Series Hybrid Powertrain Optimization and Control

M.B. Arnolds
DCT Report number: 2005-26
March 2005

TU/e Master Thesis Report
March 2005

Supervisors:

Prof. dr. ir. M. Steinbuch
dr. P. A. Veenhuizen

Eindhoven University of Technology
Department of Mechanical Engineering
Section Dynamics and Control Technology

ir. M. Rondel

TNO Automotive, Delft, The Netherlands
Advanced Powertrains
Abstract

Hybrid Electric Vehicles receive a lot of attention in the automotive branch nowadays. Some manufactures have already launched vehicles with multiple power sources. However, none of them are of the so-called series topology. For research purposes, TNO Automotive in Delft, the Netherlands, has developed and build a prototype of a series hybrid powertrain. For this powertrain, TNO believes that the present control structure can be optimized to minimize fuel consumption.

Before a new control strategy can be designed, a benchmark has to be set. This benchmark will be the optimal fuel consumption that can be obtained for certain drive-cycles or drive-situation. During this project, performed at TNO Automotive, a search for a suitable optimization algorithm has been performed and three different accumulator types have been analyzed in order to set a benchmark. A powerful dynamic programming algorithm has been designed for this specific problem whose results, showing a significant fuel reduction, will be very useful for the design of the proposed control structure.
# Contents

Abstract i

1 Introduction 1
   1.1 Hybrid Carlab 2
   1.2 Purpose of the project 4
   1.3 Project Outline 4

2 Literature investigation 5

3 Modelling 7
   3.1 Model 7
   3.2 Benchmark 11
   3.3 Optimizing the control structure 11
   3.4 Summary 14

4 Sequential Quadratic Programming 15
   4.1 Fmincon 15
      4.1.1 (Quasi-)Newton’s Method 15
      4.1.2 Sequential Quadratic Programming 16
      4.1.3 Line-search 16
   4.2 Disadvantages 17
   4.3 Implementation of Fmincon 17
   4.4 Results 19
   4.5 Summary 19

5 Simulated Annealing 20
   5.1 Example of Simulated Annealing 21
      5.1.1 Example test results 22
   5.2 Implementation of SA in the Optimization Problem 23
      5.2.1 Initialization 24
      5.2.2 Scheduling of the SA algorithm 24
      5.2.3 Simulations 25
   5.3 Summary 27
6 Dynamic Programming

6.1 Bellman’s Principle of Optimality

6.1.1 Basic idea of DP

6.2 Design of the DP algorithm

6.2.1 Control variable

6.2.2 Battery model

6.2.3 Boundaries on the control variable

6.2.4 State-Cost-Matrix "U"

6.2.5 Optimal Path

6.3 Results for MVEG cycle

6.4 Different drive-cycles

6.4.1 USFTP72-cycle

6.4.2 Modem cycle

6.5 Operating points

6.6 Summary

7 Introducing a super-capacitor

7.1 "Super-battery"

7.1.1 Results for the super-battery

7.2 Super-capacitor model

7.3 Initialization

7.4 Results

7.5 Summary

8 Proposal for a new control strategy

8.1 Driving Pattern Recognition

9 Conclusions and Recommendations

9.1 Conclusions

9.2 Recommendations

A Results MVEG-cycle

B Results USFTP72-cycle

C Results Modem-cycle

D Graphical Results of DP
Chapter 1

Introduction

One of the greatest environmental problems nowadays is air pollution, causing global warming effect. One of the largest remaining contributors to air-pollution are the fossil fuel-burning vehicles. Though the emissions and fuel consumptions have been reduced the last years, the amount of vehicles still causes environmental pollution. Electric vehicles are seen as a long term solution to the problem, but the present battery technology is not capable yet to satisfy the desired distance range. Hybrid powertrains have originally been introduced to compensate for the shortcoming of the battery technology. Nowadays, this kind of powertrain has become a serious alternative for the "conventional" vehicles. Hybrid powertrains can reduce the air pollution in the cities and are more fuel efficient than conventional vehicles.

Like the name hybrid powertrain implies, it uses two or more power sources to propel the vehicle. The most common used power sources are an internal combustion engine (ICE) and an electric motor/generator (EMG). This combines the advantage of a conventional vehicle (regarding distance range) with some of the environmental benefits of an electric vehicle. The power sources can be arranged in different topologies:

- **Parallel Hybrid Electric Vehicle (PHEV):** both the ICE and the EMG are mechanically connected to the drive-shaft. Both power sources can propel the vehicle independently or assist each other.

- **Series Hybrid Electric Vehicle (SHEV):** only the EMG is connected to the drive-shaft. The ICE is used here to supply the electrical energy which will be stored in an accumulator or will be sent to the EMG directly, depending on the speed of the vehicle and demand for power of the EMG.

- **Combined HEV:** This topology integrates the merits of series and parallel topologies. The most typical combined topology is the split subtype based on a planetary gear set. This splits the engine power to a generator or to the wheels with variable share. It is also possible to propel the vehicle purely electrically. This powertrain is very clever but also complex; the Toyota Prius is an example of where this kind of topology has been build in.

One of the largest environmental advantages of a hybrid electric vehicle is the concept of regenerative braking. In a conventional vehicle, the desired speed is controlled by the driver by depressing the accelerator pedal for a higher speed (positive torque delivered by the combustion engine) and depressing the brake pedal for a lower speed (negative torque delivered by the brakes). The energy that is present in negative torque can be
seen as lost energy. In a hybrid vehicle, to slow down, the driver releases the accelerator pedal or pushes the brake pedal, causing the electric motor to function as a generator, slowing down the vehicle and using as much as possible the kinetic energy to induce a current that recharges the battery. This "free" energy can now be used for future energy demand. By regenerating braking energy, the fuel consumption will decrease. Mechanical brakes are still needed for safety reasons.

Another advantage of the hybrid powertrain is that because of the use of two energy sources, the ICE can be downsized, and above all, it can be operated in its most efficient operating points.

Besides the advantages a HEV can bring, it also brings disadvantages with it. One of them is the maximum vehicle speed. The maximum vehicle speed generally defines the maximum power needs (rated, not peak power). This power can not be provided by the accumulator over a long range, so the ICE needs to provide the necessary power. The bigger the ICE, the worse the HEV achievements are. A PHEV and Combined HEV can overcome this problem for a part but this will be paid with a slightly reduced fuel consumption. A SHEV is strongly depending on the size of the accumulator and the ICE for this problem. Therefor, the SHEV is very suitable for neighborhood cars and buses.

1.1 Hybrid Carlab

At TNO-Automotive in Delft, the Netherlands, a Serial Hybrid Electric Vehicle has been developed for research purposes. The Hybrid Carlab, as it is called, is a Volkswagen NewBeetle with a hybrid powertrain. The vehicle is equipped with:

- an electric motor 150 kW peak / 50 kW continuous,
- a 256 cell Nickel Metal Hydride battery pack,
- a light and compact 40 kW generator set formed by an ICE and a generator

Figure 1.1 shows a schematic view of a series hybrid electric vehicle with its energy flows.

![Figure 1.1: Energy flows in a Series Hybrid configuration](image)

The Hybrid Carlab has been realized in december 2002. After rebuilding the original NewBeetle the total mass of the vehicle increased with about 250 kg to a total mass of 1498 kg. The EMG drives the front wheels through a fixed ratio transmission. It is an AC Propulsion air-cooled 3-phase induction motor. The characteristics of this EMG yields a maximum torque and power of 220 Nm and 150 kW respectively. Although
the peak power of the motor is rather high, it is limited by the maximum power of the battery and the generator set. The terms generator set and genset will be used alternately in this report but refer to the same thing. The set formed by the ICE and the generator.

The NiMH battery pack is located where once the rear seats were. It consists of 256 NiMH cells of 10 Ah. It has a nominal voltage of 320 V and an effective capacity of approximately 2 Ah within a defined State of Charge (SOC) window. The peak power of (dis)charging is 50 kW and the maximum current is 200 A.

The generator set consists of a turbo diesel engine and a permanent magnet generator that are connected to each other through a flexible coupling. This flexible coupling isolates the generator from the engine’s torque fluctuations. The engine is a 1.2l TDI 3-cylinder diesel engine with turbo charger. This engine is also known as the 3l/100km engine of a Volkswagen Lupo. The maximum torque is 140 Nm at 1800 rpm and the maximum power is 45 kW at 4000 rpm. The engine has been chosen on the basis of criteria as: efficiency, mass, dimensions, power and emissions.

The generator is a permanent magnet and has been especially prepared to match the diesel engine so the maximum generator power matches the maximum engine power of 45 kW at 4000 rpm.

The master control unit is an automotive controller called MACS (Modular Automotive Control System). This unit is especially developed by TNO for rapid control prototyping purposes. The controller consists of an Application Processing Module (APM) based on an MPC555 controller with Osek operating system and several I/O modules based on an automotive ECU (Engine Control Unit). The APM is programmed with the control algorithms that are designed in Matlab/Simulink and compiled with the Real Time Workshop. Data communication is realized through 4 CAN buses.

A schematic picture of the control system is shown in Figure 1.2.

![Figure 1.2: Schematic picture of the control system](image)

The main function of the powertrain management is to control the power flows within the powertrain. This means the controller has to control the power flows from and to the battery, coming from the genset or the electric motor. The most important power
flow to control is the electrical output of the genset. This is the variable that influences the fuel consumption. So in order to obtain the lowest possible fuel consumption, the power generated by the genset has to be minimized. The SOC of the electric accumulator is a very important factor/constraint here. Since it is not desirable to deplete the accumulator this level has to be maintained by power flows from the genset and/or regenerative braking. Also limitations of the power demand at the wheels, $P_{\text{drive}}$, are introduced to avoid accumulator depletion (limited maximum power that can be asked).

1.2 Purpose of the project

The present control of the system is a straightforward proportional control algorithm. Roughly, one can say that for every percent difference in the SOC-level (regarding a certain defined desired SOC level) the genset has to generate 1 kW of electric power. It is a load-follower strategy. Low powers are drawn from the battery. When more power is required (i.e. when accelerating to high speeds) or when the SOC-level of the battery becomes too low, the genset will switch on, so the battery will not be depleted. So a desired SOC-level is introduced which will be maintained over a (given) drive-cycle so the battery will not be depleted. This control law works properly, though it could be optimized.

The goal of this thesis is to find a way to define an optimal control strategy for online implementation in a series hybrid powertrain, to reduce the fuel consumption while maintaining the SOC-level. In other words, the power flow management system has to be controlled and optimized, preferably online and during real life unknown drive-cycles. To be able to define an optimal control strategy, the optimal fuel consumption must be found to serve as a benchmark. Off-line optimization will have to provide such a benchmark and possibly a way to define the corresponding strategy. As one of many possible applications of series hybrid powertrains, the Hybrid Carlab will be used as the application during this project.

1.3 Project Outline

First of all, the model of the series hybrid powertrain will be discussed (chapter 3). In this chapter, the extended model will be reduced to a simple model that can be used for optimization purposes. The emphasis of this project will lie on the optimization of the power flow. Different optimization techniques will be introduced, implemented and discussed (chapter 4, 5 and 6). During the project, some changes regarding the accumulator of the Hybrid Carlab will be made. The battery-pack, mentioned earlier, will be replaced by super-capacitors. So besides optimizing the power flow for the battery, also the configuration with the super-capacitors will be optimized (chapter 7). In the end, a proposal for an optimal control law will be discussed, conclusions will be drawn and proposals for future research will be given.
Chapter 2

Literature investigation

The project starts with searching the literature for already existing control strategies for hybrid powertrains. Most of the literature that has been found focused on the parallel topology and not on the series topology.

However, these articles could very well be used to obtain ideas of control techniques that are able to control hybrid powertrains. This can be explained by the fact that despite the difference in topology, the goals of the two different types of powertrains are the same: reduce fuel consumption and reduce emission while maintaining the SOC-level of the accumulator.

Amano et.al. [Aman-2003] describe the introduction of indices: the Driving index and Charge index. These indices calculate the optimal operating points for certain situations during a known drive cycle. It is possible to switch between these optimal situations. Although, it is explained for known drive cycles, they assure that with a few adjustments it should work for real life unknown drive cycles.

Another article, [Jons-2003], investigates the use of Model Predictive Control (MPC). With the help of the past and present it is possible to predict the future and the controller can adapt to it. It is an application which is large-scaled applied in the Process Industry. For this case it is very hard to predict the future since no information will be present of i.e. the driver behavior, uphill/downhill driving, traffic jam, road turns etc. Also, the MPC-application in the Process Industry has a large time-constant (i.e. hours, days). This is certainly not the case within this kind of control where typical time-constants are in the order of seconds. Above all, a MPC requires a lot of calculating power. Despite these disadvantages, it is possible to use the model predictive controller for offline use and investigation.

A train of thought has been to combine these two techniques. So if the indices are determined using different known drive cycles and are stored in a control sequence and the MPC is used in real life drive cycles for determining the tendency of the past few minutes. Then it is possible to simply determine whether the vehicle is driving at a highway or in city traffic. Once this knowledge is present, it is possible to switch between different optimal situation of the control. With this combination, a lot of calculating power is already performed offline and the partial MPC can work online. It does not have to predict the complete future but only in what kind of tendency the vehicle is driving. From this tendency it must determine the optimal index.

A kind of this idea is described in [Lin-2002-1]. They describe a Driving Pattern Recognition (DPR) method which is used to classify the current driving pattern into one of the, in advance, calculated Representative Driving Pattern (RDP). Once a RDP
is selected, a proper control algorithm can be chosen. The RDPs are designed based on the driving characteristics.

Other control strategies that can be found look like the load-following structure that is present in the Hybrid Carlab, thermostatic control laws and rule based control laws.

Another literature research has been done on optimization techniques. This research is integrated into the project line. This will become clear while reading this report. It starts with the investigation of a Sequential Quadratic Programming technique. More about this as from chapter 4.
Chapter 3

Modelling

Before an optimization of the fuel consumption can be performed, a model that calculates this fuel consumption has to be available. TNO has developed a Matlab/Simulink tool, called Advance, in which a model of the Hybrid Carlab is available. However, this model is very detailed and calculates with a lot of variables that are not interesting for the model that is necessary for this project. Therefore, a new model, based on this Advance-model, will be developed which will only calculate with and give the information that is necessary. This modelling will cover the first part of this chapter. In the second part, the obtained model will be used to provide a benchmark which will form a guideline throughout the rest of the report. In the end, a first optimization step, leading to a second benchmark, will be discussed.

