

ON DYNAMIC PROGRAMMING TECHNIQUE APPLIED TO A PARALLEL HYBRID ELECTRIC VEHICLE

FARZAD IRANI

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden

EX049/2009

To my parents who always support me with great love.

Abstract

Lack of fossil fuel supplies as well as greenhouse gases effect on the environment, have motivated car manufacturers to introduce new generations of cars in order to cope with fuel consumption and emissions issues. One of the most interesting structures that is introduced recently to the production lines belongs to hybrid electric vehicles.

A hybrid electric vehicle powertrain, generally, contains an electric energy buffer and an electric machine as well as the conventional internal combustion engine that can work together in several different architectures known as series, parallel and series-parallel depending on how the electric machine is coupled with the internal combustion engine. This extra degree of freedom in the powertrain has raised several different research routes on how to optimize the power split between the electric machine and the conventional internal combustion engine.

The present work presents a Dynamic Programming approach that solves the optimal power split between the internal combustion engine and the electric machine in parallel hybrid electric vehicles in an efficient way, taking minimal fuel consumption considerations into account. The power split must be carried out in such a way that in every moment the demanding power on the final drive is fulfilled by either the internal combustion engine alone, the electric machine alone or both together. Another important characteristic of hybrid electric vehicles is the possibility to regenerate breaking energy that is dissipated in conventional vehicles, by efficiently using the electric machine in generator mode while braking, and storing this energy into the electric buffer for further use. This is also taken into account while designing the optimal controller in the presented work.

This optimal control problem is complicated in the sense that the future driving demands are not known a priori to the controller and hence the decision making is impossible if we treat the problem in a simple way. What can be done to cope with this issue is to divide the problem into two different cases. The first and the most straightforward case is the deterministic case in which the whole driving cycle is known to the controller beforehand, as is considered through the whole thesis. This can be applied efficiently to vehicles that are driven through a specific route many times and have stop-and-go driving cycles such as commuter busses or refuse vehicles. The second case that is much more complicated and can be applied to any vehicle is the stochastic case that contains no predefined driving cycle and instead uses different methods to predict or to estimate the future driving demands depending on the specific technique that is applied. This can easily be realized having the new GPS and GIS equipments in hand, though it is not covered in the presented work.

A Dynamic Programming-based (DP-based) algorithm has been developed as the control system design and applied to the derived vehicle model based on fully deterministic driving profiles. The employed controller shows highly satisfactory reductions in the fuel consumption compared to a simple non-hybrid model in simulations. This algorithm has then been used as the heart of a newly developed toolbox to be working together with QSS Toolbox under Matlab and Simulink environments for much easier further case studies.

KEYWORDS: Hybrid Electric Vehicle, Powertrain Control, Energy Management Strategy, Dynamic Programming, Toolbox.

Acknowledgments

First of all, I would like to acknowledge all the support from my supervisor, Prof. Bo Egardt, who has always been the inspiration to carry on the research with his encouragements and never left me alone either in the very hectic days of the research or in the days that I was completely stuck facing many different problems. Thank you for not letting me get discouraged by always emphasizing the fact that we could handle the problems in the given time and reducing the stress during the whole project period and thank you for all your directions that really made the project easy to deal with. It is pretty fruitful and fun working with you.

Thanks also to Lars Johannesson for all his useful technical consultations and advices and the data he provided me with, Lars Börjesson who helped me a lot in solving computer problems and Madeleine Persson who provided me with many different documents during the project.

Special thanks to Nicolce Murgovski for all his kind guidance to solve programming problems and also for his advices during the coffee time.

I would also like to express my gratitude for the great help from Anna Gharibi in graphic design of the cover image during her very busy time.

Last but definitely not least, I want to thank my parents who always encourage me in life with all their support and love from the bottom of their hearts. This work could not be done without you all.

Göteborg, August 12, 2009

FARZAD IRANI

CONTENTS

1	INTRODUCTION	5
1.1	Main Contributions	6
1.2	Outline	6
2	NOTATION	7
3	Hybrid Electric Vehicle Basics	9
3.1	Definition and Overview	9
3.2	HEV Architectures	10
3.2.1	Series HEV	10
3.2.2	Parallel HEV	11
3.2.3	Series-parallel HEV	12
4	Parallel HEV Powertrain and Chassis Modeling	14
4.1	Chassis model	14
4.2	Powertrain model	16
4.2.1	Battery module	17
4.2.2	Engine	18
4.2.3	Electric machine	18
4.2.4	Constraints	18
5	Dynamic Programming Basics	20
5.1	Principle of Optimality	20
5.2	Optimal Control Using DP	23
5.3	A Simple Example	24
6	DP Approach to HEV Fuel Consumption Optimization	31
6.1	Real HEV Data	31
6.2	Control Algorithm	32

6.3	Simulation Results	32
6.3.1	A Japanese Standard Driving Cycle, 10 - Mode	33
6.3.2	An American Standard Driving Cycle, FTP - HIGHWAY	36
6.3.3	Electric Machinery Sizing	38
7	The Designed Toolbox	41
7.1	Toolbox Hierarchy	41
7.1.1	Main GUI	41
7.1.2	Simulink Model	42
7.1.3	Sub-GUIs	43
7.1.4	DP-based Cost Matrix Calculator	47
7.1.5	System Simulation	48
7.1.6	Plot Results	48
7.2	Communication and Data Sharing	49
8	CONCLUSION AND DISCUSSION	51
	APPENDIX	55
A	QSS Toolbox Review	55
A.1	The quasi-static approach	55
A.2	The elements of the QSS Toolbox	55
A.2.1	Driving Cycle	56
A.2.2	Vehicle	58
A.2.3	Manual Gear Box	59
A.2.4	Combustion Engine (based on consumption map)	60
A.2.5	Electric Motor	61
A.2.6	Battery	62
A.2.7	Tank	64
A.2.8	Controller (empty template)	64

List of Figures

3.1	Series HEV Configuration	11
3.2	Parallel HEV Configuration.....	12
3.3	Power flow in a parallel HEV	12
3.4	Series-parallel HEV	13
4.1	Free-body diagram of a moving vehicle	15
4.2	Parallel HEV architecture. PE stands for Power Electronics.	16
4.3	Simplified battery circuit	18
5.1	Optimal paths in a 3-stage optimization problem	20
5.2	A Multistage decision process	21
5.3	Backward solution to the multistage decision process.....	22
5.4	Resulting trajectory and cost with a sparse grid of state values. The total cost is 0.7921	26
5.5	Resulting trajectory and cost with a dense grid of state values. The total cost is 0.7343	27
5.6	Resulting trajectory and cost with a sparse grid of admissible controls. The total cost is 3.25	28
5.7	Resulting trajectory and cost with a dense grid of admissible controls. The total cost is 1.2358	29
5.8	LQ approach on the same system as that in Figure (5.5). The total cost is improved to 0.6853 as shown in Figure (5.10).	29
5.9	Resulting state trajectory for the system in Figure (5.8).	30
5.10	Optimal control signal for the system shown in Figure (5.8).	30
5.11	Cost value for the system shown in Figure (5.8).	30
6.1	Japanese Standard Driving Cycle, 10 - Mode.....	33
6.2	DP - Based Controller	33
6.3	Propulsion System Operating points with constraints	34
6.4	Torque Split among ICE, EM and Friction Brake	34
6.5	Electric Power flowing through the electric path	34
6.6	Battery State of Charge during the driving cycle	35

6.7	American Standard Driving Cycle, FTP - HIGHWAY	36
6.8	DP - Based Controller	36
6.9	Propulsion System Operating points with constraints	37
6.10	Torque Split among ICE, EM and Friction Brake	37
6.11	Electric Power flowing through the electric path	37
6.12	Battery State of Charge during the driving cycle	38
6.13	DP - Based Controller	39
6.14	Torque Split among ICE, EM and Friction Brake	39
6.15	Battery State of Charge during the driving cycle	40
7.1	Main GUI appearance	42
7.2	Simulink model	43
7.3	Drive Cycle GUI	44
7.4	Vehicle GUI	45
7.5	Gearbox GUI	46
7.6	ICE GUI	46
7.7	EM GUI	47
7.8	Battery GUI	47
7.9	Fuel Tank GUI	48
A.1	The top level of the QSS Library	56
A.2	Schematic presentation of the block "Driving Cycle"	57
A.3	The mask for the driving cycle block	58
A.4	Vehicle block - first level	58
A.5	The mask for the vehicle block	59
A.6	Top level of the block "Manual Gear Box"	60
A.7	The mask for the vehicle block	60
A.8	Top level of the model "Combustion Engine" based on consumption map ..	61
A.9	The mask for the combustion engine based on consumption map block	61
A.10	Top level of the model "Electric Motor"	62
A.11	The mask for the electric motor block	62

A.12 Top level of the model "Battery"	63
A.13 The mask for the block "Battery"	63
A.14 Top level of the model "Tank"	64
A.15 The mask for the block "Tank"	64

1 INTRODUCTION

Hybrid electric vehicles are recently introduced to the automotive mass market as one of the cleanest cars ever due to fuel economy and low greenhouse gases emissions. This fact motivates researchers as well as car companies to think seriously about different ways to approach the problem of reducing the fuel consumption and emissions and this close competition of finding better and better solutions has given rise to many different new ideas, methodologies, papers and patents, so far. As the nature of the problem is to optimize some factors in a complex system, almost all of the suggested techniques are based on optimization techniques.

Previous works that have been done use methods such as Fuzzy Logic as in (Montazeri *et al.* 2008), (Schouten *et al.* 2002) and (Won and Langari 2002), Dynamic Programming as in (Sciaretta and Guzzella 2007) and many other famous methods. Of course these methods can be and have been mixed together with different control methods such as adaptive control, robust control, different predictive approaches, etc., to achieve a much better performance in hybrid electric vehicles depending on different aims and structures.

The motivation behind using Dynamic Programming method in the presented work is its ability to be applied to nonlinear as well as linear systems with or without constraints. This ability that can hardly be found in any other method allows the control engineers to cope with challenging nonlinearities. The method suffers from the curse of dimensionality, making it hard to be applied to complex systems unless using appropriate approximation techniques.

Although using dynamic programming in hybrid electric vehicle fuel consumption optimization is nothing new, in this thesis a new algorithm based on Parallel Hybrid Electric Vehicle (PHEV) architecture has been developed. In this algorithm, the computational burden is taken into considerations and the algorithm is written in a way that imposes as low burden as possible to the computer by designing the controller in an off-line fashion, i.e., calculating the cost matrix for the backwards DP-based controller off-line and then applying this controller to the simulated propulsion system, online. Static memory allocation is also used in different parts to reduce the computational burden.

Further, a need to have a more sophisticated tool than just an algorithm, specialized for carrying out this research parts as well as further investigations on Hybrid Electric Vehicles, has motivated developing a toolbox working in Matlab and Simulink environments in order to be more flexible, accurate and relaxed when carrying out different scenarios and studies on a hybrid electric vehicle. This toolbox is mainly inspired by the QSS Toolbox developed at ETH, Zürich but the designed controller is completely new and is added to the model. Also, a completely new graphical user interface is designed to cover the natural roughness of a piece of programmed algorithm. Since the toolbox uses the powerful Graphical User Interface (GUI) design tool of Matlab, it is pretty user friendly and the user absolutely does not need to know the DP algorithm used behind the toolbox deeply; instead, by just modifying the parameters in a totally graphical environment and pushing a couple of buttons, one can study the effect of different parameters on the fuel economy, as well as doing component sizing and so forth in a user friendly environment.

1.1 Main Contributions

The main contributions of the presented work are as follows:

- An optimized dynamic programming algorithm has been developed to design controllers for energy management in Parallel Hybrid Electric vehicles in predefined routes.
- A useful toolbox has been developed to employ the above mentioned algorithm in a much easier and more flexible way for the researchers to carry out investigations and component sizing in parallel HEVs.

1.2 Outline

The presented material can be outlined as follows:

Chapter 2 introduces the reader to some useful notations and acronyms used throughout this thesis. Chapter 3 is dedicated to describing the basic ideas behind hybrid electric vehicles and explains different possible architectures and their pros and cons for a hybrid electric vehicle. Chapter 4 describes the physical modeling approaches that are used to model a parallel HEV chassis, powertrain and batteries. Chapter 5 spends a couple of pages on explaining the Dynamic Programming method and goes into the principle of optimality as well as basic DP-based optimal control formulation. A simple optimal control problem and its DP-based solution together with some simulation results are given at the end. Chapter 6 presents the main results of applying the developed DP algorithm, modified from the simplistic algorithm, on the real data taken from a real hybrid vehicle. Chapter 7 goes through the developed toolbox and shows the facilities available in it. Chapter 8 is dedicated to a short discussion and concluding remarks on the thesis work and some suggestions for further work. Finally, an appended chapter exposes a short overview on QSS Toolbox and its abilities in vehicle studies to the reader.

2 NOTATION

Acronyms

BDP	backwards dynamic programming
BT	battery
DP	dynamic programming
EB	electric buffer
EM	electric machine
EMS	energy management strategy
FB	friction brake
FC	fuel cell
GB	gearbox
GUI	graphical user interface
HEV	hybrid electric vehicle
ICE	internal combustion engine
LQ	linear quadratic (control)
PPU	primary propulsion unit
RB	regenerative brake
SISO	single input single output
SoC	state of charge
ZEV	zero emission vehicle

Latin

A	area (m^2)
a	vehicle acceleration (m/s^2)
F	force (N)
g	gravitational acceleration (m/s^2)
h	sampling time (sec)
i	electric current (A)
J	cost function (-)
k	time sample (sec)
m	total vehicle mass (kg)
u	input or control signal
x	state variable

Indexed Letters

A_{front}	vehicle front area (m ²)
C_d	drag coefficient (-)
F_a	aerodynamic friction force (N)
F_g	gravitational force in non-horizontal roads (N)
F_r	rolling resistance force (N)
F_t	traction force (N)
P_{dem}	demanding power on wheels (N.m)
P_{elec}	electric power (W)
Q_{BT}	battery total capacity (Ah)
r_{wheel}	wheel radius (m)
$R_{c/d}$	battery charge/discharge internal resistance (Ω)
SoC_k	battery state of charge at sample k, scaled in [0,1] (-)
T_{dem}	demanding torque on wheels (W)
T_{EM}	electric machine torque (N.m)
T_{FB}	friction brake torque (N.m)
T_{ICE}	engine torque (N.m)
T_{RB}	regenerative braking torque (N.m)
V_{oc}	battery open-circuit voltage (V)
ω_{dem}	demanding rotational speed on wheels (rad/s)
ω_{EM}	electric machine rotational speed (rad/s)
ω_{ICE}	engine rotational speed (rad/s)

Greek

θ	road angle
ρ	air density (kg/m ³)
ν	vehicle speed (m/s)

3 Hybrid Electric Vehicle Basics

The very concept of hybrid electric vehicles dates back to 1901 when Ferdinand Porsche at Lohner Coach Factory designed the 'Mixte', a series hybrid vehicle based on his earlier electric vehicle. This hybrid vehicle contained a gasoline engine combined with an electric generator and an electric motor with a small battery for more reliability. This trend grew until the late 60s and early 70s and then turned into an HEV much more similar to the contemporary HEVs when Viktor Wouk installed his prototype hybrid powertrain into the Buick Skylark provided by GM in 1972 and earned the title 'Godfather of the Hybrid'. This turning point ignited the very fast growing progress in HEVs as we can see nowadays and branched in many different aspects such as regenerative braking issues, fuel consumption considerations, emission, battery wear and so on.

3.1 Definition and Overview

Conventional vehicles are propelled by only igniting fossil fuel in a combustion chamber inside and integral to the engine and converting the ignition energy into mechanical rotation and translation. Hence, it is only the expansion of high pressure and temperature gases that applies force to the movable components. In contrast, HEVs are characterized by using some 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 (Guzzella and Sciarretta 2007). In addition to the above mentioned components, at least one electric motor is also necessary in any HEV to help propel the vehicle either fully or partially. This combination of electric and fossil fuel energy, supervised with a high level controller called Energy Management Strategy (EMS), can improve the performance of the vehicle from a fuel consumption and emission point of view. Comparing HEVs with conventional vehicles shows that the former is more fuel efficient due to the engine operation optimization and the possibility of recovering the kinetic energy during braking (Chan 2007).

