Modelling, Real-Time Simulation and Control of Automotive Windscreen Wiper Systems for Electronic Control Unit Development

by

Mark Dooner

A thesis submitted in partial fulfilment of the requirements

for the degree of

Doctor of Philosophy in Engineering

The University of Warwick

School of Engineering

January 2016

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to take this opportunity to thank my supervisor Professor Jihong Wang for her continued support thorough this project and for ensuring that I could financially support myself in my final year.

I would also like to thank my initial second supervisor Dr Jianlin Wei for his assistance with the dSPACE system, and my replacement second supervisor, Dr Xianping Liu. From Jaguar LandRover I would like to thank Dr Alexandros Mouzakitis, Mr Thomas Tsimos and Dr Georgios Tsampardoukas for their assistance. From the Power and Control Systems Research Lab I would like to thank my colleagues for their support, suggestions and friendship throughout my time here.

I must also thank my friends at the University of Warwick who kept me sane and made sure that I had some fun in my time off.

Finally I would like to thank my parents for supporting me in my decisions and not complaining too much when I was too busy to call home.

Mark Dooner

September 2015

Declarations

I hereby declare that the material in this thesis has not been submitted for a higher degree at any other university. This thesis entirely contains research work carried out by Mr Mark Dooner under the supervision of Professor Jihong Wang, unless references are given.

Some parts of this thesis were included in the following published papers:

M. Dooner, J. Wei, J. Wang, A. Mouzakitis, "Physical Modelling of a Windshield Wiper Linkage System for Hardware-in-the-Loop Simulation" in *The 13th Mechatronics Forum International Conference*, Linz, 2012

M. Dooner, J. Wang, A. Mouzakitis, "Development of a simulation model of a windshield wiper system for Hardware in the Loop simulation" in *Automation and Computing (ICAC), 2013 19th International Conference on*, London, 2013

M. Dooner, J. Wang, A. Mouzakitis, "Dynamic modelling and experimental validation of an automotive windshield wiper system for hardware in the loop simulation", *Systems Science & Control Engineering*, vol.3, no.1 pp. 230-239, 2015

Mark Dooner

September 2015

Abstract

In recent years there has been a growth in the automotive industry, coupled with a growth in the amount of electronic components and systems in a modern vehicle. The higher amount of electronics has led to an increased amount of Electronic Control Units (ECU) in a vehicle which require advanced simulation based testing procedures throughout their development process. One such method is Hardware in the Loop (HIL) simulation in which a real ECU is connected to simulation models of its environment via a real-time simulator. This project is concerned with developing a plant model of a windscreen wiper system for use in the development of Jaguar Land Rover's (JLR) body electronics ECU.

The system is divided into four parts which are modelled separately: Wiper motor, linkages, arm and blades, and the windscreen environment. The wiper motor and mechanical elements models are derived and implemented using the physical modelling tools SimScape and SimMechanics. A dynamic friction model describing the interaction between the wiper blades and the windscreen is developed, based on results presented in the literature. A simple aerodynamic model describing the forces on the wiper blades is also established.

The parameters of the models are derived using three sequential optimisation methods: Transfer function parameter identification, Genetic Algorithms (GA) and a nonlinear least squares local optimiser. A transfer function relating the motor current to the voltage was derived for step one, and a bespoke GA has been developed for step two. The parameters were successfully identified. Following this, Artificial Neural Networks (ANN) were used to convert the physical models into real-time capable models suitable for HIL simulation. Finally, adaptive control systems are designed in order to maintain the motor at a constant velocity.

The models are presented in a Simulink library and graphical user interface modelling tool for ease of use.

List of Abbreviations

ADC	Analogue to Digital Converter
ANN	Artificial Neural Network
CAN	Control Area Network
CFD	Computational Fluid Dynamics
CM	Centre of Mass
CS	Coordinate System
DAC	Digital to Analogue Converter
DDM	Driver Door Module
DOF	Degree of Freedom
ECU	Electronic Control Unit
EIL	Engine in the Loop
EMF	Electromotive Force
FEA	Finite Element Analysis
FPGA	Field Programmable Gate Array
GA	Genetic Algorithm
GUI	Graphical User Interface
HIL	Hardware in the Loop
HVAC	Heating, Ventilating and Air-Conditioning
I/O	Input/output
IS	Input Shaping
JLR	Jaguar Land Rover
LSE	Least Squares Estimation
MAAB	MathWorks Automotive Advisory Board

MIL	Model in the Loop
MSE	Mean Squared Error
NARX	Nonlinear Autoregressive Network with Exogenous Inputs
NB	Negative Big
NM	Negative Medium
NN	Neural Network
NS	Negative Small
OEM	Original Equipment Manufacturer
PB	Positive Big
PID	Proportional-Integral-Derivative
PIL	Processor in the Loop
PM	Positive Medium
PMDC	Permanent Magnet Direct Current
PMSM	Permanent Magnet Synchronous Motor
PS	Positive Small
RLS	Recursive Least-Squares
SAE	Society of Automotive Engineers
SIL	Software in the Loop
SNPID	Single Neuron Proportional-Integral-Derivative
ZO	Zero

Chapter 1 - Introduction

1.1 Background

The value of the automotive industry has been estimated by Clearwater Corporate Finance LLP in the Global Automotive Report 2013 at $800bn [1] and worldwide passenger car sales exceeded 60 million units in 2012 [2], as shown in Figure 1-1. This is matched by an increased production of passenger cars to greater than 60 million in 2012, increasing 5.3% from 2011 – a trend that has been sustained for the past decade [3]. Clearly, an automotive company who can produce high quality products in an efficient time scale can make large profits in such a market. Automotive products are safety critical by their nature, are subject to pressure/regulation for the reduction of greenhouse gas emissions and customer demand for quality and performance at low costs is high. As a result of this, automotive product development is a complicated process and industry standards such as ISO26262 are widely observed to help manage it [4].

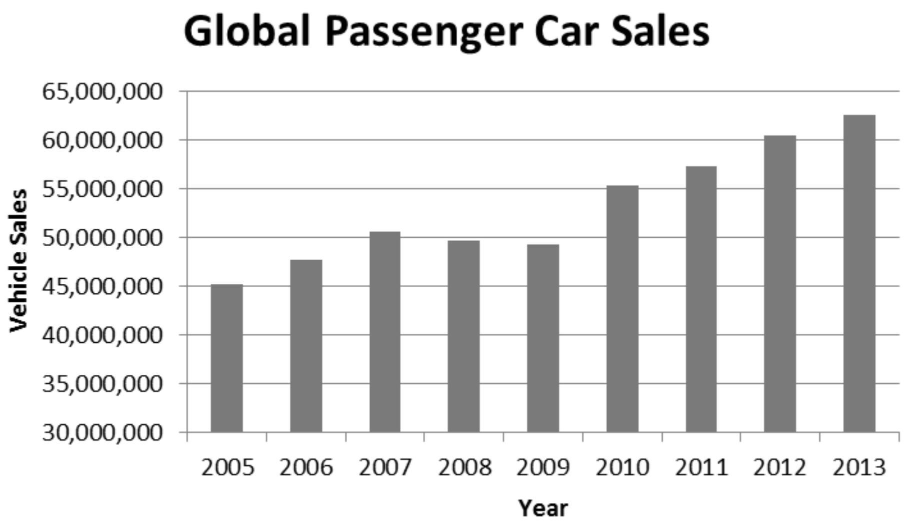


Figure 1-1: Global Passenger Car Sales

A modern luxury passenger vehicle is a highly complex product which includes numerous state of the art technologies. An ongoing trend in passenger vehicles is the increased use of electronic and mechatronic systems to carry out functions traditionally achieved with mechanical/hydraulic systems (such as steering and braking) in addition to adding new functionality to a vehicle, as demonstrated by a survey carried out in 2010 by Frede et al [5]. The annual market growth rate for electronics in vehicles is 7% per annum [6] as the market tends towards comfort, safety and reduced carbon emissions. It was estimated by Bosch [7] that the value of automotive electronics in the European market grew from €36 billion to €52 billion with 80% of that growth being new electronic components and the remaining 20% being replaced mechanical/hydraulic systems. The outcome of this migration to electronics is the need for highly integrated, safety critical electronic control systems.

Modern passenger vehicles can have in excess of 100 Electronic Control Units (ECUs) [8] running 100 million lines of software code controlling the various functions of the vehicle [9] [10]. This presents a significant challenge to automotive companies who need to develop these systems quickly enough to remain competitive whilst meeting stringent safety and quality standards. The development process for a product in the automotive industry, in this case an ECU or system of ECUs, follows the classic V model which is shown in Figure 1-2. The figure has been adapted from Bosch [7] to highlight the use of models in the development process of a modern product. Models used range from a model containing the system requirements, derived requirements and test requirements to plant models of vehicle components such as the engine or electronic components.

Figure 1-2 shows that when testing a product from component level to vehicle level, the preferred simulation method is Hardware in the Loop (HIL) simulation, which is an advanced simulation method in which a simulated system is interfaced with a real ECU (or multiple ECUs) [11]. Multiple sources demonstrate that HIL is used extensively in the development of automotive ECUs [8] [12] [13]. To carry out HIL testing, simulation models of the ECU's sensors, actuators and loads need to be developed. A derived requirement of such a model is that it must be able to be simulated in real time because it is interfacing with a real component. In this project a simulation model of a windscreen wiper system will be developed for use in the HIL simulation of the body electronics ECU.

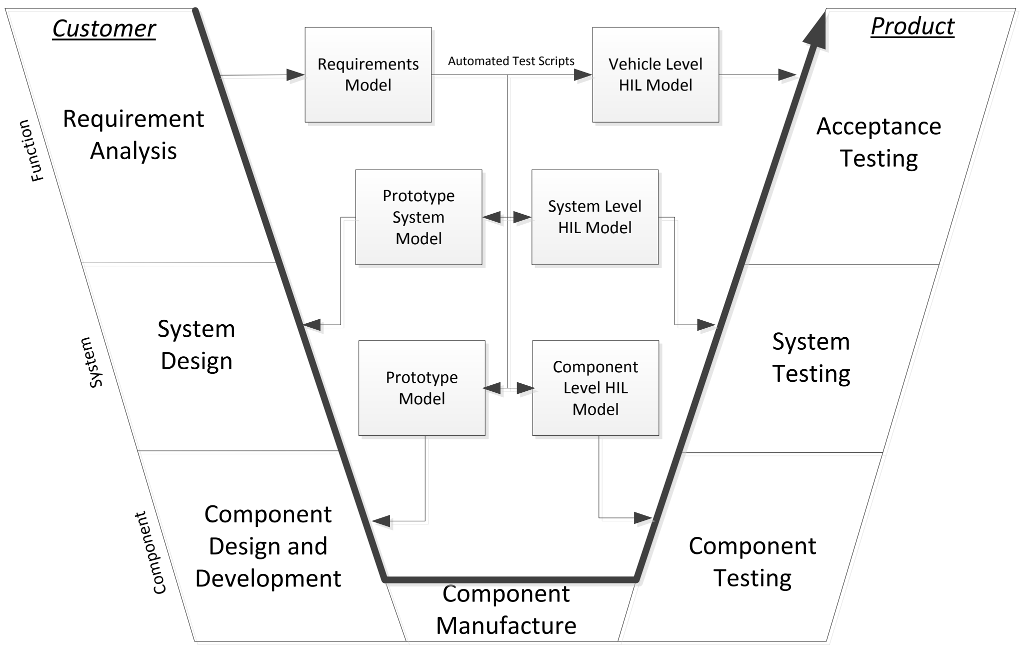


Figure 1-2: V Model for Product Development

A block diagram demonstrating the HIL testing of a body electronics ECU at component or system level is shown in Figure 1-3. The ECU is a real component being developed/tested and is connected to a HIL simulator, inside which are simulation models of the actuators, sensors and plant dynamics of the component(s) (such as the wiper system) connected to the ECU[1]. The HIL simulator contains power and communication I/O which imitate the real connections to the ECU and the simulation models contained within the simulator simulate

[1] Note that the actuators and/or the sensors could also be hardware components interfacing with the HIL simulator.

the dynamics of the components, outputting electronic loads and/or communication signals to be applied to the ECU under test. Numerous test cases can be carried out and repeated in using this configuration.

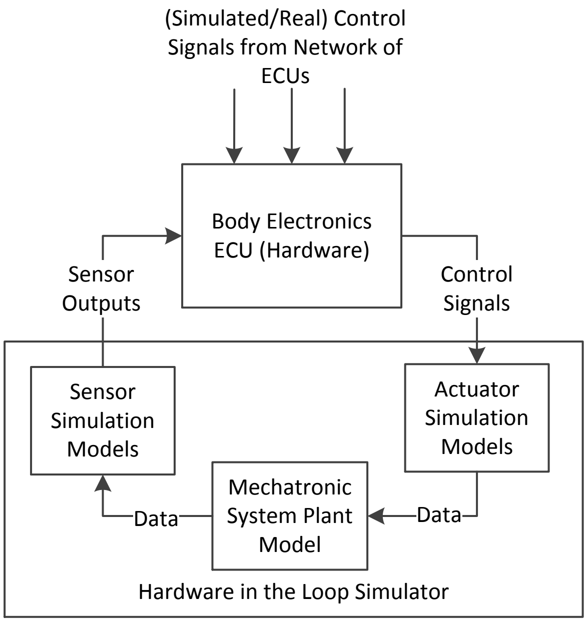

(Simulated/Real) Control Signals from Network of ECUs
Body Electronics ECU (Hardware)
Sensor Outputs
Control Signals
Sensor Simulation Models
Actuator Simulation Models
Mechatronic System Plant Model
Data
Data
Hardware in the Loop Simulator

Figure 1-3: Body Electronics ECU HIL Simulation

A block diagram of a wiper system is shown in Figure 1-4. The system's overall purpose is to expel water and debris from the windscreen and can be broken down into four parts: The wiper motor, the linkages, the arms/blades and the windscreen. The wiper motor is a unique design of a Permanent Magnet Direct Current (PMDC) motor with two electrical inputs for speed control. The motor is connected directly to the battery meaning that speed control via changing the input voltage is not possible. The linkages convert the constant rotational motion of the motor to the two output oscillatory motions of the wiper blades. The arms and blades combine to achieve the forward and reverse wipe of the windscreen. Finally, the windscreen protects the driver from rain, debris and wind and must be kept clear at all times.

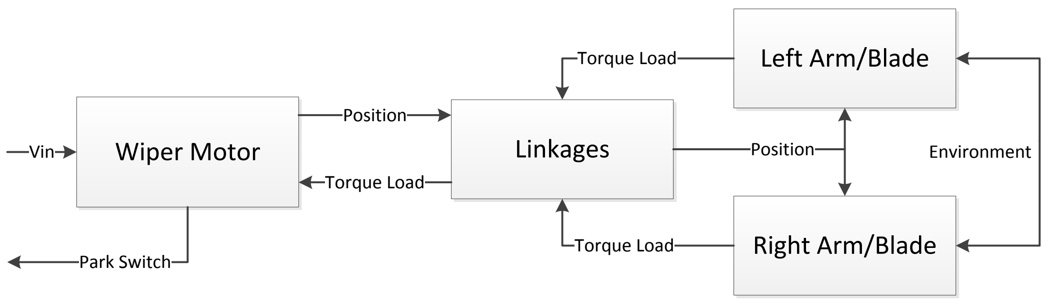


Figure 1-4: Wiper System Block Diagram

Currently, the body electronics ECU at Jaguar Land Rover (JLR) is connected directly to a real wiper system consisting of the wiper motor and linkages, which is known as a "no-load" wiper system. The disadvantages of this are that new prototypes of the wiper system are expensive and design updates take time to implement, the no-load system does not capture the full behaviour of the wiper system but a full wiper system is difficult to implement in a lab due to space restrictions and water management, and the real component cannot be used in early, simulation based control design and testing. The real-time capable model developed in this project alleviates these issues by allowing updates to be implemented immediately by changing model parameters, capturing the behaviour of the full wiper system without occupying lab space or needing water management, and allowing the same plant model to be used throughout the entire control design process.

1.2 Objectives

The main objective of this project is to design a tool to be used by JLR to develop windscreen wiper models suitable for HIL testing of ECUs to replace the real components. The tool must allow for modular design, meaning that the different elements of the wiper system defined above must be modelled separately and they must be able to be connected to each other. The models must be parametized in a way that is easy to update in order to incorporate design changes in the system. The models must accurately represent the real system whilst being capable of being simulated in real-time. The modelling tool is to be presented in the form a Graphical User Interface (GUI).

A secondary objective is to develop a controller for the wiper system that maintains the wiper velocity at a reference speed whilst rejecting the disturbances caused by the torque load. This will allow the wiper system to be operated at a lower speed, thus reducing the power consumption and reducing the wear on the PMDC motor brushes.

Concerning the area of HIL simulation as a whole, this project seeks to contribute a strong and repeatable methodology for developing simulation models of mechatronic system and investigate ways in which complicated multibody dynamic system can be simulated in real time. This should improve the effectiveness of automotive HIL testing and as a result improve the product development process and performance of new vehicles.

1.3 Structure

The remainder of this thesis is structured as described below:

Chapter 2 reviews the background literature pertaining to the work of the project. The review initially focuses on the strategies for automotive product development and how HIL is used to assist in this endeavour. Subsequently, existing research on wiper systems is presented and evaluated – highlighting the need for a flexible and real-time capable simulation model. Specific technical papers are then reviewed on subjects such as friction in wiper systems, parameter identification and adaptive motor control.

Chapter 3 analyses the structure of a wiper motor and determines how to model the system. Multiple models of the motor are designed and compared and the relationships between the motor's parameters are determined. The switching control strategies and implementations are also analysed and modelled.

Chapter 4 analyses and models the mechanical elements of the wiper system, i.e. the linkages, arms, blades and windscreen interface. Physical and mathematical models of the systems are synthesised and demonstrated. A model of the friction between the wiper blades and the windscreen is developed along with a simple model to simulate the effects of aerodynamic forces on the system.

Chapter 5 shows the work done to identify the unknown parameters of the wiper model. A three stage identification system is developed, starting with a motor transfer function parameter identification method, followed by a Genetic Algorithm (GA) designed to identify all of the system parameters and finishing with a local optimizer to refine the parameters and improve the accuracy of the model.

Chapter 6 shows how the models developed in the earlier chapters are simplified in order to make them appropriate for real-time implementation, and thus suitable for HIL simulation. The chapter focuses on the use of Artificial Neural Networks (ANN) to approximate the models. A hybrid physical-ANN model of the wiper system is developed, along with a full ANN model.

Chapter 7 describes the development of adaptive control strategies for the wiper system to track an input reference speed. The control system is based on a Single Neuron PID (SNPID) controller with adaptive weights and system gain. The results of the adaptive controller are compared to a classic PID controller.

Chapter 8 shows the development and implementation of the generic wiper system modelling tool which incorporates the work shown in previous chapters. The tool can be used by industry to quickly develop and update wiper models for the HIL testing of ECUs. The dSPACE real-time simulator used to generate validation data for the models is also shown.

Chapter 9 concludes the work done in this project and presents suggested further work.

1.4 Publications

Currently, the publications generated by this project are as follows:

M. Dooner, J. Wei, J. Wang, A. Mouzakitis, “Physical Modelling of a Windshield Wiper Linkage System for Hardware-in-the-Loop Simulation” in *The 13th Mechatronics Forum International Conference*, Linz, 2012

M. Dooner, J. Wang, A. Mouzakitis, “Development of a simulation model of a windshield wiper system for Hardware in the Loop simulation” in *Automation and Computing (ICAC), 2013 19th International Conference on*, London, 2013

M. Dooner, J. Wang, A. Mouzakitis, “Dynamic modelling and experimental validation of an automotive windshield wiper system for hardware in the loop simulation”, *Systems Science & Control Engineering*, vol.3, no.1 pp. 230-239, 2015

Chapter 2 - Literature Review

2.1 Introduction

This chapter presents a literature review of the previous work done in this project's research area. The chapter begins by investigating the use of HIL simulation and model based design in the automotive industry, and how it has evolved from a concept to an essential part of product development. Examples of how HIL improves the process of ECU development in the automotive industry are given. The purpose of this section is to give justification and scope to the work carried out in this project.

Following the investigation of HIL and model based design, a review of the current windscreen wiper models and control schemes available in the literature is given. This section also includes a review of papers published regarding previous work in this project. The purpose of this section is to demonstrate the need for a flexible, real-time wiper model.

The subsequent sections are concerned with reviewing previous work pertaining to the technical aspects of this project. Papers concerning online and offline parameter identification are reviewed in order to identify the best methods to apply to this project. Papers concerning the identification of the friction between the wiper blades and the windscreen are presented and methods of modelling the friction are explored. Also the aerodynamic elements of wipers are researched and discussed. Finally, the adaptive control of DC motors is reviewed and its application to a wiper motor discussed.

2.2 Automotive Product Development: HIL and Model Based Design

In 1999 a paper entitled "An investigation into the use of hardware-in-the-loop simulation testing for automotive electronic control systems" [14] was published, which explored the feasibility of using HIL to develop ECUs. Previous methods of ECU and vehicle development relied on physical prototypes of vehicle components; however this method was becoming less fit for purpose as the amount of electrical components and controller complexity in vehicles increased. HIL simulation was recognised as a possible solution for modernising the development of ECUs, with the authors of [14] citing a 1993 SAE paper called "Hardware-in-the Loop Simulation as a Standard Approach for Development, Customization, and Production Test of ECU's" [15] as evidence of this. The authors of [14] developed three separate plant models of the three subsystems of the Driver Door Module (DDM), namely the mirror, window and lock subsystems (building on work by the same authors previously published in [16]). The models were developed using Matlab/Simulink and it is stated that it is the introduction of Stateflow to the Simulink tool suite that facilitated the modelling of automotive subsystems in this way. State charts were already used in developing vehicle body electronics software, such as in [17] where a state model of a wash/wipe system was demonstrated, and the inclusion of a state chart simulator in Simulink greatly improved its suitability for vehicle body electronics modelling. The work demonstrated that, with relative ease, simple off-line plant models could be converted to online models and uploaded onto a real-time simulator and simulated in real time. The paper concludes that HIL can make a positive difference in ECU development and has the potential to be implemented in the future.

A review carried out in 2006 called "Review of hardware-in-the-loop simulation and its prospects in the automotive area" [18] confirmed the conclusion of the authors of [14]. The review found that, in the automotive industry, HIL is the preferred method of ECU testing, development and calibration. The paper also discussed other applications of HIL, with the focus being on an Engine in the Loop (EIL) system in which a real engine is interfaced with a simulated vehicle for testing and development. The review also highlighted that systems of HIL simulators connected online could allow separate elements of the vehicle to be modelled independently and then tested in a connected system. A survey of 80 technical papers in 2007 [12] confirms that HIL is now widely employed in ECU development in the automotive industry. The paper also states that there is a trend towards model based design, which is a recurrent theme amongst recent publications in the area of ECU development.

The advantages of using HIL over traditional prototype based production development methods are explicitly stated in references [19], [8] and [14], amongst others. The main advantages listed are 1) The ability to carry out testing in pre-production stages, 2) The ability to develop and test systems concurrently without having to wait for prototypes, 3) Dangerous and/or extreme condition tests can be carried out safely and easily repeated and 4) Tests can be standardized and automated.

Model based design of automotive ECUs generally refers to developing the control software in three stages: Model in the Loop (MIL), Software in the Loop (SIL) and HIL (sometimes Processor in the Loop (PIL) is used). References [20] and [21] give details as to how this process works. MIL refers to designing a control system and its associated plant model(s) in a simulation package such as Simulink and running test cases purely in simulation. Once a suitable control system has been developed, the controller model is compiled into production code (usually via automated methods) and simulated with the plant models used in the MIL testing stage. This is the SIL stage. Exactly the same test cases can be applied in both stages. Finally, the code tested and developed in the SIL stage can be implemented in an ECU (HIL) or another processor (PIL) and once again tested with the same test cases and plant models as before. This back to back testing procedure allows thorough testing of the controller before it is implemented, potentially saving time and money by identifying faults early. Model based design is also encouraged for meeting standards such as ISO26262 [4] [22] due to its iterative nature and traceability of results [23] [24].

It can be seen that HIL and model based design have become popular tools in ECU development and, as has been previously stated, this is largely down to the increase in electronics in modern vehicles [6]. In 2006, top of the range vehicles could have up to 70 interconnected ECUs controlling the various functions of the vehicle [8]; the number is now around 100 ECUs operating on an estimated 100 million lines of code [9]. It is also found that the majority of issues with vehicles that require OEM recall are electronics based, leading to increased pressure to improve the quality of testing in the design stage [13].

In summary, HIL and model based design have become important tools in automotive ECU testing, and numerous examples of their application are available in the literature, many of which are available from dSPACE: [25], [26], [27] and [28] are given as examples.

2.3 Current Wiper System Models and Research

This section explores current research into the wiper system and the models currently in the literature. The purpose is to justify the need for a real-time capable wiper model that models the system from the perspective of the wiper motor I/O. Note that research relating specifically to the friction between the blades and windscreen is discussed in Section 2.5 .

2.3.1 Vibration Analysis and Mitigation Research

The majority of the research in the area of windscreen wipers concerns the vibrations in the system. This research can further be split into modelling, measurement and reduction through control techniques or mechanical design. In 2000, a paper called "Dynamic Analysis of Blade Reversal Behavior in a Windshield Wiper System" [29] was published which developed a highly accurate three dimensional model of the wiper system's linkages, arms and blades. The purpose was to analytically investigate the causes of reversal noise (the noise caused by vibrations when the blade reverses its direction) and investigate ways to reduce it. The paper found that the angles and clearances between the wiper rubber and the blade had a large effect on the reversal noise, but also on other sources of noise in the system. The model is a complex multibody dynamic model which is not suitable for real time simulation and models behaviour that is not significant for the model to be developed in this project. Two later papers with shared authors to [29] used a similar model to investigate and mitigate squeal noise[2] in the wiper system [30] [31]. Both papers used measurements, FEA and mathematical models to investigate the noise. They concluded that it could be reduced by changing the configuration of the wiper (as in [29]), material choice and surface treatment. The three papers clearly show that multibody dynamic modelling is capable of modelling the wiper system to a higher fidelity than is needed in this project.

In 2002, chatter vibrations[3] were investigated in the paper "Simulation of Chatter Vibrations for Wiper Systems" [32] using a similar technique to the previous papers discussed. A mathematical model was developed and validated with real data to determine the factors affecting chatter vibrations. It was found, as in [31] [30] and [29], that the geometry of the blade and the rubber material had the biggest impact. Chatter vibrations were also investigated in [33] which used a validated finite element model to accurately simulate the system. Instabilities in the system that caused the chatter vibrations were then found using a complex eigenvalue method. The paper then proposes structural modifications to mitigate such vibrations.

A common method used to reduce friction induced vibrations is input shaping control. Input shaping is a control technique used to remove vibrations by combining impulse

[2] Defined as the high frequency vibration of the wiper system (around 1000 Hz)
[3] Defined as low frequency vibration in the wiper system (around 100Hz)

signals, designed to cancel vibrations, with the normal input command. In a 2008 paper called "Application of Input Shaping Control Strategy for Reducing Chatter Noise in the Automotive Wiper System" [34], the model developed in [29] was used to design an input shaping control system to reduce chatter noise. The vibrations that caused chatter were reduced by 30%. A paper released in 2010 with some shared authors with [34] follows a similar approach to [34] in order to use input shaping to reduce reversal vibrations with the same results (i.e. 30% decrease in vibrations) [35]. Similarly, in [36], input shaping is used to reduce unwanted vibrations. In this case, a particle swarm optimisation algorithm was used to time and shape the impulses in order to optimise the controller. Reduced vibrations were observed. Finally, in a paper entitled "Practical multi-objective controller for preventing noise and vibration in an automobile wiper system" [37], unwanted vibrations were reduced using input shaping, whose impulses were optimised using a Genetic Algorithm (GA). The model proposed in [29] was used to generate I/O data to train a recursive Artificial Neural Network (ANN) which was then implemented in the control system. A similar technique is used in this thesis to produce a real-time capable model. A collective weakness of these papers is that none of them implement the input shaping control in a real system, which will be driven by a wiper motor with its own dynamics. In reality, achieving an accurate target velocity is difficult to accomplish (see Chapter 7).

Other papers investigating vibration in wiper systems include [38] and [39], which specifically concentrate on how friction induces vibration using models and experimental measurements. Paper [38] concluded that squeal noise could be explained and modelled using the Stribeck friction effect. Paper [39] concurs with this, demonstrating that it is the negative friction-velocity curve at low velocities that causes the instabilities. Paper [40] takes experimental measurements of friction induced vibrations using microphones. The paper concluded that the wetness of the windscreen directly affected the amplitude and position (with respect to the wipe angle) of the vibrations. Note that [40] used the "Wet", "Dry" and "Half-Dry/Tacky" windscreen conditions that are used in this project. The chaotic modelling and control of the wiper system have been studied by Chang and Lin in [41] and Chang in [42]. These papers serve to demonstrate how complex the behaviour of a wiper system can get, but are out of the scope if this project.

In general, the papers researching the vibrational behaviour and control of a wiper system present models that are too complicated to be used in this project, but do serve as a useful tool for understanding the behaviour of a wiper system and as starting points for simpler models. A general weakness in most vibration based papers is that they do not take into account the wiper motor, and thus are ignoring an integral part of the system which has an effect on the dynamics and the potential control solutions.

2.3.2 Control of Wiper Systems

A paper entitled "Research on Passenger Car Windscreen Wiper Controller and Control Method Based on CAN" [43] designed a control system which operates using the Control Area Network (CAN) bus, which is used in electric vehicles. The idea is to replace older control systems, which tend to route power via switches, with a network based control method. Most modern vehicles employ wiper control based on CAN, the specifics of which depend on the vehicle model.

Most wiper systems work using one motor to drive both wiper arms; however some research is carried out into controlling systems with separate motors driving each arm[4]. The control challenge in this case is to synchronise the wipers to avoid collision. A recent paper released in 2015 [44] reviews current proposed control methods and suggests its own method using a Peripheral Interface Controller and an FPGA, although no details or results are provided. The most relevant paper cited by [44] is "On the Synchronization of a Pair of

[4] An advantage of this style of wiper system is the reduction in weight, noise and mechanical failure. However the system is more difficult to control.

Independent Windshield Wipers" [45] which models a two motor wiper system and controls the motors using reference trajectory planning and PID controllers. Current research on these types of wiper systems is limited to simulation.

A project is reported in [46], published in 2014, which developed an automatic system which controls the wipers using a rain and dust sensor and the sun visor via a light sensor. The purpose of the project is to fully automate the driver's visual comfort. It is likely that such systems will be implemented in future vehicles, for example many modern vehicles already implement a rain sensor.

2.3.3 Dynamic Models of Wiper Systems

The literature contains some dynamic models of wiper systems that share similarities with the model developed in this project. The paper "Dynamic Modeling and Control of the Windshield Wiper Mechanisms" [47] details the design process and control of a complete wiper system. The design and synthesis of the mechanical elements is implemented in the 3D modelling suite CATIA, with the geometry then transferred to the multibody dynamics modelling tool ADAMS for simulation. The ADAMS model is then co-simulated with a standard PMDC motor model and control system designed for controlling the motor velocity which are simulated in Matlab. The number of CAD packages used in this design process could be reduced by using the physical modelling tools in Simulink, streamlining the process. The controller is based on PID control and achieves good results in the test case shown, however only one ideal test case is presented, whereas a real wiper system undergoes dynamic torque loads and is required to be operated at multiple speeds.

A paper published in 2011 [48] develops a model of the wiper system's mechanical elements using the mechanical physical modelling tool in Simulink, called SimMechanics. The model does not include the motor, nor is it validated against real data, but the paper does show that SimMechanics is a powerful tool for modelling multibody dynamic systems.

SimMechanics will be utilised heavily in this project. The SAE paper [49] also reports a modelling system whose purpose is to aid the design of the mechanical element of the wiper system, although details of the system are not given. The paper presents results that indicate the dynamics of the wiper system and can be used as an early validation of the models developed in this project.

Finally, two papers have been published that report work previously carried on this project. These are: “Vehicle windscreen wiper mathematical model development and optimisation for model based hardware-in-the-loop simulation and control” [50] and “Accurate Model Based Hardware-in-the-Loop Test for a Windscreen Wiper System” [51]. Both report a “no-load” model[5] of the wiper system, whose unknown parameters are identified with a GA. The models successfully simulate the positional elements of the linkages; however the torque load is identified as constant, which is untrue. Also the motor equations do not model the two speed functionality essential for wiper motor operation, and the parameters for the transient conditions are incorrect, i.e. the motor takes many seconds to settle into steady state, whereas in reality it takes less than a second. This project will take the models presented here and add flexibility, fidelity and functionality to them.

The analysis of the literature concerning automotive product development and current wiper models has shown there is a gap in the literature for model of a windscreen wiper system that accurately represents the behaviour of the system which can be simulated in real-time, making it suitable for HIL simulation, to be developed.

[5] A model of the wiper system that only includes the wiper motor and the linkages, i.e. the arms, blades, windscreen and environment are not included.

2.4 Parameter Identification

2.4.1 Off-line Parameter Identification using Genetic Algorithms

Genetic Algorithms were introduced by Holland in 1975 [52] and have since been developed by numerous authors such as Goldberg [53]. They have become a popular tool for solving optimisation problems (such as parameter identification) and are applied in this project to identify the motor and friction parameters of the wiper system. The advantages of GAs are that they are well suited to solving non-linear and non-differentiable problems and are less susceptible to convergence on local minima than local optimisers [54].

GAs have been used in previous work on this project in papers [51] and [50] and there are many examples of them being used for parameter identification across the literature. Some examples are [55] – which showed that GAs can outperform a Least Squares Estimation (LSE) algorithm in PMDC motor parameter identification, [56] – which successfully identified the parameters of an induction motor using a GA and [57] – which used a GA to identify the parameters of a coal fired supercritical power plant. It can be seen from the examples that GAs can identify the parameters of complex models.

2.4.2 On-line Parameter Identification

The parameters of the wiper system are subject to change during operation due to factors such as wear and temperature changes. This is not an issue for a model to be used in HIL; however control systems based on state estimators rely on model parameters to estimate unknown states, such as the velocity of a DC motor. The paper "Sensorless speed control of DC servo motor using Kalman filter" [58] gives an example of using a Kalman filter to estimate the speed on a DC motor using measured voltage and current data along with a mathematical model of DC motor. The system performs well in ideal conditions but any changes in motor parameters will cause an error in the estimated speed.

Similarly, a paper published in 2010 [59] uses an observer to estimate the velocity of a DC motor, however in this case the resistance and inductance were estimated online using a Recursive Least-Squares (RLS) algorithm. This technique is relatively common in sensorless motor control [60] [61]. However, it is difficult to implement in this project due to the dynamic and unmeasured torque load and models based on physical modelling and ANNs.

Implementing sensorless speed control is suggested as further work in this project.

2.5 Friction in a Wiper System

One of the defining features of a windscreen wiper system is the friction between the rubber wiper blade and the glass windscreen. Friction is the dominant force contributing to the torque load acting on the wiper motor. For these reasons, and for the facts that friction has an effect on the lifetime of wiper blades, vibration and wiping performance, a lot of research into the friction in wiper systems is carried out. In addition, research into friction in general is active and has been for decades through numerous research streams such as physics, tribology, modelling and control. This review, and subsequent modelling work, will focus on the issues of defining and modelling friction, and friction specifically relating to wiper systems.

In 1968, Dahl released a much cited paper considering the physical phenomena causing friction and presented a simulation model for mechanical systems undergoing sliding (or rolling) motion [62]. The solution models three elements of friction: Static friction, Coulomb friction and Sliding/Rolling friction. Static friction is the force that must be overcome before a body subject to an external force will begin to move. Coulomb friction is the constant resistive force experienced by a moving body; its magnitude is less than that of static friction. Sliding or rolling friction (depending on the nature of the mechanical structure) is the frictional force experienced by a body which is dependent on the velocity of the body. This model has been widely used (and developed further) when simulating and controlling mechanisms or machines, as demonstrated in the survey paper entitled "A survey of models, analysis tools and compensation methods for the control of machines with friction" [63].

In 1994, the first survey on friction compensation that brought together research from numerous fields of study was published [63]. The study cites 280 research articles and examines many aspects of friction, from its cause to suitable compensation methods. In this project, certain conclusions of the survey should be taken into account: 1) There should be a theoretical as well as experimental justification for the friction model used in order to validate any assumptions and simplifications made. To satisfy this, research articles specifically in the area of windscreen wiper friction are studied. 2) Lubrication must be considered. This conclusion specifically refers to engineering lubricants and their effect on a system; however in the case of a wiper system the water on the windshield will have a large effect on the frictional force. 3) Analyses must be verified across the full range of application. The inevitable simplifications made due to the complexity of friction as a phenomenon mean that any model developed in this project must satisfy experimental data across all speeds and wetness levels to ensure that the simplifications are valid.

The paper "Friction Models and Friction Compensation" published in 1998 [64] briefly covers existing friction models, generally splitting them between static models[6] and dynamic models. Dynamic models take into account the hysteresis effects of friction, such as a lower friction force for decreasing velocities than for increasing. Most of the dynamic models given in the paper are extensions of the Dahl model.

[6] Most static models of friction are in fact dynamic since they are functions of velocity due to viscous friction.

A widely used friction model based on approximating the surface contact between two materials using the action of bristles was introduced in [65], and is also described in [66]. The purpose of the model was to capture the hysteresis effect of friction observed as the velocity oscillates around zero. If this effect is not taken into account the performances of friction models are poor at low velocities and are thus unsuitable for precise feedback control. The proposed brush model is an example of a dynamic friction model, which perform well at low velocities. Due to the nature of the model developed in this project, it is unlikely that a dynamic model such as this is needed. Also, dynamic models tend to be computationally expensive, which is unsuitable for HIL.

Identification of friction model parameters is an important aspect of friction modelling. A 2007 paper by Borsotto et al [67] considers the static and coulomb forces to be the most important and identifies them using a limit cycle method. A paper by Fujii [68] shows a method to dynamically measure the coefficient of friction between two bodies (using a wiper blade as an example) by measuring the inertial forces acting on the masses. In a paper by Nakajima et al [69], a GA was successfully used to identify friction coefficients. This method will be employed in this project. Similarly Kim et al [70] used an accelerated genetic algorithm to identify the parameters of a seven parameter friction model.

Papers studying friction specifically in wiper systems are now considered.

A paper by Bodai et al [71] investigates the friction force measured at the windscreen and wiper blade contact area using a rotating cylinder of glass in contact with a wiper blade. The paper largely focuses on explaining erroneous results by using an analytical model to demonstrate eccentricities in the cylinder of glass. However, the paper also gives results of a measured friction coefficient for different slip speeds and normal force components. The coefficient decreases with decreasing normal force and also with increasing speed. This disagrees with many friction models presented in papers above, which suggest that that the friction force increases with speed. These results correspond with a later paper by the same authors [72] which uses finite element modelling and experimental measurements to investigate sliding friction of a wiper blade.

Work done by Buta [73] measured the friction force of a wiper blade at three different speeds in wet and dry conditions. The results suggested that the friction increases with speed, which disagrees with [71] and [72], and is higher in dry conditions.

A 2009 paper by Deleau et al [74] attempts to measure sliding friction between a wiper blade and glass in wet, dry and tacky conditions. The results show a largely linear relationship between the normal force applied to the blade and the frictional force. In agreement with [71] and [72], the results also show an initial increase in frictional force with velocity up to around 100mm/s, followed by a constant or decreasing force as speed increases. The paper also shows that for a constant force and speed, the friction force/coefficient is at its lowest in the windscreen's wet condition, the highest in its tacky condition and second highest in the dry condition. This is agreement with [73].

A 2007 paper by Koenen et al [75] agrees with [74] and [73] by showing that the coefficient of friction is highest when the windscreen is tacky, lowest when wet and medium when dry. The paper shows results giving a higher wet coefficient of friction at low speeds which level out at higher speeds. This agrees partially with [71] in that at 100mm/s the coefficient of friction levels out, however in this case it seems to rise slightly as the velocity increases. Also, an increased load causes a decrease in the coefficient of friction in wet conditions.

In Chapter 4 data from the papers [72], [68] and [75] is plotted and used to develop a dynamic friction model to apply to the wiper model. A six parameter continuously differentiable friction model is presented in [76] which is used as a basis for the model developed in this project. A dynamic friction model is developed instead of, for example, an average friction or worst case friction model for a number of reasons. Firstly, it allows the model to be used to analyse the dynamic current load on the vehicle's battery. Secondly, the load dynamics of the system can be modelled to a higher accuracy, allowing the development of control systems (such as that shown in Chapter 7) that can adapt to changing load conditions. Finally, the addition of the dynamic friction model does not add a large burden on simulation time when compared with other elements of the model, particularly the multibody dynamic element calculating the position and inertial forces of the mechanical system.

2.6 Vehicle and Wiper Aerodynamics

There is a large amount of work and research conducted into the aerodynamics of vehicles due to the fact that the aerodynamic design has such a great effect on the performance of the vehicle. In general, the study of vehicle aerodynamics is out of the scope of this project; however the forces experienced by the wiper system when the vehicle is moving will have an effect on its performance. This section begins by giving some example studies on vehicle aerodynamics to demonstrate the general process of developing and validating models, and then gives details on papers specifically related to the aerodynamic forces experienced by the wipers.

A paper published in 1993 called "Aerodynamics of road vehicles" [77] gives a good overview of the implications of aerodynamic forces on vehicles, such as drag and trailing vortices, and how they affect the design of vehicles. Importantly, the use of Computational Fluid Dynamics (CFD) as a method of design and testing are discussed. The reference models used in the CFD simulation of vehicles are studied in [78], which shows how the complexity and fidelity of the models have increased as the technology has developed and that simple reference models are often suitable for CFD purposes. There are numerous cases of CFD being used to simulate the fluid flow over an entire vehicle, such as [79] which investigates the effect of the rear slant angle of a simplified reference vehicle model on the air flow over it and [80] which uses a wind tunnel to validate and fine tune a CFD model of a vehicle to simulate the flow field around it. Also, more specific analyses can be carried out, such as [81] which investigates the heat distribution across the windscreen due to heat jet nozzles placed at the bottom of the windscreen. One such specific analysis is the study of aerodynamic forces on wiper systems.

A 2009 paper called "The Appropriate Use of CFD in the Automotive Design Process" [82] investigates ways of reducing development costs of a vehicle in the design phase by using CFD to its fullest potential[7]. One element discussed by the paper is the effect of the wipers on vortices which induce drag. Results are not given, however it is clear that the effects are complex and could not be captured in a real-time model. A study in 2013 [83] investigated, using CFD validated with wind tunnel experiments, the optimal placement of the wipers in the park position in order to reduce audible noise due to the vortices that they cause. As in [82], the paper serves as an example of the complexity of the fluid flow around wipers. Two studies by Valeo Wiper Systems in 2001 [84] [85] specifically investigate the drag and lift forces on wiper systems due to the velocity of the car. Paper [84] compared forces calculated with CFD with experimental data and found that the error was generally less than 10%. The paper demonstrates that different wiper blades can experience significantly different drag and lift forces under the same test conditions and that the addition of a spoiler to the blades can develop a negative lift force (i.e. a lift force towards the windscreen). Typical measured results were around 12N for the drag force and 4.5N (-4.5N with a spoiler) for lift. Paper [85] has a similar work flow to [84] but compares the forces experienced by a classical wiper blade and a new design of blade formed by one

[7] An analogy can be drawn between the costs saved using by using HIL (and other "X-in-the-loop" style simulations) and CFD simulation, although the methodology and fidelity of the models are very different.

piece of rubber[8]. Again, the paper highlights the differences in aerodynamic forces experienced by different wiper designs under the same test conditions. Finally, a Korean paper published in 2001 [86] determines the drag and lift coefficients of the driver and passenger side wiper blades at high speeds. Results of the CFD analysis show that the drag and lift coefficients are dependent on the wiper angles and speed of the vehicles. Results from papers [84] [85] and [86] are used in Chapter 4 in a simple aerodynamic model implemented in Simulink.

2.7 Adaptive Control Using Single Neuron PID Controllers

A common method of PMDC motor control is to use closed loop PID (Proportional-Integral-Derivative) control. The angular velocity of the DC motor is fed back and compared to a target velocity to form an error signal. The error signal is the input to the PID controller which uses constant gains to generate a control signal based on the error, its derivative and its integral to apply a control signal to the motor. The method generally has good performance but it is unable to adapt to changes in conditions and usually requires a trade-off between transient performance, reference tracking and disturbance rejection. Single Neuron PID (SNPID) controllers are a method of emulating PID controllers with the added feature of adaptive gains. The three PID gains are replaced with three weighted inputs to a single output Neuron. The weights can be adjusted in real time depending on the performance of the system.

A paper released in 2010 [87] successfully implements a SNPID controller for the speed control of a PMDC motor. The system was designed in Matlab/Simulink and implemented in a dSPACE simulator, demonstrating that the method is feasible for this project. The system in paper [87] was able to update the SNPID gains, however the output of the Neuron is multiplied by a linear gain, K, which has a large effect on the system

[8] A design now commonly used in road vehicles.

performance. Many studies extend the SNPID controller to include an adaptive value of K. One such paper [88] achieves this using an adaptive controller based on the biological T cellular immunity adaption mechanism. It was shown that by adapting K the disturbance rejection of the controller was improved along with the rise time.

A more conventional method of tuning K is to incorporate a fuzzy logic controller into the system. This technique is used in the papers [89] and [90]. The input to the fuzzy controller is the error signal and the increment of the error (to capture the error change rate) and the output is the value of K. Paper [90] compared a fuzzy controller with only the error signal as an input to one with the addition incremental error input and found that the performance was not improved with the addition of the incremental input. Both studies reported a very good performance from the controller in terms of start-up performance, reference tracking and disturbance rejection.

There are also many examples of SNPID controllers being used for purposes other than PMDC motor control such as Permanent Magnet Synchronous Motor (PMSM) control [91], a Heating, Ventilating and Air-Conditioning (HVAC) system [92], a switched reluctance generator system [93] and a mechanical prestressing system [94]. A SNPID system with a fuzzy controller adapting K is implemented in Chapter 7 of this thesis. SNPID control was chosen for this project due to its ability to quickly update the PID gains to adapt to changes in load and velocity demand and its previously successful applications in PMDC motor control. It also builds on the use of the neurons in Artificial Neural Networks (ANN) used in the real-time modelling of the wiper system, shown in Chapter 6.

Chapter 3 - Wiper Motor Modelling

3.1 Introduction

The electrical motor used for driving the wiper system has a different structure from ordinary PMDC motors. The literature study presented in Chapter 2 revealed that there is currently no available model representing the behaviour of a wiper motor which is suitable for this project. The only model found was presented in two papers by Wei et al [51] [50]; however the model does not take account of the two input speed control and, due to its parameters, is only suitable for steady state simulation, which is not representative of wiper motor behaviour. Models of standard PMDC motors are common and well known; however the two input speed control of the wiper motor has not been adequately modelled, although the underlying physics and operation is known.

This chapter begins by presenting the basic structure of a wiper motor in order to understand how the position of the two input brushes, the winding pattern of the armature conductors and the gearing on the shaft affect the behaviour of the motor. This information is then used to develop a mathematical representation that describes the wiper motor and how the parameters differ from a standard PMDC motor model. Following this, simulation models of the wiper motor are developed. Firstly, a simple state space model is developed to verify the validity of the equations in terms of describing the behaviour of the motor. Secondly, a physical model[9] of the motor is presented which can interface with other elements of the simulated wiper system (i.e. the control system and the linkages). Once the basic wiper motor model is designed, the wiring and switching strategies are presented and modelled. When a wiper system is switched off, the motor

[9] In this document, physical models refer to the computer modelling technique of representing each physical element of the system to be modelled with a corresponding simulation block connected with lines representing physical connections in the system.

carries on until the wipers are in their park position – which is achieved by the use of a park switch. The simulation models of these methods are demonstrated.

The models developed here are validated against real data in Chapter 5.

3.2 Wiper Motor Structure

A wiper motor consists of a brushed PMDC motor, with two electrical inputs and one electrical output, driving a worm and wheel gear configuration to decrease the speed and increase the torque of the driven load, i.e. the linkages, arms and blades. Each wiper motor also includes a park switch, the nature of which varies and is determined by the design of the specific motor. This is discussed in more detail in Section 3.6. An example of a wiper motor can be seen in Figure 3-1. The left picture (a) shows the motor casing and the worm and wheel gear configuration. The driving shaft, connected to the crank of the linkages, is attached to the centre of the wheel gear. The left picture also shows the park switch connection on the wheel gear. The right picture (b) has the casing removed and the windings and armature of the PMDC motor can be seen.

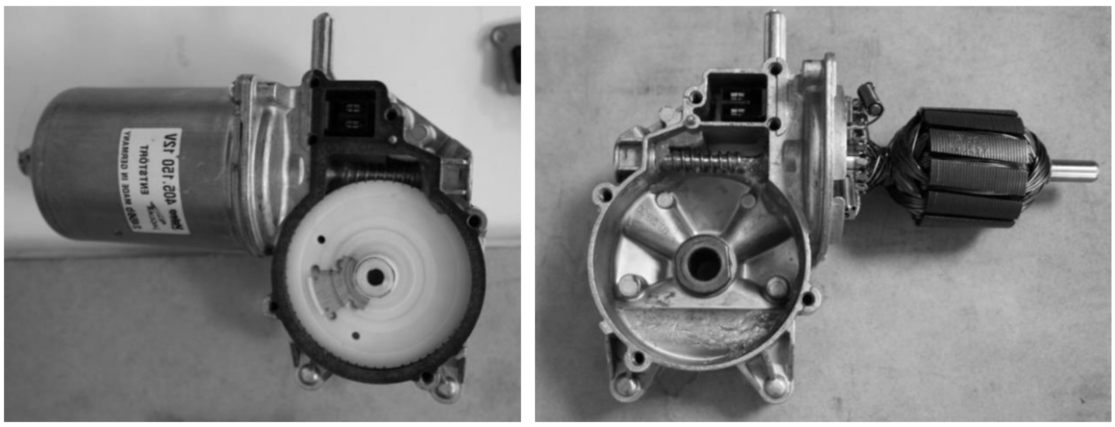

(a) **(b)**

Figure 3-1: Wiper Motor Construction

To comprehend the nature of the two input speed control, the position of the two electrical inputs in relation to the permanent magnets and the commutator must be examined. This configuration can be seen in Figure 3-2. Diagram (a) shows the input brushes on the left and the output on the right of the base. The slow speed input brush is in line with the common output brush and the fast speed input brush is offset. This can clearly be seen in the top of diagram (b). The bottom of diagram (b) shows the position of the brushes with respect to the magnets, highlighted in blue. The magnetic neutral line dissects the slow and common brushes, thus the fast input brush is offset, and is known to cause an increase in rotational speed [95].

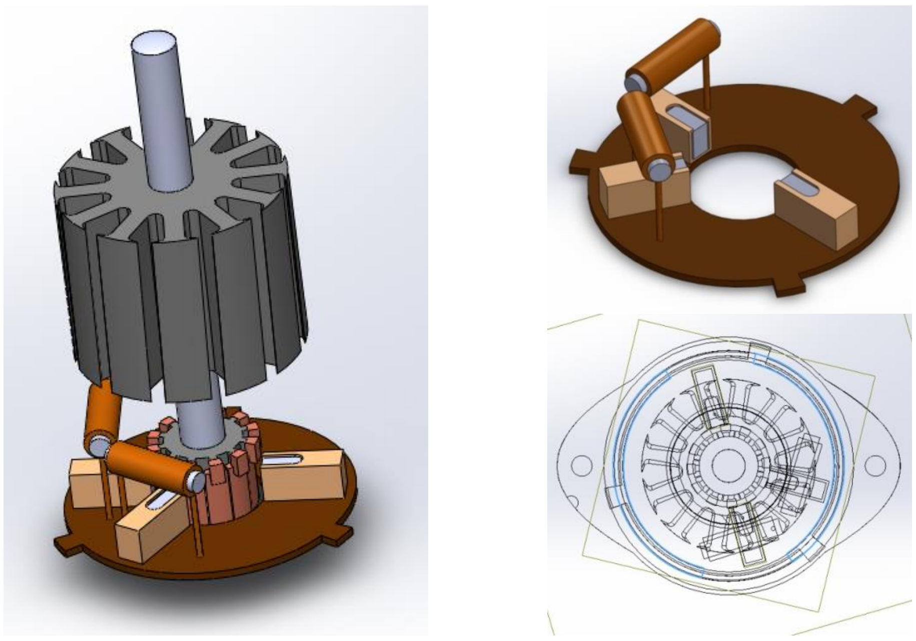

(a) (b)

Figure 3-2: Wiper Motor Electrical Input Placement

The wiring strategy of a wiper motor is now examined. First, the connection of each winding to the commutator is analysed, this is shown in Figure 3-3. The figure imagines that the commutator has been laid flat and each segment is labelled from A to L. Segments A and L are also adjacent to each other. It can be seen that the return path of each coil of wire (denoted by n' where n identifies the coil) connects to the commutator segment adjacent to its origin. This is known as lap winding and is important for determining the lengths and position of the current paths.

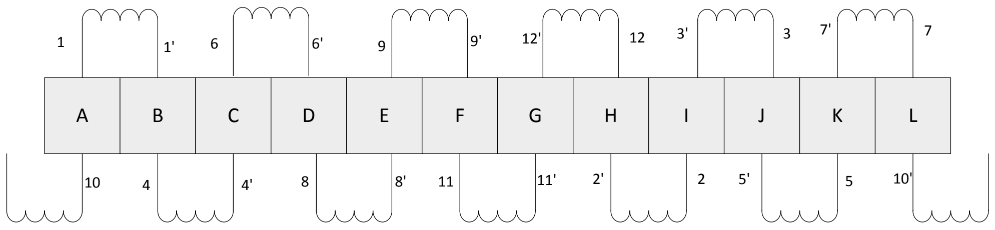


Figure 3-3: Wiper Motor Commutator Connection

The final element to consider when analysing the structure of a wiper motor is the physical configuration of the coils around the armature. This is shown diagrammatically in Figure 3-4. The coils, in this case 12, can be considered as being in pairs. For example, coils 10 and 11 are both wound on the bottom layer of the coils. On top of these are coils 7 and 9, followed by 5 and 8, 3 and 6, 2 and 4, and finally 1 and 12 which are on top. This method of winding has disadvantages. For example, coils 10 and 11 are noticeably shorter than coils 1 and 12 because they are on the bottom of the windings. This causes a difference in the resistance and inductance of the coils and they are subject to a different magnetic field. This can cause imbalances in the motor performance. However, it is an easy way to wind the motor for a mass produced product such as a wiper motor and is thus fairly common. In this analysis the slight changes in the parameters of each coil caused by this winding strategy are assumed to be negligible.

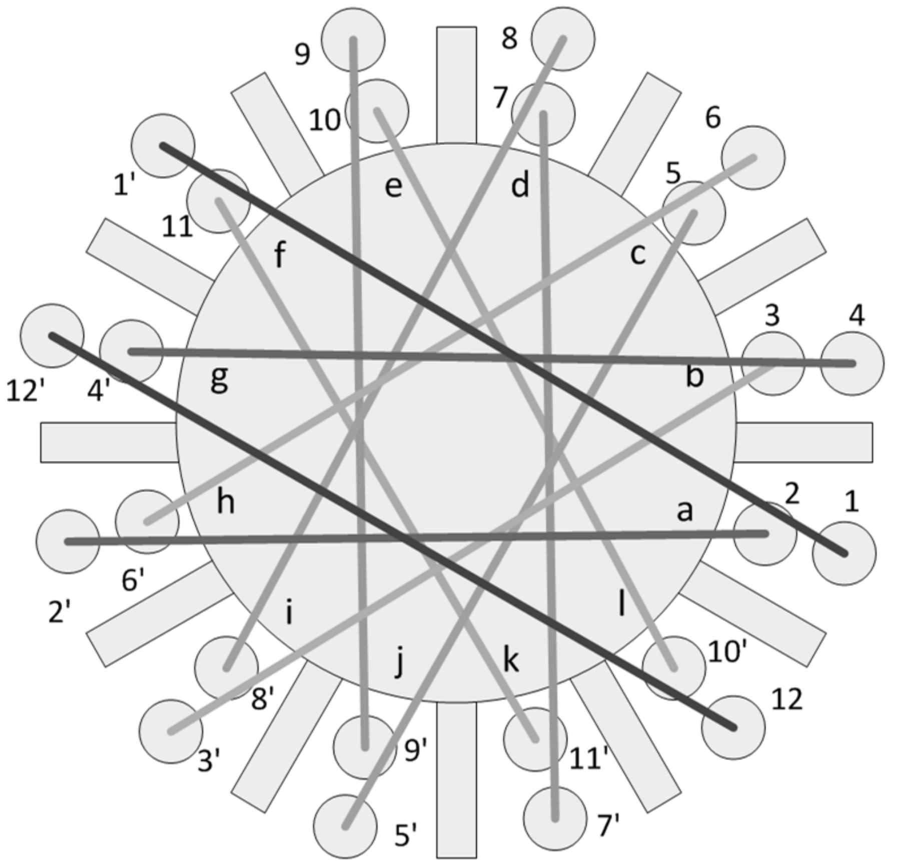


Figure 3-4: Wiper Motor Wiring Diagram

3.3 Model Derivation of the Wiper Driving DC Motor

With knowledge of the structure and winding strategy of the wiper motor, the parameters and dynamic equations governing its behaviour can be derived. The derivation shown here is based on the process shown in reference [96]. The assumptions made when modelling the motor are as follows:

- None of the coils are electrically shorted during commutation
- Torque ripple is negligible
- The slow and fast input brushes have equal resistance
- The parameters are not affected by changes in temperature
- The magnetic flux generated by the PMs is directed radially out of, or into, the armature.

3.3.1 Torque Production

To understand how the PMDC motor produces torque when a current is supplied, how it can be mathematically modelled and how the wiper motor behaves differently in its slow and fast modes of operation, it is useful to examine the armature current paths in each mode. For the 12 coil wiper motor examined, and when ignoring commutation effects, the current paths can be displayed as shown in Figure 3-5 for the slow operation and Figure 3-6 for the fast operation.

The 12 armature slots are labelled a to l and the coils are labelled 1 to 12 with a tick denoting its return path. The instantaneous direction of the current is represented classically by a dot meaning current out of the page and a cross meaning current into the page. The colours signify the two current paths from the input brush to the common ground brush. The blue current path is current path one and the red is current path two. The arrows show the direction of the magnetic field generated by the permanent magnets, with the dotted line denoting the magnetic neutral line. The magnetic field is described either by the flux density vector, $\vec{\mathbf{B}}$, or by the magnitude of the magnetic flux density, B , and the unit vector, $\hat{\mathbf{r}}$, which always points radially away from the centre of the armature.

It can be seen in Figure 3-5 that for the slow speed operation the current paths around the armature are symmetrical about the magnetic neutral line. This is the case in a normal PMDC motor. In the fast speed operation shown in Figure 3-6 however, two observations can be made. The first is that the current paths are now unbalanced, which will affect the behaviour of the motor. The second observation is that current path one is physically longer than current path two, meaning that the two current paths will have a different resistance and inductance. This needs to be taken into account when modelling the motor.

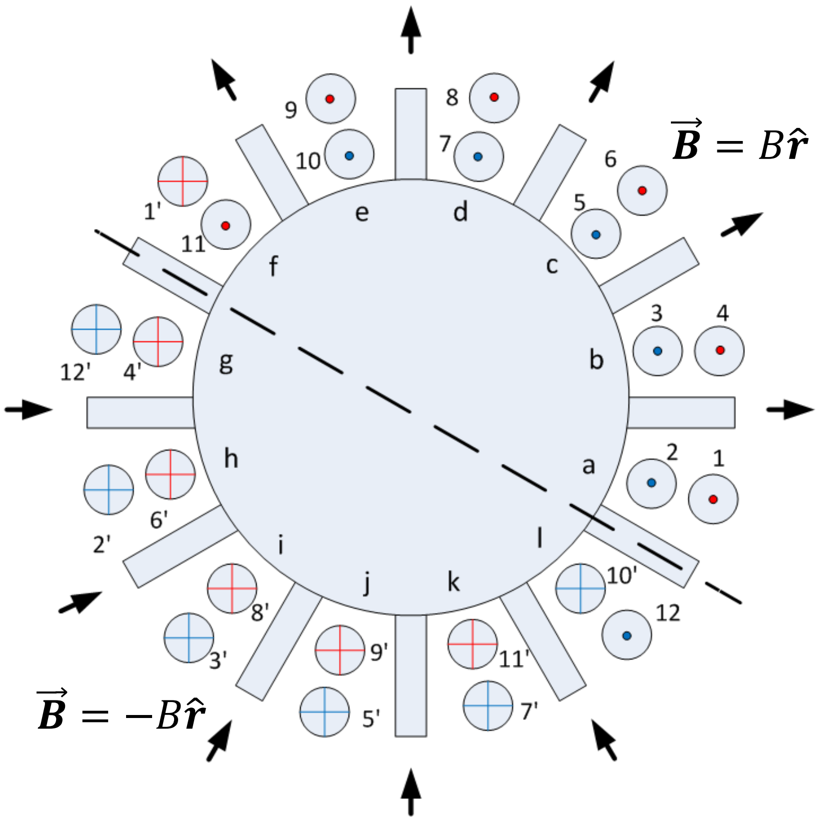


Figure 3-5: Wiper Motor Current Paths (Slow)

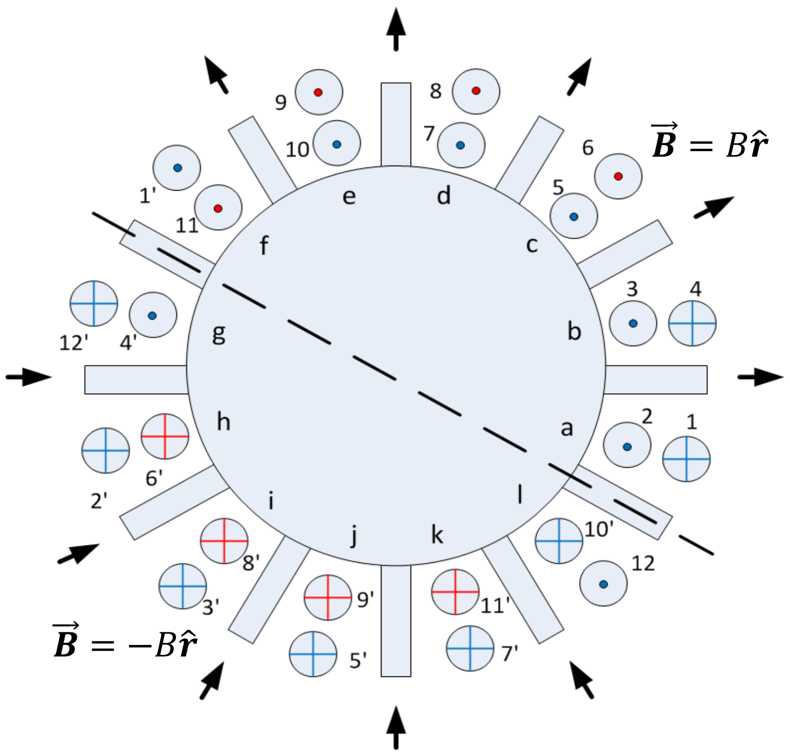


Figure 3-6: Wiper Motor Current Paths (Fast)

To analyse the torque produced by the motor, the torque developed in each armature slot is determined. There are three cases to consider:

1) The currents in the slot are equal and flow in the same direction.
2) The currents in the slot are equal and flow in opposite directions.
3) The currents in the slot are unequal and flow in the same direction.

Starting with case 1, such as that shown in slot a of Figure 3-5 with coils 1 and 2 residing in it, the torque can be derived as follows. First the force produced by coils 1 and 2 are derived:

$$\begin{aligned}\vec{F}_{coil1} &= i\vec{\mathbf{l}} \times \vec{\mathbf{B}} \\ &= il_1\hat{\mathbf{z}} \times B\hat{\mathbf{r}} \\ &= il_1 B\hat{\boldsymbol{\theta}}\end{aligned} \tag{3.1}$$

$$\begin{aligned}\vec{F}_{coil2} &= i\vec{\mathbf{l}} \times \vec{\mathbf{B}} \\ &= il_1\hat{\mathbf{z}} \times B\hat{\mathbf{r}} \ . \\ &= il_1 B\hat{\boldsymbol{\theta}}\end{aligned} \tag{3.2}$$

where $\vec{\mathbf{l}}$ is the length of the conducting coil in the direction of the current, l_1 is the height of the armature, i is the current in the coil and vectors $\hat{\mathbf{z}}$, $\hat{\mathbf{r}}$ and $\hat{\boldsymbol{\theta}}$ conform to the cylindrical coordinate system shown in Figure 3-7.

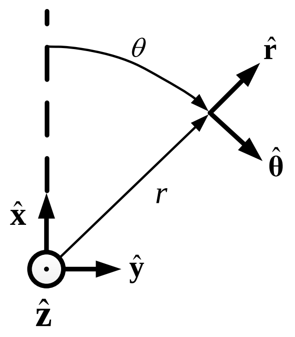


Figure 3-7: Cylindrical Coordinate System

Next, the torques produced by these forces are determined:

$$\begin{aligned}\vec{\mathbf{T}}_{coil1} &= \left(l_2/2\right)\hat{\mathbf{r}}\times\vec{F}_{coil1}\\ &= \left(l_2/2\right)il_1B\hat{\mathbf{r}}\times\hat{\boldsymbol{\theta}}\ ,\\ &= \left(l_2/2\right)il_1B\hat{\mathbf{z}}\end{aligned} \tag{3.3}$$

$$\begin{aligned}\vec{\mathbf{T}}_{coil2} &= \left(l_2/2\right)\hat{\mathbf{r}}\times\vec{F}_{coil2}\\ &= \left(l_2/2\right)il_1B\hat{\mathbf{r}}\times\hat{\boldsymbol{\theta}}\ ,\\ &= \left(l_2/2\right)il_1B\hat{\mathbf{z}}\end{aligned} \tag{3.4}$$

where l_2 is the diameter of the armature.

The total torque produced in the armature slot is simply:

$$\begin{aligned}\vec{\mathbf{T}}_a &= \vec{\mathbf{T}}_{coil1}+\vec{\mathbf{T}}_{coil2}\\ &= 2\left[\left(l_2/2\right)il_1B\hat{\mathbf{z}}\right]\\ &= l_1l_2Bi\hat{\mathbf{z}}\\ &= K_Ti\hat{\mathbf{z}}\end{aligned}\ , \tag{3.5}$$

where the torque constant for a single armature slot, K_t, is defined as

$$K_t = l_1l_2B\,. \tag{3.6}$$

For case 2, such as that shown in slot f of Figure 3-5, it can be seen that since the current in the two coils is travelling in the opposite direction but under the same direction of magnetic flux, the torque produced will be of equal value but in opposite directions and they will cancel each other out. Mathematically, in the case of slot f :

$$\begin{aligned}\vec{F}_{coil1'} &= i\vec{\mathbf{l}}\times\vec{\mathbf{B}}\\ &= -il_1\hat{\mathbf{z}}\times B\hat{\mathbf{r}}\\ &= -il_1B\hat{\boldsymbol{\theta}}\end{aligned} \quad \text{and} \quad \begin{aligned}\vec{F}_{coil11} &= i\vec{\mathbf{l}}\times\vec{\mathbf{B}}\\ &= il_1\hat{\mathbf{z}}\times B\hat{\mathbf{r}}\ ,\\ &= il_1B\hat{\boldsymbol{\theta}}\end{aligned}$$

and thus

$$\vec{\mathbf{T}}_f = \vec{\mathbf{T}}_{coil1'} + \vec{\mathbf{T}}_{coil11} = 0 \ . \tag{3.7}$$

For case 3 such as that shown in slot c of Figure 3-6 where the currents in the windings are in the same direction but with different magnitudes the torque produced in armature slot can be described by the following:

$$\vec{\mathbf{T}}_{coil5} = l_1 \left(l_2/2 \right) B i_1 \hat{\mathbf{z}} \ , \tag{3.8}$$

$$\vec{\mathbf{T}}_{coil6} = l_1 \left(l_2/2 \right) B i_2 \hat{\mathbf{z}} \ , \tag{3.9}$$

where i_1 and i_2 are the currents in current paths 1 and 2 respectively. Thus, the final torque produced in the armature slot is:

$$\vec{\mathbf{T}}_c = \vec{\mathbf{T}}_{coil5} + \vec{\mathbf{T}}_{coil6} = l_1 l_2 B \left(i_1 + i_2 \right) \hat{\mathbf{z}} \ . \tag{3.10}$$

An Expression for the torque produced by the motor in its slow and fast modes is now determined by summing the torque produced by each individual slot and using Figure 3-5 and Figure 3-6 to determine which of the three cases to use for each slot – depending on the current path pattern:

$$T_{slow} = T_a + T_b + \cdots T_l = 10 K_t i_{slow} , \tag{3.11}$$

$$T_{fast} = T_a + T_b + \cdots T_l = 8 K_t i_{fast} \ , \tag{3.12}$$

where T_{slow} and i_{slow}, and T_{fast} and i_{fast} are the torques and currents produced in the fast and slow operation, respectively, with i_{fast} equalling $\left(i_1 + i_2 \right)$ and i_{slow} equalling i because the currents in the two paths to ground are equal in the slow operation.

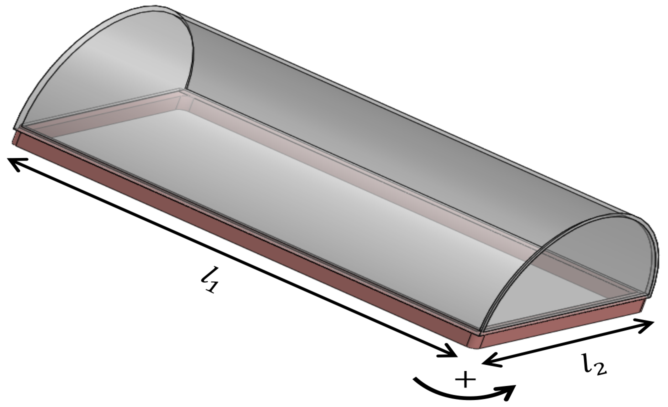
3.3.2 Back EMF

A similar analysis is now carried out for the back EMF produced by the motor in its slow and fast modes of operation. In this case, rather than analyse the system based on the separate armature slots, the armature winding loops are analysed.

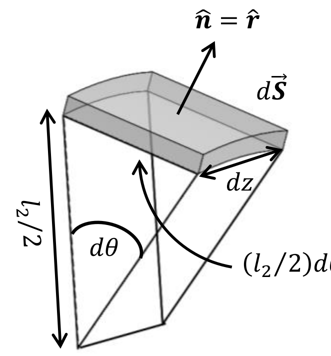
Figure 3-8a shows a single coil of wire in the armature with the transparent section representing the air gap of the motor between the armature windings and the permanent magnets. This can also be seen as the flux surface of the magnetic field, given the symbol S. The flux surface is approximately a half cylinder with a length/height of l_1 and diameter of l_2. The positive direction of travel around the coil is defined as being anticlockwise in accordance with vector $\hat{\mathbf{r}}$ pointing radially outwards from the armature centre and the right hand grip rule. The magnetic field in the air gap between the magnets and the winding is known to be approximately constant and radially directed with a magnitude of B. Thus an expression for the vector $\vec{\mathbf{B}}$ is given as:

$$\vec{\mathbf{B}} = \begin{cases} +B\hat{\mathbf{r}} & \textit{for} \quad 0 < \theta < \pi \\ -B\hat{\mathbf{r}} & \textit{for} \quad \pi < \theta < 2\pi \end{cases}, \tag{3.13}$$

where θ is the position of the rotor and is zero or π when in line with the magnetic neutral line.



(a)



(b)

Figure 3-8: Single Coil Flux Surface and Flux Surface Element

From Figure 3-8b an expression for the differential flux surface element $d\vec{\mathbf{S}}$ can be derived as:

$$d\vec{\mathbf{S}} = (l_2/2)\,d\theta dz\hat{\mathbf{r}} \tag{3.14}$$

Using the fact that the flux is defined as the rate of change of flux density, and equations (3.13) and (3.14), an expression for the flux in the air gap for $0 < \theta_R < \pi$ (i.e. when the flux density is positive), where θ_R is the position of the rotor, can be derived as follows. Beginning with the fact that

$$\phi(\theta_R) = \int_S \vec{\mathbf{B}} \cdot d\vec{\mathbf{S}}\ , \tag{3.15}$$

and then substituting in (3.14) to get

$$\phi(\theta_R) = \int_0^{l_1} \int_{\theta_R}^{\pi+\theta_R} \vec{\mathbf{B}} \cdot \left((l_2/2)\,d\theta dz\hat{\mathbf{r}}\right)\ . \tag{3.16}$$

Now using (3.13) the integral can be split into

$$\phi(\theta_R) = \int_0^{l_1} \int_{\theta_R}^{\pi} (B\hat{\mathbf{r}}) \cdot \left((l_2/2)\,d\theta dz\hat{\mathbf{r}}\right) + \int_0^{l_1} \int_{\pi}^{\pi+\theta_R} (-B\hat{\mathbf{r}}) \cdot \left((l_2/2)\,d\theta dz\hat{\mathbf{r}}\right), \tag{3.17}$$

which when evaluated leads to

$$\phi\left(\theta_R\right) = -l_1 l_2 B\left(\theta_R - \pi/2\right) \ . \tag{3.18}$$

Likewise for $\pi < \theta_R < 2\pi$ the flux can be shown to be

$$\phi\left(\theta_R\right) = l_1 l_2 B\left(\theta_R - 3\pi/2\right) \ . \tag{3.19}$$

The definition of Electromotive Force (EMF) states that the EMF produced in a loop is the rate of change of flux and seeks to resist current flow caused by the flux [95], the EMF, ξ, in a loop can now be defined as:

$$\xi = -\frac{d\phi}{dt} = \left(l_1 l_2 B\right)\frac{d\theta_R}{dt} = K_e \omega_R \ , \tag{3.20}$$

where ω_R is the angular velocity of the rotor and K_e is the EMF constant in a single loop and is equal to $l_1 l_2 B$.

