left(T^*_{dsd_k} - T^*_{dsd\,k-1}\right) \tag{6.3}$$

$$\tau^*_{mot} \cong \tau^*_{mot\,k-1} + \frac{\partial \tau^*_{mot}}{\partial T_{dsd}}\left(T^*_{dsd_k} - T^*_{dsd\,k-1}\right) \tag{6.4}$$

where $T^*_{dsd_k}$ is calculated by Equation 6.1. At each time step $\tau^*_{eng\,k-1}$, $\tau^*_{gen\,k-1}$, and $\tau^*_{mot\,k-1}$ are all calculated directly by the torque selection NN using states from the previous time step.

The drivetrain dynamic equations introduced in Section 3.2.3 were simplified to ignore gear meshing efficiency losses, and the difference in speed between the front left and front right wheels resulting from the drivetrain differential is disregarded. The drivetrain output speed and torque used in the control-oriented model are $\omega_{ds}$ and $\tau_{ds}$, respectively. The driveshaft is assumed to pass torque from the front axle, through the two front wheels, and onto the road.

The NN front and rear braking torque models were replaced by a simpler nonlinear map of total braking torque that is a function of only $BPP$. Based on observation of experimental braking torque data, a generic asymmetrical sigmoid function was selected to map braking in the control-oriented model. The parameters of the function were identified in the MATLAB Curve Fitting Tool using the trust-region algorithm. Equation 6.5 represents the simplified braking map used in the control-oriented model:

$$\tau^*_{Br} = f_{Br}(BPP) = \frac{a}{\left(b\left(1+e^{BPP-d}\right)\right)^{\frac{1}{v}}}\,Nm \tag{6.5}$$

where $a = 6261, b = 25.07, d = 0.2522,$ and $v = 0.4388$ are all unitless constants, which were determined from parameter identification. The equation fit has an $R^2$ value of 0.9478, which indicates a good fit for the simplified model. Figure 6.1 depicts the fit of curve to experimental data.

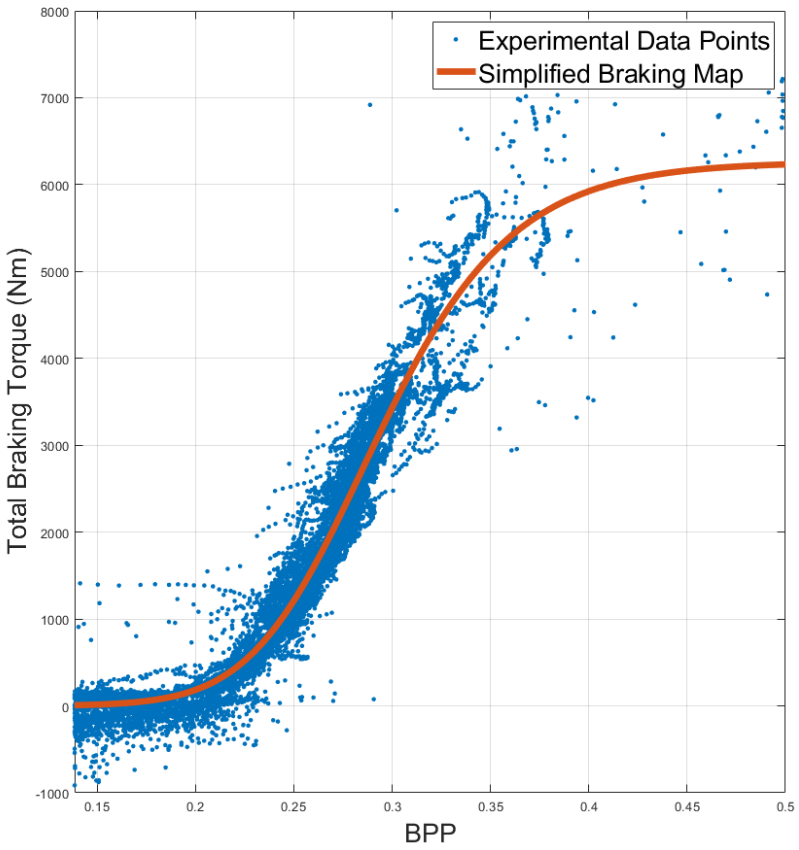


**Figure 6.1: Fit of the control-oriented Braking Torque Map to the experimental data.**

Rather than handle the dynamic changes in torque distribution between the front and rear axle, the control-oriented model assumes a static front-rear braking bias of 3:1, which is the approximate bias ratio when the rear brakes are engaged; therefore, $\tau_{FBr}^* = 0.75\tau_{Br}^*$ and $\tau_{RBr}^* = 0.25\tau_{Br}^*$. The assumptions of the simplified braking torque model limit its accuracy in certain ranges of operating conditions. Notably, as depicted in Figure 6.1, high frequency tire slip oscillations occurring when $BPP$ is greater than 0.35 cause the output wheel torque to be unpredictable. In addition, assuming a static

front-rear braking bias may result in degraded model performance for conditions where front and rear wheel slip are significantly different, such as when individual wheels hit ice.

The discontinuity between the powertrain and braking models, which was handled in the high-fidelity model using a clutch component, cannot be added to the control-oriented model. The discontinuity between acceleration and braking was handled by introducing an analytic approximation of the Heaviside switching function. First, since $APP$ and $BPP$ are never applied simultaneously, the control-oriented model was simplified to include only one control input, the generalized pedal position ($GPP$). The $GPP$ input has a range of -100 to 100, and it maps to $APP$ and $BPP$ by Equations 6.6 and 6.7, respectively:

$$APP = GPP \tag{6.6}$$

$$BPP = -0.004615GPP + 0.1385 \tag{6.7}$$

The approximated Heaviside function is represented by Equation 6.8:

$$H = \frac{1}{1+e^{-5GPP}} \tag{6.8}$$

$H$ is applied to individual terms of the drivetrain dynamics equations such that during braking, the drivetrain inertias are fully decoupled from the motor speed in the control-oriented model.

Based on the control-oriented model simplifications described above, and rearranging the drivetrain dynamic equations described in Sections 3.2.3 and 3.4, the drivetrain dynamic equations of the control-oriented model were formulated. Equation 6.9 represents the control-oriented model's simplified nonlinear dynamics of the drivetrain during EV mode (engine shaft is locked):

$$\begin{bmatrix} \left(I_{mot} + \left(\frac{R_1}{R_2\rho}\right)^2 I_{gen}\right)\dot{\omega}_{mot} \\ I_{eng}\dot{\omega}_{eng} \end{bmatrix} = \begin{bmatrix} \left(\tau^*_{mot} - \frac{R_1}{R_2\rho}\tau^*_{gen}\right)H - \frac{1}{R_f R_2}\tau_{ds} - \tau^*_{FB} \\ 0 \end{bmatrix} \tag{6.9}$$

Similarly the control-oriented model's simplified nonlinear dynamics of the drivetrain during engine-cranking or power-split mode (engine shaft unlocked) is represented by Equation 6.10:

$$\begin{bmatrix} \left(I_{mot} + \left(\frac{R_1}{R_2\rho}\right)^2 I_{gen}\right)\dot{\omega}_{mot} - \left(\frac{R_f R_1(1+\rho)}{R_f R_2 \rho^2}\right)\dot{\omega}_{eng} \\ \left(I_{eng} + \left(\frac{1+\rho}{\rho}\right)^2 I_{gen}\right)\dot{\omega}_{eng} - \left(\frac{R_f R_1(1+\rho)}{R_f R_2 \rho^2}\right)\dot{\omega}_{eng} \end{bmatrix} = \begin{bmatrix} \left(\tau^*_{mot} - \frac{R_1}{R_2\rho}\tau^*_{gen}\right)H - \frac{1}{R_f R_2}\tau_{ds} - \tau^*_{FB} \\ \tau^*_{eng} + \left(\left(\frac{1+\rho}{\rho}\right)\tau^*_{gen}\right)H \end{bmatrix} \quad (6.10)$$

Note that when the engine is unlocked, the dynamics of the motor and engine are coupled. Equations 6.9 and 6.10 were both obtained by replacing $\tau_s$ in Equations 3.12 and 3.13b with a rearranged form of equation 3.11, and combining terms.

A simple longitudinal dynamics vehicle model integrates with the drivetrain in the control-oriented model. Figure 6.2 depicts a diagram of the vehicle model.

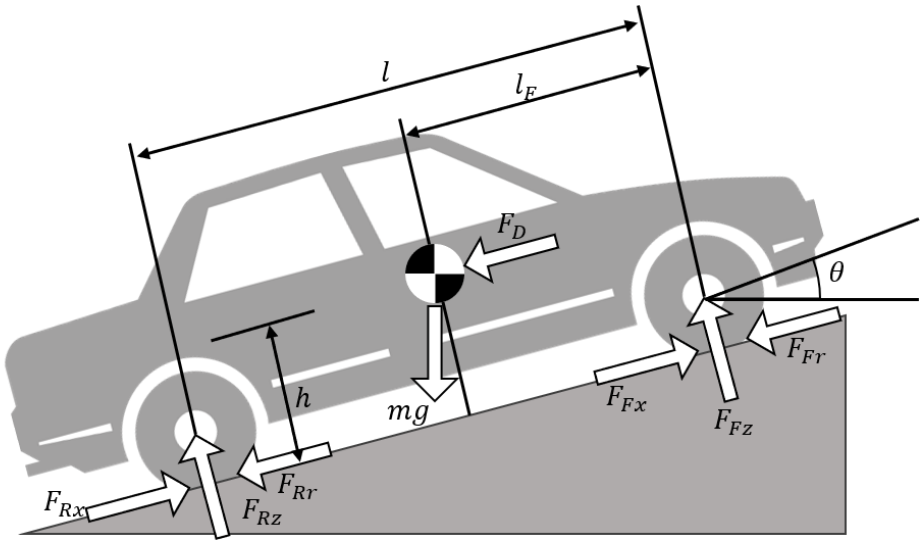


**Figure 6.2: Simplified longitudinal dynamics vehicle model [31].**

The acceleration of the vehicle is determined by the longitudinal tire forces, $F_{Fx}$ and $F_{Rx}$, road slope force, $mgsin\theta$, force of drag, $F_D$, and tire rolling resistance forces, $F_{Fr}$ and $F_{Rr}$. The small angle approximations, $sin\theta \approx \theta$ and $cos\theta \approx 1$, were applied to the road slope force and rolling resistance terms. The model assumes that on each axle the left and right wheels rotate at the same velocity. The EOM for longitudinal motion of the vehicle is represented by Equation 6.11:

$$m\dot{v}_x = F_{Fx} + F_{Rx} - F_D - F_{Fr} - F_{Rr} - mg\theta \quad (6.11)$$

where forces of drag and rolling resistance are represented by Equations 6.11 and 6.12, respectively:

$$F_D = \frac{1}{2} C_d \rho_{air} A_f v_x^2 \tag{6.11}$$

$$F_{Fr} + F_{Rr} = F_r = mgC_r \tag{6.12}$$

Refer to Appendix C for a complete table defining longitudinal vehicle dynamics variables and parameters. The longitudinal tire forces were modelled as functions of longitudinal slip, $\sigma$, and normal force. For simplicity, linear tire models were used. Equations 6.13 and 6.14 depict the equations for linear longitudinal tire forces acting on the front wheels and rear wheels, respectively:

$$F_{Fx} = C_l F_{Fz} \sigma_F \tag{6.13}$$

$$F_{Rx} = C_l F_{Rz} \sigma_R \tag{6.14}$$

where the normal forces on the tires are determined by Equations 6.15 and 6.16:

$$F_{Fz} = mg\left(\frac{l - l_F}{l}\right) - \frac{h}{l}(m\dot{v}_x + F_D + mg\theta) \tag{6.15}$$

$$F_{Rz} = mg\left(\frac{l_F}{l}\right) + \frac{h}{l}(m\dot{v}_x + F_D + mg\theta) \tag{6.16}$$

