Neural Network Trained via Reinforced Learning as Control for a Battery Super-Capacitor Hybrid Energy Storage System in Electric Vehicles

by

Kevin Dueck

B.A.Sc. Hons., The University of British Columbia, 2018

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE COLLEGE OF GRADUATE STUDIES

(Electrical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

August 2020

The following individuals certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis/dissertation entitled:

Neural Network trained via Reinforced Learning as Control for a Battery Super-Capacitor Hybrid Energy Storage System in Electric Vehicles

Submitted by Kevin Dueck in partial fulfilment of the requirements of the degree of Master of Applied Science

Dr. Wilson Eberle, Faculty of Applied Science, UBC Okanagan

Supervisor

Dr. Liwei Wang, Faculty of Applied Science, UBC Okanagan

Supervisory Committee Member

Dr. Morad Abdelaziz, Faculty of Applied Science, UBC Okanagan

Supervisory Committee Member

Dr. Zheng Liu, Faculty of Applied Science, UBC Okanagan

University Examiner

Abstract

Electric vehicles (EVs) are becoming a more popular alternative to internal combustion engine vehicles, however a concern among EV manufacturers and customers is the longevity of the EV's energy source, the battery. The battery is a large contributor to the cost of an EV and is susceptible to wear due to charge/discharge cycles and heat. This wear is a main depreciator to the EV's worth. Batteries are considered a low power density energy storage device, however a hybrid energy storage system (HESS) can be formed with an improved power density by interfacing batteries and super-capacitors (SCs). An HESS utilizes the high energy density of batteries and the high power density of SCs; this lowers the wear of the batteries by directing high current transients to the capacitors and lowering the heat generated by the batteries. Proper control is imperative to developing an effective HESS that will extend the life of the batteries.

This thesis presents a novel control for a typical EV with a battery size of $24kWh$ coupled with a minimally sized HESS comprised of a SC bank with a $94.5kJ$ or $26.3Wh$ capacity using a neural network (NN) trained by a genetic algorithm. This method uses a NN to find patterns in simulated driving profiles to optimize the SCs' state of charge and SCs' current in a way that reduces the RMS current delivered by the batteries by up to 15% and reduces the peak currents by up to 52.5%. A 15% reduction in battery RMS current correlates to

a 28% reduction the thermal energy produced by the battery due to its internal resistance. This reduction in heat reduces wear on the battery and simplifies thermal management strategies for the battery. This thesis will discus the construction of such a control system, the code of the genetic algorithm, parameter selection, and the effectiveness of this solution in controlling an HESS for practical use in consumer EVs.

Lay Summary

Electric vehicles (EVs) offer many benefits over vehicles with internal combustion engines, however the battery of an EV looses energy capacity over time, resulting in a loss of driving range. This loss in capacity is due to heat generated by the battery, which degrades the battery's internals. A solution to this problem is to supplement the battery with super-capacitors which generate less heat than batteries when delivering power; combining these energy sources forms a hybrid energy storage system (HESS). To lower the battery's heat, the HESS must be properly controlled. A neural network (NN) is a computer algorithm which mimics the function of a brain and is an effective tool to recognize patterns in data that, once trained, provides an appropriate output. This thesis uses a NN trained by a genetic algorithm to control power flow in an HESS and extend the life of an EVs battery.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AC Alternating Current

ADVISOR Advanced Vehicle Simulator

ANN Artificial Neural Network

DC Direct Current

EV Electric Vehicle

FC Fuel Cell

FFT Fast Fourier Transform

FIR Finite Impulse Response

FTP-72 Federal Test Procedure-72

HESS Hybrid Energy Storage System

HNN Hammerstein-type Neural Network

ICE Internal Combustion Vehicle

IIR Infinite Impulse Response

KCL Kirchoff's Current Law

LPF Low-Pass Filter

MSI Multi-Sourced Inverter

NEDC New European Driving Cycle

NN Neural Network

PAPR Peak to Average Power Ratio

PAVR Peak to Average Velocity Ratio

PBC Passivity-Based Controller

SC Super-Capacitor

SoC State of Charge

UDDS Urban Dynamometer Driving Schedule

UC Ultra-Capacitor

List of Symbols

A	Function Constant
$A_{n\times n}x(t)_{n\times 1}$	State Space State Matrix
α	Feedback Gain
B	Function Constant
$B_{n\times m}u(t)_{m\times 1}$	State Space Input Matrix
β	Neural Network Gain
C	Capacitance
$C_{p\times n}x(t)_{n\times 1}$	State Space Output Matrix
C_{Pn}	Capacitance of pole n
C_{SC} or C_{UC}	Super-Capacitor or Ultra-Capacitor Capacitance
D	Multiplication Constant
$D_{p\times m}u(t)_{m\times 1}$	State Space Direct Transition Matrix
$\Delta i_{bat,max}$	Change in Maximum Instantaneous Battery Current
ΔV	Voltage Range
$\varepsilon_{initial}$	Randomization Factor
γ	Input Gain
i_{bat}	Instantaneous Battery Current

$i_{bat,max}$	Maximum Instantaneous Battery Current
i_L	Instantaneous Inductor Current
I_{bat}	Average Battery Current
$I_{high\ side}$	Current Sourced from High Voltage Side of Converter
I_{limit}	DC-DC Converter Current Limit
$I_{low\ side}$	Current Sourced from Low Voltage Side of Converter
I_{Load}	Average Load Current
I_{ref}	Reference Current from Neural Network
I_{RMS}	Root Mean Square Current of Simulation
I_{SC} or I_{UC}	Average Super-Capacitor or Ultra-Capacitor Current
k_T	Temperature Constant
L	Inductance
η	Converter Efficiency
P_{max}	Maximum Power
r	Resistance
R_{Pn}	Resistance of pole n
R_{SC} or R_{UC}	Super-Capacitor or Ultra-Capacitor Resistance
R_{Series}	Series Resistance
T	Temperature
T_{ref}	Reference Temperature
$\Theta_{m,n}$	Neural Network Weight Matrix
V_{buffer}	Buffer Voltage When Super-Capacitor Voltage Limit Control is Active

V_{gain}	Gain Applied to Voltage Control Signal
V_{Load}	Load Voltage
V_{min}	Minimum Voltage of Super-capacitor Bank
$V_{OC}(SOC)$	Battery Cell Voltage
V_{SC} or V_{UC}	Super-Capacitor or Ultra-Capacitor Voltage
x_n	Input
$\dot{x}(t)_{n\times 1}$	State Space State Vector
$\hat{x}$	Normalized Input
$y(t)_{p\times 1}$	State Space Output Vector
y_n	Output
$Z(S)_{Batt}$	Impedance of Battery Cell
$Z(S)_{SC}$ or $Z(S)_{UC}$	Impedance of Super-Capacitor or Ultra-Capacitor

Acknowledgements

I would like to thank my partner, Jesuina McDonald for her continued support during my studies. I would like to thank my professor Dr. Wilson Eberle for his support and guidance throughout this project. This project is funded by Ford Motor Co., without this funding the project would not be possible, so I would also like to thank their contributions. Additionally I would like to thank my colleague Tobias Lindsay for his work in starting this project of which I had the opportunity to continue. Lastly I would like to thank my family for their support, both emotional and financial throughout my education leading to and including this project.

Chapter 1

Introduction

1.1 Motivation

As climate change becomes a more recognized problem, vehicle manufactures are investing into electric vehicles (EVs) as a clean alternative to vehicles using internal combustion engines (ICEs). However, EVs are not without their drawbacks, which include charging rates, vehicle range, battery power density, and battery longevity. The later is of concern as EVs are gaining popularity among the public; lithium-ion batteries, the prominent battery used in EVs, are susceptible to heat that is generated by rapid discharging or charging of the batteries, as is the case when an EV accelerates or decelerates. Heat accelerates the degradation of batteries and as the battery ages in a vehicle, driving range is reduced. A solution to mitigate this problem is to supplement the battery of an EV with a super-capacitor (SC), or ultra-capacitor (UC) bank to form a hybrid energy storage system (HESS). The block diagram describing a SC/battery HESS is shown in figure 1.1, here the SC bank is linked to the EV's traditional powertrain components via a bi-directional DC/DC converter. Forming an HESS by adding a SC pack to a EV's battery pack can extend the life of a battery by directing potentially harmful current transients while driving to the SC bank, leaving the batteries to supply a baseline DC current. The SC pack can also aid the performance of

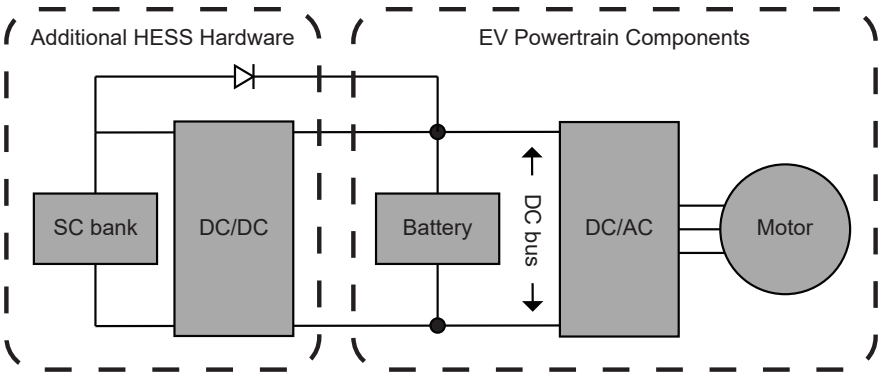


Figure 1.1: Block diagram of an active HESS where the SC bank is decoupled from a the EV powertrain components via a DC/DC converter.

an EV by increasing the available power to the motor(s) of the EV. Proper control of the DC/DC converter is critical to create an effective HESS. This thesis discusses the use of machine learning and neural networks (NNs) for control of power flow, reviews previous HESS control strategies and develops novel control utilizing a NN trained with reinforced learning by a genetic algorithm.

1.2 Control for Active Hybrid Energy Storage Systems

Each energy source of an HESS has its own advantages and disadvantages, the current sourced from either energy source must be controlled to fully utilize the advantages offered by each energy source. Effective control splits a driving current profile into subsequent current profiles to direct these profiles to either energy storage component. An ideal HESS will separate and divert high and low frequency components of the driving power profile to the energy storage element with a high-power density and the storage with the high energy density respectively, as shown in figure 1.2. Previous methods are discussed in section 2.4 and include rule-based and optimization-based control. Rule-based control is more com-

putationally simple, yet may not offer the same improvements as a more computationally taxing optimization-based control technique. This thesis will focus on optimization-based control utilizing a NN to control current flow in the HESS. The NN will solve the problem of determining when to discharge and charge the SCs and how much current to source from each energy storage component based on current vehicle and HESS conditions to obtain a SC current profile and battery current profile like those shown in figure 1.2.

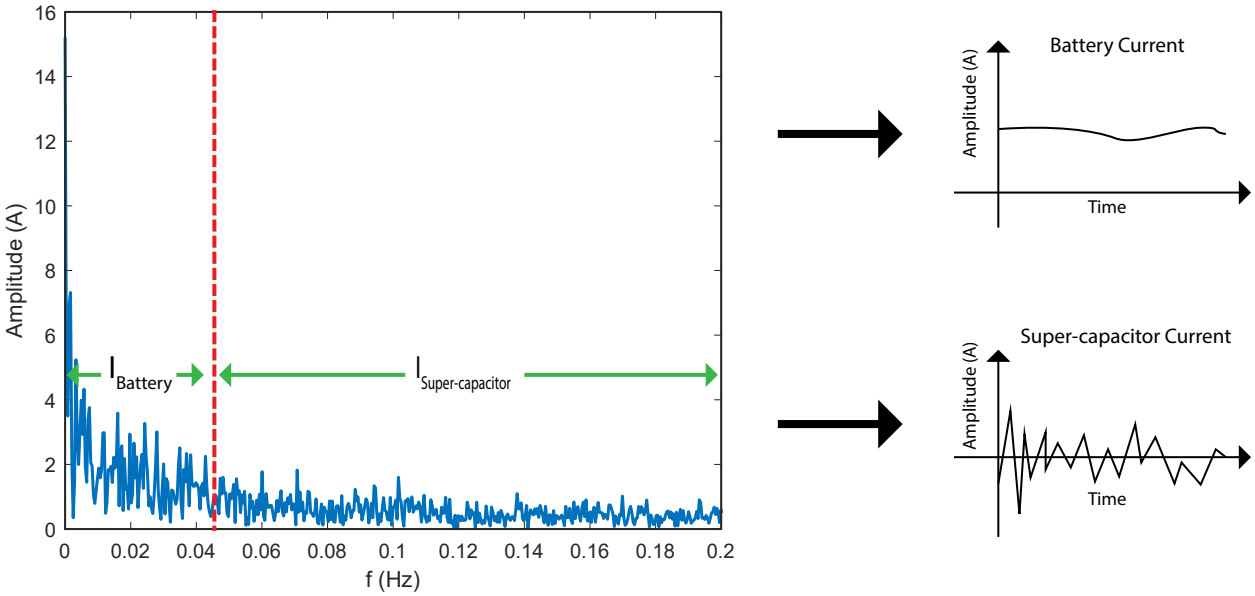


Figure 1.2: Frequency spectral analysis of an EVs load current. Effective HESS control will split into high-frequency components from low-frequency components and directed these components to the SC bank and battery bank respectively.

1.3 Operation of an Artificial Neural Network

NNs are meant to address computing problems not easily solved by conventional control solutions, such as classifying data or pattern recognition [3]. The concept of an artificial neural network (ANN), or NN, mimics the progression of a signal through neurons in a biological neural network like the brain. A biological neural network is a intricate web of

interconnections that allows a series of electrical signals to be transmitted and influence each other to produce an output. In order to construct an ANN, one must understand the operation of a biological NN. A neuron is comprised of dendrites, which receive an electrical signal, and an axon, which transmits an output signal based on the inputs the dendrites receive [3]. The actual operation of the human brain is of course much more complicated and considers many more elements, but for the sake of analogy, this understanding can be translated to ANNs. NNs can be referred to as a "connectionist" computational system, which follows no linear path when executing a problem. This is contrary to traditional linear programs which execute one line of code followed by another. Given the unpredictable nature of vehicle loading when driving, NNs are a good solution to HESS control for electric vehicles because of their ability to recognize patterns in datasets. A NN operates by modifying a

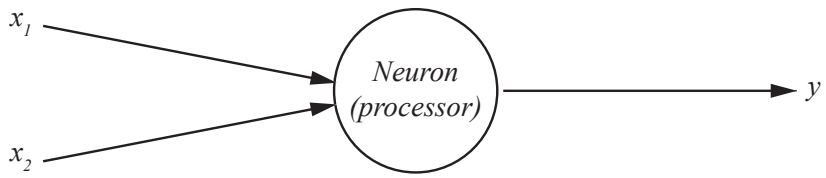


Figure 1.3: A single neuron of a NN with two inputs (x_1 and x_2) and one output (y).

signal as it passes through rows of interconnected "neurons". Four individual steps are completed as a signal progresses through each neuron for each time step of an ANN. The following steps will reference a two input neuron, as shown in figure 1.3. The first step receives both inputs to the neuron, illustrated as x_1 and x_2. Each signal input to the neuron has a weight applied to it referred to as Θ for the second step. The third step sums the weighted inputs and the fourth step generates an output by passing the summed signal through an

activation function. These steps are represented by the function shown in equation 1.1.

$$y = tanh(\Theta_1 x_1 + \Theta_2 x_2) \tag{1.1}$$

Typically an activation function for a discrete NN is the signum function which classifies the output discretely as either negative, or positive 1 given the inputs, however for the case of a linear NN, the activation function is the hyperbolic tan function; this allows a linear representation of the weights but limits the signal to ± 1. Mathematically, the inputs (x) are

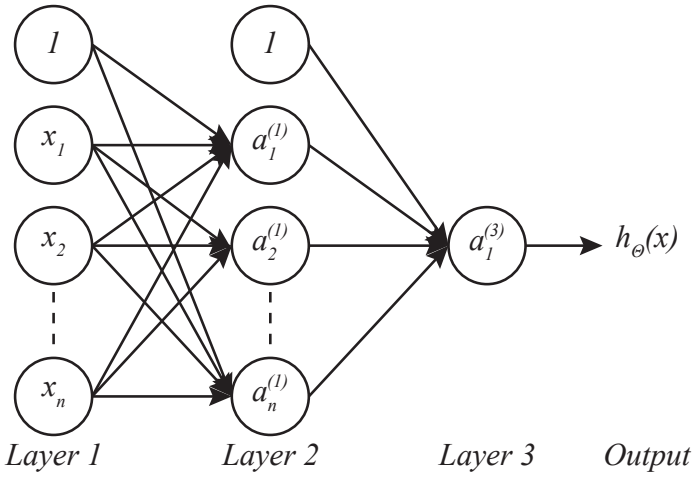


Figure 1.4: Architecture of a 3 layer ANN.

represented as a column vector and the weight function (Θ) of the neurons is represented as a matrix, shown in equation 1.2; the output of one neuron is the sigmoid ($g(z)$) function of the dot product of the inputs and the corresponding row of the weight matrix as shown in equation 1.3 and figure 1.4. Here $a_n^{(m)}$ represents the operations of one neuron, m signifies the layer and n signifies the neuron of that layer. Each arrow represents a scaling factor applied between an input and a neuron. The product is a column vector that acts as inputs to the next layer of neurons. In the column vectors, the subscript refers to the number of a

neuron in a layer and the superscript refers to layer number. The column vector result for the first layer of a network is shown in equation 1.3. Inputs to the NN include driving profile variables such as speed, acceleration, power, and parameters of energy sources like state of charge (SoC).

$$x_{n\times 1} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \Theta_{m\times n} = \begin{bmatrix} \Theta_{1,1} & \Theta_{1,2} & \dots & \Theta_{1,n} \\ \Theta_{2,1} & \Theta_{2,2} & \dots & \Theta_{2,n} \\ & \vdots & & \\ \Theta_{m,1} & \Theta_{m,2} & \dots & \Theta_{m,n} \end{bmatrix} \tag{1.2}$$

$$\begin{aligned} a_1^{(2)} &= g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + ... + \Theta_{1n}^{(1)}x_n\right) \\ a_2^{(2)} &= g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + ... + \Theta_{2n}^{(1)}x_n\right) \\ &... \\ a_n^{(2)} &= g\left(\Theta_{n0}^{(1)}x_0 + \Theta_{n1}^{(1)}x_1 + ... + \Theta_{nn}^{(1)}x_n\right) \\ h_\Theta(x) = a_1^{(3)} &= g\left(\Theta_{10}^{(2)}a_0^{(2)} + \Theta_{11}^{(2)}a_1^{(2)} + ... + \Theta_{1n}^{(2)}a_3^{(2)}\right) \end{aligned} \tag{1.3}$$

1.4 Training a Neural Network

NNs are traditionally used in conjunction with machine learning to solve classification problems. Examples of this include determining spam in an email inbox or recognizing handwriting. The output of an ANN can be made linear instead of discrete to make an ANN usable as control for an HESS. A step to building an effective NN is training the value of the weight matrices (Θ). Training a NN means adjusting the values of weights so the NN provides an effective output. These weights are trained by either supervised learning, unsupervised learning, or reinforced learning.

1.4.1 Supervised Learning

Supervised learning is a process where an ideal output for the NN is already known, and the weights are trained such that the output of the NN matches the ideal output given a set of inputs. The trained network is then used for control in environments similar to the training data set. The first step is to randomize the weights of the Θ matrices, then apply the input signals. The signal propagates through the NN and produces an output and a cost function evaluates the error of the signal, that is the difference between the desired signal and the NN's output, and produces a value. This value is then used to back-propagate through the NN and adjust the weights of the Θ matrices. This is done repetitively until the cost function is at the desired level, the trained NN is then tested on other datasets to measure its effectiveness.

1.4.2 Unsupervised Learning

Unsupervised learning is used in discrete classification NNs to determine like clusters of data. Here, the number of categories a data set can be categorized into is selected, and the algorithm groups data that are related into one of these categories. Unsupervised learning is a technique for discrete NNs is not applicable for linear NNs and HESS applications.

1.4.3 Reinforced Learning

The final learning method for NNs is reinforced learning. The optimal output is not needed for this technique, instead a genetic algorithm and Darwinian theory is used to determine weights (in this case referred to as the genes) of a NN. Reinforced learning is

comprised of three main components: heredity, variation, and selection. NN genes are tested in generations and the algorithm allows better performing genes to continue to subsequent generations while eliminating poor performing genes. The genetic algorithm operates over four main steps; the first step of a genetic algorithm is to generate a randomized population of genes. Each member of the population is run and the effectiveness of each member is recorded as a fitness value. The second step is to select members with a high fitness whose genes will transfer to subsequent generations. A matting pool is created from the population, where members with a higher fitness occupy a larger portion of the matting pool. Those with a larger portion of the mating pool have a higher probability of being selected for the next generation. The third step is reproduction, where a number of members from the mating pool are selected and their genes are combined to produce a new gene matrix. In the last step, some of the newly generated members' genes are selected at random and are given a new value in a process referred to as mutation; this prevents the algorithm from being constrained by its initial genes and converging at a local minima. This process is repeated until the fitness from one generation to the next no longer improves.

1.5 Thesis Organization

This thesis is organized into 5 chapters. Chapter 1 discusses the need for an HESS in electric vehicles, different methods of developing control for an HESS, the fundamental operation of a NN, and methods of training a NN. Chapter 2 explores previous strategies of HESS control and the use of NN for control of power flow. It compares the computational effort of other methods using the big O notation, a method to compare computational

complexity. Chapter 3 presents the design process and structure of the NN developed in this thesis, discussing the different components of the control and the method of which it is trained. Chapter 4 presents the results of the genetic algorithm training the NN and the results of the NN control on the HESS. Chapter 5 presents the conclusions, summarizes the thesis results and discusses opportunities for future work.

Chapter 2

Literature Review

2.1 Overview

This chapter reviews and analyses previous works of HESS construction and control and compares their benefits and drawbacks. This chapter's first section 2.2 discusses the purpose of an HESS as studied by others. Section 2.3 explores how an HESS is implemented with passive and various active topologies and discusses the advantages and drawbacks of each. Section 2.4 discusses various methods of control of an HESS and compares the computational effort of these types of control.

2.2 Purpose of a Hybrid Energy Storage System

An HESS is used to combine the benefits of two different energy storage devices. This is a solution not needed with ICE vehicles, as the energy and power density of fossil fuels sources is 2 orders of magnitude larger than that of batteries making it unnecessary to supplement fossil fuel with other energy sources [4]. To make EVs a compelling option compared to traditional ICE vehicles, EV manufactures are focusing on increasing the performance and longevity of their vehicles. With the increase of demanded performance comes increased stress on the batteries of an EV. Batteries produce current from an electrochemical reaction

which has inherent problems like a slow transient response to current demands and a high internal resistance as compared to SCs. The heat generated in the lithium-ion cell causes accelerated wear if not removed or reduced. A study done by [5] shows a lithium-ion battery will reduce its capacity from 90% to 60% over a period of 5 years if the operating temperature is increased from 15°C to 55°C. The battery temperature stress model is shown in equation 2.1 [5] where K_T is the temperature stress coefficient, obtained by experimentally testing specific cells, T_{ref} is a reference temperature, typically $293K$, and T is the operating temperature. This equation models the temperature dependence of the rate of a chemical reaction.

$$S_T(T) = e^{k_T(T-T_{ref})\frac{T_{ref}}{T}} \tag{2.1}$$

Figure 2.1 plots the stress factor between 280K and 330K, which is approximately between 15°C and 55°C.

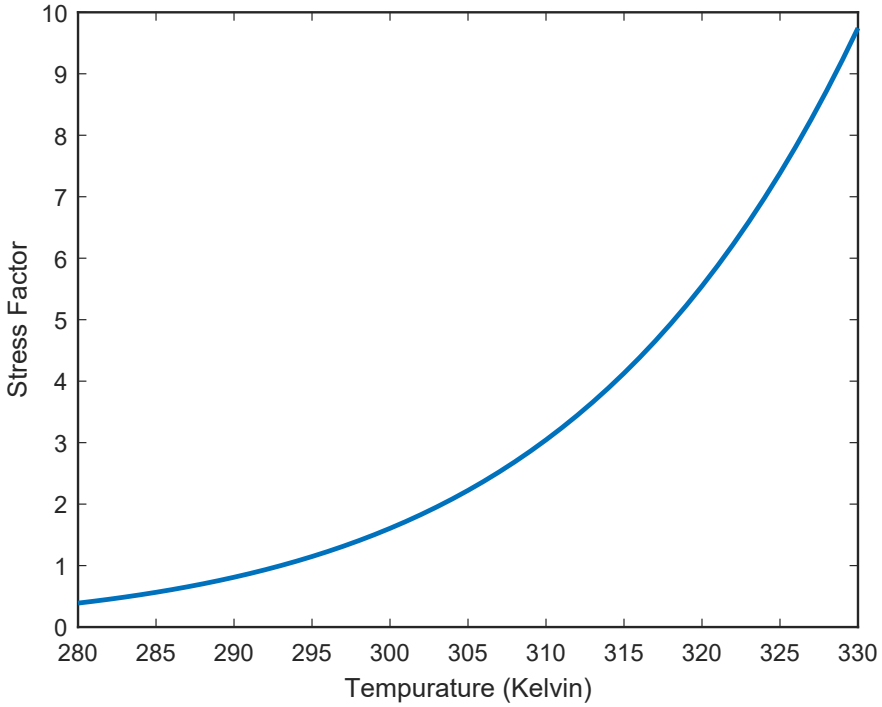


Figure 2.1: Lithium-ion battery stress factor as a function of temperature based on equation 2.1.

To compare parasitic heat loss between lithium-ion batteries and SCs, an RMS current value of $23.48A_{RMS}$, taken from the Federal Test Procedure-72 (FTP-72) driving profile [6] using an EV with a mass of $912kg$ to analyse I^2R losses. This is based on a battery pack with an operating voltage of $350V$ like that of the Nissan Leaf® which has a $24kWh$ lithium ion battery pack with 96 cells in series and 2 in parallel [7]. To quantify the heat produced by a lithium ion cell, the model of a lithium-ion battery bank can be used, shown in figure 2.2 and the formula for impedance of the battery is given by equation 2.2 [8].

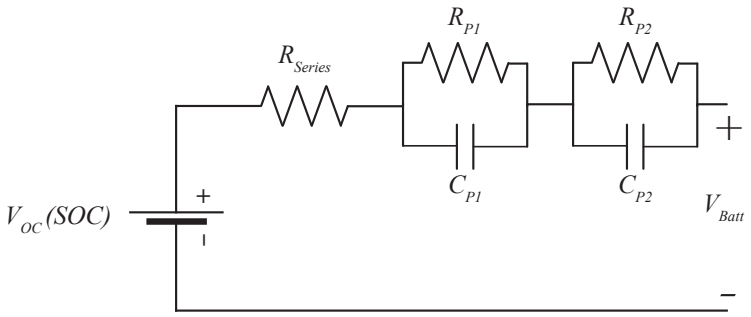


Figure 2.2: Equivalent Thevenin battery cell model.

$$Z(s)_{Batt} = R_{Series} + \frac{R_{P1}}{1 + R_{P1}C_{P1}s} + \frac{R_{P2}}{1 + R_{P2}C_{2}s} \tag{2.2}$$

Table 2.1: Specification of Li-ion battery cell model components used in equation 2.2 [1].