If all of the coils conformed to the derivation above, the total EMF could be expressed by multiplying equation (3.20) by the number of coils, in this case 12. However this is not the case in a wiper motor. It can be seen from both Figure 3-5 and Figure 3-6 that the forward and return paths of two coils (1 and 12 in the diagram) lie under the same magnet, and thus the value of (3.13) is either always positive or always negative. The effect of this can be shown mathematically as follows. For coil 1:

$$\phi_{coil1}\left(\theta_R\right) = \int_S \vec{\mathbf{B}} \cdot d\vec{\mathbf{S}}$$

$$= \int_0^{l_1} \int_{\theta_R}^{\pi-\theta_R} \left(B\hat{\mathbf{r}}\right) \cdot \left(\left(l_2/2\right) d\theta dz \hat{\mathbf{r}}\right)$$

$$= \int_{0}^{l_1} \int_{\theta_R}^{\pi-\theta_R} B\left(l_2/2\right) d\theta dz \ , \tag{3.21}$$

which evaluates to

$$\phi_{coil1}\left(\theta_R\right) = -Bl_1 l_2 \left(\theta_R - \pi/2\right) \ . \tag{3.22}$$

Likewise for coil 12 it can be shown that

$$\phi_{coil12}\left(\theta_R\right) = \int_S \vec{\mathbf{B}} \cdot d\vec{\mathbf{S}}$$

$$= \int_{0}^{l_1} \int_{\pi+\theta_R}^{2\pi-\theta_R} \left(-B\hat{\mathbf{r}}\right) \cdot \left(\left(l_2/2\right) d\theta dz \hat{\mathbf{r}}\right)$$

$$= \int_{0}^{l_1} \int_{\pi+\theta_R}^{2\pi-\theta_R} -B\left(l_2/2\right) d\theta dz \ . \tag{3.23}$$

which evaluates to

$$\phi_{coil1}\left(\theta_R\right) = Bl_1 l_2 \left(\theta_R - \pi/2\right) \ . \tag{3.24}$$

When summed, the fluxes described by equations (3.22) and (3.24) and their subsequent EMFs will cancel out.

Finally, it can be seen that coil 4 in Figure 3-6 has a current flow opposite to the positive direction of current flow. Mathematically, this will cause the definition of $d\vec{\mathbf{S}}$ to become:

$$d\vec{\mathbf{S}} = -\left(l_2/2\right) d\theta dz \hat{\mathbf{r}} \ . \tag{3.25}$$

When the flux due to coil 4 is derived, it evaluates as

$$\phi_{coil4}\left(\theta_R\right) = Bl_1l_2\left(\theta_R - \pi/2\right) , \tag{3.26}$$

i.e. the opposite sign of a ‘standard’ coil. This means that when the EMFs per coil are summed, coil 4 in the fast operation will be subtracted from the total, not added.

The total EMFs for the fast and slow operations of the wiper motor are given respectively as:

$$\xi_{slow} = \xi_1 + \xi_2 + \cdots + \xi_{12} = 10K_e\omega_R , \tag{3.27}$$

$$\xi_{fast} = \xi_1 + \xi_2 + \cdots + \xi_{12} = 8K_e\omega_R \tag{3.28}$$

3.3.3 Resistance and Inductance

Figure 3-9 is used to understand how the resistance and inductance are affected by the slow and fast operation modes of the wiper motor. Resistors R_{Bin} and R_{Bout} are the resistors of the input and output brushes respectively and are assumed to remain constant. It can be seen that the current supplied by the battery has two possible paths to ground, i.e. current path one and current path two. In the motor’s slow operation, current paths one and two are approximately the same length and thus have the same resistance and inductance, meaning that $i_1 = i_2$. However, in the motor’s fast operation, current path two is physically shorter than current path one, meaning that its resistance and inductance is lower and $i_1 \neq i_2$. It is known that when two resistances or inductances are connected in parallel, the overall resistance/inductance will be lower than the lowest total resistance/inductance in a path. Thus in slow operation, the resistance and inductance will be higher than in fast operation.

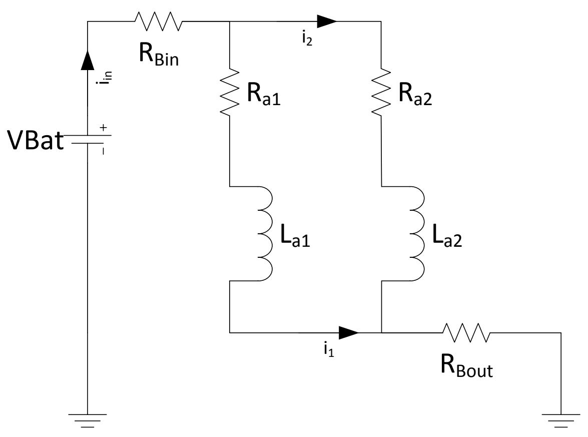


Figure 3-9: Wiper Motor Electrical Equivalent Circuit

3.4 Dynamic Equations

The dynamic equations of a PMDC motor are well known and can be stated as:

$$\begin{aligned} V - R_a i_a - L\frac{di_a}{dt} - K_E\frac{d\theta}{dt} = 0 \\ K_T i_a - J\frac{d^2\theta}{dt^2} - b\frac{d\theta}{dt} - T_L = 0 \end{aligned} , \tag{3.29}$$

where V is input voltage, R_a is armature resistance, i_a is armature current, L is armature inductance, K_E is the EMF constant, θ is the rotor position, K_T is the torque constant, J is the motor's inertia, b is the damping coefficient and T_L is the torque load.

Using equations (3.11), (3.12), (3.27) and (3.28) and the previous observations made in this chapter, the dynamic equations of a wiper motor can be determined. It is assumed that $K_E = K_T$, which is valid assuming the same units are used for both. The equations are:

$$\left.\begin{aligned} V - R_{slow} i_a - L_{slow}\frac{di_a}{dt} - K_{slow}\frac{d\theta}{dt} = 0 \\ K_{slow} i_a - J\frac{d^2\theta}{dt^2} - b\frac{d\theta}{dt} - T_L = 0 \end{aligned}\right\} \text{ for slow operation }, \tag{3.30}$$

$$\left.\begin{aligned} V - R_{fast} i_a - L_{fast}\frac{di_a}{dt} - K_{fast}\frac{d\theta}{dt} = 0 \\ K_{fast} i_a - J\frac{d^2\theta}{dt^2} - b\frac{d\theta}{dt} - T_L = 0 \end{aligned}\right\} \text{ for fast operation },\tag{3.31}$$

where

$$\begin{aligned} R_{slow} &> R_{fast} \\ L_{slow} &> L_{fast} \\ K_{slow} &> K_{fast} \end{aligned}\quad .\tag{3.32}$$

Hence the model is increased from a 5 parameter model to an 8 parameter model.

3.5 Simulation Models

Three simulation models implementing the equations derived above are now presented. The first is a state space model which is solved in the MATLAB workspace using the "lsim" function, the second is a physical model to be implemented in Simulink/Simscape and the third is a model developed in the Simulink tool Stateflow.

3.5.1 State Space Model

The state space implementation of equations (3.30) and (3.31) are respectively:

$$\begin{gathered} \frac{d}{dx}\begin{bmatrix} i \\ \omega \end{bmatrix} = \begin{bmatrix} -\dfrac{R_{slow}}{L_{slow}} & -\dfrac{K_{slow}}{L_{slow}} \\ \dfrac{K_{slow}}{J} & -\dfrac{b}{J} \end{bmatrix}\begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} \dfrac{1}{L_{slow}} & 0 \\ 0 & -\dfrac{1}{J * ratio} \end{bmatrix}\begin{bmatrix} V \\ T_L \end{bmatrix} \\ \begin{bmatrix} y1 \\ y2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1/ratio \end{bmatrix}\begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} V \\ T_L / ratio \end{bmatrix} \end{gathered}\tag{3.33}$$

$$\frac{d}{dx}\begin{bmatrix} i \\ \omega \end{bmatrix} = \begin{bmatrix} -\frac{R_{fast}}{L_{fast}} & -\frac{K_{fast}}{L_{fast}} \\ \frac{K_{fast}}{J} & -\frac{b}{J} \end{bmatrix}\begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L_{fast}} & 0 \\ 0 & -\frac{1}{J*ratio} \end{bmatrix}\begin{bmatrix} V \\ T_L \end{bmatrix}$$

$$\begin{bmatrix} y1 \\ y2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1/ratio \end{bmatrix}\begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} V \\ T_L / ratio \end{bmatrix} \tag{3.34}$$

where the constant term $ratio$ is the gear ratio of the worm and wheel connected to the motor shaft and the linkage crank. The torque load, T_L , is defined as the load applied by the mechanical element of the wiper system to the gears, thus the actual torque load seen by the motor will be reduced by a factor of $ratio$. Likewise, the speed of rotor is ω but the output speed of the entire motor system is reduced by a factor of $ratio$.

Using equations (3.33) and (3.34), the state space model can be defined in 6 matrices:

$$A_{slow} = \begin{bmatrix} -\frac{R_{slow}}{L_{slow}} & -\frac{K_{slow}}{L_{slow}} \\ \frac{K_{slow}}{J} & -\frac{b}{J} \end{bmatrix}, \quad B_{slow} = \begin{bmatrix} \frac{1}{L_{slow}} & 0 \\ 0 & -\frac{1}{J*ratio} \end{bmatrix}$$

$$A_{fast} = \begin{bmatrix} -\frac{R_{fast}}{L_{fast}} & -\frac{K_{fast}}{L_{fast}} \\ \frac{K_{fast}}{J} & -\frac{b}{J} \end{bmatrix}, \quad B_{fast} = \begin{bmatrix} \frac{1}{L_{fast}} & 0 \\ 0 & -\frac{1}{J*ratio} \end{bmatrix} \tag{3.35}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1/ratio \end{bmatrix}, \qquad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The state space model is now simulated to demonstrate its functionality. The m-file used to carry out this simulation can be found in Appendix A. The simulation switches the motor on in its slow operation at $t = 1s$, then switches to its fast operation at $t = 2s$ and finally switches off at $t = 3s$. Note that the park switch functionality is not implemented here. The simulation assumes that the motor parameters are:

Table 3-1: Example Motor Parameters

$R_{slow} = 8\Omega$	$R_{fast} = 3.9\Omega$
$L_{slow} = 9\times10^{-5} H$	$L_{fast} = 1.2\times10^{-5} H$
$K_{slow} = 9.55\times10^{-4} V/(rad/s)$	$K_{fast} = 6.88\times10^{-4} V/(rad/s)$
$J = 2\times10^{-9} kgm^2$	$b = 0.002\, Nm/(rad/s)$
$ratio = 1$	$T_L = 0$

The parameters are arbitrary but adhere to the inequalities in equation (3.32). The results of the simulation can be seen in Figure 3-10. It can be seen that in the motor's slow operation its angular velocity and current are lower than in the fast mode.

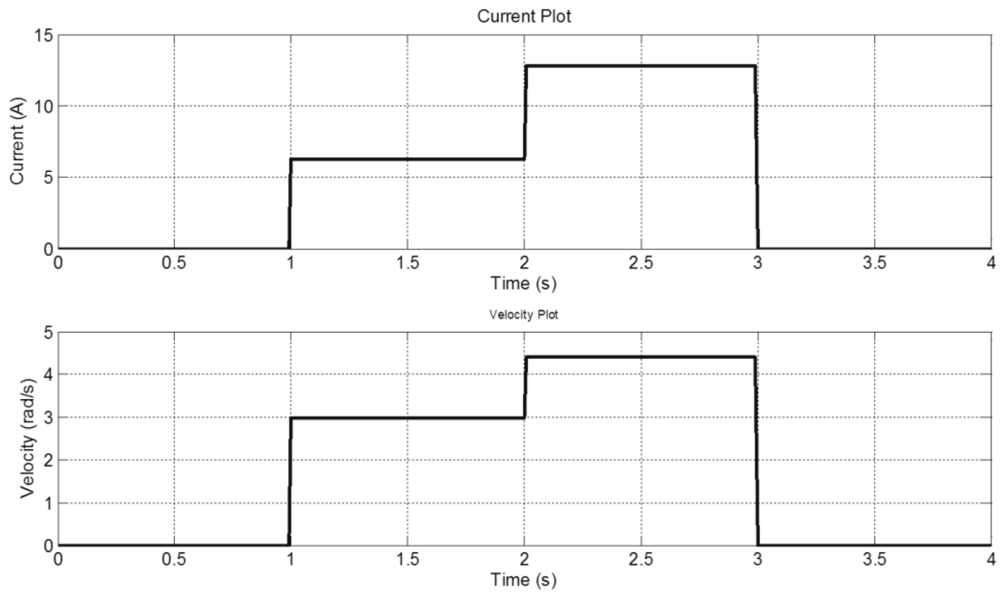


Figure 3-10: Motor State Space Model Simulation Results

3.5.2 Physical Model

To implement the model directly in Simulink a physical model has been used based on the DC motor model in SimElectronics. The DC motor model implements the equations shown in (3.29). In order to model the two input speed control of the wiper motor, two DC motor models were connected as shown in Figure 3-11. The figure demonstrates that the mechanical characteristics of the motor are shared, and thus connected together, and the electrical characteristics are separate. A gear box is connected to the mechanical output to simulate the worm and wheel gears. As is the strategy of physical modelling, the I/O of the model is the same as for a real wiper motor. The model is simulated under the same conditions as the state space model and the results are shown in Figure 3-12. As expected the results are the same as those shown in Figure 3-10

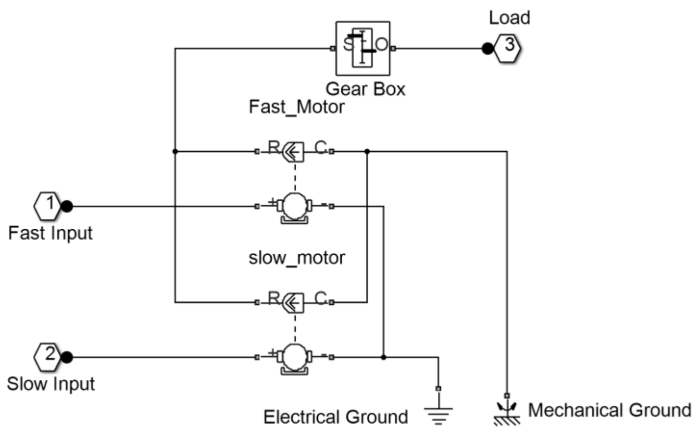


Figure 3-11: Wiper Motor Physical Model

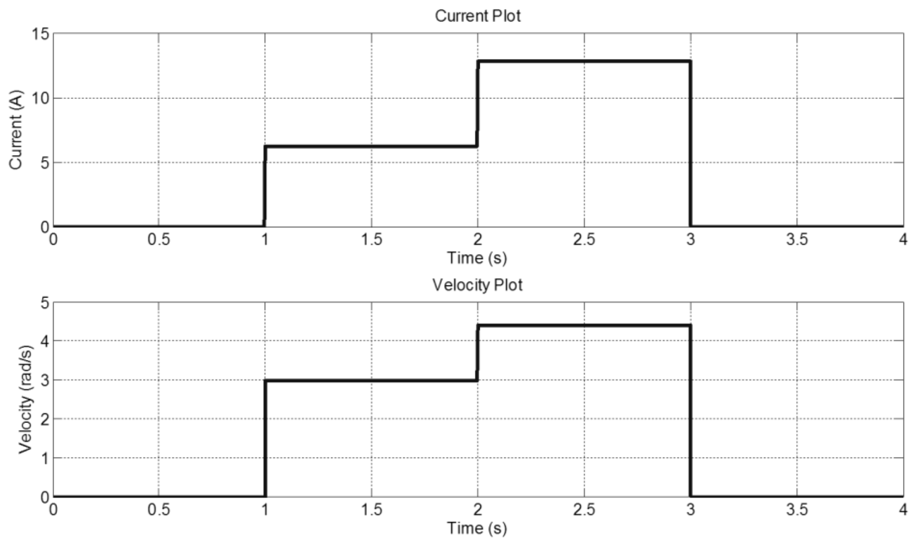
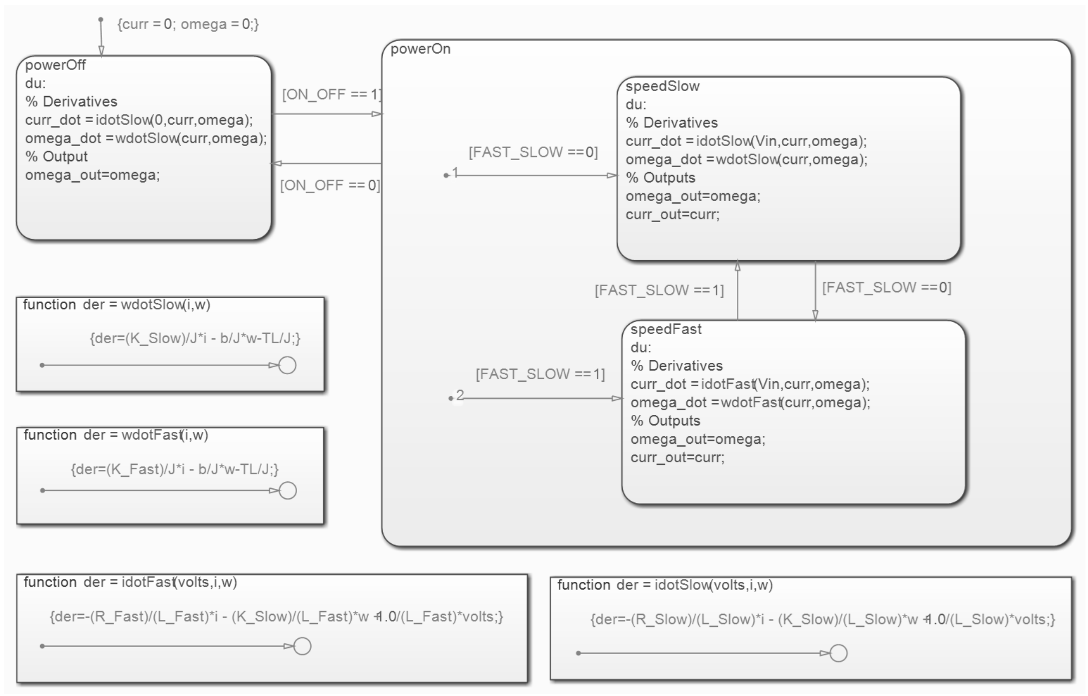


Figure 3-12: Motor Physical Model Simulation Results

3.5.3 Stateflow Model

The fact that the wiper motor could be seen as having two states, i.e. fast and slow, means that it lends itself comfortably to being modelled in the Simulink expansion tool Stateflow. This implementation is shown in Figure 3-13. The default state of the system (called powerOff) sets, on entry, the current and position of the motor to 0 radians. The functions “idotslow” and “omegadotslow” are then called to calculate the change in current and the speed of the motor, respectively. When the motor is switched on, the system enters its second main state (called powerOn) which has two sub states called speedSlow and speedFast. Which of these substates is entered is determined by the state of the FAST_SLOW control signal. In both states, the change in the current and the speed of the motor are calculated by calling functions. In the “speedSlow” state the functions “idotslow” and “omegadotslow” are used and in the “speedFast” state the “idotfast” and “omegadotfast” functions are used. The equations are the same in both the fast and slow cases but the parameters are different, capturing the behaviour of the wiper motor.



```
{curr = 0; omega = 0;}

powerOff
du:
% Derivatives
curr_dot = idotSlow(0,curr,omega);
omega_dot =wdotSlow(curr,omega);
% Output
omega_out=omega;

[ON_OFF == 1]
[ON_OFF == 0]

powerOn

speedSlow
du:
% Derivatives
curr_dot = idotSlow(Vin,curr,omega);
omega_dot =wdotSlow(curr,omega);
% Outputs
omega_out=omega;
curr_out=curr;

[FAST_SLOW ==0]
[FAST_SLOW ==1]
[FAST_SLOW ==0]
[FAST_SLOW ==1]

speedFast
du:
% Derivatives
curr_dot = idotFast(Vin,curr,omega);
omega_dot =wdotFast(curr,omega);
% Outputs
omega_out=omega;
curr_out=curr;

function der = wdotSlow(i,w)
{der=(K_Slow)/J*i - b/J*w-TL/J;}

function der = wdotFast(i,w)
{der=(K_Fast)/J*i - b/J*w-TL/J;}

function der = idotFast(volts,i,w)
{der=-(R_Fast)/(L_Fast)*i - (K_Slow)/(L_Fast)*w +1.0/(L_Fast)*volts;}

function der = idotSlow(volts,i,w)
{der=-(R_Slow)/(L_Slow)*i - (K_Slow)/(L_Slow)*w +1.0/(L_Slow)*volts;}
```

Figure 3-13: Wiper Motor Stateflow Implementation

This implementation produces the same results as the state space and physical models, however the simulation speed is significantly slower and thus this method of modelling dynamic systems will not be given further consideration.

3.6 Wiper Motor Park Switch Strategies

In normal operation, even if the wipers are turned off by the driver they will continue to wipe until they reach their park position, which is when the blades reach the end of their wipe cycle at the bottom of the windscreen as shown in Figure 3-14.

The method of stopping the wiper motor in the park position varies depending on the particular system being used; however they all involve the use of the park switch. The park switch can perform two functions. The first is to provide position feedback of the wipers in the form of a pulse when the wipers reach their park position. The second, in some wiper systems, is to route current from the battery to the motor when the wipers have been switched off until the park position is reached. Three main methods of achieving this are explained and modelled subsequently.

1) Mechanical Park Switch: A switch that provides a current path to the motor when the main switching relays controlled by the ECU have been switched off

2) Depressed Mechanical Park Switch: Plays a similar role to the mechanical park switch described above but is used in wiper systems in which, when the park position is reached, reverses the motor for a period of time to recess the wiper blades.

3) Digitally Controlled Park Switch: The park switch only provides positional feedback to the controller, and it is the controller that dictates when the wipers stop.

The switching strategies and subsequent simulation models for the three wiper motor systems described above will now be given in more detail.

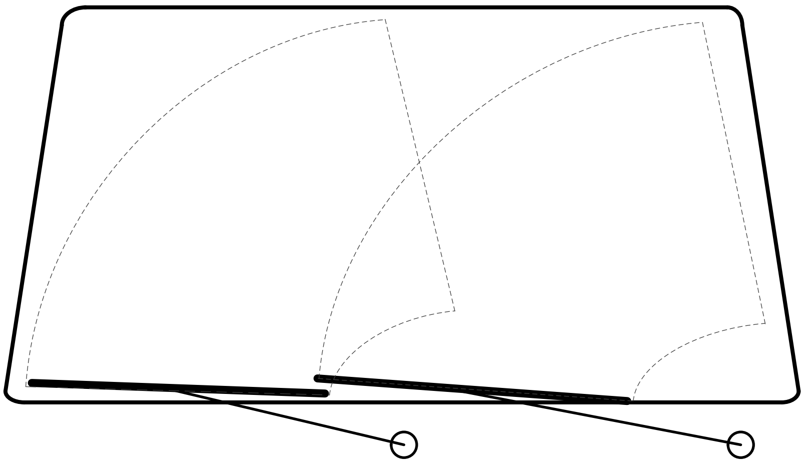
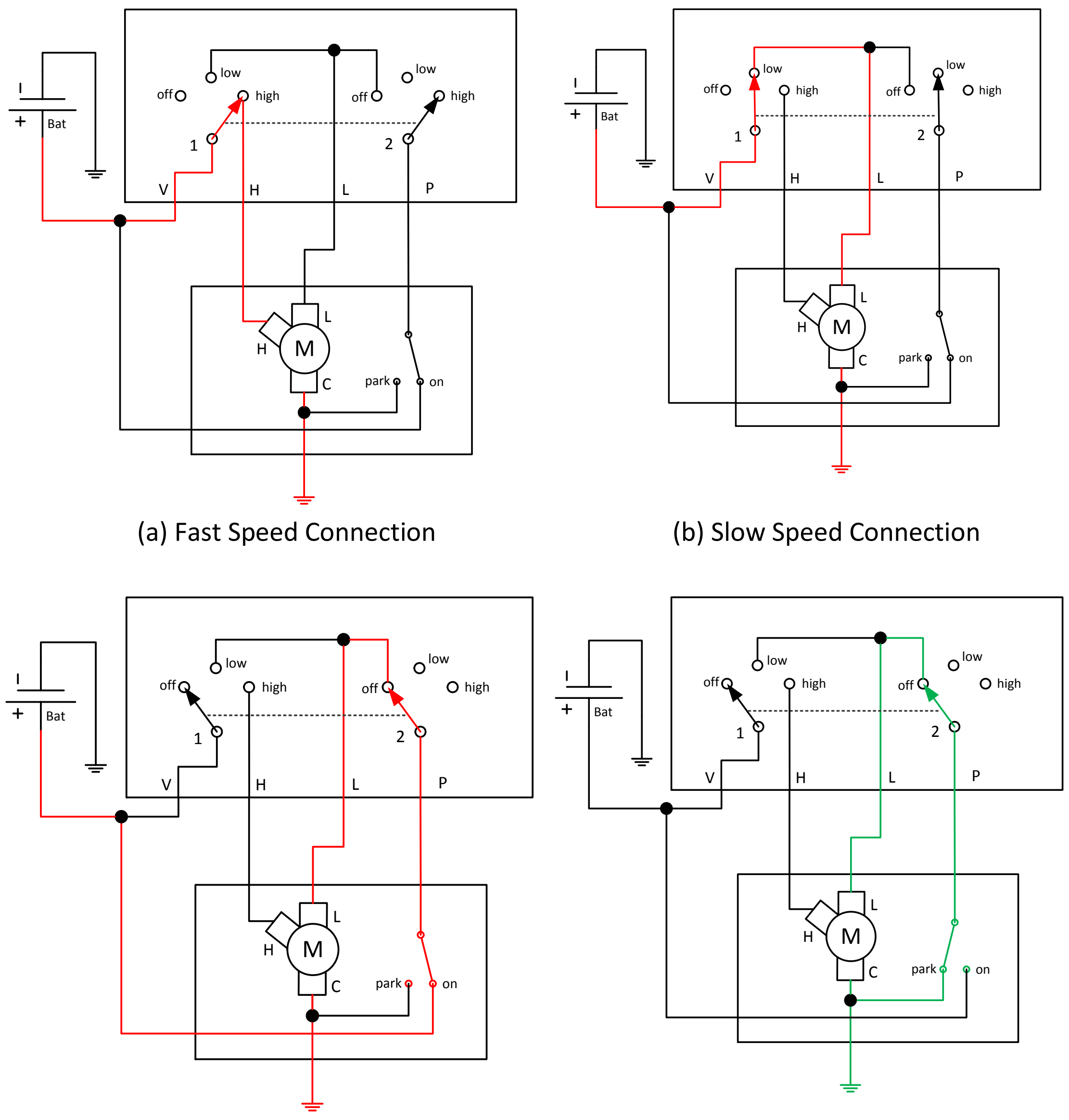


Figure 3-14: Wipers in Park Position

3.6.1 Mechanical Park Switch

The switching strategy for the mechanical park switch is shown in Figure 3-15. In high speed or low speed mode, current from the battery is routed through switch 1 to the high or low speed motor input respectively and reaches ground through the common motor connection (Figure 3-15 (a) and (b)). If the wiper is switched off, the mutually controlled switches 1 and 2 immediately switch to their off positions (Figure 3-15 (c)). This causes current from the battery to be routed through the park switch in the motor, through switch 2 and thus through the slow speed motor input and to ground through the common motor connection. When the wipers reach their park position the park switch changes state (Figure 3-15 (d)). This disconnects the battery from the motor and connects both the slow and common motor connections to ground; thus providing regenerative breaking.



(a) Fast Speed Connection

(b) Slow Speed Connection

(c) Off, Not Parked Connection

(d) Off, Parked Connection

Figure 3-15: Mechanical Park Switch Strategy

A physical modelling implementation of the system in Simulink and Simscape is shown in Figure 3-16, along with the control signals and switching relays.

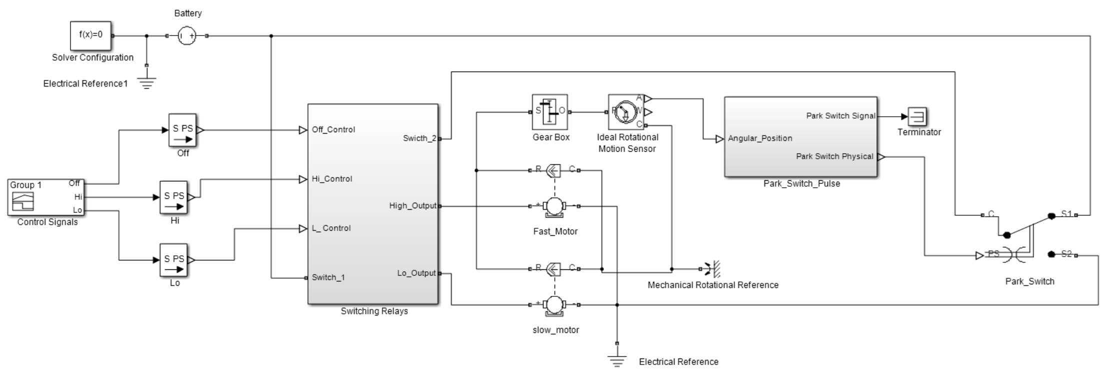


Figure 3-16: Mechanical Park Switch Strategy Simulation Model

The simulation results of the model shown in Figure 3-16 are shown in Figure 3-17. It can be seen that at time 0 to 2s, the control signals dictate that the motor runs in its slow speed. Then the speed switches to fast at 2s up to 4s. The angular velocity of the motor seen in Figure 3-17 demonstrates this functionality. At time 4s, the motor is switch off. It can be seen that the motor enters its slow speed until the park switch fires at around 5.3s. At this point the park switch changes state and the motor immediately stops.

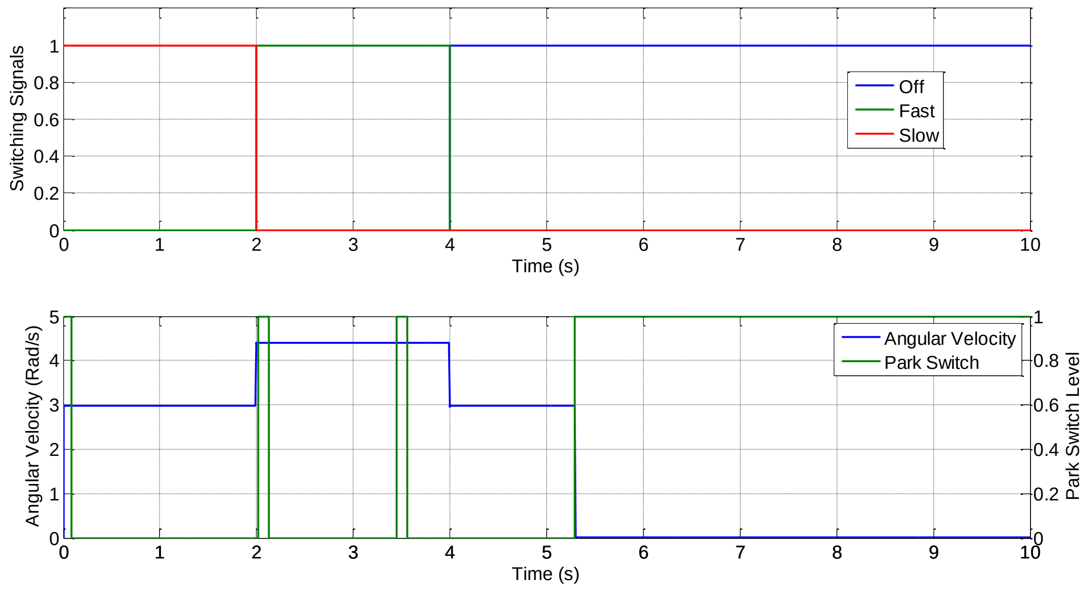


Figure 3-17: Mechanical Park Switch Strategy Simulation Results

The switching configuration for the control system is shown in Figure 3-18.

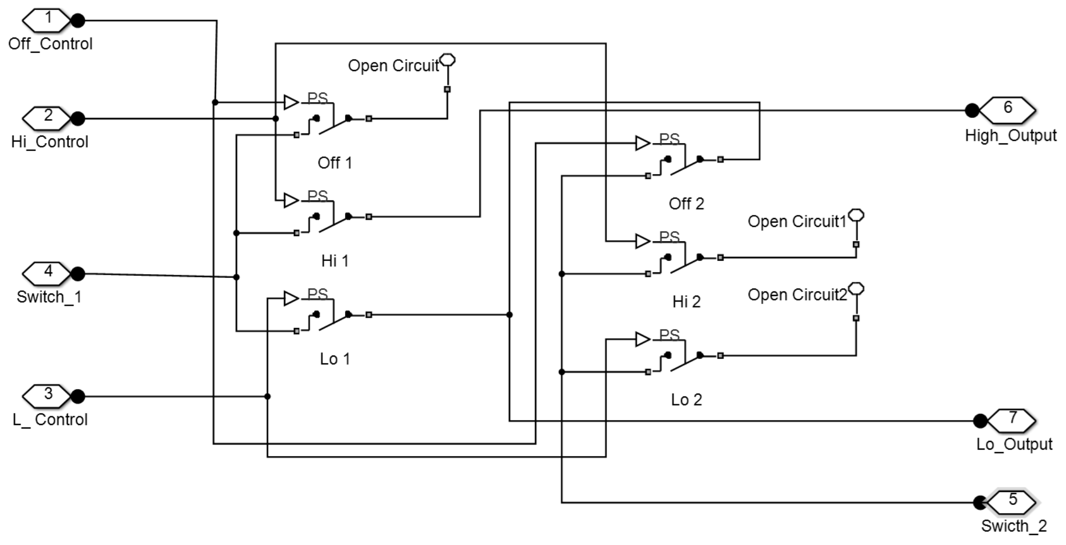
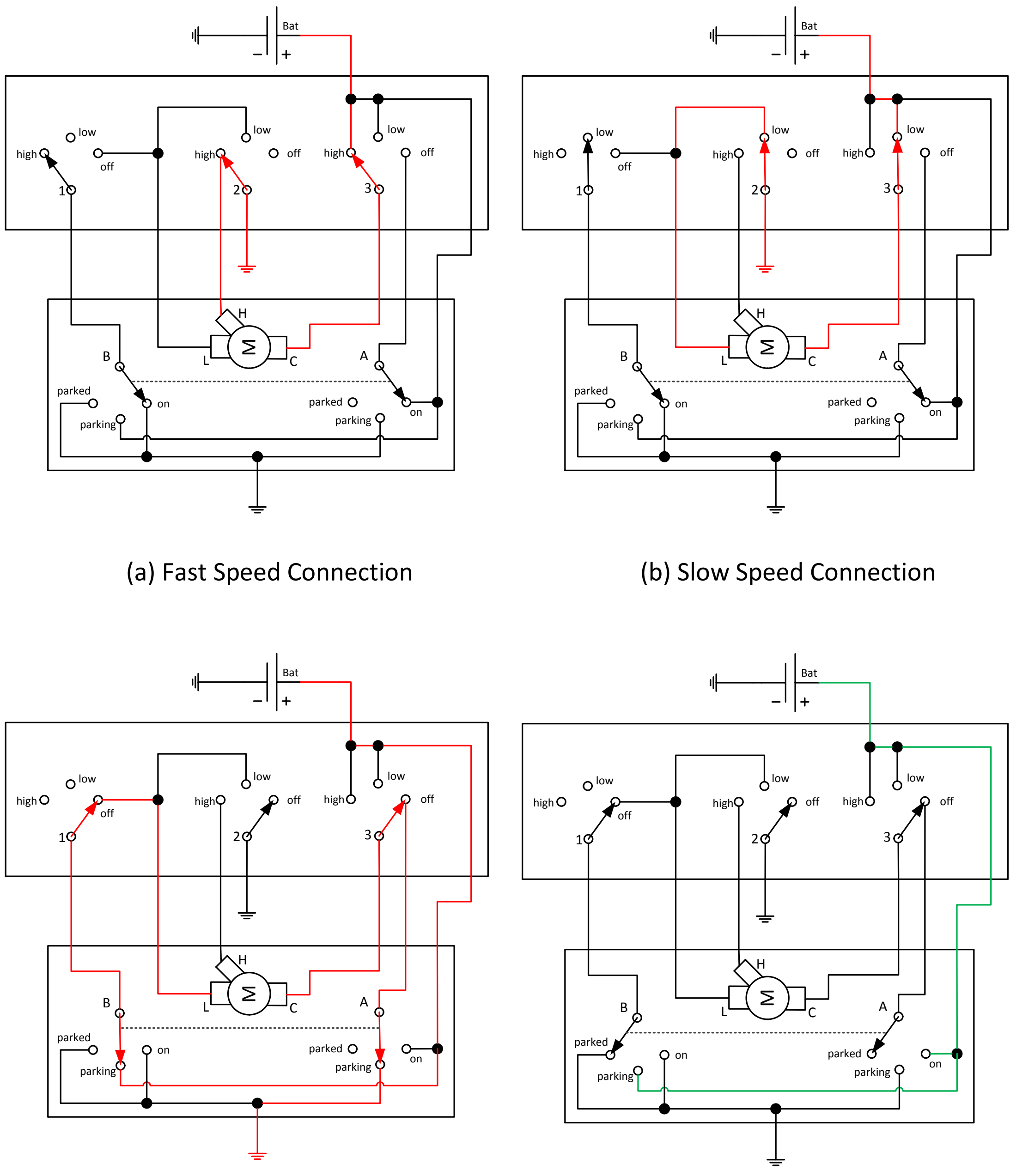


Figure 3-18: Mechanical Park Switch Switching Configuration

3.6.2 Depressed Mechanical Park Switch

The switching of the depressed mechanical park switch system is shown in Figure 3-19. Control switches 1, 2 and 3 are mutually controlled, as are both park switches: A and B. When the motor is in its slow or fast mode, the current from the battery is conducted through switch 3, through the common input of the motor and out the slow or fast motor connection, depending on the state of switch 2, through to ground (Figure 3-19 (a) and (b)). If the windscreen wipers are turned off, switches 1, 2 and 3 enter their off positions and the park switches A and B remain in their “on” position. This conducts the current from the battery through park switch A and then control switch 3 to the common connection of the motor through to the low speed connection and then to ground through switch 2. When the wipers reach their park position, switches A and B change to their parking state (Figure 3-19 (c)). This causes current from the battery to be conducted via park switch B and control switch 1, to the slow connection of the motor. Current is then conducted to ground through the common connection of the motor, through switches 3 and A. In this configuration, the motor runs in reverse with respect to its normal operation. Once the motor has rotated a certain distance in reverse, park switches A and B switch to their "parked" position (Figure 3-19 (d)). This disconnects the motor from the battery.



(a) Fast Speed Connection

(b) Slow Speed Connection

(c) Reversal Connection

(d) Off, Parked Connection

Figure 3-19: Depressed Mechanical Park Switch Strategy

A physical modelling implementation of the system in Simulink and Simscape is shown in Figure 3-20, along with the control signals and switching relays.

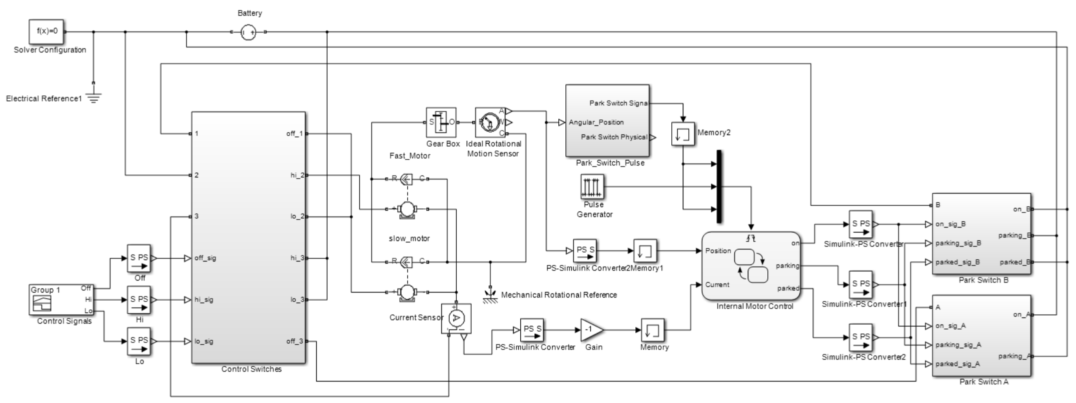


Figure 3-20: Depressed Mechanical Park Switch Strategy Simulation Model

The internal logic of the wiper motor controlling the park switches is modelled using Stateflow and is shown in Figure 3-21.

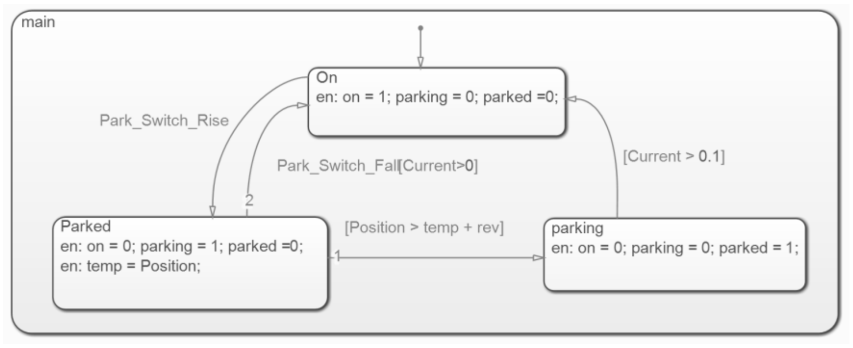

main
On
en: on = 1; parking = 0; parked =0;
Park_Switch_Rise
Park_Switch_Fall[Current>0]
[Current > 0.1]
2
Parked
en: on = 0; parking = 1; parked =0;
en: temp = Position;
[Position > temp + rev]
1
parking
en: on = 0; parking = 0; parked = 1;

Figure 3-21: Depressed Mechanical Park Switch motor Internal Logic

Simulation results of the model shown in Figure 3-20 are shown in Figure 3-22. The upper plot shows that at times 0s to 2s, the motor is in its slow operation and at time 2s to 4s, the motor is in its fast operation. This can be seen by the angular velocity of the motor in the lower plot. At time 4s the wiper system is switched off and the motor changes to its slow operation. At around 5.25s, the park switch triggers and the motor begins to operate in reverse, as can be seen in the angular velocity in the lower plot. After the rotor has rotated a certain distance (i.e. the variable “rev”) at around 5.5s, the motor switches off.

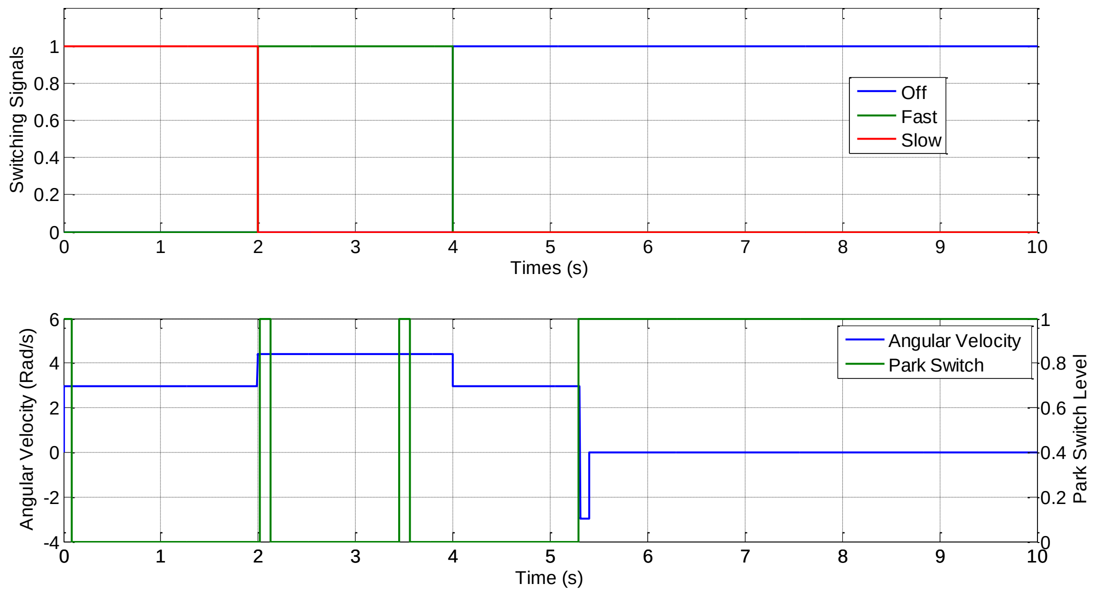


Figure 3-22: Depressed Mechanical Park Switch Strategy Simulation Results

The switching configuration for the control system is shown in Figure 3-23.

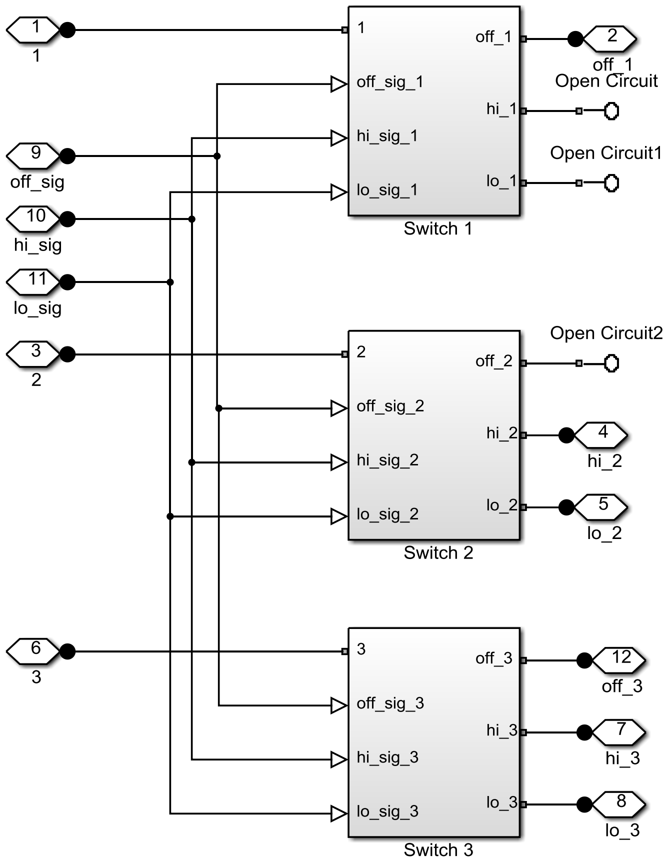


Figure 3-23: Depressed Mechanical Park Switch Switching Configuration

3.7 Discussion

The unique structure of a wiper motor allows it to operate in two speeds, slow and fast, without changing the input voltage. The two speed operation is achieved by using two input brushes, one being in line with the magnetic neutral line and used for the slow operation, and one offset from the magnetic neutral line used to speed up the angular velocity of the motor. The increase in speed is due to a decrease in armature resistance and magnetic flux. The two speed behaviour can be modelled using the dynamic equations of a PMDC motor with three added parameters to account for the lowered armature resistance, inductance and EMF/torque constant of the motor when operating in its fast mode.

Three modelling techniques were used to model the wiper motor: A state space approach implemented in MATLAB, a physical modelling approach implemented in Simscape (Simulink) and a state-flow model implemented in Stateflow (Simulink). All approaches can simulate the motor successfully with the state space model being the fastest to solve but most difficult to integrate with other models, the stateflow model being the slowest to simulate but easily handling the speed switching operation and the physical model being the easiest to integrate with other models, i.e. the linkages and control systems, whilst maintaining fast simulation speeds. The switching strategies to control the motor and stop it when the wipers are in their park position have been implemented in Simulink and Simscape and integrated with the physical model of the wiper motor successfully.

The model designed here is validated against real data in Chapter 5 and can be integrated with the mechanical models developed in Chapter 4 and the control systems developed in Chapter 7.

Chapter 4 - Multibody Dynamics Mechanical Modelling

4.1 Introduction

It can be seen in Figure 1-4 that the wiper motor whose model was developed in the previous chapter drives a mechanical system consisting of a set of linkages, two wiper arms and two wiper blades. The design of the linkages can vary depending on the vehicle model but is generally a 6 bar linkage system which can be approximated as acting in a 2-dimensional plane. The arms connect the oscillating rockers of the linkages to the wiper blades and provide a downwards force on the blades with respect to the windscreen. The blades remove water and debris from the windscreen using a rubber contact. The combination of the inertia and external forces acting on these mechanical elements provide the torque load to the wiper motor. The forces acting on the system are dominated by the friction between the blade rubber and the windscreen, and the aerodynamic drag and lift forces on the blades caused by the motion of the vehicle. A Solidworks representation of the wiper system's mechanical elements is given in Figure 4-1.

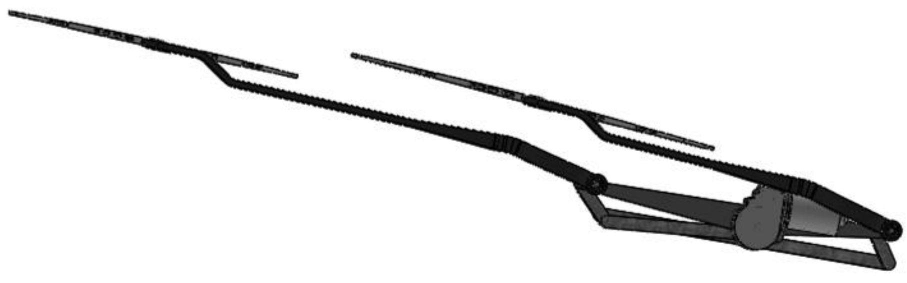


Figure 4-1: Solidworks Model of the Wiper System

In this chapter, the aforementioned mechanical elements are modelled and simulated. Firstly, a planar kinematic model of the linkages, arms and blades are developed using computational dynamics methods. The model only considers the dimensions of the systems, i.e. it ignores inertial and external forces, and is used to ensure that the dimensional parameters of the model are feasible and to give an initial indication of the model dynamics. Secondly, a full physical model of the mechanical system is developed in SimMechanics, taking into account inertial forces. Finally, the friction and aerodynamic forces acting on the blades are modelled.

To ensure that the model is flexible, a modular design method is chosen. This means that each element of the system, i.e. the linkages, arms, blades, friction and aerodynamic forces, must be modelled independently and must be able to connect to each design of its interfacing elements. Also, the fact that the model is to be used in real time simulation presents a challenge when modelling the friction and aerodynamic forces. It was shown in Chapter 2 that, in general, complex analytical or FEA models are used to model friction and aerodynamic forces acting on wiper blades. Advanced experimental rigs are also needed to validate such models, which were not available in this project. For these reasons, simplifications to these forces must be made based on their dominant elements derived from the literature.

The validation against real data of the models developed in this chapter is shown in Chapter 5.

4.1.1 Generic Modelling Strategy

For both the planar kinematic and full physical model, the same modelling methodology, in terms of user input, is used. The user defines a set of *design points,* from which the entire geometry of the system is calculated. The nature of the design points are given in more detail in their respective sections, however in general they are the 3-dimensional Cartesian coordinates of the position of the joints in the system. The algorithm to calculate the body lengths, initial angles and centre of gravity (CG) position (i.e. the parameters needed to solve to system) can be found in Appendix B.

4.2 Planar Kinematics Model based on Computational Dynamics

In this section, planar kinematic models of the linkages, blades and arms are developed separately and then combined to generate the full mechanical system. Three types of linkage system have been modelled in this project, however only one is shown in this chapter. The equations and diagrams of the other two linkage systems are given in Appendix C. The analyses given here are based on the theory shown in [97] and [98].

4.2.1 Slave Driven Linkage System

Figure 4-2 shows the slave driven 6-bar planar linkage system to be modelled. The design points of the system are the positions of the revolute joints and are called O_link, A_link, … , F_link (hence forth referred to as O, A, …, F). The rigid bodies in the system are labelled 1 to 6, with body 1 being the base and having a stationary coordinate system (CS) with +X being horizontally right, +Y being vertically upwards and +Z (rotational axis of the joints) being out of the page. Design point O is attached to the rotor of the wiper motor, whose angle is defined as θ^2, i.e. the angle between body 2 and the +X axis of the stationary CS, rotated anticlockwise.

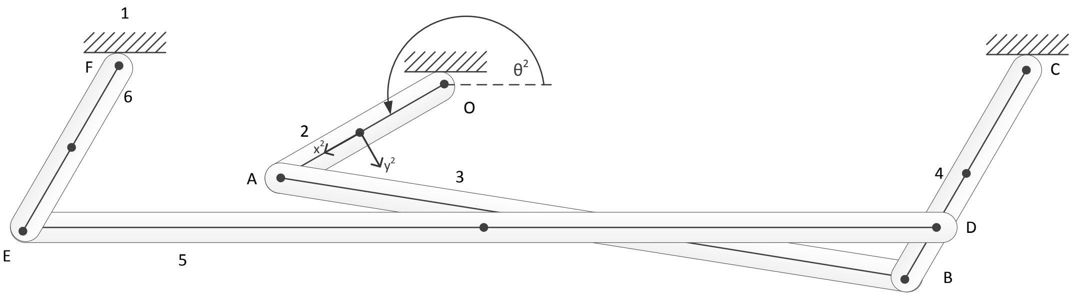


Figure 4-2: Slave Driven Linkage System Diagram

Each body has three coordinates associated with it. For an arbitrary body i the three coordinates are R_x^i , R_y^i and θ^i , respectively denoting the position of the body's centre of mass (CM) on the X and Y plane, and the angle by which the body's CS is rotated anticlockwise from the global +X axis as shown in Figure 4-2. The coordinate vector for the entire 6-body linkage system is given as

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}^1 & \mathbf{q}^2 & \mathbf{q}^3 & \mathbf{q}^4 & \mathbf{q}^5 & \mathbf{q}^6 \end{bmatrix}^T , \tag{4.1}$$

where

$$\mathbf{q}^i = \begin{bmatrix} \mathbf{R}^i & \theta^i \end{bmatrix}^T \qquad i = 1, 2, \ldots, 6, \tag{4.2}$$

and

$$\mathbf{R}^i = \begin{bmatrix} R_x^i & R_y^i \end{bmatrix}^T . \tag{4.3}$$

The absolute position of an arbitrary point on an arbitrary body, i.e. its position with respect to a global CS, is given as

$$\mathbf{r}_p^i = \mathbf{R}^i + \mathbf{A}^i \bar{\mathbf{u}}_p^i , \tag{4.4}$$

where $\mathbf{A}^i$ is the 2-dimenstional transformation matrix relating body CS i to the global CS and is given as

$$\mathbf{A}^i = \begin{bmatrix} \cos\theta^i & -\sin\theta^i \\ \sin\theta^i & \cos\theta^i \end{bmatrix} , \tag{4.5}$$

and $\bar{\mathbf{u}}_p^i$ is the local position vector, i.e. defined in the bodies CS, of an arbitrary point on the body (in this case always one of the design points) with respect to the body CM and is given as

$$\bar{\mathbf{u}}_p^i = \begin{bmatrix} \bar{x}_p^i & \bar{y}_p^i \end{bmatrix}^T . \tag{4.6}$$

In this case, the constraints on the motion of the system are caused by the joints between the linkages. The joints are all approximated as revolute joints. In reality many of the joints are spherical to add flexibility to the linkages, however functionally they operate as revolute joints and thus the approximation is considered valid. When considered in two dimensions, a revolute joint will remove two degrees of freedom from the attached bodies, i.e. the x and y position of the bodies in space relative to each other, and leave one degree of freedom, i.e. the rotational coordinate in the z-axis direction. Applying this definition, the constraint equations of the system in Figure 4-2 are given as

$$\begin{gathered}\mathbf{R}^1+\mathbf{A}^1\bar{\mathbf{u}}_O^1-\mathbf{R}^2-\mathbf{A}^2\bar{\mathbf{u}}_O^2=\mathbf{0}\\ \mathbf{R}^2+\mathbf{A}^2\bar{\mathbf{u}}_A^2-\mathbf{R}^3-\mathbf{A}^3\bar{\mathbf{u}}_A^3=\mathbf{0}\quad \mathbf{R}^3+\mathbf{A}^3\bar{\mathbf{u}}_B^3-\mathbf{R}^4-\mathbf{A}^4\bar{\mathbf{u}}_B^4=\mathbf{0}\\ \mathbf{R}^4+\mathbf{A}^4\bar{\mathbf{u}}_C^4-\mathbf{R}^1-\mathbf{A}^1\bar{\mathbf{u}}_C^1=\mathbf{0}\quad \mathbf{R}^5+\mathbf{A}^5\bar{\mathbf{u}}_D^5-\mathbf{R}^4-\mathbf{A}^4\bar{\mathbf{u}}_D^4=\mathbf{0}\\ \mathbf{R}^5+\mathbf{A}^5\bar{\mathbf{u}}_E^5-\mathbf{R}^6-\mathbf{A}^6\bar{\mathbf{u}}_E^6=\mathbf{0}\quad \mathbf{R}^6+\mathbf{A}^6\bar{\mathbf{u}}_F^6-\mathbf{R}^1-\mathbf{A}^1\bar{\mathbf{u}}_F^1=\mathbf{0}\end{gathered} \quad . \tag{4.7}$$

In addition, the coordinates of body 1 can be seen as being stationary, and are thus

$$R_x^1 = R_y^1 = \theta^1 = 0 \ . \tag{4.8}$$

Finally, for kinematic analysis, a driving constraint is applied representing the motion of the crankshaft angle, i.e. θ^2 , due to the motor shaft. The driving constraint is

$$\theta^2 - \omega^2 t - \theta_0^2 = 0 \ , \tag{4.9}$$

where ω^2 is the angular velocity of the crank shaft and θ_0^2 is its initial angular position.

The values of the local position vectors are given as

$$\begin{array}{lll} \bar{\mathbf{u}}_O^1 = \begin{bmatrix} 0 & 0 \end{bmatrix}^{\mathrm{T}} & \bar{\mathbf{u}}_O^2 = \begin{bmatrix} -l^2/2 & 0 \end{bmatrix}^T & \bar{\mathbf{u}}_A^2 = \begin{bmatrix} l^2/2 & 0 \end{bmatrix}^T \\ \bar{\mathbf{u}}_A^3 = \begin{bmatrix} -l^3/2 & 0 \end{bmatrix}^T & \bar{\mathbf{u}}_B^3 = \begin{bmatrix} l^3/2 & 0 \end{bmatrix}^T & \bar{\mathbf{u}}_B^4 = \begin{bmatrix} -l^4/2 & 0 \end{bmatrix}^T \\ \bar{\mathbf{u}}_C^4 = \begin{bmatrix} l^4/2 & 0 \end{bmatrix}^T & \bar{\mathbf{u}}_D^5 = \begin{bmatrix} -l^5/2 & 0 \end{bmatrix}^T & \bar{\mathbf{u}}_D^4 = \begin{bmatrix} -\left(l^7 - l^4/2\right) & 0 \end{bmatrix}^T \\ \bar{\mathbf{u}}_E^5 = \begin{bmatrix} l^5/2 & 0 \end{bmatrix}^T & \bar{\mathbf{u}}_E^6 = \begin{bmatrix} -l^6/2 & 0 \end{bmatrix}^T & \bar{\mathbf{u}}_F^6 = \begin{bmatrix} l^6/2 & 0 \end{bmatrix}^T \end{array} , \qquad (4.10)$$

where l^i for $i = 1 \to 6$ are the lengths of the respective bodies and l^7 is the distance between joints C and D.

The first step in solving the kinematic constraint equations given in equations (4.7) to (4.10) is to use them to form a vector of constraint equations, this vector is shown in equation (4.11). Note that the number of bodies in the system is equal to 6, i.e. $n_b = 6$ and the number of system coordinates is 18, i.e. $n = n_b \times 3 = 18$. In this case the number of constraint equations n_c is also 18, i.e. $n_c = n$. This is the definition of a kinematically driven system in which the positions, velocities and accelerations of the system's bodies can be determined without consideration of forces.

$$\mathbf{C}(\mathbf{q},t)=\begin{bmatrix} C_1(\mathbf{q},t)\\ C_2(\mathbf{q},t)\\ C_3(\mathbf{q},t)\\ C_4(\mathbf{q},t)\\ C_5(\mathbf{q},t)\\ C_6(\mathbf{q},t)\\ C_7(\mathbf{q},t)\\ C_8(\mathbf{q},t)\\ C_9(\mathbf{q},t)\\ C_{10}(\mathbf{q},t)\\ C_{11}(\mathbf{q},t)\\ C_{12}(\mathbf{q},t)\\ C_{13}(\mathbf{q},t)\\ C_{14}(\mathbf{q},t)\\ C_{15}(\mathbf{q},t)\\ C_{16}(\mathbf{q},t)\\ C_{17}(\mathbf{q},t)\\ C_{18}(\mathbf{q},t) \end{bmatrix}=\begin{bmatrix} R_x^1\\ R_y^1\\ \theta^1\\ R_x^2-\frac{l^2}{2}\cos\theta^2\\ R_y^2-\frac{l^2}{2}\sin\theta^2\\ R_x^2+\frac{l^2}{2}\cos\theta^2-R_x^3+\frac{l^3}{2}\cos\theta^3\\ R_y^2+\frac{l^2}{2}\sin\theta^2-R_y^3+\frac{l^3}{2}\sin\theta^3\\ R_x^3+\frac{l^3}{2}\cos\theta^3-R_x^4+\frac{l^4}{2}\cos\theta^4\\ R_y^3+\frac{l^3}{2}\sin\theta^3-R_y^4+\frac{l^4}{2}\sin\theta^4\\ R_x^4+\frac{l^4}{2}\cos\theta^4-X^1\\ R_y^4+\frac{l^4}{2}\sin\theta^4-Y^1\\ R_x^5-\frac{l^5}{2}\cos\theta^5-R_x^4+\left(l^7-\left(l^4/2\right)\right)\cos\theta^4\\ R_y^5-\frac{l^5}{2}\sin\theta^5-R_y^4+\left(l^7-\left(l^4/2\right)\right)\sin\theta^4\\ R_x^5+\frac{l^5}{2}\cos\theta^5-R_x^6+\frac{l^6}{2}\cos\theta^6\\ R_y^5+\frac{l^5}{2}\sin\theta^5-R_y^6+\frac{l^6}{2}\sin\theta^6\\ R_x^6+\frac{l^6}{2}\cos\theta^6-X^2\\ R_y^6+\frac{l^6}{2}\sin\theta^6-Y^2\\ \theta^2-\omega^2 t-\theta_0^2 \end{bmatrix}=\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}. \quad (4.11)$$

The next step in solving the equations is to form the constraint Jacobian matrix, denoted as $\mathbf{C}_{\mathbf{q}i}$ and defined as shown in (4.12). Note that for a kinematically driven system the constraint Jacobian matrix will be a square matrix because the number of coordinates equals the number of constraints.

$$\mathbf{C}_{\mathbf{q}i} = \begin{bmatrix} \frac{\partial C_1}{\partial q_1} & \frac{\partial C_1}{\partial q_2} & \frac{\partial C_1}{\partial q_3} & \cdots & \frac{\partial C_1}{\partial q_n} \\ \frac{\partial C_2}{\partial q_1} & \frac{\partial C_2}{\partial q_2} & \frac{\partial C_2}{\partial q_3} & \cdots & \frac{\partial C_2}{\partial q_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial C_{n_c}}{\partial q_1} & \frac{\partial C_{n_c}}{\partial q_2} & \frac{\partial C_{n_c}}{\partial q_3} & \cdots & \frac{\partial C_{n_c}}{\partial q_n} \end{bmatrix}. \tag{4.12}$$

The constraint Jacobian matrix for the system shown in Figure 4-2 is shown in equation (4.13).

$$\mathbf{C}_{\mathbf{q}} = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&\frac{l^2}{2}\sin\theta^2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&-\frac{l^2}{2}\cos\theta^2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&-\frac{l^2}{2}\sin\theta^2&-1&0&-\frac{l^3}{2}\sin\theta^3&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&\frac{l^2}{2}\cos\theta^2&0&-1&\frac{l^3}{2}\cos\theta^3&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&-\frac{l^3}{2}\sin\theta^3&-1&0&-\frac{l^4}{2}\sin\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&\frac{l^3}{2}\cos\theta^3&0&-1&\frac{l^4}{2}\cos\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&-\frac{l^4}{2}\sin\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&\frac{l^4}{2}\cos\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&-1&0&-\left(l^7-\frac{l^4}{2}\right)\sin\theta^4&1&0&\frac{l^5}{2}\sin\theta^5&0&0&0\\
0&0&0&0&0&0&0&0&0&0&-1&\left(l^7-\frac{l^4}{2}\right)\cos\theta^4&0&1&-\frac{l^5}{2}\cos\theta^5&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&-\frac{l^5}{2}\sin\theta^5&-1&0&-\frac{l^6}{2}\sin\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&\frac{l^5}{2}\cos\theta^5&0&-1&\frac{l^6}{2}\cos\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&-\frac{l^6}{2}\sin\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&\frac{l^6}{2}\cos\theta^6\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix} \tag{4.13}$$

The equations of motion are solved using a Newton-Raphson based algorithm. First, the vector of Newton differences must be found, which shows the deviation of the solution to the constraint equations from zeros and is denoted as

$$\mathbf{\Delta q} = \begin{bmatrix} \Delta q_1 & \Delta q_2 & \cdots & \Delta q_n \end{bmatrix}^T, \tag{4.14}$$

and is calculated, for a particular iteration number j, using the constraint Jacobian matrix and constraint matrix as shown

$$\mathbf{C}_{\mathbf{q}_j}\Delta\mathbf{q}_j = -\mathbf{C}\left(\mathbf{q}_j, t\right). \tag{4.15}$$

This is then used to update the vector of system coordinates as shown

$$\mathbf{q}_{j+1} = \mathbf{q}_j + \Delta\mathbf{q}_j \ . \tag{4.16}$$

Vector $\mathbf{q}_{j+1}$ is then used to reconstruct (4.15) to generate the vector $\Delta\mathbf{q}_{j+1}$. The process continues for a number of specified iterations, or until a predefined accuracy is reached.

The velocity vector of the coordinates, $\dot{\mathbf{q}}$, is determined by differentiating the vector of constraint equations (4.11) which yields

$$\mathbf{C}_{\mathbf{q}}\dot{\mathbf{q}} + \mathbf{C}_t = \mathbf{0}, \tag{4.17}$$

where $\mathbf{C}_t$ is the partial differentiation of (4.11) with respect to time and is defined as

$$\mathbf{C}_t = \begin{bmatrix} \frac{\partial C_1}{\partial t} & \frac{\partial C_2}{\partial t} & \cdots & \frac{\partial C_n}{\partial t} \end{bmatrix}^T . \tag{4.18}$$

Equation (4.17) can then be used to solve for $\dot{\mathbf{q}}$. The acceleration vector of coordinates, $\ddot{\mathbf{q}}$, can be obtained by differentiating (4.17), after some manipulation of this

$$\mathbf{C}_{\mathbf{q}}\ddot{\mathbf{q}} = \mathbf{Q}_d, \tag{4.19}$$

where vector $\mathbf{Q}_d$ is

$$\mathbf{Q}_d = -\left(\mathbf{C_q}\dot{\mathbf{q}}\right)_q \dot{\mathbf{q}} - 2\mathbf{C}_{\mathbf{q}t}\dot{\mathbf{q}} - \mathbf{C}_{tt}\,. \tag{4.20}$$

If the angular velocities in the system are assumed to be constant, the vectors $\mathbf{C}_{\mathbf{q}t}$ and $\mathbf{C}_{tt}$ are equal to zero because they are not explicit functions of time or the system coordinates. Vector $\mathbf{Q}_d$ can now be easily determined and thus (4.19) can be used to solve for $\ddot{\mathbf{q}}$. The matrix $\mathbf{C_q}\dot{\mathbf{q}}$ is defined as

$$\mathbf{C_q}\dot{\mathbf{q}} = \begin{bmatrix} \dot{R}_x^1 \\ \dot{R}_y^1 \\ \dot{\theta}^1 \\ \dot{R}_x^2 + \frac{\dot{\theta}^2 l^2}{2}\sin\theta^2 \\ \dot{R}_y^2 - \frac{\dot{\theta}^2 l^2}{2}\cos\theta^2 \\ \dot{R}_x^2 - \frac{\dot{\theta}^2 l^2}{2}\sin\theta^2 - \dot{R}_x^3 - \frac{\dot{\theta}^3 l^3}{2}\sin\theta^3 \\ \dot{R}_y^2 + \frac{\dot{\theta}^2 l^2}{2}\cos\theta^2 - \dot{R}_y^3 + \frac{\dot{\theta}^3 l^3}{2}\cos\theta^3 \\ \dot{R}_x^3 + \frac{\dot{\theta}^3 l^3}{2}\sin\theta^3 - \dot{R}_x^4 + \frac{\dot{\theta}^4 l^4}{2}\sin\theta^4 \\ \dot{R}_y^3 + \frac{\dot{\theta}^3 l^3}{2}\cos\theta^3 - \dot{R}_y^4 + \frac{\dot{\theta}^4 l^4}{2}\cos\theta^4 \\ \dot{R}_x^4 + \frac{\dot{\theta}^4 l^4}{2}\sin\theta^4 \\ \dot{R}_y^4 + \frac{\dot{\theta}^4 l^4}{2}\cos\theta^4 \\ \dot{R}_x^5 + \frac{\dot{\theta}^5 l^5}{2}\sin\theta^5 - \dot{R}_x^4 - \dot{\theta}^4\left(l^7 - \left(l^4/2\right)\right)\sin\theta^4 \\ \dot{R}_y^5 - \frac{\dot{\theta}^5 l^5}{2}\cos\theta^5 - \dot{R}_y^4 + \dot{\theta}^4\left(l^7 - \left(l^4/2\right)\right)\cos\theta^4 \\ \dot{R}_x^5 - \frac{\dot{\theta}^5 l^5}{2}\sin\theta^5 - \dot{R}_x^6 - \frac{\dot{\theta}^6 l^6}{2}\sin\theta^6 \\ \dot{R}_y^5 + \frac{\dot{\theta}^5 l^5}{2}\cos\theta^5 - \dot{R}_y^6 + \frac{\dot{\theta}^6 l^6}{2}\cos\theta^6 \\ \dot{R}_x^6 - \frac{\dot{\theta}^6 l^6}{2}\sin\theta^6 \\ \dot{R}_y^6 + \frac{\dot{\theta}^6 l^6}{2}\cos\theta^6 \\ \dot{\theta}^2 \end{bmatrix}. \tag{4.21}$$

Matrix $\left(\mathbf{C_q}\dot{\mathbf{q}}\right)_q$ is

$$\left(\mathbf{C_q}\dot{\mathbf{q}}\right)_q = \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&\frac{\dot{\theta}^2 l^2}{2}\cos\theta^2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&\frac{\dot{\theta}^2 l^2}{2}\sin\theta^2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&-\frac{\dot{\theta}^2 l^2}{2}\cos\theta^2&0&0&-\frac{\dot{\theta}^3 l^3}{2}\cos\theta^3&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&-\frac{\dot{\theta}^2 l^2}{2}\sin\theta^2&0&0&-\frac{\dot{\theta}^3 l^3}{2}\sin\theta^3&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^3 l^3}{2}\cos\theta^3&0&0&-\frac{\dot{\theta}^4 l^4}{2}\cos\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^3 l^3}{2}\sin\theta^3&0&0&-\frac{\dot{\theta}^4 l^4}{2}\sin\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^4 l^4}{2}\cos\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^4 l^4}{2}\sin\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-\dot{\theta}^4\left(l^7-\frac{l^4}{2}\right)\cos\theta^4&0&0&\frac{\dot{\theta}^5 l^5}{2}\cos\theta^5&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-\dot{\theta}^4\left(l^7-\frac{l^4}{2}\right)\sin\theta^4&0&0&\frac{\dot{\theta}^5 l^5}{2}\sin\theta^5&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\cos\theta^5&0&0&-\frac{\dot{\theta}^6 l^6}{2}\cos\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\sin\theta^5&0&0&-\frac{\dot{\theta}^6 l^6}{2}\sin\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^6 l^6}{2}\cos\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^6 l^6}{2}\sin\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}, \quad (4.22)$$

and therefore the matrix $\mathbf{Q}_d$ can be found and is shown in (4.23).

$$\mathbf{Q}_d = -\left(\mathbf{C_q}\dot{\mathbf{q}}\right)_q \dot{\mathbf{q}} = \left[\mathbf{Q}_d^1 \quad \mathbf{Q}_d^2\right]^T$$

$$\mathbf{Q}_d^1 = \begin{bmatrix}
0\\
0\\
0\\
\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\cos\theta^2\\
\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\sin\theta^2\\
-\left(\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\cos\theta^2+\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\cos\theta^3\right)\\
-\left(\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\sin\theta^2+\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\sin\theta^3\right)\\
-\left(\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\cos\theta^3+\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\cos\theta^4\right)\\
-\left(\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\sin\theta^3+\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\sin\theta^4\right)
\end{bmatrix}, \quad \text{and} \quad \mathbf{Q}_d^2 = \begin{bmatrix}
-\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\cos\theta^4\\
-\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\sin\theta^4\\
-\left(\dot{\theta}^4\right)^2\left(l^7-\frac{l^4}{2}\right)\cos\theta^4+\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\cos\theta^5\\
-\left(\dot{\theta}^4\right)^2\left(l^7-\frac{l^4}{2}\right)\sin\theta^4+\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\sin\theta^5\\
-\left(\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\cos\theta^5+\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\cos\theta^6\right)\\
-\left(\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\sin\theta^5+\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\sin\theta^6\right)\\
-\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\cos\theta^6\\
-\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\sin\theta^6\\
0
\end{bmatrix} \quad (4.23)$$

Grashof's Law

The linkage systems modelled in this project can be seen as two mutually coupled 4-bar linkage systems. For example, the slave driven system in Figure 4-2 can be split into two 4-bar systems described by design points O-A-B-C (crank-rocker linkage) and C-D-E-F (double-rocker linkage) respectively. Because these are examples of closed systems, there are restrictions on the relative lengths of the linkages in order to allow the system to move as desired. For a planar 4-bar linkage system, Grashof's law can be applied, which states that: the sum of the shortest and longest links must be less than or equal to the sum of the other two links. If this inequality is satisfied, one of the links can make a full revolution. An algorithm has been developed to apply Grashof's law directly to linkage O-A-B-C using the design points; the code to implement the algorithm is found in Appendix D.

Step 1: Measure the lengths of the linkages using (4.24) and assign the parameters names shown in (4.25).

$$AB = \sqrt{\left(B_x - A_x\right)^2 + \left(B_y - A_y\right)^2 + \left(B_z - A_z\right)^2} \tag{4.24}$$

$$\begin{gathered} L_1 = OC,\quad L_2 = OA,\quad \mathrm{L}_3 = \mathrm{AB},\quad \mathrm{L}_4 = \mathrm{BC} \\ \mathrm{L}_5 = \ CD,\quad \mathrm{L}_6 = DE,\quad \mathrm{L}_7 = EF,\quad \mathrm{L}_8 = FC \end{gathered} \tag{4.25}$$

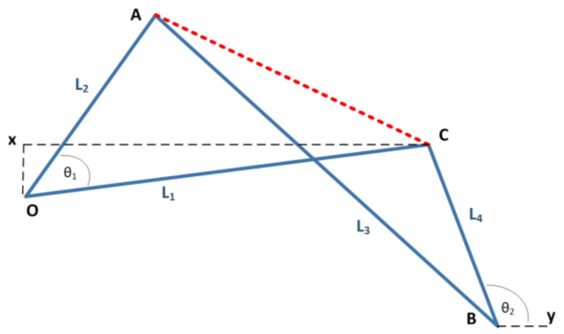
Step 2: Sort lengths $L1$ to $L4$ from smallest to largest.