The longitudinal slip ratios of the front and rear axles are represented as functions of a linear relaxation length equation, defined in Equations 6.17 and 6.18:

$$\dot{\sigma}_F = \frac{\frac{\omega_{mot} r}{R_f R_2} - v_x - v_{op}\sigma_F}{l_{rlx}} \tag{6.17}$$

$$\dot{\sigma}_R = \frac{\omega_R r - v_x - v_{op}\sigma_R}{l_{rlx}} \tag{6.18}$$

where $\sigma_F$ and $\sigma_R$ are the longitudinal slips of the front and rear wheels, respectively. The velocity operating point, $v_{op}$, is usually the vehicle velocity at the point of linearization, but the value must never be small enough to create instability in the control-oriented model. For this system, a minimum

operating speed, $v_{op_{min}} = 1\ m/s$, was found to be sufficient to ensure stability. The relaxation length term, $l_{rlx}$, was not selected based on any known tire parameters. Instead $l_{rlx}$ was used as a means of combating stiffness in the set of equations and stabilizing the response of the control-oriented model. The parameter was tuned manually to maximize controller stability and minimize MPC turnaround time. A non-physical value of $l_{rlx} = 100\ m$ was selected despite the fact that typical tire relaxation lengths are three orders of magnitude smaller. Future work will explore the unusual response of the control-oriented model that makes controller stability highly dependent on the value of $l_{rlx}$.

The time step of the supervisory controller model, $Ts_{NN} = 0.05s$, will not necessarily be the same as the time step of the MPC, $Ts$. Therefore, the augmented vehicle state $T^*_{dsd_{k-1}}$ was converted to continuous form before discretizing the entire system of equations. Equation 6.19 represents the desired torque dynamics in continuous form:

$$\dot{T}^*_{dsd_{k-1}} \approx \frac{T^*_{dsd_k} - T^*_{dsd_{k-1}}}{Ts_{NN}} \tag{6.19}$$

Performing this approximation allows the supervisory torque map to be interpolated at any MPC time step size.

The dynamic equations of the control-oriented model, in continuous form, were rearranged into a nonlinear state space representation. Equation 6.20 depicts the nonlinear state space for EV mode:

$$\begin{bmatrix} \left(m + \frac{\sigma_F mh}{l} - \frac{C_l \sigma_R mh}{l}\right)\dot{v}_x \\ -C_l\sigma_F mh\frac{r}{R_2R_{fl}}\dot{v}_x + \left(H\left(I_{mot} + \left(\frac{R_1}{R_2\rho}\right)^2 I_{gen}\right) + \frac{2I_{wh}}{(R_2R_f)^2}\right)\dot{\omega}_{mot} \\ I_{eng}\dot{\omega}_{eng} \\ C_l\sigma_R mh\frac{r}{l}\dot{v}_x + 2I_{wh}\dot{\omega}_R \\ \dot{\sigma}_F \\ \dot{\sigma}_R \\ \dot{T}^*_{dsd_{k-1}} \end{bmatrix} =$$

$$\begin{bmatrix} C_l\sigma_F\left(\frac{mgl_F}{l} - \frac{h}{l}(F_D + mg\theta)\right) + C_l\sigma_R\left(\frac{mgl_F}{l} + \frac{h}{l}(F_D + mg\theta)\right) - F_D - F_r - mg\theta \\ \left(\tau^*_{mot} - \frac{R_1}{R_2\rho}\tau^*_{gen}\right)H - \frac{0.75}{R_fR_2}\tau^*_{Br} - \frac{C_l\sigma_F r}{R_fR_2}\left(\frac{mg(l-l_F)}{l} - \frac{h}{l}(F_D + mg\theta)\right) \\ 0 \\ -C_l\sigma_R r(\frac{mgl_F}{l} + \frac{h}{l}(F_D + mg\theta)) - 0.25\tau^*_{Br} \\ \frac{\frac{\omega_{mot}r}{R_fR_2} - v_x - v_{op}\sigma_F}{l_{rlx}} \\ \frac{\omega_R r - v_x - v_{op}\sigma_R}{l_{rlx}} \\ \frac{T^*_{dsd_k} - T^*_{dsd_{k-1}}}{Ts_{NN}} \end{bmatrix} \quad (6.20)$$

Equation 6.21 represents the nonlinear state space for engine cranking and power-split modes:

$$\begin{bmatrix} \left(m + \frac{\sigma_F mh}{l} - \frac{C_l \sigma_R mh}{l}\right)\dot{v}_x \\ -C_l\sigma_F mh\frac{r}{R_2R_{fl}}\dot{v}_x + \left(H\left(I_{mot} + \left(\frac{R_1}{R_2\rho}\right)^2 I_{gen}\right) + \frac{2I_{wh}}{\left(R_2R_f\right)^2}\right)\dot{\omega}_{mot} - H\left(\frac{R_1(1+\rho)}{R_2\rho^2}Igen\right)\dot{\omega}_{eng} \\ -H\left(\frac{R_1(1+\rho)}{R_2\rho^2} * Igen\right)\dot{\omega}_{mot} + \left(I_{eng} + \left(\frac{1+\rho}{\rho}\right)^2 Igen\right)\dot{\omega}_{eng} \\ C_l\sigma_R mh\frac{r}{l}\dot{v}_x + 2I_{wh}\dot{\omega}_R \\ \dot{\sigma_F} \\ \dot{\sigma}_R \\ \dot{T}^*_{dsd_{k-1}} \end{bmatrix} =$$

$$\begin{bmatrix} C_l\sigma_F\left(\frac{mg(l-l_F)}{l} - \frac{h}{l}(F_D + mg\theta)\right) + C_l\sigma_R\left(\frac{mgl_F}{l} + \frac{h}{l}(F_D + mg\theta)\right) - F_D - F_r - mg\theta \\ H\left(\tau^*_{mot} - \frac{R_1}{R_2\rho}\tau^*_{gen}\right) - \frac{0.75}{R_fR_2}\tau^*_{Br} - \frac{C_l\sigma_F r}{R_fR_2}\left(\frac{mg(l-l_F)}{l} - \frac{h}{l}(F_D + mg\theta)\right) \\ \tau^*_{eng} + H\left(\frac{1+\rho}{\rho}\tau^*_{gen}\right) \\ -C_l\sigma_R r(\frac{mgl_F}{l} + \frac{h}{l}(F_D + mg\theta)) - 0.25\tau^*_{Br} \\ \frac{\frac{\omega_{mot}r}{R_fR_2} - v_x - v_{op}\sigma_F}{l_{rlx}} \\ \frac{\omega_R r - v_x - v_{op}\sigma_R}{l_{rlx}} \\ \frac{T^*_{dsd_k} - T^*_{dsd_{k-1}}}{Ts_{NN}} \end{bmatrix} \quad (6.21)$$

## 6.2 Linearizing MPC Algorithm

To address the nonlinear dynamics of the control-oriented model, a continuously linearizing MPC algorithm was designed. The proposed method provides the computational speed of linear MPC while allowing the controller to respond to large changes in plant dynamics at different operating points. Figure 6.3 depicts the block diagram of the linearizing MPC.

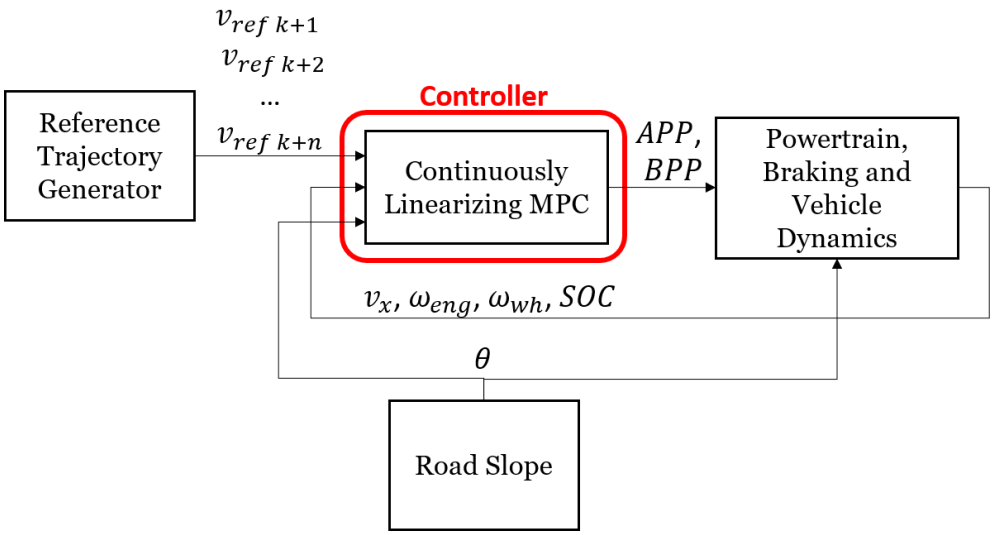


**Figure 6.3: Block diagram of the longitudinal dynamics MPC.**

As depicted in Figure 6.3, at each time step, $k$, the controller receives a reference speed vector, $v_{ref}$, that represents the desired speed of the Moose between time step $k+1$ and time step at the end of the prediction horizon, $k+n$. Current measurements of $v_x$, $\omega_{eng}$, $\omega_{wh}$ (of each wheel), and $SOC$ are used to calculate the states of the vehicle. $\omega_{mot}$ is calculated as the average of the two front wheel speeds, divided by the ratio $R_f R_2$, and $\omega_R$ is calculated as the average of the two rear wheels speeds. As described in Section 6.1, at each time step the simplified TCM model predicts the powertrain operating mode and calculates the numerically linearized terms of Equations 6.2 to 6.4. Road slope, $\theta$, is treated as a measurable disturbance. Future work will investigate integrating a road slope estimator with the controller.