Parameter	$\boldsymbol{R_{Series}}$	$\boldsymbol{R_{P1}}$	$\boldsymbol{C_{P1}}$	$\boldsymbol{R_{P2}}$	$\boldsymbol{C_{P2}}$
Value	0.07 Ω	0.05 Ω	703.6 F	0.5 Ω	4475 F

In this configuration of two parallel banks of lithium-ion cells, each cell with an internal resistance of 0.62Ω [1] will produce an excess of $171W$ of heat when the battery bank supplies $23.48A_{RMS}$ of current. For comparison an SC cell with a internal resistance of $R_{UC} = 0.0022\Omega$, such as Maxwell Technologies'® BCAP0310 cell, will generate $1.2W$ of heat given

the same current demand. A model of such a cell is shown in figure 2.3 with the impedance represented in equation 2.3 and $C_{UC} = 310F$.

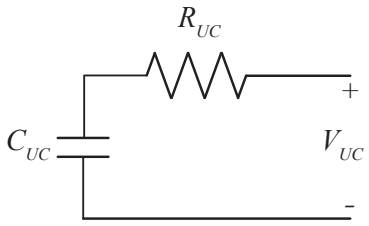


Figure 2.3: Equivalent circuit of SC cells.

$$Z(s)_{UC} = R_{UC} + \frac{1}{C_{UC}s} \tag{2.3}$$

One can see the benefits of using SCs to supply current, however because SCs store their energy in electrostatic fields they are limited by their relatively low energy density ($5Wh/kg$) compared to lithium ion batteries ($200 - 250Wh/kg$) [9]. The relationship of power density and energy density of different energy storage devices is shown in figure 2.4. The motivation

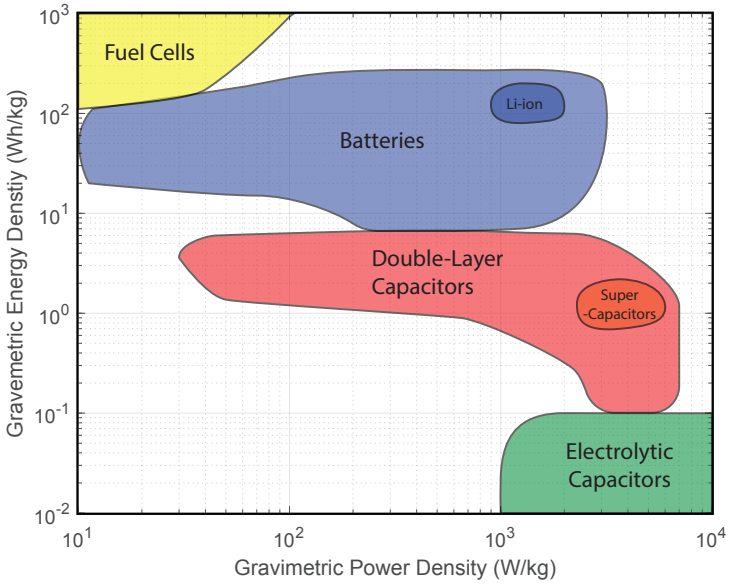


Figure 2.4: Ragone plot of different energy storage devices.

for an HESS is to combine the benefits of two energy storage devices while eliminating their drawbacks, such as combining SCs and lithium-ion batteries. Proper control ensures that harmful current transients are diverted away from the batteries to the SCs to maximise longevity of the batteries. The power density of a SC cell is much higher ($5000W/kg$) than that of a lithium-ion battery ($400 - 500W/kg$) [9] and an effective HESS utilizes the power density of an SC bank. An HESS can also aid the performance of a vehicle by lowering RMS current demanded from the battery and offering more power when accelerating, making an EV a more appealing alternative to an ICE vehicle. An HESS provides the ability to better utilize the advantages available from electric motors in EVs, whose performance has historically been limited by battery technology.

2.3 Construction of a Hybrid Energy Storage System

The topology of an HESS falls within two main categories: passive and active [10]. Active HESS topologies include subcategories with different configurations of the DC/AC inverter, DC/DC converter(s), and energy sources. Figure 2.5 illustrates these various configurations. Each topology will impact the cost of the HESS, the effectiveness of the HESS, and the performance of the EV.

2.3.1 Passive Hybrid Energy Storage Systems

The most simple HESS topology is a passive system, which is made by connecting both energy sources of the HESS to the same DC bus with no use of intermediate DC/DC converters, as represented by figure 2.5(a). This topology removes the need for an intermediate

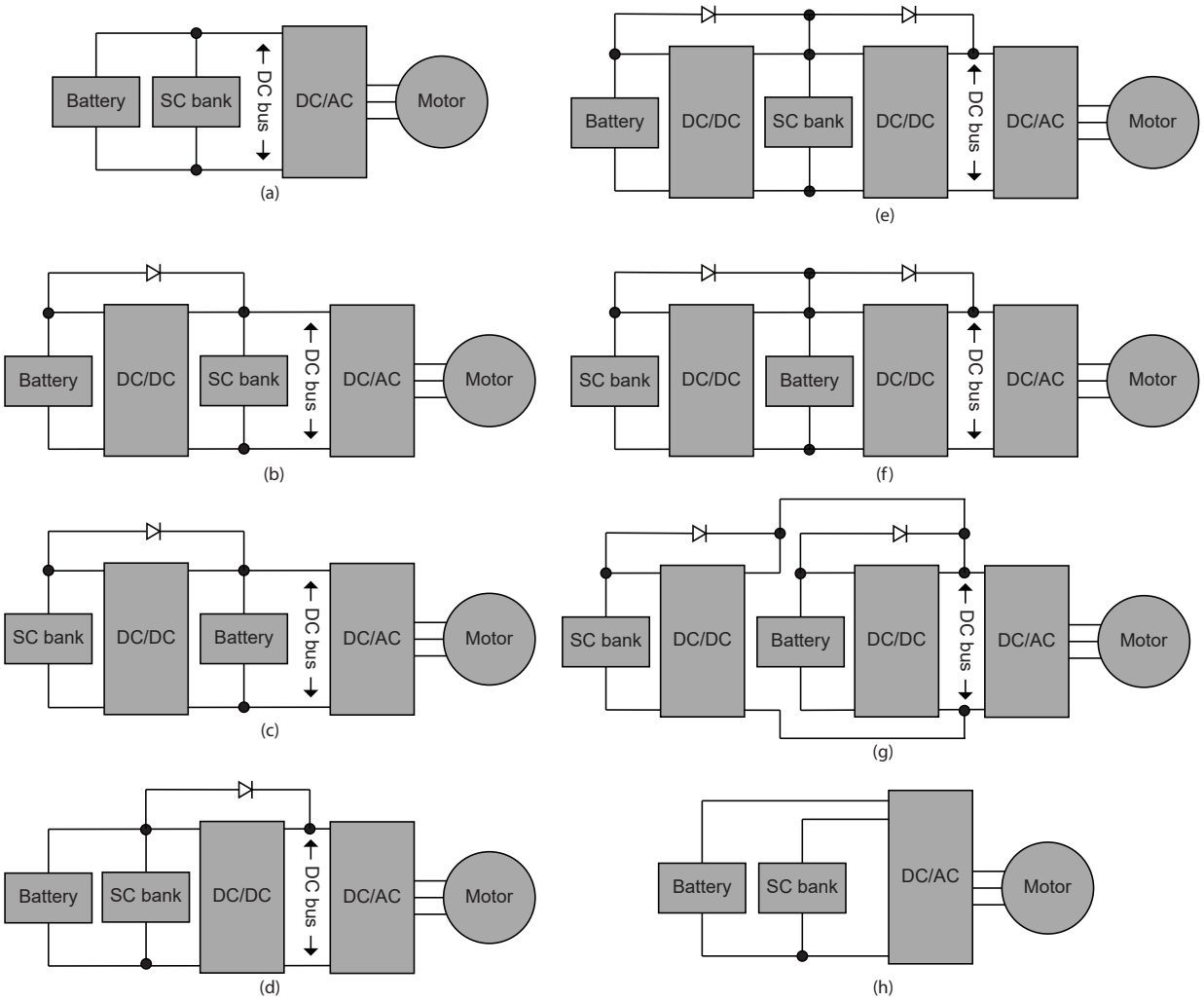


Figure 2.5: Different configurations of interfacing the battery and SC bank to the DC bus in a drive train: (a) Direct connection of battery and SC bank to the DC bus (passive control), (b) Partially-decoupled configuration, type I, (c) Partially-decoupled configuration, type II, (d) Fully-decoupled configuration of parallel-connected battery and SC bank, (e) Cascaded configuration type I, (f) Cascaded configuration, type II, (g) Parallel converter configuration, and (h) integrated configuration.

converter(s), control software and hardware, and the costs associated with these components. A passive HESS relies on the inherent impedances of each source to control power flow [10]. In the case of a SC–lithium-ion battery HESS, the relatively higher impedance of the batteries compared to the SCs, as shown in formula 2.2 and 2.3, will direct high frequency load currents to the SC bank, and low frequency components to the battery. However this

technique does have drawbacks which make it unideal for use. In this topology the capacity of the SC bank is not fully utilized because the ΔV of the SC bank is constrained by the voltage drop of the impedance of the batteries.

2.3.2 Active Hybrid Energy Storage Systems

A more effective HESS system is an active HESS which uses DC/DC converters to direct power flow. In the case of an active HESS, power flow from each energy source is controlled by either one or two converters. This allows the SC bank to fully cycle between its minimum and maximum SoC, fully utilizing the energy available in the SC bank. Active HESS topologies can be categorized as either partially decoupled, fully decoupled, or integrated.

Partially Decoupled Topologies

A partially decoupled HESS has one energy source connected to the DC bus of the inverter and one energy source connected to the DC bus via a DC/DC converter. In this way, power from both energy sources can be controlled independently; the source connected to the converter is controlled directly and the other energy source is controlled via Kirchhoff's current law (KCL), see formula 2.4.

$$I_{load} = I_{SC} + I_{bat} \tag{2.4}$$

Examples of partially decoupled systems are shown in figure 2.5 (b) and (c). An HESS where the battery is decoupled from the DC bus will make the battery immune to large fluctuations in DC bus voltage, however the DC bus voltage must vary to charge and discharge the SC

bank. This is a disadvantage, as this will vary the available peak voltage that can be applied to the motor through the inverter, unless a inverter with boost capabilities is used or the voltage rating of the motor is less than that of the minimum voltage of the SC bank [10]. If the SCs are decoupled, the SCs can be fully cycled without affecting the inverters peak voltage, however the batteries are exposed to small voltage fluctuations on the DC bus. SC decoupled HESSs are a popular topology, and are done by [8, 11–16]. Partially decoupled systems greatly improve the effectiveness of an HESS however some elements are not controlled.

Fully Decoupled Topologies

A fully decoupled system separates both energy sources from the DC bus through use of DC/DC converters. Figure 2.5 (d) shows a fully decoupled HESS topology that provides a DC bus voltage with no fluctuations, however the battery is exposed to voltage fluctuations of the SCs. While this provides the advantages of fully decoupling the energy sources, this topology shares disadvantages of the passive HESS and as such is not a common implementation of an HESS. Cascaded topologies is another fully decoupled topology option where converters separate each energy source from the DC bus and each other, represented in figures 2.5 (e) and (f). These topologies offer the ability to directly control current from each source, however they require control to consider the power flow from each source, and are susceptible to instabilities "as they can represent a DC/DC converter with a constant power load, thus requiring significant amount of care" [10]. The voltage of the energy sources must be designed so that the diodes parallel to each converter are reverse biased; that is, the energy source furthest removed from the inverter must have a voltage lower than the intermediate energy

source. Another fully decoupled system connects both energy sources in parallel to the DC bus, as represented in figure 2.5 (g). In this, the voltage of the battery or SC is not constrained to any value, however similar to cascaded topologies, control considering current flow from each energy source must be developed to ensure a stable system. Fully decoupled topologies are chosen by the authors of the papers: [2, 15].

Integrated Topologies

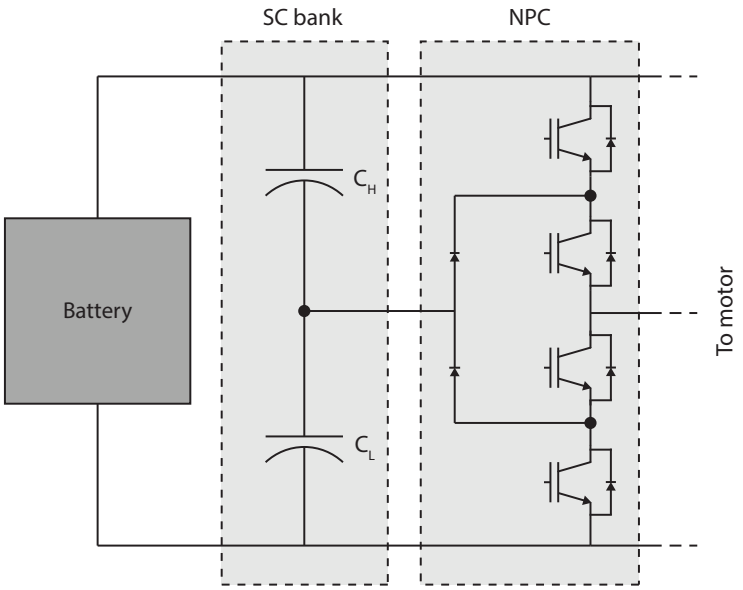


Figure 2.6: Schematic representation of an NPC used in an integrated HESS topology.

A different approach to a HESS is to control power flow from each energy source with the inverter used to supply the EVs drive motor. An example of this is done by [17], where a neutral point clamped converter is used, and the capacitors are connected to the midpoint of the converter, as shown in figure 2.6. This configuration allows a wide range of voltages for the SC, offering the advantages of an active HESS without the need of an additional converter [17].

A multi-sourced inverter (MSI) topology, demonstrated by [18], controls power flow from

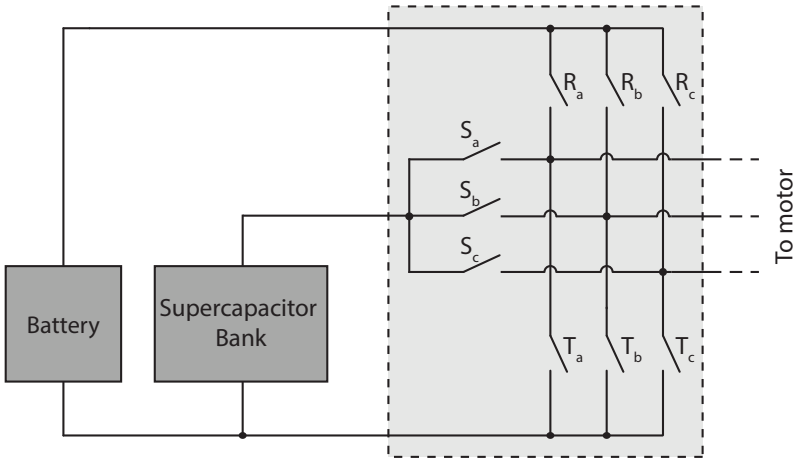


Figure 2.7: Schematic representation of an MSI used in an integrated HESS topology.

either SCs or a lithium-ion battery. The MSI topology is shown in figure 2.7. In this topology, the two sources are connected to their own respective high side switches, and both share low side switches. The MSI powers the induction motor which drives the EV. The motivation for integrated HESS systems is to reduce cost compared to other active HESS topologies.

2.4 Control of a Hybrid Energy Storage System

Effective control is needed to fully utilize the advantages provided by active HESSs. The fundamental concept of an HESS is to split the load current profile into high-frequency transients and a low-frequency baseline power. High-frequency load components will be directed to the energy source with higher specific power density, and the higher specific energy density energy source will supply the baseline power. Figure 2.8 describes such an energy split for an HESS, where the top figure shows the load current over the simulation length in the time domain, and the bottom figure shows the results of a fast Fourier transform (FFT) done over the length of the top figure. High-frequency and low-frequency current waveforms are directed to the SCs and battery respectively through use of an ideal low-pass filter (LPF).

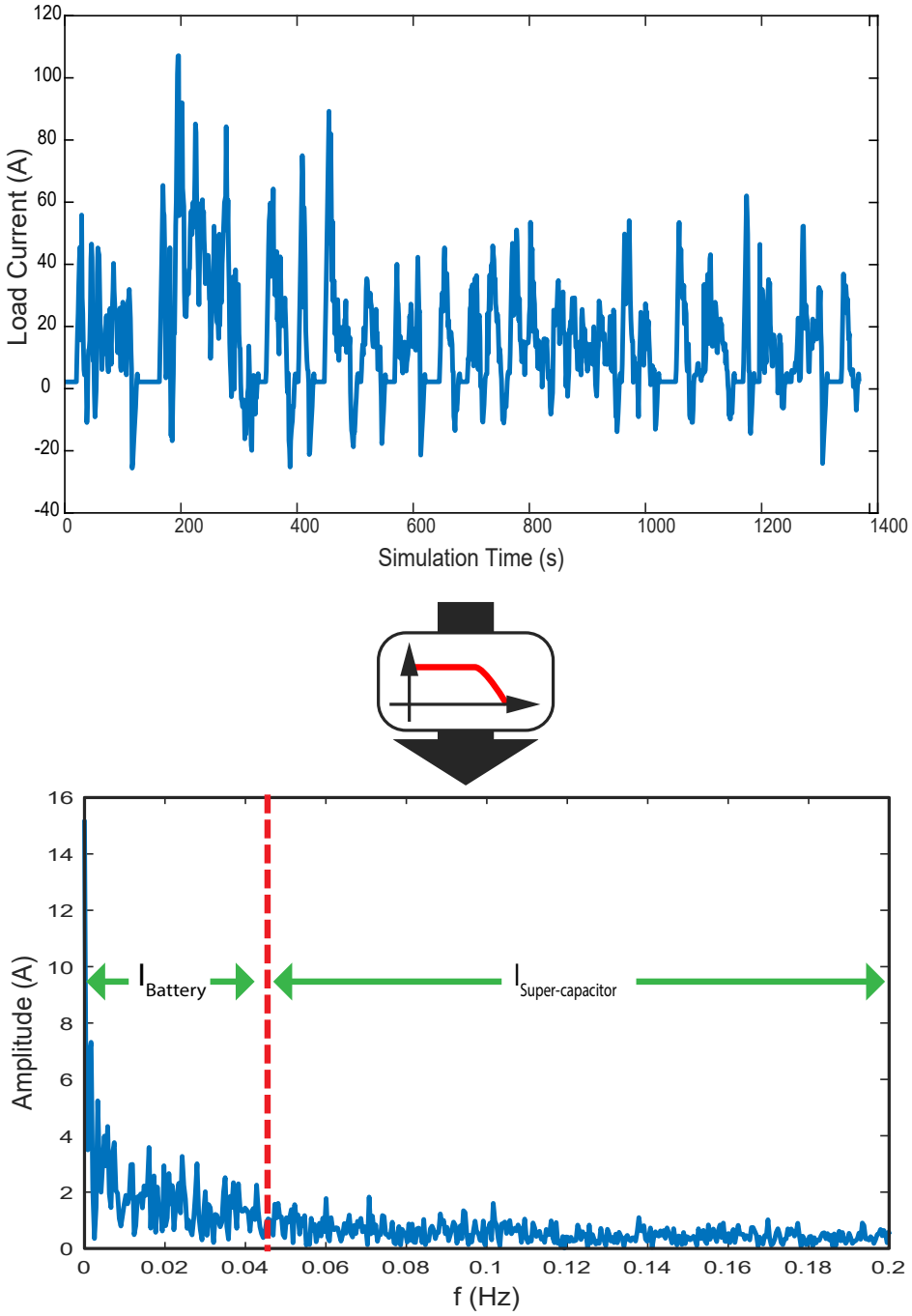


Figure 2.8: Power split between components of an HESS (top graph: time analysis of load current profile, lower graph: FFT applied to the top graph with a period of the simulation length). The split frequencies of the bottom graph represent frequency the sharing plan for LPF control with high-frequency waveforms directed to the SCs and low-frequency waveforms directed to the battery.

Control for an HESS can be categorized as rule-based or optimization-based. Rule-based requires less computational effort than optimization-based at the cost of effectiveness. In

computer science, computational load of an algorithm is represented in big O notation and is referred to as complexity. This notation will be used to compare computational requirements of discussed control solutions; $O(1)$ represents a problem where execution time is constant, $O(n)$ represents a problem where execution time is linear with input size, $O(n\ log\ n)$ represents a problem where complexity is mostly linear with a logarithmic element, and $O(n^2)$ represents a problem where execution time has a quadratic relationship with input size. The complexity of the control solutions explored in this chapter can be described by one of these big O notations, and a figure comparing these notations is shown in figure 2.9, where complexity is a function of input size for each big O representation.

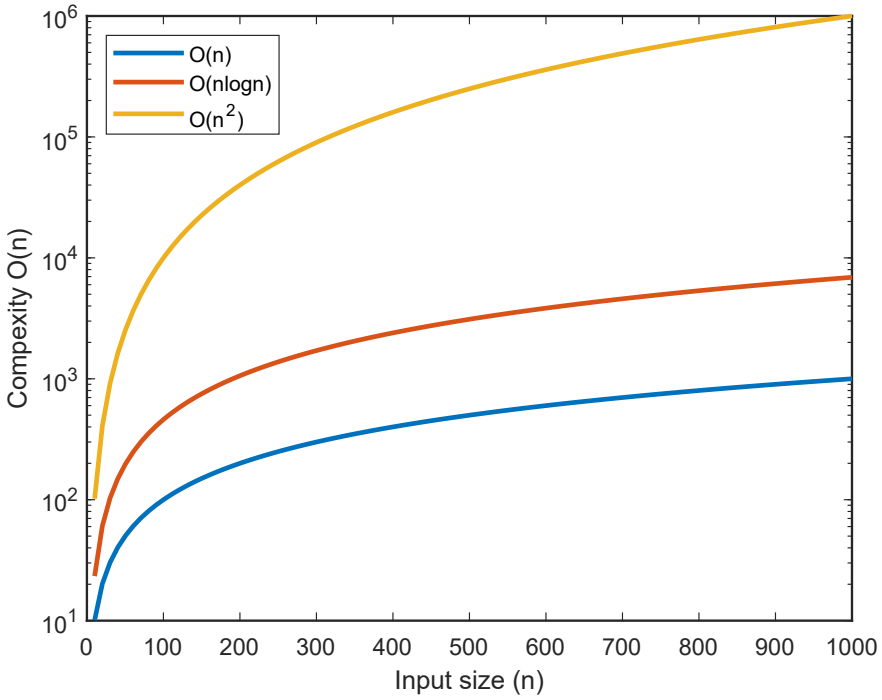


Figure 2.9: Semilog plot comparing the computational complexity of $O(n)$, $O(n\ log\ n)$, and $O(n^2)$ with input size.

2.4.1 Rule-Based Control

Rule-based control is one where an algorithm has a specific output for given inputs and approaches are either "deterministic or fuzzy type" [10]. Rule-based control "offer(s) feasibility of real-time implementation as well as robustness to the system uncertainties" [8]. An advantage of this control option is the requirement for little computational effort, making it an appealing choice for low-cost consumer vehicles. Rule-based control is used with the NPC converter, as explored by [17] which consisted of three control loops. The function of the 3 control loops are as follows: the outer most loop controls DC-link energy used to optimize the energy in the SCs based on the vehicles speed; the middle loop creates a reference DC-link voltage based on a reference battery current; and the inner loop creates an optimized DC-link current profile, which indirectly controls the SoC of the SCs. This provides effective control, however it is unique to the NPC topology used in this instance.

Another method of rule-based control is to maintain a constant battery current and vary the SCs' current based on the vehicles needs. This was implemented with a cascaded converter HESS comprised of SCs and lithium-ion batteries (figure 2.5 (f)), and the parallel converter with fuel cells (FCs) and SCs (figure 2.5 (g)) [15]. In this configuration a substantial sized SC pack is needed to compensate for the large amount of energy needed to fully accelerate or decelerate in order to keep the battery voltage constant.

In another rule-based control strategy, a two part framework is used to control an HESS comprised of batteries and SCs, developed by [13]. The first part considers load elements such as vehicle dynamics, motor characteristics, and regenerative braking to generate a SC SoC reference point. The goal of which is to predict future driving profiles and optimize the

SC bank to hand future power loads. The second framework optimizes power flow in the HESS by minimizing the magnitude and variation of battery power and power loss.

Rule-based control is also implemented by authors [11], where control is designed around a minimally sized SC bank. The control algorithm consists of two separate layers: the top layer "determines the real-time optimal energy to be shared between the two energy sources, while maintaining maximum EV battery SoC, and adjust SC SoC to react to future power requests" [11], the lower layer regulates power between the two energy sources and the inverter, based on the power flow determined by the upper layer. The author claims a RMS battery current reduction of 19% and a max peak battery current reduction of 50% with an SC bank with an energy capacity of $64.7kJ$ or $18.0Wh$.

Another example of rule-based control is done by [19], where an algorithm is developed around a flow chart considering power required by the electric motor, voltage of the SCs, and power available from the SCs. The output is one of four scenarios which either charge the SCs, discharge the SCs with assistance from the battery, use the SCs to fully supply the DC motors current needs, or disconnect the SCs from the DC bus. This HESS uses a large SC bank with a capacity of $176kJ$ or $48.9Wh$ which reduces battery current by 56.25%.

Authors [20] develop a fully decoupled HESS for use in trains with FCs, batteries, and SCs and use simple feedback loops to determine the power flow from each component. The feedback loop uses PI controllers to manage power flow, which is determined from a operation mode controller. This controller operates in three modes including discharge mode, charge mode, and fast charge mode. Since FCs have a very slow response, the battery and SCs absorb driving transients.

Rule-based control is effective control for an HESS using low computational power; typi-

cally rule-based control requires comparisons or subtractions to determine error and progress through an algorithm. The complexity for a subtraction operations is linear or $O(n)$. Rule-based control is simple to implement and execute but may not produce the best outcome due to unpredictable nature of a vehicles driving profile and limitations in algorithm flexibility as compared to optimization-based control.

2.4.2 Optimization-Based Control

Optimization-based control is typically more computational taxing than rule-based control. Previously, optimization-based control may not have been feasible, but as computational hardware has grown more powerful and cheaper, optimization-based control is becoming more researched. Optimization systems are constructed by developing control around a known drive cycle and developing or optimising the control for the given set of data. The driving cycle is chosen such that it will represent the expected range of environments a vehicle is subject too. A number of various control strategies for optimization-based control have been explored, one example is the wavelet-fuzzy power allocation strategy to control an HESS, implemented by [21]. This control consists of two parts; the first part, wavelet, refers to wavelet transform which is a relatively new tool in signal processing and the novel idea presented by the authors. Wavelet transform is the sum of signals that are scaled and shifted versions of the mother wavelet over time, this paper uses a method known as discrete wavelet transform and the Mallat algorithm to split the load power between SCs and batteries. Secondary control is a rule-based fuzzy logic controller. The authors claim that this control is not as computationally heavy as previous optimization control strategies due to the use of the Mallat algorithm.

Another optimization-based control method is a passivity-based controller (PBC). A PBC controller is a state space controller that achieves closed loop stability "with respect to the storage function which has a minimum at the desired equilibrium point" [22]. While this method has been implemented in the past, the authors use a Kalman filter to reduce the number of required sensors, simplifying the system.

Chaos-synchronization is a more elaborate method of optimization-based control, employed by [23]. The control is developed by modelling the SC, battery, and converter as a state space system, then a robust adaptive controller is developed which automatically regulates and synchronizes the chaotic systems. The systems are referred to as chaotic due to the fact that future driving patterns are not cyclic and follow no predictive behaviour.

