# Automated Topology Synthesis and Optimization of Hybrid Electric Vehicle Powertrains

by

Adam H. Ing

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2014

## Abstract

This thesis presents a framework to automate the process of designing Hybrid Electric Vehicle (HEV) powertrain architectures. An algorithm was developed to assemble and compare all possible configurations of powertrain components. Combinatorics was used to discover all possible combinations of: an internal combustion engine, high-torque low-speed electric motor, low-torque high-speed electric motor, planetary gearset, and five-speed discrete gearbox. The Graph Theoretic Method was used to generate the powertrain models.

The powertrain models were comprised of steady-state equations in symbolic form. An optimal control strategy is required to fairly compare the different topologies because a powertrain control strategy is dependant on the configuration. Dynamic Programming was used to determine the control law that minimizes the energy consumption for a given drivecycle. Evaluating every possible topology would take an extremely long time, so topologies were evaluated using a multi-stage screening process.

The first stage examined the structure of the powertrain and used heuristics to eliminate infeasible topologies; 512 topologies were feasible.

The second stage eliminated topologies that could not meet basic driving performance; 193 topologies were feasible. Basic driving performance was tested using a section of the US06 drivecycle. The sizes of three components was optimized to ensure the topology is feasible independent of the size of the components.

The third stage eliminated topologies which could not achieve driving performance design goals; 159 could achieve the performance requirements, but only 119 were reasonably fuel efficient. The driving performance goals were implemented with a drivecycle based on the Partnership for a New Generation of Vehicles (PNGV) goals. The sizes for five components were optimized at this stage.

The 20 most fuel efficient powertrains were selected for further evaluation. Additionally, 4 common powertrains were evaluated for reference. The size of the components were optimized for a combination of the PNGV drivecycle and the HWFET drivecycle.

The most fuel efficient topology was found to be a Powersplit hybrid which has a discrete gearbox between the final drive and the powersplit device. The electric motor, planetary carrier gear, and gearbox were connected in parallel. It was found that Parallel-like, Powersplit-like, and Complex-like topologies were were the most efficient powertrain configurations. Powertrains containing two gearboxes were more efficient because the geartrain models ignored mechanical inefficiencies.

**Acknowledgements**

I would like express appreciation to my supervisor, Professor John McPhee for providing guidance, financial support and the opportunity to do this project. When things weren't going as planned, and everything seemed like it was falling apart, he remained confident in my abilities. Being available for meetings, small words of encouragement, and timely feedback made this all possible. Thank you.

My gratitude also extends to Professor Azad for being a great body of knowledge in the field of automotive and controls development. Without his help, I would keep myself up at night questioning every assumption. I acknowledge Professor Gordon Savage for volunteering his time to be on my thesis committee and being a helpful resource for linear graph theory.

I give my sincere thanks to Dr. Sam Dao of Maplesoft. If I hadn't met you before coming to Waterloo I might not have ended up in the MoRG. Thank you for the troubleshooting sessions, math models, and support with Maple/MapleSim. I thank Dr. Chad Schmitke for guidance on the limitations with Maple software.

None of this would be possible without my fellow MoRG-ians. Without Joydeep I would have not been able to decipher Maple code or learn linear graph theory. Without help from Andrew and Naser, coursework would have taken all my time and I would have never gotten to do any research. Thank you Amir for being teaching me how to Dynamic Program; I had never learned anything like that before! Thank you Aden (and later Dan!) for always being available to help with Linux problems. Xinxin, Asaad and Tony; you came after I did, but totally made my time in Waterloo a whole lot more fun ;)

Although I've never met you in person, I'd like to thank *acer*, Carl Love, and Joe Riel on the MaplePrimes website. You guys really spent a lot of time reading a total strangers' code. You have no idea how much you all have helped me, but without you all I would not have been able to complete this thesis on time.

I acknowledge my parents for birthing and raising me. I'm dedicating this thesis to you, so that's all the acknowledgement you get here.

## Dedication

This thesis is dedicated to my family, for being a constant source of emotional support and chaos in my life.

I dedicate this thesis to Ma: for encouraging my curiosity, ensuring I was raised in a nurturing environment, packing my lunches when I was busy with exams, and proofreading 20 years of school work. I cannot promise this will be the last time you proofread my work.

I dedicate this thesis to Ba: for continuously challenging me to push my own boundaries, serving as an ideal role model and mentor, and encouraging me to take risks while still having fun. You better finish your PhD before I get bored of working and start my own.

I dedicate this thesis to my brothers because you are all so different than me. To me, you all are a constant source of inspiration, emotional support and questions.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Problem Statement

Conventional transportation vehicles use combustion engines for propulsion because fossil fuels are energy dense and relatively inexpensive. Rising fuel prices, depletion of fossil fuels, and public environmental awareness have made electric vehicles an attractive alternative. Electric vehicles use electricity instead of gas for propulsion. Electric propulsion is more efficient, quieter, and does not create tailpipe emissions. Unfortunately, conventional electric vehicles are heavily reliant on batteries for energy storage. Batteries are expensive, heavy, and require special charging equipment to charge quickly. Fuel cells have been proposed as an alternative to batteries, but hydrogen fuelling infrastructure is not widely available, and high pressure hydrogen storage technology is not mature. As an intermittent step from gasoline to electric vehicles, Hybrid Electric Vehicles (HEVs) have emerged in the marketplace.

Hybrid Electric powertrains combine elements from traditional gasoline powertrains and electric powertrains. This is advantageous because they operate 10-20% more efficiently, and can still refuel at any gasoline station [6, 7, 8]. Including additional powertrain components create challenges. Complex controllers are required to ensure the components are operating efficiently. Cost and weight also increase with the number of components.

As the number of components increase, the number of possible powertrain configurations increase exponentially. Powertrain architectures can be identified through human judgement or numerical enumeration. Human judgement relies on experience, but is subject to human error and cognitive bias. Enumerative methods involve numerically generating all possible permutations, then testing each one. This potentially requires a lot of computing power to solve in a practical amount of time. As computing power becomes cheaper, enumeration becomes a more attractive option. Evaluating topologies involve building prototypes or performing modelling and simulation. Mathematical models are faster and cheaper to create than building physical prototypes, but in the words of George E. P. Box, *"essentially, all models are wrong, but some are useful"*. Thus, prototypes will still need to be created and iterated. Math models allow us to reduce the number of prototypes built by identifying optimal starting points.

## 1.2 Research Objective

The objective of this research is to create a tool can be used to: (i) compare Hybrid Electric powertrains, and (ii) identify novel Hybrid Electric powertrains. Due to time constraints, this research shows proof of concept by limiting the components to: 1 large electric motor, 1 small electric motor, 1 engine, 1 planetary gearset, and 1 5-speed gearbox. The resultant tool is an open ended framework, in which any number of components could be added.

## 1.3 Approach

This thesis proposes a method to automate the synthesis and optimization of Hybrid Electric Vehicle powertrains. Liu [9], first proposed similar work focusing on the automated synthesis of planar mechanisms, and named this computer implementation 'AutoSyn'. As tribute, this work will be known as ASHev — AutoSyn for Hybrid Electric Vehicles.

In ASHev, each topology is represented as a 'genome'. A genome is a string of '1s' and '0s' that contains all the topological information for a powertrain. Enumerating a genome string yields a impractically large set of genomes. Heuristics are employed to reduce this set to a practical number. These genomes will be parsed into an Incidence Matrix which describes the topology of the powertrain. The Graph Theoretic Method (GTM) [10, 11, 12] can then be used to generate the system equations for the given genome.

GTM is used to generate the system interconnection equations, and is augmented by the appropriate component equations to create the system equations. Using GTM is advantageous because the acausal nature of the equations is flexible for solving topological equations. This system of equations is then systematically analysed for appropriate control variables, and the system is solved symbolically in terms of these variables. This symbolic solution is a backwards (system behaviour is defined first, then the corresponding component behaviour is calculated) steady-state model of the Hybrid Electric powertrain.

Using this backwards steady-state model, the vehicle powertrain is simulated using a drivecycle (experimentally determined average driving behaviour). As a powertrains' efficiency is highly dependant on its controller, an optimal control law approach is used to ensure the topologies compared fairly. Dynamic Programming [13, 14, 15, 16] is used to find the optimal control law which minimizes the energy consumption for each topology.

The powertrain is further improved by optimizing the component sizes. The Pattern Search optimization algorithm is used because it is a global optimization method that does not require gradient information. As an initial screening step, the powertrain is optimized for a drivecycle based on the Partnership for the New Generation of Vehicles (PNGV) goals [17, 18, 19]. The top 20 topologies are chosen, as well as some reference topologies, the optimized for a combination of the PNGV and HWFET drivecycle.

## 1.4 Document Overview

This is the first chapter which introduces the thesis. The second chapter contains a literature review of: Hybrid Electric Vehicles powertrains, modelling techniques, and topology search methods. The third chapter discusses the equations that govern the powertrain components. The fourth chapter explains the process of assembling the component equations into a system model using the Graph Theoretic Method; an example of a Powersplit Hybrid powertrain is given. The fifth chapter validates two system models assembled with the GTM by comparing them to the Autnonmie software package. The sixth chapter describes the overall methodology and implementation of ASHev, including: Dynamic Programming, cluster computing, and the Pattern Search optimization algorithm. The seventh chapter describes the screening processes, and presents 24 efficient topologies. The final chapter concludes the thesis, provides lessons learned, and possible future work.

# Chapter 2

# Literature Review

This chapter starts by defining the degrees of of powertrain hybridization. Common Hybrid Electric Vehicle powertrain topologies are described, and a few examples of unique topologies are given. Modelling methods previously applied to HEV powertrains are explained. Finally, methods for generating and evaluating new hybrid powertrain configurations are reviewed.

## 2.1 Hybrid Electric Vehicles

HEV powertrains can be comprised of different components in different configurations. Certain configurations have multiple operating modes; they can achieve propulsion, braking, or charge the battery in multiple ways. A powertrain with more operating modes generally will operate have a higher overall efficiency. Also, a powertrain with more operating modes will have a higher the 'degree of hybridization'. The size of the components can also affect the number of operating modes; generally, larger components allow for additional functionality. This section will describe the difference between micro, mild, full and plug-in hybrids, as well as describe a few common and commercially available HEV architectures.

### 2.1.1 Degrees of Powertrain Hybridization

While the minimum requirement for a powertrain to be considered a hybrid powertrain is to have both gasoline powered and electric components, there are many levels of hybridization.

This section will highlight the differences between the degrees of hybridization.

**Micro-Hybrids** are the lowest level of hybridization, characterized by engine stop-start operation and limited regenerative braking. **Engine stop-start** operation enables the engine to stop when idling (the vehicle is not moving), then immediately resume by using the small electric motor (powered by the battery) to crank the engine. The greatest fuel savings are achieved when there are frequent stops, such as in urban driving scenarios. **Regenerative braking** can be achieved by using the eletric motor to capture the kinetic energy from braking an converting it to electrical energy which can be stored in the battery [20, 21]. The cost of implementing a micro-hybrid system is comparatively small; a typical motor is 2.5kW and can operate on a standard 12V battery. In city driving, Micro-HEVs typically reduce energy consumption by 5-10% [8]. The Citroen C3 is an example of a micro hybrid.

**Mild and Medium Hybrids** can operate as a Micro-HEVs, but the electric motor can provide substantial tractive power. For a sedan passenger vehicle, a typical electric motor peak output is 10-20kW. Mild hybrids provide tractive power only at low speeds. Medium hybrids can provide tractive power at low and high speeds and most braking energy is done regeneratively [21]. Mild and Medium hybrids require more power, thus their batteries operate at higher voltages of 100-200V. In city driving, energy savings are typically 20-30%. Vehicle cost will also increase 20-30%. Honda Civic Hybrid and Honda Insight are examples of Mild hybrids [8].

**Full or Power Assist Hybrids** can operate as fully electric vehicles for short distances. Operating as an electric vehicle requires a larger motor and a high voltage battery. Typical electric motors are 50kW [8], which use batteries that operate at 100-300V. Full Hybrids are 30-50% more fuel efficient, but cost 30-40% more than a conventional gasoline vehicle. Full Hybrid powertrains can be tuned for improved fuel efficiency, as (i.e. the Toyota Prius), or for improved driving performance, (i.e. the Toyota Highlander) [8, 21].

**Plug-in Hybrids** can operate as an electric vehicle for 30-60km. Like all hybrids, they can use any gasoline station to refuel. However, they can also recharge their batteries at a home or commercial charging stations [8]. Driving long distances requires large batteries. If only short distances were driven, one could potentially never use gasoline [21].

### 2.1.2 Hybrid Electric Powertrain Architectures

As previously mentioned, HEVs have both electric and gasoline powertrain components. These components can be connected in different configurations to create architectures, each with unique properties and operating modes. Shown in figure 2.1 are three common

HEV architectures: Parallel, Series, and Powersplit. This section will compare and contrast the differences between hybrids, as well as introduce a few commercially available configurations.

## Common Hybrid Electric Powertrain Architectures

**Parallel Hybrids** (figure 2.1 A) have the simplest powertrain architecture because they have the fewest components, and structure a structure simular to a traditional gasoline vehicle. The powertrain coupling in this hybrid is mechanical; the engine is connected through a gearbox to the wheels, which shares a drive shaft with an electric motor. This allows the engine and/or electric motor to provide tractive force to the wheels. The engine will burn gasoline to create tractive force, and the electric motor can assist by using energy from the battery. The main advantage of this architecture is its simplicity; there are few added components making this a cheap, light alternative. In certain cases, Parallel powertrains can be more efficient than other hybrid powertrains because there is no transformation of energy from mechanical to electric power. However, there is less control over the engine operating conditions because the engine is mechanically connected to the wheels, and therefore the engine speed which minimizes fuel consumption can not always be selected.

**Series Hybrids** (figure 2.1 B) are also known as range extended electric vehicles. This type of hybridization can be found in diesel-electric locomotives and Hybrid Electric buses. The coupling in this hybrid is electrical; the Series hybrid effectively runs as an electric vehicle by powering the electric motor with the battery. When the battery's State-Of-Charge is low, the gasoline engine will crank the generator to recharge the battery. The main advantage of this architecture is that the engine can operate at the most efficient speed at all times.

**Powersplit Hybrids** (figure 2.1 C) are also known as Series-Parallel Hybrids because they can operate both as a Series Hybrid or a Parallel Hybrid. The Toyota Prius is built upon this architecture. The Powersplit Hybrid uses a planetary gearset which allows multiple degrees of freedom during operation. For example, in a Prius-like architecture, if the sun gear is locked, the planetary gearset will act as a single speed gear, and the engine will provide traction out of the ring gear to the final drive, like a Parallel Hybrid. Conversely, if the sun gear is not locked, the engine will turn the generator to generate electricity like a Series Hybrid [22, 23]. This flexibility makes the Powersplit Hybrid powertrain very efficient and thus has been adopted, with some modifications, for commercial application by Toyota, General Motors and Ford.

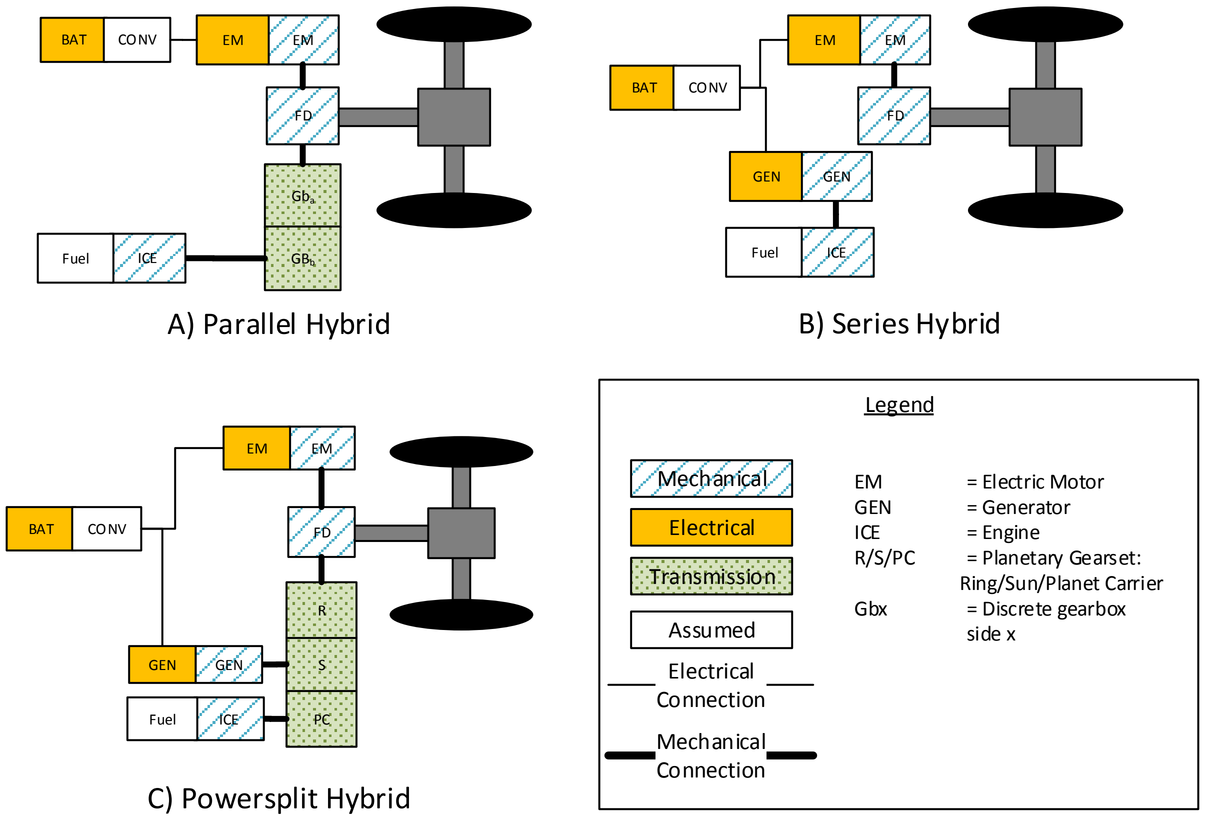

Figure 2.1: Common Hybrid Electric Vehicle Architectures.

### Specific Hybrid Electric Powertrain Architectures

Individual automotive manufacturers have made improvements to the original common hybrid powertrain architectures. Shown in figure 2.2 are the architectures created by Ford, GM-Allison, as well a 'complex' architecture.

Shown in figure 2.2 A, the **Ford Hybrid System** (FHS) is very similar to the Toyota Hybrid System I & II [23] (Powersplit architecture) developed by Toyota. The main difference is the presence of additional reduction gears (N1, N2, N3) at the final drive transmission [24].

Shown in figure 2.2 B, the **GM-Allsion Hybrid system** (AHS) contains two planetary gears mechanically joined by the planetary carrier and three clutches. The ring gears are connected in parallel with a electric generator, and separated by two clutches connected

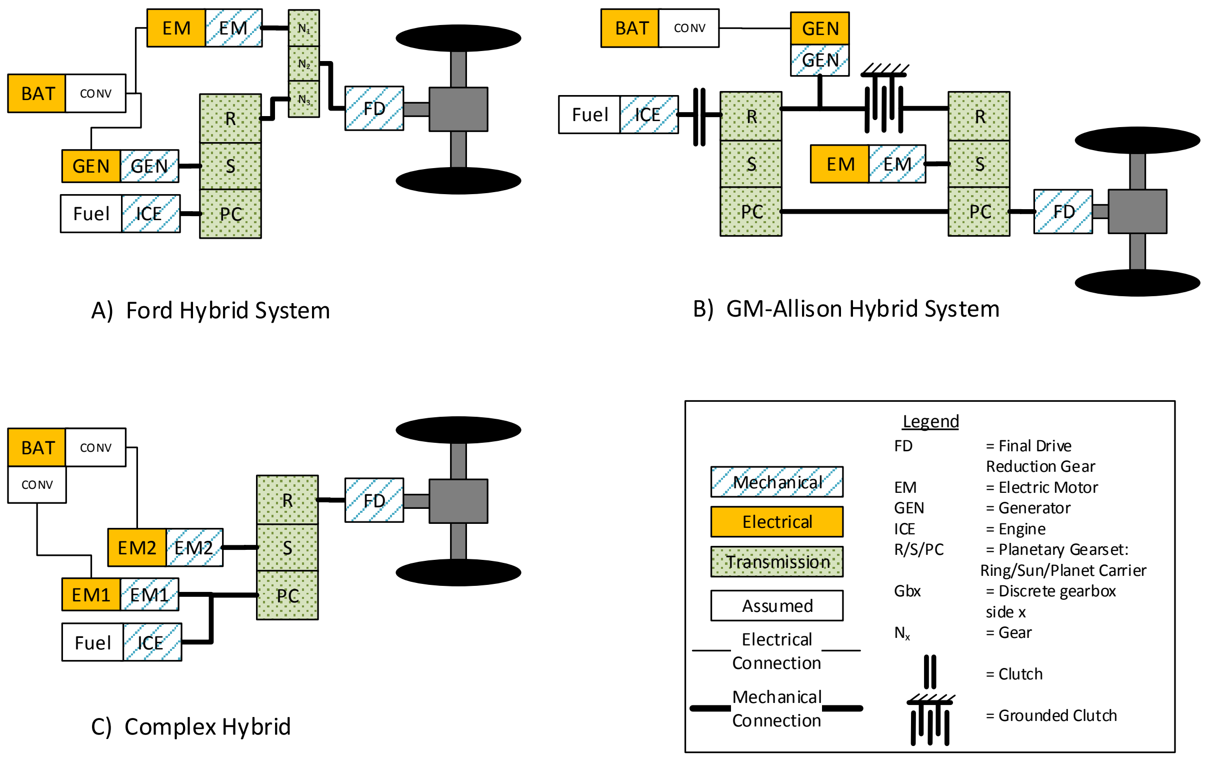

Figure 2.2: Uncommon Hybrid Electric Vehicle Architectures.

to ground. More operating modes are made available by opening and closing the clutches, and thus this architecture can achieve higher overall efficiency than the Powersplit Hybrid architecture. This system is advantageous because it has many operating modes, including: stationary charging, a Series-like low speed and reverse mode, and a high speed mode. However, the increased number of components leads to higher controller complexity and cost [24, 23, 22].

The **Complex hybrid** in 2.2 C, is similar to the Series-Parallel architecture in many ways; the key difference is that the generator is typically used both as a motor and generator. This requires the electric motor and the generator to have separate converters. This architecture has been adopted by some 4 wheel drive systems [18].

## 2.2 Modelling Methodology

Vehicles can be modelled at various levels of detail. A model's fidelity is dependent on the application; for conceptual design, a low fidelity model is sufficient, whereas a high fidelity model is usually required for control design and detailed vehicle performance. The most suitable modelling methodology is dependent on the model application. For example, Chan et al. [25] has suggested that a backwards, functional, quasi-static, causal model may be appropriate for global energy management problems (i.e., minimizing the fuel consumption of a vehicle for a drivecycle). Conversely, it has been suggested that a forward, functional, dynamic, acausal model may be appropriate for local energy management problems, such as subsystem power optimization [25]. A low fidelity model that solves quickly with few errors is best suited for topology optimization of a hybrid electric powertrain because many different configurations must be examined. In this section, model classification is explained, a number of modelling approaches are described, and approaches to modelling HEV powertrains are introduced.

### 2.2.1 Model Classification

#### Steady-state Dynamic and Quasi-static models

A model's fidelity is dependant on the equation structure and how time is represented. Models can be represented in steady-state, dynamic and quasi-static form. Steady-state models are simple because the system behaviour is captured as discrete snapshots in time. Transient behaviour is assumed to be negligible, and is often based on empirical data in the form of lookup tables. The equations in steady-state models are often linear functions. Dynamic models are more complex; the model assumes that time is continuous, and will capture all dynamic behaviour. Dynamic models often employ differential equations, which require more computational power to solve.

Quasi-static models combine dynamic and static models; the system model includes both steady-state and differential equations [25]. Engine operation is extremely complex, and thus a simplified model that approximates fuel consumption using a steady-state map is commonly used. The vehicle dynamics and engine speed are modelled with differential equations. Together the differential equations describing vehicle dynamics and the steady-state fuel consumption approximations create a quasi-static system model.

## Structural and Functional Models

Structural and functional models use mathematics to predict the behaviour of a system but are implemented in different ways. Most systems can be modelled using a structural or functional model, so the choice is dependant on the application and user preference. Structural models focus on the relationships between components, rather than the individual components. Structural models can be used to quickly create systems by defining the relationship between components. The software will interpret the structure and automatically generate the system equations. Most 'drag-and-drop' software packages, such as Autonomie [5] and MapleSim [26], use structural modelling.

Functional models focus on mathematical equations (or code) to represent the components and their connections. Functional models model are written using mathematical relationships, will solve quicker, and allow more flexible relationships [25]. However, it is often difficult to discern or change a system structure by looking at mathematical relationships, and errors can be made because there are no constraints to what relationships can be made.

## Forward and Backwards Models

Forward and backward models refer to the direction in which the dynamics are calculated. Forward models, also known in the automotive industry as 'engine-to-wheel' models, start with a component generating a force/torque, and the system behaviour is subsequently observed. For example, a forward model of a traditional ICE automobile would start by specifying the engine torque, and subsequently calculating the vehicle acceleration. Forward models better represent the reality of the system, and can be useful for controls development. Backwards models, also known as 'wheel-to-engine' models, start with a specified behaviour of the system, and the required input from the components to create that behaviour is back-calculated. For example, if a backward model of the same automobile was created, the vehicle speed would be specified and subsequently the engine torque to achieve that speed would be calculated. As it is simpler to know the expected outcome of the system, backward models are popular for conceptual system design [25]. Autonomie is an example of software that generates both forward and backward models [5].

## Causal and Acausal Models

Causal models use the principle of cause and effect. Causal models always have an output which is a function of the system input. Conversely, acausal models assume that relationships are bidirectional; inputs and outputs are easily interchangeable. For this reason software packages such as Modellica [27], MapleSim[26], or Mathworks SimScape [28] that use structural models often use acasual relationships [25].

### 2.2.2 Model Representations

Models are simplifications of reality. Different simplifications yield different model representations. Some model representations, such as block diagrams, are focused on the computation of the model and neglect the physical structure real system. Other model representations such as bond graphs and linear graphs focus on the structure of the physical system and use computer algorithms to generate the system equations. Each representation has a different graphical appearance and equation formulation.

#### Block Diagrams

Block diagrams are often used for controls development, especially for linear time-independent systems [1, 29]. Block diagrams are shown in the order in which they are executed and not the physical structure of the system. The components of the system are often mathematically described using state-space notation. Block diagrams are casual; there is no feedback between components unless a feedback loop is specifically designed into the model [2]. Transfer functions are derived from differential equations to linearize the system, resulting in a causal model. Mathworks Simulink is an industry accepted software package that uses block diagram to represent models.

#### Bond Graphs

Bond graphs can be used to model multi-domain systems, but are best suited for electromechanical systems. Bond graphs are causal models, and have a structured procedure for assigning causality [2].

Bond graphs describe energy interchange as effort and flow variables, which have the product of power. For example, in an electrical system, the effort variable would be voltage and the flow variable would be current. In a mechanical system, force would be the effort variable and velocity would be the flow variable.

Sources and Sinks (S) add or remove effort (Se) and flow (Sf). Resistance is a dissipative (R) element that energy and relates effort to flow. Similar to how a capacitor can

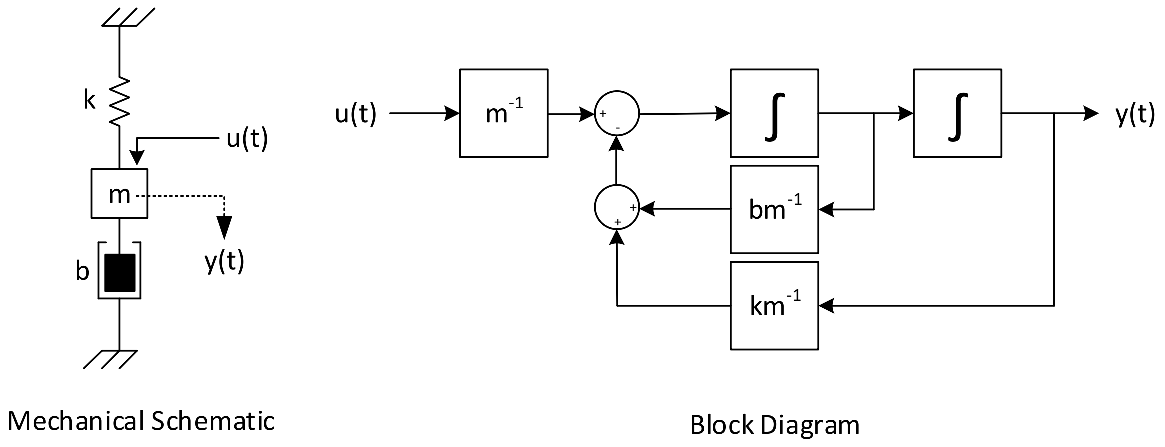

Figure 2.3: Mechanical schematic and Block Diagram of Spring-Mass-Damper system, adapted from [1].

accumulate a charge, flow and effort (C and I) can be stored. Transformers (TF) and Gyrators (GY) transform effort and flow from one domain to another. Some sources can be modulated (M) by an external signal, meaning the value is a predetermined user value [30, 2].

Junctions are used to build a model structure. There are two types of junctions: 0 junctions and 1 junctions. 0-junctions have their flows sum to zero. 1-junctions have their efforts sum to zero. A schematic and bond diagram for an electromechanical DC motor is shown in figure 2.4.

### Energetic Macroscopic Representation

Energetic Macroscopic Representation (EMR) is a causal method for modelling electromechanical systems [31, 3]. Compared to Bond Graphs, EMR is focused more on the system function.

EMR is based on the action-reaction principle. The product of an action and its reaction is power regardless of the domain. Power is transmitted between connected elements by a combination of action and reaction. In an example given by Chen [32]: *"if a current source (s1) is connected to a capacitor (s2), the action of s1 is its current and the voltage is its reaction by s2"*. The action-reaction principle can be described using most causal

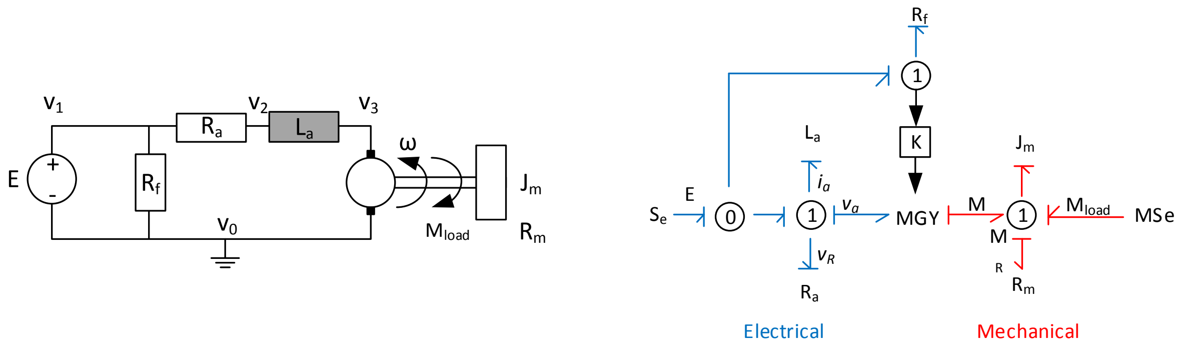

Figure 2.4: Electromechanical schematic and Bond graph of a shunt motor [2].

modeling tools such as transfer functions. Shown in figure 2.5 is a simplified EMR of an elevator with a counterweight, adapted from Barrade et al. [3].