3.1 Model

The basis of the model will be formed by the power balance, equation (3.1).

\[ P_{\text{drive}} = P_{\text{generator set}} - P_{\text{accumulator}} \]  
(3.1)

Note that the negative sign in Equation 3.1 is a consequence of the definition of \( P_{\text{accumulator}} \). A negative sign means the accumulator is giving energy, and a positive sign yields that the accumulator is absorbing energy.

This balance will divide the desired power to drive \( (P_{\text{drive}}) \) into the two variable powers \( P_{\text{genset}} \) and \( P_{\text{accumulator}} \) in such a way that the power balance and, later on, the energy balance will be correct.

This basic equation, modelled in Simulink, is shown in Figure 3.1.

![Modelled power balance, the basis of the model](image)

Figure 3.1: Modelled power balance, the basis of the model
As can be seen in the Figure, several parameters are required in this part. First of all, the desired $P_{\text{drive}}$ has already been calculated once with the *Advance* model. Second, the generator set power (including losses) is calculated on the basis of a certain torque ($T$) and engine speed ($\omega$) of the generator set. The calculation of these $T$ and $\omega$ will be discussed later on in this chapter.

An overall demand during the optimization that will be performed in this project will be that over a (specific) drive-cycle, the difference in State of Charge ($d\text{SOC}$) must be equal to zero. $d\text{SOC}$ is defined as the initial SOC minus the SOC at the end of a cycle. So,

$$d\text{SOC} = \text{SOC}_{\text{begin}} - \text{SOC}_{\text{end}} \quad (3.2)$$

and

$$d\text{SOC} = 0 \quad (3.3)$$

This demand is necessary because of the fact that otherwise the battery would be depleted in order to obtain an as low as possible fuel consumption. Although, this last fact is desirable, it is not convenient since a depleted battery is taking the vehicle nowhere. Besides that, a real view on the costs is preserved since the energy in the battery is also generated using fuel (and regenerative braking) and this way it is possible to compare different drive-cycles. The SOC will be calculated with a accumulator model, or battery model, that is present in *Advance* and will in this model be considered as a black-box battery model (see Figure 3.2). A power (resulting from the power balance) and an initial condition are inputs and the SOC is the output.

One of the advantages of a series hybrid electric vehicle is that the ICE can be utilized in its most efficient operating points, all to be found on the Optimal Operating Line (O.O.L.) of the engine. To calculate the necessary torque ($T$) and engine speed ($\omega$) this O.O.L. will be utilized using look-up-tables which are placed in the subsystem in Figure 3.3. A certain desired $P_{\text{generator set}}$ as an input results in the corresponding optimal $T$ and $\omega$ as the output.
The most important part in this model will be the calculation of the desired $P_{\text{genset}}$. This calculation will influence the amount of energy that will be generated by the generator set and so, conform to the power balance, the amount of energy that will be delivered by the accumulator. To calculate the desired $P_{\text{genset}}$, the following function will be introduced. This function represents the present control of the powertrain and will be called the control structure (CS) or just structure.

\[ P_{\text{genset}} = c_{eSOC} \cdot eSOC + c_{P\text{drive}} \cdot P_{\text{drive}} \]  

(3.4)

This structure calculates the desired genset power on behalf of the error in the State of Charge of the battery ($eSOC$) and the desired electric drive power ($P_{\text{drive}}$). $eSOC$ is defined as the error of the value of the SOC regarding the reference value of the SOC during a drive cycle. The parameters $c_{eSOC}$ and $c_{P\text{drive}}$ have been introduced to be able to investigate the influence of the two different inputs.

Besides the structure parameters, several other parameters have been added to the calculation of the generator set power. These parameters will tell the genset when to turn on and when to turn off. Hysteresis is involved in this. The genset is not allowed to turn on before a power of 14 kW is asked from it ($c_{\text{genset on}}$). Once turned on it has to stay on until a lower limit of 10 kW has been reached ($c_{\text{genset off}}$). Second, the genset has to stay on for at least 18 seconds. If the lower limit of 10 kW is reached within 18 seconds ($c_{\text{min time on}}$) the genset should be kept running, delivering a power of 10 kW ($c_{\text{border power}}$). This time-delay has been build-in for several reasons. First of all, starting the engine requires fuel. Constantly switching on and off the engine will not be beneficial for the fuel consumption. Besides that, the turbo must have time to startup and to shut down. If the genset is constantly switching on or off, the life-cycle of the turbo will not be very long. As a drivability aspect, constantly turning on and off will not influence the drivability in a positive way. The generator set is able to generate a maximum of 30 kW of power ($c_{\text{max genset power}}$). Hence,
• $c_{\text{genset on}}$ = Minimum value for the genset to turn on
• $c_{\text{genset off}}$ = Maximum value for the genset to turn off
• $c_{\text{min time on}}$ = Minimum time the genset has to stay on
• $c_{\text{border power}}$ = Minimum power delivery for genset when $T < c_{\text{min time on}}$
• $c_{\text{max genset power}}$ = Maximum power the genset is able to generate.

Finally, the fuel consumption has to be determined by using a look-up table where the obtained torque $T$ and engine speed $\omega$ result in a fuel consumption per second. By integration, the total fuel consumption over the given drive cycle will be obtained (see Figure 3.5).

Now every part of the model has been discussed, modelled in Simulink and ready to be connected to each other forming the total model presented in Figure 3.6.
3.2 Benchmark

The obtained model of the powertrain will be used to provide a benchmark. This benchmark will be the fuel consumption that has been obtained with the present control structure and will be used as a guideline throughout the rest of the project to be able to see whether the utilized optimization method is able to find a lower fuel consumption than the present control structure. The reason that this model shall provide a benchmark in stead of the *Advance* model lies in the fact that this model shall be used during the optimization further on in this report. This avoids comparing two different things with each other.

Initially, all the parameters have been set to the values as they appear in the *Advance* model. This way, it represents the present control structure completely. These parameters have been obtained by calculations and partially by trial and error.

- $c_{eSOC} = 1$
- $c_{Pdrive} = 1$
- $c_{genset\ on} = 14$ [kW]
- $c_{genset\ off} = 10$ [kW]
- $c_{min\ time\ on} = 18$ [s]
- $c_{border\ power} = 10$ [kW]
- $c_{max\ genset\ power} = 30$ [kW]

The overall demand $dSOC = 0$ must be satisfied. A pragmatic way to achieve this is to run a simulation and determine the end value of the SOC for this simulation. Using this end value again as the initial value for a second simulation will lead to a $dSOC = 0$ for the second simulation. The fact that it works like this can be assigned to the proportional action that is present in the control structure.

The obtained fuel consumption for, e.g., the MVEG cycle is 391 gram (4.32 l/100 km) over the entire cycle while $dSOC = 0$. The corresponding behavior of the generatorset (exactly like is prescribed by the control structure) and the SOC deviation over the drive-cycle are shown in Figure 3.7.

3.3 Optimizing the control structure

Now that the benchmark for control structure (CS) as it appears in the present control of the powertrain has been set, a first optimization can be performed. The optimization problem can be written as:

$$\min_{f_c} f(x(t))$$

with:

$$f(x(t)) = \int_{t=0}^{n} f_c(x(t))dt$$

Where $f_c$ represents the fuel consumption, $n$ the length of the drive-cycle and $x(t)$ represents the control variable which in this case is set as $P_{genset}$ and depends on the parameters $c_{eSOC}$ and $c_{Pdrive}$. 
The principle of this optimization is simple; vary $c_{eSOC}$ and $c_{P_{\text{drive}}}$ and find the combination of these two that leads to a minimal fuel consumption while maintaining the SOC and satisfying all the other constraints.

$c_{eSOC}$ and $c_{P_{\text{drive}}}$ have been varied within the limits:

\begin{align*}
  c_{eSOC} &= 0.1 : 0.1 : 2.5 \\
  c_{P_{\text{drive}}} &= 0.1 : 0.1 : 2.0
\end{align*}

The result of this optimization for the MVEG-cycle is that the minimum fuel consumption can be found for:

\begin{align*}
  c_{eSOC} &= 2.0 \\
  c_{P_{\text{drive}}} &= 0.6
\end{align*}

The obtained fuel consumption has reduced to 387.7 gram (4.28 l/100 km), which is a reduction of only 0.85 %. The results show that, when calculating $P_{\text{genset}}$, the contribution of the error in the SOC ($c_{eSOC}$) is more important than the contribution of the demanded drive power ($c_{P_{\text{drive}}}$), where their contributions in the non-optimized control structure are equal. This means that it is beneficial for the control structure to focus on the error in the SOC and that the contribution of $P_{\text{drive}}$ is not as important as it has been thought it would.
CHAPTER 3. MODELLING

The results for the generator set behavior and the SOC are shown in Figure 3.8.

Figure 3.8: Generator set behavior and SOC for the optimized structure

Together with the fact that the reduction of the fuel consumption is not that large, there is not much difference between the behavior of the generator set and the corresponding SOC path. The energy balance is very useful while analyzing the behavior of the different power sources. The energy balance is given by:

\[ E_{\text{drive}} = E_{\text{genset}} - E_{\text{accumulator}}, \tag{3.7} \]

where:

\[ E = \int_{t=0}^{t} P \, dt \tag{3.8} \]

The resulting total energies for this optimized structure parameters are given in Table 3.1 below.

<table>
<thead>
<tr>
<th>Energy balance for optimized structure (MVEG)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( E_{\text{drive}} )</td>
<td>5.59 MJ</td>
</tr>
<tr>
<td>regen.eng</td>
<td>-1.21 MJ</td>
</tr>
<tr>
<td>demand.eng</td>
<td>6.80 MJ</td>
</tr>
<tr>
<td>( E_{\text{genset}} )</td>
<td>5.73 MJ</td>
</tr>
<tr>
<td>( E_{\text{accumulator}} )</td>
<td>0.14 MJ</td>
</tr>
<tr>
<td>charge</td>
<td>2.36 MJ</td>
</tr>
<tr>
<td>discharge</td>
<td>2.22 MJ</td>
</tr>
<tr>
<td>battery efficiency</td>
<td>93.7 %</td>
</tr>
</tbody>
</table>

Table 3.1: Energy balance for optimized structure (MVEG-cycle)

From the energy balance the efficiency of the accumulator can be determined. In this case, the efficiency is 93.7%. Also, 48% of the charge energy is delivered by regenerative
braking (negative $P_{\text{drive}}$), meaning that the remaining 52% is delivered by the generator set which in turn is 20% of the total generated energy by the genset. The other 80% is used to satisfy the power demand of the vehicle directly.

### 3.4 Summary

In this chapter a representative model has been designed and implemented which has to replace the detailed and complicated model that is has already been present. This model reduces calculation time while giving only the necessary and desired information.

From the model a fuel consumption over a given drive-cycle has been derived. This fuel consumption will serve as a benchmark during the entire project.

As an initial optimization step, the parameters $c_{\text{SOC}}$ and $c_{P_{\text{drive}}}$ have been varied in order to obtain a combination of these parameters that leads to a lower fuel consumption. Though, a lower fuel consumption has been obtained, the reduction is not large. The resulting $f_c$ will be taken along the project as a second benchmark.

<table>
<thead>
<tr>
<th></th>
<th>FC [gr]</th>
<th>FC [l/100km]</th>
<th>Reduction [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial structure parameters</td>
<td>391</td>
<td>4.32</td>
<td>-</td>
</tr>
<tr>
<td>Optimized structure parameters</td>
<td>387.7</td>
<td>4.28</td>
<td>0.85 %</td>
</tr>
</tbody>
</table>

It is for sure that the fuel consumption can be reduced even more. More parameters, like ($c_{\text{genset on}}$, $c_{\text{genset off}}$, $c_{\text{border power}}$), could be varied to obtain a lower fuel consumption but besides the fact that it will result in a lot of calculations, the emphasis of the optimization during the project will lie on the optimization independent to the structure. Therefore, another optimization method has to be investigated and implemented which is able to find the optimal behavior of the generator set which leads to the minimum fuel consumption.
Chapter 4

Sequential Quadratic Programming

During a literature study, the optimization algorithm Fmincon came up as a method that is able to handle (non)linear problems with constraints ([Strom-2003], [Cole-1999], [Papa-2000]). The fact that it can handle constraints is very useful in this project. For example, the SOC has to stay within desired boundaries and the $dSOC = 0$ demand has to be satisfied. Fmincon is an optimization routine that is already implemented as a function in Matlab. The goal of the optimization is to find the optimal fuel consumption for a given drive-cycle. Therefor the fuel consumption has to be minimized:

$$
\min_{fc} \int_{t=0}^{n} \dot{fc}(x(t)) dt \tag{4.1}
$$

where $x(t)$ is the control variable and the minimization has to satisfy certain constraints. This will be discussed as from section 4.3. First the principles and functioning of the algorithm must be clear. This is described in the first part of this chapter.

4.1 Fmincon

Fmincon is a Sequential Quadratic Programming (SQP) method which is used to find the minimum of an objective function. This method allows one to closely mimic Newton’s method for constrained optimization just as is done for unconstrained optimization. At each major iteration an approximation is made of the Hessian of the Lagrangian function using a quasi-Newton update method. This is then used to generate a Quadratic Programming (QP) subproblem whose solution is used to form a search direction for a line-search procedure.

4.1.1 (Quasi-) Newton’s Method

Newton’s method calculates the Hessian ($H$) directly from the gradient of the objective function and proceeds in a direction of descent using line-search methods to locate the minimum after a number of iterations. This involves a large amount of calculations for $H$ and therefor a Quasi-Newton’s method can be used. This method avoids this large
amount by using the observed behavior of $f(x)$ and the gradient of $f(x)$ ($\nabla f(x)$) to build up curvature information. The gradient of $f(x)$ is defined as:

$$\nabla f(x) = \frac{\partial f}{\partial x_i}$$

(4.2)

To make an approximation of $H$ an appropriate updating technique is utilized. The most common used is the BFGS-method (Broyden-Fletcher-Goldfarb-Shannon). This method approximates the Hessian by calculating:

$$H_{k+1} = H_k + \left( \frac{\delta g \cdot \delta g^T}{\delta g^T \cdot \delta x} \right) - \left( \frac{(H \cdot \delta x) \cdot (H \cdot \delta x)^T}{\delta x^T \cdot H \cdot \delta x} \right)$$

(4.3)

where:

$$g = \nabla f(x)$$

(4.4)

$$\delta g = g_{k+1} - g_k$$

(4.5)

$$H_0 = I$$

(4.6)

The Hessian, $H$, is always maintained to be positive definite so that the direction of search is always in descent direction. This means that for a certain step $\alpha$ in direction $d$, the objective function decreases in value. This positive definiteness of $H$ is achieved by ensuring that the Hessian is initialized positive definite and thereafter $\delta g^T \cdot \delta x$ is positive.