The above optimization-based methods use state space systems to provide control. Big O notation can be used to estimate the computational effort of these systems: complexity of matrix multiplication between matrices of size $n \times m$ and $n \times p$ is $O(nmp)$ and the complexity of matrix addition is $O(n)$ where n is the number of elements in the matrices. So for a state space system such as:

$$\begin{aligned} \dot{x}(t)_{n\times 1} &= A_{n\times n}x(t)_{n\times 1} + B_{n\times m}u(t)_{m\times 1} \\ y(t)_{p\times 1} &= C_{p\times n}x(t)_{n\times 1} + D_{p\times m}u(t)_{m\times 1} \end{aligned} \tag{2.5}$$

where n is the number of states, m is the number of inputs, and p is the number of outputs,

the complexity is represented as:

$$\begin{aligned} & O(n^2) + O(n) + O(nm) + O(pn) + O(p) + O(pm) \\ &= O(n^2 + n + nm + pn + p + pm) \\ &= O(n^2 + nm + pn + pm) \\ &\approx O(4n^2) \\ &= O(n^2) \end{aligned} \tag{2.6}$$

Here computation time is dominated by matrix multiplication, and because big O notation is upper-bound, matrix addition can be neglected. Similarly, additional quadratic terms (nm, pn, and pm) can be dropped as they are all of the same power as n^2. A state space controller like the one used by [23] will then have a complexity of $O(n^2)$.

A common implementation of optimization-based control is to use a LPF to separate power components of an HESS. Figure 2.8 shows an optimal outcome of LPF based control like that shown in figure 2.10, where a filter is used to generate current profiles for the high (SC) and low (battery) power dense components of the HESS, and outer-loop control is used to ensure SoC and currents of each source stay within their limits. Examples of LPF control for a HESS are demonstrated by [2, 8, 14–16]. The corner frequency of the LPF may be fixed or made variable to increase effectiveness. The authors of [16] and [2] determine two corner frequencies for either urban or highway driving, and select either based on the vehicle's speed. A fully variable corner frequency is done by [14] where the authors develop an algorithm to determine the optimal corner frequency based on driving conditions of a vehicle. The control diagram for LPF control similar to that used by [2] is shown in figure

2.10. This control is for a fully-decoupled system; the top control elements include the LPF

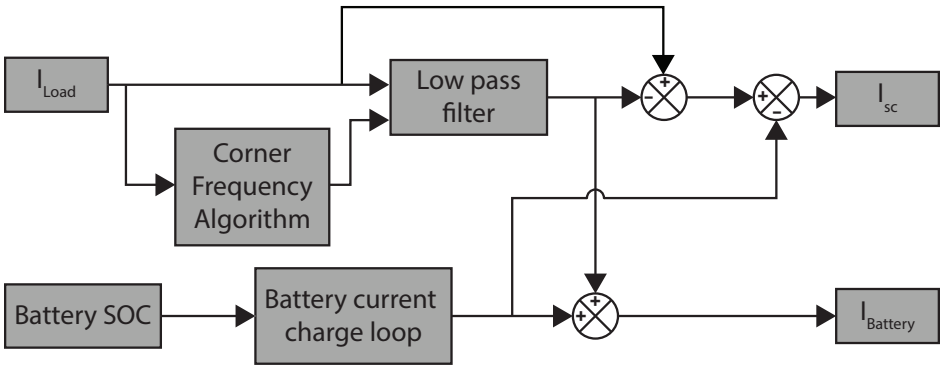


Figure 2.10: Control diagram of HESS control using an LPF with a variable corner frequency to directed load current frequency components to the batteries or SCs. Battery and SC current signals are further modified by the battery current charge loop control to consider hardware constraints (cited from [2]).

which splits the current reference between the two sources and the bottom control adjusts the reference to ensure that both currents are within their limits. The corner frequency of the LPF can be optimized for a given driving profile or based on the impedances of the SC bank, converter, and battery. Authors [8, 13] developed mathematical models of the lithium-ion battery banks, SCs, and DC/DC converter and analysing their respective impedances and determine the optimal corner frequency for LPF control. The transfer function for the battery and SC bank is shown previously in equations 2.2 and 2.3. The final component to model is the DC/DC converter that links the SCs to the DC bus. These converters operate in two different modes, buck or boost, depending on the direction of power flow in the HESS, as such two different models are required as shown in figures 2.11 and 2.12. The input impedance of these two models is represented by the equations 2.7 and 2.8 [8].

$$Z_{in}^{boost}(s) = \frac{CC_{SC}Ls^2 + (R_{SC} + r_1)CC_{SC}s + [C_{SC}(1-D)^2 + C]}{CC_{SC}s} \tag{2.7}$$

$$Z_{in}^{buck}(s) = \frac{C_{SC}Ls^2 + (R_{SC+r_1})C_{SC}s + 1}{CC_{SC}Ls^3 + (R_{SC} + r_1)C_{SC}Cs^2 + (C_{SC}D^2 + C)s} \tag{2.8}$$

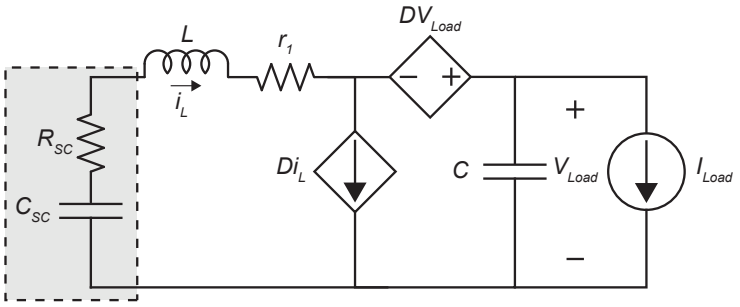


Figure 2.11: Average model of a boost converter.

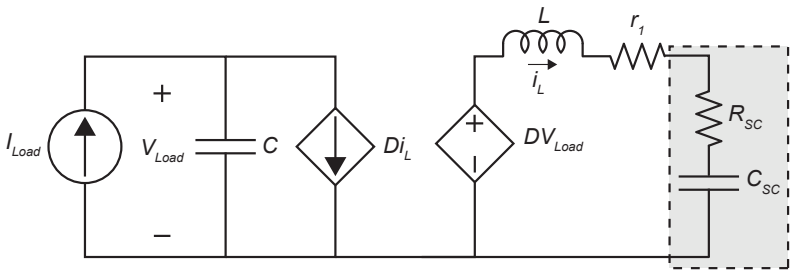


Figure 2.12: Average model of a buck converter.

The above mathematical models can then be used with the previous transfer functions represented by equations 2.2 and 2.3 to analyse the frequency response of the batteries and the SC bank/DC/DC converter pair. Once the bode plots are generated, an optimal corner frequency for the LPF can be selected for the control. LPFs can be categorized as either digital or analog, and digital filters are either finite impulse response (FIR) or infinite impulse response (IIR) filters. IIR filters offer flexibility in design with the presence of both poles and zeros in the mathematical formulation. Filters can be tested in simulation or with prototypes to determine their effectiveness and stability. Most filters are based on the FFT operation, which has a complexity of $O(n \log n)$.

2.4.3 Neural Network Control

Another form of optimization-based control is the NN. In recent years NNs have made a larger presence in control. As discussed in section 1.3, NN control algorithm loosely mimics neurons in the brain, where a signal is taken from inputs and propagates through one or more layers of neurons before generating an output signal. A NN developed by [18] where the current is controlled by a multi source inverter, similar to the integrated HESS topology demonstrated by [17]. The NN is trained with supervised learning and optimal duty cycles for a given driving profile are obtained by a "dynamic programming algorithm" [18]. The structure of the NN used has 7 inputs, 1 output, and two hidden layers, the first with 10 neurons and the second with 5. The optimization problem is discovered with the formulae 2.9 - 2.11.

$$f_1(k) = (i_{bat}^2(k)R_{bat} + i_{UC}^2(k)R_{UC})/P_{max} \tag{2.9}$$

$$f_2(k) = i_{bat}^2/i_{bat,max}^2 \tag{2.10}$$

$$f_3(k) = (i_{bat}(k) - i_{bat}(k-1))^2/(\Delta i_{bat,max}^2) \tag{2.11}$$

Here R_{UC} is the internal resistance of the UCs, R_{bat} is the internal resistance of the battery, i_{UC} is the UC's current, and i_{bat} is the batteries' current. A weighted sum of equations 2.9 - 2.11 is applied to the optimization problem, where the weights were chosen so that the batteries current magnitude and fluctuations will have the biggest influence. The resulting equation is 2.12.

$$\sum_{k=1}^{N} Af_1(k) + Bf_2(k) + Cf_3(k) \tag{2.12}$$

Using the optimal driving profile generated, the NN weights are then trained using the scaled gradient method and the performance is measured by the mean squared error. The algorithm is left to run until the error is within specified acceptable margins, then the NN weights are saved to by used by the NN in practice. The authors claim a RMS battery current reduction of 50% with a SC bank having an energy capacity of $1440kJ$ or $400Wh$.

Authors [24] develop NN control using a Hammerstein-type neural-network (HNN). This is implemented to control power flow in a large scale electrical grid HESS rather than a EV. A Hammerstein-type neural-network is a network "which formulates the Hammerstein model with a non-linear static gain in cascade with a linear dynamic block" [24]. This paper uses FCs and SCs to increase stability with load frequency control for a multi-area interconnected power system. The block diagram of a Hammerstein-type NN is shown in figure 2.13. This

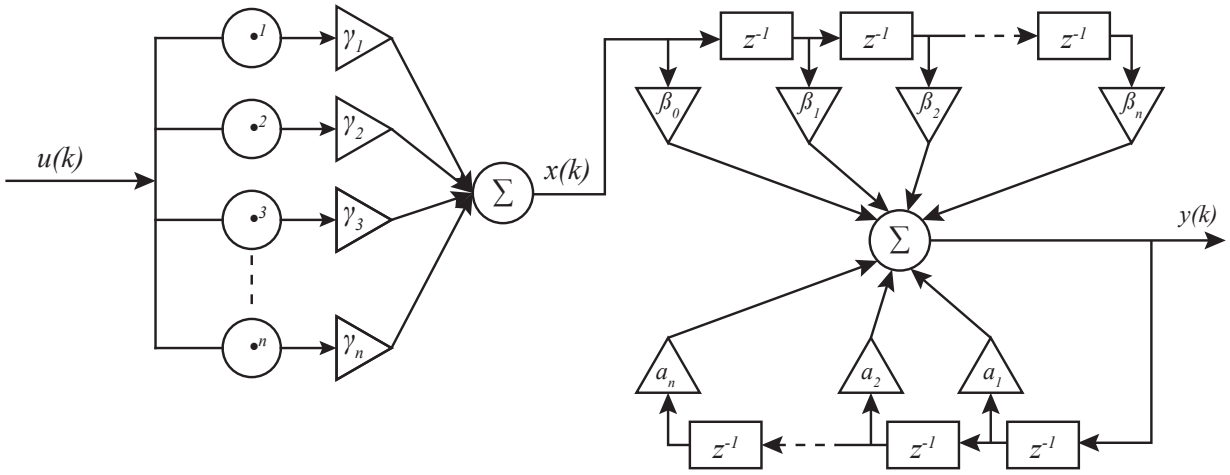


Figure 2.13: Structure of the Hammerstein-type neural network.

structure is "composed of a single dynamic node with two tapped delay lines, and these delay lines form the non-linear static element and linear dynamic element" [24]. In this scenario, the weights are trained online through reinforced learning based on a sample power profile

provided by the controller. The output of the HNN can be represented by the formula 2.13,

$$\begin{aligned}\hat{y}(k) =& -\hat{\alpha}_1\hat{(y)}(k-1) - \hat{\alpha}_2\hat{y}(k-2) - ... - \hat{\alpha}_{n_\alpha}\hat{y}(k-n_\alpha) \\ &+ \beta_0\hat{x}(k) + \hat{\beta}_1\hat{x}(k-1) + ... + \hat{\beta}_{n_\beta}\hat{x}(k-n_\beta) \\ =& -\sum_{i=1}^{n_\alpha}\hat{\alpha}_i\hat{y}(k-i) + \sum_{j=0}^{n_\beta}\hat{\beta}_j\hat{x}(k-j)\end{aligned} \tag{2.13}$$

Where the hidden layer is represented by the equation 2.14 [24].

$$\hat{x}(k) = \sum_{l=1}^{n_\gamma}\hat{\gamma}_l u^l(k) \tag{2.14}$$

The negative gradient method is applied for adjusting the HNN weights. This example demonstrates the improvements available from a NN over traditional PID control. This NN differs from [18] in that no optimal duty cycle is needed to achieve the proper output, instead the NN continually improves with use. This makes the control easier to implement, and more adaptable to various scenarios.

NNs are a compelling solution to HESS control because of their ability to generate non-linear outputs. This makes them ideal to find a solution to the abstract inputs a driving profile has. The big O notation can be used to determine the complexity of a NN by knowing the number of layers and their sizes. A 3 layer NN with n inputs, m neurons in the first layer, o neurons in the second layer, and p outputs can be represented as follows:

$$y(t)_{p\times 1} = \Theta_{3(p\times o)}\left(\Theta_{2(o\times m)}\left(\Theta_{1(m\times n+1)}x_{n+1\times 1}\right) + 1_{0,0}\right) + 1_{0,0} \tag{2.15}$$

where $+1_{0,0}$ represents an inserted bias unit. The complexity of this NN is then:

$$\begin{aligned} & O(m(n+1)+O(o(m+1))+O(p(o+1)) \\ =\ & O(m(n+1)+o(m+1)+p(o+1)) \\ \approx\ & O(3n^2) \\ =\ & O(n^2) \end{aligned} \tag{2.16}$$

Similar to state-space controllers, the complexity of a NN is $O(n^2)$.

2.5 Summary

This chapter reviewed the work of other researchers' HESS topologies and control techniques and discussed their respective advantages and disadvantages, including computational effort. Each work offers effective control of an HESS, however some are more suited to larger, more robust systems and some smaller, more practical systems. This thesis aims to explore the potential of a low cost HESS for use in consumer vehicles. The system most suited for this situation is that of an active, partially decouple topology, where the battery is directly connected to the DC bus of the vehicle, and the capacitor's power contribution is controlled via a bi-directional DC/DC converter. Having a larger capacitor bank adds weight and cost to a vehicle, so it is more practical to design a smaller SC bank and develop effective control to utilize the bank. Optimization-based control using a NN offers effective control and allows the opportunity to be trained on unique driving profiles. This combination of HESS topology and control lowers battery stress and extends the battery life of an EV, ultimately increasing its value.

Chapter 3

Neural Network Control and Genetic Algorithm

3.1 Overview

This section discusses the control developed for an active partially-decoupled HESS. Section 3.1 presents the parameters of the vehicle and elements of the HESS of which the NN will be developed around. Section 3.3 discusses the inputs to the NN, the number of layers and the number of neurons per layer (known as hyper-parameters) of the NN, and the construction of the NN. Section 3.4 discusses the control that considers SC and converter limits and conditions the signal produced by the NN. Section 3.5 discusses the genetic algorithm used to train the NN and the effect various parameters of the genetic algorithm have on the learning rate.

3.2 Modelled Hybrid Energy Storage System Parameters

The modelled HESS is based off a typical small economic 4 seat EV. The parameters of the vehicle are listed in table 3.1. The HESS topology is a partially decoupled, where the SC bank is connected to the DC bus via a bi-directional boost converter. The SC Bank is assembled from 100 series connected, $310F$ SCs like that of Maxwell's® BCAP0310 P270 T10. The result is an SC bank capacity of $3.1F$ and a total weight of $6kg$. The

Table 3.1: Parameters of the EV and modelled HESS.

Component	Value
Vehicle Weight	$912kg$
Li-ion Battery Pack Capacity	$24kWh$
Li-ion Battery Pack Nominal Voltage	$350V$
SC Bank Capacitance	$3.1F$
SC Bank Weight	$6kg$
SC Bank Rated Voltage	$270V$
SC Bank Voltage Operating Range	$50V - 250V$
SC Bank Energy Capacity	$94.5kJ$ or $26.3Wh$
SC Bank Equivalent DC Series Resistance	$220m\Omega$
Converter Efficiency	95%
Converter Limit	$15kW$ or $60A$ at $250V$

diagram of the HESS is shown in figure 3.1. Load current is determined by demands from the motor and inverter, the battery supplies the majority of the load current with the SC bank supplementing current via the converter. The control developed in this thesis aims

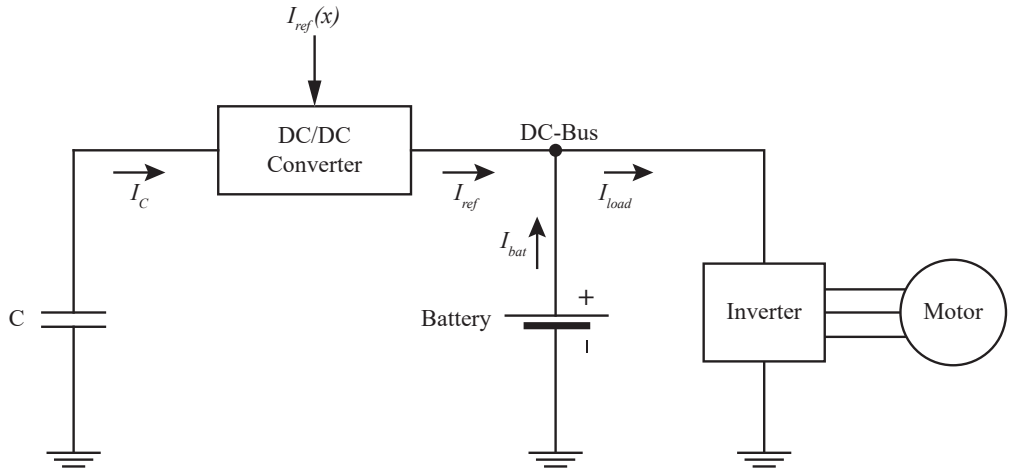


Figure 3.1: Block diagram of the modelled HESS used to develop NN based control

to minimize the RMS current of the batteries and is optimized around the FTP-72 driving profile, also called the urban dynamometer driving schedule (UDDS) profile, like that of [6]. The control must also ensure that the components of the HESS operate within their respective power limits. This includes preventing over or under charging the SCs, and

ensuring the current through the bi-directional converter does not exceed its $60A$ limit. The current limit for the converter is determined by two methods, the first is by determining the time taken to fully charge or discharge the SC bank. The intention is that the SC bank and converter be sized such that SC bank will be fully discharged when accelerating the vehicle from $0kph$ to $100kph$ in a reasonable time. The formula is shown below which calculates the total available charge of the SC bank and determines the time required to discharge or charge the bank based on the maximum available current:

$$t = \frac{C_{SC\,bank} \cdot \Delta V_{SC\,bank}}{I_{limit}} \tag{3.1}$$

The results are shown in figure 3.2 The other method is to train the NN on the UDDS profile with various converter limits and observe the point at which increasing the converter limit offers a diminishing increases in HESS effectiveness. The results of this method are shown in figure 3.3. Figure 3.3 shows that increasing the converter limit past approximately $40A$ offers little improvement in performance when trained on the FTP-72 driving profile, however $40A$ offers a discharge cycle time of $15.5s$. A converter limit of $60A$ was chosen which offers a discharge cycle time of $10.3s$; this is a more reasonable $0kph$ to $100kph$ time for a practical vehicle. The parameters for the vehicle and HESS system listed in table 3.1 are used to develop the NN control in the following sections. The HESS system and control are modelled in Simulink, a block diagram of the average model of the converter that considers efficiency losses is shown in figure 3.4. This control circuit models losses in the converter by limiting output current to 95% depending on the direction of current flow.

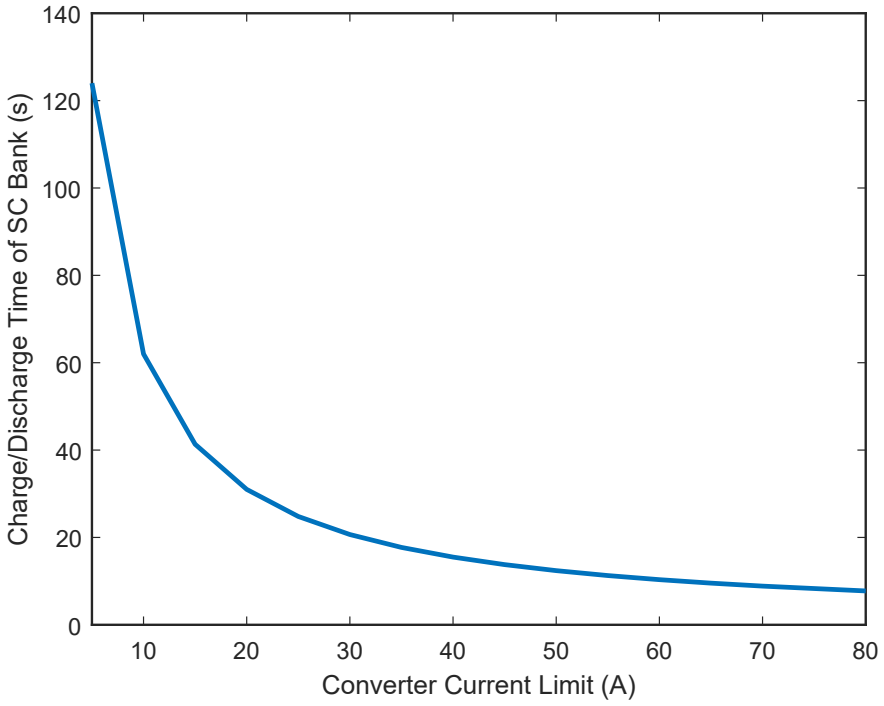


Figure 3.2: Time taken of a 3.1F SC bank to discharge from $250V$ to $50V$ based on converter limit.

3.3 Neural Network Parameters

3.3.1 Inputs to the Neural Network

The driving current profile of an EV is determined by unrelated external conditions such as traffic speed, road incline, wind speed, and vehicle weight; and the control for a HESS system used by an EV must consider the influences of all these conditions. As previously discussed, a trained NN offers the ability to find patterns in the inputs to optimize the reference current for future inputs. If the full driving profile is known, the current profiles of the EV's battery and SC bank can be optimized by a dynamic programming algorithm knowing future demands and the NN trained by supervised learning, like that done by [18]. Another option is to use reinforced learning and allow a genetic algorithm to optimize the weights of a NN. Reinforced learning introduces the opportunity to find an optimal solution

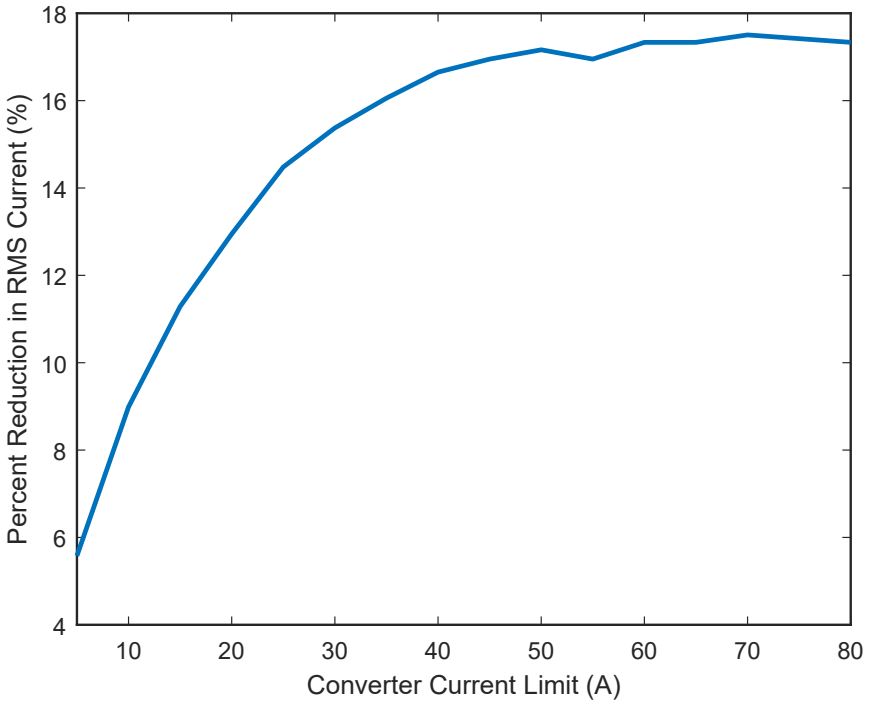


Figure 3.3: Percent reduction in RMS battery current of a trained NN controlling a converter of ranging current limits.

that may not be apparent when designing a current profile for the HESS externally.