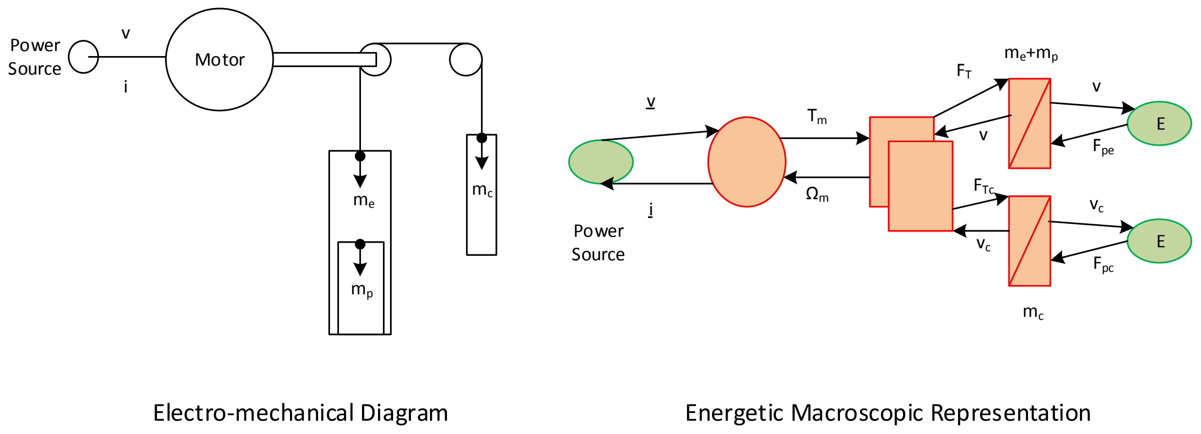

Figure 2.5: Electromechanical Diagram and Energetic Macroscopic Representation an Elevator with Counterweight [3].

EMR has been adapted by Chen et al. [6, 33] to create a framework which could model Powersplit/Series/Parallel/BEV/ICEV topologies by changing a set of Boolean values, which represented the interconnections between components. The system was represented using a backward-dynamic quasi-static model.

### Power Oriented Graphs

Power Oriented Graphs (POG) were developed from Bond Graphs, but have different graphical notation and terminology [4]. Bond graphs are well suited for systems represented in state-space form. This form is particularly well suited for implemented using Simulink.

Like Bond Graphs, the principle of power being exchanged between connected components is applied, and there are two "conjugate variables" who have the product of power flow [34]. Elaboration blocks are used to store and dissipate energy (for example: springs, masses and dampers), and connection blocks are used to transform energy (for example: gear reduction or electrical inverter) [25, 35, 4]. An example POG of a DC motor is shown in figure 2.6.

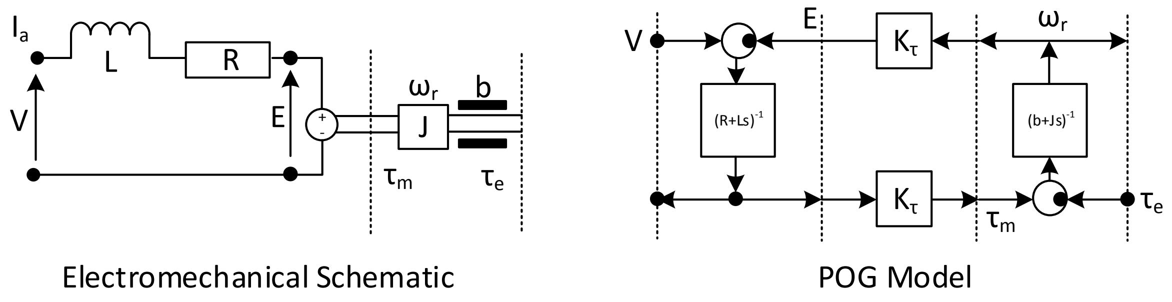

Figure 2.6: Electromechanical Schematic and Power Oriented Graph of a DC Motor Adapted from [4].

POG has been used to model many automotive systems, such as: transmission, control systems, and electric motors [36, 37, 35, 4].

## 2.2.3 Graph Theoretic Method

The mathematician Euler used Linear Graphs to represent system topologies in the 1700s. In this thesis, Graph Theoretic Method (GTM) is applied and will be described in greater detail in section 4.1. Electrical, mechanical, hydraulic, and electrochemical domains have been modelled using the GTM. DynaFlex Pro and MapleSim are examples of software packages that use the Graph Theoretic Method to generate equations. The GTM was used to model tire dynamics for various topologies in Dynaflex pro, and was validated against a verified model in MSC.ADAMS [38].

Briefly, the topology of a system can be described using a linear graph that pictorially represents the structure using lines and circles; circles are called nodes, and lines are called edges. Edges represent components, and circles represent the component terminals. The direction of measurements are defined by specifying a direction in the graph; the direction of the relationship is represented with an arrowhead. Shown in figure 2.7, a mechanical gear train is represented as a linear graph [39].

System equations are obtained by describing the linear graph as an incidence matrix, then applying the GTM [40, 41, 12, 11]. The GTM is applied to incidence matrix to generate interconnection equations which describe the relationships between measurements. Component behaviour is described by including constitutive equations which are dependant on the physical domain. The system equations can be obtained by combining the constitutive equations and the interconnection equations. The result is an acausal symbolic system of equations.

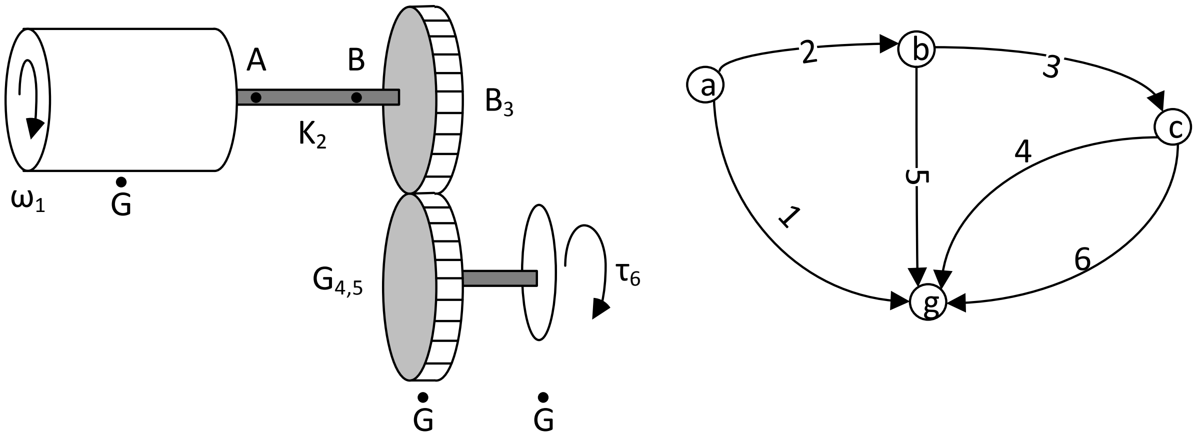

Figure 2.7: Mechanical geargrain and Linear Graph Representation.

## 2.3 Topology Search Methods Applied to Hybrid Electric Powertrain Architecture

Heuristics or selective human judgement can be applied to assess the feasibility of a topology [22]. Heuristics use 'rules of thumb' to constrain the design space. It can be difficult to create heuristics that are general enough to reduce the number of topologies to a practical number but not accidentally eliminate feasible solutions. Heuristics that can be described using mathematics are well suited for computer-based enumerative approaches. Selective processes rely on human judgement to asses a topology, thus are prone to error and cognitive bias. If a powertrain architecture is feasible, it can be further improved by optimizing the size of its components. This section will describe a few methods used by other authors to generate and evaluate HEV powertrain topologies.

### Multiple Planetary Gearsets

As various automotive manufacturers hold patents for specific powertrain architectures, research has been done to identify and evaluate novel concepts. Since planetary gearsets can connect 3 components together, multiple planetary gearsets can results in many different powertrain configurations. Bayrak et al. [22] described a systematic method to select the optimal powertrain configuration containing two planetary gearsets, one engine, and two electric motors. The topology is represented using bond graphs, and the system equations are generated in state space form. An adjacency matrix is used to describe the connections in the bond graph. A heuristic search is applied to find valid topologies, and eigenvalue information is used to eliminate isomorphs. Equivalent Consumption Minimization Strategy (ECMS) is used to determine the optimal control law which minimizes fuel consumption. ECMS assumes the objective score is a function of the fuel consumed, and the energy discharged from the battery [42]. An architecture was found that uses 8% less fuel than a Prius-like configuration.

Liu et al. [43] proposed an exhaustive search to discover all HEV configurations involving two planetary gears, an ICE, and two electric motors. The powertrain is designed for a military vehicle (Humvee), with a curb weight of approximately 5000kg. Once a configuration is structurally feasible, the component sizes are optimized (including gear ratio, but not battery), and its drive performance is evaluated. Dynamic Programming was used to evaluate the fuel consumption for each configuration. 288 kinematically feasible designs were found, but only 2 were able to meet both low speed and high speed driving performance.

Ma et al. [44] used an enumerative approach to discover HEV geartrains comprised of two planetary gear sets, and two clutches. All configurations are evaluated for top speed and torque. It was found that the second planetary gearset can act as a variable reduction gear, thus better accelerations can be achieved at higher velocities.

**Clutches and Gearboxes**

Mechanical clutches can increase the number of degrees of freedom a powertrain by allowing/disallowing the transfer of rotational energy. Clutches can connect and disconnect rotating bodes. If one end of the clutch is connected to ground, it will prevent the other end from rotating when closed. These additional operating modes can potentially improve system efficiency. Zhang et al. [45] used an enumerative approach to compare variations of Prius-like and Volt-like HEV powertrains. The configurations are comprised of: one planetary gear, one ICE, two electric motors, and any number of clutches. The topology design space is enumerated then evaluated manually for realism. Dynamic programming is used to minimize the fuel consumption for comparison of configurations. Simulations ignore: gear efficiency, acceleration performance, and stationary charging operation (ICE charging battery at standstill). Adding a clutch to the Prius configuration (originally no clutches) improved fuel economy for urban driving, but not for highway driving. Removing a clutch from the Volt configuration (originally 3 clutches) resulted in similar fuel consumption for both drive cycles, and therefore can theoretically be removed to simplify the system.

In addition to clutches, the position of various gearboxes can have an effect on the overall system efficiency. Hofman et al. [46] used an enumerative approach to optimize powertrains consisting of: a transmission, electric motor, and ICE. Numerous transmissions were evaluated. Component sizes were also optimized. Topology feasibility was manually assessed, then the equations were assembled manually. Dynamic programming was used to find the optimal control law that minimizes carbon dioxide emissions. It was found that the optimal topologies were: an automatic transmission with electric motor between ICE/transmission; or a continuously-variable transmission with an electric motor between transmission/differential.

## 2.4 Summary

There is a significant body of literature regarding HEV powertrain modelling. Of the reviewed work, Linear Graphs and the Graph Theoretic Method have yet to be applied to HEV topology optimization. This thesis will report the effectiveness of applying the GTM to exploring and optimizing HEV topology.

# Chapter 3

# Hybrid Electric Vehicle Components

This chapter describes the equations which govern the vehicle powertrain components. The models described in this chapter are: vehicle dynamics, final drive reduction gear losses, internal combustion engine, electric motor/generator, powersplit device, and battery. The components are described using steady-state models, thus all the equations in algebraic form. Where possible, data was collected from the Autonomie 2012 software package [5] for use in the ASHev model. The steady-state powertrain model assumes any component can reach any operation point within 1 second, regardless of its original operating condition; however, penalties are applied during Dynamic Programming to prevent unrealistic behaviour (described in section 6.2). The method for scaling component sizes is described at the end of the chapter.

## 3.1 Component Models

### 3.1.1 Longitudinal Dynamics

The vehicle is described only by its longitudinal dynamics [47, 7, 13, 48, 49]. These equations assume the vehicle is: (i) driving forward, (ii) in a straight line, (iii) on a flat plane, (iv) of variable slope. Lateral effects such as turning and crosswinds are ignored. The free body diagram for the vehicle is shown in figure 3.1. The equation for longitudinal dynamics is described in equation 3.1.

$$F_t = m_v \frac{d}{dt} v(t) + F_a + F_r + F_g + F_d \tag{3.1}$$

where $m_v$ is the vehicle mass, $v(t)$ is the vehicle velocity, $F_a$ is aerodynamic friction, $F_r$ is the rolling friction, $F_g$ is the component of weight parallel to the (sloped) road, and $F_d$ is the inertial force of the driveline. $F_t$ is the traction force generated by the powertrain, which is transmitted through the final drive to the wheels.

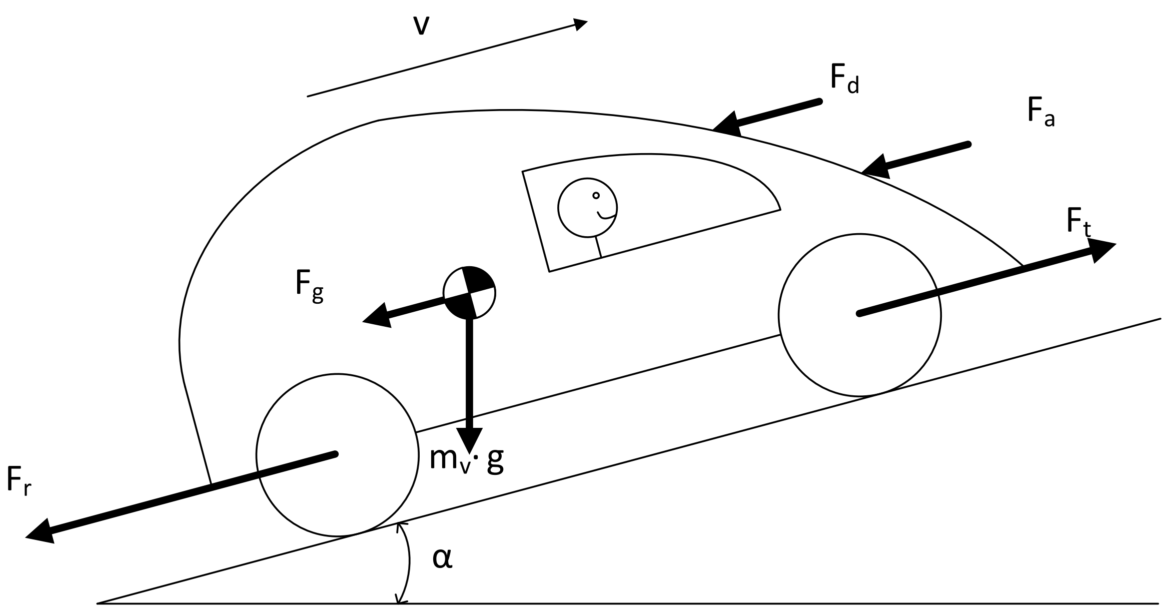

Figure 3.1: Free Body Diagram of Vehicle Linear Dynamics.

Aerodynamic forces are described in equation 3.2.

$$F_a = \frac{1}{2}\rho A_f c_d v^2 \tag{3.2}$$

where $\rho$ is the density of air, $A_f$ is the frontal area of the vehicle, and $c_d$ is the constant coefficient of drag.

Rolling friction is defined as the normal component of weight multiplied by the coefficient of rolling friction, and is described by equation 3.3.

$$F_r = c_r m_v g \cos(\alpha) \tag{3.3}$$

where $c_r$ is the coefficient of rolling friction (assumed to be constant), $g$ is the acceleration due to gravity, and $\alpha$ is the slope of the road.

The force of gravity is described equation 3.4.

$$F_g = m_v g sin(\alpha) \tag{3.4}$$

Where $g$ is the acceleration due to gravity ($9.81 m/s^2$), $m_v$ is the vehicle mass, and $\alpha$ is the road slope.

The inertial driveline force is described by equation 3.5.

$$F_d = \lambda m_v \frac{dv}{dt} \tag{3.5}$$

Where $\lambda$ is the rolling inertia of the driveline.

### 3.1.2 Final Drive

The final drive model is a simple reduction gear assuming a 3% power loss [5]. The equation for torque and speed are shown in equation 3.6.

$$\begin{aligned} \tau_{FD} &= \frac{\tau_{Wheel}}{0.97 i_0} \\ \omega_{FD} &= \omega_{Wheel} i_0 \end{aligned} \tag{3.6}$$

where $\tau_{FD}$ and $\omega_{FD}$ is the final drive torque and angular speed; $\tau_{Wheel}$ and $\omega_{Wheel}$ is the

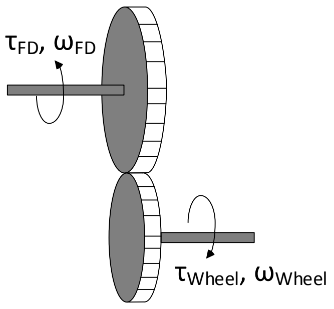

Figure 3.2: Final Drive is Modeled as a Reduction Gear.

final drive torque and angular speed of the wheel; and $i_0$ is the ratio of the reduction gear for the final drive.

### 3.1.3 Internal Combustion Engine

The Internal Combustion Engine (ICE) burns fuel (usually gasoline) in a small chamber containing a piston connected to a crankshaft. The expansion of gas pushes on the piston, causing it to move, and forcing the crankshaft to rotate. Higher engine torques can generally be produced by combusting more fuel. The efficiency of the engine is highly dependent on its operating conditions. The engine efficiency is generally determined experimentally and described using Brake Specific Fuel Consumption (BSFC) map.

The ICE fuel consumption is modelled by interpolating the BSFC map. This experimentally validated the data was taken from Autonomie [5] and is shown in figure 3.3. It is assumed the engine is always hot, so a 'cold map' describing fuel consumption of a cold engine is not required. Realistically, frictional forces cause the engine to naturally slow down when the engine is not supplied with fuel. However, ASHev uses a steady-state model, so these dynamics are not captured. Instead, it is assumed that no fuel is consumed if the engine is not outputting torque. The fuel consumption rate is approximated by a lookup map, and is described by equation 3.3. Attempts to model the ICE fuel consumption rate as a 5-3 polynomial (2 variable polynomial largest powers 3 and 5) were successful, but took longer to solve and was less accurate.

$$\dot{m}_{fuel}[g/s] = \frac{BSFC(\tau_{ICE}, \omega_{ICE}) \left[\frac{g}{kWh}\right]}{3.6 * 10^6 \left[\frac{J}{kWh}\right]} \tau_{ICE}\omega_{ICE} \tag{3.7}$$

The maximum torque constraint is described using a 3rd order polynomial [13, 7], shown in equation 3.8 .

$$\tau_{ICE\,\max} = p_1x^3 + p_2x^2 + p_3x + p_4 \tag{3.8}$$

where $x$ is the engine speed in rads.

### 3.1.4 Electric Motor / Generator

3-3 polynomials are usually sufficient to describe the efficiency of an electric motor generator (EM/GEN)[50]. However, it was found that a 4-4 polynomial had a better fit to the efficiency maps provided by Autonomie [5]. The electric motor/generator efficiency ($\eta_{EM}$) is modelled as a 4-4 polynomial described in equation 3.9.

$$\begin{aligned}\eta_{EM} = {} & p_{00} + p_{10}\tau + p_{01}\omega + p_{20}\tau^2 + p_{11}\tau\omega + p_{02}\omega^2 + p_{30}\tau^3 + p_{21}\tau^2\omega \\ & + p_{12}\tau\omega^2 + p_{03}\omega^3 + p_{40}\tau^4 + p_{31}\tau^3\omega + p_{22}\tau^2\omega^2 + p_{13}\tau\omega^3 + p_{04}\omega^4\end{aligned} \tag{3.9}$$

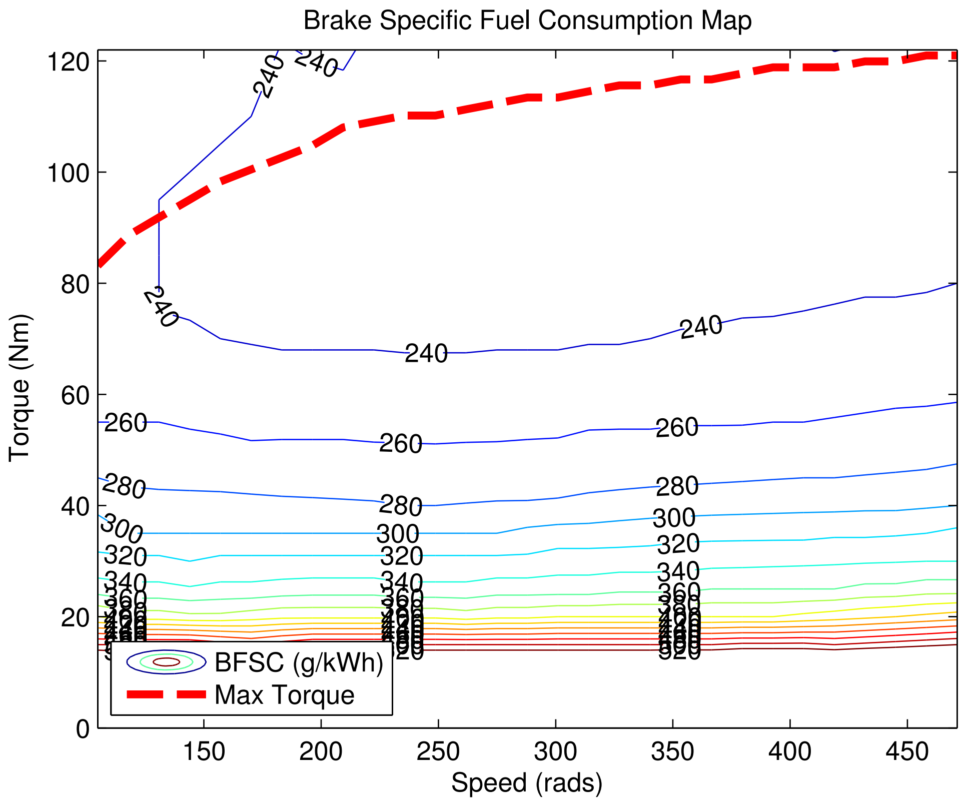

Figure 3.3: Brake Specific Fuel Consumption Hot Map for 57kW Internal Combustion engine. Data source from Autonomie 2012 [5].

where $\tau$ and $\omega$ is the torque and speed of the EM respectively. The polynomial fit is acceptable ($R^2 = 0.9995$), but will accumulate small errors over time. The EM can operate as a motor and/or generator. Operating as a generator is sometimes less efficient than operating as a motor, and thus each operation mode uses a different efficiency map. If the product of the torque and speed is positive (first and third quadrant), it is operating as a motor. If the product is negative (second and fourth quadrant), then it is operating as a generator. This behaviour is shown in figure 3.4. In equation 3.10, a piecewise equation describes the behaviour of the EM while: not operating, operating as a motor, or operating as a generator.

$$P_{EM}(\tau,\omega) = \begin{cases} \tau\omega = 0, 0 \\ \tau\omega > 0, \eta_{EM+}(\tau,\omega)\tau\omega \\ \tau\omega < 0, \eta_{EM-}(\tau,\omega)\tau\omega \end{cases} \tag{3.10}$$

where $\eta_{EM+}$ and $\eta_{EM-}$ is the efficiency of the EM during motor (+) or generator (-) operation respectively.

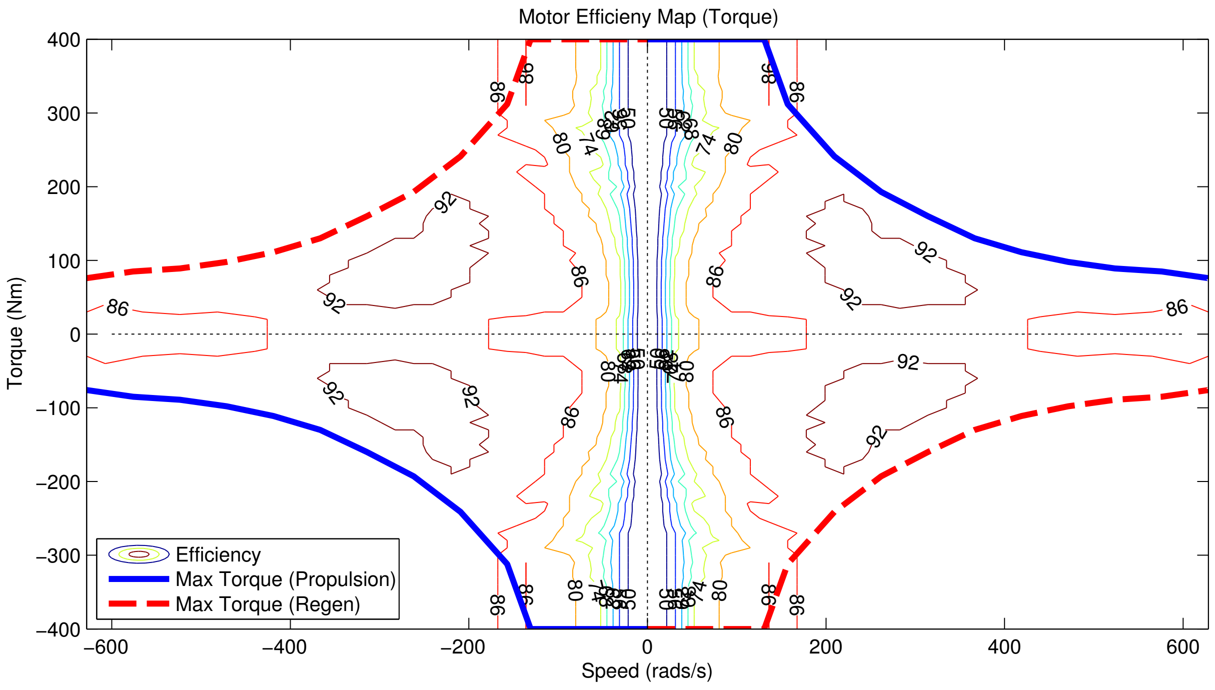

Figure 3.4: Electric Motor Efficiency Map. Data source from Autonomie 2012 [5].

Using GTM, 'Power' is not a through or across variable. Therefore an additional equation is used to relate the electrical power to the current measured at the EM terminals.

$$I_{EMG} = \frac{P_{EMG}}{V_{EMG}} \tag{3.11}$$

### 3.1.5 Discrete Gearbox

The 5-speed manual gearbox is comprised of a train of spur gears [51]. Such gears are commonly applied in automotive applications because they are 98-99% efficient [52]. The schematic shown in figure 3.5, the gearbox ratio can be selected by moving the gear selector fork, which will engage the gear collars. A rotational speed at Flange A $\omega_{GB_a}$ will be multiplied or reduced by the selected gear ratio $R_{GB}$ resulting in speed $\omega_{GB_b}$ seen at

Flange B. Conversely (assuming no losses), the torque seen at Flange B will be equal to the negative product of inverse of $R_{GB}$ and the torque seen at Flange A.

The modelled gearbox has 5 gear ratios, ranging from less than 1 to greater than 3; the gearbox can act as either a speed reducer or a speed multiplier. As the system equations of the vehicle will be generated acasually, and the gearbox will not know which flange is the input/output, the 5-speed gearbox is modelled as an ideal (lossless) gear. The equation for the gearbox is shown in equation 3.12 [5, 26].

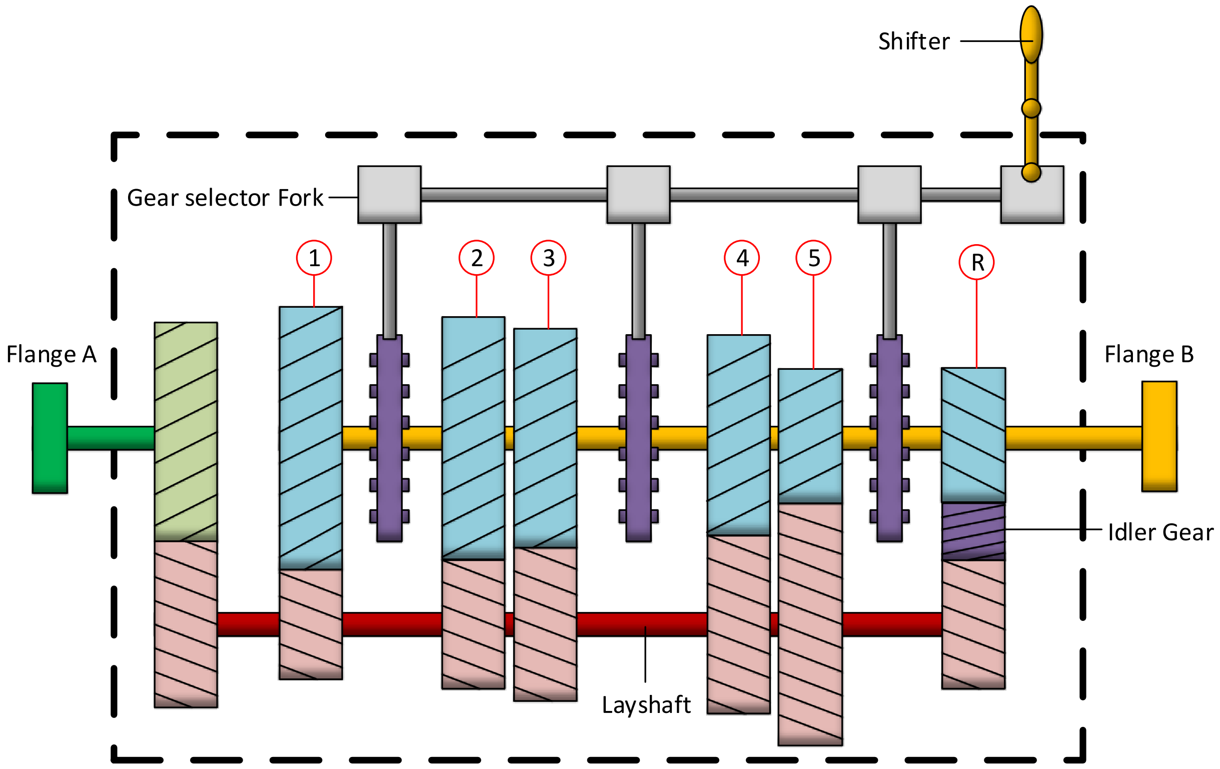

Figure 3.5: Diagram of a 5 Speed Sequential Gearbox with Reverse.

$$\begin{aligned} \tau_{GBb} &= \frac{-1}{R_{GB}(i_g)}\tau_{GBa} \\ \omega_{GBb} &= R_{GB}(ig)\omega_{GBa} \end{aligned} \tag{3.12}$$

where $GBa$ and $GBb$ are rotational flanges on the gearbox, $R_{GB}$ is the gear ratio equal to {3.32,2,1.36,1.01,0.82} (the default gearbox in Autonomie [5]), and $i_g$ is the gear number.