The electric motor(s) play the role of optimizing the efficiency of the ICE as well as energy recovery during braking or coasting. It can also use the excess power of the engine to charge the battery if the power demands on the final drive is lower than the power converted by the engine. Another role can be to assist the ICE in the cases that the ICE alone cannot fulfill the driver's demands.

There are basically four main advantages in hybridizing a conventional vehicle as follows:

- **Engine Downsizing:** When we use both an electric motor and an ICE in one single vehicle, it is possible to size the engine for the mean instead of the peak power demand. In such a way, the electric buffer can compensate the lack of power in high power demand periods of driving via the electric motor. Hence, having a smaller ICE, the vehicle can be driven more efficiently in normal driving compared to having a large ICE.

- **Regenerative Braking possibility:** The energy that is dissipated in conventional vehicles during braking can be regenerated and stored in the electric buffer using the electric machine in its generator mode.
- **Pure Electric Drive:** Having an Electric Machine (EM) together with the ICE has also the advantage of letting the ICE shut down during the vehicle low speeds in such a way that the controller shuts the engine down and makes the EM propel the vehicle at a low speed. This can also reduce both fuel consumption and emissions to a large extent.
- **Improved control of the ICE:** Since in HEVs the propulsion power demand is a mix of power from the PPU and the EB, control of the ICE operating point, i.e., engine torque and speed, can be carried out with higher degree of freedom compared to a conventional vehicle and depending on which type of HEV we have at hand. In addition, having an EM in the propulsion system responding to quick changes in propulsion power demand due to its smaller time constant compared to that of the ICE, gives the possibility to avoid transient ICE utilization.

Although HEVs have many advantages, they have some limitations as well. The first issue is the increased cost due to the presence of electric motor, energy storage system, electric converters and so on. Safety issues due to existence of high voltage electricity and electromagnetic interference due to high frequency switching circuits are also other problems in HEVs.

3.2 HEV Architectures

As mentioned before, any hybrid electric vehicle contains at least two sources of energy. These two sources can be coupled together in several different combinations according to the application. The most famous configurations are called series, parallel, series-parallel and heavy HEV (Chan 2007). The latter is though not a separate category but either a series or a parallel hybrid that is used in heavy delivery vehicles and can run on gasoline or diesel. The following is a brief description of each configuration.

3.2.1 Series HEV

Series powertrain architecture is used in large vehicles such as locomotives and heavy duty trucks and not in passenger vehicles according to efficiency problems due to their component inefficiencies (Miller 2006). In this configuration, the ICE is not directly connected to the final transmission but via two electric machines. In such architecture, the engine mechanical output is first converted into electric energy using an electric generator and then this electric energy can either charge the battery when needed or bypass the buffer and propel the vehicle via a separate electric motor or a combination of these as is shown in Figure (3.1). Regenerative braking is also possible using the traction motor in generator mode and storing the electric energy into the buffer. So a series HEV needs three machines: one engine, one electric generator and one electric traction motor.

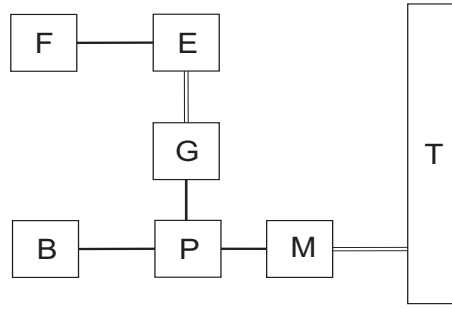


Figure 3.1. Series HEV Configuration

B: Battery
 E: ICE
 F: Fuel tank
 G: Generator
 M: Motor
 P: Power converter
 T: Transmission(including brakes, clutches and gears)

3.2.2 Parallel HEV

Since the model that is used in this thesis is a parallel hybrid, it is described here in more details. In parallel hybrid electric vehicles the propulsion energy is delivered to the final drive by the EM and the ICE in two separate parallel paths. The ICE is though the primary propulsion unit and the EM will be considered as an assistant to the PPU. As shown in Figure (3.2), both the ICE and EM are coupled to the final transmission via mechanical links in parallel HEVs. Also it is worth mentioning that each energy flow path can have a separate clutch to engage/disengage the final transmission from either the ICE and/or the EM. Hence, six different operating modes are possible for a parallel HEV as follows:

1. Motor assist mode: Power from both the ICE and EM are mixed together to provide the final drive power request. This can be the case in peak power demands. See Figure (3.3) A.
2. Regenerative braking mode: In this mode, the braking energy can be converted into electric energy, using the traction motor working in generator mode, and be stored into the electric buffer via the lower path. See Figure (3.3) B.
3. Power split mode: This is the case when the ICE delivers more power than the power demand on final drive and the electric buffer is somehow empty (better say, its charge is close to its lower limit). So, ICE power is split in such a way that it drives the vehicle and the excess power charges the EB at the same time. In this mode, the electric machine works in generator mode. See Figure (3.3) C.
4. Motor alone (ZEV) mode: In this mode, the ICE is turned off and the vehicle is fully powered by the EM. In this case, no fuel is used and hence there is no emission.

This situation can happen mostly in idling or low power demand driving cases. See Figure (3.3) D.

5. ICE alone mode: In ICE alone mode, the vehicle is propelled by the ICE and the EM is switched off via an electric circuit. See Figure (3.3) E.
6. Stationary charging mode: In this mode, the vehicle is at standstill, all machinery are off and the vehicle is plugged into the power outlet.

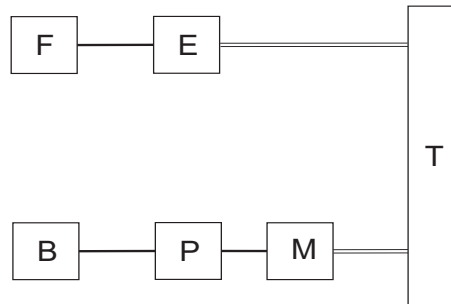


Figure 3.2. Parallel HEV Configuration

As can be seen in Figure (3.2), one single electric machine can be used in parallel HEVs, working either as a motor or an electric generator. This results in a decrease in expenses as well as in total vehicle mass that can prevent an excess in fuel consumption due to the vehicle weight. Different operating modes of a parallel HEV are shown in Figure (3.3). Note that the arrowheads show the direction of power flow in each operating mode of the vehicle powertrain.

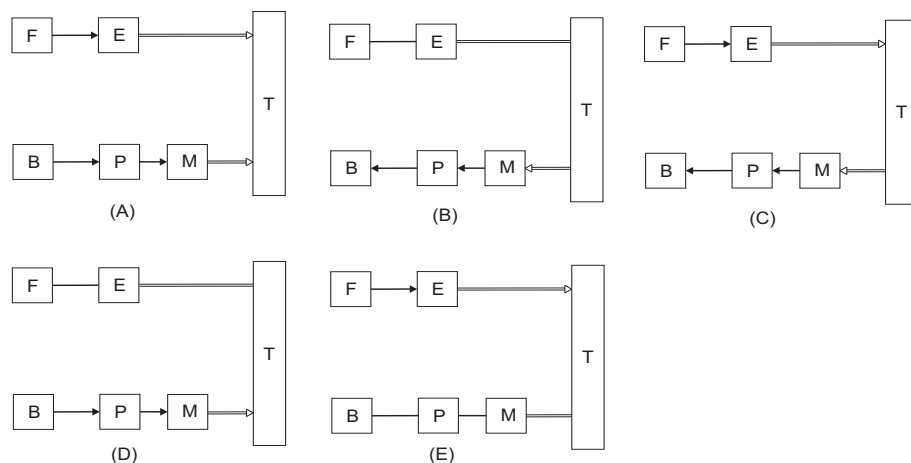


Figure 3.3. Power flow in a parallel HEV

3.2.3 Series-parallel HEV

As shown in Figure (3.4) a third configuration of HEVs can be realized by just adding another power flow path to the parallel architecture having the features of both series and

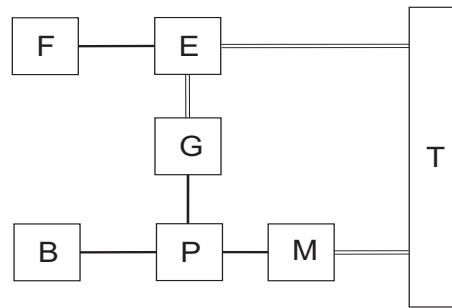


Figure 3.4. Series-parallel HEV

parallel HEV. In this configuration, an extra electric machine is added to the system via a mechanical and an electrical link so that all advantages of series and parallel hybrids can be kept together in one single vehicle. The disadvantage of series-parallel HEVs is though their complexity and costly structure, though they are still preferred to be employed in some special cases.

4 Parallel HEV Powertrain and Chassis Modeling

In this chapter, the chassis and powertrain models that are used in this work are presented. Chassis model is the model of a parallel HEV based on the Newton's second law. Powertrain model is based on the demanding power on vehicle wheels and how this power is delivered to the wheels from the prime movers. These models are then used in backwards simulations in the sense that the driving profile is given a priori and based on this profile and the vehicle specifications, the fuel consumption will be calculated and optimized by the energy management strategy.

4.1 Chassis model

To derive the longitudinal dynamics of a vehicle on a road one can use the Newton's second law as follows:

$$ma = F_t - \frac{1}{2}\rho\nu^2 C_d A_{front} - mg \sin(\theta) - f_r mg \cos(\theta) \quad (4.1)$$

where m is the vehicle total mass, a is the vehicle acceleration, F_t is the traction force from the prime mover(s) minus the force used for accelerating the rotating parts minus all friction losses in the powertrain, ρ is the ambient air density, ν is the vehicle speed or the speed profile, C_d is drag coefficient, A_{front} is the vehicle front area, g is the gravitational acceleration, θ is the road angle and f_r is rolling resistance. Note also that no disturbance is taken into account in this formulation. Note that in calculating the Equation (4.1), a reversal in causality has been considered while modeling the chassis. This means that in contrast to the real world that the speed and acceleration of a vehicle are effects of the traction force, in this work, the road profile, vehicle speed and consequently acceleration profile are causes of the traction force. In other words, having speed, acceleration and road profile, the demanded force, torque and power are calculated. Another important issue to be pointed out is using a quasi-static approach in modeling the vehicle, i.e., considering fast dynamics to be static for simplicity reasons.

In order to be able to make a comparison between the Equation (4.1) and the Figure(4.1) let's name every component of the above mentioned force as the following:

- Aerodynamic friction: $F_a = \frac{1}{2}\rho\nu^2 C_d A_{front}$
- Rolling friction: $F_r = f_r mg \cos(\theta)$
- Gravitational force in non-horizontal roads: $F_g = mg \sin(\theta)$

Note that in Equation (4.1) the aerodynamic friction force is calculated considering the vehicle to be approximately a prismatic body with the front area A_f . A_f is an approximate

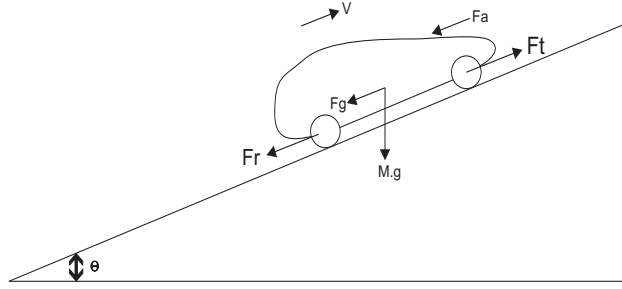


Figure 4.1. Free-body diagram of a moving vehicle

for car body effect, wheel housings effect, side mirror effect, window housings effect, engine ventilation, antennas etc (Guzzella and Sciarretta 2007).

Another simplification is to consider the drag coefficient, C_d , as a constant regardless of the vehicle speed. This parameter is usually estimated using CFD programs or by experiments in wind tunnel (Guzzella and Sciarretta 2007).

The other important component of the Equation (4.1) is the rolling resistance, f_r , that is also considered to be constant for simplicity reasons. This parameter is in fact a function of vehicle speed, tires pressure and road surface. For moderate speed variations it is a reasonable approximation to consider f_r as constant though.

Having calculated the traction force, F_t , and having the speed profile at hand, it is pretty straightforward to calculate the speed, torque and power demand on the final drive using the Equations (4.2) through (4.4) :

$$T_{dem} = F_t / r_{wheel} \quad (4.2)$$

in which r_{wheel} is the average wheel radius.

$$\omega_{dem} = \nu / r_{wheel} \quad (4.3)$$

$$P_{dem} = T_{dem} \omega_{dem} \quad (4.4)$$

with efficiency of the final drive to be equal to 1.

The power demand on the final drive can be either positive or negative depending on the driving conditions. A positive power demand on the final drive means a power request from prime mover(s) and a negative one means an extra power that can be regenerated and saved in an EB as described in Chapter 3. The lower limitation on P_{demand} due to avoid locking the wheels is not taken into account in this work.

4.2 Powertrain model

The powertrain model that is used in the presented work is as mentioned before the model of a parallel hybrid electric vehicle as shown in Figure (4.2). As can be seen, the architecture is in such a way that having the clutch engaged, there will be the same speed on both ICE and EM. With this architecture as we saw in Section 3.2.2 there are six different possible operating modes. These six operating modes are taken into account when modeling the powertrain.

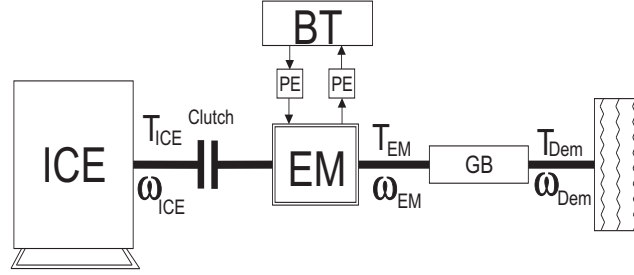


Figure 4.2. Parallel HEV architecture. PE stands for Power Electronics.

The aim of the powertrain is to deliver the demanded power, P_{demand} , to the wheels via its two paths, i.e., the engine path and the electric machine path, to absorb the excess energy from the internal combustion engine and save it in the electric buffer and finally to convert the extra braking energy via the electric machine working in generator mode to electric energy and save it into the battery.

The mechanical power is transferred from/to the final drive via a 5-stepped gearbox and a differential gear and the gearbox efficiency is considered to be the same for all different gears. There is only one clutch in the powertrain that is located between ICE and EM. As can be seen in Fig (4.2), there are two separate electric paths between the electric machine and the electric buffer (battery in this case), that are not modelled here and are simplified to have only switching behavior to switch according to the electric machine operating region.

The signals T_{dem} and ω_{dem} are inputs to the controller coming from powertrain and T_{ICE} , ω_{ICE} , T_{EM} and ω_{EM} are outputs from the controller to the powertrain. Note that according to the proposed powertrain architecture, ω_{ICE} and ω_{EM} are equal as long as the clutch is engaged. The engine efficiency is given by linear interpolation in a static engine efficiency map and the fuel mass rate, $c(\omega_{ICE}, T_{ICE})$, will be determined having the engine efficiency, engine speed and engine torque at every sample using the interpolation in a fuel mass rate map. There is no limitation on gear shifts and the gear changes are calculated off-line by the controller taking into account a known drive cycle and road profile. T_{dem} is defined to be positive when power is delivered to the final drive and negative when power can be absorbed from the final drive. Taking all these conventions into account and using Equations (4.5) through (4.7) the torque split problem can be handled:

$$\omega_{EM} = r_g \omega_{dem} \quad (4.5)$$

$$T_{ICE} - T_{EM} = \frac{\eta_g T_{dem}}{r_g}, T_{dem} \leq 0 \quad (4.6)$$

$$T_{ICE} - T_{EM} = \frac{T_{dem}}{\eta_g r_g}, T_{dem} > 0 \quad (4.7)$$

where η_g and r_g represent the gearbox mechanical efficiency and gear ratio at the selected gear, respectively. Note that T_{EM} is defined as negative in generator mode.