Step 3: Apply the inequality

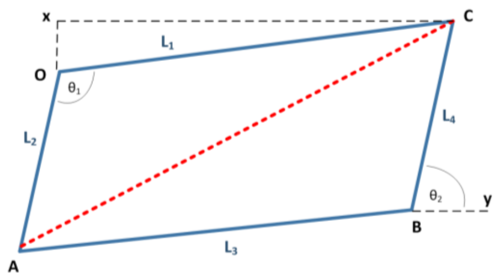
$$s + l \le p + q \tag{4.26}$$

where s and l are the shortest and longest lengths respectively and p and q are the two remaining lengths. If (4.26) is satisfied the configuration is feasible.

Since linkage C-D-E-F is a double-rocker linkage, inequality (4.26) does not need to be satisfied because no linkage needs to make a complete revolution. However, intuitively it can be deduced that there are still constraints on the lengths of the linkages in order for body 4 (and thus the crank, body 2) to have its range of motion. The first step is to determine the angle through which BC, and thus CD, oscillates. Figure 4-3 shows the O-A-B-C linkage in two configurations. Taking θ_1 as the angle AOC and defining it as zero when the vector in the direction $\mathbf{OA}$ is the same as $\mathbf{OC}$ it can be seen that Figure 4-3 (a) and (b) show the linkages for $0 \le \theta_1 \le \pi$ and $\pi \le \theta_1 < 2\pi$ respectively.



(a)



(b)

Figure 4-3: Double Rocker Linkage Inequality Derivation (1)

The length AC is calculated using

$$AC = \sqrt{L_1^2 + L_2^2 - 2L_1L_2\cos\theta_1} \ . \tag{4.27}$$

Using AC, the angles $\angle ACO$ and $\angle ACB$ can be found as shown, respectively

$$\angle ACO = \cos^{-1}\left(\frac{L_1^2 + (AC)^2 - L_2^2}{2L_1(AC)}\right)$$
$$. \tag{4.28}$$
$$\angle ACB = \cos^{-1}\left(\frac{L_4^2 + (AC)^2 - L_3^2}{2L_4(AC)}\right)$$

The position x can be found using the definition of O and C, and thus the angle $\angle xCO$ can easily be found. It can be seen by inspection of Figure 4-3 that the angle $\angle xCB$ is equal to θ_2. In the case of Figure 4-3a, θ_2 can be expressed as

$$\theta_2 = \angle xCO + \angle ACB - \angle ACO \ , \tag{4.29}$$

and in the case of Figure 4-3b, θ_2 can be expressed as

$$\theta_2 = \angle xCO + \angle ACB + \angle ACO \ . \tag{4.30}$$

Sweeping θ_1 from 0 to 2π radians gives the range of motion for θ_2.

Figure 4-4 shows linkage C-D-E-F. Aim is to find the maximum length of FD in terms of θ_2 in order to determine the allowable lengths of L_6 and L_7.

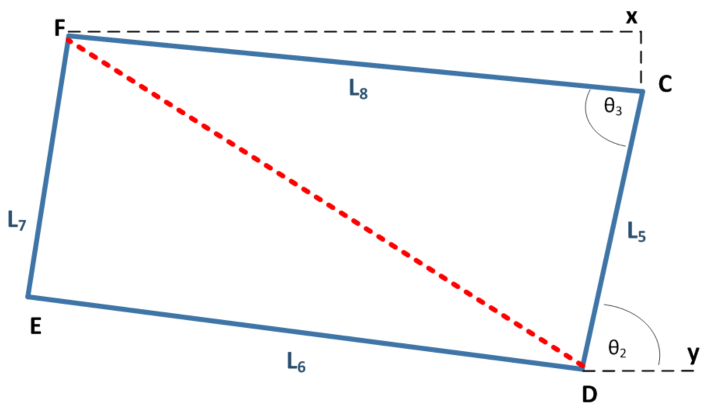


Figure 4-4: Double Rocker Linkage Inequality Derivation (2)

In a similar way to the previous linkage O-A-B-C, angle θ_3 can be expressed as

$$\theta_3 = \theta_2 + \angle xFC \tag{4.31}$$

and thus FD can be calculated using

$$FD = \sqrt{L_5^2 + L_8^2 - 2L_5 L_8 \cos\theta_3} \ . \tag{4.32}$$

The maximum value of FD can be found using the range of values of θ_2 and thus the inequality shown in (4.33) can be applied.

$$L_6 + L_7 \geq FD_{\max} \ . \tag{4.33}$$

4.2.2 Wiper Arms

Two styles of wiper arm are modelled in this project: the straight wiper arm shown in Figure 4-5 and the bent wiper arm shown in Figure 4-6. In this case, the design points represent rigid joints, meaning that the arms are modelled as multiple rigidly connected bodies. The input to the wiper arm is the angle of the linkage rocker that it is attached to. This means that the driving constraint is not explicitly a function of time making the vector $\mathbf{C}_t$ (see equation (4.18)), and thus the velocity and acceleration vectors $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, equal to zero. For this reason, for each time-step, the velocity (acceleration) of the bodies is calculated by subtracting the current position (velocity) by the previous position (velocity) and dividing by the time-step length.

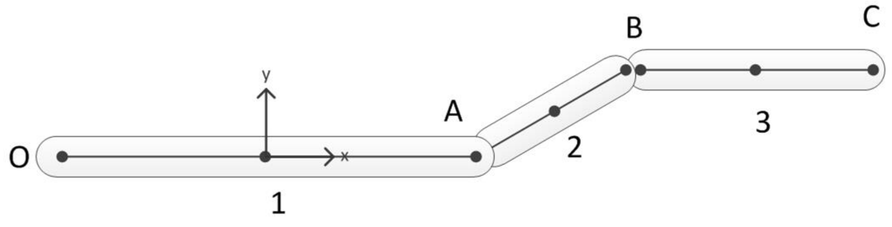


Figure 4-5: Straight Wiper Arm

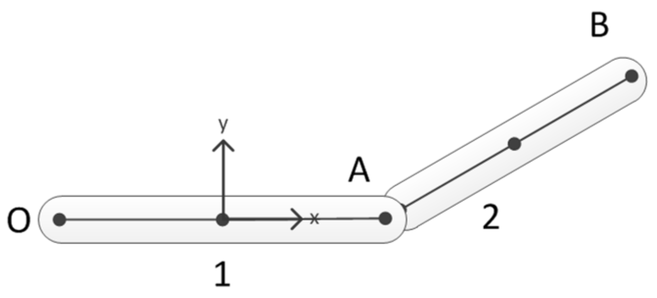


Figure 4-6: Bent Wiper Arm

4.2.1 Straight Wiper Arm

The coordinate vector for the straight wiper arm shown in Figure 4-5 is given as

$$\mathbf{q} = \begin{bmatrix} R_x^1 & R_y^1 & \theta^1 & R_x^2 & R_y^2 & \theta^2 & R_x^3 & R_y^3 & \theta^3 \end{bmatrix}^T , \tag{4.34}$$

and the vector of constraints is given as

$$\mathbf{C}(\mathbf{q},t) = \begin{bmatrix} C_1(\mathbf{q},t) \\ C_2(\mathbf{q},t) \\ C_3(\mathbf{q},t) \\ C_4(\mathbf{q},t) \\ C_5(\mathbf{q},t) \\ C_6(\mathbf{q},t) \\ C_7(\mathbf{q},t) \\ C_8(\mathbf{q},t) \\ C_9(\mathbf{q},t) \end{bmatrix} = \begin{bmatrix} R_x^1 - \frac{l^1}{2}\cos\theta^1 \\ R_y^1 - \frac{l^1}{2}\sin\theta^1 \\ \theta^1 - \theta^{link} - \theta_o^1 \\ R_x^1 + \frac{l^1}{2}\cos\theta^1 - R_x^2 + \frac{l^2}{2}\cos\theta^2 \\ R_y^1 + \frac{l^1}{2}\sin\theta^1 - R_y^2 + \frac{l^2}{2}\sin\theta^2 \\ \theta^2 - \theta^1 - k_\theta^1 \\ R_x^2 + \frac{l^2}{2}\cos\theta^2 - R_x^3 + \frac{l^3}{2}\cos\theta^3 \\ R_y^2 + \frac{l^2}{2}\sin\theta^2 - R_y^3 + \frac{l^3}{2}\sin\theta^3 \\ \theta^3 - \theta^2 - k_\theta^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} , \tag{4.35}$$

where θ^{link} is the angle of the connecting linkage rocker, θ_o^1 is the initial angle between body 1 and the +X direction minus the initial angle of the connecting linkage rocker.

Constant k_θ^1 is the angle between body 1 and body 2, and k_θ^2 is the angle between bodies 2 and 3. The Jacobian matrix is

$$\mathbf{C}_q = \begin{bmatrix} 1 & 0 & \frac{l^1}{2}\sin\theta^1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -\frac{l^1}{2}\cos\theta^1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -\frac{l^1}{2}\sin\theta^1 & -1 & 0 & -\frac{l^2}{2}\sin\theta^2 & 0 & 0 & 0 \\ 0 & 1 & \frac{l^1}{2}\cos\theta^1 & 0 & -1 & \frac{l^2}{2}\cos\theta^2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\frac{l^2}{2}\sin\theta^2 & -1 & 0 & -\frac{l^3}{2}\sin\theta^3 \\ 0 & 0 & 0 & 0 & 1 & \frac{l^2}{2}\cos\theta^2 & 0 & -1 & \frac{l^3}{2}\cos\theta^3 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}. \tag{4.36}$$

4.2.2 Bent Wiper Arm

The coordinate vector for the straight wiper arm shown in Figure 4-6 is given as

$$\mathbf{q} = \begin{bmatrix} R_x^1 & R_y^1 & \theta^1 & R_x^2 & R_y^2 & \theta^2 \end{bmatrix}^T . \tag{4.37}$$

and the vector of constraints is given as

$$\mathbf{C}(\mathbf{q},t) = \begin{bmatrix} C_1(\mathbf{q},t) \\ C_2(\mathbf{q},t) \\ C_3(\mathbf{q},t) \\ C_4(\mathbf{q},t) \\ C_5(\mathbf{q},t) \\ C_6(\mathbf{q},t) \end{bmatrix} = \begin{bmatrix} R_x^1 - \frac{l^1}{2}\cos\theta^1 \\ R_y^1 - \frac{l^1}{2}\sin\theta^1 \\ \theta^1 - \theta^{link} - \theta_o^1 \\ R_x^1 + \frac{l^1}{2}\cos\theta^1 - R_x^2 + \frac{l^2}{2}\cos\theta^2 \\ R_y^1 + \frac{l^1}{2}\sin\theta^1 - R_y^2 + \frac{l^2}{2}\sin\theta^2 \\ \theta^2 - \theta^1 - k_\theta^1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.38}$$

where θ^{link} , θ_o^1 and k_θ^1 are defined in the same way as for the straight wiper arm. The Jacobian matrix is

$$\mathbf{C}_q = \begin{bmatrix} 1 & 0 & \frac{l^1}{2}\sin\theta^1 & 0 & 0 & 0 \\ 0 & 1 & -\frac{l^1}{2}\cos\theta^1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & -\frac{l^1}{2}\sin\theta^1 & -1 & 0 & -\frac{l^2}{2}\sin\theta^2 \\ 0 & 1 & \frac{l^1}{2}\cos\theta^1 & 0 & -1 & \frac{l^2}{2}\cos\theta^2 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}. \tag{4.39}$$

4.2.3 Wiper Blade

The wiper blade is defined as shown in Figure 4-7 with the origin (O) connecting to design point C of the straight wiper blade or B of the curved. The input to the wiper blade is the angle of the wiper arm that it is attached to, and thus is not an explicit function of time.

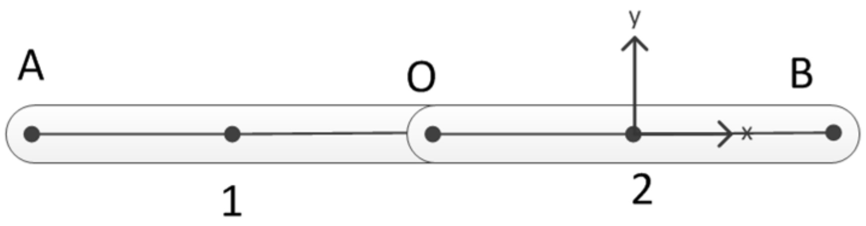


Figure 4-7: Wiper Blade

The coordinate vector is given as

$$\mathbf{q} = \left[R_x^1 \quad R_y^1 \quad \theta^1 \quad R_x^2 \quad R_y^2 \quad \theta^2 \right]^T , \tag{4.40}$$

and the vector of constraints is

$$\mathbf{C}(\mathbf{q},t)=\begin{bmatrix} C_1(\mathbf{q},t) \\ C_2(\mathbf{q},t) \\ C_3(\mathbf{q},t) \\ C_4(\mathbf{q},t) \\ C_5(\mathbf{q},t) \\ C_6(\mathbf{q},t) \end{bmatrix} = \begin{bmatrix} R_x^1-\frac{l^1}{2}\cos\theta^1 \\ R_y^1-\frac{l^1}{2}\sin\theta^1 \\ \theta^1-\theta^{arm}-\theta_o^1 \\ R_x^1+\frac{l^1}{2}\cos\theta^1-R_x^2+\frac{l^2}{2}\cos\theta^2 \\ R_y^1+\frac{l^1}{2}\sin\theta^1-R_y^2+\frac{l^2}{2}\sin\theta^2 \\ \theta^2-\theta^{arm}-\theta_o^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{4.41}$$

where θ^{arm} is the angle of the connecting arm, θ_o^1 and θ_o^2 are the initial angles between body 1 and body the +X direction respectively, minus the initial angle of the connecting arm. The Jacobian matrix is

$$\mathbf{C}_q=\begin{bmatrix} 1 & 0 & \frac{l^1}{2}\sin\theta^1 & 0 & 0 & 0 \\ 0 & 1 & -\frac{l^1}{2}\cos\theta^1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \frac{l^2}{2}\sin\theta^2 \\ 0 & 0 & 0 & 0 & 1 & -\frac{l^2}{2}\cos\theta^2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.42}$$

4.2.4 Planar Kinematic Simulation

In this section, an example of the entire mechanical wiper system is simulated. A flow diagram of the simulation algorithm is shown in Figure 4-8.

Figure 4-9 shows plots of the wiper system in four different wipe positions (or crank angles) from the simulation results, with the scales in cm.

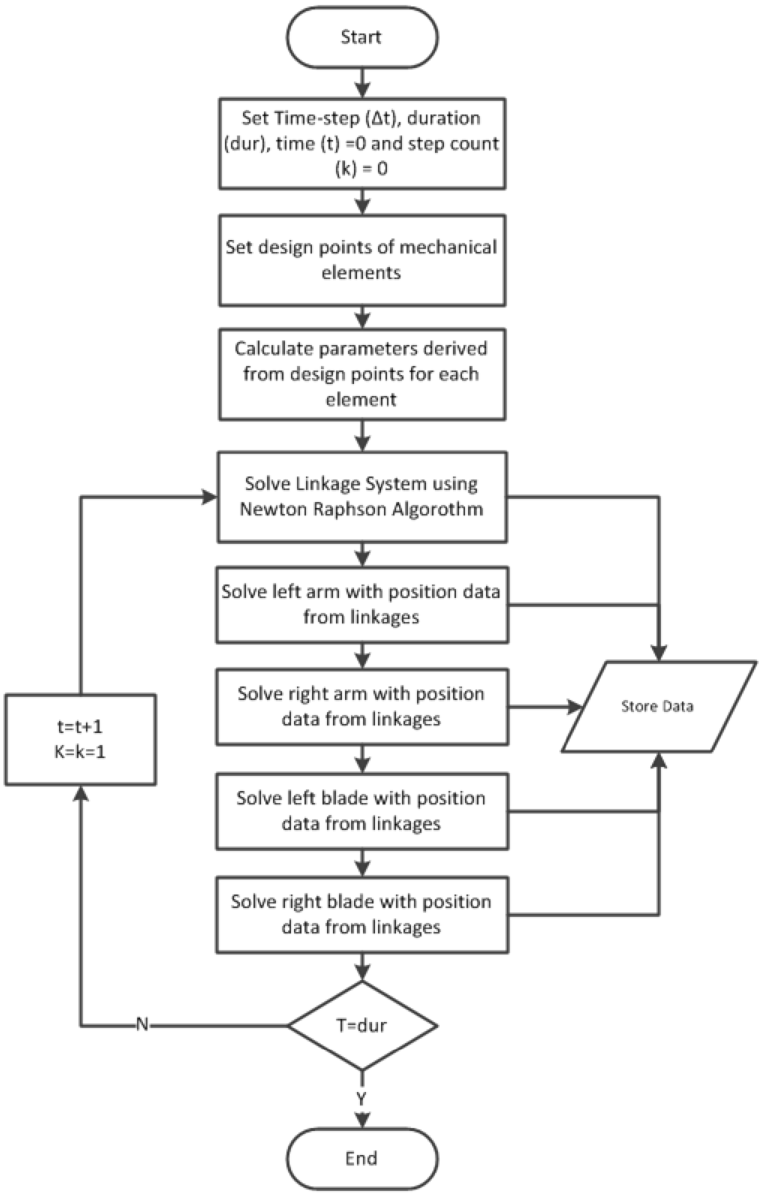
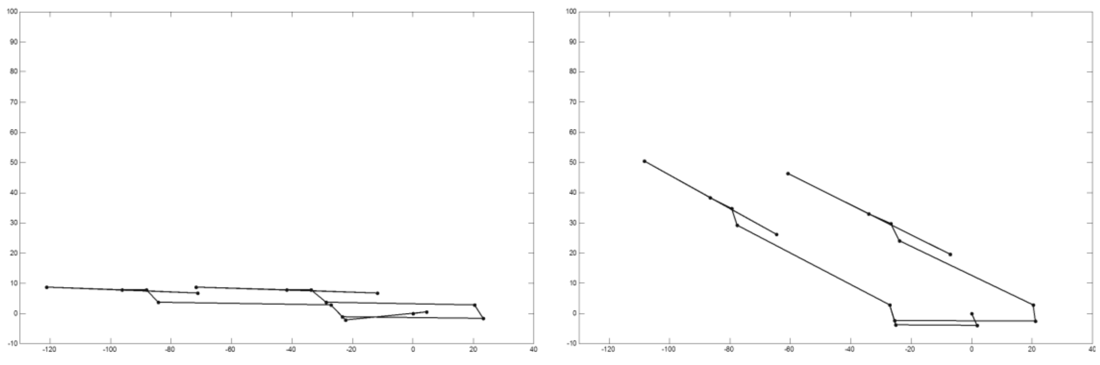

Start
Set Time-step (Δt), duration (dur), time (t) =0 and step count (k) = 0
Set design points of mechanical elements
Calculate parameters derived from design points for each element
Solve Linkage System using Newton Raphson Algorothm
Solve left arm with position data from linkages
Solve right arm with position data from linkages
Solve left blade with position data from linkages
Solve right blade with position data from linkages
Store Data
t=t+1
K=k=1
T=dur
N
Y
End

Figure 4-8: Planar Kinematics Simulation Flow Diagram



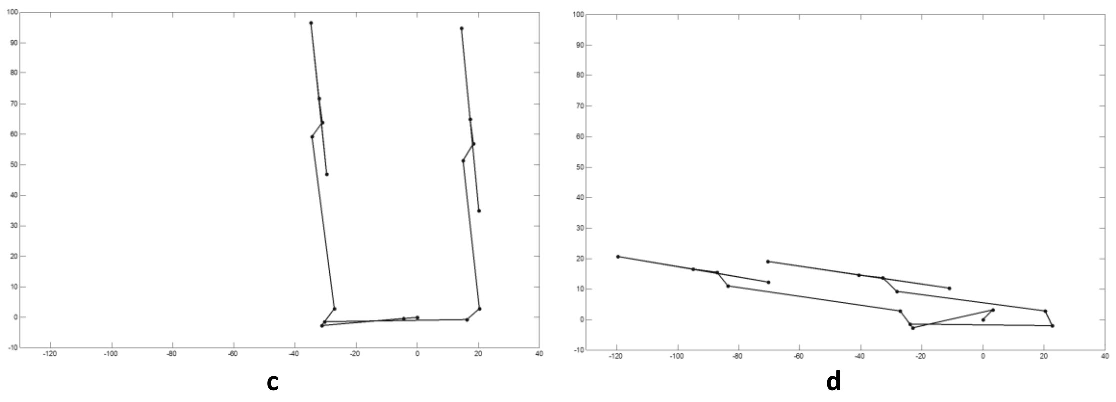


Figure 4-9: Planar Kinematics Simulation Plots

4.3 Physical Model Based on SimMechanics

In this section, physical models of the mechanical elements of the wiper system are developed using the physical modelling tool SimMechanics, from the Mathworks.

4.3.1 Slave Driven Linkage System

Figure 4-10 shows the top level block representing the right hand drive (RHD) slave linkage system (all linkage systems modelled have the same I/O). As is the standard with physical modelling, the I/O of the model is the same as that of the real system and represents two way physical ports. The Input_Follower and Input_Body connect to the shaft and case of the motor respectively and allow a two way torque and rotational motion interface. The Left_Rocker and Right_Rocker connections interface with the wiper arms. The Left_Dynamics and Right_Dynamics are Simulink signals with position, angular velocity and angular acceleration data of the rockers to be measured.

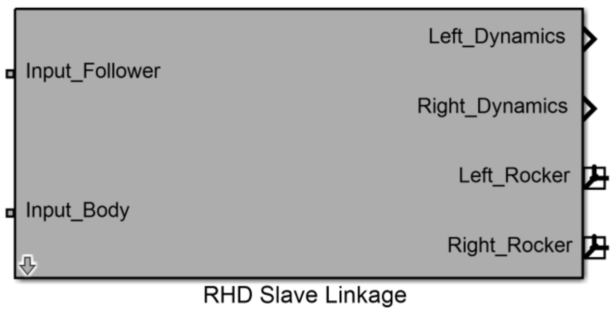


Figure 4-10: Linkage System Physical Model I/O

Figure 4-11 shows the model underneath the mask shown in Figure 4-10 of the system in Figure 4-2. Each joint is represented by a revolute joint block, labelled with its corresponding design point. Each linkage is represented by a body block and labelled with its corresponding body number. The connections on the model imitate the physical connection on the real system. The Revolute-Rotational Interface block connects the revolute joint at design point O with the motor.

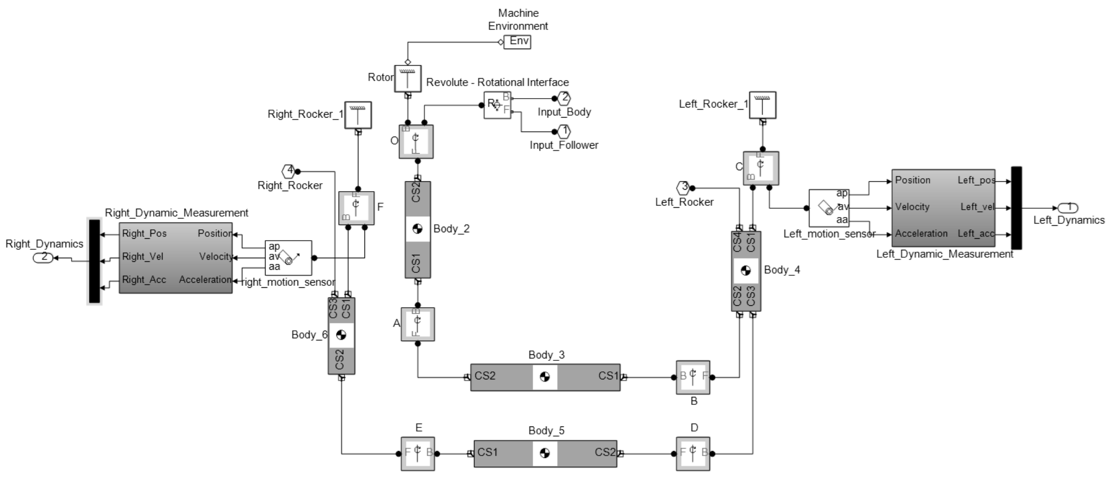
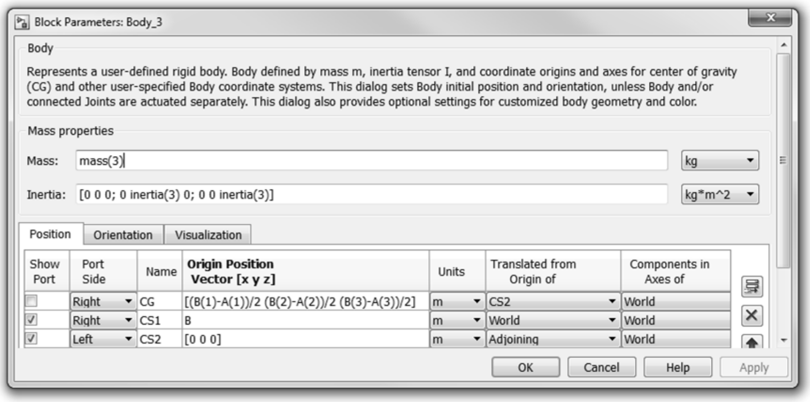


Figure 4-11: Linkage System Physical Model

Figure 4-12 gives a typical example of the generic modelling strategy in terms of setting the parameters for a body, in this case body 3. The mass of the body is defined directly as a parameter by the user; the inertia tensor is then derived from this. Each linkage is approximated as a slender rod and as such has an inertia tensor defined as

$$I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{12}ml^2 & 0 \\ 0 & 0 & \frac{1}{12}ml^2 \end{bmatrix}, \tag{4.43}$$

where m is the mass of the body and l is the length.


Block Parameters: Body_3

Body

Represents a user-defined rigid body. Body defined by mass m, inertia tensor I, and coordinate origins and axes for center of gravity (CG) and other user-specified Body coordinate systems. This dialog sets Body initial position and orientation, unless Body and/or connected Joints are actuated separately. This dialog also provides optional settings for customized body geometry and color.

Mass properties

Mass:	mass(3)	kg
Inertia:	[0 0 0; 0 inertia(3) 0; 0 0 inertia(3)]	kg*m^2

Position | Orientation | Visualization

Show Port	Port Side	Name	Origin Position Vector [x y z]	Units	Translated from Origin of	Components in Axes of
☐	Right	CG	[(B(1)-A(1))/2 (B(2)-A(2))/2 (B(3)-A(3))/2]	m	CS2	World
✓	Right	CS1	B	m	World	World
✓	Left	CS2	[0 0 0]	m	Adjoining	World

Figure 4-12: Linkage System Physical Model Body Level Parameterisation

Typically, a body will have three CS's associated with it: CS1, CS2 and CG defining the position of the two joints and the centre of gravity of the body respectively. Referring to Body 3 in Figure 4-11, it can be seen that CS2_Body_3 is connected to CS1_Body_2, which is defined as being design point A. Thus, the origin of CS2_Body_3 is defined as being [0 0 0] with respect to its adjoining CS, meaning that CS1_Body_2 and CS2_Body_3 are in the same position in space. CS1_Body_3 defines the position of design point/joint B in terms of the World CS (WCS) which is stationary at the design point O. The centre of gravity is calculated automatically in CG_Body_3 based on the position of the joints at either end of the body. All of the bodies are parameterised in a similar way and thus the model is systematically built up.

In order for the inertia tensor shown in (4.43) to be valid, the CG CS must be orientated in such a way that its x-axis is pointing in the direction of the linkage, i.e. for the case of Body_3, the direction of the vector $\mathbf{AB}$. This rotation is specified in terms of quaternions using the following method (for the case of Body_3 by way of an example).

First, the dot product between the vector $\mathbf{AB}$ and the positive X axis of the WCS is taken to calculate the angle through which the CG CS must rotate:

$$\theta = \cos^{-1}\left(\left\{[\mathbf{B}-\mathbf{A}]\bullet[1 \quad 0 \quad 0]\right\}/L^3\right). \tag{4.44}$$

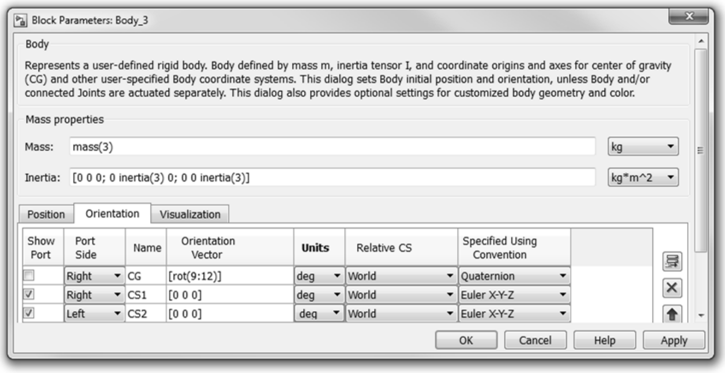
Second, take the cross product between vector $\mathbf{AB}$ and the positive X axis of the WCS to calculate the unit vector, $\hat{\mathbf{n}}$, about which the rotation takes place:

$$\hat{\mathbf{n}} = \left([\mathbf{A}-\mathbf{B}]\times[1 \quad 0 \quad 0]\right)/\left(L^3 \sin\theta\right). \tag{4.45}$$

Finally, the elements of the quaternion vector are defined as:

$$\begin{aligned} w &= \cos\left(\theta/2\right) \\ \mathbf{x} &= \sin\left(\theta/2\right)\hat{\mathbf{n}}(1) \\ \mathbf{y} &= \sin\left(\theta/2\right)\hat{\mathbf{n}}(2) \\ \mathbf{z} &= \sin\left(\theta/2\right)\hat{\mathbf{n}}(3) \\ \text{quaternion\_vector} &= [w \quad \mathbf{x} \quad \mathbf{y} \quad \mathbf{z}] \end{aligned} \tag{4.46}$$

The orientation of the body CG CS is then parameterised as shown in Figure 4-14, with the “rot(9:12)” vector being the four quaternion elements defined in (4.46) for Body_3.


Block Parameters: Body_3

Body

Represents a user-defined rigid body. Body defined by mass m, inertia tensor I, and coordinate origins and axes for center of gravity (CG) and other user-specified Body coordinate systems. This dialog sets Body initial position and orientation, unless Body and/or connected Joints are actuated separately. This dialog also provides optional settings for customized body geometry and color.

Mass properties

Mass: mass(3) kg

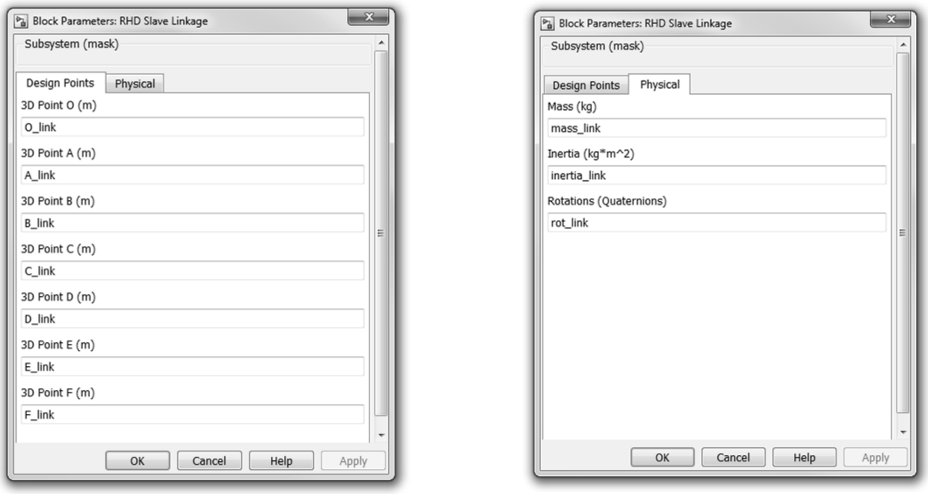
Inertia: [0 0 0; 0 inertia(3) 0; 0 0 inertia(3)] kg*m^2

Position | Orientation | Visualization

Show Port	Port Side	Name	Orientation Vector	Units	Relative CS	Specified Using Convention
☐	Right	CG	[rot(9:12)]	deg	World	Quaternion
☑	Right	CS1	[0 0 0]	deg	World	Euler X-Y-Z
☑	Left	CS2	[0 0 0]	deg	World	Euler X-Y-Z

Figure 4-13: Linkage System Physical Model Body Orientation Parameterisation

The inner workings of the model are hidden from the user and the model is parameterised on the top level mask as shown in Figure 4-14.


Block Parameters: RHD Slave Linkage

Subsystem (mask)

Design Points | Physical

3D Point O (m): O_link

3D Point A (m): A_link

3D Point B (m): B_link

3D Point C (m): C_link

3D Point D (m): D_link

3D Point E (m): E_link

3D Point F (m): F_link

OK | Cancel | Help | Apply

Block Parameters: RHD Slave Linkage

Subsystem (mask)

Design Points | Physical

Mass (kg): mass_link

Inertia (kg*m^2): inertia_link

Rotations (Quaternions): rot_link

OK | Cancel | Help | Apply

Figure 4-14: Linkage System Physical Model Top Level Parameterisation

4.3.2 Wiper Arms

The straight and curved wiper arms are modelled in a similar way to the linkage system. The straight wiper arm is shown here by way of an example. The curved arm is modelled in the same way but with less bodies and joints.

Figure 4-15 shows the top level I/O of the wiper arm. The Right_Arm_Con port is a weld style joint that connects to the right rocker of the linkage system interfacing with the arm. The Right_Blade port is a body connection that connects to the interfacing wiper arm.

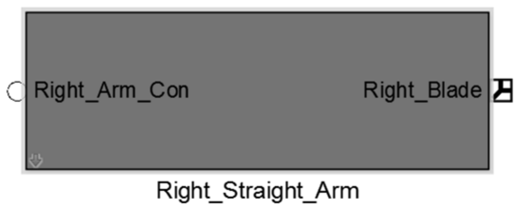


Figure 4-15: Straight Wiper Arm Physical Model I/O

Figure 4-16 shows the actual SimMechanics model of the arm shown in Figure 4-5. The joints are all weld joints since there is no relative motion between the bodies. Joint O1 and Body_1 allow the arm to be defined in a different X-Y plane to the linkages, as is the case with wiper systems.

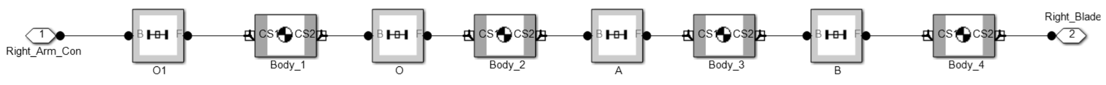
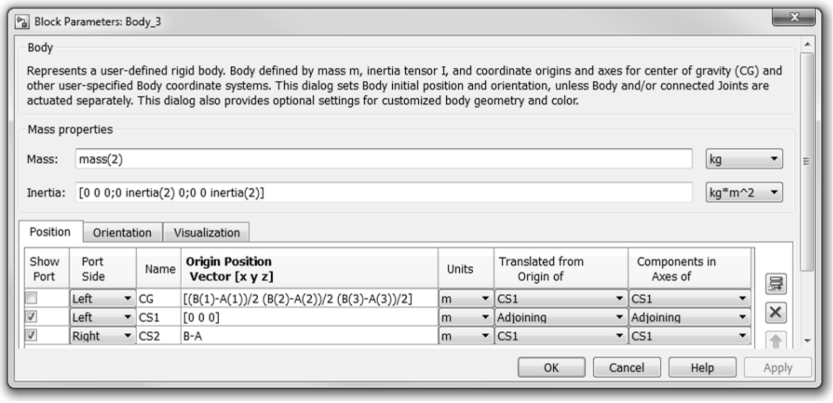


Figure 4-16: Straight Wiper Arm Physical Model

Design points A, B and C are defined in terms of the origin of the blade, i.e. point O which is [0 0 0]. This means that, unlike the linkage system, the position of the joints cannot be defined in terms of the WCS and must be defined completely in terms of adjoining CSs. The ramifications of this are demonstrated in Figure 4-17, showing the parameters of Body_3. CS2_Body_3 is defined by the vector $\mathbf{B}-\mathbf{A}$ translated from CS1_Body_3, as opposed to simply design point B translated with respect to the WCS as was the case in the linkage system. This pattern continues through the bodies through to Body_O1 which is connected directly to the rocker of the linkage system, which is defined in terms of the WCS.


Block Parameters: Body_3

Body

Represents a user-defined rigid body. Body defined by mass m, inertia tensor I, and coordinate origins and axes for center of gravity (CG) and other user-specified Body coordinate systems. This dialog sets Body initial position and orientation, unless Body and/or connected Joints are actuated separately. This dialog also provides optional settings for customized body geometry and color.

Mass properties

Mass: mass(2) kg

Inertia: [0 0 0;0 inertia(2) 0;0 0 inertia(2)] kg*m^2

Position | Orientation | Visualization

Show Port	Port Side	Name	Origin Position Vector [x y z]	Units	Translated from Origin of	Components in Axes of
☐	Left	CG	[(B(1)-A(1))/2 (B(2)-A(2))/2 (B(3)-A(3))/2]	m	CS1	CS1
✓	Left	CS1	[0 0 0]	m	Adjoining	Adjoining
✓	Right	CS2	B-A	m	CS1	CS1

Figure 4-17: Straight Wiper Physical Model Body Level Parameterisation

4.3.3 Wiper Blade

The wiper blades are modelled in a similar way to the wiper arms in that all of the joints are weld joints because there is no relative motion between the bodies. Figure 4-18 shows the I/O of the wiper blade model. Right_Blade_Con connects the blade to the interfacing arm, Vel_rblade outputs the velocity of the wiper blade in the blade CSs positive y direction and Right_Blade_Force is the input accepting the forces acting on the blade due to the arm, friction and aerodynamic forces.

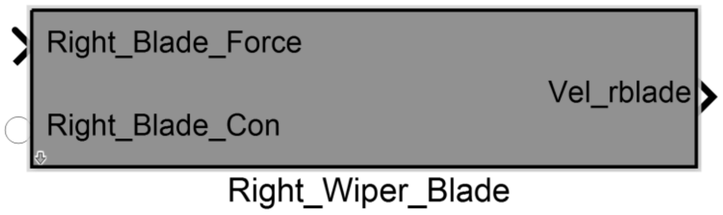


Figure 4-18: Wiper Blade Physical Model I/O

Figure 4-19 shows the actual SimMechanics model of the wiper blade. Body_2_CS3 (from which the velocity is measured) is connected to the origin of the blade and orientated such that its Y axis always points in the direction that the blade is moving. Body_1_CS4 (to which the arm, friction and aerodynamic forces are applied) is positioned and orientated in the same way as Body_2_CS3.

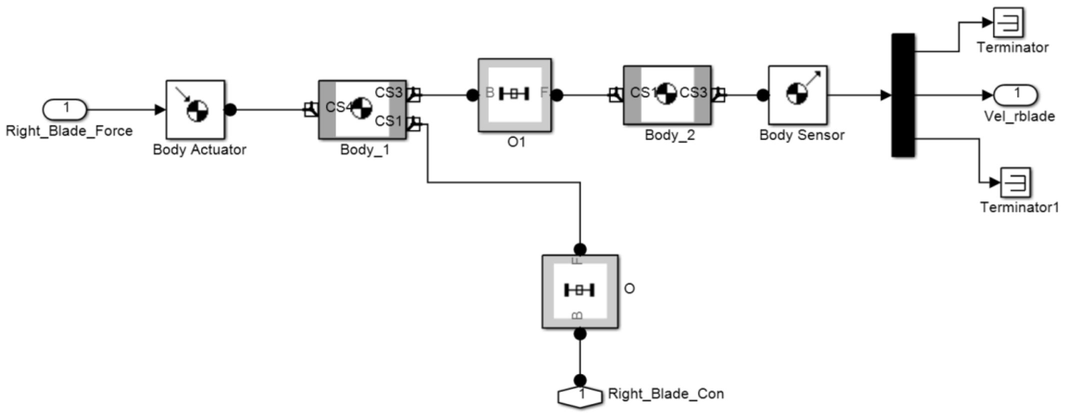


Figure 4-19: Wiper Blade Physical Model

4.3.4 Gravity Vector

Currently, the entire wiper system has been defined in one, or a series, of x-y planes. This is acceptable for kinematic analysis but in reality the system is rotated about the global x-axis by approximately the angle of the windscreen. This angle will be referred to as γ

and in general is a negative rotation about the positive x-axis. Such a rotation is defined by the rotation matrix

$$\mathbf{A}^{\gamma} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}. \tag{4.47}$$

To allow continuity between the physical and kinematic model design points, the external forces applied to the model will be rotated using (4.47), rather than changing the 3-dimensional position of the design points. For example, suppose the acute angle of the windscreen is 30°, meaning that gravity can no longer be considered as acting in the negative y-axis direction. The gravity vector applied to the bodies will be calculated by

$$\begin{aligned} Gravity_{\gamma=30^\circ} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 30 & -\sin 30 \\ 0 & \sin 30 & \cos 30 \end{bmatrix} \begin{bmatrix} 0 \\ -9.81 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ -9.81\cos 30 \\ -9.81\sin 30 \end{bmatrix} = \begin{bmatrix} 0 \\ -8.496 \\ -4.905 \end{bmatrix} \end{aligned}. \tag{4.48}$$

Note that it is possible to completely define the 3-dimensional orientation of the wiper system using the design points, taking the angle of the windscreen into account. In this case γ is set to zero.

4.4 Wiper Blade Friction Force

The nature of friction in a wiper system has been discussed in Chapter 2. In this section, data from the papers previously discussed is used to formulate models of friction that can be applied to the wiper blade model. Figure 4-20 is adapted from [74] and shows how the coefficient of friction, and thus friction force, is determined by the wetness of the windscreen wiper. Three conditions, wet, tacky and dry, are considered, where the tacky state describes the observed condition of a significantly higher friction coefficient when only a small amount of water remains on the screen. Because it is rare and difficult to define exactly when the tacky condition is active, it will not be modelled here.

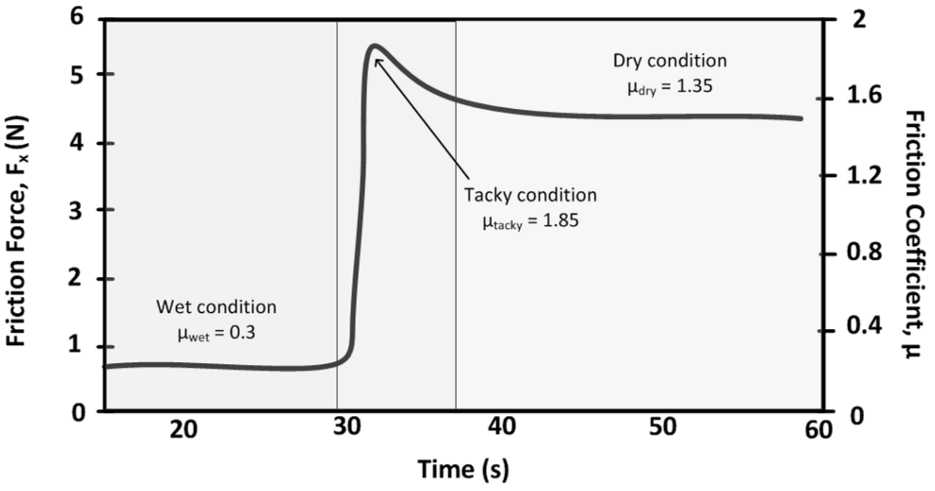


Figure 4-20: Friction Coefficient in Different Windscreen Conditions

4.4.1 Dry Friction

Figure 4-21 displays data from the literature (Bodai et al [72], Fuji [68] and Deleau et al [74]) showing the change in the dry friction coefficient of wiper blades against the velocity. This data serves two purposes. The first is to establish a realistic range for the dry friction coefficient and the second is to establish the general shape of the friction curve. It can be seen that, in general, the curves show no Stribeck effect, a relatively gentle gradient and knee once static friction is overcome and then a levelling off after around 20mm/s. No significant viscous friction effect is observed.

To model the friction, the model presented in [76] is adapted to conform to the curve shape suggested in Figure 4-21. The expression for the dry coefficient of friction is

$$\mu_{dry} = \mu_{dry\_1}\left[\tanh\left(\mu_{dry\_2}v\right)\right] + \mu_{dry\_3}\left[\tanh\left(\mu_{dry\_4}v\right)\right] , \qquad (4.49)$$

where μ_{dry_i}, $i = 1, 2, 3, 4$, are the sub-coefficients of dry friction and v is the velocity of the blade. The sum of μ_{dry_1} and μ_{dry_3} determine the steady state value of μ_{dry}. The slope of the transient and settling velocity is determined by μ_{dry_2} and the initial steep rise approximating static friction is determined by μ_{dry_4}, which is given the value $10\mu_{dry\_3}$ for simplicity. Results of the simulation of equation (4.49) are shown in Figure 4-22, the parameters used in the simulations are shown in Table 4-1.

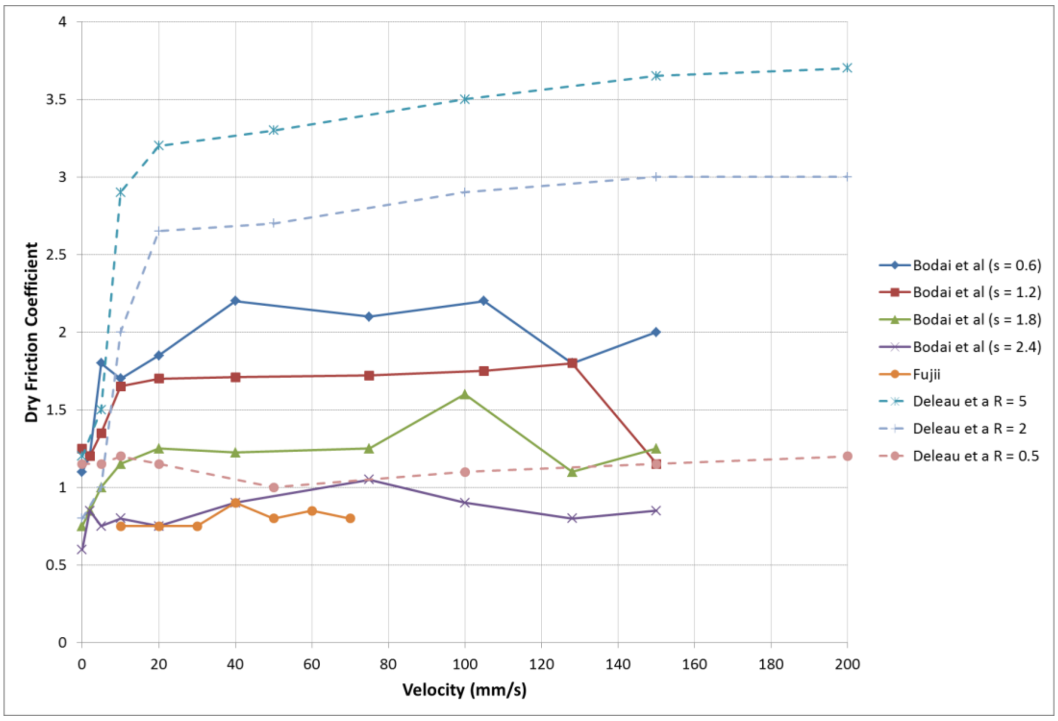


Figure 4-21: Dry Friction Coefficient against Velocity (Literature)

Table 4-1: Parameters for Dry Friction Simulation

Parameter	Friction$_1$	Friction$_2$	Friction$_3$	Friction$_4$
μ_{dry_1}	0.25	0.5	0.4	0.15
μ_{dry_2}	0.1	0.1	0.3	0.05
μ_{dry_3}	0.75	0.75	0.75	0.75
μ_{dry_4}	7.5	7.5	7.5	7.5

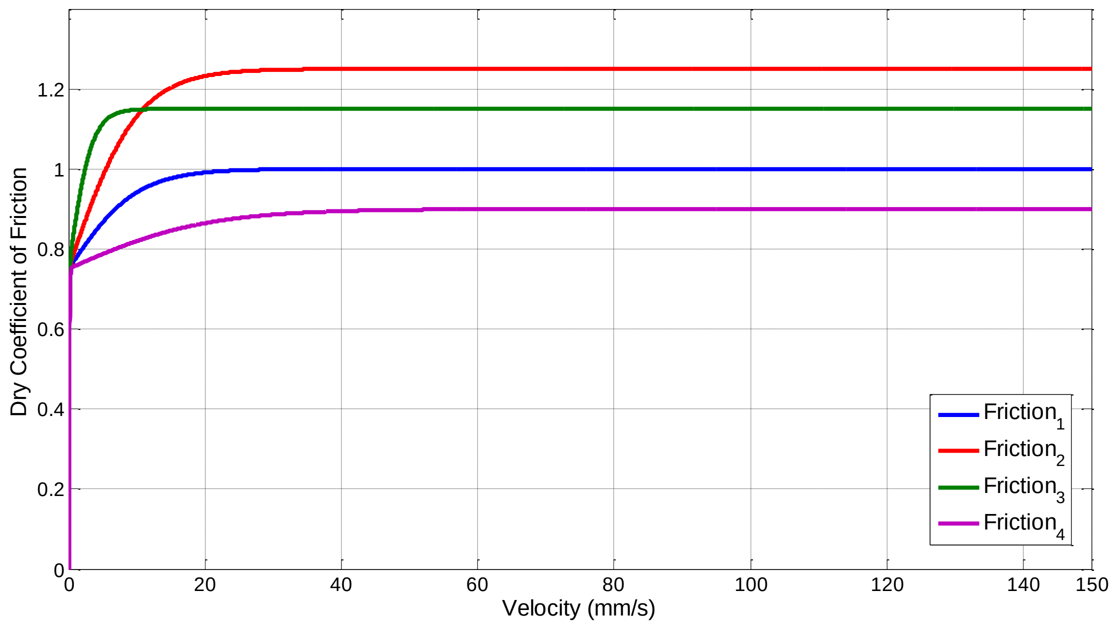


Figure 4-22: Dry Coefficient of Friction Simulation

The Simulink model used to simulate the expression in (4.49) and implement friction in the final model is shown in Figure 4-23.

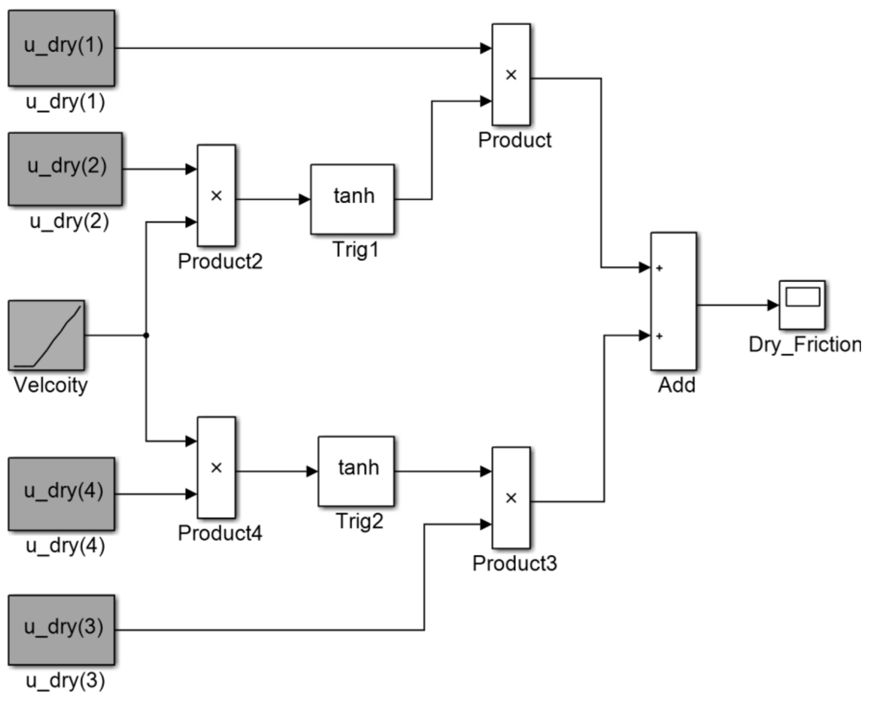


Figure 4-23: Dry Friction Coefficient Simulink Model

4.4.2 *Wet Friction*

Figure 4-24 shows plots of the wet friction coefficient against velocity for different input forces, generated using data from the literature (dash-dot plots from Bodai et al [71], solid lines from Bodai et al [72], and the dotted lines from Le Rouzic et al [99]). The plots can be used to determine realistic values of the wet friction coefficient and to determine the general shape of the friction curve. Unlike the dry friction, it can be seen that the wet friction is subject to the Stribeck effect which tends to end before the velocity reaches 100mm/s. The curve also shows a decrease in the friction coefficient as the velocity increases at higher velocities, implying a negative coefficient of viscous friction, unlike most materials. Finally, it can be seen that the wet friction coefficient decreases with applied force. This is important because the applied force is not necessarily constant and is a function of the wiper position and the aerodynamic lift. Figure 4-25 plots the friction coefficient against force at different velocities, along with linear approximations of the curves. The friction coefficient is modelled based on the method shown in [76].

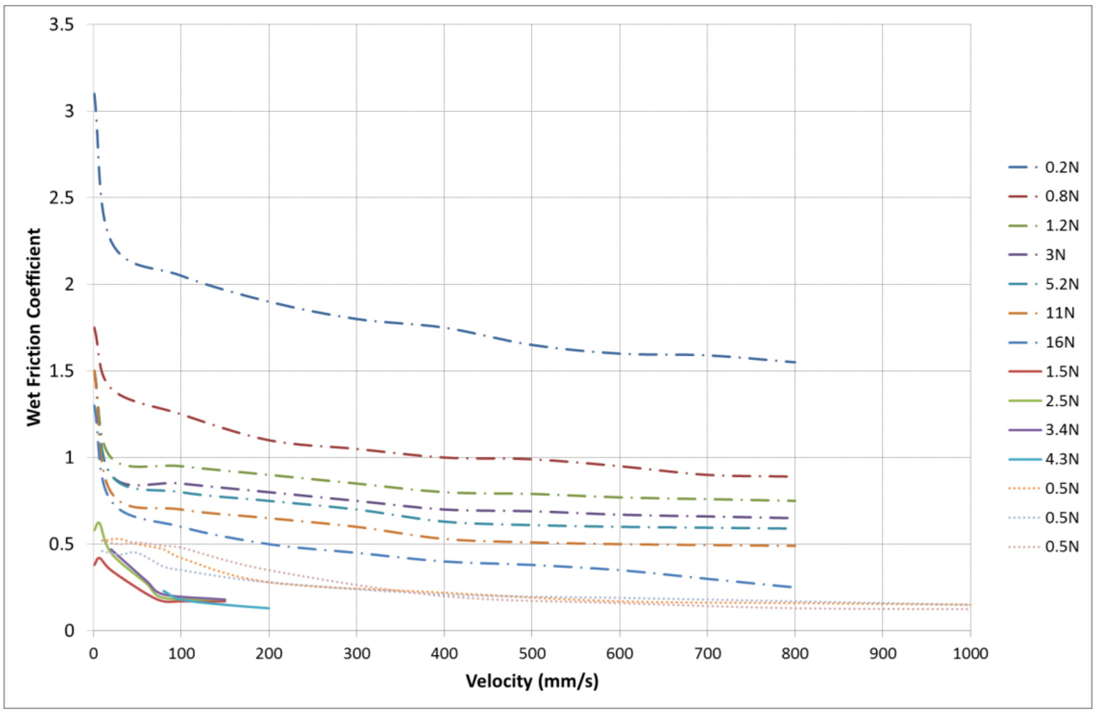


Figure 4-24: Wet Friction Coefficient vs Velocity for Different Forces (Literature)

The expression for the wet coefficient of friction is

$$\begin{aligned}\mu_{wet} = &\left[\left(\mu_{wet\_7} \times F\right) + \mu_{wet\_4}\right] \\ &\times\left\{\mu_{wet\_1}\left[\tanh\left(\mu_{wet\_2} \times v\right) - \tanh\left(\mu_{wet\_3} \times v\right)\right]\right. \\ &+\mu_{wet\_4}\left[\tanh\left(\mu_{wet\_5} \times v\right)\right] \\ &\left.-\mu_{wet\_6} \times v\right\}\end{aligned} \tag{4.50}$$

where μ_{wet_1} determines the overshoot due to the Stribeck effect, μ_{wet_2} and μ_{wet_3} determine the rise and fall rates of the Stribeck effect respectively, μ_{wet_4} determines the steady state Coulomb friction, μ_{wet_5} determines the rise rate approximating static friction, μ_{wet_6} determines the viscous friction and μ_{wet_7} determines the effect of applied force on μ_{wet}. The velocity is v and the applied force is F. For simplicity, μ_{wet_3} and μ_{wet_5} are defined as $10\mu_{wet\_1}$ and $10\mu_{wet\_4}$ respectively. Results of the simulation of equation (4.50) are shown in Figure 4-26 and the parameters used in the simulations are shown in Table 4-2.

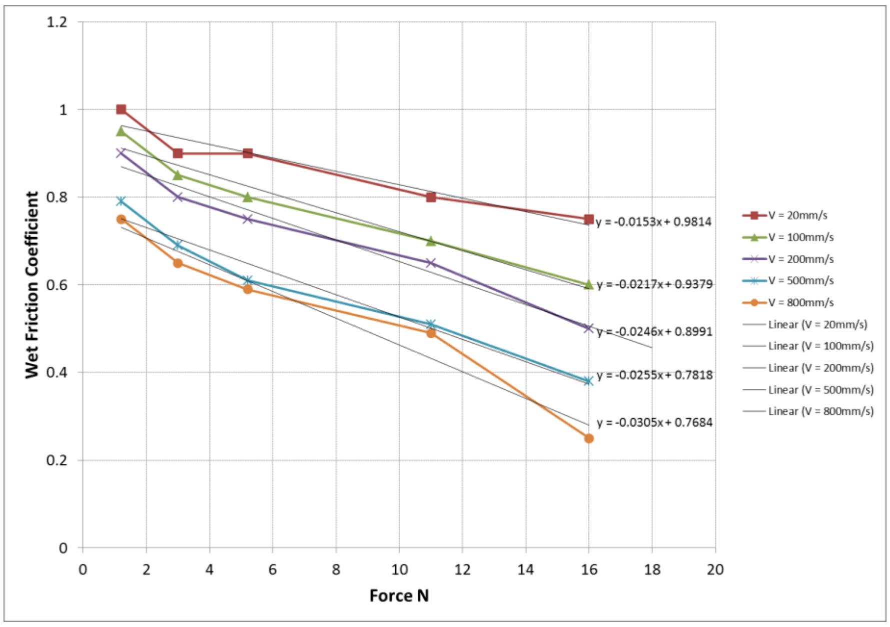


Figure 4-25: Wet Friction Coefficient vs Force for Different Velocities (Literature)

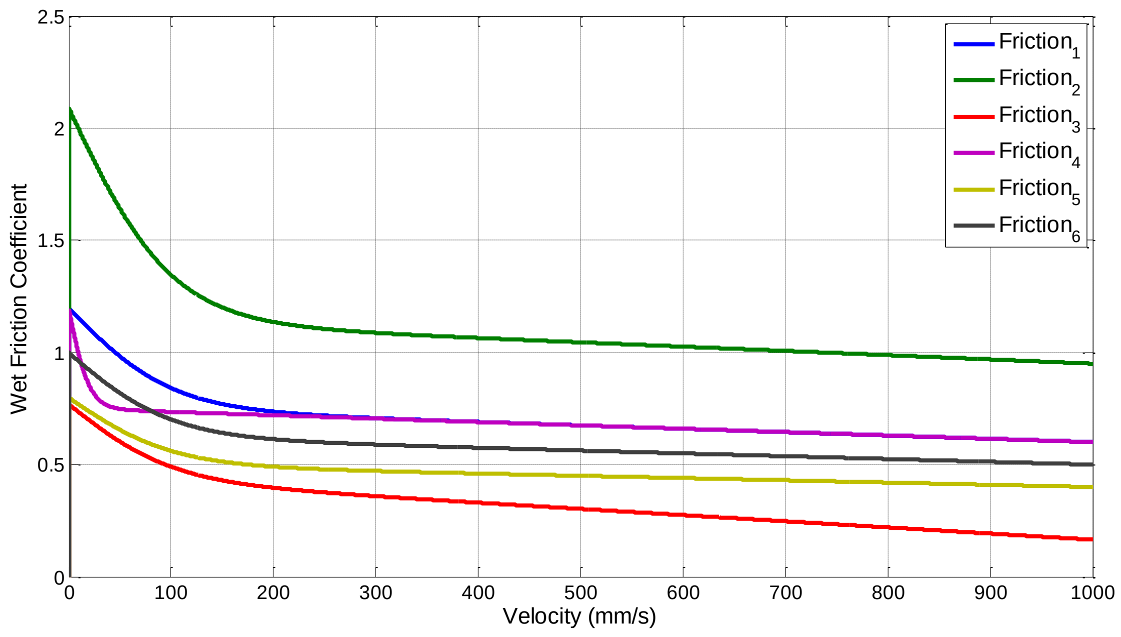


Figure 4-26: Wet Coefficient of Friction Simulation

Table 4-2: Parameters for Wet Friction Simulation

	μ_{wet_1}	μ_{wet_2}	μ_{wet_3}	μ_{wet_4}	μ_{wet_5}	μ_{wet_6}	μ_{wet_7}	$F(N)$
Friction$_1$	0.6	6	0.01	1	10	0.0002	0.025	10
Friction$_2$	1	10	0.01	1.2	12	0.0002	0.025	10
Friction$_3$	0.6	6	0.01	0.8	8	0.0005	0.025	10
Friction$_4$	0.6	6	0.05	1	10	0.0002	0.025	10
Friction$_5$	0.6	6	0.05	1	10	0.0002	0.05	10
Friction$_6$	0.6	6	0.01	1	10	0.0002	0.025	15

The Simulink model used to simulate the expression in (4.50) and implement friction in the final model is shown in Figure 4-27.

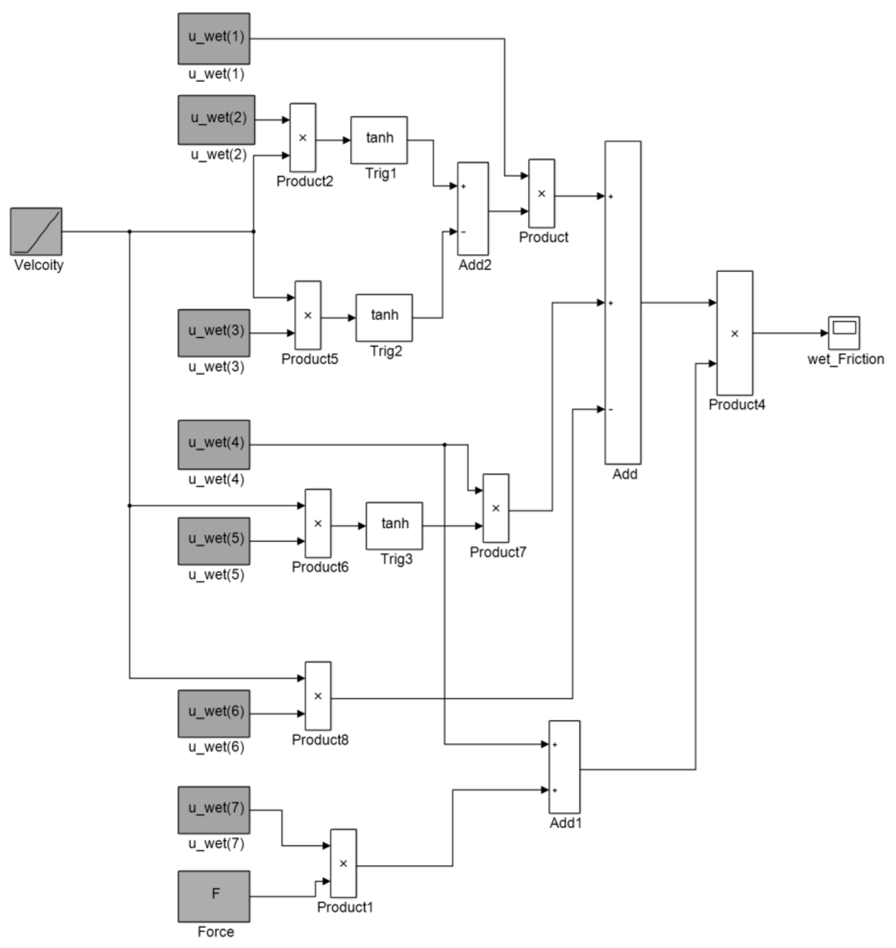


Figure 4-27: Wet Friction Coefficient Simulink Model

4.5 Aerodynamic Lift and Drag Forces

It was demonstrated in Chapter 2 that the aerodynamic forces acting on the wiper blades and arms are very complex and there exists only a small amount of data in the literature reporting their effect. To simulate using CFD software or measure in a wind tunnel the forces acting on the wiper system is impractical for this project. Therefore, a simple model based on basic fluid dynamics and using the limited data available in the literature has been developed.

Figure 4-28 shows data taken from a Korean paper [86] which gives results regarding the drag and lift coefficients of wipers at different high vehicle speeds against wiper angle. It gives an indication of realistic values of the coefficient and shows that the two wipers (driver and passenger sides) can have different coefficients (due to different sizes and being subject to different aerodynamic conditions). Also, it shows that the coefficients can change with respect to the wiper angle. In this project the coefficients are assumed to be constant, which will introduce an error.

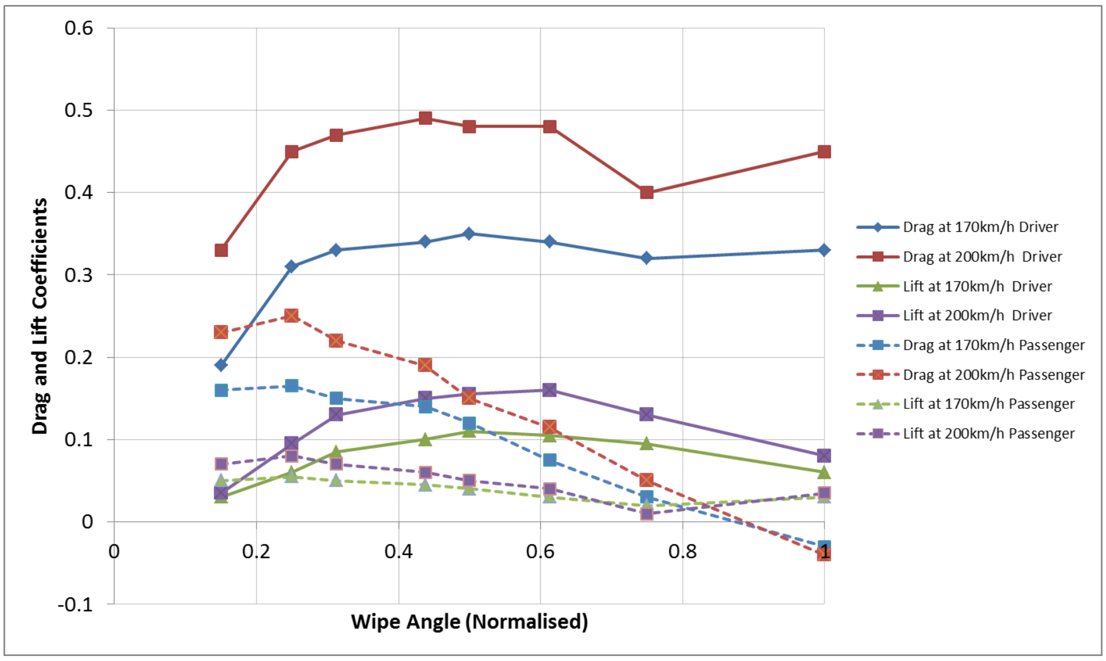


Figure 4-28: Drag and Lift Coefficients against Wiper Angle

The equations to determine drag and lift are respectively given as

$$F_D = \frac{1}{2}\rho V^2 A C_D \ , \tag{4.51}$$

$$F_L = \frac{1}{2}\rho V^2 A C_L \ , \tag{4.52}$$

where ρ is the mass density of air ($1.225\ kg/m^3$, V is the velocity of air (or the vehicle), A is the surface area of the wiper blade and C_D and C_L are the drag and lift coefficients respectively. Representative parameters derived from [86] and [84] have been used to simulate the model given in Figure 4-29. The parameters used are: $A = 0.07m^2$, V sweeps from $0ms^{-1}$ to $40ms^{-1}$, $C_D = 0.175$ and $C_L = 0.0517$ for blades without a spoiler and $C_D = 0.219$ and $C_L = -0.0694$ for blades with a spoiler. The results of the simulation are shown in Figure 4-30. Positive lift forces will act away from the windscreen, positive drag forces will act in the direction of a forward wipe, and against a reverse wipe.

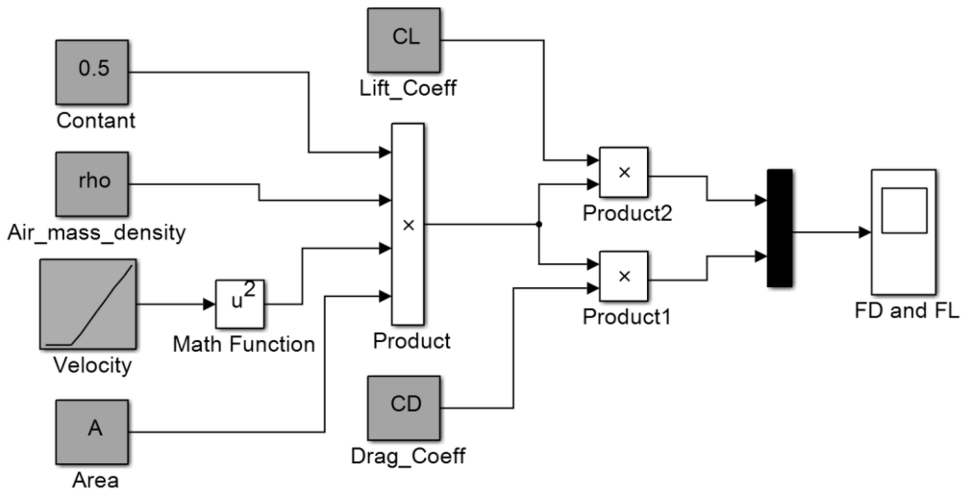


Figure 4-29: Drag and Lift Force Simulation Model

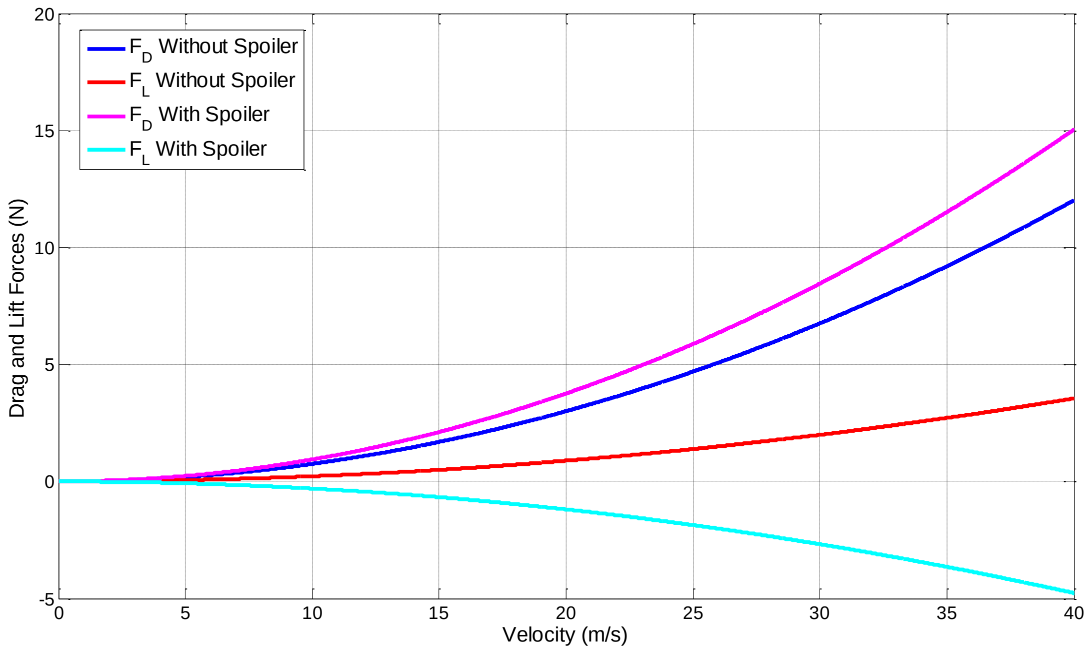


Figure 4-30: Aerodynamic Forces Simulation Results

4.6 *Whole System Simulation*

To demonstrate the operation of the system, a simulation has been performed with representative arbitrary parameters. The system is driven by a constant crank velocity of 2π rad/s, i.e. one full wipe per second. Figure 4-31 shows the model that has been simulated. The 'Environment' blocks incorporate the friction and aerodynamic forces shown above.

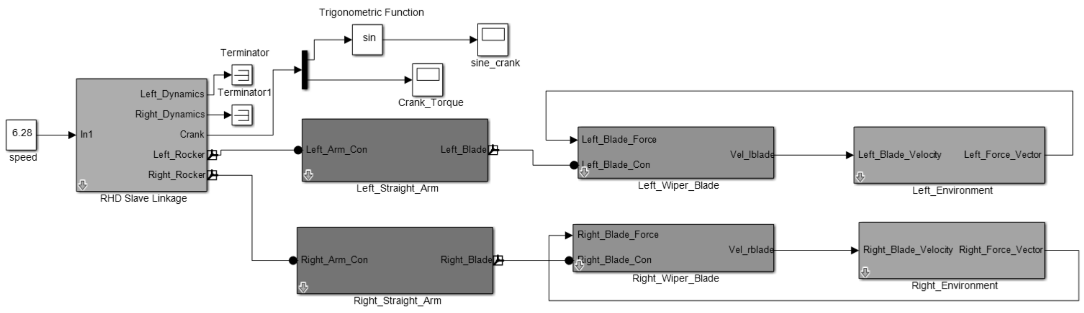


Figure 4-31: Whole SimMechanics Model Example

Figure 4-32 show the results of the simulation. The results show the linkage crank torque load that will be applied to the motor against time, with the system's conditions changing after every wipe/second. Initially, the windscreen is assumed to be wet and so the wet friction coefficient is calculated. In the first wipe, the vehicle speed is 0m/s, and after 1s the speed steps up to 20m/s. It can be seen that the overall torque drops; this is due to the lift force increasing and thus decreasing the friction force. After 2s the speed increases to 40m/s and the torque drops further. It can also be seen that the reverse wipe now develops a greater torque than the forward wipe. This is because the drag force is increasing with vehicle speed and the drag force acts against the wipers in the reverse sweep. At 3s the speed drops down to 0m/s and the conditions are the same as for the first wipe. After 4s the windscreen is assumed to be dry and thus the dry friction coefficient is calculated. It can be seen that the overall torque increases and the fundamental shape changes slightly, due to the different shape of the friction curve.