### 4.1.2 Sequential Quadratic Programming

As a sub-optimization problem, a Quadratic Programming problem is utilized to find a suitable search direction. In this QP problem, a quadratic approximation of the augmented objective (Quasi-Newton method) is minimized while subjected to the constraints.

$$\min_d \left( \frac{1}{2} d^T H_k d + \nabla f(x_k)^T d \right)$$

(4.7)

Here, $H$ is updated, as described in the previous section, with a quasi-Newton method. When this subproblem is solved the solution (a search direction $d$) is used to form a new iterate:

$$x_{k+1} = x_k + \alpha_k \cdot d_k$$

(4.8)

The step length $\alpha_k$ is determined using an appropriate line search procedure so that a sufficient decrease in the objective function is obtained.

A nonlinear constrained problem is often solved in fewer iterations than an unconstrained problem when using SQP. This phenomenon can be explained by the fact that because of the boundaries on the feasible area, the optimizer can make well-informed decisions regarding directions of the search and step-size.

### 4.1.3 Line-search

Generally, one can define line-search as: the minimization along the line that passes though $x_k$ and has direction $d_k$ with step-size $\alpha_k$ which is performed every iteration of the optimization.
The purpose of line-search is to determine the value $x_k$ which ensures an acceptable decrease of the function value $f_k$ to $f_{k+1}$ or $f_k - f_{k+1} < 0$. During line-search the algorithm searches in one direction at a time for a lower objective function value. The next iteration step it will determine the search direction again. This could be the same direction but this does not have to be so.

So now the procedure for SQP with line-search can be given as:

1. initialize
2. solve QP-problem to obtain search direction $d_k$
3. determine step-size $x_k$ along $d_k$
4. set $x_{k+1} = x_k + \alpha_k d_k$
5. check for termination. If not satisfied, return to step 2

### 4.2 Disadvantages

Of course Fmincon has its disadvantages. One of them is that the algorithm does not always explicitly find the global minimum of the objective function. In this part of the project we want to find the optimal fuel consumption over a drive-cycle, satisfying the constraints. In most of the cases this is not the global minimum because our global minimum is located at zero engine speed ($\omega = 0$) and zero torque ($T = 0$) but overall, the algorithm has to find the behavior of the generator set leading to minimum fuel consumption.

The second disadvantage is more a point of attention. When working with look-up tables one has to be careful. Because these tables work with measurement data, the values in between are being interpolated. Though the function may be continuous, the derivative of the function around a certain value may not be (saddle-point). This is also called the monotonicity of a function. If the function is not monotonic increasing or decreasing, the algorithm could experience some trouble while calculating the Hessian (or the approximation of it). Simple experiments have been performed to check whether the algorithm is robust enough to deal with possible discontinuities.

### 4.3 Implementation of Fmincon

The goal of the optimization is to minimize the fuel consumption subjected to certain constraints. Since the fuel consumption depends on the behavior of the generator set, the control variable of the algorithm can be stated as the vector: $x(t) = P_{genset}(t)$.

The objective function must result in a fuel consumption over the given drive-cycle. The Simulink model described in Chapter 3 will be used as a reflection of the objective function. Given a certain $x(t)$, the model will calculate the corresponding fuel consumption like already stated:

$$f(x(t)) = \int_{t=0}^{t=n} \dot{f}(x(t))dt$$

(4.9)

The objective function will be subjected to constraints. These constraints are, besides the upper and lower boundary of the control variable ($x$), the minimum and maximum allowed SOC-level. These inequality constraints will ensure that the SOC will stay within a certain user-defined boundary. Besides the inequality constraints, one equality constraint exists. This is the overall demand of dSOC=0.
So the optimization problem can be written as:

\[
\min_{f_c} f(x(t)) \\
\text{subject to :} \\
g_1 : SOC_{\text{min}} \geq SOC_{\text{lowerbound}} \\
g_2 : SOC_{\text{max}} \leq SOC_{\text{upperbound}} \\
g_3 : P_{\text{genset}} \geq P_{\text{genset.lowerbound}} \\
g_4 : P_{\text{genset}} \leq P_{\text{genset.upperbound}} \\
h_1 : SOC_{\text{begin}} = SOC_{\text{end}}
\]
or in the so-called negative null from

\[
\min_{f_c} f(x(t)) \\
\text{subject to :} \\
g_1 : -SOC_{\text{min}} + SOC_{\text{lowerbound}} \leq 0 \\
g_2 : SOC_{\text{max}} - SOC_{\text{upperbound}} \leq 0 \\
g_3 : -P_{\text{genset}} + P_{\text{genset.lowerbound}} \leq 0 \\
g_4 : P_{\text{genset}} - P_{\text{genset.upperbound}} \leq 0 \\
h_1 : SOC_{\text{begin}} - SOC_{\text{end}} = 0
\]

Officially, the SOC should also be written in terms of the control variable \( x \). It is possible to rewrite them, but this is a lot of work while the SOC shall be calculated during the simulations. So their values will be obtained from the simulations just like the fuel consumption. Worth to mention is the fact that the SOC is depending on the control variable \( x(t) \) (or \( P_{\text{genset}} \)) by means of the energy balance that has to be correct. A certain \( P_{\text{genset}} \) will result in a certain \( P_{\text{accumulator}} \) given a specific \( P_{\text{drive}} \). The change in \( P_{\text{accumulator}} \) will result in a change of the SOC. This way, the variation of \( P_{\text{genset}} \) will influence the SOC.

The optimization problem described above is implemented into Matlab. For every point of the vector with genset powers, a new search for a better point is performed. So every time a new generator set power has been defined on the basis of \texttt{Fmincon}’s criteria, a simulation is performed to check whether the fuel consumption is decreased while satisfying the constraints. The obtained fuel consumption is taken from the optimal operating line of the generator set.

Furthermore, \texttt{Fmincon} requires an initial condition for the control variable so different initial conditions have been used.
4.4 Results

The results are rather disappointing. The algorithm seems not to be able to find an optimal solution at all. Although the algorithm satisfies the demand of $dSOC = 0$, the fuel consumption is not reduced as has been expected. In fact, it increased. Some of the results are shown in the figures below.

![Figure 4.1: Results for $P_{\text{genset}}$ using \text{Fmincon} with different initial conditions](image)

As one can see, the results look like a kind of ”grass”. The fuel consumptions corresponding to these results are 516 and 522 gram, respectively, over the entire drive-cycle (MVEG). This is an enormous increase regarding the benchmark. This is caused by the fact that the algorithm does not see the fact that it is profitable to turn the generator set off at some moments. These results have been obtained after a lot of tuning of the algorithm. Unfortunately, the efforts have not resulted in useful results. Even though when the obtained behavior of the generator set from the optimization with the structure (see section 3.3) is taken as the initial condition for the algorithm, the fuel consumption did not improve.

4.5 Summary

\text{Fmincon} proved to be an algorithm that is not able to solve the optimization problem that is stated for the project. Not only the amount of parameters to optimize (i.e. for the MVEG-cycle, 1180 parameters) is far too much, also the lack of a real objective function which can be used to solve the QP problem is playing a part. Because of this, the gradient is not clear and the approximation of the Hessian becomes very unreliable so no proper search directions and line-searches can be performed. It seems that this ”smart” optimization algorithm will only work for a reasonable finite amount of parameters where the objective function is nicely specified.

Although not much attention to the results of the algorithm is paid in this report, a lot of time is put in this algorithm to get it working. Unfortunately, without success. The problem is that the amount of parameters to optimize is very large and no specific objective function is specified. So another optimization algorithm has to be found that is able to deal with a large amount of parameters without using gradient information to smartly find a directions to search in.
Chapter 5

Simulated Annealing

In the preceding chapter, Fmincon proved to be incapable to solve the optimization problem. Searching the literature for new ways to solve a constrained optimization problem, the Simulated Annealing (SA) algorithm appeared ([Delp-1999] and [Moin-2002]).

Simulated Annealing is commonly said to be the oldest among the metaheuristics and surely one of the first algorithms having an explicit strategy to avoid local minima. This numerical optimization technique is based on the principles of thermodynamics and finds its origin in 1953 when the idea of SA has been published in a paper by Metropolis et al. ([Metr-1953]). The algorithm in this paper simulated the cooling of material in a heat bath where the aim is to obtain perfect crystallization by a slow enough temperature reduction to give the atoms the time to obtain the lowest energy state. This process is known as annealing. In 1983, SA has been presented as a search algorithm for the first time by [Kirk-1983] and [Cern-1985]. They prove that by analogy the generalization of this method to combinatorial problems is straightforward. The current state of the thermodynamic system is analogous to the current solution of the combinatorial problem, the energy equation is analogous the the objective function, and the ground state to the global minimum.

The fundamental idea is to allow moves resulting in a solution of a worse quality than the current solution (uphill moves) to be able to escape from local minima. The probability of doing such a move is decreased (temperature reduction) during the search. Immediately, this is one of the main points of difference with an SQP method. In a SQP problem, any solution that is worse than a previous one will not be accepted, within SA it will be. The major difficulty in implementation of the algorithm is that there is no obvious analogy for the temperature $T$ with respect to the free parameter. To avoid a local minimum is depending on the annealing schedule, the choice of the initial temperature, how many iterations are performed at each temperature and how much the temperature is decremented at each step as the cooling proceeds. These described variables can be denoted as the tuning-parameters of the algorithm.

First, to get the working principle of the algorithm clear, an example is given in the next section.
5.1 Example of Simulated Annealing

Visualize a geographical terrain, for example a mountain range. The goal is to find the lowest valley in this terrain. Introducing two "parameters" (North-South and East-West direction), SA approaches this problem using a bouncing ball which is able to bounce over the mountain tops from valley to valley. Starting at a high "temperature", it permits the ball to make very high bounces to bounce over any mountain top to access any valley (given enough bounces). As the temperature is getting lower (depending on a specified "cooling schedule", see Figure 5.1), the acceptance of jumping out of a valley becomes smaller.

![Figure 5.1: Typical temperature cooling schedule](image)

The mountain range is aptly described by a cost function and a probability distribution is defined on the two directional parameters called generating distributions since they generate possible valleys or states to explore. Also, a so-called acceptance distribution is defined which depends on the difference of cost functions of the present generated valley we are to explore and the last saved lowest (or optimal) valley. The acceptance distribution decides probabilistically whether to stay in a new lower valley or to bounce out of it. All the generating and acceptance distributions depend on temperatures.

In a more general formulation, the SA algorithm starts by generating an initial solution which is either randomly or heuristically chosen. Also the so-called temperature parameter T is initialized. After this the following is repeated until the termination criterium is satisfied: A solution $s'$ from the neighborhood $N(s)$ of the solution $s$ is randomly sampled and it is accepted as new current solution depending on $f(s)$, $f(s')$ and $T$. If $f(s') < f(s)$, $s$ will be replaced by $s'$, or in case $f(s') \geq f(s)$ with a certain probability, $s$ will also be replaced. This probability is a function of $T$ and $f(s') - f(s)$ and is generally computed following the Boltzmann distribution equation (5.1):

$$p = \exp^{-\frac{(f(s') - f(s))}{T}}$$

(5.1)

This equation yields that when $T$ is high, the chance of acceptance is high and so the probability of uphill moves is high and vice versa.

A general layout of the algorithm is given in Table 5.1
Initialization (initial solution, temperature)
Calculation of the current cost \( f(x) \)

While Stop \( SA=0 \)
  For \( m = 1 : n_{\text{temp}} \)
    For \( n = 1 : n_{\text{stepadj}} \)
      Generate a new random solution \( x' \) within \( N(x) \)
      Calculate the new cost \( f(x') \)
      If \( f(x') < f(x) \)
        \( s := s' \)
      Else
        If \( p = \exp^{-f(x')-f(x)/T} < \text{rand}(1, 1) \)
          accept \( \Rightarrow x := x' \)
        Else
          reject
      End
    End
  End
Reduce step-size
End
Reduce temperature
If \( f(x) < f_{\text{opt}} \)
  Stop \( SA=1 \)
End

Table 5.1: The general layout of the SA algorithm

5.1.1 Example test results

The algorithm has been implemented into Matlab. In these experiments a continuous parabolic function (geographical landscape) with several local minima and only one global minimum has been set as the cost function (see Figure 5.2). The goal, of course, is to find the global minimum; the value for the control parameter which leads to the lowest function value. The control variable is the position \( x \) in the landscape

After tuning the parameters, successful experiments have been performed. One of the results are presented in Figures 5.3(a) and 5.3(b)

From Figure 5.3(a) it is obvious that the algorithm jumps from one local minimum (valley) to another until it has found the global minimum. Figure 5.3(b) contains every function-value that has been accepted and it perfectly illustrates that not only better solutions but also worse solutions have been accepted during the process of finding the global minimum.

For these test several parabolic functions have been analyzed and all of the experiments have lead to the global minimum, even when the initial condition of the algorithm has been located in a global minimum.

This simple example gives a clear view on the algorithm. Although the algorithm is pretty robust for this example, the influence of varying the tuning parameters has
CHAPTER 5. SIMULATED ANNEALING

Figure 5.2: Parabolic cost function with its global minimum (red circle) and the initial solution of the SA algorithm (green star). Notice that the initial solution is located in a local minimum.

Figure 5.3: Results of the experiments with the SA algorithm.

been experienced.

5.2 Implementation of SA in the Optimization Problem

In the previous section, the algorithm has been searching for a value $x$ for which the objective function $f(x)$ is minimal. Analogous, for the optimization problem of this project, the vector with powers of the generatorset ($P_{genset}$) is stated as $x$ and the corresponding fuel consumption as the objective function $f(x)$. Just as in the previous chapter, the fuel consumption is not a real function but is obtained via a simulation of the simplified model in Matlab/Simulink, and so is defined by $a$, over the drive cycle calculated, number. Every time a new $P_{genset}$ has been calculated, a simulation is performed.
5.2.1 Initialization

Before the SA can start it is necessary to initialize the parameters. Besides the initial condition of $P_{gs}$ also the parameters of the algorithm have to be defined.