Using Maple offline, Equations 6.20 and 6.21 were symbolically linearized and discretized by the Forward Euler method to determine two forms of the Jacobian matrices, $A$ and $B$, and the state increment, $\Delta X$. By computing these matrices symbolically offline, linearization about a given operating point can be computed online without performing numerical differentiation. At each time step the controller uses the predicted operating mode to select which linearized state space equations to use. The

model is linearized about the previous time step, $k-1$. Equation 6.22 represents the linearized state space equation at controller time step $k$:

$$X_{k+i+1} \cong X_{k-1} + A_{k-1}(X_{k+i} - X_{k-1}) + B_{k-1}(GPP_{k+i} - GPP_{k-1}) + \Delta X_{k-1} \tag{6.22}$$

where $A_{k-1}$, $B_{k-1}$, and $\Delta X_{k-1}$ are computed algebraically from the terms of state vector $X_{k-1}$, which is represented by Equation 6.23:

$$X_{k-1} = \left\{ v_{x_{k-1}},\ \omega_{mot_{k-1}}, \omega_{eng_{k-1}}, \omega_{R_{k-1}}, \sigma_{F_{k-1}}, \sigma_{R_{k-1}}, T^*_{dsd_{k-2}} \right\}^T \tag{6.23}$$

The longitudinal slip operating point terms are calculated by Equations 6.24 and 6.25:

$$\sigma_{F_{k-1}} = \frac{\omega_{mot_{k-1}} r - R_f R_2 v_{x_{k-1}}}{\omega_{mot_{k-1}} r} \tag{6.24}$$

$$\sigma_{R_{k-1}} = \frac{\omega_{R_{k-1}} r - v_x}{\omega_{R_{k-1}} r} \tag{6.25}$$

A quadratic objective function was selected for the MPC to ensure convexity of the optimization problem. The cost function includes the speed tracking error over the prediction horizon. A weighting was assigned to the rate of change of control input, $\Delta GPP$, over the control horizon. The weighting assigned to $\Delta GPP$ was tuned to limit the jerk of the vehicle's response. Equation 6.26 depicts the objective function used for the MPC:

$$J = \sum_{i=k+1}^{k+n} \left\| v_{x_i} - v_{ref\ i} \right\|^2 + \sum_{j=k}^{k+c} R_{\Delta u} \Delta GPP_j^2 \tag{6.26}$$

subject to the constraints:

$$-100 \leq GPP \leq 100$$

$$-50/s \leq \Delta GPP \leq 50/s$$

where $n$ is the prediction horizon, $c$ is the control horizon, and $R_{\Delta u}$ is the relative weighting of $\Delta GPP$. The horizon lengths are constrained such that $c \leq n$. For any prediction step after the end of the control

horizon, $i > k + c$, the control input is set to the control input at the final control horizon step, $GPP_{k+c}$. The MPC was implemented for model in the loop (MIL) simulation of the high-fidelity vehicle model described in Chapter 3. The parameters $n$, $c$, $R_{\Delta u}$, and the controller time step, $Ts$, were tuned using multiple $v_{ref}$ tracking scenarios. The tuned MPC parameters are summarized in Table 6.1.

**Table 6.1: List of tuned MPC parameter values.**

| Symbol | Value | Units |
|---|---|---|
| $n$ | 50 | N/A |
| $c$ | 25 | N/A |
| $R_{\Delta u}$ | 0.15 | N/A |
| $Ts$ | 0.02 | s |

An additional modification was made to the controller to ensure its stability at low speeds. The definition of longitudinal slip used in the control-oriented model creates an instability at low speeds. At low speeds the denominator terms of Equations 6.24 and 6.25 will approach zero, which will cause the magnitudes of longitudinal slip to grow very large. MPC performance will become unpredictable as it attempts to prevent excessive wheel slip. To prevent instability, at velocities below 14km/h the controller switches from an MPC to a low gain PI algorithm for low velocity tracking. A ±4 km/h switching dead band was implemented to prevent chatter between the control modes. The PI algorithm was tuned specifically for smooth velocity tracking at low speeds with low gain control action.

As depicted in Figure 6.3, at the output of the controller, $GPP$ is converted to the real controller commands $APP$ and $BPP$ by Equations 6.6 and 6.7, respectively. The two outputs are subjected to saturation limits such that $0\% \leq APP \leq 100\%$ and $0.1385 \leq BPP \leq 0.5$ based on the input range of the Dataspeed drive-by-wire system.

## 6.3 Controller Simulation Results

For the purposes of performance comparison, a PI controller was used as a benchmark for testing the MPC because a similarly designed controller is currently implemented on the Moose. Like the MPC, the PI controller outputs a value for $GPP$ that are converted to $APP$ and $BPP$ values at its output. The PI controller was tuned manually to minimize rise time, maximum overshoot, and settling time, but high-fidelity model nonlinearities affect its performance at different operating points. To track $v_{ref}$ at high speeds, the PI must have high values of proportional and integral gain, which makes the controller sensitive to unmeasured disturbances. In each of the following simulated tests, an unmeasured disturbance was added to the loop as white noise in the measured states. The noise is intended to test the controller's disturbance rejection characteristics.

### 6.3.1 Ramping Velocity Simulation

The first simulated benchmark test is the ramping velocity test. After an initial model settling time of 2 seconds, $v_{ref}$ is set to 0 and ramped up with a constant acceleration of 1m/s$^2$ (3.6km/h/s). Figures 6.4 and 6.5 depict the velocity tracking and velocity error, respectively, of the ramp simulation.

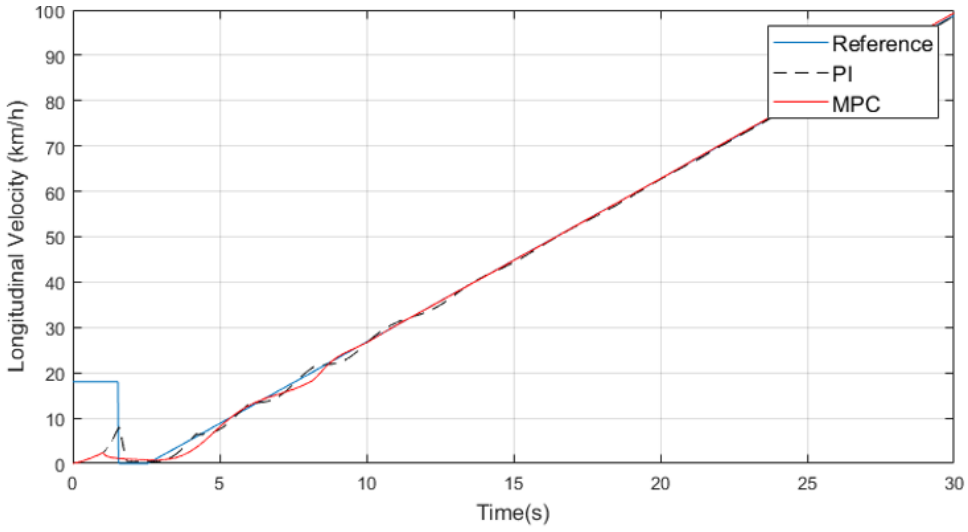


**Figure 6.4: Velocity tracking performance of the ramp simulation.**

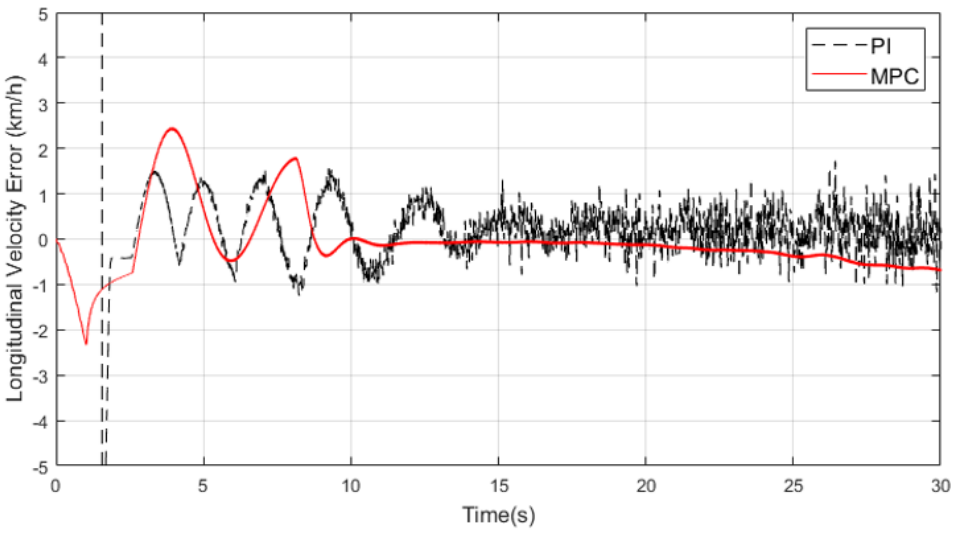


**Figure 6.5: Velocity tracking error for the ramp test.**

Both the MPC and PI controllers track $v_{ref}$ within 2.5 km/h during the entire simulation. At speeds below 18km/h, which is the upper threshold of the switching deadband, the MPC defaults to using its low gain PI controller, and tracking performance is marginally worse than the higher gain benchmark PI controller. This was an intentional design choice to prevent high frequency pedal actuation at low speeds, but the controller may be tuned for other preferred performance characteristics. After 8 seconds, the MPC switches to MPC control mode and smoothly tracks the $v_{ref}$ profile; however the MPC has a small tracking offset error (less than 0.5km/h) at velocities greater than 80km/h. This is likely the result of model mismatch between the control-oriented model and high-fidelity model in the high speed operating range.

By comparison, the high gain PI controller fails to handle the white noise disturbance, so it oscillates with high frequency about $v_{ref}$. The comparatively jerky behavior of the high gain PI controller is clearly displayed in the plots of $APP$ and $BPP$ that are depicted in Figures 6.6 and 6.7, respectively.

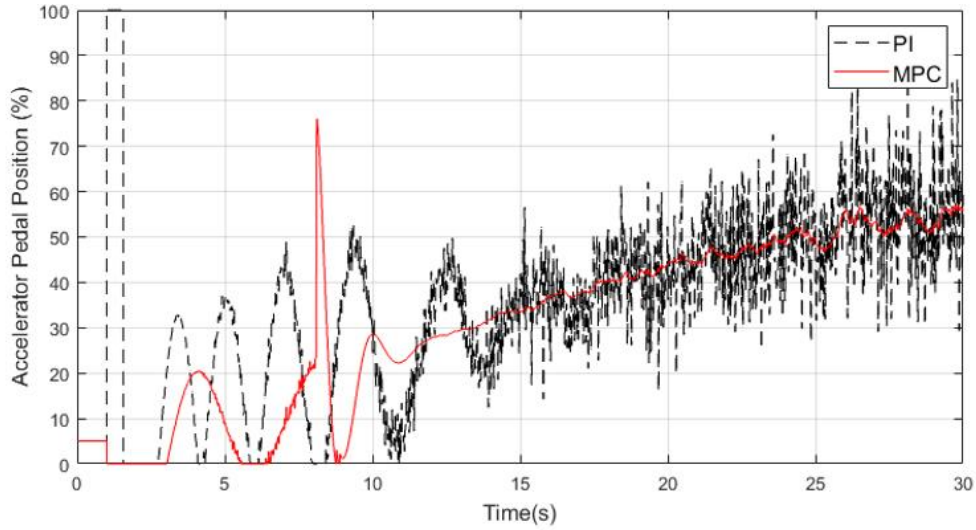


**Figure 6.6: APP control input for the ramp simulation.**

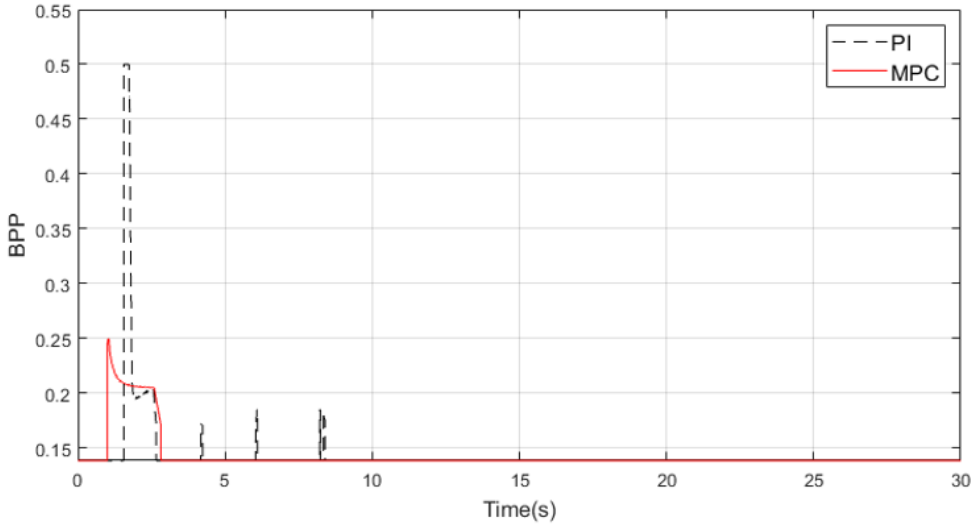


**Figure 6.7: BPP control input for the ramp simulation.**

The MPC controller rejects disturbances well, and shows minimal oscillation of the *APP* control input. It also does not chatter between *APP* and *BPP* control action. The only notable spike In the MPC controller's performance is at 8 seconds, when the controller switches from PI to MPC mode. Future

controller modifications could be used to ensure bumpless transfer during switching from PI to MPC mode. Comparatively, due to its poor disturbance rejection characteristics, the high gain PI controller performance shows large oscillations in the $APP$ control input. At low speeds it occasionally chatters between $APP$ and $BPP$ control action.

