Master Thesis

Modeling and Controller Synthesis of Hybrid Propulsion Systems using Artificial Intelligence

Alin Ionascu

November 2012

Supervisors:
Prof. Dr. Ing. Dirk Abel
Prof. Dr. Erika Ábrahám

Declaration

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Aachen, November 15, 2012

Name

Acknowledgments

Thank you.

Aachen, November 2012

Alin Ionascu

Contents

1 Experimental results **1**
1.1 Model parameters 1
1.2 GA configuration one 1
1.3 GA configuration two 1

2 Conclusion and future work **3**

Bibliography **5**

1 Introduction

1.1 Background

Human kind has always strived for technological advancement and better quality of life. Part of this was to develop new and better means of transportation. Over the course of a century we saw the number of passenger cars skyrocket. Thus in the year 2010 we have reached over 1 billion cars world wide [13], and with the current predictions, the year 2012 will be the first in which over 60 million new cars will be sold [2]. By analyzing the current vehicle density worldwide in Figure **??**[1] we can see that only the strongly industrialized or wealthy nations have a relatively high density of passenger cars relative to population size. If most of the developing countries would reach a similar density then we would see a far greater number of passenger cars.

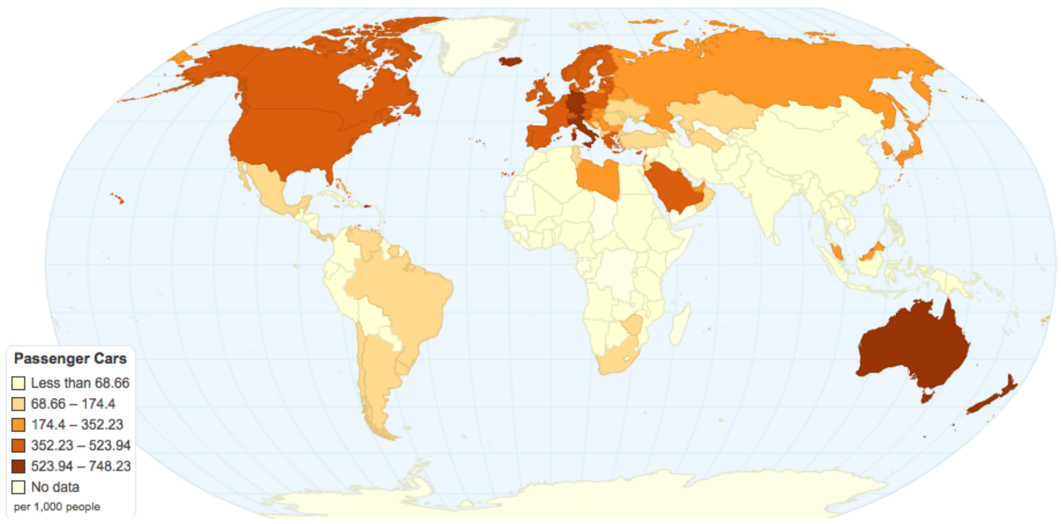


Figure 1.1: Worldwide passenger cars density (per 1000 people)

The ever increasing number of vehicles poses a series of problems from raw materials that are needed in construction, disposing of old and unused cars to more important issues like pollution and the threat of fossil fuel scarceness. In an effort to solve the later of the problems the car industry has introduced hybrid electrical

[1]Source: http://chartsbin.com/view/1113

vehicles (HEVs). They are intended to be "cleaner" vehicles due to low fuel consumption and reduced greenhouse emissions.

The industry and the research communities have ardently tried to find ways to improve the efficiency of HEVs. This search has brought up a lot of new ideas and methodologies, but as the nature of the problem is to optimize factors in a rather complex system most of the suggested solutions are based on optimization techniques.

1.2 Goals

The difference between a hybrid and a normal vehicle is that, in our case, in addition to the combustion engine we also have an electrical engine. One possible configuration for a HEV is shown in Figure **??** [2]. The electrical engine is used for different purposes: to aid the combustion engine, to increase the overall power of the car, as the primary driving engine, etc. What is interesting for this configuration is that the power demand can be split between the two engines. This is the main focus of this thesis: decide *how* to split the power demand.

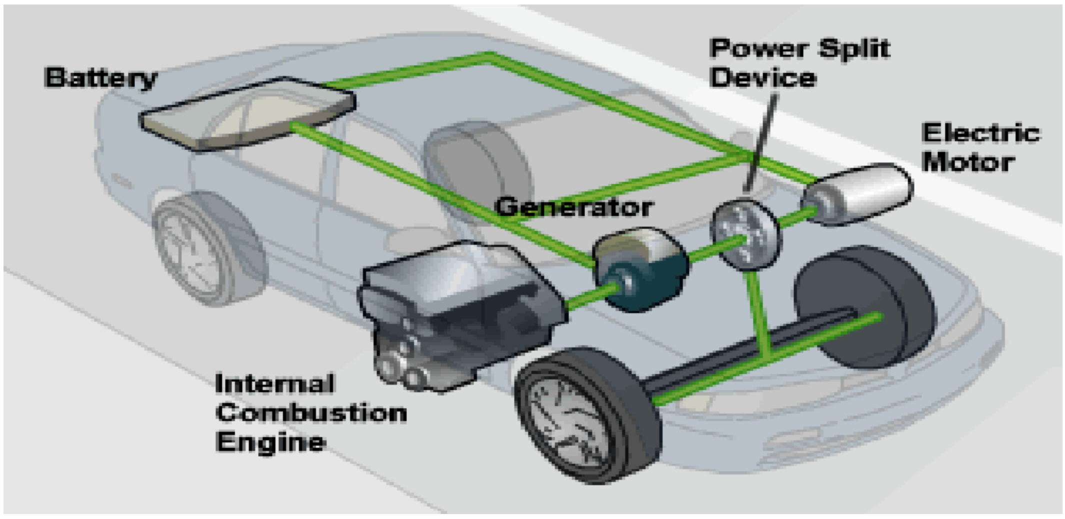


Figure 1.2: Hybrid electrical vehicle schematic

Previous work in this filed was done using methods like Fuzzy logic (Montazeri *et al.* 2008), Dynamic Programming (Sciaretta and Guzzella 2007) and many others.

The goal of this thesis is to do the optimization using a different method with the use of Genetic Algorithms (GA). Besides implementation, testing the algorithm

[2]Source: http://www.fueleconomy.gov/

is also very important. In order to do this a HEV model is needed. The second goal is to build a model, incorporate the GA into it and run simulations in order to determine and analyze the fuel consumption of the HEV.

1.3 Outline

This thesis is organized as follows:

In Chapter 2 the reader in introduced to the basic ideas behind HEV: different configurations with pros and cons, and the Matlab/Simulink simulation environment alongside with the toolbox that was used to create the model vehicle.

Chapter 3 presents the model used for the simulations with some particularities that were introduced due to using GA.

Chapter 4 describes the GA along the used library, beginning with how they work and finishing with the configuration parameters.

In Chapter 5 the simulation parameters and experimental results are illustrated and finally Chapter 6 provides some conclusions.

Notations

Acronyms

CDF	cumulative distribution function
CVT	continuous variable transmission
EB	energy buffer
EE	electrical engine
EM	electric motor
EMS	energy management strategy
FC	fuel cell
GB	gear box
HEV	hybrid electrical vehicle
HP	horse power
ICE	internal combustion engine
PPU	primary propulsion unit
rpm	rotations per minute
SoC	state of charge
TB	toolbox
ZEV	zero emission vehicle

2 Preliminaries

The hybrid electrical vehicle was first developed in 1901 by Ferdinand Porsche while he was employed at Lohner Coach Factory. The vehicle was first presented at the *Paris Auto Show* that same year. Some interesting facts about the car were that it could travel 65 Km on battery power alone, broke several Austrian speed records and won the *Exelberg Rally* in 1901 driven by Porsche himself.

The next important milestone was when Victor Wouk build his HEV prototype. His work related to HEVs in the 1960s and 1970s eard him the title *Godfather of the Hybrid* [1]. His project was to install a hybrid drivetrain with a 21HP electric motor into a 1972 Buick Skylark, which was provided by GM for the 1970 Federal Clean Car Incentive Program. Despite the performance of the vehicle the program was halted because it was considered that such a car was to expensive to produce and the gasoline prices were so low that the economy in fuel consumption didn't make much of a difference.

It wasn't until 1997 when the first mass-produced HEV was released: the *Toyota Prius*. Since then the HEVs market share increased constantly due to the fact that they have a great fuel consumption and a lower pollution rate compared to fossil-fuel-only driven vehicles.

2.1 Hybrid Electrical Vehicle basics

The term *hybrid* means: a thing made by combining two different elements [2]. Related to vehicles this means that we combine two types of propelling systems. Conventional vehicles are propelled by igniting fossil fuel into a combustion chamber, the resulting energy being converted into mechanical rotation and translation. This means that all the mechanical parts in the engine are driven by the explosion (expansion) of high pressurized gases. In contrast HEVs are characterized by using a combination of a Primary Propulsion Unit (PPU) that can be a Fuel Cell (FC) or an Internal Combustion Engine (ICE) and an Electric Buffer (EB) that can be either an electrochemical storage system such as a battery or an electrostatic super capacitor [4]. In addition a HEV must have

[1]Engineering & Science. California Institute of Technology.

[2]New Oxford Dictionary

an electric motor (EM) to help propel the vehicle partially or fully. By using a high level controller named Energy Management Strategy (EMS) the vehicle can improve its fuel consumption and reduce the pollutants that are generated.

The electric motor can help to optimize the effectiveness of the ICE or help to recuperate energy while braking. The EM can also be used to recharge the battery if there is excess energy from the ICE at some point during the operation. Another possibility is to use the EM to assist the ICE if the power demand exceeds its capabilities.

The following advantages can be achieved by using a hybrid construct:

- Recuperation of energy: The goal is to use the energy that normally would be dissipated by brakes and use the electric motor for braking, while slowing down or when driving downhill. The energy could be used to charge the battery or power the auxiliaries of the vehicle.
- Optimization of the ICE's operation: The ICE's efficiency and therefore the consumption varies with the different operating points(the combination of the rpm and the demanded torque). The electric motor could be used to shift these operating points toward values where the ICE has a better efficiency.
- Downsizing the ICE: Because not all the required force has to be produced by the ICE, some of it coming from the electric motor, the size of the ICE can be reduced in comparison to a conventional vehicle. By having a smaller engine the fuel consumption and the pollution are reduced. By using the two engines simultaneously (ICE and EM) one can achieve a greater power output. This can be needed for example when accelerating, driving uphill or towing.
- Purely electric drive: It is possible to switch to EM-only regime while driving at low speeds and turn off the ICE completely. In this operating mode the consumption of fossil fuel and the emissions are equal zero.

A HEV has also drawbacks. For instance, it is 10-30% heavier then the ICE-based vehicles [4], which means that more power is required to propel it. We have an increased cost of the vehicle, due to the presence of the electric motor, electrical energy storage, electric converters, and other components. In addition several safety issues arise: for example the use of a large electrochemical battery or the high operating voltages of some of the components.

2.1.1 HEV architectures

We can differentiate between 3 types of HEVs:

- Parallel hybrid: Both engines are connected to the same drive shaft, they can power the vehicle simultaneously or individually.
- Series hybrid: The electric motor alone is connected to the drive shaft, thus it is the one to drive the vehicle. The ICE is used to power a generator, which provides additional energy.
- Series-parallel or combined hybrid: This is a combination of the first two types.

Parallel HEVs

Parallel HEVs can be considered ICE-based vehicles that have an additional electrical path (Figure **??**). In a full hybrid configuration both engines (combustion and electrical) can work either independently or together to provide the traction force.

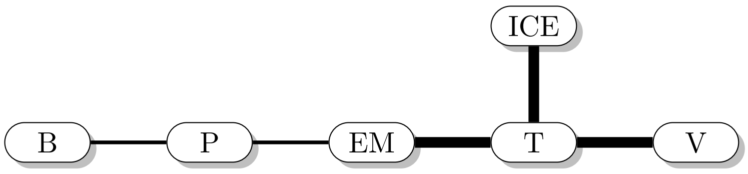


Figure 2.1: Full parallel hybrid configuration; B - battery, P - power converter, EM - electrical motor, ICE - internal combustion engine, T - transmission, V - axles and vehicle; bold lines - mechanical link, solid lines - electrical link.

This configuration facilitates the following: normally the combustion engine can be turned off while idle. The electrical motor can be used for the start-stop automatic, assist accelerations and in high-power demanding situations. The electric motor can and should be used to recuperate energy while braking or as a generator powered by the ICE. A clutch is needed because the ICE is mechanically linked to the drive train [4]. The extra weight brought by the auxiliary components is not as important as the advantages that we have for this configuration.

The most basic configuration that can be considered a parallel hybrid is the so-called *mild hybrid*. In this configuration we have a small electrical engine connected to the ICE by a belt. The basic schematic is depicted in Figure **??**.

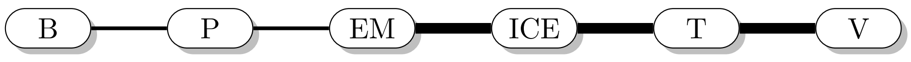


Figure 2.2: Mild parallel hybrid configuration; B - battery, P - power converter, EM - electrical motor, ICE - internal combustion engine, T - transmission, V - axles and vehicle; bold lines - mechanical link, solid lines - electrical link.

Series HEVs

In Series HEVs the prime mover is the electrical motor. Because the autonomy of the vehicle would be very low compared to conventional vehicles, we have a generator powered by an ICE as an auxiliary power unit feeding the electrical motor and the battery. The basic configuration is presented in Figure **??**. Regenerative braking is possible by using the traction motor as a generator. The operation of the ICE is not directly related to the power requirements of the vehicle. The ICE is only used when the battery is low. A series hybrid vehicle has three machines: electrical motor, internal combustion engine and the generator. At least the electrical motor has to be seized so that the power requirements of the vehicle are satisfied. With respect to the fuel consumption the difference is not notable or even higher compared to the most efficient ICEs [4], due to the extra weight of the reinforced car body, battery, additional engines, etc..