- initial stepsize ($step$)
- stepsize reduction ($step_{reduc}$)
- number of efforts to find a more optimal solution before stepsize reduction ($n_{stepad}$)
- initial temperature ($T$)
- temperature reduction ($T_{reduc}$)
- number of efforts before decreasing the acceptance of a worse solution ($n_{tempad}$)

These tuning-parameters will influence the performance of the algorithm as learned from the implementation of the example in the previous section. By taking very large steps, the possibility of the algorithm to find the optimum decreases. On the other hand, by taking small steps and a very slow temperature decrease, the possibilities of finding a global minimum rises but also the time of calculation will rise with it. So a trade-off has to be made here.

5.2.2 Scheduling of the SA algorithm

To assure that the overall SOC will be neutral it is first necessary to find the constant power line for $P_{genset}$ for which this is the case. This part of the optimization problem is performed by running two simulations every time a new $P_{genset}$ is calculated, just as is done with the previous methods. The end-value of the SOC of the first simulation will become the initial SOC of the second simulation. This way, the optimal constant power line is obtained for which $dSOC = 0$.

Because of the fact that looking for an optimal solution every second of the 1180 seconds long MVEG cycle has proven to be too much to ask from the algorithm, a smart strategy has to be introduced. This strategy will split the obtained $P_{genset}$-line into parts and will let SA look for a solution by only looking at a few parts while the rest is kept constant. This looks like redividing the mean power, and in a certain way it is, but the strategy allows the mean power to decrease (as is expected) a little bit so the FC will drop while maintaining the always present demand of $dSOC = 0$. For
this strategy, the drive-cycle has been extended from 1180 to 1200 seconds. This has been done by adding a part of 20 seconds with zero velocity. Of course, adding the 20 seconds will have influence on the FC and SOC, but this problem is to overcome to determine the FC and SOC at 1180 seconds. This way, the strategy can divide the drive-cycle in an even number of equal time-spans without influencing the results.

The algorithm starts to divide the $P_{\text{genset}}$ into two parts and will vary them while looking for a better fuel consumption. When it has found a better solution, both parts will be divided into 2 again. Now it will look for a better solution by changing the different parts in pairs. It will start with part 1 & 2 then 2 & 3 and 3 & 4 (see Figure 5.5). The reason that the algorithm is able to find a lower FC is ensured in the fact that for different values of $P_{\text{genset}}$, different (more or less efficient) fuel consumptions are the result.

This scheduling is repeated until a time-span of 3 seconds is reached. This means $P_{\text{genset}}$ will be divided at that moment into 400 parts.

**5.2.3 Simulations**

After implementing the strategy described above, simulations have been performed. In order to find the right parameters which lead to a solid search to the optimal solution, a lot of simulations have been performed. As mentioned before, a trade-off has to be made between step-sizes and calculation time. Slow cooling schedules should lead to a more reliable solution but will take long calculation time. Faster cooling schedules will lead to a less reliable solution in a shorter time. The parameters mentioned in Section 5.2.1 are the ones who are responsible for this. They have been varied within the following ranges.

\[
T = [1, ..., 15] \quad [-]
\]
\[
T_{\text{red}} = [0.1, ..., 0.9] \quad [%]
\]
\[
n_{\text{tempad}} = [5, ..., 20] \quad [-]
\]
\begin{align*}
\text{step} &= [200, \ldots, \text{mean}(P_{\text{opt}})][W] \\
\text{step}_{\text{red}} &= [0.3, \ldots, 0.9]\% \\
\text{n}_{\text{stepad}} &= [5, \ldots, 20][\text{-}]
\end{align*}

It may be obvious that these parameters have a correlation with each other. For example, a larger \(T\) and small \(T_{\text{red}}\) leads to a larger \(n_{\text{tempad}}\) to be sure the temperature will be very small (nearly zero) at the end of the algorithm.

The results of the first simulations are quite different from each other. Examples of the results are shown in Figure 5.6. The fuel consumption for these simulations are 395, 392, 389, 393 grams, respectively, for the entire drive cycle.

![Figure 5.6: 4 different results of the SA algorithm, all leading to different FC](image)

The results of the simulations shown above are all produced using the same parameters. It is obvious that one could not speak of a global solution. Also the corresponding fuel consumptions are not satisfying. They are approximately the same as the fuel consumption that is obtained with the control structure from Chapter 3. This could mean that the fuel consumption that is found with the control structure is very close to a global solution, but this is not reasonable to assume. But why would the algorithm not find a global solution or better; why does it find another FC and combination of \(P_{\text{genset}}\) every time a search has been performed?

The answer to the question why the algorithm is not capable of finding the same minimum (global or not) over and over again could be assigned to the fact that it is a heuristic method which uses a random factor. This means that every time the step-size the algorithm takes is different (randomly chosen), which could lead to different combinations of \(P_{\text{genset}}\) while the FC is approximately the same.

A disadvantage of the schedule that has been introduced is that the power, once divided, stays in that area. For example, the determined high powers in the end of the cycle are never able to be relocated to the beginning of the cycle. This way it is never possible to explore every possible combination. For this reason the schedule is
rewritten in such a way that "the pairs" are not in successive order anymore, but are formed by the time-spans from the outside to the inside of the cycle. This way it is possible to introduce a surge into $P_{gs}$ and it is possible to relocate powers. For these experiments, a relative slow cooling schedule and also small step reductions have been used.

Although the algorithm is able to relocate powers now, the results were even more disappointing than the previous one. Figure 5.7 shows one of the $P_{genset}$ for these simulations. The overall fuel consumption is 415 grams, which is about 6.5% higher than the FC from the control structure and without the new scheduling.

![Figure 5.7: Results of the simulation after "rescheduling"](image)

### 5.3 Summary

During the research on the simulated annealing algorithm whether it is capable of finding a global solution for $P_{genset}$ leading to a minimum fuel consumption, a lot of simulations have been performed where the different tuning parameters have been varied. All the results of the simulations are the same as or worse than the solution that has been obtained with the control structure. Two main disadvantages of this method have been experienced. First of all, SA is a heuristic method which uses a random factor to search for a solution. Though, the chance of finding a global minimum is, thanks to the method, bigger than with the SQP method, it is not guaranteed that it will find one. Simple and defined functions are not a problem for SA, as seen in the example. The real optimization problem of this project may be too much for the algorithm, because the vector to optimize is too large. Although the fact that there is more than one way to come to an optimal solution, one should expect that the algorithm should be capable of finding this solution for which the FC is lower than the benchmark so far. The reason for this is not known but the second disadvantage surely has something to do with this. The second disadvantage is the fact that tuning the algorithm is very difficult. As mentioned before, a trade-off has
to be made, but even though when a relative bad trade-off has been chosen (long simulation time, high accuracy), the solution is still not satisfying.

The method of Simulated Annealing has been implemented into the optimization problem and is tested thoroughly. The overall performance is not satisfying. The problem stated in this project, could be too large for the method. Too large in senses of, using this method to find a solution for $P_{\text{genset}}$ for which the FC is minimal, many ways to come to this solutions are possible since the number of optimization parameters is rather large. The random factor that is present in the algorithm will not ensure that every time the same solution is found.

A smart optimization algorithm to solve this optimization problem is hard to find. They all have restrictions or other factors (i.e. randomness) which avoid the algorithm to work properly for such a large problem. With Simulated Annealing, the idea has come up to ban the random factor and the temperature schedule so the obtained algorithm literally examines every solution that is possible. Such an algorithm starts to look like a Dynamic Programming problem, a powerful method which analyses all possible solutions within a given problem. Therefore, in the next chapter, all the smart optimization methods will be thrown overboard and Dynamic Programming will be introduced.
Chapter 6

Dynamic Programming

Dynamic Programming (DP) is known as a very powerful method to obtain an optimal solution to a problem. The method has first been developed by R. Bellman in the late 50’s [Bell-1957]. As the name indicates, DP is a dynamic optimization method. This means that instead of giving a constant value as an optimum, the method will calculate an optimal trajectory. By dividing the defined control variable into a grid (numerical quantization) and calculate every possible next state (that can be taken from the present state) with the corresponding cost, the method examines every possible way to obtain the optimal solution to the stated problem. The method is based on Bellman’s Principle of Optimality, see section 6.1. Despite the power of the method and the guarantee of an optimal solution, dynamic programming is limited by the so-called curse of dimensionality; the more state and control variables or the larger the grid-size, the higher the memory and time requirements of the approach. Therefore, the method is not useful for online implementation but from the solution, a rule-based control law can be derived that is. In this chapter, the basics of dynamic programming, the implementation and the results will be discussed.

6.1 Bellman’s Principle of Optimality

There are several definitions of the principle of optimality. Naturally, they all come to the same thing.

Definition 6.1 In an optimization problem, components of a globally optimal solution are themselves globally optimal

Definition 6.2 From any point on an optimal trajectory, the remaining trajectory is optimal for the corresponding problem initiated at that point.

Assume that the optimization problem is to find a set of control variables \( u(0), ..., u(p-1) \) that brings a system, \( x(k+1) = f(x(k), u(k)) \), from state \( x(0) = x_0 \) to \( x(p) = x_p \) and at the same time minimizes a cost criterium:

\[
J = \sum_{k=0}^{p-1} L(x(k), u(k)).
\] (6.1)
CHAPTER 6. DYNAMIC PROGRAMMING

$L$ describes the cost going from present state \((x(k))\) to the next state \((x(k + 1))\) by applying \(u(k)\). The optimization principle of Bellman says that if the total state trajectory from \(x(0)\) to \(x(p)\) is optimal, every part of the trajectory has to be optimal. This principle is valid for continuous as well as discrete systems. As an example, Figure 6.1 reproduces the principle of optimality for a discrete system. The optimal trajectory from \(x(0)\) to \(x(p)\) has been divided into two trajectories, one from state \(x(0)\) to an arbitrary state \(x(k)\) and another from \(x(k)\) to \(x(p)\). In Figure 6.1 these are denoted by \(x_1\) and \(x_{2a}\), respectively. The fact that the total trajectory is optimal implicates that the \(x_1\) as well as \(x_{2a}\) have to be optimal too. If not, another way \(x_{2b}\) would exist, which together with \(x_1\) would then be an optimal trajectory. Since \(x_1\) and \(x_{2a}\) have already been stated as the optimal trajectory this can not be the case.

![Figure 6.1: Bellman’s Principle of optimality](image)

6.1.1 Basic idea of DP

The basic idea of DP can be well illustrated with the simple ”shortest path” problem.

![Figure 6.2: A shortest path problem](image)

In Figure 6.2 several nodes with their transition cost are shown. The goal is to travel
from node $A$ to node $H$ over a path with minimum costs. This simple example could easily be solved by hand but it illustrates the working principle of DP. Assume that indices $i$ and $j$ represent node numbers. With $F(i)$ the cumulative cost to reach the previous node and $c_{ij}$ the cost of transition from node $i$ to node $j$, the general equation to solve this problem can be defined as:

$$F(j) = \min\{F(i) + c_{ij}\}$$

Starting at node $A$ (with initial condition $F(0) = 0$), it is possible to go to node $B$ or $C$. The costs of these transition are:

$$F(B) = \min\{0 + 7\} = 7$$
$$F(C) = \min\{0 + 6\} = 6$$

If $F(j)$ are stored in a matrix and $c_{ij}$ are already known the sequence continues with node $D$ which can be reached from node $B$ and node $C$. So now it is necessary to consider both these nodes:

$$F(D) = \min\left\{\frac{F(B) + c_{BD}}{F(C) + c_{CD}}\right\} = \min\left\{\frac{7 + 2}{6 + 4}\right\} = 9$$
$$F(E) = \min\left\{\frac{F(B) + c_{BE}}{F(D) + c_{DE}}\right\} = \min\left\{\frac{7 + 1}{9 + 3}\right\} = 8$$
$$F(F) = \min\left\{\frac{F(C) + c_{CE}}{F(D) + c_{DE}}\right\} = \min\left\{\frac{6 + 7}{9 + 2}\right\} = 11$$

and so on.

This way the sequence has taken care of the principle of optimality and will find the path with the minimum cost (A-B-D-F-H).

One can see that while calculating the total cost, the method works with a cumulative cost every possible step there is to take. So during the calculation in forward direction the algorithm uses cumulative costs and will save these costs and corresponding transitions in matrices. After this forward calculation, a backward search will recall the path, leading to the minimum cost, from the matrices that have been created. This backward search has to be performed, although during the forward calculation the cumulative costs are known, because all possible transitions and paths are calculated and stored during this forward sequence. So the algorithm has already calculated the optimal path and stored it, but does not remember it anymore. Therefore, the backward search is used to recall it.

### 6.2 Design of the DP algorithm

The goal of DP is, analogue to the other methods, to find the optimal power curve for the generator set which leads to the minimum fuel consumption while maintaining the SOC of the accumulator over a specified drive-cycle. For the design of the dynamic programming algorithm for this problem several variables have to be defined, assumptions have to be made and some constraints have to be implemented. In this section, this process will be discussed step by step leading to the total DP sequence that is able to solve the problem.
6.2.1 Control variable

First of all, the control variable \( u \) has to be defined. The control variable is the parameter which will be divided into a grid (quantization) and will be varied during the search for an optimal solution. Here, the SOC [%] of the electric accumulator has been chosen as the control variable. Since it is possible to define the initial state and the end state of the control variable, it is now possible to comply with the demand of maintaining the SOC over the given drive-cycle.

\[
dSOC = SOC_{end} - SOC_{begin} = 0 \quad (6.2)
\]

\( P_{drive} \), the electric power that is necessary to propel the vehicle, is calculated beforehand just like is done previously. By varying the SOC-level \( u \) of the accumulator (and so the power that will be delivered by the accumulator), it is possible to calculate the power that is needed from the generatorset keeping in mind that the energy balance, and so the power balance, should be correct and the SOC will be maintained. The power balance is given by:

\[
P_{drive} = P_{genset} + P_{accumulator}. \quad (6.3)
\]

Since a negative \( P_{accumulator} \) is defined as discharging or giving power, the power balance that have been used becomes:

\[
P_{genset} = P_{drive} + P_{accumulator}. \quad (6.4)
\]

6.2.2 Battery model

The DP algorithm will work with discrete steps (quantized grid), therefor, the available dynamic battery model with SOC dependency and time-varying resistances will have to be replaced by a discrete battery model which approaches this dynamic model. This implies a small error but it will be so small, this should not have any effect on the results except the fact that theoretically the battery can now deliver its powers instantaneously. At the end of this chapter, the two different models will be compared to see what the influence of the replacement is. The battery model is visualized in Figure 6.3.