4.2.1 Battery module

The battery module is modelled simply with a resistive circuit and the open circuit voltage, V_{oc} , is dependent on the SoC of the battery at every moment. The battery resistance, $R_{c/d}$, depends on if the battery is in charge or discharge mode and is calculated at every sample from linear interpolation in a static charge/discharge vector. Hence, the next state of charge of the battery in every sample can be determined given the electric machine torque, the electric machine speed, the battery total capacity, and the current state of charge as shown in Equation (4.8) :

$$SoC_{k+1} = SoC_k + \frac{hi(T_{EM}, \omega_{EM})}{Q_{BT}} \quad (4.8)$$

where h is the time step (sampling time in seconds in this case), i is the battery current that can be either positive or negative, and Q_{BT} is the battery total physical capacity. All signals are considered to be stationary during the period h . Figure (4.3) shows the simplified model of the battery.

Battery current can also be calculated at every sample using Equation (4.9) and linear interpolation in the battery open circuit voltage vector as well as battery charge/discharge resistance vectors based on the current state of charge of the battery.

$$i = \frac{V_{oc}(SoC)}{2R_{c/d}} + \sqrt{\frac{V_{oc}^2(SoC)}{2R_{c/d}} + \frac{P_{elec}(SoC)}{R_{c/d}}} \quad (4.9)$$

in which P_{elec} represents the electric consumption/generation of the electric machine at every sample. In fact, P_{elec} is a function of ω_{ICE} , T_{ICE} and the SoC of the battery.

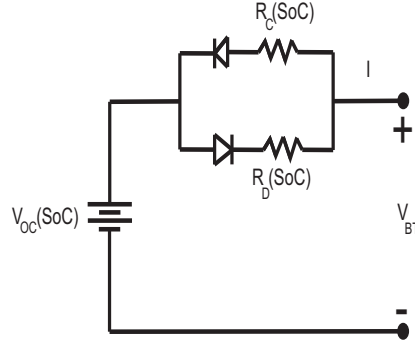


Figure 4.3. Simplified battery circuit

4.2.2 Engine

The internal combustion engine model that is utilized in this work is primarily a simplified model of Atkinson 43 kW ICE (Johannesson 2006), though it can be replaced with any other engine as long as the constraints be taken into consideration in the dynamic programming algorithm. The simplification is done in such a way that the fuel mass rate is considered to be only a function of engine speed and torque and can be calculated at each sample by linear interpolation in a static fuel mass rate map. The engine efficiency is also considered when interpolating in the fuel mass rate map. The maximum torque that the engine can deliver to the engine flywheel is considered to be dependent only on the engine speed and this can be calculated from linear interpolation in a one dimensional map. Below a certain speed, i.e, the idling speed the engine is considered to be in idling mode and a small value of fuel consumption is taken into account.

4.2.3 Electric machine

The electric machine can work in either motor or generator regions depending on the direction of the torque on its shaft. The model that is used here is a simplified model of a 20 kW machine with its efficiency considered to be dependent only on the electric machine torque and speed. This can be calculated again with a quasi-static approach in terms of linear interpolation in an efficiency map of the electric machine that is used in Toyota Prius I. The maximum and minimum torque that can be delivered to/regenerated from the electric machine is also simplified to be solely dependent on the electric machine speed and not on the cooling conditions and temperature of the EM (Krause *et al.* 2002). The electric machine model can also be replaced with any other model as long as the constraints be taken into consideration in the dynamic programming algorithm.

4.2.4 Constraints

There are also a couple of constraints that should be taken care of while dealing with the powertrain model as follows:

- The battery power is limited according to the battery type as is shown in (4.10a).
- The SoC is limited according to (4.10b). Note that this is a physical limitation on the battery showing "Empty" and "Fully Charged" states for the battery.
- The internal combustion engine power is limited according to (4.10c).
- The internal combustion engine torque and the electric machine torque are also constrained according to (4.10d) and (4.10e), respectively.

$$p_{BT} \in [p_{BT,min}, p_{BT,max}], \quad (4.10a)$$

$$SoC \in [0, 1], \quad (4.10b)$$

$$p_{ICE} \leq p_{ICE,max}, \quad (4.10c)$$

$$T_{ICE}(\omega_{ICE}) \leq T_{ICE,max}(\omega_{ICE}), \quad (4.10d)$$

$$T_{EM}(\omega_{ICE}) \in [T_{EM,min}(\omega_{ICE}), T_{EM,max}(\omega_{ICE})], \quad (4.10e)$$

5 Dynamic Programming Basics

A very powerful method for input shaping and trajectory optimization of the systems that can be broken into several stages is *dynamic programming* developed by Richard Bellman (Luus 2000). The very essence of this technique is based on the principle of optimality. Generally speaking, having a dynamical process and the corresponding performance function, there are two ways to approach the optimal solution to this problem; one is the *Pontryagin maximum principle* and the other is Bellman's dynamic programming (Naidu and Naidu 2002). The latter is the focus of this chapter and has the advantage of being applicable to both linear and nonlinear systems as well as constrained and unconstrained problems. Having such powerful benefits, it also suffers from an important disadvantage called *curse of dimensionality*. This means that if the dimension of the problem grows, in sense of the number of states, e.g., the computational burden of this method grows drastically in such a way that it might even not be implementable on today's computers. This limitation keeps dynamic programming away from complicated systems and only applicable to some simple optimal control problems.

5.1 Principle of Optimality

The principle of optimality can be stated as follows:

An optimal control strategy has the property that at any state along the optimal path, the remaining path must be optimal with respect to that state, regardless how it was reached.

Many systems in the world are in the form of individual stages or can be broken into some sub-systems that each one can be considered as a single or multi-stage system. This quality can give rise to the idea of using the principle of optimality to optimize such problems by breaking a complex system into a number of sub-systems and solve the optimal control problem for each sub-system individually while still getting to the global optimal solution, if there is any. Note that the existence of an optimal solution is not of any concern in this thesis and is just taken for granted here. Since optimal control problems involving optimization over a trajectory, as that of a hybrid vehicle fuel consumption optimization, can be broken into a sequence of time stages, it turned out that dynamic programming can be a good solution to these family of problems.

To illustrate this, consider a simple 3-stage optimization problem as shown in Figure (5.1).

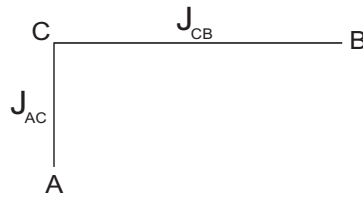


Figure 5.1. Optimal paths in a 3-stage optimization problem

If the optimal path to go from A to B passes through C with the partial costs J_{AC} and

J_{CB} and with the total optimal cost of (5.1), then the optimal cost to go from C to B will be J_{CB} .

$$J_{AB} = J_{AC} + J_{CB} \quad (5.1)$$

In other words, having the optimal cost to go from C to B one can find the optimal cost to go from A to B by just adding the cost to go from A to C to the optimal cost to go from C to B and this can easily be expanded to larger problems with many stages.

As can be seen in Figure (5.1), the whole problem can be broken into two sub-problems and having the optimal cost to go from A to C on the entire route AB, we can conclude that J_{CB} is the optimal cost of the remaining segment CB. In other words, if one can find the optimal solution for each sub-system, it will be possible to find the optimal solution to the whole complex system having all sub-optimal solutions at hand.

There are basically two ways to approach these types of problems called Backward solution and Forward solution depending on which starting point we choose to start our calculations from. In this thesis, the backward approach is applied to the hybrid vehicle fuel consumption optimization problem and hence it will be illustrated here.

Let's take a look at a larger example. Consider a more complex system that can be broken into several stages as shown in Figure (5.2) (Naidu and Naidu 2002). This can be for example the different paths to go from Washington to Texas for an aircraft and the costs can be the fuel consumption for each single flight between any two states. The optimization problem will then be to find the best route to go from WA to TX considering the lowest possible fuel consumption. Note that to go from WA to TX only movements to the right are allowed. The backward approach to solve the optimization problem works as follows:

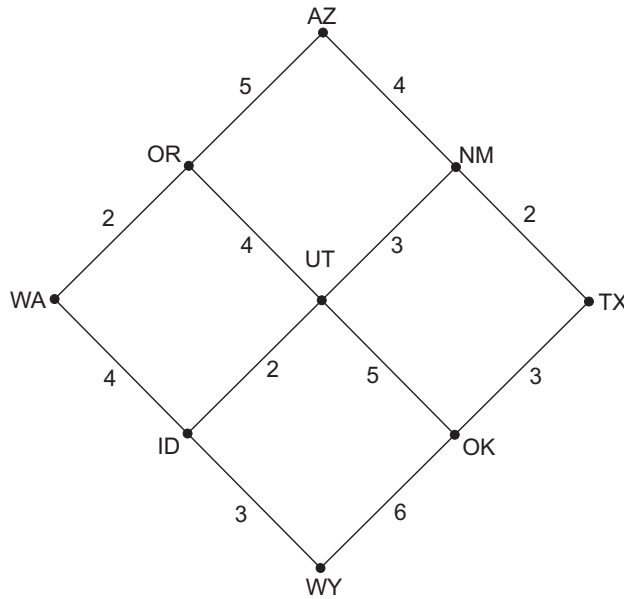


Figure 5.2. A Multistage decision process

Consider WA as system's first stage and TX as the last stage. Now if we start from TX,

go backwards and check different paths' costs until we reach WA and find the optimal path to go backwards from TX to WA, we actually find the total optimal solution using backward method.

We have 5 stages here from $k = 0$ to $k = k_f = 4$. The backward dynamic programming solution is shown in Figure (5.3). In this figure, the numbers on each connecting path show the cost involved to go from each state to the other one via the corresponding route.

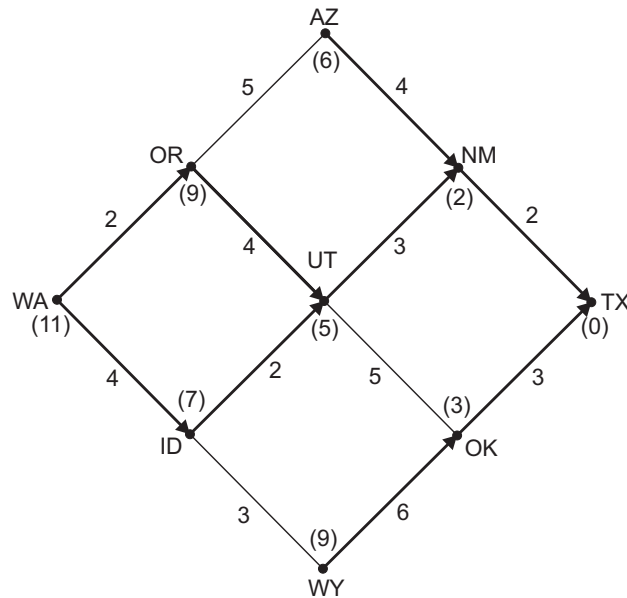


Figure 5.3. Backward solution to the multistage decision process

Stage 5: $k = k_f = 4$

The starting point; no cost involved (Total cost to go from TX to TX will be 0 as is written under TX).

Stage 4: $k = 3$

At this stage, there are two states to fly from; NM and OK and to go from either of these states to TX, there is only one route to choose from, i.e., NM,TX or OK,TX. Since there is no other option both routes will be considered as partially optimal for now. This is shown with bold lines between NM and TX and also between OK and TX. The resulting cost to go from NM will be $2+0$ while that of starting from OK will be $3+0$. These costs are also written under the corresponding states in Figure (5.3).

Stage 3: $k = 2$

At stage 3, we have the possibilities to fly from AZ, UT and WY to TX through NM and OK. Let's explore each possibility individually. Consider AZ first; the only possible route is AZ,NM,TX and the cost will be $4+2=6$. So again this route is bolded and the only possible cost is written under AZ. From UT we have two different options; one going through NM and the other passing through OK. The former costs $2+3=5$ and the latter costs $3+5=8$. Hence the optimal route will be UT,NM,TX with the cost equal to 5 that is written under UT. From WY though we have only one option of going through WY,OK,TX with the cost equal to $6+3=9$. Hence the corresponding route is bolded and the involved cost is written under WY.

Stage 2: $k = 1$

With the same procedure as described above, the optimal cost to go from OR to TX will be 9 through the route OR,UT,NM,TX and from ID to TX will be 7 through the route ID,UT,NM,TX.

Stage 1: $k = 0$

Here from WA there will be two options with the same total cost 11. Hence, we conclude that both paths WA,OR,UT,NM,TX and WA,ID,UT,NM,TX are optimal paths with the same total cost 11. Such problems are categorized under the so called "Shortest Path Problems" (Bertsekas 2000).

5.2 Optimal Control Using DP

Consider a system described by the following discrete time model:

$$x(k+1) = f(x(k), u(k)) \quad (5.2)$$

where k is the discrete time index, x is the state vector, u is the input vector and f is a function (generally, a nonlinear function) of both the state and input vectors.

The objective is then to find an optimal controller to generate the best control sequence $\{u(k)\}$, using the principle of optimality, that minimizes the following cost function:

$$J_0(x(0)) = J = S(x(k_f), k_f) + \sum_{k=0}^{k_f-1} V(x(k), u(k)) \quad (5.3)$$

in which J represents the cost function value that has to be minimized, the first term is the cost function over the final state and the second term is the sum of cost to go from every stage to the next stage as a function of the current state and control. So we have to find the optimal control sequence $\{u^*(k)\}$ which is applied to the system described by (5.2) and minimizes the cost function (5.3) over the optimal trajectory $x^*(k)$. Assume that we have evaluated the optimal control, state and cost for all values starting from $k+1$ to k_f , at any time or stage k ; then using the principle of optimality we can write (Naidu and Naidu 2002):

$$J_k^*(x(k)) = \min_{u(k)} [V[x(k), u(k)] + J_{k+1}^*(x^*(k+1))] \quad (5.4)$$

This is called *functional equation of dynamic programming* and means that if one had found the optimal control, state and cost from any stage $k+1$ to the final stage k_f , then one can find the optimal values for any stage k to the end stage k_f .

Note that for a continuous time system we must first convert the system into a discrete time system using some sampling method and then apply the dynamic programming technique into the equivalent discrete time model of the system.

Note also that another disadvantage of dynamic programming method pops up when it comes to nonlinear systems. In such cases, the interpolation between grid points sometimes does not exactly reach a grid point in the next stage. This is called *menace of the expanding grid* (Luus 2000).

5.3 A Simple Example

As the starting point, consider the discrete time model of a dynamic system described as follows:

$$x(k+1) = -ax(k) + u(k); a \in \{-1, -0.5\} \quad (5.5)$$

For the sake of simplicity, the number of discrete states is considered to be one and the system is considered to be SISO. Note that one system is marginally stable due to having the state matrix equal to -1 and the other one is asymptotically stable. The cost function to be minimized is in the following form:

$$J = \frac{1}{2}x^2(k_f) + \frac{1}{2} \sum_{k=0}^{k_f-1} (x^2(k) + u^2(k)) \quad (5.6)$$

where k_f is the final time for the DP algorithm, the first term is the cost function over the final state and the second term is the total cost to go through the rest of the states via the control sequence.

The example is based on the following assumptions:

- Admissible state values are considered to be in the interval $[-1, 2]$. The DP algorithm is tested on two different numbers of admissible states, one in a sparse grid (5 different values) and the other in a dense grid (1001 linearly spaced discrete state values). The resulting total cost turned to be almost the same and raised the idea that a rather sparse grid of states can work perfectly and there is no need of too many state values that can cause computational burdens. See Figures (5.4) and (5.5) for comparisons. Note that this is done on an asymptotically stable version of the system with the state matrix equal to -0.5 to get rid of the oscillations. It will be discussed in the next figures.

- Admissible control sequence is considered to be picked from the interval $[-1,1]$. To make a comparison, a sparse set of admissible controls is chosen (containing 5 different values) and resulting trajectories and cost value are compared with those of a much denser set of admissible controls (1001 linearly spaced values). The result was that a sparse set of admissible controls can cause oscillations in the state value due to the fact that the control signal reaches zero before settling the state value to the final value (zero in this case) and since the system is inherently marginally stable, it starts oscillating around the final value in the absence of control signal. See Figures (5.6) and (6.4) for comparisons.