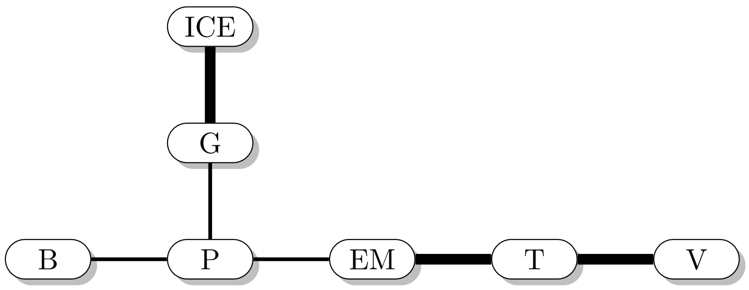


Figure 2.3: Series hybrid configuration; B - battery, P - power converter, EM - electrical motor, ICE - internal combustion engine, G - generator, T - transmission, V - axles and vehicle;Bold lines - mechanical link, Solid lines - electrical link.

The series HEV configuration has following advantages: the ICE can always be used at the optimal operating point. Another plus is the fact that a clutch isn't needed since the ICE is disengaged from the drive train.

Combined HEVs

The combined HEV is mostly a parallel hybrid with some components from the series hybrid. Both mechanical and electrical links are present along with two electric machines. One of them is used as prime mover and for regenerative braking as in parallel hybrids, and the other one is used as generator as in the series hybrid. Two possible configurations are presented in Figure **??** (Toyota THS, Ford Hybrid System) and in Figure **??** (Nissan Tino).

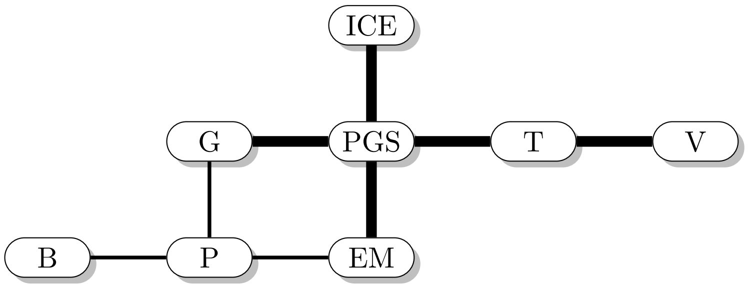


Figure 2.4: Combined hybrid configuration(version A); B - battery, P - power converter, EM - electrical motor, PGS - planetary gear set, ICE - internal combustion engine, G - generator, T - transmission, V - axles and vehicle; bold lines - mechanical link, solid lines - electrical link.

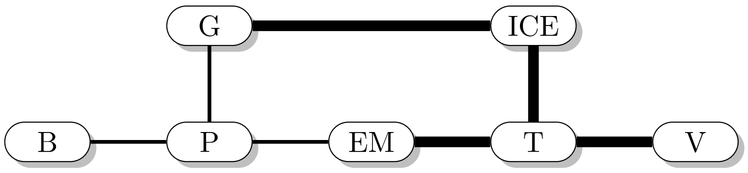


Figure 2.5: Combined hybrid configuration(version B); B - battery, P - power converter, EM - electrical motor, ICE - internal combustion engine, G - generator, T - transmission, V - axles and vehicle; bold lines - mechanical link, solid lines - electrical link.

Power flow

Every driver has his own driving style but depending on where he is driving we can differentiate between a few driving behaviors. In the following paragraphs the power flow within the HEV will be presented for each of the operation modes.

Power flow for parallel HEVs

In parallel hybrid vehicles the link between the ICE path and the electric path is mechanical [4]. The two power flows can be considered, for a full parallel design, to be combined into a *torque coupler*, because usually the electric motor is mounted on the crankshaft between the ICE and the transmission [5]. The power distribution controller is responsible for regulating the the power balance at the power coupler. It also selects the operating mode and the ratio r between the power from/to the electric motor and the total power at the coupler.

Based on the ratio r we can distinguish between different operating modes. During braking or deceleration, the electric motor is used as a generator and produces energy that is feed into the battery ($r = 1$). At startup or when accelerating the ICE provides just part of the required force, the rest being produced by the electrical engine ($0<r<1$). This is the so called *power assist* concept. At times when there is only light load, the ICE may be required to produce more power than it is demanded in order to propel the vehicle, the excess energy being used for charging the battery ($r<0$). The full hybrid configuration allows the engines to work independently. Thus we can have the pure electric drive, in the case that the battery is charged enough, and we are not in a high power-demand situation (r=1). This is called the zero emission vehicle mode. And we also have the pure ICE operation mode (r=0). All these driving modes are depicted in Figure **??**.

Power flow for series HEVs

In an standard series hybrid configuration the link between the ICE path and the battery path is electrical. Also the terminal current at the battery is determined by the terminal current from the electrical motor and the terminal current from the generator. The power balance at the power link is controlled by the *torque distribution controller*. It selects the operating mode for the vehicle and the ratio r between the power from/to the battery and the total power at the link [4].

As in the case of full parallel hybrids we have different operating modes. In the case of series hybrids we have basically four modes. If the battery is sufficiently charged and we have a moderate power demand, i.e., urban driving, then the vehicle could be operated purely electric (ZEV), r=1. When the battery charge is to low normally the ICE is turned on which powers the generator. The energy produced by the generator is used to feed the prime mover and to charge the battery, $r<0$. While braking or decelerating the electrical motor is used as generator and some of the energy is used to recharge the battery. In the case that the power demand is high and the power provided by the generator is not enough, the battery is also used to fill the gap until the required power level is reached, $0<r<1$. Although it's possible it is very unlikely for this scenario to happen. All four operating modes are presented in Figure**??**.

Power flow for combined HEVs

Being a combination of series and parallel hybrid vehicles the combined HEV can operate as either one or the other. Thus, all the operating modes that have been presented in the previous cases also apply here. The operating modes will be depicted in Figure **??**. However there is a difference: the use of an planetary gear drive adds some constraints to the possible energy paths. The pure ICE operation is often associated with a power flow through the generator and the electric motor [4].

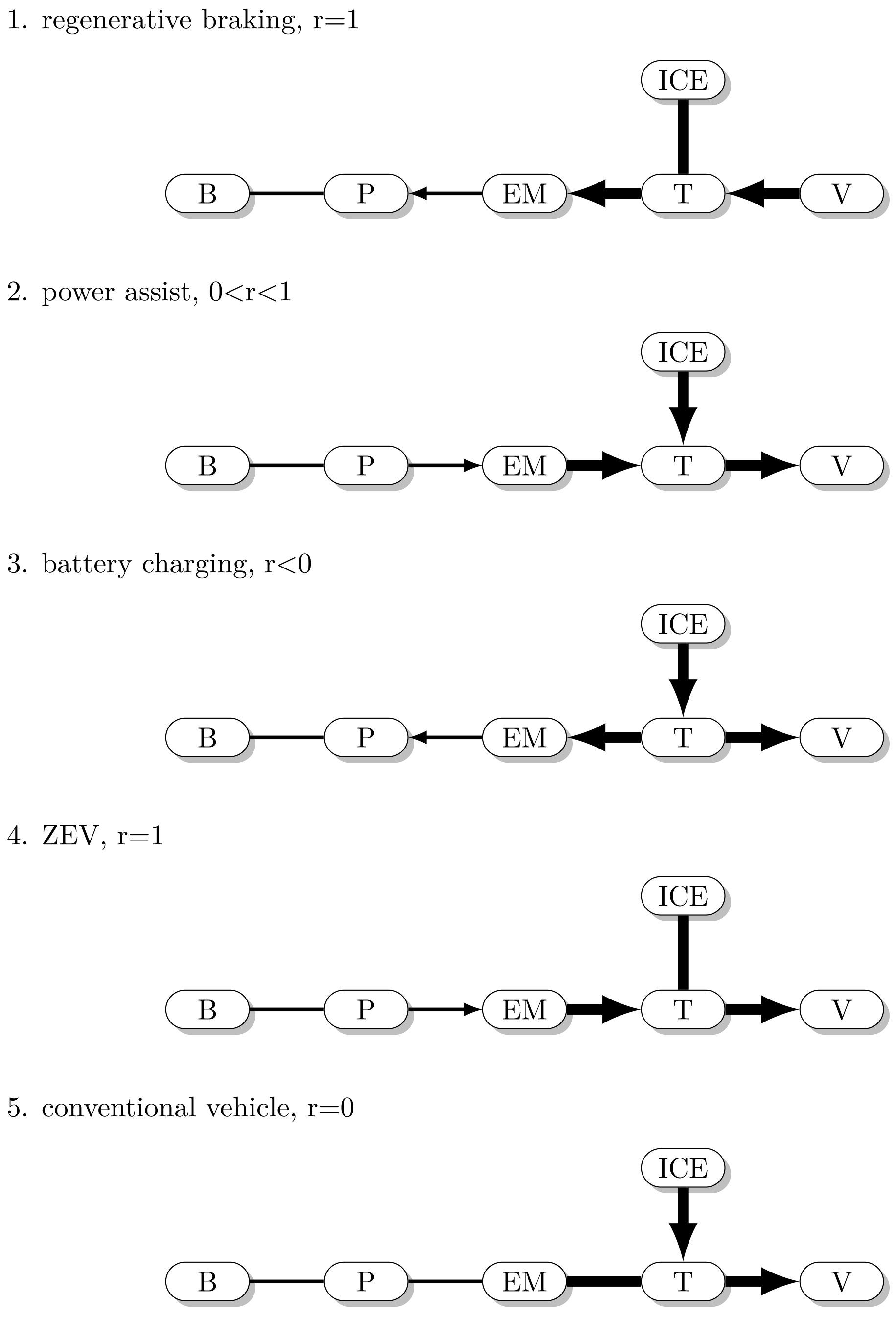
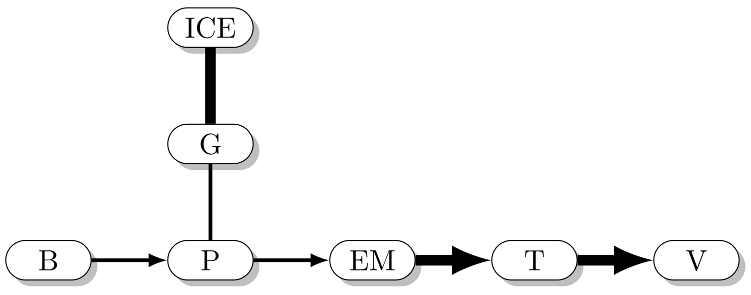
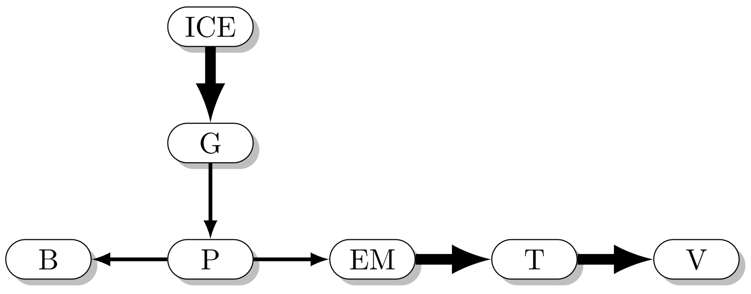


Figure 2.6: Power flow for a full hybrid vehicle; B - battery, P - power converter, EM - electrical motor, ICE - internal combustion engine, T - transmission, V - axles and vehicle; bold lines - mechanical link, solid lines - electrical link.

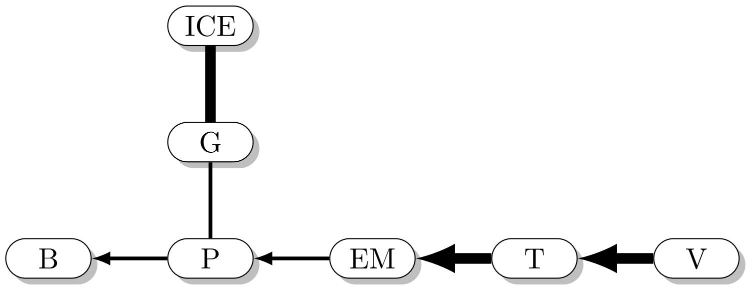
1. ZEV, r=1



2. battery recharge, r<0



3. regenerative braking, r=1



4. hybrid drive, 0<r<1

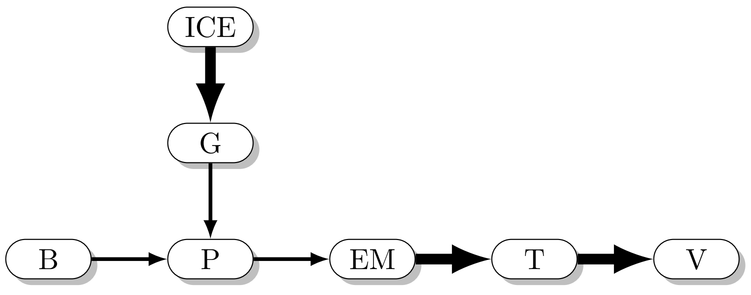
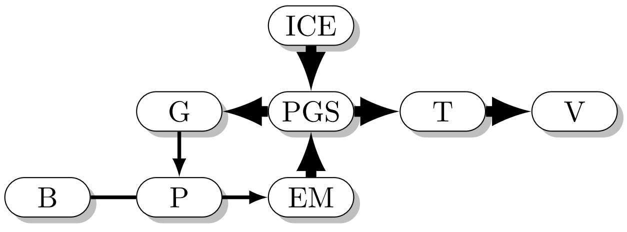
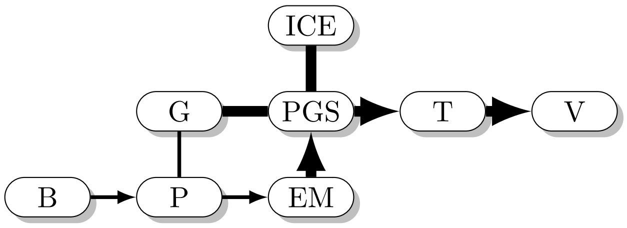


Figure 2.7: Power flow for a series hybrid configuration; B - battery, P - power converter, EM - electrical motor, ICE - internal combustion engine, G - generator, T - transmission, V - axles and vehicle; bold lines - mechanical link, solid lines - electrical link.