The NN used in the HESS control takes eight inputs to produce a reference signal. The UDDS driving profile is used to train and test the NN control for the HESS. The National Renewable Energy Laboratory's advanced vehicle simulator (ADVISOR) uses the UDDS driving profile, along with the vehicle parameters listed in table 3.1 to generate the input profiles used by the NN. ADVISOR generates three separate profiles: load current, vehicle speed, and motor torque. Other NN inputs are generated in Simulink from the ADVISOR profiles, these inputs include vehicle acceleration, a moving average of speed, and local maximum of speed. The NN also monitors the voltage of the capacitor bank and the final input is a bias unit with a constant value of one. The goal of the inputs, apart from the bias unit, is to provide as much information of the vehicles current situation that would

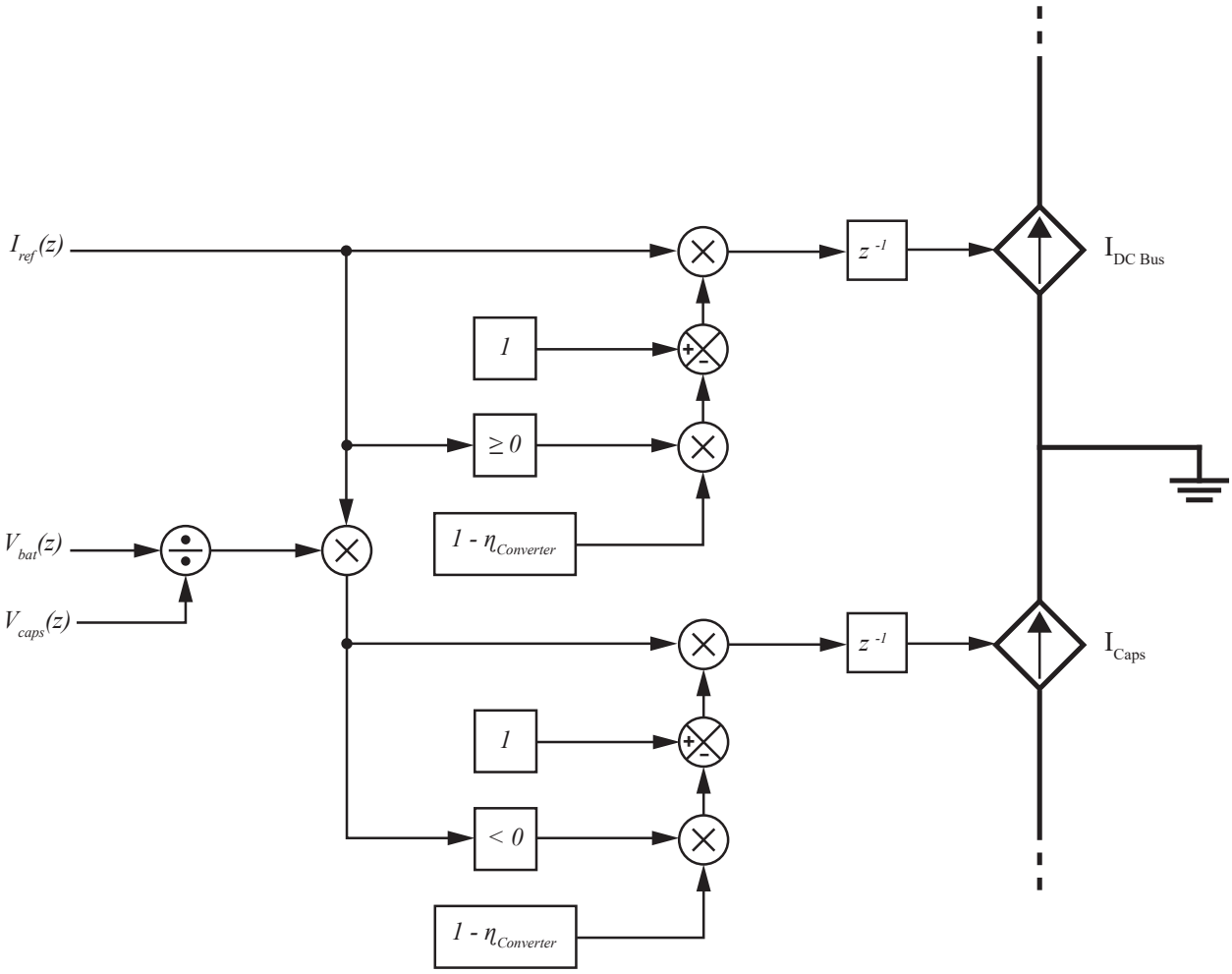


Figure 3.4: Block diagram of the modelled converter which considers operating efficiency.

affect or be affected by the current demanded by the motor and the SoC of the capacitors, whether that be an urban environment or a highway. The NN can recognize patterns in driving profiles from these inputs and adjust the reference HESS current appropriately. The torque, and acceleration profiles are shown in figure 3.6, and the load current profile and the resulting capacitor voltage profiles from a trained NN are shown in figure 3.7. The speed, moving average of speed, and local max of speed profiles are shown in figure 3.5. The block diagram of the control used to generate the moving average and local max are shown in figure 3.8. The window length for the moving average was chosen to be 1200s

by analysing the speed profile and adjusting so that the moving average profile depicts the driving circumstances the vehicle is currently in, such as highway driving or city traffic. The goal is to provide a fast enough response to adjust for a current driving environment, yet with enough memory to provide a useful input of the vehicles current driving environment. The local max is generated by analysing the acceleration of the vehicle and storing the maximum speed when acceleration changes from positive to negative. The objective for a local maximum input is that in urban, stop and go driving environments, the vehicle's top speed will remain relatively the same, as the speed limit of a section of road will likely be consistent. The effectiveness of each input can be analysed after the NN weights have

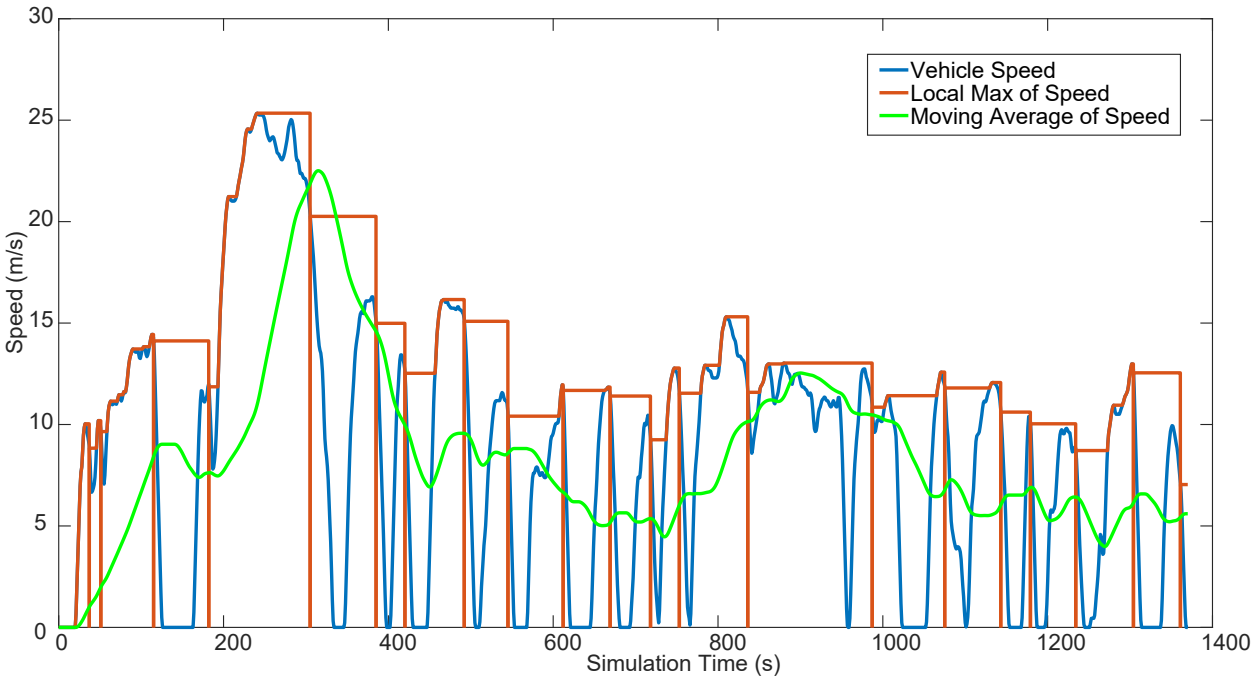


Figure 3.5: UDDS profiles for speed, moving average, and local max, used as inputs to the NN.

been trained by the genetic algorithm. This is done by evaluating the RMS value of the first layer weights corresponding to each input. Consider figure 1.4, to evaluate the effectiveness of input x_1, the RMS value of all the weights represented by all the arrows exiting x_1 would

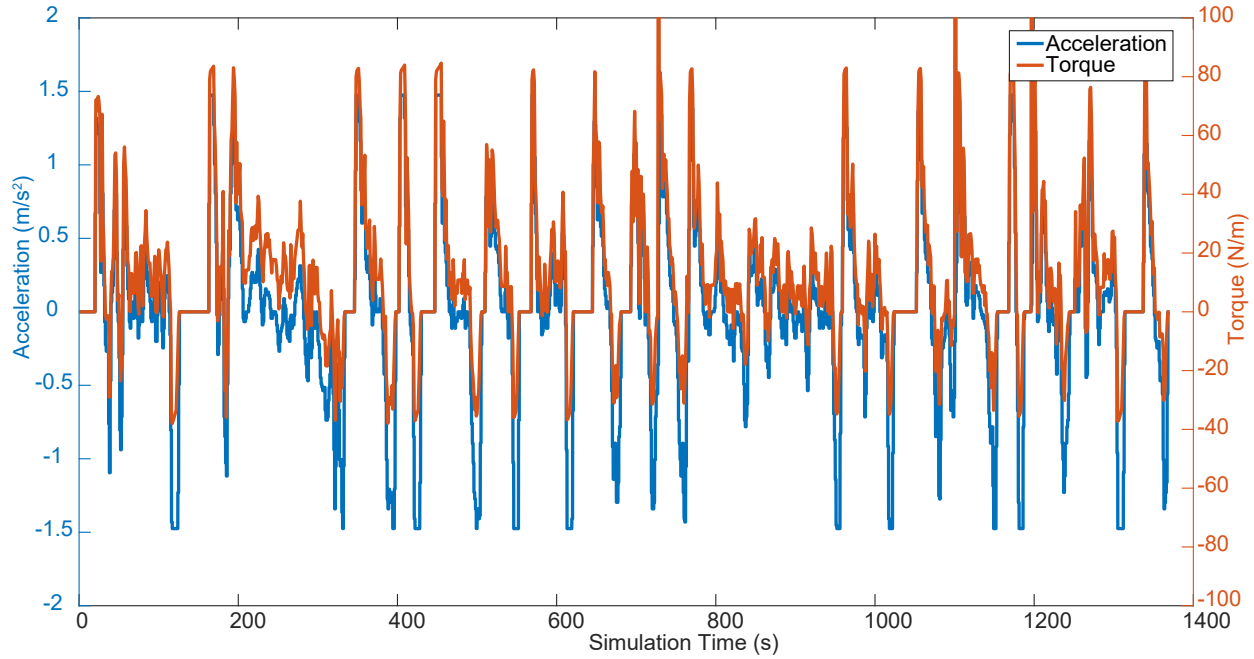


Figure 3.6: UDDS profiles for torque and acceleration, used as inputs to the NN.

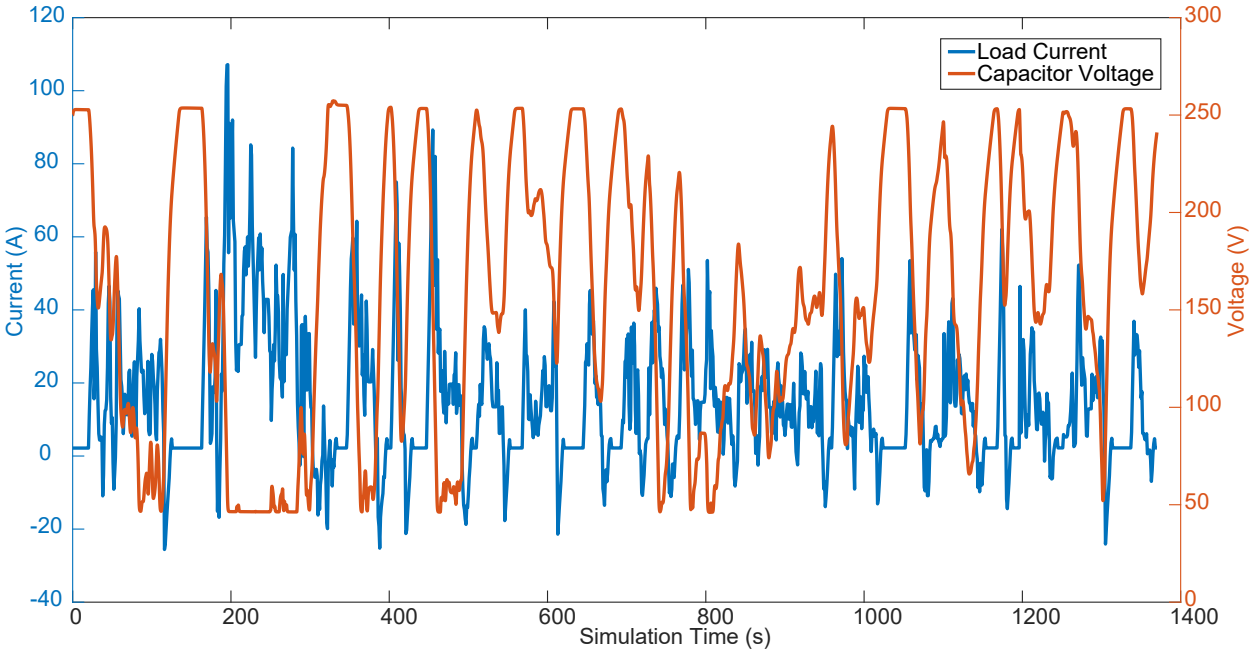


Figure 3.7: UDDS profiles for motor current and capacitor voltage, used as inputs to the NN.

be evaluated. Figure 3.9 shows RMS value of the layer 1 weights of a NN trained on the UDDS profile. The effectiveness of each input can be compared to one another to determine

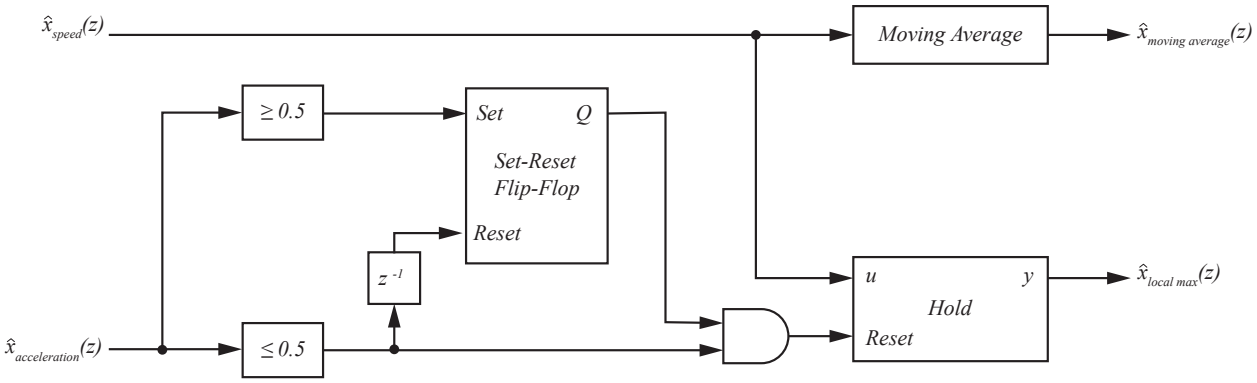


Figure 3.8: Control diagram which produces the local max and average profiles used as inputs to the NN.

which inputs provide more influence on the performance of the NN compared to others.

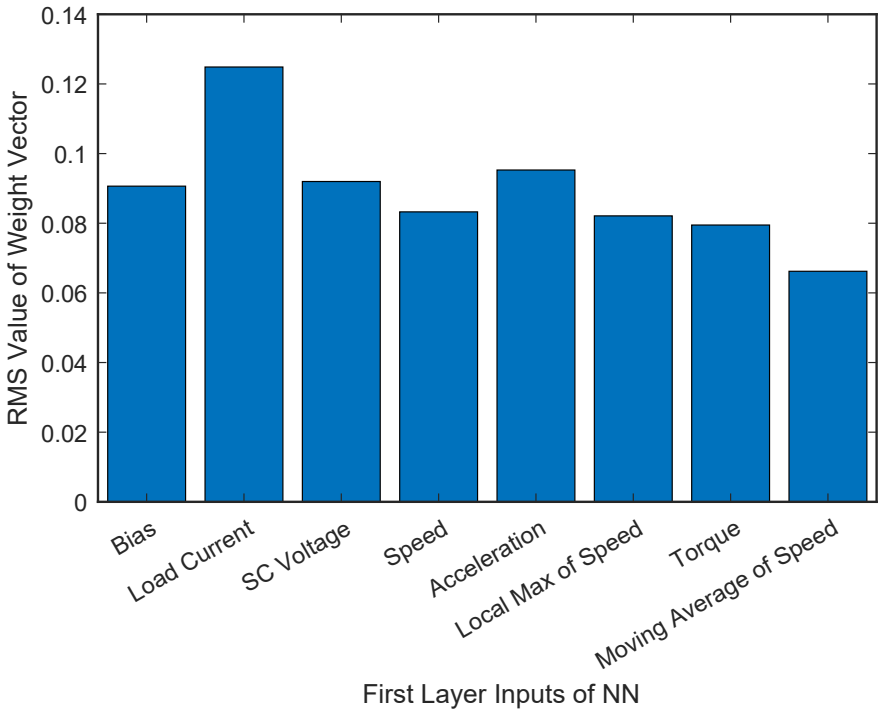


Figure 3.9: RMS value of the first layer weight vectors applied to the input signals of the NN, used to evaluate the effectiveness of a particular input against other inputs.

3.3.2 Number of Layers and Neurons of Neural Network

To develop a NN, the number of layers and number of neurons (known as hyper-parameters) are chosen such that an effective output will be produced yet not over designed such that

the NN will not require an unnecessary amount of computational hardware. Before proceeding with development of NN control for an HESS, it was necessary to prove that a NN could provide effective control for an HESS. To prove the functionality of NN control for an HESS, a network saturated with neurons was built; this was an over designed solution that required more computational power than necessary, however it would be a starting point to later optimize the NN for HESS control. Initially a four layer network was chosen, with 50 neurons per each of the three hidden layers. The genetic algorithm was allowed to train the network with the results shown in figure 3.10 and a minimum battery RMS current value of $19.82A$. Firstly the number of layers were reduced to observe an impact on HESS perfor-

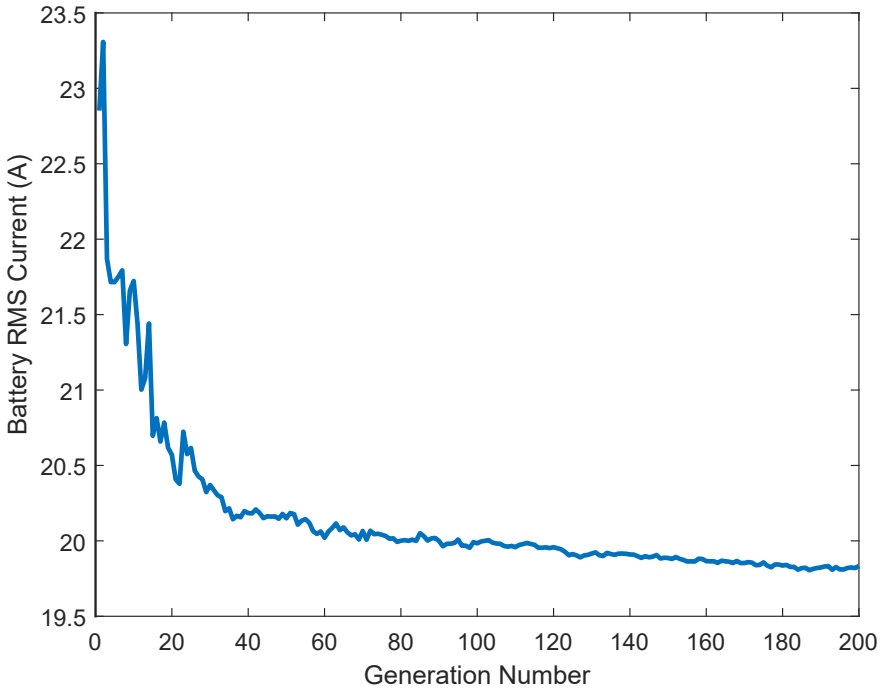


Figure 3.10: Genetic algorithm progression of a 4 layer network with hidden layers consisting of 50 neurons per layer.

mance. A layer was removed from the four layer NN and the genetic algorithm was again allowed to train the NN. The trained network offered a battery RMS current of $19.01A$ and the progression of the algorithm is shown in 3.11. This was repeated again with two layers,

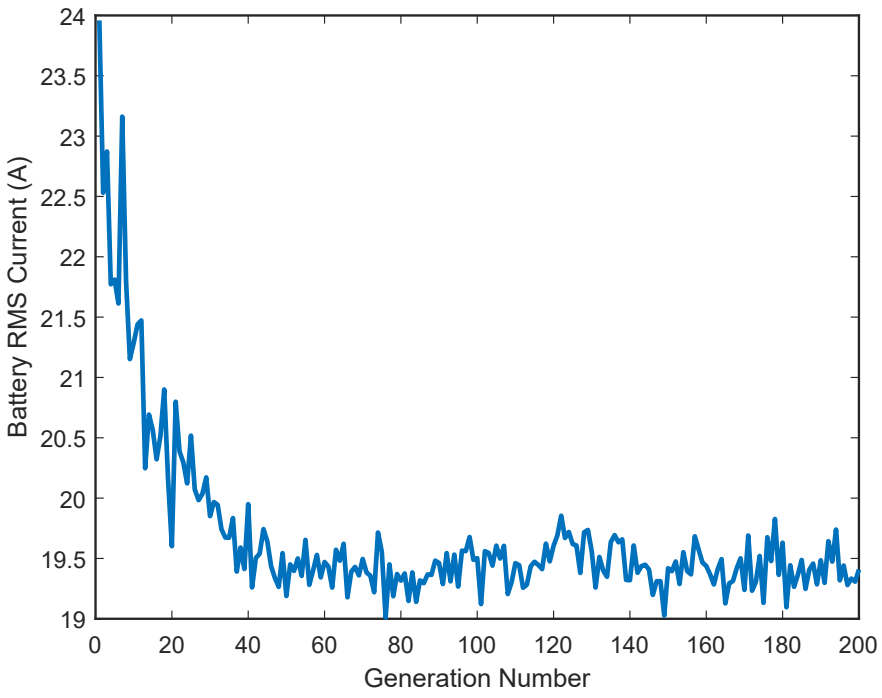


Figure 3.11: Genetic algorithm progression of a 3 layer network with hidden layers consisting of 50 neurons per layer.

with the results shown in figure 3.12 and a minimum battery RMS current value of $19.80A$. From this, it is determined that a three layer network had enough layers to provide optimal control for the HESS. To further optimize the NN the number of neurons needed for each layer is minimized. To determine this, the genetic algorithm repetitively trained the NN for various values of neurons for the first and second hidden layers. The value for the number of neurons is listed in table 3.2 and the parameters of the genetic algorithm are the same as that listed in table 3.3 with the exception of the simulation time step, which was changed to $1s$ in the interest of achieving quicker simulations. The genetic algorithm trained every

Table 3.2: Neuron combinations of the hidden layers of a 3 layer NN.

layer 2 neurons:	4	5	6	7	8	9	10	12	15	20	25	30
layer 3 neurons:	4	5	6	7	8	9	10	12	15	20	25	30

combination of neurons shown in table 3.2 and the results of the minimum battery RMS

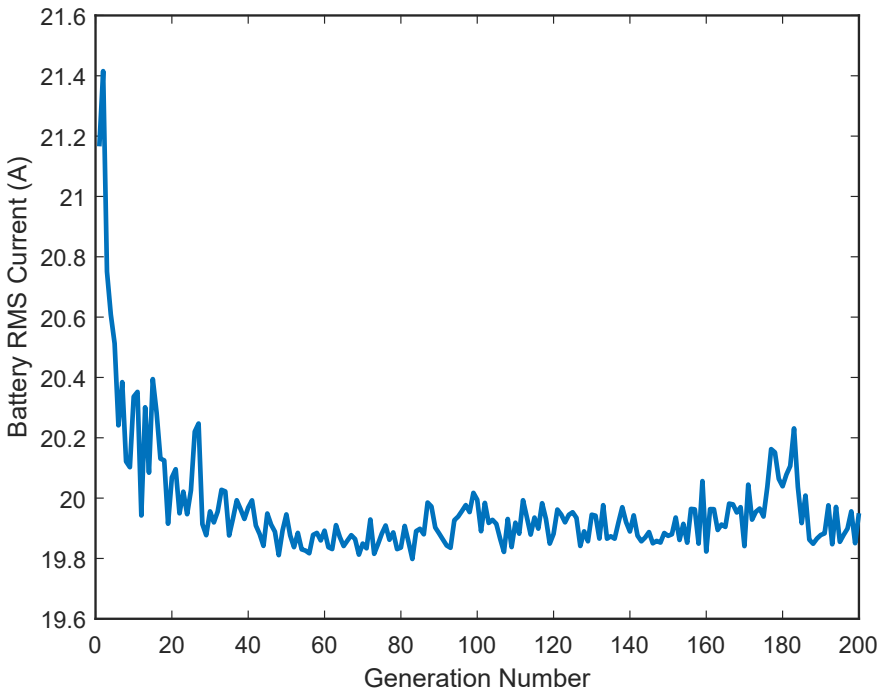


Figure 3.12: Genetic algorithm progression of a 2 layer network with hidden layers consisting of 50 neurons per layer.

current are shown in figure 3.13. The RMS current data represented in figure 3.13 has a

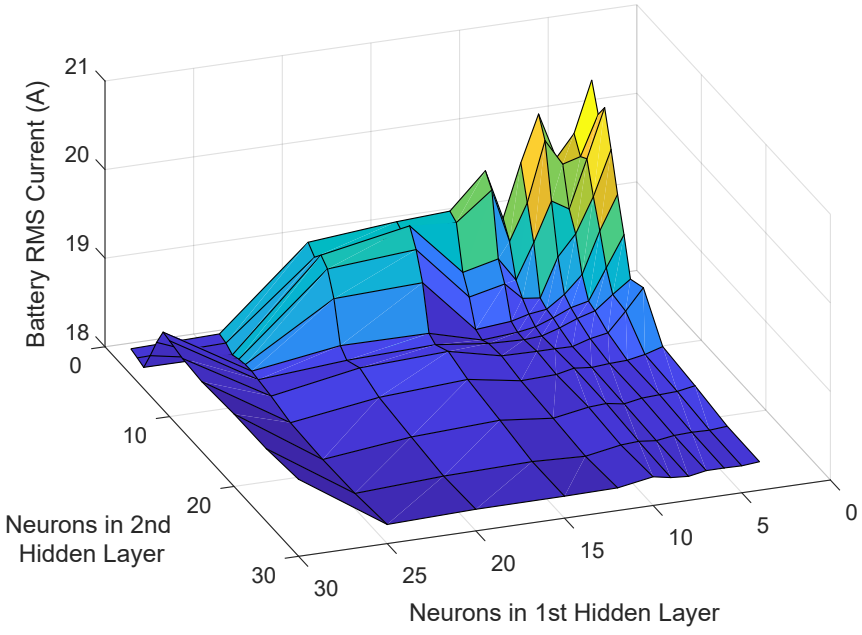
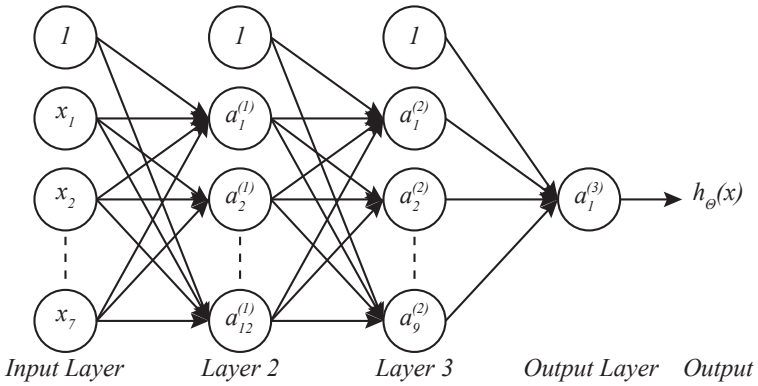


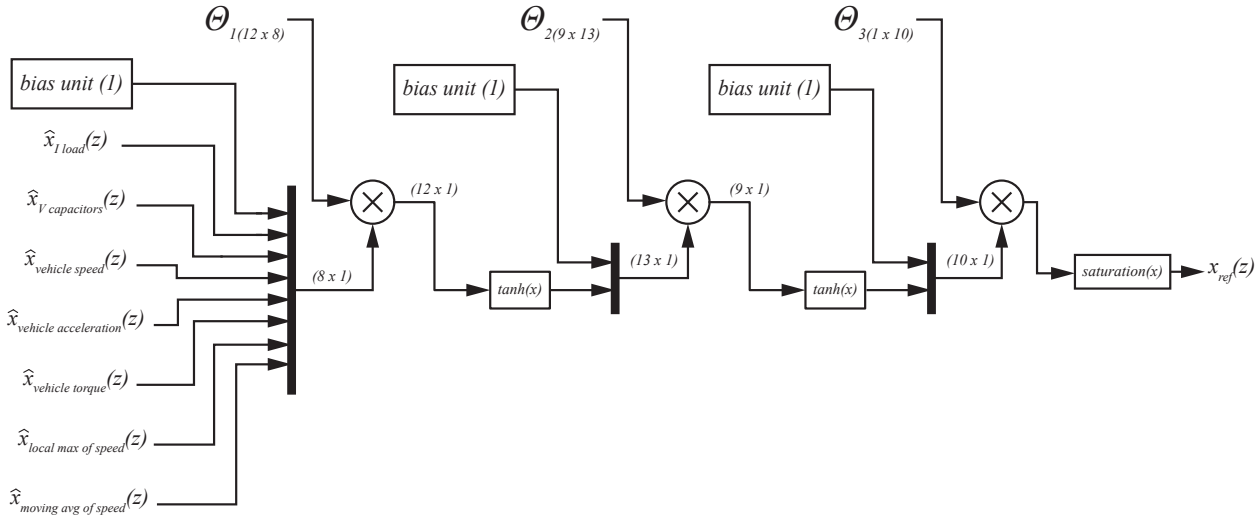
Figure 3.13: Battery RMS current of a trained NN with various combinations of hidden layer neurons based on table 3.2.

minima when the number of neurons in the first layer is 12 and the number of neurons in the second layer is 9. The combination was chosen such that the number of neurons per layer is the lowest without effecting the RMS current reduction of the trained network.