![Battery Model](image.png)
with:

\[ P_0 = u_0 \cdot I \]  \hspace{1cm} (6.5)
\[ u_0 = u_{oc} + I \cdot R_{int}. \]  \hspace{1cm} (6.6)

Substituting equation 6.6 into 6.5 yields:

\[ P_0 = (u_{oc} + I \cdot R_{int}) \cdot I = u_{oc} \cdot I + I^2 \cdot R_{int}. \]  \hspace{1cm} (6.7)

In the dynamic battery model, the internal resistance, \( R_{int} \), is depending on two build-in 1\textsuperscript{st} order RC - circuits, and the open circuit voltage, \( u_{oc} \), is depending on the SOC, as shown in equation (6.8):

\[ u_{oc} = (u_{min} + u_{slope} \cdot SOC) \cdot n_{cells} \]  \hspace{1cm} (6.8)
\[ u_{oc} = (1.2897 + 0.1 \cdot SOC) \cdot n_{cells} \]  \hspace{1cm} (6.9)

The influence of the RC-circuits is illustrated in Figure 6.4. Considering Ohm’s law, one can see the results of the time-dependance of the RC-circuits (black solid line).

By assuming that \( R_{int} \) and \( u_{oc} \) are constant, a discrete approximation of the dynamic battery model is obtained. The internal resistance is calculated, using the constant resistance (\( R \)) and the resistance of the RC - circuit with shortest time constant (\( R_1 \)), according to equation 6.10.

\[ R_{int} = (R + 0.5 \cdot R_1) \cdot n_{cells} = 0.448 \, \Omega \]  \hspace{1cm} (6.10)

For \( u_{oc} \) the mean voltage is taken,

\[ u_{oc} = \frac{u_{oc}SOC_{min} + u_{oc}SOC_{max}}{2} = 345 \, V. \]  \hspace{1cm} (6.11)

In Figure 6.4 the result of using Ohm’s law on the discrete model is also shown (red dotted line).

### 6.2.3 Boundaries on the control variable

During the search for an optimal power curve of the generator set, the minimum fuel consumption will be obtained if the accumulator can be depleted (natural behavior). This is certainly not desirable and so to be sure this can not occur, boundaries on the SOC will be introduced. One can distinguish 2 types of boundaries, each divided in an upper and lower bound.

- **Upper Boundary**: maximum SOC of the accumulator
- **Lower Boundary**: minimum SOC of the accumulator
- **Maximum Charge Power**: maximum power the accumulator can absorb
- **Maximum Discharge Power**: maximum power the accumulator can deliver

The upper and lower boundaries will form the SOC-window the accumulator works in. A common used SOC-window is ±10\% SOC difference from the desired value.

Concerning the maximum (dis)charge power, these parameters are different for every accumulator. For example, the used accumulator (battery) can give and take 50 kW
of power. This means that every second the accumulator is able to deliver or absorb a maximum of 50 kJ. For the boundaries this can be rewritten into percentages of SOC difference the accumulator can maximum handle. This is done by solving equations 6.7 for charge and discharge parameters.

\[ 0.448 \times I^2 + 345 \times I = 50000 \quad \rightarrow \quad I_{ch} = 124 \ A \quad (6.12) \]

\[ 0.448 \times I^2 + 345 \times I = -50000 \quad \rightarrow \quad I_{dsch} = -193 \ A \quad (6.13) \]

The obtained values correspond to respectively 0.34% and −0.53% maximum SOC-level difference that is allowed per second. These values will form the maximum charge and discharge SOC and will be used later on while calculating state-cost-matrix \( U \) and while finding the optimal path.

So now the boundaries between which the algorithm can search for the most efficient path of the battery power, and thus, the most efficient path of the generator set, can be presented as in Figure 6.5

### 6.2.4 State-Cost-Matrix "U"

To reduce the amount of calculations in the DP sequence itself later on, a "state-cost-matrix" \( U \) is introduced. This matrix contains all possible states of the control variable (within its maximum (dis)charge range) with the corresponding costs for every time-step. The advantage of calculating matrix \( U \) before the searching for the optimal path is that it saves time.

\( U \) is calculated on the basis of the power balance (equation 6.4):

\[ P_{drive} + P_{accumulator} = P_{genset} \]
So $P_{drive}$ is known at every time-step and $P_{accumulator}$ can be calculated from the control variable, $P_{genset}$ can be determined and so, the fuel consumption that comes with these transitions.

This means that $U$ will look like:

$$U = \begin{bmatrix} f_{c11} & f_{c12} & \ldots \\ f_{c21} & f_{c22} & \ldots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

The columns in this matrix represent time [s] and the rows represent the different SOC-levels. This difference in SOC is divided in steps that have been defined by the grid and will be between the maximum charge and discharge. For example, a step-size of 0.005\% in SOC-level will result in $(0.34\% - (-0.53\%))/0.005\% = 175$ possible SOC-level steps.

For every step in the grid of the SOC, a corresponding battery power can be calculated with equation 6.7, a power that can be delivered during that second. By substituting this vector of battery powers and the known $P_{drive}$ at that moment into the energy balance (equation 6.4), a vector with all possible powers for the generator set will be obtained. To avoid that the genset power becomes negative, an inequality constraint (6.14) will be introduced so only positive powers of the genset will be allowed.

$$P_{accumulator} \geq -P_{drive} \quad \text{(6.14)}$$

The "-" sign in front of $P_{drive}$ is due to the way the current ($I$) is defined, a negative $P_{accumulator}$ means that the accumulator delivers power. This way, only the positive powers of the genset will be considered.

The genset has a certain maximum power it can deliver. Every calculated value exceeding this power is useless to take into account so another constraint is put on this maximum power.
CHAPTER 6. DYNAMIC PROGRAMMING

\[ P_{\text{genset}} \leq P_{\text{genset max}} \]  
\( (6.15) \)

Once the vector with all possible positive generator set powers has been calculated for that time step, it can be converted in terms of fuel consumption. For efficiency reasons, it is desirable for the generator set to work only in the most efficient areas. Therefore, the specific fuel consumption is taken (interpolation) from the optimal operating line (OOL) of the generator set, see Figure 6.6.

\[ \text{Optimal Operating Line for } P_{\text{genset}} \text{ vs. Fuel Consumption} \]

Figure 6.6: Optimal Operating Line for \( P_{\text{genset}} \) vs. Fuel Consumption

The horizontal line is manually defined since no data is available for \( P_{\text{genset}} < 4.5 \text{ kW} \). Since it is inefficient to work in this area and, moreover, the generator set is not able to generate such low powers, the fuel consumption is set to the value of the fuel consumption of the first data-point (except for zero of course). Hence, the calculation of \( U \) is completed and contains the fuel consumption values at a specific time-step for different possible steps in SOC-level.

Example

To illustrate what this matrix \( U \) yields, a small example is given below.

Assume that at time-step \( k \) the power demand is 5 kW. If the \( (i) \text{th} \) step in SOC-level corresponds to 5 kW, then it is possible to satisfy the power demand completely by the battery, so \( P_{\text{genset}} = 0 \) and the \( FC = 0 \). If the \( (i - 1) \text{th} \) step corresponds to 4 kW, the genset has to deliver another 5-4=1 kW which, in turn, corresponds to a certain \( FC \). If the \( (i + 1) \text{th} \) step corresponds to 6 kW, this means that according to the power balance, the genset has to generate a negative power of -1 kW. So this step is not considered. All the obtained \( FCs \) will be stored in the right place (corresponding to time and step) of the matrix \( U \).
6.2.5 Optimal Path

Once initialized and calculated the state-cost-matrix $U$, the optimal path leading to the minimal fuel consumption can be determined. In this part, the actual method, on which DP is based, will be called. Starting at the given initial condition the algorithm will search (in forward direction) for every time-step and every point in $U$, to the most optimal path until it reaches the given end-state. It will do this by analyzing every possible path there is to take and compare every cumulative value for the fuel consumption for these paths. Hence, the principle of optimality for making the optimal decision at the $k^{th}$ step can be expressed as:

$$J_k(x(k)) = \min_{u(k)}[f(x(k), u(k)) + J_{k+1}(x(k+1))] \quad 0 \leq k < n - 1$$

(6.16)

Each possible path, together with the costs, is saved in matrices. Because the forward method calculates every possible path, and does not remember the optimal one as such, a backward search is needed to recall the path with the minimum fuel consumption while maintaining the SOC of the accumulator. Since every possible path is considered, it takes a while before it is finished, but it is for sure that the solution is the optimal one.

Since the SOC-level has been set as the control variable, the obtained optimal path is this SOC-level. From this solution it is possible to calculate the powers that are needed from the accumulator, and so also the needed $P_{\text{genset}}$.

6.3 Results for MVEG cycle

The algorithm with its constraints are implemented in Matlab. After extensively testing the algorithm, the first simulation results are obtained. For these simulations, the MVEG-cycle has been used. Later on in this chapter the results of several other drive-cycles will be discussed.

First of all, the fuel consumption that corresponds to this calculated optimal $P_{\text{genset}}$ is 381.6 grams for the entire drive-cycle. This is a reduction of 1.65 % regarding the
current benchmark of 387.9 grams. Second, the total energy balance has to be
correct. This means that the sum of the total energy generated by the generator set
and the total energy of the accumulator \( E_{\text{batt charge}} - E_{\text{batt discharge}} \) must be equal
to the total desired energy \( E_{\text{drive pos}} + E_{\text{drive neg}} \). Since this energy balance is used
in the algorithm, it is expected that it is correct. The energy balance gives useful
information about the efficiencies of the accumulator and how the powers have been
divided.

Though a reduction in the fuel consumption is obtained, there is still an error present
in the algorithm. This error can be discovered when Figure 6.8 is magnified, as shown
in Figure 6.9.
In Figure 6.9 one can see that when the velocity on the drive-cycle is equal to 0 (Figure 6.7), there is still a certain power demand. This power demand is called the accessory power, and is the result of powers that are always needed although the velocity is zero; for example, the power of the processors, cooling fans and lights. This accessory power, as can be seen in the figure, is now delivered by the generator set \( P_{\text{accumulator}} = 0 \), leading to very low powers the generator set has to deliver. Naturally, this is certainly not desirable since the efficiency of the generator set in this working area is much worse than in higher working areas. Moreover, the generator set is not even capable of generating such powers in a stable way.

The reason for this error to occur can be explained by the fact that the chosen grid is too large. At this moment, one grid step in the SOC-level of 0.005 % results in a power step of approximately 620 Watt (charge and discharge). The mean accessory power is approximately 260 Watt, resulting in the fact that the inequality constraint (6.14) can not be satisfied. So, or the generated power should be negative, or the accessory power should completely be delivered from the generator set. In practice, the second option shall be chosen since negative values for the generator set are not allowed. The problem of taking a grid that is too large is the fact that gaps exist between the demanded power and the possible accumulator power (defined by the grid). Note that this problem does not only account for zero-velocity, at every point there will be a gap between the two powers \( P_{\text{drive}} \) and \( P_{\text{accumulator}} \) causing the generator set to work in a very inefficient working area (see the irregular \( P_{\text{genset}} \) line in the figure).

To solve this problem, two solutions have been considered.

1. Reduce the step-size in the grid, enlarging the grid-size,
2. Adapt \( P_{\text{drive}} \).

By reducing the step-size in the grid, it is possible to reduce the gap between \( P_{\text{drive}} \) and the possible \( P_{\text{accumulator}} \). For example, if the step-size should be reduced 10 times (enlarging the grid-size 10 times), every step in SOC-level will correspond to approximately 62 Watt. Now a gap of maximal 31 Watt would occur, still causing the generator set to work in an inefficient area. Another possibility is to reduce the step-size so steps of 1 Watt can be taken. This will cause an enormous grid-size, but there will be no gap anymore. There are strict limitations on enlarging the grid-size. Not only the time it would take to calculate a solution will rise, also the matrices become of such a size, the program is not able to handle them anymore.

The other solution is to adapt the obtained \( P_{\text{drive}} \) to the values that correspond to the possible values of \( P_{\text{accumulator}} \). This is possible since the fixed step-size in the grid results in a fixed \( P_{\text{accumulator}} \) vector since \( P_{\text{accumulator}} \) has been made independent of the SOC. The gap between the two powers will vanish. During adapting, the values that yield the smallest error between the two powers will be taken.

After applying this trick, the accessory power will be delivered completely by the accumulator while \( P_{\text{genset}} = 0 \). The results of the simulation are presented in Figure 6.10.
The fuel consumption for this simulation is 377.8 gram for the entire drive-cycle. This is a reduction, as expected, of 1 % regarding the previous result (381.6 gram). A small correction must be made over here since the total $E_{drive}$ has changed a little bit caused by the adaption of $P_{drive}$ to $P_{accumulator}$. The difference between the two is small, but to be able to compare the two different results, it is necessary to normalize the fuel consumption. Therefore, the fuel consumption will not only be given in grams over the entire drive-cycle and liter per 100 km but also in grams per Joule. The reduction regarding the current benchmark is 2.82 %. This reduction is a pretty good result and shows that the DP algorithm is able to find a lower FC than the current control structure (CS) while the constraints have been satisfied and the results are realistic.