The controller's algorithm is written in such a way that the least number of calculations take place inside the dynamic programming iterations to lighten the computational burden of the whole task and those calculations are done outside the iteration loop (Johannesson and Egardt 2008). Matrix power is also employed as much as possible to avoid unnecessary time consuming loops. Static memory allocation is another important fact that is considered in order to use the memory as efficiently as possible. Finally a comparison between this DP approach and an LQ controller is carried out that resulted in having a better performance in applying LQ regulator to this problem. Even manipulating the density of the state values grid or that of the admissible controls cannot result in a better performance than the LQ controller. Making the grid denser will naturally increase the computational burden and does not necessarily give lower costs. See Figures (5.8) through (5.11).

Figure (5.4) shows the simulation results for the asymptotically stable system described in Equation (5.5) with state matrix equal -0.5, a sparse grid of possible state values and a sparse grid of possible controls with an initial state equal to 1.1. As can be seen, the cost to go function has an extremum at state equal to 0. Note that in the down right part of the Figure (5.4), admissible state value 1 represents -1 in the admissible state values $[-1,2]$. Hence, 1 through 5 in this figure represent state values -1.0 0.0 0.5 1.0 2.0 as defined in the algorithm. The total cost is 0.7921.

Figure (5.5) is the result of simulating the same system as the one in the previous example but with a very dense grid of state values. This grid is defined to have 1001 values linearly spaced between -1 and 2 to see how this can affect the total cost to go. As can be seen the total cost to go does not differ much from that of the previous example. In fact a sparse grid of state values can work as well as a dense grid and with much less computation.

In Figure (5.6), the simulation is done on the marginally stable version of the system described in Equation (5.5). A sparse grid of possible state values as well as a sparse grid of possible control values is chosen and the same initial state as all other simulations is applied. As can be seen, the state trajectory falls into oscillations since basically having a marginally stable system with a very limited number of admissible control values, the control signal reaches zero and gets stuck there (to keep the cost low) before the state reaches zero. Now let see how a dense admissible control can improve this.

As can be seen in Figure (5.7), adding more admissible control values solves the problem. This is the simulation result of the same system as the one in the previous example but having a dense admissible control of 1001 values linearly spaced between -1 and 1.

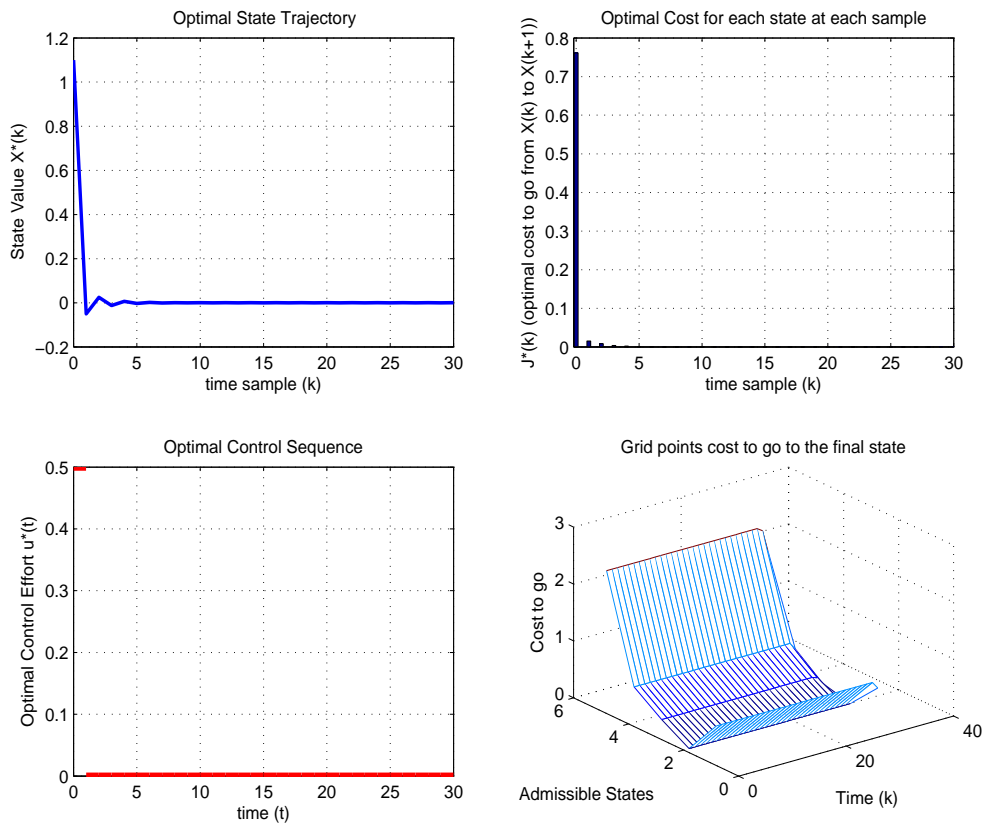


Figure 5.4. Resulting trajectory and cost with a sparse grid of state values. The total cost is 0.7921

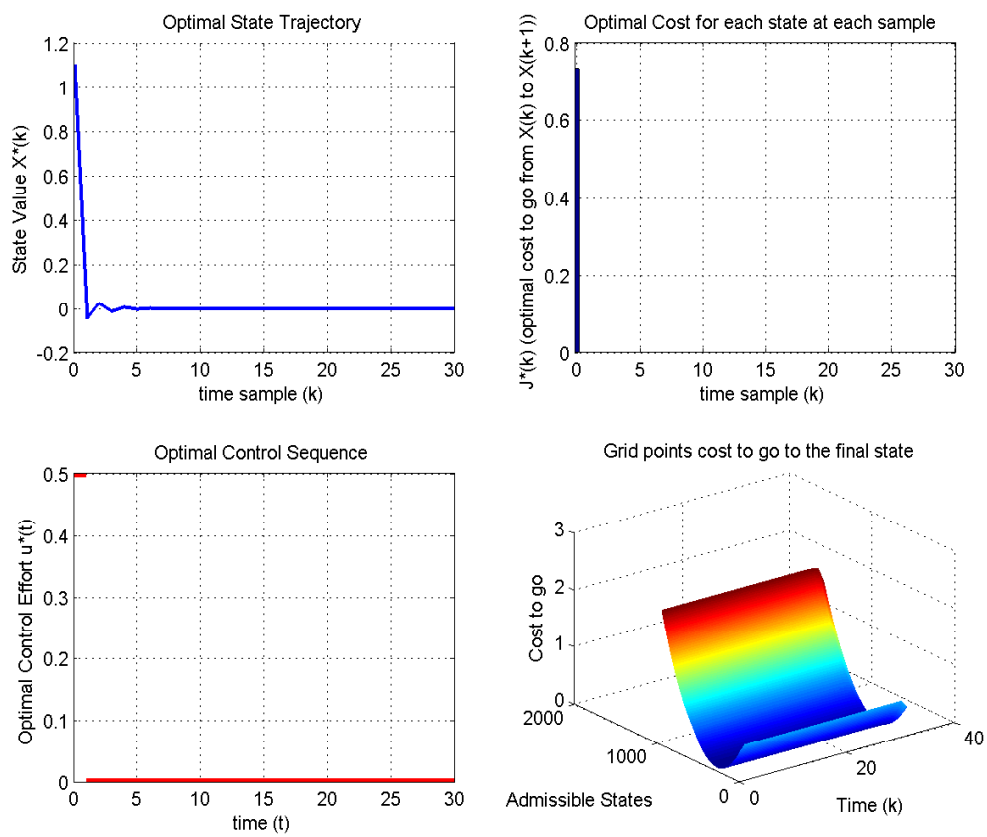


Figure 5.5. Resulting trajectory and cost with a dense grid of state values. The total cost is 0.7343

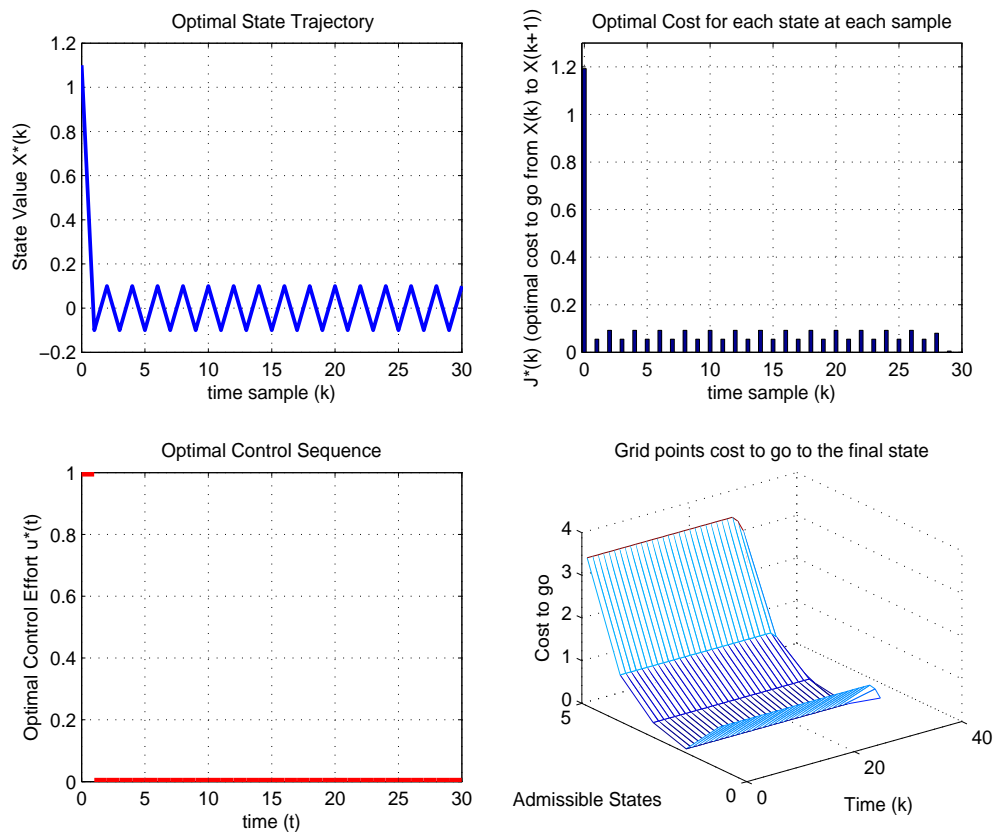


Figure 5.6. Resulting trajectory and cost with a sparse grid of admissible controls. The total cost is 3.25

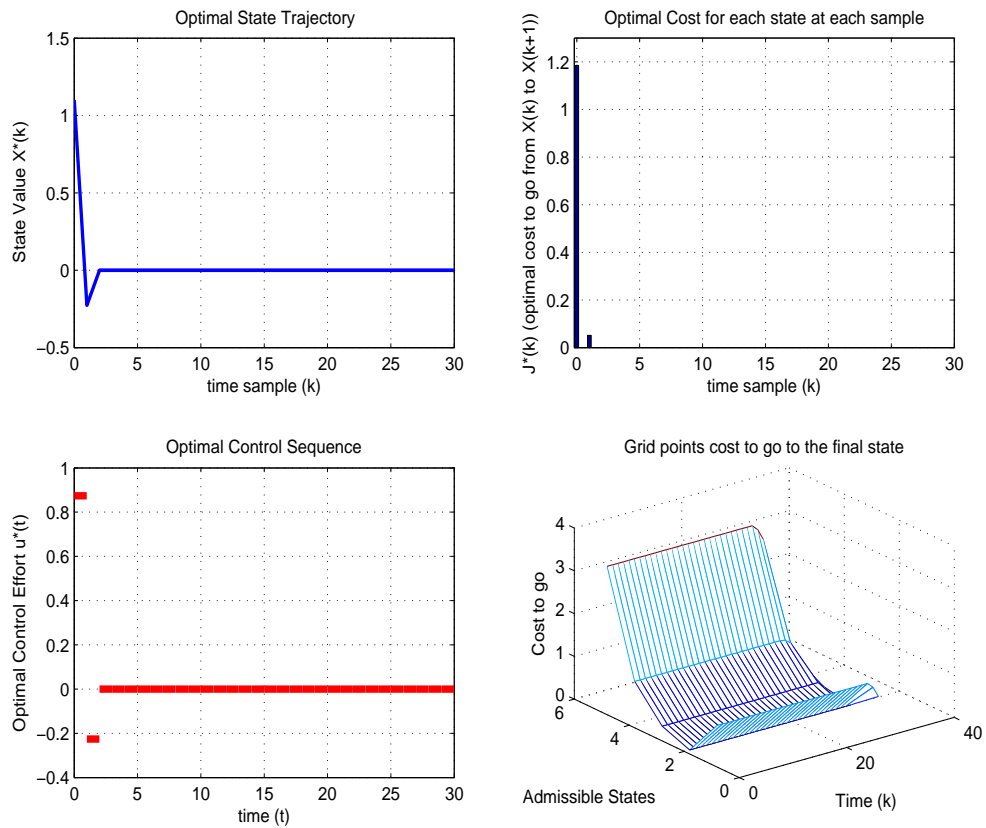


Figure 5.7. Resulting trajectory and cost with a dense grid of admissible controls. The total cost is 1.2358

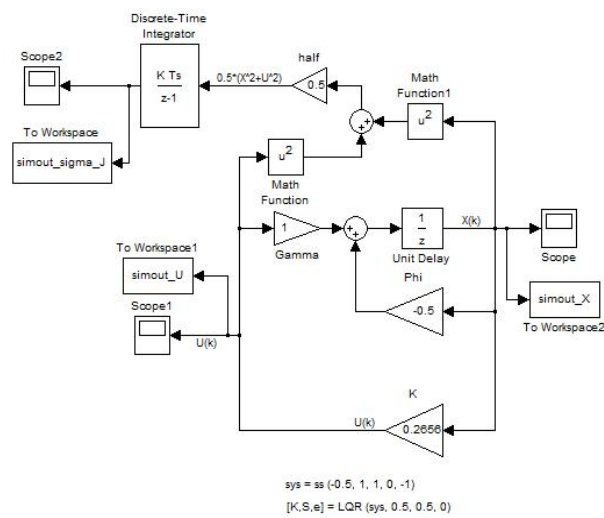


Figure 5.8. LQ approach on the same system as that in Figure (5.5). The total cost is improved to 0.6853 as shown in Figure (5.10).

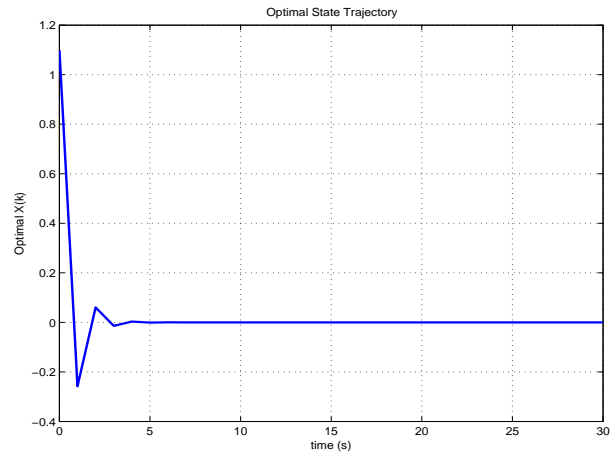


Figure 5.9. Resulting state trajectory for the system in Figure (5.8).

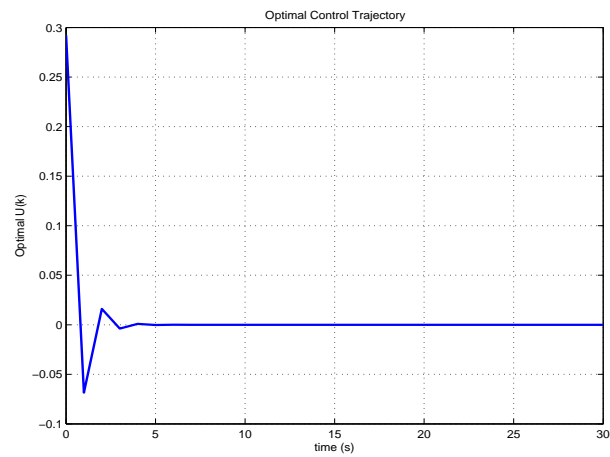


Figure 5.10. Optimal control signal for the system shown in Figure (5.8).