### 6.3.2 Multi-Ramp Velocity Simulation

The second simulated benchmark test is the multi-ramp velocity test. In this test $v_{ref}$ is ramped at different rates to several velocity set points. The intention of the simulation is to test the controller's ability to track various accelerations and hold different velocity set points. Figures 6.8 and 6.9 depict the velocity tracking performance and velocity error for the multi-ramp test.

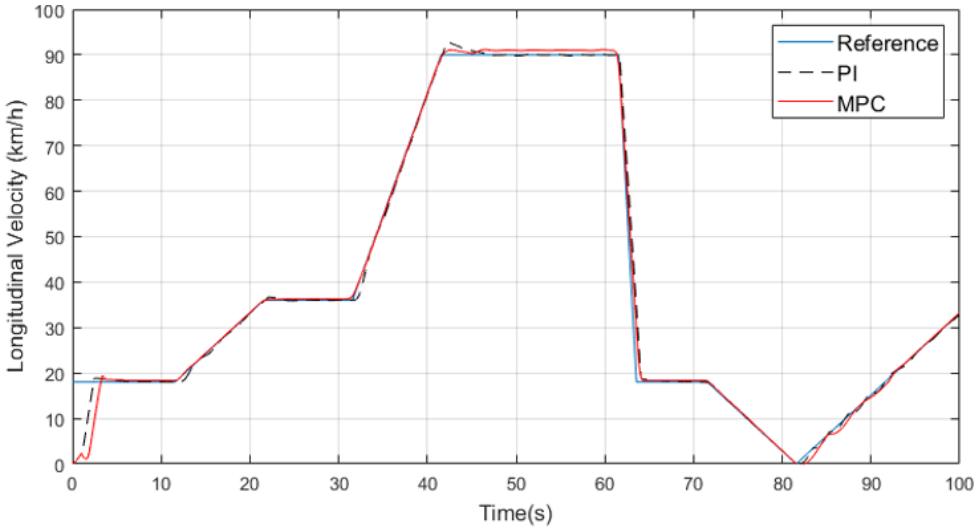


**Figure 6.8: Velocity tracking performance of the multi-ramp simulation.**

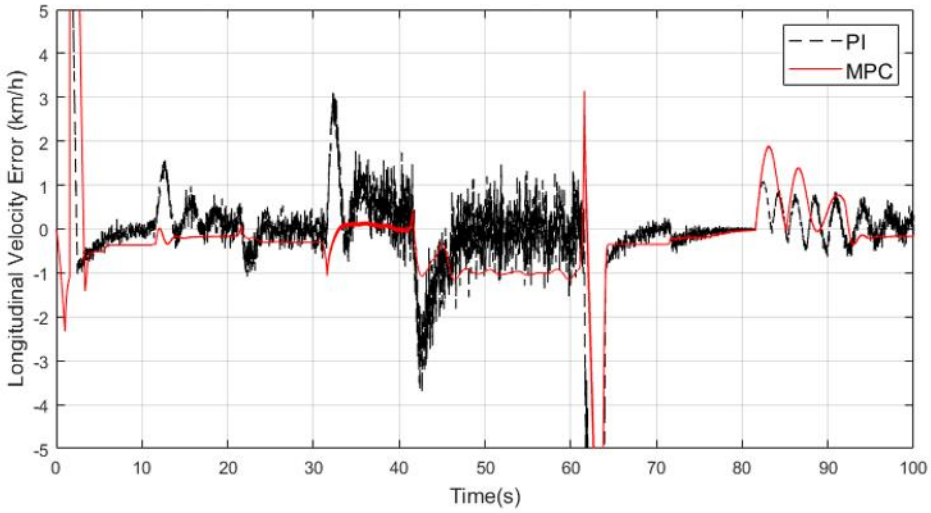


**Figure 6.9: Velocity tracking error for the multi-ramp simulation.**

Between 5 seconds and 75 seconds the MPC controller operates entirely in MPC mode. The tracking performance is smooth, and the predictive nature of the controller allows it to ramp into the hard acceleration and hard braking maneuvers beginning at 32 seconds and 62 seconds, respectively. Like the previous test, the MPC has a small offset tracking error at high speeds due to model mismatch. The MPC only switches back to PI mode for the low speed tracking between 75 and 93 seconds.

Like the previous simulation the benchmark high gain PI controller displays unacceptable levels of oscillation in reference tracking, particularly at high speeds. Due to integral windup, the PI controller is slow to respond to changes in the slope of $v_{ref}$, which results in large tracking errors at 12 seconds, 32 seconds, and 42 seconds. Both the high gain PI and the MPC fail to track the steep negative ramp of $v_{ref}$ since the demanded acceleration is outside the limits of the high-fidelity vehicle model's braking ability.

The $APP$ and $BPP$ control inputs for the multi-ramp simulation are depicted in Figures 6.10 and 6.11.

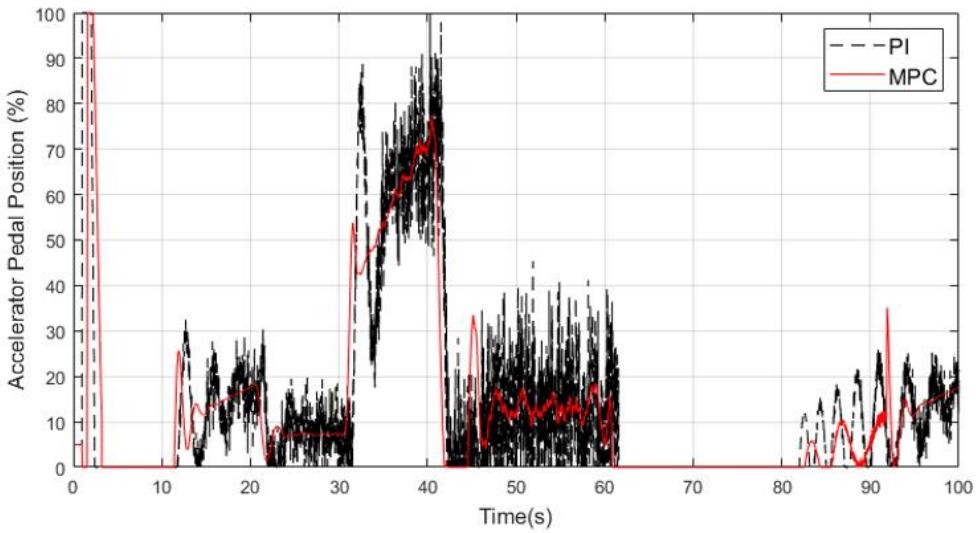


**Figure 6.10: APP control input for the multi-ramp simulation.**

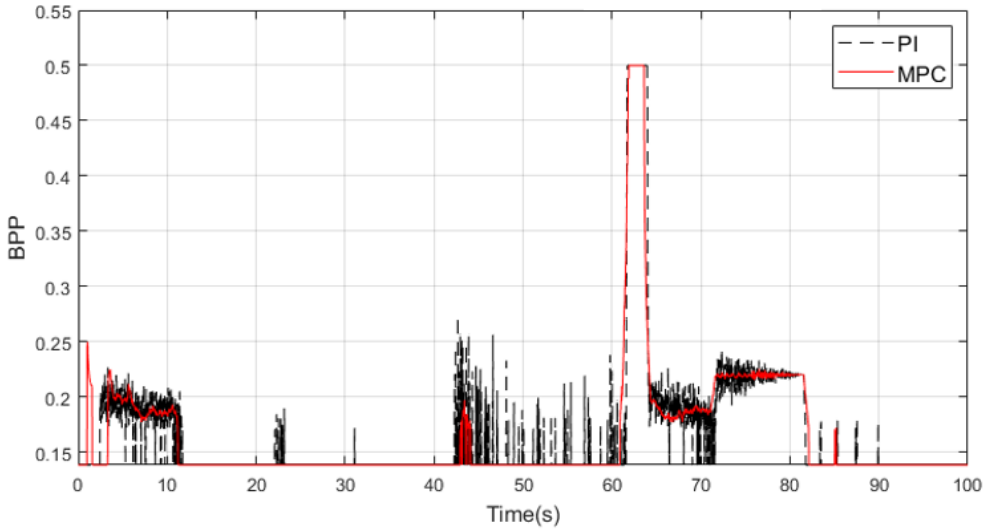


**Figure 6.11: BPP control input for the multi-ramp simulation.**

The MPC control action is much smoother than the high gain PI controller. The high gain PI displays highly jerky behavior and frequently chatters between $APP$ and $BPP$ control action.

### 6.3.3 Sinusoidal Velocity Simulation

The third simulated benchmark test is the sinusoidal velocity test. In this simulation, $v_{ref}$ follows a sinusoidal path through a range of speeds that are typical on city roads with light traffic. Figures 6.12 and 6.13 depict the velocity tracking performance and velocity error for the sinusoidal test.