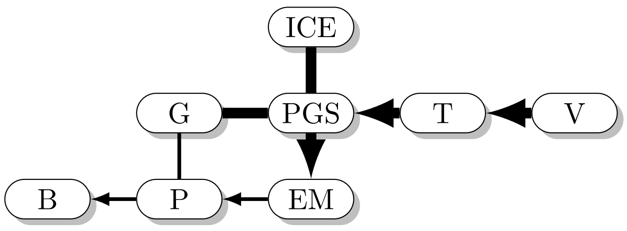
1. ICE-only mode



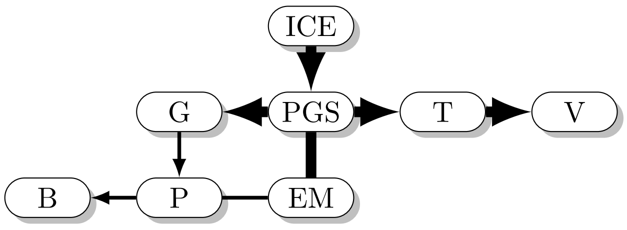
2. ZEV



3. regenerative braking



4. battery recharging



5. power assist

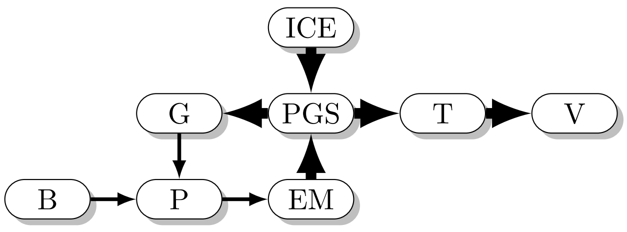


Figure 2.8: Power flow for a combined hybrid configuration; B - battery, P - power converter, EM - electrical motor, ICE - internal combustion engine, G - generator, T - transmission, PGS - planetary gear system, V - axles and vehicle; bold lines - mechanical link, solid lines - electrical link.

2.2 Simulation environment and toolbox

In order to test the control strategy we need a model of a HEV. The HEV model is built using the Matlab simulation environment. In the following some basics about Matlab and Simulink will be presented along with the toolbox that was used to build the model.

2.2.1 Matlab/Simulink

Matlab is a high-level language and interactive environment for numerical computation, visualization, and programming [7]. It can be used to analyze data, write algorithms and create models and applications. The name Matlab is an acronym for **Mat**rix **lab**oratory. This is because in the beginning all the data that was used in Matlab was interpreted as a matrix. In the latest releases the support for symbolic programming has been added in form of a toolbox (*MuPAD symbolic engine*).

Matlab is an interpreted language used for numerical computations. In the following some particularities for Marlab will be discussed. It allows the user to perform numerical calculations, and visualize the results without the need for complicated and time consuming programming. Being user friendly can also have disadvantages, for example Matlab can be slow. If one adds poor programming practices then the program becomes unacceptably slow. Although this has been an issue that has been addressed continuously over the years and there have been improvements it's not as fast as, for example, C or C++. Luckily Matlab offers the possibility to export a program or a model as a C/C++ executable file, thus overcoming this problem.

Matlab has a wide variety of toolboxes and add-ons, the most relevant one for the present thesis being Simulink.

Simulink

Simulink is a Matlab add-on developed by Mathworks. It is used for modeling, simulation and analyzing multi-domain dynamic systems. Simulink provides a graphical editor, standard and customizable block libraries, and solvers for simulation. It is also tightly integrated into Matlab so data can be easily passed between Simulink and Matlab.

Simulink's key features [8] are:

- Graphical editor for building and managing hierarchical block diagrams;
- libraries of predefined blocks for modeling continuous-time and discrete-time systems;

- simulation engine with fixed-step and variable-step ODE solvers;
- scopes and data displays for viewing simulation results;
- project and data management tools for managing model files and data;
- model analysis tools for refining model architecture and increasing simulation speed;
- MATLAB function block for importing MATLAB algorithms into models;
- Legacy Code Tool for importing C and C++ code into models;

One of the particularities of Simulink is that a model is built using *blocks*. A block is the fundamental building piece. A block can represent a constant, a variable, a function or even a system in another system and include a model as block in another model. For example the rectangle marked with "A" in Figure **??** is a block. A complete list of blocks that are available in Simulink can be found here [9]. Blocks can be grouped hierarchically. This means that a block can contain other blocks and these in turn can contain other blocks.

Connecting blocks in an Simulink model is done via input and output ports (Figure **??**). The information flow form one block to another is represented as signals. Signals can be continuous or discrete. In a Simulink model the information flow is always from left to right. As can be seen in Figure **??** all blocks have the input ports on the left side and the output ports on the right side. The input signal is acquired from the input port, modified in a specific way (depending on the block) and written at the output port. This means the leftmost block is the one where we have the initial signal and the rightmost is the one where we have the model output. This applies for the model in Figure **??**. More complex designs are possible, designs with multiple input points and multiple output points, as well as more complex connections between the blocks are possible.

2.2.2 QSS toolbox

The QuasiStatic Simulation Toolbox (QSS Toolbox) is used to determine the fuel consumption of many powertrain systems [3]. This toolbox includes a number of Simulink blocks that can be used to build a model of various vehicles. As the name suggests the toolbox uses the quasistatic approach in order to determine the fuel consumption.

The key idea behind the quasistatic approach is to reverse the usual cause-and-effect relationships of dynamic systems [4]. Thus, rather than calculating

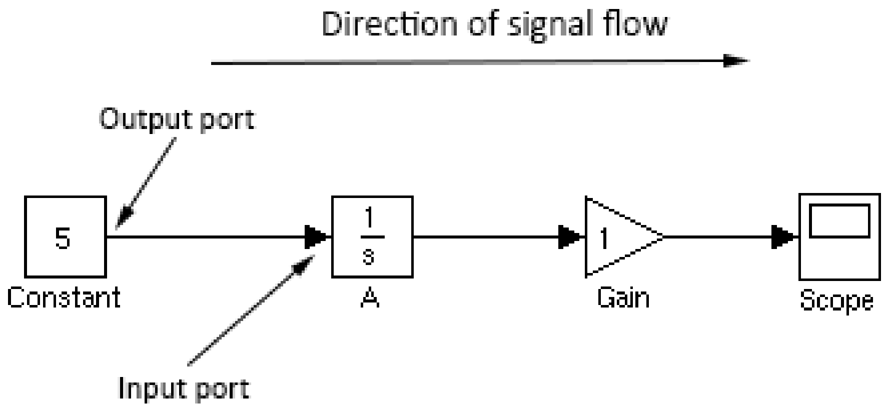


Figure 2.9: Basic Simulink model;

speeds from given forces, the toolbox calculates accelerations and determines the necessary forces based on given speeds (at discrete times). One important aspect in this context is that the toolbox can determine the necessary forces only at discrete times. This means that our driving cycle is divided into many short time steps h, where the speed v and acceleration a are constant. Once the necessary forces F_t are calculated, F_t - *total force*, being the combination of all present forces, the fuel consumption can be estimated.

The fuel consumption is estimated as the mean over the consumptions for each time step h which compose the driving cycle.

$$fc_t = (fc_{1,2} + fc_{2,3} + ... + fc_{(i-1)\cdot h, i\cdot h} + ... + fc_{(n-1)\cdot h, n\cdot h})/n \tag{2.1}$$

fc_t - total fuel consumption; $fc_{(i-1)\cdot h, i\cdot h}$ - fuel consumption for time interval $(i-1, i)$; n - total number of time steps for the driving cycle.

In order to compute the total fuel consumption we have to know the fuel consumption for each short time interval individually. These can be computed based on the fact that the fuel power consumed by the ICE, P_f, can be determined and the lower heating value[3] [4] for the fuel is known a priori. The following equation is used to determine the fuel consumption:

$$m_f = P_f/H_l \tag{2.2}$$

m_f - mean fuel mass flow; P_f - fuel power consumed by the ICE; H_l - lower

[3]Lower heating value - is determined by subtracting the heat of vaporization of the water vapor from the higher heating value.

[4]Higher heating value - is the amount of heat released during the combustion of a specified amount of fuel. The higher heating value does not incorporate losses due to vaporization of water.

heating value for fuel.

Further details on how to determine the forces and losses in a HEV and also other details related to ICE and drivetrains can be found in [4].

The QSS toolbox comes in form of a Simulink library. The contents of the library are depicted in Figure ??. The various elements of the QSS TB library are grouped according to their function and to open one of the blocks one must just double click on it.

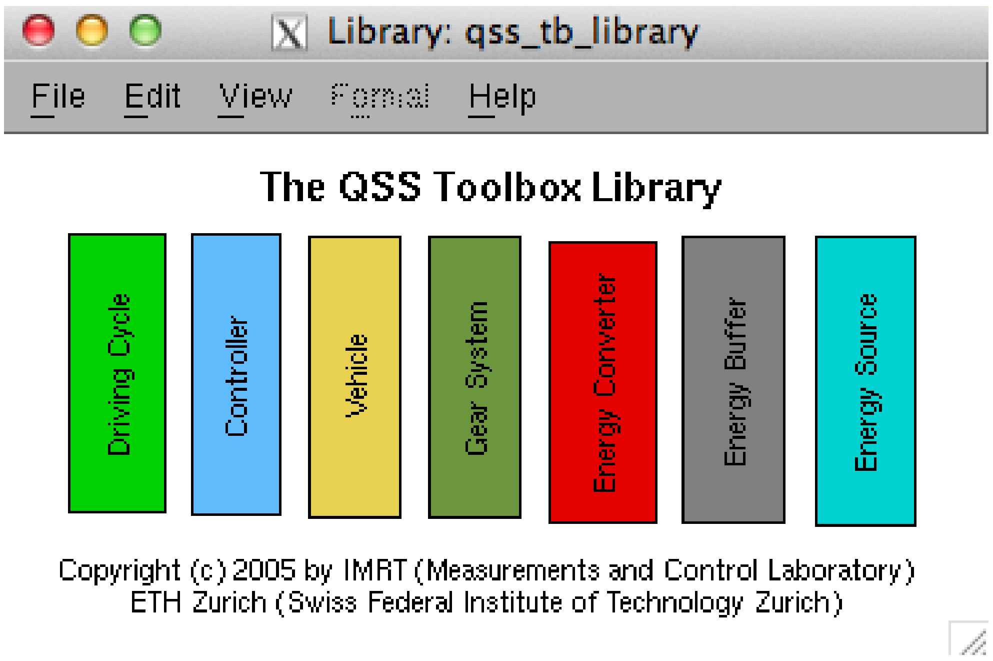


Figure 2.10: QSS toolbox library;

The blocks that are available in the QSS library [3] are:

- Driving Cycle : This block loads the cycle data such as time, speed, acceleration and gear number, from the specified data file and makes it available on the output of the cycle block.
- Controller:
  - Control Unit (empty template): A control unit that can be defined by the user.
  - Battery Controller: A simple control algorithm that starts the ICE powering the generator as soon as the battery SoC has reached the

lower threshold Q_BT_min. The ICE/Gen are operated at a given load/speed for a given amount of time. The amount of time is so selected that by the end of the driving cycle we will have the same SoC for the battery as it was at the beginning.

  – CVT Controller: This block calculates the optimum gear ratio and when to change the gear during one integration step for a CVT.

- Vehicle: This block computes the forces that are required at the wheels in order to fulfill the demand at the input.
- Gear System:
  – Simple transmission - Simulates a gear box with a fixed gear ratio.
  – Manual gear box - Simulates a gear box where the gears can be selected by the user.
  – Continuously variable transmission - Simulates a CVT transmission.
- Energy Converter:
  – Combustion Engine (consumption map) - Simulates the behavior of an ICE. Block is based on a consumption map.
  – Combustion Engine (Willans-approximation) - Simulates the behavior of an ICE. The block is based on the Willans approximation with friction part based on the ETH model.
  – Electric Motor - Simulates the behavior on an electric motor. The block is based on a efficiency map.
  – Electric Generator - Simulates the behavior on an electric generator. The block is based on a efficiency map.
  – Fuel Cell - This block simulates a fuel cell and converts fuel into electric energy.
- Energy Buffer:
  – Supercapacitor - The supercapacitor is a fast electric energy buffer.
  – Battery - The block simulates a battery. The resistance depends on the charge state and the charge/discharge current of the battery. The battery has an open circuit voltage of 130 V (fully charged).
- Energy Source : This block integrates the power requirement and calculates the fuel that is required in liter/100Km.

Each model is stored in a .mdl file (this is the extension for Simulink model files). In the following Figure **??** a basic ICE vehicle is represented using the QSS TB.

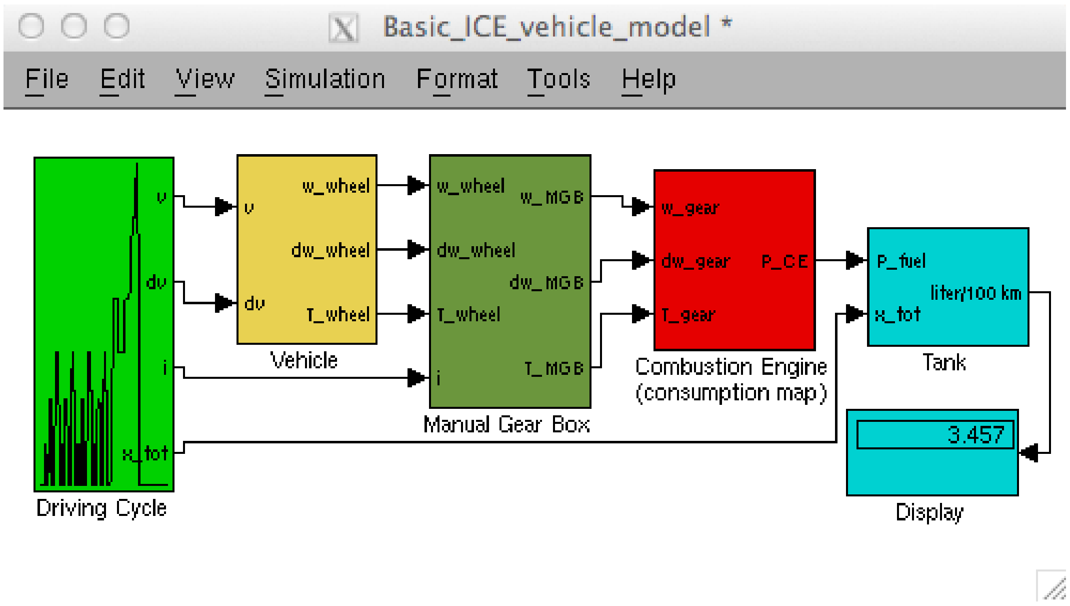


Figure 2.11: The .mdl file for a basic ICE vehicle using the QSS TB;

The blocks will be explained from left to right. First a driving cycle must be selected. Here we can select from the list of standard driving cycles that comes with the QSS TB or create a custom one. Given the speed v and acceleration a (noted dv in the library) the Vehicle block calculates the torque that is required at the wheels. While the torque is determined based on the selected gear, the transmission block (Manual Gear Box block) calculates the net torque that is required from the ICE. From the total torque, which consists of net torque plus acceleration torque, the ICE then "produces" an amount of fuel consumption. In the end the Tank block determines the average fuel consumption measured in litters/100Km.