### 3.1.6 Powersplit Device

The Powersplit Device (PSD) (also known as the planetary gearset, or epicyclic gear train) can be found in automatic gearboxes, automotive differentials, and aircraft propeller reductions. The PSD allows 3 rotational components to be connected together. Shown in figure 3.6, the Powersplit Device is comprised of a ring gear, a sun gear, a planetary carrier gear, and the planet gears. There are always at least 3 planets on the planetary carrier to ensure rotational balance and reduce stress on the planet gears. As the sun gear turns, it will force the planet gears to turn the opposite direction, which turns the ring gear. The planet carrier can also turn, which causes the planet gears to rotate around the sun gear allowing for different speed outputs (including negative rotations) at the ring gear [52, 51]. The Powersplit device is described using 6 variables and 3 algebraic equations,

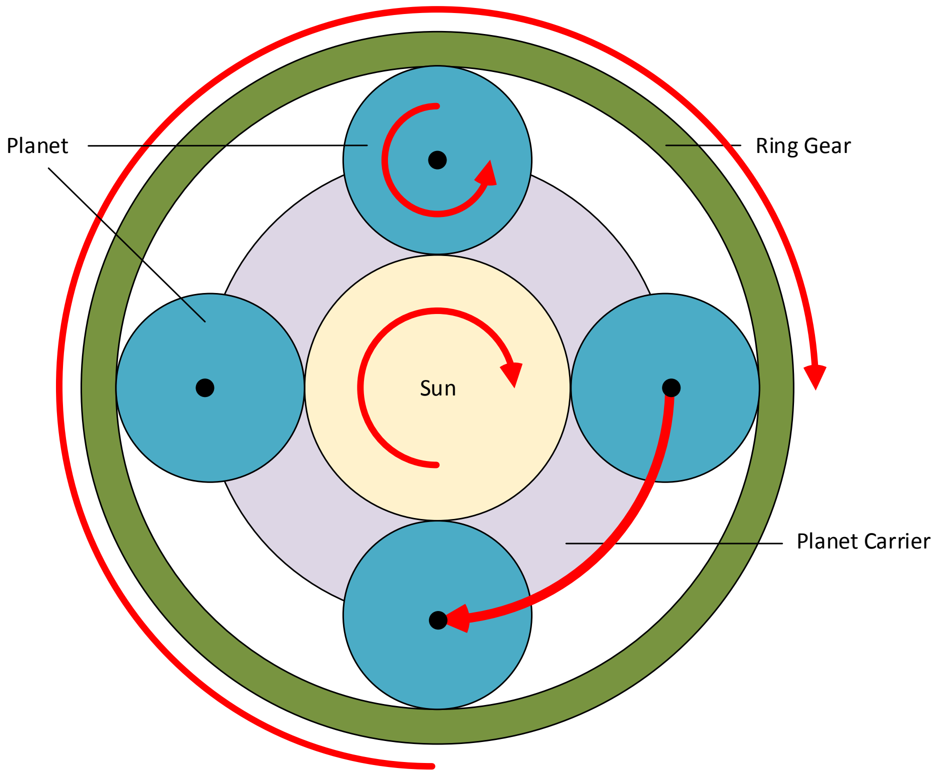

Figure 3.6: Diagram of a Planetary Gearset.

thus allowing 3 degrees-of-freedom (DOF). During operation, if one of the gears is locked in place (not allowed to move) the system can be treated as a static gear ratio. In the case of a Powersplit Hybrid, the speed of one of the rotational flanges is specified as a function of vehicle (wheel) speed, so this system is reduced to a 2 DOF system.

The PSD is modelled as a lossless device, shown in equation 3.13.

$$\begin{aligned} \tau_{PSD_R} - R_{PSD}\tau_{PSD_S} &= 0 \\ \tau_{PSD_{PC}} + \tau_{PSD_S} + \tau_{PSD_R} &= 0 \\ \omega_{PSD_{PC}}(1 + R_{PSD}) - \omega_{PSD_S} - R_{PSD}\omega_{PSD_R} &= 0 \end{aligned} \tag{3.13}$$

where $R$, $S$, $PC$ represent the ring, sun, and planetary carrier gear, respectively; $R_{PSD}$ is the ratio of teeth on the sun gear to the ring gear, where $R_{PSD} < 1$.

### 3.1.7 Battery

The battery is generally considered one of the most difficult powertrain components to model due to its electrochemical nature. Battery temperature, age, and State-Of-Charge (SOC describes how much energy is left in the battery) have non-linear effects on the battery terminal voltage [53, 54, 55, 56]. Physics or electrochemical based battery models solve slowly but reflect the physics of the battery [57, 58, 59]. There is a growing body of work to simplify the physics and electrochemical based battery models so they can be used for real-time applications [60, 61].

Shown in figure 3.7, the equivalent circuit model is considered one of the simplest battery models [5, 55]. Open circuit voltage ($V_{oc}$), maximum input and output power, and internal resistance ($R_{int}$) are functions of the battery SOC. Battery internal resistance is used to used to calculate the battery terminal voltage $V$. As the terminal voltage is only a function of the State-Of-Charge, voltage drops are ignored during high output transients. Realistically, a drop in voltage will require more current must be supplied to maintain the same power output. Although this effect is lost in the equivalent circuit model, it is still useful for most purposes.

The battery model equations are as followed in equation 3.14.

$$\begin{aligned} &V_{oc}, R_{\text{int}} = f(SOC) \\ &V = \frac{V_{oc} + \sqrt{{V_{oc}}^2 - 4R_{\text{int}}\eta_{inv}P_{batt}}}{2} \\ &SOC_{k+1} = SOC_k - \frac{P}{3600VQ_{battcap}} \end{aligned} \tag{3.14}$$

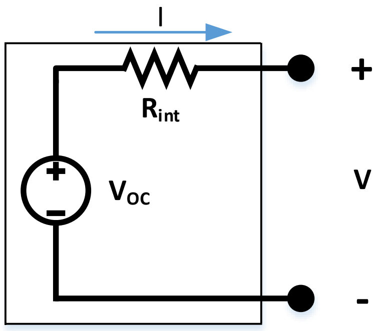

Figure 3.7: Equivalent Circuit Battery Model.

where $\eta_{inv}$ is the inverter efficiency (assumed to be a constant of 0.95), $P_{batt}$ is the battery power, $SOC$ is the State-of-Charge of the battery, and $Q_{battcap}$ is the maximum battery capacity.

As the system is to be simulated in steady-state and the battery voltage is a function of SOC, it is necessary to first assume a voltage to calculate battery power. The battery voltage is assumed to be nominal for 168 cells in series which is 271V. The equation for nominal battery voltage can be found in equation 3.15. When Dynamic Programming is performed, the battery current and power are checked against constraints to ensure the model is representative of the physical system.

$$V_{BAT} = 271[V] \tag{3.15}$$

## 3.2 Component Masses and Scaling

A common way to represent a different sized component is to linearly scale the model. Most of the components described in this chapter can be scaled for performance and mass. For example, the ICE could be scaled up by 1.5 times by multiplying the BFSC map by 1.5, the maximum torque constraint by 1.5, and the ICE mass by 1.5. Shown in table 3.1, the baseline masses and scaling factors can be found for the components. The $S$ symbol denotes how the scaling factor would be applied to the original data. Note that the planetary gear was not scaled, and the discrete gearbox can scale the gear ratios while maintaining a constant mass.

Table 3.1: Baseline Mass and Scaling Equations for HEV Components.

| Component | Baseline mass (kg) | Baseline | Scaling Equations |
|---|---|---|---|
| Internal Combustion Engine (ICE) | 34.00 * S | Prius MY04<br>57 kW<br>1.5L 4 Cylinder | $BSFC = BSFC * S$<br>$\tau_{ICE_{max}} = \tau_{ICE_{max,base}} * S$ |
| Electric Motor (Large) (EM) | 86.76 * S | Permanent Magnet<br>50 kW peak<br>25 kW continuous | $P_{EM} = P_{EM} * S$<br>$\tau_{EM_{max}} = \tau_{EM_{max,base}} * S$ |
| Electric Motor (Small) (GEN) | 25.00 * S | Permanent Magnet<br>30 kW peak<br>14 kW continuous | $P_{GEN} = P_{EM} * S$<br>$\tau_{GEN_{max}} = \tau_{GEN_{max,base}} * S$ |
| Discrete Gearbox (GB) | 75.00 | [3.32,2,1.36,1.01,0.82] | $R_{GB} = R_{GB,base} * S$ |
| Battery (BAT) | 35.70 * S | NiMH<br>6.5 Ah 51xSeries @ 273V | $BAT_{Cap} = BAT_{Cap,base} * S$ |
| Planetary Gear (PSD) | 40.00 | Ring/Sun teeth ratio = 78/30 | —- |

The ICE, EM and GEN all start using baseline performance maps for a 2004 Prius. The first equation linearly scales the power requirement, i.e. $BSFC$ (gasoline) for the ICE, and $P_{EM}$ and $P_{GEN}$ (electrical power) for the electric motor and generator. For the electric motor, the appropriate maps are scaled before curve-fit to polynomials. The second equation describes their maximum output constraint (torque).

As the number of battery cells is assumed to be constant, the capacity of these cells will scale with the scaling ratio, but the nominal voltage is kept constant.

## 3.3 Summary

The equations for Hybrid Electric Vehicle powertrain components are described in this chapter. The components can be linearly scaled to represent different size components. When interconnection equations are introduced, these equations can be used to describe many different powertrain configurations. The next chapter will explain how the configuration are represented and the vehicle powertrain structure is generated.

# Chapter 4

# Hybrid Electric Vehicle System Representation and Modelling

This chapter describes the Graph-Theoretic Method (GTM) in detail. The equations for a Powersplit hybrid powertrain are generated using the GTM as an example. An enumerative approach to finding the control varaibles is shown, and heuristics for finding feasible powertrains is presented.

## 4.1 Graph-Theoretic Method

The Graph-Theoretic Method is a systematic method for generating system equations by specifying a topology and assigning the appropriate constitutive (terminal) equations [40, 10, 12, 11]. This method is very systematic and thus is suitable for computer implementation. The topology of a system can be represented using a linear graph, which is formed from a collection of nodes and directed edges. Before describing in detail the GTM process, it is necessary to define GTM terminology:

**Graph Theoretic Method Terminology [11]**

**Graph** A graph is a collection of edges which intersect upon nodes. A graph is a pictorial representation of the topology for a given system.

**Edge** Edges connect nodes and are pictorially described in a graph using an arrow. Each edge is directed; it has a beginning point and an end point. If a node is the end point for an edge, it is said that the edge is incident upon that node.

**Node** Nodes (also called vertices) are represented in a graph using dots or circles. In a physical system, nodes represent component terminals.

**Through Variable** Through measurements are made in *series* with the component and two selected terminals. The type of variable is dependent on the physical domain. For example, in an electrical system the through variable is current. In a rotational mechanical system, the through variable is torque.

**Across Variable** Complementary to the through measurement, across measurements are made in *parallel* with the component and two selected terminals. Every edge has a corresponding across variable dependent on the domain. For example, for an electrical system the across variable is voltage; for a rotational mechanical system the across variable is angular velocity.

**Path** If a node can be reached by traversing through the edges to another node, it is said that these nodes are connected by a *path*. For example, in figure 4.1 nodes $A$ and $G_M$ are connected, and nodes $A$ and $C$ are also connected.

**Connected** If every node can be reached by every other node, it is said that this graph is *connected*. For example, the graph in figure 4.1 is not connected because there is no path from node $(A/B/G)$ to node D.

**Circuit** A circuit is a subgraph that has exactly two distinct paths between every pair of nodes in the subgraph.

**Tree** A tree is defined as a subgraph that: is connected; contains all the nodes in the graph; and has no circuits. Edges in a tree are called *branches*.

**Cotree** A cotree is a subgraph of G that remains after deleting the edges of a tree. Edges in a tree are called *chords*.

**Fundamental Circuit** A fundamental circuit consists of one chord, and a unique set of branches. There is only one fundamental circuit for each chord in a graph.

**Fundamental Cutset** A cutset is a subset of edges which divides the graph into two parts. The fundamental cutset contains one branch and a unique set of chords. There is only one fundamental cutset per branch in a linear graph.

### 4.1.1 Incidence Matrix Representation

The topology of a Hybrid Electric Vehicle can be represented by a directed linear graph. This linear graph can be described mathematically as an incidence matrix. The **columns** in the incidence matrix represent edges corresponding to vehicle components. The **rows** in the incidence matrix represent nodes to which the components connect. For example, the mechanical incidence matrix shown in equation 4.1 corresponds to the mechanical linear graph in figure 4.1 (left). Similarly, the electrical incidence matrix shown in equation 4.12 corresponds to the electrical linear graph in figure 4.1 (right).

Some components only appear in the mechanical graph (i.e. final drive and ICE), some in the electrical graph (i.e. battery), and some in both (i.e. transducers, such as electric motor or generator). The components that appear in both the mechanical and electrical graph have additional terminal equations which relate their mechanical and electrical behaviour.

The incidence matrix $IM$ is a matrix composed of zeros and ones and negative ones, where the ones represent the edge 'entering' the node, and a negative one represents the edge 'exiting' the node. Each edge of the linear graph has a through variable: a physical quantity measured in series. Each node of the linear graph represents a place where an across variable could be measured; these are places where a variable measured in parallel would be measured. Zeros represent no connection between the edge (column) and node (row).

$$[IM_{MECH}] = \begin{array}{c} \\ \textit{node} \\ A \\ B \\ C \\ D \\ G_M \end{array} \begin{array}{c} \overbrace{\begin{array}{ccccccc} \quad ICE & FD & EM & GEN & PSD_R & PSD_S & PSD_{PC} \end{array}}^{\text{components}} \\ \left[\begin{array}{ccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{array}\right] \end{array} \tag{4.1}$$

$$[IM_{ELEC}] = \begin{array}{c} \\ \textit{node} \\ E \\ F \\ H \\ G_E \end{array} \begin{array}{c} \overbrace{\begin{array}{ccc} \quad EM & GEN & BAT \end{array}}^{\text{components}} \\ \left[\begin{array}{ccc} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{array}\right] \end{array} \tag{4.2}$$

The incidence matrix contains all of the topological information of the system. To generate the system of equations from this matrix, a vector of *through variables* and *across variables*

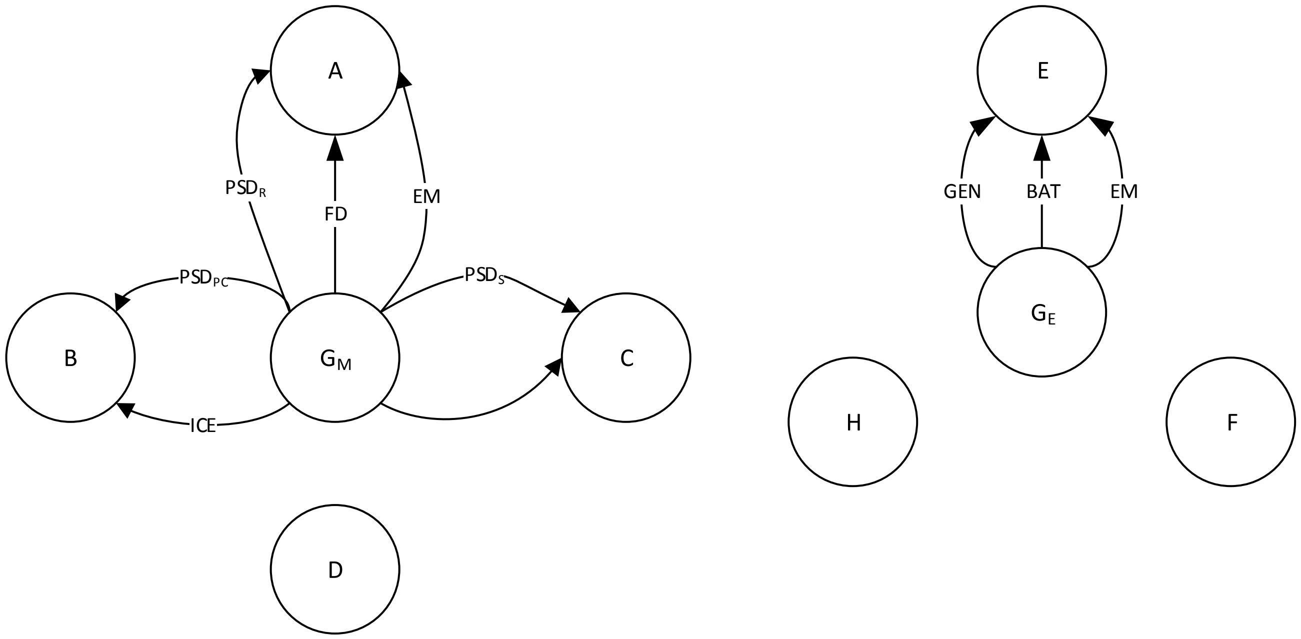

Figure 4.1: Linear Graph Representing a Series Hybrid. The Left Graph is the Mechanical Graph and the Right Graph is the Electrical Graph.

are required.

**Through variables** represent a quantity that would be measured in series with the corresponding edge (component). For example, an ammeter is put in series with an electrical resistor to measure current in an electrical system; a torque meter is placed on a rotational shaft to measure torque in a rotating mechanical system.

**Across variables** are quantities measured in parallel with the edge (component). For example: in an electrical system a voltmeter may be placed in parallel with a resistor to determine the voltage drop across the resistor; in a rotational mechanical system a rotational sensor may be used to measure the angular velocity of a shaft with respect to a non-rotating point.

Across variables are often most useful when measured with respect to a fixed point, or ground, which corresponds to the **datum** or **ground node**. In this work, all of the measurements are taken with respect to the ground node, because we are interested in the absolute values of the measurement, and not the measurements relative to the nodes. For example, it is useful to know the speed and torque of the engine relative to the observer (ground), rather than measuring it with respect to a moving part of the powertrain.

For this example, the vector of mechanical and electrical through variables, representing torques ($\tau$) and electrical current ($I$) respectively, can be found in equation 4.3. Similarly, the vector of mechanical and electrical across variables, representing angular velocity ($\omega$) and voltage ($V$) can be found in equation 4.18.

$$\begin{aligned} \{\tau_{MECH}\} &= \{\tau_{ICE}, \tau_{FD}, \tau_{EM}, \tau_{GEN}\, \tau_{PSD_R}, \tau_{PSD_S}, \tau_{PSD_{PC}}\}^T \\ \{\tau_{ELEC}\} &= \{I_{EM}, I_{GEN}, I_{BAT}\}^T \end{aligned} \tag{4.3}$$

$$\begin{aligned} \{\alpha_{MECH}\} &= \{\omega_{ICE}, \omega_{FD}, \omega_{EM}, \omega_{GEN}\, \omega_{PSD_R}, \omega_{PSD_S}, \omega_{PSD_{PC}}\}^T \\ \{\alpha_{ELEC}\} &= \{V_{EM}, V_{GEN}, V_{BAT}\}^T \end{aligned} \tag{4.4}$$

### 4.1.2 Equation Generation

In this section, the equations for a Powersplit (Prius-like) architecture are derived. The system level architecture is shown in figure 5.1.

In GTM, the vertex postulate states that: *"the sum of through variables at any node of a linear graph must equal zero when due account is taken of the orientation of edges incident upon that node"* [11]. This is equivalent to Kirchoff's Current Law [41] and can be expressed mathematically by equation 4.5.

$$[IM]\{\tau\} = \{0\} \tag{4.5}$$

where $\{\tau\}$ is a vector of through measurement variables. In this example, the vector of through measurement variables for a Powersplit Hybrid shown in equation 4.3.

#### Cutset Equations

First, to ensure a connected graph, the nodes unconnected to the graph (rows of zeros) are deleted. Then, the reduced incidence matrix is obtained by deleting the row corresponding to the datum node. Starting with equation 4.1, node $G_M$ is removed from the IM to obtain the reduced incidence matrix A shown in equation 4.6.

$$[A] = \begin{array}{c} \scriptstyle node \\ \scriptstyle A \\ \scriptstyle B \\ \scriptstyle C \end{array} \begin{array}{ccccccc} \scriptstyle ICE & \scriptstyle FD & \scriptstyle EM & \scriptstyle GEN & \scriptstyle PSD_R & \scriptstyle PSD_S & \scriptstyle PSD_{PC} \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array} \tag{4.6}$$

Column swapping, and Gauss Jordan elimination is performed on reduced incidence matrix $[A]$ to transform it to the fundamental-cutset matrix $[A_f]$, where $[A_f]$ is in the form:

$$[A_f] = [[1_b][A_C]] \tag{4.7}$$

where, $1_b$ is an identity matrix with as many rows and columns as nodes in the graph (minus the datum node), and $A_C$ is the remainder of the $A_f$ matrix. Please note that the leftmost columns of $A_f$ or the $1b$ matrix correspond to the branches in the tree, defining the fundamental cutset [11].

$$[A_{f,MECH}] = \begin{array}{c} \scriptstyle node \\ \scriptstyle A \\ \scriptstyle B \\ \scriptstyle C \end{array} \begin{array}{ccccccc} \scriptstyle FD & \scriptstyle ICE & \scriptstyle GEN & \scriptstyle EM & \scriptstyle PSD_R & \scriptstyle PSD_S & \scriptstyle PSD_{PC} \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \tag{4.8}$$

Note: if column swapping is performed, the vector of through variables $\tau$ is correspondingly rearranged.

$$\{\tau_{f,MECH}\} = \{\tau_{FD}, \tau_{ICE}, \tau_{EM}, \tau_{GEN}, \tau_{PSD_R}, \tau_{PSD_S}, \tau_{PSD_{PC}}\}^T \tag{4.9}$$

Using equation 4.10, the system of cutset equations can be obtained by multiplying the fundamental cutset matrix $[A_f]$ by the vector of through variables $\{\tau_{f,MECH}\}$. The cutset equations for the Powersplit mechanical graph can be found in equation 4.11.

$$[A_f]\{\tau_{f,MECH}\} = \{0\} \tag{4.10}$$

$$\begin{aligned} \tau_{FD} + \tau_{EM} + \tau_{PSD_R} &= 0 \\ \tau_{ICE} + \tau_{GEN} &= 0 \\ \tau_{PSD_S} + \tau_{PSD_{PC}} &= 0 \end{aligned} \tag{4.11}$$

Using a similar approach to the electrical graph, the $F$, $H$, and $G_E$ rows are deleted. After, column swapping is performed, and Gauss Jordan elimination is performed to obtain the reduced incidence matrix (resulting in the same matrix). The fundamental electrical cutset matrix for the Powersplit configuration is shown in equation 4.12.

$$[A_{f,ELEC}] = \begin{array}{c} \scriptstyle node \\ \scriptstyle E \end{array} \begin{array}{ccc} \scriptstyle EM & \scriptstyle GEN & \scriptstyle BAT \\ \left[ 1 \right. & 1 & \left. 1 \right] \end{array} \tag{4.12}$$

The reduced incidence matrix is multiplied by the vector of through variables, resulting in the system of through equations for the electrical graph, found in equation 4.13.

$$I_{EM} + I_{GEN} + I_{BAT} = 0 \tag{4.13}$$

## Circuit / Across Equations

Using the circuit postulate in GTM, the across equations can be generated. The circuit postulate states: *"the sum of across variables around any circuit of a graph must equal zero when due account is taken of the direction of edges in the circuit"*. Mathematically this can be described by equation 4.14.

$$[B_f][\alpha] = \{0\} \tag{4.14}$$

where,

$$[B_f] = [[B_b][1c]] \tag{4.15}$$

Luckily, it is not required to revisit the linear graph to obtain $[B_b]$. Exploiting the principle of orthogonality [11, 62], $[B_b]$ can be obtained from $[A_c]$ using equation 4.16.

$$[B_b] = -[A_c]^T \tag{4.16}$$

Therefore in this example, $[B_f]$ can be found in equation 4.17.

$$[B_{f,MECH}]^T = \left[\begin{array}{ccccccc} \scriptstyle FD & \scriptstyle ICE & \scriptstyle GEN & \scriptstyle EM & \scriptstyle PSD_R & \scriptstyle PSD_S & \scriptstyle PSD_{PC} \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{4.17}$$

Similar to the vector of equations found in equation 4.3, the across variables for the Powersplit hybrid powertrain are shown in equation 4.18.

$$\begin{aligned} \{\alpha_{MECH}\} &= \{\omega_{ICE}, \omega_{FD}, \omega_{EM}, \omega_{GEN}\, \omega_{PSD_R}, \omega_{PSD_S}, \omega_{PSD_{PC}}\}^T \\ \{\alpha_{ELEC}\} &= \{V_{EM}, V_{GEN}, V_{BAT}\}^T \end{aligned} \tag{4.18}$$

By applying equation 4.14 to 4.17 and 4.18, we can obtain the system of across equations for the mechanical graph, which are shown in equation 4.19.

$$\begin{aligned} -\omega_{FD} + \omega_{EM} &= 0 \\ -\omega_{FD} + \omega_{PSD_R} &= 0 \\ -\omega_{GEN} + \omega_{PSD_S} &= 0 \\ -\omega_{ICE} + \omega_{PSD_{PC}} &= 0 \end{aligned} \tag{4.19}$$

Similarly the connection equations for the electrical graph are derived by permuting the $[B_b]$ matrix, shown in equation 4.20.

$$[B_{f,ELEC}]^T = \begin{array}{c} \begin{array}{ccc} EM & GEN & BAT \end{array} \\ \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \end{array} \tag{4.20}$$

Multiplying the matrix in equation 4.20 by the vector of across variables found in equation 4.18 results in the system of circuit equations for the electrical graph, which is shown in equation 4.21.

$$\begin{aligned} -V_{EM} + V_{GEN} &= 0 \\ -V_{GEN} + V_{BAT} &= 0 \end{aligned} \tag{4.21}$$

### 4.1.3 Terminal Equations

The interconnection equations are obtained by combining the mechanical cutset equations (4.11), electrical cutset equations (4.19), mechanical circuit equations (4.19) and electrical circuit equations (4.21). By examining the interconnection equations in terms of their variables, the respective constitutive equations can be systematically chosen from a library of equations (library of component models). The system equations are obtained by combining the interconnection equations with the constitutive equations.

For the Powersplit powertrain example, the variables for the engine $\tau_{ICE}$ and $\omega_{ICE}$ are in the equations, so the equation 3.7 describing fuel consumption is added to the system. Similarly, the equations for the EM (equation 3.9), GEN (equation 3.9) and PSD (equation 3.13) are included. As a mechanical brake is required in the final system, the $\tau_{FD}$ variable is replaced with $\tau_{FD} + \tau_{brake}$. A more rigorous approach would be to add the mechanical brake to the graph. This approach was taken for its simplicity, it is a logical place to put the brake, and to limit the design space. The system comprised of the interconnection and terminal equations is found below in equation 4.22.

$$
\begin{aligned}
\tau_{FD} + \tau_{brake} + \tau_{EM} + \tau_{PSD_R} &= 0 \\
\tau_{ICE} + \tau_{PSD_{PC}} &= 0 \\
\tau_{GEN} + \tau_{PSD_S} &= 0 \\
I_{EM} + I_{GEN} + I_{BAT} &= 0 \\
-\omega_{FD} + \omega_{EM} &= 0 \\
-\omega_{FD} + \omega_{PSD_R} &= 0 \\
-\omega_{GEN} + \omega_{PSD_S} &= 0 \\
-\omega_{ICE} + \omega_{PSD_{PC}} &= 0 \\
-V_{GEN} + V_{BAT} &= 0 \\
-V_{EM} + V_{GEN} &= 0 \\
V_{BAT} - 271 &= 0 \\
I_{EM} - \frac{P_{EM}}{V_{EM}} &= 0 \\
P_{EM}(\tau,\omega) = \begin{cases} \tau\omega = 0, 0 \\ \tau\omega > 0, \eta_{EM+}(\tau,\omega)\tau\omega \\ \tau\omega < 0, \eta_{EM-}(\tau,\omega)\tau\omega \end{cases} \\
I_{GEN} - \frac{P_{GEN}}{V_{GEN}} &= 0 \\
P_{GEN}(\tau,\omega) = \begin{cases} \tau\omega = 0, 0 \\ \tau\omega > 0, \eta_{GEN+}(\tau,\omega)\tau\omega \\ \tau\omega < 0, \eta_{GEN-}(\tau,\omega)\tau\omega \end{cases} \\
\dot{m_{fuel}} - \frac{BSFC(\tau_{ICE},\omega_{ICE})}{3.6 * 10^6}\tau_{ICE}\omega_{ICE} &= 0 \\
\tau_{PSD_R} - R_{PSD}\tau_{PSD_S} &= 0 \\
\tau_{PSD_{PC}} + \tau_{PSD_S} + \tau_{PSD_R} &= 0 \\
\omega_{PSD_{PC}}(1 + R_{PSD}) - \omega_{PSD_S} - R_{PSD}\omega_{PSD_R} &= 0
\end{aligned}
\tag{4.22}
$$

### 4.1.4 Driver Equation Selection

Examining equation 4.22, it can be seen that there are 19 equations and 24 variables. As there must be an equal number of equations and variables to solve a system, 24 −

$19 = 5$ variables need to have specified values. By specifying values to certain variables, those variables become the inputs to the system. This section will describe how the input variables are chosen.

Three variables are always specified: final drive torque, final drive speed, and brake torque. A drivecycle is used to evaluate the vehicle architectures. The drivecycle is a predefined driving pattern that specifies speed and acceleration. From the speed and acceleration, the values of the two final drive variables, $\tau_{FD}$ and $\omega_{FD}$ in equation 3.2 can be calculated. In every architecture, a mechanical brake is included, and therefore a mechanical brake torque $\tau_{brake}$ may be specified during deceleration. To reduce the number of unknowns, $\tau_{brake}$ is assigned a percentage of the braking torque (i.e., {0%, 50%, 100 %}). By specifying the values of $\tau_{FD}$, $\omega_{FD}$, and $\tau_{brake}$, 3 of the 5 variables have been specified values, and 2 more are required. An enumeration method is used to systematically pick the remaining 2 inputs.