3.3.3 Construction of Neural Network



(a)



(b)

Figure 3.14: NN simulated using Simulink and used as control. (a) Architecture of the NN constructed in Simulink with 12 neurons in the first hidden layer and 9 in the second. (b) The block diagram of the same NN implemented in Simulink.

The NN and control for the HESS is developed in Simulink. Figure 3.14 shows the structure of the NN. Each signal input to the NN has varying magnitude and DC bias, to ensure that the NN is not dominated by a single input, the inputs are normalized by a control circuit like that shown in figure 3.15. Signals from all the inputs pass through the NN as described in section 1.3. The inputs are normalized then made into a vector using a multiplexor, then multiplied by the matrix of layer 1 weights. The output is a column vector which is conditioned by the hyperbolic tan block. A bias unit is added to the vector and the process repeats for the remaining layers. The output of the NN is not conditioned by a hyperbolic tan block, but rather limited by a saturation block. The NN generates an unconditioned reference current signal for the converter of the HESS which is later modified by control as described in section 3.4.

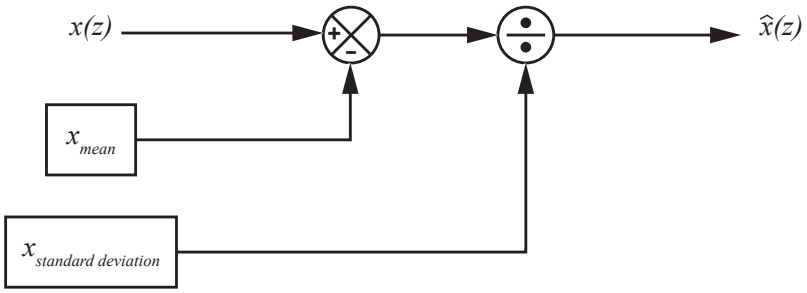


Figure 3.15: Control diagram used to normalize an input signal to the NN.

3.4 External Control

The NN of provides the optimal reference current for the converter to source from the SCs, however the NN does not consider the limits of the hardware of the HESS. To ensure the converter is not damaged due to over power and the SCs are not damaged due to over voltage situations, additional control is needed to condition the current reference signal

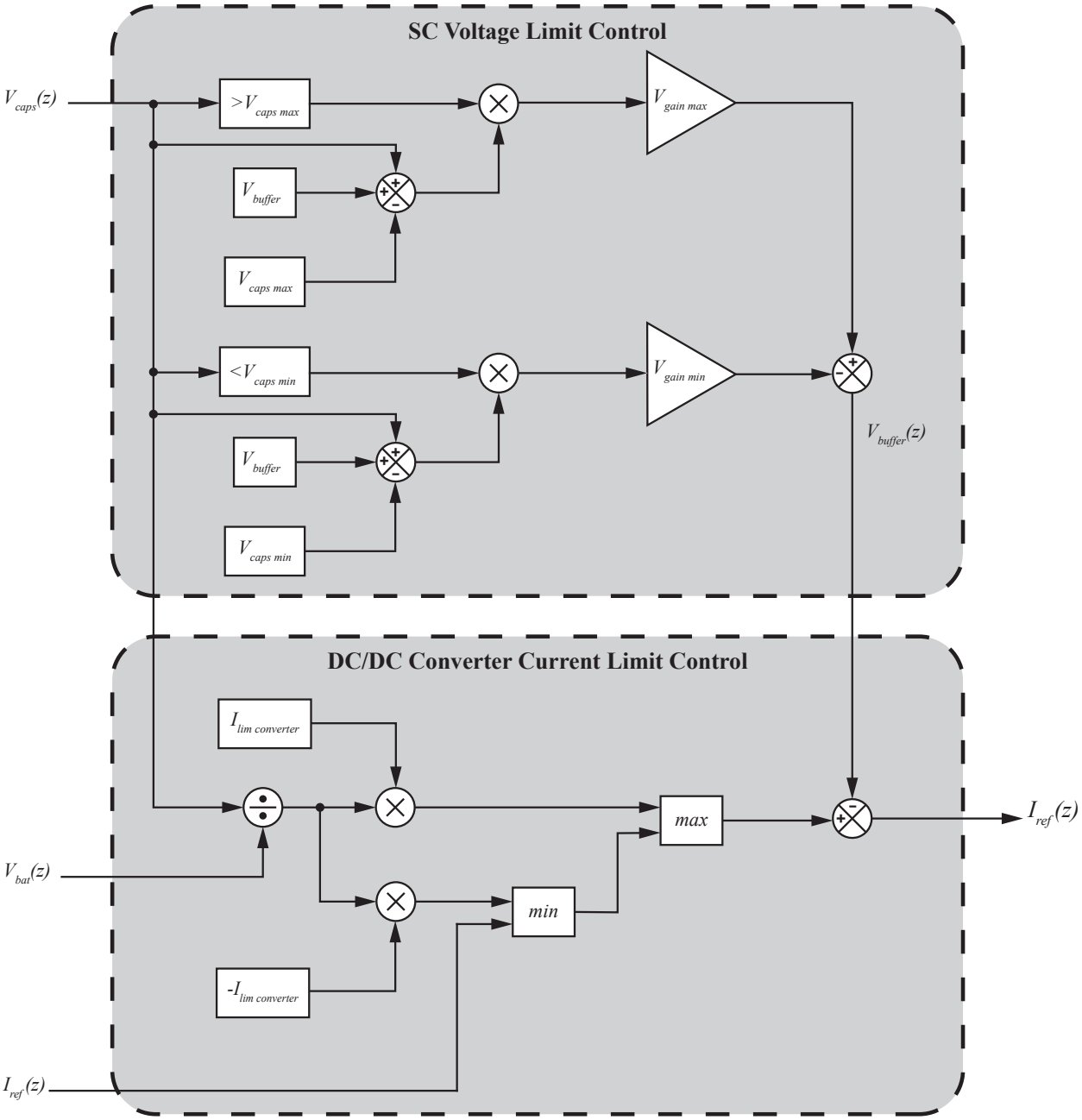


Figure 3.16: Diagrams of control used to ensure the SC bank does not exceed its voltage operation limits (top) and diagram of control used to ensure that the converter does not exceed its current operation limits (bottom).

provided from the NN to the converter. The reference current generated by the NN is the current requested from the converter on the side that is connected to the batteries which is the high voltage side of the converter. This is necessary so that the NN's generated reference signal is directly controlling the current that supplements the batteries, if the reference were

applied to the low side, the current injected into the DC bus of the HESS would be scaled based on the voltage of the SCs and the battery. The converter is rated at a current of 60A, however this is from the low side of the converter, this means the available current to the motor of the vehicle depends on the SoC of the SCs and is determined by formula 3.2.

$$I_{high\ side} = \left(\frac{V_{batt}}{V_{SC\ bank}}\right) I_{low\ side} \tag{3.2}$$

The first consideration of the external control is to ensure that the converter operates within its current limits, which are $\pm 60A$ on the low side. Controlling the current is not as simple as adding a saturation block to the output of the NN, because with a fixed current limit on the low side of the converter, the current limit of the high side of the converter changes through the operating range of the SC bank. Instead the limits must change as the SC bank's SoC changes. The DC/DC converter current limit control of figure 3.16 shows the control for the converter current limits. The minimum converter current is determined in formula 3.3 and the maximum converter current is determined by the formula 3.4.

$$I_{min} = \left(\frac{V_{batt}}{V_{SC\ bank}}\right) I_{converter\ limit} \tag{3.3}$$

$$I_{max} = -\left(\frac{V_{batt}}{V_{SC\ bank}}\right) I_{converter\ limit} \tag{3.4}$$

While the HESS is operating, the reference signal will be the minimum of either equation 3.3 or the NN's reference current, or the maximum of either equation 3.4 or the NN's reference current. This ensures the current from the low side of the converter does not exceed $\pm 60A$.

To ensure that the SCs operate within their limits, SC voltage limit control block shown

in figure 3.16 is implemented. This is a proportional control meant to counter the reference generated from the NN. When the voltage of the SCs reach within $10V$ of their specified limit, the control is active, and a buffer signal whose polarity is opposite the current reference signal is added to the NN's signal to negate it and prevent further current transfer in a direction that would damage the SCs. The formula showing this control when the SCs approach their minimum voltage is shown in equation 3.5 and the formula showing the control when the SCs approach their maximum voltage is shown in equation 3.6. Here, V_{max} and V_{min} are the specified limits of the SC bank, V_{buffer} is the $10V$ buffer where the control becomes active, and V_{gain} is the gain applied to the signal so that it will match the maximum of the current reference signal when the SC bank's voltage reaches its limit.

$$\begin{aligned} F(z)_{max} &= (V_{min} + V_{Buffer} - V(z)_{SC})\, V_{gain} \\ V_{gain} &= \left(\frac{V_{SC\,max}}{V_{batt}}\right)\left(\frac{I_{limit}}{V_{buffer}}\right) \end{aligned} \tag{3.5}$$

$$\begin{aligned} F(z)_{min} &= (V(z)_{SC} + V_{Buffer} - V_{max})\, V_{gain} \\ V_{gain} &= \left(\frac{V_{SC\,min}}{V_{batt}}\right)\left(\frac{I_{limit}}{V_{buffer}}\right) \end{aligned} \tag{3.6}$$

The gain values differ because, similar to the current limit control, the current reference signal changes as the SoC of the SC bank changes. The current and voltage control ensure that the HESS operates within its limits, and no damage occurs to the hardware of the HESS. The NN is then allowed to learn and generate a signal without a need to consider these constraints.

3.5 Genetic Algorithm

For the NN to have an effective output for the HESS, it must first be trained. Training is the process of determining a weighting value, referred to as Θ, to each input of each neuron as discussed in section 1.4. The process used to train the NN used in the control for this HESS is reinforced learning which makes use of a genetic algorithm. A previous NN control uses supervised learning, where an optimal SC current profile must first be generated by a "dynamic programming algorithm" [18] and the NN is trained so that it's output matches that of this algorithm's. In this scenario, the effectiveness of the NN is limited by that of the algorithm, and care must be made to ensure the algorithm produces a SC profile that is most effective at reducing the battery RMS current. Alternatively, reinforced learning does not require a predetermined SC current waveform for the NN to learn from. Instead a genetic algorithm will continue to adjust the weights of the NN to produce an SC current profile that reduces the batteries' delivered RMS current. This presents the opportunity for the NN to find solutions that may not be generated by an external algorithm. While the system model and control is constructed in Simulink, Matlab code is used to implement the genetic algorithm. The code used for this thesis is shown in appendix section B and the parameters initialized by the genetic algorithm are listed in table 3.3.

3.5.1 Genetic Algorithm Parameters

The rate at which a NN is trained depends on the parameters of the genetic algorithm and the properties of the NN. The input layer size is determined by the number of inputs to the NN, as discussed in section 3.3.1. The hidden layer sizes are the size of the hidden layers of the

Table 3.3: Parameters initialized at the beginning of the genetic algorithm.

Parameter	Parameter Name	Value
Input Layer Size	`input_layer_size`	7
Hidden Layer 1 Size	`hidden_layer_1_size`	12
Hidden Layer 2 Size	`hidden_layer_2_size`	9
Output Layer Size	`output_layer_size`	1
Population Size	`population_size`	20
Simulation Time Step	`T_step`	$0.1s$
Converter Limit	`converter_limit`	$60A$
Converter Efficiency	`converter_efficiency`	95%
Number of Parents	`number_of_parents`	2
Mutation Rate	`mutation_rate`	0.3%
Randomization Factor	`epsilon_ini`	0.15

NN, and the size of these layers are discussed in section 3.3.2. The output layer size for this NN is 1, and is the reference current to be used by the converter. This output signal is later conditioned by additional control to ensure that the capacitor and converter do not exceed their respective operation parameters as discussed in section 3.4. Population size refers to the number of vectors of NN weights that are to be run for each generation (one weight vector includes all the genes necessary for the NN to operate). This value needs to be large enough to allow sufficient variation among members of the population, however not to large as to unnecessarily slow the progression of the genetic algorithm. Figure 3.17 shows the difference a population size will make on the learning rate of the genetic algorithm. Simulation time step is the incremental step made by Simulink when the simulation is running. A time step of 0.1 second is used for the genetic algorithm which translates to a simulation sampling frequency of 10Hz. According to the Nyquist Sampling Theorem, if a sampling rate of 10Hz is used, a signal below 5Hz can be accurately interpreted without loss of information. Figure 3.18 shows an RMS frequency analysis of the input signals generated from ADVISOR, above a frequency 1Hz the RMS value of the signals are near zero and contain no useful information.

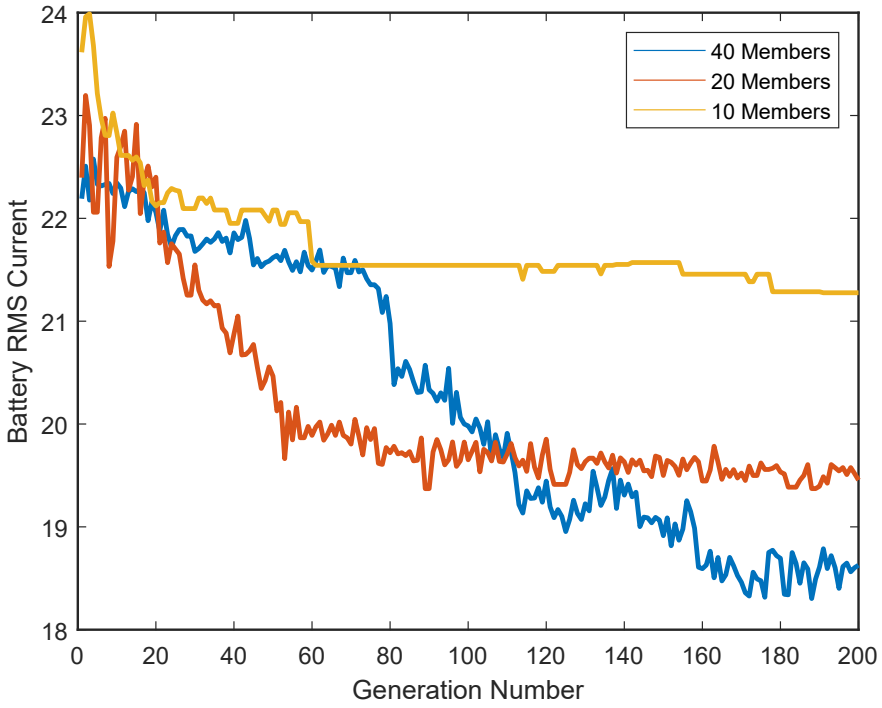


Figure 3.17: The effect of various population sizes on the progression of a genetic algorithm.

This justifies a time step of 0.1Hz as more than sufficient when running the simulation as

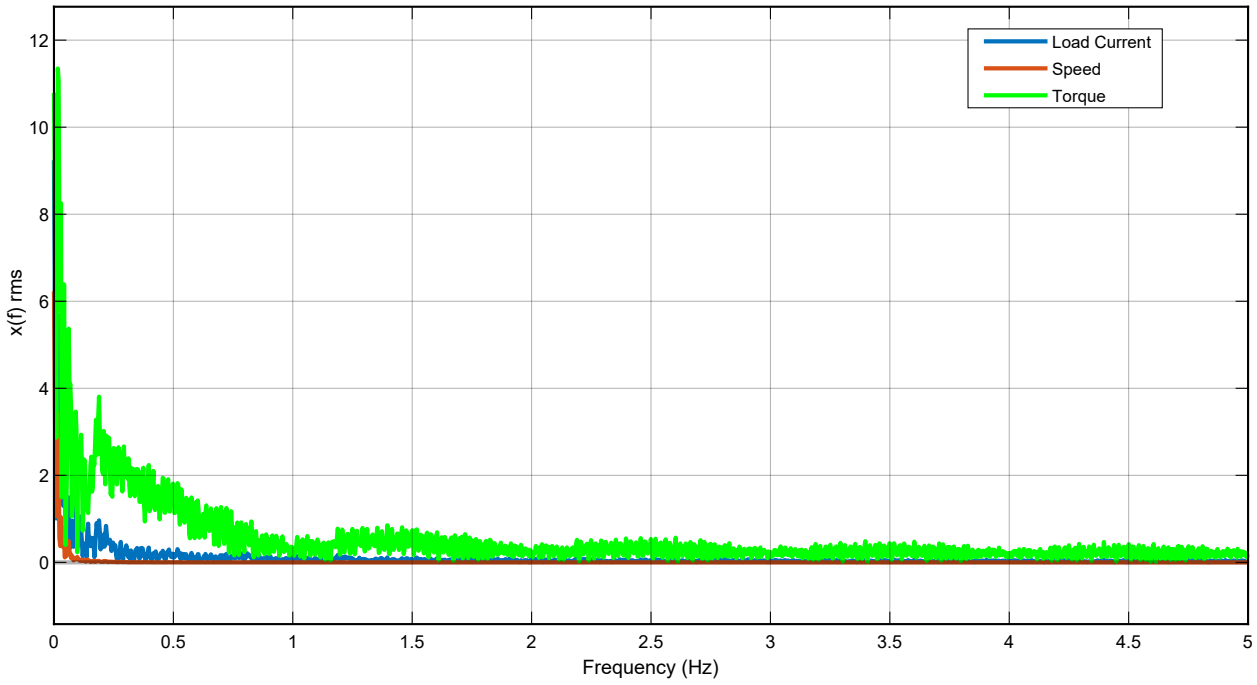


Figure 3.18: An RMS frequency spectrum analysis of the NN input signals generated in ADVISOR.

decreasing the time step does not provide any improvements in simulation quality. The converter limit is the hardware constraints of the converter, this value is discussed in section 3.1. The converter efficiency is the model's losses in the converter, this value is chosen to be 95% which is the typical efficiency of a conventional buck/boost converter [13]. The number of parents refer to the number of members of a population that will contribute genes to a new member. A value of 2 is chosen as higher values limit progression of the genetic algorithm. If more parents are used to reproduce NN genes, poorly performing members will influence subsequent generations and inhibit improvements of the genetic algorithm. The mutation rate is the probability that a gene is mutated and the randomization factor is range that the new value can fall between. A higher mutation rate will allow the genetic algorithm to train the NN more quickly, however the outcome of the genetic algorithm has relatively high fluctuations from one generation to the next. A lower mutation rate results in a slower progression of the genetic algorithm, but lower fluctuations in the result as the genetic algorithm progresses. Figure 3.19 compares the progression of a genetic algorithm with only the mutation rate changed. A mutation rate to low slows or even stops genetic algorithm progression, while a mutation rate too high results highly varying generation progression. The randomization factor is the range of values which new and mutated genes will fall between. The inputs to the NN are normalized so the mean of each input is 0 and the standard deviation is 1, and the output is scaled by 200. A randomization factor of 0.15 allows a single input to the NN to saturate the output, if the weights associated with that

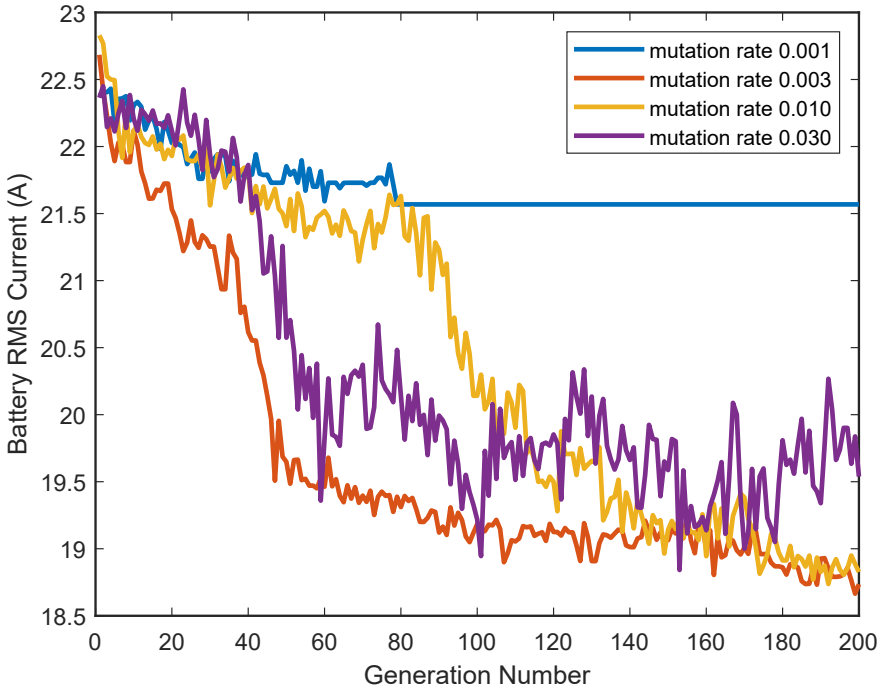


Figure 3.19: The effect of various mutation rates on the progression of a genetic algorithm.

input are all approximately 0.15, as shown in equation 3.7.

$$\begin{aligned} a_1^{(2)} &= g\left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + ... + \Theta_{17}^{(1)} x_7\right) \\ a_1^{(2)} &= g\left(0.15 \times 1.0 + 0.15 \times 1.0 + ... + 0.15 \times 1.0\right) \\ a_1^{(2)} &= 1.05 \end{aligned} \tag{3.7}$$

It is important that a poorly performing NN results in an undesirable output, this ensures a poor fitness is assigned to that member, and the member's genes have a low chance of being selected for subsequent generations.

3.5.2 Genetic Algorithm Operation

This section describes the operation of the genetic algorithm. When training a NN's weights, the genetic algorithm first loads the driving profiles created by the ADVISOR

program. Next the parameters of the NN, simulation, and genetic algorithm are initialized; these parameters are listed in table 3.3.

The program either loads parameters from a previous run to continue to develop, or uses a function to randomize the parameters of the NN based on the mutation rate specified shown in equation 3.8, where rand is a randomized matrix whose size is the size of the input layer by the size of the output layer plus 1, and ε_{init} is the randomization factor. The program uses these matrices to build a population to run for the first generation of the genetic algorithm.

$$\Theta_n = 2\ Random_{(L_{out} \times l_{in}+1)} \varepsilon_{initial} - \varepsilon_{initial} \tag{3.8}$$

The total parameters are deconstructed to form column vectors and are combined to form a matrix, where each column is a member of the total population for that generation. When one member of the population is to be simulated, the parameters are organized into Θ arrays for each layer of the NN and the simulation is run. Information of each run is stored including: battery current, capacitor current, simulation time, the RMS value of battery current, and the NN parameters of each layer. After each member of the population is run, a function uses the battery RMS results to determine the fitness of each population member. The operation of the function is described by equation 3.9, where $\hat{I}_{RMS}$ is a matrix of the normalized RMS battery currents, $i_{RMS\,min}$ is the minimum RMS value of the current generation, $i_{RMS\,max}$ is the maximum RMS value of the current generation, and $I_{fitness}$ is a scaled and inverted $\hat{I}_{RMS}$ matrix. This equation produces a higher fitness value from a lower RMS current value.

$$\hat{I}_{RMS} = \frac{I_{RMS} - i_{RMS\,min}}{i_{RMS\,max} - i_{RMS\,min}} \tag{3.9}$$

$$I_{fitness} = (1 - \hat{I}_{RMS})100 \tag{3.10}$$

After the fitness of each population member has been determined, a function uses the fitness values to create a matting pool where population members with a higher fitness occupy a larger portion of the mating pool. The mating pool is a matrix where columns are occupied by the NN parameters, the number of columns occupied be each population member is the fitness value of that member determined by the previous generation, shown in equation 3.11. Here $\Theta_{mating\,pool}$ notates the mating pool and $\Theta^n_{(m\times fitness)}$ is a matrix comprised of identical columns which are the NN parameters of length m of a specific member and the number of columns is the fitness value of that member.

$$\Theta_{mating\,pool} = \Theta^1_{(m\times fitness)} + \Theta^2_{(m\times fitness)} + ... + \Theta^n_{(m\times fitness)} \tag{3.11}$$

Next, a function produces members for the next generation by selecting members of the mating pool determined by the number of parents specified. Random genes of the NN parameters are combined to create a new population member for the next generation. This is described by equation 3.12 which shows the next generation of NN parameters, labelled Θ^2, whose elements are comprised of elements of 3 parents, Θ^1_a, Θ^1_b, and Θ^1_c. The Θ columns multiply with columns containing 1s and 0s, which signify the random selection of genes to produce the next generation's weight matrix.

$$\begin{bmatrix} \Theta^2_{1a} \\ \Theta^2_{2c} \\ \Theta^2_{3b} \\ \vdots \\ \Theta^2_{nb} \end{bmatrix} = \begin{bmatrix} \Theta^1_{1a} \\ \Theta^1_{2a} \\ \Theta^1_{3a} \\ \vdots \\ \Theta^1_{na} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}^\intercal + \begin{bmatrix} \Theta^1_{1b} \\ \Theta^1_{2b} \\ \Theta^1_{3b} \\ \vdots \\ \Theta^1_{nb} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}^\intercal + \begin{bmatrix} \Theta^1_{1c} \\ \Theta^1_{2c} \\ \Theta^1_{3c} \\ \vdots \\ \Theta^1_{nc} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}^\intercal \tag{3.12}$$

The final step is to mutate the new population members by changing random genes based on the mutation rate and the randomization factor. The output of this function is similar to that of equation 3.8, however a percentage of random genes are changed which is dictated by the specified mutation rate. At this point, a new generation has been created and the next generation is ready to be run. The simulation is re-run repeatedly until the RMS value of battery current no longer improves. The NN parameters are then saved to be used in the NN control.

3.6 Summary

This chapter discusses the Simulink model of the HESS and how it is controlled by the NN and the external control. Once trained, the NN generates the fundamental current reference for the converter to supplement battery current and lower the overall RMS current supplied by the battery. The external control is meant to adjust the reference current signal to ensure that the converter and capacitor bank do not exceed their respective rated operational limits. This chapter then discussed the process in which parameters for the NN and genetic algorithm are selected and compares the effect that various parameters have on the learning rate and the operation of the NN.

Chapter 4

Simulation Results

4.1 Overview

This chapter will cover the results of the genetic algorithm used to train a NN used in a HESS, and the effect of a NN controlled HESS on battery current. Section 4.2 covers in detail the advancements in the HESS's effectiveness as the NN is trained on a standard driving profile. Section 4.3 shows the battery and load current waveforms and discusses the effectiveness of the HESS in reducing the battery's RMS current. Section 4.4 compares the effectiveness of the NN control with other works discussed in section 2.

4.2 Progression of Genetic Algorithm

As the genetic algorithm runs, the genes of the NN are adjusted to continually improve the performance of the HESS, as described in section 3.5. The progression of the genetic algorithm is shown in figure 4.1, where the battery RMS current of best performing member of a population for each generation is recorded and plotted with its generation number. Initially, substantial improvements are made to the battery RMS current genes randomly find their way to positions providing great improvements. With a mutation rate of 0.01%, the majority of change within the NN comes from rearranging existing genes, which show

the biggest improvements at the beginning of the simulation. Values of genes which aid the performance of the NN are likely to propagate into subsequent generations, and genes that do not aid performance occupy a much smaller section of the mating pool. Later genes are mutated which provide slight improvements until the genes are optimized around the given profile.