3 Full parallel HEV model

Throughout this chapter the model used in this master thesis is presented. A full parallel HEV model is used because all the operation modes (Section 2.1.1) are possible to implement and test. A full HEV schematic is illustrated in Figure **??**. The model has two important components: the chassis model and the powertrain model. The chassis model is built on Newton's second law. Powertrain model is built around the power that is needed at the wheels and how the power gets there from the prime mover(s).

Because the model is designed using the QSS TB the cause-and-effect relationship is reversed meaning that the driving cycle is known a priori, and based on the parameters of the vehicle the fuel consumption can be estimated at the end of the simulation.

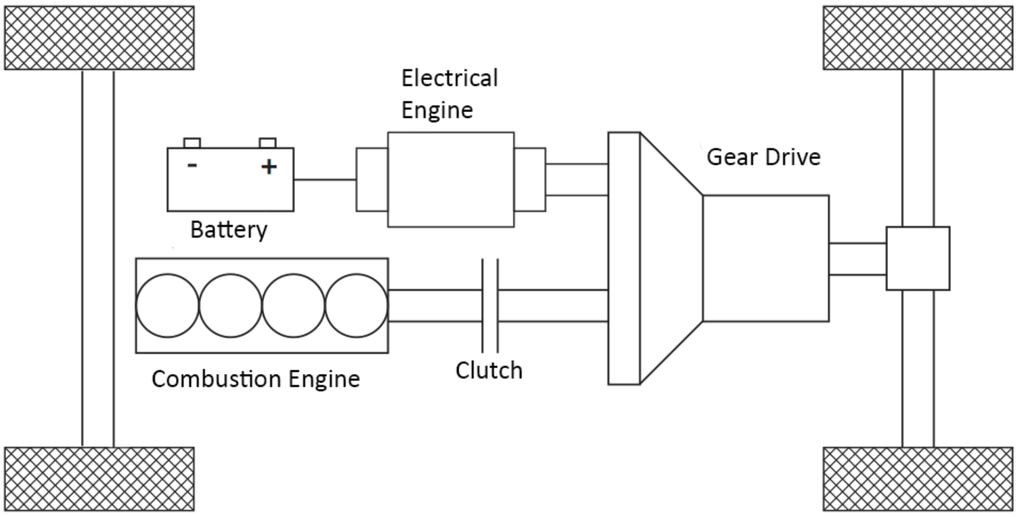


Figure 3.1: Full parallel HEV schematic.

The starting values for the parameters of the vehicle have been taken from the Toyota Prius III specification [11], [10]. These parameters along with the modification of some of them will be presented in the following sections.

The full parallel HEV model used in this master thesis is depicted in Figure

??.

3.1 Driving cycle and chassis model

3.1.1 Driving cycle

In order to be able to compare fuel consumptions for different vehicles and configurations we need a standardized test drive, also called *driving cycle*. In Europe the most used driving cycle is the EU cycle and in the US the FTP cycle. Generally, these driving cycles tend to be "soft" compared to the situations in real life. This means that in a normal situation the driver would drive with values for accelerations and top speeds that exceed the ones from the driving cycle.

In Figure **??** the leftmost block (landscape view) is the "Driving Cycle". The block offers the opportunity to select which driving cycle to use, and the length of the time steps. It is also possible to select a customly designed driving cycle.

In general a driving cycle is defined by at least two vectors:

- a time vector (usually with equal time steps),
- a vehicle speed vector $v(h)$, where h is the position in the speed vector.

The block then calculates from the given speed the acceleration. The acceleration is treated predictively (Equation **??**). It is also possible to be treated retrospectively, but this would yield less realistic results [3].

$$a(k \cdot h) = \frac{v(k \cdot h + h) - v(k \cdot h)}{h} \tag{3.1}$$

There are some differences between the built-in driving cycles. For example, gear shift points for the EU driving cycle are given as opposed to the US FTP cycle where there is no gear shift vector declared.

In order to define a EU like driving cycle three vectors have to be declared: the time vector, the speed vector and the gear vector. Each vector is declared in its own .mat[1] file. The Matlab files in the case of the present model are: `T_Z.mat`, `V_Z.mat`, `G_Z.mat`.

[1]Matlab file extension for variables.

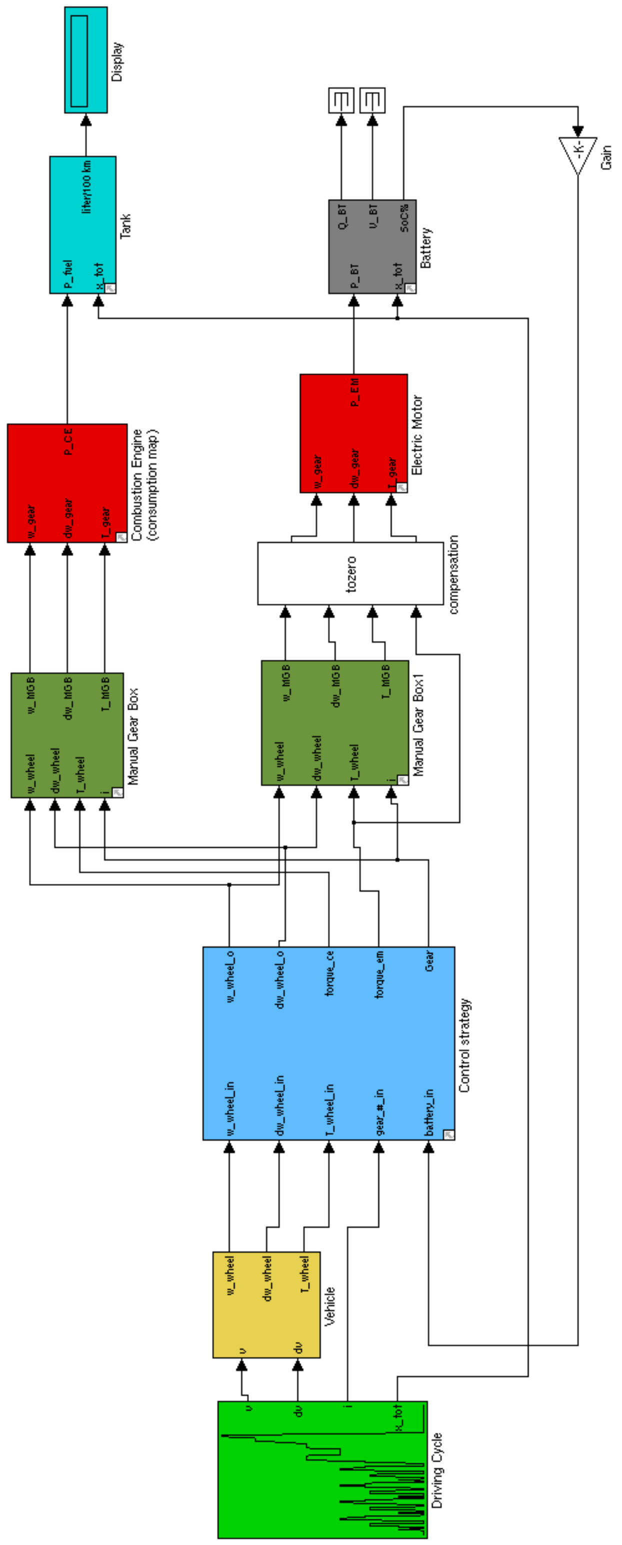


Figure 3.2: The .mdl file for the full parallel HEV model.

Figure **??** depicts the "Driving Cycle" block with given gear values. The output values for this block are the speed v, the acceleration a (dv), the gear i and total distance for the driving cycle x_tot.

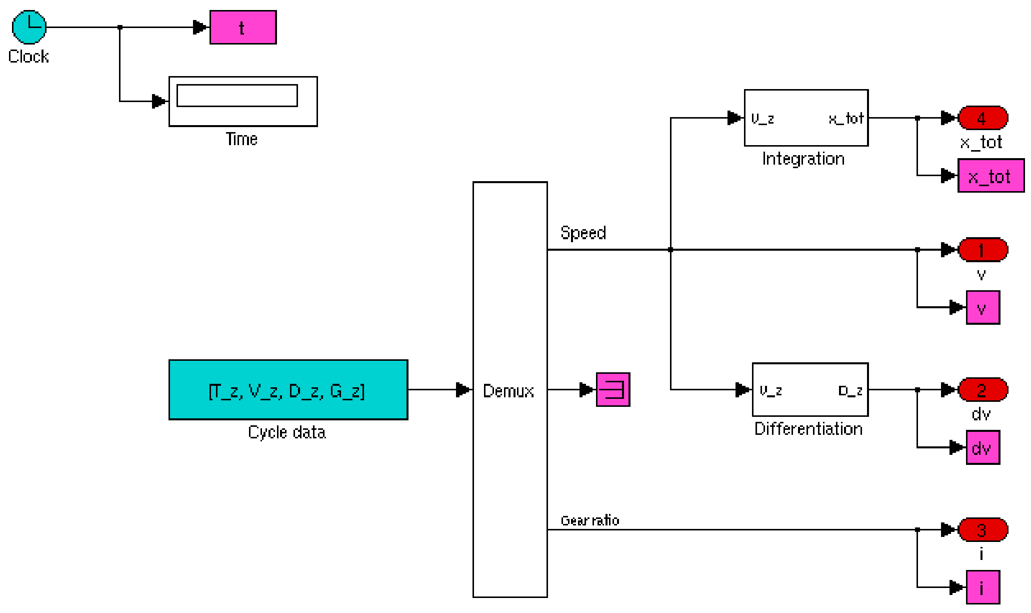


Figure 3.3: Schematic representation of the block "Driving Cycle" with given gear values.

3.1.2 Chassis model

In order to determine the longitudinal dynamics of a vehicle on a road Newton's second law of motion can be utilized [4]. This law states that the acceleration a of a body is parallel and directly proportional to the force F that acts on the body and inversely proportional to its mass m. The equation can be rewritten:

$$m \cdot a = F_{tot}, F_{tot} = F_t - F_{rez} \tag{3.2}$$

F_{tot} is the combination of all the forces that act upon the vehicle. F_{tot} is the result resulting from the difference between the traction force F_t and the forces that act in a restive way F_{rez}.

The total running resistance can be divided [3] into four groups:

- aerodynamic friction losses.
- rolling friction losses.

- losses due to inertia.
- external resistance losses, for example climbing.

In the QSS TB, the "Vehicle" block, Figure **??** incorporates the first three effects.

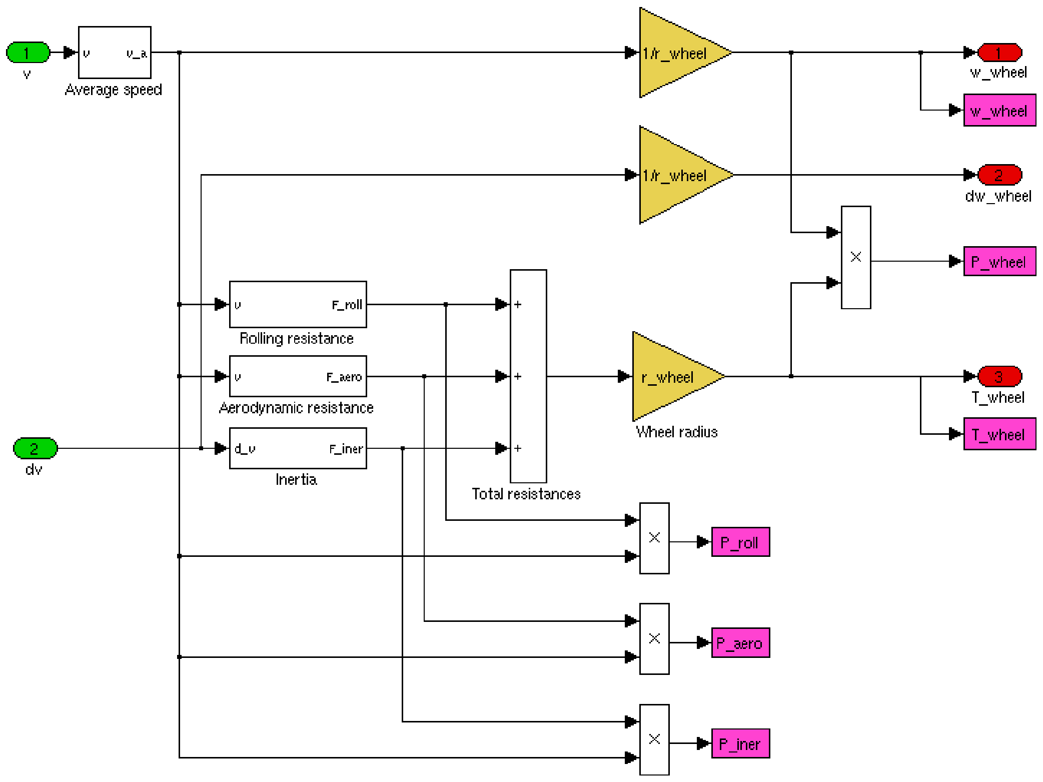


Figure 3.4: Vehicle block - First level.

Equation (**??**) where F_{tot} is represented by the difference between the traction force and the resistive force can be expressed as follows:

$$m \cdot a = F_t - \frac{1}{2} \cdot \rho_a \cdot A_{front} \cdot c_d(v, ...) \cdot v^2 - c_r(v, p, ...) \cdot m \cdot g \cdot cos(\alpha). \quad (3.3)$$

The first negative part of the right side expression is the formula for the aerodynamic friction losses:

$$F_a = \frac{1}{2} \cdot \rho_a \cdot A_{front} \cdot c_d(v, ...) \cdot v^2 \quad (3.4)$$

Here v is the vehicle speed and ρ_a is the density of the ambient air. A_{front} is the front area of the car. It accounts for the approximated values of the car body effect, wheel housings effect, side mirror effect, window housings effect, engine ventilation, antennas, etc. [4]. The parameter $c_d(v, ...)$ is the drag coefficient which must be determined using CFD programs or experiments in wind tunnels. For the estimation of the required energy this parameter may be assumed to be constant [4].