## Systematic Method for Selecting Input Variables

1. An ordered list of predetermined control variables is loaded from memory. These variables are chosen in this order because they make logical sense as input variables. For example, $R_{GB}$ is a logical choice of a input for any vehicle with a discrete gearbox. $\tau_{ICE}$ is a logical choice for any vehicle with an ICE. Fuel consumption $\dot{m}_{fuel}$ is a poor choice for a control variable as it is function of both $\tau_{FD}$ and $\omega_{FD}$. In this example, the list of logical control drivers is:

$$Driverlist = [R_{GB}, \tau_{ICE}, \omega_{ICE}, \tau_{EM}, \omega_{EM}, \tau_{GEN}, \omega_{GEN}] \tag{4.23}$$

   If the equations are solved symbolically, the control variables will be independent of each other. Torque and speed linked variables, such as $\tau_{ICE}$ and $\omega_{ICE}$, can be controlled independently due to degrees of freedom in the system. For example, if an ICE is connected to an EM/GEN, the EM/GEN can control how much torque is absorbed to create electricity, and subsequently control the speed of the ICE.

2. A list of system variables is extracted from the system equations. The intersect of the list of system variables and the driver list is taken to determine the possible control variables. In this example, the intersect of the variables in equation 4.22 and equation 4.23 is:

$$\{\tau_{ICE}, \omega_{ICE}, \tau_{EM}, \omega_{EM}, \tau_{GEN}, \omega_{GEN}\} \tag{4.24}$$

3. As we only require 2 control variables, we can use combinatorics to determine all combinations of two variables. To save time, this list of permutations is sorted preferentially. In this example the list of permutations is:

$$\begin{aligned} Driverlist = [&\{\tau_{ICE}, \omega_{ICE}\}, \{\tau_{EM}, \tau_{ICE}\}, \{\omega_{EM}, \tau_{ICE}\}, \{\tau_{EM}, \omega_{ICE}\}, \\ &\{\tau_{GEN}, \tau_{ICE}\}, \{\omega_{GEN}, \tau_{ICE}\}, \{\omega_{EM}, \omega_{ICE}\}, \{\tau_{GEN}, \omega_{ICE}\}, \{\tau_{EM}, \omega_{EM}\}, \\ &\{\omega_{GEN}, \omega_{ICE}\}, \{\tau_{EM}, \tau_{GEN}\}, \{\tau_{EM}, \omega_{GEN}\}, \{\omega_{EM}, \tau_{GEN}\}, \{\omega_{EM}, \omega_{GEN}\}, \{\tau_{GEN}, \omega_{GEN}\}] \end{aligned} \tag{4.25}$$

4. An attempt is made to find a symbolic solution to the powertrain model. If not found, the next set of driver variables are selected, and a new attempt is made. As there are more variables (24) than equations (19), the Maple function *eliminate()* is well suited to find a symbolic solution. *eliminate()* eliminates a set of variables from a system of equations by performing substitutions. The result is system of functions, which have the eliminated variables on the left hand side of the equation, and the remaining variables on the right hand side. In this example, all the non-driver equations are eliminated from the system of equations in 4.22 (i.e. $\tau_{EM}, \tau_{GEN}, \tau_{PSD_R}...$) and can be found on the left hand side in the system of equations 4.27. The non-driver variables are all described in terms of the driver variables shown in equation 4.26.

$$\{\tau_{FD}, \omega_{FD}, \tau_{Brake}, \tau_{ICE}, \omega_{ICE}\} \tag{4.26}$$

The functions $f1, f2, f3$ are too large to show as they are comprised of piece-wise functions containing polynomials; their arguments are shown instead.

$$\begin{aligned}
\tau_{EM} &= -\tau_{FD} - \tau_{Brake} - 0.722\tau_{ICE} \\
\tau_{GEN} &= -0.277\tau_{ICE} \\
\tau_{PSD_R} &= 0.722\tau_{ICE} \\
\tau_{PSD_S} &= 0.277\tau_{ICE} \\
\tau_{PSD_{PC}} &= -\tau_{ICE} \\
\omega_{EM} &= \omega_{FD} \\
\omega_{GEN} &= 3.599\omega_{ICE} - 2.599\omega_{FD} \\
\omega_{PSD_{PC}} &= \omega_{ICE} \\
\omega_{PSD_R} &= \omega_{FD} \\
\omega_{PSD_S} &= 3.599\omega_{ICE} - 2.599\omega_{FD} \\
V_{EM} &= 271 \\
V_{GEN} &= 271 \\
V_{BAT} &= 271 \\
I_{EM} &= f_1(\tau_{FD}, \omega_{FD}, \tau_{Brake}, \tau_{ICE}) \\
I_{GEN} &= f_2(\omega_{FD}, \tau_{ICE}, \omega_{ICE}) \\
I_{BAT} &= f_3(\tau_{FD}, \omega_{FD}, \tau_{Brake}, \tau_{ICE}, \omega_{FD}) \\
m_{fuel_{ICE}} &= f_4(\tau_{ICE}, \omega_{ICE})
\end{aligned} \tag{4.27}$$

5. When a symbolic solution is found, an attempt is made to get a numerical solution by assigning the driver variables a numerical value. If a numerical solution is successfully obtained, then the solution has been found. Else, the another attempt to find the symbolic solution is made with another set of the driver variables. If the list of driver variables is exhausted without finding a symbolic solution, it is assumed the topology is invalid.

## 4.2 Heuristics for Valid Vehicle Topologies

### 4.2.1 Genomes

As shown in section 4.1.1, the topology of a vehicle is represented in a reduced incidence matrix consisting of only zeros and ones. There are many possible matrix permutations;

however, few permutations result in viable topologies. Each of these incidence matrices are represented using a genome. A **genome** contains information describing both the mechanical and electrical topology, and is written as a linear string. For example, the genome for the matrix in equation 4.6 and equation 4.12 is shown in equation 4.28.

$$[010000110101000001010111] \Longleftrightarrow \begin{cases} [A_{f,MECH}] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \\ [A_{f,ELEC}] = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \end{cases} \tag{4.28}$$

The Incidence Matrix is generated by parsing the genome into a matrix of predefined size. The first part of the genome parsed into the mechanical graph, and the second part the electrical graph. For example, in equation 4.28, the genome is 24 characters long, and the matrix dimensions for the mechanical and electrical IMs are 3x7 and 1x3 respectively. The first 21 characters of the genome are parsed row by row into the mechanical matrix, then the last 3 characters are parsed into the electrical matrix.

In this thesis the design space is limited to a 4x9 mechanical matrix because an additional node and 2 columns representing the gearbox are included. The total genome is 39 characters long. The electrical matrix could be expanded; however, this work assumes the converters are integrated with the battery and all electrical components are connected in parallel. If multiple discrete converters or power sources were included in the design space, it would be necessary to increase the number of rows in the electrical matrix to allow for serial electrical connections.

### 4.2.2 Heuristics

Instead of evaluating each genome using driver selection (described in section 4.1.4) it is much faster to use heuristics to eliminate invalid topologies. Each incidence matrix is formed, then checked for the following criteria:

1. An Incidence Matrix cannot have more than a single 1 in each column by definition. According to GTM, components are connected between two nodes. All measurements are taken with respect to the ground node, which appears as a row of -1s, that is removed to obtain the reduced incidence matrix, described in section 4.1.2. Therefore, there should only be one 1 in each column.

2. A row cannot contain a single one. A row with a single 1 represents a component disconnected from the graph (a floating arm), and thus is an invalid topology.

3. The first entry in the genome must be 1. Forcing the final drive to be in the first equation reduces the number of permutations due to isomorphisms.

4. The ICE and the Battery must be in the graph to be a valid HEV powertrain; the columns corresponding to these two components must contain a single 1.

5. If the electric motor / generator is in the mechanical graph, it must also exist in the electrical graph to ensure the correct terminal equations appear in the final system of equations.

6. The Powersplit Device must have either 0 or 3 connections. As the Powersplit Device has 3 flanges, they must all be connected, or none of them connected (not in the graph).

7. Similar to the Powersplit Device, the gearbox must have 0 or 2 connections.

8. There should be no loops between the gearboxes. For example, the discrete gearbox should not connect the sun and the planetary gearset. This effectively reduces the DOF of the gearset from a 3 DOF system to a 1 DOF system, where 1 speed and no torques can be specified. In a dynamic system, such a mechanism will not move and is not useful.

The set of genomes that pass the heuristic criteria are checked for isomorphs. Isomorphism occurs when different genomes result in the same system of equations. This can occur when the rows of the incidence matrix are swapped. To remove isomorphs, a reference genome is selected and its mechanical incidence matrix is constructed. Similarly, a test genome is selected and its mechanical incidence matrix is constructed. If all the rows in the reference genome IM are found in the test genome IM, then an isomorph is discovered and it is removed from the pool of genomes. Alternatively, Gauss-Jordan elimination can be performed on both matrices; if the result is the same, they are isomorphs.

To save computational memory, enumeration was performed on two genomes: one 27 characters long, and one 9 characters long. The two enumerated genomes were appended together. The genomes only contained zeros and ones. The 36 character genome represents a 4x9 mechanical incidence matrix. Each genome was evaluated against the heuristic criteria above. The number of unique genomes was categorized by number of '1s' in the mechanical incidence matrix. The results are presented in table 4.1.

Table 4.1: Number of Unique Genomes per '1' picked in Mechanical Graph

| Pick | # unique genomes |
|---|---|
| 1-2 | 0 |
| 3 | 2 |
| 4 | 4 |
| 5 | 12 |
| 6 | 38 |
| 7 | 36 |
| 8 | 72 |
| 9 | 360 |
| 10+ | 0 |
| Total | 520 |

## 4.3 Summary

In this chapter it was shown that the Graph Theoretic Method can be used to generate the system of equations for a Hybrid Electric Vehicle powertrain. The equations for a Powersplit Hybrid Electric powertrain were derived as an example. A process to select the control variables for any topology was presented. This process is used in chapter 6 to evaluate different topologies.

# Chapter 5

# System Model Validation

To validate the model formulation approach, two architectures are presented and compared against Autonomie: a Powersplit Hybrid and a Parallel Hybrid. Together, these two architectures use all of the component equations presented in this thesis. These system models are validated by the following process:

1. Creating the desired architecture to be validated in Autonomie.
2. Running the simulation in Autonomie.
3. Extracting the control inputs and outputs from Autonomie.
4. Using the control inputs from Autonomie as inputs to ASHev.
5. Comparing the outputs between Autonomie and ASHev.

The model is validated if the two simulations have similar outputs. Both models use the same maps for electric motor/generator power, and ICE fuel consumption. The maps are linearly scaled to fit the component size. Differences in the two models are highlighted in table 5.1.

## 5.1 Verification of Powersplit Architecture versus Autonomie

Shown in figure 5.1, the Powersplit Hybrid has the final drive connected in parallel with the EM and the planetary gearset ring gear; this allows allowing torque to be delivered

Table 5.1: Model Differences Between Autonomie and ASHev

| - | Autonomie | ASHev |
|---|---|---|
| Model Type | Quasi-Static, forward | Static, backward |
| E. Motor/Generator Equation | Interpolation | 4-4 Curve-fit Polynomial |
| ICE Fuel Consumption | Interpolation | Interpolation |
| PSD Efficiency | Losses (3%) | Lossless |
| Discrete Gearbox Efficiency | Losses (interpolated) | Lossless |
| Electric Converter | Dynamic 0.95-1 Efficiency | Static 0.95 |
| Clutch | Yes | No |

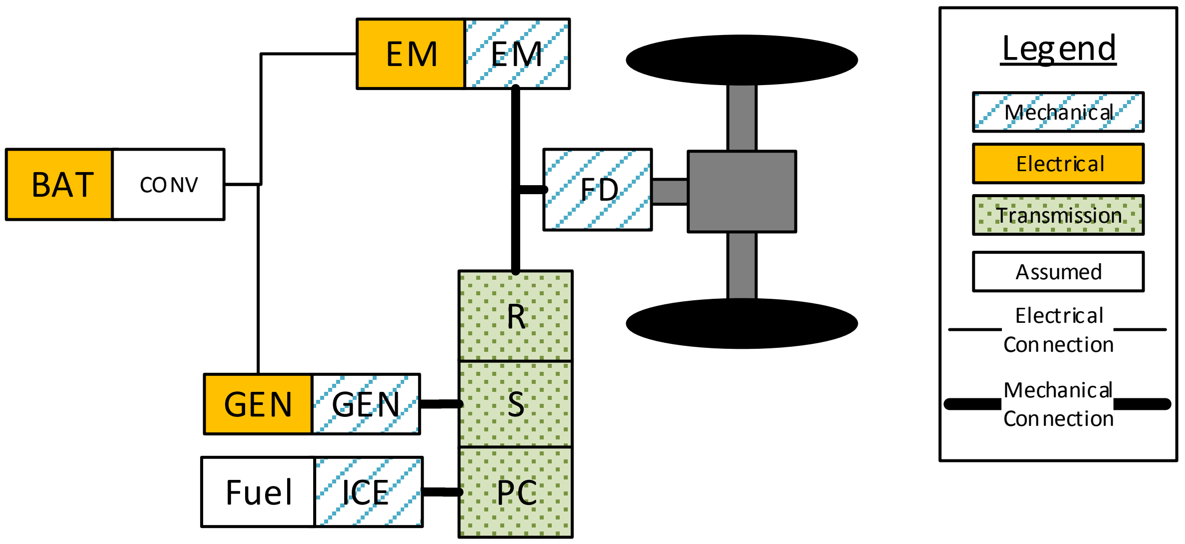

Figure 5.1: Architecture of a Powersplit Vehicle

from either the EM or the ring gear. The second electric motor, GEN is connected to the sun gear of the planetary gearset. The ICE is connected to the planetary carrier. Torque can be transmitted from the ICE to the GEN via the PSD to charge the battery or supply the EM with power. Torque can also be transmitted from the ICE to the FD through the ring gear.

The Autonomie model is simulated using a 0.1s timestep. For validation purposes, ASHev was simulated using both a 0.1s and 1s timestep. The 1s timestep ASHev model used inputs from Autonomie averaged over 1s. The final drive and ICE torque and speed were chosen as inputs. The 0.1 and 1s timestep model have very similar results, so for readability, only select graphs in figures 5.2, 5.3, and 5.4 were chosen to show both results.

The drivecycle speed and torque can be seen in figures 5.2a and b; these are exactly the same because these outputs are taken from Autonomie and used as inputs to the ASHev model. Similarly, shown in figure 5.2c and d, the ICE torque and speed were taken from Autonomie and used as inputs to the ASHev model. Vehicle parameters and component information for these models can be found in table 5.2.

Table 5.2: Powersplit Model Parameters

| | |
|---|---|
| Vehicle Mass | 1400 kg |
| Internal Combustion Engine | Prius MY04, 57kW 1.5L, 4 Cylinder |
| EM1 (Generator) | Permanent Magnet 30kW peak, 14kW continuous |
| EM2 (Traction Motor) | Permanent Magnet 50kW peak, 25kW continuous |
| Transmission | Powersplit device |
| Final Drive Reduction Gear Ratio | 3.93 |
| Wheel Radius | 0.287 m |
| Battery Type | NiMH 51x Series |
| Battery Capacity | 6.5Ah |
| Frontal Surface Area | 1.746 $m^2$ |
| Coefficient of Drag | 0.3 |
| Rolling Coefficient of Friction | 0.015 |
| Input Variables | $\tau_{FD}$, $\omega_{FD}$, $\tau_{ICE}$, $\omega_{ICE}$ |

There are a few differences in the models, but the resulting fuel consumption rates in figure 5.2e are identical. Autonomie uses a different fuel consumption map when the engine is signal off, whereas ASHev only uses one fuel consumption map. Although this phenomenon is not encountered in this particular example, Autonomie allows the engine to slow down due to frictional forces when disconnected mechanically from the powertrain, thus reporting negative torques while still consuming fuel. Shown in figure 5.2f, these assumptions are acceptable as 442.5g is the reported fuel consumption for both models.

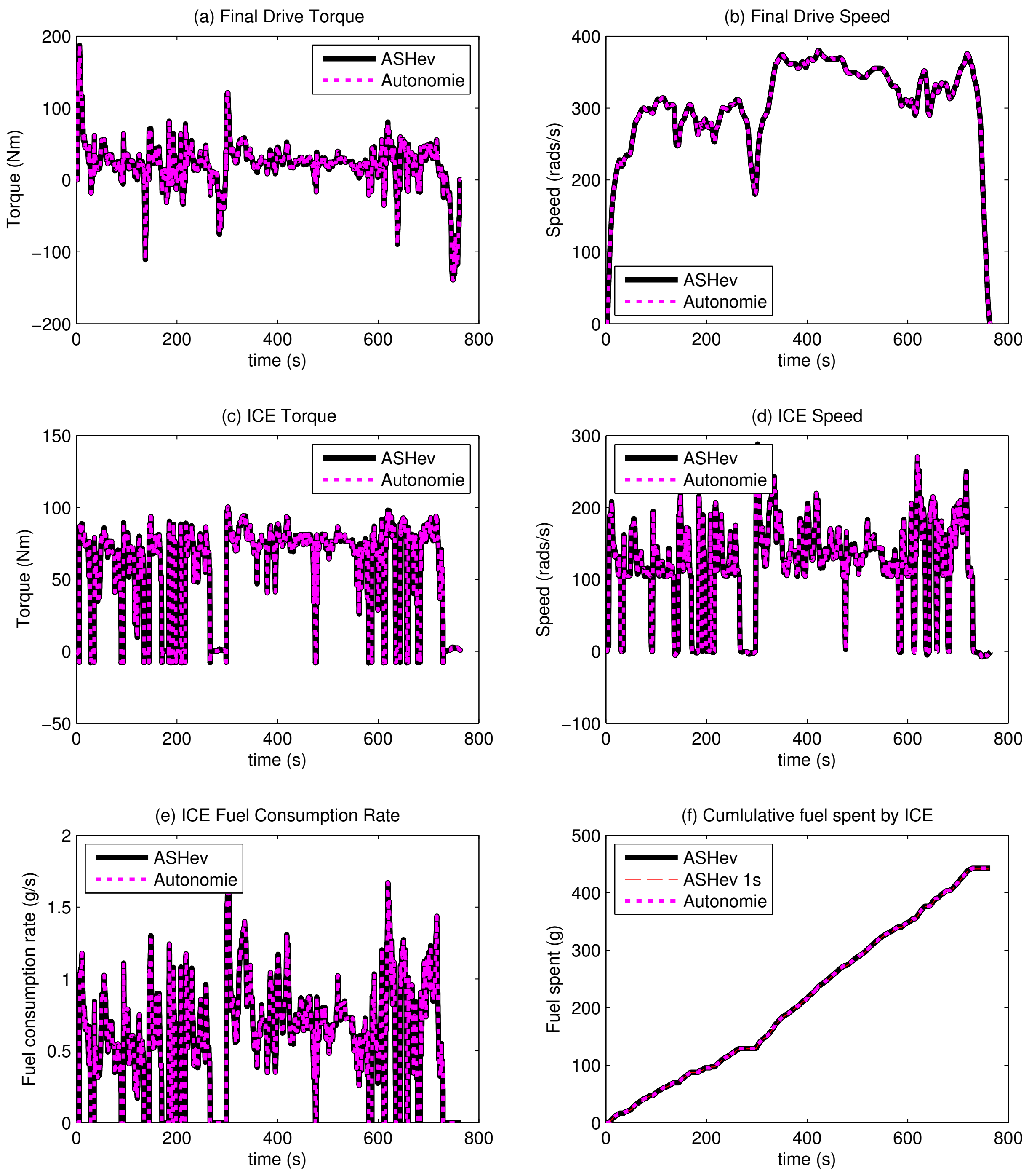

Figure 5.2: Validation Results for Powersplit Architecture: ICE

Shown in figure 5.3a, the Electric Motor torque in ASHev approximately tracks the reference torque. Autonomie assumes there are power losses in the PSD, whereas ASHev assumes no losses. This results in the ICE transmitting less torque than expected through the PSD, and the electric motor must make up the difference. Shown in figure 5.4a, the maximum absolute difference is less than 30Nm.

The electric motor is mechanically connected to the final drive, shown in figure 5.3c, the electric motor tracks the reference speed exactly. Shown in figure 5.3e, the electric motor power approximately tracks the reference. The error is due to model differences; the electric motor power is calculated from a curve fit of an efficiency map provided by Autonomie. There is a small error introduced when curve-fitting a polynomial to a discrete map. Combined with the torque error due to the PSD power losses, this results in a small, but acceptable, power difference. Shown in figure 5.4c, the electrical power error at any one point is less than 10kW (for a 0.1s timestep).

The generator output is shown in figure 5.3b, d, f. The torque error during transient periods is shown in figure 5.3b. The error has three sources: the PSD inefficiencies; rotational inertial forces in the PSD; and frictional forces in the engine. As inertial forces at the component level are ignored in the steady-state models, these terms cannot be corrected. The transients are small and result in an acceptable amount of cumulative error for the calculated generator power. The difference in torque and power between Autonomie and ASHev can be found in figures 5.4b and d.

The battery output is shown in figure 5.4e and f. The battery output is a function of the electric motor and generator power output. As can be seen in 5.4e, the battery power output generally tracks Autonomie's output quite well, even with the error introduced by the torque and efficiency differences in the electric motor and generator. Figure 5.4b shows the battery State-Of-Charge, which indicates the effect of the cumulative error in of the battery power output. The final difference in SOC is -0.0413, which is a 6.4% error. This is an acceptable amount of error for conceptual design.

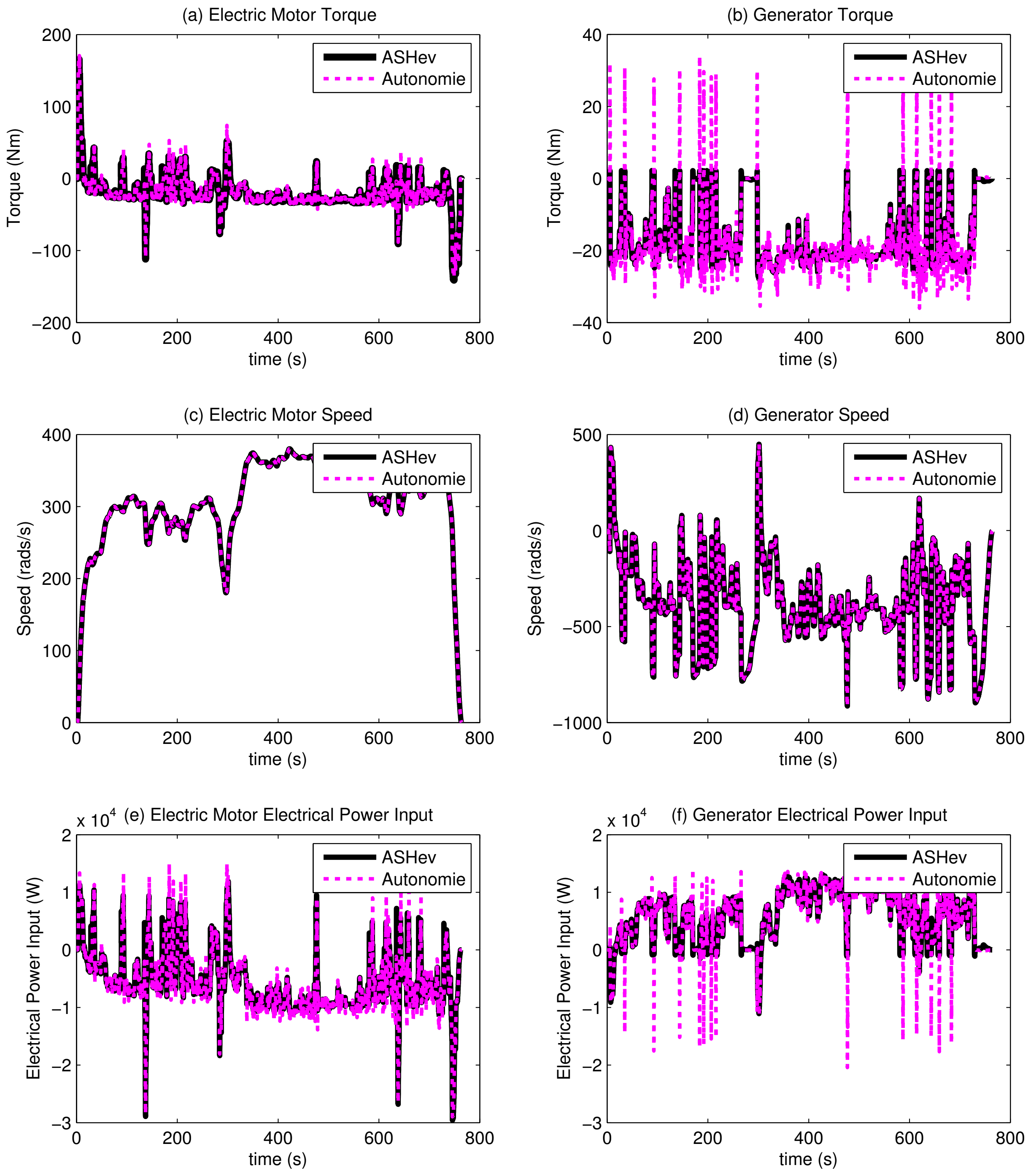

Figure 5.3: Validation Results for Powersplit Architecture: Electric Motor and Generator

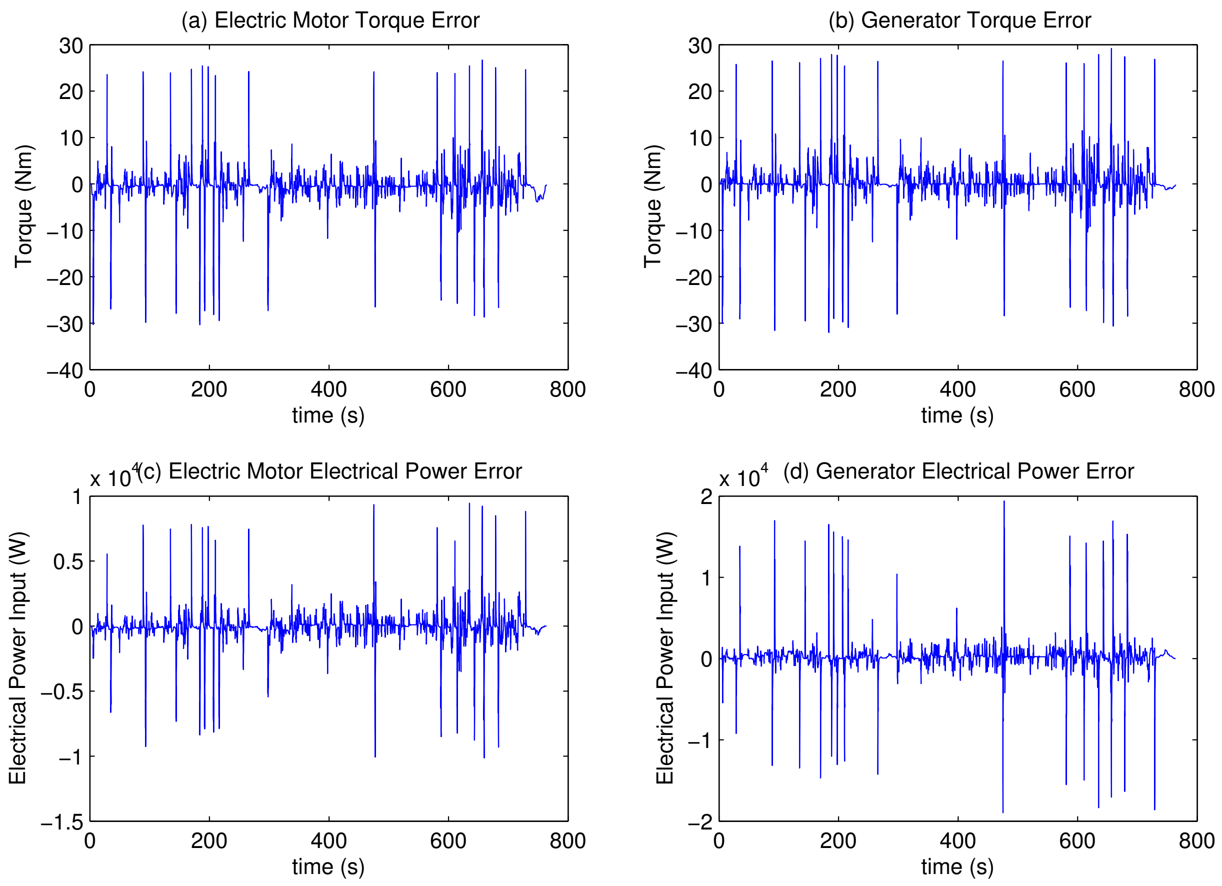

Figure 5.4: Validation Results for Powersplit Architecture: Electric Motor and Generator Error

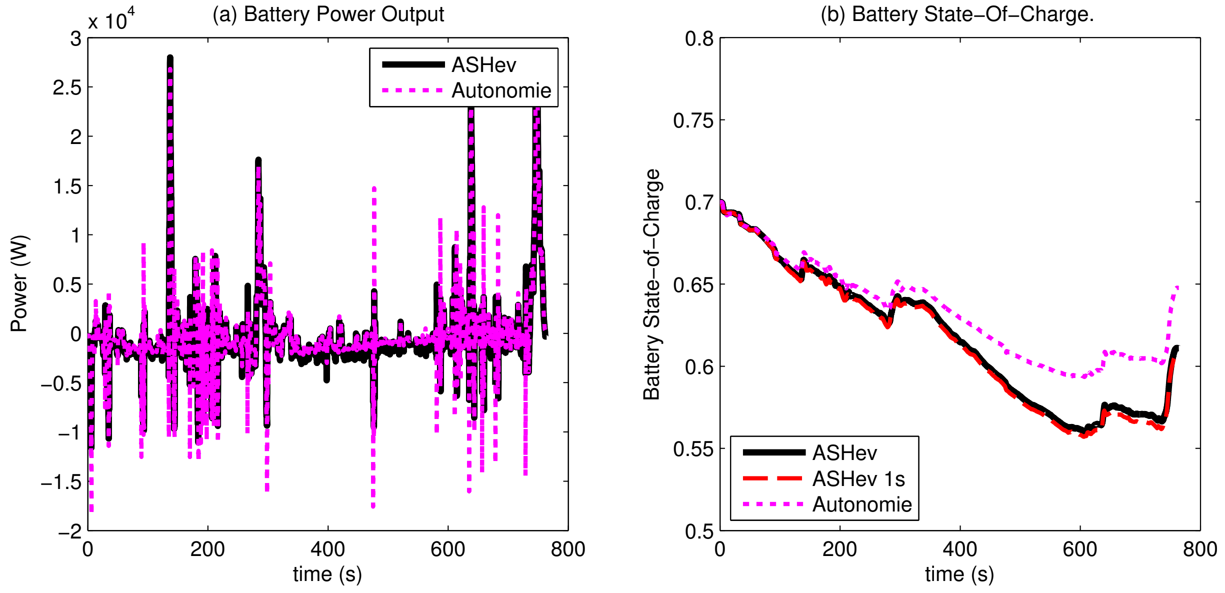

Figure 5.5: Validation Results for Powersplit Architecture: Battery

## 5.2 Verification of Parallel Architecture versus Autonomie

The Parallel system model was validated using a similar approach as described at the beginning of chapter 5. Parameters and component data for the simulation are shown in table 5.3. The final drive torque and speed are specified, thus have exactly the same values as the Powersplit example in figures 5.2a and b. The input to this system was the ICE torque (shown in figure 5.7a) and the gearbox ratio (shown in figure 5.8a).