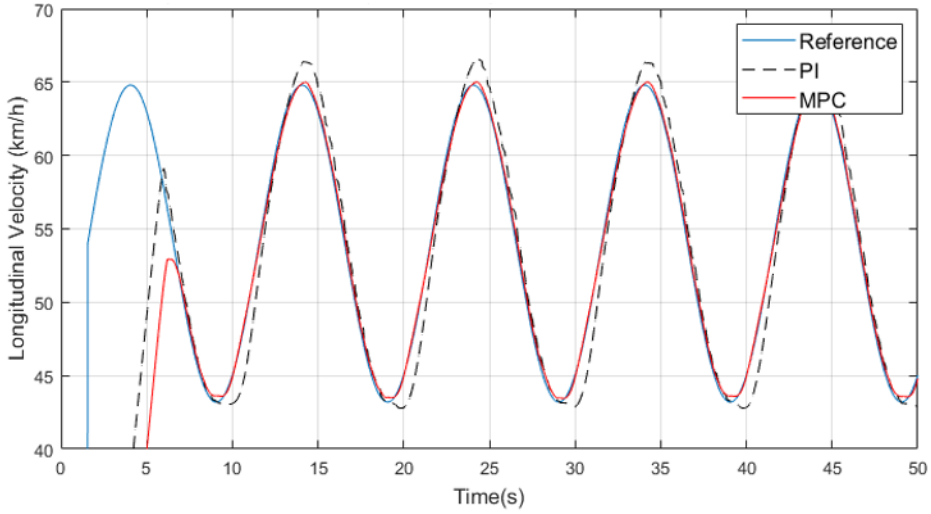


**Figure 6.12: Velocity tracking performance of the sinusoidal simulation.**

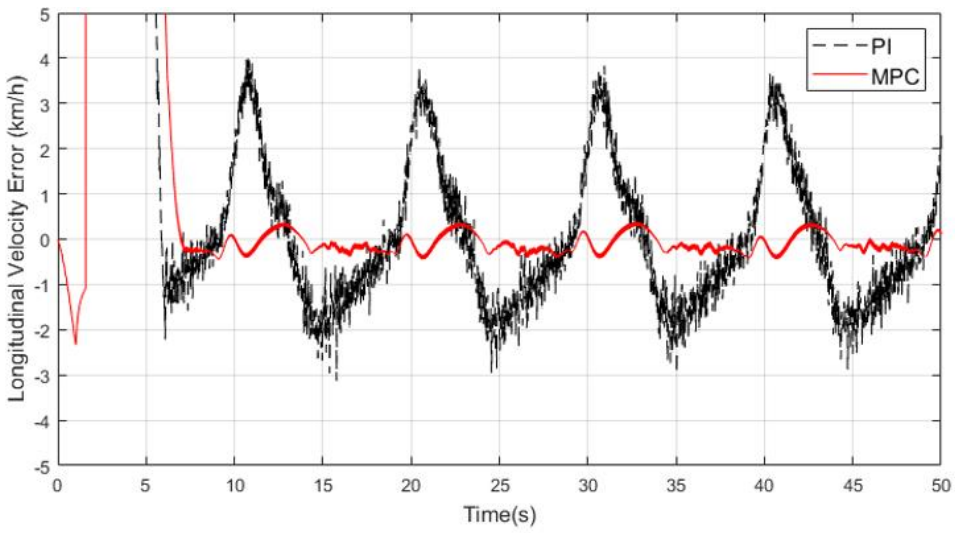


**Figure 6.13: Velocity tracking error for the sinusoidal simulation.**

After the initial 6 seconds required to reach the reference speed, the MPC tracks $v_{ref}$ smoothly for the remainder of the simulation. The MPC's tracking error never exceeds 0.5km/h, and the predictive quality of the controller helps it to follow changes to the slope of $v_{ref}$. Comparatively, the high gain PI controller displays much larger tracking error, and high frequency oscillations persist throughout the simulation. Like the multi-ramp test, integral windup causes the high gain PI controller to diverge from the reference path at the peaks and troughs of the $v_{ref}$ sinusoid.

Figures 6.14 and 6.15 depict the $APP$ and $BPP$ control inputs for the sinusoidal simulation.

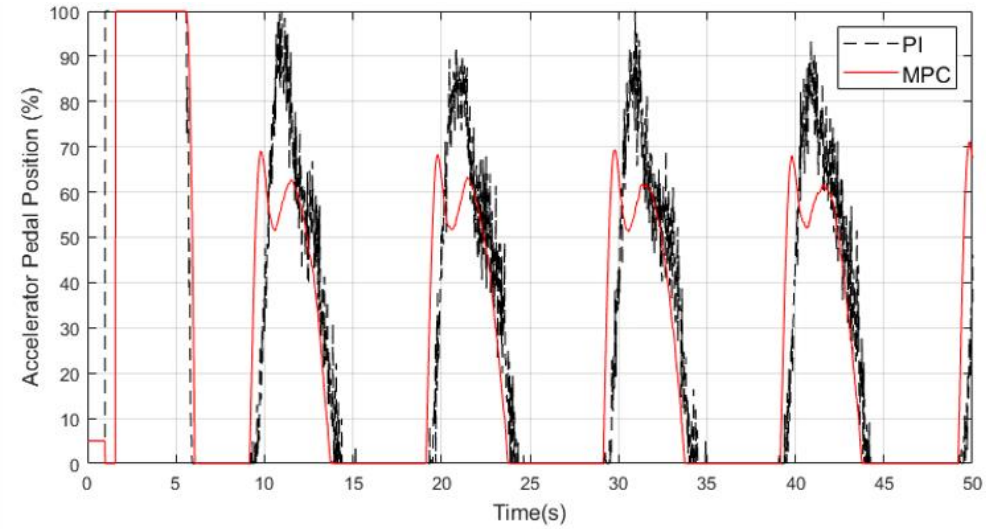


**Figure 6.14: APP control input for the sinusoidal simulation.**

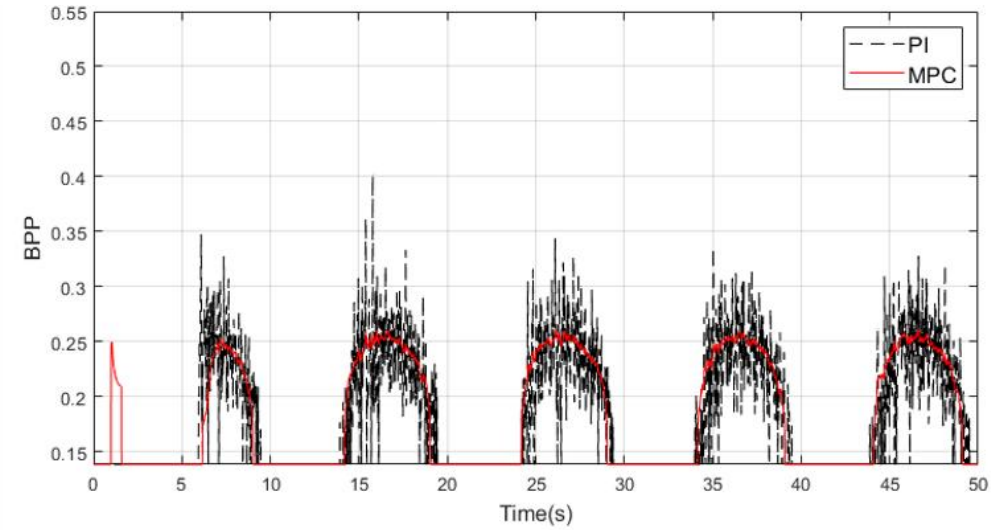


**Figure 6.15: BPP control input for the sinusoidal simulation.**

Like the previous simulations, the MPC control input is smooth and does not chatter between $APP$ and $BPP$. The high gain PI applies more aggressive $APP$ and $BPP$ inputs that oscillate with high frequencies.

### 6.3.4 Simulation Discussion

Comparison of the MPC and the high gain PI controller in all three benchmark simulation scenarios indicates that the MPC has several advantages over classical control. The only way to obtain adequate PI tracking performance at high speeds was to tune the controller to respond aggressively to $v_{ref}$ tracking error. The MPC can achieve similar or superior tracking performance over a broad range of velocities while also applying smoother control inputs.

The high gain PI's oscillating behavior and chatter between $APP$ and $BPP$ implies that classical control is poorly suited to the highly nonlinear behavior of the vehicle, particularly at high speeds. The more conservative PI controller that is currently used by the Moose will only perform satisfactorily for low speed driving, and it is untested at velocities greater than 50km/h. Simulation results indicate that implementation of an MPC will allow the Moose to smoothly track reference velocities up to 100 km/h.

# Chapter 7
# Full Vehicle MPC Testing

## 7.1 MPC Implementation

The Moose's software stack communicates between modules using the Robot Operating System (ROS). ROS is an open-source communication structure that allows various modules and processes in a robotic system to send and receive messages. The modules and processes are each implemented as ROS nodes, and they communicate messages, such as measurements and commands, over ROS topics. ROS provides an excellent environment for integration, logging, and testing distributed computing systems. These functions make it a powerful tool for implementation of complex autonomous systems, such as self-driving cars. To implement the MPC into the Moose's stack, it must be converted to a form that is implementable as a ROS node.

Using the Simulink embedded Coder tool, the full MPC controller block was exported as an embedded real-time target (ERT) model. The model consists of a C code header and source file, as well as several supporting files. For integration into the Moose's software stack, the controller code was restructured as a C++ class. Carlos Wang, a research engineer with the Autonomoose team, added a ROS node wrapper to the MPC controller and integrated it into a custom build of the Moose's stack.

The MPC controller node receives messages containing measurements of wheel speeds, $v_x$ , $\omega_{eng}$ , $SOC$, and the vector $v_{ref}$ over the prediction horizon. The road slope, $\theta$, is assumed to be zero. At the time of testing, the Moose's stack was not capable of extracting the $\omega_{eng}$ or $SOC$ signals from the vehicle's CAN bus. For initial controller testing, $SOC$ was set to a constant value of 70%, and $\omega_{eng}$ was set to 0. The implication of this is that the MPC will always assume the Moose is operating in EV mode. It is likely that this will affect controller performance, particularly at higher speeds. The MPC controller outputs signals for $APP$ and $BPP$.

## 7.2 Vehicle Test Procedure

For controlled proof of concept testing of the MPC, it was implemented in a controlled test scenario at the Region of Waterloo's Emergency Services Training Centre test track. For all MPC tests, the Moose's steering wheel was locked in neutral position, and the vehicle was driven down a straight section of track. Due to the size of the test track, tests were limited to approximately 250m of vehicle travel.

Two of the three benchmark tests, the ramped velocity test and the sinusoidal velocity test, were repeated for vehicle testing. Before the start of each test run, the Moose was driven to the start of the straight length of track. When the supervising test engineer, Carlos Wang, provided the command for autonomous control to begin, the MPC controller began reading the vector $v_{ref}$ from a time-stamped CSV file. Due to the limited length of track, all tests ended after 15 to 25 seconds.

## 7.3 Vehicle Testing Results

### 7.3.1 Ramping Velocity Vehicle Test

The first benchmark vehicle test was the ramping velocity test. Figures 7.1 and 7.2 depict the velocity tracking performance and the velocity error, respectively, for the vehicle ramp test.