The second negative expression stands for the rolling friction losses.

$$F_r = c_r(v, p, ...) \cdot m \cdot g \cdot cos(\alpha) \tag{3.5}$$

Here m stands for the mass of the vehicle, g is the gravitational acceleration (9.81 m/s). The term $cos(\alpha)$ models the influence of a inclined road. The rolling friction coefficient c_r depends on many factors. The most important ones are the vehicle speed v and the tire pressure p. The vehicle speed has little influence at lower speeds but becomes more influential as it approaches a critical value where resonance phenomena start. When the vehicle speed remains moderate the friction coefficient c_r may be assumed to remain constant [4].

The resistance due to inertia forces can be approximated by determining the equivalent mass of the rotating parts and it should be added to the vehicle total mass m [4].

The power demand at the final drive (at the output of the vehicle block) can be positive or negative. A positive power demand at the final drive means a power request from the prime mover(s) and a negative one means an extra power that was produced by the vehicle and that can be used to charge the EB (battery in this case).

As noted before, the QSS approach requires that all values are constant during the time steps h.

3.2 Powertrain model

The power train model for the present work is the model of a full parallel HEV. The aim of the powertrain is to deliver the requested power P_{demand} to the wheels via its two paths (the engine path and the electrical machine path), to utilize the excess power produced by the ICE and store the power in the EB and finally to use the braking energy by using the electric motor as generator and feed the excess power into the battery.

The control strategy that controls how much torque goes one path and how much the other will be presented in detail in the next chapter. In the next sections the various components of the drivetrain will be described.

3.2.1 ICE model

The ICE model used is based on the quasistatic approach (Figure **??**), meaning that we have the torque T_d and the angular speed ω_d demand as inputs and the produced power P_{ICE} as output.

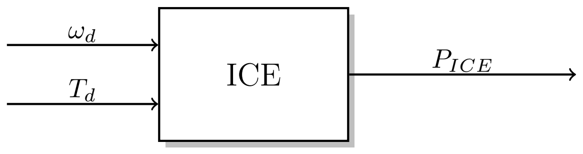


Figure 3.5: ICE input and output variables in the quasistatic system description.

Important is that with the ICE block we can estimate the fuel consumption for the given inputs. The QSS TB offers two possible ways to model an ICE: based on an engine map or based on the Willans-approximation. In our case the fuel consumption is determined based on an engine map. According to this approach the internal combustion engine essentially produces a fuel consumption from a torque and a rotational speed demand (program map block "Engine consumption map" in Figure **??**).

Based on the considerations in [4] the QSS TB uses two-dimensional fuel consumption maps:

$$V_{ICE} = f(T_d, \omega_d) \tag{3.6}$$

V_{ICE} is the fuel volume, T_d is the demanded torque and ω_d is the demanded rotational speed.

The block "Engine consumption map" requires 3 parameters:

- a vector of dimensions $1xn$, *w_CE_row* containing the rotational speed support points[2].
- a vector of dimensions $mx1$, *T_CE_col* containing the torque support points.
- a map nxm named *V_CE_map* representing the fuel use data (in kg/s) based on the inputs: rotational speed and requested torque.

[2]The points do not have to be equidistant, but the values have to increase strictly monotonously.

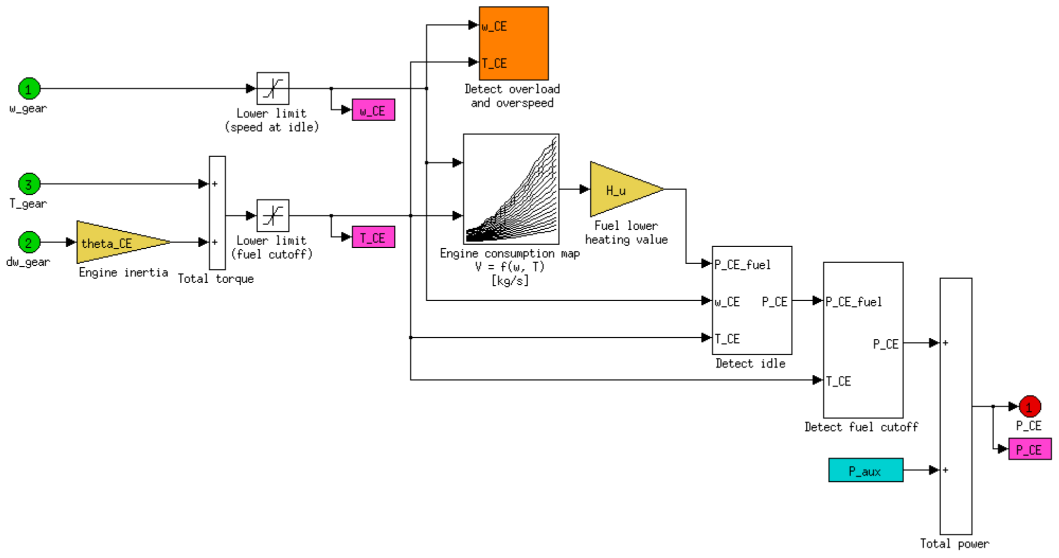


Figure 3.6: "Combustion engine" block - First level;

The engine map that was used for the simulations in illustrated in Figure **??**.

The ICE torque consists of the net torque plus the engine acceleration torque. Thus a few special engine operation modes have to be considered. As can be seen in Figure **??** these operation modes are: idle operation, fuel cutoff and we have to consider the situations for overload and overspeed.

The consumption in the idle state is determined in the following way. If the requested rotational speed falls under the lower limit specified by the parameter *w_CE_idle* and the requested torque drops below the lower limit specified by *T_CE_idle* (which is typically 0 [3]) the engine power is set to the value of *P_CE_idle*.

The detection of overload and overspeed is realized as follows. As soon as the values for the rotational speed and the torque, which is based on the rotation speed, exceed their maximum value the simulation is stopped.

The fuel cutoff function is activated as soon as the total load falls below a specified limit (*T_CE_cutoff*) and the power of the ICE is no longer determined by the consumption map but is set to the value of a fixed parameter *P_CE_cutoff*. The fuel cutoff feature overrides the idle state [3]. This means that even if we are in the idle state and the load falls below *T_CE_cutoff*

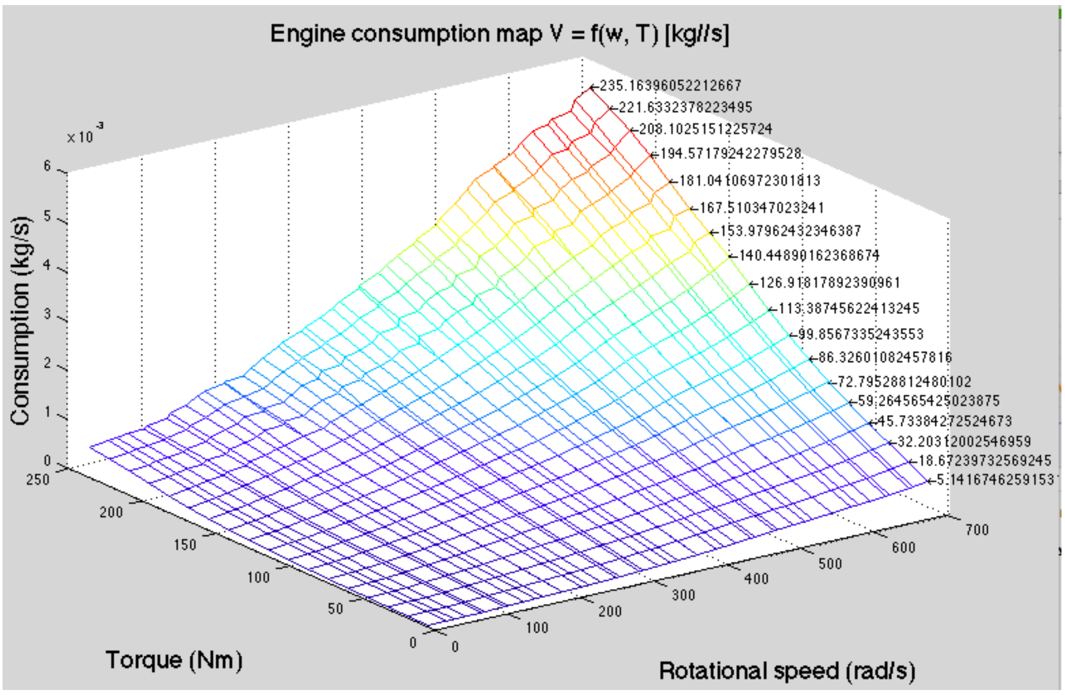


Figure 3.7: Consumption map for the ICE;

then the power of the ICE is set to *P_CE_cutoff*.

3.2.2 Electric motor model

The model for the electric motor is similar to the one for the ICE, in the sense that we have as input the rotational speed and required torque. The output is the required electric power. The electric motor can be used as an generator, i.e., it produces energy , $P_{EM} < 0$ or as a prime mover, i.e., energy is consumed, $P_{EM} > 0$.

As in the case of the ICE the EM needs a detection for overspeed and overload. The EM model is illustrated in Figure **??**. The behavior is similar to the one form the ICE. As soon as the limits for rotational speed or torque are surpassed the simulation is halted.

The output power of the EM (P_{EM}) is determined based on an efficiency map. The map is similar to the consumption map of the ICE but the data is interpreted in a sightly different way. The electric motor can be operated in two quadrants based on the values of rotational speed ω_{EM} and torque T_{EM}. The power required in the first quadrant $\omega_{EM} > 0$ and $T_{EM} > 0$ can be

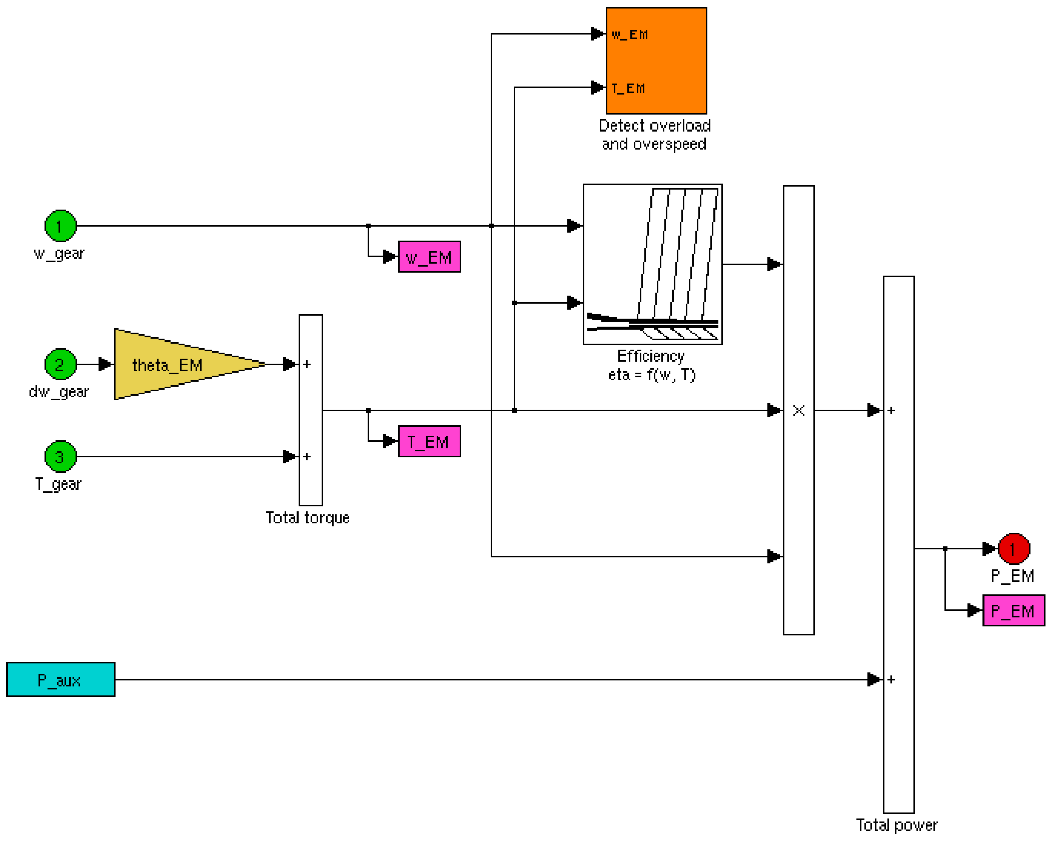


Figure 3.8: QSS TB model for an electric motor.

expressed as follows:

$$P_{EM} = \omega_{EM} \cdot T_{EM} \cdot \frac{1}{\eta_{EM}(\omega_{EM} \cdot T_{EM})}, \tag{3.7}$$

the expression for the second quadrant ($\omega_{EM} > 0$ and $T_{EM} < 0$) being:

$$P_{EM} = \omega_{EM} \cdot T_{EM} \cdot \eta_{EM}(\omega_{EM} \cdot T_{EM}). \tag{3.8}$$

In both cases, $\eta_{EM}(\omega_{EM} \cdot T_{EM})$ represents the efficiency of the electric motor for the given rotational speed and torque.

3.2.3 Battery

The battery in the model is built in accordance with the quasistatic approach presented in [4], depicted in Figure ??.

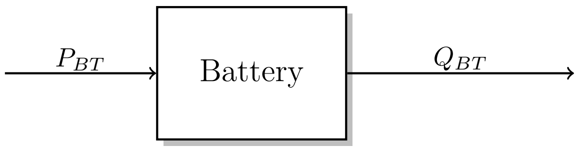


Figure 3.9: Battery input and output variables in the quasistatic system description; P_{BT} - electric power, Q_{BT} - battery charge.

According to the quasistatic approach, the electric power P_{BT} is the input for the battery model (positive while discharging and negative during loading). The output is represented by the battery charge Q_{BT}. Figure **??** shows the top level of the battery model.