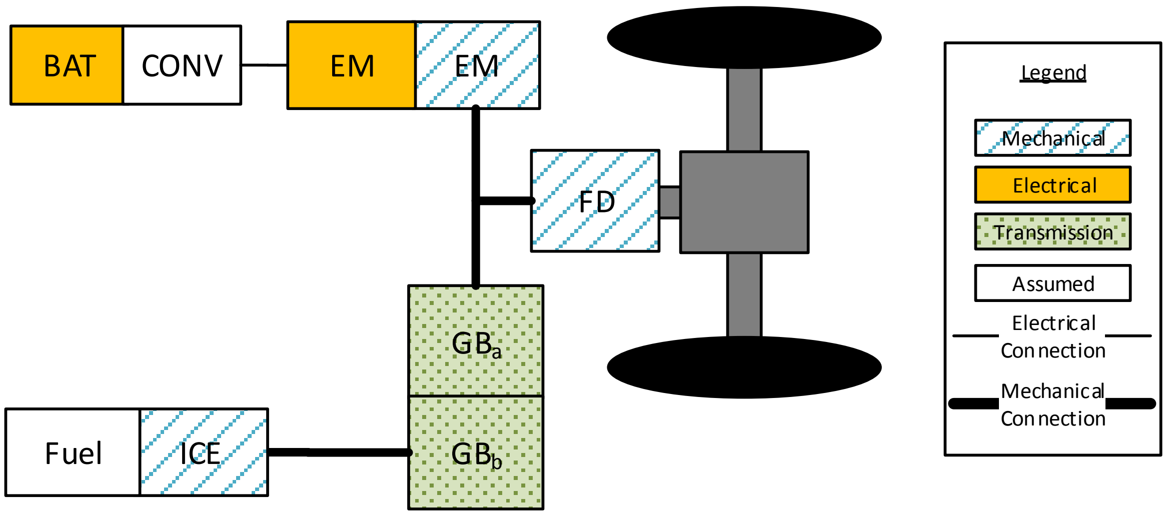

Figure 5.6: Architecture of a Parallel Hybrid Vehicle

Table 5.3: Parallel Model Parameters

| | |
|---|---|
| Vehicle Mass | 1619 kg |
| ICE | Prius MY04, 85kW (scaled) 1.5L, 4 Cylinder |
| EM | Permanent Magnet 25kW peak, 12.5kW continuous |
| Transmission | 5 Speed Manual Gearbox |
| Gearbox Ratios | {3.32,2,1.36,1.01,0.82} |
| Final Drive Reduction Gear Ratio | 3.93 |
| Wheel Radius | 0.287 m |
| Battery Type | NiMH 51x Series |
| Battery Capacity | 6.5Ah |
| Frontal Surface Area | 1.746 $m^2$ |
| Coefficiecnt of Drag | 0.3 |
| Rolling Coefficient of Friction | 0.015 |
| Input Variables | $\tau_{ICE}$, $R_{GB}$ |

Shown in figure 5.7c, the ICE speed in the ASHev model tracks the reference speed in the Autonomie model, except when torque is reported to be negative. Shown in figure 5.7a the negative ICE torques represent the engine slowing down due to frictional forces. This can occur when the clutch between the engine and final drive is open, and no fuel supplied to the engine. Shown in figure 5.8a, Autonomie uses a gearbox ratio of ‘0’ to represent an open clutch. ASHev uses a gear ratio of ‘1’ instead of ‘0’ to avoid division by zero errors. This results in a small error in the cumulative fuel consumption, which is shown in figure 5.7e. The fuel consumption for Autonomie and ASHEV is 640.9 g and 637.3 g respectively; an acceptable -0.55% error.

The electric motor is mechanically connected to the final drive and thus the speed tracks perfectly as seen in figure 5.7d. Shown in figure 5.7b, there is some error between the electric motor torque. The error is because the ASHev model assumes the gearbox is lossless. As the electric motor torque is equal to the difference between the final drive torque and the gearbox output torque, the small error in the gearbox output torque results in error in the electric motor torque. Due to the differences in torque, and that the curve fit to the electric motor efficiency is not perfect, there is some error in the electric motor power input, as seen in figure 5.7f.

As previously stated, ASHev assumes a lossless gearbox, whereas Autonomie assumes a gearbox efficiency of 0.89-99%. The differences can be seen in figures 5.8c and e. This iteration of ASHev does not detect which side of the gearbox is the input, so losses are ignored to avoid an efficiency of greater than 1.

In figure 5.8d, the ASHev battery power output can be seen to approximately track the reference. Shown in figure 5.2e, the battery State-Of-Charge finishes at a higher value. The final State-Of-Charge in the Autonomie model and ASHev is 0.643 and 0.65, or 1.1% error. The higher value is expected; the lossless gearbox results in less energy spent, and a higher final SOC for the same fuel consumption.

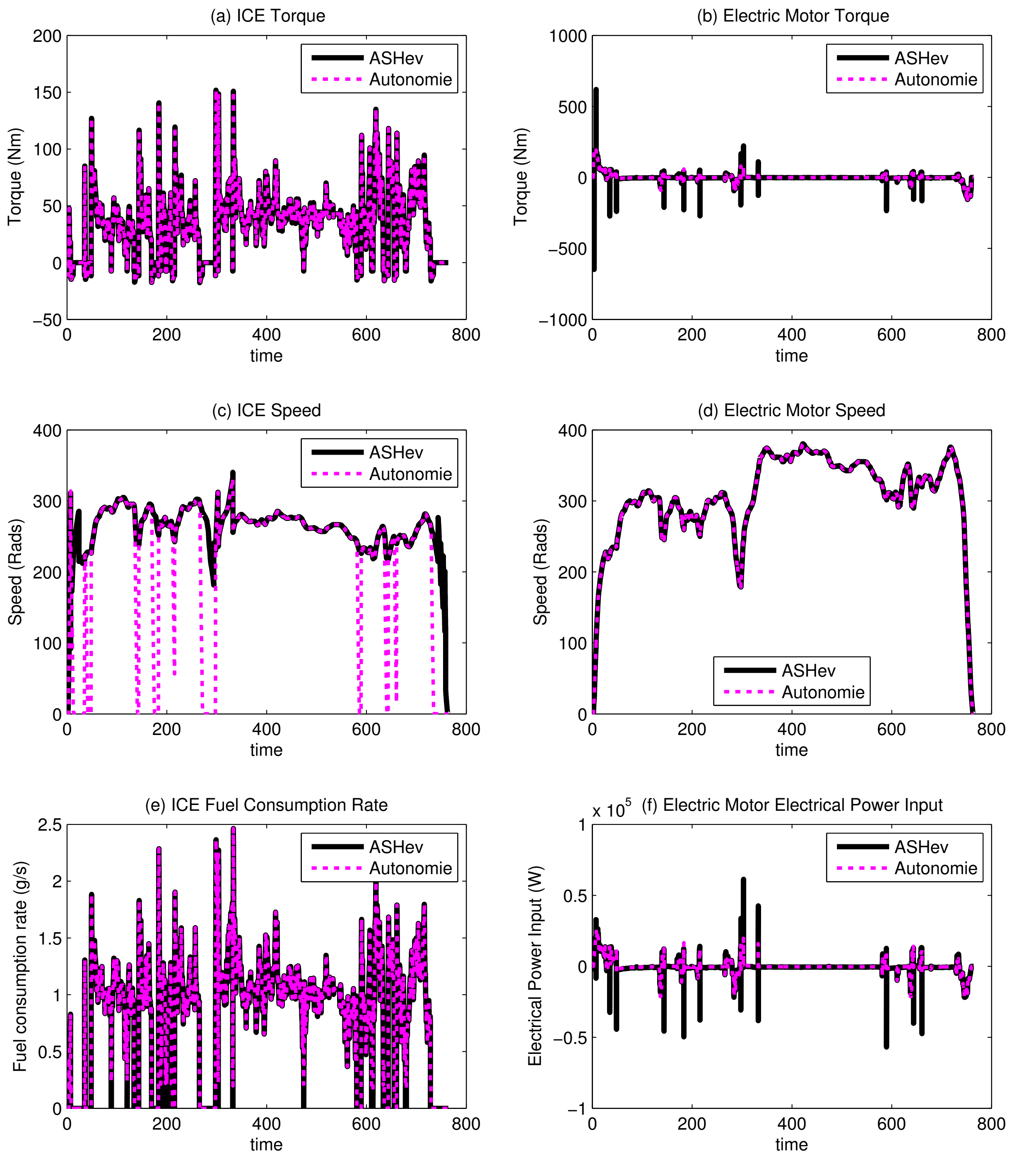

Figure 5.7: Valiation Results for Parallel Architecture: ICE and electric motor

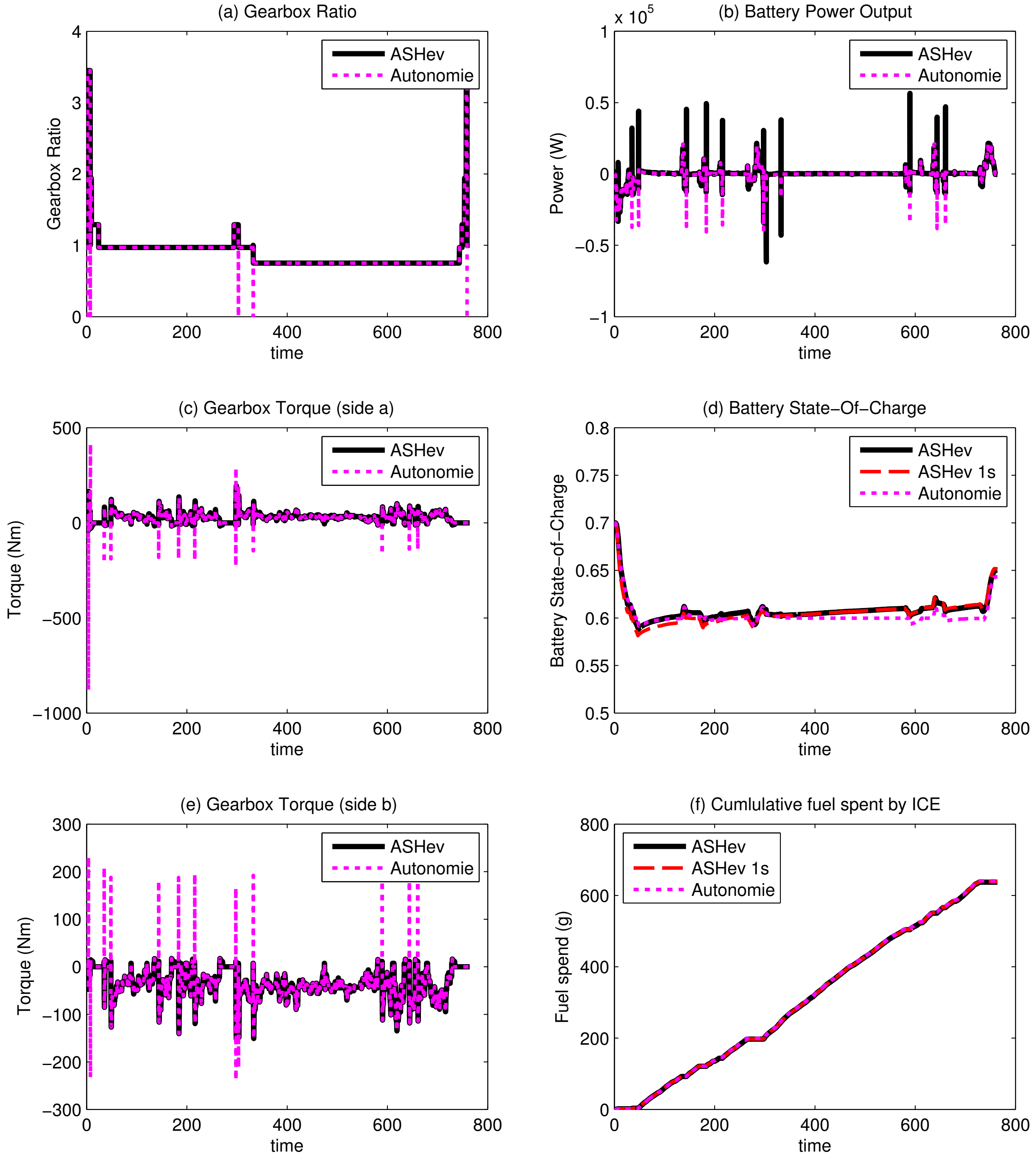

Figure 5.8: Valiation Results for Parallel Hybrid: Gearbox and Battery

## 5.3 Summary

In this chapter two vehicles were validated: Powersplit and Parallel. The ASHev static model is compared to the quasi-static model provided by Autonomie. Numerous assumptions were made about the efficiencies of electric motors and gearboxes. The resulting error between the two models is a 6.4% error in the final battery State-Of-Charge, and less than 1% error in fuel consumption. This small amount of error is considered acceptable for the purpose of this thesis.

# Chapter 6

# Methodology

This section will describe the implementation and optimization of the ASHev model. Matlab 2012b and its native toolboxes were used to implement: component size optimization; component map scaling and curve-fitting; and the Dynamic Program. The Maple 17 symbolic math toolbox was used to implement the Graph Theoretic Method. Fully simulating a topology would take between 1 - 90 minutes on a 2.2 GHz single core CPU; the computation time is dependent on the input resolution and the length of the drivecycle. Approximately 99% of the computation was spent generating calculating the steady-state vehicle response, which was stored in a 'vehicle state lookup table'. As the vehicle is modeled in steady-state, parallel computing could be used to reduce computation time.

### Parallel Computing

Parallel computing was used to significantly reduce simulation time. A cluster computer contains many central processing units (CPU), and each CPU contains multiple cores. Each core can simultaneously perform a different task. A core that is available for work is called an 'open worker'. The cluster used in this work has an architecture shown in figure 6.1. A PC was used to remotely access the HSPC (High Speed Performance Computer) cluster, where Matlab instances with the Maple toolbox could be opened on various nodes. The HSPC contained 2 nodes, each containing 8 CPUs, with 4 cores per CPU; in total 64 workers were available.

Matlab 2012b artificially limits the number of workers to 12 per instance. To maximize worker usage, 4 Matlab instances were opened simultaneously. Each instance was programmed to evaluate a different set of genomes. In total, 48 of the 64 workers were

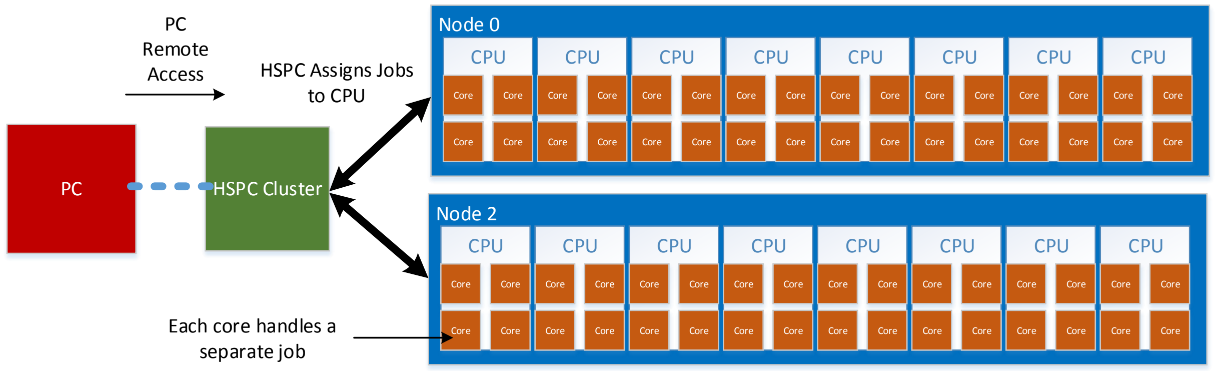

Figure 6.1: Cluster Computer Configuration

used; 16 workers were unused because the HSPC was a shared computing resource. The result was up to a 2000% reduction in computation time; the 90 minute evaluations were reduced to 4.5 minutes. This limitation was removed in Matlab 2014a, and therefore future versions of this work will not need to use multiple instances.

## 6.1 Topology Simulation and Evaluation

### Vehicle State Lookup Table Generation

The torque and speed of the vehicle is calculated for each timestep from a drivecycle. It is assumed that the system can reach any desired behaviour within 1s of applying the input (i.e. engine speed or torque) regardless of the previous state of the system. The control input is discretized over its operation range, and stored in a vector (i.e. $\tau_{ICE} =$ [0,10,20,30..150]Nm). The response of the other components are calculated as a function of the control input and the torque and speed of the vehicle.

In this implementation, the first step of Dynamic Programming is to generate a lookup table of vehicle responses with respect to vehicle speeds and torques and component inputs. This 'vehicle state lookup table' is then used to evaluate the response for various vehicle states.

For each input and timestep the system of equations is solved numerically and stored as a table entry. The procedure is as described below, and a flowchart of the procedure is shown in figure 6.2.

**Vehicle State Lookup Table Generation Process**

1. Accept incidence matrix describing the vehicle topology, and scaling factors describing vehicle component size.

2. Using the Graph Theoretic Method (described in section 4.1), generate the model equations from the incidence matrix.

3. Given a drivecycle, discretize speed into 1s timesteps.

4. Calculate vehicle mass from base chasis mass, and scaled component masses. Calculate the vehicle torque and speed at each timestep in the drivecycle.

5. Determine the appropriate input variables by enumerating the possible drivers, and testing them until a symbolic solution is found. This process is described in detail in section 4.1.4. If no solution is found, then a 'Failed to find symbolic solution' is returned and the corresponding score described in table 6.1 is assigned.

6. The symbolic solution is compiled into a procedure for speed. This procedure calculates the vehicle system response as a function of the inputs and states. This can be achieved using the Maple command $varsol := unapply(symbsol, notdrivers)$, where *varsol* is a function which returns the vehicle system response, *symbsol* is the symbolic solution, and *notdrivers* is a set of the variables in *sys* minus *driverlist*. The arguments to *varsol* are the vehicle speed and torque, and the component inputs.

7. The compiled solution is saved to disk.

8. For every entry in the control input vector, and every vehicle state, the system response is calculated using the symbolic solution from the previous step. For faster simulation, this calculation is parallelized on the cluster computer by:

   (a) Splitting the systems to be evaluated into blocks. For example, there could be more than 4 million systems of equations to solve. A block could consist of 5000 systems.

   (b) Assigning a block to an open worker.

   (c) Prior to calculating the system responses, restart Maple (using $restart()$) and load the symbolic solution from disk.

   (d) All the blocks have been evaluated, the data is collected in local memory.

9. Store collected system states and responses in a 'vehicle state lookup table'.

10. Table entries that violate engine/electric motor torque and speed constraints are eliminated. If there is a timestep for which the vehicle cannot find an input to meet the vehicle performance, a score corresponding to ‘Missing entries in table’ (shown in table 6.1) is returned.

11. Calculate ICE fuel consumption rate based on lookup table.

12. Export table to Dynamic Program.

It was found that using the Maple toolbox to repeatedly evaluate functions would cause the Maple kernel to accumulate large amounts of memory and eventually crash. This is likely due to faulty garbage collection. To work around this problem, the symbolic solution to the system of equations was solved to disk and *restart*; was periodically invoked to clear Maple’s memory. After the memory was cleared, evaluating the functions could be resumed by reading the symbolic solution from disk.

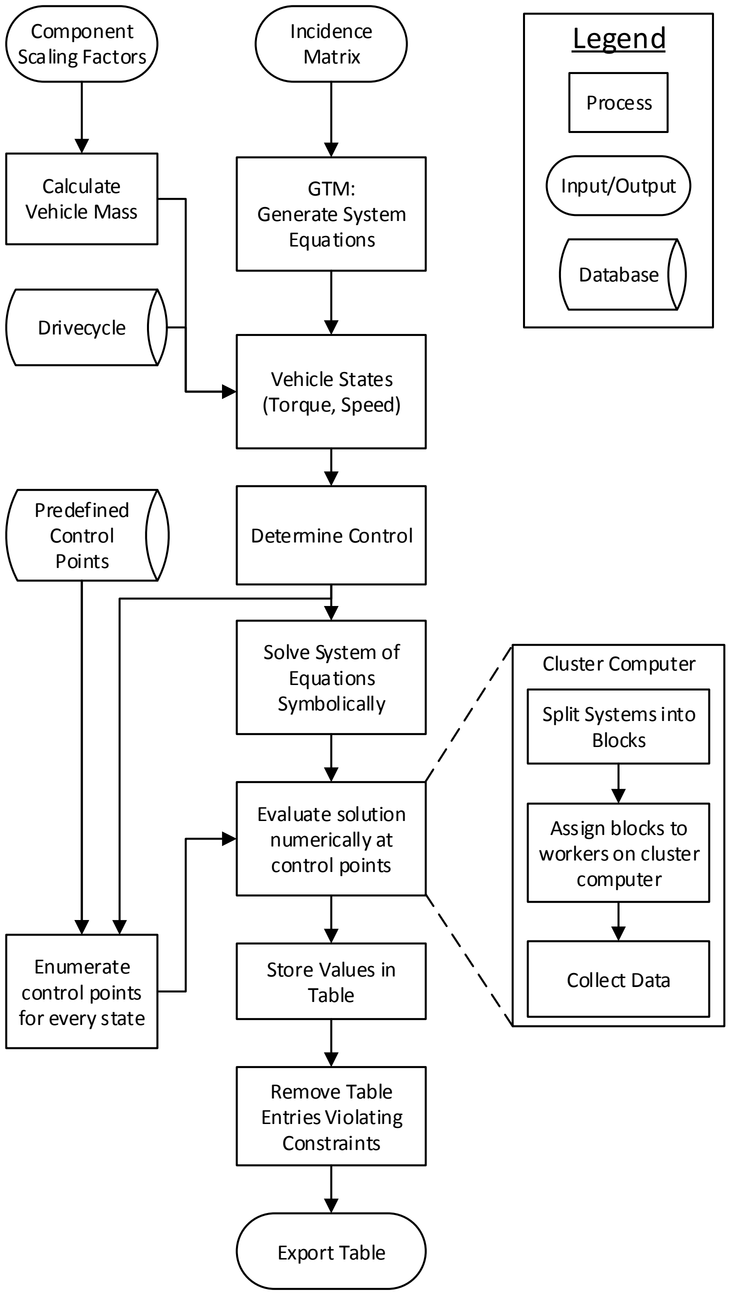
Component Scaling Factors
Incidence Matrix
Legend
Process
Input/Output
Database
Calculate Vehicle Mass
GTM: Generate System Equations
Drivecycle
Vehicle States (Torque, Speed)
Predefined Control Points
Determine Control
Solve System of Equations Symbolically
Cluster Computer
Split Systems into Blocks
Assign blocks to workers on cluster computer
Evaluate solution numerically at control points
Collect Data
Enumerate control points for every state
Store Values in Table
Remove Table Entries Violating Constraints
Export Table

Figure 6.2: Flow Chart for Simulating Vehicle States and Inputs.

## 6.2 Dynamic Programming

Dynamic programming is a popular off-line method to determine the optimal control law given all prior trip information [63, 49, 64]. The dynamic program is based on Bellman's Principle of Optimality [14], which states that a problem can be broken into sub-problems, and the solution to each sub-problem is dependent to the sub-problem before it. By solving backwards in time, the optimal control law for each time step is obtained. When the final time step is reached, the control law can be followed forwards in time, and the optimal control law is obtained.

At every timestep, every possible input is evaluated for every possible state and assigned an objective score. For each state, the input which minimizes the objective score is selected. This is effectively a method of enumeration, making it robust but also slow. Therefore the code implementation is extremely important, which is described in section 6.2.1.

### 6.2.1 Dynamic Program Pseudocode

#### Dynamic Program Terminology

Pseudocode for the dynamic program is provided in this section. A description of the variables used in the pseudocode will be provided first for readability.

Continuing the example of a Powersplit Hybrid, the input is assumed to be $u = f(\tau_{ICE}, \tau_{brake}, \omega_{ICE})$. The vector of all inputs is defined as $U = (u_1, u_2, ...u_n)$, where $n = n_{\tau_{ICE}} * n_{\tau_{brake}} * n_{\omega_{ICE}}$, and $n_{\tau_{ICE}}, n_{\tau_{brake}}, n_{\omega_{ICE}}$ are equal to the number of elements in the vector describing the input resolution. For example, if the input discretization was done as follows: $\tau_{ICE} =$ [0,10,20..150], $\tau_{brake} =$ [0,50,100], $\omega_{ICE} =$ [0,25,50..450], then $n_{\tau_{ICE}} = 15$, $\tau_{brake} = 3$, $\omega_{ICE} = 18$. Therefore the vector of $U$ contains $15 * 3 * 18 = 810$ elements.

In this work, $X$ is a vector which represents the battery State-Of-Charge discretized between its minimum and maximum charge. For example, $X = [0.5000, 0.5002, 0.5004...0.9000]$. $X_{tmax}$ is the state at the final timestep (first subproblem) and $X_{target}$ is the desired final state.

$G(u)$ is a matrix representing the score of each element of vector $U$ applied to each element of vector $X$ for a given subproblem. In other words, it is the objective score of each input applied to each state for a given timestep. $G(u)$ is calculated in equation 6.1.

$$G(u) = \dot{m}_{fuel} * \left[3.6 * 10^6 \left[\frac{J}{kWh}\right]\right] - P_{BAT} \tag{6.1}$$

$J(X_t)$ is a matrix representing the cumulative score to travel from $X$ at timestep $t$ to the final state $X_{tmax}$. $J(X_t)$ represents the minimum cost to go from $X$ at timestep $t$ to the final state $X_{tmax}$.

**Dynamic Program Pseudocode Process**

1. With reference to the top of figure 6.3, receive lookup table.

2. At the final timestep, calculate initial objective function by applying an off-target penalty, $J_{kmax} = 1e12 * (X_{tmax} X_{target})$.

3. Starting at the second-final timestep and solving backwards in time, loop:

    (a) Calculate a grid-matrix of next states by adding a specified state (battery SOC) vector $X = (x_1, x_2, ..., x_m)$ to a vector of inputs $U = (u_1, u_2, ...u_n)$.

    (b) Calculate the matrix $G(u)$ from the inputs and specified state vector, where $G(u)$ is described in equation 6.1. If violating constraints, add penalties to objective function. Constraints include:

        i. Maximum charge/discharge power as a function of battery SOC.

        ii. Gear change ratio must be $+1, -1, 0$.

        iii. $SOC_{min} < SOC < SOC_{max}$.

    (c) Calculate matrix of $J(X_{t+1})$; the cost to go to from the current state $x_{i,t}$ to the next state $x_{j,t+1}$.

    (d) Calculate cost $J(X_t) = J(X_{t+1}) + G(U)$.

    (e) For each $x_{i,t}$, $min(J_i)$. The corresponding $U^*$ is the optimal control for $x_{i,t}$.

    (f) Store $u^*_{i,t}$ for every $x_{i,t}$.

    (g) Return to 3a, $t = t - 1$. Terminate at $t = 0$ and pass matrix $J$ to forward solver in step 4.

4. With reference to figure 6.4, start the forward solver. Let $x^*_1 = x0^*$, where $x0^*$ is the predefined initial value of battery SOC.

5. Starting at $t = 0$, and using table of $U^*_t$ for every $X_t$ get $u^*_t$ for specified $x^*_t$. If $x^*_t$ is not a value on state vector X, use the nearest neighbour instead. If $SOC_{min} > x^*_t$ or $SOC_{max} < x^*_t$ (exceed battery SOC bounds), apply a $SOC_{BoundViolationPenalty}$ penalty as described in table 6.1.

6. Calculate $x_{t+1}^* = x_t^* + u_t$

7. Let $t = t + 1$. Terminate when $t = tmax$.

8. If $x_{final}^* \neq SOC_{final}$ apply a $SOC_{FinalTargetPenalty}$ penalty as described in table 6.1.

It is interesting to note that the table $J$ obtained at step 3(g) contains the decisions which minimize the objective score to go from any state $X_t$ to the final state $X_{tmax}$. Approximately 95% of the computation time in Dynamic Programming is spent generating this table, and the remaining 5% of time is spent at the forwards solver. Using this method, results for different initial states $x0^*$ can be quickly computed without additional calculations.