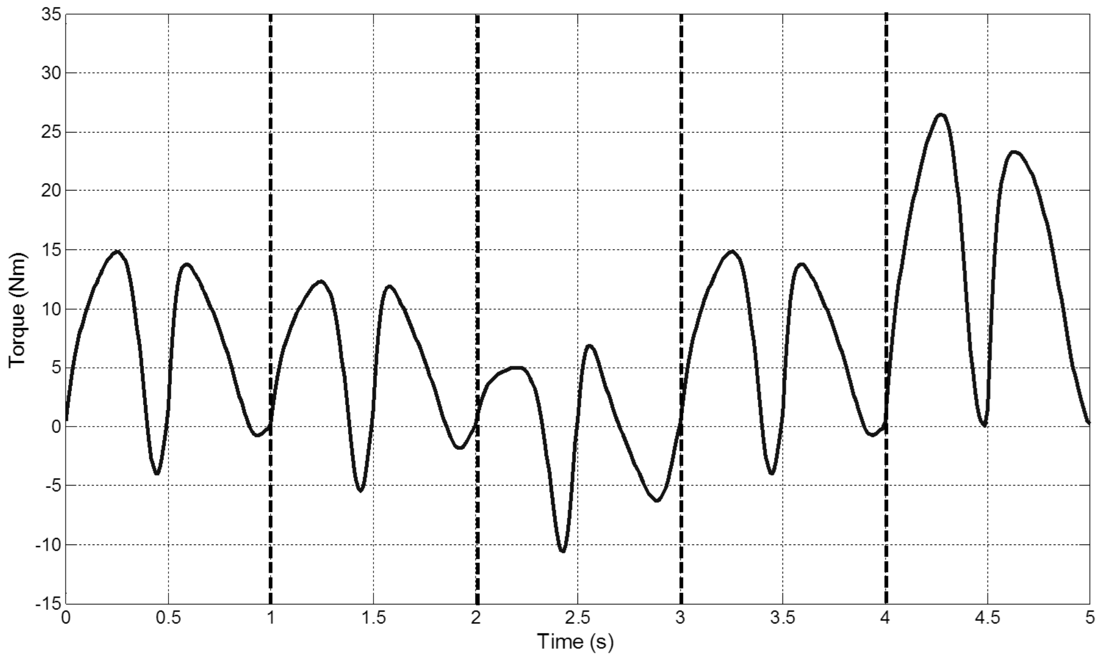


Figure 4-32: SimMechanics Model Simulation Results

Figure 4-33 shows examples of the SimMechanics visualisation tool and demonstrates the simulation solving the motion of the system.

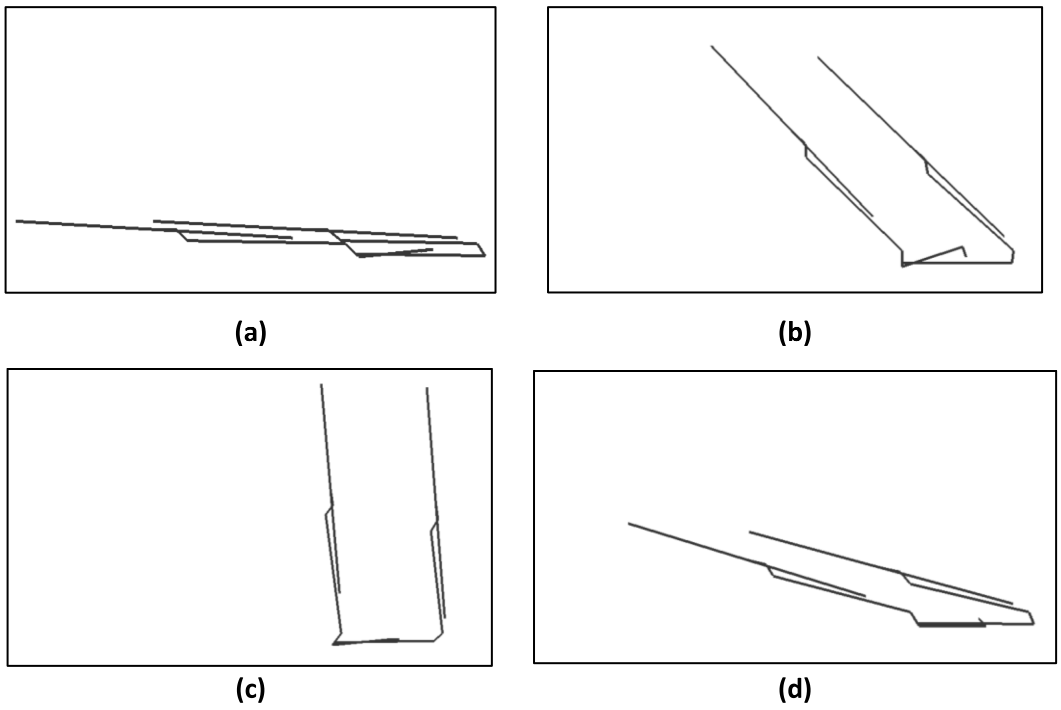


Figure 4-33: SimMechanics Model Simulation Plots

4.7 Discussion

In this chapter, two modelling strategies have been used to model the mechanical elements of the windscreen wiper system. Firstly, a planar kinematic model of the linkages, wiper arms and wiper blades using multibody dynamic methods was developed. The model is solved using a Newton-Raphson method and successfully calculates the position, velocity and acceleration analysis of the system. Different designs of linkages, arms and blades can be connected together to model different wiper system configurations. Grashof's law has been applied to the linkage systems to ensure that the dimensions input by the user are feasible. The model can be used to visually and mathematically check that the dimensions used to define the system are realistic and feasible before they are applied to the physical model, which will fail to simulate if the dimensions are incorrect. It will also be used in Chapter 6 in order to generate the look-up tables used in the HIL implementation of the model.

Secondly, a full physical model of the wiper system has been developed in SimMechanics. The physical model simulates both the kinematic elements of the system and the forces, i.e. the inertial forces, arm force on the blade, friction between the blade and windscreen and the aerodynamic forces applied to blade. Each element of the system is modelled in a modular fashion and can be connected to each other element, allowing for the synthesis of different wiper system designs. The friction model is based on data from the literature. Two models are generated, one for the dry windscreen case and one for the wet case. The friction is then applied to the blades as a resistive force. The aerodynamic model is simple and is based on the drag and lift coefficients of the wiper blades. The lift force acts upwards with respect to the windscreen and the drag force acts in the direction of a forward wipe, and against a reverse wipe.

The models developed in this chapter are validated and their unknown parameters identified in Chapter 5. Then, both models are then used to develop a real time capable HIL model in Chapter 6 and their deployment in the generic modelling tool is shown in Chapter 8.

Chapter 5 - Parameter Identification and Model Validation

5.1 Introduction

Models of the wiper motor and mechanical wiping system have been developed in the previous two chapters. The parameters of these models now need to be identified so the models can be used for ECU testing. It is not always possible or feasible to directly measure certain parameters, and in some cases no hardware is available and thus data driven methods of identification must be explored. Therefore, a variety of optimisation methods based on limited I/O data are used to estimate the parameters.

A three stage optimisation method is proposed here, using three techniques to converge on a suitable solution:

1. The wiper motor dynamics are modelled in a transfer function relating the input voltage to the output current. The transfer function coefficients are then identified, using a System Identification Tool, and are then used to obtain the motor's equivalent parameters. These parameters are then used to narrow down the search space of a Genetic Algorithm (GA).
2. A GA is designed to identify all of the parameters of the wiper motor system, i.e. the motor parameters, blade/windshield friction parameters and the force applied by the wiper arms to the wiper blades. A GA uses the principles of genetics and natural evolution to identify an optimal solution.
3. Once the GA has identified the global optimal solution, a nonlinear least squares algorithm is used to refine the solution and further improve its accuracy.

The parameters are identified, and the model validated, using data measured from the wiper test rig which is described in greater detail in Chapter 8. The data that can be measured from the test rig is the voltage applied to the wiper motor, the wiper motor armature current and the wiper motor park switch. Note that no continuous position or speed data was available. The torque applied to the motor could also not be measured. The limited amount of data available was the main motivation for using large scale stochastic identification methods such as GAs. The measured data also replicates data that can easily be measured in a real vehicle, i.e. no additional sensors are needed to implement the algorithms presented here.

Once the parameters have been estimated the simulated data from the model is graphically compared to the measured validation data to demonstrate the model's performance in replicating the real system.

The models identified in this chapter will be simplified to improve their robustness and increase their simulation speed in order for them to be suitable for real-time simulation. This is presented in Chapter 6.

5.2 Identification Methodology

The simulation model used in the parameter identification and model verification process is shown in Figure 5-1. From left to right it consists of: A control signal which controls when the motor switches on, a Control System with a relay drive circuit to control the input voltage level and speed mode of the motor, a Digital Park Switch style motor, a mechanical system made up of a right hand drive (RHD) slave driven linkage system, two straight style wiper arms and two wiper blades, and two Environment blocks implementing the friction forces. The system in Figure 5-1 models the real system used to generate the identification and validation data.

There are in total 17 parameters that must be identified. These are the eight motor parameters, three dry friction parameters, five wet friction parameters and the force applied to the blades by the arms, shown in equation (5.1).

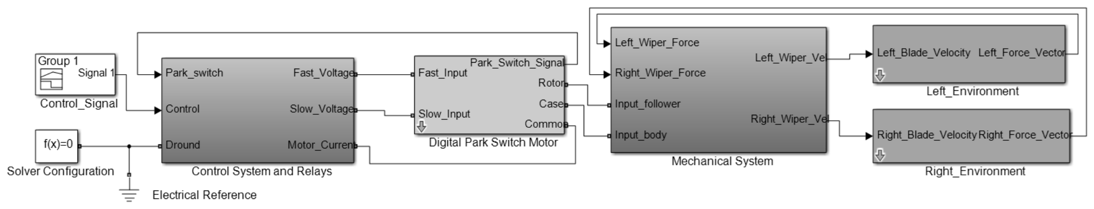


Figure 5-1: Simulation Model used for Identification

$$\begin{aligned} Motor\_param &= \begin{bmatrix} R_{slow} & R_{fast} & L_{slow} & L_{fast} & K_{slow} & K_{fast} & b & J \end{bmatrix} \\ dry\_param &= \begin{bmatrix} \mu_{dry\_1} & \mu_{dry\_2} & \mu_{dry\_3} \end{bmatrix} \\ wet\_param &= \begin{bmatrix} \mu_{wet\_1} & \mu_{wet\_2} & \mu_{wet\_4} & \mu_{wet\_6} & \mu_{wet\_7} \end{bmatrix} \\ Force &= \begin{bmatrix} F \end{bmatrix} \end{aligned} \quad . \qquad (5.1)$$

Due to the large number of parameters and the highly nonlinear nature of the mechanical load, this is a difficult identification task and requires advanced optimisation techniques. Further compounding the difficulty is the lack of any continuous position/speed data for the system. The available measurable data for identification is the voltage applied to the motor, the motor current and the motor's park switch, which pulses after each revolution.

In general, simulated data is compared to the measured data on a point by point basis. This method is suitable for comparing voltage and current, however a problem is encountered when comparing the park switches. This issue is illustrated using Figure 5-2, where the solid pulses represent measured data and the dashed pulses represent simulated data. Consider case 'a'; the measured and simulated results are almost equal, however since there is no actual crossover between the respective park switches the error produced by case 'a' will be the same as that produced by case 'b', despite 'b' being significantly less accurate. In fact, case ‘c’ which has no simulated park switches (implying that the simulated motor is moving very slowly or not at all) will have the smallest error.

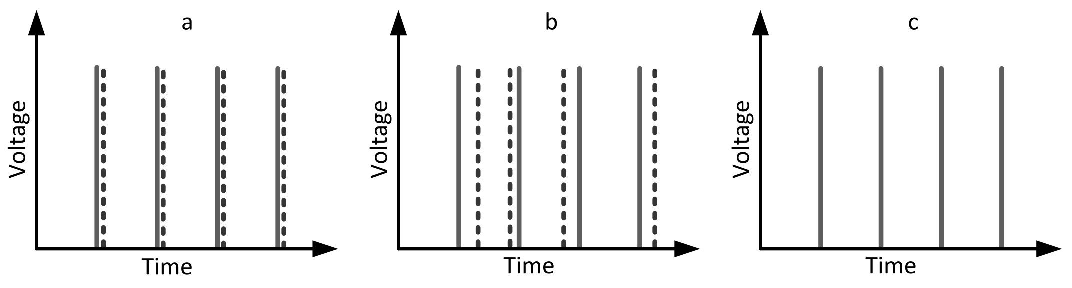


Figure 5-2: Park Switch Comparison Issue Example

To overcome this, an algorithm has been developed that detects a rising edge of the park switch and increments an output called “Park Step” by 1 on every pulse. This creates a staircase style output. The effect of the algorithm on the cases shown in Figure 5-2 is shown in Figure 5-3. It can now be seen that a point by point comparison of the data will accumulate a small error in case ‘a’, medium sized error in ‘b’ and a large error in ‘c’.

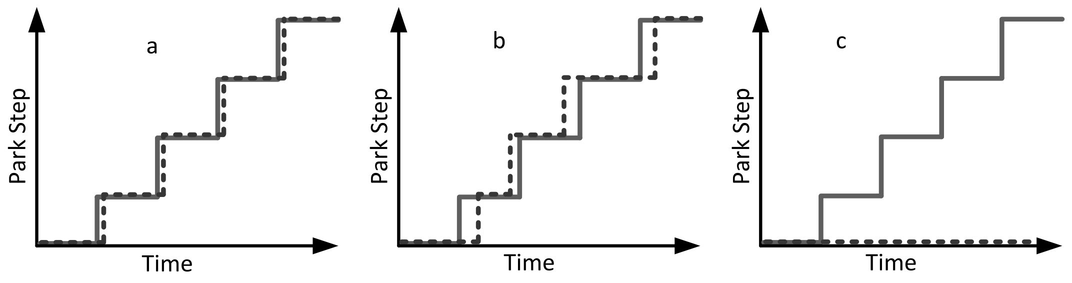


Figure 5-3: Park Step Demonstration

The model must represent the wiper system over its entire operational range and thus the data used to identify the parameters must also cover the system’s operating range. Figure 5-4 shows an example of a state space algorithm, built in Stateflow, that progresses through a range of input voltages. On entry, the state “Slow_10V” turns the motor on and sets its speed mode to slow. The voltage is set to 10V. When the park switch pulses the state “Slow_12V” is entered which sets the voltage to 12V. This continues until all of the states have been entered, at which point the motor switches off.

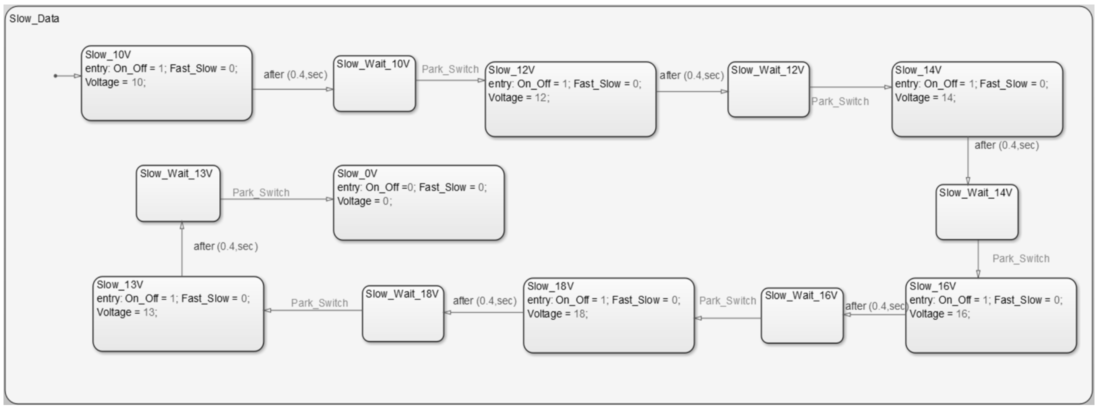


Figure 5-4: Stateflow Sub Chart for Parameters Identification Data Generation

An example of the data generated by the model shown in Figure 5-4, along with a set of validation data, is shown in Figure 5-5.

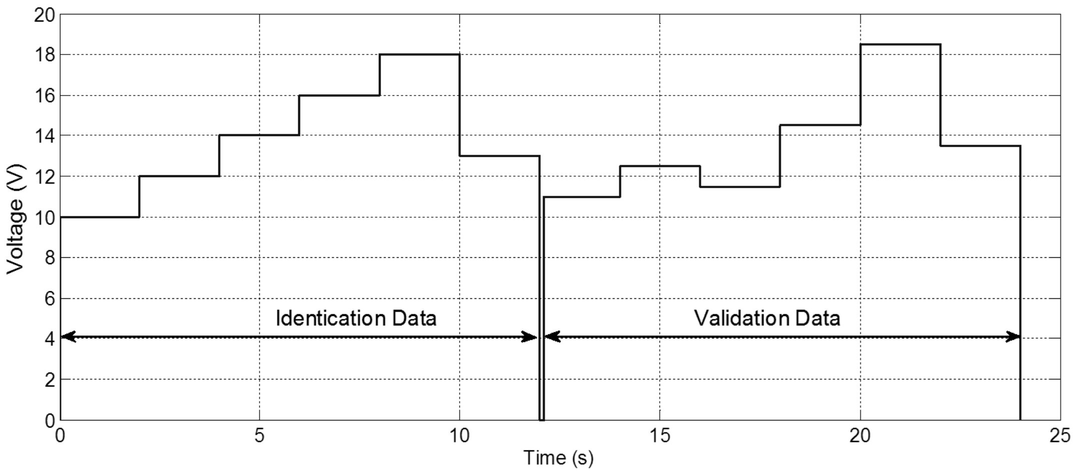


Figure 5-5: Parameter Identification and Model Validation Data Example

The parameter estimation will be performed in three stages. The first stage is to perform an initial estimation of the motor parameters by deriving a transfer function from the dynamic equations of the motor and using the System Identification Tool in Matlab to identify the transfer function parameters. These parameters are then used as a first estimate for the GA to refine. This is beneficial because it reduces the search space that the GA must explore, meaning that the population size, convergence and thus algorithm running time can be reduced. The GA is used to estimate all of the model parameters shown in equation (5.1). Once the GA has identified the parameters, a final local optimiser stage is employed to further increase the accuracy of the results. A diagram demonstrating the process is shown in Figure 5-6.

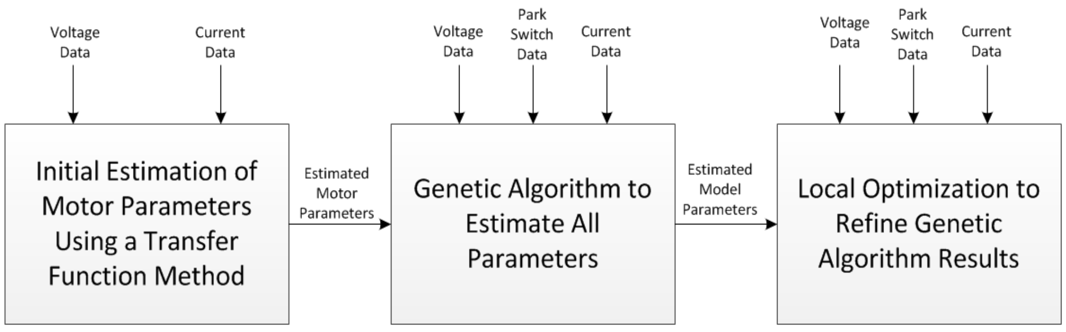

Voltage Data
Current Data
Initial Estimation of Motor Parameters Using a Transfer Function Method
Estimated Motor Parameters
Voltage Data
Park Switch Data
Current Data
Genetic Algorithm to Estimate All Parameters
Estimated Model Parameters
Voltage Data
Park Switch Data
Current Data
Local Optimization to Refine Genetic Algorithm Results

Figure 5-6: Parameter Identification Procedure

5.3 *Wiper Motor Transfer Function Identification*

Before identifying the entire system's parameters using a GA, it is beneficial to estimate the motor parameters using a transfer function identification method. Under certain assumptions, the dynamic equations of a DC motor shown in Chapter 3 can be considered to be linear. Most of these assumptions have already been made when developing the motor model, i.e. ignoring non-linear magnetisation effects and the temperature dependence of parameters such as resistance and the motor constants. The additional assumption made here is that the torque load applied to the motor by the mechanical system is directly proportional to the angular velocity of the motor's output shaft. It is clear from the work shown in Chapter 4 that this is not true and this assumption will introduce an error into the results; however it allows a transfer function to be developed and its coefficients can then be identified.

The starting point for developing the transfer function is the equations shown in (5.2)

$$\begin{aligned} L\frac{di}{dt} &= V - Ri - K\omega \\ J\frac{d\omega}{dt} &= Ki - b\omega - \gamma\omega \end{aligned} , \tag{5.2}$$

where the torque and back EMF constants are assumed to be equal and are denoted as K. The symbol γ is the constant of proportionality relating the torque load applied to the motor by the mechanical system and the angular velocity. The constant γ has the units $Nm/(rad/s)$. This is considered reasonable because the load applied to the motor is effectively zero when the velocity is zero and increases with velocity due to the inertial forces, friction and losses in the system. Taking the Laplace transform of (5.2) and assuming zero initial conditions yields

$$sLI(s) = V(s) - RI(s) - K\Omega(s) , \tag{5.3}$$

$$sJ\Omega(s) = KI(s) - b\Omega(s) - \gamma\Omega(s) . \tag{5.4}$$

By rearranging equation (5.4) an expression for $\Omega(s)$ can be found to be

$$\Omega(s) = \frac{KI(s)}{sJ + b + \gamma} , \tag{5.5}$$

and by substituting (5.5) into (5.3) the angular velocity term can be removed

$$sLI(s) = V(s) - RI(s) - \frac{K^2 I(s)}{sJ + b + \gamma} . \tag{5.6}$$

Equation (5.6) can now be rearranged into the form of a standard second order transfer function relating the motor current to the input voltage:

$$\frac{I(s)}{\mathrm{V}(s)}=\frac{1}{L}\frac{s+\left(\frac{b+\gamma}{J}\right)}{s^2+s\left(\frac{b+\gamma}{J}+\frac{R}{L}\right)+\frac{R}{L}\left(\frac{b+\gamma}{J}+\frac{K^2}{RJ}\right)}\ . \tag{5.7}$$

The transfer function equation (5.7) has been simulated under the same conditions as the motor models in Chapter 3 and by comparing Figure 5-7 with Figure 3-10 and Figure 3-11 it can be seen that the results are the same, showing that equation (5.7) is valid.

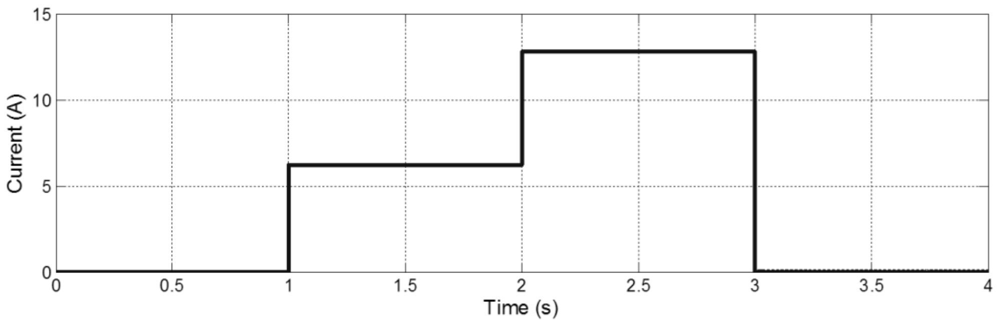


Figure 5-7: Motor Transfer Function Simulation Results

Equation (5.7) is then expressed as

$$G(s)=K_{sys}\frac{s+t_r}{s^2+s\left(t_r+\frac{1}{t_{elec}}\right)+\frac{1}{t_{elec}}\left(t_r+\frac{1}{t_{mech}}\right)}\ , \tag{5.8}$$

where the transfer function coefficients are defined in Table 5-1.

Table 5-1: Motor Transfer Function Coefficients Definition

Coefficient	Equivalent Parameters	Unit (Laplace Domain)	Description
K_{sys}	$1/L$	A/V	System gain
t_{elec}	L/R	s	Electrical time constant
t_{mech}	RJ/K^2	s	Mechanical time constant
t_r	$(b+\gamma)/J$	$1/s$	Load Constant

It was shown in Chapter 3 that a wiper motor can be expressed using eight parameters and that the dynamics of the motor are different in its fast and slow modes. This means that the parameters for two dynamic equations need to be found, i.e:

$$G_{slow}(s) = K_{sys\_slow} \frac{s + t_{r\_slow}}{s^2 + s\left(t_{r\_slow} + \frac{1}{t_{elec\_slow}}\right) + \frac{1}{t_{elec\_slow}}\left(t_{r\_slow} + \frac{1}{t_{mech\_slow}}\right)} , \tag{5.9}$$

$$G_{fast}(s) = K_{sys\_\mathrm{fast}} \frac{s + t_{r\_fast}}{s^2 + s\left(t_{r\_fast} + \frac{1}{t_{elec\_fast}}\right) + \frac{1}{t_{elec\_fast}}\left(t_{r\_fast} + \frac{1}{t_{mech\_fast}}\right)} . \tag{5.10}$$

Since parameters b and J are mechanical they remain the same in both fast and slow modes. Combined with the assumption that γ is constant it can be seen that $t_{r\_slow} = t_{r\_fast}$ and identifying the coefficients of (5.9) and (5.10) will yield 7 equations to identify 8 motor parameters. To make the number of unknowns equal to the number of equations, the fact that the dominant mechanical force in the system is the torque load applied by the mechanical system is used and it is therefore assumed that $b + \gamma \approx \gamma$. Thus b is eliminated.

By way of an example to demonstrate the ability of this method to identify motor parameters, a simulated motor with parameters shown in Table 5-2 has been used to generate data to input into the Matlab System Identification tool. The results are also shown in Table 5-2. The constant γ was assumed to be $0.002\, Nm/(rad/s)$. The identified transfer functions were

$$G_{slow}(s) = \frac{666.6s + 6667}{s^2 + 1343s + 1.667 \times 10^4} , \tag{5.11}$$

$$G_{fast}(s) = \frac{1000s + 1\times10^4}{s^2 + 1010s + 1.125\times10^4} \quad . \tag{5.12}$$

The Error column in Table 5-2 shows the absolute difference between the target system and estimated system results. It can be seen that, under ideal simulation conditions, the motor parameters can be correctly identified using this method.

Table 5-2: Motor Transfer Function Coefficient Identification (Simulated)

Simulated Motor Parameters	Identified Transfer Function Coefficients	Estimated Motor Parameters	Error
$R_{slow} = 2\Omega$	$t_{elec\_slow} = 7.502\times10^{-4}s$	$R_{slow} = 2\Omega$	0
$R_{fast} = 1\Omega$	$t_{elec\_fast} = 1\times10^{-3}s$	$R_{fast} = 1\Omega$	0
$L_{slow} = 1.5\times10^{-3}H$	$t_{mech\_slow} = 0.399s$	$L_{slow} = 1.5\times10^{-3}H$	0
$L_{fast} = 1.0\times10^{-3}H$	$t_{mech\_fast} = 0.8s$	$L_{fast} = 1.0\times10^{-3}H$	0
$K_{slow} = 0.01\frac{V}{(rad/s)}$	$t_{r\_slow} = 10.0015s^{-1}$	$K_{slow} = 0.01\frac{V}{(rad/s)}$	0
$K_{fast} = 0.005\frac{V}{(rad/s)}$	$t_{r\_fast} = 10s^{-1}$	$K_{fast} = 0.005\frac{V}{(rad/s)}$	0
$J = 2\times10^{-5}kgm^2$	$K_{sys\_slow} = 666.6\,A/V$	$J = 2\times10^{-5}kgm^2$	0
$b = 0\frac{Nm}{(rad/s)}$	$K_{sys\_fast} = 1000\,A/V$	$b = 0\frac{Nm}{(rad/s)}$	0

Now that the method has been shown to work, real data is input into the System Identification Tool. The data for the slow transfer function (equation (5.9)) is shown in Figure 5-8 and the data for the fast transfer function (equation (5.10)) is shown in Figure 5-9. In both figures, the blue line shows the raw data and the red line shows the data once a filter has been applied.

The identified transfer functions are given below,

$$G_{slow}(s) = \frac{89.47s + 958.8}{s^2 + 96.33s + 4733} \quad , \tag{5.13}$$

$$G_{fast}\left(s\right)=\frac{62.97s+1326}{s^{2}+74.7s+6245}\ . \tag{5.14}$$

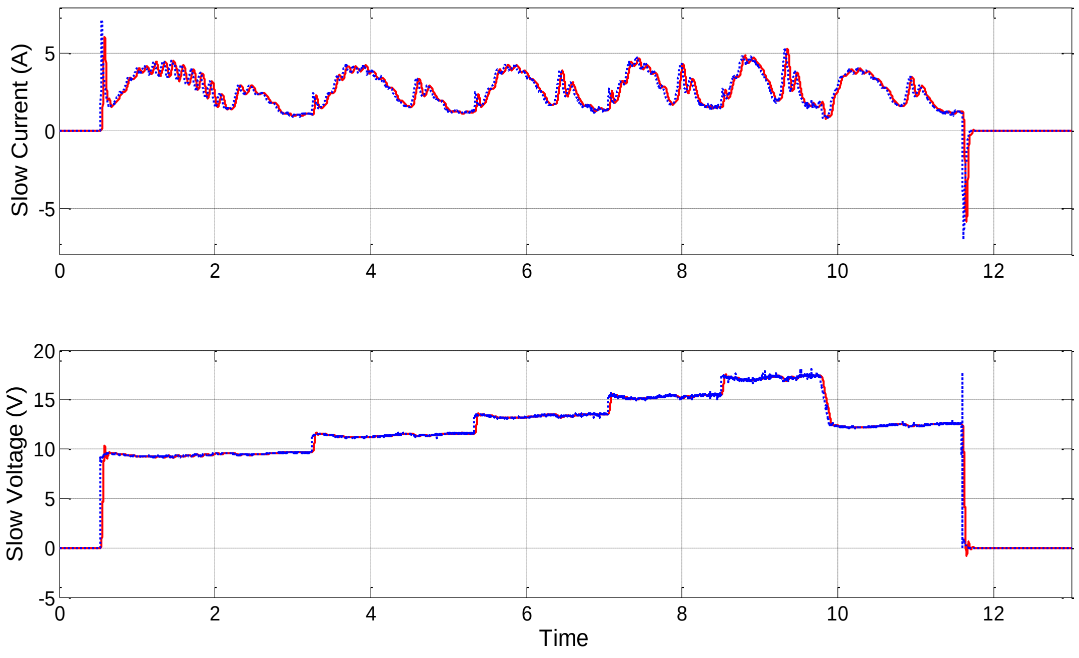


Figure 5-8: Transfer Function Parameters Identification Input Data (Slow Motor)

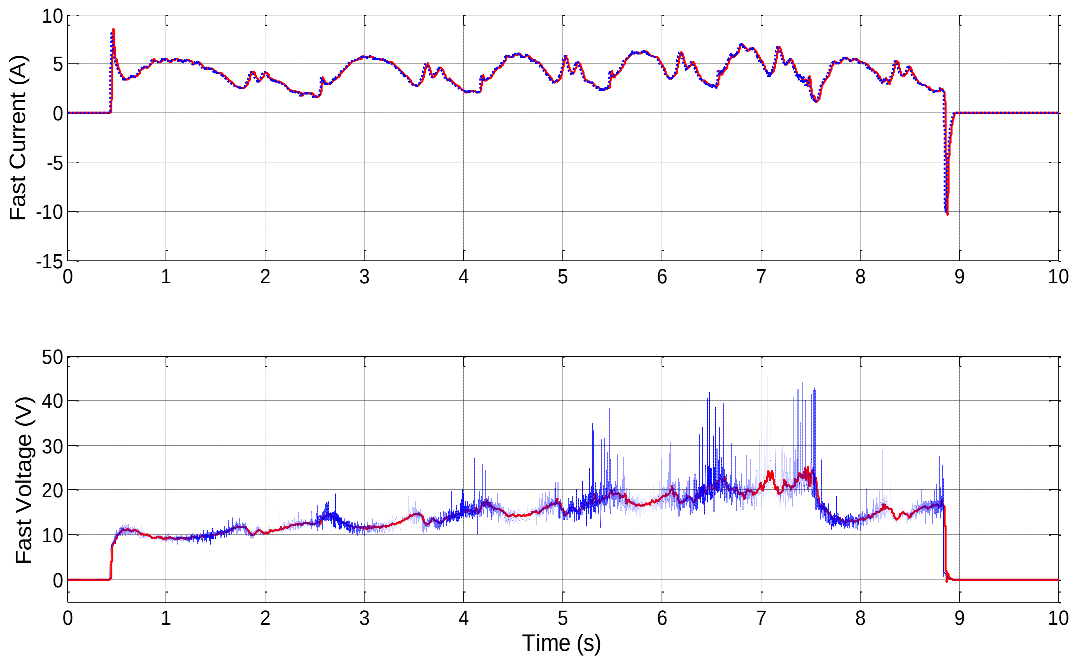


Figure 5-9: Transfer Function Parameters Identification Input Data (Fast Motor)

The results of the identification are shown in Figure 5-10 and Figure 5-11 for the fast and slow modes respectively. The dotted lines represent the real validation current data and the solid lines are the outputs of the transfer functions shown in equations (5.13) and (5.14). It can be seen that the slow mode transfer function successfully identifies the average currents and the transients. The fast mode transfer function is less successful, likely due to the increased noise on the identification data.

The motor transfer function coefficients and the associated equivalent parameters are shown in Table 5-3. As in the simulated example, it is assumed that $b=0$. Based on the work done in Chapter 4 and results given in the literature [49], the load torque applied to the motor's gear is assumed to be $3Nm$ at a velocity of 2π radians and taking into account the gear ratio of the motor (63), the value of γ can be calculated as

$$\begin{gathered} T_{L\_shaft} = \gamma \omega_{shaft} \\ \frac{3}{63} = \gamma\left(63 \times 2\pi\right) \qquad , \qquad (5.15) \\ \gamma = \frac{3}{63 \times 126\pi} = 1.2 \times 10^{-4}\left(Nm/\left(rad/s\right)\right) \end{gathered}$$

where T_{L_shaft} is the torque applied to the motor shaft and ω_{shaft} is the angular velocity of the shaft.

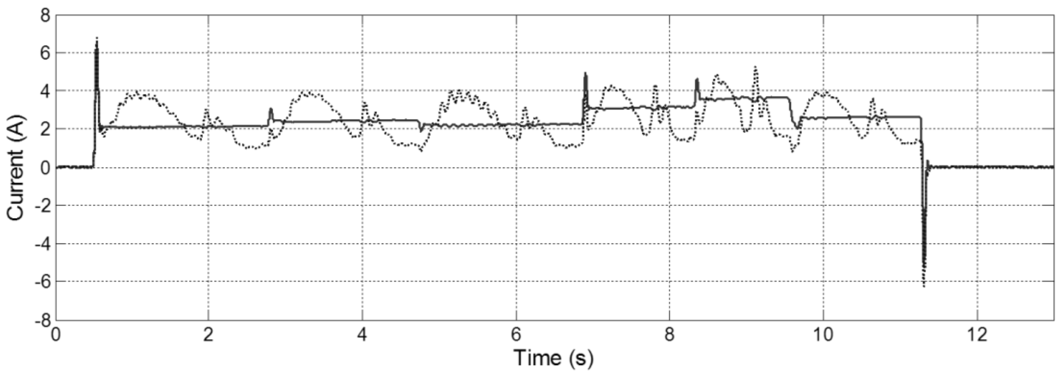


Figure 5-10: Transfer Function Parameters Identification Results (Slow)

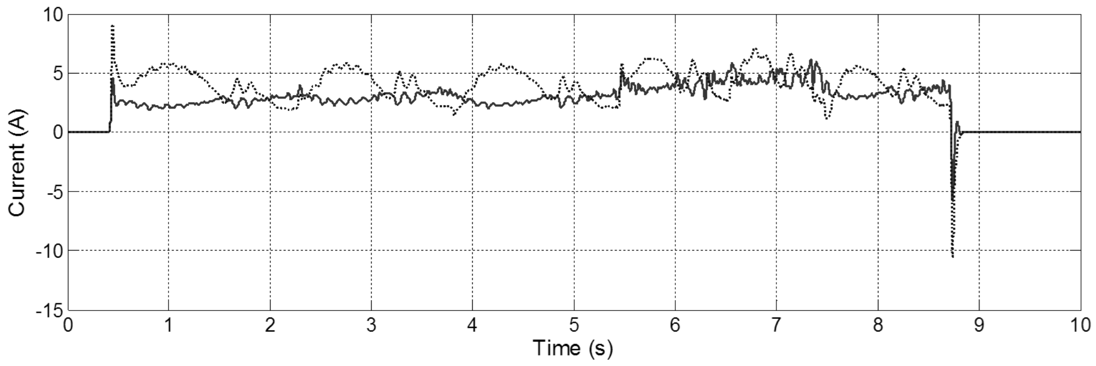


Figure 5-11: Transfer Function Parameters Identification Results (Fast)

Table 5-3: Motor Transfer Function Coefficient Identification (Real Data)

Identified Transfer Function Coefficients	Estimated Motor Parameters
$t_{elec\_\mathrm{slow}} = 0.0117s$	$R_{slow} = 0.9569\Omega$
$t_{elec\_fast} = 0.0186s$	$R_{fast} = 0.8519\Omega$
$t_{mech\_slow} = 0.0224s$	$L_{slow} = 0.0112H$
$t_{mech\_fast} = 0.0105s$	$L_{fast} = 0.0159H$
$t_{r\_slow} = 10.7164s^{-1}$	$K_{slow} = 0.0219V/(rad/s)$
$t_{r\_fast} = 21.0543s^{-1}$	$K_{fast} = 0.0215V/(rad/s)$
$K_{sys\_slow} = 89.4688\,A/V$	$J_{slow} = 1.12\times10^{-5}kgm^2$
$K_{sys\_fast} = 62.9685\,A/V$	$J_{fast} = 5.70\times10^{-6}kgm^2$

Due to the different values of t_r for the fast and slow modes (caused by the assumption that γ is the same in the fast and slow modes when in reality it will be higher when the motor is running faster), two values of J are calculated. Also, the value of L_{fast}

is higher than L_{slow} , which violates the inequality shown in equation (3.32). However, it can be observed in Figure 5-11 that the switching transient in the motor's fast mode are significantly underestimated, implying that the estimated value of L_{fast} is actually too high.

The validity of these estimated parameters are now examined by comparing them to more accurately measured parameters.

5.4 Wiper Motor Parameter Measurement

In order to determine the validity of the parameters identified above, the equivalent motor parameters have been directly measured/derived. This was achieved by removing the linkage system from a wiper motor so zero torque load tests could be performed.

5.4.1 Armature Resistance

The values of the armature resistances were obtained using a Multimeter to measure the resistance between the slow input and common output for R_{slow} and the fast input and common output R_{fast}. The values measure were

$$\begin{aligned} R_{slow} &= 1\Omega \\ R_{fast} &= 0.8\Omega \end{aligned} . \tag{5.16}$$

5.4.2 EMF and Torque Constants

To calculate the back EMF and torque constants, the following equation is used

$$V = R_a I_a + L_a \frac{dI_a}{dt} + K_e \omega \ . \tag{5.17}$$

By assuming a constant value of armature current, as is the case for a constant load and input voltage, equation (5.17) can be simplified and rearranged to give an expression for K_e:

$$K_e = \frac{V - R_a I_a}{\omega} \ . \tag{5.18}$$

A range of constant voltages, V, are input into the system and the corresponding currents, I_a, are measured. The average angular velocity of the motor's gear can be measured using the park switch. This is multiplied by the gear ratio of 63 to obtain the angular velocity, ω, of the motor shaft. The results are shown in Table 5-4. The average of three measurements for both constants is taken as the final value.

Table 5-4: Motor Back EMF Constant Measurements

Motor State	Voltage (V)	Current (A)	Velocity (rad/s)	K_e (V/(rad/s))	Average K_e (V/(rad/s))
Slow	12	1.450	230.38	0.045794	0.047163
	14	1.605	245.21	0.050548	
	16	1.708	316.56	0.045147	
Fast	12	2.537	322.96	0.030872	0.029981
	14	2.785	394.40	0.029847	
	16	2.949	466.78	0.029223	

5.4.3 Armature Inductance

To measure the inductance, a step input voltage 5 to 0V is applied to the motor whilst the mechanical output is locked. Locking the output will cause the term $K_e\omega$ in equation (5.17) to equal zero, and thus (5.17) can be written in the standard form of a 1st order system

$$\frac{V}{R_a} = I_a + \frac{L_a}{R_a}\frac{dI_a}{dt} \ , \tag{5.19}$$

and an expression for the rise/fall in current due to an step voltage is

$$I_a = \frac{V}{R_a}\left(1 - e^{-\frac{t}{L_a/R_a}}\right) , \tag{5.20}$$

where L_a / R_a is the time constant of the circuit and is defined as the time taken for the current to fall to 36.8% of its original value. The time constant can be measured and since R_a is known, L_a can be calculated. Table 5-5 shows the results of the inductance measurement.

Table 5-5: Motor Armature Inductance Measurements

	Step Time (s)	36.7% Time (s)	Time Constant (s)	Inductance (H)
Slow	3.720	3.278	0.008	0.0080
Fast	6.651	6.659	0.008	0.0064

5.4.4 Motor Inertia

Methods for measuring the inertia generally involve measuring the mechanical acceleration of motor shaft. However, due to the construction of the motor, it was impractical to attach an encoder to the shaft. Therefore, an estimation of the inertia was performed by taking a motor apart and weighing the armature, shaft and windings which had a combined weight of 0.275kg. The radius of the armature is 2cm. Approximating the structure as cylinder rotating about its central axis allows the inertia to be determined using

$$J = \frac{1}{2} m r^2 = 5.5 \times 10^{-5} kgm^2 \ , \tag{5.21}$$

where m is the mass of the cylinder and r is its radius.

5.4.5 Refining Measurements and Damping Coefficient Estimation

The damping parameter, b , could not be measured and is thus identified using a Nonlinear least squares method available in Simulink. The values of the motor's other parameters are also refined in the same estimation procedure. The parameters were identified against the measured data that switched between the slow and fast motor modes at a constant voltage, and were validated against a dataset at different constant voltages. The results showing a comparison of the real data and simulated data is shown in Figure 5-12. It can be seen that the simulation results closely match the measure data.

The differences between the measured data and simulated data are shown in Figure 5-13. It can be seen that the biggest errors occur during switch on and switch off. However it should be noted that the size and shape of the transients are closely matched, however due to a small error in the speed of the motor they occur at slightly different times.

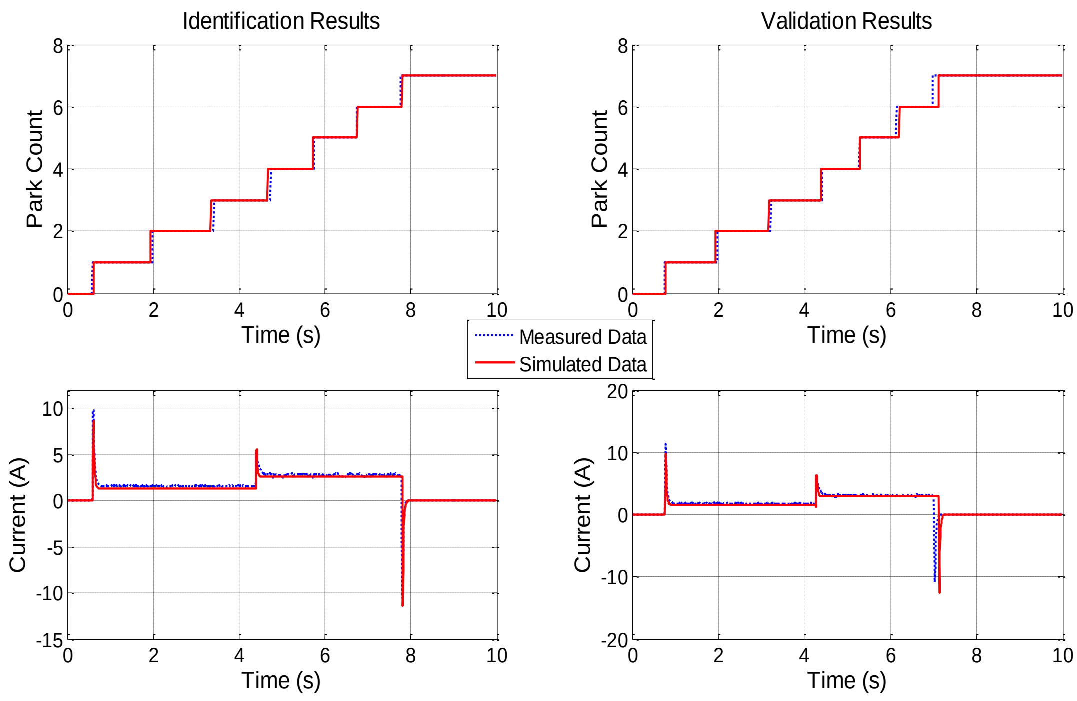


Figure 5-12: Wiper Motor Parameter Identification Comparison Results

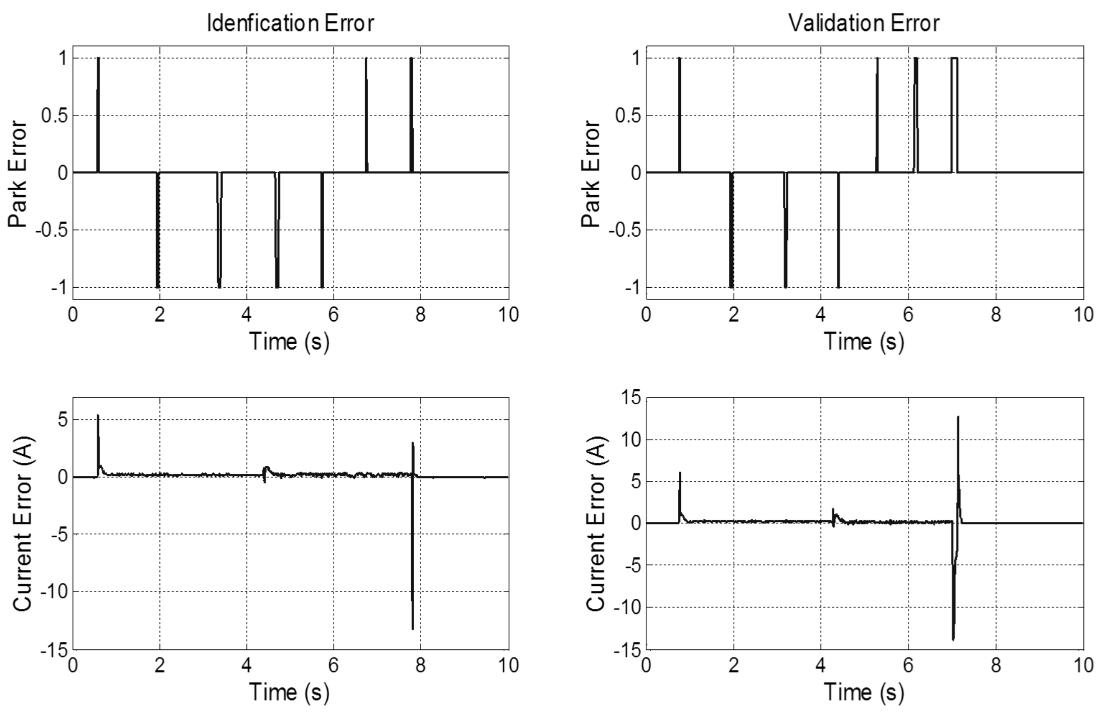


Figure 5-13: Wiper Motor Parameter Identification Residuals

The final parameters are given in column 1 of Table 5-6. They are also compared to the parameters identified using the transfer function method. It can be seen that most of the parameters are relatively close, showing that if it isn't feasible to measure the parameters of the motor directly (i.e. if only data is available) the transfer function method is suitable for finding close parameters to reduce the GA search space. There is a significant error of an order of magnitude in the inductance estimation, with the transfer function method greatly overestimating the inductance. This should be taken into account if using these values to set the search space of a GA. It should be noted Figure 5-7 shows that the transfer function parameters tend to underestimate the switching transients, implying an overestimation of the inductances.

Table 5-6: Measured and Estimated Motor Parameters Comparison

Measured and Refined Parameters	Transfer Function Estimated Parameters	Difference (Measured – Estimated)
$R_{slow} = 1.1704\Omega$	$R_{slow} = 0.9569\Omega$	0.2135Ω
$R_{fast} = 0.7005\Omega$	$R_{fast} = 0.8519\Omega$	-0.1514Ω
$L_{slow} = 0.0019H$	$L_{slow} = 0.0112H$	$-0.0093H$
$L_{fast} = 0.004H$	$L_{fast} = 0.0159H$	$-0.0119H$
$K_{slow} = 0.0442V/(rad/s)$	$K_{slow} = 0.0219V/(rad/s)$	$0.0223V/(rad/s)$
$K_{fast} = 0.0321V/(rad/s)$	$K_{fast} = 0.0215V/(rad/s)$	$0.0223V/(rad/s)$
$J = 1.9967\times10^{-5}kgm^2$	$J_{slow} = 1.12\times10^{-5}kgm^2$	$8.767\times10^{-6}kgm^2$
$bm = 1.084\times10^{-4}\,Nm/(rad/s)$	$bm = NA$	NA

Based on the results shown in Table 5-6, the search area of the GA is reduced using the following rules: if the parameter identified by the Transfer Function Estimation method is defined as U and the parameter identified by the GA is $\hat{U}$, then search space of the GA is defined as

$$0.5U \le \hat{U} \le 2U \tag{5.22}$$

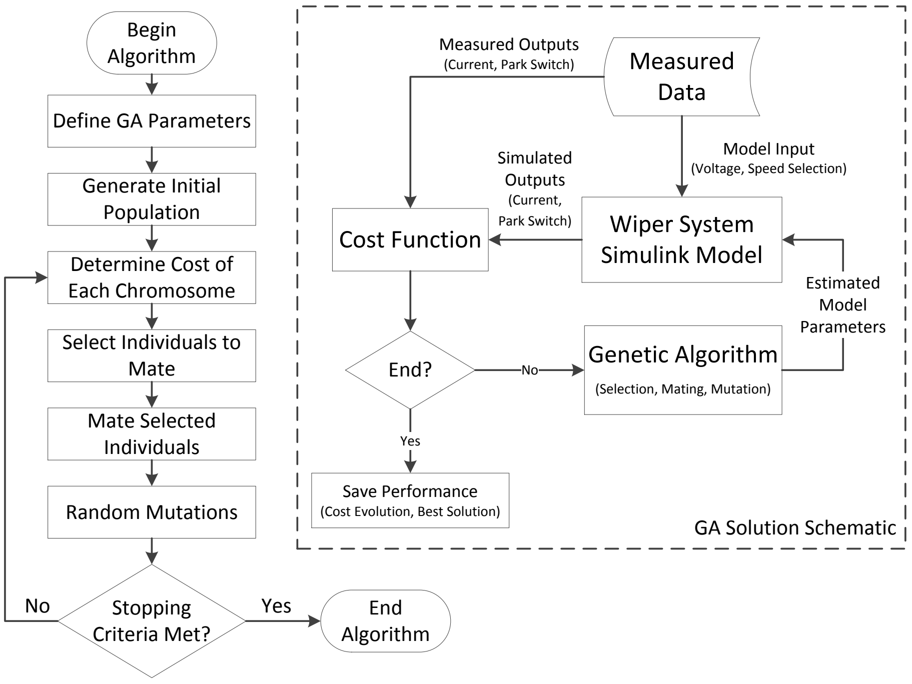
apart from the inductances, which is $0.25U \le \hat{U} \le 4U$ to account for the larger error between the estimated and measure parameters.

5.5 Genetic Algorithm Identification of System Parameters

5.5.1 Genetic Algorithm Theory

Genetic algorithms are a form of optimization algorithm that imitate the process of natural evolution in order to reduce or maximise a cost function (also known as a fitness function; the terms are henceforth used interchangeably). A GA causes individuals in a population to evolve towards an optimal solution across a number of generations by selecting and combining individuals with high fitness scores to form a new generation. A flow diagram of a classic genetic algorithm [100] is shown in Figure 5-14 along with a schematic [50] of the solution method, showing how the GA, simulation model, cost function and measured data interact. Detailed descriptions of each element of the flow diagram are given in this section.

The Matlab code to implement the GA is given in Appendix E and is based on the continuous GA given in [100].



Begin Algorithm
Define GA Parameters
Generate Initial Population
Determine Cost of Each Chromosome
Select Individuals to Mate
Mate Selected Individuals
Random Mutations
Stopping Criteria Met?
No
Yes
End Algorithm
Measured Outputs (Current, Park Switch)
Measured Data
Model Input (Voltage, Speed Selection)
Simulated Outputs (Current, Park Switch)
Cost Function
Wiper System Simulink Model
Estimated Model Parameters
End?
No
Genetic Algorithm (Selection, Mating, Mutation)
Yes
Save Performance (Cost Evolution, Best Solution)
GA Solution Schematic

Figure 5-14: Genetic Algorithm Flow Diagram and Schematic

The GA acts upon a population of individuals called chromosomes. A chromosome is a vector made up of a number of genes, which in this case are the unknown parameters of the wiper system model. The chromosome is shown in equation (5.23), whose elements are defined in equation (5.1)

$$Chromosome = \{Motor\_param \quad dry\_param \quad wet\_param \quad F\} \, . \qquad (5.23)$$

Each chromosome in a generation is input into the simulation model, which is simulated based on voltage and control inputs from measured data. The simulated motor current and park switch are compared to measured data using the cost function. The value of a chromosome's cost function, along with the GA parameters, determines how likely it is that it will be retained and its genes passed on to the next generation. If a chromosome's cost is less than a pre-determined minimum, or if a maximum number of generations are reached, the algorithm will terminate and output the best solution.

5.5.2 Cost Function

The fitness of a chromosome is determined by comparing the simulated motor currents and park switches that it produces to measured data under the same conditions. The two conditions are 1) perform one wipe each at 12V, 14V and 16V in the motor's slow mode and in dry conditions and 2) perform one wipe each at 12V, 14V and 16V in the motor's fast mode and in wet conditions. The cost function has two objectives: Minimise the summed squared errors in the currents (slow/dry and fast/wet modes) and minimise the summed squared errors in the park switches (slow/dry and fast/wet modes). The final cost is calculated by summing the two individual costs and dividing it by the number of measured data points. The functions that perform this are shown in equations (5.24) and (5.25) for the current and park switch objectives respectively and equation (5.26) for the total cost:

$$fitness_{current} = \sum_{i=1}^{n} \left\{ \left(I_{slow}(i) - I'_{slow}(i) \right)^2 + \left(I_{fast}(i) - I'_{fast}(i) \right)^2 \right\} \ , \tag{5.24}$$

$$fitness_{park} = \sum_{i=1}^{n} \left\{ \left(\xi_{slow}(i) - \xi'_{slow}(i) \right)^2 + \left(\xi_{fast}(i) - \xi'_{fast}(i) \right)^2 \right\} \ , \tag{5.25}$$

$$fitness_{total} = \left(\sum_{i=1}^{n} \left\{ fitness_{current}(i) + fitness_{park}(i) \right\} \right) \Big/ n \ . \tag{5.26}$$

where I_{slow} and I_{fast} are the measured motor currents respectively and likewise I'_{slow} and I'_{fast} are the simulated currents. Similarly, ξ_{slow} and ξ_{fast} are the measured park switches (converted into step form) and ξ'_{slow} and ξ'_{fast} are simulated. The number of data-points compared is n. It can be seen in equations (5.24) and (5.25) are not weighted. This is for two reasons. Firstly, neither the current nor the velocity of the motor is considered as more important than the other and secondly, the magnitude of the errors is similar in both cases, so neither will dominate result of equation (5.26).

The Matlab code that implements the cost function is shown in Appendix E. The code includes a method that allows the simulation model to be carried out using a variable step solver, which is significantly faster and more robust than a fixed step solver.

5.5.3 Defining GA Parameters

The values of the GA parameters used in this algorithm along with a description of their effect and a justification for their choice is given in Table 5-7.

Table 5-7: Genetic Algorithm Parameters

GA Parameter	Value	Description
Chromosome Size	17	Determines the number of genes in each chromosome/individual. In a continuous GA this is determined by the number of parameters to be identified.
Population Size	40	Determines the number of chromosomes in a population. A larger population size means that the algorithm can search in a larger space and that there is more variation in a population. However, a large population can cause convergence problems and requires more calls to the cost function which increases the running time of the algorithm. A medium population size is chosen here to keep computation time low whilst maintaining an acceptable variability in the population.
Selection Rate	0.5	Determines the percentage of the population that is retained in each generation. The algorithm then repopulates the population using the retained members and their offspring.
Elite Members	5	Elite members of a population are always kept for the next generation and are protected from mutations. The chromosomes chosen to be elite are the best members of the population in a particular generation. It is generally beneficial to use elite members; however too many elites can cause a lack of variation in the population, and the algorithm has a higher chance of converging on a local optimal solution rather than the global optimum.

Mutation Rate	0.1	Defines the number of mutations per generation as a percentage of the number of genes. A relatively high mutation rate was chosen to ensure that variability in the population was kept high in spite of a relatively small population and the use of elitism.
Maximum Iterations	200	Determines the stopping criteria of the algorithm. Unless a minimum cost function is reached, the algorithm will stop after 200 iterations.

5.5.4 *Generating the Initial Population*

An initial population of 40 chromosomes is randomly generated within the algorithm's search space. The search space it determined by the maximum and minimum values that the 17 genes (or parameters) can take. The size of the maximum and minimum values depends on the user's knowledge of the parameters and how accurately they have already been identified using previous measurements and optimization methods. Tighter boundaries means that the algorithm is more likely to converge, but also reduces the number of potential good solutions that algorithm can find. The motor parameter inequalities given in equation (3.32) are implemented by restricting certain gene values to be greater than others within their chromosome.

Once the initial population has been generated, the cost of each chromosome in the population is calculated using the cost function. The costs are then sorted and the chromosomes with the best 20 costs (i.e. half of the population) are retained and the remaining chromosomes are discarded. The retained chromosomes are now selected for mating.

5.5.5 *Selecting Mates using Tournament Selection*

Members of the remaining population are selected for mating using a tournament selection method. Three chromosomes are selected at random and the member with the lowest cost is selected as the first parent. The process is repeated with three new chromosomes and the second parent is found. These two parents will go on to produce two offspring. The whole process is repeated until 10 mating pairs are chosen.

A tournament selection algorithm was chosen because it allows stronger chromosomes to have a higher chance of being chosen to mate whilst preventing a particularly strong individual (or individuals) from dominating the mating pool. This is because the initial three contenders are chosen at random, not based on their cost, which allows weaker but potentially useful chromosomes the chance to mate.

The chosen parents are now mated to form the offspring for the new generation.

5.5.6 Carry out Mates Using Random Blending

In the mating process, information from the genes of each of the mating pair is shared to form two new offspring. This is repeated 10 times in order to produce 20 offspring, who are added to the existing twenty chromosomes to make up the next generation's population. The algorithm for a single mate is now described.

First, a random number, α, between 1 and 17 is generated, the value of which determines the number of genes that are to be combined. Following this, α unique random numbers between 1 and 17 are generated, called $\boldsymbol{\beta}$, which determines which genes are modified. Finally, a random number, γ, between 0 and 1 is generated, which is the blending factor and determines the amount by which the genes are blended [100]. The new genes for the two offspring are calculated using

$$\rho_1\left(\boldsymbol{\beta}(i)\right) = ma\left(\boldsymbol{\beta}(i)\right) - \gamma\left\{ma\left(\boldsymbol{\beta}(i)\right) - da\left(\boldsymbol{\beta}(i)\right)\right\} , \tag{5.27}$$

$$\rho_2\left(\boldsymbol{\beta}(i)\right) = da\left(\boldsymbol{\beta}(i)\right) + \gamma\left\{ma\left(\boldsymbol{\beta}(i)\right) - da\left(\boldsymbol{\beta}(i)\right)\right\} , \tag{5.28}$$

where ρ_1 and ρ_2 are the two offspring chromosomes, ma and da are the mating chromosomes and i is the iteration number which incrementally increases from 1 and α. The new chromosome ρ_1 is formed from the original genes of ma, along with the blended genes determined by α, β, γ and equation (5.27). Likewise ρ_2 is formed by da and the blended genes determined by α, β, γ and equation (5.28).

The population now undergoes random mutation to a select number of genes.

5.5.7 Mutations

To introduce a further level of randomness to the algorithm and to reduce the chance of the GA prematurely converging on a local optimal solution, a selection of genes are randomly mutated. The number of mutations depends on the Mutation Rate parameter of the GA and in this case is 0.1 (based on trial and error and in order to keep high variation in the population), i.e. a maximum of 10% of the genes in the population will mutate. Genes to mutate are selected randomly from the population and any mutations acting upon elite chromosomes are dismissed. The genes selected for mutation are assigned a new random value within their maximum and minimum values and subject to any inequalities.

The mutation rate is relatively high in this algorithm. This is to ensure that enough randomness remains in the population to explore the entire search space – this is important because the population size is relatively small. The high mutation rate is also the reason that elitism is being used.

5.5.8 Recalculate Costs

Once the new generation has been produced and mutations have taken place, the cost of each chromosome in the population is evaluated using the cost function. The costs of elite chromosomes and chromosomes from the previous generation that were not mutated are not calculated because they have not changed from the previous generation. This adds complexity to the algorithm but means that fewer calls to the cost function are made, saving on simulation time.

Once the costs of the new generation are calculated, the GA determines whether it should stop. If a chromosome in the population has a cost below a certain threshold, or if the maximum number of iterations is reached, the algorithm will terminate and output the results. If neither of the termination conditions is met, the algorithm continues to the next generation and repeats the selection, mating and mutation processes.

5.5.9 Results of Parameter Optimisation Using Genetic Algorithms

The evolution of the cost functions for each iteration is shown in Figure 5-15 which shows the best solution and the population mean. It can be seen that after around 40 generations the GA converges on a good solution. It then continues to explore the search space to refine the solution up to around generation 135, at which point the GA settles on a solution.

The chromosome with the best solution was made up of the parameters shown in Table 5-8. It can be seen that the parameters that are identified are realistic when compared to measured data and data from the literature. The values of the parameters estimated here will be used as the first estimate for the local optimiser stage shown in section 5.6 .

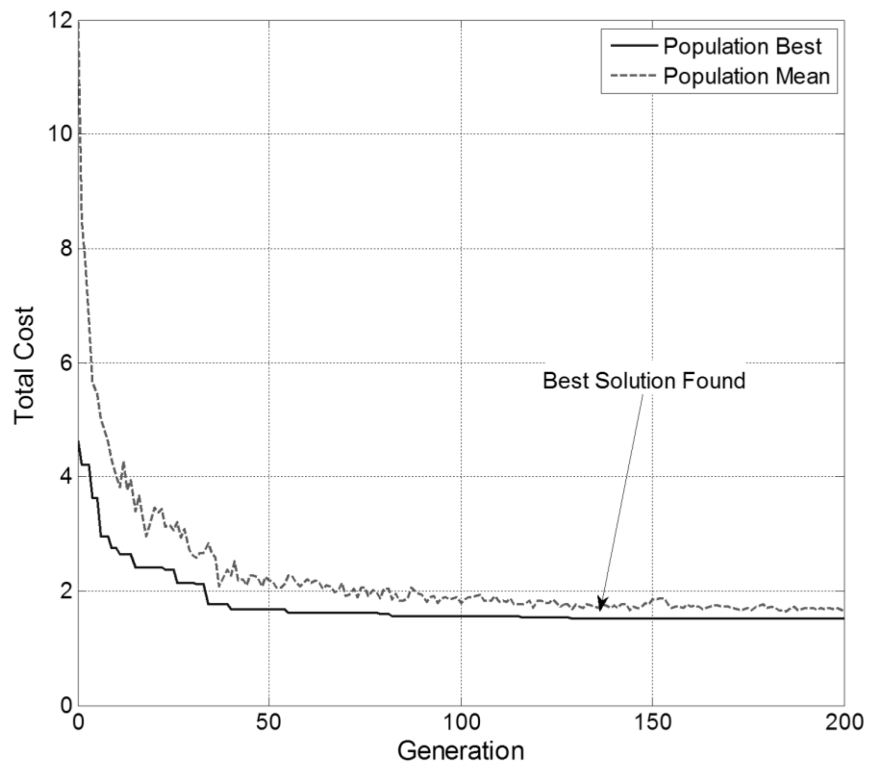


Figure 5-15: Genetic Algorithm Cost Function Plot

Table 5-8: Genetic Algorithm Best Parameters

Motor_Param	Dry_Param	Wet_Param	Force
$R_{slow} = 1.2236\Omega$	$\mu_{dry\_1} = 0.3531$	$\mu_{wet\_1} = 0.5543$	$F = 7.615N$
$R_{fast} = 1.0737\Omega$	$\mu_{dry\_2} = 0.1364$	$\mu_{wet\_2} = 6.348\times10^{-3}$	
$L_{slow} = 3.0465\times10^{-3}H$	$\mu_{dry\_3} = 2.01$	$\mu_{wet\_4} = 0.7085$	
$L_{fast} = 3.0156\times10^{-3}H$		$\mu_{wet\_6} = 2.376\times10^{-4}$	
$K_{slow} = 4.2707\times10^{-2}\dfrac{V}{(rad/s)}$		$\mu_{wet\_7} = 2.900\times10^{-3}$	
$K_{fast} = 3.0119\times10^{-2}\dfrac{V}{(rad/s)}$			
$b = 1.5899\times10^{-4}\dfrac{Nm}{(rad/s)}$			
$J = 2.2489\times10^{-5}kgm^2$			

A comparison of simulated data using the parameters in Table 5-8 and measured data, along with their corresponding residuals, are shown in Figure 5-16 and Figure 5-17 respectively.

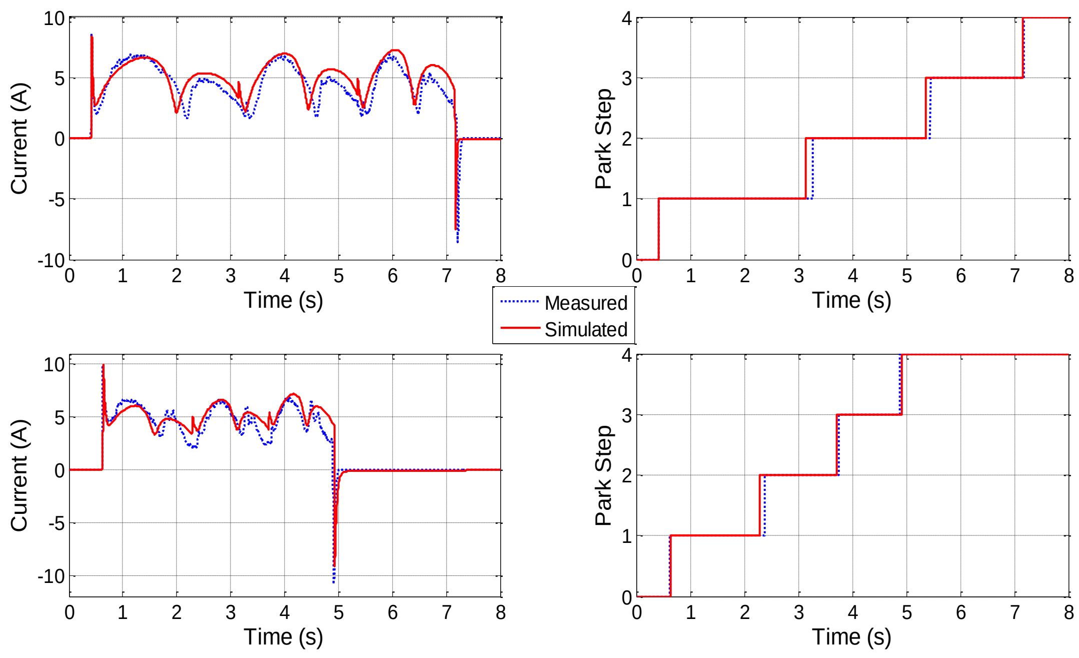


Figure 5-16: Genetic Algorithm Results Comparison

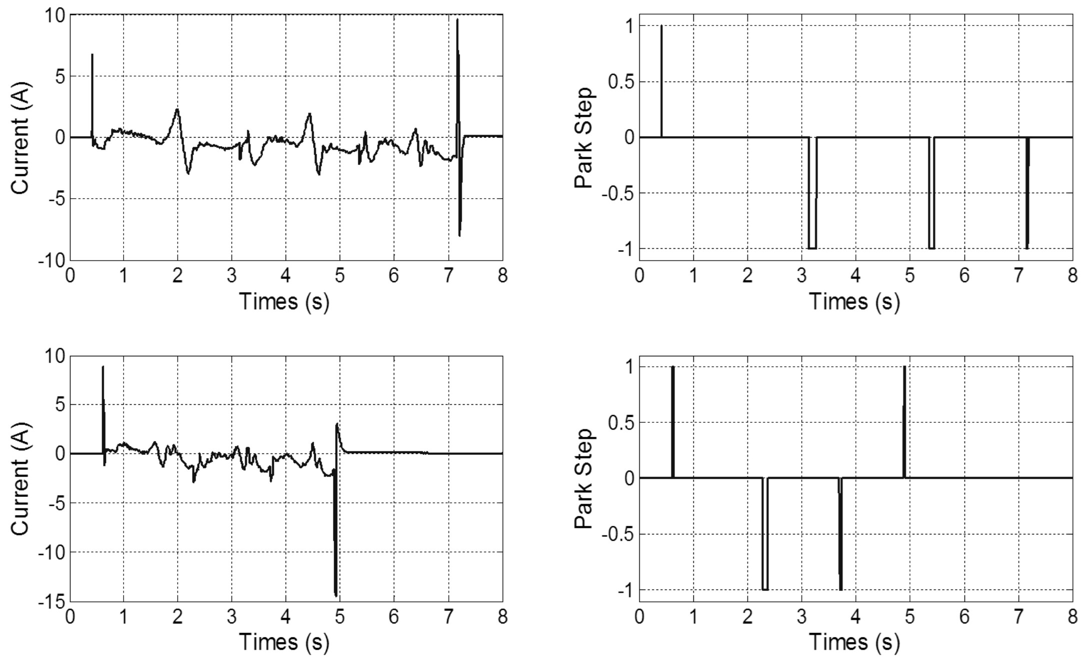


Figure 5-17: Genetic Algorithm Results Residuals

It can be seen that the simulated data follows the measured data reasonably well, with the simulated current results generally being slightly too high and the largest error occurring during the switching transients. However it should be noted that the error in shape and magnitude of the transients is very small and the large error seen is due the transients occurring at slightly different times due to the slight error in the estimated velocity of the motor.

5.6 Local Optimiser Stage

It is beneficial to employ a local optimiser at the end of a GA. This is because GAs are best suited to finding a global optimum but are less adept at refining the solution. Using the measured data taken at different input voltages to that used in the GA identification stage, the local optimiser method used in section 5.4.5 is employed to further improve the solution. The best set of parameters found using the local optimiser is given in Table 5-9. It can be seen that they are marginally different from the parameters identified by the GA given in Table 5-8. The search range of all of the parameters in the local optimiser conform to equation (5.22), with in this case U being the parameters estimated by the GA and $\hat{U}$ being the parameters estimated using the local optimizer.

A comparison of the simulated data using the parameters given in Table 5-9 and the measured data is given in Figure 5-18 along with their residuals in Figure 5-19. It can be seen that the local optimiser has marginally improved the performance of the model, particularly the error in the slow motor mode’s current.