<table>
<thead>
<tr>
<th></th>
<th>FC [gr]</th>
<th>FC [l/100km]</th>
<th>FC [gr/kJ]</th>
<th>Reduction [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS parameters</td>
<td>391</td>
<td>4.32</td>
<td>0.0699</td>
<td>-</td>
</tr>
<tr>
<td>Opt. CS parameters</td>
<td>387.7</td>
<td>4.28</td>
<td>0.0693</td>
<td>0.86 %</td>
</tr>
<tr>
<td>Dynamic Programming</td>
<td>377.8</td>
<td>$\approx$ 4.17</td>
<td>0.0674</td>
<td>2.82 %</td>
</tr>
</tbody>
</table>

Table 6.1: Fuel reduction for MVEG-cycle

From Figure 6.10 one can see that the behavior of the generator set can be compared to the generator set behavior with the present control strategy (see Figure 3.7 and 3.8). It is a kind of load following character where energy is mostly delivered from the accumulator except when the power demand becomes higher than a certain value ($\pm 8$ kW), see Figure 6.12. From this figure it is also clear to see that the regenerated braking energy is completely absorbed by the accumulator. The maximum charge and discharge power of the accumulator (50 kW) are never reached which is good for the life-span of the accumulator. The fact that the accumulator will be recharged at the moments that $P_{drive}$ asks a certain amount of energy or more, can be explained by the fact that, apparently, it is rather "cheap" to let the generator generate more
energy at the moment it is already running in stead of generating energy later on so the genset has to switch on again. In Figure 6.12, roughly 4 operating points of the generator set can be distinguished. Roughly, because the region between 18 and 20 kW is almost fully utilized. Later on in this chapter (section 6.5) this phenomena will be a point of discussion. Another thing that can be derived from Figure 6.12 is a simple control law. It is easy to see that a certain desired $P_{drive}$ will result in a corresponding generator set power ($P_{genset}$) or battery power ($P_{accumulator}$). For example, for a $P_{drive}$ between 21 ∼ 45 kW, it is optimal to operate the generator set at 26 kW. This is how for the considered drive-cycle a simple control structure can be derived from the results of dynamic programming. If the drive-cycle is unknown, extra constraints and rules have to be brought in to be able to maintain the SOC.
Looking at the behavior of the SOC in Figure 6.10, the battery is slowly depleting during the city-parts of the cycle (0 - 800 seconds). For the freeway-part, the battery will be charged at first before a great amount of energy is asked. Regenerative braking energy is nicely stored in the battery at the end of the freeway-part. The SOC-window that is allowed is ±10% but it only uses ±5%.

To examine the error that is introduced by using a discrete battery model in stead of the dynamic battery model, the obtained $P_{\text{genset}}$ is used as an input for the dynamic battery model. The result is presented in Figure 6.13. Here the maximum difference between the two trajectories is 0.35% and the error in SOC at the end of the cycle is only 0.02 %, an error which can be neglected. So it can be stated that the use of the discrete model, in stead of the dynamic model, does not have any large influence on the results and that the assumptions that have been made are justified.

![Figure 6.13: Difference between the discrete and dynamic battery model](image-url)
6.4 Different drive-cycles

So far, only the MVEG-cycle has been considered. Besides this drive-cycle it is also interesting to see what reduction DP can bring to other cycles. Therefor, the USFTP72 and the Modem cycle are also investigated.

6.4.1 USFTP72-cycle

The United States Federal Test Procedure (USFTP) is the official cycle for vehicle legislation in the United States. Two different USFTP’s can be distinguished. The first one, the USFTP72 is also known as the Urban Dynamometer Drive Schedule (UDDS). It consists of two parts, a cold transient part and a cold stabilized part. The second one, the USFTP75 is based on the UDDS only now, after the UDDS has finished a period of approximately 10 minutes follows in which the engine is heated before the cold transient part of the UDDS is repeated. Since a distinction between cold and hot engines is not taken into account in this project, the USFTP72 will be considered.

The velocity profile of the USFTP72 is shown in Figure 6.14.

The benchmark for this cycle has been calculated and is shown in Table 6.2.

<table>
<thead>
<tr>
<th>CS parameters</th>
<th>FC [gr]</th>
<th>FC [l/100km]</th>
<th>FC [gr/kJ]</th>
<th>Reduction [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>433.2</td>
<td>4.36</td>
<td>0.0718</td>
<td>-</td>
</tr>
<tr>
<td>Opt. CS parameters</td>
<td>432.7</td>
<td>4.35</td>
<td>0.0717</td>
<td>0.1 %</td>
</tr>
</tbody>
</table>

Table 6.2: Initial benchmarks for USFTP72-cycle

As can be seen, the difference between the structure with Advance parameters and the optimized structure is minimal.

The results that have been obtained with dynamic programming for the USFTP72 cycle are shown below. For the three different power-lines ($P_{\text{drive}}, P_{\text{batt}}$ and $P_{\text{genset}}$), the reader is referred to Appendix B.

Looking at Figure 6.16, 4 operating points can be distinguished again. These points are the same points as obtained with the MVEG cycle. The moment at which the generator set will be turned on has moved from 8 kW to 10 kW, meaning that for a longer time energy is solely delivered from the accumulator. Again a complete control
law for this cycle can be derived which will be approximately the same as the one for the MVEG.

The fuel consumption that is obtained with dynamic programming is 408 grams over the entire drive-cycle, which is a reduction of approximately 6 % regarding the benchmark obtained with the optimized structure.

<table>
<thead>
<tr>
<th></th>
<th>FC [gr]</th>
<th>FC [l/100km]</th>
<th>FC [gr/kJ]</th>
<th>Reduction [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS parameters</td>
<td>433.2</td>
<td>4.36</td>
<td>0.0718</td>
<td>-</td>
</tr>
<tr>
<td>Opt. CS parameters</td>
<td>432.7</td>
<td>4.35</td>
<td>0.0717</td>
<td>0.1 %</td>
</tr>
<tr>
<td>Dyn. Program.</td>
<td>408</td>
<td>~4.10</td>
<td>0.0676</td>
<td>6.06 %</td>
</tr>
</tbody>
</table>

Table 6.3: Fuel reduction for USFTP72-cycle
6.4.2 Modem cycle

The Modem cycle can be divided into 4 parts: slow urban, road, free flow urban and a highway part. This drive cycle represent actual driving conditions of a passenger vehicle. The maximum accelerations are higher than that from the MVEG or the USFTP72.

![Velocity profile of the Modem cycle](image.png)

Figure 6.17: Velocity profile of the modem drive cycle

The benchmark for this cycle has been calculated again (Table 6.4).

<table>
<thead>
<tr>
<th></th>
<th>FC [gr]</th>
<th>FC [l/100km]</th>
<th>FC [gr/kJ]</th>
<th>Reduction [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS parameters</td>
<td>1135.3</td>
<td>5.46</td>
<td>0.0709</td>
<td>-</td>
</tr>
<tr>
<td>Opt. CS parameters</td>
<td>1128</td>
<td>5.42</td>
<td>0.0704</td>
<td>0.7 %</td>
</tr>
</tbody>
</table>

Table 6.4: Initial benchmarks for Modem-cycle

![SOC profile](image.png)

Figure 6.18: SOC for the Modem cycle with battery

Analyzing the results of the Modem-cycle, it is clear from Figure 6.19 that for this rougher drive cycle the same operating points of the generator set are obtained and that the behavior of the generator set is the same as for the other drive-cycles. A remarkable phenomena is the fact that on the interval from 18 to 21 kW, there are two possibilities to choose from. This overlap of powers is also present in other cycles but never on this interval. Apparently, there are certain drive powers that are demanded in different situations which causes the generator set to behave different.

The fuel reduction that is achieved with dynamic programming is 4.45 %.
Though the energy balance for all drive cycles must be correct, for completeness, it is listed in the table below.

<table>
<thead>
<tr>
<th></th>
<th>MVEG</th>
<th>USFTP72</th>
<th>Modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{\text{drive}}$</td>
<td>5.60 MJ</td>
<td>6.03 MJ</td>
<td>16.05 MJ</td>
</tr>
<tr>
<td>regen</td>
<td>-1.21 MJ</td>
<td>-1.81 MJ</td>
<td>-3.49 MJ</td>
</tr>
<tr>
<td>demand</td>
<td>6.81 MJ</td>
<td>7.84 MJ</td>
<td>19.54 MJ</td>
</tr>
<tr>
<td>$E_{\text{genset}}$</td>
<td>5.74 MJ</td>
<td>6.19 MJ</td>
<td>16.44 MJ</td>
</tr>
<tr>
<td>$E_{\text{accumulator}}$</td>
<td>0.14 MJ</td>
<td>0.16 MJ</td>
<td>0.39 MJ</td>
</tr>
<tr>
<td>charge</td>
<td>2.35 MJ</td>
<td>2.81 MJ</td>
<td>4.84 MJ</td>
</tr>
<tr>
<td>discharge</td>
<td>2.23 MJ</td>
<td>2.66 MJ</td>
<td>4.45 MJ</td>
</tr>
</tbody>
</table>

Eff. of battery ($\eta_{\text{battery}}$) | 94.6 % | 94.4 % | 91.2 %

Table 6.6: Energy balance DP optimization with battery (3 different cycles)
6.5 Operating points

As already seen from the results for the different drive cycles, 4 operating points have come up. Analyzing these 4 points, it is not very strange only these 4 appeared. As a matter of course, 0 kW will not be considered. Concerning the other three, looking at the brake specific fuel consumption (bsfc) of these loads, they lie extremely close to each other. Notice that only the optimal operating line (O.O.L) has been used to determine the points, meaning that the obtained points are the optimal points on the O.O.L.

<table>
<thead>
<tr>
<th>OP</th>
<th>bsfc [gr/kWh]</th>
<th>Eff. in genset eff. map [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 kW</td>
<td>237.56</td>
<td>34.9</td>
</tr>
<tr>
<td>20 kW</td>
<td>237.24</td>
<td>35.4</td>
</tr>
<tr>
<td>26 kW</td>
<td>236.88</td>
<td>35.0</td>
</tr>
</tbody>
</table>

Table 6.7: Operating Points

In Figure 6.20, the operating points vs. bsfc are presented. It is easy to see that the obtained operating points correspond to the optimal brake specific fuel consumption.
6.6 Summary

A dynamic programming algorithm has been developed which is able to find an optimal $P_{\text{genset}}$ trajectory which leads to a minimum fuel consumption. The overall demand of $dSOC = 0$ has been satisfied. The approximations that have been made regarding the battery-model have proved to be tolerable.

Three drive-cycles have been examined which are, looking at the characteristics of the cycles, different from each other. The results show approximately the same behavior for the generator set except for the moments of changing operating points. Approximately 4 operating points have been obtained which are all the same for the different cycles. Section 6.5 has shown why these operating points have been chosen by DP. A significant fuel reduction over the given drive-cycles has been obtained with DP which will be used as the new benchmark.

Dynamic Programming is not suitable for online implementation because of the amount of calculations it has to perform to come to an optimal solution. But the numerical results of DP will be very useful to extract parameters to create an optimal control law. For the analyzed drive-cycles, a (near-) optimal control structure can be derived almost directly from figures like 6.12, 6.16 and 6.19, as has been explained at the end of Section 6.3. For unknown drive-cycles, the obtained results can be also be used to create a near-optimal control law for certain situations complied with constraints to maintain the SOC.

Since DP proved to be a powerful tool to calculate an optimal trajectory, the demand for an universal tool is inevitable. One can think about implementing a super-capacitor in the system in stead of using a battery. This will lead to whole different characteristics of the system behavior. The goal of the next chapter will be to generalize the DP algorithm so the behavior of the generator set using a super-capacitor can be investigated.
Chapter 7

Introducing a super-capacitor

A lot of knowledge, in the area of battery behavior in a series hybrid electric vehicle, is already obtained. To widen the knowledge in the HEV area, it has been decided to switch from a battery to a super-capacitor (supercap) as an accumulator.

A super-capacitor resembles a regular capacitor with the exception that it offers very high capacitance in a small package. Energy is stored by means of static charge rather than of an electro-chemical process that is inherent to the battery. A current flow will charge or discharge the super-capacitor according to:

\[ Q = Q_0 + \int i \, dt \]  

(7.1)

The voltage of the super-capacitor is a direct result of the charge together with the capacity of the capacitor.

\[ V = \frac{1}{C} \cdot Q \]  

(7.2)

In contrast to a normal capacitor, the storage principle of a supercap crosses the battery technology by using special electrodes and some electrolyte. Here, the surface area of the electrodes is an important measure for the amount of electrical charge the supercap can store. The larger the surface area, the more it can store.

Due to the fact that the process of storing energy is not electro-chemical but static, the super-capacitor is able to charge and discharge in a very short time. Together with the low internal resistance (causing a higher efficiency) and the fact that there is no danger for ”memory-effects”, super-capacitors are a very efficient type of accumulators.

The main advantages of the supercap can be stated as:

- Rapid (dis)charge
- Efficient
- No degradation or memory-effect

A limitation of a supercap is the kind of electrolyte. An aqueous variety offers low internal resistance but limits the voltage to 1 Volt. An organic electrolyte allows 2.5 Volt, but the internal resistance is higher. TNO has chosen for the organic variety. Each supercap has a nominal voltage of 2.5 Volt and a capacity of 3500 F. To operate at a higher voltage, 156 supercaps have been connected in series so a maximum
nominal voltage of 390 Volts is reached. The entire pack has an internal resistance of $57.53 \times 10^{-3} \Omega$.

In line with the optimization that has already been done with the battery, an optimization with the super-capacitor has to be performed. Therefore, the battery model has to be replaced by a super-capacitor model and the algorithm has to be adjusted. Before doing so, a so-called super-battery will be analyzed as a transition state between battery and super-capacitor. This chapter will first discuss this step. After that, the necessary changes and results for a super-capacitor will be the point of discussion.

7.1 "Super-battery"

To avoid changing two properties at once, a step between the use of a battery and a super-capacitor as an accumulator will be taken. In this step, the optimization is carried out using a so-called "super-battery". This battery will have the same characteristics as the already used battery only now the internal resistance ($R_{int}$) has been changed in order to see what the behavior of the SOC and so the generator set power will be. A lower resistance yields a smaller energy loss and so a higher efficiency of the accumulator. The new resistance is taken the same as the resistance the super-capacitors will have: $57.53 \times 10^{-3} \Omega$. This is considerably lower than the resistance of the normal battery which is $448 \times 10^{-3} \Omega$. Energy can now be stored and taken from the battery with a higher efficiency, causing that the generator set can work in its most efficient points.

7.1.1 Results for the super-battery

The implementation stays unchanged except for the internal resistance. The results of the three cycles, that also have been examined in Chapter 6, are shown in the figures below.

Figures 7.1(a) to 7.1(c) show that the SOC stays abundantly within its boundaries (which are the same as for the normal battery).