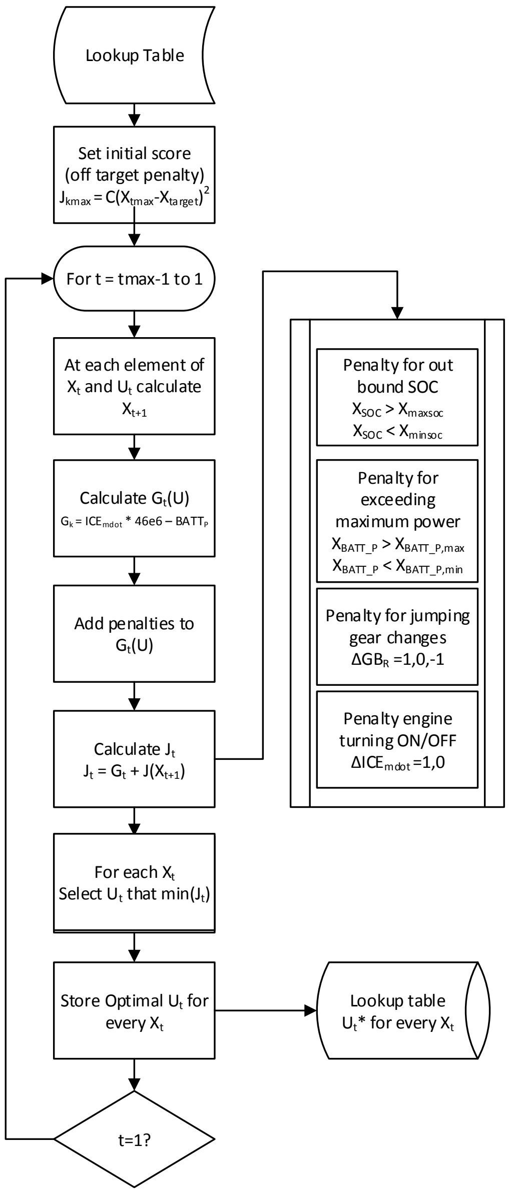
Lookup Table
Set initial score (off target penalty)
$J_{kmax} = C(X_{tmax} - X_{target})^2$
For t = tmax-1 to 1
At each element of $X_t$ and $U_t$ calculate $X_{t+1}$
Calculate $G_t(U)$
$G_k = ICE_{mdot} * 46e6 - BATT_P$
Add penalties to $G_t(U)$
Calculate $J_t$
$J_t = G_t + J(X_{t+1})$
For each $X_t$ Select $U_t$ that $\min(J_t)$
Store Optimal $U_t$ for every $X_t$
t=1?
Penalty for out bound SOC
$X_{SOC} > X_{maxsoc}$
$X_{SOC} < X_{minsoc}$
Penalty for exceeding maximum power
$X_{BATT\_P} > X_{BATT\_P,max}$
$X_{BATT\_P} < X_{BATT\_P,min}$
Penalty for jumping gear changes
$\Delta GB_R = 1,0,-1$
Penalty engine turning ON/OFF
$\Delta ICE_{mdot} = 1,0$
Lookup table $U_t$* for every $X_t$

Figure 6.3: Flow Chart for Dynamic Programming

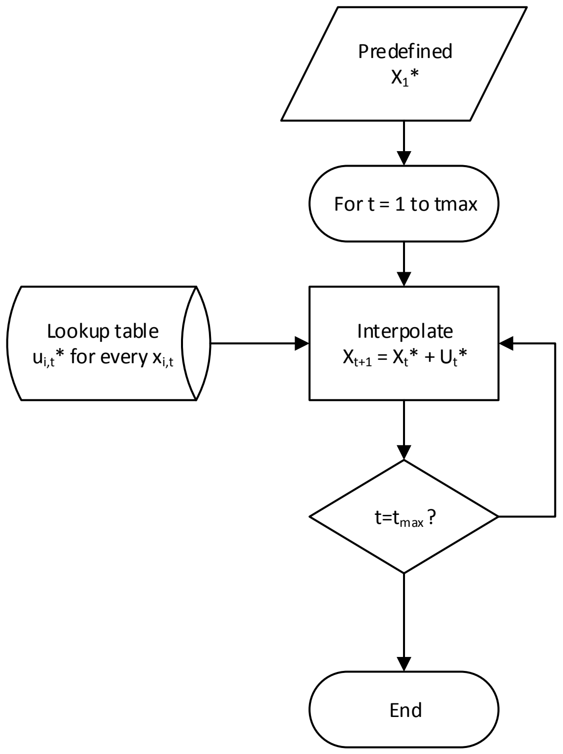
Predefined $X_1$*
For t = 1 to tmax
Lookup table $u_{i,t}$* for every $x_{i,t}$
Interpolate $X_{t+1} = X_t^* + U_t^*$
$t=t_{max}$ ?
End

Figure 6.4: Forward Solve Flow Chart for Dynamic Programming

### Matlab Vectorization Method

The Dynamic Program implementation in Matlab was based on an approach developed by Guzella (see chapter on Dynamic Programming for pseudocode in [47]). In Matlab, a code vectorized approach is orders of magnitude faster than a loop approach [47]. This process is graphically described in figure 6.5.

1. To enumerate each input for each state, two grids were used: the state grid with identical states along the rows, and an input grid with identical states along the columns. Adding these two matrices together resulted in a matrix of the next states.

2. The cost function matrix ($J(X_{t+1})$) is calculated for the next state matrix. Linear interpolation is used to calculate scores that fall inbetween scores. Matlabs $linearp1()$ contains rigorous checks, and is relatively slow. In this implementation a custom speed-optimized interpolation function was available at [65] was used.

3. The input cost matrix is calculated from the input grid matrix $G(U)$

4. The cost matrix J(Xt) was calculated from the input cost matrix $J(X_{t+1})$

5. Using min(), on the cost matrix $J(X_t)$ should return a column vector of the minimum cost of every row, and the array index position of the minimum cost

### 6.2.2 Dynamic Program Settings

A convergence study was performed to determine the best settings for Dynamic Programming. The variables studied in the convergence study are: the number of discretized inputs, resolution of the state grid, and solver time. The study was performed on an Intel Dual Core 2.2 GHz, 6 GB RAM desktop computer; however,only 1 core was used for table generation and Dynamic Programming. The system was simulated over the HWFET drivecycle which contains 765 discrete 1 second timesteps [66]. Shown in figure 6.6, it was found that increasing the number of inputs would greatly increase the overall solve time due to the greater amount of table entries, but have relatively little effect on the dynamic program computation time. The input vector $U$ was limited to 1035 elements to reduce solve time to 10 minutes. The discretization values can be found in the dynamic program settings in the results section.

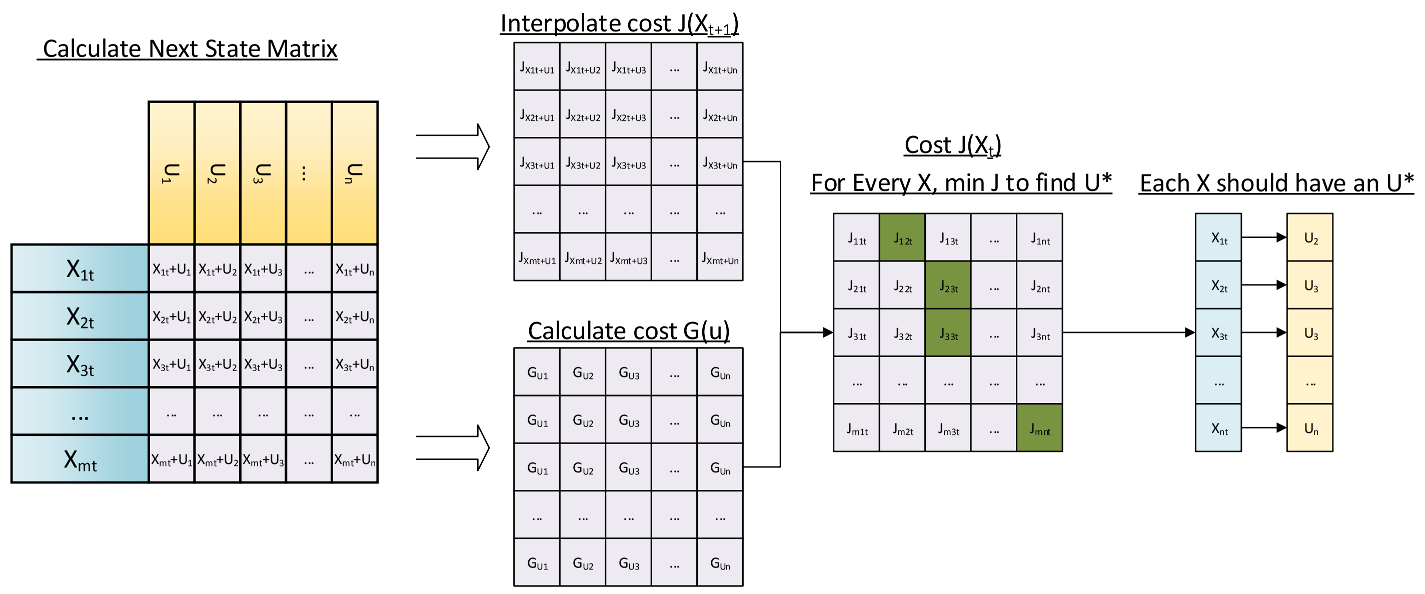
Calculate Next State Matrix
Interpolate cost $J(X_{t+1})$
Calculate cost G(u)
Cost $J(X_t)$
For Every X, min J to find U*
Each X should have an U*

Figure 6.5: Code Vectorization method for Dynamic Programming

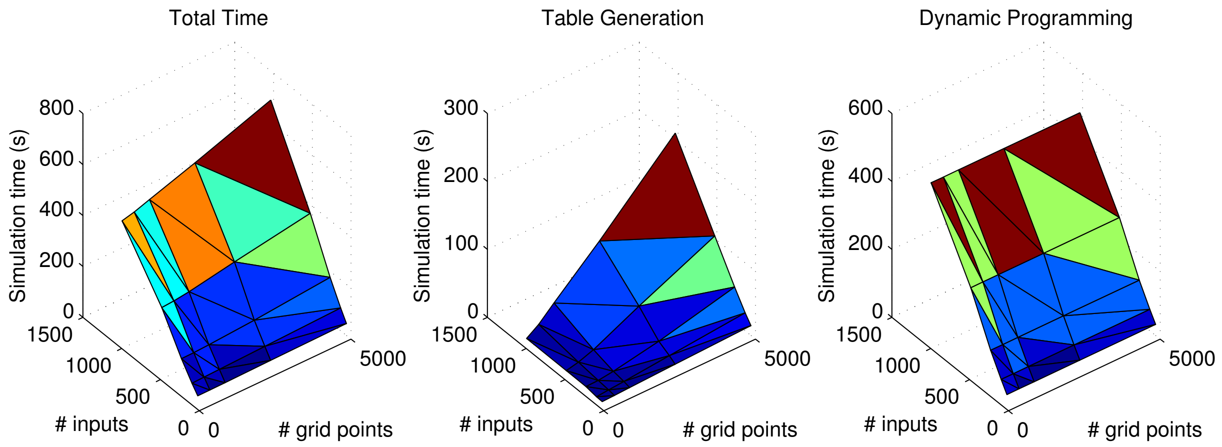

Figure 6.6: Solver Time versus Number of Discretized Inputs and Number of Dynamic Program (Battery SOC) Grid Points.

As the proposed dynamic program allows for off-grid state values through interpolation, it is also possible for there to be error in the final state. Shown in figure 6.7, about 2000 SOC grid points are required to have less than 1% error.

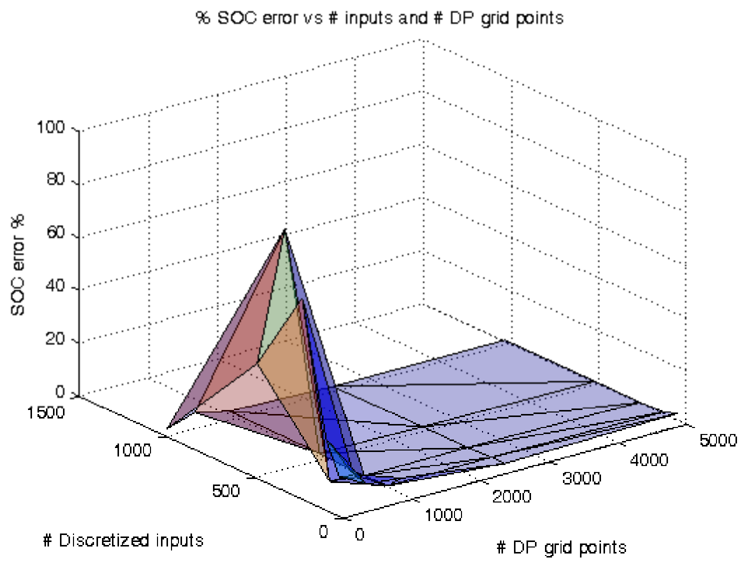

Figure 6.7: SOC Error versus Number of Discretized Inputs and Number of DP (Battery SOC) Grid Points.

Autonomie's rule based controller and ASHev's optimal controller obtained from Dynamic Programming were compared using the aforementioned settings. A Powersplit hybrid simulation was built in Autonomie, and the control signal ($\tau_{ICE}$, $\omega_{ICE}$) was extracted. Shown in 6.8, the Autonomie control strategy was applied to the ASHev model. The same initial and final conditions for the battery SOC were set in the Dynamic Program.

As expected, seen in figure 6.8, third graph from the top, the Dynamic Program control strategy uses less fuel than the rule based controller. Autonomie consumes 523.3g (55.3 MPG), whereas the dynamic program consumes 397.1g (72.9 MPG). In figure 6.8, bottom graph, it can be seen that the ICE will only output torque at its most fuel efficient operating points. Shown in figure 6.9, the engine operates at the highest efficiency at high torques.

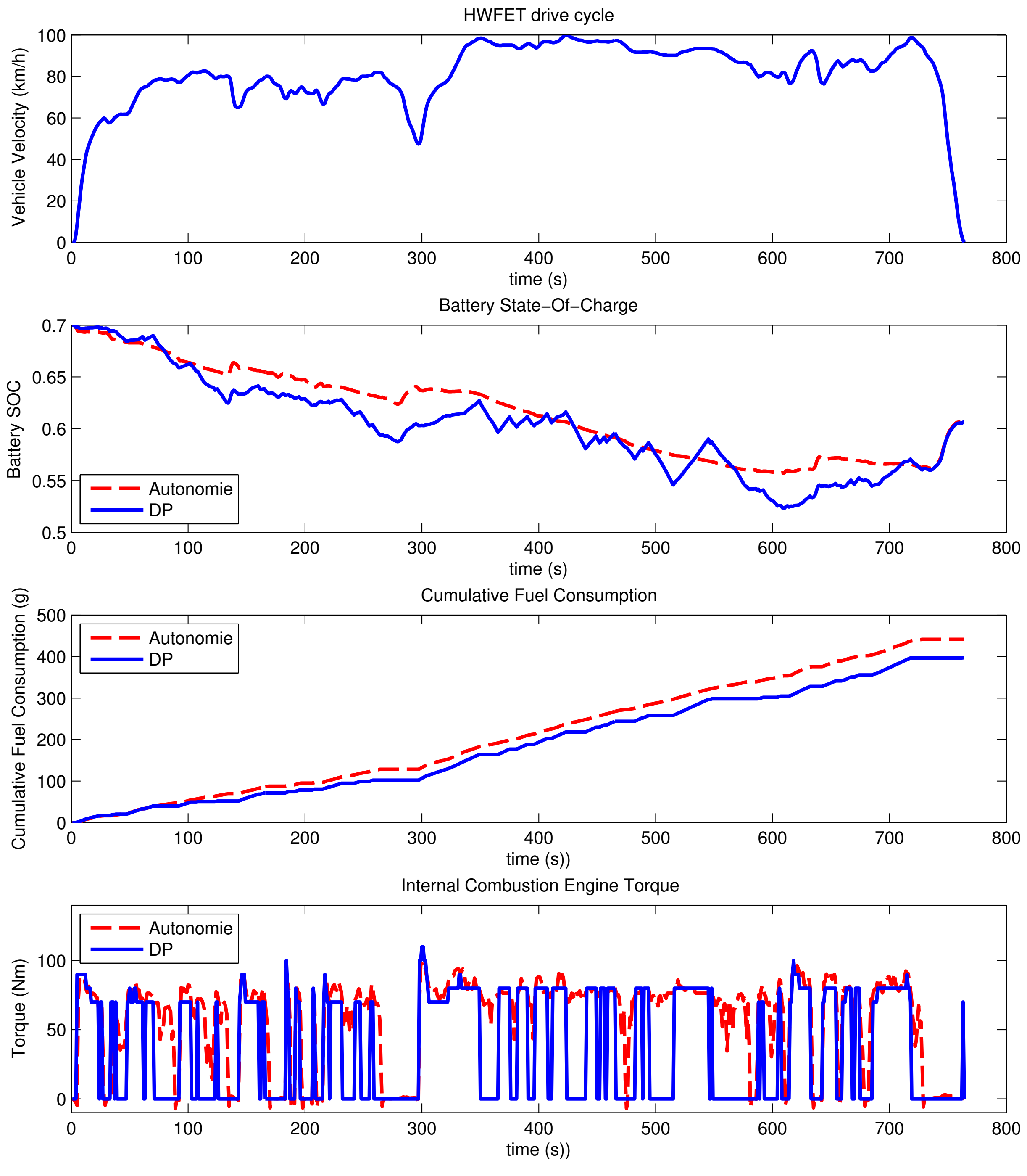

Figure 6.8: Comparison of Autonomie Control Strategy versus Dynamic Program Strategy. Both Simulations were Performed Using the ASHev model.

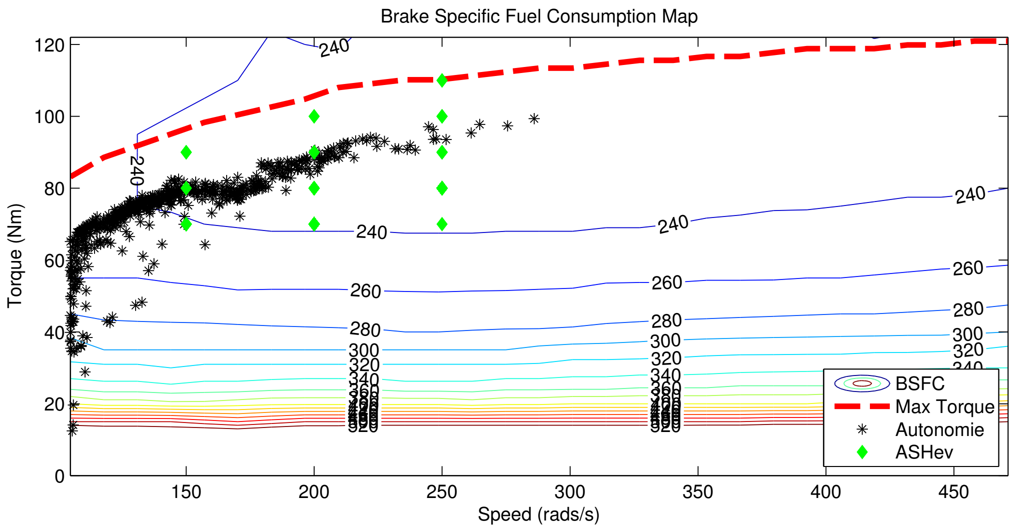

Figure 6.9: Comparison of Autonomie Control Strategy versus Dynamic Program Strategy. Both Simulations were Performed Using the ASHev model.

The engine operating points selected by DP and Autonomie are shown in figure 6.9. Autonomies' control algorithm selects the highest efficiency region for the speed, and thus it is optimized locally at the engine level. Although it appears that only a few operating points have been selected by Dynamic Programming, these operating points (torque and speed) have been repeatedly selected. Dynamic Programming achieves a lower fuel consumption by optimizing the fuel consumption globally over the entire trip, rather than only optimizing it locally. It is likely that increasing the resolution of the inputs will slightly increase the fuel efficiency of the vehicle, but at a significant increase in computation time.

## 6.3 Objective Score

Shown in table 6.1 are the scenarios for calculating the objective score. The following list is a description of the objective scoring outcomes.

- **FailValue** is an arbitrarily set value which should be more than triple the expected fuel spent during a drivecycle.
- **Pass** is the best-case scenario: the algorithm completes without error, and the fuel consumption over a drivecycle is returned. Typical values are under 1000.

- **Fail to find symbolic solution** is the worst-case scenario, and a value based on $FailValue$ is returned.
- **Missing entries in table** occurs when a symbolic solution can be found, but the components are sized such that it cannot meet the performance targets.
- **Pass with off target final SOC/out of bound SOC** - Final SOC is out off target, or SOC violates SOC min/max bounds.

Table 6.1: Values and Calculations for Objective Score

| **Value** | **Objective Score Value** |
|---|---|
| $FailValue$ (default) | 4000 |
| Pass | Calculated Fuel Consumption (g) |
| Fail to find symbolic solution | $FailValue$ |
| Missing entries in table | $FailValue * 0.66 + FailValue * 0.33 * (\#TimestepsMissed)/(\#TimestepsInDrivecycle)$ |
| Pass with off target final SOC/out of bound SOC | Calculated Fuel Consumption (g) $+SOC_{FinalTargetPenalty} + SOC_{BoundViolationPenalty}$ |

A penalty is applied based on the absolute value of the off-target. The 'equivalent fuel consumption' penalty is calculated by using equation 6.2.

$$\begin{aligned} SOC_{FinalTargetPenalty} &= |FinalSOC - FinalTargetSOC| * dSOCtoGas \\ SOC_{BoundViolationPenalty} &= \sum(|OutOfBoundSOC|) * dSOCtoGas \end{aligned} \tag{6.2}$$

where the value of $dSOCtoGas$ is shown in equation 6.3. The last value in the equation (0.153) is assumed to be representative of a power plant charging the battery. This is be an approximation of charging the battery to a higher initial State-Of-Charge so it can avoid the minimum SOC constraint. Assuming a power plant can convert gasoline to electricity at a 20% efficiency, power transmission is 85% efficient, and battery charging is 90% efficient; the overall system efficiency is 15.3% [67, 68]. Performing this way is penalized; the lower end efficiencies for power conversion and transmission were chosen.

$$dSOCtoGas = 3600s * BAT_{Cap}(Ah) * (271V)/46e3(kJ/g)/0.153 \tag{6.3}$$

## 6.4 Pattern Search

Generalized Pattern Search belongs to a category of direct search methods that do not require derivative information to determine search directions. The Pattern Search optimization algorithm was chosen for sizing components because it is a well-known global optimization method with a relatively fast convergence rate [69]. A global optimization algorithm is required because the objective score is non-continuous; 'cliffs' can be created when a topology does not meet the performance requirements. As the objective function is a functional (function of a function), it is not differentiable.

Pattern search systematically finds the global optimum through exploratory moves. At each iteration, the objective function is sampled around the iterate point. The sampled point with the lowest objective score becomes the iterate point for the next iteration. The Pattern Search algorithm is briefly described below:

**Pattern Search Algorithm [70, 71]**

1. Staring at $x_k$, compute the objective function $f(x_k)$. Loop:
2. Determine step length and direction, $s_k$.
3. Evaluate the difference in objective function, $\rho_k$ around $x_k$, where $\rho_k = f(x_k) - f(x_k + s_k)$.
4. If the new objective function is smaller, $\rho_k > 0$, that point becomes the starting point for the next iteration, $x_{k+1} = x_k + s_k$. Otherwise, no change for the next iteration, $x_{k+1} = x_k$.
5. Update the search direction matrix $C_k$ and step length variable $\Delta_k$.
    (a) If $x_{k+1} = x_k$, shrink step length $\delta_k$ a factor $\theta$: $\Delta_{k+1} = \theta\Delta_k$.
    (b) $C_k$ is updated depending on the type of pattern search performed. Generally it is a matrix containing one orthogonal direction for every dimension.
6. While step length is larger than a tolerance, $\Delta_k > \Delta_{tol}$ return to step 2.

Like any global optimization algorithm, convergence is guaranteed, but convergence to the global minima is not [70, 71].

## 6.5 Summary

This chapter described the implementation for creating and comparing vehicle topologies. Topology equations are generated based on the incidence matrix. These equations are used to solve the system under different operating conditions, and the results are stored in a vehicle state lookup table. The lookup table is passed to Dynamic Program where the optimal control law is determined through enumeration of the table. The result is assigned an objective score. Pattern search is used minimize this objective score by varying the component sizes.

# Chapter 7

# Results and Discussion

As evaluating every permutation may be extremely time consuming, a screening process is used to eliminate infeasible topologies so only promising topologies are evaluated. This chapter breaks down the screening process into 4 stages. The first stage uses heuristics to quickly eliminate infeasible topologies, and reduce the number of genomes to a workable number (described in section 4.2.2). The second stage eliminates topologies that cannot meet driving requirements based on a section of the US06 drivecycle. Component sizing is performed during the second stage to ensure the topology is feasible, regardless of the component size. The third stage uses a performance-focused drivecycle based on the PNGV (Partnership for New Generation of Vehicles) requirements to screen out poorly performing topologies. The genomes from the third stage are ranked, and the top 20 continue to the fourth stage. In the fourth stage, the component sizes are optimized for a PNGV-HWFET drivecycle.

## 7.1 Multi-Stage Topology Screening

As previously mentioned, using a single threaded CPU to simulate a drivecycle such as HWFET may take up to 90 minutes. Therefore, it is impractical to blindly attempt to optimize all possible topologies. Instead, a method is presented which systematically reduces the number of genomes by using short drive cycles and performance targets. This method is presented below and in a flow chart shown in figure 7.1.

1. All permutations of genomes were generated using combinatorics. Genomes were parsed into mechanical and electrical IM, and evaluated against heuristics for validity.

Of the $6.78 * 10^{10}$ combinations, only 524 genomes could generate valid powertrain. A breakdown of these genomes is shown in table 4.1.

2. The genomes' first 3 components (ICE/EM/GEN) were optimized for part of the US06 drivecycle, as shown in figure 7.3. This short 'gentle' drivecycle was used to screen out topologies that had extremely poor driving performance or lacked regenerative braking. US06 was chosen because it is a standard drivecycle that slowly accelerates to 110 km/h.

   Component size was optimized for each topology to ensure it could meet the performance criteria regardless of the component size. The optimization used the scaling factors as arguments and the fuel consumption as the objective function. To reduce computation time, only the sizes of three of the five components (ICE/EM/GEN) were optimized. The battery and gearbox ratios were assumed to not greatly affect driving performance, especially on a short, low-performance drivecycle, . If the topology cannot meet the performance criteria, the genomes were assigned a 'Missing entries in table' score described in 6.1. 193 genomes were found to meet the US06 driving performance.

3. The 193 genomes were screened for driving performance using a drivecycle based on PNGV design goals, shown as the second graph in figure 7.2. All 5 component sizes were optimized. 159 genomes passed the performance screening. Of the 159 topologies, the best 20 which minimized fuel consumption were taken to the next round.

4. The top 20 topologies that minimized fuel consumption were chosen and optimized against the PNGV performance and HWFET drive cycle, seen in the bottom graph in figure 7.2. The top 20 topologies are shown in figure 7.4, and are shown in table 7.5.

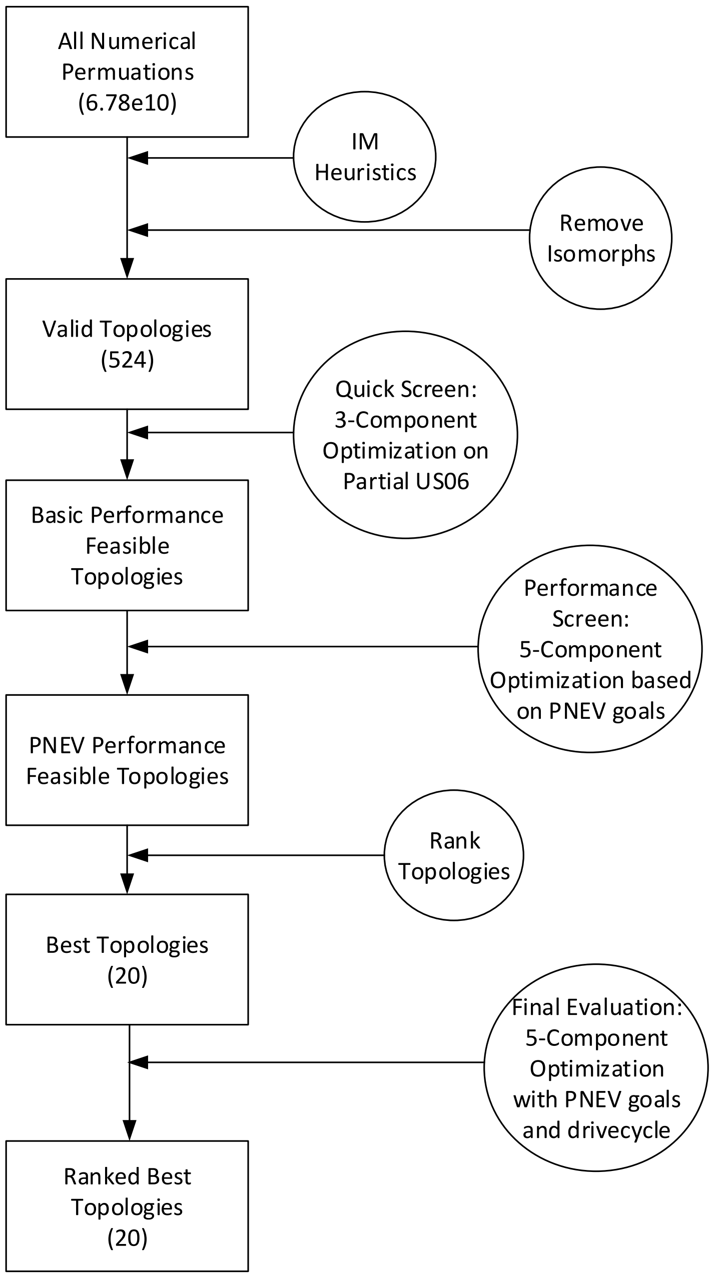
All Numerical Permuations (6.78e10)
IM Heuristics
Remove Isomorphs
Valid Topologies (524)
Quick Screen: 3-Component Optimization on Partial US06
Basic Performance Feasible Topologies
Performance Screen: 5-Component Optimization based on PNEV goals
PNEV Performance Feasible Topologies
Rank Topologies
Best Topologies (20)
Final Evaluation: 5-Component Optimization with PNEV goals and drivecycle
Ranked Best Topologies (20)

Figure 7.1: Flow Chart of the Topology Screening Process.

The 3 drivecycles used to evaluate the topologies for fuel consumption are shown in figure 7.2. The US06 drivecycle is found in the top graph. The shaded area is the section that was used to first screen the topologies. This highlighted area had the most difficult acceleration profile.

The middle graph in figure 7.2 is a hand-built drivecycle based on the PNGV goals. The guidelines descried in the PNGV goals were: accelerating from 0 to 96 km/h (60 mph) in 13.5s (rounded to 14s), maintaining a speed of 88 km/h (55 mph) at a 6.5 degree grade, and acceleration from 55 to 70 mph (88 to 112 km/h) in 8s. Additionally, a top speed of 130 km/h was desired. This drivecycle could potentially be changed to achieve any driving performance.

The bottom graph in figure 7.2 is a combination of PNGV goals (seen in middle graph), followed by the HWFET drive cycle. This drivecycle was created to optimize a performance drivetrain, while introducing average driving behavior.