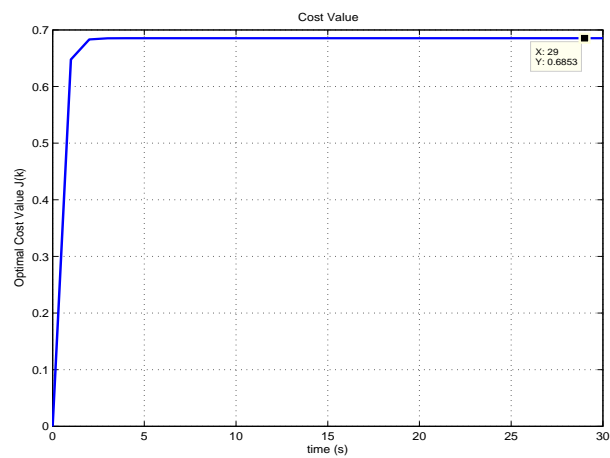


Figure 5.11. Cost value for the system shown in Figure (5.8).

6 DP Approach to HEV Fuel Consumption Optimization

In this chapter, the reader will get acquainted with the method used in the presented thesis to optimize the fuel consumption in a parallel hybrid electric vehicle based on Dynamic Programming.

Inspired by the very simple model from Section 5.3 and having the real data from Toyota Prius I, the deterministic case is considered where the whole drive cycle is known to the controller beforehand. Such cases as mentioned before are applicable to vehicles that travel in predetermined routes many times and have stop-and-go characteristics. In such vehicles, there are not drastic differences among different travels and hence a predefined drive cycle can work acceptably.

6.1 Real HEV Data

Having studied all these chapters, it is time to apply the developed Dynamic Programming algorithm to a much more real case to see the effect of different limitations and constraints in a real hybrid electric vehicle, of course limited in such a way that it fits the scope of the presented work. To do so, engine, electric machine, battery and vehicle data are taken from real machines.

As our experiment, let's consider the following information as the given data for these machines to see how the algorithm performs. The data are given in the following tables.

Vehicle Data	
Mass (kg)	1500
Wheel Inertia	0
Wheel Base (m)	2.55
Vehicle Front Area (m ²)	1.746
Wheel Radius (m)	0.2870
Air Density (kg/m ³)	1.2
Rolling Resistance (-)	0.009
Drive (-)	4WD

EM Data	
Nominal Power (kW)	20
Max. Torque (N.m)	55
Max. Speed (rpm)	5500

ICE Data	
Nominal Power (kW)	43
Max. Torque (N.m)	101.97
Max. Speed (rpm)	4000

Battery Data	
Max. Continuous Current (A)	125
Max. Transient Current (A)	250
Capacity (Ah)	7.5
Number of cells in series	125
Number of cells in parallel	1

As can be seen, this is a fairly light vehicle that can be categorized under the passenger vehicles family.

6.2 Control Algorithm

In order to treat the problem of optimal DP-based controller design and applying it to the system model, the following considerations are taken into account. It is considered at the following simulations that the engine works with gasoline and the whole pre-assumptions about a hybrid electric vehicle, such as regenerative braking option, power assist mode, zero emission mode, etc., as explained in Chapter 3 are taken into considerations. The driving cycle is taken from standard driving cycles and the initial charge of the battery is considered to be 50% of the total charge. Note that the time vector in the following figures is given as the sample instead of time, though since the sampling time is 1 second in all the simulations, the time axis in samples corresponds exactly to the time axis in seconds. The battery is allowed to work only in a region of 15% of the whole battery capacity placed symmetrically around the 50% of the battery capacity for battery wear considerations (Johannesson 2006). Another issue to be pointed out is that the 15% stripe in the middle of the battery capacity is quantized into 16 different linearly spaced values between the SoC boundaries.

Since the route and the speed profile are known a priori, the controller is designed in such a way that the algorithm starts from the end of the driving cycle and having torque demand, speed demand, engine and the electric machine torque and speed limitations it is pretty straightforward to find a set of different possible gears that can handle the demands. In this phase, based on the engine fuel consumption map the optimal gear number for every sample can be found. Now having all these data at hand together with the constraints over the battery current and state of charge, it will be possible to look for the best mix of EM and ICE torque that can deliver the driver demands to the final drive. Finally, for the selected torque and speed, some cost will be calculated and assigned to each specific sample based on the fuel consumption at that sample. In such a way, going once from the end of the driving cycle to the starting point in an off-line fashion gives a map as the designed controller to the computer to be used as a reference (look-up table), in the online application. Hence, applying this controller to the simulated system will be nothing but a look-up table to check the best solution for this optimal control problem.

6.3 Simulation Results

In this section the results of applying the developed control algorithm on the real vehicle data with two different standard driving cycles are presented. The first driving cycle is

picked from Japanese standard cycles and the second one is one from American standard cycles.

6.3.1 A Japanese Standard Driving Cycle, 10 - Mode

Given the data and algorithm presented above, it is desirable to test the designed controller with a standard driving cycle. Here as the first example, a Japanese cycle is chosen from QSS Toolbox.

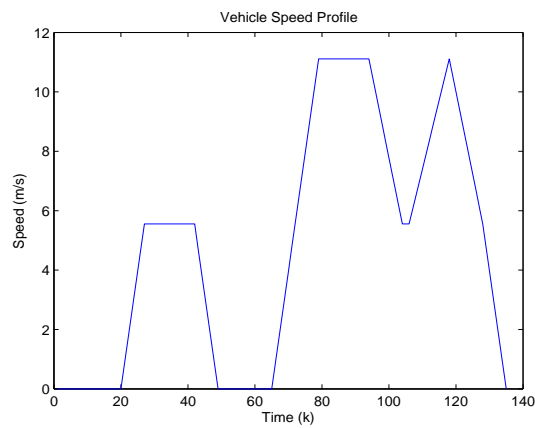


Figure 6.1. Japanese Standard Driving Cycle, 10 - Mode

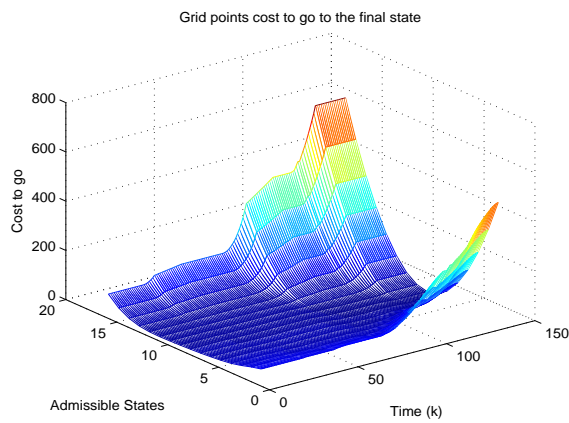


Figure 6.2. DP - Based Controller

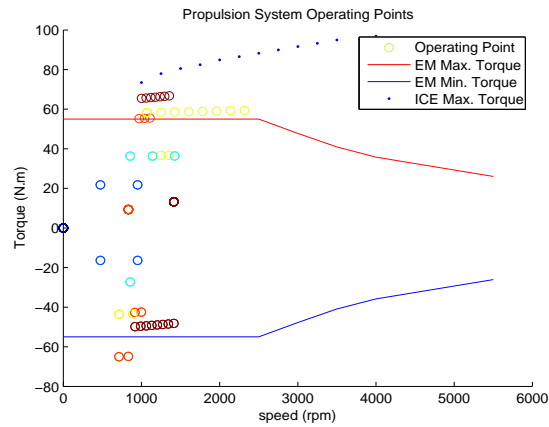


Figure 6.3. Propulsion System Operating points with constraints

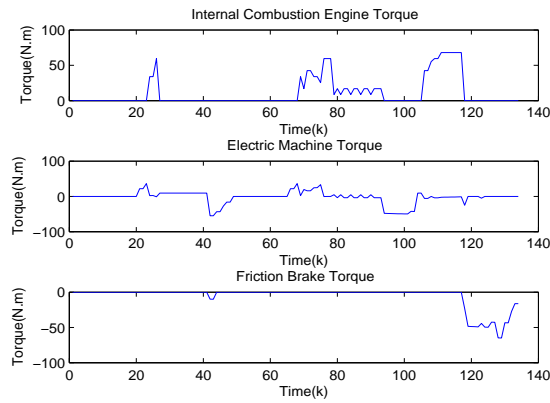


Figure 6.4. Torque Split among ICE, EM and Friction Brake

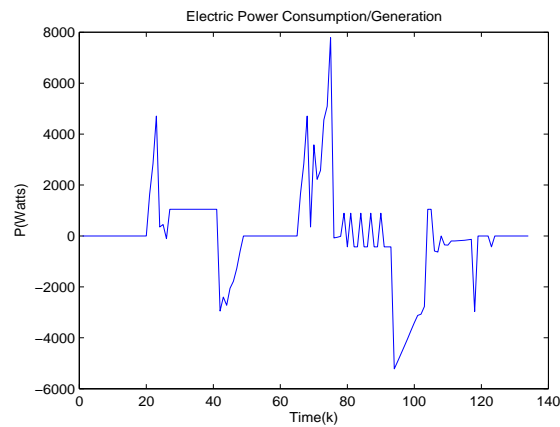


Figure 6.5. Electric Power flowing through the electric path

As can be seen in Figure (6.1), the driving cycle to be driven is a fairly smooth one

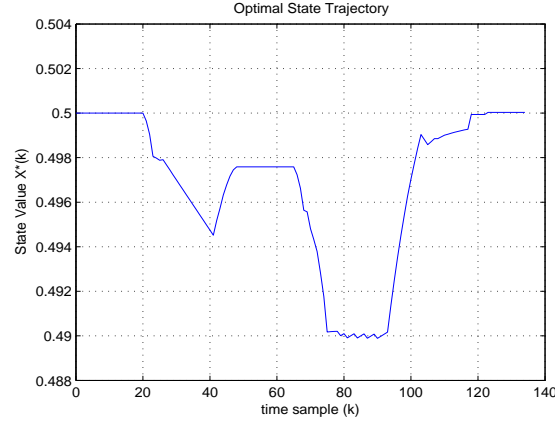


Figure 6.6. Battery State of Charge during the driving cycle

containing acceleration, deceleration, constant speed and standstill phases. This is chosen for the sake of readers to be able to follow the cycle easily. Since it is desirable to have the same SoC value at the end of a driving cycle as that of the starting point (charge sustainability), i.e., 50% at the simulations presented here, the controller is designed with a high penalty over the deviations of the SoC value from the desired final value. This can be seen in Figure (6.2) where the controller map has a very deep parabolic shape. The rest of the controller map also has the same characteristics but with a much lower penalty over the deviations from the SoC middle value. However, the state of charge is never allowed to cross the hard constraints and go beyond them and will be kept within the boundaries. Please note that the controller algorithm is written in such a way that the state equal to 1 according to Figure (6.2) corresponds to the lower bound of the SoC and the state equal to 16 in the same figure corresponds to the upper bound of the SoC, i.e., $SoC = 43\%$ and $SoC = 58\%$, respectively.

Figure (6.3) shows the power distribution in the propulsion system together with the torque limitations in the ICE and the EM. The interpretation from this figure is that the selected torque has never hit the constraints. Note that there are two points in the figure that are located under the EM minimum torque. These points correspond to the FB torque that are not limited to these constraints. The total power has also never hit any constraint (the maximum power delivery according to the Figure (6.3) is 14.425 kW).

Figure (6.4) shows the torque split among the two prime movers as well as the friction brake, based on the designed DP-based controller. The considerable percentage of Regenerative Braking (RB) over Friction Braking (FB) can be seen easily. There is only some short time at the end of the driving cycle that the FB has been seriously used and that is because of both the hard penalty over the battery final state which must be kept at 50 percent and the steep deceleration at the end of the driving cycle. The electric power flow is also shown in Figure (6.5) especially to show the peak values of the electric power during the cycle.

Finally, Figure (6.6) shows the most important value in the whole system, that is the SoC of the battery. As can be seen, it is well located in the allowed boundaries; it starts from 0.5 and ends in 0.5 (remember the SoC quantization between $[0,1]$). The charge/discharge

phases are shown clearly and can be compared with the electric power flow, torque split and speed profile plots.

6.3.2 An American Standard Driving Cycle, FTP - HIGHWAY

Let's take a look at another standard driving cycle. This time an American cycle (FTP-HIGHWAY) is picked from the QSS toolbox cycles. As can be seen, all the above general conclusions can be made based on these simulation results as well. One major difference is for example the charge/discharge behavior of the battery, which has a major charging phase at the beginning of the driving as can be seen in Figure (6.12), that totally depends on the speed profile characteristics, or better say, on the driving demands. The rest of the conclusions hold in these simulations as well.