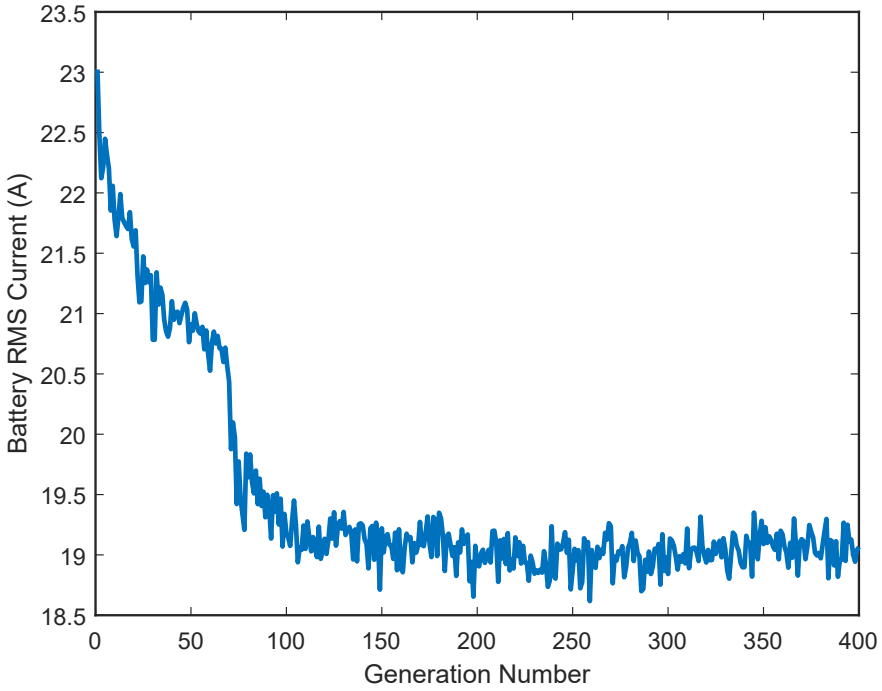


Figure 4.1: Progression of the genetic algorithm initialized with the parameters of table 3.3 used to train a 3 layer NN with 8 inputs, 12 neurons in the second layer, 9 neurons in the 3rd layer, and 1 output.

4.3 Results of Neural Network Control for Hybrid Energy Storage System

The objective of the trained NN and external control is to reduce the RMS current that the battery bank must source to the load. Lowering the RMS current of the battery reduces the thermal stress of the battery and prolongs the battery's life. Training the NN

conditions the NN to provide a reference signal based on patterns found in the input signals, these patterns may not be easily discovered by manually creating an HESS control. The NN aims to optimize the SC bank's voltage to prepare for unforeseen driving circumstances and source the appropriate amount of current from the converter. The results of the HESS controlled by the NN trained on the UDDS driving profile are shown in figure 4.2. The NN is effective at lowering battery current, shown in green, throughout the simulation with the exception of when the vehicle accelerates to highway speeds (approximately $27m/s$ or $100km/h$) between 200s and 300s. Figure A.1 in the appendix shows the speed of the vehicle throughout the UDDS profile. A figure showing the results of an urban driving scenario

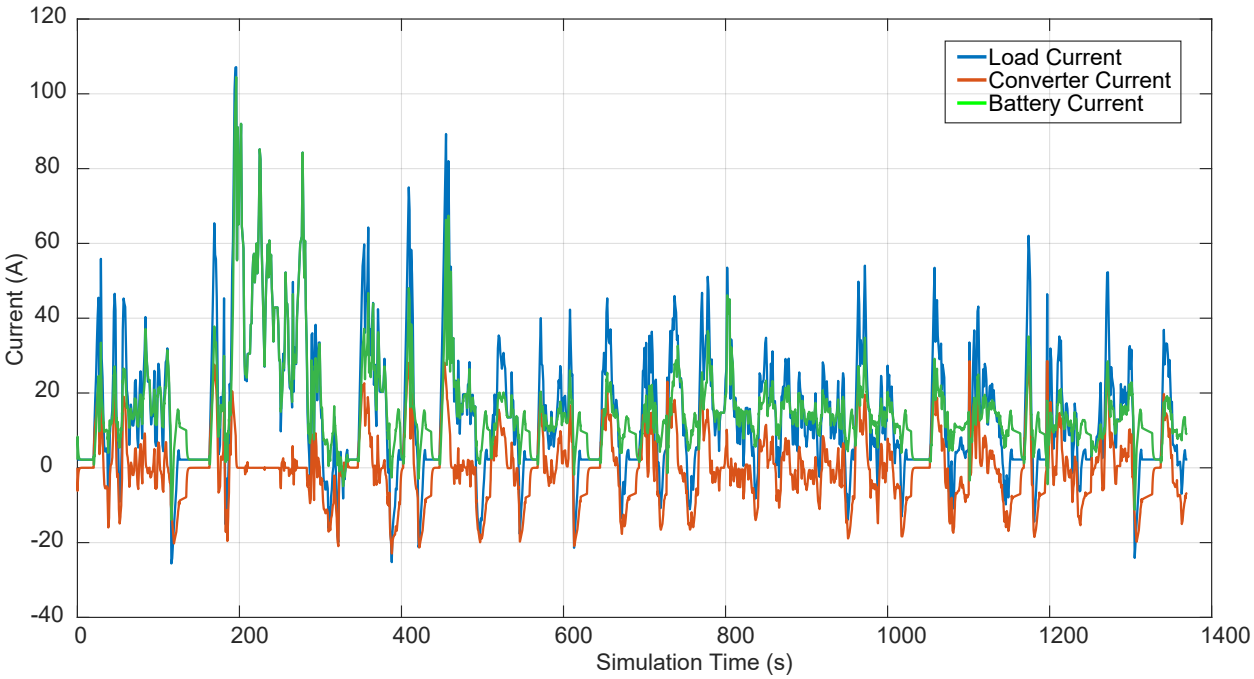


Figure 4.2: Graph showing the reduction a trained NN controlled HESS will have on an EVs load current. The load current represents the battery current of an EV with no added HESS, the battery current represents the reduction in battery current an HESS can offer, and the converter current is the difference of the two currents that is directed to the SC bank.

between $600s$ and $1000s$ is show in figure 4.3, where the speed of the vehicle does not exceed $15m/s$ or approximately $50km/h$, as would be the case of a central urban environment. The

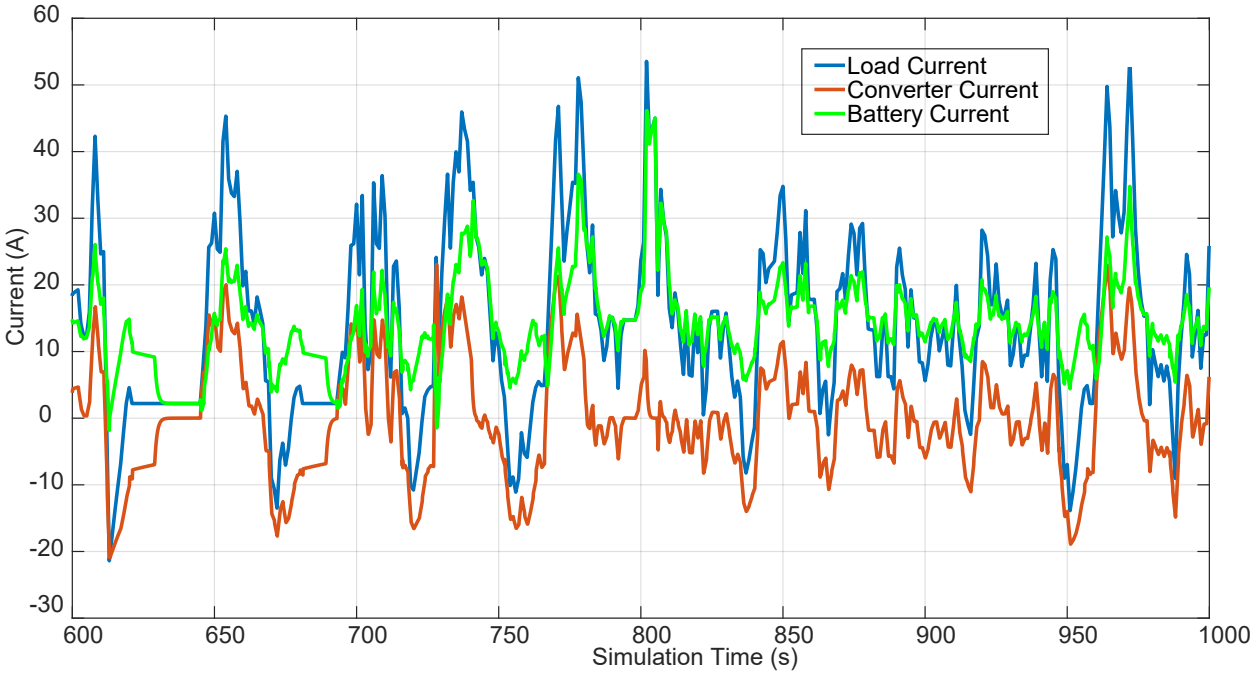


Figure 4.3: An urban driving section of the currents shown in figure 4.2.

other objective of the control is to ensure that the SC bank does not exceed its safe working voltage parameters and the converter does not exceed its safe working current parameters. These current and voltage values are shown in figure 4.4 which plots the SC bank's voltage and current values throughout the simulation. It is noted that the NN fully utilized the current and voltage limits of the converter and SC bank without exceeding these limits. The use of this HESS control reduces the battery RMS current value from $23.48A$ to $19.92A$ on the UDDS driving profile, this is a reduction of 15.15%. This reduces the heat produced by a battery cell from $171W$ to $123W$ based on equation 2.2 and battery values of table 2.1 like that of [1]. This is a 28.1% reduction in heat energy. The addition of an HESS also reduces peak current draw from the battery, as would be the case when the vehicle is accelerating. Other measures of the HESS effectiveness are: maximum peak battery current reduction, where the maximum reduction of the batteries' peak current is used to evaluate the control,

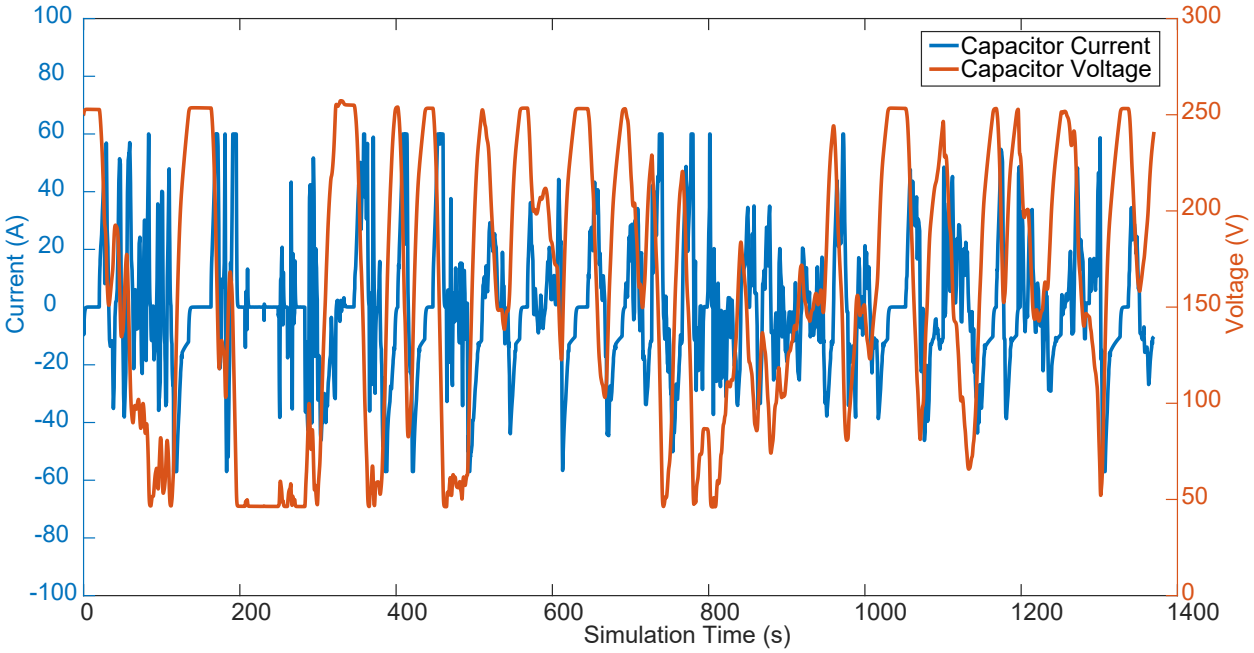


Figure 4.4: Graph showing the voltages and current sourced from the SC bank of the HESS controlled by a NN trained on the UDDS profile.

and peak to average power ratio (PAPR) over peak to average velocity ratio (PAVR). The later method is done by [14], where the performance of an HESS is evaluated by calculating PAPR and normalizing it with respect to velocity by dividing the PAPR by PAVR, as shown in formula 4.1.

$$PAPR/PAVR\,\%_{reduction} = \left(\frac{P_{peak}/P_{mean}}{v_{peak}/v_{mean}}\right) 100 \tag{4.1}$$

An effective HESS will decrease peaks in current. The reduction in RMS current and the reduction of $PAPR/PAVR$ ratios is used to determine the effectiveness of a NN that is trained on the UDDS profile and simulated over the UDDS profile. Three other driving profiles like those used by [14] are used to test the effectiveness of the NN trained on the UDDS profile. The other profiles include US06, a high speed driving profile, LA92, another low speed urban driving profile, and the new European driving cycle (NEDC), a standardized driving test to test emissions of passenger vehicles at a variety of fixed speeds. The speed, load

current, and torque waveforms of each profile are shown in appendix section A. Battery RMS reduction, max peak battery current reduction, and $PAPR/PAVR$ reduction evaluate the effectiveness of the HESS in various ways. Measuring battery RMS current reduction does not reflect the impact peak current draws will have on the HESS, because when optimizing the NN weights, the genetic algorithm favours a reduction in battery RMS over the whole driving cycle. $PAPR/PAVR$ reduction and maximum peak battery current reduction however reflects a reduction of peak battery currents and is less influenced by a reduction in battery RMS current over a driving profile. Maximum peak battery current reduction is a value of the largest reduction between a peak in battery current before the addition of an HESS, and the battery current after the addition of an HESS. The $PAPR/PAVR$ will be largely influenced by one large peak power value, regardless of the length of the simulation. The results of the HESS over these driving profiles are listed in table 4.1. These results show

Table 4.1: Percent RMS current reduction, percent max peak current reduction, and percent $PAVR/PAPR$ reduction of a NN controlled HESS trained the UDDS profile.

Driving Profile	RMS Current (A)		RMS Current Reduction (%)	Peak Battery Current (A)		Max Peak Battery Current Reduction (%)
	Before	After		Before	After	
UDDS	23.48	19.92	15.15	36.89	17.52	52.51
US06	62.55	59.68	4.59	100.67	75.84	24.68
LA92	30.68	27.62	9.99	43.74	20.34	53.50
NEDC	22.07	21.24	3.78	33.43	20.50	38.68

Driving Profile	Peak Power (kW)		Average Power (kW)		PAPR		PAVR	PAPR/PAVR		PAPR/PAVR Reduction (%)
	Before	After	Before	After	Before	After		Before	After	
UDDS	42.93	41.85	5.91	6.01	7.26	6.96	2.90	2.51	2.40	4.15
US06	106.10	106.09	17.76	17.82	5.97	5.95	1.68	3.56	3.54	0.32
LA92	47.40	43.72	7.52	7.61	6.31	5.75	2.70	2.34	2.13	8.86
NEDC	42.53	42.53	5.26	5.32	8.08	8.00	3.61	2.34	2.21	1.03

a NN that is most effective at lowering both RMS currents and $PAPR/PAVR$ ratios in the low speed urban profiles UDDS and LA92 where the voltage range of the SC bank is effectively used. The control offers a reduction in the high-speed US06 profile and NEDC

profile, however it does not fully optimize capacitor voltages and depletes the energy available from the SC bank before reaching the maximum speed, shown in figure 4.5. The same results

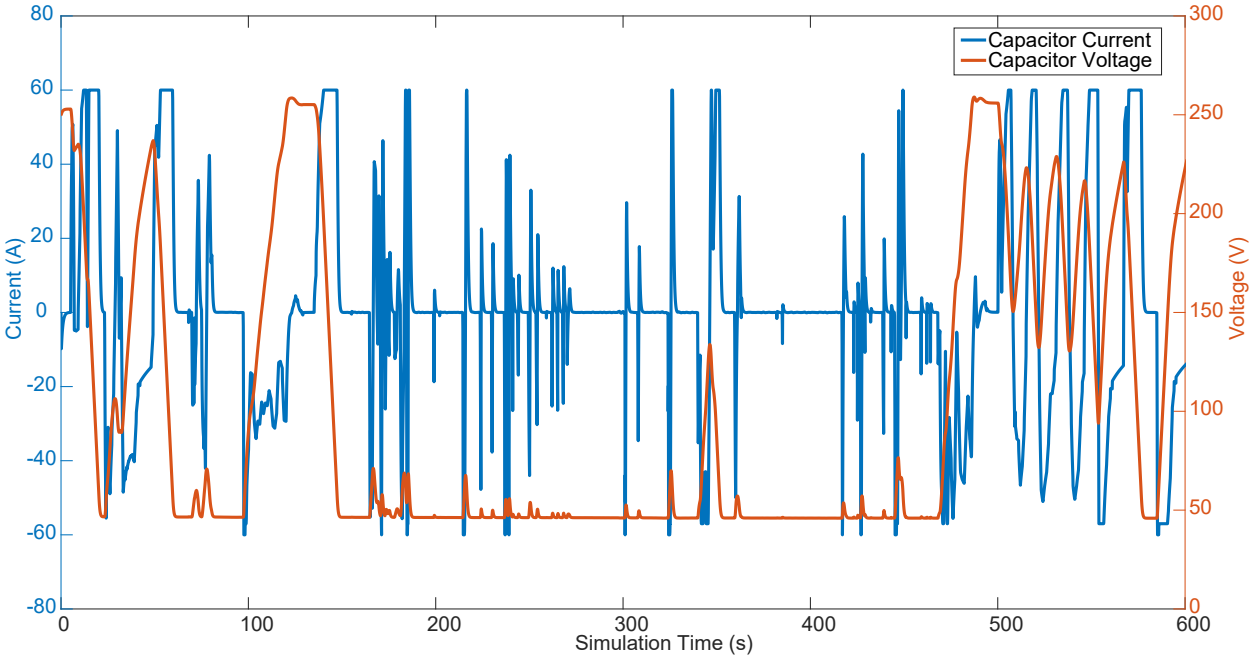


Figure 4.5: Graph showing the voltages and current sourced from the SC bank of the HESS controlled by a NN trained on the US06 profile.

are collected and compared with a NN trained on the US06 profile. When the NN is trained

Table 4.2: Percent RMS current reduction, percent max peak current reduction, and percent $PAVR/PAPR$ reduction of a NN controlled HESS trained the US06 profile.

Driving Profile	RMS Current (A)		RMS Current Reduction (%)	Peak Battery Current (A)		Max Peak Battery Current Reduction (%)
	Before	After		Before	After	
UDDS	23.48	22.69	3.39	33.52	16.97	49.38
US06	62.55	57.75	7.67	100.67	75.57	24.94
LA92	30.68	29.43	4.11	37.73	25.62	32.10
NEDC	22.07	21.77	1.36	110.62	98.90	10.60

Driving Profile	Peak Power (kW)		Average Power (kW)		PAPR		PAVR	PAPR/PAVR		PAPR/PAVR Reduction (%)
	Before	After	Before	After	Before	After		Before	After	
UDDS	42.93	34.31	5.91	5.96	7.26	5.75	2.90	2.51	1.99	20.87
US06	106.10	95.25	17.76	17.91	5.97	5.32	1.68	3.56	3.17	11.03
LA92	47.42	43.66	7.51	7.58	6.31	5.76	2.70	2.34	2.13	8.77
NEDC	42.56	38.05	5.26	5.28	8.09	7.21	3.61	2.24	1.99	10.92

on a high speed profile, the US06 profile shows an improvement in battery RMS current reduction and PAPR/PAVR reduction, however all other profiles have lower improvements

than when compared to the NN trained on the more urban UDDS profile. NN control can offer an effective solution provided it is supplied enough inputs to effectively determine the vehicle's condition.

4.4 Results Compared to Other Control Strategies

HESS control strategies are typically designed to either reduce battery RMS current and reduce battery peak current, or prolong the driving range of the vehicle. If the later is the control objective, the SC bank is much larger than the SC bank used in this thesis and is a larger energy contributor to the vehicle. The HESS in this thesis and similar HESSs use a relatively smaller SC bank which has negligible impact on additional energy available to the vehicle after charging. The results from the NN control developed in this thesis are compared against control strategies that lower battery peak and RMS currents developed by authors [11, 14, 18, 19] in table 4.3. The energy capacity of the SC banks used in this

Table 4.3: Results of the proposed control compared with HESSs designed with the similar function of reducing battery RMS and peak currents and reducing PAPR/PAVR ratios.

System	Driving Profile	SC Bank Energy Capacity (Wh)	Normalized SC Bank Energy Capacity
Proposed HESS and NN Control	UDDS	26.3	1
HESS and Rule-Based Control From [19]	"typical urban drive cycle" [19]	48.9	1.86
HESS and Rule-Based Control From [11]	UDDS	18.0	0.68
HESS and NN Control From [18]	NYC Drive Cycle	400	15.2
HESS and LPF Control From [14]	UDDS	∼55.6	2.12

System	Complexity	Battery RMS Current Reduction (%)	Max Peak Battery Current Reduction (%)	PAPR/PAVR Reduction (%)
Proposed HESS and NN Control	$O(n^2)$	15	52.5	4.15
HESS and Rule-Based Control From [19]	$O(n)$	NA	56.25	NA
HESS and Rule-Based Control From [11]	$O(n)$	19	50	NA
HESS and NN Control From [18]	$O(n^2)$	50	NA	NA
HESS and LPF Control From [14]	$O(n \log n)$	NA	NA	11.3 acceleration 6.0 deceleration

control range from $18Wh$ to $400Wh$, while a relatively small EV will have a battery with a

capacity of $24kWh$ like the one mentioned in section 2.2. It is worth noting the difficulty in comparing results between control solutions as vehicle, HESS, and simulation parameters differ greatly from one experiment to the next. Additionally the same driving profile can result in various load current waveforms depending on conditions such as vehicle weight and battery capabilities. For example, reducing the vehicle weight from 906kg to 770kg will reduce the load current RMS from $23.72A$ to $16.46A$; a reduction such as this will have a drastic impact on the performance of the HESS.

4.5 Summary

This chapter presented the progression of the genetic algorithm training the NN weights used in the control. This chapter also presented the results that the trained NN had on a vehicle's battery current, capacitor current, and capacitor voltage; where a trained NN offered a 15.15% reduction in battery RMS current, a 52.5% maximum reduction in peak battery current, and a 4.15% PAPR/PAVR reduction coupled with a $94.5kJ$ SC bank. The effectiveness of the NN trained on various profiles had over different driving profile simulations was evaluated and compared. It was found that a NN trained on a urban driving profile offered a better reduction in overall RMS currents, even if major spikes in current in highway scenarios where not mitigated. This HESS and control developed in this thesis was then compared to other works using both rule-based and optimization-based control for their developed HESSs. However, direct comparison is difficult due to varying vehicle, simulation, and HESS parameters.

Chapter 5

Conclusion

5.1 Overview

This chapter covers the work done in this thesis and present the final thoughts and findings of this thesis. Section 5.2 will discuss the results of the NN control and its effectiveness. Section 5.3 will suggest opportunity for future work and improvements to be made to this control to allow continual development of this control.

5.2 Summary

Chapter 1 discussed current stress problems associated with electric vehicle batteries and introduced the HESS as a solution to reduce these stresses. The chapter discusses the need for control of active HESS topologies and a brief explanation on the principals of NNs was given, including the theory of their operation and the mathematics on how a signal progressed through a NN. The chapter then explained various strategies to assign values to the weights of a NN, known as training the NN.

Chapter 2 presented the contributions of other researchers on HESS and the control for their respective HESS. This included work on various topologies used on vehicles of various sizes, or on HESSs used for power distribution. It discussed the advantages and drawbacks

of various controls and computational effort of each type of control strategy.

Chapter 3 presented the HESS for which control would be developed in this thesis. This chapter then described the NN that would be used to control the HESS. It discussed the inputs to the NN, the size of the NN, the effect various sizes had on the final result of control, and the method in which the NN was implemented in the model. The chapter then explained the external control used to protect elements of the HESS. Finally the genetic algorithm used to train the NN was introduced. The effect of varied parameters of the genetic algorithm were compared, and the operation of the genetic algorithm was covered.

Chapter 4 presented the progression of a NN as it was trained by a genetic algorithm and the results that the NN control had on the HESS when trained and tested on various driving profiles. Three methods to evaluate the effectiveness of control were presented and compared over various scenarios, these methods were: RMS current reduction, maximum peak battery current reduction, and $PAPR/PAVR$ reduction. The effect of the NN control over multiple aspects of the HESS was shown in various figures and the effectiveness over mixed driving profiles was compared.

5.3 Future Work

The NN control developed in this thesis provided effective control, yet there is opportunity for further improvement. When designing the NN, the optimal hyper-parameters of the NN were chosen through a repetitive process of training the NN with each combination of neurons and evaluating its performance. This method is time consuming and lacks mathematical validation. An improvement could be to derive a mathematical model to calculate the

effectiveness of a set of hyper parameters.

The trained network will function best in settings similar to the profile which it is trained. If a NN is trained in an urban environment for example, the HESS will correctly utilize the capacitor banks energy until the setting changes and the vehicle's speed increases. Adding an input to better indicate what driving situation the vehicle is in would allow the NN to train with this input and adjust the reference converter current accordingly. Future work could involve determining the correct input to achieve this adding inputs to the NN that were not available from the ADVISOR simulator could aid in NN performance.

Another potential for future improvement is providing better evaluation of a population member's effectiveness in the genetic algorithm. Currently the genetic algorithm considers the total battery RMS current value of a simulation run to determine effectiveness. Adding other evaluation factors could train a NN to perform better across various scenarios.

Driving profiles to test on were limited by what was available on ADVISOR. Generating new driving profiles with more parameters on the vehicle and its environment could aid in NN training. Implementing this solution would be desirable to allow the NN to train on the scenario an individual vehicle is placed. If this solution is to be implemented into commercial vehicles, a method of continually adjusting NN weights while the vehicle is in operation could be developed.