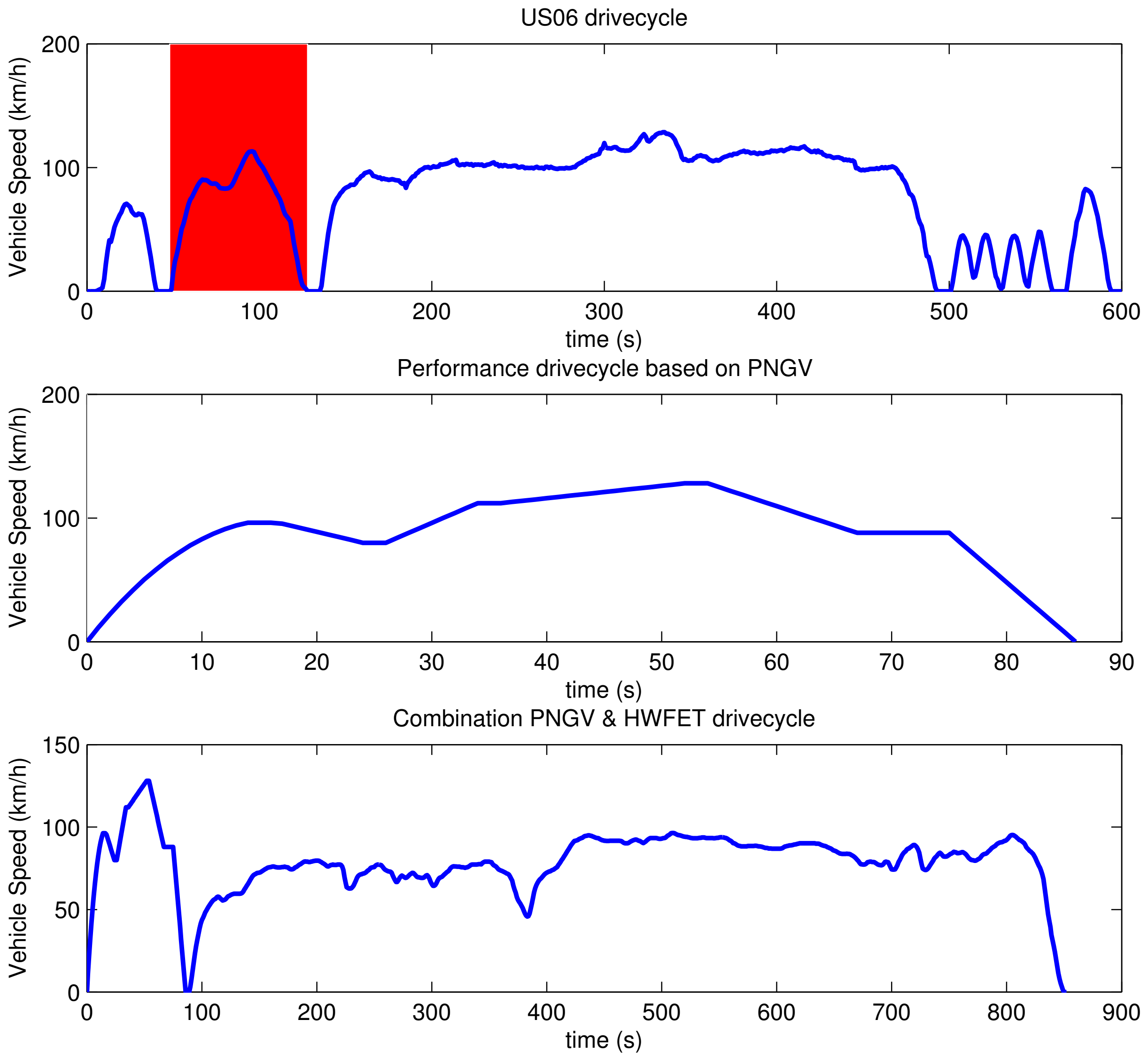

Figure 7.2: Drivecycles. Top: US06 drivecycle. Highlighted Area is Used to Screen Topologies. Middle: Drivecycle based on PNGV Design Guidelines. Bottom: Drivecycle based on PNGV Design Guidelines with HWFET.

Pattern search was used to optimize the component sizes. The settings for pattern search can be found in table 7.1.

| Setting | Value |
|---|---|
| Arguments (Scaling Factors) | [ICE, EM, GEN, GB, BAT] |
| Tolerance | 0.1 |
| Maximum Iterations | 100 |
| Upper Bound | [2,2,2,2,2] |
| Lower Bound (PNGV) | [0.5,0.5,0.5,0.5,0.5] |
| Lower Bound (PNGV/HWFET) | [0.25,0.25,0.25,0.25,0.25] |

Table 7.1: Pattern Search Settings

## 7.2 Topology Optimization Results for the PNGV goals

As stated in section 7.1, the component sizes were optimized for fuel consumption for a drivecycle based on the PNGV goals. This section will show the results from this optimization. Results for all of the topologies are summarized in the top graph seen in figure 7.3. On the x-axis are the topologies sorted by fuel consumption. On the y-axis is the fuel consumption. Shown in the top figure, the topologies with a fuel consumption over 2000 failed to meet either the dynamic or kinematic requirements of the drive cycle. Only 159 topologies could meet the constraints. It was found that 119 topologies have an (arbitrarily set target) fuel consumption score of 200 or less.

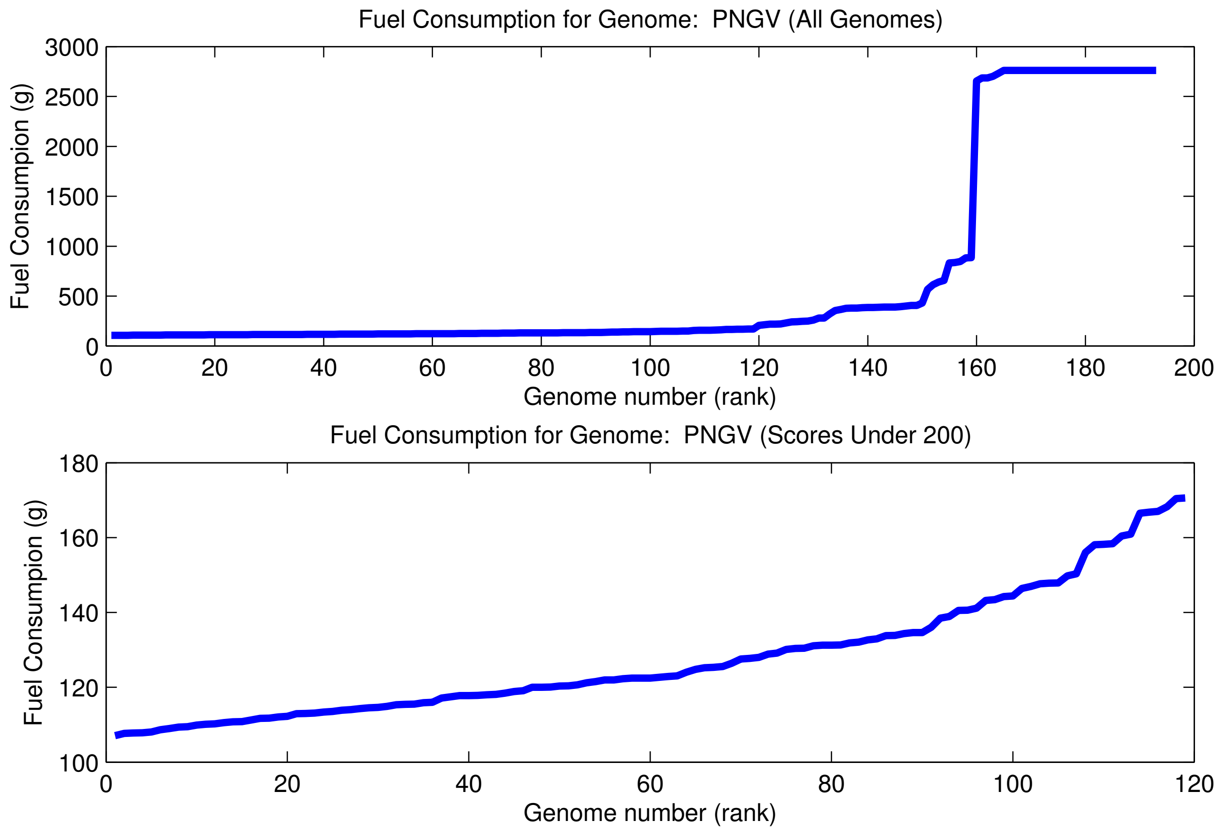

Figure 7.3: Results from the PNEV Screening.

Shown in figure 7.4 are the 20 topologies have the lowest fuel consumption for the PNGV drivecycle. Other notable topologies are shown in figure 7.5. Shown in table 7.2 are the topologies are named by their description. Their fuel consumption and component scaling factors are also shown.

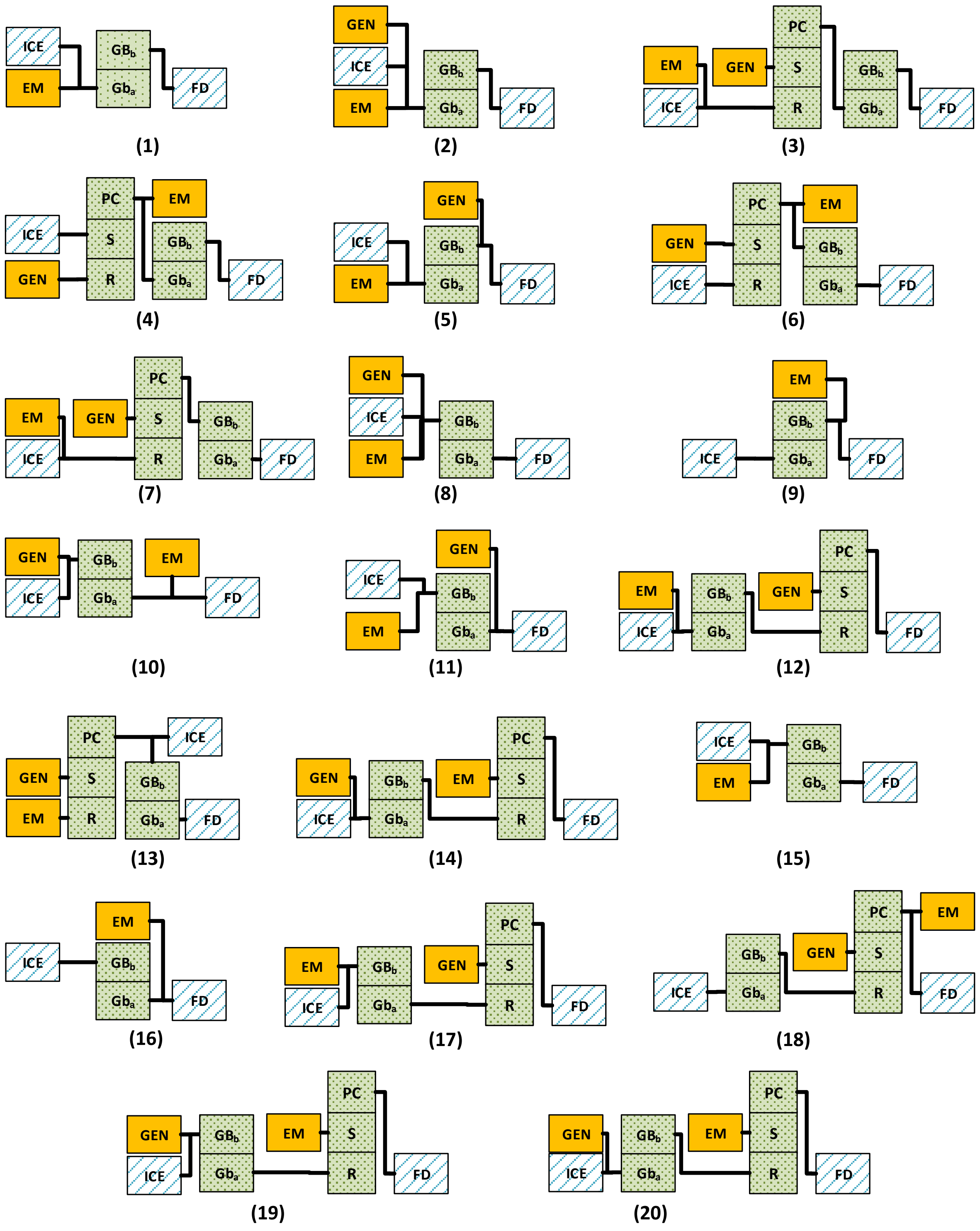

Figure 7.4: Top 20 Topologies Obtained from the PNGV drivecycle. Legend can be found in 7.5.

The topologies seen in figure 7.5 were included in the optimization for reference. Topology #33 is a Prius-like hybrid with an additional discrete gearbox between the planetary gear and the final drive. Topology #36 is similar to topology #33, but the discrete gearbox is put before both the electric motor and planetary gearset. Topology #71 is a Prius-like powertrain. Topology #117 is a Series powertrain. These topologies perform relatively well, but are not in the top 20.

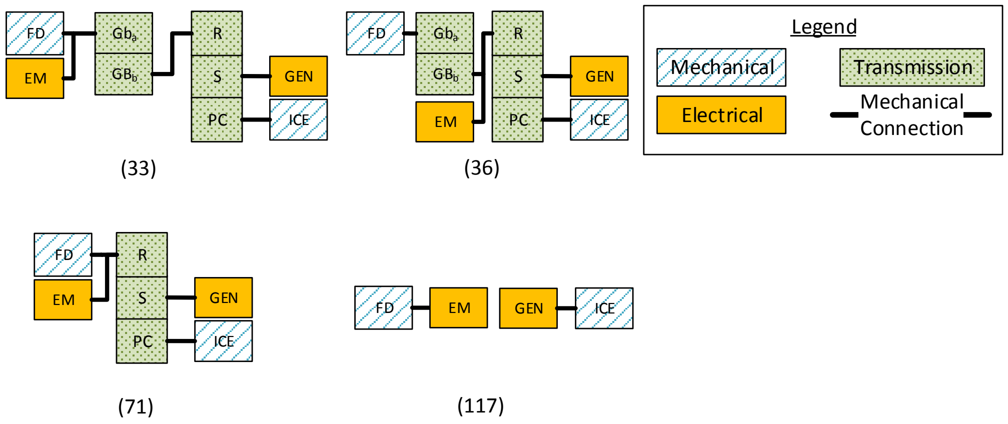

Figure 7.5: Notable Topologies.

The descriptions are loosely based off the naming convention used in Autonomie. Post/pre-Gearbox refers to the position of the powertrain component relative to the gearbox, as seen from the final drive. This comes in two forms: speed multiplying (M), and speed reducing (R). As the gearbox ratios are linearly scaled by a factor ranging from 0.5 to 2 (while maintaining a static weight of 75kg), either side can act as a speed multiplier or reducer, depending on the gear selected. All of the gearboxes that functioned as a speed multiplier had their gear ratios scaled to 0.5. This changed their baseline ratios of [3.32,2,1.36,1.01,0.82] to [1.66, 1.00, 0.68, 0.51, 0.41], or 'tuning for speed', rather than 'tuning for acceleration (torque)'. This is likely because the component sizes were optimized for a drivecycle with a high average speed.

The first 20 genomes have fuel consumptions ranging from 107.1g to 112.2g; a 4.8% difference. The curve-fits for the electric motor/generator have a $R^2$ value of at least 0.9551; therefore, there is a small error when evaluating the EM/GEN energy consumption and subsequently the battery State-Of-Charge. This small error may change the order of these topologies when a detailed simulation is performed.

In table 7.2 the topologies are shown sorted by fuel consumption. Component scaling ratios of ‘n/a’ denote topologies which do not include that particular component. It can be seen that for all topologies, the ICE was chosen to be scaled up. Of the shown topologies, not a single ICE was scaled down, and only the Series topology (genome 117) maintained a scaling ratio of 1 (57kW). As fuel consumption are generally lower for smaller engines, the scaling up of the ICE can only be explained by the difficult drivecycle; a smaller engine would not be able to provide the torque necessary to meet the acceleration profile.

The EM and GEN were generally scaled down, likely to reduce weight. As it is possible for both the EM and the GEN to be used as motors, they could potentially be used together to efficiently provide tractive force.

As all-electric range does not factor into this optimization, it is seen that all of the topologies have scaled down their battery to reduce weight. Electric-range could easily be incorporated by changing the initial and final values of the battery SOC. Dynamic Programming was used so the battery usage is optimized over the drivecycle and never reaches the battery minimum and maximum limits. Optimal controllers are unrealistic, unless one is driving the same route repeatedly, thus larger batteries provide the system with more flexibility and are much easier to implement.

Table 7.2: PNGV drivecycle results: Description of Top 20 Genomes Seen in Figure 7.4 and Notable Genomes in Figure 7.5.

| Genome # | Description | Component Scaling Ratios [ICE, EM, GEN, R, BAT] | FC (g) |
|---|---|---|---|
| 1 | Post-Gearbox (M) Parallel | [1.25,0.50,n/a,0.50,0.50] | 107.1 |
| 2 | Post-Gearbox (M) Complex | [1.25,0.50,0.50,0.50,0.50] | 107.7 |
| 3 | Post-Gearbox (M) with PC output and Parallel ICE-EM on Ring and GEN on Sun | [1.13,0.50,0.88,0.50,0.50] | 107.8 |
| 4 | Pre-Gearbox (M) EM, Powersplit with PC output, ICE on Sun and GEN on Ring | [1.13,0.50,0.75,0.50,0.50] | 107.8 |
| 5 | Complex with Pre-Gearbox GEN | [1.25,0.50,0.50,0.50,0.50] | 108.0 |
| 6 | Post-Gearbox (R) EM in Parallel with PC output, ICE on Ring, GEN on Sun | [1.13,0.50,0.75,0.75,0.50] | 109.4 |
| 7 | Post-Gearbox (R) with PC output, GEN on Sun and Parallel ICE-EM on Ring | [1.13,0.50,0.75,0.75,0.50] | 109.0 |
| 8 | Post-Gearbox (R) Complex | [1.50,0.63,0.50,0.50,0.50] | 109.4 |
| 9 | Parallel with Gearbox (M) on ICE | [1.25,0.75,n/a,0.50,0.50] | 109.5 |
| 10 | Complex with Pre-Gearbox (R) EM | [1.50,0.50,0.88,0.50,0.50] | 109.9 |
| 11 | Complex with Pre-Gearbox (R) GEN | [1.50,0.50,0.50,0.50,0.50] | 110.1 |
| 12 | PC output gear with GEN on SUN and Post-Gearbox ICE-EM on Ring | [1.13,0.50,0.88,0.75,0.50] | 110.3 |
| 13 | Post-Gearbox (R) ICE with PC output in Parallel, GEN on Sun, EM on Ring | [1.38,0.50,0.50,0.50,0.50] | 110.6 |
| 14 | PC output gear, with EM on Sun, Post-Gearbox (M) Parallel ICE-GEN on Ring | [1.13,0.50,0.88,0.50,0.50] | 110.8 |
| 15 | Post Gearbox (R) Parallel | [1.63,0.50,n/a,0.50,0.63] | 110.8 |
| 16 | Parallel Post-Gearbox (R) ICE | [1.50,0.75,n/a,0.50,0.63] | 111.3 |
| 17 | PC output, GEN on Sun, Post-Gearbox (R) Parallel ICE-EM | [1.13,0.50,0.88,0.50,0.50] | 111.7 |
| 18 | Parallel EM - PC output with GEN on Sun | [1.13,0.75,0.63,0.75,0.50] | 111.8 |
| 19 | PC output, EM on Sun, Post-Gearbox (R) Parallel ICE-GEN | [1.13,0.50,1.00,0.75,0.50] | 112.1 |
| 20 | PC output, EM on Sun, Post-Gearbox (M) Parallel ICE-GEN | [1.13,0.75,0.50,0.75,0.50] | 112.2 |
| 33 | Pre-Gearbox (R) EM Powersplit | [1.13,0.50,0.50,1.13,0.50] | 115.4 |
| 36 | Post-Gearbox (R) Powersplit | [1.13,0.75,0.75,1.25,0.50] | 116.0 |
| 71 | Powersplit | [1.50,1.00,1.38,n/a,0.50] | 127.7 |
| 117 | Series | [1.00,1.88,2.00,n/a,0.50] | 168.2 |

As it would be impractical to show the results for all topologies, only the results for the best topology is shown. The behaviour of Genome # 1 is shown in figure 7.6 and 7.7. Shown in figure 7.6, the ICE will only consume fuel in its most efficient operating region. Shown in figure 7.7 is the BFSC with the operating points overlay. As expected, the ICE is operating in its most efficient regions, or is off (providing 0 torque).

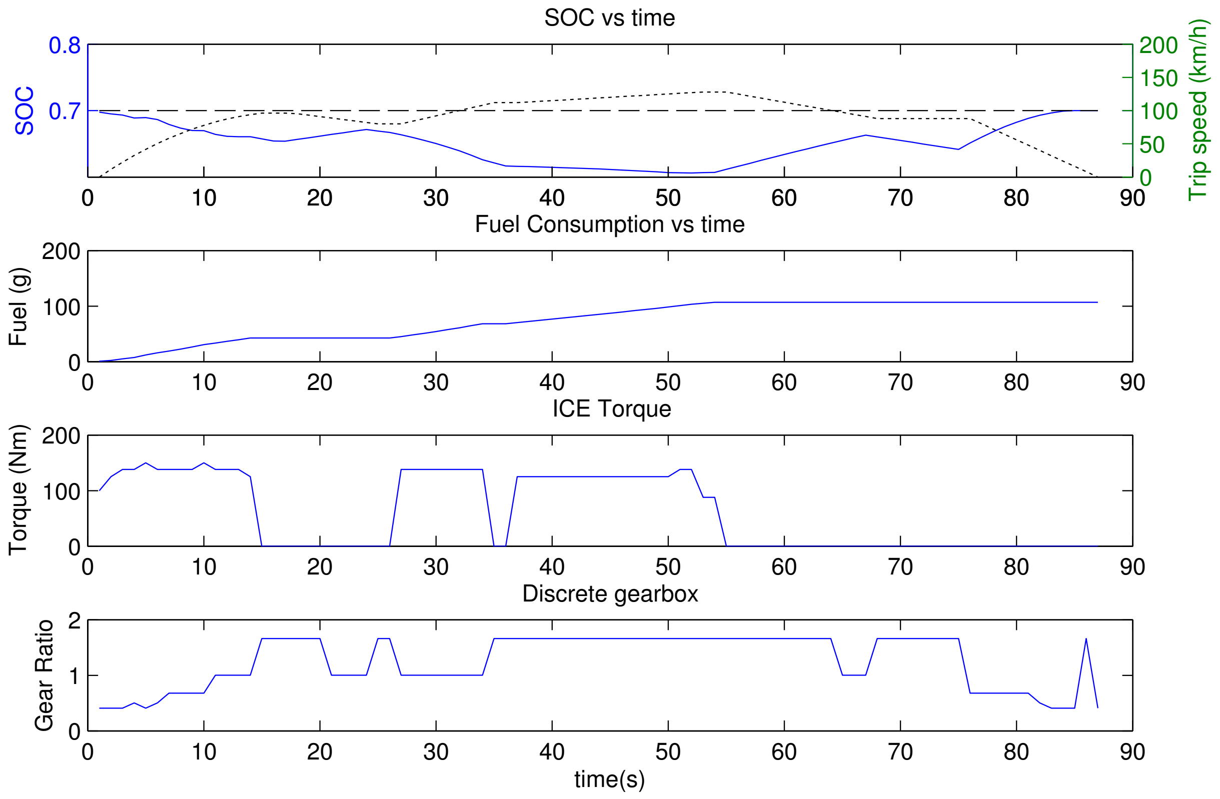

Figure 7.6: Dynamic programming results for Genome #1. Top: (left, solid line) battery State-Of-Charge vs time, (right, dotted line) vehicle velocity vs time. Second from top: cumulative fuel consumption vs time. Third from top: Internal Combustion Engine torque output. Bottom: discrete gearbox ratios.

Pattern search was used to find the optimal component sizes. Shown in figure 7.8, the median number of iterations required for convergence in a 5 dimension problem is 18. As there were 193 topologies, a histogram of the number of number of iterations and function calls is shown for groups of topologies rather than for the individual topologies. For each iteration, the function was called multiple times to sample the surrounding objective score

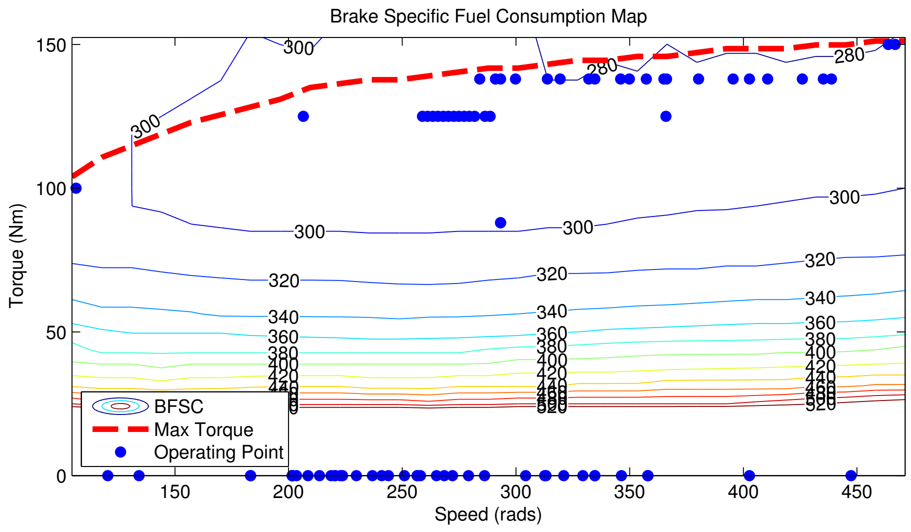

Figure 7.7: Brake Specfic Fuel Consumption Map with operating points for Genome #1 on the PNGV drivecycle.

and choose a new search direction. A few topologies had 4 rather than 5 components (i.e. Genome #1 does not have a GEN); they required fewer iterations to reach convergence.

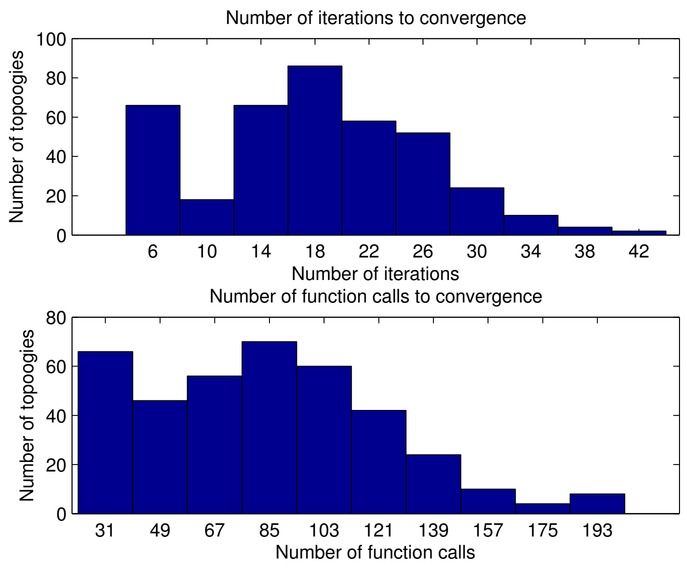

Figure 7.8: Optimization statistics for PNGV drivecycle.

## 7.3 Topology Optimization Results for the PNGV-HWFET drivecycle

The final stage of optimization combined the PNGV and HWFET drivecycles to create representative highway driving behaviour that must meet certain performance criteria. It is expected the optimization results may change because the combined drivecycle is longer with more regen-braking opportunites. The results for the optimization can be found in table 7.3. The genome number and naming were preserved from table 7.2, but sorted in ascending order by fuel consumption. The component scaling ratios and fuel consumption were updated for the drivecycle.

The most fuel efficient topology is genome #6, a Powersplit Hybrid with a discrete gearbox between the final drive and output gear of the planetary carrier.

Table 7.3: HWFET-PNGV drivecycle results: Description of Top 20 Genomes Seen in Figure 7.4 and Notable Genomes in Figure 7.5.

| Genome # | Description | Component Scaling Ratios [ICE, EM, GEN, R, BAT] | FC (g) |
|---|---|---|---|
| 6 | Post-Gearbox (R) EM in Parallel with PC output, ICE on Ring, GEN on Sun | [0.88,0.50,0.75,0.50,0.50] | 614.0 |
| 15 | Post Gearbox (R) Parallel | [1.25,0.75,n/a,0.50,0.38] | 617.0 |
| 17 | PC output, GEN on Sun, Post-Gearbox (R) Parallel ICE-EM | [1.00,0.50,0.88,0.50,0.38] | 618.4 |
| 8 | Post-Gearbox (R) Complex | [1.25,0.50,0.38,0.50,0.38] | 619.0 |
| 11 | Complex with Pre-Gearbox (R) GEN | [1.25,0.50,0.38,0.50,0.38] | 619.9 |
| 33 | Pre-Gearbox (R) EM Powersplit | [1.00,0.75,0.50,0.75,0.38] | 620.4 |
| 5 | Complex with Pre-Gearbox GEN | [1.13,0.50,0.25,0.50,0.38] | 620.8 |
| 1 | Post-Gearbox (M) Parallel | [1.25,0.50,n/a,0.50,0.38] | 623.4 |
| 16 | Parallel Post-Gearbox (R) ICE | [1.25,0.88,n/a,0.50,0.50] | 624.3 |
| 2 | Post-Gearbox (M) Complex | [1.00,0.50,0.50,0.50,0.63] | 626.5 |
| 19 | PC output, EM on Sun, Post-Gearbox (R) Parallel ICE-GEN | [1.00,0.50,1.00,0.50,0.25] | 628.5 |
| 9 | Parallel with Gearbox (M) on ICE | [1.00,0.75,n/a,0.50,0.63] | 630.8 |
| 4 | Pre-Gearbox (M) EM, Powersplit with PC output, ICE on Sun and GEN on Ring | [0.88,0.50,0.63,0.50,0.38] | 630.9 |
| 13 | Post-Gearbox (R) ICE with PC output in Parallel, GEN on Sun, EM on Ring | [1.13,0.50,0.38,0.50,0.50] | 634.5 |
| 36 | Post-Gearbox (R) Powersplit | [1.00,1.00,0.75,0.75,0.50] | 635.2 |
| 3 | Post-Gearbox (M) with PC output and Parallel ICE-EM on Ring and GEN on Sun | [1.00,0.25,1.00,0.50,0.25] | 637.3 |
| 10 | Complex with Pre-Gearbox (R) EM | [1.13,0.50,0.25,0.75,0.50] | 639.1 |
| 12 | PC output gear with GEN on SUN and Post-Gearbox ICE-EM on Ring | [1.00,0.25,0.88,0.50,0.25] | 640.7 |
| 14 | PC output gear, with EM on Sun, Post-Gearbox (M) Parallel ICE-GEN on Ring | [1.00,0.50,0.75,0.50,0.25] | 641.5 |
| 20 | PC output, EM on Sun, Post-Gearbox (M) Parallel ICE-GEN | [0.75,0.75,0.50,0.50,0.75] | 643.0 |
| 18 | Parallel EM - PC output with GEN on Sun | [0.88,0.75,0.50,0.50,0.50] | 646.9 |
| 7 | Post-Gearbox (R) with PC output, GEN on Sun and Parallel ICE-EM on Ring | [1.00,0.50,0.50,1.00,0.38] | 654.3 |
| 71 | Powersplit | [1.25,1.00,1.63,n/a,1.00] | 679.5 |
| 117 | Series | [1.00,2.00,2.00,n/a,0.38] | 774.1 |

The optimization results for the PNGV-HWFET drivecycle is shown in figure 7.9. The number of iterations and function calls are shown for each of the top 20 genomes individually. All of the optimizations terminated when the step size was less than the tolerance step size of 0.1.