Figures 7.2(a) to 7.2(c) show the numerical results for the simulations. The first thing to notice is that the small slope has disappeared. This means only 3 of the 4 operating points are left. This is expected since the efficiency of the accumulator has increased. Therefore, the middle operating point (20 kW) is apparently not necessary anymore. The other operating points are the same as already obtained. Table 7.1 presents the energy balance for the three different cycles. More energy is charged into and discharged from the super-battery at a higher efficiency causing a decrease in the total energy production of the generator set leading to a decrease in fuel consumption.
Figure 7.1: SOC results for the simulations with super-battery for different drive-cycles

<table>
<thead>
<tr>
<th></th>
<th>MVEG</th>
<th>USFTP72</th>
<th>Modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{drive}$</td>
<td>5.59 MJ</td>
<td>6.03 MJ</td>
<td>16.05 MJ</td>
</tr>
<tr>
<td>regen</td>
<td>-1.21 MJ</td>
<td>-1.81 MJ</td>
<td>-3.49 MJ</td>
</tr>
<tr>
<td>demand</td>
<td>6.80 MJ</td>
<td>7.84 MJ</td>
<td>19.54 MJ</td>
</tr>
<tr>
<td>$E_{genset}$</td>
<td>5.61 MJ</td>
<td>6.05 MJ</td>
<td>16.1 MJ</td>
</tr>
<tr>
<td>$E_{accumulator}$</td>
<td>0.02 MJ</td>
<td>0.02 MJ</td>
<td>0.05 MJ</td>
</tr>
<tr>
<td>charge</td>
<td>2.63 MJ</td>
<td>3.01 MJ</td>
<td>5.12 MJ</td>
</tr>
<tr>
<td>discharge</td>
<td>2.61 MJ</td>
<td>2.99 MJ</td>
<td>5.07 MJ</td>
</tr>
<tr>
<td>Eff. of superbatt ($\eta_{superbatt}$)</td>
<td>99.2 %</td>
<td>99.3 %</td>
<td>99.0 %</td>
</tr>
</tbody>
</table>

Table 7.1: Energy balance DP optimization with super-battery (3 different cycles)
Figure 7.2: Graphical results for the simulations with super-battery for different drive-cycles
The reduction in the fuel consumption can not be compared to a model with structure control like has been done for the battery, simply because a model with super-battery is not available. However, it can be compared to the dynamic programming results of the normal battery. This way it gives an impression of the reduction in fuel consumption a super-battery can bring comparing to a normal battery. Because of the more efficient super-battery, a considerable reduction has been obtained. The fuel consumption reduction is listed in Table 7.2.

<table>
<thead>
<tr>
<th></th>
<th>MVEG</th>
<th>USFTP72</th>
<th>Modem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FC [gr]</td>
<td>FC [l/100km]</td>
<td>FC [gr]</td>
</tr>
<tr>
<td>Opt. CS parameters</td>
<td>387.7</td>
<td>4.32</td>
<td>432.7</td>
</tr>
<tr>
<td>Dyn. Programming (batt)</td>
<td>377.8</td>
<td>4.17</td>
<td>408</td>
</tr>
<tr>
<td>Dyn. Program. (superbatt)</td>
<td>369</td>
<td>~4.07</td>
<td>398.6</td>
</tr>
</tbody>
</table>

Table 7.2: Fuel reduction table using a super-battery

Again small remark must be made over here about comparing the 2 different accumulator types. Though the step-size in SOC for the super-battery is still fixed and $P_{drive}$ is still adapted to $P_{accumulator}$, a small difference in total drive energy ($E_{drive}$) is present due to the fact that the internal resistance of the super-battery is different from the internal resistance of the normal battery. Due to this difference, a different vector $P_{accumulator}$ will be calculated and so $P_{drive}$ will be slightly different. In Table 7.3 the differences within $E_{drive}$ for the different drive cycles are listed. As one can see, the difference is far from large but to be able to completely compare the two different accumulator types, the fuel consumption will again be given in a normalized form [gr/kJ] (Table 7.4).

<table>
<thead>
<tr>
<th></th>
<th>MVEG</th>
<th>USFTP72</th>
<th>MODEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_{drives}$ battery [MJ]</td>
<td>5.602</td>
<td>6.034</td>
<td>16.049</td>
</tr>
<tr>
<td>$E_{drives}$ super-battery [MJ]</td>
<td>5.587</td>
<td>6.031</td>
<td>16.049</td>
</tr>
<tr>
<td>Difference [MJ]</td>
<td>0.015</td>
<td>0.003</td>
<td>5e-5</td>
</tr>
<tr>
<td>Difference [%]</td>
<td>0.26</td>
<td>0.05</td>
<td>~0</td>
</tr>
</tbody>
</table>

Table 7.3: Difference in $E_{drive}$ for different accumulator types

These results show a move in the right direction of fuel reduction while using other (better) accumulators. It is expected that the use of super-capacitors will also show a reduction in fuel consumption regarding the normal battery.
### Table 7.4: Reduction of normalized fuel consumptions

<table>
<thead>
<tr>
<th></th>
<th>MVEG</th>
<th>USFTP72</th>
<th>MODEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel Consumption battery [gr/kJ]</td>
<td>0.0674</td>
<td>0.0676</td>
<td>0.0674</td>
</tr>
<tr>
<td>Fuel Consumption super-battery [gr/kJ]</td>
<td>0.0660</td>
<td>0.0661</td>
<td>0.0660</td>
</tr>
<tr>
<td>Reduction [%]</td>
<td>2.12</td>
<td>2.26</td>
<td>2.12</td>
</tr>
</tbody>
</table>

#### 7.2 Super-capacitor model

The model of a super-capacitor looks very similar to the one of the battery (see Figure 7.3). The big difference between the two is the way the power will be calculated, the power that the accumulator can give or absorb.

![Figure 7.3: Model of the super-capacitor](image)

The open-circuit voltage difference can be written as:

\[
\frac{du_{oc}}{dt} = \frac{1}{C} \cdot i(t) \tag{7.3}
\]

With current \(i (\frac{dq}{dt})\) and \(C\) the capacity of the super-capacitor (3500 F).

The SOC of the supercapacitor is quadratic equal to the open-circuit voltage as showed by equation 7.4:

\[
SOC = \left(\frac{u_{oc}}{u_{max}}\right)^2 \cdot 100\% \tag{7.4}
\]

The voltage is given by:

\[
u_0 = u_{oc} + i \cdot R_{int} \tag{7.5}
\]

and the power of the capacitor is given by:

\[
P_{\text{capa}} = i \cdot u_{oc} + i^2 \cdot R_{int} \tag{7.6}
\]

Since \(i(t)\) is analyzed over a constant \(dt\), it can be stated constant. So now, equation 7.3 can be written as:

\[
i = \frac{\Delta u_{oc}}{\Delta t} \cdot C \tag{7.7}
\]
The full equation for determining the power of the capacitor can now be written as:

\[ P_{\text{capa}} = u_{oc} \cdot \frac{\Delta u_{oc}}{\Delta t} \cdot C + \left( \frac{\Delta u_{oc}}{\Delta t} \cdot C \right)^2 \cdot R_{\text{int}} \]  

(7.8)

with the assumption that \( \Delta t = 1 \).

The main problem here is that, different from the battery, a certain step in SOC (\( \Delta \text{SOC} \Delta t \)) does not explicitly result in a constant value for \( P_{\text{capa}} \). This is a result of the fact that \( P_{\text{capa}} \) is depending on the \( u_{oc} \) and \( u_{oc} \), in turn, is depending on the SOC (see equations 7.3, 7.4 and 7.6). Therefore, a certain step in SOC at a certain \( u_{oc} \) will lead to a different \( P_{\text{capa}} \) as when the same step is taken at a higher or lower \( u_{oc} \).

The result of this SOC dependency is that it is not possible anymore to pre-calculate the state-cost matrix \( U \). This will lead to longer simulation times. Also the trick to the "gap-problem" can not be applied anymore since a constant vector of \( P_{\text{capa}} \) can not be defined anymore. This last remark will be a point of attention later on in this chapter.

As has been seen in equation (7.4), the SOC and the open-circuit voltage are linked to each other. With the calculation of the accumulator power, knowledge about the \( u_{oc} \) is necessary. Therefore, and to avoid an extra calculation step, the open-circuit voltage (\( u_{oc} \)) will now be defined as the control variable. By varying the control variable and by defining the begin and end value of this variable, the optimal path of \( u_{oc} \) (and so of the SOC) leading to the minimum fuel consumption over a given drive-cycle will be obtained while satisfying the constraint \( d\text{SOC} = 0 \).

### 7.3 Initialization

Before simulations can start, some characteristic values for the super-capacitors have to be defined.

First of all, the maximum current the capacitors can handle is \( \pm 350 \) A. This leads to a maximum change in open circuit voltage of:

\[ \frac{\Delta u_{oc}}{\Delta t} = \frac{i}{C} = \frac{350}{22.4} = 15 \text{ V/s} \]  

(7.9)

Under the assumption that \( \Delta t = 1 \) [s] the maximum change in \( u_{oc} \) is 15 V in 1 second.

The upper and lower boundary of the control variable have to be defined as well. It has been chosen to work between the 240 V (38% SOC) and 390 V (100% SOC).

Recalling the trick that has been performed with the battery optimization, adapting \( P_{\text{drive}} \) to \( P_{\text{batt}} \) so \( P_{\text{genset}} \) can become zero ("closing the gap"), it can now be concluded that this trick can not be performed when using a super-capacitor. Since \( P_{\text{capa}} \) is depending on \( u_{oc} \), the values of \( P_{\text{capa}} \) will not be constant values like they have been with the battery. To bridge the gap, the natural behavior of the algorithm will be that \( P_{\text{genset}} \) will be small at some points to make sure the energy balance is correct. This behavior is not desirable since, as mentioned before, these low powers are very inefficient and above all, are not realistic operating points. Therefore, another way of solving this problem is to allow the algorithm to analyze small negative generator set powers. These small negative powers shall close the gap between \( P_{\text{drive}} \) and \( P_{\text{accumulator}} \) and can be interpreted as extra \( P_{\text{drive}} \) at a certain moment (or as an extra total \( E_{\text{drive}} \)). Since negative powers for the genset implies a higher \( P_{\text{accumulator}} \) at the specific moment (\( P_{\text{accumulator}} \geq P_{\text{drive}} \)), the extra fuel consumption has been
taken into account and for the calculations, the fuel consumption corresponding to a negative $P_{\text{genset}}$ can be defined as zero. This way, the total energy balance shall be correct and the problem of closing the gap is "solved" in a decent way.

### 7.4 Results

After adapting the algorithm, the first simulations have been performed. The results of these simulations are that the algorithm has not been able to return to the initial value of SOC which has been defined as 60%. With hindsight, the solution to this problem is quite logical.

The super-capacitors are able to store a lot more energy in a much shorter time. During the last regenerative braking action (from highway speed to zero speed), a lot of energy is produced and stored by the super-cap, resulting in a large increase of the open-circuit voltage. This is a direct cause of the introduction of the lower boundary on the $u_{oc}$. This boundary appears to be "too high" when the present initial condition is used. The amount of regenerated energy is so much, it can increase the $u_{oc}$ of the super-capacitor from the lower $u_{oc}$ boundary until far above the initial condition of the $u_{oc}$. After that, there is no possibility anymore for the accumulator to lose its energy to reach the initial $u_{oc}$ again. Since it is desired to maintain this lower boundary, the solution is clear; in stead of starting at an initial condition of 60% SOC, the initial condition should be a lot higher. So it has been raised up to 90%.

The results of the simulations are shown in the figures below (7.4 and 7.5).

![SOC level for MVEG-cycle, super-capacitor, initial condition of 90%](image1)

![UOC level for MVEG-cycle, super-capacitor, initial condition of 90%](image2)

**Figure 7.4:** SOC and $u_{oc}$ levels for MVEG with super-capacitor

As can be seen in Figure 7.5, at constant $P_{\text{drive}}$ and so constant speed, the generator set is constantly switching from a small negative value to one of the beneficial operating points, from now on called *jumping*. This phenomenon is not desirable. Figure 7.6(a) shows a magnification of Figure 7.5 where the jumping behavior is clear to see. Figure 7.6(b) shows that this behavior does not lead to a clear and useful solution.
The main reason for this behavior is hard to explain but it is a result of the step-size that has been taken to form the grid. This can be explained as follows. For a constant $P_{\text{drive}}$ the generator set is jumping from one operating point to another because the power the generator set has to generate approaches and crosses zero. At that moment, a very inefficient operating point is reached and the algorithm will jump to a more efficient operating point. The fact that the generator power approaches zero is caused by the fact that $P_{\text{accumulator}}$ is rising a little bit. Since $P_{\text{accumulator}}$ is depending on $u_{oc}$, every time a new power is calculated it is different from the previous one while it is desirable that it should be constant. When $P_{\text{genset}}$ crosses zero and a peak power is applied, the $u_{oc}$ rises again and approximately the same lower power for the generator set can be obtained again.

DP calculated the path of the generator set and the $u_{oc}$ as shown in Figure 7.7.

Knowledge about the super-capacitor tells that the higher the $u_{oc}$ the higher the efficiency at a certain power level. So one would expect that it should be energetic
cheaper to first turn the generator set on for a while and than switch it off causing the $u_{oc}$ to rise first before decreasing. By hand it is possible to show that this path of the generator set is energetic more efficient (less losses) than the path that has been calculated by DP. By analyzing a short time-interval in which the $P_{drive}$ is constant, all the generator set powers that DP has calculated in that time-interval shall be collected. By sorting these powers in descending order, the "expected" behavior of the generator set is obtained (Figure 7.8(b)) while the total energy the generator set generates $E_{gen\_set}$ is the same.

When the corresponding behavior of the $u_{oc}$ is calculated it is clear that this path is more efficient. Since, a priori, it is known that this is a more efficient path the end
value of the $u_{oc}$ should be larger than the end value of the $u_{oc}$ in the other (first) path, meaning that with the same amount of generator set energy and drive energy, energy is taken from the accumulator at a higher efficiency (less losses). For a specific time-interval that has been used over here, the difference is only 0.065 Volt. This is a marginal difference but since it is for certain that DP always takes the most optimal path (principle of optimality), one can state that the step-size in the grid is causing the fact that the algorithm takes this path and not the other one, simply because the values of the other path do not lie on the grid but somewhere in between.

To prove this hypothesis about the step-size in the grid, a simulation for this specific time-interval has been performed using a much smaller step-size (12.5 times smaller). Only this specific time interval has been taken because of two main reasons. First of all, it is already known what the behavior is of the accumulator and generator set in this interval. Secondly, decreasing the step-size by $n$ times will increase the simulation time by $\sim n^2$ times.

The results of these simulations are presented in Figure 7.9.