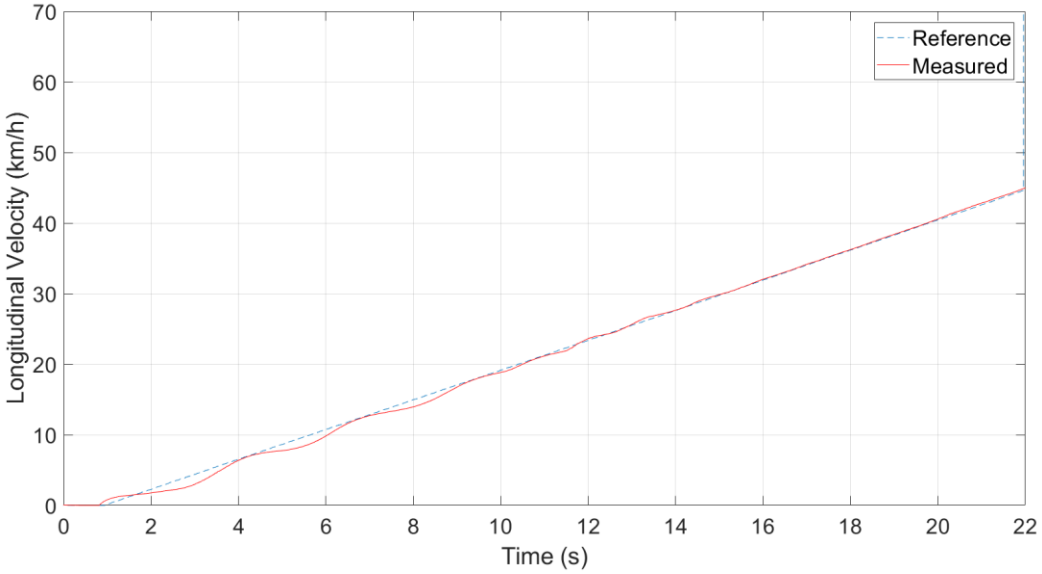


**Figure 7.1: Velocity tracking performance for the vehicle ramp test.**

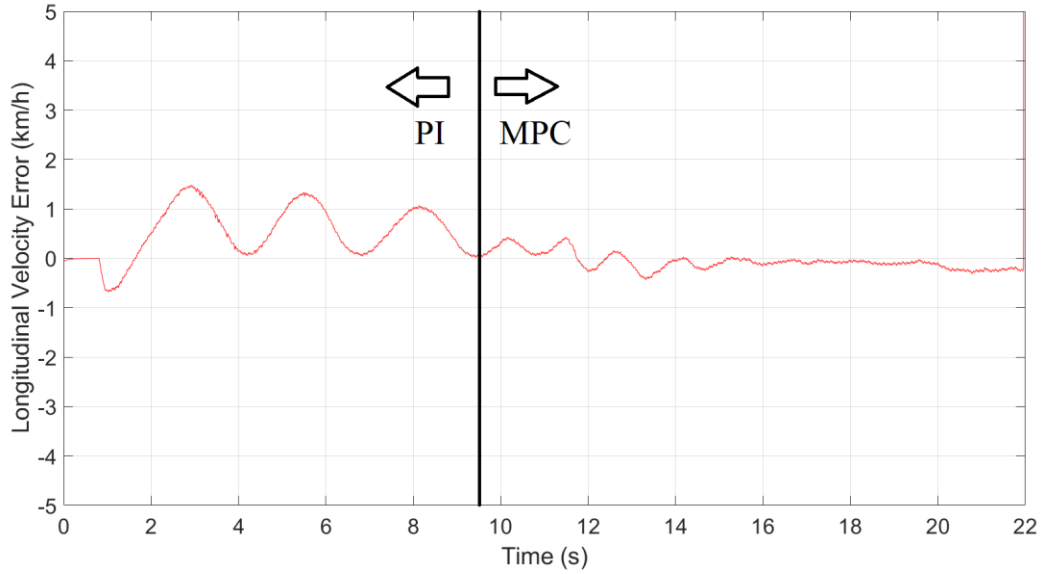


**Figure 7.2: Velocity tracking error for the vehicle ramp test.**

It is evident that during the ramp test the MPC performs satisfactorily. For the first 9.5 seconds, the controller operates in low gain PI mode. In PI mode the controller is tuned for smooth control application, so it is tolerant of some tracking error. The tracking error could be eliminated by selecting different gains for the PI controllers. After 9.5 seconds the controller switches to MPC mode, and the tracking error smoothly converges to zero.

Figures 7.3 and 7.4 depict the *APP* and *BPP* control inputs for the vehicle ramp test.

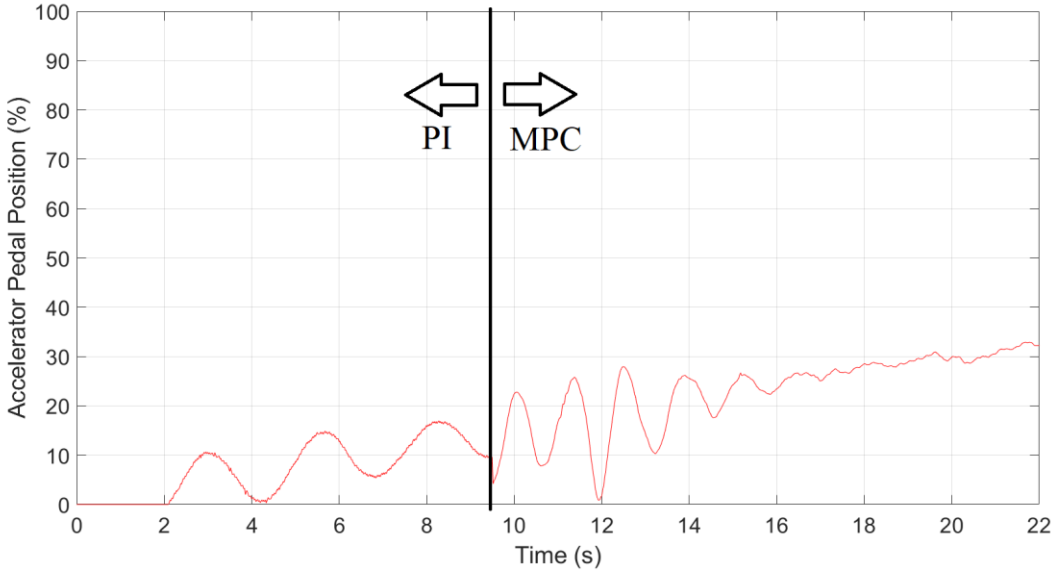


**Figure 7.3: APP control input for the vehicle ramp test.**

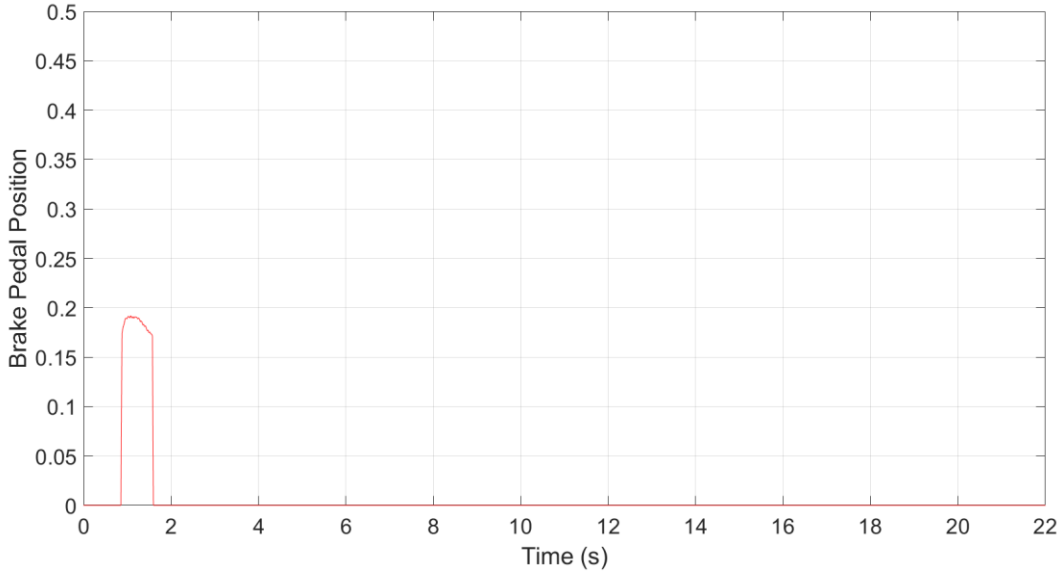


**Figure 7.4: BPP control input for the vehicle ramp test.**

The application of $APP$ is smooth throughout the test; only one small discontinuity occurs at 9.5 seconds where the controller switches to MPC mode.

### 7.3.2 Sinusoidal Velocity Test

The second benchmark vehicle test was the sinusoidal velocity test. Figures 7.5 and 7.6 depict the velocity tracking performance and velocity error for the sinusoidal test.

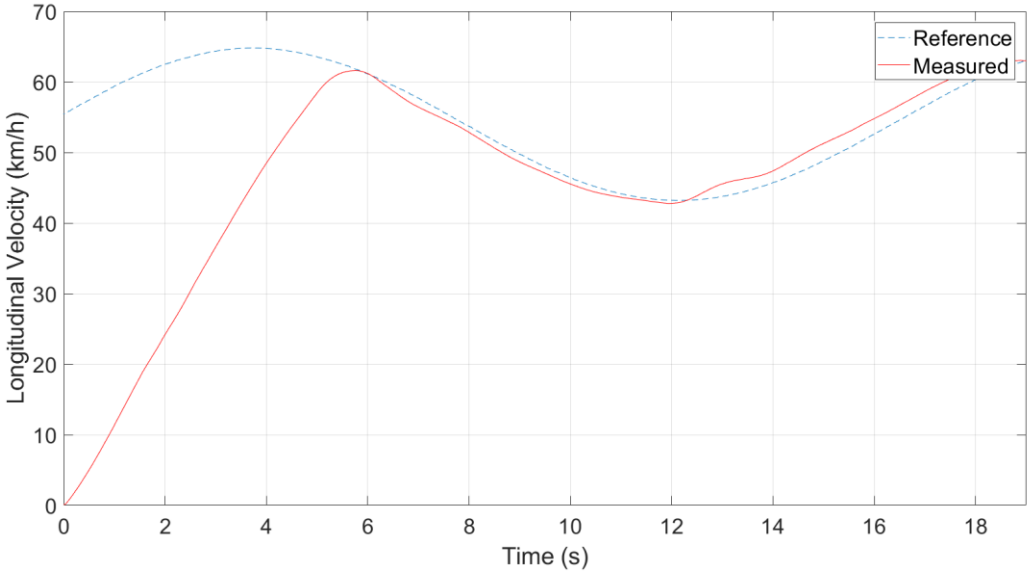


**Figure 7.5: Velocity tracking performance for the sinusoidal test.**

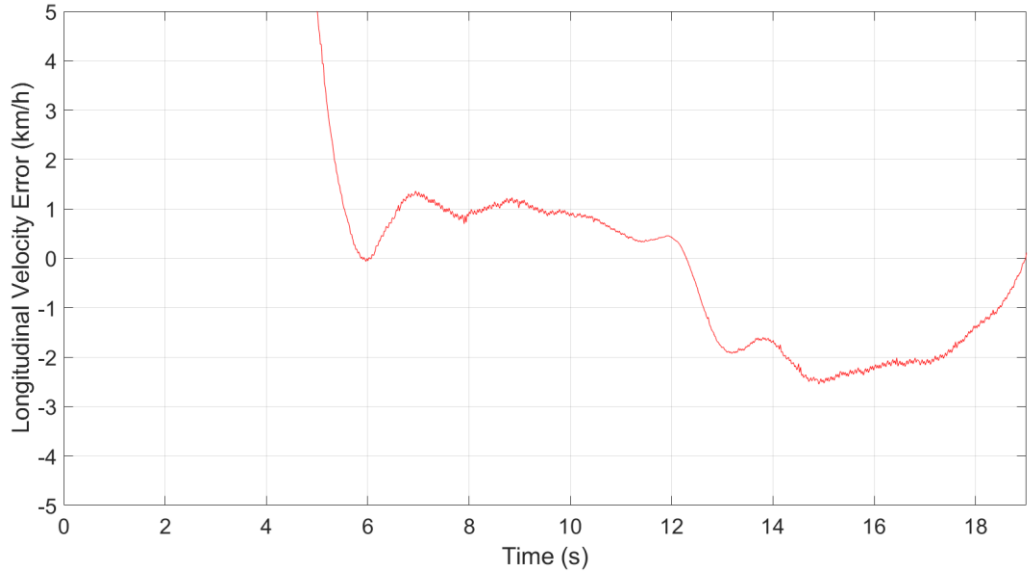


**Figure 7.6: Velocity tracking error for the sinusoidal velocity test.**

The MPC rapidly accelerates the vehicle up to $v_{ref}$ and begins tracking the sinusoidal reference. The controller tracks well during the braking maneuver between 6 and 12 seconds, but during acceleration some tracking error occurs. The MPC successfully reduces tracking error back to zero by the end of the test. A possible cause of the tracking error can be inferred by referring to the plots of $APP$ and $BPP$, depicted in Figures 7.7 and 7.8 respectively.