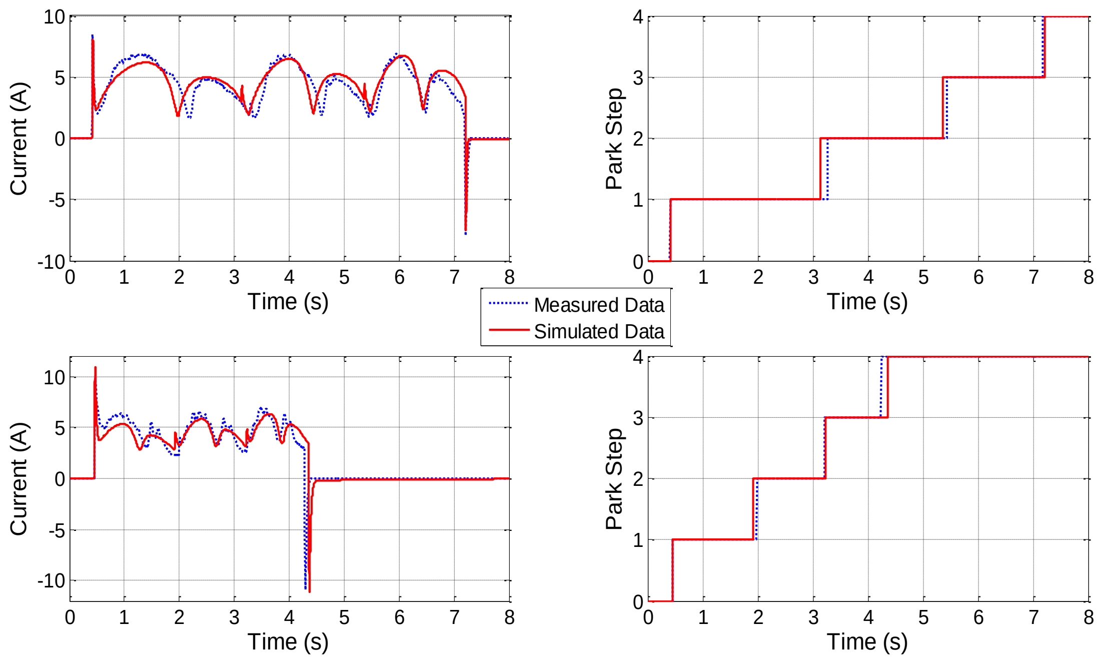


Figure 5-18: Local Optimiser Results Comparison

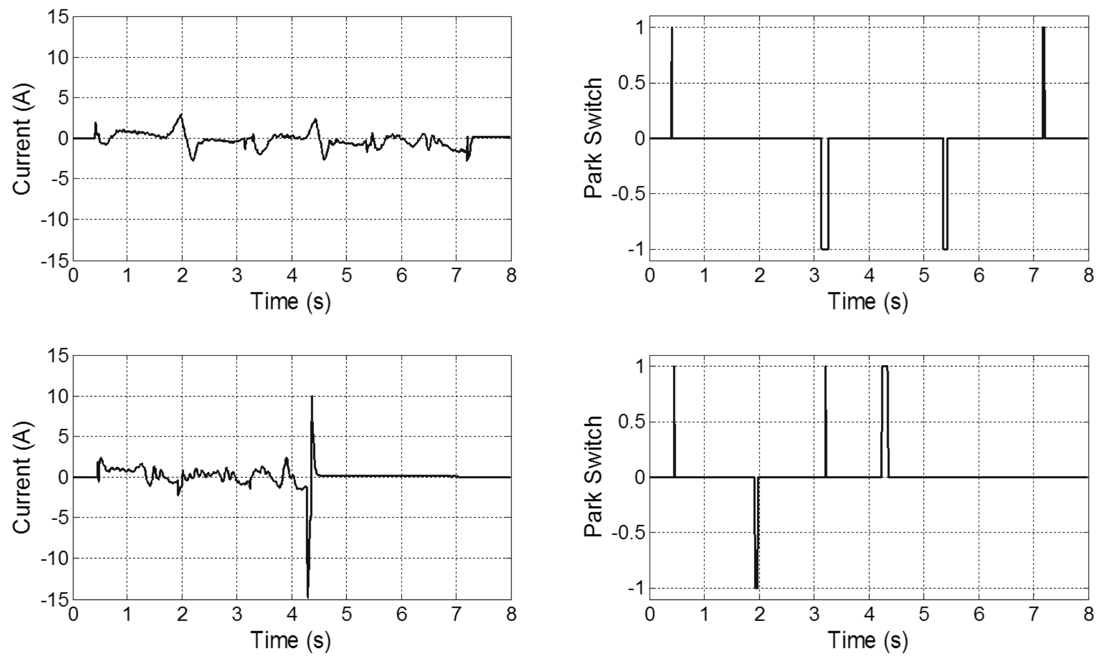


Figure 5-19: Local Optimiser Residuals

Table 5-9: Local Optimiser Best Parameters

Motor_Param	Dry_Param	Wet_Param	Force
$R_{slow} = 1.237\Omega$	$\mu_{dry\_1} = 0.3186$	$\mu_{wet\_1} = 0.4454$	$F = 7.615N$
$R_{fast} = 1.026\Omega$	$\mu_{dry\_2} = 0.1238$	$\mu_{wet\_2} = 7.578\times10^{-3}$	
$L_{slow} = 3.148\times10^{-3}H$	$\mu_{dry\_3} = 1.996$	$\mu_{wet\_4} = 0.6756$	
$L_{fast} = 2.571\times10^{-3}H$		$\mu_{wet\_6} = 2.795\times10^{-4}$	
$K_{slow} = 4.579\times10^{-2}\dfrac{V}{(rad/s)}$		$\mu_{wet\_7} = 3.456\times10^{-3}$	
$K_{fast} = 3.412\times10^{-2}\dfrac{V}{(rad/s)}$			
$b = 1.582\times10^{-4}\dfrac{Nm}{(rad/s)}$			
$J = 1.983\times10^{-5}kgm^2$			

5.7 Discussion

In this chapter, the unknown parameters of the wiper motor and mechanical forces have been identified using a three stage optimisation method. Seventeen parameters were identified, i.e. the eight motor parameters, three dry friction parameters, five wet friction parameter and the force applied by the arms to the blades. The data used to identify the parameters was measured from a real wiper system and consisted of the voltage applied to the motor, the motor current and the motor park switch. The identification was achieved with no continuous position/speed or load torque data.

The first step in identifying the parameters was to perform an initial prediction of the motor parameters by deriving a transfer function from the motor's dynamic equations which related the output current to the input voltage. The coefficients of the transfer function could then be identified used the System Identification Tool in Matlab, and from these coefficients the motor parameters could be derived. The identified parameters were compared against measured parameters and were shown to be correct to with an order of magnitude, apart from the inductances which were significantly overestimated. The parameters identified in this stage were used to determine the search space of the Genetic Algorithm, which is the next stage of parameter identification.

A Genetic Algorithm has been designed that attempts to minimise the calls to the cost function and the simulation time of the model, whilst maintaining a suitable variability in the population. The algorithm attempted to minimise a two objective cost function, i.e. the difference in the measured and simulated currents and the difference between the measured and simulated park switches. The algorithm successfully identified all seventeen parameters after 135 generations. The final simulated data matched the measured data reasonably well, with a tendency to overestimate the current and speed of the motor. The parameters identified by the Genetic Algorithm were used as an initial guess for a non-linear least squares local optimisation algorithm, which is the final stage of the identification process.

A non-linear least squares local optimisation algorithm built into the Simulink environment was used to hone the solution identified by the GA. Different data was used to identify the parameters from that used in the Genetic Algorithm. The optimiser marginally improved the overall accuracy of the model by improving the steady state error in the current.

Considering the highly dynamic and nonlinear nature of the mechanical load applied to the motor, and the limited identification data available, the final estimated parameters caused the model to perform well against measured data across its entire range of operation. The models can now be simplified in order to make them suitable for real-time simulation and thus HIL simulation; this work in shown in Chapter 6.

Chapter 6 - Model Simplification for HIL Implementation

6.1 Introduction

In the previous three chapters a physical model capable of accurately simulating the behaviour of the windshield wiper system has been developed and its equivalent parameters identified. However, in order to solve the model in real-time, a variable step solver is required. This is unsuitable for HIL simulation because HIL simulation requires a fixed step solver. In order to solve the physical model using a fixed step solver, a time-step in the order of 0.0001s is required to ensure that the simulation does not violate constraints and cause simulation errors. However, the system cannot be solved fast enough with a time-step of 0.0001s to be used in real-time; and in this case the smallest time-step available when using the HIL simulator is 0.001s. Therefore the model must be adapted to make it more robust and reduce its simulation time.

There are four strategies for adjusting a desktop simulation model for real-time simulation [101], these are:

1. Choose a solver more suited to real-time simulation.
2. Reduce the number of solver iterations.
3. Reduce the step size of the solver.
4. Decrease the size and fidelity of the model.

Regarding point 1, generally explicit solvers are preferred to implicit solvers for real-time simulation. However physical systems tend to be stiff, meaning that they possess dynamics with significantly different time constants, and implicit solvers are better suited to solving such systems. In its current form the model developed in this project does require an implicit solver[10]. It can be seen in Figure 6-1 (adapted from [101]) that the best solvers for a physical system developed in Simscape are the Backward Euler and Trapezoidal Rule implicit solvers. These work by locally solving the physical network whilst another solver solves the non-physical elements of the model. However, the local Simscape solvers cannot be used if a SimMechanics model is connected to the physical network. Hence it is beneficial to remove the SimMechanics element so that the local solvers can be used.

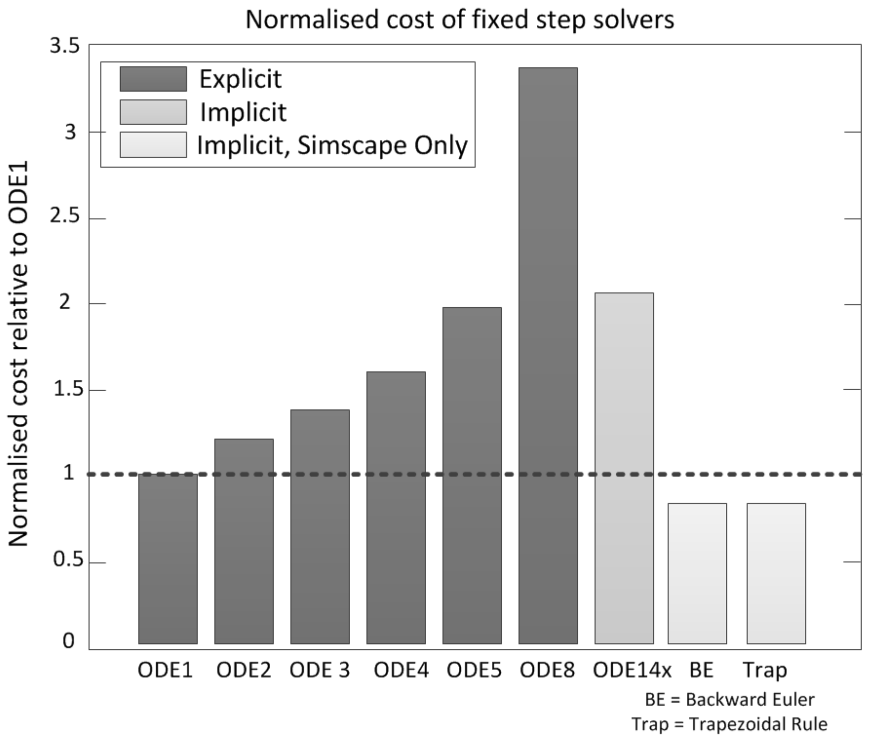


Figure 6-1: Simulink Fixed Step Solver Comparison

Regarding point 2, the number of iterations of an implicit solver has a relatively small effect on the simulation time and in this case is set at three to trade-off between accuracy and time. Regarding point 3, the step size is fixed at 0.001s due to the type of HIL system being used, so this cannot be modified.

[10] Currently the fixed step solver with the best performance in Simulink is ODE14x.

Addressing point 4 is the main focus of the rest of this chapter. It has been stated that significant gains in terms of simulation time and robustness can be achieved by using the Simscape local solvers, which requires the removal of the SimMechanics element of the model. This is achieved using look-up tables, polynomials and Artificial Neural Networks (ANN) in order to model the kinematic and force/torque elements of the system. ANNs have also been used to model the wiper motor to further increase the speed of simulation.

6.2 Kinematic Approximation Using Look-up Tables

It was shown in Chapter 4 that the mechanical element of the wiper system is approximated as a fully kinematically defined system of rigid bodies with one DOF. This means that there is a direct relationship between the position, velocity and acceleration of any arbitrary point on the system and the system's DOF (i.e. the motor output shaft). An algorithm has been developed that extracts the positional data of the linkage rockers and the wiper blades with respect to the position of the motor's output shaft (any desired points could be chosen). This information is then implemented in look-up tables.

The position of the motor shaft generated from the motor model developed in Chapter 3 is processed so that it rises from 0 to 2π radians during a single wipe and then resets to 0, as is required by the look-up tables. The outputs of the look-up tables can then be differentiated with respect to time to determine the velocity and acceleration of the desired points. Using this method the SimMechanics model is not needed to generate kinematic information.

6.3 No-Load Torque Polynomial Model

The so called no-load model is commonly used when implementing a HIL scheme that incorporates a wiper system. The no-load model refers to a wiper system consisting of just the motor and linkages and is clearly much simpler to implement than the full system in both simulation and hardware. Despite its relative simplicity, it is still beneficial to replace the SimMechanics element of the models with a simpler system.

Because the wiper blades are not included and the damping in the linkages is approximated as being lumped in with the motor[11], only the inertia of the linkages contributes to the external torque load applied to the motor. The fundamental shape of the torque is periodic and can be modelled using two polynomials, one for a forward wipe and one for a reverse wipe. An algorithm has been written to extract the fundamental torque shape from the SimMechanics model and uses the Mathworks line fitting tool to identify the polynomials. An example of the polynomials identified for the slave driven system described in Chapter 4 are given as

$$f(x) = -0.19224x^4 + 0.021769x^3 - 0.15435x^2 + 0.077812x + 0.68626 \, , \tag{6.1}$$

$$\begin{aligned} g(x) = 0.4184x^7 - 0.30315x^6 - 0.42441x^5 + 0.12634x^4 + 0.21484x^3 \\ -0.10093x^2 - 0.29859x + 0.63276 \end{aligned} \, , \tag{6.2}$$

where x is the sine of the position of the motor shaft, $f(x)$ describes the fundamental shape of the forward wipe and $g(x)$ describes the reverse wipe's shape. The torque is dependent on the position and speed of the motor output shaft with the speed determining the offset of the torque and its peak to peak value. The equation used to model the torque is

$$\tau = \begin{cases} (vC_2)\times\left(f(x)+(vC_1)\right) & \text{if } \sin\theta \text{ is } +\text{ve} \\ (vC_2)\times\left(g(x)+(vC_1)\right) & \text{if } \sin\theta \text{ is } -\text{ve} \end{cases} \, , \tag{6.3}$$

[11] A small amount of damping was included in the linkage's joints to stabilise the simulation.

where v is the velocity of the motor output shaft, θ is the position of the shaft and C_1 and C_2 are constants to be identified.

The constants C_1 and C_2 are identified using the GA described in Chapter 5, which has been modified to identify 2 parameters rather than 17. Five datasets at five different speeds were used to construct the cost function. The cost function for each individual dataset, σ_j, where j is an integer between 1 and 5 identifying the dataset, is

$$\sigma_j = \sum_{i=1}^{N} \left| \tau(i) - \tau'(i) \right| , \tag{6.4}$$

where τ is the target torque, τ' is the simulated torque and i determines data point between 1 and N, with $N = 1001$. The final cost function is the sum of the five cost functions given in (6.4) and is given as

$$\text{cost} = \frac{1}{N} \sum_{j=1}^{5} \sigma_i \ . \tag{6.5}$$

The final values of C_1 and C_2 are identified as 0.0035 and 0.2738 respectively. A comparison of results from the SimMechanics model and the polynomial model is given in Figure 6-2. The no load model with a SimMechanics element was measured as taking 86s (real-time) to simulate 60s (simulation time) whereas the polynomial model took 4s to simulate 60s, demonstrating a significant increase in simulation speed for a negligible loss of accuracy. The evolution of the fitness function is given in Figure 6-3.

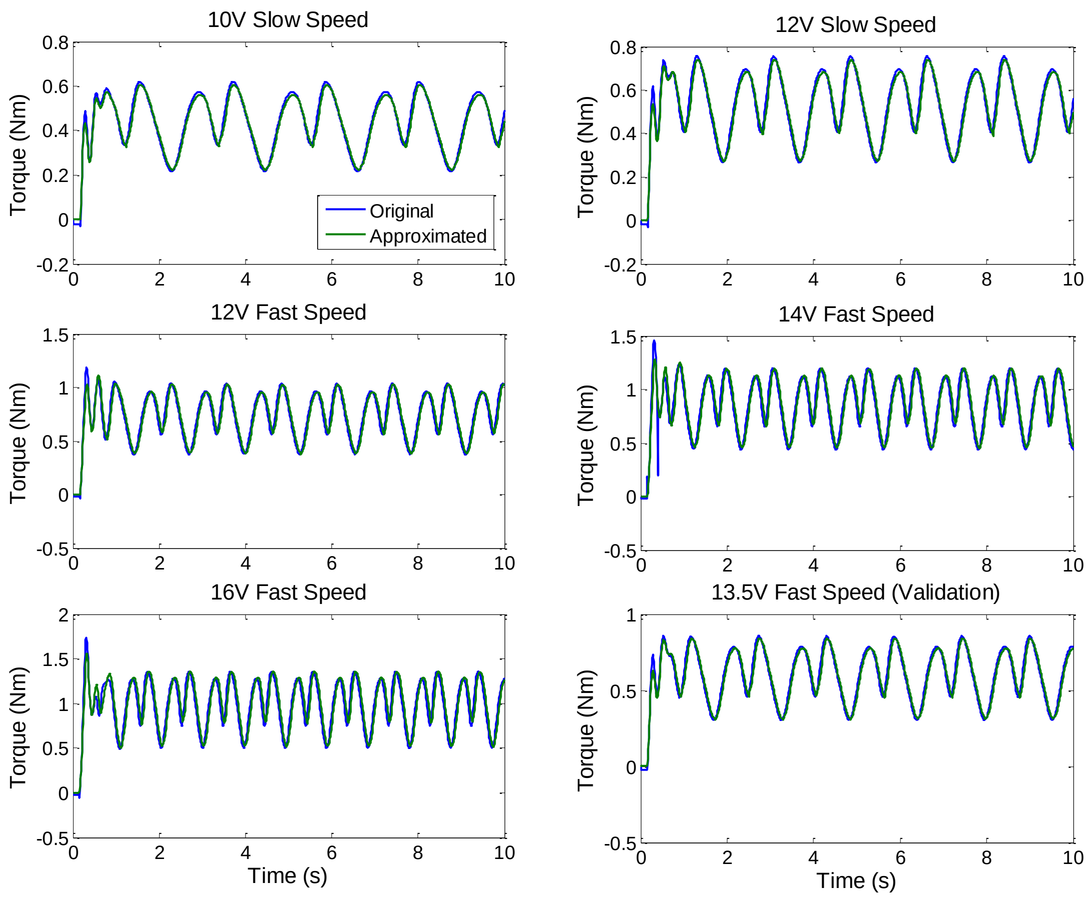


Figure 6-2: No-Load Torque Approximation Performance

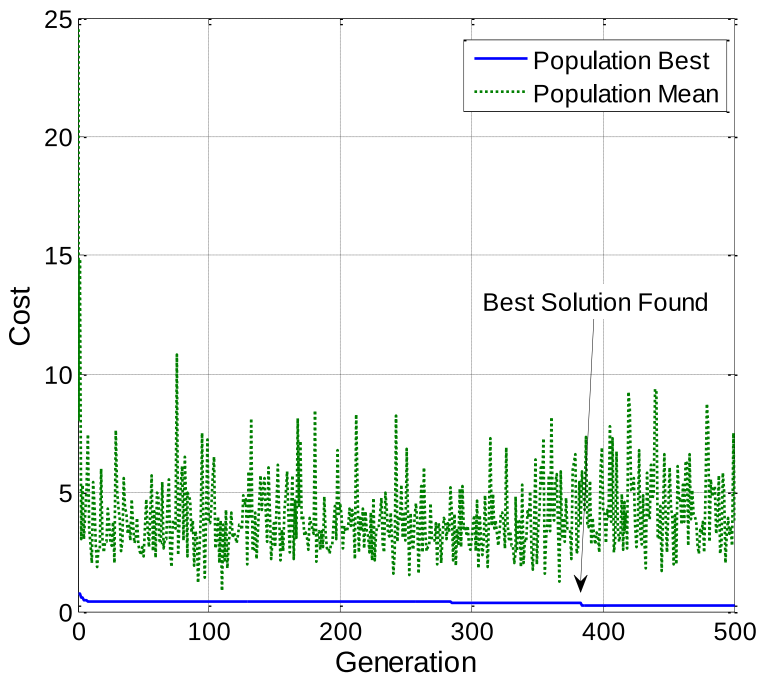


Figure 6-3: No-Load Torque Approximation GA Performance

6.4 Neural Network Modelling

It has been shown in the previous section that polynomials are an effective way of modelling the torque load on the wiper motor caused by the so called no-load model. However, when the torque load is produced by full wiper system, i.e. the linkages, arms, blades and windscreen, more advanced modelling tools are required. Artificial Neural Networks (ANN)[12] can be trained to represent complex systems and will now be used to simplify the wiper model.

6.4.1 Neural Network Theory

The information given here is developed from [102] and [103] unless otherwise stated. Neural Networks map an input space to an output space via a series of interconnected layers of neurons that act as transfer functions. The interconnections between layers are generally weighted and biased, and it is the numerical values of theses weights and biases that are adjusted using a learning algorithm to optimise the performance of the network. The general form of a three layer feed forward NN is shown in Figure 6-4. The network is arbitrarily chosen to have two inputs, one output and three neurons in the hidden layer.

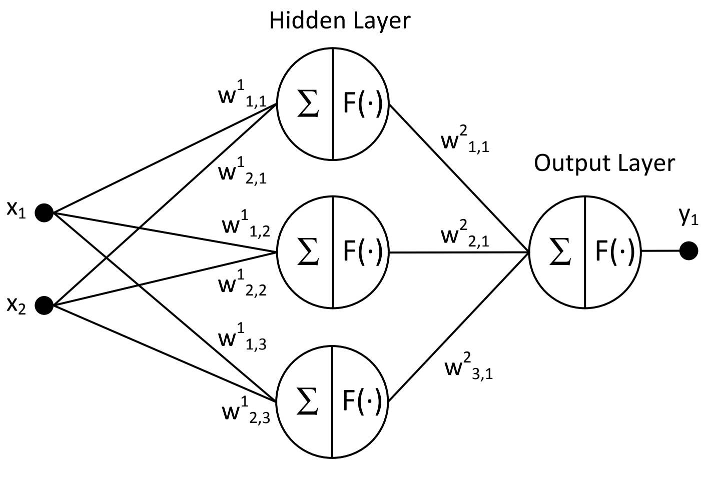


Figure 6-4: General Feed Forward Neural Network Structure

[12] Henceforth referred to as Neural Networks (NN)

The input layer generally consists of a pre-processing rule, usually to normalize the inputs, which improves the efficiency of the algorithm. Transfer functions can also be applied in the input layer. Each input in the input layer is connected to each of the neurons in the hidden layer via weighted connectors. The weighted inputs of a single neuron, sometimes along with a bias (not shown in Figure 6-4), are summed as shown

$$c = \sum_{i=1}^{n} \left(w_{i,\mathrm{j}}^{L} x_i \right) + w_0 \ , \tag{6.6}$$

where i, j, and L are the layer input number, layer neuron number and layer number, respectively, w is the weight, x is the input and w_0 is the bias and n is the number of inputs to the layer. The output of (6.6) is fed into a transfer function which can take numerous forms. Generally, hidden layer transfer functions are sigmoidal because they are differentiable, which is essential for many optimization algorithms used to train neural networks[13]. The transfer function used in the hidden layers in this project is the Tan-Sigmoid transfer function, implemented as [104]

$$a = \frac{2}{1+e^{-2c}} - 1 \ , \tag{6.7}$$

where c is the output of (6.6). The transfer functions in the output layer neurons tend to be purely linear and act on the outputs before the reverse normalisation process is applied to them.

Once the network has been designed it must be trained using input and target data. This project uses supervised training methods, a block diagram of which is shown in Figure 6-5. Generally, the performance of a network is evaluated by calculating the Mean Squared

[13] Note that it is not always necessary for the transfer function to be differentiable. For example a NN optimized using a GA would not require differentiable transfer functions.

Error (MSE) between the output of the network and the target outputs provided by the supervisor. The MSE is calculated as

$$MSE = \frac{1}{N}\sum_{k=1}^{N}\left(y_k - \hat{y}_k\right)^2 \tag{6.8}$$

where N is the number of data points, y_k is a target data point and $\hat{y}_k$ is a network output data point at interval k. The input and output data is generally split into three datasets: Train, Validate and Test. The Train dataset is used to directly train the network to fit the outputs and is the data used in (6.8), the Validation dataset is used to measure the generalization of the network (i.e. how well it extrapolates) and the Test dataset gives a measure of performance using data that has had no effect on the training of the network.

The training algorithm used in this project is the Levenberg-Marquardt backpropagation algorithm, which is generally considered the fastest backpropagation algorithm for networks of the size used here (in the order of a few hundred weight elements).

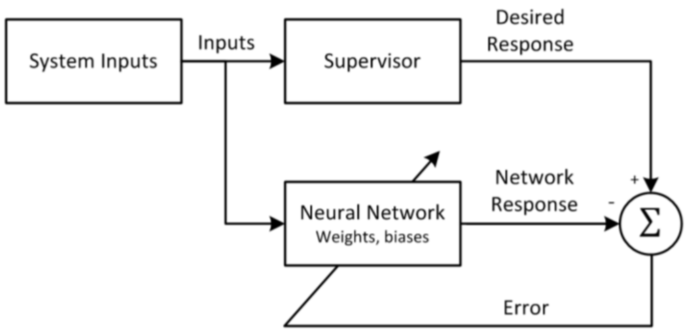


Figure 6-5: Neural Network Supervised Training Block Diagram

Another type of NN is a Nonlinear Autoregressive Network with Exogenous Inputs (NARX), which is commonly used in time series modelling. The defining equation of a NARX model is

$$\mathbf{y}(\mathrm{t}) = f\left(\mathbf{y}(t-1), \mathbf{y}(t-2), \ldots, \mathbf{y}\left(t-n_{Dy}\right), \mathbf{x}(t-1), \mathbf{x}(t-2), \ldots, \mathbf{x}\left(t-n_{Dx}\right)\right), \quad (6.9)$$

where $\mathbf{y}(\mathrm{t})$ is the output vector, $\mathbf{x}(t)$ is the input vector and n_{Dy} and n_{Dx} are the number of output and input time delays respectively. It can be seen from (6.9) that the output of a NARX network depend on previous values of the output and input. A simple diagram of a NARX network is given in Figure 6-6 when D^n indicates a delay of n time-steps.

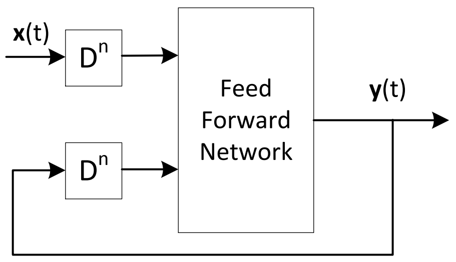


Figure 6-6: NARX Model Diagram

6.4.2 Mechanical Load Modelling

The torque applied to the wiper motor is now modelled using a feed forward NN such as that shown in Figure 6-4. Use is made of the fact that there is a direct relationship between the torque applied to the motor and the position and velocity of the motor output shaft which can be mapped by the NN. The input and output data used to train the network is shown in Appendix F and consists of four wipes at different constant velocities. The data is randomly split into 70% training data, 15% validation data and 15% test data. A second set of data, also shown in Appendix F, shows data used purely for validation and was not used to train the NN.

A study to indicate the best number of hidden layers in the network has been carried out, the results of which are presented in Table 6-1 and graphically Figure 6-7. The network was trained three times for each layer number and the average of the three results are presented here (full results are presented in Appendix F). For each layer number, the MSE of the train, validate and test data, along with the total MSE, training time and simulation time (Simulating 5 seconds of data) and the total MSE of the separate validation data are recorded and compared.

Table 6-1: Neural Network Performance against Number of Hidden Layers

Hidden Layers	Training MSE				Validation MSE	Training Time (s)	Simulation Time (s)
	Train	Validate	Test	Total	Total		
1	21.523	21.845	22.704	21.749	20.098	0.933	0.129
2	13.979	13.900	13.724	13.775	10.885	0.933	0.081
3	4.520	4.010	4.391	4.424	2.498	4.667	0.091
4	2.781	2.716	3.052	2.812	1.731	14.000	0.081
5	4.958	5.150	5.110	5.010	6.322	18.333	0.089
6	0.663	0.672	0.738	0.675	0.492	34.333	0.092
7	1.487	1.414	1.616	1.496	2.751	9.000	0.093
8	1.104	1.001	1.102	1.088	1.948	15.667	0.087
9	0.218	0.259	0.216	0.224	0.847	26.333	0.088
10	0.217	0.188	0.192	0.209	0.407	11.667	0.110
11	0.208	0.190	0.235	0.209	25.430	28.000	0.104
12	0.329	0.364	0.386	0.343	131.727	15.000	0.102
13	0.368	0.342	0.380	0.360	359.821	21.333	0.095
14	0.125	0.130	0.122	0.125	8.147	35.333	0.107
15	0.125	0.103	0.114	0.120	662.871	47.667	0.097
16	0.056	0.054	9.321	1.445	7345.731	42.333	0.098
17	0.126	0.161	0.144	0.134	269.474	31.000	0.111
18	0.059	0.057	2.136	0.370	589.505	40.333	0.103
19	0.029	0.030	0.034	0.030	357.209	40.333	0.120
20	0.037	0.046	0.091	0.047	176.187	34.667	0.122

It can be seen in Figure 6-7 that an increase in the number of layers tends to decrease the total MSE of the output of network when analysing the training data. Table 6-1 shows that the MSE in the validate and test datasets of the training data also tend decrease with the number of layers. However, Figure 6-7 also shows that the total MSE of the separate validation data only decreases with an increase in the number of layers of up to around 10, after this point the error increases significantly[14]. This is because the NN is being over-trained and simply memorizes the training data, i.e. the network does not generalize well. Based on the results shown in Table 6-1 and Figure 6-7, 10 hidden layers were chosen for the network design. A schematic of the network is shown in Figure 6-8. Note that the connections from the input layer to the weight blocks in the first hidden layer have two components. The network simultaneously outputs the torque for the wet and dry windscreen conditions.

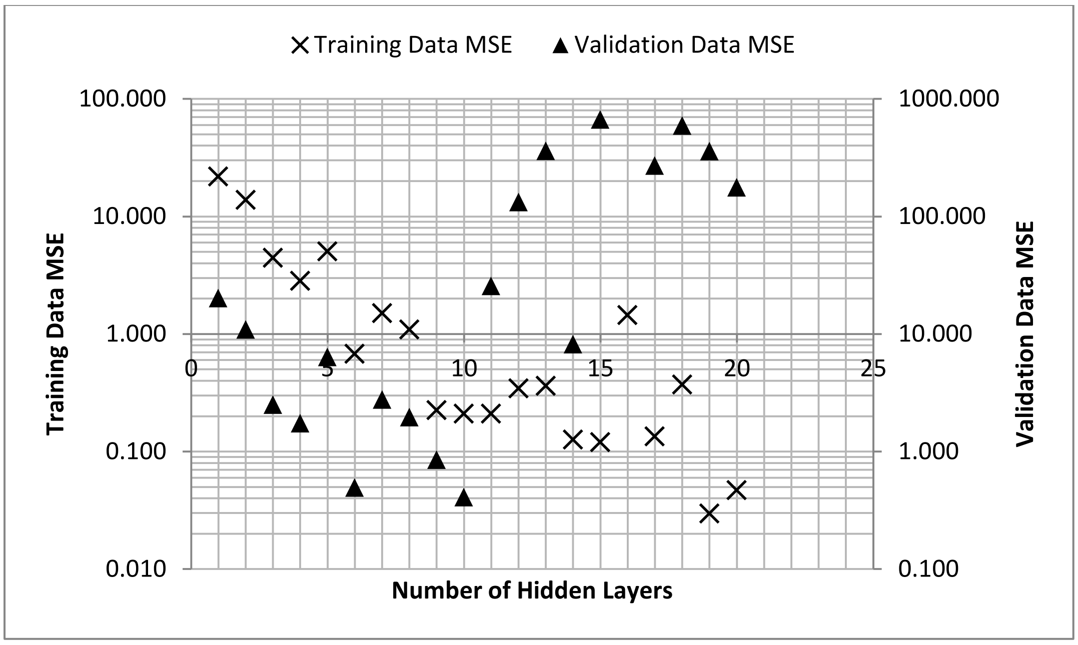


Figure 6-7: Neural Network Performance against Number of Hidden Layers

Plots comparing the trained NN output with target data, along with their errors, are shown in Figure 6-9 and Figure 6-10 for the dry and wet condition torques for the training data and Figure 6-11 and Figure 6-12 for the dry and wet torques for the separate validation data respectively. It can be seen that the NN accurately models the torque load for both training and validation data.

[14] Note that the result for the total MSE of the separate validation data for 16 hidden layers has been omitted from the plot for clarity because it was significantly larger than the other results.

The MSEs, training time and simulation time of the network shown in Figure 6-8 are shown in Table 6-2. It can be seen that the MSE for the training and validation data is very small and the training time is low. The simulation time result refers to the amount of time taken to simulate 5 seconds of operation in a Simulink implementation of the NN. It can be seen that the simulation is significantly faster than real-time.

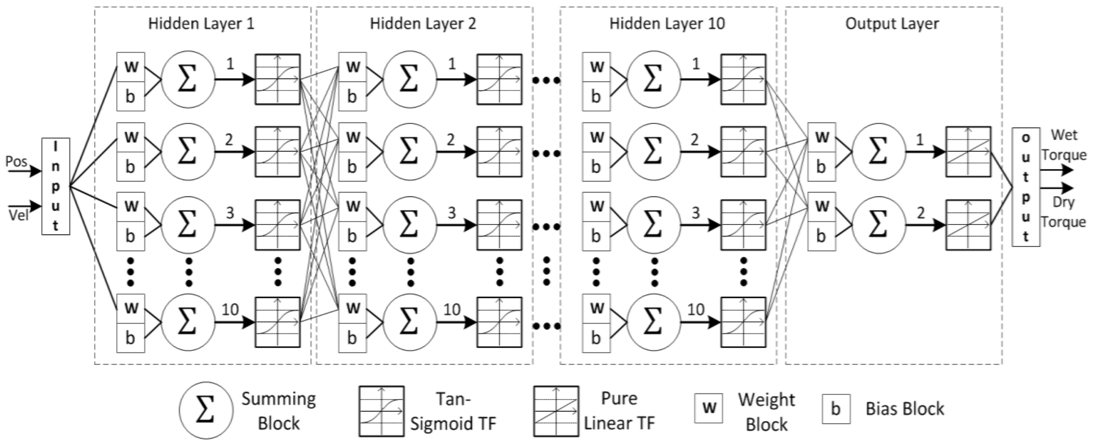


Figure 6-8: Mechanical System Feed Forward NN Approximation Schematic

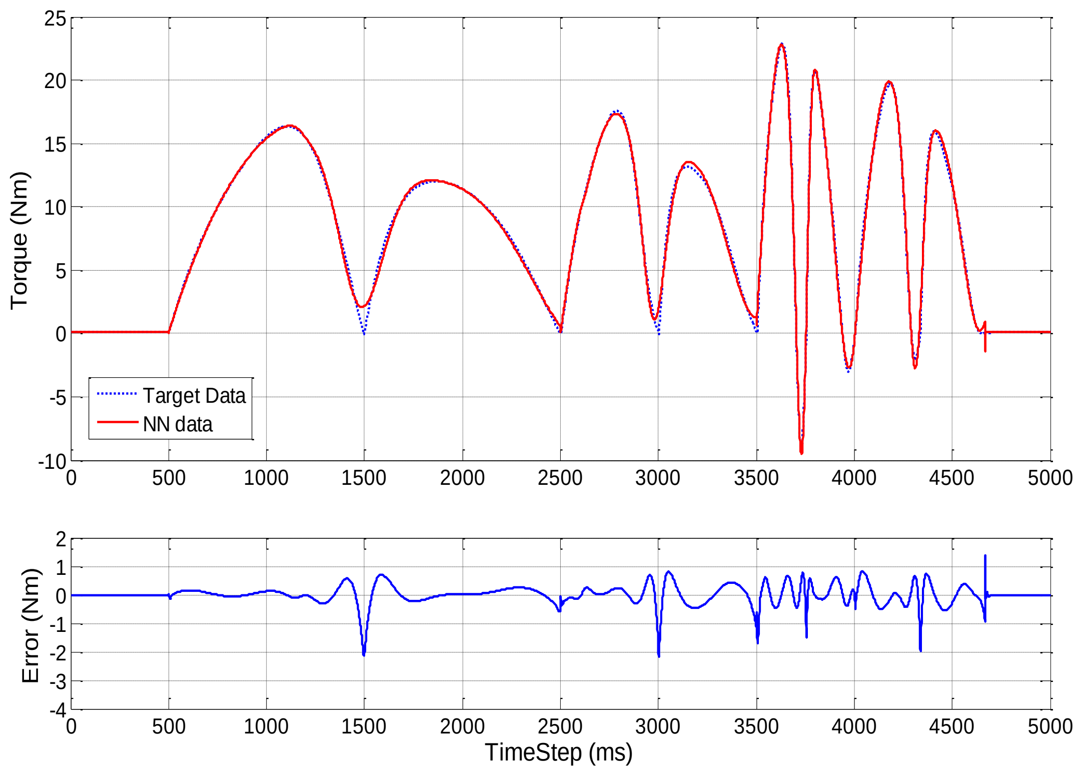


Figure 6-9: Mechanical System NN Dry Torque Performance (Training)

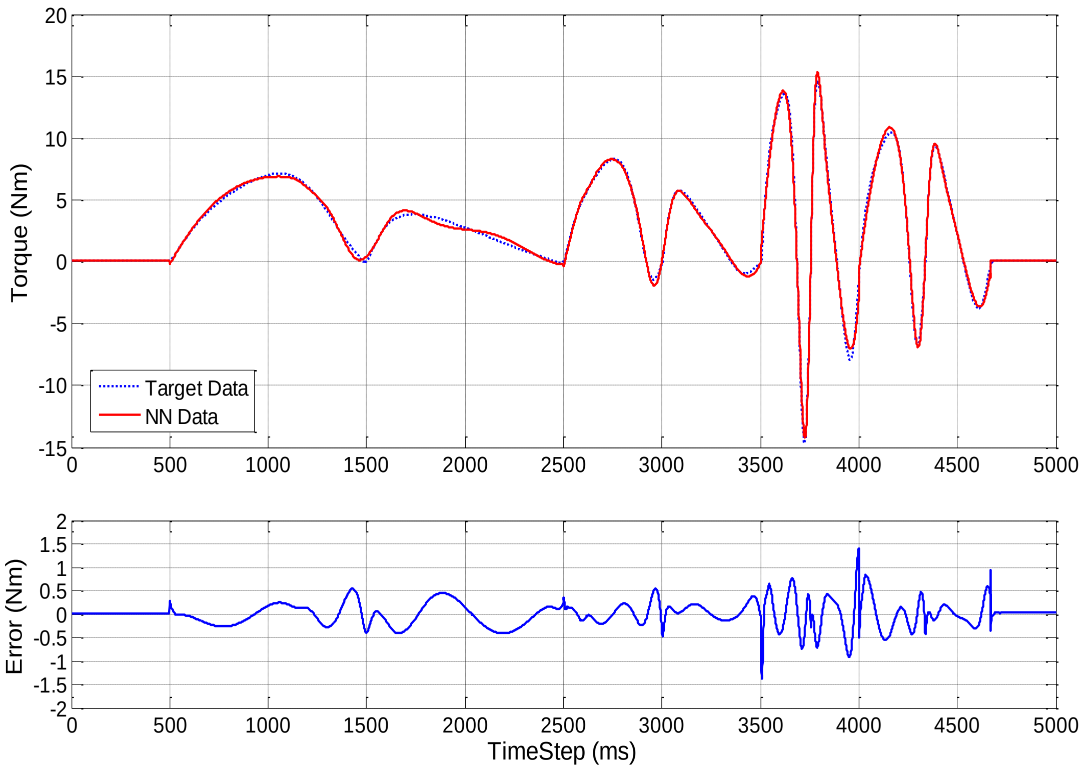


Figure 6-10: Mechanical System NN Wet Torque Performance (Training)

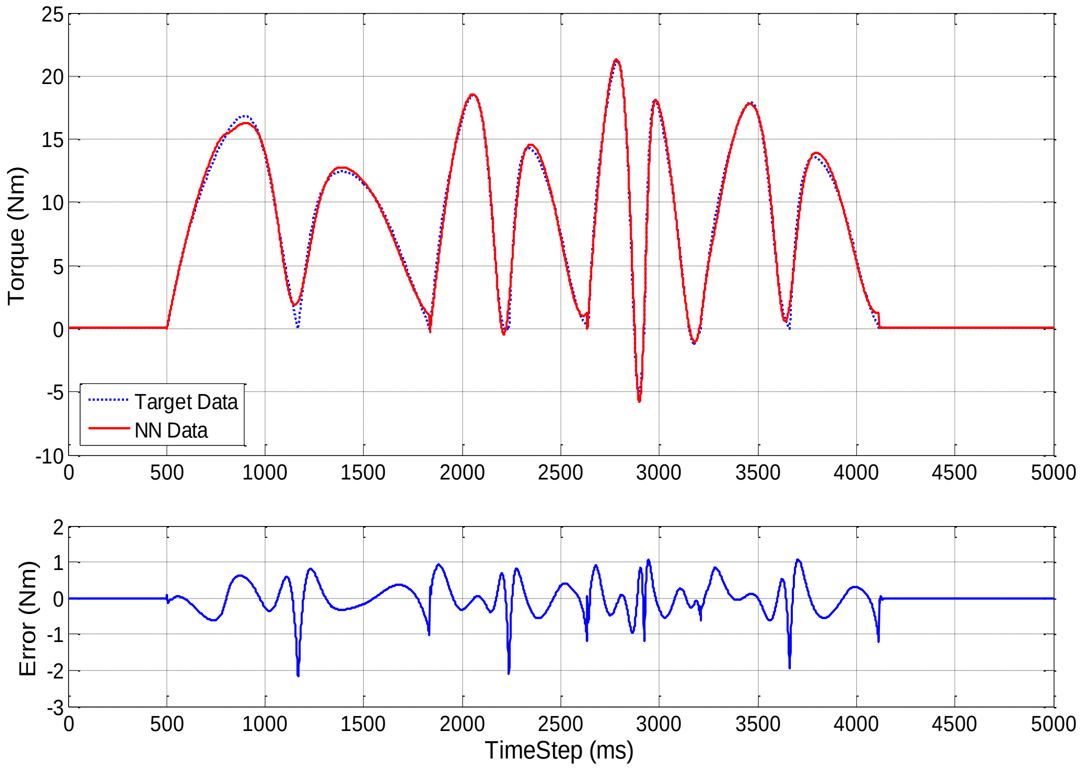


Figure 6-11: Mechanical System NN Dry Torque Performance (Validation)

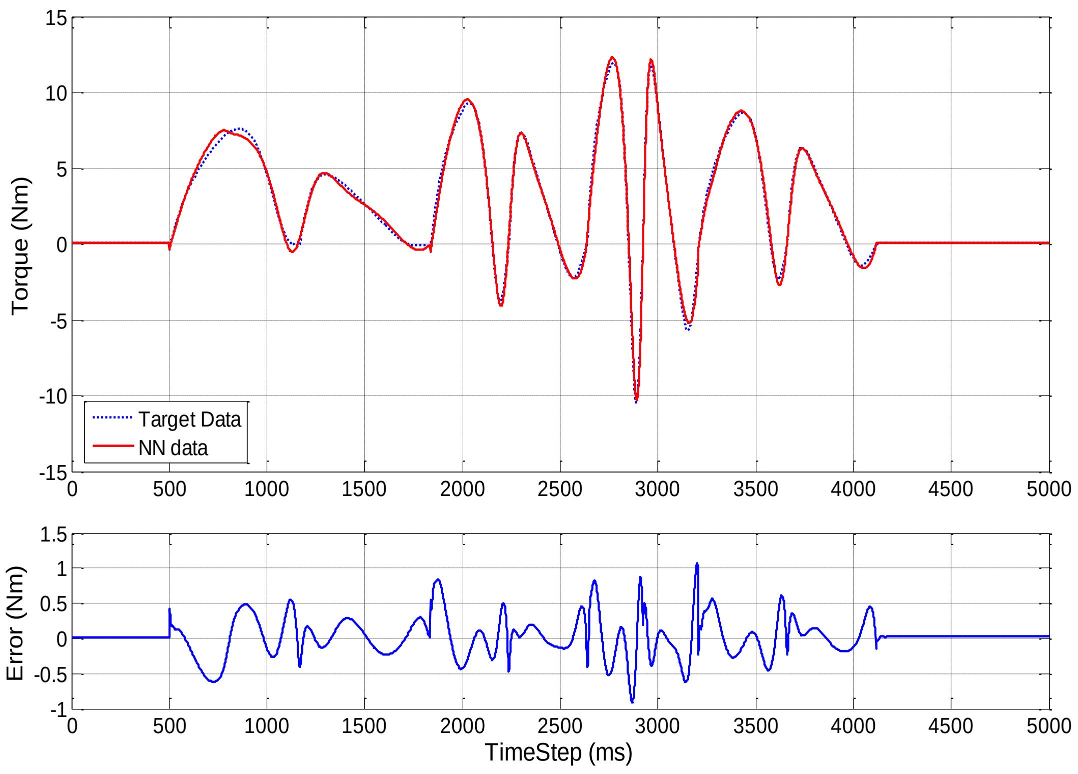


Figure 6-12: Mechanical System NN Wet Torque Performance (Validation)

Table 6-2: Mechanical System NN Approximation Performance

Training MSE				**Validation MSE**	**Training Time (s)**	**Simulation Time (s)**
Train	**Validate**	**Test**	**Total**	**Total**		
0.0967	0.085	0.0977	0.0951	0.1102	45	0.103612

6.2.1 Simulink Implementation

An implementation of the full wiper model with the NN shown in Figure 6-8 replacing the SimMechanics element of the model is shown in Figure 6-13. The system is controlled by a Simscape control system switching between different input voltages and driving a relay driver circuit connected to the motor. The motor position is conditioned to reset to zero when it reaches 2π radians, as is required by the NN. The output of the NN is routed through a switch which selects either the wet or dry condition and is multiplied by -1 before being connected to the motor. The model has been measured to simulate 60s (simulation time) of data in 5.92s (real time), i.e. the model is suitable for real-time simulation.

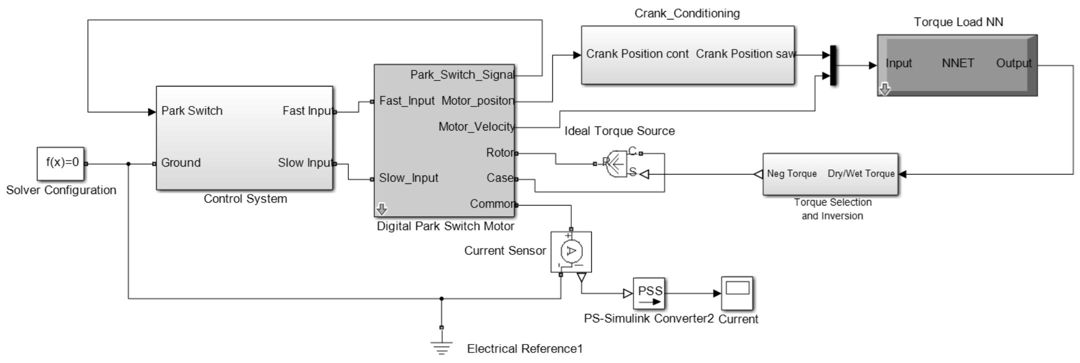


Figure 6-13: Hybrid Physical and Neural Network Model

6.4.3 Whole Wiper System Modelling

It is now investigated whether the whole wiper system can be modelled using NNs. To do this, a NN must be designed that can model the dynamics of the wiper motor and interface with the torque load NN. To do this, a custom NN is designed which is based on the principle of a NARX network. The input to the motor NN is formed of the voltage from the battery and the torque from the mechanical element of the model. The outputs are the motor current and the velocity of the motor output shaft. Both of the outputs are fed back to the NN via a delay block. The structure of the motor NN is shown in Figure 6-14. The Matlab code used to build this NN is shown in Appendix F.

Figure 6-14 shows that the network is essentially formed of two networks with a common input[15] operating in parallel with each other, both outputting the motor current and velocity. One network models the motor dynamics in its slow mode and the other models it in its fast mode. Each parallel network has 8 hidden layers with 8 neurons each. The inputs are routed through a delay of two time-steps, as are the feedback signals. Hence

[15] For training purposes the inputs for the Slow and Fast networks are different. This is to allow the networks to be trained at the same time. In normal operation the inputs are the same and user selects which dynamics are required at the output, depending on the state of the motor.

for the first two time-steps of the simulation the inputs will always be zero. The input data used to train the motor network is shown in Appendix F.

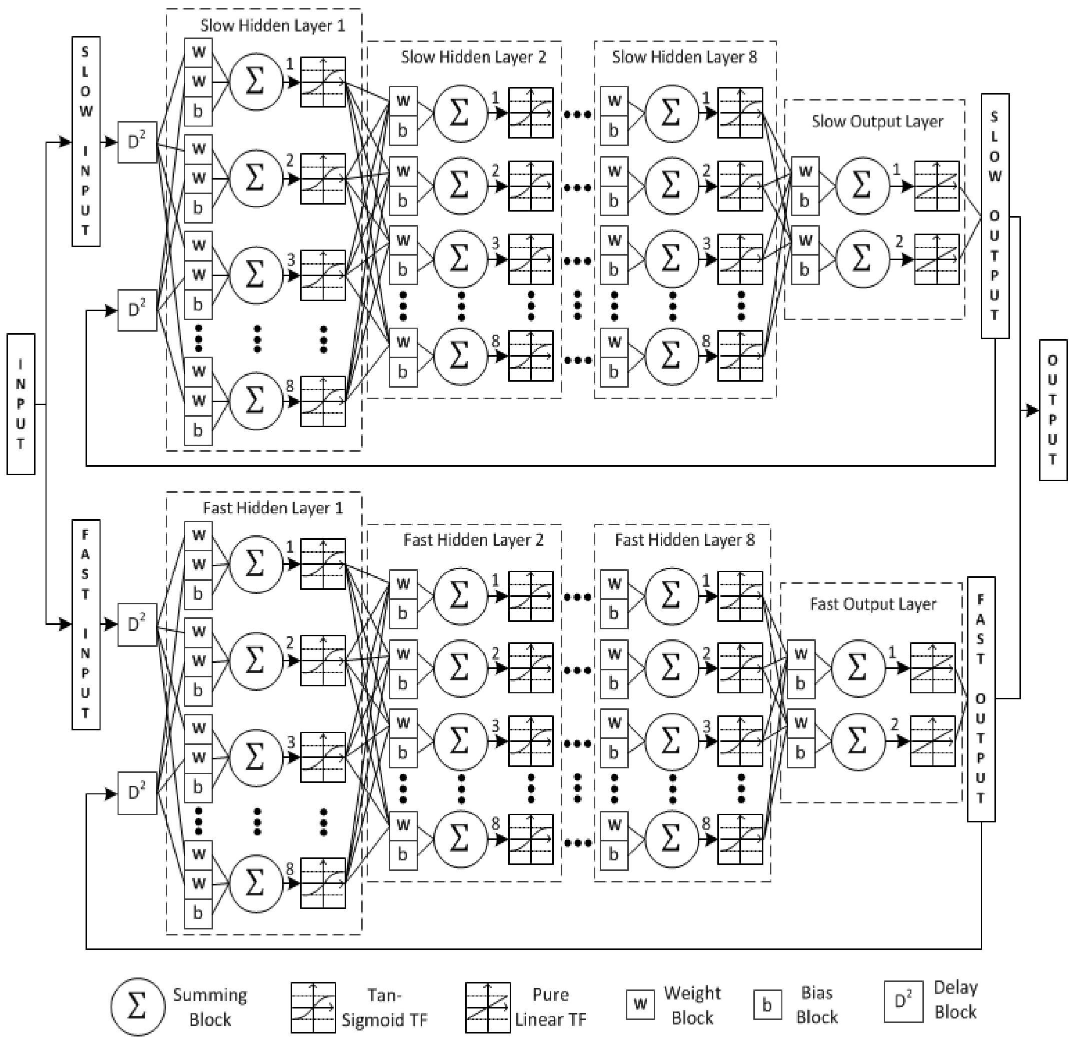


Figure 6-14: Motor Neural Network NARX Approximation Schematic

As with the feed forward network, the training data is separated into three different datasets: Train, Validate and Test in percentages of 70%, 15% and 15% respectively. However, in this case the data is not split up randomly; rather the first 70% of the data is used for training, the next 15% for validation and the final 15% for testing. This is to ensure that the time dependent relationships are maintained when training the network. When splitting the data like this it must also be ensured that all of the system dynamics are present in the first 70% (equivalent to 11.2s) of data so that the network is trained across the system's entire range of dynamics. The validation and test data should be different but remain within the trained dynamic range.

The network is initially trained in an open loop configuration. This is where the feedback loops are disconnected and the known output target data is fed into the first hidden layer instead. This improves training performance and decreases training time. The feedback loops are then reconnected and the performance of the network is evaluated. The closed loop performance is always inferior to the open loop performance. If the closed loop system performance is unsatisfactory, the network can be retrained (initialised with the weights and biases found during open loop training) in its closed loop configuration. This generally improves performance but takes much longer to train. A comparison of target and output data of the trained motor network for the slow mode are shown in Figure 6-15 and Figure 6-16 and for the fast mode, Figure 6-17 and Figure 6-18 respectively.

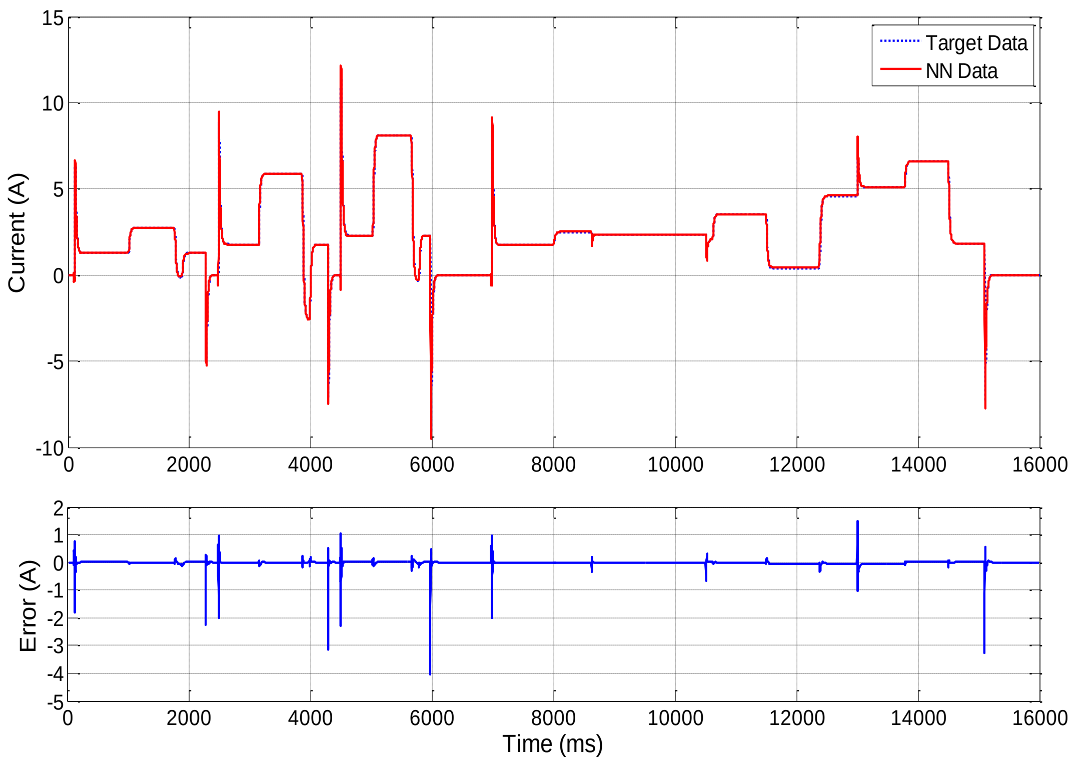


Figure 6-15: Motor NARX Model Slow Mode Performance (Current)

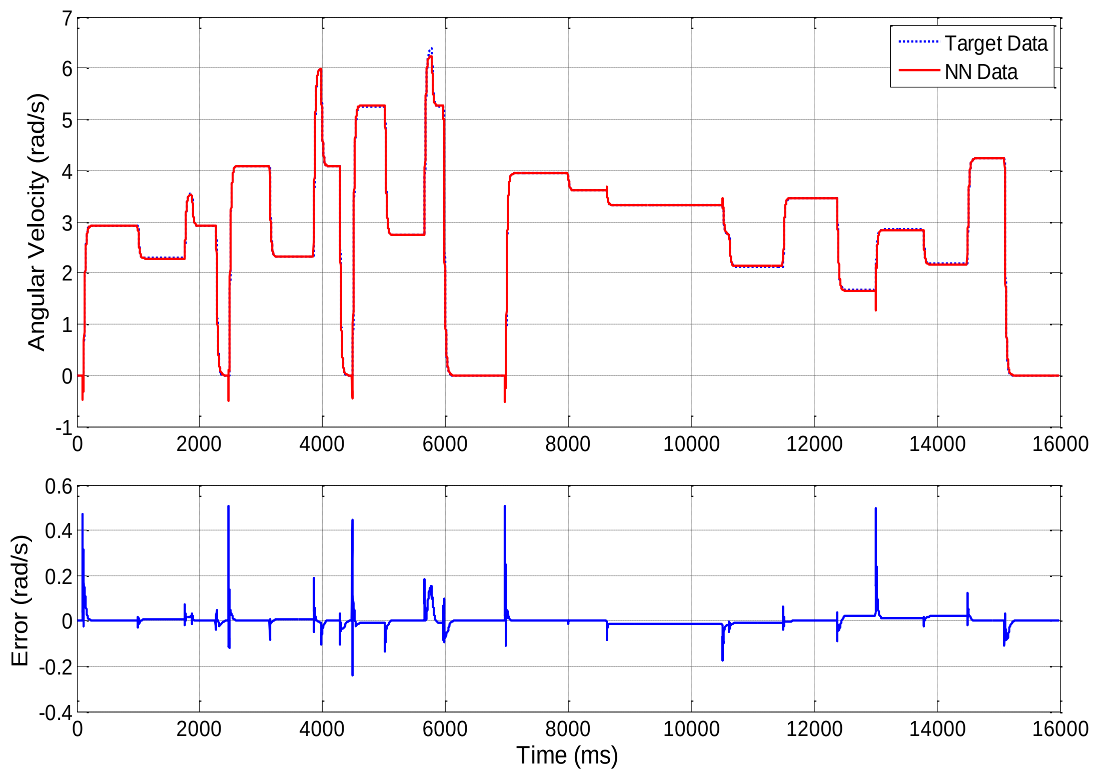


Figure 6-16: Motor NARX Model Slow Mode Performance (Velocity)

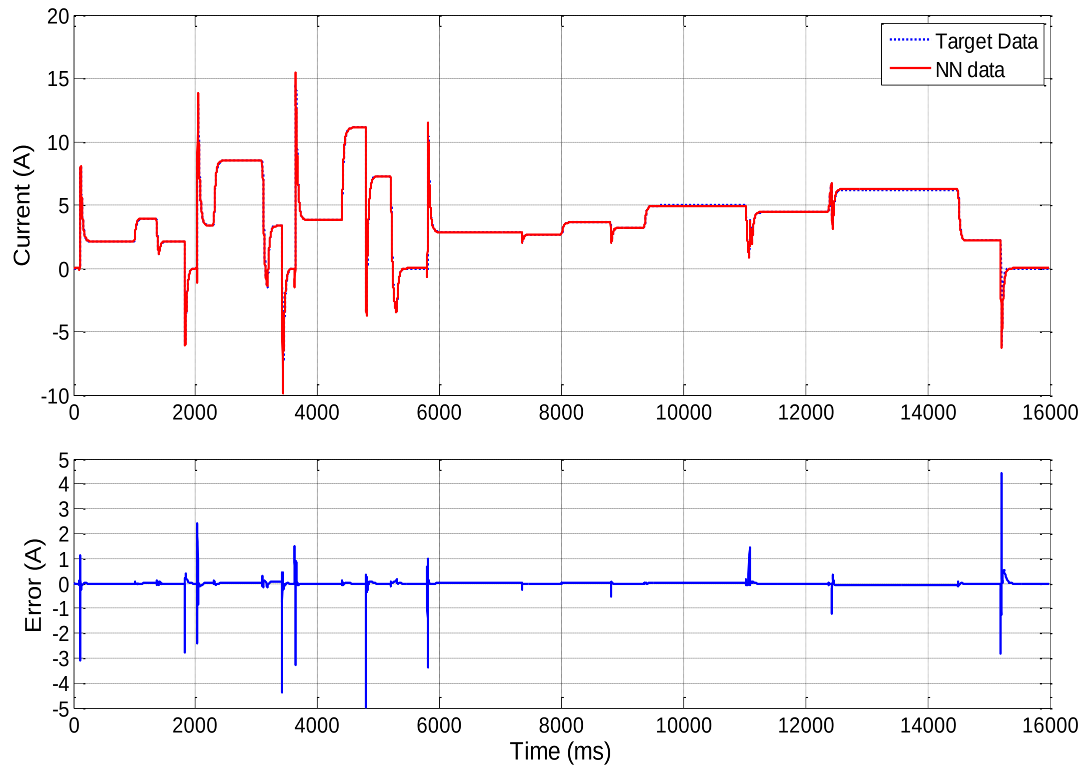


Figure 6-17: Motor NARX Model Fast Mode Performance (Current)

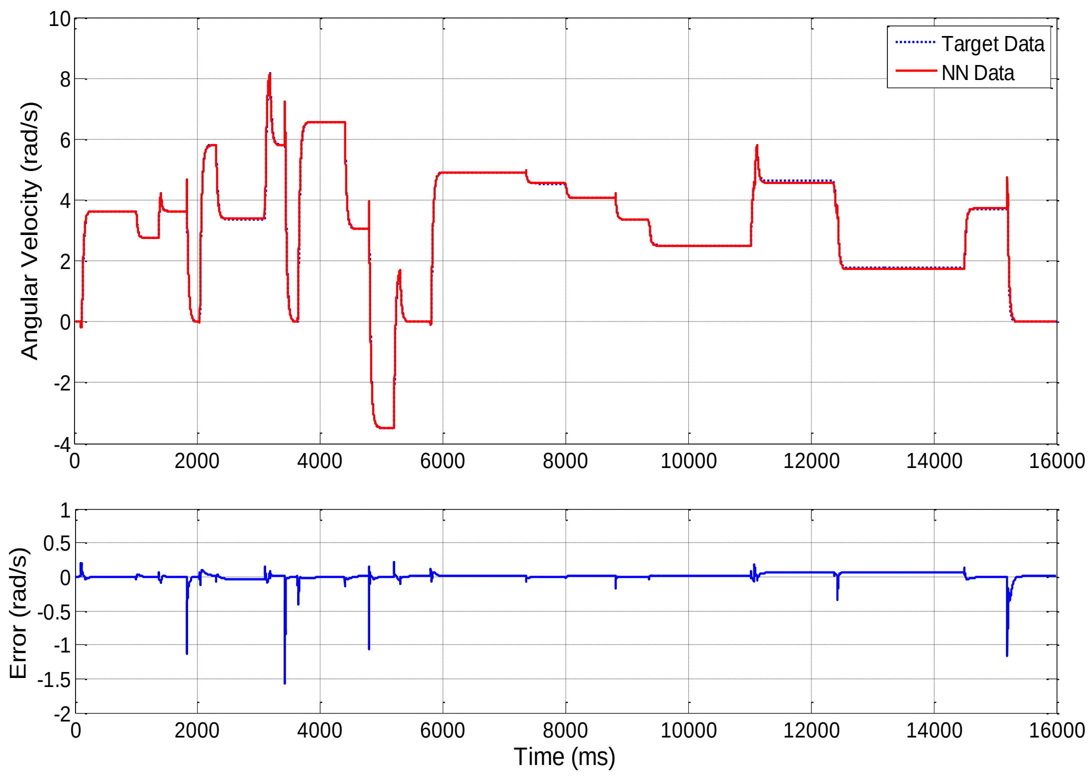


Figure 6-18: Motor NARX Model Fast Mode Performance (Velocity)

The figures show that the network is very successful in modelling the dynamics of the motor across its full range of operation. The network has no problem in modelling both the transient and the steady-state behaviour of the motor across a range of different voltages and torque loads. Table 6-3 shows the MSE of the network when trained in open loop and closed loop configurations. The results demonstrate that the best performance is seen in open loop mode; however this clearly cannot be implemented in a simulation model because the target data is unknown and therefore it must be implemented in its closed loop configuration. It can be seen that by retraining the network in its closed loop configuration the closed loop system performance improves by an order of magnitude, however, training time is significantly longer than for open loop training (for less epochs).

Table 6-3: Motor NARX Approximation Performance (Open and Closed Loop)

		Training MSE				Epochs	Time
Training		**Train**	**Validate**	**Test**	**Total**		
Open Loop	Open Results	0.0040	2.5e-4	0.0049	0.0036	81	1m37s
	Closed Results	0.0784	0.0237	0.0591	0.0673		
Closed Loop		0.0068	0.0038	0.0094	0.0068	56	7h19m

6.3.1 Simulink Implementation

The torque load and the motor NN models are now combined and implemented in Simulink for deployment in a HIL system. Figure 6-19 shows the top level Simulink diagram of the system. The motor is controlled via a Stateflow controller which turns the motor on and off and selects the motor's operating state (i.e. fast, slow or intermittent). The controller uses the park switch feedback of the motor to determine when to switch between states. The input battery voltage is set at 16V and the torque load can be switched between wet a dry conditions.

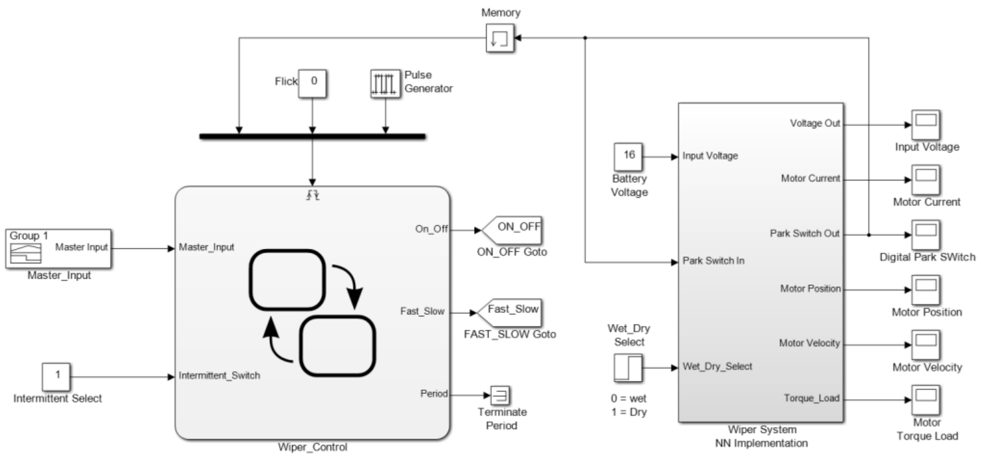


Figure 6-19: Full Wiper System NN Implementation (Top Level)

Figure 6-20 shows the "Wiper System NN Implementation" subsystem from Figure 6-19. The battery voltage (which is turned on and off via switch 5) and the output from the torque load NN form the inputs to the motor NN. The fast and slow output velocities from the motor NN are integrated to obtain the position of the motor output shaft. The position and velocity is then routed to the input of the torque load NN, via switch 3 which connects either the fast or slow motor dynamics. Once the position of the chosen motor output shaft reaches 2π radians, the park switch is switched to high (using switches 1 and 2) and fed back into the controller. The park switch also resets the position of the motor output shafts to zero, generating the saw style input that the torque load NN requires. The dry or wet condition is selected using switch 4 before it is fed back into the motor NN.

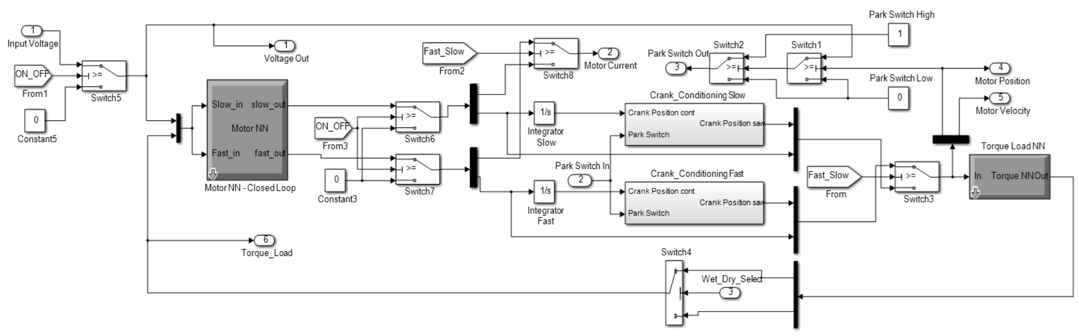


Figure 6-20: Full Wiper System NN Implementation (Bottom Level)

The system shown in Figure 6-19 was measured to simulate 60s (simulation time) in 2.17s (real time) making it faster than the hybrid physical-NN model shown in section 6.2.1 and therefore suitable for HIL simulation. Figure 6-21 and Figure 6-22 show simulation results of the model shown in Figure 6-19. Figure 6-21 shows the wiper motor current and the position of the motor's output shaft, rising from zero to 2π radians before resetting back to zero. Figure 6-22 shows the torque load applied to the motor and the velocity of the motor output shaft. For the first three wipes the slow motor state and wet windscreen conditions are chosen, and then the motor is switched to its fast mode for four wipes. The motor is switched off until 12s where it is switched back on in its slow mode but with dry windscreen conditions. After three wipes the motor is switched to its fast mode for four wipes, after which it is switched off. It can be seen from Figure 6-21 and Figure 6-22 that the NN implementation of the full wiper system can successfully capture the dynamics of the wiper system across its full operating range, at a much faster simulation speed than the physical model developed previously.

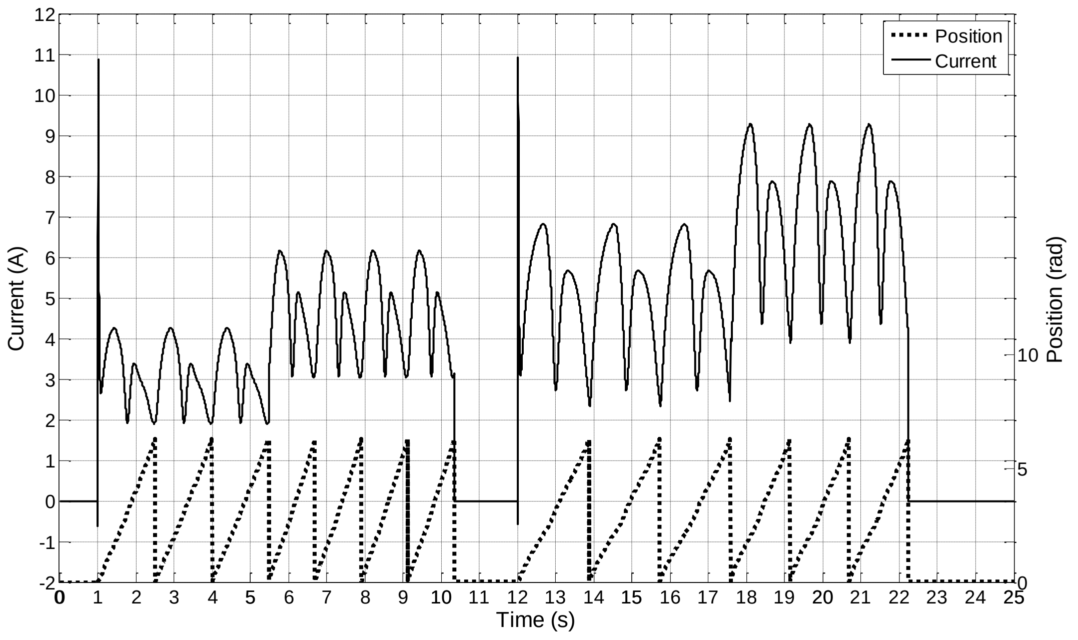


Figure 6-21: Full Wiper System NN Simulink Results (Current and Position)

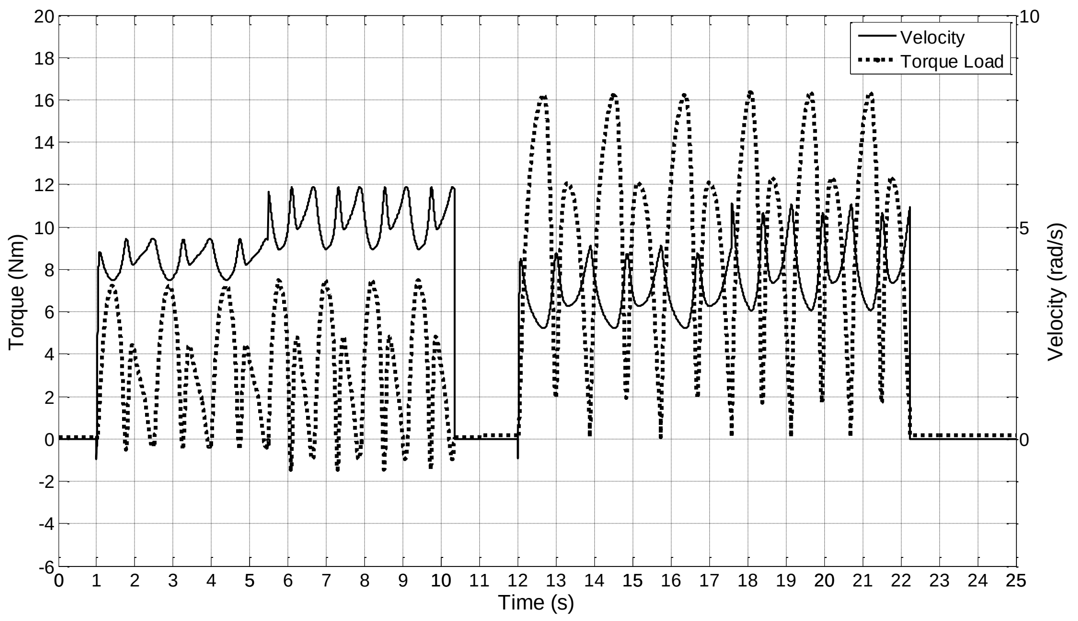


Figure 6-22: Full Wiper System NN Simulink Results (Torque and Velocity)

6.4.4 Neural Network Limitations

A weakness of NNs is that they are unable to extrapolate the behaviour of the modelled system beyond the training data. For example, if the input voltage to the wiper motor physical model was increased to, say, 25V, the model would be able to simulate the behaviour. However, the NN model would fail because it was only trained up to 18V. For this reason care must be taken when selecting training data and when running the NN models.

An example of the NN models developed in this chapter failing is shown in Figure 6-23. In this case the NN motor model is subject to a constant torque when the motor is switched off[16] at around 5.2s. This causes a large error in both the motor velocity and current. It should also be noted that the NN model fails to recover once the motor inputs return to their normal, trained values at around 5.8s. Thus, the NN models developed here would not be suitable for implementations such as fault insertion testing.