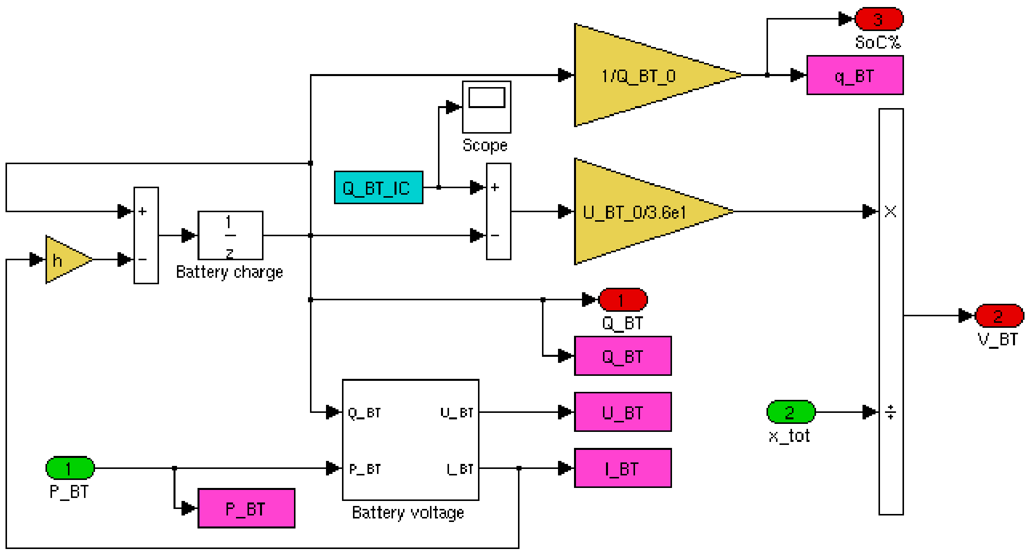


Figure 3.10: QSS TB battery model.

The battery is modeled basically with the use of an open circuit voltage (V_{oc}) and a resistive circuit (Figure **??**). The open circuit voltage depends at any moment on the battery state of charge (SoC) and the resistive circuit depends on whether the battery is charging or discharging. The battery resistances R_{ch} and R_{dch} are determined in each time step by interpolating a static charge/discharge vector. Based on the current SoC of the battery, the requested torque of the EM, the rotational speed of the EM, the total capacity of the battery, and the battery current, the next SoC can be determined using the equation (**??**).

$$SoC_{k+1} = SoC_k + \frac{i \cdot h \cdot (T_{EM}, \omega_{EM})}{Q_{BT}} \tag{3.9}$$

h is the time step, SoC the state of charge, i the battery current that can be positive/negative depending whether the battery is charging or discharging respectively, Q_{BT} the battery capacity.

The battery current can be calculated as following:

$$i = \frac{V_{oc}}{2 \cdot R_{ch/dch}} + \sqrt{\frac{V_{oc}}{2 \cdot R_{ch/dch}} + \frac{P_{EM}}{R_{ch/dch}}} \tag{3.10}$$

where V_{oc} is the open circuit voltage, $R_{ch/dch}$ is the equivalent resistance of the charging/discharging resistance, P_{EM} is the power from the EM. The open circuit voltage depends on the SoC as well as the EM power.

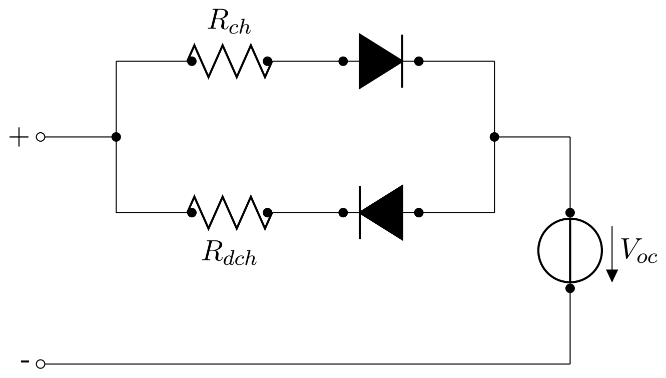


Figure 3.11: Simplified battery schematic.

The model described in this section only functions correctly in the 10% - 90% battery charge range [3]. Outside of this interval we have strong non-linear effects. Operating outside this range may also reduce the life span of the battery and its efficiency. During operation the battery may be subject to short, but high-energy peaks, which may occur during acceleration phases. It is recommended to use supercapacitors in order to deal with the high-energy peaks, or else there is a need for a high capacity battery.

The battery model differs in a slight way form the model incorporated in the QSS TB, due to an additional output port. This port is used as a replacement for the Q_BT port. The new port SoC% represents the state of charge of the battery rather than the amount of energy that was still in the battery, Q_BT. The range for the SoC is [0,1], this is why we have a gain placed on the connection from this port. In this way, the values obtained values are between 0% and 100% for the SoC.

3.2.4 Constraints

The constraints referring to the powertrain presented so far can be summarized as:

- the ICE power is upper bounded (**??**);
- ICE torque is limited according to **??**;
- EM torque is limited according to **??**;
- battery Soc is represented as ,0% - battery empty, 100% - battery full(**??**);
- limited power for the battery (**??**).

$$P_{ICE} \leq P_{ICE,max} \tag{3.11a}$$
$$T_{ICE}(\omega_{ICE}) \leq T_{ICE,max}(\omega_{ICE}) \tag{3.11b}$$
$$T_{EM}(\omega_{EM} \in [T_{EM,min}(\omega_{em}), T_{EM,max}(\omega_{em})] \tag{3.11c}$$
$$SoC \in [0\%, 100\%] \tag{3.11d}$$
$$P_{BT} \in [P_{BT,min}, P_{BT,max}] \tag{3.11e}$$

3.3 Model particularities

The Toyota Prius III vehicle uses a planetary gear drive. However we use two manual gear boxes in this model, in order to be able to simulate the presence of a gear drive, but at the same time to be able to split the power demand between the two prime movers, because there is no model for the planetary gear drive in the QSS TB and planetary gear drives are rather specific for each manufacturer (ratio between the gears).

As can be seen in Figure **??**, the rotational speed and acceleration is the same for the two manual gear boxes, only the torque demand is different. The behavior of the two gear boxes might not be the same as the one for the planetary gear drive, but from the point of view of the control strategy there is no difference. The purpose of the control strategy is to decide how to split the power demand between the prime movers.

One of the operating modes for a full parallel HEV is the ICE-only operation. This means that the vehicle is propelled by the ICE, the EM being decoupled. One of the problems that we encountered was that if the split between the two engines is 100% and 0% (ICE, EM respectively), but we have a positive rotational speed and a given acceleration, the EM still produces a positive power. This means that the EM is consuming energy from the battery. We bypassed the problem by introducing an additional block between the gear drive for the electrical motor and the electrical motor. This block checks if the requested torque is equal to 0, and if so then all the outputs of the port are set to 0. In all the other cases the outputs reflect the values at the input ports. Adopting this solution ensures that no energy is lost from the battery in the ICE-only operation mode.

4 Control strategy

In this chapter the control strategy will be covered. As described in the previous chapters the strategy is tested on a model that has a reversed cause and effect relationship. Thus the input is represented by the driving cycle. This means that we get values for the acceleration a and speed v as stating points. Starting from this the Vehicle block (Figure **??**) determines what speed, acceleration and torque(power) is needed at the wheels. The next block in the model represents the control strategy.

The basic idea behind the strategy is to decide how to divide the needed torque between the prime movers of the vehicle (ICE and EM). In order to find the optimum split (or close to optimum) a genetic algorithm is used.

The following sections will start with the basics about genetic algorithms and continue with the details of the control strategy.

4.1 Genetic Algorithms

Genetic Algorithms are search heuristics based optimization algorithms, for which the foundation was laid by the work of De Jong [6] and Goldberg [1]. The idea behind GA is the natural evolution, meaning that we start from a data set,which represents the initial solution, and evolve (modify) this data set towards a better one. The initial data set or *Population* has a number of initial solutions (*Individuals*) that are randomly generated based on the representation that is chosen (*Initialization*). From the initial population some of the individuals are selected (*Selection*) based on an *Objective Function*. Using the objective function the decision is made which of the individuals are *good*, good in the sense that they represent a better solution compared to the oder individuals. Each individual has a value, or a set of values. In order to evolve to the next *Generation* the values of the selected individuals are combined (*Crossover*) or modified (*Mutation*) ad thus creating a new generation. On the newly created generation the same mechanism for evolving (Selection, Crossover and Mutation) is applied to reach the next generation. The evolution goes on until the stopping criteria is satisfied. One possibility is to stop after a specific number of generations is reached. In the present work the library created by M. Wall is used [12]. In the next sections the mechanisms and operators present in this library will be presented.

The implementation used by Wall describes four flavors of genetic algorithms. The main difference between the flavors is how many, or if at all, form the individuals in the current generation they transfer to the next one. The four types are:

- *Simple GA*(non-overlapping populations). Each new generation creates an entire new population. Only the best individuals are selected form the previous generation for mating.
- *Steady-State GA*(overlapping populations). It uses overlapping populations. This means that individuals form the current population are transferred to the next. The user can specify how much of the old population should be replaced by specifying a percentage or by specifying how many of the old individuals should be replaced(the number must be lower than the population size). The worst individuals are destroyed and the population in repopulated with new individuals until the population size is reached.
- *Incremental GA*(overlapping populations with 1 or 2 children per generation) This strategy also uses overlapping populations but with very little overlap. For the new generation only one or two individuals are selected(the parents). Form the parents there can be one or two children generated(this can be specified by the user). It is also possible to select whom the children should replace in the new generation, possible choices are: the parents, random individuals or the worst individuals.
- *Deme GA*(parallel populations with migration). The genetic algorithm has multiple independent populations that evolve using a steady-state genetic algorithm, but each generation some individuals migrate from one population to the other. Each population migrates a fixed number of it's best individuals to the neighbor. There is a master population which is updated every generation with the best individuals form the other populations. The final solution is represented by the best individuals form the master population.

4.1.1 Evaluation

In order to decide which of the individuals in the current generation to pick for the purpose of evolving the next one, a measure of goodness is needed. In the scope of the GAlib library this is done with the use of two mechanisms: *Objective Function* and *Fitness Scaling.* Although they are a measure of how good an individual is, it is important to distinguish between the objective score and the fitness score.

The Objective function determines the raw score for each individual. The mechanism for giving an individual a specific score is determined by the user. He can decide what is important for an individual, and based on the selected measures to give a greater score if the values of the individuals are satisfactory.

On the other hand, the fitness score is a possibly transformed rating used by the genetic algorithm to determine the fitness of individuals for mating. The fitness score depends on the objective score and usually is obtained by a linear scaling of the raw objective scores. The way the fitness scores are determined can be changed. In the selection step done by the genetic algorithm the fitness scores are used, not the objective scores.

4.1.2 Selection

During the evolution of a genetic algorithm a proportion of the current generation is *selected* for mating in order to generate the next generation. Individual solutions are selected through a fitness-based process, where fitter individuals are more likely to be selected for mating. Different selection methods are available in GAlib:

- Rank Selection. The selection scheme selects the individual with the highest fitness score every time.
- Roulette Wheel Selection. The selector picks an individual based on the magnitude of the fitness score compared to the rest of the population. The individuals with the highest scores are likely to be chosen. The probability p with which an individual is chosen is calculated by dividing the fitness score of the individual by the sum of fitnesses of each individual in the population.
- Tournament Selection. The tournament selector uses the roulette wheel selector to chose two individuals and then picks the one with the highest fitness score. This method of selection yields higher scored individuals more often than the roulette wheel selector.
- Uniform Selection. This selector chooses randomly an individual form the population. Each individual has the probability p, where p is equal to 1 divided by the population size.

4.1.3 Crossover

To derive a new generation first the candidate individuals are selected. The selected individuals can be transferred as they are or modified to the new generation. Before they are put in the new generation a variable is checked if there should be a crossover between the individuals or not. This variable represents the probability for crossover. For this purpose a random number is generated and if the random number is lower than the crossover possibility then the crossover takes place.

Crossover or recombination resembles the natural reproduction and biological crossover. Crossover is the process in which more than a solution is taken(the parents) and a child solution in derived from them. The crossover techniques depend on the data structures that store the individuals information. Although crossover techniques can be implemented by the user there are a couple of techniques that are most common.

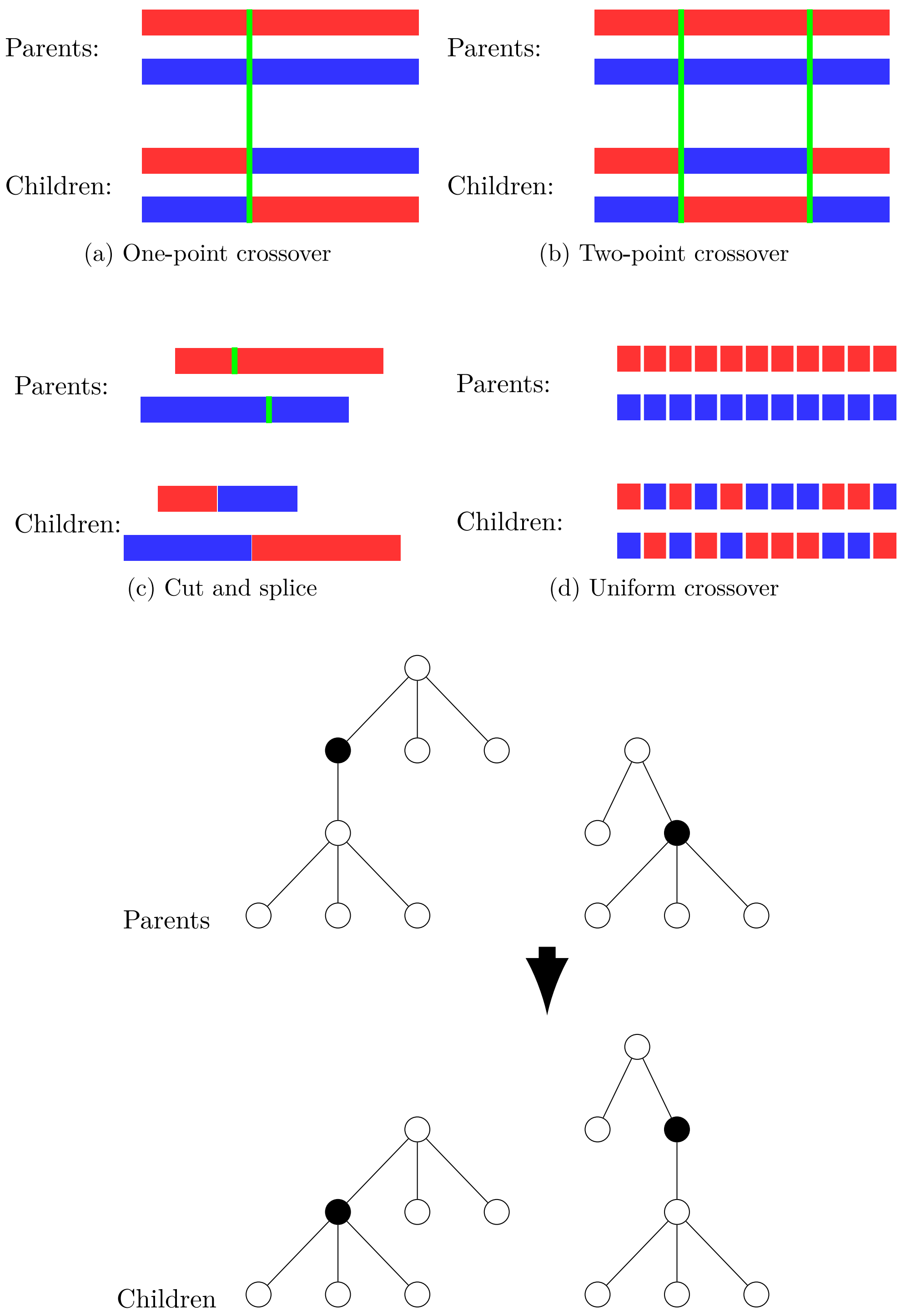
- One-point crossover. In the one-point crossover a random point is selected. This is the crossover point for both parent's organism strings(arrays). Beyond this point the information for the two parents is swapped between them. The resulting individuals are the children (Figure **??**).