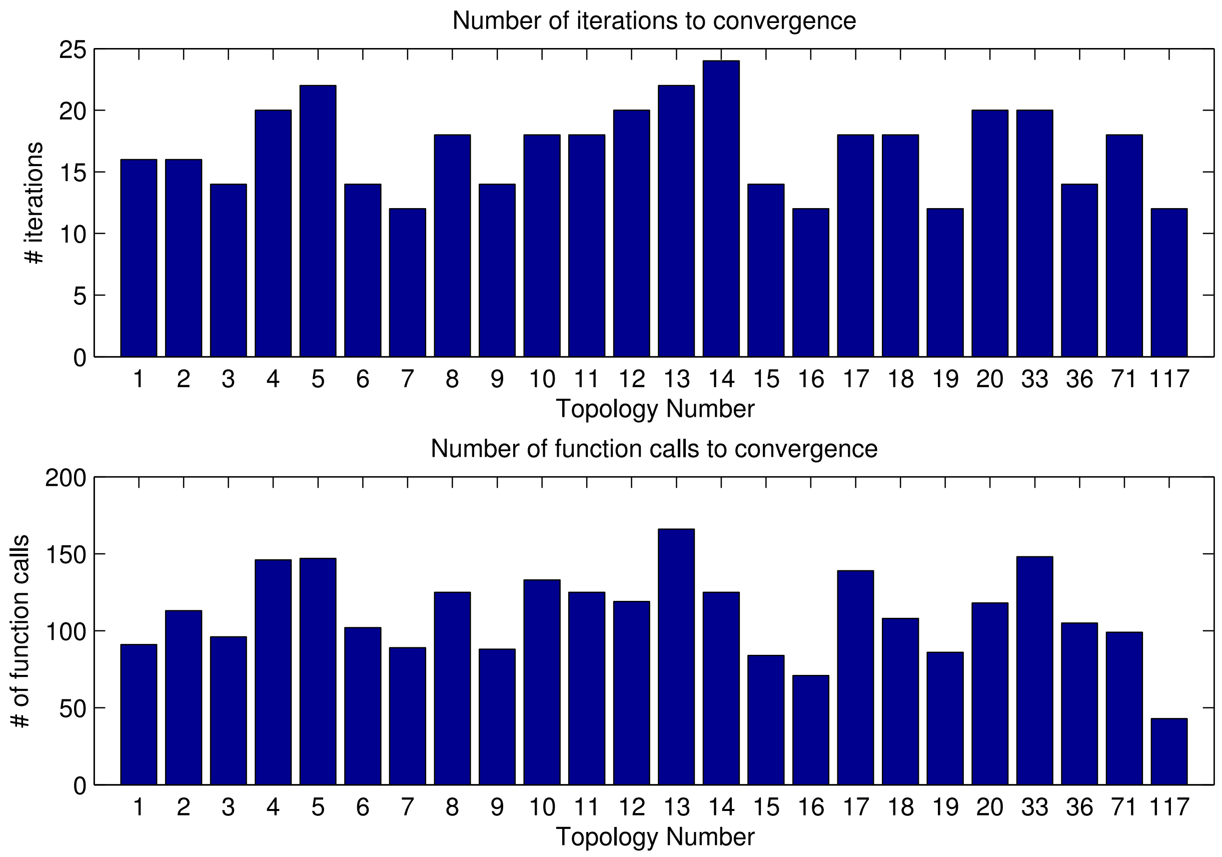

Figure 7.9: Optimization statistics for PNGV-HWFET drivecycle. The numbering for the topologies correspond to the genome number in table 7.2.

## 7.4 Discussion and Interpretation of Results

The topologies rankings are important, but it is more important to understand the trends that make these powertrains efficient. Of the top 24 evaluated topologies, there were 4 Parallel-like, 1 Series-like, 5 Complex-like, 7 Complex-Powersplit-like, and 7 Powersplit-like. Topologies #1, 9, 15, 16 are Parallel-like, because they have an ICE mechanically connected in parallel with an electric motor and the final drive. Topology # 117 is the reference Series hybrid powertrain and is the only Series-like topology that was evaluated. Topologies #2, 5, 8, 10, 11 are Complex-like. Complex-like topologies are defined as 3 movers mechanically connected to a transmission (without a powersplit device). Topologies #4, 6, 13, 18, 33, 36, 71 are Powersplit-like, defined as one mover (ICE, EM, GEN) per gear (ring, sun, planetary carrier) on the PSD. Topologies #3, 7, 12, 14, 17, 19, 20 are Complex-Powersplit-like, as defined by 3 movers, with more than one mover per gear on the PSD.

Long, complex geartrains suffered no penalties because mechanical losses were ignored. This skewed the results towards mechanically coupled powertrains, such as Parallel-like, Complex-Powersplit-like and Powersplit-like topologies. All possible Parallel-like topologies were in top 20 architectures. Complex-like and Powersplit-Complex-like topologies together made the majority of the genomes, however they have more components and thus have more possible configurations. Electrically coupled hybrids (such as Series hybrids) performed comparatively worse because electrical inefficiencies were included. Shown in table 7.2 and 7.3, the Series Hybrid performed the worst of all 24 evaluated topologies. Powersplit-like and Complex-Powersplit-like topologies could operate as a Parallel hybrid or Series hybrid. The extra modes of operation make these topologies perform more efficiently; however, to realistically build a configuration such as #3, additional clutches and more complex controllers would be required.

It is interesting to note that the Series (#117) and Powersplit (#71) powertrains did not make the top 20 topologies for the PNGV screening stage. This is likely due to the optimization objective function being a function of fuel consumption and does not include vehicle cost or controller complexity. Powertrain connectors, such as intermediate gears and driveshafts, were also not factored into this optimization. Additional constraints that could be included are: packaging, intellectual property, vehicle type, development time, safety and chassis weight.

All of the top 20 topologies contained the discrete gearbox. Therefore, the extra 75 kg of gearbox mass is always offset by improved fuel efficiency. The gearboxes were assumed to be perfectly efficient. In reality they have a dynamic efficiency ranging from 95-97%.

Furthermore, transmission efficiency decreases as the reduction ratio increases, with very high reduction ratios being as low as 75-80% efficient [18]. Introducing mechanical inefficiencies into the powertrain model may yield results with fewer gearboxes. For the purpose of conceptual design, these are all topologies which should be considered.

Although the FD/ICE/EM/GEN could realistically be placed on the ring/sun/planet carrier gear in any combination [18], it was found that the planetary carrier was favoured as the output gear. The Chevrolet Volt is an example of a commercially available HEV that uses the planetary carrier as the output gear (albeit in combination with a second planetary gear). Generally the GEN should be placed on the sun gear because it runs at the highest speed and lowest torque, and thus a smaller generator with a high speed can be used, which takes up less space and weight [18].

## 7.5 Summary

The results for the Hybrid Electric Vehicle powertrain screening and optimization were presented in this chapter. Parallel-like, Powersplit-like, and Complex-like topologies were found to be the most efficient. Mechanical inefficiencies were ignored in the models, so electrically coupled topologies, such as a Series powertrain, were comparatively less efficient.

# Chapter 8

# Conclusion

An algorithm was developed to assemble and compare all possible configurations of powertrain components. Combinatorics was used to discover all possible combinations of: an internal combustion engine, high-torque low-speed electric motor, low-torque high-speed electric motor, planetary gearset, and five-speed discrete gearbox. The Graph Theoretic Method was used to generate the powertrain models.

The powertrain models were comprised of steady-state equations in symbolic form. An optimal control strategy is required to fairly compare the different topologies because a powertrain control strategy is dependant on the configuration. Dynamic Programming was used to determine the control law that minimizes the energy consumption for a given drivecycle. Evaluating every possible topology would take an extremely long time, so topologies were evaluated using a multi-stage screening process.

The first stage examined the structure of the powertrain and used heuristics to eliminate infeasible topologies; 512 topologies were feasible.

The second stage eliminated topologies that could not meet basic driving performance; 193 topologies were feasible. Basic driving performance was tested using a section of the US06 drivecycle. The sizes of three components was optimized to ensure the topology is feasible independent of the size of the components.

The third stage eliminated topologies which could not achieve driving performance design goals; 159 could achieve the performance requirements, but only 119 were reasonably fuel efficient. The driving performance goals were implemented with a drivecycle based on the Partnership for a New Generation of Vehicles (PNGV) goals. The sizes for five components were optimized at this stage.

The 20 most fuel efficient powertrains were selected for further evaluation. Additionally, 4 common powertrains were evaluated for reference. The size of the components were optimized for a combination of the PNGV drivecycle and the HWFET drivecycle.

The most fuel efficient topology was found to be a Powersplit hybrid which has a discrete gearbox between the final drive and the powersplit device. The electric motor, planetary carrier gear, and gearbox were connected in parallel. It was found that Parallel-like, Powersplit-like, and Complex-like topologies were were the most efficient powertrain configurations. Powertrains containing two gearboxes were more efficient because the geartrain models ignored mechanical inefficiencies.

## 8.1 Contributions

The main contributions in this thesis are: (i) the creation of a framework to automate the design of hybrid electric vehicle powertrains, (ii) heuristics to quickly eliminate invalid topologies, (iii) a process to speed up the evaluation of a steady-state model using parallel computing, and (iv) novel hybrid electric vehicle topologies.

## 8.2 Lessons Learned

#### MapleSim Implementation

As MapleSim uses the GTM to formulate equations, attempts were made to exploit the built in solver, then export a model from MapleSim to Simulink. This approach used a ‘control switchbox’ custom component, which accepted an adjacency matrix as an input parameter. The model components were connected to this control switch custom component. The component interconnections be manipulated by changing the values of the adjacency matrix. Using this approach, a Series hybrid was successfully into a Parallel Hybrid.

This approach was abandoned when it was discovered that the ‘output model to Simulink’ routine would create a functional model (written in C) with a fixed structure. A fixed structure model is not appropriate when changing topologies as the code needs to be reordered or flexible.

It was attempted to pass in parameters and execute the MapleSim model externally. Unfortunately the only way to externally run the simulation is to compile the model and call it from Maple. Compiling the model will fix the equation structure, rendering it useless for topology analysis.

### 8.2.1 Maple Toolbox Implementation

When solving a system in terms of its symbolic variables, *eliminate()* is more appropriate function and executes much faster than *solve()*. While repeatedly solving systems of solutions, Maple does not perform garbage collection effectively until *restart()* is called. Memory would allocate in the kernel until disk swapping caused major system slowdown or a crash occurred. A workaround was created by saving all of the necessary variables to disk, calling *restart()* periodically, then reloading the variables from disk.

### 8.2.2 Curve-fitting Toolbox

A higher $R^2$ value does not always mean the fit is good. Always check the extrema of your polynomials, otherwise unexpected values could be obtained. A small algorithm was created to double-check that the error between the curve-fit and the map data was not great.

### 8.2.3 Genetic Algorithm

The design of hybrid powertrains could potentially be automated using this framework if more computing power was available. The design space can be expanded by increasing the number of components and the number of nodes allowed in the incidence matrix. Potential components could be: super-capacitors, fuel cells, additional planetary gears, flywheels, and clutches. Additionally the powertrain design framework could easily be expanded to heavy trucks, trains, planes and boats by changing the base vehicle parameters and the drivecycle.

Expanding the design space has the disadvantage of exponentially increasing the number of possible combinations. Liu [9] used an approach which involved describing planar mechanisms with a genome. In Liu's work, the design space was much larger and enumeration was not practical. Instead of enumeration, the genetic algorithm was applied to the genome describing each topology resulting in a heuristic search. A disadvantage to this method is large number of generations (5000+) required for convergence, and is thus is only suitable for simulations that complete quickly.

## 8.3 Future Work

### 8.3.1 Mechanical Inefficiencies

Ignoring mechanical inefficiencies skewed the topology optimization to the Parallel-like powertrains rather than Series-like topologies. For a more realistic balance of topologies these inefficiencies should be included. By examining the incidence matrix, it could be determined which end of the gearbox was facing the final drive. A damper variable could be included to introduce power losses. The power loss would be a function of the output torque. For example, in the case of a simple ICE vehicle, the current equations (only torque equations shown for brevity):

$$\begin{aligned} \tau_{FD} &= \tau_{GB_a} \\ \tau_{GBb} &= \frac{-1}{R_{GB}(i_g)}\tau_{GBa} \\ \tau_{GBb} &= \tau_{ICE} \end{aligned} \tag{8.1}$$

would be replaced with the equations:

$$\begin{aligned} \tau_{FD} &= \tau_{GB_a} + \tau_{GBaloss} \\ \tau_{GBb} &= \frac{-1}{R_{GB}(i_g)}\tau_{GBa} \\ \tau_{GBb} &= \tau_{ICE} \\ \tau_{GBaloss} &= \tau_{GBa}\begin{cases} \tau_{GBa} * \omega_{GBa} > 0 & GBLoss \\ else & -GBLoss \end{cases} \end{aligned} \tag{8.2}$$

where GBLoss would be the power loss, ranging from 1-5%. It was found that this method was effective for some topologies, but caused *eliminate()* to cease working. Further analysis is required to ensure this is a robust solution.

### 8.3.2 Genetic Algorithm

As previously mentioned, adding more components exponentially expand the design space. For a certain large number of components, it may be feasible to use the genetic algorithm

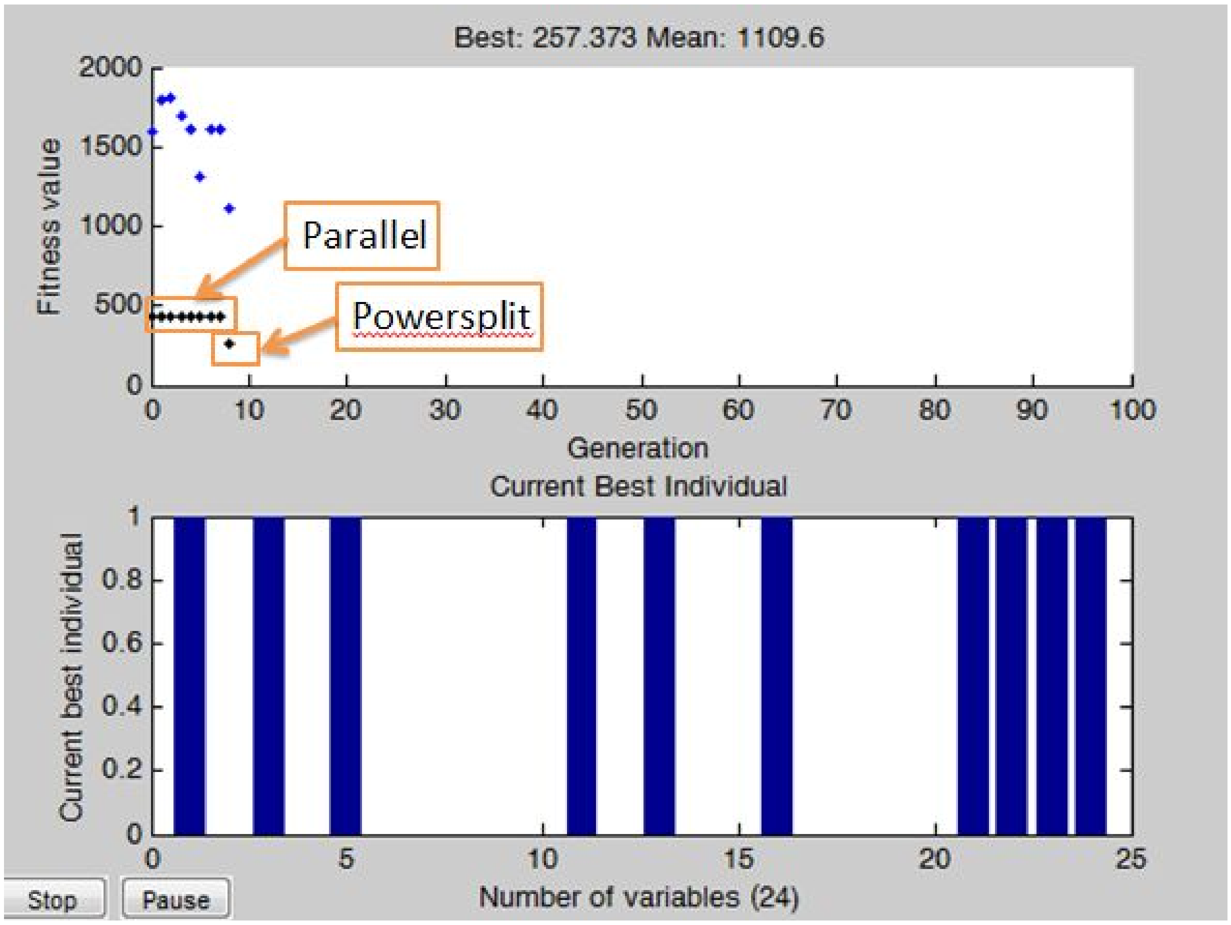

Figure 8.1: Genetic Algorithm Topology Search Results

to find the best topology. The original goal of this work was to use the genetic algorithm, but it was found to be no better than random number generation for this application.

Shown in figure 8.1, are results from earlier work. The top graph shows the results of the genetic algorithm search as a function of generation. It can be seen that the Parallel and Powersplit architectures were found after 9 generations (population size of 50), or 450 simulation calls. Unfortunately the Series Hybrid topology was not found, so it was decided that the usefulness of this approach is limited. It is likely that this approach will become more useful as the design space increases, or as simulation time decreases. The bottom graph is a genome.

This concludes the thesis. Thank you very much for taking the time to read this. I really hope you learned something. I have no more knowledge to impart on you, but I will leave you with a lesson. In the words of George E. P. Box, *“essentially, all models are wrong, but some are useful”*.

# References

[1] K. Ogata, *Modern Control Engineering.* New Jersey: Prentice Hall, 1997.

[2] W. Boruzky, *Bond Graph Methodology.* London: Springer, 2010.

[3] P. Barrade and A. Bouscayrol, "Energetic Macroscopic Representation - An Energy-Flow Based Methodology dedicated for the control of multiphysics systems." 2011.

[4] R. Zanasi, "Power-oriented graphs for modeling electrical machines," in *Electrotechnical Conference, 1996. MELECON '96., 8th Mediterranean (Volume:3 )*, no. 39, (Bari), pp. 1211–1214, IEEE, 1996.

[5] ANL, "Autonomie," 2012.

[6] K. Chen, A. Bouscayrol, A. Berthon, P. Delarue, D. Hissel, and R. Trigui, "Global modeling of different vehicles," *IEEE Vehicular Technology Magazine*, vol. 4, 2009.

[7] I. K. Luigi del Re, Frank Allgower , Glielmo Luigi, Carlos Guardiola, *Automotive Model Predictive Control.* Springer, 2010.

[8] C. C. Chan, "The State of the Art of Electric, Hybrid, and Fuel Cell Vehicles," *Proceedings of the IEEE*, vol. 95, pp. 704–718, Apr. 2007.

[9] Y. Liu, *Automated Type and Dimensional Synthesis of Planar Mechanisms Using Numerical Optimization with Genetic Algorithms.* Phd thesis, University of Waterloo, 2004.

[10] J. J. McPhee, "On the use of linear graph theory in multibody system dynamics," 1996.

[11] J. J. Mcphee, "Dynamics of Multibody Systems : Conventional and Graph-Theoretic Approaches." 2004.

[12] J. J. McPhee, "Unified Modelling Theories for the Dynamics of Multidisciplinary Multibody Systems," in *Advances in Computational Multibody Systems* (J. A. Ambrosio, ed.), ch. Unified Mo, pp. 125–154, Lisbon, Portugal: Springer, 2 ed., 2005.

[13] R. Wang and S. M. Lukic, "Dynamic programming technique in hybrid electric vehicle optimization," *2012 IEEE International Electric Vehicle Conference*, pp. 1–8, 2012.

[14] D. S. Naidu, *Optimal Control Systems*. 2003.

[15] M. Neuman, H. Sandberg, and B. Wahlberg, "Rule-Based Control of Series HEVs Derived from Deterministic Dynamic Programming.".

[16] M. P. O. Keefe and T. Markel, "Dynamic Programming Applied to Investigate Energy Management Strategies for a Plug-in HEV," No. November, 2006.

[17] S. Chanda, *POWERTRAIN SIZING AND ENERGY USAGE ADAPTATION STRATEGY*. PhD thesis, University of Akron, 2008.

[18] M. Ehsani, Y. Gao, and A. Emadi, *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles*. CRC Press, second edi ed., 2010.

[19] M. Chehresaz, "Modeling and Design Optimization of Plug-In Hybrid Electric Vehicle Powertrains," 2013.

[20] O. Bitsche and G. Gutmann, "Systems for hybrid cars," *Journal of Power Sources*, vol. 127, pp. 8–15, Mar. 2004.

[21] E. Karden, S. Ploumen, B. Fricke, T. Miller, and K. Snyder, "Energy storage devices for future hybrid electric vehicles," *Journal of Power Sources*, vol. 168, pp. 2–11, May 2007.

[22] A. E. Bayrak, Y. Ren, and P. Y. Papalambros, "Design of Hybrid-Electric Vehicle Architectures Using Auto-Generation of Feasible Driving Modes," in *Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 1–9, 2013.

[23] A. Kawahashi and Toyota Motor Corporation, "New-Generation Hybrid Electric Vehicle and Its Supporting Power Semiconductor Devices," pp. 23–29, 2004.

[24] J. M. Miller and M. Everett, "An Assessment of Ultra-capacitors as the Power Cache in Toyota THS-11 , GM-Allision AHS-2 and Ford FHS Hybrid Propulsion Systems," *IEEE*.

[25] C. Chan, A. Bouscayrol, and K. Chen, "Electric, Hybrid, and Fuel-Cell Vehicles: Architectures and Modeling," *IEEE Transactions on Vehicular Technology*, vol. 59, 2010.

[26] Maplesoft, "MapleSim," 2014.

[27] Modelica Association, "Modellica."

[28] The MathWorks Inc, "Simscape," 2014.

[29] K. J. Astrom and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2012.

[30] J. Eborn, "Bond Graph Modelling," 1960.

[31] University Lille1, "Energetic Macroscopic Representation."

[32] K. Chen, A. Bouscaryrol, and W. Lhomme, "Energetic Macroscopic Representation and Inversion-based Control: Application to an Electric Vehicle with an Electric Differenetial," *Journal of Asian Electric Vehicles*, vol. 6, no. 1, pp. 1097–1102, 2008.

[33] K. Chen, A. Bouscayrol, A. Berthon, P. Delarue, D. Hissel, and R. Trigui, "Global modeling of different vehicles using Energetic Macroscopic Representation," *2008 IEEE Vehicle Power and Propulsion Conference*, 2008.

[34] R. Zanasi, "Dynamics of a n-links manipulator by using power-oriented graphs," in *Symposium on Robot Control-SYROCO*, pp. 535—-542, 1994.

[35] R. Morselli and R. Zanasi, "Modeling of Automotive Control Systems Using Power Oriented Graphs," *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*, pp. 5295–5300, Nov. 2006.

[36] R. Zanasi, A. Viscontit, G. Sandoni, and R. Morselli, "Dynamic Modeling and Control of a Car Transmission System," no. July, pp. 416–421, 2001.

[37] R. Zanasi and F. Grossi, "The POG technique for modeling planetary gears and hybrid automotive systems," *2009 IEEE Vehicle Power and Propulsion Conference*, pp. 1301–1307, Sept. 2009.

[38] K. W. Morency, *Automatic Generation of Real-Time Simulation Code for Vehicle Dynamics using Linear Graph Theory and Symbolic Computing by*. PhD thesis, University of Waterloo, 2007.

[39] G. Savage, "Robust Design Coursenotes," 2013.

[40] T.-S. Dao and J. McPhee, "Dynamic modeling of electrochemical systems using linear graph theory," *Journal of Power Sources*, vol. 196, pp. 10442–10454, Dec. 2011.

[41] B. Bollbas, *Modern Graph Theory.* Springer New York, 1998.

[42] J. S. G. Paganell, S. Delprat, T.M. Guerra, J. Rimaux, "Equivalent Consumption Minimization Strategy For Parallel Hybrid Powertrains," *IEEE VTC*, vol. 4, pp. 2076–2081, 2002.

[43] J. Liu and H. Peng, "A systematic design approach for two planetary gear split," *Vehicle System Dynamics*, vol. 48, no. 11, pp. 1395–1412, 2010.

[44] X. Ma, Y. Zhang, and C. Yin, "Kinematic Study and Mode Analysis of a New 2-Mode Hybrid Transmission," *Proceedings of the FISITA 2012 World Automotive Congress*, vol. 193, 2013.

[45] X. Zhang, C.-t. Li, D. Kum, and H. Peng, "Prius + and Volt : Configuration Analysis of Power-Split Hybrid Vehicles With a Single Planetary Gear," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 8, pp. 3544–3552, 2012.

[46] T. Hofman, S. r. Ebbesen, and L. Guzzella, "Topology Optimization for Hybrid Electric Vehicles With Automated Transmissions," *IEEE Transactions on Vehicular Technology*, vol. 61, pp. 2442–2451, July 2012.

[47] A. S. Lino Guzzella, *Vehicle Propulsion Systems.* 2013.

[48] J. A. P. L. Rodrigo Garvia-Valle, *Electric Vehicle Integration into Modern Power Networks.* 2013.

[49] F. Irani, *On Dynamic Programming Technique Applied to a Parallel Hybrid Electric Vehicle.* PhD thesis, 2009.

[50] R. A. Mcdonald, "Electric Motor Modeling for Conceptual Aircraft Design.".

[51] W. N. Harald Naunheimer, Bernd Bertsche, Joachim Ryborz, *Automotive Transmissions.* Springer Berlin Heidelberg, second edi ed., 1994.

[52] R. L. Norton, *Design of Machinery: An Introduction to the Synthesis and Analysis of Mechanisms and Machines.* New York: McGraw-Hill, second edi ed., 1992.

[53] J. Gomez, R. Nelson, E. E. Kalu, M. H. Weatherspoon, and J. P. Zheng, "Equivalent circuit model parameters of a high-power Li-ion battery: Thermal and state of charge effects," *Journal of Power Sources*, vol. 196, pp. 4826–4831, May 2011.

[54] J. Gomez, R. Nelson, E. E. Kalu, M. H. Weatherspoon, and J. P. Zheng, "Corrigendum to Equivalent circuit model parameters of a high-power Li-ion battery: Thermal and state of charge effects [J. Power Sources 196 (10) (2011) 48264831]," *Journal of Power Sources*, vol. 218, p. 5, Nov. 2012.

[55] V. Johnson, "Battery performance models in ADVISOR," *Journal of Power Sources*, vol. 110, pp. 321–329, Aug. 2002.

[56] A. Seaman, T.-S. Dao, and J. McPhee, "A survey of mathematics-based equivalent-circuit and electrochemical battery models for hybrid and electric vehicle simulation," *Journal of Power Sources*, vol. 256, pp. 410–423, June 2014.

[57] M. Doyle, T. F. Fuller, and J. Newman, "Modeling of Galvanostatic Charge and Discharge of the Lithium / Polymer / Insertion Cell," *J. Electrochem. Soc*, vol. 140, no. 6, pp. 1526–1533, 1993.

[58] J. Newman and W. Tiedemann, "Porous-electrode theory with battery applications," *AIChE Journal*, vol. 21, pp. 25–41, Jan. 1975.

[59] J. N. Marc Doyle, "Doyle - The Use of Mathematical Modeling in the Design of LithiumPolymer Battery Systems," *Electrochimica Acta*, vol. 40, no. 13, pp. 2191–2196, 1995.

[60] V. R. Subramanian, V. D. Diwakar, and D. Tapriyal, "Efficient Macro-Micro Scale Coupled Modeling of Batteries," *Journal of The Electrochemical Society*, vol. 152, no. 10, p. A2002, 2005.

[61] T.-S. Dao, C. P. Vyasarayani, and J. McPhee, "Simplification and order reduction of lithium-ion battery model based on porous-electrode theory," *Journal of Power Sources*, vol. 198, pp. 329–337, Jan. 2012.

[62] G. Andrews, "Dynamics Using Vector-Network Techniques," tech. rep., Mechanical Engineering, University of Waterloo, Waterloo, 1977.

[63] A. Sciarretta, M. Back, and L. Guzzella, "Optimal Control of Parallel Hybrid Electric Vehicles," *IEEE Transactions on Control Systems Technology*, vol. 12, pp. 352–363, May 2004.

[64] Z. Yuan, L. Teng, S. Fengchun, and H. Peng, "Comparative Study of Dynamic Programming and Pontryagins Minimum Principle on Energy Management for a Parallel Hybrid Electric Vehicle," *Energies*, vol. 6, pp. 2305–2318, Apr. 2013.

[65] Matlab Central File Exchange, "LERP: fast n-dimensional linear interpolation & extrapolation."

[66] A. Rahmoun and H. Biechl, "Modelling of Li-ion batteries using equivalent circuit diagrams," *PRZEGLAD ELEKTROTECHNICZNY*, vol. 2, pp. 152–156, 2012.

[67] Wikipedia, "Internal Combustion Engine."

[68] Schneider Electric, "How big are Power line losses?."

[69] Mathworks, "Global Optimization Toolbox," 2014.

[70] V. Torczon, "On the Convergence of Pattern Search Algorithms," *SIAM Journal of Optimization*, vol. 7, no. 1, pp. 1–25, 1997.

[71] C. Audet and J. E. Dennis, "Analysis of generalized pattern searches," *SIAM Journal of Optimization*, vol. 13, no. 3, pp. 889–903, 2003.

[72] K. Ahn, S. Cho, and S. W. Cha, "Optimal operation of the power-split hybrid electric vehicle powertrain," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 222, pp. 789–800, May 2008.

[73] X. Ai, "An Electro-Mechanical Infinitely Variable," vol. 2005, no. 724, 2014.

[74] N. L. Biggs, E. K. Lloyd, and R. J. Wilson, *Graph Theory 1736-1936*. 1999.

[75] S. Golbuff, *Optimization of A Plug-In Hybrid Electric Vehicle*. PhD thesis, Georgia Institute of technology, 2006.

[76] H. G. H. Herman E. Koenig, Yilmaz Tokad, Hiremaglur K. Kesavan, *Analysis of Discrete Physical Systems*. McGraw-Hill, 1967.

[77] A. Ing, "Public Acceptance of Electric Vehicles in Toronto," in *International Society for the System Sciences*, pp. 1–12, 2011.

[78] T. Katrašnik, "Analytical framework for analyzing the energy conversion efficiency of different hybrid electric vehicle topologies," *Energy Conversion and Management*, vol. 50, pp. 1924–1938, Aug. 2009.

[79] T. Katrašnik, "Analytical framework for analyzing the energy conversion efficiency of different hybrid electric vehicle topologies," *Energy Conversion & Management*, vol. 50, pp. 1924–1938, 2009.

[80] J. Nocedal, S. J. Wright, and S. M. Robinson, *Numerical Optimization*.

[81] T. L. S. Robert G. Busacker, *Finite Graphs and Networks: An Introduction with Applications*. McGraw-Hill, 1965.

[82] T. Solutions, "Hybrid and Electric Vehicle Solutions Guide," 2013.

[83] M. B. R. Sundram Seshu, *Linear Graphs and Electrical Networks*. Addison-Wesley Publishing Company, 1961.

[84] E. Wilhelm, J. Hofer, W. Schenler, and L. Guzzella, "Optimal implementation of lightweighting and powertrain effi ciency technology in passengers' vehicles," *Transport*, vol. 27, pp. 237–249, Sept. 2012.