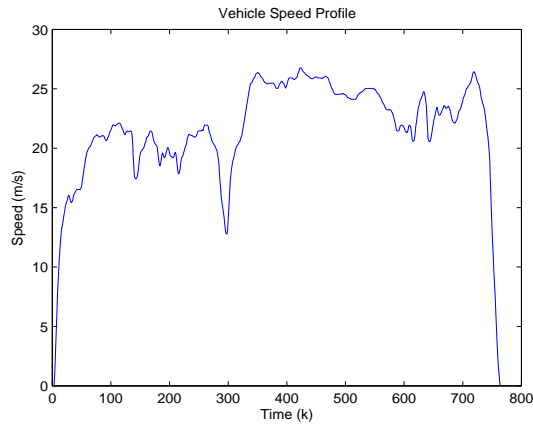


Figure 6.7. American Standard Driving Cycle, FTP - HIGHWAY

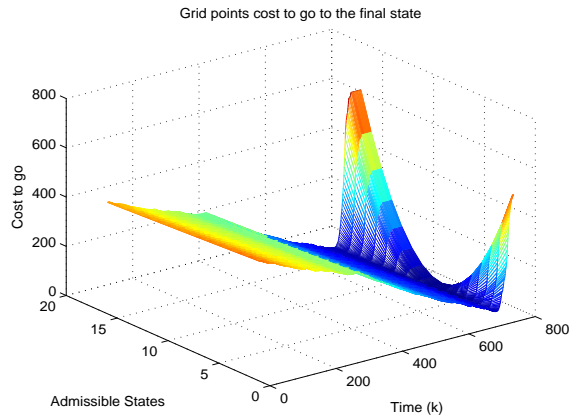


Figure 6.8. DP - Based Controller

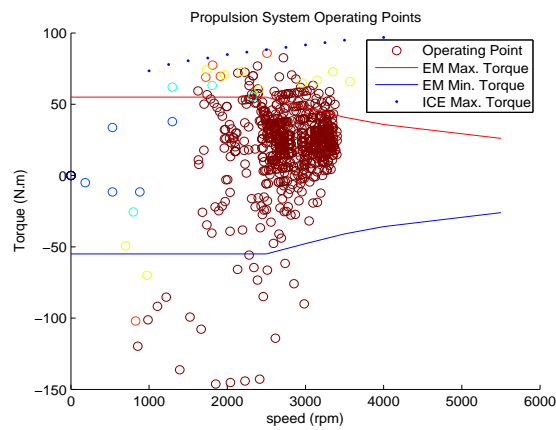


Figure 6.9. Propulsion System Operating points with constraints

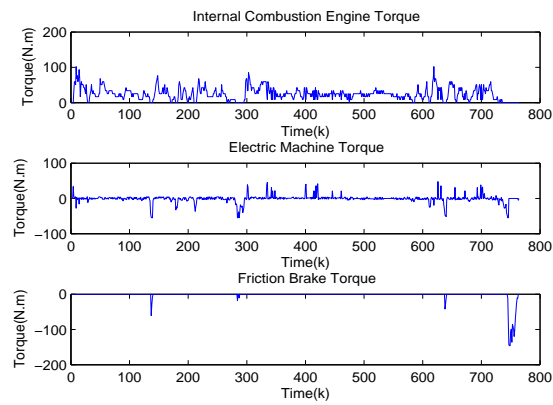


Figure 6.10. Torque Split among ICE, EM and Friction Brake

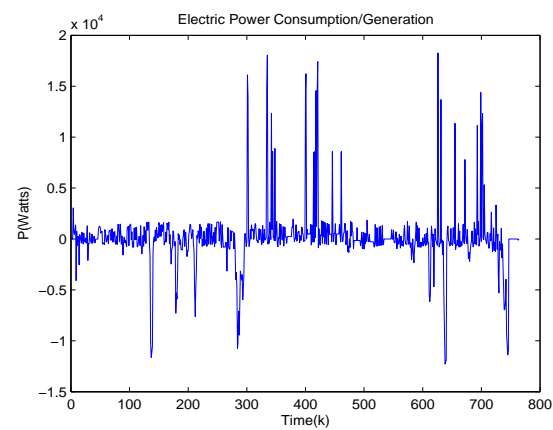


Figure 6.11. Electric Power flowing through the electric path

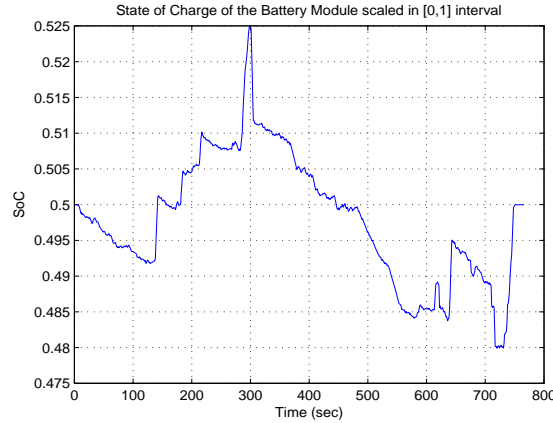


Figure 6.12. Battery State of Charge during the driving cycle

6.3.3 Electric Machinery Sizing

As an example to see how the developed algorithm together with the toolbox power can help a researcher investigate the effect of different parameters and components, let's investigate the effect of electric machine downsizing. Basically, the electric machine weight (and cost as a direct result) are of quite importance in any hybrid electric vehicle design. Hence, having a tool to ease these investigations is crucial. Here, utilizing the toolbox power, the effect of using a considerably smaller electric machine in the same vehicle as we had in the previous sections is illustrated. Since a smaller electric machine is used, the first thing that comes into mind is to downsize the battery pack as well because battery is another expensive part of any hybrid vehicle. Therefore, the battery total capacity is decreased as summarized in the following table. Note that "EXP1" corresponds to the experiment in section 6.3.2 in which the American standard driving cycle FTP-HIGHWAY was applied to the vehicle with the previously explained parameters. "EXP2" on the other hand, is the new experiment on the same vehicle and with the same driving cycle but with a smaller EM and BT pack as pointed out in the following table.

Parameter	EXP1	EXP2
Driving cycle	FTP - HIGHWAY	FTP - HIGHWAY
EM scaling factor (-)	1	0.5
Battery capacity (Ah)	7.5	3.75
# of battery cells in series (-)	125	125
# of battery cells in parallel (-)	1	1

As can be seen, the new electric machine in EXP2 is half the one in the previous experiment and so the battery pack is. Now let see how these two together with the same ICE perform.

The new DP-based controller (cost matrix) in Figure (6.13) seems almost the same as that of the previous experiment shown in Figure (6.8) with slightly different parts especially at the final 200 samples. Based on the new controller, the total torque demand is requested from ICE, EM and FB as shown in Figure (6.14). As can be seen, the torque distribution

is different from that of EXP1 shown in Figure (6.10). The reason is first having a smaller EM that obviously can provide lower torque and power and also having lower battery energy.

Looking at the state of charge trajectory Figure (6.15), one can easily see that in EXP2 having lower battery capacity is resulted in having the battery work mostly below the 50% while in EXP1 the battery mostly works above the 50%. It is also important to mention that this issue affects the fuel consumption as well, since it affects the ICE torque directly. In fact the experiment shows 0.53% increase in fuel consumption that is the cost of having smaller electric parts.

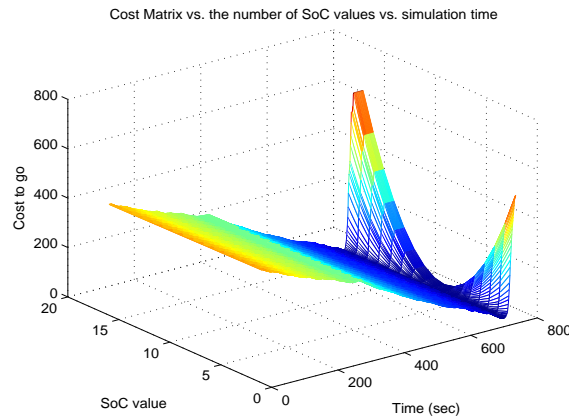


Figure 6.13. DP - Based Controller

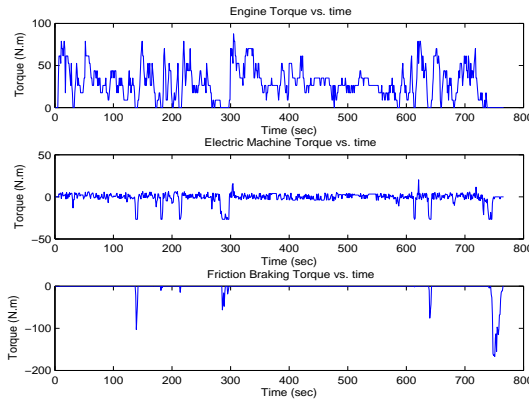


Figure 6.14. Torque Split among ICE, EM and Friction Brake

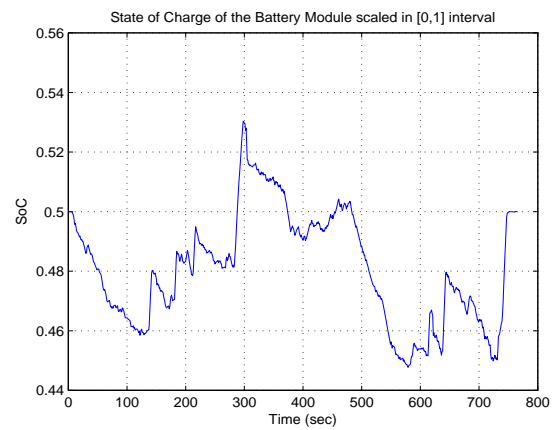


Figure 6.15. Battery State of Charge during the driving cycle

7 The Designed Toolbox

Having a fully functional algorithm as a piece of code, it feels more natural to develop it in such a way that it can be used by future researchers in their investigations on hybrid electric vehicle fuel consumption optimization. This fact inspired the idea to design and develop a toolbox that uses the DP algorithm as its beating heart in order to both apply the DP algorithm and let the users work with it in a much easier way. This is done by utilizing the GUI design power of Matlab together with handy tools of Simulink and QSS Toolbox. Although this toolbox is designed in such a way that it works with this specific architecture that is used throughout this thesis work, i.e., this parallel HEV, it can be modified to be used with any other architecture or even with a library of different existing architectures provided that the programmer has enough insight of how this toolbox is written, organized and working. This chapter goes through the developed toolbox and its abilities in hybrid electric vehicle studies.

7.1 Toolbox Hierarchy

The toolbox appearance contains a main GIU and this main graphical user interface consists of some sub-GUIs, each one responsible for some pre-specified task(s). The toolbox alone contains 68 different files in 5 folders and together with the QSS toolbox with 82 files will deal with around 150 different files. The total number of Matlab code lines for the toolbox alone is something around 3400 lines. Having so much information and the fact that the toolbox has to deal with Matlab, Simulink and QSS almost at the same time makes having an accurate communication among different workspaces and data types vital. This will be explained further.

In order to be able to give the reader a clear understanding of how the toolbox works, let's have a closer look at each part of the toolbox separately.

7.1.1 Main GUI

The first and somehow the most important part of the toolbox is its main GUI. This major user interface pops up upon user request via Matlab command line as:

```
>> MainGUI
```

or by simply double-clicking on its corresponding icon in the toolbox folder. As any other command in Matlab, this one is also case sensitive and in case of misspelling the user may get Matlab warnings or even errors in different cases. The Main GUI is actually a palette containing four separate sub-palettes as can be seen in Figure (7.1).

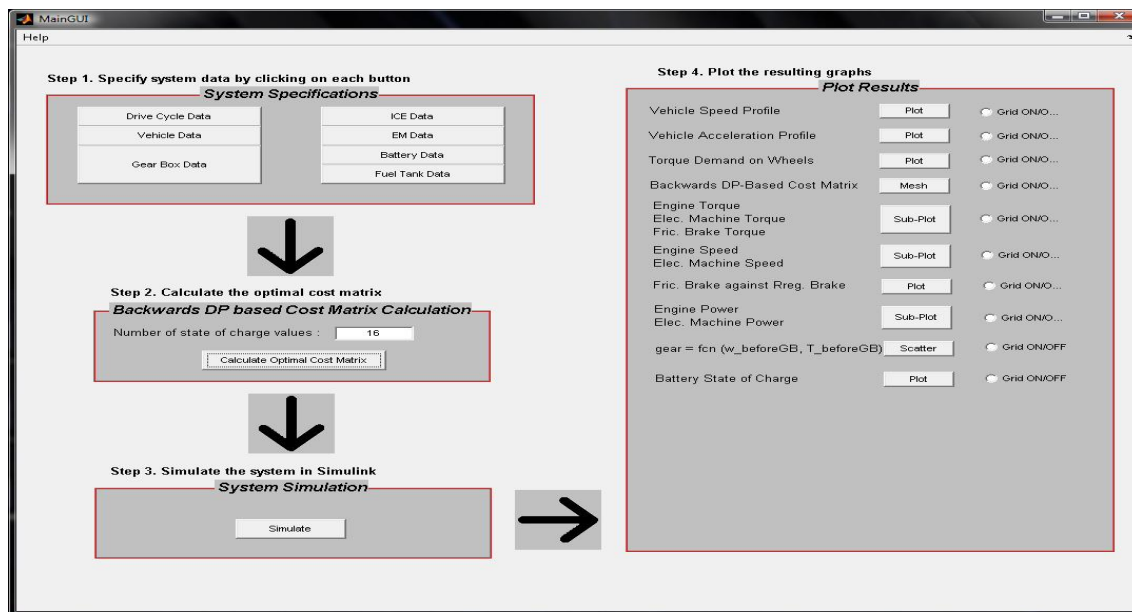


Figure 7.1. Main GUI appearance

These four separate sub-palettes are as follows:

1. System Specification
2. Backwards DP-based Cost Matrix Calculation
3. System Simulation
4. Plot Results

Each one of these palettes will be described in the following sections. The main role of the Main GUI is to gather all the so called sub-GUIs as well as their push buttons, edit boxes and so on together in a single unit and this makes accessing all features of the toolbox much easier for the user.

In the Main GUI, four separate steps are illustrated to be done one after the other as shown in Figure (7.1). The order should be followed as described in order to make a complete simulation from system specification to results plots. However, starting from a step other than step 1 right after running the toolbox is also possible in cases that the previous steps are done beforehand in previous runs, since basically the structure of the toolbox allows data saving and reload from previous simulation runs. This mechanism will be described later.

7.1.2 Simulink Model

Together with the Main GUI, the system model as a Simulink .mdl file is also loaded automatically. This is the model of the previously described parallel HEV together with the DP-based controller as a Matlab function added to the system as can be seen in Figure (7.2).

- ICE Data
- EM Data
- Battery Data
- Fuel Tank Data

Drive Cycle Data sub-GUI

This sub-GUI allows the user to choose a standard drive cycle among 17 available cycles in the toolbox, as well as to define the step size for the whole toolbox and simulation model and also has the option to stop or not to stop the simulation after the end of the driving cycle. These options are provided using a popup menu, an edit box and a check box as is shown in Figure (7.3).

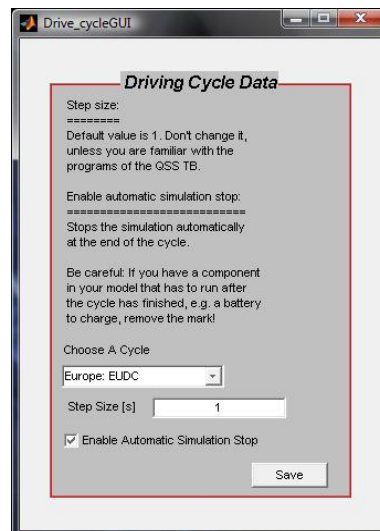


Figure 7.3. Drive Cycle GUI

Note that the default value for the step size is 1 sec and must not be changed unless the user is fully familiar with the QSS TB and the toolbox program.

After entering all the data, pushing "Save" button saves the selected data as structure in a .mat file and also assigns those parameters in the corresponding Simulink block (Drive Cycle block in this case). This strategy goes for all other six push buttons/sub-GUIs.

Vehicle Data sub-GUI

This sub-GUI is responsible for gathering the vehicle data via its six edit boxes as shown in Figure (7.4). These data are vehicle mass, rotating mass, vehicle cross section area, wheel diameter, drag coefficient and rolling friction coefficient. Again, having entered all these data, pushing "Save" button saves the data as structure in their corresponding .mat file to be accessible to other parts of the toolbox as well as the Simulink model.

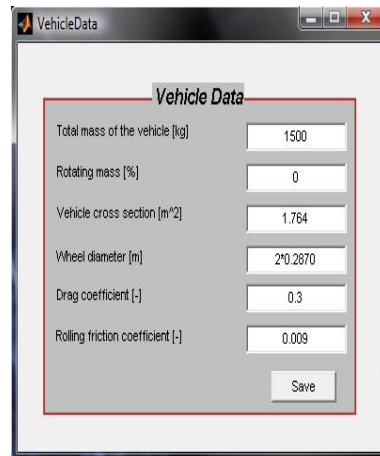


Figure 7.4. Vehicle GUI

Gearbox sub-GUI

The Gearbox sub-GUI gathers the gearbox data through its nine edit boxes. These data are gear ratios, gearbox efficiency (equal for all gears), idling losses and minimum wheel speed beyond which losses are generated as shown in Figure (7.5). It goes without saying that pushing "Save" button works as in other sub-GUIs.

One big difference here is that in the Simulink model two separate gearbox models are used because of QSS restrictions. The data given in the Gearbox sub-GUI is hence applied to both gearbox models in the same way (as they are practically one single gearbox).

It must also be mentioned that the gear ratios are calculated based on the driving cycles demand and standards. Hence, manipulating the gear ratios especially for the highest and the lowest gears must be done carefully, otherwise this may even stop the whole toolbox from working.

ICE sub-GUI

This graphical user interface lets the user choose the engine type if it is Otto or Diesel as well as set the engine parameters such as engine displacement, engine scaling factor, engine inertia, engine idling speed, engine idling power, power required by the auxiliary equipments and fuel cutoff features via its edit boxes, popup menu and check box as is shown in Figure (7.6). These data are partly used in the DP algorithm and fully in the Simulink model. The fuel cutoff option makes it possible to have a more realistic model of the system by cutting off the fuel injection to the engine in low torque demands. For further information refer to (Guzzella and Sciarretta 2007)

EM sub-GUI

The electric machine data user interface is simply gathering the data for the electric machine model as scaling factor, motor inertia and power required from the auxiliaries.