- Two-point crossover. As the name suggests two points are selected for the parent's strings. Everything that is between this two points is swapped among the two parents (Figure **??**).

- "Cut and splice". Each parent string has a different choice for the crossover point. The resulting children are of a different length (Figure **??**)

- Uniform crossover. The offspring is generated from the parents by inheriting their values. Which from the two values is inherited is decided every time with a 50-50 probability (Figure **??**).

- Tree one point crossover. Two crossover points are selected in the parent trees, one in each of them. The sub trees starting form these points are swapped between the two parents (Figure **??**).



(a) One-point crossover (b) Two-point crossover

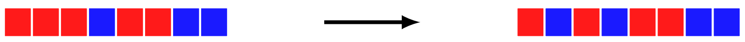
(c) Cut and splice (d) Uniform crossover

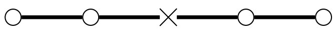
(e) Tree one point crossover

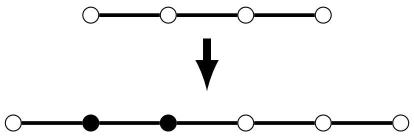
Figure 4.1: Crossover techniques.

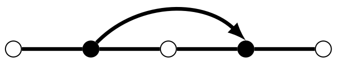
4.1.4 Mutation

After the individuals are selected and the probable recombinations are done, there is another step before the new generation is ready. As in the case of crossover there is a mutation probability p that is set by the user. For each individual a random number is generated and if this number is lower than the mutation probability then the individual will mutate. What type of mutation the individual my suffer depends on the data type and on the user settings.


(a) Bit flip mutation


(b) List destructive mutation


(c) List generative mutation


(d) List node swap mutation

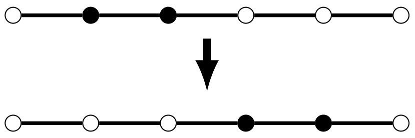

(e) List sequence swap mutation

Figure 4.2: Different mutations

Mutation assures the diversity of the individuals and helps to cover the search spaces that were not covered before. Some of the possible mutations are:

- Flip mutation. This mutation can be applied to individuals that have a binary representation. The mutator has the effect that a bit is flipped, for example, form 0 to 1 or vice versa (Figure **??**).
- List mutations: list destructive mutations, list generative mutations, list node swap mutations and list sequence swap mutations (Figure **??**).
  - The list destructive mutator removes a random node form a list. The length of the list is shortened (Figure **??**).
  - The list generative mutator has an opposite to the destructive mutator. A random node in the list is selected and a random number of nodes are inserted instead of the selected node. The length of the list in increased (Figure **??**).

- List node swap mutators select two random nodes in the list in exchange their places (Figure **??**).
- List sequence swap mutators select a random number of neighboring nodes and change their place in the list based on a random principle. The list's length remains unchanged (Figure **??**).

4.1.5 Stopping criteria

In order for a genetic algorithm to stop the evolution a stopping criteria is needed. This criteria is checked every time before the new generation is evolved. The steps of a genetic algorithm are:

1. Initialization.
2. Selection .
3. Deriving new individuals through combination(crossover) and mutation.
4. Stop when the termination criteria is satisfied.

As long as the termination criteria is not satisfied the first three steps are repeated. Common termination criteria are: a solution is found that satisfies minimum criteria, a fixed number of generations is reached, manual inspection, allocated budget is reached, i.e.,computation time, and others.

4.2 Control Strategy

The control strategy is represented by the decision on how to split the requested power demand between the vehicle's prime movers (ICE and EM). The simulation in Matlab/Simulink, as mentioned in Chapter 3, is represented by a series of small time intervals (time steps) in which all variables don't change their values, they remain constant. It is very important that the time step length is in such a way chosen that the information from the modeled signals. Normally the time step is chosen to be half the value of the smallest time constant of the model. A time constant, from a system point of view, is the time that passes between two changes in the values of a signal.

The control strategy has to decide how much power to request form the ICE and how much power from the EM at each of the time steps of the simulation. For the European standard driving cycle a time step of 1 second is sufficient (Guzzella and Sciarretta) [4]. This means that the control strategy determines a series of local optima. In order to incorporate a more global optimum the control strategy has the ability to access future data within a prediction horizon. Determining the prediction horizon is not in the scope of the present work. Because all the driving data is available form the driving cycle a priori the prediction horizon represents the interval of values for all variables between the current time step and the current position plus a specified amount of time:

$$horizon = [h_c(v1_c, v2_c, ..., vn_c), h_{c+k}(v1_{c+k}, v2_{c+k}, ..., vn_{c+k})] \tag{4.1}$$

h_c - current step time, k - length of the prediction horizon,$v1, v2, ..., vn$ represent the system variables at different step times.

The control strategy has to obey certain restrictions:

1. The consumption should be lower than in the ICE-only mode.
2. Battery should operate in the interval 60% - 90% Soc.
3. The difference between two consecutive splits should be chosen so that a normal operation is assured. What is not desired is for example to have a 90% power split to the ICE and in the next time step (one second later) a 10% power split. A smooth change is required.
4. To use as much as possible the EM in generator mode for braking. This is not always possible, for example in a braking situation where the vehicle has to slow down in a small amount of time.

The decision on how to split the power demand is done using the genetic algorithm.

4.2.1 Incorporating the GA in the model

The genetic algorithm is used to determine in each time step of the simulation the optimum, or close to the optimum, power split between the two engines. During the simulation there are two possibilities: the situation in which we require a positive torque demand (require power from the engines, meaning that we have fuel consumption) and the situation in which we have a negative torque demand,i.e., during braking. There is a third possibility. This is the case that the demanded torque is equal to 0. This situation is not important form the strategy point of view because there is no decision that needs to be made. In this care the inputs for the control strategy are written to the outputs without changing.

Demanded torque is negative

In the second situation(negative torque) we can recuperate part of the energy. How much of the energy is recuperated depends on how fast the vehicle needs to slow down. Based on the braking power T_b we can differentiate between following scenarios:

$T_b > T_b, wmb$. This is the case in which the slowing down of the vehicle can be done without using the mechanical brakes. T_b, wmb - negative torque that can be produced without using mechanical brakes. Usually this is called *engine brake*. In the interval $[T_b, wmb, 0]$ all the energy can be recuperated using the EM as generator. The control strategy checks if the torque is in this interval and directs all the torque to the EM. The split is 100% EM. The ICE is disengaged by having a power demand of 0 Nm. This ensures the operation in the *fuel cutoff regime*, meaning that no fossil fuel is used.

$T_{b,mb} < T_b \leq T_{b,wmb}$. The engine brake is not sufficient to slow down the vehicle. In addition to the engine brake the mechanical brakes are used. $T_{b,mb}$ represents the value of torque for which only the mechanical brakes are used. For all the values in between $T_{b,mb}$ and $T_{b,wmb}$ some of the energy is recuperated. If the value for T_b is closer to $T_{b,wmb}$ then the proportion of recuperated energy from the total braking energy is greater. As the value for T_b approaches $T_{b,mb}$ the proportion of recuperated energy gets smaller until it reaches 0 when $T_b = T_{b,mb}$. For torque values lower than $T_{b,mb}$ only the mechanical brakes are used.

Demanded torque is positive

The required torque is positive, meaning that the vehicle is accelerating. This is the case in which the GA is used to decide how to split the demand. It is possible to have a split that is greater than 100%. All the power that is over

100/

The GA initializes a population of individuals. For the representation of the individuals a real valued genome with alleles was selected. The alleles are the bounds for the values of the individual. Basically each individual is an array of real values that are bounded by the alleles $min_{val} \leq I_{i,j} \leq max_{val}$. $I_{i,j}$ stands for all the values of the individual j with $i = 0, ..., horizon\_length$. In conclusion each individual is represented by an array that has the length of the prediction horizon, ph.

Given a time step h and the length of the prediction horizon ph the GA simulates the prediction horizon by starting from time step h and simulating for ph steps. The values that are used for the simulation steps of the horizon are the values from each individual. Thus the first time step of the prediction horizon has a split represented by the first value of an individual, the second time step has the second value as the split and so on (Equation **??**). The length of the prediction horizon and the array with the values for each individual is the same.

$$split_k = I_k, j \tag{4.2}$$

The split at time step k is the k-th value of the individual j. j represents the number of the individual from the current population, and it ranges form 0 to pop_size - population size. k has values between 0 and ph - length of the prediction horizon.

In Figure **??** the basic idea of how the GA is used in the model is illustrated.

The simulation works in the following way. There are two models that are simulated. The models are the same, only the control strategy is different. For convenience the two models are named A and B. First simulation A is started. This is the main simulation that is used to estimate fuel consumption. From simulation A in every time step simulation B is started to simulate the prediction horizon. The B simulation is started form within the GA. A population is initialized with a fixed number of individuals. The algorithm runs simulation B with the values from the individuals. Based on the objective function, which will be explained in detail in one of the next subsections, the best individuals are selected and the a new generation is generated. For the new generations the process is repeated until a fixed number of generation is reached (stopping criteria). Out of the final generation the best individual is selected, the one with the highest score, and the first of it's values is used as the split for the current time step.

An important fact is that simulation B must have the same starting conditions as the time step for which the prediction simulation is run. This is why in Figure **??** it's suggested that the GA knows what happened in the previous

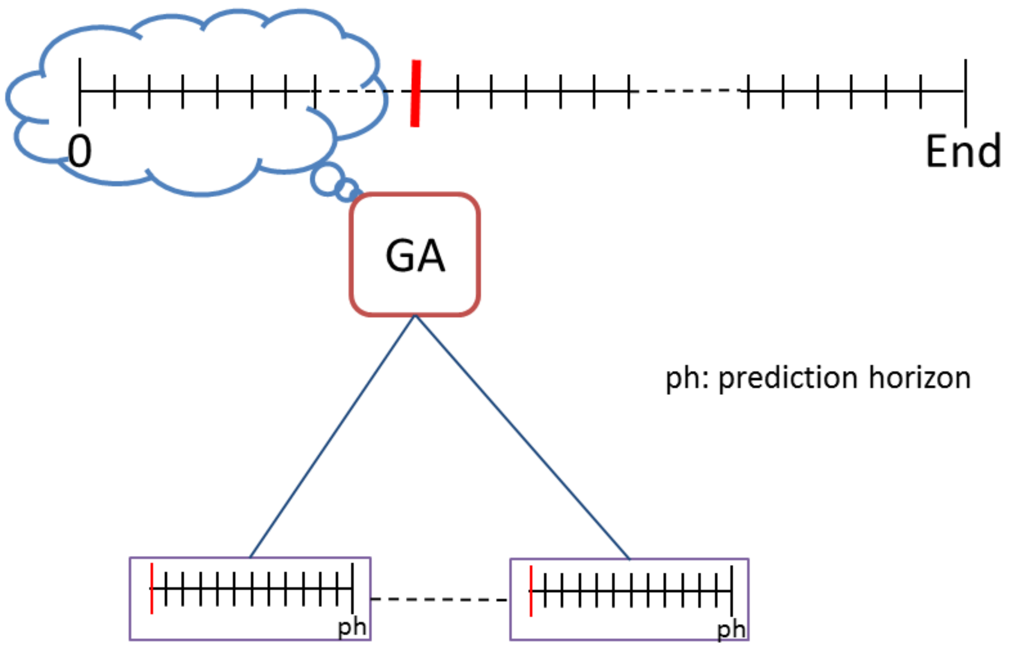


Figure 4.3: Basic idea behind using the GA in the model

time steps. In this figure the upper interval, marked form 0 to END represents simulation A and the ones that are started form the GA are simulation B, which is run multiple times.
The total number that simulation B is started can be calculated as follows:

$$N_{tot} = Nr_{ind} \cdot Nr_{gen} \cdot Nr_{ts} \tag{4.3}$$

N_{tot} - total number of times that simulation B is run, Nr_{ind} - number of individuals in each generation, Nr_{ts} - number of time steps for simulation A minus the length of the prediction horizon ph.

Determining the optimal split in this way is very time consuming. This is why the current solution is used as an off line optimization step.

4.2.2 Modifications to the library

In order to say which individuals are better than other a measure is needed. GAlib offers the possibility to compare the values from the individuals with a value that is known to be good and based on this to give an objective score. This is not the case in the current situation. The "good" value is not known a priori. The solution is to compare the results of the simulation for each individual among themselves. This a so called *population evaluation*.

In the scope of this master thesis a population comparator has been developed. It's purpose is to run the simulation for each individual and after it is finished to store the values for the consumption and state of charge in special fields of each individual. The individuals that are generated using GAlib have the possibility to store, among other things, user specified data. This is the section where the consumption and state of charge are stored.

The genetic algorithm type that is used is the *Simple GA*. In the standard implementation of the library the population comparator is not transfered form one generation to the next. This is another addition that was made to the library in order to satisfy the given conditions. The implementation of the simple GA was extended so that when a new generation is evolved the comparator form the old generation is inherited to the new one.

4.2.3 Determining the objective score

After each time the second simulation (B) finishes the consumption and state of charge are transmitted to the first simulation. Based on these values the objective score of the individual is calculated.