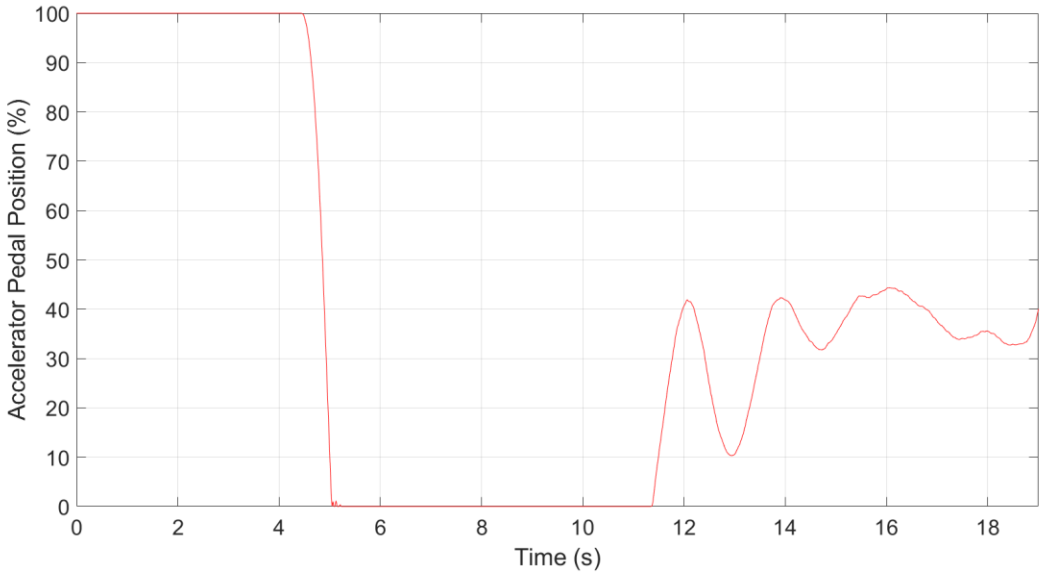


**Figure 7.7: APP control input for the sinusoidal test.**

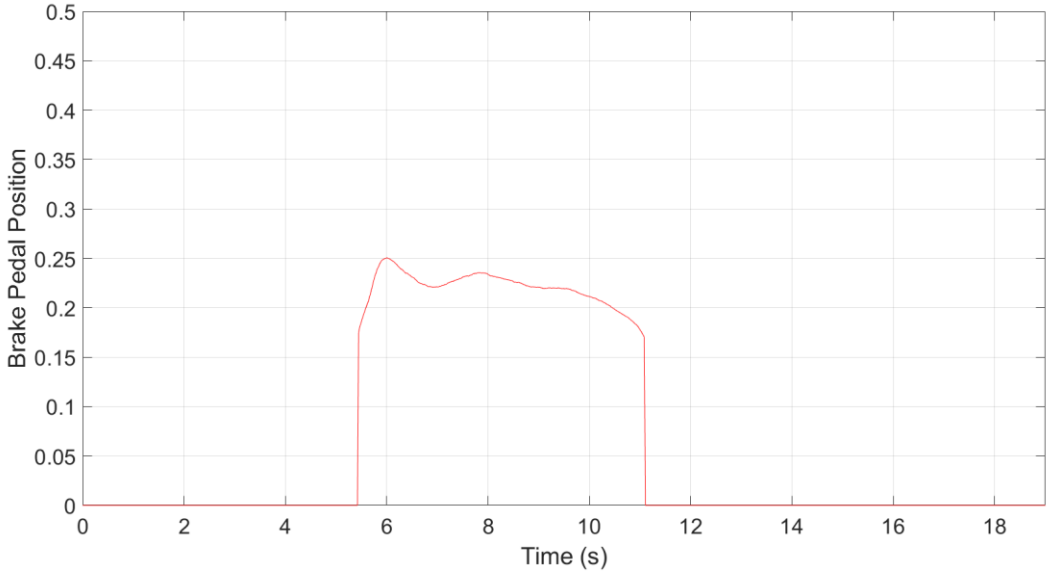


**Figure 7.8: BPP control input for the sinusoidal test.**

During the initial acceleration up to $v_{ref}$ and the subsequent braking maneuver, the MPC applies a suitable control input for velocity reference tracking. At the start of the first few seconds of the acceleration maneuver, from 11.5 to 15 seconds, the MPC produces an $APP$ input that oscillates with low frequency; later in the acceleration maneuver this behavior is damped out. It is likely that the cost function weighting on $\Delta GPP$, which was tuned in simulation, is too large. Further tuning of this weighing parameter, as well as the predication and control horizons, could be used to improve the performance of the MPC on the Moose. It is also likely that acquiring live measurements of $\omega_{eng}$ and $SOC$ would improve the predictive qualities of the MPC, and, as a result, the tracking performance would improve.

# Chapter 8
# Conclusions

## 8.1 Summary

The three primary objectives of this thesis have been successfully completed. Chapters 3, 4, and 5 described the process of modelling and identifying parameters of high-fidelity models of the Lincoln MKZ hybrid powertrain and braking systems. A grey-box modelling approach utilizing a combination of shallow neural networks and analytical modelling was used to emulate the system behavior. The models were identified from empirical vehicle performance data and fully integrated with a high-fidelity model of the vehicle dynamics. The complete model was validated by open-loop simulation.

In Chapter 6 the high-fidelity model of the powertrain and braking was used to inform the design of an instantaneously linearizing MPC for longitudinal vehicle dynamics control. The viability of the MPC and its advantages over classical control were proven by MIL simulation of multiple velocity tracking scenarios. Compared to a benchmark PI controller, the MPC showed superior reference tracking performance and better disturbance rejection. In addition, the MPC control action was smoother than the PI, and it did not chatter between $APP$ and $BPP$ control application.

Finally, in chapter 7 a process was developed for integration of the MPC into the Moose's autonomous stack. The extent of vehicle testing was limited by the length of available straight road at the test track. Early vehicle testing of the MPC has shown promising results, but some tracking error was observed. Acquisition of live measurements of $\omega_{eng}$ and $SOC$ and tuning of MPC parameters will likely improve performance.

## 8.2 Future Work

Multiple improvements can be made to the powertrain and braking models that will improve their accuracy. A limitation of the models described in this thesis is that they are both modelled for longitudinal applications only. Neither model considers the effects of lateral dynamics on torque transmission, but some experimental data has shown asymmetrical distribution of torque at the front

wheels during lateral maneuvers. It is possible that the Lincoln MKZ has an unmodeled torque vectoring controller that applies asymmetrical mechanical braking to aid in lateral maneuvers. These effects should be considered if the lateral dynamics of the Moose are to be modelled accurately.

The braking model of the Moose may be improved considerably if a means of isolating the measurements of mechanical and regenerative braking is determined. Modelling the regenerative behavior of the braking would eliminate the need for the virtual drivetrain clutch component, so the powertrain and braking models could fully integrate with one another. Modelling regenerative braking would also be necessary for any future modelling of the Moose's high voltage battery. An accurate battery model will allow the Autonomoose team to consider fuel efficiency in the design of future local planners and vehicle controllers.

The longitudinal dynamics MPC described in this thesis must undergo rigorous track testing before it can confidently be implemented for driving on public roads. The controller parameters should be tuned using a larger set of $v_{ref}$ profiles. In addition, live measurements of $SOC$ and $\omega_{eng}$ should be added to the controller to improve its predictive capabilities. The MPC should also be tested simultaneously with the Moose's existing lateral dynamics controller to confirm that its performance is satisfactory through lateral maneuvers. Additional modifications to account for lateral effects may be required.

A major contribution of this thesis work to the Autonomoose team was identifying a method for transferring controllers designed in Maple and MATLAB/Simulink to the Moose's autonomous stack. Using this method it is now straightforward to prototype new vehicle controllers and transfer them over to the Moose's stack for vehicle testing. There are several improvements that can be made to the existing MPC controller and implemented for future vehicle tests. One advantage of the existing MPC design is that it is possible to account for effects of measured or estimated disturbances. A state estimator for road slope, road conditions, drag, and other time-varying effects should be implemented alongside the controller, which will improve the predictions of the control-oriented model.

Future controller design work will explore the viability of nonlinear methods for the longitudinal dynamics MPC. Recent development of advanced strategies for model-order reduction by other

members of the Motion-Research Group may facilitate the real-time implementability of nonlinear MPC. A more advanced MPC for control of both lateral and longitudinal dynamics should also be designed and implemented to eliminate the need for two independent vehicle dynamics controllers. The combined controller should account for the interdependence of longitudinal and lateral dynamics on vehicle behavior, particularly with the tire forces. By simultaneously controlling the actuation of $APP$, $BPP$, and steering wheel angle, the combined MPC would control all aspects of local path tracking.

## References


[1] SAE International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.," SAE International, Warrendale, 2018.

[2] Ö. S. Tas, F. Kuhnt, J. M. Zöllner and C. Stiller, "Functional System Architectures towards Fully Automated Driving," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, Gothenburg, 2016.

[3] K. Jo, J. Kim, D. Kim, C. Jang and M. Sunwoo, "Development of Autonomous Car—Part I: Distributed System Architecture and Development Process," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 7131-7140, 2014.

[4] S. Thrun, "Simultaneous Localization and Mapping," in *Robotics and Cognitive Approaches to Spatial Mapping*, Berlin, Springer, 2008, pp. 13-41.

[5] AutonomouStuff, "AutonomousStuff," AutonomouStuff, 2018. [Online]. Available: https://autonomoustuff.com/. [Accessed 5 September 2018].

[6] Dataspeed Inc., "Dataspeed," Element5 Digital, 2018. [Online]. Available: http://dataspeedinc.com/. [Accessed 5 September 2018].

[7] K. Hornik, M. Stinchcombe and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989.

[8] R. Hecht-Nielsen, "Theory of the Backpropagation Neural Network," in *Neural Networks for Perception*, Academic Press, 1992, pp. 65-93.

[9] D. Q. Mayne, J. B. Rawlings, C. V. Rao and P. O. M. Scokaert, "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, vol. 36, no. 6, pp. 789-814, 2000.

[10] M. Behrendt, "MPC Scheme Basic," 2 October 2009. [Online]. Available: https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg. [Accessed 13 August 2018].

[11] S. Buggaveeti, M. Batra, J. McPhee and N. Azad, "Longitudinal Vehicle Dynamics Modeling and Parameter Estimation for Plug-in Hybrid Electric Vehicle," in *SAE World Congress*, Detroit, 2017.

[12] A. Taghavipour, R. Masoudi, N. Azad and J. McPhee, "High-Fidelity Modeling of a Power-Split Plug-In Hybrid Electric Powertrain for Control Performance Evaluation," in *15th International Conference on Advanced Vehicle Technologies*, Portland, 2013.

[13] F. U. Syed, M. L. Kuang, J. Czubay and H. Ying, "Derivation and Experimental Validation of a Power-Split Hybrid Electric Vehicle Model," *IEEE Transactions on Vehicular Technology,* vol. 55, no. 6, pp. 1731-1747, 2006.

[14] J. Liu, H. Peng and Z. Filipi, "Modeling and Analysis of the Toyota Hybrid System," in *International Conference on Advanced Intelligent Mechatronics*, Monteray, 2005.

[15] J. Liu and H. Peng, "Modeling and Control of a Power-Split Hybrid Vehicle," *IEEE Transactions on Controls and Technology,* vol. 16, no. 6, pp. 1242-1251, 2008.

[16] T. Hofman, M. Steinbuch and R. M. Druten, "Modeling for Simulation of Hybrid Drivetrain Components," in *Vehicle Power and Propulsion Conference*, Windsor, UK, 2006.

[17] B. Ganji, A. Z. Kouzani, S. Y. Khoo and M. Shams-Zahraei, "Adaptive Cruise Control of a HEV Using Sliding Mode Control," *Expert Systems with Applications,* vol. 41, pp. 607-615, 2014.