Bibliography

[1] M. Chen and G. A. Rincón-Mora, "Accurate electrical battery model capable of predicting runtime and I-V performance," *IEEE Transactions on Energy Conversion*, vol. 21, no. 2, pp. 504–511, 2006. → pages x, 12, 61

[2] A. Jaafar, C. Turpin, X. Roboam, E. Bru, and O. Rallieres, "Energy management of a hybrid system based on a fuel cell and a Lithium Ion battery: Experimental tests and integrated optimal design," *Mathematics and Computers in Simulation*, vol. 131, pp. 21–37, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.matcom.2016.01.007 → pages xiii, 18, 26, 27

[3] D. Shiffman, S. Fry, and Z. Marsh, *The Nature of Code*. D. Shiffman, 2012. [Online]. Available: https://books.google.ca/books?id=hoK6lgEACAAJ → pages 3, 4

[4] M. Fischer, M. Werber, and P. V. Schwartz, "Batteries: Higher energy density than gasoline?" *Energy Policy*, vol. 37, no. 7, pp. 2639–2641, 2009. → pages 10

[5] B. Xu, A. Oudalov, A. Ulbig, G. Andersson, and D. S. Kirschen, "Modeling of lithium-ion battery degradation for cell life assessment," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 1131–1140, 2018. → pages 11

[6] N. T. Jeong, S. M. Yang, K. S. Kim, M. S. Wang, H. S. Kim, and M. W. Suh, "Urban

driving cycle for performance evaluation of electric vehicles," *International Journal of Automotive Technology*, vol. 17, no. 1, pp. 145–151, 2016. [Online]. Available: https://link.springer.com/article/10.1007{%}2Fs12239-016-0014-0 → pages 12, 34

[7] T. Bruen and J. Marco, "Modelling and experimental evaluation of parallel connected lithium ion cells for an electric vehicle battery system," *Journal of Power Sources*, vol. 310, pp. 91–101, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.jpowsour.2016.01.001 → pages 12

[8] O. Salari, I. S. Member, K. H. Zaad, I. S. Member, A. Bakhshai, I. S. Member, P. Jain, and F. Ieee, "Filter Design for Energy Management Control of Hybrid Energy Storage Systems in Electric Vehicles," *9th IEEE International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, pp. 1–7, 2018. → pages 12, 17, 22, 26, 27

[9] I. Aharon and A. Kuperman, "Topological overview of powertrains for battery-powered vehicles with range extenders," *IEEE Transactions on Power Electronics*, vol. 26, no. 3, pp. 868–876, 2011. → pages 13, 14

[10] A. Ostadi, M. Kazerani, and S. K. Chen, "Hybrid Energy Storage System (HESS) in vehicular applications: A review on interfacing battery and ultra-capacitor units," *IEEE Transportation Electrification Conference and Expo*, pp. 1–7, 2013. → pages 14, 15, 17, 22

[11] B. Kerns, T. Lindsay, T. Williams, and W. Eberle, "A control algorithm to reduce electric vehicle battery pack RMS currents enabling a minimally sized supercapacitor pack,"

IEEE Transportation and Electrification Conference and Expo, pp. 376–380, 2017. → pages 17, 23, 65

[12] M. B. Camara, H. Gualous, F. Gustin, and A. Berthon, "Design and new control of DC/DC converters to share energy between supercapacitors and batteries in hybrid vehicles," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 5, pp. 2721–2735, 2008. → pages

[13] M. E. Choi, J. S. Lee, and S. W. Seo, "Real-time optimization for power management systems of a battery/supercapacitor hybrid energy storage system in electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 8, pp. 3600–3611, 2014. → pages 22, 27, 53

[14] L. Sun, N. Zhang, M. Awadallah, and P. Walker, "An innovative control strategy for a hybrid energy storage system (HESS)," *IEEE International Conference on Mechatronics*, pp. 434–439, 2017. → pages 26, 62, 65

[15] A. Tani, M. B. Camara, and B. Dakyo, "Energy management based on frequency approach for hybrid electric vehicle applications: Fuel-cell/lithium-battery and ultracapacitors," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 8, pp. 3375–3386, 2012. → pages 18, 22

[16] H. Xiaoliang and H. Yoichi, *International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium*. → pages 17, 26

[17] M. Porru, A. Serpi, I. Marongiu, and A. Damiano, "A novel hybrid energy storage

system for electric vehicles," *41st Annual Conference of the IEEE Industrial Electronics Society*, vol. 2015, pp. 3732–3737, 2015. → pages 18, 22, 29

[18] J. Ramoul, E. Chemali, L. Dorn-Gomba, and A. Emadi, "A Neural Network Energy Management Controller Applied to a Hybrid Energy Storage System using Multi-Source Inverter," *IEEE Energy Conversion Congress and Exposition, ECCE 2018*, pp. 2741–2747, 2018. → pages 18, 29, 31, 36, 50, 65

[19] R. Carter and A. Cruden, "Strategies for control of a battery/ supercapacitor system in an electric vehicle," *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, pp. 727–732, 2008. → pages 23, 65

[20] P. Garcia, J. P. Torreglosa, L. M. Fernandez, and F. Jurado, "Control strategies for high-power electric vehicles powered by hydrogen fuel cell, battery and supercapacitor," *Expert Systems with Applications*, vol. 40, no. 12, pp. 4791–4804, 2013. → pages 23

[21] B. Wei, S. Ding, J. Fang, J. Hang, and Q. Wang, "A wavelet-fuzzy power allocation strategy of battery-supercapacitor hybrid system for electric vehicles," *IEEE Conference on Industrial Electronics and Applications*, no. 2, pp. 2095–2099, 2018. → pages 24

[22] I. R. Scola, M. Patrone, and D. Feroldi, "Observer-based Controller for an Electric Vehicle with Hybrid Energy Storage," *2018 Argentine Conference on Automatic Control (AADECA)*, no. 1, pp. 1–6, 2018. → pages 25

[23] X. Li, Z. Zhao, J. Zhang, and M. Zhu, "Robust Adaptive Output Feedback Control Scheme for Chaos Synchronization with Input Nonlinearity," *Journal of Control Science and Engineering*, vol. 2016, 2016. → pages 25, 26

[24] D. Xu, J. Liu, X. G. Yan, and W. Yan, "A Novel Adaptive Neural Network Constrained Control for a Multi-Area Interconnected Power System with Hybrid Energy Storage," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 8, pp. 6625–6634, 2018. → pages 30, 31

Appendix A

Driving Profile Waveforms

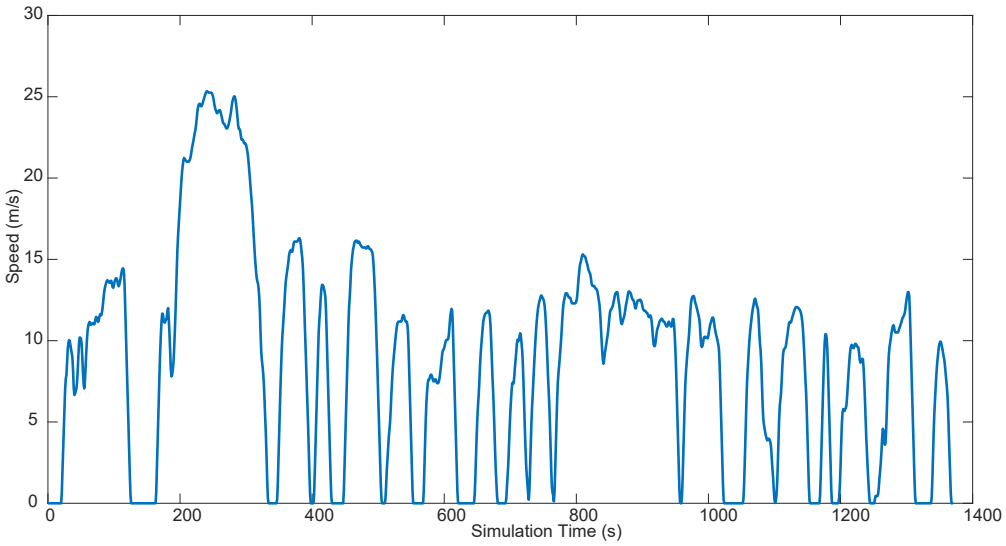


Figure A.1: Speed input of UDDS driving profile.

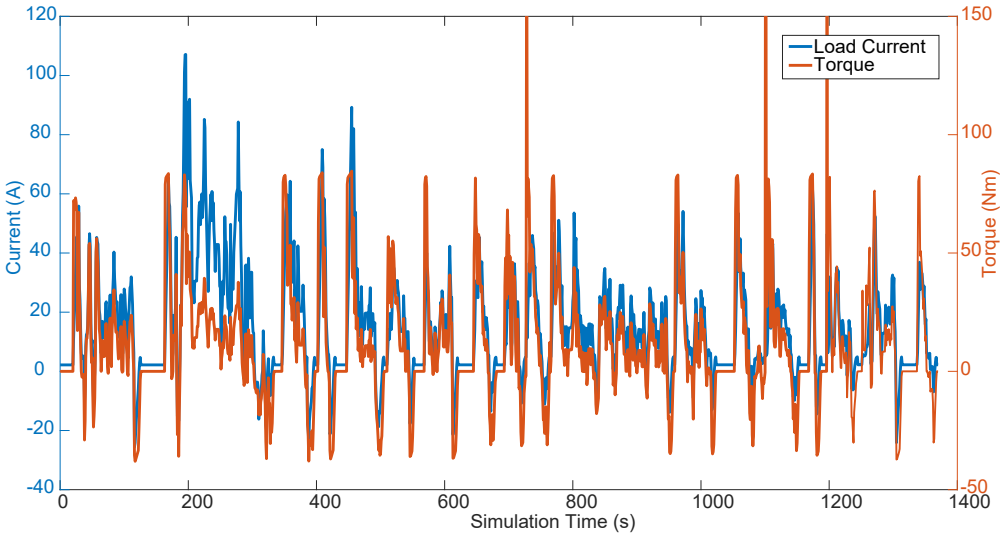


Figure A.2: Load current and torque inputs of UDDS driving profile.

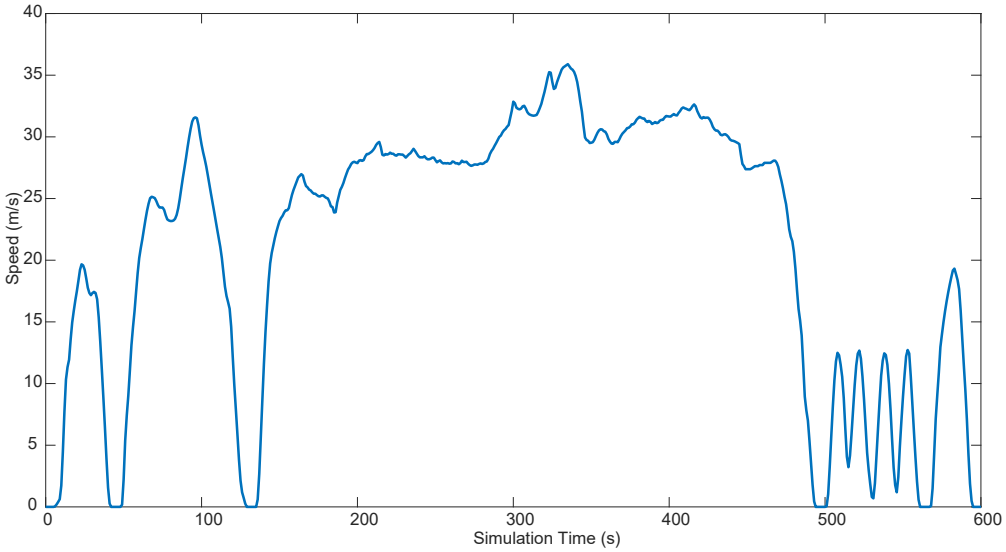


Figure A.3: Speed input of US06 driving profile.

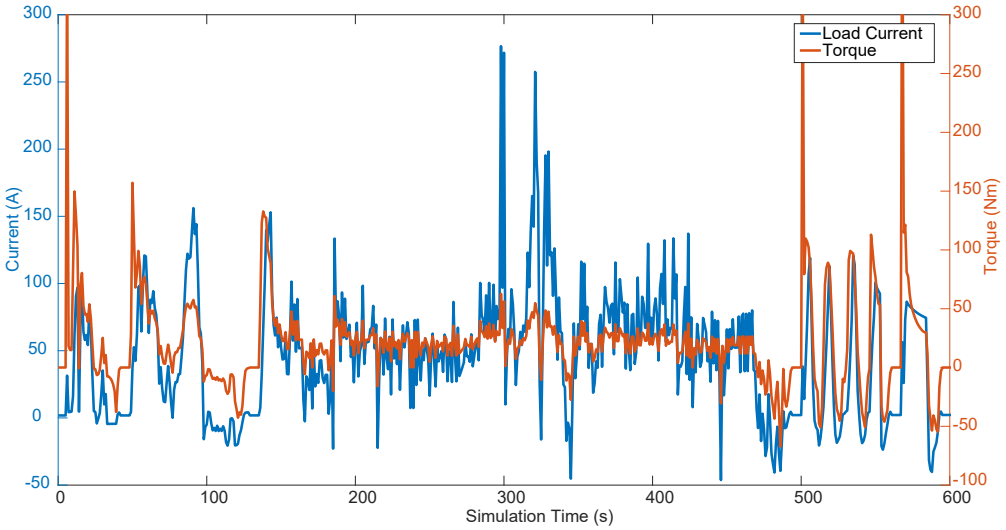


Figure A.4: Load current and torque inputs of US06 driving profile.

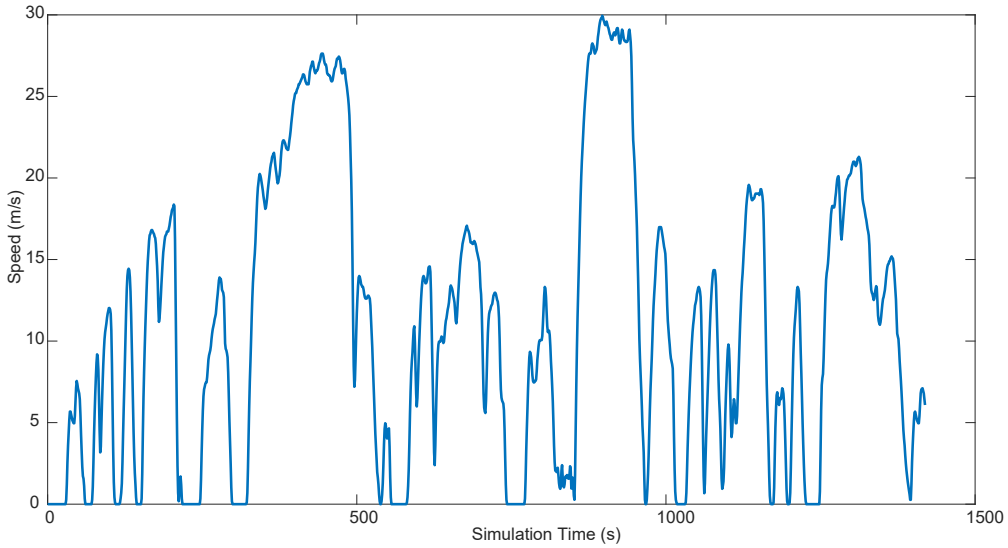


Figure A.5: Speed input of LA92 driving profile.

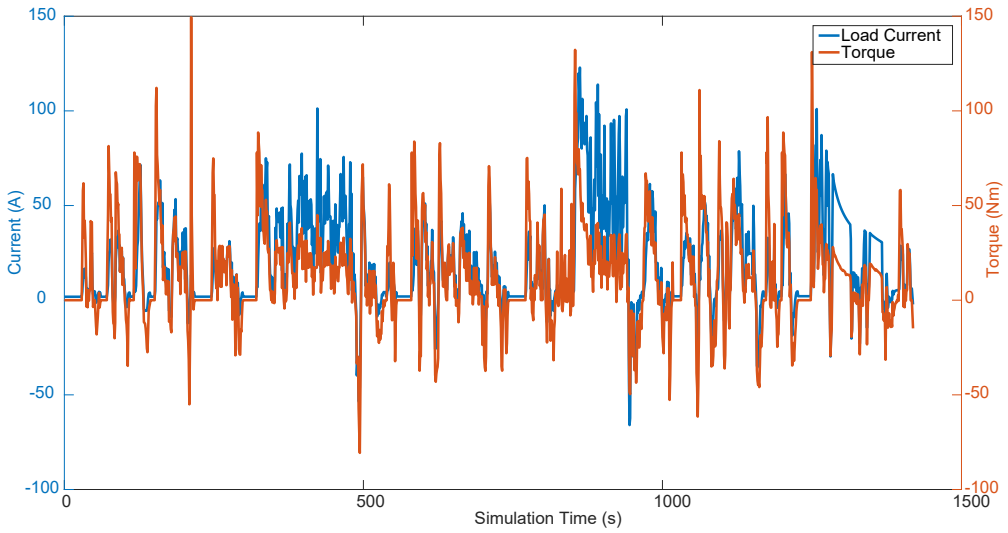


Figure A.6: Load current and torque inputs of LA92 driving profile.

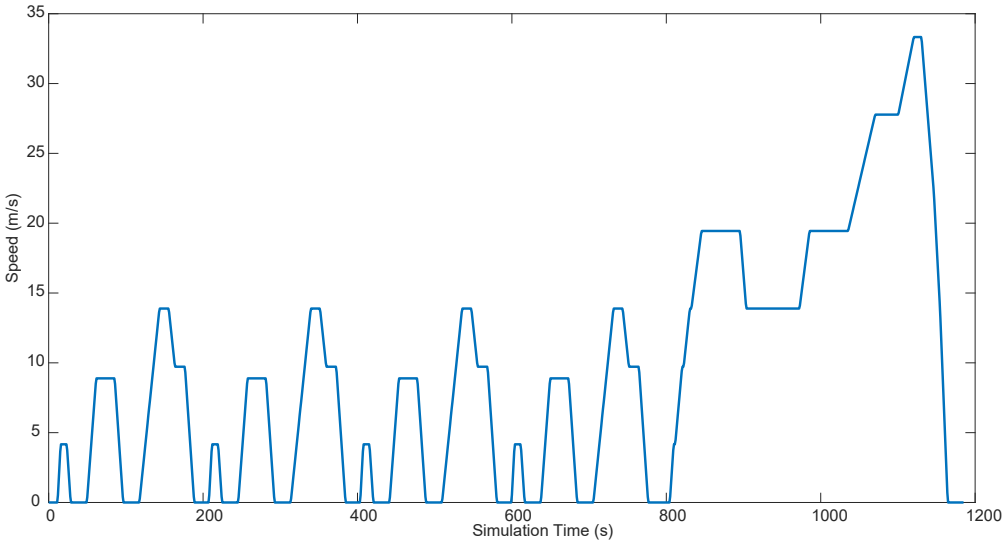


Figure A.7: Speed input of NEDC driving profile.

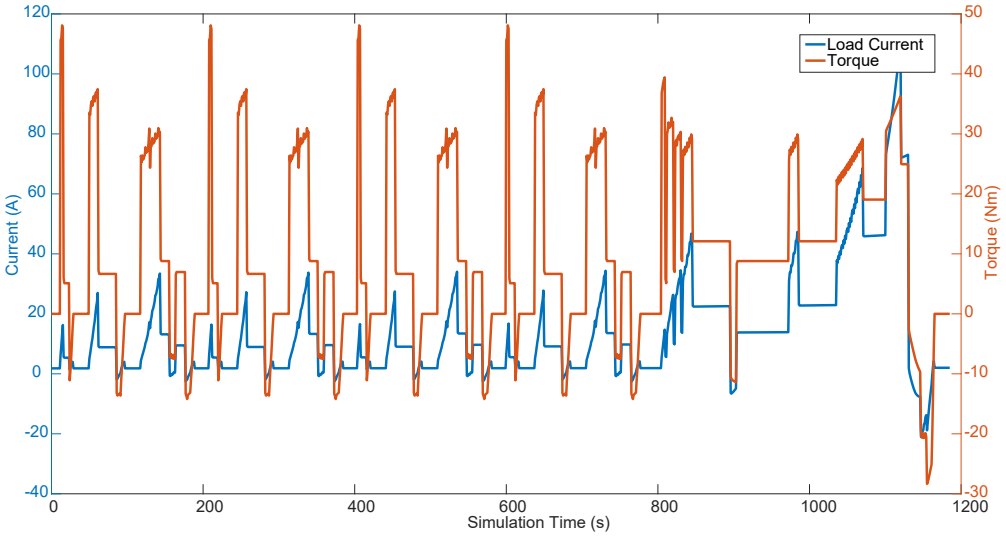


Figure A.8: Load current and torque inputs of NEDC driving profile.

Appendix B

Matlab Code

B.1 Main code for training a NN

```
%% MainDirectNN4L
% This code trains a NN through use of a genetic glgorithm.
   Requires
% ConverterDirectNN4L.slx Simlulink model to run.
clc; clear all; close all;
%% Load driving profile parameters or generate them from advisor.

prompt = 'Specify input paremters to load, otherwise hit enter to
   run advisor, then restart main. [input_parameters_XXXXX.mat]:
   ';
str = input(prompt,'s');

if isempty(str)
    run(advisor)
    load_current = [t, ess_current];    % Save load current
       profile
    speed = [t, veh_spd_a];             % Sace speed profile
    torque = [t, mc_trq_out_a];         % Save torque profile
      % Save profiles under specified save name
    save('input_parameters_NEDC', 'load_current', 'speed', 't', '
       torque')
else
    save_name = ['input_parameters_', str];
    load(save_name);            % Load parameters from existing
       save file
end

%% Setup Parameters
% Initialize the nearal network and genetic algorithm parameters

input_layer_size  = 7;
```

```
hidden_layer_1_size = 12;
hidden_layer_2_size = 9;
output_layer_size = 1;

population_size = 20;
T_step = 0.1;
converter_limit = 60;   % Limit in amperes
converter_efficiency = 0.95;
generation_limit = 200; % Number of generations to run

number_of_parents = 2;
mutation_rate = 0.01;
epsilon_init = 0.15;  % Initial randomization factor

% Number of total neurons for the NN
param_size = (input_layer_size + 1)*hidden_layer_1_size +...
    (hidden_layer_1_size + 1)*hidden_layer_2_size + ...
    (hidden_layer_2_size + 1)*output_layer_size;

sim_length = length(t); %length of Simulink simulation

%% Initialization
% Populates the neuron vector with weights, either random or
   loaded from a
% previous run

prompt = 'Randomize inital theta? Y/N [Y]: ';
str = input(prompt,'s');
if isempty(str)
    str = 'Y';
end

nn_params = zeros(param_size, population_size);
% For loop creates weight matrixes for each layer of each
   population member
if ((str == 'Y')||(str == 'y'))
    for i = 1:population_size
        initial_Theta1 = randInitializeWeights(input_layer_size,
           hidden_layer_1_size, epsilon_init);
        initial_Theta2 = randInitializeWeights(hidden_layer_1_size
           , hidden_layer_2_size, epsilon_init);
        initial_Theta3 = randInitializeWeights(hidden_layer_2_size
           , output_layer_size, epsilon_init);
        nn_params(:,i) = [initial_Theta1(:); initial_Theta2(:);
           initial_Theta3(:)];
```

```
    end
else
    % Otherwise weight matrixes are loaded from a previous
       generation
    prompt = 'Which generation number to load?: ';
    load_gen = input(prompt);
    load(['generationsDirect4L', num2str(load_gen),'.mat'])
    fprintf('Loaded file: generationsDirect4L%d.mat \n', load_gen)
    for i = 1:population_size
        initial_Theta1 = Data.generation(end).Theta1;
        initial_Theta2 = Data.generation(end).Theta2;
        initial_Theta3 = Data.generation(end).Theta3;
        nn_params(:,i) = [initial_Theta1(:); initial_Theta2(:);
           initial_Theta3(:);];
    end
    load_mut_rate = 0.01;
    % Neurons are mutated to continue algorithm progression from
       last
    % generation
    nn_params = mutate(nn_params, load_mut_rate, epsilon_init);
end
generation_num = 1;

%% Run Simulation

ButtonHandle = uicontrol('Style', 'PushButton', ...
    'String', 'Finish sim', ...
    'Callback', 'delete(gcbf)');

while (generation_num <= generation_limit)
    fprintf('Generation: %d\n', generation_num)
    fitness_bat = zeros(population_size, 1);    %create fitness
       array

    % Create gene vectors from nn_param array
    for i = 1:population_size
        Theta1 = reshape(nn_params(1:hidden_layer_1_size * (
           input_layer_size + 1),i), ...
            hidden_layer_1_size, (input_layer_size + 1));
        Theta2 = reshape(nn_params((1 + (hidden_layer_1_size * (
           input_layer_size + 1))): ...
            ((hidden_layer_1_size * (input_layer_size + 1))) + (
               hidden_layer_2_size * (hidden_layer_1_size + 1)),i)
               , ...
            hidden_layer_2_size, (hidden_layer_1_size + 1));
```

```
        Theta3 = reshape(nn_params(((1 + hidden_layer_1_size * (
            input_layer_size + 1))) + (hidden_layer_2_size * (
            hidden_layer_1_size + 1)): ...
             end, i), output_layer_size, (hidden_layer_2_size + 1))
                ;
        % Run the simulation with simulink
        sim('ConverterDirectNN4L')

        fprintf('The RMS current of member %d is: %5.2fA.\n',i,
            RMS_current_bat.data(end))
        %Save values from this population member
        currents.itteration(i).bat_current = CurrentProfiles.
            signals(1).values; % battery currents
        currents.itteration(i).cap_current = CurrentProfiles.
            signals(2).values; % capacitor currents
        currents.itteration(i).time = CurrentProfiles.time; %
            simulation time
        currents.itteration(i).fitness = RMS_current_bat.data(end)
            ; %fitness of each itteration
        currents.itteration(i).Theta1 = Theta1; %neural network
            parameters of each itteration
        currents.itteration(i).Theta2 = Theta2; %neural network
            parameters of each itteration
        currents.itteration(i).Theta3 = Theta3; %neural network
            parameters of each itteration


        fitness_bat(i) = RMS_current_bat.data(end); % save battery
             RMS current to be used as fitness
end

    index = find(fitness_bat == min(fitness_bat));
    index = index(1);
    best_of.generation(generation_num) = currents.itteration(index
        );

    %plot RMS value vs generation number
    RMS_current = zeros(size(best_of.generation));
    for i = 1:length(best_of.generation)
        RMS_current(i) = best_of.generation(i).fitness;
    end
    figure(1)
    plot([1:length(best_of.generation)], RMS_current);
    title('4 Layer Direct')
```

```
    % map the fitness between 0 and 100
    norm_fitness = mapFitness(fitness_bat);
    fprintf('Normalized fitness from RMS values. \n')

    %create mating pool
    mating_pool = createMatingPool(norm_fitness, nn_params);
    fprintf('Created mating pool. \n')

    %create next generation
    new_gen = produceOffspring(number_of_parents, mating_pool,
       population_size);
    fprintf('Created next generation. \n')

    %mutate new generation
    nn_params = mutate(new_gen, mutation_rate, epsilon_init);
    fprintf('Mutated new generation and updated nn parameters. \n
       ')

    fprintf('Generation completed. \n')

    generation_num = generation_num + 1; %progress genetic
       algorithm

    % Exit loop on keypress
    if ~ishandle(ButtonHandle)
        disp('Loop stopped by user');
        break;
    end
    pause(0.01); % A NEW LINE
end

% Save all data from entire genetic algorithm run
best_of.metadata.input_layer_size = input_layer_size;
best_of.metadata.hidden_layer_1_size = hidden_layer_1_size;
best_of.metadata.hidden_layer_2_size = hidden_layer_2_size;
best_of.metadata.output_layer_size = output_layer_size;
best_of.metadata.population_size = population_size;
best_of.metadata.T_step = T_step;
best_of.metadata.converter_limit = converter_limit;
best_of.metadata.number_of_parents = number_of_parents;
best_of.metadata.mutation_rate = mutation_rate;
best_of.metadata.epsilon_init = epsilon_init;
best_of.metadata.save_name = save_name;

save_num = SaveWithNumber('generationsDirect4L', best_of);
```

```
fprintf('Generations saved to file: generationsDirect4L%d.mat \n',
    save_num)


%% Find Best Generation
% clear unneeded variables

clearvars -except save_num T_step save_name converter_limit
   sim_length converter_efficiency; close all;
load(save_name)

% Rerun best generation
prompt = 'Run current generation? Y/N [Y]: ';
str = input(prompt,'s');
if ((str == 'n')||(str == 'N'))
    prompt = 'Which generation to load? ';
    save_num = input(prompt);
end

load(['generationsDirect4L', num2str(save_num), '.mat'])
RMS_current = zeros(size(Data.generation));

for i = 1:length(Data.generation)
    RMS_current(i) = Data.generation(i).fitness;
end

close all
plot([1:length(Data.generation)], RMS_current);
title('Genetic Algorithm Progression')
xlabel('Generation Number')
ylabel('Battery RMS Current')

[best_RMS, index] = min(RMS_current);

fprintf('The lowest RMS bat current is %5.2fA at generation number
    %3.f.\n', best_RMS, index)

Theta1 = Data.generation(index).Theta1;
Theta2 = Data.generation(index).Theta2;
Theta3 = Data.generation(index).Theta3;
converter_limit = Data.metadata.converter_limit;

fprintf('Simulation running. \n')
sim('ConverterDirectNN4L')
```

```
layer_1_weights = Weight_Evaluator(Theta1);
fprintf('Layer one weights: %d \n', layer_1_weights)

fprintf('Simulation finished.\n')
```