Based on the constraints that are imposed to the control strategy (Section 4.2) the objective score incorporates four components:

1. Consumption - *cons*.
2. State of charge - *soc*.
3. The energy that is stored in the battery - *energy*.
4. The difference between two consecutive splits - *diff*.

Each of the four components is normalized so that is has values in the closed interval $[0, 1]$. Additionally each of the components has a weight assigned to it. Like this it can be decided which out of the four components is more important. Based on the decision of the user the more important component has a higher weighting value. The sum of all weights is 1 (Equation **??**).

$$w_{cons} + w_{soc} + w_{energy} + w_{dif} = 1 \tag{4.4}$$

w_{cons} - weight for consumption component, w_{soc} - weight for state of charge component, w_{energy} - weight for battery energy component, w_{dif} - weight for the difference component.

The objective score for each individual is calculated as a weighted sum of the four components:

$$score = w_{cons} \cdot cons + w_{soc} \cdot soc + w_{energy} \cdot energy + w_{dif} \cdot diff \qquad (4.5)$$

In the following sections each of the four components will be described.

Consumption component

By using the population comparator we have for each individual the consumption at the end of the simulation for the prediction horizon. In a given population there are pop_size different consumptions. In the normal case the individual which has the lowest consumption is the best, but because we have to compare this value to the rest of the objective score's components a normalized value is needed.

The normalization follows the following steps:

- Determine the lowest value for consumption within the current population - determine min.
- Subtract the min from every consumption value. This assures that the lower end of the interval [0,1] is covered.
- After the minimum is subtracted determine the maximum form the resulted values - determine max.
- Divide every consumption value by max. This assures that the greater end of the interval [0,1] is covered.
- To obtain the value $cons$ for each individual subtract the value obtained at the previous point from 1. This assures that the individual with the lowest consumption has the value $cons = 1$ and the one with the highest has the value $cons = 0$.

SoC component

As in the case of the consumption component we has a set of state of charges form the individuals of a population. The approach is different than in the case of the consumption component. The requirement for SoC states that the operating interval for the battery should be in the interval $[60\%, 90\%]$ SoC. This means that we already know what the good values are.

As the SoC approaches the limits of the interval the SoC component should have a lower value. Thus it had been decided to use a normal distribution with the mean at 75% SoC, and a variance $\sigma^2 = 1$. 60% SoC should be at -2σ and 90% SoC at 2σ. This configuration ensures that in the interval [60%,90%]($[-2\sigma, 2\sigma]$) are 95,45% of the values from the distribution.

A normal distribution with it's mean=0 and a variance of $\sigma^2 = 1$, has cumulative distribution function(CDF) value of 0.4. In order to cover the interval [0,1] for the SoC component all the values obtained form the CDF of the distribution are divided by 0.4.

The values of the SoC component are normalized in the following way:

- First the SoC interval of the battery is shifted so that 75% state of charge corespond to 0 on the x axis.
- The values are calculated so that 60% SoC corresponds to -2σ and 90% SoC corresponds to $+2\sigma$.
- The values for the obtained SoC component are divided by 0.4.

After all the steps are completed the *SoC* component is obtained. The schematic for the normalization using a normal distribution is depicted in Figure **??**.

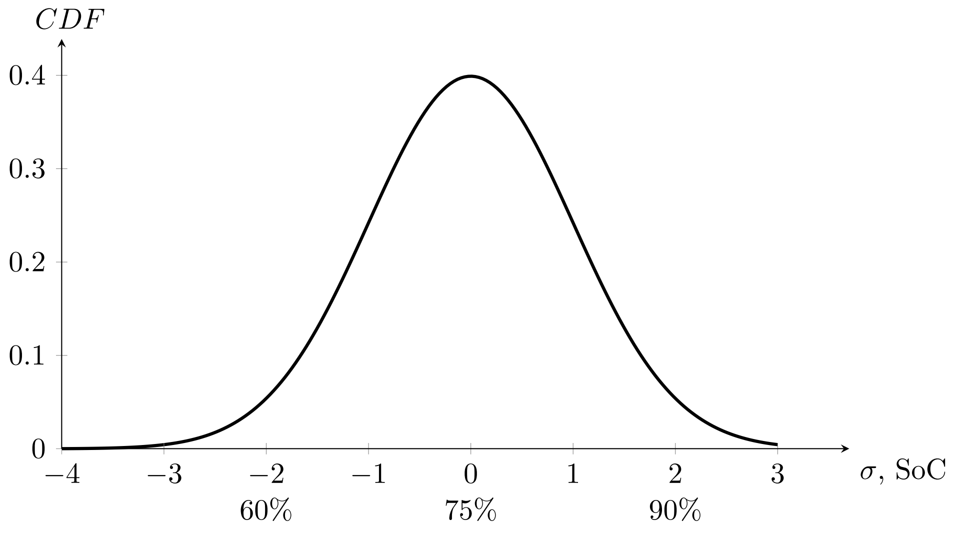


Figure 4.4: SoC normalization using a normal distribution.

Energy component

Incorporating the energy component is important because there can be a situation as depicted in Figure **??**.

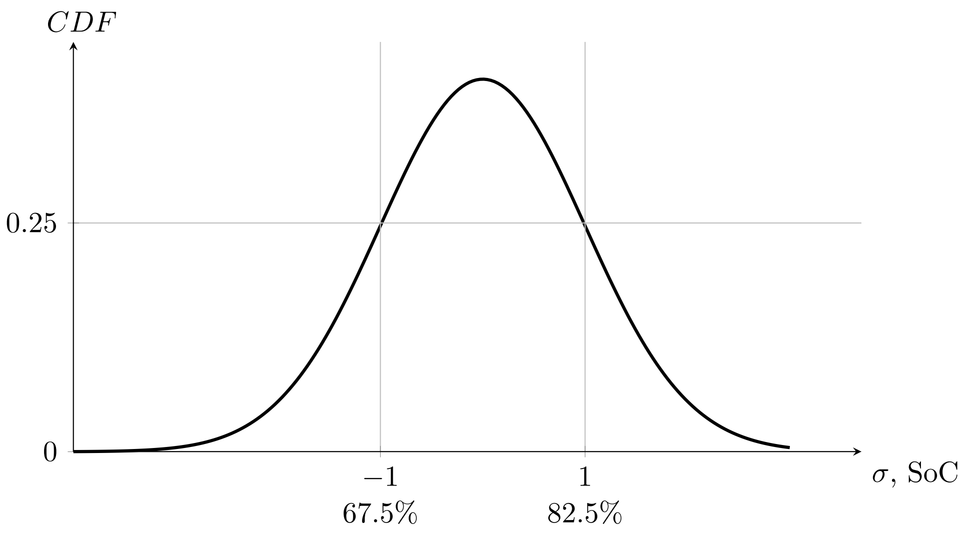


Figure 4.5: Normalized SoC scenario.

This means that for the battery state of charge values of 67.5% and 82.5% the resulting SoC component score is the same(0.25/0.4): $SoC = 0.625$. A higher state of charge value should have a higher score. The *energy* component represents a normalized state of charge of the battery where we take the actual state of charge and divide it by 100. In this manner the *energy* component represents the state of charge of the battery but has values in the interval [0,1] (Figure **??**).

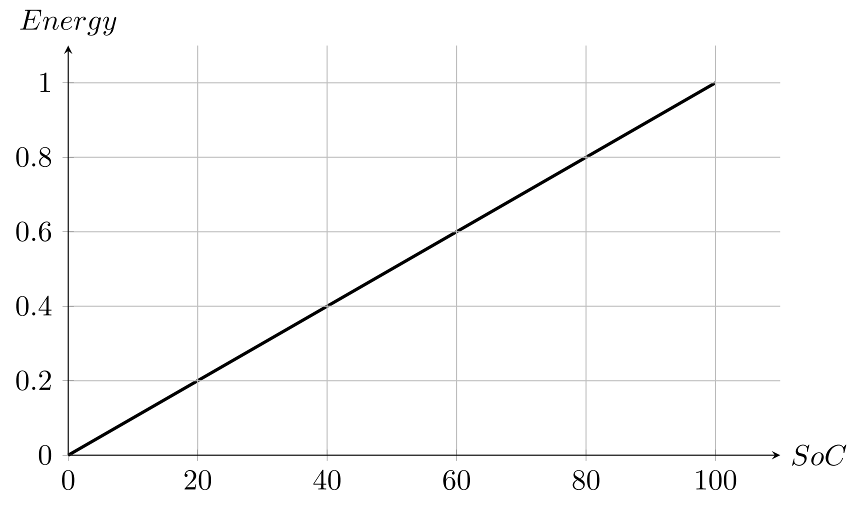


Figure 4.6: *energy* function representation.

Difference component

The difference component is introduced to ensure that there are no big differences between two consecutive splits. The difference between the previous split and the current split is represented by Δ.

$$\Delta = split_{prev} - split_{curr} \tag{4.6}$$

Because the selected GA type is Simple GA we have non-overlapping generations. Only the individual with the highest score is selected for the next generation. The $diff$ component was initially calculated in the following way:

$$diff = 1 - (\Delta / max_{split}) \tag{4.7}$$

where max_{split} is the maximum value that the split can have(for example 130%). The $diff$ component has also values in the interval $[0, 1]$.

Because the fact that only the individual with the highest score was picked it maned that this was the individual with the lowest Δ. If $\Delta = 0$ then the current individual has the same split as it's predecessor. As a result, during the simulation, after a certain point in time, all the individuals form a population were identical.

To solve this problem a *tolerance* variable was introduced. All individuals for which $\Delta \leq tolerance$ had the maximum dif value. The rest of the vales for $diff$ were calculated as before (Equation **??**). In Figure **??** we have the values of the $diff$ component for a *tolerance* of 10 and a maximum split of 130%.

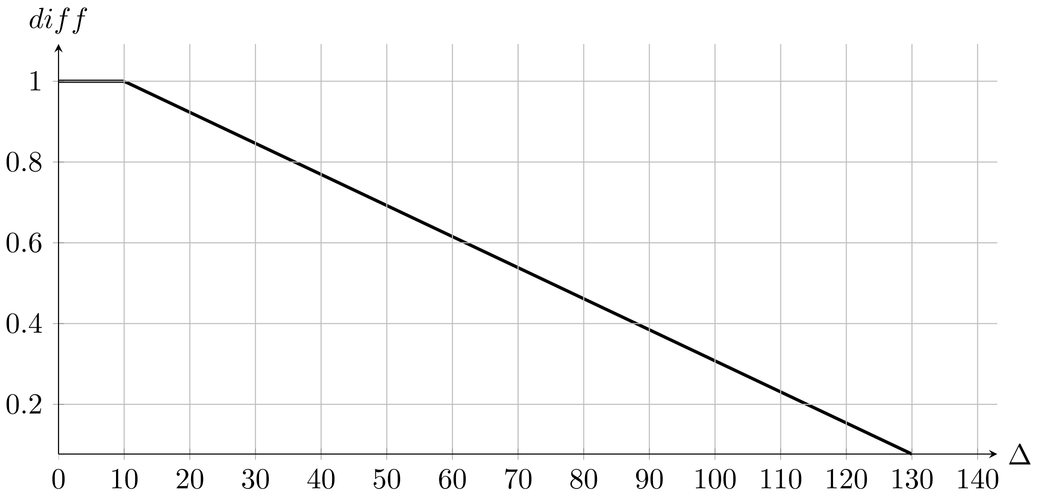


Figure 4.7: $diff$ value schematic for a tolerance of 10 and a maximum split of 130.

The values for *tolerance* and maximum split are used in the above figure just as an example. The values that the GA is using can be set by the user.

4.2.4 Managing the information flow

The Matlab/Simulink simulation environment has two possibilities of transferring variables between two simulations: by using files or by using one of Matlab's workspaces. As in the previous sections the main simulation is named A and the simulation for the prediction horizon is named B. Because we simulate model B very often it is time consuming to transfer variables between the models via files. Every time simulation B has finished the file(s) need to be opened for writing, write the data and the close them. The same operation steps happen when the data is read from model A, first open for reading, read data then close the file.

In the present work the variables are transferred between the simulations by using Matlab's workspaces. Matlab has three built-in workspaces that can be used: *current* workspace, *caller* workspace and the *global* workspace. Because the global workspace is "visible" from every simulation it is used as the space where all the needed variables are stored. When,for example, data form simulation A is needed in B, the variable is written in the global workspace and from there it can be read by simulation B.

5 Experimental results

In this chapter the results of applying the developed control strategy on the real vehicle data are presented. The chosen driving cycle is the standard European cycle. In the first section the parameters for the vehicle are presented. The vehicle parameters remain the same for all test runs. What changes each run are the parameters for the control strategy, which will be described as part of the test run.

5.1 Model parameters

Here are presented the parameters for each of the blocks that are incorporated in the HEV model. The vehicle data is taken form the Toyota Prius III specification, and presented in the following tables.

Vehicle data

5.2 GA configuration one

5.3 GA configuration two

6 Conclusion and future work

Conclusions and future work.

Bibliography

[1] D.E. Goldberg. Genetic algorithms in search, optimization, and machine learning. 1989.

[2] Scotiabank Group. Global auto report. url = http://www.gbm.scotiabank.com/English/bns_econ/bns_auto.pdf, February 2012.

[3] L. Guzzella and A. Amstutz. The qss toolbox manual. *ETH Zürich, http://www. imrt. ethz. ch/research/qss*, 2005.

[4] L. Guzzella and A. Sciarretta. *Vehicle propulsion systems: introduction to modeling and optimization.* Springer Verlag, 2005.

[5] H.L. Husted. A comparative study of the production applications of hybrid electric powertrains. *SAE paper*, pages 01–2307, 2003.

[6] A. Kenneth. *De Jong. An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, PhD thesis, Dept. of Computer and Comm. Sciences, Univ. of Michigan, Ann Arbor, MI, 1975. Univ. Microfilms, 1975.

[7] Mathworks. Matlab. url = http://www.mathworks.com/products/matlab/.

[8] Mathworks. Matlab/simulink. url = http://www.mathworks.com/products/simulink/description1.html.

[9] Mathworks. Matlab/simulink. http://www.mathworks.com/help/simulink/blocklist.html.

[10] Priuswiki.de. url = http://www.priuswiki.de/wiki/.

[11] Toyota. Toyota prius 3 website. url = http://www.toyota.de/cars/new_cars/prius/index.tmex.

[12] M. Wall. Galib: A c++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology*, 87, 1996.

[13] Wardsauto. World vehicle population. url = http://wardsauto.com/ar/world_vehicle_population_110815.

List of Figures