[18] S. Moon, I. Moon and K. Yi, "Desgn, Tuning, and Evaluation of a Full-Range Adaptive Cruise Control System with Collision Avoidance," *Control Engineering Practice,* vol. 17, no. 4, pp. 442-455, 2009.

[19] M. Batra, A. Maitland, J. McPhee and N. Azad, "Non-Linear Model Predictive Anti-Jerk Cruise Control for Electric Vehicles with Slip-Based Constraints," in *American Control Conference*, Milwaukee, 2018.

[20] D. Corona and B. De Schutter, "Adaptive Cruise Control for a SMART Car: A Comparison Benchmark for MPC-PWA Control Methods," *IEEE Transactions on Control Systems Technology,* vol. 16, no. 2, pp. 365-372, 2008.

[21] S. Li, K. Li, R. Rajamani and J. Wang, "Model Predictive Multi-Objective Vehicular Adaptive Cruise Control," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, pp. 556-566, 2011.

[22] S. E. Li, K. Li and J. Wang, "Economy-oriented vehicle adaptive cruise control with coordinating multiple objectives function," *Vehicle System Dynamics International Journal of Vehicle Mechanics and Mobility,* vol. 51, no. 1, pp. 1-17, 2012.

[23] M. Vajedi and N. L. Azad, "Ecological Adaptive Cruise Controller for Plug-In Hybrid Electric Vehicles Using Nonlinear Model Predictive Control," *IEEE Transactions on Intelligent transportation Systems*, vol. 17, no. 1, pp. 113-122, 2016.

[24] A. H. Borhan, A. Vahidi, A. M. Phillips, M. L. Kuang and I. V. Kolmanovsky, "Predictive Energy Management of a Power-Split Hybrid Electric Vehicle," in *American Controls Conference*, St. Louis, 2009.

[25] M. L. Kuang, F. U. Syed, A. M. Phillips, D. Ramaswamy and B. R. Masterson, "System and Method for Obtaining an Adjustable Acceleration Pedal Response in a Vehicle Powertrain". United States of America Patent 0112439 A1, 30 April 2009.

[26] M. Van Gennip, "Parameter Identification and Vehicle Dynamic Modelling of an Autonomous Vehicle," University of Waterloo, Waterloo, ON, 2018.

[27] M. Van Gennip and J. McPhee, "Parameter Identification for Combined Slip Tire Models using Vehicle Measurement System," in *SAE World Congress*, Detroit, 2018, Paper No. 2018-01-1339.

[28] Dataspeed Incorporated, "Throttle-Brake Combination By-Wire," Dataspeed Incorporated, Rochester Hills, MI, 2016.

[29] M. Green and J. B. Moore, "Persistence of Excitation in Linear Systems," *Systems & Control Letters*, vol. 7, no. 5, pp. 351-360, 1986.

[30] J. T. Heaton, Introduction to Neural Networks for Java, 2nd Edition, St. Louis: Heaton Research, Inc., 2008.

[31] Bundesamt für Strassen, "Leichte Motorwagen," 18 March 2009. [Online]. Available: https://commons.wikimedia.org/wiki/File:CH-Zusatztafel-Leichte_Motorwagen.svg. [Accessed 6 December 2017].

# Appendices

# Appendix A

# Signals Measured During Vehicle Parameter Identification Testing

**Table A.1: Table of signals measured during parameter identification testing.**

| Signal(s) | Symbol(s) | Signal Source | Signal Measurement | Sample Rate (Hz) |
|---|---|---|---|---|
| Wheel Speeds | $\omega_{FR}, \omega_{FL}, \omega_{RR}, \omega_{RL}$ | VMS WFS | Vector VN1640A CAN Interface | 100 |
| Wheel Torques | $\tau_{FR}, \tau_{FL}, \tau_{RR}, \tau_{RL}$ | VMS WFS | Vector VN1640A CAN Interface | 100 |
| Electric Motor Speeds | $\omega_{mot}, \omega_{gen}$ | VCM II Diagnostics | Vector VN1640A CAN Interface | ~4 |
| Power Source Torques | $\tau_{mot}, \tau_{eng}, \tau_{gen}$ | VCM II Diagnostics | Vector VN1640A CAN Interface | ~4 |
| Total Desired Torque | $T_{dsd}$ | VCM II Diagnostics | Vector VN1640A CAN Interface | ~4 |
| Vehicle Speed | $v_x$ | OBD II | Vector VN1640A CAN Interface | 10 |
| Engine Speed | $\omega_{eng}$ | OBD II | Vector VN1640A CAN Interface | 10 |
| Battery State of Charge | $SOC$ | OBD II | Vector VN1640A CAN Interface | 10 |
| Accelerator Pedal Position | $APP$ | OBD II | Vector VN1640A CAN Interface | 10 |
| Brake Pedal Position | $BPP$ | Throttle-Brake Combination By-Wire Interface | Moose's Onboard Linux Computer | 50 |

# Appendix B

## Double Layer Perceptron Regression Plots

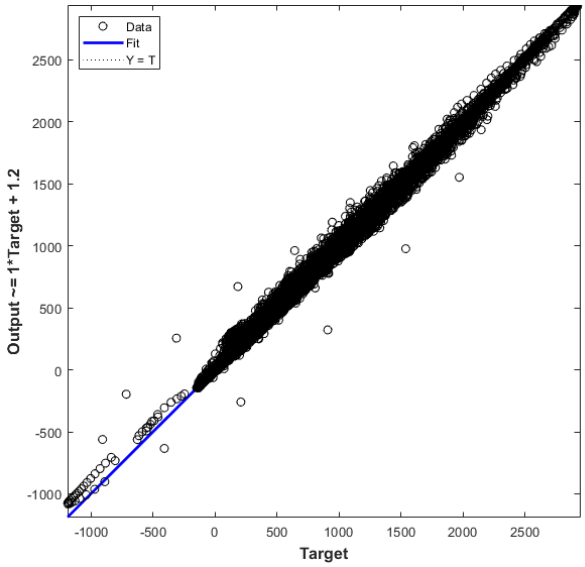


**Figure A.1: Regression plot of the supervisory torque NN model (R=0.999).**

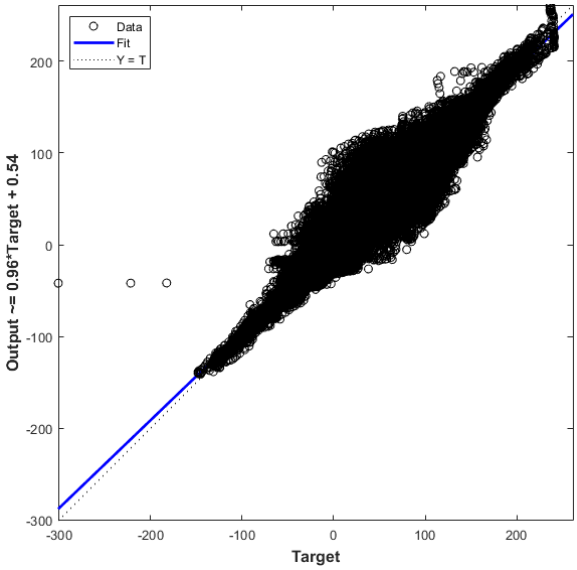


**Figure A.2: Regression plot of the TCM output torque NN model (R=0.979).**

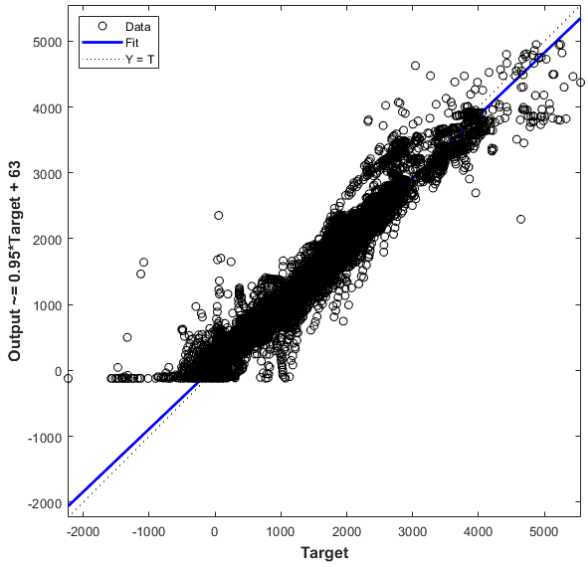


**Figure A.3: Regression plot for the front brake NN model (R=0.976).**

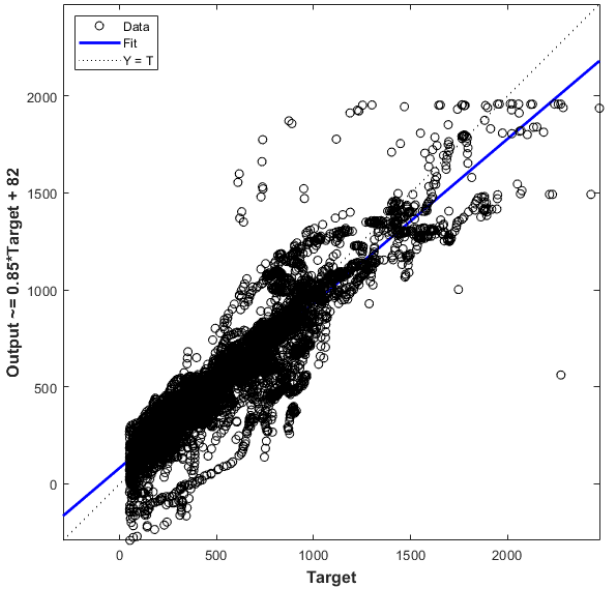


**Figure A.4: Regression plot of the rear brake NN model (R=0.920).**

# Appendix C

# Longitudinal Vehicle Dynamics Model Variable and Parameter Definitions

| Symbol | Description | Value (if a parameter) | Units |
|---|---|---|---|
| $A_f$ | Frontal vehicle area | 2.08 | $m^2$ |
| $C_d$ | Coefficient of drag | 0.8156 | Unitless |
| $C_l$ | Longitudinal tire stiffness | 11.0 | Unitless |
| $C_r$ | Rolling resistance coefficient | 0.01 | Unitless |
| $F_D$ | Force of drag | N/A | $N$ |
| $F_r$ | Rolling resistance force | N/A | $N$ |
| $F_x$ | Longitudinal tire force | N/A | $N$ |
| $F_z$ | Vertical tire force | N/A | $N$ |
| $a_x$ | Longitudinal vehicle acceleration | N/A | $m/s^2$ |
| $l$ | Vehicle wheelbase | 2.85 | $m$ |
| $l_F$ | Distance from front axle to COM | 1.32 | $m$ |
| $v_{op}$ | Vehicle velocity of the longitudinal slip operating point | N/A | $m/s$ |
| $\rho_{air}$ | Air density | 1.225 | $kg/m^3$ |
| $\omega_R$ | Rear axle rotational velocity | N/A | $rad/s$ |
| $h$ | COM height | 0.531 | $m$ |
| $g$ | Acceleration due to gravity | 9.81 | $m/s^2$ |
| $m$ | Vehicle mass | 2274 | $kg$ |
| $r$ | Tire effective radius | 0.347 | $m$ |
| $\theta$ | Road slope angle | N/A | $rad$ |
| $\sigma$ | Longitudinal wheel slip | N/A | Unitless |

**Table A.2: List of parameters and variables used in the control-oriented longitudinal dynamics model, simplified from the high-fidelity vehicle dynamics model [26].**