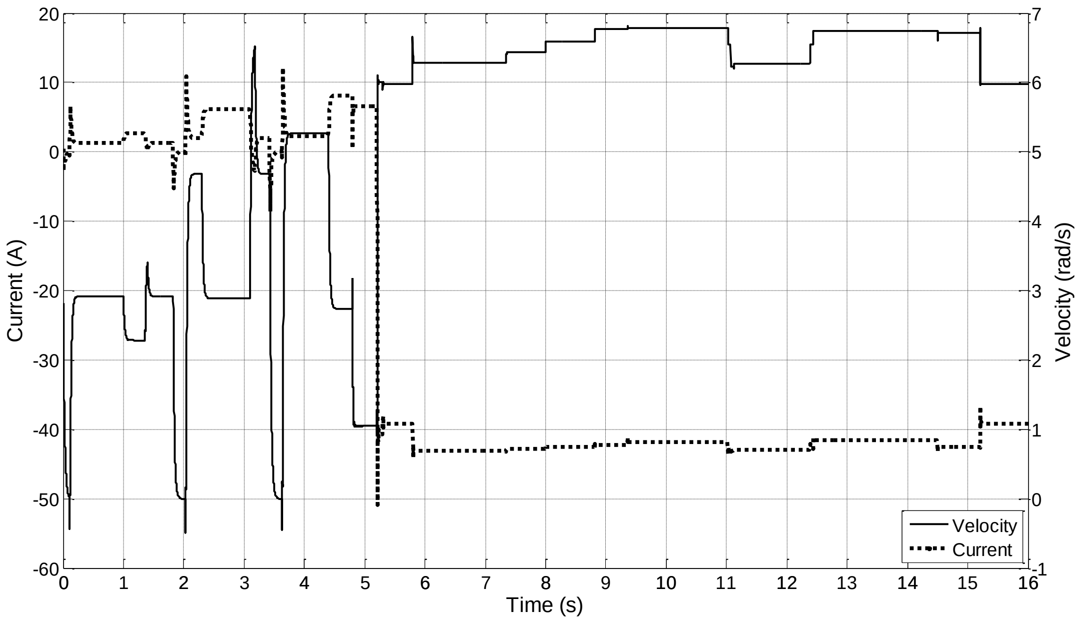


Figure 6-23: Neural Network Limitation Example

[16] The implemented simulation models prevent this from happening to emulate the worm and wheel gear system in the motor.

6.5 Discussion

The goal of the work presented in this chapter was to modify the wiper system model developed, parametized and validated in the previous three chapters in order to make it suitable for real-time simulation, and thus HIL implementation. It was decided that the best way to achieve this was to replace the SimMechanics element of the physical model with a model which could be simulated outside of the physical domain. Doing this allowed the simulation speed of the model to be increased for two reasons. Firstly, local Simscape solvers could be used to solve the physical network which are faster than the global solvers (and could not be used if a SimMechanics model was attached to the physical network). Secondly, models could be developed that only simulated elements of the system that were needed, making them inherently faster to simulate.

The first step was to make use of the fact that the mechanical system is a fully kinematically defined system of rigid bodies with one DOF. This meant that the positional behaviour of any point on the system could be represented using look-up tables driven by the position of the motor output shaft. The velocity and acceleration behaviour could then be obtained by differentiating the position.

The second step was to model the torque load applied to the motor by the wiper system's mechanical element. Initially, this was done for the system's so called no-load condition, which only takes into account the load applied by the linkages. The torque was modelled using polynomials to define the fundamental shape of the torque load, with respect to the motor shaft position, across its forward and reverse wipe. An equation was then developed to model the effect of the motor velocity on the torque, whose unknown constants were identified using a GA. By replacing the SimMechanics implementation of the linkages with the polynomial implementation, the model went from taking 86s (real-time) to simulate 60s (simulation time) to taking 4s to simulate 60s – making it suitable for real-time and HIL simulation.

The third step was to model the whole mechanical system using feed forward NN. An investigation revealed that, in this case, a NN with 10 hidden layers with 10 neurons each are the optimal dimensions. The network was trained to simultaneously output the torque developed in the wet and dry windscreen conditions from input data consisting of the motor output shaft's position and velocity. The network was trained with data from the physical model. The trained network successfully models the torque with an average MSE of 0.0951 for the training data and 0.1102 for the validation data. A simulation model of the wiper system with the torque NN replacing the SimMechanics system was measured as simulating 60s (simulation time) of data in 5.92s (real time).

The final step was to attempt to model the entire wiper system using NNs. For this, a NN modelling the wiper motor that could be interfaced with the torque NN was developed, based on the principles of a NARX network. The motor NN accepts voltage and torque (from the torque NN) inputs and outputs the motor current and output shaft velocity for both fast and slow modes of operation. The trained motor NN is able to model the dynamics of the wiper motor with an average MSE of 0.0068, however to achieve this low MSE the NN had a closed loop training time of 7h19m. Both the motor NN and torque NN are implemented together in a Simulink model and successfully simulated the entire wiper system. The full NN system can simulate 60s (simulation time) in 2.17s (real time) making it the fastest model developed here.

Chapter 7 - Adaptive Control of the Wiper System

7.1 Introduction

This chapter demonstrates a number of control techniques designed to implement constant velocity control of the windscreen wiper system. Currently, wiper systems have little in the way of direct velocity control and rely on decisions from the driver and sometimes a rain sensor to determine a number of wipes per minute, measured using the park switch. A more detailed example of this type of control is given in section 7.1.1. It should be noted that this type of control is justified because the wiper system does not need accurate speed control to perform its task and the system can receive immediate intelligent feedback and control from the driver should its performance be insufficient.

However, there are benefits to implementing more accurate speed control in terms of reducing power consumption and wear on the brushes. Because of the nature of the wiper motor system, the only variable that can be controlled is the input voltage. Therefore, to reduce power consumption, the input voltage must be reduced. Thus, the control strategy is to specify a constant velocity which is the slowest velocity, and thus lowest voltage, needed to adequately clear the windscreen. This is not achieved with the current control system because only discrete speeds can be chosen, and thus it is likely that the wiper motor will be rotating faster than in needs to. By driving the motor at its slowest possible speed, the voltage can be reduced and thus the power consumption is reduced. Also, operating the motor at a lower speed will reduce the wear on the brushes, thus increasing their lifetime [105].

It was shown in Chapter 4 that the torque load on the motor is highly dynamic, which has an effect on the velocity of the motor. Thus a control system must be able to maintain a constant velocity when subject to the dynamic load torque. Because the load applied to the motor is variable, adaptive control techniques are used to address the disturbances to the system and maintain a constant velocity, whilst not compromising on the overshoot of the system. The control systems developed in this chapter are based on PID control with the error signal being formed from the set point velocity minus the measured velocity of the motor. The control variable is the input voltage to the motor (and the speed state, i.e. fast or slow). Initially, a classic non-adaptive PID control system is developed to demonstrate its drawbacks and provide a benchmark for the adaptive controllers developed subsequently.

The adaptive controllers are based on Single Neuron PID (SNPID) control, where the gains of the PID controller are replaced by the weights of the inputs to a neuron, which can be adjusted in real-time based on a learning rule. The SNPID controllers are then augmented using a fuzzy inference system to vary the overall gain of the system. This adds another layer of adaptability to the controller and can be used to achieve constant optimal performance in real-time.

There are clear advantages in applying accurate speed control in the wiper system, however implementing the system in a real vehicle will require a speed sensor and a DC-DC converter to control the input voltage. Current systems do not incorporate these features. The vehicle manufacturer must trade off the advantages of such a control system against the cost and complexity of its implementation.

7.1.1 Classic Wiper Control Example

An example of an existing wiper control system is shown in Figure 7-1. The specifics and complexity of the system are dependent on the model of the vehicle. A discrete wiper speed is selected by the driver (and/or the rain sensor if one is available) which determines whether the motor is operated in intermittent mode (if available), slow continuous mode or fast continuous mode. The implementation of the ON/OFF and FAST/SLOW states of the motor is realised using two relays with form-C contacts whose states are switched using the

ECU. The ON/OFF relay can connect to electrical ground (for OFF) or the vehicle's battery (for ON). This is then routed to the FAST/SLOW relay which connects to both inputs of the wiper motor. If the ON/OFF relay is connected to the battery, current is supplied to the motor and it drives the wipers. Depending on the state of the FAST/SLOW relay, the wiper operates in its fast or slow operation. The motor brush not in use is left open. The park switch from the motor is fed back to the ECU for average speed measurement and control purposes.

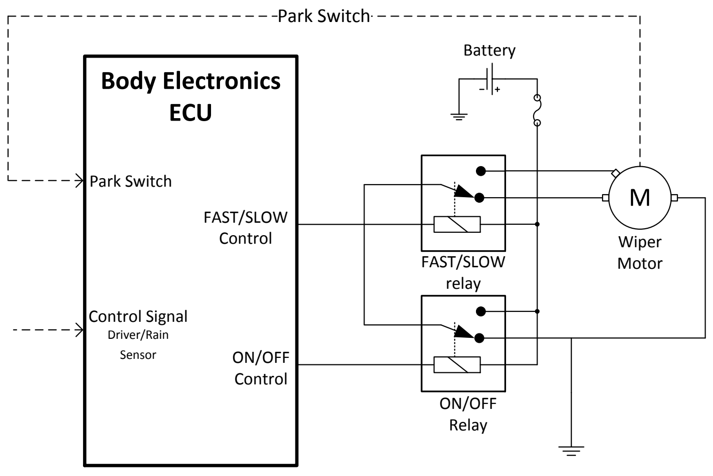


Figure 7-1: Classic Wiper System Control

7.2 PID Velocity Control

PID (Proportional-Integral-Derivative) control is a widely used method of control which is often used as a benchmark for assessing the performance of more advanced control techniques. Stated in the Laplace domain, the transfer function relating the output control signal $U(s)$ to the input error $E(s)$ for a PID controller is

$$G_{PID}(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + sK_d \,, \tag{7.1}$$

where K_p is the proportional gain which determines the rise time and overshoot, K_i is the integral gain which determines the steady state accuracy and K_d is the derivative gain which overcomes oscillations. In classic PID control all of the gains are constants and can be equal to zero. There are many techniques for tuning the gains of a PID controller and their optimal values depend on the performance requirements of the system. Often there must be a trade-off between performance measures, for example a PID controller tuned to minimise overshoot will likely have a slow rise time and poor reference tracking in the presence of disturbances.

A PID controller is used to achieve constant velocity control of the wiper system using the tuning algorithms built into Matlab. A block diagram showing the system is given in Figure 7-2. The reference velocity begins at π rad/s, then at 10s it switches to 1.2π rad/s and at 15s it switches to 0.8π rad/s. The torque load operates in its wet condition up until 5s, then it switches to dry. At 12s the load state switches back to wet. The control system is tested over a range of input velocities and torque loads using these test signals. The control voltage is limited to $0 \le U(s) \le 20V$.

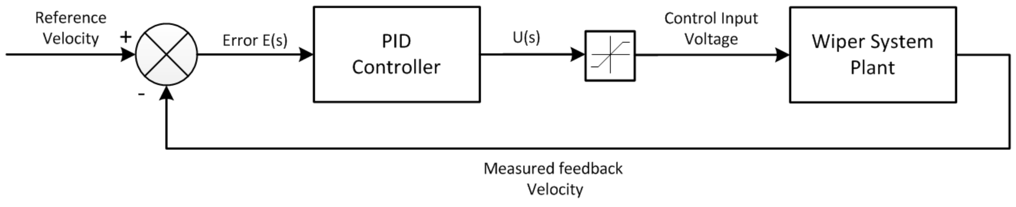


Figure 7-2: PID Controller Block Diagram

The tuned PID parameters (tuned in Simulink) are shown in Table 7-1.

Table 7-1: Tuned PID Controller Gains

Controller Gain	Value
K_p	0.00177
K_i	35.452
K_d	0

The results of the simulation are shown in Figure 7-3. It can be seen that the control system successfully tracks the input reference velocity across all velocities and loads. However the system has relatively poor disturbance rejection[17], particularly at higher speeds and torque loads. Numerical results are shown in Table 7-2. The overshoot and rise time are measured from when the motor is first switched on. The rise time is defined here as the time taken to rise from zero rad/s to π rad/s. The errors in disturbance rejection are divided into the five system states used for testing, which are shown in Table 7-3. The errors with the highest positive and negative errors were recorded for each state.

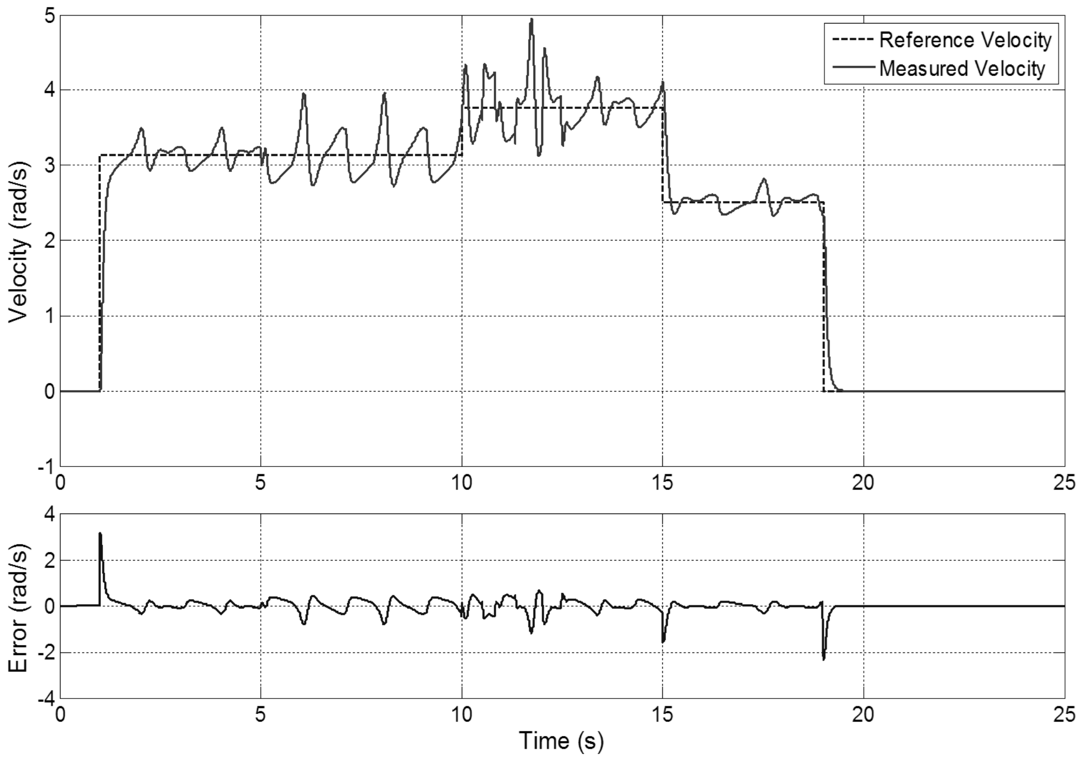


Figure 7-3: PID Control Graphical Results

[17] In this case the torque load on the motor is considered as a disturbance to the system.

Table 7-2: PID Control Numerical Performance

	Over-shoot (rad/s)	Rise Time (s)	State 1 Error (rad/s)		State 2 Error (rad/s)		State 3 Error (rad/s)		State 4 Error (rad/s)		State 5 Error (rad/s)	
			Low(-)	High	Low(-)	High	Low(-)	High	Low(-)	High	Low(-)	High
PID	0	0.765	0.35	0.2	0.818	0.416	1.179	0.453	0.410	0.263	0.311	0.186

Table 7-3: Control System Test States

State Number	Target Velocity	Load Condition
State 1	π radians	Wet
State 2	π radians	Dry
State 3	1.2π radians	Dry
State 4	1.2π radians	Wet
State 5	0.8π radians	Wet

It has been stated that the PID control system shown in this section has relatively poor disturbance rejection, and thus the velocity of the motor varies quite considerably. This could be improved by changing the PID controller parameters; however doing so would negatively impact the overshoot of the velocity. In the next two sections, adaptive control techniques are used to overcome this and improve the overall performance of the system.

7.3 Single Neuron Adaptive PID Control

In Single Neuron PID (SNPID) control the PID controller portrayed in equation (7.1) is replaced by a single neuron with three inputs and one output. As in the case of a full ANN, the inputs to the neuron are subject to weight elements which can be adjusted to produce a desired output of the neuron based on some learning rule. The weights of the SNPID effectively replace the gain parameters of a classic PID controller and can be adjusted in real-time, thus making SNPID a form of adaptive control. SNPID controllers combine the advantages of neural networks, i.e. the ability to learn and adapt their behaviour, with the advantages of PID control. The basic element of the single neuron PID controller is shown in Figure 7-4.

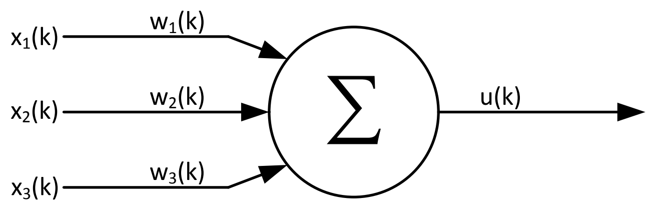


Figure 7-4: Single Neuron Model

The three inputs to the neuron are defined as follows

$$\begin{aligned} x_1(k) &= e(k) \\ x_2(k) &= e(k) - e(k-1) \\ x_3(k) &= e(k) - 2e(k-1) - e(k-2) \end{aligned} \quad , \tag{7.2}$$

where $e(k)$ is the error in the system at time k. Therefore, $x_1(k)$ corresponds to current error in the system, $x_2(k)$ to the first differential of the error and $x_3(k)$ to the second differential. The three states are analogous to the incremental form of the PID controller equation shown in equation (7.3)

$$\Delta u(k) = k_p\{e(k) - e(k-1)\} + k_i e(k) + k_d\{e(k) - 2e(k-1) - e(k-2)\} \ , \tag{7.3}$$

where $\Delta u(k)$ is the incremental output of the controller which is added to the previous control value (see equation (7.5))

It can be seen from Figure 7-4 and equation (7.3) that the weights of the single neuron act as the PID gains. Hence, the output of the single neuron at time k can be written as

$$\Delta u(k) = w_1 e(k) + w_2\{e(k) - e(k-1)\} + w_3\{e(k) - 2e(k-1) - e(k-2)\} \ . \tag{7.4}$$

Figure 7-5 shows the entire SNPID control system. The incremental output of the single neuron, $\Delta u(k)$, is multiplied by a factor K which is the proportionality coefficient of the neuron and has a large effect on the output. The error signal is defined as $e(k) = \{r(k) - y(k)\}$ where $r(k)$ is the reference signal and $y(k)$ is the measured output from the plant that the system seeks to control. The weights are varied at each time step by some learning rule, the inputs to which depend on the procedure used in the rule, but it is always dependent on the error.

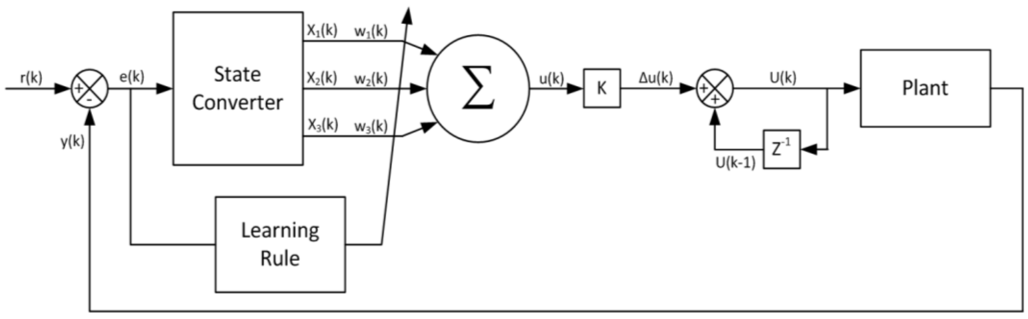


Figure 7-5: Single Neuron PID Control System Block Diagram

From Figure 7-5 it can be deduced that the output of the system, i.e. the control input to the plant is

$$U(k) = U(k-1) + \Delta u(k), \tag{7.5}$$

where

$$\Delta u(k) = K \sum_{i=1}^{3} \hat{w}_i(k) x_i(k), \tag{7.6}$$

and $\hat{w}_i(k)$ is the normalised[18] value of weight i and is defined as

$$\hat{w}_i(k) = \frac{w_i(k)}{\sum_{j=1}^{3} |w_j(k)|} . \tag{7.7}$$

[18] The weights are normalised to improve robustness and convergence of the algorithm.

Three learning rules to update the weights of the neuron are implemented and compared. These are the Hebb learning rule as used in references [90] [94] [91] [92] and [93], the Error-Hebb learning rule proposed in reference [94] and the quadratic learning rule used in [89] and [88]. Three learning rules are being investigated because it is not clear from theory which will provide the best performance in this case.

7.3.1 Hebb Learning Rule

The equations describing the weights at time $(k+1)$ for the Hebb learning rule are given as

$$\begin{aligned} w_1(k+1) &= w_1(k) + \eta_I e(k) u(k) x_1(k) \\ w_2(k+1) &= w_2(k) + \eta_P e(k) u(k) x_2(k) \ , \\ w_3(k+1) &= w_3(k) + \eta_D e(k) u(k) x_3(k) \end{aligned} \tag{7.8}$$

where η_I, η_P and η_D are the learning rates of the integral, proportional and derivative elements of the algorithm, respectively.

7.3.2 Error Only Hebb Learning Rule

The equations describing the Error-Hebb learning rule are similar to the standard Hebb learning rule but replace the state terms of the equation with a term derived from the error signal:

$$\begin{aligned} w_1(k+1) &= w_1(k) + \eta_I e(k) u(k) \{e(k) + \Delta e(k)\} \\ w_2(k+1) &= w_2(k) + \eta_P e(k) u(k) \{e(k) + \Delta e(k)\} \ , \\ w_3(k+1) &= w_3(k) + \eta_D e(k) u(k) \{e(k) + \Delta e(k)\} \end{aligned} \tag{7.9}$$

where $\Delta e(k) = e(k) - e(k-1)$.

7.3.3 Quadratic Learning Rule

The equations describing the Quadratic learning rule are given as

$$\begin{aligned} w_1(k+1) &= w_1(k) + \eta_I K \left[Pb_0 e(k) x_1(k) - QK \sum_{i=1}^{3} \left(w_i(k) x_i(k) \right) x_1(k) \right] \\ w_2(k+1) &= w_2(k) + \eta_P K \left[Pb_0 e(k) x_2(k) - QK \sum_{i=1}^{3} \left(w_i(k) x_i(k) \right) x_2(k) \right], \quad (7.10) \\ w_3(k+1) &= w_3(k) + \eta_D K \left[Pb_0 e(k) x_3(k) - QK \sum_{i=1}^{3} \left(w_i(k) x_i(k) \right) x_3(k) \right] \end{aligned}$$

where P and Q are weight values of output error and control increments respectively and b_0 is the initial value of the control output (set to 1 in this case for simplicity).

7.3.4 Single Neuron Adaptive PID Control Results and Discussion

The SNPID control system using the three learning rules discussed are now tested and compared to the PID control system described in section 7.2. The same control and load inputs to the system are applied and the SNPID control systems are assessed based on the same criteria. The parameters used for each of the learning rules are shown in Table 7-4. The exact values of K used are given on their respective graphs and results table.

Table 7-4: Single Neuron PID Control Test Parameters

Learning Rule	η_I	η_P	η_D	K	P	Q	b_0
Hebb	1	1.5	1	0.08 to 0.5	N/A	N/A	N/A
Error Hebb	1	1.5	1	0.2 to 1	N/A	N/A	N/A
Quadratic	1	1.5	1	0.05 to 0.5	2	1	1

Figure 7-6 shows the results of the SNPID Hebb learning rule simulations, Figure 7-7 shows the results of the SNPID Error-Hebb learning rule and Figure 7-8 shows the results of the SNPID quadratic learning rule. Results of all the learning rules are shown in Table 7-5 showing the overshoot, rise time and maximum positive and negative errors in the five states (see Table 7-3 for state details).

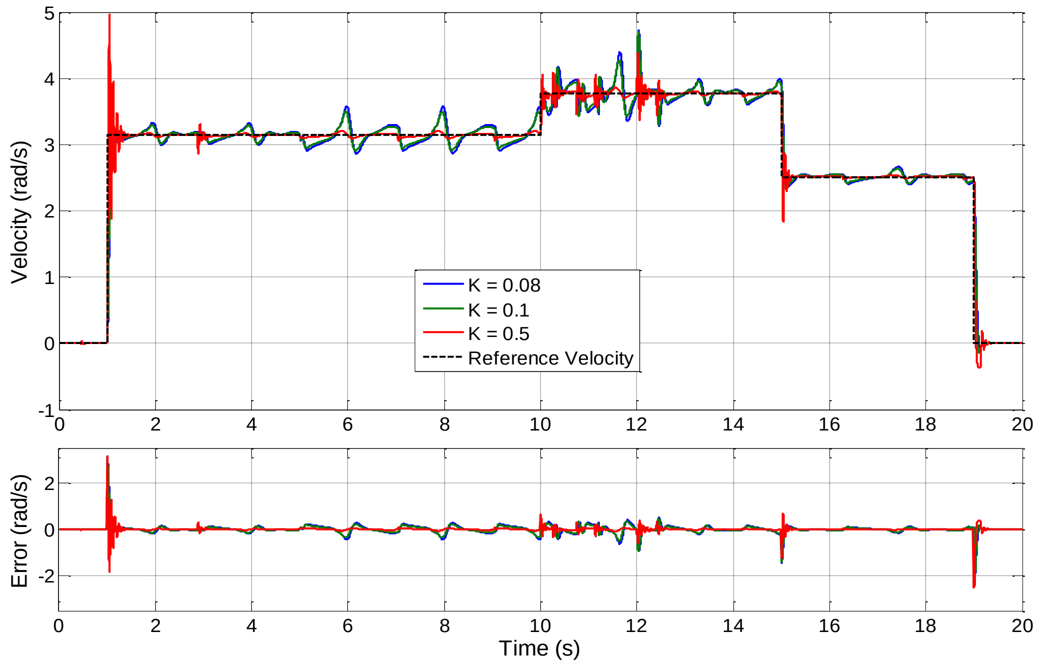


Figure 7-6: Single Neuron Hebb Learning Rule Results

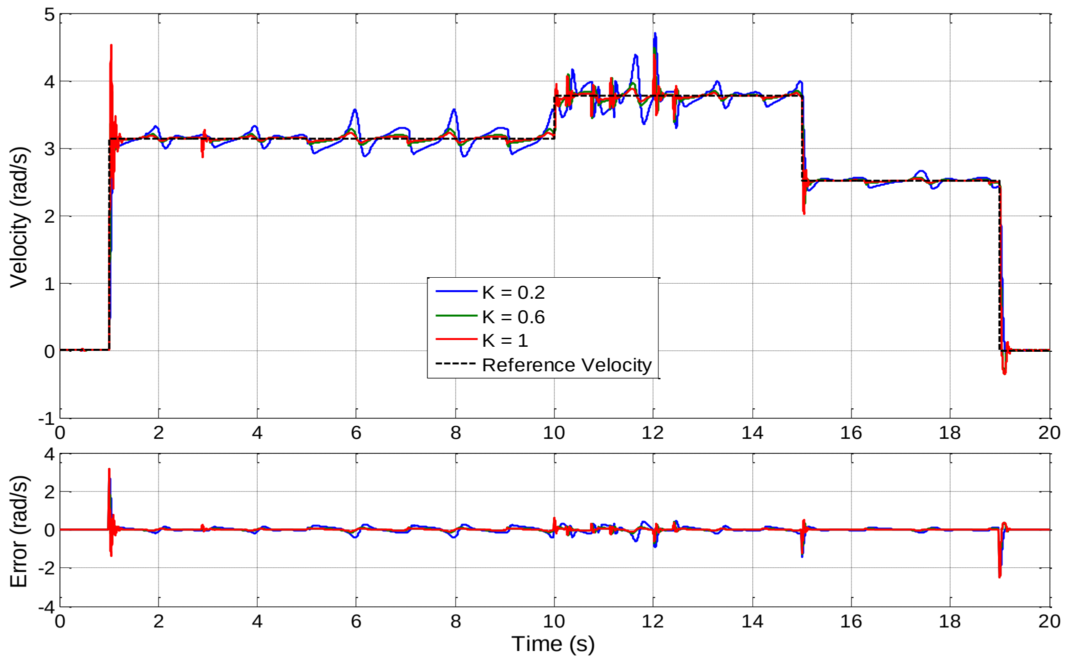


Figure 7-7: Single Neuron Error-Hebb Learning Rule Results

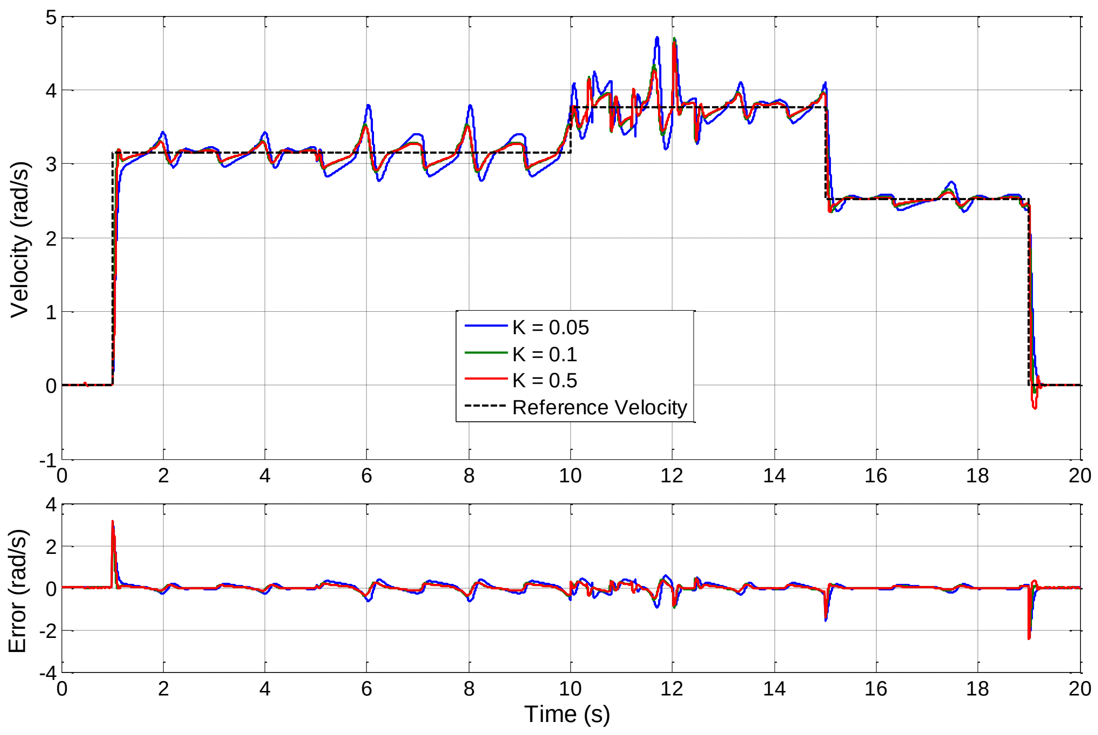


Figure 7-8: Single Neuron Quadratic Learning Rule Results

Table 7-5: Single Neuron Control System Results

	Over-shoot (rad/s)	Rise Time (s)	State 1 Error (rad/s)		State 2 Error (rad/s)		State 3 Error (rad/s)		State 4 Error (rad/s)		State 5 Error (rad/s)	
			Low(-)	High	Low(-)	High	Low(-)	High	Low(-)	High	Low(-)	High
Hebb (K=0.08)	0	0.127	0.189	0.151	0.435	0.277	0.626	0.421	0.223	0.184	0.153	0.118
Hebb (K=0.1)	0.137	0.082	0.156	0.127	0.355	0.2315	0.507	0.342	0.183	0.158	0.124	0.097
Hebb (K=0.5)	1.826	0.028	0.031	0.026	0.067	0.049	0.281	0.309	0.037	0.045	0.025	0.020
E-Hebb (K=0.2)	0	0.126	0.184	0.148	0.427	0.272	0.615	0.411	0.220	0.180	0.153	0.117
E-Hebb (K=0.6)	0.739	0.036	0.063	0.054	0.141	0.099	0.268	0.327	0.076	0.069	0.052	0.042
E-Hebb (K=1)	1.387	0.025	0.038	0.032	0.082	0.060	0.264	0.301	0.045	0.041	0.031	0.025
Quadratic (K=0.05)	0	0.716	0.282	0.195	0.656	0.375	0.947	0.584	0.330	0.229	0.242	0.164
Quadratic (K=0.1)	0.052	0.098	0.171	0.138	0.394	0.255	0.569	0.384	0.204	0.172	0.140	0.109
Quadratic (K=0.5)	0.055	0.118	0.155	0.126	0.361	0.232	0.495	0.326	0.176	0.150	0.099	0.079

It can be seen that all three adaptive learning rules out-perform classic PID control. Because the PID controller produced results with no overshoot, one of the K values chosen for each learning rule was also chosen to produce no overshoot, so a direct comparison can be made. All of the SNPID controllers with zero overshoot had faster rise times and lower positive and negative errors in all of the five states than the PID controller. This is due to the weights of the SNPID controller updating in response to the error.

The two best systems were the Hebb learning rule and the Error-Hebb learning rule, although a direct comparison between the two is difficult to make because of the different values of K required. At all chosen values of K the tracking errors were lower than that of the PID controller and at the highest K value the average tracking error was 0.089 rad/s for the Hebb Learning rule and 0.0919 rad/s for the Error-Hebb learning rule.

The weakest of the three adaptive control system tested was the Quadratic Learning rule system. The system generally produced good overshoot and rise time results but the disturbance rejection performance was inferior to the other two adaptive systems. This was because, at higher values of K which would improve the disturbance rejection, the control output began to oscillate and caused an error in the simulation.

The results in Figure 7-6 and Figure 7-7 clearly demonstrate the effect that the value of K has on the performance of the system. A low value of K can achieve zero overshoot whilst maintaining reasonable rise times. However this is at the expense of disturbance rejection, resulting in the system having difficulty tracking the reference input, particularly during high load disturbance, i.e. when the windscreen condition is dry. Conversely, a lower value of K successfully tracks the reference velocity with very small errors, but tends to cause overshoot and oscillations before system settles. Attempting to alleviate this problem by using an adaptive value of K is the subject of the next section.

7.4 Single Neuron Fuzzy Adaptive PID Control

It was demonstrated in the previous section that the K value of a SNPID controller has a large effect on the performance of the controller. If the value of K could adapt to the error in the system the performance of the controller could be further improved. To do this, a Mamdani fuzzy inference system is used whose input is the error signal $e(k)$ and whose output is a variable value of K. Versions of this technique have been implemented in references [90] and [89].

A fuzzy logic system is a type of multivalued logic system which does not use sharp boundaries between its states, and membership of a state exists on a spectrum. Fuzzy logic systems utilise a linguistic approach that impersonates the way human's use and process language. Fuzzy inference systems use fuzzy logic to map an input variable to an output variable and are capable of dealing with uncertain and complicated systems. The design of the fuzzy inference system used here is now shown.

The input error to the system is divided into 7 fuzzy sets: Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (ZO), Positive Small (PS), Positive Medium (PM) and Positive Big (PB). Likewise, the output value of K is divided into four fuzzy sets: Very Small (VS), Small (S), Medium (M) and Big (B). These fuzzy sets must be assigned to corresponding membership functions. The input membership functions are shown in Figure 7-9.

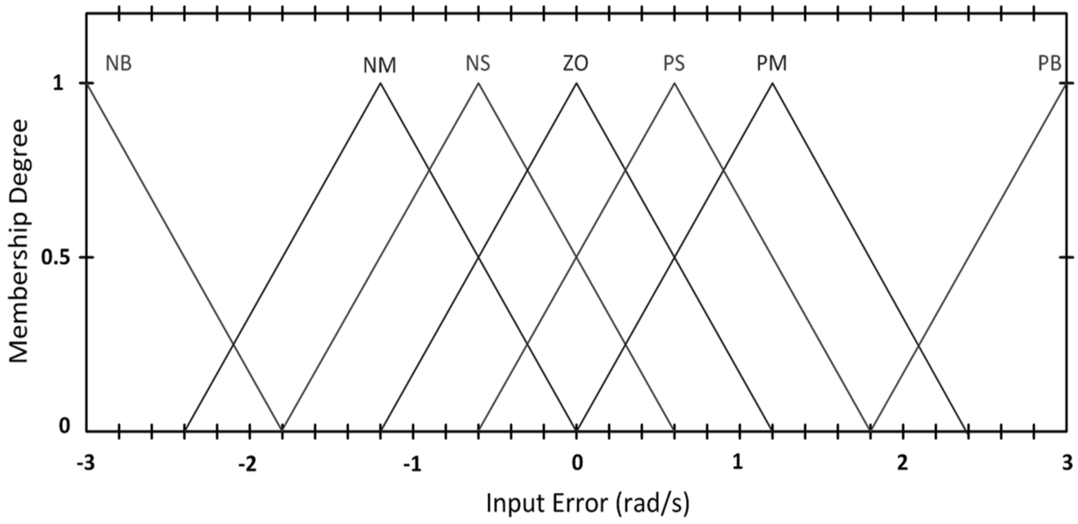


Figure 7-9: Fuzzy Membership Functions – Input

The input error is a real number between ± 3 rad/s. Any error larger than this is treated as ± 3. The value of $e(k)$ will determine which membership function(s) that the input belongs to and by what degree. For example, suppose at time k that $e(k) = -2$. From Figure 7-9 it can be deduced that the input belongs to the NB set by a degree of around 0.125 and the NM set by a degree of around 0.3. The membership functions are grouped tightly around the zero error mark to allow for tighter control during steady state. Larger errors will belong mostly to either the NB or PB sets which allow for a steeper increase in K during transients.

To determine the relationship between the input and output sets, 7 fuzzy rules in the form of "IF *A* THEN *B*" are used. A large error value implies that the system is in transient mode and thus a small value of K is needed. A small or zero error implies that the system is in steady state mode and large value of K is needed to maintain the small error. Hence, the 7 rules are:

1) IF (*Input Error* is *NB*) THEN (*K Value* is VS)
2) IF (*Input Error* is *PB*) THEN (*K Value* is VS)
3) IF (*Input Error* is *NM*) THEN (*K Value* is *S*)
4) IF (*Input Error* is *PM*) THEN (*K Value* is *S*)
5) IF (*Input Error* is *NS*) THEN (*K Value* is *M*)
6) IF (*Input Error* is *PS*) THEN (*K Value* is *M*)
7) IF (*Input Error* is *ZO*) THEN (*K Value* is *B*)

Thus the membership set(s) that the input belongs to, and the degree to which the input belongs to the set(s) is used to map the input sets to the output sets. The output set's membership functions are shown in Figure 7-10.

The VS membership function is a Z-shaped function (rather than a triangle) and mostly does not cross the other membership functions to allow the K value to reach smaller values in response to large transient errors. The centroid de-fuzzification strategy is used to determine the value of K from the membership function(s) that it belongs to, and the degree to which it belongs to it/them.

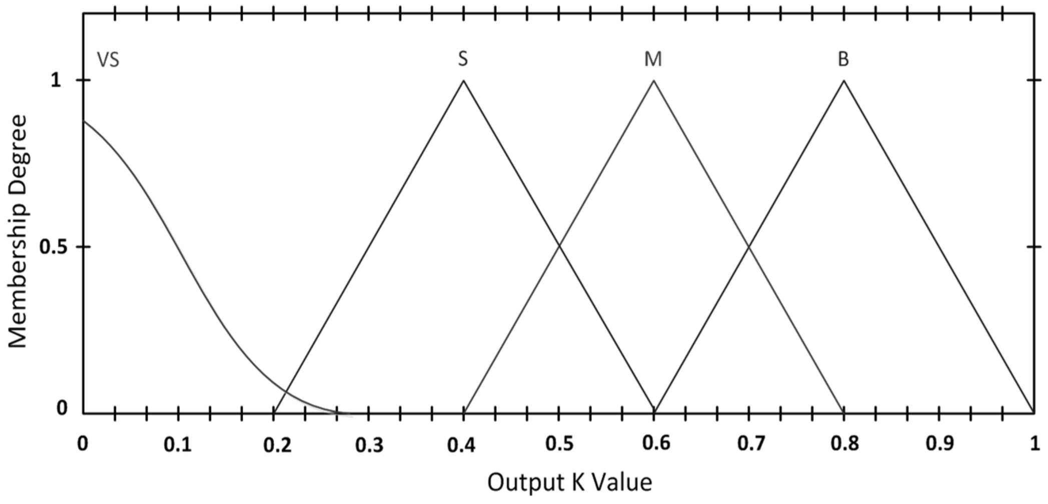


Figure 7-10: Fuzzy Membership Functions – Output

The block diagram of the SNPID controller with the added Fuzzy controller to tune the value of K is shown in Figure 7-11. It can be seen that the controller accepts the error signal and outputs a value of K which is $0 \le K \le 1$. This value is then multiplied by a constant gain to bring it into the range that the controller needs. Note that this could have been achieved by the fuzzy controller without a gain block, however in this way the same fuzzy controller can be used for all SNPID controllers and only the value of the gain needs to be changed.

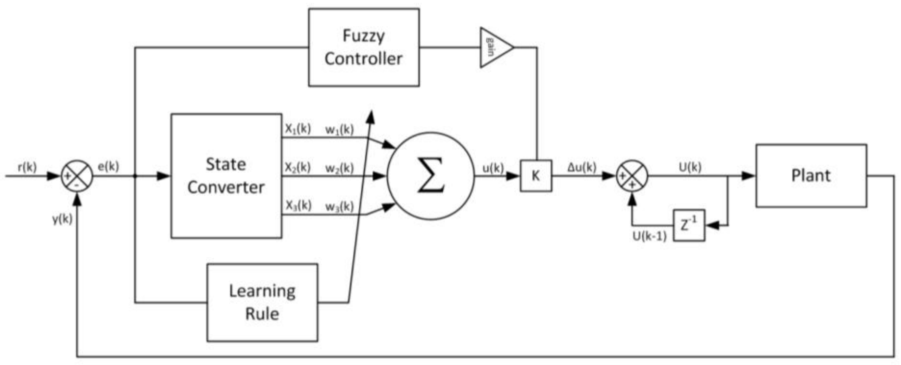


Figure 7-11: Single Neuron Fuzzy PID Control System Block Diagram

7.4.1 SNPID with Fuzzy Controller Results and Discussion

The performance of the SNPID controller with adaptive K value controlled by the fuzzy controller described in section 7.4 will now be given and discussed. The same input conditions, controller parameters and performance criteria used in section 7.3.4 to test the SNPID controllers are applied here.

Figure 7-12 shows the results of the SNPID controller with the Hebb learning rule and fuzzy regulator, Figure 7-13 shows the results of the SNPID controller with the Error-Hebb learning rule and fuzzy regulator and Figure 7-14 shows the results of the SNPID controller with the quadratic learning rule and fuzzy regulator. The value of K is included in the plots to demonstrate how it changes with respect to the error. The gains applied to the outputs of the three fuzzy controllers are 0.5, 1.5 and 0.5 respectively.

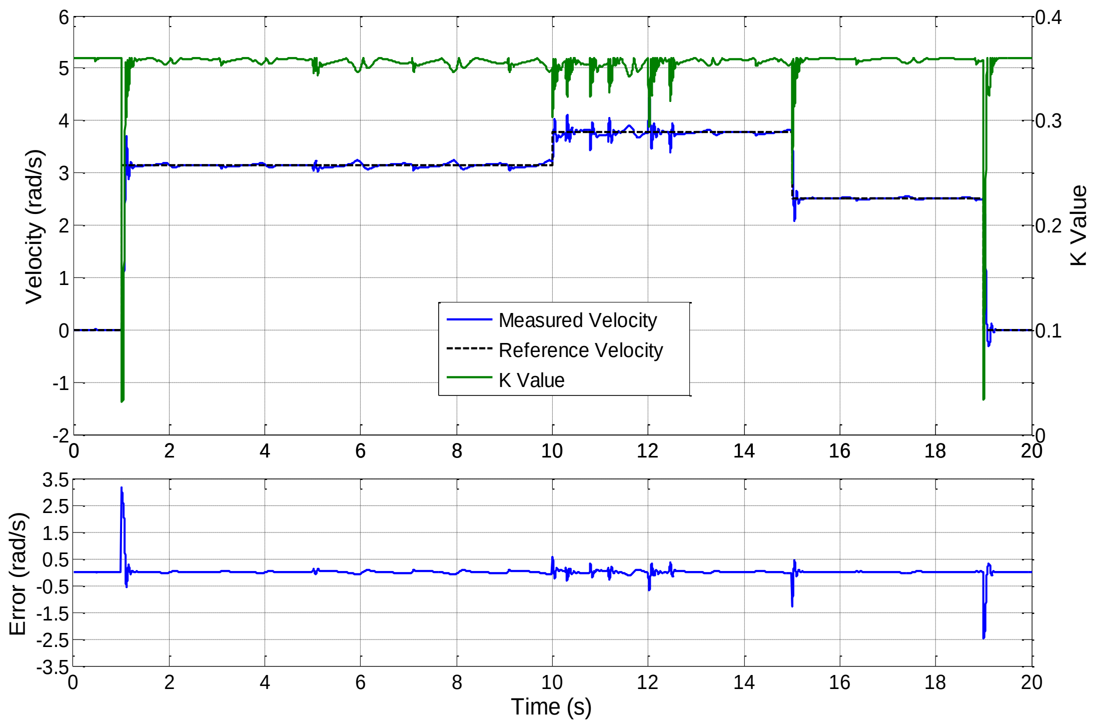


Figure 7-12: Single Neuron Hebb Learning Rule with Fuzzy Controller Results

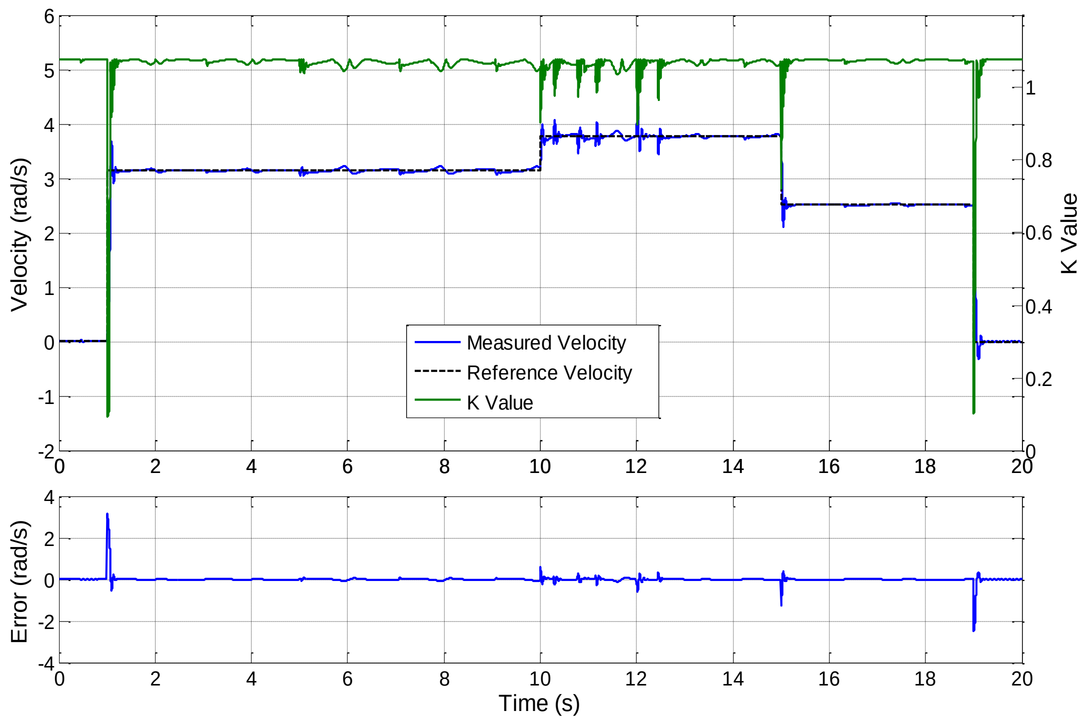


Figure 7-13: Single Neuron Error-Hebb Learning Rule with Fuzzy Controller Results

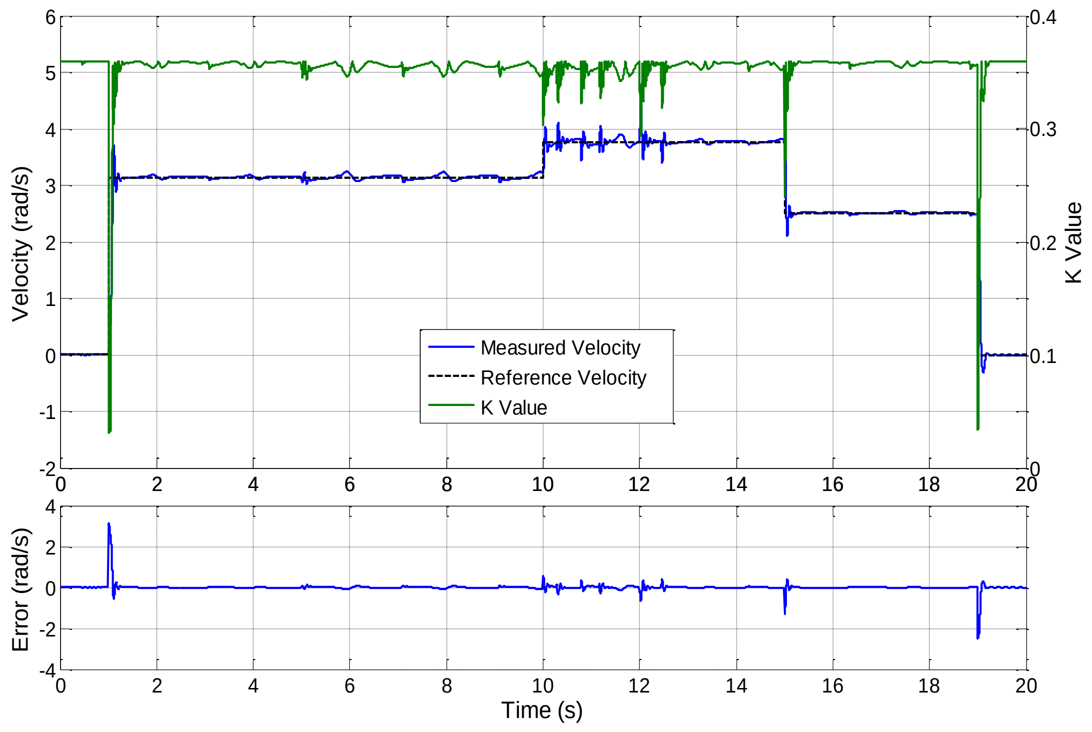


Figure 7-14: Single Neuron Quadratic Learning Rule with Fuzzy Controller Results

Table 7-6 shows the numerical results of the simulations.

Table 7-6: Single Neuron with Fuzzy Control System Results

	Over-shoot (rad/s)	Rise Time (s)	State 1 Error (rad/s)		State 2 Error (rad/s)		State 3 Error (rad/s)		State 4 Error (rad/s)		State 5 Error (rad/s)	
			Low(-)	High	Low(-)	High	Low(-)	High	Low(-)	High	Low(-)	High
Hebb (gain=0.5)	0.559	0.092	0.043	0.037	0.097	0.069	0.133	0.096	0.052	0.047	0.035	0.028
E-Hebb (gain=1.5)	0.529	0.08	0.035	0.030	0.079	0.057	0.109	0.078	0.042	0.039	0.029	0.023
Quadratic (gain=0.5)	0.553	0.094	0.044	0.037	0.099	0.070	0.135	0.097	0.052	0.048	0.038	0.030

It can immediately be seen from the results that the fuzzy controller tuning the value of K has a positive effect on the performance. For all three of the SNPID control systems tested, the controller found a good compromise between reducing the overshoot and rise times whilst rejecting the disturbance caused by the torque load. The largest improvement in performance was found in the SNPID controller with the quadratic learning rule. Without the fuzzy controller, the quadratic learning rule performed significantly worse than the Hebb and Error-Hebb learning rules in terms of disturbance rejection due to instabilities in the system at high K values. However, the addition of the fuzzy controller meant that the performance of the quadratic learning rule system was comparable to the other two, making it a viable option.

7.5 Sensorless Control Neural Network

The goal of the control systems described and simulated in this chapter has been to operate the wiper motor at a reference speed whilst rejecting the disturbances in velocity caused by the dynamic torque load. In order to achieve this, the velocity of the wiper motor is needed. This can be achieved in simulation, however the actual wiper system has no method of continuous speed measurement. In general, vehicle manufacturers are reluctant to add sensors to the vehicle due to cost and complexity. Therefore methods of estimating the volcity must be considered.

One method demonstrated here is to use a feed forward neural network whose inputs are the voltage applied to the motor and the motor current, and whose outputs are the velocity of the motor in its fast and slow mode. A diagram of such a NN is shown in Figure 7-15.

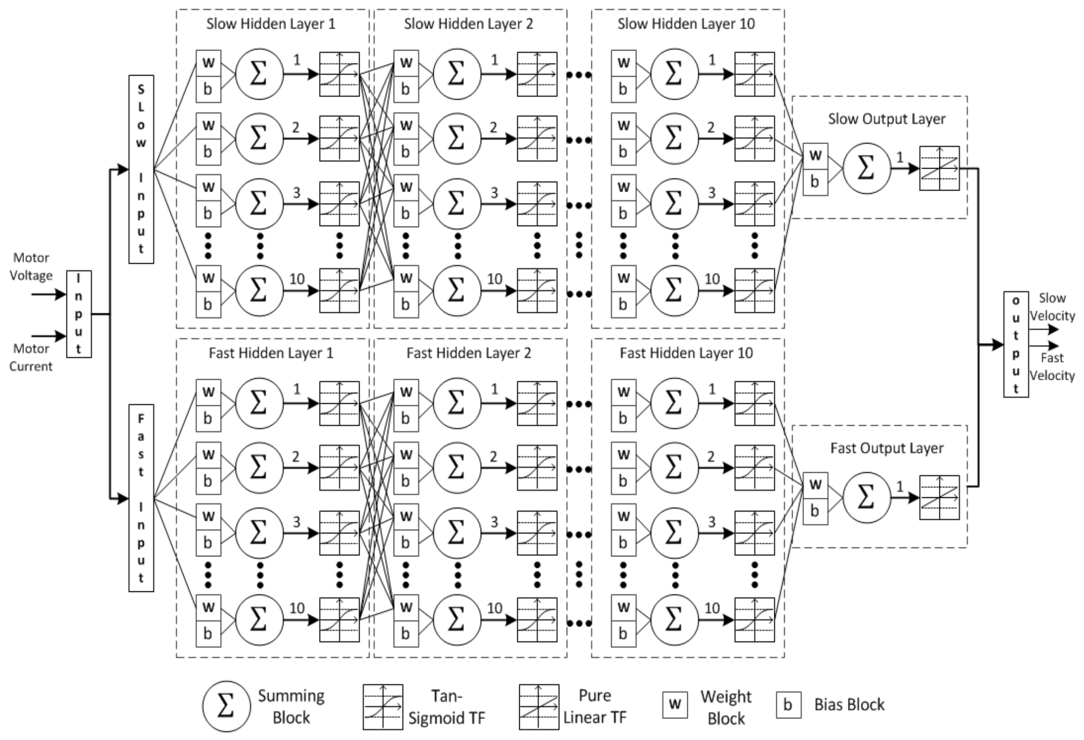


Figure 7-15: Motor Velocity Estimator NN Schematic

The NN shown in Figure 7-15 effectively acts as two networks in parallel with one path mapping the input voltage and motor current to the motor's velocity in its slow state and the other path mapping the inputs to the velocity in its fast state. The inputs to both of the paths are the same in normal operation, although different inputs were used for training. The network is trained across its full range of input voltages in both wet and dry windscreen conditions. The input training data is given in Appendix F, along with the Matlab code to build the network.

The performance of the trained network is given in Figure 7-16 and Figure 7-17. Figure 7-16 shows the performance of the slow path in the network and Figure 7-17 shows the performance of the fast path. It can be seen that the NN very accurately maps the input voltage and wiper current to the output velocity in both slow and fast states.

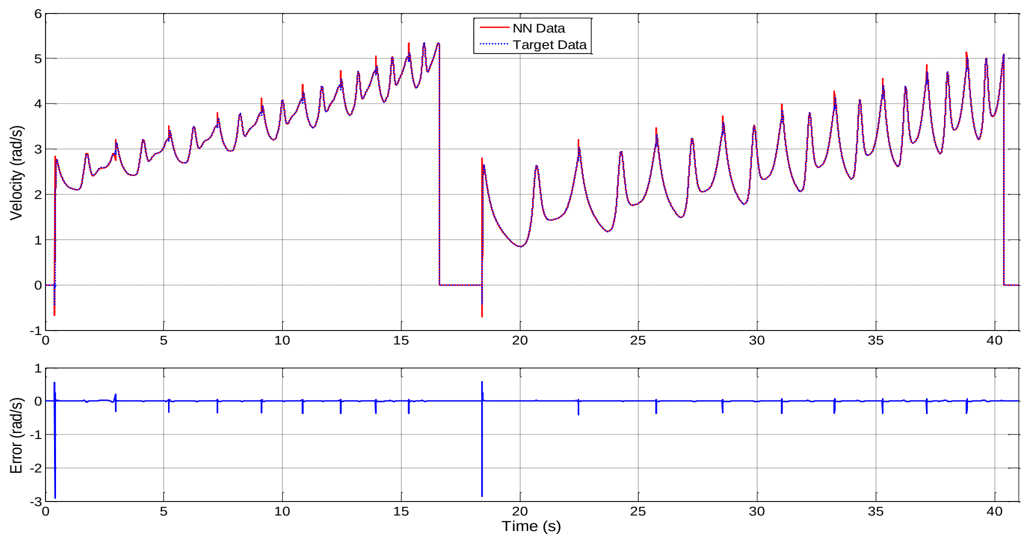


Figure 7-16: Sensorless NN Slow Mode Training Data

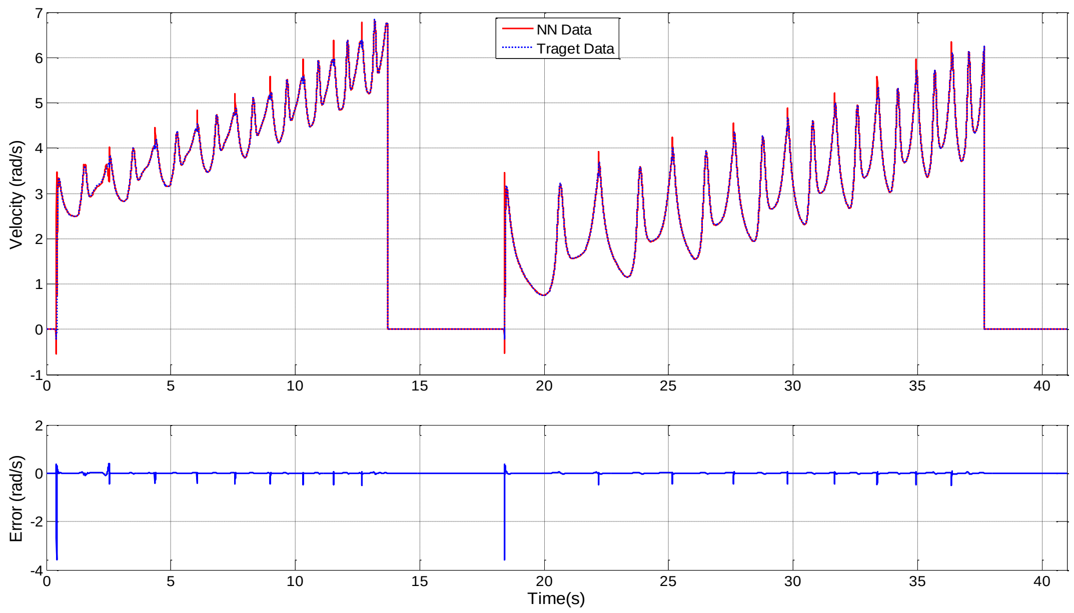


Figure 7-17: Sensorless NN Fast Mode Training Data

Clearly the NN described in this chapter is capable of estimating the velocity of the motor; however it is vulnerable to changes in system parameters and torque load conditions. It is well known that PMDC motor parameters can change during operation, largely due to temperature. A NN trained in this way is not able to adapt to changes in parameters and will output the velocity with an associated error. This problem could potentially be overcome using on-line parameter estimation techniques (such as those discussed in Chapter 2). This is left as further work, dependent on whether the vehicle manufacturer wishes to use the control system developed in this chapter.

7.6 Stateflow Control Implementation

Because the control systems developed in this chapter use continuous feedback to control the velocity of the wiper motor at a reference speed, the fast, slow and increment speed modes of the classic wiper modes are not used. Instead, the motor always operates in its slow mode unless the voltage reaches a certain threshold (18V in this case), at which point the motor switches to its fast mode to achieve a greater speed at the same voltage. The simple Stateflow control system to achieve this is shown in Figure 7-18.

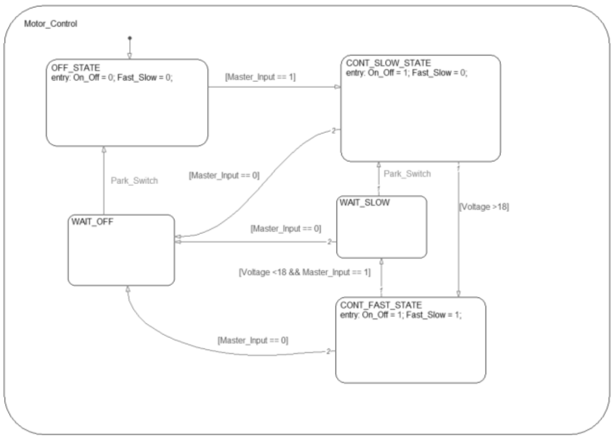


Figure 7-18: Stateflow Continuous Speed Motor Controller Implementation

7.7 Discussion

In this chapter, methods of directly controlling the angular velocity of the wiper motor have been investigated. The benefits of directly controlling the velocity of the motor are reduced wear on the brushes and commutator of the motor, and reduced energy consumption because the motor can run at the lowest speed needed to clear the windscreen. The control problem is challenging because the torque load applied to the motor is highly dynamic. A PID controller has been used to implement the direct velocity control in order to set a benchmark to assess the performance of adaptive controllers based on single neurons and fuzzy logic.

The parameters of the PID controller were tuned using Matlab. The controller successfully tracked the input reference voltage with zero overshoot. However the rise time was relatively high at 0.765 seconds and the disturbance rejection was poor. At the highest reference velocity (1.2π rad/s) and highest torque (dry windscreen condition) the largest deviation was 1.179 rad/s. The average steady state error across the entire test procedure was 0.459rad/s.

In order to overcome the drawbacks of PID control, adaptive controllers based on Single Neuron PID (SNPID) control have been implemented. The method replaces the classic PID controller with a single neuron with three weighted inputs and one output. The weights of the inputs can be tuned in real-time in response to the error in the system and are analogous to the PID gain parameters. Three learning rules were used to adapt the weights: Hebb, Error-Hebb and Quadratic. All methods showed a significant improvement over PID control, with lower steady state error (i.e. higher disturbance rejection) and lower rise times being observed. The Hebb and Error-Hebb learning rules showed the best performance, with the quadratic learning rule having good transient performance but weak disturbance rejection and stability. It was clear that the gain applied to the system, K, has a large effect on the performance. Therefore a system designed to tune K in real time was implemented.

To adapt the value of K in real-time a fuzzy logic controller has been designed. The input to the controller is the error between the reference and measured motor velocity and the output is the normalised value of K. The output is then fed through a gain block, the value of which depends on the learning rule used to adjust the neuron's weights. The results showed that the addition of the fuzzy controller improved the performance of the SNPID controllers by identifying an optimal compromise between transient and steady state performance. All three learning rules had an overshoot of around 0.55 rad/s and a rise time of less than 0.1s. The maximum steady state errors were: Hebb = 0.133 rad/s (0.064 rad/s average), Error-Hebb = 0.109 rad/s (0.052 rad/s average) and quadratic = 0.135 rad/s (0.065 rad/s average). This demonstrates greatly improved performance over the original PID control system. The SNPID incorporating the quadratic learning rule showed the greatest improvement with the addition of the fuzzy controller, with its results being comparable to the Hebb and Error-Hebb systems where previously it was inferior.

To implement the control system developed in this chapter, continuous measurement of the wiper motor velocity is needed. Currently, wiper systems do not do this and it is up to the vehicle manufacturer to determine whether the benefits of implementing these control methods outweigh the costs of adding a speed sensor. This issue could be overcome by using sensorless speed control techniques. A Feed Forward Neural Network has been designed and trained to map the motor voltage and current to the output velocity. The network produced very accurate results; however it is vulnerable to variable parameters in the wiper system which will cause an error in the output velocity. Online parameter identification could potentially overcome this issue; however this is left as further work should the vehicle manufacturer wish to implement the controllers.

Chapter 8 - Generic Modelling and Real-Time Implementation

8.1 Introduction

This chapter briefly demonstrates the modelling tools designed to be used by Jaguar Land Rover (JLR) in order to develop plant models of the windscreen wiper system to replace the real system in HIL simulation. The tools presented here implement the models, optimisation algorithms and model simplification strategies reported in previous chapters.

The chapter begins by showing the Simulink physical library of the separate wiper system elements. Then, the Graphical User Interface (GUI) used to easily design and update models is shown and described. Finally, details of the real-time simulation system used to generate experimental data for this project is given.

8.2 Simulink Library

The wiper system modelling library is shown in Figure 8-1. The physical models of the elements of the wiper system and the equations describing the friction and aerodynamic effects are contained in six separate sub libraries.

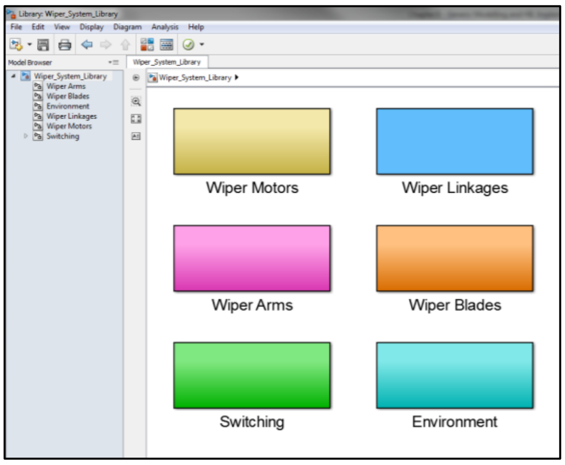
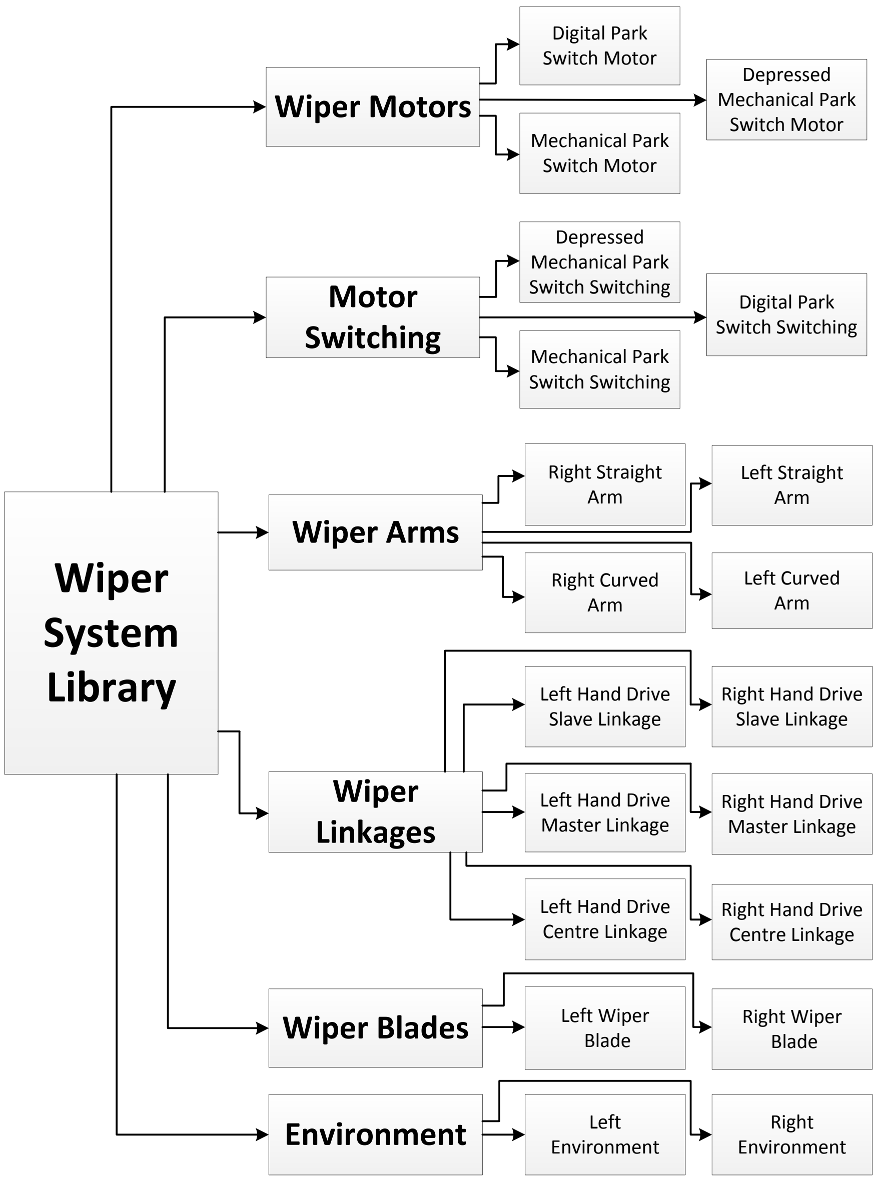


Figure 8-1: Wiper System Simulink Library

Figure 8-2 shows the structure of the library. The engineer is able to select the model elements needed for each sub system and connect them together in the Simulink modelling environment.



Wiper System Library
Wiper Motors
Digital Park Switch Motor
Depressed Mechanical Park Switch Motor
Mechanical Park Switch Motor
Motor Switching
Depressed Mechanical Park Switch Switching
Digital Park Switch Switching
Mechanical Park Switch Switching
Wiper Arms
Right Straight Arm
Left Straight Arm
Right Curved Arm
Left Curved Arm
Wiper Linkages
Left Hand Drive Slave Linkage
Right Hand Drive Slave Linkage
Left Hand Drive Master Linkage
Right Hand Drive Master Linkage
Left Hand Drive Centre Linkage
Right Hand Drive Centre Linkage
Wiper Blades
Left Wiper Blade
Right Wiper Blade
Environment
Left Environment
Right Environment

Figure 8-2: Wiper System Library Structure

8.3 Wiper System Generic Modelling Tool

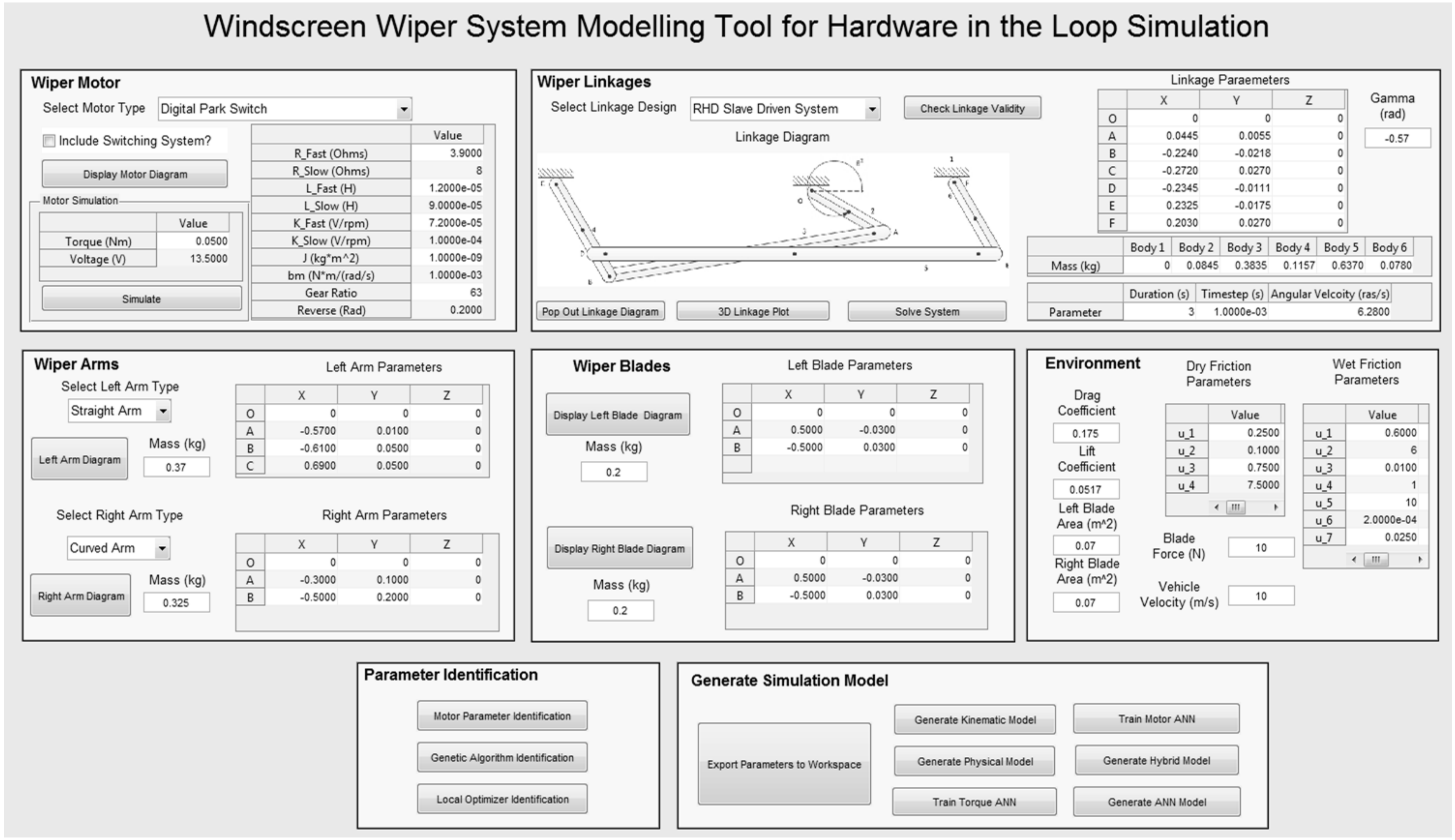
The generic modelling tool for the wiper system is shown in Figure 8-3. Its purpose is to allow the engineer to easily select and parametize the elements of the wiper system needed to build the model. The GUI is split into seven panels, each of which is now described.

8.3.1 Wiper Motor Panel

The operator can select between the three types of motor model: Digital Park Switch, Mechanical Park Switch and Depressed Mechanical Park Switch. There is a check box to allow the inclusion of the switching system if it is needed. The user can then input the motor parameters into the table and display a model of the motor using the "Display Motor Diagram" button. The motor can also be simulated under selected input voltages and torque loads to ensure that motor parameters are realistic. It is the state space motor model presented in Section 3.5.1 that is simulated and the system outputs graphs showing the output current and velocity of the motor in its fast and slow states.