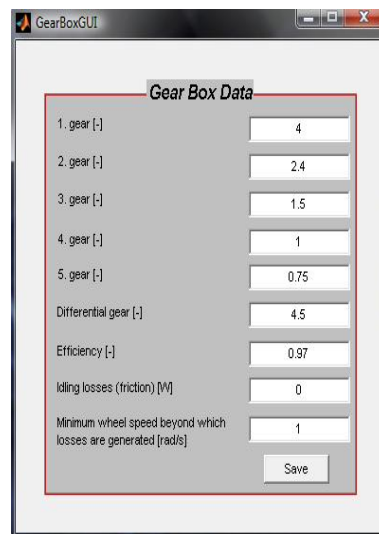


Figure 7.5. Gearbox GUI

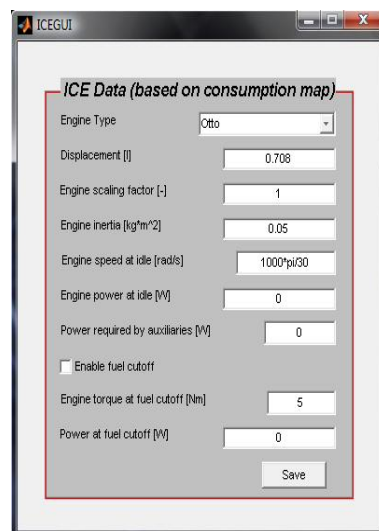


Figure 7.6. ICE GUI

EM GUI is shown in Figure (7.7). The scaling factor allows using smaller or larger electric machines in sense of power consumption/generation.

Battery sub-GUI

Battery GUI is a simple interface for gathering the battery data from the user. These data are the battery capacity, battery initial charge, number of battery cells in series, number of battery cells in parallel and battery current limit. The battery initial state of charge is chosen to be 50 % due to the application in this thesis work though it can be changed easily via its user interface. You can see the interface in Figure (7.8). The battery model

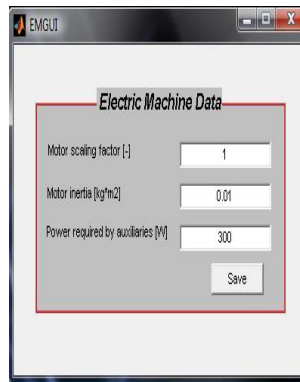


Figure 7.7. EM GUI

is basically inspired by that of the QSS toolbox but modified to a large extent to match the control algorithm used in this work. In fact, the battery model is a much simpler one compared to that of the QSS toolbox.

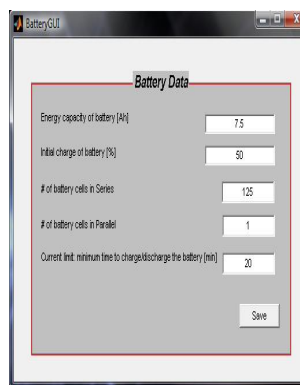


Figure 7.8. Battery GUI

Fuel Tank sub-GUI

Last but not least is the graphical user interface for the fuel tank. This GUI allows the user to pick the fuel type from a list of three different fuels, i.e., gasoline, diesel or hydrogen from a popup menu. It is also possible to choose if cold start losses are taken into account or not via a check box. Obviously, considering cold start losses will increase the total average fuel consumption in long run. The fuel tank GUI is shown in Figure (7.9).

7.1.4 DP-based Cost Matrix Calculator

The second palette in the main GUI belongs to the very heart of the toolbox, the dynamic programming algorithm. As can be seen in Figure (7.1), the number of SoC quantized values is available to the user to choose from the main GUI. This parameter is described in Chapter 6. Pushing the "Calculate Optimal Cost Matrix" loads the whole data required

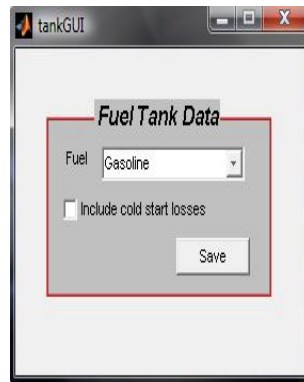


Figure 7.9. Fuel Tank GUI

by the DP algorithm to this push button's workspace, initializes all Simulink blocks by running their initialization files from QSS TB, models the chassis and powertrain and loads their data from their structures into its local workspace, finds the optimal gear shift map taking minimum fuel consumption into account and finally calculates the optimal cost matrix using Backwards Dynamic Programming (BDP) algorithm and saves the resulting data into a separate .mat file as well as sharing them with Matlab base workspace for further use. More details on communication and data sharing will be given later.

7.1.5 System Simulation

The third step in using the toolbox is simulating the system using the DP-based optimal cost matrix calculated in step 2. This is done using the push button in the third palette coined "Simulate" as can be seen in Figure (7.1). Pushing this button simply runs the Simulink model to apply the designed controller to the system model and to run it through the driving cycle. A parameter 'time' is also defined for some calculation purposes that must be set to zero before and after each run for synchronization reasons between the toolbox and the Simulink model. This is also done automatically upon "Simulate" button press.

7.1.6 Plot Results

The last step in every run is to look at the simulation results for analysis reasons. This has been realized in the fourth palette of the main GUI. This palette as shown in Figure (7.1) contains a series of push buttons together with their corresponding radio buttons. Each push button is responsible for loading their own data from .mat files into their own local work spaces and making them available to the user as output figures. These figures can be plots, meshes, sub-plots, etc., as shown in Figure (7.1). The radio buttons are just responsible for turning the grid option on/off for every figure. These push buttons in the "Plot Results" palette function respectively as follows:

- Vehicle speed profile plot

- Vehicle acceleration profile plot
- Torque demand on wheels plot
- Backwards DB-based optimal cost matrix mesh
- ICE/EM/FB Torque plots
- ICE/EM rotational speed
- FB vs RB torque plots
- ICE/EM power plots
- Selected gear scattered as a function of speed and torque blend of the ICE and EM just before entering Gearbox
- Battery SoC plot

7.2 Communication and Data Sharing

Working with a rather large program and having many different variables, functions, sub-functions, work spaces, etc., especially when it comes to matching two or more different programs together, like the problem we have at hand, make it vital to decide on choosing a reliable strategy to share all these data among different parts of those programs and to communicate among them. Here in our case, we have as mentioned before, a toolbox working in Matlab environment together with Simulink and QSS toolbox. The toolbox alone has around 180 functions and subfunctions that many of them have their own work spaces, i.e., their local variables will not be accessible to the other work spaces and functions. QSS TB has also its own functions, variables and work spaces.

One solution that may come to mind is to declare local and global variables in such a way that shared variables can be accessible among all corresponding work spaces. Although this is a feasible solution, it takes a lot of time separating different variables and thinking of which variable is going to be used by which workspace. The risk of altering the data due to wrong declaration and overwriting on existing data is also very high in this way. This is specially dangerous since the toolbox works with the QSS library and if this library gets corrupted then everything loaded from that will not be correct and standard data and hence the whole simulation will be jeopardized, if it works at all!

The remedy that is chosen in this work is to use files to save some specific data and then to load them wherever they are needed. This solution together with local/global declarations, when necessary but as few as possible, are applied to the toolbox to let the different work spaces access the shared data. These data are saved in .mat files in either arrays or structures depending on the data type. Whenever there are some data related to a single machine, like electric machine data e.g., those data are saved as structures. Hence, accessing any data at any time will be plausible for any function by using Matlab load/save commands. This is also done inside the toolbox code and therefore the user will not need to deal with or even think of data sharing issues and can have a seat and think of the main problem that he/she is dealing with.

Basically, the data in this toolbox are saved in three different ways:

- Right upon push button press
- During the program run
- Simulink sinks "To File"

The first and the second ways are basically the same, but in the first case, the objective of pressing a push button is specifically to save the data and this will be run right away after releasing the push button, like what is done in "Save" buttons. In the second case, instead, the objective is to carry out some other task but in the meanwhile, there are some data that need to be saved in .mat files. This can be found in "Calculate Optimal Cost Matrix" button for example. In either cases, Matlab "save" is used to save the data in the appropriate location. Note that all these files will be saved to/loaded from the current Matlab directory. So, setting the current directory to "...My Toolbox\System Settings" is a necessity. The third case is using Simulink sinks called "To File". This option allows the data to be saved in a .mat file as an array of two rows. The first row will be time vector and the second row will contain the actual data.

Those data that are needed to be loaded as default values into the toolbox upon first time runs of the toolbox are loaded from the previously saved data. This is done automatically using Matlab "load" command whenever applicable.

There is also an m-file called "startup.m" that declares the rest of the global variables for the whole complex of programs. This file must be either run before running the toolbox or be added to Matlab "autorun" part to be run by default whenever Matlab is launched.

8 CONCLUSION AND DISCUSSION

The problem to optimize the fuel consumption in a hybrid electric vehicle subject to engine constraints, electric machine constraints and battery constraints and with pre-defined driving profiles is treated formally. This is based on developing a dynamic programming based algorithm to be applied to the vehicle and powertrain model.

To do so, some studies have been done on hybrid electric vehicles in general and more specifically on parallel HEVs. Based on these studies, mathematical models for chassis and powertrain have been derived and used for simulation and control design purposes. Another subject to be studied was Dynamic Programming method which was the underlying methodology to approach this problem. This has been done as well as some more sophisticated methods known as Approximate Dynamic Programming techniques that can be utilized in similar but much more complex problems in order to reduce the so called curse of dimensionality, though these methods have never been used in the developed algorithm.

Finally, the control algorithm has been applied to the hybrid electric vehicle model and many different simulations have been carried out on the model. The simulations show satisfactory results in fuel consumption of the vehicle model. Then, a toolbox working under Matlab and Simulink environments together with the QSS Toolbox has been developed to employ the algorithm in a more flexible environment with a graphical user interface for the sake of the user to be able to manipulate the parameters and data more easily and to carry out further investigations on the subject.

All the investigations ended in the conclusion that the developed algorithm can satisfactorily be applied to deterministic optimization cases such as that of the commuter buses or refuse trucks that have almost the same stop-and-go driving profile all the time.

As future work, one natural step to be taken is to extend the problem to a more complicated stochastic one or one with higher number of state variables and to employ appropriate approximation techniques to overcome the hardships of the curse of dimensionality. Further, some investigations on sensitivity and robustness of the controller can also be carried out.

Bibliography

- Bertsekas, Dimitri P. (2000). *Dynamic Programming and Optimal Control*. Athena Scientific. 2nd edn. Athena Scientific. Belmont, MA., USA.
- Chan, C. C. (2007). The state of the art of electric, hybrid, and fuel cell vehicles. *Proceedings of the IEEE*, **95**(4), 704–718.
- Guzzella, L. and Amstutz, A. (2005). *The QSS Toolbox Manual*. IMRT - ETH. ML K 32.3, Sonneggstrasse 3, 8092 Zürich, Switzerland.
- Guzzella, L. and Sciarretta, A. (2007). *Vehicle Propulsion Systems, Introduction to Modeling and Optimization*. Springer. 2nd edn. ETH. Zürich.
- Johannesson, L. (2006). On Energy Management Strategies for Hybrid Electric Vehicles. PhD thesis. Chalmers University of Technology. SE-412 96 Göteborg, Sweden.
- Johannesson, L. and Egardt, B. (2008). Approximate dynamic programming applied to parallel hybrid powertrains. *Proceedings of the 17th IFAC World Congress*.
- Krause, Paul C., Wasynczuk, Oleg and Sudhoff, Scott D. (2002). *Analysis of Electric Machinery and Drive Systems*. IEEE Press. 2nd edn. Wiley. USA.
- Luus, R. (2000). *Iterative dynamic programming, electronic resource*. 1st edn. Chapman and Hall/CRC. USA.
- Miller, John M. (2006). *Propulsion Systems for Hybrid Vehicles*. 1st edn. IET. UK.
- Montazeri, M., Ahmadi, A. and Asadi, M. (2008). Driving condition recognition for genetic-fuzzy hev control. *3rd international workshop on genetic and evolving systems*.
- Naidu, Subbaram and Naidu, D. S. (2002). *Optimal Control Systems*. 1st edn. CRC Press. USA.
- Schouten, N. J., Salman, M. A. and Kheir, N. A. (2002). Fuzzy logic control for parallel hybrid vehicle. *IEEE transactions on control systems technology*, **10**(3), 460–468.
- Sciarretta, A. and Guzzella, L. (2007). Control of hybrid electric vehicles. *IEEE Control Systems Magazine*, **27**(2), 60–70.
- Won, J. S. and Langari, R. (2002). Fuzzy torque distribution control for a parallel hybrid vehicle. *The international of knowledge engineering and neural networks*, **19**(1), 4–10.

Appendix A

QSS Toolbox Review

app:QSSTB

In this appended chapter, a short introduction to QSS Toolbox, developed and written in the Measurement and Control Laboratory of Swiss Federal Institute of Technology, Zürich (ETH) is presented. Although the aim is not to make a QSS expert out of the reader, it gives the readers a glimpse of what is going on inside the QSS Toolbox and gives them the clue for further use of the toolbox. It mostly covers the general scheme of the QSS toolbox and the components that are used in the presented work. It is based on the second version of the Quasi-Static Simulation Toolbox which provides the user with a fast and simple estimation of the fuel consumption for various powertrain and architectures. Since this toolbox is working together with the toolbox which is designed in this thesis work, this approach is followed in that toolbox as well.

A.1 The quasi-static approach

As mentioned before, the method that the QSS Toolbox is based on is the quasi-static approach. The idea behind the quasi-static approach is to reverse the cause and effect direction in a physical model. In a vehicle moving procedure for example, traditionally the cause is the force applied from the prime movers and the effect is the vehicle speed, but in the QSS Toolbox, this is completely reversed; cause is the speed given at any time and the effects are mean speed, acceleration and driving force (Guzzella and Amstutz 2005).

A.2 The elements of the QSS Toolbox

As shown in Figure (A.1), different elements available in the QSS Toolbox are grouped together according to their functions.

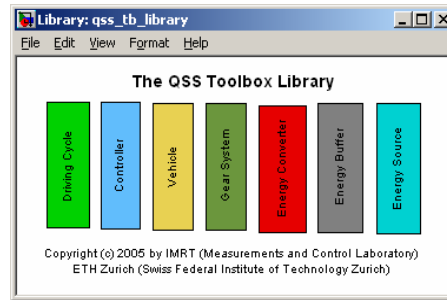


Figure A.1. The top level of the QSS Library

Under each block there are different elements belonging to that specific group of elements that can be reached by double clicking on each main block. Among all these elements located under these blocks, those are used in this thesis work will be explained here. These elements are as the following:

- Driving Cycle
- Vehicle
- Manual Gear Box
- Combustion Engine (based on consumption map)
- Electric Machine
- Battery
- Tank
- Controller (empty template)

These elements are described in more details in the following subsections.

A.2.1 Driving Cycle

In the driving cycle tool of the QSS toolbox, there are a bunch of different standard test cycles are available to the user. These driving cycles are picked from famous European, American and Japanese test cycles.

Each cycle is defined by at least two vectors (Guzzella and Amstutz 2005):

1. A time vector
2. A vehicle speed vector $v_f(k.h)$

Based on the given speed profile as mentioned before, the acceleration is calculated with the following equation:

$$a_f(k.h) = \frac{v_f(k.h + h) - v_f(k.h)}{h}, k = 1, \dots, k_{max} - 1, a_f(k_{max}) = 0, \quad (A.1)$$

The gear shift points are also defined in EU cycles in a separate vector but this does not go for FTP cycles, though these predefined gear shift points are never used in this thesis work and as mentioned before, the gear selection routine is a part of control design.

Hence, there will be generally 4 vectors available as outputs of the driving cycle block. These outputs are as the following:

1. $T_z.mat$: Time vector
2. $V_z.mat$: Speed vector
3. $G_z.mat$: Gear shift vector
4. $D_z.mat$: Acceleration vector

There is also the possibility to define your own driving cycles easily and add them to the toolbox library. To see a schematic diagram of the driving cycle generator see Fig. (A.2).

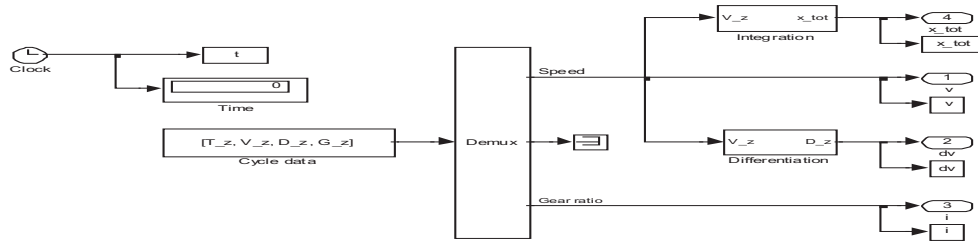


Figure A.2. Schematic presentation of the block "Driving Cycle"

This system is masked and only 3 parameters are available to be used to choose the driving cycle, the step size and if automatic simulation stop at the end of each driving cycle is allowed or not. See the mask in Figure (A.3). Note that the data shown as the mask parameters throughout this appendix are not necessarily the ones used in the simulations in previous chapters.