B.2 Randomization function.

```
function W = randInitializeWeights(L_in, L_out, epsilon_init)
%Randomly initialize the weights of a layer with L_in
%incoming connections and L_out outgoing connections

W = zeros(L_out, 1 + L_in);

W = rand(L_out, 1 + L_in) * 2 * epsilon_init - epsilon_init;
end
```

B.3 Mutate function.

```
function nn_params = mutate(new_gen, mutation_rate, epsilon_init)
% mutate a specific gene of the nueral network parameters if a
   random value
% is less than the mutation rate.

nn_params = zeros(size(new_gen));
for i = 1:size(new_gen, 2)
    threshold = rand(size(new_gen, 1), 1);

    for j = 1:size(new_gen, 1)
        if (threshold(j) < mutation_rate)
            nn_params(j,i) = rand() * 2 * epsilon_init -
               epsilon_init;
        else
            nn_params(j,i) = new_gen(j,i);
        end
    end
end
```

B.4 Map fitness function.

```
function normalized = mapFitness(fitness_bat)
% Maps fitness values between 1 and 100

max_bat = max(fitness_bat);
min_bat = min(fitness_bat);
```

```
norm_bat = (fitness_bat - min_bat)./(max_bat - min_bat);

normalized = 100*(1 - norm_bat);
end
```

B.5 Mating pool function.

```
function mating_pool = createMatingPool(norm_fitness, nn_params)
% Creates large array from neural network parameters based on
   fitness.
% Higher fitness parameters will occupy a larger portion of the
   array

mating_pool = zeros(size(nn_params, 1), 1);
for i = 1:size(norm_fitness,1)
    for j = 1:norm_fitness(i)
        mating_pool = [mating_pool, nn_params(:,i)];
    end
end
mating_pool = mating_pool(:, 2:end);
end
```

B.6 Offspring function.

```
function new_gen = produceOffspring(num_parents, mating_pool,
   population_size)
% Creates new gen of population size based on a specifiend number
   of
% parents for each child

parent = zeros(num_parents,1);
new_gen = zeros(size(mating_pool, 1),population_size);

for i = 1:population_size
    % determing three random parents from mating pool
    for k = 1:num_parents
        parent(k) = floor(size(mating_pool, 2)*rand() + 1);
    end

    % create child from parents, looping through all indices of nn
        parameters
    for j = 1:size(mating_pool, 1)
        new_gen(j,i) = mating_pool(j,parent(floor(num_parents*rand
           () + 1)));
    end
```

```
end
```

B.7 Save function.

```
function save_num = SaveWithNumber(FileName, Data)
% Function saves the generation run of the genetic algorithm


[fPath, fName, fExt] = fileparts(FileName);
if isempty(fExt) % No '.mat' in FileName
  fExt     = '.mat';
  FileName = fullfile(fPath, [fName, fExt]);
end
save_num = 1;
if exist([fName, num2str(save_num), fExt], 'file')
  % Increment file starting from base name.
  fDir     = dir(fullfile(fPath, [fName, num2str(save_num), fExt])
     );
  while(~isempty(fDir))
      save_num = save_num + 1;
      fDir     = dir(fullfile(fPath, [fName, num2str(save_num),
         fExt]));
  end
end
save([fName, num2str(save_num), fExt], 'Data');
end
```

B.8 Weight evaluator function.

```
function input_weights = Weight_Evaluator(Theta1)
% Find RMS value of weight vector for each input

input_weights = rms(Theta1,1);
end
```

B.9 Generation progression and NN weight analyser code.

```
%% Generation progression
% This program loads NNs trained on MainDirectNN4L.m and generates
    figures
% to analyse the effectiveness. It also generates PAPR/PAVR and
   RMS current
% reduction values.

clear all; clc;
```

```
prompt = 'Which profile to load? ';
save_name = input(prompt,'s');
prompt = 'Which generation to load? ';
save_num = input(prompt);

load(['input_parameters_', save_name, '.mat'])
load(['generationsDirect4L', num2str(save_num), '.mat'])
RMS_current = zeros(size(Data.generation));

for i = 1:length(Data.generation)
    RMS_current(i) = Data.generation(i).fitness;
end
close all
plot([1:length(Data.generation)], RMS_current);
title('Genetic Algorithm Progression')
xlabel('Generation Number')
ylabel('Battery RMS Current (A)')

[best_RMS, index] = min(RMS_current);

fprintf('The lowest RMS bat current is %5.2fA at generation number
    %3.f.\n', best_RMS, index)

Theta1 = Data.generation(index).Theta1;
Theta2 = Data.generation(index).Theta2;
Theta3 = Data.generation(index).Theta3;
converter_limit = Data.metadata.converter_limit;
T_step = Data.metadata.T_step;
converter_efficiency = 0.95;

sim_length = length(t);

fprintf('Simulation running. \n')
sim('ConverterDirectNN4L')
fprintf('Simulation finished. \n')

layer_1_weights = Weight_Evaluator(Theta1);
fprintf('Layer one weights: %d \n', layer_1_weights)

I_load = CurrentProfiles.signals(1).values;
I_converter = CurrentProfiles.signals(2).values;
I_bat = CurrentProfiles.signals(3).values;
time = CurrentProfiles.time;

%%
```

```
%Plot current profiles
plot(time,I_load,time,I_converter,time,I_bat,'g','linewidth',1.5);
legend('Load Current','Capacitor Current','Battery Current')
xlabel('Simulation Time (s)')
ylabel('Current (A)')
grid on

V_caps = CapacitorValues.signals(1).values;
I_caps = CapacitorValues.signals(2).values;
time = CapacitorValues.time;

% Plot SC bank current and voltage
figure(2)
yyaxis left
plot(time,I_caps,'linewidth',1.5);
ylabel('Current (A)')
ylim([-80,80])
yyaxis right
plot(time,V_caps,'linewidth',1.5);
ylabel('Voltage (V)')
xlabel('Simulation Time (s)')
legend('Capacitor Current', 'Capacitor Voltage')
xlim([0, 600])


Speed = NNinputs.signals(3).values;
local_max = NNinputs.signals(7).values;
moving_avg = zeros(size(Speed));
for i = 1:length(Speed)
    moving_avg(i) = NNinputs.signals(6).values(1,1,i);
end

%Plot speed profiles
figure(3)
plot(time,Speed,time,local_max,time,moving_avg,'g','linewidth
   ',1.5)
legend('Vehicle Speed','Local Max of Speed','Moving Average of
   Speed')
ylabel('Speed (m/s)')
xlabel('Simulation Time (s)')
ylim([0,30])

torque_nn = NNinputs.signals(5).values;
acceleration = NNinputs.signals(4).values;
```

```
%Plot Torque and Acceleration
figure(4)
yyaxis right
plot(time,torque_nn,'linewidth',1.5);
ylabel('Torque (N/m)')
ylim([-100,100])
yyaxis left
plot(time,acceleration,'linewidth',1.5);
ylabel('Acceleration (m/s^2)')
ylim([-2,2])
xlabel('Simulation Time (s)')
legend('Torque', 'Acceleration')

I_load = NNinputs.signals(1).values;

% Plot load current and capacitor voltage
figure(5)
yyaxis left
plot(time,I_load,'linewidth',1.5);
ylabel('Current (A)')
yyaxis right
plot(time,V_caps,'linewidth',1.5);
ylabel('Voltage (V)')
xlabel('Simulation Time (s)')
legend('Load Current', 'Capacitor Voltage')

% Plot load current and torque
figure(6)
yyaxis left
plot(time, I_load, 'linewidth',1.5)
ylabel('Current (A)')
yyaxis right
plot(time,torque_nn,'linewidth',1.5);
ylabel('Torque (Nm)')
xlabel('Simulation Time (s)')
legend('Load Current', 'Torque')

% Plot speed
figure(7)
plot(time, Speed, 'linewidth',1.5)
ylabel('Speed (m/s)')
xlabel('Simulation Time (s)')

%% Determine PAPR/PAVR and RMS reduction
```

```
RMS_current_before = RMS_current_load.data(end)
RMS_current_after = RMS_current_bat.data(end)
current_per_reduction = (RMS_current_load.data(end) -
    RMS_current_bat.data(end))/RMS_current_load.data(end)*100

peak_PWR_before = max(load_PWR)/1000
peak_PWR_after = max(battery_PWR)/1000

peak_speed = max(Speed);
avg_PWR_before = mean(load_PWR)/1000
avg_PWR_after = mean(battery_PWR)/1000

avg_speed = mean(Speed);

PAPR_before = peak_PWR_before/avg_PWR_before
PAPR_after = peak_PWR_after/avg_PWR_after
PAVR = peak_speed/avg_speed
PAPR_PAVR_before = PAPR_before/PAVR
PAPR_PAVR_after = PAPR_after/PAVR
PAPR_PAVR_per_reduction = (PAPR_PAVR_before - PAPR_PAVR_after)/
    PAPR_PAVR_before*100
```

Appendix C

Simulink Model

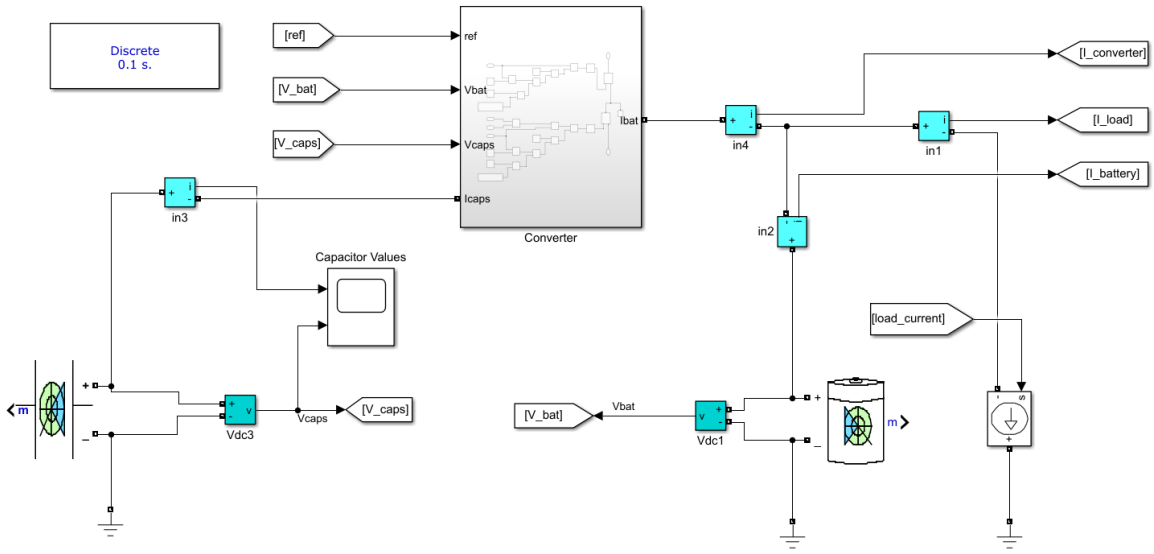


Figure C.1: Main Simulink model (HESS Layout with converter subsystem).

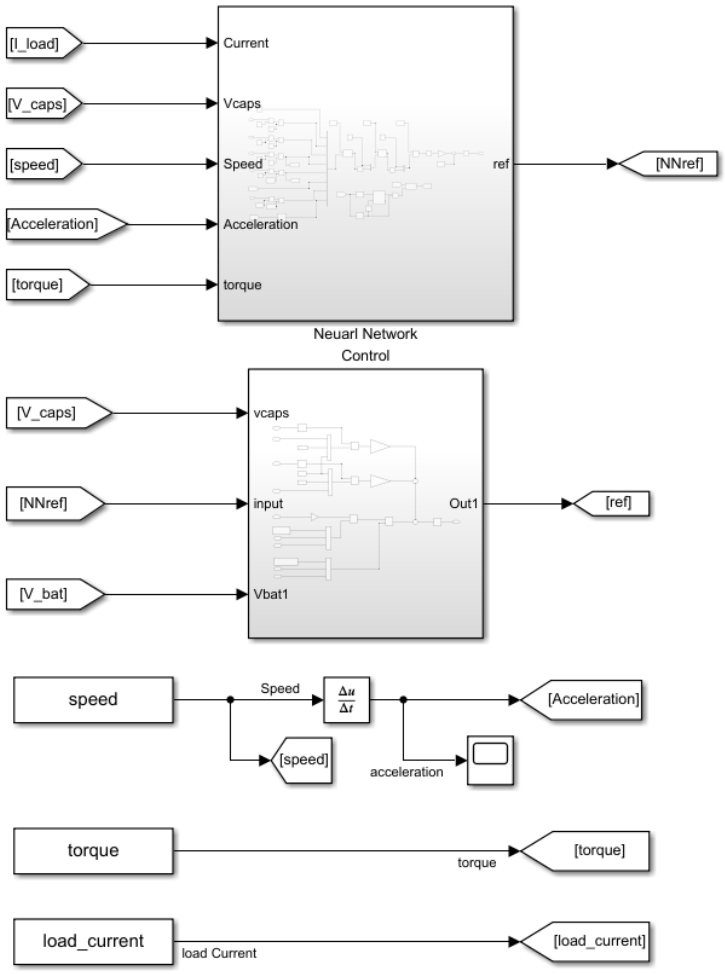


Figure C.2: Main Simulink model (inputs from Matlab workspace and Neural Network and Control subsystems.)

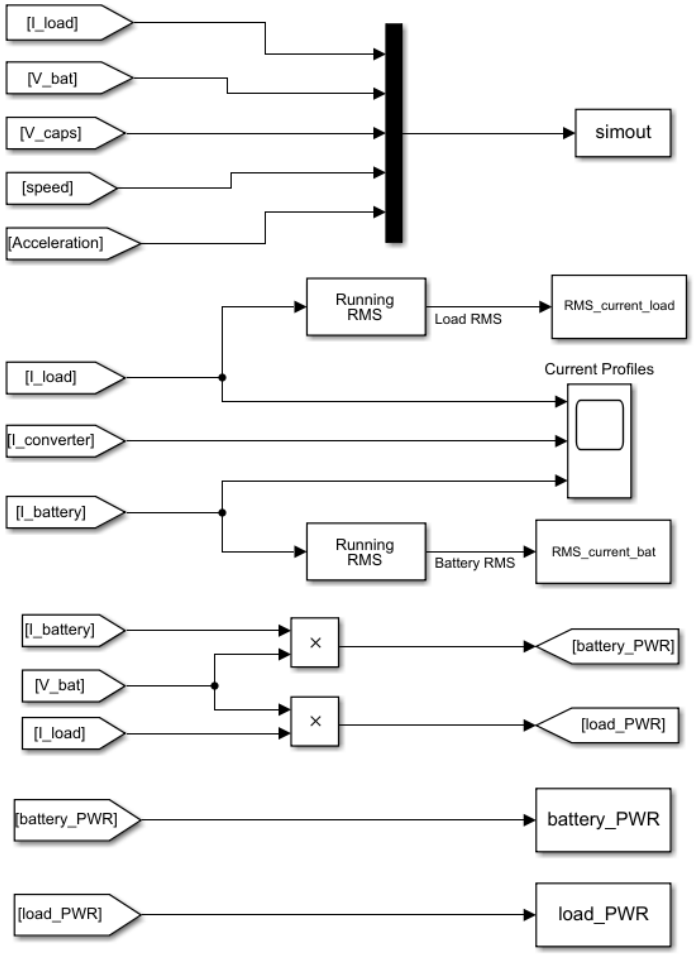


Figure C.3: Main Simulink model (output signals to workspace).

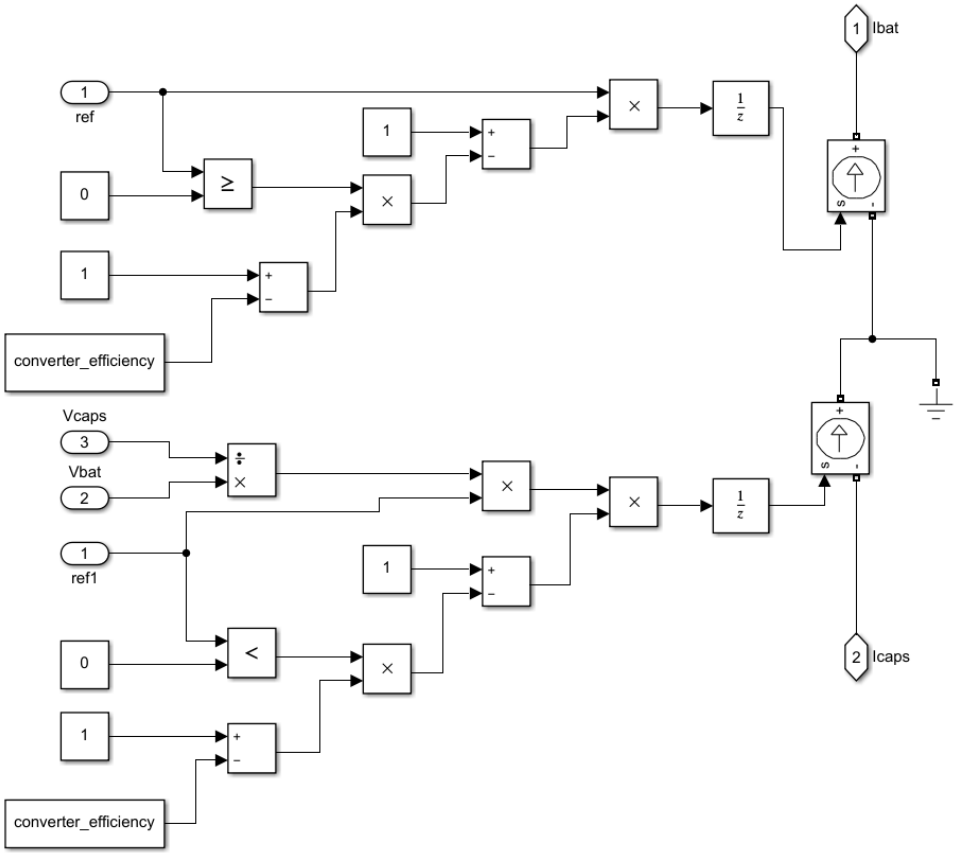


Figure C.4: Converter subsystem.

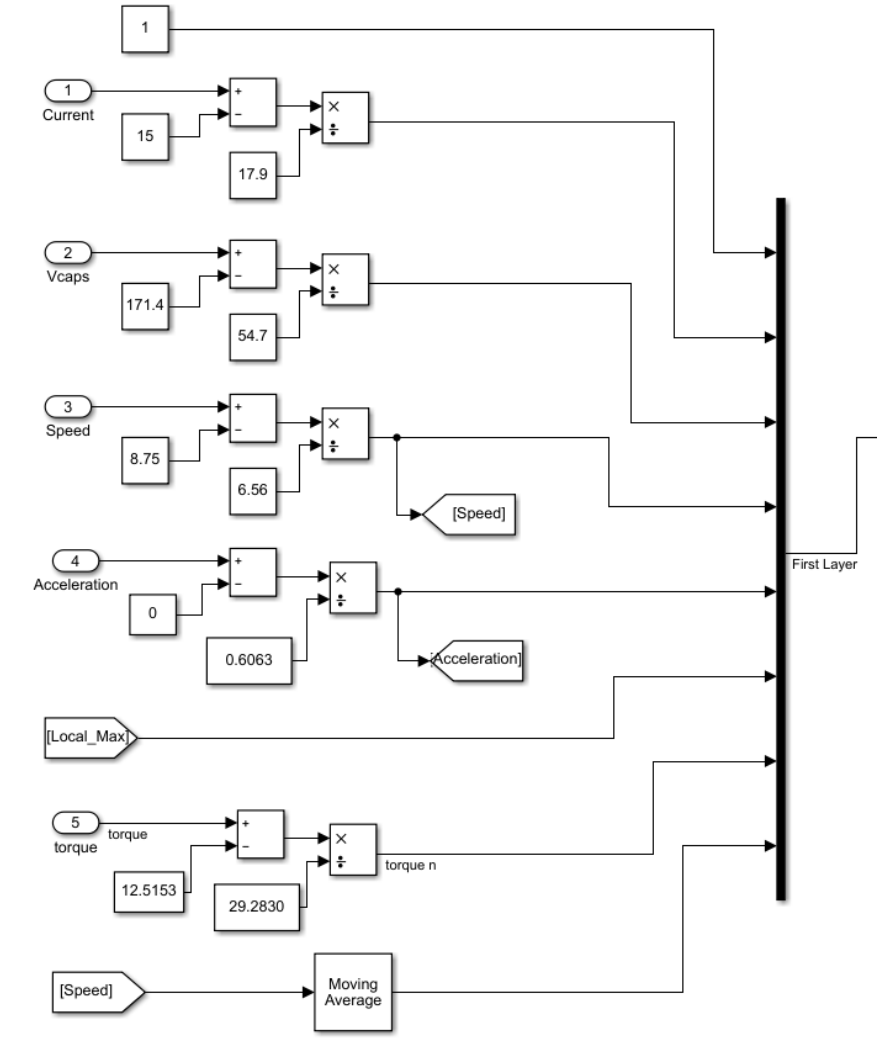


Figure C.5: Neural network subsystem input normalizing and first layer bus.

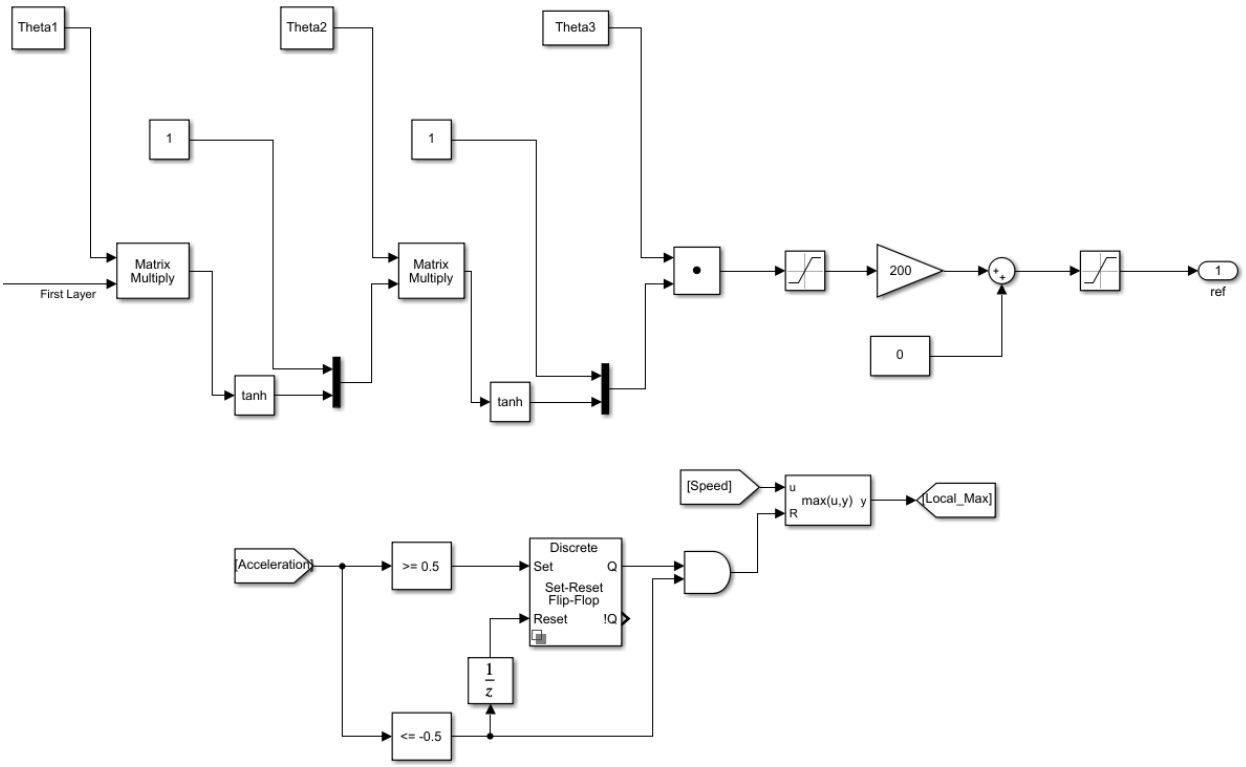


Figure C.6: Neural network subsystem internal layers, output, and local max generator control.

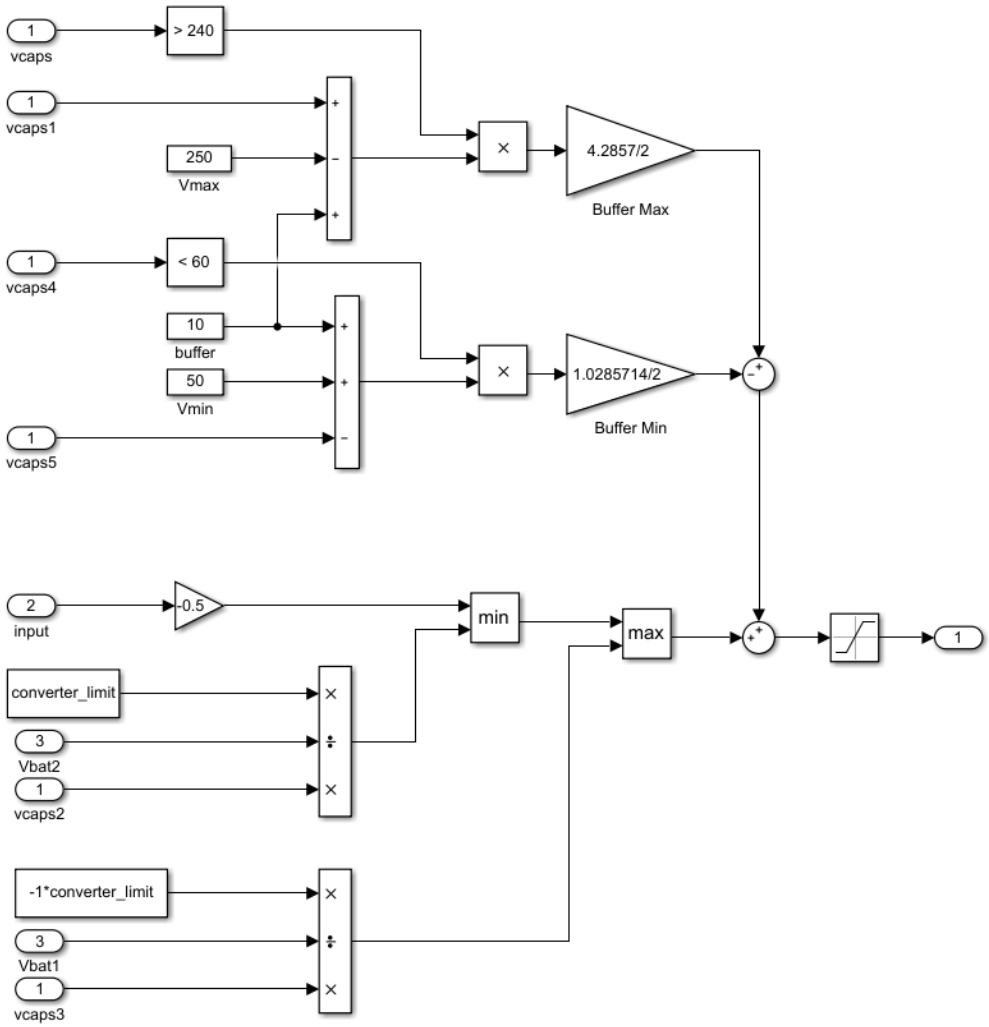


Figure C.7: SC voltage and converter current control subsystem.