8.3.2 Wiper Linkages Panel

The operator chooses between six linkage configurations: Master, Slave and Centre Driven, with the option of right hand or left hand drive for each. A diagram of the chosen linkage system is displayed, detailing the position of the design points. The user enters the design points and masses of the linkages in the tables, along with the system angle, γ, with respect to the vertical axis. A 3D plot of the linkages based on the parameters entered by the user can be plotted to visually check the validity of the parameters. Also, a button that checks the configuration of the linkages based on the equations shown in Section 4.1.1 can be pressed which outputs a message informing the user that the configuration is possible, or that it violates a constraint. The violated constraint is specified so changes to the parameters can be made. The system can be simulated in isolation based on the multibody dynamics model shown in Section 4.2.



Windscreen Wiper System Modelling Tool for Hardware in the Loop Simulation

Wiper Motor

Select Motor Type: Digital Park Switch

☐ Include Switching System?

Display Motor Diagram

Motor Simulation

	Value
Torque (Nm)	0.0500
Voltage (V)	13.5000

Simulate

	Value
R_Fast (Ohms)	3.9000
R_Slow (Ohms)	8
L_Fast (H)	1.2000e-05
L_Slow (H)	9.0000e-05
K_Fast (V/rpm)	7.2000e-05
K_Slow (V/rpm)	1.0000e-04
J (kg*m^2)	1.0000e-09
bm (N*m/(rad/s)	1.0000e-03
Gear Ratio	63
Reverse (Rad)	0.2000

Wiper Linkages

Select Linkage Design: RHD Slave Driven System

Check Linkage Validity

Linkage Diagram

Pop Out Linkage Diagram | 3D Linkage Plot | Solve System

Linkage Paraemeters

	X	Y	Z
O	0	0	0
A	0.0445	0.0055	0
B	-0.2240	-0.0218	0
C	-0.2720	0.0270	0
D	-0.2345	-0.0111	0
E	0.2325	-0.0175	0
F	0.2030	0.0270	0

Gamma (rad): -0.57

	Body 1	Body 2	Body 3	Body 4	Body 5	Body 6
Mass (kg)	0	0.0845	0.3835	0.1157	0.6370	0.0780

	Duration (s)	Timestep (s)	Angular Velcoity (ras/s)
Parameter	3	1.0000e-03	6.2800

Wiper Arms

Select Left Arm Type: Straight Arm

Left Arm Diagram

Mass (kg): 0.37

Left Arm Parameters

	X	Y	Z
O	0	0	0
A	-0.5700	0.0100	0
B	-0.6100	0.0500	0
C	0.6900	0.0500	0

Select Right Arm Type: Curved Arm

Right Arm Diagram

Mass (kg): 0.325

Right Arm Parameters

	X	Y	Z
O	0	0	0
A	-0.3000	0.1000	0
B	-0.5000	0.2000	0

Wiper Blades

Display Left Blade Diagram

Mass (kg): 0.2

Left Blade Parameters

	X	Y	Z
O	0	0	0
A	0.5000	-0.0300	0
B	-0.5000	0.0300	0

Display Right Blade Diagram

Mass (kg): 0.2

Right Blade Parameters

	X	Y	Z
O	0	0	0
A	0.5000	-0.0300	0
B	-0.5000	0.0300	0

Environment

Drag Coefficient: 0.175

Lift Coefficient: 0.0517

Left Blade Area (m^2): 0.07

Right Blade Area (m^2): 0.07

Dry Friction Parameters

	Value
u_1	0.2500
u_2	0.1000
u_3	0.7500
u_4	7.5000

Blade Force (N): 10

Vehicle Velocity (m/s): 10

Wet Friction Parameters

	Value
u_1	0.6000
u_2	6
u_3	0.0100
u_4	1
u_5	10
u_6	2.0000e-04
u_7	0.0250

Parameter Identification

Motor Parameter Identification

Genetic Algorithm Identification

Local Optimizer Identification

Generate Simulation Model

Export Parameters to Workspace

Generate Kinematic Model | Train Motor ANN

Generate Physical Model | Generate Hybrid Model

Train Torque ANN | Generate ANN Model

Figure 8-3: Wiper System Generic Modelling Tool

8.3.3 Wiper Arms Panel

In this panel the user can choose between the straight or curved arms for the left and right wiper arms. Diagrams of the arms can be viewed by pressing the “Left Arm Diagram” or “Right Arm Diagram” buttons. The design point parameters can be entered into the tables and the masses into the static text boxes.

8.3.4 Wiper Blade Panel

The parameters of the left and right wiper blade are input into the tables and the masses into the static text boxes. Diagrams of the blades can also be generated.

8.3.5 Environment Panel

The operator can input the dry and wet friction parameters into their respective tables. The drag and lift coefficients, left and right blade areas, the velocity of the vehicle and the force on the windscreen applied by the blade are entered into the static text boxes.

8.3.6 Parameter Identification Panel

Using this panel, the operator can access the parameter identification procedures reported in Chapter 5. The validation data needs to be in the Matlab workspace. The operator instigates the three algorithms by pressing the respective buttons. The system runs the algorithms and updates the parameters of the model accordingly. The performance of the optimised model is plotted and displayed.

8.3.7 Generate Simulation Model Panel

This panel allows the user to automatically generate the various simulation models described in the previous chapters. Firstly, the “Export Parameters to Workspace” button takes the parameters from the five modelling panels and exports them to the Matlab workspace. These can then be accessed by the Simulink/Matlab models. The “Generate

Kinematic Model” automatically builds the multibody dynamic model shown in Section 4.2 based on the model selections and parameters selected in the five modelling panels.

The “Generate Physical Model” button automatically generates a Simulink model of the wiper system configuration selected in the GUI. An example of a model generated like this is shown in Figure 8-4 and the Matlab code to achieve this is shown in Appendix G. The process utilises the library presented in section 8.2.

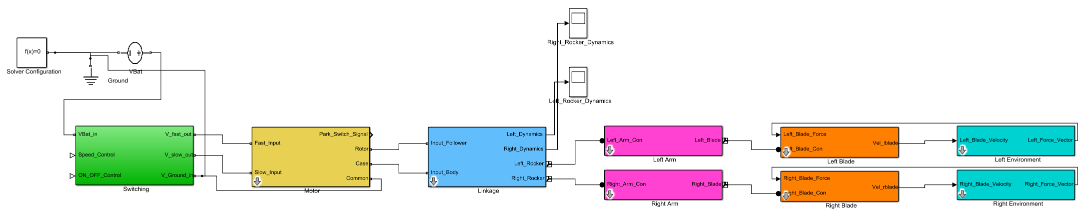


Figure 8-4: Simulink Wiper Model Generate Automatically

The “Train Torque ANN” button trains the feed forward NN model shown in Section 6.4.2 using training data in the Matlab workspace. Likewise the “Train Motor ANN” trains the recursive NN shown in Section 6.4.3 using training data from the Matlab workspace.

The “Generate Hybrid Model” button uses the Torque ANN and information from the Wiper Motor Panel to generate the hybrid physical-ANN model shown in 6.2.1. Similarly, the “Generate ANN Model” button uses the Torque and Motor ANNs to generate the ANN model of the wiper system shown in Section 6.3.1.

8.4 dSPACE Real-Time Simulator Implementation

A dSPACE Ecoline HIL simulator was used to generate data used to verify the models developed in this project and to implement the real-time plant models to prove their suitability for HIL simulation. Figure 8-5 shows the structure of the real time simulator used, configured to test an ECU using the wiper system plant model.

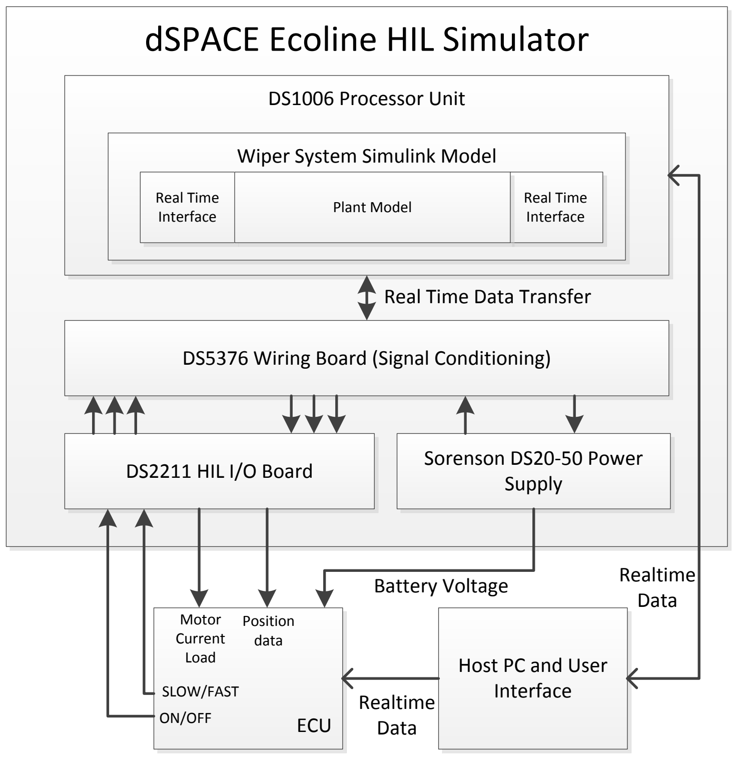


Figure 8-5: Real Time Simulator Schematic

The real-time simulator interfaces with the hardware using the DS2211 I/O board. The board has a range of ADC inputs and DAC outputs for signal measurement and control outputs. The I/O board also incorporates digital inputs and outputs which were used to measure the park switch and control the motor drive relays, respectively. The simulator also includes an EV1025 module for high current measurement. This was used to measure the motor current directly. The simulator incorporates a Sorensen DS20-50E switched mode power supply which can be controlled using the Host PC. The power supply acts as the vehicle battery during simulation.

When generating data and validating wiper models, the system is configured as in Figure 8-5 but with the wiper system as the hardware and elements of the ECU simulated in the dSPACE simulator[19]. The controller used in this configuration is designed in Simulink

[19] This configuration is sometimes known as Rapid Control Prototyping

and then converted to executable C code using the Real-Time Workshop tool in Matlab and uploaded to the dSPACE simulator. The Simulink model is shown in Figure 8-6.

The inputs from the hardware are represented using dSPACE input blocks shown on the left of the diagram. These inputs are processed using standard Simulink blocks and then outputted to the hardware using dSPACE output blocks, or displayed in ControlDesk using Simulink scopes. The inputs from the wiper are conditioned using gains, filters and switches to convert raw measured data to useable data. The voltage of the battery (i.e. the Sorensen power supply) is controlled with the configuration in the top right of Figure 8-6. The voltage can be switched on and off and its value can be set in real time via ControlDesk. The voltage can also be varied using predesigned signals such as sine waves or a repeating test sequence. The wiper motor is controlled using the Stateflow chart in the bottom right corner of Figure 8-6. The chart itself is shown in Figure 8-7.

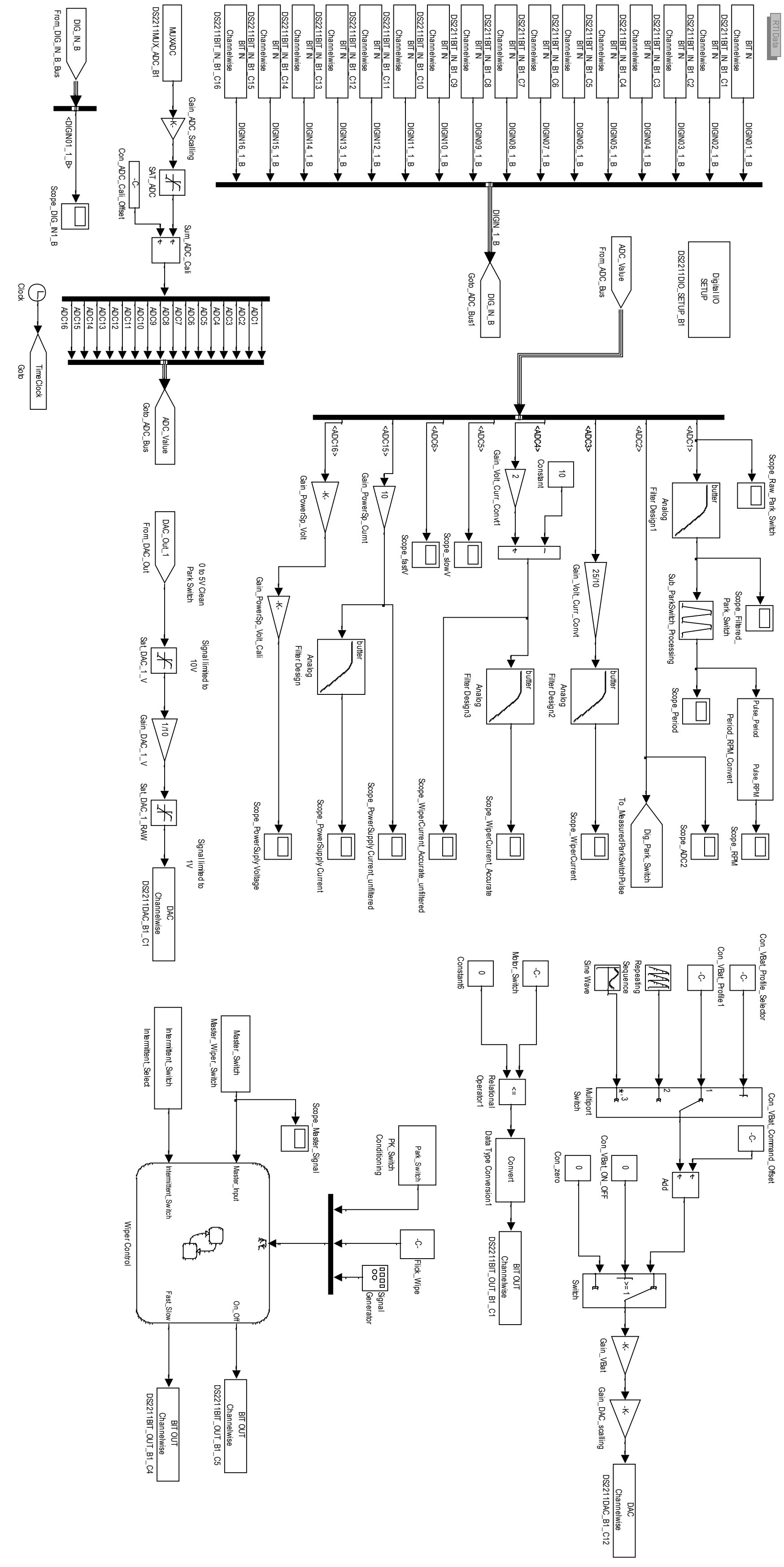


Figure 8-6: Real-Time Simulator Simulink Model

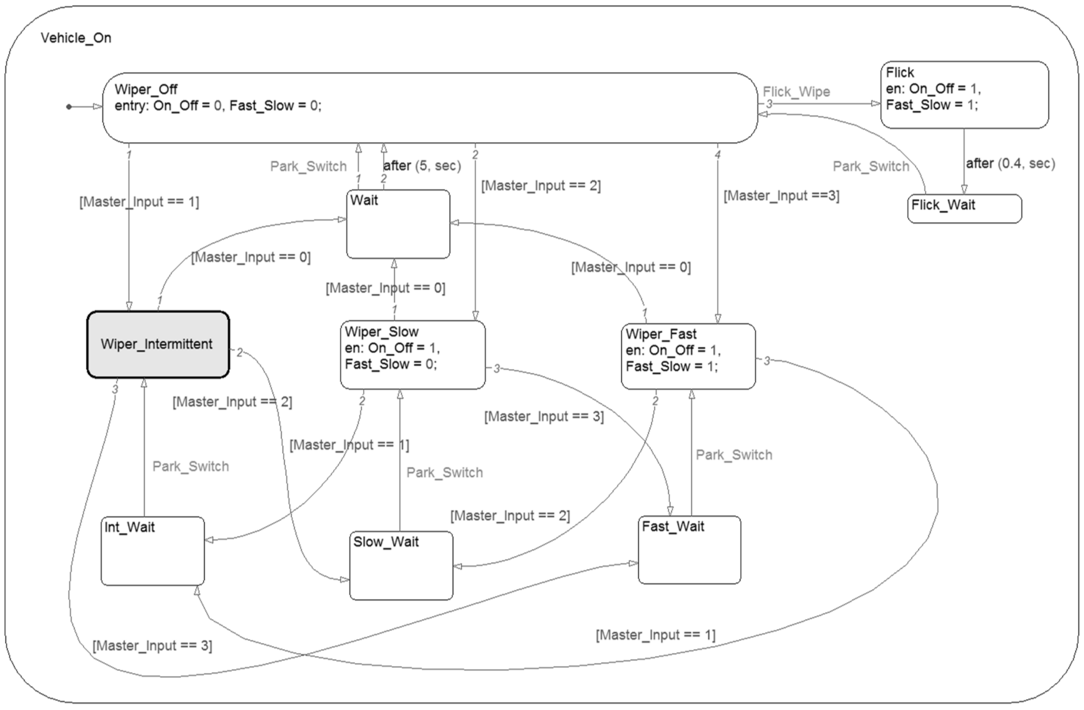


Figure 8-7: Wiper Controller Stateflow Chart

The system always begins in the Wiper_Off state, in which the motor input is connected to ground. The system will leave the Wiper_Off state under four conditions. The first is if the Flick_Wipe signal is received. In this case the system enters the Flick state in which the motor operates in its fast mode unitl the Park_Switch signal is received, upon which it returns to the Wiper_Off state. If the Master_Input equals 1, 2 or 3 the sytem enters the Wiper_Intermittent, Wiper_Slow or Wiper_Fast states, respectively. The system remains in one of these states until the value of Master_Input changes. If Master_Input equals zero, the system enters the Wait state until the Park_Switch is received, after which the Wiper_Off state is entered and the motor switches off. The three “motor-on” states can be switched between by changing the value of Master_Input. If the value changes, the system enters either the Int_Wait, Slow_Wait or Fast_Wait state until the Park_Switch is received. At which point it enters its new state and the motor changes its speed. The Wiper_Intermittent state sub-chart is shown in Figure 8-8.

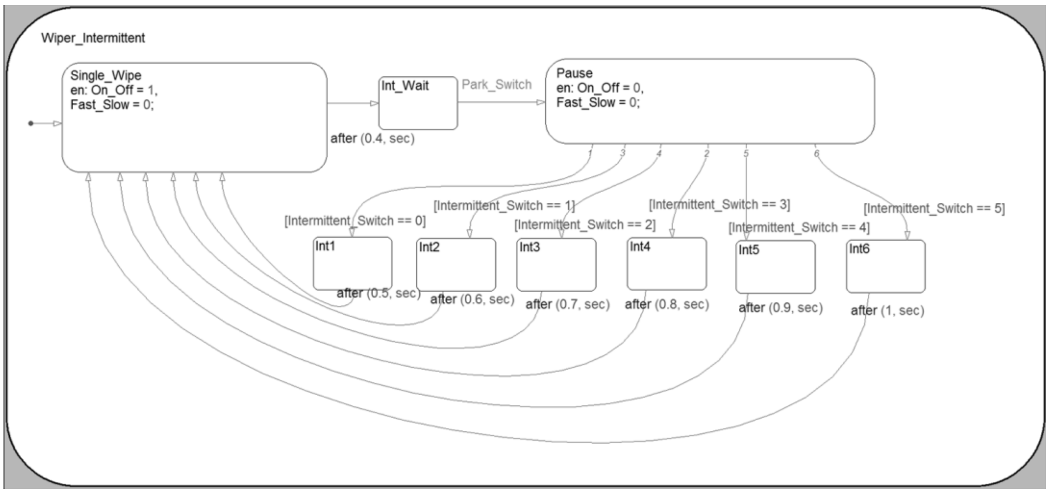


Figure 8-8: Wiper Intermittent Stateflow Sub-Chart

Upon entry to the Wiper_Intermittent state, the Single_Wipe state is entered which operates the wiper motor in its slow mode until the Park_Switch is received. At this point the Pause state is entered which switches the motor off. Immediatley, one of six states is entered, depending on the value of the Intermittent_Switch input. The system remains in this state for a specific amount of time to achieve a pause between each wipe. After the allotted time has expired, the system enters the Single_Wipe state again.

The Stateflow models shown in Figure 8-7 and Figure 8-8 are designed to imitate the operation of an ECU controlling the wiper motor for experimental purposes only and are not necessarily the same as the actual ECU design.

The ControlDesk system used to monitor and control the wiper system in real-time is shown in Figure 8-9. Measured outputs such as the wiper current and park switch can be seen in the plots on the right of the diagram. The main power can be switched on and off and its values set using the controls on the bottom left of the diagram. The motor itself is controlled using the buttons in the “Manual Control” panel, which also shows numerical outputs of the motor. The data logger on the top left of the diagram collects data which can be imported into Matlab for model optimization and verification.

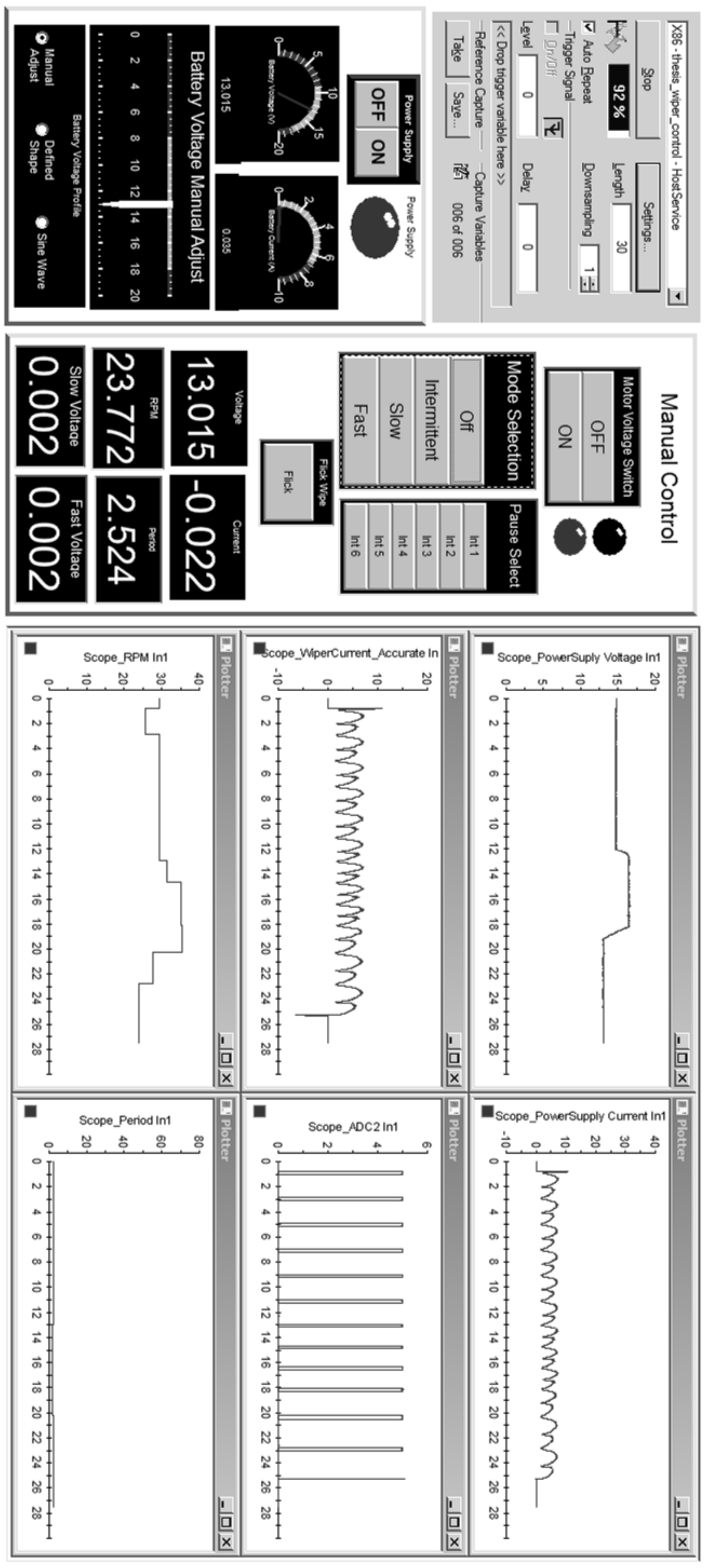


Figure 8-9: Real-Time Simulator ControlDesk System

A photograph of the dSPACE simulator and the wiper test rig used in this project is shown in Figure 8-10. The dSPACE simulator supplies power to the motor using the relay circuit shown in Figure 7-1 with digital outputs controlling the relays. The wiper current and the park switch are directly measured by the simulator. The test rig is waterproof, allowing the wipers to be operated in both dry and wet modes. To wet the windscreen, water is pumped from a bucket to a sprinkler which continuously wets the windscreen. The system can operate in wet mode for around three minutes before the water must be replaced.

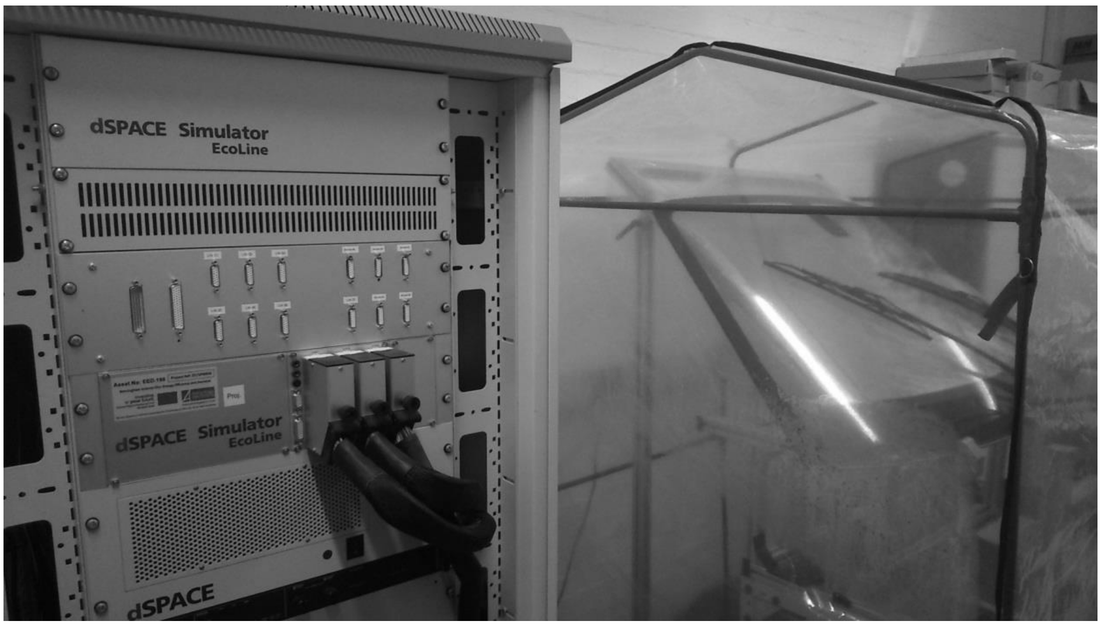


Figure 8-10: dSPACE Simulator and Wiper Test Rig

8.5 Discussion

This chapter has briefly detailed the generic modelling and real-time simulation aspects of the project. The generic modelling elements are designed to make the modelling of the wiper system as easy and as flexible as possible. The physical models of the separate elements of the wiper system are organised into a Simulink library. The wiper system can easily be built up by dragging the relevant elements of the system into the Simulink workspace and connecting them together. All of the modules are connected together as they are in the real system and the complexities of the models are hidden from the operator.

A modelling tool, presented in a Graphical User Interface (GUI), has been developed to design and parameterize the wiper models. The elements of the wiper models presented in Chapter 3 and Chapter 4 are selected and their parameters manually entered into the GUI. The parameter identification methods shown in Chapter 5 can be accessed from the GUI. By pressing the relevant buttons the operator can run the algorithms using data from the Matlab workspace and the parameters of the model are updated accordingly. Finally, the multibody kinematics model, full physical Simulink model, hybrid physical-ANN model and the full ANN model can be automatically generated from the GUI. The system uses model and parameter information from the GUI to train the ANNs (if needed) and build the models in Simulink or the Matlab Workspace.

The real-time simulation system implemented using a dSPACE simulator has been presented. The control and measurement system is designed in Simulink using dSPACE interface blocks and standard Simulink blocks and Stateflow charts. The wiper system can be controlled and monitored in real time using the ControlDesk software from dSPACE.

Chapter 9 - Conclusions and Further Work

9.1 Conclusions

The automotive industry has seen steady growth in recent years and passenger car sales have exceeded 60 million units per year worldwide. In parallel with this growth, the amount of electronics in a modern vehicle has also increased significantly. Electronic components are gradually replacing mechanical components and adding additional utilities to vehicles. Inevitably the increase in electronic components and systems has led to an increase in the need for Electronic Control Units (ECU); a modern luxury vehicle can include in the order of 100 ECUs running 100 million lines of code. The sheer number and complexity of ECUs means that thorough testing must be carried out throughout the development process in order for the vehicle manufacturer to produce a quality product in a relatively short time frame. Advanced simulation techniques are used to achieve this, including Hardware in the Loop (HIL) simulation. In HIL simulation, a real ECU is connected to real-time simulation models of its environment, such as plant models of its loads and artificially generated control signals. In this project a plant model of the vehicle's windscreen wiper system has been designed and made suitable for real-time simulation.

The windscreen wiper model is separated into four parts: The wiper motor which drives the system, the linkages, the arms and blades, and the windscreen interface. The model is designed using a modular strategy so that versions of each of the four separate elements can be connected together and simulated as one model. The elements of the model are incorporated into a Simulink library and a modelling tool presented as a Graphical User Interface (GUI) so that modelling new wiper systems or updating old systems can be done quickly and easily.

A wiper motor is a unique design of PMDC motor which has two electrical inputs and one electrical output (electrical ground). The two inputs are used for speed control of the motor because in the vehicle the motor is connected directly to the battery so varying the voltage is not possible. The input brush in line the magnetic neutral line is the motor's slow input and the input brush offset from the magnetic neutral line is its fast input. Operating the motor in its fast mode causes an imbalance in the current paths and this is captured by specifying different electrical parameters for the fast and slow modes. In the fast mode the inductance and resistance of the armature winding, and the motor constants, are lower than in the slow mode. This causes the motor to rotate faster and draw a higher current. Using this knowledge, eight parameter state space and physical models of the wiper motor have been designed that successfully simulate the behaviour of the system in real-time.

The mechanical elements of the system, i.e. the linkages, arms and blades and the windscreen interface have been modelled using the physical modelling tool SimMechanics. Each component is modelled using Body blocks interconnected with Joint or Weld blocks. The separate elements are modelled individually and can be connected to one another using physical ports. The friction between the wiper blades and the windscreen has been mathematically modelled using a four parameter equation for friction for when the windscreen is dry and a seven parameter equation for when it is wet. The equations derived are based on data presented in the literature. The aerodynamic forces acting on the wiper blades due to the motion of the vehicle has also been modelled based on the drag and lift coefficients of the wiper blades. The whole system (mechanical structure, friction, aerodynamics) successfully models the mechanical element of the wiper system.

The unknown motor, friction and force parameters of the wiper model need to be identified using experimental data and optimisation techniques. A three stage parameter identification process is proposed starting with a transfer function method, followed by a Genetic Algorithm (GA) and ending with a local optimiser. A transfer function of the wiper motor has been derived relating the motor current to the input voltage. The model assumed a constant torque load and that the damping coefficient was negligible. Using experimental data, the system identification tool in Matlab was used to identify the motor parameters. The system successfully identified the parameters, which are then used to reduce the search space of the GA. A bespoke GA was designed and used to refine the motor parameters and identify the remaining friction and force parameters. The GA successfully identified all seventeen model parameters in 135 generations to a reasonable accuracy. These parameters were then used as the initial guess of a nonlinear least squares local optimiser in order to further improve the accuracy of the model.

The model described above is not suitable for HIL simulation due to the complexity of the SimMechanics element. The model has been simplified using Artificial Neural Network (ANN) and look-up table methods. The look-up tables are simply populated with positional data from the model and driven by the position of the motor output. Two ANNs were designed. The first is a feed forward ANN whose inputs are the position and speed of the motor output and whose outputs are the torque loads applied to the motor in the windscreen's dry and wet condition. The Mean Squared Error (MSE) of the torque ANN is 0.0951, demonstrating a very high accuracy. The second ANN is a recursive network that models the wiper motor. Its inputs are the torque from the torque ANN and the battery voltage and its outputs are the current and velocity of the motor in its fast and slow mode. In its closed loop configuration, the motor ANN has a MSE of 0.0068. The torque ANN is used to replace the mechanical element of the model to generate a hybrid physical-ANN model of the wiper system which as was measured to simulate 60s (simulation time) of data in 5.92s (real time). Both ANNs are used to replace the motor and mechanical elements of the system to create a full ANN implementation which was measured to simulate 60s (simulation time) of data in 2.17s (real time).

A control method based on Single Neuron PID (SNPID) control has been proposed in order to operate the motor at a constant speed. This means that the motor can operate at the lowest speed necessary to clear the windscreen, reducing power consumption and wear on the brushes. The SNPID controller replaces a normal PID controller with a single neuron with three weighted inputs and one output. The weights can be modified in real time based on the error between the measured velocity and the reference velocity. The output of the neuron is the control signal controlling the voltage of the wiper motor which is multiplied by a gain. The value of the gain has a large effect on the performance of the system and is adapted in real-time using a fuzzy controller. The complete control systems had average steady state errors in the range of 0.052 rad/s to 0.065 rad/s, depending on the learning rule used. Rise times of 0.1s and overshoots of 0.55 rad/s were observed. The control system requires velocity measurement and voltage control methods to be added to the wiper system in order to work.

The models, optimisation and simplification methods have been implemented in a GUI to be used by Jaguar LandRover (JLR). The purpose of the GUI is to make the whole modelling process simple for the vehicle manufacturer. This means that the time taken to develop or update models for ECU development is reduced and more efficient and thorough testing of the ECUs can take place.

9.2 Suggested Further Work

There are some elements to this project that would benefit from further research. The first is the aerodynamic force analysis of the wiper system. The model presented in this thesis is very simple and it was found in the literature review that the actual behaviour of the aerodynamic forces is very complex. The fidelity of the wiper model could be improved by using Computational Fluid Dynamics (CFD) simulations and wind tunnel experiments to comprehensively study the behaviour of the wiper system over the full range of vehicle speed, windscreen and bonnet design, and wiper design. This information could then be modelled dynamically to replace the simple model presented in this project. This analysis is very complicated and requires a strong knowledge of CFD and as such was not feasible for this project. This analysis would be a strong addition to this project and to the wider literature in this area.

It would be useful to derive the full dynamic equations of the mechanical wiper system, rather than relying on physical modelling tools. This would mean that a greater number of model simplification methods would be available to convert the model to a HIL suitable one. Although ANNs have been shown to be a powerful tool in converting the physical model to a real-time capable one, they have some weaknesses such as failing when subject to inputs outside of their training data and not being suitable for fault insertion testing. They are also black box systems, meaning that they offer no insight into the operation of the system beyond the I/O relationships.

The control system developed in Chapter 7 relies on continuous speed measurements from the motor. Ideally, this will be achieved without the addition of a speed sensor, meaning that only the voltage, current and park switch can be used. There are methods of estimating the velocity of the motor using estimators such as Kalman filters and other model based control schemes. For model based methods of estimating the velocity to be effective, the parameters of the model will have to be estimated in real-time because they are subject to change. The difficulty in using standard methods of state and parameter estimation in this project is the fact that the load torque is dynamic, not measured and is subject to changing parameters. An investigation into whether the information contained in the motor current and average speed derived from the park switch could achieve sensorless control would be very useful for this project and motor control in general.

Finally, the if the models and control systems developed in this project are to be used in the automotive industry, they must be modified to conform to guidelines such as The MathWorks Automotive Advisory Board (MAAB) which gives rules on subjects such as signal and subsystem labelling.

References

[1] Clearwater Global Finance, “The Global Automotive Report,” 2013.

[2] OICA, [Online]. Available: http://www.oica.net/category/sales-statistics/. [Accessed February 2014].

[3] OICA, [Online]. Available: http://www.oica.net/category/production-statistics/. [Accessed February 2014].

[4] S. Jeon, J. Cho, Y. Jung, S. Park and T. Han, “Automotive hardware development according to ISO 26262,” in *13th International Conference on Advanced Communication Technology (ICACT)*, Seoul, 2011.

[5] D. Frede, M. Khodabakhshian and D. Malmquist, “A state-of-the-art survey on vehicular mechatronics focusing on by-wire systems,” KTH Royal Institute of Technology, Stockholm , 2010.

[6] V. Von Tils, “Trends and Challenges in Automotive Electronics,” in *IEEE International Symposium on Power Semiconductor Devices and IC's*, Naples, 2006.

[7] Robert Bosch GmbH, Automotive Electrics Automotive Electronics, Sussex: John Wily and Sons Ltd, 2007.

[8] P. Waltermann, “Hardware-in-the-Loop: The technology for testing electronic controls in automotive engineering,” in *Translation of 6th Paderborn Workshop "Designing Mechatronic Systems"*, Paderborn, 2009.

[9] R. N. Charette, “This Car Runs on Code,” February 2009. [Online]. Available: http://spectrum.ieee.org/transportation/systems/this-car-runs-on-code.

[10] J. Mossinger, “Software in Automotive Systems,” *Software, IEEE ,* vol. 27, no. 2, pp. 92-94, 2010.

[11] J. Scharf, R. Hopler and J. Hillyard, “Automotive Real-Time Automotive Real-Time: Modeling and Applications,” in *Real Time Simulation Technologies*, Florida, Taylor & Francis Group, 2013.

[12] H. Schuette and M. Ploeger, “Hardware-in-the-Loop Testing of Engine Control Units - A Technical Survey,” *SAE Technical Paper,* 2007.

[13] B. Ganesh, “Hardware in the Loop Simulation (HIL) for Vehicle Electronics Systems Testing and Validation,” *SAE Technical Paper 2005-26-304,* 2005.

[14] I. R. Kendall and R. P. Jones, “An investigation into the use of hardware-in-the-loop simulation testing for automotive electronic control systems,” *Control Engineering Practice,* pp. 1343 - 1356, 1999.

[15] H. Hanselmann, “Hardware-in-the Loop Simulation as a Standard Approach for Development, Customization, and Production Test of ECU's,” *SAE Technical Paper 931953,* 1993.

[16] I. R. Kendall, R. P. Jones and S. M. Thomas, “Simulation as a means of achieving “the impossible”: an investigation into the use of simulation in the development of electronic control systems at Jaguar cars,” in *Simulation '98. International Conference on*, York, 1998.

[17] P. D. Edwards, “The use of statecharts in developing body electronics software [for automobiles],” in *Integrity of Automotive Electronic Systems, IEE Colloquium on*, London, 1993.

[18] H. K. Fathy, Z. S. Filipi, J. Hagena and J. L. Stein, “Review of hardware-in-the-loop simulation and its prospects in the automotive area,” in *Proceedings of SPIE*, 2006.

[19] H. Krisp, K. Lamberg and R. Leinfellner, “Automated Real-Time Testing of Electronic Control Units,” in *SAE Paper, In-Vehicle Software & Hardware Systems*, Detroit, 2007.

[20] V. Jaikamal, “Model-based ECU development – An Integrated MiL-SiL-HiL Approach,” *SAE Technical Paper 2009-01-0153,* 2015.

[21] D. Fleischer, M. Beine and U. Eisemann, “Applying Model-Based Design and Automatic Production Code Generation to Safety-Critical System Development,” *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.,* vol. 2, no. 1, pp. 240-248, 2009.

[22] C. Mu, X. Ma, H. Ma, X. Huang, L. Zhang and R. Fan, “Current Issues and Future Trends in Analysis of Automotive Functional Safety,” in *Communication Systems and Information Technology*, Springer Berlin Heidelberg, 2011, pp. 171-178.

[23] M. Beine, “Model-Based Software Development According to ISO 26262,” *Elektronik automotive,* 2009.

[24] K. Lamberg, “Model-Based Testing of Automotive Electronics,” in *Proceedings of the Design Automation & Test in Europe Conference*, Munich, 2006.

[25] A. Filgerdamm, D. M. Plöger and T. Schulte, “HIL Simulation for Mechatronic

Automotive Electronic Control Units: Current applications in vehicle dynamics and electric power steering," *Elektronik automotive,* 2009.

[26] A. Wagener, T. Schulte, P. Waeltermann and H. Schuette, "Hardware-in-the-Loop Test Systems for Electric Motors in Advanced Powertrain Applications," *SAE Technical Paper 2007-01-0498,* 2007.

[27] A. Dhaliwal, S. C. Nagaraj and S. Ali, "Hardware-in-the-loop simulation for hybrid electric vehicles-an overview, lessons learnt and solutions implemented," *SAE Technical Paper 2009-01-0735,* 2009.

[28] T. Schulte and M. Plöger, "Electric Motors : Hardware-in-the- Loop Testing at Full Power," *Automobil Elektronik,* 2010.

[29] S. Okura, T. Sekiguchi and T. Oya, "Dynamic Analysis of Blade Reversal Behavior in a Windshield Wiper System," *SAE Technical Paper 2000-01-0127,* 2000.

[30] S. Goto, H. Takahashi and T. Oya, "Investigation of Wiper Blade Squeal Noise Reduction Measures," *SAE Technical Paper 2001-01-1410,* 2001.

[31] S. Goto, H. Takahashi and T. Oya, "Clarification of the mechanism of wiper blade rubber squeal noise generation," *JSAE review,* vol. 22, pp. 57-62, 2001.

[32] R. Grenouillat and C. Leblanc, "Simulation of Chatter Vibrations for Wiper Systems," *SAE Technical Paper 2002-01-1239,* 2002.

[33] I. M. Awanga, A. R. AbuBakar, B. A. Ghani, R. A. Rahman and M. Z. Zain, "Complex eigenvalue analysis of windscreen wiper chatter noise and its suppression through structural modifications," *International Journal of Vehicle Structures & Systems,* vol.

1, 2009.

[34] A. Ahmad, M. Z. Zain, A. R. Abu-Baker, B. Abd-Ghani, R. Abd-Rahman and A. As'arry, “Application of Input Shaping Control Strategy for Reducing Chatter Noise in the Automotive Wiper System,” in *Information Technology, 2008. ITSim 2008. International Symposium on*, Kuala Lumpur, 2008.

[35] M. A. Salim, A. Noordin, M. Z. Zain and A. R. A. Bakar, “The Analysis of Friction Effect in Automotive Wiper System Using Input Shaping Technique,” in *Proceedings of the World Congress on Engineering 2010*, London, 2010.

[36] A. Zolfagharian, M. Z. Zain, A. R. Abubakar and M. Hussein, “Particle Swarm Optimization Approach on Flexible Structure at Wiper Blade System,” *Proceedings of World Academy of Science Engineering and Technology ,* vol. 18, pp. 97-102, 2011.

[37] A. Zolfagharian, A. Noshadi, M. Z. Zain and A. R. Bakar, “Practical multi-objective controller for preventing noise and vibration in an automobile wiper system,” *Swarm and Evolutionary Computation,* vol. 8, pp. 54-68, 2013.

[38] J. Rouzic, A. Bot, J. Perret-Liaudet, M. Guibert, A. Rusanov, L. Douminge and F. Bretagnol, “Friction-Induced Vibration by Stribeck’s Law: Application to Wiper Blade Squeal Noise,” *Tribology Letters,* vol. 49, no. 3, pp. 563-572, 2103.

[39] M. Huang, “Analysis of Friction Induced Stability, Bifurcation, Chaos, Stick-slip Vibration and their Impacts on Wiping Effect of Automotive Wiper System,” *SAE Technical Paper 2014-01-0021,* 2014.

[40] L. Zhang, “Experimental Investigation into Friction Induced Noise of Automotive

Wiper System," *SAE Technical Paper 2001-01-0749,* 2010.

[41] S. Change and H. Lin, "Chaos attitude motion and chaos control in an automotive wiper system," *International Journal of Solids and Structures,* vol. 41, no. 13, pp. 3491-3504, 2004.

[42] S. Chang, "Application of Synchronization and Continuous Control to a Chaotic Automotive Wiper System," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering,* vol. 220, no. 8, pp. 1119-1130, 2006.

[43] H. Jian, G. Li, D. Chu and J. Xu, "Research on Passenger Car Windscreen Wiper Controller and Control Method Based on CAN," in *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, Changchun, 2009.

[44] R. D. Pawar and S. K. Shah, "A Review : Front Windshield Wiper Controller System for Synchronisation of Two Wiper Motors," in *2015 International Conference on Pervasive Computing (ICPC)*, Pune, 2015.

[45] L. Jean, "On the Synchronization of a Pair of Independent Windshield Wipers," *Control Systems Technology, IEEE Transactions on,* vol. 12, no. 5, pp. 787 - 795, 2003.

[46] F. Elahi and M. S. Rahman, "Intelligent Windshield for Automotive Vehicles," in *17th International Conference on Computer and Information Technology*, Dhaka, 2014.

[47] C. Alexandru and C. Pozna, "Dynamic modeling and control of the windshield wiper mechanisms," *WSEAS Transactions on Systems,* vol. 8, no. 7, pp. 825-834, 2009.

[48] Z. Xiaoyu, X. Yanfeng and L. Yengjie, "Based on Matllab Electrically Operated Windshield Wiper Systems Design Method Research," in *Third International*

Conference on Measuring Technology and Mechatronics Automation, Shanghai, 2011.

[49] H. Y. Chang, “Windshield Wiper System Design Integration,” *SAE Technical Paper 2011-01-0239,* 2011.

[50] J. Wei, A. Mouzakitis, J. Wang and S. Hao, “Vehicle windscreen wiper mathematical model development and optimisation for model based hardware-in-the-loop simulation and control,” in *Automation and Computing (ICAC), 2011 17th International Conference on*, Huddersfield, 2011.

[51] A. Mouzakitis, J. Wei and J. Wang, “Accurate Model Based Hardware-in-the-Loop Test for a Windscreen Wiper System,” *SAE Technical Paper 2012-01-1164,* 2012.

[52] H. H. John, Adaptation in natural and artificial Systems, The University of Michigan Press, 1975.

[53] G. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, 1989.

[54] S. N. Sivanandam and S. N. Deepa, Introduction to Genetic Algorithms, Berlin: Springer, 2008.

[55] M. Lankarany and A. Rezazade, “Parameter Estimation Optimization Based on Genetic Algorithm Applied to DC Motor,” in *2007 International Conference on Electrical Engineering*, Lahore, 2007.

[56] A. C. Megherbi, H. Megherbi, K. Benmahamed., A. G. Aissaoui and A. Tahour, “Parameter Identification of Induction Motors using Variable-weighted Cost Function of Genetic Algorithms,” *Journal of Electrical Engineering and Technology,* vol. 5, no. 4,

pp. 597-605, 2010.

[57] O. Mohamed, J. Wang, S. Guo, J. Wei, B. Al-Duri, J. Lv and Q. Gao, "Mathematical Modelling for Coal Fired Supercritical Power Plants and Model Parameter Identification Using Genetic Algorithms," *Electrical Engineering and Applied Computing,* vol. 90, pp. pp 1-13, 2011.

[58] S. Praesomboon, S. Athaphaisal, S. Yimman, R. Boontawan and K. Dejhan, "Sensorless speed control of DC servo motor using Kalman filter," in *Information, Communications and Signal Processing, 2009. ICICS 2009. 7th International Conference on* , Macau, 2009.

[59] M. F. Moussa, M. Saad and Y. G. Dessouky, "Adaptive control and one-line identification of sensorless Permanent Magnet DC motor," in *Computational Technologies in Electrical and Electronics Engineering (SIBIRCON), 2010 IEEE Region 8 International Conference on*, Listvyanka, 2010.

[60] S. Ichikawa, M. Tomita and S. Doki, "Sensorless Control of Permanent-Magnet Synchronous Motors Using Online Parameter," *Industrial Electronics, IEEE Transactions on,* vol. 53, no. 2, pp. 363-372, 2006.

[61] S. Morimoto, S. Sanada and Y. Takeda, "Mechanical Sensorless Drives of IPMSM With Online Parameter Identification," *IEEE Transactions on Industry Applications,* vol. 42, no. 5, pp. 1241-1248, 2006.

[62] P. R. Dahl, "A Solid Friction Model," EROSPACE CORP EL SEGUNDO CA, 1968.

[63] B. Armstrong-Hélouvry, P. Dupont and C. C. De Wit, "A survey of models, analysis

tools and compensation methods for the control of machines with friction," *Automatica,* vol. 30, no. 7, pp. 1083-1138, 1994.

[64] H. Olsson, K. J. Åström, C. Canudas de Wit, M. Gäfvert and P. Lischinsky, "Friction Models and Friction Compensation," *European Journal of Control,* vol. 4, no. 3, p. 176–195, 1998.

[65] C. Canudas De Wit, H. Olsson and P. Lischinsky, "A New Model for Control of Systems with Friction," *IEEE Transaction of Automatic Control,* vol. 40, no. 3, pp. 419-425, 1995.

[66] D. A. Haessig and B. Friedland, "On the Modeling and Simulation of Friction," *Journal of Dynamic Systems, Measurement, and Control,* vol. 113, no. 3, p. 354, 1991.

[67] B. Borsotto, E. Godoy, D. Beauvois and E. Devaud, "An identification method for static and dynamic friction coefficients," in *International Conference on Control, Automation and Systems*, Seoul, 2007.

[68] Y. Fujii, "Method for measuring transient friction coefficients for rubber wiper blades on glass surface," *Tribology International,* vol. 41, no. 1, pp. 17-23, 2008.

[69] K. Nakajima and J. Nakajima, "Identification method of nonlinear systems with friction based on Genetic Algorithm," in *3rd International Symposium on Knowledge Acquisition and Modeling*, Wuhan, 2010.

[70] J. Kim, H. Chae, J. Jeon and S. Lee, "Identification and Control of Systems with Friction Using Accelerated Evolutionary Programming," *IEEE Control Systems,* vol. 16, no. 4, pp. 38-47, 1996.

[71] G. Bódai and T. Goda, “Friction Force Measurement at Windscreen Wiper/Glass Contact,” *Tribology Letters,* vol. 45, no. 3, pp. 515-523, 2012.

[72] G. Bódai and T. Goda, “Sliding friction of wiper blade: Measurement, FE modeling and mixed friction simulation,” *Tribology International,* vol. 70, no. 0, pp. 63-74, 2014.

[73] A. Buta, “Study on the friction between the wiper blade and the windshield,” *Bulletin of the Transilvania University of Braşov Series I: Engineering Sciences,* vol. 7, no. 1, 2014.

[74] F. Deleau, D. Mazuyer and A. Koenen, “Sliding friction at elastomer/glass contact: Influence of the wetting conditions and instability analysis,” *Tribology International,* vol. 42, no. 1, pp. 149-159, 2009.

[75] A. Koenen and A. Sanon, “Tribological and vibroacoustic behavior of a contact between rubber and glass (application to wiper blade),” *Tribology International,* vol. 40, no. 10-12 SPEC. ISS., pp. 1484-1491, 2007.

[76] C. Makkar, W. E. Dixon, W. G. Sawyer and G. Hu, “A new continuously differentiable friction model for control systems design,” in *Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.*, Monterey, 2005.

[77] W. Hucho and G. Sovran, “Aerodynamics of road vehicles,” *Annual Review of Fluid Mechanics,* vol. 25, pp. 485-537, 1993.

[78] G. M. LeGood and K. P. Garry, “On the Use of Reference Models in Automotive Aerodynamics,” *SAE TECHNICAL PAPER SERIES 2004-01-1308,* 2004.

[79] E. Guilmineau, “Computational study of flow around a simplified car body,” *Journal of Wind Engineering and Industrial Aerodynamics,* vol. 96, pp. 1207-1217, 2008.

[80] V. K. Yakkundf and S. S. Mantha, “Numerical Simulation of Flowfield around a Car & Aerodynamic Analysis by Fine Tuning Wind Tunnel Pressure Distribution with CFD Results,” *CURIE,* vol. 3, no. 2, pp. 82-102, 2010.

[81] K. M. Ashok and K. Kanniah, “Computational Investigation for Flow and Heat Transfer Characteristics of Automobile Windshield with Impinging Slot Jets,” *SAE Technical Paper 2012-01-1219,* p. 2012.

[82] A. P. Gaylard, “The Appropriate Use of CFD in the Automotive Design Process,” *SAE Technical Paper 2009-01-1162,* 2009.

[83] S. Senthooran, L. Mutnuri, J. Amodeo, R. Powell and C. Freeman, “A Computational Approach to Evaluate the Automotive Windscreen Wiper Placement Options Early in the Design Process,” *SAE Int. J. Passeng. Cars - Mech. Syst.,* no. 1262-1268, 2013.

[84] S. Jallet, S. Devos, D. Maubray, J. Sortais, F. Marmonier and T. Dreher, “Numerical Simulation of Wiper System Aerodynamic Behavior,” in *SAE 2001 World Congress*, Detroit, 2001.

[85] P. Billot, S. Jallet and F. Marmonier, “Simulation of Aerodynamic Uplift Consequences on Pressure Repartition – Application on an Innovative Wiper Blade Design,” *SAE Technical Paper 2001-01-1043,* 2001.

[86] S. H. Lee, S. W. Lee, N. Hur, W. Choi and J. Sul, “Numerical Study on Aerodynamic Lift on Windshield Wiper of High-Speed Passenger Vehicles,” *Transactions of the Korean*

Society of Mechanical Engineers, vol. 35, no. 4, pp. 345-352, 2011.

[87] T. He and L. Peng, “Application of neuron adaptive PID on DSPACE in double loop DC motor control system,” in *2010 International Conference on Computing, Control and Industrial Engineering, CCIE 2010*, Wuhan , 2010.

[88] W. Wang and X. Z. Gao, “A single neuron PID controller based on immune tuning and it's application,” in *Natural Computation (ICNC), 2010 Sixth International Conference on*, Yantai, 2010.

[89] W. Chen, G. Zeng, H. Zou, H. Zhang and C. Tan, “Study of a Single Neuron Fuzzy PID DC Motor Control Method,” in *2012 Second International Conference on Intelligent System Design and Engineering Application*, Sanya, 2012.

[90] L. Xia, H. Wei and J. Gu, “Comparative Analysis on Performances of Adjustable-gain Single-neuron PID Controllers Based on General Fuzzy Logic and Normal Cloud Model,” in *Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on*, Toronto, 2014.

[91] S. Dan and M. Jun, “A single neuron PID controller based PMSM DTC drive system fed by fault tolerant 4-switch 3-phase inverter,” in *2006 1st IEEE Conference on Industrial Electronics and Applications*, Singapore, 2006.

[92] J. Wang, Y. Jing and D. An, “Study of Neuron Adaptive PID Controller in a Single-zone HVAC System,” in *First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06)*, Beijing, 2006.

[93] X. Zan and F. Xie, “Switched reluctance generator system based on single neuron

adaptive PID controller," in *The 2011 International Conference on Advanced Mechatronic Systems*, Zhengzhou, 2011.

[94] J. Minzhi, G. Chao and S. Xiaomin, "Study on the application of the single neuron adaptive PID controller in prestressed tension device," in *IEEE 2011 10th International Conference on Electronic Measurement & Instruments*, Chengdu, 2011.

[95] E. Hughes, Electrical and Electronic Technology, Essex: Pearson Education Limited, 2008.

[96] J. Chiasson, Modeling and High-Performance Control of Electric Machines, New Jersey: John Wiley & Sons, 2005.

[97] A. Shabana, Computational Dynamics, New Jersey: John Wiley & Sons, 2001.

[98] F. Amirouche, Fundamentals of Multibody Dynamics: Theory and Applications, New York: Birkhauser Boston, 2006.

[99] J. Le Rouzic, A. Le Bot, J. Perret-Liaudet, M. Guibert, A. Rusanov, L. Douminge, F. Bretagnol and D. Mazuyer, "Friction-Induced Vibration by Stribeck's Law: Application to Wiper Blade Squeal Noise," *Tribology Letters,* vol. 49, no. 3, pp. 563-572, 2013.

[100] R. Haupt and S. Haupt, Practical Genetic Algorithms, New Jersey: John Wiley & Sons, 2004.

[101] S. Miller and J. Wendlandt, "Real-Time Simulation of Physical Systems Using Simscape," in *Real-Time Simulation Technologies: Principles, Methodologies and Applications*, Boca Raton, FL, Taylor Francis Group, 2013, p. 581 – 597.

[102] K. L. Priddy and P. E. Keller, Artificial Neural Networks: An Introduction, Washington: The International Society for Optical Engineering, 2005.

[103] M. Brown and C. Harris, Neurofuzzy Adaptive Modelling and Control, Hemel Hempstead: Prentice Hall International (UK) Limited, 1994.

[104] Mathworks, "Hyperbolic tangent sigmoid transfer function," [Online]. Available: http://uk.mathworks.com/help/nnet/ref/tansig.html?refresh=true. [Accessed 1 September 2015].

[105] R. Baines, "Sensorless Control for PMDC Motors," *Control Engineering Practice,* pp. 34 - 38, 2005.

Appendix A - Wiper Motor State Space Simulation Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Solve StateSpace Wiper Motor Function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [] = motor_solve (motor_param,torque,vin,ton,tsw,toff,tend)
%Function to solve motor off simulink
%% Extract Motor Parameters
R_Fast=motor_param(1);R_Slow=motor_param(2);L_Fast=motor_param(3);
L_Slow=motor_param(4);K_Fast=motor_param(5);K_Slow=motor_param(6);
J=motor_param(7);b=motor_param(8);ratio=motor_param(9);
%% Define both systems in state space
A1_ss  =  [-R_Slow/L_Slow  -K_Slow/L_Slow;K_Slow/J  -b/J];%  Define  A1
Matrix
B1_ss = [1/L_Slow 0;0 -1/(J*ratio)]; % Define B1 Matrix
C1_ss  =  [1  0;0  1/ratio];  D1_ss  =  zeros(2,2);  %  Define  C1  and  D1
Matrices
motor1_ss = ss(A1_ss,B1_ss,C1_ss,D1_ss); % Define State Space Model
of Motor 1

A2_ss  =  [-R_Fast/L_Fast  -K_Fast/L_Fast;K_Fast/J  -b/J];%  Define  A
Matrix
B2_ss = [1/L_Fast 0;0 -1/(J*ratio)]; % Define B Matrix
motor2_ss = ss(A2_ss,B2_ss,C1_ss,D1_ss); % Define State Space Model
of Motor 2
%% Simulate System 1
in = 2; %Number of Motor inputs (Voltage and load torque)
dt = 0.01; %Simulation timestep
t_1 = 0:dt:tsw; %Time vector for simulation
u_1=zeros((tsw/dt)+1,in); %Input Vector size definition
i=1;
while i<=(tsw/dt)+1 % Populate input vector
    if (i >= ton/dt) % After switch on time
        u_1(i,1)=vin;u_1(i,2)=-torque; %Set input voltage and torque
load
    else
        u_1(i,1)=0;u_1(i,2)=0; %Set input voltage and torque load to
0
    end
i=i+1;
end
[y_1,t_1] = lsim(motor1_ss, u_1, t_1);
%% Simulate System 2
i_0 = y_1(i-1,1);w_0 = y_1(i-1,2);
X0 = [i_0 ; w_0];
tsim = tend - tsw;
t_2 = tsw:dt:tend; %Time vector for simulation
u_2=zeros((tsim/dt)+1,in); %Input Vector size definition
i=1;
while i<=(tsim/dt)+1 % Populate input vector
    if (i >= (toff-tsw)/dt) % After switch on time
        u_2(i,1)=0;u_2(i,2)=0; %Set input voltage and torque load
    else
        u_2(i,1)=vin;u_2(i,2)=-torque; %Set input voltage and torque
load to 0
    end
i=i+1;
end
[y_2,t_2] = lsim(motor2_ss, u_2, t_2, X0);
%% Append Results and plot
```

```
y = vertcat (y_1,y_2);t = vertcat (t_1,t_2);
figure()
subplot(2,1,1)
plot     (t(:,1),y(:,1),'LineWidth',     2);xlabel('Time      (s)');
ylabel('Current (A)');title('Current Plot');grid on
subplot(2,1,2)
plot     (t(:,1),y(:,2),'LineWidth',     2);xlabel('Time      (s)');
ylabel('Velocity (rad/s)');title('Velocity Plot');grid on
end
```

Appendix B - Slave Driven Linkage Parameters Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Slave Driven Linkage System Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function    [len,    rot,    inertia]    =    simmech_slave_parameters
(O,A,B,C,D,E,F,mass)
len = all_lengths (O,A,B,C,D,E,F); %Call all_lengths function
rot    =    quaternion_rotations    (O,A,B,C,D,E,F,len);    %    call
quaternion_rotations function
I1=((1/12)*mass(1)*(len(1)^2)); I2=((1/12)*mass(2)*(len(2)^2));
I3=((1/12)*mass(3)*(len(3)^2)); I4=((1/12)*mass(4)*(len(4)^2));
I5=((1/12)*mass(5)*(len(5)^2)); I6=((1/12)*mass(6)*(len(6)^2));
inertia = [I1 I2 I3 I4 I5 I6]; %Set inertia values
function [lengths] = all_lengths (O,A,B,C,D,E,F)
L1  =  0;  L2  =  single_length  (O,A);L3  =  single_length  (A,B);L4  =
single_length (B,C);
L5 = single_length (D,E);L6 = single_length (E,F);
lengths =[L1, L2, L3, L4, L5, L6];
end
function [L] = single_length(A,B)
temp = B - A;
L = sqrt(temp(1)^2+temp(2)^2+temp(3)^2);
end
function      [quaternion_rotations]      =      quaternion_rotations
(O,A,B,C,D,E,F,len)
q1   =   [0   0   0   0];   q2   =   quat_vec   (O,A,len(2));q3   =   quat_vec
(A,B,len(3));
q4 = quat_vec (B,C,len(4));q5 = quat_vec (D,E,len(5));q6 = quat_vec
(E,F,len(6));
quaternion_rotations = [q1,q2,q3,q4,q5,q6];
end
function [q] = quat_vec (A,B,C)
theta = acosd((dot((B-A),[1 0 0])/C));%dot product to find theta
if theta == 180
    w_quat = 1;    x_quat = 0;    y_quat = 0;    z_quat = 0;
elseif theta == 0
    w_quat = 1;    x_quat = 0;    y_quat = 0;    z_quat = 0;
else
n = ((cross(A-B,[1 0 0]))/((C*sind(theta)))); %calculate unit vector
w_quat = cosd(theta/2);x_quat = sind(theta/2)*n(1);
y_quat = sind(theta/2)*n(2);z_quat = sind(theta/2)*n(3);
end
q = [x_quat, y_quat, z_quat, w_quat];
end
end
```

Appendix C - Linkage System Kinematic Equations

C.1 Master Driven Linkages

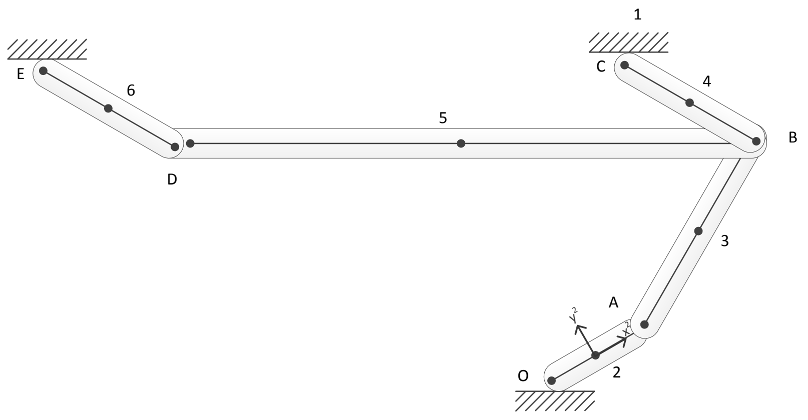


Figure C-1: Master Driven Linkage System Diagram

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}^1 & \mathbf{q}^2 & \mathbf{q}^3 & \mathbf{q}^4 & \mathbf{q}^5 & \mathbf{q}^6 \end{bmatrix}^T \tag{9.1}$$

where

$$\mathbf{q}^i = \begin{bmatrix} \mathbf{R}^i & \theta^i \end{bmatrix}^T \text{ and } \mathbf{R}^i = \begin{bmatrix} R_x^i & R_y^i \end{bmatrix}^T \text{ and } i = 1, 2, \ldots, 6 \tag{9.2}$$

$$\mathbf{C}(\mathbf{q},t) = \begin{bmatrix} \mathbf{C}(\mathbf{q},t)^1 & \mathbf{C}(\mathbf{q},t)^2 \end{bmatrix}^T = [\mathbf{0}]$$

$$\mathbf{C}(\mathbf{q},t)^1 = \begin{bmatrix} R_x^1 \\ R_y^1 \\ \theta^1 \\ R_x^2 - \frac{l^2}{2}\cos\theta^2 \\ R_y^2 - \frac{l^2}{2}\sin\theta^2 \\ R_x^2 + \frac{l^2}{2}\cos\theta^2 - R_x^3 + \frac{l^3}{2}\cos\theta^3 \\ R_y^2 + \frac{l^2}{2}\sin\theta^2 - R_y^3 + \frac{l^3}{2}\sin\theta^3 \\ R_x^3 + \frac{l^3}{2}\cos\theta^3 - R_x^4 + \frac{l^4}{2}\cos\theta^4 \\ R_y^3 + \frac{l^3}{2}\sin\theta^3 - R_y^4 + \frac{l^4}{2}\sin\theta^4 \end{bmatrix}, \text{ and } \mathbf{C}(\mathbf{q},t)^2 = \begin{bmatrix} R_x^4 + \frac{l^4}{2}\cos\theta^4 - X^1 \\ R_y^4 + \frac{l^4}{2}\sin\theta^4 - Y^1 \\ R_x^3 + \frac{l^3}{2}\cos\theta^3 - R_x^5 + \frac{l^5}{2}\cos\theta^5 \\ R_x^3 + \frac{l^3}{2}\cos\theta^3 - R_x^5 + \frac{l^5}{2}\cos\theta^5 \\ R_x^5 + \frac{l^5}{2}\cos\theta^5 - R_x^6 + \frac{l^6}{2}\cos\theta^6 \\ R_y^5 + \frac{l^5}{2}\sin\theta^5 - R_y^6 + \frac{l^6}{2}\sin\theta^6 \\ R_x^6 + \frac{l^6}{2}\cos\theta^6 - X^2 \\ R_y^6 + \frac{l^6}{2}\sin\theta^6 - Y^2 \\ \theta^2 - \omega^2 t - \theta_0^2 \end{bmatrix} \tag{9.3}$$

$$
\mathbf{C_q} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & \frac{l^2}{2}\sin\theta^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -\frac{l^2}{2}\cos\theta^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -\frac{l^2}{2}\sin\theta^2 & -1 & 0 & -\frac{l^3}{2}\sin\theta^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \frac{l^2}{2}\cos\theta^2 & 0 & -1 & \frac{l^3}{2}\cos\theta^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{l^3}{2}\sin\theta^3 & -1 & 0 & -\frac{l^4}{2}\sin\theta^4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{l^3}{2}\cos\theta^3 & 0 & -1 & \frac{l^4}{2}\cos\theta^4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{l^4}{2}\sin\theta^4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{l^4}{2}\cos\theta^4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{l^3}{2}\sin\theta^3 & 0 & 0 & 0 & -1 & 0 & -\frac{l^5}{2}\sin\theta^5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{l^3}{2}\cos\theta^3 & 0 & 0 & 0 & 0 & -1 & \frac{l^5}{2}\cos\theta^5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{l^5}{2}\sin\theta^5 & -1 & 0 & -\frac{l^6}{2}\sin\theta^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{l^5}{2}\cos\theta^5 & 0 & -1 & \frac{l^6}{2}\cos\theta^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{l^6}{2}\sin\theta^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{l^6}{2}\cos\theta^6 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \qquad (9.4)
$$

$$
\mathbf{C_q\dot{q}} = \begin{bmatrix} \mathbf{C_q\dot{q}}^1 & \mathbf{C_q\dot{q}}^2 \end{bmatrix}
$$

$$
\mathbf{C_q\dot{q}}^1 = \begin{bmatrix}
\dot{R}_x^1 \\
\dot{R}_y^1 \\
\dot{\theta}^1 \\
\dot{R}_x^2 - \frac{\dot{\theta}^2 l^2}{2}\sin\theta^2 \\
\dot{R}_y^2 - \frac{\dot{\theta}^2 l^2}{2}\cos\theta^2 \\
\dot{R}_x^2 + \frac{\dot{\theta}^2 l^2}{2}\sin\theta^2 - \dot{R}_x^3 + \frac{\dot{\theta}^3 l^3}{2}\sin\theta^3 \\
\dot{R}_y^2 + \frac{\dot{\theta}^2 l^2}{2}\cos\theta^2 - \dot{R}_y^3 + \frac{\dot{\theta}^3 l^3}{2}\cos\theta^3 \\
\dot{R}_x^3 + \frac{\dot{\theta}^3 l^3}{2}\sin\theta^3 - \dot{R}_x^4 + \frac{\dot{\theta}^4 l^4}{2}\sin\theta^4 \\
\dot{R}_y^3 + \frac{\dot{\theta}^3 l^3}{2}\cos\theta^3 - \dot{R}_y^4 + \frac{\dot{\theta}^4 l^4}{2}\cos\theta^4
\end{bmatrix}, \quad \text{and} \quad
\mathbf{C_q\dot{q}}^2 = \begin{bmatrix}
\dot{R}_x^4 + \frac{\dot{\theta}^4 l^4}{2}\sin\theta^4 \\
R_y^4 + \frac{\dot{\theta}^4 l^4}{2}\cos\theta^4 \\
\dot{R}_x^3 + \frac{\dot{\theta}^3 l^3}{2}\sin\theta^3 - \dot{R}_x^5 + \frac{\dot{\theta}^5 l^5}{2}\sin\theta^5 \\
\dot{R}_y^3 + \frac{\dot{\theta}^5 l^3}{2}\cos\theta^5 - \dot{R}_y^5 + \frac{\dot{\theta}^5 l^5}{2}\cos\theta^5 \\
\dot{R}_x^5 + \frac{\dot{\theta}^5 l^5}{2}\sin\theta^5 - \dot{R}_x^6 + \frac{\dot{\theta}^6 l^6}{2}\sin\theta^6 \\
\dot{R}_y^5 + \frac{\dot{\theta}^5 l^5}{2}\cos\theta^5 - \dot{R}_y^6 + \frac{\dot{\theta}^6 l^6}{2}\cos\theta^6 \\
\dot{R}_x^6 + \frac{\dot{\theta}^6 l^6}{2}\sin\theta^6 \\
\dot{R}_y^6 + \frac{\dot{\theta}^6 l^6}{2}\cos\theta^6 \\
\dot{\theta}^2
\end{bmatrix} \qquad (9.5)
$$

$$
\left(\mathbf{C_q}\dot{\mathbf{q}}\right)_q = \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&\frac{\dot{\theta}^2 l^2}{2}\cos\theta^2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&\frac{\dot{\theta}^2 l^2}{2}\sin\theta^2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&-\frac{\dot{\theta}^2 l^2}{2}\cos\theta^2&0&0&-\frac{\dot{\theta}^3 l^3}{2}\cos\theta^3&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&-\frac{\dot{\theta}^2 l^2}{2}\sin\theta^2&0&0&-\frac{\dot{\theta}^3 l^3}{2}\sin\theta^3&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^3 l^3}{2}\cos\theta^3&0&0&-\frac{\dot{\theta}^4 l^4}{2}\cos\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^3 l^3}{2}\sin\theta^3&0&0&-\frac{\dot{\theta}^4 l^4}{2}\sin\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^4 l^4}{2}\cos\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^4 l^4}{2}\sin\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^3 l^3}{2}\cos\theta^3&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\cos\theta^5&0&0&0\\
0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^3 l^3}{2}\sin\theta^3&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\sin\theta^5&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\cos\theta^5&0&0&-\frac{\dot{\theta}^6 l^6}{2}\cos\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\sin\theta^5&0&0&-\frac{\dot{\theta}^6 l^6}{2}\sin\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^6 l^6}{2}\cos\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^6 l^6}{2}\sin\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix} \tag{9.6}
$$

$$
\mathbf{Q}_d = -\left(\mathbf{C_q}\dot{\mathbf{q}}\right)_q \dot{\mathbf{q}} = \begin{bmatrix}\mathbf{Q}_d^1 & \mathbf{Q}_d^2\end{bmatrix}
$$

$$
\mathbf{Q}_d^1 = \begin{bmatrix}
0\\
0\\
0\\
\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\cos\theta^2\\
\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\sin\theta^2\\
-\left(\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\cos\theta^2 + \frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\cos\theta^3\right)\\
-\left(\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\sin\theta^2 + \frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\sin\theta^3\right)\\
-\left(\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\cos\theta^3 + \frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\cos\theta^4\right)\\
-\left(\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\sin\theta^3 + \frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\sin\theta^4\right)
\end{bmatrix}, \quad \text{and} \quad
\mathbf{Q}_d^2 = \begin{bmatrix}
-\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\cos\theta^4\\
-\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\sin\theta^4\\
-\left(\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\cos\theta^3 + \frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\cos\theta^5\right)\\
-\left(\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\sin\theta^3 + \frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\sin\theta^5\right)\\
-\left(\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\cos\theta^5 + \frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\cos\theta^6\right)\\
-\left(\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\sin\theta^5 + \frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\sin\theta^6\right)\\
-\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\cos\theta^6\\
-\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\sin\theta^6\\
0
\end{bmatrix} \tag{9.7}
$$