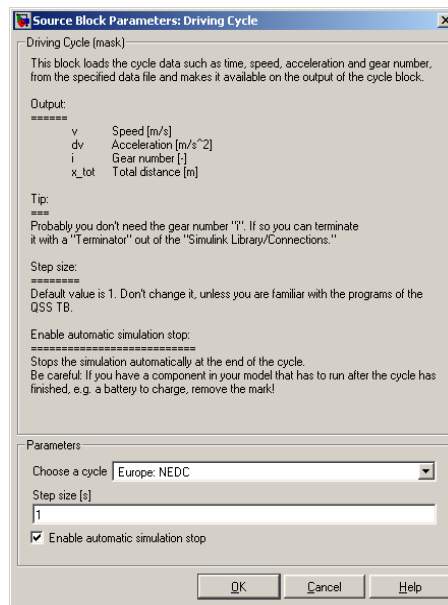


Figure A.3. The mask for the driving cycle block

A.2.2 Vehicle

The vehicle block actually converts the driving cycle speed and acceleration, or better say, the vehicle speed and acceleration into those of the wheels. This is done by utilizing all the forces included in Equation (4.1). The underlying system behind the vehicle block and the mask used in QSS Toolbox are shown in Figures (A.4) and (A.5), respectively.

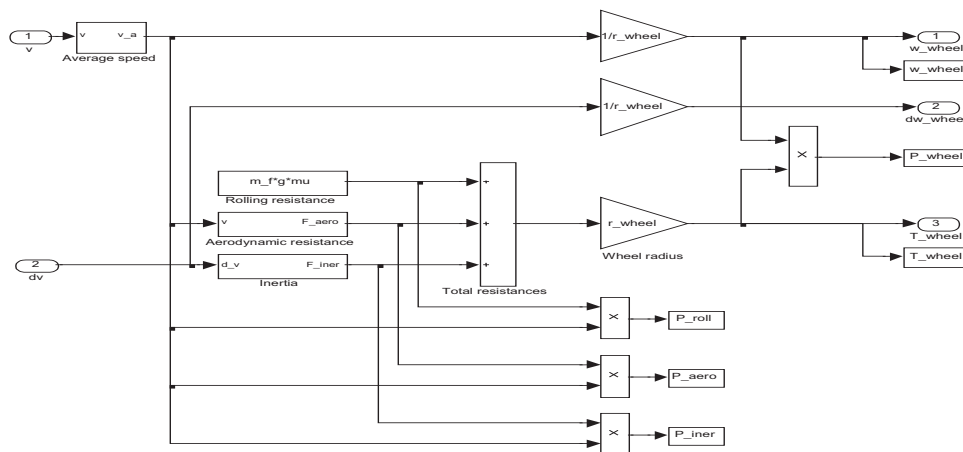


Figure A.4. Vehicle block - first level

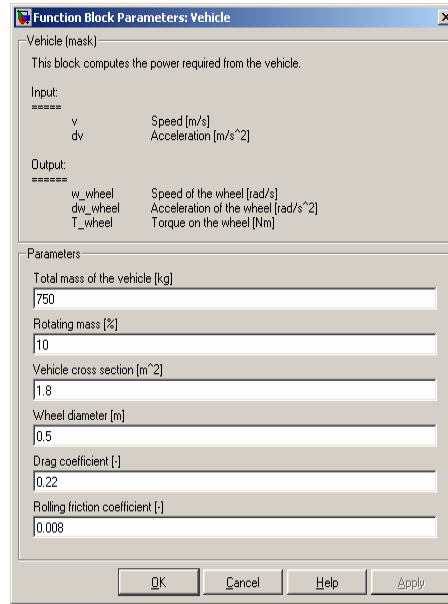


Figure A.5. The mask for the vehicle block

It must be noted that the vehicle inertia reflects the mass of the vehicle without wheels plus the wheel inertia data (parameter "Rotating mass" in the mask), as follows:

$$m_f + 4 \cdot \theta_{wheel} / r_{wheel}^2 \quad (\text{A.2})$$

Notice also that any additional inertia such as engine, flywheel, etc., are included in their own blocks.

A.2.3 Manual Gear Box

The top schematic diagram of Manual Gear Box is shown in Figure (A.6). It is possible to define fixed gear ratios (5 stepped gear in our case) in this block and to shift among them. The block "Power Flow" allows the power to be flown bidirectionally, i.e., from/to the wheels and this fact allows the simulation to flow the energy in both directions. In our case, this can be very useful when dealing with one single electric machine to work in motor or generator modes and be connected to only one single gearbox.

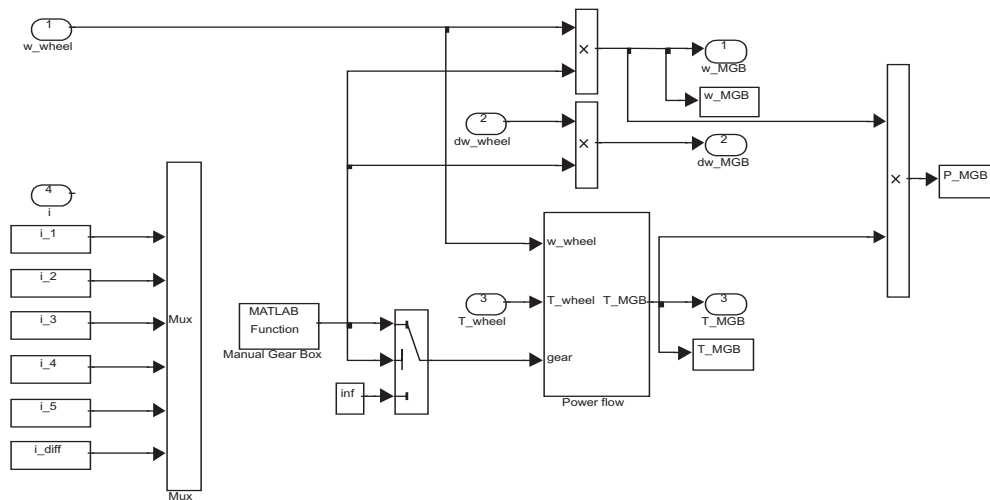


Figure A.6. Top level of the block "Manual Gear Box"

The masked parameters for manual gear box block are shown in Figure (A.7).

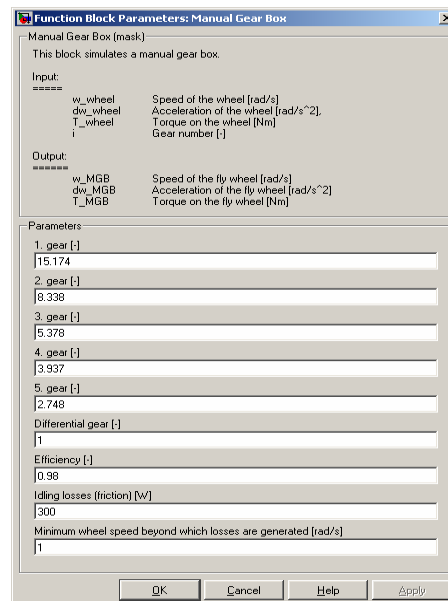


Figure A.7. The mask for the vehicle block

A.2.4 Combustion Engine (based on consumption map)

The combustion engine model is located under "Energy converter" block in the QSS Toolbox. This engine model is based on a static fuel consumption map, i.e., the engine power is calculated from a look-up table consisting of the fuel consumption of the engine based on given torque and speed at every sample. There are also some constraints on engine speed and torque as well that can be seen in Figure (A.8) as a block named

quadrants, i.e., motor/generator mode. Top level model of the electric machine is shown in Figure (A.10). As can be seen in this figure, there are constraints on the speed and torque of the electric machine modelled in the block "Detect overload and over speed". The mask parameters can also be found in Figure (A.11).

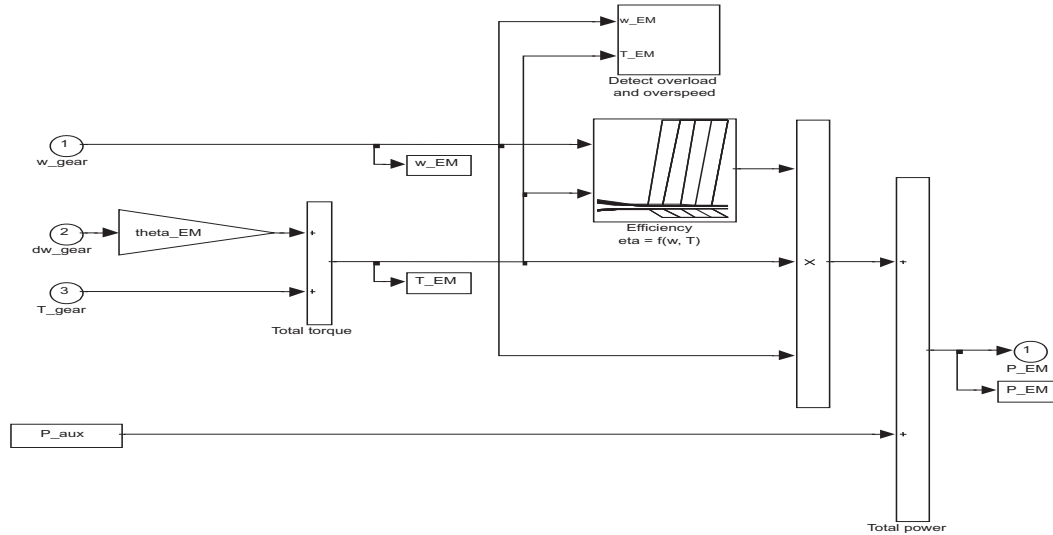


Figure A.10. Top level of the model "Electric Motor"

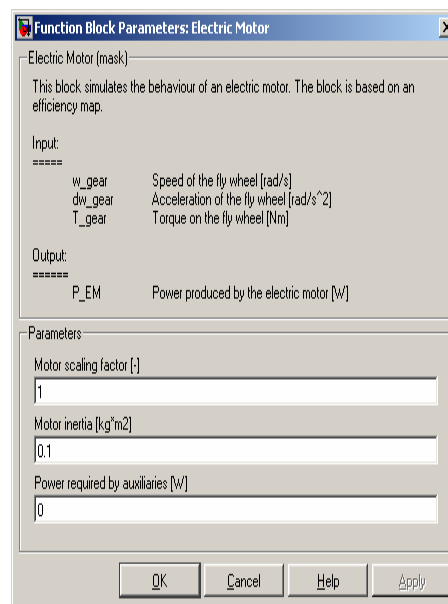


Figure A.11. The mask for the electric motor block

A.2.6 Battery

Battery model is given under the "Energy Buffer" block. This is a dynamic model as shown in Figure (A.12) containing an integrator to makes the battery charge calculations

possible. The electric power flows into the battery (negative when loading and positive during discharge) plays the role of input to the battery and the battery actual charge as well as the battery voltage are outputs of the battery block as also shown in the schematic diagram of the battery in Figure (A.12).

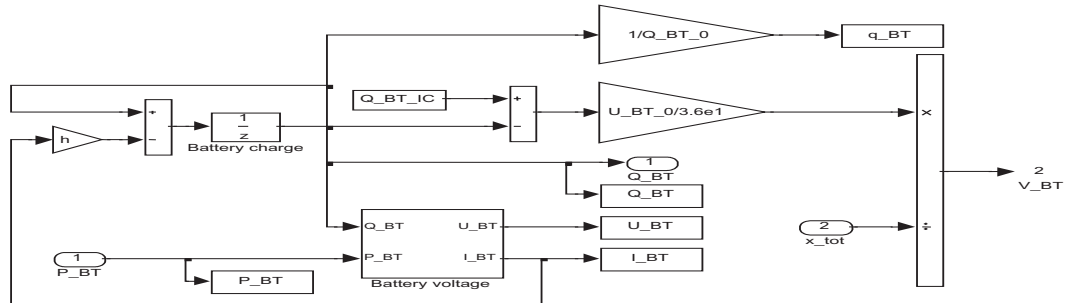


Figure A.12. Top level of the model "Battery"

There is a gray shaded block in Figure (A.12) that represents the model to calculate the amount of electric power (in kWh/100 km) drawn up to a certain instance (Guzzella and Amstutz 2005). There are a couple of constraints also taken care of under the shaded block on battery max. current and voltage. The mask parameters of the battery block are also shown in Figure (A.13).

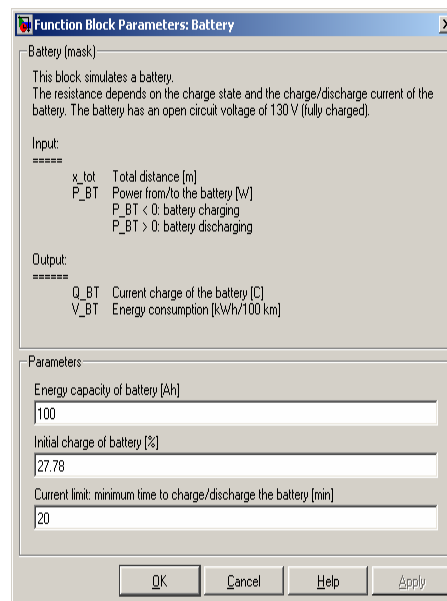


Figure A.13. The mask for the block "Battery"

A.2.7 Tank

Tank block models the fuel source of the vehicle that supplies the combustion engine with the demanded fuel. As the quasi-static convention of the QSS Toolbox, the inputs to the tank block are the power required from the tank and the total distance that must be travelled and the output that is the fuel consumption over 100 km will be calculated as shown in Figure (A.14). The parameters for the tank are also masked as shown in Figure (A.15).

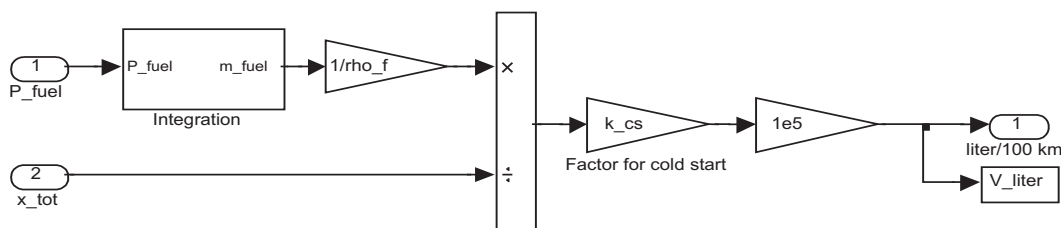


Figure A.14. Top level of the model "Tank"

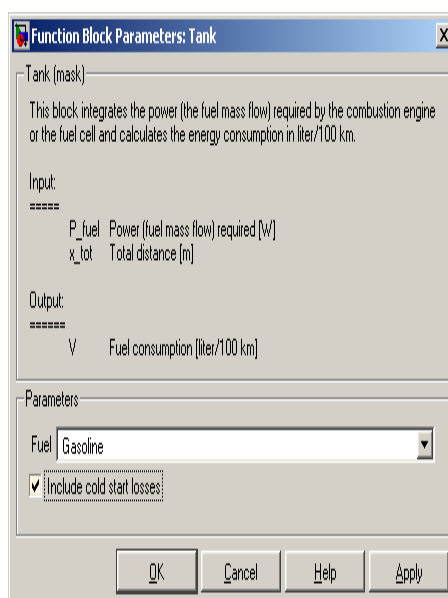


Figure A.15. The mask for the block "Tank"

A.2.8 Controller (empty template)

The empty template for the controller is chosen in this thesis basically because the controller is a brand new design specified for this application in the thesis work. Hence, there is not much to talk about based on the QSS Toolbox on this block. Instead the design and structure of this block is brought in Chapter 7 when describing the toolbox designed for this thesis work.

For further information about QSS Toolbox refer to (Guzzella and Amstutz 2005).