![Graph showing powers and $u_{oc}$](image)

Figure 7.9: Powers and $u_{oc}$ for simulations with an extra small step over a short time-interval

As can be seen from these figures, the results are not yet as good as the theoretical ones but the fact that the $u_{oc}$ first increases (charging the supercap) before it decreases (discharging the supercap) is certainly a step in the right direction and proves that a smaller step-size can find a solution that is energetic more efficient ($u_{oc}$ reaches higher values) than the one that has been obtained the first time. The fact that in the first part of this solution the genset is still showing jumping behavior is caused by the fact that the step-size is still too large so it can not find the "desired" path yet. What is the most important here is that the behavior of the $u_{oc}$ is now like the way it is expected to be, energetic more efficient leading to less fuel consumption. Another thing that can be seen in this solution is the $u_{oc}$ dependency of $P_{genset}$.

Since for a supercap the $P_{accumulator}$, in contrary to the battery, does depend on the SOC it is expected and logical that the control strategy shall not only depend on $P_{drive}$ but also on the SOC. Although the genset is still jumping, from the solutions
one can see that for lower $u_{oc}$ levels, DP shall choose the highest optimal operating point of the generator set. When the $u_{oc}$ is rising in time, DP shall choose lower optimal operating points since the efficiency of the accumulator becomes higher proving the statement described above.

The goal of the optimization has been a reduction in the fuel consumption. The results shown in Figures 7.4(a) and 7.5 correspond to a fuel consumption of 371.4 grams over the entire MVEG cycle. At first sight this does not look like a reduction but remembering that the allowed negative generator set powers are considered as an extra $E_{drive}$ the normalized fuel consumption becomes 0.0658 gr/kJ.

<table>
<thead>
<tr>
<th></th>
<th>$E_{drive}$ [MJ]</th>
<th>FC [gr]</th>
<th>FC [gr/kJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>battery</td>
<td>5.602</td>
<td>377.8</td>
<td>0.0674</td>
</tr>
<tr>
<td>super-battery</td>
<td>5.587</td>
<td>369</td>
<td>0.0660</td>
</tr>
<tr>
<td>super-capacitor</td>
<td>5.639</td>
<td>371.4</td>
<td>0.0658</td>
</tr>
</tbody>
</table>

Table 7.5: $E_{drive}$ and fuel consumption for three different accumulators for the MVEG-cycle

Again, just as with the super-battery, a considerable reduction in fuel consumption has been obtained compared to the battery. This reduction has been obtained with an inefficient jumping behavior of the generator set as a result of the step-size in the grid of the DP algorithm that is too large. Looking at the results for the simulations where this fact has been proven, it is clear that the reduction will be larger when a smaller step-size is considered and so an energetic more efficient path is recognized.

<table>
<thead>
<tr>
<th></th>
<th>FC [gr]</th>
<th>FC [gr/kJ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial step-size</td>
<td>12.70</td>
<td>0.0647</td>
</tr>
<tr>
<td>reduced step-size (12.5 ×)</td>
<td>11.43</td>
<td>0.0582</td>
</tr>
<tr>
<td><strong>Reduction [%]</strong></td>
<td><strong>11.1</strong></td>
<td><strong>11.1</strong></td>
</tr>
</tbody>
</table>

Table 7.6: Reduction of normalized fuel consumption for the considered short time-interval where $P_{drive}$ is constant

As can be seen in Table 7.6, the reduction for the considered short time interval is already 11.1% while every condition has been the same except from the step-size. This looks promising for the entire cycle.
7.5 Summary

The dynamic programming algorithm has been extended from a battery to a super-capacitor as an accumulator. As a transition accumulator, a super-battery (a battery with the resistance of the super-capacitor) has been introduced. For this type of battery, the DP-sequence generated very useful results just as for the battery. A significant fuel reduction of about 2% has been obtained and the results of the behavior of the accumulator and the generator set can be used again for the design of a control structure.

Together with the introduction of the super-capacitor some problems have come up. The super-capacitor is a continuous system which can not be transformed in a discrete model like is done for the battery. Therefor, the problem of "closing the gap" could not be solved in the way like is done for the (super-) battery. A solution for this problem is proposed in the form of allowing the generator set powers to be negative and consider them as extra drive power (increasing $E_{drive}$). The results show an undesirable behavior, here called jumping. The reason for this behavior has been successfully investigated and can be described as:

"Dynamic Programming has the advantage of finding the true optimal for a certain problem within the accuracy of the computational grid".

A solution has been introduced but with this solution, the calculation time rises quadratic and so a full analysis of the in the project used drive-cycles has unfortunately been called off.

Nevertheless, the first results of the simulations with the super-capacitor already show a slight reduction. The results of the simulations where the step-size has been reduced show a larger reduction which looks promising for the overall performance when an entire drive-cycle is analyzed. Just as with the battery and super-battery, the results of dynamic programming will be very useful to derive a control law only now one has to take into account that not only $P_{drive}$ will be important for this control law, also the SOC will be.
Chapter 8

Proposal for a new control strategy

This short chapter will introduce a control structure which can very well be used for the control problem of this project. It is only an introduction so methods and facts will be omitted.

8.1 Driving Pattern Recognition

In Chapter 2, a paper is mentioned in which a multi-mode control strategy is proposed using Driving Pattern Recognition (DPR) techniques [Lin-2002-1]. As mentioned, the DPR controller uses drive-parameters of the current drive pattern to classify them into certain, pre-computed, Representative Driving Patterns (RDP). These pre-determined RDPs are coupled to different control actions that are near-optimal for that driving behavior of that moment.

This kind of control can be very interesting to use for the control problem in this project. In this paper and papers that are related to it ([Jeon-2002], [Lin-2004]), the method of how to determine several RDPs, which contain near-optimal control laws, has been explained.

The results of DP are, as we already know, the optimal solution to a certain problem but are not useful for online implementation. However, parameters for certain driving conditions can be derived from the results and together with representative driving parameters that can be derived from the different drive-cycles, an approximation of the DP results can be made leading to near-optimal control laws. Near-optimal is explicitly mentioned since concessions have to be made. An example of how RDPs can be classified is on basis of the mean desired drive power ($P_{dr\ mean}$) with a certain standard deviation on it ($P_{dr\ mean\ std}$). A small $P_{dr\ mean}$ with a small $P_{dr\ mean\ std}$ can yield a constant urban drive while a large $P_{dr\ mean}$ with a large $P_{dr\ mean\ std}$ can yield a sub-urban drive with some traffic-jams.

Once these RDPs have been determined a smart algorithm must be utilized which is able to recognize different driving situations on the basis of i.e. the same drive-parameters. This makes it a powerful method for known as well as unknown drive-cycles. A short historical horizon (~ 150 seconds) will be introduced over which the algorithm determines which RDP is as close as possible to the past and current
drive behavior. It is assumed that the near future (until the next determination) will maintain the same behavior. Therefor, the future horizon can not be chosen too large.

The results show that the designed multi-mode driving strategy is able to reduce the fuel-consumption and the overall performance of the vehicle regarding the previous single-mode driving strategy. Also a comparison has been made to the dynamic programming results which are, in turn, better than the multi-mode driving results. This is not strange knowing that concessions have been made in determining the near-optimal control laws.

This control strategy looks very promising for the control problem in this project but will certainly demand a lot of investigation with respect to the classification of the RDPs. The results that have been obtained with the DP-algorithm in this project shall be of great use for this process.
Chapter 9

Conclusions and Recommendations

9.1 Conclusions

The goal of the project has been to develop an optimal control law for a series hybrid powertrain to minimize the fuel consumption over a (given) drive-cycle. To design such an optimal control law, a benchmark has to be set for the control law to aim at. The benchmark, the minimum fuel consumption under certain constraints, has to be obtained from off-line optimization of the system. During the project, several methods have been investigated to create such a benchmark.

First of all, a Sequential Quadratic Programming method has been investigated. This "smart" optimization method is utilized in a Matlab optimization tool called fmincon. The goal has been to minimize the fuel consumption by optimizing the behavior of the generator set while certain constraints has been taken into account. The method is based on specifying an objective function from which gradient information is derived. Due to the lack of a real objective function, and so an unreliable approximation of the gradient and Hessian matrix, and due to the large amount of parameters that have to be optimized (behavior of generator set powers) the method has turned out to be not suitable for this kind of optimization.

Secondly, another method called Simulated Annealing has been investigated. This method has the advantage that it can optimize a vector of parameters and above all, allows sub-solutions which can be worse than a previous obtained solution (which is not the case with SQP). The disadvantages that have come up with this method are bipartite. First of all, the algorithm contains parameters that have to be tuned. The tuning of these parameters proved to be quite difficult. Secondly, a random factor that is present in the algorithm causes that several solutions have been found for the same set of parameters. This is not what is expected from an optimization algorithm. Again, also the size of the problem does not contribute to the working principle of the algorithm.

The last algorithm that has been investigated is Dynamic Programming. Where Simulated Annealing tries to find the optimum by smart chosen steps, dynamic programming analyses every possible solution. Dynamic Programming is based on Bellman’s Principle of Optimality which implies that every (sub-) solution is the optimal one. This powerful method proved to return a reliable benchmark for
different drive-cycles. The goal of the optimization, minimizing the fuel consumption, has been satisfied while taking desired constraints into account.

Dynamic Programming has been utilized for three different drive-cycles.

- MVEG
- USFTP72
- Modem

Looking at the driving characteristics, the first two cycles can be compared to each other. The Modem-cycle has been chosen since it is totally different from the other two. For all analyzed cycles, the results show approximately the same behavior; specific operating points at a certain power demand. From these results an optimal control structure can easily be derived since $P_{\text{genset}}$ is known for every desired $P_{\text{drive}}$. However, this control structure is only valid for that specific cycle. For unknown cycles, an extra constraint has to be put on the SOC so the accumulator will not be depleted. Moreover, the results show a significant reduction regarding the benchmark that has been set so far by the optimized present control structure. The reduction is in the order of 2, 4 and 6% for the MVEG, Modem and USFTP72-cycle respectively.

Besides analyzing different drive-cycles, the algorithm has also been utilized to investigated three different accumulator types:

- a battery
- a super-battery
- a super-capacitor

The super-battery, a battery with the same internal resistance as the super-capacitor, has been introduced as a transition accumulator type from the battery to the super-capacitor. This has been done to investigate the influence of the lower resistance while everything is kept the same. For the battery and the super-battery, DP has produced very useful results. A significant fuel reduction has been obtained and the results for the super-battery can, just like has been done with the battery, be well-used to derive control structures. Table 9.1 shows the results of the reduction in fuel consumption for the three defined drive-cycles and the two types of batteries.

<table>
<thead>
<tr>
<th>Fuel reduction</th>
<th>MVEG</th>
<th>USFTP72</th>
<th>Modem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opt. CS param. → DP battery</td>
<td>2.82 %</td>
<td>6.06 %</td>
<td>4.45 %</td>
</tr>
<tr>
<td>DP battery → DP super-battery</td>
<td>2.12 %</td>
<td>2.26 %</td>
<td>2.12 %</td>
</tr>
</tbody>
</table>

Table 9.1:

Together with the introduction of the super-capacitor, some problems come up. These problems are the result of the SOC dependency of the super-capacitor causing that the accumulator power is strongly depending on the open-circuit voltage now. Together with the chosen accuracy of the computational grid, this SOC-dependency causes that the energy balance can not be solved in a decent way (resulting in inefficient operating points of the generator set). The solution for this is to allow negative generator set powers which can be interpreted as extra $E_{\text{drive}}$ at the end of the cycle. The results of the simulations with the super-capacitor show a so-called jumping behavior. This behavior, fast transitions of the generator set from one operating point to another, is not desirable. It has been proven that this is caused by accuracy of the grid. So it can be stated that DP has found the optimal solution
within the accuracy of the defined computational grid. Just as with the super-battery, the fuel reduction with the use of the super-capacitor, comparing to the fuel consumption with the battery, is $\approx 2\%$. But this is with the undesired behavior of the generator set. The fuel reduction during the proof of this behavior is significant ($\approx 11\%$) comparing to the fuel consumption with the jumping behavior. This looks promising for an entire cycle.

Though the real design of a control law has been reduced to just a proposal for a new control strategy, a universal powerful tool (DP) has been introduced which is able to set benchmarks for different types of components, different drive-cycles or situations and is able to give insight information in the relation of drive and system parameters. The fuel reduction that can be obtained with dynamic programming is significant (order of $2 \sim 6\%$) regarding the present control structure and the results will be very useful in the design-procedure of the proposed optimal control strategy.

### 9.2 Recommendations

First of all, it is desirable to find a way to solve the "time of calculation" problem that will arise by introducing the solution with the super-capacitor problem. This shall not be easy since there will always stay a certain amount of calculations the algorithm has to perform.

The main recommendation for a next project is to investigate the proposed control structure. The main issue here is to find out which parameters are necessary to define certain driving patterns and how they can be used to do so. With the help of the results of dynamic programming, several control laws can be defined which operate within these driving patterns. Above all, a driving pattern recognition method has to be designed which is able to switch among the different defined control laws.
Bibliography


Appendix A

Results MVEG-cycle

Figure A.1:

(a) 1-200 seconds

(b) 200-400 seconds

Figure A.2:

(a) 400-600 seconds

(b) 600-800 seconds

69
Figure A.3:

(a) 800-1000 seconds

(b) 1000-1180 seconds
Appendix B

Results USFTP72-cycle

(a) 1-200 seconds

(b) 200-400 seconds

Figure B.1:

(a) 400-600 seconds

(b) 600-800 seconds

Figure B.2:
Figure B.3:

(a) 800-1000 seconds

(b) 1000-1200 seconds

Figure B.4:

(a) 1200-1372 seconds
Appendix C

Results Modem-cycle

(a) 1-200 seconds  
(b) 200-400 seconds

Figure C.1:

(a) 400-600 seconds  
(b) 600-800 seconds

Figure C.2:
APPENDIX C. RESULTS MODEM-CYCLE

Figure C.3:

(a) 800-1000 seconds
(b) 1000-1200 seconds

Figure C.4:

(a) 1200-1400 seconds
(b) 1400-1600 seconds

Figure C.5:

(a) 1600-1877 seconds
Appendix D

Graphical Results of DP

Figure D.1: Graphical results of DP for MVEG-cycle with battery
Figure D.2: Graphical results of DP for USFTP72-cycle with battery

Figure D.3: Graphical results of DP for MODEM-cycle with battery
Figure D.4: Graphical results of DP for MVEG-cycle with super-battery

Figure D.5: Graphical results of DP for USFTP72-cycle with super-battery
APPENDIX D. GRAPHICAL RESULTS OF DP

Figure D.6: Graphical results of DP for MODEM-cycle with super-battery

Figure D.7: Graphical results of DP for MVEG-cycle with super-capacitor