C.2 Centre Driven Linkages

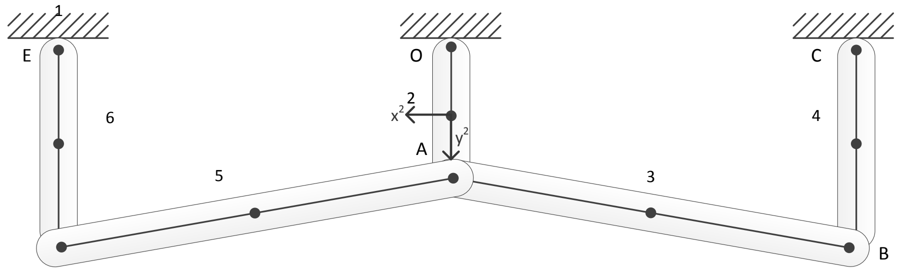


Figure C-2: Centre Driven Linkage System Diagram

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}^1 & \mathbf{q}^2 & \mathbf{q}^3 & \mathbf{q}^4 & \mathbf{q}^5 & \mathbf{q}^6 \end{bmatrix}^T \tag{9.8}$$

where

$$\mathbf{q}^i = \begin{bmatrix} \mathbf{R}^i & \theta^i \end{bmatrix}^T \text{ and } \mathbf{R}^i = \begin{bmatrix} R_x^i & R_y^i \end{bmatrix}^T \text{ and } i = 1,2,\ldots,6 \tag{9.9}$$

$$\mathbf{C}(\mathbf{q},t) = \begin{bmatrix} \mathbf{C}(\mathbf{q},t)^1 & \mathbf{C}(\mathbf{q},t)^2 \end{bmatrix}$$

$$\mathbf{C}(\mathbf{q},t)^1 = \begin{bmatrix} R_x^1 \\ R_y^1 \\ \theta^1 \\ R_x^2 - \frac{l^2}{2}\cos\theta^2 \\ R_y^2 - \frac{l^2}{2}\sin\theta^2 \\ R_x^2 + \frac{l^2}{2}\cos\theta^2 - R_x^3 + \frac{l^3}{2}\cos\theta^3 \\ R_y^2 + \frac{l^2}{2}\sin\theta^2 - R_y^3 + \frac{l^3}{2}\sin\theta^3 \\ R_x^3 + \frac{l^3}{2}\cos\theta^3 - R_x^4 + \frac{l^4}{2}\cos\theta^4 \\ R_y^3 + \frac{l^3}{2}\sin\theta^3 - R_y^4 + \frac{l^4}{2}\sin\theta^4 \end{bmatrix}, \text{ and } \mathbf{C}(\mathbf{q},t)^2 = \begin{bmatrix} R_x^4 + \frac{l^4}{2}\cos\theta^4 - X^1 \\ R_y^4 + \frac{l^4}{2}\sin\theta^4 - Y^1 \\ R_x^2 - \frac{l^2}{2}\cos\theta^2 - R_x^5 + \frac{l^5}{2}\cos\theta^5 \\ R_y^2 - \frac{l^2}{2}\sin\theta^2 - R_y^5 + \frac{l^5}{2}\sin\theta^4 \\ R_x^5 + \frac{l^5}{2}\cos\theta^5 - R_x^6 + \frac{l^6}{2}\cos\theta^6 \\ R_y^5 + \frac{l^5}{2}\sin\theta^5 - R_y^6 + \frac{l^6}{2}\sin\theta^6 \\ R_x^6 + \frac{l^6}{2}\cos\theta^6 - X^2 \\ R_y^6 + \frac{l^6}{2}\sin\theta^6 - Y^2 \\ \theta^2 - \omega^2 t - \theta_0^2 \end{bmatrix} \tag{9.10}$$

$$
\mathbf{C_q} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & \frac{l^2}{2}\sin\theta^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -\frac{l^2}{2}\cos\theta^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -\frac{l^2}{2}\sin\theta^2 & -1 & 0 & -\frac{l^3}{2}\sin\theta^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \frac{l^2}{2}\cos\theta^2 & 0 & -1 & \frac{l^3}{2}\cos\theta^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{l^3}{2}\sin\theta^3 & -1 & 0 & -\frac{l^4}{2}\sin\theta^4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{l^3}{2}\cos\theta^3 & 0 & -1 & \frac{l^4}{2}\cos\theta^4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{l^4}{2}\sin\theta^4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{l^4}{2}\cos\theta^4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -\frac{l^2}{2}\sin\theta^2 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -\frac{l^5}{2}\sin\theta^5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \frac{l^2}{2}\cos\theta^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & \frac{l^5}{2}\cos\theta^5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{l^5}{2}\sin\theta^5 & -1 & 0 & -\frac{l^6}{2}\sin\theta^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{l^5}{2}\cos\theta^5 & 0 & -1 & \frac{l^6}{2}\cos\theta^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -\frac{l^6}{2}\sin\theta^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \frac{l^6}{2}\cos\theta^6 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \quad (9.11)
$$

$$
\mathbf{C_q}\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{C_q}\dot{\mathbf{q}}^1 & \mathbf{C_q}\dot{\mathbf{q}}^2 \end{bmatrix}
$$

$$
\mathbf{C_q}\dot{\mathbf{q}}^1 = \begin{bmatrix}
\dot{R}_x^1 \\
\dot{R}_y^1 \\
\dot{\theta}^1 \\
\dot{R}_x^2 + \frac{\dot{\theta}^2 l^2}{2}\sin\theta^2 \\
\dot{R}_y^2 - \frac{\dot{\theta}^2 l^2}{2}\cos\theta^2 \\
\dot{R}_x^2 - \frac{\dot{\theta}^2 l^2}{2}\sin\theta^2 - \dot{R}_x^3 - \frac{\dot{\theta}^3 l^3}{2}\sin\theta^3 \\
\dot{R}_y^2 + \frac{\dot{\theta}^2 l^2}{2}\cos\theta^2 - \dot{R}_y^3 + \frac{\dot{\theta}^3 l^3}{2}\cos\theta^3 \\
\dot{R}_x^3 - \frac{\dot{\theta}^3 l^3}{2}\sin\theta^3 - \dot{R}_x^4 - \frac{\dot{\theta}^4 l^4}{2}\sin\theta^4 \\
\dot{R}_y^3 + \frac{\dot{\theta}^3 l^3}{2}\cos\theta^3 - \dot{R}_y^4 + \frac{\dot{\theta}^4 l^4}{2}\cos\theta^4
\end{bmatrix}, \text{ and } \mathbf{C_q}\dot{\mathbf{q}}^2 = \begin{bmatrix}
\dot{R}_x^4 - \frac{\dot{\theta}^4 l^4}{2}\sin\theta^4 \\
\dot{R}_y^4 + \frac{\dot{\theta}^4 l^4}{2}\cos\theta^4 \\
\dot{R}_x^2 - \frac{\dot{\theta}^2 l^2}{2}\sin\theta^2 - \dot{R}_x^5 + \frac{\dot{\theta}^5 l^5}{2}\sin\theta^5 \\
\dot{R}_y^2 + \frac{\dot{\theta}^2 l^2}{2}\cos\theta^2 - \dot{R}_y^5 + \frac{\dot{\theta}^5 l^5}{2}\cos\theta^4 \\
\dot{R}_x^5 - \frac{\dot{\theta}^5 l^5}{2}\sin\theta^5 - \dot{R}_x^6 - \frac{\dot{\theta}^6 l^6}{2}\sin\theta^6 \\
\dot{R}_y^5 + \frac{\dot{\theta}^5 l^5}{2}\cos\theta^5 - \dot{R}_y^6 + \frac{\dot{\theta}^6 l^6}{2}\cos\theta^6 \\
\dot{R}_x^6 - \frac{\dot{\theta}^6 l^6}{2}\sin\theta^6 \\
\dot{R}_y^6 + \frac{\dot{\theta}^6 l^6}{2}\cos\theta^6 \\
\dot{\theta}^2
\end{bmatrix} \quad (9.12)
$$

$$
\mathbf{C_q} = \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&\frac{\dot{\theta}^2 l^2}{2}\cos\theta^2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&\frac{\dot{\theta}^2 l^2}{2}\sin\theta^2&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&-\frac{\dot{\theta}^2 l^2}{2}\cos\theta^2&0&0&-\frac{\dot{\theta}^3 l^3}{2}\cos\theta^3&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&-\frac{\dot{\theta}^2 l^2}{2}\sin\theta^2&0&0&-\frac{\dot{\theta}^3 l^3}{2}\sin\theta^3&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^3 l^3}{2}\cos\theta^3&0&0&-\frac{\dot{\theta}^4 l^4}{2}\cos\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^3 l^3}{2}\sin\theta^3&0&0&-\frac{\dot{\theta}^4 l^4}{2}\sin\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^4 l^4}{2}\cos\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^4 l^4}{2}\sin\theta^4&0&0&0&0&0&0\\
0&0&0&0&0&-\frac{\dot{\theta}^2 l^2}{2}\cos\theta^2&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\cos\theta^5&0&0&0\\
0&0&0&0&0&-\frac{\dot{\theta}^2 l^2}{2}\sin\theta^2&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\sin\theta^5&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\cos\theta^5&0&0&-\frac{\dot{\theta}^6 l^6}{2}\cos\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^5 l^5}{2}\sin\theta^5&0&0&-\frac{\dot{\theta}^6 l^6}{2}\sin\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^6 l^6}{2}\cos\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&-\frac{\dot{\theta}^6 l^6}{2}\sin\theta^6\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix} \tag{9.13}
$$

$$
\mathbf{Q}_d = -\left(\mathbf{C_q}\dot{\mathbf{q}}\right)_q \dot{\mathbf{q}} = \left[\mathbf{Q}_d^1 \quad \mathbf{Q}_d^2\right]^T
$$

$$
\mathbf{Q}_d^1 = \begin{bmatrix}
0\\
0\\
0\\
\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\cos\theta^2\\
\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\sin\theta^2\\
-\left(\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\cos\theta^2+\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\cos\theta^3\right)\\
-\left(\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\sin\theta^2+\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\sin\theta^3\right)\\
-\left(\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\cos\theta^3+\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\cos\theta^4\right)\\
-\left(\frac{\left(\dot{\theta}^3\right)^2 l^3}{2}\sin\theta^3+\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\sin\theta^4\right)
\end{bmatrix}, \quad \text{and} \quad
\mathbf{Q}_d^2 \begin{bmatrix}
-\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\cos\theta^4\\
-\frac{\left(\dot{\theta}^4\right)^2 l^4}{2}\sin\theta^4\\
-\left(\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\cos\theta^5+\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\cos\theta^5\right)\\
-\left(\frac{\left(\dot{\theta}^2\right)^2 l^2}{2}\sin\theta^5+\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\sin\theta^5\right)\\
-\left(\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\cos\theta^5+\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\cos\theta^6\right)\\
-\left(\frac{\left(\dot{\theta}^5\right)^2 l^5}{2}\sin\theta^5+\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\sin\theta^6\right)\\
-\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\cos\theta^6\\
-\frac{\left(\dot{\theta}^6\right)^2 l^6}{2}\sin\theta^6\\
0
\end{bmatrix} \tag{9.14}
$$

Appendix D - Grashof's Law for Slave Driven Linkages Matlab Code

```
function [state] = grashof_slave (O,A,B,C,D,E,F) %Function to
determine whether the slave driven linkages satisfy Grashof's law
L1 = single_length (O,C);L2 = single_length (O,A);L3 = single_length
(A,B);L4 = single_length (B,C);
L5 = single_length (C,D);L6 = single_length (D,E);L7 = single_length
(E,F);L8 = single_length (F,C);%Calculate all linkage lengths
function [L] = single_length(A,B)%Calculates Sigle Lengths
temp = B - A;
L = sqrt(temp(1)^2+temp(2)^2+temp(3)^2);
end
config_1 = sort([L1, L2, L3, L4]);%Sorts by size the 4-bar linkage
1's linkages
constraint_1 = (config_1(1) + config_1(4)) - (config_1(2) +
config_1(3));%Grashof's law for configuration 1
if (constraint_1 <= 0)%Output state depending on whether the law is
satified, and if not, how is it violated
    state = 0;%Satisfied
else
    state = 1;%Violated by 4 bar linkage 1
end
if state ==0 %Execute if first 4 bar linkage satisfies Grashof's law
th2 = (0:(4*pi/1000):4*pi);ACO_m = zeros(1,1001);ACB_m =
zeros(1,1001);i = 1;th4_m = zeros(1,1001);gamma_m = zeros(1,1001);
e_m = zeros(1,1001);height = abs(C(2)-F(2)); width = abs(C(1)-
F(1));sigma = atan(height/width);%Initialise size of matrices
height_1 = abs(O(2)-C(2)); width_1 = abs(O(1)-C(1));sigma_1 =
atan(height_1/width_1);
    while i <= 1001 %Sweep crank by 360 Deg
    th = th2(i);
    AC = sqrt((L1)^2+(L2)^2-(2*L1*L2*cos(th))); % Distance between A
and C
    ACO    =    real(acos(((L1)^2+(AC)^2-(L2)^2)/(2*L1*AC)));%Angle
between A, C and O
    ACB   =   real(acos(((L4)^2+(AC)^2-(L3)^2)/(2*L4*AC)));   %Angle
between A, C and B
    ACO_m(i) = ACO; ACB_m(i) = ACB; %Convert to matrix
        if th >= 0  && th <= pi%Value of th4 depends on whether th2
is between 0 and pi or pi and 2*pi radians
            th4_m(i) = sigma_1 + ACB - ACO;
        else
            th4_m(i) = sigma_1 + ACB + ACO;
        end
   % gamma_m(i) = th4_m(i)-pi-sigma; %Find interim angle
   gamma_m(i) = th4_m(i)+sigma; %Find interim angle
    e_m(i)  =  sqrt((L5)^2+(L8)^2-(2*L5*L8*cos(gamma_m(i))));  %Find
all lengths of e
    i = i+1;
    end
    max_e = max(e_m);% Find max length of i
        if ((L6 + L7) >= max_e) %Apply inequality
            state = 0; %Satisfied
        else
            state = 2; %Violated by 4 bar linkage 2
        end
    end
end
```

Appendix E - Genetic Algorithm Matlab Code

E.1 Genetic Algorithm

```
   %% Genetic algorithm (based on Haupt & Haupt, 2003)

   %% Set the GA parameters
   ff='cost_function';
   npar=17;              % number of optimization variables
   varlo = min_param; varhi = max_param; % Sets the limits of the
variables
   maxit=200;             % max number of iterations
   mincost=0.1;           % minimum cost
   popsize=40;          % set population size
   mutrate=.1;          % set mutation rate
   selection=0.5;       % fraction of population kept
   elite = 5;             %number of chromosomes (best) to save from
mutation,
   Nt=npar;              % continuous parameter GA Nt=#variables
   par = zeros (popsize,npar); %initialise size of par
   keep=floor(selection*popsize);% #population members that survive
   nmut=ceil((popsize-1)*Nt*mutrate);  % total number of mutations
   M=ceil((popsize-keep)/2);       % number of matings
   off = zeros(popsize-keep,npar); %Initialise offspring matrix size
   %_____________________________________________________________
   %% Create the initial population
   iga=0;              % generation counter initialized
   j=1;
   while j <= popsize % random paramerters generated between high
and low boundaries for each parameter
       i=1;
   while i <= npar
       if (i == 2) || (i==4) || (i==6)  %Handle Motor Inequalities
           par(j,i)=(par(j,i-1)-varlo(i))*rand+varlo(i);   %Restrict
values of R_Fast, L_Fast and K_Fast
       else
       par(j,i)=(varhi(i)-varlo(i))*rand+varlo(i);  %No inequalities
       end
       i=i+1;
   end
       j=j+1;
   end
   cost=cost_function(Slow_Current,Fast_Current,Slow_Park,Fast_Park,
par,popsize);% calculates population cost
   [cost,ind]=sort(cost);          % sort for combined cost
   par_1=par(ind(:,2),:);          % sort for cost 1
   par_2=par(ind(:,3),:);          % sort for cost 2
   par=par(ind(:,1),:);            % Define population matrix
   minc(1)=min(cost(:,1));            % minc contains min cost in
population
   meanc(1)=mean(cost(:,1));         % meanc contains mean cost of
population
   minc_1(1)=min(cost(:,2));         % minc_1 contains min current
cost cost in population
   minc_2(1)=min(cost(:,3));        % minc contains min park switch
cost in population

   par = par (1:1:keep,:); % Removes weakest chromosomes
   %% Begin iterations
   while iga<maxit
       iga=iga+1;
   %% Tournament Selection

   contender = zeros(3,1); %Initialize matrices
   draw = zeros(3,1);
   winner = zeros(keep,npar);
   j=1;
   while j <= keep %To generate #"Keep" parents
       i=1;
   while i<=3 % 3 random parents are selected
       draw(i) = ceil(rand*keep); % A random integer between 1 and
#"keep" is selected
       contender(i)=  cost(draw(i),1);%  The  cost  of  the  random
chromosomes from the mating pool are accessed
       i=i+1;
   end
   [contender,ind_1]=sort(contender); % The cost of contenders are
sorted
   winner(j,:)=par(draw(ind_1(1)),:);  %  The  chromosome  with  the
lowest cost is chosen as the parent.
   j=j+1;
   end


   %% Mating
   xm=1:2:keep; % Index of members of mating pool denoted as mothers
   xp=2:2:keep; % Index of members of mating pool denoted as fathers
   beta = zeros(M,npar+1); %Initialise the size of beta (ensure an
extra 0 at the end)
   i=1;
   while i<=M
   alpha = ceil(rand*npar); %Generate a random integer between 1 and
Npar, this determines the number of genes to be modified.
   j=1;
   while j <= alpha
   beta(1,j)  =  ceil(rand*npar);  %  For  each  "mate"  alpha  random
numbers are generated which denote the genes to be combined.
   j=j+1;
   end
       beta=unique(beta(1,:));  %Select  only  unique  genes  to  avoid
repetition
       alpha  =  (length  ((beta(1,:)))-1);  %Reset  alpha  to  match  the
number of genes to change
       gamma  =  rand;  %Generate  a  random  number  between  0  and  1  as
the blending coefficient
           ma = winner(xm(i),:); %Select mother gene
           pa = winner(xp(i),:); %Select father gene
       j=1;
       while j <= alpha % Iterate for number of genes to be modified
           rho_new1     =     ma(beta(1,j+1))-gamma*(ma(beta(1,j+1))-
pa(beta(1,j+1))); %Blending operation from Haupt & Haupt (pg 59)
           rho_new2     =     pa(beta(1,j+1))+gamma*(ma(beta(1,j+1))-
pa(beta(1,j+1)));
           ma(1,beta(j+1))=rho_new1;  %  Replace  gene  in  original
chromosome mother with new gene
           pa(1,beta(j+1))=rho_new1;  %  Replace  gene  in  original
chromosome father with new gene
           j=j+1;
       end
        off(xm(i),:)=ma; % Add offspring to population
        off(xp(i),:)=pa;
       i=i+1;
   end
   par = vertcat (par,off); %Generates new population
   %% Mutations
   mrow=sort(ceil(rand(1,nmut)*(popsize-1))+1);   %randomly   select
genes to mutate
   mcol=ceil(rand(1,nmut)*Nt);
   for ii=1:nmut
       if (mrow(ii)>elite) % Allows first #'elite' rows to be passed
over for mutation
           if  (mcol(ii)  ==  2)  ||  (mcol(ii)==4)  ||  (mcol(ii)==6)
%Handle Motor Inequalities
                   par(mrow(ii),mcol(ii))=(par(mrow(ii),(mcol(ii)-
1))-varlo(mcol(ii)))*rand+varlo(mcol(ii));  %Ensures  mutation  does
not violate inequality
           elseif (mcol(ii) == 1) || (mcol(ii)==3) || (mcol(ii)==5)
                   par(mrow(ii),mcol(ii))=(varhi(mcol(ii))-
par(mrow(ii),(mcol(ii)+1)))*rand+par(mrow(ii),(mcol(ii)+1));
%Ensures mutation does not violate inequality
           else
                   par(mrow(ii),mcol(ii))=(varhi(mcol(ii))-
varlo(mcol(ii)))*rand+varlo(mcol(ii));%  normal  mutation  (i.e.  no
inequalities to handle)
           end
           else
       end
   end % ii

   %% Only solve new and mutated chromosomes

   new_gen  =  par((popsize-keep)+1:1:popsize,:);  %selects  the  new
generation

   mut_gen = unique(mrow); % Extracts index of chromosomes that have
been muatated
   mut_gen = mut_gen(mut_gen > elite & mut_gen < (keep+1)); %Filters
out elite chromosomes and new generation
   i=1;
   while i <= length (mut_gen)
      old_gen_mut(i,:)  =  par(mut_gen(i),:);  %Generate  matrix  with
mutated old generation
      i=i+1;
   end
   new_gen = vertcat (new_gen,old_gen_mut); % Matrix containing new
generation chromosomes different to old generation
   gen_size  =  size(new_gen);gen_size  =  gen_size(1,1);  %  Calculates
the size of the new generation

   cost_ng=cost_function(Slow_Current,Fast_Current,Slow_Park,Fast_Pa
rk,new_gen,gen_size);  %Calculates  the  cost  of  only  the  changed
chromosomes

   %% Rebuld and sort matrix

   i=elite+1;
   j=1;k=1;
   while i <= keep
       if mut_gen (j) == i
           i=i+1;
           if j < length(mut_gen)
```

```
                j=j+1;
            end
        else
            old_gen(k)=i; %old_gen lists the indices of the parent
non-elite chromosomes not mutated
            i=i+1;k=k+1;
        end
   end

   if elite ~= 0 % Add elites
   elite_gen = 1:1:elite;
   old_gen = horzcat (elite_gen,old_gen); %If elitism is used, the
elite chromosomes are added to the start of old_gen
   end
   i=1;
   while i<= length(old_gen)
        old_par(i,:) = par(old_gen(i),:); %Extract chromosomes from
old generation
        old_cost(i,:) = cost(old_gen(i),:); %Extract costs from old
generation
   i=i+1;
   end

   par = vertcat(old_par, new_gen); %Generate complete next
generation matrix
   cost = vertcat(old_cost, cost_ng); %Generate complete costs
matrix

   [cost,ind]=sort(cost);          % sort costs based on total cost
   par_1=par(ind(:,2),:);          % sort for cost 1
   par_2=par(ind(:,3),:);          % sort for cost 2
   par=par(ind(:,1),:);            % sort for combined cost

   minc(iga+1)=min(cost(:,1));        % minc contains min cost in
population
   meanc(iga+1)=mean(cost(:,1));     % meanc contains mean cost of
population
   minc_1(iga+1)=min(cost(:,2));     % minc_1 contains min current
cost in population
   minc_2(iga+1)=min(cost(:,3));   % minc_2 contains min park switch
cost in population
   %% Stopping Criteria
   if iga>maxit || cost(1)<mincost %Assess stopping Criteria
      break
   end

   [iga cost(1,1)] %Output running cost and generation

   end %end of iterations from iga reaching maximum iterations
```

E.2 Cost Function

```
   function
[cost]=cost_function(Slow_Current,Fast_Current,Slow_Park,Fast_Park,p
ar,popsize)

   global J K_Fast K_Slow L_Fast L_Slow R_Fast R_Slow bm u_dry1
u_dry2 u_dry3 u_wet1 u_wet2 u_wet4 u_wet6 u_wet7 F;

   cost = zeros (popsize,3);% Initialise cost matrix as zeros
   j=1;
   while (j<=popsize) %J < popsize
   R_Slow = par (j,1); R_Fast = par (j,2); L_Slow = par (j,3);
L_Fast = par (j,4);
   K_Slow = par (j,5); K_Fast = par (j,6); J = par (j,7); bm = par
(j,8);
   u_dry1 = par(j,9); u_dry2 = par(j,10); u_dry3 = par(j,11);
   u_wet1 = par(j,12);u_wet2 = par(j,13); u_wet4 = par(j,14);u_wet6
= par(j,15);u_wet7= par(j,16);
   F = par(j,17);% Set parameters

   sim ('GA_Model'); %Simulate model for each chromosome

   toutsim = round(tout*1000)/1000; %Rounds all members of Tout to
3dp (i.e. 1ms)
   [toutsim,indexsim,ic] = unique(toutsim); %Finds the unique values
of toutsim and the index in which the occur
   toutsim = horzcat(toutsim,indexsim); %Appends the unique values
of tout and their index

   i=2;
   while i <= length(toutsim) % Extract Data values for times in
toutsim so that they can be directly compared
       k=toutsim(i,2);
       k2 = toutsim(i,1);
       Sim_Slow_Current    (1,i)    =    Sim_Motor_Current_slow(k,2);
%Samples the simulatd data
       Sim_Fast_Current    (1,i)    =    Sim_Motor_Current_fast(k,2);
%Samples the simulatd data
       Sim_Slow_Park  (1,i)  =  Sim_Park_Switch_slow  (k,2);  %Samples
the simulatd data
       Sim_Fast_Park  (1,i)  =  Sim_Park_Switch_fast  (k,2);  %Samples
the simulatd data
       Data_Slow_Current   (1,i)   =   Slow_Current   (round(k2*1000));
%Samples the real data
       Data_Fast_Current   (1,i)   =   Fast_Current   (round(k2*1000));
%Samples the real data
       Data_Slow_Park  (1,i)  =  Slow_Park  (round(k2*1000));  %Samples
the real data
       Data_Fast_Park  (1,i)  =  Fast_Park  (round(k2*1000));  %Samples
the real data
       i=i+1;
   end

   temp = 0;
   temp_1 = 0;
   temp_2 = 0;%Initialise costs to 0
   i = 1; %Initialise i
   n=length(Data_Fast_Park); % Number of data samples
   while (i <= n) % Sample each datapoint
   fitness_1           =           ((Data_Slow_Current(i)           -
Sim_Slow_Current(i))^2+(Data_Fast_Current(i)                        -
Sim_Fast_Current(i))^2); % Cost Function for the current
   fitness_2           =             ((Data_Slow_Park(i)            -
Sim_Slow_Park(i))^2+(Data_Fast_Park(i) - Sim_Fast_Park(i))^2);% Cost
Function for the park switch
   temp_1 = temp_1 + fitness_1; %Summing function
   temp_2 = temp_2 + fitness_2; %Summing function
   temp = temp_1 + temp_2; %Total cost
   i = i+1; %Increment i
   end
   temp = temp/n; %Divides the total cost by the number of data
points used
   temp_1 = (temp_1)/n;
   temp_2 = (temp_2)/n;
   temp = [temp temp_1 temp_2]; %Stores total cost and cost of each
fitness function
   i=1;
   while i<= 3
       j=1;
   cost (j,i) = temp(i); %Set the cost of each chromosome
   i=i+1;
   end
   j=j+1; % Increment j
   end
```

Appendix F - Neural Network Training Data, Extended Results and Matlab Code

F.1 Torque NN training and Validation Data

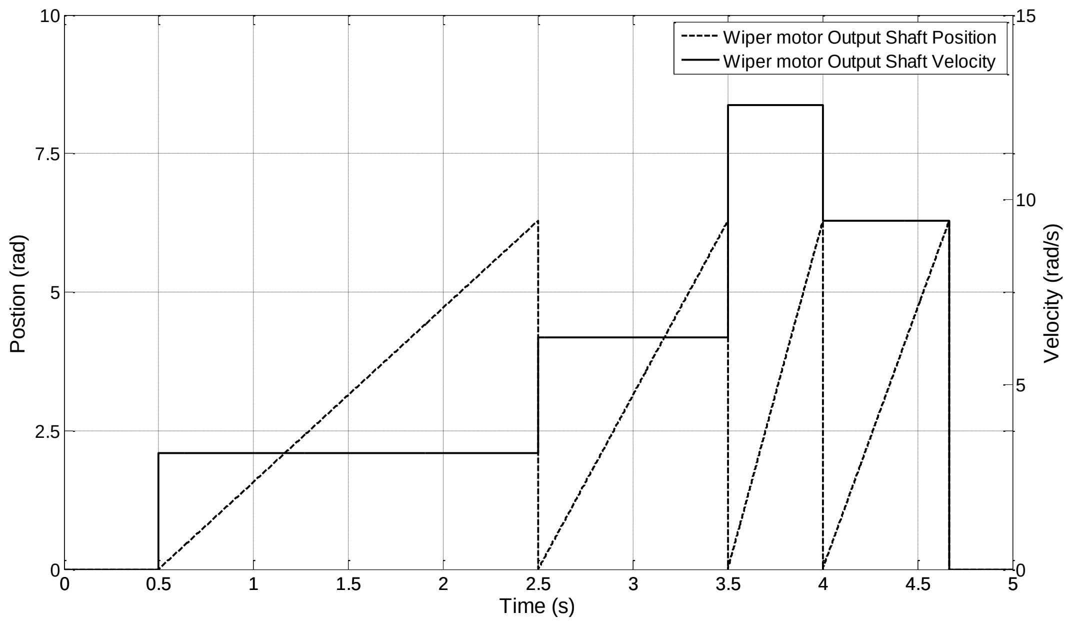


Figure F-1: Torque NN Input Training Data

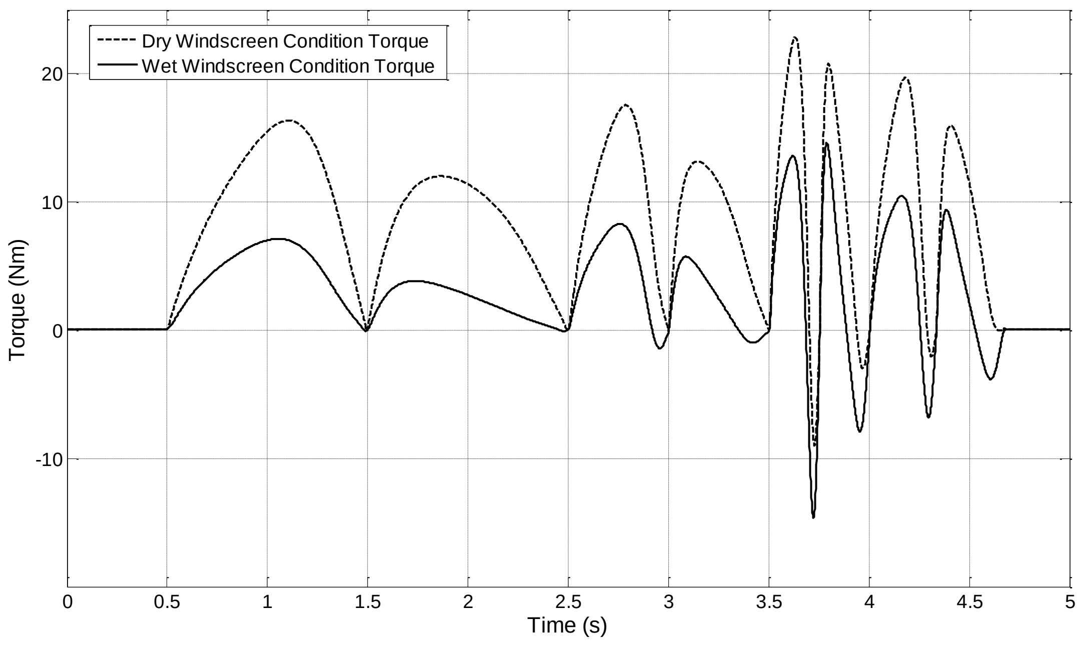


Figure F-2: Torque NN Output Training Data

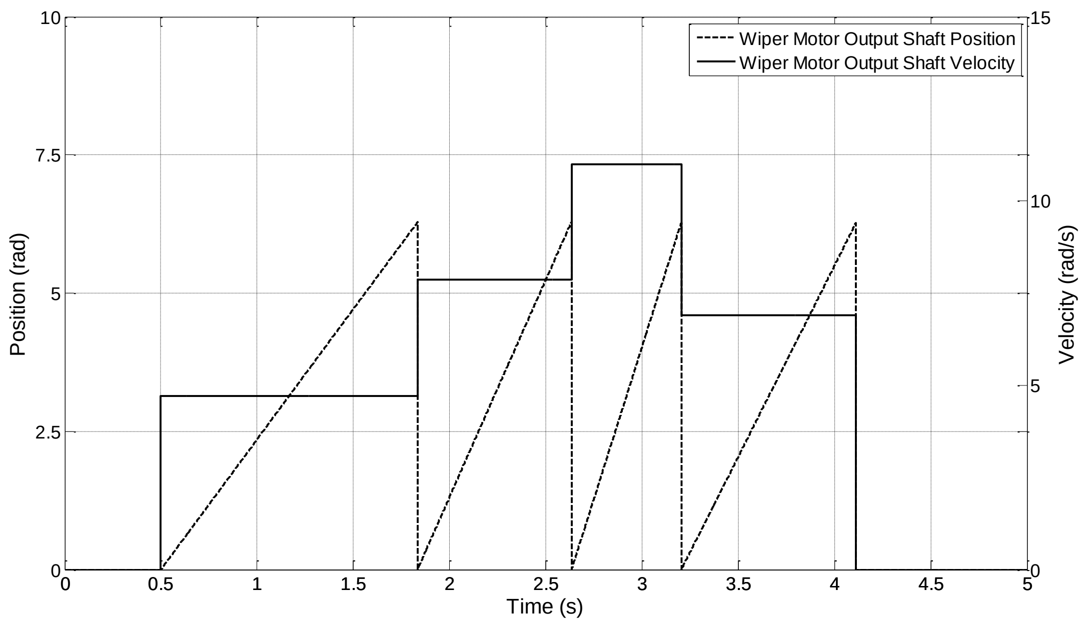


Figure F-3: Torque NN Input Validation Data

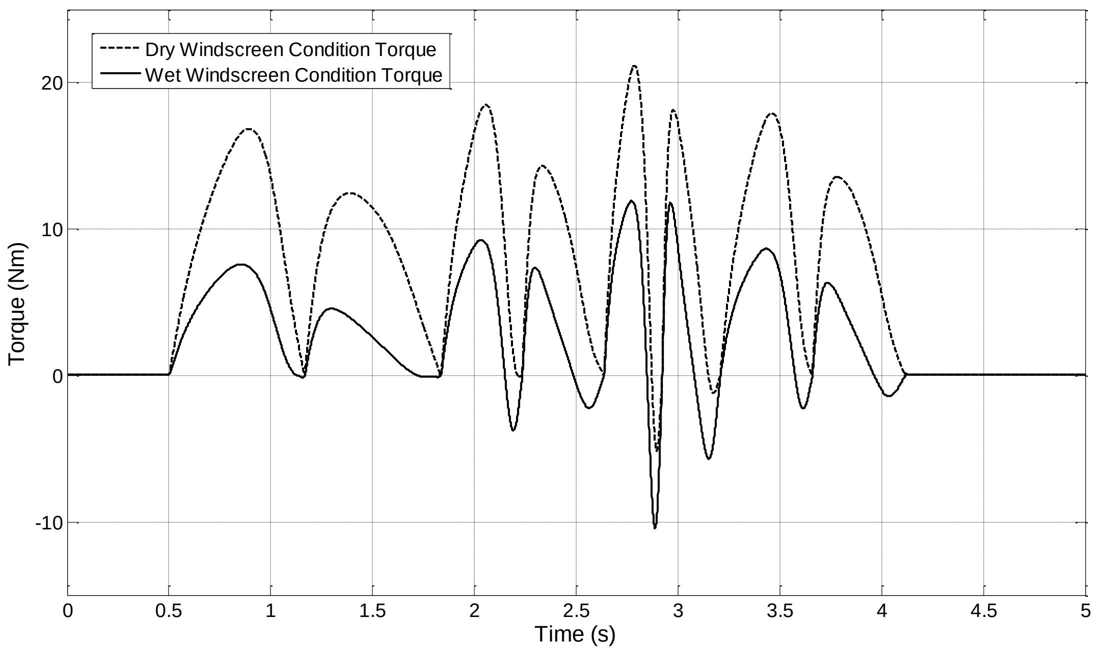


Figure F-4: Torque NN Output Validation Data

F.2 NN Hidden Layer Performance

Table F-1: NN Performance against Number of Hidden Layers (Full)

Hidden Layers	Training MSE				Validation MSE	Training Time (s)	Simulation Time (s)
	Train	Validate	Test	Total	Total		
1	24.613	25.210	25.305	24.807	26.226	0.800	0.183
	19.720	21.666	21.375	20.260	17.619	1.000	0.111
	20.236	18.660	21.432	20.179	16.448	1.000	0.094
	21.523	21.845	22.704	21.749	20.098	0.933	0.129
2	13.896	13.807	14.668	13.537	10.893	1.000	0.070
	13.994	15.873	11.455	13.895	10.882	0.800	0.088
	14.048	12.020	15.049	13.894	10.881	1.000	0.087
	13.979	13.900	13.724	13.775	10.885	0.933	0.081
3	4.418	3.945	4.832	4.409	2.480	6.000	0.089
	4.577	4.032	4.144	4.430	2.513	4.000	0.083
	4.565	4.054	4.198	4.433	2.502	4.000	0.101
	4.520	4.010	4.391	4.424	2.498	4.667	0.091
4	1.871	2.149	2.066	1.942	1.214	35.000	0.077
	3.879	3.467	4.624	3.929	2.451	1.000	0.086
	2.594	2.533	2.468	2.566	1.527	6.000	0.079
	2.781	2.716	3.052	2.812	1.731	14.000	0.081
5	10.621	10.327	11.079	10.646	15.003	4.000	0.085
	3.370	4.236	3.486	3.517	3.423	14.000	0.087
	0.884	0.887	0.766	0.867	0.541	37.000	0.095
	4.958	5.150	5.110	5.010	6.322	18.333	0.089
6	0.916	0.872	1.030	0.926	0.702	14.000	0.091
	0.550	0.532	0.577	0.551	0.400	45.000	0.087
	0.523	0.611	0.607	0.549	0.373	44.000	0.097
	0.663	0.672	0.738	0.675	0.492	34.333	0.092
7	3.059	2.939	3.351	3.085	3.866	5.000	0.107
	0.316	0.293	0.333	0.315	0.254	13.000	0.085
	1.087	1.011	1.163	1.087	4.135	9.000	0.086
	1.487	1.414	1.616	1.496	2.751	9.000	0.093
8	0.182	0.207	0.178	0.185	0.153	18.000	0.084
	2.849	2.540	2.839	2.801	5.386	7.000	0.087
	0.282	0.255	0.289	0.279	0.303	22.000	0.090
	1.104	1.001	1.102	1.088	1.948	15.667	0.087
9	0.270	0.342	0.262	0.279	2.208	9.000	0.091
	0.242	0.289	0.234	0.248	0.232	50.000	0.087
	0.143	0.147	0.153	0.145	0.102	20.000	0.086
	0.218	0.259	0.216	0.224	0.847	26.333	0.088
10	0.146	0.138	0.148	0.145	0.163	21.000	0.099

	0.099	0.093	0.104	0.099	0.134	12.000	0.139
	0.405	0.333	0.324	0.382	0.924	2.000	0.090
	0.217	0.188	0.192	0.209	0.407	11.667	0.110
11	0.181	0.163	0.265	0.191	1.935	9.000	0.091
	0.340	0.303	0.322	0.332	74.242	24.000	0.109
	0.103	0.104	0.119	0.106	0.113	51.000	0.113
	0.208	0.190	0.235	0.209	25.430	28.000	0.104
12	0.380	0.374	0.417	0.384	335.523	6.000	0.106
	0.045	0.045	0.050	0.046	0.044	21.000	0.097
	0.562	0.674	0.692	0.598	59.613	18.000	0.102
	0.329	0.364	0.386	0.343	131.727	15.000	0.102
13	0.355	0.303	0.328	0.328	1058.300	43.000	0.093
	0.413	0.448	0.418	0.419	3.053	4.000	0.099
	0.335	0.276	0.393	0.335	18.110	17.000	0.094
	0.368	0.342	0.380	0.360	359.821	21.333	0.095
14	0.093	0.082	0.079	0.089	3.547	28.000	0.110
	0.125	0.116	0.121	0.123	1.303	23.000	0.130
	0.158	0.193	0.165	0.164	19.592	55.000	0.083
	0.125	0.130	0.122	0.125	8.147	35.333	0.107
15	0.260	0.193	0.238	0.247	829.636	14.000	0.099
	0.012	0.014	0.014	0.012	294.867	66.000	0.096
	0.103	0.101	0.091	0.101	864.111	63.000	0.095
	0.125	0.103	0.114	0.120	662.871	47.667	0.097
16	0.041	0.040	27.826	4.208	21539.000	69.000	0.102
	0.054	0.057	0.058	0.055	497.333	31.000	0.101
	0.074	0.064	0.080	0.073	0.860	27.000	0.092
	0.056	0.054	9.321	1.445	7345.731	42.333	0.098
17	0.035	0.039	0.032	0.035	0.742	42.000	0.119
	0.082	0.105	0.072	0.084	805.695	48.000	0.112
	0.261	0.338	0.329	0.283	1.986	3.000	0.103
	0.126	0.161	0.144	0.134	269.474	31.000	0.111
18	0.079	0.076	0.075	0.078	187.142	28.000	0.108
	0.061	0.060	6.291	0.995	1561.900	20.000	0.108
	0.038	0.035	0.042	0.038	19.472	73.000	0.093
	0.059	0.057	2.136	0.370	589.505	40.333	0.103
19	0.023	0.023	0.024	0.023	578.286	40.000	0.105
	0.035	0.035	0.044	0.037	409.179	59.000	0.167
	0.028	0.032	0.032	0.029	84.163	22.000	0.088
	0.029	0.030	0.034	0.030	357.209	40.333	0.120
20	0.031	0.028	0.034	0.031	188.685	40.000	0.104
	0.061	0.088	0.220	0.089	60.733	35.000	0.158
	0.019	0.023	0.019	0.020	279.144	29.000	0.105
	0.037	0.046	0.091	0.047	176.187	34.667	0.122

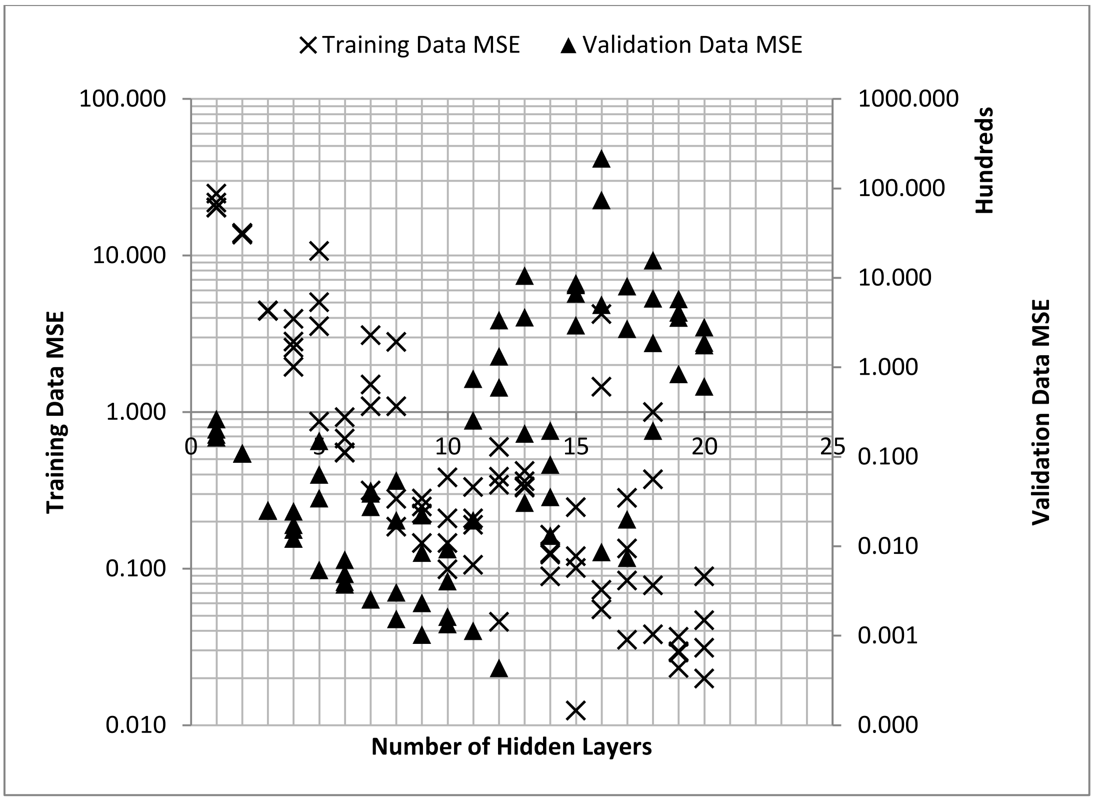


Figure F-5: NN Performance against Number of Hidden Layers (Full)

F.3 Motor NN Training Data

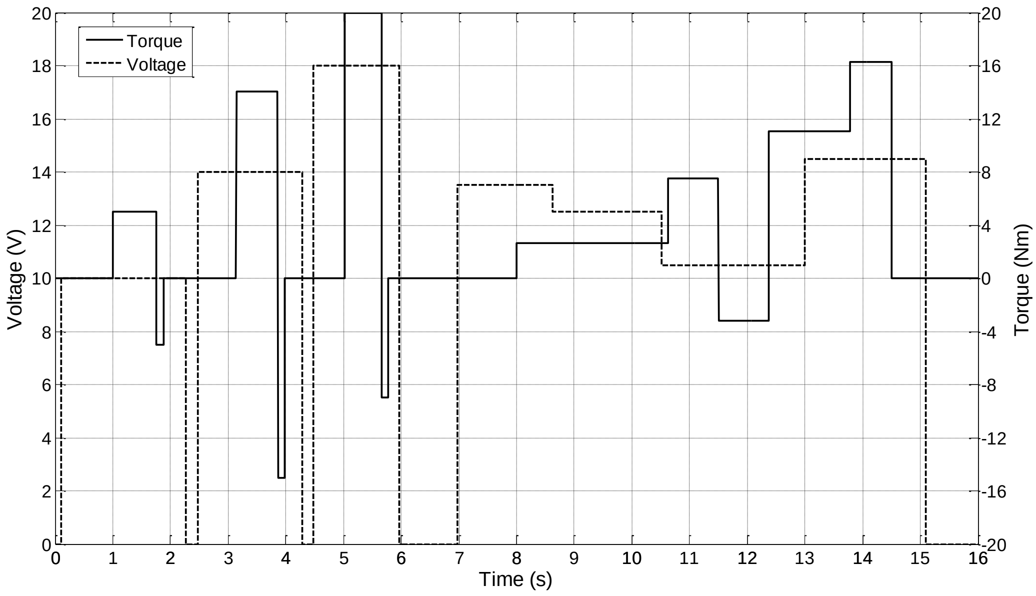


Figure F-6: Motor NN Input Training Data (Slow Motor State)

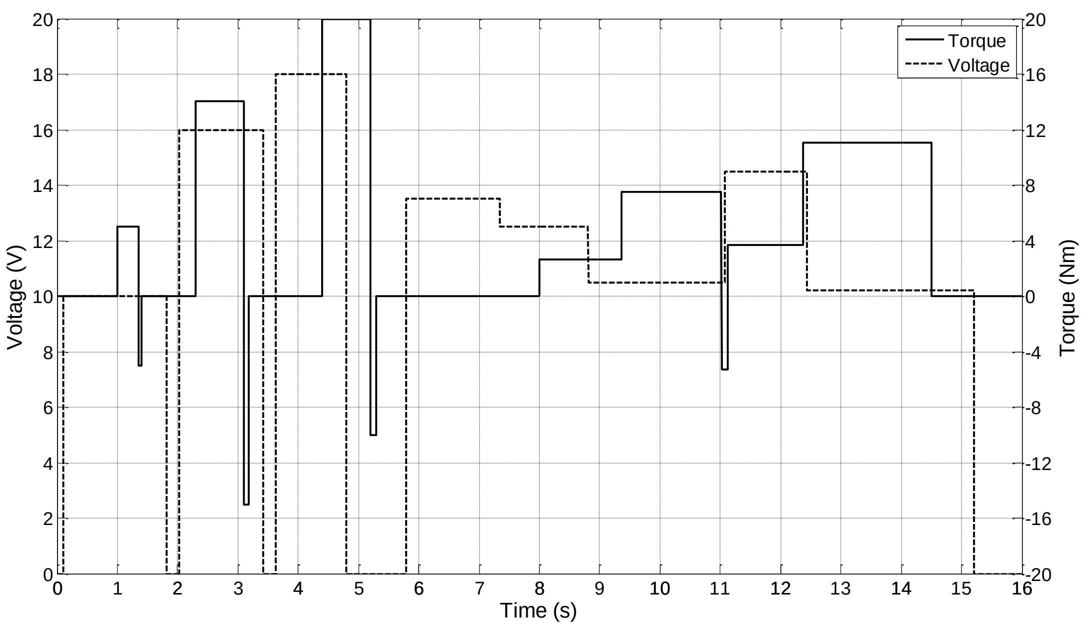


Figure F-7: Motor NN Input Training Data (Fast Motor State)

F.4 Matlab Code for Motor Neural Network

```
net = network;

net.numInputs = 2;
net.numLayers = 4;
net.sampleTime = 0.001;

%% Network Structure and connections
net.biasConnect = [1;1;1;1;];% Apply Biases to every layer
net.layerConnect = [0 0 0 0; 1 0 0 0; 0 0 0 0; 0 0 1 0]; % Connect
the layers: 1 - 2 and 3 - 4.
net.outputConnect = [0 1 0 1]; % Connect hidden layer 1 to output 1
and hidden layer 2 to output 2

net.outputs{2}.name = 'slow_out';
net.outputs{2}.feedbackInput = 2; %Index the feedback input
net.outputs{2}.feedbackDelay = 0; % No Feedback Delay (implemented
in the hidden layers)
net.outputs{2}.feedbackMode = 'open'; % Set the loop to open
net.outputs{2}.processFcns  =  {'removeconstantrows',  'mapminmax'};
%Process Signal

net.outputs{4}.name = 'fast_out';
net.outputs{4}.feedbackInput = 3; %Index the feedback input
net.outputs{4}.feedbackDelay = 0; % No Feedback Delay (implemented
in the hidden layers)
net.outputs{4}.feedbackMode = 'open'; % Set the loop to open
net.outputs{4}.processFcns          =          {'removeconstantrows',
'mapminmax'};%Process Signal

net.inputConnect= [1 0 1 0; 0 0 0 0;0 1 0 1;0 0 0 0]; %Connect the
inputs
%% Network Layers
```

```
net.layers{1}.name = 'Hidden Slow';
net.layers{1}.dimensions = 10; %Set number of Neurons in second
hidden layer
net.layers{1}.size = 10; %Set number of layers in second hidden
layer
net.layers{1}.initFcn ='initnw';
net.layers{1}.transferFcn ='tansig'; %Change function from linear

net.layers{2}.name = 'Output Slow';

net.layers{3}.name = 'Hidden Fast';
net.layers{3}.dimensions = 8; %Set number of Neurons in second
hidden layer
net.layers{3}.size = 8; %Set numer of layers in second hidden layer
net.layers{3}.initFcn ='initnw';
net.layers{3}.transferFcn ='tansig'; %Change function from linear

net.layers{4}.name = 'Output Fast';
%% Format Biases
net.biases{1}.learnFcn = 'learngdm';
net.biases{2}.learnFcn = 'learngdm';
net.biases{3}.learnFcn = 'learngdm';
net.biases{4}.learnFcn = 'learngdm';
%% Format input weights
net.inputWeights{1,1}.delays = [1 2]; % Set delays for data input
net.inputWeights{1,1}.learnFcn = 'learngdm'; % Set learning function

net.inputWeights{3,2}.delays = [1 2]; % Set delays for data input
net.inputWeights{3,2}.learnFcn = 'learngdm'; % Set learning function

net.inputWeights{1,3}.delays = [1 2]; % Set delays for data input

net.inputWeights{3,4}.delays = [1 2]; % Set delays for data input
%% Format Layer Weights
net.layerWeights{2,1}.learnFcn = 'learngdm'; % Set learning function
net.layerWeights{4,3}.learnFcn = 'learngdm'; % Set learning function

%% Assign Functions

net.adaptFcn = 'adaptwb';
net.derivFcn = 'defaultderiv';
net.divideFcn = 'divideblock'; % Divide data using the block
function
net.divideMode = 'time';
net.performFcn = 'mse' ; %Test performance using Mean Squared Error
net.plotFcns                                                       =
{'plotperform','plottrainstate','ploterrhist','plotregression','plot
response','ploterrcorr','plotinerrcorr'};
net.trainFcn = 'trainlm';

net.performParam.normalization = 'none';
```

F.5 Velocity Estimator NN training Data

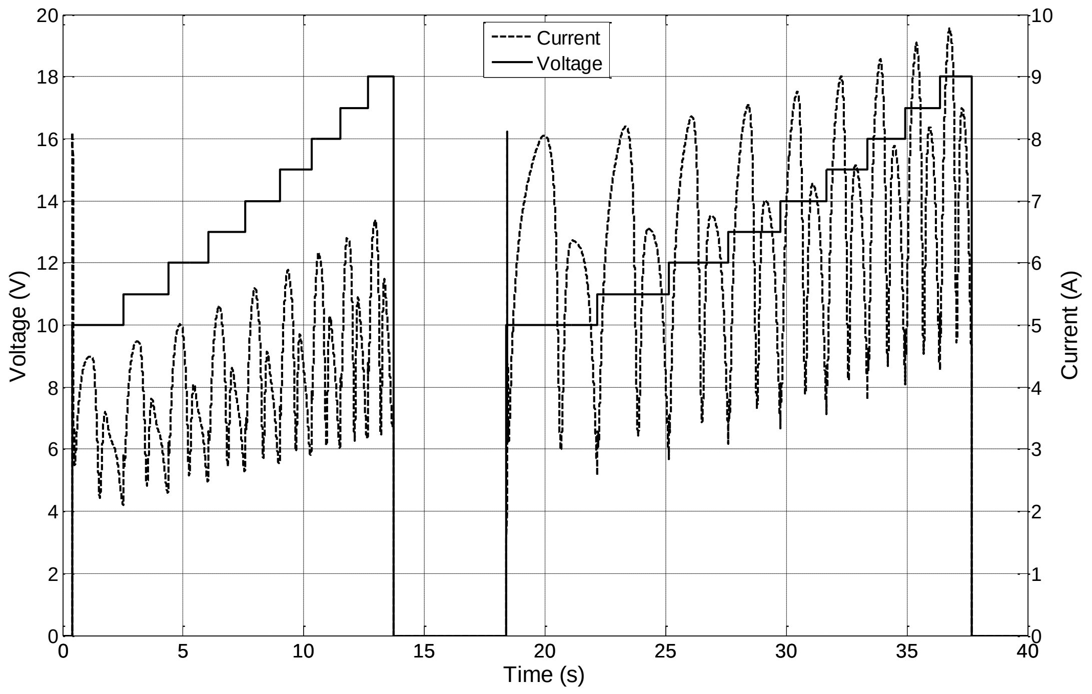


Figure F-8: Velocity NN Input Training Data (Fast)

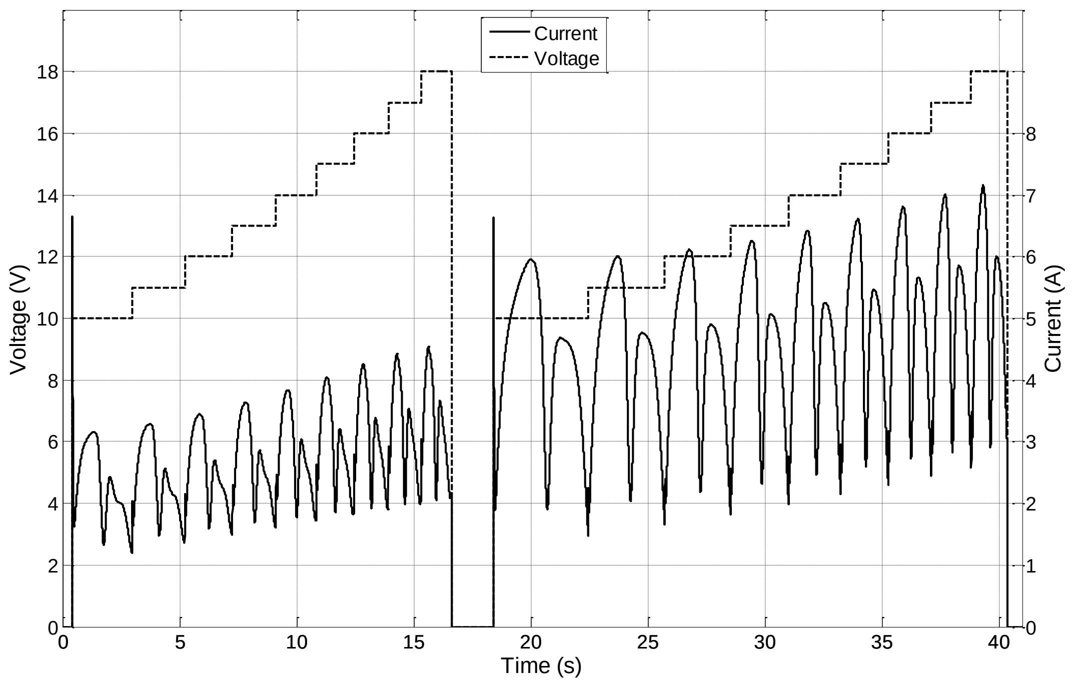


Figure F-9: Velocity NN Input Training Data (Slow)

F.6 Matlab Code for Velocity Estimator Neural Network

```
   clear net
   net = network;

   net.numInputs = 2; % 1 for each path
   net.numLayers = 4; % 1 hidden and 1 output for each path
   net.sampleTime = 0.001;

   %% Network Structure and connections
   net.biasConnect = [1;1;1;1;];% Apply Biases to every layer
   net.layerConnect = [0 0 0 0; 1 0 0 0; 0 0 0 0; 0 0 1 0]; %
Connect the layers: 1 - 2 and 3 - 4.
   net.outputConnect = [0 1 0 1]; % Connect hidden layer 1 to output
1 and hidden layer 2 to output 2
   net.inputs{1}.name = 'slow_in'; %Rename I/O
   net.inputs{2}.name = 'fast_in';
   net.outputs{2}.name = 'slow_out';
   net.outputs{4}.name = 'fast_out';
   net.inputConnect= [1 0; 0 0;0 1;0 0]; %Connect the inputs
   %% Network Layers
    net.layers{1}.name = 'Hidden Slow';
    net.layers{1}.dimensions = 10; %Set number of Neurons in second
hidden layer
    net.layers{1}.size = 10; %Set numer of layers in second hidden
layer
    net.layers{1}.initFcn ='initnw';
    net.layers{1}.transferFcn  ='tansig';  %Change  function  from
linear

    net.layers{2}.name = 'Output Slow';

    net.layers{3}.name = 'Hidden Fast';
    net.layers{3}.dimensions = 10; %Set number of Neurons in second
hidden layer
    net.layers{3}.size = 10; %Set numer of layers in second hidden
layer
    net.layers{3}.initFcn ='initnw';
    net.layers{3}.transferFcn  ='tansig';  %Change  function  from
linear

    net.layers{4}.name = 'Output Fast';
   %% Format Biases
    net.biases{1}.learnFcn = 'learngdm';
    net.biases{2}.learnFcn = 'learngdm';
    net.biases{3}.learnFcn = 'learngdm';
    net.biases{4}.learnFcn = 'learngdm';
   %% Format input weights
   net.inputWeights{1,1}.learnFcn  =  'learngdm';  %  Set  learning
function
   net.inputWeights{3,2}.learnFcn  =  'learngdm';  %  Set  learning
function
   %% Format Layer Weights
   net.layerWeights{2,1}.learnFcn  =  'learngdm';  %  Set  learning
function
   net.layerWeights{4,3}.learnFcn  =  'learngdm';  %  Set  learning
function

   %% Assign Functions
```

```
    net.inputs{1}.processFcns = {'removeconstantrows', 'mapminmax'};
%Normalise inputs/outputs
    net.inputs{2}.processFcns = {'removeconstantrows', 'mapminmax'};
    net.outputs{1}.processFcns        =       {'removeconstantrows',
'mapminmax'};
    net.outputs{2}.processFcns        =       {'removeconstantrows',
'mapminmax'};

    net.adaptFcn = 'adaptwb';
    net.derivFcn = 'defaultderiv'; %70%;15%;15%
    net.divideFcn = 'dividerand'; %Divide data randomly (no time
dependent data)
    net.divideMode = 'sample';
    net.performFcn = 'mse' ;  %Select performance measure
    net.plotFcns                                                     =
{'plotperform','plottrainstate','ploterrhist','plotregression','plot
response','ploterrcorr','plotinerrcorr'};
    net.trainFcn  =  'trainlm';  %Train  with  Levenberg-Marquardt
backpropagation algorithm

    net.performParam.normalization = 'none';
```

Appendix G - Automatic Generation of Wiper System Physical Model Matlab Code

```
   %% Generate New Simulink Model
   sys = 'Wiper_system';
   new_system(sys); open_system(sys);
   %% Select Blocks
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Get Selected Blocks From GUI
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %% Set Block Size
   x = 100;y = 100;w = 100;h = 100;offset = 300; %Sets the block
size of the model, used to position the new blocks
   %% Add Motor Block
   pos_motor = [x y x+2*w y+h];
   switch Motor
       case 'Digital Park Switch Motor'
           switch Switching;
               case  'Yes'
           pos = [x-offset y (x-offset)+2*w y+h]; %Sets the position
variable 'position' used in placing the next block
           add_block('Wiper_System_Library/Switching/Digital       PS
Switching',[sys '/Switching'],'Position',pos); %Adds a motor block
           end
           add_block('Wiper_System_Library/Wiper Motors/Digital Park
Switch Motor',[sys '/Motor'],'Position',pos_motor);
       case 'Mechanical Park Switch Motor'
           switch Switching;
               case  'Yes'
           pos     =     [x-offset     y     (x-offset)+2*w     y+h];
add_block('Wiper_System_Library/Switching/Mechanical             PS
Switching',[sys '/Switching'],'Position',pos);
           end
           add_block('Wiper_System_Library/Wiper   Motors/Mechanical
Park Switch Motor',[sys '/Motor'],'Position',pos_motor);
       case 'Depressed Mechanical Park Switch Motor'
           switch Switching;
               case  'Yes'
           pos     =     [x-offset     y     (x-offset)+2*w     y+h];
add_block('Wiper_System_Library/Switching/Depressed  Mechanical  PS
Switching',[sys '/Switching'],'Position',pos);
           end
           add_block('Wiper_System_Library/Wiper    Motors/Depressed
Mechanical Park Switch Motor',[sys '/Motor'],'Position',pos_motor);
   end
   clear pos_motor
   %% Add Linkage Block
   pos  =  [x+offset  y  (x+offset)+2*w  y+h];  %Sets  the  position
variable 'position' used in placing the next block
   switch Linkage
       case 'LHD Centre Linkage'
           add_block('Wiper_System_Library/Wiper Linkages/LHD Centre
Linkage',[sys '/Linkage'],'Position',pos); %Adds a linkage block
       case 'RHD Centre Linkage'
           add_block('Wiper_System_Library/Wiper Linkages/RHD Centre
Linkage',[sys  '/Linkage'],'Position',pos        case  'LHD  Master
Linkage'
           add_block('Wiper_System_Library/Wiper Linkages/LHD Master
Linkage',[sys '/Linkage'],'Position',pos);
```

```
        case 'RHD Master Linkage'
            add_block('Wiper_System_Library/Wiper Linkages/RHD Master
Linkage',[sys '/Linkage'],'Position',pos);
        case 'LHD Slave Linkage'
            add_block('Wiper_System_Library/Wiper  Linkages/LHD  Slave
Linkage',[sys '/Linkage'],'Position',pos);
        case 'RHD Slave Linkage'
            add_block('Wiper_System_Library/Wiper  Linkages/RHD  Slave
Linkage',[sys '/Linkage'],'Position',pos);
    end
    %% Add Arms
    switch Left_Arm
        case 'Left_Straight_Arm'
            pos = [(x+(2*offset)) y (x+(2*offset))+2*w y+h/2]; %Sets
the position variable 'position' used in placing the next block
            add_block('Wiper_System_Library/Wiper
Arms/Left_Straight_Arm',[sys '/Left Arm'],'Position',pos); %Adds an
arm block
            case 'Left_Curved_Arm'
            pos  =  [(x+(2*offset))  y  (x+(2*offset))+2*w  y+h/2];
add_block('Wiper_System_Library/Wiper     Arms/Left_Curved_Arm',[sys
'/Left Arm'],'Position',pos);
    end
    switch Right_Arm
        case 'Right_Straight_Arm'
            pos  =  [(x+(2*offset))  y+offset/4  (x+(2*offset))+2*w
(y+offset/4)+h/2              add_block('Wiper_System_Library/Wiper
Arms/Right_Straight_Arm',[sys '/Right Arm'],'Position',pos);
        case 'Right_Curved_Arm'
            pos  =  [(x+(2*offset))  y+offset/4  (x+(2*offset))+2*w
(y+offset/4)+h/2];
            add_block('Wiper_System_Library/Wiper
Arms/Right_Curved_Arm',[sys '/Right Arm'],'Position',pos);
    end
    %% Add Blades
    pos  =  [(x+(3*offset))  y  (x+(3*offset))+2*w  y+h/2];  %Sets  the
position variable 'position' used in placing the next block
    add_block('Wiper_System_Library/Wiper
Blades/Left_Wiper_Blade',[sys  '/Left  Blade'],'Position',pos);  %Adds
a blade block

    pos     =     [(x+(3*offset))     y+offset/4     (x+(3*offset))+2*w
(y+offset/4)+h/2];           add_block('Wiper_System_Library/Wiper
Blades/Right_Wiper_Blade',[sys '/Right Blade'],'Position',pos);
    %% Add Environment
    pos  =  [(x+(4*offset))  y  (x+(4*offset))+2*w  y+h/2];  %Sets  the
position variable 'position' used in placing the next block
    add_block('Wiper_System_Library/Environment/Left_Environment',[sy
s '/Left Environment'],'Position',pos); %Adds an Environment block

    pos     =     [(x+(4*offset))     y+offset/4     (x+(4*offset))+2*w
(y+offset/4)+h/2];
add_block('Wiper_System_Library/Environment/Right_Environment',[sys
'/Right Environment'],'Position',pos);

    %% Add Scopes
    pos = [(x+offset)+2.4*w  y-h  (x+offset)+2.7*w  y-h/2];  %Sets  the
position variable 'position' used in placing the next block
    add_block('built-in/Scope',[sys
'/Left_Rocker_Dynamics'],'Position',pos); %Adds scope
```

```
   pos = [(x+offset)+2.4*w y-2*h (x+offset)+2.7*w y-3*h/2]; %Sets
the position variable 'position' used in placing the next block
   add_block('built-in/Scope',[sys
'/Right_Rocker_Dynamics'],'Position',pos); %Adds scope
   %% Connect Linkages to Scopes
   add_line(sys,'Linkage/1','Left_Rocker_Dynamics/1','autorouting','
on') %Connects the left rocker to a scope
   add_line(sys,'Linkage/2','Right_Rocker_Dynamics/1','autorouting',
'on') %Connects the right rocker to a scope
   %% Get port handles of block to add connection
   MotorPortHandles = get_param('Wiper_system/Motor','PortHandles');
%Gets the port handles of the motor block
   LinkagePortHandles                                               =
get_param('Wiper_system/Linkage','PortHandles');   %Gets   the   port
handles of the linkage block
   LeftArmPortHandles         =         get_param('Wiper_system/Left
Arm','PortHandles'); %Gets the port handles of the Left Arm block
   RightArmPortHandles        =        get_param('Wiper_system/Right
Arm','PortHandles'); %Gets the port handles of the Right Arm block
   LeftBladePortHandles        =        get_param('Wiper_system/Left
Blade','PortHandles');  %Gets  the  port  handles  of  the  Left  Blade
block
   RightBladePortHandles       =       get_param('Wiper_system/Right
Blade','PortHandles');  %Gets  the  port  handles  of  the  Right  Blade
block
   LeftEnvironmentPortHandles     =     get_param('Wiper_system/Left
Environment','PortHandles');  %Gets  the  port  handles  of  the  Left
Blade block
   RightEnvironmentPortHandles    =    get_param('Wiper_system/Right
Environment','PortHandles');  %Gets  the  port  handles  of  the  Right
Blade block
   %% Connect Motor and Linkages and arms and blades
   add_line('Wiper_system',MotorPortHandles.RConn(1),LinkagePortHand
les.LConn(1),'autorouting','on');   %Connects   the   rotor   to   the
follower
   add_line('Wiper_system',MotorPortHandles.RConn(2),LinkagePortHand
les.LConn(2),'autorouting','on'); %Connects the case to the base
   add_line('Wiper_system',LinkagePortHandles.RConn(1),LeftArmPortHa
ndles.LConn(1),'autorouting','on'); %Connects the Left Rocker to the
Left Arm
   add_line('Wiper_system',LinkagePortHandles.RConn(2),RightArmPortH
andles.LConn(1),'autorouting','on');  %Connects  the  Right  Rocker  to
the Right Arm
   add_line('Wiper_system',LeftArmPortHandles.RConn(1),LeftBladePort
Handles.LConn(1),'autorouting','on');  %Connects  the  Left  Arm  to  the
Left Blade
   add_line('Wiper_system',RightArmPortHandles.RConn(1),RightBladePo
rtHandles.LConn(1),'autorouting','on');  %Connects  the  Right  Arm  to
the Right Blade
   add_line('Wiper_system',LeftBladePortHandles.Outport,LeftEnvironm
entPortHandles.Inport,'autorouting','on');  %Connects  the  Left  Blade
Vel to the Left Environment
   add_line('Wiper_system',RightBladePortHandles.Outport,RightEnviro
nmentPortHandles.Inport,'autorouting','on');   %Connects   the   Right
Blade Vel to the Right Environment
   add_line('Wiper_system',LeftEnvironmentPortHandles.Outport,LeftBl
adePortHandles.Inport,'autorouting','on');    %Connects    the    Left
Environment Force to the Left Blade
   add_line('Wiper_system',RightEnvironmentPortHandles.Outport,Right
BladePortHandles.Inport,'autorouting','on');   %Connects   the   Right
Environment Force to the Right Blade
```

```
   %% Connect Switching System and add external blocks if needed
   SW = 'Yes';
   temp = strcmp(Switching,SW);
   if temp == 1 % Checks to see if Switching is required
           SwitchingPortHandles                                    =
get_param('Wiper_system/Switching','PortHandles');  %Gets  the  port
handles of the Switching block
           pos  =  [(x-offset)-w  y-1.5*h  (x-offset)-0.5*w  y-h];
%Position of Solver Configuration
           add_block('nesl_utility/Solver        Configuration',[sys
'/Solver Configuration'],'Position',pos); %Adds Solver Configuration
           pos  =  [(x-offset)  y-h  (x-offset)+0.5*w  y-0.5*h];
%Position of Electrical reference Block
           add_block('fl_lib/Electrical/Electrical
Elements/Electrical   Reference',[sys   '/Ground'],'Position',pos);
%Adds Electrical Reference (Find blocks by clicking on them in the
library and typing gcb into matlab)
           pos = [(x-offset+0.75*w) y-1.5*h (x-offset)+1.25*w y-h];
           add_block('fl_lib/Electrical/Electrical        Sources/DC
Voltage  Source',[sys  '/VBat'],'Position',pos);  %Adds  Electrical
Reference
           set_param('Wiper_system/VBat','orientation','left')
           SolverPortHandles    =    get_param('Wiper_system/Solver
Configuration','PortHandles'); %Gets the port handles of the Solver
Configuration
           GroundPortHandles                                       =
get_param('Wiper_system/Ground','PortHandles');   %Gets   the   port
handles of the Electrical Reference
           BatteryPortHandles                                      =
get_param('Wiper_system/VBat','PortHandles'); %Gets the port handles
of the Voltage Source

add_line('Wiper_system',SolverPortHandles.RConn(1),GroundPortHandles
.LConn(1),'autorouting','on'); %Solver to Electrical Ref Connection

add_line('Wiper_system',GroundPortHandles.LConn(1),BatteryPortHandle
s.RConn(1),'autorouting','on');   %Battery   to   Electricl   Ground
Connection
       switch Motor % Wiring strategy depends on the specific motor
chosen
           case 'Digital Park Switch Motor'

add_line('Wiper_system',SwitchingPortHandles.RConn(1),MotorPortHandl
es.LConn(1),'autorouting','on'); %Fast Connection

add_line('Wiper_system',SwitchingPortHandles.RConn(2),MotorPortHandl
es.LConn(2),'autorouting','on'); %Slow Connection

add_line('Wiper_system',SwitchingPortHandles.RConn(3),MotorPortHandl
es.RConn(3),'autorouting','on'); %Common Connection

add_line('Wiper_system',SwitchingPortHandles.RConn(3),GroundPortHand
les.LConn(1),'autorouting','on'); %Common/Ground Connection

add_line('Wiper_system',SwitchingPortHandles.LConn(1),BatteryPortHan
dles.LConn(1),'autorouting','on'); %VBat Connection
           case 'Mechanical Park Switch Motor'

add_line('Wiper_system',SwitchingPortHandles.RConn(1),MotorPortHandl
es.LConn(4),'autorouting','on'); %Park Switch Connection
```

```
add_line('Wiper_system',SwitchingPortHandles.RConn(2),MotorPortHandl
es.LConn(1),'autorouting','on'); %Fast Connection

add_line('Wiper_system',SwitchingPortHandles.RConn(3),MotorPortHandl
es.LConn(2),'autorouting','on'); %Slow Connection

add_line('Wiper_system',SwitchingPortHandles.LConn(4),MotorPortHandl
es.LConn(3),'autorouting','on'); %Vbat Connection

add_line('Wiper_system',MotorPortHandles.RConn(3),GroundPortHandles.
LConn(1),'autorouting','on'); %Common/Ground Connection

add_line('Wiper_system',SwitchingPortHandles.LConn(4),BatteryPortHan
dles.LConn(1),'autorouting','on'); %VBat Connection
           case 'Depressed Mechanical Park Switch Motor'

add_line('Wiper_system',SwitchingPortHandles.RConn(2),MotorPortHandl
es.LConn(1),'autorouting','on'); %Fast Connection

add_line('Wiper_system',SwitchingPortHandles.RConn(3),MotorPortHandl
es.LConn(2),'autorouting','on'); %Slow Connection

add_line('Wiper_system',SwitchingPortHandles.RConn(3),SwitchingPortH
andles.RConn(1),'autorouting','on'); %Interconnection 1

add_line('Wiper_system',SwitchingPortHandles.RConn(4),SwitchingPortH
andles.RConn(5),'autorouting','on'); %Interconnection 2

add_line('Wiper_system',SwitchingPortHandles.RConn(4),MotorPortHandl
es.LConn(3),'autorouting','on'); %Vbat Connection

add_line('Wiper_system',SwitchingPortHandles.RConn(6),MotorPortHandl
es.LConn(5),'autorouting','on'); %A Connection

add_line('Wiper_system',SwitchingPortHandles.LConn(1),MotorPortHandl
es.LConn(6),'autorouting','on'); %B Connection

add_line('Wiper_system',SwitchingPortHandles.LConn(2),MotorPortHandl
es.RConn(3),'autorouting','on'); %Common Connection

add_line('Wiper_system',SwitchingPortHandles.LConn(3),MotorPortHandl
es.LConn(4),'autorouting','on'); %Motor Common Connection

add_line('Wiper_system',SwitchingPortHandles.RConn(4),BatteryPortHan
dles.LConn(1),'autorouting','on'); %VBat Connection

add_line('Wiper_system',SwitchingPortHandles.LConn(2),GroundPortHand
les.LConn(1),'autorouting','on'); %Common/Ground Connection
       end
   end
```